Announcements

• Project 4 is Due Friday  May 2nd  at 11:59 PM

• Final exam:
  – Friday, May 12th. Noon - 2:00pm   DRLB A6

• Today:
  – Cookies & State
  – Phishing

• Next (and last!) class:
  – Course Review
  – Course evaluations (please come!)
Maintaining State

• HTTP is a stateless protocol
  – Server doesn't store any information about the connections it handles (each request is treated independently)
  – Makes it hard to maintain session information

• Encode state in the URL:
  – …/cgi-bin/nxt?state=-189534fjk
  – Used commonly on message boards, etc. to track thread

• Use HIDDEN input fields
  – When user fills in web forms, the POST request gives server the data
  – You can embed state in invisible "input" fields

• Cookies
  – Store data on the client's machine
Hidden Fields

<html>
<head>  <title>My Page</title>  </head>
<body>
<form name="myform"
     action="http://.../handle.cgi"
     method="POST"> 
  <div align="center">
    <input type="text" size="25" value="Name?">
    <input type="hidden" name="Language" value="English">
  </div>
</form>
</body>
</html>
Cookies (Client-side state)

- Server can store cookies on the client machine by issuing:

  \[
  \text{Set-Cookie: NAME}=\text{VALUE}; \text{[expires=DATE;]} \\
  \text{[path=PATH;]} \text{[domain=DOMAIN\_NAME;]} \text{[secure]}
  \]

- Domain and Path restrict the servers (and paths on those servers) to which the cookie will be sent
- The "secure" flag says that the cookie should only be sent over HTTPS
- Uses:
  - User authentication
  - Personalization
  - User tracking: e.g. Doubleclick (3rd party cookies)
Cookies (cont'd)

• When the client requests a URL from a server, the browser matches the URL against all cookies on the client.

• If they match, then the client request includes the line:
  
  Cookie: NAME1=VALUE1; NAME2=VALUE2;...

• Notes:
  – New instances of cookies overwrite old ones
  – Clients aren't required to purge expired cookies (though they shouldn't send them)
  – Cookies can be at most 4k, at most 20 per site
  – To delete a cookie, the server can send a cookie with expires set to a past date
  – HTTP proxy servers shouldn't cache Set-cookie headers…
Cookies

Http is stateless protocol; cookies add state
• Used to store state on user’s machine

HTTP Header:
Set-cookie: NAME=VALUE ;
domain = (who can read) ;
expires = (when expires) ;
secure = (only over SSL)

If expires=NULL:
this session only

GET ...

Browser

Server

GET ...

Browser

Server

Cookie: NAME = VALUE
Cookie/Hidden Field Risks

• Danger of storing data on browser:
  – User can change values

• Silly example: Shopping cart software.
  
  Set-cookie: shopping-cart-total = 150 ($)

  – User edits cookie file (cookie poisoning):
    Cookie: shopping-cart-total = 15 ($)

  – ... bargain shopping.

• Similar behavior with hidden fields:
  
  <INPUT TYPE="hidden" NAME=price VALUE="150">
Example: dansie.net shopping cart

- http://www.dansie.net/demo.html  (April, 2008)

```
<FORM METHOD=POST ACTION="http://www.dansie.net/cgi-bin/scripts/cart.pl">

<FONT FACE="Times New Roman" COLOR="#000099" SIZE=+1>Black Leather purse with leather straps<br>Price: $20.00</FONT>

<INPUT TYPE=HIDDEN NAME=name VALUE="Black leather purse">
<INPUT TYPE=HIDDEN NAME=price VALUE="20.00">
<INPUT TYPE=HIDDEN NAME=sh VALUE="1">
<INPUT TYPE=HIDDEN NAME=img VALUE="purse.jpg">
<INPUT TYPE=HIDDEN NAME=img2 VALUE="purse_large.jpg">
<INPUT TYPE=HIDDEN NAME=return VALUE="http://www.dansie.net/demo.html">
<INPUT TYPE=HIDDEN NAME=custom1 VALUE="Black Leather purse with leather straps">

<INPUT TYPE=SUBMIT NAME="add" VALUE="Put in Shopping Cart">
</FORM>
```
Solution

- When storing state on browser use a Message Authentication Code (MAC) with server's secret key to enforce data integrity.

- .NET 2.0 (probably similar in 3.0):
    - Secret web server key intended for cookie protection
  
  - HttpCookie cookie = new HttpCookie(name, val);
  - HttpCookie encodedCookie = HttpSecureCookie.Encode (cookie);

  - HttpSecureCookie.Decode (cookie);
Cookie authentication (over https)

Browser

POST login.cgi
Username & pwd

Web Server

Set-cookie: auth=val

Auth server

Validate user

auth=val

Store val

GET restricted.html
Cookie: auth=val

If YES, restricted.html

Check val

restricted.html
auth=val

YES/NO
Cookie auth is insufficient

- **Example:**
  - User logs in to bank.com. Forgets to sign off.
  - Session cookie remains in browser state
    - Then user visits another site containing:
      ```html
      <form name=F action=http://bank.com/BillPay.php>
        <input name=recipient value=badguy> ...
      <script> document.F.submit(); </script>
      ```
    - Browser sends user auth cookie with request
      - Transaction will be fulfilled

- **Problem:**
  - Cookie auth is insufficient when side effects can happen
  - Correct use: use cookies + hidden fields
  - Hidden fields: store nonces that must be presented to the server
    - Can't be guessed by the malicious web site
Managing cookie policy via proxy

- Proxy intercepts request and response
- May modify cookies before sending to Browser
- Can do other checks: filter ads, block sites, etc.
- This is just a reference monitor for cookies
Sample Proxy:

• Cookie management by policy in *cookiefile*
  – Default: all cookies are silently crunched
  – Options
    • Allow cookies only to/from certain sites
    • Block cookies to browser (but allow to server)
    • Send vanilla wafers instead

• Block URLs matching any pattern in *blockfile*
  – Example: pattern /*.*/ad matches
    http://nomatterwhere.com/images/advert/g3487.gif

Easy to write your own http proxy; you can try *this* at home
Phishing

• Phishing:
  – Trojan horse e-mails and web sites designed to trick the user into giving up account/pin/password/credit card information.

• December 17, 2007: Gartner Survey
  – Estimated $3.2 BILLION was lost to phishing attacks
  – 3.3% of those surveyed lost money due to phishing
  – (more than in prior years)
  – Most spoofed: PayPal and eBay
  – See:
    www.doshelp.com/scams-fraud/Services/Ebay-Scams.htm

• Goal: Present a plausible experience to the user
Phishing Techniques

- See "Technical Trends in Phishing Attacks"
  - by Jason Milletary (US-CERT)
- Social Engineering
- Bot nets
  - Same infrastructure as Spam mail
- Web site hosting
  - Redirects / obfuscated URLs etc.
- Phishing Kits
  - Pre-generated HTML/e-mail that looks official (graphics, etc.)
- Browser vulnerabilities
  - Borderless popup windows that don't display the address bar
  - Cross-domain vulnerabilities
- XSS using URL redirectors that don't sanitize inputs
Reading browser history

• CSS properties of hyperlinks
• Can also use cache-based techniques:
  – Images and other data in the cache take less time to load, so a script can time how long it takes to load a resource to get some hints about a user's prior browsing.

Violation of the same-origin principle:

“One site cannot use information belonging to another site.”
Visited link tracking

- Visited links displayed in different color (74% of sites)
  - Information easily accessible by javascript
- Attacks also without javascript

```html
<html><head>
<style> a { position:absolute; border:0; } a:link { display:none } </style>
</head><body>
<a href='http://www.bankofamerica.com/'><img src='bankofamerica.gif'></a>
<a href='https://www.wellsfargo.com/'><img src='wellsfargo.gif'></a>
<a href='http://www.usbank.com/'><img src='usbank.gif'></a>
...
</body></html>
```

- Bank logo images are stacked on top of each other
- CSS rules cause the un-visited links to vanish
- Page displays bank logo of site that user has visited
Countermeasures?

• Education and awareness training
• "Security indicators" in the web browser
  – E.g. the yellow address background for https connections in FireFox
• Browser extensions that act as a firewall
  – Can blacklist known phishing sites
• Internet lists of known phishing sites:
  – www.phishtank.com
Do they work?

• Paper: "The Emperor's New Security Indicators: An evaluation of website authentication and the effect of role playing on usability studies" (Schechter et al. 2007)
  – Available on the course web pages

• Will customers of an online bank…
  – enter their passwords even if their browsers' HTTPS indicators are missing?
  – enter their passwords even if their site-authentication images are missing?
  – enter their passwords even if they are presented with an IE7 warning page?
Study

• 67 participants:
  – All had accounts at the same bank
  – Mostly Harvard students (not computer scientists/engineers)

• Divided into 3 groups:
  – Group 1: Played a "role" but not told that security was important
  – Group 2: Played a "role" but told that security was important
  – Group 3: Not role playing

• Participants were asked to complete several tasks
  – Check facts about their account balance, last login, last transaction, last statement

• Hints that someone was spoofing:
  – Remove HTTPS indicator
  – Remove site authentication images
  – Present a warning page
## Results

<table>
<thead>
<tr>
<th></th>
<th>Group 1 Role playing</th>
<th>Group 2 Role playing</th>
<th>Group 3 Per. Acct.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upon noticing HTTPS missing</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Image removed</td>
<td>0%</td>
<td>0%</td>
<td>9%</td>
</tr>
<tr>
<td>After Warning</td>
<td>47%</td>
<td>29%</td>
<td>55%</td>
</tr>
<tr>
<td>Never (Always logged in)</td>
<td>53%</td>
<td>71%</td>
<td>36%</td>
</tr>
</tbody>
</table>
Security Skins

• See the paper "The Battle Against Phishing: Dynamic Security Skins" by Dhamija and Tygar (2005)

• Use two techniques to prevent web page spoofing:
  – Trusted path for username/password entry
  – "Visual hash" to identify legitimate web servers

• Trusted path:
  – User picks a personal image
  – Dedicated username/password window uses that image as a background (overlapping the text entry fields too)
  – Hard to spoof
Secure Remote Password Protocol

• Due to Wu
• Setup (done once):
  – Alice picks a password PWD, random salt R, creates $V = \text{Hash}(\text{PWD}, R)$
  – Sends her userid, V and R to the server (Bart)
• Authentication and key generation:
  – Alice's client sends random number $N_A$ and userID to B
  – B sends Alice a random number $N_B$
  – Using $N_A$, $N_B$, and V, Alice and Bart compute a fresh key K
  – Alice sends $H_A = \text{Hash}(K, N_A, N_B)$ to B
  – B sends $H_B = \text{Hash}(H_A, K, N_A, N_B)$ to Alice
Security Skins

- Use the final hash value $H_B$ to generate a random image:
  - There are various techniques for doing this

- The client and server can reach the same image by doing the same calculation.
- Client marks the "secure" window with the image
- Server marks its web pages with the image too
- User does a "visual diff" to see that the images are the same
Example from the paper

• Client password window:

• Server-generated page has the same image embedded in the form backgrounds: