Thadomal Shahani Engineering College

Bandra (W.), Mumbai- 400 050.

CERTIFICATE

Certify that ~~Mr.~~/Miss SHIPRA SUVARNA

of IT Department, Semester V with

Roll No. 133 has completed a course of the necessary

experiments in the subject Security Lab under my

supervision in the **Thadomal Shahani Engineering College**

Laboratory in the year 2023 - 2024

27/10/23

Teacher In-Charge

Head of the Department

Date 27/10/23

Principal

TSEC

CONTENTS

SR. NO.	EXPERIMENTS	PAGE NO.	DATE	TEACHERS SIGN.
1	Breaking shift cipher & mono alphabetic substitution cipher using frequency analysis		18/7/23	
2	Cryptoanalysis or decoding of poly-alphabetic cipher - playfair, vigenere		25/7/23	
3	Block cipher modes of operation using advanced encryption system (AES)		8/8/23	
4	Implementation & analysis of RSA cryptosystem & digital signature scheme using RSA		22/8/23	
5	To explore hashdeep tool in kali linux for generating, matching and auditing hash for files		24/8/23	
6	Study of network reconnaissance tools like ~~dig~~, whois etc		29/8/23	A20
7	Study of packet sniffer tools like wireshark & TCP Dump		12/9/23	27/10/23
8	Installation of NMAP using it with diff options to scan, open & ports.		28/9/23	
9	Simulate DOS attack using HPing 3			
10	Study & configure firewall using IP tables		3/10/23	
11	Installing snort, setting in intrusion detection mode		17/10/23	
12	Explore GPG tool of linux to implement email security.		17/10/23	
13	Written assignment 1			
14	Written assignment 2			
15	Presentation.			

Assignment 1

Name: Shipra Suvarna

Roll No: T23-133

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

Lab Outcome: LO1: Illustrate symmetric cryptography by implementing classical cipher.

Theory:

Shift Cipher - A shift cipher, also known as a Caesar cipher, is a type of substitution cipher where each letter in the plaintext is shifted a certain number of positions down the alphabet. It's named after Julius Caesar, who is believed to have used this simple encryption technique to protect his military messages.

The shift cipher works by selecting a key, which is an integer value representing the number of positions each letter will be shifted. The key can be any number from 1 to 25 since there are 26 letters in the English alphabet.

Let's look at an example with a key of 3:

Plaintext: HELLO Key: 3

To encrypt the message, we shift each letter in the plaintext 3 positions down the alphabet: H

-> K (shifted 3 positions to the right)

E -> H

L -> O

L -> O

O -> R

So, the encrypted cipher text would be: KHOOR

To decrypt the cipher text , you simply shift each letter in the opposite direction, using the same key. In this case, since the key is 3, we shift 3 positions to the left:

K -> H (shifted 3 positions to the left) H -> E
O -> L
O -> L

R -> O
And we get back the original plaintext: HELLO

Keep in mind that the shift cipher is not very secure by itself since there are only 25 possible keys (excluding the key 0, which does not change the plaintext). It can be easily cracked through brute force or frequency analysis. However, it serves as a basic example of a substitution cipher and can be a starting point for more complex encryption algorithms.

How and why it can be broken using brute force attack?

A brute force attack is an exhaustive trial-and-error method used to crack encryption by trying all possible combinations until the correct one is found. In the context of a shift cipher, a brute force attack involves trying all possible keys (numbers from 1 to 25) to decrypt the cipher text .

Here's why a shift cipher can be broken using a brute force attack:

1. Limited number of possible keys: Since there are only 25 possible keys for a shift cipher (excluding the key 0, which results in the same plaintext), an attacker can easily try all of them.

2. Known cipher text : In many cases, the attacker has access to the encrypted cipher text . This allows them to attempt decrypting it using different keys.

3. Frequency analysis: Although the shift cipher is a substitution cipher, it does not change the frequency of letters in the plaintext. For example, the letter 'E' is the most common letter in the English language. So, in the cipher text , the most frequent letter is likely to represent 'E'. By analyzing the frequency of letters in the cipher text , the attacker can make educated guesses about the key and narrow down the possible solutions.

4. Short messages: If the plaintext message is relatively short, the attacker has fewer possibilities to try, making the brute force attack even more feasible.

By trying all 25 possible keys and examining the decrypted output, the attacker can quickly identify the correct key and reveal the original plaintext.

Mono Alphabetic Cipher - A mono alphabetic cipher is a type of substitution cipher where each letter in the plaintext is replaced with a fixed corresponding letter in the cipher text. Unlike the shift cipher (Caesar cipher), where letters are shifted by a fixed number of positions, the mono alphabetic cipher employs a more general substitution method.

In a mono alphabetic cipher, the same substitution rule is used consistently throughout the encryption process. For example, if 'A' is substituted with 'R', every occurrence of 'A' in the plaintext will be replaced with 'R' in the cipher text .

Let's look at a simple example of a mono alphabetic cipher: Plaintext: HELLO Cipher text : RBYYQ

In this example, we are using a fixed substitution rule, where each letter in the plaintext is replaced with a corresponding letter in the cipher text :

H -> R E -> B L -> Y O -> Q

So, the original message "HELLO" is encrypted as "RBYYQ" using the mono alphabetic cipher.

Can it be broken using Brute force attack? Why?

Yes, a mono alphabetic cipher can be broken using a brute force attack, but it becomes computationally challenging as the length of the cipher text increases.

Here's why a mono alphabetic cipher can be susceptible to a brute force attack:

1. Limited key space: In a mono alphabetic cipher, the key space (number of possible keys) is limited to all possible permutations of the 26 letters in the alphabet. Since there are 26! (factorial) possible keys, the number of potential keys is enormous, but still finite.

2. Frequency analysis: Although the substitution is fixed for each letter, the frequency of letters in the language is preserved in the cipher text . For example, the letter 'E' is the most common letter in the English language. So, the most frequently occurring letter in the cipher text is likely to correspond to 'E'. By analyzing the frequency of letters in the cipher text , an attacker can make educated guesses about the key and narrow down the possible solutions.

3. Short cipher text : If the length of the cipher text is relatively short, the attacker has fewer possibilities to try, making the brute force attack more feasible.

Despite the large number of possible keys, the mono alphabetic cipher's security is compromised because the frequency analysis provides clues about the letter mapping, making it significantly easier for an attacker to try different keys and crack the encryption. With sufficient computing power and knowledge of the language of the plaintext, an attacker could

try all possible keys (through a brute force attack) and eventually find the correct key that decrypts the cipher text into meaningful text.

How it is broken using frequency analysis attack?

A frequency analysis attack is a technique used to break mono alphabetic ciphers, including the simple substitution cipher. This attack relies on the fact that certain letters or patterns occur with predictable frequencies in natural languages like English. By analysing the frequency of letters in the cipher text, an attacker can make educated guesses about the substitutions made in the mono alphabetic cipher, and ultimately, deduce the key used to encrypt the message.

Here's how a frequency analysis attack works

1. Collect cipher text: The attacker starts by obtaining the encrypted cipher text that was generated using the mono alphabetic cipher.

2. Count letter frequencies: The attacker analyses the cipher text and counts the frequency of each letter in the text.

3. Identify common letters: In English, some letters appear more frequently than others. For example, 'E', 'T', 'A', 'O', and 'I' are among the most common letters. The attacker looks for the most frequently occurring letters in the cipher text.

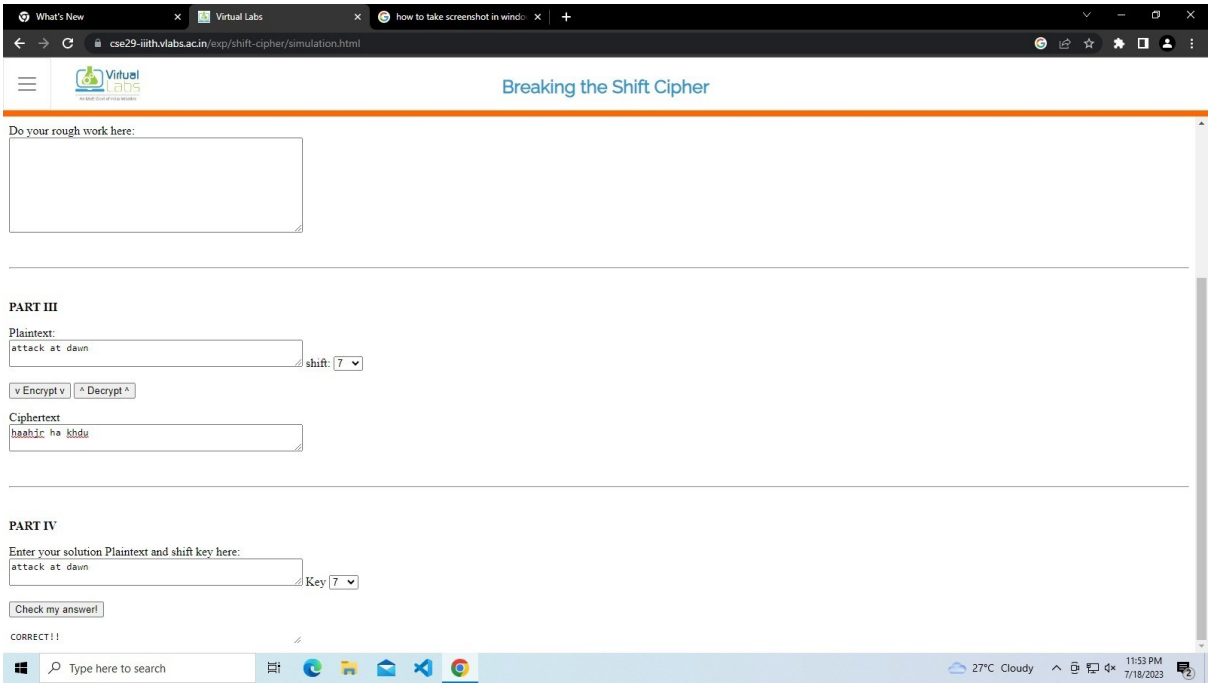
4. Make educated guesses: Based on the frequency analysis, the attacker may assume that the most frequent letter in the cipher text corresponds to 'E', the second most frequent letter corresponds to 'T', and so on. By making these educated guesses, the attacker creates partial mappings between cipher text letters and their probable plaintext counterparts.

5. Match the mapping: The attacker continues to make guesses and maps more cipher text letters to their potential plaintext counterparts based on the frequency analysis.

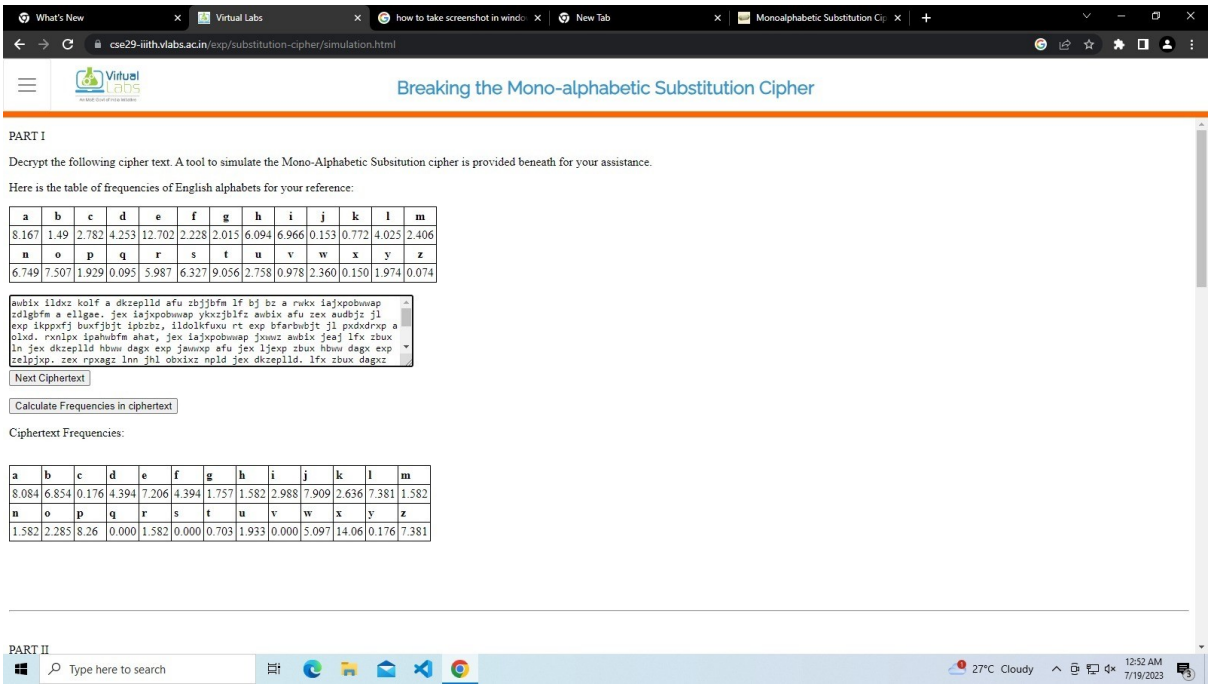
6. Trial and error: Once the attacker has some plausible letter mappings, they can attempt to decrypt the cipher text using those mappings. If the decrypted output looks meaningful, the attacker is on the right track. If not, they adjust the mappings and try again.

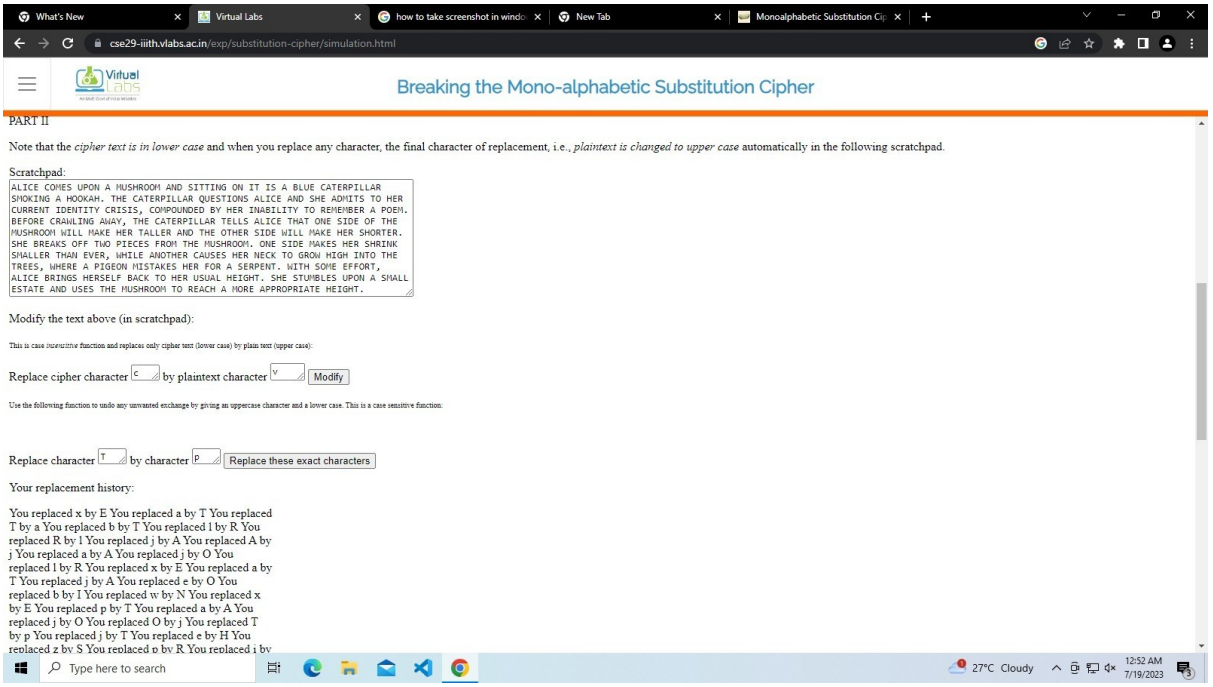
7. Iteration: The attacker iterates through this process, refining the mappings, and decrypting the cipher text until the entire message is deciphered and becomes readable.

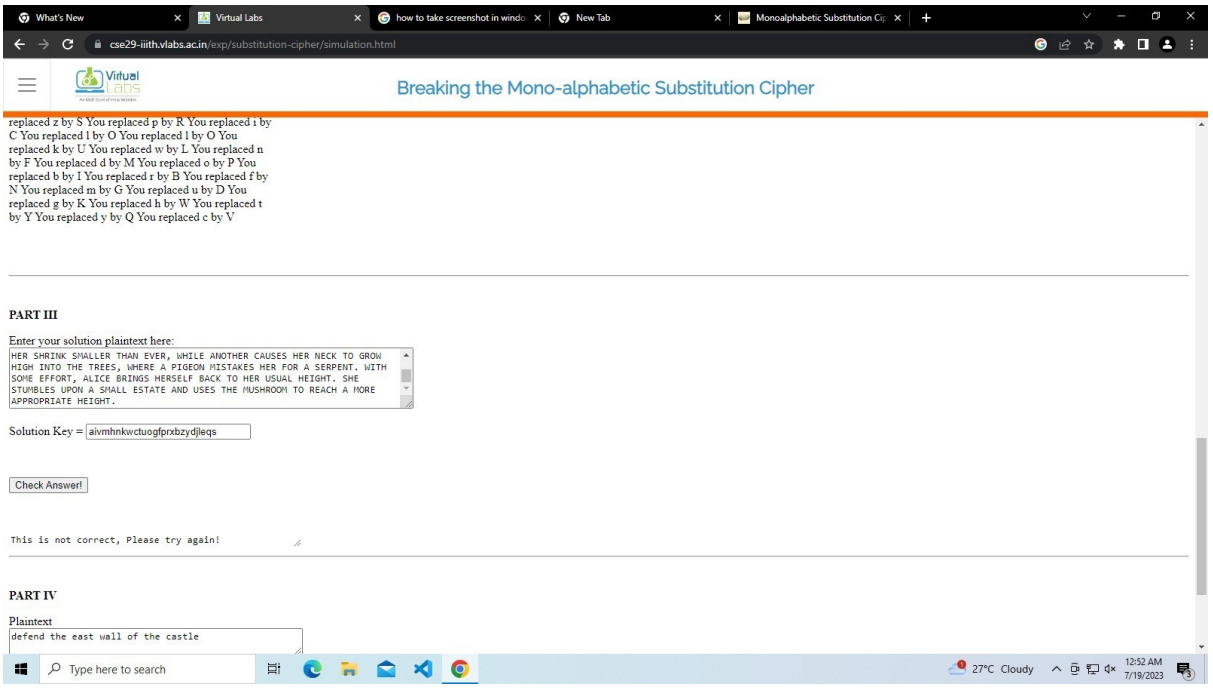
1. **Shift cipher**



2. Mono alphabetic substitution cipher







Conclusion:

In conclusion, the mono alphabetic cipher is a simple substitution cipher where each letter in the plaintext is consistently replaced with a fixed letter in the cipher text. While it can be

easily broken using frequency analysis and brute force attacks due to the limited key space and predictable letter frequencies, it serves as a basic example of substitution ciphers. To enhance security, modern encryption techniques use more complex algorithms and larger key spaces, making the encryption much more resistant to attacks.

Assignment 2

Name: Shipra Suvarna

Roll No: T23-133

Aim: Cryptanalysis or decoding of polyalphabetic ciphers: Playfair, Vigenere cipher.

Lab Outcome: LO2

Theory:

The Vigenere cipher is a method of encrypting alphabetic text using a simple form of polyalphabetic substitution. It was invented by the French diplomat Blaise de Vigenerein the 16th century. The cipher uses a keyword or phrase to encrypt the plaintext, making it more secure than the simpler Caesar cipher.

Here's how the Vigenerecipher works with an example: 1. Key Generation:

Choose a keyword or phrase. For this example, let's use the keyword "KEY" (all uppercase). Repeat the keyword to match the length of the plaintext.

Plaintext: ATTACKATDAWN

Keyword: KEYKEYKEYKEY

2. Convert the letters to numerical values:

Assign each letter a numerical value, usually starting from 0. In this example, we'll use A=0, B=1, C=2, and so on.

Plaintext (numerical): 0 19 19 0 2 10 19 3 0 22 13 Keyword

(numerical): 10 4 24 10 4 24 10 4 24 10 4

3. Encryption:

Add the numerical values of the plaintext and keyword together (mod 26) to obtain the ciphertext.

Ciphertext (numerical): (0 + 10) % 26 = 10 (K)

(19 + 4) % 26 = 23 (X) (19 + 24) % 26 = 17 (R) (0 + 10) % 26 = 10 (K) (2 + 4) % 26 = 6 (G) (10 + 24) % 26 = 8 (I) (19 + 10) % 26 = 3 (C) (3 + 4) % 26 = 7 (H)

(0 + 24) % 26 = 24 (Y) (22 + 10) % 26 = 6 (G) (13 + 4) % 26 = 17 (R)

Ciphertext: KXRKGICHYGR

So, the encrypted message using the Vigenerecipher with the keyword "KEY" is "KXRKGICHYGR".

To decrypt the ciphertext, you would reverse the process by subtracting the numerical values of the keyword from the numerical values of the ciphertext (mod 26).

The Kasiski test is a technique used to break the Vigenerecipher by exploiting the repeating patterns in the ciphertext. It was first introduced by the German military officer Friedrich Kasiski in the mid-19th century.

The main idea behind the Kasiski test is that when the same keyword is used multiple times to encrypt different parts of the plaintext, it creates repeating patterns in the ciphertext. By identifying these patterns, it becomes possible to estimate the length of the keyword, which is a crucial step in breaking the Vigenerecipher.

Here's how the Kasiski test works:

1. Identify Repeating Patterns:

Look for repeated sequences of letters in the ciphertext that are at least 3 characters long. Longer repetitions are more helpful, but shorter ones can still provide some clues.

2. Calculate Distances:

For each repeating sequence, calculate the distance between the occurrences. The distance is the number of letters between the repetitions in the ciphertext. Record all the distances.

3. Find Common Factors:

Examine the list of distances and identify common factors among them. The idea is that the keyword length may be a multiple of these common factors.

4. Estimate the Keyword Length:

The most likely keyword length is the highest common factor found in the distances. For example, if the distances are 6, 12, 18, and 24, the highest common factor is 6, which suggests that the keyword length is likely 6 characters.

5. Perform Frequency Analysis:

Now that you have an estimate of the keyword length, you can divide the ciphertext into groups of letters based on the assumed keyword length. Each group corresponds to the letters encrypted with the same letter of the keyword. Perform frequency analysis on each group to determine the most likely letter for each group.

6. Determine the Keyword:

With the frequency analysis results, you can now reconstruct the keyword and use it to decrypt the ciphertext using the Vigenerecipher decryption process.

It's important to note that the Kasiski test provides an estimate of the keyword length, and it may not be 100% accurate. In cases where the keyword is very short or the ciphertext is too short to reveal meaningful patterns, the Kasiski test may not be as effective. However, when the ciphertext is long enough and the keyword is relatively long, the Kasiski test can be a valuable tool in breaking the Vigenerecipher.

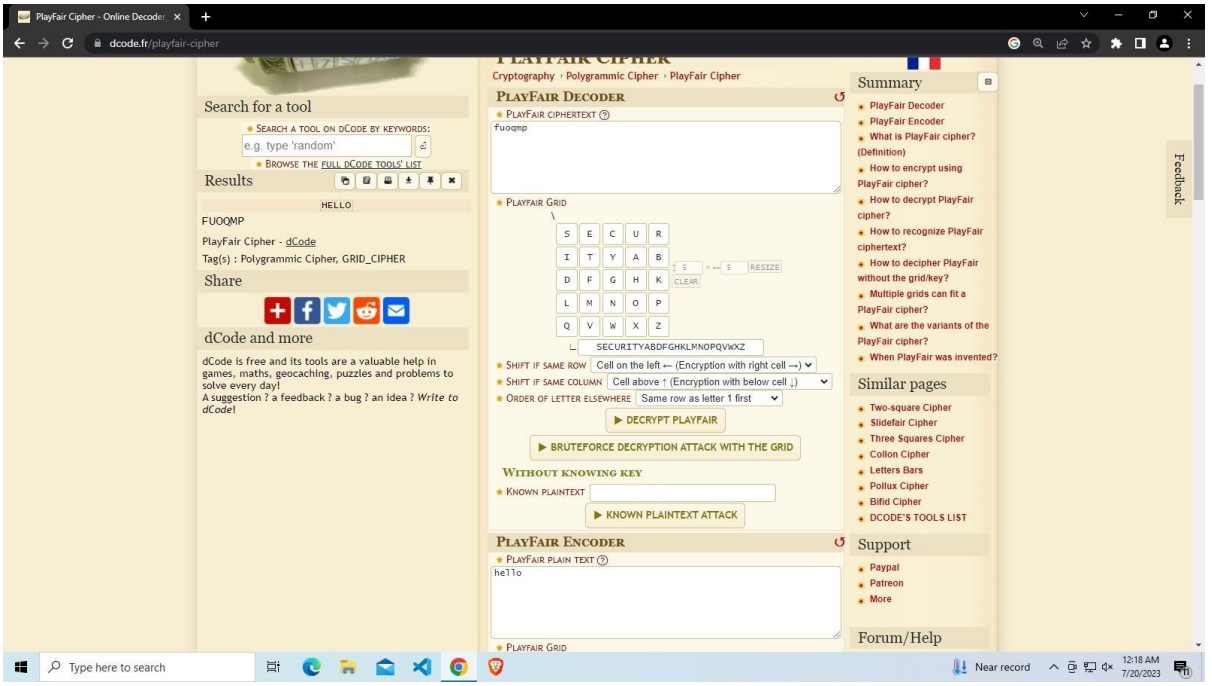
The Playfair cipher is a manual symmetric encryption technique that was invented by Sir Charles Wheatstone in 1854 but later popularized by Baron Playfair during the 19th century. It was used primarily for protecting sensitive information before the advent of modern cryptographic methods. The Playfair cipher encrypts digraphs (pairs of letters) rather than individual letters, which makes it more secure than simple substitution ciphers.

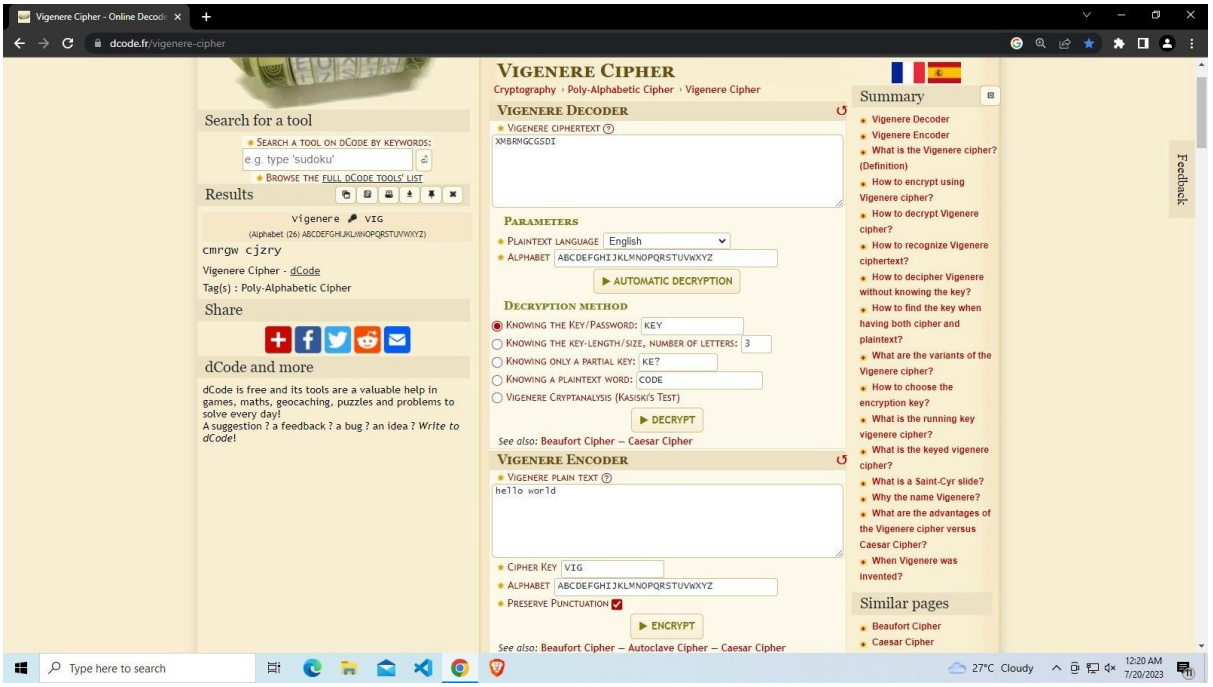
Here's how the Playfair cipher works: 1. Key Generation:

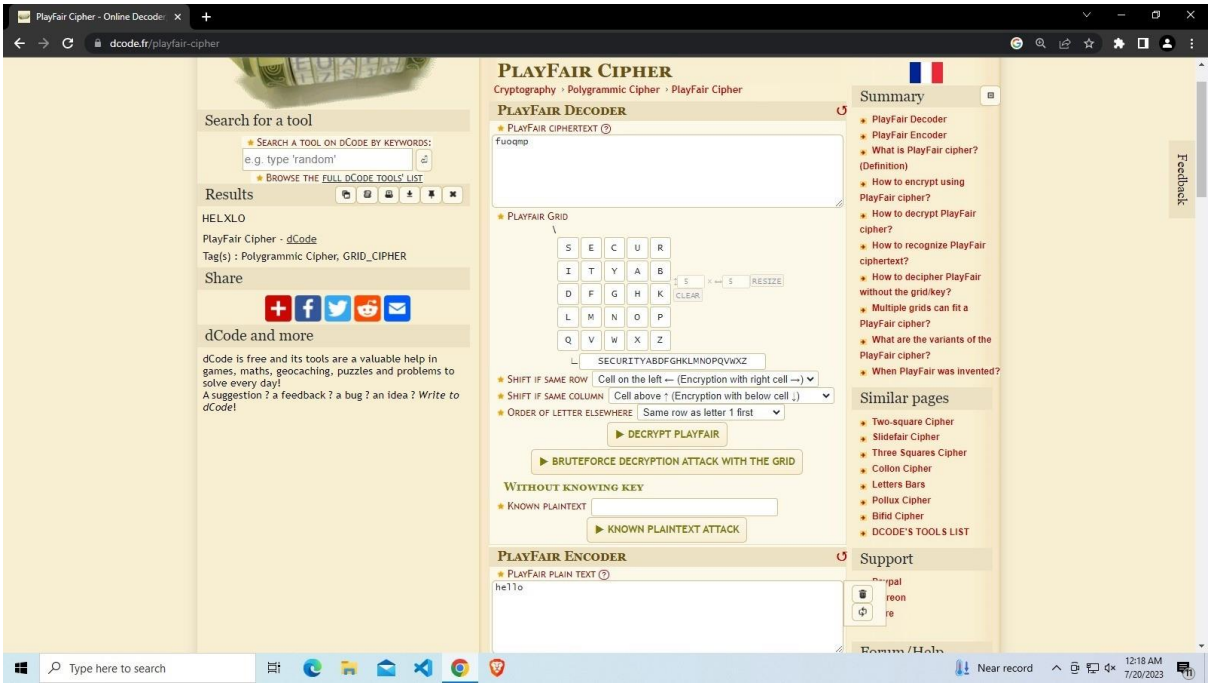
Choose a keyword or phrase to create a 5x5 matrix (key square). The key square is used to determine the encryption and decryption rules. The matrix is filled with unique letters of the keyword, excluding duplicates and the letters "J" (to avoid confusion with "I"). If the keyword does not have 25 unique letters, fill the rest of the matrix with the remaining letters of the alphabet in order, omitting "J."

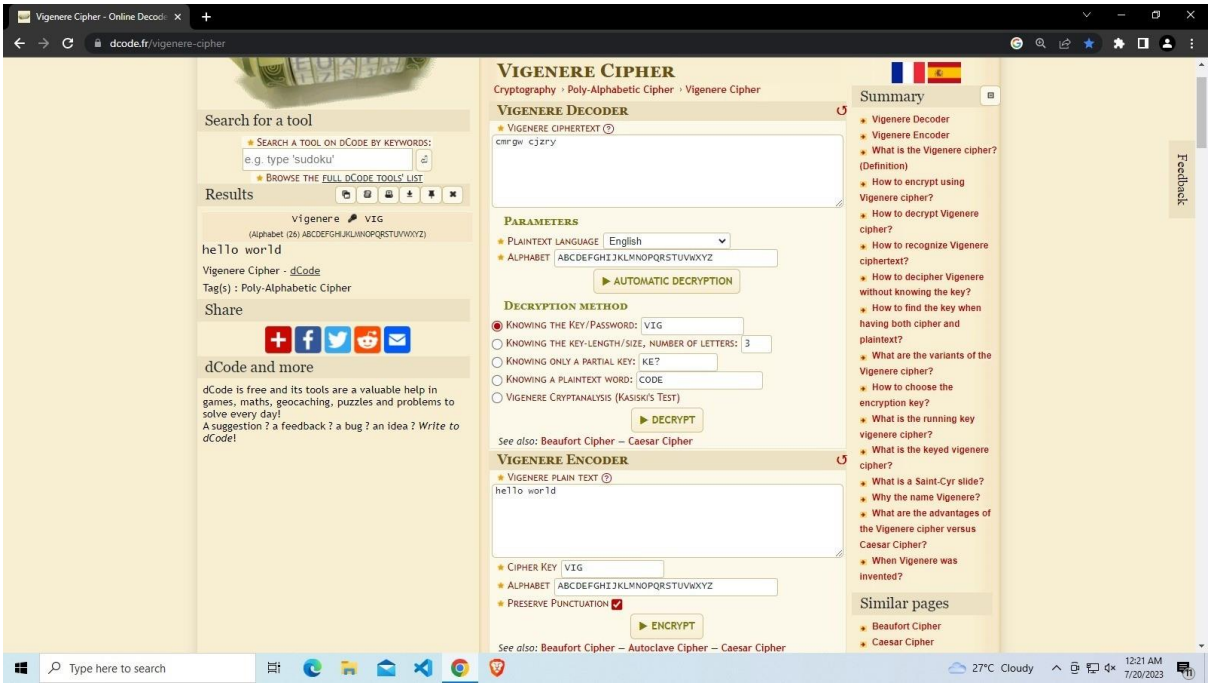
For example, using the keyword "KEYWORD": K| E| Y | W| O
R | D| A | B | C F
|G


PlayFair Cipher - Online Decoder
dcode.fr/playfair-cipher
Search for a tool
Search a tool on dCode by keywords:
e.g. type 'random'
Browse the full dCode tools' list
Results
HELXLO
PlayFair Cipher - dCode
Tag(s) : Polygrammic Cipher, GRID_CIPHER
Share
dCode and more
dCode is free and its tools are a valuable help in games, maths, geocaching, puzzles and problems to solve every day!
A suggestion ? a feedback ? a bug ? an idea ? Write to dCode!
PlayFair Cipher
Cryptography › Polygrammic Cipher › PlayFair Cipher
PlayFair Decoder
PlayFair ciphertext
fuoqmp
PlayFair Grid
S E C U R
I T Y A B
D F G H K
L M N O P
Q V W X Z
SECURITYABDFGHKLMNOPQVWXZ
Shift if same row
Cell on the left ← (Encryption with right cell →)
Shift if same column
Cell above ↑ (Encryption with below cell ↓)
Order of letter elsewhere
Same row as letter 1 first
▶ DECRYPT PLAYFAIR
▶ BRUTEFORCE DECRYPTION ATTACK WITH THE GRID
Without knowing key
Known plaintext
▶ KNOWN PLAINTEXT ATTACK
PlayFair Encoder
PlayFair plain text
he11o
Summary
PlayFair Decoder
PlayFair Encoder
What is PlayFair cipher? (Definition)
How to encrypt using PlayFair cipher?
How to decrypt PlayFair cipher?
How to recognize PlayFair ciphertext?
How to decipher PlayFair without the grid/key?
Multiple grids can fit a PlayFair cipher?
What are the variants of the PlayFair cipher?
When PlayFair was invented?
Similar pages
Two-square Cipher
Slidefair Cipher
Three Squares Cipher
Collon Cipher
Letters Bars
Pollux Cipher
Bifid Cipher
DCODE'S TOOLS LIST
Support
Paypal
Feedback
Type here to search
Near record
12:18 AM
7/20/2023


PlayFair Cipher - Online Decoder
dcode.fr/playfair-cipher
Search for a tool
Search a tool on dCode by keywords:
e.g. type 'random'
Browse the full dCode tools' list
Results
HELLO
FUOQMP
PlayFair Cipher - dCode
Tag(s) : Polygrammic Cipher, GRID_CIPHER
Share
dCode and more
dCode is free and its tools are a valuable help in games, maths, geocaching, puzzles and problems to solve every day!
A suggestion ? a feedback ? a bug ? an idea ? Write to dCode!
Cryptography › Polygrammic Cipher › PlayFair Cipher
PlayFair Decoder
PlayFair ciphertext
fuoqmp
PlayFair Grid
S E C U R
I T Y A B
D F G H K
L M N O P
Q V W X Z
SECURITYABDFGHKLMNOPQVWXZ
Shift if same row
Cell on the left ← (Encryption with right cell →)
Shift if same column
Cell above ↑ (Encryption with below cell ↓)
Order of letter elsewhere
Same row as letter 1 first
▶ DECRYPT PLAYFAIR
▶ BRUTEFORCE DECRYPTION ATTACK WITH THE GRID
Without knowing key
Known plaintext
▶ KNOWN PLAINTEXT ATTACK
PlayFair Encoder
PlayFair plain text
he11o
PlayFair Grid
Summary
PlayFair Decoder
PlayFair Encoder
What is PlayFair cipher? (Definition)
How to encrypt using PlayFair cipher?
How to decrypt PlayFair cipher?
How to recognize PlayFair ciphertext?
How to decipher PlayFair without the grid/key?
Multiple grids can fit a PlayFair cipher?
What are the variants of the PlayFair cipher?
When PlayFair was invented?
Similar pages
Two-square Cipher
Slidefair Cipher
Three Squares Cipher
Collon Cipher
Letters Bars
Pollux Cipher
Bifid Cipher
DCODE'S TOOLS LIST
Support
Paypal
Patreon
More
Forum/Help
Feedback
Type here to search
Near record
12:18 AM
7/20/2023


Vigenere Cipher - Online Decod
dcode.fr/vigenere-cipher
VIGENERE CIPHER
Cryptography › Poly-Alphabetic Cipher › Vigenere Cipher
Search for a tool
SEARCH A TOOL ON DCODE BY KEYWORDS:
e.g. type 'sudoku'
BROWSE THE FULL DCODE TOOLS' LIST
Results
vigenere VIG
(Alphabet (26) ABCDEFGHIJKLMNOPQRSTUVWXYZ)
cmrgw cjzry
Vigenere Cipher - dCode
Tag(s) : Poly-Alphabetic Cipher
Share
dCode and more
dCode is free and its tools are a valuable help in games, maths, geocaching, puzzles and problems to solve every day!
A suggestion ? a feedback ? a bug ? an idea ? Write to dCode!
VIGENERE DECODER
VIGENERE CIPHERTEXT
XMBRMGCGSDI
PARAMETERS
PLAINTEXT LANGUAGE English
ALPHABET ABCDEFGHIJKLMNOPQRSTUVWXYZ
AUTOMATIC DECRYPTION
DECRYPTION METHOD
KNOWING THE KEY/PASSWORD: KEY
KNOWING THE KEY-LENGTH/SIZE, NUMBER OF LETTERS: 3
KNOWING ONLY A PARTIAL KEY: KE?
KNOWING A PLAINTEXT WORD: CODE
VIGENERE CRYPTANALYSIS (KASISKI'S TEST)
DECRYPT
See also: Beaufort Cipher – Caesar Cipher
VIGENERE ENCODER
VIGENERE PLAIN TEXT
hello world
CIPHER KEY VIG
ALPHABET ABCDEFGHIJKLMNOPQRSTUVWXYZ
PRESERVE PUNCTUATION
ENCRYPT
See also: Beaufort Cipher – Autoclave Cipher – Caesar Cipher
Summary
Vigenere Decoder
Vigenere Encoder
What is the Vigenere cipher? (Definition)
How to encrypt using Vigenere cipher?
How to decrypt Vigenere cipher?
How to recognize Vigenere ciphertext?
How to decipher Vigenere without knowing the key?
How to find the key when having both cipher and plaintext?
What are the variants of the Vigenere cipher?
How to choose the encryption key?
What is the running key vigenere cipher?
What is the keyed vigenere cipher?
What is a Saint-Cyr slide?
Why the name Vigenere?
What are the advantages of the Vigenere cipher versus Caesar Cipher?
When Vigenere was invented?
Similar pages
Beaufort Cipher
Caesar Cipher
Feedback
Type here to search
27°C Cloudy
12:20 AM
7/20/2023


PlayFair Cipher - Online Decode
dcode.fr/playfair-cipher
PLAYFAIR CIPHER
Cryptography › Polygrammic Cipher › PlayFair Cipher
Search for a tool
SEARCH A TOOL ON DCODE BY KEYWORDS:
e.g. type 'random'
BROWSE THE FULL DCODE TOOLS' LIST
Results
HELXLO
PlayFair Cipher - dCode
Tag(s) : Polygrammic Cipher, GRID_CIPHER
Share
dCode and more
dCode is free and its tools are a valuable help in games, maths, geocaching, puzzles and problems to solve every day!
A suggestion ? a feedback ? a bug ? an idea ? Write to dCode!
PLAYFAIR DECODER
PLAYFAIR CIPHERTEXT
fuoqmp
PLAYFAIR GRID
S E C U R
I T Y A B
D F G H K
L M N O P
Q V W X Z
RESIZE
CLEAR
SECURITYABDFGHKLMNOPQVWXZ
SHIFT IF SAME ROW Cell on the left ← (Encryption with right cell →)
SHIFT IF SAME COLUMN Cell above ↑ (Encryption with below cell ↓)
ORDER OF LETTER ELSEWHERE Same row as letter 1 first
DECRYPT PLAYFAIR
BRUTEFORCE DECRYPTION ATTACK WITH THE GRID
WITHOUT KNOWING KEY
KNOWN PLAINTEXT
KNOWN PLAINTEXT ATTACK
PLAYFAIR ENCODER
PLAYFAIR PLAIN TEXT
hello
Summary
PlayFair Decoder
PlayFair Encoder
What is PlayFair cipher? (Definition)
How to encrypt using PlayFair cipher?
How to decrypt PlayFair cipher?
How to recognize PlayFair ciphertext?
How to decipher PlayFair without the grid/key?
Multiple grids can fit a PlayFair cipher?
What are the variants of the PlayFair cipher?
When PlayFair was invented?
Similar pages
Two-square Cipher
Slidefair Cipher
Three Squares Cipher
Collon Cipher
Letters Bars
Pollux Cipher
Bifid Cipher
DCODE'S TOOLS LIST
Support
Feedback
Type here to search
Near record
12:18 AM
7/20/2023



Conclusion:

Through this experiment we were able to understand the concept of playfair and vigenere cipher and were able to understand its working through solving the example.

Assignment 3

Name: Shipra Suvarna

Roll No: T23-133

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

Theory:-

Steps of RSA key generation.

The computational steps for key generation are

- Generate two different primes including p and q.
- Compute the modulus $n = p \times q$
- Compute the totient $\phi(n) = (p - 1) \times (q - 1)$
- Select for public exponent an integer e such that $1 < e < \phi(n)$ and $\gcd(\phi(n), e) = 1$.
- Compute for the private exponent a value for d such that $d = e{-}1 \bmod \phi(n)$
- Public Key = [e, n]
- Private Key = [d, n]
- A digital signature is a cryptographic technique used to ensure the authenticity, integrity, and non-repudiation of digital documents, messages, or transactions. It verifies the source and integrity of electronic data, confirming that it remains unchanged after being signed.

Here's an explanation of how a digital signature works:

The signing process involves the use of a private key to generate a unique digital signature for a document or message. This private key is a concealed cryptographic key held by the signer.

A hash function is applied to the content before creating the signature. This generates a fixed-size "digest" (hash value) that exclusively represents the original content.

Using the private key, the signer encrypts the hash value, resulting in the digital signature. This signature is distinct to both the content and the private key employed.

The digital signature is then added to the original content, producing a signed message or document.

To verify the digital signature, the recipient employs the corresponding public key, which is associated with the signer's private key. This public key is publicly shared and is used to decrypt the digital signature.

Through hash comparison, the recipient applies the same hash function to the received content and decrypts the digital signature using the public key. If the decrypted signature aligns with the computed hash value, the content is considered untampered and authenticated.

Digital signatures offer several important advantages:

- Authenticity: A valid digital signature confirms the identity of the signer, ensuring that the content originates from the expected source.

- Integrity: The hash value guarantees that the content remains unaltered since the signature was generated.

- Non-Repudiation: The unique connection between the signature and the signer's private key prevents the signer from denying their involvement in signing the content.

-Tamper Detection: Any modification to the original content leads to digital signature verification failure.

- Security: Digital signatures rely on robust cryptographic algorithms, making it difficult to forge or impersonate.

Digital signatures are commonly used in various contexts, including electronic documents, email communication, software distribution, and online transactions. Their purpose is to establish trust and security in the digital realm by providing assurance of origin and content integrity.

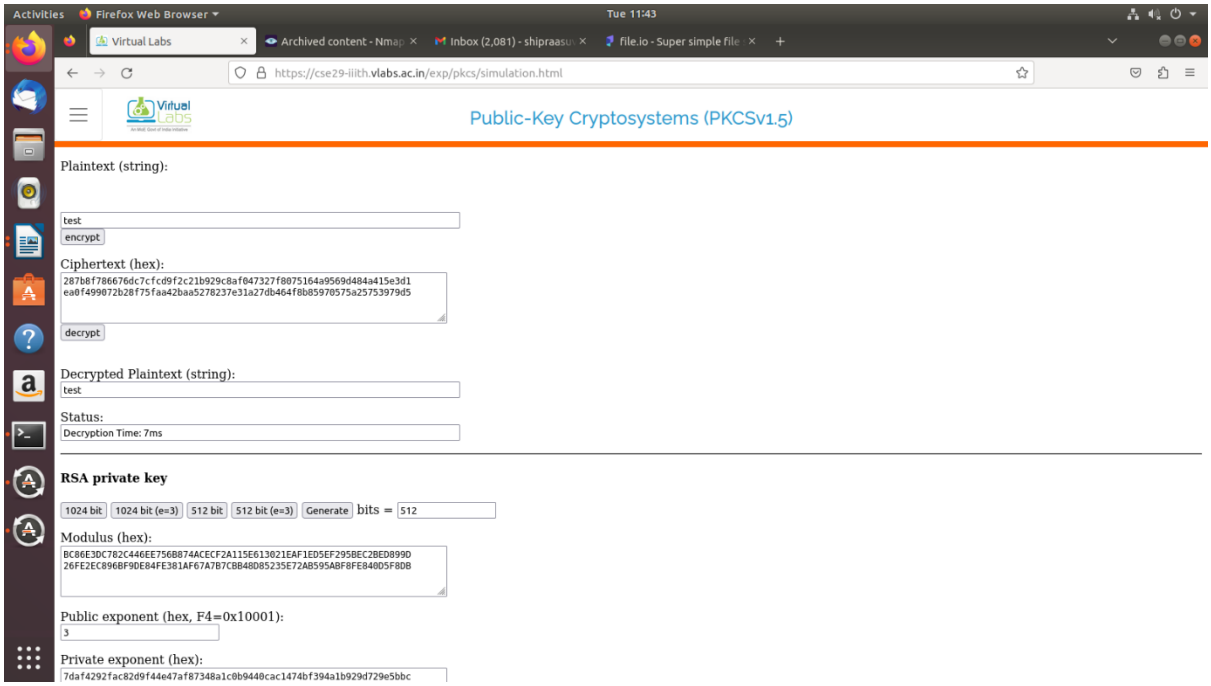
Steps of Digital signature generation and verification process.

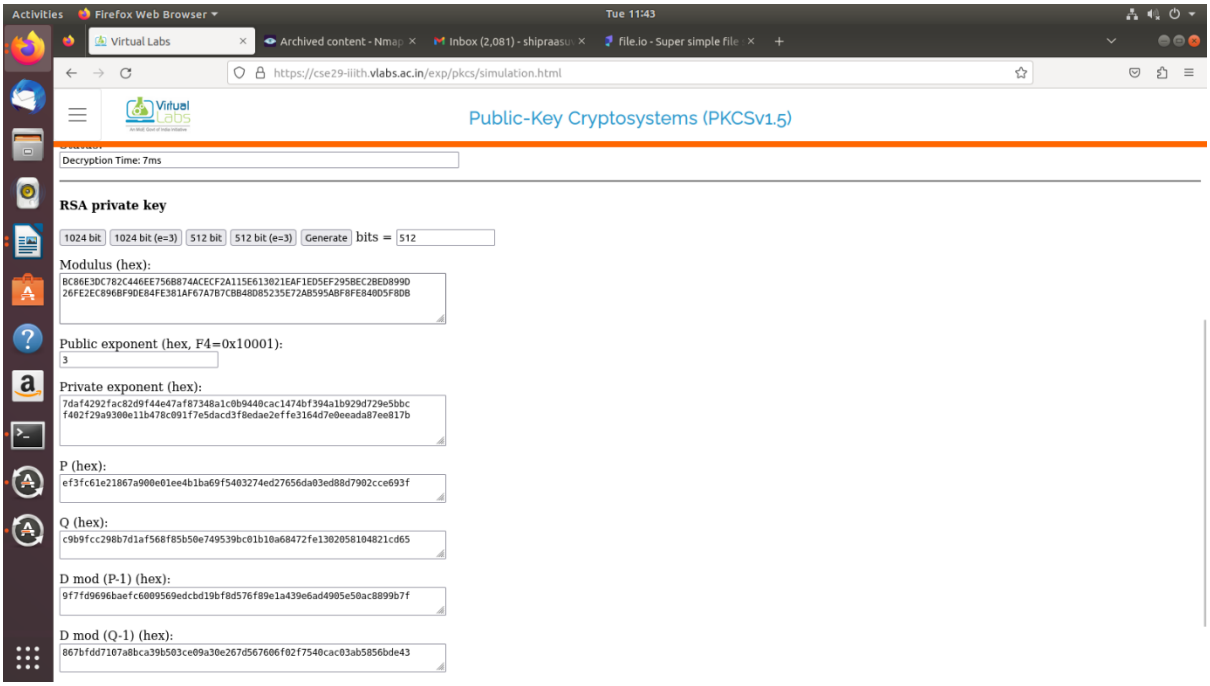
You send a document to Person B with both the Public and Private key. Remember that the verifier needs the Public key to verify the signature and also assurances that the private key is actually owned by the originator of the document.

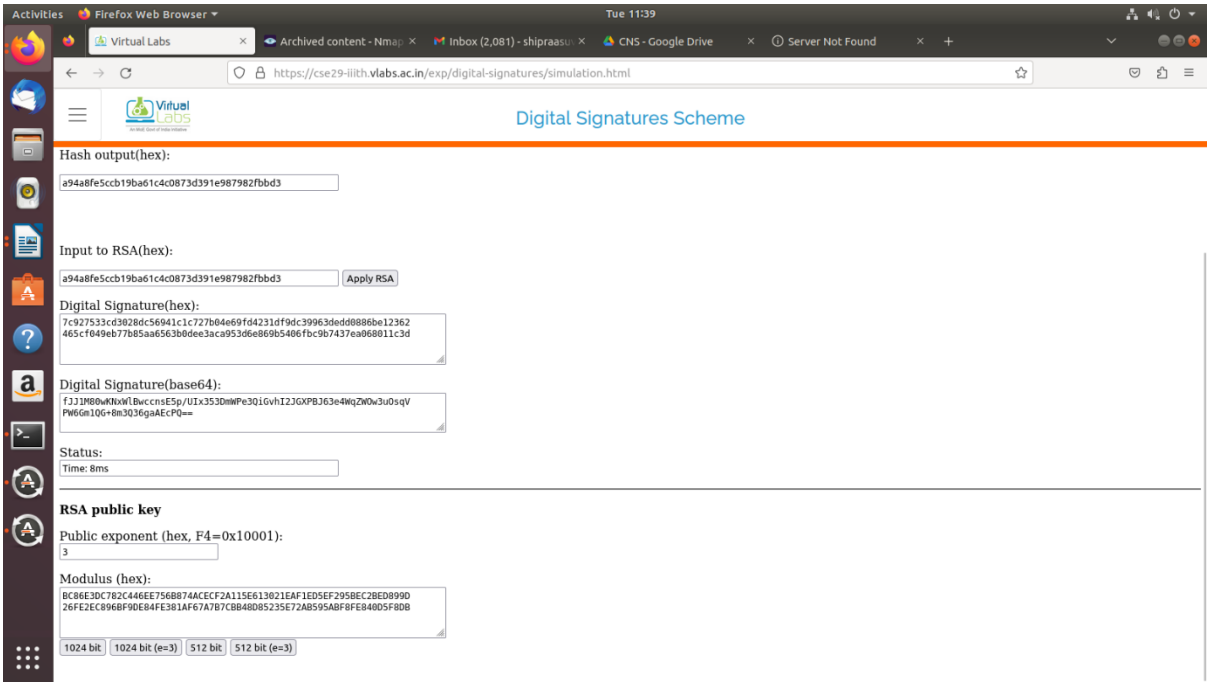
The next step is to verify the public key. The verifier can use the Certifying Authority to ensure the validity and the public key. The CA will also help the verifier authenticate the identity of the sender, ensuring that they are who they say they are.

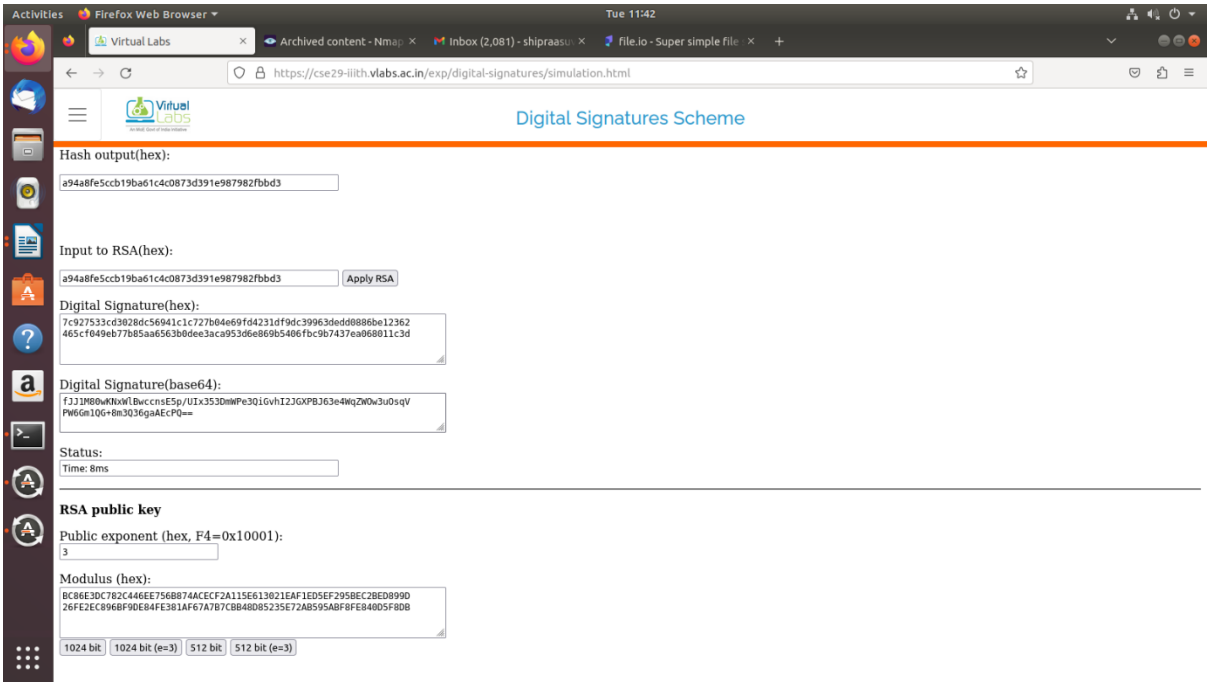
If the authenticity of the public key is confirmed, you can then enter the secret private key to decrypt the document and the document is signed. If the private key is incorrect the signature of the document cannot be verified. This is why it is essential to verify the identity of the sender with the Certificate Authority.

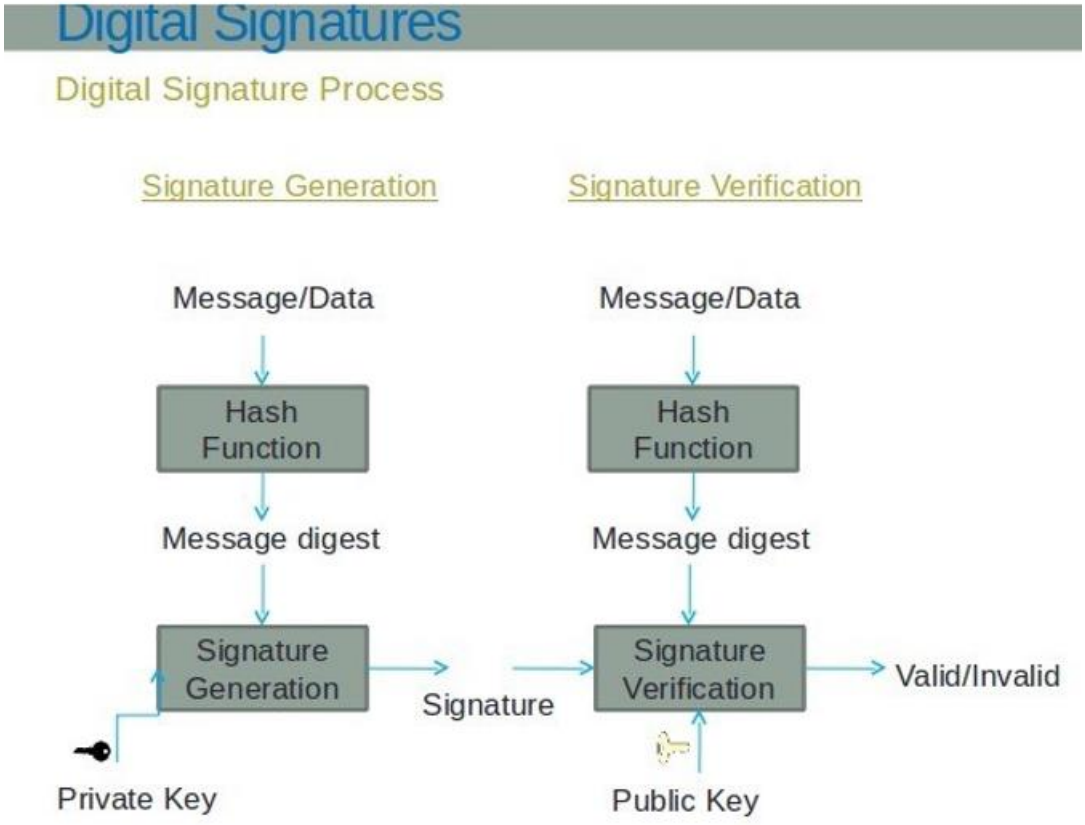
Output Screenshots:






Activities
Firefox Web Browser
Tue 11:39
Virtual Labs
Archived content - Nmap
Inbox (2,081) - shipraasu
CNS - Google Drive
Server Not Found
https://cse29-iiith.vlabs.ac.in/exp/digital-signatures/simulation.html
Digital Signatures Scheme
Hash output(hex):
a94a8fe5ccb19ba61c4c0873d391e987982fbbd3
Input to RSA(hex):
a94a8fe5ccb19ba61c4c0873d391e987982fbbd3
Apply RSA
Digital Signature(hex):
7c927533cd3028dc56941c1c727b04e69fd4231df9dc39963dedd0886be12362
465cf049eb77b85aa6563b0dee3aca953d6e869b5406fbc9b7437ea068011c3d
Digital Signature(base64):
fJJ1M80wKNxWlBwccnsE5p/UIx353DmWPe3QiGvhI2JGXPBJ63e4WqZWOw3uOsqV
PW6Gm1QG+8m3Q36gaAEcPQ==
Status:
Time: 8ms
RSA public key
Public exponent (hex, F4=0x10001):
3
Modulus (hex):
BC86E3DC782C446EE756B874ACECF2A115E613021EAF1ED5EF295BEC2BED899D
26FE2EC896BF9DE84FE381AF67A7B7CBB48D85235E72AB595ABF8FE840D5F8DB
1024 bit
1024 bit (e=3)
512 bit
512 bit (e=3)


Activities
Firefox Web Browser
Tue 11:42
Virtual Labs
Archived content - Nmap
Inbox (2,081) - shipraasu
file.io - Super simple file
https://cse29-iiith.vlabs.ac.in/exp/digital-signatures/simulation.html
Digital Signatures Scheme
Hash output(hex):
a94a8fe5ccb19ba61c4c0873d391e987982fbbd3
Input to RSA(hex):
a94a8fe5ccb19ba61c4c0873d391e987982fbbd3
Apply RSA
Digital Signature(hex):
7c927533cd3028dc56941c1c727b04e69fd4231df9dc39963dedd0886be12362
465cf049eb77b85aa6563b0dee3aca953d6e869b5406fbc9b7437ea068011c3d
Digital Signature(base64):
fJJ1M80wKNxWlBwccnsE5p/UIx353DmWPe3QiGvhI2JGXPBJ63e4WqZWOw3uOsqV
PW6Gm1QG+8m3Q36gaAEcPQ==
Status:
Time: 8ms
RSA public key
Public exponent (hex, F4=0x10001):
3
Modulus (hex):
BC86E3DC782C446EE756B874ACECF2A115E613021EAF1ED5EF295BEC2BED899D
26FE2EC896BF9DE84FE381AF67A7B7CBB48D85235E72AB595ABF8FE840D5F8DB
1024 bit
1024 bit (e=3)
512 bit
512 bit (e=3)



LO Mapped: LO2

Conclusion: Demonstrated key management, distribution and user authentication.

With this we have learned the usage, features and advantages of RSA cryptosystem and Digital signature scheme using RSA.

Assignment 4

Name: Shipra Suvarna
Roll No: T23-133

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

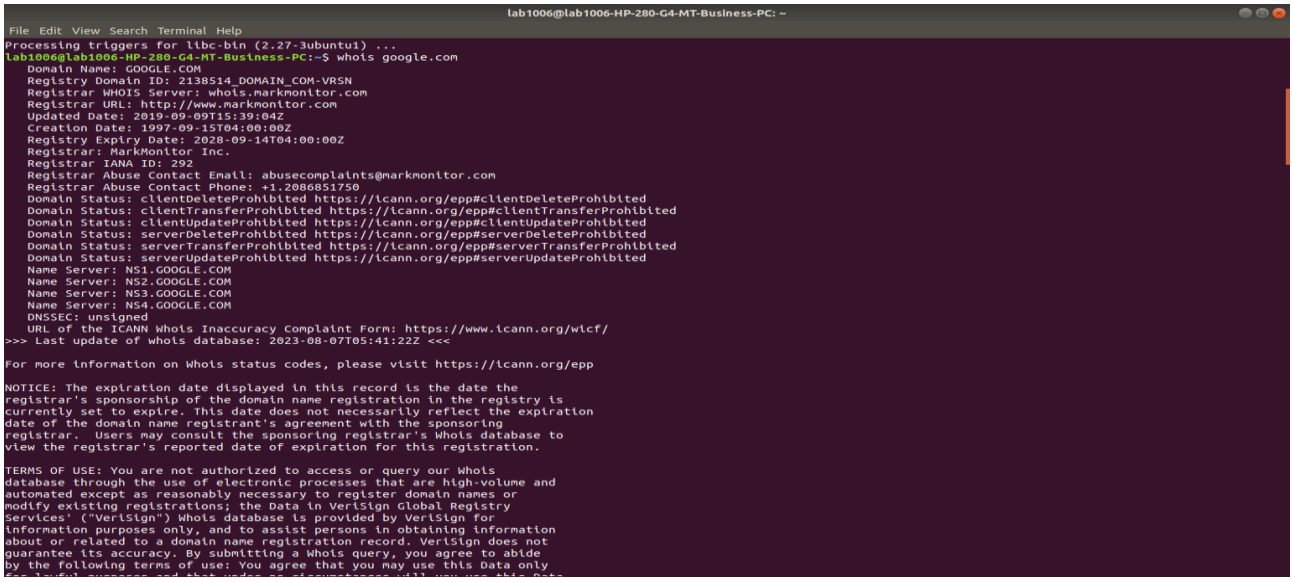
Lab Outcome: LO3

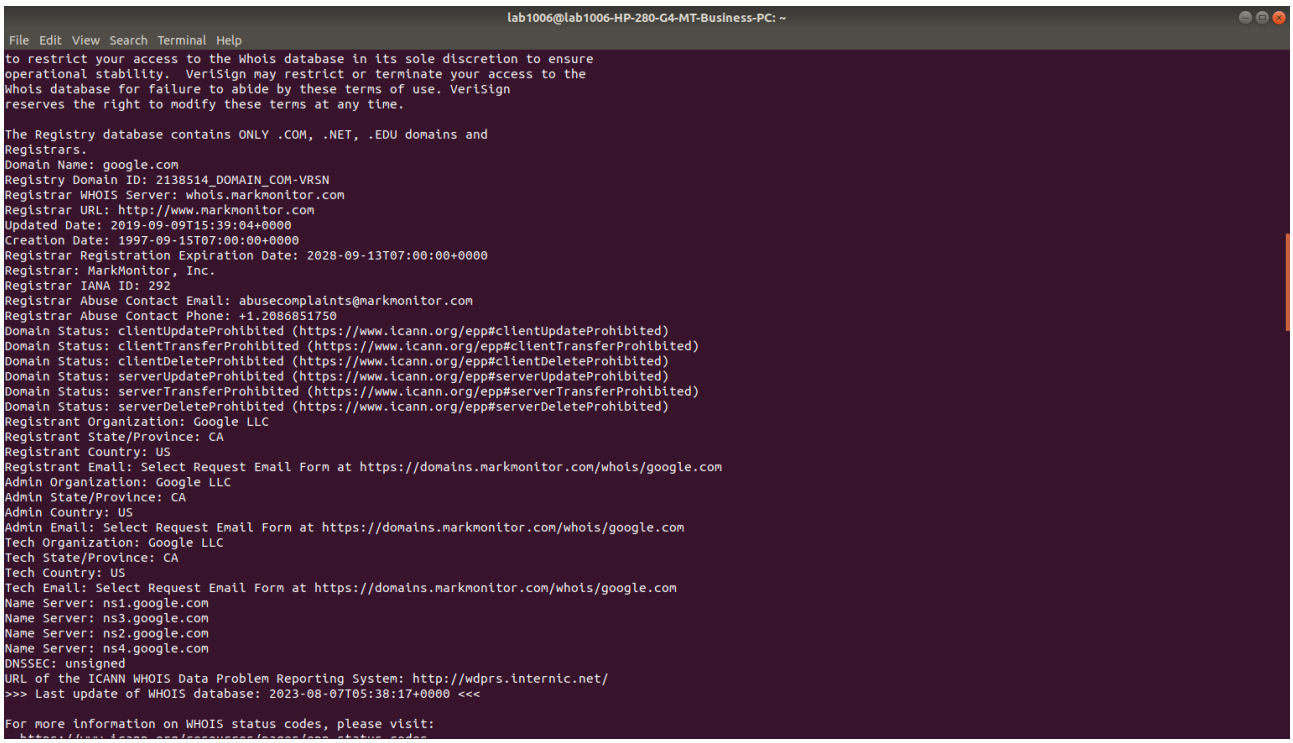
Theory:
WHOIS:
The whois command displays information about a website's record. You may get all the information about a website regarding its registration and owner's information.

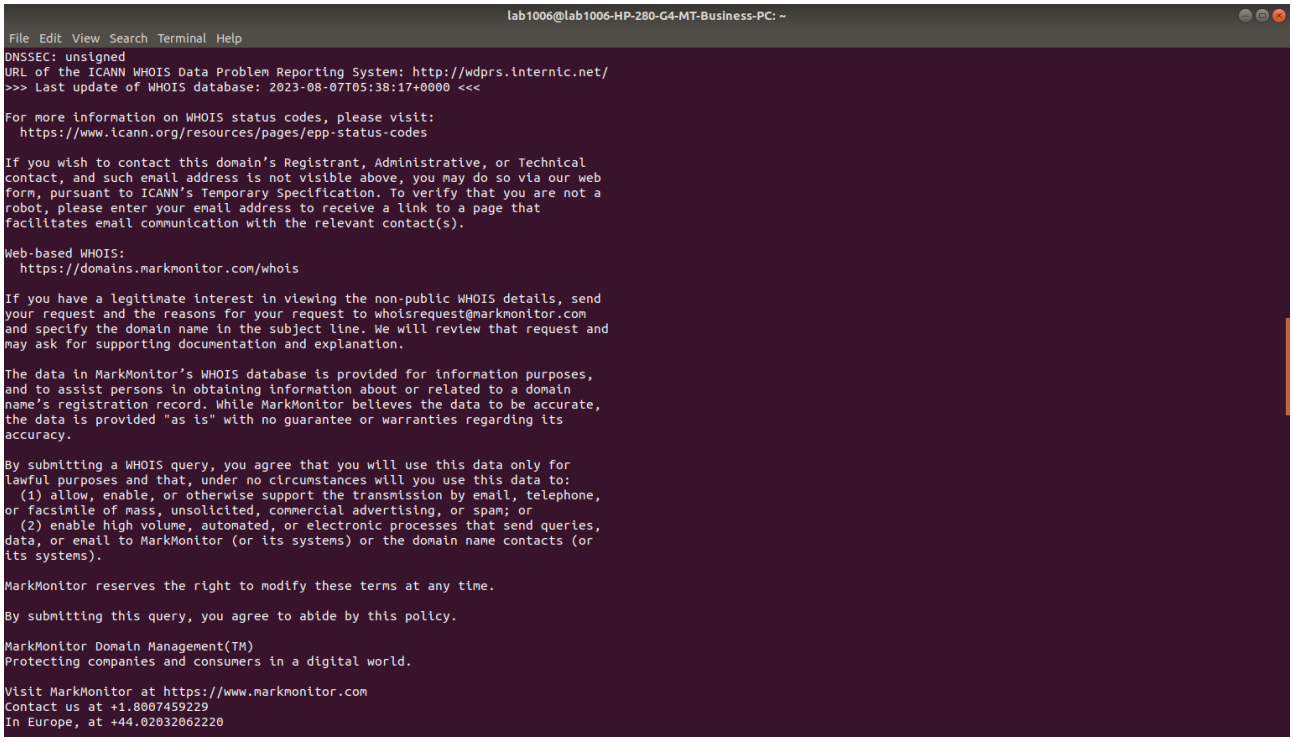
Syntax:

whois <websiteName>

Example:







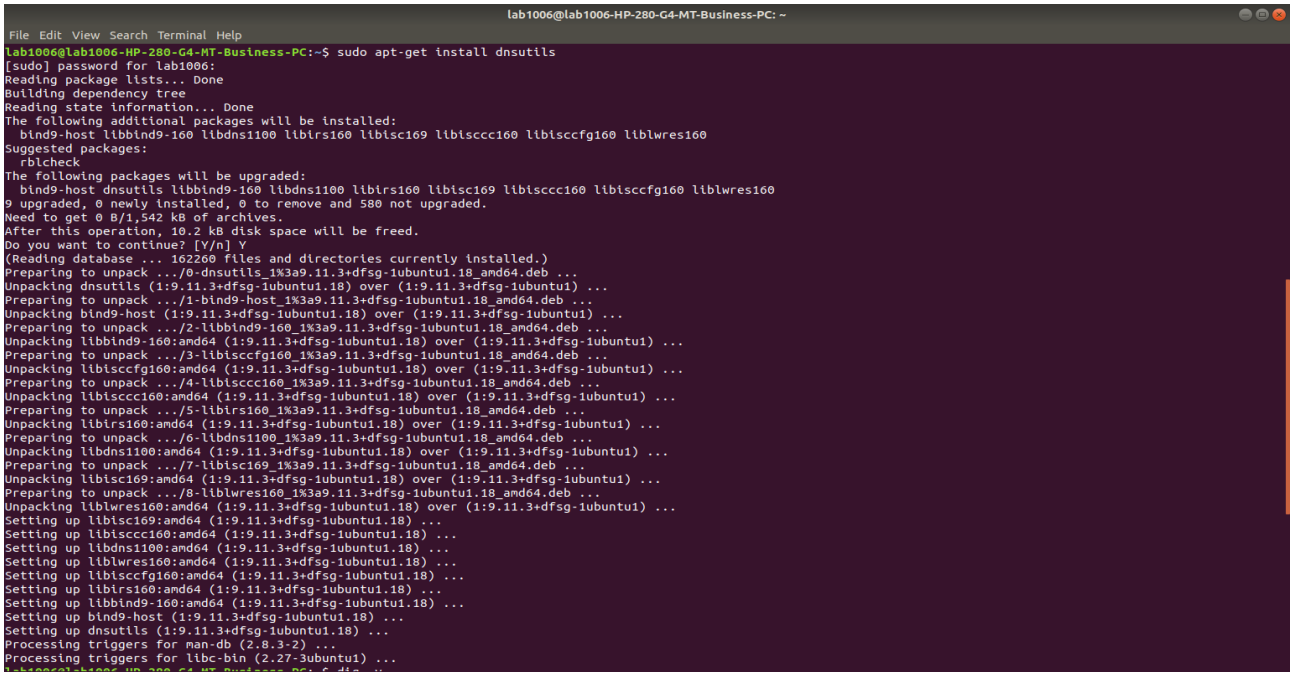
dig:

Linux dig command stands for **Domain Information Groper**. This command is used for tasks related to DNS lookup to query DNS name servers. It mainly deals with troubleshooting DNS related problems. It is a flexible utility for examining the DNS (Domain Name Servers). It is used to perform the DNS lookups and returns the queried answers from the name server. Usually, it is used by most DNS administrators to troubleshoot the DNS problems. It is a straightforward tool and provides a clear output. It is more functional than other lookups tools.

Syntax:

The general syntax of the dig command is as follows:

dig @server name type



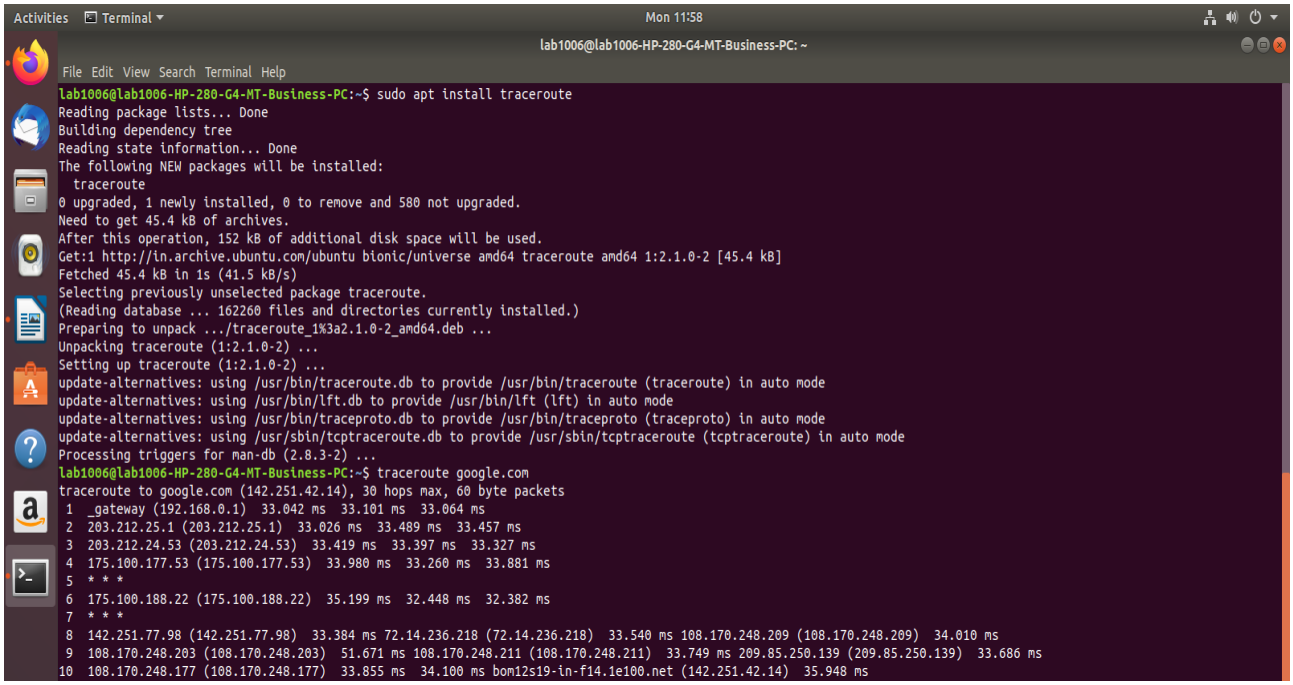

```
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ dig -v
DiG 9.11.3-1ubuntu1.18-Ubuntu
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ dig google.com

; <<>> DiG 9.11.3-1ubuntu1.18-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 7578
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             28      IN      A       142.251.42.14

;; Query time: 4 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Mon Aug 07 11:49:26 IST 2023
;; MSG SIZE  rcvd: 55

lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ dig @120.120.40.80 google.com

; <<>> DiG 9.11.3-1ubuntu1.18-Ubuntu <<>> @120.120.40.80 google.com
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$
```

Traceroute:

Linux traceroute command is a network troubleshooting utility that helps us determine the number of hops and packets traveling path required to reach a destination. It is used to display how the data transmitted from a local machine to a remote machine. Loading a web page is one of the common examples of the traceroute. A web page loading transfers data through a network and routers. The traceroute can display the routes, IP addresses, and hostnames of routers over a network. It can be useful for diagnosing network issues.

Syntax:

Traceroute [OPTION...] HOST



nslookup:

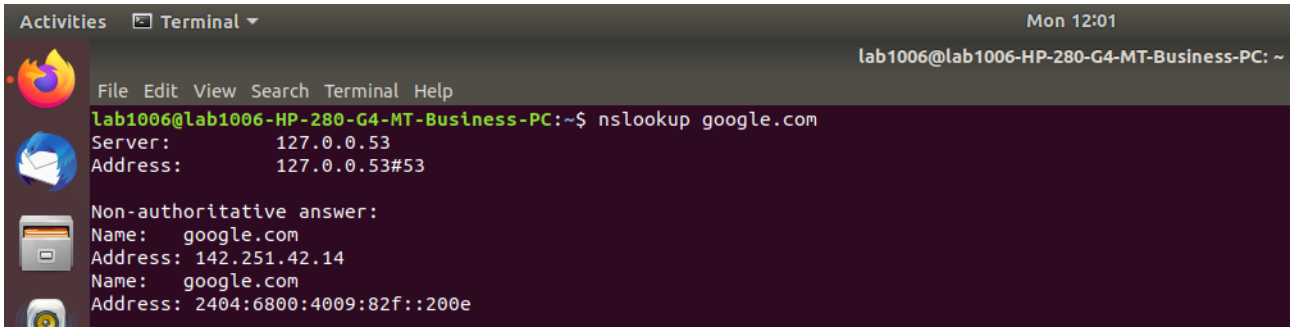
This command is also used to find DNS related query.

Syntax:

nslookup <domainName>

Example:

nslookup google.com



Nikto:

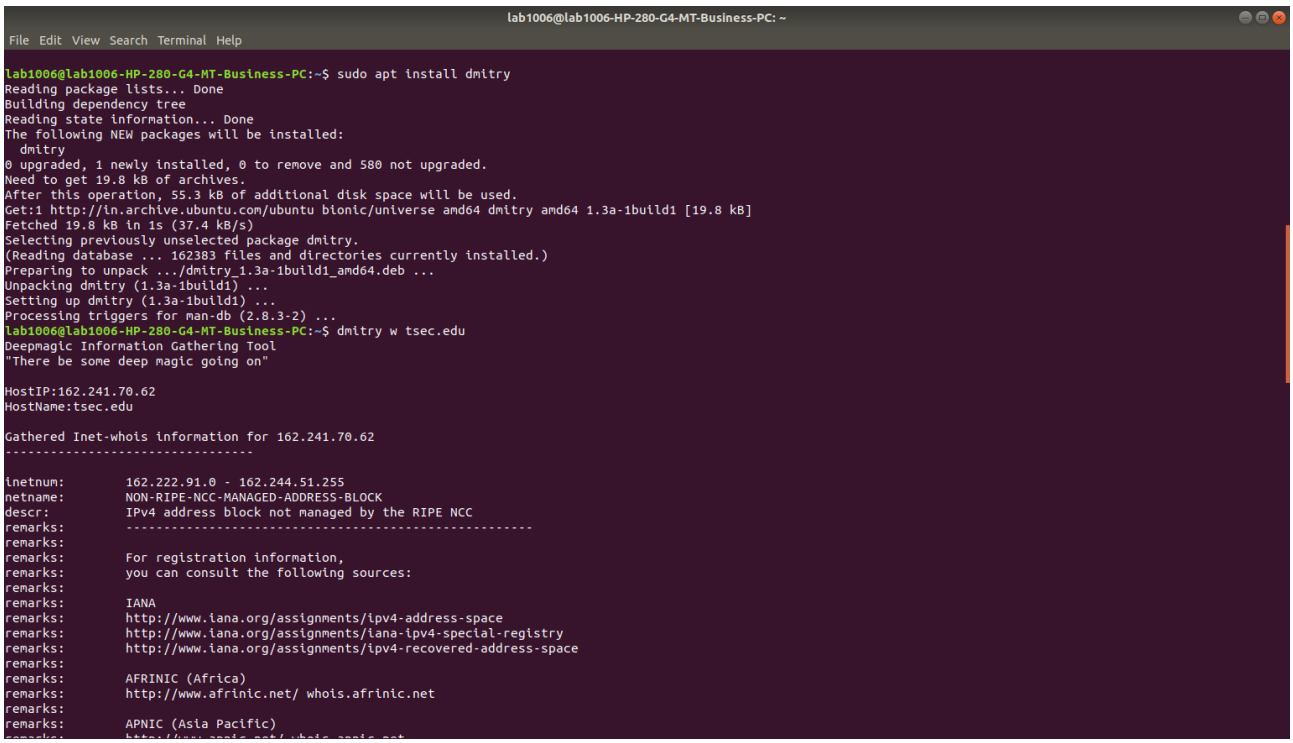
It is an open-source web server scanner which performs comprehensive tests against web servers for multiple items. You can use Nikto with any web servers like Apache, Nginx, IHS, OHS, Litespeed, and so on. Nikto can check for server configuration items such as the presence of multiple index files, HTTP server options, and will attempt to identify installed web servers and software. Items and plugins scanned by Nikto are frequently updated and can be automatically updated.

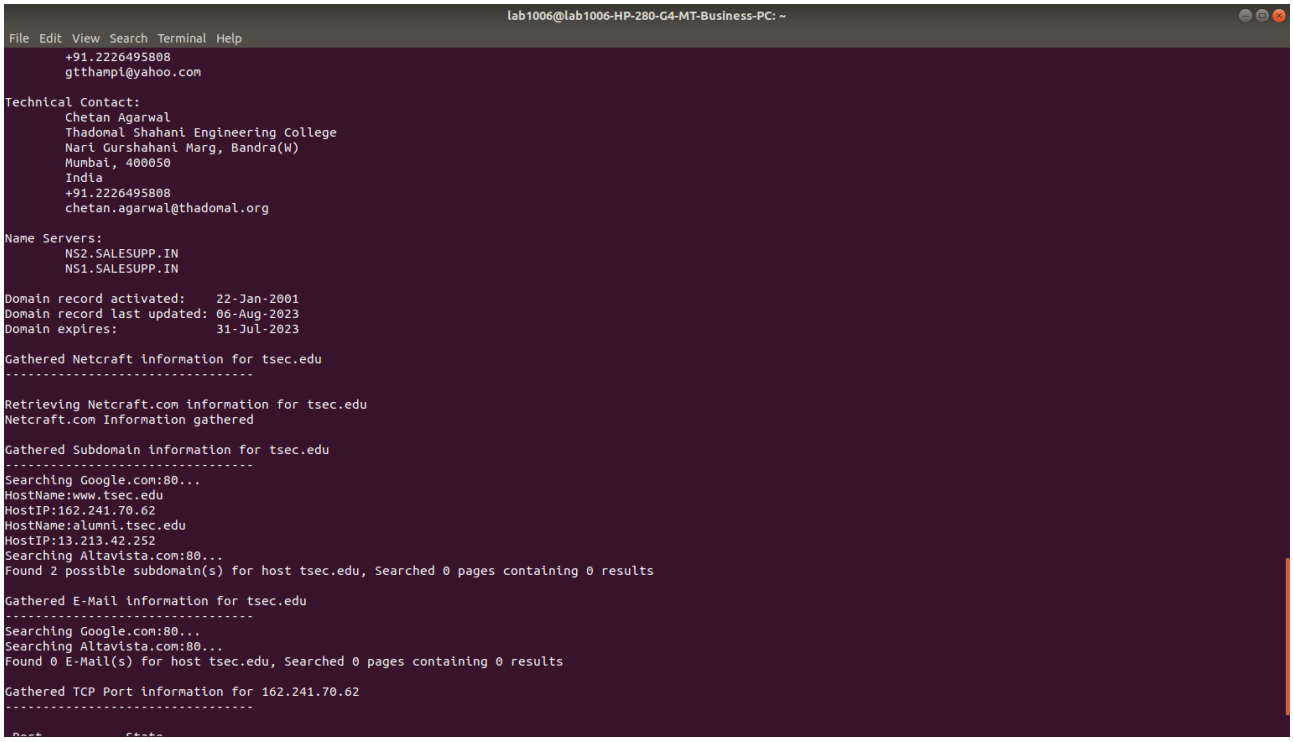
```
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ sudo apt install nikto
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libwhisker2-perl
The following NEW packages will be installed:
  libwhisker2-perl nikto
0 upgraded, 2 newly installed, 0 to remove and 580 not upgraded.
Need to get 365 kB of archives.
After this operation, 2,273 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 libwhisker2-perl all 2.5-1 [119 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu bionic/multiverse amd64 nikto all 1:2.1.5-2 [246 kB]
Fetched 365 kB in 2s (180 kB/s)
Selecting previously unselected package libwhisker2-perl.
(Reading database ... 162280 files and directories currently installed.)
Preparing to unpack .../libwhisker2-perl_2.5-1_all.deb ...
Unpacking libwhisker2-perl (2.5-1) ...
Selecting previously unselected package nikto.
Preparing to unpack .../nikto_1%3a2.1.5-2_all.deb ...
Unpacking nikto (1:2.1.5-2) ...
Setting up libwhisker2-perl (2.5-1) ...
Setting up nikto (1:2.1.5-2) ...
Processing triggers for man-db (2.8.3-2) ...
```

```
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ nikto -h google.com
- Nikto v2.1.5
---------------------------------------------------------------------------
+ Target IP:          142.251.42.14
+ Target Hostname:    google.com
+ Target Port:        80
+ Start Time:         2023-08-07 12:08:46 (GMT5.5)
---------------------------------------------------------------------------
+ Server: gws
+ Uncommon header 'content-security-policy-report-only' found, with contents: object-src 'none';base-uri 'self';script-src 'nonce-9DijV5m1u5mbqEKu0
c' 'report-sample' 'unsafe-eval' 'unsafe-inline' https: http:;report-uri https://csp.withgoogle.com/csp/gws/other-hp
+ Uncommon header 'x-xss-protection' found, with contents: 0
+ Uncommon header 'x-frame-options' found, with contents: SAMEORIGIN
+ Root page / redirects to: http://www.google.com/
+ Uncommon header 'referrer-policy' found, with contents: no-referrer
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Server banner has changed from 'gws' to 'sffe' which may suggest a WAF, load balancer or proxy is in place
+ Uncommon header 'cross-origin-resource-policy' found, with contents: cross-origin
```

Dmitry:

It stands for **DeepMagic Information Gathering Tool.** Dmitry is a **free** and **open-source** tool that is available on **GitHub.** We used this tool for information gathering. Dmitry is a **command-line** tool. With the help of the Dmitry tool, we can gather information about the target, which we can then use for **social engineering attacks**. It can be used to collect a variety of useful information.





Conclusion: Explored the different network reconnaissance tools to gather information about networks.

In conclusion, the study of network reconnaissance tools has revealed their vital role in gathering valuable information about networks and domain registrars. Each tool serves a specific purpose, contributing to a comprehensive understanding of the network infrastructure and its associated components.

Lab Assignment

Name: Shipra Suvarna

Roll No: T23-133

Aim: Block cipher modes of operation using Advanced Encryption Standard (AES)

Theory:

AES (Advanced Encryption Standard) is a widely used symmetric encryption algorithm that operates on blocks of data. AES modes of operation determine how the algorithm processes and encrypts multiple blocks of data.

Electronic Codebook (ECB):

Electronic Codebook (ECB) is one of the simplest AES modes of operation. In ECB mode, the plaintext is divided into blocks, typically 128 bits in size, and each block is encrypted independently using the same encryption key. This means that identical plaintext blocks will produce identical ciphertext blocks. While it's straightforward to implement, ECB is not recommended for secure encryption of large amounts of data because patterns in the plaintext can still be visible in the ciphertext. Attackers can exploit this predictability to infer information about the plaintext.

- ECB is the simplest AES mode of operation.
- In ECB mode, each block of plaintext is encrypted independently with the same key.
- This means that identical plaintext blocks will produce identical ciphertext blocks.
- It is not recommended for secure encryption of large amounts of data because patterns in the plaintext can still be visible in the ciphertext.

Cipher Block Chaining (CBC):

Cipher Block Chaining (CBC) is a more secure AES mode compared to ECB. In CBC mode, each plaintext block is XORed with the ciphertext of the previous block before encryption. This chaining of blocks ensures that even if the plaintext contains repeated blocks, the corresponding ciphertext blocks will differ. To start the process, an Initialization Vector (IV) is used as the previous ciphertext block for the first block of plaintext. CBC mode is suitable for secure data encryption and helps conceal patterns in the data. However, it's essential to use a unique IV for each message to maintain security.

- CBC is a more secure mode than ECB.
- In CBC mode, each plaintext block is XORed with the ciphertext of the previous block before encryption.
- The first block is XORed with an Initialization Vector (IV).
- This chaining of blocks ensures that identical plaintext blocks don't produce the same ciphertext, adding an element of security.
- CBC requires an IV, and it's important to use a different IV for each message.

Counter (CTR):

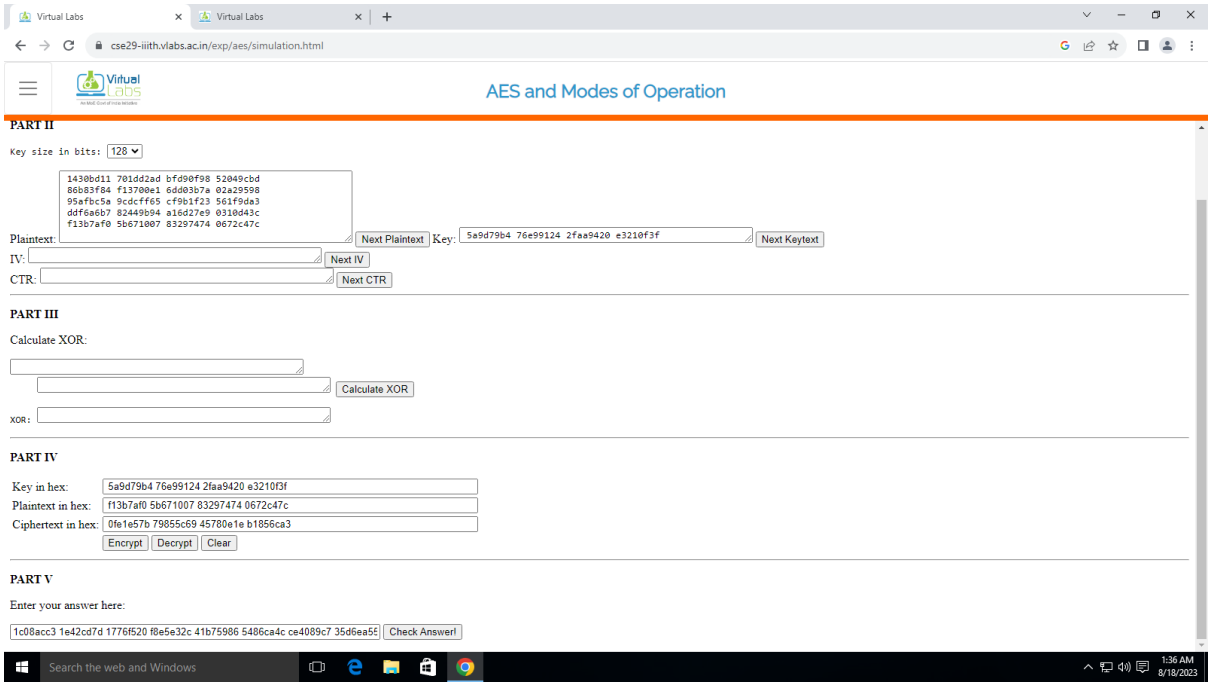
Counter (CTR) mode transforms AES into a stream cipher. Instead of encrypting plaintext blocks directly, CTR mode encrypts a counter value using the encryption key to produce a pseudorandom stream of bits, often referred to as the keystream. This keystream is then XORed with the plaintext to generate the ciphertext. CTR mode is efficient and allows for parallelization, making it suitable for scenarios where random access to data is required. It also doesn't require padding, making it suitable for variable-length plaintext. One advantage of CTR mode is that it doesn't require re-encrypting the key for each block.

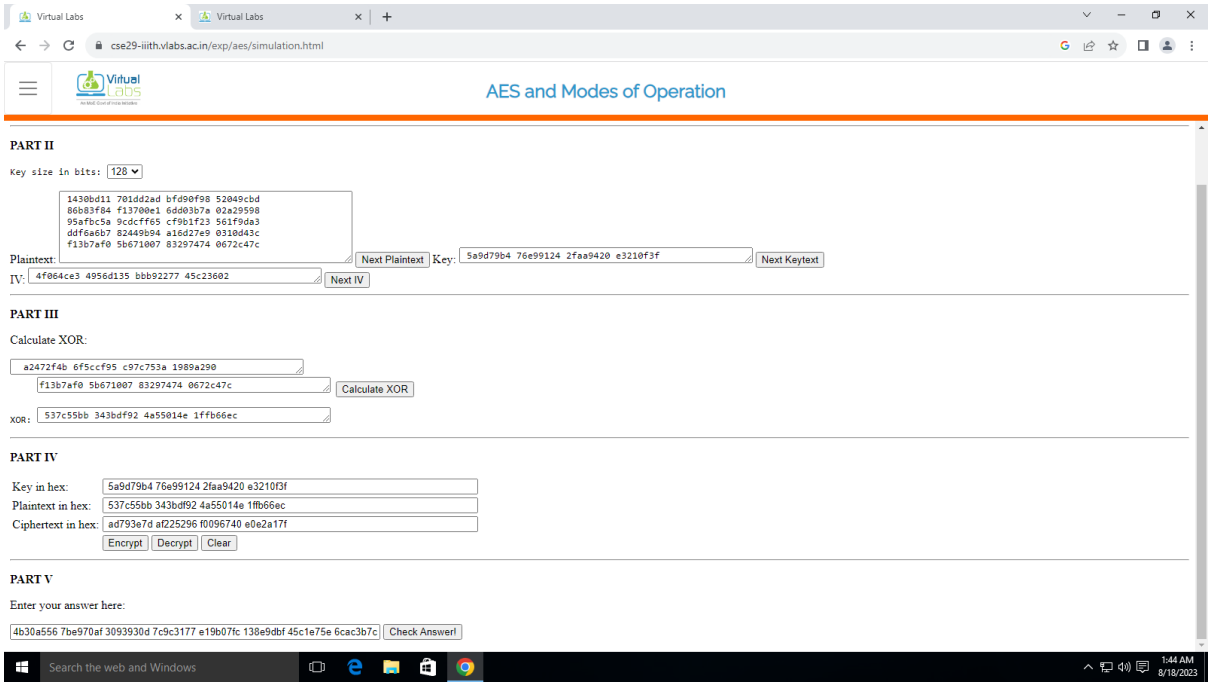
- CTR mode turns AES into a stream cipher.
- A counter value is encrypted with the key to produce a stream of pseudorandom bits.
- These bits are then XORed with the plaintext to produce ciphertext.
- CTR mode allows for parallelization and is efficient for random access to data.
- It does not require padding, making it suitable for variable-length plaintext.

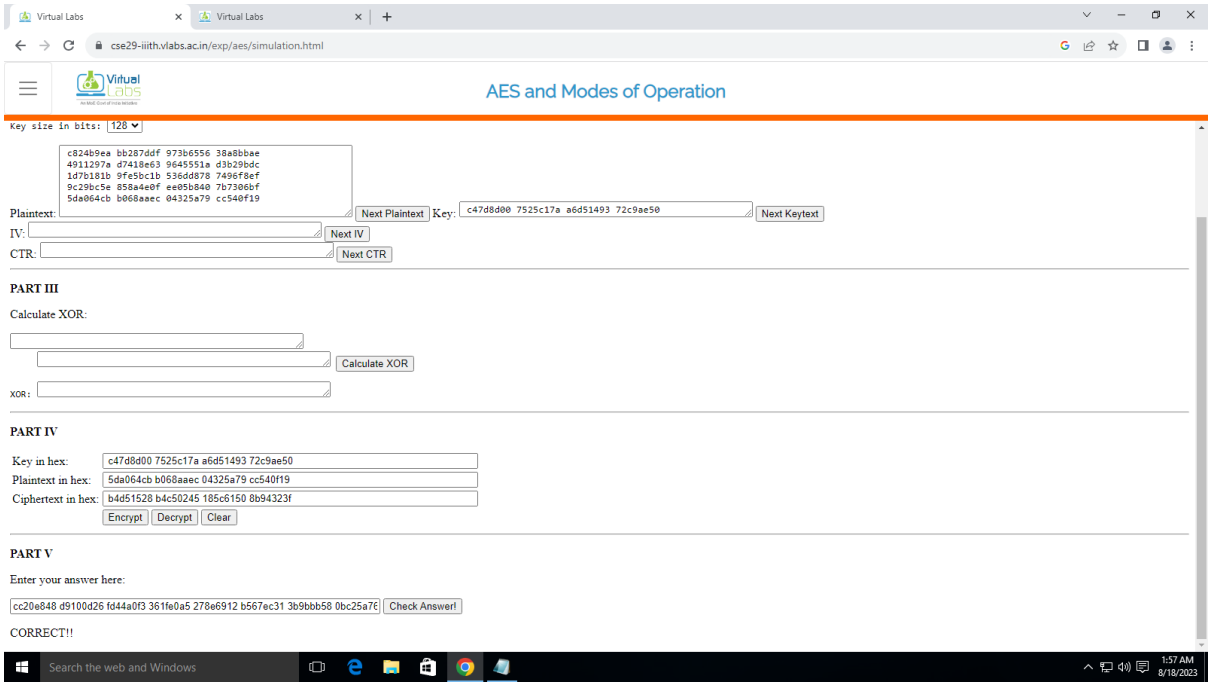
Output Feedback (OFB):

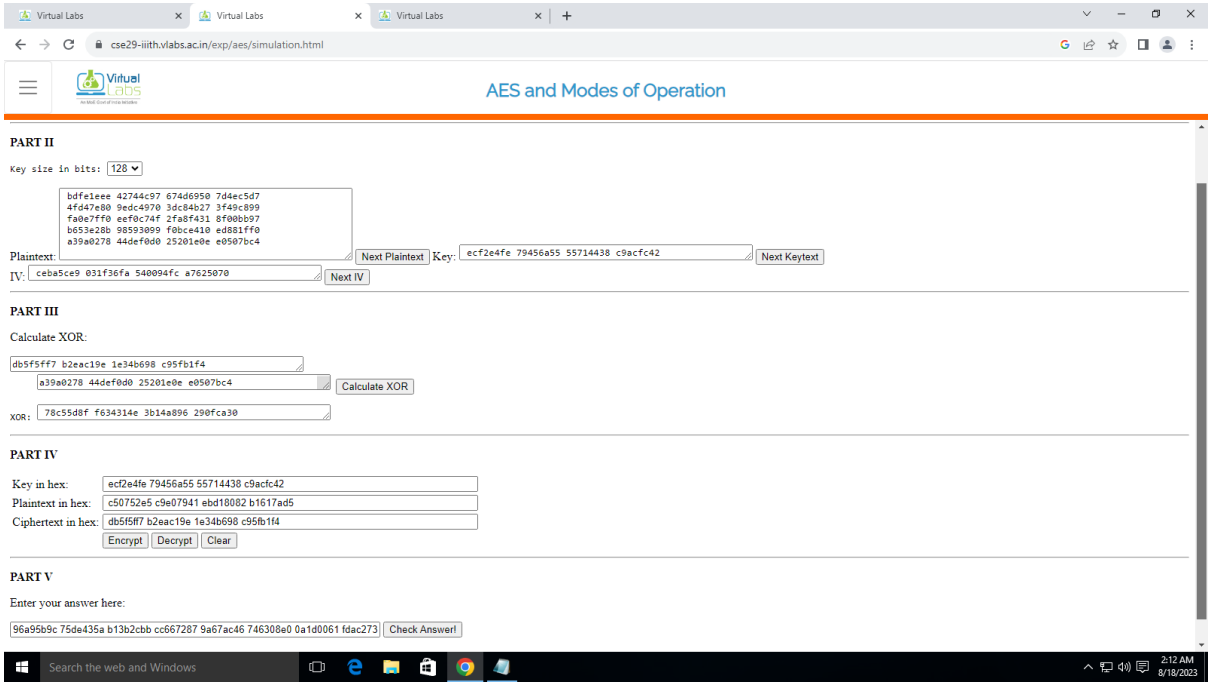
Output Feedback (OFB) is another mode that transforms AES into a stream cipher. It encrypts an Initialization Vector (IV) using the encryption key to produce a pseudorandom stream of bits. This bitstream is then XORed with the plaintext to generate ciphertext. OFB mode doesn't directly depend on the plaintext, so it can be used for both encryption and decryption. However, it's not as widely used as other modes due to its security limitations. Unlike some other modes, OFB does not require re-encrypting the key for each block, which can be an advantage in certain situations. However, it's important to note that OFB should not be used for encrypting large volumes of data where stronger security guarantees are required.

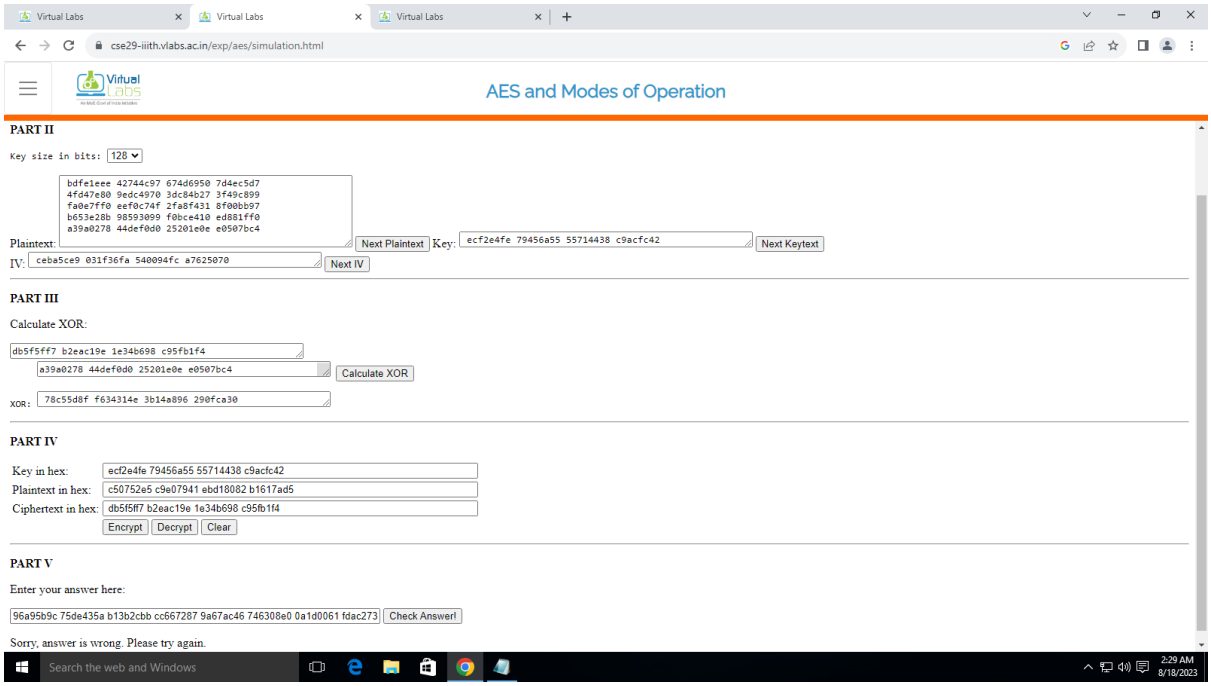
- OFB is another mode that turns AES into a stream cipher.
- It encrypts an IV with the key to produce a pseudorandom stream of bits.
- The bits are then XORed with the plaintext to generate ciphertext.
- Unlike CTR, OFB does not require re-encrypting the key for each block; it only needs the IV.
- However, it is not recommended for encrypting large volumes of data.

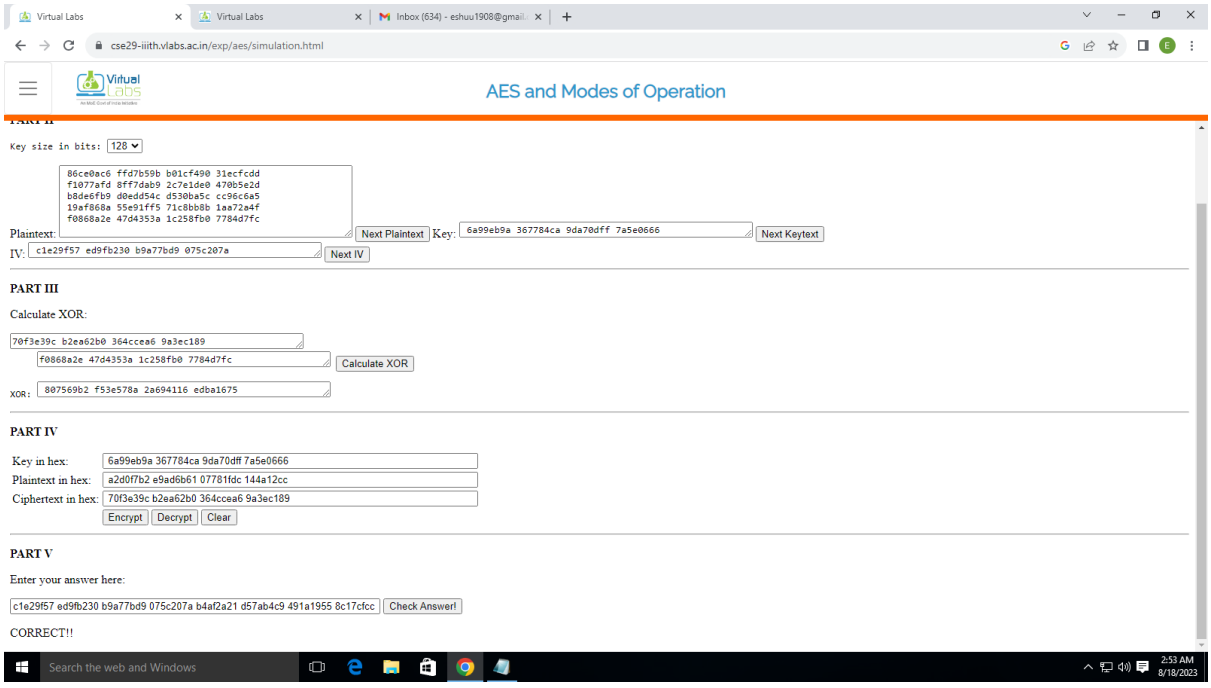

Virtual Labs
Virtual Labs
cse29-iiith.vlabs.ac.in/exp/aes/simulation.html
Virtual Labs
AES and Modes of Operation
PART II
Key size in bits: 128
1430bd11 701dd2ad bfd90f98 52049cbd
86b83f84 f13700e1 6dd03b7a 02a29598
95afbc5a 9cdcff65 cf9b1f23 561f9da3
ddf6a6b7 82449b94 a16d27e9 0310d43c
f13b7af0 5b671007 83297474 0672c47c
Plaintext:
Next Plaintext
Key: 5a9d79b4 76e99124 2faa9420 e3210f3f
Next Keytext
IV:
Next IV
CTR:
Next CTR
PART III
Calculate XOR:
Calculate XOR
XOR:
PART IV
Key in hex: 5a9d79b4 76e99124 2faa9420 e3210f3f
Plaintext in hex: f13b7af0 5b671007 83297474 0672c47c
Ciphertext in hex: 0fe1e57b 79855c69 45780e1e b1856ca3
Encrypt
Decrypt
Clear
PART V
Enter your answer here:
1c08acc3 1e42cd7d 1776f520 f8e5e32c 41b75986 5486ca4c ce4089c7 35d6ea55
Check Answer!
Search the web and Windows
1:36 AM
8/18/2023


Virtual Labs
Virtual Labs
cse29-iiith.vlabs.ac.in/exp/aes/simulation.html
Virtual Labs
AES and Modes of Operation
PART II
Key size in bits: 128
1430bd11 701dd2ad bfd90f98 52049cbd
86b83f84 f13700e1 6dd03b7a 02a29598
95afbc5a 9cdcff65 cf9b1f23 561f9da3
ddf6a6b7 82449b94 a16d27e9 0310d43c
f13b7af0 5b671007 83297474 0672c47c
Plaintext:
Next Plaintext
Key: 5a9d79b4 76e99124 2faa9420 e3210f3f
Next Keytext
IV: 4f064ce3 4956d135 bbb92277 45c23602
Next IV
PART III
Calculate XOR:
a2472f4b 6f5ccf95 c97c753a 1989a290
f13b7af0 5b671007 83297474 0672c47c
Calculate XOR
XOR: 537c55bb 343bdf92 4a55014e 1ffb66ec
PART IV
Key in hex: 5a9d79b4 76e99124 2faa9420 e3210f3f
Plaintext in hex: 537c55bb 343bdf92 4a55014e 1ffb66ec
Ciphertext in hex: ad793e7d af225296 f0096740 e0e2a17f
Encrypt
Decrypt
Clear
PART V
Enter your answer here:
4b30a556 7be970af 3093930d 7c9c3177 e19b07fc 138e9dbf 45c1e75e 6cac3b7c
Check Answer!
Search the web and Windows
1:44 AM
8/18/2023


Virtual Labs
cse29-iiith.vlabs.ac.in/exp/aes/simulation.html
AES and Modes of Operation
Key size in bits: 128
c824b9ea bb287ddf 973b6556 38a8bbae
4911297a d7418e63 9645551a d3b29bdc
1d7b181b 9fe5bc1b 536dd878 7496f8ef
9c29bc5e 858a4a0f ee05b840 7b7306bf
5da064cb b068aaec 04325a79 cc540f19
Plaintext:
Next Plaintext
Key: c47d8d00 7525c17a a6d51493 72c9ae50
Next Keytext
IV:
Next IV
CTR:
Next CTR
PART III
Calculate XOR:
Calculate XOR
XOR:
PART IV
Key in hex: c47d8d00 7525c17a a6d51493 72c9ae50
Plaintext in hex: 5da064cb b068aaec 04325a79 cc540f19
Ciphertext in hex: b4d51528 b4c50245 185c6150 8b94323f
Encrypt
Decrypt
Clear
PART V
Enter your answer here:
cc20e848 d9100d26 fd44a0f3 361fe0a5 278e6912 b567ec31 3b9bbb58 0bc25a76
Check Answer!
CORRECT!!
Search the web and Windows
1:57 AM
8/18/2023


Virtual Labs
cse29-iiith.vlabs.ac.in/exp/aes/simulation.html
AES and Modes of Operation
PART II
Key size in bits: 128
bdfe1eee 42744c97 674d6950 7d4ec5d7
4fd47e80 9edc4970 3dc84b27 3f49c899
fa0e7ff0 eef0c74f 2fa8f431 8f00bb97
b653e28b 98593099 f0bce410 ed881ff0
a39a0278 44def0d0 25201e0e e0507bc4
Plaintext:
Next Plaintext
Key: ecf2e4fe 79456a55 55714438 c9acfc42
Next Keytext
IV: ceba5ce9 031f36fa 540094fc a7625070
Next IV
PART III
Calculate XOR:
db5f5ff7 b2eac19e 1e34b698 c95fb1f4
a39a0278 44def0d0 25201e0e e0507bc4
Calculate XOR
XOR: 78c55d8f f634314e 3b14a896 290fca30
PART IV
Key in hex: ecf2e4fe 79456a55 55714438 c9acfc42
Plaintext in hex: c50752e5 c9e07941 ebd18082 b1617ad5
Ciphertext in hex: db5f5ff7 b2eac19e 1e34b698 c95fb1f4
Encrypt
Decrypt
Clear
PART V
Enter your answer here:
96a95b9c 75de435a b13b2cbb cc667287 9a67ac46 746308e0 0a1d0061 fdac273
Check Answer!
Search the web and Windows
2:12 AM
8/18/2023





Lab Outcome: LO2

Conclusion: Demonstrated key management, distribution and user authentication

Experiment No 10

Name: Shipra Suvarna

Roll No: T23-133

Aim : Different commands To find the hash of files and to perform audit

Theory :

Hashing is a fundamental concept in computer science and cryptography. It involves taking input data (in this case, a file) and applying a mathematical function to it to produce a fixedsize string of characters, which is typically a hexadecimal number. This output is known as the "hash value" or simply "hash."

The key properties of a good hashing function are:

Deterministic: For the same input data, the hash function should always produce the same hash value.

Fast to compute: Hashing should be a quick process, even for large files.

Pre-image resistance: Given a hash value, it should be computationally infeasible to reverse the process and determine the original input data.

Collision resistance: It should be extremely unlikely for two different inputs to produce the same hash value.

That contains some text data. So to get the MD5 (Message Digest 5) hash of the file, we would have to execute the command.

certutil -hashfile "C:\Users\Public\spars.txt" MD5

The command upon execution would produce an output similar to this.

MD5 hash of spars.txt:

cb21e6741817a2d3020e02bb94301ae4

CertUtil: -hashfile command completed successfully.

To get SHA512 hash of the above file the command and the output would appear as following :

Command: Type cd followed by the path to the folder.

Tip: You can drag and drop a folder from Windows Explorer to insert the path.

certutil -hashfile <file> MD5

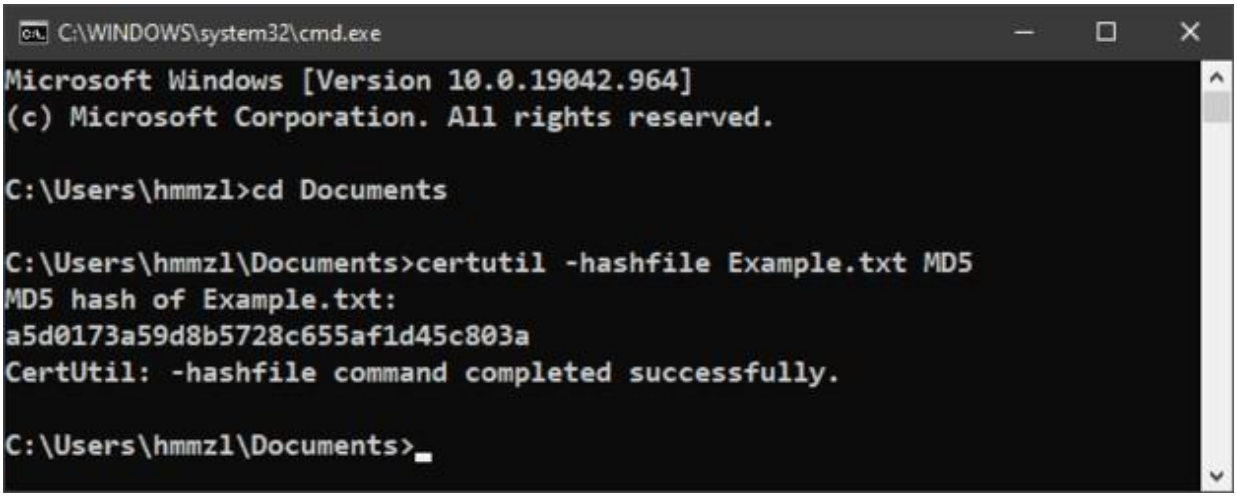
Replace <file> with the filename.

Tip: You can use the Tab key to have Windows complete the file name.

Example to get the MD5 hash for the file Example.txt:

certutil -hashfile Example.txt MD5

Output:



Conclusion:

In summary, hashing is a crucial technique in computer science and cryptography for converting data into fixed-size hash values. These hash values serve various purposes, including data integrity verification and password storage. A good hash function is deterministic, efficient, and resistant to pre-image and collision attacks.

LAB ASSIGNMENT 7

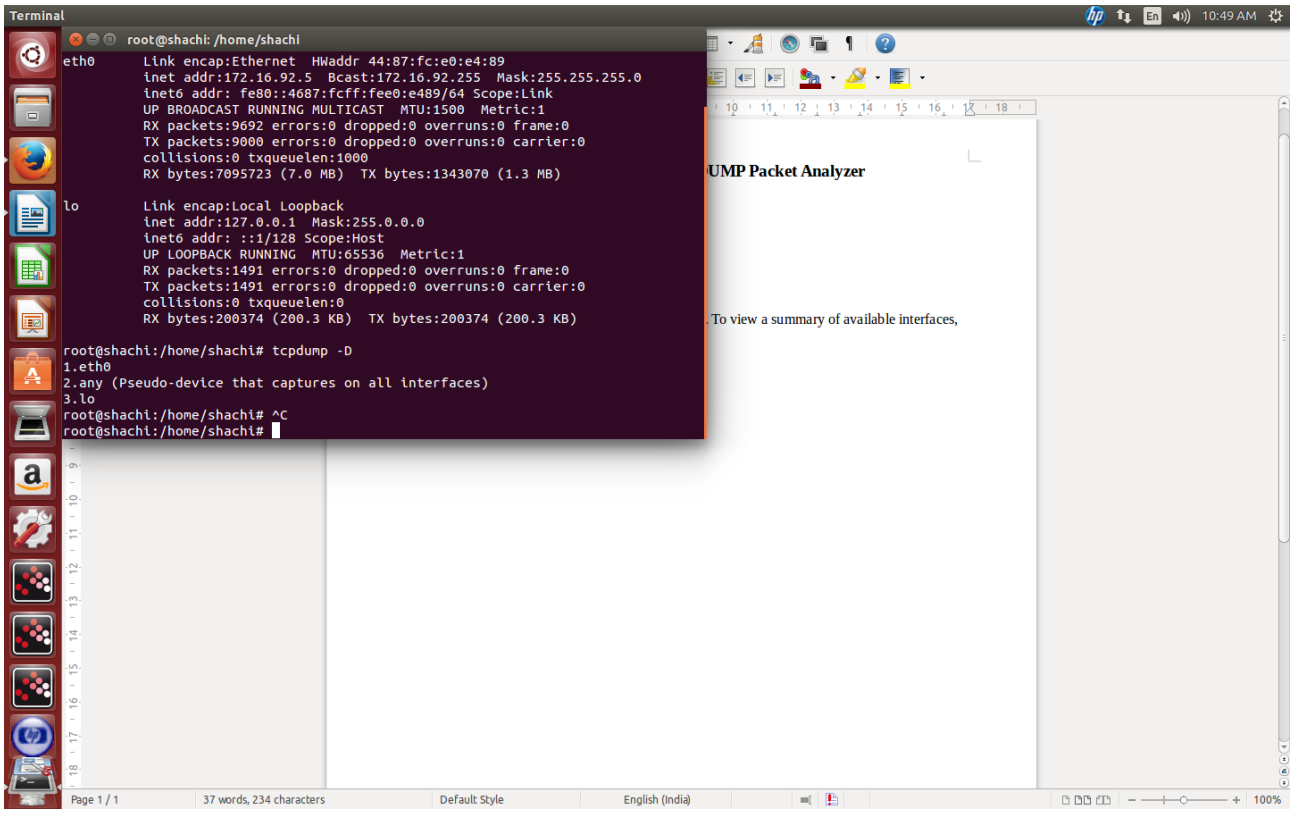
Name: Shipra Suvarna
Roll No: T23-133

Aim: Study of Packet Sniffer tools wireshark and tcpdump.

Theory:
1. To install tcpdump
$ sudo apt-get install tcpdump

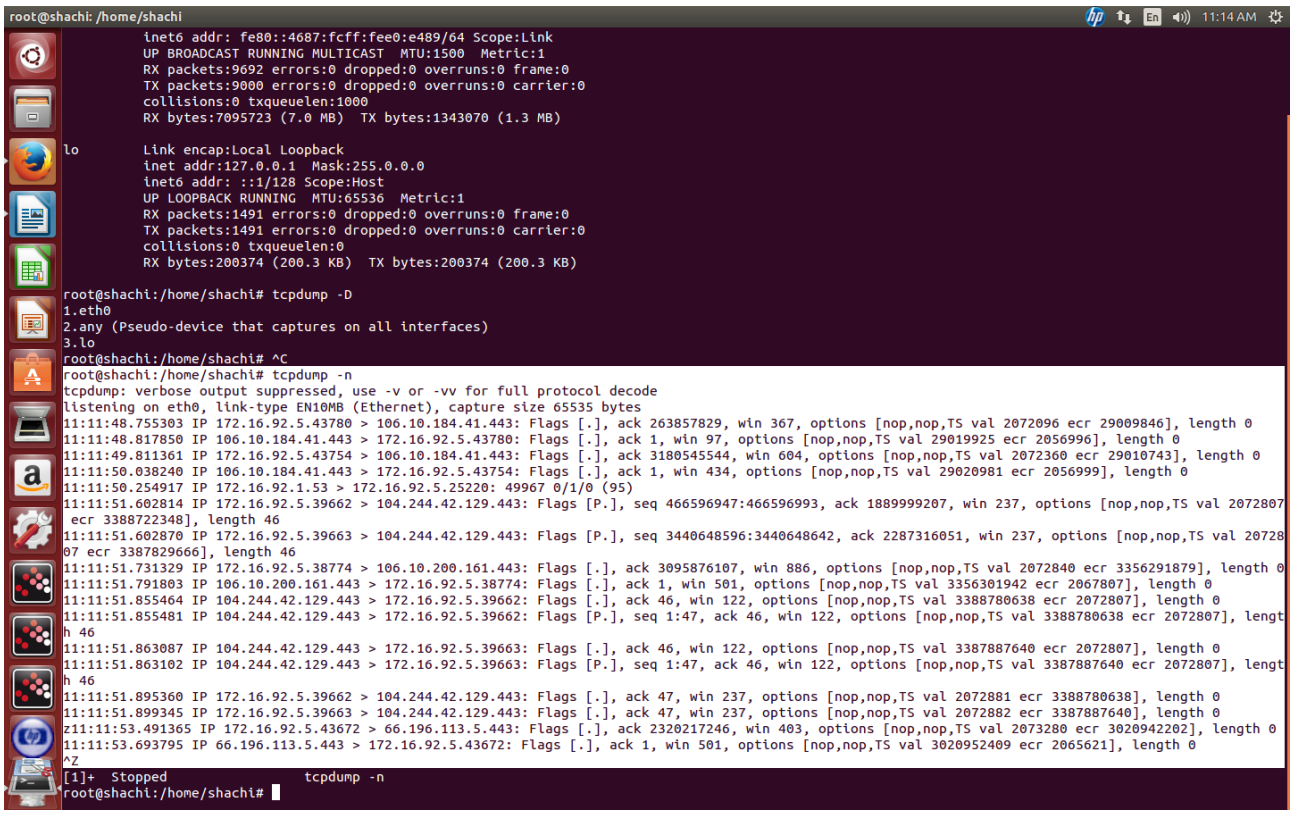
2. Choosing an interface:
By default, tcpdump captures packets on all interfaces. To view a summary of available interfaces, run

tcpdump -D



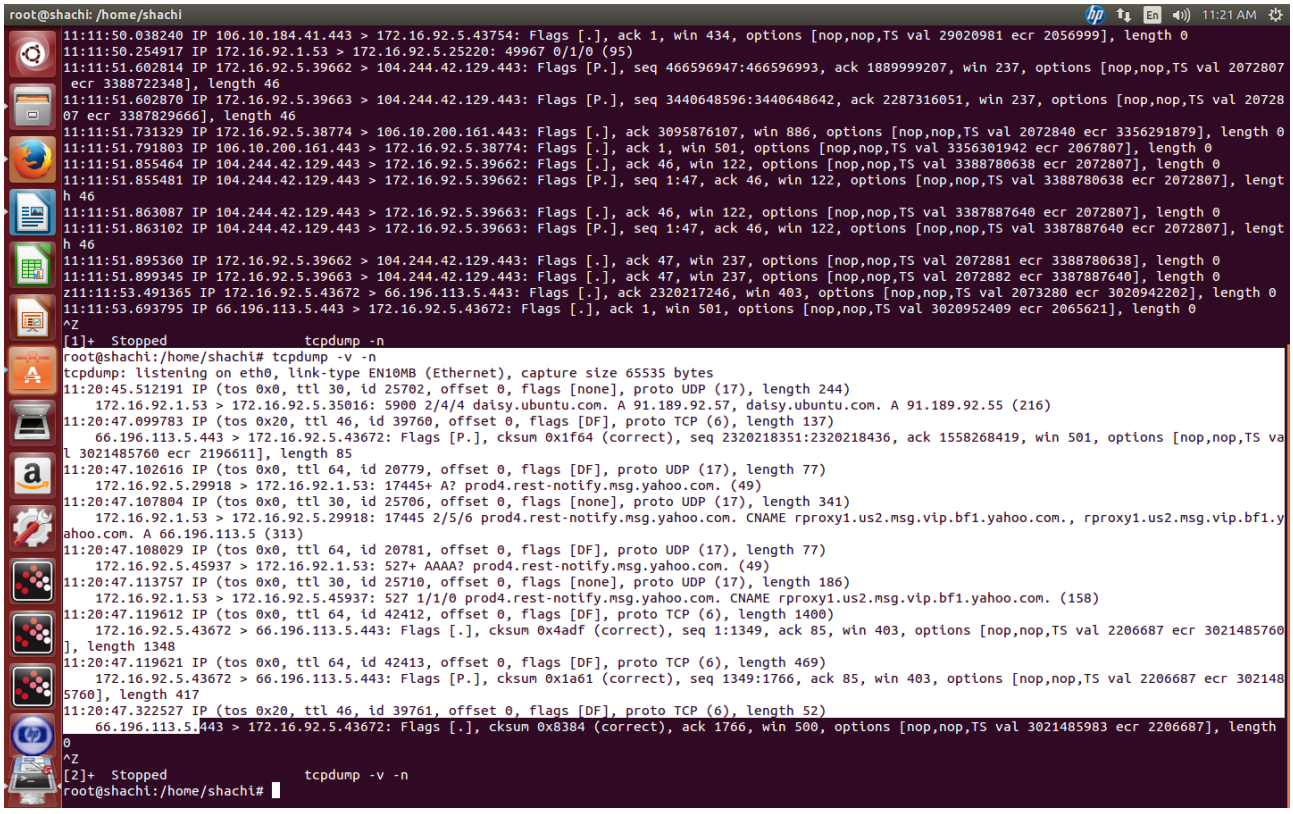
3. Basic command for sniffing
tcpdump -n

The -n parameter is given to stop tcpdump from resolving ip addresses to hostnames, which take look and not required right now.



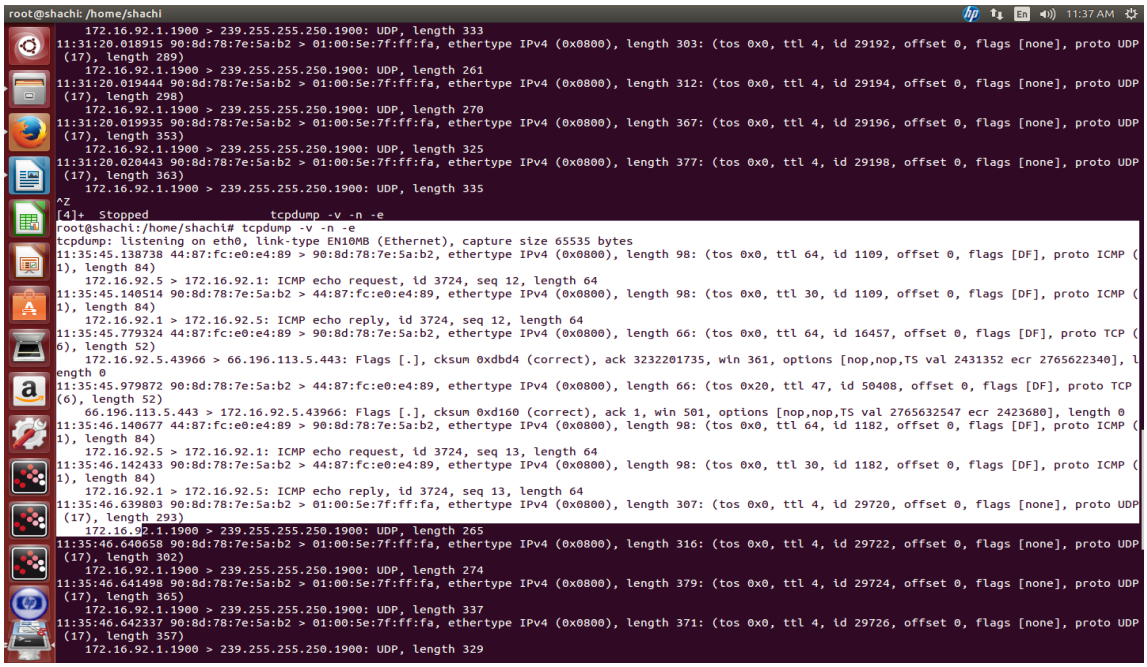
4. tcpdump -v -n

Now with the verbose switch lots of additional details about the packet are also being displayed. And these include the ttl, id, tcp flags, packet length etc.



5. Getting the ethernet header (link layer headers)

In the above examples details of the ethernet header are not printed. Use the -e option to print the ethernet header details as well.

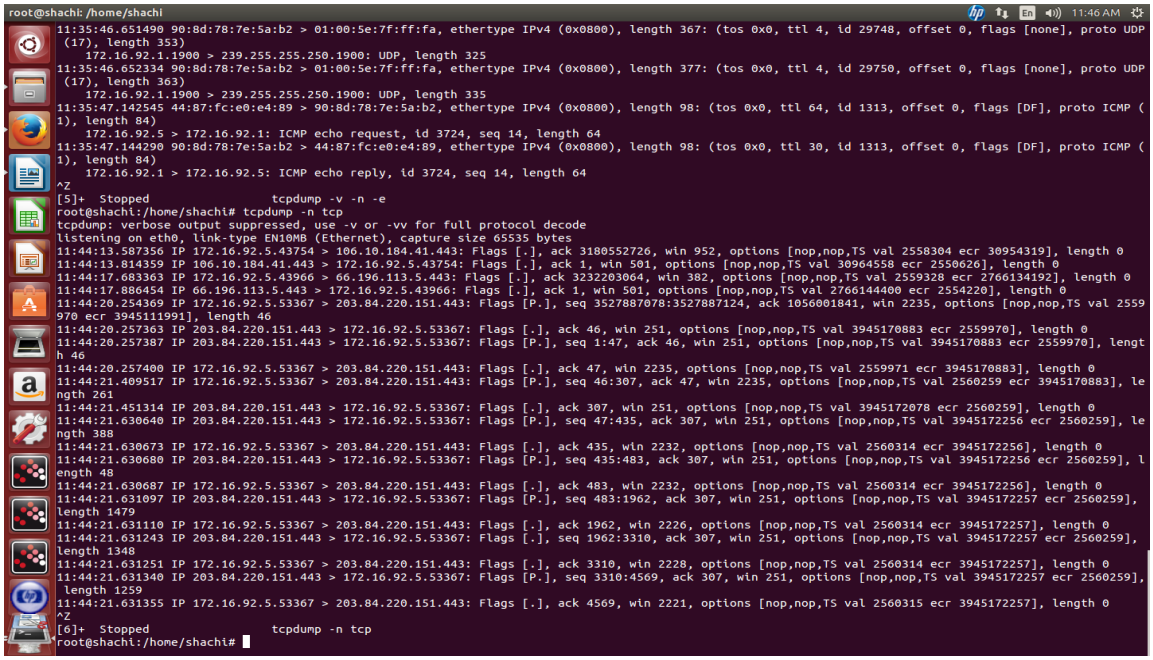


Filtering packets using expressions

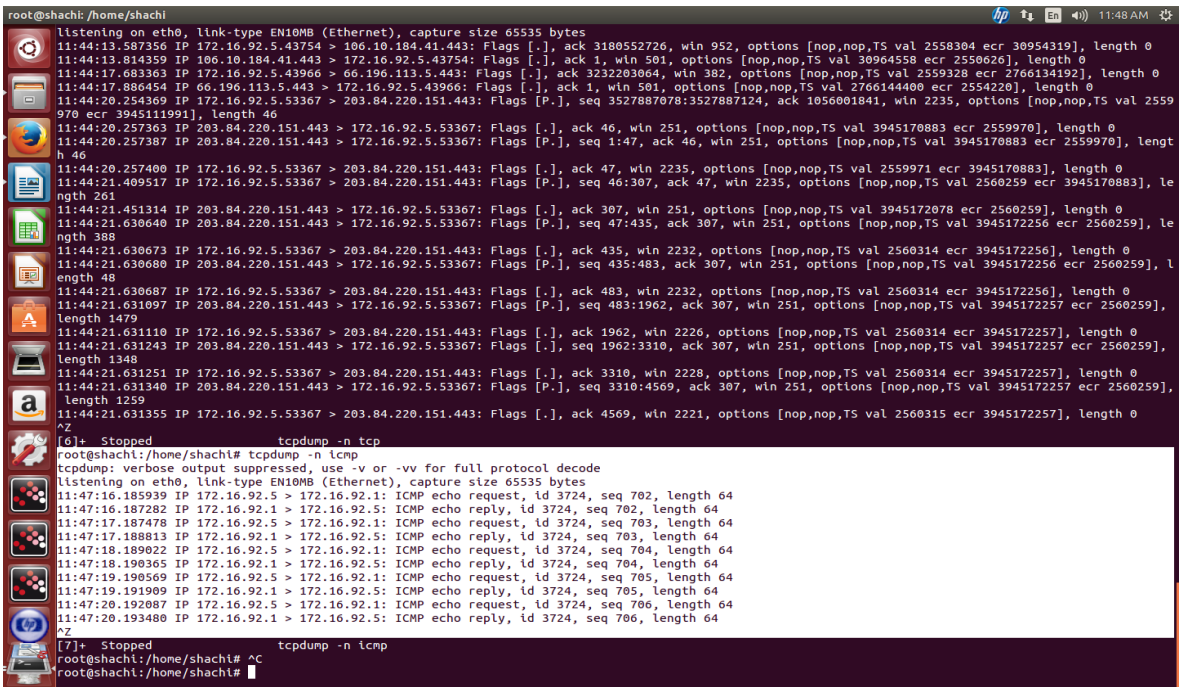
6. selecting packets with specific protocol

tcpdump -n tcp

#



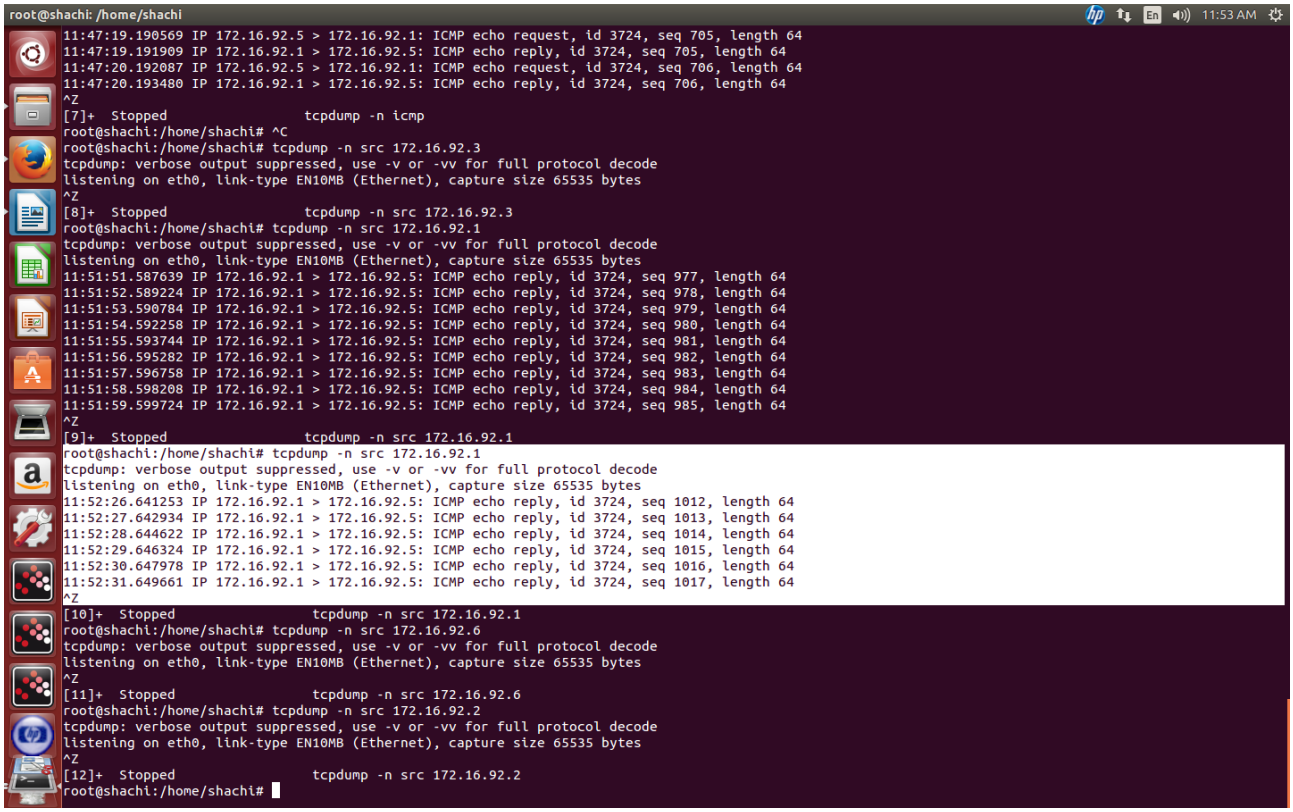
tcpdump -n icmp



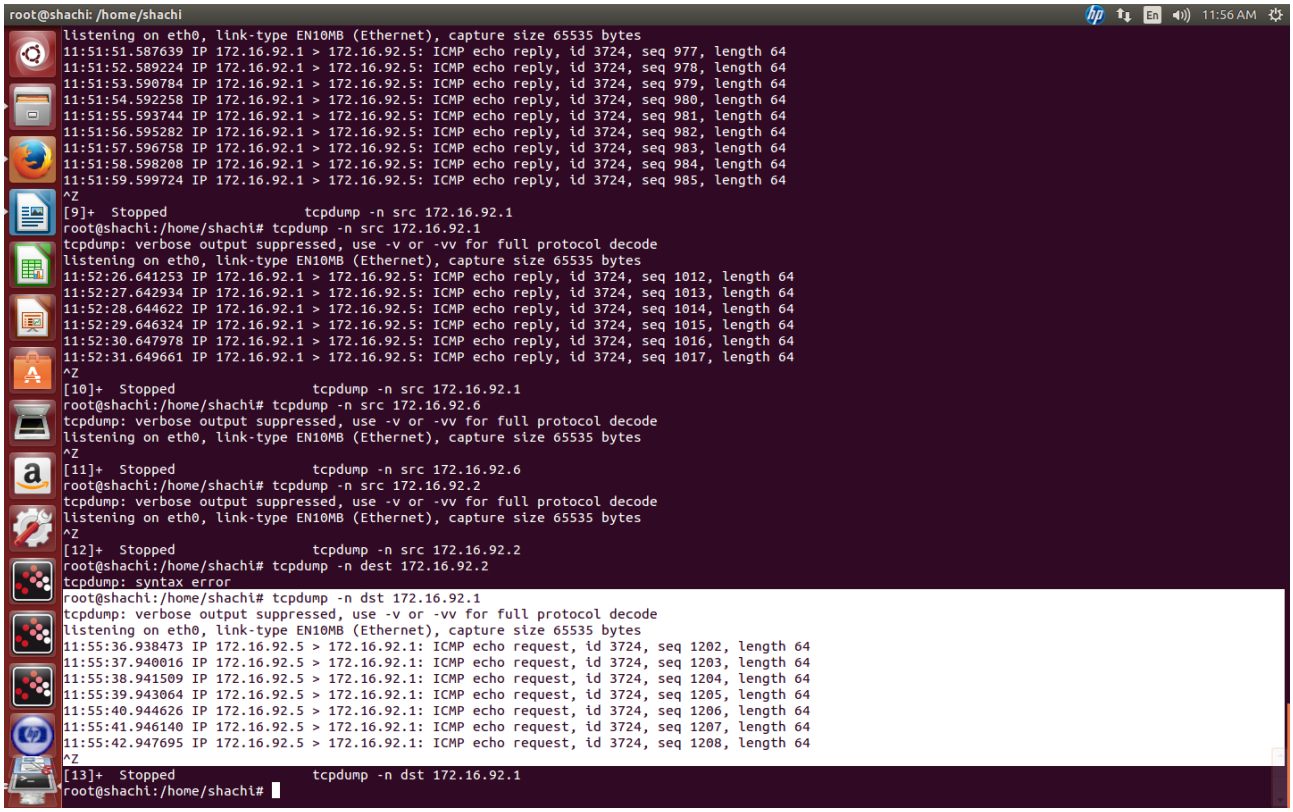
7. Particular host or port

Expressions can be used to specify source ip, destination ip, and port numbers. The next example picks up all those packets with source address 172.16.92.1

tcpdump -n src 172.16.92.1

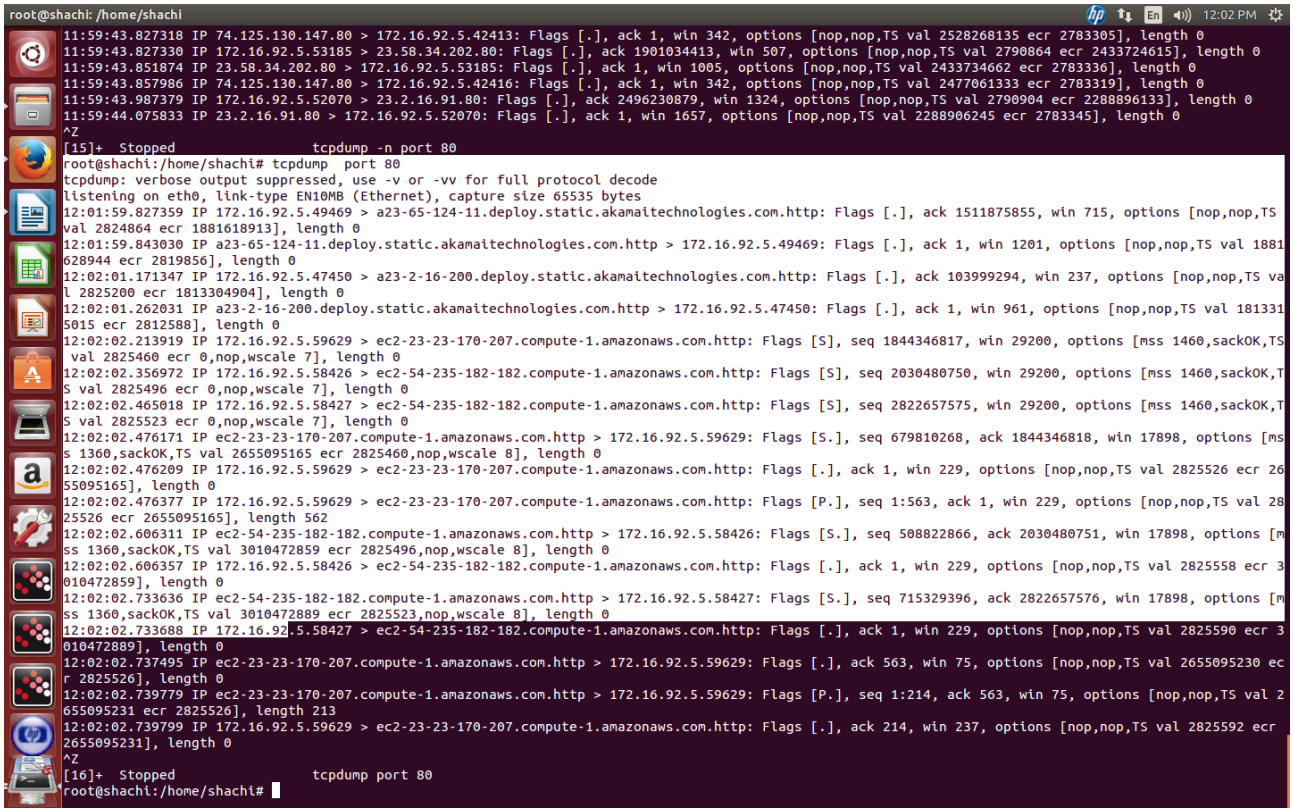


tcpdump -n dst 172.16.92.1



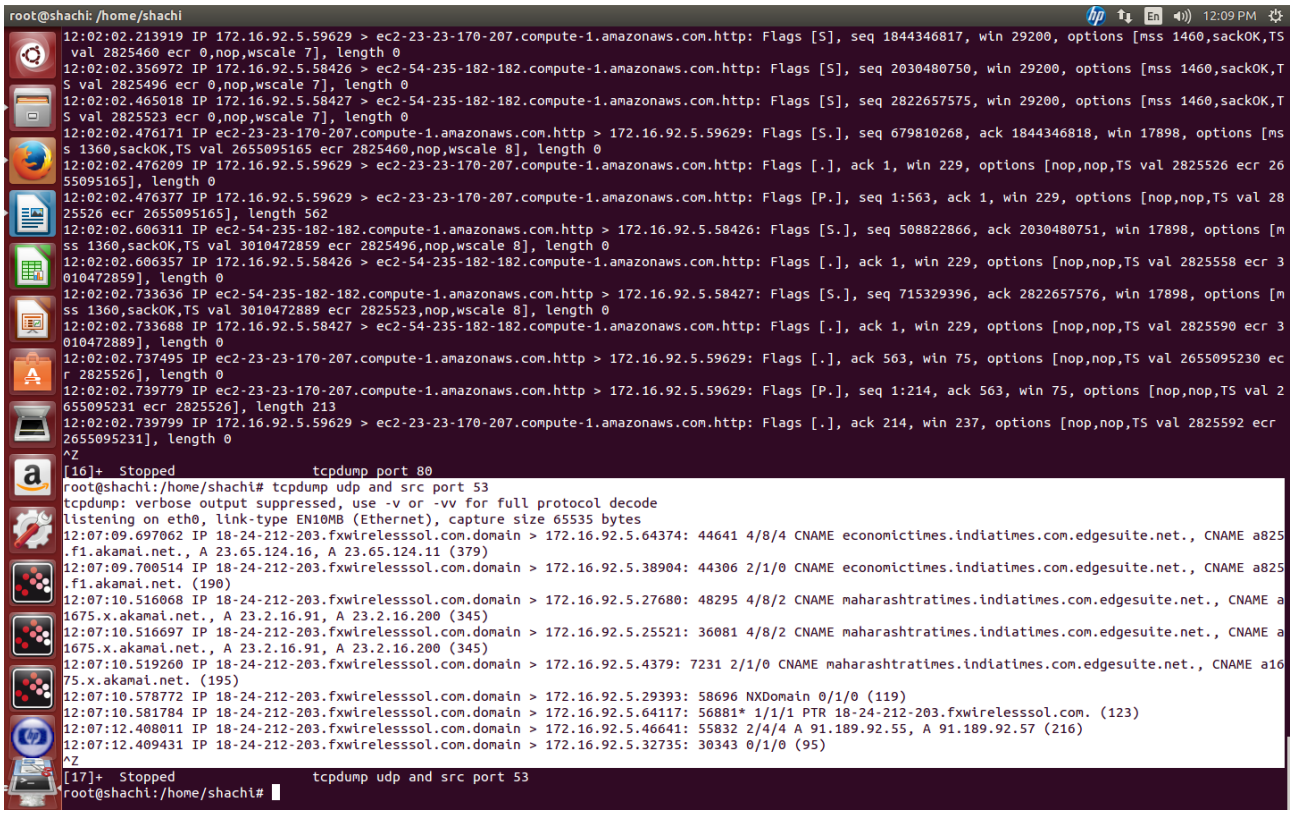
tcpdump -n port 80

tcpdump port 80



Specific Packets from specific port

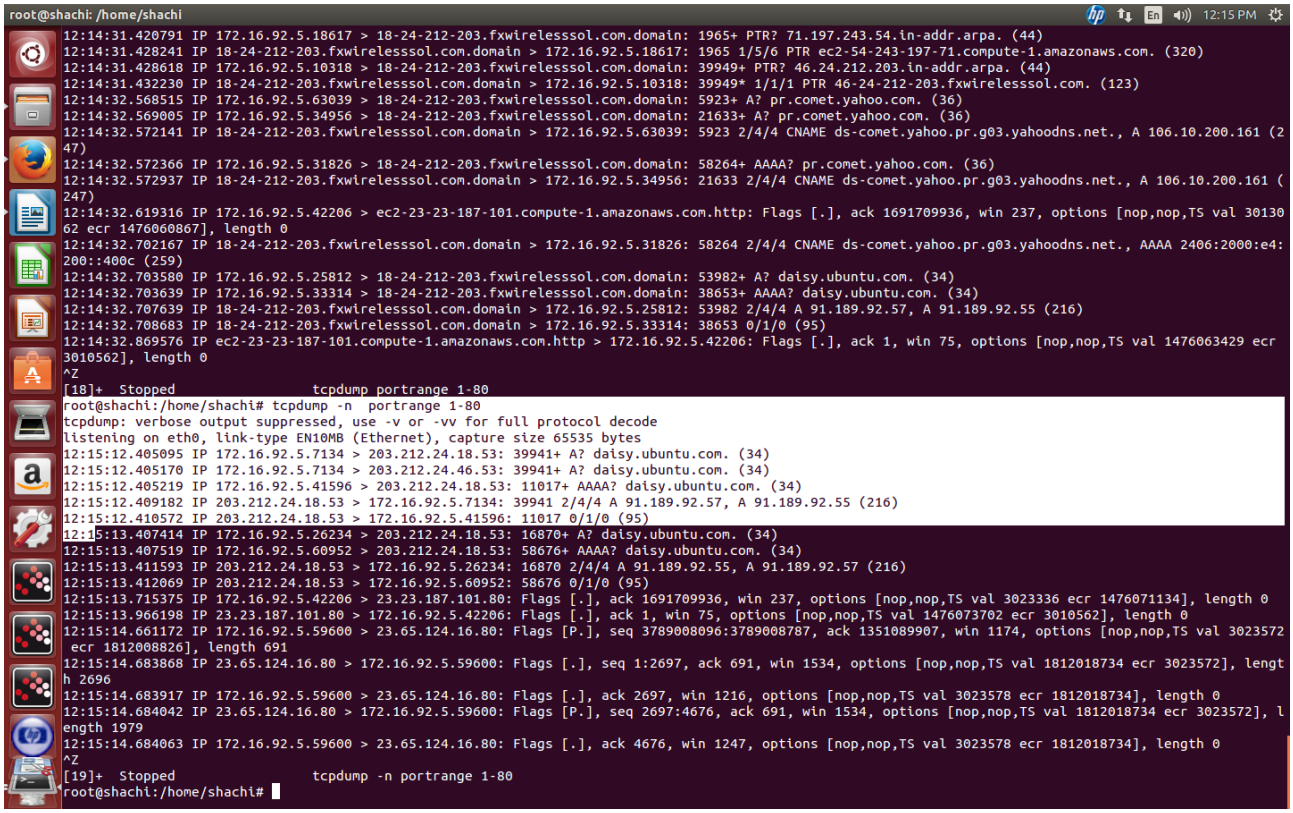
tcpdump udp and src port 53



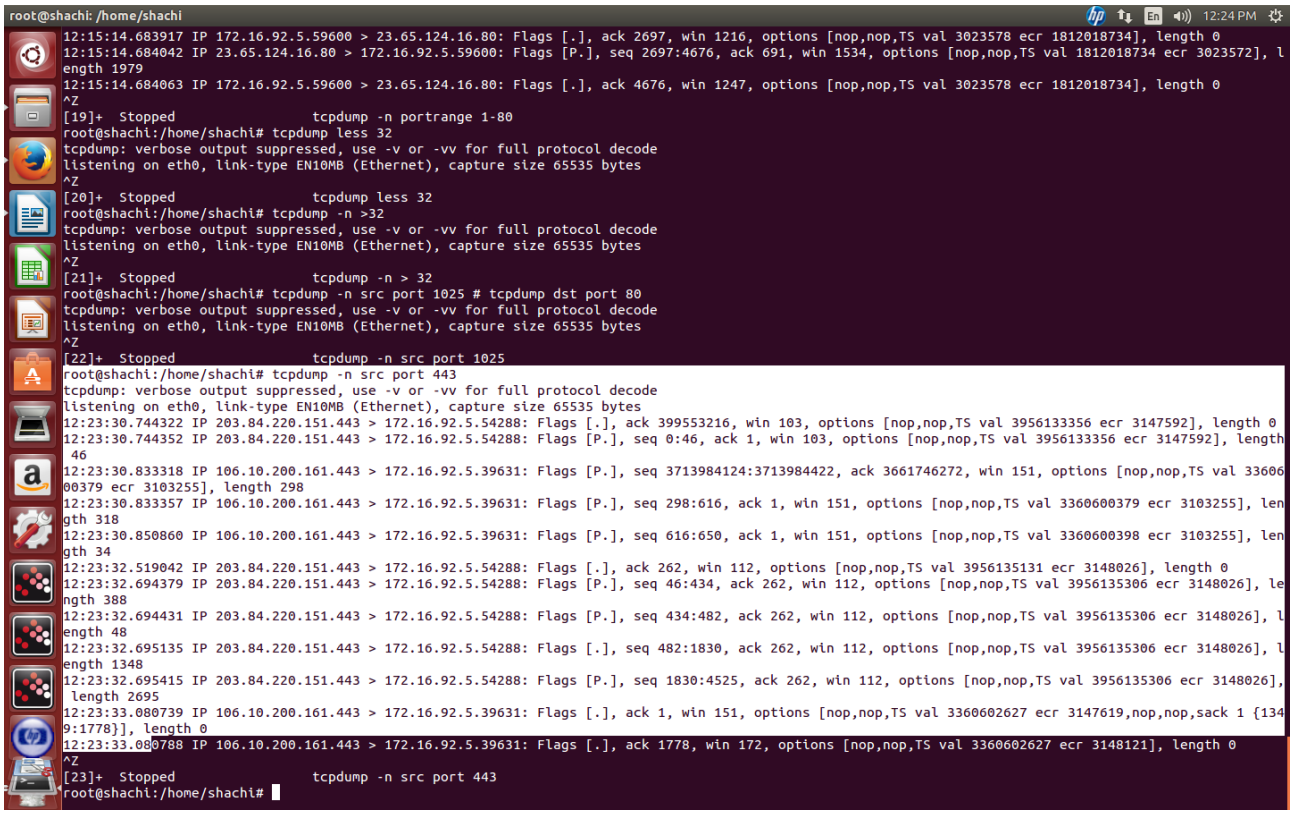
observing packets within a specific port range

tcpdump -n portrange 1-80

It shows all packets whose source or destination port is between 1 to 80



#tcpdump -n src port 443



Lab Outcome: LO3

Conclusion: Explored the different network reconnaissance tools to gather information about networks.

Assignment 8

Name: Shipra Suvarna
Roll No: T23-133

Aim: Installation of NMAP and using it with different options to scan open ports, perform OS fingerprinting, ping scan, TCP port scan, UDP post scan, etc.

Theory:

Port scanning:
While Nmap has grown in functionality over the years, it began as an efficient port scanner, and that remains its core function. The simple command **nmap** <target> scans 1,000 TCP ports on the host <target>. While many port scanners have traditionally lumped all ports into the open or closed states, Nmap is much more granular. It divides ports into six states: open, closed, filtered, unfiltered, open|filtered, or closed|filtered.

-sS (TCP SYN scan)
SYN scan is the default and most popular scan option for good reasons. It can be performed quickly, scanning thousands of ports per second on a fast network not hampered by restrictive firewalls. It is also relatively unobtrusive and stealthy since it never completes TCP connections. SYN scan works against any compliant TCP stack rather than depending on idiosyncrasies of specific platforms as Nmap's FIN/NULL/Xmas, Maimon and idle scans do. It also allows clear, reliable differentiation between the open, closed, and filtered states.

Command: nmap -sS google.com

```
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ sudo su
[sudo] password for lab1006:
root@lab1006-HP-280-G4-MT-Business-PC:/home/lab1006# nmap -sS tsec.edu

Starting Nmap 7.60 ( https://nmap.org ) at 2023-08-22 11:25 IST

root@lab1006-HP-280-G4-MT-Business-PC:/home/lab1006# nmap -sS google.com

Starting Nmap 7.60 ( https://nmap.org ) at 2023-08-22 11:27 IST
Nmap scan report for google.com (142.251.42.46)
Host is up (0.0034s latency).
Other addresses for google.com (not scanned): 2404:6800:4009:830::200e
rDNS record for 142.251.42.46: bom12s20-in-f14.1e100.net
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 17.68 seconds
```

```
admini@it906m13-HP-280-G2-MT:~$ sudo nmap -sU tsec.edu

Starting Nmap 7.60 ( https://nmap.org ) at 2023-08-09 15:09 IST
Stats: 0:01:00 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
UDP Scan Timing: About 5.74% done; ETC: 15:26 (0:16:08 remaining)
Stats: 0:01:02 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
UDP Scan Timing: About 5.77% done; ETC: 15:27 (0:16:53 remaining)
Stats: 0:01:09 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
UDP Scan Timing: About 6.04% done; ETC: 15:28 (0:17:53 remaining)
```

-sT (TCP connect scan)

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. This is the same high-level system call that web browsers, P2P clients, and most other network-enabled applications use to establish a connection. Rather than read raw packet responses off the wire, Nmap uses this API to obtain status information on each connection attempt.

Command: nmap -sT google.com

```
Nmap done: 1 IP address (1 host up) scanned in 21.04 seconds
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ nmap -sT tsec.edu

Starting Nmap 7.60 ( https://nmap.org ) at 2023-08-08 15:07 IST
Nmap scan report for tsec.edu (162.241.70.62)
Host is up (0.25s latency).
rDNS record for 162.241.70.62: 162-241-70-62.webhostbox.net
Not shown: 929 filtered ports, 59 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
25/tcp   open  smtp
53/tcp   open  domain
80/tcp   open  http
110/tcp  open  pop3
143/tcp  open  imap
443/tcp  open  https
465/tcp  open  smtps
587/tcp  open  submission
993/tcp  open  imaps
995/tcp  open  pop3s
3306/tcp open  mysql

Nmap done: 1 IP address (1 host up) scanned in 10.00 seconds
```

```
root@lab1006-HP-280-G4-MT-Business-PC:/home/lab1006# nmap -sT google.com

Starting Nmap 7.60 ( https://nmap.org ) at 2023-08-22 11:45 IST
Stats: 0:00:01 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 7.05% done; ETC: 11:45 (0:00:26 remaining)
Nmap scan report for google.com (142.251.42.46)
Host is up (0.0025s latency).
Other addresses for google.com (not scanned): 2404:6800:4009:830::200e
rDNS record for 142.251.42.46: bom12s20-in-f14.1e100.net
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 5.10 seconds
```

-sU (UDP scans)

While most popular services on the Internet run over the TCP protocol, UDP services are widely deployed. DNS, SNMP, and DHCP (registered ports 53, 161/162, and 67/68) are three of the most common. Because UDP scanning is generally slower and more difficult than TCP, some security auditors ignore these ports. This is a mistake, as exploitable UDP services are quite common and attackers certainly don't ignore the whole protocol. Fortunately, Nmap can help inventory UDP ports.

Command: nmap -sU google.com

```
admini@it906m13-HP-280-G2-MT:~$ sudo nmap -sU tsec.edu

Starting Nmap 7.60 ( https://nmap.org ) at 2023-08-09 15:09 IST
Stats: 0:01:00 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
UDP Scan Timing: About 5.74% done; ETC: 15:26 (0:16:08 remaining)
Stats: 0:01:02 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
UDP Scan Timing: About 5.77% done; ETC: 15:27 (0:16:53 remaining)
Stats: 0:01:09 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
UDP Scan Timing: About 6.04% done; ETC: 15:28 (0:17:53 remaining)

admini@it906m13-HP-280-G2-MT:~$ sudo nmap -sU google.com

Starting Nmap 7.60 ( https://nmap.org ) at 2023-08-09 15:11 IST
Stats: 0:00:09 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
UDP Scan Timing: About 85.45% done; ETC: 15:11 (0:00:02 remaining)
Nmap scan report for google.com (142.251.42.46)
Host is up (0.0038s latency).
Other addresses for google.com (not scanned): 2404:6800:4009:830::200e
rDNS record for 142.251.42.46: bom12s20-in-f14.1e100.net
Not shown: 999 open|filtered ports
PORT      STATE  SERVICE
33459/udp closed unknown

Nmap done: 1 IP address (1 host up) scanned in 10.70 seconds
```

-sN; -sF; -sX (TCP NULL, FIN, and Xmas scans) :

These three scan types (even more are possible with the --scanflags option described in the next section) exploit a subtle loophole in the TCP RFC to differentiate between open and closed ports. Page 65 of RFC 793 says that “if the [destination] port state is CLOSED .... an incoming segment not containing a RST causes a RST to be sent in response.” Then the next page discusses packets sent to

open ports without the SYN, RST, or ACK bits set, stating that: “you are unlikely to get here, but if you do, drop the segment, and return.”

When scanning systems compliant with this RFC text, any packet not containing SYN, RST, or ACK bits will result in a returned RST if the port is closed and no response at all if the port is open. As long as none of those three bits are included, any combination of the other three (FIN, PSH, and URG) are OK. Nmap exploits this with three scan types:

Null scan (-sN)

Does not set any bits (TCP flag header is 0)

command: nmap -sN google.com

FIN scan (-sF)

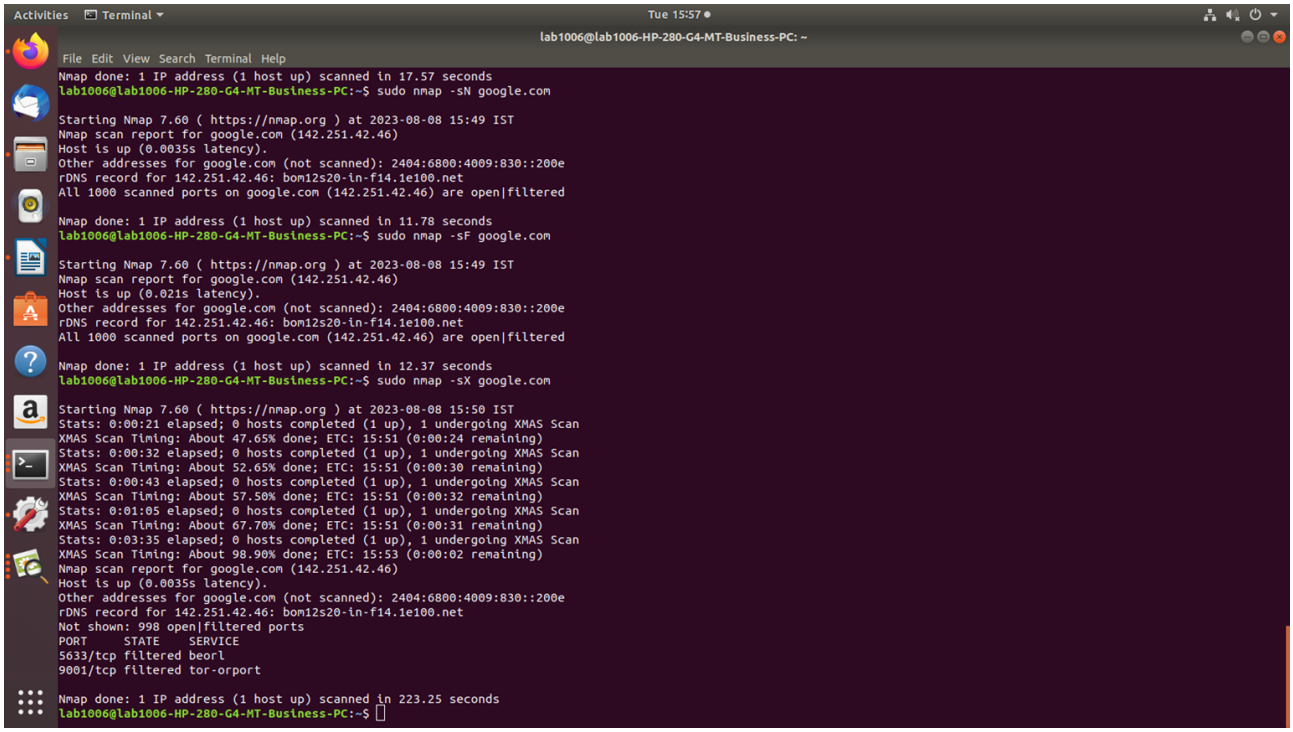
Sets just the TCP FIN bit.

Command: nmap -sF google.com

Xmas scan (-sX)

Sets the FIN, PSH, and URG flags, lighting the packet up like a Christmas tree.

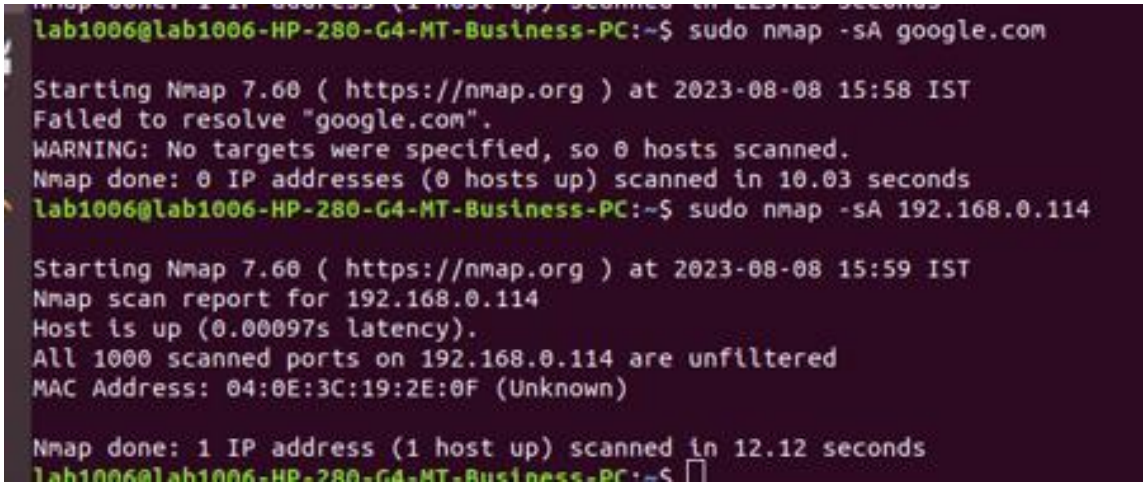
Command: nmap -sX google.com



-sA (TCP ACK scan)

This scan is different than the others discussed so far in that it never determines open (or even open|filtered) ports. It is used to map out firewall rulesets, determining whether they are stateful or not and which ports are filtered.

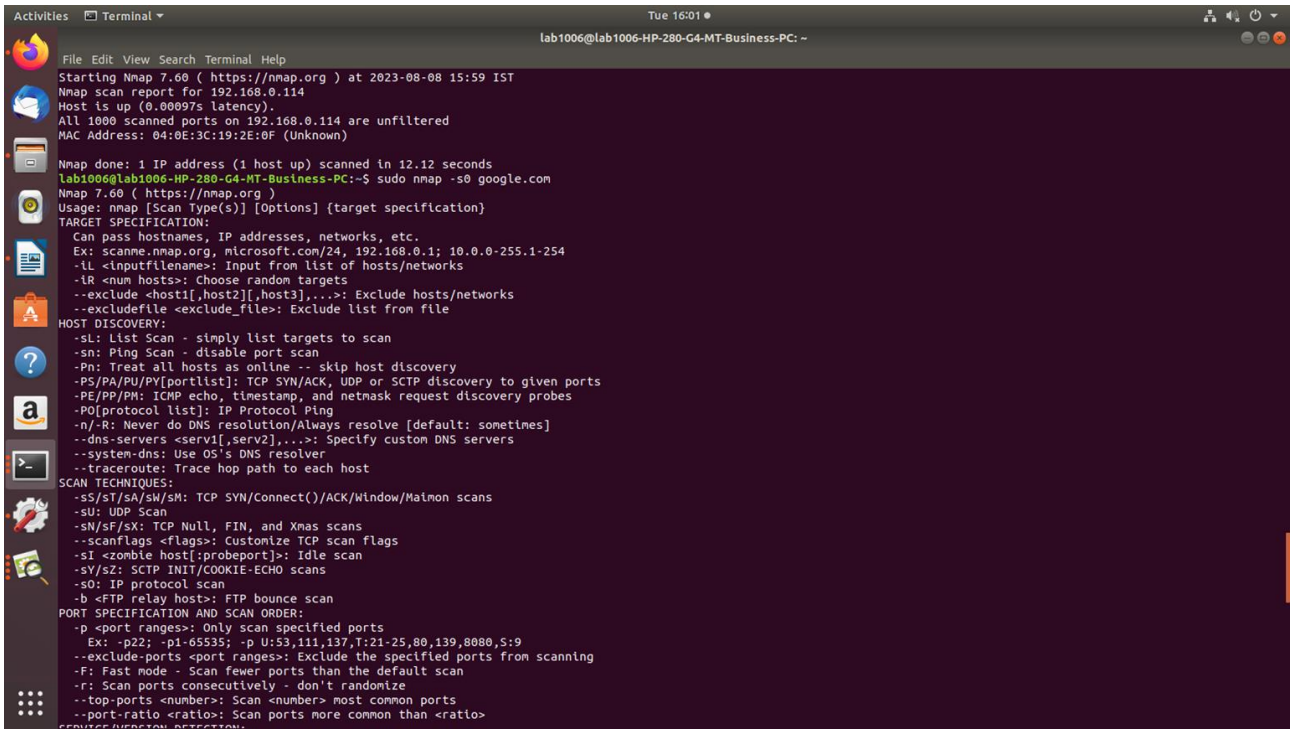
Command: nmap -sA google.com

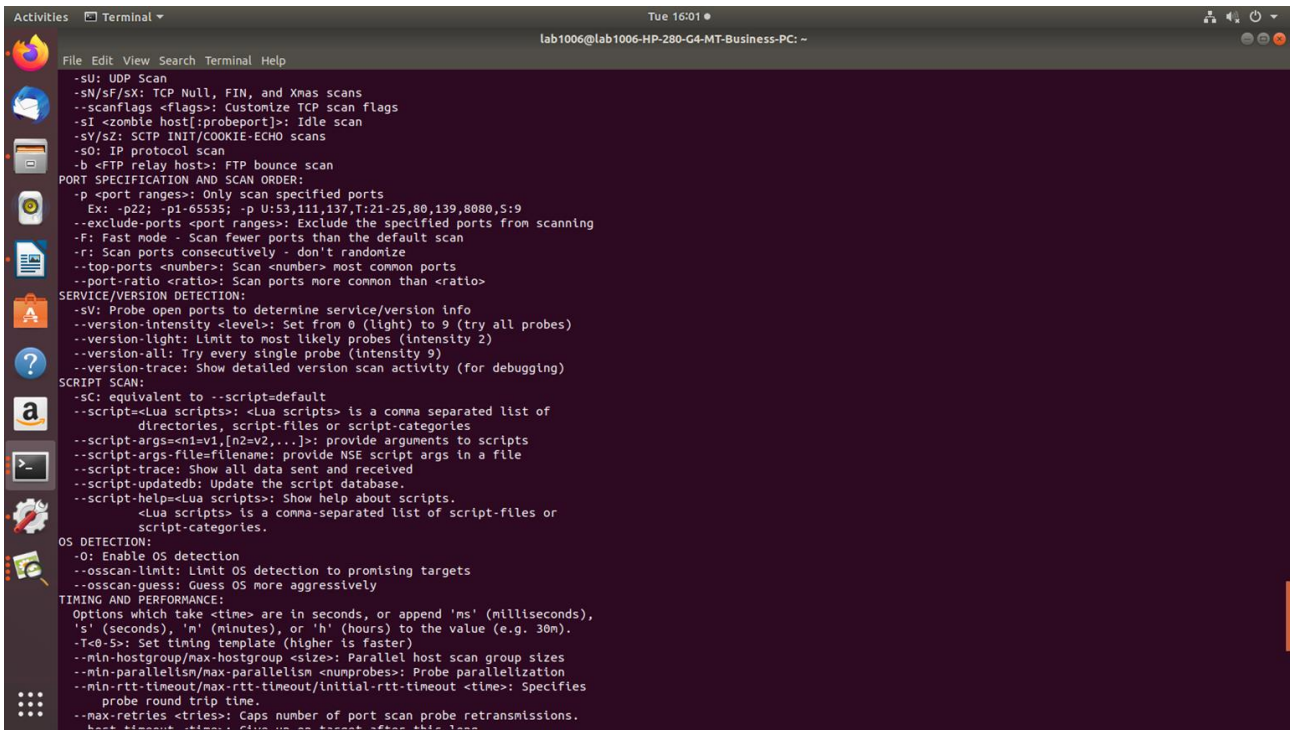


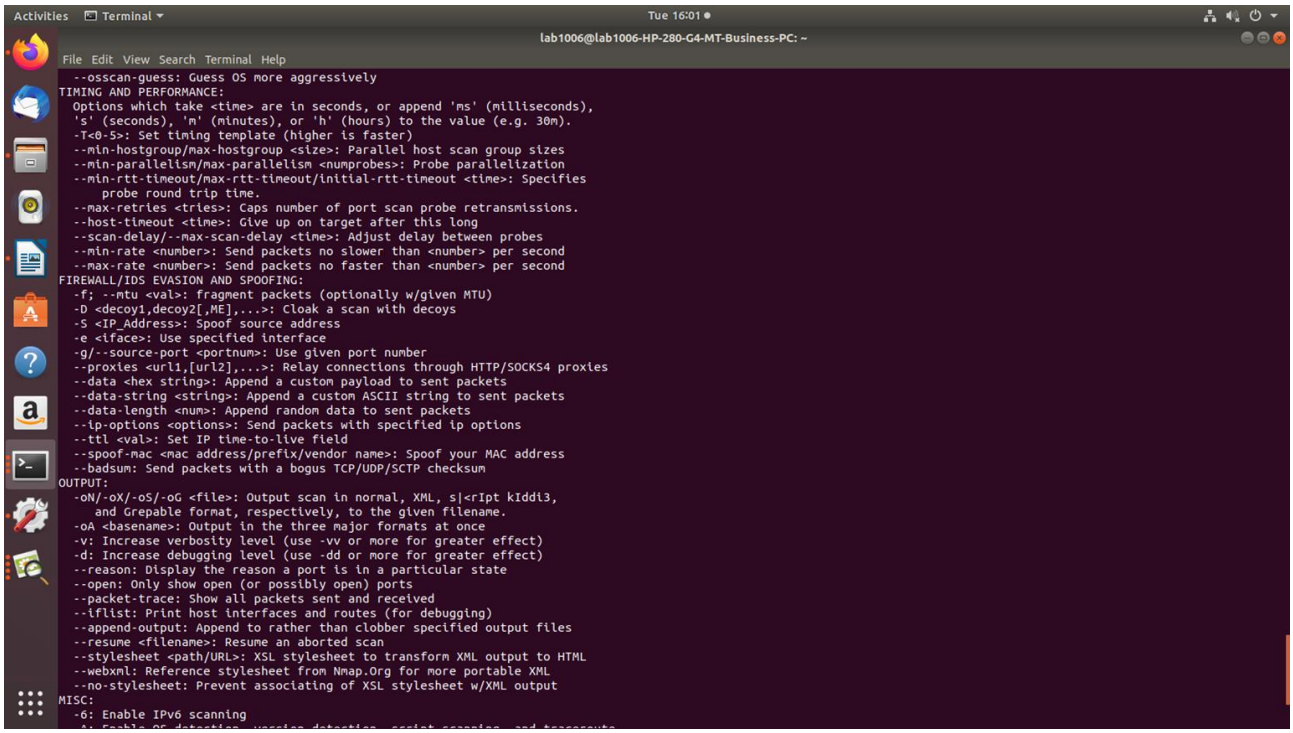
-sO (IP protocol scan)

IP protocol scan allows you to determine which IP protocols (TCP, ICMP, IGMP, etc.) are supported by target machines. This isn't technically a port scan, since it cycles through IP protocol numbers rather than TCP or UDP port numbers. Yet it still uses the -p option to select scanned protocol numbers, reports its results within the normal port table format, and even uses the same underlying scan engine as the true port scanning methods. So it is close enough to a port scan that it belongs here.

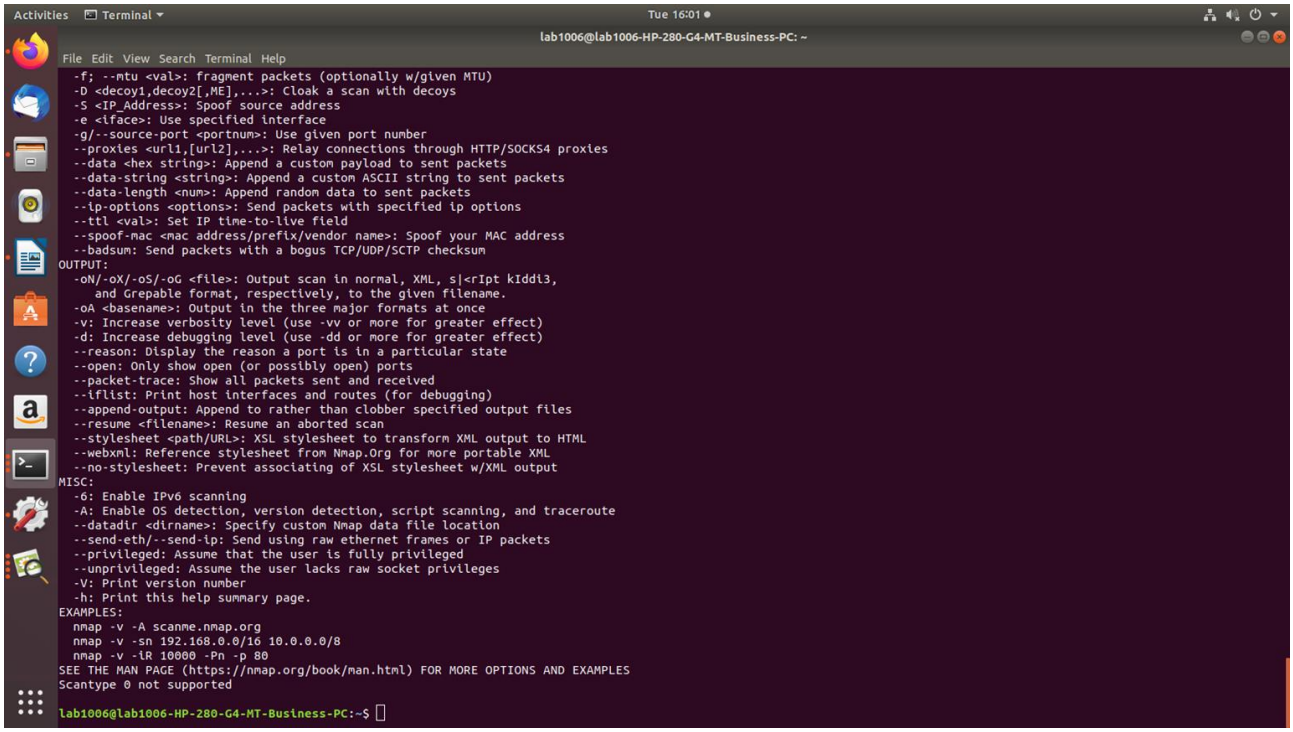
Command: nmap -sO google.com





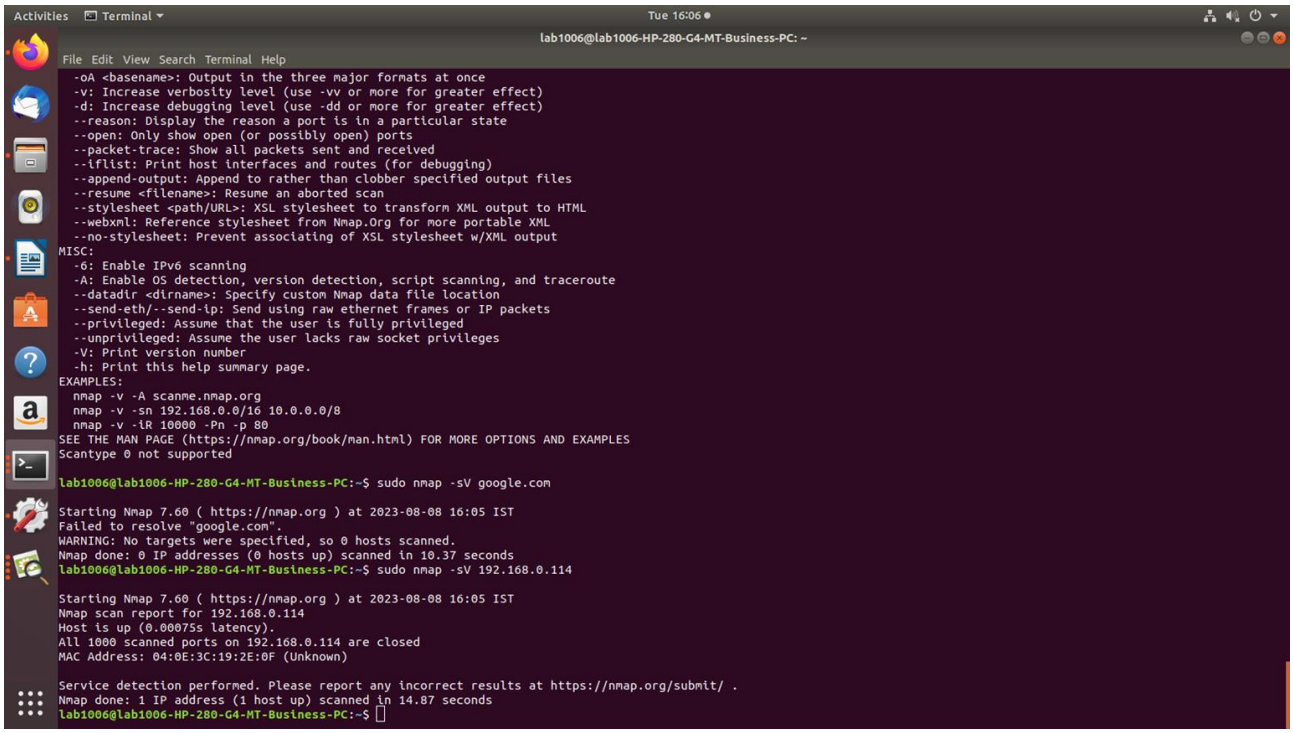



```
[chaos]# nmap -sO 127.0.0.1

Starting Nmap 4.01 at 2006-07-14 12:56 BST
Interesting protocols on chaos(127.0.0.1):
(The 251 protocols scanned but not shown below are
        in state: closed)
PROTOCOL STATE         SERVICE
1        open          icmp
2        open|filtered igmp
6        open          tcp
17       open          udp
255      open|filtered unknown

Nmap finished: 1 IP address (1 host up) scanned in
        1.259 seconds
```



-sV (Version detection)

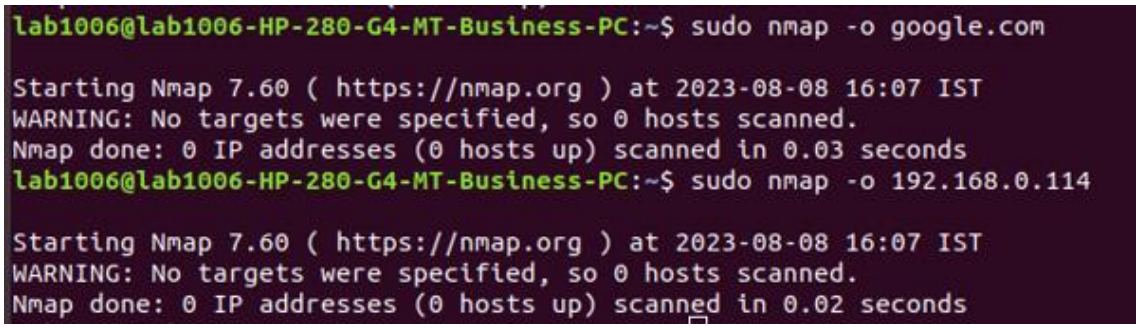
Enables version detection, as discussed above. Alternatively, you can use -A, which enables version detection among other things.



-O (Enable OS detection)

Enables OS detection, as discussed above. Alternatively, you can use -A to enable OS detection along with other things.

Command: nmap -o google.com



Ping scan Command:

This identifies all the IP addresses that are currently online without sending any packets to these hosts. This command is used to check the online status of a range of IP addresses by sending pings. It's a quick way to determine which hosts are active on a network without performing extensive port scans.

Command: nmap -sP 192.168.0.*

```
admini@it906m13-HP-280-G2-MT:~$ sudo nmap -sP 192.168.0.*

Starting Nmap 7.60 ( https://nmap.org ) at 2023-08-09 15:17 IST
Nmap scan report for _gateway (192.168.0.1)
Host is up (0.00031s latency).
MAC Address: AC:15:A2:B9:9E:29 (Unknown)
Nmap scan report for 192.168.0.105
Host is up (0.00044s latency).
MAC Address: A4:AE:12:84:7F:CF (Unknown)
Nmap scan report for 192.168.0.114
Host is up (-0.10s latency).
MAC Address: 04:0E:3C:19:2E:0F (Unknown)
Nmap scan report for 192.168.0.115
Host is up (-0.10s latency).
MAC Address: 04:0E:3C:1A:5C:AD (Unknown)
Nmap scan report for 192.168.0.116
Host is up (0.00047s latency).
MAC Address: 04:0E:3C:1A:60:A0 (Unknown)
Nmap scan report for 192.168.0.117
Host is up (0.00035s latency).
MAC Address: 04:0E:3C:19:2D:1C (Unknown)
Nmap scan report for 192.168.0.118
Host is up (0.00014s latency).
MAC Address: E4:54:E8:C6:37:76 (Unknown)
Nmap scan report for 192.168.0.119
Host is up (0.00032s latency).
MAC Address: 04:0E:3C:1A:5F:16 (Unknown)
Nmap scan report for 192.168.0.121
Host is up (0.00059s latency).
MAC Address: 90:8D:78:7E:5A:B3 (D-Link International)
Nmap scan report for 192.168.0.123
Host is up (0.00045s latency).
MAC Address: F4:39:09:49:0A:33 (Unknown)
Nmap scan report for 192.168.0.126
Host is up (-0.100s latency).
```

LO Mapped: LO4

Conclusion: We have thus completed installation of NMAP and used it with different options to scan open ports, perform OS fingerprinting, ping scan, TCP port scan, UDP post scan, etc.

Experiment No 9

Name: Shipra Suvarna
Roll No: T23-133

Aim : Simulate DOS attack using Hping3.

Lab Outcome :
LO5

Theory :

Understanding Denial of Service (DoS) Attack

A Denial of Service (DoS) attack is a malicious attempt to disrupt the normal functioning and availability of a network, system, service, or resource. The primary goal of a DoS attack is to overwhelm the target with a flood of traffic or to exploit vulnerabilities in such a way that it becomes inaccessible to its legitimate users. Let's delve deeper into some common types of DoS attacks:

SYN Flood Attack

A SYN Flood Attack is a sophisticated form of DoS attack that exploits the way TCP (Transmission Control Protocol) connections are established. In a standard TCP handshake, when a client wants to establish a connection with a server, it sends a SYN (synchronize) packet to initiate the connection. The server responds with a SYN-ACK (synchronizeacknowledge) packet, and the client completes the handshake with an ACK (acknowledge) packet.

In a SYN flood attack, the attacker sends an excessive number of SYN packets to the target server but does not follow up with ACK packets to complete the handshake. Instead, the attacker continually sends new SYN packets, causing the server to allocate resources for incomplete connections. Over time, these half-open connections can accumulate and exhaust the server's resources, making it unable to respond to legitimate connection requests. This effectively denies service to legitimate users.

ICMP Flood Attack (Ping Flood Attack)

An ICMP Flood Attack, commonly known as a Ping Flood Attack, targets the Internet Control Message Protocol (ICMP). ICMP is used for various network diagnostic purposes, including the famous "ping" utility, which checks the reachability of a network host. In a Ping Flood Attack,

the attacker sends an overwhelming number of ICMP echo-request packets (ping requests) to a target device.

The target device, as per standard ICMP behavior, responds to each incoming echo-request with an echo-reply. In the case of a flood attack, the attacker's goal is to generate an excessive number of echo-requests, forcing the target to respond with an equal number of echo-replies. This massive traffic can quickly consume the target's network and computing resources, causing it to become unresponsive to legitimate network traffic.

SMURF Attack

A SMURF attack is a type of Denial of Service (DoS) attack that targets the Internet Control Message Protocol (ICMP) and leverages a technique called "amplification." In a SMURF attack, the attacker sends a large number of ICMP echo-request packets (commonly known as "pings") to an intermediate network, which then reflects these packets to a victim's IP address. This results in a flood of responses overwhelming the victim's network and causing a DoS condition.

Here's how a SMURF attack works:

1. The attacker sends a large number of ICMP echo-request packets (pings) to the broadcast address of an intermediate network.

2. The routers on the intermediate network, as per standard behavior, broadcast these ICMP requests to all hosts on the network.

3. Numerous hosts on the intermediate network respond to these ICMP requests by sending ICMP echo-reply packets to the source IP address specified in the requests. Since the source IP address in the requests is the victim's IP address, these responses flood the victim's network.

4. The victim's network becomes overwhelmed with ICMP traffic, leading to high resource utilization and unavailability of services, effectively causing a DoS condition.

Hping3 Commands for SYN Flood and ICMP Flood

SYN Flood using Hping3

```bash
hping3 -c 15000 -d 120 -S -w 64 -p 80 --flood --rand-source 192.168.1.159
```

- `-c 15000`: Specifies sending 15,000 packets.
- `-d 120`: Sets the data size to 120 bytes in each packet.
- `-S`: Sets the SYN flag in TCP packets.
- `-w 64`: Defines a window size of 64.
- `-p 80`: Targets port 80 (commonly used for HTTP).
- `--flood`: Floods the target with packets continuously.
- `--rand-source`: Utilizes random source IP addresses.
- `192.168.1.159`: Specifies the target IP address.

ICMP Flood using Hping3

```bash
hping3 -1 --flood -a 192.168.103
192.168.1.255
```

- `-1`: Indicates the use of ICMP echo (ping) requests.
- `--flood`: Initiates the continuous flooding of the target.
- `-a 192.168.103`: Spoofs the source IP address as 192.168.103.
- `192.168.1.255`: Targets the broadcast address, causing multiple devices on the network to respond.

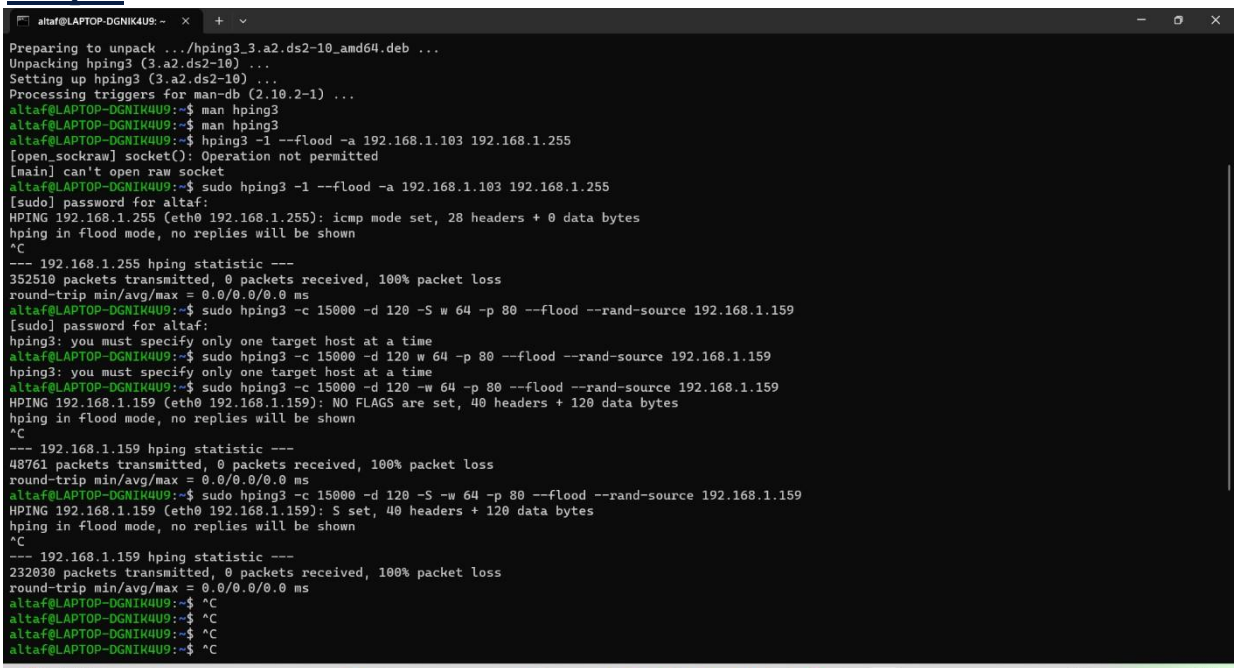
Example Hping3 Command for a SMURF Attack

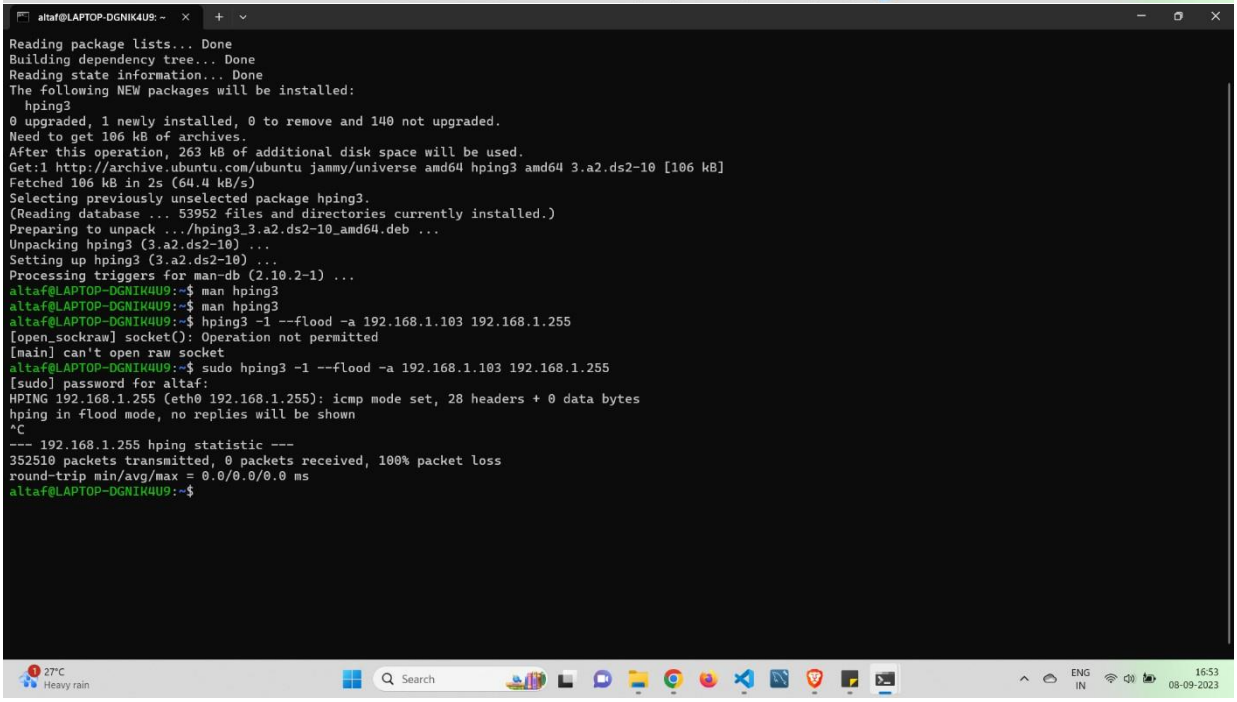
In a SMURF attack, Hping3 can be used to generate ICMP echo-request packets and send them to the broadcast address of an intermediate network, causing amplification and flooding. Below is an example command:

```bash
hping3 -1 --flood -a <spoofed_source_ip>
<broadcast_address>
```

- `-1`: Indicates the use of ICMP echo (ping) requests.
- `--flood`: Initiates the continuous flooding of the target.
- `-a <spoofed_source_ip>`: Spoofs the source IP address as `<spoofed_source_ip>`. The attacker often uses a spoofed IP address to hide their identity.
- `<broadcast_address>`: Specifies the broadcast address of the intermediate network. This is where the ICMP requests are sent.

Output:





Conclusion:

In summary, DoS attacks disrupt network services, with SYN Flood exploiting TCP handshakes and ICMP Flood inundating with ping requests. Effective defense requires awareness, security policies, and intrusion detection.

Assignment 12

Name: Shipra Suvarna
Roll No: T23-133

Aim: Explore the GPG tool of linux to implement email security.
Theory:
The GNU Privacy Guard, also known as GnuPG or simply GPG, is a popular open source OpenPGP (RFC4880) implementation. The system is widely trusted for securing integrity and confidentiality of internet communications through various cryptographic methods.
The GNU Privacy Guard (GPG or gpg) tool is a native/baseos security tool for encrypting files. According to the gpg man page: gpg is the OpenPGP (Pretty Good Privacy) part of the GNU Privacy Guard (GnuPG). It is a tool to provide digital encryption and signing services using the OpenPGP standard.

Step 1: Generate private key and public key pairs for sender and receiver using command

```
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --gen-key
gpg (GnuPG) 2.2.4; Copyright (C) 2017 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Note: Use "gpg --full-generate-key" for a full featured key generation dialog.

GnuPG needs to construct a user ID to identify your key.

Real name: Shipr
Email address: rajeshwarimahapatra5@gmail.com
You selected this USER-ID:
    "Shipr <rajeshwarimahapatra5@gmail.com>"

Change (N)ame, (E)mail, or (O)kay/(Q)uit? o
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key 61DE05010FABA03E marked as ultimately trusted
gpg: directory '/home/lab1006/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/lab1006/.gnupg/openpgp-revocs.d/E0A259E4CFED0871ECAFF2F461DE05010FABA03E.rev'
public and secret key created and signed.

pub   rsa3072 2023-09-26 [SC] [expires: 2025-09-25]
      E0A259E4CFED0871ECAFF2F461DE05010FABA03E
uid                      Shipr <rajeshwarimahapatra5@gmail.com>
sub   rsa3072 2023-09-26 [E] [expires: 2025-09-25]
```

```
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --output -a Shipr>Shipra
gpg: WARNING: no command supplied.  Trying to guess what you mean ...
gpg: can't open 'Shipr'
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --output -a Shipr>cns.txt
gpg: WARNING: no command supplied.  Trying to guess what you mean ...
gpg: can't open 'Shipr'
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --output -a Shipr>Cns.txt
gpg: WARNING: no command supplied.  Trying to guess what you mean ...
gpg: can't open 'Shipr'
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --export -a Shipr>Cns.txt
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --export -a Amisha>Cns.txt
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --export-secret-key -a Shipr>Cns.txt
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --fingerprint shahiamisha808@gmail.com
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   2  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 2u
gpg: next trustdb check due at 2025-09-25
pub   rsa3072 2023-09-26 [SC] [expires: 2025-09-25]
      CD17 B015 BFD1 D9B3 A12C  872E 0777 14E4 7AE7 7D5A
uid           [ultimate] amisha <shahiamisha808@gmail.com>
sub   rsa3072 2023-09-26 [E] [expires: 2025-09-25]
```

Step 2: Create a file containing sender's public key which then can be sent to other users.
Step 3: Similarly create file containing sender's private key.

```
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --import Shipr
gpg: can't open 'Shipr': No such file or directory
gpg: Total number processed: 0
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --export -a Amisha>Cns.txt
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --export -a Amisha>Cns1.txt
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --import Cns1.txt
gpg: key 077714E47AE77D5A: "amisha <shahiamisha808@gmail.com>" not changed
gpg: Total number processed: 1
gpg:              unchanged: 1
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --list-keys
/home/lab1006/.gnupg/pubring.kbx
--------------------------------
pub   rsa3072 2023-09-26 [SC] [expires: 2025-09-25]
      E0A259E4CFED0871ECAFF2F461DE05010FABA03E
uid           [ultimate] Shipr <rajeshwarimahapatra5@gmail.com>
sub   rsa3072 2023-09-26 [E] [expires: 2025-09-25]

pub   rsa3072 2023-09-26 [SC] [expires: 2025-09-25]
      CD17B015BFD1D9B3A12C872E077714E47AE77D5A
uid           [ultimate] amisha <shahiamisha808@gmail.com>
sub   rsa3072 2023-09-26 [E] [expires: 2025-09-25]
```

Step 4: You can create a fingerprint of key using the command
Step 5: Sender needs to add in his public key ring, the public key of receiver

```
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --list-keys
/home/lab1006/.gnupg/pubring.kbx
--------------------------------
pub   rsa3072 2023-09-26 [SC] [expires: 2025-09-25]
      E0A259E4CFED0871ECAFF2F461DE05010FABA03E
uid           [ultimate] Shipr <rajeshwarimahapatra5@gmail.com>
sub   rsa3072 2023-09-26 [E] [expires: 2025-09-25]

pub   rsa3072 2023-09-26 [SC] [expires: 2025-09-25]
      CD17B015BFD1D9B3A12C872E077714E47AE77D5A
uid           [ultimate] amisha <shahiamisha808@gmail.com>
sub   rsa3072 2023-09-26 [E] [expires: 2025-09-25]

lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --list-keys shahiamisha808@gmail.com
pub   rsa3072 2023-09-26 [SC] [expires: 2025-09-25]
      CD17B015BFD1D9B3A12C872E077714E47AE77D5A
uid           [ultimate] amisha <shahiamisha808@gmail.com>
sub   rsa3072 2023-09-26 [E] [expires: 2025-09-25]
```

Step 6: Listing public keys in keyring
Step 7: Snder can sign the public key of receiver using command

```
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --sign-key shahiamisha808@gmail.com

sec  rsa3072/077714E47AE77D5A
     created: 2023-09-26  expires: 2025-09-25  usage: SC
     trust: ultimate      validity: ultimate
ssb  rsa3072/127566F988B5CC04
     created: 2023-09-26  expires: 2025-09-25  usage: E
[ultimate] (1). amisha <shahiamisha808@gmail.com>


sec  rsa3072/077714E47AE77D5A
     created: 2023-09-26  expires: 2025-09-25  usage: SC
     trust: ultimate      validity: ultimate
 Primary key fingerprint: CD17 B015 BFD1 D9B3 A12C  872E 0777 14E4 7AE7 7D5A

     amisha <shahiamisha808@gmail.com>

This key is due to expire on 2025-09-25.
Are you sure that you want to sign this key with your
key "Shipr <rajeshwarimahapatra5@gmail.com>" (61DE05010FABA03E)

Really sign? (y/N) y
```

Step 8: Encrypt the data to send. (create a file beforehand to be encrypted)
Step 9: Decrypt the file

```
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --encrypt -r shahiamisha808@gmail.com Cns1.txt
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   2  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 2u
gpg: next trustdb check due at 2025-09-25
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --encrypt --sign --armor -r shahiamisha808@gmail.com Cns1.txt
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg -o myfiledecrypted -d Cns1.txt.gpg
gpg: encrypted with 3072-bit RSA key, ID 127566F988B5CC04, created 2023-09-26
      "amisha <shahiamisha808@gmail.com>"
```

Lab Outcome attained: LO6
Conclusion: Demonstrated the network security system using open source tools

Experiment No 11

Name: Shipra Suvarna
Roll No: T23-133

Aim : Installing Snort, configuring in Intrusion Detection Mode and writing rules for detecting piging activity .

Lab Outcome :

LO6 : Demonstrate the network security system using open source tools.

Theory :

1. What is Intrusion Detection System?

An Intrusion Detection System (IDS) is a critical cybersecurity tool designed to monitor network traffic, system activities, and user behavior to detect and respond to unauthorized or malicious activities. It serves as a proactive defense mechanism against cyber threats by identifying anomalies, patterns, and signs of potential attacks within a computer network or system. IDS operates by analyzing data in real-time, comparing it against predefined signatures, behavioral baselines, or anomaly detection algorithms.

IDS can be categorized into two main types: Network-based IDS (NIDS) and Host-based IDS (HIDS). NIDS monitors network traffic at various points within the network, identifying signs of intrusion or malicious activity that might bypass perimeter defenses. On the other hand, HIDS focuses on individual hosts or endpoints, analyzing system logs, files, and activities for any signs of compromise.

The primary goal of an IDS is to provide early detection of cyber threats, mitigate the risk of security breaches, and enable rapid response to incidents. By generating alerts or alarms when suspicious behavior is identified, IDS empowers network administrators and security teams to take appropriate action and protect the integrity, confidentiality, and availability of critical systems and data.

2. What are different modes in which Snort works? (refer user mannual on snort.org for this)?

Snort, a widely used open-source Intrusion Detection System (IDS), operates in several distinct modes, each catering to specific monitoring and analysis needs. As outlined in the official Snort user manual available on snort.org, these modes offer flexibility and versatility in network security:

1. Sniffer Mode: In this mode, Snort behaves like a packet sniffer, capturing and displaying network traffic in real-time. It is valuable for troubleshooting network issues and gaining insights into the flow of data. However, it doesn't involve analysis or rule-based detection.

2. Packet Logger Mode: In this mode, Snort logs captured packets to disk for later analysis. It's useful for preserving evidence and performing forensic investigations after potential security incidents.

3. Network Intrusion Detection Mode: This is the primary and most crucial mode of Snort. In this mode, Snort analyzes network traffic against a set of predefined rules and signatures. If it detects any patterns or behaviors that match the specified rules, it generates alerts to notify administrators of potential security threats. This mode enables real-time detection and response to unauthorized or malicious activities.

4. Network Intrusion Prevention Mode: This advanced mode not only detects but also actively prevents identified attacks by blocking or mitigating suspicious traffic. It involves inline deployment and requires careful configuration to avoid disrupting legitimate network communication.

Each of these modes addresses different use cases and security requirements, making Snort a versatile tool that can adapt to various monitoring and protection needs. By offering multiple modes of operation, Snort empowers security professionals to choose the mode that aligns best with their specific goals and helps enhance the overall security posture of their networks.

3. Write the commands used for installing snort, editing its configuration file and configurig it in intrusion detection mode ?

Here are the commands you need to follow for installing Snort, editing its configuration file, and configuring it in intrusion detection mode as per the instructions provided:

1. Check the interface name: ```shell ifconfig
```

2. Install Snort: ```shell sudo apt-get update sudo apt-get install snort
```

3. During installation, specify the interface name observed in step 1 when asked.

4. Edit Snort configuration file: ```shell sudo gedit /etc/snort/snort.conf
```
```

5. Make the following changes in the configuration file: - In section 1, set:
   ```
   ipvar HOME_NET 192.168.44.0/24
   ```
6. Open a new terminal and open the `ftp.rules` file (optional): ```shell
sudo gedit /etc/snort/rules/ftp.rules
```

7. In a new terminal, validate rules: ```shell
sudo snort -T -c /etc/snort/snort.conf -i ens33
```

8. Start Snort in NIDS mode: ```shell
sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i ens33 ```

9. On Kali Linux, run port scanning: ```shell
nmap 192.168.44.128
```

10. Observe the output in the Snort terminal.

11. Ping the Ubuntu machine from Kali Linux:
    ```shell
ping 192.168.44.128
```

12. Adding a rule for detecting ping activity:
a. Create a local.rules file: ```shell
  sudo gedit /etc/snort/rules/local.rules
  ```
b. Write the rule in local.rules:
  ```
  alert icmp any any -> $HOME_NET any (msg:"ICMP test detected"; GID:1; sid:10000001; rev:001; classtype:icmp-event;)
  ```
c. Save the local.rules file.

d. Comment the following lines in snort.conf:
  ```
  include $RULE_PATH/icmp.rules
  include $RULE_PATH/icmp-info.rules
  ```

e. Include the local.rules file in the configuration:
```
include $RULE_PATH/local.rules
```

f. Validate changes in snort.conf: ```shell
sudo snort -T -c /etc/snort/snort.conf -i ens33
```

g. Start Snort in Intrusion Detection Mode: ```shell
sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i ens33
```

h. Ping the Ubuntu machine from Kali and observe alerts.

i. Compare alerts generated using different rules.

Ensure to follow each step carefully and observe the outputs as instructed to effectively configure and use Snort in intrusion detection mode.

Conclusion:

In conclusion, this assignment involved the installation and configuration of Snort, a powerful Intrusion Detection System. By following the step-by-step instructions, we successfully installed Snort, edited its configuration file, and executed rules to detect ICMP activities. This hands-on experience enhanced our understanding of network security and IDS functionality.

Lab Assignment 10

Name: Shipra Suvarna

Roll No: T23-133

Aim: To study and configure Firewalls using IP tables

Theory:

A firewall is a system designed to prevent unauthorized access to or from a private network. You can implement a firewall in either hardware or software form, or a combination of both. Generally the firewall has two network interfaces: one for the external side of the network, one for the internal side. Its purpose is to control what traffic is allowed to traverse from one side to the other. As the most basic level, firewalls can block traffic intended for particular IP addresses or server ports.

TCP network traffic moves around a network in packets, which are containers that consist of a packet header—this contains control information such as source and destination addresses, and packet sequence information—and the data (also known as a payload). While the control information in each packet helps to ensure that its associated data gets delivered properly, the elements it contains also provides firewalls a variety of ways to match packets against firewall rules.

Three basic types of network firewalls: packet filtering (stateless), stateful, and application layer.

Packet filtering, or stateless, firewalls work by inspecting individual packets in isolation. As such, they are unaware of connection state and can only allow or deny packets based on individual packet headers.

Stateful firewalls are able to determine the connection state of packets, which makes them much more flexible than stateless firewalls. They work by collecting related packets until the connection state can be determined before any firewall rules are applied to the traffic.

Application firewalls go one step further by analyzing the data being transmitted, which allows network traffic to be matched against firewall rules that are specific to individual services or applications. These are also known as proxy-based firewalls.

Basic of iptables

The rules in IPTables are:

1.Those packets entering your machine that are destined for your machine. (INPUT)

2.Those packets leaving your machine. (OUTPUT)

3.Those packets entering your machine, but are destined for another machine and will pass through your machine (FORWARD).

In Iptables, these scenarios are referred to as INPUT, OUTPUT, and FORWARD, respectively.

Once the traffic type has been specified, three actions may be taken:

1.ACCEPT allows packets to pass through the firewall.

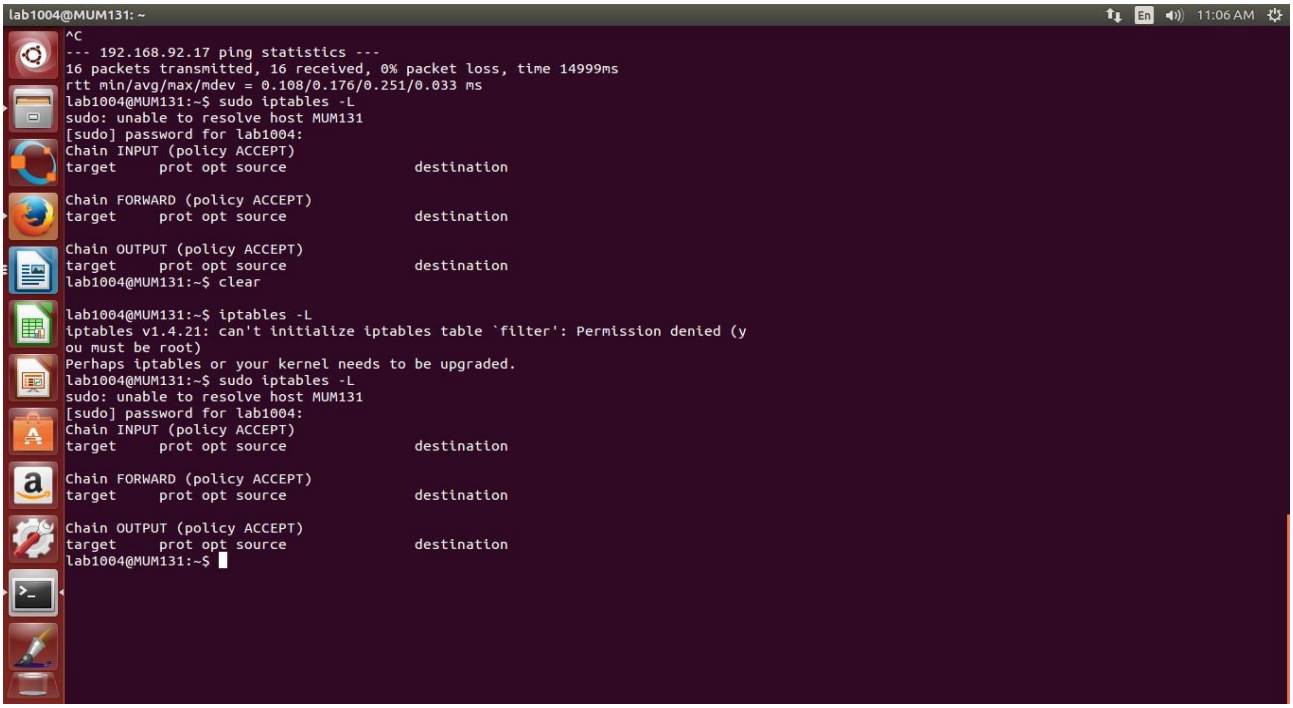
2.DROP ignores the packet and sends no response to the request.

3.REJECT ignores the packet, but responds to the request with a packet denied message.

Basic Commands

sudo iptables -L

(-L - List the current filter rules.)



Basic Iptables Options

- -A - Append this rule to a rule chain. Valid chains for what we're doing are INPUT, FORWARD and OUTPUT, but we mostly deal with INPUT in this tutorial, which affects only incoming traffic.
- -p - The connection protocol used.
- --dport - The destination port(s) required for this rule. A single port may be given, or a range may be given as start:end, which will match all ports from start to end, inclusive.
- -j - Jump to the specified target. By default, iptables allows four targets:
- ACCEPT - Accept the packet and stop processing rules in this chain.
- REJECT - Reject the packet and notify the sender that we did so, and stop processing rules in this chain.
- DROP - Silently ignore the packet, and stop processing rules in this chain.
- LOG - Log the packet, and continue processing more rules in this chain.
  Allows the use of the --log-prefix and --log-level options.
- -i - Only match if the packet is coming in on the specified interface.

- -I - Inserts a rule. Takes two options, the chain to insert the rule into, and the rule number it should be.
- -I INPUT 5 would insert the rule into the INPUT chain and make it the 5th rule in the list.
- -v - Display more information in the output. Useful for if you have rules that look similar without using -v.
- -s --source - address[/mask] source specification
- -d --destination - address[/mask] destination specification
- -o --out-interface - output name[+] network interface name ([+] for wildcard)

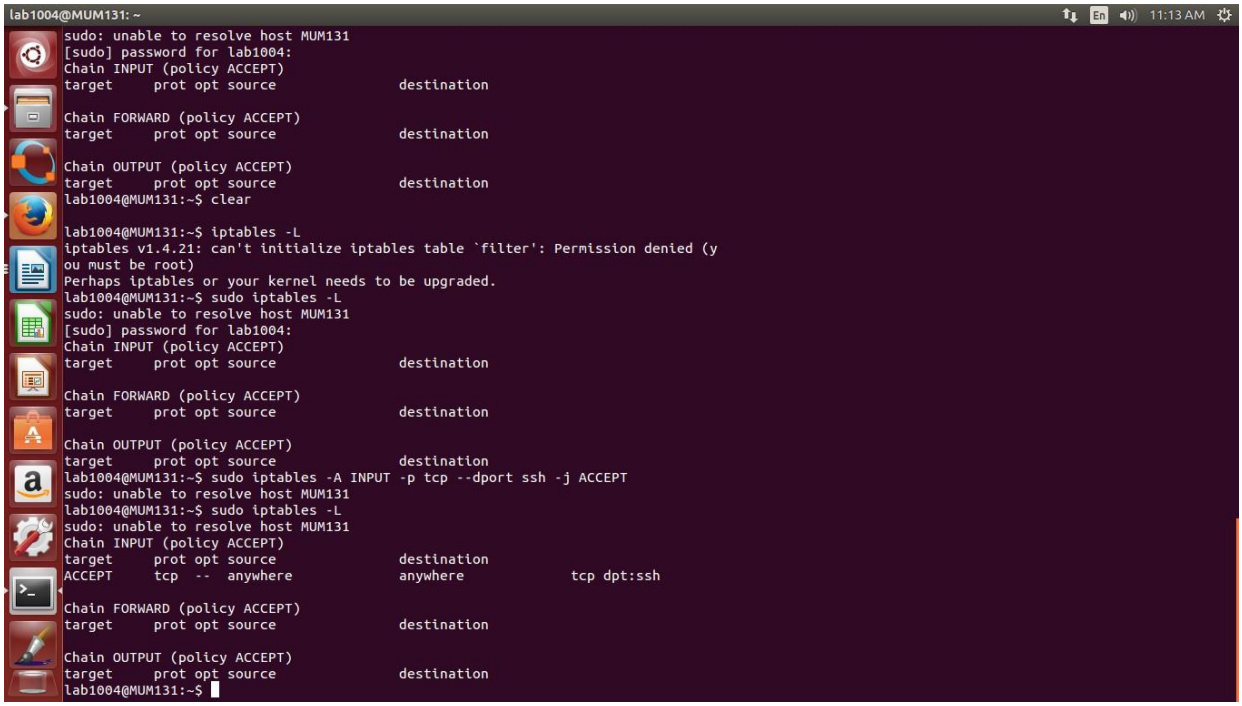
Allowing Incoming Traffic on Specific Ports

To allow incoming traffic on the default SSH port (22), you could tell iptables to allow all TCP traffic on that port to come in.

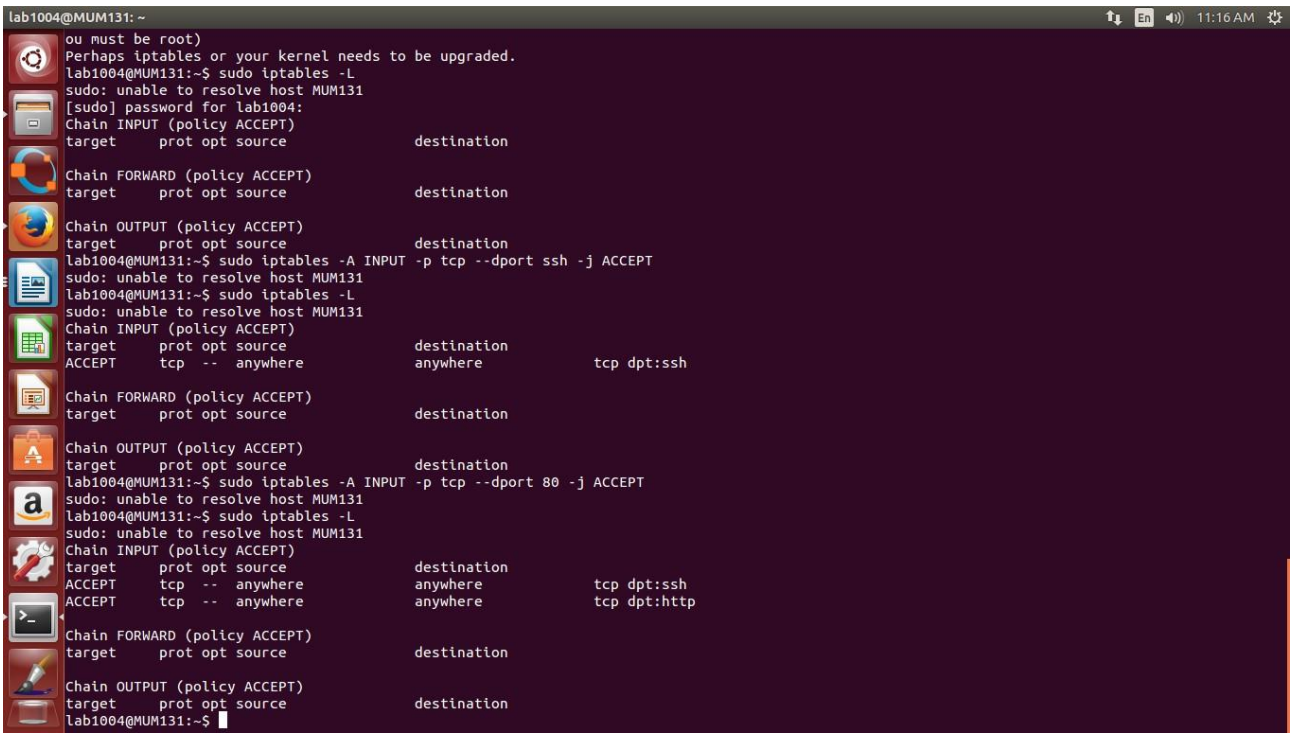
sudo iptables -A INPUT -p tcp --dport ssh -j ACCEPT

Referring back to the list above, you can see that this tells iptables:

- append this rule to the input chain (-A INPUT) so we look at incoming traffic
- check to see if it is TCP (-p tcp).
- if so, check to see if the input goes to the SSH port (--dport ssh).
- if so, accept the input (-j ACCEPT).



sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT

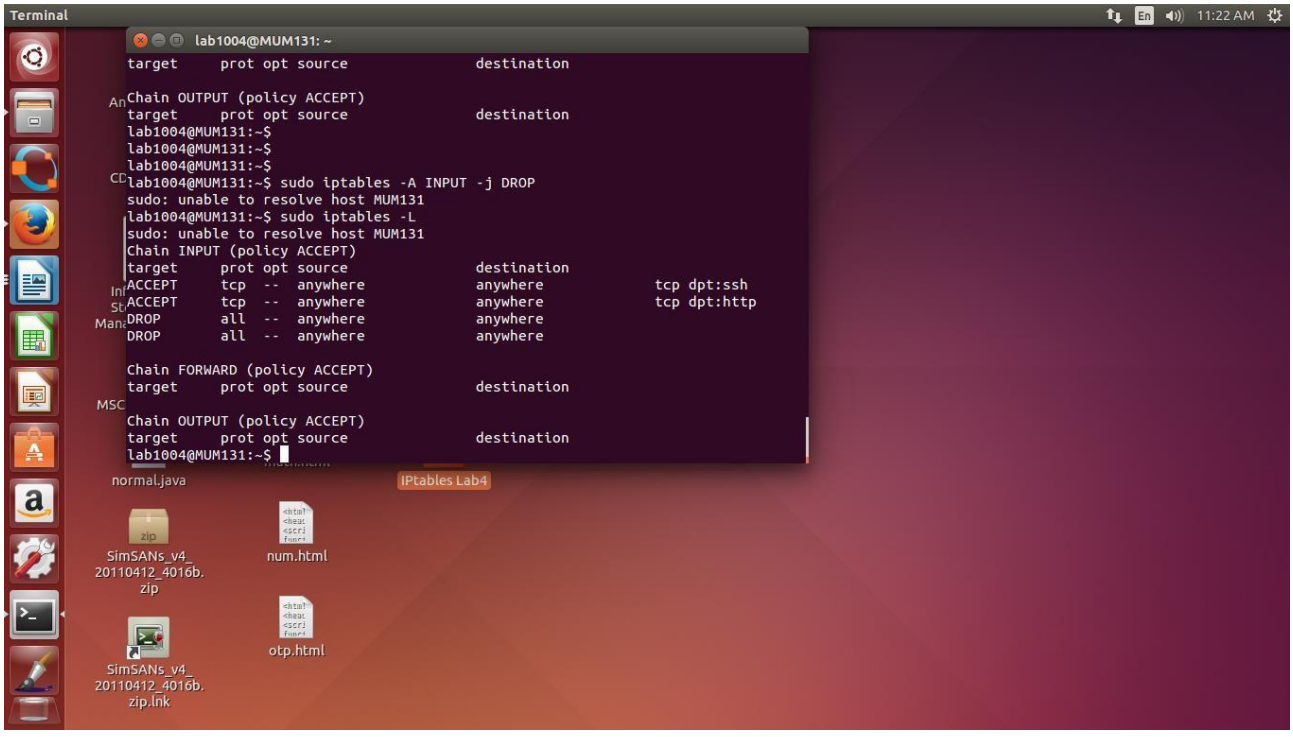


Blocking Traffic

Once a decision is made to accept a packet, no more rules affect it. As our rules allowing ssh and web traffic come first, as long as our rule to block all traffic comes after them, we can still accept the traffic we want. All we need to do is put the rule to block all traffic at the end.

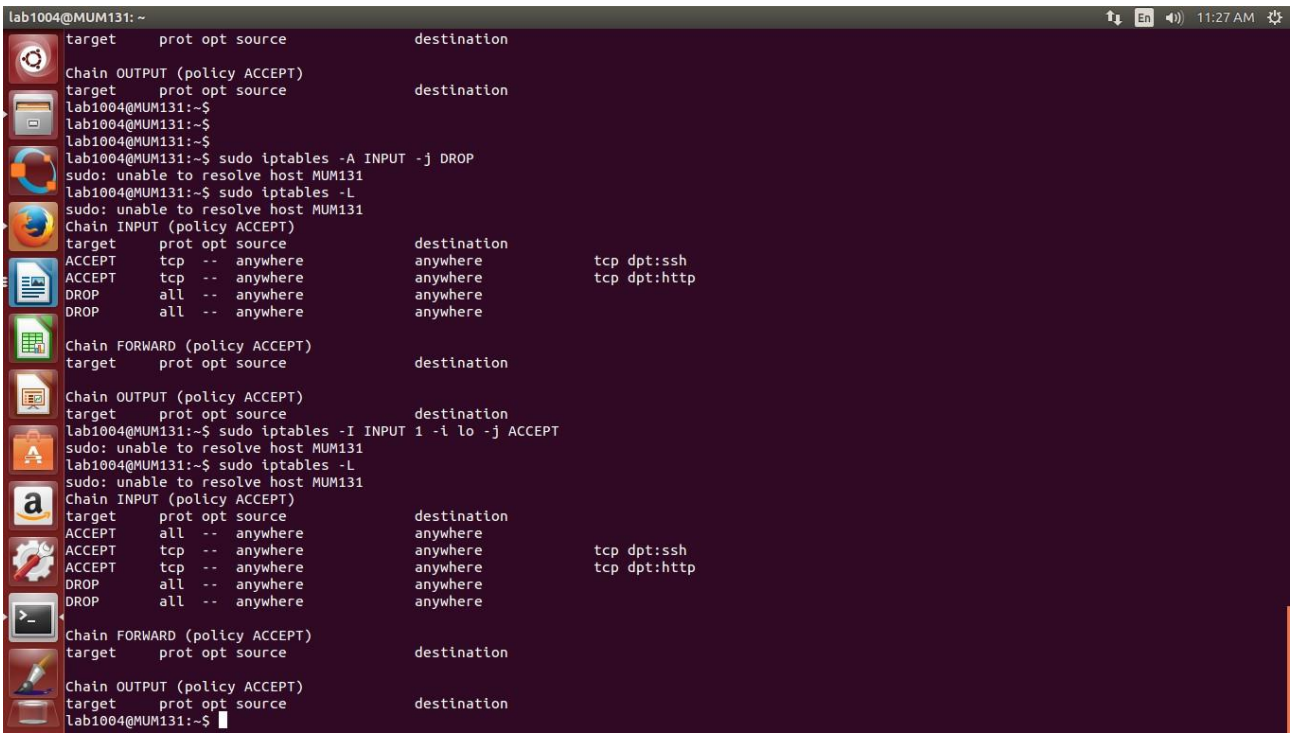
sudo iptables -A INPUT -j DROP

sudo iptables -L



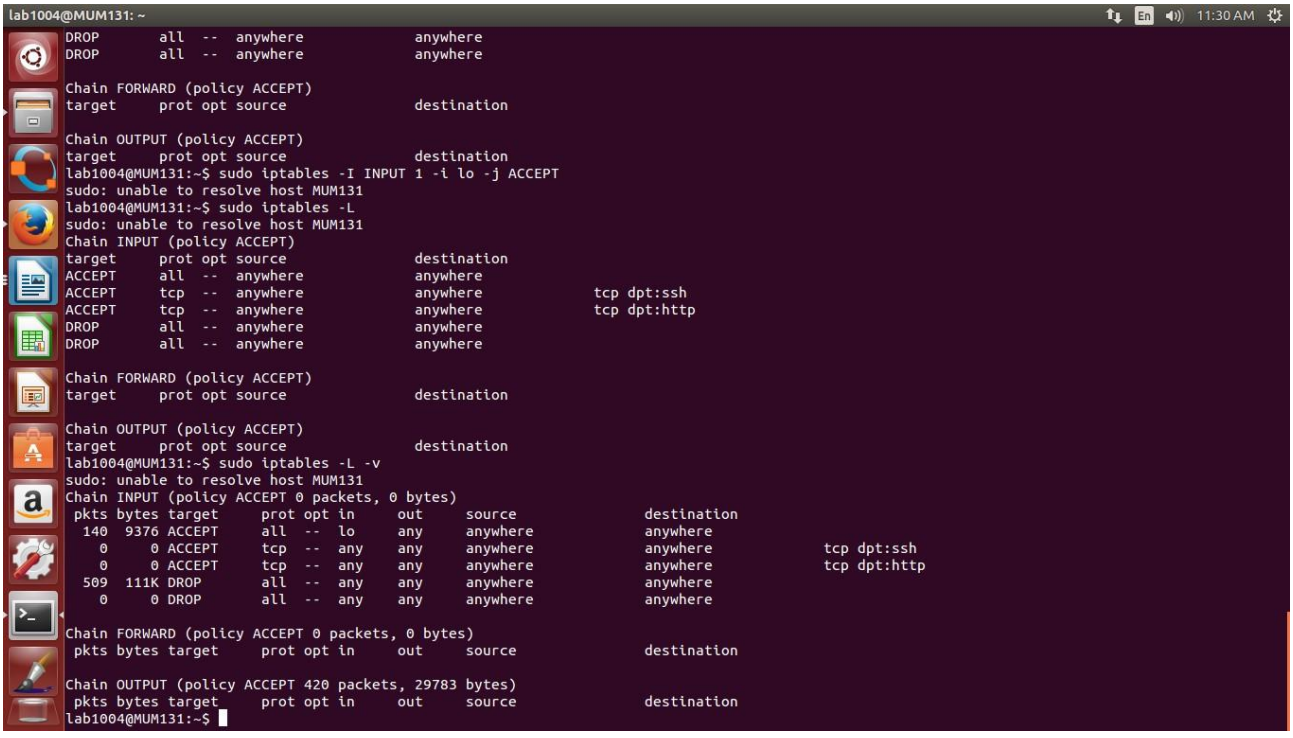
Editing iptables

sudo iptables -I INPUT 1 -i lo -j ACCEPT sudo iptables -L



we will list iptables in greater detail.

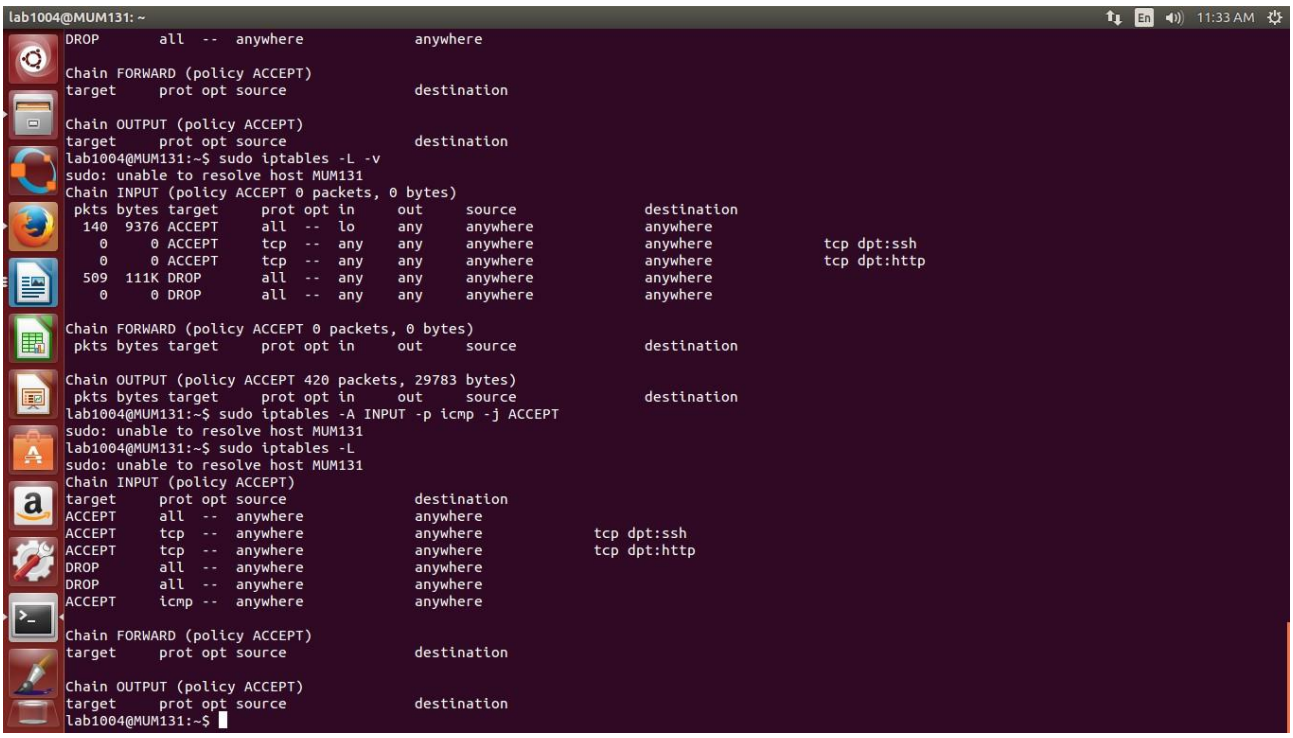
sudo iptables -L -v



Allow traffic on ICMP port sudo iptables -

A INPUT -p icmp -j ACCEPT now list the

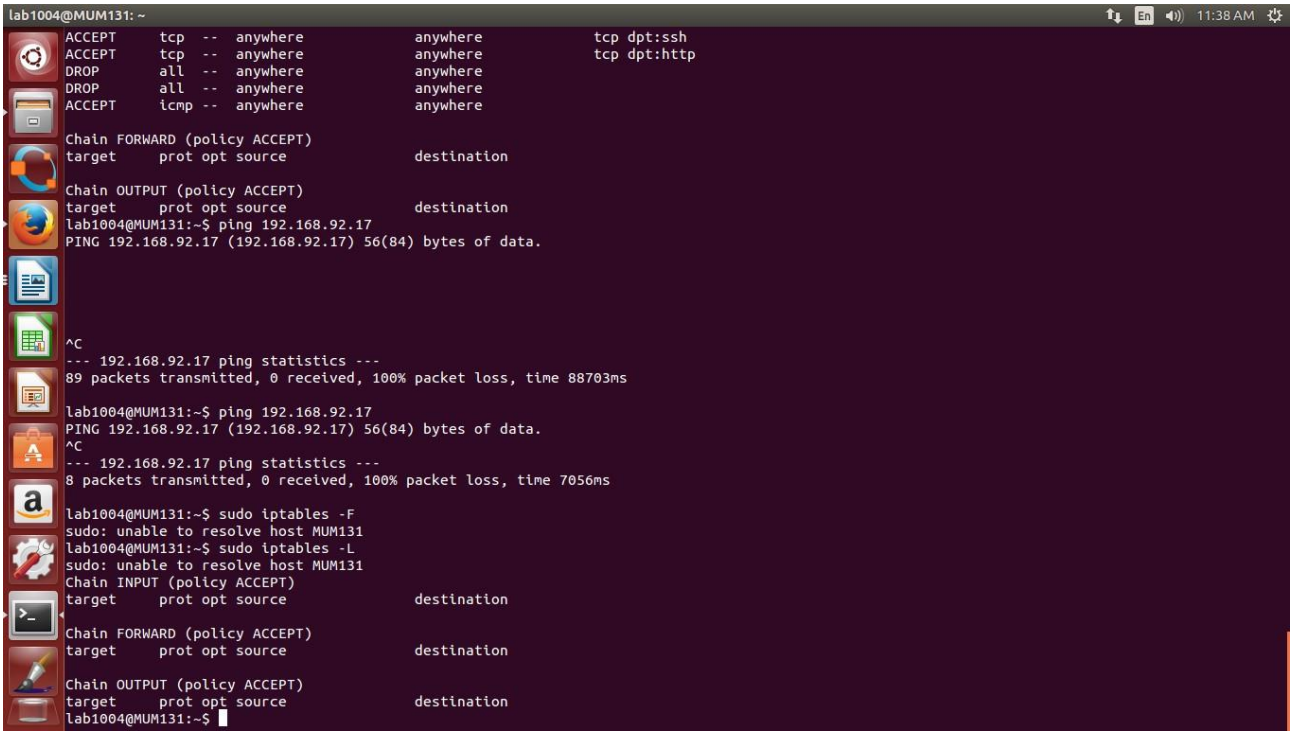
rules again.. sudo iptables -L



clearing all rules

sudo iptables -F sudo

iptables -L



Dropping icmp packets try to ping ur neighbour machine ping 192.168.92.17

u can see the response packets received.

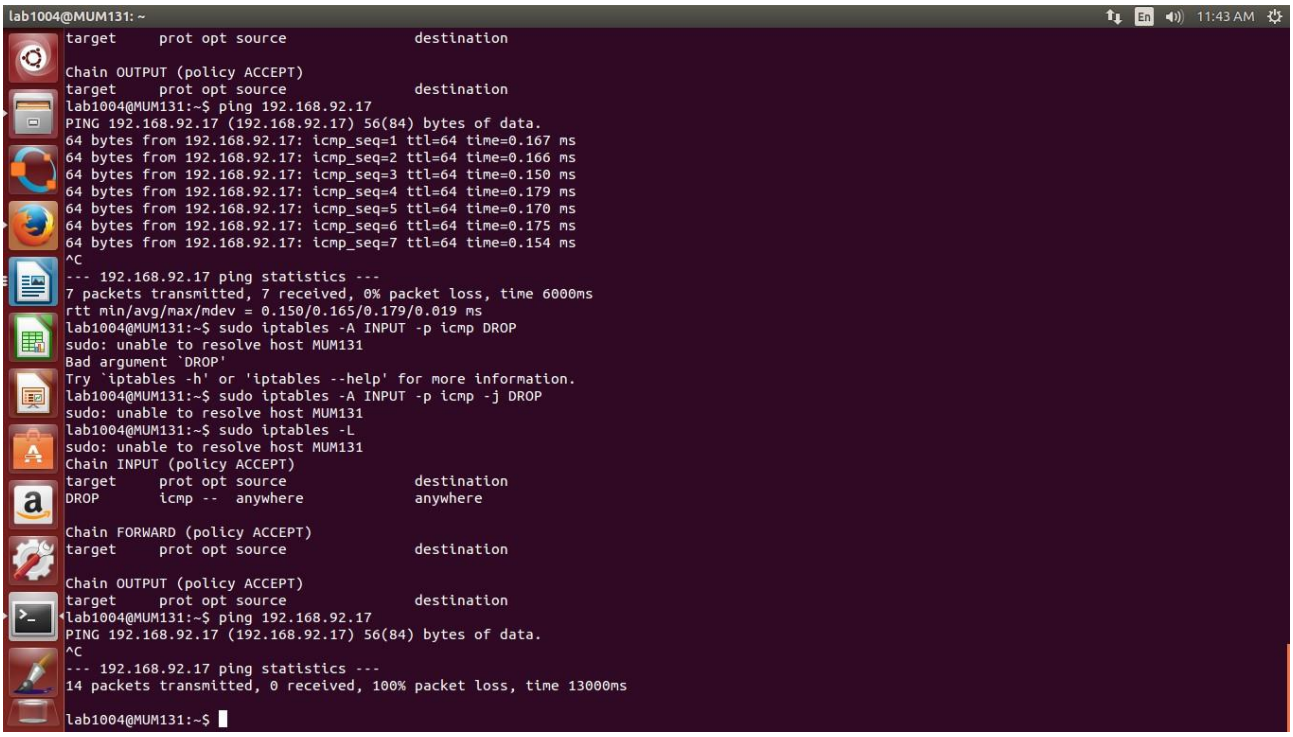
Now block incoming icmp packets from the neighbour using command:

sudo iptables -A **INPUT** -p icmp -j DROP list the rule: sudo iptables -L

try to ping ur neighbour machine again ping

192.168.92.17 u can not see receive icmp echo

reply packets..



Now try to restrict outgoing icmp packets by adding rule sudo

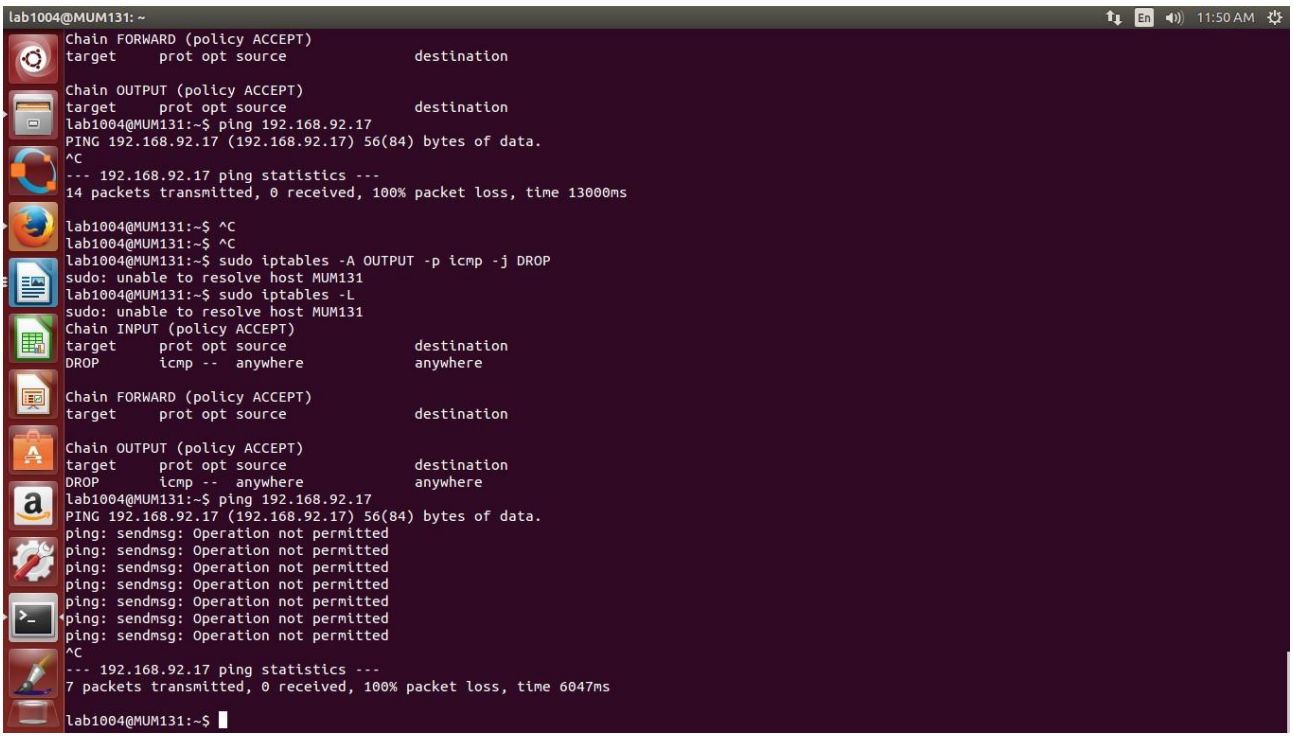
iptables -A **OUTPUT** -p icmp -j DROP

List the rule: sudo iptables

-L now try to ping

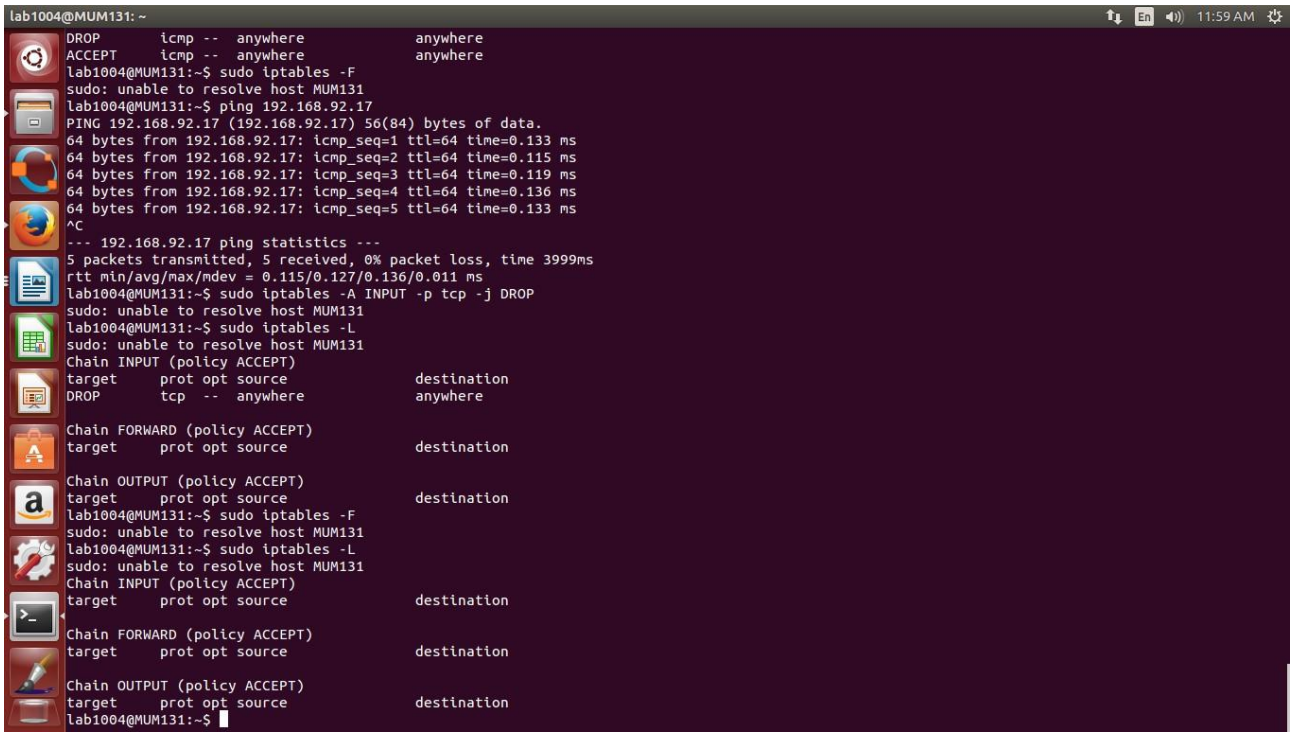
neighbour ping

192.168.92.17



Blocking TCP port traffic will not allow u to browse the Internet

sudo iptables -A INPUT -p tcp -j DROP List the rule: sudo iptables -L



Blocking ICMP packets from specific source machine:

sudo iptables -A INPUT -s 192.168.92.17 -p icmp -j DROP

sudo iptables -L ping 192.168.92.17

---does not allow (192.168.92.17 can not send u icmp packets)

ping any other machine: ping 192.168.92.11

---allowed (192.168.92.11 can send u icmp packets)

Types of iptables:

I. IPTABLES TABLES and CHAINS

IPTables has the following 4 built-in tables.

1. Filter Table

Filter is default table for iptables. So, if you don't define you own table, you'll be using filter table. Iptables's filter table has the following built-in chains.

- INPUT chain – Incoming to firewall. For packets coming to the local server.
- OUTPUT chain – Outgoing from firewall. For packets generated locally and going out of the local server.
- FORWARD chain – Packet for another NIC on the local server. For packets routed through the local server.

  Type the following command and see the result sudo

  iptables -t filter -L

2. NAT table

Iptable's NAT table has the following built-in chains.

- PREROUTING chain – Alters packets before routing. i.e Packet translation happens immediately after the packet comes to the system (and before routing). This helps to translate the destination ip address of the packets to something that matches the routing on the local server. This is used for DNAT (destination NAT).
- POSTROUTING chain – Alters packets after routing. i.e Packet translation happens when the packets are leaving the system. This helps to translate the source ip address of the packets to something that might match the routing on the desintation server. This is used for SNAT (source NAT).
- OUTPUT chain – NAT for locally generated packets on the firewall. Type the following command and see the result sudo iptables -t nat -L

3. Mangle table

Iptables's Mangle table is for specialized packet alteration. This alters QOS bits in the TCP header. Mangle table has the following built-in chains.

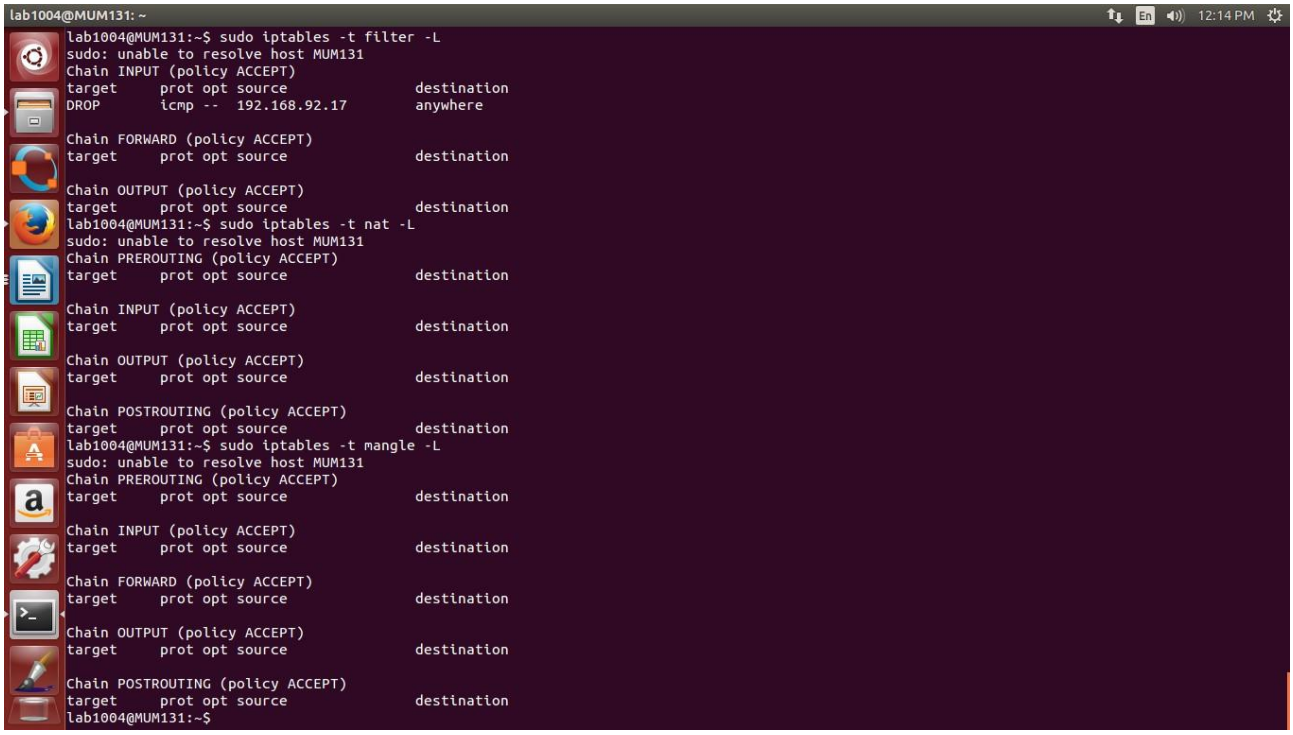
- PREROUTING chain
- OUTPUT chain
- FORWARD chain
- INPUT chain
- POSTROUTING chain

  Type the following command and see the result
  sudo iptables -t nat -L

4. Raw table

Iptable's Raw table is for configuration exemptions. Raw table has the following built-in chains.

- PREROUTING chain
- OUTPUT chain



Lab Outcome: LO6

Conclusion: Demonstrated the network security system using open source tools

Security Lab Theory Assignment 2

Name: Shipra Suvarna

Roll No: T23-133

Q] What is Intrusion Detection System? Explain different types of intrusion detection systems with their working. State the advantages and limitations of each.

An **Intrusion Detection System (IDS)** is a critical cybersecurity tool that monitors a network or system for suspicious or malicious activities and helps detect and respond to security incidents. IDSs work by analyzing network traffic or system behavior and comparing it against predefined signatures or behavioral patterns to identify potential intrusions or security threats. There are different types of IDSs, each with its own working principles, advantages, and limitations:

1. Network-Based Intrusion Detection System (NIDS):

Working: NIDS operates at the network level, analyzing network traffic as it flows through the network. It examines data packets for specific signatures or patterns that match known attack patterns or behaviors. Signature-based NIDS uses a database of predefined attack signatures, while anomaly-based NIDS establishes a baseline of normal network behavior and generates alerts when deviations occur.

Advantages: NIDS can monitor all network traffic, making it suitable for large networks. It is effective at detecting known attacks and can provide real-time alerts. It is non-intrusive, meaning it does not actively interfere with network traffic.

Limitations: NIDS may struggle to detect new or sophisticated attacks that don't match known signatures. False positives can be common, especially in anomaly-based NIDS, and analyzing encrypted traffic can be challenging.

2. Host-Based Intrusion Detection System (HIDS):

Working: HIDS is installed on individual host machines and focuses on monitoring the internal state of the hosts. It examines system logs, file integrity, and system configuration for signs of unauthorized access, tampering, or malicious activity.

Advantages: HIDS provides a deeper level of insight into the security of individual host systems. It can detect attacks that occur at the host level, such as malware installations and insider threats. HIDS can also help with compliance monitoring.

Limitations: HIDS may not be as effective at detecting network-based attacks that don't directly impact the host's internal state. Managing and maintaining agents on each host can be resource-intensive, and the IDS may generate a large volume of logs.

3. Behavior-Based Intrusion Detection System:

Working: Behavior-based IDSs establish a baseline of normal behavior for a network or system. They continuously monitor and analyze network traffic or system activity and trigger alerts when deviations from the baseline occur. These systems do not rely on specific attack signatures but instead focus on patterns of behavior.

Advantages: Behavior-based IDSs are effective at detecting novel or zero-day attacks since they don't rely on known signatures. They can also identify insider threats and unusual activities that might not trigger traditional IDS alerts.

Limitations: One of the main limitations is the potential for false positives if the baseline is not accurately tuned or if legitimate changes in system or network behavior occur. Setting up an accurate baseline can be time-consuming.

4. Anomaly-Based Intrusion Detection System:

Working: Anomaly-based IDSs also rely on establishing a baseline of normal behavior. They use statistical analysis or machine learning algorithms to detect deviations from this baseline, signaling potential intrusions.

Advantages: Anomaly-based IDSs are effective at detecting new and previously unknown threats. They can adapt to changing network environments and are less reliant on specific attack signatures.

Limitations: Similar to behavior-based IDSs, they can produce false positives if the baseline is not properly calibrated or if there are legitimate changes in network or system behavior. Complex attacks that mimic normal behavior may also go undetected.

5. Hybrid Intrusion Detection System:

Working: Hybrid IDS combines multiple detection methods, such as signature-based, anomaly-based, and behavior-based techniques, to improve accuracy and reduce false positives. It attempts to provide a more comprehensive approach to intrusion detection.

Advantages: Hybrid IDSs offer a balanced detection rate by leveraging the strengths of different detection methods. They are better equipped to handle a wide range of threats.

Limitations: The complexity and resource requirements of hybrid IDSs may be higher compared to single-method IDSs. Properly configuring and managing hybrid systems can be challenging.

Security Lab Theory Assignment1

Name: Shipra Suvarna

Roll No: T23-133

Q) Explain the padding scheme used in RSA. Why it is used? What is its limitation?

The padding scheme used in RSA encryption, particularly in the context of PKCS #1 v1.5, is employed to address certain security and practical concerns when using the RSA algorithm.

Why it is used:

- Security: The primary reason for using padding in RSA encryption is to enhance security. Without padding, encrypting the same plaintext with the same public key would result in identical ciphertexts. Attackers could exploit this to gain information about the plaintext, leading to a security risk. Padding ensures that each encryption of the same message produces a different ciphertext, making it more difficult for attackers to identify patterns or gain insights into the plaintext.
- Textual Data: RSA encryption was originally designed for mathematical values, not for arbitrary binary data or text. Padding allows for the encryption of text and other types of data, ensuring that the data can be correctly processed by the RSA algorithm.

Limitation:

While padding in RSA provides security benefits, it also has limitations and potential vulnerabilities:

- Padding Oracle Attacks: One limitation is the susceptibility to padding oracle attacks. In certain situations, an attacker may gain information about the plaintext by interacting with a system that decrypts RSA ciphertexts and provides feedback on the padding validity. Such attacks can lead to the recovery of the original message, compromising security.
- Deterministic Padding: PKCS #1 v1.5 padding, while providing randomness through the inclusion of a random byte sequence, is still somewhat deterministic in nature. This means that if an attacker can guess the padding, they might be able to recover some or all of the plaintext.
- Dependence on Implementation: The security of the padding scheme can also depend on its correct implementation. If there are implementation flaws or vulnerabilities, it can weaken the security of the RSA encryption.

To address these limitations, more advanced padding schemes like OAEP (Optimal Asymmetric Encryption Padding) have been developed. OAEP provides stronger security guarantees and is considered more robust against certain types of attacks compared to PKCS #1 v1.5 padding. Therefore, the choice of padding scheme in RSA encryption depends on the specific security requirements and considerations of the application.

In the video, the RSA PKCS #1 v1.5 padding scheme is introduced to address the issue of identical ciphertexts when encrypting the same message multiple times using RSA encryption. This padding scheme involves appending a random number "r" to the message before encryption, ensuring that each encryption operation produces a distinct ciphertext.

Key points from the video:

- Padding Requirements: The PKCS #1 v1.5 padding scheme has specific requirements, such as the use of "0" and "0x02" bytes in the padding, as well as the avoidance of zeros in the random number "r." Padding Format: An example is provided where the message to be encrypted is "1." The resulting padding format includes "00," "02," the random number, and "00." This format is designed to obscure the original message, making it challenging to deduce the plaintext from the ciphertext.
- Size Consideration: It's mentioned that the size of the original message "m" should not be too large to accommodate the padding properly.
- Encoding and Decoding: The video explains the encoding and decoding process in the PKCS #1 v1.5 padding scheme. Encoding involves adding a zero prefix to the encoded message and then applying the RSA function. Decoding reverses this process, searching for specific byte patterns to remove random bytes.
- Security Weaknesses: The video hints at fundamental weaknesses in this padding scheme, which are to be discussed later. These weaknesses likely include vulnerabilities to chosen plaintext attacks, which can compromise security.