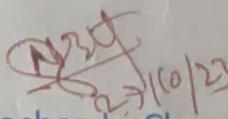


Thadomal Shahani Engineering College
Bandra (W.), Mumbai- 400 050.

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


Teacher In-Charge

Head of the Department

Date 27/10/23

Principal

CONTENTS

S.R. NO.	EXPERIMENTS	PAGE NO.	DATE	TEACHERS SIGN.
1	Breaking shift cipher & mono alphabetic substitution cipher using frequency analysis		18/7/23	
2	Cryptanalysis or decoding of poly-alphabetic cipher - playfair, vigenere		25/7/23	
3	Block cipher modes of operation using advanced encryption system (AES)		2/8/23	
4	Implementation & analysis of RSA cryptosystem & digital signatures 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 Nmap, armitage etc		29/8/23	✓
7	Study of packet sniffer tools like wireshark & TCPdump		12/9/23	✓
8	Installation of NMAP using it with diff options to scan, open & ports		28/9/23	
9	Simulate DOS attack using Hping3			
10	Study & configure firewalls 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

What's New x Virtual Labs x how to take screenshot in windows | +

cse29-iith.labs.ac.in/exp/shift-cipher/simulation.html

Virtual LABS
An IITM Govt. Sponsored Project

Breaking the Shift Cipher

Do your rough work here:

PART III

Plaintext:
attack at dawn

shift: 7 ▾

v Encrypt v ^ Decrypt ^

Ciphertext:
haahic ha khdu

PART IV

Enter your solution Plaintext and shift key here:

attack at dawn

Key 7 ▾

Check my answer!

CORRECT!!

Type here to search

27°C Cloudy

11:53 PM

7/18/2023

2. Mono alphabetic substitution cipher

What's New Virtual Labs how to take screenshot in windows New Tab Monoalphabetic Substitution Ci...

cse29-iith.vlabs.ac.in/exp/substitution-cipher/simulation.html

Virtual Labs

Breaking the Mono-alphabetic Substitution Cipher

PART I

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

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

a	b	c	d	e	f	g	h	i	j	k	l	m
8.167	1.49	2.782	4.233	12.702	2.228	2.015	6.094	6.966	0.153	0.772	4.025	2.406
n	o	p	q	r	s	t	u	v	w	x	y	z
6.749	7.507	1.929	0.095	5.987	6.327	9.056	2.758	0.978	2.360	0.150	1.974	0.074

sample illoes kofh a dkzeplid sfu zbhjlmfm If ej bi a -ufo, lajkpxbnuop =
zdgph a elgjor jex ljeqpxmipw yexzbjifc amib ufu zetjyjbj 31
exp ikppoxfj buxfjhs ipobis, ilddoktuxvrt exp bfarbwuojt jil pxodxdixp a
oikd, rxnlpk ipahufre shat, jex iajxkpohwep jxwz ambix jeoj lfx zbxu
ln jex dkzeplid hbww dagx exp jowwpf sfu jex ljexp zbxu hbww dagx exp
zeljxp, zex rpxagz inn jhl obvix npld jex dkzeplid. lfx zbxu dagxz

[Next Ciphertext](#)

[Calculate Frequencies in ciphertext](#)

Ciphertext Frequencies:

a	b	c	d	e	f	g	h	i	j	k	l	m
8.084	6.854	0.176	4.394	7.206	4.394	1.757	1.582	2.988	7.909	2.636	7.381	1.582
n	o	p	q	r	s	t	u	v	w	x	y	z
1.582	2.285	8.26	0.000	1.582	0.000	0.703	1.933	0.000	5.097	14.06	0.176	7.381

PART II

Type here to search

12:52 AM 7/19/2023

PART II

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

Scratchpad:

ALICE COMES UPON A MUSHROOM AND SITTING ON IT IS A BLUE CATERPILLAR SMOKING A HOOKAH. THE CATERPILLAR QUESTIONS ALICE AND SHE ADMITS TO HER CURRENT IDENTITY CRISIS, COMPOUNDED BY HER INABILITY TO REMEMBER A POEM, BEFORE CRAWLING AWAY. THE CATERPILLAR TELLS ALICE THAT ONE SIDE OF THE MUSHROOM WILL MAKE HER GROW AND THE OTHER SIDE MAKE HER SHRINK. SHE BREAKS OFF TWO PIECES FROM THE MUSHROOM. ONE SIDE MAKES HER SHRINK SMALLER THAN EVER, WHILE ANOTHER CAUSES HER NECK TO GROW HIGH INTO THE TREES, WHERE A PIGEON MISTAKES HER FOR A SERPENT. WITH SOME EFFORT, ALICE BRINGS HERSELF BACK TO HER USUAL HEIGHT. SHE STUMBLES UPON A SMALL ESTATE AND USES THE MUSHROOM TO REACH A MORE APPROPRIATE HEIGHT.

Modify the text above (in scratchpad):

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

Replace cipher character by plaintext character

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

Replace character by character

Your replacement history:

You replaced x by E You replaced a by T You replaced
T by Y You replaced b by I You replaced l by R You
replaced R by V You replaced j by A You replaced A by
j You replaced o by Y You replaced j by O You
replaced o by R You replaced x by E You replaced a by
T You replaced i by A You replaced n by Q You
replaced b by I You replaced v by N You replaced x
by E You replaced p by T You replaced a by A You
replaced j by O You replaced O by j You replaced T
by p You replaced j by T You replaced e by H You
replaced z by S You replaced o by R You replaced ; by

Windows taskbar: Type here to search, Start button, Task View, File Explorer, Mail, Taskbar icons, Google Chrome icon, System tray: 27°C Cloudy, 12:52 AM, 7/19/2023.

replaced z by S You replaced p by R You replaced i by
C You replaced l by O You replaced b by Y You
replaced k by U You replaced w by L You replaced n
by F You replaced d by M You replaced o by V You
replaced b by I You replaced r by B You replaced A by
N You replaced m by G You replaced u by D You
replaced g by K You replaced h by W You replaced t
by Y You replaced y by Q You replaced c by V

PART III

Enter your solution plaintext here:

HER SHRINK SMALLER THAN EVER, WHILE ANOTHER CAUSES HER NECK TO GROW HIGH INTO THE TREES, WHERE A PIGEON MISTAKES HER FOR A SERPENT, WITH SOME EFFORT, ALICE BRINGS HERSELF BACK TO HER USUAL HEIGHT. SHE STUMBLES UPON A SMALL ESTATE AND USES THE MUSHROOM TO REACH A MORE APPROPRIATE HEIGHT.

Solution Key =

This is not correct, Please try again!

PART IV

Plaintext

Windows taskbar: Type here to search, Start button, Task View, File Explorer, Mail, Taskbar icons, Google Chrome icon, System tray: 27°C Cloudy, 12:52 AM, 7/19/2023.

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 Vigenere in 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 Vigenere cipher 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 ($\text{mod } 26$) 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)

$$(0 + 24) \% 26 = 24 \text{ (Y)} \\ (22 + 10) \% 26 = 6 \text{ (G)} \\ (13 + 4) \% 26 = 17 \text{ (R)}$$

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

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

SECURITYABDFGHJKLMNPQVWXYZ

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

he110

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
- Colon Cipher
- Letters Bars
- Polux Cipher
- Bifid Cipher
- DCODE'S TOOLS LIST

Support

Paypal

Feedback

12:18 AM 7/20/2023

PlayFair Cipher - Online Decoder

dcode.fr/playfair-cipher

Search for a tool

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

SECURITYABDFGHJKLMNPQVWXYZ

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

he110

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
- Colon Cipher
- Letters Bars
- Polux Cipher
- Bifid Cipher
- DCODE'S TOOLS LIST

Support

Paypal

Patreon

More

Forum/Help

12:18 AM 7/20/2023

Vigenere Cipher - Online Decoder

Type here to search

VIGENÈRE CIPHER
Cryptography · Poly-Alphabetic Cipher · Vigenere Cipher

VIGENÈRE DECODER
XMBRKGGSIDI

PARAMETERS

- PLAINTEXT LANGUAGE: English
- ALPHABET: ABCDEFGHIJKLMNOPQRSTUVWXYZ

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
- VIGENÈRE CRYPTANALYSIS (KASISKI'S TEST)

AUTOMATIC DECRIPTION

DECRYPT

See also: Beaufort Cipher – Caesar Cipher

VIGENÈRE ENCODER
VIGENÈRE PLAIN TEXT: hello world

CIPHER KEY: VIG

ALPHABET: ABCDEFGHIJKLMNOPQRSTUVWXYZ

PRESERVE PUNCTUATION: ✓

ENCRYPT

See also: Beaufort Cipher – Autoclave Cipher – Caesar Cipher

Feedback

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

27°C Cloudy 12:20 AM 7/20/2023

PlayFair Cipher - Online Decoder

Type here to search

PLAYFAIR CIPHER
Cryptography · Polymographic Cipher · PlayFair Cipher

PLAYFAIR DECODER
fuouqp

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
SECURITY: A B D F G H K L M N O P Q V W X Z

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

BRUTEFORCE DECRYPTION ATTACK WITH THE GRID

WITHOUT KNOWING KEY

KNOWN PLAINTEXT: KNOWN PLAINTEXT ATTACK:

PLAYFAIR ENCODER
PLAYFAIR PLAIN TEXT: hello

Support

Forum / Help

Near record 12:18 AM 7/20/2023

The screenshot shows the dCode.fr website with the URL dcode.fr/vigenere-cipher. The main content is the "VIGENÈRE DECODER" tool. It has a search bar at the top with the query "cmrgw cjrzy". Below it are sections for "PARAMETERS" and "DECRYPTION METHOD". In the "PARAMETERS" section, "PLAINTEXT LANGUAGE" is set to "English" and "ALPHABET" is set to "ABCDEFGHIJKLMNOPQRSTUVWXYZ". Under "DECRYPTION METHOD", the radio button "KNOWING THE KEY/PASSWORD" is selected with the value "VIG". A "DECRYPT" button is present. To the right, there is a "Summary" sidebar with links to other cipher tools and articles. At the bottom, there are "Similar pages" links to Beaufort and Caesar cipher.

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:

The screenshot shows the Firefox Web Browser window with the URL <https://cse29-iiith.vlabs.ac.in/exp/pkcs/simulation.html>. The page title is "Public-Key Cryptosystems (PKCSv1.5)".

Plaintext (string):
test

Ciphertext (hex):
287bbf786676dc7cfcd9f2c21b929c8af047327f8075164a9569d484a415e3d1
ea0f499972b28f75fa42ba5278237e31a27db464f8b85970575a25753979d5

decrypt

Decrypted Plaintext (string):
test

Status:
Decryption Time: 7ms

RSA private key

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

Modulus (hex):
BC86E30C782C44EE756B874ACECF2A115E613021EAF1ED05EF2958EC2BED099D
26FE2EC896BF90E84FE381AF67A7B7C8B48085235E72AB595ABF8FE84805F8D8

Public exponent (hex, F4=0x10001):
3

Private exponent (hex):
7da f4292fac82d9f44e47af87348a1c0b9440cac1474bf394a1b929d729e5bbc

The screenshot shows the Firefox Web Browser window with the same URL and title.

Decryption Time: 7ms

RSA private key

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

Modulus (hex):
BC86E30C782C44EE756B874ACECF2A115E613021EAF1ED05EF2958EC2BED099D
26FE2EC896BF90E84FE381AF67A7B7C8B48085235E72AB595ABF8FE84805F8D8

Public exponent (hex, F4=0x10001):
3

Private exponent (hex):
7da f4292fac82d9f44e47af87348a1c0b9440cac1474bf394a1b929d729e5bbc
f402f29a9300e11b478c091f7e5dacd3f8edae2ffe3164d7e0eedada87ee817b

P (hex):
ef3fc61e21867a900e01ee4b1ba69f5403274ed27656da03ed88d7902cce693f

Q (hex):
c9b9fcc298b7d1af568f85b50e749539bc01b10a68472fe1302058104821cd65

D mod (P-1) (hex):
9f7fd96969baefc6009569edcbd19bf8d576f89e1a439e6ad4905e50ac8899b7f

D mod (Q-1) (hex):
867bfd7107a8bca39b503ce09a30e267d567606f02f7540cac03ab5856bde43

Activities Firefox Web Browser ▾

Tue 11:39

Virtual Labs Archived content - Nmap ✎ Inbox (2,081) - shirraasu ✎ CNS - Google Drive ✎ Server Not Found ✎ +

Digital Signatures Scheme

Hash output(hex):
a94a8fe5ccb19ba61c4c0873d391e987982fbcd3

Input to RSA(hex):
a94a8fe5ccb19ba61c4c0873d391e987982fbcd3

Digital Signature(hex):
7c927533cd5b28dc56941c1c727b04e69fd4231df9dc39963dedd8886be12362
465cf049eb77b85aa6563b0dee3aca953d6e809b5406fb9b7437ea068011c3d

Digital Signature(base64):
f3J1M80wXkXlBwcncnsE5p/U1x353DmWPe301GvhI2JGXPB63e4WqZw03u0sqV
PW6Gm10G+8m3036gaEcPQ==

Status:
Time: 8ms

RSA public key

Public exponent (hex, F4=0x10001):
3

Modulus (hex):
BC86E3DC782C446E756B874ACECF2A115E613021EAF1ED5EF295BE28ED8990
26FE2EC896BF90E84FE381AF67A7B7CBBA4B085235E72AB8595ABF8FEB4B05F80B

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) - shirraasu ✎ file.io - Super simple file ✎ +

Digital Signatures Scheme

Hash output(hex):
a94a8fe5ccb19ba61c4c0873d391e987982fbcd3

Input to RSA(hex):
a94a8fe5ccb19ba61c4c0873d391e987982fbcd3

Digital Signature(hex):
7c927533cd5b28dc56941c1c727b04e69fd4231df9dc39963dedd8886be12362
465cf049eb77b85aa6563b0dee3aca953d6e809b5406fb9b7437ea068011c3d

Digital Signature(base64):
f3J1M80wXkXlBwcncnsE5p/U1x353DmWPe301GvhI2JGXPB63e4WqZw03u0sqV
PW6Gm10G+8m3036gaEcPQ==

Status:
Time: 8ms

RSA public key

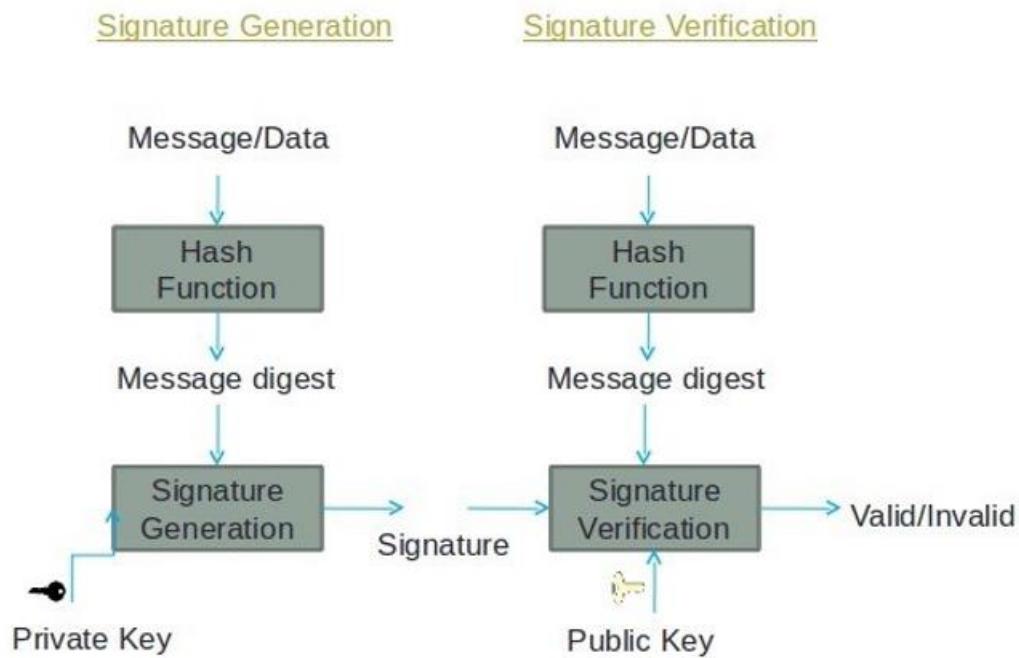
Public exponent (hex, F4=0x10001):
3

Modulus (hex):
BC86E3DC782C446E756B874ACECF2A115E613021EAF1ED5EF295BE28ED8990
26FE2EC896BF90E84FE381AF67A7B7CBBA4B085235E72AB8595ABF8FEB4B05F80B

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

Digital Signatures

Digital Signature Process



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:

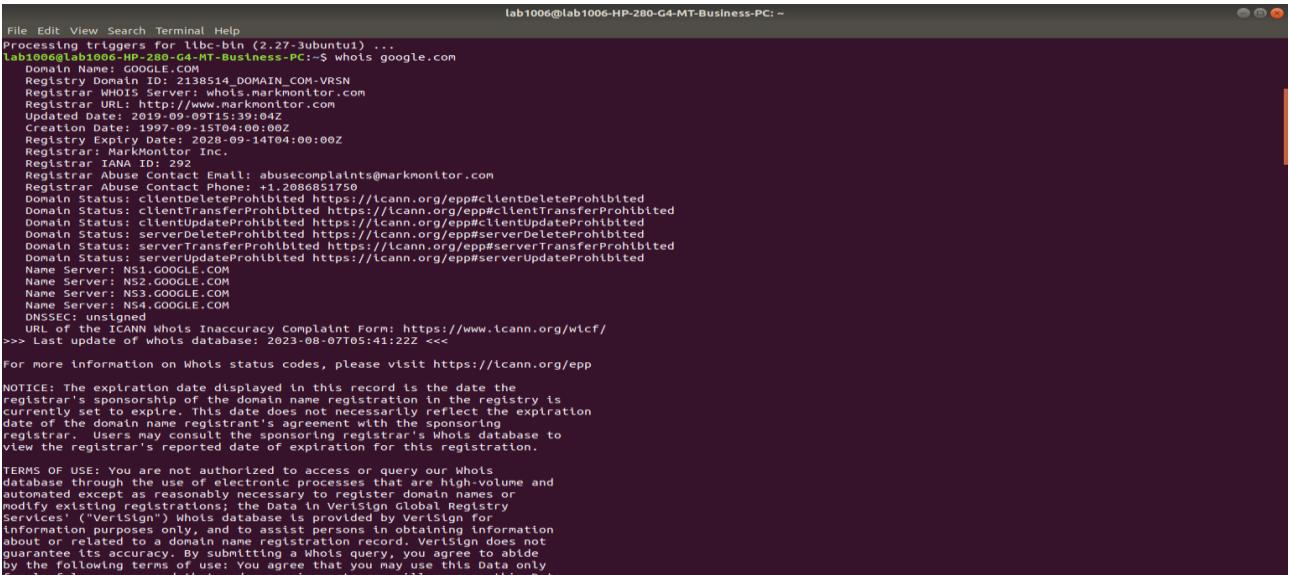
WHOIS:

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:



```
File Edit View Search Terminal Help
Processing triggers for libc-bin (2.27-3ubuntu1) ...
lab1006@lab1006-HP-280-G4-MT-Business-PC: ~
whois google.com
Domain Name: GOOGLE.COM
Registry Domain ID: 2138514_DOMAIN_COM-VRSN
Registrar: MarkMonitor Inc.
Registrar URL: http://www.markmonitor.com
Updated Date: 2019-09-09T15:39:04Z
Creation Date: 1997-09-15T04:00:00Z
Registry Expiry Date: 2028-09-14T04:00:00Z
Registrar: MarkMonitor Inc.
Registrar IP: 192.168.1.12
Registrar Abuse Contact Email: abusecomplaints@markmonitor.com
Registrar Abuse Contact Phone: +1.2086851750
Domain Status: clientDeleteProhibited https://icann.org/epp#clientDeleteProhibited
Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
Domain Status: clientUpdateProhibited https://icann.org/epp#clientUpdateProhibited
Domain Status: serverDeleteProhibited https://icann.org/epp#serverDeleteProhibited
Domain Status: serverTransferProhibited https://icann.org/epp#serverTransferProhibited
Domain Status: serverUpdateProhibited https://icann.org/epp#serverUpdateProhibited
Name Server: NS1.GOOGLE.COM
Name Server: NS2.GOOGLE.COM
Name Server: NS3.GOOGLE.COM
Name Server: NS4.GOOGLE.COM
DNSSEC: unsigned
URL for the ICANN Whois Inaccuracy Complaint Form: https://www.icann.org/wicf/
>>> Last update of whois database: 2023-08-07T05:41:22Z <<<

For more information on Whois status codes, please visit https://icann.org/epp

NOTICE: The expiration date displayed in this record is the date the
registrar's sponsorship of the domain name registration in the registry is
currently set to expire. This date does not necessarily reflect the expiration
date of the domain name registrant's agreement with the sponsoring
registrar. Users may consult the sponsoring registrar's Whois database to
view the registrar's reported date of expiration for this registration.

TERMS OF USE: You are not authorized to access or query our Whois
database through the use of electronic processes that are high-volume and
automated except as reasonably necessary to register domain names or
modify existing registrations; the Data in VeriSign Global Registry
Services' ("VeriSign") Whois database is provided by VeriSign for
information purposes only, and to assist persons in obtaining information
about a domain name registration record. VeriSign does not
guarantee its accuracy. By submitting a Whois query, you agree to abide
by the following terms of use: You agree that you may use this Data only
for lawful purposes and that you will not use this Data to
attempt to circumvent our security measures.
```

```
lab1006@lab1006-HP-280-G4-MT-Business-PC: ~
File Edit View Search Terminal Help
to restrict your access to the Whois database in its sole discretion to ensure
operational stability. VeriSign may restrict or terminate your access to the
Whois database for failure to abide by these terms of use. VeriSign
reserves the right to modify these terms at any time.

The Registry database contains ONLY .COM, .NET, .EDU domains and
Registrars.
Domain Name: google.com
Registry Domain ID: 2138514_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.markmonitor.com
Registrar URL: http://www.markmonitor.com
Updated Date: 2019-09-09T15:39:04+0000
Creation Date: 1997-09-15T07:00:00+0000
Registrar Registration Expiration Date: 2028-09-13T07:00:00+0000
Registrar: MarkMonitor, Inc.
Registrar IANA ID: 292
Registrar Abuse Contact Email: abusecomplaints@markmonitor.com
Registrar Abuse Contact Phone: +1.2086851750
Domain Status: clientUpdateProhibited (https://www.icann.org/epp#clientUpdateProhibited)
Domain Status: clientTransferProhibited (https://www.icann.org/epp#clientTransferProhibited)
Domain Status: clientDeleteProhibited (https://www.icann.org/epp#clientDeleteProhibited)
Domain Status: serverUpdateProhibited (https://www.icann.org/epp#serverUpdateProhibited)
Domain Status: serverTransferProhibited (https://www.icann.org/epp#serverTransferProhibited)
Domain Status: serverDeleteProhibited (https://www.icann.org/epp#serverDeleteProhibited)
Registrant Organization: Google LLC
Registrant State/Province: CA
Registrant Country: US
Registrant Email: Select Request Email Form at https://domains.markmonitor.com/whois/google.com
Admin Organization: Google LLC
Admin State/Province: CA
Admin Country: US
Admin Email: Select Request Email Form at https://domains.markmonitor.com/whois/google.com
Tech Organization: Google LLC
Tech State/Province: CA
Tech Country: US
Tech Email: Select Request Email Form at https://domains.markmonitor.com/whois/google.com
Name Server: ns1.google.com
Name Server: ns3.google.com
Name Server: ns2.google.com
Name Server: ns4.google.com
DNSSEC: unsigned
URL of the ICANN WHOIS Data Problem Reporting System: http://wdprs.internic.net/
>>> Last update of WHOIS database: 2023-08-07T05:38:17+0000 <<

For more information on WHOIS status codes, please visit:
https://www.icann.org/resources/pages/epp-status-codes
```

```
lab1006@lab1006-HP-280-G4-MT-Business-PC: ~
File Edit View Search Terminal Help
DNSSEC: unsigned
URL of the ICANN WHOIS Data Problem Reporting System: http://wdprs.internic.net/
>>> Last update of WHOIS database: 2023-08-07T05:38:17+0000 <<

For more information on WHOIS status codes, please visit:
https://www.icann.org/resources/pages/epp-status-codes

If you wish to contact this domain's Registrant, Administrative, or Technical
contact, and such email address is not visible above, you may do so via our web
form, pursuant to ICANN's Temporary Specification. To verify that you are not a
robot, please enter your email address to receive a link to a page that
facilitates email communication with the relevant contact(s).

Web-based WHOIS:
https://domains.markmonitor.com/whois

If you have a legitimate interest in viewing the non-public WHOIS details, send
your request and the reasons for your request to whoisrequest@markmonitor.com
and specify the domain name in the subject line. We will review that request and
may ask for supporting documentation and explanation.

The data in MarkMonitor's WHOIS database is provided for information purposes,
and to assist persons in obtaining information about or related to a domain
name's registration record. While MarkMonitor believes the data to be accurate,
the data is provided "as is" with no guarantee or warranties regarding its
accuracy.

By submitting a WHOIS query, you agree that you will use this data only for
lawful purposes and that, under no circumstances will you use this data to:
(1) allow, enable, or otherwise support the transmission by email, telephone,
or facsimile of mass, unsolicited, commercial advertising, or spam; or
(2) enable high volume, automated, or electronic processes that send queries,
data, or email to MarkMonitor (or its systems) or the domain name contacts (or
its systems).

MarkMonitor reserves the right to modify these terms at any time.

By submitting this query, you agree to abide by this policy.

MarkMonitor Domain Management(TM)
Protecting companies and consumers in a digital world.

Visit MarkMonitor at https://www.markmonitor.com
Contact us at +1.8007459229
In Europe, at +44.62032062220
```

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  7 11:49:26 IST 2023
;; MSG SIZE  rcvd: 55

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

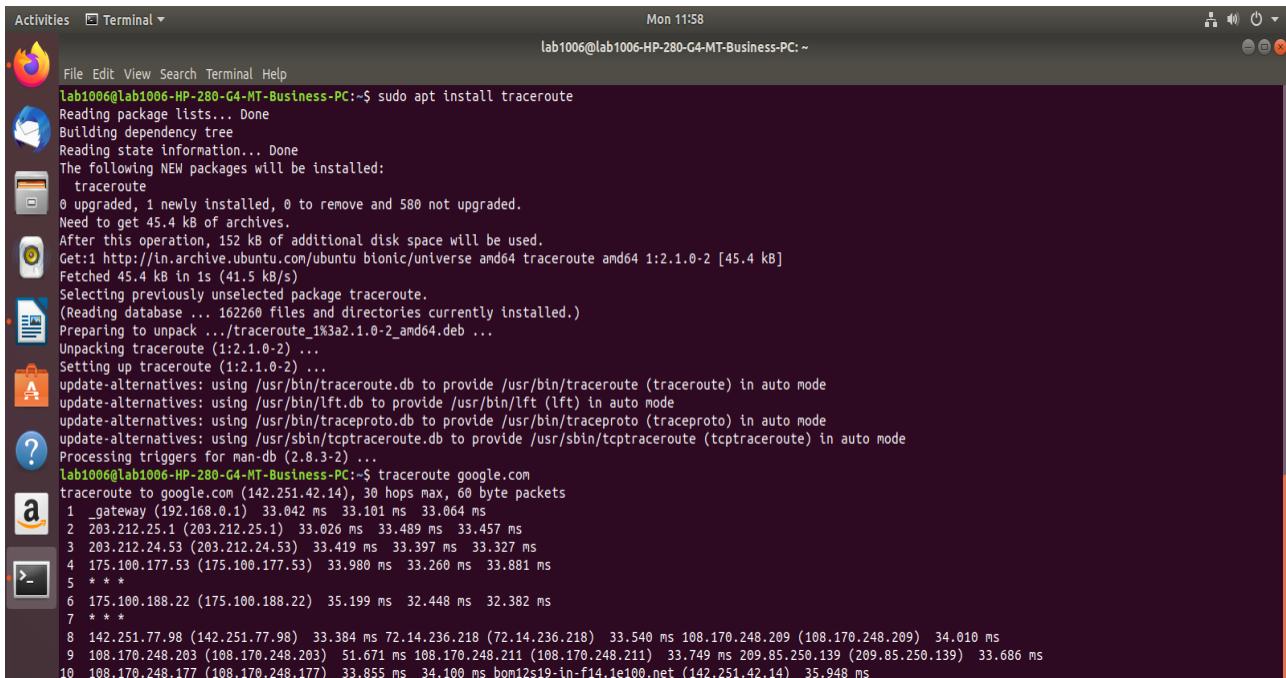
; <>> DiG 9.11.3-1ubuntu1.18-Ubuntu <>> @120.120.40.80 google.com
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ █
```

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



```
Activities Terminal Mon 11:58
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ sudo apt install traceroute
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
traceroute
0 upgraded, 1 newly installed, 0 to remove and 580 not upgraded.
Need to get 45.4 kB of archives.
After this operation, 152 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 traceroute amd64 1:2.1.0-2 [45.4 kB]
Fetched 45.4 kB in 1s (41.5 kB/s)
Selecting previously unselected package traceroute.
(Reading database ... 162260 files and directories currently installed.)
Preparing to unpack .../traceroute_1%3a2.1.0-2_amd64.deb ...
Unpacking traceroute (1:2.1.0-2) ...
Setting up traceroute (1:2.1.0-2) ...
update-alternatives: using /usr/bin/traceroute.db to provide /usr/bin/traceroute (traceroute) in auto mode
update-alternatives: using /usr/bin/lft.db to provide /usr/bin/lft (lft) in auto mode
update-alternatives: using /usr/bin/traceproto.db to provide /usr/bin/traceproto (traceproto) in auto mode
update-alternatives: using /usr/sbin/tcptraceroute.db to provide /usr/sbin/tcptraceroute (tcptraceroute) in auto mode
Processing triggers for man-db (2.8.3-2) ...
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ traceroute google.com
traceroute to google.com (142.251.42.14), 30 hops max, 60 byte packets
1 _gateway (192.168.0.1) 33.042 ms 33.101 ms 33.064 ms
2 203.212.25.1 (203.212.25.1) 33.026 ms 33.489 ms 33.457 ms
3 203.212.24.53 (203.212.24.53) 33.419 ms 33.397 ms 33.327 ms
4 175.100.177.53 (175.100.177.53) 33.980 ms 33.260 ms 33.881 ms
5 * * *
6 175.100.188.22 (175.100.188.22) 35.199 ms 32.448 ms 32.382 ms
7 * * *
8 142.251.77.98 (142.251.77.98) 33.384 ms 72.14.236.218 (72.14.236.218) 33.540 ms 108.170.248.209 (108.170.248.209) 34.010 ms
9 108.170.248.203 (108.170.248.203) 51.671 ms 108.170.248.211 (108.170.248.211) 33.749 ms 209.85.250.139 (209.85.250.139) 33.686 ms
10 108.170.248.177 (108.170.248.177) 33.855 ms 34.100 ms bom1s19-in-f14.1e100.net (142.251.42.14) 35.948 ms
```

nslookup:

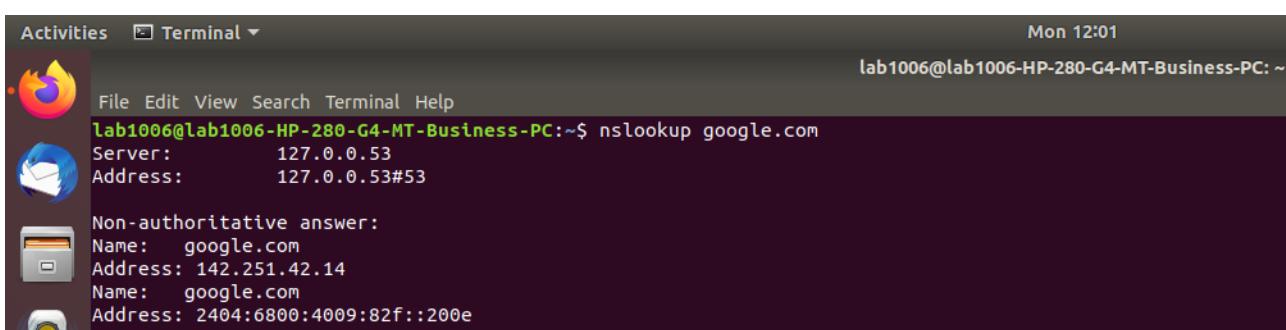
This command is also used to find DNS related query.

Syntax:

nslookup <domainName>

Example:

nslookup google.com



```
Activities Terminal Mon 12:01
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ nslookup google.com
Server: 127.0.0.53
Address: 127.0.0.53#53

Non-authoritative answer:
Name: google.com
Address: 142.251.42.14
Name: google.com
Address: 2404:6800:4009:82f::200e
```

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 'sf(fe)' 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.

```
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ sudo apt install dmitry
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  dmitry
0 upgraded, 1 newly installed, 0 to remove and 580 not upgraded.
Need to get 19.8 kB of archives.
After this operation, 55.3 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 dmitry amd64 1.3a-1build1 [19.8 kB]
Fetched 19.8 kB in 0s (37.4 kB/s)
Selecting previously unselected package dmitry.
(Reading database ... 162383 files and directories currently installed.)
Preparing to unpack .../dmitry_1.3a-1build1_amd64.deb ...
Unpacking dmitry (1.3a-1build1) ...
Setting up dmitry (1.3a-1build1) ...
Processing triggers for man-db (2.8.3-2) ...
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ dmitry w tsec.edu
Deepmagic Information Gathering Tool
"There be some deep magic going on"

HostIP:162.241.70.62
HostName:tsec.edu

Gathered Inet-whois information for 162.241.70.62
-----
inetnum:      162.222.91.0 - 162.244.51.255
netname:      NON-RIPE-NCC-MANAGED-ADDRESS-BLOCK
desc:         IPv4 address block not managed by the RIPE NCC
remarks:      -----
remarks:      -----
remarks:      For registration information,
remarks:      you can consult the following sources:
remarks:      -----
remarks:      IANA
remarks:      http://www.iana.org/assignments/ipv4-address-space
remarks:      http://www.iana.org/assignments/iana-ipv4-special-registry
remarks:      http://www.iana.org/assignments/ipv4-recovered-address-space
remarks:      -----
remarks:      AFRINIC (Africa)
remarks:      http://www.afrinic.net/ whois.afrinic.net
remarks:      -----
remarks:      APNIC (Asia Pacific)
remarks:      http://www.apnic.net/ whois.apnic.net

```

```
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ 
File Edit View Search Terminal Help
+91.2226495888
gtthampi@yahoo.com

Technical Contact:
  Chetan Agarwal
  Thadomal Shahani Engineering College
  Narl Gurshahani Marg, Bandra(W)
  Mumbai, 400050
  India
  +91.2226495888
  chetan.agarwal@thadomal.org

Name Servers:
  NS2.SALESUPP.IN
  NS1.SALESUPP.IN

Domain record activated: 22-Jan-2001
Domain record last updated: 06-Aug-2023
Domain expires: 31-Jul-2023

Gathered Netcraft information for tsec.edu
-----
Retrieving Netcraft.com information for tsec.edu
Netcraft.com Information gathered

Gathered Subdomain information for tsec.edu
-----
Searching Google.com:80...
HostName:www.tsec.edu
HostIP:162.241.70.62
HostName:alumni.tsec.edu
HostIP:13.213.42.252
Searching Altavista.com:80...
Found 2 possible subdomain(s) for host tsec.edu, Searched 0 pages containing 0 results

Gathered E-Mail information for tsec.edu
-----
Searching Google.com:80...
Searching Altavista.com:80...
Found 0 E-Mail(s) for host tsec.edu, Searched 0 pages containing 0 results

Gathered TCP Port information for 162.241.70.62
-----
Port      State
```

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 x Virtual Labs x | +

cse29-iith.vlabs.ac.in/exp/aes/simulation.html

AES and Modes of Operation

PART II

Key size in bits: **128**

Plaintext: **1430bd11 701dd2ad b7d90f98 52049cbd
86e3f64 f13700e1 6dd03b7a 02a29598
95afbc5a 9dcffef6 c9fb1f23 561f9da3
ddf6ea6b7 82449904 a16d27e9 0310d43c
f13b7af0 5b671007 83297474 0672c47c**

Key: **5a9d79b4 76e99124 2faa9420 e3210f3f**

Next Plaintext | Next Keytext

IV: **[]** Next IV

CTR: **[]** Next CTR

PART III

Calculate XOR:

XOR: **[]** Calculate 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 35d6ea5c | Check Answer

1:36 AM 8/18/2023

Virtual Labs x Virtual Labs x | +

cse29-iith.vlabs.ac.in/exp/aes/simulation.html

AES and Modes of Operation

PART II

Key size in bits: **128**

Plaintext: **1430bd11 701dd2ad b7d90f98 52049cbd
86e3f64 f13700e1 6dd03b7a 02a29598
95afbc5a 9dcffef6 c9fb1f23 561f9da3
ddf6ea6b7 82449904 a16d27e9 0310d43c
f13b7af0 5b671007 83297474 0672c47c**

Key: **5a9d79b4 76e99124 2faa9420 e3210f3f**

Next Plaintext | Next Keytext

IV: **4f064ce3 4956d135 bbb92277 45c23602** Next IV

PART III

Calculate XOR:

XOR: **[]** Calculate XOR

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

1:44 AM 8/18/2023

Virtual Labs x Virtual Labs x | +

cse29-iiith.vlabs.ac.in/exp/aes/simulation.html

AES and Modes of Operation

Key size in bits: 128

Plaintext: Next Plaintext Key: Next Keytext

IV: Next IV

CTR: Next CTR

PART III

Calculate XOR:

XOR:

PART IV

Key in hex:

Plaintext in hex:

Ciphertext in hex:

PART V

Enter your answer here: Check Answer!

CORRECT!!

Search the web and Windows 1:57 AM 8/18/2023

Virtual Labs x Virtual Labs x | +

cse29-iiith.vlabs.ac.in/exp/aes/simulation.html

AES and Modes of Operation

Key size in bits: 128

Plaintext: Next Plaintext Key: Next Keytext

IV: Next IV

PART III

Calculate XOR:

XOR:
 Calculate XOR

PART IV

Key in hex:

Plaintext in hex:

Ciphertext in hex:

PART V

Enter your answer here: Check Answer!

Search the web and Windows 2:12 AM 8/18/2023

PART II

Key size in bits: 128

Plaintext:	Next Plaintext	Key:	Next Keytext
IV:	Next IV		

PART III

Calculate XOR:

db5fffff b2eac19e 1e34b698 c95fb1f4	Calculate XOR
a39a0278 44def0d0 25201e0e e0507bc4	
XOR:	78c55d8f f634314e 3b14a896 290fcfa30

PART IV

Key in hex: ec2e4fe 79456a55 55714438 c9acf42

Plaintext in hex: c50752e5 c9e07941 ebd18082 b1617ad5

Ciphertext in hex: db5fffff b2eac19e 1e34b698 c95fb1f4

PART V

Enter your answer here:

96a95b9c 75de435a b13b2ccb cc667287 9a67ac46 746308e0 0a1d0061 fdac273

Sorry, answer is wrong. Please try again.

2:29 AM
8/18/2023

PART II

Key size in bits: 128

Plaintext:	Next Plaintext	Key:	Next Keytext
IV:	Next IV		

PART III

Calculate XOR:

70f3e39c b2ea62b0 364cce06 9a3ec189	Calculate XOR
f0868a2e 47d4353a 1c258fb0 7784d7fc	
XOR:	807569b2 f53e578a 2a694116 edba1675

PART IV

Key in hex: 6a99eb9a 367784ca 9da70dff 7a5e0666

Plaintext in hex: a2d07b2 e9ad6b61 07781fdc 144a12cc

Ciphertext in hex: 70f3e39c b2ea62b0 364cce06 9a3ec189

PART V

Enter your answer here:

c1e29f57 ed9fb230 b9a77bd9 075c207a b4af2a21 d57ab4c9 491a1955 8c17fc00

CORRECT!!

2:53 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 fixed-size 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
```

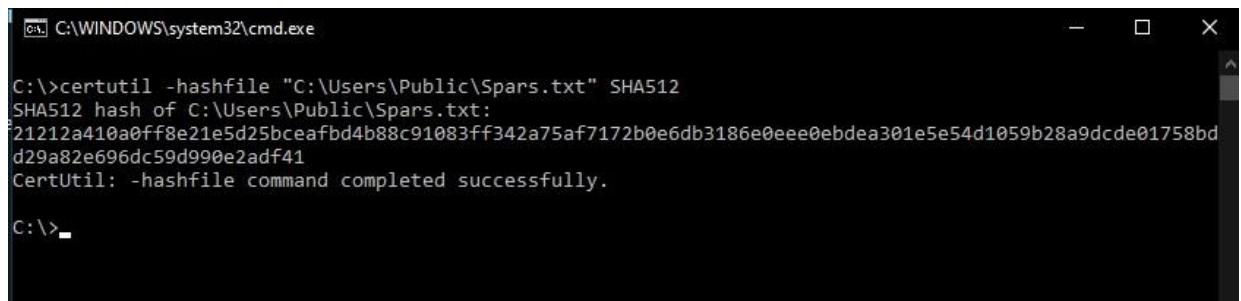
Replace <file> with the filename.

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:

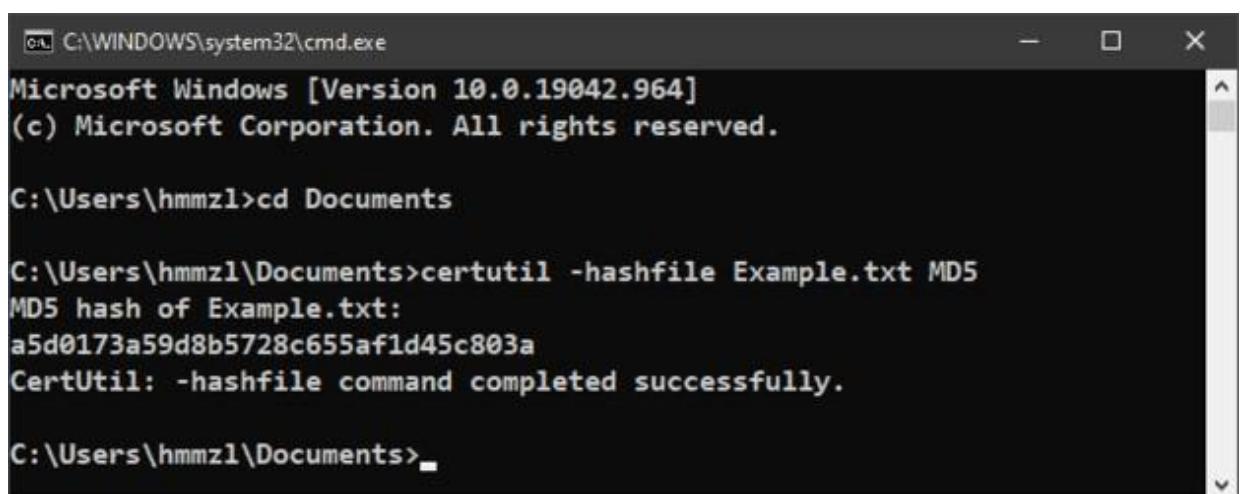
```
certutil -hashfile Example.txt MD5
```

Output:



```
C:\>certutil -hashfile "C:\Users\Public\Spars.txt" SHA512
SHA512 hash of C:\Users\Public\Spars.txt:
21212a410a0ff8e21e5d25bceafbd4b88c91083ff342a75af7172b0e6db3186e0eee0ebdea301e5e54d1059b28a9dcde01758bd
d29a82e696dc59d990e2adf41
CertUtil: -hashfile command completed successfully.

C:\>
```



```
C:\>C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19042.964]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hmmz1>cd Documents

C:\Users\hmmz1\Documents>certutil -hashfile Example.txt MD5
MD5 hash of Example.txt:
a5d0173a59d8b5728c655af1d45c803a
CertUtil: -hashfile command completed successfully.

C:\Users\hmmz1\Documents>
```

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:

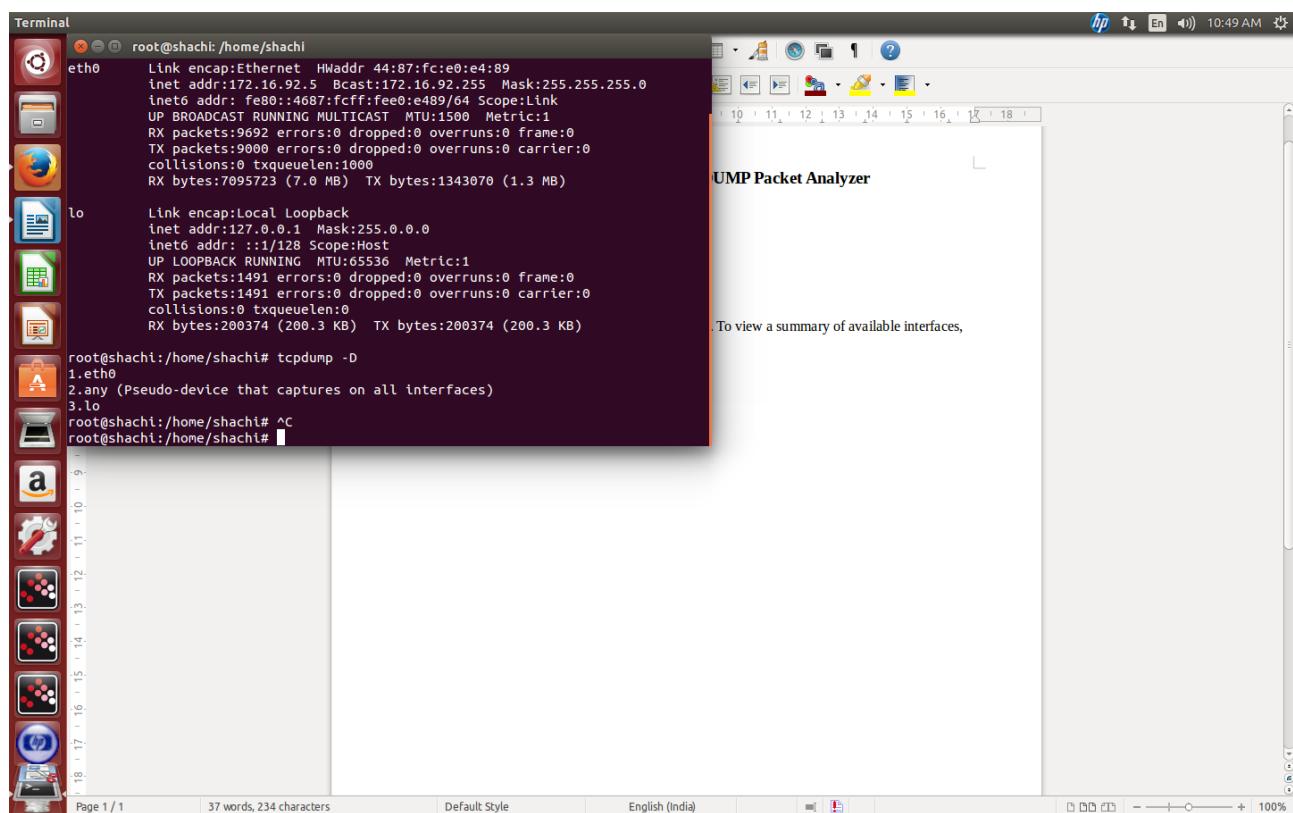
1. To install tcpdump

```
$ 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.

```

root@shachi:/home/shachi#
inet6 addr: fe80::4687:fcff:fee0:e489/64 Scope:Link
  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
  RX packets:9692 errors:0 dropped:0 overruns:0 frame:0
  TX packets:9000 errors:0 dropped:0 overruns:0 carrier:0
  collisions:0 txqueuelen:1000
  RX bytes:7095723 (7.0 MB) TX bytes:1343070 (1.3 MB)

lo      Link encap:Local Loopback
        inet6 addr: 127.0.0.1 Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
        UP LOOPBACK RUNNING MTU:65536 Metric:1
        RX packets:1491 errors:0 dropped:0 overruns:0 frame:0
        TX packets:1491 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:200374 (200.3 KB) TX bytes:200374 (200.3 KB)

root@shachi:/home/shachi# tcpdump -D
1.eth0
2.any (Pseudo-device that captures on all interfaces)
3.lo
root@shachi:/home/shachi# ^C
root@shachi:/home/shachi# tcpdump -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
11:11:48.755303 IP 172.16.92.5.43780 > 106.10.184.41.443: Flags [.], ack 263857829, win 367, options [nop,nop,TS val 2072096 ecr 29009846], length 0
11:11:48.817850 IP 106.10.184.41.443 > 172.16.92.5.43780: Flags [.], ack 1, win 97, options [nop,nop,TS val 29019925 ecr 2056996], length 0
11:11:49.811361 IP 172.16.92.5.43754 > 106.10.184.41.443: Flags [.], ack 3180545544, win 604, options [nop,nop,TS val 2072360 ecr 29010743], length 0
11:11:50.038240 IP 106.10.184.41.443 > 172.16.92.5.43754: Flags [.], ack 1, win 434, options [nop,nop,TS val 29020981 ecr 2056999], length 0
11:11:50.254917 IP 172.16.92.1.53 > 172.16.92.5.25220: 49967 0/1/0 (95)
11:11:51.602814 IP 172.16.92.5.39662 > 104.244.42.129.443: Flags [P.], seq 466596947:466596993, ack 1889999207, win 237, options [nop,nop,TS val 2072807
ecr 3388722348], length 46
11:11:51.602870 IP 172.16.92.5.39663 > 104.244.42.129.443: Flags [P.], seq 3440648596:3440648642, ack 2287316051, win 237, options [nop,nop,TS val 20728
07 ecr 3387829666], length 46
11:11:51.731329 IP 172.16.92.5.38774 > 106.10.200.161.443: Flags [.], ack 3095876107, win 886, options [nop,nop,TS val 2072840 ecr 3356291879], length 0
11:11:51.791803 IP 106.10.200.161.443 > 172.16.92.5.38774: Flags [.], ack 1, win 501, options [nop,nop,TS val 3356301942 ecr 2067807], length 0
11:11:51.855464 IP 104.244.42.129.443 > 172.16.92.5.39662: Flags [.], ack 46, win 122, options [nop,nop,TS val 3388788638 ecr 2072807], length 0
11:11:51.855481 IP 104.244.42.129.443 > 172.16.92.5.39662: Flags [P.], seq 1:47, ack 46, win 122, options [nop,nop,TS val 3388788638 ecr 2072807], lengt
h 46
11:11:51.863087 IP 104.244.42.129.443 > 172.16.92.5.39663: Flags [.], ack 46, win 122, options [nop,nop,TS val 3387887640 ecr 2072807], length 0
11:11:51.863102 IP 104.244.42.129.443 > 172.16.92.5.39663: Flags [P.], seq 1:47, ack 46, win 122, options [nop,nop,TS val 3387887640 ecr 2072807], lengt
h 46
11:11:51.895360 IP 172.16.92.5.39662 > 104.244.42.129.443: Flags [.], ack 47, win 237, options [nop,nop,TS val 2072881 ecr 3388788638], length 0
11:11:51.899345 IP 172.16.92.5.39663 > 104.244.42.129.443: Flags [.], ack 47, win 237, options [nop,nop,TS val 2072882 ecr 3387887640], length 0
11:11:51.931365 IP 172.16.92.5.43672 > 66.196.113.5.443: Flags [.], ack 2320217246, win 403, options [nop,nop,TS val 2073280 ecr 3020942202], length 0
11:11:53.693795 IP 66.196.113.5.443 > 172.16.92.5.43672: Flags [.], ack 1, win 501, options [nop,nop,TS val 3020952409 ecr 2065621], length 0
[2]+ Stopped                 tcpdump -n
root@shachi:/home/shachi# 

```

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.

```

root@shachi:/home/shachi#
11:11:50.038240 IP 106.10.184.41.443 > 172.16.92.5.43754: Flags [.], ack 1, win 434, options [nop,nop,TS val 29020981 ecr 2056999], length 0
11:11:50.254917 IP 172.16.92.1.53 > 172.16.92.5.25220: 49967 0/1/0 (95)
11:11:51.602814 IP 172.16.92.5.39662 > 104.244.42.129.443: Flags [P.], seq 466596947:466596993, ack 1889999207, win 237, options [nop,nop,TS val 2072807
ecr 3388722348], length 46
11:11:51.602870 IP 172.16.92.5.39663 > 104.244.42.129.443: Flags [P.], seq 3440648596:3440648642, ack 2287316051, win 237, options [nop,nop,TS val 20728
07 ecr 3387829666], length 46
11:11:51.731329 IP 172.16.92.5.38774 > 106.10.200.161.443: Flags [.], ack 3095876107, win 886, options [nop,nop,TS val 2072840 ecr 3356291879], length 0
11:11:51.791803 IP 106.10.200.161.443 > 172.16.92.5.38774: Flags [.], ack 1, win 501, options [nop,nop,TS val 3356301942 ecr 2067807], length 0
11:11:51.855464 IP 104.244.42.129.443 > 172.16.92.5.39662: Flags [.], ack 46, win 122, options [nop,nop,TS val 3388788638 ecr 2072807], length 0
11:11:51.855481 IP 104.244.42.129.443 > 172.16.92.5.39662: Flags [P.], seq 1:47, ack 46, win 122, options [nop,nop,TS val 3388788638 ecr 2072807], lengt
h 46
11:11:51.863087 IP 104.244.42.129.443 > 172.16.92.5.39663: Flags [.], ack 46, win 122, options [nop,nop,TS val 3387887640 ecr 2072807], length 0
11:11:51.863102 IP 104.244.42.129.443 > 172.16.92.5.39663: Flags [P.], seq 1:47, ack 46, win 122, options [nop,nop,TS val 3387887640 ecr 2072807], lengt
h 46
11:11:51.895360 IP 172.16.92.5.39662 > 104.244.42.129.443: Flags [.], ack 47, win 237, options [nop,nop,TS val 2072881 ecr 3388788638], length 0
11:11:51.899345 IP 172.16.92.5.39663 > 104.244.42.129.443: Flags [.], ack 47, win 237, options [nop,nop,TS val 2072882 ecr 3387887640], length 0
11:11:51.931365 IP 172.16.92.5.43672 > 66.196.113.5.443: Flags [.], ack 2320217246, win 403, options [nop,nop,TS val 2073280 ecr 3020942202], length 0
11:11:53.693795 IP 66.196.113.5.443 > 172.16.92.5.43672: Flags [.], ack 1, win 501, options [nop,nop,TS val 3020952409 ecr 2065621], length 0
[2]+ Stopped                 tcpdump -v -n
root@shachi:/home/shachi# tcpdump -v -n
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
11:20:45.512191 IP (tos 0x0, ttl 30, id 25702, offset 0, flags [none], proto UDP (17), length 244)
  172.16.92.1.53 > 172.16.92.5.35016: 5900 2/4/4 daisy.ubuntu.com. A 91.189.92.57, daisy.ubuntu.com. A 91.189.92.55 (216)
11:20:47.099783 IP (tos 0x20, ttl 46, id 39760, offset 0, flags [DF], proto TCP (6), length 137)
  66.196.113.5.443 > 172.16.92.5.43672: Flags [P.], cksm 0x1f64 (correct), seq 2320218351:2320218436, ack 1558268419, win 501, options [nop,nop,TS va
l 3021485760 ecr 2196611], length 85
11:20:47.102616 IP (tos 0x0, ttl 64, id 20779, offset 0, flags [DF], proto UDP (17), length 77)
  172.16.92.5.29918 > 172.16.92.1.53: 17445+ A? prod4.rest-notify.msg.yahoo.com. (49)
11:20:47.107884 IP (tos 0x0, ttl 30, id 25706, offset 0, flags [none], proto UDP (17), length 341)
  172.16.92.1.53 > 172.16.92.5.29918: 17445 2/5/6 prod4.rest-notify.msg.yahoo.com. CNAME rproxy1.us2.msg.vip.bf1.yahoo.com., rproxy1.us2.msg.vip.bf1.y
ahoo.com. A 66.196.113.5 (313)
11:20:47.108829 IP (tos 0x0, ttl 64, id 20781, offset 0, flags [DF], proto UDP (17), length 77)
  172.16.92.5.45937 > 172.16.92.1.53: 527+ AAAA? prod4.rest-notify.msg.yahoo.com. (49)
11:20:47.113757 IP (tos 0x0, ttl 30, id 25710, offset 0, flags [none], proto UDP (17), length 186)
  172.16.92.1.53 > 172.16.92.5.45937: 527 1/1/0 prod4.rest-notify.msg.yahoo.com. CNAME rproxy1.us2.msg.vip.bf1.yahoo.com. (158)
11:20:47.119612 IP (tos 0x0, ttl 64, id 42412, offset 0, flags [DF], proto TCP (6), length 1400)
  172.16.92.5.43672 > 66.196.113.5.443: Flags [.], cksm 0x4adf (correct), seq 1:1349, ack 85, win 403, options [nop,nop,TS val 2206687 ecr 3021485760
], length 1348
11:20:47.119621 IP (tos 0x0, ttl 64, id 42413, offset 0, flags [DF], proto TCP (6), length 469)
  172.16.92.5.43672 > 66.196.113.5.443: Flags [P.], cksm 0x1a61 (correct), seq 1349:1766, ack 85, win 403, options [nop,nop,TS val 2206687 ecr 302148
5760], length 417
11:20:47.322527 IP (tos 0x20, ttl 46, id 39761, offset 0, flags [DF], proto TCP (6), length 52)
  66.196.113.5.443 > 172.16.92.5.43672: Flags [.], cksm 0x8384 (correct), ack 1766, win 500, options [nop,nop,TS val 3021485983 ecr 2206687], length
  0
[2]+ Stopped                 tcpdump -v -n
root@shachi:/home/shachi# 

```

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.

```
root@shachi:/home/shachi#
172.16.92.1.1900 > 239.255.255.250.1900: UDP, length 333
[1]: 31:20.018915 90:8d:78:7e:5a:b2 > 01:00:5e:7f:ff:fa, ethertype IPv4 (0x0800), length 303: (tos 0x0, ttl 4, id 29192, offset 0, flags [none], proto UDP (17), length 289)
172.16.92.1.1900 > 239.255.255.250.1900: UDP, length 261
[1]: 31:20.019444 90:8d:78:7e:5a:b2 > 01:00:5e:7f:ff:fa, ethertype IPv4 (0x0800), length 312: (tos 0x0, ttl 4, id 29194, offset 0, flags [none], proto UDP (17), length 298)
172.16.92.1.1900 > 239.255.255.250.1900: UDP, length 270
[1]: 31:20.019935 90:8d:78:7e:5a:b2 > 01:00:5e:7f:ff:fa, ethertype IPv4 (0x0800), length 367: (tos 0x0, ttl 4, id 29196, offset 0, flags [none], proto UDP (17), length 353)
172.16.92.1.1900 > 239.255.255.250.1900: UDP, length 325
[1]: 31:20.020443 90:8d:78:7e:5a:b2 > 01:00:5e:7f:ff:fa, ethertype IPv4 (0x0800), length 377: (tos 0x0, ttl 4, id 29198, offset 0, flags [none], proto UDP (17), length 363)
172.16.92.1.1900 > 239.255.255.250.1900: UDP, length 335
^Z
[4]+ Stopped tcpdump -v -n -e
root@shachi:/home/shachi# tcpdump -v -n -e
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
11:35:45.138738 44:87:fc:e0:e4:89 > 90:8d:78:7e:5a:b2, ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 1109, offset 0, flags [DF], proto ICMP (1), length 64)
172.16.92.5 > 172.16.92.1: ICMP echo request, id 3724, seq 12, length 64
[1]: 35:45.140514 90:8d:78:7e:5a:b2 > 44:87:fc:e0:e4:89, ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 30, id 1109, offset 0, flags [DF], proto ICMP (1), length 84)
172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 12, length 64
[1]: 35:45.1779324 44:87:fc:e0:e4:89 > 90:8d:78:7e:5a:b2, ethertype IPv4 (0x0800), length 66: (tos 0x0, ttl 64, id 16457, offset 0, flags [DF], proto TCP (6), length 52)
172.16.92.5.43966 > 66.196.113.5.4433: Flags [.], cksum 0xdbd4 (correct), ack 3232201735, win 361, options [nop,nop,TS val 2431352 ecr 2765622340], length 64
[1]: 35:45.1979872 90:8d:78:7e:5a:b2 > 44:87:fc:e0:e4:89, ethertype IPv4 (0x0800), length 66: (tos 0x20, ttl 47, id 50408, offset 0, flags [DF], proto TCP (6), length 52)
66.196.113.5.443 > 172.16.92.5.43966: Flags [.], cksum 0xd160 (correct), ack 1, win 501, options [nop,nop,TS val 2765632547 ecr 2423680], length 0
[1]: 35:46.140677 44:87:fc:e0:e4:89 > 90:8d:78:7e:5a:b2, ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 1182, offset 0, flags [DF], proto ICMP (1), length 84)
172.16.92.5 > 172.16.92.1: ICMP echo request, id 3724, seq 13, length 64
[1]: 35:46.1424323 90:8d:78:7e:5a:b2 > 44:87:fc:e0:e4:89, ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 30, id 1182, offset 0, flags [DF], proto ICMP (1), length 84)
172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 13, length 64
[1]: 35:46.16.39883 90:8d:78:7e:5a:b2 > 01:00:5e:7f:ff:fa, ethertype IPv4 (0x0800), length 307: (tos 0x0, ttl 4, id 29720, offset 0, flags [none], proto UDP (17), length 293)
172.16.92.1.1900 > 239.255.255.250.1900: UDP, length 265
[1]: 35:46.640656 90:8d:78:7e:5a:b2 > 01:00:5e:7f:ff:fa, ethertype IPv4 (0x0800), length 316: (tos 0x0, ttl 4, id 29722, offset 0, flags [none], proto UDP (17), length 302)
172.16.92.1.1900 > 239.255.255.250.1900: UDP, length 274
[1]: 35:46.6414993 90:8d:78:7e:5a:b2 > 01:00:5e:7f:ff:fa, ethertype IPv4 (0x0800), length 379: (tos 0x0, ttl 4, id 29724, offset 0, flags [none], proto UDP (17), length 365)
172.16.92.1.1900 > 239.255.255.250.1900: UDP, length 337
[1]: 35:46.642337 90:8d:78:7e:5a:b2 > 01:00:5e:7f:ff:fa, ethertype IPv4 (0x0800), length 371: (tos 0x0, ttl 4, id 29726, offset 0, flags [none], proto UDP (17), length 357)
172.16.92.1.1900 > 239.255.255.250.1900: UDP, length 329
```

Filtering packets using expressions

6. selecting packets with specific protocol

```
# tcpdump -n tcp
```

```
root@shachi:/home/shachi#
11:35:46.651490 90:8d:78:7e:5a:b2 > 01:00:5e:7f:ff:fa, ethertype IPv4 (0x0800), length 367: (tos 0x0, ttl 4, id 29748, offset 0, flags [none], proto UDP (17), length 353)
172.16.92.1.1900 > 239.255.255.250.1900: UDP, length 325
[1]: 35:46.652334 90:8d:78:7e:5a:b2 > 01:00:5e:7f:ff:fa, ethertype IPv4 (0x0800), length 377: (tos 0x0, ttl 4, id 29750, offset 0, flags [none], proto UDP (17), length 363)
172.16.92.1.1900 > 239.255.255.250.1900: UDP, length 335
[1]: 35:47.142545 44:87:fc:e0:e4:89 > 90:8d:78:7e:5a:b2, ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 1313, offset 0, flags [DF], proto ICMP (1), length 84)
172.16.92.5 > 172.16.92.1: ICMP echo request, id 3724, seq 14, length 64
[1]: 35:47.144290 90:8d:78:7e:5a:b2 > 44:87:fc:e0:e4:89, ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 30, id 1313, offset 0, flags [DF], proto ICMP (1), length 84)
172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 14, length 64
^Z
[5]+ Stopped tcpdump -v -n -e
root@shachi:/home/shachi# tcpdump -v -n -e
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
Listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
11:44:13.597356 IP 172.16.92.5.43754 > 106.10.184.41.443: Flags [.], ack 3180552726, win 952, options [nop,nop,TS val 2558304 ecr 3945170839], length 0
11:44:13.814359 IP 106.10.184.41.443 > 172.16.92.5.43754: Flags [.], ack 1, win 501, options [nop,nop,TS val 25584558 ecr 25586261], length 0
11:44:17.683363 IP 172.16.92.5.43966 > 66.196.113.5.443: Flags [.], ack 3232203064, win 382, options [nop,nop,TS val 2559328 ecr 2766134192], length 0
11:44:17.886454 IP 66.196.113.5.443 > 172.16.92.5.43966: Flags [.], ack 1, win 501, options [nop,nop,TS val 2766144400 ecr 2554220], length 0
11:44:20.254369 IP 172.16.92.5.53367 > 203.84.220.151.443: Flags [P..], seq 3527887078:3527887124, ack 1056001841, win 2235, options [nop,nop,TS val 2559
970 ecr 394511991], length 46
11:44:20.257363 IP 203.84.220.151.443 > 172.16.92.5.53367: Flags [.], ack 46, win 251, options [nop,nop,TS val 3945170883 ecr 2559970], length 0
11:44:20.257387 IP 203.84.220.151.443 > 172.16.92.5.53367: Flags [P..], seq 1:47, ack 46, win 251, options [nop,nop,TS val 3945170883 ecr 2559970], lengt
h 46
11:44:20.257406 IP 172.16.92.5.53367 > 203.84.220.151.443: Flags [.], ack 47, win 2235, options [nop,nop,TS val 2559971 ecr 3945170883], length 0
11:44:21.409517 IP 172.16.92.5.53367 > 203.84.220.151.443: Flags [P..], seq 46:307, ack 47, win 2235, options [nop,nop,TS val 3945170883], le
ngth 0
11:44:21.451314 IP 172.16.92.5.53367 > 203.84.220.151.443: Flags [.], ack 307, win 251, options [nop,nop,TS val 3945172278 ecr 2560259], length 0
11:44:21.630640 IP 203.84.220.151.443 > 172.16.92.5.53367: Flags [P..], seq 47:435, ack 307, win 251, options [nop,nop,TS val 3945172256 ecr 2560259], le
ngth 368
11:44:21.630673 IP 172.16.92.5.53367 > 203.84.220.151.443: Flags [.], ack 435, win 2232, options [nop,nop,TS val 2560314 ecr 3945172256], length 0
11:44:21.630680 IP 203.84.220.151.443 > 172.16.92.5.53367: Flags [P..], seq 435:483, ack 307, win 251, options [nop,nop,TS val 3945172256 ecr 2560259], l
ength 48
11:44:21.630687 IP 172.16.92.5.53367 > 203.84.220.151.443: Flags [.], ack 483, win 2232, options [nop,nop,TS val 2560314 ecr 3945172256], length 0
11:44:21.631097 IP 203.84.220.151.443 > 172.16.92.5.53367: Flags [P..], seq 483:492, ack 307, win 251, options [nop,nop,TS val 3945172257 ecr 2560259], le
ngth 368
11:44:21.631110 IP 172.16.92.5.53367 > 203.84.220.151.443: Flags [.], ack 1962, win 2226, options [nop,nop,TS val 2560314 ecr 3945172257], length 0
11:44:21.631243 IP 203.84.220.151.443 > 172.16.92.5.53367: Flags [.], seq 1962:3310, ack 307, win 251, options [nop,nop,TS val 3945172257 ecr 2560259], l
ength 1348
11:44:21.631251 IP 172.16.92.5.53367 > 203.84.220.151.443: Flags [.], ack 3310, win 2228, options [nop,nop,TS val 2560314 ecr 3945172257], length 0
11:44:21.631340 IP 203.84.220.151.443 > 172.16.92.5.53367: Flags [P..], seq 3310:4569, ack 307, win 251, options [nop,nop,TS val 3945172257 ecr 2560259], l
ength 1259
11:44:21.631355 IP 172.16.92.5.53367 > 203.84.220.151.443: Flags [.], ack 4569, win 2221, options [nop,nop,TS val 2560315 ecr 3945172257], length 0
^Z
[6]+ Stopped tcpdump -n tcp
root@shachi:/home/shachi#
```

tcpdump -n icmp

```

root@shachi:/home/shachi
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
11:44:13.587356 IP 172.16.92.5.43754 > 106.10.184.41.443: Flags [.], ack 3180552726, win 952, options [nop,nop,TS val 2558304 ecr 30954319], length 0
11:44:13.814359 IP 106.10.184.41.443 > 172.16.92.5.43754: Flags [.], ack 1, win 501, options [nop,nop,TS val 30964558 ecr 25506261], length 0
11:44:17.683363 IP 172.16.92.5.43966 > 66.196.113.5.443: Flags [.], ack 3232203064, win 382, options [nop,nop,TS val 2559328 ecr 2766134192], length 0
11:44:20.254369 IP 172.16.92.5.53367 > 203.84.220.151.443: Flags [.], ack 1, win 501, options [nop,nop,TS val 2766144400 ecr 2554220], length 0
970 ecr 3945111991, length 46
11:44:20.257363 IP 203.84.220.151.443 > 172.16.92.5.53367: Flags [.], ack 46, win 251, options [nop,nop,TS val 3945170883 ecr 2559970], length 0
11:44:20.257387 IP 203.84.220.151.443 > 172.16.92.5.53367: Flags [.], seq 1:47, ack 46, win 251, options [nop,nop,TS val 3945170883 ecr 2559970], length 46
11:44:20.257400 IP 172.16.92.5.53367 > 203.84.220.151.443: Flags [.], ack 47, win 2235, options [nop,nop,TS val 2559971 ecr 3945170883], length 0
11:44:21.409517 IP 172.16.92.5.53367 > 203.84.220.151.443: Flags [.], seq 46:307, ack 47, win 2235, options [nop,nop,TS val 2560259 ecr 3945170883], length 261
11:44:21.451314 IP 203.84.220.151.443 > 172.16.92.5.53367: Flags [.], ack 307, win 251, options [nop,nop,TS val 3945172078 ecr 2560259], length 0
11:44:21.630640 IP 203.84.220.151.443 > 172.16.92.5.53367: Flags [.], seq 47:435, ack 307, win 251, options [nop,nop,TS val 3945172256 ecr 2560259], length 388
11:44:21.630673 IP 172.16.92.5.53367 > 203.84.220.151.443: Flags [.], ack 435, win 2232, options [nop,nop,TS val 2560314 ecr 3945172256], length 0
11:44:21.630680 IP 203.84.220.151.443 > 172.16.92.5.53367: Flags [.], seq 435:483, ack 307, win 251, options [nop,nop,TS val 3945172256 ecr 2560259], length 48
11:44:21.630687 IP 172.16.92.5.53367 > 203.84.220.151.443: Flags [.], ack 483, win 2232, options [nop,nop,TS val 2560314 ecr 3945172256], length 0
11:44:21.631097 IP 203.84.220.151.443 > 172.16.92.5.53367: Flags [.], seq 483:1962, ack 307, win 251, options [nop,nop,TS val 3945172257 ecr 2560259], length 1479
11:44:21.631110 IP 172.16.92.5.53367 > 203.84.220.151.443: Flags [.], ack 1962, win 2226, options [nop,nop,TS val 2560314 ecr 3945172257], length 0
11:44:21.631243 IP 203.84.220.151.443 > 172.16.92.5.53367: Flags [.], seq 1962:3310, ack 307, win 251, options [nop,nop,TS val 3945172257 ecr 2560259], length 1348
11:44:21.631251 IP 172.16.92.5.53367 > 203.84.220.151.443: Flags [.], ack 3310, win 2228, options [nop,nop,TS val 2560314 ecr 3945172257], length 0
11:44:21.631340 IP 203.84.220.151.443 > 172.16.92.5.53367: Flags [.], seq 3310:4569, ack 307, win 251, options [nop,nop,TS val 3945172257 ecr 2560259], length 1259
11:44:21.631355 IP 172.16.92.5.53367 > 203.84.220.151.443: Flags [.], ack 4569, win 2221, options [nop,nop,TS val 2560315 ecr 3945172257], length 0
^Z
[6]+ Stopped tcpdump -n tcp
root@shachi:/home/shachi# tcpdump -n icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
11:47:16.185939 IP 172.16.92.5 > 172.16.92.1: ICMP echo request, id 3724, seq 702, length 64
11:47:16.187282 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 702, length 64
11:47:17.187478 IP 172.16.92.5 > 172.16.92.1: ICMP echo request, id 3724, seq 703, length 64
11:47:17.188813 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 703, length 64
11:47:18.189022 IP 172.16.92.5 > 172.16.92.1: ICMP echo request, id 3724, seq 704, length 64
11:47:18.189035 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 704, length 64
11:47:19.190569 IP 172.16.92.5 > 172.16.92.1: ICMP echo request, id 3724, seq 705, length 64
11:47:19.191909 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 705, length 64
11:47:20.192087 IP 172.16.92.5 > 172.16.92.1: ICMP echo request, id 3724, seq 706, length 64
11:47:20.193480 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 706, length 64
^Z
[7]+ Stopped tcpdump -n icmp
root@shachi:/home/shachi# ^C
root@shachi:/home/shachi#

```

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
```

```

root@shachi:/home/shachi
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
11:47:19.190569 IP 172.16.92.5 > 172.16.92.1: ICMP echo request, id 3724, seq 705, length 64
11:47:19.191909 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 705, length 64
11:47:20.192087 IP 172.16.92.5 > 172.16.92.1: ICMP echo request, id 3724, seq 706, length 64
11:47:20.193480 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 706, length 64
^Z
[7]+ Stopped tcpdump -n icmp
root@shachi:/home/shachi# ^C
root@shachi:/home/shachi# tcpdump -n src 172.16.92.1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
[8]+ Stopped tcpdump -n src 172.16.92.1
root@shachi:/home/shachi# tcpdump -n src 172.16.92.1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
[9]+ Stopped tcpdump -n src 172.16.92.1
root@shachi:/home/shachi# ^C
root@shachi:/home/shachi# tcpdump -n src 172.16.92.1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
11:51:51.587639 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 977, length 64
11:51:52.589224 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 978, length 64
11:51:53.590784 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 979, length 64
11:51:54.592528 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 980, length 64
11:51:55.593744 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 981, length 64
11:51:56.595282 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 982, length 64
11:51:57.596758 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 983, length 64
11:51:58.598208 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 984, length 64
11:51:59.599724 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 985, length 64
^Z
[9]+ Stopped tcpdump -n src 172.16.92.1
root@shachi:/home/shachi# ^C
root@shachi:/home/shachi# tcpdump -n src 172.16.92.1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
11:52:26.641253 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 1012, length 64
11:52:27.642934 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 1013, length 64
11:52:28.644622 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 1014, length 64
11:52:29.646324 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 1015, length 64
11:52:30.647978 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 1016, length 64
11:52:31.649661 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 1017, length 64
^Z
[10]+ Stopped tcpdump -n src 172.16.92.1
root@shachi:/home/shachi# ^C
root@shachi:/home/shachi# tcpdump -n src 172.16.92.1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
[11]+ Stopped tcpdump -n src 172.16.92.1
root@shachi:/home/shachi# ^C
root@shachi:/home/shachi# tcpdump -n src 172.16.92.1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
[12]+ Stopped tcpdump -n src 172.16.92.1
root@shachi:/home/shachi# ^C
root@shachi:/home/shachi#

```

```
# tcpdump -n dst 172.16.92.1
```

```

root@shachi:/home/shachi
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
11:51:51.587639 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 977, length 64
11:51:52.589224 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 978, length 64
11:51:54.592258 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 979, length 64
11:51:55.593744 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 980, length 64
11:51:56.595282 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 981, length 64
11:51:57.596758 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 983, length 64
11:51:58.598208 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 984, length 64
11:51:59.599724 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 985, length 64
^Z
[10]+ Stopped tcpdump -n src 172.16.92.1
root@shachi:/home/shachi# tcpdump -n src 172.16.92.1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
11:52:26.641253 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 1012, length 64
11:52:27.642934 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 1013, length 64
11:52:28.644622 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 1014, length 64
11:52:29.646324 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 1015, length 64
11:52:30.647978 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 1016, length 64
11:52:31.649661 IP 172.16.92.1 > 172.16.92.5: ICMP echo reply, id 3724, seq 1017, length 64
^Z
[11]+ Stopped tcpdump -n src 172.16.92.1
root@shachi:/home/shachi# tcpdump -n src 172.16.92.1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
^Z
[11]+ Stopped tcpdump -n src 172.16.92.1
root@shachi:/home/shachi# tcpdump -n src 172.16.92.1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
^Z
[12]+ Stopped tcpdump -n src 172.16.92.2
root@shachi:/home/shachi# tcpdump -n dst 172.16.92.1
tcpdump: syntax error
root@shachi:/home/shachi# tcpdump -n dst 172.16.92.1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
^Z
[12]+ Stopped tcpdump -n src 172.16.92.2
root@shachi:/home/shachi# tcpdump -n dst 172.16.92.1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
^Z
[12]+ Stopped tcpdump -n src 172.16.92.2
root@shachi:/home/shachi# tcpdump -n dst 172.16.92.1
tcpdump: syntax error
root@shachi:/home/shachi# tcpdump -n dst 172.16.92.1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
^Z
[13]+ Stopped tcpdump -n dst 172.16.92.1
root@shachi:/home/shachi#

```

tcpdump -n port 80

```

root@shachi:/home/shachi
length 389
11:59:10.250287 IP 172.16.92.5.60751 > 23.65.124.32.80: Flags [.], ack 1738, win 272, options [nop,nop,TS val 2782469 ecr 1810011394], length 0
11:59:10.262485 IP 23.65.124.25.80 > 172.16.92.5.56518: Flags [.], seq 2097:4045, ack 665, win 947, options [nop,nop,TS val 1965639424 ecr 2782467], len
gth 1348
11:59:10.262512 IP 172.16.92.5.56518 > 23.65.124.25.80: Flags [.], ack 17492, win 523, options [nop,nop,TS val 2782472 ecr 1965639403,nop,nop,sack 1 {26
97:4045}], length 0
11:59:10.262707 IP 23.65.124.25.80 > 172.16.92.5.56518: Flags [.], seq 4045:5393, ack 665, win 947, options [nop,nop,TS val 1965639424 ecr 2782467], len
gth 1348
11:59:10.262717 IP 172.16.92.5.56518 > 23.65.124.25.80: Flags [.], ack 17492, win 523, options [nop,nop,TS val 2782472 ecr 1965639403,nop,nop,sack 1 {40
45:5393}], length 0
11:59:10.263056 IP 23.65.124.25.80 > 172.16.92.5.56518: Flags [.], seq 5393:6741, ack 665, win 947, options [nop,nop,TS val 1965639424 ecr 2782467], len
gth 1348
11:59:10.263066 IP 172.16.92.5.56518 > 23.65.124.25.80: Flags [.], ack 17492, win 523, options [nop,nop,TS val 2782472 ecr 1965639403,nop,nop,sack 1 {53
93:6741}], length 0
11:59:10.263177 IP 23.65.124.25.80 > 172.16.92.5.56518: Flags [.], seq 6741:8089, ack 665, win 947, options [nop,nop,TS val 1965639424 ecr 2782467], len
gth 1348
11:59:10.263182 IP 172.16.92.5.56518 > 23.65.124.25.80: Flags [.], ack 17492, win 523, options [nop,nop,TS val 2782472 ecr 1965639403,nop,nop,sack 1 {67
41:8089}], length 0
11:59:10.263403 IP 23.65.124.25.80 > 172.16.92.5.56518: Flags [.], seq 8089:9437, ack 665, win 947, options [nop,nop,TS val 1965639424 ecr 2782467], len
gth 1348
11:59:10.263412 IP 172.16.92.5.56518 > 23.65.124.25.80: Flags [.], ack 17492, win 523, options [nop,nop,TS val 2782473 ecr 1965639403,nop,nop,sack 1 {80
89:9437}], length 0
11:59:10.265416 IP 23.65.124.25.80 > 172.16.92.5.56518: Flags [.], ack 665, win 947, options [nop,nop,TS val 1965639428 ecr 2782469], length 0
11:59:10.266724 IP 23.65.124.25.80 > 172.16.92.5.56518: Flags [.], ack 665, win 947, options [nop,nop,TS val 1965639430 ecr 2782469], length 0
11:59:10.267394 IP 23.65.124.25.80 > 172.16.92.5.56518: Flags [.], ack 665, win 947, options [nop,nop,TS val 1965639430 ecr 2782469], length 0
11:59:10.283304 IP 23.2.16.91.80 > 172.16.92.5.52070: Flags [P.], seq 108257:109074, ack 9383, win 1498, options [nop,nop,TS val 2288872455 ecr 2782455]
, length 817
11:59:10.283339 IP 172.16.92.5.52070 > 23.2.16.91.80: Flags [.], ack 109074, win 1324, options [nop,nop,TS val 2782477 ecr 2288872455], length 0
11:59:10.289595 IP 23.2.16.91.80 > 172.16.92.5.52067: Flags [P.], seq 101844:104084, ack 6505, win 1319, options [nop,nop,TS val 2288872458 ecr 2782455]
, length 2240
11:59:10.295197 IP 172.16.92.5.52067 > 23.2.16.91.80: Flags [.], ack 104084, win 1324, options [nop,nop,TS val 2782480 ecr 2288872458], length 0
^Z
[14]+ Stopped tcpdump -n port 80
root@shachi:/home/shachi# tcpdump -n port 80
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
11:59:42.993474 IP 54.225.189.34.80 > 172.16.92.5.57552: Flags [.], ack 3070349890, win 76, options [nop,nop,TS val 952097209 ecr 2788093], length 0
11:59:43.763310 IP 172.16.92.5.42413 > 74.125.130.147.80: Flags [.], ack 357444536, win 237, options [nop,nop,TS val 2790848 ecr 2528258055], length 0
11:59:43.795321 IP 172.16.92.5.42416 > 74.125.130.147.80: Flags [.], ack 1841024414, win 237, options [nop,nop,TS val 2790856 ecr 2477051253], length 0
11:59:43.827318 IP 74.125.130.147.80 > 172.16.92.5.42413: Flags [.], ack 1, win 342, options [nop,nop,TS val 2528268135 ecr 2783305], length 0
11:59:43.827330 IP 172.16.92.5.53185 > 23.58.34.202.80: Flags [.], ack 1901034413, win 507, options [nop,nop,TS val 2790864 ecr 2433724615], length 0
11:59:43.851874 IP 23.58.34.202.80 > 172.16.92.5.53185: Flags [.], ack 1, win 1005, options [nop,nop,TS val 2433734662 ecr 2783336], length 0
11:59:43.857986 IP 74.125.130.147.80 > 172.16.92.5.42416: Flags [.], ack 1, win 342, options [nop,nop,TS val 2477061333 ecr 2783319], length 0
11:59:44.075833 IP 23.2.16.91.80 > 172.16.92.5.52070: Flags [.], ack 1, win 1657, options [nop,nop,TS val 2288906245 ecr 2783345], length 0
^Z
[15]+ Stopped tcpdump -n port 80
root@shachi:/home/shachi# tcpdump -n port 80

```

tcpdump port 80

```

root@shachi:/home/shachi#


```

Specific Packets from specific port

tcpdump udp and src port 53

```

root@shachi:/home/shachi#


```

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

```
root@shachi:/home/shachi#
[12]:+ 12:14:31.420791 IP 172.16.92.5.18617 > 18-24-212-203.fxwirelessol.com.domain: 1965+ PTR? 71.197.243.54.in-addr.arpa. (44)
[12]:+ 12:14:31.428241 IP 18-24-212-203.fxwirelessol.com.domain > 172.16.92.5.18617: 1965 1/5/6 PTR ec2-54-243-197-71.compute-1.amazonaws.com. (320)
[12]:+ 12:14:31.428618 IP 172.16.92.5.10318 > 18-24-212-203.fxwirelessol.com.domain: 39949+ PTR? 46.24.222.203.in-addr.arpa. (44)
[12]:+ 12:14:31.432230 IP 18-24-212-203.fxwirelessol.com.domain > 172.16.92.5.10318: 39949* 1/1 PTR 46-24-212-203.fxwirelessol.com. (123)
[12]:+ 12:14:32.568515 IP 172.16.92.5.63039 > 18-24-212-203.fxwirelessol.com.domain: 5923+ A? pr.comet.yahoo.com. (36)
[12]:+ 12:14:32.569005 IP 172.16.92.5.34956 > 18-24-212-203.fxwirelessol.com.domain: 21633+ A? pr.comet.yahoo.com. (36)
[12]:+ 12:14:32.572141 IP 18-24-212-203.fxwirelessol.com.domain > 172.16.92.5.63039: 5923 2/4/4 CNAME ds-comet.yahoo.pr.g03.yahoodns.net., A 106.10.200.161 (247)
[12]:+ 12:14:32.619316 IP 172.16.92.5.42206 > ec2-23-23-187-101.compute-1.amazonaws.com.http: Flags [.], ack 1691709936, win 237, options [nop,nop,TS val 30130 ecr 1476060867], length 0
[12]:+ 12:14:32.792167 IP 18-24-212-203.fxwirelessol.com.domain > 172.16.92.5.31826: 58264 2/4/4 CNAME ds-comet.yahoo.pr.g03.yahoodns.net., AAAA 2406:2000:e4:200::400c (259)
[12]:+ 12:14:32.793580 IP 172.16.92.5.25812 > 18-24-212-203.fxwirelessol.com.domain: 53982+ A? daisy.ubuntu.com. (34)
[12]:+ 12:14:32.793639 IP 172.16.92.5.33314 > 18-24-212-203.fxwirelessol.com.domain: 38653+ AAAA? daisy.ubuntu.com. (34)
[12]:+ 12:14:32.797639 IP 18-24-212-203.fxwirelessol.com.domain > 172.16.92.5.25812: 53982 2/4/4 A 91.189.92.57, A 91.189.92.55 (216)
[12]:+ 12:14:32.798683 IP 18-24-212-203.fxwirelessol.com.domain > 172.16.92.5.33314: 38653 0/1/0 (95)
[12]:+ 12:14:32.869576 IP ec2-23-23-187-101.compute-1.amazonaws.com.http > 172.16.92.5.42206: Flags [.], ack 1, win 75, options [nop,nop,TS val 1476063429 ecr 3010562], length 0
^Z
[18]+ Stopped tcpdump portrange 1-80
root@shachi:/home/shachi# tcpdump -n portrange 1-80
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
[12]:+ 12:15:12.405095 IP 172.16.92.5.7134 > 203.212.24.18.53: 39941+ A? daisy.ubuntu.com. (34)
[12]:+ 12:15:12.405170 IP 172.16.92.5.7134 > 203.212.24.46.53: 39941+ A? daisy.ubuntu.com. (34)
[12]:+ 12:15:12.405219 IP 172.16.92.5.41596 > 203.212.24.18.53: 11017+ AAAA? daisy.ubuntu.com. (34)
[12]:+ 12:15:12.409182 IP 203.212.24.18.53 > 172.16.92.5.7134: 39941 2/4/4 A 91.189.92.57, A 91.189.92.55 (216)
[12]:+ 12:15:12.410572 IP 203.212.24.18.53 > 172.16.92.5.41596: 11017 0/1/0 (95)
[12]:+ 12:15:13.407414 IP 172.16.92.5.26234 > 203.212.24.18.53: 16870+ A? daisy.ubuntu.com. (34)
[12]:+ 12:15:13.407519 IP 172.16.92.5.60952 > 203.212.24.18.53: 58676+ AAAA? daisy.ubuntu.com. (34)
[12]:+ 12:15:13.412069 IP 203.212.24.18.53 > 172.16.92.5.60952: 58676 0/1/0 (95)
[12]:+ 12:15:13.715375 IP 172.16.92.5.42206 > 23.23.187.101.80: Flags [.], ack 1691709936, win 237, options [nop,nop,TS val 3023336 ecr 1476071134], length 0
[12]:+ 12:15:13.966198 IP 23.23.187.101.80 > 172.16.92.5.42206: Flags [.], ack 1, win 75, options [nop,nop,TS val 1476073702 ecr 3010562], length 0
[12]:+ 12:15:14.661172 IP 172.16.92.5.59600 > 23.65.124.16.80: Flags [P.], seq 3789008896:37890088787, ack 1351089907, win 1174, options [nop,nop,TS val 3023572 ecr 18120088261], length 691
[12]:+ 12:15:14.683868 IP 23.65.124.16.80 > 172.16.92.5.59600: Flags [.], seq 1:2697, ack 691, win 1534, options [nop,nop,TS val 1812018734 ecr 3023572], length 0
[12]:+ 12:15:14.683917 IP 172.16.92.5.59600 > 23.65.124.16.80: Flags [.], ack 2697, win 1216, options [nop,nop,TS val 3023578 ecr 1812018734], length 0
[12]:+ 12:15:14.684042 IP 23.65.124.16.80 > 172.16.92.5.59600: Flags [P.], seq 2697:4676, ack 691, win 1534, options [nop,nop,TS val 1812018734 ecr 3023572], length 179
[12]:+ 12:15:14.684063 IP 172.16.92.5.59600 > 23.65.124.16.80: Flags [.], ack 4676, win 1247, options [nop,nop,TS val 3023578 ecr 1812018734], length 0
^Z
[19]+ Stopped tcpdump -n portrange 1-80
root@shachi:/home/shachi#
```

#tcpdump -n src port 443

```
root@shachi:/home/shachi#
[12]:+ 12:15:14.683917 IP 172.16.92.5.59600 > 23.65.124.16.80: Flags [.], ack 2697, win 1216, options [nop,nop,TS val 3023578 ecr 1812018734], length 0
[12]:+ 12:15:14.684042 IP 23.65.124.16.80 > 172.16.92.5.59600: Flags [P.], seq 2697:4676, ack 691, win 1534, options [nop,nop,TS val 1812018734 ecr 3023572], length 179
[12]:+ 12:15:14.684063 IP 172.16.92.5.59600 > 23.65.124.16.80: Flags [.], ack 4676, win 1247, options [nop,nop,TS val 3023578 ecr 1812018734], length 0
^Z
[19]+ Stopped tcpdump -n portrange 1-80
root@shachi:/home/shachi# tcpdump less 32
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
^Z
[20]+ Stopped tcpdump less 32
root@shachi:/home/shachi# tcpdump -n >32
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
^Z
[21]+ Stopped tcpdump -n >32
root@shachi:/home/shachi# tcpdump less 32
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
^Z
[22]+ Stopped tcpdump -n src port 1025
root@shachi:/home/shachi# tcpdump -n src port 443
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
[12]:+ 12:23:30.744322 IP 203.84.220.151.443 > 172.16.92.5.54288: Flags [.], ack 399553216, win 103, options [nop,nop,TS val 3956133356 ecr 3147592], length 0
[12]:+ 12:23:30.744352 IP 203.84.220.151.443 > 172.16.92.5.54288: Flags [P.], seq 0:46, ack 1, win 103, options [nop,nop,TS val 3956133356 ecr 3147592], length 46
[12]:+ 12:23:30.833318 IP 106.10.200.161.443 > 172.16.92.5.39631: Flags [P.], seq 3713984124:3713984422, ack 3661746272, win 151, options [nop,nop,TS val 336060379 ecr 3103255], length 298
[12]:+ 12:23:30.833357 IP 106.10.200.161.443 > 172.16.92.5.39631: Flags [P.], seq 298:616, ack 1, win 151, options [nop,nop,TS val 3360600379 ecr 3103255], length 318
[12]:+ 12:23:30.850800 IP 106.10.200.161.443 > 172.16.92.5.39631: Flags [P.], seq 616:650, ack 1, win 151, options [nop,nop,TS val 3360600398 ecr 3103255], length 34
[12]:+ 12:23:32.519042 IP 203.84.220.151.443 > 172.16.92.5.54288: Flags [.], ack 262, win 112, options [nop,nop,TS val 3956135131 ecr 3148026], length 0
[12]:+ 12:23:32.694379 IP 203.84.220.151.443 > 172.16.92.5.54288: Flags [P.], seq 46:434, ack 262, win 112, options [nop,nop,TS val 3956135306 ecr 3148026], length 388
[12]:+ 12:23:32.694431 IP 203.84.220.151.443 > 172.16.92.5.54288: Flags [P.], seq 434:482, ack 262, win 112, options [nop,nop,TS val 3956135306 ecr 3148026], length 48
[12]:+ 12:23:32.695135 IP 203.84.220.151.443 > 172.16.92.5.54288: Flags [.], seq 482:480, ack 262, win 112, options [nop,nop,TS val 3956135306 ecr 3148026], length 1348
[12]:+ 12:23:32.695415 IP 203.84.220.151.443 > 172.16.92.5.54288: Flags [P.], seq 1830:4525, ack 262, win 112, options [nop,nop,TS val 3956135306 ecr 3148026], length 2695
[12]:+ 12:23:33.0080739 IP 106.10.200.161.443 > 172.16.92.5.39631: Flags [.], ack 1, win 151, options [nop,nop,TS val 3360602627 ecr 3147619,nop,nop,sack 1 {1349:1778}], length 0
[12]:+ 12:23:33.0080788 IP 106.10.200.161.443 > 172.16.92.5.39631: Flags [.], ack 1778, win 172, options [nop,nop,TS val 3360602627 ecr 3148121], length 0
^Z
[23]+ Stopped tcpdump -n src port 443
root@shachi:/home/shachi#
```

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 port 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)

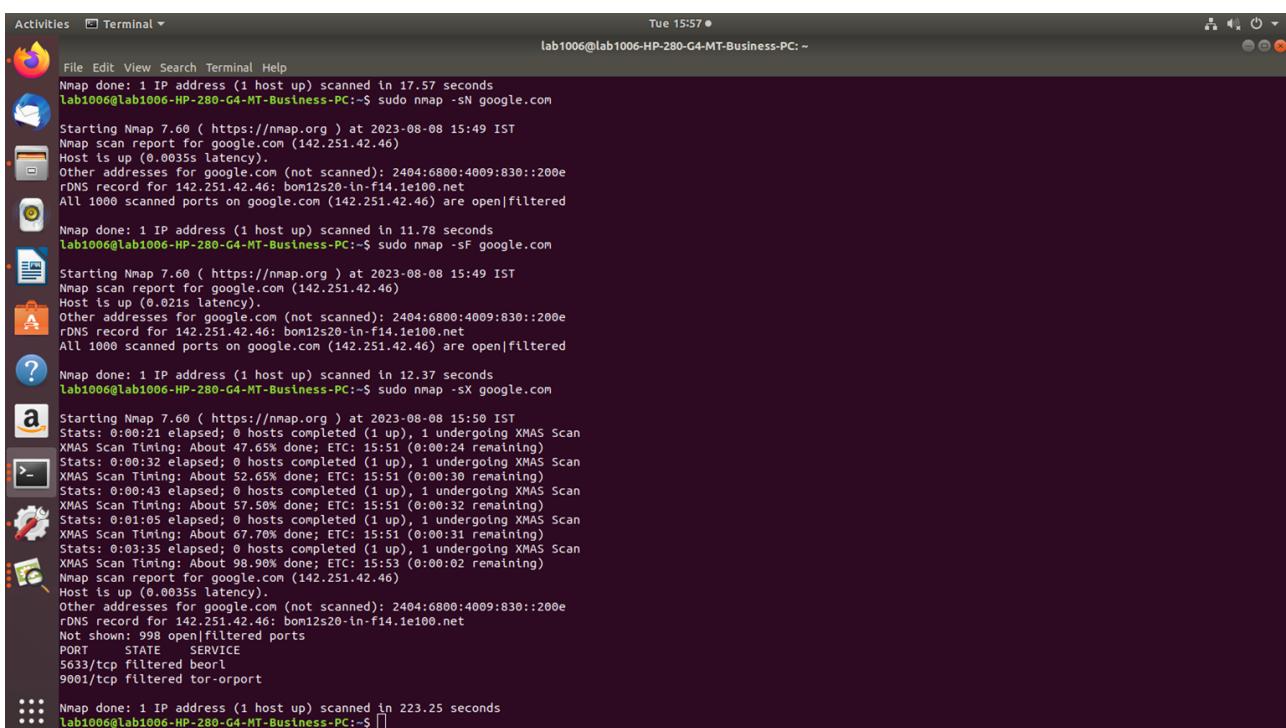
Sets just the TCP FIN bit.

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



The screenshot shows a terminal window titled "Terminal" with the command line "lab1006@lab1006-HP-280-G4-MT-Business-PC:~\$". The window displays the output of three Nmap scans for the IP address 142.251.42.46. The first scan (nmap -sN) took 17.57 seconds. The second (nmap -sF) took 11.78 seconds. The third (nmap -sX) took 12.37 seconds. The output includes details about the host being up, latency, and port status (open/filtered). It also shows XMAS Scan timing statistics and a list of open ports (5633/tcp, 9001/tcp).

```
Nmap done: 1 IP address (1 host up) scanned in 17.57 seconds
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ sudo nmap -sN google.com

Starting Nmap 7.60 ( https://nmap.org ) at 2023-08-08 15:49 IST
Nmap scan report for google.com (142.251.42.46)
Host is up (0.0035s 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
All 1000 scanned ports on google.com (142.251.42.46) are open|filtered

Nmap done: 1 IP address (1 host up) scanned in 11.78 seconds
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ sudo nmap -sF google.com

Starting Nmap 7.60 ( https://nmap.org ) at 2023-08-08 15:49 IST
Nmap scan report for google.com (142.251.42.46)
Host is up (0.021s 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
All 1000 scanned ports on google.com (142.251.42.46) are open|filtered

Nmap done: 1 IP address (1 host up) scanned in 12.37 seconds
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ sudo nmap -sX google.com

Starting Nmap 7.60 ( https://nmap.org ) at 2023-08-08 15:50 IST
Stats: 0:00:12 elapsed; 0 hosts completed (1 up), 1 undergoing XMAS Scan
XMAS Scan Timing: About 47.65% done; ETC: 15:51 (0:00:24 remaining)
Stats: 0:00:32 elapsed; 0 hosts completed (1 up), 1 undergoing XMAS Scan
XMAS Scan Timing: About 52.65% done; ETC: 15:51 (0:00:30 remaining)
Stats: 0:00:43 elapsed; 0 hosts completed (1 up), 1 undergoing XMAS Scan
XMAS Scan Timing: About 57.50% done; ETC: 15:51 (0:00:32 remaining)
Stats: 0:01:05 elapsed; 0 hosts completed (1 up), 1 undergoing XMAS Scan
XMAS Scan Timing: About 67.70% done; ETC: 15:51 (0:00:31 remaining)
Stats: 0:03:35 elapsed; 0 hosts completed (1 up), 1 undergoing XMAS Scan
XMAS Scan Timing: About 98.90% done; ETC: 15:53 (0:00:02 remaining)
Nmap scan report for google.com (142.251.42.46)
Host is up (0.0035s 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 open|filtered ports
PORT      STATE      SERVICE
5633/tcp  filtered  beorl
9001/tcp  filtered  tor-orport

Nmap done: 1 IP address (1 host up) scanned in 223.25 seconds
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$
```

-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

```
Starting Nmap 7.60 ( https://nmap.org ) at 2023-08-08 15:58 IST
Failed to resolve "google.com".
WARNING: No targets were specified, so 0 hosts scanned.
Nmap done: 0 IP addresses (0 hosts up) scanned in 10.03 seconds
Starting Nmap 7.60 ( https://nmap.org ) at 2023-08-08 15:59 IST
Nmap scan report for 192.168.0.114
Host is up (0.00097s latency).
All 1000 scanned ports on 192.168.0.114 are unfiltered
MAC Address: 04:0E:3C:19:2E:0F (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 12.12 seconds
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$
```

-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

```
Activities Terminal Tue 16:01 *
lab1006@lab1006-HP-280-G4-MT-Business-PC: ~

File Edit View Search Terminal Help
Starting Nmap 7.60 ( https://nmap.org ) at 2023-08-08 15:59 IST
Nmap scan report for 192.168.0.114
Host is up (0.00097s latency).
All 1000 scanned ports on 192.168.0.114 are unfiltered
MAC Address: 04:0E:3C:19:2E:0F (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 12.12 seconds
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ sudo nmap -sO google.com
Nmap 7.60 ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -lL <inputfilename>: Input From list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,...]>: Exclude hosts/networks
  --excludedfile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sN: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,serv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
SCAN TECHNIQUES:
  -sS/-sA/-sM: TCP SYN/Connect()/ACK/Window/Maimon scans
  -sU: UDP Scan
  -sNF/SX: TCP Null, FIN, and Xmas scans
  --scanflags <flags>: Customize TCP scan flags
  -sI <zombie host[:probeport]>: Idle scan
  -sV/sZ: SCTP INIT/COOKIE-ECHO scans
  -sO: IP protocol scan
  -b <FTP relay host>: FTP bounce scan
PORT SPECIFICATION AND SCAN ORDER:
  -p: port ranges: Only scan specified ports
    Ex: -p22; -p1-65535; -p U53,111,137,T:21-25,80,139,8080,S:9
  --exclude-ports <port ranges>: Exclude the specified ports from scanning
  -F: Fast Mode - Scan fewer ports than the default scan
  -f: Scan ports consecutively - don't randomize
  --top-ports <number>: Scan <number> most common ports
  --port-ratio <ratio>: Scan ports more common than <ratio>
SERVICE VERSION DETECTION:
```

```

Activities Terminal Tue 16:01 lab1006@lab1006-HP-280-G4-MT-Business-PC: ~
File Edit View Search Terminal Help
--sU: UDP Scan
--sV/sX: TCP Null, FIN, and Xmas scans
--scanflags <flags>: Customize TCP scan flags
-sI: <zombie host[:probeport]>: Idle scan
-sY/sZ: SCTP INIT/COOKIE-ECHO scans
-sO: IP protocol scan
-b <FTP relay host>: FTP bounce scan
PORT SPECIFICATION AND SCAN ORDER:
-p <port ranges>: Only scan specified ports
  Ex: -p22; -p1-65535; -p U:53,I:111,137,T:21-25,80,139,B:8080,S:9
--exclude-ports <port ranges>: Exclude the specified ports from scanning
-F: Fast Mode - Scan fewer ports than the default scan
-r: Scan ports consecutively - don't randomize
--top-ports <number>: Scan <number> most common ports
--port-ratio <ratio>: Scan ports more common than <ratio>
SERVICE/VERSION DETECTION:
-sV: Probe open ports to determine service/version info
--version-intensity <level>: Set from 0 (light) to 9 (try all probes)
--version-light: Limit to most likely probes (Intensity 2)
--version-all: Try every single probe (Intensity 9)
--version-trace: Show detailed version scan activity (for debugging)
SCRIPT SCAN:
-sC: equivalent to --script=default
--script=<Lua scripts>: <Lua scripts> is a comma separated list of
  directories, script-files or script-categories
--script-args=<n1=v1,[n2=v2,...>: provide arguments to scripts
--script-args-file=<filename>: provide NSE script args in a file
--script-trace: Show all data sent and received
--script-updatedb: Update the script database
--script-help=<Lua scripts>: Show help about scripts.
  <Lua scripts> is a comma-separated list of script-files or
  script-categories.
OS DETECTION:
-O: Enable OS detection
--osscan-limit: Limit OS detection to promising targets
--osscan-guess: Guess OS more aggressively
TIMING AND PERFORMANCE:
Options which take <tme> are in seconds, or append 'ms' (milliseconds),
's' (seconds), 'm' (minutes), or 'h' (hours) to the value (e.g. 30m).
-T<0-5>: Set timing template (higher is faster)
--min-hostgroup/max-hostgroup <sizes>: Parallel host scan group sizes
--min-parallelism/max-parallelism <nprobes>: Probe parallelization
--min-rtt-timeout/max-rtt-timeout/initial-rtt-timeout <tme>: Specifies
  probe round trip time.
--max-retries <tries>: Caps number of port scan probe retransmissions.
  host timeout. This will give up on target after this long.

```

```

Activities Terminal Tue 16:01 lab1006@lab1006-HP-280-G4-MT-Business-PC: ~
File Edit View Search Terminal Help
--osscan-guess: Guess OS more aggressively
TIMING AND PERFORMANCE:
Options which take <tme> are in seconds, or append 'ms' (milliseconds),
's' (seconds), 'm' (minutes), or 'h' (hours) to the value (e.g. 30m).
-T<0-5>: Set timing template (higher is faster)
--min-hostgroup/max-hostgroup <sizes>: Parallel host scan group sizes
--min-parallelism/max-parallelism <nprobes>: Probe parallelization
--min-rtt-timeout/max-rtt-timeout/initial-rtt-timeout <tme>: Specifies
  probe round trip time.
--max-retries <tries>: Caps number of port scan probe retransmissions.
--host-timeout <tme>: Give up on target after this long
--scan-delay/-max-scan-delay <tme>: Adjust delay between probes
--min-rate <number>: Send packets no slower than <number> per second
--max-rate <number>: Send packets no faster than <number> per second
FIREWALL/IDS EVASION AND SPOOFING:
-f: --mtu <val>; fragment packets (optionally w/given MTU)
-D <decoy1,decoy2[,ME],...>: Cloak a scan with decoys
-S <IP Address>: Spoof source address
-e <iface>: Use specified interface
-g/-source-port <portnum>: Use given port number
--proxies <url1,[url2],...>: Relay connections through HTTP/SOCKS4 proxies
--data <hex strings>: Append a custom payload to sent packets
--data-string <string>: Append a custom ASCII string to sent packets
--data-length <num>: Append random data to sent packets
--ip-options <options>: Send packets with specified ip options
--ttl <val>: Set IP time-to-live field
--spoof-mac <mac address/prefix/vendor name>: Spoof your MAC address
--badsum: Send packets with a bogus TCP/UDP/SCTP checksum
OUTPUT:
-oN/-oX/-oG <file>: Output scan in normal, XML, s|<Rpt Kiddi3,
  and Grepable format, respectively, to the given filename.
-oA <basename>: Output in the three major formats at once
-v: Increase verbosity level (use -vv or more for greater effect)
-d: Increase debugging level (use -dd or more for greater effect)
--reason: Display the reason a port is in a particular state
--open: Only show open (or possibly open) ports
--packet-trace: Show all packets sent and received
--iflist: Print host interfaces and routes (for debugging)
--append-output: Append to rather than clobber specified output files
--resume <filename>: Resume an aborted scan
--stylesheet <path/URL>: XSL stylesheet to transform XML output to HTML
--webxml: Reference stylesheet from Nmap.Org for more portable XML
--no-stylesheet: Prevent associating of XSL stylesheet w/XML output
MISC:
-6: Enable IPv6 scanning
  --enable-OS-detection --version-detection --script-scanning --script-updatedb

```

```

1 [chaos]# nmap -sO 127.0.0.1
2
3 Starting Nmap 4.01 at 2006-07-14 12:56 BST
4 Interesting protocols on chaos(127.0.0.1):
5 (The 251 protocols scanned but not shown below are
6         in state: closed)
7 PROTOCOL STATE          SERVICE
8 1      open            icmp
9 2      open|filtered  igmp
10 6     open            tcp
11 17    open            udp
12 255   open|filtered  unknown
13
14 Nmap finished: 1 IP address (1 host up) scanned in
15              1.259 seconds

```

```

Activities Terminal Tue 16:01
lab1006@lab1006-HP-280-G4-MT-Business-PC: ~

File Edit View Search Terminal Help
-f: --mtu <val>; fragment packets (optionally w/given MTU)
-D <decoy1[,decoy2[,ME]]...>; Cloak a scan with decoys
-S <IP_Address>; Spoof source address
-e <iface>; Use specified interface
-g/-source-port <portnum>; Use given port number
--proxies <url1,[url2],...>; Relay connections through HTTP/SOCKS4 proxies
--data <hex_string>; Append a custom payload to sent packets
--data-string <string>; Append a custom ASCII string to sent packets
--data-length <num>; Append random data to sent packets
--ip-options <options>; Send packets with specified ip options
--ttl <val>; Set IP time-to-live field
--spoof-mac <mac address/prefix/vendor name>; Spoof your MAC address
--badsum; Send packets with a bogus TCP/UDP/SCTP checksum
OUTPUT:
--oN/-oX/-oS/-oG <file>; Output scan in normal, XML, s|<rIpt Kiddi3,
    and Grepable format, respectively, to the given filename.
-oA <basename>; Output in the three major formats at once
-v; Increase verbosity level (use -vv or more for greater effect)
-d; Increase debugging level (use -dd or more for greater effect)
--reason; Display the reason a port is in a particular state
--open; Only show open (or possibly open) ports
--packet-trace; Show all packets sent and received
--iflist; Print host interfaces and routes (for debugging)
--append-output; Append to rather than clobber specified output files
--resume <filename>; Resume an aborted scan
--stylesheet <path/URL>; XSL stylesheet to transform XML output to HTML
--webxml; Reference stylesheet from Nmap.Org for more portable XML
--no-stylesheet; Prevent associating of XSL stylesheet w/XML output
MISC:
--6; Enable IPv6 scanning
-A; Enable OS detection, version detection, script scanning, and traceroute
--datadir <dirname>; Specify custom Nmap data file location
--send-eth/-send-ip; Send using raw ethernet frames or IP packets
--privileged; Assume that the user is fully privileged
--unprivileged; Assume the user lacks raw socket privileges
-V; Print version number
-h; Print this help summary page.

EXAMPLES:
nmap -v -A scanme.nmap.org
nmap -v -sn 192.168.0.0/16 10.0.0.0/8
nmap -v -LR 10000 -Pn -p 80
SEE THE MAN PAGE (https://nmap.org/book/man.html) FOR MORE OPTIONS AND EXAMPLES
Scantype 0 not supported
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ 
```

-sV (Version detection)

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

```

Activities Terminal Tue 16:06
lab1006@lab1006-HP-280-G4-MT-Business-PC: ~

File Edit View Search Terminal Help
-oA <basename>; Output in the three major formats at once
-v; Increase verbosity level (use -vv or more for greater effect)
-d; Increase debugging level (use -dd or more for greater effect)
--reason; Display the reason a port is in a particular state
--open; Only show open (or possibly open) ports
--packet-trace; Show all packets sent and received
--iflist; Print host interfaces and routes (for debugging)
--append-output; Append to rather than clobber specified output files
--resume <filename>; Resume an aborted scan
--stylesheet <path/URL>; XSL stylesheet to transform XML output to HTML
--webxml; Reference stylesheet from Nmap.Org for more portable XML
--no-stylesheet; Prevent associating of XSL stylesheet w/XML output
MISC:
--6; Enable IPv6 scanning
-A; Enable OS detection, version detection, script scanning, and traceroute
--datadir <dirname>; Specify custom Nmap data file location
--send-eth/-send-ip; Send using raw ethernet frames or IP packets
--privileged; Assume that the user is fully privileged
--unprivileged; Assume the user lacks raw socket privileges
-V; Print version number
-h; Print this help summary page.

EXAMPLES:
nmap -v -A scanme.nmap.org
nmap -v -sn 192.168.0.0/16 10.0.0.0/8
nmap -v -LR 10000 -Pn -p 80
SEE THE MAN PAGE (https://nmap.org/book/man.html) FOR MORE OPTIONS AND EXAMPLES
Scantype 0 not supported
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ sudo nmap -sV google.com
Starting Nmap 7.60 ( https://nmap.org ) at 2023-08-08 16:05 IST
Failed to resolve "google.com".
WARNING: No targets were specified, so 0 hosts scanned.
Nmap done: 0 IP addresses (0 hosts up) scanned in 10.37 seconds
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ sudo nmap -sV 192.168.0.114
Starting Nmap 7.60 ( https://nmap.org ) at 2023-08-08 16:05 IST
Nmap scan report for 192.168.0.114
Host is up (0.0007s latency).
All 1000 scanned ports on 192.168.0.114 are closed
MAC Address: 04:0E:3C:19:2E:0F (Unknown)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 14.87 seconds
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ 
```

-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

```
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ sudo nmap -o google.com
Starting Nmap 7.60 ( https://nmap.org ) at 2023-08-08 16:07 IST
WARNING: No targets were specified, so 0 hosts scanned.
Nmap done: 0 IP addresses (0 hosts up) scanned in 0.03 seconds
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ sudo nmap -o 192.168.0.114

Starting Nmap 7.60 ( https://nmap.org ) at 2023-08-08 16:07 IST
WARNING: No targets were specified, so 0 hosts scanned.
Nmap done: 0 IP addresses (0 hosts up) scanned in 0.02 seconds
```

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.*

```
Nmap done: 0-17 addresses (0 hosts up) scanned in 0.03 seconds
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 (synchronize/acknowledge) 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:

The terminal window shows the following session:

```
Preparing to unpack .../hping3_3.a2.ds2-10_amd64.deb ...
Unpacking hping3 (3.a2.ds2-10) ...
Setting up hping3 (3.a2.ds2-10) ...
Processing triggers for man-db (2.10.2-1) ...
altaf@LAPTOP-DGNIK4U9:~$ man hping3
altaf@LAPTOP-DGNIK4U9:~$ man hping3
altaf@LAPTOP-DGNIK4U9:~$ hping3 -l --flood -a 192.168.1.103 192.168.1.255
[open_sockraw] socket(): Operation not permitted
[main] can't open raw socket
altaf@LAPTOP-DGNIK4U9:~$ sudo hping3 -l --flood -a 192.168.1.103 192.168.1.255
[sudo] password for altaf:
HPING 192.168.1.255 (eth0 192.168.1.255): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.1.255 hping statistic ---
352510 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
altaf@LAPTOP-DGNIK4U9:~$ sudo hping3 -c 15000 -d 120 -w 64 -p 80 --flood --rand-source 192.168.1.159
[sudo] password for altaf:
hping3: you must specify only one target host at a time
altaf@LAPTOP-DGNIK4U9:~$ sudo hping3 -c 15000 -d 120 w 64 -p 80 --flood --rand-source 192.168.1.159
hping3: you must specify only one target host at a time
altaf@LAPTOP-DGNIK4U9:~$ sudo hping3 -c 15000 -d 120 -w 64 -p 80 --flood --rand-source 192.168.1.159
HPING 192.168.1.159 (eth0 192.168.1.159): NO FLAGS are set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.1.159 hping statistic ---
48761 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
altaf@LAPTOP-DGNIK4U9:~$ sudo hping3 -c 15000 -d 120 -w 64 -p 80 --flood --rand-source 192.168.1.159
HPING 192.168.1.159 (eth0 192.168.1.159): S set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.1.159 hping statistic ---
232030 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
altaf@LAPTOP-DGNIK4U9:~$ 
altaf@LAPTOP-DGNIK4U9:~$ 
altaf@LAPTOP-DGNIK4U9:~$ 
altaf@LAPTOP-DGNIK4U9:~$ ^C
```

Below the terminal, the desktop environment shows the following status bar:

- Bandra Kurla Co... Construction
- 17:33 08-09-2023

The terminal window shows the following session:

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  hping3
0 upgraded, 1 newly installed, 0 to remove and 140 not upgraded.
Need to get 106 kB of archives.
After this operation, 263 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 hping3 amd64 3.a2.ds2-10 [106 kB]
Fetched 106 kB in 2s (64.4 kB/s)
Selecting previously unselected package hping3.
(Reading database ... 53952 files and directories currently installed.)
Preparing to unpack .../hping3_3.a2.ds2-10_amd64.deb ...
Unpacking hping3 (3.a2.ds2-10) ...
Setting up hping3 (3.a2.ds2-10) ...
Processing triggers for man-db (2.10.2-1) ...
altaf@LAPTOP-DGNIK4U9:~$ man hping3
altaf@LAPTOP-DGNIK4U9:~$ man hping3
altaf@LAPTOP-DGNIK4U9:~$ hping3 -l --flood -a 192.168.1.103 192.168.1.255
[open_sockraw] socket(): Operation not permitted
[main] can't open raw socket
altaf@LAPTOP-DGNIK4U9:~$ sudo hping3 -l --flood -a 192.168.1.103 192.168.1.255
[sudo] password for altaf:
HPING 192.168.1.255 (eth0 192.168.1.255): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.1.255 hping statistic ---
352510 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
altaf@LAPTOP-DGNIK4U9:~$ 
```

Below the terminal, the desktop environment shows the following status bar:

- 27°C Heavy rain
- 16:53 08-09-2023

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: Sender 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>"

lab1006@lab1006-HP-280-G4-MT-Business-PC:~$
```

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 manual 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 configuring 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.

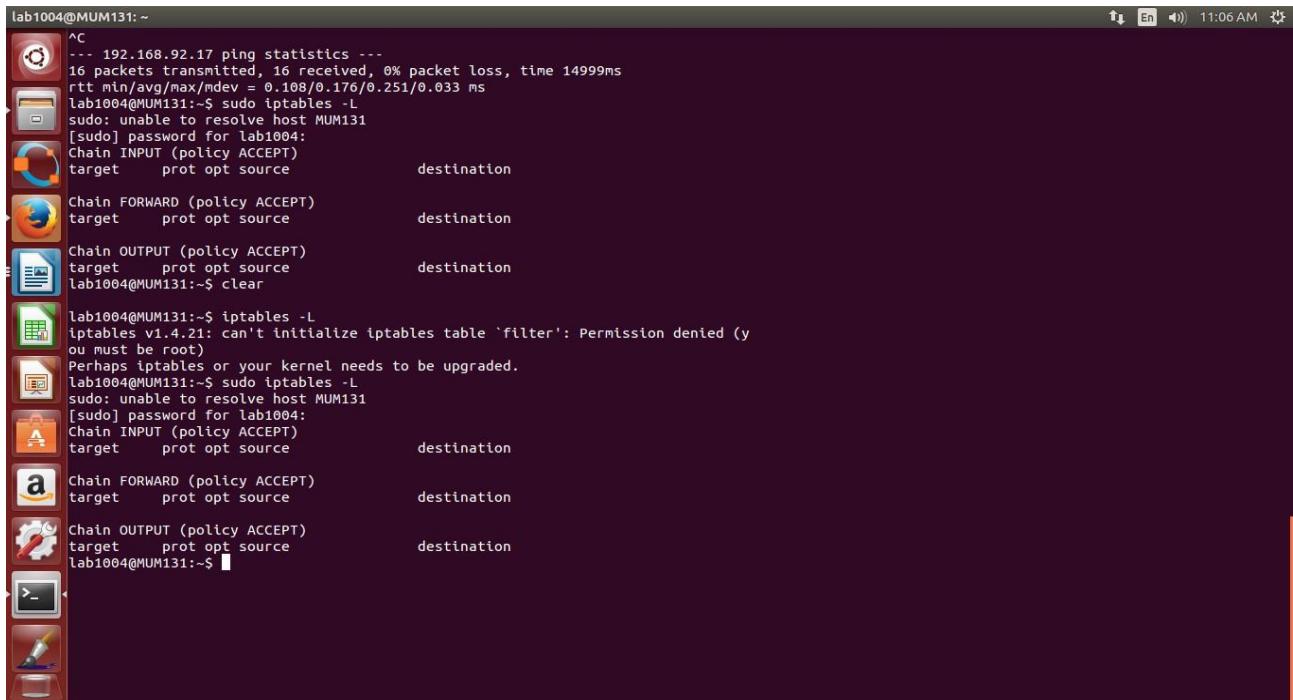
Once the traffic type has been specified, three actions may be taken:

- 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. )



```
lab1004@MUM131:~
^C
--- 192.168.92.17 ping statistics ---
16 packets transmitted, 16 received, 0% packet loss, time 14999ms
rtt min/avg/max/mdev = 0.108/0.176/0.251/0.033 ms
lab1004@MUM131:~$ sudo iptables -L
sudo: unable to resolve host MUM131
[sudo] password for lab1004:
Chain INPUT (policy ACCEPT)
target prot opt source destination
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$ clear

lab1004@MUM131:~$ iptables -L
iptables v1.4.21: can't initialize iptables table `filter': Permission denied (you must be root)
Perhaps iptables or your kernel needs to be upgraded.
lab1004@MUM131:~$ sudo iptables -L
sudo: unable to resolve host MUM131
[sudo] password for lab1004:
Chain INPUT (policy ACCEPT)
target prot opt source destination
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$
```

## 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 5<sup>th</sup> 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)

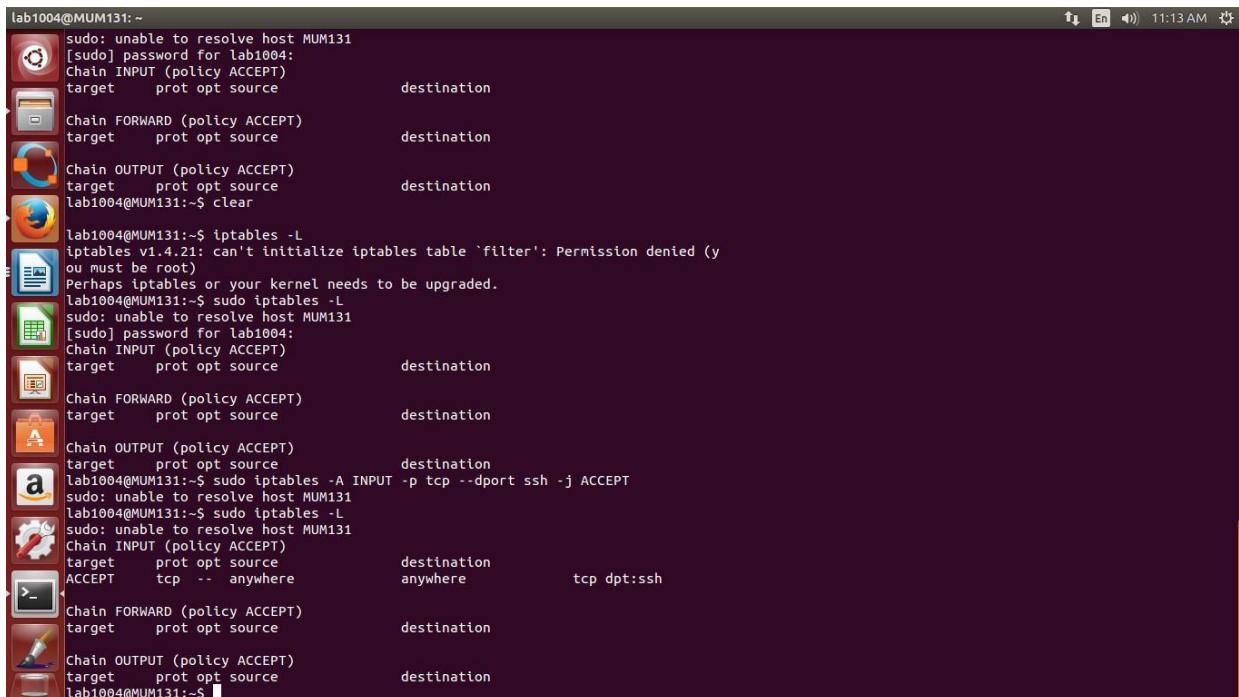
## Allowing Incoming Traffic on Specific Ports

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).

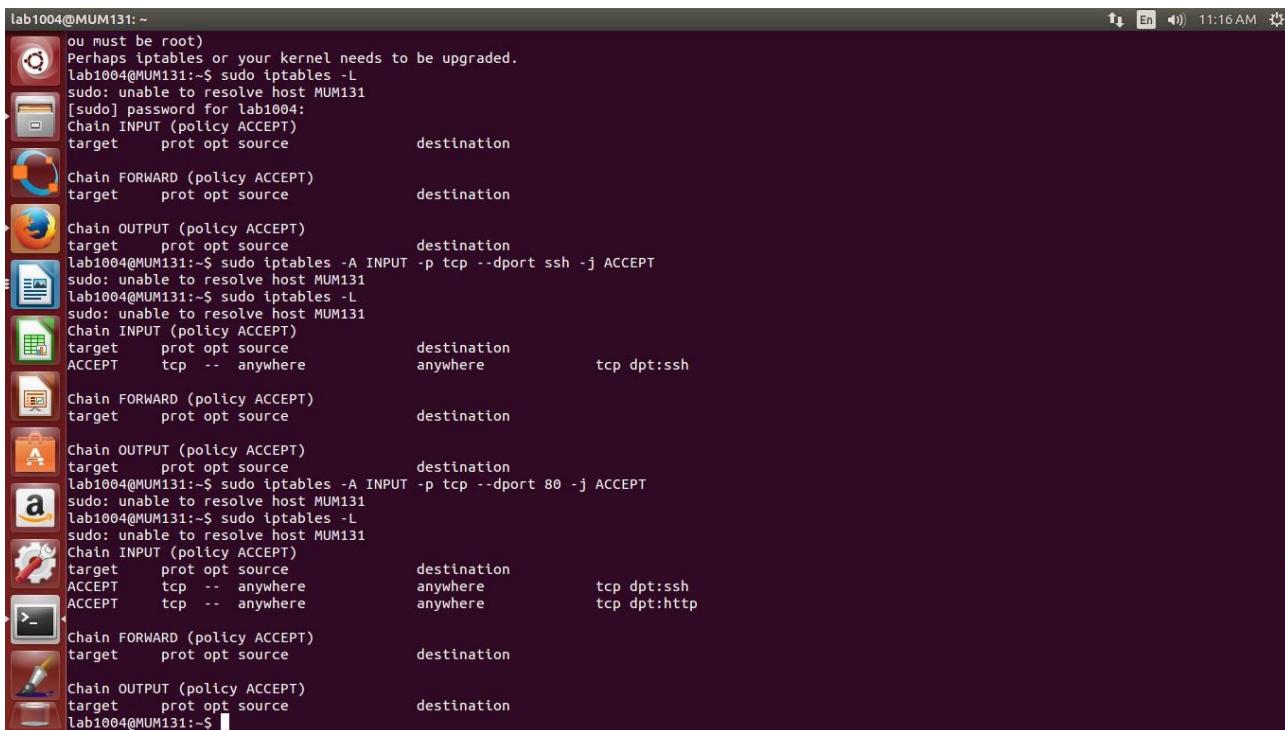


The screenshot shows a terminal window with a dark background and light-colored text. It displays the following command history and output:

```
lab1004@MUM131: ~
sudo: unable to resolve host MUM131
[sudo] password for lab1004:
Chain INPUT (policy ACCEPT)
target prot opt source destination
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$ clear

lab1004@MUM131:~$ iptables -L
iptables v1.4.21: can't initialize iptables table `filter': Permission denied (you must be root)
Perhaps iptables or your kernel needs to be upgraded.
lab1004@MUM131:~$ sudo iptables -L
sudo: unable to resolve host MUM131
[sudo] password for lab1004:
Chain INPUT (policy ACCEPT)
target prot opt source destination
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$ sudo iptables -A INPUT -p tcp --dport ssh -j ACCEPT
sudo: unable to resolve host MUM131
lab1004@MUM131:~$ sudo iptables -L
sudo: unable to resolve host MUM131
Chain INPUT (policy ACCEPT)
target prot opt source destination
ACCEPT tcp -- anywhere anywhere tcp dpt:ssh
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$
```

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



```

lab1004@MUM131: ~
 ou must be root)
Perhaps iptables or your kernel needs to be upgraded.
lab1004@MUM131:~$ sudo iptables -L
[sudo] password for lab1004:
Chain INPUT (policy ACCEPT)
target prot opt source destination
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$ sudo iptables -A INPUT -p tcp --dport ssh -j ACCEPT
sudo: unable to resolve host MUM131
lab1004@MUM131:~$ sudo iptables -L
[sudo] password for lab1004:
Chain INPUT (policy ACCEPT)
target prot opt source destination
 ACCEPT tcp -- anywhere anywhere tcp dpt:ssh
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
sudo: unable to resolve host MUM131
lab1004@MUM131:~$ sudo iptables -L
[sudo] password for lab1004:
Chain INPUT (policy ACCEPT)
target prot opt source destination
 ACCEPT tcp -- anywhere anywhere tcp dpt:ssh
 ACCEPT tcp -- anywhere anywhere tcp dpt:http
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$

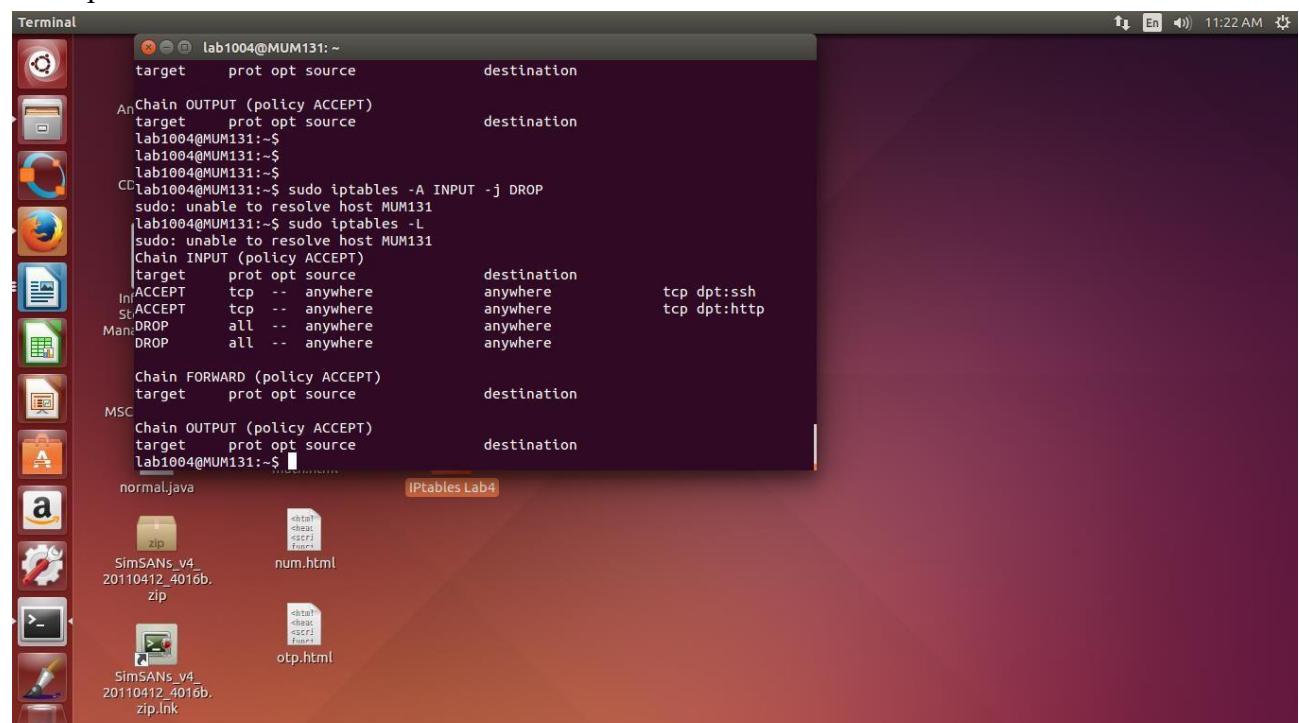
```

## 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
```



```

Terminal lab1004@MUM131: ~
 target prot opt source destination
Chain INPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$
lab1004@MUM131:~$
lab1004@MUM131:~$
C)lab1004@MUM131:~$ sudo iptables -A INPUT -j DROP
[sudo] password for lab1004:
sudo: unable to resolve host MUM131
lab1004@MUM131:~$ sudo iptables -L
[sudo] password for lab1004:
Chain INPUT (policy ACCEPT)
target prot opt source destination
 ACCEPT tcp -- anywhere anywhere tcp dpt:ssh
 ACCEPT tcp -- anywhere anywhere tcp dpt:http
 DROP all -- anywhere anywhere
 DROP all -- anywhere anywhere
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$

```

normal.java

IPTables Lab4

zip

SimsANS\_v4\_20110412\_4016b.zip

num.html

otp.html

SimsANS\_v4\_20110412\_4016b.zip.lnk

## Editing iptables

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

```
lab1004@MUM131: ~
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$
lab1004@MUM131:~$
lab1004@MUM131:~$
lab1004@MUM131:~$ sudo iptables -A INPUT -j DROP
sudo: unable to resolve host MUM131
lab1004@MUM131:~$ sudo iptables -L
sudo: unable to resolve host MUM131
Chain INPUT (policy ACCEPT)
target prot opt source destination
ACCEPT tcp -- anywhere anywhere tcp dpt:ssh
ACCEPT tcp -- anywhere anywhere tcp dpt:http
DROP all -- anywhere anywhere
DROP all -- anywhere anywhere
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$ sudo iptables -I INPUT 1 -i lo -j ACCEPT
sudo: unable to resolve host MUM131
lab1004@MUM131:~$ sudo iptables -L
sudo: unable to resolve host MUM131
Chain INPUT (policy ACCEPT)
target prot opt source destination
ACCEPT all -- anywhere anywhere
ACCEPT tcp -- anywhere anywhere tcp dpt:ssh
ACCEPT tcp -- anywhere anywhere tcp dpt:http
DROP all -- anywhere anywhere
DROP all -- anywhere anywhere
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$
```

we will list iptables in greater detail.

`sudo iptables -L -v`

```
lab1004@MUM131: ~
DROP all -- anywhere anywhere
DROP all -- anywhere anywhere
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$ sudo iptables -I INPUT 1 -i lo -j ACCEPT
sudo: unable to resolve host MUM131
lab1004@MUM131:~$ sudo iptables -L
sudo: unable to resolve host MUM131
Chain INPUT (policy ACCEPT)
target prot opt source destination
ACCEPT all -- anywhere anywhere
ACCEPT tcp -- anywhere anywhere tcp dpt:ssh
ACCEPT tcp -- anywhere anywhere tcp dpt:http
DROP all -- anywhere anywhere
DROP all -- anywhere anywhere
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$ sudo iptables -L -v
sudo: unable to resolve host MUM131
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target prot opt in out source destination
 140 9376 ACCEPT all -- lo any anywhere anywhere
 0 0 ACCEPT tcp -- any any anywhere anywhere tcp dpt:ssh
 0 0 ACCEPT tcp -- any any anywhere anywhere tcp dpt:http
 509 111K DROP all -- any any anywhere anywhere
 0 0 DROP all -- any any anywhere anywhere
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target prot opt in out source destination
Chain OUTPUT (policy ACCEPT 420 packets, 29783 bytes)
 pkts bytes target prot opt in out source destination
lab1004@MUM131:~$
```

**Allow traffic on ICMP port** `sudo iptables -A INPUT -p icmp -j ACCEPT` now list the

rules again.. `sudo iptables -L`

```

lab1004@MUM131: ~
DROP all -- anywhere anywhere
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$ sudo iptables -L -v
sudo: unable to resolve host MUM131
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
 140 9376 ACCEPT all -- lo any anywhere anywhere
 0 0 ACCEPT tcp -- any any anywhere anywhere tcp dpt:ssh
 0 0 ACCEPT tcp -- any any anywhere anywhere tcp dpt:http
 509 111K DROP all -- any any anywhere anywhere
 0 0 DROP all -- any any anywhere anywhere

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
Chain OUTPUT (policy ACCEPT 420 packets, 29783 bytes)
pkts bytes target prot opt in out source destination
lab1004@MUM131:~$ sudo iptables -A INPUT -p icmp -j ACCEPT
sudo: unable to resolve host MUM131
lab1004@MUM131:~$ sudo iptables -L
sudo: unable to resolve host MUM131
Chain INPUT (policy ACCEPT)
target prot opt source destination
ACCEPT all -- anywhere anywhere
ACCEPT tcp -- anywhere anywhere tcp dpt:ssh
ACCEPT tcp -- anywhere anywhere tcp dpt:http
DROP all -- anywhere anywhere
DROP all -- anywhere anywhere
ACCEPT icmp -- anywhere anywhere

Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$

```

clearing all rules

sudo iptables -F sudo

iptables -L

```

lab1004@MUM131: ~
ACCEPT tcp -- anywhere anywhere tcp dpt:ssh
ACCEPT tcp -- anywhere anywhere tcp dpt:http
DROP all -- anywhere anywhere
DROP all -- anywhere anywhere
ACCEPT icmp -- anywhere anywhere

Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$ ping 192.168.92.17
PING 192.168.92.17 (192.168.92.17) 56(84) bytes of data.

^C
--- 192.168.92.17 ping statistics ---
89 packets transmitted, 0 received, 100% packet loss, time 88703ms

lab1004@MUM131:~$ ping 192.168.92.17
PING 192.168.92.17 (192.168.92.17) 56(84) bytes of data.
^C
--- 192.168.92.17 ping statistics ---
8 packets transmitted, 0 received, 100% packet loss, time 7056ms

lab1004@MUM131:~$ sudo iptables -F
sudo: unable to resolve host MUM131
lab1004@MUM131:~$ sudo iptables -L
sudo: unable to resolve host MUM131
Chain INPUT (policy ACCEPT)
target prot opt source destination
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$

```

**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..

```
lab1004@MUM131: ~
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$ ping 192.168.92.17
PING 192.168.92.17 (192.168.92.17) 56(84) bytes of data.
64 bytes from 192.168.92.17: icmp_seq=1 ttl=64 time=0.167 ms
64 bytes from 192.168.92.17: icmp_seq=2 ttl=64 time=0.166 ms
64 bytes from 192.168.92.17: icmp_seq=3 ttl=64 time=0.150 ms
64 bytes from 192.168.92.17: icmp_seq=4 ttl=64 time=0.179 ms
64 bytes from 192.168.92.17: icmp_seq=5 ttl=64 time=0.170 ms
64 bytes from 192.168.92.17: icmp_seq=6 ttl=64 time=0.175 ms
64 bytes from 192.168.92.17: icmp_seq=7 ttl=64 time=0.154 ms
^C
--- 192.168.92.17 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6000ms
rtt min/avg/max/mdev = 0.150/0.165/0.179/0.019 ms
lab1004@MUM131:~$ sudo iptables -A INPUT -p icmp DROP
sudo: unable to resolve host MUM131
Bad argument 'DROP'
Try 'iptables -h' or 'iptables --help' for more information.
lab1004@MUM131:~$ sudo iptables -A INPUT -p icmp -j DROP
sudo: unable to resolve host MUM131
lab1004@MUM131:~$ sudo iptables -L
sudo: unable to resolve host MUM131
Chain INPUT (policy ACCEPT)
target prot opt source destination
DROP icmp -- anywhere anywhere
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$ ping 192.168.92.17
PING 192.168.92.17 (192.168.92.17) 56(84) bytes of data.
^C
--- 192.168.92.17 ping statistics ---
14 packets transmitted, 0 received, 100% packet loss, time 13000ms
lab1004@MUM131:~$
```

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

```

lab1004@MUM131: ~
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$ ping 192.168.92.17
PING 192.168.92.17 (192.168.92.17) 56(84) bytes of data.
^C
--- 192.168.92.17 ping statistics ---
14 packets transmitted, 0 received, 100% packet loss, time 13000ms
lab1004@MUM131:~ ^C
lab1004@MUM131:~ ^C
lab1004@MUM131:~$ sudo iptables -A OUTPUT -p icmp -j DROP
sudo: unable to resolve host MUM131
lab1004@MUM131:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target prot opt source destination
DROP icmp -- anywhere anywhere
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
DROP icmp -- anywhere anywhere
lab1004@MUM131:~$ ping 192.168.92.17
PING 192.168.92.17 (192.168.92.17) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
^C
--- 192.168.92.17 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6047ms
lab1004@MUM131:~$

```

**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

```

lab1004@MUM131: ~
DROP icmp -- anywhere anywhere
ACCEPT icmp -- anywhere anywhere
lab1004@MUM131:~$ sudo iptables -F
sudo: unable to resolve host MUM131
lab1004@MUM131:~$ ping 192.168.92.17
PING 192.168.92.17 (192.168.92.17) 56(84) bytes of data.
64 bytes from 192.168.92.17: icmp_seq=1 ttl=64 time=0.133 ms
64 bytes from 192.168.92.17: icmp_seq=2 ttl=64 time=0.115 ms
64 bytes from 192.168.92.17: icmp_seq=3 ttl=64 time=0.119 ms
64 bytes from 192.168.92.17: icmp_seq=4 ttl=64 time=0.136 ms
64 bytes from 192.168.92.17: icmp_seq=5 ttl=64 time=0.133 ms
^C
--- 192.168.92.17 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3999ms
rtt min/avg/max/mdev = 0.115/0.127/0.136/0.011 ms
lab1004@MUM131:~$ sudo iptables -A INPUT -p tcp -j DROP
sudo: unable to resolve host MUM131
lab1004@MUM131:~$ sudo iptables -L
sudo: unable to resolve host MUM131
Chain INPUT (policy ACCEPT)
target prot opt source destination
DROP tcp -- anywhere anywhere
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$ sudo iptables -F
sudo: unable to resolve host MUM131
lab1004@MUM131:~$ sudo iptables -L
sudo: unable to resolve host MUM131
Chain INPUT (policy ACCEPT)
target prot opt source destination
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
lab1004@MUM131:~$

```

**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 your 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 destination 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

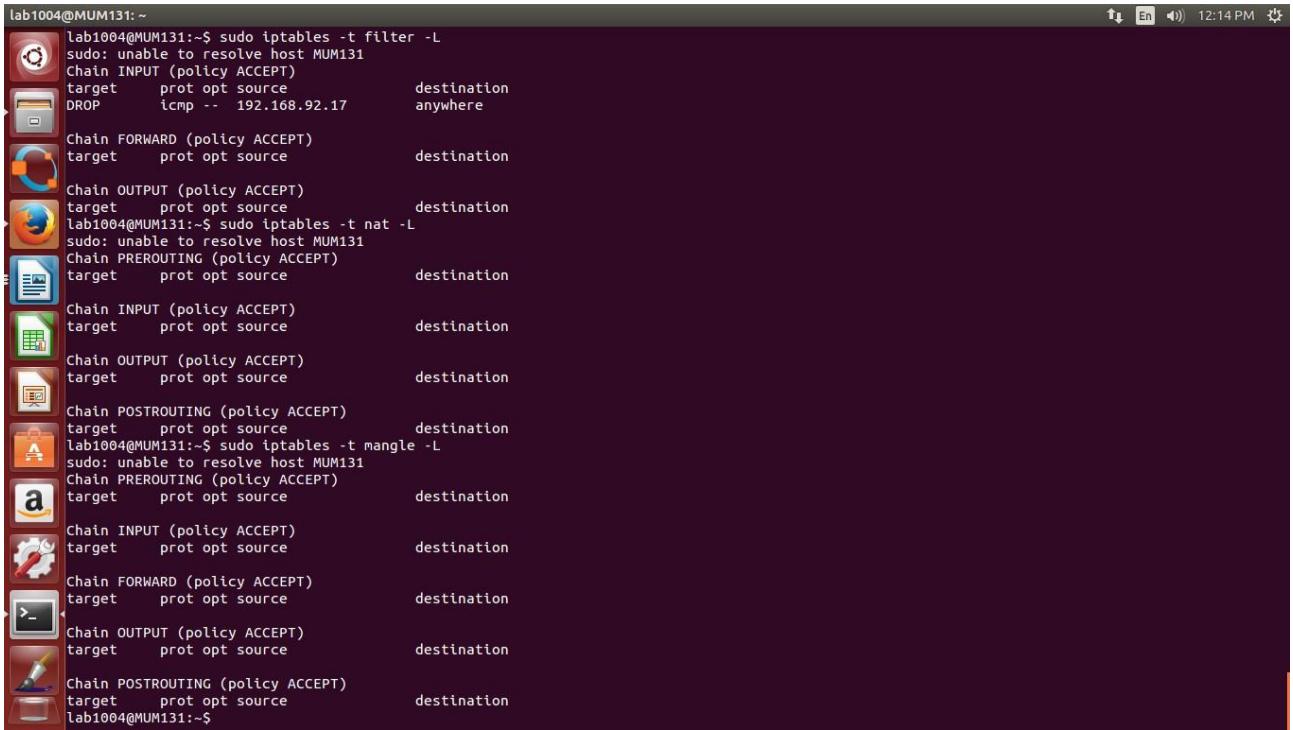
Type the following command and see the result

```
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



lab1004@MUM131: ~

```
lab1004@MUM131:~$ sudo iptables -t filter -L
sudo: unable to resolve host MUM131
Chain INPUT (policy ACCEPT)
target prot opt source destination
DROP icmp -- 192.168.92.17 anywhere

Chain FORWARD (policy ACCEPT)
target prot opt source destination

Chain OUTPUT (policy ACCEPT)
target prot opt source destination

lab1004@MUM131:~$ sudo iptables -t nat -L
sudo: unable to resolve host MUM131
Chain PREROUTING (policy ACCEPT)
target prot opt source destination

Chain INPUT (policy ACCEPT)
target prot opt source destination

Chain OUTPUT (policy ACCEPT)
target prot opt source destination

Chain POSTROUTING (policy ACCEPT)
target prot opt source destination

lab1004@MUM131:~$ sudo iptables -t mangle -L
sudo: unable to resolve host MUM131
Chain PREROUTING (policy ACCEPT)
target prot opt source destination

Chain INPUT (policy ACCEPT)
target prot opt source destination

Chain FORWARD (policy ACCEPT)
target prot opt source destination

Chain OUTPUT (policy ACCEPT)
target prot opt source destination

Chain POSTROUTING (policy ACCEPT)
target prot opt source destination

lab1004@MUM131:~$
```

## 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.