CMSC414 Computer and Network Security

Introduction to Web and SQL Injection

Yizheng Chen | University of Maryland
surrealyz.github.io

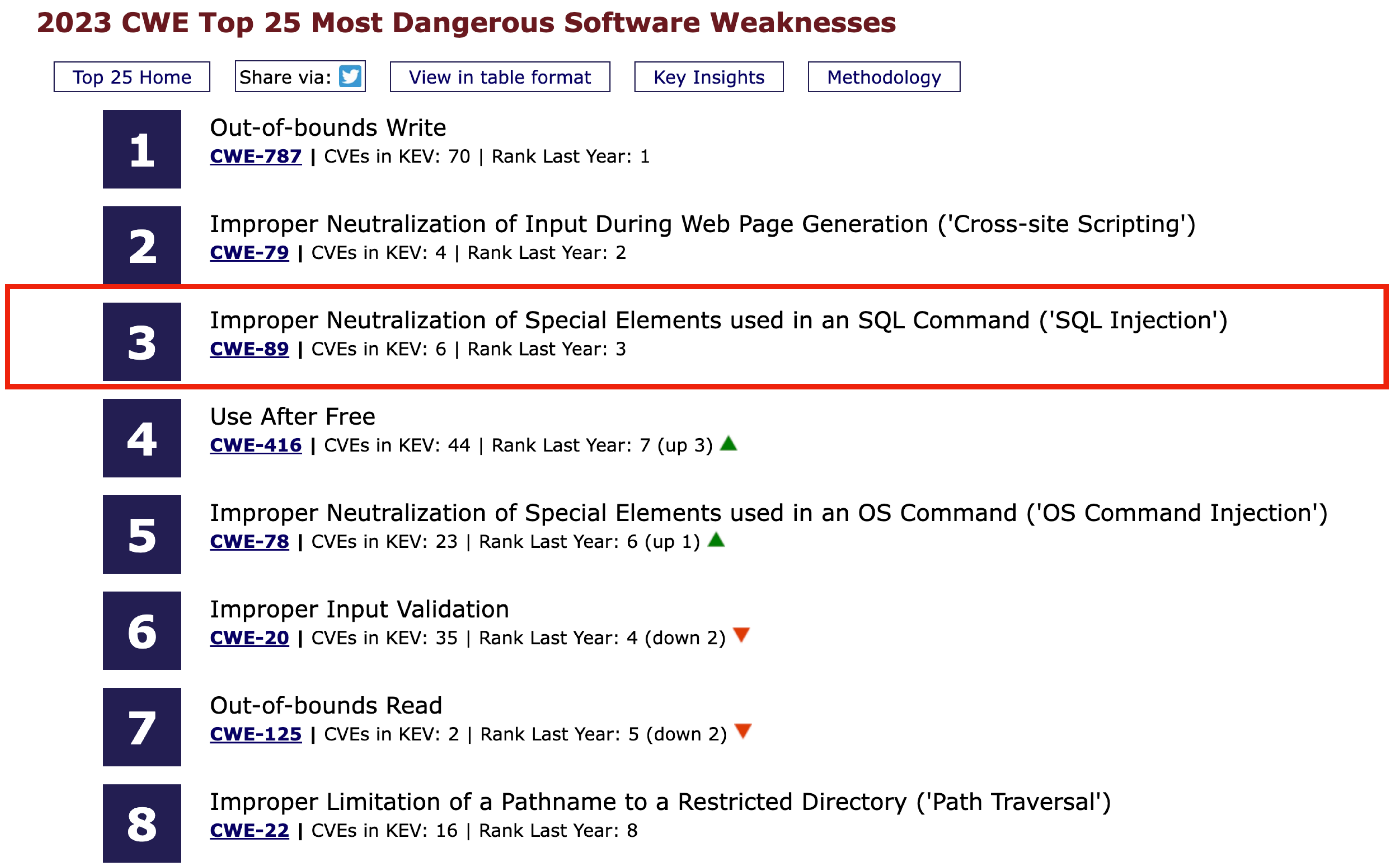
Feb 13, 2024

Announcements

- Project 1 Deadline extended to 11:59pm, Tuesday, Feb 20
- If you still haven’t set up gitlab, it is very late now!

Agenda

- SQL Injection
- Introduction to Web



2023 CWE Top 25 Most Dangerous Software Weaknesses

1 Out-of-bounds Write
CWE-787 | CVEs in KEV: 70 | Rank Last Year: 1

2 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
CWE-79 | CVEs in KEV: 4 | Rank Last Year: 2

3 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
CWE-89 | CVEs in KEV: 6 | Rank Last Year: 3

4 Use After Free
CWE-416 | CVEs in KEV: 44 | Rank Last Year: 7 (up 3) ▲

5 Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
CWE-78 | CVEs in KEV: 23 | Rank Last Year: 6 (up 1) ▲

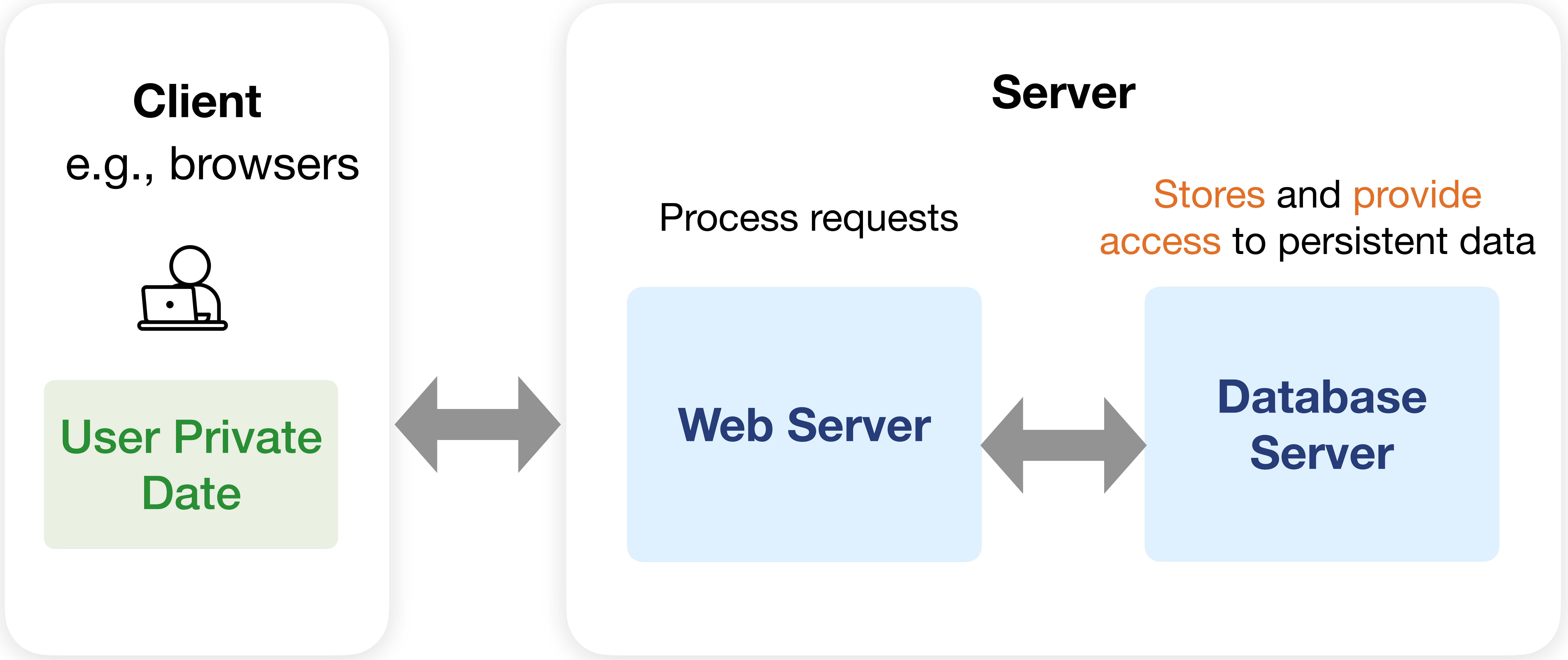
6 Improper Input Validation
CWE-20 | CVEs in KEV: 35 | Rank Last Year: 4 (down 2) ▼

7 Out-of-bounds Read
CWE-125 | CVEs in KEV: 2 | Rank Last Year: 5 (down 2) ▼

8 Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
CWE-22 | CVEs in KEV: 16 | Rank Last Year: 8

https://cwe.mitre.org/top25/archive/2023/2023_top25_list.html

A Very Basic Web Architecture



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

Database Transactions

- A transaction is a unit of work in a database (may contain multiple reads and writes, e.g., read an entry and update some fields)
- Good database servers are **ACID**
  - **A**tomicity: Transactions complete entirely or not at all
  - **C**onsistency: The database is always in a *valid* state (but not necessarily *correct*)
  - **I**solation: Results from a transaction aren’t visible until it is complete
  - **D**urability: Once a transaction is committed, it remains, despite, e.g., power failures

TOCTOU

- **Time-of-check to time-of-use** vulnerability
    - Check: no problem
    - Use: has problem
- Race condition
    - e.g., Reading in a state where other writes are in progress, or writing some partial content before finishing, and then another transaction reads

SQL Databases

- SQL: Structured Query Language
  - Create and query data
- A database has some tables
- A table has a predefined structure

SQL Database Example

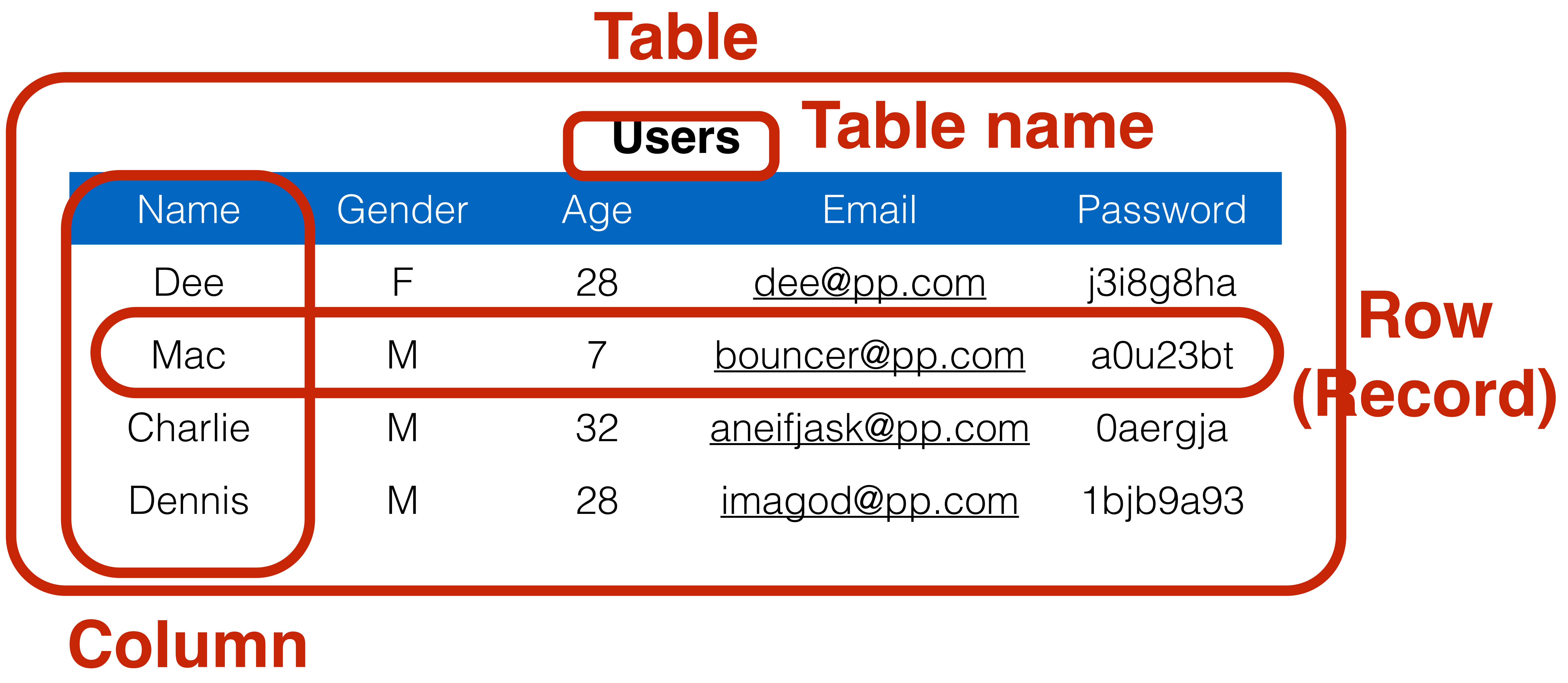


Table
Table name
Row (Record)
Column

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) Example

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

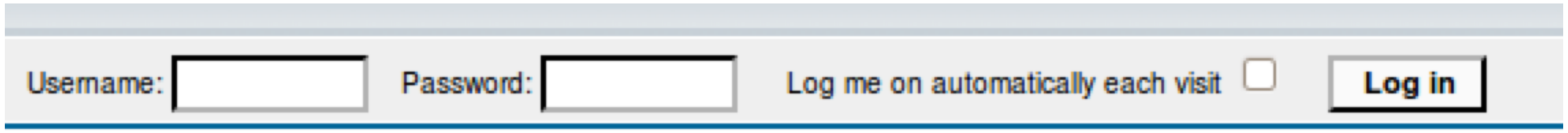
```
SELECT Age FROM Users WHERE Name='Dee';      28
SELECT Age FROM Users WHERE Name='Dee' OR Name='Mac'; 28, 7
UPDATE Users SET email='readgood@pp.com'
   WHERE Age=32; -- this is a comment
INSERT INTO Users Values('Frank', 'M', 57, ...);
DROP TABLE Users;
```

Some SQL Syntax

- SELECT * FROM table
  - The asterisk (*) is shorthand for “all columns.” Select all columns from the table, keeping all rows.
- WHERE can be used to filter out certain rows
  - Arithmetic comparison: <, <=, >, >=, =, <>
  - Arithmetic operators: +, - , * , /
  - Boolean operators: AND, OR
  - AND has precedence over OR

Server-side code

Website



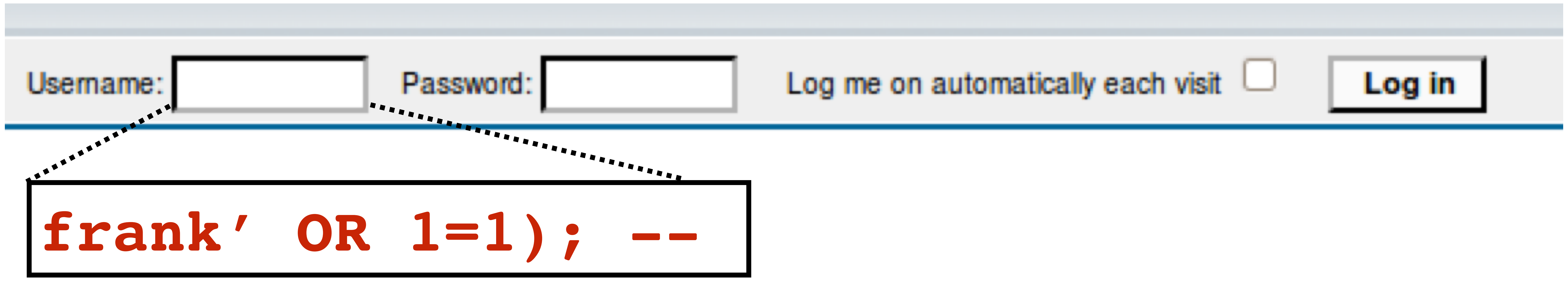
"Login code" (php)

```
$result = mysql_query("select * from Users
       where(name='$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

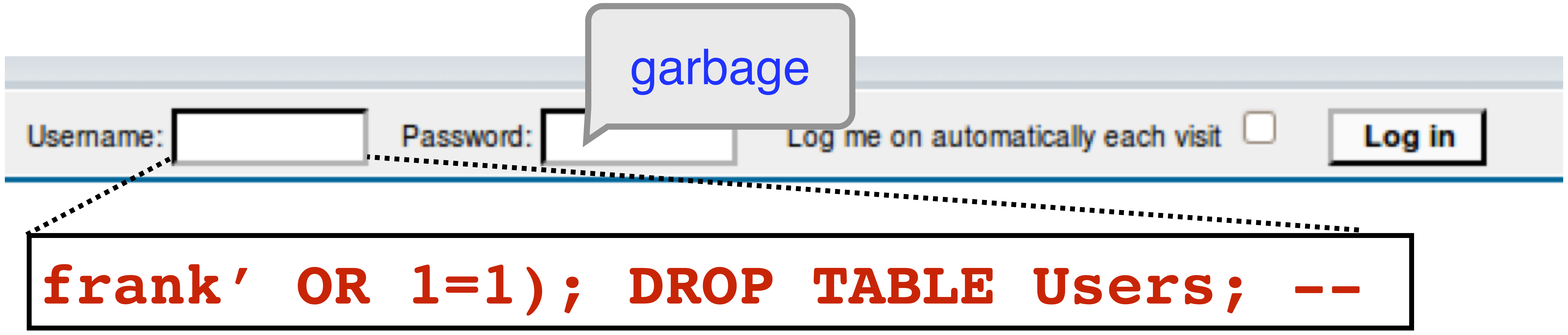


```
frank' OR 1=1); --
```

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

```
$result = mysql_query("select * from Users where
(name='frank' OR 1=1); -- ' and password='x');");
```

SQL injection

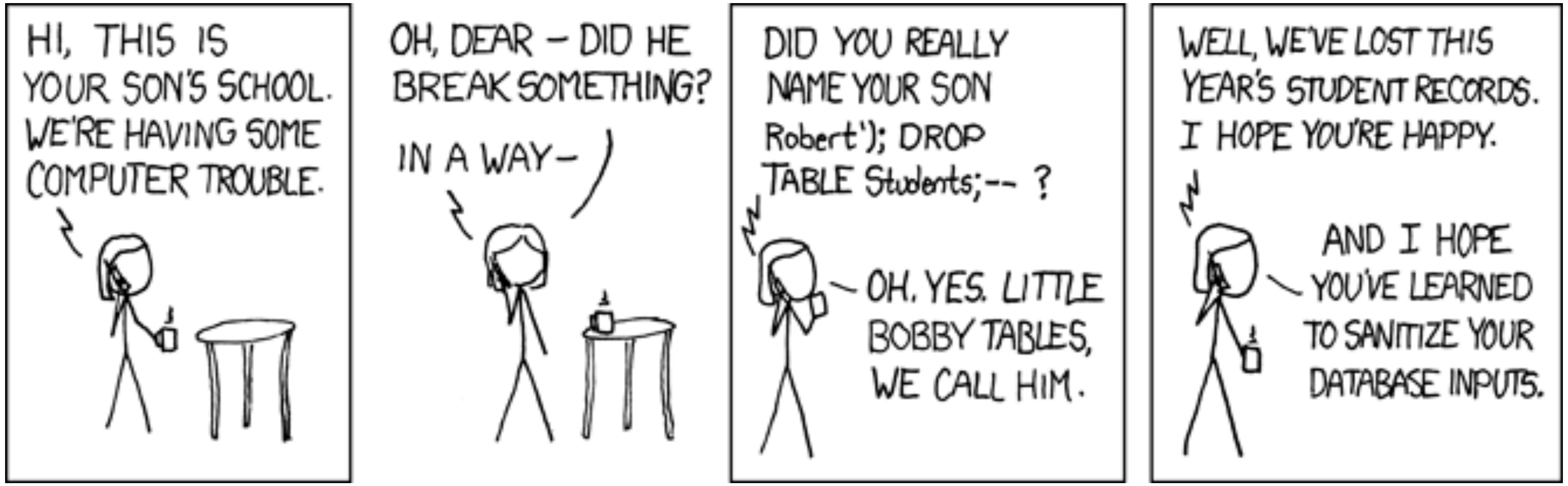


```
frank’ OR 1=1); DROP TABLE Users; --
```

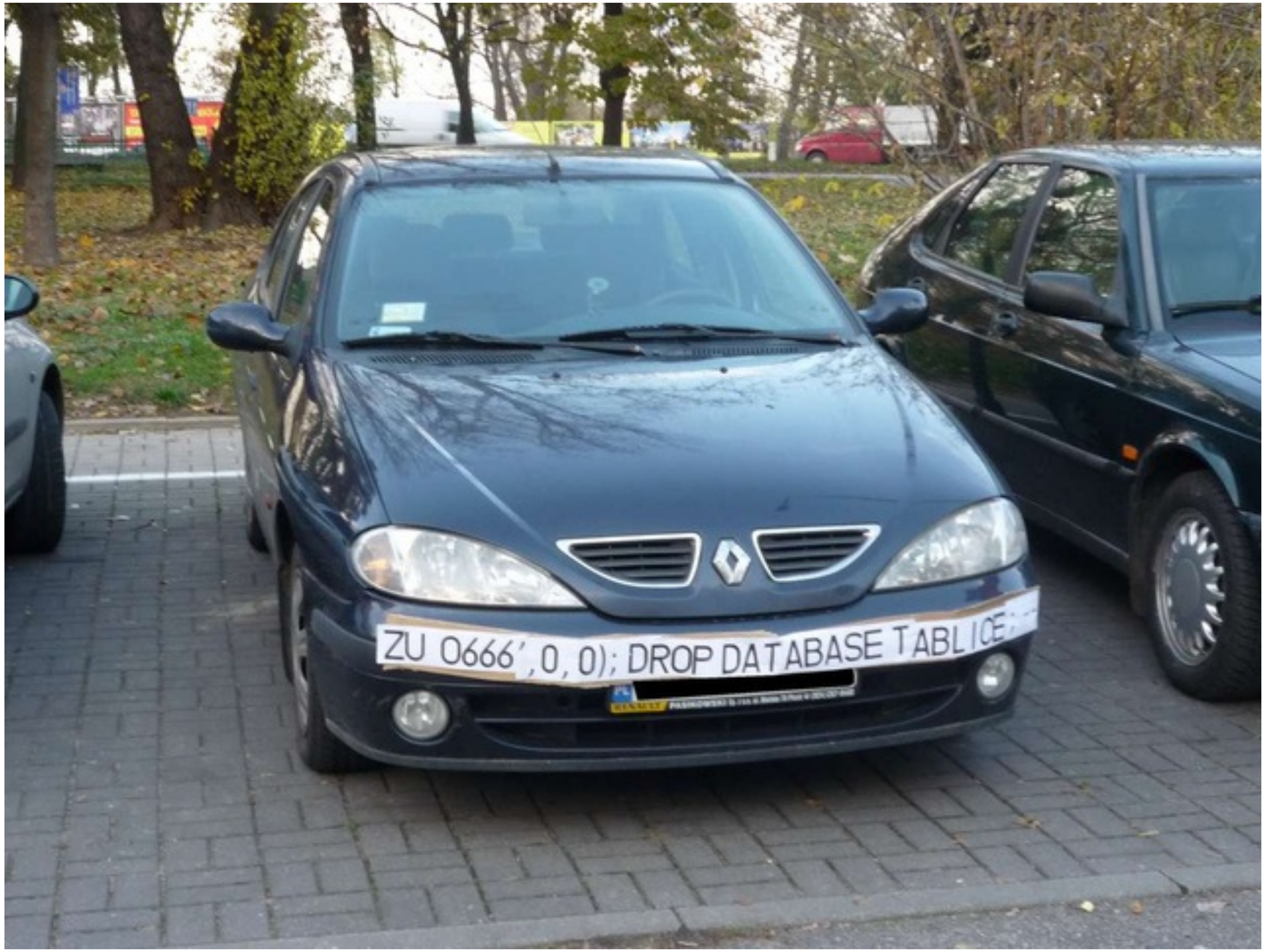
```
$result = mysql_query(“select * from Users
       where(name=‘$user’ and password=‘$pass’);”);

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

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

Exploits of a Mom


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.

https://www.explainxkcd.com/wiki/index.php/327:_Exploits_of_a_Mom



A “Licence plate” with an SQL injection attack as a way to fight back traffic cameras.
https://www.reddit.com/r/geek/comments/1j9tn3/speed_camera_sql_injection/

SQL Injection Defense: Input Sanitization

- Block special characters: ’ -- ;
- Allow: input within range, e.g., integer values for some fields
- Escape special characters: \; \’
    - Escape the escape? \\

SQL Injection Defense: Input Sanitization

- Block special characters: ’ -- ;
- Allow: input within range, e.g., integer values for some fields
- Escape special characters: \; \’
    - Escape the escape? \\
- Secure escaper exists in SQL libraries
- May not be an effective solution, if we run SQL queries with raw user inputs

What else can we do?

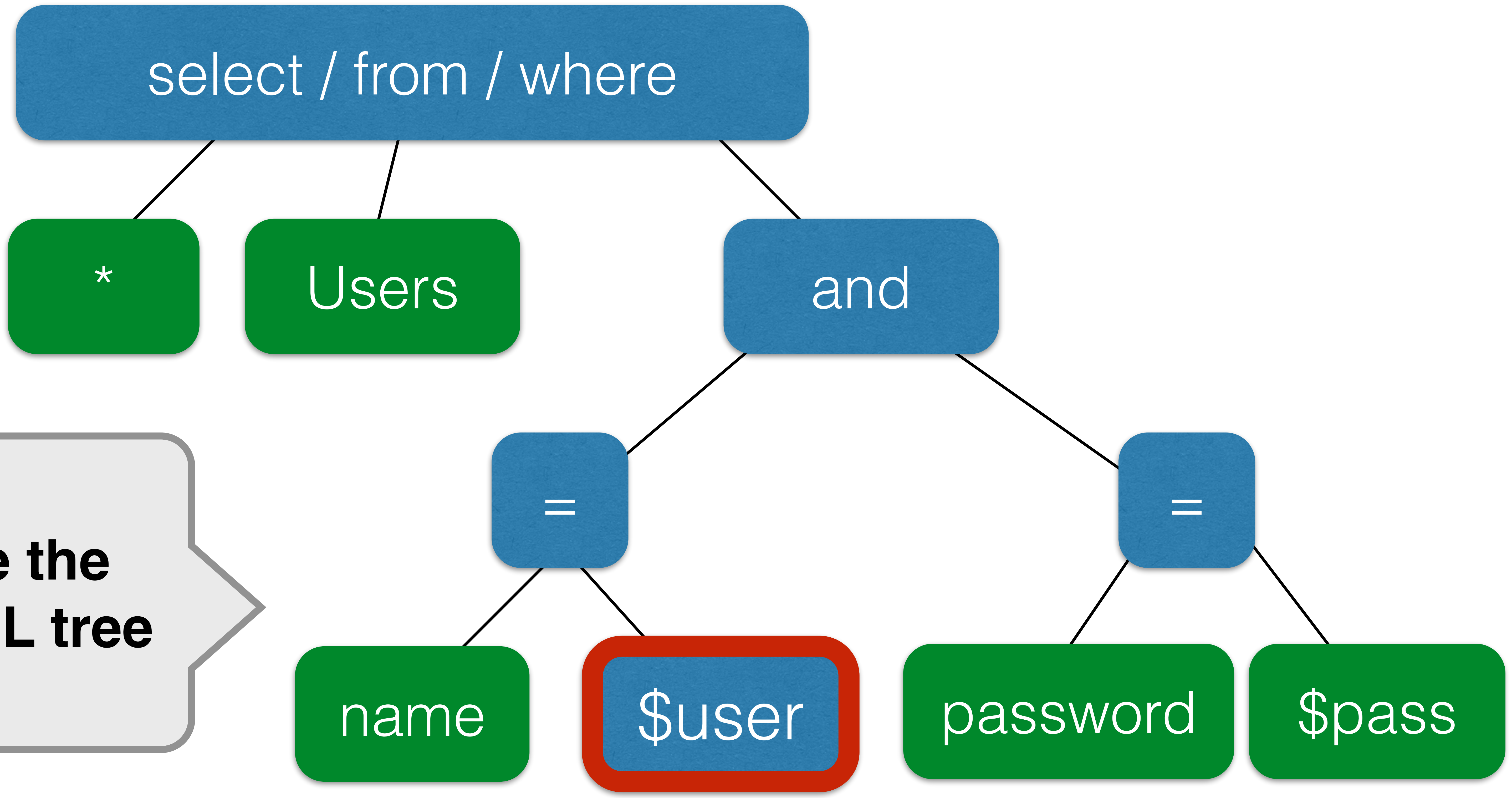
- Hint: data vs code
- User input, SQL queries
- Similar problem structure as buffer overflow problem:
    - User input, instruction

Parameterized SQL / Prepared Statements

- Idea: Parse the SQL query structure first, then insert the data
- Use a question mark (?) for data when writing SQL statements
- When the parser encounters the ?, it fixes it as a single node in the syntax tree
- After parsing, only then, it inserts data
- The untrusted input cannot change the SQL query structure

Example without Prepared Statements

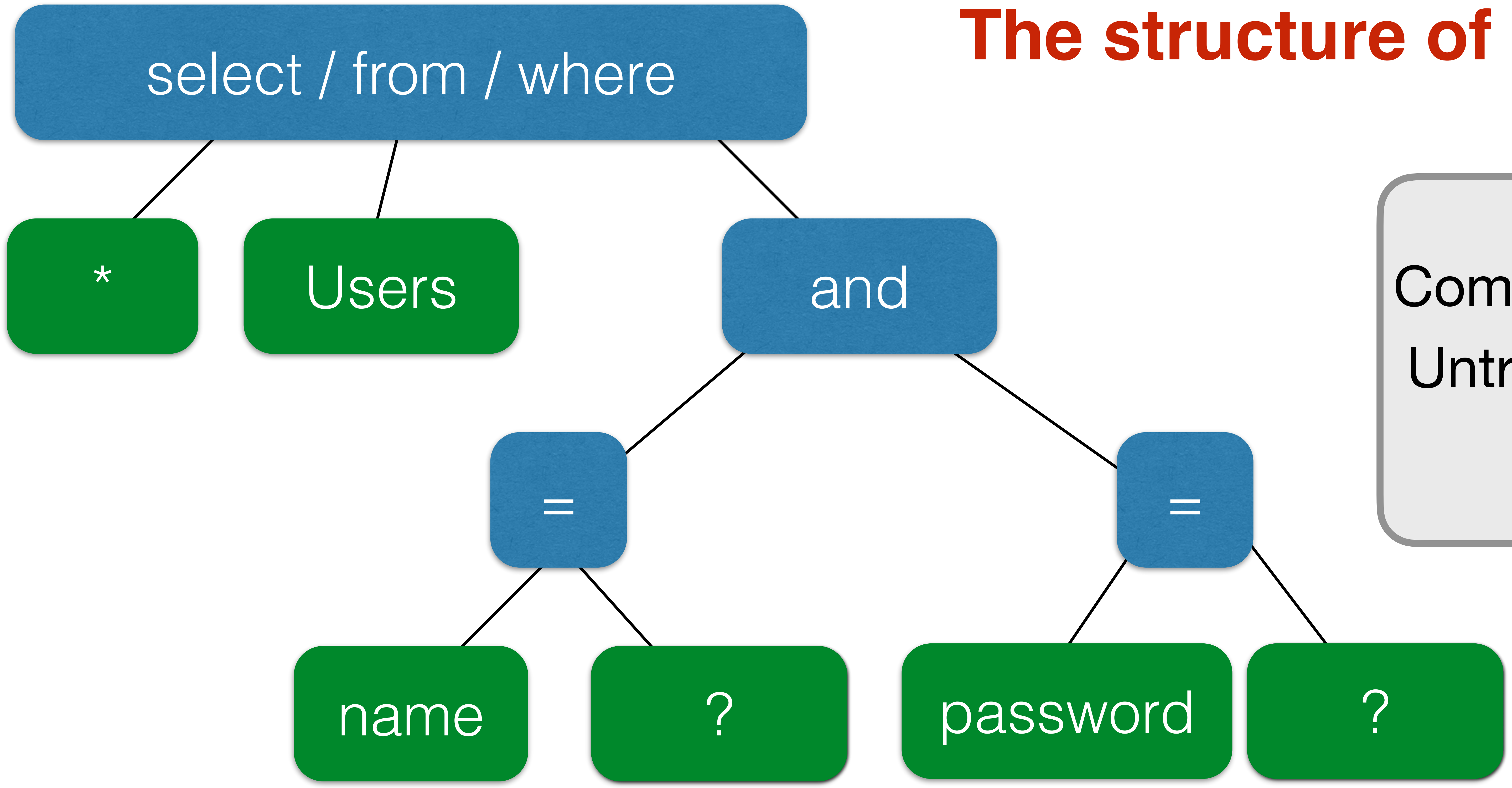
```
$result = mysql_query("select * from Users
         where(name='$user' and password='$pass');");
```



$user can change the structure of the SQL tree

Prepared Statement Example

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



The structure of the tree is *fixed*

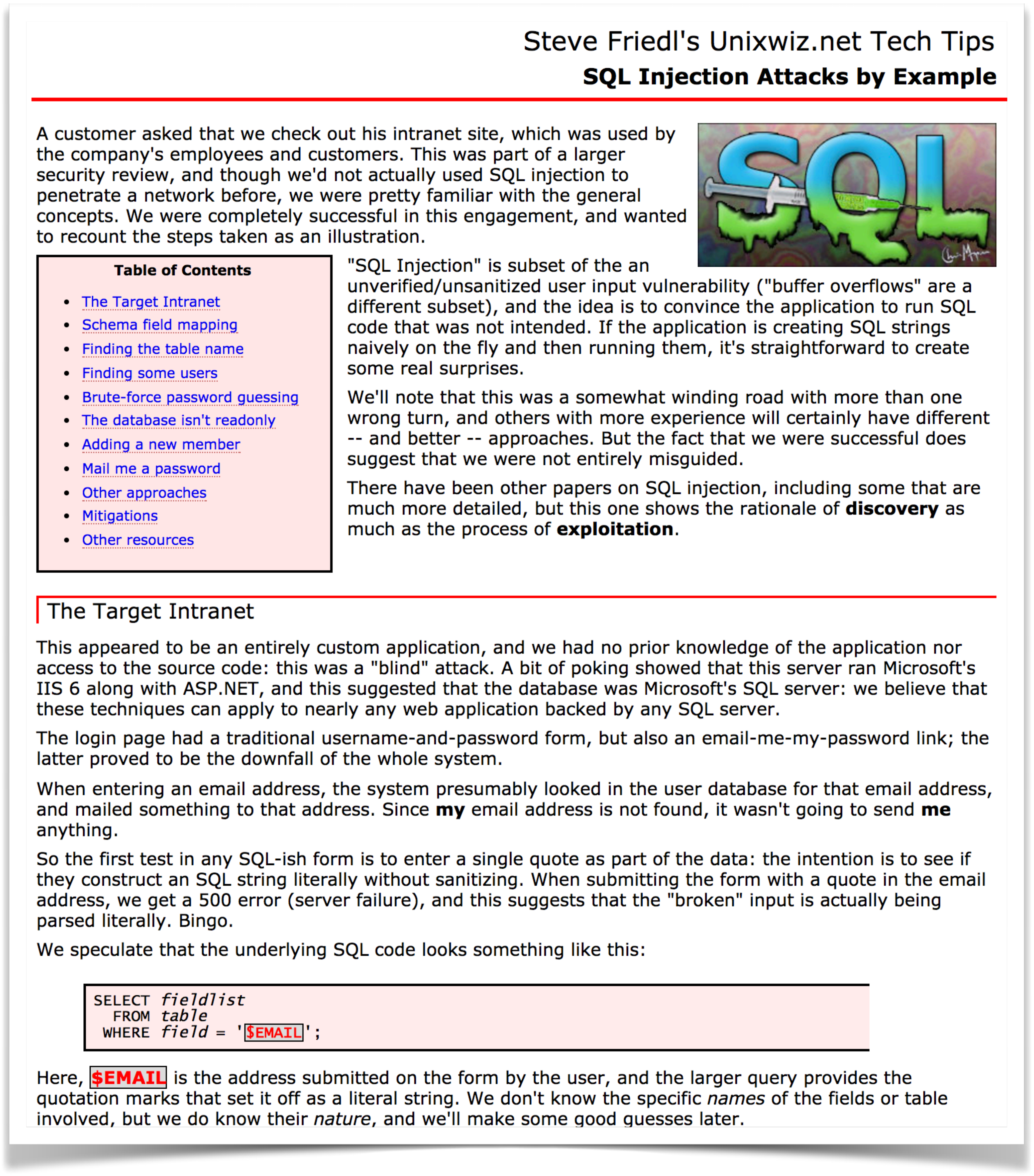
Compile first, bind data later

Untrusted input will only be treated as data

Mitigate the Impact of Attacks

- Least privilege
  - Limit commands and tables a user can access
- Encrypt sensitive data in the SQL table

Followup Reading



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.

Table of Contents

"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';
```

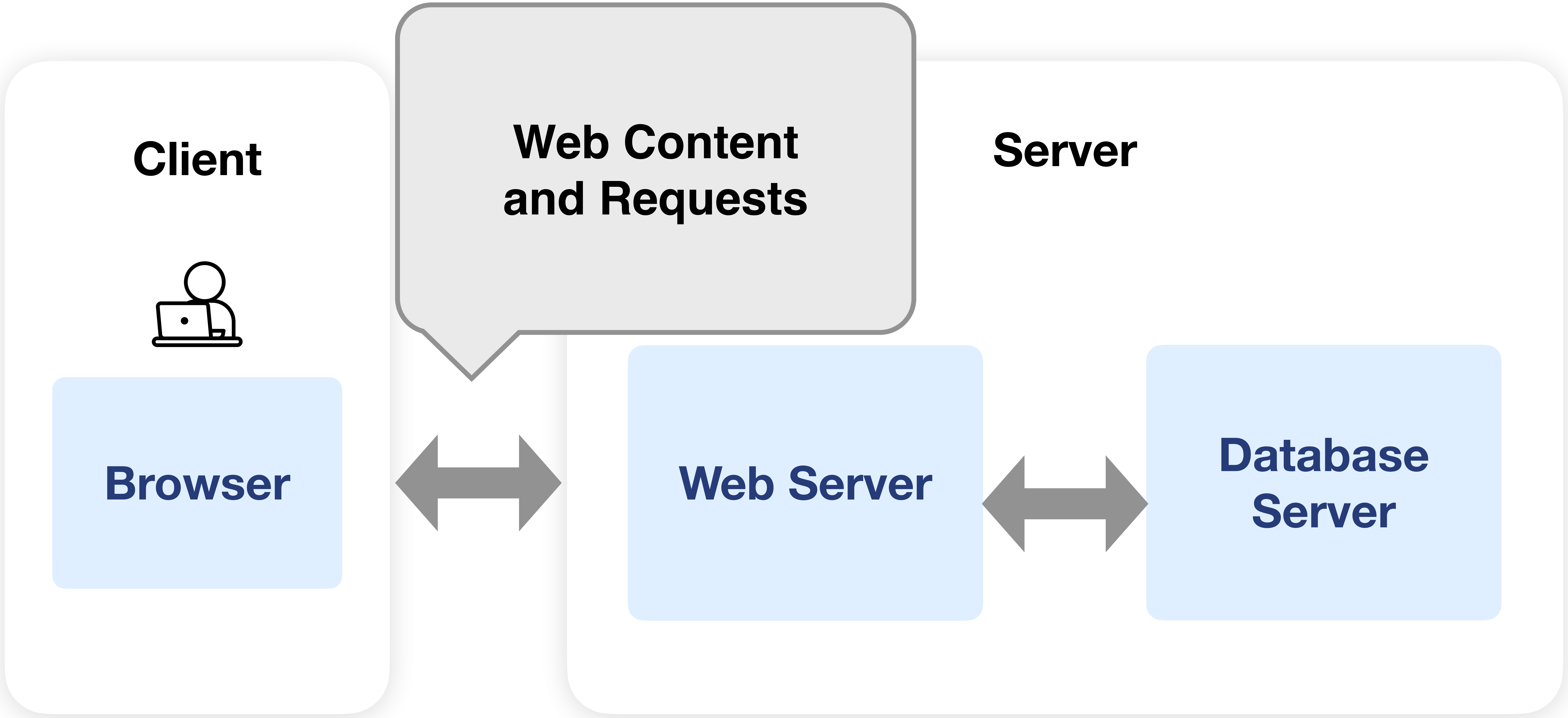
Here, **$EMAIL** is the address submitted on the form by the user, and the larger query provides the quotation marks that set it off as a literal string. We don't know the specific *names* of the fields or table involved, but we do know their *nature*, and we'll make some good guesses later.

http://www.unixwiz.net/techtips/sql-injection.html

Agenda

- SQL Injection
- Introduction to Web

A Very Basic Web Architecture



URL

Every resource (webpage, image, PDF, etc.) on the web is identified by a URL (Uniform Resource Locator).
http://www.example.com/index.html

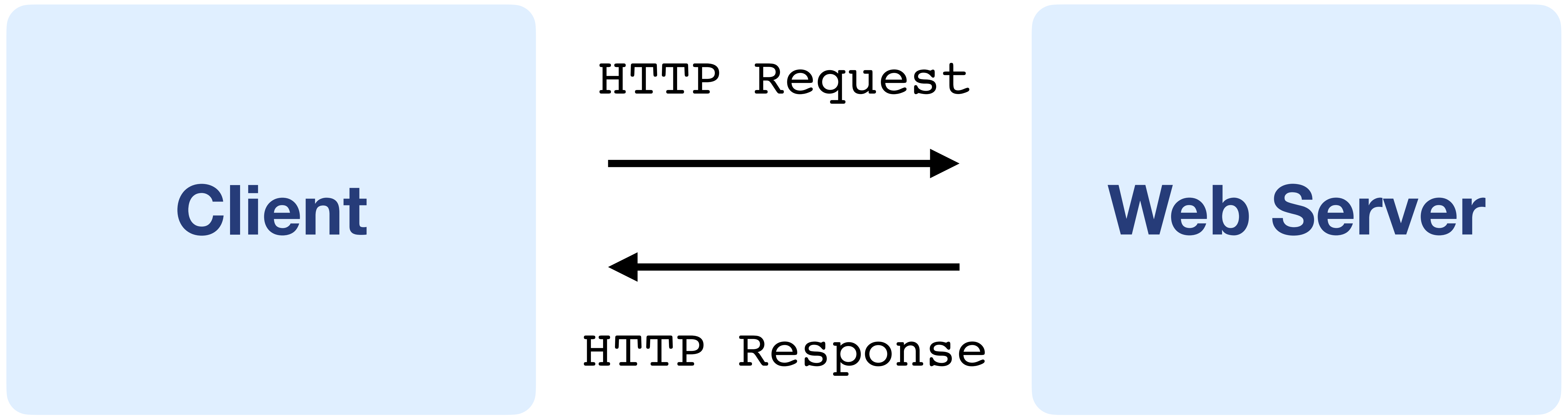
- Protocol: http, https, git+ssh, ftp
  - HyperText Transfer Protocol (HTTP): An “application-layer” protocol for exchanging collections of data
- Location: www.example.com
  - Web server domain name, IP address
- Path: /index.html

URL

Every resource (webpage, image, PDF, etc.) on the web is identified by a URL (Uniform Resource Locator).
http://alice@www.example.com:414/index.html?param1=val1&param2=val2#anchor

- Username: alice
- Port: 414
  - Default HTTP port: 80, default HTTPS port: 443
- URL arguments: key value pairs ?param1=val1&param2=val2
- Anchor: scroll to a certain part of the webpage #anchor

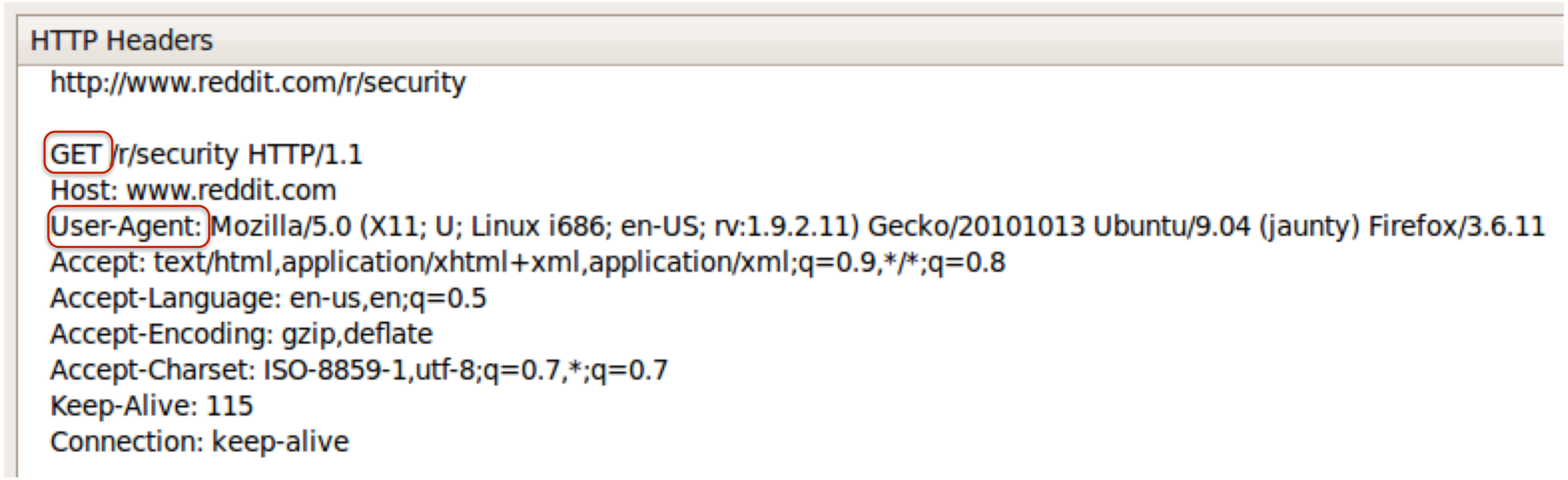
HTTP: Request-Response Model



- 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)

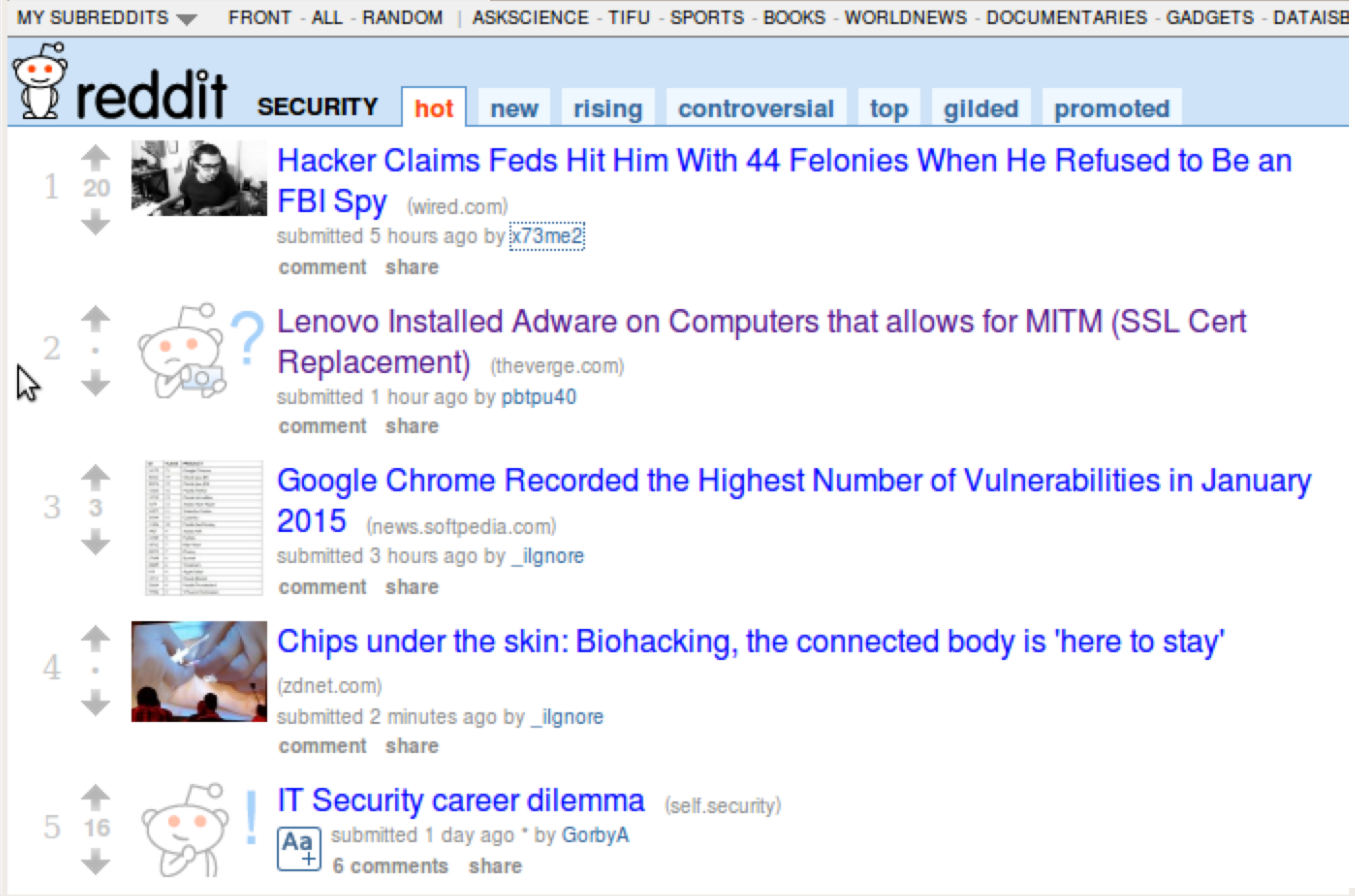
HTTP GET requests

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



```
http://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 - DATAISB

reddit SECURITY hot new rising controversial top gilded promoted

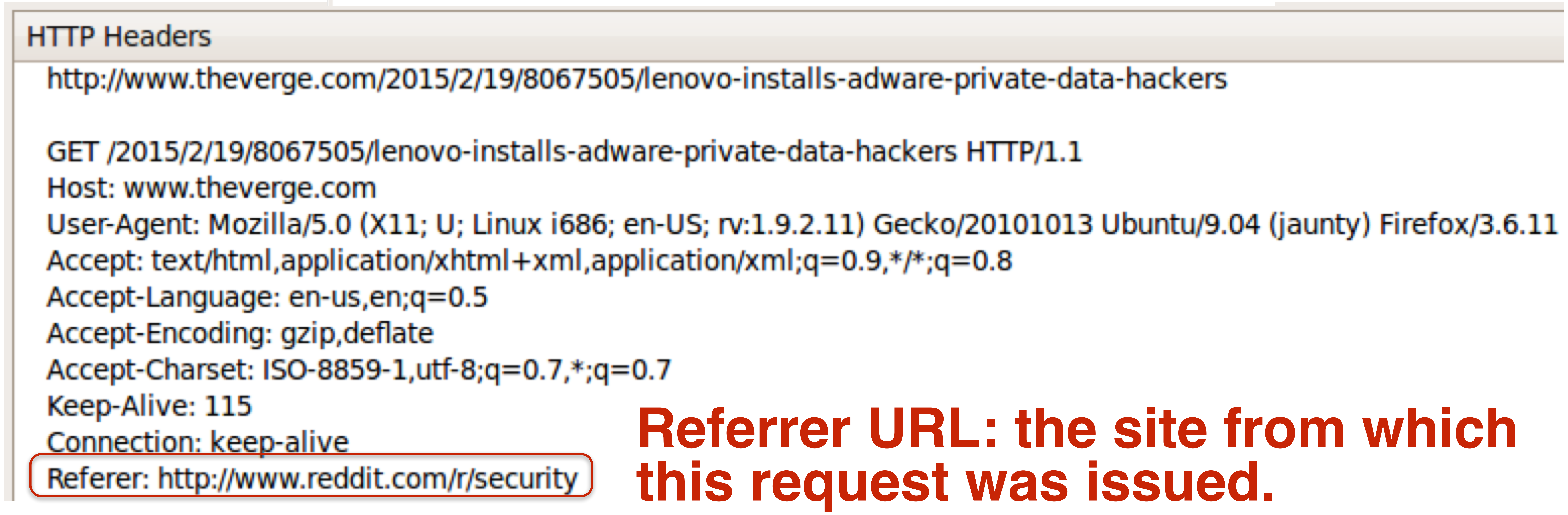
1 20 Hacker Claims Feds Hit Him With 44 Felonies When He Refused to Be an FBI Spy (wired.com)
submitted 5 hours ago by x73me2
comment share

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 _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

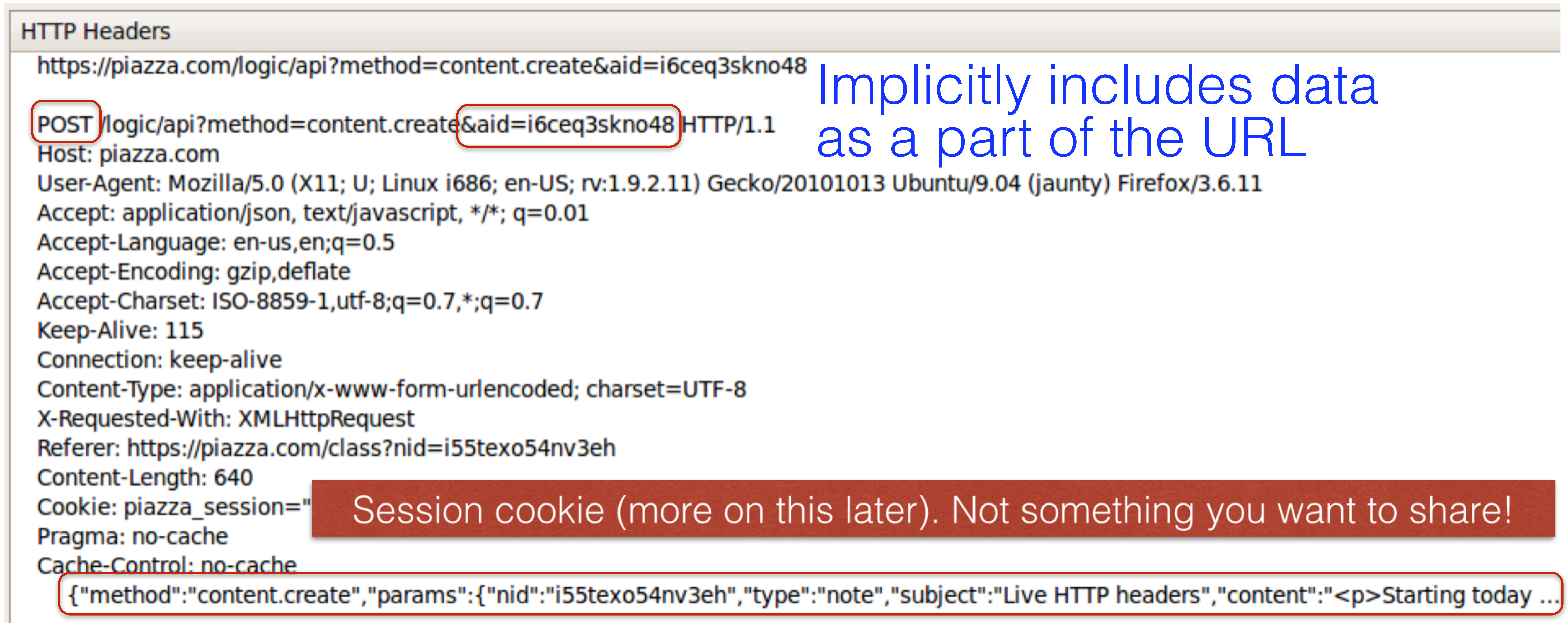

HTTP Headers

```
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



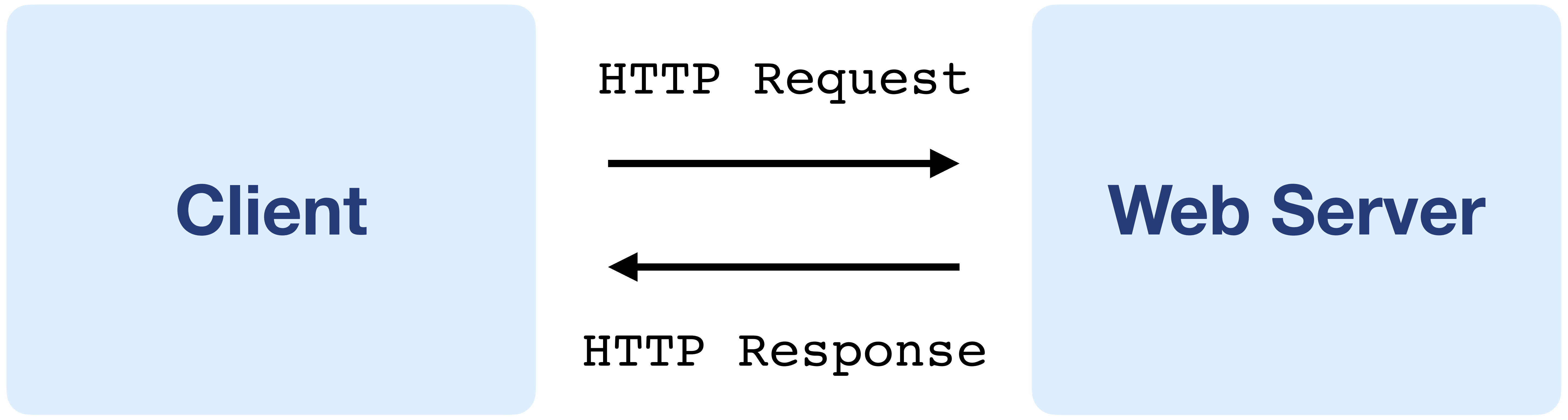
```
HTTP Headers
https://piazza.com/logic/api?method=content.create&aid=i6ceq3skno48

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 ...
```

Implicitly includes data as a part of the URL

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

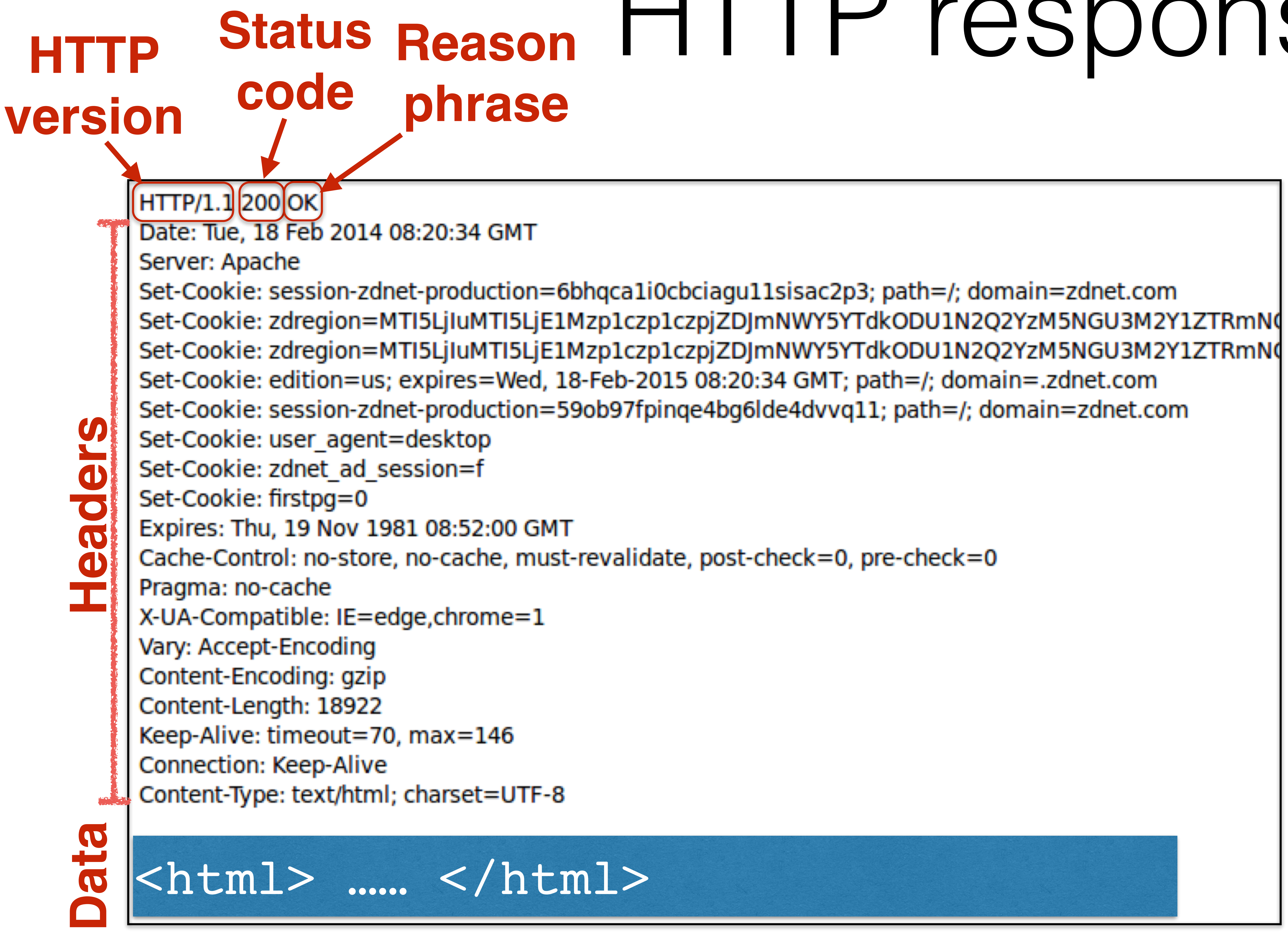
Explicitly includes data as a part of the request's content

HTTP: Request-Response Model



- 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>
```

Elements of a Webpage

- HTML
  - Create a link to Google: <a href="http://google.com">Click me</a>
  - Embed a picture in the webpage: <img src=“http://example.com/picture.png">
  - Include JavaScript in the webpage: <script>alert(1)</script> Security risk!
  - Embed another webpage: <iframe src=“http://example.org”></iframe> Security risk!
- CSS
  - CSS (Cascading Style Sheets) lets us modify the appearance of an HTML page

Elements of a Webpage

- JavaScript
  - Assume JavaScript can arbitrarily modify any HTML or CSS on a webpage
  - Security risk