CSE 484 / CSE M 584 (Autumn 2011)

Web Security (cont.)

Daniel Halperin
Tadayoshi Kohno

Thanks to Dan Boneh, Dieter Gollmann, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

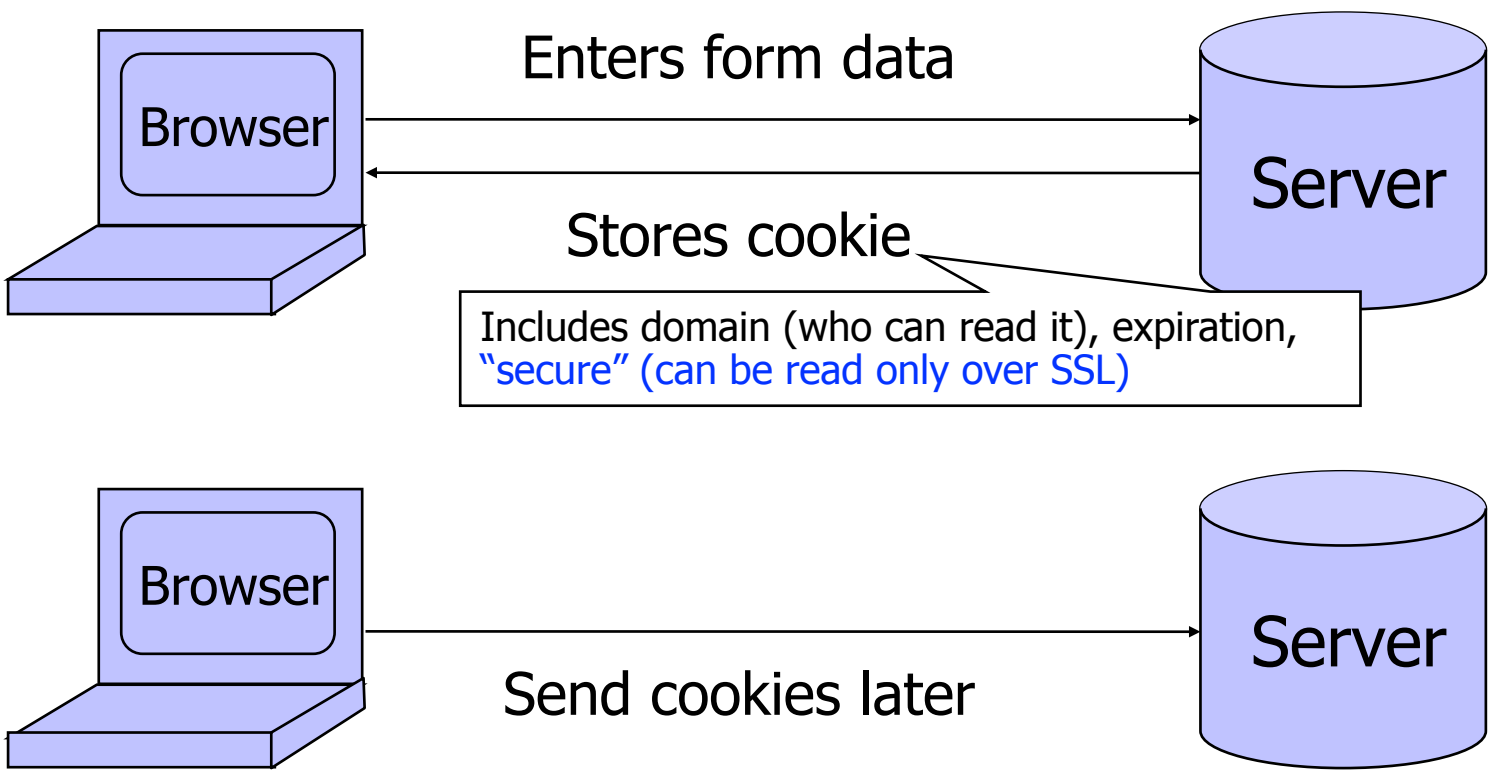
Today, 10/26

- Web Security
- Office hours after class in CSE 210
- Security Reviews & Current Event Reports
  - Groups of 1-3, 1 submission per group
  - First of each due: Friday, Nov. 4

Cookies

Storing Info Across Sessions

- A cookie is a data blob created by an Internet site to store information on your computer



HTTP is traditionally a stateless protocol; cookies add state

What Are Cookies Used For?

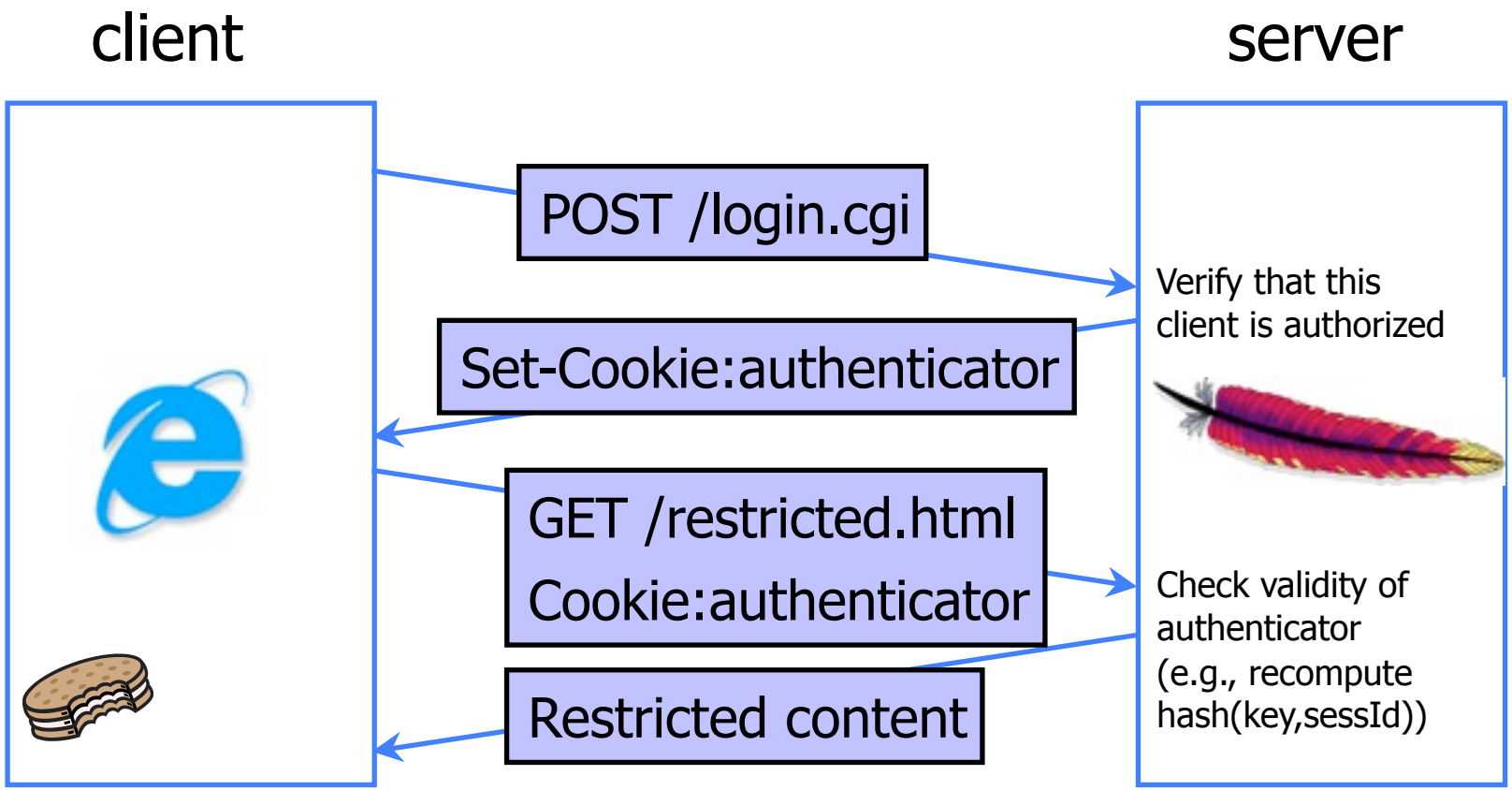
- Authentication
  - Use the fact that the user authenticated correctly in the past to make future authentication quicker
- Personalization
  - Recognize the user from a previous visit
- Tracking
  - Follow the user from site to site; learn his/her browsing behavior, preferences, and so on

Web Authentication via Cookies

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

- 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: MAC(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



Authenticators must be unforgeable and tamper-proof

(malicious client shouldn't be able to compute his own or modify an existing authenticator)

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
    - Set by sites embedded within other sites (e.g., ads)

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 (and **many more** since):

  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

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<BR>Pri

  <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>
```

Change this to 2.00

Shopping Cart Form Tampering

http://xforce.iss.net/xforce/xfdb/4621

- 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
  - @Retail Corporation: @Retail Any version
  - Adgrafix: Check It Out Any version
  - Baron Consulting Group: WebSite Tool Any version
  - ComCity Corporation: SalesCart Any version
  - Crested Butte Software: EasyCart Any version
  - Dansie.net: Dansie Shopping Cart Any version
  - Intelligent Vending Systems: Intellivend Any version
  - Make-a-Store: Make-a-Store OrderPage Any version
  - McMurtrey/Whitaker & Associates: Cart32 2.6
  - McMurtrey/Whitaker & Associates: Cart32 3.0
  - pknutsen@nethut.no: CartMan 1.04
  - Rich Media Technologies: JustAddCommerce 5.0
  - SmartCart: SmartCart Any version
  - Web Express: Shoptron 1.2

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; MAC(ServerKey, 299.99)

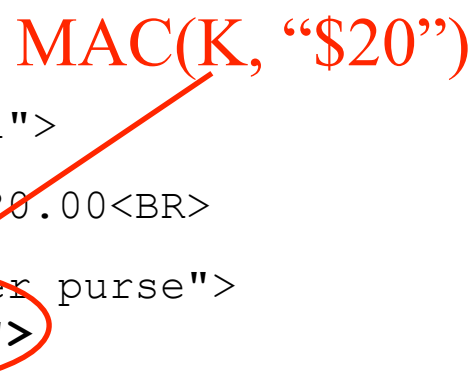
Storing State in Browser

- 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=pricemac VALUE="F13A3....B2">
  <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<BR>Price: $20.00<BR>

  <INPUT TYPE=HIDDEN NAME=name     VALUE="Black leather purse">
  <INPUT TYPE=HIDDEN NAME=pricemac VALUE="F13A....B2">
  <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<BR>Price: $20.00<BR>

  <INPUT TYPE=HIDDEN NAME=name     VALUE="Black leather purse">
  <INPUT TYPE=HIDDEN NAME=pricemac VALUE="F13A3...B2">
  <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>
```

MAC(K, "$20")

A319F...3C

MAC(K, "$2")

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=pricemac VALUE="F13A3...B2">
  <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>
```

MAC(K, "$20")

A319F...3C

MAC(K, "$2")

Better: MAC(K, "$20,Black leather purse, product number 12345, ...")

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.
  - || is string concatenation
- 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

Attack

username	crypt(username,key,"00")	authenticator cookie
AliceBob1	008H8LRfzUXvk	AliceBob1008H8LRfzUXvk
AliceBob2	008H8LRfzUXvk	AliceBob2008H8LRfzUXvk

Attack

username	crypt(username,key,"00")	authenticator cookie
AliceBob1	008H8LRfzUXvk	AliceBob1008H8LRfzUXvk
AliceBob2	008H8LRfzUXvk	AliceBob2008H8LRfzUXvk

"Create" an account with a 7-letter user name…

Attack

username	crypt(username,key,"00")	authenticator cookie
AliceBob1	008H8LRfzUXvk	AliceBob1008H8LRfzUXvk
AliceBob2	008H8LRfzUXvk	AliceBob2008H8LRfzUXvk

"Create" an account with a 7-letter user name...

AliceBoA	0073UYEre5rBQ	Try logging in: access refused

Attack

username	crypt(username,key,"00")	authenticator cookie
AliceBob1	008H8LRfzUXvk	AliceBob1008H8LRfzUXvk
AliceBob2	008H8LRfzUXvk	AliceBob2008H8LRfzUXvk

"Create" an account with a 7-letter user name...

AliceBoA	0073UYEre5rBQ	Try logging in: access refused
AliceBoB	00bkHcfOXBKno	Access refused

Attack

username	crypt(username,key,"00")	authenticator cookie
AliceBob1	008H8LRfzUXvk	AliceBob1008H8LRfzUXvk
AliceBob2	008H8LRfzUXvk	AliceBob2008H8LRfzUXvk

"Create" an account with a 7-letter user name...

AliceBoA	0073UYEre5rBQ	Try logging in: access refused
AliceBoB	00bkHcfOXBKno	Access refused
AliceBoC	00ofSJV6An1QE	Login successful! 1st key symbol is C

Attack

username	crypt(username,key,"00")	authenticator cookie
AliceBob1	008H8LRfzUXvk	AliceBob1008H8LRfzUXvk
AliceBob2	008H8LRfzUXvk	AliceBob2008H8LRfzUXvk

"Create" an account with a 7-letter user name…

AliceBoA	0073UYEre5rBQ	Try logging in: access refused
AliceBoB	00bkHcfOXBKno	Access refused
AliceBoC	00ofSJV6An1QE	Login successful! 1st key symbol is C

Now a 6-letter user name…

AliceBCA	001mBnBErXRuc	Access refused
AliceBCB	00T3JLLfuspdo	Access refused… and so on

Attack

username	crypt(username,key,"00")	authenticator cookie
AliceBob1	008H8LRfzUXvk	AliceBob1008H8LRfzUXvk
AliceBob2	008H8LRfzUXvk	AliceBob2008H8LRfzUXvk

"Create" an account with a 7-letter user name…

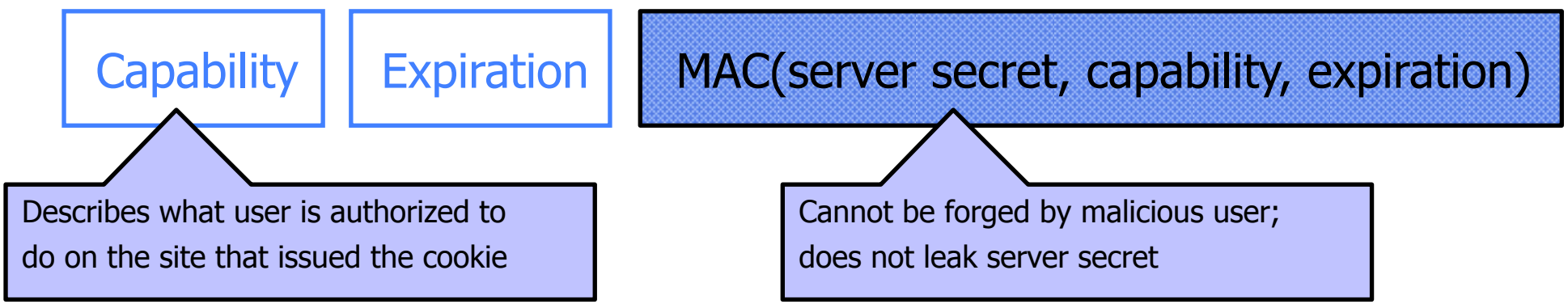
AliceBoA	0073UYEre5rBQ	Try logging in: access refused
AliceBoB	00bkHcfOXBKno	Access refused
AliceBoC	00ofSJV6An1QE	Login successful! 1st key symbol is C

Now a 6-letter user name…

AliceBCA	001mBnBErXRuc	Access refused
AliceBCB	00T3JLLfuspdo	Access refused… and so on

- Only need 128 x 8 queries instead of intended 128^8
- Minutes with a simple Perl script vs. billions of years

Better Cookie Authenticator



- Main lesson: be careful rolling your own
  - Homebrewed authentication schemes are easy to get wrong
- There are standard cookie-based schemes

Web Applications

- Online banking, shopping, government, 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 a potential concern.
  - Poorly written scripts
  - Sensitive data stored in world-readable files

General issue: 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 *.dat");

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-site 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

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 server, protocol, and port
  - 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="Security is Happiness"

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

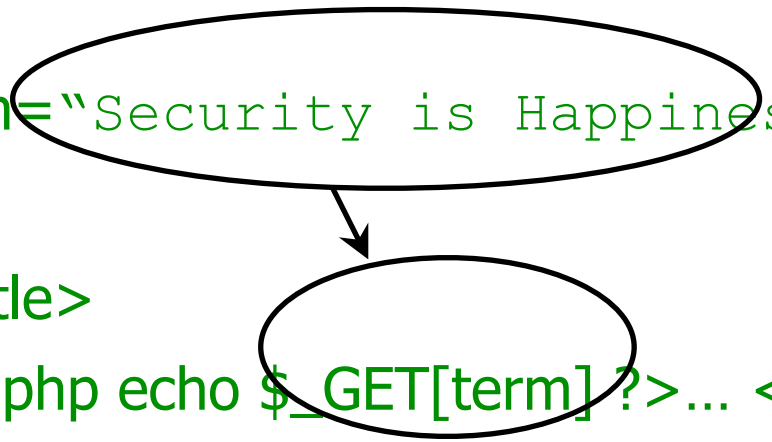
<html>Welcome, dear Bob</html>

Risks of Poorly Written Scripts

- For example, echo user's input

http://naive.com/search.php?term="Security is Happiness"

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>
```