DATTA MEGHE COLLEGE OF ENGINEERING,
Airoli, Navi Mumbai

Department of Information Technology

Academic Year: 2024-25 (Term I)

LAB MANUAL

(ACADEMIC RECORD)

NAME OF THE SUBJECT: Security Lab (R-19)

CLASS:TE

SEMESTER:V

	DATTA MEGHE COLLEGE OF ENGINEERING, AIROLI, NAVI MUMBAI
	DEPARTMENT OF INFORMATION TECHNOLOGY

Institute Vision	:	**To create value - based technocrats to fit in the world of work and research**
Institute Mission	:	● **To adopt the best engineering practices** ● **To empower students to work in the world of technology and research** ● **To create competent human beings**
Department Vision	:	**To develop and foster students for successful careers in the dynamic field of Information Technology.**

Department Mission:

M1 :	**To create and disseminate knowledge through research, teaching & learning and to enhance society in meaningful and sustainable ways.**
M2:	**To impart suitable environment for students and staff to showcase innovative ideas in the field of IT.**
M3:	**To bridge the curriculum gap by facilitating effective interaction among industry and Staff/Students.**

Program Educational Objectives (PEO)

PEO1 - Develop proficiency as IT technocrat with an ability to solve a wide range of computational problems in industry, government, or other work environments.

PEO2 - Attain the ability to adapt quickly to new environments and technologies, assimilate new information, and work in multi-disciplinary areas with a strong focus on innovation and entrepreneurship.

PEO3 - Prepare graduates with the ability of life-long learning to innovate in ever-changing global economic and technological environments of the current era.

PEO4 - Possess the ability to function ethically and responsibly with good cultural values and integrity to apply the best principles and practices of Information Technology towards society.

Program Specific Outcomes (PSO)

PSO1 - Apply Core Information Technology knowledge to develop stable and secure IT system

PSO2 - Design, IT infrastructures for an enterprise using concepts of best practices in information

PSO3 - Ability to work in multidisciplinary IT enabled projects for industry and society by adapting latest trends and technologies like Analytics, Blockchain, Cloud, Data science.

DattaMeghe College of Engineering, Airoli

Department of Information Technology

Course Name: Security Lab (R-19)

Course Code: ITC502

Year of Study: 2024-25

T.E., Semester: V

Course Outcomes

ITC502.1	Illustrate symmetric cryptography by implementing classical ciphers
ITC502.2	Demonstrate Key management, distribution and user authentication.
ITC502.3	Explore the different network reconnaissance tools to gather information about networks
ITC502.4	Use tools like sniffers, port scanners and other related tools for analyzing packets in a network
ITC502.5	Use open-source tools to scan the network for vulnerabilities and simulate attacks.
ITC502.6	Demonstrate the network security system using open source tools.

DattaMeghe College of Engineering Airoli, Navi Mumbai	**DEPARTMENT OF INFORMATION TECHNOLOGY** **ACADEMIC YEAR : 2024 – 25 (TERM – I)** **List of Experiments** **Course Name : Security Lab** **Course Code : ITL502**

Sr. No.	Name of the Experiment	LO Covered
1	Breaking the Mono-alphabetic Substitution Cipher using Frequency analysis method.	LO1
2	Design and Implement a product cipher using Substitution ciphers.	LO1
3	Cryptanalysis or decoding Playfair, vigenere cipher	LO1
4	Encrypt long messages using various modes of operation using AES or DES.	LO2
5	Cryptographic Hash Functions and Applications (HMAC): to understand the need, design and applications of collision resistant hash functions.	LO2
6	Implementation and analysis of RSA cryptosystem and Digital signature scheme using RSA.	LO2
7	Study the use of network reconnaissance tools like WHOIS, dig, traceroute, nslookup to gather information about networks and domain registrars.	LO3
8	Study of packet sniffer tools wireshark: - a. Observer performance in promiscuous as well as non-promiscuous mode. b. Show the packets can be traced based on different filters.	LO3
9	Download, install nmap and use it with different options to scan open ports, perform OS fingerprinting, ping scan, tcp port scan, udp port scan, etc.	LO4
10	Study of malicious software using different tools: a) Keylogger attack using a keylogger tool. b) Simulate DOS attack using Hping or other tools c) Use the NESSUS/ISO Kali Linux tool to scan the network for vulnerabilities.	LO5
11	Study of Network security by a) Set up IPSec under Linux. b) Set up Snort and study the logs. c) Explore the GPG tool to implement email security	LO6

EXPERIMENT NUMBER: 1

Date of Performance :	
Date of Submission :	

Aim: Breaking the Mono-alphabetic Substitution Cipher using Frequency analysis method.

Theory:

Breaking the Mono-alphabetic Substitution Cipher

Consider we have the plain text "cryptography". By using the substitution table below, we can encrypt our plain text as follows: abcdefgh i j k l mnopqr s t u vwxyz

JI BRKTCNOFQYG AUZHSVWMXL DEP

plain text: c r y p t o g r a p h y

cipher text: B S E Z W U C S J Z N E

Hence we obtain the cipher text as "BSEZWUCSJZNE".

Cryptanalysis

Note that the frequency of occurrence of characters in the plaintext is "preserved" in the ciphertext. For instance, the most frequent character in the ciphertext is likely to be the encryption of the plaintext character "e" which is the most frequently occurring charecter in English. For a very brief theory of the mono-alphabetic substitution cipher and its cryptanalysis.

Procedure:

STEP 1 : For the given ciphertext in the **PART I** of the experiment page, the first step is to generate ciphertext by clicking on the "Next CipherText" button.

STEP 2 : Calculate frequencies of generated ciphertext by clicking on "Calculate Frequencies in Ciphertext" button

STEP 3 : Copy the generated ciphertext from **PART I** and paste in "Scratchpad" area of **PART II**

STEP 4 : Analyse similarties between "Calculated Frequencies Table" and "English Alphabet Frequencies Table"

STEP 5 : Based on similarities,try to make a frequency based estimation for each character of ciphertext

STEP 6 : Replace characters of CipherText in Scratchpad with a character estimated previously using a **Modify** function of **PART II**

STEP 7 : Based on Hints from Ciphertext in "Scratchpad" area make more replacement of ciphertext characters

STEP 8 : Repeat **Step 7** till you get a meaningful English Text

STEP 9 : Finally, observe the deciphered plaintext in Scratchpad Area,if a meaningful English text is formed cut-and-paste it in the text-field named "Solution Plaintext" of **PART III**. Also enter the final character mapping in the"Solution Key" in **PART III** and click on "Check Answer" button.

STEP 10[OPTIONAL] : Verify that your answer is correct, by encrypting the solution plaintext with your key in **PART IV**.

PART I

Decrypt the following cipher text. A tool to simulate the Mono-Alphabetic Subsitution cipher is provided beneath for your assistance.

Here is the table of frequencies of English alphabets for your reference:

a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**
8.167	1.49	2.782	4.253	12.702	2.228	2.015	6.094	6.966	0.153	0.772	4.025	2.406
n	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**
6.749	7.507	1.929	0.095	5.987	6.327	9.056	2.758	0.978	2.360	0.150	1.974	0.074

Ciphertext Frequencies:

PART II

Note that the *cipher text is in lower case* and when you replace any character, the final character of replacement, i.e., *plaintext is changed to upper case* automatically in the following scratchpad.

Scratchpad:

Modify the text above (in scratchpad):

This is case *insensitive* function and replaces only cipher text (lower case) by plain text (upper case):

Replace cipher character  by plaintext character 

Use the following function to undo any unwanted exchange by giving an uppercase character and a lower case. This is a case sensitive function:

Replace character  by character 

Your replacement history:

PART III

Enter your solution plaintext here:

Solution Key =

PART IV

Plaintext

key = phqgiumeaylnofdxjkrcvstzv

☐ Remove Punctuation

Ciphertext

CONCLUSION/ Outcome:

we successfully implemented breaking the mono-alphabetic Substitution cipher using frequency analysis method.

Marks & Signature:

R1 (5 Marks)	R2 (5 Marks)	R3 (5 Marks)	Total (15 Marks)	Signature

EXPERIMENT NUMBER: 2

Date of Performance :	
Date of Submission :	

AIM: Design and Implement a product cipher using Substitution ciphers.

THEORY:

- The Caesar cipher is a simple encryption technique that was used by Julius Caesar to send secret messages to his allies. It works by shifting the letters in the plaintext message by a certain number of positions, known as the "shift" or "key".
- The Caesar Cipher technique is one of the earliest and simplest methods of encryption technique. It's simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter with a fixed number of positions down the alphabet. For example with a shift of 1, A would be replaced by B, B would become C, and so on. The method is apparently named after Julius Caesar, who apparently used it to communicate with his officials.
- Thus to cipher a given text we need an integer value, known as a shift which indicates the number of positions each letter of the text has been moved down.
  The encryption can be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme, A = 0, B = 1,…, Z = 25. Encryption of a letter by a shift n can be described mathematically as.
- For example, if the shift is 3, then the letter A would be replaced by the letter D, B would become E, C would become F, and so on. The alphabet is wrapped around so that after Z, it starts back at A.
- Here is an example of how to use the Caesar cipher to encrypt the message "HELLO" with a shift of 3:

1. Write down the plaintext message: HELLO
2. Choose a shift value. In this case, we will use a shift of 3.
3. Replace each letter in the plaintext message with the letter that is three positions to the right in the alphabet.
   H becomes K (shift 3 from H)

   E becomes H (shift 3 from E)

   L becomes O (shift 3 from L)

   L becomes O (shift 3 from L)

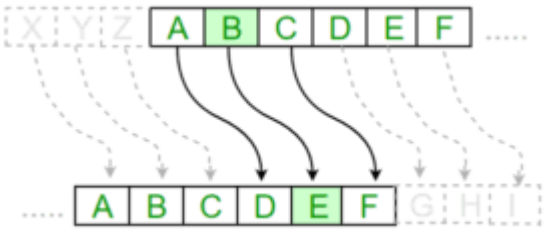
Algorithm of Caesar Cipher

The algorithm of Caesar cipher holds the following features −

- Caesar Cipher Technique is the simple and easy method of encryption technique.
- It is simple type of substitution cipher.
- Each letter of plain text is replaced by a letter with some fixed number of positions down with alphabet.

.

EXAMPLE:



Examples:

Plain Text: I am studying Data Encryption
Key: 4
Output: M eqwxyhCmrkHexeIrgvCtxmsr

Plain Text: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Key: 4
Output:EFGHIJKLMNOPQRSTUVWXYZabcd

Caesar Cipher

The program implementation of Caesar cipher algorithm is as follows −

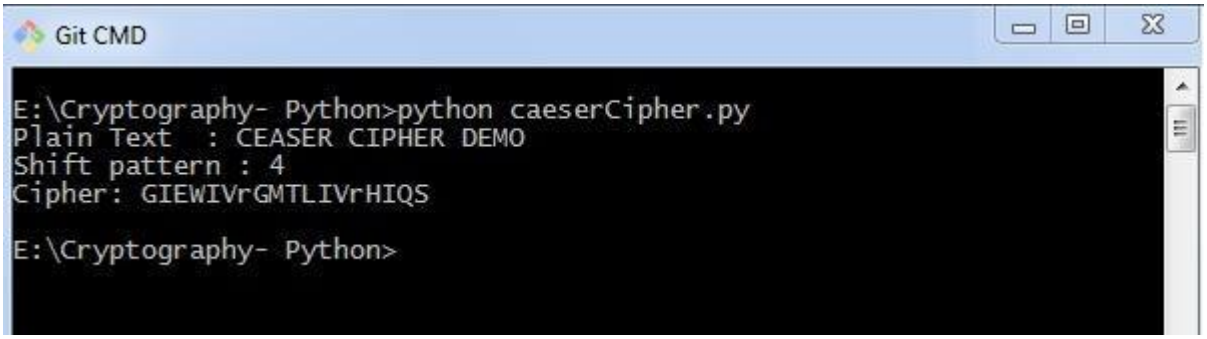
```
def encrypt(text,s):
result =""
# transverse the plain text
for i in range(len(text)):
char= text[i]
# Encrypt uppercase characters in plain text

if(char.isupper()):
result+=chr((ord(char)+ s-65)%26+65)
# Encrypt lowercase characters in plain text
else:
result+=chr((ord(char)+ s -97)%26+97)
return result
#check the above function
text="CEASER CIPHER DEMO"
s =4

print"Plain Text : "+ text
print"Shift pattern : "+str(s)
print"Cipher: "+ encrypt(text,s)
```

Output

You can see the Caesar cipher, that is the output as shown in the following image −



CONCLUSION/ Outcome:

we successfully product cipher using Substitution ciphers.

Marks & Signature:

R1 (5 Marks)	R2 (5 Marks)	R3 (5 Marks)	Total (15 Marks)	Signature

EXPERIMENT NUMBER: 3

Date of Performance :	
Date of Submission :	

AIM: Cryptanalysis or decoding Playfair, vigenere cipher.

THEORY:

The Playfair Cipher encryption technique can be used to encrypt or encode a message. It operates exactly like typical encryption. The only difference is that it encrypts a digraph, or a pair of two letters, instead of a single letter.

An initial 5×5 matrix key table is created. The plaintext encryption key is made out of the matrix's alphabetic characters. Be mindful that you shouldn't repeat the letters. There are 26 alphabets however, there are only 25 spaces in which we can place a letter. The matrix will delete the extra

letter because there is an excess of one letter (typically J). Despite this, J is there in the plaintext before being changed to I.

This blog provides a full explanation of the Playfair Cipher, its advantages and disadvantages, its applicability, and the Playfair encryption and decryption algorithms.
The Algorithm consists of 2 steps:

1. **Generate the key Square(5×5):**
   - The key square is a 5×5 grid of alphabets that acts as the key for encrypting the plaintext. Each of the 25 alphabets must be unique and one letter of the alphabet (usually J) is omitted from the table (as the table can hold only 25 alphabets). If the plaintext contains J, then it is replaced by I.

   - The initial alphabets in the key square are the unique alphabets of the key in the order in which they appear followed by the remaining letters of the alphabet in order.

2. **Algorithm to encrypt the plain text:** The plaintext is split into pairs of two letters (digraphs). If there is an odd number of letters, a Z is added to the last letter.
   For example:

PlainText: "instruments"
After Split: 'in' 'st' 'ru' 'me' 'nt' 'sz'
1. Pair cannot be made with same letter. Break the letter in single and add a bogus letter to the previous letter.
Plain Text: "hello"
After Split: 'he' 'lx' 'lo'
Here **'x'** is the bogus letter.
2. If the letter is standing alone in the process of pairing, then add an extra bogus letter with the alone letter
Plain Text: "helloe"
AfterSplit: 'he' 'lx' 'lo' 'ez'
Here **'z'** is the bogus letter.
Rules for Encryption:

If both the letters are in the same column: Take the letter below each one (going back to the top if at the bottom).
For example:

Diagraph: "me"
Encrypted Text: cl
Encryption:
m -> c
e -> l

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

If both the letters are in the same row: Take the letter to the right of each one (going back to the leftmost if at the rightmost position).
For example:

Diagraph: "st"
Encrypted Text: tl
Encryption:
s -> t
t -> l

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

If neither of the above rules is true: Form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.
For example:

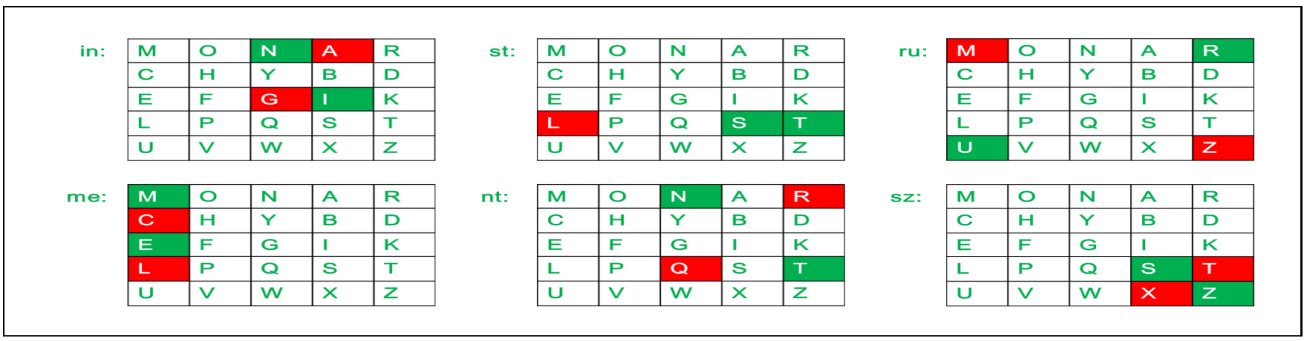
Diagraph: "nt"
Encrypted Text: rq
Encryption:
n -> r
t -> q

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

For example:

Plain Text: "instrumentsz"
Encrypted Text: gatlmzclrqtx
Encryption:
i -> g
n -> a
s -> t
t -> l
r -> m
u -> z
m -> c
e -> l

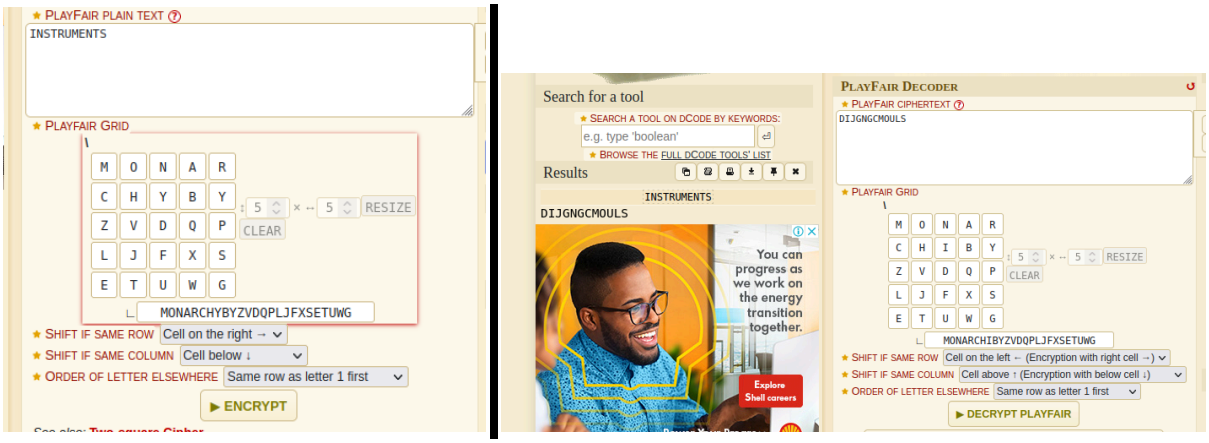
n -> r
t -> q
s -> t
z -> x



Below is an implementation of Playfair Cipher using tool

https://www.dcode.fr/playfair-cipher#f1
https://www.dcode.fr/vigenere-cipher

Encryption process:-



Decryption process

Vigenere cipher. :-

Vigenere Cipher is a method of encrypting alphabetic text. It uses a simple form of [polyalphabetic substitution](). A polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets. The encryption of the original text is done using the *Vigenère square or Vigenère table*.

The table consists of the alphabets written out 26 times in different rows, each alphabet shifted cyclically to the left compared to the previous alphabet, corresponding to the 26 possible Caesar Ciphers.

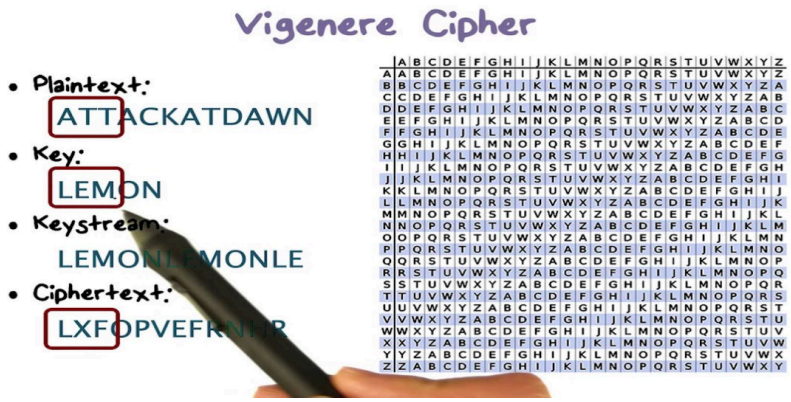
At different points in the encryption process, the cipher uses a different alphabet from one of the rows.

The alphabet used at each point depends on a repeating keyword.

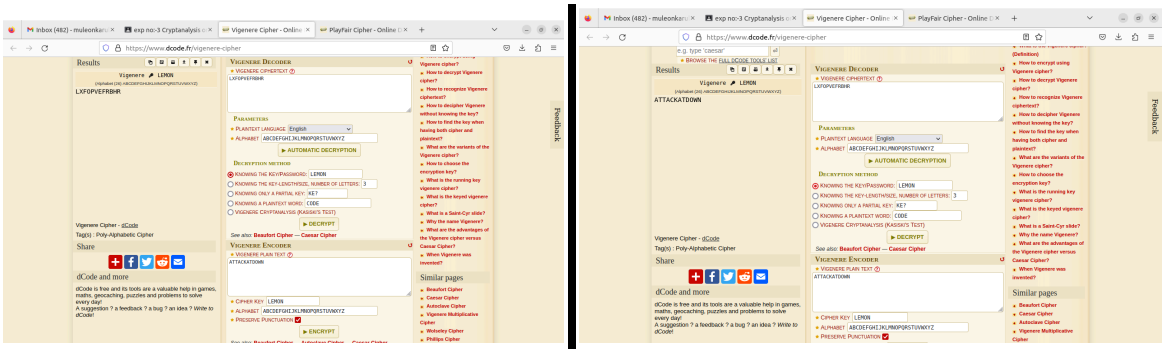
, you will only use as many keys as there are unique letters in the key string, here just 5 keys, {L, E, M, O, N}. For successive letters of the message,weareg oingtotakesuccessivelettersofthekeystring,andenciphereachmessage

letter using its corresponding key row. Choose the next letter of the key, going that row to find the column heading that m atches the message character; the letter at the intersection of [key-row, msg-col] is the enciphered letter.

EXAMPLE:



Output



CONCLUSION/ Outcome:

we successfully Playfair, vigenere cipher.

Marks & Signature:

R1 (5 Marks)	R2 (5 Marks)	R3 (5 Marks)	Total (15 Marks)	Signature

EXPERIMENT NUMBER: 4

Date of Performance :	
Date of Submission :	

AIM: Encrypt long messages using various modes of operation using AES or DES.

THEORY:

Theory – Encryption is a way of scrambling data so that only authorized parties can understand the information. In technical terms, it is the process of converting human-readable plaintext to incomprehensible text, also known as ciphertext. In simpler terms, encryption takes readable data and alters it so that it appears random. Encryption requires the use of a cryptographic key a set of mathematical values that both the sender and the recipient of an encrypted message agree on.

There are two types of encryption:

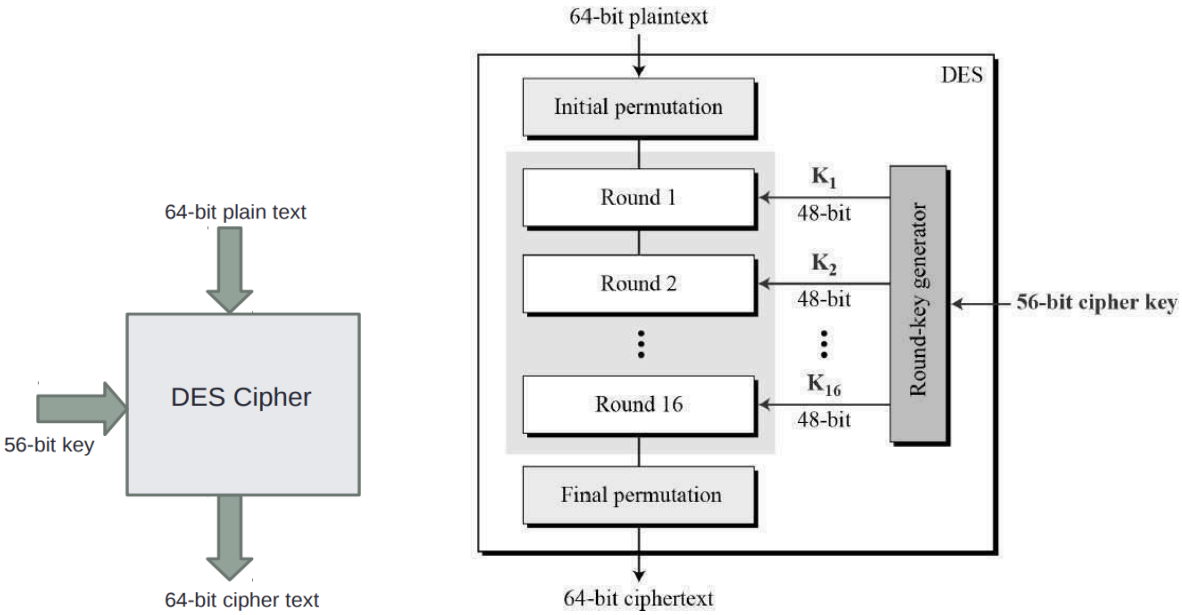
1. Symmetric Encryption.
2. Asymmetric Encryption

Symmetric Encryption: In symmetric encryption, there is only one key, and all parties involved use the same key to encrypt and decrypt information. By using a single key, the process is straightforward, as per the following example: you encrypt an email with a unique key, send that email to your friend Tom, and he will use the same symmetric- key to unlock/decrypt the email. The perks of symmetric encryption are its faster performance and low resource consumption, but it is inherently older and less secure than its counterpart. The reason is simple: if you scale your encryption to a company- wide scale, it means you're putting all your trust into a single key you will need to share around a lot. For this reason, Symmetric encryption is great when working with sensitive data in bulk.

Asymmetric Encryption: Asymmetric encryption, on the other hand, was created to solve the inherent issue of symmetric encryption: the need of sharing a single encryption key around that is used both for encrypting and decrypting data. This newer and safer method utilizes two keys for its encryption process, the public key, used for encryption, and the

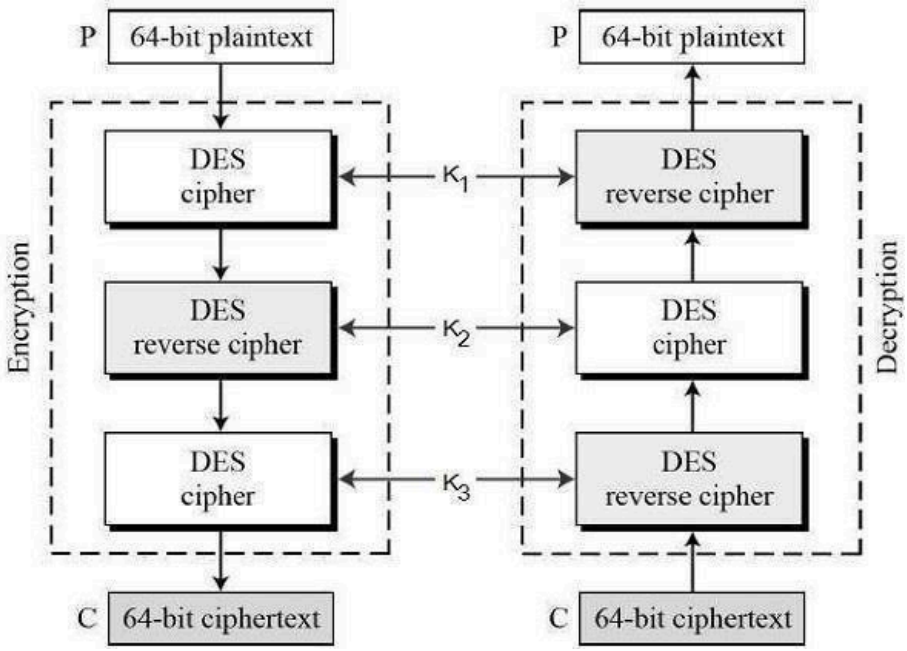
private key used for decryption. A public key is available for anyone who needs to encrypt a piece of information. This key doesn't work for the decryption process. A user needs to have a secondary key, the private key, to decrypt this information. This way, the private key is only held by the actor who decrypts the information, without sacrificing security as you scale security. A good example is email encryption.

Data encryption standard (DES) has been found vulnerable against very powerful attacks and therefore, the popularity of DES has been found slightly on decline. DES is a block cipher, and encrypts data in blocks of size of 64 bit each, means 64 bits of plain text goes as the input to DES, which produces 64 bits of cipher text. The same algorithm and key are used for encryption and decryption, with minor differences. The key length is 56 bits. The basic idea is show in figure.

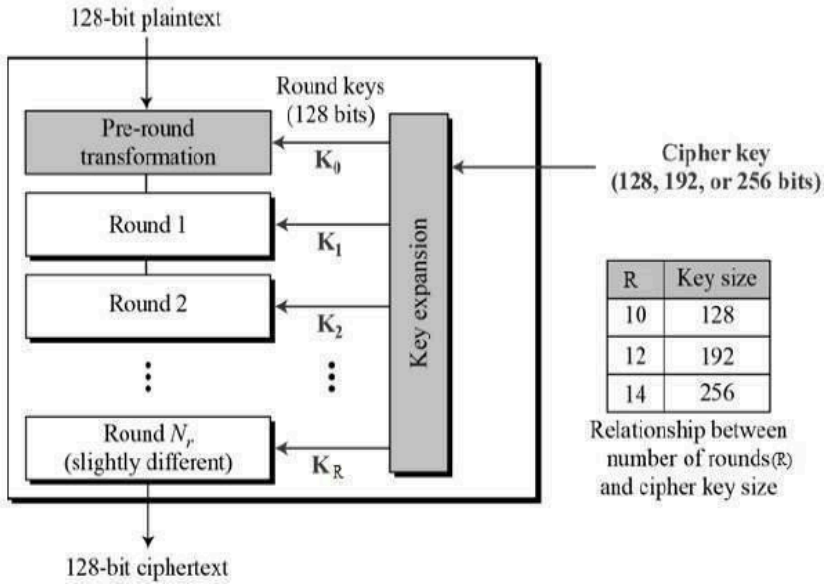


3- KEY Triple DES

Before using 3TDES, user first generate and distribute a 3TDES key K, which consists of three different DES keys K1, K2 and K3. This means that the actual 3TDES key has length 3×56 = 168 bits. The encryption scheme is illustrated as follows −



Advance Encryption Standard (AES) is an iterative rather than Feistel cipher. It is based on 'substitution–permutation network'. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).



R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

Modes of Operation

Mode 1 - Electronic Code Book(ECB) Mode

Mode 2 – Cipher Block Chaining(CBC) Mode Mode 3 – Output Feedback(OFB) Mode

Mode 4 – Counter(CTR) Mode

AES and Modes of Operation

Step I : Choose a mode of operation from **PART I**

Step II : Select KeySize, Plaintext, KeyText, Intialization vector(IV)(for ECB and OFB modes only) and CTR(forctr mode only) in **PART II**

Step III : Whenever necessay use XOR opeartion in **PART III** in accordance with choosen mode of operation

Step IV : Use fuction **FK** and "Key in hex:" field in **PART IV** should be filled keytext generated in **Step2**

Step V : Fill "Plaintext in hex:" field with approriate value in accordance with choosen mode of operation and click on encrypt button

Step VI : Enter your answer in **PART V** to check your ciphertext

From DES to 3-DES

PART I

Message 00010100 11010111 01001001 00010010 01111100 10011110 00011011 1000C Change plaintext

Key Part A 3b389837152of75e Change Key A
Key Part B 922fb510c71f436e Change Key B

PART II

Your text to be encrypted/decrypted: 10101011 10101110 01111110 01111111 01111000 10000100 10011100 1001011
Key to be used: 3b389837152of75e
DES Encrypt DES Decrypt

Output: 00011101 11100100 10001000 01101111 11010001 00011011 00110000 1100C

PART III

Enter your answer here:

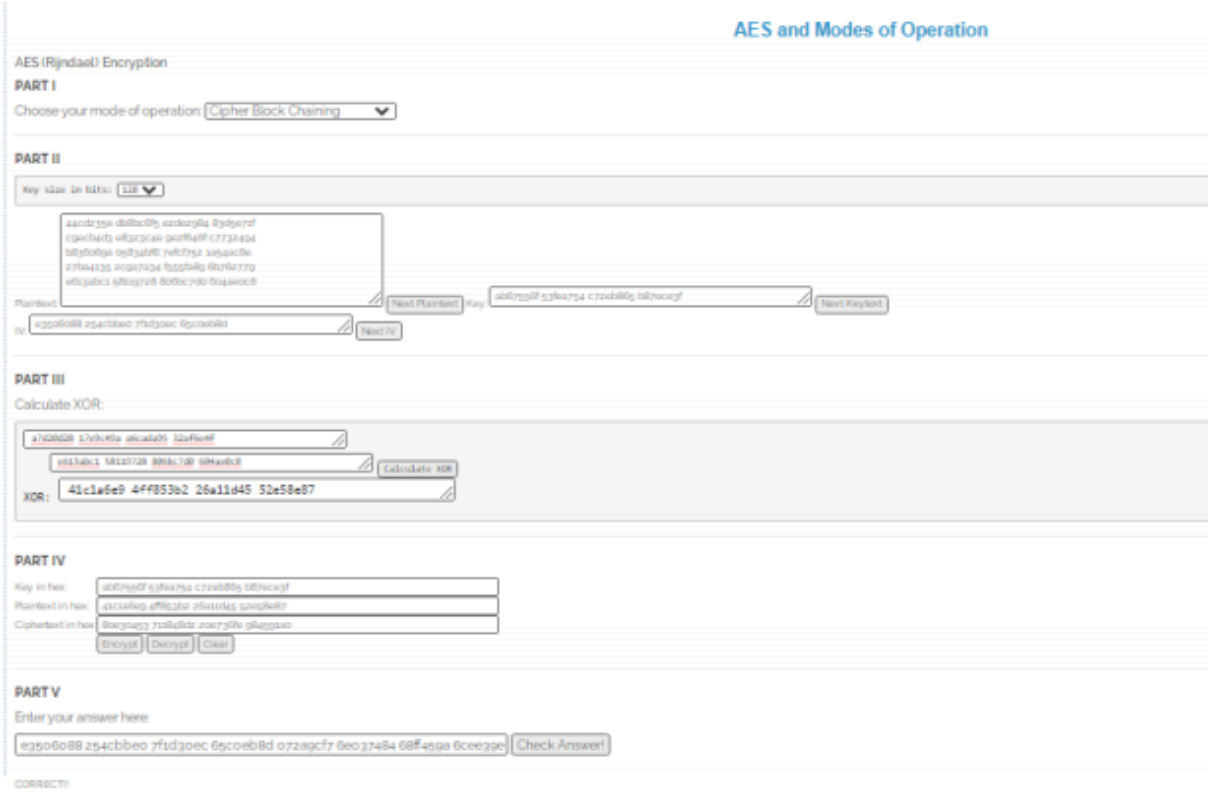
00011101 11100100 10001000 01101111 11010001 00011011 00110000 1100C

Check Answer!

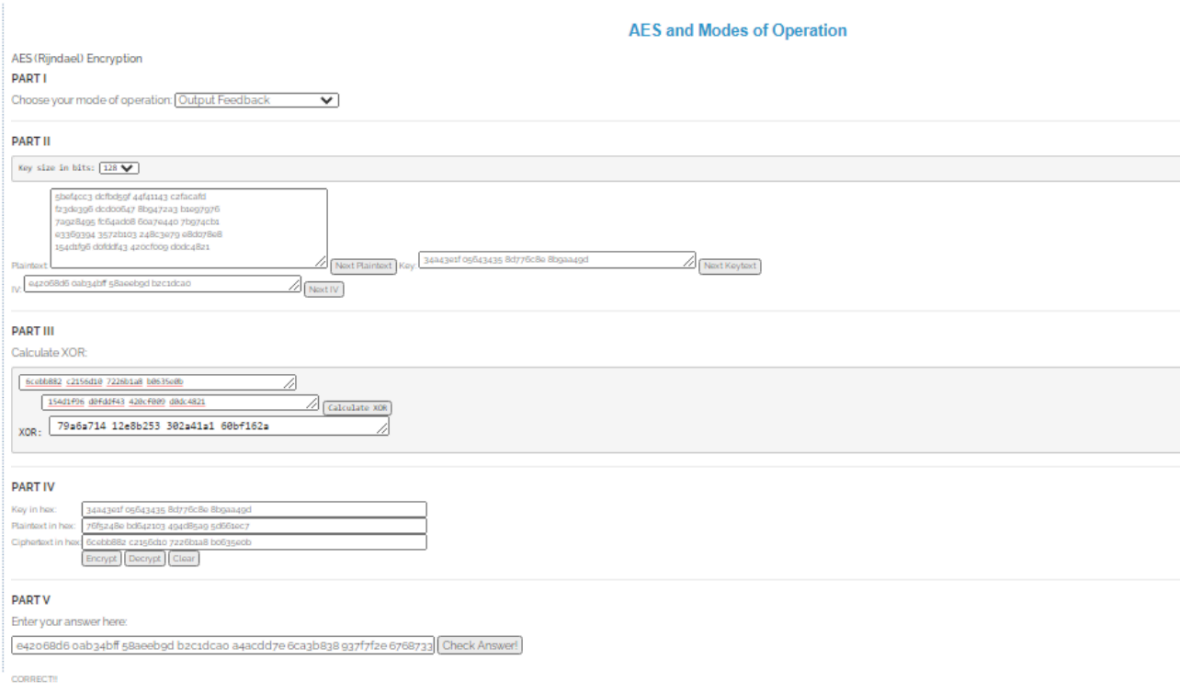
CORRECT!

OUTPUT :-

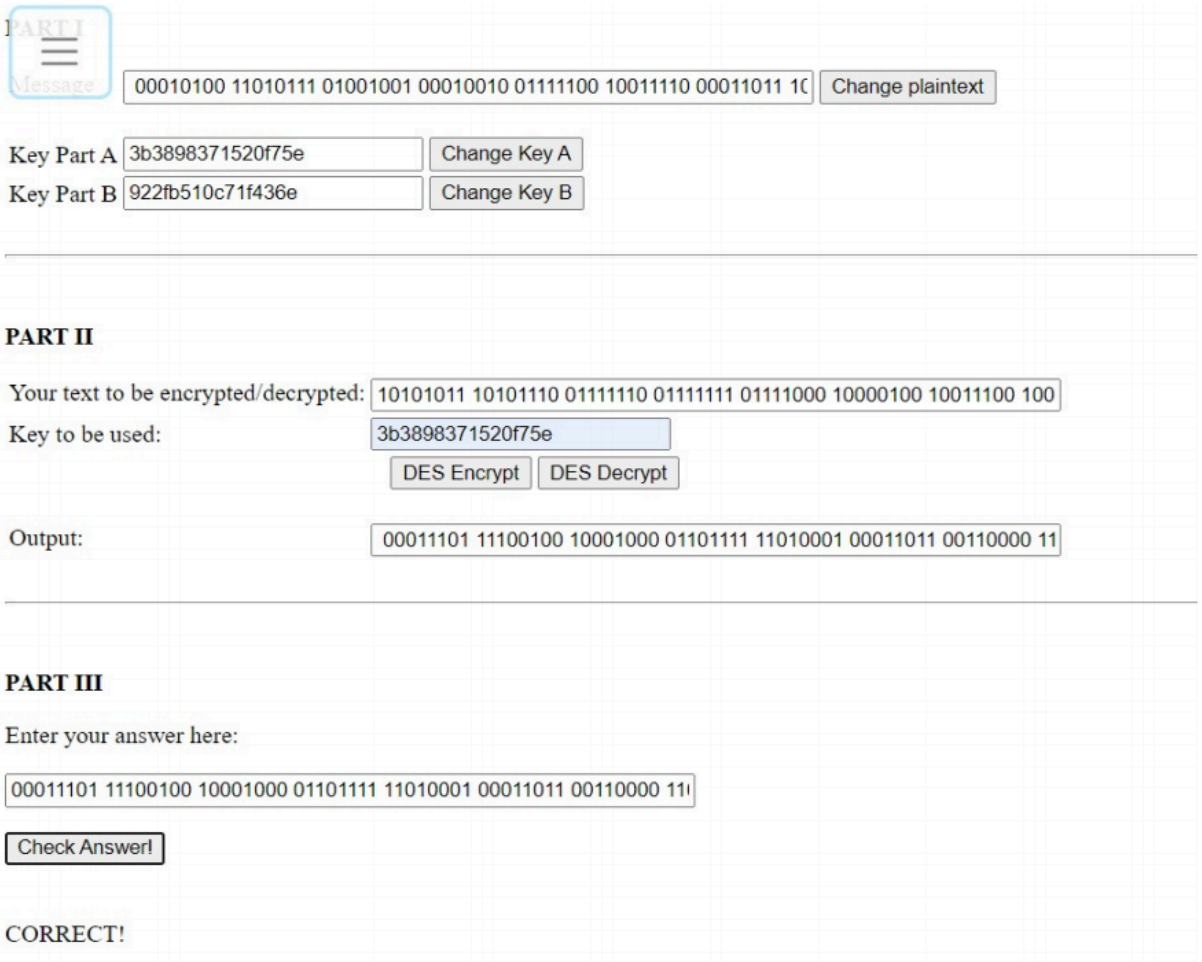
Cipher block chaining



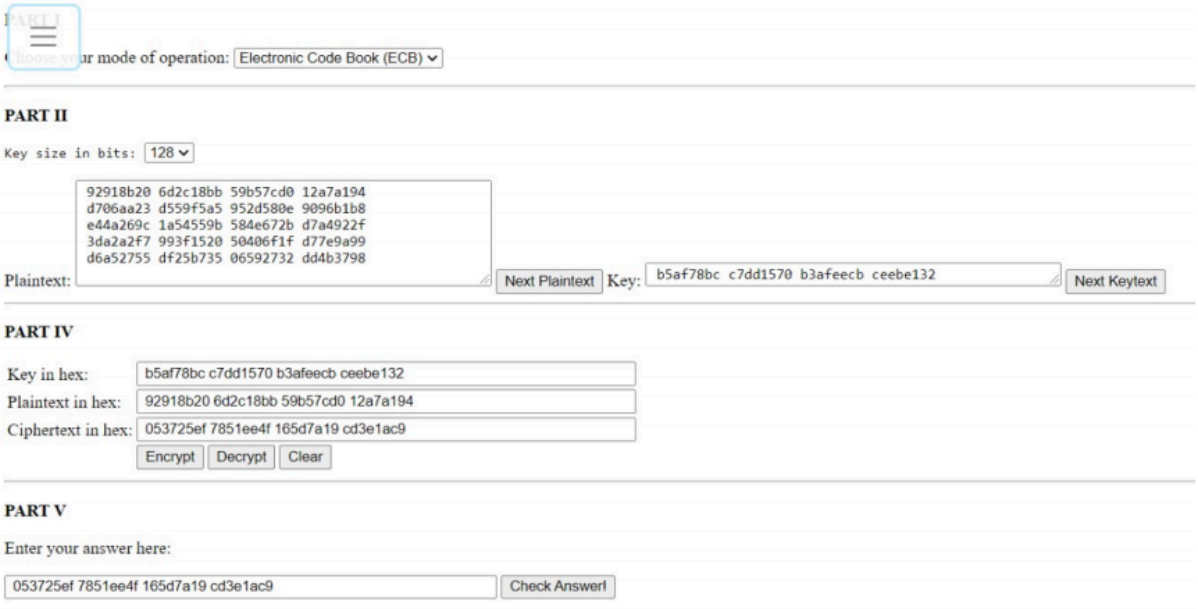
Output feedback


AES and Modes of Operation
AES (Rijndael) Encryption
PART I
Choose your mode of operation: Output Feedback
PART II
Key size in bits: 128
Plaintext
Next Plaintext
Key
Next Keytext
IV
Next IV
PART III
Calculate XOR:
Calculate XOR
XOR: 79a6a714 12e8b253 302a41a1 60bf162a
PART IV
Key in hex:
Plaintext in hex:
Ciphertext in hex:
Encrypt
Decrypt
Clear
PART V
Enter your answer here:
e42068d6 0ab34bff 58aeeb9d b2c1dcao a4acdd7e 6ca3b838 937f7f2e 6768733
Check Answer!
CORRECT!

Counter


PART I
Message
00010100 11010111 01001001 00010010 01111100 10011110 00011011 10
Change plaintext
Key Part A 3b3898371520f75e
Change Key A
Key Part B 922fb510c71f436e
Change Key B
PART II
Your text to be encrypted/decrypted: 10101011 10101110 01111110 01111111 01111000 10000100 10011100 100
Key to be used: 3b3898371520f75e
DES Encrypt
DES Decrypt
Output: 00011101 11100100 10001000 01101111 11010001 00011011 00110000 11
PART III
Enter your answer here:
00011101 11100100 10001000 01101111 11010001 00011011 00110000 11
Check Answer!
CORRECT!

Electronic Code Book



CONCLUSION/ Outcome:

we successfully Hence,long messages have been encrypted using various modes of operation using AES or DES

Marks & Signature:

R1 (5 Marks)	R2 (5 Marks)	R3 (5 Marks)	Total (15 Marks)	Signature

EXPERIMENT NUMBER: 5

Date of Performance :	
Date of Submission :	

AIM: Cryptographic Hash Functions and Applications (HMAC): to understand the need, design and applications of collision resistant hash functions

.

THEORY:

HMAC :

HMAC (Hash-based Message Authentication Code) is a type of a message authentication code (MAC) that is acquired by executing a cryptographic hash function on the data (that is) to be authenticated and a secret shared key.

Two parties want to communicate, but they want to ensure that the contents of their connection remain private. They also distrust the internet, and they need a way to verify that the packets they receive haven't been tampered with. HMAC is a valid solution.

HMAC keys consist of two parts. These are:

Cryptographic keys. An encryption algorithm alters data, and a recipient needs a specific code (or key) to make it readable once more. HMAC relies on two sets of keys. One is public, and one is private.

Hash function. A hash algorithm alters or digests the message once more. HMAC uses generic cryptographic hash functions, such as SHA-1, MD5, or RIPEMD- 128/60.

Like any of the MAC, it is used for both data integrity and authentication. Checking data integrity is necessary for the parties involved in communication.

HTTPS, SFTP, FTPS, and other transfer protocols use HMAC. The cryptographic hash function may be MD-5, SHA-1, or SHA-256. Digital signatures are nearly similar to HMACs i.e they both employ a hash function and a shared key.

The difference lies in the keys i.e HMACs use symmetric key(same copy) while Signatures use asymmetric (two different keys). If attackers tamper this data, it may affect the processes and business decisions. So while working online over the internet, care must be taken to ensure integrity or least know if the data is changed. That is when HMAC comes into use.

Applications

- Verification of e-mail address during activation or creation of an account.
- Authentication of form data that is sent to the client browser and then submitted back.
- HMACs can be used for Internet of things (IoT) due to less cost.
- Whenever there is a need to reset the password, a link that can be used once is sent without adding a server state.
- It can take a message of any length and convert it into a fixed-length message digest. That is even if you got a long message, the message digest will be small and thus permits maximizing bandwidth.

Working of HMAC

- HMACs provides client and server with a shared private key that is known only to them.
- The client makes a unique hash (HMAC) for every request. When the client requests the server, it hashes the requested data with a private key and sends it as a part of the request.
- Both the message and key are hashed in separate steps making it secure. When the server receives the request, it makes its own HMAC. Both the HMACS are compared and if both are equal, the client is considered legitimate.

PRACTICAL:

Plaintext = 11000000 00111100 101010

Divide the plaintext into k chunks of 8 bits. If the kth chunk's bits are less than 8, make it 8 by padding O's at the end.

M1 = 11000000

M2 = 00111100

M3 = Mk = 10101000

|| denotes concatenation

M = M1 || M2 || Mk

M = 110000000011110010101000

L = length of M = 24 in decimal = 00011000

A = Key XOR ipad = 10000101 XOR 01011100 = 11011001

B = Key XOR opad = 10000101 XOR 00110110 = 10110011

Z0 = IV||A= 1100110011011001

Z1 = Hashed value of Z0 || M1 = 0000011111000000

Z2 = Hashed value of Zi || M2= 0010000000111100

Z3 = Zk= Hashed value of Z2 || M3 = 0100000010101000

Z(k+1) = Hashed value of Zk || L = 1000111000011000 **Hashed value of Z(k+1)** = 10010110

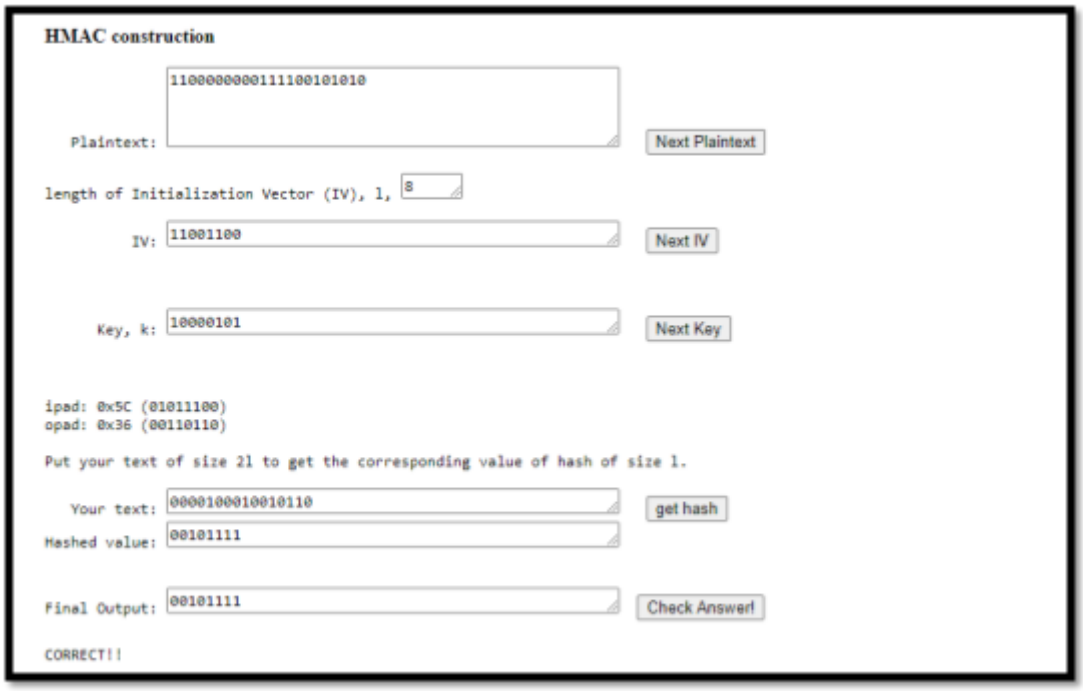
P = IV || B = 1100110010110011

Q = Hashed value of P = 00001000

R = Q || Hashed value of Z(k+1) = 0000100010010110

T = Hashed value of R = Final Output = 00101111

HMAC Construction



CONCLUSION/ Outcome:

Hence, we successfully designed HMAC construction by understanding the need, design and applications of collision resistant hash functions.

Marks & Signature:

R1 (5 Marks)	R2 (5 Marks)	R3 (5 Marks)	Total (15 Marks)	Signature

EXPERIMENT NUMBER: 6

Date of Performance :	
Date of Submission :	

AIM: Implementation and analysis of RSA cryptosystem and Digital signature scheme using RSA.

THEORY:-

The RSA algorithm is a public-key signature algorithm developed by Ron Rivest, Adi Shamir, and Leonard Adleman. Their paper was first published in 1977, and the algorithm uses logarithmic functions to keep the working complex enough to withstand brute force and streamlined enough to be fast post-deployment. The image below shows it verifies the digital signatures using RSA methodology.

RSA can also encrypt and decrypt general information to securely exchange data along with handling digital signature verification.

Steps in RSA Algorithm

Keeping the image above in mind, go ahead and see how the entire process works, starting from creating the key pair, to encrypting and decrypting the information.

Key Generation

You need to generate public and private keys before running the functions to generate your ciphertext and plaintext. They use certain variables and parameters, all of which are explained below:

- Choose two large prime numbers (p and q)
- Calculate n = p*q and z = (p-1)(q-1)
- Choose a number e where 1 < e < z
- Calculate d = e-1mod(p-1)(q-1)
- You can bundle private key pair as (n,d)
- You can bundle public key pair as (n,e)

Encryption/Decryption Function

Once you generate the keys, you pass the parameters to the functions that calculate your ciphertext and plaintext using the respective key.

- If the plaintext is m, ciphertext = me mod n.
- If the ciphertext is c, plaintext = cd mod n

To understand the above steps better, you can take an example where p = 17 and q=13. Value of e can be 5 as it satisfies the condition 1 < e < (p-1)(q-1).

N = p * q = 221

D = e-1mod(p-1)(q-1) = 29

Public Key pair = (221,5)

Private Key pair = (221,29)

If the plaintext(m) value is 10, you can encrypt it using the formula me mod n = 82.

To decrypt this ciphertext(c) back to original data, you must use the formula cd mod n = 29.

Implementation and Analysis of RSA Cryptosystem

Code :

```
import math


# step 1

p = 3

q = 7


# step 2

n = p*q
```

```
print("n =", n)


# step 3

phi = (p-1)*(q-1)


# step 4

e = 2

while(e<phi):

    if (math.gcd(e, phi) == 1):

        break

    else:

        e += 1


print("e =", e)

# step 5

k = 2

d = ((k*phi)+1)/e

print("d =", d)

print(f'Public key: {e, n}')

print(f'Private key: {d, n}')


# plain text

msg = 11
```

```
print(f'Original message:{msg}')



# encryption

C = pow(msg, e)

C = math.fmod(C, n)

print(f'Encrypted message: {C}')



# decryption

M = pow(C, d)

M = math.fmod(M, n)



print(f'Decrypted message: {M}')
```

output
n = 21
e = 5
d = 5.0
Public key: (5, 21)
Private key: (5.0, 21)
Original message:11
Encrypted message: 2.0
Decrypted message: 11.0

Step 1 : Enter the input text to be encrypted in the 'Plaintext' area and generate hash value for message by clicking on the **SHA-1** button

Step 2 : Copy content of Hash **Output(hex)** field and paste it in **Input to RSA(hex)** field.

Step 3 : Select keysize of public key from **RSA Public key** section by clicking on any key button.

Step 4 : Click on **Apply RSA** button to generate a digital signature.

Digital Signature Scheme using RSA

Digitally sign the plaintext with Hashed RSA.

Plaintext (string):

secret message | SHA-1

Hash output(hex):

4dea808c5b4e74af6f70fa10cec96d5f98e143e8

Input to RSA(hex):

4dea808c5b4e74af6f70fa10cec96d5f98e143e8 | Apply RSA

Digital Signature(hex):

```
078107f3739c9e66caf39649f4b7baf99f3bc75fd7c838e122e1e6e45dfcf234
dcc1a3caf8a6a6afb9c7f9e9e1b61727f1c25f264b60cc09970347573d2f8ddd
56f69d94a847e0585f7b6f0045ef426de045c3f3c6b38c1f216cf1488164e211
f769e817e186c1788c9d34c50b8656282422a0e80d3ead7bc47c0981b89b14ab
```

Digital Signature(base64):

```
B4EH83OcnmbK85ZJ9Le6+Z87x1/XyDjhIuHm5F388jTcwaPK+Kamr7nH+enhthcn
8cJfJktgzAmXA0dXPS+N3Vb2nZSoR+BYX3tvAEXvQm3gRcPzxrOMHyFs8UiBZOIR
92noF+GGwXiMnTTFC4ZWKCQioOgNPq17xHwJgbibFKs=
```

Status:

Time: 1ms

RSA public key

Public exponent (hex, F4=0x10001):

10001

Modulus (hex):

```
a5261939975948bb7a58dffe5ff54e65f0498f9175f5a09288810b8975871e99
af3b5dd94057b0fc07535f5f97444504fa35169d461d0d30cf0192e307727c06
5168c788771c561a9400fb49175e9e6aa4e23fe11af69e9412dd23b0cb6684c4
c2429bce139e848ab26d0829073351f4acd36074eafd036a5eb83359d2a698d3
```

1024 bit | 1024 bit (e=3) | 512 bit | 512 bit (e=3)

Digitally sign the plaintext with Hashed RSA.
Plaintext (string):
secret message SHA-1
Hash output(hex):
4dea808c5b4e74af6f70fa10cec96d5f98e143e8
Input to RSA(hex):
4dea808c5b4e74af6f70fa10cec96d5f98e143e8 Apply RSA
Digital Signature(hex):
078107f3739c9e66caf39649f4b7baf99f3bc75fd7c838e122e1e6e45dfcf234
dcc1a3caf8a6a6afb9c7f9e9e1b61727f1c25f264b60cc09970347573d2f8ddd
56f69d94a847e0585f7b6f0045ef426de045c3f3c6b38c1f216cf1488164e211
f769e817e186c1788c9d34c50b8656282422a0e80d3ead7bc47c0981b89b14ab
Digital Signature(base64):
B4EH83OcnmbK85ZJ9Le6+Z87x1/XyDjhIuHm5F388jTcwaPK+Kamr7nH+enhthcn
8cJfJktgzAmXA0dXPS+N3Vb2nZSoR+BYX3tvAEXvQm3gRcPzxrOMHyFs8UiBZOIR
92noF+GGwXiMnTTFC4ZWKCQioOgNPq17xHwJgbibFKs=
Status:
Time: 1ms
RSA public key
Public exponent (hex, F4=0x10001):
10001
Modulus (hex):
a5261939975948bb7a58dffe5ff54e65f0498f9175f5a09288810b8975871e99
af3b5dd94057b0fc07535f5f97444504fa35169d461d0d30cf0192e307727c06
5168c788771c561a9400fb49175e9e6aa4e23fe11af69e9412dd23b0cb6684c4
c2429bce139e848ab26d0829073351f4acd36074eafd036a5eb83359d2a698d3
1024 bit 1024 bit (e=3) 512 bit 512 bit (e=3)

CONCLUSION : Hence, we implemented and analysed RSA cryptosystem and digital signature scheme using RSA.

Marks & Signature:

R1 (5 Marks)	R2 (5 Marks)	R3 (5 Marks)	Total (15 Marks)	Signature

EXPERIMENT NUMBER: 7

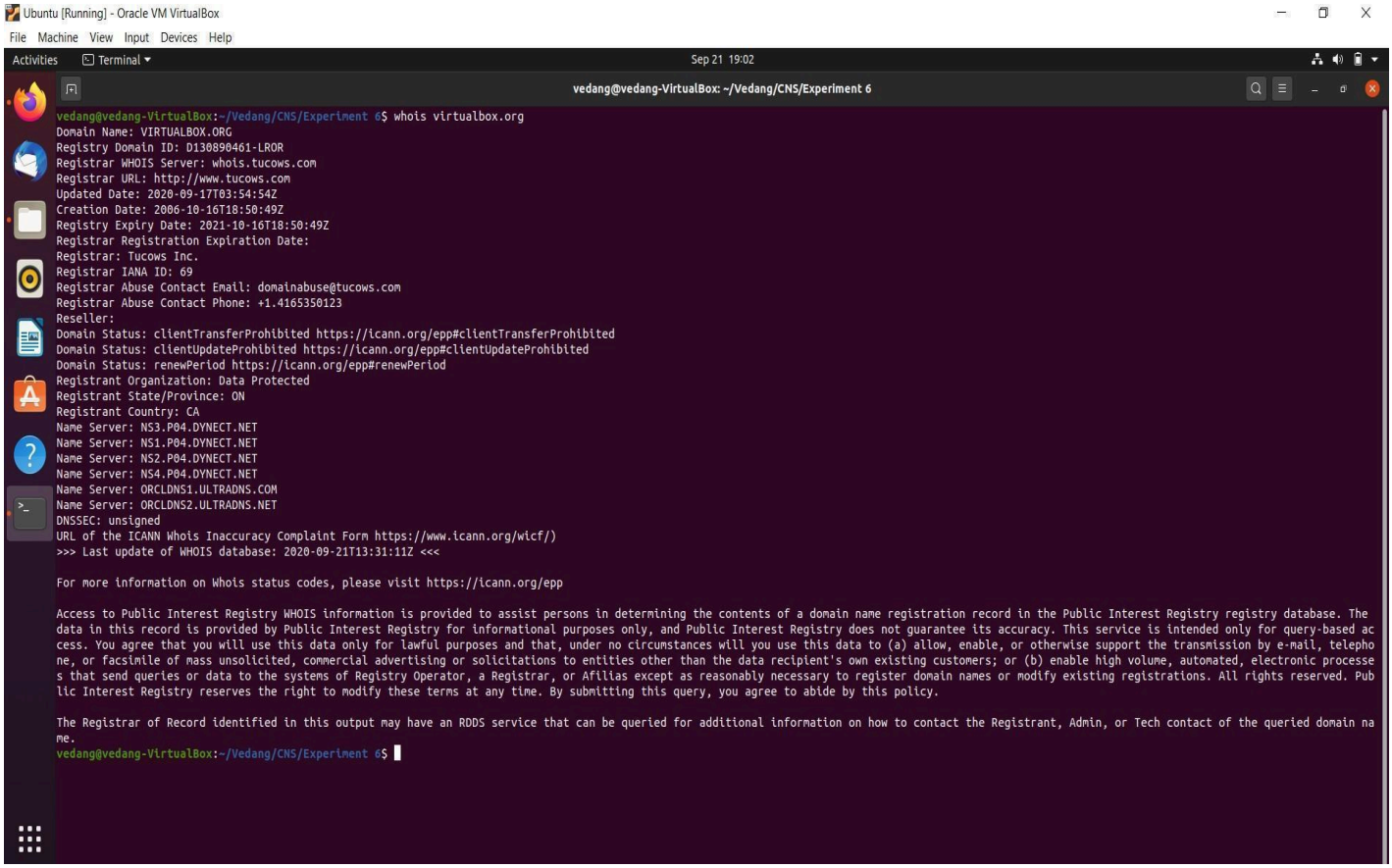
Date of Performance :	
Date of Submission :	

AIM: Study the use of network reconnaissance tools like WHOIS, dig, traceroute, nslookup to gather information about networks and domain registrars.

THEORY:

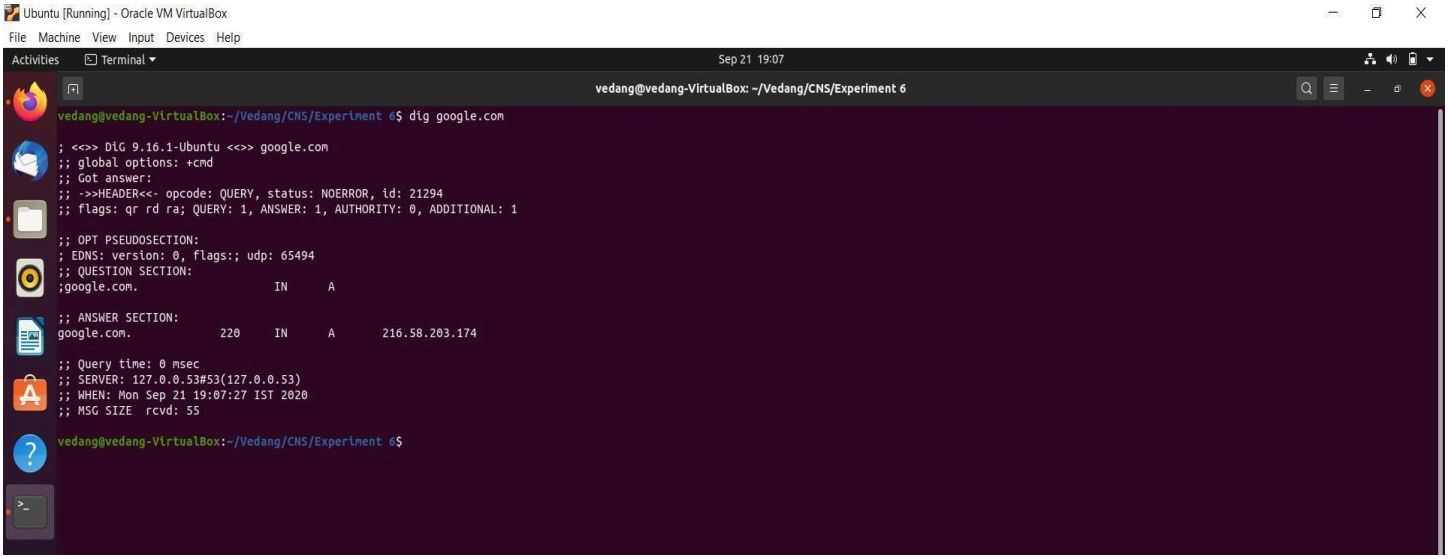
1) Whois :-

USE:-The whois command looks up the registration record associated with a domain name. This can show you more information about who registered and owns a domain name, including their contact information. whois searches for an object in a WHOIS database.
WHOIS is a query and response protocol that is widely used for querying databases that store the registered users of an Internet resource, such as a domain name or an IP address block, but is also used for a wider range of other information.



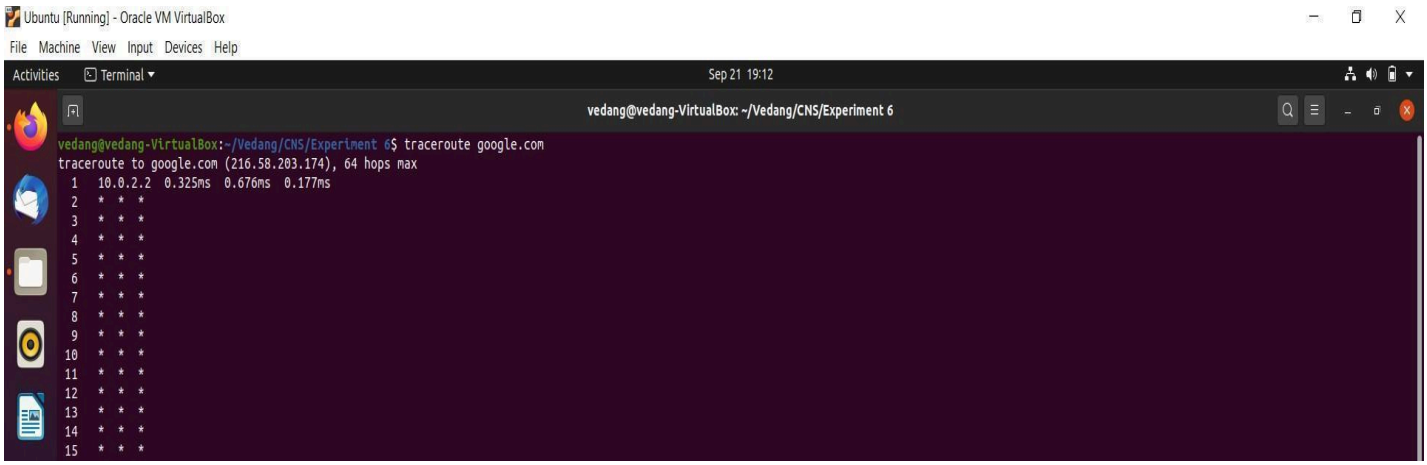
2) Dig :-

USE:-Dig is a networking tool that can query DNS servers for information. It can be very helpful for diagnosing problems with domain pointing and is a good way to verify that your configuration is working. The most basic way to use dig is to specify the domain we wish to query: dig example.com**.**



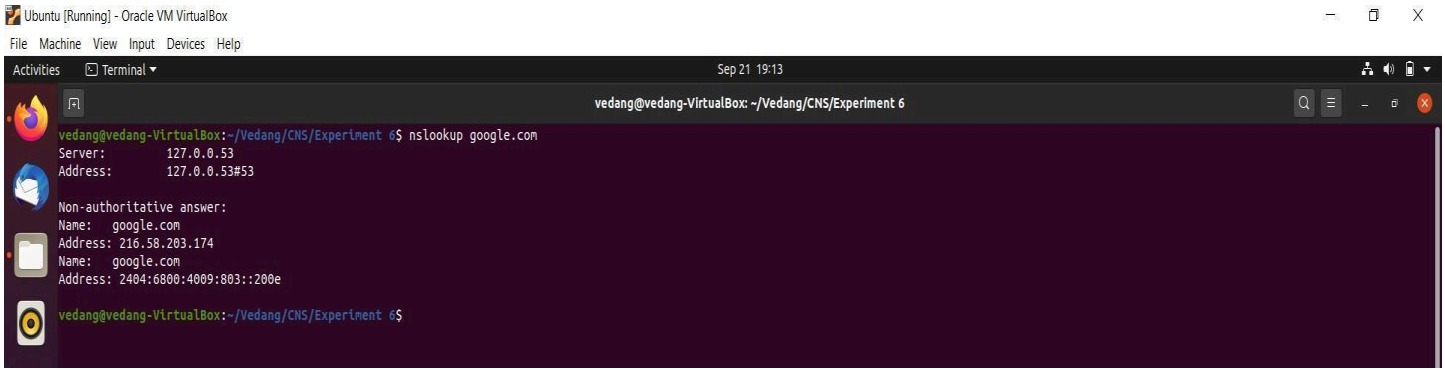
3) Traceroute :-

USE:-The traceroute, tracert, or tracepath command is similar to ping, but provides information about the path a packet takes. traceroute sends packets to a destination, asking each Internet router along the way to reply when it passes on the packet. This will show you the path packets take when you send them between your location and a destination.
Traceroute prints the route that packets take to a network host.



4) NSLookup :-

USE:-The nslookup command will look up the IP addresses associated with a domain name. It is used to query internet name servers interactively for information. nslookup, which stands for "name server lookup", is a useful tool for finding out information about a named domain**.**



CONCLUSION/ Outcome:

Hence, we implemented the use of network reconnaissance tools like WHOIS, dig, traceroute, nslookup to gather information about networks and domain registrars.

Marks & Signature:

R1 (5 Marks)	R2 (5 Marks)	R3 (5 Marks)	Total (15 Marks)	Signature

EXPERIMENT NUMBER: 8

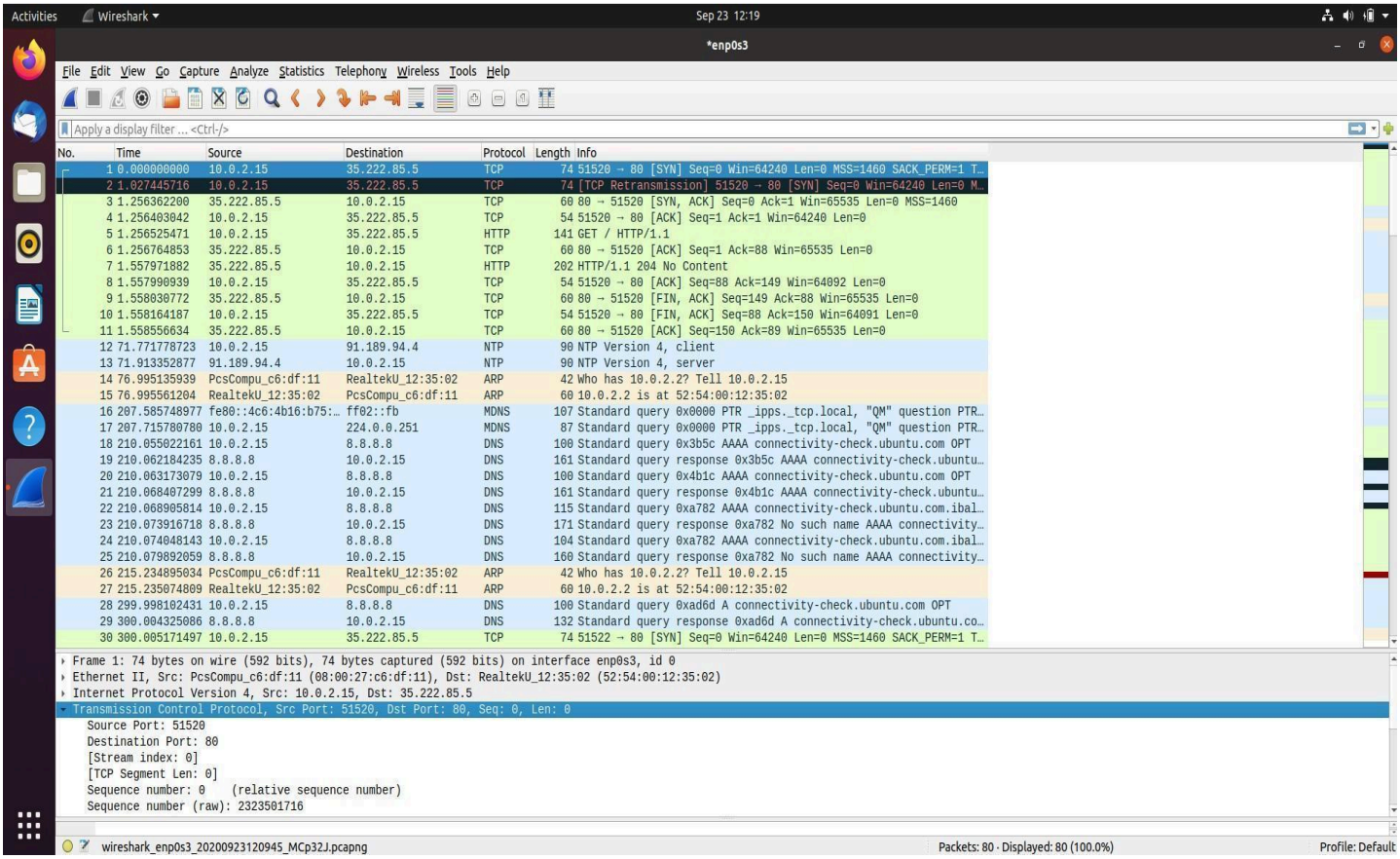
Date of Performance :	
Date of Submission :	

AIM: Study of packet sniffer tools wireshark: -
a. Observer performance in promiscuous as well as non-promiscuous mode.
b. Show the packets can be traced based on different filters.

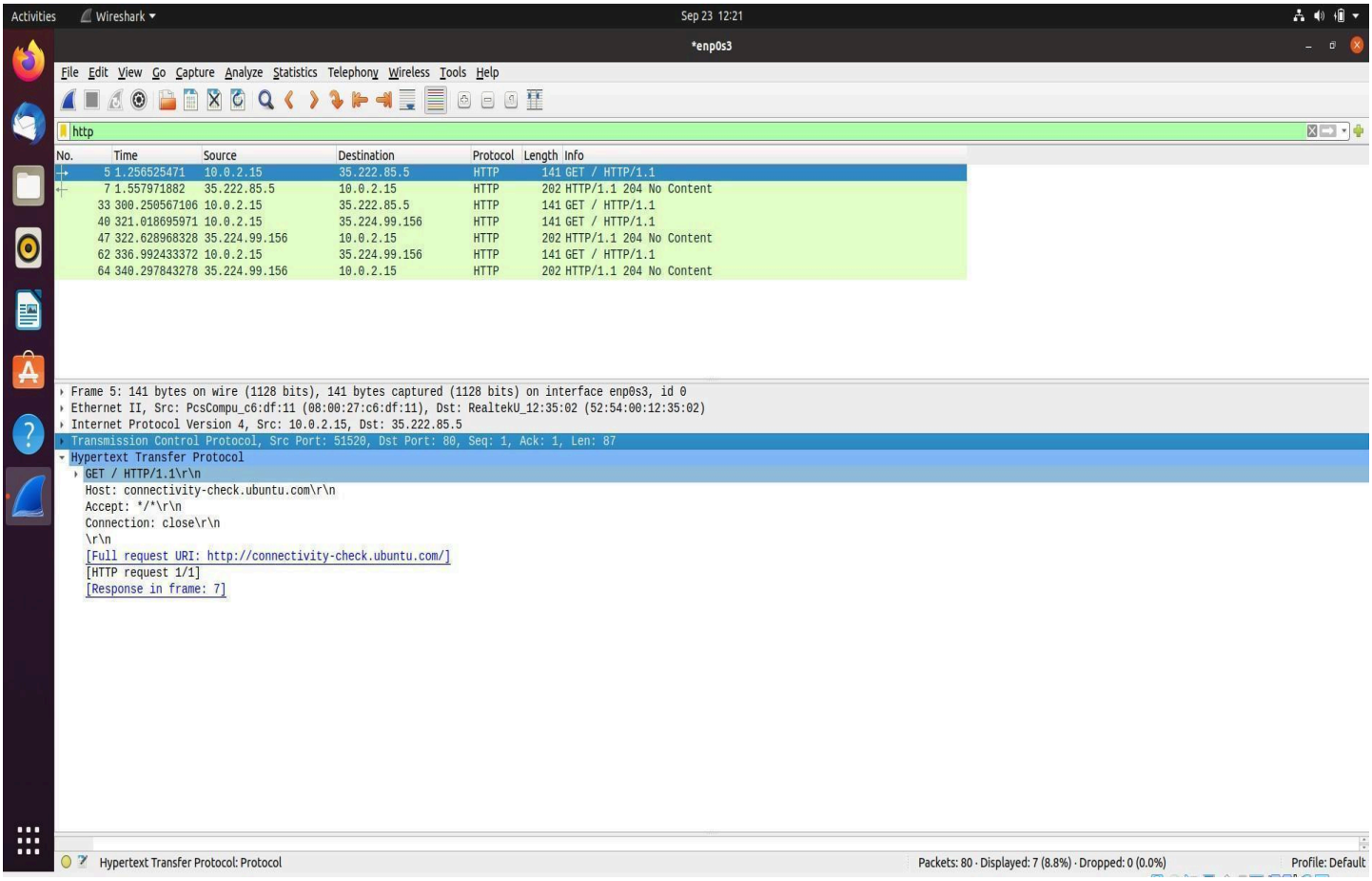
THEORY:

1. Observer performance in promiscuous as well as non-promiscuous mode.

Promiscuous Mode :-

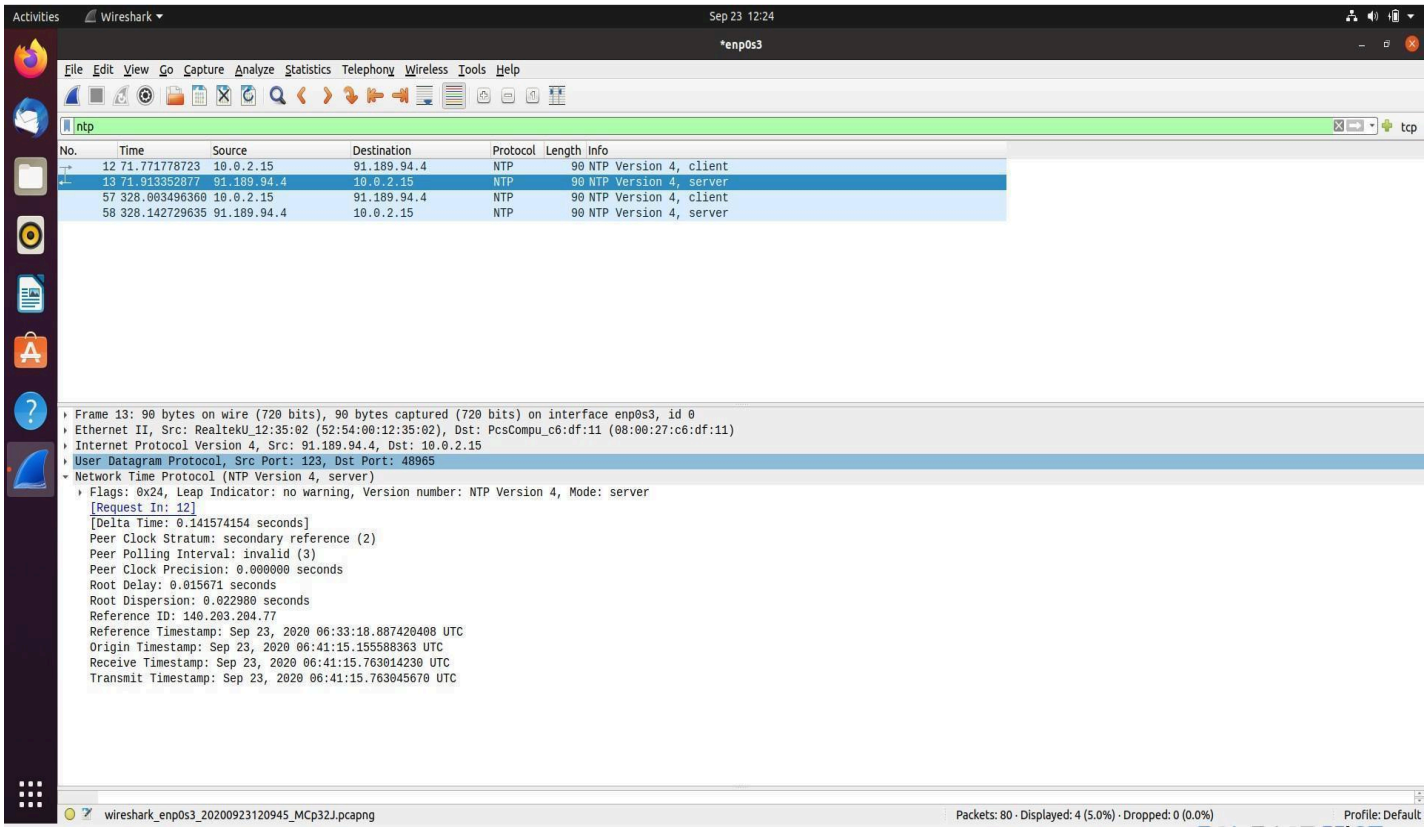


Packets captured in Non-Promiscuous Mode and with http filter :-

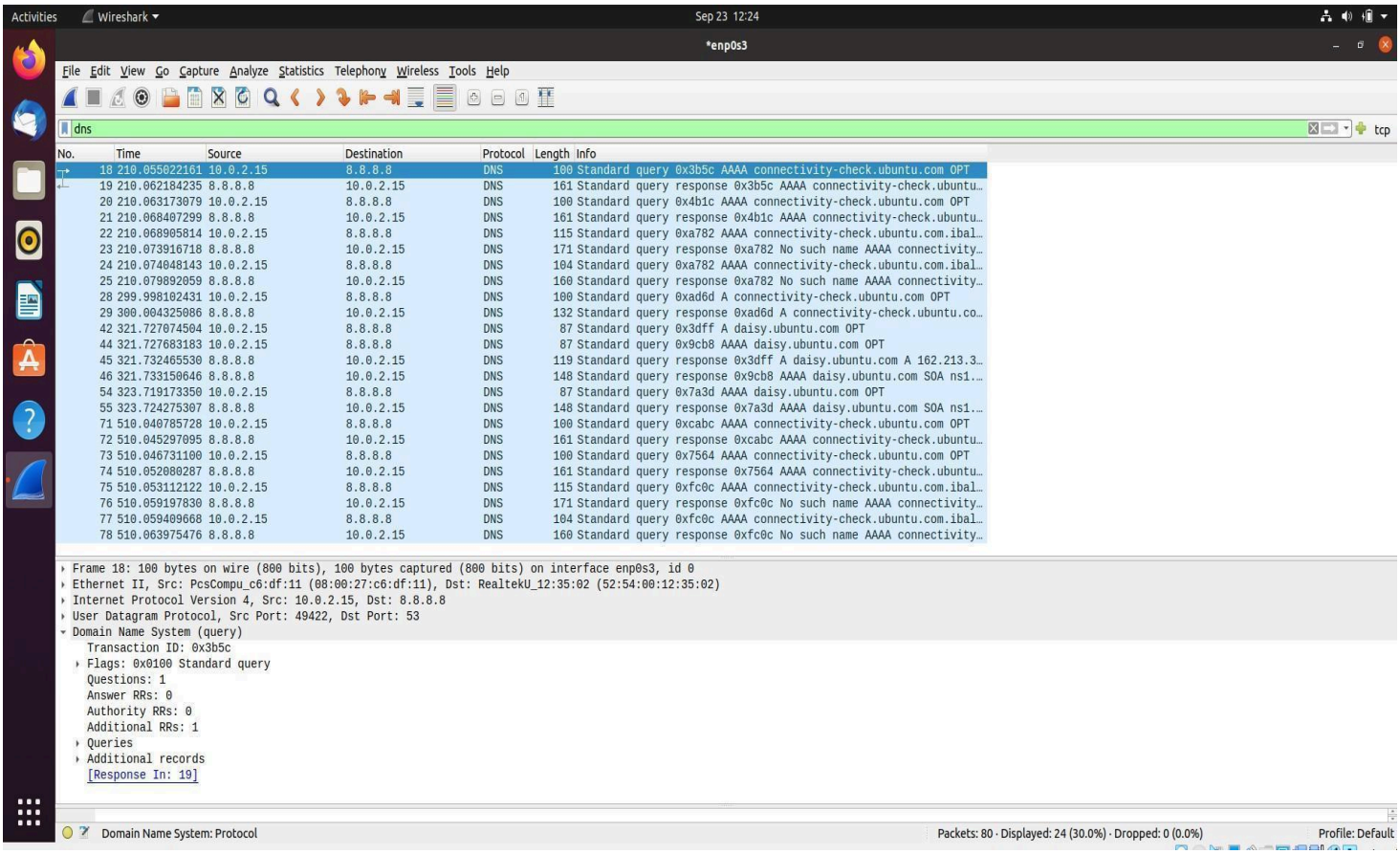


2. Show the packets can be traced based on different filters.

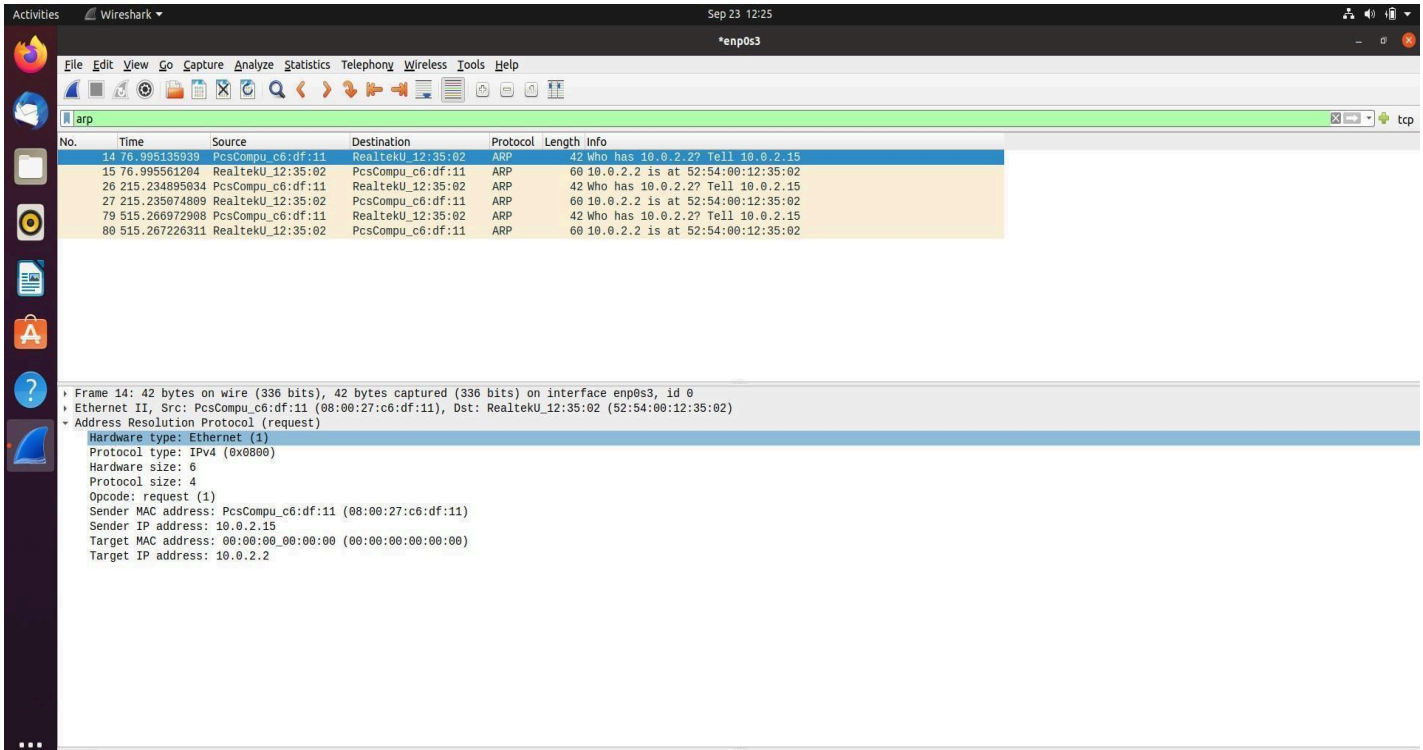
1) NTP Filter :-



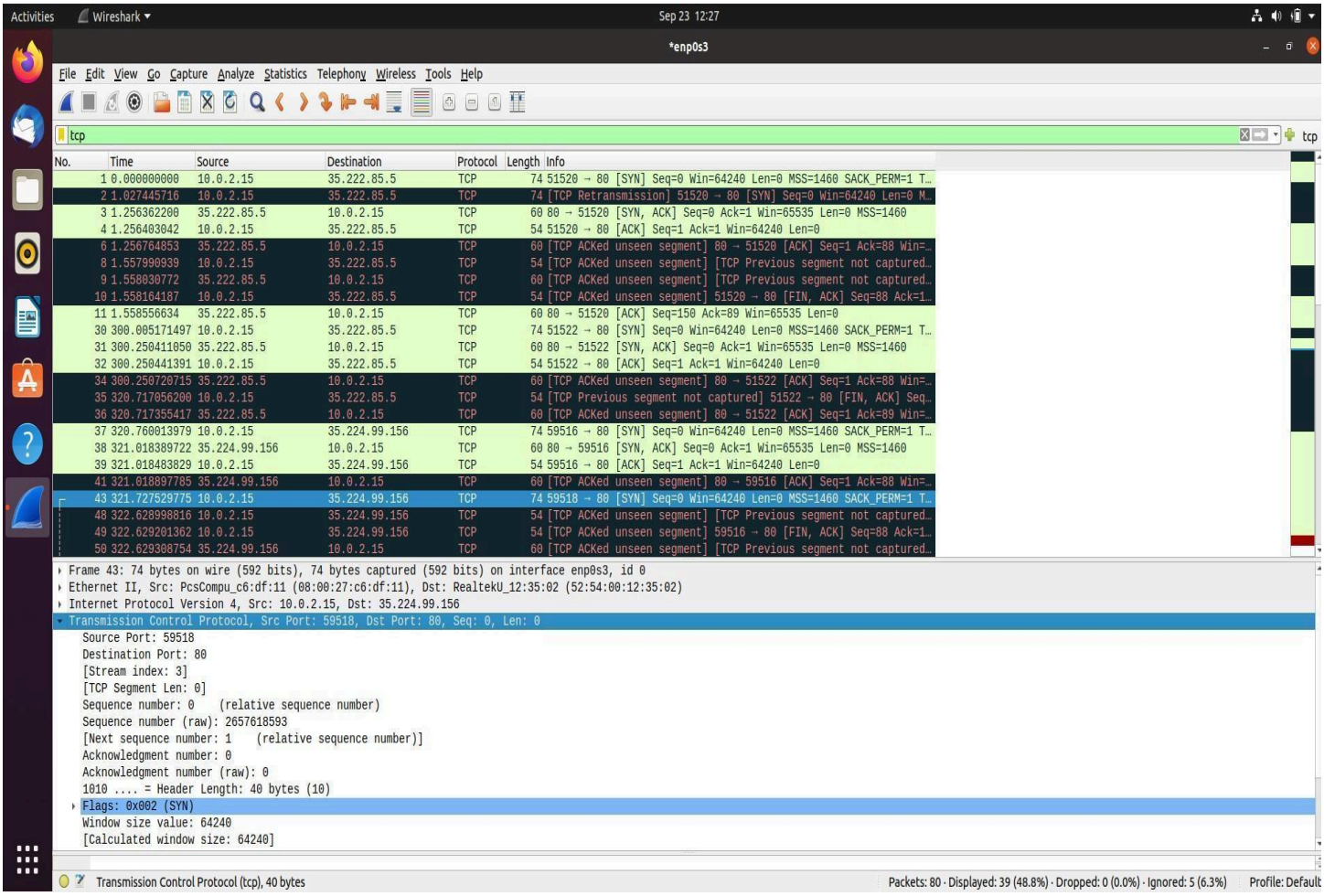
2) DNS Filter :-



3) ARP Filter :-



4) TCP Filter :-



CONCLUSION/ Outcome:

Hence, we implement nmap and use it with different options to scan open ports, perform OS fingerprinting, ping scan, tcp port scan, udp port scan, etc.

Marks & Signature:

R1 (5 Marks)	R2 (5 Marks)	R3 (5 Marks)	Total (15 Marks)	Signature

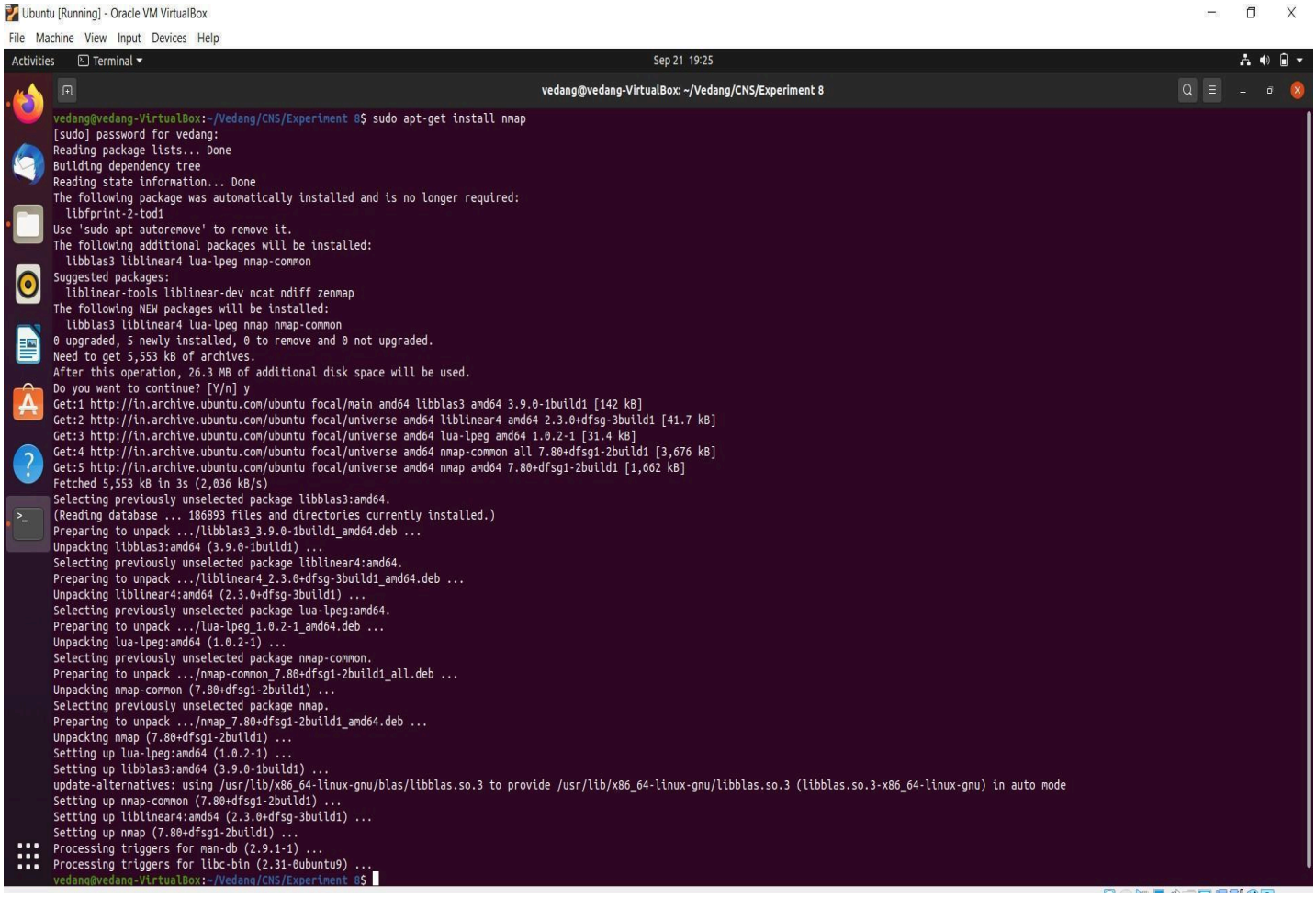
EXPERIMENT NUMBER: 9

Date of Performance :	
Date of Submission :	

AIM: Download, install nmap and use it with different options to scan open ports, perform OS fingerprinting, ping scan, tcp port scan, udp port scan, etc.

NmapInstallation :-

Command :-$ sudo apt-get install nmap

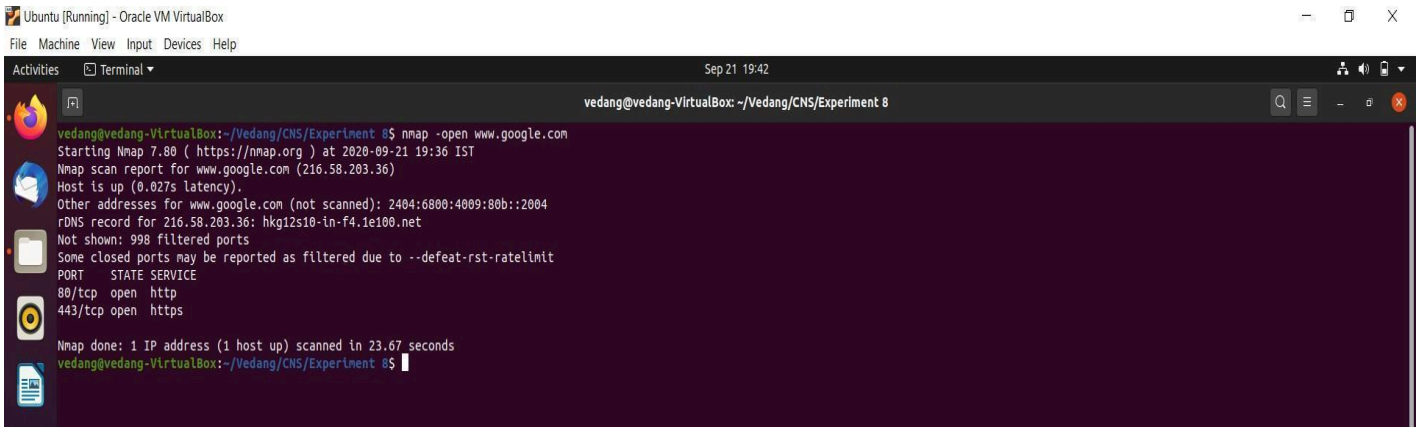


Nmap (Network Mapper) is a security scanner originally written by Gordon Lyon (also known by his pseudonym Fyodor Vaskovich) used to discover hosts and services on a computer network, thus creating a "map" of the network. To accomplish its goal, Nmap sends specially crafted packets to the target host and then analyzes the responses. Unlike many simple port scanners that just send packets at some predefined constant rate, Nmap accounts for the network conditions (latency fluctuations, network congestion, the target interference with the scan) during the run.

Different options to scan :-

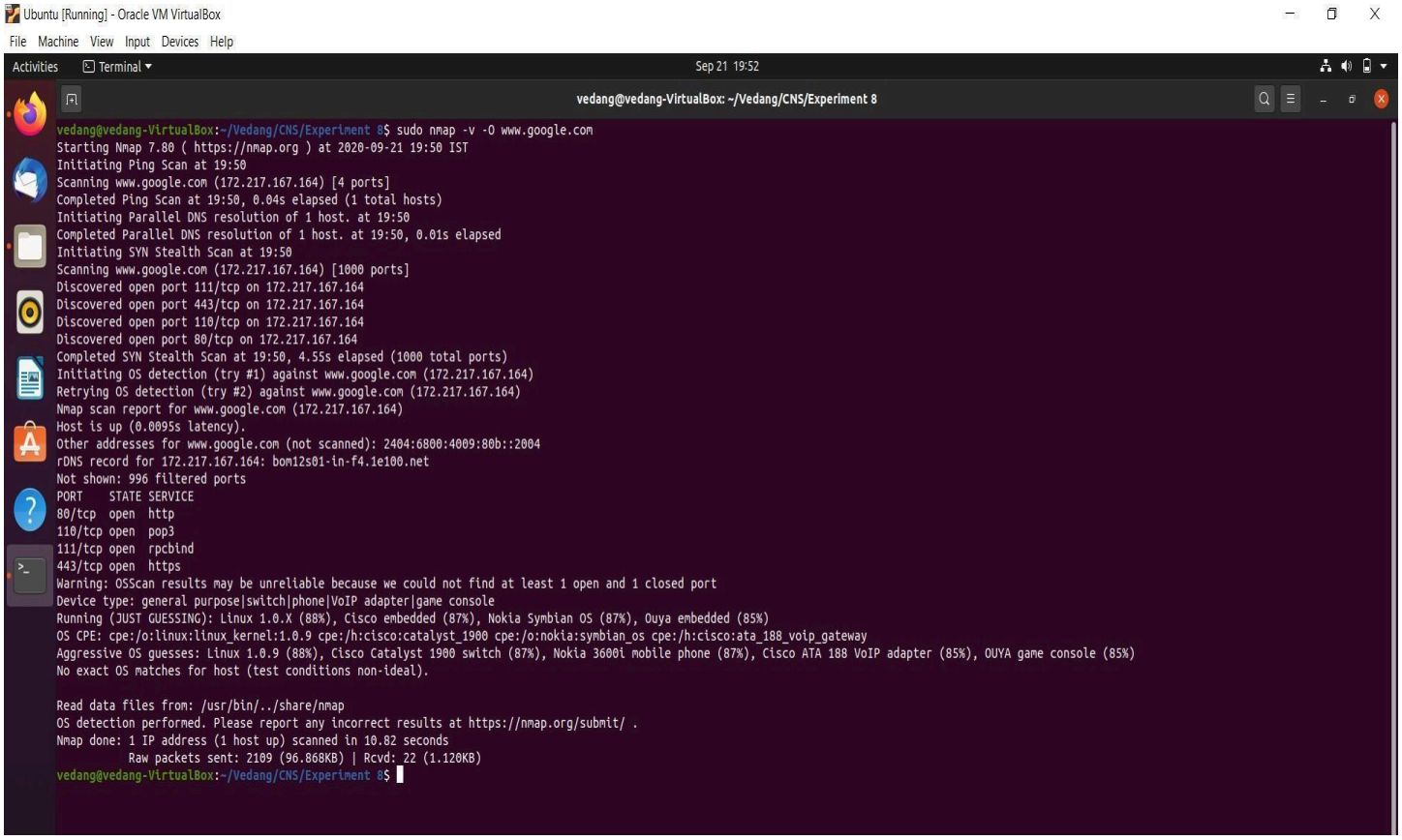
1) Scan Open Ports :-

USE:-A **port scan** or port scanning can be defined as a process that sends client requests to a range of server port addresses on a host, with the goal of finding an active port.



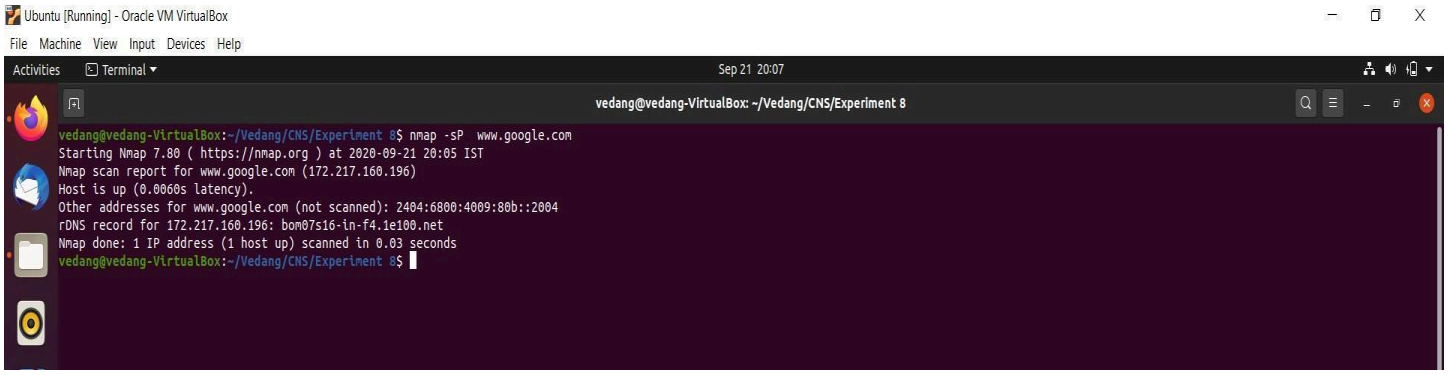
2) Perform OS Fingerprinting :-

USE:-TCP/IP stack fingerprinting is the passive collection of configuration attributes from a remote device during standard layer 4 network communications. The combination of parameters may then be used to infer the remote machine's operating system (aka, **OS fingerprinting**), or incorporated into a device fingerprint.



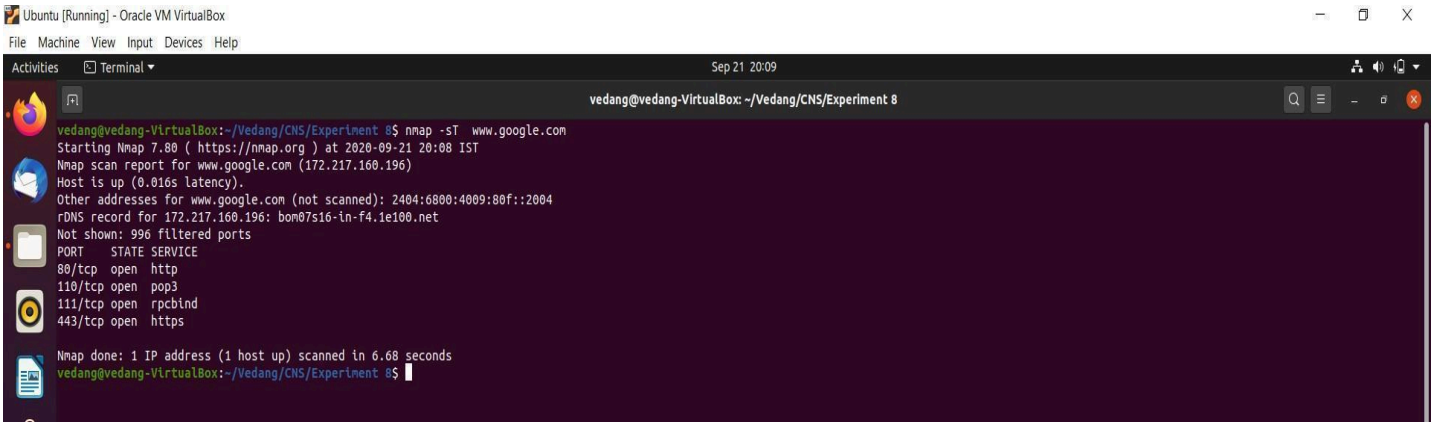
3) Ping Scan :-

USE:- **Ping Scan** In Nmap Is Done To Check If The Target Host Is Alive Or Not.



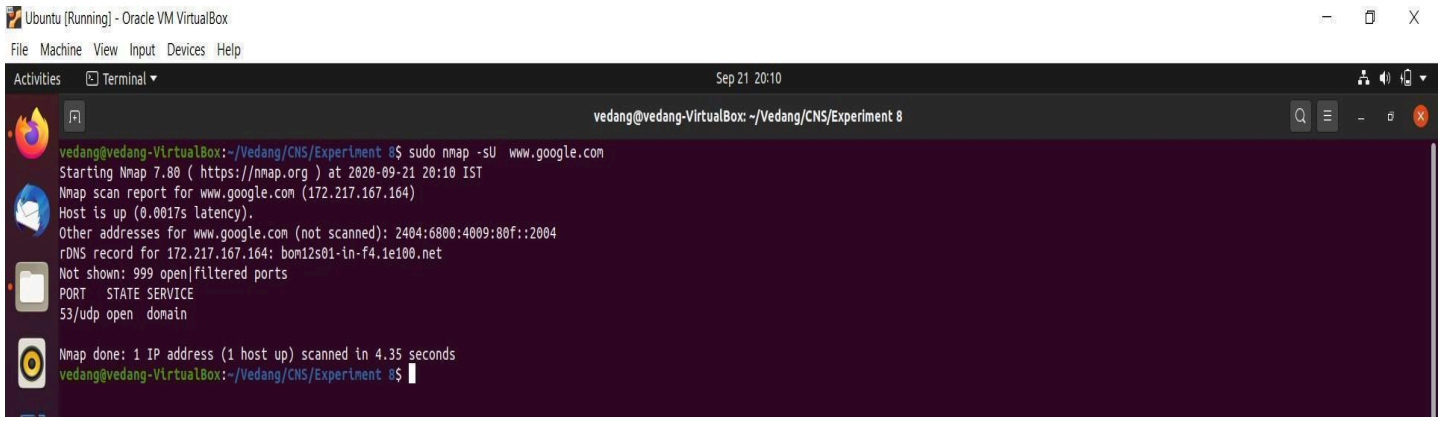
4) TCP Port Scan:-

USE:- TCP connect scan is the default TCP scan type when SYN scan is not an option. This is the case when a user does not have raw packet privileges. Instead of writing raw packets as most other scan types do, Nmap asks the underlying operating system to establish a connection with the target machine and port by issuing the connect system call. It completes Three-Way Handshake And Port Scanner Closes Connection To Avoid Performing A DOS Attack.



5) UDP Port Scan:-

USE:- UDP scan works by sending a UDP packet to every targeted port. UDP scan is activated with the -sU option. It can be combined with a TCP scan type such as SYN scan (-sS) to check both protocols during the same run.



CONCLUSION/ Outcome:

Hence, we implement nmap and use it with different options to scan open ports, perform OS fingerprinting, ping scan, tcp port scan, udp port scan, etc.

Marks & Signature:

R1 (5 Marks)	R2 (5 Marks)	R3 (5 Marks)	Total (15 Marks)	Signature

EXPERIMENT NUMBER: 10

Date of Performance :	
Date of Submission :	

AIM: Study of malicious software using different tools:
a) Keylogger attack using a keylogger tool.
b) Simulate DOS attack using Hping or other tools
c) Use the NESSUS/ISO Kali Linux tool to scan the network for vulnerabilities.

THEORY :

Keyloggers :

A keylogger is an insidious form of spyware. You enter sensitive data onto your keyboard, believing nobody is watching. In fact, keylogging software is hard at work logging everything that you type.

Keyloggers are activity-monitoring software programs that give hackers access to your personal data. The passwords and credit card numbers you type, the webpages you visit – all by logging your keyboard strokes. The software is installed on your computer, and records everything you type. Then it sends this log file to a server, where cybercriminals wait to make use of all this sensitive information.

Some keyloggers are hardware devices embedded within your internal PC hardware. They also come as a form of a plug placed between the CPU box and keyboard cable in an inconspicuous manner. In either case, someone will have to physically plant the hardware into your PC or its peripherals. This will require a degree of secrecy if it needs to be achieved clandestinely.

The second type of keyloggers are software that can be easily installed on victims' devices. While this software is a type of malware, it is "good" malware, wherein it doesn't harm its host. Its sole job is to snoop into the keystrokes and not impact the computer. You merrily go about your business, while undetectable keyloggers start stealing personal or sensitive data, without you ever knowing.

DOS Attack

A Denial-of-Service (DoS) attack is an attack meant to shut down a machine or network, making it inaccessible to its intended users. DoS attacks accomplish this by flooding the target with traffic, or sending it information that triggers a crash. In both instances, the DoS attack deprives legitimate users (i.e.employees, members, or account holders) of the service or resource they expected.

Victims of DoS attacks often target web servers of high-profile organizations such as banking, commerce, and media companies, or government and trade organizations. Though DoS attacks do not typically result in the theft or loss of significant information or other assets, they can cost the victim a great deal of time and money to handle.

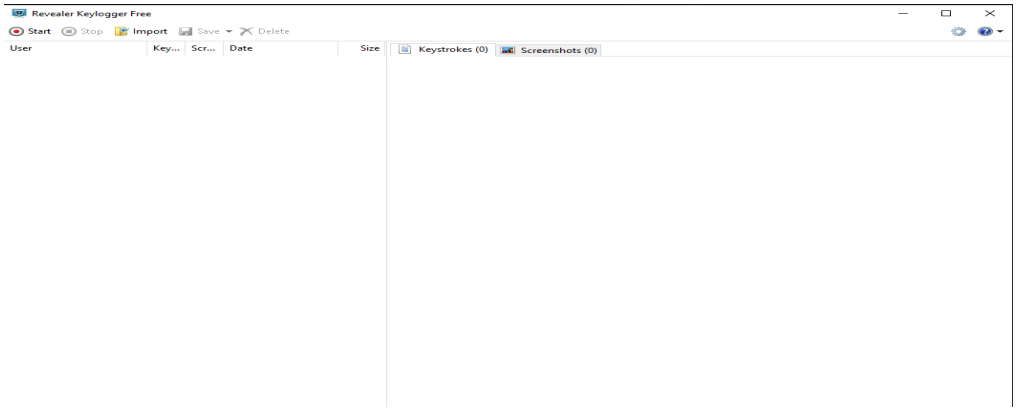
hping

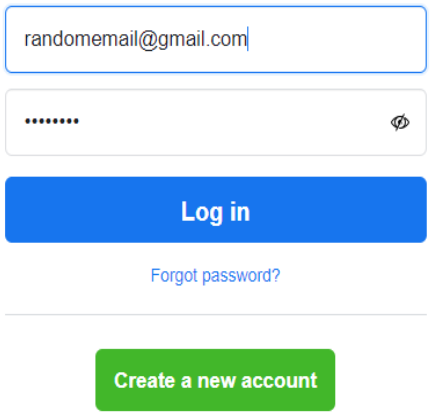
hping is a command-line oriented TCP/IP packet assembler/analyzer. The interface is inspired to the ping(8) unix command, but hping isn't only able to send ICMP echo requests. It supports TCP, UDP, ICMP and RAW-IP protocols, has a traceroute mode, the ability to send files between a covered channel, and many other features.

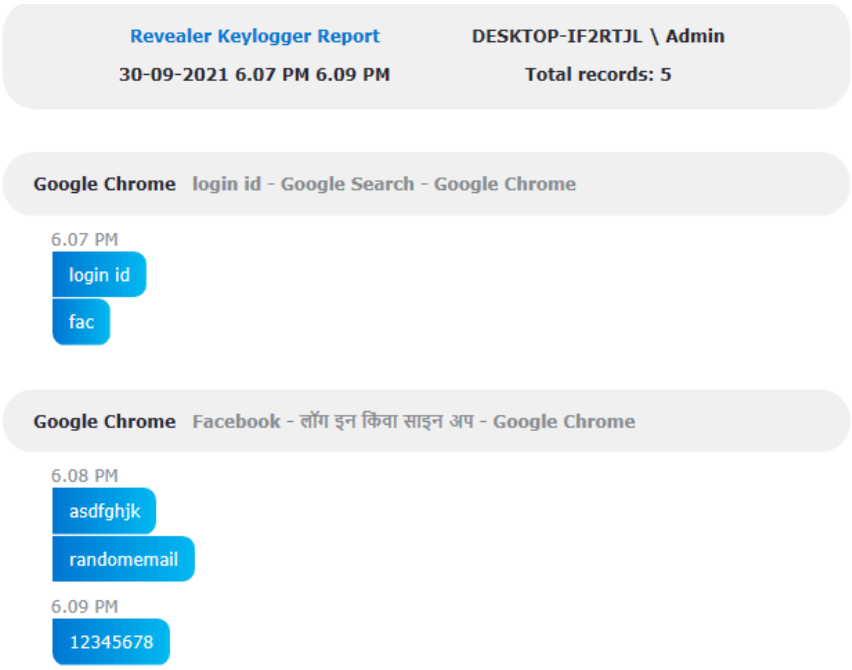
While hping was mainly used as a security tool in the past, it can be used in many ways by people that don't care about security to test networks and hosts. A subset of the stuff you can do using hping:

- Firewall testing
- Advanced port scanning
- Network testing, using different protocols, TOS, fragmentation
- Manual path MTU discovery
- Advanced traceroute, under all the supported protocols
- Remote OS fingerprinting
- Remote uptime guessing
- TCP/IP stacks auditing
- hping can also be useful to students that are learning TCP/IP.

Keylogger attack using a keylogger tool.







2) Nessus is a proprietary comprehensive vulnerability scanner which is developed by Tenable Network Security. It is free of charge for personal use in a non-enterprise environment.

Operation

- Nessus allows scans for the following types ofvulnerabilities:
  - Vulnerabilities that allow a remote hacker to control or access sensitive data on asystem.
- Misconfiguration (e.g. open mail relay, missing patches,etc.).

Default passwords, a few common passwords, and blank/absent passwords on some system accounts. Nessus can also call Hydra (an external tool) to launch a dictionary attack.Denials of service against the TCP/IP stack by using malformed packets

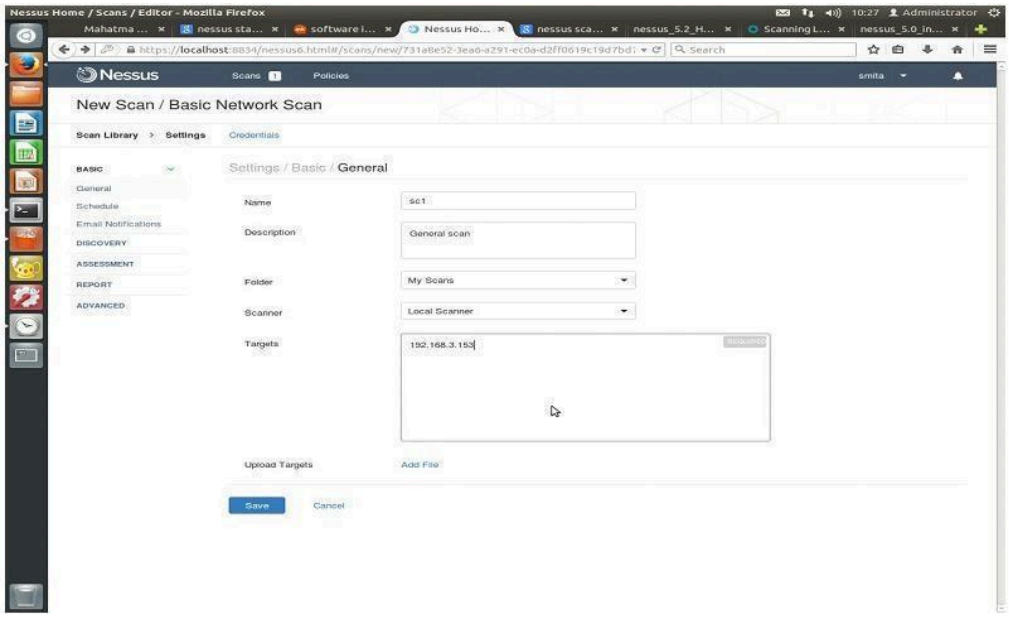
Preparation for PCI DSS audits

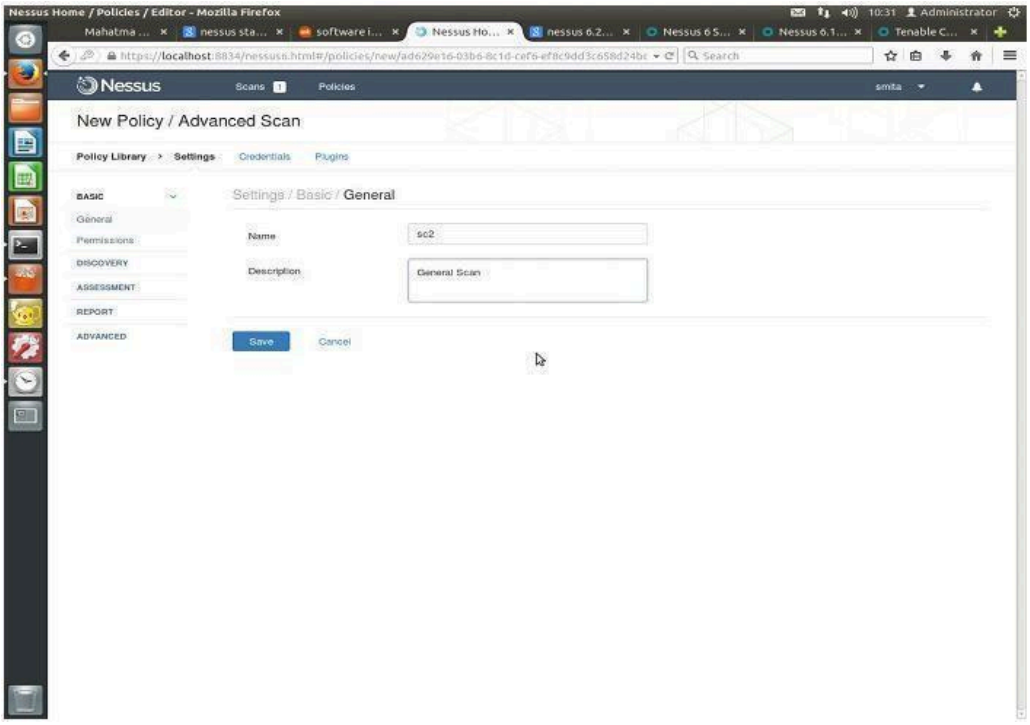
On UNIX (including Mac OS X), it consists of nessusd, the Nessus daemon, which does the scanning, and nessus, the client, which controls scans and presents the vulnerability results to the user. In typical operation, Nessus begins by doing a port scan with one of its four internal

port scanners (or it can optionally use AmapM[4] or Nmap[5]) to determine which ports are open on the target and then tries various exploits on the open ports. The vulnerability tests, available as subscriptions, are written in NASL(Nessus Attack Scripting Language), a scripting language optimized for custom network interaction. Tenable Network Security produces several dozen new vulnerability checks (called plugins) each week, usually on a daily basis. These checks are available for free to the general public; commercial customers are not allowed to use this Home Feed any more. The Professional Feed (which is not free) also give access to support and additional scripts (e.g. audit files, compliance tests, additional vulnerability detection plugins). Optionally, the results of the scan can be reported in various formats, such as plain text, XML, HTML and LaTeX. The results can also be saved in a knowledge base for debugging. On UNIX, scanning can be automated through the use of a command-line client. There exist many different commercial, free and open source tools for both UNIX and Windows to manage individual or distributed Nessus scanners. If the user chooses to do so (by disabling the option 'safe checks'), some of Nessus' vulnerability test may try to cause vulnerable services or operating systems to crash. This lets a user test the resistance of a device before putting it in production. Nessus provides additional functionality beyond testing for known network vulnerabilities. For instance, it can use Windows credentials to examine patch levels on computers running the Windows operating system, and can perform password auditing using dictionary and brute force methods. Nessus 3 and later can also audit systems to make sure they have been configured per a specific policy, such as the NSA's guide for hardening Windowsservers.

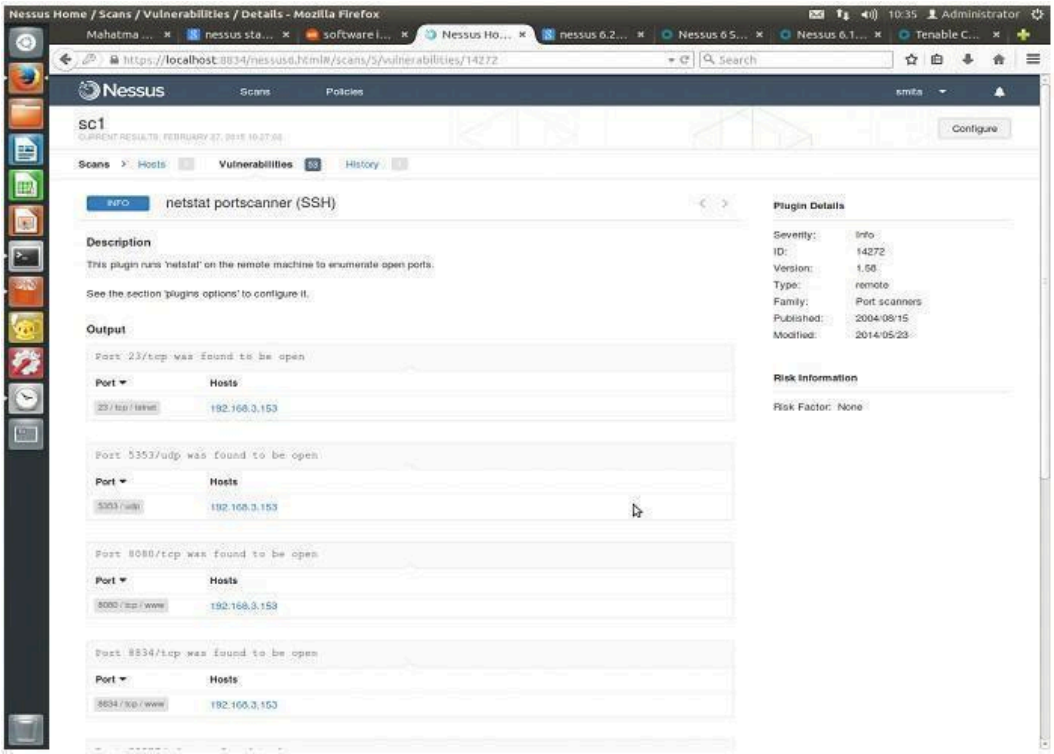
Basic Network scanning:

Advanced scanning in general search:

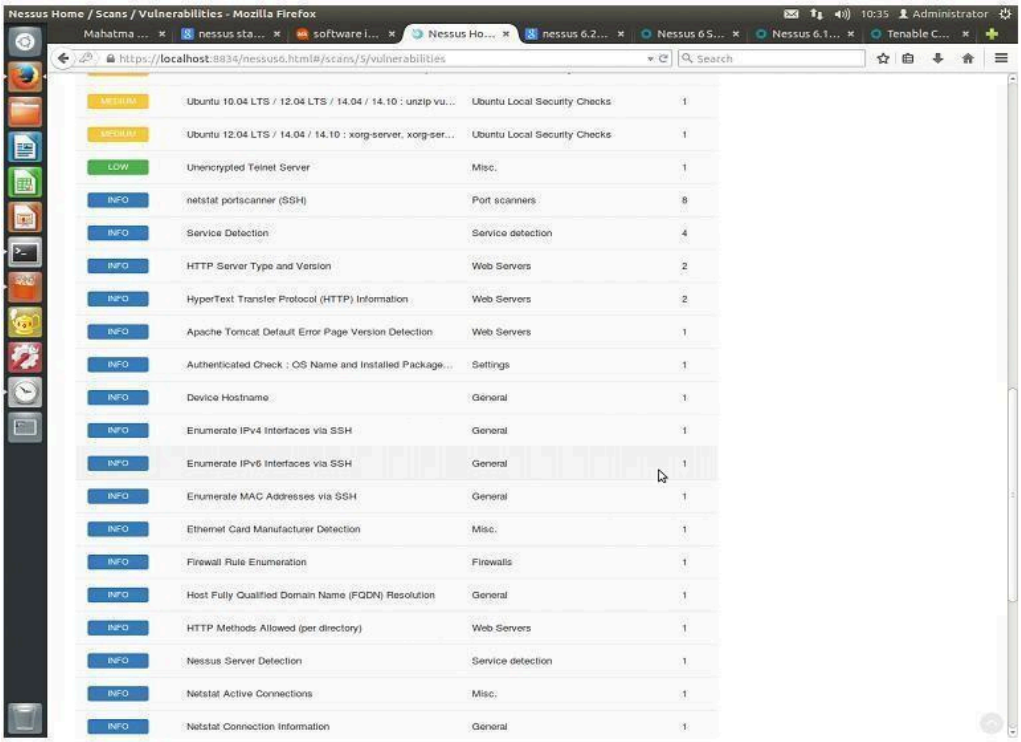




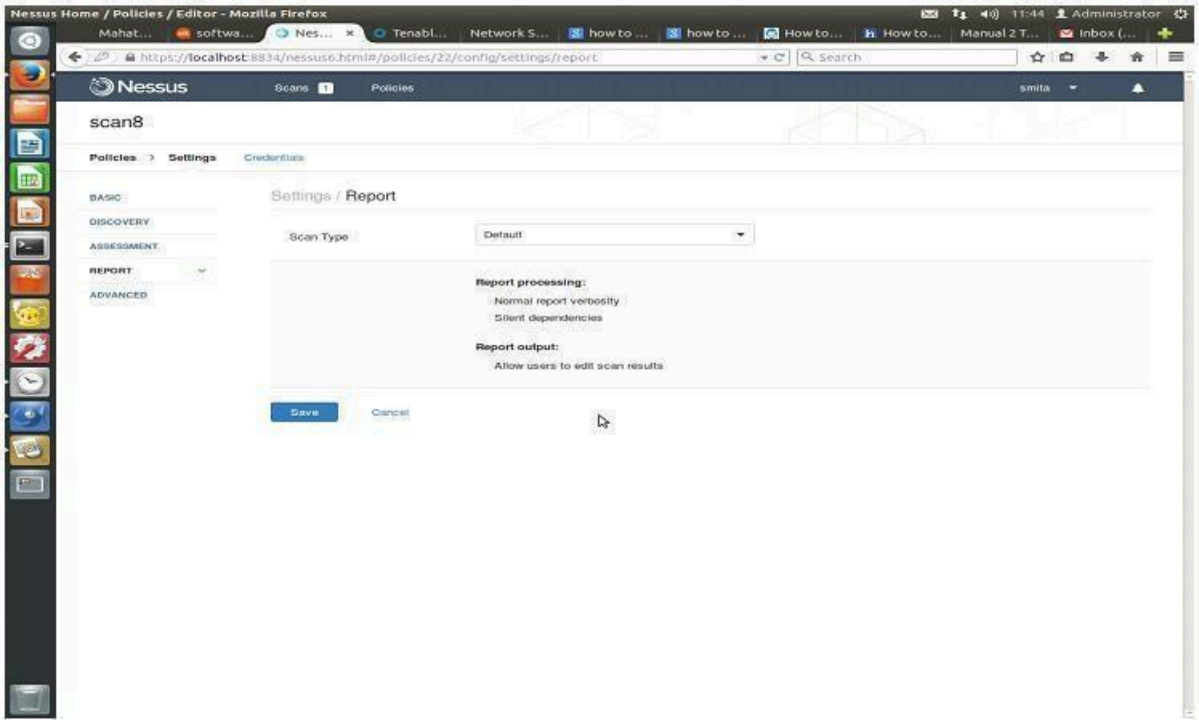
Ntstat port scanning:



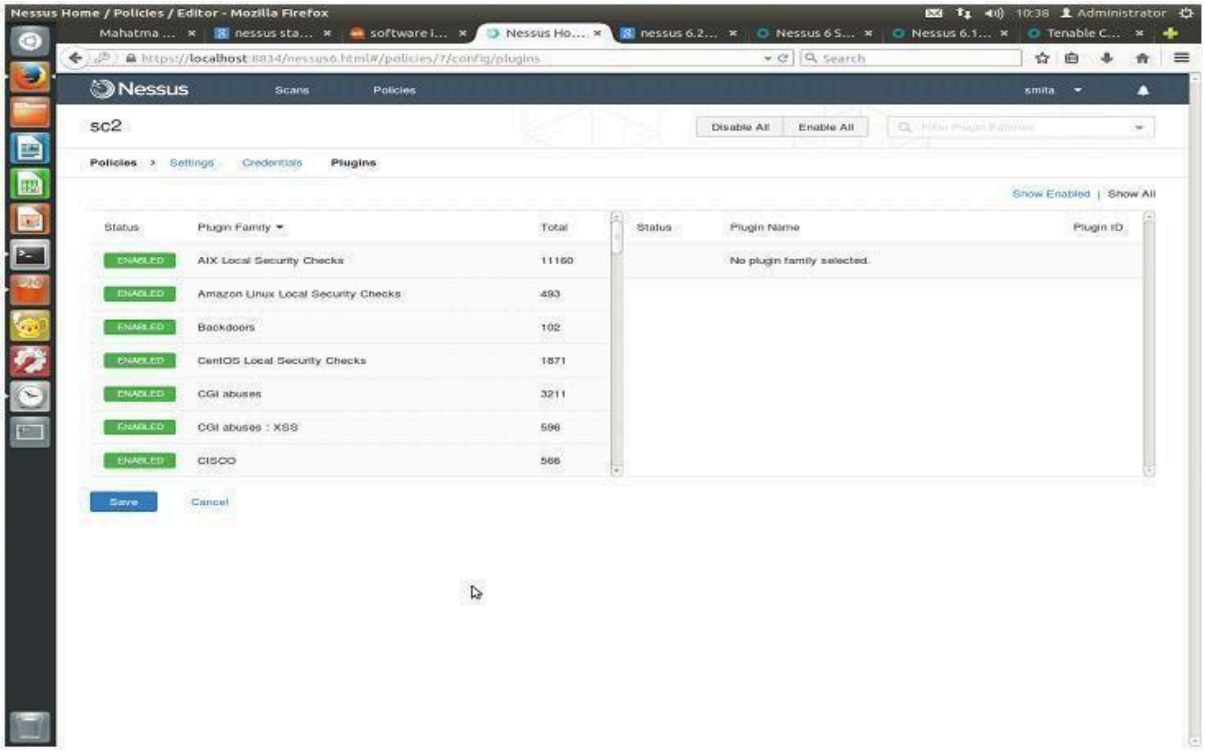
Vulnerability Mapping:



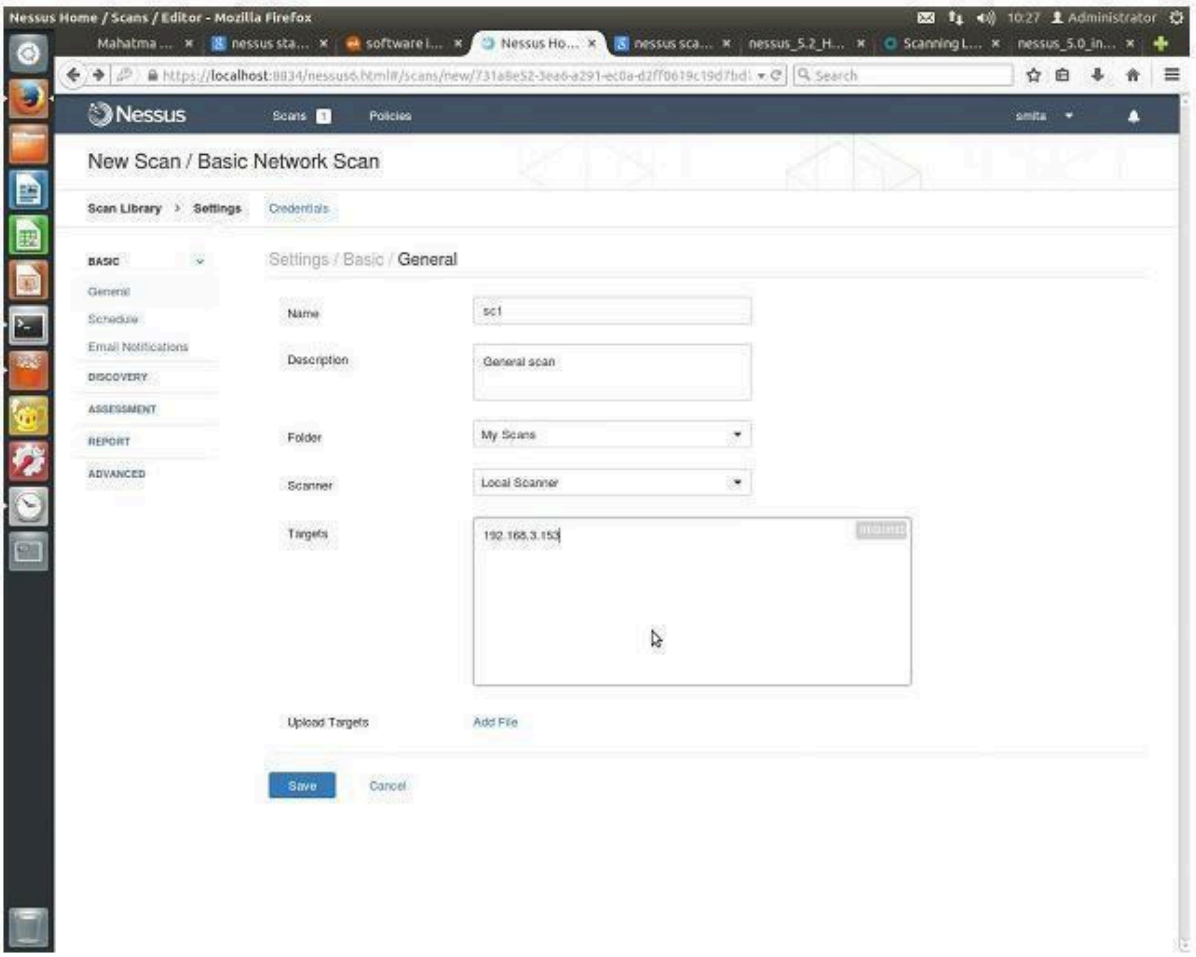
Policies:



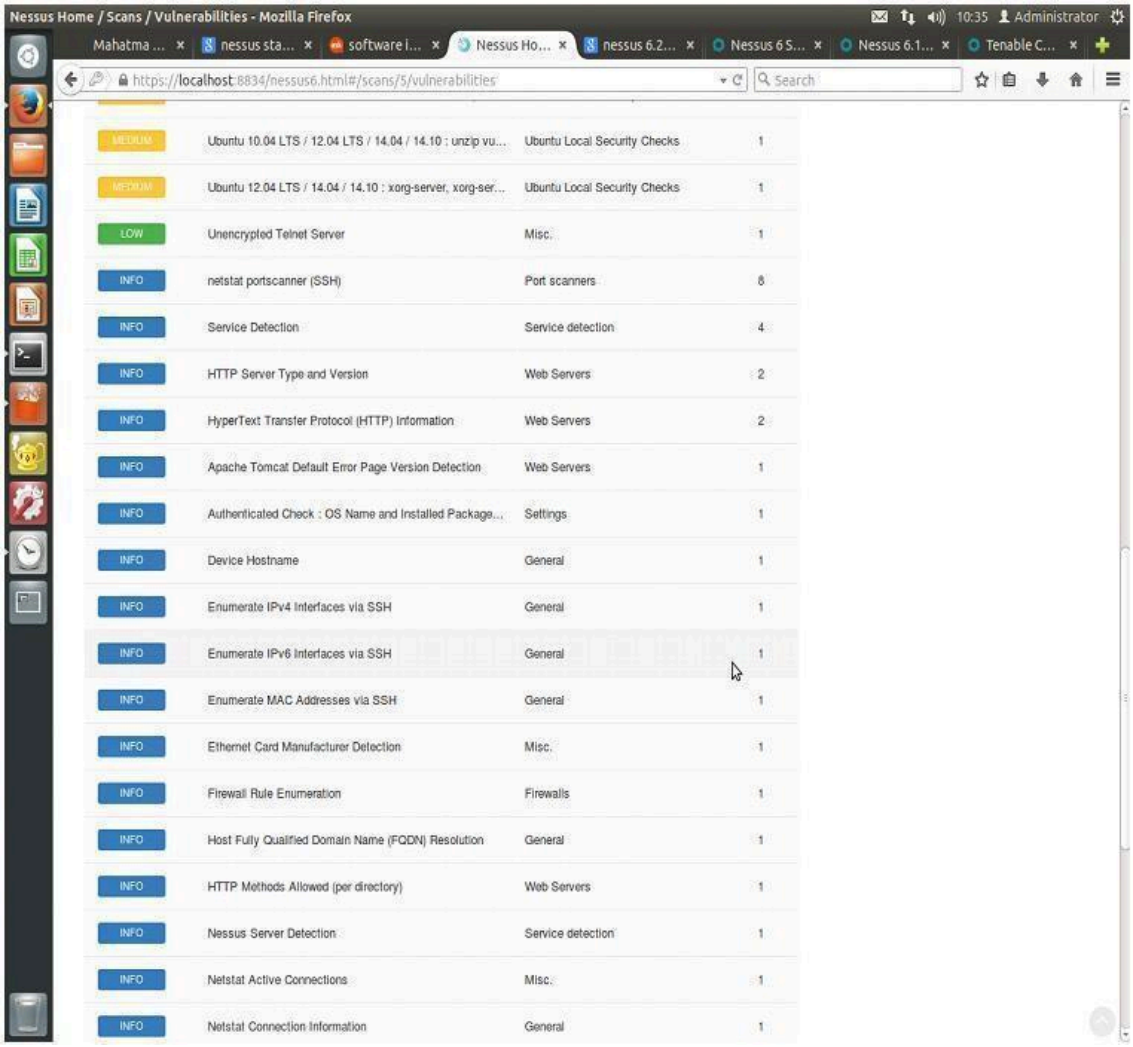
Plugins:



General Scanning:



Port Scanning:



CONCLUSION/ Outcome:

:Thus we have successfully studied about keyloggers, dos attack, hping, nessus tool and implemented it.

Marks & Signature:

R1 (5 Marks)	R2 (5 Marks)	R3 (5 Marks)	Total (15 Marks)	Signature

Date of Performance :	
Date of Submission :	

AIM: Study of Network security by
a) Set up IPSec under Linux.
b) Set up Snort and study the logs.
c) Explore the GPG tool to implement email security

THEORY:

Internet Protocol Security (IPsec) is a protocol suite for securing Internet Protocol (IP) communications by authenticating and encrypting each IP packet of a communication session. IPsec includes protocols for establishing mutual authentication between agents at the beginning of the session and negotiation of cryptographic keys to be used during the session. IPsec can be used in protecting data flows between a pair of hosts (host-to-host), between a pair of security gateways (network-to-network), or between a security gateway and a host (network-to-host).

Internet Protocol security (IPsec) uses cryptographic security services to protect communications over Internet Protocol (IP) networks. IPsec supports network-level peer authentication, data origin authentication, data integrity, data confidentiality (encryption), and replay protection.

IPsec is an end-to-end security scheme operating in the Internet Layer of the Internet Protocol Suite, while some other Internet security systems in widespread use, such as Transport Layer Security (TLS) and Secure Shell (SSH), operate in the upper layers at Application layer. Hence, only IPsec protects any application traffic over an IP network. Applications can be automatically secured by IPsec at the IP layer.

The following commands will add the werner-jaeger PPA into your repo's, and then install the 'l2tp-ipsec-vpn'package:

>>sudo apt-add-repository ppa:werner-jaeger/ppa-werner-vpn

>>sudo apt-get update

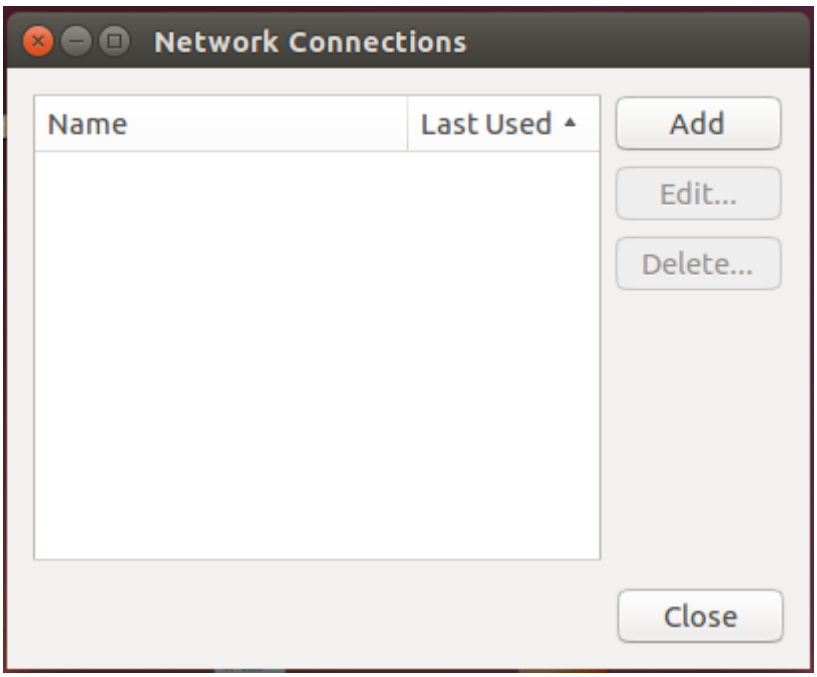
>>sudo apt-get install l2tp-ipsec-vpn

· Now, we will whitelist our system tray which will allow our newly installed package to show up on our systemtray:

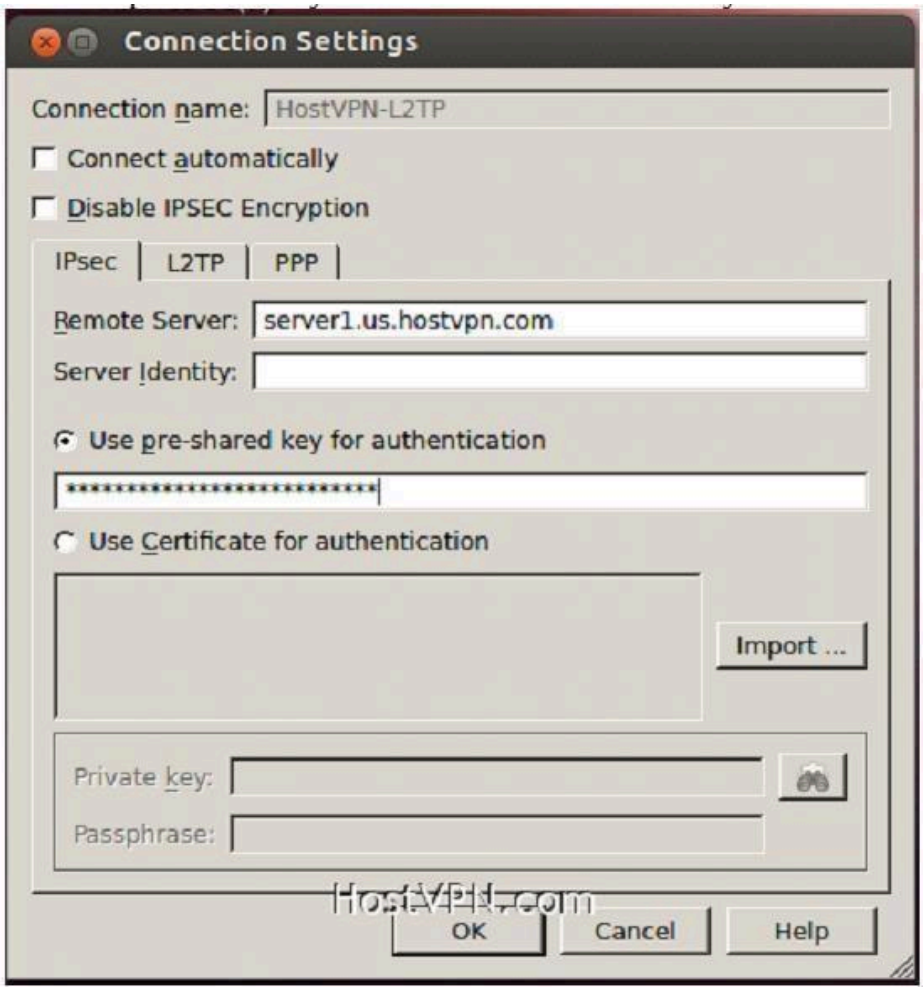
>>gsettings set com.canonical.Unity.Panelsystray-whitelist "['all']"

- After whitelisting our system tray, it's imperative that you reboot/restart your machine.
- Once your machine has rebooted, click on the new icon, and click 'Edit Connections ...' from themenu.

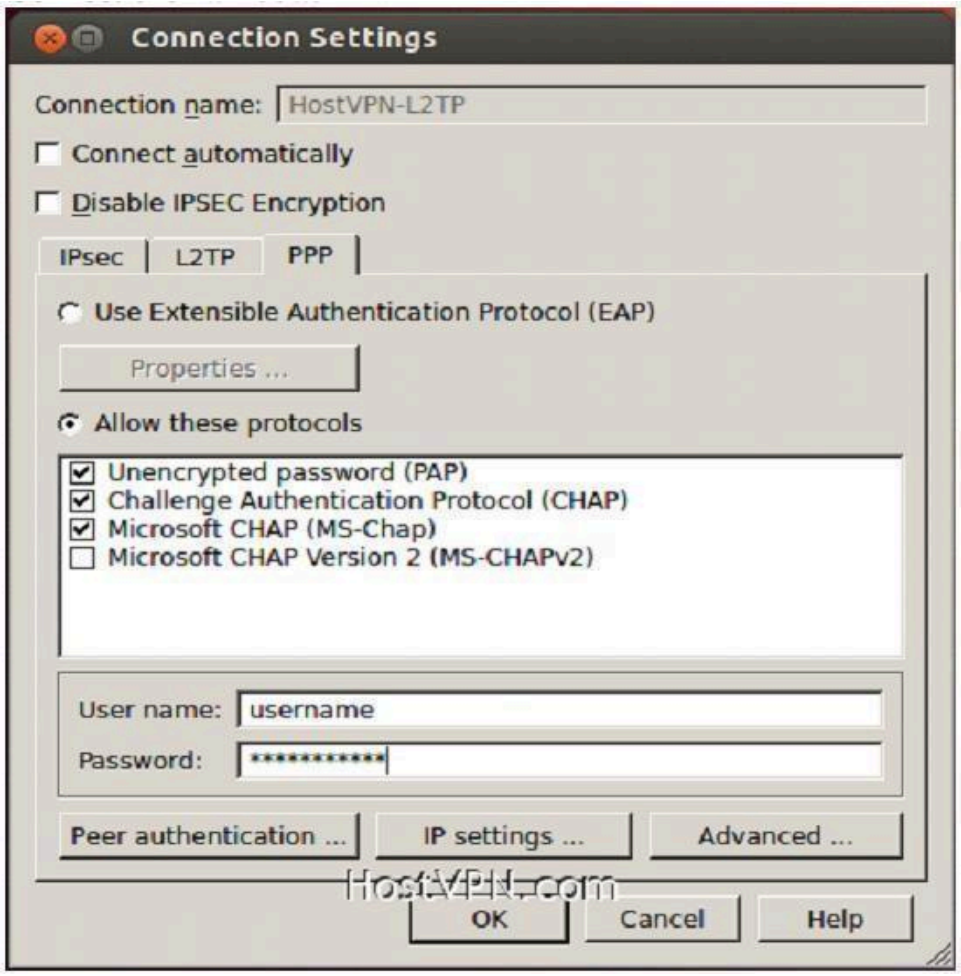
- This will show the "VPN Connections" window. Click the "Add ..." button and set the connection name to anything you'd like, e.g. "HostVPN-L2TP", and click"OK".



- Now select your newly added connection, and click "Edit...".
- On the IPSec tab, set the remote server to the server name from your HostVPN e-mail. Select the "Use pre-shared key for authentication" and enter your PSK from the HostVPNe-mail.



- On the PPP tab, select "Allow these protocols", and ensure all are selected except "Microsoft CHAP Version 2 (MS-CHAPv2)". Fill in the "User name:" and "Password:" fields with your HostVPN username and password, and then click "OK". Now click "Close" on the "VPN Connections"window.



- Click on the L2TP/IPSec VPN icon in the systray again and click on the connection name that we justcreated.

1. Theory:

Snort is an intrusion detection system written by Martin Roesch. Snort is a lightweight network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching/matching and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more. Snort uses a flexible rules language to describe traffic that it should collect or pass, as well as a detection engine that utilizes a modular pluginarchitecture.

Snort can be configured to run in three modes:

- ***Sniffer mode,*** which simply reads the packets off of the network and displays them for you in a continuous stream on the console(screen).
- ***Packet Logger mode,*** which logs the packets todisk.
- ***Network Intrusion Detection System (NIDS) mode,*** the most complex and configurable configuration, which allows Snort to analyze network traffic for matches against a user-defined rule set and performs several actions based upon what itsees.***Inline mode,*** which obtains packets from iptables instead of from libpcap and then causes iptables to drop or pass packets based on Snort rules that use inline-specific ruletypes

Sniffer Mode:

If you just want to print out the TCP/IP packet headers to the screen (i.e. sniffer mode), try following, this command will run Snort and just show the IP and TCP/UDP/ICMP headers, nothing else.

./snort –v

If you want to see the application data in transit, try the following, this instructs Snort to display the packet data as well as the headers.

./snort –vd

If you want an even more descriptive display, showing the data link layer headers, do this:

./snort -vde

Packet Logger Mode:

If you want to record the packets to the disk, you need to specify a logging directory and Snort will automatically know to go into packet loggermode:

./snort -dev -l ./log

Of course, this assumes you have a directory named log in the current directory. If you don't, Snort will exit with an error message. When Snort runs in this mode, it collects every packet it sees and places it in a directory hierarchy based upon the IP address of one of the hosts in thedatagram.

If you just specify a plain -l switch, you may notice that Snort sometimes uses the address of the remote computer as the directory in which it places packets and sometimes it uses the local host address. In order to log relative to the home network, you need to tell Snort which network is the home network:**./snort -dev -l ./log -h 192.168.1.0/24**

This rule tells Snort that you want to print out the data link and TCP/IP headers as well as application data into the directory ./log, and you want to log the packets relative to the 192.168.1.0 class C network. All incoming packets will be recorded into subdirectories of the log directory, with the directory names being based on the address of the remote (non- 192.168.1) host.

Note: Note that if both the source and destination hosts are on the home network, they are logged to a directory with a name based on the higher of the two port numbers or, in the case of a tie, the source address.

Once the packets have been logged to the binary file, you can read the packets back out of the file with any sniffer that supports the tcpdump binary format (such as tcpdump or Ethereal). Snort can also read the packets back by using the -r switch, which puts it into playback mode. Packets from any tcpdump formatted file can be processed through Snort in any of its run modes. For example, if you wanted to run a binary log file through Snort in sniffer mode to dump the packets to the screen, you can try something like this:

./snort -dv -r packet.log

You can manipulate the data in the file in a number of ways through Snort's packet logging and intrusion detection modes, as well as with the BPF interface that's available from the command line. For example, if you only wanted to see the ICMP packets from the log file, simply specify a BPF filter at the command line and Snort will only see the ICMP packets in the file:

./snort -dvr packet.log icmp

Network Intrusion Detection System (NIDS) mode:

To enable Network Intrusion Detection System (NIDS) mode so that you don't record every single packet sent down the wire, try this:**./snort -dev -l ./log -h 192.168.1.0/24 -c snort.conf**

wheresnort.conf is the name of your rules file. This will apply the rules configured in the snort.conf file to each packet to decide if an action based upon the rule type in the file should be taken. If you don't specify an output directory for the program, it will default to
/var/log/snort.

One thing to note about the last command line is that if Snort is going to be used in a long term way as an IDS, the -v switch should be left off the command line for the sake of speed. The screen is a slow place to write data to, and packets can be dropped while writing to the display.

It's also not necessary to record the data link headers for most applications, so you can usually omit the -e switch, too.

./snort -d -h 192.168.1.0/24 -l ./log -c snort.conf

This will configure Snort to run in its most basic NIDS form, logging packets that trigger rules specified in the snort.conf in plain ASCII to disk using a hierarchical directory structure (just like packet loggermode).

Inline Mode:

Snort 2.3.0 RC1 integrated the intrusion prevention system (IPS) capability of Snort Inline into the official Snort project. Snort Inline obtains packets from iptables instead of libpcap and then uses new rule types to help iptables pass or drop packets based on Snortrules.

There are three rule types you can use when running Snort with Snort Inline:

- **drop**- The drop rule type will tell iptables to drop the packet and log it via usual Snortmeans.
- **reject**- The reject rule type will tell iptables to drop the packet, log it via usual Snort means, and send a TCP reset if the protocol is TCP or an icmp port unreachable if the protocol isUDP.
- **sdrop**- The sdrop rule type will tell iptables to drop the packet. Nothing islogged.

When using a reject rule, there are two options you can use to send TCP resets:

- You can use a RAW socket (the default behavior for Snort Inline), in which case you must have an interface that has an IP address assigned to it. If there is not an interface with an IP address assigned with access to the source of the packet, the packet will be logged and the reset packet will never make it onto the network.
- You can also now perform resets via a physical device when using iptables. We take the indev name from ip_queue and use this as the interface on which to send resets. We no longer need an IP loaded on the bridge, and can remain pretty stealthy as the config layer2_resets in snort_inline.conf takes a source MAC address which we substitue for the MAC of the bridge.

For example:

config layer2resets

tells Snort Inline to use layer2 resets and uses the MAC address of the bridge as the source MAC in the packet,and:**config layer2resets: 00:06:76:DD:5F:E3**

will tell Snort Inline to use layer2 resets and uses the source MAC of 00:06:76:DD:5F:E3 in the reset packet.

- The command-line option **-disable-inline-initialization** can be used to not initialize IPTables when in inline mode. To be used with command-line option -T to test for a valid configuration without requiring opening inline devices and adversely affecting trafficflow.

What is GnuPG?GNU Privacy Guard (GnuPG), also known as GPG, is a tool for secure communication that was created by Werner Koch as Free Software under the GNU Project. GnuPG follows the OpenPGP protocol, which defines and standardizes all the necessary components involved in sending encrypted messages–signatures, private keys, and public key certificates. This piece of free software is notably used by journalists around the world to ensure that their sensitive email communication is kept secure and private.

GPG uses a combination of symmetric-key cryptography and public-key cryptography. Public key cryptography is likely already familiar to you since it is the recommended way to authenticate when SSHing in to your Linode. Public-key cryptography uses a key-pair system where any single user has a private and public key pair. The public key can be shared with anyone, while the private key should be protected and secret to maintain the integrity of the system.

Create GPG Keys

1. Download and install the most recent version of the GPG command line tools for Ubuntu:

2. sudo apt update
3. sudo apt install gnupg

4. Create a new primary keypair:

5. gpg --full-generate-key

Several prompts will appear before the keypair is generated:

- Select (1) RSA and RSA (default) for the type of key.
- Enter 4096 for the key size.
- Specify the duration the key should be valid in days, weeks, months, or years. For example, 1y will set an expiration date of one year from the time of keypair creation.
- Enter a name, email address, and comment to associate with the key pair. Any one of these three values can be used to identify the keypair for future use. Enter the desired information for each value and confirm when prompted.
- Provide a passphrase. The passphrase is used to unlock the private key, so it is important to ensure the passphrase is strong. Use a mix of alphanumeric characters.

Once you have responded to all prompts, the keypair will be generated. This may take a few minutes to generate depending on the key size that was chosen.

If your system seems to hang at the following message:

The system may require more entropy to generate the keypair, in a new shell session, install the rng-utils package:

sudo apt install rng-tools

o Check and feed random data from an entropy source (e.g. hardware RNG device) to an entropy sink (e.g. kernel entropy pool) to provide the needed entropy for a secure keypair to be generated:

o rngd -f -r /dev/urandom

o Check the amount of entropy available on your Linode. The value should be somewhere near 3000 for keypair generation.

o cat /proc/sys/kernel/random/entropy_avail

6. Verify the keys on your public keyring:

7. gpg --list-keys

Each value in the list represents the following information:

o Public key: pub
o Key size and type: 4096R
o Short key ID: A11C0F78
o Creation date: 2018-08-02
o Expiration date: [expires: 2018-09-01]
o User IDs: exampleName2 (example comment) <user2@example.com>
o Subkey: sub

Throughout the remainder of this guide, the first public key will be used to encrypt our message. The output may vary slightly depending on the version of Ubuntu you are using.

Generate a Revocation Certificate

A revocation certificate is useful if you forget your passphrase or if your private key is somehow compromised. It is used to notify others that the public key is no longer valid. Create the revocation certificate immediately after generating your public key.

Generate a revocation certificate. Replace user@example.com with the email address associated with the public key:

gpg --output revoke.asc --gen-revoke user@example.com

- A prompt will ask you to select a reason for the revocation and provide an optional description. The default reason is recommended.
- The revocation certificate will be saved to the current directory as a file named revoke.asc. Save the certificate to a safe location on a different system so that you can access it in case your key is compromised in the future.

Once you've revoked a public key it cannot be used to encrypt future messages to you. It can still be used to verify signatures that you made in the past and to decrypt past messages sent to you.

Exchange Public Keys

You will need to exchange public keys with someone in order to securely communicate with them. If you do not want to make your key available on a key server, you can exchange keys with someone directly by exporting your public key and sending them directly to the recipient.

Export Your Public Key

1. Export the public key. Replace public-key.gpg with a desired name for the file and user@example.com with the email address associated with your key's user id:

2. gpg --armor --output public-key.gpg --export user@example.com

The file will save to the current directory.

3. Send the public-key.gpg file to the recipient in an email or copy and paste the contents of the public-key.gpg file.
4. The recipient should import the public key and validate it in order to use it to decrypt a message sent by you.

Import and Validate a Public Key

You can add someone else's public key to your public keyring by importing it. The user's public key must first be sent to you, by email or some other format, before you can import it to your public key ring. When the key is imported you should verify the key by checking its fingerprint and then signing it.

1. Once you've received the user's public key and the .gpg file is saved to your Linode, import it to your public key ring. Replace public-key.gpg with the file name of the public key you will import. If your file is saved somewhere other than the current directory, make sure you use the full path to the file:

2. gpg --import public-key.gpg

3. Verify that the public key has been added to your public key ring:

4. gpg --list-keys

5. Check the key's fingerprint:

6. gpg --fingerprint public-key.gpg

Ask the owner of the public key to send you their public key's fingerprint and verify that the fingerprint values match. If they match, you can be confident that the key you have added is a valid copy of the owner's public key.

7. When you have verified the public key's fingerprint, sign the public key with your own key to officially validate it. Replace user3@example3.com with the associated email for the key you are validating:

8. gpg --sign-key user3@example3.com

Enter your passphrase when prompted.

9. View the public key's signatures to verify that your signature has been added:

10. gpg --check-sigs user3@example3.com

11. You can export the signature to the public key and then send the signed copy back to the owner of the public key to boost the key's level of confidence for future users:

12. gpg --output signed-key.gpg --export --armor user3@example3.com

Send the signed key to the public key owner via email so they can import the signature to their GPG database.

Submit Your Public Key to a Key Server

You can submit your public key to a GPG server to make it available to the general public. The GnuPG configuration file ~/.gnupg/gpg.conf by default sets the key server as hkp://keys.gnupg.net and provides examples of other key servers that can be used in the file's comments. Since key servers around the globe synchronize their keys to each other it should not be necessary to change the default value set in the configuration file.

1. Find the long key ID for the public key you would like to send to the key server:

2. gpg --keyid-format long --list-keys user@example.com

You will see an output similar to the example. The long key ID is the value after the key size 4096R in the pub row. In the example the long key ID is C7277DE1A11C0F78:

```
pub   4096R/C7277DE1A11C0F78 2018-08-02 [expires: 2018-09-01]
uid                  example user <user@example.com>
sub   4096R/B838757D5C4E6643 2018-08-02 [expires: 2018-09-01]
```

3. To send your public key to the default key server use the following command and replace keyid with your public key's long key ID:

4. gpg --send-keys keyid

5. Anyone can request your public key from the key server with the following command:

6. gpg --recv-keys keyid

The public key will be added to the user's trust database using thetrustdb.gpg file.

Encrypt a Message

After you have obtained someone's public keys, you can send them encrypted messages. When you are encrypting a message to send to someone, you are using their public key to encrypt the message. Only the holder of the corresponding private key will be able to decrypt the message.

To encrypt a message:

```
gpg --output encrypted-doc.gpg --encrypt --sign --armor --recipient user3@example3.com -recipient user@example.com doc-to-encrypt.txt
```

Replace encrypted-doc.gpg with a name for the encrypted version of your document, user3@example3.com with the email associated with the public key of the encrypted message's recipient, user@example.com with your own public key's associated email and doc-to-encrypt.txt with the name of the document you will encrypt. If the document is not in the current directory, include the full path to the document.
The extension .gpg is used for encrypted/binary data and .asc or .sig is used for detached or clearsign signatures. Including the --armor flag will encrypt the message in plain text.

Decrypt a Message

A message will need to have been encrypted with your public key for you to able to decrypt it with your private key. Ensure that anyone that will be sending you an encrypted message has a copy of your public key.

To decrypt a message:

```
gpg --output decrypted-doc --decrypt doc-to-decrypt.gpg
```

Replace decrypted-doc with the name you want to assign to the decrypted message and doc-to-decrypt.gpg with the name of the encrypted document. If the document is not in the current directory, include the full path to the document.

CONCLUSION/ Outcome:

Hence we Study of Network security by
a) Set up IPSec under Linux.
b) Set up Snort and study the logs.
c) Explore the GPG tool to implement email security

Marks & Signature:

R1 (5 Marks)	R2 (5 Marks)	R3 (5 Marks)	Total (15 Marks)	Signature