

Attacks Against Websites

Tom Chothia
Computer Security, Lecture 10

Introduction

- Last lecture covered the basics of how websites work.
- This lecture describes some of the things that can go wrong:
 - SQL injection attacks
 - Stolen cookies
 - URL hacking

Reminder: Websites

- HTML and HTTP
- SSL/TLS
- Cookies for state
- JavaScript for client side computation
- e.g. JSP for server side computation
- SQL database back end.

JavaScript

- JavaScript provides no security what so ever.
- It should never be used for security.
- Never use JavaScript for a password check.

Eavesdropping

- If the connection is not encrypted, it is possible to eavesdrop, by
 - ISP,
 - anyone on the route
 - Anyone on your local network, e.g. using the same wi-fi.

Eavesdropping

- Log in should be done using SSL (https)
- This makes it impossible to eavesdrop the password.
- After login many websites drop to http
- But, how does it keep track of who I am?

Authenticating web users

- IP address-based
 - NAT may cause several users to share the same IP
 - DHCP may cause same user to have different IPs
- Certificate-based
 - Who has a certificate and what is it, and who will sign it?
- Cookie-based
 - The most common

Simple authentication scheme

- A Capability List access control system.
- The Web Application:
 - Verifies the credentials, e.g., against database
 - Generates a cookie which is sent back to the user
 - Set-Cookie: auth=secret
- When browser contacts the web site again, it will include the session authenticator
 - Cookie: auth=secret

Steal the Cookie

- So the attacker doesn't need the username and password
 - Just the cookie
- If the website uses https (SSL) it's safe
- But many websites drop back down to http after a secure login.

Countermeasure –

- Use https (SSL) all the time.
 - But an attacker can force the user to drop to http
- Set the secure flag, cookie is sent only over secure connections:

```
Cookie secureCookie = new Cookie("credential",c);
secureCookie.setSecure(true);
```

SQL Injection Attacks

<http://www.eshop.co.ukaction=buy&product=17453>

Web server might look up "17453" in a SQL database using:

```
...
SELECT * FROM products WHERE code='17453';
...
INSERT INTO sales VALUES (id, customer, 17453)
...
```

SQL Injection Attacks

<http://www.eshop.co.ukaction=buy&product=X>
=>

```
SELECT * FROM products WHERE code='X';
```

What else could we use for X?

```
X= ' ; DROP TABLE products; --
```

=>

```
SELECT * FROM products WHERE code="'; DROP
TABLE products; --'
```

SQL Injection Attacks

Secret Item: dh2*%Bgo

=>

```
SELECT * FROM users WHERE item = 'dh2*%Bgo';
```

If found, then item details are given.

SQL Injection Attacks

Secret Item: ' OR '1' = '1' --

=>

```
SELECT * FROM users WHERE item = '' OR '1' = '1'; --'
```

1 does equal 1! Therefore I will return all item's details

Fixing SQL attacks

- You can stop SQL attacks by checking the input.
- For more on SQL attacks and defense see the [Secure Programming](#) module
- All user controlled data must be "sanitized"
 - often done wrong / in wrong place.
 - Hard to keep track of which inputs the attacker can control.



Not Just Websites.



Any source of data can be used for an SQL attack

- 2D bar
- RFID cards





URL hacking

- The user can type anything they want into the URL bar, or even form the request by hand.

`http://nameOfHost/filePath`

- Attacker can try to guess filenames,
 - Guessable directory names will be found.

Path Transversal

- The user can type anything they want into the URL bar, or even form the request by hand.

`http://nameOfHost/../../../../etc/shadow`

- If the webserver is running with root permission this will give me the password file.

Path Transversal: Fix

- Use access control settings to stop Path Transversal.
- Best practice, make a specific user account for the webserver.
- Only give that account access to public files.

Query String

Query strings are used to tell dynamic webpages (e.g. php and JSP).

```
http://myWebShop.com/index.php?
                                account=tpc&action=add
http://myWebShop.com/index.php?
                                account=tpc&action=show
```

What if the attacker tries:

```
http://myWebShop.com/index.php?
                                account=admin&action=delete
```

Fix

No security through obscurity

Never rely on just the URL request for authentication.

E.g. Use cookies to control access.

Summary

- This lecture showed you some basic web attacks and how to stop them:
 - SQL injection attacks
 - Stolen cookies
 - URL hacking
- Next lecture, more web attacks:
 - Cross site scripting attacks (XSS)
 - Cross-site request forgery (CSRF)