Chapter 1 - Web Application (In)security

The evolution of Web applications

- Browsers used to treat each request equally, all static pages

Common web application Functions

- Most mobile applications employ either a browser or a customiszed client that uses HTTP-based APIs to communicate with the server

Highly sensitive data and functionality:

- HR applications = payroll information
- Admin interfaces = key infrastructure, web, mail serves, user workstations.
- Collaboration software = sharing documents, tracking workflow and projects
- Business applications such as enterprise resource planning (ERP) software
- email, Outlook with web access

All of these “internal” applications are increasingly hosted externally as cloud appications

Benefits of Web Applications

- HTTP is lightweight and connectionless
- HTTP can be proxied and tunneled over other protocols
- Every user has pre installed web broswer and thus client
- core technoligies and languages

“This site is secure”

% of affected websites in a 100+ sample

- Broken authentication 62%
- Broken access controls 71%
- SQL injection 32%
- Cross-site scripting 94%
- Information leakage 78%
- Cross site request forgery 92%

The core security problem: Users can submit arbitrary input

- The application must assume that all input is potentially malicious
- users can interfere with any piece of data transmitted between the client and server, including request parameters, cookies, HTTP headers
- Users can send requests in *any sequence* and can submit parameters at a different stage than the application expect, more than once, or not at all
- Users are do not need to use web browsers, there are tools to provided inexpected results

Examples of submitting crafted input to achieve this objective:

- Changing the price of a product transmitted to a hidden HTML form field to fraudulently purchase the product for a cheaper amount
- Modifying the session token to hijack the session of another authenticated user
- Removing certain parameters that are normally submitted to exploit a logic flaw in the application’s processing
- Altering some input that will be processed by a back-end database to inject a malicious database query and access sensitive data

Key problem factors

Underdeveloped security awareness

- Security is not in the developer's mind when deving
- Developers rely to much on the inbuilt security of their frameworks

Custom development

- Most companies will have custom components which are not industry standard

Deceptive Simplicity

- Frameworks make it very easy to get an *ok* standard of product, this tricks new developers into not making secure results
- Frameworks include Liferat and Appfuse

Rapidly Evolving Threat profile

- A development team that begins a project with complete knowledge of current threats may have lost the status by the time the application is completed or deployed

Resource and Time Constraints

- Deadlines and costs will often deter companies from properly considering security

Overextended technologies

- Technologies are made to solve problems *of the time* when a tech is stretched to meet new demands, security issues can occur

The new security perimeter

- organisation *used* to only care about perimeter defences, with modern web apps where the user can interact with backend databases, apis and more perimeter defences are not enough
- You don't even need to attack the backend anymore you can leverage the web app to attack users

The future of web app security

- Well known vulns are becoming less common and are now *usually* only found with context specific edge cases
- Web 2.0 refers to the greater use of functionality that enables user-generated content and information sharing
- Cloud computing refers to greater use of external service providers