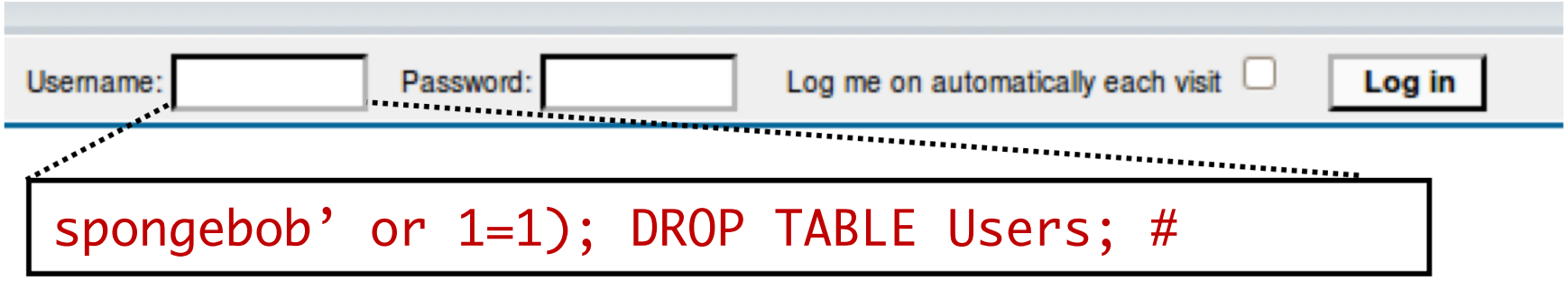
CS 88: Security and Privacy

08: Web Security: HTTP and Cookies

02-15-2024

slides adapted from Dave Levine, Vitaly Shmatikov, Christo Wilson

SQL Injection



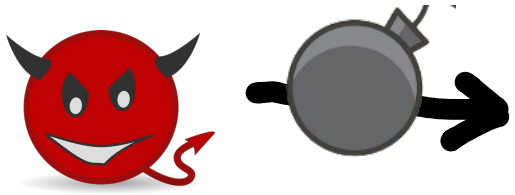

```
$result = mysql_query("select * from Users
        where(name='$user' and password='$pass');");

$result = mysql_query("select * from Users
        where(name='spongebob' or 1=1);#
        DROP TABLE Users; --
        ' and password='whocares');");
```

Can chain together statements, and can modify existing statements

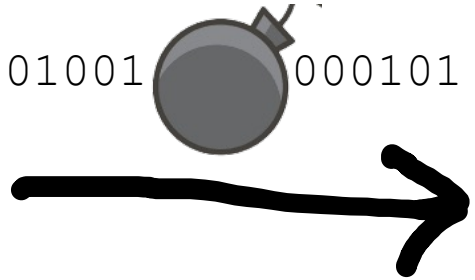
Not Just SQL!

User



Front end

Forms a string containing user input



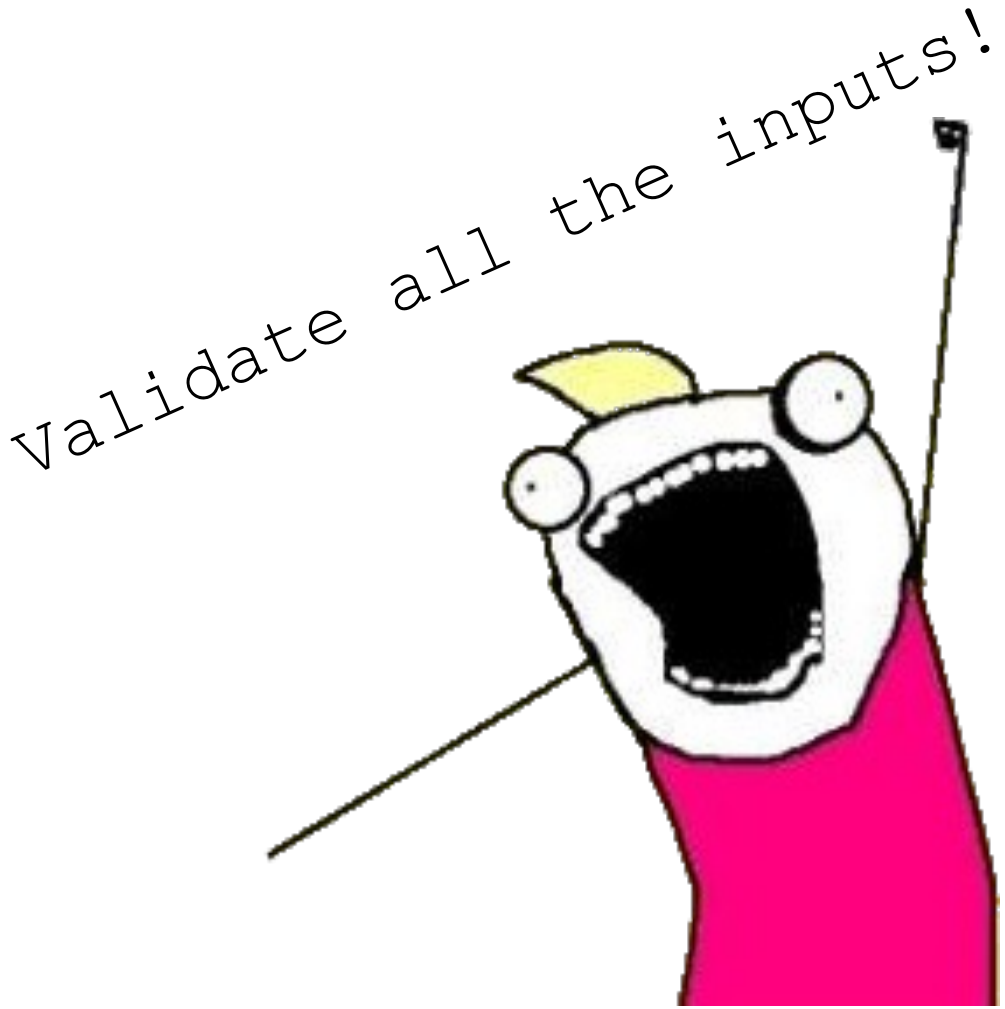
Back end

Executes this string as a command or query

Database
NoSQL storage
Javascript
eval(…)

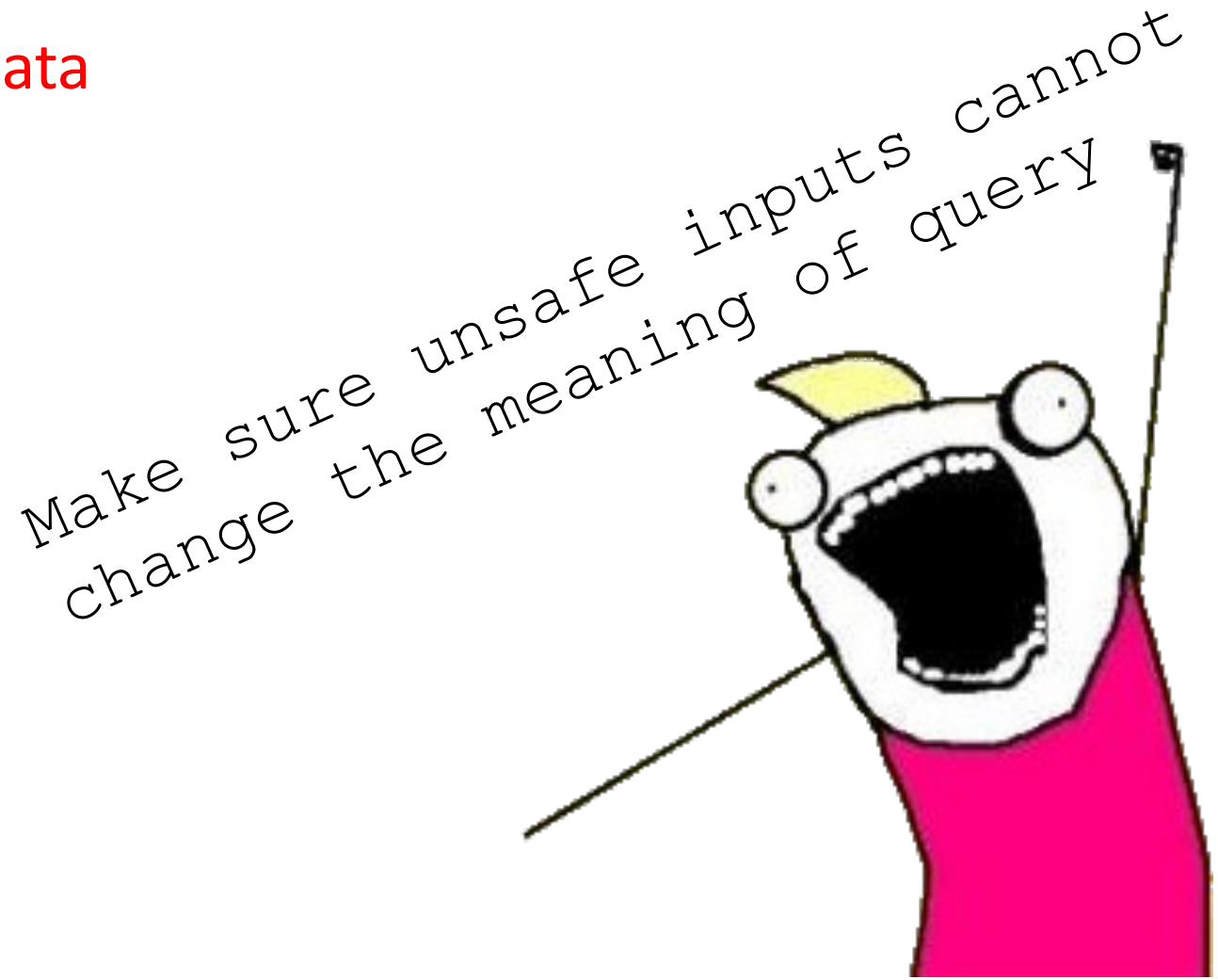
Injection vulnerabilities are a generic issue!

PREVENTING INJECTION ATTACKS

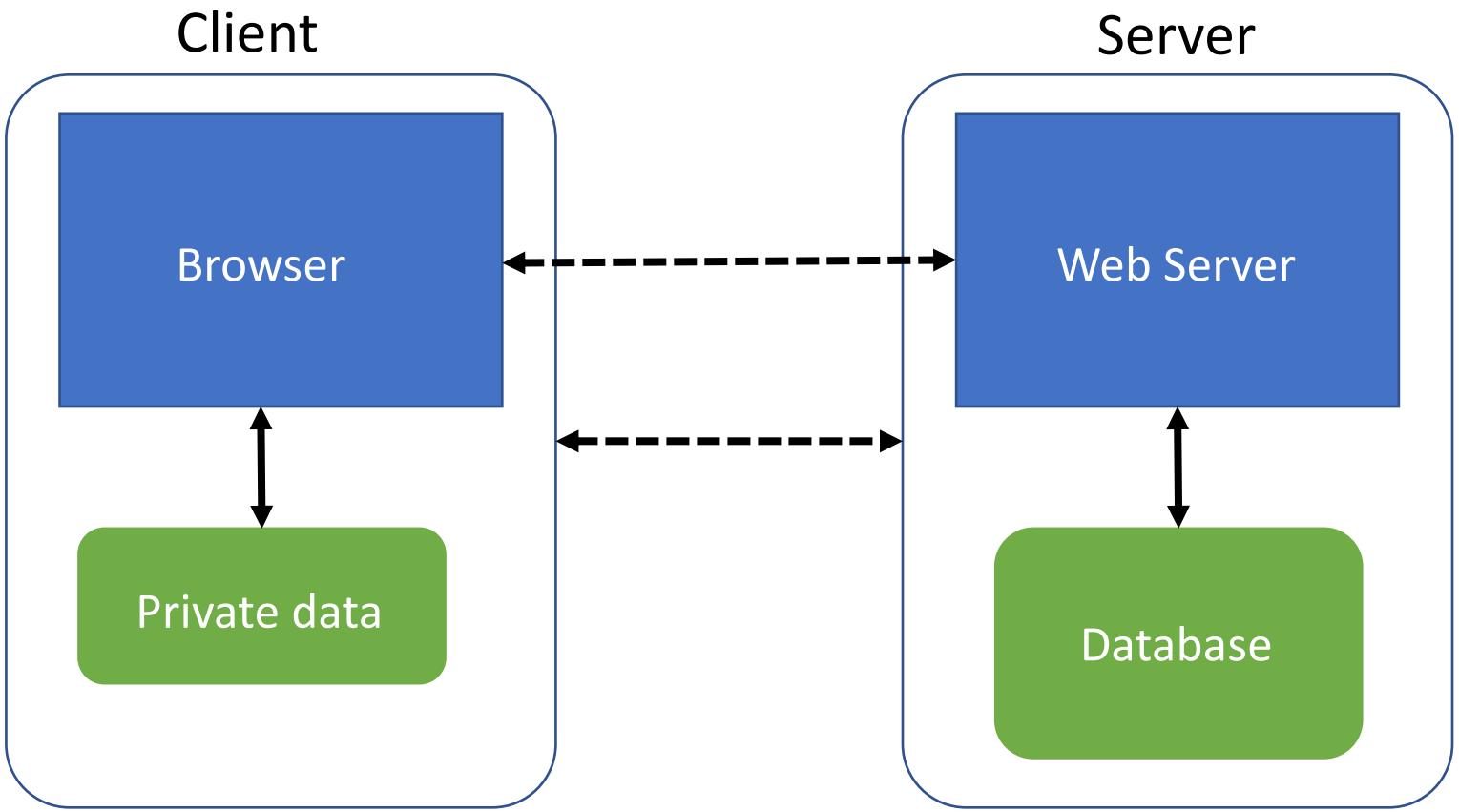


Most injection attacks trick application into interpreting data as code

This changes the semantics of a query or command generated by the application



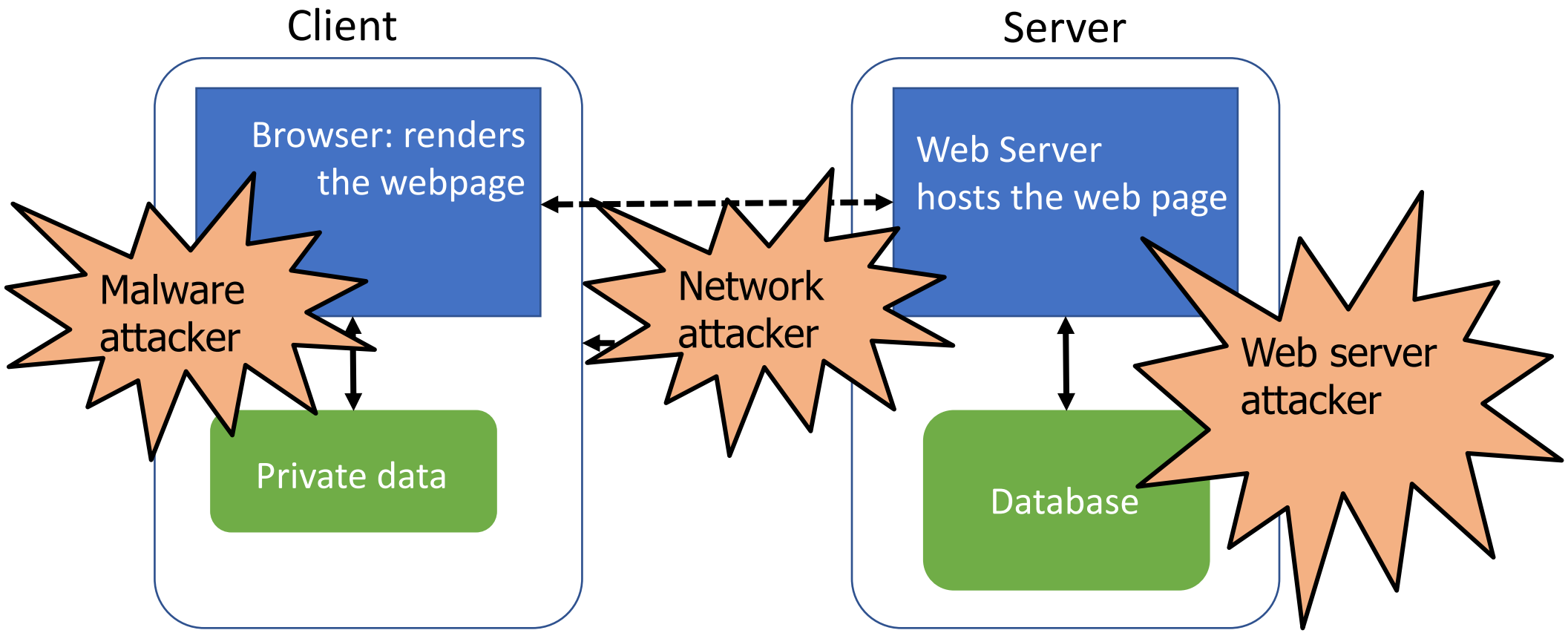
A basic web architecture



Much of the user data is part of the browser

DB is a separate entity, logically (and often physically)

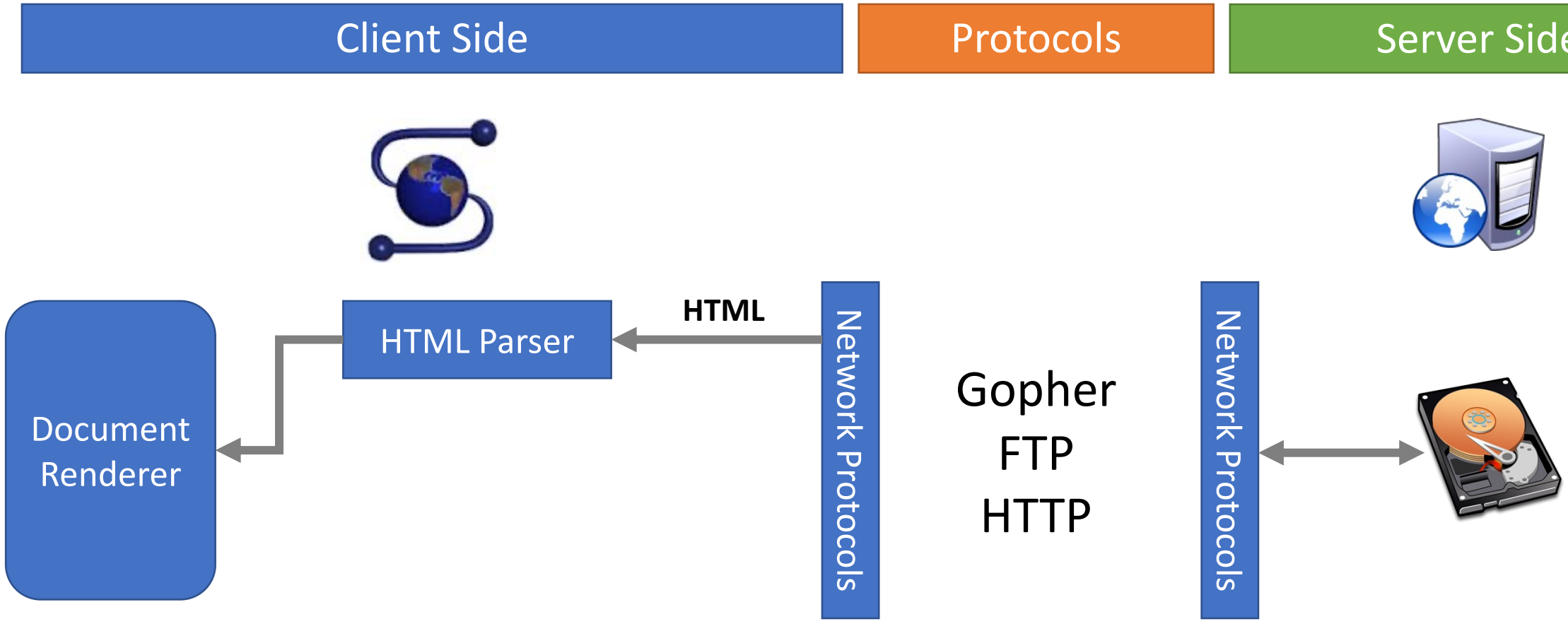
Where Does the Attacker Live?



Much of the user data is part of the browser

DB is a separate entity, logically (and often physically)

Web Architecture: Simplified View



Web Browser
Responsible for securely confining Web content presented by visited websites

Web servers: Responsible for securely parsing input data

PHP, Ruby, ASP, JSP

Web Applications

- Big trend: software as a Web-based service
  - Online banking, shopping, government, bill payment, tax prep, customer relationship management, etc.
  - Cloud-hosted applications
- Application code split between client and server
  - Client (Web browser): JavaScript
  - Server: PHP, Ruby, Java, Perl, ASP …
- Security is rarely the main concern
  - Poorly written scripts with inadequate input validation
  - Inadequate protection of sensitive data

Top Web Vulnerabilities

- SQL injection
  - Malicious data sent to a website is interpreted as code in a query to the website's back-end database
- XSRF (CSRF) - cross-site request forgery
  - Bad website forces the user's browser to send a request to a good website
- XSS (CSS) – cross-site scripting
  - Malicious code injected into a trusted context (e.g., malicious data presented by a trusted website interpreted as code by the user's browser)

Overview

- The Web Model
  - What components make up today's browsers and web servers?
  - How has this functionality evolved over time?
  - What security model governs the browser?

Overview: The Web Model

- What is the web?
- What components make up today's browsers and web servers?
- How has this functionality evolved over time?
- What security model governs the web browser?

What is the web?

- **Web (World Wide Web)**: A collection of data and services
  - Data and services are provided by **web servers**
  - Data and services are accessed using **web browsers** (e.g. Chrome, Firefox)
- The web is not the Internet
  - The Internet describes *how* data is transported between servers and browsers

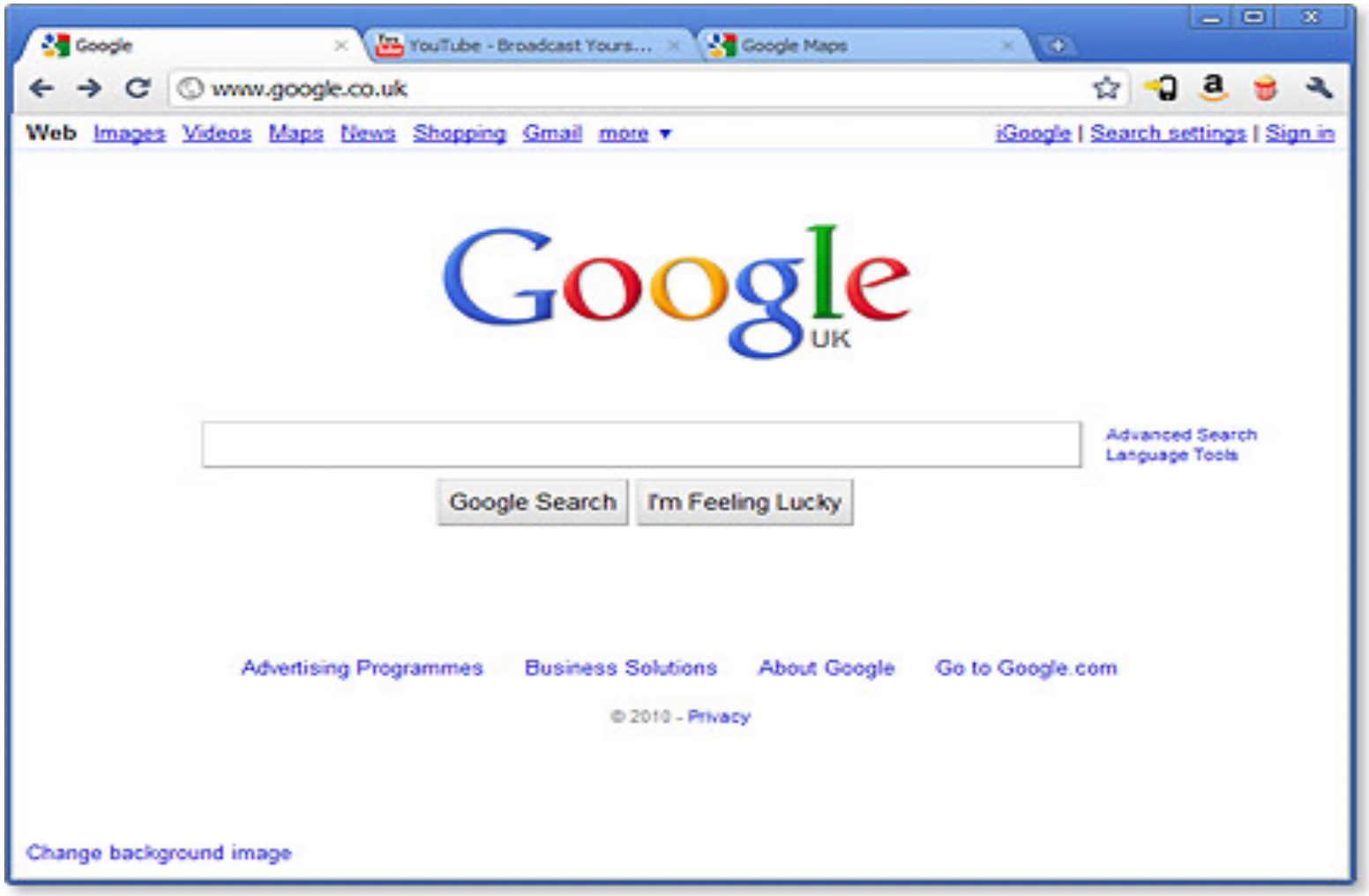
Elements of the Web

- **URLs**: How do we uniquely identify a piece of data on the web?
- **HTTP**: How do web browsers communicate with web servers?
- Data on the webpage can contain:
  - HTML: A markup language for static webpages
  - CSS: A style sheet language for defining the appearance of webpages
  - Javascript: a programming language for running code in the web browser

Elements of the Web

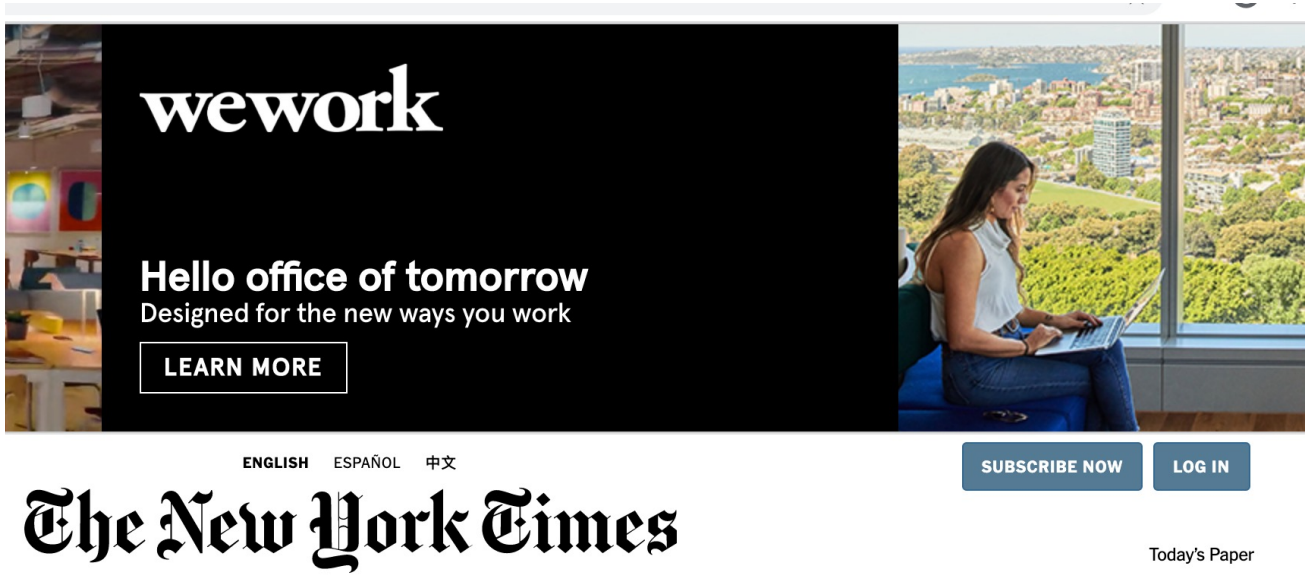
- Data on the webpage can contain:
  - HTML: A markup language for static webpages
  - CSS: A style sheet language for defining the appearance of webpages
  - Javascript: a programming language for running code in the web browser

What IS A Web Browser?

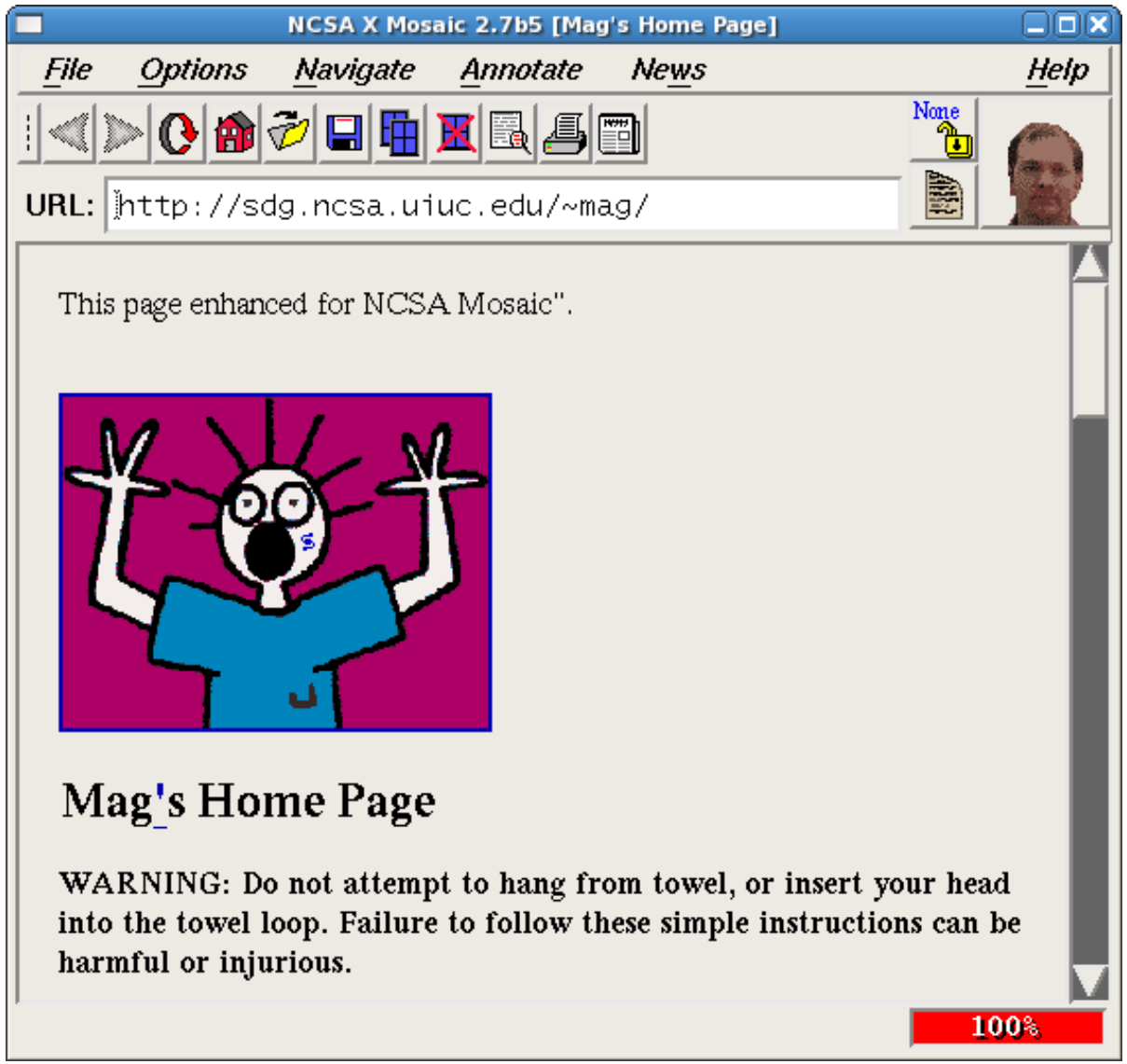


Web Browser: Basic Execution Model

- Each browser window or frame:
  - Loads content
  - Renders:
    - Processes HTML and scripts to display the page
    - May involve images, subframes, etc.
  - Responds to events
- Events
  - User actions: OnClick, OnMouseover
  - Rendering: OnLoad, OnUnload
  - Timing: setTimeout(), clearTimeout()



Generating a static webpage: HTML



HTML

```
<!doctype html>

<html>
<head>
    <title>Hello World</title>
</head>
    <body>
        <h1>Hello World</h1>
     <img src="/img/my_face.jpg"></img>
        <p>
            I am 12 and what is
            <a href="wierd_thing.html">this</a>?
        </p>
        <img src="http://www.images.com/cats/adorablekitten.jpg"></img>
    </body>
</html>
```

HTML may embed other resources from the same origin

… or from other origins (cross origin embedding)

JavaScript

"Java is to JavaScript as car is to carpet"

- Language executed by the browser
  - Scripts are embedded in Web pages
- Inline
  - `<a onclick="doSomething();"></a>`
- Embedded
  - `<script>alert('Hello');</script>`
- External
  - `<script src="/js/main.js"></script>`
- Potentially malicious website gets to execute some code on user's machine

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

Elements of the Web

- **URLs**: How do we uniquely identify a piece of data on the web?
- **HTTP**: How do web browsers communicate with web servers?

Interacting with web servers

Protocol:
ftp
https
tor

Hostname/server
- translated to an IP address by DNS

Path to the resource

index.html is static content
i.e., a fixed file returned by the serve

Interacting with web servers

`http://www.cs.swarthmore.edu/~chaganti/index.html`

Protocol:
ftp
https
tor

Hostname/server
translated to an IP address by DNS

Path to the resource

index.html is static content
i.e., a fixed file returned by the server

`http://facebook.com/delete.php`

Path to the resource

delete.php is dynamic content
i.e., a server generates the content on the fly

Interacting with web servers: dynamic content

`http://facebook.com/delete.php` Path to the resource

`http://facebook.com/delete.php?f=eva264&w=16`

arguments

server generates the content on the fly

URL Escaping

http://facebook.com/delete.php?f=eva264&w=16

- URLs are designed to contain printable, human-readable characters (ASCII)
  - include non-printable characters in the URL?
- URLs have special characters that have assigned meaning (?, #, /)

URL Escaping

http://facebook.com/delete.php?f=eva264&w=16

- What if we want to use a special character *in* the URL?
  - Solution: URL encoding
  - Notation: Percent sign (%) followed by the hexadecimal value of the character
  - Example: %20 = ' ' (spacebar) %35 = '#' (hash sign)

    %50 = '2' (printable characters can be encoded too!)
- Security issues: makes scanning for malicious URLs harder
  - Suppose you want to block all requests to the path /etc/passwd
  - What if an attacker makes a request to %2F%65%74%63%2F%70%61%73%73%77%64?

HTTP and the Web

First, a review…

- web page consists of objects
- object can be HTML file, JPEG image, Java applet, audio file,…
- web page consists of base HTML-file which includes several referenced objects
- each object is addressable by a URL, e.g.,

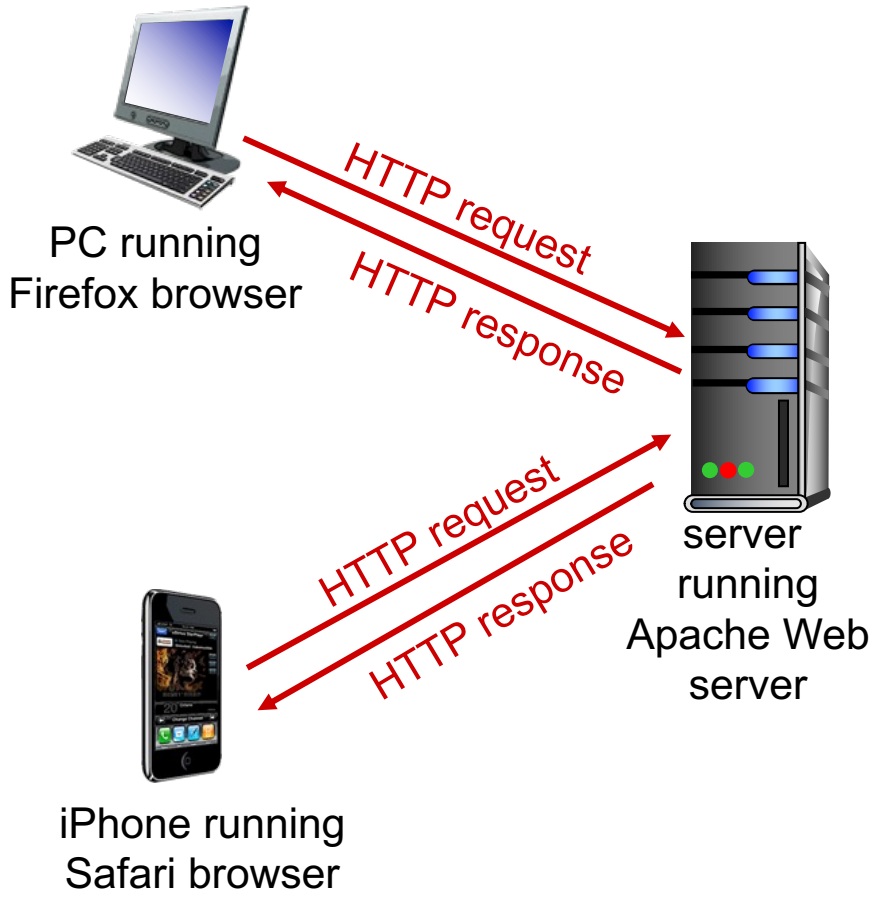
Protocol:
ftp
https
tor

Hostname/server
- translated to an IP address by DNS

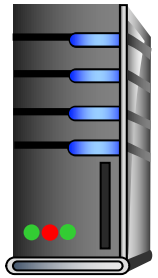
Path to the resource

HTTP: Hypertext transfer protocol

- client/server model
  - client: browser that requests, receives, (using HTTP protocol) and "displays" Web objects
  - server: Web server sends (using HTTP protocol) objects in response to requests



HTTP Overview



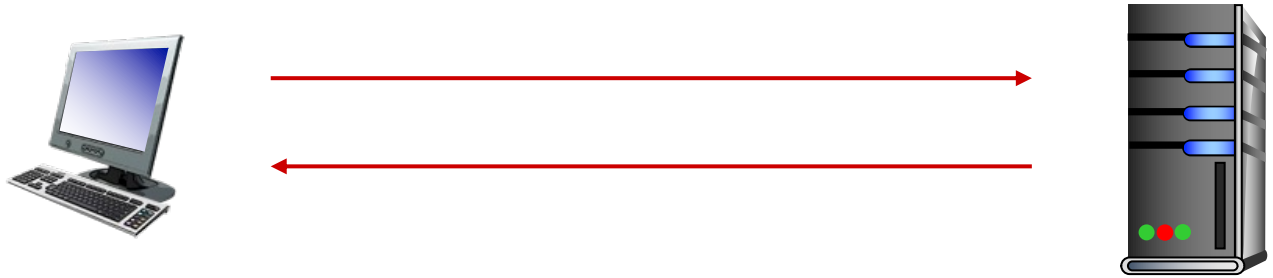
1. User types in a URL.

http://some.host.name.tld/directory/name/file.ext

host name

path name

HTTP Overview



2. Browser establishes connection with server.
   Looks up “some.host.name.tld”
   connects

HTTP Overview

3. Browser requests the corresponding data.

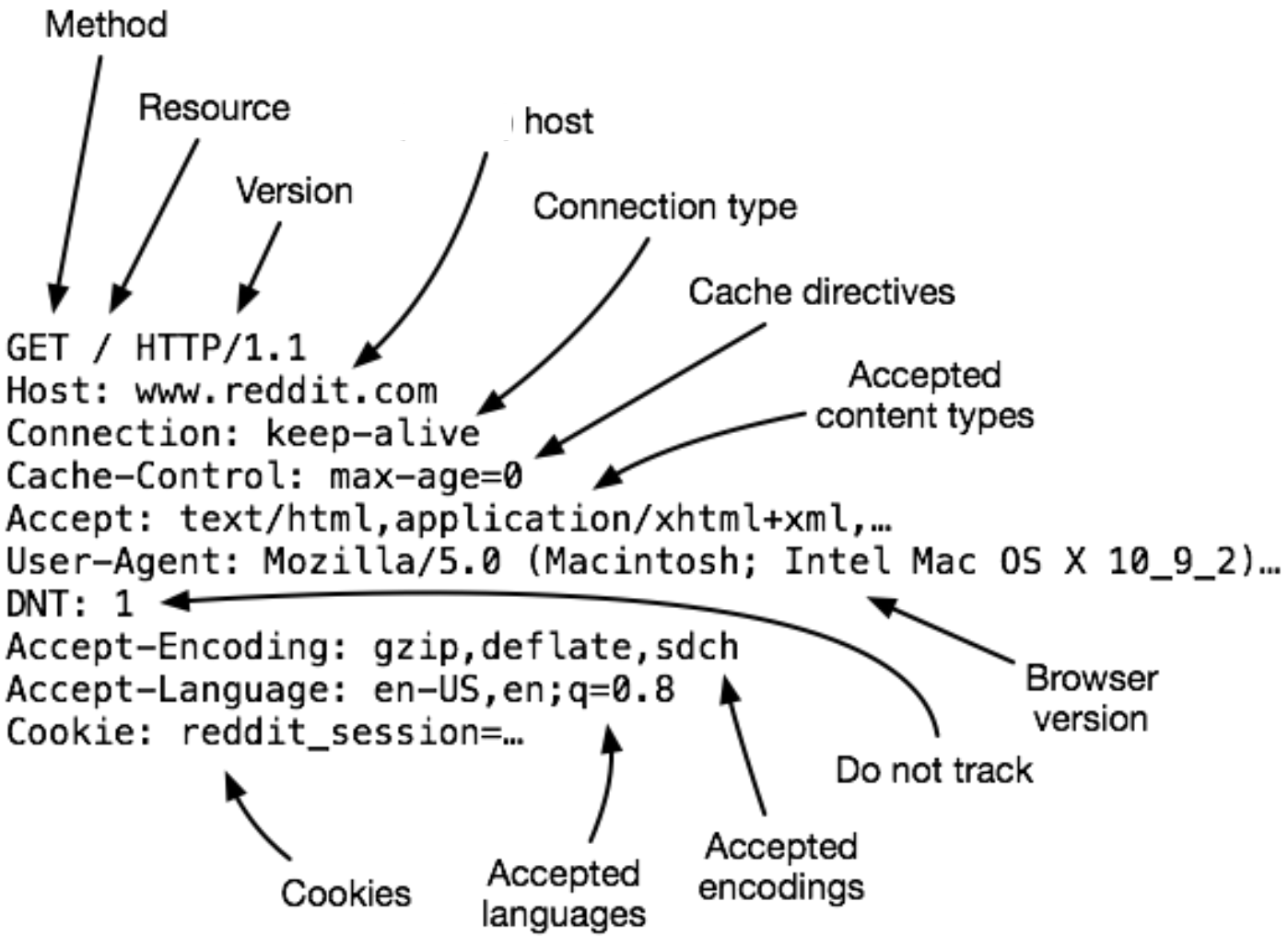
   GET /directory/name/file.ext HTTP/1.0
   Host: some.host.name.tld
   [other optional fields, for example:]
   User-agent: Mozilla/5.0 (Windows NT 6.1; WOW64)
   Accept-language: en

HTTP Overview

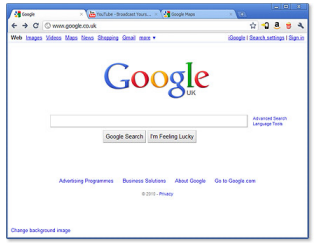
4. Server responds with the requested data.

   HTTP/1.0 200 OK
   Content-Type: text/html
   Content-Length: 1299
   Date: Sun, 01 Sep 2013 21:26:38 GMT
   [Blank line]
   (Data data data data…)

HTTP Request Header

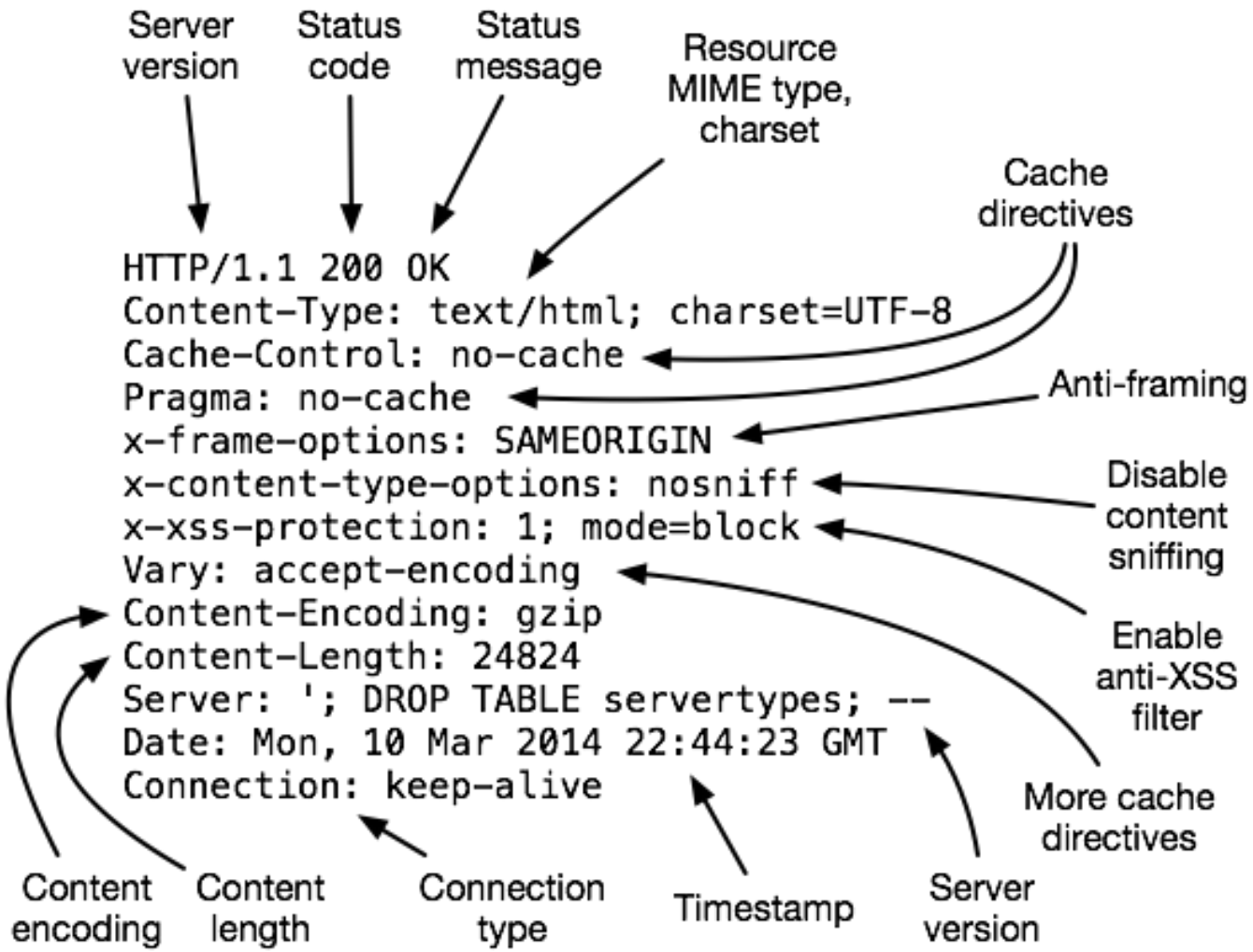


HTTP Overview



5. Browser renders the response, fetches any additional objects, and closes the connection.

HTTP Response Header



Example

```
GET / HTTP/1.1
Host: demo.cs.swarthmore.edu

HTTP/1.1 200 OK
Vary: Accept-Encoding
Content-Type: text/html
Accept-Ranges: bytes
ETag: "316912886"
Last-Modified: Wed, 04 Jan 2017 17:47:31 GMT
Content-Length: 1062
Date: Wed, 05 Sep 2018 17:27:34 GMT
Server: lighttpd/1.4.35
```

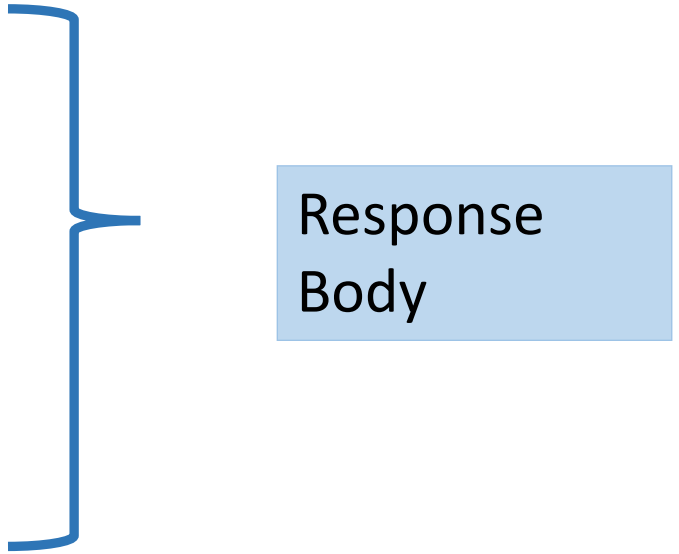
Response headers

Response Body

Example

GET / HTTP/1.1
Host: demo.cs.swarthmore.edu

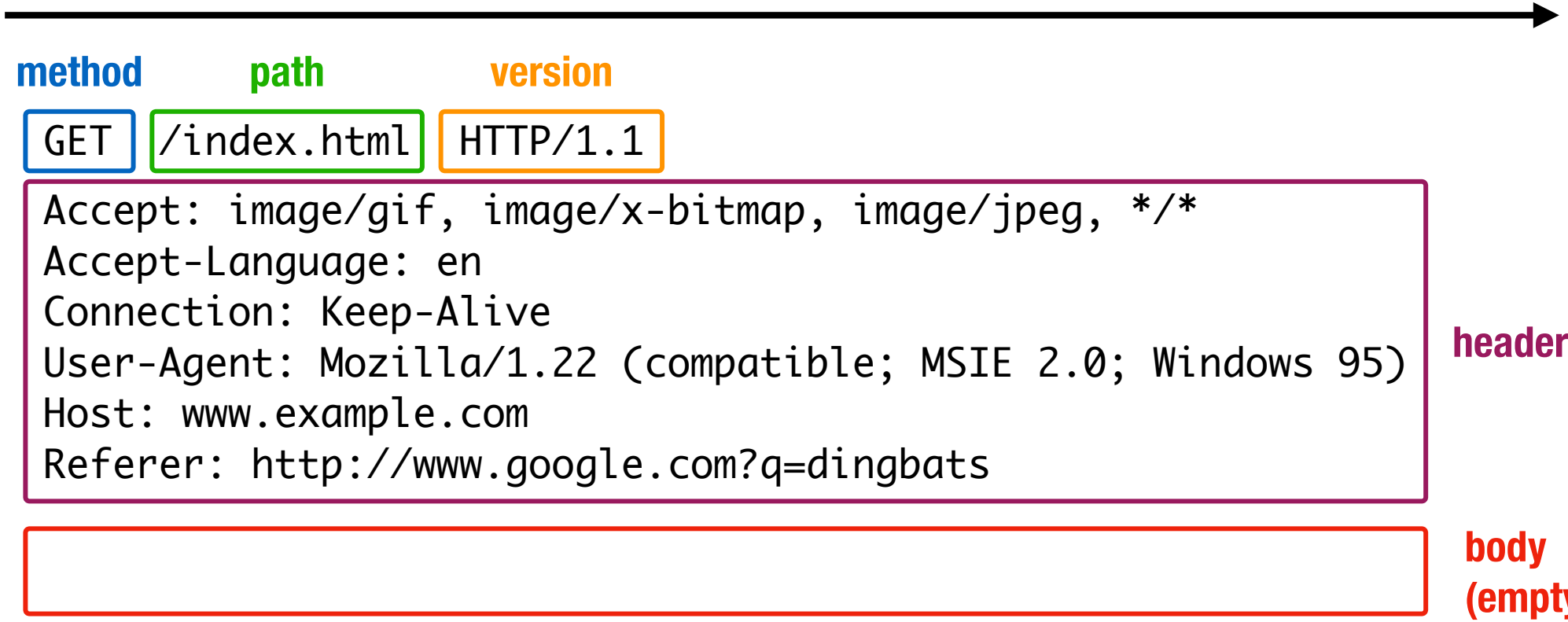
Response Headers



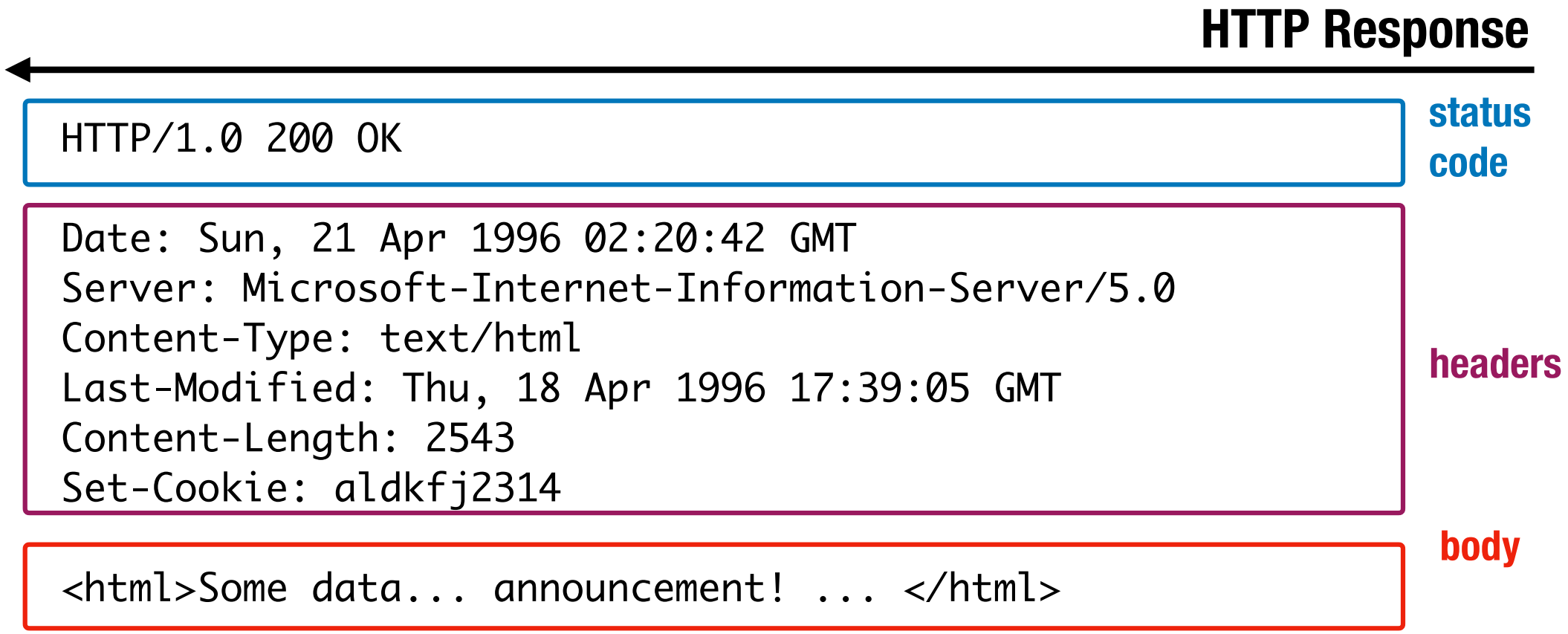
<html><head><title>Demo Server</title></head>
<body>
.....
</body>
</html>

Anatomy of Request

HTTP Request



HTTP Response



HTTP Methods

GET: Get the resource at the specified URL (does not accept message body)

POST: Create new resource at URL with payload

PUT: Replace target resource with request payload

PATCH: Update part of the resource

DELETE: Delete the specified URL

HTTP Methods

Not all methods are created equal — some have different security protections

GETs should not change server state; in practice, some servers do perform side effects

- Old browsers don't support **PUT**, **PATCH**, and **DELETE**
- Most requests with a side affect are **POST**s today
- Real method hidden in a header or request body

🤦‍♀️ **Never do…**

```
GET
http://bank.com/transfer?fromAcct=X&toAcct=Y&amount=1000
```

Goals of Web Security: Safely Browse the Web

- Safe to visit an evil website
  - sandboxing Javascript
  - privilege separation

- Safe to visit two pages at the same time,
  - same-origin policy

- Safe delegation

Web Security Model

Subjects

"Origins" — a unique **scheme://domain:port**

Objects

DOM tree, DOM storage, cookies, javascript namespace, HW permission

Same Origin Policy (SOP)

Goal: Isolate content of different origins

- **Confidentiality**: script on evil.com should not be able to *read* bank.ch
- **Integrity**: evil.com should not be able to *modify* the content of bank.ch

Same Origin Policy

- rule that prevents one website from tampering with *other unrelated websites.*
  - *enforced by browser*

Same-Origin Policy

- Every webpage has an **origin** defined by its URL with three parts:
  - **Protocol**: The protocol in the URL
  - **Domain**: The domain in the URL's location
  - **Port**: The port in the URL's location
    - If no port is specified, the default is 80 for HTTP and 443 for HTTPS

https://cs.swarthmore.edu:443/assets/lock.PNG

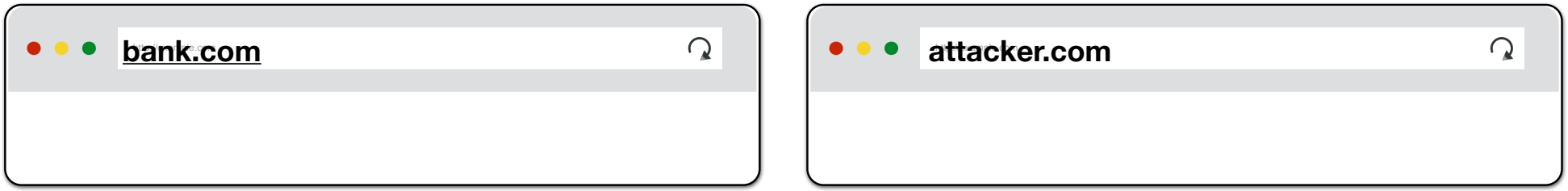
http://cs.swarthmore.edu/assets/images/404.png

80 (default port)

Bounding Origins — Windows

Every Window and Frame has an origin

Origins are blocked from accessing other origin's objects



`attacker.com` cannot…

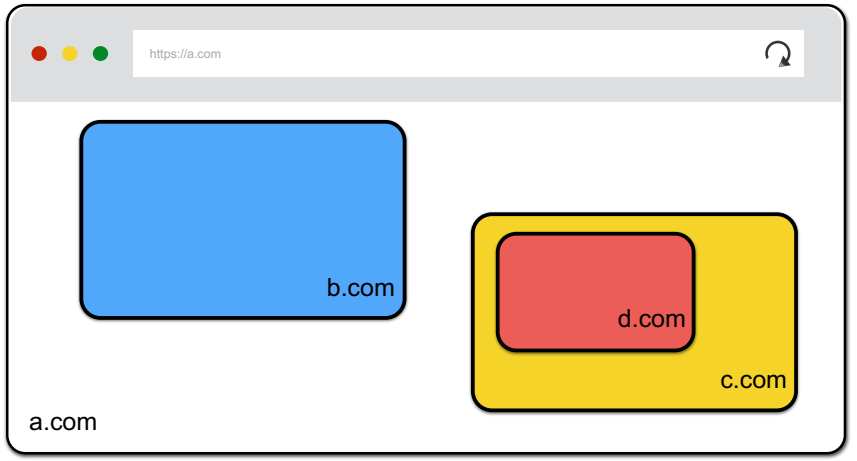
- *read or write* content from **`bank.com`** tab
- *read or write* **`bank.com`**'s *cookies*
- *detect* that the other tab has **`bank.com`** loaded

(i)Frames

Beyond loading individual resources, websites can also load other *websites* within their window

- Frame: rigid visible division
- iFrame: floating inline frame

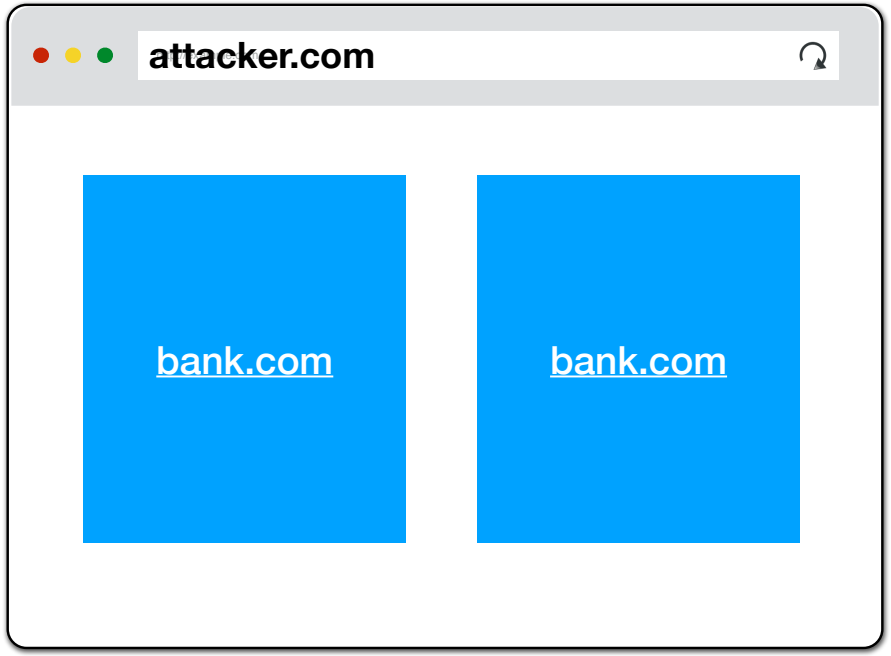
Allows delegating screen area to content from another source (e.g., ad)



Bounding Origins — Frames

Every Window and Frame has an origin

Origins are blocked from accessing other origin's objects



`attacker.com` cannot…

- *read* content from **`bank.com`** frame
- *access* **`bank.com`**'s *cookies*
- *detect* that has **`bank.com`** loaded

Same-Origin Policy

- Two webpages have the same origin *if and only if* the protocol, domain, and port of the URL all match exactly: string matching:
  - The `protocol`, `domain`, and `port` strings must be equal

First domain	Second domain	Same origin?
`http://cs88.swat.org`	`https://cs88.swat.org`	
`http://cs88.swat.org`	`http://swat.org`	
`http://cs88.swat.org` `[:80]`	`http://cs88.swat.org:8000`	

Same-Origin Policy: Two websites with different origins can't interact with each other.

Example: If **cs88.org** embeds **google.com**, the inner frame cannot interact with the outer frame, and the outer frame cannot interact with the inner-frame

So what happens when…

1. JavaScript runs with the origin of the page that loads it? E.g., `cs88.org` fetches Javascript from Google analytics.
2. Websites fetch and display images from other origins? E.g. if we include `<img src="http://google.com/logo.jpg>` on `http://cs88.org`, the image has origin `http://google.com`.
3. We load frames such as `<iframe src="http://google.com"></iframe>` on `cs88.org`?

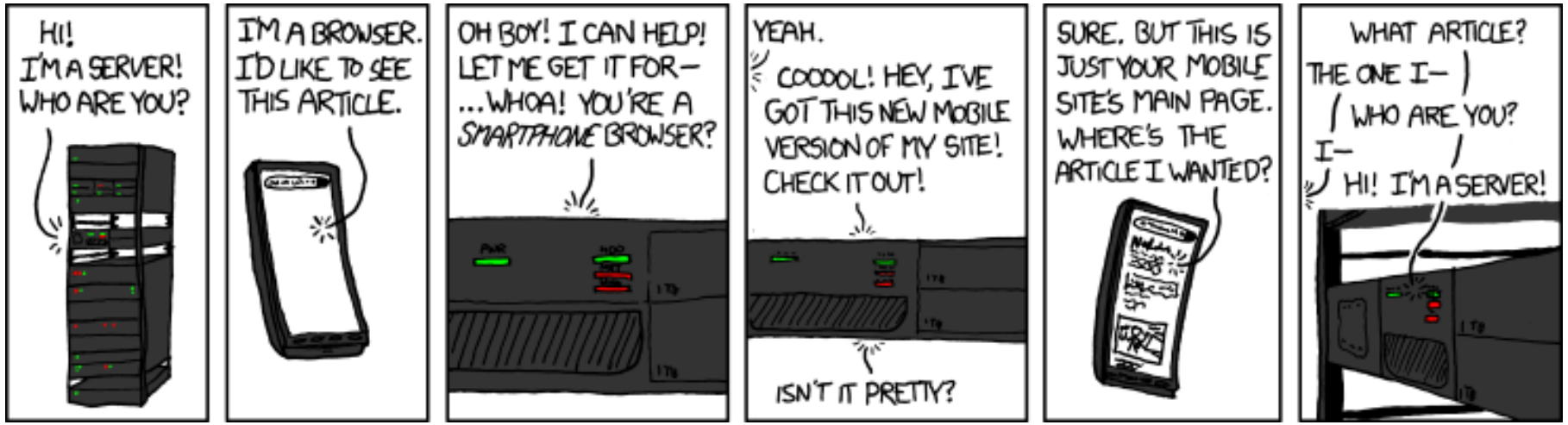
Same-Origin Policy

- Two websites with different origins cannot interact with each other
  - Example: If **cs88.org** embeds **google.com**, the inner frame cannot interact with the outer frame, and the outer frame cannot interact with the inner-frame
- Exception: JavaScript runs with the origin of the page that loads it
  - Example: If **cs88.org** fetches JavaScript from **google.com**, the JavaScript has the origin of **cs88.org**
  - *Intuition: **cs88.org** has "copy-pasted" JavaScript onto its webpage*
- Exception: Websites can fetch and display images from other origins
  - However, the website only knows about the image's size and dimensions (*cannot actually manipulate the image*)
- Exception: Websites can agree to allow some limited sharing
  - Cross-origin resource sharing (CORS)
  - The **postMessage** function in JavaScript

Same-Origin Policy: Summary

- Rule enforced by the browser: Two websites with different origins cannot interact with each other
- Two webpages have the same origin *if and only if* the protocol, domain, and port of the URL all match exactly (string matching)
- Exceptions
  - JavaScript runs with the origin of the page that loads it
  - Websites can fetch and display images from other origins
  - Websites can agree to allow some limited sharing

State(less)

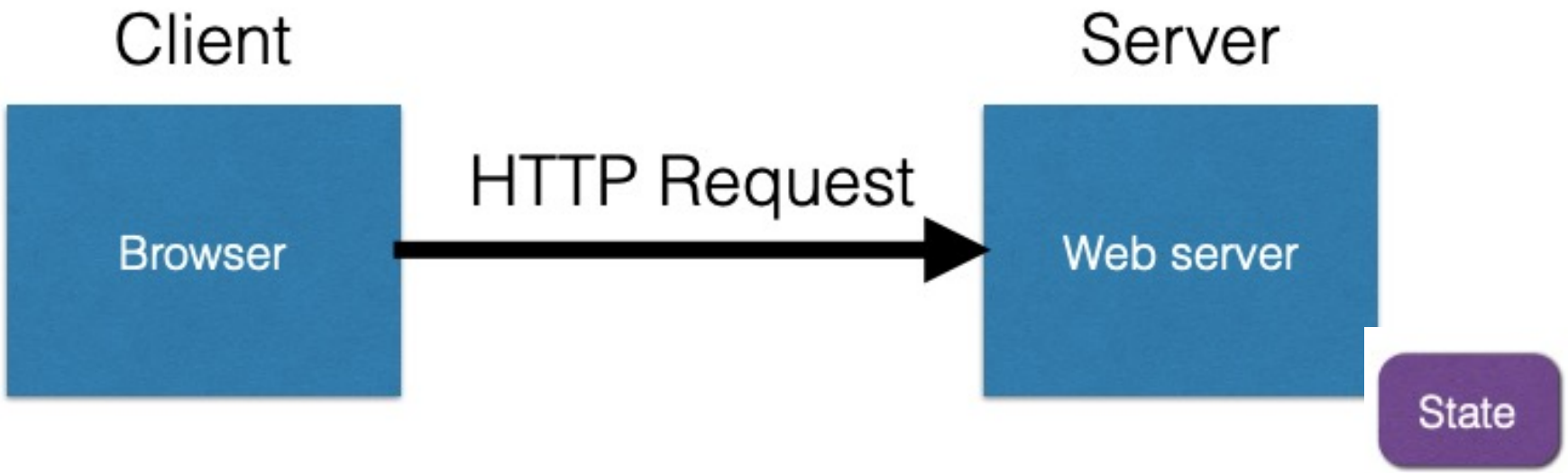


(XKCD #869, “Server Attention Span”)

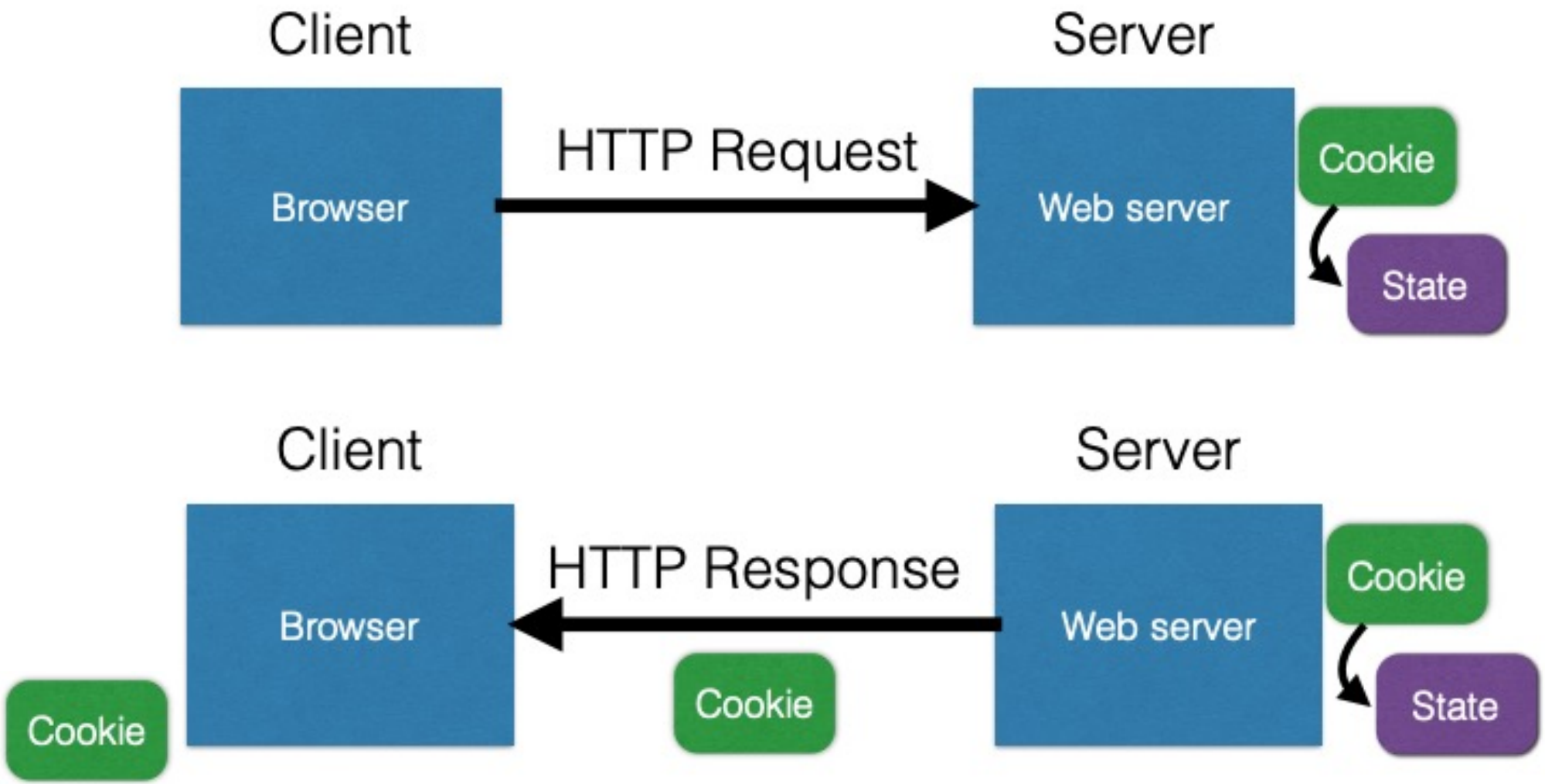
State(less)

- Original web: simple document retrieval
- Maintain State? Server is not required to keep state between connections

  ...often it might want to though
- Authentication: Client is not required to identify itself
  - server might refuse to talk otherwise though



- Server stores state, indexes it with a cookie
- Send this cookie to the client
- Client stores the cookie and returns it with subsequent queries to that same server


Client
Server
Browser
HTTP Request
Web server
Cookie
State
Client
Server
Browser
HTTP Response
Web server
Cookie
State
Cookie
Cookie


Client
Server
Server
Browser
HTTP Request
Web server
Cookie
Cookie
State


Client
Server
Server
Browser
HTTP Request
Web server
Cookie
Cookie
Cookie
State

Browser Cookie Management

- Cookie Same-origin ownership
  - Once a cookie is saved on your computer, only the Web site that created the cookie can read it.
- Variations
  - Temporary cookies
    - Stored until you quit your browser
  - Persistent cookies
    - Remain until deleted or expire
  - Third-party cookies
    - Originates on or sent to a web site other than the one that provided the current page

Third-party cookies

- Get a page from merchant.com
  - Contains <img src=http://doubleclick.com/advt.gif>
  - Image fetched from DoubleClick.com
    - DoubleClick knows IP address and page you were looking at
- DoubleClick sends back a suitable advertisement
  - Stores a cookie that identifies "you" at DoubleClick
- Next time you get page with a doubleclick.com image
  - Your DoubleClick cookie is sent back to DoubleClick
  - DoubleClick could maintain the set of sites you viewed
  - Send back targeted advertising (and a new cookie)
- Cooperating sites
  - Can pass information to DoubleClick in URL, …

Cookie issues

- Cookies maintain record of your browsing habits
  - Cookie stores information as set of name/value pairs
  - May include *any* information a web site knows about you
  - Sites track your activity from multiple visits to site
- Sites can share this information (e.g., DoubleClick)
- Browser attacks could invade your "privacy"

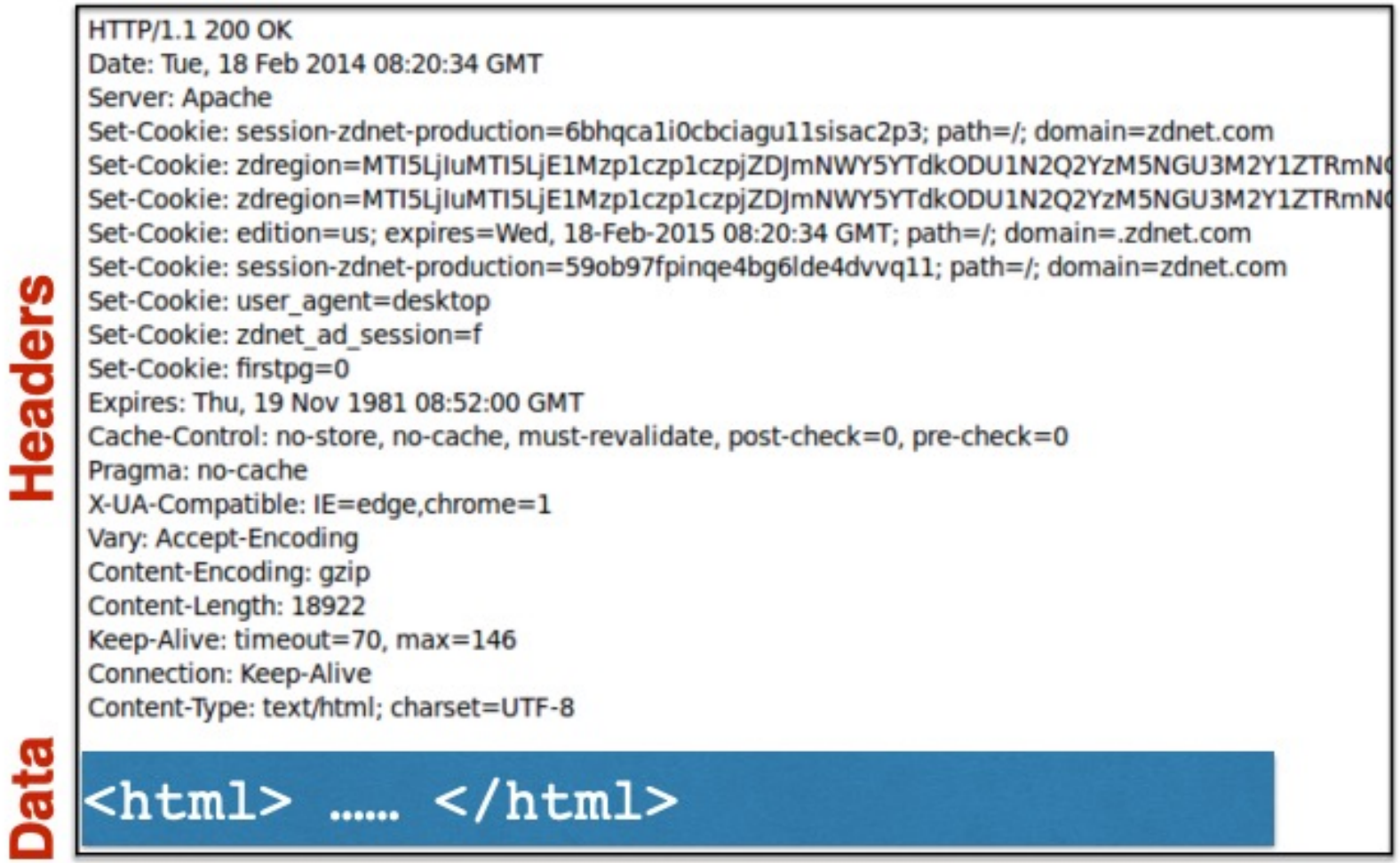
Browser Fingerprinting

- Browser sends HTTP head information, which includes
  - User agent: e.g., “Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/38.0.2125.111 Safari/537.36”
  - HTTP header: e.g., “text/html, */* gzip,deflate en-US,en;q=0.8”
  - Javascript can collect font information, installed browser-plugin information
  - Using canvas, e.g., how to render emoji
  - Can achieve high entropy.
  - Can be used to track users/browsers.

- https://panopticlick.eff.org/

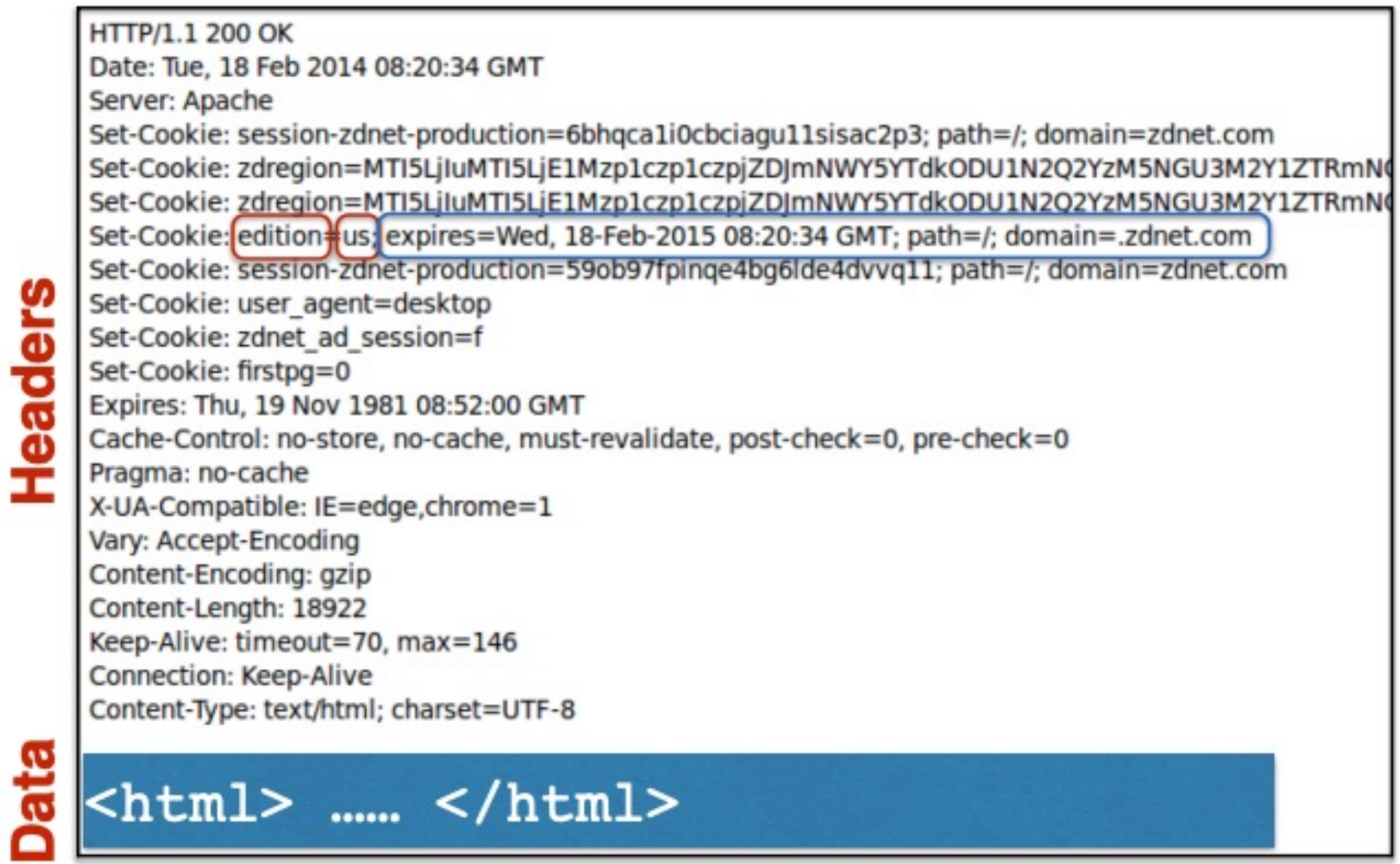
Cookies are key-value pairs

Set-Cookie:key=value; options; ….



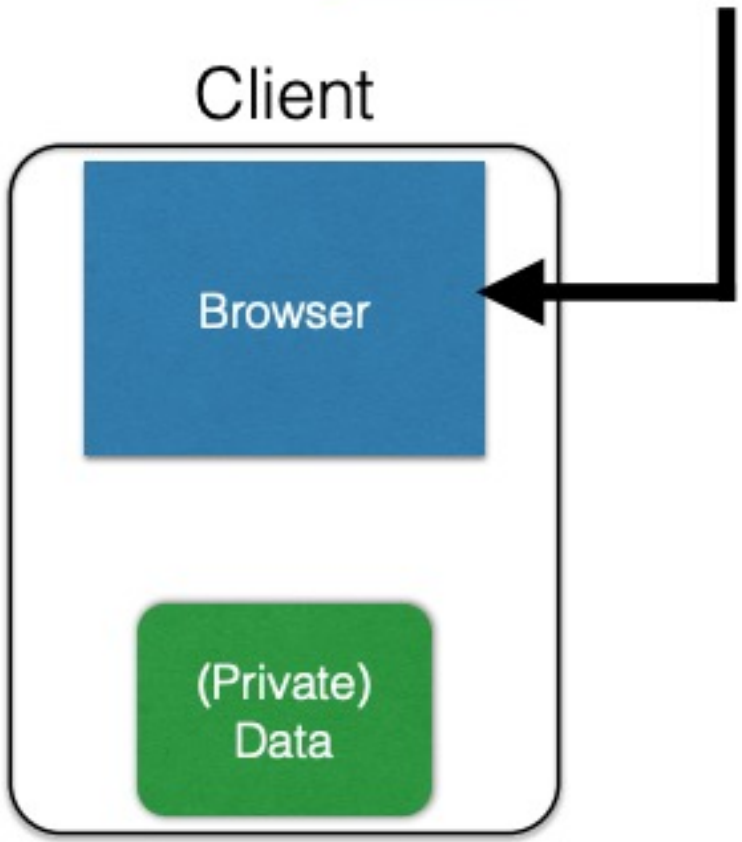
Cookies are key-value pairs

Set-Cookie:key=value; options; ….



Cookies

Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com

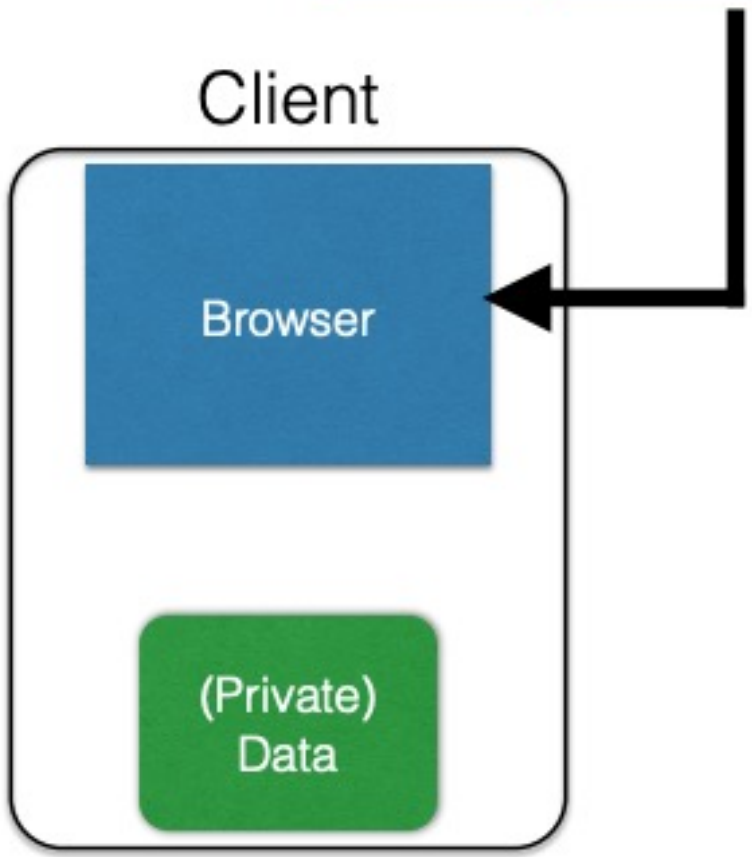


Semantics

- Store “us” under the key “edition” (think of it like one big hash table)

Cookies

Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com

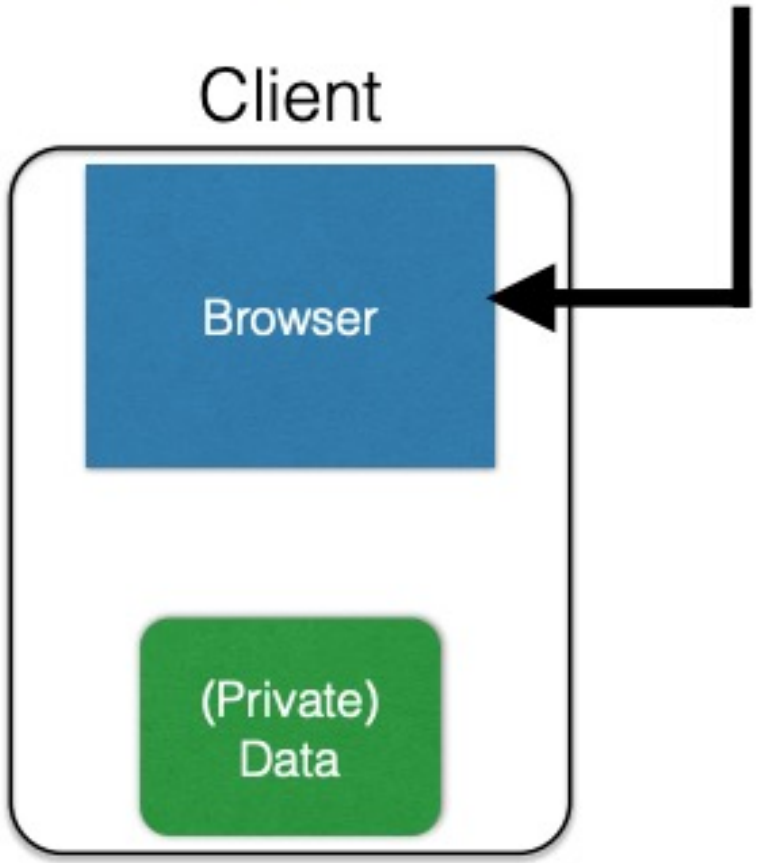


Semantics

- Store “us” under the key “edition” (think of it like one big hash table)
- This value is no good as of Wed Feb 18…

Cookies

Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com

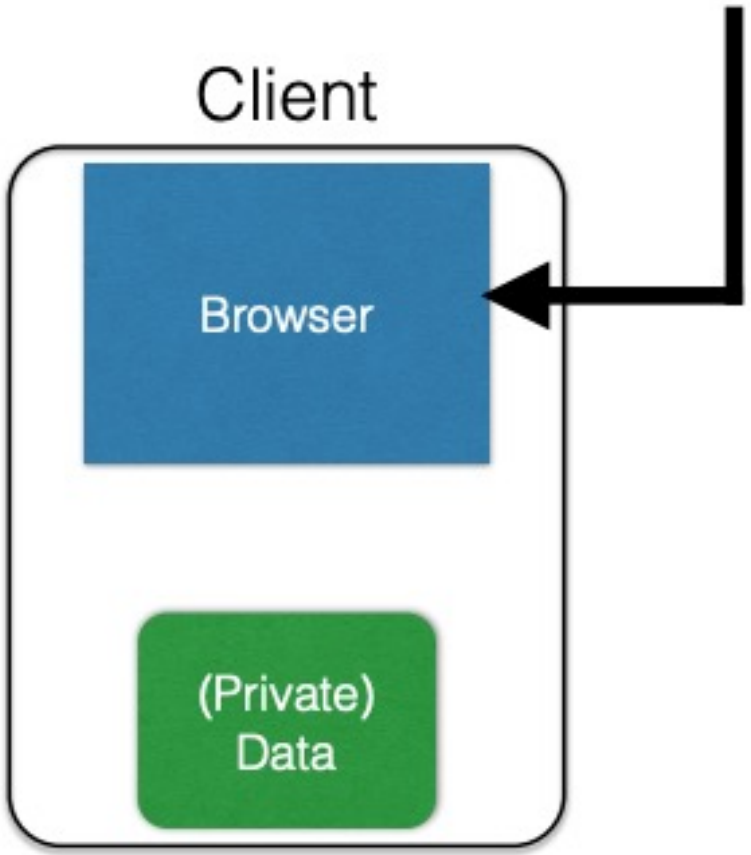


Semantics

- Store “us” under the key “edition” (think of it like one big hash table)
- This value is no good as of Wed Feb 18…
- This value should only be readable by any domain ending in `.zdnet.com`

Cookies

Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com

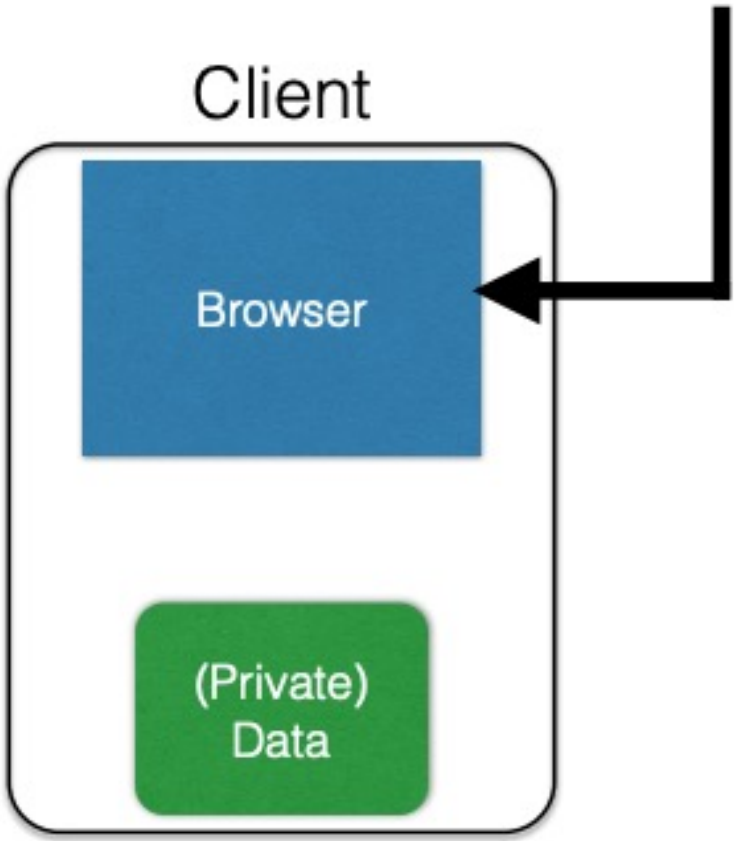


Semantics

- Store “us” under the key “edition” (think of it like one big hash table)
- This value is no good as of Wed Feb 18…
- This value should only be readable by any domain ending in `.zdnet.com`
- This should be available to any resource within a subdirectory of `/`

Cookies

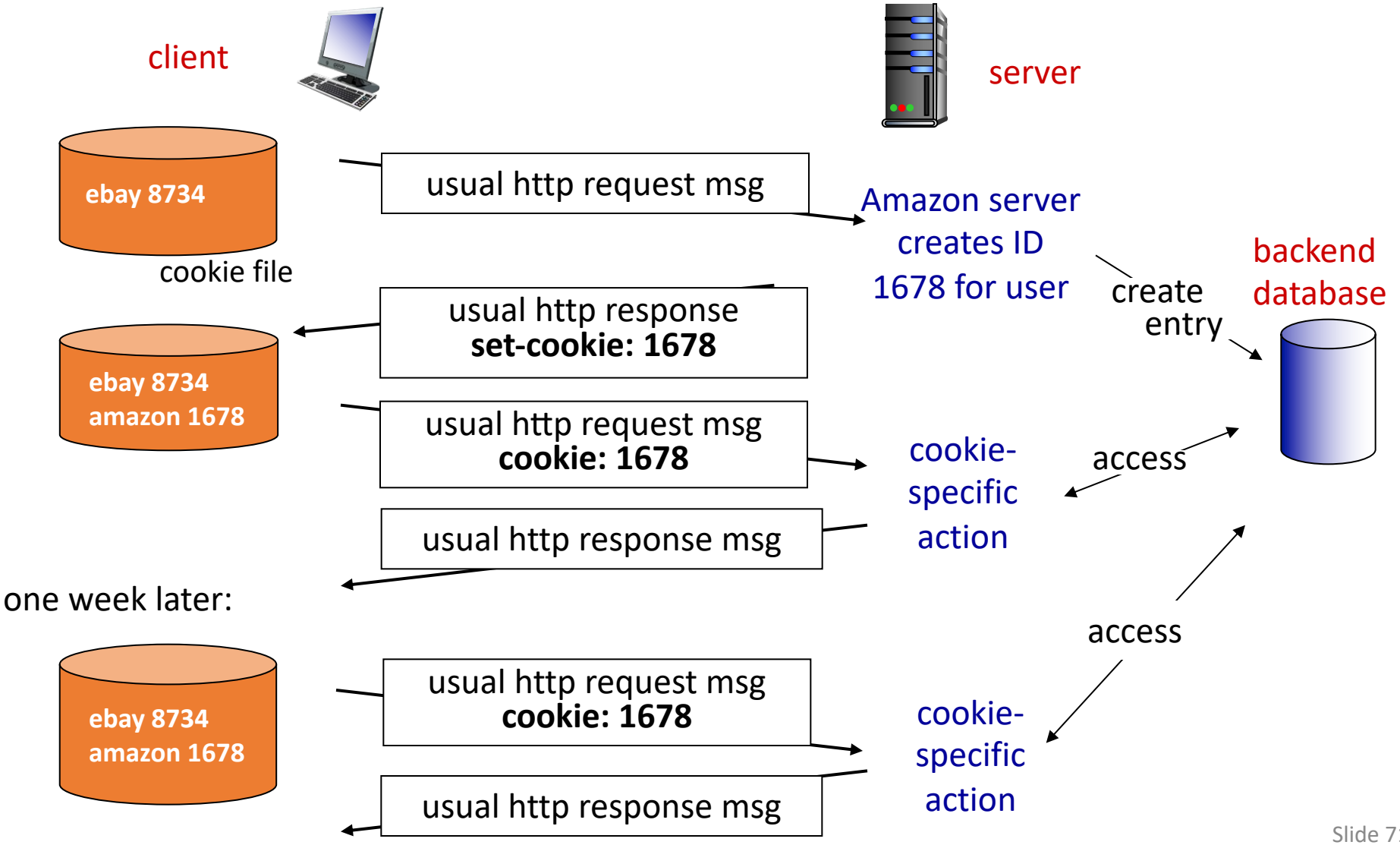
Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com



Semantics

- Store “us” under the key “edition” (think of it like one big hash table)
- This value is no good as of Wed Feb 18…
- This value should only be readable by any domain ending in `.zdnet.com`
- This should be available to any resource within a subdirectory of `/`
- Send the cookie to any future requests to `<domain>/<path>`

Cookies: keeping "state" (cont.)



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;
  - Read about iPads on CNN and see ads on Amazon 😱
  - How can an advertiser (A) know what you did on another site (S)?

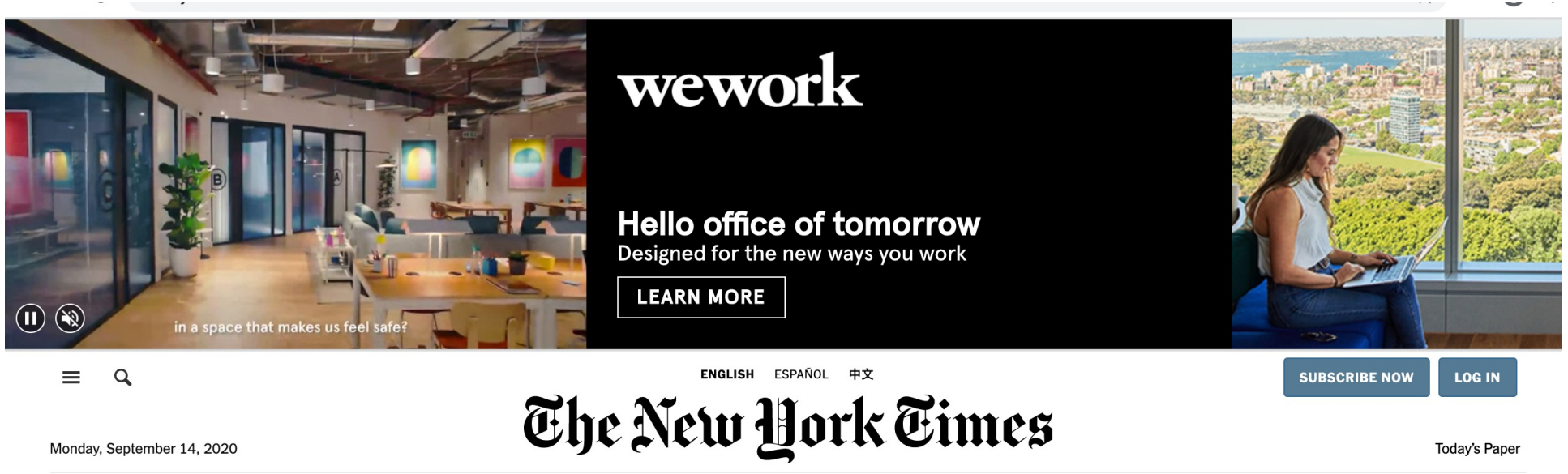
HTTP Request/Responses with Cookies

Cookies and Privacy

Cookies permit sites to learn a lot about you

supply name and e-mail to sites (and more!)

third-party cookies (ad networks) follow you across multiple sites.



Why use cookies?

- Tracking users
  - Advertisers want to know your behavior
  - Ideally build a profile *across different websites*
    - Read about iPad on CNN, then see ads on Amazon?!
  - How can an advertiser (A) know what you did on another site (S)?

Why use cookies?

- Tracking users
  - Advertisers want to know your behavior
  - Ideally build a profile *across different websites*
    - Read about iPad on CNN, then see ads on Amazon?!
  - How can an advertiser (A) know what you did on another site (S)?

S shows you an ad from A; A scrapes the referrer URL

Why use cookies?

- Tracking users
  - Advertisers want to know your behavior
  - Ideally build a profile *across different websites*
    - Read about iPad on CNN, then see ads on Amazon?!
  - How can an advertiser (A) know what you did on another site (S)?

S shows you an ad from A; A scrapes the referrer URL

Option 1: A maintains a DB, indexed by your IP address

Problem: IP addrs change

Why use cookies?

- Tracking users
  - Advertisers want to know your behavior
  - Ideally build a profile *across different websites*
    - Read about iPad on CNN, then see ads on Amazon?!
  - How can an advertiser (A) know what you did on another site (S)?

S shows you an ad from A; A scrapes the referrer URL

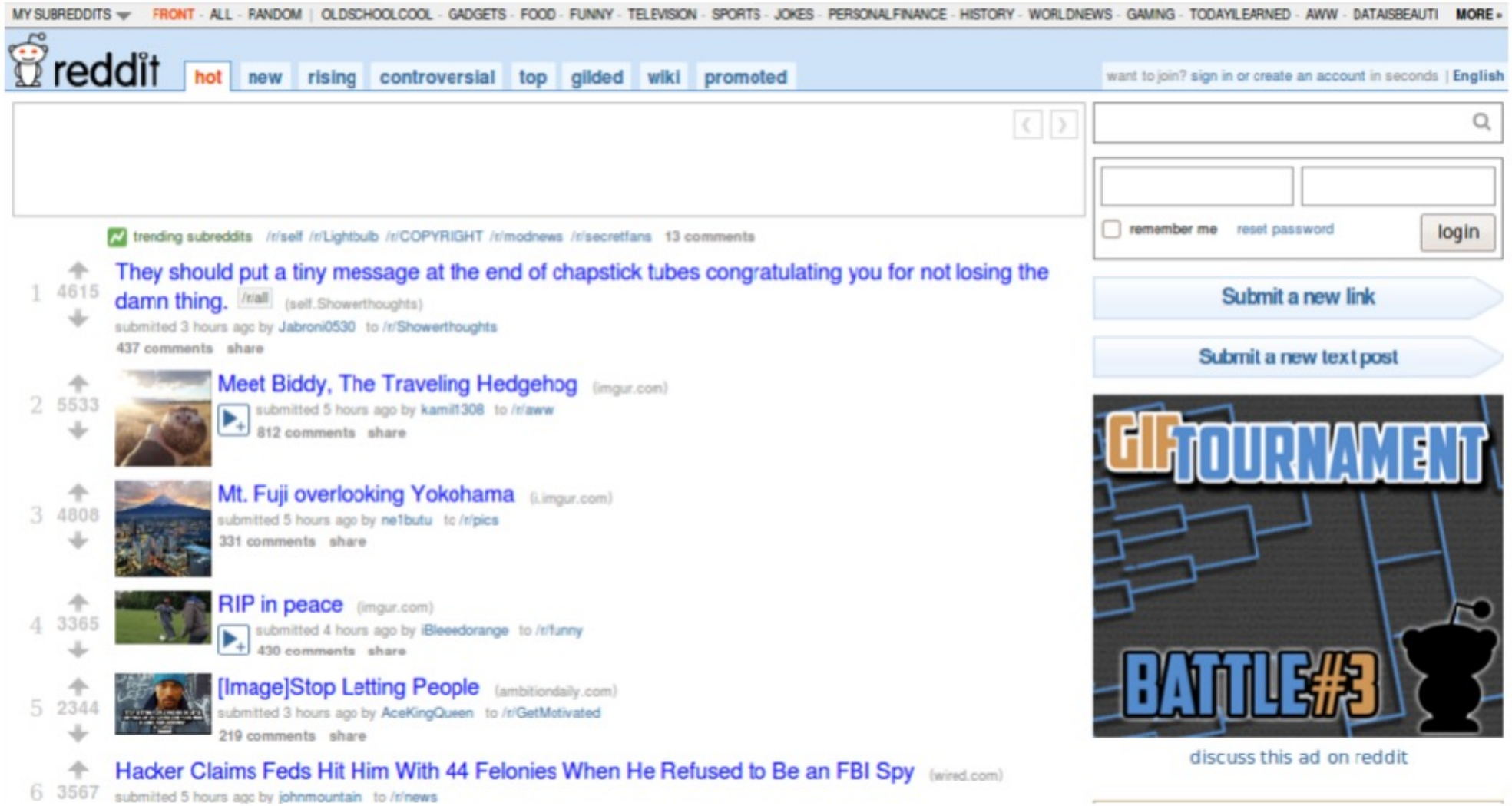
Option 1: A maintains a DB, indexed by your IP address

Problem: IP addrs change

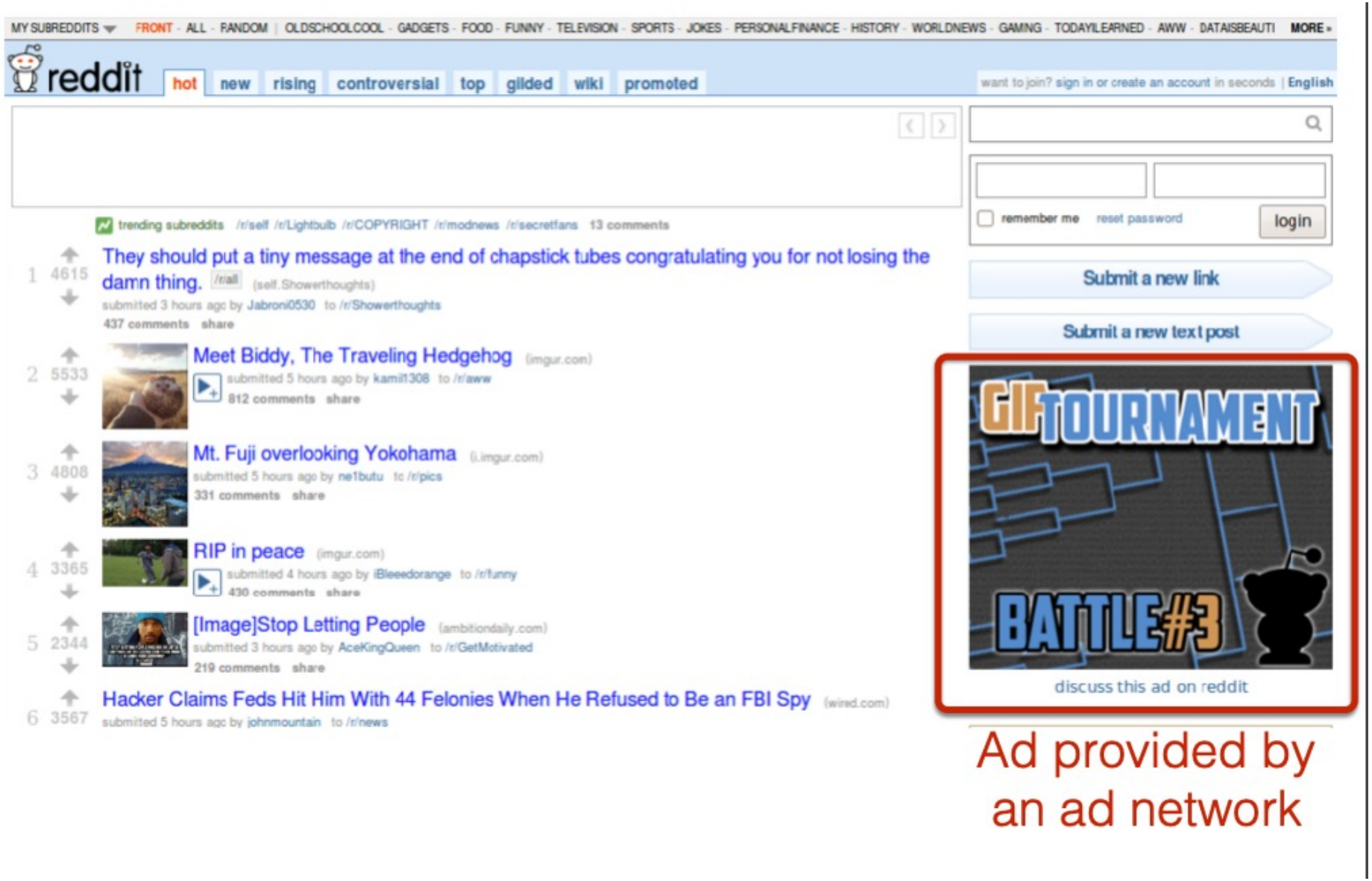
Option 2: A maintains a DB indexed by a *cookie*

- **"Third-party cookie"**
- **Commonly used by large ad networks (doubleclick)**

Cookie tracking



Cookie tracking



Cookie tracking

Snippet of reddit.com source

```
<div class="side">
    <div class="spacer">
    <div class="spacer">
    <div class="spacer">
    <div class="spacer">
    <div class="spacer">
    <div class="spacer">
        <iframe id="ad_main" scrolling="no" frameborder="0" src="http://static.adzerk.net
        /reddit/ads.html?sr=-reddit.com,loggedout&bust2#http://www.reddit.com" name="ad_main">
            <html>
                <head>
                    <style>
                    <script type="text/javascript" async="" src="http://engine.adzerk.net
                    /ados?t=1424367472275&request={"Placements":
                    [{"A":5146,"S":24950,"D":"main","AT":5},
                    {"A":5146,"S":24950,"D":"sponsorship","AT":8}],"Keywords":"-reddit.com%2Clog
                    %3A%2F%2Fwww.reddit.com%2F","IsAsync":true,"WriteResults":true}">
                    <script src="//ajax.googleapis.com/ajax/libs/jquery/1.7.1
                    /jquery.min.js" type="text/javascript">
                    <script src="//secure.adzerk.net/ados.js?q=43" type="text/javascript">
                    <script type="text/javascript">
                    <script type="text/javascript">
                    <script type="text/javascript" src="http://static.adzerk.net/Extensions
                    /adFeedback.js">
                    <link rel="stylesheet" href="http://static.adzerk.net/Extensions
                    /adFeedback.css">
                </head>
```

Cookie tracking

Snippet of reddit.com source

Our first time accessing adzerk.net

```
<div class="side">
  <div class="spacer">
  <div class="spacer">
  <div class="spacer">
  <div class="spacer">
  <div class="spacer">
  <div class="spacer">
    <iframe id="ad_main" scrolling="no" frameborder="0" src="http://static.adzerk.net
    /reddit/ads.html?sr=-reddit.com,loggedout&bust2#http://www.reddit.com" name="ad_main">
      <html>
        <head>
          <style>
          <script type="text/javascript" async="" src="http://engine.adzerk.net
          /ados?t=1424367472275&request={"Placements":
          [{"A":5146,"S":24950,"D":"main","AT":5},
          {"A":5146,"S":24950,"D":"sponsorship","AT":8}],"Keywords":"-reddit.com%2Clog
          %3A%2F%2Fwww.reddit.com%2F","IsAsync":true,"WriteResults":true}">
          <script src="//ajax.googleapis.com/ajax/libs/jquery/1.7.1
          /jquery.min.js" type="text/javascript">
          <script src="//secure.adzerk.net/ados.js?q=43" type="text/javascript">
          <script type="text/javascript">
          <script type="text/javascript">
          <script type="text/javascript" src="http://static.adzerk.net/Extensions
          /adFeedback.js">
          <link rel="stylesheet" href="http://static.adzerk.net/Extensions
          /adFeedback.css">
        </head>
```

Cookie tracking

I visit reddit.com

HTTP Headers

http://static.adzerk.net/reddit/ads.html?sr=-reddit.com,loggedout&bust2#http://www.reddit.com

GET /reddit/ads.html?sr=-reddit.com,loggedout&bust2 HTTP/1.1
Host: static.adzerk.net
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20101013 Ubuntu/9.04 (jaunty) Firefox/3.6.11
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Referer: http://www.reddit.com/

HTTP/1.1 200 OK
Date: Thu, 19 Feb 2015 17:37:51 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
Set-Cookie: __cfduid=dc3a93cd30ca47b76600d63cde283e9b81424367471; expires=Fri, 19-Feb-16 17:37:51 GMT; path=/; domain=.adzerk.net...

Later, I go to reddit.com/r/security

HTTP Headers

http://static.adzerk.net/reddit/ads.html?sr=security,loggedout&bust2#http://www.reddit.com

GET /reddit/ads.html?sr=security,loggedout&bust2 HTTP/1.1
Host: static.adzerk.net
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20101013 Ubuntu/9.04 (jaunty) Firefox/3.6.11
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Referer: http://www.reddit.com/r/security
Cookie: __cfduid=dc3a93cd30ca47b76600d63cde283e9b81424367471

Cookie tracking

I visit reddit.com

HTTP Headers

http://static.adzerk.net/reddit/ads.html?sr=-reddit.com,loggedout&bust2#http://www.reddit.com

GET /reddit/ads.html?sr=-reddit.com,loggedout&bust2 HTTP/1.1
Host: static.adzerk.net
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20101013 Ubuntu/9.04 (jaunty) Firefox/3.6.11
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Referer: http://www.reddit.com/

HTTP/1.1 200 OK
Date: Thu, 19 Feb 2015 17:37:51 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
Set-Cookie: __cfduid=dc3a93cd30ca47b76600d63cde283e9b81424367471; expires=Fri, 19-Feb-16 17:37:51 GMT; path=/; domain=.adzerk.net...

We are only sharing this cookie with *.adzerk.net; but we are telling them about where we just came from

Later, I go to reddit.com/r/security

HTTP Headers

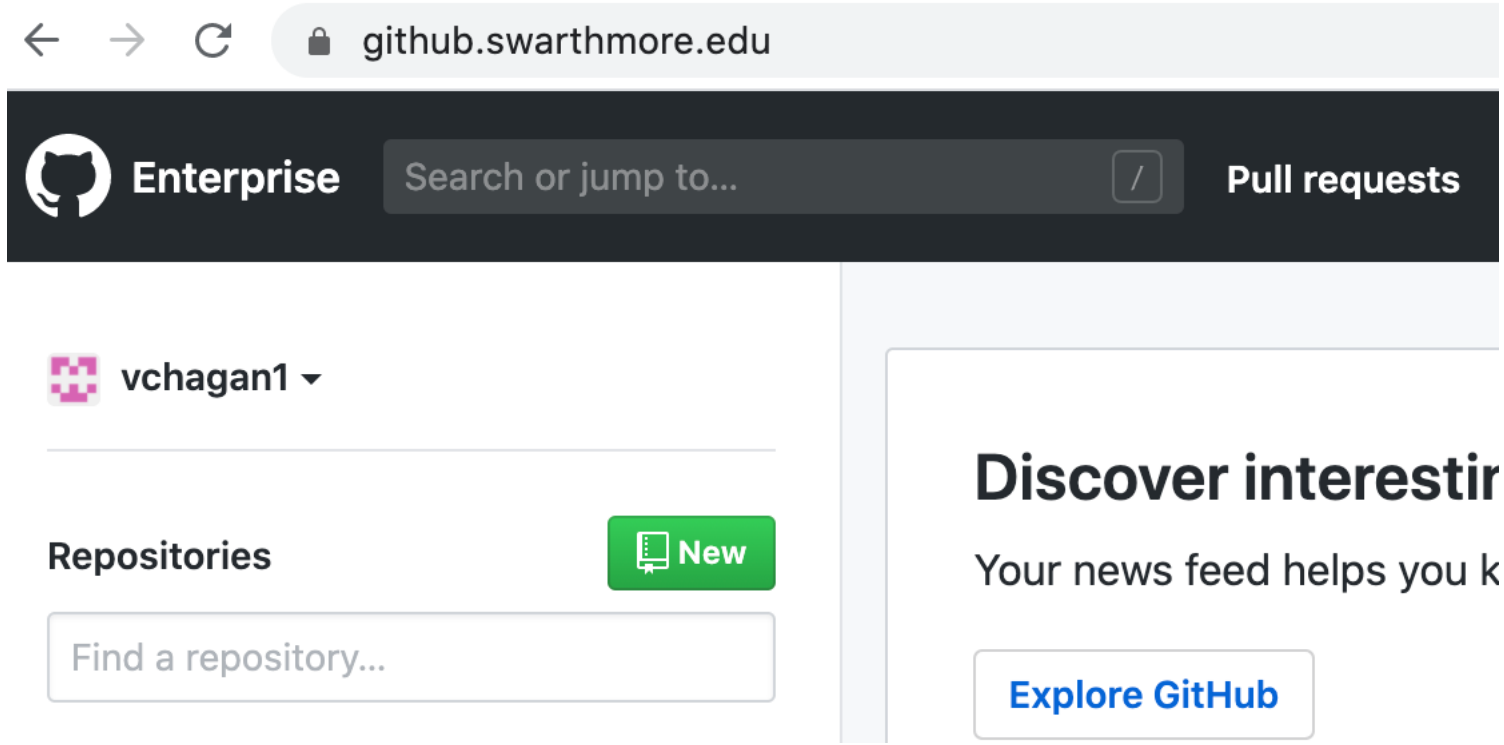
http://static.adzerk.net/reddit/ads.html?sr=security,loggedout&bust2#http://www.reddit.com

GET /reddit/ads.html?sr=security,loggedout&bust2 HTTP/1.1
Host: static.adzerk.net
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20101013 Ubuntu/9.04 (jaunty) Firefox/3.6.11
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Referer: http://www.reddit.com/r/security
Cookie: __cfduid=dc3a93cd30ca47b76600d63cde283e9b81424367471

Cookies and Privacy

Cookies permit sites to learn a lot about you

You could turn them off ...but good luck doing anything on the internet!



Cookie Policy

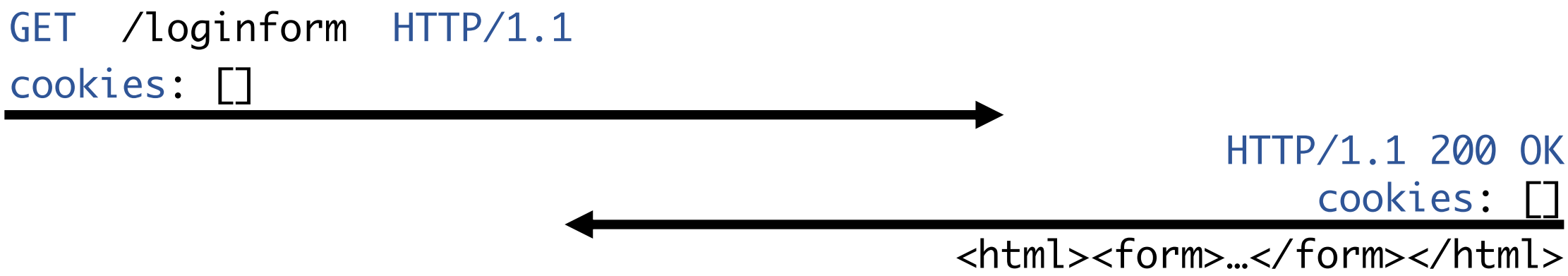
Cookie Policy

- **Cookie policy**: A set of rules enforced by the browser
  - When the browser receives a cookie from a server, should the cookie be accepted?
  - When the browser makes a request to a server, should the cookie be attached?
- Cookie policy is **not** the same as same-origin policy

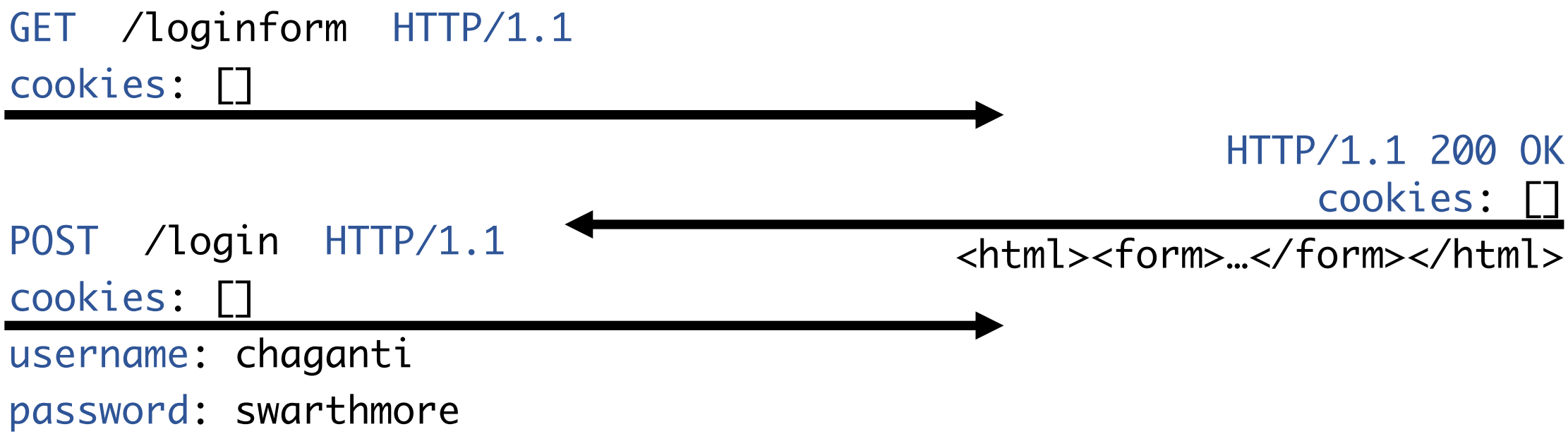
Login Session

```
GET  /loginform  HTTP/1.1
cookies: []
```

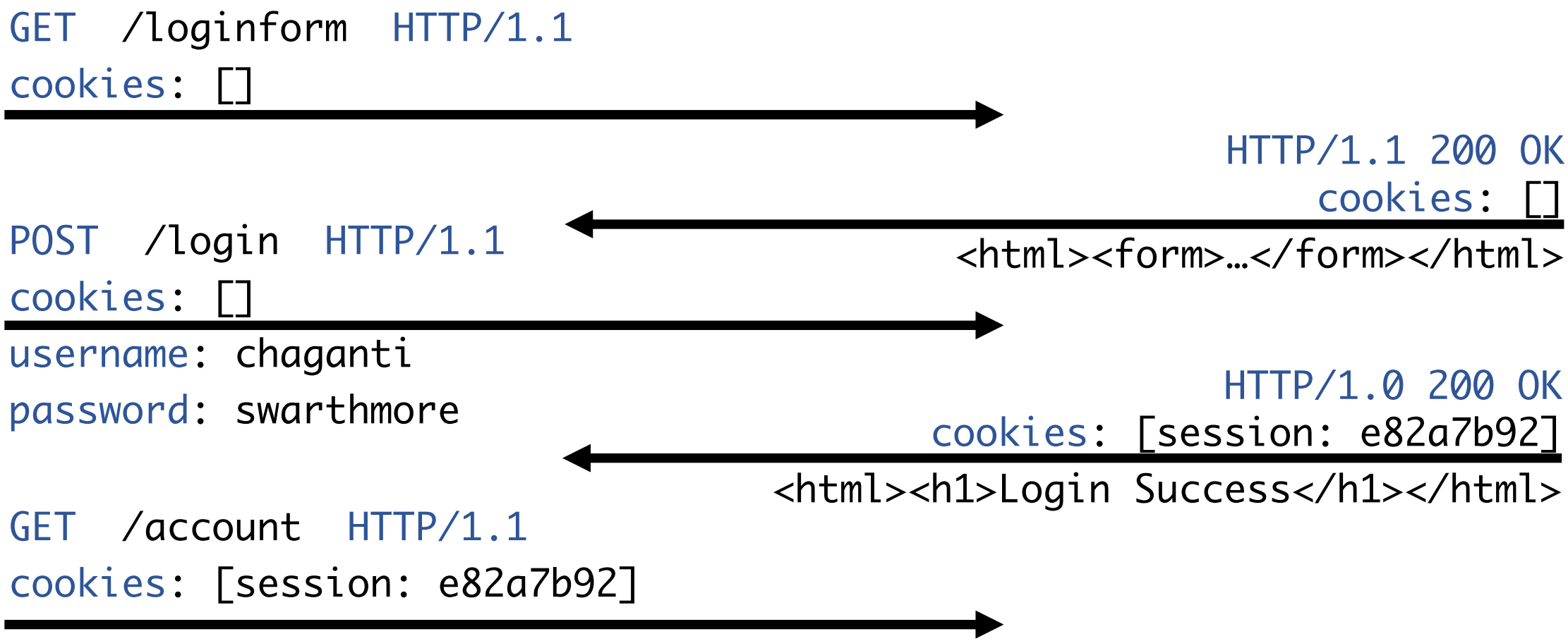
Login Session



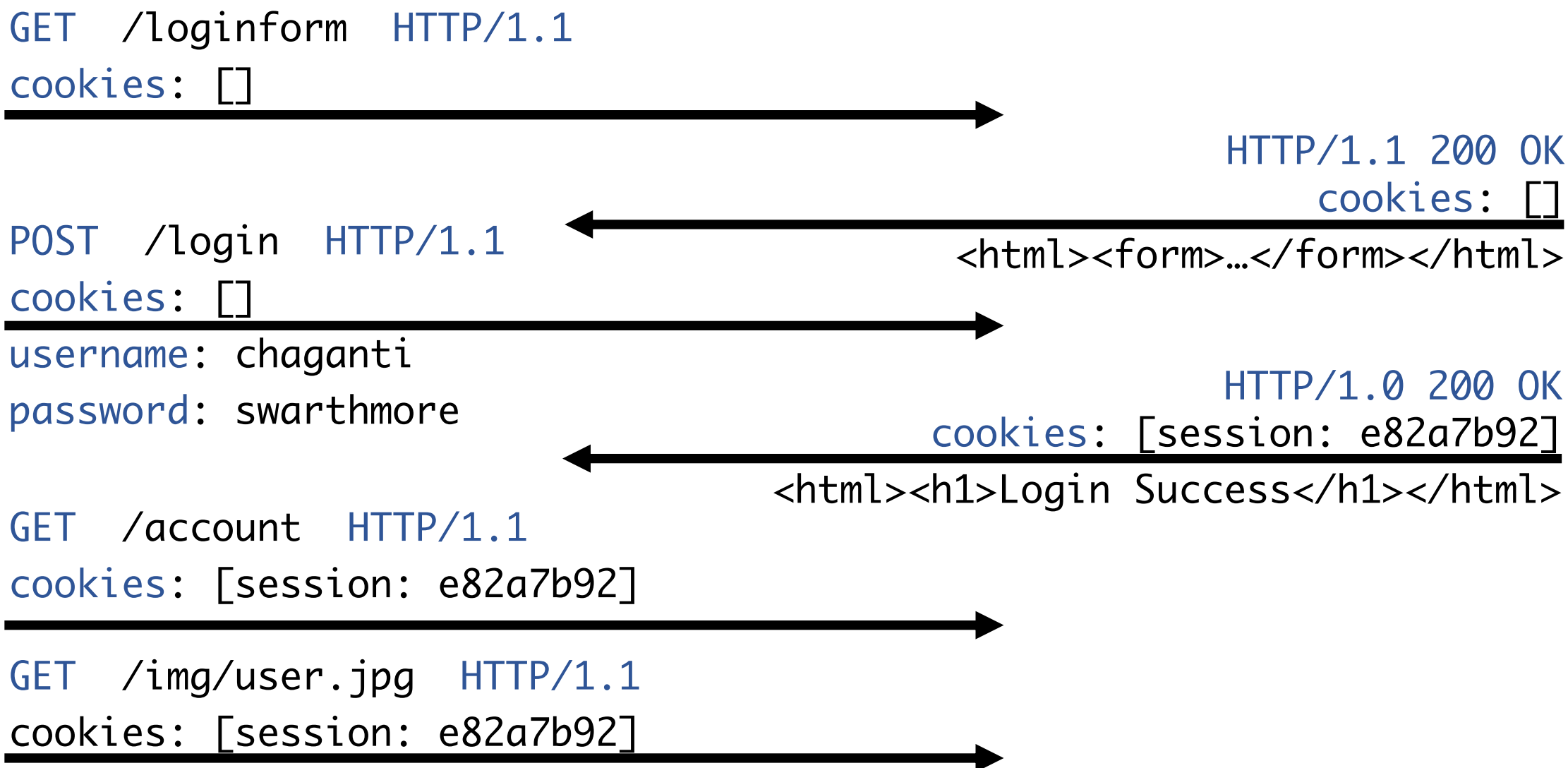
Login Session



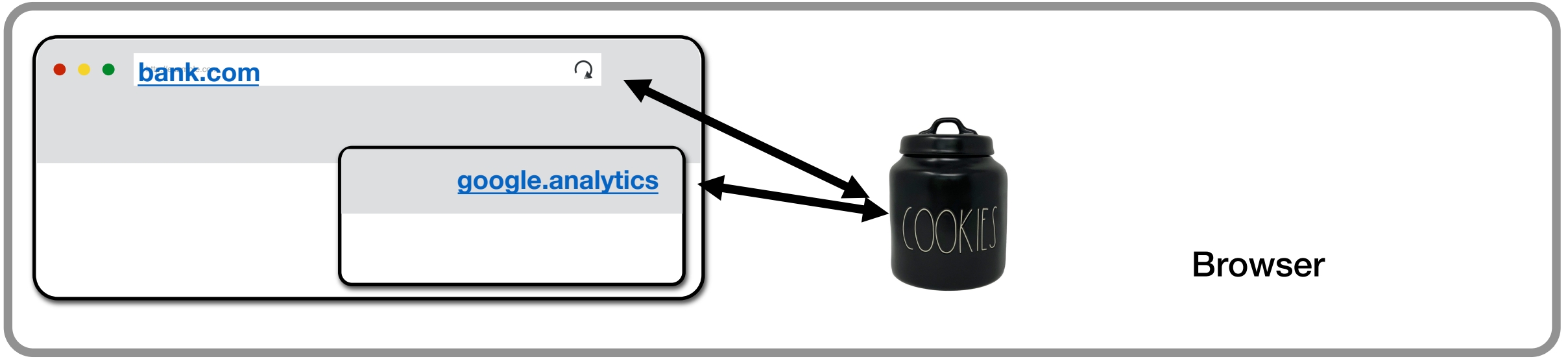
Login Session



Login Session



Can the following attack succeed?

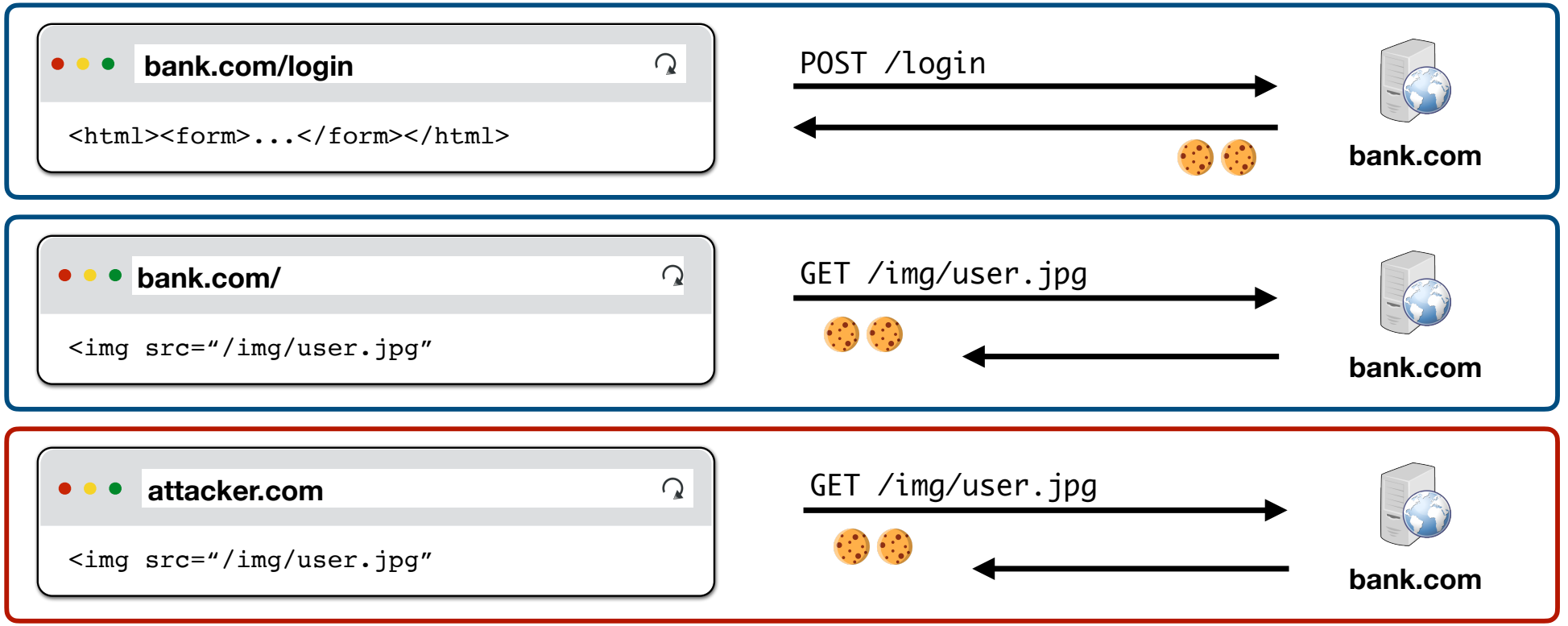


If we have a google analytics Javascript running on bank.com's login page. Assume that the site has no frames, and everything on this page has the same origin. Can google analytics see Alice's session cookie on bank.com?

A. Yes B. No C. Maybe D. Something Else

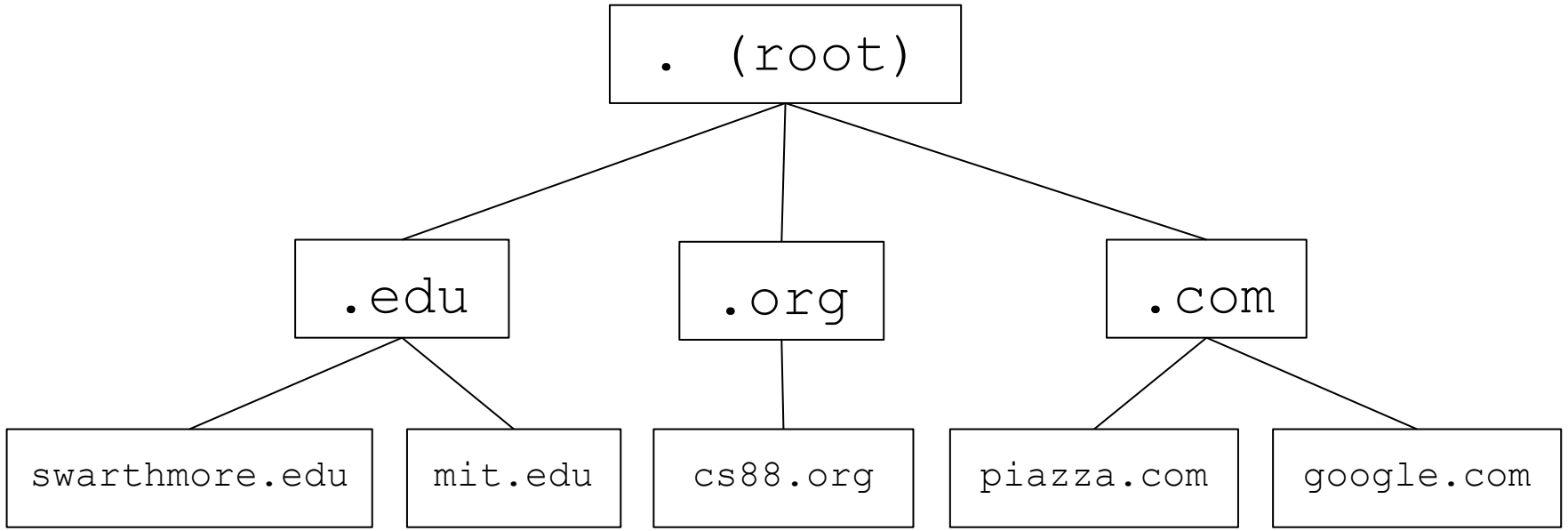
Cookies

“In scope” cookies are sent based on origin regardless of requester

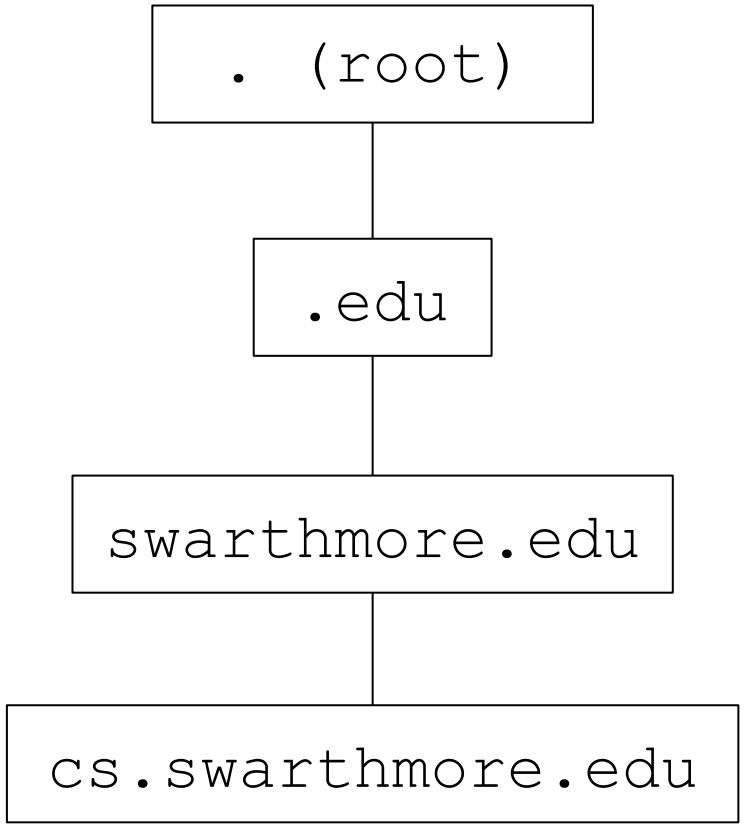


Aside: Domain Hierarchy

- Domains
  - Located after the double slashes, but before the next single slash
  - Written as several phrases separated by dots
- Domains can be sorted into a hierarchy
  - The hierarchy is separated by dots



Aside: Domain Hierarchy



`.edu` is a **top-level domain** (TLD), because it is directly below the root of the tree.

`swarthmore.edu` is a **subdomain** of `edu`

`cs.swarthmore.edu` is a **subdomain** of `swarthmore.edu`

Cookie Policy: Setting Cookies

- When the browser receives a cookie from a server, should the cookie be accepted?
- Server with domain X can set a cookie with domain attribute Y if
  - The domain attribute is a **domain suffix** of the server's domain
    - X ends in Y
    - X is below or equal to Y on the hierarchy
    - X is more specific or equal to Y
  - The domain attribute Y is not a top-level domain (TLD)
  - No restrictions for the Path attribute (the browser will accept any path)
- Examples:
  - mail.**google.com** can set cookies for Domain=google.com
  - **google.com** can set cookies for Domain=google.com
  - google.**com** **cannot** set cookies for Domain=com, because com is a top-level domain

Cookie Policy: Sending Cookies

- When the browser makes a request to a server, should the cookie be attached?
- The browser sends the cookie if both of these are true:
  - The domain attribute is a **domain suffix** of the server's domain
  - The path attribute is a **prefix** of the server's path

Cookie Policy: Sending Cookies

(server URL)

`https://cs88.swat.edu/cryptoverse/oneshots/subway.html`

`cs88.swat.edu` `/cryptoverse` ✅

(cookie domain) (cookie path)

Quick method to check cookie sending: Concatenate the cookie domain and path. Line it up below the requested URL at the first single slash.

If the domains and paths all match, then the cookie is sent.

Cookie Policy: Sending Cookies

(server URL)

`https://cs88.swat.org/cryptoverse/oneshots/subway.html`

`cs88.swat.org/xorcist` ✗

(cookie domain) (cookie path)

Quick method to check cookie sending: Concatenate the cookie domain and path. Line it up below the requested URL at the first single slash.

If the domain or path doesn't match, then the cookie is not sent.

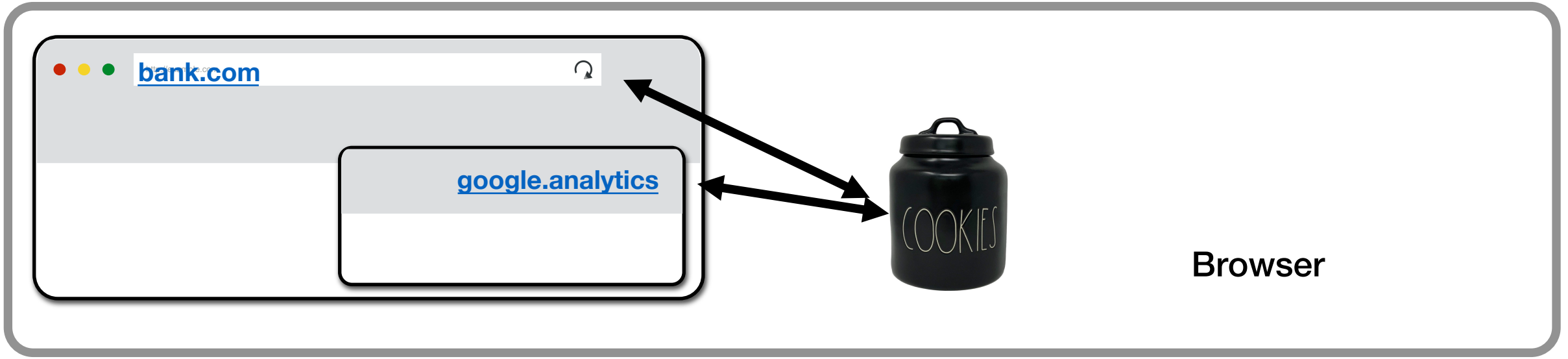
Scoping Example

name = cookie1 value = a domain = login.site.com path = /	name = cookie2 value = b domain = site.com path = /	name = cookie3 value = c domain = site.com path = /my/home

Concatenate the cookie domain and path. Line it up below the requested URL at the first single slash.

	Cookie 1	Cookie 2	Cookie 3
checkout.site.com			
login.site.com			
login.site.com/my/home			
site.com/account			

Can the following attack succeed?



If we have a google analytics Javascript running on bank.com's login page. Assume that the site has iframes. Can google analytics see Alice's session cookie on bank.com?

Cookies and web authentication

- An extremely common use of cookies is to track users who have already authenticated
- If the user already visited http://website.com/login.html?user=alice&pass=secret with the correct password, then the server associates a “session cookie” with the logged-in user’s info
- Subsequent requests (GET and POST) include the cookie in the request headers and/or as one of the fields: http://website.com/doStuff.html?sid=81asf98as8eak
- The idea is for the server to be able to say “I am talking to the same browser that authenticated Alice earlier.”

Aside: Trust in Web Advertising

- Advertising, by definition, is ceding control of Web content to another party
- Webmasters must trust advertisers not to show malicious content
- Sub-syndication allows advertisers to rent out their advertising space to other advertisers
  - Companies like Doubleclick have massive ad trading desks, also real-time auctions, exchanges, etc.
- Trust is not transitive!
  - Webmaster may trust his advertisers, but this does not mean he should trust those trusted by his advertisers

Aside: Example of an Advertising Exploit

[Provos et al.]

- Video sharing site includes a banner from a large US advertising company as a single line of JavaScript…
- … which generates JavaScript to be fetched from another large US company
- … which generates more JavaScript pointing to a smaller US company that uses geo-targeting for its ads
- … the ad is a single line of HTML containing an iframe to be fetched from a Russian advertising company
- … when retrieving iframe, "Location:" header redirects browser to a certain IP address
- … which serves encrypted JavaScript, attempting multiple exploits against the browser

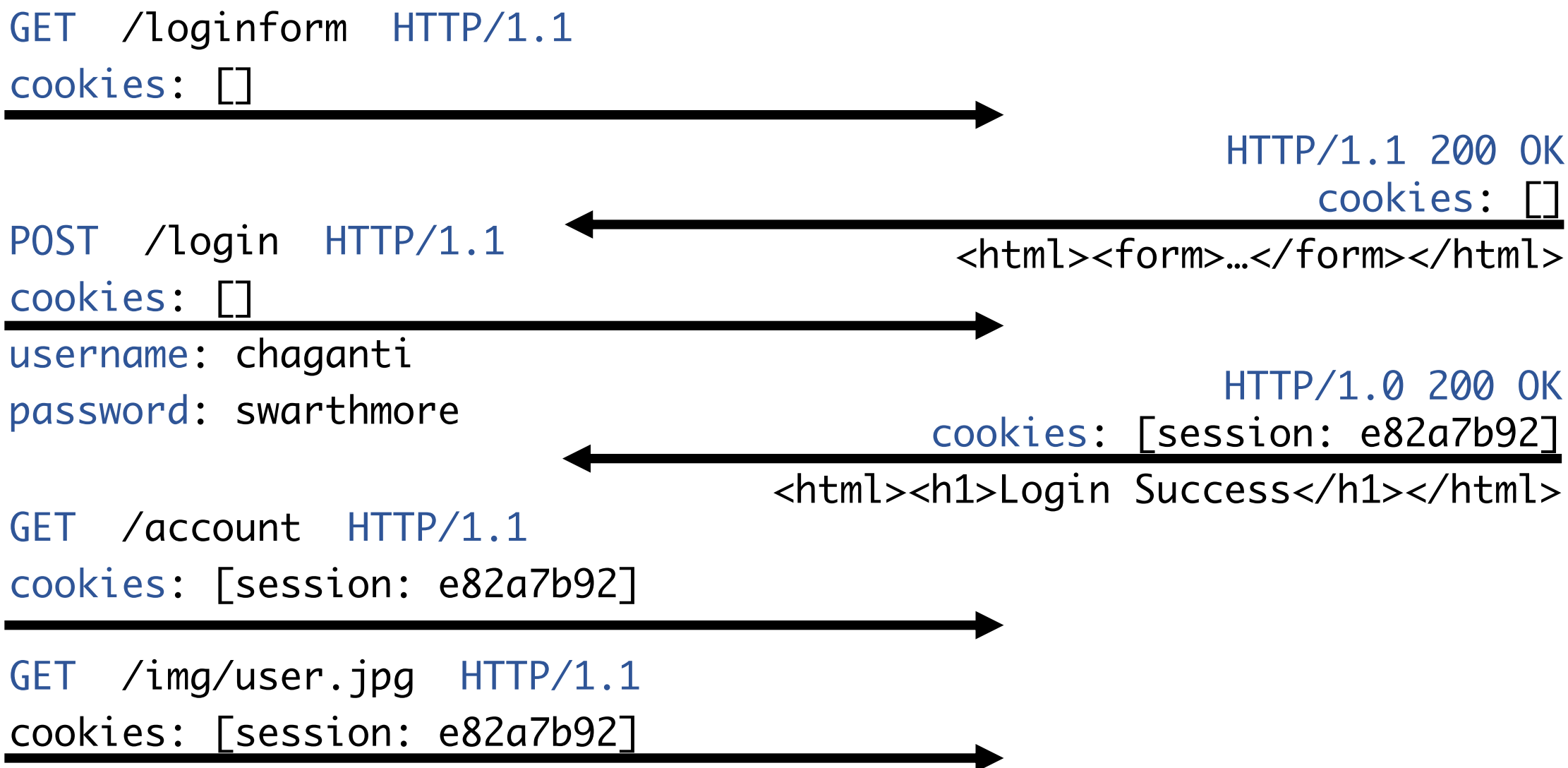
Aside: Third-Party Widgets

[Provos et al.]

- Make sites "prettier" using third-party widgets
  - Calendars, visitor counters, etc.
- Example: free widget for keeping visitor statistics operates fine from 2002 until 2006
- In 2006, widget starts pushing exploits to all visitors of pages linked to the counter

http://expl.info/cgi-bin/ie0606.cgi?homepage
http://expl.info/demo.php
http://expl.info/cgi-bin/ie0606.cgi?type=MS03-11&SP1
http://expl.info/ms0311.jar
http://expl.info/cgi-bin/ie0606.cgi?exploit=MS03-11
http://dist.info/f94mslrfum67dh/winus.exe

Login Session



Session Tokens: Security

- If an attacker steals your session token, they can log in as you!
  - The attacker can make requests and attach your session token
  - The browser will think the attacker's requests come from you
- Servers need to generate session tokens *randomly* and *securely*
- Browsers need to make sure malicious websites cannot steal session tokens
  - Enforce isolation with cookie policy and same-origin policy
- Browsers should not send session tokens to the wrong websites
  - Enforced by cookie policy

Session Token Cookie Attributes

What attributes should the server set for the session token?

- Domain and Path: Set so that the cookie is only sent on requests that require authentication
- Secure: Can set to True to so the cookie is only sent over secure HTTPS connections
- HttpOnly: Can set to True so JavaScript can't access session tokens
- Expires: Set so that the cookie expires when the session times out

Name	**token**
Value	**{random value}**
Domain	**mail.google.com**
Path	**/**
Secure	**True**
HttpOnly	**True**
Expires	**{15 minutes later}**
(*other fields omitted*)	