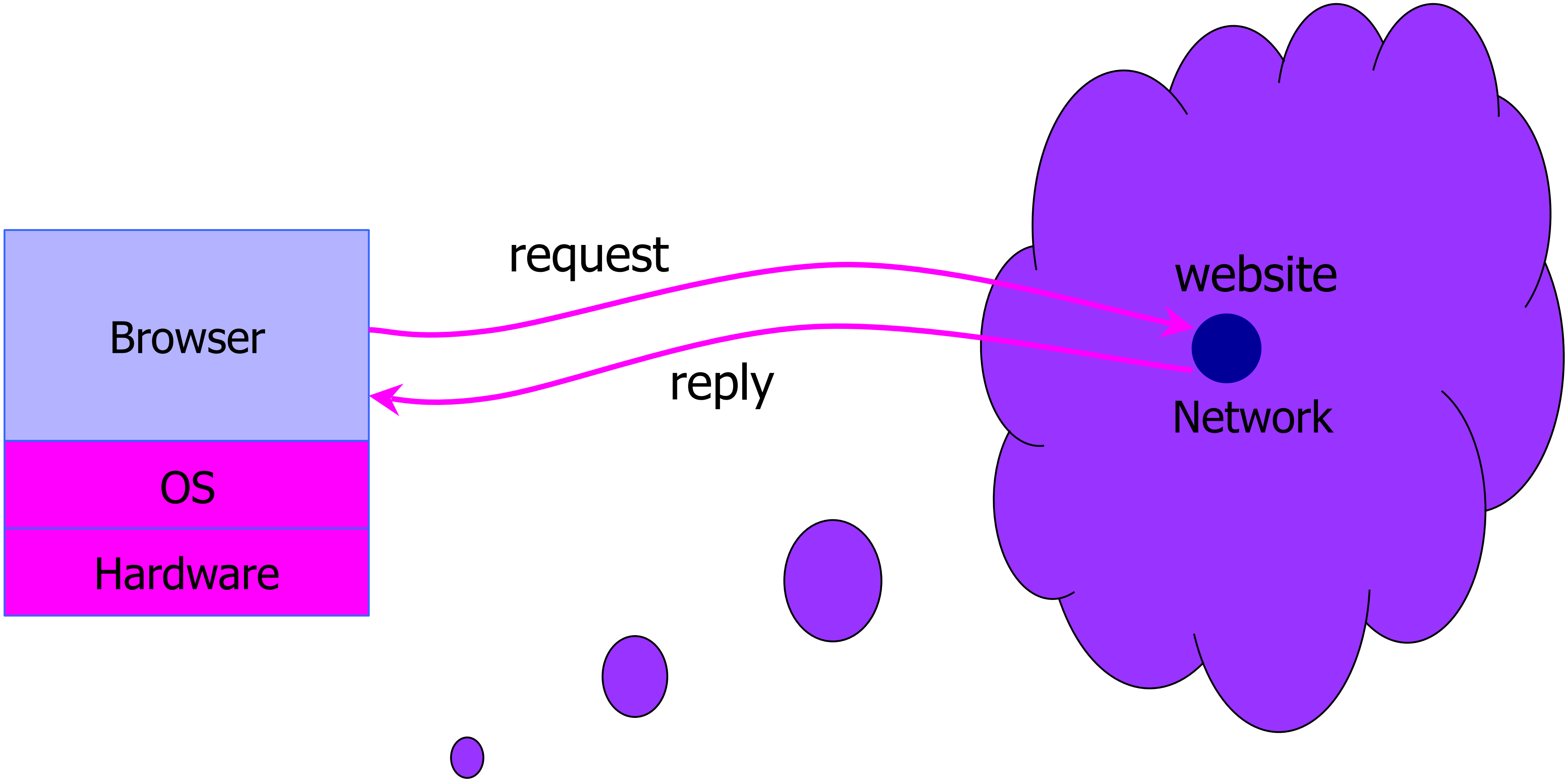
Web Security

The Same Origin Policy

Yan Huang

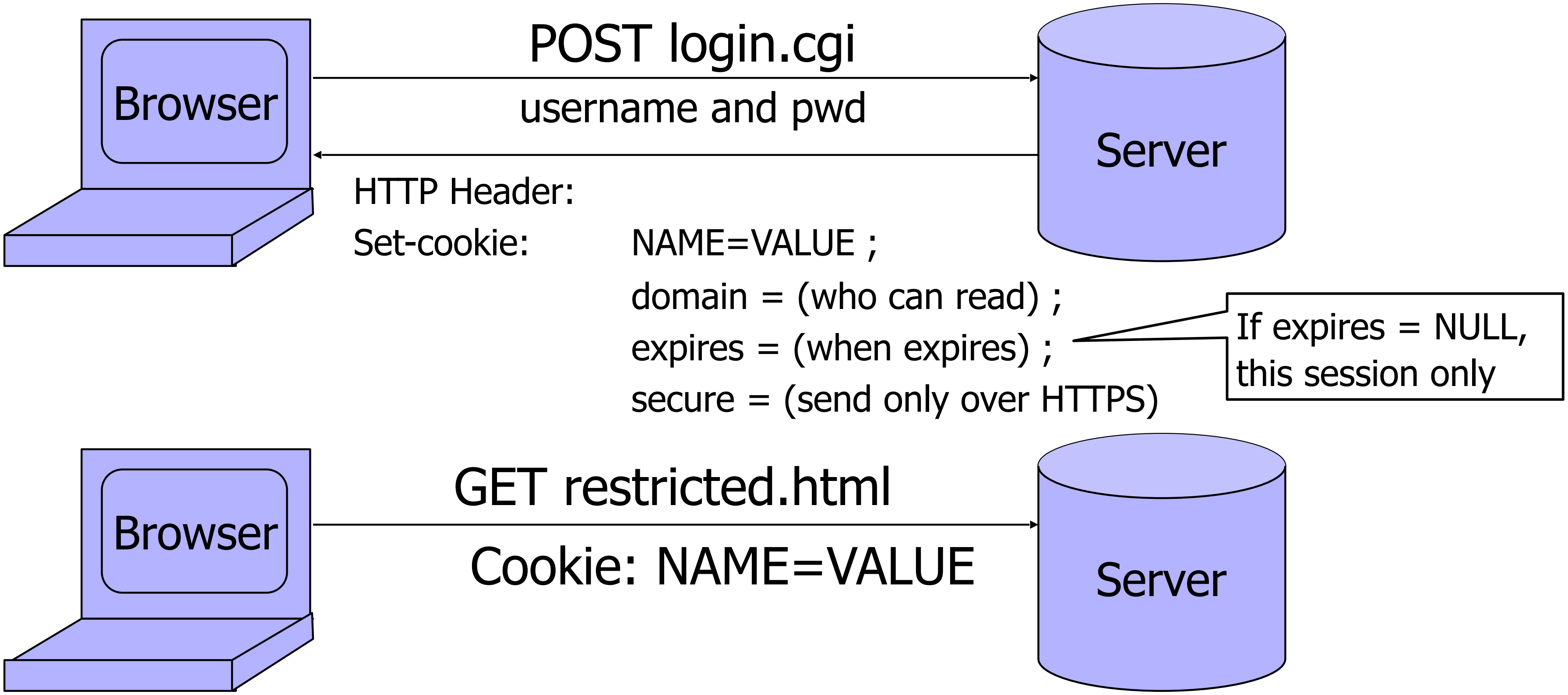
Credits: slides adapted from Stanford and Cornell Tech

Browser and Network



Website Storing Info In Browser

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



Browser

POST login.cgi

username and pwd

Server

HTTP Header:

Set-cookie: NAME=VALUE ;

domain = (who can read) ;

expires = (when expires) ;

secure = (send only over HTTPS)

If expires = NULL, this session only

Browser

GET restricted.html

Cookie: NAME=VALUE

Server

HTTP is a stateless protocol; cookies add state

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

Same Origin Policy

protocol://domain:port/path?params

Same Origin Policy (SOP) for DOM:

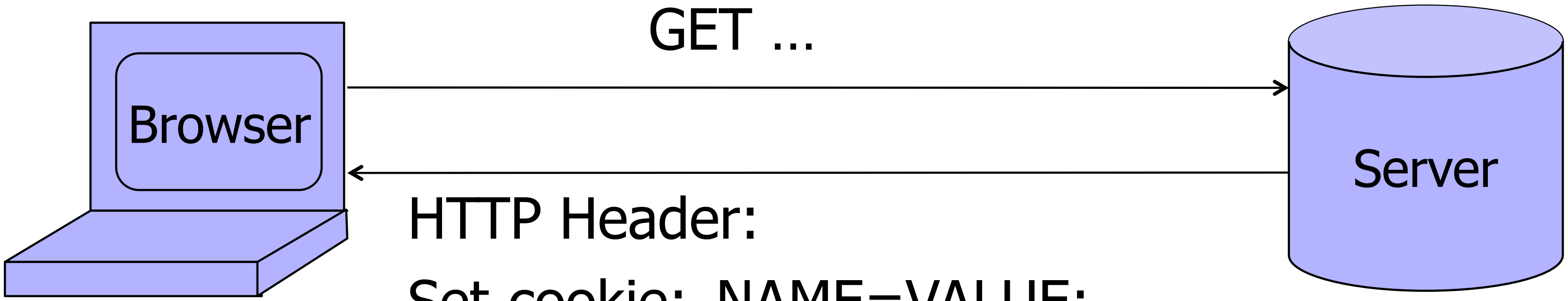
- Origin A can access origin B's DOM if A and B have same **(protocol, domain, port)**

Same Origin Policy (SOP) for cookies:

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

optional

Setting Cookies by Server



HTTP Header:

Set-cookie: NAME=VALUE;

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

scope

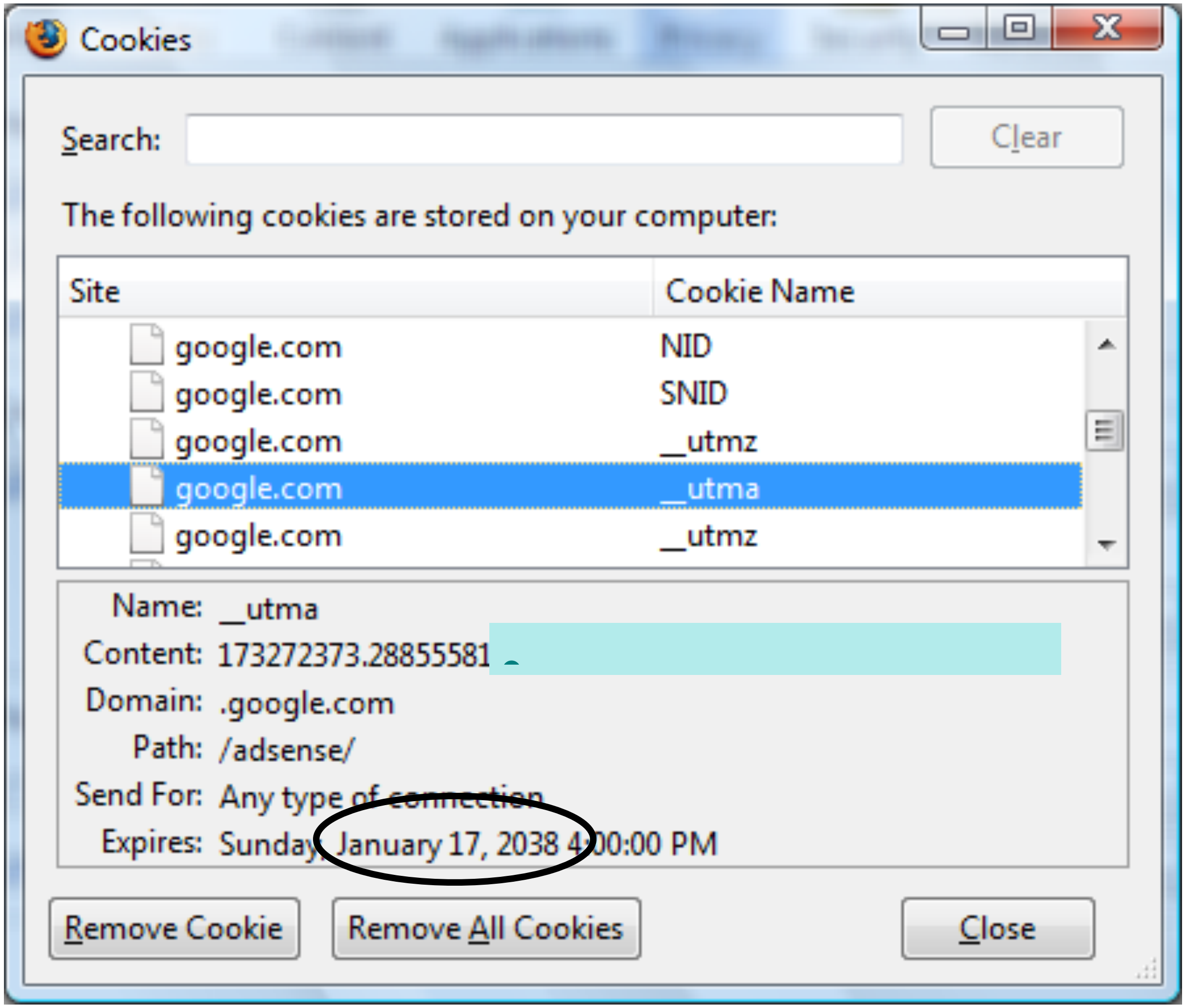
secure = (only send over HTTPS);
expires = (when expires);
HttpOnly

if expires=NULL:
this session only

Delete cookie by setting "expires" to date in past

Default scope is domain and path of setting URL

Viewing Cookies in Browser



The following cookies are stored on your computer:

Site	Cookie Name
google.com	NID
google.com	SNID
google.com	__utmz
google.com	__utma
google.com	__utmz

Name: __utma

Content: 173272373.28855581 [redacted]

Domain: .google.com

Path: /adsense/

Send For: Any type of connection

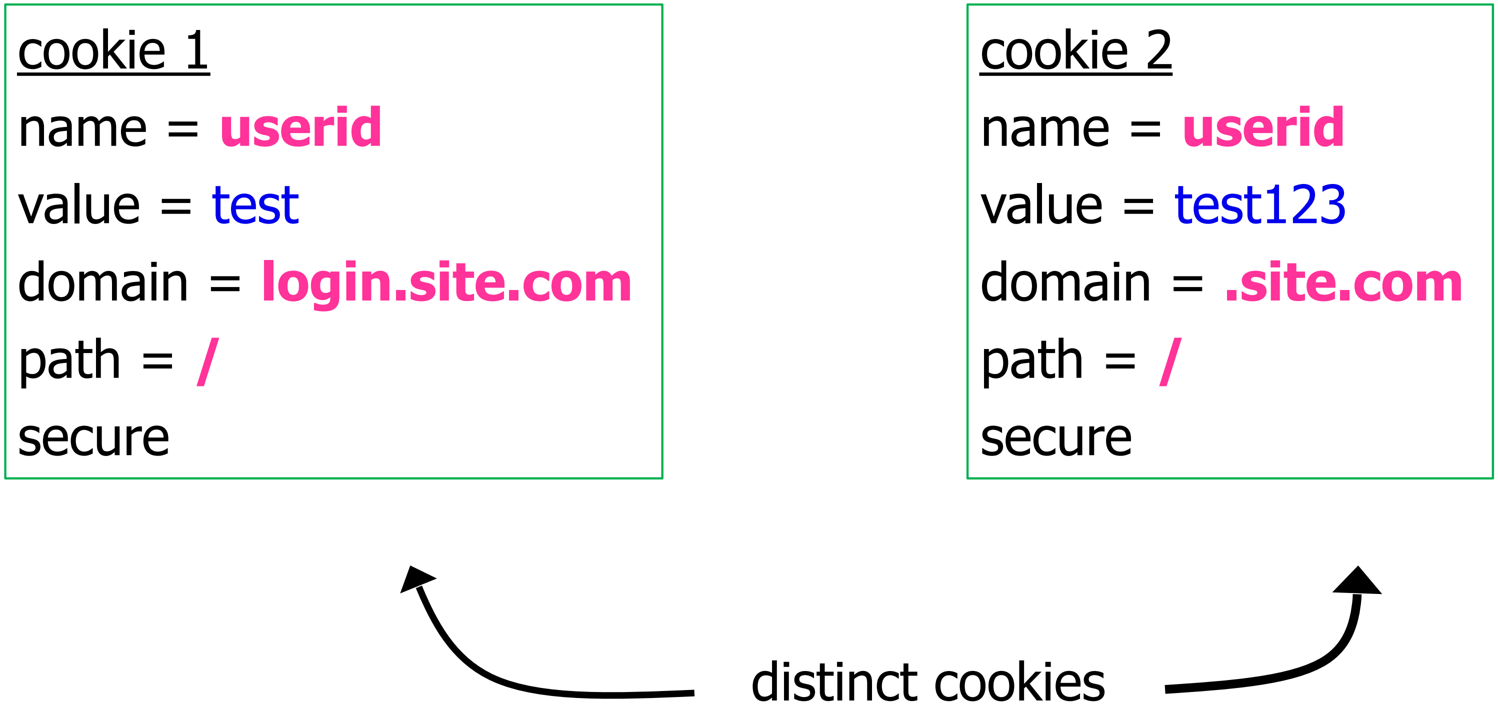
Expires: Sunday, January 17, 2038 4:00:00 PM

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

name = **userid**

value = test

domain = **login.site.com**

path = **/**

secure

cookie 2

name = **userid**

value = test123

domain = **.site.com**

path = **/**

secure

Both cookies stored in browser's cookie jar,

both are in scope of **login.site.com**

SOP for Writing Cookies

resource domain URL has to be a suffix of
the principal domain URL
(except top-level domains (TLD))

Which cookies can be set by **login.site.com**?

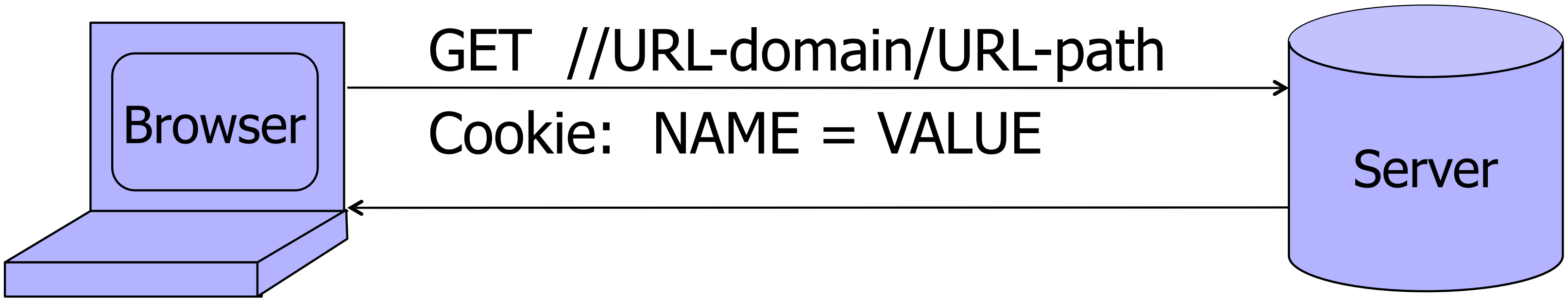
allowed domains	disallowed domains
✓ **login.site.com**	✗ **user.site.com**
✓ **.site.com**	✗ **othersite.com**
	✗ **.com**

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

Problematic for sites like .indiana.edu

path: anything

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
- "secure" cookie if protocol=HTTPS

Goal: server only sees cookies in its scope

Examples of Cookie SOP

cookie 1

name = **userid**

value = u1

domain = **login.site.com**

path = **/**

secure

cookie 2

name = **userid**

value = u2

domain = **.site.com**

path = **/**

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

(arbitrary order; in FF3 most specific first)

Cookie Protocol Issues

- What does the server know about the cookie received from the browser?
- Server only sees Cookie: Name=Value

… does not see cookie attributes (e.g., “secure”)

… does not see which domain set the cookie

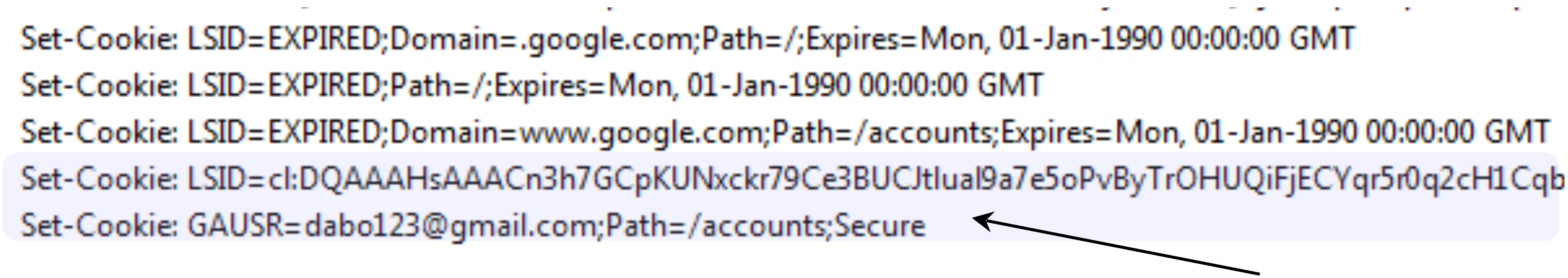
- RFC 2109 (cookie RFC) has an option for including domain, path in Cookie header, but not typically supported by browsers

Who Set The Cookie?

- Alice logs in at login.iu.edu
  - login.iu.edu sets session-id cookie for .iu.edu
- Alice visits evil.iu.edu
  - Overwrites .iu.edu session-id cookie with session-id of user "badguy" - not a violation of SOP! (why?)
- Alice visits i433.iu.edu to submit homework
  - i433.iu.edu thinks it is talking to "badguy"
- Problem: i433.iu.edu expects session-id from login.iu.edu but 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”

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

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

```
<iframe src=“x.com/B"></iframe>

alert(frames[0].document.cookie);
```

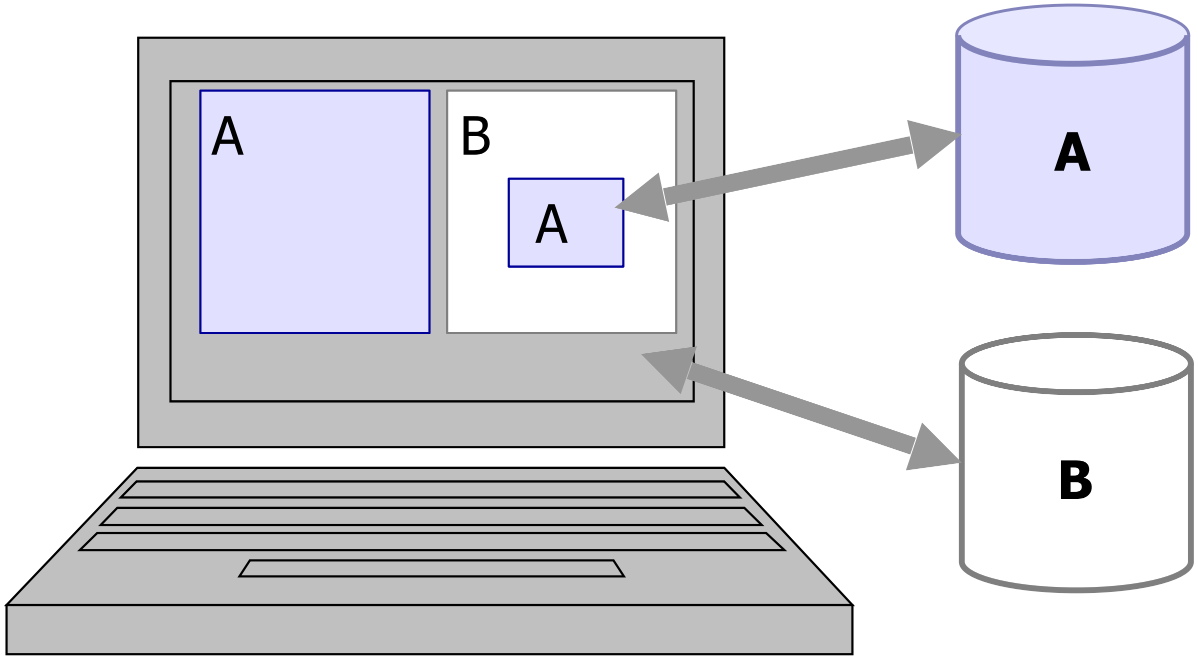
Frames

- Window may contain frames from different sources
  - 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>
```

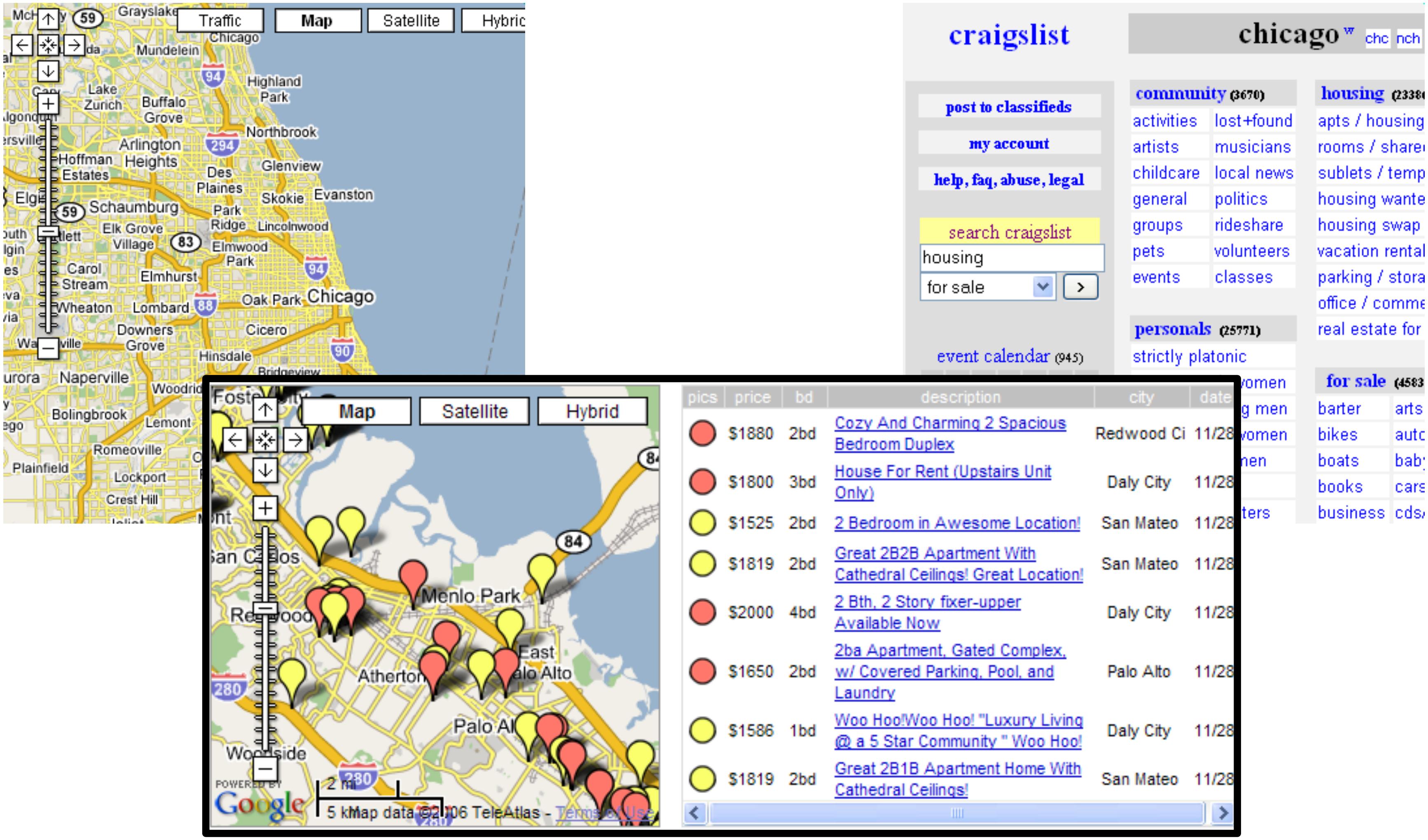
- Why use frames?
  - Delegate screen area to content from another source
  - Browser provides isolation based on frames
  - Parent may work even if frame is broken

Browser Security 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 w 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
nen
ters

housing (2338

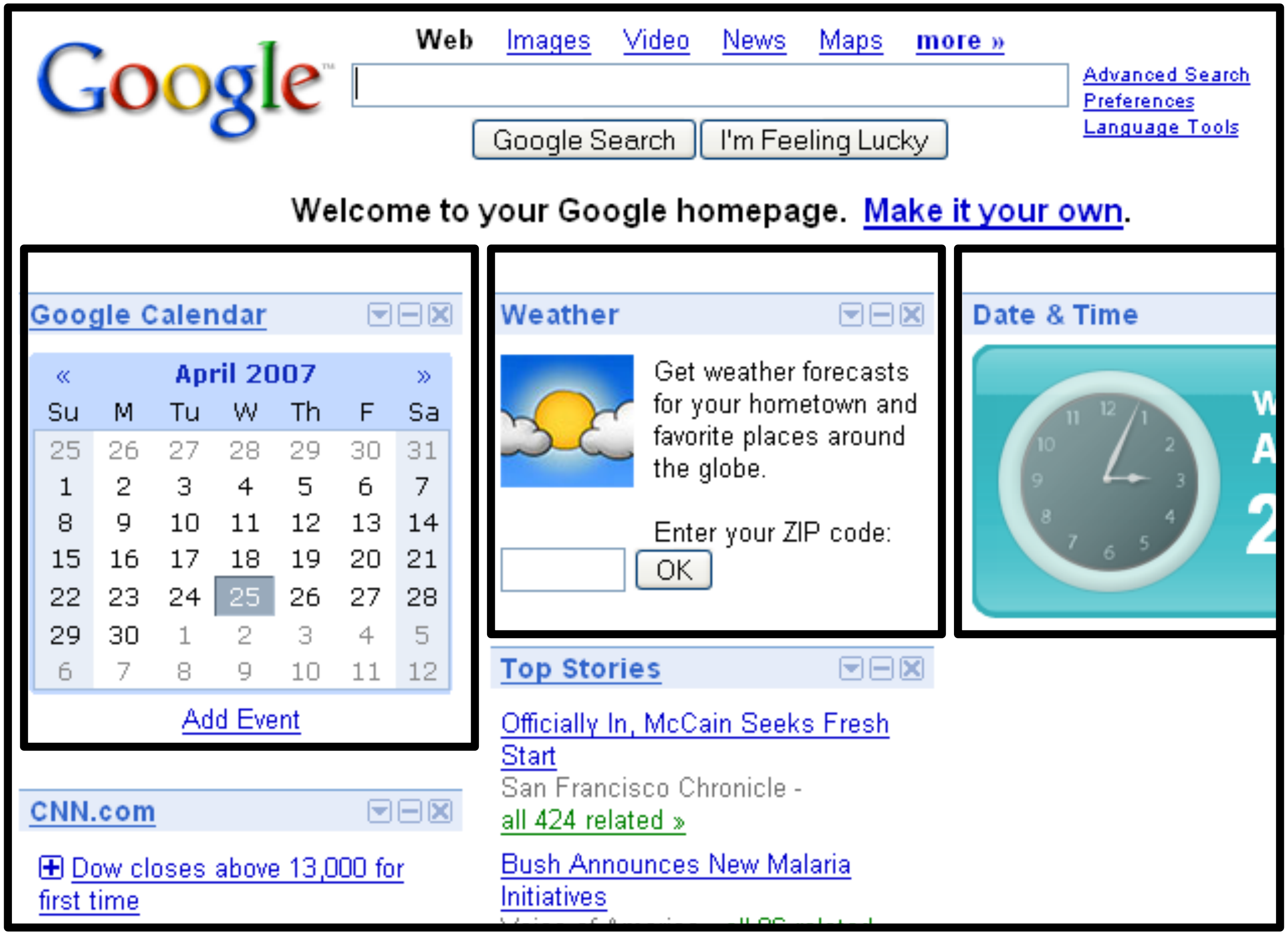
apts / housing
rooms / share
sublets / temp
housing wante
housing swap
vacation rental
parking / stora
office / comme
real estate for

for sale (4583

barter arts
bikes autc
boats baby
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

iGoogle (Now Defunct)



Welcome to your Google homepage. Make it your own.

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

Get weather forecasts for your hometown and favorite places around the globe.

Enter your ZIP code:

Officially In, McCain Seeks Fresh Start
San Francisco Chronicle -
all 424 related »

Bush Announces New Malaria Initiatives

Dow closes above 13,000 for first time

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 identified by (protocol, domain, port)
- 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

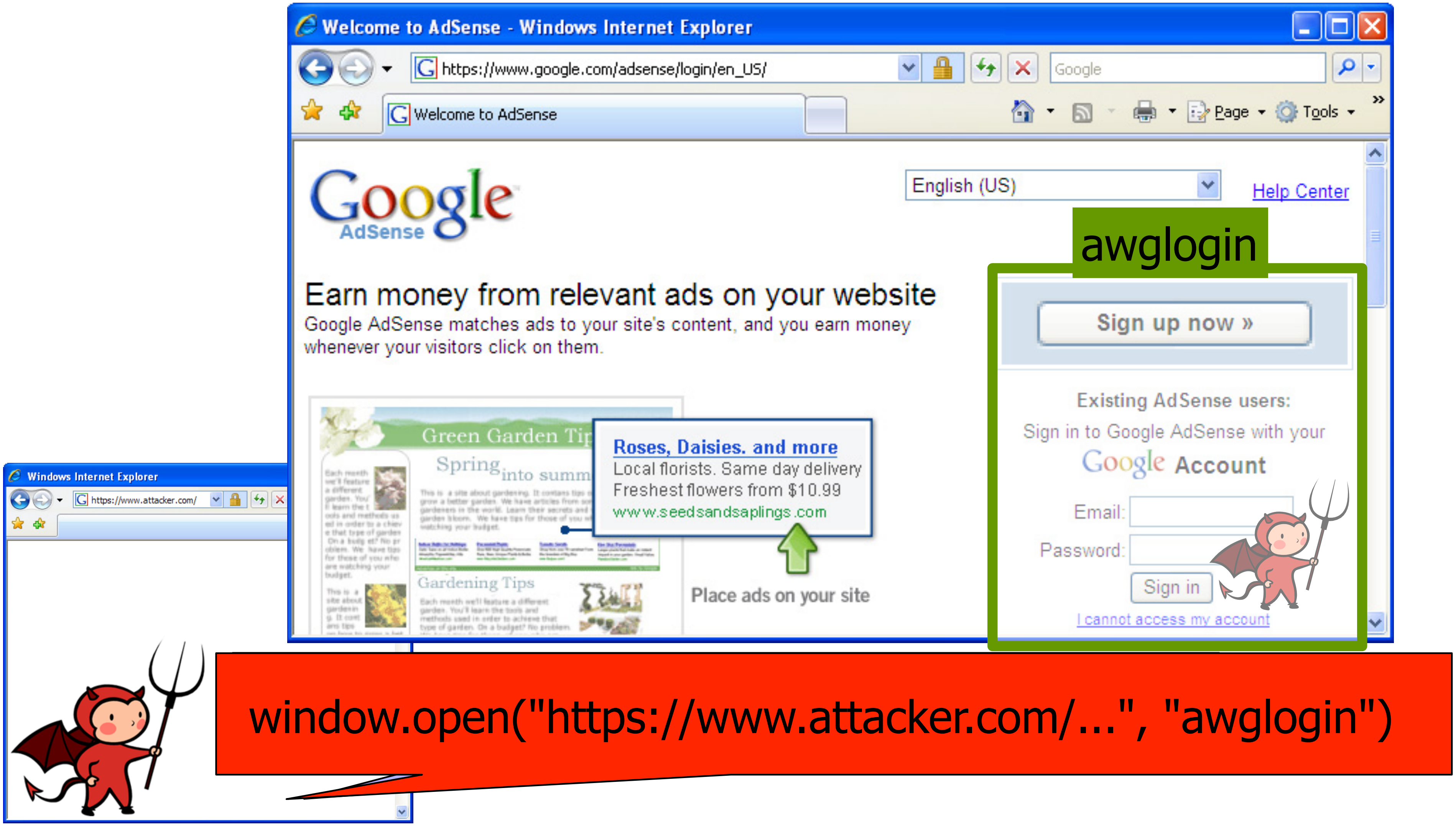
```
<iframe src="http://othersite.com"></iframe>
alert( frames[0].contentDocument.body.innerHTML )
alert( frames[0].src )
```

- 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/
Google
Welcome to AdSense
Page
Tools
English (US)
Help Center
Google AdSense
Windows Internet Explorer
https://www.attacker.com/

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.

Green Garden Tips

Spring into summer

Gardening Tips

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:

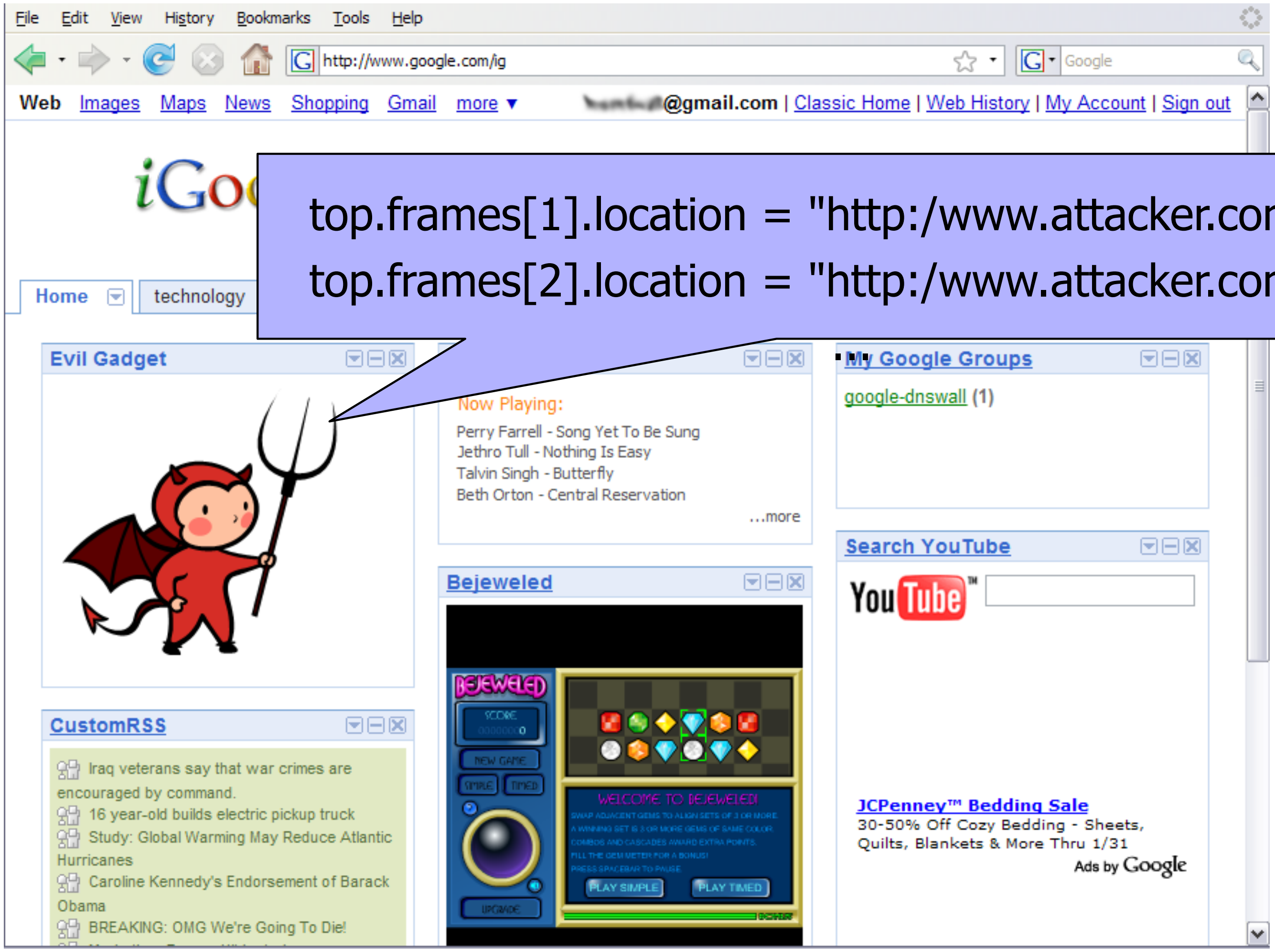
Sign in

I cannot access my account

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

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

Gadget Hijacking in Mashups



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

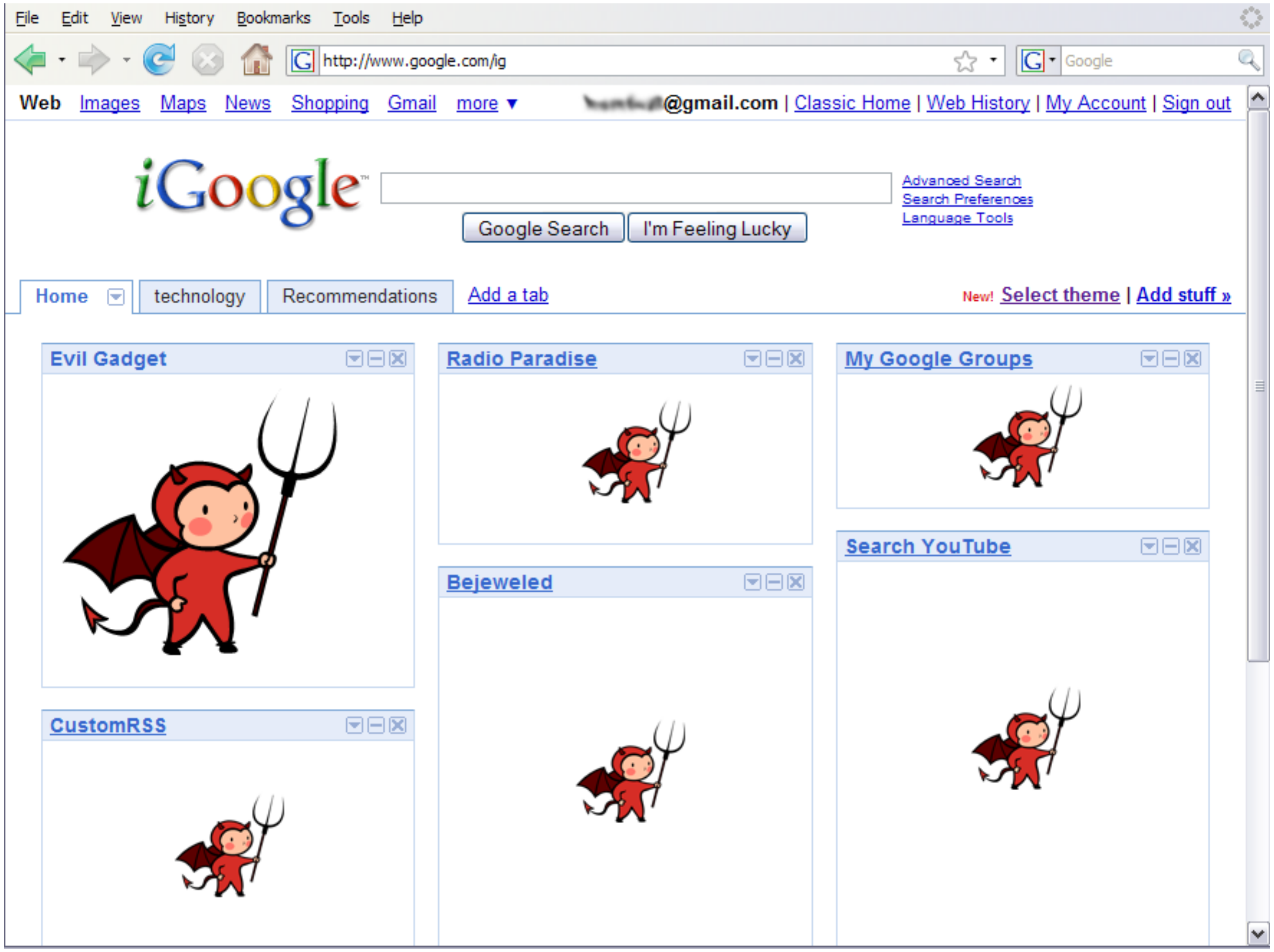
Now Playing:

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

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



Modern browsers only allow a frame to navigate its “descendant” frames