

Computer Security

CS 426

Lecture 27

Secure Web Site Design

(Most Slides taken from Prof. Dan Boneh CS 155 Slides
at Stanford)

Cross site request forgery

Cross site request forgery (abbrev. CSRF or XSRF)

- Also known as **one click attack** or **session riding**
- Transmits unauthorized commands from a user who has logged in to a website to the website.

- Example:

- User logs in to bank.com. Forgets to sign off.
- Session cookie remains in browser state
- Then user visits another site containing:

```
<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:

- browser is a confused deputy

Some Attack Methods

- HTML Methods

- IMG SRC**

- ``

- SCRIPT SRC**

- `<script src="http://host/?command">`

- IFRAME SRC**

- `<iframe src="http://host/?command">`

- JavaScript Methods

- 'Image' Object**

- `<script>`
`var foo = new Image();`
`foo.src = "http://host/?command";`
`</script>`

GMail Incidence: Jan 2007

- Google docs has a script that run a callback function, passing it your contact list as an object. The script presumably checks a cookie to ensure you are logged into a Google account before handing over the list.
- Unfortunately, it doesn't check what page is making the request. So, if you are logged in on window 1, window 2 (an evil site) can make the function call and get the contact list as an object. Since you are logged in somewhere, your cookie is valid and the request goes through.

Prevention

- Server side:
 - use cookie + hidden fields to authenticate
 - hidden fields values need to be unpredictable and user-specific
 - requires the body of the POST request to contain cookies
- User side:
 - logging off one site before using others

SQL Injection

See Slides by Prof. Venkat

Another Example of SQL Injection

- User input is used in SQL query
- Example: login page (ASP)

```
set ok = execute("SELECT * FROM UserTable  
WHERE username=' " & form("user") &  
" ' AND password=' " & form("pwd") & " ' " );  
If not ok.EOF  
    login success  
else fail;
```

- Is this exploitable?

SQL Injections could be used to Run commands on hosts as well

- Suppose user =

```
'exec cmdshell  
      'net user badguy badpwd' / ADD --
```
- Then script does:

```
ok = execute( SELECT ...  
              WHERE username= ' ' exec ... )
```

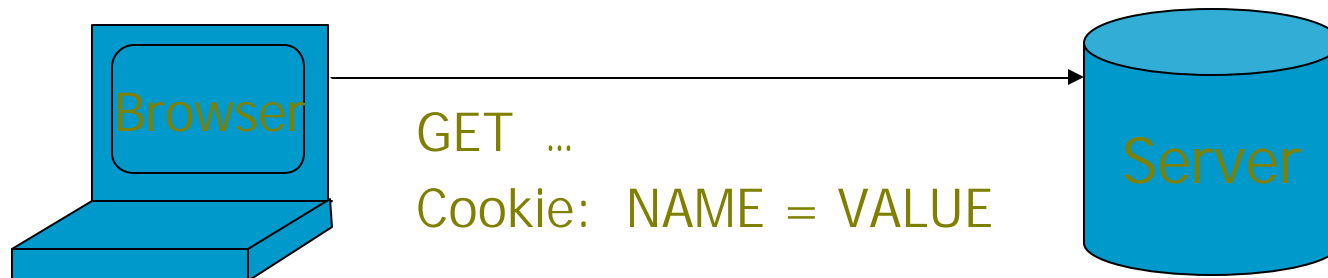
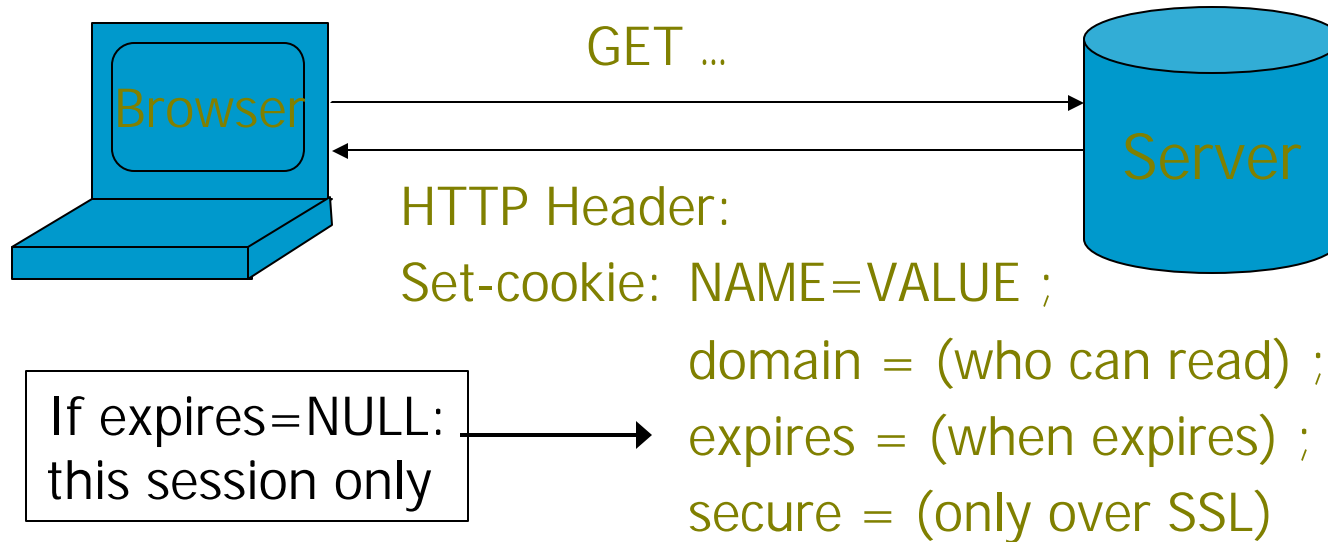
If SQL server context runs as "sa", attacker gets account on DB server.

Session Management

Cookies, hidden fields,
and user authentication

Cookies

- Used to store state on user's machine



Http is stateless protocol; cookies add state

Cookies

- Uses:
 - User authentication
 - Personalization
 - User tracking: e.g. Doubleclick (3rd party cookies)

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

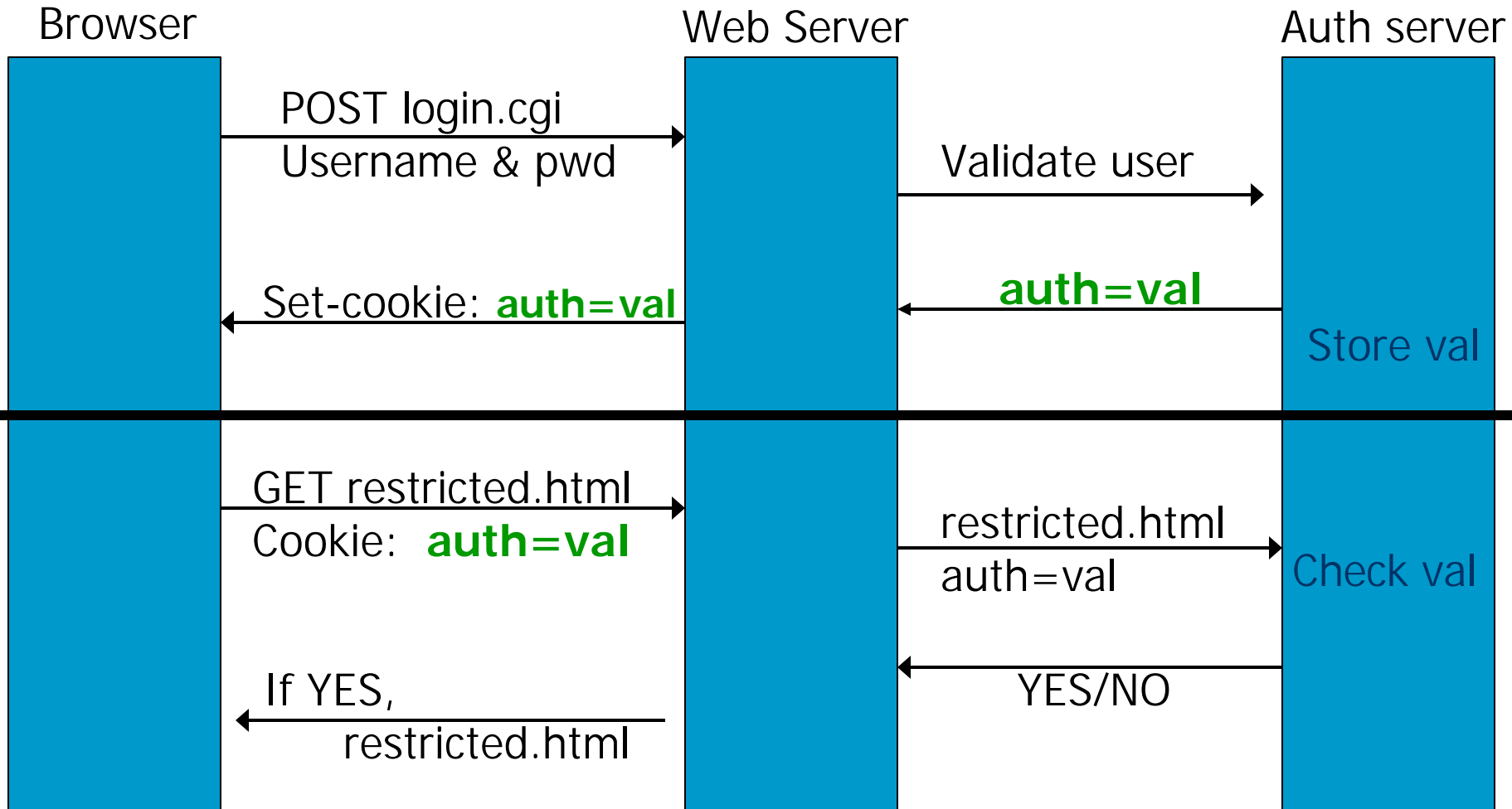
```
<FORM METHOD=POST
    ACTION="http://www.dansie.net/cgi-bin/scripts/cart.pl">
    Black Leather purse with leather straps<BR>Price: $20.00<BR>
    <INPUT TYPE=HIDDEN NAME=name      VALUE="Black leather purse">
    <INPUT TYPE=HIDDEN NAME=price     VALUE="20.00">
    <INPUT TYPE=HIDDEN NAME=sh       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
- <http://www.dansie.net/demo.html> (May, 2006)

Solution

- When storing state on browser MAC data using server secret key.
- .NET 2.0:
 - `System.Web.Configuration.MachineKey`
 - Secret web server key intended for cookie protection
 - `HttpCookie cookie = new HttpCookie(name, val);`
`HttpCookie encodedCookie =`
`HttpSecureCookie.Encode (cookie);`
 - `HttpSecureCookie.Decode (cookie);`

Cookie authentication



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: [Fu et al. '01]
 - WSJ.com: cookie = {user, $\text{MAC}_k(\text{user})$ }
 - Weak MAC exposes K from few cookies.
- Apache Tomcat: generateSessionID()
 - MD5(PRNG) ... but weak PRNG [GM'05].
 - Predictable SessionID's

Coming Attractions ...

- December 4:
 - DBMS Security

