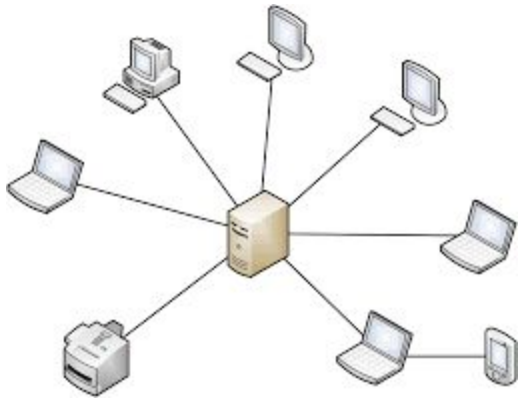
Client Server(SSL/TLS)



by

Abhishek Bhagate

Aditya Singh

Rajendra Prajapat

Jul 7, 2020

INTRODUCTION

The Client-server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters called clients. In the client-server architecture, when the client computer sends a request for data to the server through the internet, the server accepts the requested process and delivers the data packets requested back to the client.

How Client-Server Works

In a Client-Server Architecture, we have a model which consists of two fundamental parts – A Client System and A Server System. The client process always initiates a connection to the server, while the server process always waits for requests from any client. When both the client process and server process are running on the same computer, this is called a single seat setup.

A server host runs one or more server programs, which share their resources with clients. A client does not share any of its resources, but it requests content or service from a server. Clients, therefore, initiate communication sessions with servers.

SSL/TLS Layer

SSL, or Secure Sockets Layer, is an encryption-based Internet security protocol. It was first developed by Netscape in 1995 for the purpose of ensuring privacy, authentication, and data integrity in Internet communications. SSL is the predecessor to the modern TLS encryption used today.

An SSL handshake basically consists of following steps between the server and the client :

1. Client Hello - The client will send the information that will be required by the server to start an HTTPS connection.

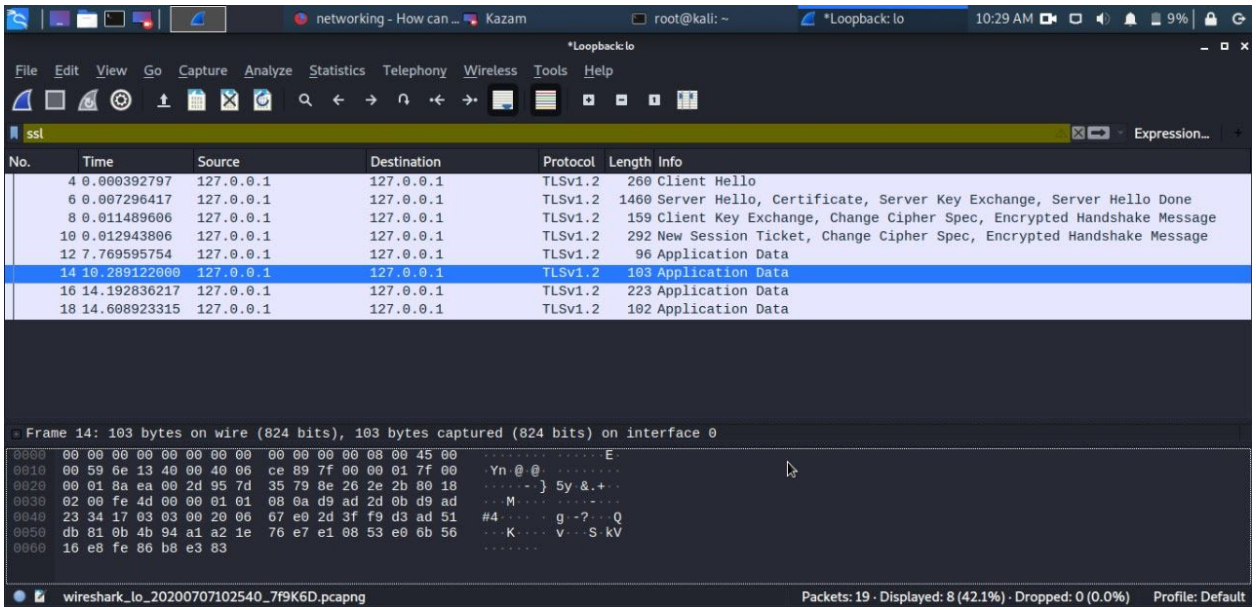
2. Server Hello - The server will respond back with the configuration it selected from the Client Hello along with its information to proceed with the handshake.

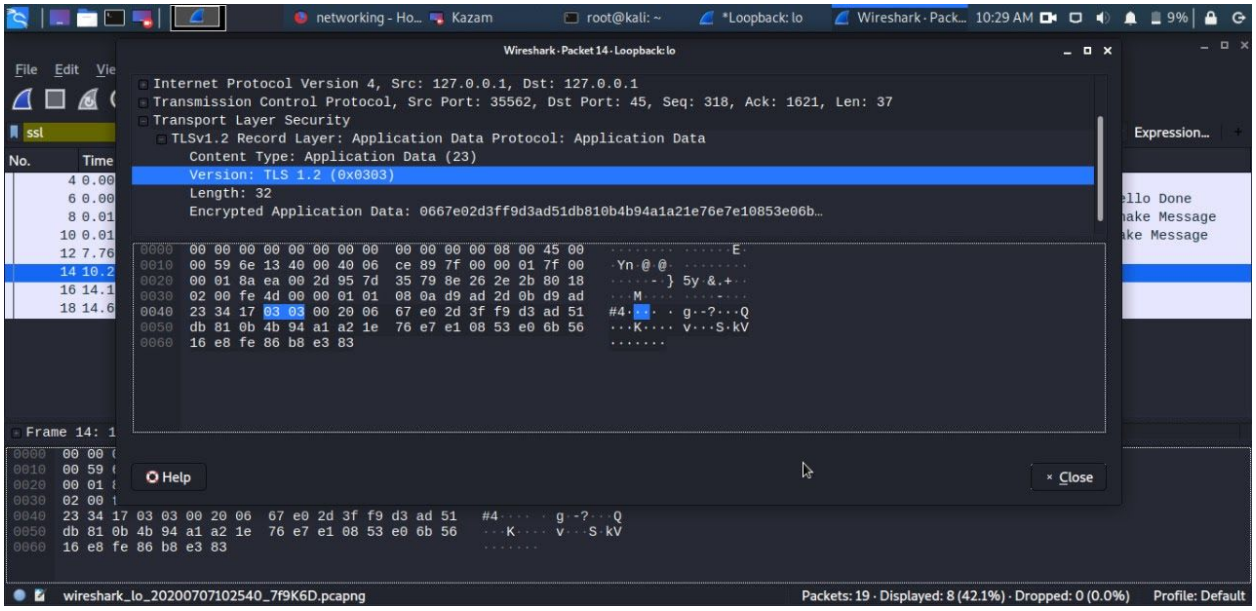
3. Server Key Exchange Message - This message will be sent by the server to the client carrying the required details for the client to generate the pre-master secret. This message will not be sent if the RSA key exchange algorithm or any other key exchange algorithms are used that do not require the information from the server to generate a pre-master secret.

4. Certificate Request - During this step, the server will send a certificate request from the client with the certificate type, certificate signature algorithms and certificate authorities supported by the server. There can be situations where the certificate authorities list can be empty. In such scenarios, the client may choose whether to send or avoid sending the client certificate (depends on the client implementation).

5. Client Certificate - The client presents its certificate chain to the server. The certificate needs to be appropriate for the negotiated cipher suite's key exchange algorithm, and any negotiated extensions.

6. Client Key Exchange Message -This message needs to be sent by the client following the Client Certificate message. If the client certificate is not being presented (in one-way SSL), the client key exchange message should be sent after the client receives the ServerHelloDone message.





About the project

In this project we have implemented a multithreaded, client-server project over ssl-tls layer. In this project, to store the password, we used SHA-512 hashing algorithm, so if the database is compromised, then also the attacker can not get the real passwords.This project implements basic operations such as data retrieval,

updation and deletion from the database.

The Project has an interface in which the user will be presented with a Login/ Signup portal. After successful login, the user can retrieve, view, update and delete his/her information stored in the database. The database used on the server side is SQLite3 database which stores all the information of signed up users and their accounts.

REFERENCES

1. https://itpeernetwork.intel.com/top-10-reasons-to-setup-a-client-server-network/#gs.9nhhbn
2. https://www.websecurity.digicert.com/en/in/security-topics/what-is-ssl-tls-https
3. https://www.wikipedia.org/