COMPUTER NETWORK SECURITY

Name: Duha Masoodi

ID: H00393274

Introduction:

In the following coursework, we are given four tasks based on different key encryption and some common attacks on encryption. The environment that we used for this coursework is Linux.

The following things were covered by this coursework:

- Frequency analysis to decode a cipher text
- Padding, to encrypt files properly
- Corrupting a ciphertext changes the output for different modes.
- Cryptanalysis with or without padding

TASK 1: Frequency Analysis: Monoalphabetic Substitution Cipher:

In this task we were given a ciphertext which was encrypted using monoalphabetic cipher, in which each letter was substituted by another different letter. The following steps were used to decrypt the cipher:

First step:

I found monograms, bigrams and trigrams with highest frequency, and they came out to be:

1) **S, Q, I, D, E**
2) **EQ, EV, IL, DV, ER, QI, GI, EG, DQ, JS, DR, IR, EX, EL, XI, QY, YS, ST.**
3) **QYS, ERA, DRG and MIV**

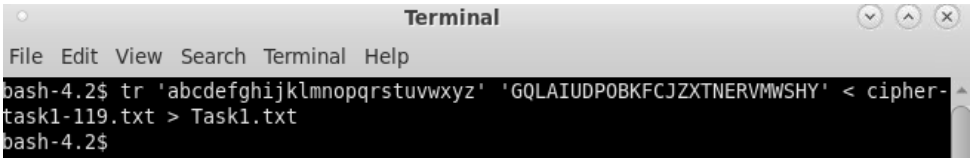
Second step:

The frequency of the monograms like **S, D, E** are the highest and when we compare them to the most frequently occurred English letter words, I came up with the conclusion that these letters are basically **E**, **A** and **I**.

Then after guessing the monograms, I checked the most occurred bigrams were **EV** and **DV** that helped me finding **V**, **EQ** and **DQ** which further helped me in finding **Q** and so on. After I used the letters like **v** and **q** and found more bigrams involving them to guess more letters. Finally, I was left with trigrams which I could easily guess by using these monograms and bigrams

Third Step:

The following command was used for decryption:



My encryption table:

Ciphertext	key
A	g

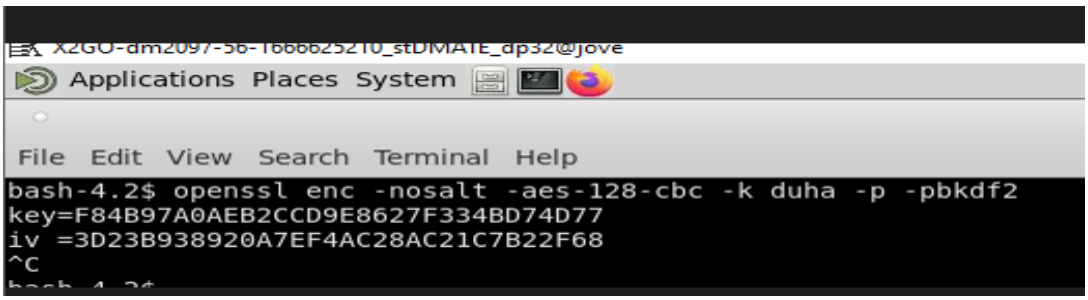
B	q
C	l
D	a
E	i
F	u
G	d
H	p
I	o
J	b
K	k
L	f
M	c
N	j
O	z
P	x
Q	t
R	n
S	e
T	r
U	v
V	m
W	w
X	s
Y	h
Z	y

Task 2: Symmetric encryption: Padding:

Padding is required when the plaintext is not the multiple of the block size. In this task we used aes-128-cbc to encrypt a 5-byte, 10 byte and 16-byte files. The block size of AES-128 is 16 bytes. So, we need to pad the 5-byte file with 11 bytes, 10-byte file with 6 bytes and for 16 bytes file padding is not required as the file size is equal to 16 bytes, but it was still padded and it was done by more 16 bytes, overall, 32 bytes.

The following steps are shown below:

1.Generation of key and iv for my files:



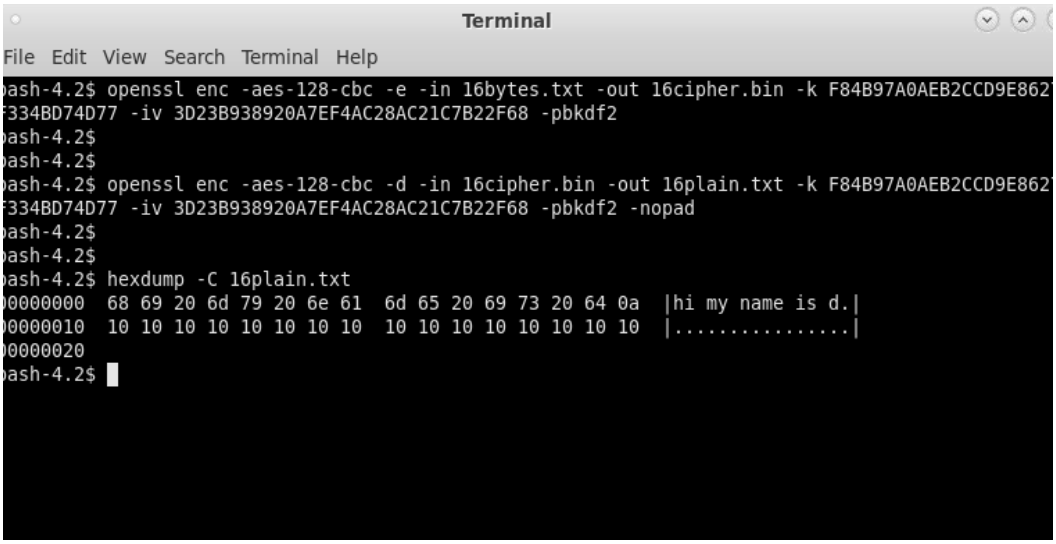
2.Padding for 5byte file:

```
bash-4.2$ openssl enc -nosalt -aes-128-cbc -k duha -p -pbkdf2
key=F84B97A0AEB2CCD9E8627F334BD74D77
iv =3D23B938920A7EF4AC28AC21C7B22F68
^C
bash-4.2$
bash-4.2$ openssl enc -aes-128-cbc -e -in 5bytes.txt -out 5cipher.bin -k F84B97A0AEB2CCD9E8627F334BD74D77 -iv 3D23B938920A7EF4AC28AC21C7B22F68 -pbkdf2
bash-4.2$
bash-4.2$
bash-4.2$ openssl enc -aes-128-cbc -d -in 5cipher.bin -out 5plain.txt -k F84B97A0AEB2CCD9E8627F334BD74D77 -iv 3D23B938920A7EF4AC28AC21C7B22F68 -pbkdf2 -
bash-4.2$
bash-4.2$ hexdump -C 5plain.txt
00000000  31 32 33 34 0a 0b 0b 0b  0b 0b 0b 0b 0b 0b 0b 0b  |1234............|
00000010
```

3.Padding for 10byte file:

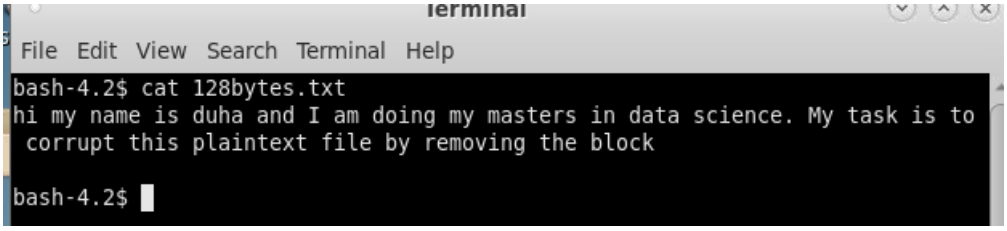
```
bash-4.2$ openssl enc -aes-128-cbc -e -in 10bytes.txt -out 10cipher.bin -k F84B97A0AEB2CCD9E8627
F334BD74D77 -iv 3D23B938920A7EF4AC28AC21C7B22F68 -pbkdf2
bash-4.2$
bash-4.2$ openssl enc -aes-128-cbc -d -in 10cipher.bin -out 10plain.txt -k F84B97A0AEB2CCD9E8627
F334BD74D77 -iv 3D23B938920A7EF4AC28AC21C7B22F68 -pbkdf2 -nopad
bash-4.2$
bash-4.2$
bash-4.2$ hexdump -C 10plain.txt
00000000  68 69 20 6d 79 20 6e 61  6d 0a 06 06 06 06 06 06  |hi my nam.......|
00000010
bash-4.2$
```

4.Padding for 16byte file:



Task 3: Encryption Mode — Corrupted Cipher Text:

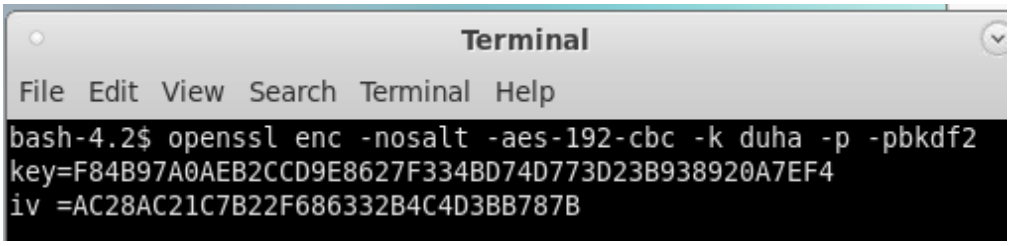
1.CREATION OF 128BYTES.TXT:



2.Generation of KEY and IV:

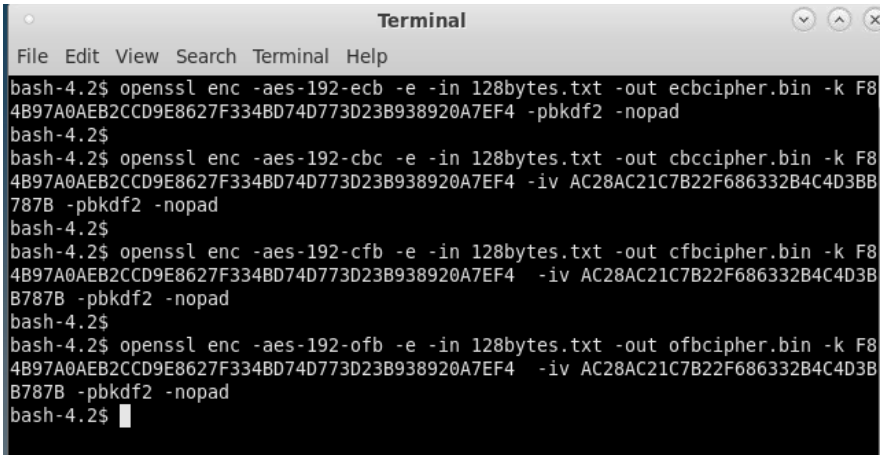
I used the same key and IV for all the modes.

ECB only requires key



3.ENCRYPTION:

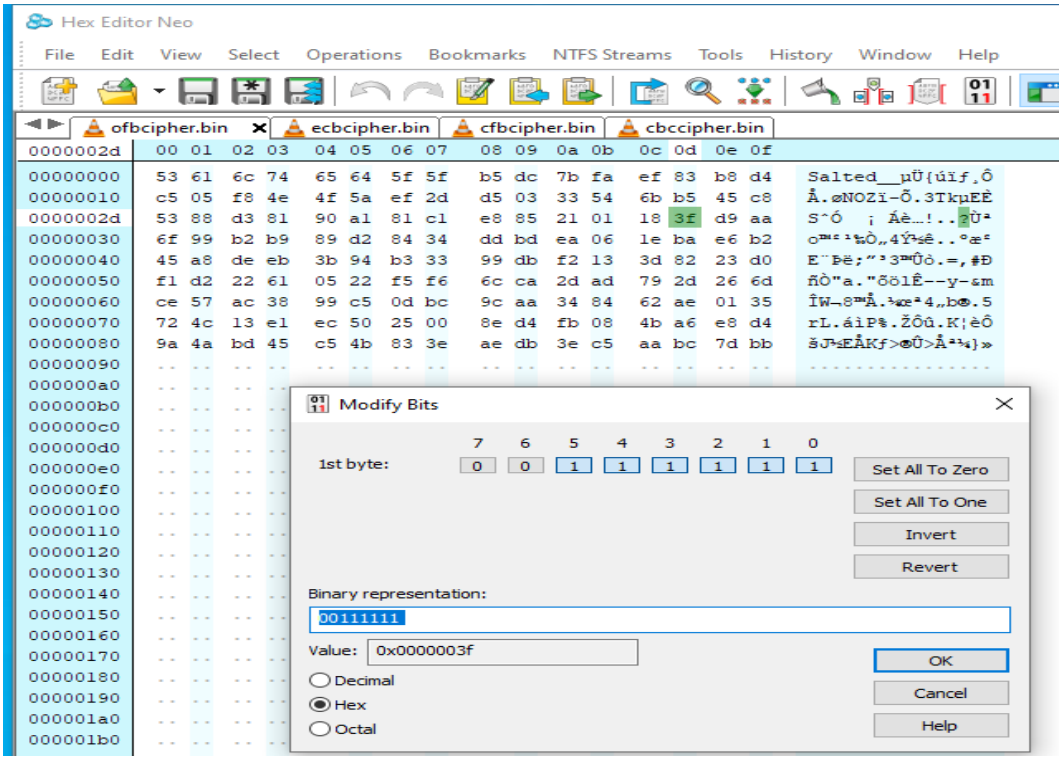
The encryption of the file was done with the following modes:



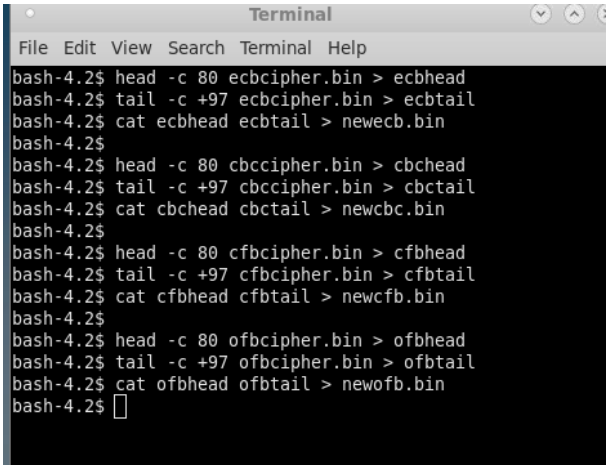
4.CORRUPTION OF CIPHERTEXT:

For the corruption of ciphertext file, first we removed the 1 bit of 46th byte in hex editor and after that we removed the 6th block which contained the 86th byte. It was done as follows:

a. Changing 1 bit of 46th byte:

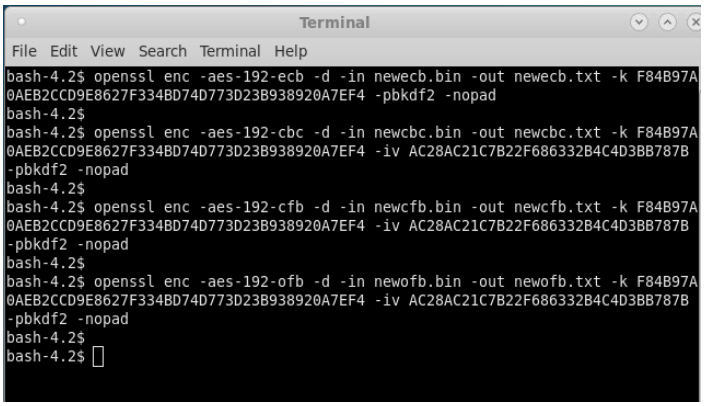


b. Removing block containing 86th byte:

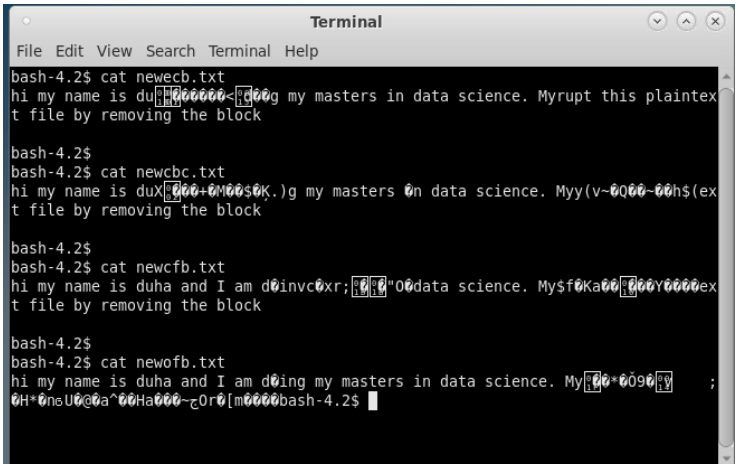


5.DECRYPTION:

Following commands were used for the decryption of the corrupted ciphertext.



6.RESULTS:



The results are summarized as follows:

- ECB MODE: This mode takes a key and encrypts every block with the same key. So, after the corruption of ciphertext file it is clearly visible only that block has been affected, rest is same as the original plaintext. This mode is the weakest and is not recommended.
- CBC MODE: In this mode we use an initialization vector with the key. The plaintext is XOR with the iv and then encrypted with the help of key. So, after corrupting the ciphertext we saw that the block followed by the missing block was also corrupted. This mode is somewhat secure but is vulnerable to padding attacks
- CFB MODE: It will first encrypt the IV with the key and then XOR it with the plaintext. It is quite like CBC mode. Even in the results it is seen that the following block was corrupted in this also.
- OFB MODE: In this mode the iv is encrypted first then the encryption results are XOR with the plaintext to produce cipher. In this method if the block is broken it wont be affected but for missing block it will corrupt the whole ciphertext. It is the most secure mode than all other three.

Task 4: Ciphertext-only cryptanalysis, with and without padding:

The main aim of the code was to get the plaintext and key from the given ciphertext. The key had to be less than 16 letters and appended by digit 0-9. It is an example of brute force attack, where we generated the entire key and recovered the plaintext.

I achieved the following by creating a text file and then encrypting it by aes-128-cbc method. After that I created a small file with the key and a word from the plaintext file. After that I encrypted the plain text file to generate a ciphertext file to be used for my code.Then I used the following code to generate all the keys appended by 0-9 numbers and assigned them as the password for the decryption method. After decryption I used a condition if words in text files matches word.txt. It then printed the plaintext with its key.

CODE:

The code to find the plain text from the given ciphertext is showed as follows:

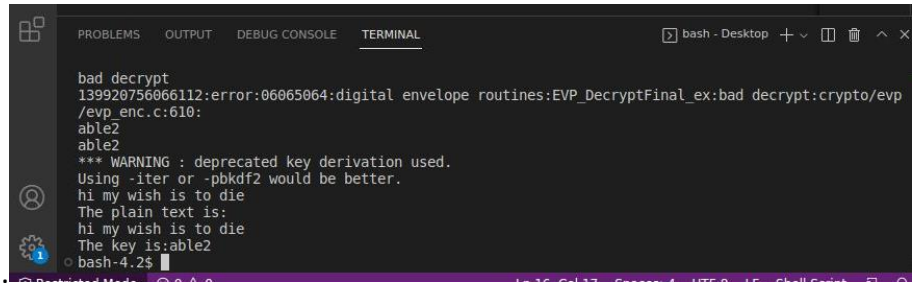
```
declare -a arr1=(0 1 2 3 4 5 6 7 8 9)

read -r -p "Enter your cipher text file:" cipher

read -r -p "Enter your word file:" words

 while read x;

do

   for y in "${arr1[@]}";

   do

      join=$x$y

      echo $join

      size=${#join}

      if [ "$size" -lt 16 ];  #the size of the password less to be than 16

      then

         echo $join

         openssl enc -aes-128-cbc -d -in $cipher -out plain.txt -pass pass:$join
```

```
        plain=plain.txt
        if cat $plain|grep -f $words ; #checking if the words in plaintext matches with the words in words.txt
        then
          echo "The plain text is:"
           cat $plain     #prints the plaintext
           echo "The key is:$join" #prints the key
           exit
        fi
      fi
   done
done <$words
```

Output:

1.With Pad:

Padding increased the security of file . Although I got the results in the end but it iterated through each key and then it found the result. It was more time consuming.



2.Without Pad:

When removing the padding option, it was able to find the key faster without iterating through any other value making it less secure.

Plain-text attack:

In the plain-text attack, we have been given a ciphertext and corresponding to it a plaintext. In this task I followed the previous method only, i.e. ciphertext only attack. It was quicker and easier to break as half of the text was already given and we had to just guess one word. No changes in the code were required.

Conclusion:

The following conclusions were made after completion of all the tasks:

1. Frequency analysis can be used for monoalphabetic substitution. It helps us to decode the ciphertext easily by just guessing the frequent monograms and bigrams.
2. Padding is an essential component if we want to encrypt a file small enough for a particular mode because it prevents it from cryptanalysis and doesn't give the exact size of the file.
3. Corrupting a ciphertext for different modes showed which mode is the most effective and preventive against attacks.
4. In this task I performed a cipher-text only attack without knowing the key or the plain text. This task was the most challenging as my files were working for the small test word

file, but it didn't work for the large word text file. So, I created a new file using head tail that contained the first 50 words of the file and to my surprise it worked.

APPENDIX:

TASK1:

IRMED I CAN WATCH IT

REPLICATE TRY KILLING THE JOBS THEYLL COME RIGHT BACK

FROM WHERE

IM GETTING CONNECTIONS FROM FIVE PLACES STANFORD UNIVERSITY OF ROCHESTER

AEROSPACE COMPANY THE BERKELEY CAMPUS AND SOMEWHERE CALLED BRL

THATS THE ARMYS BALLISTICS RESEARCH LAB I SAID REMEMBERING A CONVERSATION

WITH BRLS MIKE MUUSS HOWS THE VIRUS GETTING INTO YOUR SYSTEM

I CANT TELL CLIFF THE CONNECTIONS ARE ALL FROM THE ARPANET BUT ITS NOT

THE USUAL WAY OF LOGGING INTO THE SYSTEM LOOKS LIKE THE VIRUS IS BREAKING IN

PAGE OF

THROUGH A HOLE IN THE MAIL SYSTEM

SOMEONES BUILT A VIRUS THAT EXPLOITS A SECURITY HOLE IN UNIX SYSTEMS THE HOLE

IS IN THE MAIL SYSTEM AND THE VIRUS SPREADS OVER THE NETWORK WHATS THE VIRUS

DOING JUST COPYING ITSELF OR DOES IT HAVE A TIME BOMB BUILT IN

ITS AM WHAT TO DO ID BETTER CALL THE ARPANET CONTROLLERS AND WARN THEM

THERES A TWENTYFOURHOUR DUTY OFFICER AT THE NETWORK OPERATIONS CENTER THAT

WATCHES OVER THE NETWORK THIS MORNING THEYVE HEARD NOTHING OF THIS VIRUS

BETTER CALL AROUND BECAUSE ITLL BE ALL OVER THE PLACE BY NINE THIS MORNING

THE NETWORKS OPERATIONS CENTER HASNT HEARD THE VIRUS IS ONLY A FEW HOURS OLD

IM SEEING VIRUSES COMING FROM A DOZEN OTHER SITES VIRULENT BY MORNING IT WILL

HAVE SPREAD TO SCORES OR EVEN HUNDREDS OF SYSTEMS WEVE GOT A PROBLEM A MAJOR

PROBLEM

AN EPIDEMIC

WEVE GOT TO UNDERSTAND THIS VIRUS AND SPREAD THE WORD FOR THE NEXT THIRTYSIX HOURS I KNOCKED MYSELF OUT TRYING TO UNDERSTAND AND DEFEAT THIS THING I KNEW I WASNT ALONE AT THE SAME TIME GROUPS AT BERKELEY MIT AND PURDUE UNIVERSITY WERE ALREADY HOT ON THE TRAIL

HERE IM ONLY DESCRIBING WHAT I SAW BUT MY STRUGGLE WAS MINOR COMPARED TO THE WORK OF UNIX WIZARDS ACROSS THE COUNTRY ONE BY ONE PROGRAMMERS REACTED GURUS LIKE KEITH BOSTIC PETER YEE GENE SPAFFORD JON ROCHLIS MARK EICHIN DONN SEELEY ED WANG AND MIKE MUUSS I WAS BUT A SMALL PART OF AN UNORGANIZED BUT DEDICATED RESPONSE TO THIS DISASTER

I DIG INTO THE CODE IN MY SYSTEM IN CAMBRIDGE RIGHT OFF I CAN SEE TWO VERSIONS OF THE VIRUS ONES CUSTOMIZED FOR VAX COMPUTERS RUNNING UNIX THE OTHERS FOR SUN WORKSTATIONS EACH FILE IS FORTYFIVE THOUSAND BYTES LONG IF IT WERE ENGLISH IT WOULD FIT IN ABOUT THIRTY PAGES BUT ITS NOT TEXT I DUMP THE FILE AND IT LOOKS LIKE GIBBERISH IT DOESNT EVEN LOOK LIKE MACHINE CODE

NOW THIS DOESNT MAKE SENSE COMPUTER PROGRAMS LOOK LIKE MACHINE CODE THIS ONE DOESNT THERES NO HEADER BLOCK INFORMATION AND ONLY A FEW COMMANDS THAT I RECOGNIZE THE REST IS GUACAMOLE

PATIENTLY I TRY TO UNDERSTAND WHAT THOSE FEW COMMANDS DO SUPPOSE I WERE A SUN WORKSTATION AND SOMEONE FED THOSE COMMANDS TO ME HOW WOULD I RESPOND WITH A PAD OF PAPER HAND CALCULATOR AND A BOOKLET OF MACHINE INSTRUCTIONS I START UNWINDING THE VIRUSS CODE

THE FIRST FEW COMMANDS JUST STRIP OFF SOME ENCRYPTION FROM THE REST OF THE VIRUS THATS WHY THE VIRUS LOOKS STRANGE THE ACTUAL COMMANDS HAVE BEEN PURPOSELY OBSCURED

AHA THE VIRUS WRITER HAS HIDDEN HIS VIRUS HES TRIED TO PREVENT OTHER PROGRAMMERS FROM UNDERSTANDING HIS CODE THROWING NAILS ON THE ROAD TO SLOW DOWN HIS PURSUERS

DIABOLICAL

TIME TO CALL DARREN AGAIN ITS AM AND WERE COMPARING NOTES HES DISCOVERED THE SAME THING AND MORE IVE UNMASKED PART OF THE VIRUS AND I CAN SEE

ITS BREAKING IN THROUGH THE MAIL SYSTEM THEN IT USES FINGER AND TELNET TO SPREAD ITSELF TO OTHER COMPUTERS ITS DECRYPTING PASSWORDS BY BRUTE FORCE GUESSING

TOGETHER OVER THE PHONE WE PRY APART THE PROGRAM ITS WHOLE PURPOSE SEEMS TO BE TO COPY ITSELF INTO OTHER COMPUTERS IT SEARCHES FOR NETWORK CONNECTIONS NEARBY COMPUTERS DISTANT SYSTEMS ANYTHING THAT IT CAN REACH

WHENEVER THE VIRUS PROGRAM DISCOVERS A COMPUTER ON THE NETWORK IT TRIES TO

BREAK INTO IT USING SEVERAL OBSCURE HOLES IN THE UNIX OPERATING SYSTEM

HOLES IN UNIX SURE

WHEN YOU SEND MAIL FROM ONE UNIX COMPUTER TO ANOTHER THE UNIX SENDMAIL PROGRAM HANDLES THE TRANSFER A MAIL MESSAGE ARRIVES FROM THE NETWORK AND SENDMAIL FORWARDS IT TO THE ADDRESSEE ITS AN ELECTRONIC POST OFFICE THAT PIGEONHOLES MAIL

SENDMAIL HAS A HOLE NORMALLY A FOREIGN COMPUTER SENDS MESSAGES INTO THIS PROGRAM AND EVERYONES HAPPY BUT IF THERES A PROBLEM YOU CAN ASK THE PROGRAM TO ENTER DEBUG MODE THE PROGRAMS BACK DOOR

WHEN YOURE IN DEBUG SENDMAIL LETS YOU ISSUE ORDINARY UNIX COMMANDS FROM A FOREIGN COMPUTER COMMANDS LIKE EXECUTE THE FOLLOWING PROGRAM

SO THATS HOW THIS VIRUS SPAWNED COPIES IT MAILED COPIES OF ITSELF TO OTHER COMPUTERS AND COMMANDED THEM TO EXECUTE THE VIRUS PROGRAM

AFTER THE VIRUS PROGRAM STARTED IT SEARCHED FOR OTHER COMPUTERS TO INFECT AND SENT MAIL MESSAGES TO THEM

ON SOME SYSTEMS SENDMAIL HAD BEEN FIXED IF SO THE VIRUS TRIED YET ANOTHER HOLE THE FINGER DAEMON

TO SEE IF IVE BEEN USING A UNIX SYSTEM YOU CAN ISSUE THE COMMAND FINGER CLIFF IF IVE BEEN LOGGED IN UNIX WILL RESPOND WITH MY NAME PHONE NUMBER AND WHAT IM UP TO IT WORKS WELL OVER THE NETWORK OFTEN ILL JUST FINGER SOMEONE BEFORE CALLING THEIR TELEPHONE

THE VIRUS INVADED THROUGH THE PROGRAM THAT HANDLED FINGER REQUESTS THE FINGER DAEMON HAS ROOM FOR CHARACTERS OF DATA THE VIRUS SENT CHARACTERS WHAT HAPPENED TO THE EXTRA CHARACTERS THEY GOT EXECUTED AS COMMANDS TO UNIX

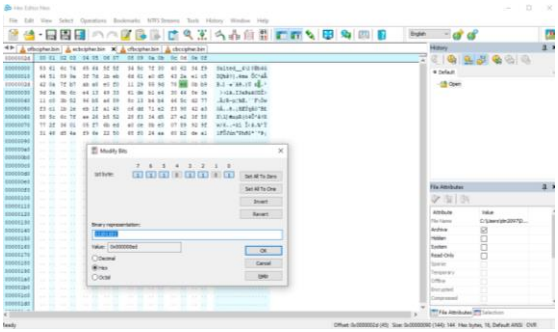
BY OVERFLOWING THE FINGER DAEMON THE VIRUS FOUND A SECOND WAY TO EXECUTE THE

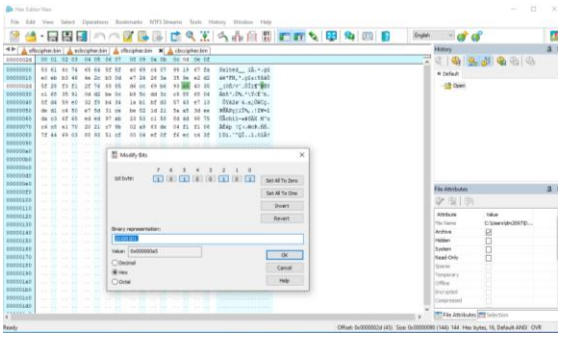
COMMAND EXECUTE THE FOLLOWING PROGRAM ON SOMEONE ELSES COMPUTER

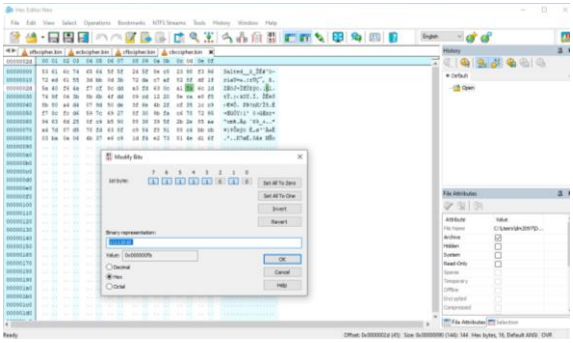
IF THAT WASNT ENOU

TASK3:

Changing of 1st bit of 46th byte using hex editor:







References:[i]

https://www.highgo.ca/2019/08/08/the-difference-in-five-modes-in-the-aes-encryption-algorithm/

https://www.geeksforgeeks.org/cryptography-introduction/

---

i