Статья Что вам нужно знать о Process Ghosting, новой атаке с подделкой исполняемого файла

xss.is/threads/53019

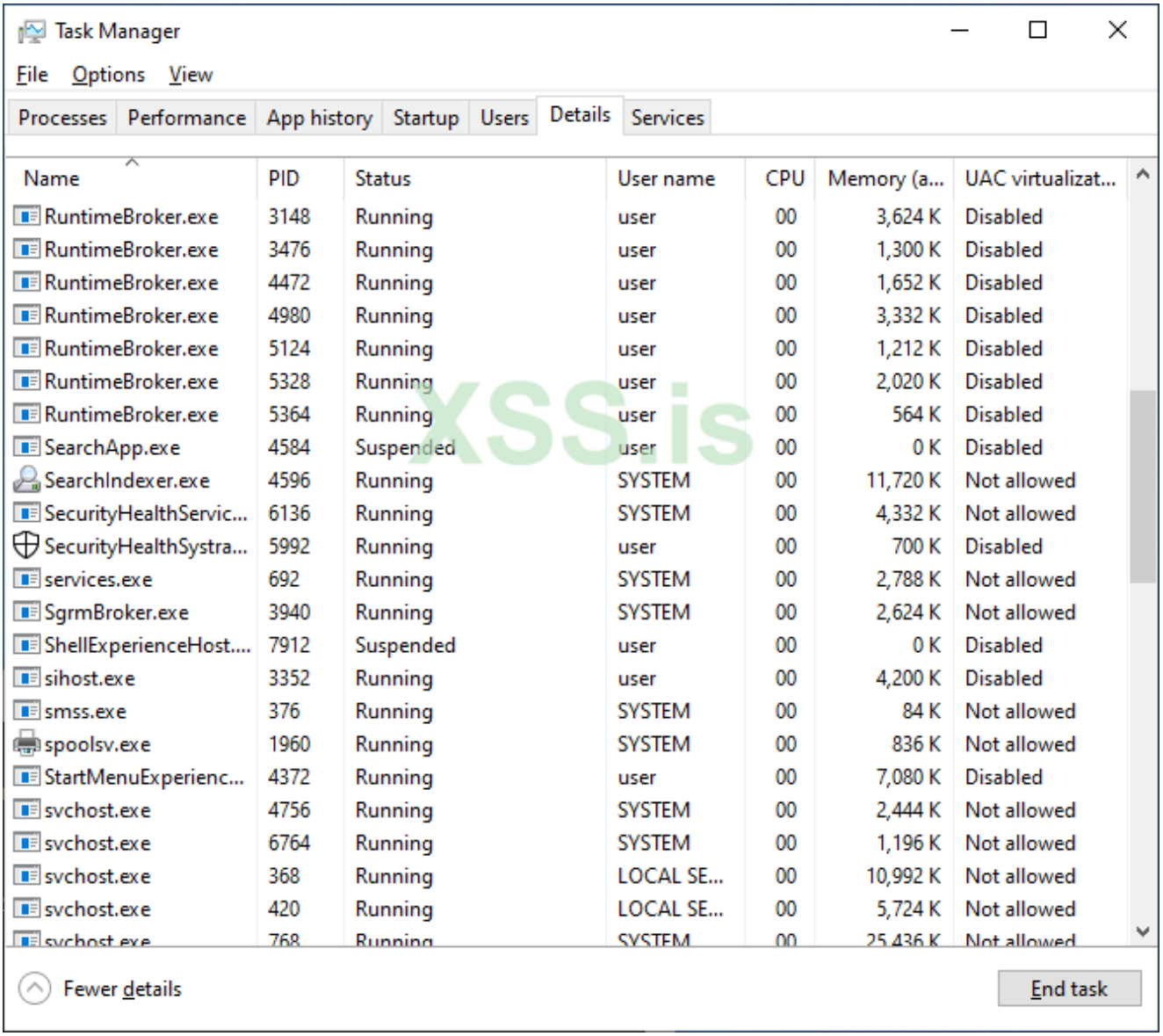
Автор: **Gabriel Landau**

Команды безопасности, защищающие среды Windows, часто полагаются на продукты для защиты от вредоносных программ как на первую линию защиты от вредоносных исполняемых файлов. Microsoft предоставляет поставщикам средств безопасности возможность регистрировать обратные вызовы, которые будут вызываться при создании процессов в системе. Разработчики драйверов могут вызывать API-интерфейсы, такие как PsSetCreateProcessNotifyRoutineEx, для получения таких событий.

Несмотря на название, обратные вызовы PsSetCreateProcessNotifyRoutineEx на самом деле вызываются не при создании процессов, а при создании первых потоков в этих процессах. Это создает разрыв между моментом создания процесса и уведомлением продуктов безопасности о его создании. Это также дает авторам вредоносных программ окно для вмешательства в резервный исполняемый файл, прежде чем продукты безопасности смогут его просканировать. Недавние примеры таких атак с подделкой включают Process Doppelgänging и Process Herpaderping, которые злоупотребляют этим поведением для обхода продуктов безопасности.

В этом посте описывается новая атака подделки исполняемого изображения, аналогичная Doppelgänging и Herpaderping, но отличная от них. С помощью этого метода злоумышленник может записать вредоносное ПО на диск таким образом, чтобы его было сложно сканировать или удалить, и где он затем запускает удаленное вредоносное ПО, как если бы это был обычный файл на диске. Этот метод не включает внедрение кода, пустоту процесса или транзакционную NTFS (TxF).

Рождение процесса
Диспетчер задач Windows показывает список процессов, запущенных в системе. Каждый из этих процессов связан с исполняемым файлом на диске, например svchost.exe. Это связано с тем, что Windows запускает процессы из исполняемых файлов, обычно заканчивающихся расширением EXE.

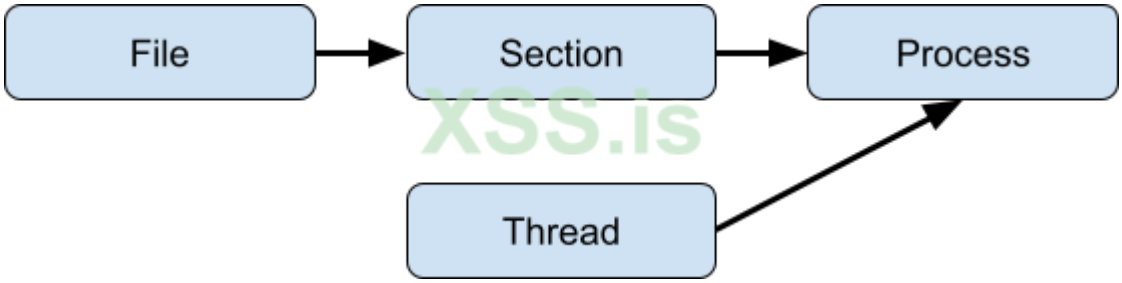


Важно отметить, что процессы не являются исполняемыми файлами, а исполняемые файлы не являются процессами. В приведенном выше примере диспетчера задач несколько процессов запускаются из RuntimeBroker.exe и svchost.exe.

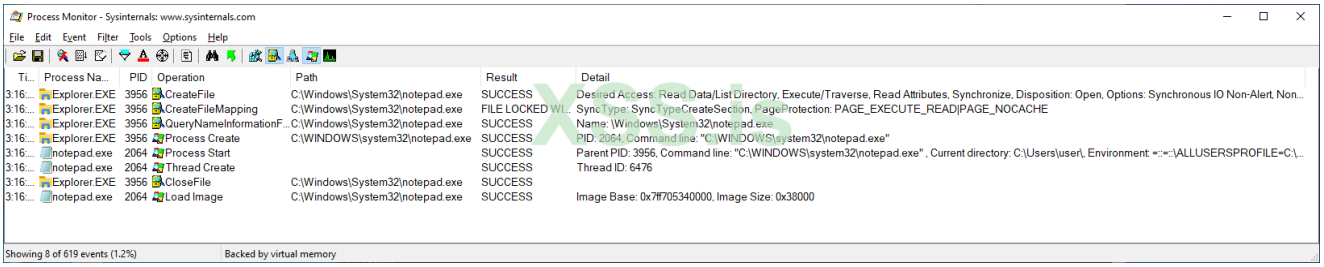
Чтобы запустить новый процесс, необходимо выполнить ряд шагов. В современных версиях Windows они обычно выполняются в ядре NtCreateUserProcess, однако API отдельных компонентов (NtCreateProcessEx и т. Д.) По-прежнему доступны и работают в целях обратной совместимости. Вот эти шаги:

1. Откройте дескриптор исполняемого файла. Пример: hFile = CreateFile («C: \ Windows \ System32 \ svchost.exe»)
2. Создайте для файла раздел «image». Раздел отображает файл или часть файла в память. Раздел изображения - это особый тип раздела, который соответствует файлам Portable Executable (PE) и может быть создан только из файлов PE (EXE, DLL и т. Д.). Пример: hSection = NtCreateSection (hFile, SEC_IMAGE)

3. Создайте процесс, используя раздел image. Пример: hProcess = NtCreateProcessEx (hSection)
4. Назначьте аргументы процесса и переменные среды. Пример: CreateEnvironmentBlock / NtWriteVirtualMemory
5. Создайте поток для выполнения в процессе. Пример: NtCreateThreadEx



Вот как это выглядит в Process Monitor:



Процессы запускаются из исполняемых файлов, но некоторые данные в исполняемом файле изменяются, поскольку они отображаются в процессе. Чтобы учесть эти изменения, диспетчер памяти Windows кэширует разделы image во время их создания. **Это означает, что разделы image могут отличаться от исполняемых файлов.**

Процессы сканирования на наличие вредоносных программ

Microsoft предоставляет поставщикам средств безопасности возможность регистрировать обратные вызовы, которые будут вызываться при создании процессов и потоков в системе. Разработчики драйверов могут вызывать API-интерфейсы, такие как PsSetCreateProcessNotifyRoutineEx и PsSetCreateThreadNotifyRoutineEx, для получения таких событий.

Несмотря на название, обратные вызовы PsSetCreateProcessNotifyRoutineEx на самом деле вызываются не при создании процессов, а при создании первых потоков в этих процессах. Это создает разрыв между моментом создания процесса и уведомлением продуктов безопасности о его создании. Это также дает авторам вредоносных программ окно для вмешательства в резервный файл и раздел, прежде чем продукты безопасности смогут их просканировать.

Обратите внимание, как недокументированное API создания процесса NtCreateProcess принимает дескриптор раздела, а не файла:
C++:

```
NTSYSCALLAPI
NTSTATUS
NTAPI
NtCreateProcess(
    _Out_ PHANDLE ProcessHandle,
    _In_ ACCESS_MASK DesiredAccess,
    _In_opt_ POBJECT_ATTRIBUTES ObjectAttributes,
    _In_ HANDLE ParentProcess,
    _In_ BOOLEAN InheritObjectTable,
    _In_opt_ HANDLE SectionHandle,
    _In_opt_ HANDLE DebugPort,
    _In_opt_ HANDLE ExceptionPort
    );
```

Когда процесс запускается, продуктам безопасности предоставляется следующая информация о запускаемом процессе:
C++:

```
typedef struct _PS_CREATE_NOTIFY_INFO {
  SIZE_T              Size;
  union {
    ULONG Flags;
    struct {
      ULONG FileOpenNameAvailable : 1;
      ULONG IsSubsystemProcess : 1;
      ULONG Reserved : 30;
    };
  };
  HANDLE              ParentProcessId;
  CLIENT_ID           CreatingThreadId;
  struct _FILE_OBJECT *FileObject;
  PCUNICODE_STRING    ImageFileName;
  PCUNICODE_STRING    CommandLine;
  NTSTATUS            CreationStatus;
} PS_CREATE_NOTIFY_INFO, *PPS_CREATE_NOTIFY_INFO;
```

Интересен FILE_OBJECT, который является объектом ядра, соответствующим HANDLE, переданному в NtCreateSection в предыдущем разделе. Этот FILE_OBJECT обычно соответствует файлу на диске, который можно сканировать на наличие вредоносных программ.

Продукты безопасности могут также использовать обратные вызовы минифильтра файловой системы, которые получают уведомления, когда файлы создаются, взаимодействуют с ними или закрываются. Влияние на систему сканирования каждой отдельной операции чтения и записи может быть значительным, поэтому файлы обычно сканируются при открытии и закрытии из соображений производительности.

Существуют и другие потенциальные точки перехвата продуктов безопасности, которые мы не будем здесь обсуждать. см. Этот доклад для получения дополнительной информации.

Преведущие техники

Process Doppelgenging

Windows Transactional NTFS (TxF) - это механизм, который позволяет приложению выполнять серию операций с файловой системой как одну атомарную транзакцию, которая затем либо фиксируется, либо откатывается. Файлы могут существовать в транзакции, которая в случае отката никогда не будет видна базовой файловой системе. Используя TxF, можно создать раздел изображения из файла внутри транзакции, а затем откатить эту транзакцию. Из таких участков изображения можно создать процесс

Process Herpaderping

После создания раздела изображения Process Herpaderping использует существующий дескриптор файла для перезаписи исполняемого файла ложным PE. Хотя при этом ловушка остается на диске, она отличается от той, которая работает в памяти. Приманка остается на диске на протяжении всего процесса полезной нагрузки.

Process Reimaging

Process Reimaging использует проблему синхронизации кеша в ядре Windows, вызывая несоответствие между путем к исполняемому файлу и путем, указанным для разделов изображения, созданных из этого исполняемого файла. Загружая DLL по ложному пути, выгружая ее, а затем загружая с нового пути, различные API Windows возвращают старый путь. Это может обмануть продукты безопасности и заставить их искать загруженные изображения по неправильному пути.

Ghosting a process

Мы можем использовать Doppelgänging и Herpaderping для запуска уже удаленных исполняемых файлов. Есть несколько способов удалить файл в Windows, в том числе:

- Создайте новый файл поверх старого с установленными флагами FILE_SUPERSEDE или CREATE_ALWAYS.
- Установите флаги FILE_DELETE_ON_CLOSE или FILE_FLAG_DELETE_ON_CLOSE при создании или открытии файла.

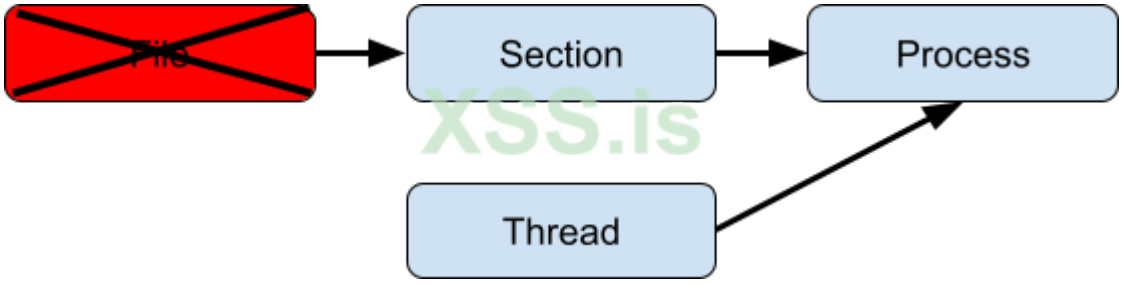
- Установите поле DeleteFile в структуре FILE_DISPOSITION_INFORMATION значение TRUE при вызове класса информации файла FileDispositionInformation через NtSetInformationFile.

Windows пытается предотвратить изменение сопоставленных исполняемых файлов. После того, как файл сопоставлен с разделом image, попытки открыть его с помощью FILE_WRITE_DATA (чтобы изменить его) завершатся ошибкой с ERROR_SHARING_VIOLATION. Попытки удаления через FILE_DELETE_ON_CLOSE / FILE_FLAG_DELETE_ON_CLOSE завершаются ошибкой: ERROR_SHARING_VIOLATION. NtSetInformationFile (FileDispositionInformation) требует права доступа DELETE. Несмотря на то, что право доступа DELETE предоставляется файлам, сопоставленным с разделами image, NtSetInformationFile (FileDispositionInformation) не работает с STATUS_CANNOT_DELETE. Попытки удаления через FILE_SUPERCEDE / CREATE_ALWAYS завершаются неудачно с ACCESS_DENIED.

Однако важно отметить, что это ограничение на удаление вступает в силу только после того, как исполняемый файл сопоставлен с разделом image. Это означает, что можно создать файл, пометить его для удаления, сопоставить его с разделом image, закрыть дескриптор файла, чтобы завершить удаление, а затем создать процесс из раздела без файлов. Это и есть Process Ghosting.

Порядок атаки:

1. Создать файл
2. Переведите файл в состояние ожидания удаления с помощью NtSetInformationFile (FileDispositionInformation). Примечание: попытка использовать FILE_DELETE_ON_CLOSE вместо этого не приведет к удалению файла.
3. Запишите исполняемый файл полезной нагрузки в файл. Содержание не сохраняется, потому что файл уже ожидает удаления. Состояние ожидания удаления также блокирует попытки открытия файла извне.
4. Создайте раздел image для файла.
5. Закройте дескриптор отложенного удаления, удалив файл.
6. Создайте процесс, используя раздел image.
7. Назначьте аргументы процесса и переменные среды.
8. Создайте поток для выполнения в процессе.



XSS.is

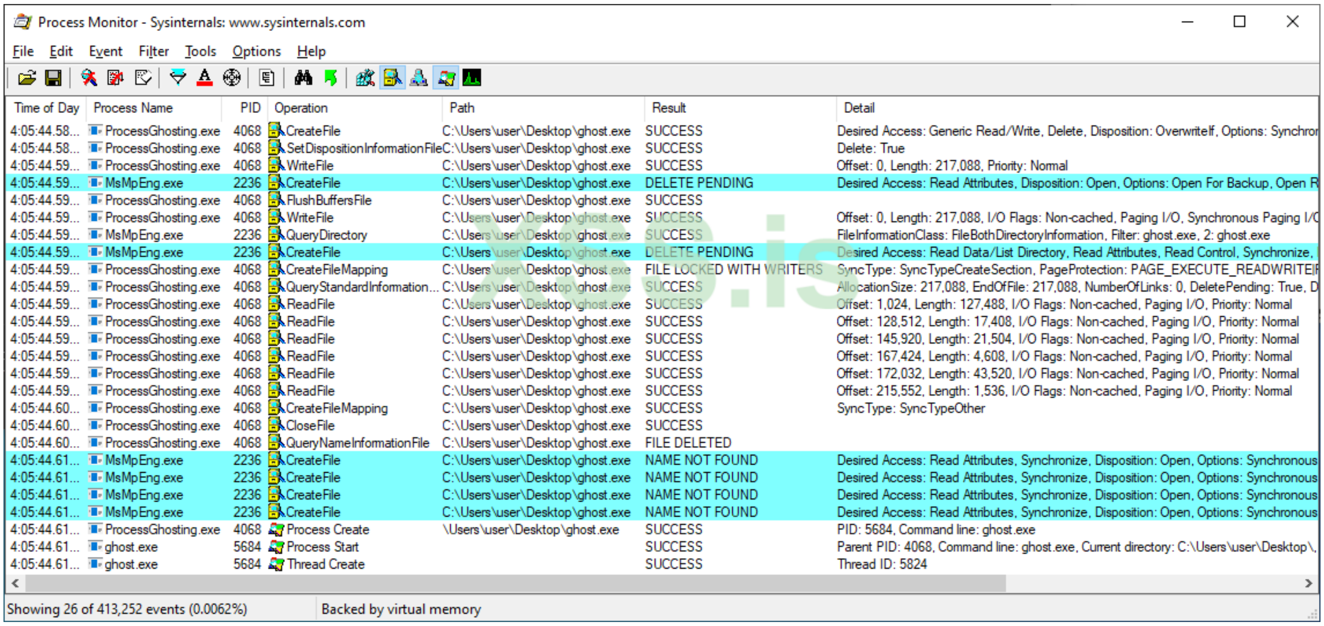
Обратные вызовы антивируса вызываются при создании потока, которое происходит после удаления файла. Попытки открыть файл или выполнить ввод-вывод для удаленного файла завершатся ошибкой с STATUS_FILE_DELETED. Попытки открыть файл до завершения удаления завершатся ошибкой с STATUS_DELETE_PENDING.

Этот тип подделки также может быть применен к библиотекам DLL, поскольку библиотеки DLL также являются разделами image.

Демо
На видео ниже показано, как Защитник Windows обнаруживает и блокирует выполнение Потенциально Нежелательной Программы/Potentially Unwanted Program (PUP), редактора учетных данных Windows, который может использоваться злоумышленниками для бокового перемещения. Затем он показывает, как Ghosting мешает Defender сканировать и блокировать PUP.

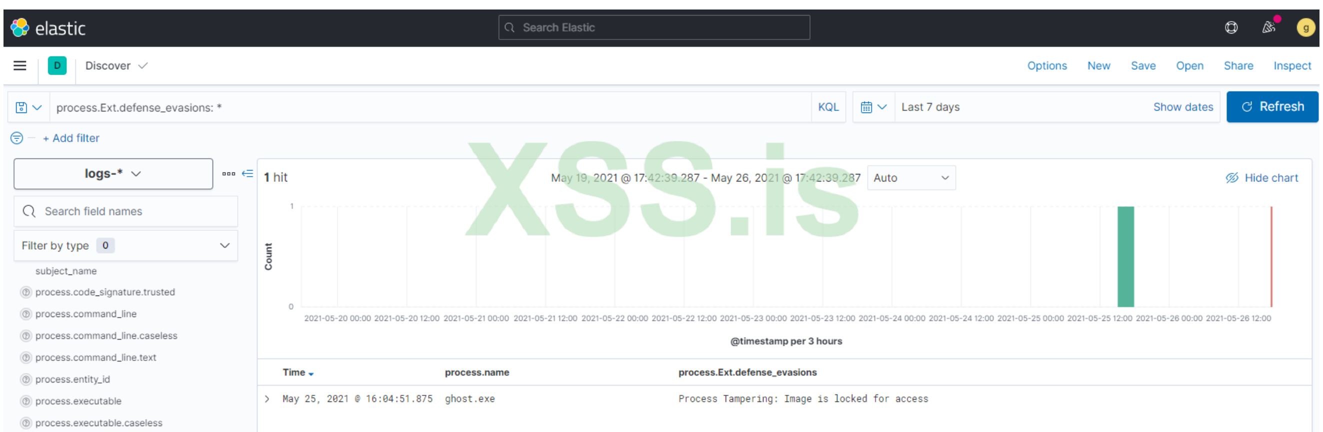
*По поводу видео читайте в разделе **От ТС**

Изучая активность системы во время демонстрации, мы видим, что Защитник сначала пытается открыть исполняемый файл полезной нагрузки для его сканирования, но терпит неудачу, потому что файл находится в состоянии ожидания удаления. Последующие попытки открыть его терпят неудачу, потому что файл уже был удален. Полезная нагрузка (ghost.exe) выполняется без проблем.



Детектирование

Elastic Security обнаруживает различные методы подделки образа процесса, включая Doppelgänging, Herpaderping и Ghosting. Он делает это, проверяя FILE_OBJECT на наличие аномалий во время обратного вызова создания процесса. Об этом сообщается в событиях создания процесса в process.Ext.defense_evasions.



Сравниваем техники

Основываясь на полезной таблице из документации Process Herpaderping, мы можем сравнить базовый поток API при различных методах:

Type	Technique
Hollowing	map → modify section → execute
Doppelgänging	transact → write → map → rollback → execute
Herpaderping	write → map → modify → execute → close
Ghosting	delete pending → write → map → close(delete) → execute

XSS.is

*Таблицу сделал, как обычно - скрином

Заключение
В этом посте мы рассмотрели современное состояние атак с подделкой исполняемых образов Windows, а затем раскрыли новую такую атаку. Затем мы продемонстрировали эту атаку в обход распространенного программного обеспечения безопасности и показали, как ее обнаружить с помощью свободно доступного программного обеспечения.

/* Тут болтовня о Elastic Security, если кому это интересно - Читайте оригинал, он приложен в конце статьи */

Раскрытие: мы подали отчет об ошибке в MSRC 2021-05-06, включая черновик этого сообщения в блоге, демонстрационное видео и исходный код для PoC. Они ответили 10 мая 2021 года, указав, что это не соответствует их планке обслуживания согласно https://aka.ms/windowscriteria.

От ТС
Эта статья является переводом. Оригинал можете найти тут:
https://www.elastic.co/blog/process-ghosting-a-new-executable-image-tampering-attack

Вообще, я начал переводить этот пост в блоге elastic.co из за вот этого:

GitHub - hasherezade/process_ghosting: Process Ghosting - a PE injection technique, similar to Process Doppelgänging, but using a delete-pending file instead of a transacted file

Process Ghosting - a PE injection technique, similar to Process Doppelgänging, but using a delete-pending file instead of a transacted file - GitHub - hasherezade/process_ghosting: Process Ghosting...

github.com

Это имплементация Process Ghosting от hasherezade

Немного внутренней кухни:
В оригинале видео залито через такой сервис как vidyard. Проблема была в том что видео с него не достать(или я просто плохо гуглил).
Я всегда стараюсь по возможности грамотно оформлять переводы, иначе их будет просто не приятно читать. И чтобы не гнать вас в оригинальный пост, я просто записал видео с экрана и залил на Vimeo.

Перевод:
Azrv3l специально для xss.is
Last edited: Jun 16, 2021