Chapter 7
Transport-Level Security

Lectured by
Nguyễn Đức Thái
Outline

- Web Security Issues
- Security Socket Layer (SSL)
- Transport Layer Security (TLS)
- HTTPS
- Secure Shell (SSH)
Secure Socket Layer (SSL) provides **security services between TCP and applications** that use TCP.

The Internet standard version is called Transport Layer Service (TLS).

SSL/TLS provides **confidentiality** using **symmetric encryption** and message **integrity** using a **message authentication code**.

SSL/TLS includes **protocol mechanisms** to enable two **TCP users** to determine the security mechanisms and **services** they will use.
Overview (2/2)

- **HTTPS** (HTTP over SSL) refers to the combination of **HTTP** and **SSL** to implement **secure communication between** a **Web browser** and a **Web server**.

- Secure Shell (**SSH**) provides **secure remote logon** and **other** secure client/server **facilities**.
Web Security

- Web now widely used by business, government, individuals
- but *Internet & Web are vulnerable*
- have a *variety of threats*
  - integrity
  - confidentiality
  - denial of service
  - authentication

➔ need *added security mechanisms*
Web Security

- One way to group these threats is in terms of **passive** and **active** attacks.
- Passive attacks include **eavesdropping** on network traffic between browser and server and gaining access to information on a Web site that is supposed to be restricted.
- Active attacks include **impersonating** another user, **altering messages** in transit between client and server, and **altering information** on a website.
- Another way to classify Web security threats is in terms of the **location** of the threat: **Web server**, **Web browser**, and **network traffic** between browser and server.
Web Traffic Security Approaches

- One way to provide Web security is to use IP security (IPsec) (Figure a). The advantage of using IPsec is that it is transparent to end users and applications and provides a general-purpose solution.

- Furthermore, IPsec includes a filtering capability so that only selected traffic need incur the overhead of IPsec processing.
Web Traffic Security Approaches

- Another relatively general-purpose solution is to implement security just above TCP (Figure b). The foremost example of this approach is the Secure Sockets Layer (SSL) and the follow-on Internet standard known as Transport Layer Security (TLS).

- At this level, there are two implementation choices. For full generality, SSL (or TLS) could be provided as part of the underlying protocol suite and therefore be transparent to applications.

- Alternatively, SSL can be embedded in specific packages. For example, Netscape and Microsoft Explorer browsers come equipped with SSL, and most Web servers have implemented the protocol.
SSL

- Netscape originated SSL.
- Version 3 of the protocol was designed with public review and input from industry and was published as an Internet draft document.
- Subsequently, when a consensus was reached to submit the protocol for Internet standardization, the TLS working group was formed within IETF to develop a common standard.
SSL Architecture

- SSL is designed to make use of TCP to provide a **reliable end-to-end secure service**.
- SSL is not a single protocol but rather **two layers** of protocols,
SSL Architecture

SSL Handshake Protocol

SSL Change Cipher Spec Protocol

SSL Alert Protocol

HTTP

SSL Record Protocol

TCP

IP
SSL Architecture

Provides two services:
- **Confidentiality**: The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads.
- **Message Integrity**: The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).
SSL Architecture

- Two important SSL concepts are the **SSL session** and the **SSL connection**, which are defined in the specification as follows.
  
  **Connection:**
  - connections are **peer-to-peer** relationships.
  - The connections are **transient**.
  - Every connection is associated with one session.

  **Session:**
  - between a client and a server.
  - Sessions are created by the Handshake Protocol.
  - Sessions define a **set of** cryptographic security **parameters** which can be shared among multiple connections.
SSL Record Protocol

- The SSL Record Protocol provides two services for SSL connections:
  - **Confidentiality**: The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads.
  - **Message Integrity**: The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).
SSL Record Protocol Services

- confidentiality
  - using symmetric encryption with a shared secret key defined by Handshake Protocol
  - AES, IDEA, RC2-40, DES-40, DES, 3DES, Fortezza, RC4-40, RC4-128
  - message is *compressed* before encryption

- message integrity
  - using a MAC with shared secret key
  - similar to HMAC but with different padding
SSL Record Protocol Operation
The Change Cipher Spec Protocol is one of the three SSL-specific protocols that use the SSL Record Protocol, and it is the simplest.

The sole purpose of this message is to cause the pending state to be copied into the current state, which updates the cipher suite to be used on this connection.
The Alert Protocol is used to **convey** SSL-related alerts **to the peer entity**.

As with other applications that use SSL, **alert messages are compressed** and **encrypted**, as specified by the current state.
SSL Handshake Protocol

- The **most complex** part of SSL is the Handshake Protocol.

- This protocol allows the **server** and **client**
  - to **authenticate each other** and
  - to **negotiate an encryption and MAC algorithm** and
  - To **negotiate cryptographic keys** to be used to protect data sent in an SSL record.

- The Handshake Protocol is used **before** any application data is transmitted
SSL Handshake Protocol

- Comprises a series of messages in phases
  - Establish Security Capabilities
  - Server Authentication and Key Exchange
  - Client Authentication and Key Exchange
  - Finish
SSL Handshake Protocol

**Phase 1**
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.

**Phase 2**
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.

**Phase 3**
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.

**Phase 4**
Change cipher suite and finish handshake protocol.

Note: Shaded transfers are optional or situation-dependent messages that are not always sent.
Cryptographic Computations

Two further items are of interest:

- the **creation of a shared master secret** by means of the key exchange and
  - a one-time 48-byte value
  - generated using secure key exchange (RSA / Diffie-Hellman) and then hashing info

- the **generation of cryptographic parameters** from the master secret.
  - 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
  - generated by hashing master secret
**TLS**

- TLS is an IETF standardization initiative whose goal is to produce an Internet standard version of SSL
- with minor differences
  - in record format version number
  - uses HMAC for MAC
  - a pseudo-random function expands secrets
    - based on HMAC using SHA-1 or MD5
  - has additional alert codes
  - some changes in supported ciphers
  - changes in certificate types & negotiations
  - changes in crypto computations & padding
HTTPS

- **HTTPS (HTTP over SSL)**
  - combination of HTTP & SSL/TLS to secure communications between browser & server
    - documented in RFC2818
    - no fundamental change using either SSL or TLS
  - use **https://** URL rather than **http://**
    - and **port 443** rather than 80
  - encrypts
    - URL, document contents, form data, cookies, HTTP headers
HTTPS Use

- **connection initiation**
  - TLS handshake then HTTP request(s)

- **connection closure**
  - have “Connection: close” in HTTP record
  - TLS level exchange close_notify alerts
  - can then close TCP connection
  - must handle TCP close before alert exchange sent or completed
SSH (Secure Shell)

- **protocol** for secure network communications
  - designed to be simple & inexpensive
- SSH1 provided secure remote logon facility
  - replace TELNET & other insecure schemes
  - also has more general client/server capability
- SSH2 fixes a number of security flaws
- documented in RFCs 4250 through 4254
- SSH clients & servers are widely available
- method of choice for remote login/ X tunnels
SSH Protocol Stack

**SSH User Authentication Protocol**
Authenticates the client-side user to the server.

**SSH Connection Protocol**
Multiplexes the encrypted tunnel into several logical channels.

**SSH Transport Layer Protocol**
Provides server authentication, confidentiality, and integrity. It may optionally also provide compression.

**TCP**
Transmission control protocol provides reliable, connection-oriented end-to-end delivery.

**IP**
Internet protocol provides datagram delivery across multiple networks.
server **authentication** occurs at transport layer, based on server/host key pair(s)

- server authentication requires clients to know host keys in advance

**packet exchange**

- establish TCP connection
- can then exchange data
  - identification string exchange, algorithm negotiation, key exchange, end of key exchange, service request
- using specified packet format
SSH User Authentication Protocol

- authenticates client to server
- three message types:
  - SSH_MSG_USERAUTH_REQUEST
  - SSH_MSG_USERAUTH_FAILURE
  - SSH_MSG_USERAUTH_SUCCESS
- authentication methods used
  - public-key, password, host-based
SSH Connection Protocol

- runs on SSH Transport Layer Protocol
- assumes secure authentication connection
- used for multiple logical channels
  - SSH communications use separate channels
  - either side can open with unique id number
  - flow controlled
  - have three stages:
    - opening a channel, data transfer, closing a channel
  - four types:
    - session, x11, forwarded-tcpip, direct-tcpip.
SSH Connection Protocol Exchange

**Establish Authenticated Transport Layer Connection**

- **Open a channel**
  - SSH.MSG_CHANNEL_OPEN
  - SSH.MSG_CHANNEL_OPEN_CONFIRMATION
  - SSH.MSG_CHANNEL_DATA
  - SSH.MSG_CHANNEL_DATA
  - SSH.MSG_CHANNEL_DATA
  - SSH.MSG_CHANNEL_DATA

**Data transfer**

**Close a channel**

- SSH.MSG_CHANNEL_CLOSE
Port Forwarding

- **convert** insecure TCP connection **into** a secure SSH connection
  - SSH Transport Layer Protocol establishes a TCP connection between SSH client & server
  - client traffic redirected to local SSH, travels via tunnel, then remote SSH delivers to server
- supports **two types** of port forwarding
  - **local forwarding** – hijacks selected traffic
  - **remote forwarding** – client acts for server
Summary

We have discussed:

- Web Security Issues
- Security Socket Layer (SSL)
- Transport Layer Security (TLS)
- HTTPS
- Secure Shell (SSH)
References