**Chapter 6**

• The World Wide Web is fundamentally a client/server application running over the Internet and TCP/IP intranets

• The following characteristics of Web usage suggest the need for tailored security tools:

• Web servers are relatively easy to configure and manage

• Web content is increasingly easy to develop

• The underlying software is extraordinarily complex

• May hide many potential security flaws

• A Web server can be exploited as a launching pad into the corporation’s or agency’s entire computer complex

• Casual and untrained (in security matters) users are common clients for Web-based services

• Such users are not necessarily aware of the security risks that exist and do not have the tools or knowledge to take effective countermeasures

Table

Description automatically generated

**Relative Location of Security facilities in the TCP/IP Protocol Stack**

Table

Description automatically generated

**Transport Layer Security (TLS)**

• One of the most widely used security services

• TLS is an Internet standard that evolved from a commercial protocol known as Secure Sockets Layer (SSL)

• TLS is a general-purpose service implemented as a set of protocols that rely on TCP

• TLS either provided as part of the underlying protocol suite or therefore be transparent to applications

• TLS can be embedded in specific packages

Table

Description automatically generated

**TLS Architecture**

**TLS Connection:** • A transport that provides a suitable type of service

• For TLS such connections are peer-to-peer relationships

• Connections are transient • Every connection is associated with one session

**TLS Session:** • An association between a client and a server • Created by the Handshake Protocol

• Define a set of cryptographic security parameters which can be shared among multiple connections

• Are used to avoid the expensive negotiation of new security parameters for each connection

**Session State parameters:**

Diagram

Description automatically generated

**Connection State Parameters:**

Text

Description automatically generated with medium confidence

**TLS Record Protocol**

The TLS Record Protocol provides two services for TLS connections

* **Confidentiality:** The Handshake Protocol defines a shared secret key that is used for conventional encryption of TLS payloads
* **Message integrity:** The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC)

**TLS Record Protocol Operation**

Diagram

Description automatically generated

**SSL Record Format**

Diagram

Description automatically generated

**TLS Record Protocol Payload**

Graphical user interface, application

Description automatically generated

**TLS Protocols**

**• The Change Cipher Spec Protocol:** it is the simplest, consists of a single message, which consists of a single byte with the value 1. The sole purpose of this message is to cause the pending state to be copied into the current state, which updates the cipher spec to be used on this connection.

**• The Alert Protocol:** used to convey TLS-related alerts to the peer entity. Each message consists of two bytes. The first byte takes the value warning (1) or fatal (2) to convey the severity of the message. If the level is fatal, TLS immediately terminates the connection. Other connections on the same session may continue, but no new connections on this session may be established. The second byte contains a code that indicates the specific alert.

**• The Handshake Protocol:** Allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in a TLS record. The Handshake Protocol is used before any application data is transmitted. The Handshake Protocol messages have the format shown in the following slide

**TLS Handshake Protocol Message Types**

Graphical user interface, text, application

Description automatically generated with medium confidence

Diagram

Description automatically generated

**Cryptographic computations**

• Two further items are of interest:

• The creation of a 1) pre-master key by means of the key exchange 2) shared master secret using the pre-master key. The shared master secret is a one-time 48-byte value generated for this session

• The generation of cryptographic parameters from the master secret

• CipherSpecs require a client write MAC secret, a server write MAC secret, a client write key, a server write key, a client write IV, and a server write IV which are generated from the master secret in that order

• These parameters are generated from the master secret by hashing the master secret into a sequence of secure bytes of sufficient length for all needed parameters

**Heartbeat Protocol**

• A heartbeat is a periodic signal generated by hardware or software to indicate normal operation or to synchronize other parts of a system

• A heartbeat protocol is typically used to monitor the availability of a protocol entity

• The heartbeat protocol runs on top of the TLS Record Protocol

• Consists of two message types: heartbeat request and heartbeat response

• The heartbeat serves two purposes:

• It assures the sender that the recipient is still alive, even though there may not have been any activity over the underlying TCP connection

• It generates activity across the connection during idle periods, which avoids closure by a firewall that does not tolerate idle connections

**SSL/TLS Attacks**

Attack categories: • Attacks on the handshake protocol • Attacks on the record and application data protocols • Attacks on the PKI • Other attacks

**HTTPS (HTTP Over SSL)**

• Refers to the combination of HTTP and SSL to implement secure communication between a Web browser and a Web server • The HTTPS capability is built into all modern Web browsers

• A user of a Web browser will see URL addresses that begin with https:// rather than http://

• If HTTPS is specified, port 443 is used, which invokes SSL • Documented in RFC 2818, HTTP Over TLS

• There is no fundamental change in using HTTP over either SSL or TLS and both implementations are referred to as HTTPS

• When HTTPS is used, the following elements of the communication are encrypted:

• URL of the requested document

• Contents of the document

• Contents of browser forms

• Cookies sent from browser to server and from server to browser

• Contents of HTTP header

**Connection Initiation**

Text

Description automatically generated

**Connection Closure**

• An HTTP client or server can indicate the closing of a connection by including the line Connection: close in an HTTP record

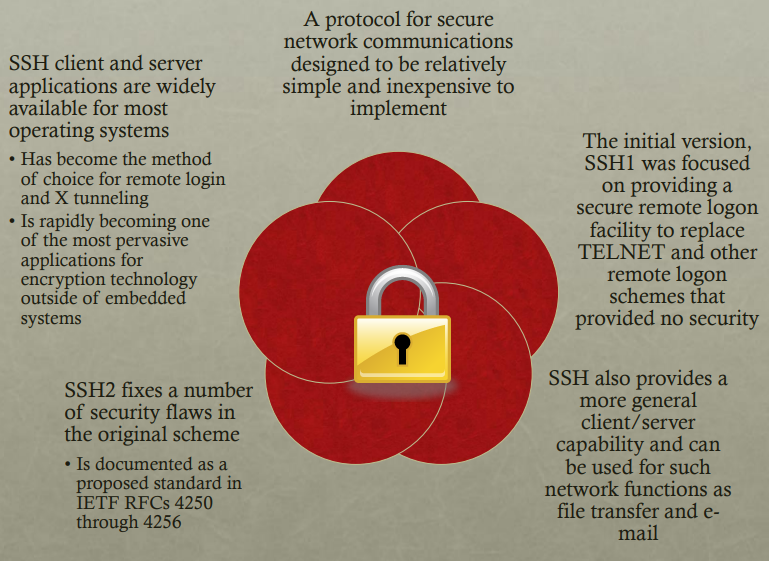
• The closure of an HTTPS connection requires that TLS close the connection with the peer TLS entity on the remote side, which will involve closing the underlying TCP connection

• TLS implementations must initiate an exchange of closure alerts before closing a connection

• A TLS implementation may, after sending a closure alert, close the connection without waiting for the peer to send its closure alert, generating an “incomplete close”.

• An unannounced TCP closure could be evidence of some sort of attack so the HTTPS client should issue some sort of security warning when this occurs

**Secure Shell (SSH)**



**SSH Protocol Stack**

Table

Description automatically generated with medium confidence

**Transport Layer Protocol**

• Server authentication occurs at the transport layer, based on the server possessing a public/private key pair

• A server may have multiple host keys using multiple different asymmetric encryption algorithms

• The server host key is used during key exchange to authenticate the identity of the host

• RFC 4251 dictates two alternative trust models:

• The client has a local database that associates each host name with the corresponding public host key

• The host name-to-key association is certified by a trusted certification authority (CA); the client only knows the CA root key and can verify the validity of all host keys certified by accepted Cas

**Transport Layer Protocol Packet Exchanges**

Diagram

Description automatically generated with medium confidence

• The key exchange method allows to establish a shared secret K and the hash value h which are used to derive the SSH session keys:

• The hash h of the initial key exchange is also taken as the session\_id

• IVClient2Server = Hash(K, h, “A”, session\_id) // initialization vector

• IVServer2Client = Hash(K, h, “B”, session\_id) // initialization vector

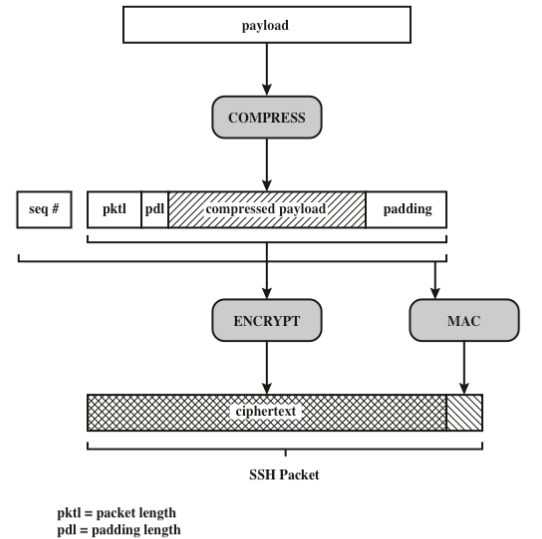
• EKClient2Server = Hash(K, h, “C”, session\_id) // encryption key

• EKServer2Client = Hash(K, h, “D”, session\_id) // encryption key

• IKClient2Server = Hash(K, h, “E”, session\_id) // integrity key

• IKServer2Client = Hash(K, h, “F”, session\_id) // integrity key

**SSH Transport Layer Protocol Packet Formation**



Table

Description automatically generated

**Authentication Methods**

1. **Public Key:** • The client sends a message to the server that contains the client’s public key, with the message signed by the client’s private key • When the server receives this message, it checks whether the supplied key is acceptable for authentication and, if so, it checks whether the signature is correct
2. **Password:** • The client sends a message containing a plaintext password, which is protected by encryption by the Transport Layer Protocol
3. **Hostbased:** • Authentication is performed on the client’s host rather than the client itself • This method works by having the client send a signature created with the private key of the client host • Rather than directly verifying the user’s identity, the SSH server verifies the identity of the client host

**Connection Protocol**

• The SSH Connection Protocol runs on top of the SSH Transport Layer Protocol and assumes that a secure authentication connection is in use

• The secure authentication connection, referred to as a tunnel, is used by the Connection Protocol to multiplex several logical channels

• Channel mechanism

• All types of communication using SSH are supported using separate channels

• Either side may open a channel

• For each channel, each side associates a unique channel number

• Channels are flow controlled using a window mechanism

• No data may be sent to a channel until a message is received to indicate that window space is available

• The life of a channel progresses through three stages: opening a channel, data transfer, and closing a channel

**SSH Connection Protocol Message Exchange**

Diagram

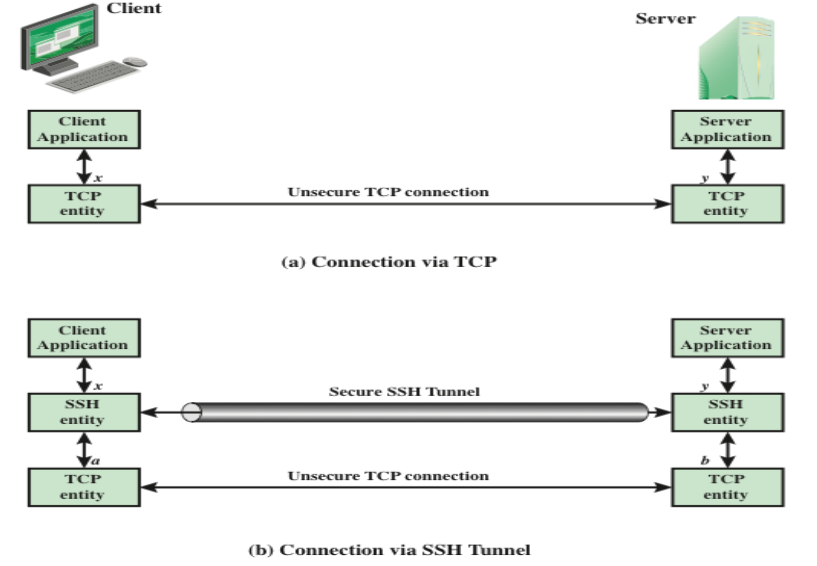
Description automatically generated

**Channel Types:**

1. **Session:** • The remote execution of a program • The program may be a shell, an application such as file transfer or e-mail, a system command, or some built-in subsystem • Once a session channel is opened, subsequent requests are used to start the remote program
2. **X11:** • Refers to the X Window System, a computer software system and network protocol that provides a graphical user interface (GUI) for networked computers • X allows applications to run on a network server but to be displayed on a desktop machine
3. **Forwarded-tcpip:** • Remote port forwarding
4. **Direct-tcpip:** • Local port forwarding

**Port Forwarding:**

• One of the most useful features of SSH • Provides the ability to convert any insecure TCP connection into a secure SSH connection (also referred to as SSH tunneling) • Incoming TCP traffic is delivered to the appropriate application based on the port number (a port is an identifier of a user of TCP) • An application may employ multiple port numbers



SSH supports two types of port forwarding:   
**• Local forwarding:** SSH Client will intercept selected application-level traffic and redirect it from an unsecured TCP connection to a secure SSH tunnel. On the other end, the SSH server sends the incoming traffic to the destination port dictated by the client application. Example: call to restricted web server through firewall

ssh -L 127.0.0.1:80:restricted.example.com:80 ssherver.example.com

• **Remote forwarding:** the user’s SSH client acts on the server’s behalf. The client receives traffic with a given destination port number, places the traffic on the correct port and sends it to the destination the user chooses. Example: access your home computer from the internet through firewall or through NAT.

ssh -R 8080:localhost:80 public.example.com