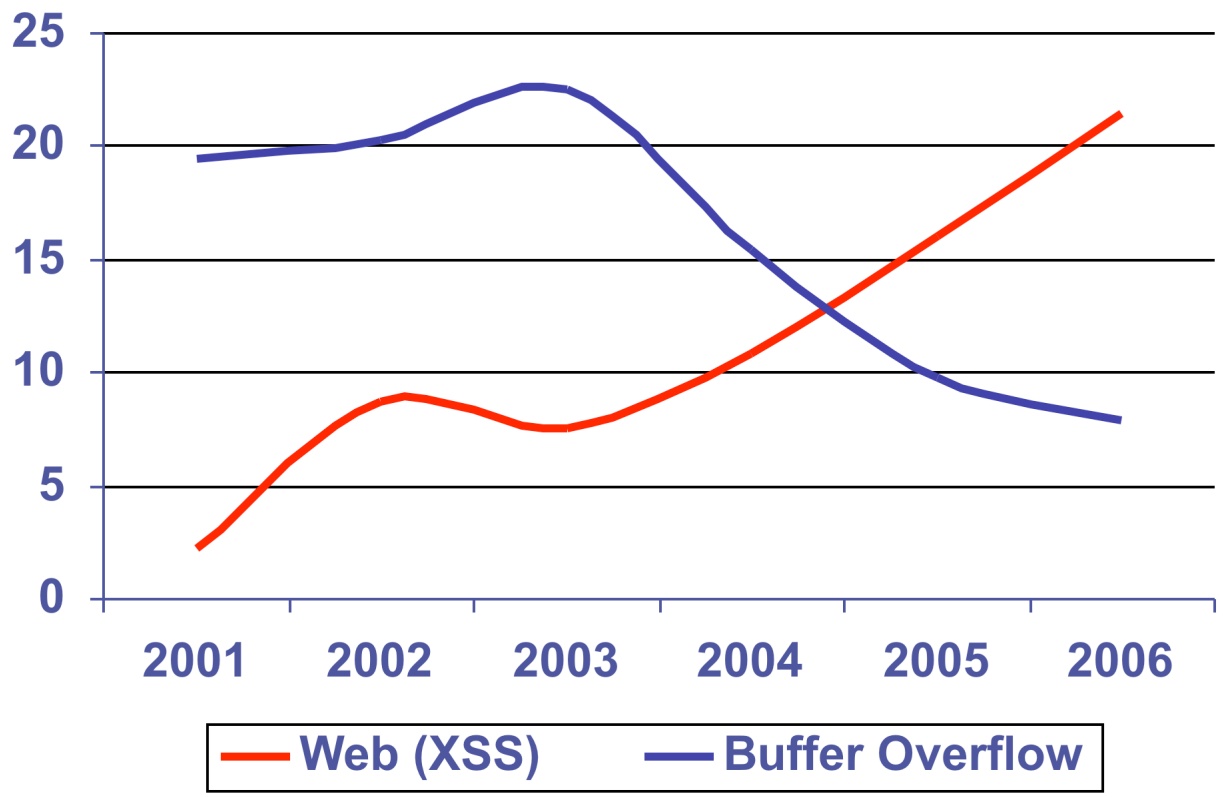
Basic web security model

Elie Bursztein CS155

Vulnerability Stats: web is "winning"

- Majority of vulnerabilities now found in web software



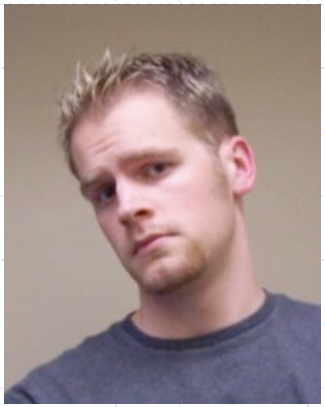
Source: MITRE CVE trends

Web security: two sides

- Web browser: (client side)
  - Attacks target browser security weaknesses
  - Result in:
    - Malware installation (keyloggers, bot-nets)
    - Document theft from corporate network
    - Loss of private data

- Web application code: (server side)
  - Runs at web site: banks, e-merchants, blogs
  - Written in PHP, ASP, JSP, Ruby, …
  - Many potential bugs: XSS, XSRF, SQL injection
  - Attacks lead to stolen CC#, defaced sites.

Credits



Adam Barth, Collin Jackson, John Mitchell, Dan Boneh and the entire websec team

Stanford Web Security Research - Windows Internet Explorer

http://crypto.stanford.edu/websec/

Stanford Web Security Research

Stanford Web Security Research

Overview

The Web Security Group is a part of the Stanford Security Laboratory. Research projects focus on various aspects of browser and web application security.

Publications

Securing the Web Platform

Beware of Finer-Grained Origins
Collin Jackson and Adam Barth
In *Web 2.0 Security and Privacy*. (W2SP 2008)

ForceHTTPS Cookies: A Defense Against Eavesdropping and Pharming
Collin Jackson and Adam Barth
In *Proc. of the 17th International World Wide Web Conference*. (WWW 2008)

Protecting Browsers from DNS Rebinding Attacks
Collin Jackson, Adam Barth, Andrew Bortz, Weidong Shao, and Dan Boneh
In *Proc. of the 14th ACM Conf. on Computer and Communications Security*. (CCS 2007)

Security for Mashups

Securing Browser Frame Communication
Adam Barth, Collin Jackson John C. Mitchell
In *Proc. of the 17th USENIX Security Symposium*. (USENIX Security '08)

Protection and Communication Abstractions for Web Browsers in MashupOS
Helen J. Wang, Xiaofeng Fan, Jon Howell, and Collin Jackson
In *Proc. of the 21st ACM Symposium on Operating Systems Principles* (SOSP 2007)

MashupOS: Operating System Abstractions for Client Mashups
Jon Howell, Collin Jackson, Helen J. Wang, and Xiaofeng Fan
In *Proc. of the 11th Workshop on Hot Topics in Operating Systems*. (HotOS 2007)

Subspace: Secure Cross-Domain Communication for Web Mashups
Collin Jackson and Helen J. Wang
In *Proc. of the 16th International World Wide Web Conference*. (WWW 2007)

Privacy in the Browser

Exposing Private Information by Timing Web Applications
Andrew Bortz, Dan Boneh, and Palash Nandy
In *Proc. of the 16th International World Wide Web Conference*. (WWW 2007)

Protecting Browser State from Web Privacy Attacks
Collin Jackson, Andrew Bortz, Dan Boneh, and John C. Mitchell
In *Proc. of the 15th International World Wide Web Conference*. (WWW 2006)

People
Adam Barth
Dan Boneh
Andrew Bortz
Collin Jackson
John C. Mitchell

Workshops
AdFraud Workshop (2007)
3rd TIPPI Workshop (2007)
UPU Workshop (2006)
2nd TIPPI Workshop (2006)
1st TIPPI Workshop (2005)

Demonstrations
Anti-Phishing Browser Extensions
JSONRequest Browser Extension
HTML 5 postMessage API
Frame Hijacking
Mixed Content
SSL Man-in-the-Middle Proxy
Randomized Hashing for Digital Certificates: Halevi-Krawczyk Hash
Defeating Frame Busting Techniques

Advisories
CVE-2007-6244
CVE-2007-5858
CVE-2007-5275
CVE-2008-1003
CVE-2008-1004
CVE-2008-1006
CVE-2008-1007

Open Source Outreach

WebKit	Mozilla
Bug 17249	Bug 408052
Bug 16837	Bug 403591
Bug 16824	Bug 403590
Bug 16775	Bug 403582
Bug 16539	Bug 401317
Bug 16522	Bug 389625
Bug 16523	Bug 387706
Bug 16313	Bug 418354
Bug 16011	Bug 418355

Internet 100%

http://crypto.stanford.edu/websec

Outline

- Web Refresher:
- Security User Interface
  - Goals of a browser
  - When is it safe to type my password?
- Same-Origin Policy
  - How sites are isolated
  - Opting out of isolation
  - Frame hijacking
  - Navigation policy
- Cookie security
- Browser security design

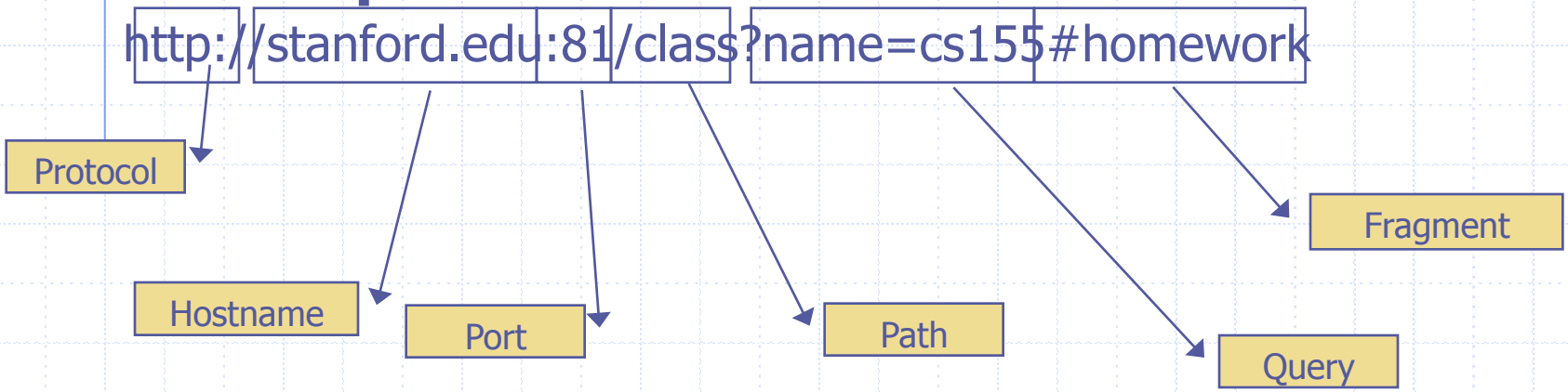
Web Refresher

HTTP protocol

- HTTP is
  - widely used
  - Simple
  - Stateless
  - Unencrypted

URLs

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



- 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>
```

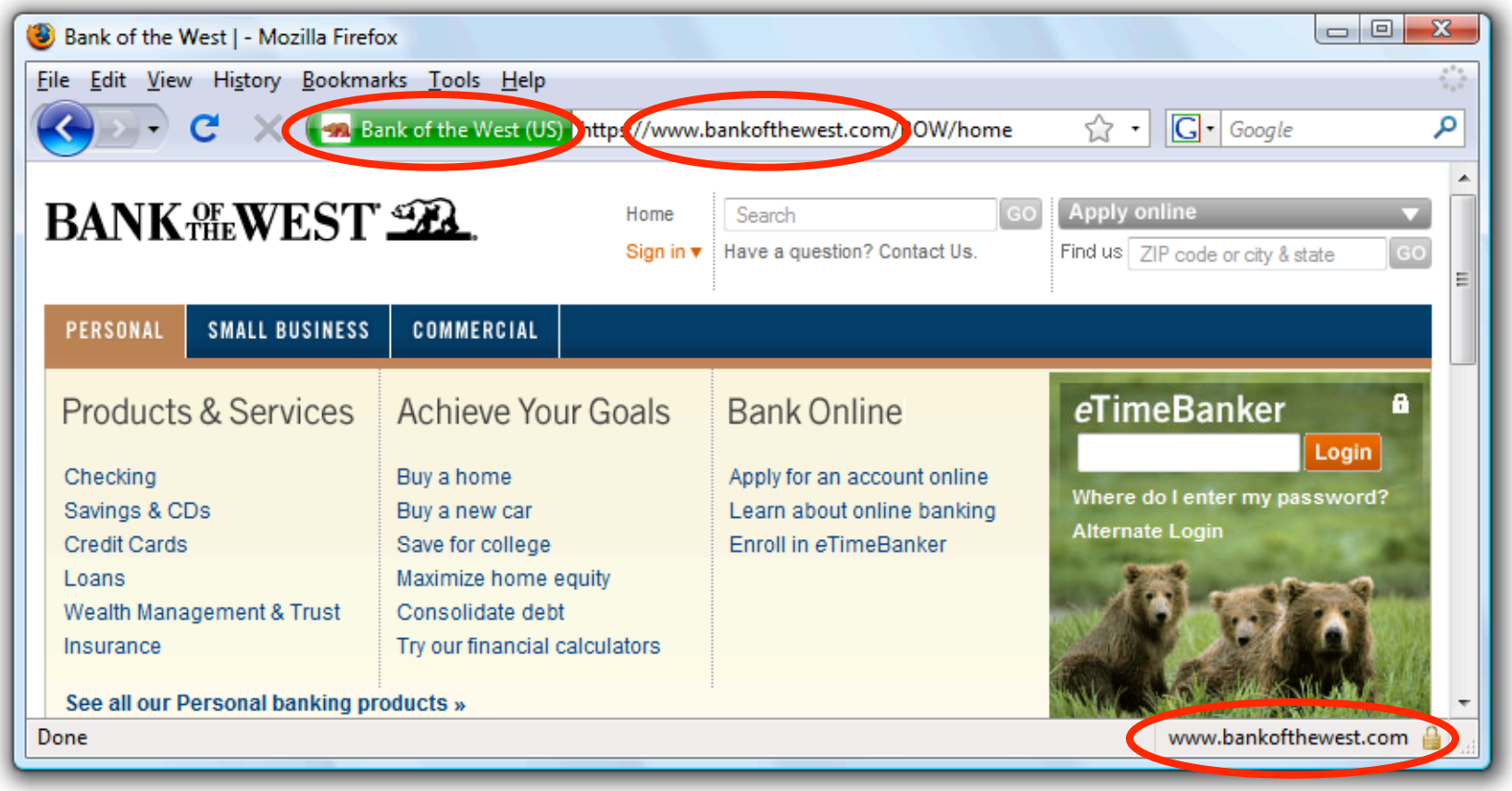
Data

Cookies

Security User Interface

When is it safe to type my password?

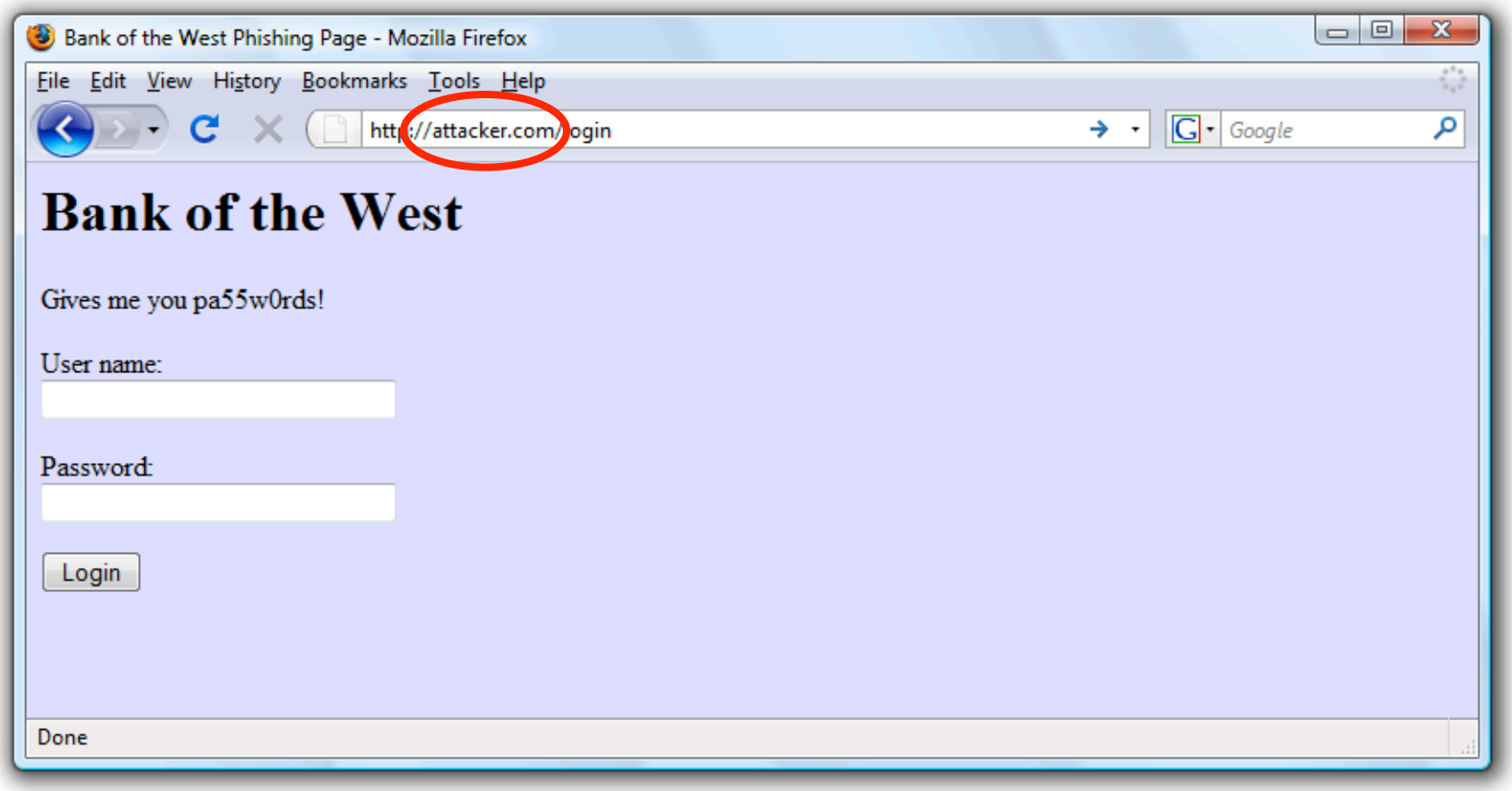
Safe to type your password?



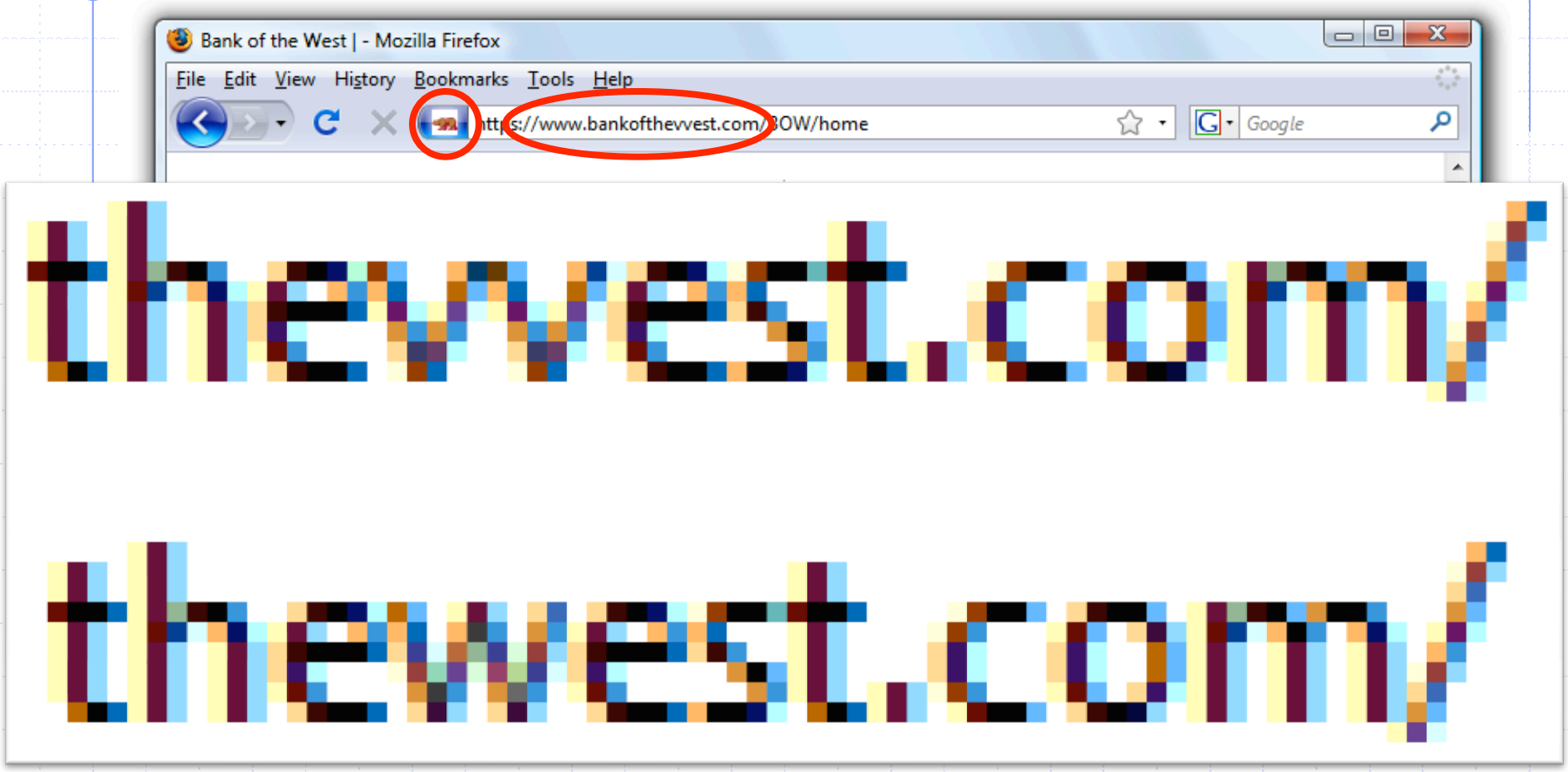
Outline

- Web Refresher:
- Security User Interface
  - Goals of a browser
  - When is it safe to type my password?
- Same-Origin Policy
  - How sites are isolated
  - Opting out of isolation
  - Frame hijacking
  - Navigation policy
- Cookie security
- Browser security design

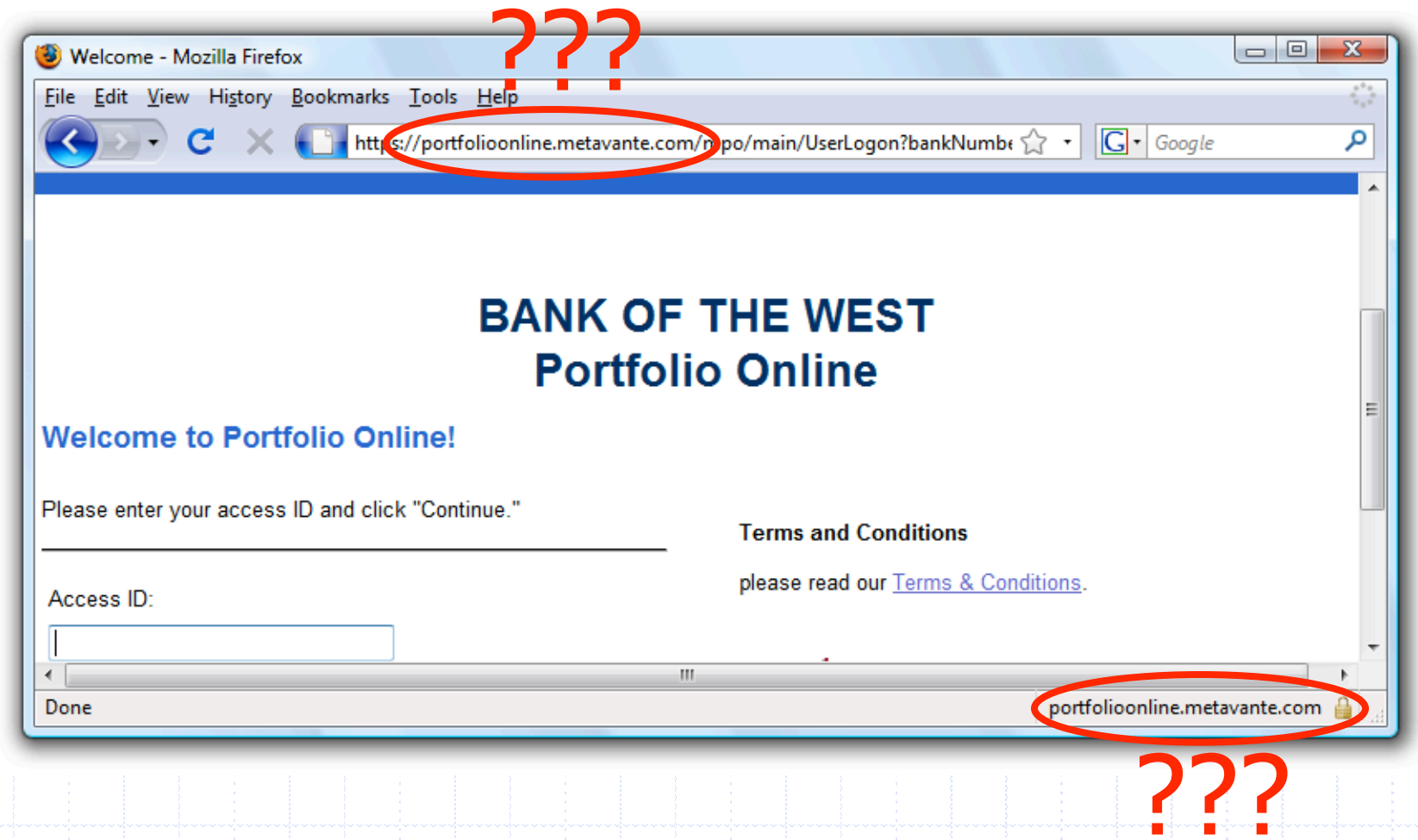
Safe to type your password?



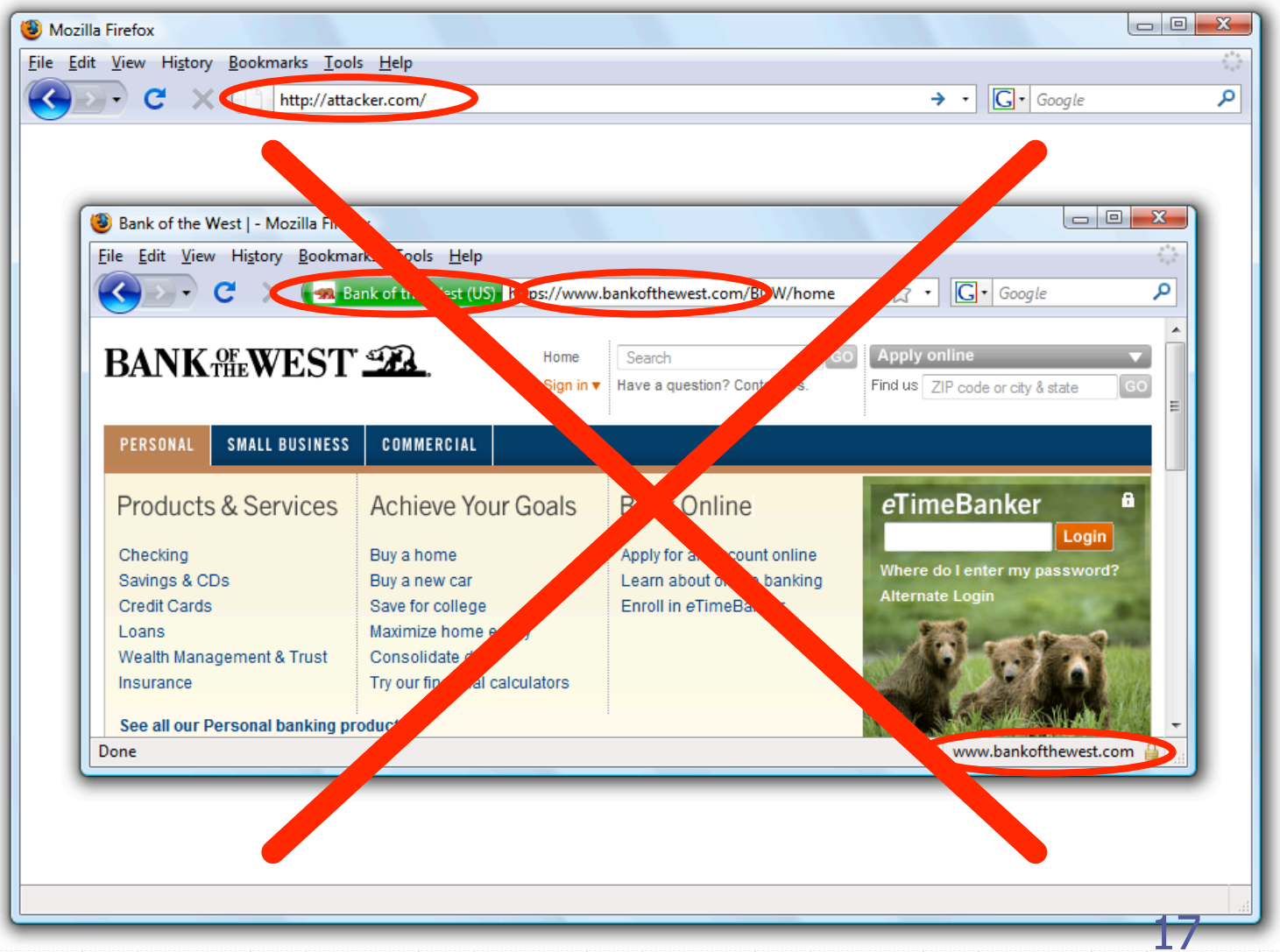
Safe to type your password?



Safe to type your password?



Safe to type your password?



Same-Origin Policy

How does the browser isolate different sites?

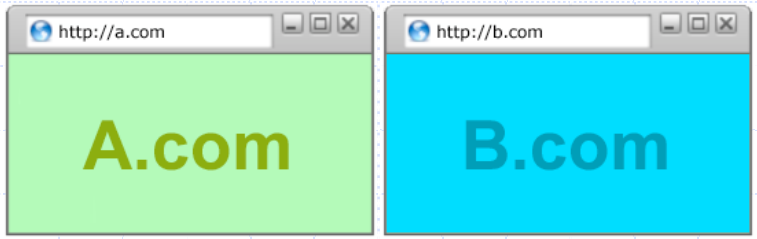
Outline

- Web Refresher:
- Security User Interface
  - Goals of a browser
  - When is it safe to type my password?
- Same-Origin Policy
  - How sites are isolated
  - Opting out of isolation
  - Frame hijacking
  - Navigation policy
- Cookie security
- Browser security design

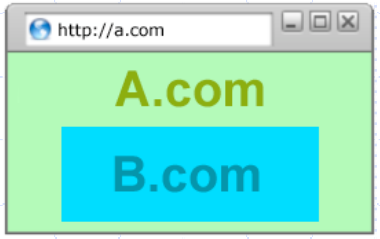
Policy Goals

- Safe to visit an evil web site

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



- Allow safe delegation



Components of browser security policy

- Frame to Frame relationships
  - canScript(A,B)
    - Can Frame A execute a script that reads or writes DOM elements of Frame B?
  - canNavigate(A,B)
    - Can Frame A change the origin of content for Frame B?
- Frame to cookie relationships
  - readCookie(A,S), writeCookie(A,S)
    - Can Frame A read/write cookies from origin S?
- SecurityIndicator (W) [ssl lock icon]
  - Is the security indicator displayed for window W?

Popup windows

- With hyperlinks

  click here

- With JavaScript

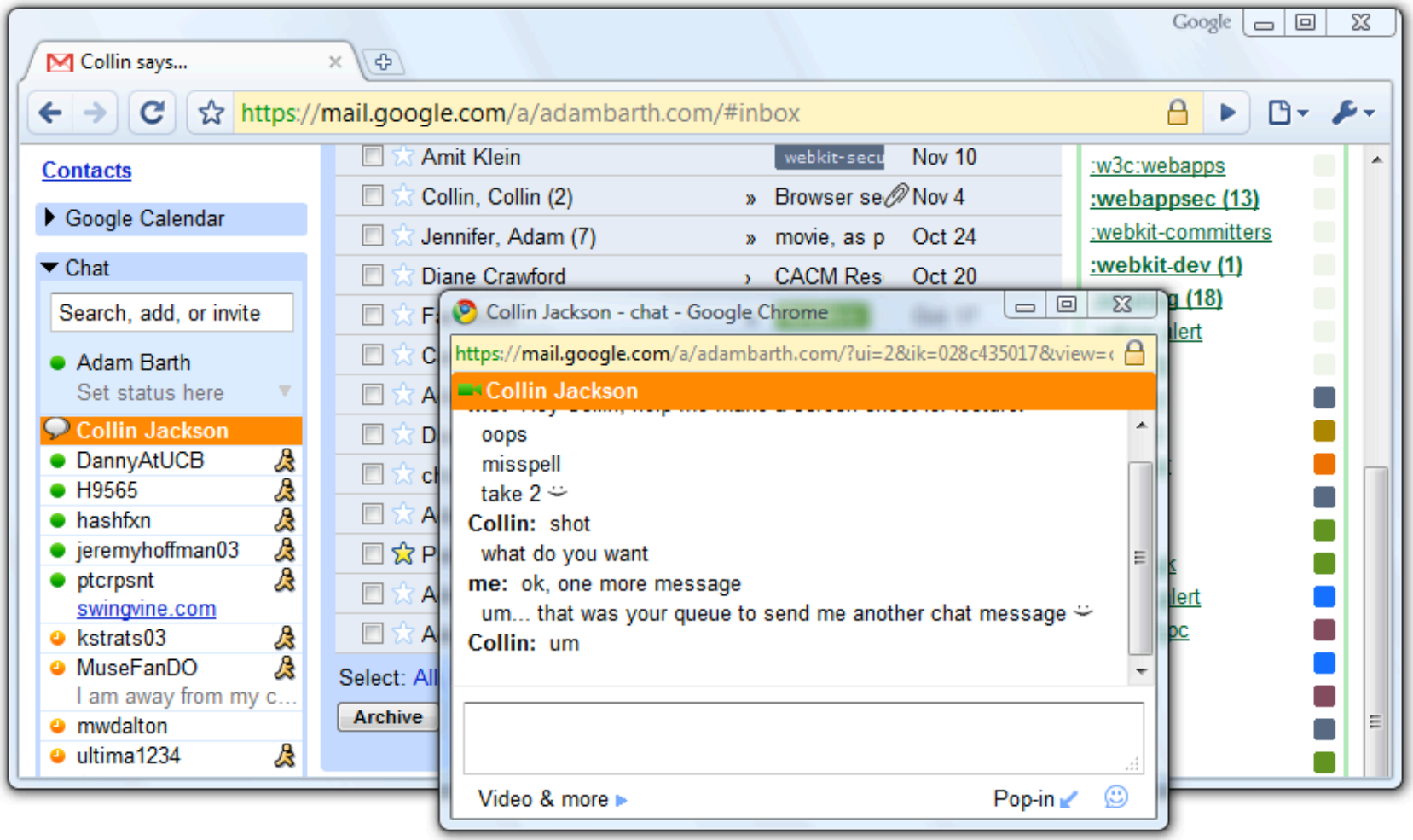
  mywin = window.open("http://www.b.com", "foo", "width=10,height=10")

  - Navigating named window re-uses existing one
  - Can access properties of remote window:

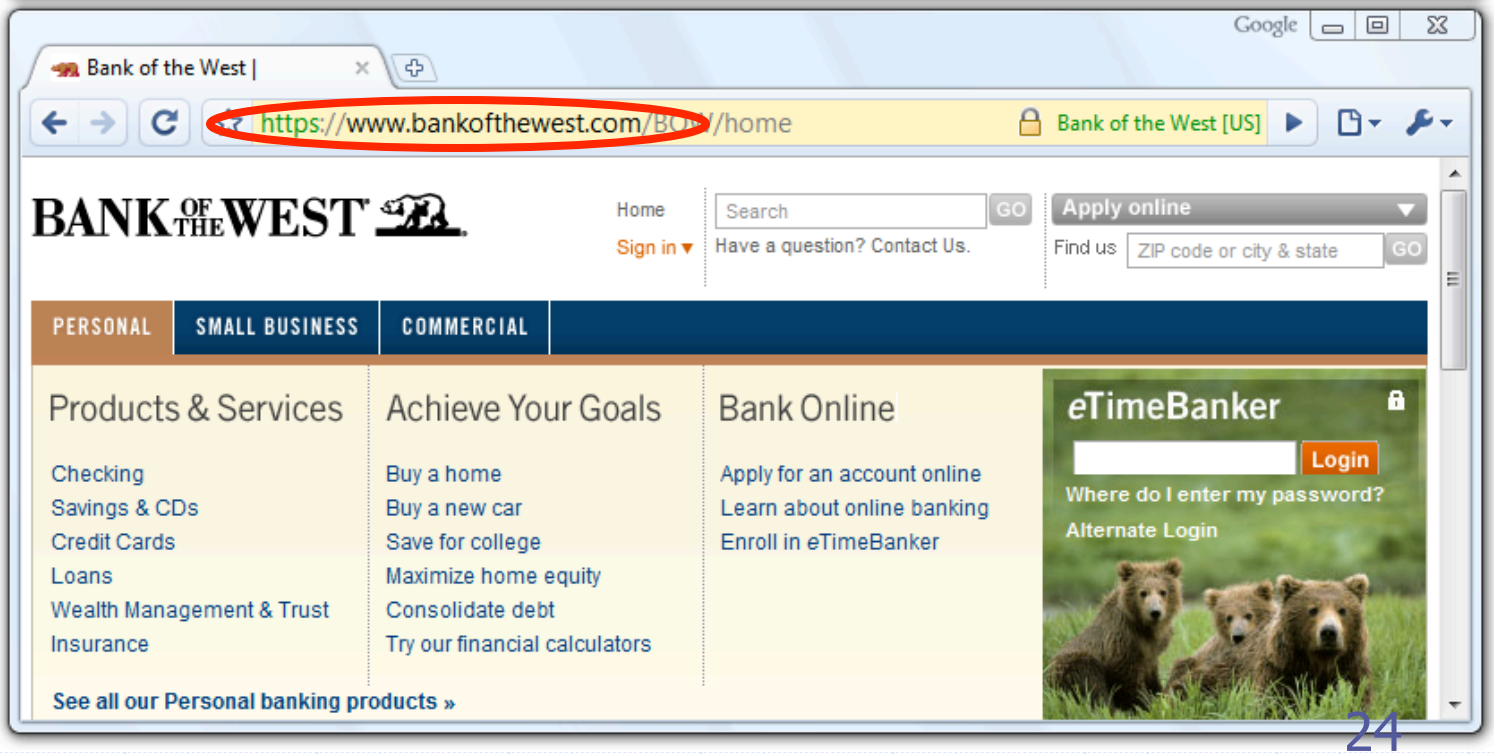
    mywin.document.body

    mywin.location = "http://www.c.com";

Windows Interact

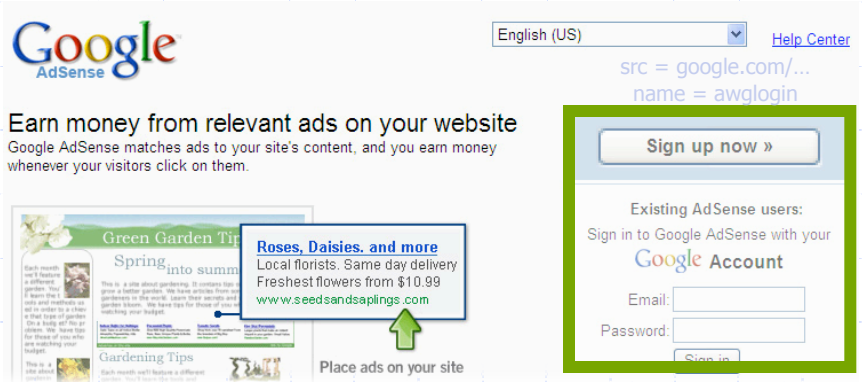


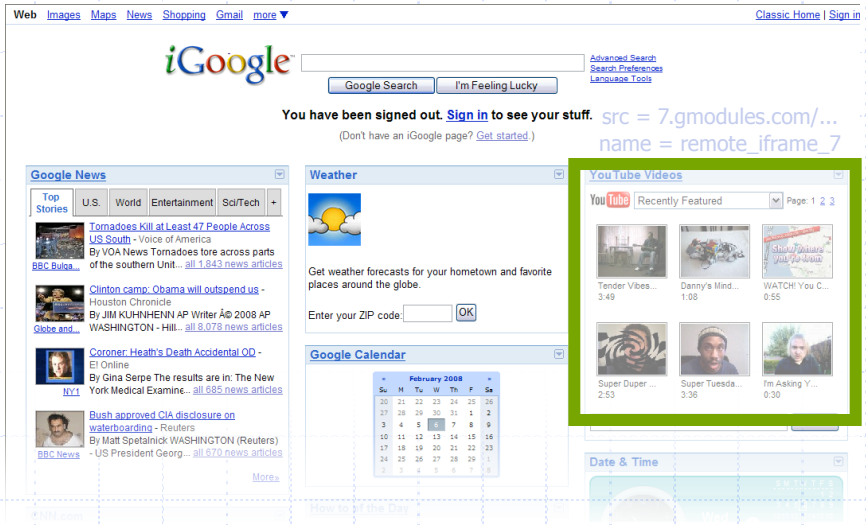
Are all interactions good?



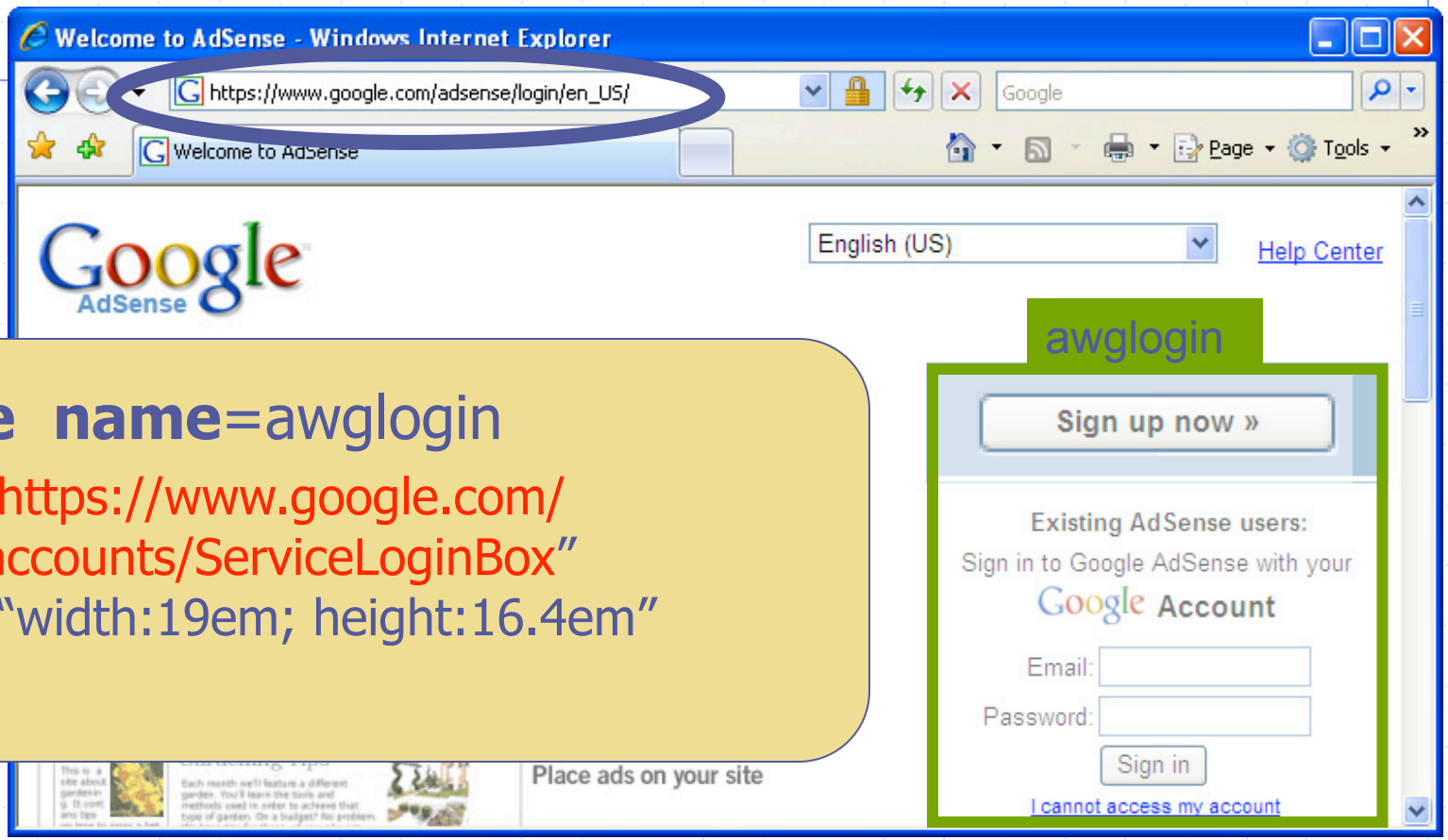
Frames

- Modularity
  - Brings together content from multiple sources
  - Client-side aggregation
- Delegation
  - Frame can draw only on its own rectangle





Frames and iFrames



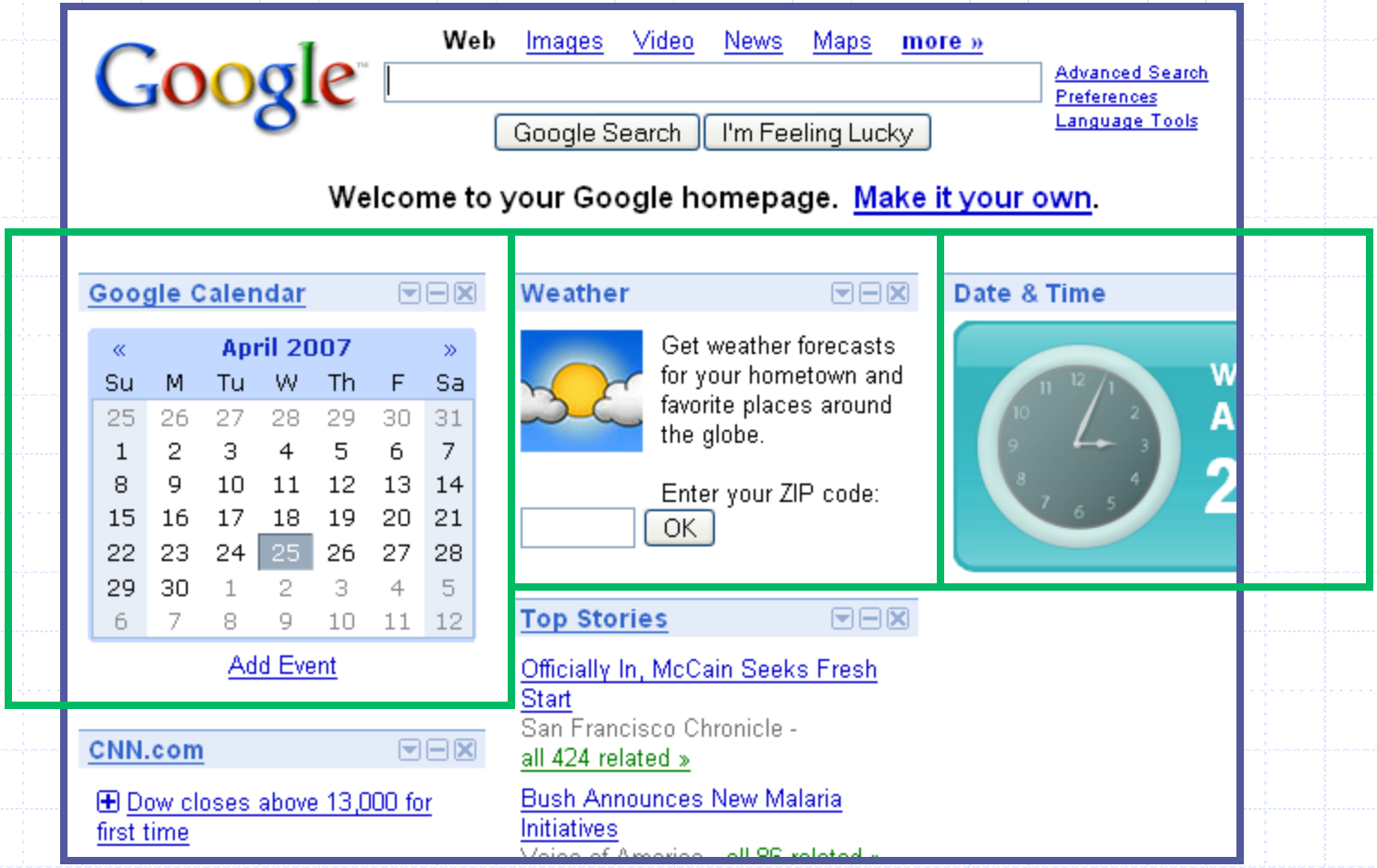

```
<iframe  name=awglogin
    src="https://www.google.com/
            accounts/ServiceLoginBox"
    style="width:19em; height:16.4em"
>
```

Address bar says nothing about origin of embedded content

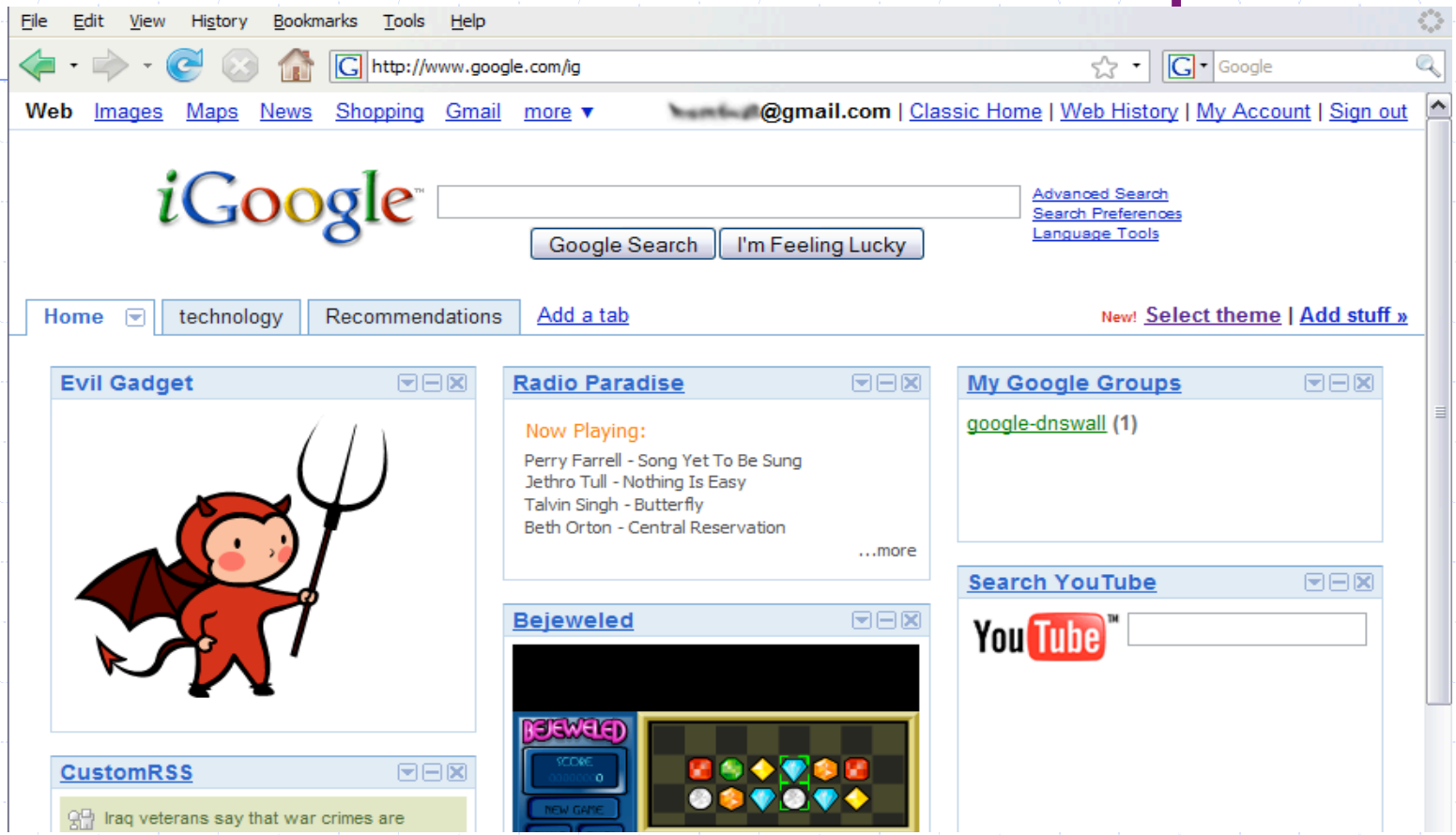
- frames (ads), scripts, flash objects, CSS

... but says nothing about where embedded content is from

Masups: lots of frames (gadgets)

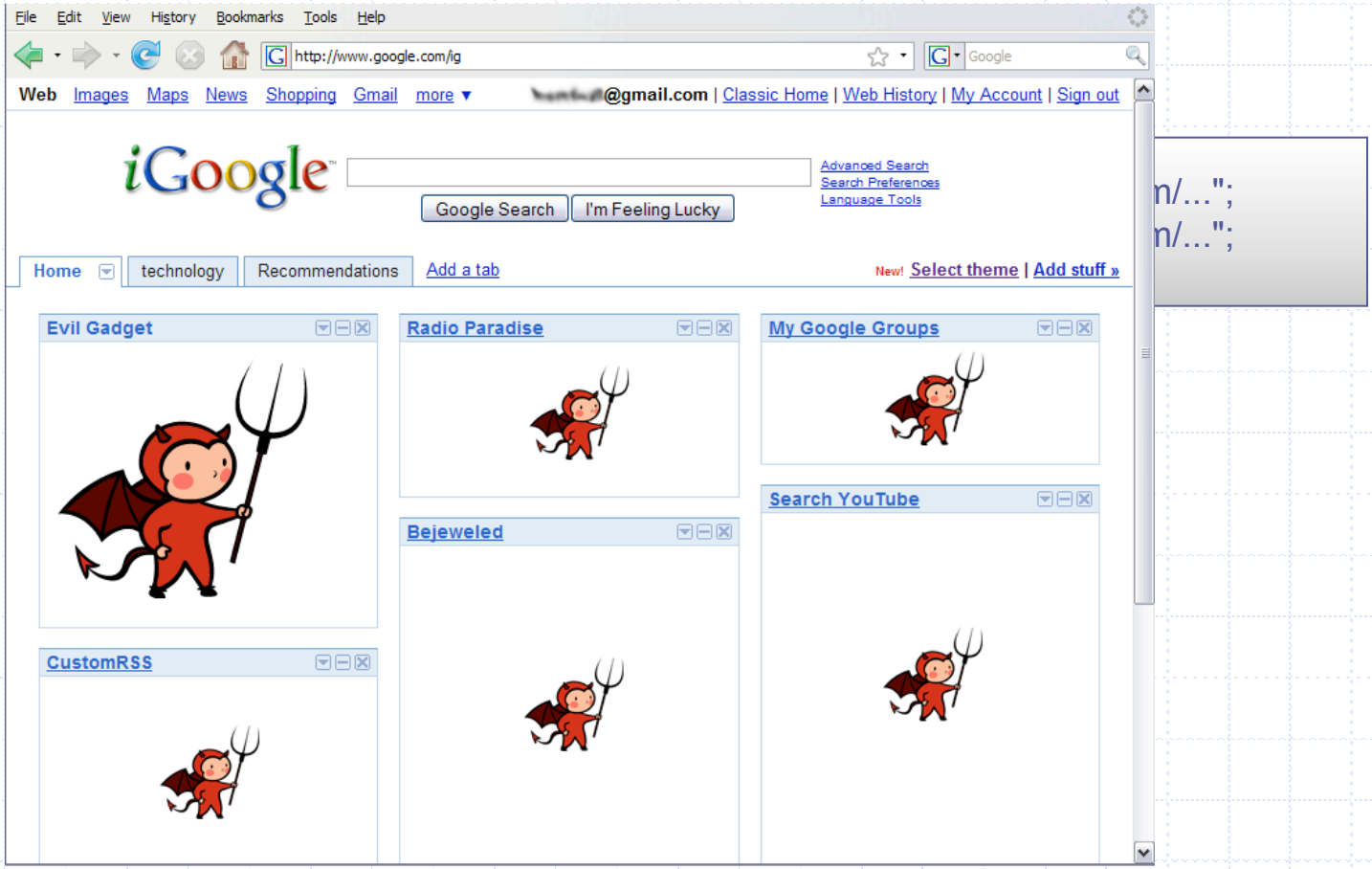


Need for isolation - mashups

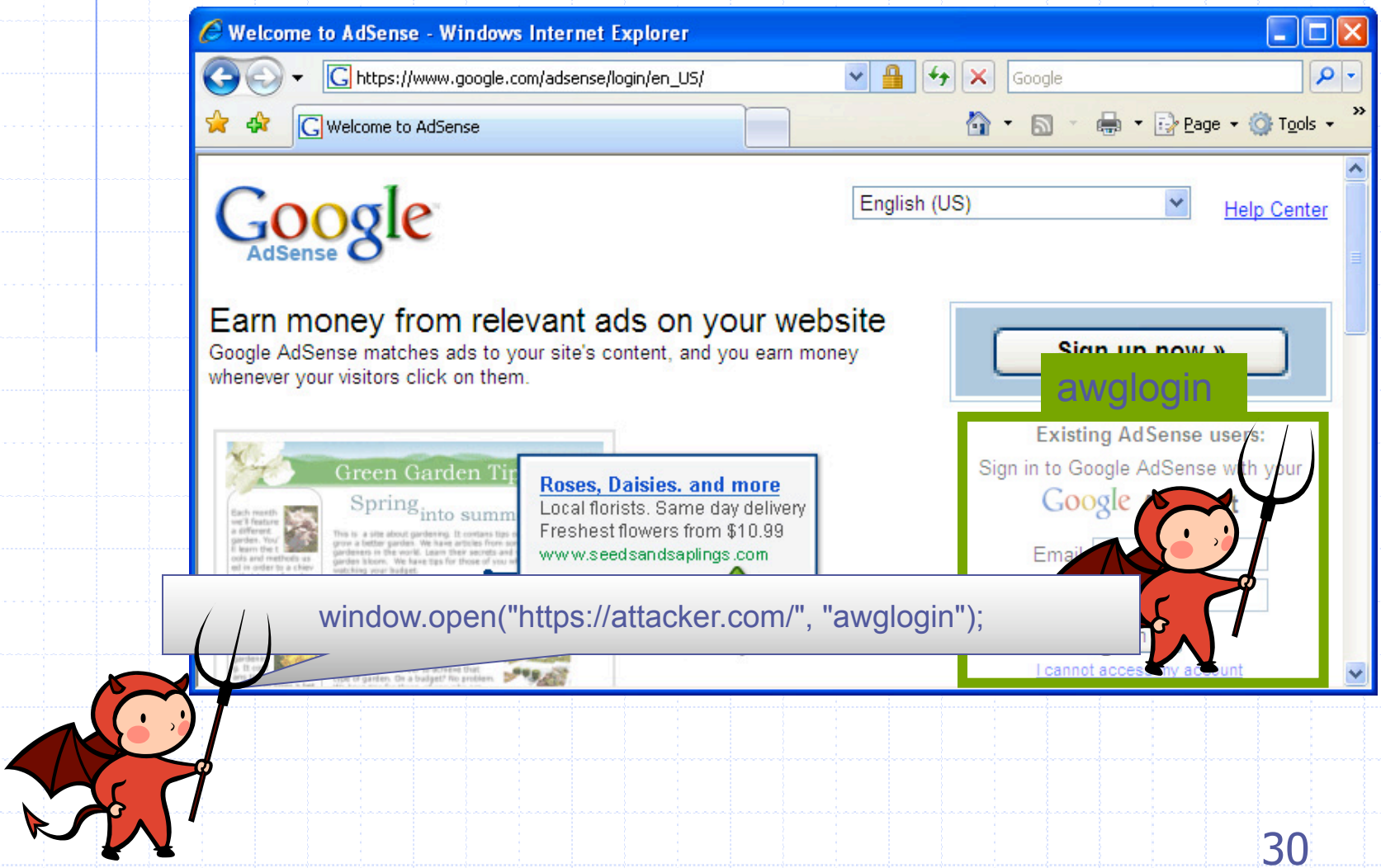


Malicious gadget should not affect other gadgets

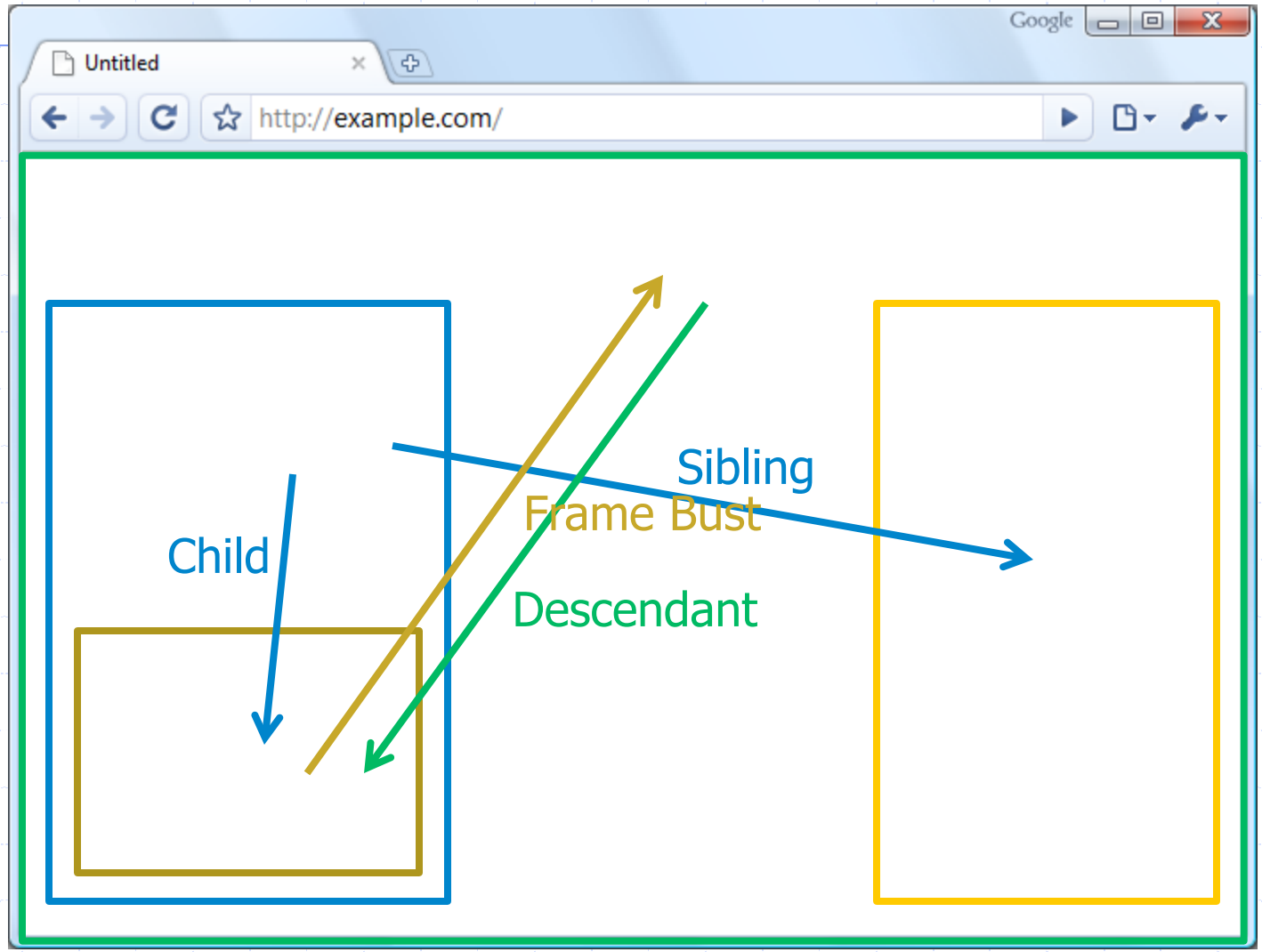
Window Policy Anomaly



A Guninski Attack



What should the policy be?



Legacy Browser Behavior

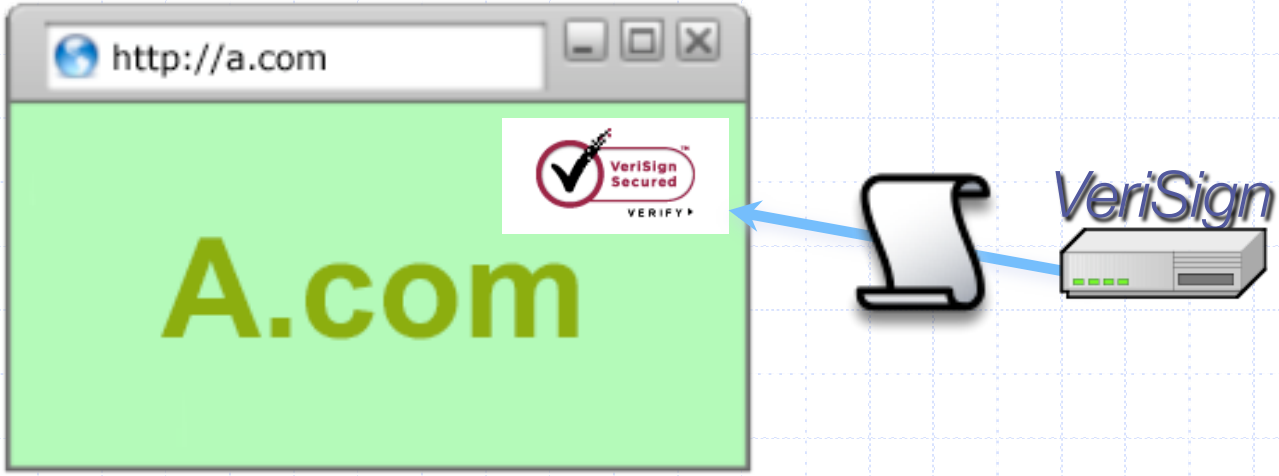
Browser	Policy
IE 6 (default)	Permissive
IE 6 (option)	Child
IE7 (no Flash)	Descendant
IE7 (with Flash)	Permissive
Firefox 2	Window
Safari 3	Permissive
Opera 9	Window
HTML 5	Child

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

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


Pages can embed content from many sources (example)

- 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>
```

Cross-origin Interaction

Sites often need to communicate:

- Google AdSense:

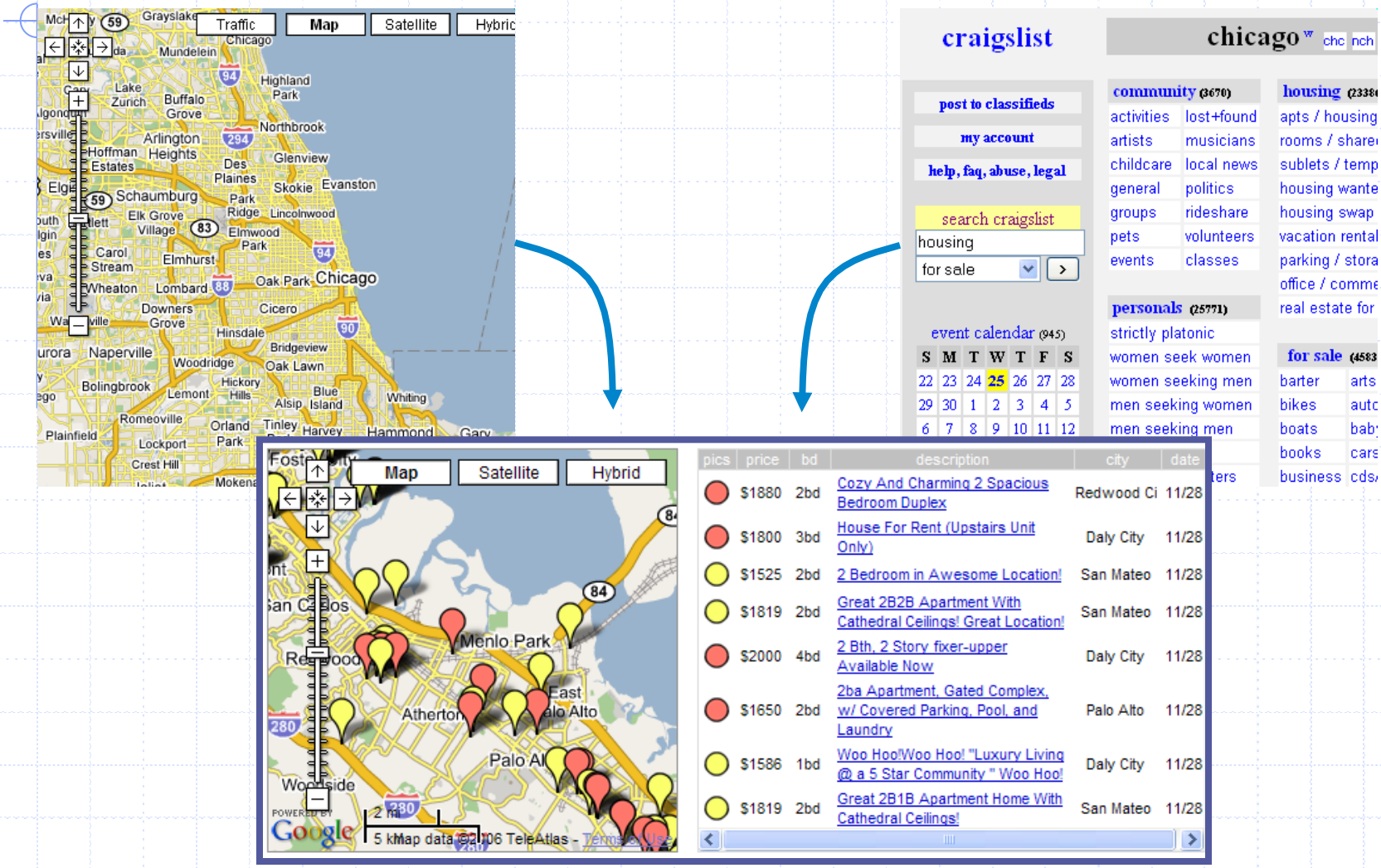
  <script src="http://googlesyndication.com/show_ads.js">

- Mashups
- Gadget aggregators (e.g. iGoogle or live.com)

- Primary method: script inclusion; site A does:
- **<script src=//siteB.com/script.js>**

- Script from B runs in A's origin: full control over A's DOM
- Note: to communicate with B, site A gives B full control !!

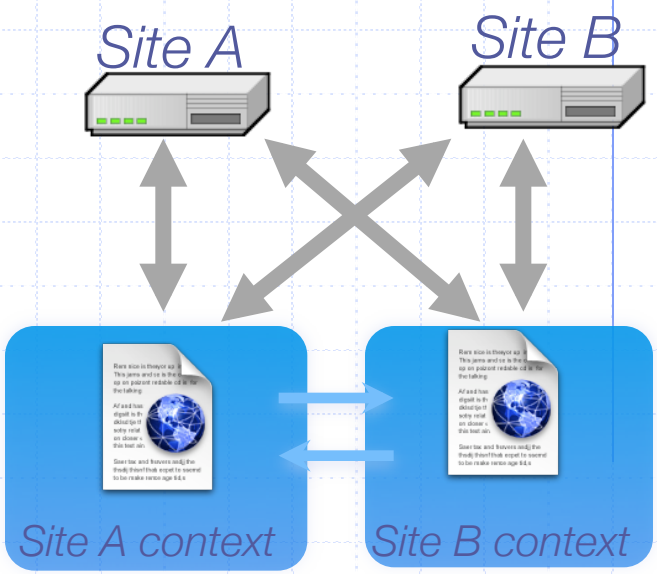
Mashups


Traffic
Map
Satellite
Hybrid
Chicago
Mundelein
Highland Park
Lake Zurich
Buffalo Grove
Northbrook
Arlington Heights
Hoffman Estates
Glenview
Des Plaines
Skokie
Evanston
Schaumburg
Elk Grove Village
Park Ridge
Lincolnwood
Elmwood Park
Carol Stream
Elmhurst
Oak Park
Wheaton
Lombard
Downers Grove
Cicero
Hinsdale
Bridgeview
Naperville
Woodridge
Oak Lawn
Bolingbrook
Lemont
Hickory Hills
Blue Island
Alsip
Whiting
Romeoville
Orland Park
Tinley
Harvey
Hammond
Plainfield
Lockport
Crest Hill
Menlo Park
Atherton
East Palo Alto
Palo Alto
Woodside
Google
5 km
Map data ©2006 TeleAtlas
craigslist
chicago
post to classifieds
my account
help, faq, abuse, legal
search craigslist
housing
for sale
event calendar (945)
S M T W T F S
community (3670)
activities
lost+found
artists
musicians
childcare
local news
general
politics
groups
rideshare
pets
volunteers
events
classes
personals (25771)
strictly platonic
women seek women
women seeking men
men seeking women
men seeking men
housing
apts / housing
rooms / share
sublets / temp
housing wante
housing swap
vacation rental
parking / stora
office / comme
real estate for
for sale
barter
arts
bikes
boats
books
business
pics
price
bd
description
city
date
$1880 2bd Cozy And Charming 2 Spacious Bedroom Duplex Redwood Ci 11/28
$1800 3bd House For Rent (Upstairs Unit Only) Daly City 11/28
$1525 2bd 2 Bedroom in Awesome Location! San Mateo 11/28
$1819 2bd Great 2B2B Apartment With Cathedral Ceilings! Great Location! San Mateo 11/28
$2000 4bd 2 Bth, 2 Story fixer-upper Available Now Daly City 11/28
$1650 2bd 2ba Apartment, Gated Complex, w/ Covered Parking, Pool, and Laundry Palo Alto 11/28
$1586 1bd Woo Hoo!Woo Hoo! "Luxury Living @ a 5 Star Community " Woo Hoo! Daly City 11/28
$1819 2bd Great 2B1B Apartment Home With Cathedral Ceilings! San Mateo 11/28

Need for isolation: embedded content

3rd party ad should not read/write enclosing DOM

Recent Developments

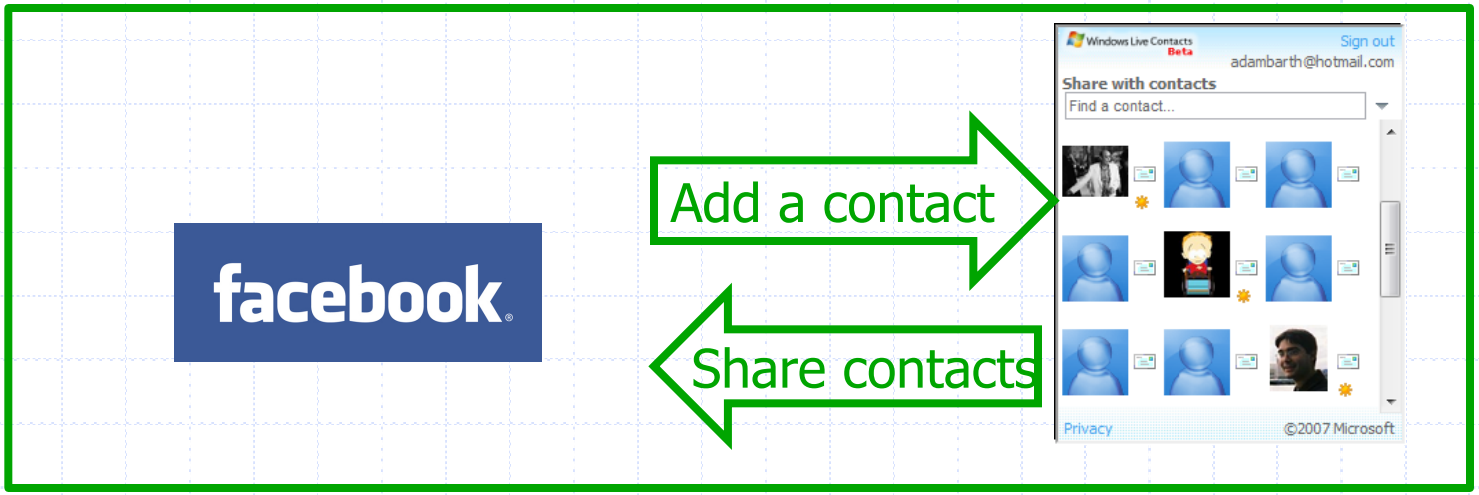


- Cross-origin network requests
  - Access-Control-Allow-Origin: <list of domains>
  - Access-Control-Allow-Origin: *
- Cross-origin client side communication
  - Client-side messaging via navigation (older browsers)
  - postMessage (newer browsers)

window.postMessage

- New API for inter-frame communication
  - Supported in latest betas of many browsers

  - A network-like channel between frames



postMessage syntax

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

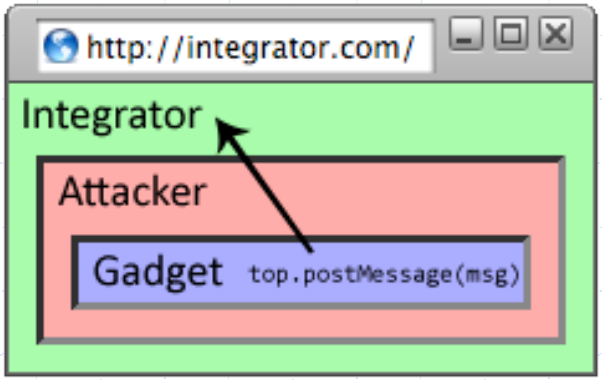
```
window.addEventListener("message", function (e) {
  if (e.origin == "http://a.com") {
    ... e.data ... }
}, false);
```

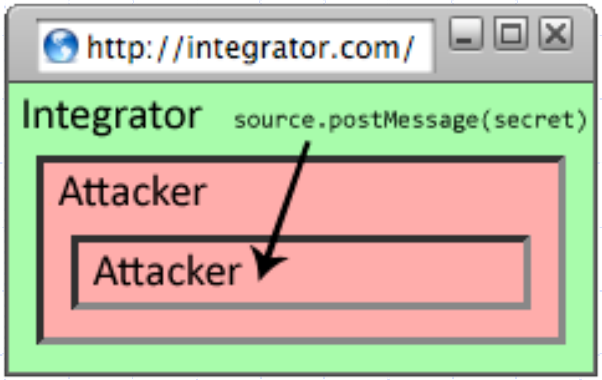
Why include "targetOrigin"?

- What goes wrong?

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

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





Data export

- Many ways to send information to other origins

```
<form action="http://www.bank.com/">
    <input name="data" type="hidden" value="hello">
</form>
<img src="http://www.b.com/?data=hello"/>
```

- No user involvement required
- Cannot read back response
- Read response only from your origin
- Some port are restricted (SMTP)

Same Origin Requests with XMLHttpRequest

```
<script>
var xhr = new XMLHttpRequest();
xhr.open("POST", "http://www.example.com:81/foo/
    example.cgi", true);  // asynchronous
xhr.send("Hello world!");
xhr.onload = function() {
  if (xhr.status == 200) {
    alert(xhr.responseText);
  }}
</script>
```

prepare request

read response

Sending a Cross-Domain GET

- Data must be URL encoded

<img src="http://othersite.com/file.cgi?foo=1&bar=x y">

- Browser sends:

  GET file.cgi?foo=1&bar=x**%20**y HTTP/1.1

  Host: othersite.com

  …

⇒ Any web page can send info to any site

- Denial of Service (DoS) using GET:
  - a popular site can DoS another site [Puppetnets '06]

Sending a Cross-Domain POST

```
<form method="POST" action="http://othersite.com/file.cgi" encoding="text/
    plain">
<input type="hidden" name="Hello world" value="4">
</form>
```

```
<script>document.forms[0].submit()</script>
```

submit
post

- Hidden iframe can do this in background
  - ⇒ user visits a malicious page, browser submits form on behalf of user
  - ⇒ e.g. page re-programs user's home router (XSRF)
- Can't send to some restricted ports, like 25 (SMTP)

Cookie Security

How to make HTTP statefull securely ?

Outline

- Web Refresher:
- Security User Interface
  - Goals of a browser
  - When is it safe to type my password?
- Same-Origin Policy
  - How sites are isolated
  - Opting out of isolation
  - Frame hijacking
  - Navigation policy
- Cookie security
- Browser security design

Same origin policy: "high level"

Review: Same Origin Policy (SOP) for DOM:

- Origin A can access origin B's DOM if match on
  (scheme, domain, port)

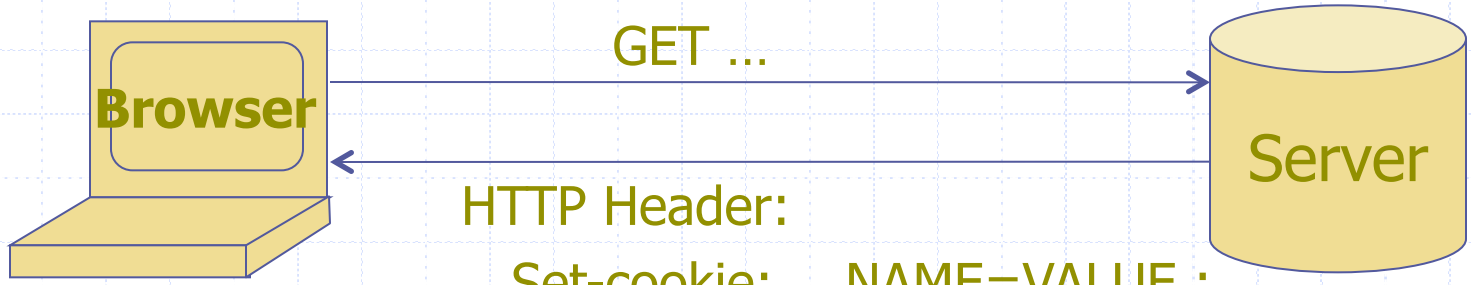
Today: Same Original Policy (SOP) for cookies:

- Generally speaking, based on:
  ([scheme], domain, *path*)

optional

scheme://domain:port/path?params

Setting/deleting cookies by server



HTTP Header:

Set-cookie: NAME=VALUE ;

domain = (when to send) ;
path = (when to send)

scope

secure = (only send over SSL);

expires = (when expires) ;

HttpOnly (later)

if expires=NULL:
this session only

- Delete cookie by setting "expires" to date in past
- Default scope is domain and path of setting URL

Scope setting rules (write SOP)

domain: any domain-suffix of URL-hostname, except TLD

example: host = "login.site.com"

allowed domains	disallowed domains
login.site.com	**user.site.com**
.site.com	**othersite.com**
	.com

⇒ **login.site.com** can set cookies for all of **.site.com** but not for another site or TLD

Problematic for sites like .stanford.edu

path: can be set to anything

Cookies are identified by (name,domain,path)

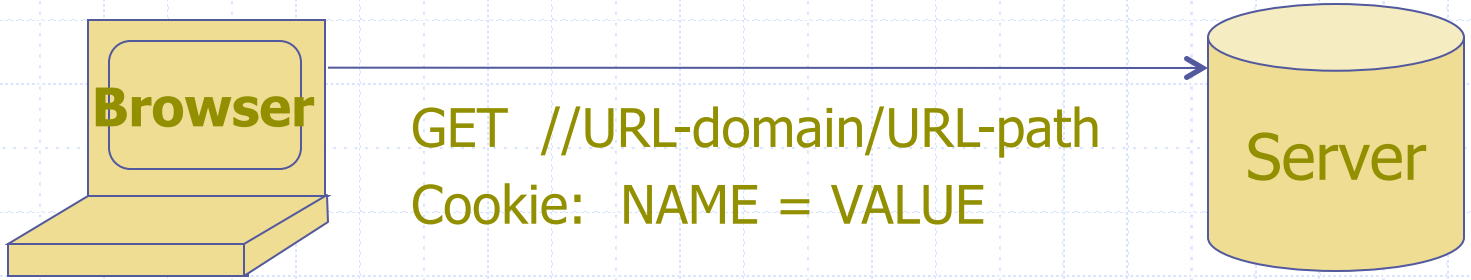
<u>cookie 1</u>
name = **userid**
value = test
domain = **login.site.com**
path = **/**
secure

<u>cookie 2</u>
name = **userid**
value = test123
domain = **.site.com**
path = **/**
secure

distinct cookies

- Both cookies stored in browser's cookie jar;
  both are in scope of **login.site.com**

Reading cookies on server (read SOP)



Browser sends all cookies in URL scope:

- cookie-domain is domain-suffix of URL-domain, and
- cookie-path is prefix of URL-path, and
- [protocol=HTTPS if cookie is "secure"]

Goal: server only sees cookies in its scope

Examples

both set by **login.site.com**

cookie 1	cookie 2
name = **userid**	name = **userid**
value = u1	value = u2
domain = **login.site.com**	domain = **.site.com**
path = **/**	path = **/**
secure	non-secure

http://checkout.site.com/	cookie: userid=u2
http://login.site.com/	cookie: userid=u2
https://login.site.com/	**cookie: userid=u1; userid=u2**

(arbitrary order)

Client side read/write: document.cookie

- Setting a cookie in Javascript:

  document.cookie = "name=value; expires=…; "

- Reading a cookie: alert(document.cookie)

  prints string containing all cookies available for document (based on [protocol], domain, path)

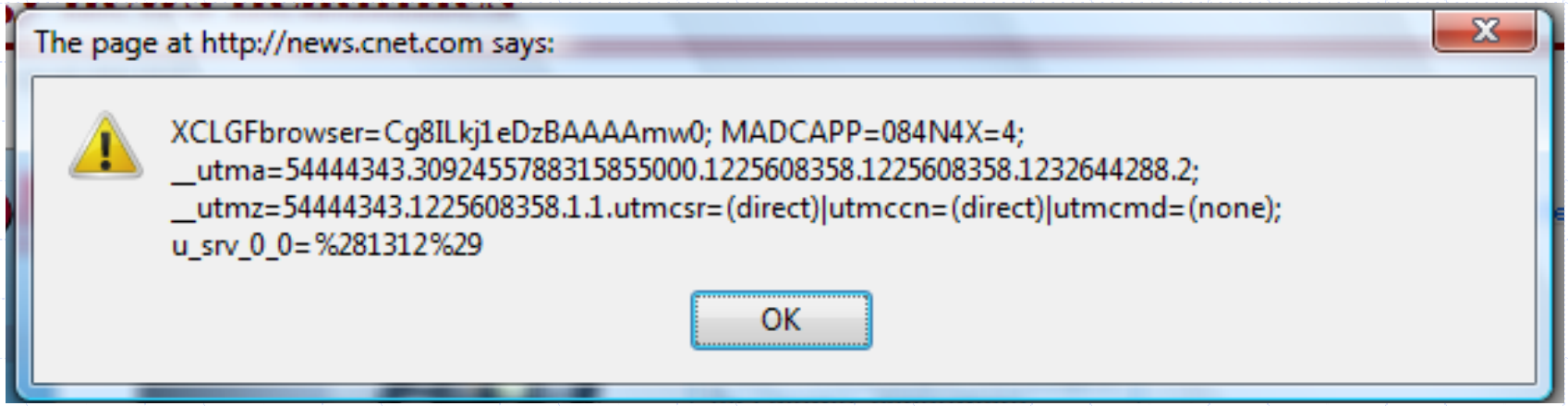
- Deleting a cookie:

  document.cookie = "name=; expires= Thu, 01-Jan-70"

document.cookie often used to customize page in Javascript

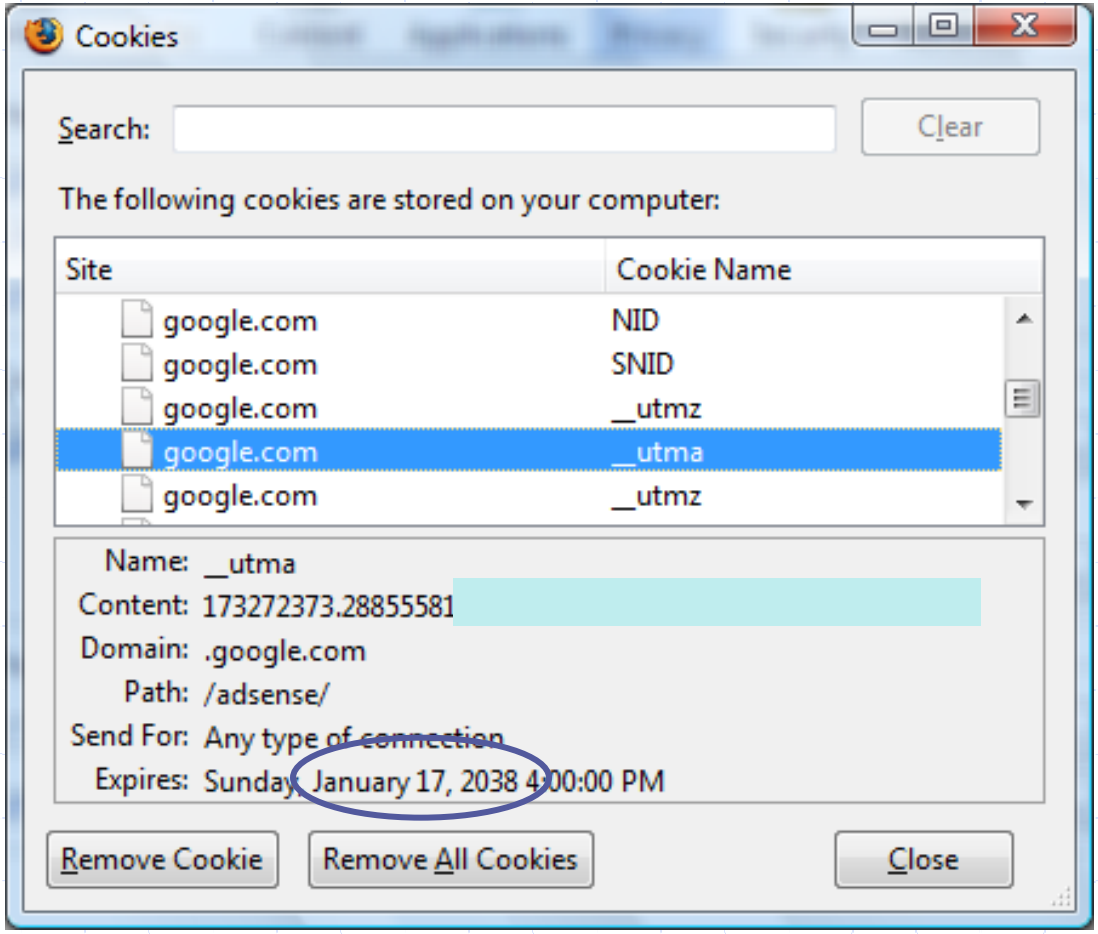
Javascript URL

javascript: alert(**document.cookie**)



Displays all cookies for current document

Viewing/deleting cookies in Browser UI



Cookie protocol problems

Server is blind:

- Does not see cookie attributes (e.g. secure)
- Does not see which domain set the cookie

Server only sees: **Cookie: NAME=VALUE**

Interaction with the DOM SOP

Cookie SOP: path separation

x.com/A does not see cookies of **x.com/B**

Not a security measure:

DOM SOP: **x.com/A** has access to DOM of **x.com/B**

```
<iframe src="x.com/B"></iframe>
alert(frames[0].document.cookie);
```

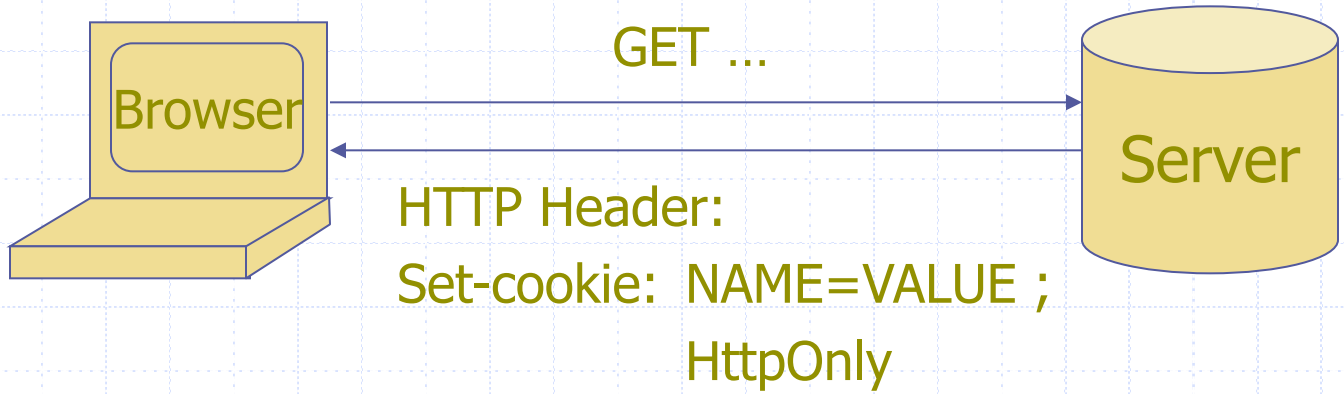
Path separation is done for efficiency not security:

x.com/A is only sent the cookies it needs

HttpOnly Cookies

IE6 SP1, FF2.0.0.5

(not Safari)



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

… but does not stop most other risks of XSS bugs.

Browser security design

How to build a secure browser ?

Outline

- Web Refresher:
- Security User Interface
  - Goals of a browser
  - When is it safe to type my password?
- Same-Origin Policy
  - How sites are isolated
  - Opting out of isolation
  - Frame hijacking
  - Navigation policy
- Cookie security
- Browser security design

Approach

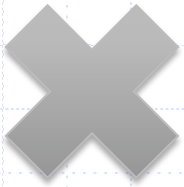
- Fact: Browsers will always have bugs
- Goal: Reduce the harm

Frequency of interactions with attacker ✖ Percentage of time vulnerability is unpatched ✖ Damage if attack works

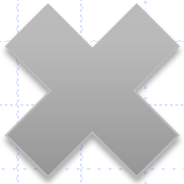
= Harm

Outline

Frequency of interactions with attacker



Percentage of time vulnerability is unpatched



Damage if attack works

1. Preventing the Introduction
2. Vulnerability Response
3. Failure Containment

Frequency of interactions with attacker

×

Percentage of time vulnerability is unpatched

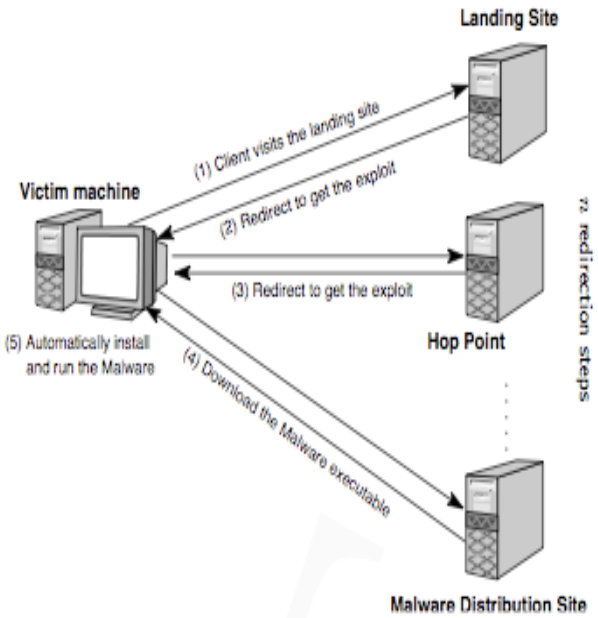
×

Damage if attack works

PREVENTING THE INTRODUCTION

Drive-by downloads

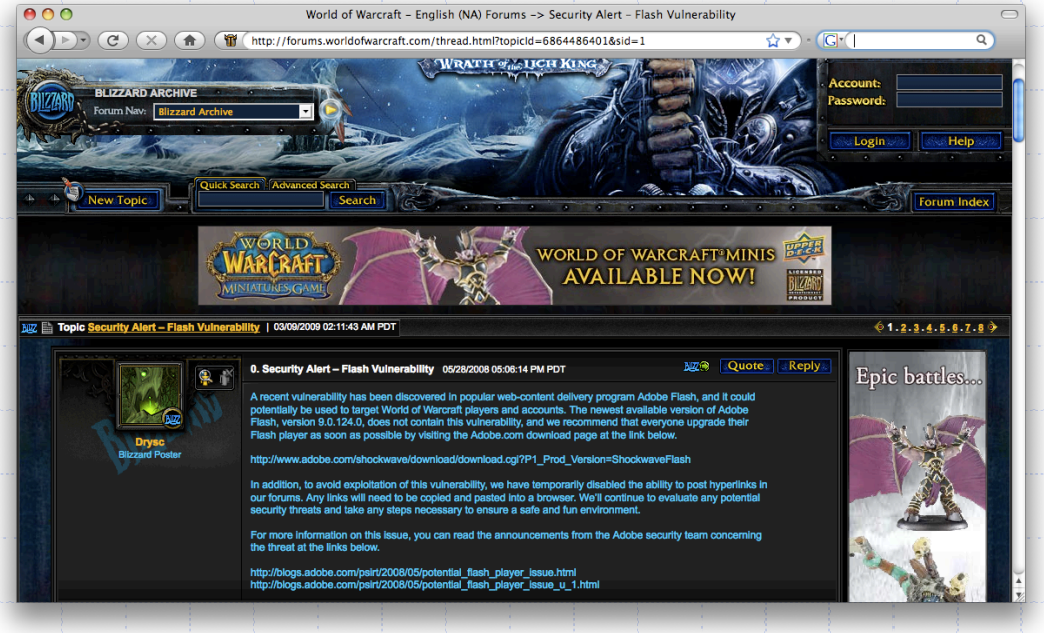
- Silently installs software when web page is loaded
- Increase exposure by compromising other sites and insert code into them
- Sites owners unaware they are participating in an attack



Provos et al. "All your iFRAMES Point to Us"

World of Warcraft keylogger

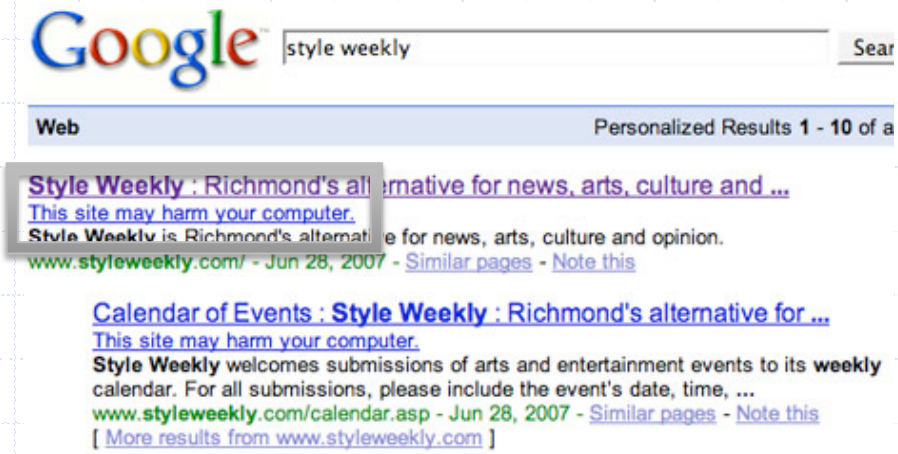
- Flash Player exploit used to install keylogger
- Links to malicious SWF posted on forums



- "Solution": Disable hyperlinks on forum

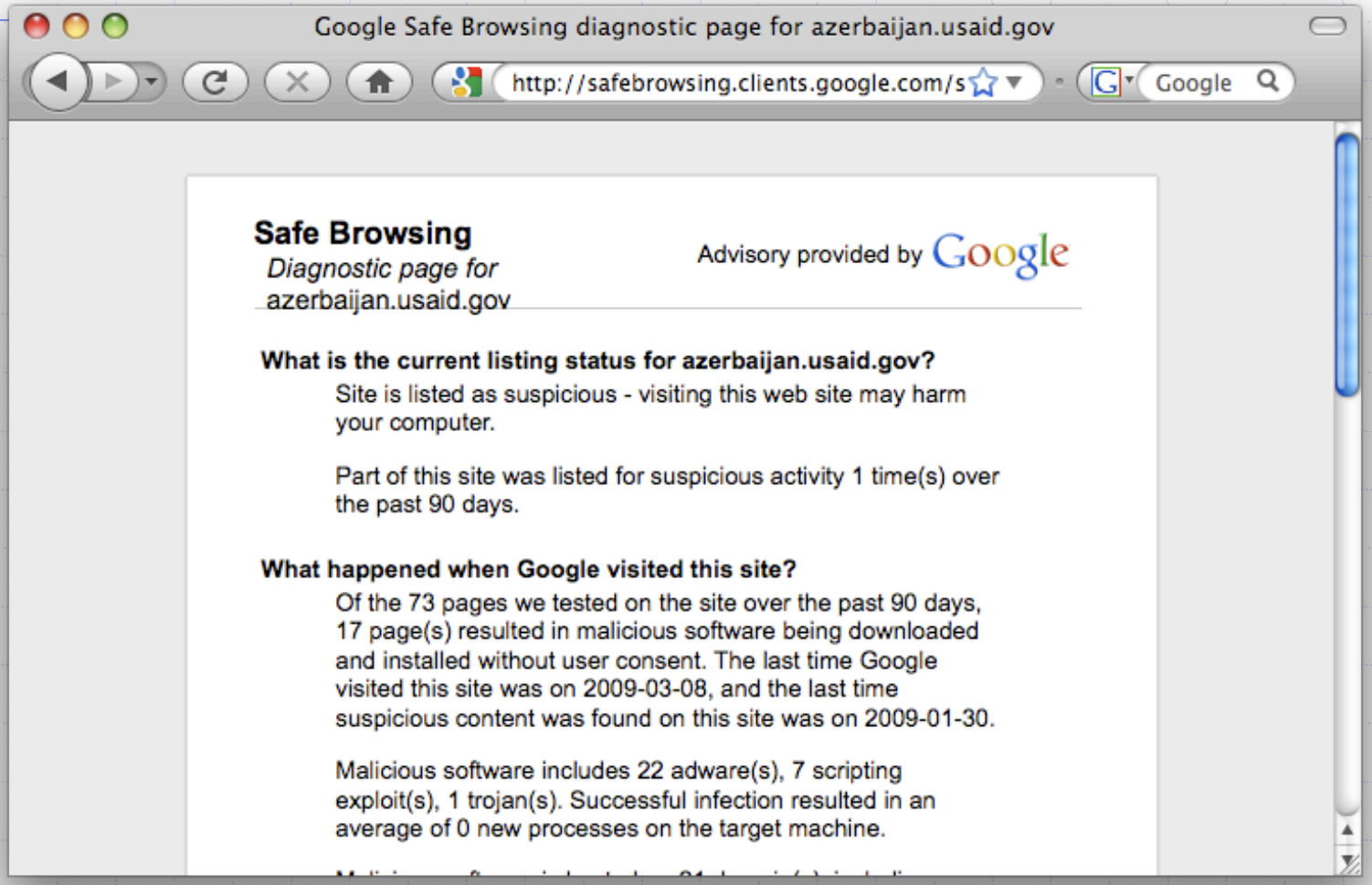
Scaling it up to the entire web

- 1.3% of the incoming search queries to Google returned at a least one malware site
- Visit sites with an army of browsers in VMs, check for changes to local system
- Indicate potentially harmful sites in search results



Now do it in the browser

Helping the webmaster out

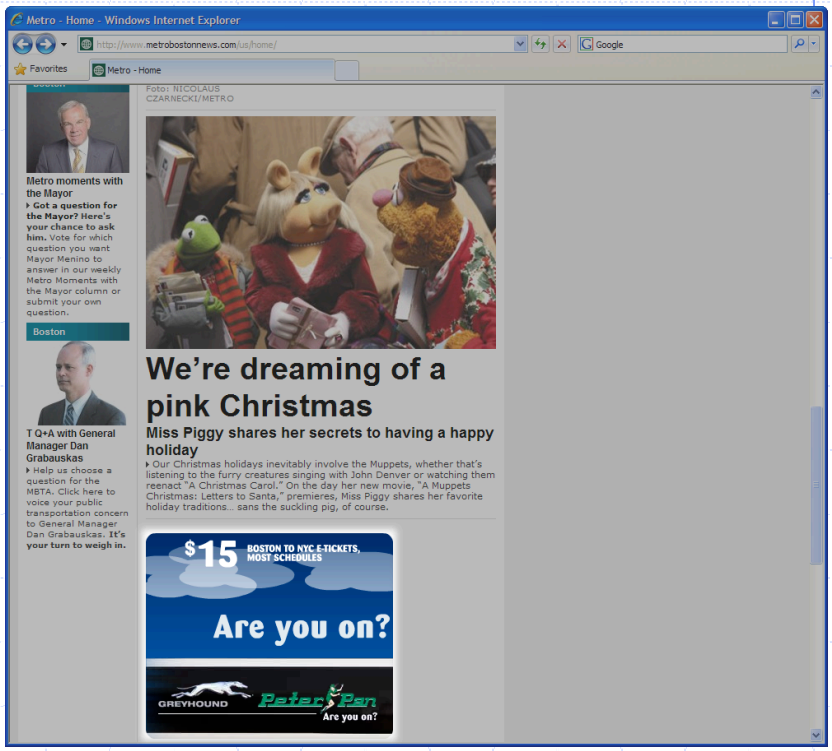


Introductions are easy

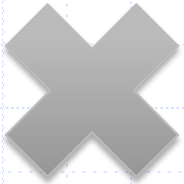
- Impressions are cheap ($1 = 2000)
- Ad that is harmless today may be malicious tomorrow
- Possible mitigations:

  <iframe security=restricted>

  <iframe sandbox>



Frequency of interactions with attacker

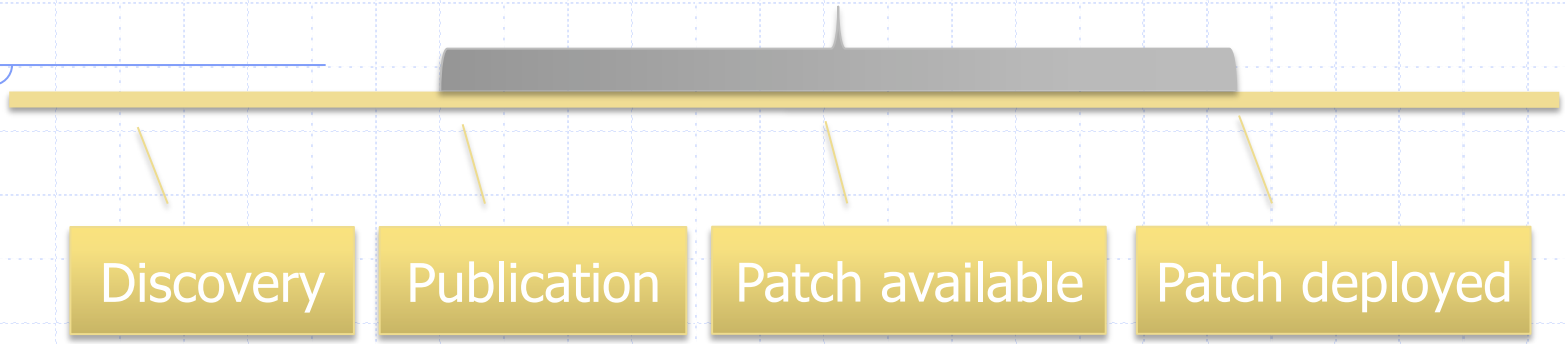


Percentage of time vulnerability is unpatched

Damage if attack works

VULNERABILITY RESPONSE

Closing the vulnerability window



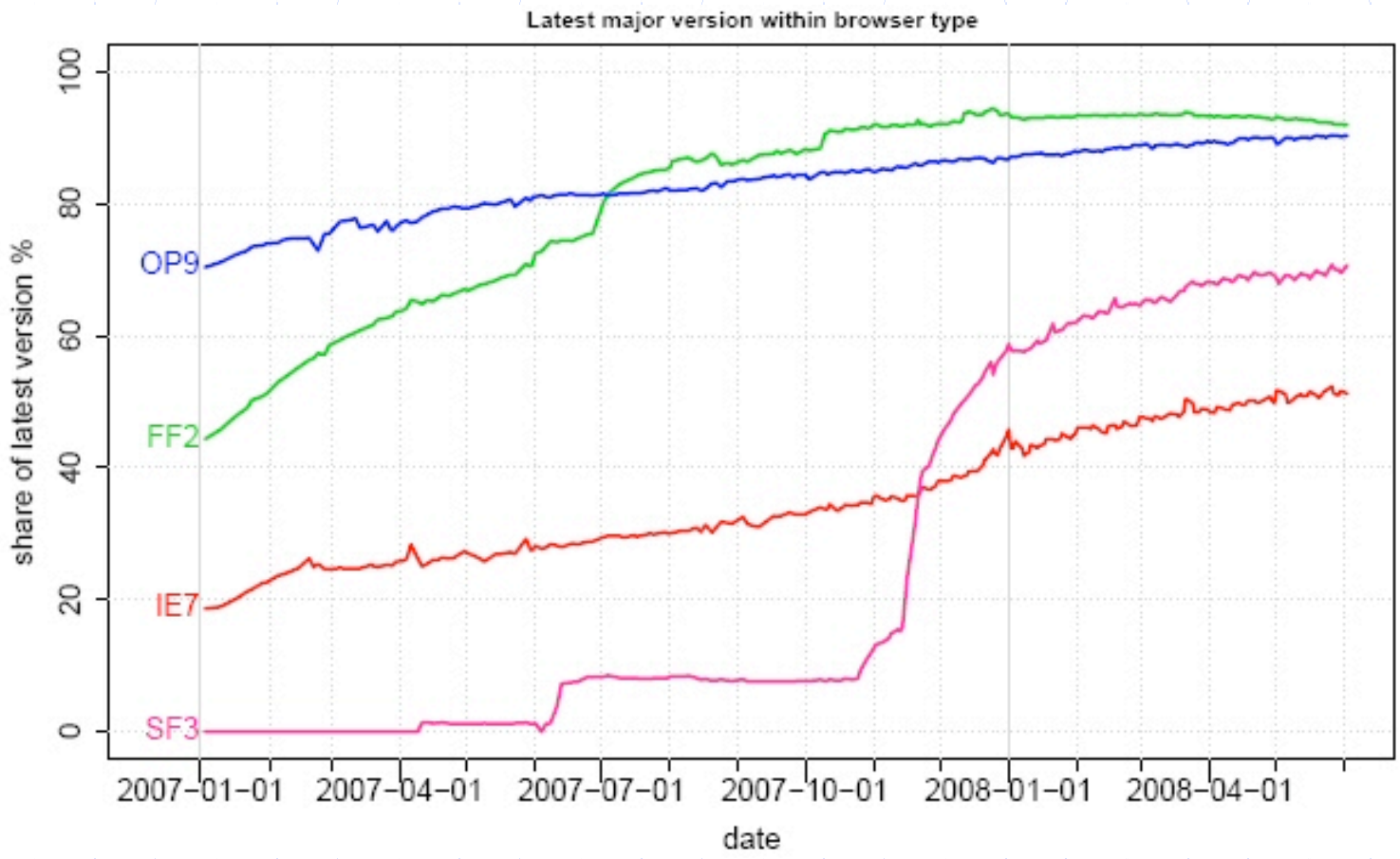
- Delay publication
  - Coordinate with security researchers
  - Offer prizes for responsibly disclosed security bugs
- Make patch available faster
- Deploy patch faster

Obstacles to patch deployment

- Interrupts work flow
- Requires adminstrator privileges
- Risk of breaking things
- Separate update mechanisms

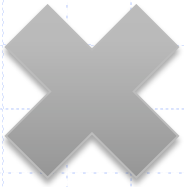
- Silent approach: GoogleUpdate.exe

Getting better, but not fast enough

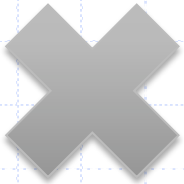


Frei et al. Examination of vulnerable online Web browser populations and the "insecurity iceberg"

Frequency of interactions with attacker



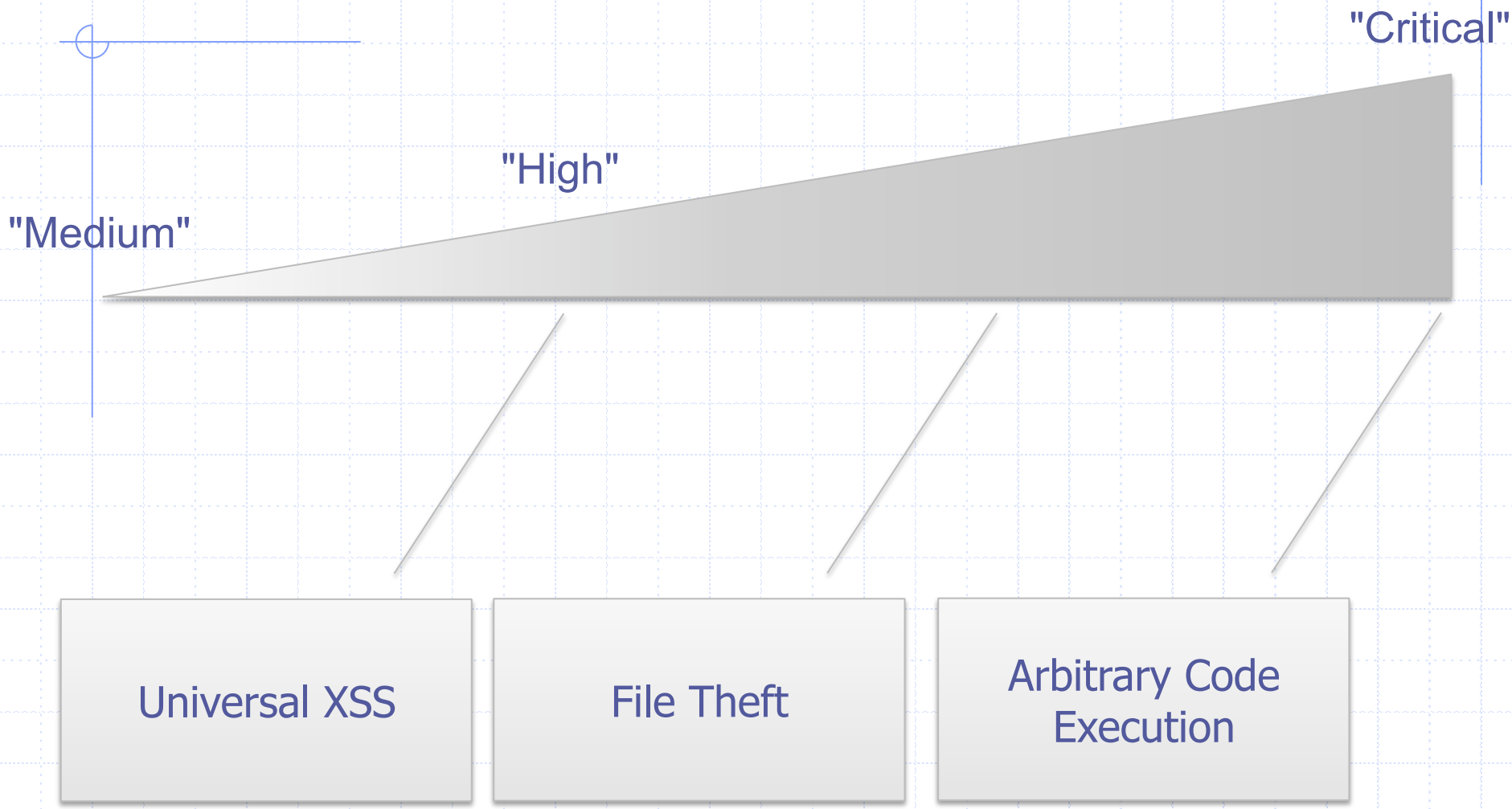
Percentage of time vulnerability is unpatched



Damage if attack works

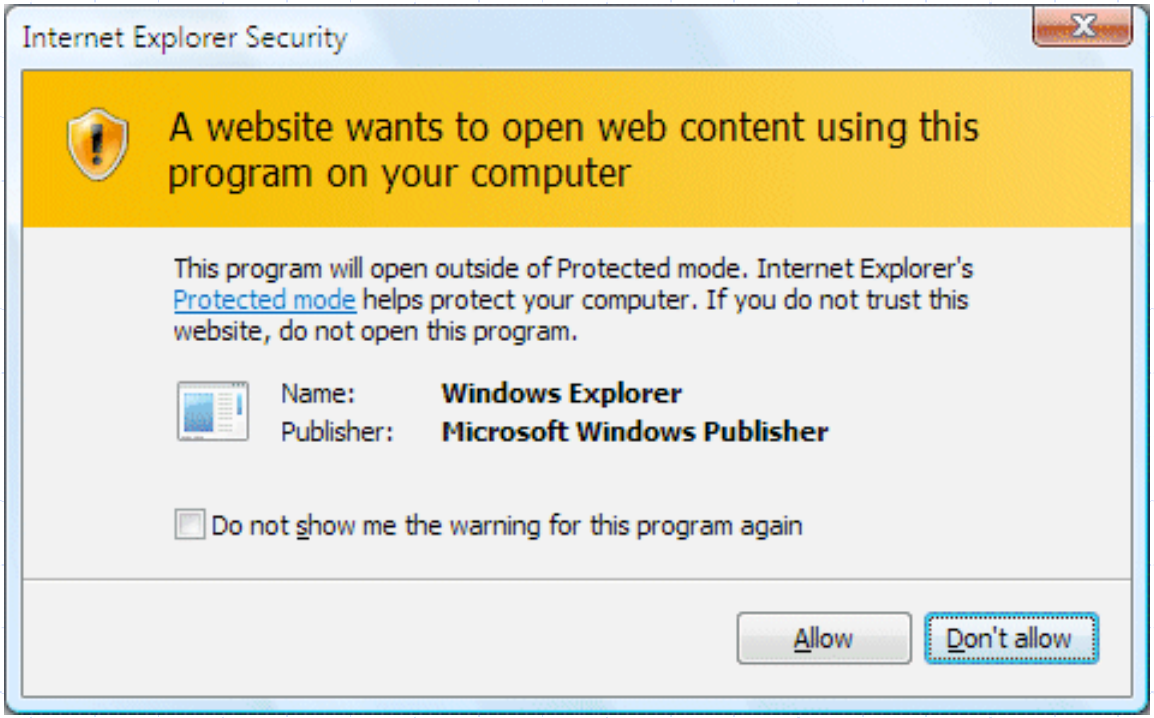
FAILURE CONTAINMENT

Severity


"Critical"
"High"
"Medium"
Universal XSS
File Theft
Arbitrary Code Execution

Protected Mode IE

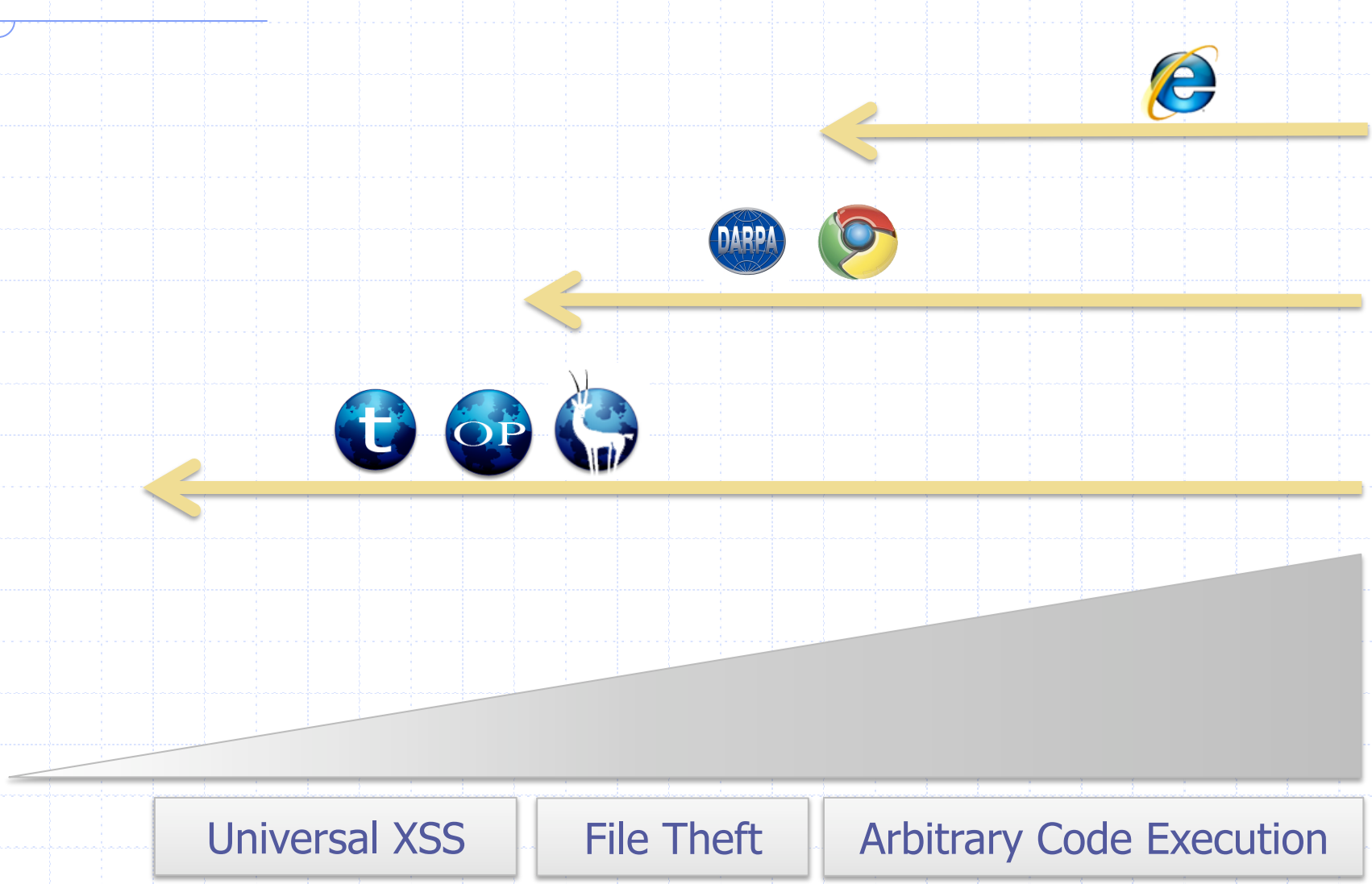
- IE7 in Vista is a "low rights" process
- Can prompt user to get more privileges



IE7 Containment Goals

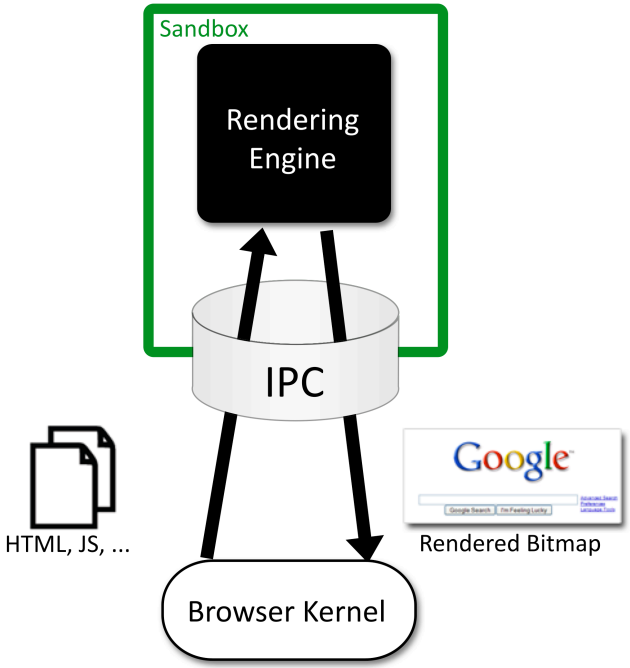
- Arbitrary code execution won't let attacker:
  - Install software
  - Copy files to startup folder
  - Change homepage or search provider setting
- Can we do more?

Containment Goals



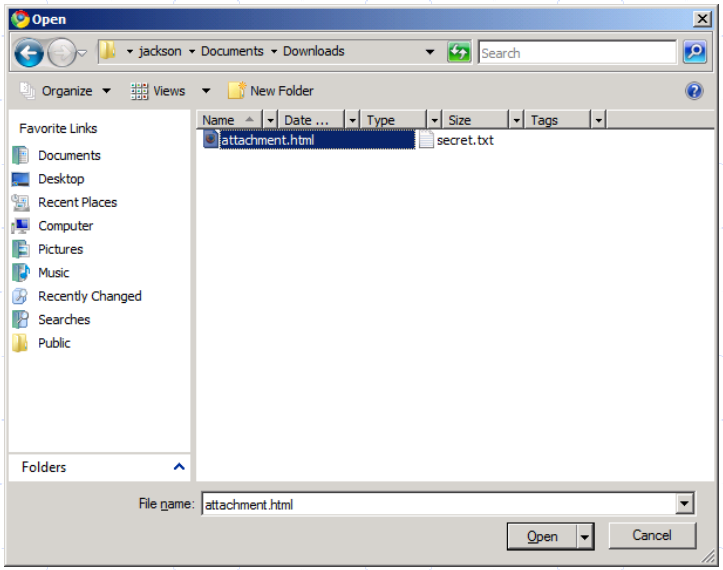
Chromium Security Architecture

- Browser ("kernel")
  - Full privileges (file system, networking)
  - Coarse-grained security policies protect local system
- Rendering engine
  - Sandboxed
  - Fine-grained same origin policy enforcement
- One process per plugin
  - Sandboxing optional



Barth et al. "The Security Architecture of the Chromium Browser"

Preventing File Theft



- File Downloads.
  - Renderer can only write files to My Documents\Downloads
- File Uploads.
  - Renderer is granted ability to upload file using browser kernel's file picker.
- Network Requests.
  - Can only request web-safe schemes (http, https, ftp)
  - Dedicated renderers for file://

Task Allocation

Rendering Engine	Browser Kernel
HTML parsing	Cookie database
CSS parsing	History database
Image decoding	Password database
JavaScript interpreter	Window management
Regular expressions	Location bar
Layout	Safe Browsing blacklist
Document Object Model	Network stack
Rendering	SSL/TLS
SVG	Disk cache
XML parsing	Download manager
XSLT	Clipboard

Both
URL parsing
Unicode parsing

Is the "kernel" too complex?

- Total CVEs:

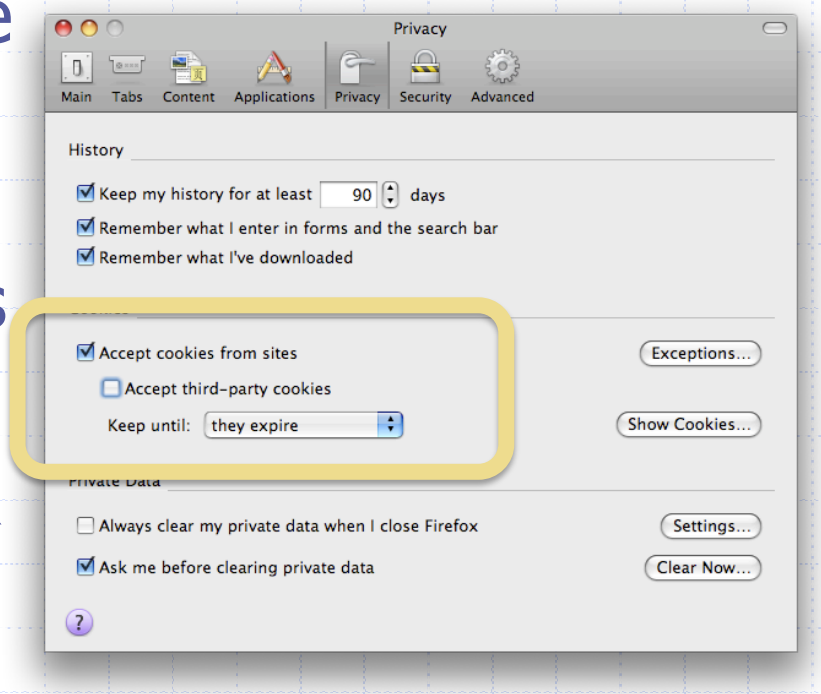
	Browser	Renderer	Unclassified
Internet Explorer	4	10	5
Firefox	17	40	3
Safari	12	37	1

- Arbitrary code execution vulnerabilities:

	Browser	Renderer	Unclassified
Internet Explorer	1	9	5
Firefox	5	19	0
Safari	5	10	0

Another approach: Cookie Blocking

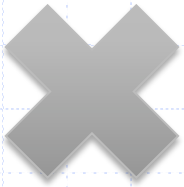
- Block the "Cookie" header for cross-domain resource loads
- Third-party cookie blocking already does this for privacy
- Third-party frames are ok
- Cross-subdomain might be ok



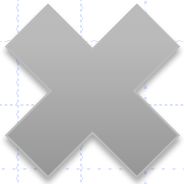
Open question: How many sites does this break compared to content type filtering?

Conclusion

Frequency of interactions with attacker



Percentage of time vulnerability is unpatched



Damage if attack works

1. Preventing the Introduction
2. Vulnerability Response
3. Failure Containment