MANDIANT

YOUR CYBERSECURITY ADVANTAGE

Flare-On Challenge 8 Solution

By Ana María Martínez Gómez

Challenge 5: FLARE Linux VM

Challenge Prompt

Because of your superior performance throughout the FLARE-ON 8 Challenge, the FLARE team has invited you to their office to hand you a special prize! Ooh – a special prize from FLARE ? What could it be? You are led by a strong bald man with a strange sense of humor into a very nice conference room with very thick LED dimming glass. As you overhear him mumbling about a party and its shopping list you notice a sleek surveillance camera. The door locks shut!

Excited, you are now waiting in a conference room with an old and odd-looking computer on the table. The door is closed with a digital lock with a full keyboard on it.

Now you realize… The prize was a trap! They love escape rooms and have locked you up in the office to make you test out their latest and greatest escape room technology. The only way out is the door – but it is locked, and it appears you must enter a special code to get out. You notice the glyph for U+2691 on it. You turn your attention to the Linux computer - it seems to have been infected by some sort of malware that has encrypted everything in the documents directory, including any potential clues.
Escape the FLARE Linux VM to get the flag - hopefully it will be enough to find your way out.
Hints:
- You can import "FLARE Linux VM.ovf" with both VMWare and VirtualBox.
- Log in as 'root' using the password 'flare'
- If you use VirtualBox and want to use ssh, you may need to enable port forwarding. The following link explains how to do it: https://nsrc.org/workshops/2014/btnog/raw-attachment/wiki/Track2Agenda/ex-virtualbox-portforward-ssh.htm.

Solution

In this challenge, you are provided with a VM (in OVF format) and an introduction. The introduction mentions some key details:

- The VM has been infected by malware
- The malware has encrypted everything in the documents directory
- Some important information has been hidden in the party shopping list

The provided VM is running openSUSE. This can be suspected by the start screen and the `Have a lot of fun` in the message of the day and confirmed in the `/etc/os-release` file. Note that `openssh` is installed and the `sshd.service` is running. So you can connect to the VM using `ssh` and copy files from/to the VM using `scp`.

```
~ $ ssh root@192.168.192.196
Password:
Last login: Thu Jul  8 14:02:06 2021 from 192.168.192.1
Welcome to the FLARE Linux VM. :)
Have a lot of fun...
localhost:~ #
```

The following tools are used in this write up:

- VMware or VirtualBox
- ssh and scp
- IDA Pro
- CyberChef

Find the malware

The `~/Documents` directory contains several files with the `.broken` extension, which seem to be encrypted:

```
localhost:~/Documents # ls
banana-chips.txt.broken     natillas.txt.broken       spaghetti.txt.broken
blackberries.txt.broken     nutella.txt.broken        strawberries.txt.broken
blue_cheese.txt.broken      oats.txt.broken           tacos.txt.broken
donuts.txt.broken           omelettes.txt.broken      tiramisu.txt.broken
dumplings.txt.broken        oranges.txt.broken        tomates.txt.broken
file.txt.broken             raisins.txt.broken        tomatoes.txt.broken
ice_cream.txt.broken        rasberries.txt.broken     udon_noddles.txt.broken
iced_coffe.txt.broken       reeses.txt.broken         ugali.txt.broken
instant_noodles.txt.broken  sausages.txt.broken       unagi.txt.broken
nachos.txt.broken           shopping_list.txt.broken
```

If you create a new file without the `.broken` extension in this directory and wait for a minute, the file will also be encrypted and renamed to add the `.broken` extension. This indicates that the malware may be scheduled as a cron job. The following command list all scheduled cron jobs:

```
localhost:~/Documents # crontab -l
* * * * * /usr/lib/zyppe
```

`/usr/lib/zyppe` is scheduled to run every minute. We can remove the scheduled cron jobs with `crontab -r`.

Decrypt documents

Using IDA disassembler, we can confirm that `/usr/lib/zyppe` is the malware. It encodes files using a slightly modified version of RC4 which uses `cyphertext[i] = plaintext[i] xor K[i] xor K[i-1]`, where K is the stream produced by RC4 , and key `A secret is no longer a secret once someone knows it.`

```
strcpy(v3, "A secret is no longer a secret once someone knows it");
result = 0x656D6F732065636ELL;
for ( i = 0; i <= 255; ++i )
{
  result = i;
  v2[i] = i;
}
v11 = 0;
for ( j = 0; j <= 255; ++j )
{
  v11 = (v2[j] + v11 + v3[j % 52]) % 256;
  v6 = v2[j];
  v2[j] = v2[v11];
  result = v11;
  v2[v11] = v6;
}
v9 = 0;
v11 = 0;
v8 = 0;
for ( k = 0; k <= 511; ++k )
{
  v9 = (v9 + 1) % 256;
  v11 = (v11 + v2[v9]) % 256;
  v5 = v2[v9];
  v2[v9] = v2[v11];
  v2[v11] = v5;
  v4 = v2[(v2[v11] + v2[v9]) % 256];
  a1[k] ^= v4 ^ v8;
  result = v4;
  v8 = v4;
}
```

As the inverse of the XOR function is XOR and the stream produced by RC4 is always the same for a given password, the following code decrypts the files in the `~/Document` directory (Note that with some versions of Ghidra, the decompiled code is not correct):

```
data[i2] = data[i2] ^ (byte)K ^ (byte)K;
```

This can't be correct as `num ^ num2 ^ num2 = num`.

Documents

The content of the decrypted shopping list is:

```
localhost:~/Documents # cat shopping_list.txt
/
[U]don noodles
[S]trawberries
[R]eese's
/
[B]anana chips
[I]ce Cream
[N]atillas
/
[D]onuts
[O]melettes
[T]acos
```

The first column of the shopping list is a hint to execute `/usr/bin/dot`:

```
localhost:~/Documents # /usr/bin/dot
Password: INVENTED_PASSWORD
Wrong password!
Password (ASCII):
```

`/usr/bin/dot` asks as for a password. Reversing this binary, we find out how this program works. It calculates the SHA256 hash of the given password and check if it is equal to the following hash:
`b3c20caa9a1a82add9503e0eac43f741793d2031eb1c6e830274ed5ea36238bf`
If it is, the program reverses the provided password, subtracts 1 to every of its bytes and appends @flare-on.com it. The program then outputs the flag.
However, as SHA256 is a cryptographic hash function, knowing how the program works doesn't give us the flag. We need to take a closer look at the rest of the files in the `~/Document` directory. Some of them seem to still be encoded. We do this in the order indicated in the shopping list.

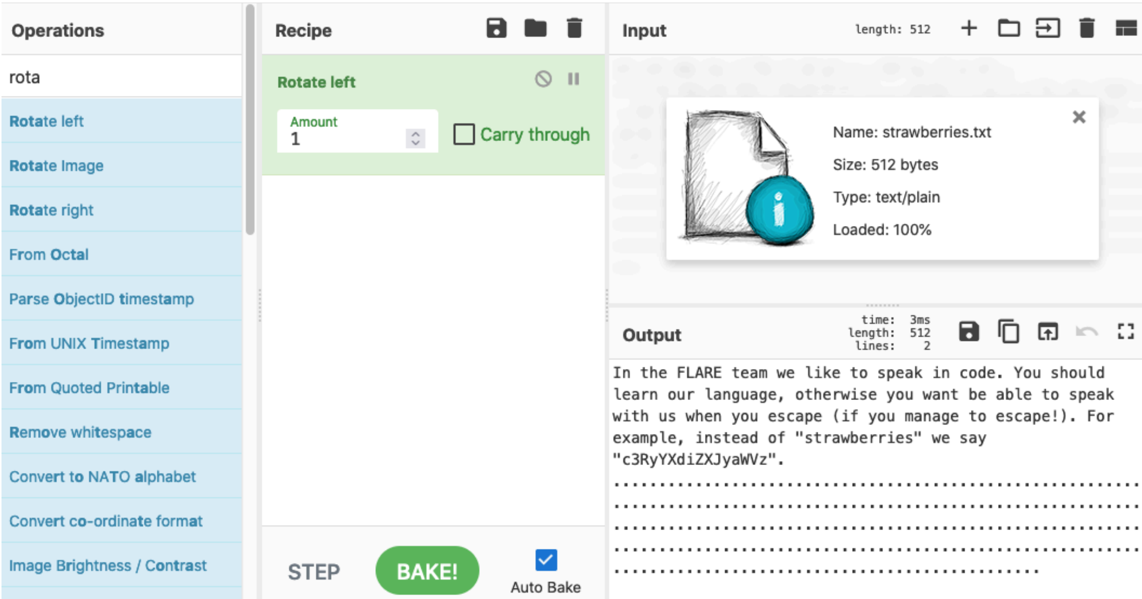
Udon noodles

`udon_noddles.txt` is in plain text and its content let us know that all files which start by the same letter are a group. `ugali.txt` gives us the hint to use [CyberChef]() and "rotate" to decode the next group of files (the one that start by "s"). `unagi.txt` contains the first byte of the password: `0x45`.

```
localhost:~/Documents # cat udon_noddles.txt
"ugali", "unagi" and "udon noodles" are delicious. What a coincidence that all of them st
art by "u"!
localhost:~/Documents # cat ugali.txt
Ugali with Sausages or Spaghetti is tasty. It doesn't matter if you rotate it left or rig
ht, it is still tasty! You should try to come up with a great recipe using CyberChef.
localhost:~/Documents # cat unagi.txt
The 1st byte of the password is 0x45
```

Strawberries

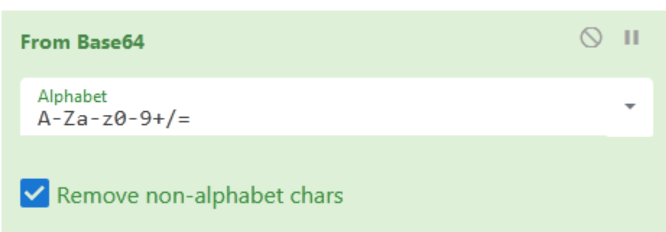
We can use [CyberChef and the recipe "Rotate left"]() to decode the files which start by "s".



`strawberries.txt` includes the string “c3RyYXdiZXJyaWVz” which is “strawberries” in Base64 and `spaghetti.txt` includes the string “c3BhZ2hldHRp” which is “spaghetti” in Base64. `sausages.txt` contains the second byte of the password: `0x34`.

Reese's

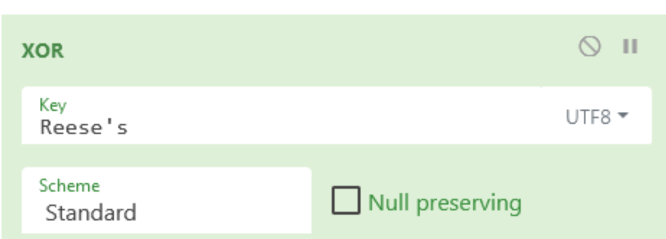
Following the hints of the files with start by “s”, we decode the files which start by “r” using [Base64].



`reeses.txt` tell us to decode the next files using XOR with key “Reese's”. `rasberries.txt` contains the third byte of the password: `0x51`.

Banana chips

We decode the files which start by “b” using [XOR with key “Reese's”].



`banana_chips.txt` provides the following formula:

```
ENCODED_BYTE + 27 + NUMBER1 * NUMBER2 - NUMBER3
```

`blackberries.txt` tells us to use our bash knowledge if we are not good at maths. This is a hint that `$NUMBER1`, `$NUMBER2` and `$NUMBER3` are environment variables defined in the VM. Getting their value allow us to simplify the formula (be careful with operator precedence):

```
ENCODED_BYTE + 27 + NUMBER1 * NUMBER2 - NUMBER3 =
ENCODED_BYTE + 27 + (NUMBER1 * NUMBER2) - NUMBER3 =
ENCODED_BYTE + 27 + (2 * 3) - 37 =
ENCODED_BYTE - 4
```

`blue_cheese.txt` contains the fourth byte of the password: `0x35`.

Ice cream

Using the formula we got from the previous files, we can decode the files which start by "i" by [subtracting 4 from every byte].

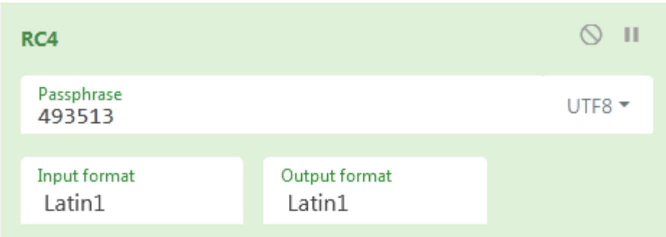
`ice_cream.txt` provides the following numbers to letter correspondence table:

```
0 - C
1 - B
2 - L
3 - E
4 - S
5 - F
6 - M
7 - I
8 - A
9 - R
```

`iced_coffe.txt` tells us to use RC4 for the next files and use the following "number" as key: "SREFBE". Using the correspondence table provided by ice_cream.txt, this is the number "493513". `instant_noodles.txt` contains the fifth byte of the password: `0xMS`. Using the correspondence table provided by `ice_cream.txt`, this is the byte `0x64`.

Natillas

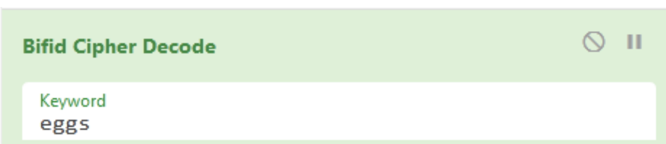
Using RC4 with key "493513", we decode the files which start by "n".



The `natillas.txt` file tells us to use an ingredient as keyword. As explained in the Wikipedia, the delicious Spanish desert called "Natillas" is made with milk and eggs. As milk is already given, "eggs" is the keyword. `nachos.txt` describes the Bifid cipher, a cipher invented by Felix Delastelle which combines Polybius square with transposition, and uses fractionation to achieve diffusion. `nutella.txt` contains the sixth byte of the password: `0x36`.

Donuts

We decode the files which start by "d" using the Bifid cipher with key "eggs".

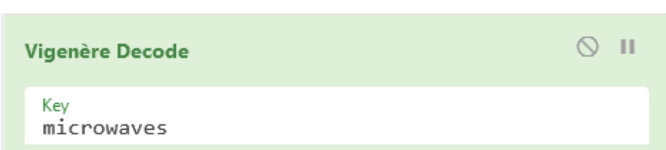


`donuts.txt` tells us to decode the next files using Vigenère cipher (first described by Giovan Battista Bellaso) with key "microwaves". `.daiquiris.txt` contains the seventh byte of the password: `0x66`. `dumplings.txt` is a hint to look for something you could have missed, meaning the following two things:

- The hidden file `.daiquiris.txt`.
- The password "microwaves", as it is not related at all to Giovan Battista Bellaso (microwaves were invented long after Giovan Battista Bellaso's death).

Omelettes

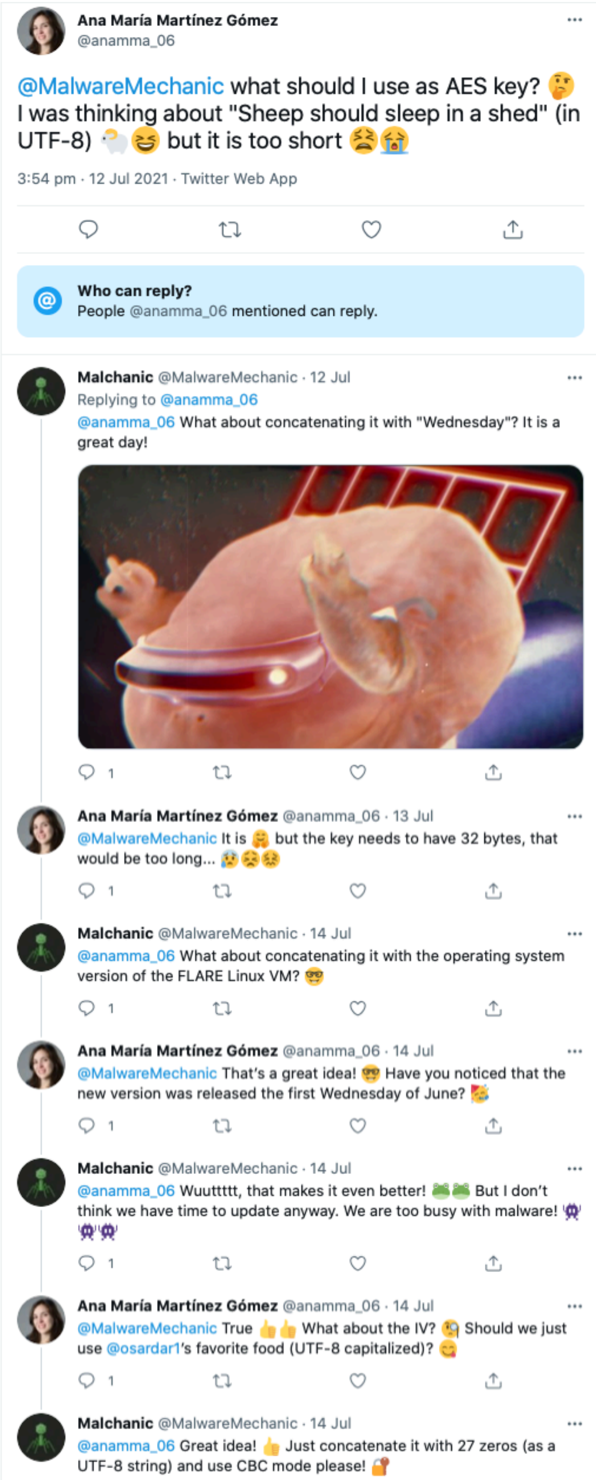
We decode the files which start by "o" using the Vigenère cipher with key "microwaves".



Both `omelettes.txt` and `oats.txt` link the following Twitter accounts:

- https://twitter.com/anamma_06
- https://twitter.com/MalwareMechanic

In the Tweets & replies tab of any of the two accounts there is a [conversation](#) between [@anamma06](#) and [@MalwareMechanic](#) about how to decode the next set of files.



The next set of files are encoded using AES. The key is the string "Sheep should sleep in a shed" followed by the OS version of the VM. The version is "15.2" as specified in the `/etc/os-release` file:

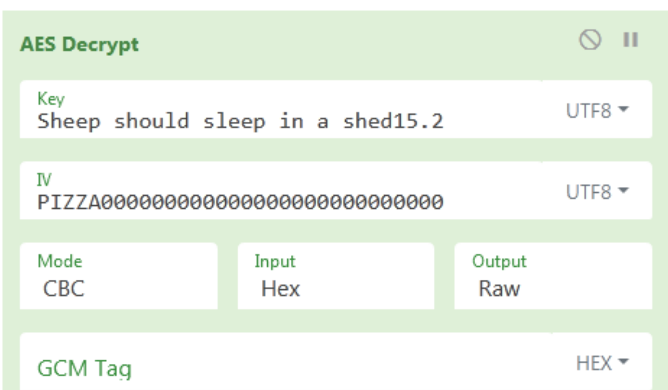
```
localhost:~/Documents # cat /etc/os-release
NAME="openSUSE Leap"
VERSION="15.2"
ID="opensuse-leap"
ID_LIKE="suse opensuse"
VERSION_ID="15.2"
PRETTY_NAME="openSUSE Leap 15.2"
ANSI_COLOR="0;32"
CPE_NAME="cpe:/o:opensuse:leap:15.2"
BUG_REPORT_URL="https://bugs.opensuse.org"
HOME_URL="https://www.opensuse.org/"
```

The IV is @osardar1's favorite food (PIZZA) followed by 27 zeros.

`oranges.txt` contains the eighth byte of the password: `0x60`.

Tacos

We decode the files which start by "t" using the following recipe in CyberChef:



`tomatoes.txt` gives the following information:

- It tells us that to count the number of unique words in `/etc/Quijote.txt` to get the 11th byte of the password: 0x6C (108 in decimal).

- It tells us to run the FLARE alias to get the 13th byte of the password: 0x35.

```
localhost:~ # FLARE
The 13th byte of the password is 0x35
```

tiramisu.txt gives us the following password bytes:

- 9th: 0x73 - 115 in decimal, the atomic number of the element moscovium
- 10th: 0x34 - 52 in decimal, the bell number preceding 203
- 12th: 0x44 - 68 in decimal, the largest known number to be the sum of two primes in exactly two different ways
- 14th (and last): 0x49 - 73 in decimal, the sum of the number of participants from Spain, Singapore and Indonesia that finished the FLARE-ON 7, FLARE-ON 6 or FLARE-ON 5. This information could be found in the Flare-On Challenge Solutions blog posts:
  - https://www.fireeye.com/blog/threat-research/2020/10/flare-on-7-challenge-solutions.html
  - https://www.fireeye.com/blog/threat-research/2019/09/2019-flare-on-challenge-solutions.html
  - https://www.fireeye.com/blog/threat-research/2018/10/2018-flare-on-challenge-solutions.html

Password

Putting together all the bytes of the password together, we get the following key in hexadecimal:

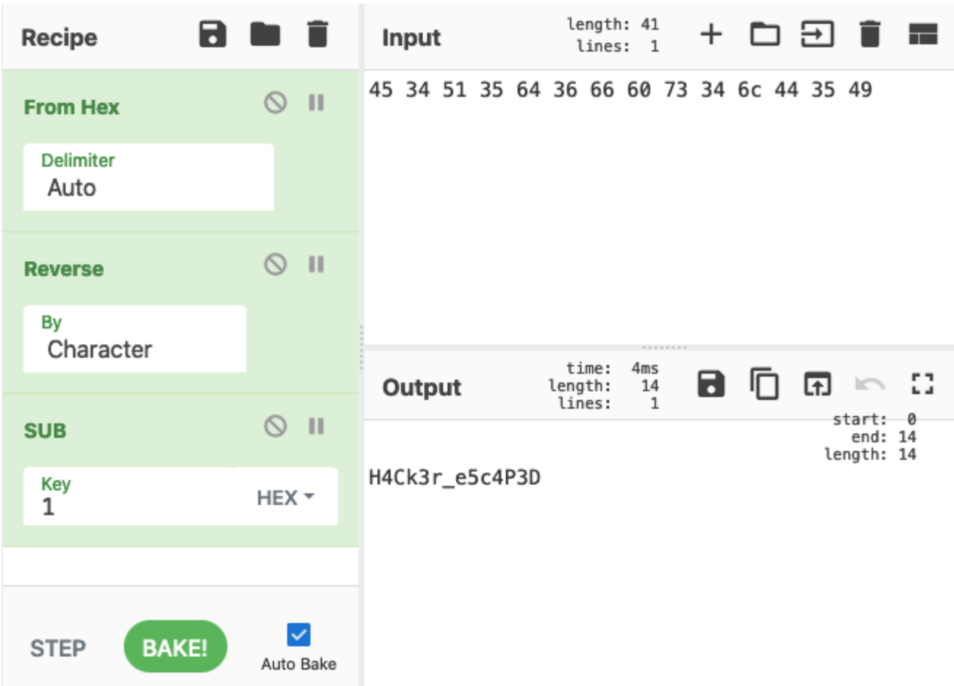
`45 34 51 35 64 36 66 60 73 34 6c 44 35 49`

The password in ASCII is `E4Q5d6f`s4lD5I`.

With the correct password, `/usr/bin/dot` outputs the flag:

```
localhost:~/Documents # /usr/bin/dot
Password: E4Q5d6f`s4lD5I
Correct password!
Flag: H4Ck3r_e5c4P3D@flare-on.com
```

Alternatively, the following CyberChef recipe decodes it as well:



Note that knowing how `/usr/bin/dot` works is also useful to check that you have the correct byte of the password in every step. You can use the previous CyberChef recipe with a part of the password and you should still get something readable.