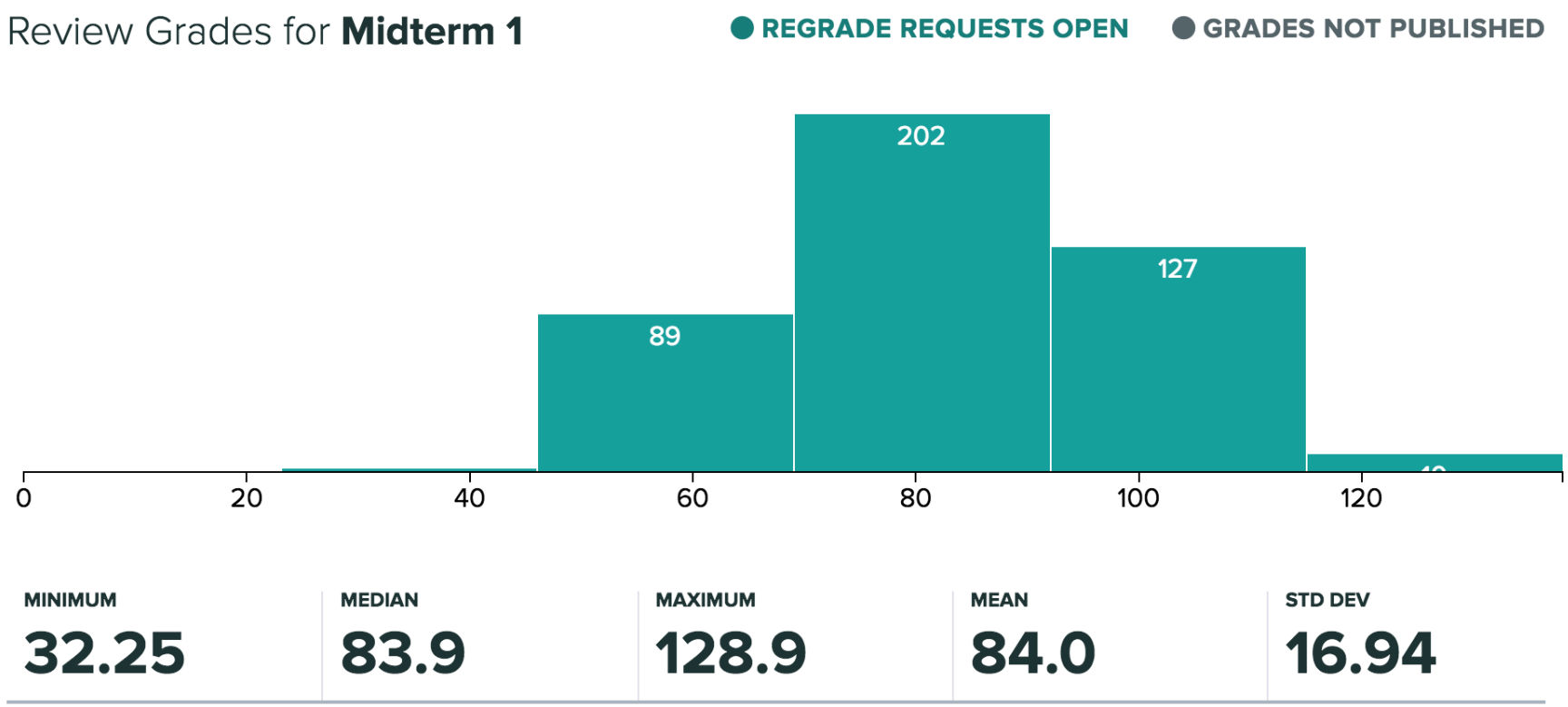
Web Security

SwiftOnSecurity
@SwiftOnSecurity

Following

"Taylor, virus is spreading to 20 nodes/min"
"Everything will be alright if we just keep dancing like we're"
"God damn it"
"TWENTY TWOOOOOO"

I Believe in Hard Tests but An Easy Curve: 3.3 Average GPA For The Class



About Project 2...

- Not only is it to teach you the difficulty of implementing crypto systems
  - Real-world cruel grading would be "1 security bug -> 0 credit!"
- But to also test/teach by doing some important software engineering skills
  - Using a safe language (Go)
  - Developing good tests
    - Go has an excellent testing infrastructure
  - Design ***first!***
  - Serialization & Deserialization of Data
    - How to go from program internal representations to blobs-of-data and back...

Don't write code first, ***design*** first!

- Read all 3 parts...
- Write your design document ***first***
  - When you ask the TAs for help, they are instructed to start with your design document!
- Good design makes the project easy
  - My 100% solution is <400 LOC
- Couple more hints on the design...
  - What do HMAC and Argon2 do? What does PBKDF2 stand for?
  - When in doubt there is the universal CS solution: add another layer of indirection!

Paradigm #1: Do It Manually...

- The C/C++ traditional world
  - Also very common in network programming
  - Python's **`struct`** module as well
- Define a byte order
  - If you need to go between different instruction sets!
- Pack/unpack data into bytes
  - If you may have endianness, use **`ntoh`** and **`hton`**
- Generally safe when adversaries hand you data...
  - Assuming you don't do classic memory screwups that is
- Generally a PitA!

Paradigm #2: Java `serialize` & python `pickle`...

- Nice and convenient:
  - Allows you to dump and restore objects
- But ***horribly dangerous!***
  - If an adversary provides an object, it can deserialize to basically anything!
- Never, ***ever ever ever*** use these if you are communicating outside your own program!
  - They are not suitable for a malicious environment!

Paradigm #3:
Google Protocol Buffers

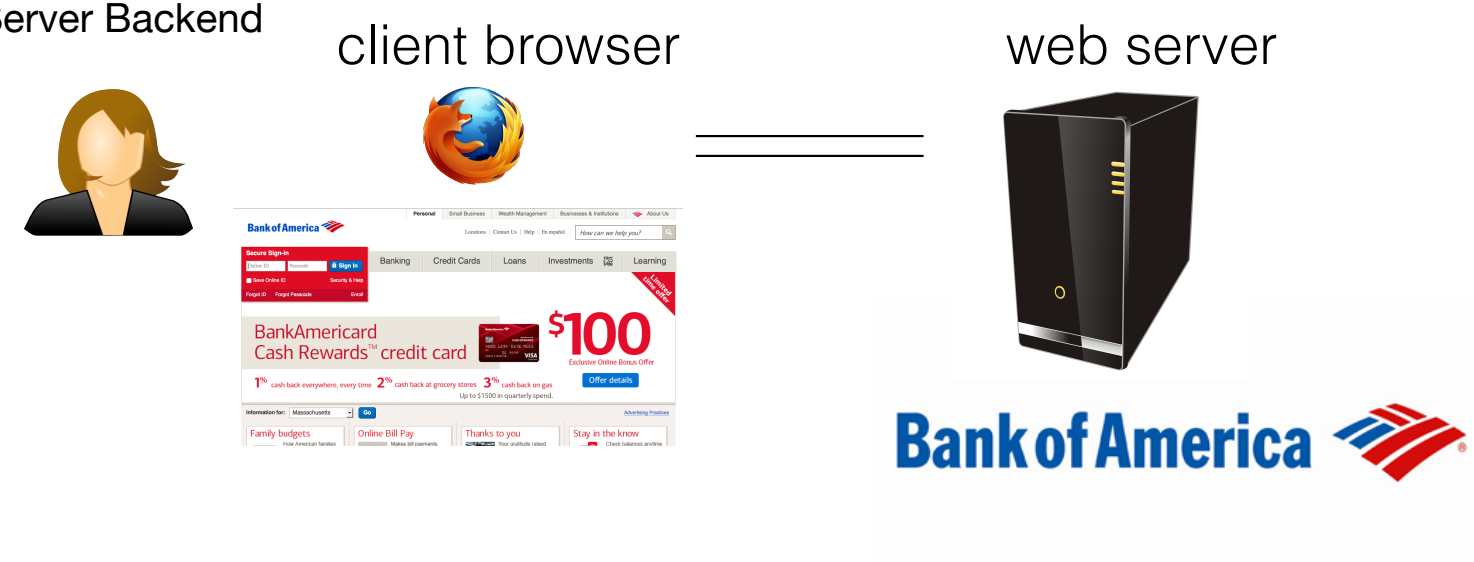
- Provides a compiler to compile code to pack/unpack structures
  - Highly efficient binary encoding
  - Available for C++, python, java, go, ruby, Objective-C, C#
- Safe, but requires using an external compiler to create code to pack/unpack structures
  - But its not human readable in the slightest

Paradigm #4: XML and JSON

- Text based formats
  - Human readable-ish:
    Don't underestimate the value in being able to read your computer data directly!
- JSON is small and simple
  - Just a few types in key/value pair structures
- XML is grody and complex
- Both are less compact however
  - Lots of useless text:
    But you can get most of that back by gzip...
- So we provide you with Json marshal/unmarshal!
  - Hint: You can coerce the bytes to a string if you want to print what is being written!

What is the Web?

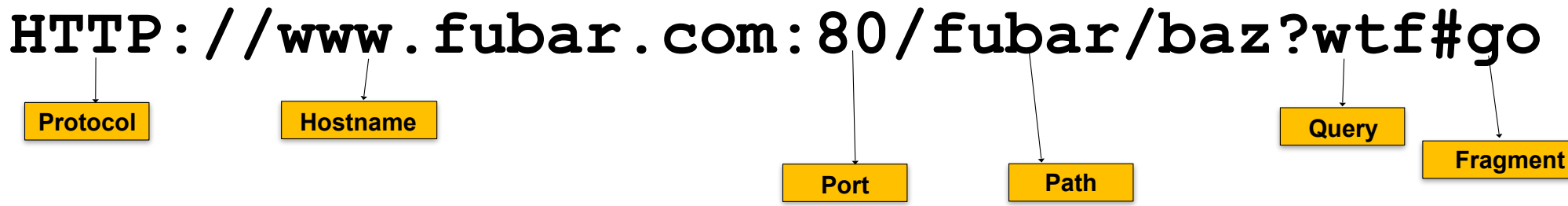
- A platform for deploying applications and sharing information, portably and ?securely?
  - Really a three part ***distributed*** programming problem:
    - The Client Browser
    - The Web Server
    - The Server Backend



HTTP
(Hypertext Transfer Protocol)

A common data communication protocol on the web

URLs:
Global Network Identifiers



- Protocol: Mandatory
  - HTTP, HTTPS, FTP, etc...
- Hostname: Mandatory
  - Either a resolvable domain name or an IP address
- Port: Optional
  - Each protocol has a default port
- Path: Mandatory
  - But can be / for the root
- Query: Optional
  - Sent to Server
- Fragrment
  - ***Local*** to the client
  - Only accessible to scripts in the web page

HTTP


CLIENT BROWSER
WEB SERVER
safebank.com
safebank.com/account.html
SAFEBANK
Alice Smith
Accounts
Bill Pay
Mail
Transfers
HTTP REQUEST:
GET /account.html HTTP/1.1
Host: www.safebank.com
HTTP RESPONSE:
HTTP/1.0 200 OK
<HTML> . . . </HTML>

HTTP Request

GET: no side effect (supposedly, HA)

POST: possible side effect, includes additional data

Method Path HTTP version

Headers

```
GET /index.html HTTP/1.1
Accept: image/gif, image/x-bitmap,
 image/jpeg, */*
Accept-Language: en
Connection: Keep-Alive
User-Agent: Chrome/21.0.1180.75 (Macintosh;
 Intel Mac OS X 10_7_4)
Host: www.safebank.com
Referer: http://www.google.com?q=dingbats
```

Blank line

Data – none for GET

HTTP


CLIENT BROWSER
WEB SERVER
safebank.com
safebank.com/account.html
SAFEBANK
Alice Smith
Accounts
Bill Pay
Mail
Transfers
HTTP REQUEST:
GET /account.html HTTP/1.1
Host: www.safebank.com
HTTP RESPONSE:
HTTP/1.0 200 OK
<HTML> . . . </HTML>

HTTP Response

HTTP version

Status code

Reason phrase

Headers

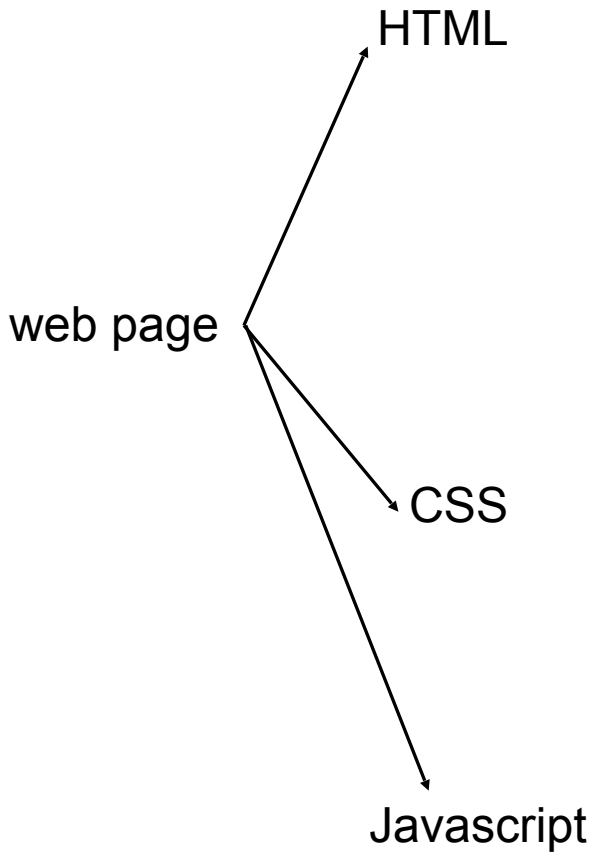
```
HTTP/1.0 200 OK
Date: Sun, 12 Aug 2012 02:20:42 GMT
Server: Microsoft-Internet-Information-Server/
5.0
Connection: keep-alive
Content-Type: text/html
Last-Modified: Thu, 9 Aug 2012 17:39:05 GMT
Set-Cookie: session=44ebc991
Content-Length: 2543

<HTML> This is web content formatted using html
</HTML>
```

Data

"Cookie" – state that server asks client to store, and return in the future
(discussed later)

Can be a webpage, image, audio, executable ...

Web page



HTML

A language to create structured documents
One can embed images, objects, or create interactive forms

index.html

```
<html>
    <body>
        <div>
            foo
            <a href="http://google.com">Go to Google!</a>
        </div>
        <form>
            <input type="text" />
            <input type="radio" />
            <input type="checkbox" />
        </form>
    </body>
</html>
```

CSS (Cascading Style Sheets)

Language used for describing the presentation of a document

index.css

```
p.serif {
font-family: "Times New Roman", Times, serif;
}
p.sansserif {
font-family: Arial, Helvetica, sans-serif;
}
```

Javascript

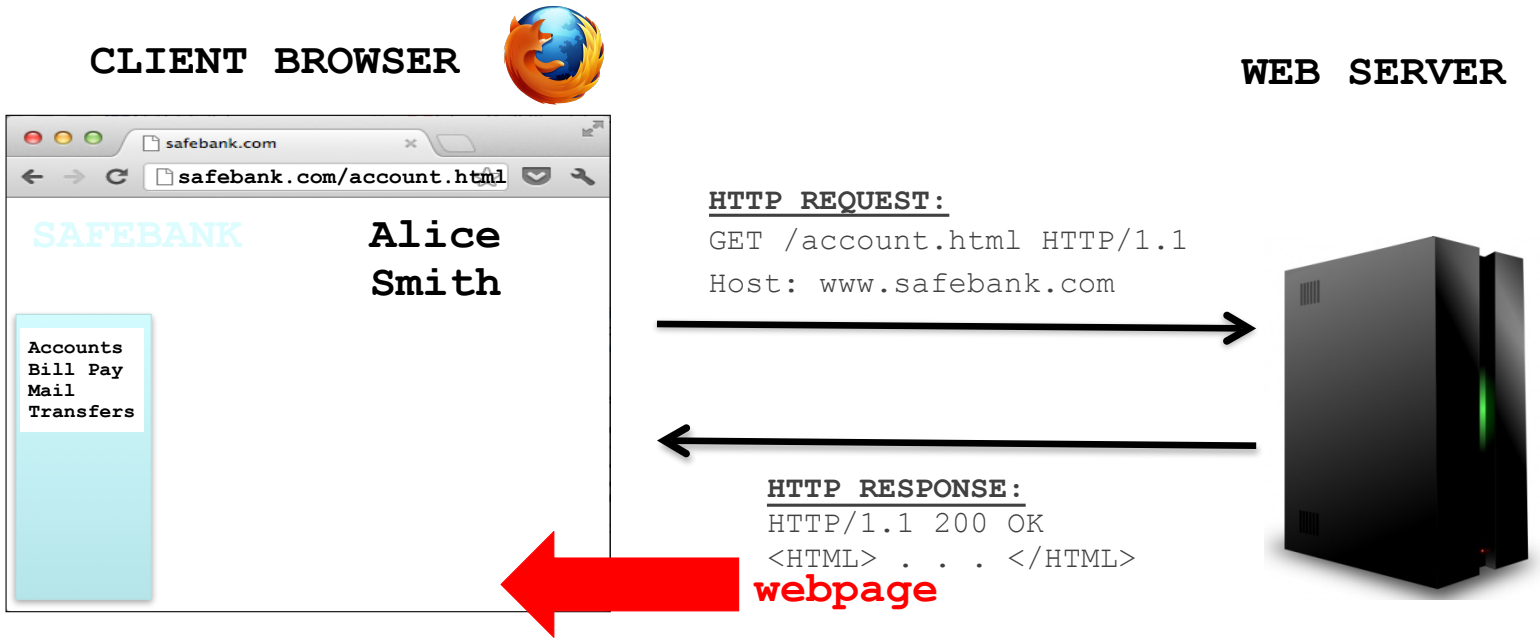
Programming language used to manipulate web pages. It is a high-level, untyped and interpreted language with support for objects.

Supported by all web browsers

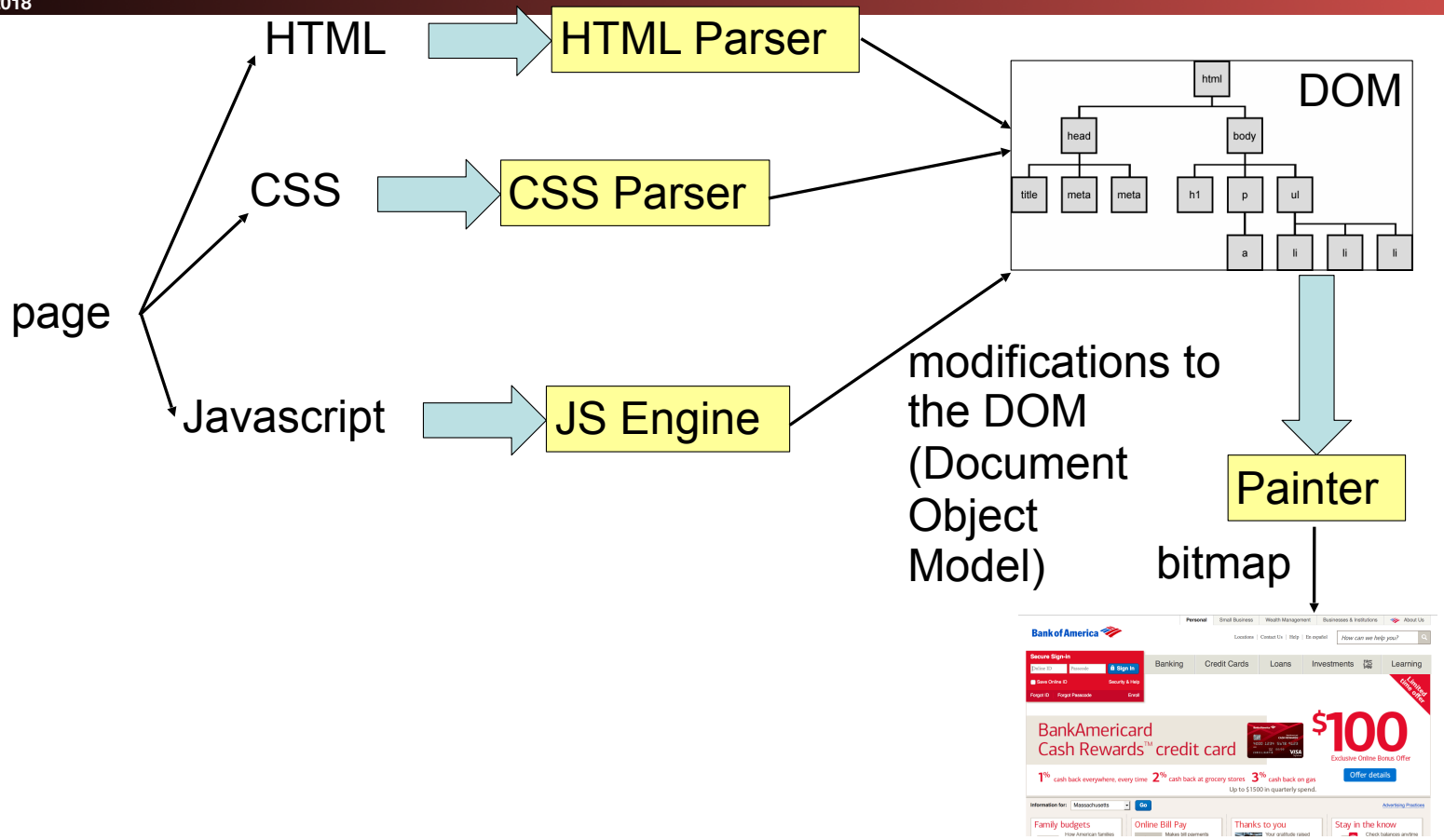
```
<script>
function myFunction()
{    document.getElementById("demo").innerHTML = ”Text
changed.";
}
</script>
```

Very powerful!

HTTP


CLIENT BROWSER
WEB SERVER
safebank.com
safebank.com/account.html
SAFEBANK
Alice Smith
Accounts
Bill Pay
Mail
Transfers
HTTP REQUEST:
GET /account.html HTTP/1.1
Host: www.safebank.com
HTTP RESPONSE:
HTTP/1.1 200 OK
<HTML> . . . </HTML>
webpage

Page rendering



DOM (Document Object Model)

Cross-platform model for representing and interacting with objects in HTML

HTML

```
<html>
    <body>
        <div>
            foo
        </div>
        <form>
            <input type="text" />
            <input type="radio" />
            <input type="checkbox" />
        </form>
    </body>
</html>
```

DOM Tree

```
|-> Document
   |-> Element (<html>)
     |-> Element (<body>)
       |-> Element (<div>)
         |-> text node
       |-> Form
            |-> Text-box
            |-> Radio Button
            |-> Check Box
```

The power of Javascript

Get familiarized with it so that you can think of all the attacks one can do with it.

What can you do with Javascript?

Almost anything you want to the DOM!

A JS script embedded on a page can modify in almost arbitrary ways the DOM of the page.

The same happens if an attacker manages to get you load a script into your page.

w3schools.com has nice interactive tutorials

Example of what Javascript can do…

Can change HTML content:

```
<p id="demo">JavaScript can change HTML content.</p>

<button type="button"
onclick="document.getElementById('demo').innerHTML =
'Hello JavaScript!'">
   Click Me!</button>
```

DEMO from
http://www.w3schools.com/js/js_examples.asp

Other examples

Can change images
Can chance style of elements
Can hide elements
Can unhide elements
Can change cursor...

Basically, can do ***anything it wants*** to the DOM

Another example: can access cookies

Read cookie with JS:

```
var x = document.cookie;
```

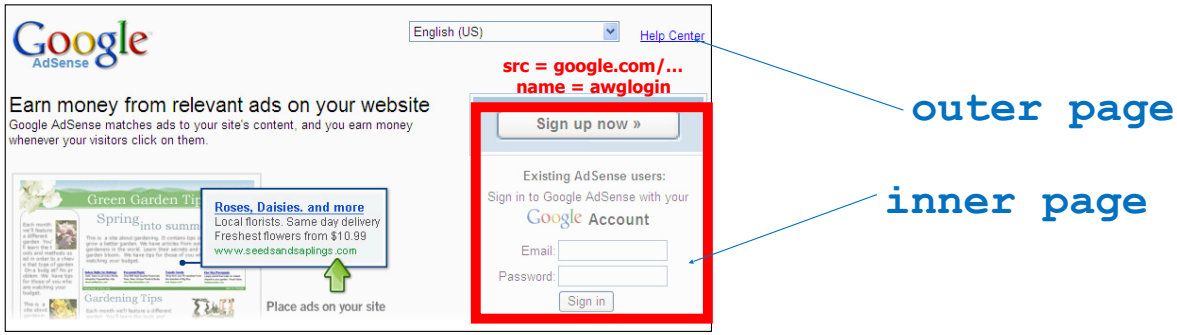
Change cookie with JS:

```
document.cookie = "username=John Smith; expires=Thu, 18
Dec 2013 12:00:00 UTC; path=/";
```

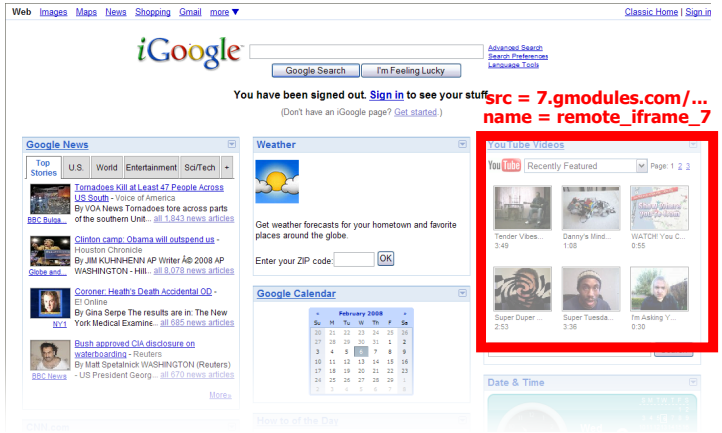
Frames

- Enable embedding a page within a page

```
<iframe src="URL"></iframe>
```



Frames



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

Frames

- Outer page can specify only sizing and placement of the frame in the outer page
- Frame isolation: Outer page cannot change contents of inner page; inner page cannot change contents of outer page

Desirable security goals

- ***Integrity***: malicious web sites should not be able to tamper with integrity of our computers or our information on other web sites
- ***Confidentiality***: malicious web sites should not be able to learn confidential information from our computers or other web sites
- ***Privacy***: malicious web sites should not be able to spy on us or our online activities
- ***Availability***: malicious parties should not be able to keep us from accessing our web resources

Security on the web

- Risk #1: we don't want a malicious site to be able to trash files/programs on our computers
  - Browsing to **`awesomevids.com`** (or **`evil.com`**) should not infect our computers with malware, read or write files on our computers, etc...
  - We generally assume an adversary can cause our browser to go to a web page of the attacker's choosing
- Mitigation strategy
  - Javascript is sandboxed: it is ***not allowed*** to access files etc...
  - Browser code tries to avoid bugs:
    - Privilege separation, automatic updates
    - Reworking into safe languages (rust)

Security on the web

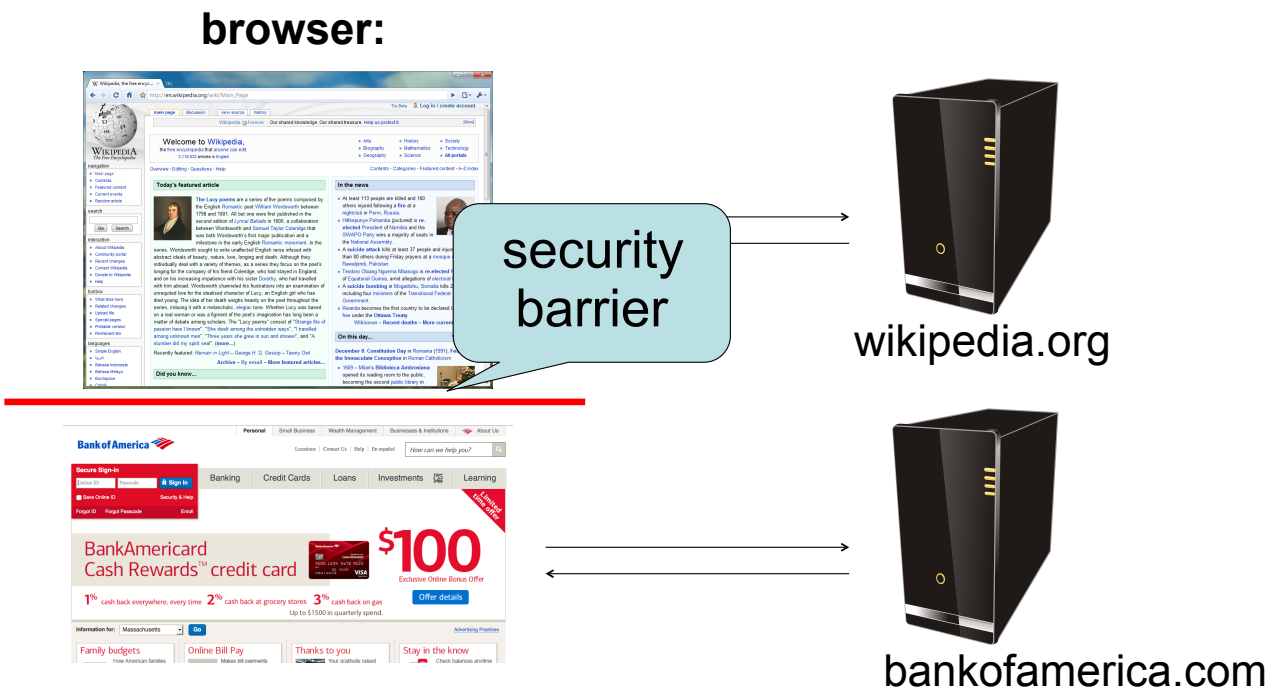
- Risk #2: we don't want a malicious site to be able to spy on or tamper with our information or interactions with other websites
  - Browsing to **`evil.com`** should not let **`evil.com`** spy on our emails in Gmail or buy stuff with our Amazon accounts
- Defense: Same Origin Policy
  - An ***after the fact*** isolation mechanism enforced by the web browser

Security on the web

- Risk #3: we want data stored on a web server to be protected from unauthorized access
- Defense: server-side security

Same-origin policy

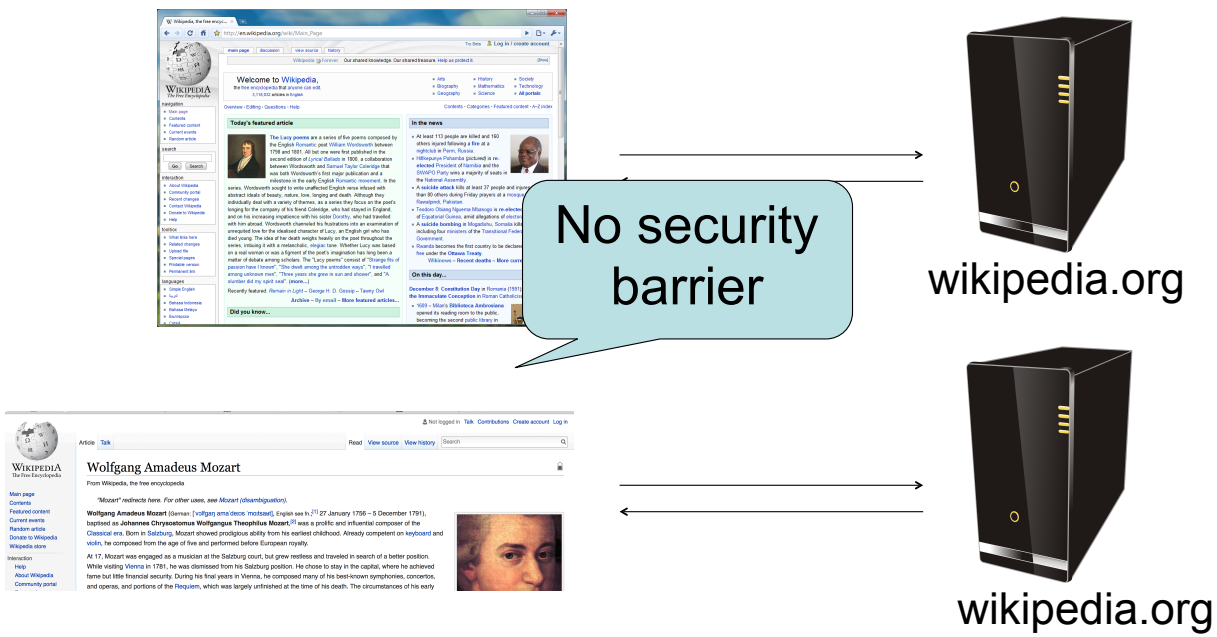
- Each site in the browser is isolated from all others



Same-origin policy

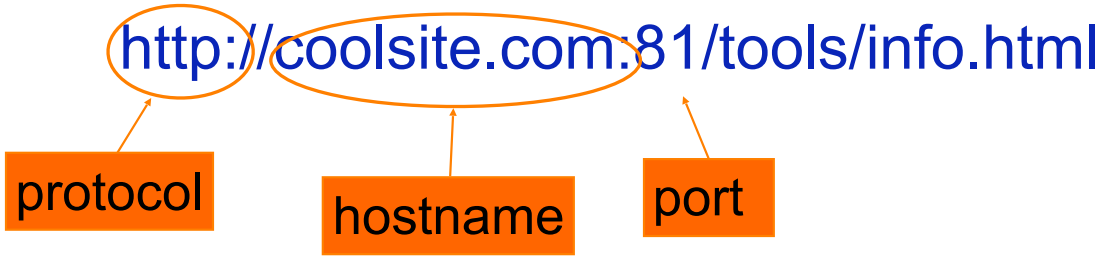
- Multiple pages from the same site are not isolated

browser:



Origin

- Granularity of protection for same origin policy
- Origin = protocol + hostname + port



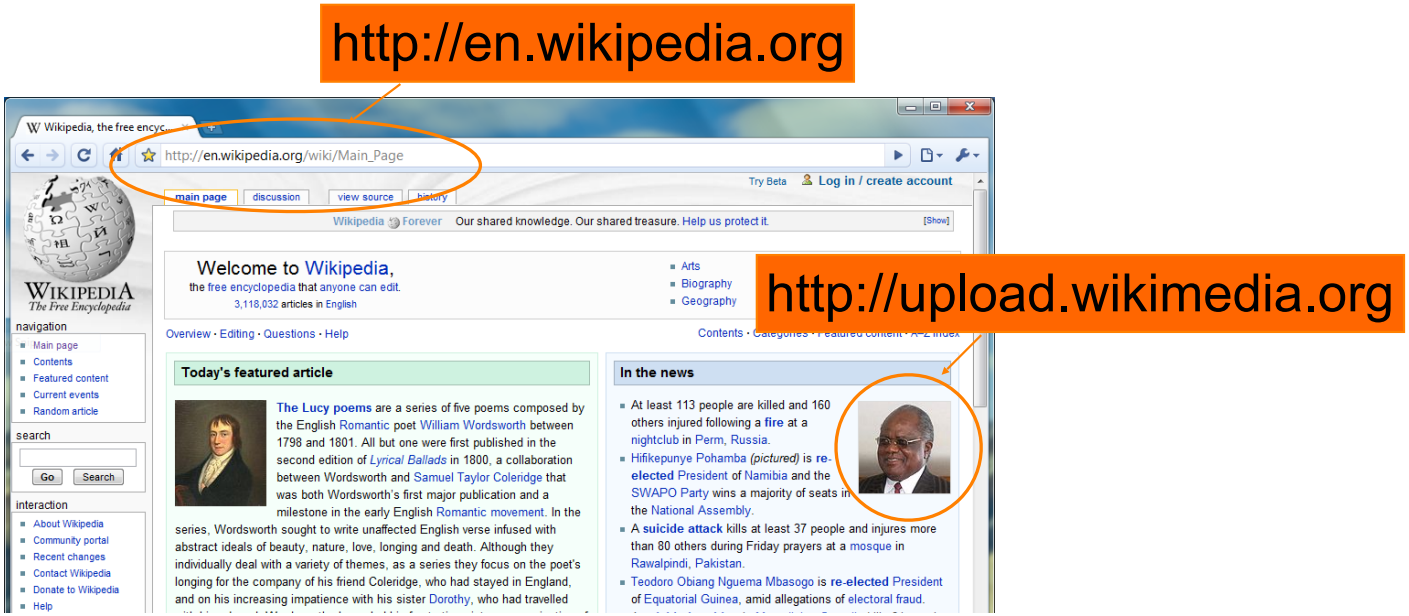
- Determined using ***string matching***! If these match, it is same origin; else it is not. Even though in some cases, it is logically the same origin, if there is no string match, it is not.

Same-origin policy

- One origin should not be able to access the resources of another origin
  - Javascript on one page cannot read or modify pages from different origins.
  - The contents of an iframe have the origin of the URL from which the iframe is served; not the loading website.
-

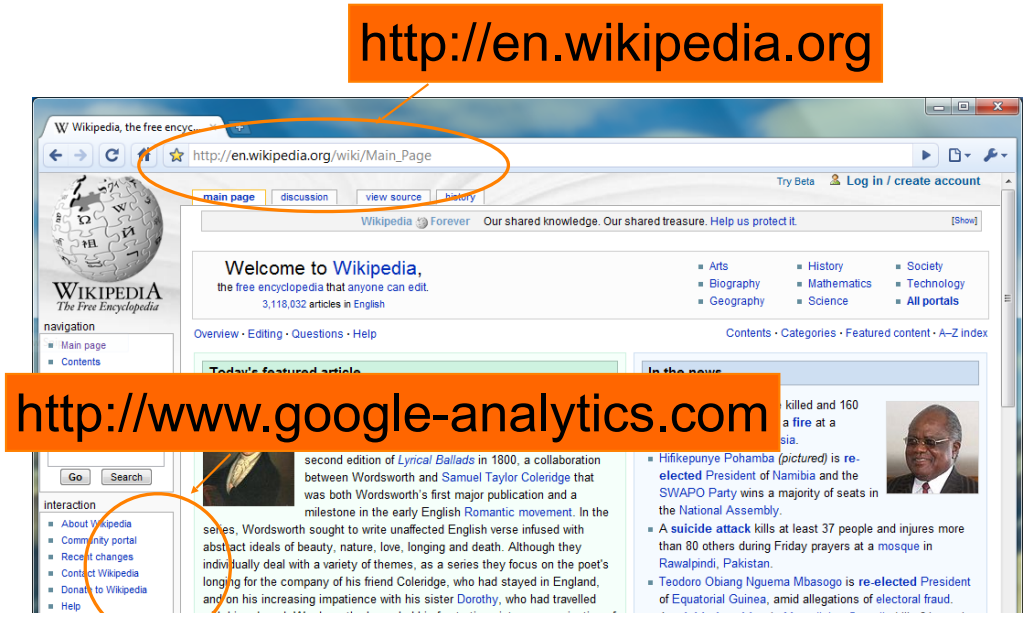
Same-origin policy

- The origin of a page is derived from the URL it was loaded from



Same-origin policy

- The origin of a page is derived from the URL it was loaded from
- Special case: Javascript runs with the origin of the page that loaded it



Assessing SOP

Originating document	Accessed document
http://wikipedia.org/**a**/	http://wikipedia.org/**b**/
http://wikipedia.org/	http://**www.**wikipedia.org/
http://wikipedia.org/	**https**://wikipedia.org/
http://wikipedia.org**:81**/	http://wikipedia.org**:82**/
http://wikipedia.org**:81**/	http://wikipedia.org/

except !

Origins of other components

- **<img src="…">** the image DOM element has the origin of the embedding page, but the image content remains in the remote origin
  - So JavaScript can't read the photo, but sees a black box on the size
- *iframe*: origin of the URL from which the iframe is served; *not* the loading website
  - Data in an iframe from a different origin can not be accessed by the enclosing page's JavaScript

Chromodo

Private Internet Browser

Fast and versatile Internet Browser based on Chromium, with highest levels of speed, security and privacy!

Issue 704: Comodo: Comodo "Chromodo" Browser disables same origin policy, Effectively turning off web security.

13 people starred this issue and may be notified of changes.

tus: Fixed
ner: tav...@google.com
sed: Yesterday
project-...@google.com
ıdor-Comodo
duct-Chromodo
erity-critical

Project Member Reported by tav...@google.com, Jan 21, 2016

When you install Comodo Internet Security, by default a new browser called Chromodo is installed and set as the default browser. Additionally, all shortcuts are replaced with Chromodo links and all settings, cookies, etc are imported from Chrome. They also hijack DNS settings, among other shady practices.

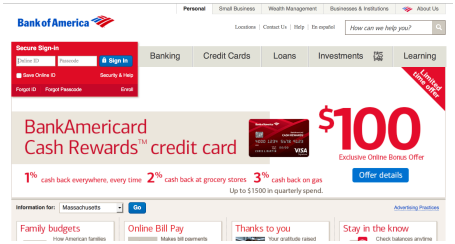
https://www.comodo.com/home/browsers-toolbars/chromodo-private-internet-browser.php

Chromodo is described as "highest levels of speed, security and privacy", but actually disables all web security. Let me repeat that, they ***disable the same origin policy***.... ?!?..

Cross-origin communication

- Allowed through a narrow API: `postMessage`
- Receiving origin decides if to accept the message based on source origin (correctness enforced by browser)

```
postMessage
("run this
script",
script)
```



Check origin, and request!

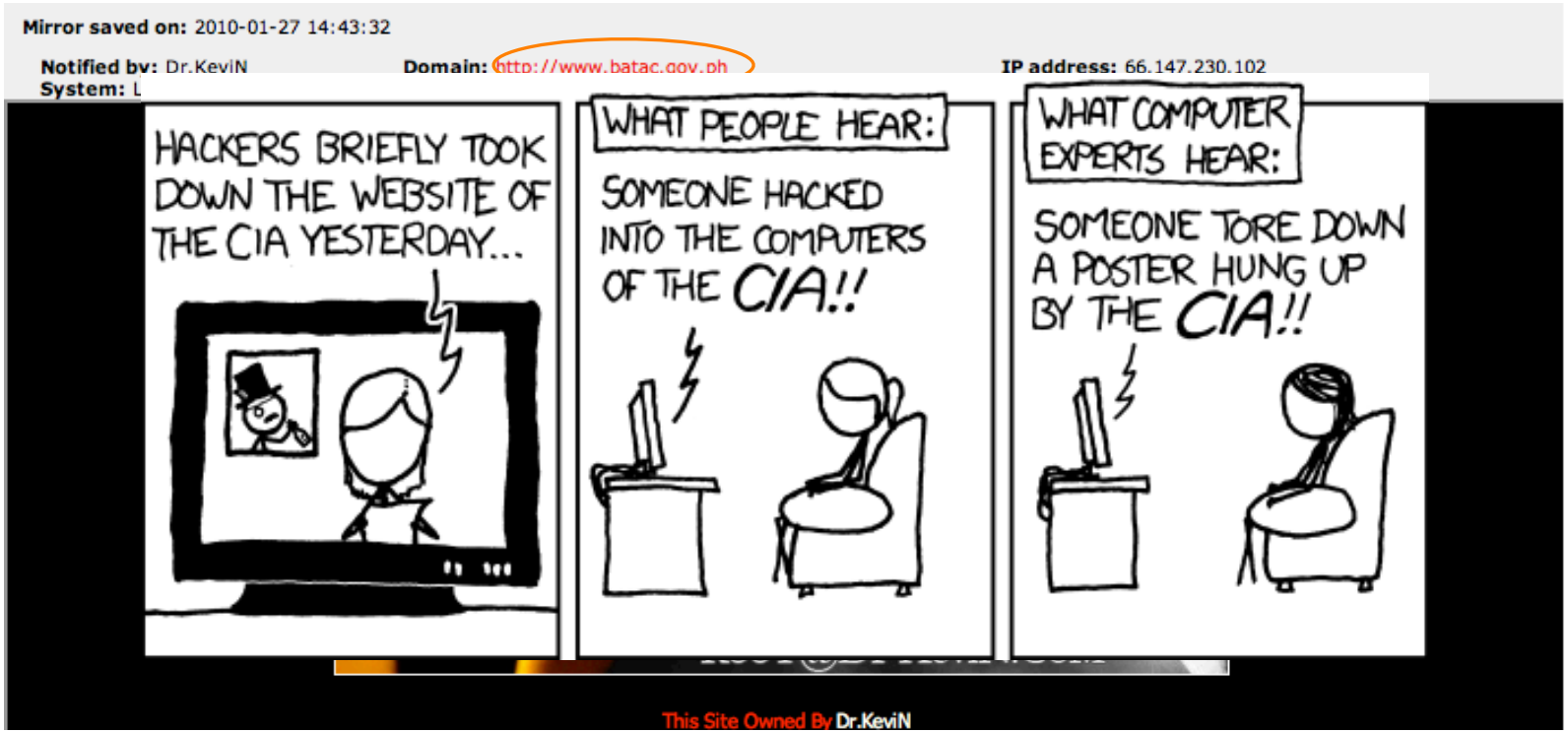
Web Server Threats

- What can happen?
  - Compromise of underlying system
  - Gateway to enabling attacks on clients
  - Disclosure of sensitive or private information
  - Impersonation (of users to servers, or vice versa)
  - Defacement
  - (not mutually exclusive)

Web Server Threats

- What can happen?
  - Compromise of underlying system
  - Gateway to enabling attacks on clients
  - Disclosure of sensitive or private information
  - Impersonation (of users to servers, or vice versa)
  - **Defacement**
  - (not mutually exclusive)

Often Done For Laughs




This Web Site Has Bee...
http://twitter.com/
Facebook
NeoGAF
Google Wave
Twitter
Flickr!
Last.fm
Next Muni
Gmail
Insight @ CCSF
Twitterfall
Other bookmarks
IRANIAN CYBER ARMY
THIS SITE HAS BEEN HACKED BY IRANIAN CYBER ARMY
iRANiAN.CYBER.ARMY@GMAiL.COM

Web Server Threats

- What can happen?
  - Compromise of underlying system
  - Gateway to enabling attacks on clients
  - Disclosure of sensitive or private information
  - Impersonation (of users to servers, or vice versa)
  - Defacement
  - (not mutually exclusive)

- What makes the problem particularly tricky?
  - ***Public access***

zone-h
unrestricted information

Home News Events Archive Archive ★ Onhold Notify Stats Register Login search...

[ENABLE FILTERS]

Total notifications: **143,830** of which **64,954** single ip and **78,876** mass defacements

Legend:
H - Homepage defacement
M - Mass defacement (click to view all defacements of this IP)
R - Redefacement (click to view all defacements of this site)
L - IP address location
★ - Special defacement (special defacements are important websites)

Date	Notifier	H	M	R	L	★	Domain	OS	View
2013/02/21	CLONING		M			★	www.sisaketspecial.go.th/56/un...	Linux	mirror
2013/02/21	CLONING		M			★	www.lalo.go.th/Joomla_1.5.22-S...	Linux	mirror
2013/02/21	CLONING		M			★	www.bareknuea.go.th/attach/unl...	Linux	mirror
2013/02/21	Dr.SHA6H		M	R		★	gallery.unicef.by/workspace/	Linux	mirror
2013/02/21	Dr.SHA6H		M	R		★	kazki.unicef.by/workspace/	Linux	mirror
2013/02/21	Dr.SHA6H			R		★	www.unicef.by/worspace/thumb/l...	Linux	mirror
2013/02/21	NoEntry Phc					★	hmc.ntuh.gov.tw/pwn.html	Win 2003	mirror
2013/02/21	1923Turk			R		★	xj.dzgtj.gov.cn/aL_Pars.htm	Win 2003	mirror
2013/02/21	1923Turk					★	gsl.cznq.gov.cn/aL_Pars.htm	Win 2003	mirror
2013/02/21	RainsevenDotMy		M	R		★	www.thapo.go.th/images/news/	Linux	mirror
2013/02/21	RainsevenDotMy		M	R		★	www.krc.go.th/images/personnel/	Linux	mirror

Web Server Threats

- What can happen?
  - Compromise of underlying system
  - Gateway to enabling attacks on clients
  - Disclosure of sensitive or private information
  - Impersonation (of users to servers, or vice versa)
  - Defacement
  - (not mutually exclusive)

- What makes the problem particularly tricky?
  - Public access
  - ***Mission creep***

HP LaserJet 8150 Series

http://128.3. /hp/device/this.LCDispatcher

Most Visited | Latest Headlines | NY Times | Google News | Daily | Weather | 294 | United | Traffic | Papers | US9 | IMC | CSET | Google Maps | RSS

HP LaserJet 8150 Series

invent

HP LaserJet 8150 Series / 128.3.

HP LaserJet 8150 Series

Home | Device | Networking

- Printer Status
- Configuration Page
- Supplies Status
- Event Log
- Usage Page
- Device Information

Other Links
My Printer
Order Supplies
Solve A Problem

Printer Status

Supplies | Media | Capabilities

Control Panel

POWERSAVE ON

Ready | Data | Attention

Go | Cancel Current Job

Control Panel Help
Refresh Control Panel
Help

Set Refresh Rate:
0 minutes
Apply | Cancel

Supplies

	% of Life Remaining
Black	54%

Media

Status	Input/Output	Size	Type
	TRAY 3	LETTER	CARDSTOCK
	TRAY 2	LETTER	PLAIN
	TRAY 1	LETTER	PLAIN
OK	STANDARD OUTBIN	N/A	N/A
OK	FACE UP BIN	N/A	N/A

Capabilities

FLASH Storage: 3 MB Capacity

Done

LaCie Ethernet Disk mini

v. 2.0

5.2. Accessing the LaCie Ethernet Disk mini via Web Browsers

While the LaCie Ethernet Disk mini is connected to the network, it is capable of being accessed via the Internet through your Internet browser.

Windows, Mac and Linux Users – Open your browser to `http://EDmini` or `http://device_IP_address` (the "device_IP_address" refers to the IP address that is assigned to your LaCie Ethernet Disk mini; for example, `http://192.168.0.207`).

Samsung SPF-85V 8-Inch Wireless Internet Photo Frame USB Mini-PC Monitor w/64MB Memory (Black)

by Samsung

(6 customer reviews) Like (0)

Available from these sellers.

1 used from $129.95

What Do Customers Ultimately Buy After Viewing This Item?

30% buy
Kodak Pulse 7-Inch Digital Frame (128)
Click to see price

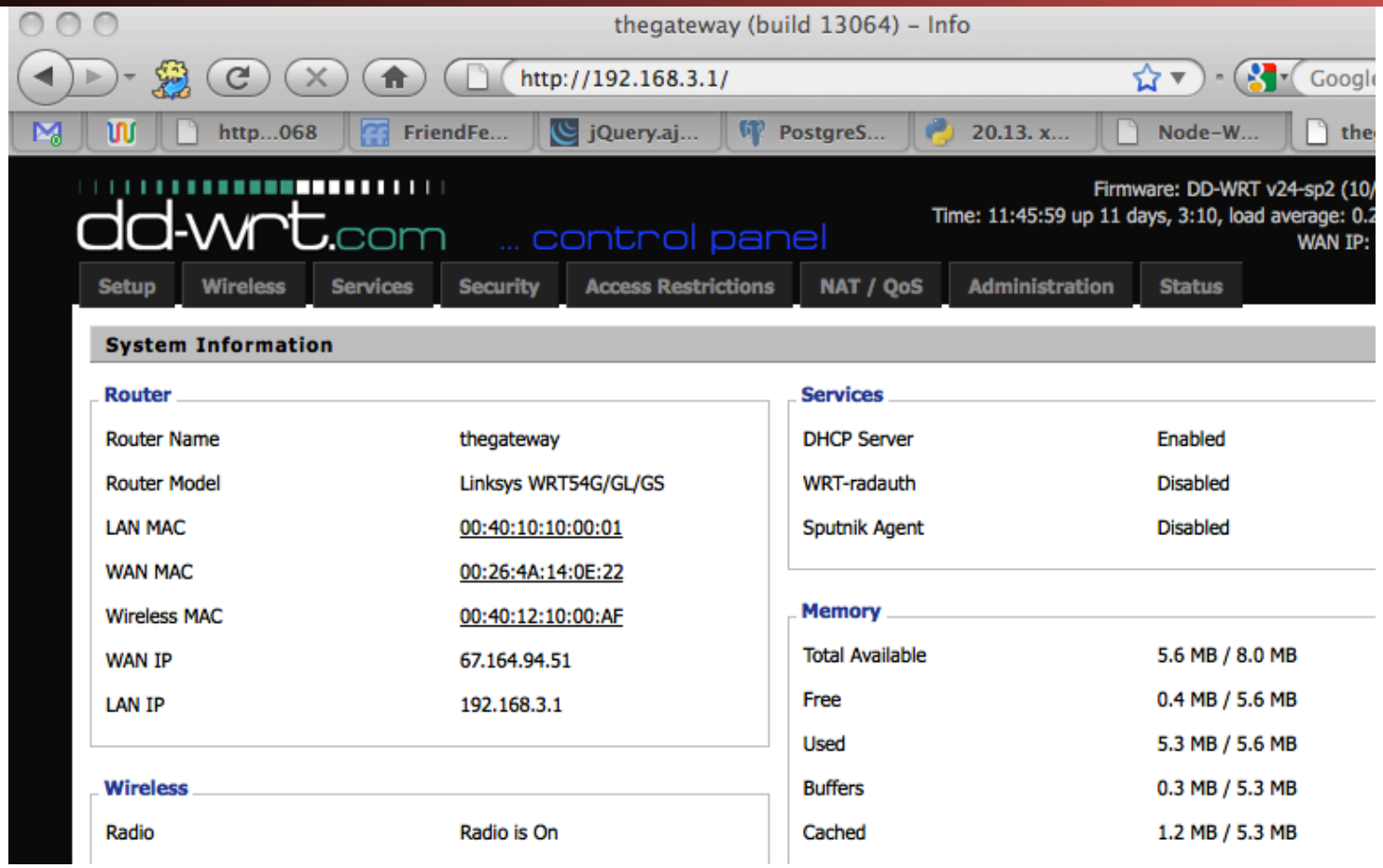
30% buy
Toshiba DMF102XKU 10-Inch Wireless Digital Media Frame (25)
$159.99

(1) There's a web interface for the frame- you use a web browser on your network that connects to the picture frame. The web interface is horrendously slow and repeatedly "times out" while trying to access the frame.

5

Using the Web Interface

Your Cisco IP Phone provides a web interface to the phone that allows you to configure some features of your phone using a web browser. This chapter contains the following sections:


thegateway (build 13064) - Info
http://192.168.3.1/
dd-wrt.com ... control panel
Firmware: DD-WRT v24-sp2 (10/
Time: 11:45:59 up 11 days, 3:10, load average: 0.2
WAN IP:
Setup
Wireless
Services
Security
Access Restrictions
NAT / QoS
Administration
Status
System Information
Router
Router Name thegateway
Router Model Linksys WRT54G/GL/GS
LAN MAC 00:40:10:10:00:01
WAN MAC 00:26:4A:14:0E:22
Wireless MAC 00:40:12:10:00:AF
WAN IP 67.164.94.51
LAN IP 192.168.3.1
Services
DHCP Server Enabled
WRT-radauth Disabled
Sputnik Agent Disabled
Memory
Total Available 5.6 MB / 8.0 MB
Free 0.4 MB / 5.6 MB
Used 5.3 MB / 5.6 MB
Buffers 0.3 MB / 5.3 MB
Cached 1.2 MB / 5.3 MB
Wireless
Radio Radio is On

Setup/Configuration	
Web user interface	Built-in web user interface for easy browser-based configuration (HTTP)
Management	
Web browser	• Internet Explorer 5.x or later • Limited support for Netscape and Firefox. Browser controls for pan/tilt/zoom (PTZ), audio, and motion detection are limited or not supported with Netscape and Firefox.
Event logging	Event logging (syslog)
Web firmware upgrade	Firmware upgradable through web browser

*Security*Tracker Archives

Category: Application (Security) > Cisco Security Agent Vendors: Cisco

Cisco Security Agent Web Management Interface Bug Lets Remote Users Execute Arbitrary Code

SecurityTracker Alert ID: 1025088

SecurityTracker URL: http://securitytracker.com/id/1025088

CVE Reference: CVE-2011-0364 *(Links to External Site)*

Date: Feb 16 2011

Impact: Execution of arbitrary code via network, User access via network

Fix Available: Yes **Vendor Confirmed:** Yes

Version(s): 5.1, 5.2, and 6.0

Description: A vulnerability was reported in Cisco Security Agent. A remote user can execute arbitrary code on the target system.

A remote user can send specially crafted data to the web management interface on TCP port 443 to execute arbitrary code on the target system. This can be exploited to modify agent policies and the system configuration and perform other administrative tasks.

Cisco has assigned Cisco Bug ID CSCtj51216 to this vulnerability.

Gerry Eisenhaur reported this vulnerability via ZDI.

Impact: A remote user can execute arbitrary code on the target system.

Solution: The vendor has issued a fix (6.0.2.145)

Interacting With Web Servers

- An interaction with a web server is expressed in terms of a URL (plus an optional data item)
- URL components:

http://coolsite.com/tools/info.html

Path to a *resource*

Here, the resource ("`info.html`") is **static content** = a fixed file returned by the server.

(Often static content is an *HTML* file = content plus markup for how browser should "render" it.)

Interacting With Web Servers

- An interaction with a web server is expressed in terms of a URL (plus an optional data item)
- URL components:

http://coolsite.com/tools/doit.php

Path to a *resource*

Resources can instead be **dynamic**
= server generates the page on-the-fly.

Some common frameworks for doing this:
CGI = run a program or script, return its *stdout*
PHP = execute script in HTML templating language
(PHP means PHP HTML Preprocessor)

Interacting With Web Servers

- An interaction with a web server is expressed in terms of a URL (plus an optional data item)
- URL components:

http://coolsite.com/tools/doit.php?cmd=play&vol=44

URLs for dynamic content generally include **arguments** to pass to the generation process

Interacting With Web Servers

- An interaction with a web server is expressed in terms of a URL (plus an optional data item)
- URL components:

http://coolsite.com/tools/doit.php?cmd=play&vol=44

First *argument* to doit.php

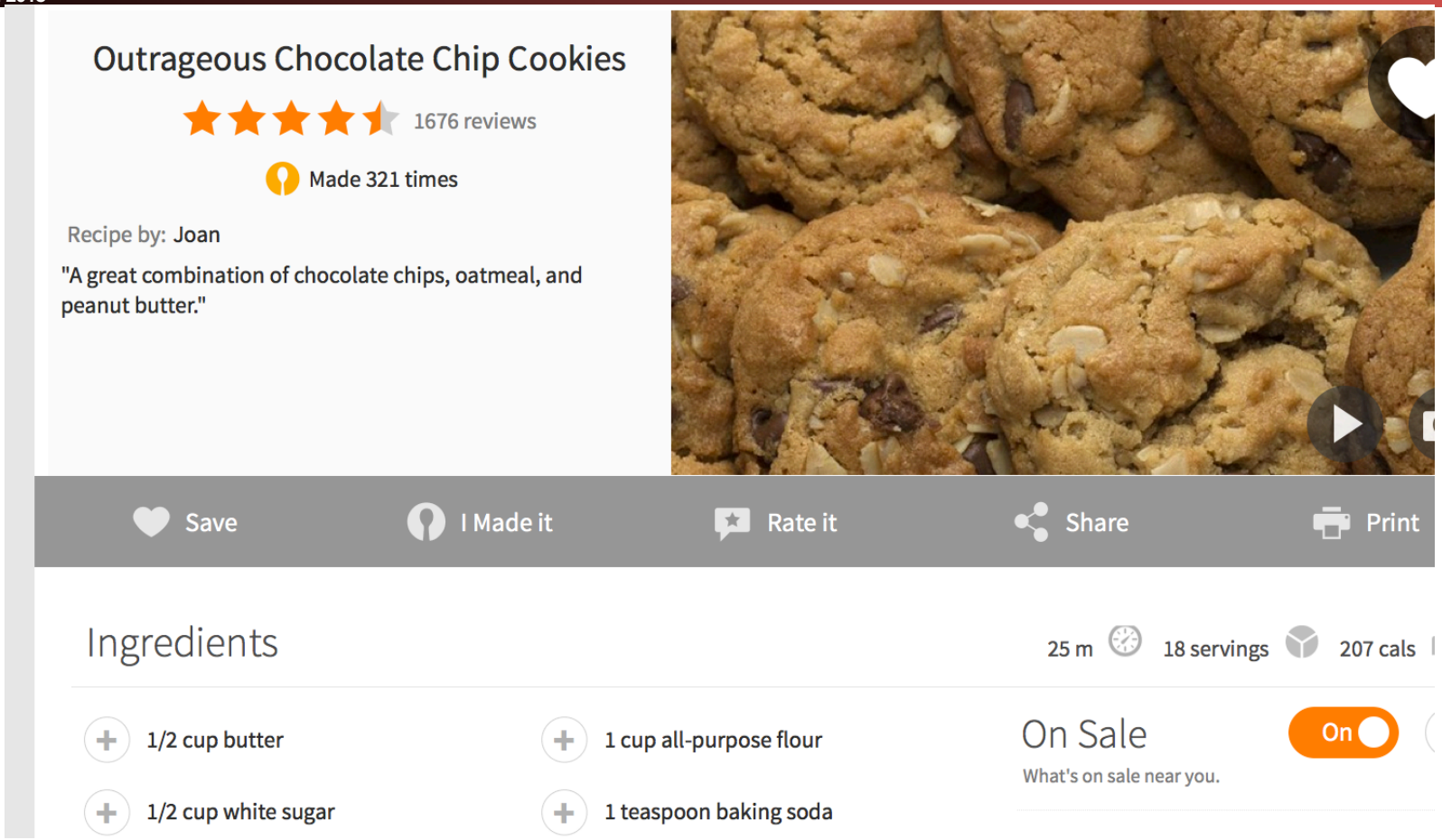
Interacting With Web Servers

- An interaction with a web server is expressed in terms of a URL (plus an optional data item)
- URL components:

http://coolsite.com/tools/doit.php?cmd=play&vol=44

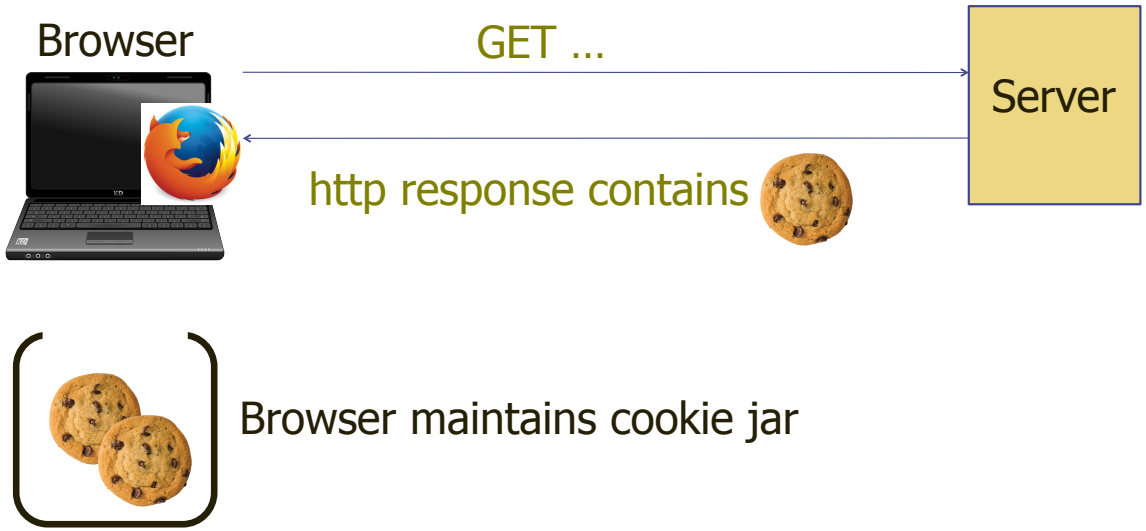
Second *argument* to doit.php

HTTP cookies

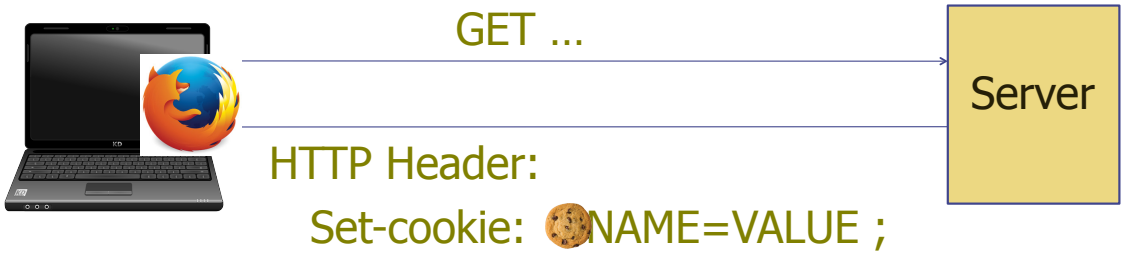


Cookies

- A way of maintaining state



Setting/deleting cookies by server



- The first time a browser connects to a particular web server, it has no cookies for that web server
- When the web server responds, it includes a Set-Cookie: header that defines a cookie
- Each cookie is just a name-value pair

View a cookie

- In a web console (firefox, tool->web developer->web console), type: **`document.cookie`**
- to see the cookie(s) for that site

Well, its not ***quite*** a name/value pair...

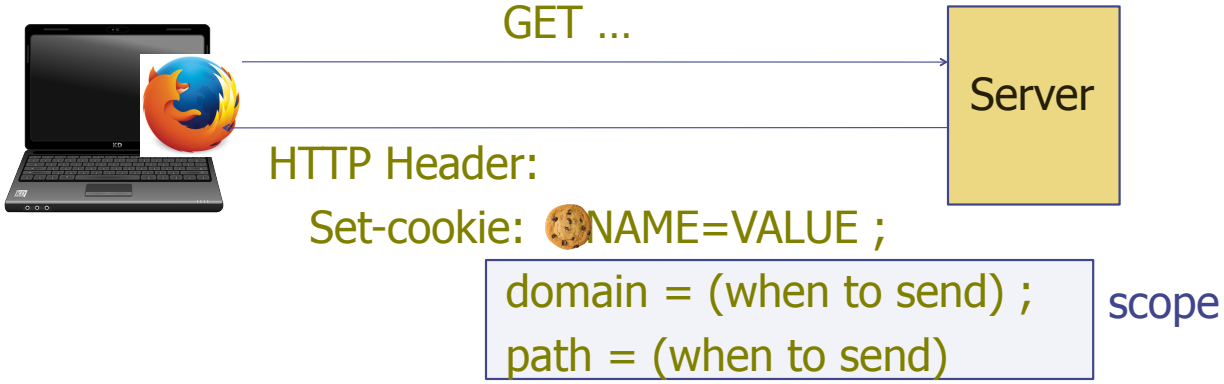
- Cookies are ***read*** by name/value pair
  - Presented to the web server or accessed in JavaScript
- But cookies are ***set*** by name/value/path
  - Both domain-path (foo.com, www.foo.com) and URL path (/pages/)
- Cookies are made available when the paths match
  - www.foo.com can read foo.com's cookies...
  - But foo.com can't read cookies pathed to www.foo.com
- A couple of other flags:
  - ***secure***: Can only be transmitted over an encrypted connection
  - ***HttpOnly***: Will be transmitted to the web server but ***not*** accessible to JavaScript

Cookie ***snooping and stuffing...***

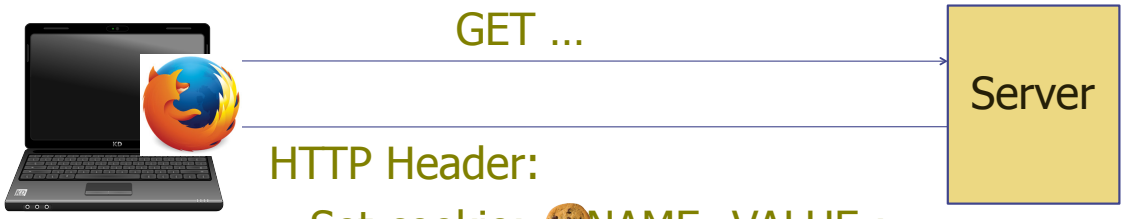
- An adversary is on your local wireless network...
  - And can therefore see all unencrypted (non-HTTPS) traffic
- They can snoop all unencrypted cookies
  - And since that is the state used by the server to identify a returning user... they can act as that user
  - ***Firesheep***: A utility to snag unencrypted cookies and then use them to impersonate others
- They can inject code into your browser
  - Enables ***setting*** (stuffing) cookies
    - State can cause problems with the server later on...
  - Can ***force*** the browser to reveal all non-secure cookies

Cookie scope

- When the browser connects to the same server later, it includes a Cookie: header containing the name and value, which the server can use to connect related requests.
- Domain and path inform the browser about which sites to send this cookie to



Cookie scope



HTTP Header:

Set-cookie: NAME=VALUE ;

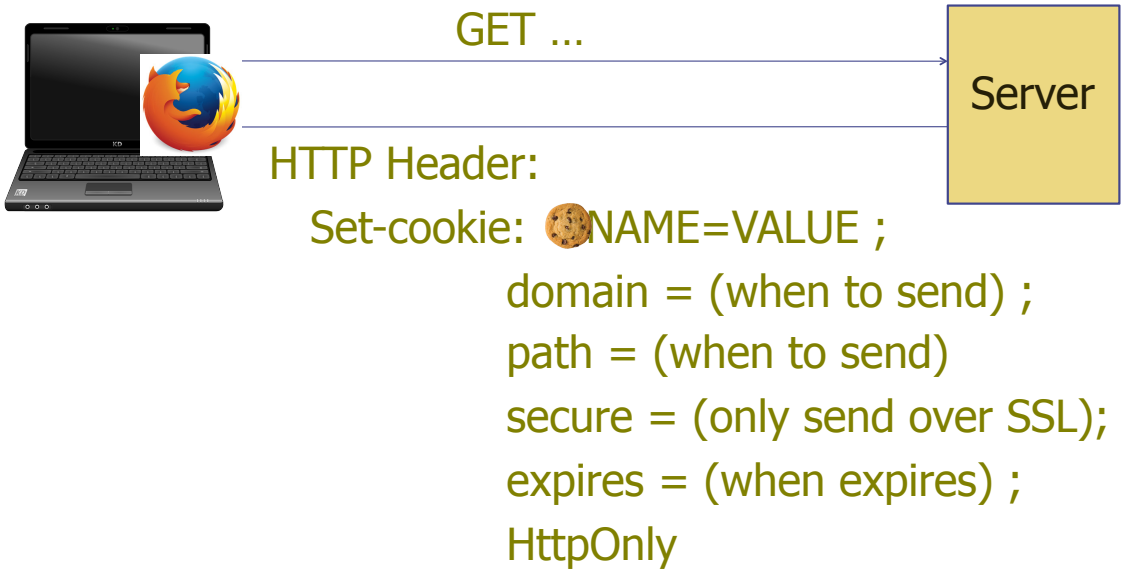
domain = (when to send) ;

path = (when to send)

secure = (only send over HTTPS);

- Secure: sent over HTTPS only
  - HTTPS provides secure communication (privacy and integrity)

Cookie scope



- Expires is expiration date
- HttpOnly: cookie cannot be accessed by Javascript, but only sent by browser

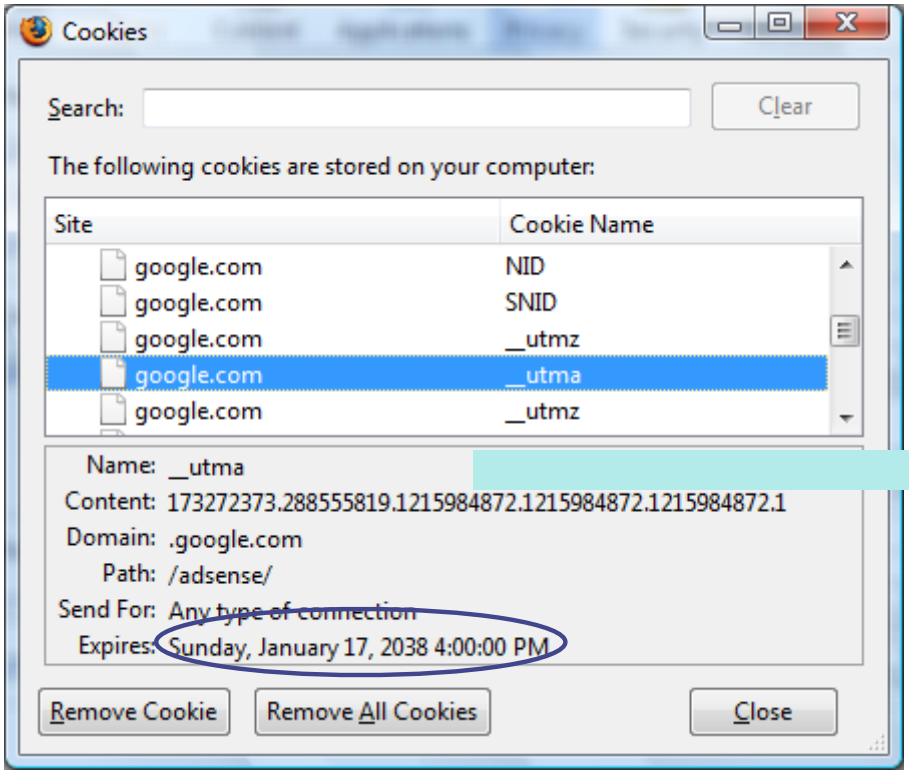
Client side read/write: document.cookie

- Setting a cookie in Javascript:
  `document.cookie = "name=value;  expires=…; "`
- Reading a cookie: **`alert(document.cookie)`**
  - prints string containing all cookies available for document (based on [protocol], domain, path)
- Deleting a cookie: write with an expiration date in the past:
  `document.cookie =  "name=;  expires= Thu, 01-Jan-70"`

document.cookie often used to customize page in Javascript

Viewing/deleting cookies in Browser UI

Firefox: Tools -> page info -> security -> view cookies



Cookie scope

- Scope of cookie might not be the same as the URL-host name of the web server setting it

- Rules on:
  - What scopes a URL-host name is allowed to set
  - When a cookie is sent to a URL

What scope a server may set for a cookie

- domain: any domain-suffix of URL-hostname, except TLD
  - Browser has a list of Top Level Domains (e.g. .com, .co.uk)
- example: host = "login.site.com"

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

- login.site.com can set and read cookies for all of .site.com but not for another site or TLD
  - Mistakenly assumes that subdomains are controlled by the same ownership:
    - This doesn't hold for domains like berkeley.edu
- path: can be set to anything

Examples

Web server at foo.example.com wants to set cookie with domain:

domain	Whether it will be set, and if so, where it will be sent to
(value omitted)	*foo.example.com* (exact)
bar.foo.example.com	
foo.example.com	*.*foo.example.com*
baz.example.com	
example.com	
ample.com	
.com	

Credits: The Tangled Web: A Guide to Securing Modern Web Applications, by Michał Zalewski

Examples

Web server at foo.example.com wants to set cookie with domain:

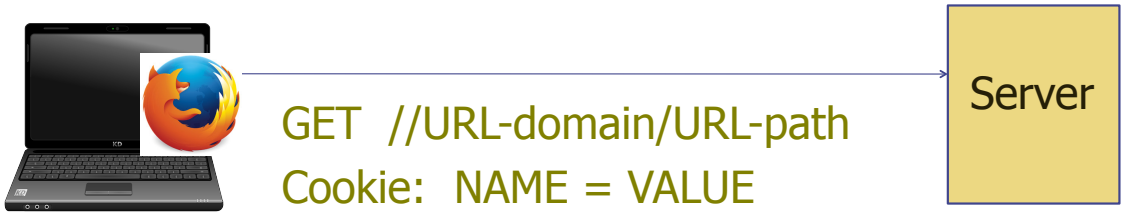
domain	Whether it will be set, and if so, where it will be sent to
(value omitted)	*foo.example.com* (exact)
bar.foo.example.com	Cookie not set: domain more specific than origin
foo.example.com	*.*foo.example.com*
baz.example.com	Cookie not set: domain mismatch
example.com	*.*example.com*
ample.com	Cookie not set: domain mismatch
.com	Cookie not set: domain too broad, security risk

Credits: The Tangled Web: A Guide to Securing Modern Web Applications, by Michał Zalewski

When browser sends cookie

Browser sends all cookies in URL scope:

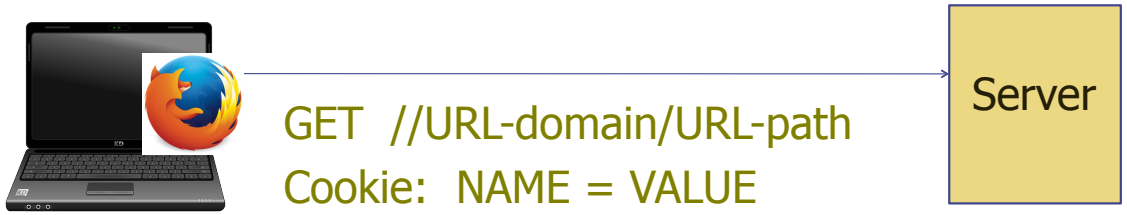
- cookie-domain is domain-suffix of URL-domain, and
- cookie-path is prefix of URL-path, and
- [protocol=HTTPS if cookie is “secure”]



Goal: server only sees cookies in its scope

When browser sends cookie

- A cookie with
- domain = example.com, and
- path = /some/path/
- will be included on a request to
- http://foo.example.com/some/path/subdirectory/hello.txt



Examples: Which cookie will be sent?

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

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

http://checkout.site.com/	cookie: userid=u2
http://login.site.com/	cookie: userid=u1, userid=u2
http://othersite.com/	cookie: none

Reflection on a problem...

- The presentation to the server (and to JavaScript) is just name/value...
  - But sent and set based on name/value/domain/path
  - And in ***unspecified order***
- And (until recently...), HTTP connections could ***set*** cookies flagged with secure
  - Create shadowing opportunities
- Can use to create "land-mine cookies"
  - Embed an attack in a cookie when someone is on the same wireless network...
  - "Cookies lack integrity, real world implications"