Web Security

Homework 1 Recap of Compartments in Biba

Word of god                          Religion A                          Religion B
|                                   |                                   |
Top guy                             |                                   |
|                                   |                                   |
Subsidiary                          |                                   |

Example:
(TopGuy, {ReligionA})

Web Security (Many Different Bugs)

Client Side vs Server side = vulnerable because of javascript running on client, client’s machine isn’t controlled by developer.

Goals:
- 1. Protect web apps from malicious clients
- 2. Protect users from malicious web apps

Things that might go wrong:
Since the browser is outside our security domain, Domain cannot trust browser to enforce security checks.
Picture a web app where a user can type in a quantity (like amount of items to order). One security goal should be that the order number can never exceed the quantity in stock.

**Client side security checking** = Big no no, the malicious user can disable this script because it’s their browser and then still be able to enter whatever quantity they desire.

---

**SQL Injection**

*freshmeat.net* - Sourceforge for web apps to exploit

```java
int LoginUser(String name, String Password)
{
    String Query =
        "SELECT * FROM users WHERE username='" + name + "' and password='" + password + "' j";
    Sql.execute(Query); }
```

Example of Good Query:
“SELECT * FROM users WHERE username =’Rob’ and passwd = ‘opensesame’;

Bug Example:
passwd = ‘ ‘ or 1 == 1’ ; - - ‘;

How to Avoid this type of bug:
Use:
Prepared Statements

Prepared statement ps =
new PreparedStatement |
"SELECT * from users WHERE
username = '1' and
passwd = '2';";

ps.setStringParam(1,name);
ps.setStringParam(2,passwd);
sqp.executePreparedStatement(ps);

To verify that you don’t have unprepared statements
you can use GREP on your source code.

Cross-Site Scripting Bugs (XSS)

Background info:

Web Authentication and Access Control

Cookie: small amount of data stored on client and submitted w/ each request

You have a browser and server, the browser will say “GET login.html” to the server.
The Server will respond w/ login.html.
The browser will POST login info from the login.html page.
The Server will send you your accountInfo.html or w/e page it takes you to.
It will also send you a SETCOOKIE, which has a name and a value.
So now every time your browser accesses the server it will send the COOKIE with the info.

Problem with this system:
- You can fake a cookie
- You can steal a cookie
   ^^^FireSheep = catches cookies over unencrypted network.

What we need to Fix this:
- Unforgeable cookies
- Unstealable cookies

The way we make unstealable cookies is by encrypting the whole connection in SSL

Cookies can also have a flag that says, SSL only, and won't transmit the cookie unless the connection is encrypted.
The way to make unforgeable cookies is by using a MAC (Message Authentication Code).

MAC k (msg) = tag

Each cookie only gets sent to the site that set it. It also only allows that same origin’s Java script to read, edit, etc… that cookie. (Same Origin Policy)