

Technical Write up – KringleCon2019

Owner of Write up: @rootsecdev

Objective 0 – Talk to Santa in the Quad

This is easy enough. Go find the guy with the big white beard (not Jack Daniel) in the quad and talk to him to receive more objectives.

Objective 1 – Find the Turtle Doves

Catch these to love birds getting warm by the fireplace in the student union.



Objective 2 - Unredact threatening document.

Someone sent a threatening letter to Elf University. What is the first word in ALL CAPS in the subject line of the letter? Please find the letter in the Quad.

Analysis

This document is in the upper left corner of the quad. I almost missed it as it was partially blocked behind a Christmas tree.

Date: February 28, 2019

To the Administration, Faculty, and Staff of Elf University
17 Christmas Tree Lane
North Pole

From: A Concerned and Aggrieved Character

S
E
Confidential

Attention All Elf University Personnel,

H
M
L
C
C
F
C
K
Confidential

If you do not accede to our demands, we will be forced to take matters into our own hands.
We do not make this threat lightly. You have less than six months to act demonstrably.

Sincerely,

—A Concerned and Aggrieved Character

Since this is an improperly redacted PDF a simple Copy and Paste into a text editor such as notepad++ will reveal the text behind the confidential redaction. Those silly elves. They need to learn how to redact PDF's properly.

```
1 Date: February 28, 2019
2
3 To the Administration, Faculty, and Staff of Elf University
4 17 Christmas Tree Lane
5 North Pole
6
7 From: A Concerned and Aggrieved Character
8
9 Subject: DEMAND: Spread Holiday Cheer to Other Holidays and Mythical Characters... OR
10 ELSE!
11
12
13 Attention All Elf University Personnel,
14
15 It remains a constant source of frustration that Elf University and the entire operation at the
16 North Pole focuses exclusively on Mr. S. Claus and his year-end holiday spree. We URGE
17 you to consider lending your considerable resources and expertise in providing merriment,
18 cheer, toys, candy, and much more to other holidays year-round, as well as to other mythical
19 characters.
20
21 For centuries, we have expressed our frustration at your lack of willingness to spread your
22 cheer beyond the inaptly-called "Holiday Season." There are many other perfectly fine
23 holidays and mythical characters that need your direct support year-round.
24
25 If you do not accede to our demands, we will be forced to take matters into our own hands.
26 We do not make this threat lightly. You have less than six months to act demonstrably.
27
28 Sincerely,
29
30 --A Concerned and Aggrieved Character |
```

Answer:

DEMAND

Objective 3 - Windows Log Analysis: Evaluate Attack Outcome

We're seeing attacks against the Elf U domain! Using the event log data, identify the user account that the attacker compromised using a password spray attack. Bushy Evergreen is hanging out in the train station and may be able to help you out.

Log Data Link: <https://downloads.elfu.org/Security.evtx.zip>

Hint from Bushy Evergreen:

Deep Blue CI Links:

<https://www.ericconrad.com/2016/09/deepbluecli-powershell-module-for-hunt.html>

<https://github.com/sans-blue-team/DeepBlueCLI>

We first confirm with deepblue cli that password spray attacks were indeed occurring.

```

PS C:\DeepBlueCLI-master> .\DeepBlue.ps1 Security.evtx

Security warning
Run only scripts that you trust. While scripts from the internet can be useful, this script can potentially harm your
computer. If you trust this script, use the Unblock-File cmdlet to allow the script to run without this warning
message. Do you want to run C:\DeepBlueCLI-master\DeepBlue.ps1?
[D] Do not run [R] Run once [S] Suspend [?] Help (default is "D"): r

Date       : 11/19/2019 4:22:46 AM
Log        : Security
EventID    : 4648
Message    : Distributed Account Explicit Credential Use (Password Spray Attack)
Results    : The use of multiple user account access attempts with explicit credentials is an indicator of a password
              spray attack.
              Target Usernames: ygoldentrifle esparklesleigh hevergreen Administrator sgreenbells cjinglebuns
              tcandybaubles bbrandyleaves bevergreen lstripyleaves gchocolatewine wopenslae ltrufflefig supatree
              mstripysleigh pbrandyberry civysparkles sscarletpie ftwinklestockings cstripyfluff gcandyfluff smullingfluff
              hcandynaps mbrandybells twinterfig civypears ygreenpie ftinseltoes smary ttinselbubbles dsparkleleaves
              Accessing Username: -
              Accessing Host Name: -

```

Digging down further into deep blu cli we will eventually run across successful attempts on accounts. Highlighted below is the first account that has multiple successful logins.

```
Command :  
Decoded :  
  
Date : 8/23/2019 5:00:20 PM  
Log : Security  
EventID : 4672  
Message : Multiple admin logons for one account  
Results : Username: pminstix  
User SID Access Count: 2  
  
Command :  
Decoded :  
  
Date : 8/23/2019 5:00:20 PM  
Log : Security  
EventID : 4672  
Message : Multiple admin logons for one account  
Results : Username: DC1$  
User SID Access Count: 12  
  
Command :  
Decoded :  
  
Date : 8/23/2019 5:00:20 PM  
Log : Security  
EventID : 4672  
Message : Multiple admin logons for one account  
Results : Username: supatree  
User SID Access Count: 2  
  
Command :  
Decoded :  
  
Date : 8/23/2019 5:00:20 PM  
Log : Security  
EventID : 4672  
Message : High number of logon failures for one account  
Results : Username: ygoldentrifle  
Total logon failures: 77  
  
Command :  
Decoded :  
  
Date : 8/23/2019 5:00:20 PM  
Log : Security  
EventID : 4672  
Message : High number of logon failures for one account  
Results : Username: esparklesleigh  
Total logon failures: 77
```

Answer: supatree

Objective 4 – Windows Log Analysis: Determine attacker technique

Using these normalized Sysmon logs, identify the tool the attacker used to retrieve domain password hashes from the lsass.exe process. For hints on achieving this objective, please visit Hermey Hall and talk with SugarPlum Mary.

URL for sysmon data

<https://downloads.elfu.org/sysmon-data.json.zip>

Hint: Sugar Plum Mary drops a nice link and good overview of how sysmon works by Carlos Perez

<https://www.darkoperator.com/blog/2014/8/8/sysinternals-sysmon>

So next I get my google kung fu on and I start searching for easy ways to search through sysmon data. Not long after searching I found a manual on EQL that is exactly how I want to conduct searching in an easy way though this data:

<https://readthedocs.org/projects/eql/downloads/pdf/latest/>

Install EQL on Kali/debian

sudo apt install python3-pip

pip3 install eql

If on Fedora

pip3 install eql

sudo yum install jq

There appears to be only one instance where lsass.exe is called in the logs. We now know the logon id and timestamp of when this event occurred.

```
[rootsecdev@localhost Downloads]$ ls
sysmon-data.json
[rootsecdev@localhost Downloads]$ eql query -f sysmon-data.json "process where parent_process_name = '*lsass*' | jq
{
  "command_line": "C:\\Windows\\system32\\cmd.exe",
  "event_type": "process",
  "logon_id": 999,
  "parent_process_name": "lsass.exe",
  "parent_process_path": "C:\\Windows\\System32\\lsass.exe",
  "pid": 3440,
  "ppid": 632,
  "process_name": "cmd.exe",
  "process_path": "C:\\Windows\\System32\\cmd.exe",
  "subtype": "create",
  "timestamp": 132186398356220000,
  "unique_pid": "{7431d376-dedb-5dd3-0000-001027be4f00}",
  "unique_ppid": "{7431d376-cd7f-5dd3-0000-001013920000}",
  "user": "NT AUTHORITY\\SYSTEM",
  "user_domain": "NT AUTHORITY",
  "user_name": "SYSTEM"
}
```


Next, we need to pivot around the sequence of events when lsass was triggered and what happened immediately afterwards.

```
[rootsecdev@localhost Downloads]$ eql query -f sysmon-data.json "process where logon_id = 999 and timestamp > 1321863983000000000 and timestamp < 1321863990000000000" | jq
{
  "command_line": "C:\\Windows\\system32\\cmd.exe",
  "event_type": "process",
  "logon_id": 999,
  "parent_process_name": "lsass.exe",
  "parent_process_path": "C:\\Windows\\System32\\lsass.exe",
  "pid": 3440,
  "ppid": 632,
  "process_name": "cmd.exe",
  "process_path": "C:\\Windows\\System32\\cmd.exe",
  "subtype": "create",
  "timestamp": 132186398356220000,
  "unique_pid": "{7431d376-dedb-5dd3-0000-001027be4f00}",
  "unique_ppid": "{7431d376-cd7f-5dd3-0000-001013920000}",
  "user": "NT AUTHORITY\\SYSTEM",
  "user_domain": "NT AUTHORITY",
  "user_name": "SYSTEM"
}
{
  "command_line": "ntdsutil.exe \\ac i ntds\\ ifm \\\"create full c:\\\\hive\\\" q q",
  "event_type": "process",
  "logon_id": 999,
  "parent_process_name": "cmd.exe",
  "parent_process_path": "C:\\Windows\\System32\\cmd.exe",
  "pid": 3556,
  "ppid": 3440,
  "process_name": "ntdsutil.exe",
  "process_path": "C:\\Windows\\System32\\ntdsutil.exe",
  "subtype": "create",
  "timestamp": 1321863984703000000,
  "unique_pid": "{7431d376-dee7-5dd3-0000-0010f0c44f00}",
  "unique_ppid": "{7431d376-dedb-5dd3-0000-001027be4f00}",
  "user": "NT AUTHORITY\\SYSTEM",
  "user_domain": "NT AUTHORITY",
  "user_name": "SYSTEM"
}
```

Ntdsutil.exe was used to dump the credentials.

Answer: ntdsutil

Objective 5: Network Log Analysis: Determine Compromised System

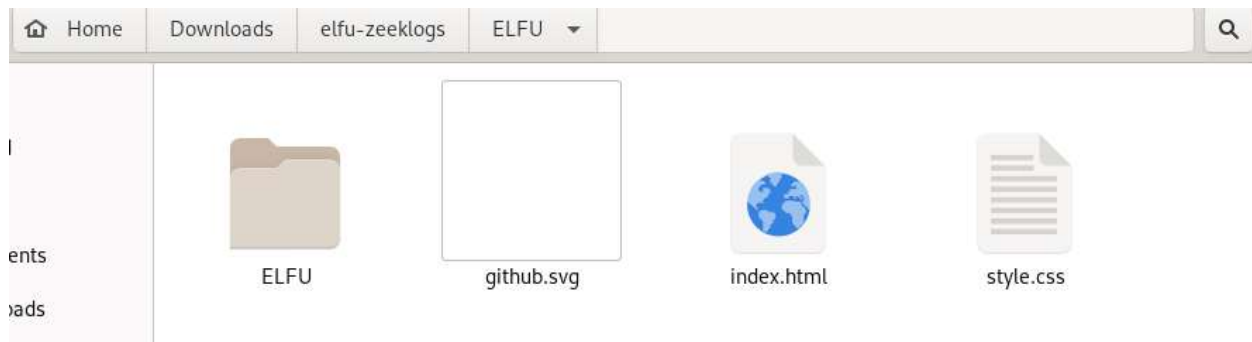
The attacks don't stop! Can you help identify the IP address of the malware-infected system using these Zeek logs? For hints on achieving this objective, please visit the Laboratory and talk with Sparkle Redberry.

Zeek Log Location: <https://downloads.elfu.org/elfu-zeeklogs.zip>


Hint: Talk to Sparkle Redberry

After talking with Sparkle we are given a hint on RITA >> <https://www.activecountermeasures.com/free-tools/rita/>

Installing RITA on a VM is unnecessary for this challenge. I saw some people chatting about doing so to complete the objective. After looking around in the logs I stumbled upon an ELFU folder that contained all the RITA outputs. Just Click on the index.html to get started.



In the beacons page we immediately see the first connection is making an abnormally high amount of connections to 144.202.46.214.

<div><div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div></div><div></div></div></div><div>RITA</div><div>Viewing: ELFU</div><div>Beacons</div><div>Strobes</div><div>DNS</div><div>BL Source IPs</div><div>BL Dest. IPs</div><div>BL Hostnames</div><div>Long Connections</div><div>User Agents</div><div>RITA on </div></div>														
Score	Source	Destination	Connections	Avg. Bytes	Intvl. Range	Size Range	Intvl. Mode	Size Mode	Intvl. Mode Count	Size Mode Count	Intvl. Skew	Size Skew	Intvl. Dispersion	Size Dispersion
0.998	192.168.134.130	144.202.46.214	7660	1156.000	10	683	10	563	6926	7641	0.000	0.000	0	0

Answer: 192.168.134.130

Objective 6: Splunk

Access <https://splunk.elfu.org/> as elf with password elfsocks. What was the message for Kent that the adversary embedded in this attack? The SOC folks at that link will help you along! For hints on achieving this objective, please visit the Laboratory in Hermey Hall and talk with Prof. Banas.

1.What is the short host name of Professor Banas' computer?

sweetums

In splunk we can do a simple search for the event logs:

sourcetype=wineventlog

08/25/2019 09:31:17 AM
LogName=Security
SourceName=Microsoft Windows security auditing.
EventCode=4688
EventType=0
Type=Information
ComputerName=sweetums.elfu.org
TaskCategory=Process Creation
OpCode=Info
RecordNumber=1183774
Keywords=Audit Success
Message=A new process has been created.

Creator Subject:
Security ID: NT AUTHORITY\SYSTEM
Account Name: SWEETUMS\$
Account Domain: WORKGROUP
Logon ID: 0x3E7

Target Subject:
Security ID: SWEETUMS\cbanas
Account Name: cbanas
Account Domain: SWEETUMS
Logon ID: 0x54399

2. What is the name of the sensitive file that was likely accessed and copied by the attacker? Please provide the fully qualified location of the file. (Example: C:\temp\report.pdf)

C:\Users\cbanas\Documents\Naughty_and_Nice_2019_draft.txt

Alice Bluebird was a little worried that the attackers may have found santa's sensitive data since the Professor was in Santa's inner circle. We are also provided with a hint from Alice after answering Question 1. Search Splunk for the following.

index=main santa

We see that the first log is where this file access and dropping PowerShell code...not good..

3. What is the fully-qualified domain name(FQDN) of the command and control(C2) server? (Example: badguy.baddies.com)

144.202.46.214.vultr.com

Alice tells us to search for event code 3 in splunk. She even gives us the search parameters in a link. We see several connections coming from the domain host above running some rogue powershell.

```
index=main sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational powershell  
EventCode=3
```

4. What document is involved with launching the malicious PowerShell code? Please provide just the filename. (Example: results.txt)

19th Century Holiday Cheer Assignment.docm

So Alice drops this splunk search command on us:

```
index=main sourcetype="WinEventLog:Microsoft-Windows-Powershell/Operational"
```

She also tells us we can pivot by piping in reverse so basically this:

```
index=main sourcetype="WinEventLog:Microsoft-Windows-Powershell/Operational" | reverse
```

Alice also tells us we need to see what launched those processes using the following query:

```
index=main sourcetype=WinEventLog EventCode=4688
```

We want to stick to 4688 events but there are 157 of them. We need to eliminate some of them. We can easily go to 57 events by pivoting to cbanas's account:

```
index=main sourcetype=WinEventLog EventCode=4688 Account_Name=cbanas
```

We can put this down to 23 events by looking at the New process name. If we look at the left hand panel of splunk we can see the top events. Since we know he opened up a document we can safely assume he used Microsoft Word to open the document up.

```
index=main sourcetype=WinEventLog EventCode=4688 New_Process_Name="C:\\Program Files  
(x86)\\Microsoft Office\\root\\Office16\\WINWORD.EXE"
```

The second event after running this query we see the professor received a zip attachment and upon extracting we will notice it's a docm file which indicated it's a macro enabled document. This deserves some scrutiny. So I take a change a put the document name as the answer and appears to be correct.

5. How many unique email addresses were used to send Holiday Cheer essays to Professor Banas? Please provide the numeric value. (Example: 1)

21

Alice provides another clue with an example stoq command to look at emails:

```
index=main sourcetype=stoq | table _time results{}.workers.smtp.to results{}.workers.smtp.from  
results{}.workers.smtp.subject results{}.workers.smtp.body | sort - _time
```

Since all Submissions must be made via email and the following subject line "Holiday Cheer Assignment Submission" we will need to filter the subject line and emails coming into Carl.

```
index=main sourcetype=stoq | table _time results{}.workers.smtp.to results{}.workers.smtp.from  
results{}.workers.smtp.subject results{}.workers.smtp.body | sort - _time | search  
results{}.workers.smtp.subject="*Holiday Cheer Assignment Submission*"  
results{}.workers.smtp.to="*carl*"
```

This puts our submission count to 21.

6. What was the password for the zip archive that contained the suspicious file?

123456789

This can be easily found by searching for the term password in the body of the email.

```
index=main sourcetype=stoq | table _time results{}.workers.smtp.to results{}.workers.smtp.from  
results{}.workers.smtp.subject results{}.workers.smtp.body | sort - _time | search  
results{}.workers.smtp.body="*password"
```

7. What email address did the suspicious file come from?

bradly.buttercups@eifu.org

This was easy to find as step 6 gave us this answer as well.

Challenge Question:

What was the message for Kent that the adversary embedded in this attack?

Alice gives the following code to help us out:

```
index=main sourcetype=stoq "results{}.workers.smtp.from"="bradly buttercups  
<bradly.buttercups@eifu.org>"
```

She also gives us the following:

```
| eval results = spath(_raw, "results{}")  
  
| mvexpand results  
  
| eval path=spath(results, "archivers.filedir.path"), filename=spath(results,  
"payload_meta.extra_data.filename"), fullpath=path."/".filename  
  
| search fullpath!=""  
  
| table filename,fullpath
```

When we combine the two we quickly are able to determine where the file is hanging out in stoq for review.

/home/ubuntu/archive/c/6/e/1/7/c6e175f5b8048c771b3a3fac5f3295d2032524af/19th Century Holiday Cheer Assignment.docm

If we drill down into this file location we get the following message when opening the file via a text editor.

[Cleaned for your safety. Happy Holidays!

In the real world, This would have been a wonderful artifact for you to investigate, but it had malware in it of course so it's not posted here. Fear not! The core.xml file that was a component of this original macro-enabled Word doc is still in this File Archive thanks to stoQ. Find it and you will be a happy elf :-)

Next we need to look at the core.xml file in the following location

`/home/ubuntu/archive/f/f/1/e/a/ff1ea6f13be3faabd0da728f514deb7fe3577cc4/core.xml`

Upon inspection of the file we see the following message in our answer:

Answer:

Kent you are so unfair. And we were going to make you the king of the Winter Carnival.

Objective 7 Get Access To The Steam Tunnels

Gain access to the steam tunnels. Who took the turtle doves? Please tell us their first and last name. For hints on achieving this objective, please visit Minty's dorm room and talk with Minty Candy Cane.

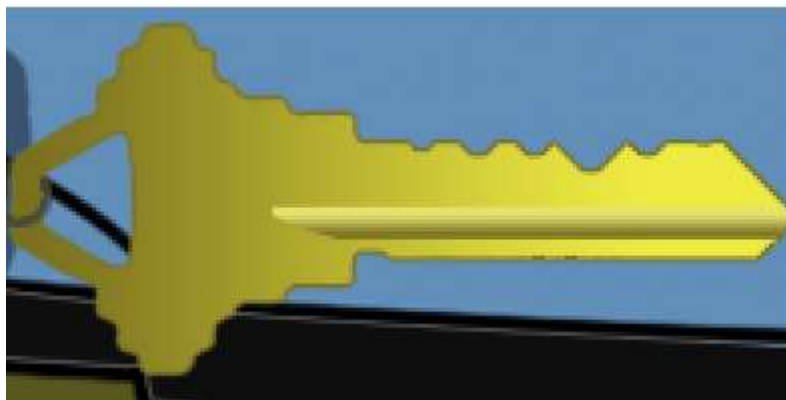
To solve this objective you really need to watch Deviant's video on optical decoding of keys

<https://www.youtube.com/watch?v=KU6FJnbkeLA>

So we run into Laslo aka Krampus and we need a screenshot of his keys. You can't get a proper screenshot of his keys within the browser. At this point I did a F12 into a console and managed to pull Krampuses avatar out of the images folder.

<https://2019.kringlecon.com/images/avatars/elves/krampus.png>

At this point you can zoom in on his image and use snip to grab an image of the key. I used paint to flip the image around until I got a really good shot of his key in the position we need it to be in.



Next I used the following key measurements page and over to the right of the document are the bit sizes. This was helpful after some trial and error in determining the correct sequence of numbers to use to cut the key.

<https://github.com/deviantollam/decoding/blob/master/Key%20Measurments/Key%20Measurments%20-%20Schlage.png>

This card will also help if you overlay it with gimp

<https://github.com/deviantollam/decoding/blob/master/Devious%20Gauge%20Card/Devious%20Gauge%20Card.jpg>

The key below used a combination of the following:

122520



Objective 8 Bypassing the Frido Sleigh CAPTEHA

Help Krampus beat the Frido Sleigh contest. For hints on achieving this objective, please talk with Alabaster Snowball in the Speaker Unpreparedness Room.

URL:

<https://fridosleigh.com/>

Hint:

Alabaster drops the following hint for this machine learning challenge. This is Chris Davis's talk on Machine Learning Use Cases for Cybersecurity

https://www.youtube.com/watch?v=jmVPLwjm_zs&feature=youtu.be

Github

https://github.com/chrisjd20/img_rec_tf_ml_demo

Next we will talk to Krampus in his lair. After talking with Krampus he gives us the following two links that will help us on this objective. The first is a python script for an api that we need to run that will help

bypass the image capteha on the Frido Sleigh website. The second link is to train tensorflow on the images Krampus provided us.

API download:

https://downloads.elfu.org/capteha_api.py

Images download:

https://downloads.elfu.org/capteha_images.tar.gz

To do this objective I opted to spin up a VM of ubuntu server. Once the Ubuntu server is up we download Chris's repository and install tensorflow with the following commands:

```
git clone https://github.com/chrisjd20/img_rec_tf_ml_demo.git
```

```
cd img_rec_tf_ml_demo
```

```
sudo apt install python3 python3-pip -y
```

```
sudo python3 -m pip install --upgrade pip
```

```
sudo python3 -m pip install --upgrade setuptools
```

```
sudo python3 -m pip install --upgrade tensorflow==1.15
```

```
sudo python3 -m pip install tensorflow_hub
```

Next remove any folders and images in the "training_images" folder and "unknown_images" folder.

Copy the capteh_images.tar.gz file to the training_images folder and extract the files.

At this point we are ready to train our server with the images provided with the following command:

```
python3 retrain.py --image_dir ./training_images/
```

```
rootsecdev@ML-OBJ8:~$ cd img_rec_tf_ml_demo/
rootsecdev@ML-OBJ8:~/img_rec_tf_ml_demo$ ls
predict_images_using_trained_model.py  README.md  retrain.py  training_images  unknown_images
rootsecdev@ML-OBJ8:~/img_rec_tf_ml_demo$ python3 retrain.py --image_dir ./training_images/
```

You will receive a messages about bottleneck files being created. This is normal. Depending on your machine speed this may take up to 20 minutes to train the server on the images provided. So please....grab a cup of coffee.

```
I0112 13:34:33.418992 140101959145280 retrain.py:477] 100 bottleneck files created.
I0112 13:34:40.174310 140101959145280 retrain.py:477] 200 bottleneck files created.
I0112 13:34:47.009959 140101959145280 retrain.py:477] 300 bottleneck files created.
```

Once bottleneck processing if finished you should see your server start training itself on the training images

```

I0112 13:48:32.769616 140101959145280 retrain.py:1131] 2020-01-12 13:48:32.769552: Step 310: Validation accuracy = 100.0% (N=100)
I0112 13:48:33.520642 140101959145280 retrain.py:1110] 2020-01-12 13:48:33.520531: Step 320: Train accuracy = 100.0%
I0112 13:48:33.520829 140101959145280 retrain.py:1112] 2020-01-12 13:48:33.520816: Step 320: Cross entropy = 0.079038
I0112 13:48:33.590001 140101959145280 retrain.py:1131] 2020-01-12 13:48:33.589939: Step 320: Validation accuracy = 100.0% (N=100)
I0112 13:48:34.308407 140101959145280 retrain.py:1110] 2020-01-12 13:48:34.308270: Step 330: Train accuracy = 100.0%
I0112 13:48:34.308671 140101959145280 retrain.py:1112] 2020-01-12 13:48:34.308656: Step 330: Cross entropy = 0.089255
I0112 13:48:34.379705 140101959145280 retrain.py:1131] 2020-01-12 13:48:34.379644: Step 330: Validation accuracy = 100.0% (N=100)

```

Once this is done we now need to combine these two python scripts:

capteha_api.py – Krampus should have supplied you with this script

predict_images_using_trained_model.py – this script was supplied in Chris’s github repository and should be located in the /img_rec_tf_ml_demo directory

In Krampus’s script the API getting called for each request is in base64 format for each image. Its extremely important to note. I probably did this script in a non efficient way as I was dumping the unknown images to the unknown images directory instead of memory >>

<https://fridosleigh.com/api/capteha/request>

I also had to speed up my threads. So I bumped it from a default of 10 to 280 threads. Otherwise you will more than likely be met with timeout issues. Also I chose to do this exercise with ubuntu server. Less overhead than running full desktop mode with gnome.

Here is the full script. I found it easier to take the existing predict_images_using_trained_model.py python script and merge it with Krampus’s script.

Here is the full script I used:

```
#!/usr/bin/python3
```

```
# Image Recognition Using Tensorflow Exmample.
```

```
# Code based on example at:
```

```
#
```

```
https://raw.githubusercontent.com/tensorflow/tensorflow/master/tensorflow/examples/label_image/label_image.py
```

```
import os
```

```
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
```

```
import tensorflow as tf
```

```
tf.logging.set_verbosity(tf.logging.ERROR)
```

```
import numpy as np
```

```
import threading
```

```
import queue
```

```
import time
```

```
import sys
```



```
import base64
```

```
import re
```

```
# sudo apt install python3-pip
```

```
# sudo python3 -m pip install --upgrade pip
```

```
# sudo python3 -m pip install --upgrade setuptools
```

```
# sudo python3 -m pip install --upgrade tensorflow==1.15
```

```
def load_labels(label_file):
```

```
    label = []
```

```
    proto_as_ascii_lines = tf.gfile.GFile(label_file).readlines()
```

```
    for l in proto_as_ascii_lines:
```

```
        label.append(l.rstrip())
```

```
    return label
```

```
def predict_image(q, sess, graph, image_bytes, img_full_path, labels, input_operation,  
output_operation):
```

```
    image = read_tensor_from_image_bytes(image_bytes)
```

```
    results = sess.run(output_operation.outputs[0], {
```

```
        input_operation.outputs[0]: image
```

```
    })
```

```
    results = np.squeeze(results)
```

```
    prediction = results.argsort()[-5:][::-1][0]
```

```
    q.put( {'img_full_path':img_full_path, 'prediction':labels[prediction].title(),  
'percent':results[prediction]} )
```

```
def load_graph(model_file):
```

```
    graph = tf.Graph()
```

```
    graph_def = tf.GraphDef()
```

```

with open(model_file, "rb") as f:
    graph_def.ParseFromString(f.read())
with graph.as_default():
    tf.import_graph_def(graph_def)
return graph

```

```

def read_tensor_from_image_bytes(imagebytes, input_height=299, input_width=299, input_mean=0,
input_std=255):

```

```

    image_reader = tf.image.decode_png( imagebytes, channels=3, name="png_reader")
    float_caster = tf.cast(image_reader, tf.float32)
    dims_expander = tf.expand_dims(float_caster, 0)
    resized = tf.image.resize_bilinear(dims_expander, [input_height, input_width])
    normalized = tf.divide(tf.subtract(resized, [input_mean]), [input_std])
    sess = tf.compat.v1.Session()
    result = sess.run(normalized)
    return result

```

```

def main_predict():

```

```

    # Loading the Trained Machine Learning Model created from running retrain.py on the
training_images directory

```

```

    graph = load_graph('/tmp/retrain_tmp/output_graph.pb')
    labels = load_labels("/tmp/retrain_tmp/output_labels.txt")

```

```

    # Load up our session

```

```

    input_operation = graph.get_operation_by_name("import/Placeholder")
    output_operation = graph.get_operation_by_name("import/final_result")
    sess = tf.compat.v1.Session(graph=graph)

```

```

    # Can use queues and threading to speed up the processing

```

```

q = queue.Queue()

unknown_images_dir = 'unknown_images'

unknown_images = os.listdir(unknown_images_dir)

#Going to iterate over each of our images.
for image in unknown_images:

    img_full_path = '{}/{}'.format(unknown_images_dir, image)

    print('Processing Image {}'.format(img_full_path))

    # We don't want to process too many images at once. 10 threads max
    while len(threading.enumerate()) > 280:

        time.sleep(0.0001)

    #predict_image function is expecting png image bytes so we read image as 'rb' to get a bytes object
    image_bytes = open(img_full_path,'rb').read()

    threading.Thread(target=predict_image, args=(q, sess, graph, image_bytes, img_full_path, labels,
input_operation, output_operation)).start()

    print('Waiting For Threads to Finish...')

    while q.qsize() < len(unknown_images):

        time.sleep(0.001)

#getting a list of all threads returned results
prediction_results = [q.get() for x in range(q.qsize())]

#do something with our results... Like print them to the screen.
uuids = {}

for prediction in prediction_results:

    uuids[re.search('/([^\s/]+?).png', prediction['img_full_path']).groups(1)[0]] = prediction['prediction']

```

```

return uuids

#!/usr/bin/env python3
# Fridosleigh.com CAPTEHA API - Made by Krampus Hollyfeld
import requests
import json
import sys

def main():
    yourREALEmailAddress = "rootsecdev@gmail.com"

    # Creating a session to handle cookies
    s = requests.Session()
    url = "https://fridosleigh.com/"

    json_resp = json.loads(s.get("{}api/capteha/request".format(url)).text)

    b64_images = json_resp['images']          # A list of dictionaries eaching containing the keys
    'base64' and 'uuid'

    challenge_image_type = json_resp['select_type'].split(',') # The Image types the CAPTEHA Challenge
    is looking for.

    challenge_image_types = [challenge_image_type[0].strip(), challenge_image_type[1].strip(),
    challenge_image_type[2].replace(' and ',').strip()] # cleaning and formatting

    for img in b64_images:
        i_uuid = img['uuid']
        i_base64 = img['base64']
        open(f"unknown_images/{i_uuid}.png", 'wb').write(base64.b64decode(i_base64))

    results = main_predict()

```

```

found_images = list()

for img in results:
    if results[img] in challenge_image_types:
        found_images.append(img)

final_answer = ','.join( found_images )

# This should be JUST a csv list image uuids ML predicted to match the challenge_image_type .
#final_answer = ','.join( [ img['uuid'] for img in b64_images ] )

json_resp = json.loads(s.post("{}api/capteha/submit".format(url), data={'answer':final_answer}).text)
if not json_resp['request']:
    # If it fails just run again. ML might get one wrong occasionally
    print('FAILED MACHINE LEARNING GUESS')
    print('-----\nOur ML Guess:\n-----\n{}'.format(final_answer))
    print('-----\nServer Response:\n-----\n{}'.format(json_resp['data']))
    sys.exit(1)

print('CAPTEHA Solved!')

# If we get to here, we are successful and can submit a bunch of entries till we win
userinfo = {
    'name':'Krampus Hollyfeld',
    'email':yourREALEmailAddress,
    'age':180,
    'about':"Cause they're so flippin yummy!",
    'favorites':'thickmints'
}

# If we win the once-per minute drawing, it will tell us we were emailed.

```

```
# Should be no more than 200 times before we win. If more, somethings wrong.

entry_response = ""

entry_count = 1

while yourREALemailAddress not in entry_response and entry_count < 200:

    print('Submitting lots of entries until we win the contest! Entry #{}'.format(entry_count))

    entry_response = s.post("{}api/entry".format(url), data=userinfo).text

    entry_count += 1

print(entry_response)

if __name__ == "__main__":

    main()
```

```
Submitting lots of entries until we win the contest! Entry #95
Submitting lots of entries until we win the contest! Entry #96
Submitting lots of entries until we win the contest! Entry #97
Submitting lots of entries until we win the contest! Entry #98
Submitting lots of entries until we win the contest! Entry #99
Submitting lots of entries until we win the contest! Entry #100
Submitting lots of entries until we win the contest! Entry #101
Submitting lots of entries until we win the contest! Entry #102
Submitting lots of entries until we win the contest! Entry #103
{"data": "<h2 id='result_header'> Entries for email address rootsecdev@gmail.com no longer accepted as our systems show your email was already randomly selected as a winner! Go check your email to get your winning code. Please allow up to 3-5 minutes for the email to arrive in your inbox or check your spam filter settings. <br><br> Congratulations and Happy Holidays!</h2>", "request": true}
rootsecdev@ML-OBJ8:~/img_rec_tf_ml_demo$
```

Winning Code:

8la8LiZEwvyZr2WO

Objective 9 - Retrieve Scraps of Paper from Server (Incomplete)

Gain access to the data on the Student Portal server and retrieve the paper scraps hosted there. What is the name of Santa's cutting-edge sleigh guidance system? For hints on achieving this objective, please visit the dorm and talk with Pepper Minstix.

URL:

<https://studentportal.elfu.org/>

Pepper Minstix Hint:

SQL Map Tamper Scripts >> <https://pen-testing.sans.org/blog/2017/10/13/sqlmap-tamper-scripts-for-the-win>

SQL Injection from OWASP >> https://www.owasp.org/index.php/SQL_Injection

At this time I have this objective not complete. I am unfortunately a SQL injection noob.

Objective 10 - Recover Cleartext Document (Incomplete)

The Elfscrow Crypto tool is a vital asset used at Elf University for encrypting SUPER SECRET documents. We can't send you the source, but we do have debug symbols that you can use.

Recover the plaintext content for this encrypted document. We know that it was encrypted on December 6, 2019, between 7pm and 9pm UTC.

What is the middle line on the cover page? (Hint: it's five words)

** For hints on achieving this objective, please visit the NetWars room and talk with Holly Evergreen

URL's:

Elfscrow tool >> <https://downloads.elfu.org/elfscrow.exe>

Debug Symbols >> <https://downloads.elfu.org/elfscrow.pdb>

Hint:

Reversing Crypto the easy way - <https://www.youtube.com/watch?v=obJdpKDpFBA&feature=youtu.be>

Crypto Talk slide decks >> <https://github.com/CounterHack/reversing-crypto-talk-public>

This is a fun objective with crypto. I'll make a good point at the end of this objective on why you should never roll your own crypto. Even for an elf. We make use of Wireshark and the NSA tool Ghidra in this objective. To start we download the three files for this objective. The EXE (Crypto tool) the encrypted document, and the debugger file:

As with any tool we should see how to use it using the universal `--help` command with the tool.

```
Command Prompt

Volume in drive C has no label.
Volume Serial Number is 7EF5-63C8

Directory of C:\Users\rootsecdev\Downloads

01/01/2020  12:07 PM    <DIR>          .
01/01/2020  12:07 PM    <DIR>          ..
01/01/2020  12:06 PM                22,528 elfscrow.exe
01/01/2020  12:06 PM               297,984 elfscrow.pdb
01/01/2020  12:06 PM          1,889,112 ElfUResearchLabsSuperSledOMaticQuickStartGuideV1.2.pdf.enc
               3 File(s)          2,209,624 bytes
               2 Dir(s)  49,295,839,232 bytes free

C:\Users\rootsecdev\Downloads>elfscrow.exe --help
Welcome to ElfScrow V1.01, the only encryption trusted by Santa!

Are you encrypting a file? Try --encrypt! For example:

    elfscrow.exe --encrypt <infile> <outfile>

You'll be given a secret ID. Keep it safe! The only way to get the file
back is to use that secret ID to decrypt it, like this:

    elfscrow.exe --decrypt --id=<secret_id> <infile> <outfile>

You can optionally pass --insecure to use unencrypted HTTP. But if you
do that, you'll be vulnerable to packet sniffers such as Wireshark that
could potentially snoop on your traffic to figure out what's going on!

C:\Users\rootsecdev\Downloads>
```

At this point we will need to fire up wireshark and encrypt a text file using the `--insecure` pass.

```

C:\Users\rootsecdev\Downloads>elfscrow.exe --encrypt --insecure SuperSecret.txt SuperSecret.txt.enc
Welcome to ElfScrow V1.01, the only encryption trusted by Santa!

*** WARNING: This traffic is using insecure HTTP and can be logged with tools such as Wireshark

Our miniature elves are putting together random bits for your secret key!

Seed = 1577904303

Generated an encryption key: db664fa271df72d2 (length: 8)

Elfscrowing your key...

Elfscrowing the key to: elfscrow.elfu.org/api/store

Your secret id is 00c632be-3a7d-4670-a117-1f9b6dc545cb - Santa Says, don't share that key with anybody!
File successfully encrypted!

  ++=====++
  ||          || |
  ||  ELF-SCROW  ||
  ||            ||
  ||  0          ||
  ||  |          ||
  ||  (0)-       ||
  ||  |          ||
  ||            ||
  ||          ||
  ||          ||
  ++=====++

C:\Users\rootsecdev\Downloads>_

```

We get some clues as to where this key escrow is ultimately going in the wireshark capture

```

v Content-Length: 16\r\n
  [Content length: 16]
  Cache-Control: no-cache\r\n
  \r\n
  [Full request URI: http://elfscrow.elfu.org/api/store]
  [HTTP request 1/1]
  [Response in frame: 10]
  File Data: 16 bytes
v Data (16 bytes)
  Data: 64623636346661323731646637326432
  File Data: 16 bytes

```

Next let's reverse and decrypt the file

```
C:\Users\rootsecdev\Downloads>elfscrow.exe --decrypt --insecure --id=00c632be-3a7d-4670-a117-1f9b6dc545cb SuperSecret.txt.enc SuperSecret.txt
Welcome to ElfScrow V1.0! , the only encryption trusted by Santa!
```

```
*** WARNING: This traffic is using insecure HTTP and can be logged with tools such as Wireshark
```

```
Let's see if we can find your key...

Retrieving the key from: /api/retrieve

We found your key!
File successfully decrypted!
```

```
+-----+
|         |
|   +---+ |
|   |   | |
|   +---+ |
|       | |
|   +---+ |
|   |   | |
|   +---+ |
|       | |
|   +---+ |
|   |   | |
|   +---+ |
|         |
+-----+
```

SECRET

```
C:\Users\rootsecdev\Downloads>
```

For giggles I encrypted a few more files to observe the encryption behavior

Encrypted file 1:

Seed = 1577904303

Generated an encryption key: db664fa271df72d2

Secret ID:00c632be-3a7d-4670-a117-1f9b6dc545cb

Encrypted File 2:

Seed = 1577906137

Generate an encryption key: 4085ab7b3363da4

Secret ID:ab90e301-6abd-4745-af88-0b8448cbb0f6

Encrypted File 3:

Seed = 1577906470

Generate an encryption key: 7ffbbbd3b407dbe5

Secret ID: e9e0dca4-497a-4301-b80a-8b792efc5283

If you notice my Seeds issued on each file don't appear to be randomly generated they appear to be possibly timestamp related

At this point I realized I dumped my windows 10 vm that had my python code and the screenshots of the encryption used that was analyzed through Ghidra.



Objective 11 - Open the Sleigh Shop Door (Incomplete)

Visit Shiny Upatree in the Student Union and help solve their problem. What is written on the paper you retrieve for Shiny?

For hints on achieving this objective, please visit the Student Union and talk with Kent Tinseltooth.

Objective 12 - Filter Out Poisoned Sources of Weather Data (Incomplete)

Use the data supplied in the Zeek JSON logs to identify the IP addresses of attackers poisoning Santa's flight mapping software. Block the 100 offending sources of information to guide Santa's sleigh through the attack. Submit the Route ID ("RID") success value that you're given. For hints on achieving this objective, please visit the Sleigh Shop and talk with Wunorse Openslae.

Zeek JSON Logs - <https://downloads.elfu.org/http.log.gz>

SRF Sleigh Route Finder - <https://srf.elfu.org/>

The first issue we have is to find credentials to the SRF website. Naturally we want to look at the Zeek JSON log files for some type of information disclosure that may reveal what these credentials are. First thing to do is clean up the logs. They are formatted in a pretty nasty way. Just look.

```
p, "id.resp_h": "10.20.3.80", "id.resp_p": 80, "trans_depth": 1, "method": "GET", "host": "srf.elfu.org", "uri": "/api/weather?station id=286245", "referrer": "-", "version": "1.1", "user_agent": "Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_6_1; en-US) AppleWebKit/532.2 (KHTML, like Gecko) Chrome/4.0.222.4 Safari/532.2", "origin": "-", "request_body_len": 0, "response_body_len": 482, "status_code": 200, "status_msg": "OK", "info_code": "-", "info_msg": "-", "tags": "(empty)", "username": "-", "password": "-", "proxied": "-", "orig_fuids": "-", "orig_filenames": "-", "orig_mime_types": "-", "resp_fuids": "Ffgrwr4cLYDiakKeyl", "resp_filenames": "-", "resp_mime_types": "application/json"}, {"ts": "2019-11-13T02:31:30-0700", "uid": "Cs3CN017TndDe6JX9g", "id.orig_h": "80.108.48.154", "id.orig_p": 53187, "id.resp_h": "10.20.3.80", "id.resp_p": 80, "trans_depth": 1, "method": "GET", "host": "srf.elfu.org", "uri": "/api/stations", "referrer": "-", "version": "1.1", "user_agent": "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2225.0 Safari/537.36", "origin": "-", "request_body_len": 0, "response_body_len": 196875, "status_code": 200, "status_msg": "OK", "info_code": "-", "info_msg": "-", "tags": "(empty)", "username": "-", "password": "-", "proxied": "-", "orig_fuids": "-", "orig_filenames": "-", "orig_mime_types": "-", "resp_fuids": "Faglr381eAS00sc0", "resp_filenames": "-", "resp_mime_types": "application/json"}, {"ts": "2019-11-13T02:31:30-0700", "uid": "c4k0Prxs8Vnqmw9P2", "id.orig_h": "80.81.167.173", "id.orig_p": 53188, "id.resp_h": "10.20.3.80", "id.resp_p": 80, "trans_depth": 1, "method": "GET", "host": "srf.elfu.org", "uri": "/", "referrer": "-", "version": "1.1", "user_agent": "Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1062.0 Safari/536.3", "origin": "-", "request_body_len": 0, "response_body_len": 5095, "status_code": 200, "status_msg": "OK", "info_code": "-", "info_msg": "-", "tags": "(empty)", "username": "-", "password": "-", "proxied": "-", "orig_fuids": "-", "orig_filenames": "-", "orig_mime_types": "-", "resp_fuids": "Fu6RtBVjhjvbkZ9u2", "resp_filenames": "-", "resp_mime_types": "text/html"}, {"ts": "2019-11-13T02:31:30-0700", "uid": "CwzsvC4PQx0XHCmuck", "id.orig_h": "95.147.132.98", "id.orig_p": 53179, "id.resp_h": "10.20.3.80", "id.resp_p": 80, "trans_depth": 3, "method": "GET", "host": "-", "uri": "/js/library-g.js", "referrer": "-", "version": "1.1", "user_agent": "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.17) Gecko/20080703 Mandriva/2.0.0.17-1.1mdv2008.1 (2008.1) Firefox/2.0.0.17", "origin": "-", "request_body_len": 0, "response_body_len": 385623, "status_code": 200, "status_msg": "OK", "info_code": "-", "info_msg": "-", "tags": "(empty)", "username": "-", "password": "-", "proxied": "-", "orig_fuids": "-", "orig_filenames": "-", "orig_mime_types": "-", "resp_fuids": "FjV6Wl4bCCsS2H2AZk", "resp_filenames": "-", "resp_mime_types": "application/javascript"}]
```

To fix this I am going to issue the following command:

```
cat http.log | jq '[]' > new_http.log
```

```
rootsecdev@Ubuntu18:~/Documents$ cat http.log | jq '[]' > new_http.log
rootsecdev@Ubuntu18:~/Documents$
```

The log is readable now. Below is just a snip of how it looks.

```
{
  "ts": "2019-11-13T02:31:30-0700",
  "uid": "CwzsvC4PQx0XHCmuck",
  "id.orig_h": "95.147.132.98",
  "id.orig_p": 53179,
  "id.resp_h": "10.20.3.80",
  "id.resp_p": 80,
  "trans_depth": 3,
  "method": "GET",
  "host": "-",
  "uri": "/js/library-g.js",
  "referrer": "-",
  "version": "1.1",
  "user_agent": "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.17) Gecko/20080703 Mandriva/2.0.0.17-1.1mdv2008.1 (2008.1) Firefox/2.0.0.17",
  "origin": "-",
  "request_body_len": 0,
  "response_body_len": 385623,
  "status_code": 200,
  "status_msg": "OK",
  "info_code": "-",
  "info_msg": "-",
  "tags": "(empty)",
  "username": "-",
  "password": "-",
  "proxied": "-",
  "orig_fuids": "-",
  "orig_filenames": "-",
  "orig_mime_types": "-",
  "resp_fuids": "FjV6Wl4bCCsS2H2AZk",
  "resp_filenames": "-",
  "resp_mime_types": "application/javascript"
}
```

Next we need to look for website creds. I notice from the above screenshot we can sort entries by status code. Here is handy chart on what status codes mean:

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>

So, we will sort by status code 200 along with unique URI's. URI's can give us clues on full paths that have been accessed in the website.

```
cat new_http.log | jq '. | select (.status_code == 200) | .uri' | sort | uniq
```



```

"/css/alt.css"
"/css/freelancer.min.css"
"/css/main.css"
"/css/weathermap.css"
"/home.html"
"/img/badweather.png"
"/img/goodweather.png"
"/img/logo_zoomed2.PNG"
"/index.html"
"/js/CustomEase.js"
"/js/freelancer.min.js"
"/js/ipaddr.js"
"/js/library-g.js"
"/js/Morph.js"
"/js/weathermap.js"
"/logout"
"/logout?id=1' UNION/**/SELECT 1223209983/*"
"/logout?id=1' UNION SELECT null,null,'autosc','autoscanner',null,null,null,null,null,null,null,null/*"
"/logout?id=<script>alert(1400620032)</script>&ref_a=avdsscanning\\\\"><script>alert(1536286186)</script>"
"/map.html"
"/README.md"
"/santa.html"
"/vendor/bootstrap/js/bootstrap.bundle.min.js"
"/vendor/fontawesome-free/css/all.min.css"
"/vendor/fontawesome-free/webfonts/fa-solid-900.woff2"
"/vendor/jquery-easing/jquery.easing.min.js"
"/vendor/jquery/jquery.min.js"
rootsecdev@Ubuntu18:~/Documents$ cat new_http.log | jq '. | select (.status_code == 200) | .uri' | sort | uniq

```

In the above screenshot we see a /README.md so we go to the following URL:

<https://srf.elfu.org/README.md>



```

# Sled-O-Matic - Sleigh Route Finder Web API

### Installation

...

sudo apt install python3-pip
sudo python3 -m pip install -r requirements.txt
...

#### Running:

`python3 ./srfweb.py`

#### Logging in:

You can login using the default admin pass:

`admin 924158F9522B3744F5FCD4D10FAC4356`

However, it's recommended to change this in the sqlite db to something custom.

```

We now have the username and password for the website. No hacking required!

Next we need to search for threat actors doing the following:

SQLi

XSS

Shellcode

LFI

The following script will detect all IP's related to the following attacks noted above. I got lazy at this point and renamed my new_http.log to http.log:

SQLi

```
cat http.log | jq '. | select (.username | contains("''")) | .id.orig_h'
```

```
cat http.log | jq '. | select (.uri | contains("''")) | .id.orig_h'
```

```
cat http.log | jq '. | select (.user_agent | contains("''")) | .id.orig_h'
```

XSS

```
cat http.log | jq '. | select (.uri | contains("<")) | .id.orig_h'
```

```
cat http.log | jq '. | select (.host | contains("<")) | .id.orig_h'
```

LFI

```
cat http.log | jq '. | select (.uri | contains("pass")) | .id.orig_h'
```

Shellshock

```
cat http.log | jq '. | select (.user_agent | contains("; ;")) | .id.orig_h'
```

Next lets sort by unique IPs. I dumped all the IP's from above into a file called IP.log

Sort Unique IP's

```
sort IP.log | uniq
```

We get approx. 75 IP's from these searches which is 25 short of 100. So, we will need to look further into the data.

This code is not going to look pretty but we are going to look at the origin hosts of every single unique IP we found by doing or statements.

```
cat http.log | jq '. | select(.id.orig_h | contains("56.5.47.137") or contains("19.235.69.221") or  
contains("69.221.145.150") or contains("42.191.112.181") or contains("48.66.193.176") or  
contains("49.161.8.58") or contains("84.147.231.129") or contains("44.74.106.131") or  
contains("106.93.213.219") or contains("42.103.246.250") or contains("2.230.60.70") or  
contains("10.155.246.29") or contains("225.191.220.138") or contains("75.73.228.192") or  
contains("249.34.9.16") or contains("27.88.56.114") or contains("238.143.78.114") or  
contains("121.7.186.163") or contains("106.132.195.153") or contains("129.121.121.48") or
```



```
contains("190.245.228.38") or contains("34.129.179.28") or contains("135.32.99.116") or  
contains("2.240.116.254") or contains("45.239.232.245") or contains("31.254.228.4") or  
contains("220.132.33.81") or contains("83.0.8.119") or contains("150.45.133.97") or  
contains("229.229.189.246") or contains("227.110.45.126") or contains("33.132.98.193") or  
contains("84.185.44.166") or contains("254.140.181.172") or contains("150.50.77.238") or  
contains("68.115.251.76") or contains("118.196.230.170") or contains("173.37.160.150") or  
contains("81.14.204.154") or contains("135.203.243.43") or contains("186.28.46.179") or  
contains("13.39.153.254") or contains("111.81.145.191") or contains("0.216.249.31") or  
contains("52.39.201.107") or contains("102.143.16.184") or contains("230.246.50.221") or  
contains("131.186.145.73") or contains("253.182.102.55") or contains("1.185.21.112") or  
contains("229.133.163.235") or contains("194.143.151.224") or contains("23.49.177.78") or  
contains("75.215.214.65") or contains("223.149.180.133") or contains("211.229.3.254") or  
contains("250.51.219.47") or contains("187.178.169.123") or contains("180.57.20.247") or  
contains("116.116.98.205") or contains("9.206.212.33") or contains("79.198.89.109") or  
contains("25.80.197.172") or contains("193.228.194.36") or contains("169.242.54.5") or  
contains("28.169.41.122") or contains("233.74.78.199") or contains("132.45.187.177") or  
contains("61.110.82.125") or contains("65.153.114.120") or contains("123.127.233.97") or  
contains("95.166.116.45") or contains("80.244.147.207") or contains("168.66.108.62") or  
contains("200.75.228.240"))' >> IP_requests.log
```

Terminal Challenges:

Bushy Evergreen (Exiting the Ed. Editor) Train Station

This challenge involved exiting out of the Ed. Editor where Bushy Evergreen is stuck. This challenge was easier than throwing Han's Grueber out the window in Die Hard.

First off, a refresher, on what the Ed Editor is. It's an old school editor somewhat like VI. A helpful little hint on Ed the Text editor from Bushy Evergreen. The URL is located here:

http://cs.wellesley.edu/~cs249/Resources/ed_is_the_standard_text_editor.html

As we can see from the documentation a standard "q" is all that is needed to exit this editor.

```

.....
.;oooooooooooooooool;,,,,,,:looooooooooooooooooll:
.:oooooooooooooooooc;,,,,,,:ooooooooooooooooolloo:
.';;;;;;;;;;;;;;;';;;;;;;;;;;;;;;ooooo:
.';;;;;;;;;;;;;;;';;;;;;;;;;;;;;;'ooooo:
;oooooooooooooooool;';;;;;;;;;:looooooooooooooooolc;';;ooooo:
.:oooooooooooooooooc;';;;;;;;;;:oooooooooooooooolccoc;,,;ooooo:
.coooooooooooooooo:;';;;;;;;;;:oooooooooooooooolcllooc;,,;ooooo,
cooooooooooooooooo,,,,,,;oooooooooooooooolooooooc;,,;ooo,
cooooooooooooooooo,,,,,,;oooooooooooooooolooooooc;,,;l'
cooooooooooooooooo,,,,,,;oooooooooooooooolooooooc;,,.
cooooooooooooooooo,,,,,,;oooooooooooooooolooooooc.
cooooooooooooooooo,,,,,,;ooooooooooooooooloooo:.
cooooooooooooooooo,,,,,,;ooooooooooooooooloo;
:llllllllllllll,';;;;;;;;;:lllllllllllllllc,

```

Oh, many UNIX tools grow old, but this one's showing gray.
That Pepper LOLs and rolls her eyes, sends mocking looks my way.
I need to exit, run - get out! - and celebrate the yule.
Your challenge is to help this elf escape this blasted tool.

-Bushy Evergreen

Exit ed.

1100

q

Loading, please wait.....

You did it! Congratulations!

elf@14647f858f73:~\$

PATH sugar plum mary (Hermey Hall)

This challenge involves an issue on directory listing where an “ls” command doesn’t work. There is clearly a PATH issue with the “ls” command that is in the path. To break down this challenge we need to first look at the path of the elf login. This can be done by issuing the command:

```
echo $PATH
```

```
I need to list files in my home/
To check on project logos
But what I see with ls there,
Are quotes from desert hobos...

which piece of my command does fail?
I surely cannot find it.
Make straight my path and locate that-
I'll praise your skill and sharp wit!

Get a listing (ls) of your current directory.
elf@e49a0aba2c88:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
elf@e49a0aba2c88:~$
```

To compare path's I notice the container we are running in is a Debian 10 based OS.

```
elf@e49a0aba2c88:~$ cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 10 (buster)"
NAME="Debian GNU/Linux"
VERSION_ID="10"
VERSION="10 (buster)"
VERSION_CODENAME=buster
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
elf@e49a0aba2c88:~$
```

I downloaded Debian 10 buster and installed in a VM environment. In my VM environment I able to compare path's on default installs (<https://cdimage.debian.org/debian-cd/current/amd64/iso-cd/>):

```
rootsecdev@debian:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
rootsecdev@debian:~$ _
```

So unfortunately, there is nothing out of the ordinary with the path between the terminal challenge and the default install of Debian 10 in a vm. So next we will need to look at file structures in each path on the terminal.

One trick to list files in a directory when "ls" isn't available is doing the following command inside a directory:

```
echo *
```

More info can be found about this on the following website: <https://www.ostechnix.com/different-ways-to-list-directory-contents-without-using-ls-command/>

```
elf@e49a0aba2c88:/usr/local/bin$ echo *  
ls  
elf@e49a0aba2c88:/usr/local/bin$
```

By process of elimination lets look at the file in /usr/local/bin on a default install of Debian

```
rootsecdev@debian:~$ cd /usr/local/bin  
rootsecdev@debian:/usr/local/bin$ ls  
rootsecdev@debian:/usr/local/bin$ _
```

Since /usr/local/bin is in the first part of the path this is more than likely the ls command that is not allowing us to do directory listings.

A basic google query on removing directory paths led me to this article:

<https://unix.stackexchange.com/questions/108873/removing-a-directory-from-path>

To drop the path you need to just exclude /usr/local/bin with the following command:

```
elf@9b87274c3c1b:/usr/local/bin$ echo $PATH  
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games  
elf@9b87274c3c1b:/usr/local/bin$ export PATH=/usr/bin:/bin:/usr/local/games:/usr/games
```

Now that our path is dropped, we do a simple "ls" command and the challenge has been passed because the correct "ls" command is getting passed which is in the /usr/bin directory.

```
elf@9b87274c3c1b:/usr/local/bin$ ls  
ls  
Loading, please wait.....  
  
You did it! Congratulations!  
elf@9b87274c3c1b:/usr/local/bin$
```

Alabaster Snowball Bash Challenge (Speaker Unpreparedness room)

This is perhaps one of the most basic problems to solve but a lot of people (including me) over thought the challenge. I probably spent a good couple of hours trying to figure out this challenge. Looking to see what kind of bash shell someone runs under in /etc/passwd is a misleading hint. It will drive some folks to look at the shell issue in the wrong way because the elf user doesn't have sudo abilities. Or atleast sudo to do most administrative functions.

Another hint that one dropped was around immutable files.

One useful command to see what sudo rights one does have is with the following command:

`sudo -l`

You can read more about that here:

<https://bencane.com/2011/08/17/sudo-list-available-commands/>

```
elf@f460f19ef5cb:~$ sudo -l
Matching Defaults entries for elf on f460f19ef5cb:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User elf may run the following commands on f460f19ef5cb:
    (root) NOPASSWD: /usr/bin/chatrr
elf@f460f19ef5cb:~$
```

From the above screen we see we can run sudo commands on the chatrr command without using a password for elevation.

For more info on what chatter does you can read the following URL:

<https://linux.die.net/man/1/chatrr>

So lets use the first hint we got which is to look up who gets what type of bash shells.

```
elf@f460f19ef5cb:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534:./nonexistent:/usr/sbin/nologin
elf:x:1000:1000:./home/elf:/bin/bash
alabaster_snowball:x:1001:1001:./home/alabaster_snowball:/bin/nsh
elf@f460f19ef5cb:~$
```

We see the alabaster_snowball has /bin/nsh while the elf we are logged into has /bin/bash. We know that the elf is having no issues and we can assume that alabaster really needs to be in /bin/bash for his shell.

So how can we do this without root access? We need to focus on what we can do without root.

My first attempt was to copy over nsh with bash. This will not work due to the rights on the file.

```
elf@f460f19ef5cb:/bin$ ls
bash      bzless  dir      gzexe    more     rbash    sync      ypdomainname
bunzip2   bzmore  dmesg    gzip     mount    readlink  tar       zcat
bzcat     cat     dnsdomainname  hostname mountpoint  rm        tempfile  zcmp
bzcmp     chgrp   domainname  ip       mv        rmdir     touch     zdiff
bzdiff    chmod   echo       ln        nisdomainname  run-parts true      zegrep
bzegrep   chown   egrep      login    nsh       sed        umount    zfgrep
bzexe     cp      false     ls        pidof     sh         uname     zforce
bzfgrep   dash    fgrep      lsblk    ping      sleep      uncompress  zgrep
bzgrep    date    findmnt    mkdir    ping4     ss         vdir      zless
bzip2     dd      grep       mkdir    ping6     stty       wdctl     zmore
bzip2recover  df      gunzip     mktemp   pwd        su         which     znew
elf@f460f19ef5cb:/bin$ cp bash nsh
cp: cannot create regular file 'nsh': Operation not permitted
elf@f460f19ef5cb:/bin$
```

At this point something occurred to me where I could use chattr to remove the immutable rights from the nsh file. At this point I was able to copy bash to nsh. This effectively overwrites nsh.


```
elf@f460f19ef5cb:/bin$ sudo chatter -i nsh
elf@f460f19ef5cb:/bin$ cp bash nsh
elf@f460f19ef5cb:/bin$
```

We can now successfully switch to alabaster_snowball into the correct shell without changing the contents to /etc/passwd

```
elf@f460f19ef5cb:/bin$ sudo chatter -i nsh
elf@f460f19ef5cb:/bin$ cp bash nsh
elf@f460f19ef5cb:/bin$ su -l alabaster_snowball
Password:
Loading, please wait.....

You did it! Congratulations!

alabaster_snowball@f460f19ef5cb:~$
```

Mongo Pilfer Challenge - Holly Evergreen (Netwars room)

If you want the best hacking music to listen to IMHO visit this room. The final countdown remix is awesome. For full disclosure I know nothing about mongodb. The secret to this challenge is to not over think things and “help” once you are in mongo DB is your friend.

So the elves have been locked out of the mongo db server and can’t get back in to grade papers. First is to launch mongo db.

We quickly find out that Mongo is not running on the default port of 27017:

```
elf@992277169b7d:~$ mongo
MongoDB shell version v3.6.3
connecting to: mongod://127.0.0.1:27017
2019-12-21T14:12:31.980+0000 W NETWORK [thread1] Failed to connect to 127.0.0.1:27017, in(checkin
g socket for error after poll), reason: Connection refused
2019-12-21T14:12:31.980+0000 E QUERY [thread1] Error: couldn't connect to server 127.0.0.1:2701
7, connection attempt failed :
connect@src/mongo/shell/mongo.js:251:13
@(<connect>):1:6
exception: connect failed

Hmm... what if Mongo isn't running on the default port?

elf@992277169b7d:~$
```

We got a hint earlier that lsof -l is not available so we will need to look up ports in a different fashion. I see that netstat is available to us:


```

elf@7d7f1a8cba9b:~$ netstat -help
usage: netstat [-vWeenNcCF] [<Af>] -r          netstat {-V|--version|-h|--help}
       netstat [-vWnNcaeol] [<Socket> ...]
       netstat { [-vWeenNac] -i | [-cnNe] -M | -s [-6tuw] }

    -r, --route           display routing table
    -i, --interfaces     display interface table
    -g, --groups         display multicast group memberships
    -s, --statistics     display networking statistics (like SNMP)
    -M, --masquerade     display masqueraded connections

    -v, --verbose         be verbose
    -W, --wide           don't truncate IP addresses
    -n, --numeric        don't resolve names
    --numeric-hosts      don't resolve host names
    --numeric-ports      don't resolve port names
    --numeric-users      don't resolve user names
    -N, --symbolic       resolve hardware names
    -e, --extend         display other/more information
    -p, --programs       display PID/Program name for sockets
    -o, --timers         display timers
    -c, --continuous     continuous listing

    -l, --listening      display listening server sockets
    -a, --all            display all sockets (default: connected)
    -F, --fib            display Forwarding Information Base (default)
    -C, --cache          display routing cache instead of FIB
    -Z, --context        display SELinux security context for sockets

<Socket>={-t|--tcp} {-u|--udp} {-U|--udplite} {-S|--sctp} {-w|--raw}
           {-x|--unix} --ax25 --ipx --netrom
<AF>=Use '-6|-4' or '-A <af>' or '--<af>'; default: inet
List of possible address families (which support routing):
inet (DARPA Internet) inet6 (IPv6) ax25 (AMPR AX.25)
netrom (AMPR NET/ROM) ipx (Novell IPX) ddp (Appletalk DDP)
x25 (CCITT X.25)

```

To take a closer look I will issue the command `netstat -aut`

To breakdown the variables we are basically asking netstat to list to all connections over tcp/udp

```

elf@7d7f1a8cba9b:~$ netstat -aut
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 localhost:12121         0.0.0.0:*               LISTEN
elf@7d7f1a8cba9b:~$

```

The port mongo is more than likely listening on is port 12121 as seen in our netstate. To connect to mongo under a different port we can find out how to do so on the following documentation page for mongo db: <https://docs.mongodb.com/manual/mongo/>

```

elf@7d7f1a8cba9b:~$ mongo --port 12121
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:12121/
MongoDB server version: 3.6.3
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
    http://docs.mongodb.org/
Questions? Try the support group
    http://groups.google.com/group/mongodb-user
Server has startup warnings:
2019-12-21T14:14:13.229+0000 I CONTROL [initandlisten]
2019-12-21T14:14:13.229+0000 I CONTROL [initandlisten] ** WARNING: Access control is not enabled
for the database.
2019-12-21T14:14:13.229+0000 I CONTROL [initandlisten] **           Read and write access to data
and configuration is unrestricted.
2019-12-21T14:14:13.229+0000 I CONTROL [initandlisten]
2019-12-21T14:14:13.230+0000 I CONTROL [initandlisten]
2019-12-21T14:14:13.230+0000 I CONTROL [initandlisten] ** WARNING: /sys/kernel/mm/transparent_hug
epage/enabled is 'always'.
2019-12-21T14:14:13.230+0000 I CONTROL [initandlisten] **           We suggest setting it to 'never'
2019-12-21T14:14:13.230+0000 I CONTROL [initandlisten]
>

```

Now that we have connected to what is next. I decided to throw the customary help command to get me going.

```

> help
db.help()                help on db methods
db.mycoll.help()         help on collection methods
sh.help()                sharding helpers
rs.help()                replica set helpers
help admin               administrative help
help connect             connecting to a db help
help keys                key shortcuts
help misc                misc things to know
help mr                  mapreduce

show dbs                 show database names
show collections          show collections in current database
show users               show users in current database
show profile             show most recent system.profile entries with time >= 1ms
show logs                show the accessible logger names
show log [name]          prints out the last segment of log in memory, 'global' is default

use <db_name>            set current database
db.foo.find()             list objects in collection foo
db.foo.find( { a : 1 } )  list objects in foo where a == 1
it                       result of the last line evaluated; use to further iterate
DBQuery.shellBatchSize = x set default number of items to display on shell
exit                     quit the mongo shell
>

```

So lets show all the databases in the mongo db server:

```
> show dbs
admin    0.000GB
config  0.000GB
elfu     0.000GB
local    0.000GB
test     0.000GB
>
```

There are only 5 databases. By process of the elimination we more than likely want the elfu database. So lets set the current database to that:

```
> show dbs
admin    0.000GB
elfu     0.000GB
local    0.000GB
test     0.000GB
> use elfu
switched to db elfu
>
```

Next we will execute show collections. This will show all collections in the current database we have selected. This command is also in the help file we executed earlier.

```
> show collections
bait
chum
line
metadata
solution
system.js
tackle
tincan
>
```

Obviously we want to search in the solution collection. Again with the helpfile, we can type `db.solution.find()` to list all objects in the solution connection. Next, and as you see below, we are given a really nice hint to display the solution with the next command we need to run:

```
> db.solution.find()
{ "_id" : "You did good! Just run the command between the stars: ** db.loadServerScripts();displaySolution(); **" }
>
```

The following command can be issued next:

```
db.loadServerScripts();displaySolution();
```

At this point as you see in the next screenshot we have solved this issue:

```
switched to db elfu  
> db.solution.find()  
{ "_id" : "You did good! Just run the command between the stars: ** db.loadServerScripts();display  
Solution(); **" }  
> db.loadServerScripts();displaySolution();
```



IP Tables Challenge – Kent Tensletooth (Student Union)

Whatever Kent has done by hooking up a linux device to his teeth... it sounds painful. This is very much a timed challenge and at some point Kent will give up and you have to start over again... which is not fun.

```
elfuuser@a79a5146f87a:~$ cat IOTeethBraces.md
# ElFU Research Labs - Smart Braces
### A Lightweight Linux Device for Teeth Braces
### Imagined and Created by ElFU Student Kent TinselTooth

This device is embedded into one's teeth braces for easy management and monitoring of dental status. It uses FTP and HTTP for management and monitoring purposes but also has SSH for remote access. Please refer to the management documentation for this purpose.

## Proper Firewall configuration:

The firewall used for this system is `iptables`. The following is an example of how to set a default policy with using `iptables`:

...
sudo iptables -P FORWARD DROP
...

The following is an example of allowing traffic from a specific IP and to a specific port:

...
sudo iptables -A INPUT -p tcp --dport 25 -s 172.18.5.4 -j ACCEPT
...

A proper configuration for the Smart Braces should be exactly:

1. Set the default policies to DROP for the INPUT, FORWARD, and OUTPUT chains.
2. Create a rule to ACCEPT all connections that are ESTABLISHED,RELATED on the INPUT and the OUTPUT chains.
3. Create a rule to ACCEPT only remote source IP address 172.19.0.225 to access the local SSH server (on port 22).
4. Create a rule to ACCEPT any source IP to the local TCP services on ports 21 and 80.
5. Create a rule to ACCEPT all OUTPUT traffic with a destination TCP port of 80.
6. Create a rule applied to the INPUT chain to ACCEPT all traffic from the lo interface.
elfuuser@a79a5146f87a:~$
```

I prefer to take my time with this approach. So the easiest way to replicate a clean iptables firewall rule is to spin up an ubuntu 18.04 server in your favorite hypervisor. As you notice above, I took a screenshot of some example iptables commands and the order that the iptables firewall configuration needs to be in for this challenge. This info is in the IOTeethBraces.md file.

Before we tackle the problem in full, we should look at the iptables current configuration by executing the command:

```
sudo iptables -L
```

```
elfuuser@0249cafc0aad:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
elfuuser@0249cafc0aad:~$
```

Problem 1:

Set the default policies to DROP for the INPUT, FORWARD, and OUTPUT chains.

The first problem is practically given to us since one of the commands for FORWARD was given in the IOTeethBraces.md file. It's a good idea to start dropping everything then build up the rules that you need. The following site also contained some really good overview of rules including the ones for the 1st solution: <http://linux-training.be/networking/ch14.html>

Solution for 1:

```
sudo iptables -P INPUT DROP
```

```
sudo iptables -P FORWARD DROP
```

```
sudo iptables -P OUTPUT DROP
```

After running the commands for solution number 1 your iptables should look like below after running

```
sudo iptables -L
```

```
Chain INPUT (policy DROP)
target     prot opt source                destination

Chain FORWARD (policy DROP)
target     prot opt source                destination

Chain OUTPUT (policy DROP)
target     prot opt source                destination
```


Problem 2

Create a rule to ACCEPT all connections that are ESTABLISHED,RELATED on the INPUT and the OUTPUT chains.

We were giving a hint for this challenge that pointed us to the following URL:

<https://upcloud.com/community/tutorials/configure-iptables-centos/>

An example of this config is located in the URL above for the INPUT connection. We just need to apply this to the OUTPUT connection as well.

To begin using iptables, you should first add the rules for allowed inbound traffic for the services you require. Iptables can track the state of the connection, so use the command below to allow established connections continue.

```
sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

Solution for 2:

```
sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

```
sudo iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

After inputting these rules our iptables should look at follows:

```
Chain INPUT (policy DROP)
target    prot opt source                destination            ctstate RELATED,ESTABLISHED
ACCEPT    all  --  anywhere              anywhere

Chain FORWARD (policy DROP)
target    prot opt source                destination

Chain OUTPUT (policy DROP)
target    prot opt source                destination            ctstate RELATED,ESTABLISHED
ACCEPT    all  --  anywhere              anywhere
```

Problem 3:

Create a rule to ACCEPT only remote source IP address 172.19.0.225 to access the local SSH server (on port 22).

This is a good rule to put in place for SSH access. The problem above will only allow SSH from the device that carries the IP 172.19.0.225. If we look at our hint URL we will see an output command example that does this very thing except for all of SSH.

Next, allow traffic to a specific port to enable SSH connections with the following.

```
sudo iptables -A INPUT -p tcp --dport ssh -j ACCEPT
```

This command completes most of what we need to do with the exception of allowing SSH from only one source IP address. A `-s 172.19.0.225` is needed.

Solution for 3:

```
sudo iptables -A INPUT -p tcp --dport 22 -s 172.19.0.225 -j ACCEPT
```

After doing the above command our iptables should look like this.

```
Chain INPUT (policy DROP)
target     prot opt source                destination           ctstate RELATED,ESTABLISHED
ACCEPT     all  --  anywhere              anywhere              tcp dpt:ssh

Chain FORWARD (policy DROP)
target     prot opt source                destination

Chain OUTPUT (policy DROP)
target     prot opt source                destination           ctstate RELATED,ESTABLISHED
ACCEPT     all  --  anywhere              anywhere
```

Problem 4:

Create a rule to ACCEPT any source IP to the local TCP services on ports 21 and 80.

This is easy enough since we already allowed this for SSH with the exception from a specific IP address. Its important to note that allowing access over ports 21, and 80 are unencrypted connections to FTP and HTTP respectively and should never be done on a production system.

Solution for 4:

```
sudo iptables -A INPUT -p tcp --dport 21 -j ACCEPT
```

```
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

Our iptables should not look like the following:

```
Chain INPUT (policy DROP)
target     prot opt source                destination          ctstate RELATED,ESTABLISHED
ACCEPT     all  --  anywhere              anywhere             tcp dpt:ssh
ACCEPT     tcp  --  172.19.0.225         anywhere            tcp dpt:ftp
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:http

Chain FORWARD (policy DROP)
target     prot opt source                destination

Chain OUTPUT (policy DROP)
target     prot opt source                destination          ctstate RELATED,ESTABLISHED
ACCEPT     all  --  anywhere             anywhere
```

Problem 5:

Create a rule to ACCEPT all OUTPUT traffic with a destination TCP port of 80.

This problem is pretty much the same as problems 3 and 4 except will work on the OUTPUT command instead of the INPUT command.

Solution for 5:

```
sudo iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
```

After doing this command our iptables should look like the following:

```
Chain INPUT (policy DROP)
target     prot opt source                destination          ctstate RELATED,ESTABLISHED
ACCEPT     all  --  anywhere              anywhere             tcp dpt:ssh
ACCEPT     tcp  --  172.19.0.225         anywhere            tcp dpt:ftp
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:http

Chain FORWARD (policy DROP)
target     prot opt source                destination

Chain OUTPUT (policy DROP)
target     prot opt source                destination          ctstate RELATED,ESTABLISHED
ACCEPT     all  --  anywhere             anywhere
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:http
```

Problem 6

Create a rule applied to the INPUT chain to ACCEPT all traffic from the lo interface

This url helped me creating the command for this: <http://linux-training.be/networking/ch14.html>

Specifically, this section:

```
[root@RHEL5 ~]# iptables -A INPUT -i lo -j ACCEPT
[root@RHEL5 ~]# iptables -A OUTPUT -o lo -j ACCEPT
```

Since we only need to do this on the INPUT chain we only need to do the top command.

Solution to 6:

```
sudo iptables -A INPUT -i lo -j ACCEPT
```

After doing our above command the iptables should look like the following:

```
Chain INPUT (policy DROP)
target     prot opt source                destination          ctstate RELATED,ESTABLISHED
ACCEPT     all  --  anywhere              anywhere             tcp dpt:ssh
ACCEPT     tcp  --  172.19.0.225          anywhere             tcp dpt:ftp
ACCEPT     tcp  --  anywhere              anywhere             tcp dpt:http
ACCEPT     all  --  anywhere              anywhere

Chain FORWARD (policy DROP)
target     prot opt source                destination

Chain OUTPUT (policy DROP)
target     prot opt source                destination          ctstate RELATED,ESTABLISHED
ACCEPT     all  --  anywhere              anywhere             tcp dpt:http
```

Now that I individually took you through step by step on this challenge lets do a quick run though since this is a timed challenge. As I stated earlier its easy to stand up a ubuntu server and play with iptables in your own environment to map out the solutions for this challenge.

```
Kent TinselTooth: I suspect someone may have hacked into my IOT teeth braces.
Kent TinselTooth: I must have forgotten to configure the firewall...
Kent TinselTooth: Please review /home/elfuuser/IOTteethBraces.md and help me configure the firewall.
Kent TinselTooth: Please hurry; having this ribbon cable on my teeth is uncomfortable.
elfuuser@1b3fd6f294a1:~$
elfuuser@1b3fd6f294a1:~$
elfuuser@1b3fd6f294a1:~$ sudo iptables -P INPUT DROP
elfuuser@1b3fd6f294a1:~$ sudo iptables -P FORWARD DROP
elfuuser@1b3fd6f294a1:~$ sudo iptables -P OUTPUT DROP
elfuuser@1b3fd6f294a1:~$ sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
elfuuser@1b3fd6f294a1:~$ sudo iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
elfuuser@1b3fd6f294a1:~$ sudo iptables -A INPUT -p tcp --dport 22 -s 172.19.0.225 -j ACCEPT
elfuuser@1b3fd6f294a1:~$ sudo iptables -A INPUT -p tcp --dport 21 -j ACCEPT
elfuuser@1b3fd6f294a1:~$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
elfuuser@1b3fd6f294a1:~$ sudo iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
elfuuser@1b3fd6f294a1:~$ sudo iptables -A INPUT -i lo -j ACCEPT
elfuuser@1b3fd6f294a1:~$ Kent TinselTooth: Great, you hardened my IOT Smart Braces firewall!

/usr/bin/inits: line 10:      17 Killed                  su elfuuser
```