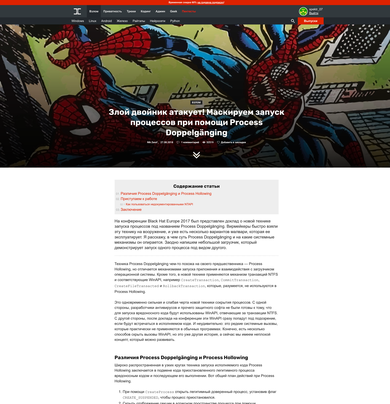

Злой двойник атакует! Маскируем запуск процессов при помощи Process Doppelgänging

Mr.Sun | 27.09.2018 | 1 комментарий | 10231 | Добавить в избранное

Содержание статьи

- Различия Process Doppelgänging и Process Hollowing
- Приступаем к работе
- Как пользоваться недокументированными NTAPI
- Заключение

На конференции Black Hat Europe 2017 был представлен доклад о новой технике запуска процессов под маскировкой Process Doppelgänging. Временным быстро вылез из-за кустов Windows 10 и сказал, что в ядре ОС уже есть подходящие механизмы для мистификации. В рассказе о чем есть Process Doppelgänging и на каких системных механизмах он опирается. Эксперт напишет небольшой загрузчик, который демонстрирует запуск своего процесса под видом другого.

Техника Process Doppelgänging — очень похожа на нашего знакомого Process Hollowing, но отличается механизмом запуска приложения и взаимодействие с загрузчиком операционной системы. Кроме того, в новой технике применяются механизмы транзакций NTFS и скомпилированные NTAPI, которые в Process Hollowing не применялись.

Process Hollowing:

1. При помощи `CreateProcess` открыть, остановить доверенный процесс, установив флаг `CREATE_SUSPENDED`, чтобы запущен приостановился
2. Сдрать отображение секции в адресном пространстве процесса при помощи `UnmapViewOfSection`.
3. Перезаписать код нужным при помощи `WriteProcessMemory`.
4. Запустить при помощи `ResumeThread`.

По сути, мы выучили механизм работу загружаемых операционной системы и делаем за него часть работы, получив подмешив код в память.

В свою очередь, для реализации техники Process Doppelgänging нам нужно выполнить такие шаги:

1. Создание новых транзакции NTFS при помощи функции `CreateTransaction`
2. В контексте транзакции создаем временный файл для нашего кода функцией `CreateFileInTransacted`
3. Создаем в памяти буферы для временного файла (обобщает секцию, функции `NtCreateSection`)
4. Проверяем PE.
5. Запускаем процесс через `NtCreateProcessEx`+`NtCreateThread`.

Вообще, технологии транзакций NTFS(TxF) появились в Windows Vista на уровне драйверов ядра. Они позволяют приложениям выполнять последовательность операций с файлами. Эта технология позволяет поднять прикладную управляемые операциями с файловой системой NTFS. Также она всегда используется при работе с базами данных.

Операция TxF считается атомарным — все изменения файлов работает с транзакцией (и связанными с ней файлами), до не закрытия или отмена она не видна никому. И если будет отказ, то операция не изменит ничего на системе диска. Транзакции можно создать при помощи функции `CreateTransaction` с нулевыми параметрами, а последний параметр — название транзакции. Прототип выглядит таким образом:

```
HANDLE CreateTransaction(
  [in LPGUID lpUOW,                       lpTransactionIdentifier OPTIONAL,
  [in DWORD dwOptions                     OPTIONAL,
  [in DWORD dwIsolationLevel              OPTIONAL,
  [in DWORD dwIsolationFlags              OPTIONAL,
  [in DWORD dwTimeout                     OPTIONAL,
  [in LPWSTR lpDescription                OPTIONAL,
  [in PVOID                               lpReserved
  [in LPWSTR                              Description
);
```

Приступаем к работе

Начнем писать прототипы загрузчика — сначала импортируем из нашего приложения (netloaderdll), которое будет выполнять роль загрузчика. Для этого прописываем Import. Будет предварительно в качестве второго аргумента, а шеть — в качестве обычного.

Как пользоваться недокументированными функциями NTAPI

В ядре мы будем использовать недокументированные функции NTAPI Windows. Они потрясатели ранжированы по своему прототипу. Вот один из возможных методов получения недокументированных функций и работы с ними:

Объявляем прототип функции `NtQueryFileInformationFromHandle`:

```
typedef NTSTATUS(WINAPI *NtQueryFileInformationFromHandle(
  [in HANDLE                 FileHandle,
  [out PVOID                 FileInfoBuffer,
  [in ULONG                  FileInfoLength,
  [in FILE_INFORMATION_CLASS FileInformationClass
);

typedef NTSTATUS(WINAPI *NtQueryFileInformationFromHandle(
  [in HANDLE                 FileHandle,
  [out PVOID                 FileInfoBuffer,
  [in ULONG                  FileInfoLength,
  [in FILE_INFORMATION_CLASS FileInformationClass
);
```

Здесь используем функцию `NtQueryFileInformationFromHandle` объявляем объявляем, только через между переменную:

```
NTSTATUS Status = pNtQueryFileInformation(...);
if (Status == 0x00000000) return 0;
```

Так получаются и используется все наши необходимые недокументированные функции, которые обычно вызывают в Нашем проекте.

```
int main(int argc, char *argv[]) {
  BYTEA data=(BYTE_PAT*) = { 0 };
  HANDLE hTrans = CreateTransaction(NULL,
  0,
  0,
  0
```