Announcements

• Project 4 is available on the web:
  – Due Friday April 20th at 11:59 PM

• Some of today's slides adopted from Dan Boneh and John Mitchell's courses at Stanford
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:

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

• 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=STRING1; NAME2=STRING2;...

• 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
Cookie 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">
Not so silly …  (as of 2/2000)

- D3.COM Pty Ltd: ShopFactory 5.8
- @Retail Corporation: @Retail
- AdgrafiX: Check It Out
- Baron Consulting Group: WebSite Tool
- ComCity Corporation: SalesCart
- Crested Butte Software: EasyCart
- Dansie.net: Dansie Shopping Cart
- Intelligent Vending Systems: Intellivend
- Make-a-Store: Make-a-Store OrderPage
- McMurtrey/Whitaker & Associates: Cart32 3.0
- pknutsen@nethut.no: CartMan 1.04
- Rich Media Technologies: JustAddCommerce 5.0
- SmartCart: SmartCart
- Web Express: Shoptron 1.2

- Source: http://xforce.iss.net/xforce/xfdb/4621
Example: dansie.net shopping cart

• http://www.dansie.net/demo.html  (April, 2007)

<FORM METHOD=POST
       ACTION="http://www.dansie.net/cgi-bin/scripts/cart.pl">
   Black Leather purse with leather straps<br>Price: $20.00<br>
   <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=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>

• CVE-2000-0253  (Jan. 2001),  BugTraq ID: 1115
Solution

• When storing state on browser MAC data using server secret key.

• .NET 2.0:
    • Secret web server key intended for cookie protection
  
    – `HttpCookie cookie = new HttpCookie(name, val);`  
      `HttpCookie encodedCookie = HttpSecureCookie.Encode(cookie);`
  
    – `HttpSecureCookie.Decode(cookie);`
Cookie authentication

- Browser
  - POST login.cgi
  - Username & pwd
  - Set-cookie: auth=val

- Web Server
  - Validate user
  - auth=val
  - Store val
  - restricted.html
  - Cookie: auth=val
  - If YES, restricted.html

- Auth server
  - Check val
  - YES/NO
Weak authenticators: security risk

- Predictable cookie authenticator
  - Verizon Wireless - counter
  - Valid user logs in, gets counter, can view sessions of other users.

- Weak authenticator generation:
  - WSJ.com: cookie = \{user, MAC_k(user) \}
  - Weak MAC exposes K from few cookies.

- Apache Tomcat: generateSessionID()
  - MD5(PRNG) ... but weak PRNG
  - Predictable SessionID's
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
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.
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
Fooling the user

Sends email: “There is a problem with your eBuy account”

Password sent to bad guy

User clicks on email link to www.ebuju.com.

User thinks it is ebuy.com, enters eBuy username and password.
Password Phishing Problem

- User cannot reliably identify fake sites
- Captured password can be used at target site
Common Password Problem

- Phishing attack or break-in at site B reveals pwd at A
  - Server-side solutions will not keep pwd safe
  - Solution: Strengthen with client-side support
Password Hashing

• Generate a unique password per site
  – HMAC_{fido:123}(banka.com) ⇒ Q7a+0ekEXb
  – HMAC_{fido:123}(siteb.com) ⇒ OzX2+ICiqc
• Hashed password is not usable at any other site
  – Protects against password phishing
  – Protects against common password problem

http://crypto.stanford.edu/PwdHash
The Spoofing Problem

- JavaScript can display password fields or dialogs:

- Unhashed password sent to attacker in clear
Password Prefix

- Original pwd should never be visible to web page
Password Prefix: How it works

• **Normal operation:** Prefix in password field
  
  @@fido:123 ⇒ @@abcdefgh ⇒ **********
  
  abcdedefgh ⇒ fido:123
  
  HMAC_{fido:123}(siteb.com) ⇒ Q7a+0ekEXb

• **Abnormal operation:** Prefix in non-password field
  
  – Can just ignore the prefix and not hash
  
  – Remind user not to enter password
The Perfect Phishing Email

Fooling the user using browser state

• Bank of America customers see:
  – “Click here to see your Bank of America statement”
• Wells Fargo customers see:
  – “Click here to see your Wells Fargo statement”
• Works in Outlook; behavior is by design
Reading browser history

- CSS properties of hyperlinks
- Can also use cache-based techniques

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
Preserving web privacy

• Your IP address may be visible to web sites
  – This may reveal your employer, ISP, etc.
  – Can link activities on different sites, different times

• Can you prevent sites from learning about you?
  – Anonymizer
    • Single site that hides origin of web request
  – Crowds
    • Distributed solution
Anonymity?

- **Sender anonymity:**
  - The identity of the sender is hidden, while the receiver (and message) might not be

- **Receiver anonymity:**
  - The identity of the receiver is hidden (message and sender might not be)

- **Unlikability of sender and receiver:**
  - Although the sender and receiver can be identified as participating in communication, they cannot be identified as communicating with each other.
Browsing Anonymizers

- Anonymizer.com
- Web Anonymizer hides your IP address

- What does anonymizer.com know about you?

www.anonymizer.com/cgi-bin/redirect.cgi?url=...
Related approach to anonymity

• Hide source of messages by routing them randomly
• Routers don’t know for sure if the apparent source of the message is the actual sender or simply another router
  – Only secure against local attackers!
• Existing systems: Freenet, Crowds, etc.
Crowds

Sender randomly chooses a path through the crowd
Some routers are honest, some corrupt
After receiving a message, honest router flips a coin
- With probability $P_f$ routes to the next member on the path
- With probability $1 - P_f$ sends directly to the recipient

http://avirubin.com/crowds.pdf

[Reiter, Rubin ‘98]
What Does Anonymity Mean?

• Degree of anonymity:
  – Ranges from absolute privacy to provably exposed

• Beyond suspicion
  – The observed source of the message is no more likely to be the actual sender than anybody else

• Probable innocence
  – Probability <50% that the observed source of the message is the actual sender

• Possible innocence
  – Non-trivial probability that the observed source of the message is not the actual sender

Guaranteed by Crowds if there are sufficiently few corrupt routers