CSE 5382: Secure Programming - Spring 2021
Capture the Flag Bonus Exercise

Student Name: Varunkumar Pande
Student ID: 1001722538

Flag Table

Place the flags once you find them in the appropriate table entries below. Each category has a maximum of 20 points with each higher level of difficulty granting an increasing number of points. Maximum score for all challenges completed is 100 points.

	Cryptography	**Malware**	**Network Capture**	**Reverse Engineering**	**Steganography**
Level 1 (2 Points)	backoff_malware	flag{sandworm_apt}	M0d1c0nF14G	lm	flag{covert_channel}
Level 2 (3 Points)	WANNACRY	2e1afcef9baa15b8db764274b0e45d3f	YWRtaW46YWRtaW4=\r\n admin:admin (YWRtaW46 YWRtaW4)	ClippyHasReturned	flag{cookies_n_milk}
Level 3 (4 Points)		flag{duqu_aint_dooku}	NTLMSSP	infectedflag	ROO7SRUS or R0075RU5
Level 4 (5 Points)	DID CICERO SAY ANYTHING	123123	1255	73C972DF8DB0E1EDC289F491A09AF330	flag{alan_turing_edm}
Level 5 (6 Points)		flag{kragany_uses_up$x}	psswordBasisk	flag{ }	Be Bold

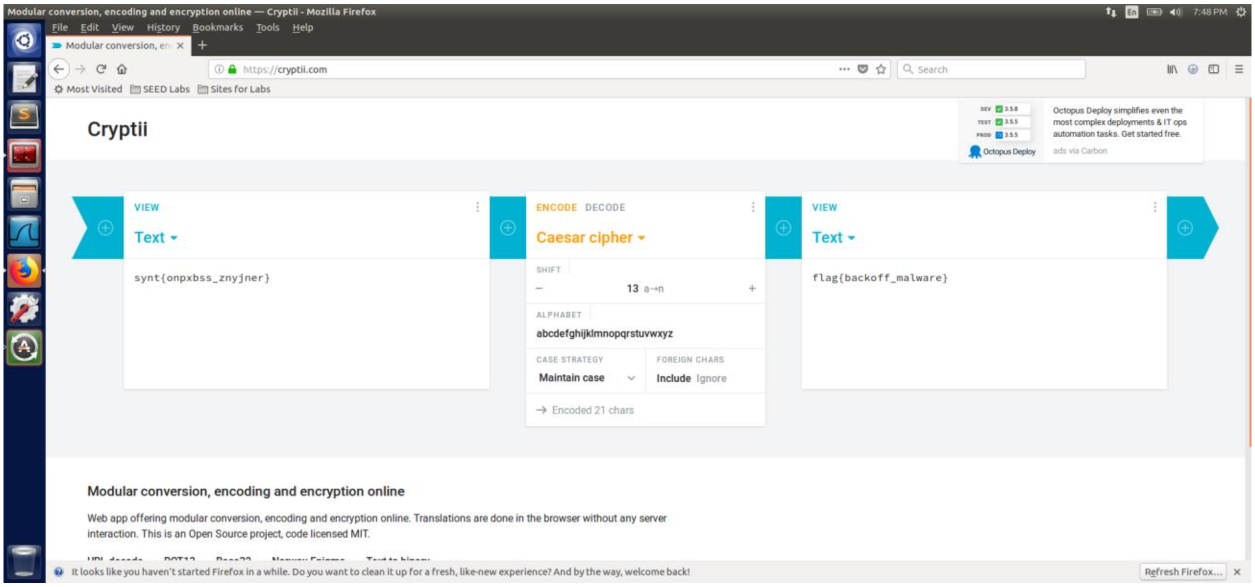
Explanation of Approaches

Include descriptions along with screen shots of the approaches you took to solve these challenges.

Cryptography

Level 1 – Orange Julius

Orange Julius loosely relates to the term Julius Caeser; hence I used the Caeser cipher to decode the text. On increasing the number of shifts, finally at 13th shift we get a string “flag{backoff_malware}” which is the only statement in proper English.



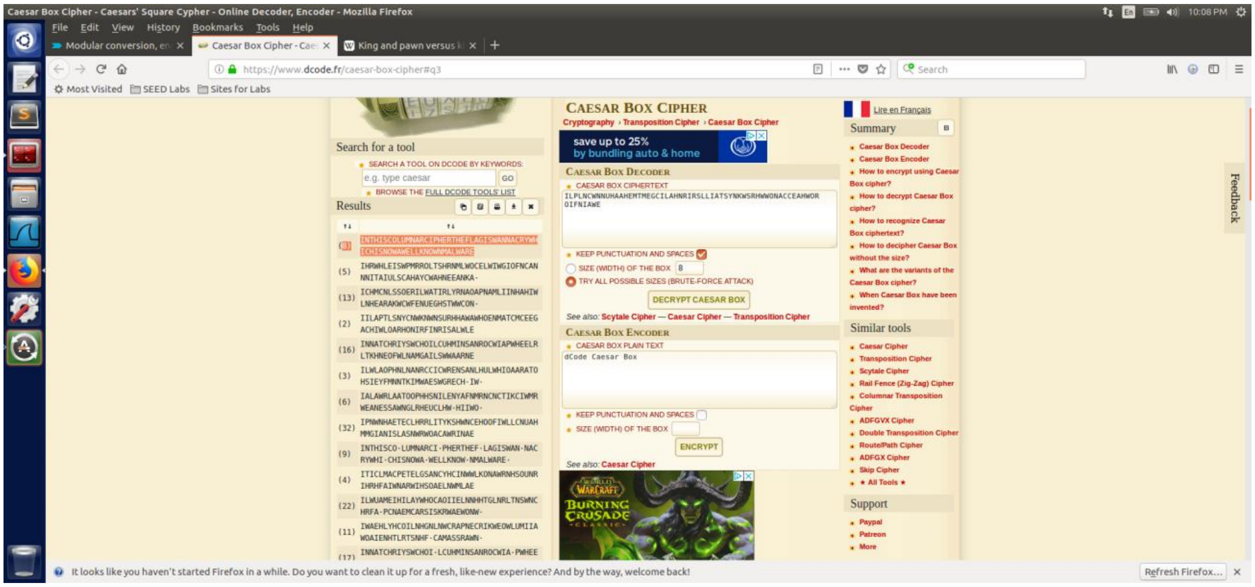
Level 2 – Block Chain

"If you square this thing, the pawn can beat the king" – (Rule of the square- The most basic rule is when the pawn can queen unassisted by its king. The *rule of the square* determines if this is possible. In this position, the pawn is on the fifth square from the queening square (counting the queening square itself). A square of 5×5 squares with the queening square in one corner and the pawn in an adjacent corner can be imagined. (An easy method is to construct the square with a diagonal from the pawn to the last rank.) If the black king can move into this square, he can catch the pawn, otherwise the pawn wins the race)

The above statement helped me realize about the technique that we must use in decrypting the given string i.e., "Caeser box cipher":

On using a brute force technique, we can see that the key used in decryption is "8" and the decrypted text is "IN THIS COLUMNAR CIPHER THE FLAG IS WANNACRY WHICH IS NOW A WELL KNOWN MALWARE".

the Answer to this CTF task is: "WANNACRY".



Level 3 – DASH DOT COM

Level 4 – VIII A Small Example

On following this link https://www.google.com/search?q=%22VIII+A+SMALL+EXAMPLE%22 I opened the RSA.pdf where-in the section “VIII” shows an example of converting normal text to encoded format. On following the same for the given string I decoded the text to get the flag as follows:

2510	0017	0102	2427	1078	1436	0164	0001	2426	1567	2503	0761
0409	0400	0309	0305	1815	0019	0125	0001	1425	2008	0914	0700
DI	D	CI	CE	RO	S	AY	A	NY	TH	IN	G

- (2510**157)%2773 = 409
- (17**157)%2773 = 400
- (102**157)%2773 = 309
- (2427**157)%2773 = 305
- (1078**157)%2773 = 1815
- (1436**157)%2773 = 19
- (164**157)%2773 = 125
- (1**157)%2773 = 1
- (2426**157)%2773 = 1425
- (1567**157)%2773 = 2008
- (2503**157)%2773 = 914
- (761**157)%2773 = 700

Post decryption:

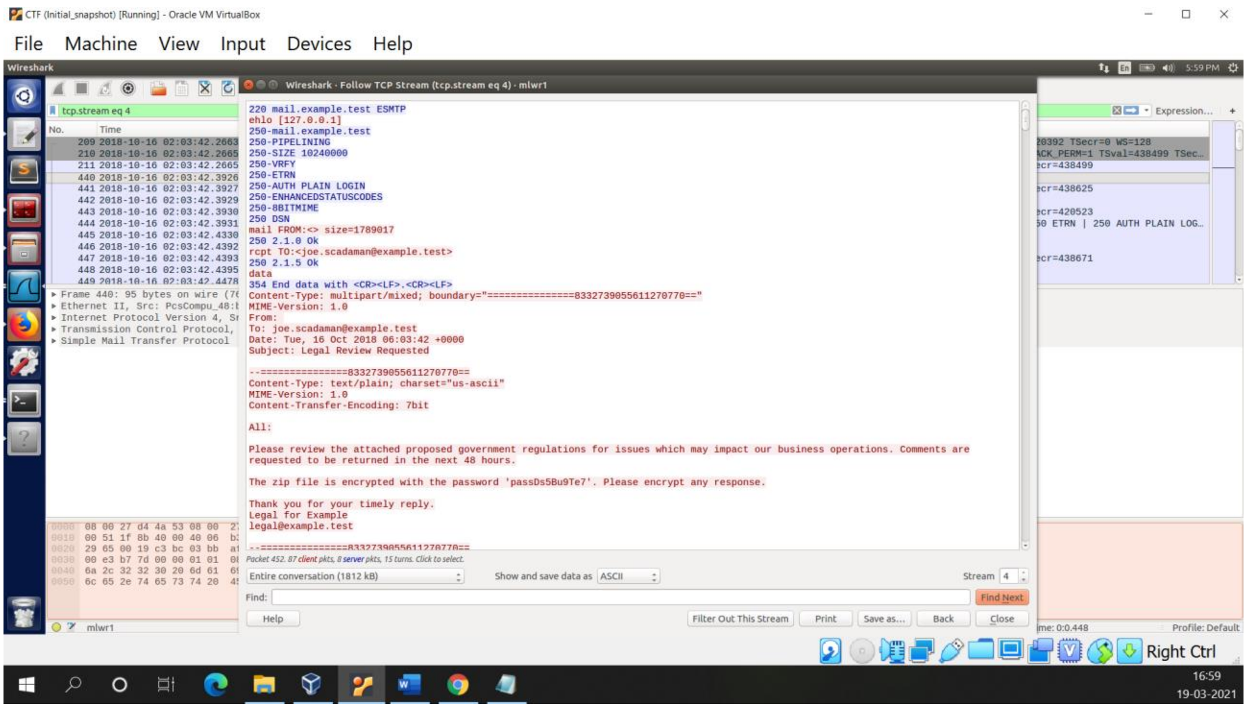
“DID CICERO SAY ANYTHING “

Level 5 – Big Blue Randu

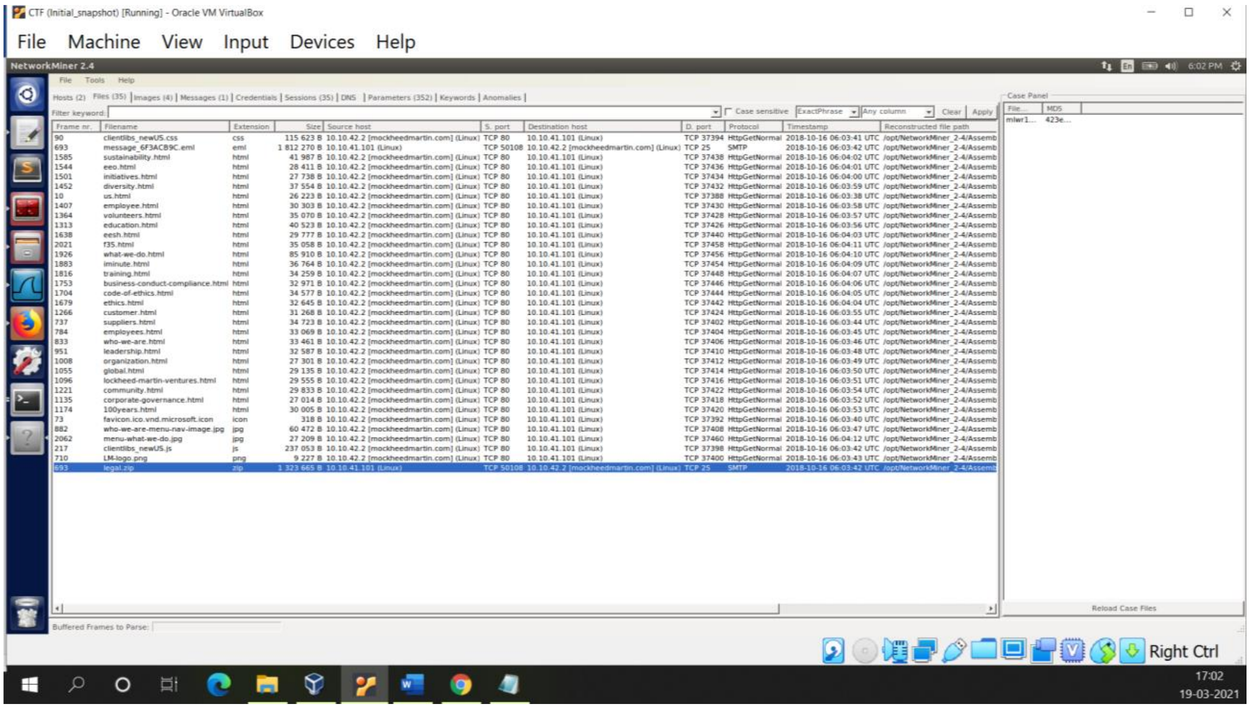
Malware

Level 1

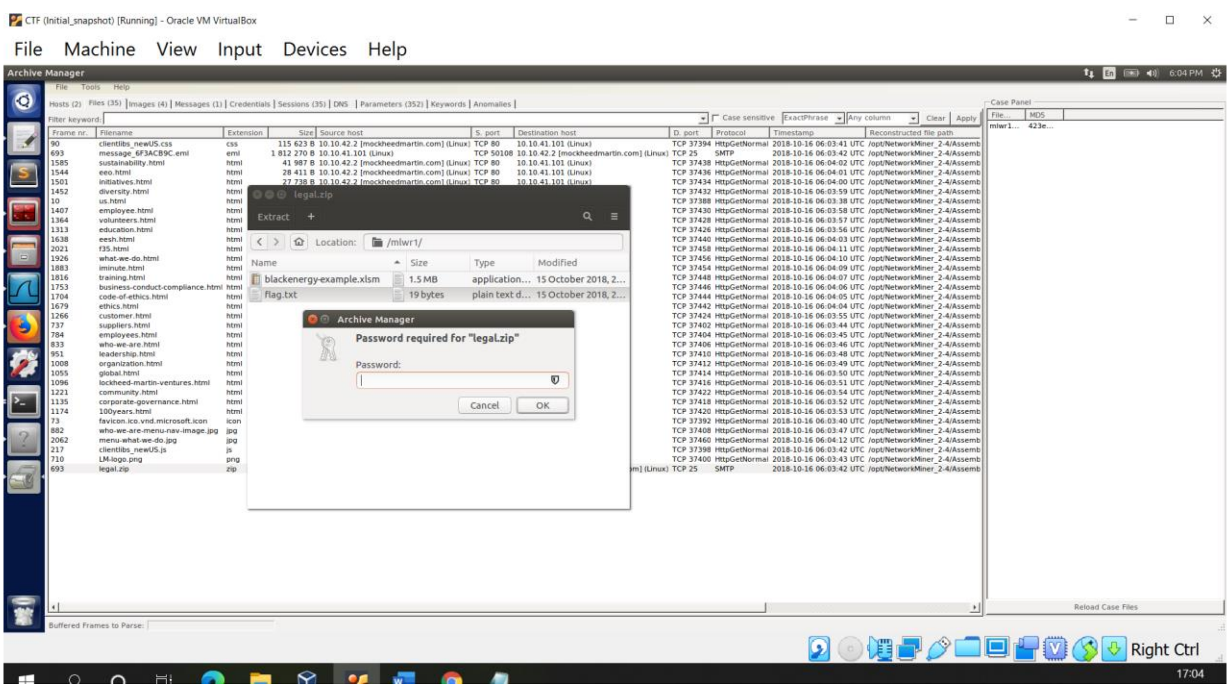
Going through the network capture in Wireshark I realized that the zip file was sent using SMTP, further on filtering based on smtp and following the TCP packets, I got the entire file and password, but the file was encoded. Hence I used a tool named NetworkMiner to extract the files.



For this task I used a tool named NetworkMiner 2.4. because it is a good tool for analyzing files in the network capture logs. Going into the files tab I saw that there is only one zip file named "legal.zip" as shown in the below screenshot:

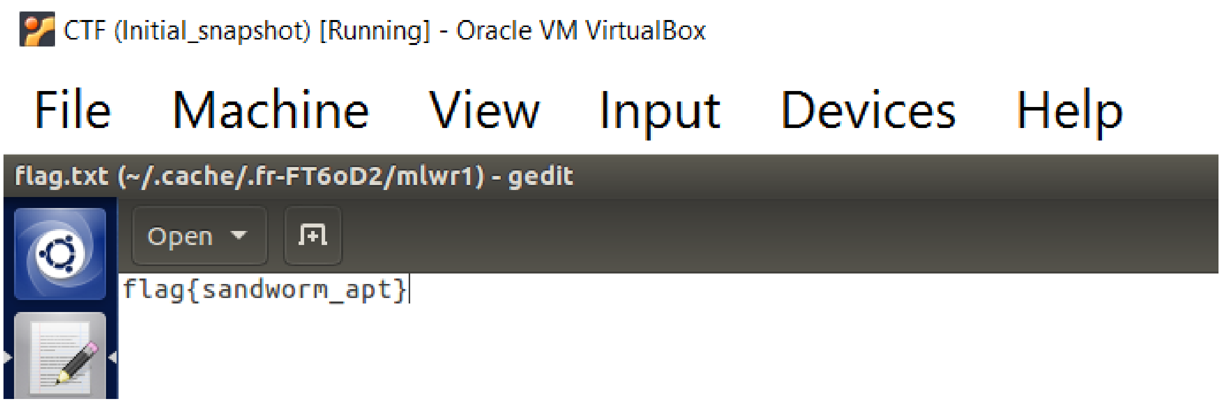


On opening the zip file, I found two files “blackenergy-example.xlsm” and “flag.txt”, on opening flag.txt I was prompted for a password. I entered the password found while analyzing the smtp packets earlier:



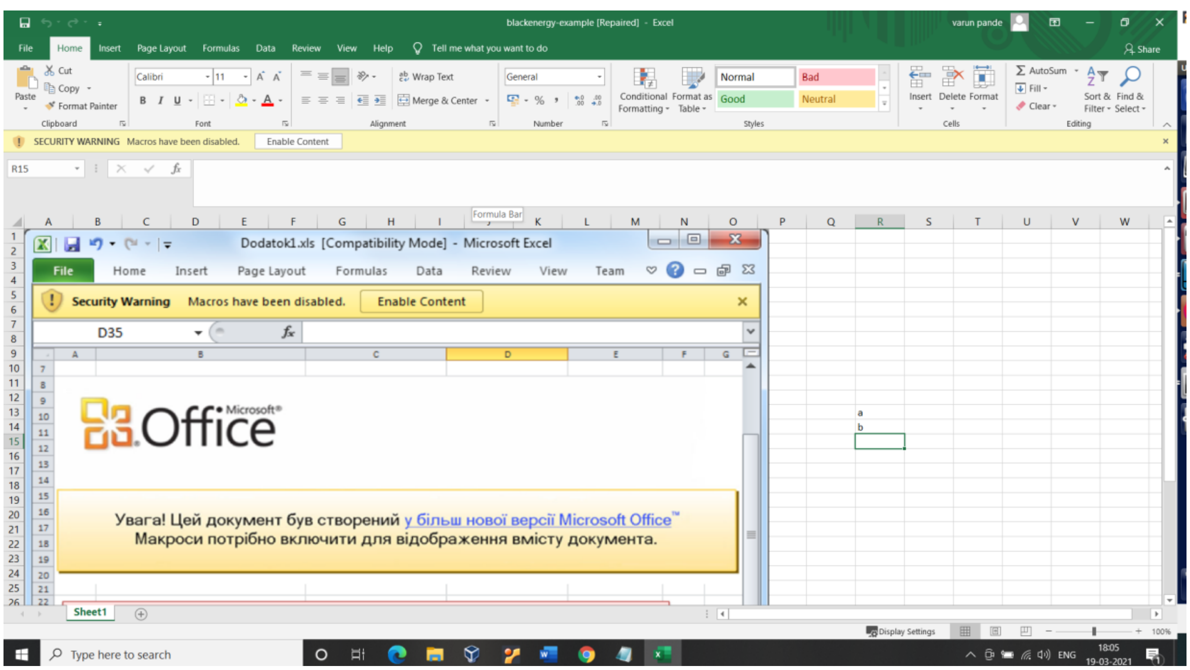
```
All:

Please review the attached proposed government regulations for issues which may impact our business operations. Comments are
requested to be returned in the next 48 hours.

The zip file is encrypted with the password 'passDs5Bu9Te7'. Please encrypt any response.

Thank you for your timely reply.
Legal for Example
legal@example.test
```

And finally found the flag:

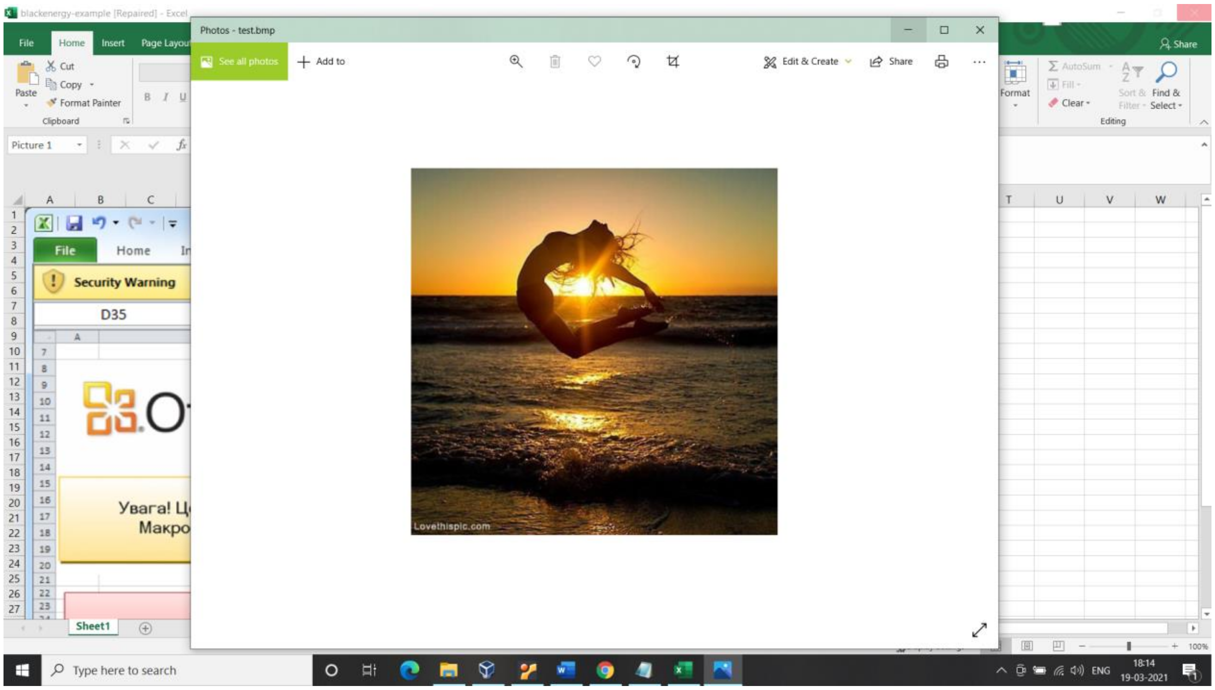


Level 2

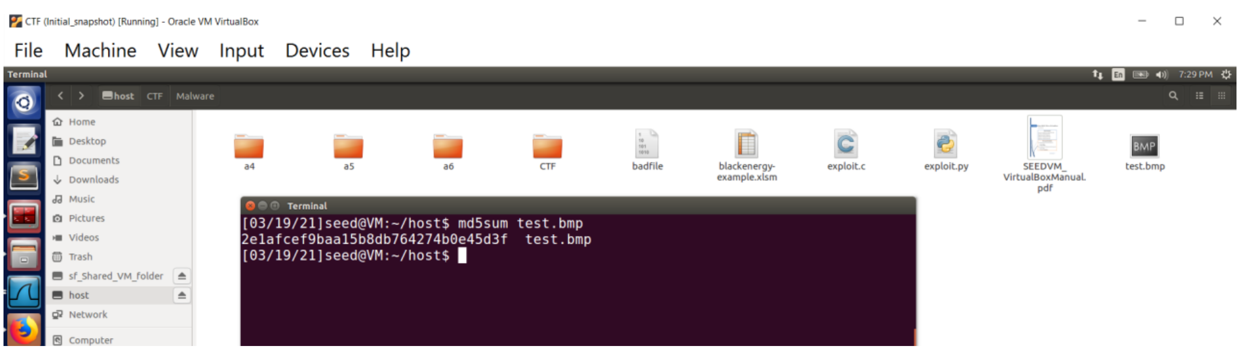
In the previous task we found a file named "blackenergy-example.xlsm" so as per the task I opened the excel file, then I got a macros disabled warning message



On enabling execution of macro a picture pops up, I saved the picture to calculate its MD5 hash to get the flag.

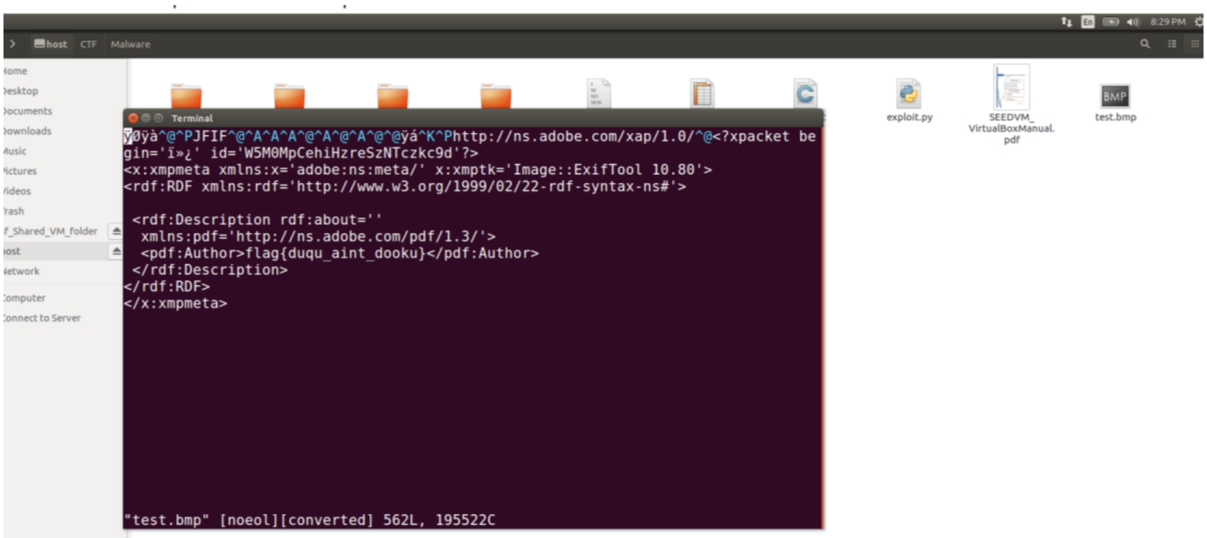


The flag: “2e1afcef9baa15b8db764274b0e45d3f”



Level 3

While trying to analyze the “test.bmp” file I was able to see the following output:

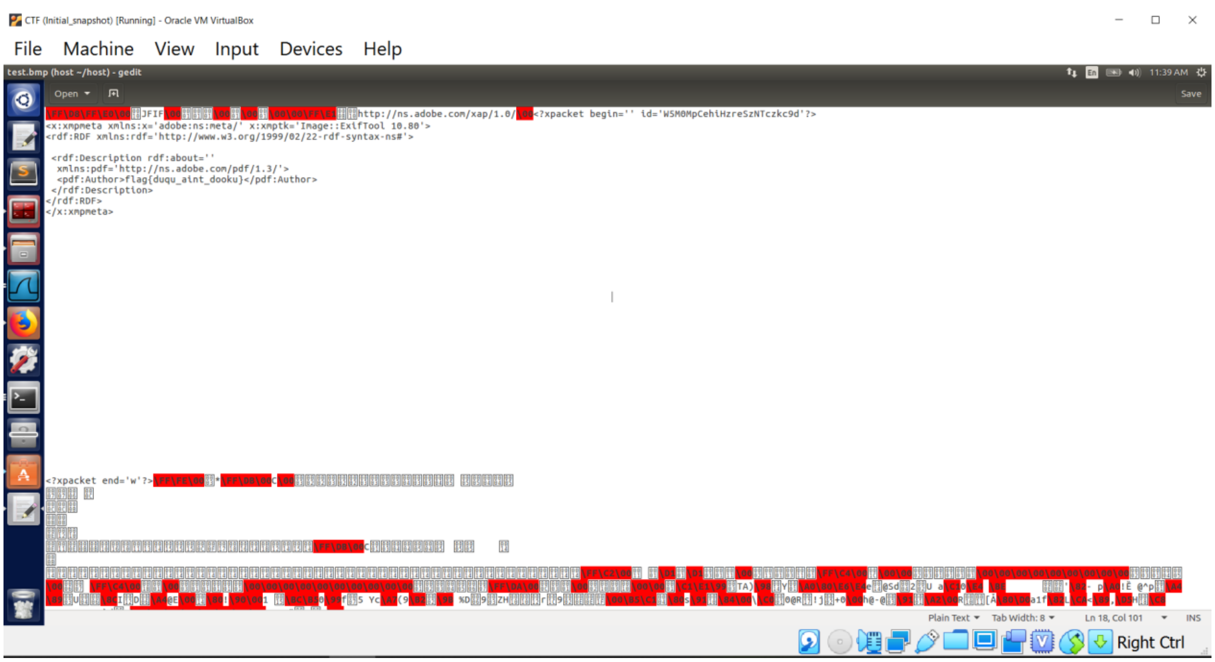


On looking closely, I found this flag:

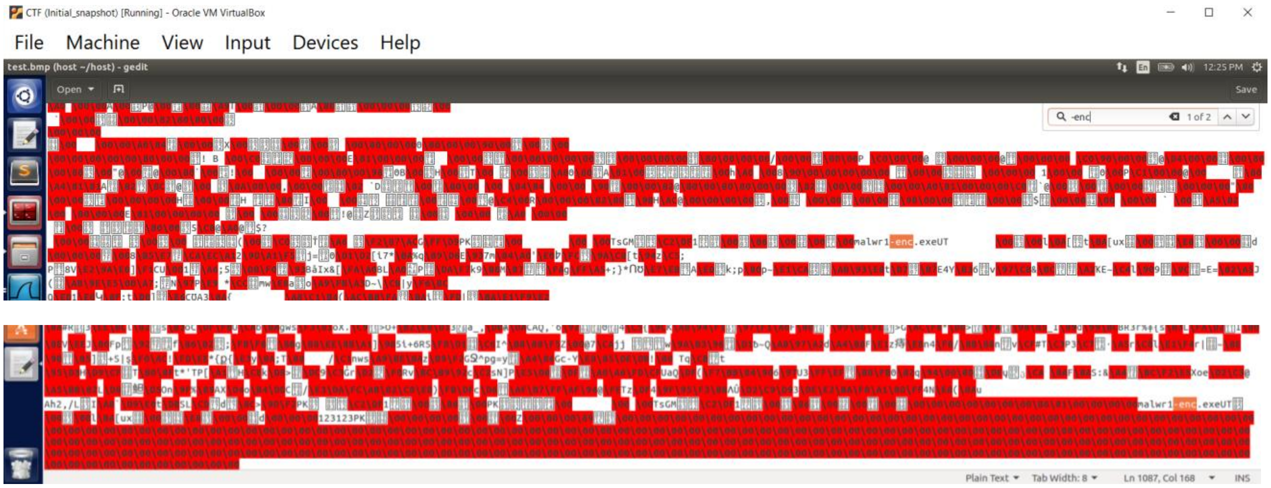
“<pdf:Author>flag{duqu_aint_dooku}</pdf:Author>”

Level 4

For this challenge it was said that the file that was found in the previous task contains the flag for this challenge. On scrolling through the file further some of the characters were not being printed properly, so I opened the file using Gedit and I saw hexcode characters being printed as shown below.



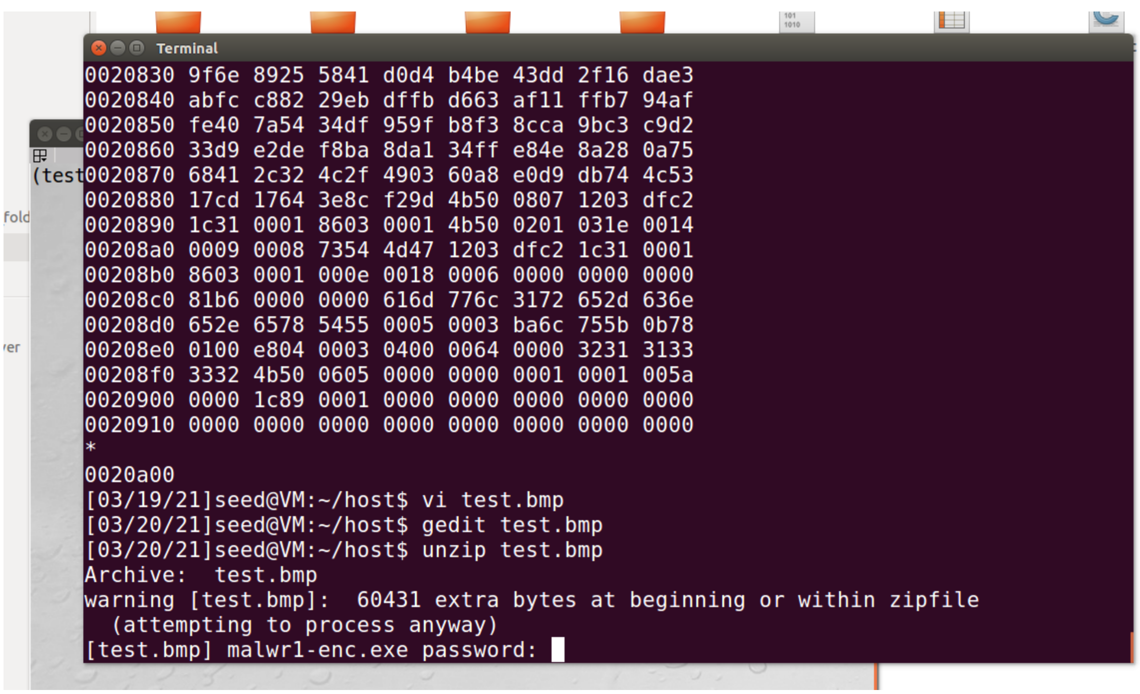
Since it was given that the file to be found was encrypted so I searched for terms such as “flag, pass, enc, -enc”. For the “-enc” I got two entries that were interesting:



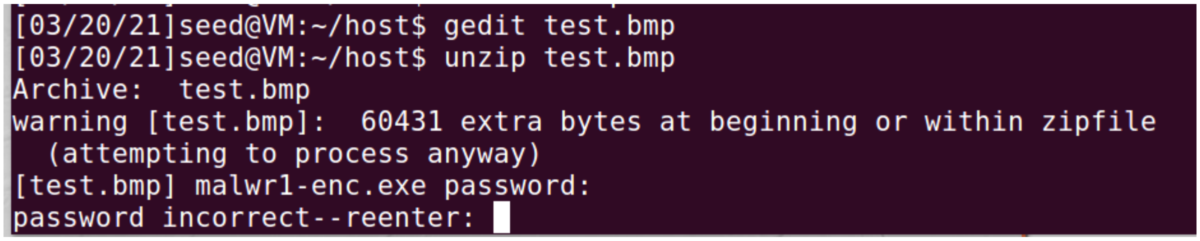
In the second search the term “123123PK” seemed to be a close candidate for a password. While trying the level 5 task I got an error for “123123PK”, so I tried “123123” which worked. Hence the flag for this task is “123123”.

Level 5

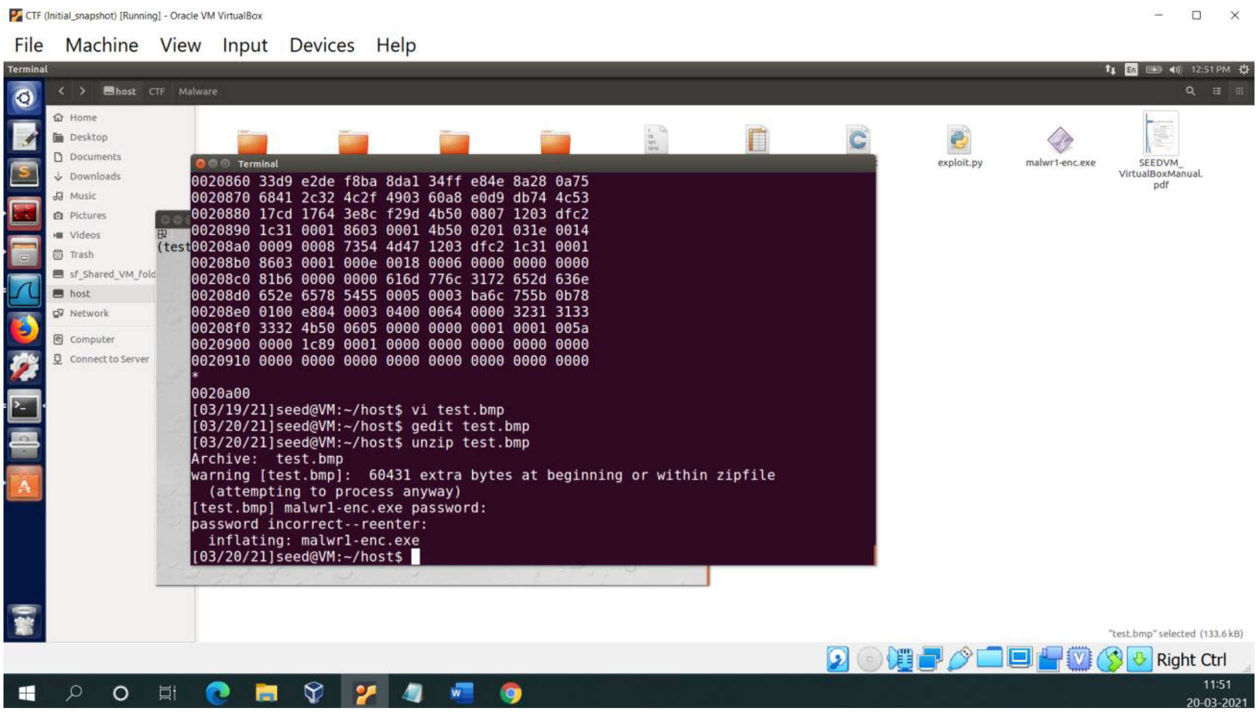
“The matryoshka is over”- (Matryoshka dolls (Russian: матрёшка) (About this soundlisten); also known as babushka dolls, stacking dolls, nesting dolls, Russian tea dolls, or Russian dolls) are a set of wooden dolls of decreasing size placed one inside another.) – Wikipedia. For this task I googled a little bit online to understand how BMP files are packed with other executables. A common way to do that is compressing the executable files in it. So, I tried unzipping the “test.bmp” file:



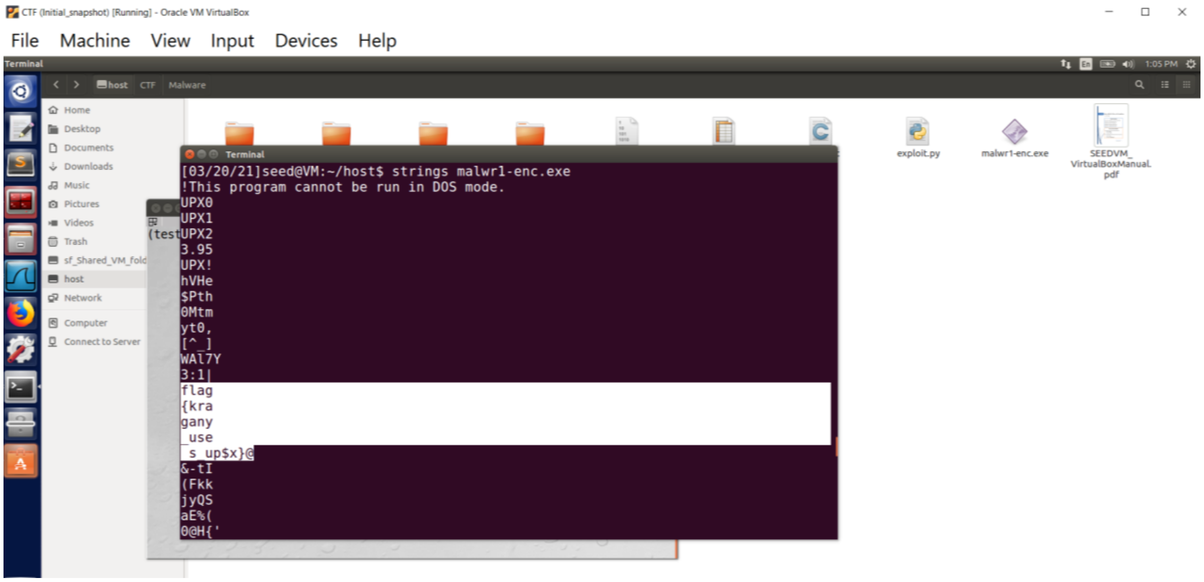
I was prompted for a password, so I entered the one that I found in the previous task.



Initially I entered the password 123123PK for which I got an error as shown in the screenshot above. So, I tried "123123" and I got the following output, also another file named "malwr1-enc.exe" was created.



Since the task states "The final malware flag is in the file extracted from MLWR-4-CRY". I used the strings command to analyze the executable and searched for the "flag" term, which led me to the flag.



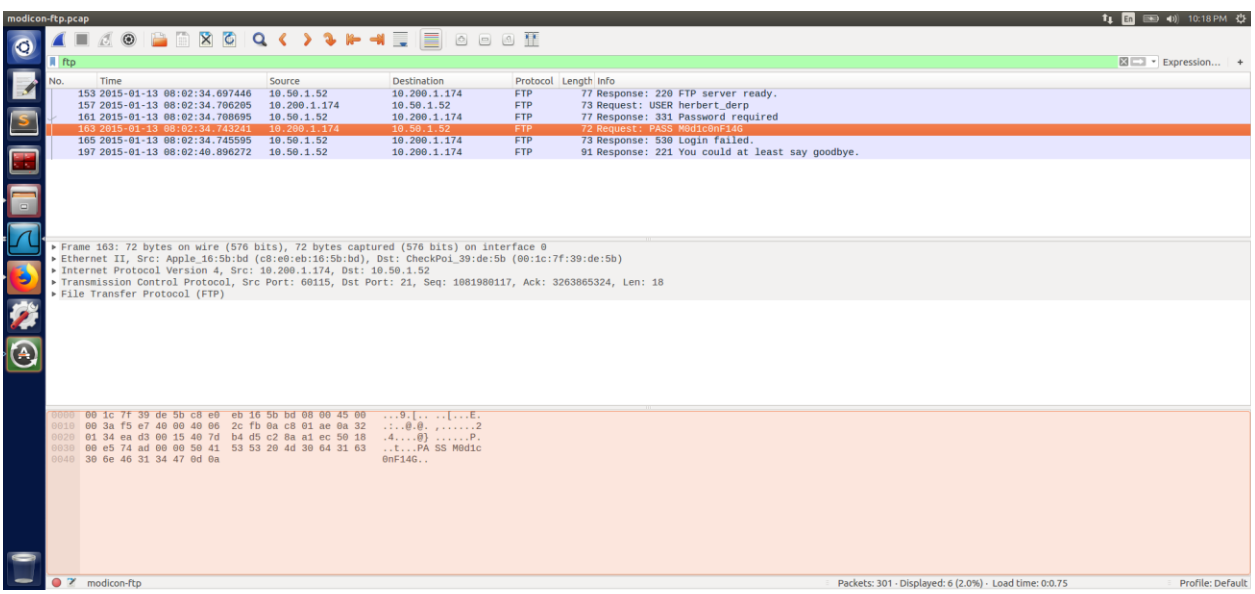
The flag is “flag{kragany_uses_up$x}”.

Network Capture

Level 1 – Modicon-FTP

Since the file has a “pcap” extension I used wireshark to observe the packet capture log. In the question it was given that the request was made using ftp protocol, So I filtered the log for ftp and then finally we can see the password present in the log screenshot below:

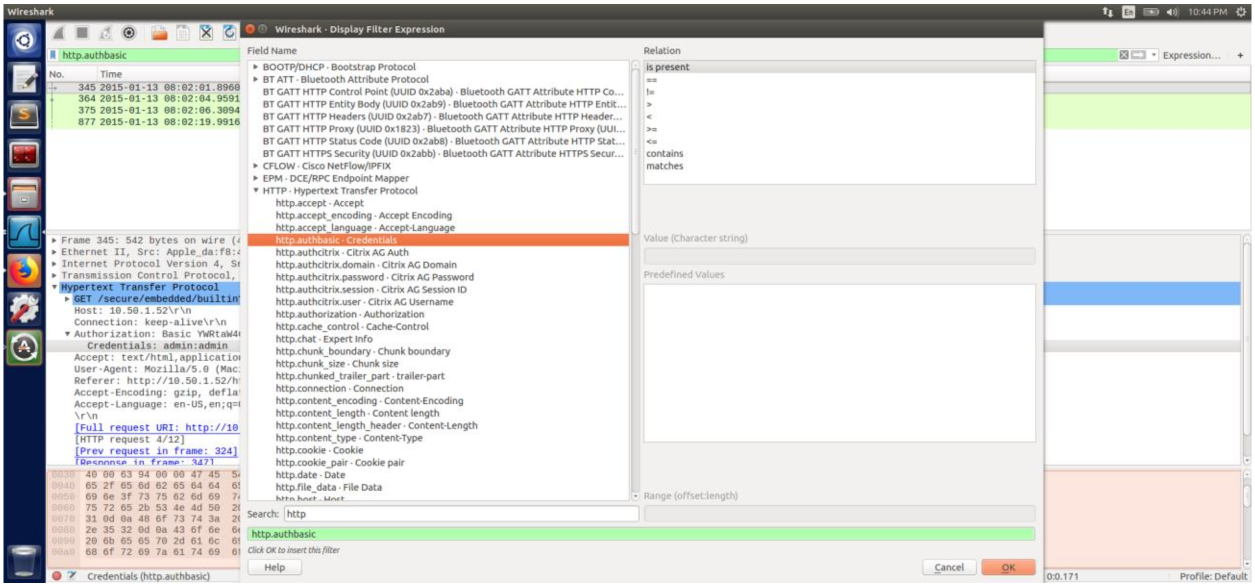
(163 2015-01-13 08:02:34.743241 10.200.1.174 10.50.1.52 FTP 72 Request: PASS M0d1c0nF14G)



Level 2 – Modicon-HTTP

In wireshark there are pre-defined filters for certain protocols and type of packets, so I went to the filter section and chose “http.authbasic” filter, since in the question itself they say the “HTTP basic Authorization password is required.”

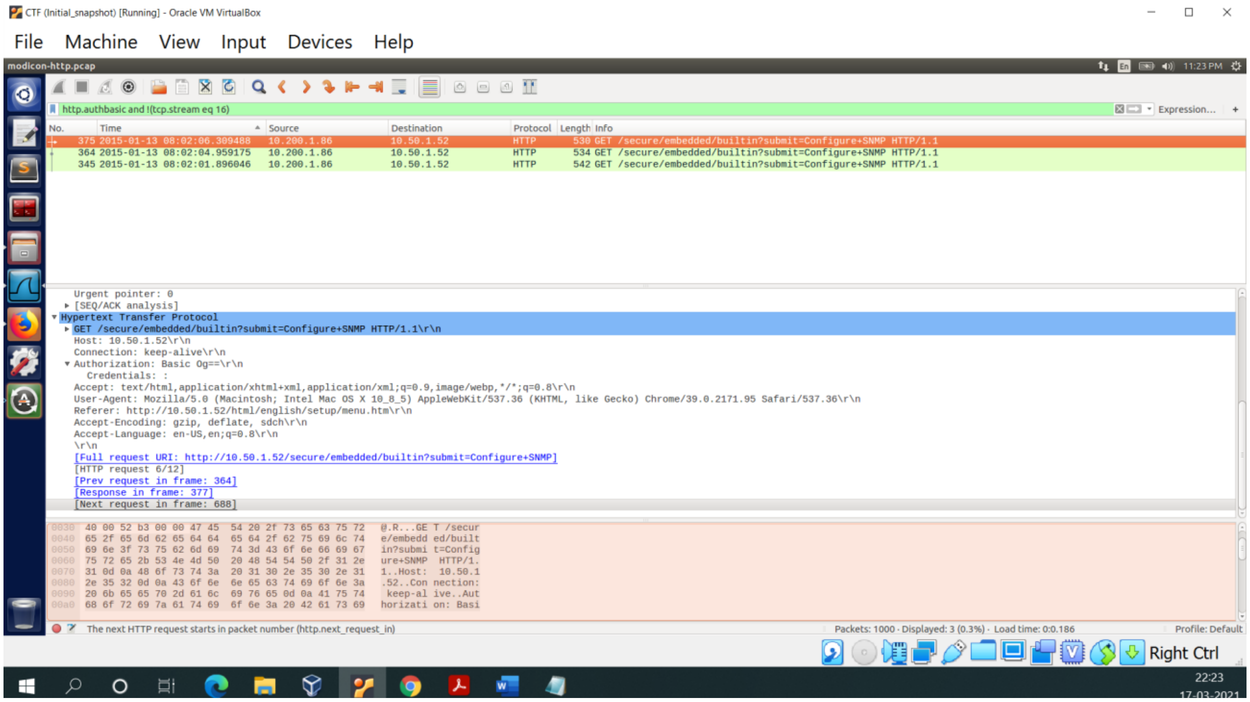
Selecting the filter:



On applying the filter we get the filtered packets as shown in the below screenshot, post which I followed the 1st packet capture and as I was going through the logs I realized that this was a successful login(credentials: USER:USER) so I filtered out this packet from my set of remaining packets. Now on going through the other packet captures, I saw the following credentials for which there were authorization errors:

1. : (0g==\r\n)
2. admin: (YWRtaW46\r\n)
3. admin:admin (YWRtaW46YWRtaW4=\r\n)

so, the password for failed user was “admin or empty credentials”.



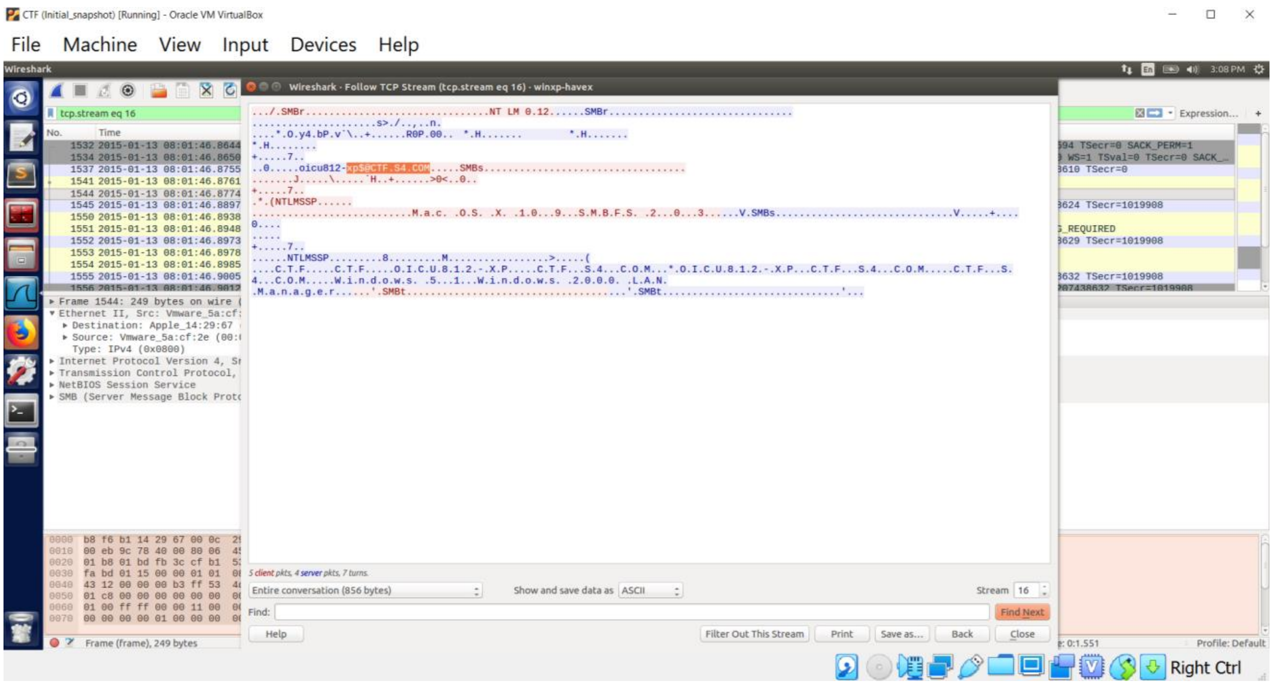
Level 3 – WinXP-HAVEX

IOC - **Indicator of compromise** (IoC) in computer forensics is an artifact observed on a network or in an operating system that, with high confidence, indicates a computer intrusion. ... Typical IoCs are virus signatures and IP addresses, MD5 hashes of malware files, or URLs or domain names of botnet command and control servers.

From this article "https://cyware.com/news/what-is-smb-vulnerability-and-how-it-was-exploited-to-launch-the-wannacry-ransomware-attack-c5a97c48" and the following paragraph helped me figure out the protocol that can be used to filter the packets to find the malware.

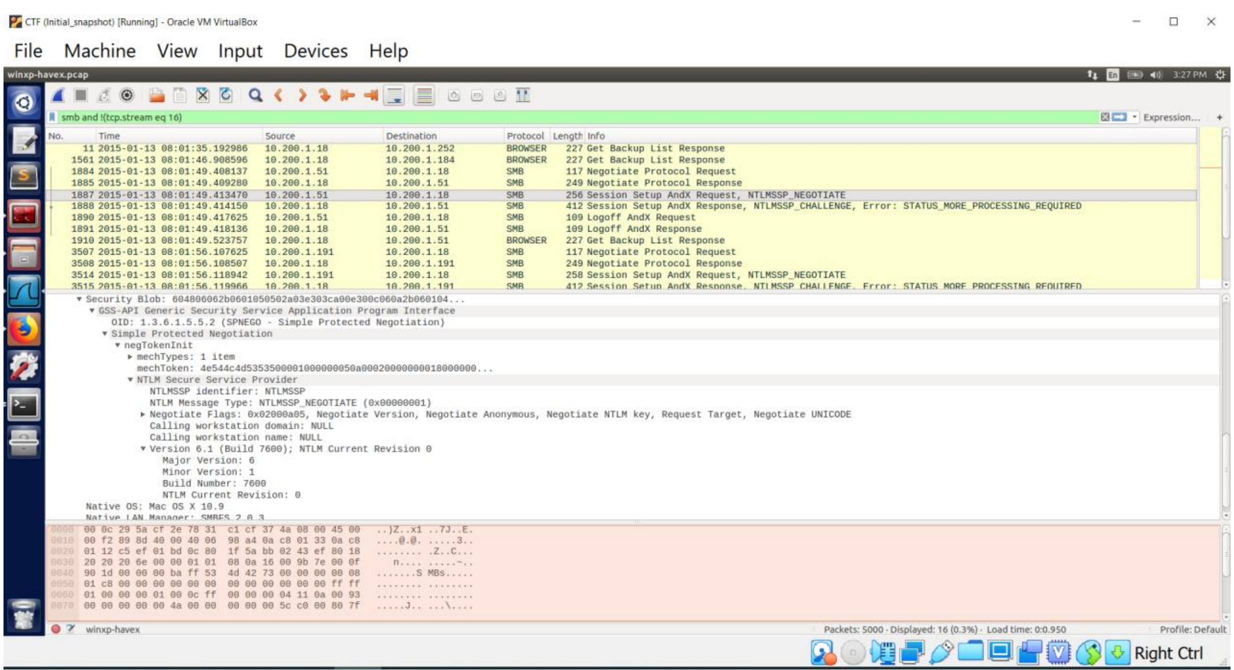
> Server Message Block (SMB) is a file sharing protocol that allows Windows systems connected to the same network or domain to share files. SMB also enables computers to share printers and serial ports from other computers within the same network.
>
> **Vulnerability in SMB version 1.0**
>
> In 2017, the WannaCry ransomware attack exploited a vulnerability in SMB version 1.0 to install malware on vulnerable clients and propagate it across networks.

Then I simply filtered the packet capture file for "smb" protocol and then on following the TCP stream we can see signs of terminal login as highlighted in the below screenshot:



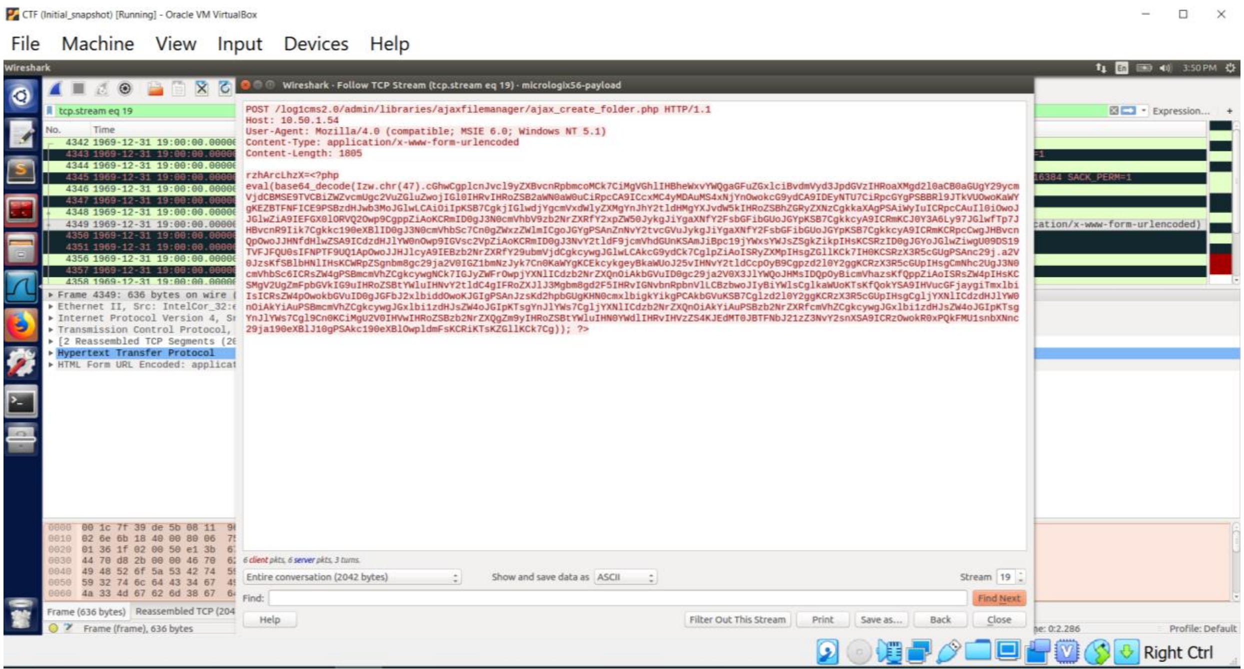
Then I used the following article to understand the SMB authentication mechanism https://richardkok.wordpress.com/2011/02/03/wireshark-determining-a-smb-and-ntlm-version-in-a-windows-environment/ from this article we can understand that this attack takes advantage of NTLMv1 to establish a session with the victim machine.

So, the I.O.C is NTLMSSP.

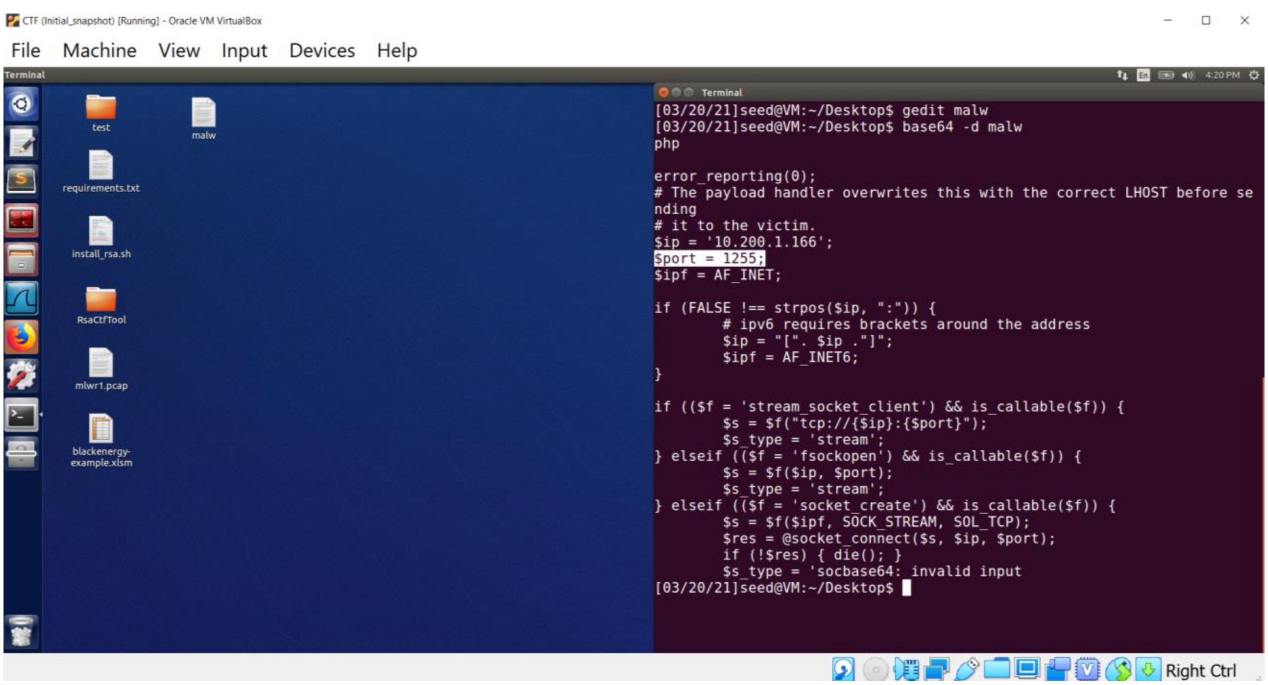


Level 4 – MICROLOGIX56

I started off by filtering packets based on protocols, after some tries on filtering the capture on various protocols; for HTTP I got an interesting POST request which had the following content:



The code consisted of a php header with eval function call which is usually used to run commands using php and in it there was a base64 encoded string. I copied the encoded malware into a file named “malw” and used the base64 util to decode the malware to find the port number being used in the attack.



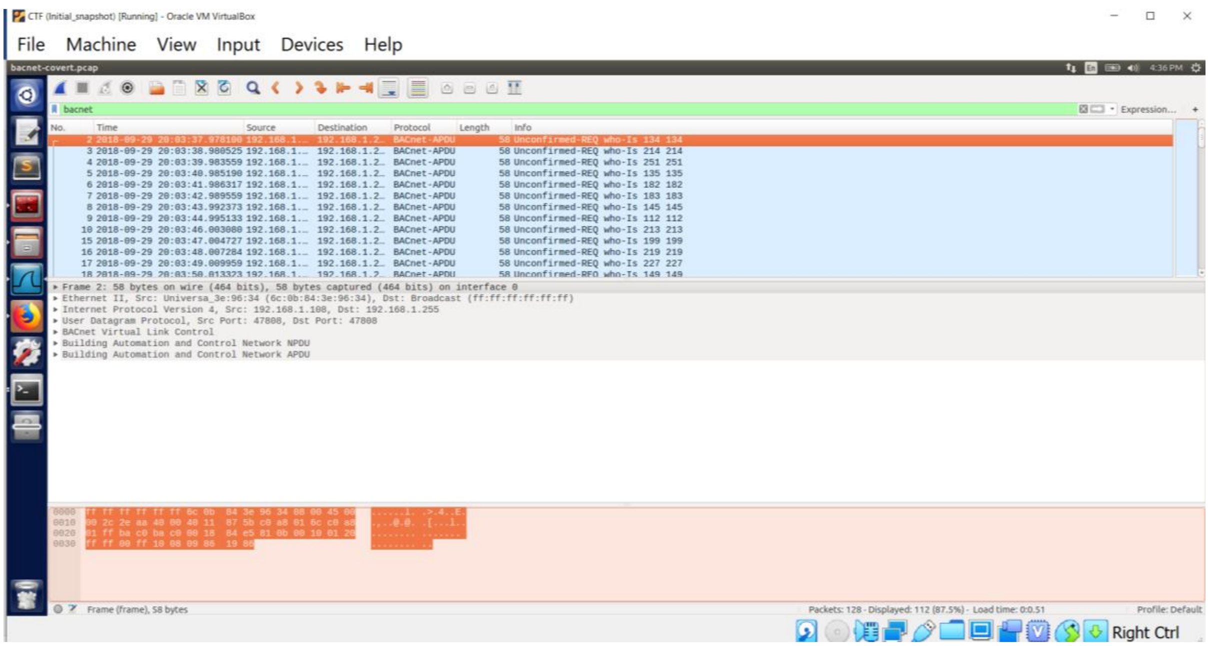
In the above screenshot we can see that the line “$port = 1255;” defines the port number for the attack. Hence the flag is “1255”.

Level 5 – BACNET-COVERT

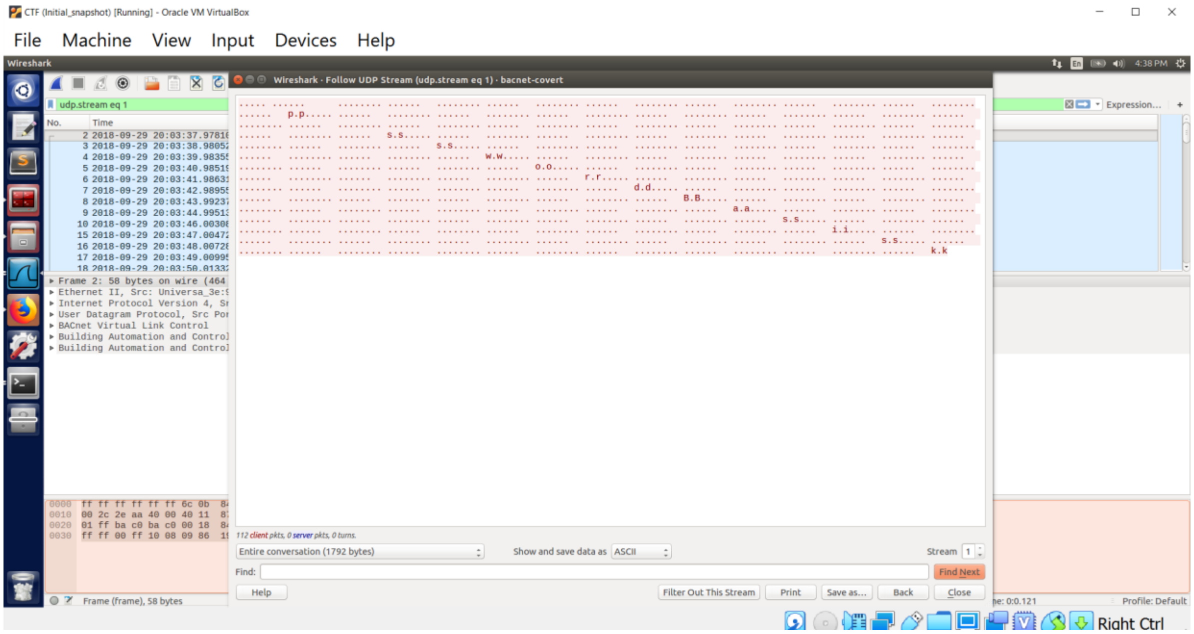
Backnet protocol – “BACnet is "a data communication protocol for building automation and control networks." A data communication protocol is a set of rules governing the exchange of data over a computer network that covers everything from what kind of cable to use to how to form a particular request or command in a standard way. What makes BACnet special is that the rules relate specifically to the needs of building automation and control (BAC) equipment, i.e., they cover things like how to ask for the value of a temperature, define a fan operating schedule, or send a pump status alarm.”

Source: http://www.bacnet.org/Bibliography/EC-9-97/EC-9-97.html

Since the question states that a secret message was being passed using the BACnet communication protocol, I filtered the packet capture log using “bacnet”.



Then I simply selected the option to follow the UDP stream to understand the order of the flow of packets. Which led me to the following output:



On joining the words present in the above stream, I got the following word:

“psswordBasisk”. Which I feel is the covert message.

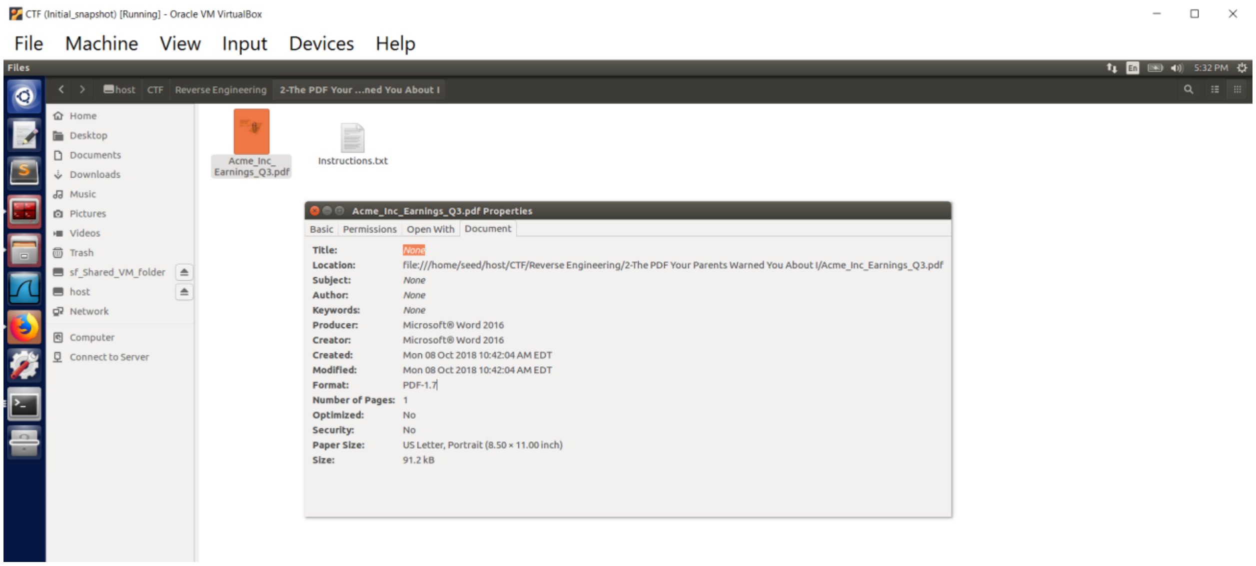
Reverse Engineering

Level 1 – Script Kiddie

On reading the code we can see that “Mid(strPass,12,2) = strIn” is used to check for the valid password, going through the api docs of mid function {**Mid**(*string*, *start*, [*length*])} we can understand that the start of the password is 12th location of the “strPass” string followed up to next 2 characters. So the flag is: “lm”.

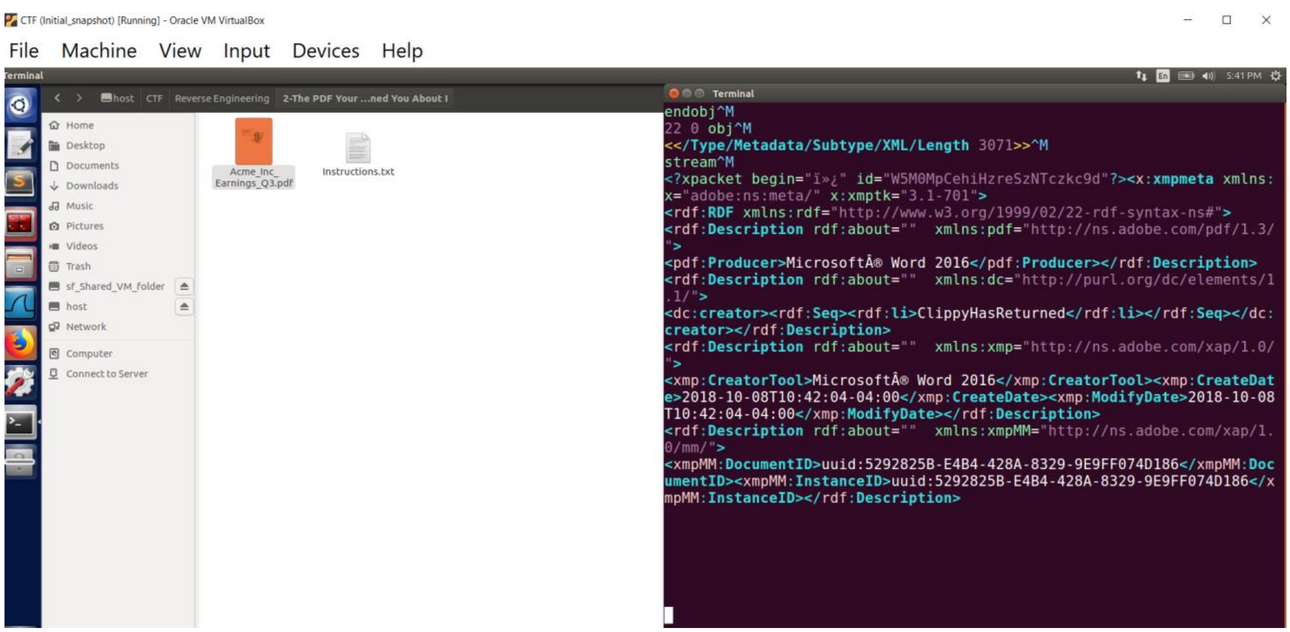
Level 2 – The PDF Your Parents Warned You About I

In this challenge the last line say's "I wonder who created it." Which shows that the challenge is to find the name of the person who created the pdf. Initially I opened the file in order to get the details via the pdf viewer, but that didn't help. I went to the properties of the file in order to get the Author name:



But no data was shown on that either. So, I opened the pdf with vim editor and on scrolling through, I reached to the following section, that contained the name of the creator. "<dc:creator><rdf:Seq><rdf:li>ClippyHasReturned</rdf:li></rdf:Seq></dc:creator></rdf:Description>"

dc:creator = document creator



So, the flag is: "ClippyHasReturned"

Level 3 – Split Ends

For this task we need to find the flag that is compiled in the executable, so a good command to search through the compiled file is the "strings" command. I used the following command to print the output of strings command to a file named "exe_string" so that it becomes easier to find the flag literal.

```
strings re200.exe > exe_string.txt
```

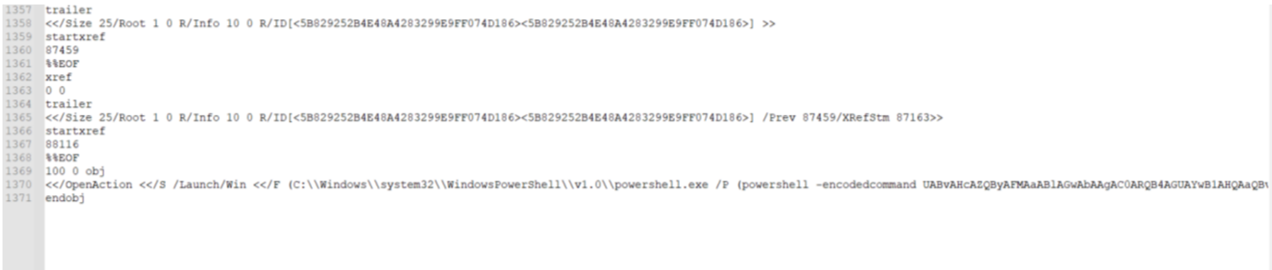
Then searching through the file I found the following section which shows the flag:

Another hint that helped me realize the correct flag was the level name i.e., "split ends".

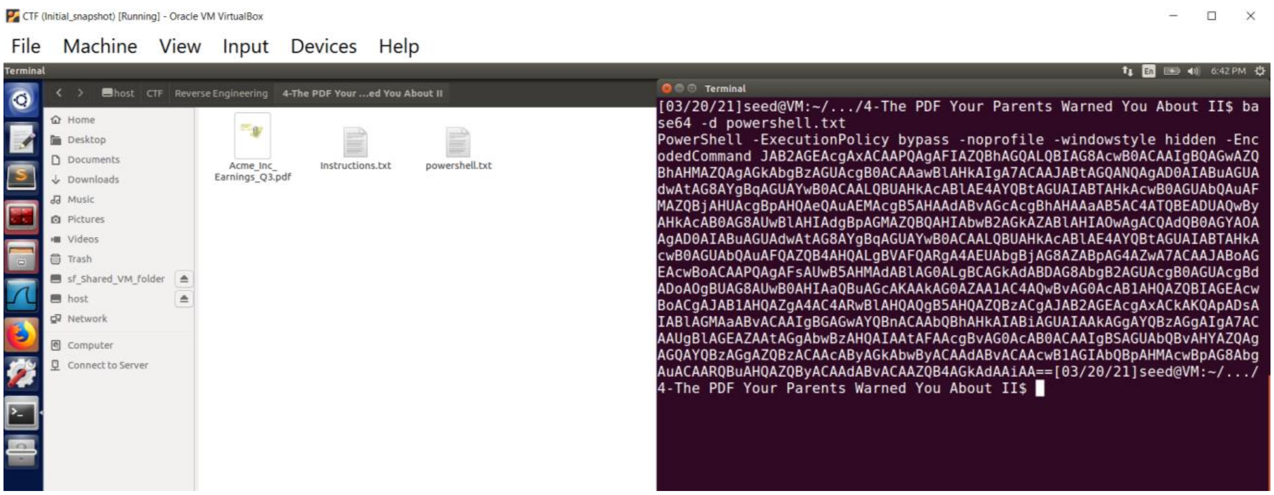
So, the flag for this level is: "infectedflag".

Level 4 – The PDF Your Parents Warned You About II

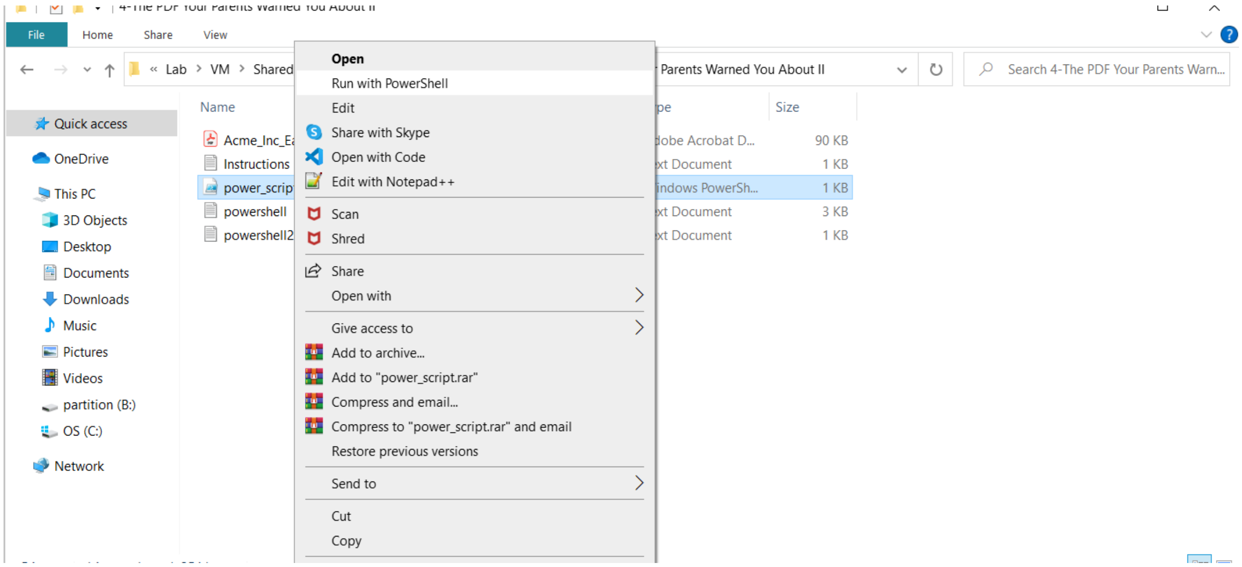
I opened the pdf using vim editor but wasn't able to find anything interesting. So, for this task I used another tool named "notepad++". While glancing through the pdf data in notepad++ I saw a Powershell launch command with some encoded string:



On decoding the string I got another encoded string as shown below:



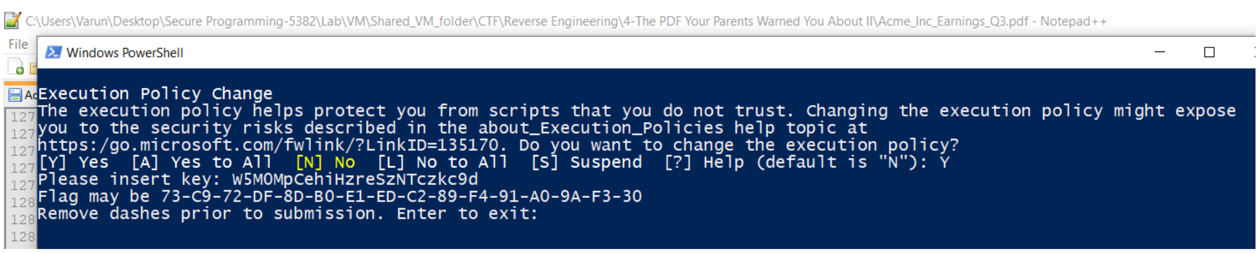
I again saved the encoded program into a file and decoded it, now I got a program as output, since in the "OpenAction" tag we saw that the command was trying to open a windows Powershell, I created a Powershell script of the above program and executed it in the Powershell terminal.



I was asked for a key input, for which I searched through the given pdf and tried various strings found in the tags, finally found the id field to be the key for the program.

```
endobj
22 0 obj
<</Type/Metadata/Subtype/XML/Length 3071>>
stream
<?xpacket begin="ï»¿" id="W5M0MpCehiHzreSzNTczkc9d"?><x:xmpmeta xmlns:x="adobe:ns:me
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Description rdf:about=""  xmlns:pdf="http://ns.adobe.com/pdf/1.3/">
<pdf:Producer>MicrosoftÂ® Word 2016</pdf:Producer></rdf:Description>
<rdf:Description rdf:about=""  xmlns:dc="http://purl.org/dc/elements/1.1/">
<dc:creator><rdf:Seq><rdf:li>ClippyHasReturned</rdf:li></rdf:Seq></dc:creator></rdf:
<rdf:Description rdf:about=""  xmlns:xmp="http://ns.adobe.com/xap/1.0/">
```

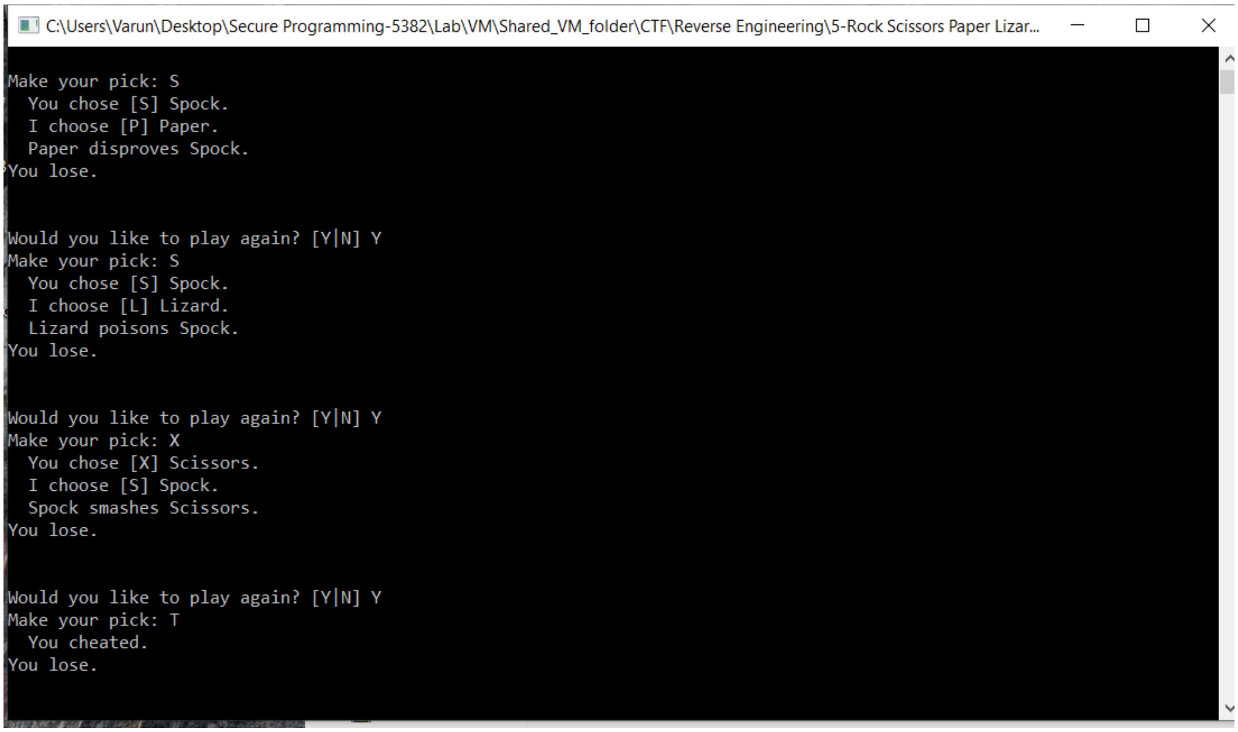
I got the following output which had the flag in it.



The flag: “73C972DF8DB0E1EDC289F491A09AF330”, I removed all the dashes because the in the output it says remove dashes prior submission.

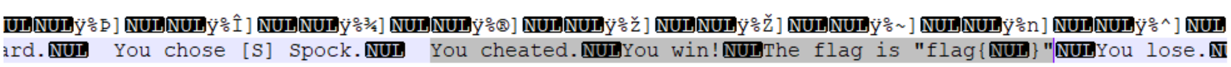
Level 5 – Rock Scissors Paper Lizard Spock

Initially I tried playing the game to see if the executable actually works. Then I tried entering various inputs as the terminal prompted but couldn't defeat the game. I also tried entering other strings just to crash the program but I couldn't.



So, I opened the program in notepad++ to analyze the code, searched for the term "flag" and found the following statement:

"You cheated. You win! The flag is "flag{ }""

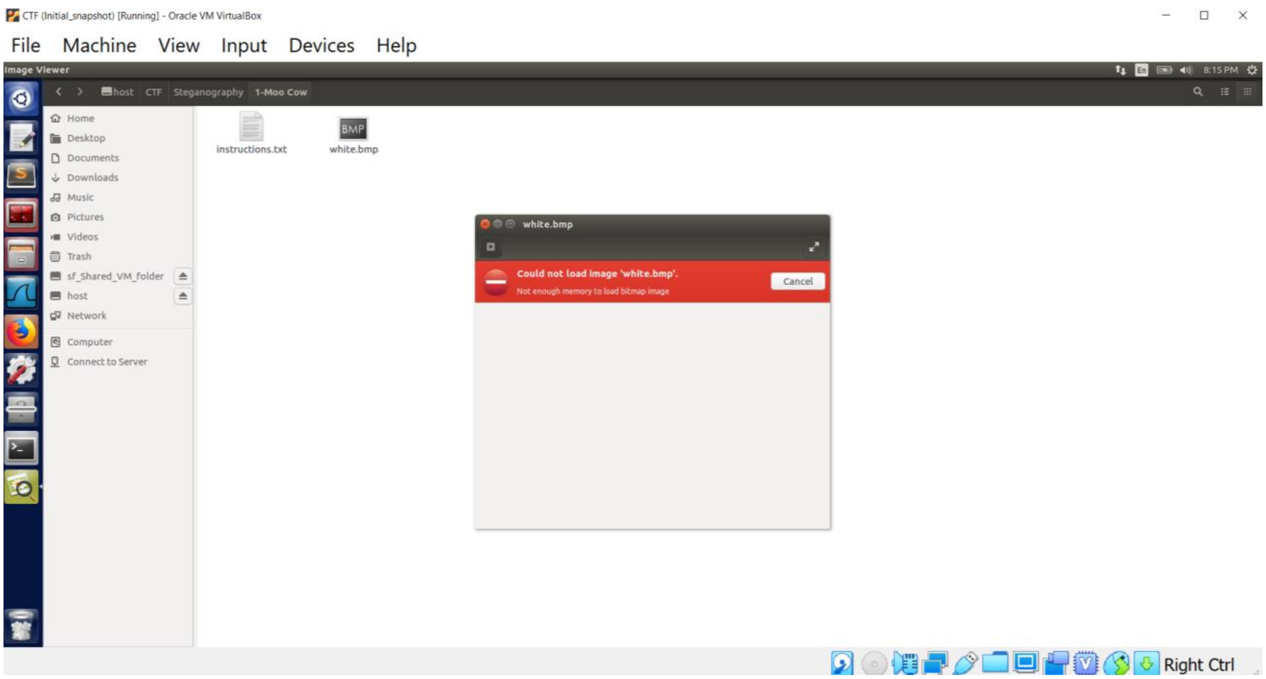


So, the flag for this task is: "flag{ }".

Steganography

Level 1 – Moo Cow

On opening the file I was getting the following error:

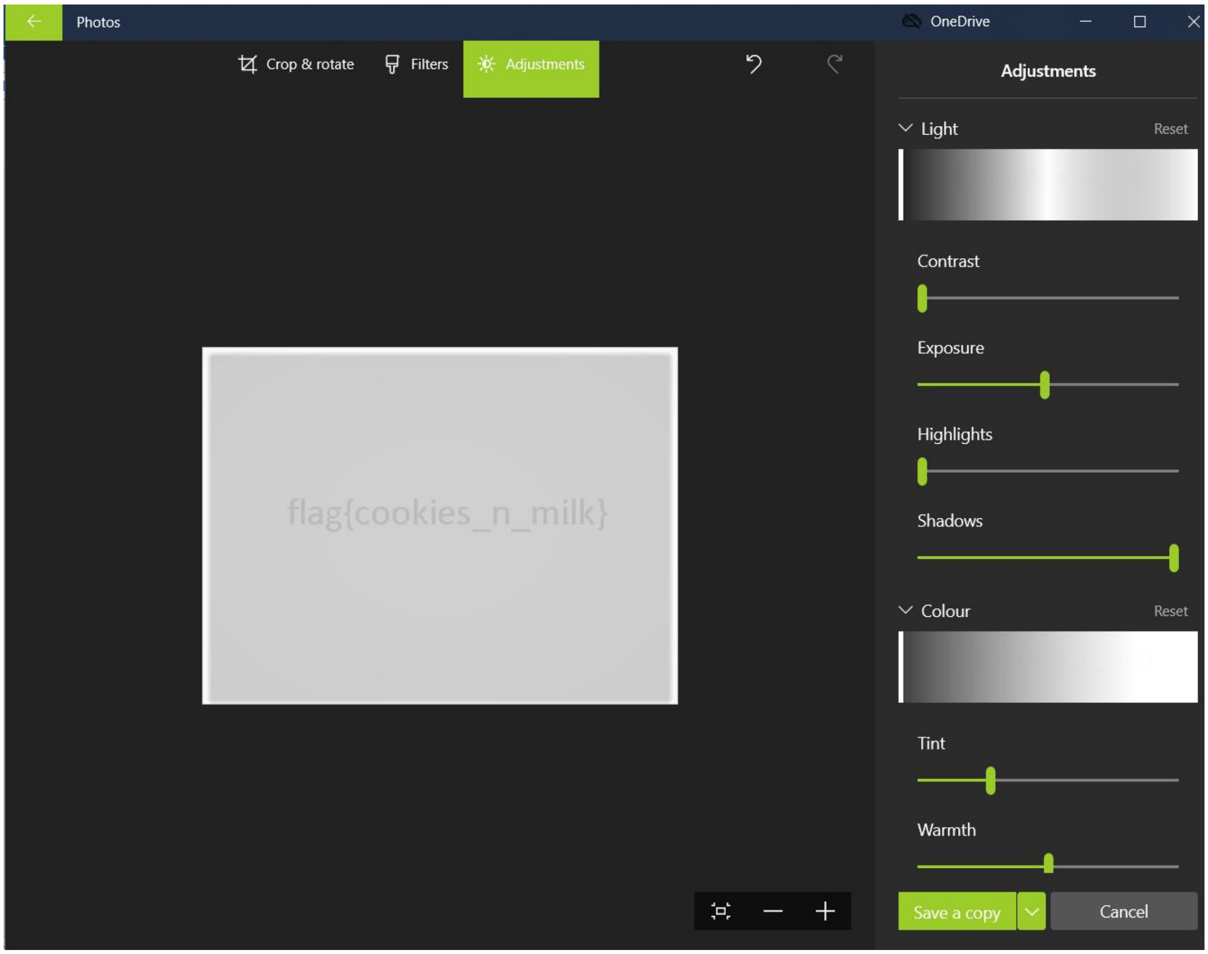


So, I opened the file using "Gedit" to check the contents of the file, where I found the flag.

For this task the flag is: "flag{covert_channel}"

Level 2 – Milk Run

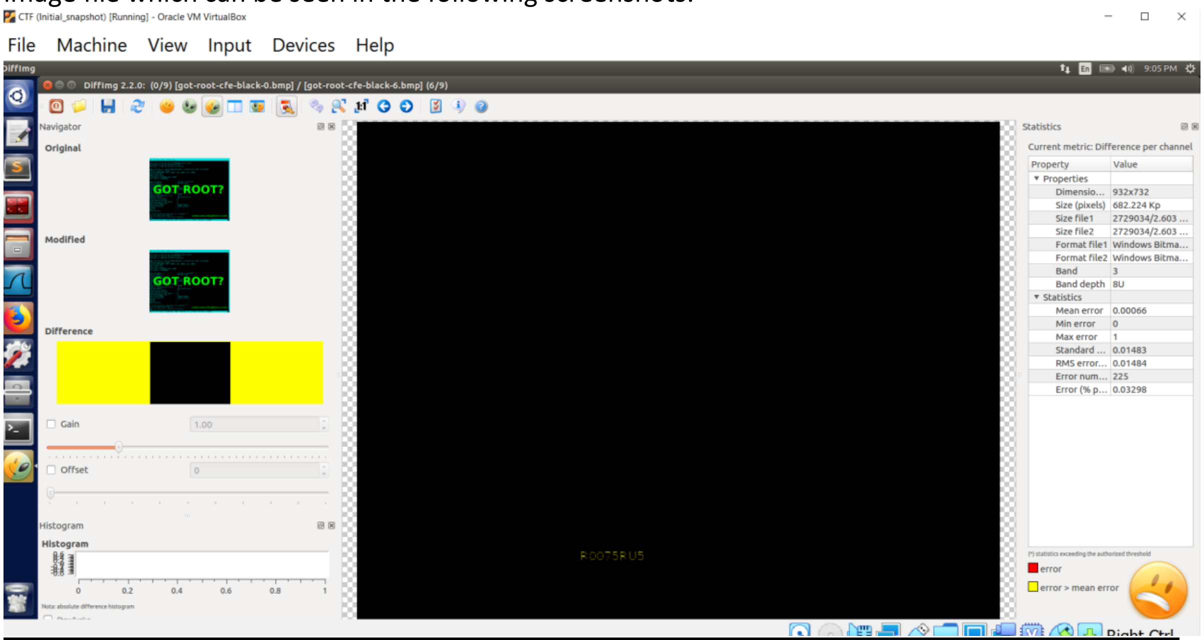
Based on the last task I tried opening the file using Gedit and vim but nothing special was found so I opened the image and it actually opened showing a white screen. Since the question states that the flag is hidden in the image, I tried editing the filters of image to find if any pixel might highlight the flag and thus found the flag.

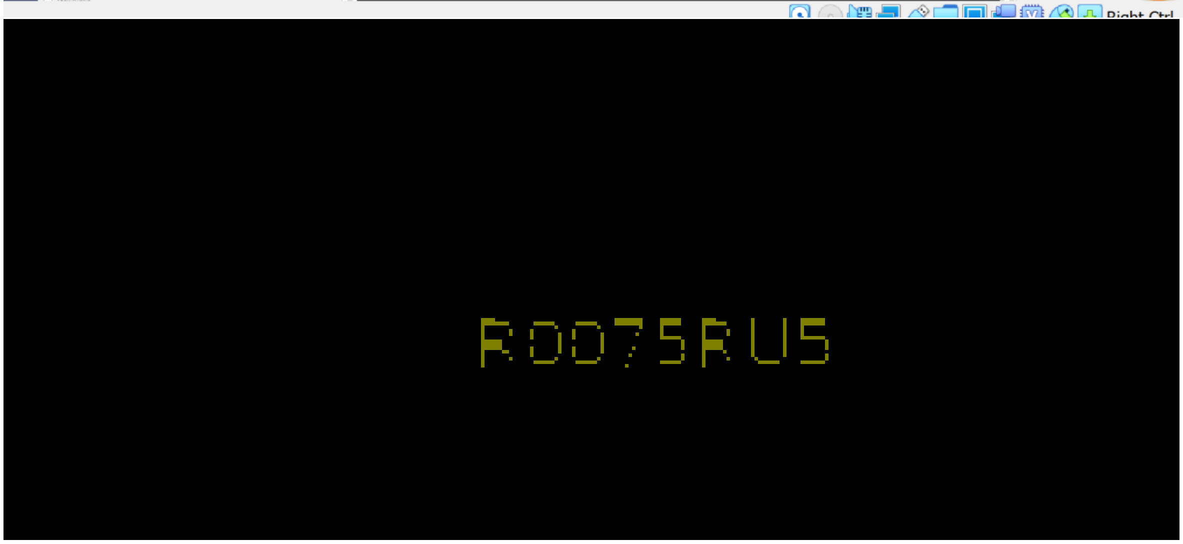


The flag: “flag{cookies_n_milk}”

Level 3 – Got Root

On extracting the given zip file, I saw 10 files named “got-root-cfe-black-*.bmp”. Opening the images one by one I found them to be visually similar. Since this was a task related to steganography on reading a little about it, I got to know about a technique where in the image differed at pixel level. So, I went through the list of tools mentioned in the “Tools” word file and found a tool named “DiffImgPortable” and installed the same. While comparing all the images, a diffrence was found in the 0th and the 6th image file which can be seen in the following screenshots:

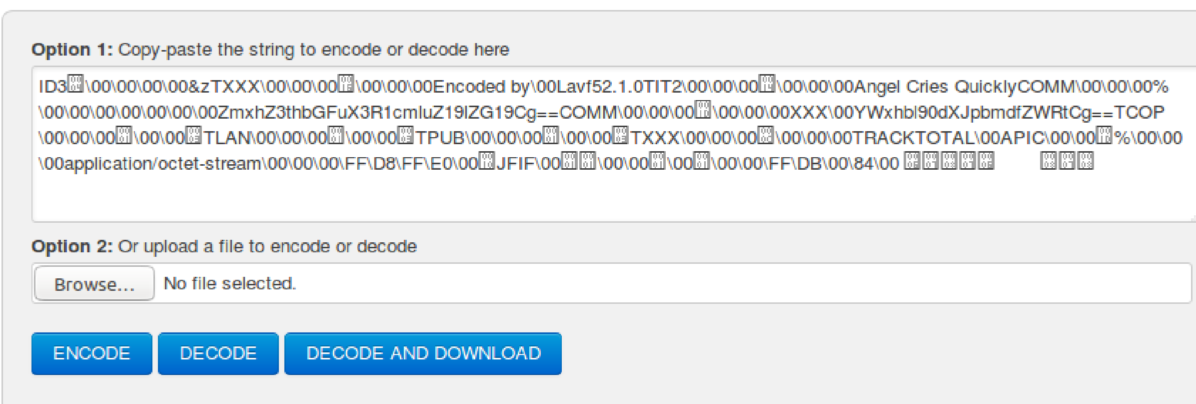


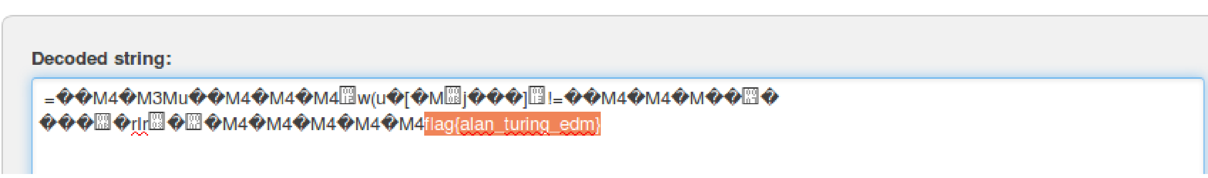


So, the flag for this task is: “ROO7SRUS or R0075RU5”.

Level 4 – AI EDM

On playing the mp3, it played normally and nothing suspicious occurred. I opened the file using gedit to search for terms like “flag, flag{ ” but nothing interesting popped up. So, I searched for special strings that might be present in the file using the strings util function of bash, but still couldn’t find any interesting string that would highlight the presence of flag. On carefully observing the characters I realized that there were some encoded strings present at the start of the file. So, I simply copy pasted the entire paragraph on a online decoder, post the decoding I found the flag:

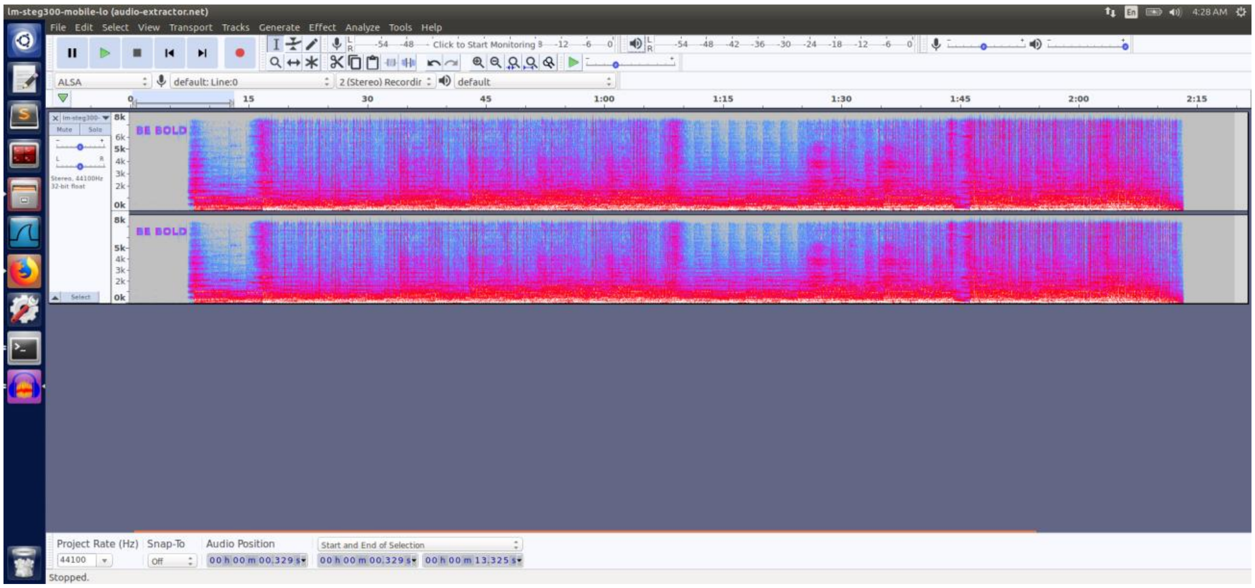


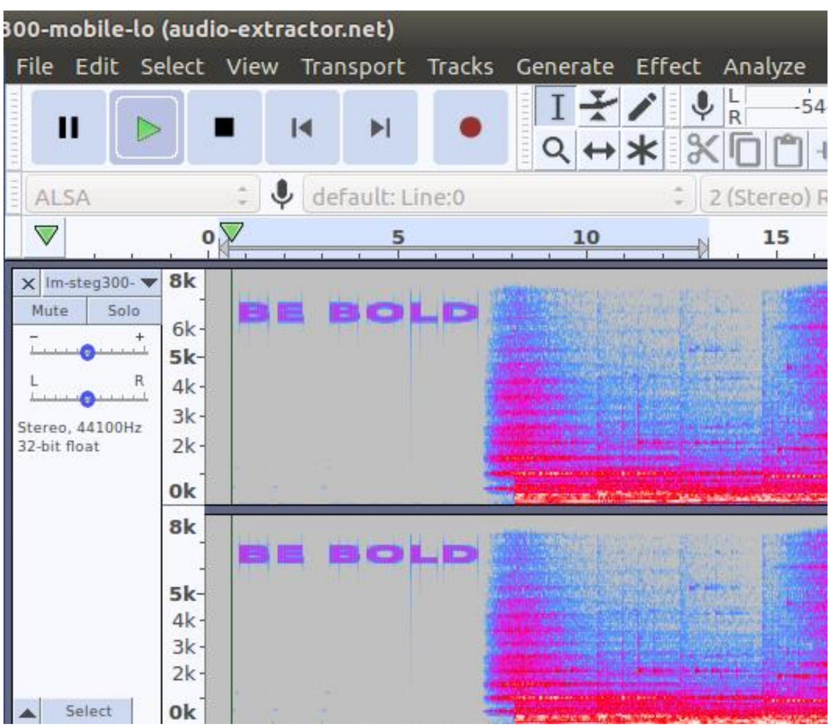


So, the flag for this task is: “flag{alan_turing_edm}”

Level 5 – Final Frontier

On looking at the video we can see that the video was a little blurred so there can be a possibility of pixel manipulation. On looking at the properties of the file I saw this encoded string “bm9wZQ==” which decodes to “nope” and going as per the information provided by the professor, we can say that this is not our required flag. As professor highlighted that the flag can also be in the audio file, I extracted the audio file using an online mp3 extractor. To analyze the audio file, I downloaded audacity. Using audacity I analyzed the waveforms of the audio and selecting the “spectrogram” option I got the following output:





And we can see “Be Bold” at the start of the waveform, which I think is the flag for this task.