PV222
Security Architectures

Lecture 2

Web Security

Lecture Overview

- What is the web?
- The web components: HTTP and HTML
- HTTP, state and cookies
- Web server hazards
- SSL/TLS
- Further information

The World Wide Web

- The World Wide Web (or just the *web*) is essentially a means of providing access to data across the Internet in a way that hides most of the complexity.
- In technical terms it does not do much more than the simple file transfer protocol FTP.
- However, the combination of transparency and hyperlinks enable the construction of the enormously complex web we have today.

Web browsers and servers

- Two key elements of the web are the browsers and servers.
- The web browser is a programme running on a PC that provides a means of viewing information provided by web servers connected to the Internet.

What is Web Security?

- Garfinkel and Spafford (in *Web Security, Privacy & Commerce*) define web security as:
  1. *"Securing the web server and the data that is on it."*
  2. *"Securing information that travels between the web server and the user."*
  3. *"Securing the end user's computer and other devices that people use to access the Internet."*

The protocol

- The web protocol, i.e. the set of rules by which data is transferred between web browsers and web servers is called **HTTP**, for *HyperText Transfer Protocol*.
- This is a very simple “request/reply” protocol running over **TCP** (the *Transmission Control Protocol*).
- Requests are directed from a web browser to a resource at a specific *address*.

Addresses (URIs and URLs)

- **URI**s (*Universal Resource Identifiers*) are means of identifying network resources.
- A URI is either a **URL** (*Uniform Resource Locator*) or a Name (*URN*).
- URL syntax is defined in RFCs 1738 and 1808.
- A URL looks like:

  `http://<host><path>`

  where `<host>` is an Internet host name or IP address.

The language

- When a web browser receives a request, it responds with information (a “web page”) in a language called **HTML** (*HyperText Markup Language*).
- An HTML file is essentially a text file containing a series of “*markup tags*” instructing the recipient how to display the text.
- A tag may also include a URI for a different web page, and the browser will display this as a *hyperlink*.

Web standards

- Some of the most fundamental web-related specifications are IETF RFCs (requests for comments).
- W3C (World Wide Web Consortium) is a forum that develops and publishes web specifications.

HTTP – overview

- There are two main versions of HTTP: Version 1.0 (HTTP/1.0 defined in RFC 1945) and version 1.1 (HTTP/1.1 defined in RFC 2616).
- HTTP is an application-level protocol.
- The fundamental unit of HTTP communication is a message (a structured sequence of bytes).

HTTP – requests/responses

- HTTP is a request/response protocol – that is, a *user agent* (typically a web browser on a PC) sends a request, and a remote *server* sends a response to that request.
- The request consists of a *request method*, a URI, and a protocol version number, followed by a MIME-like message containing a *request modifiers* (parameters), client information, and (possibly) content of some kind.

HTTP – responses

- A server response consists of:
  - a *status line*, including the protocol version number, and a success/error code, and
  - a MIME-like message, containing server information, content *meta-information* (headers), and content.
- The content will typically be written in HTML.

HTML

- The latest versions of HTML are HTML 4.01 and XHTML 1.0.
- HTML 4.01 is a W3C *Recommendation* from 1999. (HTML 2.0 was published as RFC 1866).
- HTML 5 is currently in the W3C *Working Draft* phase of publication.
- XHTML is a reformulation of HTML in XML 1.0 (the latest version was published by W3C in August 2002).

HTML syntax

- An HTML document is divided into:
  - a head section (between `<HEAD>` and `</HEAD>`) and
  - a body (between `<BODY>` and `</BODY>`).
- The title appears in the head (along with other information about the document), and the content appears in the body.
- The body will typically contain paragraphs, marked up with `<P>` … `</P>`.

HTML and SGML

- **SGML** (*Standard Generalized Markup Language*) was published as international standard ISO 8879 in 1986.
- SGML is a system for defining markup languages.
- Authors *mark up* their documents by representing structural, presentational, and semantic information alongside content.
- HTML is one example of a markup language.

SGML use

- A markup language defined in SGML is called an *SGML application*.
- An SGML application is characterised by:
  - An *SGML declaration* that specifies which characters and delimiters appear.
  - A *document type definition* (DTD) that defines the syntax of markup constructs.
  - A specification that describes the semantics of the markup.
  - Document instances containing data (content) and markup.

XML

- The *Extensible Markup Language* (XML) is a subset of SGML.
- Its goal is to enable generic SGML to be server, received, and processed on the web in the way that is now possible with HTML.
- XML has been designed for ease of implementation and for interoperability with both SGML and HTML.

Using HTML

- Writing in HTML is simple.
- The easiest way is to use a tool which automatically produces the HTML syntax (adds the correct tags).
- However, because HTML is essentially plain text plus tags, direct editing simple HTML pages is very straightforward (particularly if you have a few examples to work from).

HTTP is stateless

- The HTTP protocol does not require the server to maintain any protocol state.
- That is, the server does not keep any information to enable consecutive requests from a single user agent to be linked.
- Hence HTTP does not support “sessions”, e.g. as might be required to support e-commerce.

Cookies

- HTTP Cookies are simple means of enabling browser sessions with a server.
- The idea is that the server sends back state information in its response header, in the form of a *Cookie*.
- The Cookie is then resubmitted with the next request to the same server.
- A Cookie might, for example, specify the current contents of your shopping basket.

Cookie contents

- A cookie header (in a response header) contains:
  - *attribute*, the data payload;
  - *domain scope*, enables sharing of cookies by web hosts with specified domain name;
  - *path scope*, limits the URI path to which the cookie should be sent back;
  - *expiration*, the expiry date of the Cookie;
  - *SSL flag*, if set the Cookie should only be sent back via an HTTPS (HTTP over SSL) connection.

Cookies and privacy

- Whilst Cookies are an invaluable tool for e-commerce and other uses of the web, they also constitute a privacy threat.
- Clearly, a server can use Cookies to track individual user PCs (even if the server cannot automatically discover the owner of a particular PC).
- We look at one way this tracking can pose a threat.

Tracking cookies

- Web-based advertising agencies, e.g. DoubleClick, Focalink, Globaltrack, and ADSmart put advertisements on web sites.
- These web pages contain an `<IMG>` tag, pointing to a URL on the advertising agency's server.
- When a web browser sees this `<IMG>` tag, it contacts the agency server to retrieve the graphic.
- The first time the graphic is downloaded, the user browser will receive an agency cookie containing a random ID.

Tracking cookies

- Every time the browser visits a site containing the agency's advertisements, it sends the cookie (the random ID) along with the URL of the page that is being read (using the referer field) to the agency.
- This enables the agency to track a single user's behaviour across multiple web sites.

Countermeasures

- Software can be used to detect tracking cookies and eliminate them (and, in some cases, even prevent them being loaded).
- Sources of software include:
  - www.spybot.info (for Spybot Search and Destroy), and
  - www.lavasoftusa.com (for Ad-Aware 6.0)

Referer field

- One of the fields in the header of an HTTP request message is the Referer field.
- This allows the client to specify, for the server's benefit, the address (URI) of the resource from which the URI of this request was obtained.
- In most browsers, when you look at a new page, the browser will send the URL of the current page in the referer field.
- Under the HTTP definitions, this is means to be an option for the user, but according to Garfinkel and Spafford, they have never seen a browser where it is optional.

OWASP Top Ten – I

- The Open Web Application Security Project (OWASP) is an open community dedicated to improving the security of web applications.
- The OWASP Top Ten is a project to collate information on what the most critical web application security flaws are.

OWASP Top Ten – II

1. *Unvalidated Input*
2. *Broken Access Control*
3. *Broken Authentication and Session Management*
4. *Cross Site Scripting*
5. *Buffer Overflow*
6. *Injection Flaws*
7. *Improper Error Handling*
8. *Insecure Storage*
9. *Application Denial of Service*
10. *Insecure Configuration Management*

Unvalidated Input

- *Unvalidated Input*:
  - Covers attacks types such as: *cross site scripting*; *buffer overflows*; *format string attacks*; *SQL injection*.
  - One way to protect the web server is to filter out malicious input – this has the problem that there are a large number of ways of encoding information.
  - Other applications use only client-side mechanisms to validate input – but these are easily bypassed.
  - The best way to defend against these types of attacks is to check against a strict format that specifies what will be allowed.
  - Validate against a "positive" specification:
    - Data type; allowed character set; minimum and maximum length; …

Cross Site Scripting – I

- *Cross Site Scripting* (XSS):
  - When an attacker uses a web application to send malicious code to a different end user.
  - Can occur anywhere a web application uses input from a user in the output it generates without validating it.
  - Victim's browser has no way of knowing that the script should not be trusted, and will execute it.
  - XSS attacks can generally be categorised into two categories:
    - Stored
    - Reflected

Cross Site Scripting – II

- *Stored attacks* are those where the injected code is permanently stored on the target servers, such as in a: *database*; *message forum*; *visitor log*; …
- *Reflected attacks* are those where the injected code is reflected off the web server, such as in an error message, search result, etc.
  - They are delivered to the victim via another route, such as in an email message, or on some other web server.
  - When a user is tricked into clicking on a malicious link or submitting a specially crafted form, the injected code travels to the vulnerable web server, which reflects the attack back to the server.
  - The browser then executes the code because it came from a "trusted" server.

Cross Site Scripting – III

- XSS can cause a variety of problems for the end user.
- The most severe XSS attacks involve disclosure of the user's session cookie, allowing an attacker to hijack the user's session and take over the account.
- Other attacks include:
  - Disclosure of end user files
  - Installing a Trojan horse
  - Modifying presentation of content
- Best method of protection is to ensure that web applications perform validation of all a rigorous specification.

Web server scripting

- Most web browsers have the capability to interpret scripts embedded in the web pages downloaded from a web server.
- Such scripts may be written in a variety of scripting languages and are run by the client's browser.
- In the past most browsers were installed with the capability to run scripts enabled by default.

Impact of scripting attacks

- Users may unintentionally execute scripts written by an attacker when they follow untrusted links in web pages, mail messages, or newsgroup postings.
- Users may also unkowingly execute malicious scripts when viewing dynamically generated pages based on content provided by other users.

Scripting attack – simple example

- An attacker might post a message such as:
  - Hello message board. This is a message.
    `<SCRIPT>`malicious code`</SCRIPT>`
    This is the end of my message.

  to an Internet discussion group.
- When a victim with scripts enabled in their browser reads this message, the malicious code may be executed unexpectedly.
- Scripting tags that can be embedded in this way include `<SCRIPT>`, `<OBJECT>`, `<APPLET>`, and `<EMBED>`.

Attacks on servers

- Web servers themselves may be the victims of attacks via HTTP requests.
- For example, to cause *buffer overflow* in a web server, an attacker might induce errors at Web traffic ports by entering large character strings to find a susceptible overflow field.
- Once a field spills over into a code-executing field, an attacker will enter another string that will spill a command into the executable field.
- Buffer overflows can give an attacker access to a range of sensitive server functions.

Attacks on servers

- Certain implementations of HTTP can be used to create an *HTTP bypass*, granting access to a server's activity logging functions.
- With these implementations, a Web page can be accessed and altered without the system's Web server recording the change.
- This method is often used to deface Web pages.

Attacks on servers

- *Web-code vulnerabilities* can appear in many languages and application extensions, including VB, Visual C++, ASP, TCL, Perl, PHP, XML, CGI and Cold Fusion.
- Basically, an attacker will exploit a known weakness in an application, such as CGI scripts not checking input.

CGI scripts

- A *Common Gateway Interface* (CGI) Script is a program which is run on demand by a server to generate the content of a web page.
- If a web page has to do more than simply give an unchanging text and graphics display to the viewer, dynamic content generation, e.g. as provided by a CGI Script, is needed.

Injection Flaws – I

- *Injection Flaws*:
  - Allow attackers to relay malicious code through a web application to another system.
  - This is because many web applications use operating system features and external programs to perform their functions.
  - These include: *calls to the operating system*; *shell commands*; *calls to backend databases* (e.g. **SQL injection**).

Injection Flaws – II

- To implement SQL injection, the attacker must find a parameter that the web application passes to a database.
- By carefully embedding malicious SQL commands in the content of the parameter, the attack can trick the web application to forward the malicious query to the database.
- The consequences are particularly damaging, as an attacker can obtain, corrupt, or destroy database contents.
- Careful validation of data is required.

URL Obfuscation Attacks

- *URL Obfuscation* attacks are mechanisms used to trick users to visit an attacker's website.
  - Examples of such attacks are: *using strings*; *using @ sign*; *URL redirection*.
  - Using strings:
    - `http://254.231.52.42/ebay/account_update/now.php`
  - Using @ sign:
    - `http://www.citybank.com/update.pl@254.231.52.42/usb/upd.pl`
  - URL redirection:
    - `http://usa.visa.com/track/dyredir.jsp?rDirl=http://200.251.251.10/.verified/`

SSL/TLS overview

- SSL = Secure Sockets Layer. Current version is v3.
- TLS = Transport Layer Security. TLS 1.0 is similar to SSL 3.0 with minor tweaks.
- TLS is defined in RFC 2246.
- SSL/TLS provides security “at TCP layer”. In fact, it usually provides a thin layer between TCP and HTTP.

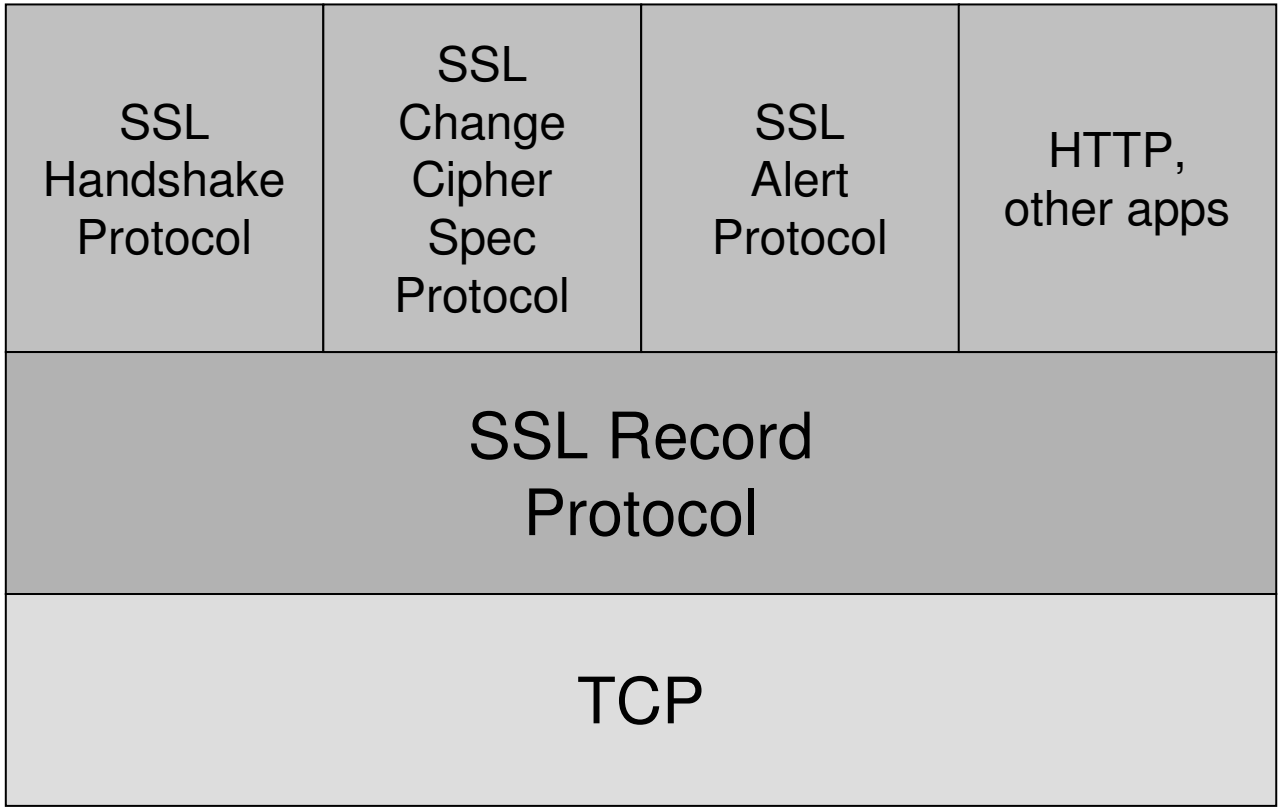
SSL/TLS basic features

- SSL/TLS widely used in Web browsers and servers to support “secure e-commerce” over HTTP.
  - Built into Microsoft IE, Netscape, Mozilla, Apache, IIS, …
  - Presence of SSL protected link indicated by the browser padlock symbol.

SSL architecture

- SSL architecture involves two layers:
  - *SSL Record Protocol*
    - Lower layer providing secure, reliable channel to upper layer.
  - *Upper layer* carrying:
    - SSL Handshake Protocol,
    - Change Cipher Spec. Protocol,
    - Alert Protocol,
    - HTTP,
    - Any other application protocols.

SSL architecture



SSL Record Protocol

- Carries application data and “management” data.
- Sessions:
  - Sessions created by handshake protocol.
  - Defines set of cryptographic parameters (encryption and hash algorithm, master secret, certificates).
  - Carries multiple connections to avoid repeated use of expensive handshake protocol.
- Connections:
  - State defined by nonces, secret keys for MAC and encryption, IVs, sequence numbers.
  - Keys for many connections derived from single master secret created during handshake protocol.

SSL Record Protocol

- SSL Record Protocol provides:
  - Data origin authentication and integrity.
    - MAC using algorithm similar to HMAC, based on MD5 or SHA-1 hash algorithms.
    - MAC protects 64 bit sequence numb for anti-replay.
  - Confidentiality.
    - Bulk encryption using symmetric algorithm (IDEA, RC2-40, DES-40 (exportable), DES, 3DES, RC4-40 and RC4-128.

SSL Record Protocol

- Data from application/upper layer SSL protocol partitioned into fragments (max size 2^{14} bytes).
- MAC first, then pad (if needed), and finally encrypt.
- Prepend header containing: Content type, version, length of fragment.
- Submit to TCP.

SSL Handshake Protocol

- SSL needs secret keys:
  - Used for MAC & encryption at Record Layer.
  - Different keys in each direction.
- These keys are established as part of the SSL Handshake Protocol.
- The SSL Handshake Protocol is a complex protocol with many options.

SSL Handshake Protocol security goals

- Entity authentication of participating parties (**client** and **server**).
  - Server nearly always authenticated, client more rarely.
  - Appropriate for most e-commerce applications.
- Establishment of a fresh, shared secret.
  - Shared secret used to derive further keys for SSL Record Protocol.
- Secure ciphersuite negotiation (including encryption and hash algorithms).

SSL Handshake Protocol – key exchange

- SSL supports several key establishment mechanisms.
- Most common is RSA encryption.
  - Client chooses `pre_master_secret`, encrypts it using public RSA key of server, and sends to server.
- Can also create `pre_master_secret` using one of several variants of Diffie-Hellman key establishment protocol.

SSL Handshake Protocol – entity authentication

- SSL supports several different entity authentication mechanisms.
- Most common based on RSA:
  - The ability to decrypt `pre_master_secret` and generate correct MAC using keys derived from `pre_master_secret` authenticates the server to the client.
- DSS or RSA signatures on nonces (and other fields, e.g. Diffie-Hellman values).

SSL key deriviation

- Keys used for MAC and encryption derived from `pre_master_secret`:
  - Derive `master_secret` from `pre_master_secret` and client/server nonces using MD5 and SHA-1.
  - Derive key material from `master_secret` and client/server nonces, by repeated use of hash functions.
  - Split key material into MAC and encryption keys as needed.

SSL Handshake Protocol run

- We choose the most common use of SSL for illustration:
  - No client authentication.
  - Client sends `pre_master_secret` using Server's public encryption key from Server certificate.
  - Server authenticated by ability to decrypt to obtain `pre_master_secret`, and construct correct finished message.

SSL Handshake Protocol run

M1: C → S: `ClientHello`

- Client initiates connection.
- Sends client version number.
  - 3.1 for TLS.
- Sends `ClientNonce`.
  - 28 random bytes plus 4 bytes of time.
- Offers list of ciphersuites.
  - key exchange and authentication options, encryption algorithms, hash functions, e.g. `TLS_RSA_WITH_3DES_EDE_CBC_SHA`.

SSL Handshake Protocol run

M2: S → C: `ServerHello`, `ServerCertChain`, `ServerHelloDone`

- Sends server version number.
- Sends `ServerNonce` and `SessionID`.
- Selects single ciphersuite from list offered by client, e.g. `TLS_RSA_WITH_3DES_EDE_CBC_SHA`.
- Sends `ServerCertChain` message.
  - Allows client to validate server's public key.
- (optional) `CertRequest` message.
  - Omitted in this protocol run – no client authentication.
- Finally, `ServerHelloDone`.

SSL Handshake Protocol run

M3: C → S: `ClientKeyExchange`, `ChangeCipherSpec`, `ClientFinished`

- `ClientKeyExchange` contains encryption of `pre_master_secret` under server's public key.
- `ChangeCipherSpec` indicates that client is updating cipher suite to be used in this session.
  - Sent using SSL Change Cipher Spec. Protocol.
- (optional) `ClientCertificate`, `ClientCertificateVerify` messages.
  - Only when client is authenticated.
- Finally, `ClientFinished` message.
  - A MAC on all messages sent so far (both sides).
  - MAC computed using `master_secret`.

SSL Handshake Protocol run

M4: S → C: `ChangeCipherSpec`, `ServerFinished`

- `ChangeCipherSpec` indicates that server is updating cipher suite to be used on this session.
  - Sent using SSL Change Cipher Spec. Protocol.
- Finally, `ServerFinished` message.
  - A MAC on all messages sent so far (both sides).
  - MAC computed using `master_secret`.
  - Server can only compute MAC if it can decrypt `pre_master_secret` in M3.

SSL Handshake Protocol run

Summary:

M1: C → S: `ClientHello`

M2: S → C: `ServerHello`, `ServerCertChain`, `ServerHelloDone`

M3: C → S: `ClientKeyExchange`, `ChangeCipherSpec`, `ClientFinished`

M4: S → C: `ChangeCipherSpec`, `ServerFinished`

SSL Handshake Protocol run

1. Is the client authenticated to the server in this protocol run?
2. Can an adversary learn the value of `pre_master_secret`?
3. Is the server authenticated to the client?
    1. No.
    2. No. Client has validated server's public key; only the holder of the private key can decrypt `ClientKeyExchange` to learn `pre_master_secret`.
    3. Yes. `ServerFinished` includes MAC on nonces computed using key derived from `pre_master_secret`.

Other SSL Handshake options

- Many optional/situation-dependent protocol messages:
  - M2 (S → C) can include:
    - `ServerKeyExchange` (e.g. for DH key exchange).
    - `CertRequest` (for client authentication).
  - M3 (C → S) can include:
    - `ClientCert` (for client authentication).
    - `ClientCertVerify` (for client authentication).

Other SSL protocols

- Alert protocol.
  - Management of SSL session, error messages.
  - Fatal errors and warnings.
- Change cipher spec protocol.
  - Not part of SSL Handshake Protocol.
  - Used to indicate that entity is changing to recently agreed ciphersuite.
- Both protocols run over Record Protocol (so peers of Handshake Protocol).

SSL and TLS

- TLS 1.0 = SSL 3.0 with minor differences:
  - TLS signalled by version number 3.1
  - Use of HMAC for MAC algorithm.
  - Different method for deriving key material (`master-secret` and `key-block`).
    - Pseudo-random function based on HMAC with MD5 and SHA-1.
  - Additional alert codes.
  - More client certificate types.
  - Variable length padding (can be used to hide lengths of short messages and so frustrate traffic analysis).
  - And more…

SSL/TLS applications

- Secure e-commerce using SSL/TLS.
  - Client authentication not needed until client decides to buy something.
  - SSL provides secure channel for sending credit card information.
  - Client authenticated using credit card information, merchant bears (most of) risk.
  - Very widely used.

SSL/TLS application issues

- Secure e-commerce: some issues.
  - No guarantees about what happens to client data (including credit card details) after session: may be stored on insecure server.
  - Does client understand meaning of certificate expiry and other security warnings?
  - Does client software actually check complete certificate chain?
  - Does the name in certificate match the URL of e-commerce site? Does the user check this?
  - Is the site the one the client thinks it is?
  - Is the client software proposing appropriate ciphersuites?

SSL/TLS application issues

- Secure electronic banking:
  - Client authentication may be enabled using client certificates.
    - Issues of registration, secure storage of private keys, revocation and re-issue.
  - Otherwise, SSL provides secure channel for sending username, password, mother's maiden name, …
    - What else does client use same password for?
  - Does client understand meaning of certificate expiry and other security warnings?
  - Is client software proposing appropriate ciphersuites?
    - Enforce from server.

Books

- Any modern book on computer networking will cover the web, HTTP, HTML, Cookies, etc.
- Eric Rescorla, *SSL and TLS: Designing and Building Secure Systems*. Addison Wesley, 2000. [There are other books on this topic].
- D. Stein, *Web Security*, Addison Wesley, 1998.
- S. Laurent, *Cookies*. McGraw Hill, 1998.

Standards

- All IETF RFCs can be obtained from: `www.ietf.org`
- The W3C recommendations are available at: `www.w3.org`
- The international standardisation body ISO has published the SGML standard (a catalogue of current standards is available at `www.iso.ch`).
- For general information about security standards see: A. W. Dent and C. J. Mitchell: *User's guide to cryptography and standards* (Artech House, 2004).
  - `http://www.isg.rhul.ac.uk/ugcs`

Acknowledgements

- Some information taken from original lecture notes by Chris Mitchell.
- SSL/TLS discussion based on an (abbreviated version of) Kenny Paterson's lecture on Secure Network Protocols.
- Information derived from a number of web sites, including the W3C web site (lots of useful tutorial information there).
- Some additional information from: Simson Garfinkel and Gene Spafford – *Web Security, Privacy & Commerce*
- Information taken on the OWASP Top 10:
  - `http://www.owasp.org/index.php/Top_10_2004`
- A more up to date version is available from:
  - `http://www.owasp.org/index.php/OWASP_Top_Ten_Project`

Conclusions

- After today's lecture you should:
  - Have a basic understanding of how the components that make up the web work.
  - Understand what are the security problems faced by clients and servers using the web as an interface.
  - Be able to describe a high level overview of how SSL allows us to build secure connections between clients and servers.
  - Be able to appreciate that security of web applications does not just start and end with SSL.