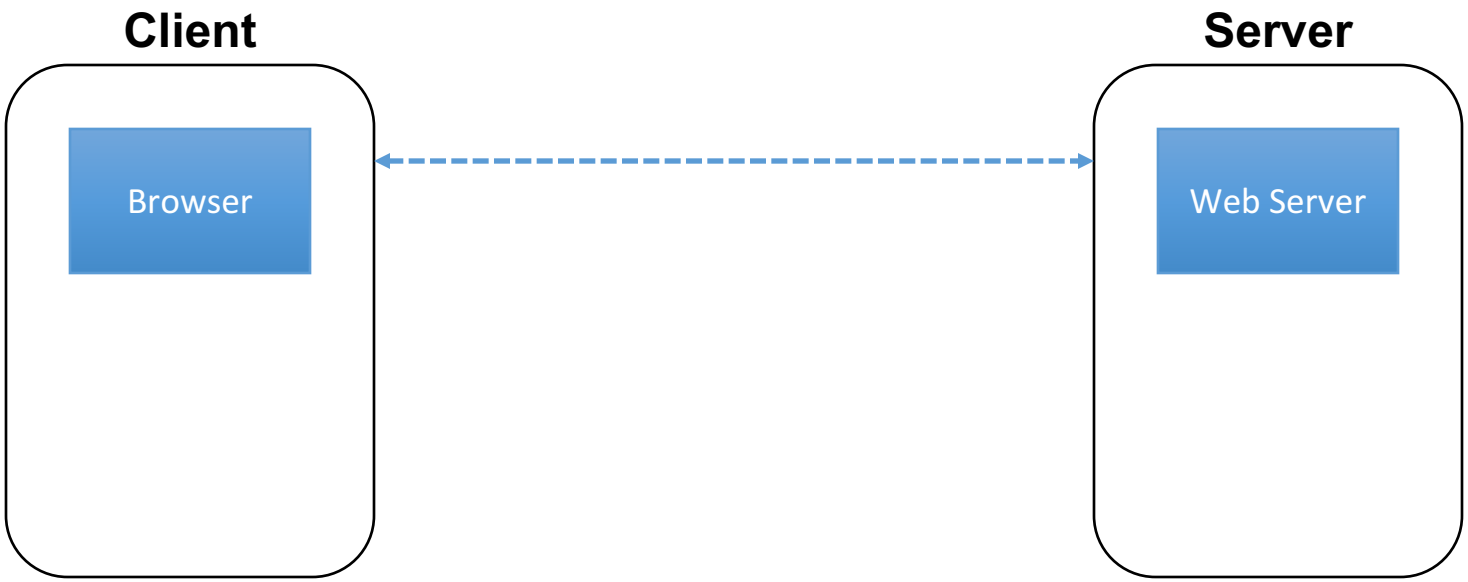
Lab 6

Cross-Site Request Forgery (CSRF)

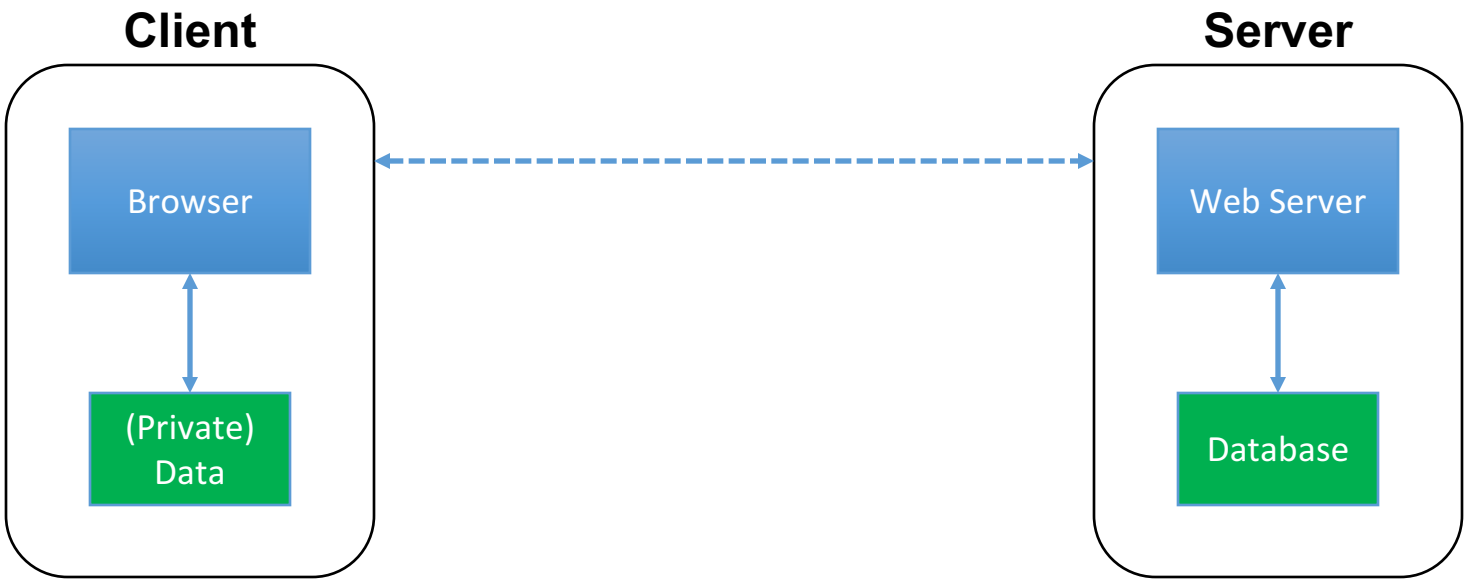
Web Basics

The Web, Basically

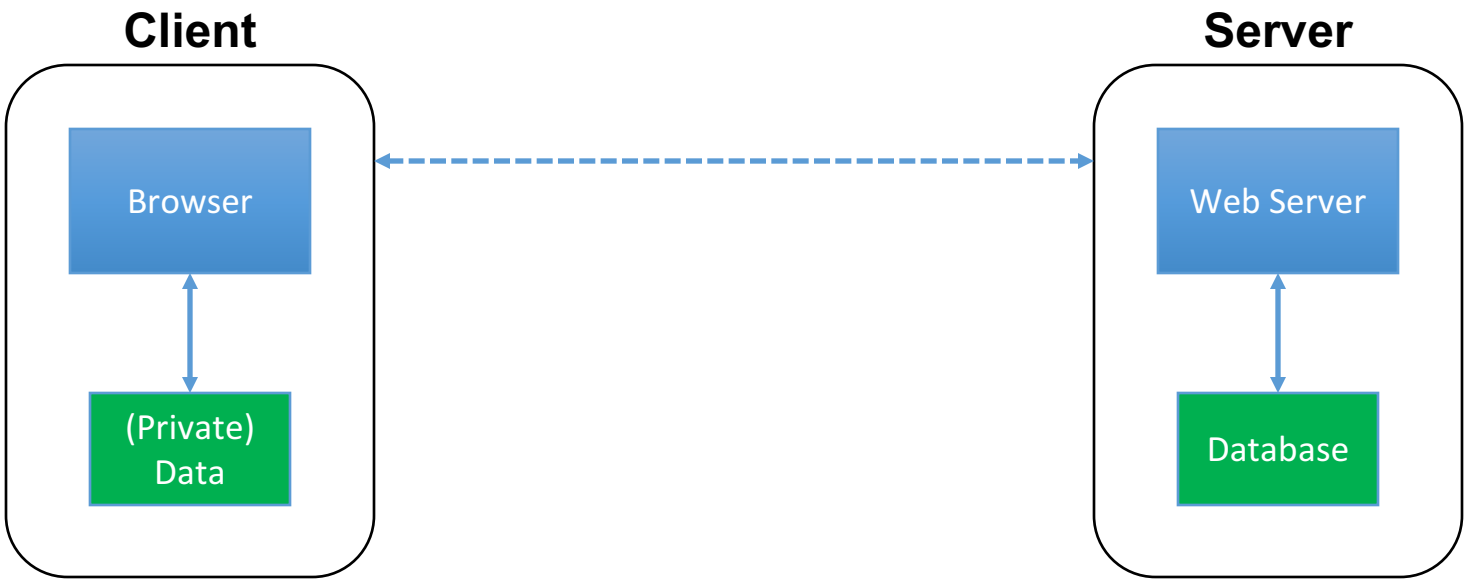
The Web, Basically



The Web, Basically

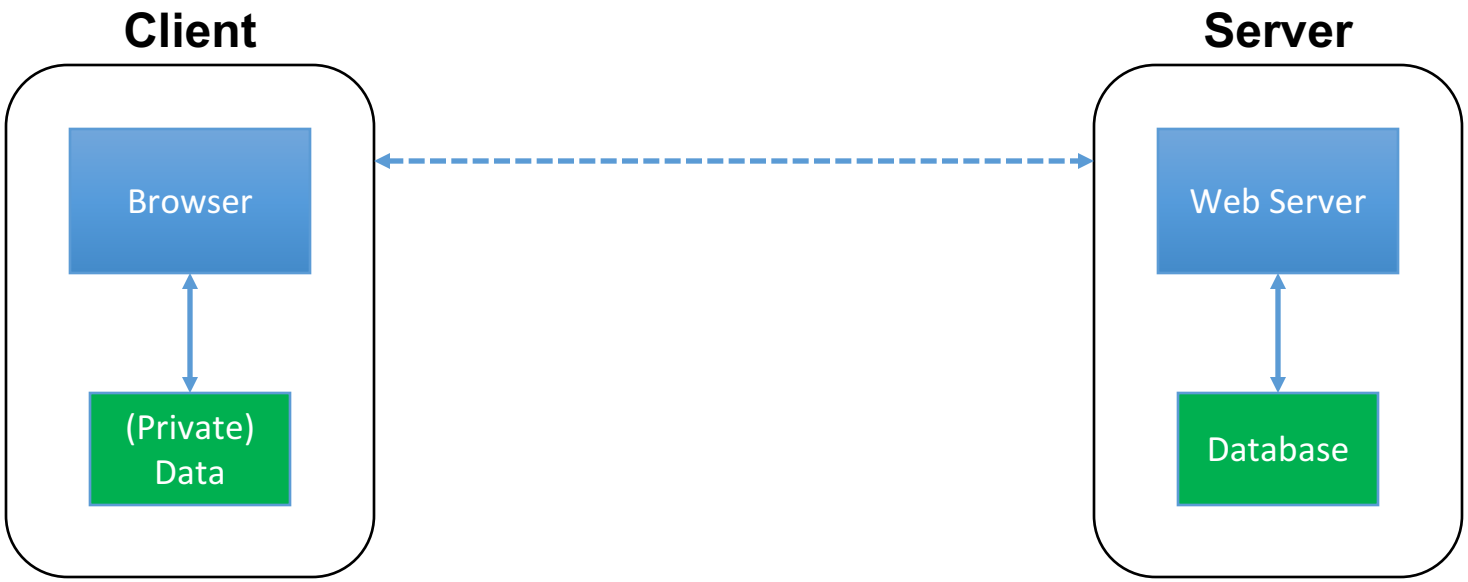


The Web, Basically



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

The Web, Basically



(Much) user data is part of the browser

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

Interacting with web servers

Resources which are identified by a URL

(Universal Resource Locator)

http://pages.erau.edu/~yuanj/index.html

Interacting with web servers

Resources which are identified by a URL

(Universal Resource Locator)

http://pages.erau.edu/~yuanj/index.html

Protocol

ftp

https

Interacting with web servers

Resources which are identified by a URL

(Universal Resource Locator)

http://pages.erau.edu/~yuanj/index.html

Hostname/server

Translated to an IP address by DNS
(e.g., 128.8.127.3)

Interacting with web servers

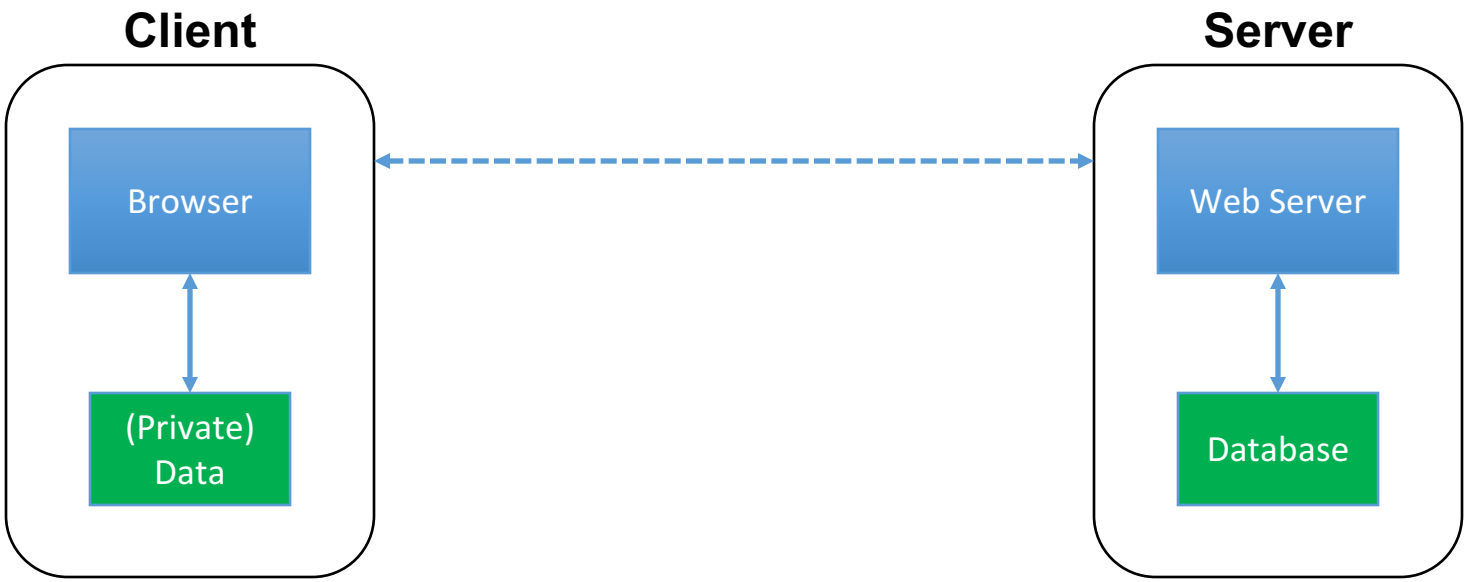
Resources which are identified by a URL

(Universal Resource Locator)

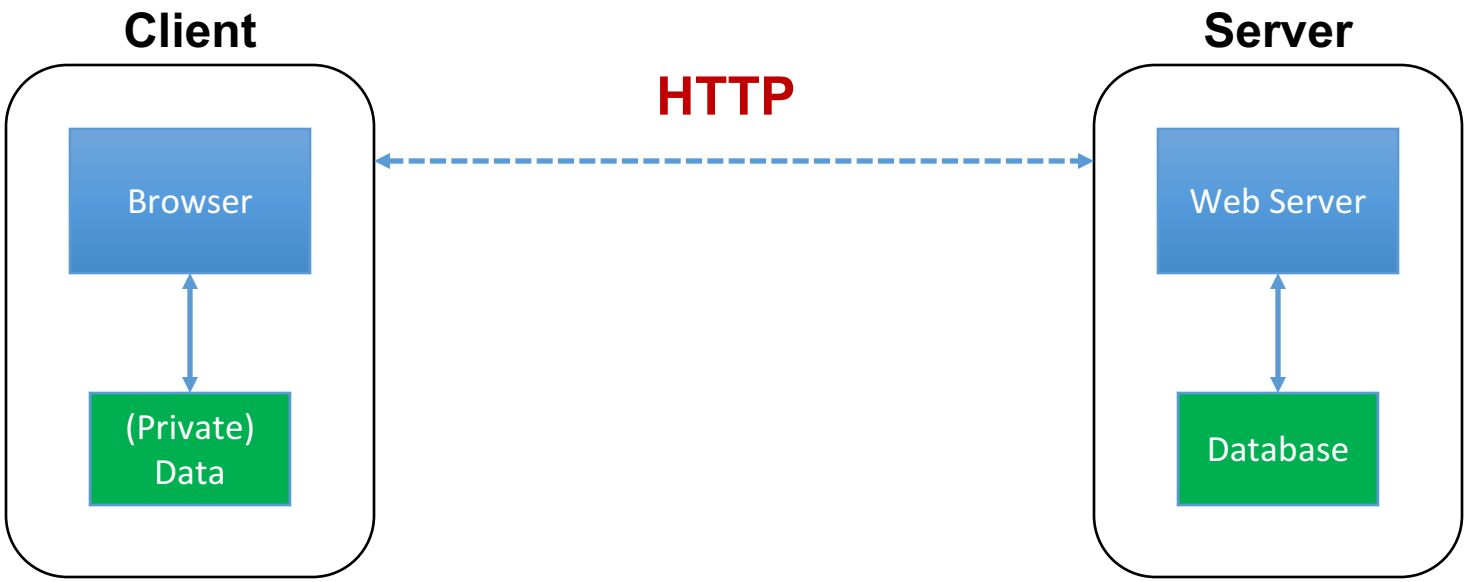
http://pages.erau.edu/~yuanj/index.html

Path to a resource

Basic structure of web traffic



Basic structure of web traffic



HyperText Transfer Protocol (HTTP)

Basic structure of web traffic

User clicks

Basic structure of web traffic

User clicks

Basic structure of web traffic

- **Requests contain**:
  - The **URL** of the resource the client wishes to obtain
  - **Headers** describing what the browser can do
- **Request types** can be **GET** or **POST**
  - **GET**: all data is in the URL itself (no server side effects)
  - **POST**: includes the data as separate fields (can have side effects)

HTTP Get Request

http://www.reddit.com/r/security

HTTP Headers

```
http://www.reddit.com/r/security

GET /r/security HTTP/1.1
Host: www.reddit.com
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
Cookie: __utma=55650728.562667657.1392711472.1392711472.1392711472.1; __utmb=55650728.1.10.1392711472; __utmc=55650...
```

HTTP Get Request

http://www.reddit.com/r/security

HTTP Headers

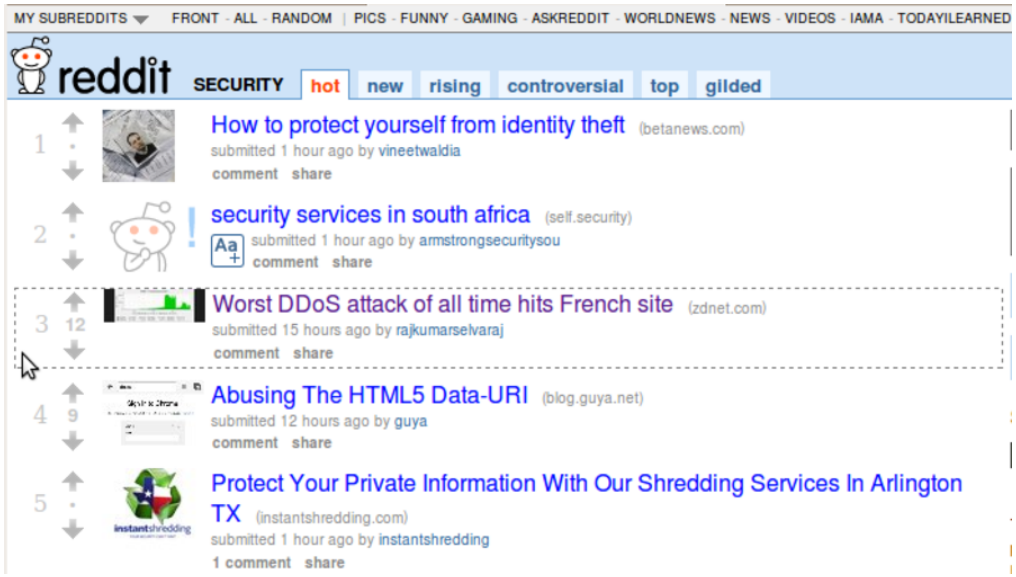
```
http://www.reddit.com/r/security

GET /r/security HTTP/1.1
Host: www.reddit.com
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
Cookie: __utma=55650728.562667657.1392711472.1392711472.1392711472.1; __utmb=55650728.1.10.1392711472; __utmc=55650...
```

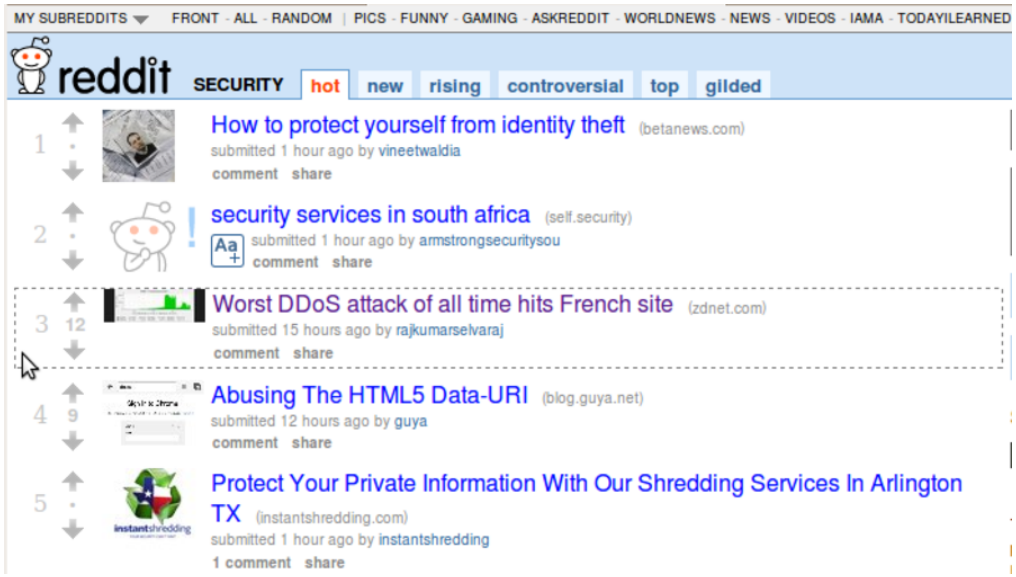
User-Agent is typically a **browser**

But it can be wget, JDK, etc

HTTP Get Request



HTTP Get Request



HTTP Headers

http://www.zdnet.com/worst-ddos-attack-of-all-time-hits-french-site-7000026330/

```
GET /worst-ddos-attack-of-all-time-hits-french-site-7000026330/ HTTP/1.1
Host: www.zdnet.com
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
```

Referrer URL: the site from which this request was issued

HTTP Post Request

Posting on a website “piazza”

HTTP Headers

```
https://piazza.com/logic/api?method=content.create&aid=hrteve7t83et

POST /logic/api?method=content.create&aid=hrteve7t83et HTTP/1.1
Host: piazza.com
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: application/json, text/javascript, */*; q=0.01
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
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: https://piazza.com/class
Content-Length: 339
Cookie: piazza_session="DFwuCEFIGvEGwwHLJyuCvHIGtHKECCKL.5%25x+x+ux%255M5%22%215%3F5%26x%26%26%7C%22%21r...
Pragma: no-cache
Cache-Control: no-cache
    {"method":"content.create","params":{"cid":"hrpng9q2nndos","subject":"<p>Interesting.. perhaps it has to do with a change to the ...
```

HTTP Post Request

Posting on a website “piazza”

HTTP Headers

```
https://piazza.com/logic/api?method=content.create&aid=hrteve7t83et

POST /logic/api?method=content.create&aid=hrteve7t83et HTTP/1.1
Host: piazza.com
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: application/json, text/javascript, */*; q=0.01
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
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: https://piazza.com/class
Content-Length: 339
Cookie: piazza_session="DFwuCEFIGvEGwwHLJyuCvHIGtHKECCKL.5%25x+x+ux%255M5%22%215%3F5%26x%26%26%7C%22%21r...
Pragma: no-cache
Cache-Control: no-cache
    {"method":"content.create","params":{"cid":"hrpng9q2nndos","subject":"<p>Interesting.. perhaps it has to do with a change to the ...
```

Explicitly includes data as a part of the request’s content

HTTP Post Request

Posting on a website "piazza"

HTTP Headers

https://piazza.com/logic/api?method=content.create&aid=hrteve7t83et

POST /logic/api?method=content.create&aid=hrteve7t83et HTTP/1.1
Host: piazza.com
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: application/json, text/javascript, */*; q=0.01
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
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: https://piazza.com/class
Content-Length: 339
Cookie: piazza_session="DFwuCEFIGvEGwwHLJyuCvHIGtHKECCKL.5%25x+x+ux%255M5%22%215%3F5%26x%26%26%7C%22%21r...
Pragma: no-cache
Cache-Control: no-cache
{"method":"content.create","params":{"cid":"hrpng9q2nndos","subject":"<p>Interesting.. perhaps it has to do with a change to the ...

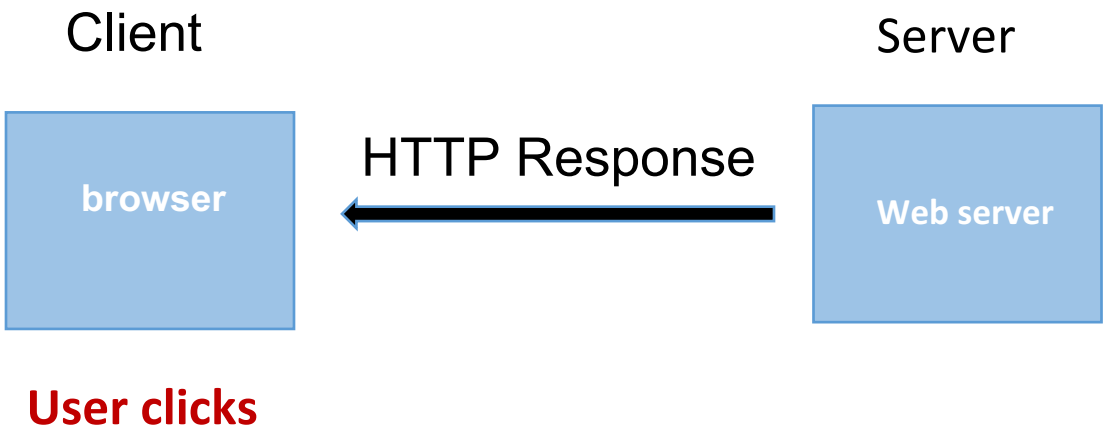
Implicitly includes data as a part of the URL

Explicitly includes data as a part of the request's content

Basic structure of web traffic

User clicks

Basic structure of web traffic



Responses contain**:**

- **Status** code
- **Headers** describing what the server provides
- **Data**
- **Cookies**: Represent state the server would like the browser to store on its behalf

HTTP Response

HTTP version

Status code

Reason phrase

Headers

Data

```
HTTP/1.1 200 OK
Date: Tue, 18 Feb 2014 08:20:34 GMT
Server: Apache
Set-Cookie: session-zdnet-production=6bhqca1i0cbciagu11sisac2p3; path=/; domain=zdnet.com
Set-Cookie: zdregion=MTI5LjIuMTI5LjE1Mzp1czp1czpjZDjmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN
Set-Cookie: zdregion=MTI5LjIuMTI5LjE1Mzp1czp1czpjZDjmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN
Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com
Set-Cookie: session-zdnet-production=59ob97fpinqe4bg6lde4dvvq11; path=/; domain=zdnet.com
Set-Cookie: user_agent=desktop
Set-Cookie: zdnet_ad_session=f
Set-Cookie: firstpg=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
X-UA-Compatible: IE=edge,chrome=1
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 18922
Keep-Alive: timeout=70, max=146
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<html> ..... </html>
```

Web-based State using Hidden Fields and Cookies

HTTP is stateless

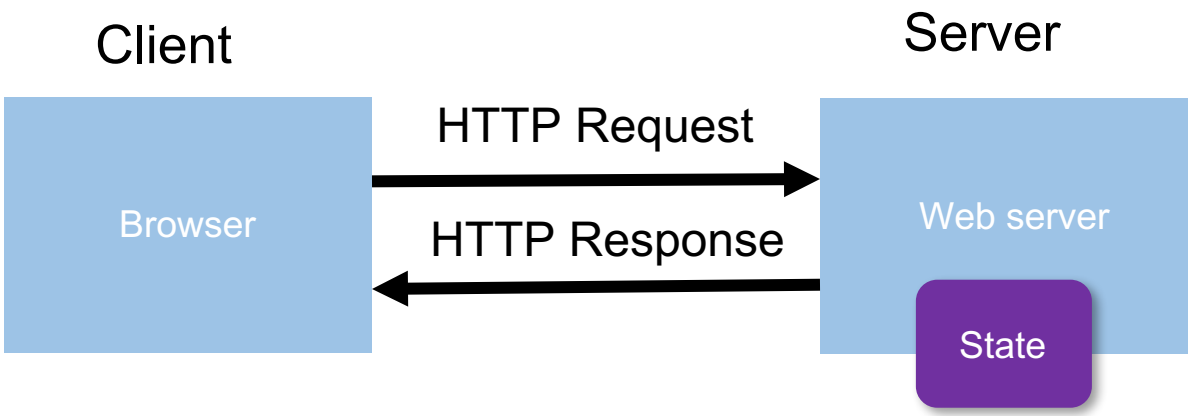
- The lifetime of an HTTP session is typically:
  - Client connects to the server
  - Client issues a request
  - Server responds
  - Client issues a request for something in the response
  - …. repeat ….
  - Client disconnects

HTTP is stateless

- The lifetime of an HTTP session is typically:
  - Client connects to the server
  - Client issues a request
  - Server responds
  - Client issues a request for something in the response
  - …. repeat ….
  - Client disconnects

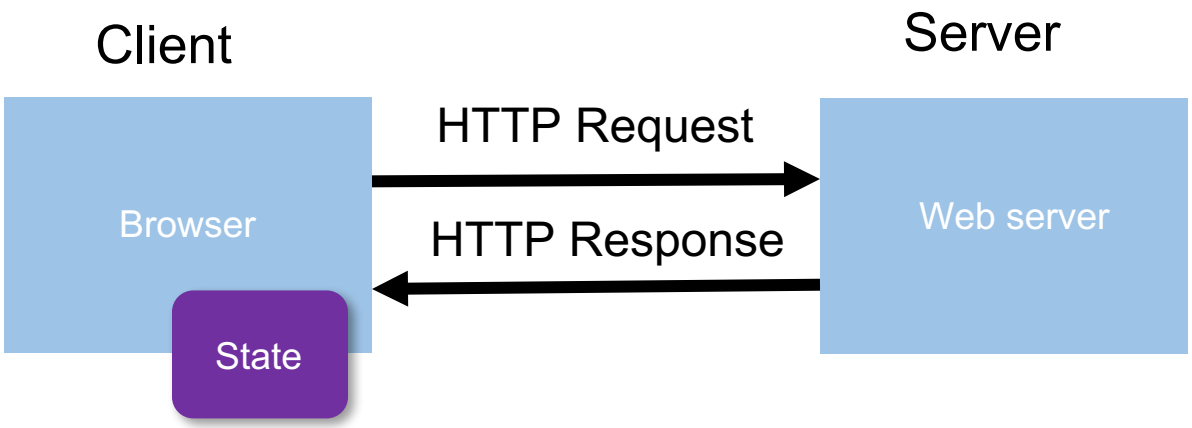
- HTTP has no means of noting “oh this is the same client from that previous session”

Maintaining State



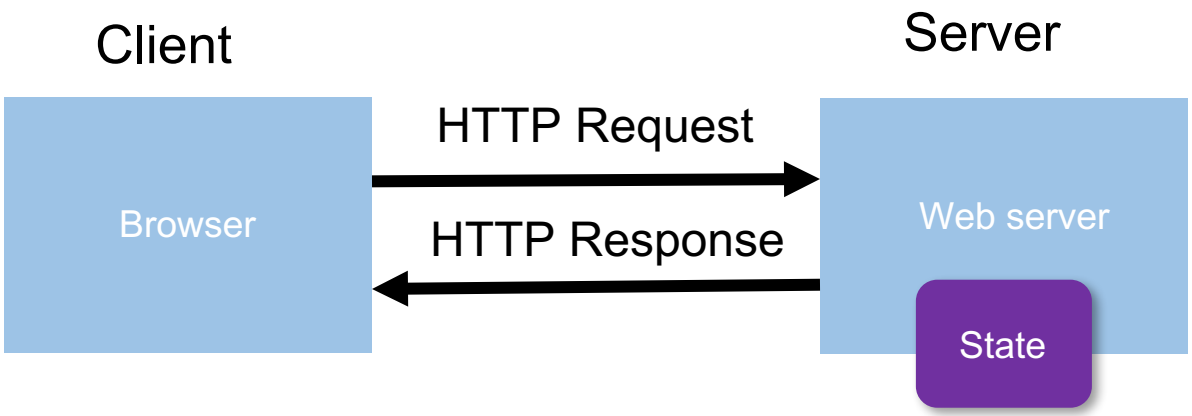
- Web application maintains *ephemeral* state
  - Server processing often produces intermediate results
    - Not ACID, long-lived state

Maintaining State



- Web application maintains *ephemeral* state
  - Server processing often produces intermediate results
    - Not ACID, long-lived state
  - **Send** such **state to the client**

Maintaining State



- Web application maintains *ephemeral* state
  - Server processing often produces intermediate results
    - Not ACID, long-lived state
  - **Send** such **state to the client**
  - Client **returns the state** in subsequent **requests**

  - Two kinds of state: hidden fields, and cookies

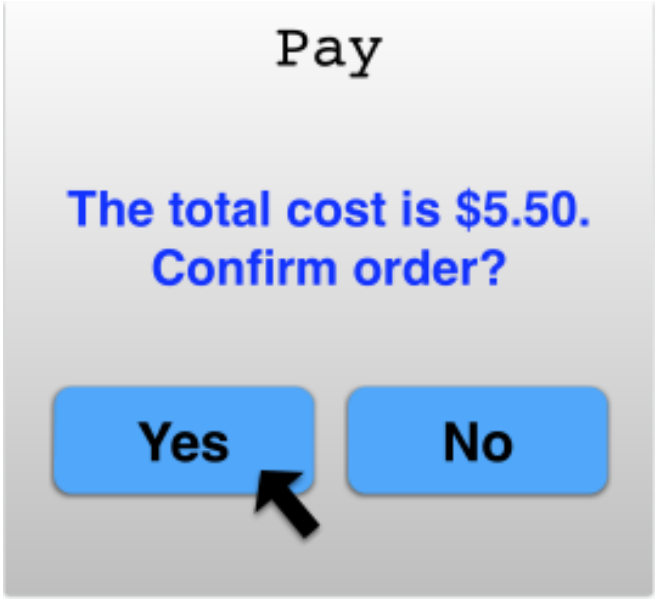
Ex: Online ordering

socks.com/order.php

Ex: Online ordering

socks.com/order.php

socks.com/pay.php



Separate page

Ex: Online ordering

What's presented to the user

Pay.php

```
<html>
<head> <title>Pay</title> </head>
<body>

<form action="submit_order" method="GET">
The total cost is $5.50. Confirm order?
<input type="hidden" name="price" value="5.50">
<input type="submit" name="pay" value="yes">
<input type="submit" name="pay" value="no">

</body>
</html>
```

Ex: Online ordering

The corresponding backend processing

```
if(pay == yes && price != NULL)
{
! bill_creditcard(price);
! deliver_socks();
}
else
! display_transaction_cancelled_page();
```

Ex: Online ordering

What's presented to the user

```
<html>
<head> <title>Pay</title> </head>
<body>

<form action="submit_order" method="GET">
The total cost is $5.50. Confirm order?
<input type="hidden" name="price" value="5.50">
<input type="submit" name="pay" value="yes">
<input type="submit" name="pay" value="no">

</body>
</html>
```

Ex: Online ordering

What's presented to the user

```
<html>
<head> <title>Pay</title> </head>
<body>

<form action="submit_order" method="GET">
The total cost is $5.50. Confirm order?
<input type="hidden" name="price" value="0.01">
<input type="submit" name="pay" value="yes">
<input type="submit" name="pay" value="no">

</body>
</html>
```

Client can change the value!

Solution: Capabilities

- **Server maintains *trusted* state** (while client maintains the rest)
  - Server stores intermediate state
  - Send a capability to access that state to the client
  - Client **references the capability** in subsequent responses

- **Capabilities should be large, random numbers**, so that they are hard to guess
  - To prevent illegal access to the state

Ex: Online ordering

What's presented to the user

```
<html>
<head> <title>Pay</title> </head>
<body>

<form action="submit_order" method="GET">
The total cost is $5.50. Confirm order?
<input type="hidden" name="price" value="5.50">
<input type="submit" name="pay" value="yes">
<input type="submit" name="pay" value="no">

</body>
</html>
```

Using capabilities

What's presented to the user

Capability;
the system will detect a change and abort

```
<html>
<head> <title>Pay</title> </head>
<body>
!
<form action="submit_order" method="GET">
The total cost is $5.50. Confirm order?
<input type="hidden" name="sid" value="781234">
<input type="submit" name="pay" value="yes">
<input type="submit" name="pay" value="no">

</body>
</html>
```

Using capabilities

The corresponding backend processing

```
price = lookup(sid);
if(pay == yes && price != NULL)
{
  bill_creditcard(price);
  deliver_socks();
}
else
  display_transaction_cancelled_page();
```

Using capabilities

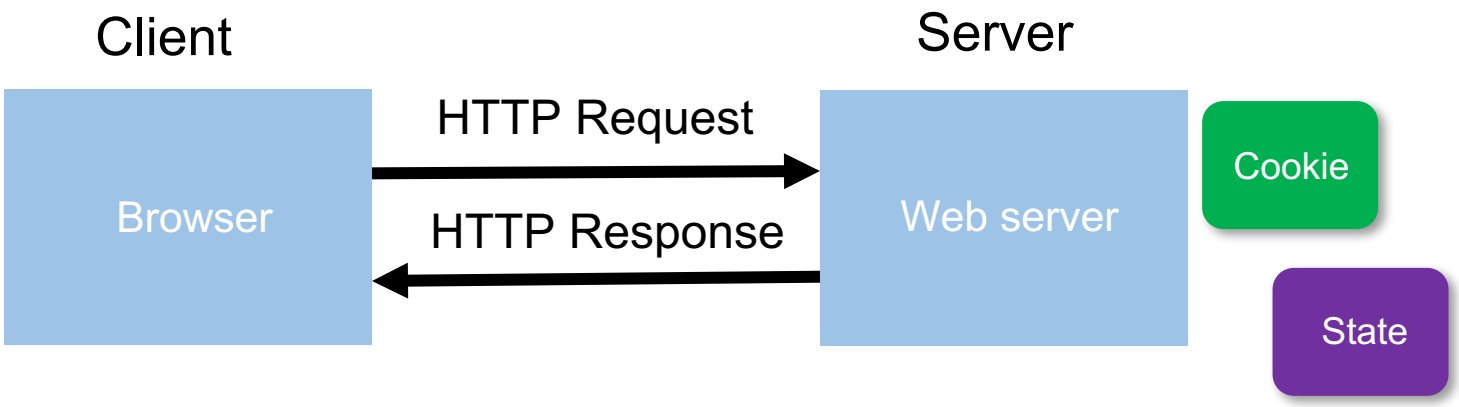
The corresponding backend processing

```
price = lookup(sid);
if(pay == yes && price != NULL)
{
  bill_creditcard(price);
  deliver_socks();
}
else
  display_transaction_cancelled_page();
```

But: we don't want to pass hidden fields around all the time

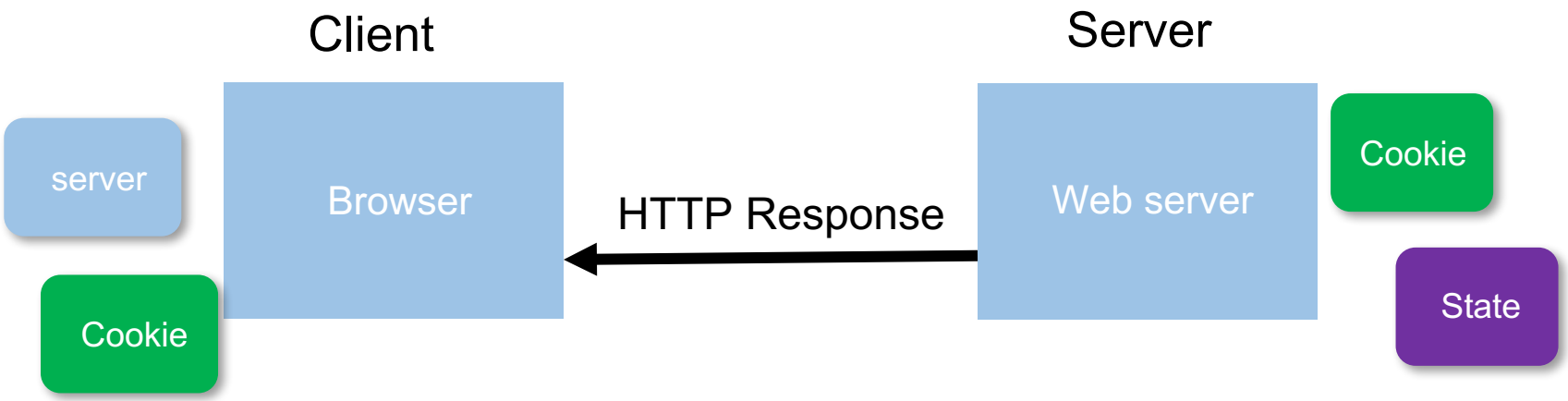
- Tedious to add/maintain on all the different pages
- Have to start all over on a return visit (after closing browser window)

Statefulness with Cookies



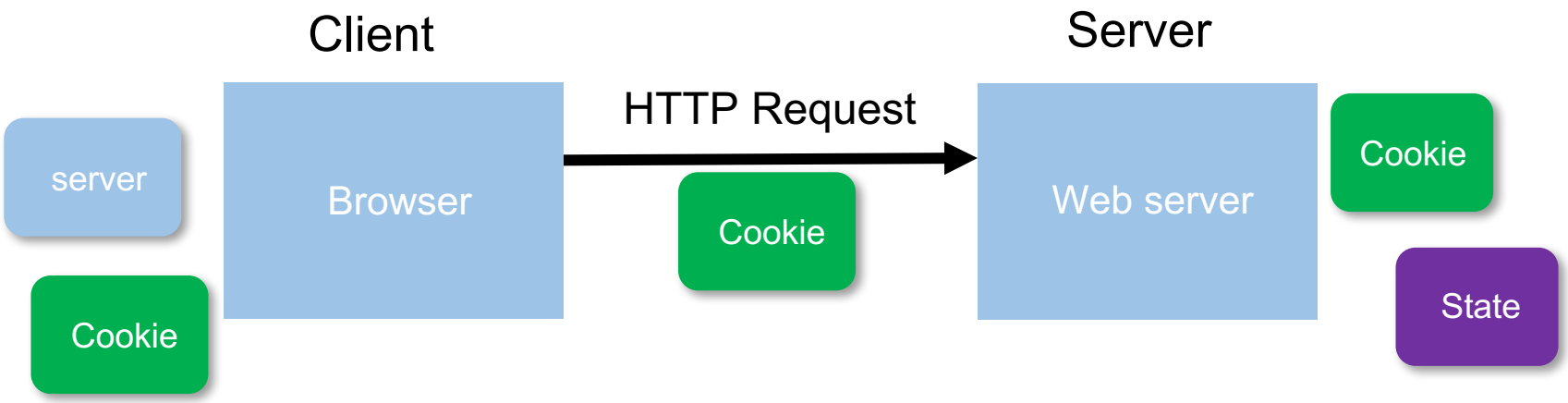
- Server **maintains trusted state**
  - Server Server indexes/denotes state with a **cookie**

Statefulness with Cookies



- Server **maintains trusted state**
  - Server Server indexes/denotes state with a **cookie**
  - Sends cookie to the client, which stores it

Statefulness with Cookies



- Server **maintains trusted state**
  - Server Server indexes/denotes state with a **cookie**
  - Sends cookie to the client, which stores it
  - Client returns it with subsequent queries to that same server

Ex: Online ordering

Set-Cookie:key=value; options; ….

Headers

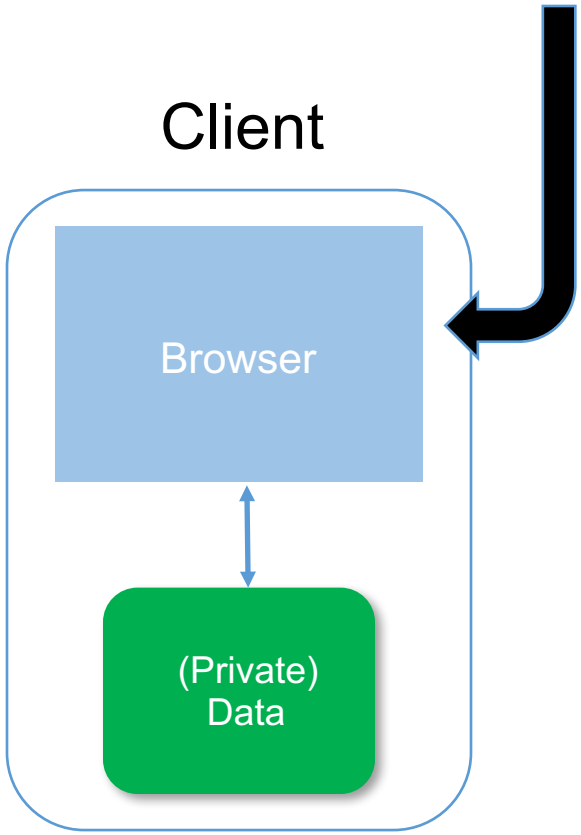
```
HTTP/1.1 200 OK
Date: Tue, 18 Feb 2014 08:20:34 GMT
Server: Apache
Set-Cookie: session-zdnet-production=6bhqca1i0cbciagu11sisac2p3; path=/; domain=zdnet.com
Set-Cookie: zdregion=MTI5LjIuMTI5LjE1Mzp1czp1czpjZDJmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN
Set-Cookie: zdregion=MTI5LjIuMTI5LjE1Mzp1czp1czpjZDJmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN
Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com
Set-Cookie: session-zdnet-production=59ob97fpinqe4bg6lde4dvvq11; path=/; domain=zdnet.com
Set-Cookie: user_agent=desktop
Set-Cookie: zdnet_ad_session=f
Set-Cookie: firstpg=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
X-UA-Compatible: IE=edge,chrome=1
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 18922
Keep-Alive: timeout=70, max=146
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

Data

```
<html> …… </html>
```

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”
- This value is no good as of Wed Feb 18…
- This value should only be readable by any domain ending in **.zdnet.com**
- Send the cookie with any future requests to <domain>/<path>

Requests with cookies

```
HTTP/1.1 200 OK
Date: Tue, 18 Feb 2014 08:20:34 GMT
Server: Apache
Set-Cookie: session-zdnet-production=6bhqca1i0cbciagu11sisac2p3; path=/; domain=zdnet.com
Set-Cookie: zdregion=MTI5LjIuMTI5LjE1Mzp1czp1czpjZDJmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN
Set-Cookie: zdregion=MTI5LjIuMTI5LjE1Mzp1czp1czpjZDJmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN
Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com
Set-Cookie: session-zdnet-production=59ob97fpinqe4bg6lde4dvvq11; path=/; domain=zdnet.com
```

Subsequent visit

```
HTTP Headers
  http://zdnet.com/

  GET / HTTP/1.1
  Host: zdnet.com
  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
  Cookie: session-zdnet-production=59ob97fpinqe4bg6lde4dvvq11; zdregion=MTI5LjIuMTI5LjE1Mzp1czp1czpjZDJmNW
```

Session Hijacking

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

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 include the cookie in the request headers and/or as one of the fields: http://website.com/doStuff.html?sid=81asf98as8eak

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 include the cookie in the request headers and/or as one of the fields: http://website.com/doStuff.html?sid=81asf98as8eak

- The idea is to be able to say "I am talking to the same browser that authenticated Alice earlier."

Cookie Theft

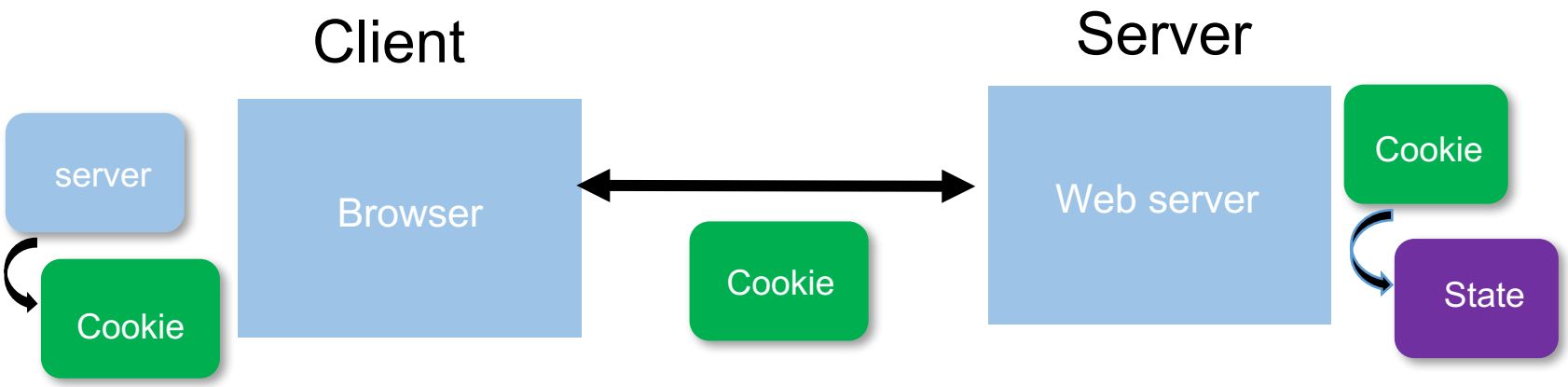
- **Session cookies** are, once again, **capabilities**
  - The holder of a session cookie gives access to a site with the privileges of the user that established that session

Cookie Theft

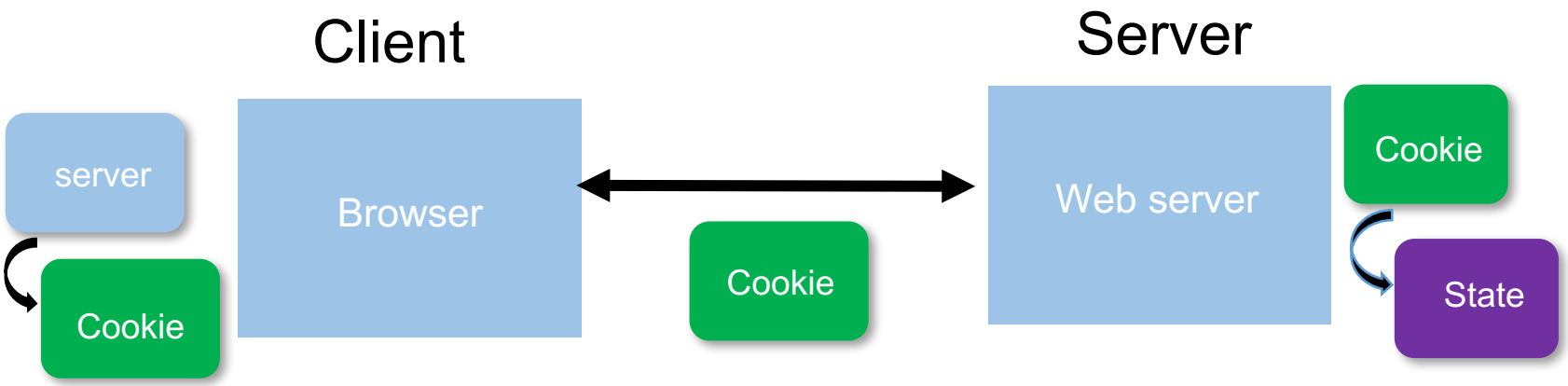
- **Session cookies** are, once again, **capabilities**
  - The holder of a session cookie gives access to a site with the privileges of the user that established that session

- Thus, **stealing a cookie** may allow an attacker to **impersonate a legitimate user**
  - Actions that will seem to be due to that user
  - Permitting theft or corruption of sensitive data

Stealing Session Cookies

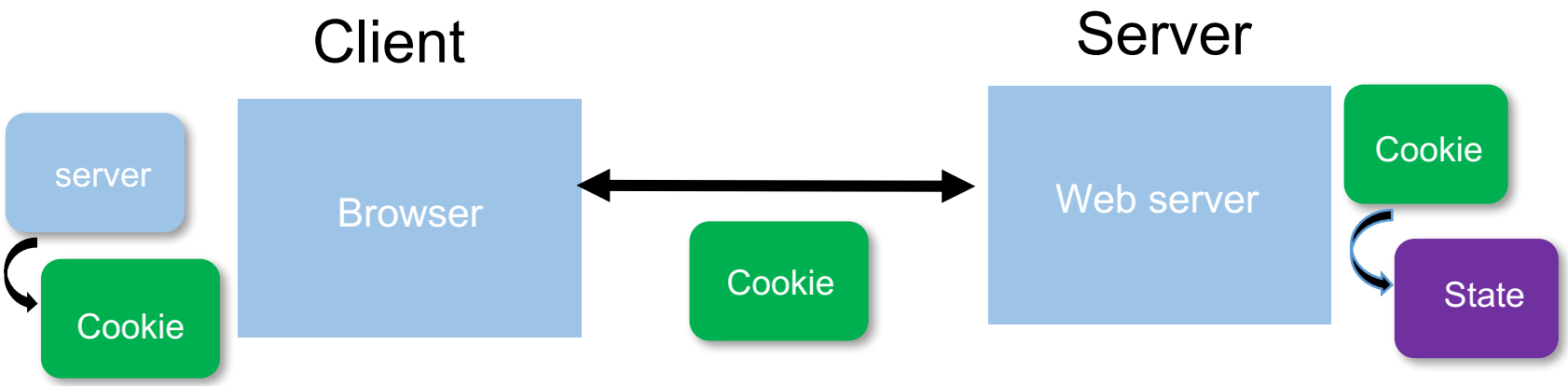


Stealing Session Cookies



- **Compromise** the server or user's machine/browser
- **Predict** it based on other information you know
- **Sniff** the network
- **DNS cache poisoning**
  - Trick the user into thinking you are Facebook
  - The user will send you the cookie

Stealing Session Cookies



- **Compromise** the server or user's machine/browser
- **Predict** it based on other information you know
- **Sniff** the network
- **DNS cache poisoning**
  - Trick the user into thinking you are Facebook
  - The user will send you the cookie

Network-based attacks

Defense: Unpredictability

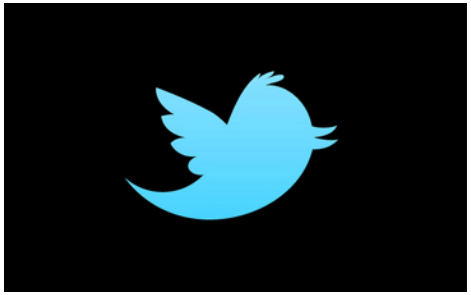
- **Avoid theft by guessing**; cookies should be
  - **Randomly** chosen,
  - Sufficiently **long**

    (Same goes with hidden field identifiers)

Mitigating Hijack

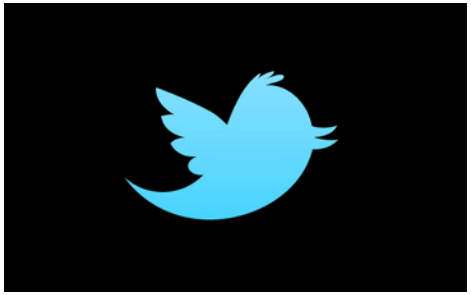
- Sad story: **Twitter**
- Uses one cookie (**auth_token**) to validate user, which is a function of
  - User name, password

Mitigating Hijack



- Sad story: **Twitter**
- Uses one cookie (**auth_token**) to validate user, which is a function of
  - User name, password
- **auth_token** weaknesses
  - Does not change from one login to the next
  - Does not become invalid when the user logs out
  - Thus: **steal this cookie once**, and you can **log in as the user any time you want** (until password change)

Mitigating Hijack



- Sad story: **Twitter**
- Uses one cookie (**auth_token**) to validate user, which is a function of
  - User name, password
- **auth_token** weaknesses
  - Does not change from one login to the next
  - Does not become invalid when the user logs out
  - Thus: **steal this cookie once**, and you can **log in as the user any time you want** (until password change)
- **Defense**: **Time out** session IDs and **delete** them once the session ends

http://packetstormsecurity.com/files/119773/twitter-cookie.txt

Cross-Site Request Forgery (CSRF)

URLs with side effects

http://bank.com/transfer.cgi?amt=9999&to=attacker

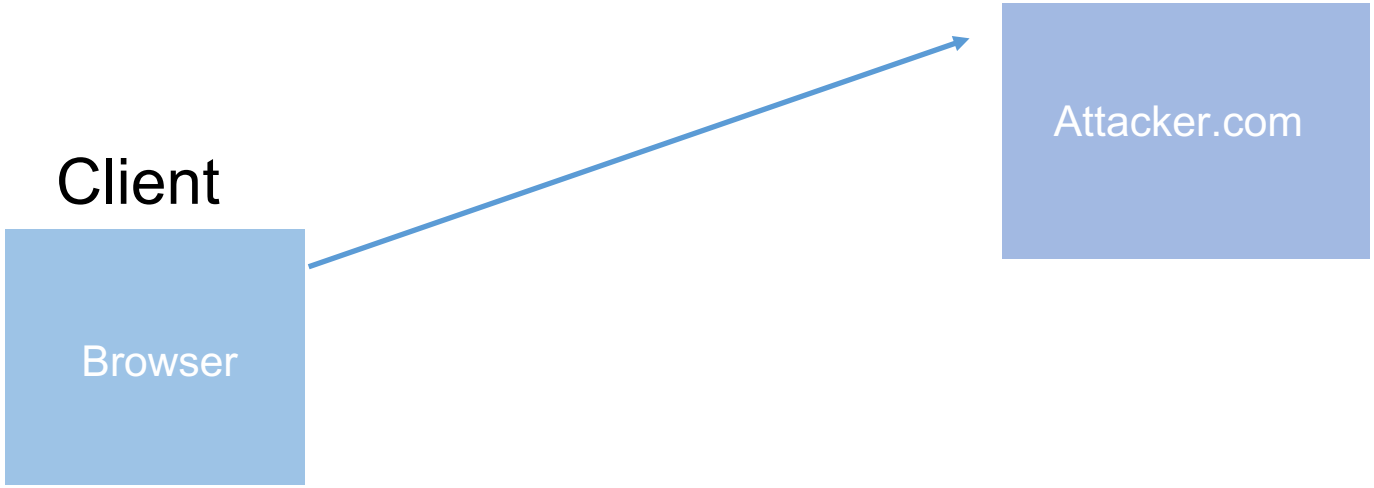
- GET requests can have **side effects on server state**
  - Even though they are not supposed to
- What happens if
  - the **user is logged in** with an active session cookie
  - a **request is issued for the above link?**

URLs with side effects

http://bank.com/transfer.cgi?amt=9999&to=attacker

- GET requests can have **side effects on server state**
  - Even though they are not supposed to
- What happens if
  - the **user is logged in** with an active session cookie
  - a **request is issued for the above link?**
- How could you get a user to visit a link?

Exploiting URLs with side-effects

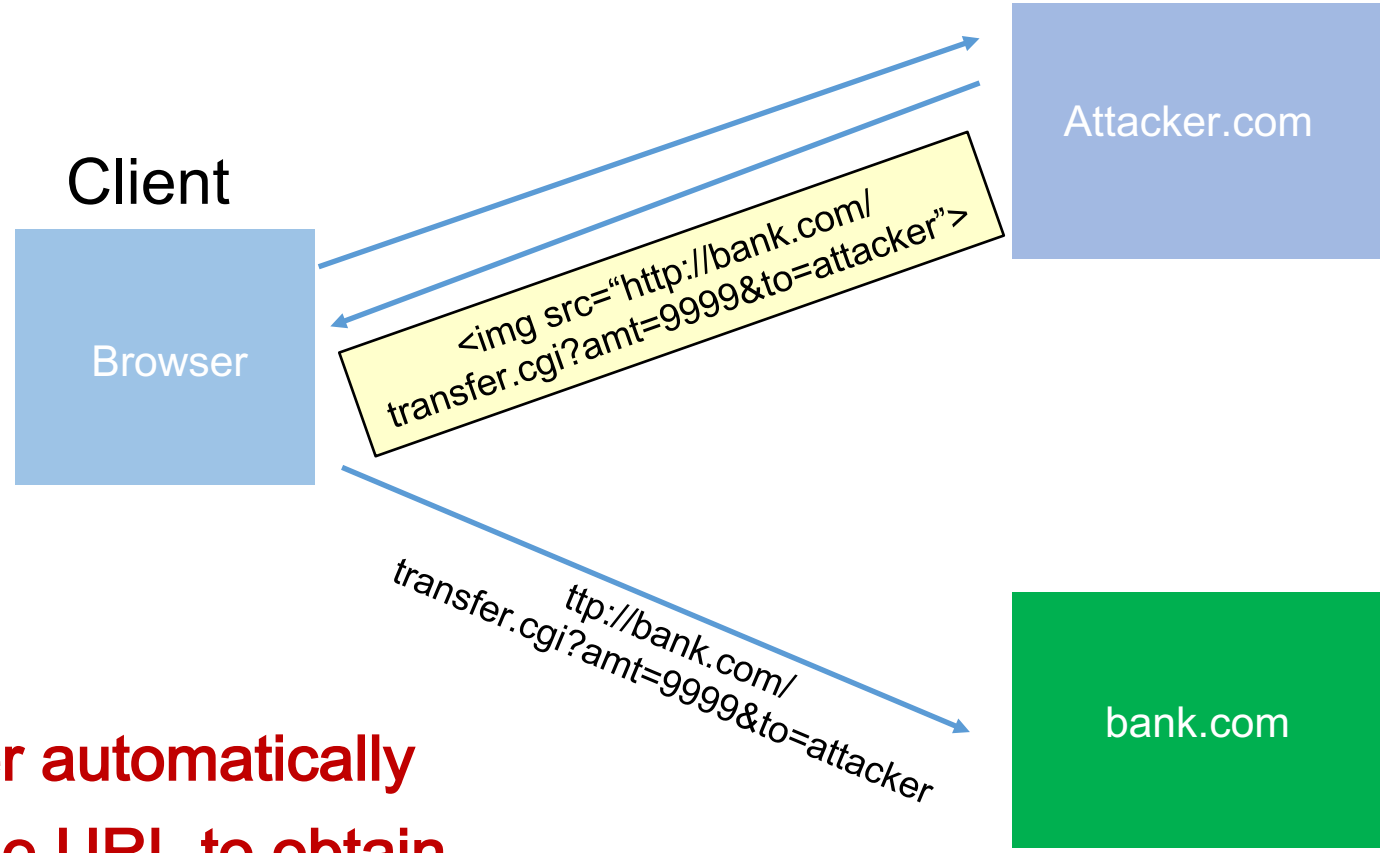


Exploiting URLs with side-effects

Exploiting URLs with side-effects

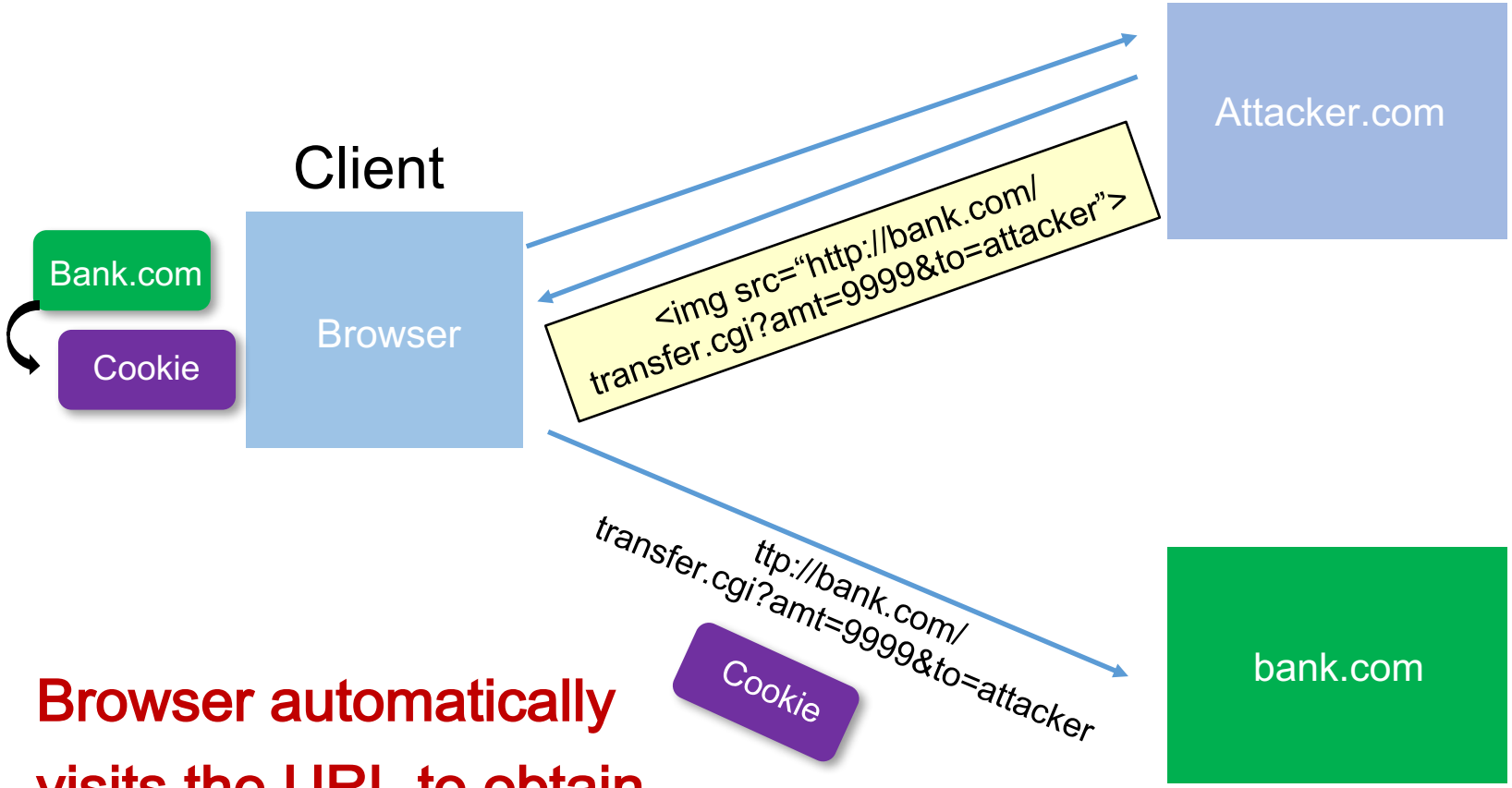
Browser automatically visits the URL to obtain what it believes will be an image

Exploiting URLs with side-effects



Browser automatically visits the URL to obtain what it believes will be an image

Exploiting URLs with side-effects



Browser automatically visits the URL to obtain what it believes will be an image

Cross-Site Request Forger

- Target: User who has an account on a vulnerable server
- Attack goal: make requests to the server via the user's browser that look to the server like the user intended to make them
- Attacker tools: ability to get the user to "click a link" crafted by the attacker that goes to the vulnerable site
- Key tricks:
  - Requests to the web server have predictable structure
  - Use of something like <img src=…> to force the victim to send it

CSRF protections: REFERER

- The browser will set the **REFERER** field to the page that hosted a clicked link

HTTP Headers

```
http://www.zdnet.com/worst-ddos-attack-of-all-time-hits-french-site-7000026330/

GET /worst-ddos-attack-of-all-time-hits-french-site-7000026330/ HTTP/1.1
Host: www.zdnet.com
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
```

- **Trust requests from pages a user could legitimately reach**
  - From good users, if referrer header present, generally trusted
  - Defends against session hijacks too