Network Security - Week 3

João Soares

DCC/FCUP

2023

Web Security Considerations

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

Web Security Considerations

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

Tailored security tools are necessary

- Web servers easy to configure and manage
- Web content increasingly easy to develop
- Underlying software extraordinarily complex
- Security flaws may be hidden

Web Security Considerations

A Web server can be exploited as a **launching pad** into the corporation/agency's entier computer complex

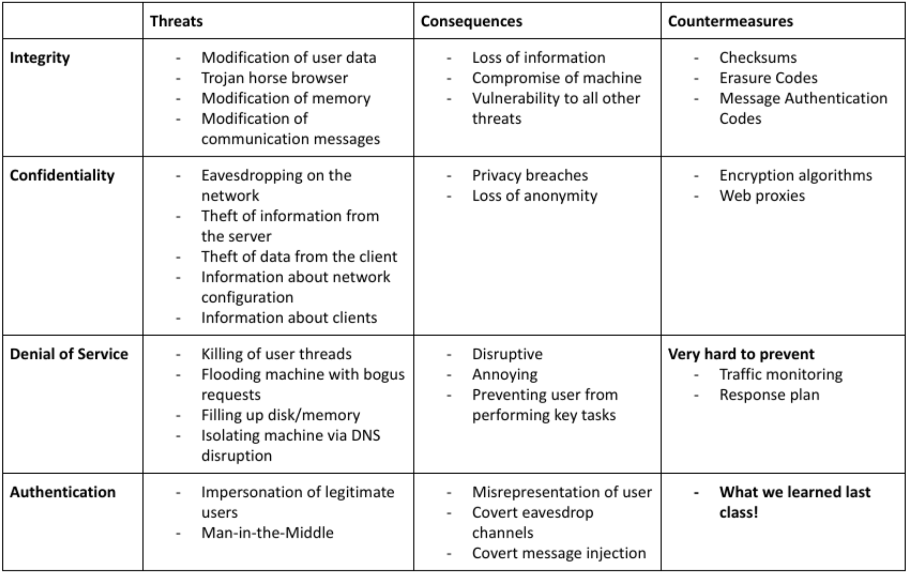
Web Security Considerations

A Web server can be exploited as a **launching pad** into the corporation/agency's entier computer complex

Casual/untrained users for web-based services

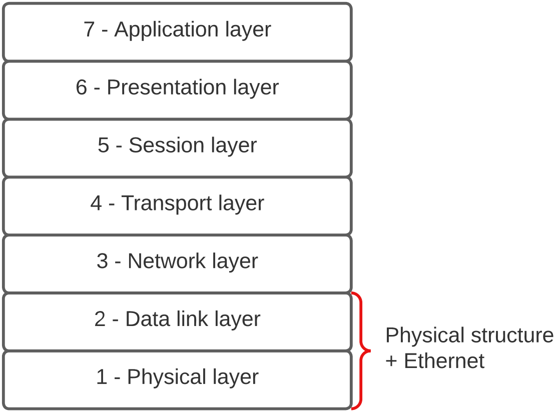
- Not aware of the security rists
- Don't have the tools/knowledge to take effective countermeasures...

Web threats - a quick list



	Threats	Consequences	Countermeasures
Integrity	- Modification of user data - Trojan horse browser - Modification of memory - Modification of communication messages	- Loss of information - Compromise of machine - Vulnerability to all other threats	- Checksums - Erasure Codes - Message Authentication Codes
Confidentiality	- Eavesdropping on the network - Theft of information from the server - Theft of data from the client - Information about network configuration - Information about clients	- Privacy breaches - Loss of anonymity	- Encryption algorithms - Web proxies
Denial of Service	- Killing of user threads - Flooding machine with bogus requests - Filling up disk/memory - Isolating machine via DNS disruption	- Disruptive - Annoying - Preventing user from performing key tasks	**Very hard to prevent** - Traffic monitoring - Response plan
Authentication	- Impersonation of legitimate users - Man-in-the-Middle	- Misrepresentation of user - Covert eavesdrop channels - Covert message injection	- **What we learned last class!**

Open Systems Interconnection Layers



7 - Application layer

6 - Presentation layer

5 - Session layer

4 - Transport layer

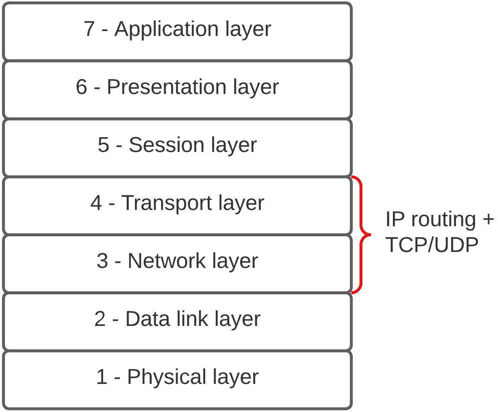
3 - Network layer

2 - Data link layer

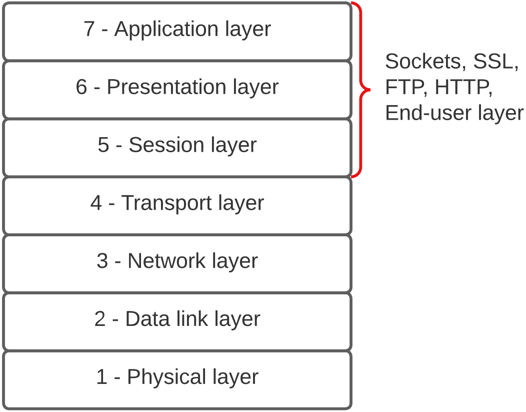
1 - Physical layer

Physical structure
+ Ethernet

Open Systems Interconnection Layers



Open Systems Interconnection Layers



7 - Application layer

6 - Presentation layer

5 - Session layer

4 - Transport layer

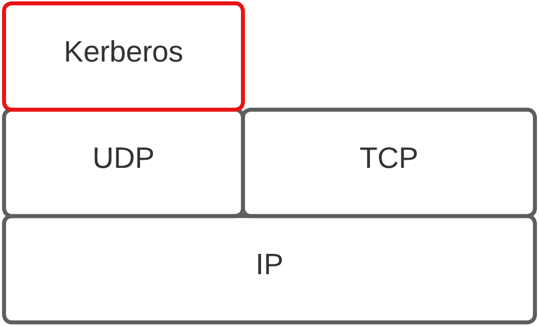
3 - Network layer

2 - Data link layer

1 - Physical layer

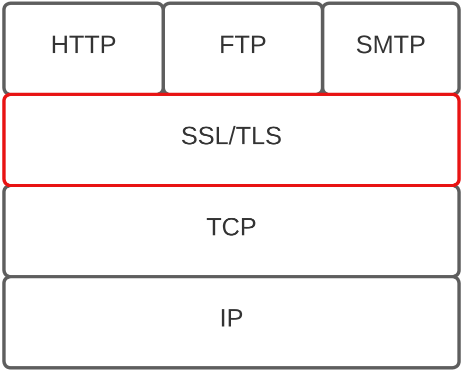
Sockets, SSL,
FTP, HTTP,
End-user layer

Security at the OSI Layers



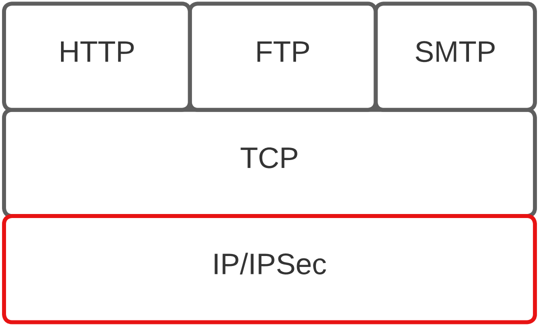
- Kerberos is at the application level - over UDP

Security at the OSI Layers



- SSL/TLS is a middleware between application and TCP

Security at the OSI Layers



- IPSec refines the IP protocol

What is SSL?

- Secure Sockets Layer (SSL) is the protocol used for the majority of secure internet transactions today

What is SSL?

- Secure Sockets Layer (SSL) is the protocol used for the majority of secure internet transactions today
- For instance, if you want to buy a book at *amazon.com* ...
  - You want to be sure you are talking with Amazon (authentication)

What is SSL?

- Secure Sockets Layer (SSL) is the protocol used for the majority of secure internet transactions today
- For instance, if you want to buy a book at *amazon.com* ...
  - You want to be sure you are talking with Amazon (authentication)
  - Credit card data must be protected (confidentiality + integrity)

What is SSL?

- Secure Sockets Layer (SSL) is the protocol used for the majority of secure internet transactions today
- For instance, if you want to buy a book at *amazon.com* ...
  - You want to be sure you are talking with Amazon (authentication)
  - Credit card data must be protected (confidentiality + integrity)
  - If payment is successful, Amazon does not care who you are
  - ... no need for mutual authentication

SSL and TLS

General-purpose system implemented as a set of protocols that **rely on TCP** to ensure message delivery guarantees

SSL and TLS

General-purpose system implemented as a set of protocols that **rely on TCP** to ensure message delivery guarantees

Implementation choices:

- Part of the underlying protocol suite
- Embedded in specific packages

SSL and TLS

General-purpose system implemented as a set of protocols that **rely on TCP** to ensure message delivery guarantees

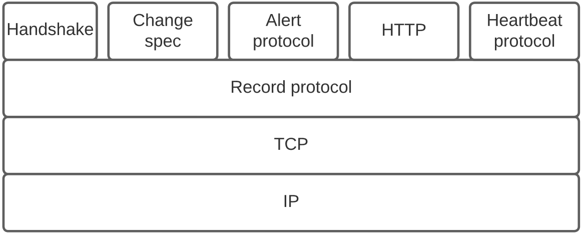
Implementation choices:

- Part of the underlying protocol suite
- Embedded in specific packages

Transport Layer Security

- Evolved from the commercial protocol SSL
- Improved configurability, protocols, ...

SSL/TLS Protocol Stack



Handshake

Change spec

Alert protocol

HTTP

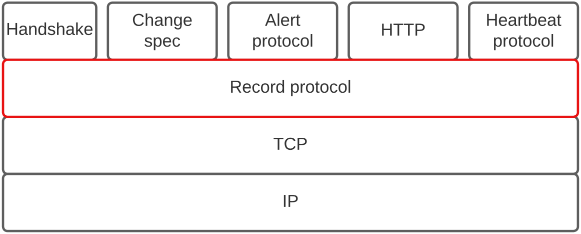
Heartbeat protocol

Record protocol

TCP

IP

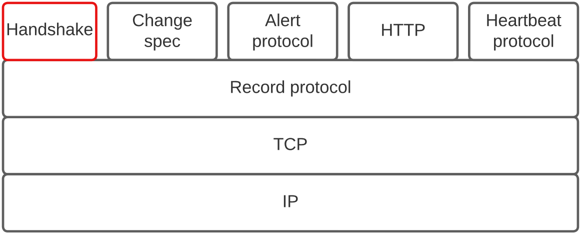
SSL/TLS Protocol Stack



Record Protocol

- Message Integrity and Confidentiality
- Uses key agreed on handshake

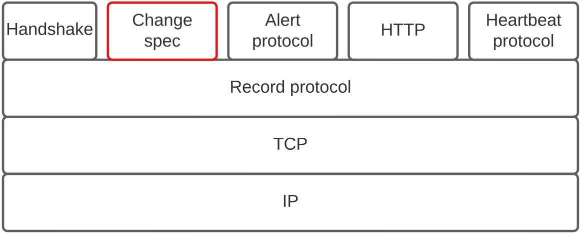
SSL/TLS Protocol Stack



Handshake

- Most complex protocol
- Crucial to establish a cryptographic key

SSL/TLS Protocol Stack



Handshake

Change spec

Alert protocol

HTTP

Heartbeat protocol

Record protocol

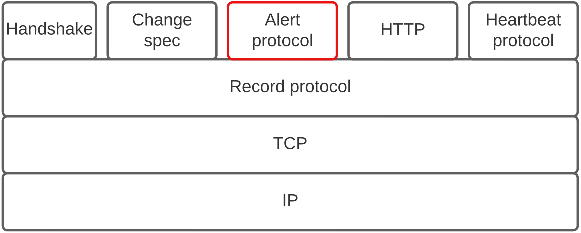
TCP

IP

Change Cipher Spec

- Single message
- Establishes agreed cipher specifications

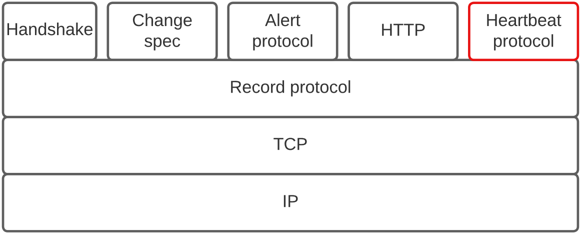
SSL/TLS Protocol Stack



Alert protocol

- TLS alerts
- Can provoke warning, or terminate connections

SSL/TLS Protocol Stack



Heartbeat protocol

- Pings regularly
- Prevents connection from shutting down

TLS Architecture

TLS connection

- A transport that provides a suitable type of service
- For TLS, such connections are peer-to-peer
- Connections are transient
- Every connection is associated with *one session*

TLS Architecture

TLS connection

- A transport that provides a suitable type of service
- For TLS, such connections are peer-to-peer
- 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
- Defines a set of crypto security parameters, shared among multiple connections
- Used to avoid expensive negotiation stages, at the start of each connection

TLS Session State

- Session identifier
- Peer certificate
- Compression method
- Cipher spec
- Master secret
- Is resumable

TLS Session State

- Session identifier
  - An arbitrary byte sequence chosen by the server to identify an active or resumable session state
- Peer certificate
- Compression method
- Cipher spec
- Master secret
- Is resumable

TLS Session State

- Session identifier
- Peer certificate
  - An X509.v3 certificate of the peer. Optional element of the state
- Compression method
- Cipher spec
- Master secret
- Is resumable

TLS Session State

- Session identifier
- Peer certificate
- Compression method
  - The algorithm used to compress data prior to encryption
- Cipher spec
- Master secret
- Is resumable

TLS Session State

- Session identifier
- Peer certificate
- Compression method
- Cipher spec
  - Specified the bulk data encryption algorithm and a hash algorithm used for MAC computation; also defines cryptographic attributes, e.g. hash_size
- Master secret
- Is resumable

TLS Session State

- Session identifier
- Peer certificate
- Compression method
- Cipher spec
- Master secret
  - A symmetric secret key shared between client and server
- Is resumable

TLS Session State

- Session identifier
- Peer certificate
- Compression method
- Cipher spec
- Master secret
- Is resumable
  - A flag indicating whether the session can be used to initiate new connections

TLS Connection State

- Server and client randomness
- Server write MAC key
- Client write MAC key
- Server write key
- Client write key
- Initialization vectors
- Sequence numbers

TLS Connection State

- Server and client randomness
  - Byte sequences chosen by the server and client for each connection
- Server write MAC key
- Client write MAC key
- Server write key
- Client write key
- Initialization vectors
- Sequence numbers

TLS Connection State

- Server and client randomness
- Server write MAC key
  - Cryptographic key used to authenticate messages sent by the server
- Client write MAC key
  - Cryptographic key used to authenticate messages sent by the client
- Server write key
- Client write key
- Initialization vectors
- Sequence numbers

TLS Connection State

- Server and client randomness
- Server write MAC key
- Client write MAC key
- Server write key
  - Cryptographic key used to encrypt messages sent by the server
- Client write key
  - Cryptographic key used to encrypt messages sent by the client
- Initialization vectors
- Sequence numbers

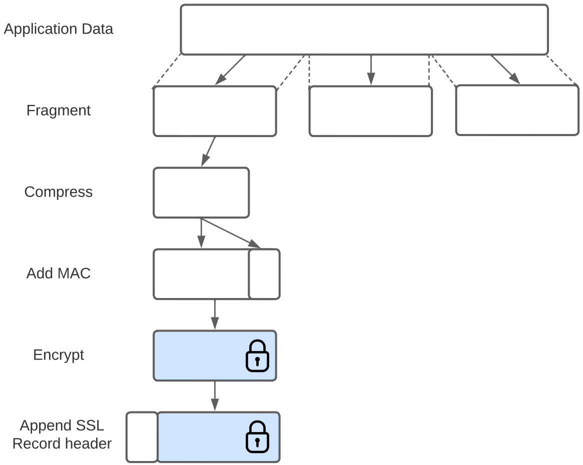
TLS Connection State

- Server and client randomness
- Server write MAC key
- Client write MAC key
- Server write key
- Client write key
- Initialization vectors
  - Values used in encryption to ensure freshness of ciphertexts, so that two encryptions of the same message do not produce the same ciphertext
  - Initialized by the handshake protocol
  - Final ciphertext of each record used as IV for the next one – chaining blocks
- Sequence numbers

TLS Connection State

- Server and client randomness
- Server write MAC key
- Client write MAC key
- Server write key
- Client write key
- Initialization vectors
- Sequence numbers
  - Each party maintains sequence numbers for messages sent/received
  - Initialized at the cipher spec message
  - May not exceed $2^{64} - 1$

Record Protocol Operation



- Resulting unit transmitted via TCP
- Receiver decrypts, verifies, decompresses and reassembles

Handshake Protocol

- Most complex part of TLS
- Used before any application data is transmitted

Handshake Protocol

- Most complex part of TLS
- Used before any application data is transmitted
- Allows the server and client to:
  - Mutually authenticate

Handshake Protocol

- Most complex part of TLS
- Used before any application data is transmitted
- Allows the server and client to:
  - Mutually authenticate
  - Negotiate encryption and MAC algorithms

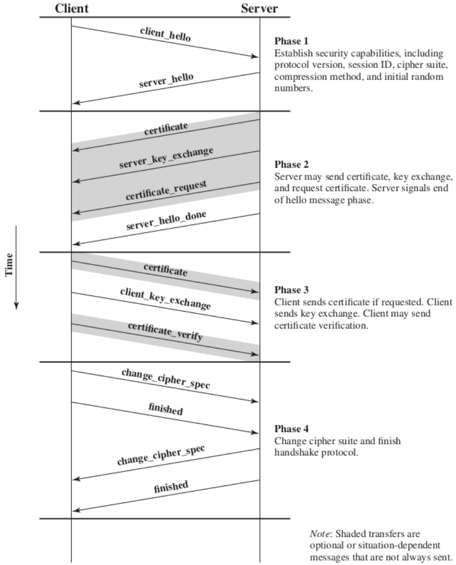
Handshake Protocol

- Most complex part of TLS
- Used before any application data is transmitted
- Allows the server and client to:
  - Mutually authenticate
  - Negotiate encryption and MAC algorithms
  - Negotiate cryptographic keys

Handshake Protocol

- Most complex part of TLS
- Used before any application data is transmitted
- Allows the server and client to:
    - Mutually authenticate
    - Negotiate encryption and MAC algorithms
    - Negotiate cryptographic keys
- Comprises a series of messages exchanged by client and server
- Exchange made on four stages

Handshake Protocol - 4 stages



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.

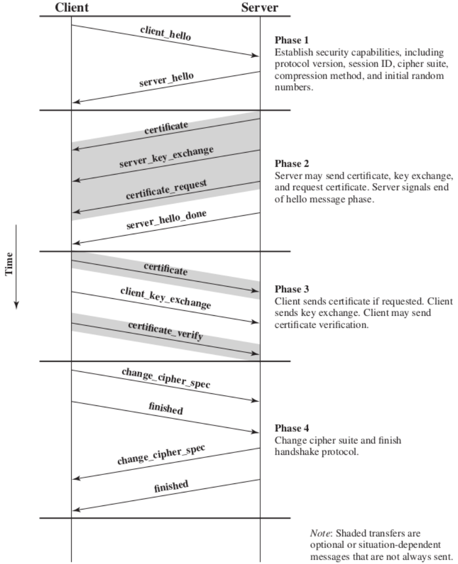
Note: Shaded transfers are optional or situation-dependent messages that are not always sent.

Figure 22.6 **Handshake Protocol Action**

Stage 1

- Hello!
- Here are the specs I can use
  - TLS version
  - Session ID
  - CipherSuite
  - Compression method

Handshake Protocol - 4 stages



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.

Figure 22.6 **Handshake Protocol Action**

Stage 2 and 3

- Certificate exchange
- Certificate verification
- Key agreement
  - RSA/Diffie-Hellman

Stage 4

- Client sends cipher specs
- Client sends a finished protected with authenticated encryption using new algorithms, keys and secrets
- Server verifies and does the same

Change Cipher Spec Protocol

- The simplest of the four
- A single message of a single byte. Value is either 0 or 1

Change Cipher Spec Protocol

- The simplest of the four
- A single message of a single byte. Value is either 0 or 1
- Sole purpose of this message is to cause pending state to be copied into the current state – used as confirmation message
- Hence updating the cipher suite in usage

Alert Protocol

- Conveys TLS-related alerts to peer entity
  - Alert messages are compressed and encrypted
  - Example of fatal alert: incorrect MAC
  - Example of non-fatal alert: close_notify (notifies the recipient that the sender will not send any more messages in this communication)

Alert Protocol

- Conveys TLS-related alerts to peer entity
  - Alert messages are compressed and encrypted
  - Example of fatal alert: incorrect MAC
  - Example of non-fatal alert: close_notify (notifies the recipient that the sender will not send any more messages in this communication)
- Each message consists of two bytes
- First byte refers to the severity; the second specifies

Alert Protocol

- Conveys TLS-related alerts to peer entity
  - Alert messages are compressed and encrypted
  - Example of fatal alert: incorrect MAC
  - Example of non-fatal alert: close_notify (notifies the recipient that the sender will not send any more messages in this communication)
- Each message consists of two bytes
- First byte refers to the severity; the second specifies
  - Fatal messages terminate the connection immediately
  - Other connections for that session may continue, but no additional connections are established

Heartbeat Protocol - P1

- A periodic signal is generated by hardware or software to indicate normal operation, or to synchronize with other parts of a system

Heartbeat Protocol - P1

- A periodic signal is generated by hardware or software to indicate normal operation, or to synchronize with other parts of a system
- Typically used to monitor the availability of a protocol entity – the name should be self-explanatory!

Heartbeat Protocol - P1

- A periodic signal is generated by hardware or software to indicate normal operation, or to synchronize with other parts of a system
- Typically used to monitor the availability of a protocol entity – the name should be self-explanatory!
- The heartbeat protocol runs on top of the TLS record protocol
- Relies on two message types

Heartbeat Protocol - P1

- A periodic signal is generated by hardware or software to indicate normal operation, or to synchronize with other parts of a system
- Typically used to monitor the availability of a protocol entity – the name should be self-explanatory!
- The heartbeat protocol runs on top of the TLS record protocol
- Relies on two message types
  - HEARTBEAT_REQUEST - prove you are alive
  - HEARTBEAT_RESPONSE - i am, indeed, alive

Heartbeat Protocol - P2

- Request includes payload length; payload; padding fields

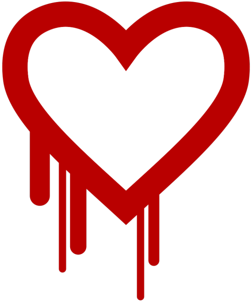
Heartbeat Protocol - P2

- Request includes payload length; payload; padding fields
- Response must include **an exact copy** of the received payload
    - Prevent replay attacks!

Heartbeat Protocol - P2

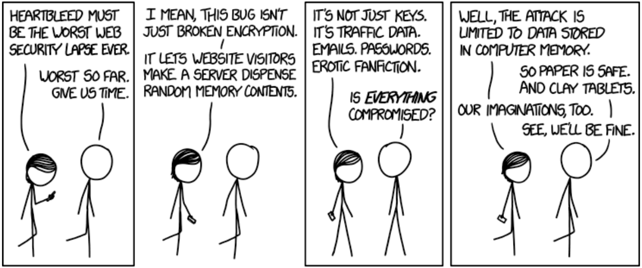
- Request includes payload length; payload; padding fields
- Response must include **an exact copy** of the received payload
    - Prevent replay attacks!
- Serves two main purposes
    - Assures the sender that the recipient is still alive, even if there is no regular activity in the underlying TCP connection
    - Generates activity across the connection during idle periods, which avoids closure by a firewall automatic mechanisms to disable idle connections

Heartbleed



- OpenSSL contains an open-source implementation of SSL/TLS
- A fatal flaw in OpenSSL, breaching privacy of log-in data
- Estimated victims: **two-thirds** of Web servers

Heartbleed - How it works


HEARTBLEED MUST BE THE WORST WEB SECURITY LAPSE EVER.
WORST SO FAR. GIVE US TIME.
I MEAN, THIS BUG ISN'T JUST BROKEN ENCRYPTION.
IT LETS WEBSITE VISITORS MAKE A SERVER DISPENSE RANDOM MEMORY CONTENTS.
IT'S NOT JUST KEYS. IT'S TRAFFIC DATA. EMAILS. PASSWORDS. EROTIC FANFICTION.
IS EVERYTHING COMPROMISED?
WELL, THE ATTACK IS LIMITED TO DATA STORED IN COMPUTER MEMORY.
SO PAPER IS SAFE. AND CLAY TABLETS.
OUR IMAGINATIONS, TOO.
SEE, WE'LL BE FINE.

Heartbeat

- Send heartbeat message
- Extract; prep payload; send reply
- Response contains exactly the expected payload size
- Check for payload validity

Heartbleed

- Small payload disguised as big one
- Extract; prep (bad) payload; send reply
- Response contains **much** more than expected
- Gets TLS keys, cookies, passwords!

SSL/TLS Applications

- Secure e-commerce using SSL/TLS
- Client authentication not needed until client decides to buy something, which triggers the authentication stage

SSL/TLS Applications

- Secure e-commerce using SSL/TLS
- Client authentication not needed until client decides to buy something, which triggers the authentication stage
- SSL provides secure channel for sending credit card information

SSL/TLS Applications

- Secure e-commerce using SSL/TLS
- Client authentication not needed until client decides to buy something, which triggers the authentication stage
- SSL provides secure channel for sending credit card information
- Client authenticated using credit card information, merchant bears (most of) the risk
- Widely deployed

SSL/TLS Applications

- Secure e-commerce using SSL/TLS
- Client authentication not needed until client decides to buy something, which triggers the authentication stage
- SSL provides secure channel for sending credit card information
- Client authenticated using credit card information, merchant bears (most of) the risk
- Widely deployed
- Secure remote login / **Secure Shell**
- Authenticated, encrypted path to the OS over the network

HTTPS

HTTP over SSL

- Combination of HTTP and SSL to implement secure communication between a Web browser and a Web server

HTTPS

HTTP over SSL

- Combination of HTTP and SSL to implement secure communication between a Web browser and a Web server
- Build into all modern Web browsers
  - URL addresses begin with `HTTPS://`

HTTPS

HTTP over SSL

- Combination of HTTP and SSL to implement secure communication between a Web browser and a Web server
- Build into all modern Web browsers
  - URL addresses begin with `HTTPS://`
- Agent acting as HTTP client also acts as the TLS client

HTTPS

HTTP over SSL

- Combination of HTTP and SSL to implement secure communication between a Web browser and a Web server
- Build into all modern Web browsers
  - URL addresses begin with `HTTPS://`
- Agent acting as HTTP client also acts as the TLS client
- When HTTPS is used, the following elements are protected:
  - URL of requested document

HTTPS

HTTP over SSL

- Combination of HTTP and SSL to implement secure communication between a Web browser and a Web server
- Build into all modern Web browsers
  - URL addresses begin with HTTPS://
- Agent acting as HTTP client also acts as the TLS client
- When HTTPS is used, the following elements are protected:
  - URL of requested document
  - Document contents
  - Contents of browser forms

HTTPS

HTTP over SSL

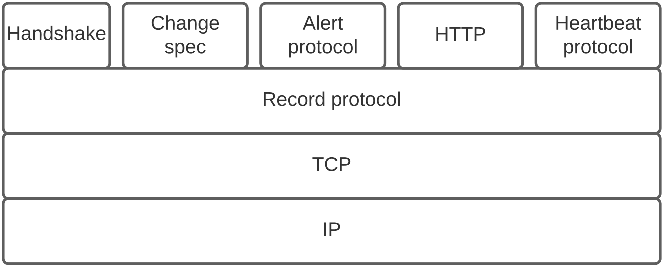
- Combination of HTTP and SSL to implement secure communication between a Web browser and a Web server
- Build into all modern Web browsers
  - URL addresses begin with `HTTPS://`
- Agent acting as HTTP client also acts as the TLS client
- When HTTPS is used, the following elements are protected:
  - URL of requested document
  - Document contents
  - Contents of browser forms
  - Cookies sent from browser to server and vice-versa

HTTPS

HTTP over SSL

- Combination of HTTP and SSL to implement secure communication between a Web browser and a Web server
- Build into all modern Web browsers
  - URL addresses begin with `HTTPS://`
- Agent acting as HTTP client also acts as the TLS client
- When HTTPS is used, the following elements are protected:
  - URL of requested document
  - Document contents
  - Contents of browser forms
  - Cookies sent from browser to server and vice-versa
  - Contents of HTTP header

First handshake, then HTTP through TLS



Handshake

Change spec

Alert protocol

HTTP

Heartbeat protocol

Record protocol

TCP

IP

Layered connection

- Connection begins with a TLS CLIENTHELLO, which triggers the TLS handshake
- When it finishes, the client sends the first HTTP request
- All data sent as TLS application data

Closing Connections

- An HTTP client or server can indicate the closing of a connection by including the line CONNECTION: CLOSE in an HTTP record

Closing Connections

- 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 TLS to close the connection with the peer TLS entity on the remote side – closing the underlying TCP connection

Closing Connections

- 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 TLS to close the connection with the peer TLS entity on the remote side – closing the underlying TCP connection
- TLS implementation must initiate an exchange of closure alerts before closing a connection

Closing Connections

- 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 TLS to close the connection with the peer TLS entity on the remote side – closing the underlying TCP connection
- TLS implementation must initiate an exchange of closure alerts before closing a connection
- TLS session ensures that cryptographic parameters are kept (avoiding expensive negotiations)

Secure Shell Protocol

- Originally developed for UNIX, now available on most OSs

Secure Shell Protocol

- Originally developed for UNIX, now available on most OSs
- Provides an authenticated, encrypted path to the OS command line over the network

Secure Shell Protocol

- Originally developed for UNIX, now available on most OSs
- Provides an authenticated, encrypted path to the OS command line over the network
- Replacement for insecure utilities such as Telnet, rlogin, rsh

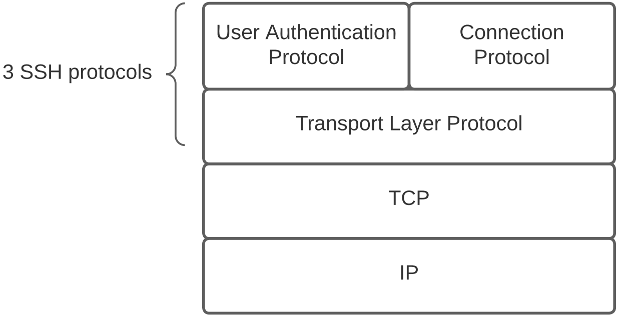
Secure Shell Protocol

- Originally developed for UNIX, now available on most OSs
- Provides an authenticated, encrypted path to the OS command line over the network
- Replacement for insecure utilities such as Telnet, rlogin, rsh
- Protects against spoofing attacks and modification of data

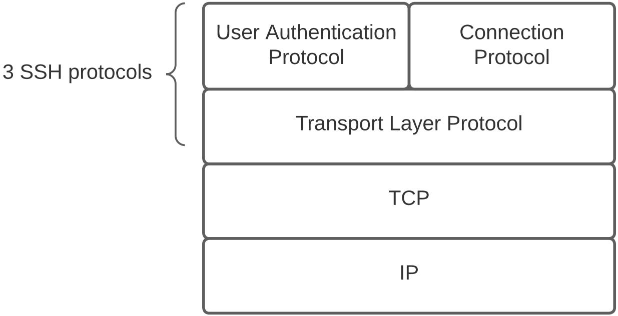
Secure Shell Protocol

- Originally developed for UNIX, now available on most OSs
- Provides an authenticated, encrypted path to the OS command line over the network
- Replacement for insecure utilities such as Telnet, rlogin, rsh
- Protects against spoofing attacks and modification of data
- The *de facto* method to access remote resources

SSH Protocol(s)

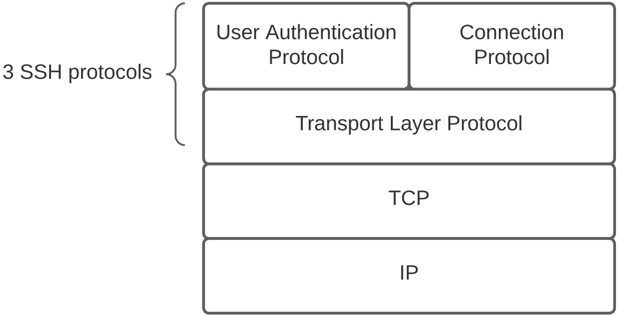


SSH Protocol(s)



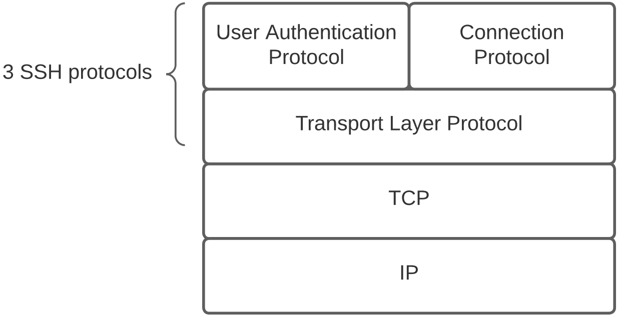
- **Transport Layer Portocol** provides server authentication, confidentiality, and integrity.

SSH Protocol(s)



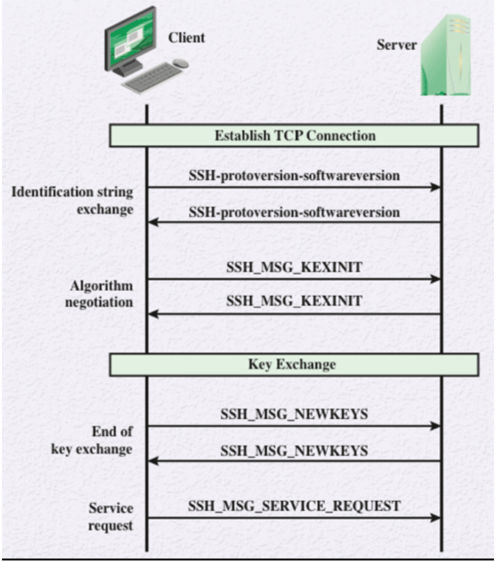
- **Transport Layer Portocol** provides server authentication, confidentiality, and integrity.
- **User Authentication Protocol** authenticates the client-side user to the server

SSH Protocol(s)



- **Transport Layer Portocol** provides server authentication, confidentiality, and integrity.
- **User Authentication Protocol** authenticates the client-side user to the server
- **Connection Protocol** multiplexes the encrypted tunnel into several logical channels

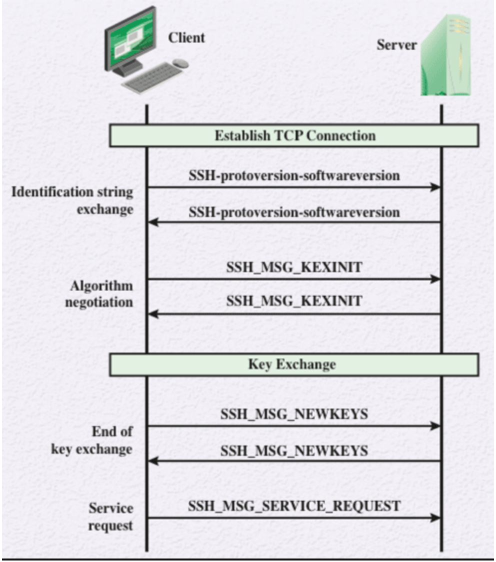
SSH Transport Layer Protocol



Multiple stages

1. Protocol and SW versions agreement
2. Supported algorithms exchanged
3. Key exchange finishes
4. Service ready to execute

SSH Transport Layer Protocol



Algorithm Agreement

- One (or more) algorithms must be listed
- Encryption algorithm used for confidentiality
- MAC algorithm used for data authentication
- Compression algorithm optional

SSH Authentication Methods

Public Key

- The client sends a message to the server that has the client's public key. Signed with the private key
- Upon receiving the message, the server check if the key is acceptable for authentication, and if the signature is correct

Password

Hostbased

SSH Authentication Methods

Public Key

Password

- The client sends a message containing a plaintext password, encrypted via the Transport Layer Protocol

Hostbased

SSH Authentication Methods

Public Key

Password

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 its host
- Instead of verifying the client identity, the host identity is checked
- Provides group anonymity

SSH Connection Protocol

- SSH Connection Protocol runs on top of the Transport Layer Protocol
    - The secure authenticated connection, referred to as *tunnel*, is used by the Connection Protocol to multiplex a number of logical channels

SSH Connection Protocol

- SSH Connection Protocol runs on top of the Transport Layer Protocol
  - The secure authenticated connection, referred to as *tunnel*, is used by the Connection Protocol to multiplex a number of logical channels
- Channel mechanism
  - All types of communications using SSH supported via separate channels
  - Either side can open a channel
  - Channel type identifies the application/purpose of the channel

Channel Types

- **Session**
  - The remote execution of a program
  - Program may be a shell, an application such as file transfer, a system command, or a built-in subsystem

Channel Types

- **Session**
  - The remote execution of a program
  - Program may be a shell, an application such as file transfer, a system command, or a built-in subsystem
- **X11**
  - Refers to the X Window System, a computer software system and network protocol that provide a GUI for networked computers

Channel Types

- **Session**
  - The remote execution of a program
  - Program may be a shell, an application such as file transfer, a system command, or a built-in subsystem
- **X11**
  - Refers to the X Window System, a computer software system and network protocol that provide a GUI for networked computers
- **Forwarded-tcpip**
  - Remote port forwarding (from a remote computer to the local computer)
- **Direct-tcpip**
  - Local port forwarding (insecure TCP connection → SSH tunnel)

Port Forwarding

- Provides the ability to convert any insecure TCP connection into a secure SSH connection – a.k.a. SSH tunneling

Port Forwarding

- Provides the ability to convert any insecure TCP connection into a secure SSH connection – a.k.a. SSH tunneling
- Incoming TCP traffic is delivered to the appropriate application on the basis of the port number (an identifier of a user in TCP)

Port Forwarding

- Provides the ability to convert any insecure TCP connection into a secure SSH connection – a.k.a. SSH tunneling
- Incoming TCP traffic is delivered to the appropriate application on the basis of the port number (an identifier of a user in TCP)
- An application may employ multiple port numbers
  - HTTP servers usually listen on port 80 (443 for HTTPS)

Wrap up

SSL and TLS

- Architecture over classical network layers

Wrap up

SSL and TLS

- Architecture over classical network layers
- TLS sessions are ephemeral, connections allow for multiple sessions

Wrap up

SSL and TLS

- Architecture over classical network layers
- TLS sessions are ephemeral, connections allow for multiple sessions
- Protocols rely on TLS for secure communication

Wrap up

SSL and TLS

- Architecture over classical network layers
- TLS sessions are ephemeral, connections allow for multiple sessions
- Protocols rely on TLS for secure communication
- HTTPS uses TLS over HTTP

Wrap up

SSL and TLS

- Architecture over classical network layers
- TLS sessions are ephemeral, connections allow for multiple sessions
- Protocols rely on TLS for secure communication
- HTTPS uses TLS over HTTP
- Attacks at the TLS layer - Heartbleed

Wrap up

SSL and TLS

- Architecture over classical network layers
- TLS sessions are ephemeral, connections allow for multiple sessions
- Protocols rely on TLS for secure communication
- HTTPS uses TLS over HTTP
- Attacks at the TLS layer - Heartbleed

SSH

- Relies on an handshake similar to TLS...

Wrap up

SSL and TLS

- Architecture over classical network layers
- TLS sessions are ephemeral, connections allow for multiple sessions
- Protocols rely on TLS for secure communication
- HTTPS uses TLS over HTTP
- Attacks at the TLS layer - Heartbleed

SSH

- Relies on an handshake similar to TLS...
- ... but authentication is not certificate-based

Wrap up

SSL and TLS

- Architecture over classical network layers
- TLS sessions are ephemeral, connections allow for multiple sessions
- Protocols rely on TLS for secure communication
- HTTPS uses TLS over HTTP
- Attacks at the TLS layer - Heartbleed

SSH

- Relies on an handshake similar to TLS...
- ... but authentication is not certificate-based
- Allows for different channels with different purposes

Network Security - Week 3

João Soares

DCC/FCUP

2023