sl3igh_bells.xmas

A Mr. Robot Holiday Special and KringleCon 2 Writeup

written by

Will Springer

INT. ELLIOT'S APARTMENT - NIGHT

ELLIOT sits at his desk. He is in the zone, staring intensely at the screen. He types out code into the text editor on his screen.

We hear CHRISTMAS MUSIC muffled from out on the street.

ELLIOT (V.O.)
At night, distractions get pushed to the corners of your mind. Night makes it easier to think, to connect directly to the machine.

He makes a mistake typing.

ELLIOT (V.O.) (CONT'D)
Until your bandwidth gets throttled by reality.

MR. ROBOT sits across the apartment.

MR. ROBOT
(calling over to Elliot)
Time to go to bed, kid.

Elliot takes a deep breath. Runs his hand over his head.

ELLIOT (V.O.)
Humans, like computers, crash when our resources are exhausted. If only we could free up memory to keep going.

MONTAGE - ELLIOT GETS READY FOR BED

- Elliot brushes his teeth in the bathroom.

- Elliot sets out new water for FLIPPER.

- Elliot hops into bed, Flipper curling up next to him.

END OF MONTAGE

Elliot turns off the light. Lays on his back. Closes eyes.

Opens eyes. Stares into the camera.

ELLIOT (V.O.)
I can't sleep.

Closes eyes.

ELLIOT (V.O.) (CONT'D)
I envy the computer's ability to shut down.

ELLIOT
(to himself)
One... two... three... four...

ELLIOT (V.O.)
There has to be an easier way to do this without counting sheep.

CHRISTMAS MUSIC playing "Rudolf the Red-Nosed Reindeer".

ELLIOT (V.O.) (CONT'D)
(sleepy)
Counting reindeer...

INT. ELF UNIVERSITY - TRAIN STATION - DAY

Elliot steps out of a red steam train onto the platform.

SANTA stumbled up to Elliot frantically.

SANTA
Mr. Alderson. Thank goodness you're here.

ELLIOT (V.O.)
What?

SANTA
They're gone! All gone!

Santa walks hastily toward the quad. Elliot follows him.

EXT. ELF UNIVERSITY - QUAD - DAY

Santa walks quickly through the quad, looking around.

SANTA
Now, normally I wouldn't call Allsafe for something so trivial, but my darling two turtle doves are missing and I have no idea where they got off too.

Santa continues looking around.

SANTA (CONT'D)
Please. Anything you can do to help me find them would be so appreciated.

Elliot glances around. He squats down and examines some bird prints in the snow.

ELLIOT
Maybe these?

Elliot follows the footprints through a door.

INT. ELF UNIVERSITY - STUDENT UNION - DAY

Elliot walks into the Student Union. There sit the two turtle doves MICHAEL and JANE next to the fireplace, basking in the warmth.

ELLIOT
There you are.

He stretches out his hands and they hop on his hand.

He carries them away.

EXT. ELF UNIVERSITY - QUAD - DAY

ELLIOT
(calling)
Hey, hey Santa. I found them.

SANTA
Oh thank goodness!

Santa jogs up to them.

SANTA (CONT'D)
There you two are. Never run off like that again. You scared the Christmas spirit right out of me! Common, let me take you home—

An EXPLOSION blows a hole in the side of the Student Union. Debris flies everywhere. Santa and Elliot and knocked to the ground. The turtle doves fly away.

Elliot's ears RING.

SANTA (CONT'D)
(distant)
Elliot?! Elliot?! Are you okay?!

Elliot stares in shock.

SANTA (CONT'D)
Elliot!?

ELLIOT
Yeah, I think so.

He runs his hand across a gash on his face. Looks at his hand. There is a small amount of blood on it.

An elf, WUNORSE OPENSLAE, runs up to them.

WUNORSE
Santa! Santa! We're under attack!

SANTA
What's going on?

WUNORSE
It looks like we were hit with a piece of industrial control malware. It built up pressure in the steam tunnels and caused the explosion. We had just received a threat.

Wunorse holds up his phone.

The screen reads: Uh, oh! It looks like you forgot to floss and now your systems have gingivitis! You can pay 5 smilecoin to fix your computers, but it won't do anything. -WannaSpry

WUNORSE (CONT'D)
What are we going to do? There was a letter attached to the email, but it was blacked out.

ELLIOT
Can I see?

Elliot takes the phone from Wunorse. Opens up the attached letter.

He presses his finger on the text.

ELLIOT (CONT'D)
The redaction was done on a different layer of the PDF. They should have merged the layers, which would have prevented you from getting access to the next underneath.

Elliot highlights the text and pasts it to a new document. The text reads:

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

ELLIOT (CONT'D)
Someone has a bone to pick with you, Santa.

SANTA
But I don't understand... it's the holiday season...

WUNORSE
(to Elliot)
Hey you...

ELLIOT
Elliot.

WUNORSE
Elliot. Can you help us respond to this? Isn't this what you do at Allsafe?

ELLIOT
I haven't done much ICS work, but I'm happy to help in whatever way I can.

WUNORSE
I'll take you to the SOC.

INT. ELF UNIVERSITY - SECURITY OPERATIONS CENTER - DAY

A row of elves look at data on their screens, sorting through data. There are big screens at the front of the room like at NASA. They are talking frantically to one another.

HOLLY EVERGREEN comes up to Elliot and Wunorse.

HOLLY
It's been a KRINGLing mess in here. We're getting hit over all of our systems. This must be a nation state or something. No way this is the Abominable Snowman flying solo again.

WUNORSE
I brought reinforcements. Holly, meet Elliot. He does incident response.

HOLLY
Glad to have you helping out.

Holly scans the row of elves. Points at KENT TINSELTOOTH.

HOLLY (CONT'D)
(at Kent)
Kent, hop off and let Elliot take a look.

Kent jumps up. Elliot sits down at his desk. Kent pulls up a chair next to him.

ELLIOT
What do we have?

KENT
I'm supposed to figure out how the attackers got in. I have a set of event logs.

ELLIOT
Let me take a look.

Elliot opens the Security.evtx file on Kent's computer in Windows Event Viewer. Elliot scans the list of events.

KENT
From what I can tell, it looks like someone tried to run a password spray attack against our domain controller.
(MORE)

KENT (CONT'D)
There are so many logon attempts with a failure reason of "Unknown user name or bad password".

Elliot sorts the events by time.

ELLIOT (V.O.)
After all of the failures, there's a "credential validation" entry? That's our culprit.

ELLIOT
Looks like the account they got access to was "**supatree**".

Kent jots down the note.

ELLIOT (CONT'D)
After they got access, they likely would have dumped the domain hashes. Do you have the Sysmon data?

KENT
I think so.

Kent hops on the computer and retrieves the file.

KENT (CONT'D)
Here.

Elliot takes control of the computer.

ELLIOT (V.O.)
ntdsutil can be used to make a backup of the domain hashes. They might have gone after it to retrieve the hashes.

Elliot runs: eql query -f sysmon-data.json "process where process_name == 'ntdsutil.exe'" | jq

The output reads: "command_line": "ntdsutil.exe \"ac i ntds\" ifm \"create full c:**hive**\" q q"

ELLIOT
Hive. That must be what they used to steal the hashes.

KENT
Got it.

Holly walks over.

HOLLY
Figured it out?

KENT
Yeah, Elliot got it.

HOLLY
Well, we're still not out of the fire wood yet. We'll need you to help us out on the network side too.

Holly walks over to another desk with PEPPER MINSTIX seated at it.

HOLLY (CONT'D)
Pepper, Elliot's here to help out.

PEPPER
But I am working on it...

HOLLY
Time's crucial. Please, let him work on it with you.

PEPPER
Fine. But I'm driving.

Holly looks at Elliot.

ELLIOT
That's fine.

PEPPER
I'm looking at some Zeek logs. We think they installed some malware on the compromised systems after they gained access. But they were pretty sneaky with it by not mass installing it.

ELLIOT
Did you already take a look at the logs in RITA?

PEPPER
What's that?

ELLIOT
It looks for trends in Zeek logs.

Pepper opens her web browser, finds RITA, and installs it. She starts clicking through the tabs.

ELLIOT (CONT'D)
Go to the "Beacons" tab.

Holly goes to the Beacons tab.

One entry has a correlation score of 0.998 with a source IP of **192.168.134.130.**

ELLIOT (V.O.)
That high correlation. That must be it. It indicates calling back to the C2 server. The small variation in likely due to network jitter.

ELLIOT
There.

PEPPER
(calling over to Holly)
You see that? We got it, even with me driving. Learn to give me some credit.

Kent walks over.

KENT
Hey Elliot, I need your help again.

Elliot slides his chair over to Kent's desk.

KENT (CONT'D)
I'm trying to figure out how the malware Pepper saw was downloaded. First, I took a look a the Powershell activity on the box.

Kent opens up Splunk and shows the query: index=main sourcetype="WinEventLog:Microsoft-Windows-Powershell/Operational"

KENT (CONT'D)
I saw there was one burst of activity, so I tried to track what was going on at that time.

He clicks the time at the start of the cluster adds "| reverse" to the query and adjusts time to plus/minus 5 seconds of the activity.

ELLIOT
Take a look at the other activity on the box around that time.

Kent removes the filters.

ELLIOT (CONT'D)
Sysmon would probably be a good place to look.

Elliot takes control of the computer. He selects the sourcetype of "XmlWinEventLog:Microsoft-Windows-Sysmon/Operational". Shows two process_id values of 6268 and 5864.

ELLIOT (V.O.)
I need to convert those PIDs to hex to be able to search for Windows Process Execution events.

Elliot converts them to hex in using an online hex calculator. Output is 0x187c and 0x16e8. He runs: index=main sourcetype=WinEventLog (0x187c OR 0x16e8).

A result contains the following: 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 ""

ELLIOT (V.O.) (CONT'D)
Must be our culprit dropper file.

ELLIOT
Do you store the file contents anywhere?

KENT
Yeah. We parse them using stoQ.

He takes back control of the keyboard and mouse.

KENT (CONT'D)
First, we'll need to reassemble the path to the file in stoQ.

Kent rebuilds the path with: sourcetype=stoq "19th Century Holiday Cheer Assignment.docm" | 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.

Output is: /home/ubuntu/archive/c/6/e/1/7/c6e175f5b8048c771b3a3fac5f3295d2032524af/19th Century Holiday Cheer Assignment.docm

He pulls up the document in the file system using the output path.

It reads: Cleaned for your safety. Happy Holidays! In the real world, This would have been a wonderful artifact for you to investigate, but it had malware in it of course so it's not posted here. Fear not! The core.xml file that was a component of this original macro-enabled Word doc is still in this File Archive thanks to stoQ. Find it and you will be a happy elf :-)

He opens the core.xml file at: /home/ubuntu/archive/f/f/1/e/a/ff1ea6f13be3faabd0da728f514deb7fe3577cc4/core.xml.

In the XML, there is a message tag that reads: **Kent you are so unfair. And we were going to make you the king of the Winter Carnival.**

KENT (CONT'D)
How... who is this from?

ELLIOT
Must have ticked someone off.

KENT
I... can't think of who that would be.

Holly comes over.

HOLLY
Elliot, can you come take a look at this? We had some unusual activity on the steam tunnel access logs, but we're getting a defect in our camera webapp.

Elliot and Holly walk over to Holly's computer.

HOLLY (CONT'D)
This image from the security camera system isn't loading properly.

Elliot opens up the Chrome developer tools and opens the Network tab. He refreshes the page.

A krampus.png file appears. The picture of someone flashes on the screen, then disappears.

HOLLY (CONT'D)
There! That must be it.

Elliot downloads the image from the Network tab, rotates it, and zooms in. There is a key attached to the figure's belt.

HOLLY (CONT'D)
That's how they got access to the steam tunnel. We need to go check it out, but we lost the master key months ago.

ELLIOT
Here.

Elliot downloads https://github.com/deviantollam/decoding/blob/master/Key%20Decoding/Decoding%20-%20Schlage.png.

He overlays the key decoder over the image of the key.

ELLIOT (CONT'D)
Bitting order is 1-2-2-5-2-0.

HOLLY
We have a key cutter in the maintenance closet. We'll stop by on the way to the steam tunnel entrance.

INT. ELF UNIVERSITY - STEAM TUNNELS - DAY

Elliot and Holly open the door to the steam tunnels. It's dark. Steam sounds come from different parts of the hall.

HOLLY
This is eerie.

A silhouette runs in front of them.

HOLLY (CONT'D)
Whose there?

They walk slowly to where they saw the figure run by. The figure from earlier, KRAMPUS HOLLYFELD, is tucked into the corner.

KRAMPUS
(frightened)
Please... I didn't know... I didn't know anything like this was going to happen.

ELLIOT
Who are you?

KRAMPUS
Krampus Hollyfeld, the steam tunnel apprentice.

HOLLY
Did you have anything to do with the explosion?

KRAMPUS
I'm not sure.

HOLLY
You know something about it?

KRAMPUS
Well... I didn't mean to...

HOLLY
(angrily)
Someone could have been killed!

Krampus pauses for a beat.

KRAMPUS
I got an email from someone. They said they had broken into my computer and they had photos of me while I was changing. They had my password and everything! They said they'd post them on the internet if I didn't help them.

HOLLY
That sounds like a scam. So what did you do?

KRAMPUS
I received a USB stick in the mail. There was a note saying to plug it into the main control computer for the steam tunnels.

HOLLY
(yelling)
And you did it? You didn't report it to security?

KRAMPUS
I was scared! After I plugged it in, it didn't seem like it did anything, so I thought it was fine. All the steam gauges were reading normally when the explosion occurred.

ELLIOT
They probably manipulated the readings on the steam pressure. Do you still have this USB drive?

KRAMPUS
Yeah.

Krampus pulls the drive out of his pocket.

ELLIOT
Holly, do you have a computer I can analyze this with?

HOLLY
Yeah, you can use my laptop.

INT. ELF UNIVERSITY - SECURITY OPERATIONS CENTER - DAY

Elliot looks at the screen. Holly and Krampus are next to him.

ELLIOT (V.O.)
Based on the malware behavior, it appears to have been reaching out to fridosleigh.com/kill.

Elliot goes to fridosleigh.com/kill. Error, page not found.

He goes to fridosleigh.com. There is a CAPTEHA on the page. It times out quickly.

ELLIOT
I think if I solve the CAPTEHA in time, it might create the kill page.

He opens a text editor and cranks out some code.

He runs the code and goes back to fridosleigh.com/kill. Success. Page reads **8Ia8LiZEwvyZr2WO**.

HOLLY
Krampus, is there anything else you can...

Krampus is gone.

HOLLY (CONT'D)
Where did he go? Krampus?

She looks around the room and then walks down the hall. She claps her hand over her mouth in shock.

HOLLY (CONT'D)
Oh!

Krampus lays dead on the floor of the hall, foam still bubbling from his mouth.

ELLIOT (V.O.)
KRINGLE this just got real.

INT. ELF UNIVERSITY - SECURITY OPERATIONS CENTER - LATER

Medics carry Krampus' body out on a stretcher.

HOLLY
Look what I found clenched in his hand.

Holly shows Elliot a piece of paper. There is a website printed on it: studentportal.elfu.org/application-check.php.

HOLLY (CONT'D)
We're going to need to get access to his files.

Elliot opens the website in a browser tab.

He starts typing "test@test.com OR 1=1;--" into the email field. Gets a "not a valid email format" error.

Elliot open Burp and runs the request with test@test.com as the email address. He intercepts the request.

It first goes to "validator.php". He checks the response. There is a token in the response.

He forwards the request. The page then goes to "application-check.php" with "elfmail" and "token" parameters. The token parameter is populated with the token from the previous request.

He swaps out "test@test.com OR 1=1;--" for the "elfmail" value and forwards the request.

He gets an invalid syntax error "near '--".

He opens Burp and creates a new macro. He defines a custom "token" parameter for the "validator.php" response and isolates the responses token.

He sets up the macro to populate the token value on the second request automatically.

He runs: sqlmap -u "https://studentportal.elfu.org/application-check.php?elfmail=test%40test.com&token=test" -p elfmail --proxy=https://localhost:8080 --dump

He opens up a dumped table called "krampu" in the "elfu" database.

In it are a set of file addresses for a set of .pngs. He retrieves the files and opens them to find image fragments of a letter.

He assembles them and sees a letter talking about sabotaging Santa's new **Super Sled-o-matic** guidance system.

HOLLY (CONT'D)
So the steam explosion must have just been a diversion to distract from this sabotage.

ELLIOT
One sec.

Elliot looks through the other dumped databases and sees file called ElfUResearchLabsSuperSledOMaticQuickStartGuideV1.2.pdf.enc. He downloads it.

ELLIOT (CONT'D)
It looks like this was encrypted with something.

HOLLY
If it was encrypted by someone here, it would have been with the Elfscrow program that we use. There's no way of retrieving the file though unless we know what the secret ID for retrieving the decryption key.

ELLIOT
Do you have a copy of this program?

HOLLY
Yeah.

Holly points Elliot to the program on her computer.

Elliot downloads IDA Free in a web browser and opens Elfscrow in it. He opens the generate_key function.

ELLIOT (V.O.)
According to the code, the current time is being used as the seed for the encryption.

Elliot opens the super_secure_random function.

ELLIOT (V.O.) (CONT'D)
Based on the values being used in the random number generation, this is Microsoft's LCG algorithm.

He goes back to the generate_key function.

ELLIOT (V.O.) (CONT'D)
It's completing eight cycles of random number generation, each time take one byte and stringing it together to create the key.

Goes to the do_encrypt function.

ELLIOT (V.O.) (CONT'D)
Then, it's using the key with the DES-CBC algorithm. Based on the metadata of the document, it must have been created on December 6, 2019 at some point.

Elliot opens a text editor and starts typing.

INT. ELF UNIVERSITY - SECURITY OPERATIONS CENTER - LATER

Elliot finishes his script and runs it.

ELLIOT
There. The document was encrypted at 1575663650 epoch time.

Tries to open the document. It doesn't open, returning a corrupted error.

ELLIOT (V.O.)
KRINGLE. Did I mess this up?

Elliot searches for a corruption repair site on the web. He finds one and uploads the file. Downloads the output.

He opens the file successfully. The file is the Super Sled-O-Matic **Machine Learning Sleigh Route Finder** Quick-Start Guide.

HOLLY
If this is about Santa's sleigh,
we'd better go to the sleigh shop.

INT. ELF UNIVERSITY - STUDENT UNION - DAY

Holly tries to open the sleigh shop door. It doesn't open.

HOLLY
KRINGLE! Can't this day just be
over already? They must have locked
it for safety.

Elliot take a look at a sign above the lock. The sign reads: "Please access from the cloud-accessible lock system at locks.elfu.org".

ELLIOT
Can I use your laptop?

Holly hands him her laptop.

Elliot navigates to the site.

First prompt shows up: You don't need a clever riddle to open the console and scroll a little.

Elliot opens the developer tools and checks the console. There is a code there. He enters it.

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

Elliot pulls up the print page preview. There is a code embedded there. He enters it.

Third prompt shows up: This code is still unknown; it was fetched but never shown.

He checks the Network tab and find an image of the code. He enters it.

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

He checks local storage. Finds another code. Enters it.

Fifth prompt shows up: Did you notice the code in the title? It may very well prove vital.

He checks the title tag in the HTML. There is another code. He enters it.

Sixth prompt shows up: In order for this hologram to be effective, it may be necessary to increase your perspective.

He opens up the Styles tag and increases the perspective value for .hologram.

Seventh prompt shows up: The font you're seeing is pretty slick, but this lock's code was my first pick.

He goes to the HTML and opens up the style tag for the font. There is another code there. He enters it.

Eighth prompt shows up: In the event that the .eggs go bad, you must figure out who will be sad.

He opens up the Event Listener tab when inspecting the .eggs value. There is the code "Veronica". He enters it.

Ninth prompt shows up: This next code will be unredacted, but only when all the chakras are :active.

He opens up the styles.css file linked to by the HTML. He find in page for "charkra" and finds a code broken up across multiple child entries. He enters it.

Tenth prompt shows up: Oh, no! This lock's out of commission! Pop off the cover and locate what's missing.

Elliot opens the lock. There is a code printed on the circuit board. There are gaps in the circuit board.

He enters the code and submits. Gets an error stating "missing macaroni". He searches for macaroni in the HTML, finds a class="macaroni" and drags it into the lock div.

He hits enter again. Gets an error stating "missing swab". He repeats the same procedure for the swab item.

He hits enter again. Gets an error stating "missing gnome". He repeats the same procedure for the gnome item.

The door opens.

Elliot and Holly run back inside.

INT. ELF UNIVERSITY - SLEIGH SHOP - DAY

It's dark.

HOLLY
(disgusted)
Ugh. What's that smell?

Elliot flips the light on.

Someone out of view begins SLOW CLAPPING.

Holly gasps.

Reveal the gargantuan, hideous figure of **the TOOTH FAIRY**. 10 feet tall, rotted, monstrous face.

TOOTH FAIRY
So... someone figured it out after all. I was wondering if someone would come along. Well, it's too late. Santa's sleigh is done for with all of the junk data we've sent it. I'm going to go eat some sugar while Christmas burns to the ground.

The Tooth Fairy stands up and lumbers out of the room.

A pause.

Elliot opens Holly's laptop.

ELLIOT
Where is the data to the sled-o-matic ingested?

HOLLY
We have an interface at srf.elfu.org.

Elliot goes there.

ELLIOT
How to I get in?

HOLLY
Creds are in the git repo. Go to /README.md.

Elliot goes to the readme page and copy/pastes the default credentials.

He goes to the log tab and downloads a copy of the log files.

ELLIOT
What should I look for?

HOLLY
Probably standard techniques.

He opens up the log files in a text editor on one side of the screen and a terminal on the other.

First searches for ".." in the text editor and sees the relevant log entries. He gathers them with: jq '.[] | select(.uri | test("[.]{2}")) ' http.log >> mal_logs.txt.

Next he searches "1=1". Grabs with: jq '.[] | select(.username | test("1=1")) ' http.log >> mal_logs.txt.

Next "UNION". Grabs them with jq '.[] | select(.user_agent | test("UNION")) ' http.log >> mal_logs.txt ***and*** jq '.[] | select(.uri | test("UNION")) ' http.log >> mal_logs.txt.

Next "/etc/passwd". Grabs with: jq '.[] | select(.uri | test("\/etc\/passwd")) ' http.log >> mal_logs.txt.

Next "/bin/". Grabs with: jq '.[] | select(.user_agent | test(".*\/bin\/.*")) ' http.log >> mal_logs.txt.

Next "<script>". Grabs with jq '.[] | select(.uri | test("<script>")) ' http.log >> mal_logs.txt ***and*** jq '.[] | select(.host | test("<script>")) ' http.log >> mal_logs.txt.

He dedupes the log list with cat mal_logs.txt | jq -s '.' | jq 'unique_by(.uid)' | pbcopy.

He counts the user_agent entries and looks at the agent names. Runs: cat mal_logs.txt | jq -r -s ".[] | unique_by(.user_agent) | .[].user_agent" > user_agents.txt ***and*** IFS=$'\n'; for each in $(cat user_agents.txt); do jq -r --arg agent "$each" '.[] | select(.user_agent==$agent) | .user_agent' http.log >> agent_count.txt; done ***and*** cat agent_count.txt | uniq -c | sort.

He looks at the output. There are four agents with ten or more entries.

ELLIOT (V.O.)
Most of these agents look weird, but the most common ones might be legit.

Elliot opens user_agents.txt and manually reviews the entries, removing those that are not malicious for those with ten or more entries.

He runs: IFS=$'\n'; for each in $(cat user_agents.txt); do jq -r --arg agent "$each" '.[] | select(.user_agent==$agent)' http.log >> surplus_ips.txt; done ***and*** for each in $(cat surplus_ips.txt | jq -s -r '.[] | ."id.orig_h"'); do printf "$each, "; done.

He gets a "route calculation success" message with a code of **0807198508261964.**

ELLIOT
Done. The sleigh should be all good to go.

HOLLY
Good. We better get back to base to check.

EXT. ELF UNIVERSITY - QUAD - DAY

Holly and Elliot walk towards the SOC.

There is a loud ROAR behind them. They whip around.

The Tooth Fairly is lumbering towards them.

TOOTH FAIRY
You! You foiled my plot. Months of labor and cost. Do you know how hard it is to steal Monero from the North Korean military? Now it's all wasted. You two are going to pay.

The Tooth Fairy lumbers over, shoving Elliot and Holly to the ground.

TOOTH FAIRY (CONT'D)
Goodbye.

The Tooth Fairy lifts a massive foot and brings it above Elliot and Holly's heads.

The small figure of Kent jumps up on the Tooth Fairy and lets out a BATTLE CRY. Kent wraps a long strand of dental floss around the Tooth Fairy.

With each word, he digs the floss into a nook in the Tooth Fairy's body. Each dig causes dental plaque to go flying and the Tooth Fairy to shriek.

KENT
(yelling)
What kind of creep leaves a message for someone in a malware file? What did you do, stalk me on LinkedIn?

The Tooth Fairy continues to scream, but its voice gets higher and higher in pitch as it shrinks to the size and appearance of a normal human tooth.

Kent picks up the Tooth Fairy.

KENT (CONT'D)
Time to stick you under someone's pillow.

Kent walks away with the still screaming Tooth Fairy.

HOLLY
Elliot, thank you for all of your help. If there is anything we can do to repay...

DARLENE (O.S.)
Elliot!

ELLIOT (V.O.)
Huh?

INT. ELLIOT'S APARTMENT - MORNING

DARLENE is standing over Elliot.

DARLENE
Hey dumb KRINGLE. Wake up. We need to fix this KRINGLE.

ELLIOT
I'm up, I'm up.

Elliot pivots to the edge of the bed.

ELLIOT (CONT'D)
What's the problem? Isn't it Christmas?

THE END

Code for Objective 8 - Bypassing the Frido Sleigh CAPTEHA

```
#!/usr/bin/env python3
# Fridosleigh.com CAPTEHA API - Made by Krampus Hollyfeld
import base64
import requests
import json
import glob
import sys
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
import tensorflow as tf
import numpy as np
import threading
import queue
import time
import re

def main():
    yourREALemailAddress = "tisop39103@mailart.top"

    # Creating a session to handle cookies
    s = requests.Session()
    url = "https://fridosleigh.com/"

    json_resp = json.loads(s.get("{}api/capteha/request".format(url)).text)
    b64_images = json_resp['images']                    # A list of
dictionaries eaching containing the keys 'base64' and 'uuid'
    challenge_image_type = json_resp['select_type'].split(',')     # The
Image types the CAPTEHA Challenge is looking for.
    challenge_image_types = [challenge_image_type[0].strip(),
challenge_image_type[1].strip(), challenge_image_type[2].replace(' and
','').strip()] # cleaning and formatting

    files = glob.glob("./unknown_images/*")
    for f in files:
        os.remove(f)

    files = glob.glob("./selected_images/*")
```

```
    for f in files:
        os.remove(f)

    for image in b64_images:
        f = open("./unknown_images/" + image["uuid"] + ".png", "xb")
        f.write(base64.b64decode(image["base64"]))
        f.close()

    answer = predict(challenge_image_types)

    # This should be JUST a csv list image uuids ML predicted to match the
challenge_image_type .
    final_answer = ','.join(answer)

    json_resp = json.loads(s.post("{}api/capteha/submit".format(url),
data={'answer':final_answer}).text)
    if not json_resp['request']:
        # If it fails just run again. ML might get one wrong occasionally
        print('FAILED MACHINE LEARNING GUESS')
        print('--------------------\nOur ML
Guess:\n--------------------\n{}'.format(final_answer))
        print('--------------------\nServer
Response:\n--------------------\n{}'.format(json_resp['data']))
        sys.exit(1)

    print('CAPTEHA Solved!')
    # If we get to here, we are successful and can submit a bunch of
entries till we win
    userinfo = {
        'name':'Krampus Hollyfeld',
        'email':yourREALemailAddress,
        'age':180,
        'about':"Cause they're so flippin yummy!",
        'favorites':'thickmints'
    }
    # If we win the once-per minute drawing, it will tell us we were
emailed.
    # Should be no more than 200 times before we win. If more, somethings
wrong.
    entry_response = ''
    entry_count = 1
    while yourREALemailAddress not in entry_response and entry_count < 200:
```

```
        print('Submitting lots of entries until we win the contest! Entry
#{}'.format(entry_count))
        entry_response = s.post("{}api/entry".format(url),
data=userinfo).text
        entry_count += 1
    print(entry_response)



def load_labels(label_file):
    label = []
    proto_as_ascii_lines = tf.gfile.GFile(label_file).readlines()
    for l in proto_as_ascii_lines:
        label.append(l.rstrip())
    return label

def predict_image(q, sess, graph, image_bytes, img_full_path, labels,
input_operation, output_operation):
    image = read_tensor_from_image_bytes(image_bytes)
    results = sess.run(output_operation.outputs[0], {
        input_operation.outputs[0]: image
    })
    results = np.squeeze(results)
    prediction = results.argsort()[-5:][::-1][0]
    q.put( {'img_full_path':img_full_path,
'prediction':labels[prediction].title(), 'percent':results[prediction],
'bytes': image_bytes} )

def load_graph(model_file):
    graph = tf.Graph()
    graph_def = tf.GraphDef()
    with open(model_file, "rb") as f:
        graph_def.ParseFromString(f.read())
    with graph.as_default():
        tf.import_graph_def(graph_def)
    return graph

def read_tensor_from_image_bytes(imagebytes, input_height=299,
input_width=299, input_mean=0, input_std=255):
    image_reader = tf.image.decode_png( imagebytes, channels=3,
name="png_reader")
    float_caster = tf.cast(image_reader, tf.float32)
```

```
    dims_expander = tf.expand_dims(float_caster, 0)
    resized = tf.image.resize_bilinear(dims_expander, [input_height,
input_width])
    normalized = tf.divide(tf.subtract(resized, [input_mean]), [input_std])
    sess = tf.compat.v1.Session()
    result = sess.run(normalized)
    return result

def predict(requested_types):
    # Loading the Trained Machine Learning Model created from running
retrain.py on the training_images directory
    graph = load_graph('/tmp/retrain_tmp/output_graph.pb')
    labels = load_labels("/tmp/retrain_tmp/output_labels.txt")

    # Load up our session
    input_operation = graph.get_operation_by_name("import/Placeholder")
    output_operation = graph.get_operation_by_name("import/final_result")
    sess = tf.compat.v1.Session(graph=graph)

    # Can use queues and threading to spead up the processing
    q = queue.Queue()
    unknown_images_dir = 'unknown_images'
    unknown_images = os.listdir(unknown_images_dir)

    #Going to interate over each of our images.
    for image in unknown_images:
        img_full_path = '{}/{}'.format(unknown_images_dir, image)
        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()) > 100:
            time.sleep(0.0001)

        #predict_image function is expecting png image bytes so we read
image as 'rb' to get a bytes object
        image_bytes = open(img_full_path,'rb').read()
        threading.Thread(target=predict_image, args=(q, sess, graph,
image_bytes, img_full_path, labels, input_operation,
output_operation)).start()

    print('Waiting For Threads to Finish...')
    while q.qsize() < len(unknown_images):
        time.sleep(0.0001)
```

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

    uid_list = []
    print_list = []
    #do something with our results... Like print them to the screen.
    print(requested_types)
    for prediction in prediction_results:
        pattern = "[a-z0-9A-Z-]+(?=\.png)"
        file_path = re.search(pattern, prediction["img_full_path"])
        uid = file_path.group(0)
        #print(uid + " " + prediction["prediction"] + " -
{:.2%}".format(prediction["percent"]))
        if prediction["prediction"] in requested_types:
            string = uid + " " + prediction["prediction"] + " -
{:.2%}".format(prediction["percent"])
            f = open("./selected_images/" + uid +  " " +
prediction["prediction"] + ".png", "xb")
            f.write(prediction["bytes"])
            f.close()
            print(string)
            print_list.append(string)
            uid_list.append(uid)

    print(print_list.sort())
    return uid_list


if __name__ == "__main__":
    main()
```

Code for Objective 10 - Recover Cleartext Document

```
from des import DesKey
from Cryptodome.Cipher import DES
import binascii
import PyPDF2

key = []
```

```
def lcg_rand(seed):
    global key
    seed = (214013 * seed + 2531011) & 0xFFFFFFFF
    seed2 = (seed | 0xFFFFFFFF00000000) >> 16
    seed2 = seed2 & 0xFFFFFFFF
    seed2 = seed2 & 0x7FFF
    seed2 = seed2 & 0x00FF
    seed2 = seed2 & 0x0FF
    key.append(seed2)
    return seed

def encrypt(key, text):
    return key.encrypt(text, initial=0, padding=True)

def decrypt(key, encrypted_text):
    return key.decrypt(encrypted_text, padding=True)

def main():
    global key

    for seed in range(1575658800, 1575666001):
        key = []
        i = 0
        print("original seed {}".format(seed))
        while i < 8:
            seed = lcg_rand(seed)
            seed2 = seed
            i += 1
        key = bytearray(key)
        des = DES.new(key, DES.MODE_CBC, iv=b'00000000')
        with
open("ElfUResearchLabsSuperSledOMaticQuickStartGuideV1.2.pdf.enc", "rb") as
f:
            f_bytes = f.read()
            dec_bytes = des.decrypt(f_bytes)
            with open("output.pdf", "wb") as outfile:
                outfile.write(dec_bytes)
                try:
                    PyPDF2.PdfFileReader(open("output.pdf", "rb"))
                    print("succeeded with seed {} key {}".format(seed,
key))
                    break
```

```
                except PyPDF2.utils.PdfReadError:
                    continue

if __name__ == "__main__":
    main()
```