Web Security

CS642:
Computer Security

Professor Ristenpart

http://www.cs.wisc.edu/~rist/

rist at cs dot wisc dot edu

Liberal borrowing from Mitchell, Boneh, Stanford CS 155

More Details On The 3rd-Party Apps That Led to Snapchat ...

Yesterday we posted a link to Computerworld's reports that (unnamed) third-party apps were responsible for a massive leak of Snapchat images from the meant-to-be-secure service. An anonymous reader writes with some more details: Ars Technica identifies the culprit as SnapSaved, which was created to allow Snapchat users to access their sent and received images from a browser but which also secretly saved those images on a SnapSaved server hosted by HostGator. Security researcher Adam Caudill warned Snapchat about the vulnerability of their API back in 2012, and although the company has reworked their code multiple times as advised by other security researchers, Caudill concludes that the real culprit is the concept behind Snapchat itself. "Without controlling the endpoint devices themselves, Snapchat can't ensure that its users' photos will truly be deleted. And by offering that deletion as its central selling point, it's lured users into a false sense of privacy."

Web security part 1

Basic web security models

Browser security

Same-origin policy / Navigation policy

Cookies / Session handling

WWW

Tim Berners-Lee and Robert Cailliau 1990
    HTTP, CERN httpd, gopher

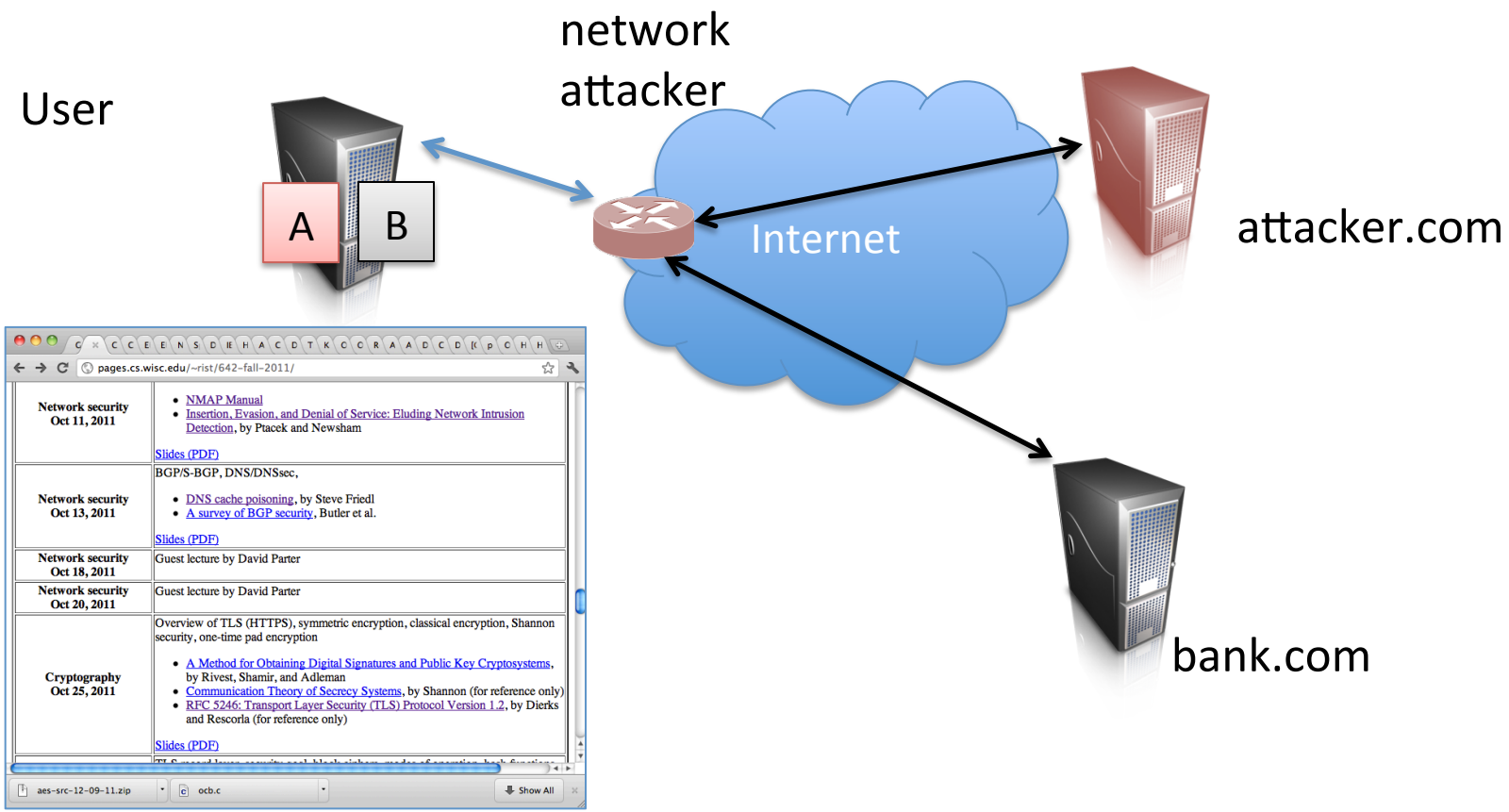
1993 Mosiac web browser (UIUC, Marc Andreesen)

1994 W3C WWW Consortium --- generate standards
    Gopher started charging licensing fees
    (Univ of Minnesota)

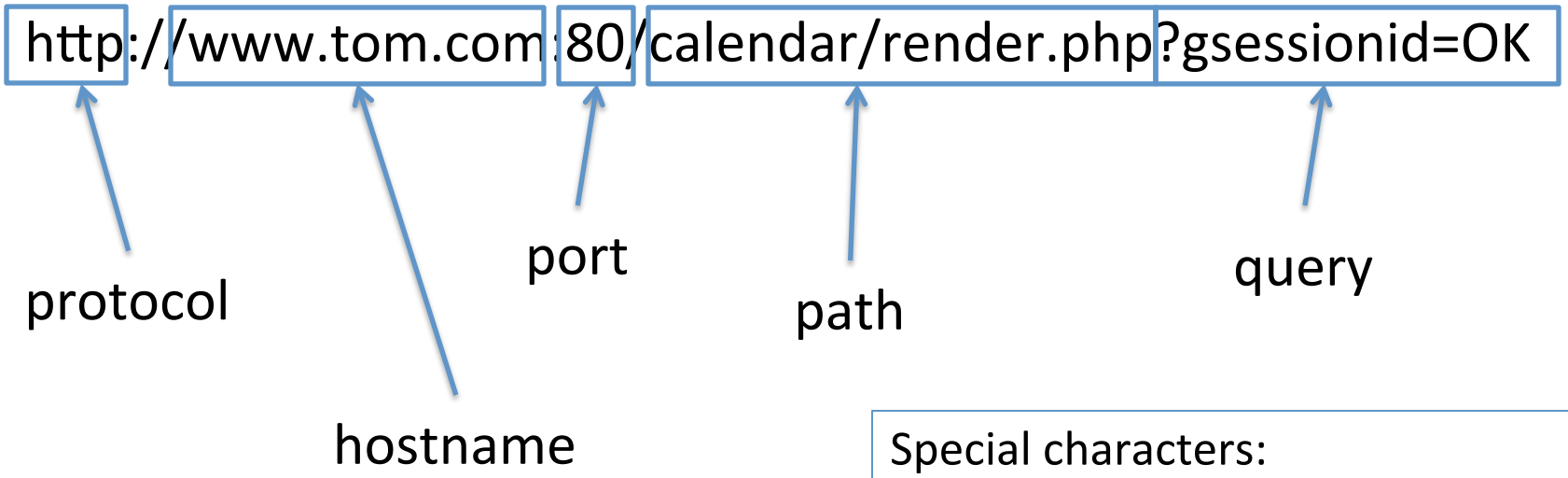
Nowadays: ecosystem of technologies

- HTTP / HTTPS
- AJAX
- PHP
- Javascript
- SQL
- Apache
- Ruby
- http://w3schools.com/

Threat model



Uniform resource locators (URLs)

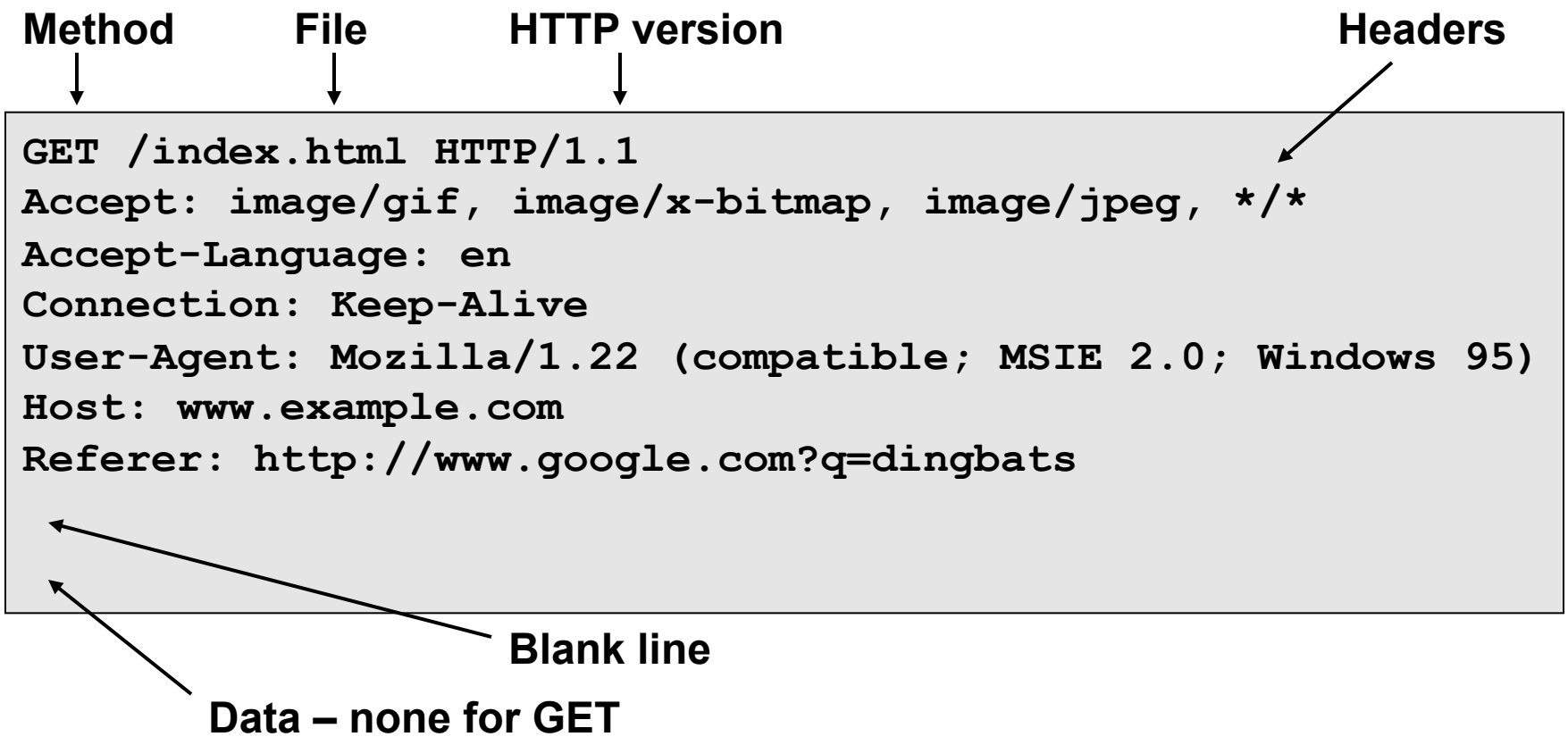


URL's only allow ASCII-US characters.
Encode other characters:

%0A = newline
%20 = space

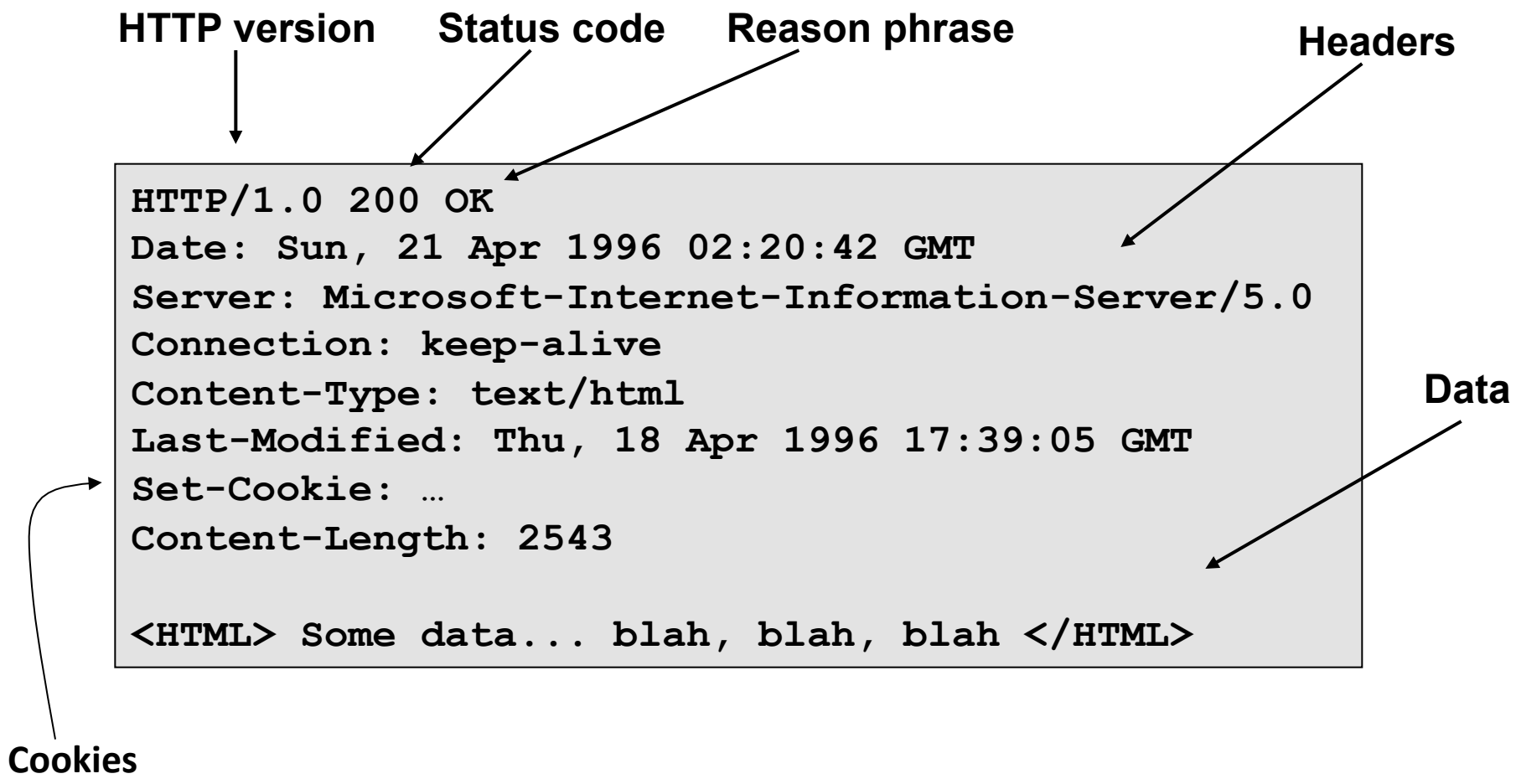
Special characters:
+ = space
? = separates URL from parameters
% = special characters
/ = divides directories, subdirectories
= bookmark
& = separator between parameters

HTTP Request

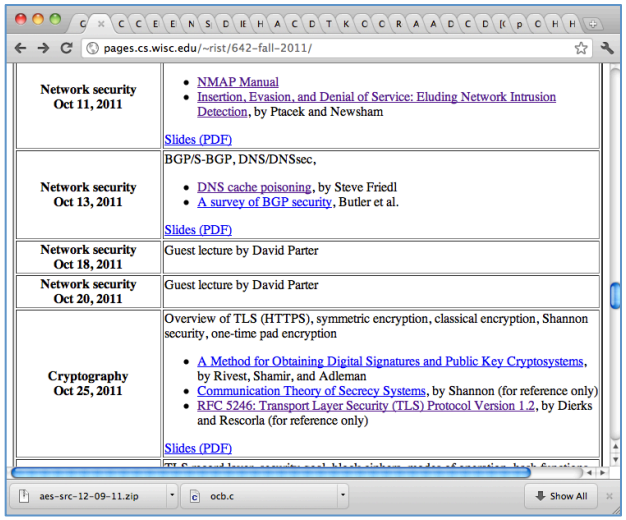


GET : no side effect POST : possible side effect

HTTP Response



Browser execution



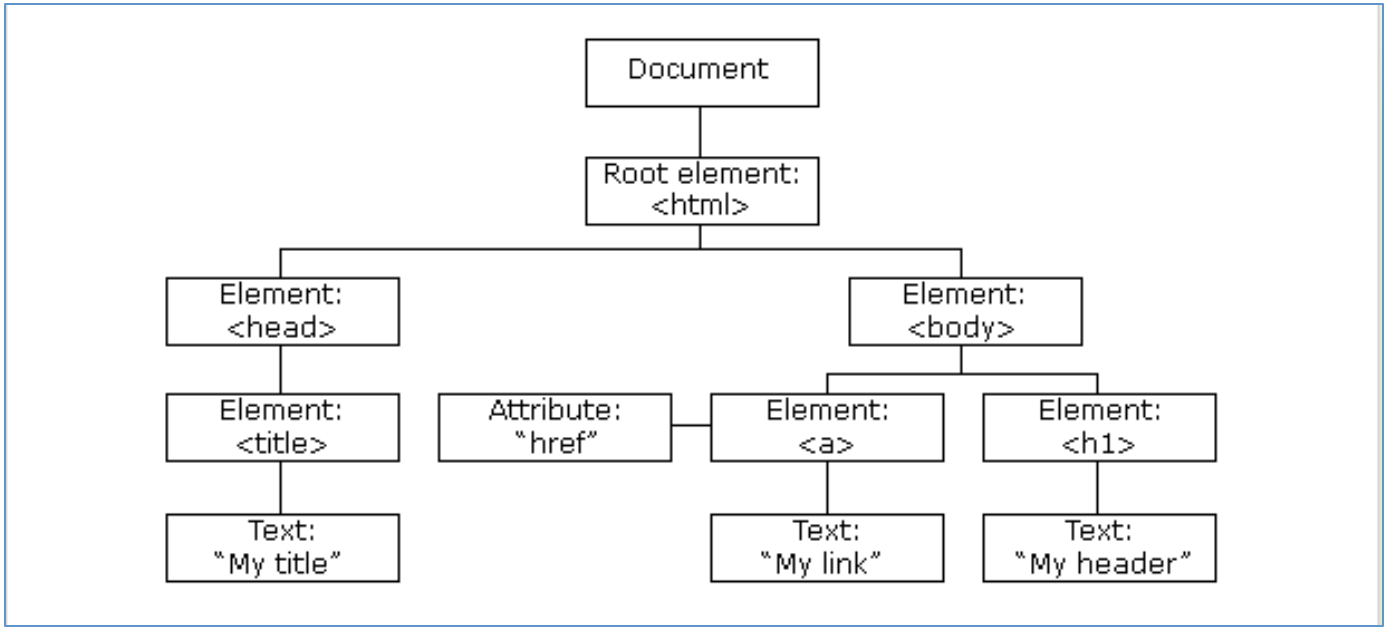
- Each window (or tab):
  - Retrieve/load content
  - Render it
    - Process the HTML
    - Might run scripts, fetch more content, etc.
  - Respond to events
    - User actions: OnClick, OnMouseover
    - Rendering: OnLoad, OnBeforeUnload
    - Timing: setTimeout(), clearTimeout()

Document Object Model (DOM)

Object-oriented way to refer to objects in a web page

Properties: document.alinkColor, document.URL, document.forms[ ], document.links[ ], document.anchors[ ]

Methods: document.write(document.referrer)



From http://w3schools.com/htmldom/default.asp

Document Object Model (DOM)

Object-oriented way to refer to objects in a web page

Properties: document.alinkColor, document.URL,
document.forms[ ], document.links[ ], document.anchors[ ]
Methods: document.write(document.referrer)

Browser Object Model (BOM)

window, document, frames[], history, location,
navigator (type and version of browser)

Seemingly innocuous features?

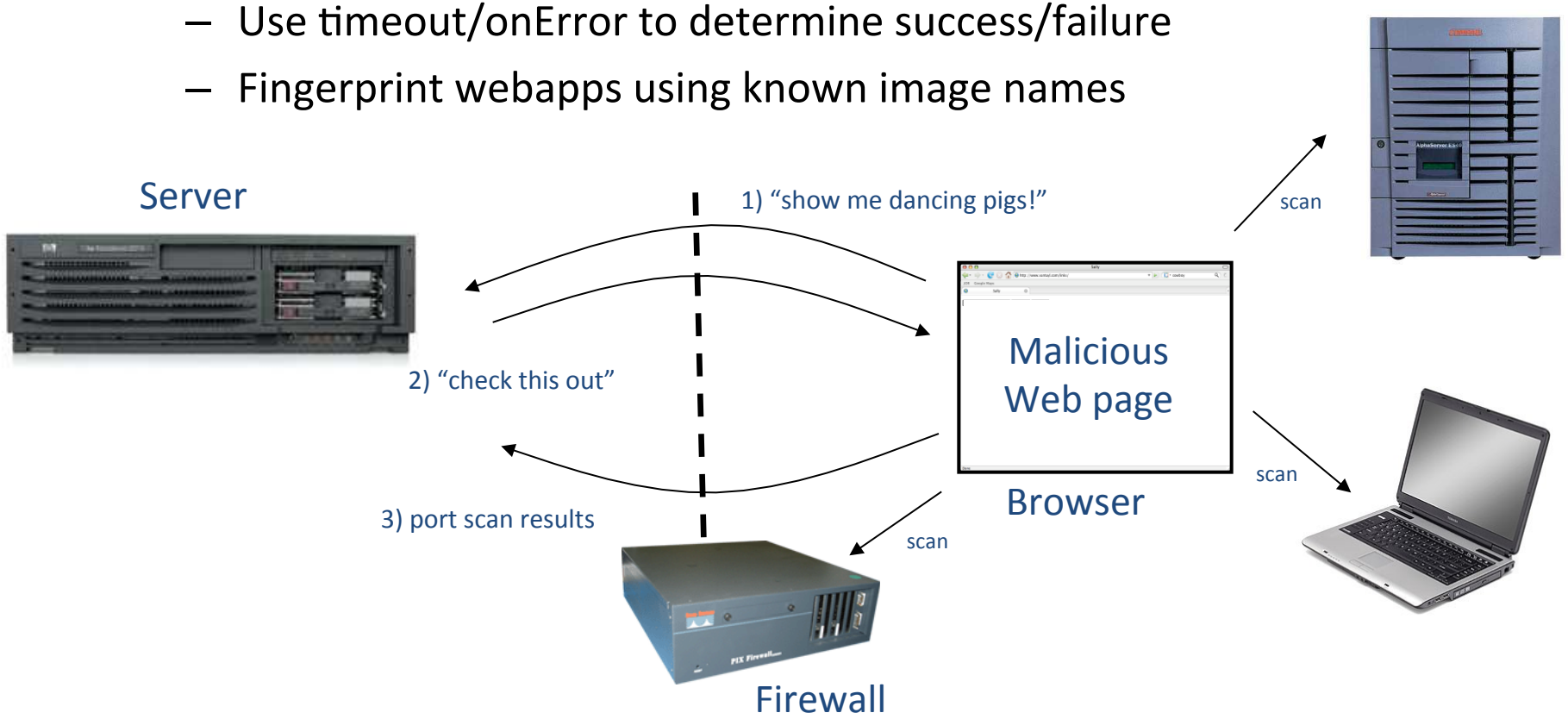
- <img src="bucky.jpg" height="50pt" width="50pt">
- Displays an image
- What can attacker do?

Javascript timing

```
<html><body><img id="test" style="display: none">
<script>
   var test = document.getElementById('test');
   var start = new Date();
   test.onerror = function() {
        var end = new Date();
        alert("Total time: " + (end - start));
    }
    test.src = "http://www.example.com/page.html";
</script>
</body></html>
```

Behind-firewall webapp scanning

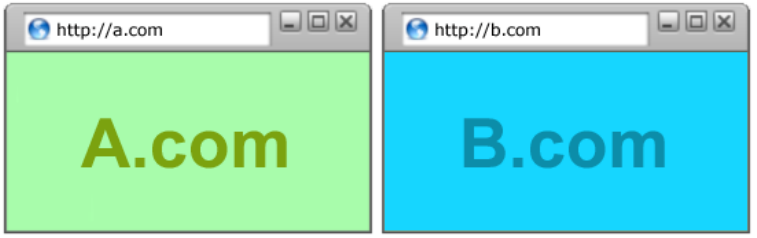
- JavaScript can:
  - Request images from internal IP addresses
    - Example:
  - Use timeout/onError to determine success/failure
  - Fingerprint webapps using known image names



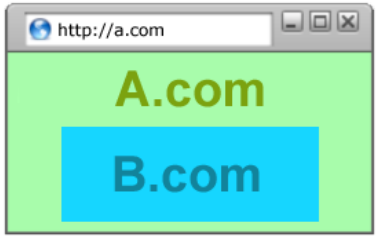
Browser security model

Should be safe to visit an attacker website

Should be safe to visit sites simultaneously



Should be safe to delegate content



Browser isolation

Browser is running untrusted inputs (attacker webpage)

Like all big, complex software, browser has security vulnerabilities

Browsers include “Rich Internet Applications” (RIAs) that increase attack surface:

e.g., Adobe Flash (see reading for today by Blazakis)

Malicious website exploits browser, from there system

Web pages are not single-origin

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

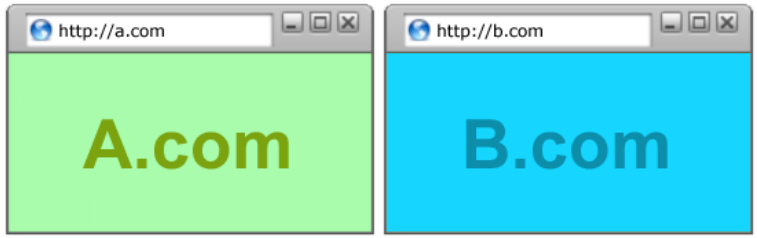
Scripts: **<script src=**“//site.com/script.js” **>** </script>

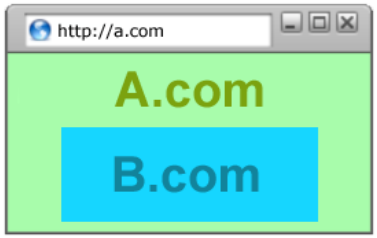
CSS:

<**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>
```





Browser handles multiple sites, must maintain separate security contexts for each

Operating system

- Primitives
  - System calls
  - Processes
  - Disks
- Principals: Users
  - Discretionary access controls
- Vulnerabilities
  - Buffer overflows
  - root exploit
  - …

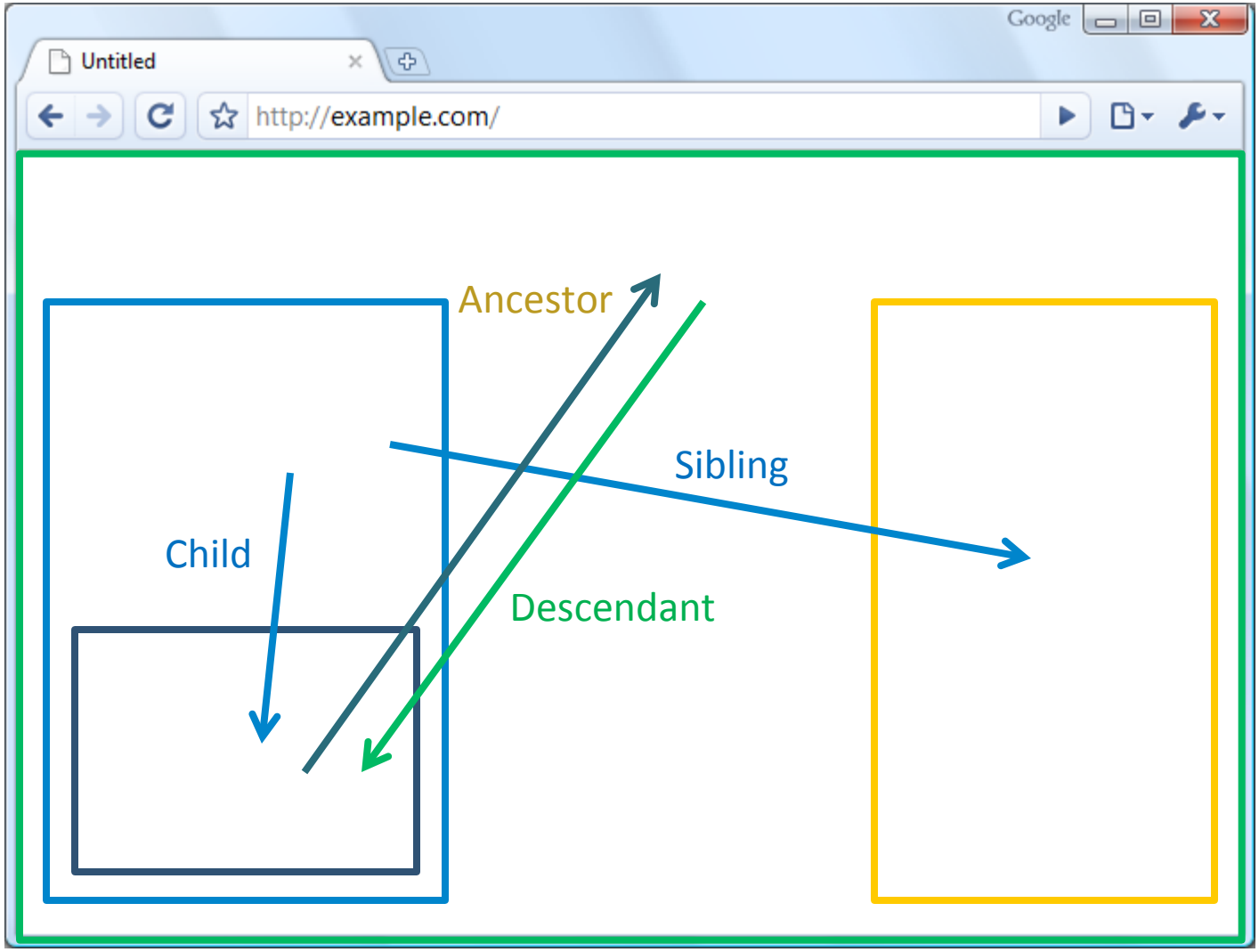
Browsers

- Primitives
  - Document object model
  - Frames
  - Cookies / local storage
- Principals: Origins
  - Mandatory access controls
- Vulnerabilities
  - Cross-site scripting (XSS)
  - Cross-site request forgery (CSRF)
  - Cache history attacks
  - …

Same-origin policy

- Each frame of page(s) has an origin
  - protocol://host:port
  - Origin is (protocol,host,port)
- Frame can access its own origin
  - Network access, Read/write DOM, storage (cookies)
- Frame cannot access data associated with another origin

Frame relationships



Frame policies

canScript(A,B) and canNavigate(A, B)

- Permissive
  - any frame can navigate any other frame
- Child
  - only can navigate if you are parent
- Descendent
  - only can navigate if you are ancestor

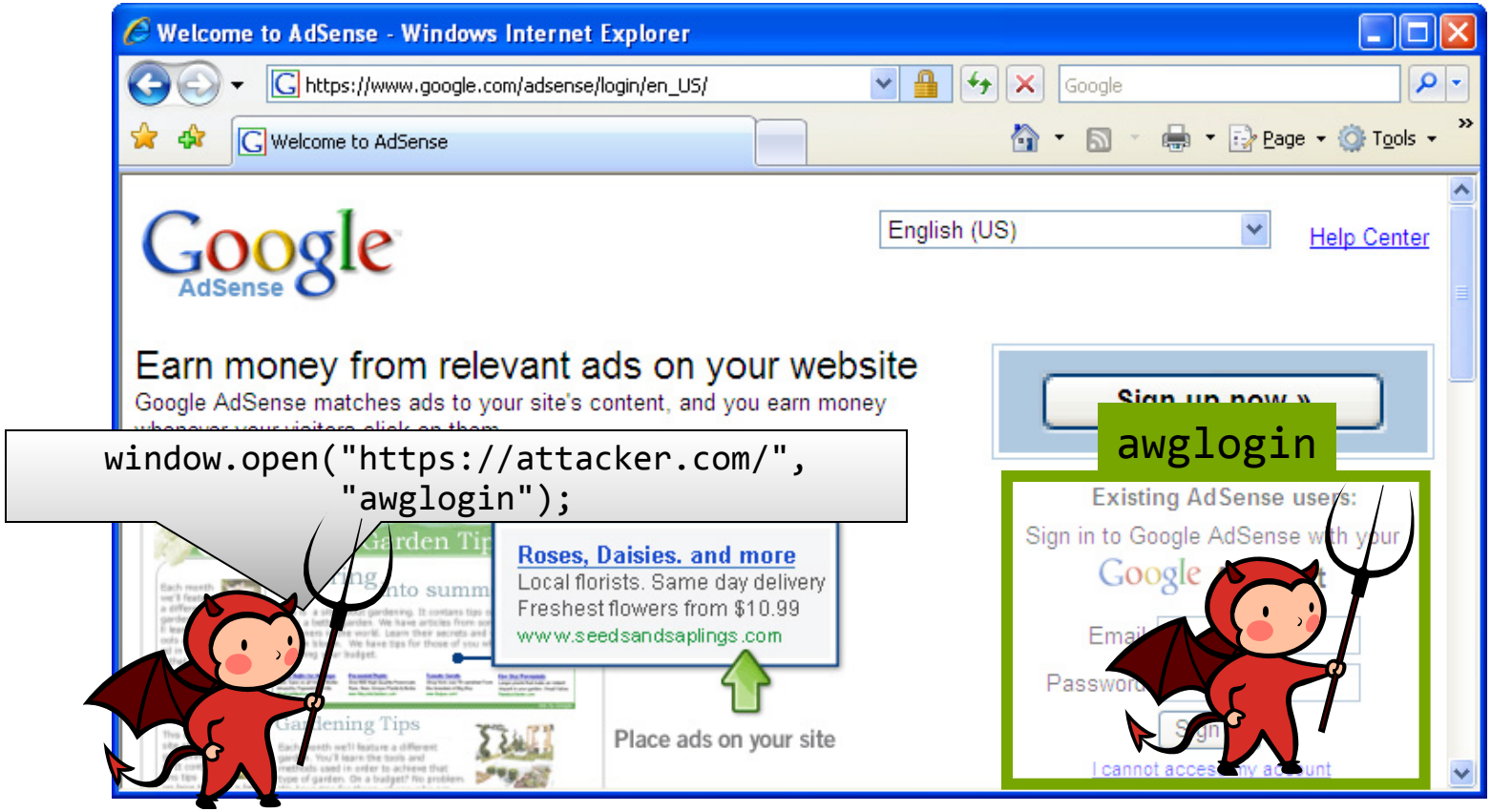
Which do you think should be used?

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

Problems with permissive

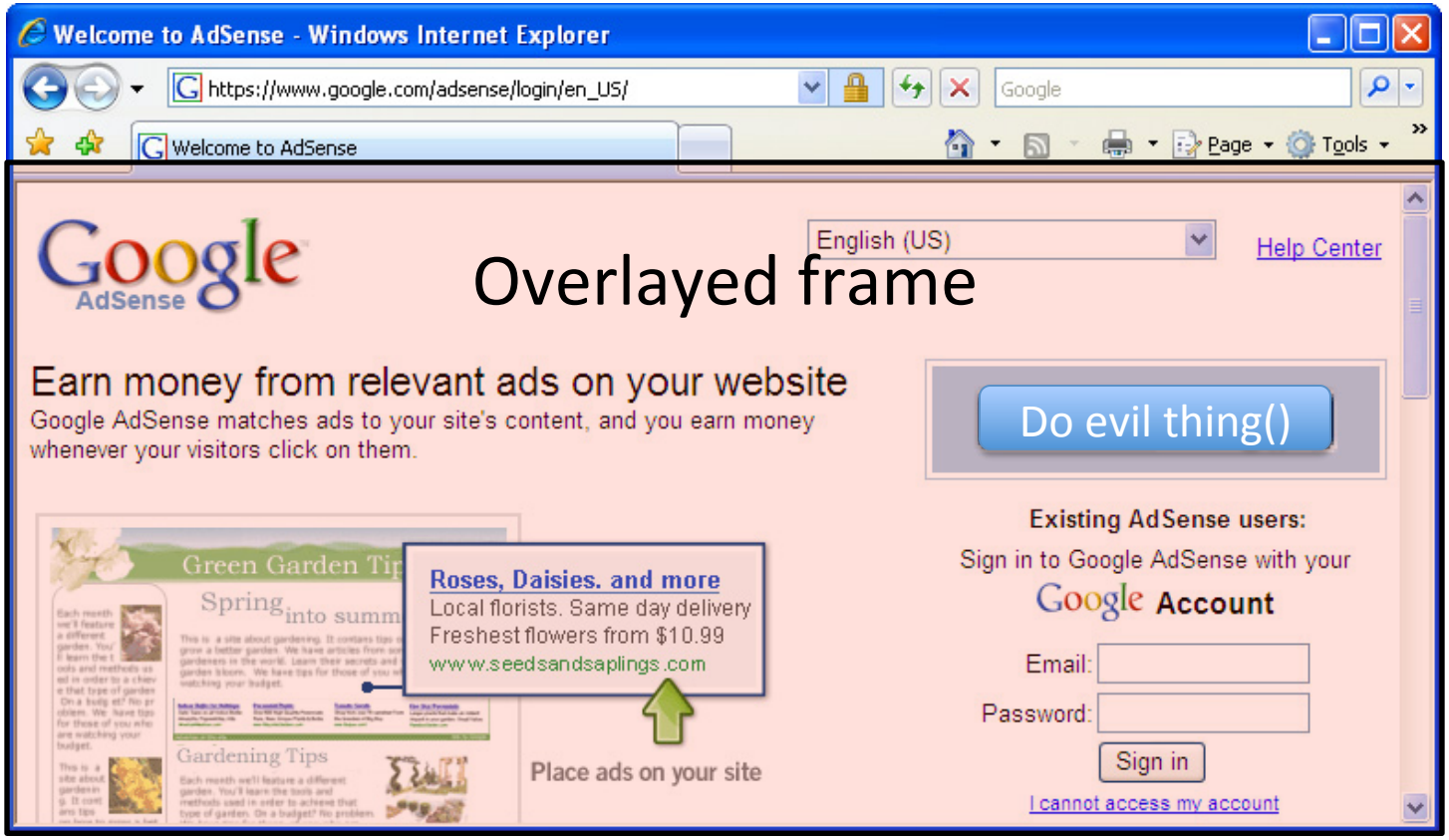
frames['right'].window.location="evil.com/login.html";



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

UI Redressing (Clickjacking)



Defense: NoScript plugin attempts to prevent this for Firefox

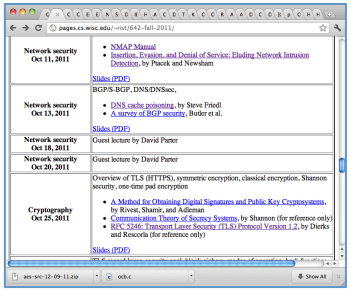
Framebusting

```
<script type="text/javascript">
   if(top != self) top.location.replace(location);
</script>
```

Embed in your webpage to avoid being rendered within another (adversarial) frame.

Has limitations. See "Busting Frame Busting: a Study of Clickjacking Vulnerabilities on Popular sites"

Cookies: Setting/Deleting



GET ...

HTTP Header:

Set-cookie: NAME=VALUE ;

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

scope

secure = (only send over SSL);

expires = (when expires) ;

HttpOnly

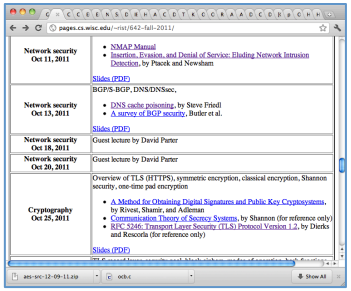
if expires=NULL:
this session only

- Delete cookie by setting “expires” to date in past
- Default scope is domain and path of setting URL
- Client can also set cookies (Javascript)

Cookie scope rules (domain and path)

- Say we are at www.wisc.edu
  - Any non-TLD suffix can be scope:
    - allowed: www.wisc.edu or wisc.edu
    - disallowed: www2.wisc.edu or ucsd.edu
- Path can be set to anything

Cookies: reading by server



GET /url-domain/url-path

Cookie: name=value

- Browser sends all cookies such that
  - domain scope is suffix of url-domain
  - path is prefix of url-path
  - protocol is HTTPS if cookie marked “secure”

Cookie security issues?

- Cookies have no integrity
  - HTTPS cookies can be overwritten by HTTP cookie (network injection)
  - Malicious clients can modify cookies
    - Shopping cart vulnerabilities
- Scoping rules can be abused
  - blog.example.com can read/set cookies for example.com
- Privacy
  - Cookies can be used to track you around the Internet
- HTTP cookies sent in clear
  - Session hijacking

Tuesday, Nov 08, 2011

PC Batman: Arkham City This Month [10:40 am ET] - Share - 2 Comments

WBIE announces the release dates for the delayed Windows PC edition of **Batman: Arkham City**, the stealth/action sequel:

> Warner Bros. Interactive Entertainment and DC Entertainment today confirmed that the Games for Windows PC version of Batman: Arkham City will be available in North America beginning November 22, in Australia beginning November 23, in France and Benelux beginning November 24, and in other European territories beginning November 25.

Answers.com Now Only With Facebook and Own Login

Posted by **timothy** on Tuesday November 08, @12:30PM
from the you-haff-been-assimilated dept.

CptnHarlock writes

> "Today the registered users of Answers.com received an email informing them that the site has ended support for Yahoo, Twitter, Google, or LinkedIn as a way to sign into their site. Facebook is the sole external way left to log in. A local login and password were generated and sent by email and the old (non-Facebook) logins deactivated. Score another one for Facebook.com in the login consolidation wars."

Read the **14** comments

facebook google privacy

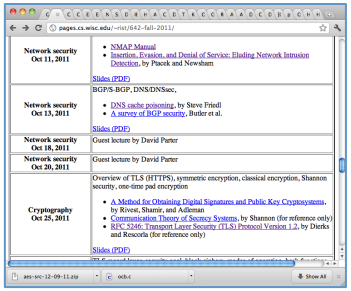
```
<script type="text/javascript">
	//<![CDATA[
		var hint   = 'mainpage';
	document.write('<script type="text/javascript" src="http://ad.doubleclick.net/adj/
ostg.slashdot/pg_index_p1_leader;pg=index2;logged_in=0;tile='+dfp_tile
+';sz=728x90;u=;ord='+dfp_ord+'?"><\/script>');
		dfp_tile++;
	//]]>
</script>
```

In addition to ads based on interest categories, Google allows advertisers (including Google) to show you ads based on your previous interactions online, such as visits to advertisers' websites. For example, someone who visited the website of an online sporting goods store can receive ads about special offers from that store.
--- http://www.google.com/privacy/ads/

Google Dominates Search Advertising With 80% Market Share Unaffected By The Rise Of Bing

Posted on June 21, 2011 by Advanced Media Productions

Session handling and login



GET /index.html

Set-Cookie: AnonSessID=134fds1431

Protocol is HTTPS. Elsewhere just HTTP

POST /login.html?name=bob&pw=12345

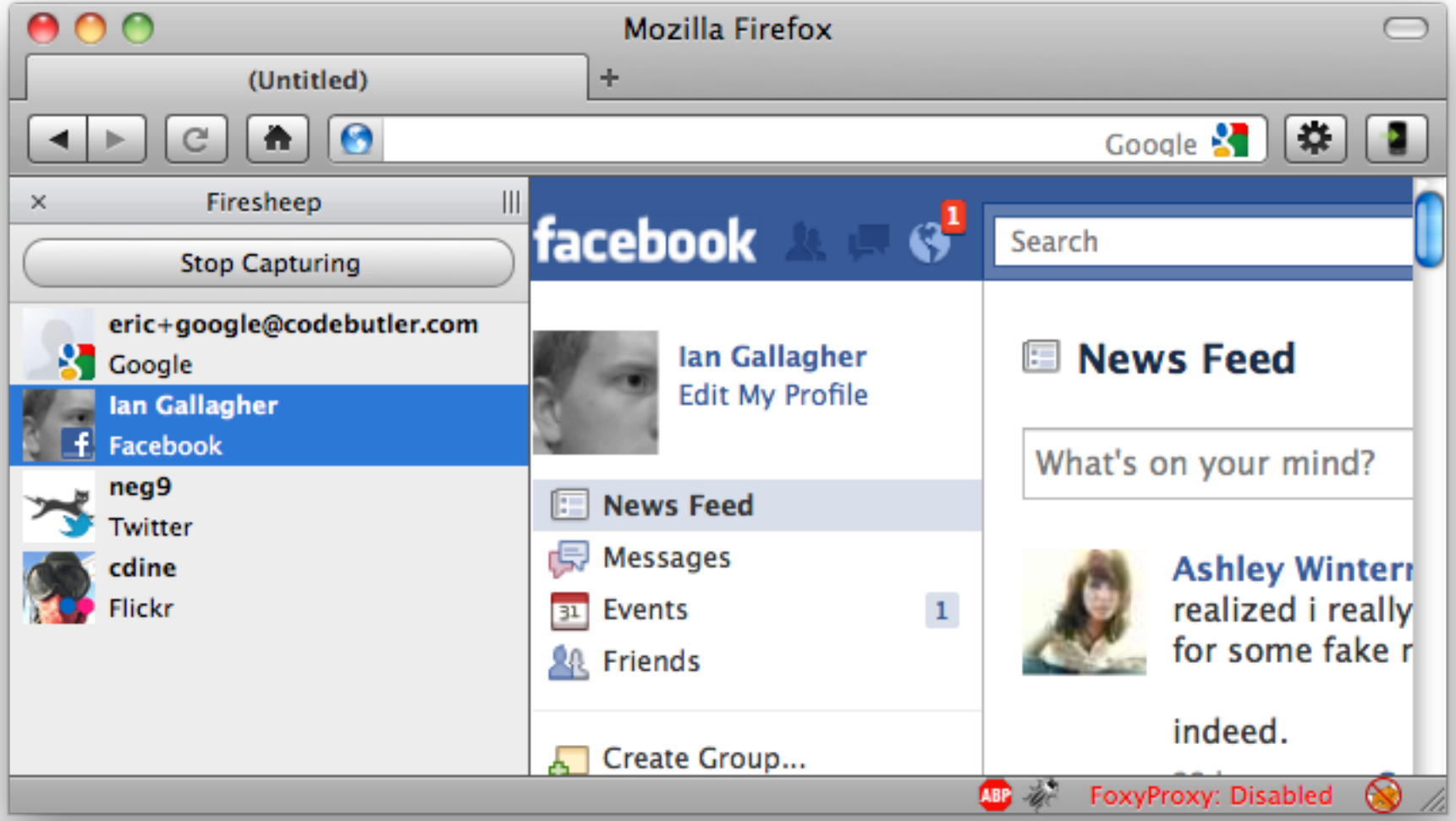
Cookie: AnonSessID=134fds1431

Set-Cookie: SessID=83431Adf

GET /account.html

Cookie: SessID=83431Adf

Session Hijacking



From http://codebutler.com/firesheep

Towards preventing hijacking

- Use encryption when setting session cookies
- SessID = Enc(K,info) where :
  - K is server-side secret key
  - Enc is Encrypt-then-MAC encryption scheme
  - info contains:
    - user id
    - expiration time
    - other data
- Server should record if user logs out
- Does this prevent Firesheep hijacking?
  - No
  - include in data machine-specific information
  - turn on HTTPS always