Experiment No. 7

Title: Challenge-Response Protocol

Batch: B-2 **Roll No.: 16010422234** **Experiment No.: 7**

Title: Design and implement a program for Challenge-Response protocol.

Resources needed: Windows/Linux OS

Theory:

Pre Lab/ Prior Concepts:

Consider a situation where a server (for example, a base station) wants to authenticate a client (a mobile phone user) by confirming that the client has the correct password.

Figure - Challange-Response Protocol

Assume there are malicious eaves-droppers who can hear the communication that is taking place. A simple authentication method is as follows: The server generates a random 3-digit random number and sends it to the client. The client computes the remainder (Ra mod Rb) and sends the result to the server. The server also computes the value (Ra mod Rb) and if it gets the same result, it concludes that the client has the correct password and authenticates the client as shown in figure 1. (Consider Ra as Alice's random number and Rb as Bob's random number).

Procedure / Approach /Algorithm / Activity Diagram:

Refer to the VIRTUAL LAB of EXPERIMENT NO. 6 simulation (https://cse29-iiith.vlabs.ac.in/exp/diffie-hellman/simulation.html) and implement the above authentication protocol shown in the figure 1 in the similar way.

```
import socket
import random

# Server-side (Bob)
# Function to handle modular arithmetic
def mod_exp(number, mod):
    return number % mod

# Bob's secret or number based on the shared password
R_a = 1567  # Bob knows Alice's number based on the password

# Create a TCP/IP socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_ip = '0.0.0.0'  # Listen on all available interfaces (e.g., '192.168.x.x')
server_port = 65432
server_socket.bind((server_ip, server_port))
server_socket.listen(1)

print(f"Bob's server is listening on port {server_port}...")

# Wait for Alice (client) to connect
conn, addr = server_socket.accept()
print(f"Connected by Alice at {addr}")

# Bob generates a random 3-digit number (Rb)
R_b = random.randint(100, 999)
print(f"Bob sends R_b = {R_b} to Alice.")
conn.sendall(str(R_b).encode())  # Send R_b to Alice

# Receive Alice's computation of (R_a mod R_b)
response = conn.recv(1024).decode()
remainder_from_alice = int(response)
print(f"Received Alice's remainder: {remainder_from_alice}")

# Bob computes the correct value of (R_a mod R_b)
remainder_from_bob = mod_exp(R_a, R_b)
print(f"Bob's computed remainder: {remainder_from_bob}")

# Compare the results
if remainder_from_alice == remainder_from_bob:
```

```
    print("Authentication successful!")
    conn.sendall("Authenticated".encode())  # Send success message
else:
    print("Authentication failed.")
    conn.sendall("Authentication failed".encode())  # Send failure message

# Close the connection
conn.close()
server_socket.close()
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SEARCH ERROR

PS C:\Users\OS_IT_B317\Desktop\234> python -u "c:\Users\OS_IT_B317\Desktop\234\bob.py"
Bob's server is listening on port 65432...
Connected by Alice at ('172.17.17.32', 49263)
Received Alice's remainder: 197
Bob's computed remainder: 197
Authentication successful!
PS C:\Users\OS_IT_B317\Desktop\234> python -u "c:\Users\OS_IT_B317\Desktop\234\bob.py"
Bob's server is listening on port 65432...
Connected by Alice at ('10.0.32.116', 54175)
Bob sends R_b = 750 to Alice.
Received Alice's remainder: 67
Bob's computed remainder: 67
Authentication successful!
```

Questions:

1. **What are the advantages and disadvantages of the above authentication method?**
   Advantages:
   - ➢ Simplicity: This method is straightforward to implement and understand.
   - ➢ Dynamic: Each session generates a new random number, making it difficult for attackers to reuse previous responses.
   - ➢ Efficiency: Requires minimal computation, which is beneficial for devices with limited computational power.

   Disadvantages:
   - ➢ Vulnerable to Replay Attacks: If an attacker captures the random challenge and the response, they can reuse it to authenticate falsely if the challenge is repeated.
   - ➢ Susceptibility to Man-in-the-Middle Attacks: An active attacker can intercept and modify the challenge or response in transit.
   - ➢ No Encryption: The challenge and response are sent in plain text, potentially exposing them to interception.

2. **Explain replay attack on this protocol?**
   In a replay attack, an attacker intercepts a valid data transmission and retransmits it to open a session in the guise of the legitimate user. In the context of this protocol, if an

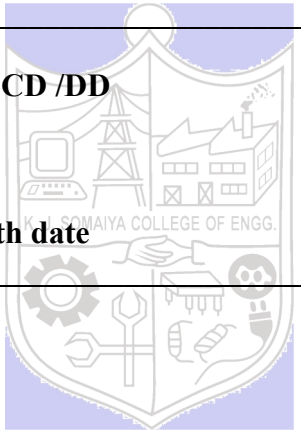
attacker captures the random challenge from the server and the response from the client, they can later use this captured data to authenticate themselves with the server as if they were the legitimate client. This is especially effective if the server does not keep track of used challenges or does not implement time stamps to ensure challenges are fresh.

Outcomes: Describe various access control policies and models

---

Conclusion: The Challenge-Response protocol provides a basic yet efficient method for authentication by using a simple challenge based on a randomly generated number and the modulo operation. While this method offers the advantages of being lightweight and dynamic, it inherently lacks robust security measures against more sophisticated attacks such as replay and man-in-the-middle attacks. To enhance security, additional features like timestamps, nonces, and encrypted communication channels should be considered in practical implementations. This experiment highlights the importance of balancing efficiency and security in the design of authentication protocols, particularly in environments where computational resources and bandwidth may be limited.

Grade: AA / AB / BB / BC / CC / CD /DD



Signature of faculty in-charge with date

References:

Books/ Journals/ Websites:

- Mark Stamp, “Information security Principles and Practice” Wiley.
- https://cse29-iiith.vlabs.ac.in/exp/diffie-hellman/simulation.html

---