CS 380S

Web Browser Security

Vitaly Shmatikov

(most slides from the Stanford Web security group)




Reading Assignment

- Jackson and Barth. "Beware of Finer-Grained Origins" (W2SP 2008).
- Chen et al. "Pretty-Bad-Proxy: An Overlooked Adversary in Browsers' HTTPS Deployments" (Oakland 2009).
- Optional: Barth et al. "Securing Frame Communication in Browsers" (Usenix Security 2008 and CACM).
- Optional: Barth et al. "Cross-Origin JavaScript Capability Leaks" (Usenix Security 2009).

JavaScript Security Model (Redux)

- Same-origin policy
  - Frame can only read properties of documents and windows from same place: server, protocol, port
- Does not apply to scripts loaded in enclosing frame from arbitrary site

```
<script type="text/javascript">
    src="http://www.example.com/scripts/somescript.js">
</script>
```

  - This script runs as if it were loaded from the site that provided the page!

OS vs. Browser Analogies (Redux)

Operating system

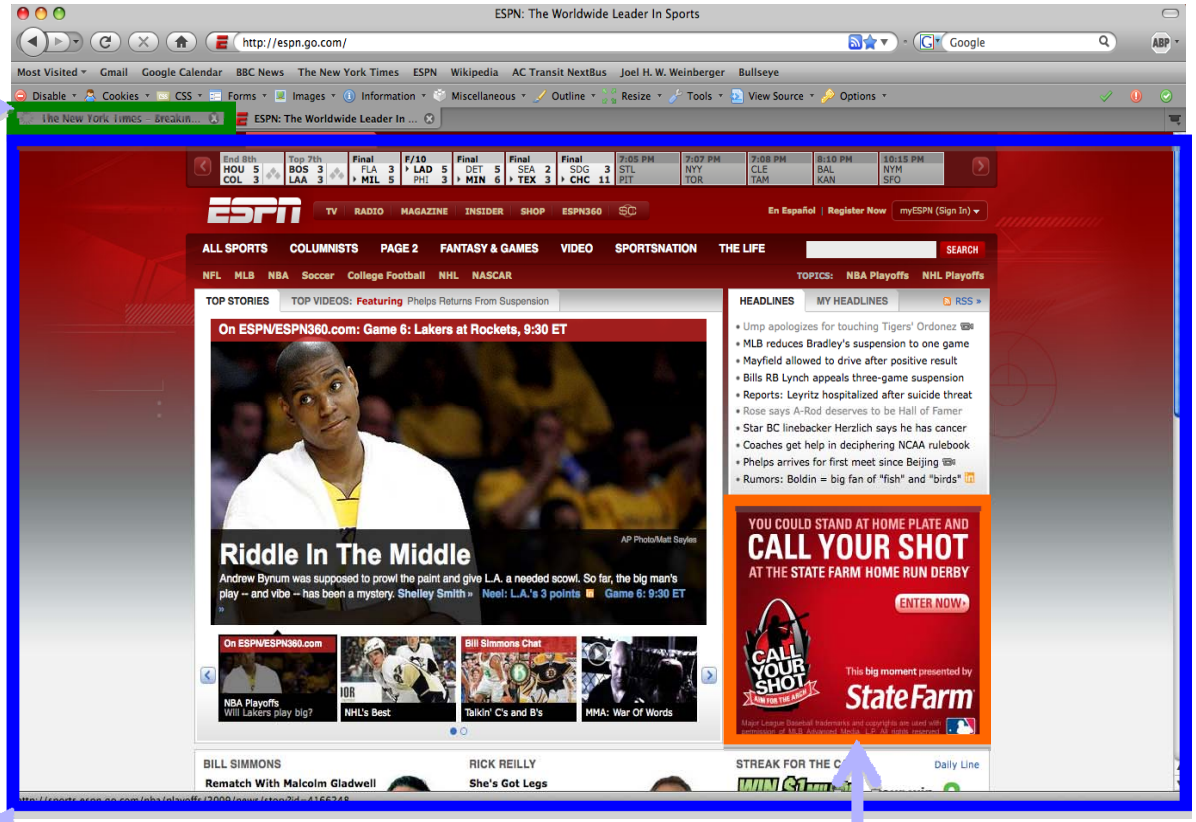
- Primitives
  - System calls
  - Processes
  - Disk
- Principals: Users
  - Discretionary access control
- Vulnerabilities
  - Buffer overflow
  - Root exploit

Web browser

- Primitives
  - Document object model
  - Frames
  - Cookies / localStorage
- Principals: “Origins”
  - Mandatory access control
- Vulnerabilities
  - Cross-site scripting
  - Universal scripting

JavaScript Contexts

JavaScript context 1

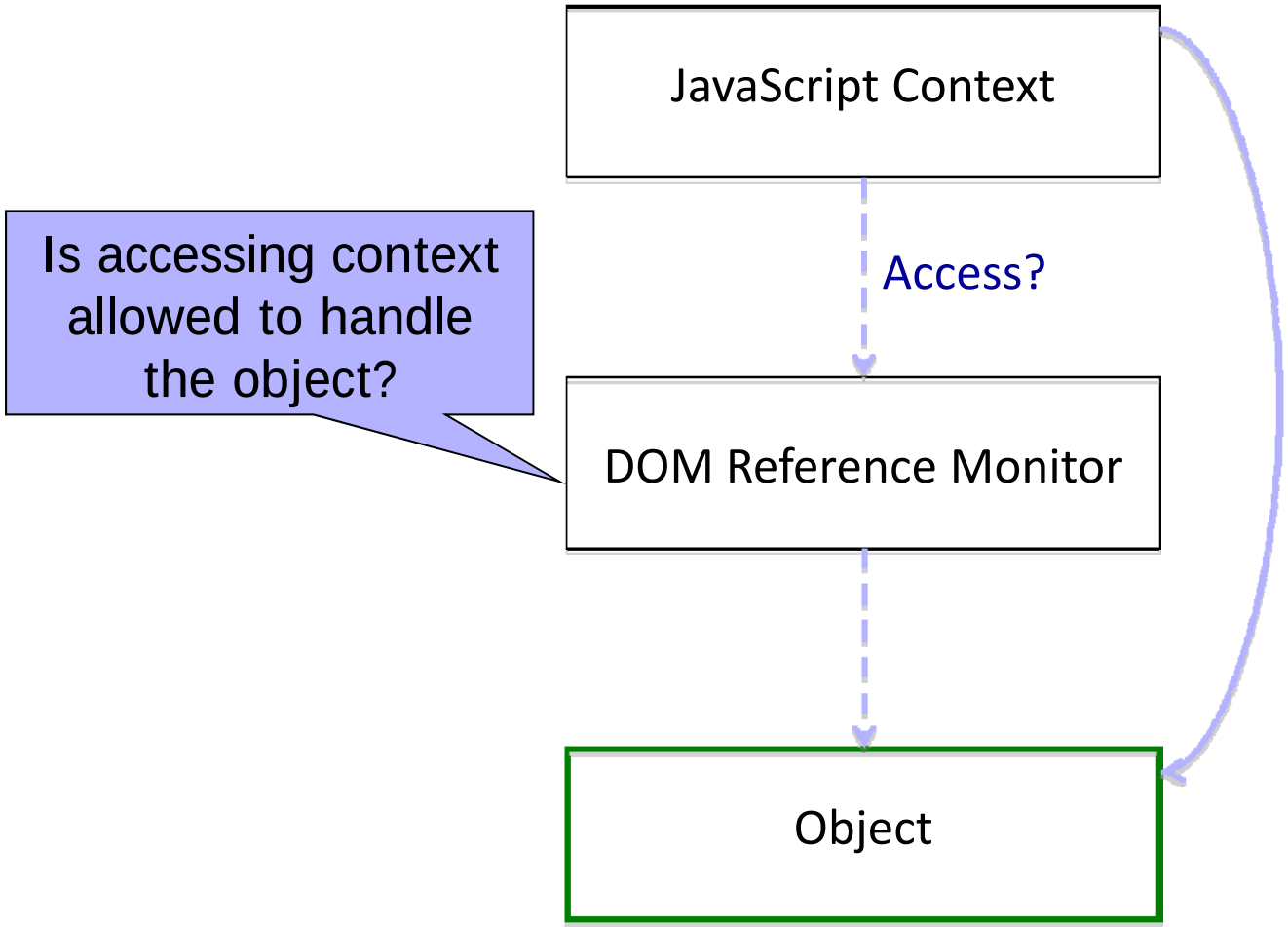


JavaScript context 2

JavaScript context 3

DOM and Access Control

[Barth et al.]



Granted: give reference to object to JavaScript

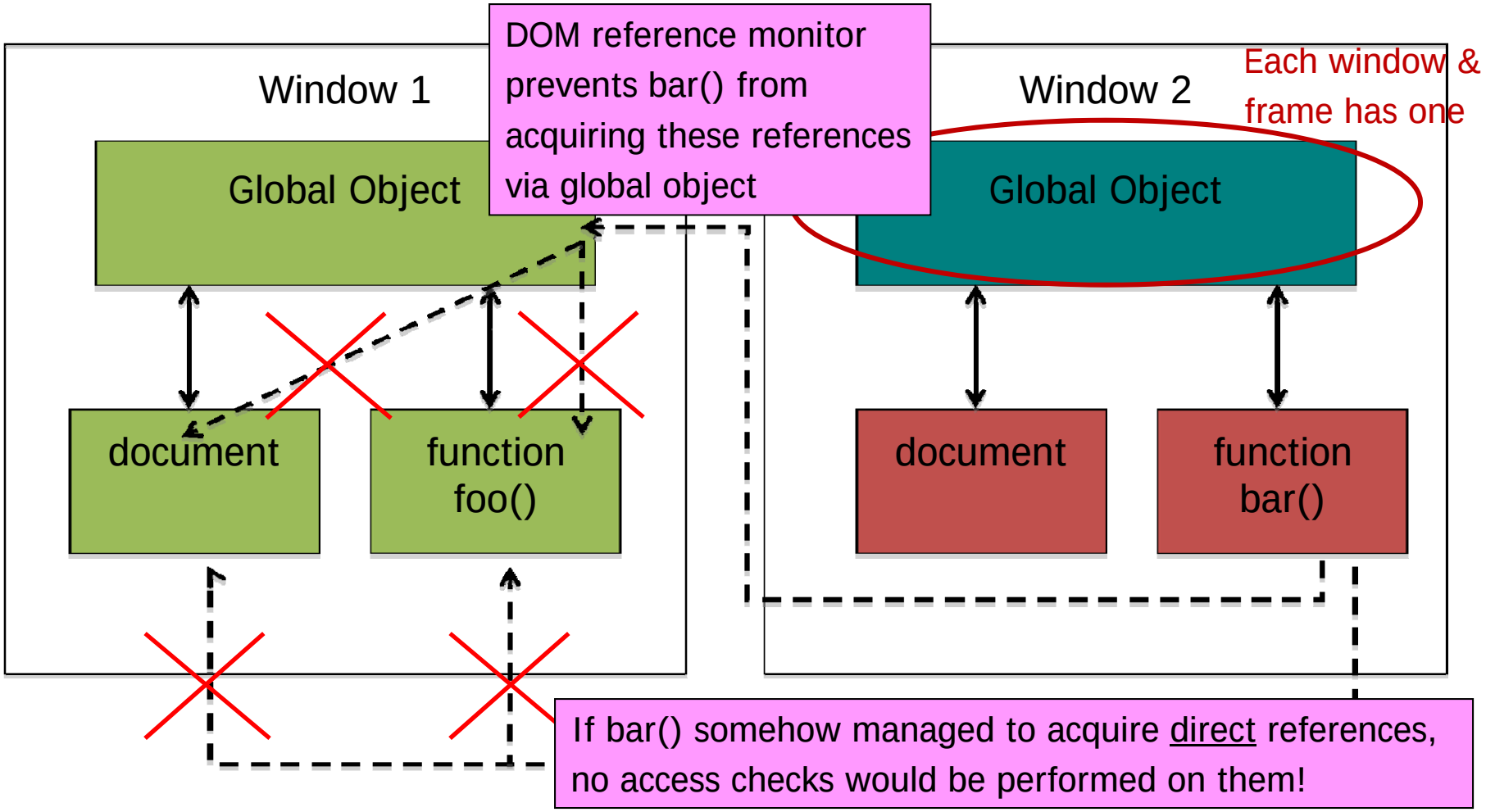
DOM vs. JavaScript Engine

[Barth et al.]

- DOM: performs access control checks
  - When a DOM object is initially accessed, check if it's Ok to give out a reference to this object
- JavaScript engine: uses references as if they were capabilities
  - If context has a reference to an object, can use it without any access control checks
- ... but these are the same DOM objects!
- What if a reference to an object leaks from one JavaScript context to another?

Cross-Context References

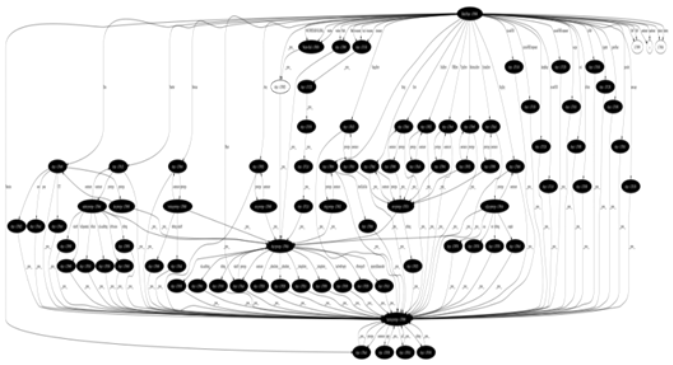
[Barth et al.]



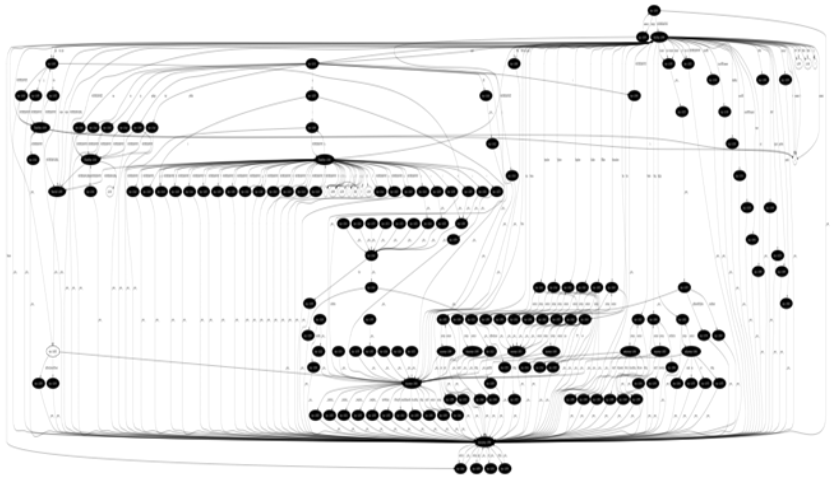
Detecting Reference Leaks

[Barth et al.]

- Instrument WebKit's JavaScript engine with calls to heap analysis library
  - On object creation, reference, and destruction
- Goal: detect references between two contexts
- Sample heap graphs



Empty page



google.com (not much JS there)

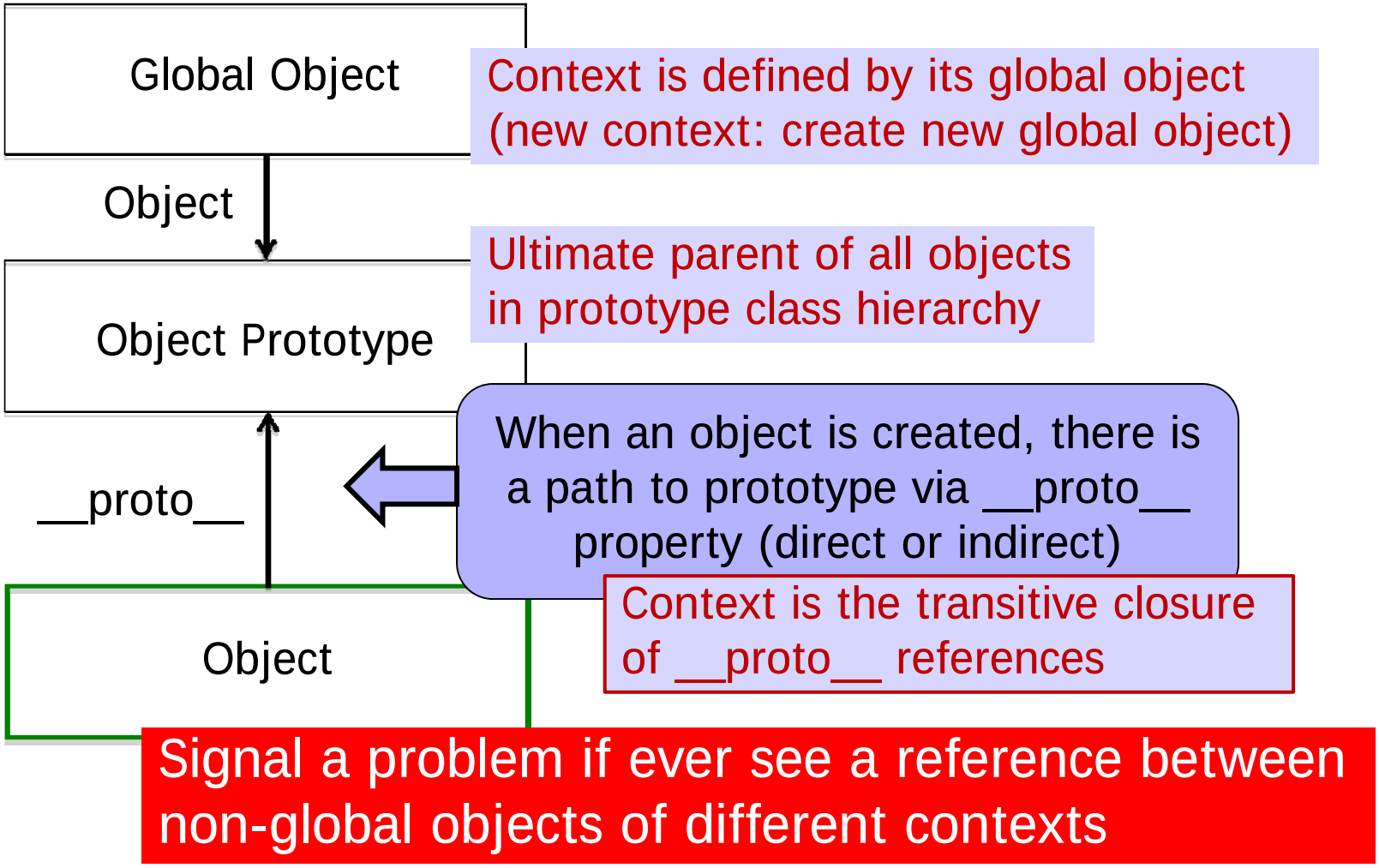
Heap Graph Statistics

[Barth et al.]

- Empty page
  - 82 nodes, 170 edges
- google.com
  - 384 nodes, 733 edges
- store.apple.com/us
  - 5332 nodes, 11691 edges
- gmail.com
  - 55106 nodes, 133567 edges

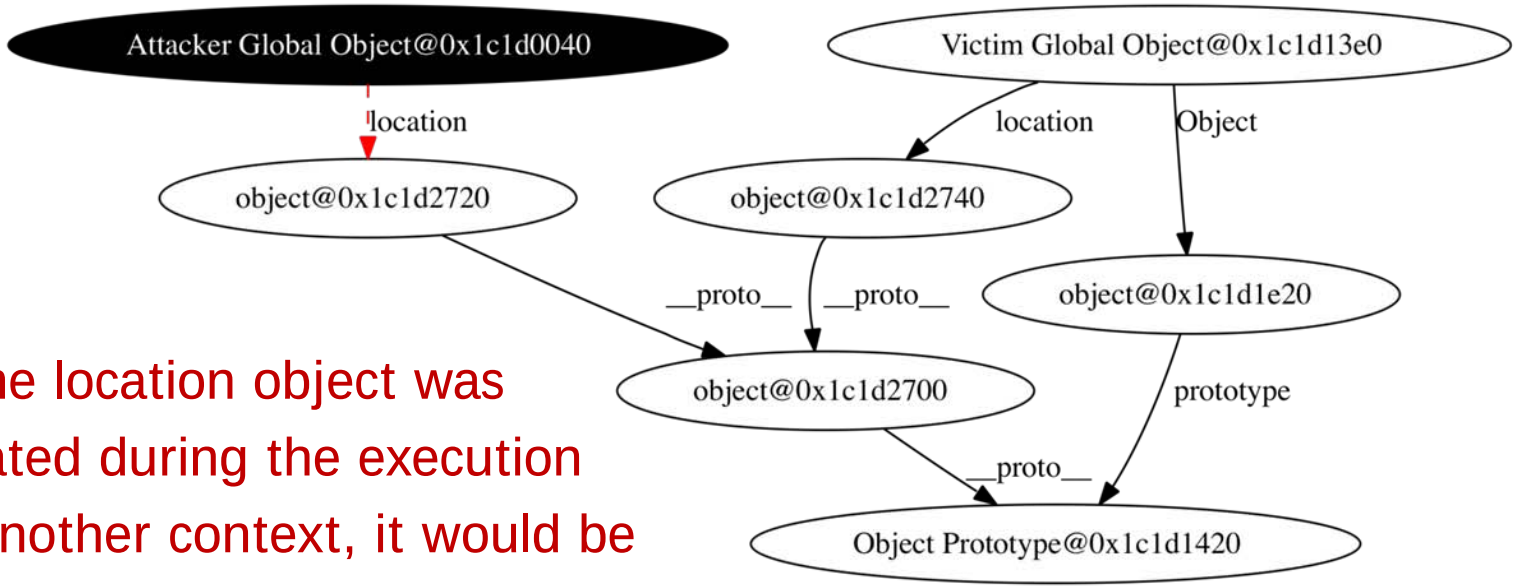
Computing JavaScript Contexts

[Barth et al.]



Example Vulnerability in WebKit

[Barth et al.]



If the location object was created during the execution of another context, it would be created with the wrong Object prototype.

Attacker's object can then redefine the behavior of functions, such as toString, that apply to all Objects created in the other context, so that they execute arbitrary JavaScript.

Solution

- Add access control to JavaScript references
  - get and put: check that context matches
- 2% overhead
  - Inline caching helps: when a property is looked up for the first time, look up in hash table and record offset; subsequent accesses use recorded offset directly
    - If offset is available, no need for access control checks (why?)
  - 10% overhead without caching
- See “Cross-Origin JavaScript Capability Leaks” for details

Web Browser: the New OS

- Origins are similar to processes
  - One origin should not interfere with another
- Sites often want and need to communicate
  - Google AdSense
    - <script src="http://googlesyndication.com/show_ads.js">
  - Mashups
  - Gadget aggregators - iGoogle, live.com ...
  - To communicate with B, site A must give B full control
    - <script src=http://siteB.com/script.html>
  - Now script from site B runs as if its origin were site A

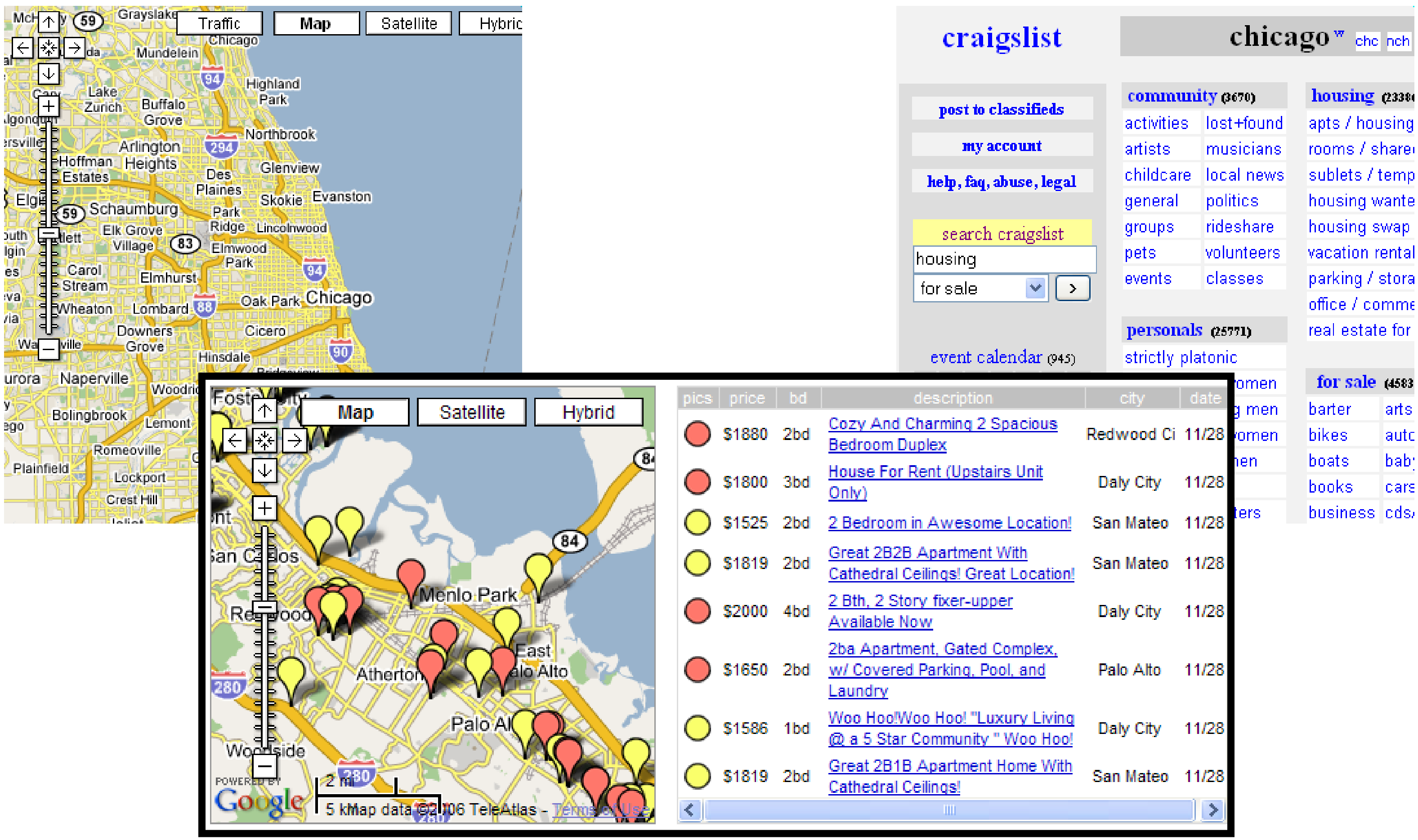
Sending a Cross-Domain GET

- Script can send anywhere
  - This is the basis of cross-site request forgery (XSRF)
- Data must be URL encoded

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

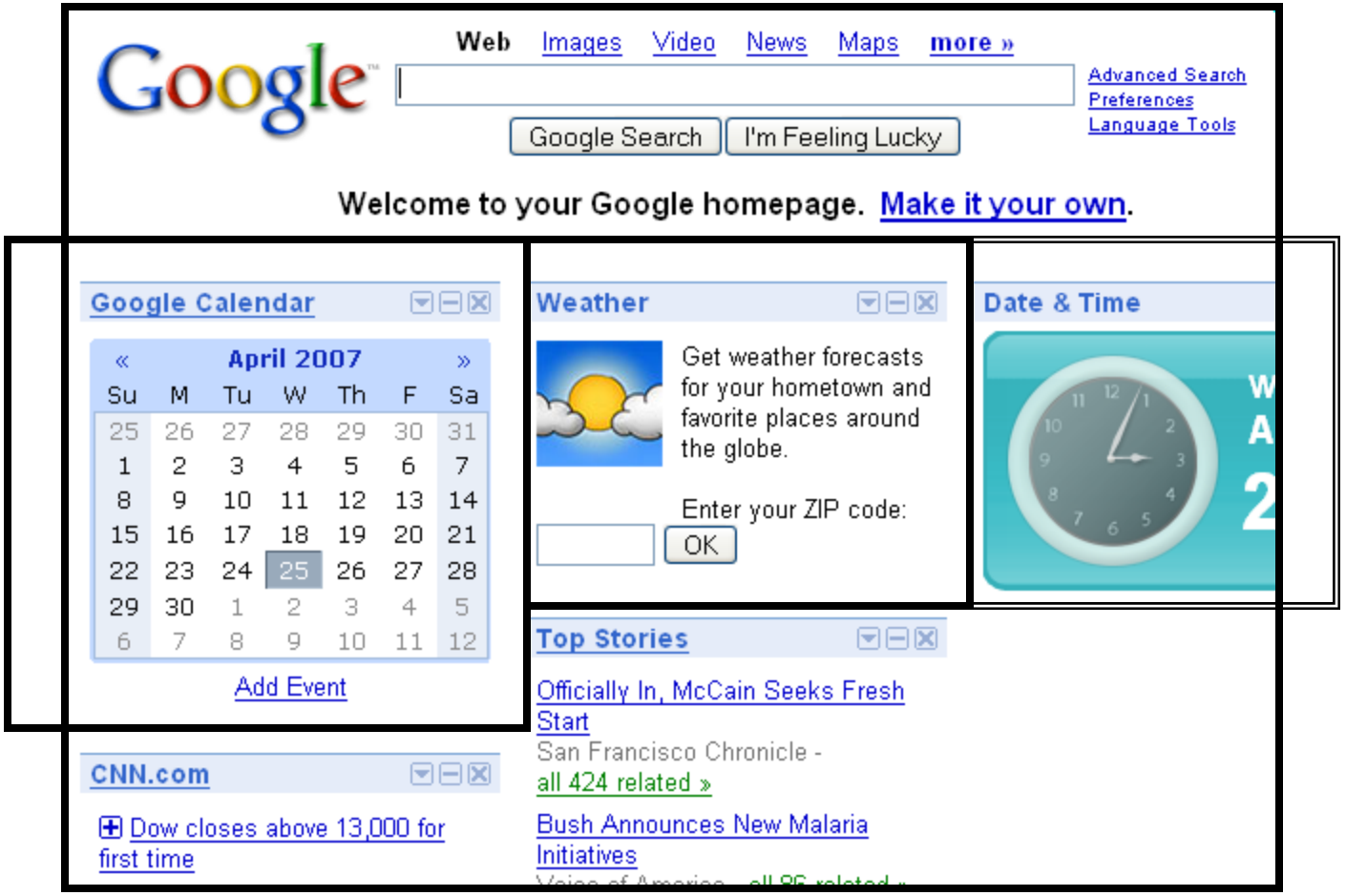
    GET file.cgi?foo=1&bar=x%20y HTTP/1.1
- Can’t send to some restricted ports
  - For example, port 25 (SMTP)
- Can use GET for denial of service (DoS) attacks
  - A popular site can DoS another site [Puppetnets]

Mashups

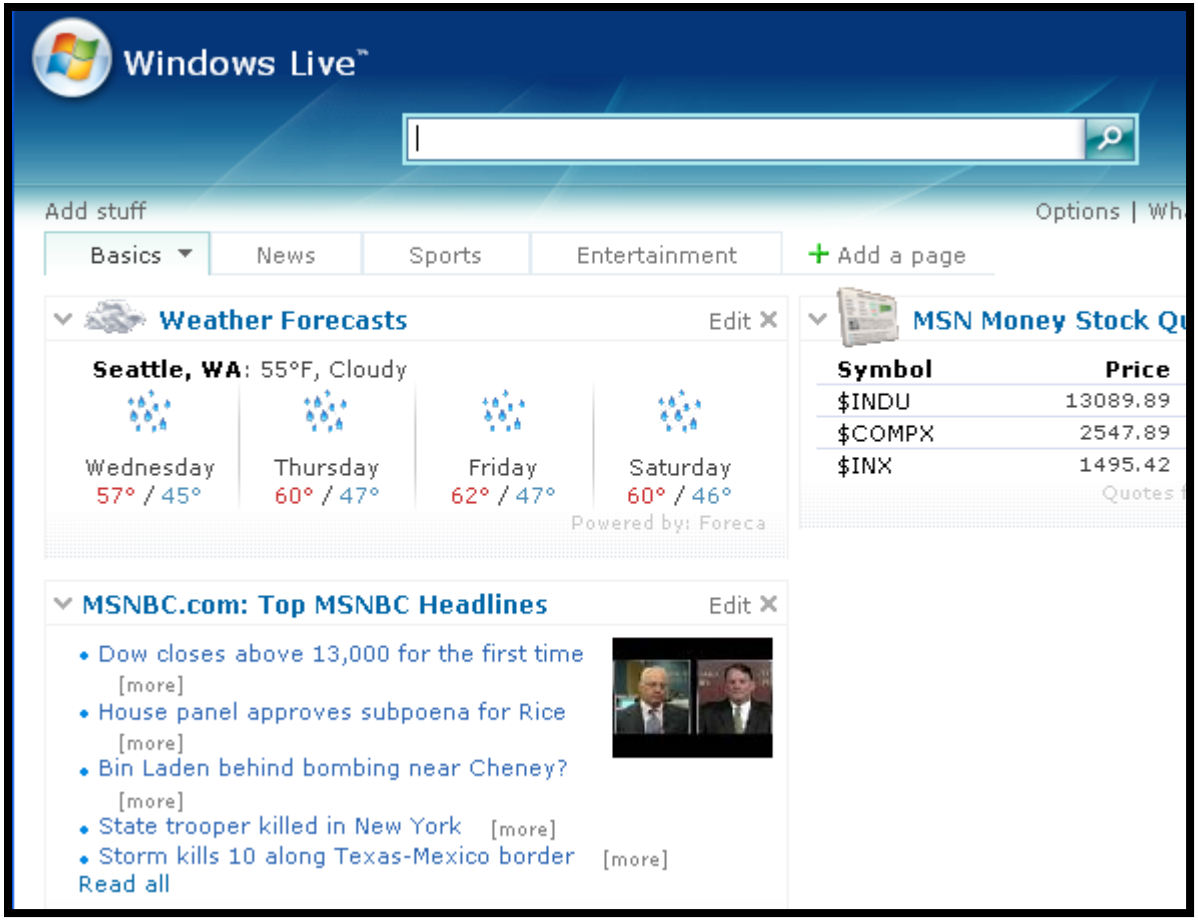


Traffic
Map
Satellite
Hybrid
Mundelein
Highland Park
Northbrook
Arlington Heights
Glenview
Des Plaines
Skokie
Evanston
Schaumburg
Elk Grove Village
Park Ridge
Lincolnwood
Elmwood Park
Carol Stream
Elmhurst
Wheaton
Lombard
Oak Park
Chicago
Downers Grove
Cicero
Hinsdale
Naperville
Bolingbrook
Lemont
Romeoville
Plainfield
Lockport
Crest Hill
craigslist
chicago
post to classifieds
my account
help, faq, abuse, legal
search craigslist
housing
for sale
event calendar (945)
community (3670)
activities
lost+found
artists
musicians
childcare
local news
general
politics
groups
rideshare
pets
volunteers
events
classes
personals (25771)
strictly platonic
apts / housing
sublets / temp
vacation rental
barter
bikes
boats
books
business
Map
Satellite
Hybrid
Menlo Park
East Palo Alto
Atherton
Palo Alto
Woodside
Map data ©2006 TeleAtlas - Terms of Use
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

iGoogle


Google
Web Images Video News Maps more »
Advanced Search
Preferences
Language Tools
Google Search
I'm Feeling Lucky
Welcome to your Google homepage. Make it your own.
Google Calendar
« April 2007 »
Su M Tu W Th F Sa
25 26 27 28 29 30 31
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 1 2 3 4 5
6 7 8 9 10 11 12
Add Event
Weather
Get weather forecasts for your hometown and favorite places around the globe.
Enter your ZIP code:
OK
Date & Time
Top Stories
Officially In, McCain Seeks Fresh Start
San Francisco Chronicle -
all 424 related »
Bush Announces New Malaria Initiatives
CNN.com
Dow closes above 13,000 for first time

Windows Live.com



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?
- Security indicator (lock icon)
  - securityIndicator(W) - is it displayed for window W?

Common Misunderstanding

- Often simply stated as “same-origin policy”
  - This usually just refers to the canScript relation
- Full policy of current browsers is complex
  - Evolved via “penetrate-and-patch”
  - Different features evolved slightly different policies
- Common scripting and cookie policies
  - canScript considers: scheme, host, and port
  - canReadCookie considers: scheme, host, and path
  - canWriteCookie considers: host

Cross-Frame Scripting

- canScript(A,B) - only if Origin(A) = Origin(B)
  - Basic same-origin policy, where origin is the scheme, host and port from which the frame was loaded
- What about frame content?
- Some browsers allow any frame to navigate any other frame

SOP Examples

- Suppose the following HTML is hosted at site.com
- Disallowed access

```
<iframe src="http://othersite.com"></iframe>
alert( frames[0].contentDocument.body.innerHTML )
alert( frames[0].src )
```

- Allowed access

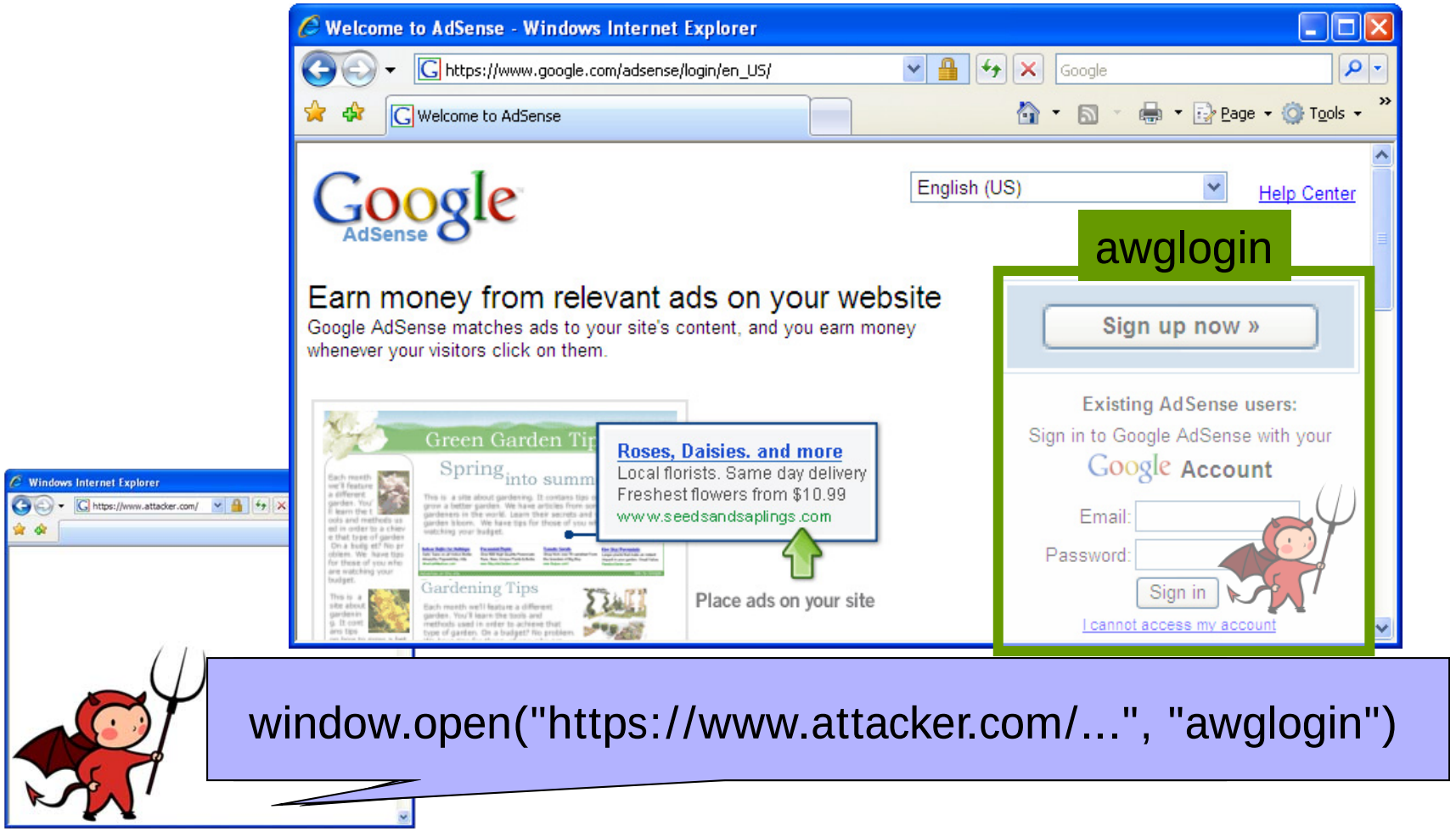
```
<img src="http://othersite.com/logo.gif">
alert( images[0].height )
```

or

```
frames[0].location.href = “http://mysite.com/”
```

Navigating child frame is allowed, but reading frame[0].src is not

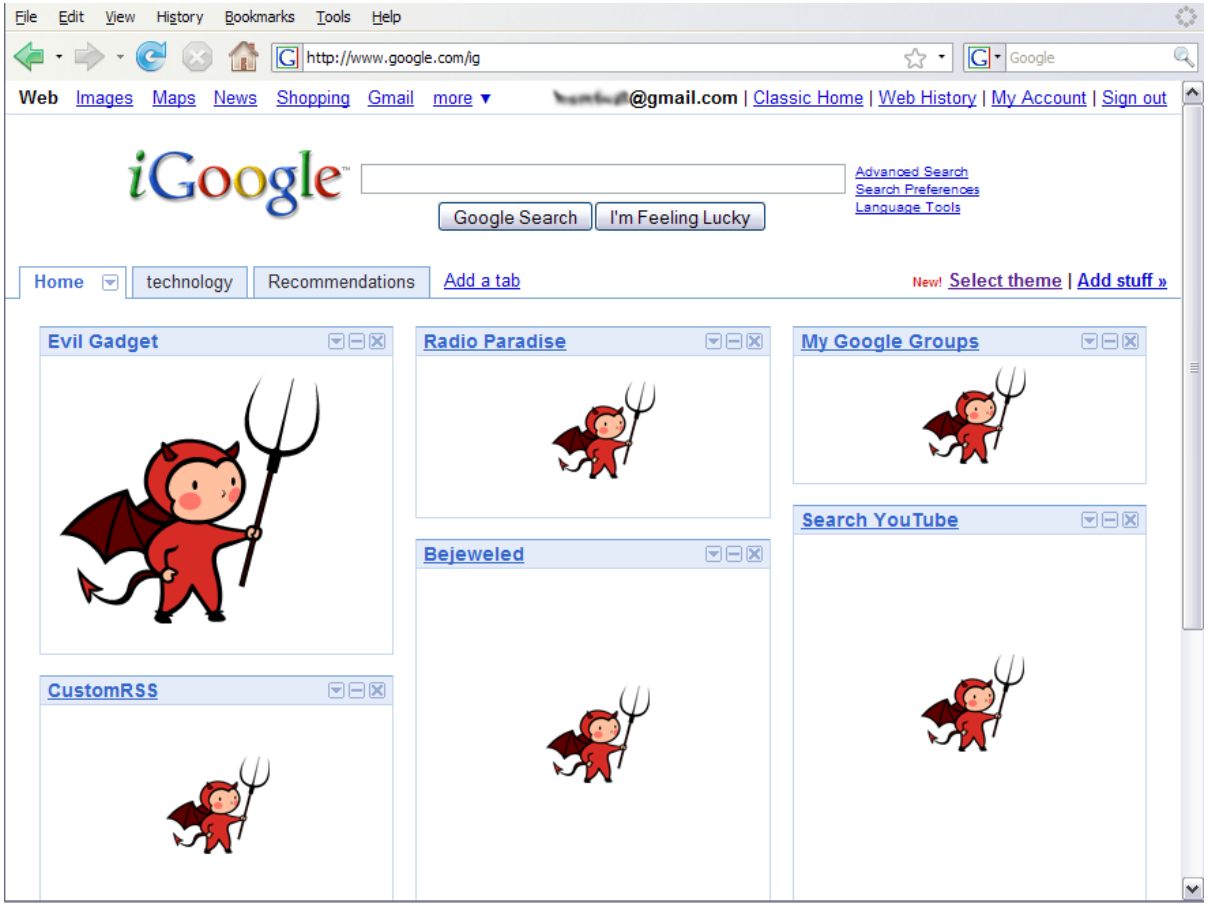
Guninski Attack



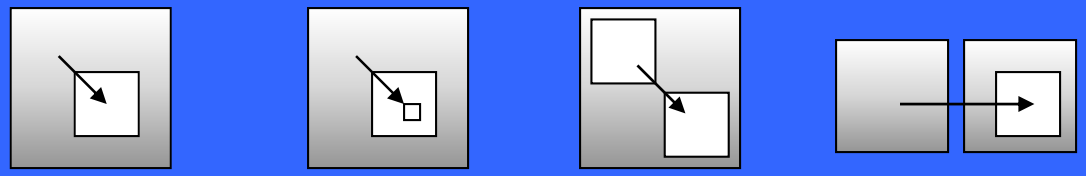
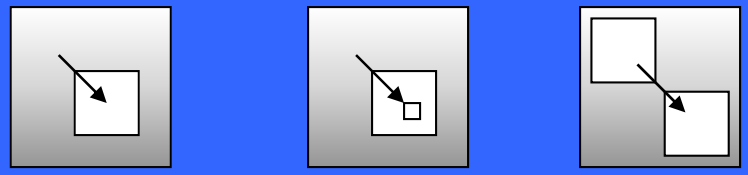
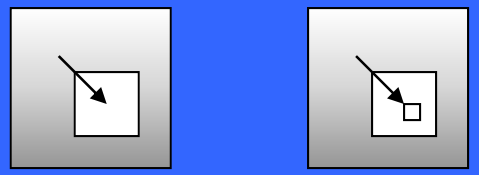
If bad frame can navigate good frame, attacker gets password!

Gadget Hijacking in Mashups

Gadget Hijacking



Possible Frame Navigation Policies

Policy	Behavior
Permissive	
Window	
Descendant	
Child	

Implemented Browser Policies

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

Principle: Pixel Delegation

- Frames delegate screen pixels
  - Child cannot draw outside its frame
  - Parent can draw over the child's pixels
- Navigation similar to drawing
  - Navigation replaces frame contents
  - "Simulate" by drawing over frame
- Policy ought to match pixel delegation
  - Navigate a frame if can draw over the frame

Best Solution: Descendant Policy

- Best security / compatiblity trade-off
  - Security: respects pixel delegation
  - Compatibly: least restrictive such policy
- Implementation (Adam Barth, Collin Jackson)
  - Wrote patches for Firefox and Safari
  - Wrote over 1000 lines of regression tests
- Deployment
  - Apple released patch as security update
  - Mozilla implemented in Firefox 3

Frame Communication

- If frames provide isolation, how can they communicate?
- Desirable properties of interframe communication
  - Confidentiality
  - Integrity
  - Authentication

Fragment Identifier Messaging

- Send information by navigating a frame
  - http://gadget.com/**#hello**
- Navigating to fragment doesn't reload frame
  - No network traffic, but frame can read its fragment
- Not a secure channel
  - Confidentiality ✓
  - Integrity ✓
  - Authentication ✗

D. Thorpe, Secure Cross-Domain Communication in the Browser
http://msdn2.microsoft.com/en-us/library/bb735305.aspx

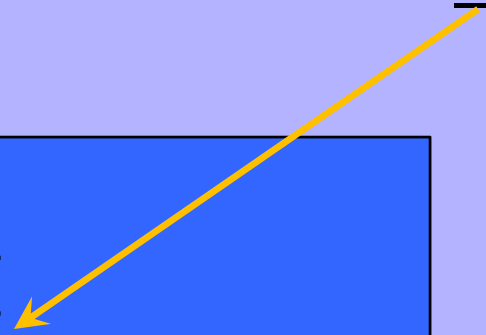
Identifier Messaging: Example

Host page: foo.com/main.html

```
function sendData() {
    iframe.src = “http://bar.com/receiver.html#data_here”;
}
```

iframe: bar.com/receiver.html

```
window.onLoad = function () {
    data = window.location.hash;
}
```



Problems and Limitations

- No ack that the iframe received the data
- Message overwrites
  - Host doesn't know when the iframe is done processing a message... when is it safe to send the next message?
- Capacity limits
  - URL length limit varies by browser family
- Data has unknown origin
- No replies
- Loss of context
  - Page is reloaded with every message, losing DOM state

With Return Communication

Host page: foo.com/main.html

```
function sendDataToBar() {
    iframe.src = " http://bar.com/receiver.html#data_here";
}
```

iframe: bar.com/receiver.html

```
window.onLoad = function () {
    data = window.location.hash;
}
function sendDataToFoo(){
    iframe2.src = "http://foo.com/receiver.html#data_here";
}
```

iframe2: foo.com/receiver.html

```
window.onLoad = function () {
    window.parent.parent.receiveFromBar(
      window.location.hash);
}
```

postMessage

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

- Not a secure channel
  - Confidentiality ✗
  - Integrity ✓
  - Authentication ✓

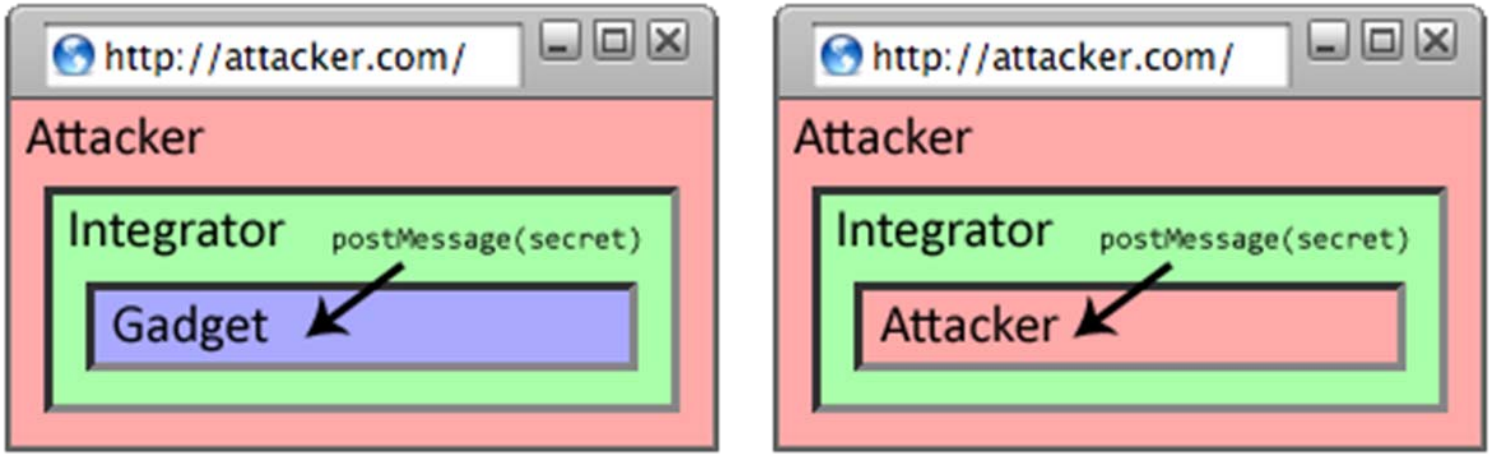
Example of postMessage Usage

```
frames[0].postMessage("Hello world.");
```

```
document.addEventListener("message", receiver);
function receiver(e) {
if (e.domain == "example.com") {
  if (e.data == "Hello world")
    e.source.postMessage("Hello");
  }
}
```

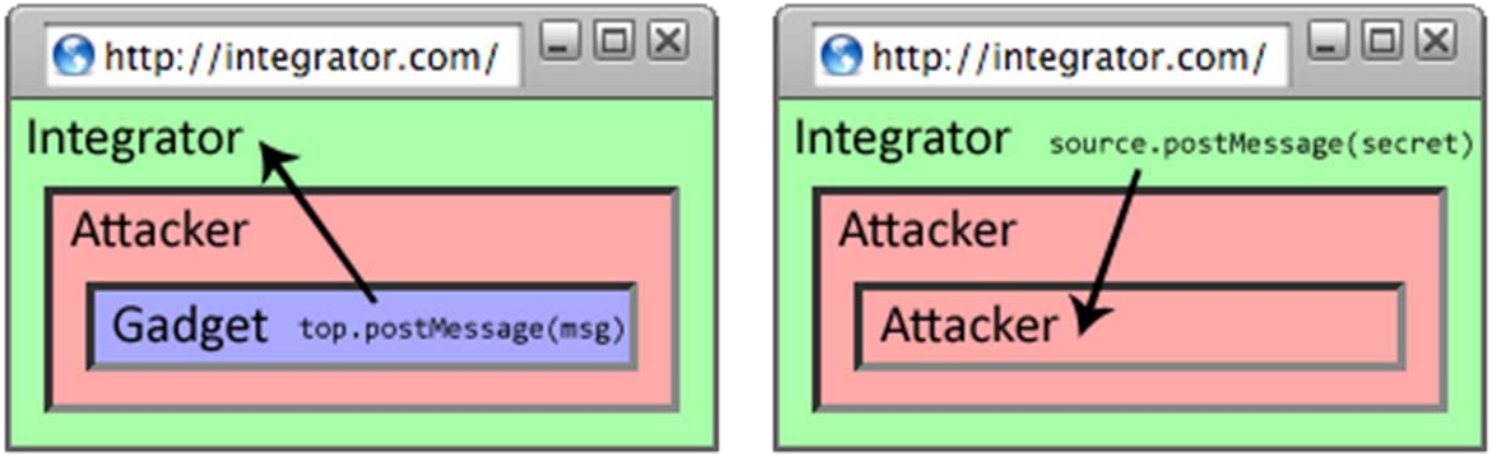
Message Eavesdropping (1)

- Descendant frame navigation policy



Message Eavesdropping (2)

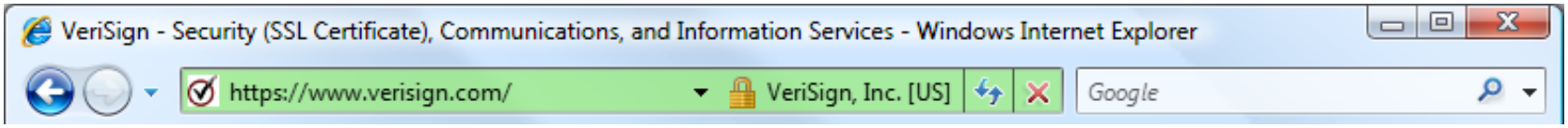
- Works in all navigation policies



Finer-Grained Origins

- Some browser features grant privileges to a subset of documents in an origin
  - Cookie paths
  - Mixed content
    - For example, documents with invalid certificates mixed with documents with valid certificates
- Any “less trusted” document can inject an arbitrary script into a “more trusted” one (why?)
  - Gain the same privileges as the most trusted document in the same origin!

The Lock Icon



- Goal: identify secure connection
  - This is a network security issue
- SSL/TLS is used between client and server to protect against active network attacker
- Lock icon should only be shown when page is secure against network attacker

Checkered History of the Lock

- Positive trust indicator
- Semantics subtle and not widely understood
  - This page is not under the control of an active network attacker (unless the principal named in the location bar has chosen to trust the attacker)
- Innovation required in user interface design
  - Lock icon largely ignored by users
  - Innovations require browser accuracy in determining whether to show security indicators

Problem with Embedded Content

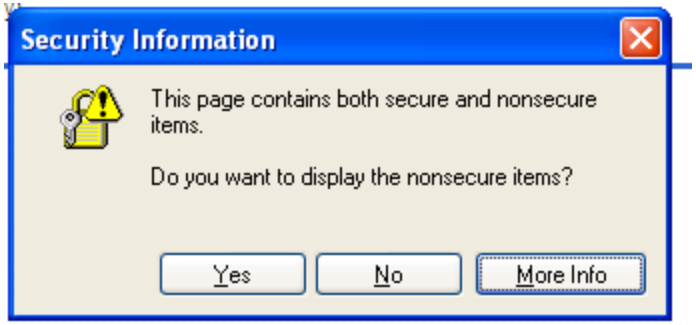
Show lock icon if ...

- Page retrieved over HTTPS
- Every embedded object retrieved over HTTPS
  - Firefox allows HTTP images, but it’s a known bug
- Every frame would have shown lock icon

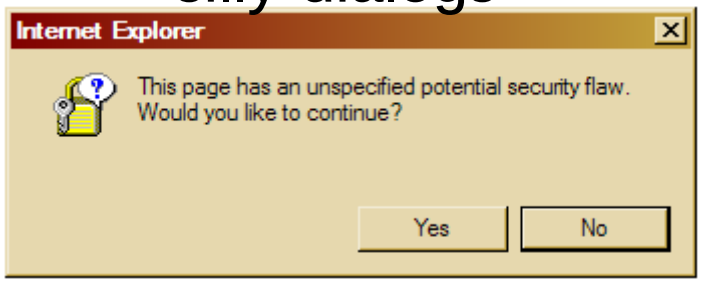
Mixed Content: HTTP and HTTPS

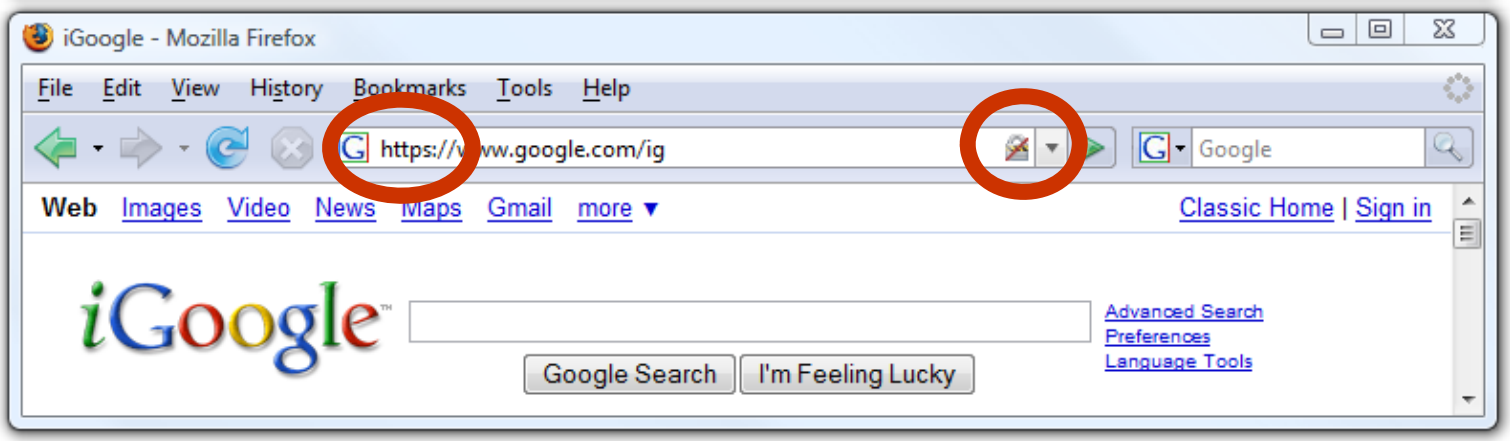
- Page loads over HTTPS, but contains content over HTTP
- IE: displays mixed-content dialog to user
  - Flash files over HTTP are loaded with no warning (!)
  - Flash can script the embedding page!
- Firefox: red slash over lock icon (no dialog)
  - Flash files over HTTP do not trigger the slash
- Safari: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

Mixed Content: UI Challenges



silly dialogs





Mixed Content and Network Attacks

- Banks: after login, all content served 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:

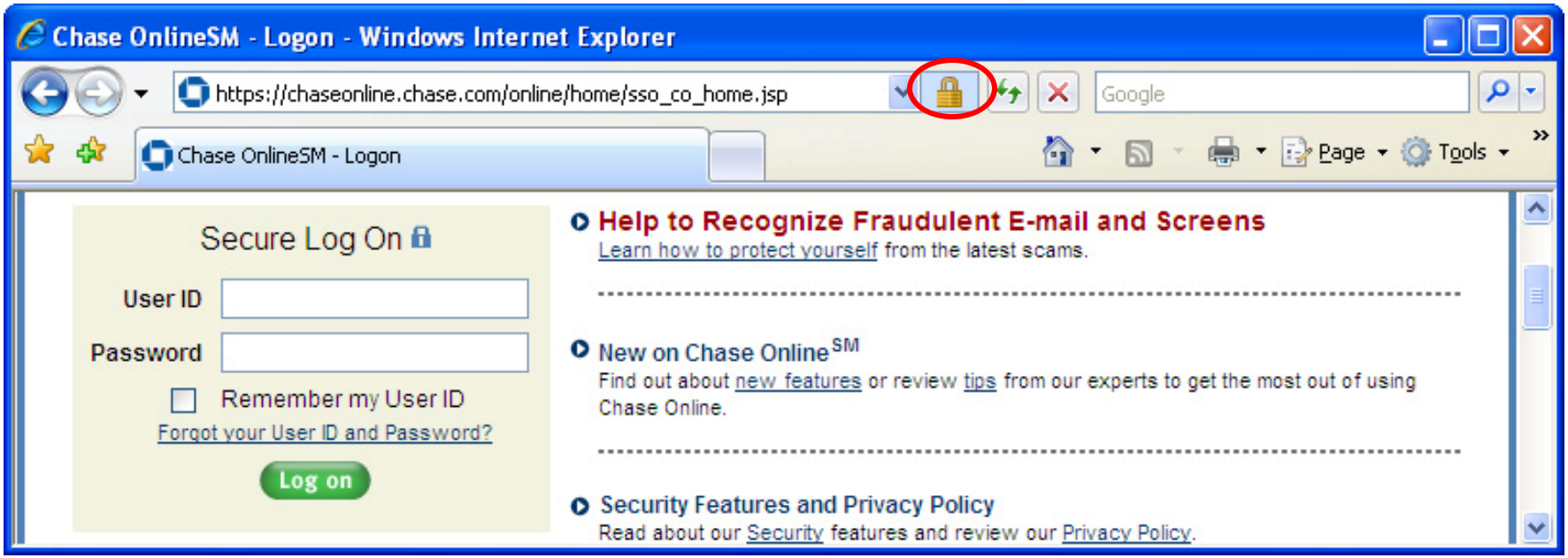
```
<script src=//www.site.com/script.js> </script>
```

  - Served over the same protocol as embedding page

Mixed Content Issues

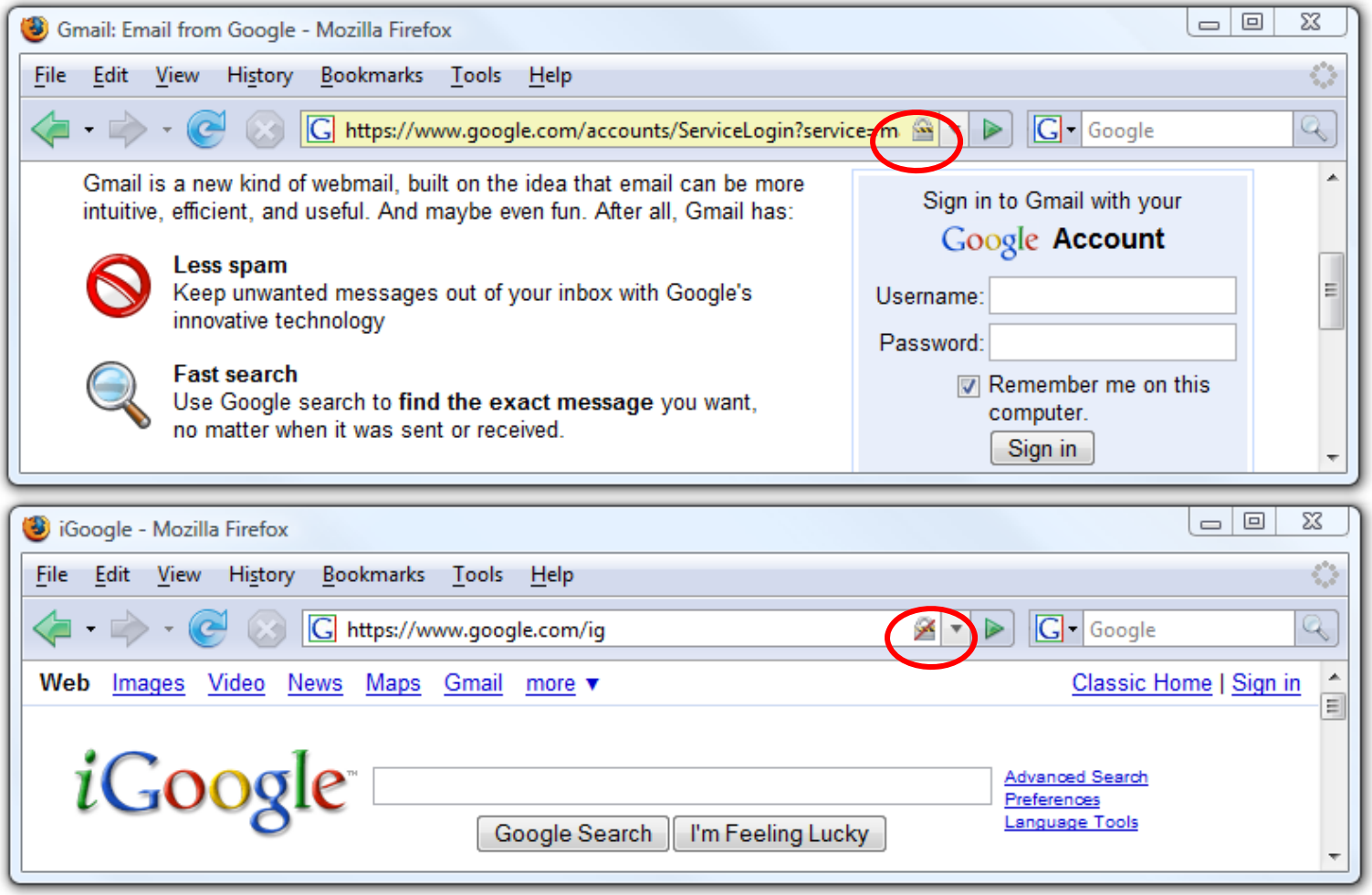
- All browsers fail to account for canScript
  - One fix: Safelock browser extension revokes the ability to dispay the lock icon from all documents in the same origin as an insecure document
- Lots of other bugs
  - Fail to detect insecure SWF movies (IE, Firefox)
  - Navigation forgets mixed content (Firefox)
  - Firefox architecture make detection difficult

Example of a Vulnerability

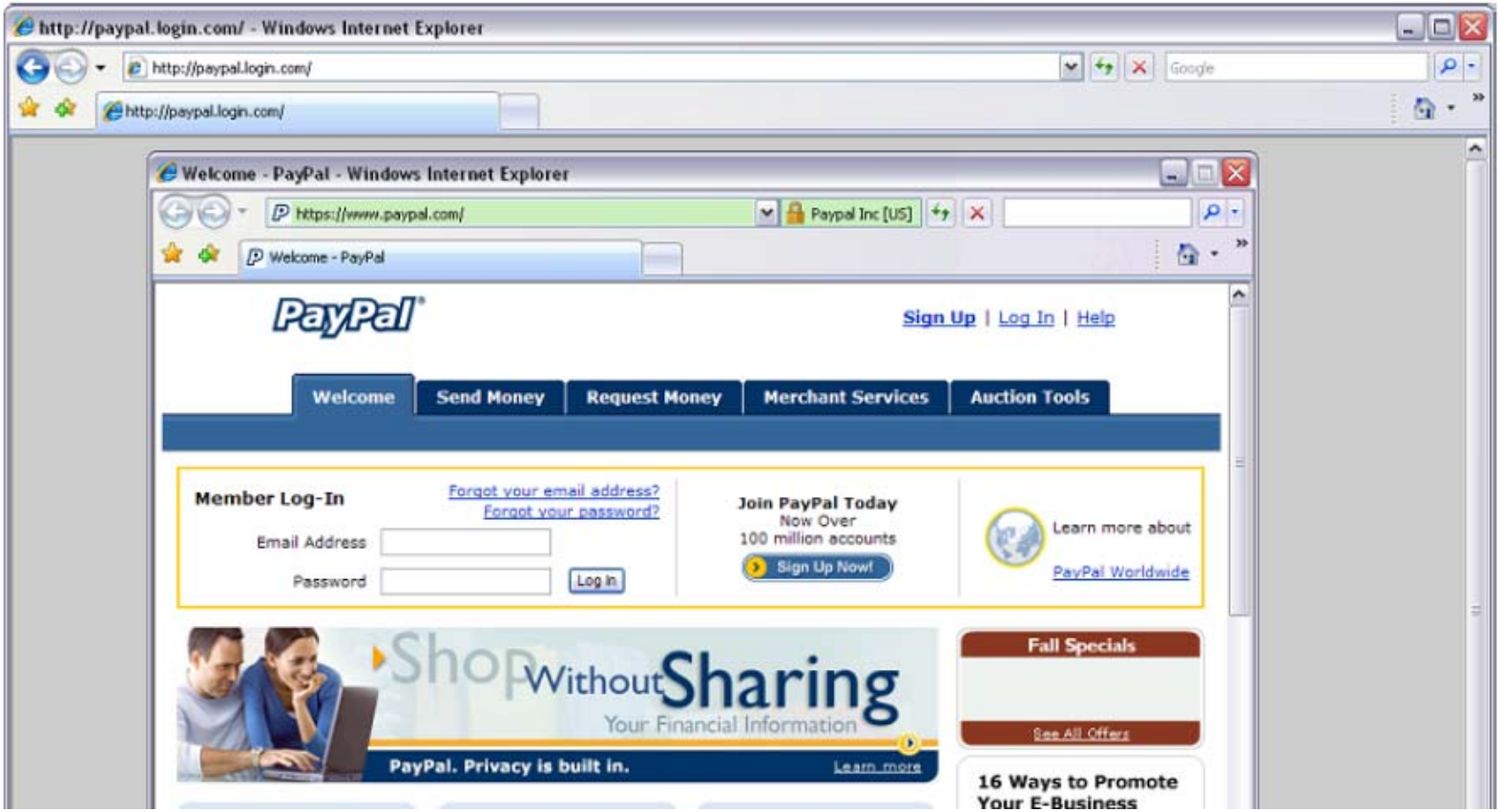


Chase used a SWF movie served over HTTP to perform authentication on the banking login page – active network attacker can steal password!

Origin Contamination



Picture-in-Picture Attacks

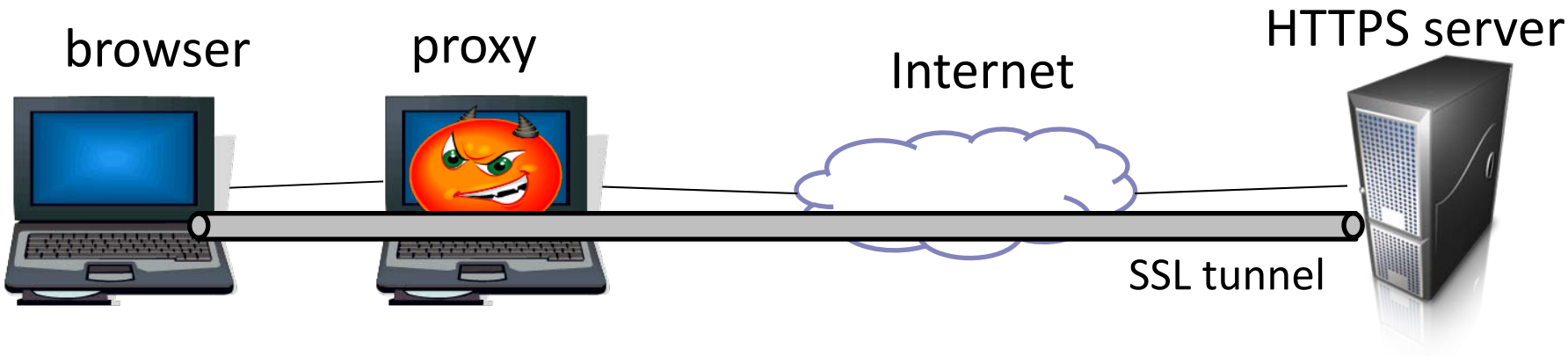


Trained users are more likely to fall victim to this!

SSL/TLS and Its Adversary Model

[Chen et al.]

- HTTPS: end-to-end secure protocol for Web
- Designed to be secure against man-in-the-middle (MITM) attacks

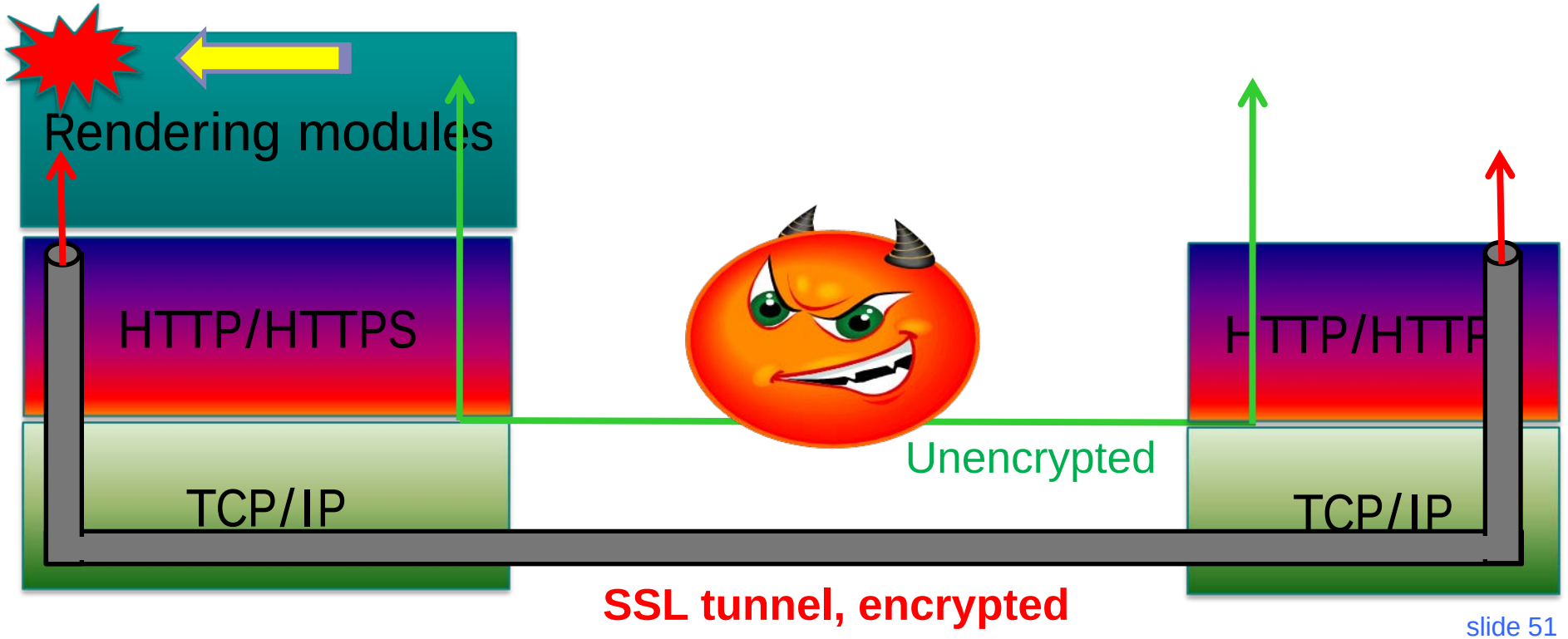


- HTTPS provides encryption and integrity checking

PBP: Pretty-Bad-Proxy

[Chen et al.]

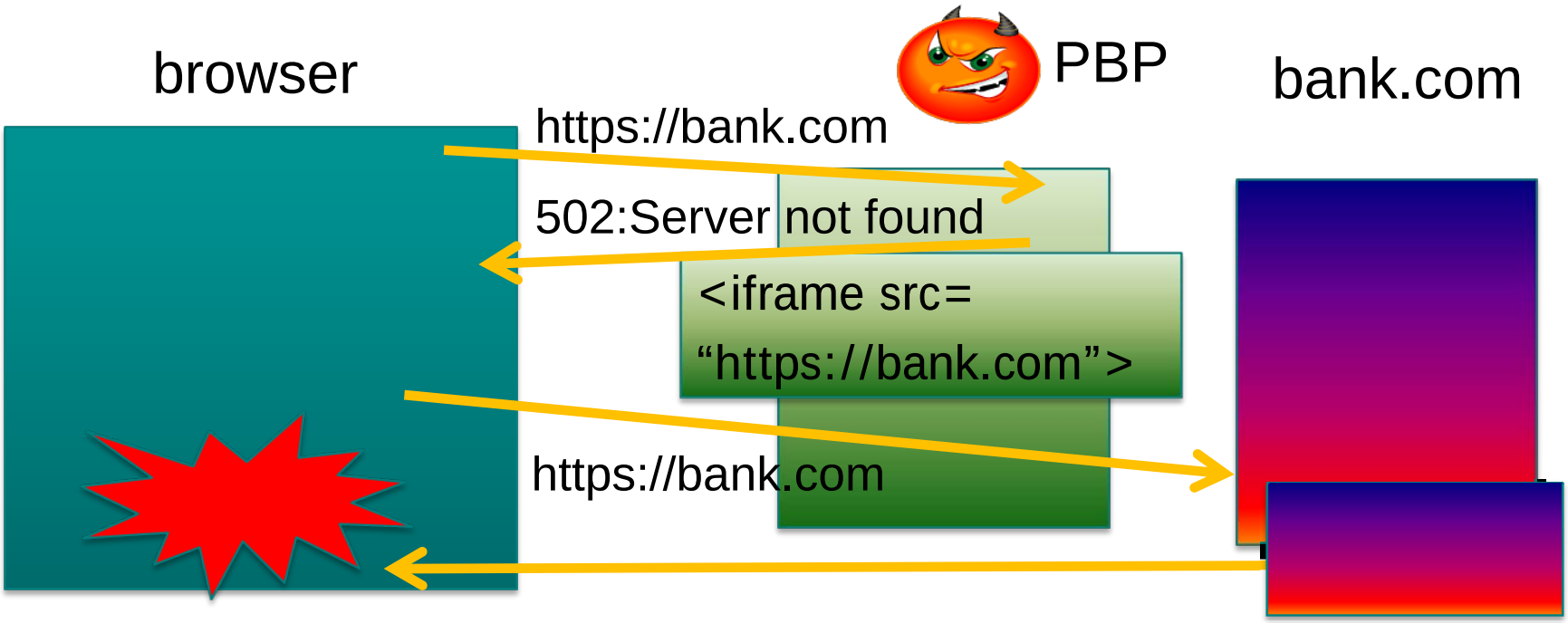
- Bad proxy can exploit browser bugs to render unencrypted, potentially malicious content in the context of an HTTPS session!



Attack #1: Error Response

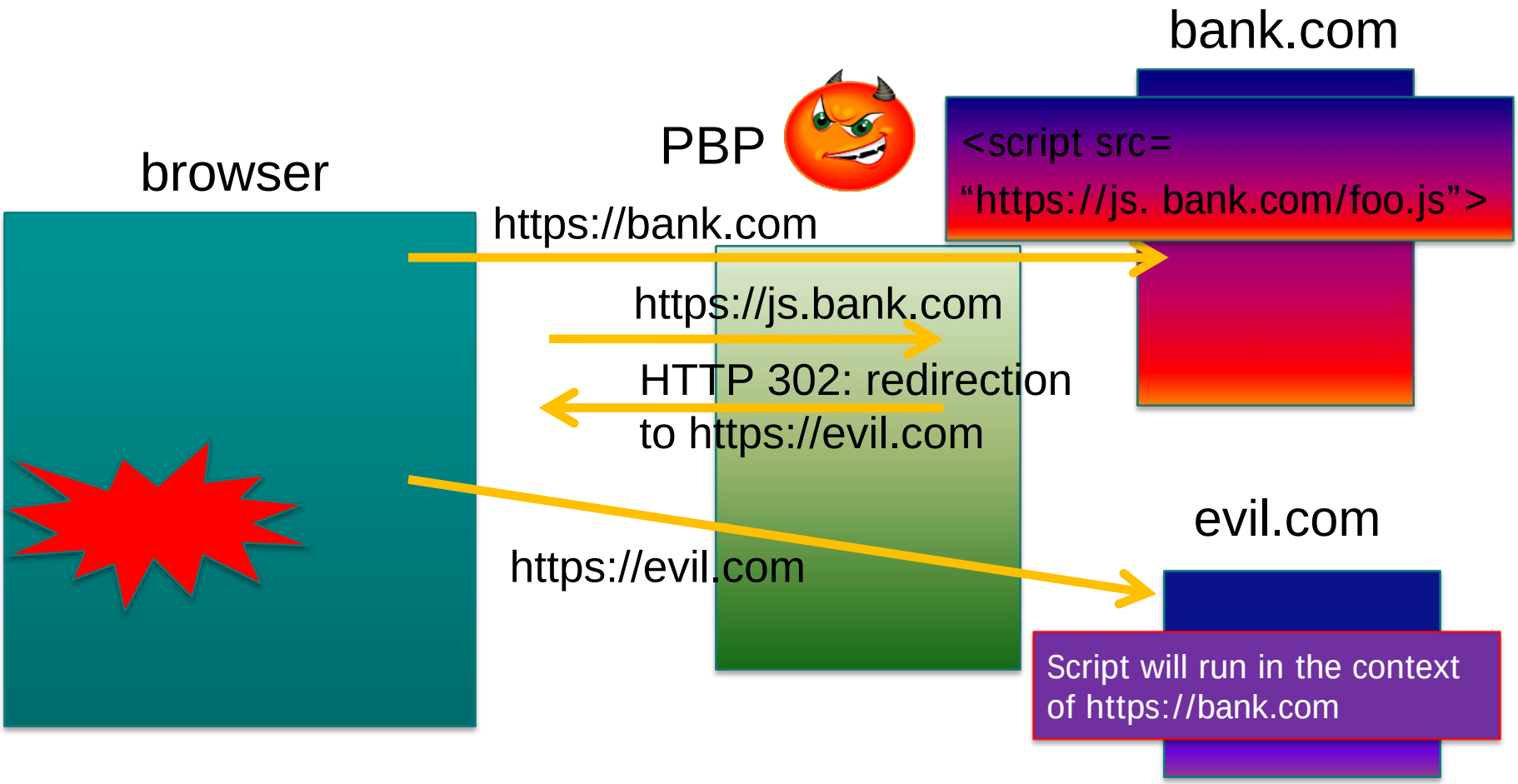
[Chen et al.]

- Proxy error page: 502, other 4xx/5xx response
- Script in error page runs in HTTPS context!



Attack #2: Redirection (3XX)

[Chen et al.]



Attack #3: HPIHSL Pages

[Chen et al.]

- Many websites provide both HTTP and HTTPS services
  - Sensitive pages: HTTPS only
    - Login, checkout, etc.
  - Non-sensitive pages: intended for HTTP
    - For example, merchandise pages
  - Non-sensitive pages often accessible through HTTPS
    - HPIHSL: HTTP-intended-but-HTTPS-loadable
- What's wrong with HPIHSL pages?
  - They often import scripts through HTTP ...
  - ... these scripts will run in HTTPS context

Browsers Warn About This, Right?

- Browsers warn about loading HTTP resources in HTTPS contexts

- The objective of this detection logic is to determine the appearance of the address bar
  - Address bar only concerns the top-level page!

Bypassing Detection Logic

[Chen et al.]

- Using an HTTPS iframe in an HTTP top-level page

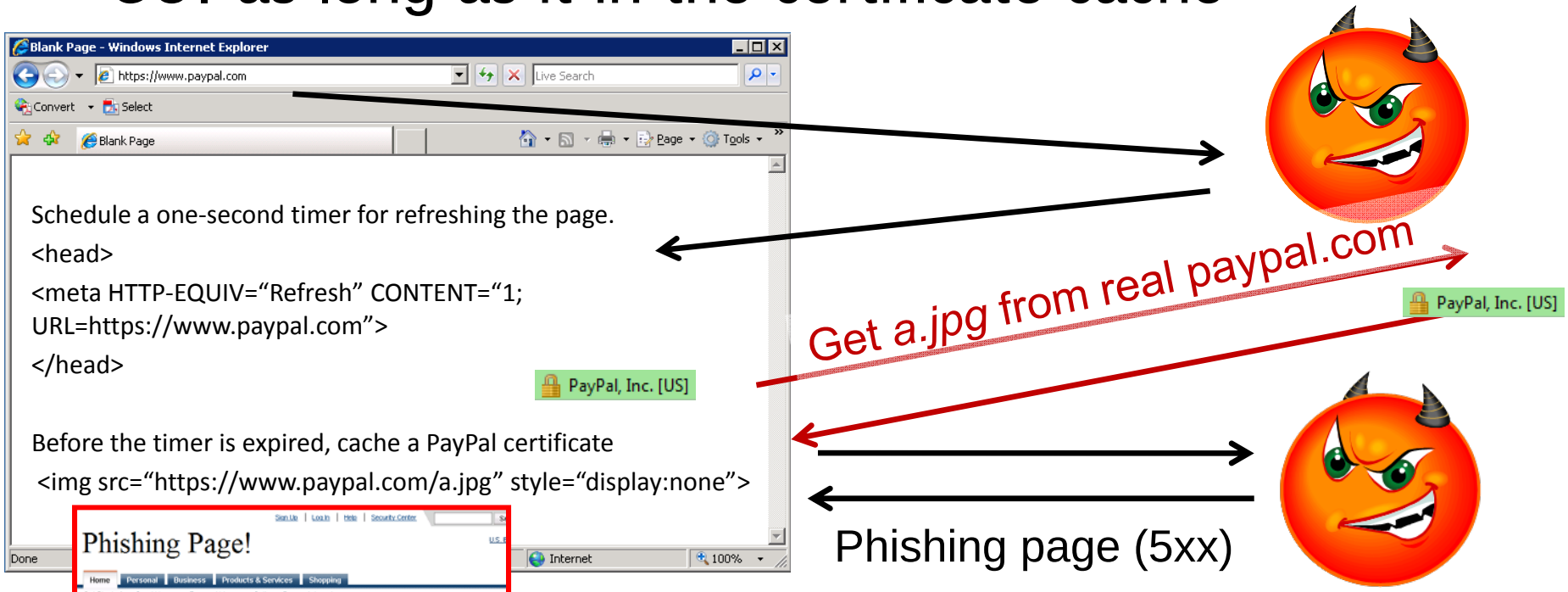
Prevalence of HPIHSL Pages

- Chen et al. show 12 major websites with HPIHSL pages that import scripts
  - Online shopping sites
  - Banks, credit card companies
  - Open-source projects management site
  - Top computer science departments
  - Even the home domain of a leading certificate authority
- You cannot trust their SSL!

Attack #4: Visual Context

[Chen et al.]

- IE, Opera, Chrome display a certificate on the GUI as long as it in the certificate cache



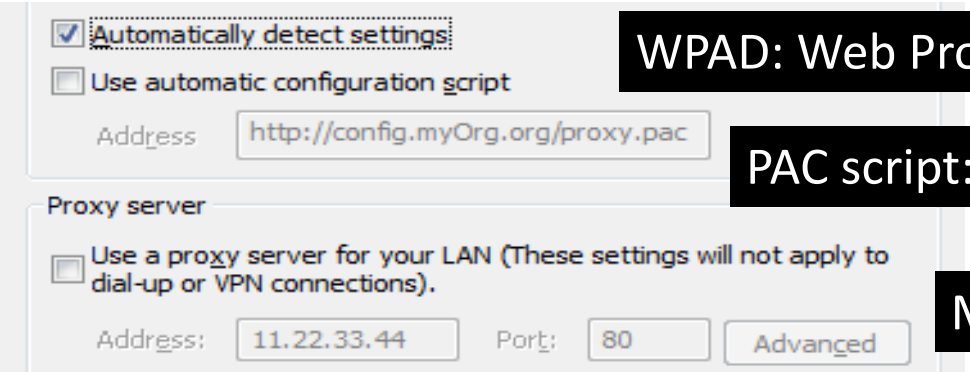
Perfect GUI spoofing attack!

Fresh browser, single tab, address bar input

Feasibility of Exploitation

[Chen et al.]

- Malicious proxy
  - Who uses proxies? Corporate and university networks, hospitals, hotels, third-party free proxies...
  - Security of HTTPS depends on proxy's security!
- Malicious link-level attacker acting as a proxy
  - Can sniff the network at the link layer
  - Browser has its proxy capability turned on



WPAD: Web Proxy Auto Discovery

PAC script: Proxy Auto Config script

Manual configuration

Vulnerability Status (May 2009)

[Chen et al.]

	IE 8 (since beta 2)	Firefox 3.0.10	Safari 3.2.2 (or before)	Opera since Dec 2007	Chrome 1.0.154.53
Error-response issue	Fixed	Fixed	Fixed	Fixed	Fixed
Redirection issue	N/A	Fixed	Fixed	Fixed	N/A
HPIHSL issue	Fix suggested for next version	Fix proposed	Acknowledged	Acknowledged	Acknowledged
Cached certificate issue	Fixed	N/A	N/A	Fixed	Fixed