Announcements

• Class cancelled on Wednesday
Web Security

CS642: Computer Security

Professor Ristenpart

http://www.cs.wisc.edu/~rist/

rist at cs dot wisc dot edu

Liberal borrowing from Mitchell, Boneh, Stanford CS 155

University of Wisconsin CS 642
Web security part 1

Basic web security models

Browser security

Same-origin policy / Navigation policy

Cookies / Session handling
WWW

Tim Berners-Lee and Robert Cailliau 1990
HTTP, CERN httpd, gopher

1993 Mosiac web browser (UIUC, Marc Andreesen)

1994 W3C WWW Consortium --- generate standards
Gopher started charging licensing fees
(Univ of Minnesota)
Nowadays: ecosystem of technologies

- HTTP / HTTPS
- AJAX
- PHP
- Javascript
- SQL
- Apache
- Ruby
- http://w3schools.com/
Threat model

User

network attacker

Internet

bank.com

attacker.com

Network security
Oct 11, 2011
- SLMAP: Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection, by Psued and Newsham
- Slides (PDF)
Network security
Oct 13, 2011
- DNS cache poisoning, by Steve Friedl
- A survey of BGP security, Butler et al.
- Slides (PDF)
Network security
Oct 18, 2011
- Guest lecture by David Pater
Cryptography
Oct 25, 2011
- Overview of TLS (HTTPS), symmetric encryption, classical encryption, Shannon security, one-time pad encryption
- A Method for Obtaining Digital Signatures and Public Key Cryptosystems, by Rivest, Shamir, and Adleman
- Communication Theory of Secrecy Systems, by Shannon (for reference only)
- Slides (PDF)
Some basics of HTTP

URL's only allow ASCII-US characters. Encode other characters:

%0A = newline
%20 = space

Special characters:
+ = space
? = separates URL from parameters
% = special characters
/ = divides directories, subdirectories
# = bookmark
& = separator between parameters
HTTP Request

- Method: GET
  - File: /index.html
  - HTTP version: HTTP/1.1
  - Headers:
    - Accept: image/gif, image/x-bitmap, image/jpeg, */*
    - Accept-Language: en
    - Connection: Keep-Alive
    - User-Agent: Mozilla/1.22 (compatible; MSIE 2.0; Windows 95)
    - Host: www.example.com
    - Referer: http://www.google.com?q=dingbats

GET: no side effect
POST: possible side effect
HTTP Response

HTTP/1.0 200 OK
Date: Sun, 21 Apr 1996 02:20:42 GMT
Server: Microsoft-Internet-Information-Server/5.0
Connection: keep-alive
Content-Type: text/html
Last-Modified: Thu, 18 Apr 1996 17:39:05 GMT
Set-Cookie: ...
Content-Length: 2543

<HTML> Some data... blah, blah, blah </HTML>
Browser execution

• Each window (or tab):
  – Retrieve/load content
  – Render it
    • Process the HTML
    • Might run scripts, fetch more content, etc.
  – Respond to events
    • User actions: OnClick, OnMouseover
    • Rendering: OnLoad, OnBeforeUnload
    • Timing: setTimeout(), clearTimeout()
Object-oriented way to refer to objects in a web page


Methods: document.write(document.referrer)

From http://w3schools.com/html/dom/default.asp
Document Object Model (DOM)

Object-oriented way to refer to objects in a web page

**Properties:** document.alinkColor, document.URL, document.forms[], document.links[], document.anchors[]

**Methods:** document.write(document.referrer)

Browser Object Model (BOM)

window, document, frames[], history, location, navigator (type and version of browser)
Seemingly innocuous features?

- `<img src="bucky.jpg" height="50pt" width="50pt">`
- Displays an image
- What can attacker do?
Javascript timing

```html
<html><body><img id="test" style="display: none">
<script>
var test = document.getElementById('test');
var start = new Date();
test.onerror = function() {
    var end = new Date();
    alert("Total time: "+ (end - start));
}
test.src = "http://www.example.com/page.html";
</script>
</body></html>
```
Behind-firewall webapp scanning

- JavaScript can:
  - Request images from internal IP addresses
    - Example: `<img src="192.168.0.4:8080"/>
  - Use timeout/onError to determine success/failure
  - Fingerprint webapps using known image names
Browser security model

Should be safe to visit an attacker website

Should be safe to visit sites simultaneously

Should be safe to delegate content
Browser isolation

Browser is running untrusted inputs (attacker webpage)

Like all big, complex software, browser has security vulnerabilities

Browsers include “Rich Internet Applications” (RIAs) that increase attack surface:
   e.g., Adobe Flash (see reading for today by Blazakis)

Malicious website exploits browser, from there system
Web pages are not single-origin

IFrames:  `<iframe src="//site.com/frame.html" >  </iframe>`

Scripts:  `<script src="//site.com/script.js" >  </script>`

CSS:
  `<link rel="stylesheet" type="text/css" href="//site/com/theme.css" />`

Objects (flash):  `[using swfobject.js script ]
  `<script>
    var so = new SWFObject(‘//site.com/flash.swf’,…);
    so.addParam(‘allowscriptaccess’, ‘always’);
    so.write(‘flashdiv’);
  </script>`
Browser handles multiple sites, must maintain separate security contexts for each

Operating system
- Primitives
  - System calls
  - Processes
  - Disks
- Principals: Users
  - Discretionary access controls
- Vulnerabilities
  - Buffer overflows
  - root exploit
  - ...

Browsers
- Primitives
  - Document object model
  - Frames
  - Cookies / local storage
- Principals: Origins
  - Mandatory access controls
- Vulnerabilities
  - Cross-site scripting (XSS)
  - Cross-site request forgery (CSRF)
  - Cache history attacks
  - ...
Same-origin policy

• Each frame of page(s) has an origin
  – protocol://host:port
  – Origin is (protocol,host,port)

• Frame can access its own origin
  – Network access, Read/write DOM, storage (cookies)

• Frame cannot access data associated with another origin
Frame relationships:

- Child
- Sibling
- Ancestor (frame bust)
- Descendant
Frame policies

canScript(A,B) and canNavigate(A, B)

• Permissive
  – any frame can navigate any other frame

• Child
  – only can navigate if you are parent

• Descendent
  – only can navigate if you are ancestor

Which do you think should be used?
## Legacy 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>
Problems with permissive frames['right'].window.location=“evil.com/login.html”;

window.open("https://attacker.com/", "awglogin");
## Adoption of Descendant Policy

<table>
<thead>
<tr>
<th>Browser</th>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE7 (no Flash)</td>
<td>Descendant</td>
</tr>
<tr>
<td>IE7 (with Flash)</td>
<td>Descendant</td>
</tr>
<tr>
<td>Firefox 3</td>
<td>Descendant</td>
</tr>
<tr>
<td>Safari 3</td>
<td>Descendant</td>
</tr>
<tr>
<td>Opera 9</td>
<td>(many policies)</td>
</tr>
<tr>
<td>HTML 5</td>
<td>Descendant</td>
</tr>
</tbody>
</table>
UI Redressing (Clickjacking)

Defense: NoScript plugin attempts to prevent this for Firefox
Framebusting

<script type="text/javascript">
  if(top != self) top.location.replace(location);
</script>

Embed in your webpage to avoid being rendered within another (adversarial) frame.

Has limitations. See "Busting Frame Busting: a Study of Clickjacking Vulnerabilities on Popular sites"
Cookies: Setting/Deleting

GET ...

HTTP Header:
Set-cookie: NAME=VALUE ;
  domain = (when to send) ;
  path = (when to send) ;
  secure = (only send over SSL) ;
  expires = (when expires) ;
  HttpOnly

- Delete cookie by setting “expires” to date in past
- Default scope is domain and path of setting URL
- Client can also set cookies (Javascript)
Cookie scope rules (domain and path)

• Say we are at **www.wisc.edu**
  – Any non-TLD suffix can be scope:
    • allowed: **www.wisc.edu** or wisc.edu
    • disallowed: www2.wisc.edu or ucsd.edu

• Path can be set to anything
Cookies: reading by server

- Browser sends all cookies such that
  - domain scope is suffix of url-domain
  - path is prefix of url-path
  - protocol is HTTPS if cookie marked “secure”
Cookie security issues?

• Cookies have no integrity
  – HTTPS cookies can be overwritten by HTTP cookie (network injection)
  – Malicious clients can modify cookies
    • Shopping cart vulnerabilities

• Scoping rules can be abused
  – blog.example.com can read/set cookies for example.com

• Privacy
  – Cookies can be used to track you around the Internet

• HTTP cookies sent in clear
  – Session hijacking
PC Batman: Arkham City This Month [10:40 am ET] - Share - 2 Comments
WBIE announces the release dates for the delayed Windows PC edition of Batman: Arkham City, the stealth/action sequel:

Warner Bros. Interactive Entertainment and DC Entertainment today confirmed that the Games for Windows PC version of Batman: Arkham City will be available in North America beginning November 22, in Australia beginning November 23, in France and Benelux beginning November 24, and in other European territories beginning November 25.

CptnHarlock writes

"Today the registered users of Answers.com received an email informing them that the site has ended support for Yahoo, Twitter, Google, or LinkedIn as a way to sign into their site. Facebook is the sole external way left to log in. A local login and password were generated and sent by email and the old (non-Facebook) logins deactivated. Score another one for Facebook.com in the login consolidation wars."

Read the 14 comments
In addition to ads based on interest categories, Google allows advertisers (including Google) to show you ads based on your previous interactions online, such as visits to advertisers’ websites. For example, someone who visited the website of an online sporting goods store can receive ads about special offers from that store.
--- http://www.google.com/privacy/ads/
Session handling and login

Protocol is HTTPS. Elsewhere just HTTP

GET /index.html
Set-Cookie: AnonSessID=134fds1431

POST /login.html?name=bob&pw=12345
  Cookie: AnonSessID=134fds1431

  Set-Cookie: SessID=83431Adf

GET /account.html
Cookie: SessID=83431Adf
Session Hijacking

From http://codebutler.com/firesheep
Towards preventing hijacking

- Use encryption when setting session cookies
- SessID = Enc(K,info) where:
  - K is server-side secret key
  - Enc is Encrypt-then-MAC encryption scheme
  - info contains:
    - user id
    - expiration time
    - other data
- Server should record if user logs out
- Does this prevent Firesheep hijacking?
  - No
  - include in data machine-specific information
  - turn on HTTPS always