**Security Programs**

**1.a) Aim:**  Program to send a message (Ex: Mid exams are from next Monday) to your friend.

**Program:**

**MyServer.py**

import socket

class MyServer:

def main(self):

try:

ss = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

ss.bind(('localhost', 9999))

ss.listen(1)

print("Server is listening on port 9999...")

s, addr = ss.accept()

print(f"Connection from {addr} has been established.")

str\_data = s.recv(1024).decode('utf-8')

print("message= " + str\_data)

s.close()

ss.close()

except Exception as e:

print(e)

if \_name\_ == "\_main\_":

server = MyServer()

server.main()

**Output:**

python server.py

Server is listening for incoming connections...

Connected to client at: ('127.0.0.1', 55436)

Message sent, server closed connection.

**client.py**

import socket

def main():

try:

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

s.connect(("localhost", 9999))

message = "midexams are from next monday"

s.sendall(message.encode('utf-8'))

s.close()

except Exception as e:

print(e)

if \_name\_ == "\_main\_":

main()

**Output:**

c:/>java MyServer

message: Mid exams are from next monday

**1.b) Aim:**  Program for the above one to show that the message has been modified (attack). (ex: “Mid exams are postponed”).

**Program:**

**MyServer.py**

import socket

def main():

try:

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_socket.bind(('localhost', 9999))

server\_socket.listen(1)

print("Server is waiting for a connection...")

client\_socket, addr = server\_socket.accept()

print(f"Connection established with {addr}")

data = client\_socket.recv(1024).decode('utf-8')

message = "mid exams are postponed"

print(f"message= {message}")

client\_socket.close()

server\_socket.close()

except Exception as e:

print(e)

if \_name\_ == "\_main\_":

main()

**MyClient.py**

import socket

try:

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

s.connect(('localhost', 9999))

message = "midexams are from next monday"

s.sendall(message.encode('utf-8'))

s.close()

except Exception as e:

print(e)

Output**:**

**2.a) Aim:** Program so that sender and receiver are sharing the same key (Symmetric Encryption).

pip install pycryptodome

**Cryptography.py**

from Crypto.Cipher import DES

from Crypto.Util.Padding import pad, unpad

import base64

def encrypt(input\_string, key):

cipher = DES.new(key.encode('utf-8')[:8], DES.MODE\_ECB)

encrypted\_bytes = cipher.encrypt(pad(input\_string.encode('utf-8'), DES.block\_size))

return base64.b64encode(encrypted\_bytes).decode('utf-8')

def decrypt(encrypted\_string, key):

cipher = DES.new(key.encode('utf-8')[:8], DES.MODE\_ECB)

decrypted\_bytes = unpad(cipher.decrypt(base64.b64decode(encrypted\_string)), DES.block\_size)

return decrypted\_bytes.decode('utf-8')

def main():

input\_str = "Hello, World!"

key = "secretKey" # 8-byte key required for DES

encrypted\_value = encrypt(input\_str, key)

print("Encrypted:", encrypted\_value)

decrypted\_value = decrypt(encrypted\_value, key)

print("Decrypted:", decrypted\_value)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**2.b) Aim :** Program so that sender sends the message and receiver decrypts the message with the above key.

pip install cryptography

**server.py**

import socket

from cryptography.fernet import Fernet

DES\_ENCRYPTION\_KEY = b'YOUR\_GENERATED\_FERNET\_KEY\_HERE'

def main():

try:

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as ss:

ss.bind(('localhost', 6666))

ss.listen()

print("Server waiting for client...")

conn, addr = ss.accept()

with conn:

data = conn.recv(1024)

str\_received = data.decode('utf-8')

print("\n\nServer Side")

print(f"Message from Client: {str\_received}")

decrypted = decrypt(str\_received, DES\_ENCRYPTION\_KEY)

print(f"Decrypted Message: {decrypted}")

except Exception as e:

print(str(e))

def decrypt(encrypted\_message, key):

f = Fernet(key)

decrypted\_message = f.decrypt(encrypted\_message.encode())

return decrypted\_message.decode()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Output:**

**client.py**

import socket

from cryptography.fernet import Fernet

import base64

DES\_ENCRYPTION\_KEY = "testString"

def main():

try:

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

s.connect(('localhost', 6666))

input\_message = input("Enter message to send: ")

print("Client Side")

print("Message from Client:", input\_message)

encrypted = encrypt(input\_message, DES\_ENCRYPTION\_KEY)

print("Encrypted Message:", encrypted)

s.sendall(encrypted.encode())

s.close()

except Exception as e:

print(e)

def encrypt(message, key):

fernet\_key = base64.urlsafe\_b64encode(key.encode().ljust(32)[:32])

fernet = Fernet(fernet\_key)

return fernet.encrypt(message.encode()).decode()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Output:**

**3) Aim:** Program which shows the public-key cryptography technique. Note: generate two keys, public and private and encrypt with one key and decrypt with the other key.

**Program:**

import math

def main():

alphabet = "0ABCDEFGHIJKLMNOPQRSTUVWXYZ"

print("Enter two prime numbers p and q:")

p = int(input())

q = int(input())

while True:

n = p \* q

if n >= 26:

break

print("\nn value is not large enough.\nPlease select p, q values such that p\*q is greater than 26")

p = int(input())

q = int(input())

z = (p - 1) \* (q - 1)

print("Enter the value of d:")

d = int(input())

# Find e such that (e \* d) % z == 1

e = None # Initialize e

for j in range(1, z):

if (j \* d) % z == 1:

e = j

break

if e is None:

print("No valid public key e found. Please check the value of d.")

return

print(f"n={n}, z={z}, e={e}")

print("ENCRYPTION-CIPHERTEXT")

print("Enter the plain text:")

plaintext = input().upper()

print("Cipher: ", end="")

ciphertext = []

for char in plaintext:

if char in alphabet:

s = alphabet.index(char)

k2 = pow(s, e, n)

ciphertext.append(k2)

print(k2, end=" ")

print()

print("Decipher: ", end="")

for k2 in ciphertext:

m = pow(k2, d, n)

print(alphabet[m], end="")

print()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Output:**

c:/>java RSA1

Enter 2 prime numbers p and q

3

11

Enter the value of d:7

N=33 z=20 e=3.0

ENCRYPTION-CIPHERTEXT

Enter the Plain text

ABC

Cipher:1 8 27

ABC

**4.a) Aim:** Program to compute the sum of integers from 1 to 10.

**Program:**

sum = 0

for i in range(1, 11):

sum += i

print("The sum of numbers from 1 to 10 is:", sum)

**Output:**

Sum is : 55

**4.b) Aim:** Program to show that the above program can be sabotaged so that during execution it computes a different sum from 3 to 20. (Program security)

**Program:**

**server.py**

import socket

import struct

def main():

try:

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_socket.bind(('localhost', 9999))

server\_socket.listen(1)

print("Server waiting for client...")

client\_socket, \_ = server\_socket.accept() # establishes connection

# Receive two integers from the client

c = struct.unpack('!i', client\_socket.recv(4))[0]

d = struct.unpack('!i', client\_socket.recv(4))[0]

# Calculate sum

sum\_result = sum(range(c + 2, d + 11))

print(f"Sum = {sum\_result}")

server\_socket.close()

except Exception as e:

print(e)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Output:**

**client.py**

import socket

import struct

try:

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

s.connect(('localhost', 9999))

a, b = 1, 10

data = struct.pack('>ii', a, b) # Pack two integers in big-endian format

s.sendall(data)

s.close()

except Exception as e:

print(e)

**Output:**

Sum=207

**7) Aim**: Program to demonstrate man-in-middle attack. **Same as 1b**

**8)** **Write a program to demonstrate the denial of service to the authorised user in the lab (Administrative Security)**

import socket

def main():

target\_host = "127.0.0.1" # Replace with the IP address or hostname of the target

target\_port = 80 # Replace with the port number of the service you want to attack

num\_connections = 100 # Number of simultaneous connections to establish

print("Initiating Denial of Service attack simulation...")

for i in range(num\_connections):

try:

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

sock.connect((target\_host, target\_port))

print(f"Connection #{i + 1} established to {target\_host}:{target\_port}")

except Exception as e:

print(f"Failed to connect: {e}")

print("Denial of Service attack simulation complete.")

if \_name\_ == "\_main\_":

main()