Information Security Lab Contents

Lab Instructor Engr. Zia Ur Rehman

Lab 01

Introduction to Encryption and Decryption Techniques and Python Implementation

- **CAESAR CIPHER**
  - Mathematical Calculations and understanding behind the cipher
  - Encryption using Python
  - Decryption using Python
  - **Caesar Cipher Tool: Overview**: This project provides a command-line interface for users to encrypt and decrypt messages using a specified shift value.
- **PLAYFAIR CIPHER**
  - Mathematical Calculations and understanding behind the cipher
  - Encryption using Python
  - Decryption using Python
  - **Encryptify: Playfair Cipher Utility:** To create a tool that utilizes the Playfair cipher for encrypting and decrypting messages, enhancing communication security.

Lab 02

HILL CIPHER

- **Mathematical Calculations and Understanding Behind the Cipher**:
- **Encryption Using Python**
- **Decryption Using Python**:
- **Hill Cipher Tool**: Overview: This project provides a command-line interface for users to encrypt and decrypt messages using the Hill cipher, emphasizing matrix operations for secure communication.

VIGENERE CIPHER

- **Mathematical Calculations and Understanding Behind the Cipher**:
- **Encryption Using Python**:
- **Decryption Using Python**:
- **Vigenère Cipher Utility**: Overview: This tool enables users to securely encrypt and decrypt messages using the Vigenère cipher, enhancing privacy through keyword-based encryption.

Lab 03

RAIL FENCE CIPHER

- **Mathematical Calculations and Understanding Behind the Cipher**:
- **Encryption Using Python**:
- **Decryption Using Python**:
- **Rail Fence Cipher Tool**: Overview: This utility allows users to encrypt and decrypt messages using the Rail Fence cipher, providing a simple method of transposition for enhanced message confidentiality.

ONE-TIME PAD CIPHER

- **Mathematical Calculations and Understanding Behind the Cipher**:
- **Encryption Using Python**:
- **Decryption Using Python**:
- **One-Time Pad Utility**: Overview: This tool facilitates secure encryption and decryption using the One-Time Pad cipher, ensuring absolute confidentiality when a truly random key is utilized.

Lab 04

DATA ENCRYPTION STANDARDS (DES)

- **Mathematical Calculations and Understanding Behind the Cipher**:
- **Encryption Using Python**:
- **Decryption Using Python**:
- **DES Utility**: Overview: This utility allows users to securely encrypt and decrypt data using the Data Encryption Standard (DES), a historically significant encryption method used for data protection.

ADVANCED ENCRYPTION STANDARDS (AES)

- **Mathematical Calculations and Understanding Behind the Cipher**:
- **Encryption Using Python**:
- **Decryption Using Python**:
- **AES Utility**: Overview: This tool provides users with the ability to securely encrypt and decrypt data using the Advanced Encryption Standard (AES), a widely accepted standard for secure communications.

Lab 05

RSA ALGORITHM

- **Mathematical Calculations and Understanding Behind the Cipher**:
- **Encryption Using Python**:
- **Decryption Using Python**:
- **RSA Utility**: Overview: This project enables secure communication using the RSA algorithm, allowing users to encrypt and decrypt messages with public and private keys for enhanced security.

DIFFIE-HELLMAN KEY EXCHANGE

- **Mathematical Calculations and Understanding Behind the Cipher**:
- **Key Exchange Using Python**:
- **Diffie-Hellman Utility**: Overview: This utility facilitates secure key exchange between parties using the Diffie-Hellman method, ensuring that a shared secret can be established securely without prior shared information.

SHA (Secure Hash Algorithm)

- **Mathematical Calculations and Understanding Behind the Cipher**:
- **Hashing Using Python**:
- **SHA Utility**: Overview: This tool allows users to generate SHA hashes for files or messages, providing a method to ensure data integrity and authenticity.

Lab 06

Introduction to Socket Programming in Python

Error Handling in Sockets: Understanding common errors and exceptions in socket communication is crucial for identifying potential vulnerabilities and ensuring robust security in networked applications.

Security Considerations: This topic covers basic socket security measures, including encryption protocols like SSL/TLS, which are essential for protecting data in transit and safeguarding against eavesdropping and man-in-the-middle attacks.

Client-Server Communication: Designing secure communication protocols between clients and servers is fundamental for preventing unauthorized access, ensuring data integrity, and implementing authentication mechanisms.

Multi-threading with Sockets: Handling multiple clients in a secure manner is vital for preventing issues such as race conditions and ensuring that sensitive data is managed correctly across concurrent connections.

Lab 07

Introduction to Network Scanning

- **Introduction to Scapy**
  - Crafting and sending custom packets.
  - Sniffing network traffic in real-time.
  - Decoding and analyzing various network protocols.
  - Performing network discovery and scanning.
  - Conducting penetration testing and security assessments.
- **Introduction to Wireshark Tool to monitor network Traffic**
  - Real-Time Packet Capture
  - Detailed Packet Analysis
  - Powerful Filtering Options
  - Protocol Decoding
  - Visualizations and Statistics

Lab 08

Introduction N-Map, Zenmap for monitoring network Traffic

Port-Scanner-Cyber-Security-Project

his Python project implements a multi-threaded port scanner that identifies open ports on a specified target IP address. It leverages the `socket` module to attempt connections to various ports and the `threading` module to perform scans concurrently, significantly speeding up the scanning process.

Key Features:

- **Target Specification**: The target IP address can be defined, with the default set to the loopback address (`127.0.0.1`) for local testing.
- **Port Queue**: A `Queue` is utilized to manage the list of ports to scan, allowing for organized and efficient handling of multiple ports.
- **Modes of Scanning**: Users can choose from different scanning modes:
  - Scan common well-known ports (1-1023)
  - Scan registered and dynamic/private ports (1-49151)
  - Scan a predefined list of important ports
  - Input custom ports for scanning
- **Threaded Execution**: The scanner employs multiple threads (up to 200) to perform simultaneous port checks, increasing efficiency and reducing the total time required for scanning.
- **Open Port Detection**: For each port checked, if a connection is successful, it is marked as open, and the results are printed to the console along with a list of all open ports found during the scan.

This project serves as a practical demonstration of network programming, concurrency, and the use of Python for network security applications.

Lab 09

Network Tracking Using Wireshark and Google Maps

In this project, I will show you Network Traffic visualization using the Python programming language, Wireshark and Google Maps. This tutorial covers the implementation steps needed to take a file of network traffic and convert it into a visual presentation using Google Maps.

Lab 10

Building-a-TCP-Chat-Room

DOS-DDOS-Simulation-Using-Python

SQL Injection Simulation Using Python

Lab 11

Attack Classification Using Machine Learning and Deep Learning Algorithms

---