**Slide 2**

## Network Layer

Network layer is a layer of TCP/IP model and is also known as an internet layer. The main work of this layer is to send the packets from any network, and any computer still they reach the destination irrespective of the route they take.

## Transport Layer

Transport layer builds on the network layer in order to provide data transport from a process on a source system machine to a process on a destination system. It is hosted using single or multiple networks, and also maintains the quality of service functions.

**Application Layer**

Application layer interacts with an application program, which is the highest level of TCP/IP model. The application layer is closest to the end-user. It means the application layer allows users to interact with other software application.

Security is provided at network layer by using IPSec protocol, at transport layer by using SSL or TLS and at application layer by using Kerberos, S/MIME, PGP (pretty good privacy), SET, etc.

**Slide 3**

The Secure Sockets Layer (SSL) protocol was first introduced by Netscape in 1994. The Internet was growing and there was a need for transport security for web browsers and for various TCP protocols. Version 1.0 of SSL was never released because it had serious security flaws. The first official release of SSL, version 2.0, was out in 1995. The final version of the SSL protocol, SSL 3.0, was released in November 1996. SSL version 3 is defined in RFC 6101 and its implementation is given at openssl.org.

**Slide 4**

In general while using SSL it first establishes a session which includes information on algorithms to be used for key exchange, encryption, hash etc, secrets including certificates & keys, and authentication & verification is performed.

When previous steps are successful, it ensures data privacy and integrity therefore application data can be transferred.

**Slide 5**

Architecture of SSL is shown in the diagram. For secure communication, initialization of algorithms to be used is done by change cipher protocol. Establishing a session is done by handshake protocol. Record Protocol is used for data compression, transfer application data & TLS information and alert protocol is used for error handling.

**Slide 6**

The main purpose of an SSL handshake is to provide privacy and data integrity for communication between a server and a client. During the Handshake, server and client will exchange important information required to establish a secure connection. Firstly, negotiation on choice of Cipher-Suite Algorithms, Symmetric cipher, Key exchange method and Message digest function to be used is done. Secondly, creating/ establishing & sharing master secret and thirdly, authentication of server and/or client is done.

**Slide 7**

Mainly three phases are involved in handshake. In first phase hello messages are exchanged, in second phase shared secrets are exchanged and in third phase the verification/ confirmation/ authentication is done.

**Slide 8**

SSL handshake is a 10-step process.

1. In first step the client who wants to communicate with the server sends a hello message to server.
2. On receipt of hello message from client, server sends a hello message to client.
3. Server sends certificate and chain to certificate authority (CA) root.
4. Server then share its own certificate (public key) to the client.
5. Server negotiation is completed.
6. Client shares symmetric key with server using public key of server.
7. Client confirms to activate encryption thereafter.
8. Client negotiation and handshake is completed here.
9. Then server activates the encryption in next messages.
10. Server handshake is completed.

**Slide 9**

Client hello message includes 4 parts. Protocol version, a random number which is 32 bytes long & is used primarily against replay attacks, session ID which is 32 bytes long which includes information on cryptographic material used previously and lastly, a compression algorithm.

**Slide 10**

Server hello message includes 5 parts. Protocol version, a random number which is again used primarily against replay attacks, session ID which is provided to resume the session later on, choice of cipher suite which is usually done as per client’s choice and lastly, a compression algorithm.

**Slide 11**

* During handshake certificates shared are:
  + Sequence of X.509 certificates
  + X.509 Certificate associates public key with identity
  + CA creates and sign cert according to policies
  + User of Certificate must ensure it is valid
* In client Key exchange a premaster secret is shared which is created by client which is:
  + - 2 bytes of SSL version + 46 random bytes
    - Sent encrypted to server using server’s public key
* In change cipher specifications it switches to newly negotiated algorithms and key material
* And lastly in finish message:
  + First message encrypted with new crypto parameters is sent
  + Digest of negotiated master secret, the ensemble of handshake messages, sender constant
  + HMAC approach of nested hashing is done

**Slide 12**

In the SSL Record Protocol application data is divided into fragments. The fragment is compressed and then encrypted MAC (Message Authentication Code) generated by algorithms like SHA (Secure Hash Protocol) and MD5 (Message Digest) is appended. After that encryption of the data is done and in last SSL header is appended to the data.

**Slide 13**

The final step of SSL Record Protocol processing is to prepend a header, consisting of:

* Content Type (8 bits): The higher layer protocol used to process the enclosed fragment.
* SSL Version (8 bits): Indicates version of SSL in use.
* Content Length (16 bits): The length in bytes of the plaintext fragment (or compressed fragment if compression is used). The maximum value is 214 + 2048.

**Slide 14**

* MAC (message authentication code) has maximum record length of 214 – 1. It is calculated from data, headers and sequence number
* In alert protocol it alerts the other side of any errors. It terminates and session cannot be resumed.
* The closure/ finished notification is used to prevent truncation attack (sending a TCP FIN before the sender is finished)