DFIR notes

notes on DFIR practice and practice

Blog Files Rules about

Port Proxy detection

How can we see port proxy configurations in DFIR?

I came across a new (to me) technique for evasion and persistence reading news today. The report(1) specifically called out the clever use of built in Windows service control and network utilities (sc, netsh) by some attackers they'd investigated. After Googling for a few minutes it was clear that this technique is known to Windows sysadmins and the attacker community (as it is featured in a Metasploit module(2).

So, let's run the process: create the behaviour in the lab, look for the artifacts, and then figure out how to capture and analyse them.

Make some in the lab

I created a portproxy configuration with netsh on archie, Win10 x64 to send traffic out to another host,port. Tested with netcat chat. Also made one in the seven VM, and the dumped memory variously.

```
netsh>add v4tov4 listenport=3333 connectaddress=192.168.0.8 connectport=8888 listenaddress=0.0.0.0
```

```
PS C:\malware> netsh interface portproxy show all

Listen on ipv4:             Connect to ipv4:

Address         Port        Address         Port
--------------- ----------  --------------- ----------
0.0.0.0         3333        192.168.0.8     8888
```

And it can be seen in running processes, with `netstat -naobp TCP` :

```
TCP    192.168.0.15:3333      192.168.0.15:16047     ESTABLISHED     1572   iphlpsvc
```

Can we find it with Memory Analysis?

Volatile capture with winpmem2, write raw for volatility's use

```
PS> winpmem-2.1.post4.exe -o archie.aff4
PS> rekal.exe imagecopy -f .\archie.aff4 -O archie.raw
```

After imageinfo to get profile(s) , some nosing about to find memory that references the port proxy…

- netsh may not have been running at time of w10 mem capture
- Yarascan for the port number as string: `rekal.exe -f .\archie.aff4 yarascan --string '3333'`
  - many results that dont' look related, probably too broad a key
- ibid for the IP address .. left running for a couple hours
- another try for 'ProxyPort' gets at least one hit:

```
archie> rekal.exe -f .\archie.aff4 yarascan --string 'PortProxy'
2017-02-08 18:50:24,236:WARNING:rekall.1:Unable to parse profile section $CONSTANT_TYPES
2017-02-08 18:50:24,237:WARNING:rekall.1:Unable to parse profile section $CONSTANT_TYPES
Owner    Rule        Offset                                  HexDump                               Symbol
----- ---------- -------------- ------------------------------------------------------------------ ------
-     r1         0x8202038bd448 50 6f 72 74 50 72 6f 78 79 00 00 05 20 00 00 00  PortProxy.......
                                a8 ff ff ff 6e 6b 20 00 02 d9 44 a6 97 3f d2 01  ....nk....D..?..
                                03 00 00 00 28 b1 4d 00 01 00 00 00 00 00 00 00  ....(.M.........
                                10 4b 5e 00 ff ff ff ff 0b 00 00 00 28 45 5e 00  .K^.........(E^.
```

How about dynamic analysis?

Of course we know we can see them in netsh as above. Win10 netsh depreciation notice refers us to Powershell. Some trial and error with thsoe modules did not uncover the portproxy settings.

```
In future versions of Windows, Microsoft might remove the Netsh functionality
for TCP/IP.

Microsoft recommends that you transition to Windows PowerShell if you currently
use netsh to configure and manage TCP/IP.

Type Get-Command -Module NetTCPIP at the Windows PowerShell prompt to view
a list of commands to manage TCP/IP.

Visit http://go.microsoft.com/fwlink/?LinkId=217627 for additional information
about PowerShell commands for TCP/IP.
```

Guessing the storage is Registry, trying RegShot in the live VM shows it clearly:

```
HKLM\SYSTEM\ControlSet001\services\PortProxy\v4tov4\tcp\0.0.0.0/4444: "192.168.0.15/8080"
HKLM\SYSTEM\CurrentControlSet\services\PortProxy\v4tov4\tcp\0.0.0.0/4444: "192.168.0.15/8080"
```

And now we can look with EG printkey on the memory dump(s)…

It's included in output of

```
PS> rekal.exe -f .\archie.aff4 printkey -r -k "\ControlSet001\Services" > archie-rekall-services.txt
```

but it's just a tease:

```
Registry: Unnamed @ 0x82020323e000
Key name: PortProxy (S) @ 0x8202038bd3fc
Last updated: 2017-02-08 15:14:52Z

Subkeys:

Values:
```

and I'm having trouble pinning it down or getting the full details out with Rekall or Vol26. Printkey is always so fiesty…

```
PS C:\malware> C:\tools\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone.exe -f .\seven-memdump.mem --pro
file Win7SP1x86_23418 printkey -K "services"
Volatility Foundation Volatility Framework 2.6
Legend: (S) = Stable   (V) = Volatile

The requested key could not be found in the hive(s) searched
PS C:\malware> C:\tools\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone.exe -f .\seven-memdump.mem --pro
file Win7SP1x86_23418 printkey -K "ControlSet001"
Volatility Foundation Volatility Framework 2.6
Legend: (S) = Stable   (V) = Volatile

----------------------------
Registry: \REGISTRY\MACHINE\SYSTEM
```

```
Key name: ControlSet001 (S)
Last updated: 2013-10-23 16:16:25 UTC+0000

Subkeys:
  (S) Control
  (S) Enum
  (S) Hardware Profiles
  (S) Policies
  (S) services

Values:
PS C:\malware> C:\tools\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone.exe -f .\seven-memdump.mem --pro
file Win7SP1x86_23418 printkey -K "ControlSet001"
```

And despite being in the services folder in the registry keys, it doesn’t show in svcscan -V output as ProxyPort, but instead as “IP Helper” in the NetSvcs.

```
Offset: 0x7bb828
Order: 127
Start: SERVICE_AUTO_START
Process ID: 876
Service Name: iphlpsvc
Display Name: IP Helper
Service Type: SERVICE_WIN32_SHARE_PROCESS
Service State: SERVICE_RUNNING
Binary Path: C:\Windows\system32\svchost.exe -k netsvcs
ServiceDll: %SystemRoot%\System32\iphlpsvc.dll
ImagePath: %SystemRoot%\System32\svchost.exe -k NetSvcs
FailureCommand:
```

These are certainly detectable, and can be collected in triage capture as well as in intensive analysis. It’s persistent into Registry but otherwise a little tricky to see outside of the *netsh* environment.

Refs

1. sc and netsh from: https://securelist.com/blog/research/77403/fileless-attacks-against-enterprise-networks/
2. MSF module: https://github.com/rapid7/metasploit-framework/blob/master/modules/post/windows/manage/portproxy.rb

Written on February 8, 2017