1 Table of Contents

2 Executive Summary

On September 19, 2020, a brute force attack compromised the example network for the Citadel network, originating from an external IP Address of 194.61.24.102. The compromised Domain Controller (DC01), residing at 10.42.85.10, was accessed via a popular exploitation tool called Hydra, under the system's Administrator account. Once inside the DC, the attackers then used internet explorer to install a malicious executable called "coreupdate.exe", allowing the actors to establish an RDP session with the controller with the intent of data exfiltration, manipulation, and reconnaissance. While present on the DC, the attackers were also able to move to a client with the hostname of Desktop-SDN1RPT and perform the same operations that plagued the controller. The actors then migrated the data and closed the sessions.

3 Incident Discovery

3.1 Summary

This evidence was presented in multiple compressed folders in .zip format. Below is a hierarchal structure of the pieces of evidence inside the compressed archives. All files include MD5 checksums to verify the integrity of their content, and all child directories/files that are not in .zip format had to be decompressed during the investigation. While unlikely, decompressing the archives may alter the content of the files if there is an unknown error during the process, so the MD5 checksums should always be verified before this investigation can be repeated. Highlighted files represent the given evidence and the rest indicate discovered evidence.

3.3 Evidence to Analyze

File name	Checksum MD5
case001-pcap.zip	422046b753cf8a4df49d2c4ce892db16
case001.pcap	f81a3ab2cb74dc3c1d91d1bab1b5fc9d
DC01-autorunsc.zip	964f2d710687d170c77c94947da29e66
autorunsc-citadel-dc01.csv	5658e399fdcb3e3219d840e519679537
DC01-E01.zip	e57fc636e833c5f1ab58dface873bbde
E01-DC01	
20200918_0347_CDrive.E01	809fc1976b8593dfcd55a7f71a755be6
20200918_0347_CDrive.E01.txt	eb7a23b719e941c30a8704d0408b0db5
20200918_0347_CDrive.E02	edc258fdf7111f0da6ae353fa4f291a8
DC01-memory.zip	64a4e2cb47138084a5c2878066b2d7b1
citadeldc01.mem	0623f97fc80c12aa508ed9926b2ec04e
DC01-pagefile.zip	964eeaf0009d08cc101de4a83a4e5d23
citadel-dc01-pagefile.sys	1349f69a0338730b1defd23eedb0aadc

DC01-ProtectedFiles.zip	ad29830a583efe49c8c1c35faffd264f
DC01-ProtectedFiles	
Protected	
default	b7a20d3ac609c0542a1e0e7d5e3bf473
ntds.dit	a272d049aa6be7f9a662917da005f65d
SAM	36456b3ccffc110e00c4a7ec5240abb1
SECURITY	919ea108536018ab651e91e2984682e0
software	477c82a44f2f59f3f36afc0f13413cc3
system	05cd86230d5bdbcade8fd6da1d5313a4
Users	
Administrator	
Default	
DESKTOP-E01.zip	71c5c3509331f472abcdf81eb6efff07

4 Forensics Tools

4.1 PCAP/Network Analysis

4.1.1 Wireshark 3.7.0

The most popular tool for performing network traffic captures and analyzing previously captured traffic. It provides a simple GUI with preconfigured options for filtering and analysis, such as identifying all conversations, endpoints, files, applications, and protocols present in the capture. This analysis was run on the MacOS version of the software.

The tool can be found here: www.wireshark.org

4.1.2 Tcpdump 4.99.0

Another popular PCAP analysis tool that uses the command line interface instead of a GUI. This can be helpful when used in tandem with Wireshark if the investigator is looking for a specific format or a more detailed view than what Wireshark can provide. This can also be used alone instead of Wireshark for the same purpose.

The tool can be found here: www.tcpdump.org

4.1.3 Snort 3.0

An open-source intrusion detection/prevention system software that can also be used to analyze the security of the packets travelling throughout the network. Snort uses a series of rules to filter packets for malicious content and notify administrators of any threats detected. This can be used in a live environment or to analyze capture files.

The tool can be found here: www.snort.org

4.2 Memory Analysis

4.2.1 Volatility 2

An open-source memory forensics tool that allows investigators to view and analyze the data that was captured from the victim machine's RAM registers and other volatile data that's deleted once the machine is powered off. Since memory data is usually presented in machine language, the tool translates this data into a human readable format with statistics on the data in the memory capture.

The tool can be found here: www.volatilityfoundation.org

4.2.2 Clamscan

An open-source command line tool included in the Clam Antivirus software package. It is used to scan memory captures for possible malware infection and reports its findings to the investigator.

The tool can be found here: www.clamav.net

4.2.3 FLOSS (FireEye Labs Obfuscated String Solver)

Tool that can automatically extract strings from memory registers and look for any processes that seem out of place. Often, malicious strings in Windows Portable Executables (PEs) are obfuscated to evade detection and appear like a legitimate process. FLOSS can decode these strings during an investigation.

The tool can be found here: www.fireeye.com/services/freeware/floss.html

4.2.4 Strings

Command line tool that can print strings embedded into binary files or executables. This is useful for finding strings that may be corrupted, analyzing strings, debugging, etc. It's able to scan object files and core dumps from the memory capture. It is included on most Linux distribution as part of the kernel image.

Information on the tool can be found here: www.ibm.com/docs/en/aix/7.2?topic=s-strings-command

4.3 Disk Analysis

4.3.1 EWF Mount

A command line utility for investigators to explore files in with the .E01 extension, known as Expert Witness Format (EWF) files. This utility mounts the disk as if it were physical so the investigator can perform analysis on the drive. EWF files are often segmented into separate files to provide the entire picture of the drive.

More information on this tool can be found here: dfir.science/2017/11/EWF-Tools-working-with-Expert-Witness-Files-in-Linux.html

5 Analysis

5.1 Network Forensics

5.1.1 Install and Configure Snort

On a Kali Linux virtual machine, Snort was installed using the command:

```
sudo apt update && sudo apt install snort
```

Once Snort is installed, it needs to be configured to analyze the PCAP file provided. This was done by first defining the home_net variable in the config file that can be found at /etc/snort/snort.conf. According to the preliminary discussion, the network begins with 10.42.x.x. Before the investigator can begin using Snort to analyze the capture, a tcpdump command must be performed, which is listed here:

```
tcpdump -nr case001.pcap 'host 10.42' -c15
```

This command will then display the first 15 packets within the capture that begin with an IP address of 10.42.x.x, while also disabling DNS resolution. According to the results, it can be inferred that the home network address is 10.42.85.0/24. The Snort configuration file can now be updated by finding the home_net variable and changing it to the following:

```
ipvar HOME_NET 10.42.85.0/24
```

At this point during the configuration, it's a good idea to test to make sure the snort config is working properly and that modifying the config file did not break the system. This was done using the following command:

```
snort -c /etc/snort/snort.conf -T -I lo
```

This command tells snort to run from the default config file using the loopback address of the test machine. This proved that snort is indeed working as expected, and the investigation can continue. Snort is useful because it can read the PCAP file and display any alerts that may be present. To generate the alerts and analyze the capture, the following command was issued:

```
snort -c /etc/snort/snort.conf -r case001.pcap -q -K none -A console | tee snort.out
```

Upon completion of this command, snort returned many alerts that were raised from traffic in the capture. Most notably, an NMAP packet was found at the time stamp of 09/19-02:19:13.414319. This indicates that the attacker ran a scan on the network to create a map of possible exploitation avenues. The packet originated from the IP address of 194.61.24.102 and probed the machine at 10.42.85.10 on the internal network. As it turns out, this address is that of the Domain Controller on the network. In terms of network security, the DC is a valuable asset for both the attackers and defenders, as it contains the network configuration, sensitive data, network services, and many more possible points of interest that must be secured properly. If an attacker gains access to the DC, they can take control of the entire network due to the trust that other network clients have with it. Just moments later, the DC is involved with multiple connections to the external attacker's IP address. This should raise a red flag, as the DC should never communicate to the internet directly. Due to the volume of requests in such a short amount of time, the investigator can view the events as a brute force attack. From the DC's side of the requests, the port 3389 is probed consistently, which indicates that the attacker is attempting to establish an RDP connection to the server. Looking at the attacker's side reveals that each request increments the source ports by 2. This confirms that the attacker is trying to brute force into the network by trying every port it can in a matter of seconds.

5.1.2 Using TCPdump

Now that the information from Snort is found, the investigator can get a more detailed picture of the events that occurred during the attack. Using the indicators that were just found, TCPdump will narrow down the results to find pieces of evidence that are relevant to the investigation. This was done by using the following command:

```
tcpdump -ttttnr case001.pcap 'host 194.61.24.102' -c 20
```

This command will display the first 20 packets that include the host of 194.61.24.102 and formats the timestamps to be more detailed for the investigator to sift through. The first items to note in these results are the pings from the external network device and the subsequent replies from the internal server. This also raises a red flag, as domain controllers should not be able to reply to requests from external sources. Inside these ping requests is a SYN flag sent from the external machine with the port of 443, which was responded to with an ACK flag from the internal server on port 80. To prove that these requests are intended to establish connection using RDP (Port 3389), a virtual network was created that's external to the DC and the attacker's methods were recreated to simulate what happened. Using Nmap, the same signature can be found when probing port 3389 on the DC, further supporting that this was the intent.

5.1.3 Identify Conversations

Using WireShark, the PCAP file can be opened, and more details can be found on the conversations that took place. After consolidating the capture to the relevant packets, the endpoints menu shows there are 3 devices interacting in the conversations:

- 194.61.24.102
- 10.42.85.10
- 10.42.85.115

As shown, the capture contains the two devices that were already found. In addition, another host is present on the internal network that is also partaking in conversations with the external address. To confirm that the host is in fact communicating externally, a display filter was used to find the packets this IP address was involved with: ip.addr==10.42.85.115. When that filter is run, TCP packets are returned that show the internal host directly communicated with the IP address without the use of DNS. This is suspicious because internal machines should route through a DNS server, as most people would not access websites by typing in the IP address. The TCP conversation can be followed by right clicking the packet and selecting "Follow" then TCP. Following the conversation reveals that reached out to a simple HTTP server on Python 2.7.18, containing the malicious "coreupdater.exe" executable. This file was downloaded to the internal host in this conversation, the same file that was downloaded on the DC. Wireshark also makes it easy to export the HTTP of the packet to get a hold of this file. Upon extraction, this file was downloaded onto the investigation VM for further analysis. Using VirusTotal, a popular malware analysis site, the hash can be uploaded to determine if this file is indeed malicious. However, upon uploaded the file hash to the website, no results were found. This is indicative of a common technique used among attackers in which a small number of characters are changed inside the file, causing a different hash to be generated with each download. This helps the file evade detection from websites like VirusTotal.

5.1.4 Extracted File Analysis

Examining the file further reveals that the strings in the coreupdater executable are obfuscated in an attempt to make its intent harder to detect. FLOSS is a tool that will make finding these strings easier to find. Running this tool against the executable shows that it contains a payload called “ExitProcess,” which is an indicator of compromise. FLOSS also indicates that there may be injection happening within the executable.

At this point in the investigation, ClamAV was used to scan the file for malware. Unfortunately, this did not return any useful results, likely due to the obfuscation of the strings inside. While analyzing the file, static analysis like this doesn’t give a relevant picture for what it’s trying to do. Therefore, the investigator should use dynamic analysis to sandbox the malware and determine what it’s doing. Any.run is a useful public sandbox for dynamic analysis of the processes. This website can be used to search for the file if anyone has sandboxed it before. Since the analysis was not already present on the site, the file must be uploaded and sandboxed. Unfortunately, this website is only able to sandbox executables using 32-bit architecture. The extracted executable is running on 64-bit architecture, so another public sandbox was used called Joe Sandbox. When uploading the file to Joe Sandbox, a warning appears that tells the investigator that it may tip off the attackers that their malware was found. This may cause them to change strategies to evade detection in the future. After a few minutes for the sandbox to run the malware, the results indicate that the file is malicious. Under the IOC tab of the analysis, the actual IP address the executable tries to reach out to is found, which is 203.78.103.109 located in Thailand. If the company does not do business with this country, it’s another major red flag that a compromise has occurred. Joe Sandbox will also generate a report of everything the malware tried to do on the machine.

5.2 Memory Analysis

5.2.1 Using Volatility 2

Upon receipt of the evidence listed in section 3, the memory file must be decompressed from the memory.zip file. The unzip command was used to extract the memory file. Volatility is a GUI program that will perform an analysis of the captured memory state. This can help paint a bigger picture of the malware’s intentions. The system info page details all system information about the DC the memory was captured from. The file indicates that the DC’s hostname is CITADEL-DC01. The DC is running Windows Server 2012 R2 x64 version 6.3.9600 and contains 2 GB of physical RAM. For further investigation, the vol.py script can be run using the following command:

```
vol.py -f citadeldc01.mem imageinfo|tee imageinfo.out
```

Once the image’s info is detailed, Volatility can be used to perform a netscan as most modern malware contains some sort of network component to reach out to command-and-control servers. This can be done by running the command:

```
vol.py -f citadeldc01.mem --profile=Win2012R2x64 netscan | tee netscan.out
```

When this command is run, the investigator can look for indicators that help narrow down the search to relevant results. In this case, known malicious IP addresses and processes can be found to determine what those processes are doing to the memory. It's also helpful to look for processes that have no reason to regularly connect to the internet, as this would be another indicator of compromise. Looking at the result provides a large volume of information that would take hours to search through. Mainly, dns.exe is facilitating a lot of traffic. To view the list of process information without dns.exe, this command was run:

```
grep -v dns.exe netscan.out|less
```

Upon review of the processes, coreupdater.exe stands out. This does not appear to be a normal process in Windows Server 2012, so it is likely malicious.

Volatility includes a useful tool called "malfind" that will analyze the memory registers and allow the investigator to find any suspicious processes. Once malfind was run against the memory file, a few entries under the PID of 3724 to note are listed here:

```
ADD [EAX], AL

ADD [EAX+EAX], AL

ADD [EAX], AL
```

Further inspection of the malfind tool reveals that it contains the Hex numbers for MZ, which is used for a portable executable (PE). This finding indicates that injected code is present. Malfind contains another useful module called maldump, which will export the carved files into a directory to aid in finding malicious processes. It can be used by inputting the following command:

```
vol.py -f citadeldc01.mem --profile=Win2012R2x64 malfind -D maldump/
```

Once the maldump command is executed, Clamscan can be run to search the directory for infected files. According to the scan, two files are infected with the Win.Exploit.Meterpreter-9752338-0 exploit. This is important, because Meterpreter is a highly powerful exploit that's used by many attackers and can give control to a large number of vital components on the network. The malfind tool reveals that the PID found earlier of 3724 belongs to the Spoolsv.exe process, which is a legitimate process used by Windows. One of the reasons Meterpreter is so powerful is because it has the ability to inject code into legitimate processes to migrate its own processes and evade detection.

5.2.2 Using FLOSS

FLOSS is useful for finding obfuscated strings in the malware. The tool can be run against one of the dmp files found previously instead of both, as they were both injected into the same process. Floss was used with the following command:

```
floss process.Oxffffe000631cb900.0x4afbf20000.dmp | tee 4afb-floss.out
```

At the end of the results, floss tells the investigator that it uses a TCP connection to 203.78.103.109 using port 443. This is the same IP address that was found earlier that resides in Thailand.

5.3 Disk Analysis

5.3.1 Mounting the E01 files

An E01 file is an image in the Expert Witness File (EWF) format that captures the contents of the system disk in a digital forensics investigation. The forensic image is broken up into multiple files to allow for decompression of certain parts of the image instead of decompressing the whole system, which saves space on the forensics lab machine. First, the image needs to be mounted to the investigators machine by navigating to the directory the file is stored in and running the ewfmount command. The image will be mounted to either /mnt/ewf, /mnt/e01, or /mnt/ewf_mount. The mount can then be verified using ll /mnt/ewf. The partition table of the image can be used to map the contents of the disk for further examination with this command:

```
mmls /mnt/ewf/ewf1
```

Once the map of the drive is returned, the Windows partition can be mounted to the system using this command:

```
mount -t ntfs-3g -o
loop,ro,show_sys_files,stream_interface=windows,offset=$((2048*512)) /mnt/ewf/ewf1
/mnt/windows_mount/
```

In the OS partition, the file system can be explored. To mount the OS partition, this command was used:

```
umount /mnt/windows_mount
```

Then, the partition must me mounted again with the partition found using MMLS earlier. The contents of the partition can be listed using the ll command again.

5.3.2 Investigating the File System

The first thing that should be done during this phase of the investigation, is to verify the time zone that the captured machine is set to. This was found by navigating to the system file under /windows/System32/config. This is a System Hive that contains the registry entries for the Windows system and will need to be extracted. Using a Windows forensics machine, the registry can be looked at using Registry Explorer. Once the hive is loaded into Registry Explorer, the logs can be examined. These logs reveal that the machine is on Pacific Standard Time, which is incorrect for the location of the company. The company is located in Colorado, which should use mountain time instead. Using the information gathered from the MMLS actions performed earlier in the investigation, FLS can be used for a more detailed picture of the drive. The boot partition was processed using this command:

```
fls -m -r -o 2048 /mnt/ewf/ewf1 > fls-dc01.body
```

Then the Windows partition must be processed:

```
fls -m -r -o 718848 /mnt/ewf/ewf1 >> fls-dc01.body
```

FLS is helpful because it can be used to create a timeline of the events that occurred on the disk partition. This was done using the following command:

```
vol.py -f citadeldc01.mem --profile=Win2012R2x64 timeliner --output=body --output-
file=/cases/szechuan/dc01/disk/memory-timeliner-dc01.body
```

5.3.3 Analyzing the Timeline

Once the CSV of the registry is created, the investigator can begin analyzing the data. Because this is the last phase of the investigation, the previous information found can be used to narrow down the results and get an accurate idea of the machines affected. Because this timeline has a large volume of data, the results must be narrowed down to vital pivot points that will present the information needed. There are a few main pivot points found already to go off of:

- 203.78.103.109
- 194.61.24.102
- Coreupdater.exe
- PID 3644
- Zip files that may contain sensitive information

Inside the CSV, the coreupdater.ex can be found by searching for the strings coreupdate and coreupdater. Note that in forensics, filenames are often changed during this stage, so coreupdate.exe will not be found.

Upon further examination, a ZIP file can be found insider the CSV that contains sensitive information. Below is a list of files that were exfiltrated:

- Szechuan Sauce.txt
- Secret_Beth.txt
- Beth_Secret.txt
- PortalGunPlans.txt

6 Conclusion

During the investigation into the Citadel's attack, it was found that the DC was targeted by a brute force attack that scanned the machine for any open ports to use with RDP. Data was exfiltrated to an external server. To prevent an attack like this from happening in the future, it's recommended that the company configure the DC with an air gap so that it cannot access the internet and vice versa. On the DC, the RDP protocol should be disabled from outside connections, and an encrypted VPN should be used when RDP is required. It would also be useful to implement a robust firewall that closes any ports not used by the network, and rules should carefully be crafted to allow only the required packets for productivity to access the network. The company should also consider updating its password policy to increase complexity and require the passwords to be updated on a regular basis with no password reuse.