Web Security (II)

Dawn Song
dawnsong@cs.berkeley.edu

Administrative Stuff

- Proposal feedback
  - Revised proposal due Oct 22
    - Timeline
    - More clear description of problem & approach
  - Feedback on Oct 23
    - 3:30-5:30pm
    - Each group 10mins
    - Sign-up sheet

- BitBlaze info session
  - 5pm, Soda 405

Access Control in OS & Browser

- Access control in OS
  - Principals
  - Resources
  - Policies?

- Access control in Browser
  - Principals
  - Owner of web content
  - Resources
    - Memory: heap of script objects
    - Persistent state: cookies
    - Display: HTML DOM
    - Network communication
  - Policies?
Same-Origin Principle (SOP)

- Documents or scripts loaded from one origin cannot get or set properties of documents from a different origin.
- Origin:
  - Two pages have the same origin if the protocol, port, host are the same for both pages.
- The origin of a script:
  - The origin that a script is loaded is the origin of the document that contains the script rather than the origin that hosts the script.
  - E.g., a.com/service.html contain <script src=http://b.com/lib.js>, can lib.js access a.com’s or b.com’s HTML DOM objects?

Problems with SOP

- Rigid: all-or-nothing
  - Insufficient for Mashup.
- Too coarse-grained if site hosts unrelated pages:
  - Example: Web server often hosts sites for unrelated parties:
    - http://www.example.com/account/
    - http://www.example.com/otheraccount/
  - Same-origin policy, allows script on one page to access properties of document from another.

Trust Models in Mashup

- Content provider P, content integrator T

<table>
<thead>
<tr>
<th>P trusts T to access P's content</th>
<th>T trusts P to access T's resources</th>
<th>Content type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 No</td>
<td>No</td>
<td>isolated access-controlled</td>
</tr>
<tr>
<td>2 No</td>
<td>Yes</td>
<td>access-controlled</td>
</tr>
<tr>
<td>3 No</td>
<td>Yes</td>
<td>open</td>
</tr>
<tr>
<td>4 Yes</td>
<td>No</td>
<td>unauthorized</td>
</tr>
<tr>
<td>5 Yes</td>
<td>Yes</td>
<td>open</td>
</tr>
</tbody>
</table>
Policy Enforcement

<table>
<thead>
<tr>
<th>Feature 1 is ac</th>
<th>Feature 2 is ac</th>
<th>Content Type</th>
<th>Abstracts</th>
<th>Role or Principal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>elite</td>
<td>&lt;Privacy&gt;</td>
<td>Provider</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>elite</td>
<td>&lt;Security&gt;</td>
<td>Provider</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>randomized</td>
<td>&lt;Security&gt;</td>
<td>User</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>open</td>
<td>&lt;Security&gt;</td>
<td></td>
</tr>
</tbody>
</table>

- What are the OS analogous counterpart?

What Other Methods Can We Design to Address These Problems?

- **Capabilities**
  - How capabilities may be used here?
  - Advantages?
  - Disadvantages?

- **Crypto**
  - How crypto may be used here?
  - Advantages?
  - Disadvantages?

- What other methods?

Discussion

- How to compare with Tahoma?

- **Open Mic**
  - Questions, comments?
Input Validation in Web Security

- System takes input strings
- Incorporates input into output
- Output is interpreted
- Unexpected input may cause problems
- Examples
  - SQL Command Injection Attack
    - 60% web applications vulnerable
    - 100ks of private records exposed in 1 attack
  - Cross-site scripting (XSS) attack
    - More than 21% vulnerabilities reported to CVE
    - #1 reported vulnerability, surpassing buffer overflows

Defenses

- Input filtering
  - Issues?
- MashupOS' defense against XSS?
- Other methods?