World Wide Web policy

Agenda

1. Introduction

- What does World Wide Web security mean?

Webmasters=> confidence that their site won’t be hacked or used as a gateway to get into their LANS

Web users=> it is the ability to browse securely through the web

But in general…

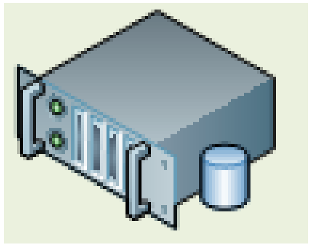
Introduction

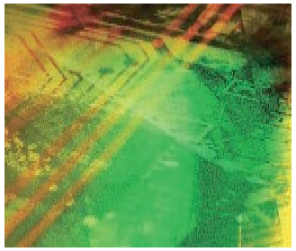
- World Wide Web security

Procedures

Technologies

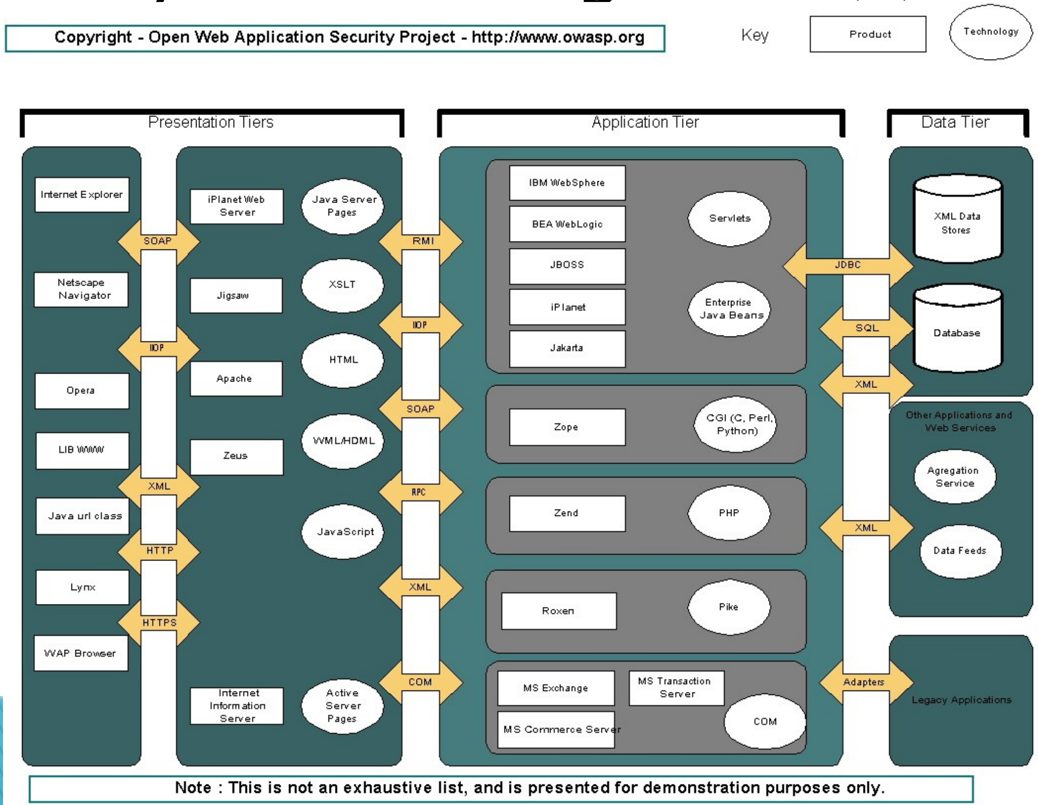
Practices





2. Web Application

- Web Application is a client/server software application that interacts with users or other systems using HTTP(S) .



3. Some “Components”

3.1. Authentication
3.2. Browser Security
3.3. Scripts and Active Code
3.4. New Technologies : e.g. Ajax

3.1 Authentication

Process of determining if a user or entity is who he/she claims to be.

- HTTP basic
- HTTP digest

For secure authentication

- SSL (https://...)

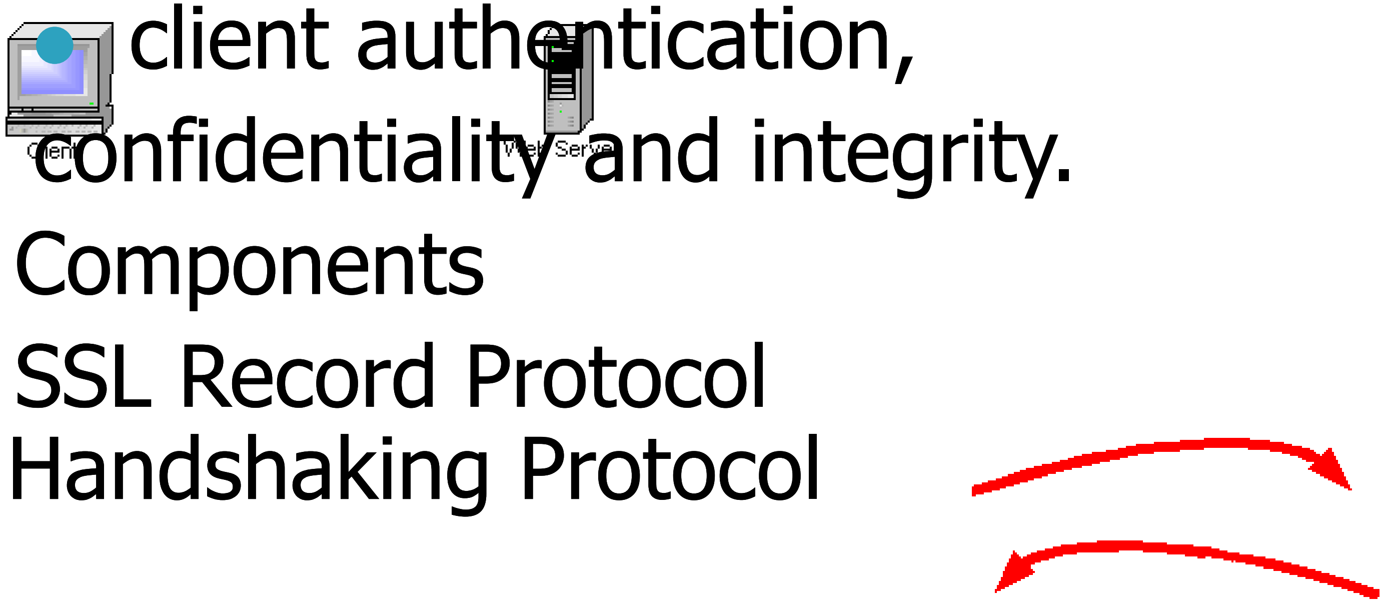
  Protect transactions in any of the TCP protocols such as HTTP, NNTP (News Transfer), FTP, among others.

Authentication

- Provides server authentication,
- client authentication,
- confidentiality and integrity.

Components

SSL Record Protocol
Handshaking Protocol



3.2 Browser Security

- *User privacy*

  Use a strong password.

  Install the latest version of your web browser.

Browser Security

- *Cookie*

  Data file originated by a web server, with the client's information (machine name, keystrokes the user types, etc)

  Types { Per-session , secure
  Persistent , nonsecure

  Cookies = vulnerability ~ privacy
- Structure Of A Cookie

Domain	Flag	Path	Secure	Expiration	Name	Value
www.redhat.com	FALSE	/	FALSE	1154029490	Apache	64.3.40.151.16018996349247480

Browser Security

- *Increasing the level of security:*

For user:

1. Limit the cookies per web site.
2. Allow cookies from the site that you are visiting for session.
3. Disabled cookies if you are using a public computer.

Browser Security

For Web designers:

1. Examine cookies that they are accepting to avoid malicious content.
2. Avoid the use cookies for authentication.
3. Store as little private or personal information from the user as possible.

3.3 Scripts and Active Code

- *Scripts*

  Programs executed on the server side performing advanced operations.

  E.g: perl, c, php, etc

- *Active Code*

Programs designed to perform detailed task on the client’s side.

E.g: javaScript, Java Applets, ActiveX,…

Scripts and Active Code

- ***Vulnerabilities***

Misusing interpreters: putting the script interpreter in the same place as the scripts directory.

http://www.victim.com/cgi-bin/perl.exe?-e+%27unlink+%3C*%3E%27

Web Server --> perl –e unlink ‘<*>’

Flawed memory management: is in the domain of programming languages that do not perform memory management internally such as c, c++.

Scripts and Active Code

- Passing unchecked user input to command interpreters: user input is passed to a command shell, allowing remote users to execute shell commands on the web server.
- Opening files based on unchecked user input.
- When writing user inputs to disk.

Scripts and Active Code

- *Security Model*
- Java Applets => sandbox
- JavaScript
  - sandbox
  - same origin policy
  - object signing

Scripts and Active Code

- *ActiveX*

  Is a binary code that extend the functionality of a web application; it can take any action as the user.

  Security is partially controlled by the web designer and a third party.

  - Security Options safe for initializing
    safe for scripting

3.4 AJAX (Asynchronous JavaScript and XML)

- presentation management using XHTML, CSS, and the Document Object Model;
- Asynchronous data retrieval using XMLHttpRequest; and,
- JavaScript

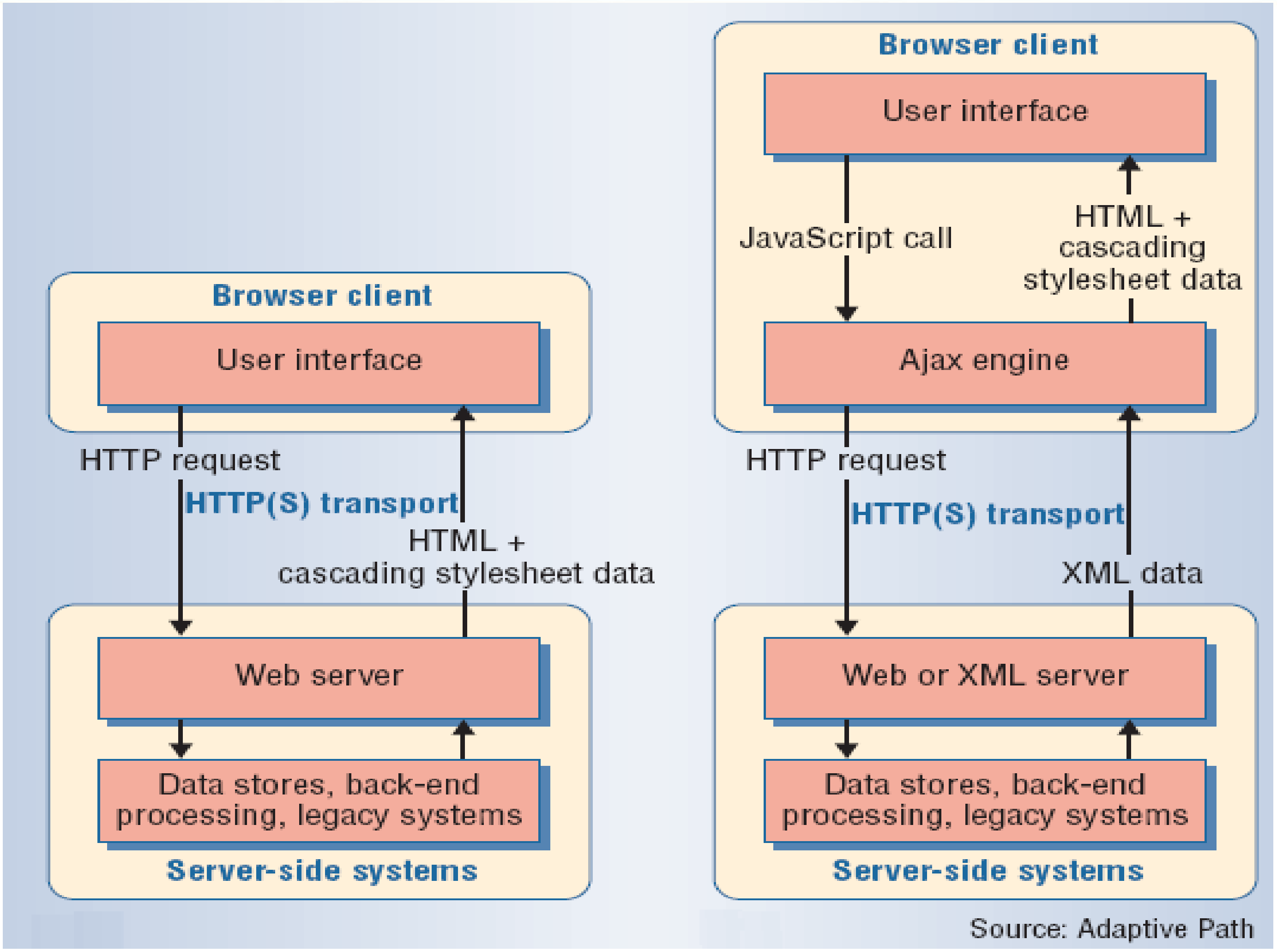
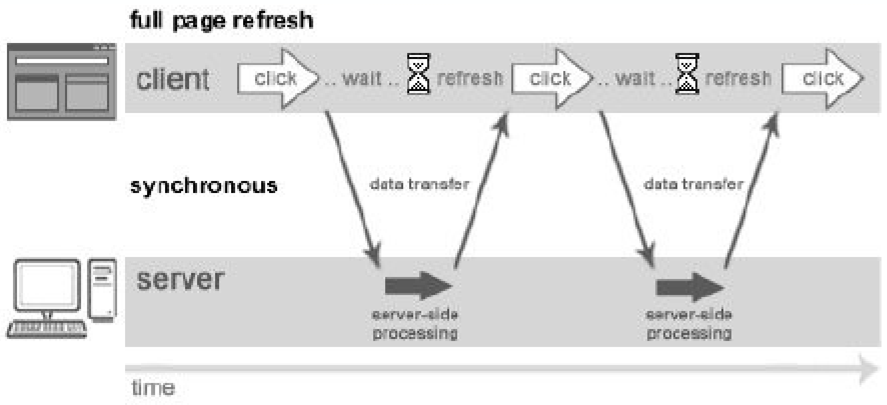


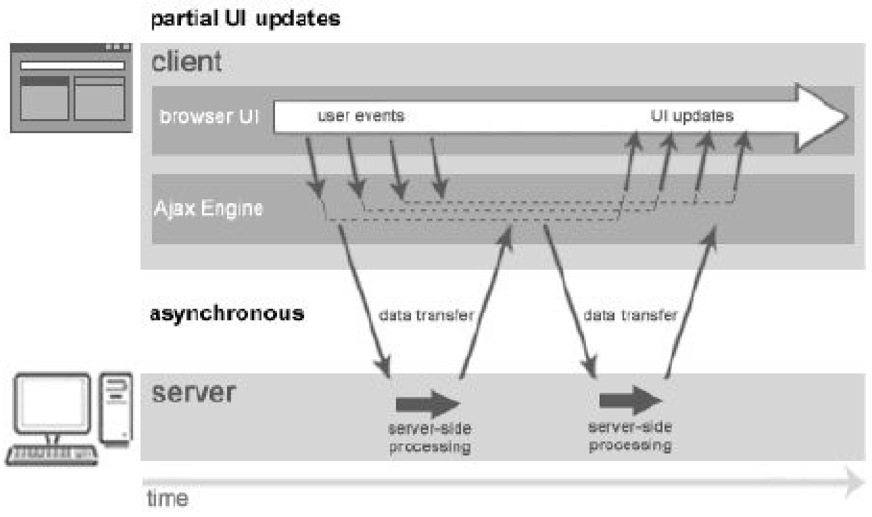
Image from http://www.adaptivepath.com/publications/essays/archives/000385.php/

AJAX

- Synchronous



- Asynchronous



4. Common Vulnerabilities

- *Cross Site Scripting (XSS)*
- *Cross Site Request Forgery (XSRF)*
- *Sql Injection*

Common Vulnerabilities

- *Cross Site Scripting (XSS)*

  An attacker inject malicious code, usually client-side scripts, into web applications from outside sources .

  $$\text{Types}\left\{\begin{array}{l}\text{– Stored}\\ \text{– Reflected}\end{array}\right.$$

  Due to lack of input/output filtering

Common Vulnerabilities

- Reflected Cross Site Scripting:

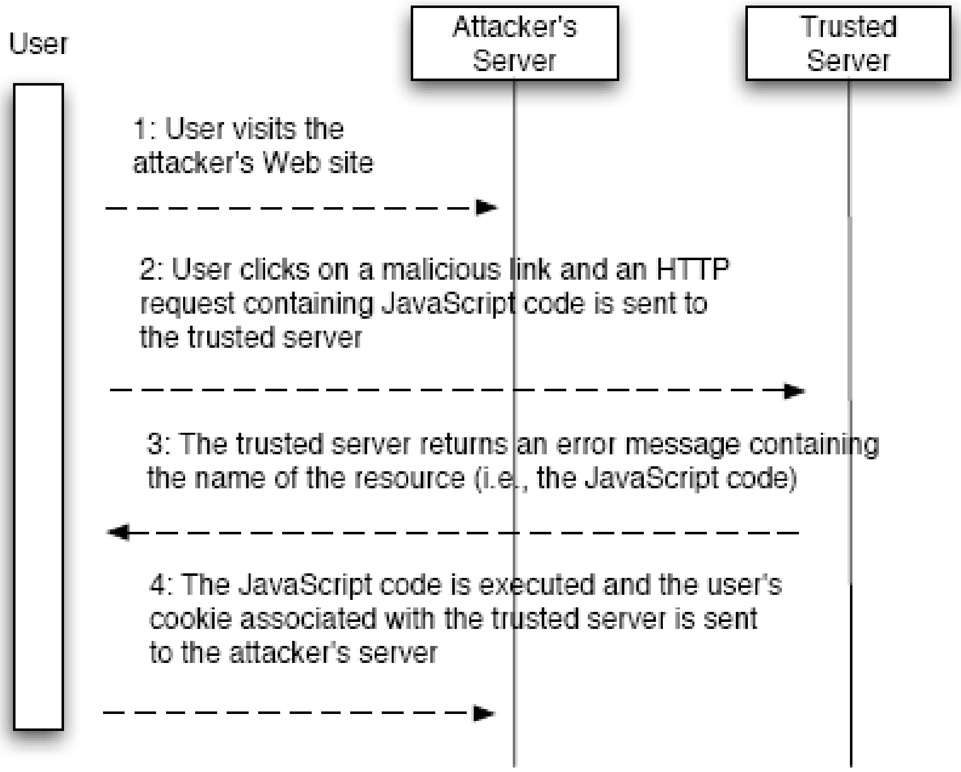

User
Attacker's Server
Trusted Server
1: User visits the attacker's Web site
2: User clicks on a malicious link and an HTTP request containing JavaScript code is sent to the trusted server
3: The trusted server returns an error message containing the name of the resource (i.e., the JavaScript code)
4: The JavaScript code is executed and the user's cookie associated with the trusted server is sent to the attacker's server

Image from Noxes:A Client-Side Solution for mitigating cross-site scripting attacks

Common Vulnerabilities

- *Cross Site Request Forgery (XSRF)*

  Merely transmits unauthorized commands from a user the website trusts.

  It is related with the predictable of the structure of the application.

Common Vulnerabilities

- *Sql Injection*

  An attacker adds SQL statements through a web application's input fields or hidden parameters to gain access to resources or make changes to data

Common Vulnerabilities

- Sql Injection

SELECT * FROM users WHERE login = ‘Bush‘ AND password = '123'

(If it returns something then login!)

PHP/PostgreSql Server login syntax

$sql = "SELECT * FROM users WHERE login = '" . *$formusr* . "' AND password = '" . *$formpwd* . "'";

Common Vulnerabilities

- Sql Injection

Injecting through Strings

$formusr = ' or 1=1 – –

$formpwd = anything

Final query would look like this:

SELECT * FROM users

WHERE username = ' ' or 1=1

– – AND password = 'anything'