Web Security

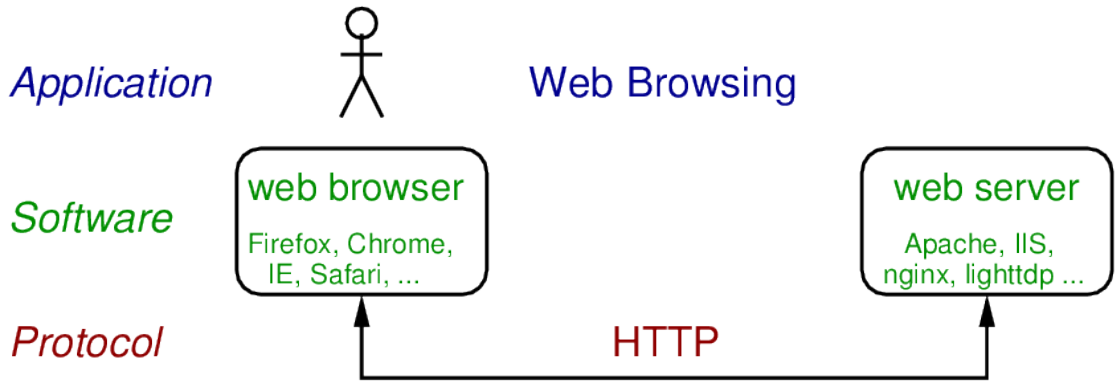
ITS335: IT Security

Sirindhorn International Institute of Technology
Thammasat University

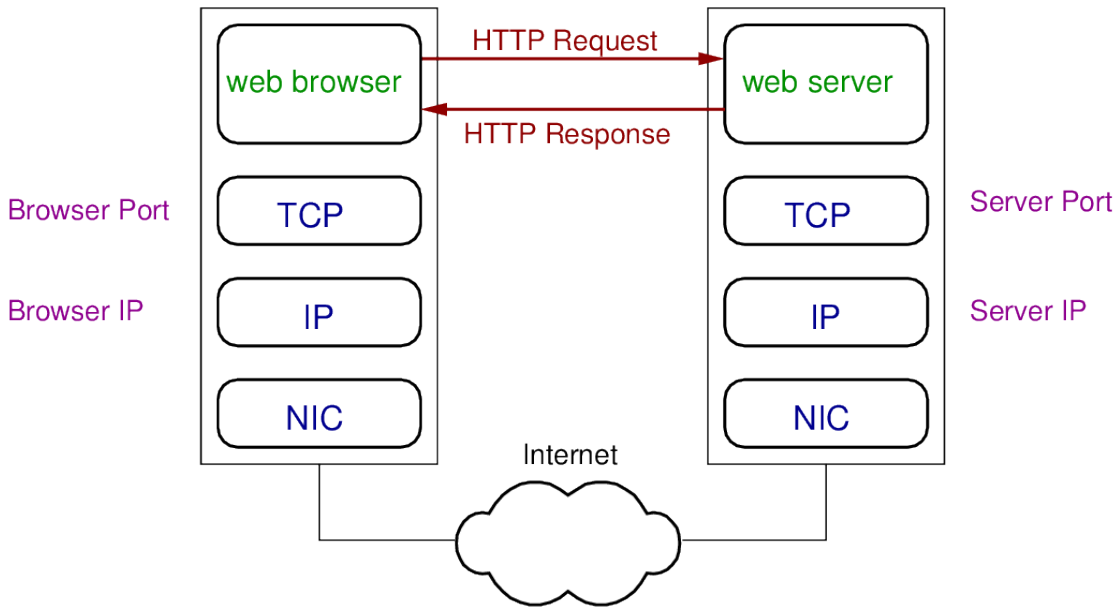
Prepared by Steven Gordon on 20 December 2015
its335y15s2l08, Steve/Courses/2015/s2/its335/lectures/websecurity.tex, r4288

Contents

Web Browsing with HTTP



Web Browsing with HTTP



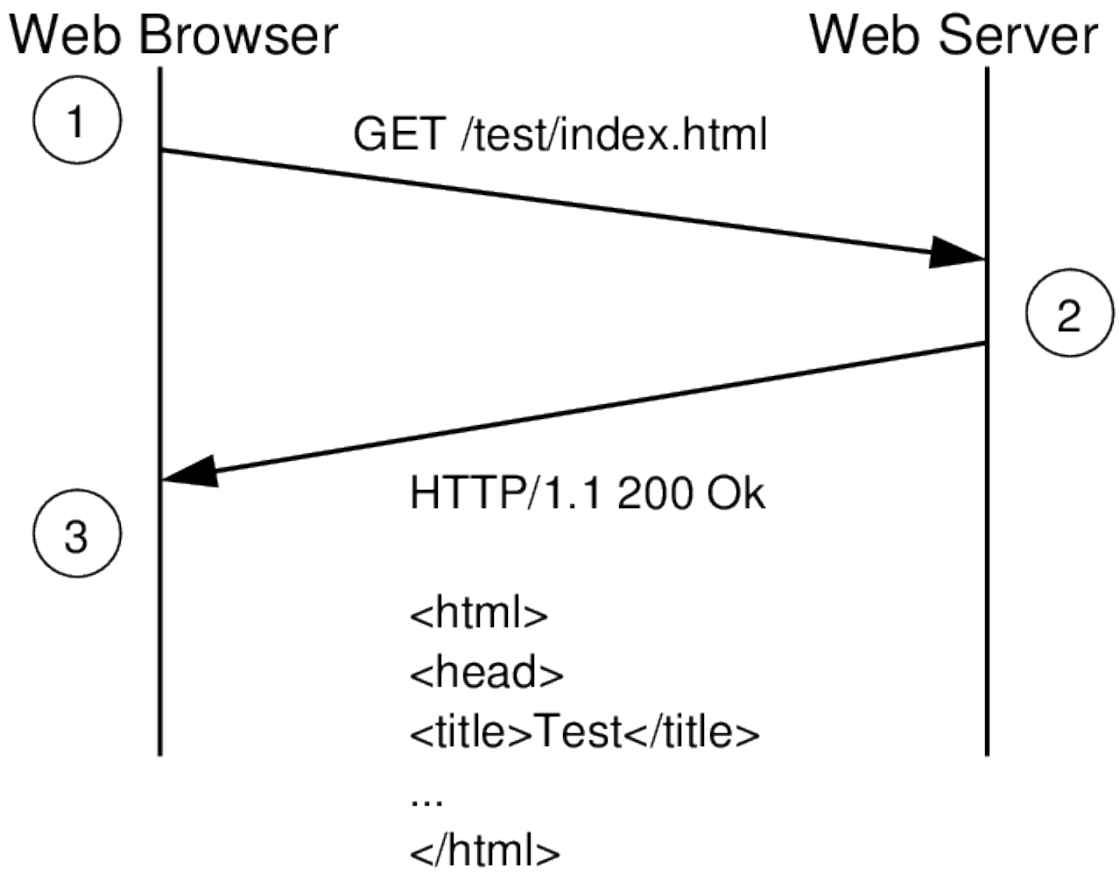
Web Access with Hypertext Transfer Protocol

- HTTP is a request/response protocol for web browsing
- HTTP is stateless; no dependence between a request and previous request
- User Agent (client) sends HTTP Request message
- Server responds with HTTP Response message
- Default server port number: 80
- Generic HTTP message format:

```
Start line
Optional header lines
<empty line>
Optional message body
```

- Start line differs for request and response
- Header format: `field-name: value`

HTTP Example



HTTP Request Messages

- Start line: `Method URL Version`
- Methods:
  - GET: retrieve the resource at the specific URL
  - HEAD: same as GET, except do not return message body (only header)
  - OPTIONS: retrieve options available for resource or server
  - POST: asks server to accept and process the attached data at the resource
  - ...
- Version: version of HTTP, e.g. HTTP/1.0, HTTP/1.1

HTTP Response Messages

- Start line: `Version StatusCode StatusReason`
- Status Codes and Reasons:
  - 100: Continue (the client should continue with its request)
  - 200: OK (the request succeeded)
  - 301: Moved Permanently (the requested resource has a new URL)
  - 304: Not Modified (resource hasnt changed since last request, client should use cached copy)
  - 401: Unauthorized (request must include user authentication)
  - 403: Forbidden (request was understood, but server refuses to process it)
  - 404: Not Found (server cannot find resource at requested URL)
  - 503: Service Unavailable (server currently unable to handle request, e.g. server is too busy)

HTTP Headers

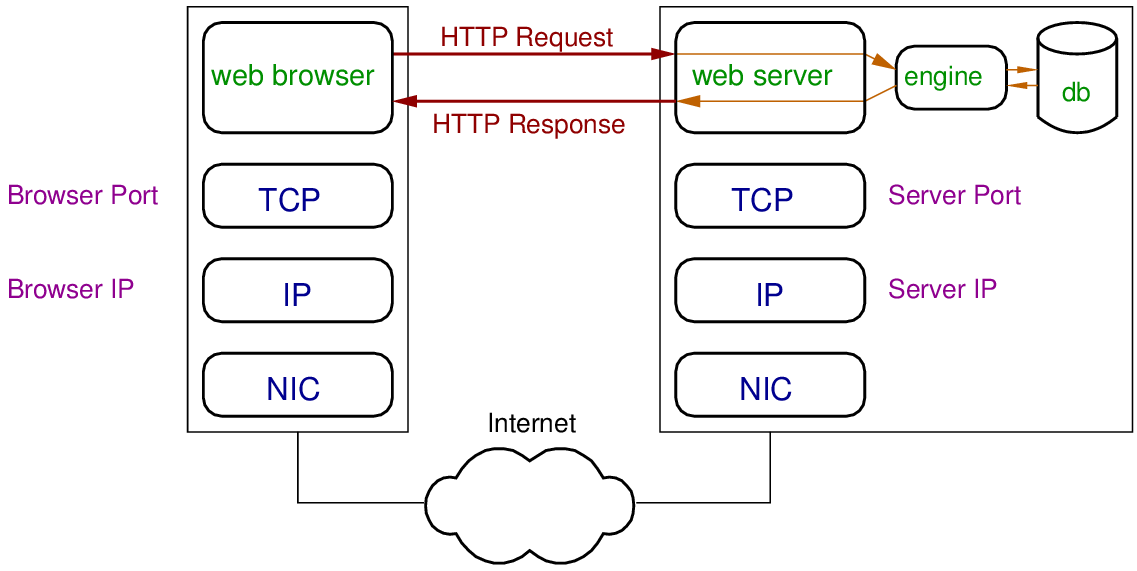
- Date: data and time of message generation
- Host: domain name of host of resource (means relative URLs can be used)
- Accept-Charset, Accept-Encoding, Accept-Language: indicate the character sets, encodings and languages that client can accept
- Authorization: include user credentials (e.g. username, password) if authorization is required
- User-Agent: indicates information about the client (user agent), e.g. web browser
- Referrer: URL from which this request came from
- Content-Encoding: encoding or compression, e.g. gzip
- Content-Length: length of message body on bytes
- Content-Type: the type of content in message body
- Last-Modified: indicates data/time when content was last modified on server

Contents

Web Applications

- Plain, static web pages: HTML, images and other files served to browser
- But many applications use dynamic content
  - Content server to browse changes depending on request
  - Provides interactive, tailored content
  - Client-side: JavaScript, Flash, Silverlight, Java
  - Server-side: CGI, ASP, PHP, Coldfusion, Java, ...
  - Content stored in databases

Dynamic Content with Server-Side Processing



What are the security issues?

- Data transmitted between browser and server is confidential: encryption with HTTPS
- Browser sure it is communicating with intended server: digital certificates
- Server sure it is communicating with intended user: password authentication, session management
- Actions performed by server (engine) are appropriate: authentication, access control
- Actions of user (of browser) are kept private: anonymity services

Contents

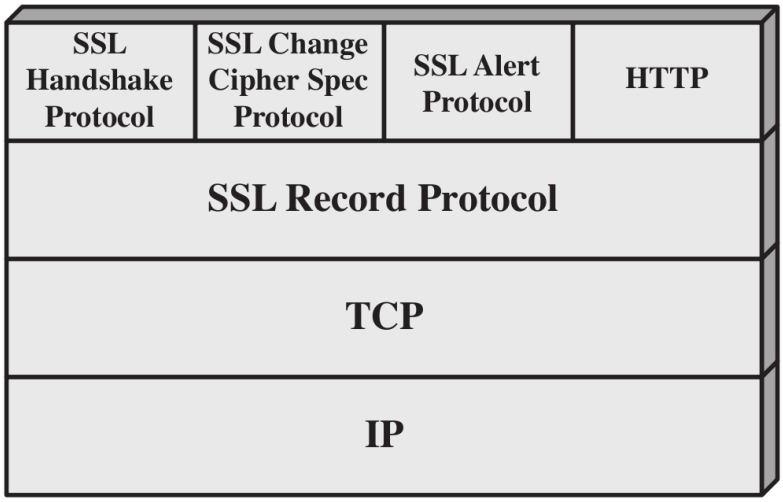
HTTPS

- HTTPS: HTTP over SSL (or TLS)
- URL uses https://
- Web server listens on port 443
- Encrypt: URL of requested document, contents of document, contents of browser forms, cookies, contents of HTTP header
- Server is authenticated using certificate (using SSL)
- Client is authenticated using password (using HTTP)

SSL and TLS

- Secure Sockets Layer (SSL) originated in Netscape web browser
- Transport Layer Security (TLS) standardised by IETF
- SSLv3 and TLS are almost the same
- SSL provides security services to application layer protocols using TCP
- SSL architecture consists of multiple protocols

SSL Architecture



Record: provides confidentiality and message integrity

Handshake: authenticate entities, negotiate parameter values

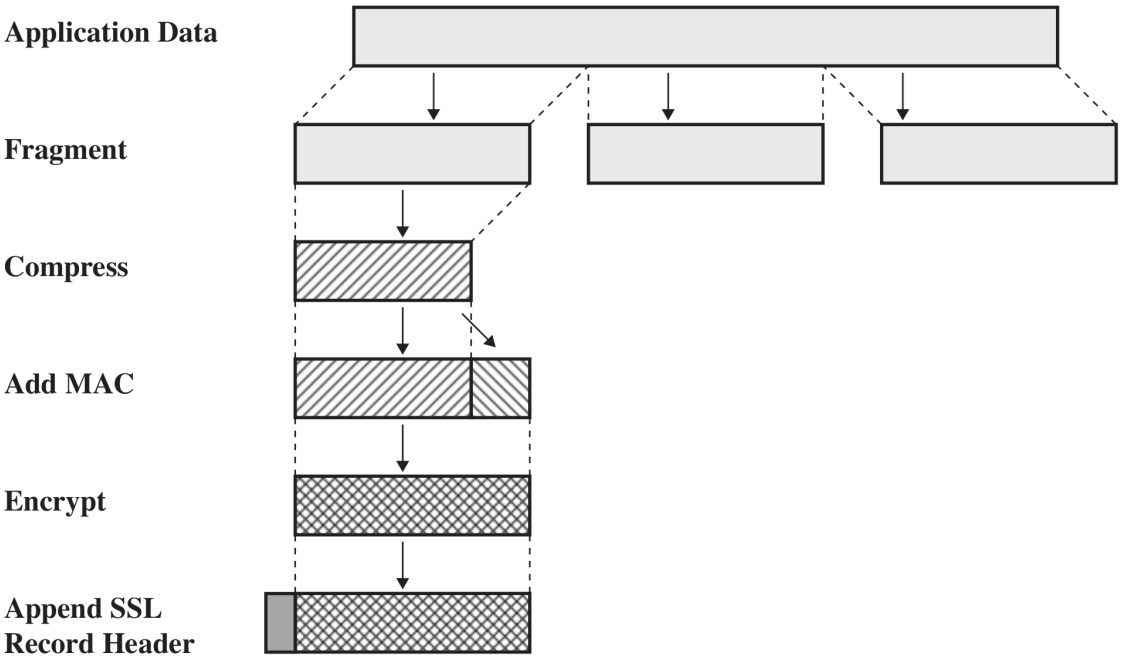
Change Cipher: change cipher for use in connection

Alert: alert peer entity of status/warning/error

Connections and Sessions

- SSL connection corresponds with TCP connection
  - Client and server may have multiple connections
- SSL session is association between client and server
  - Session created with Handshake protocol
  - Multiple connections can be associated with one session
  - Security parameters for session can be shared for connections
- State information is stored after Handshake protocol
  - Session: ID, certificate, compression, cipher spec, master secret, ...
  - Connection: random values, encrypt keys, MAC secrets, IV, sequence numbers, ...

SSL Record Protocol Operation



SSL Handshake Protocol

- Allow client and server to authenticate each other
- Negotiate encryption and MAC algorithms, exchange keys
  - Key Exchange: RSA, Diffie-Hellman
  - MAC: HMAC using SHA or MD5
  - Encryption: RC4, RC2, DES, 3DES, IDEA, AES
- Multiple phases:
  1. Establish security capabilities: client proposes algorithms, server selects one
  2. Server authentication and key exchange
  3. Client authentication and key exchange
  4. Finish setting up connection

Contents

Authentication and Encryption in Web Browsing

- Browser and server do not have pre-shared secrets
- Use public key cryptography to securely exchange secret key
  - RSA/DSA
  - Diffie-Hellman key exchange
  - Elliptic curve cryptography
- Once a secret key is exchanged, use symmetric key encryption
  - AES, RC4, 3DES, ...
- E.g. with RSA: if a server sends browser its RSA public key, how does browser know it is indeed RSA public key of server?
  - Get a trusted third party to confirm it is the servers RSA public key

Public Key for Key Exchange, Symmetric for Data

Man-in-the-Middle Attack

Digital Certificates

Step 1: Server Obtains Digital Certificate

- Server (owner) creates key pair: (PU_s, PR_s)
- Server confirms identity, ID_s, with trusted third party called Certificate Authority
- CA issues server with a digital certificate by signing relevant info:

$$C_s = ID_s||PU_s||T, \mathrm{E}(PR_{CA}, \mathrm{H}(ID_s||PU_s||T))$$

- A timestamp, T, can be used to determine how long the certificate is valid
- X.509 specifies standard format of certificates

Digital Certificates

Step 2: Server Sends Digitial Certificate to Browser

- When browser initiates communications with server, server responds with C_s
- Browser verifies signature using PU_{CA}
  - Assumes browser already knows and trusts PU_{CA}
  - PU_{CA} is stored in self-signed certificate:

$$C_{CA} = ID_{CA}||PU_{CA}||T, \mathrm{E}(PR_{CA}, \mathrm{H}(ID_{CA}||PU_{CA}||T))$$

- Once verified, browser can choose secret value and send it encrypted using PU_s to server

Key Exchange with Certificates

Attacks on Certificates

X.509 Certificates

- X.509 certificate format includes:
  - Version of X.509 certificate
  - Serial number unique to the issuer (CA)
  - Signature algorithm
  - Issuer's name and unique identifier
  - Period of validity (start time, end time)
  - Subject's name and unique identifier
  - Subject's public key information: algorithm, parameters, key
  - Signature
- Certificates may be revoked before expiry
  - CA signs a Certificate Revocation List (CRL), which is publicly available

Digital Certificates in Practice

How does a server obtain a certificate?

- Prove identity to CA by:
  - Domain validation
  - Extended validation
- Free and commercial services

How does browser obtain CA certificate?

- Pre-loaded into browsers
- Hierarchy of certificates is supported

What if CA certificate is not in browser?

- Browsers commonly present warning to user

Security Issues with Digital Certificates

- Identity verification of server (owners)
- Security of CA private key
- Pre-loaded certificates by browser publisher
- Response when invalid certificate received
- Algorithms used in certificates should be strong

Contents

Key Points

- Web browsing uses HTTP over TCP
- Secure web browsing inserts SSL in between HTTP and TCP; HTTPS
- Secret key exchange between browser and server using public key crypto
- For browser to trust server public key, must be signed by trusted third party (certificate authority)
- X.509 digital certificates used in web browsing, email and many networked applications

Security Issues

- Digital certificates rely on trustworthiness of certificate authorities
- Also rely on action by users: response with invalid certificate received; trusting browser CA list
- Man-in-the-middle interception/modification attacks on web browsing are easy *if* certificates are compromised

Areas To Explore

- Public key distribution methods
- PGP and GPG for email
- Securing web applications, OWASP