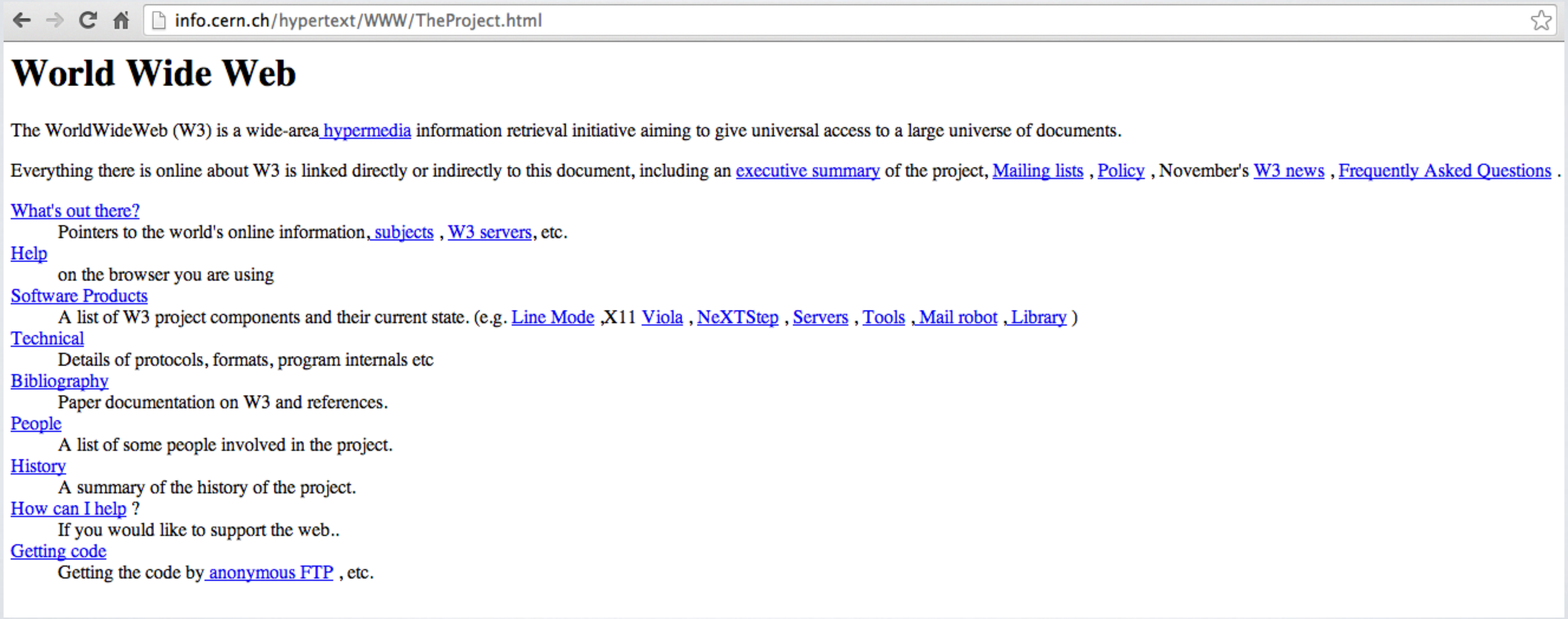
Web Insecurity

Kc Udonsi

1991

Sir Tim Berners-Lee



World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
  Pointers to the world's online information, subjects , W3 servers, etc.
Help
  on the browser you are using
Software Products
  A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)
Technical
  Details of protocols, formats, program internals etc
Bibliography
  Paper documentation on W3 and references.
People
  A list of some people involved in the project.
History
  A summary of the history of the project.
How can I help ?
  If you would like to support the web..
Getting code
  Getting the code by anonymous FTP , etc.

2014

Web Portals

Customer Resources Management

E-Health

Accounting and Billing

E-Learning

Collaboration

Content Management

Social Networks

Publishing

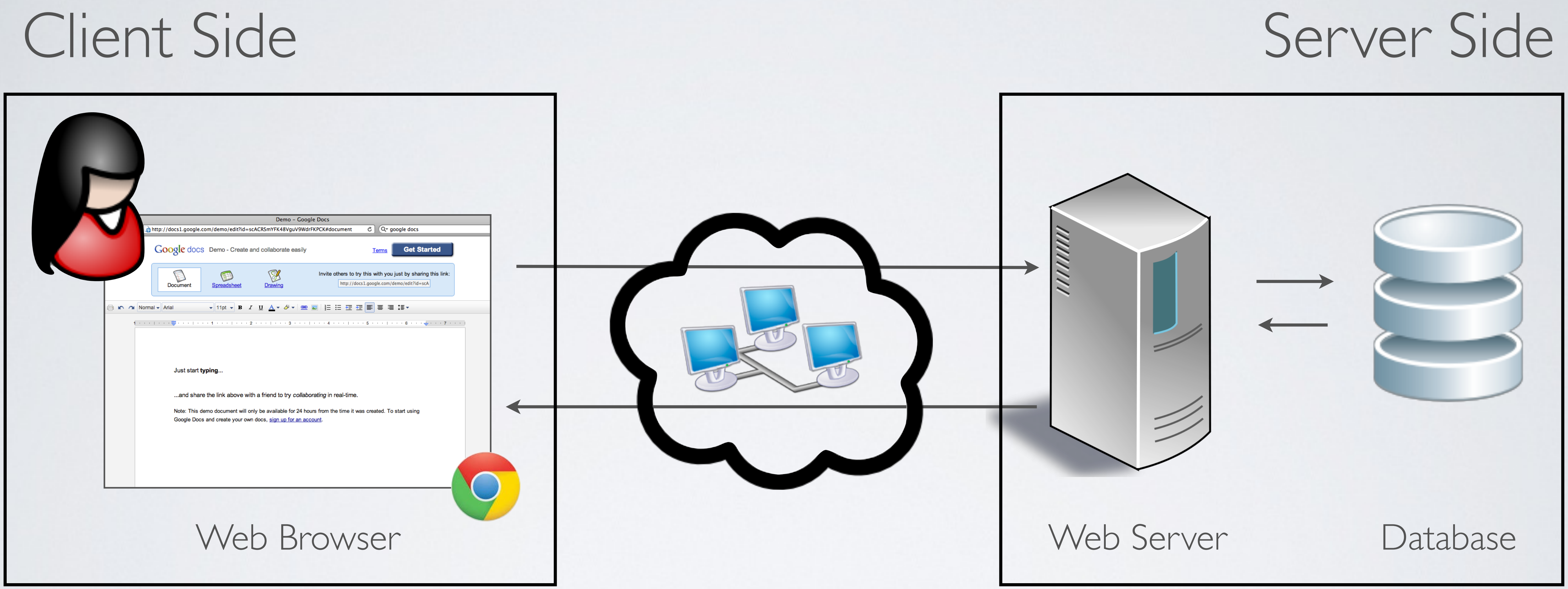
How many of us have ...

- A locally managed web-site
- Designed or built a web application
- Built a web application featuring:
    - Authentication
    - Authorization
    - Multiple backend components / modules
    - Data input & upload

Web application insecurity …

The Big Picture

The web architecture


Client Side
Server Side
Web Browser
Web Server
Database
Demo – Google Docs
http://docs1.google.com/demo/edit?id=scACRSmYFK48VguV9WdrFKPCK#document
google docs
Google docs
Demo - Create and collaborate easily
Terms
Get Started
Document
Spreadsheet
Drawing
Invite others to try this with you just by sharing this link:
http://docs1.google.com/demo/edit?id=scA
Normal
Arial
11pt

Just start **typing**...

...and share the link above with a friend to try *collaborating* in real-time.

Note: This demo document will only be available for 24 hours from the time it was created. To start using Google Docs and create your own docs, sign up for an account.

Securing the web architecture means securing ...

- The network
- The DNS (Domain Name System)
- The web server operating system
- The web server application (*Apache* for instance)
- The database application (*Oracle* for instance)
- The web user
- The web application

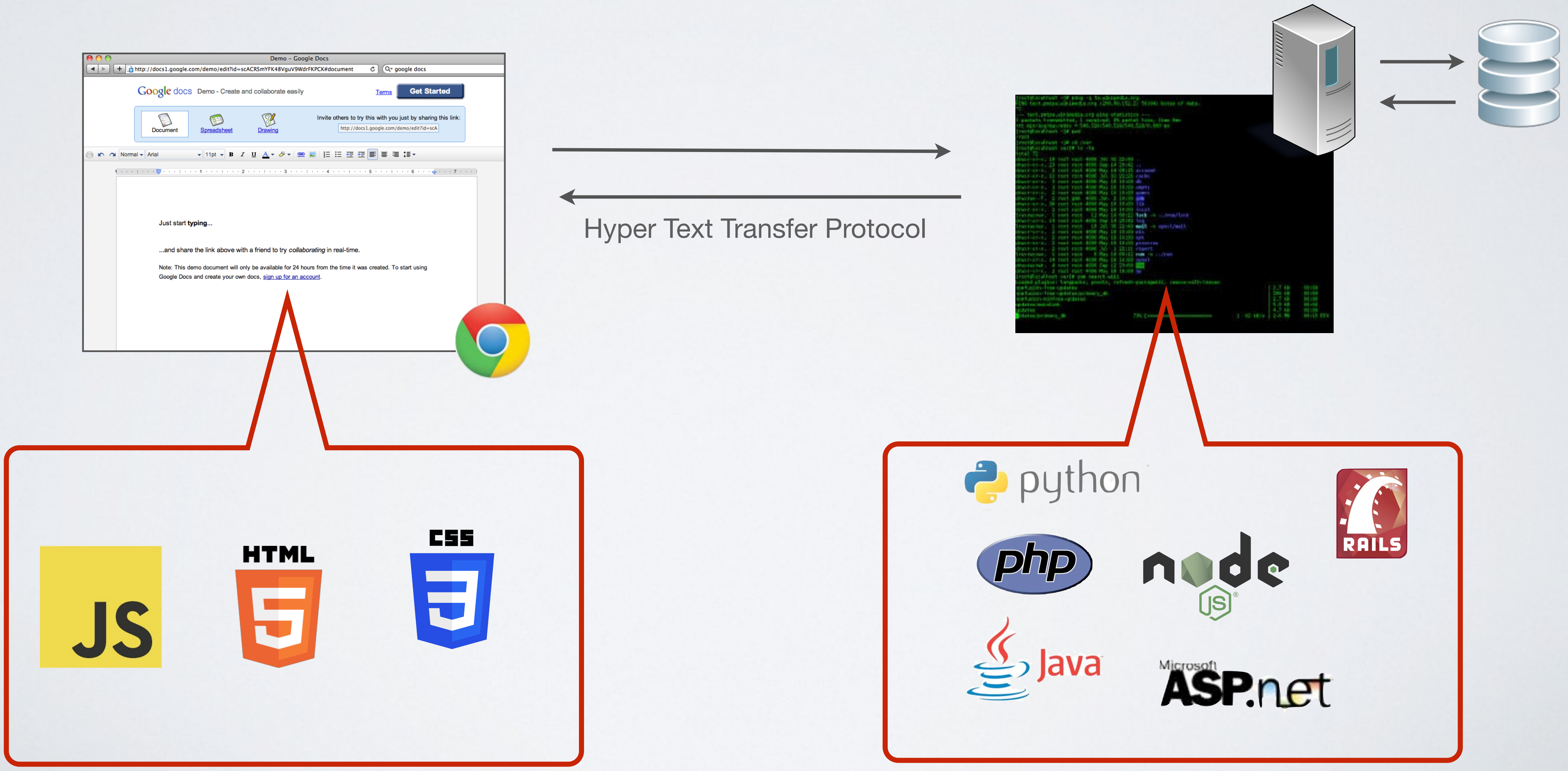
Our focus here!

What is a web application?

program running on the browser

+

program running on the server



Invite others to try this with you just by sharing this link:
http://docs1.google.com/demo/edit?id=scA

Just start **typing**...

...and share the link above with a friend to try *collaborating* in real-time.

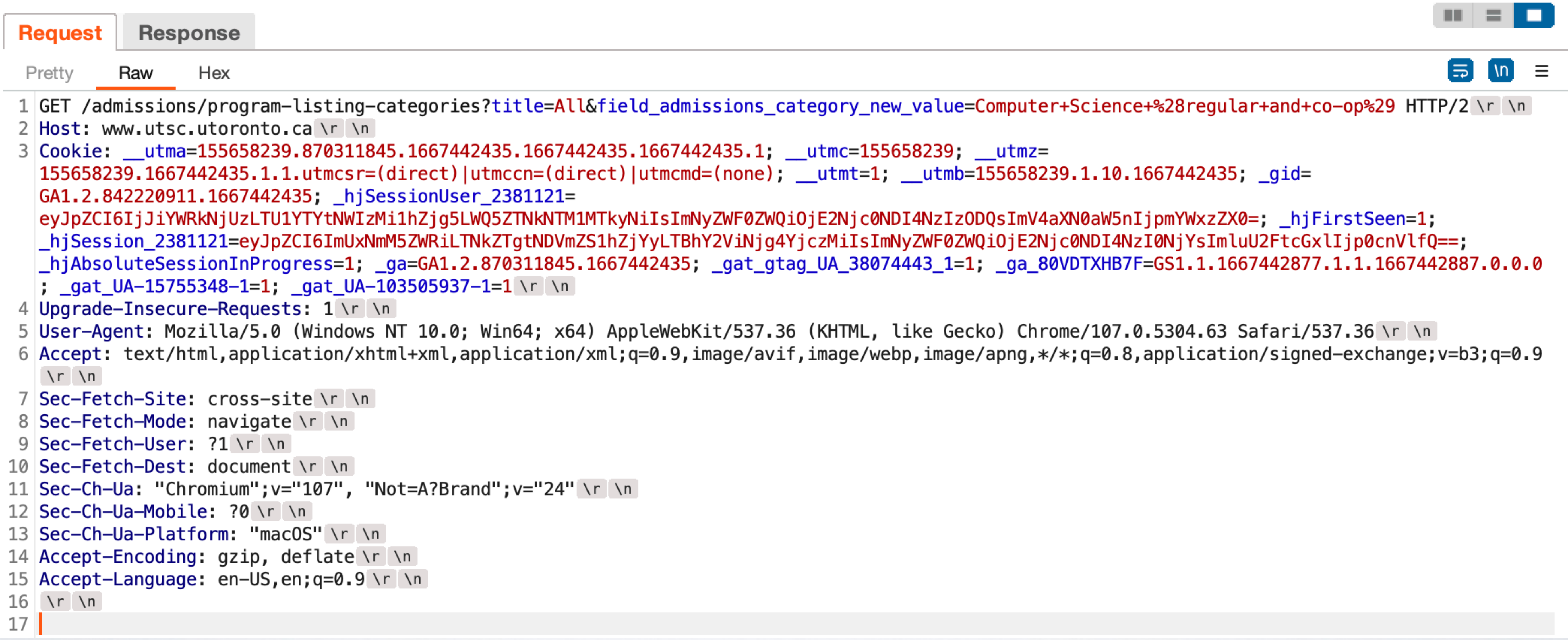
Note: This demo document will only be available for 24 hours from the time it was created. To start using Google Docs and create your own docs, sign up for an account.

The HTTP protocol

Stateless application layer protocol for requesting/receiving data on the Web

- Standard TCP protocol on **port 80** (by default)
- **URI/URL** specifies what resource is being accessed
- Different **request methods**
- Evolution: … HTTP/1.1, HTTP/2.0, HTTP/3.0
- Clients are also called "User-agents"

The HTTP protocol: Requests

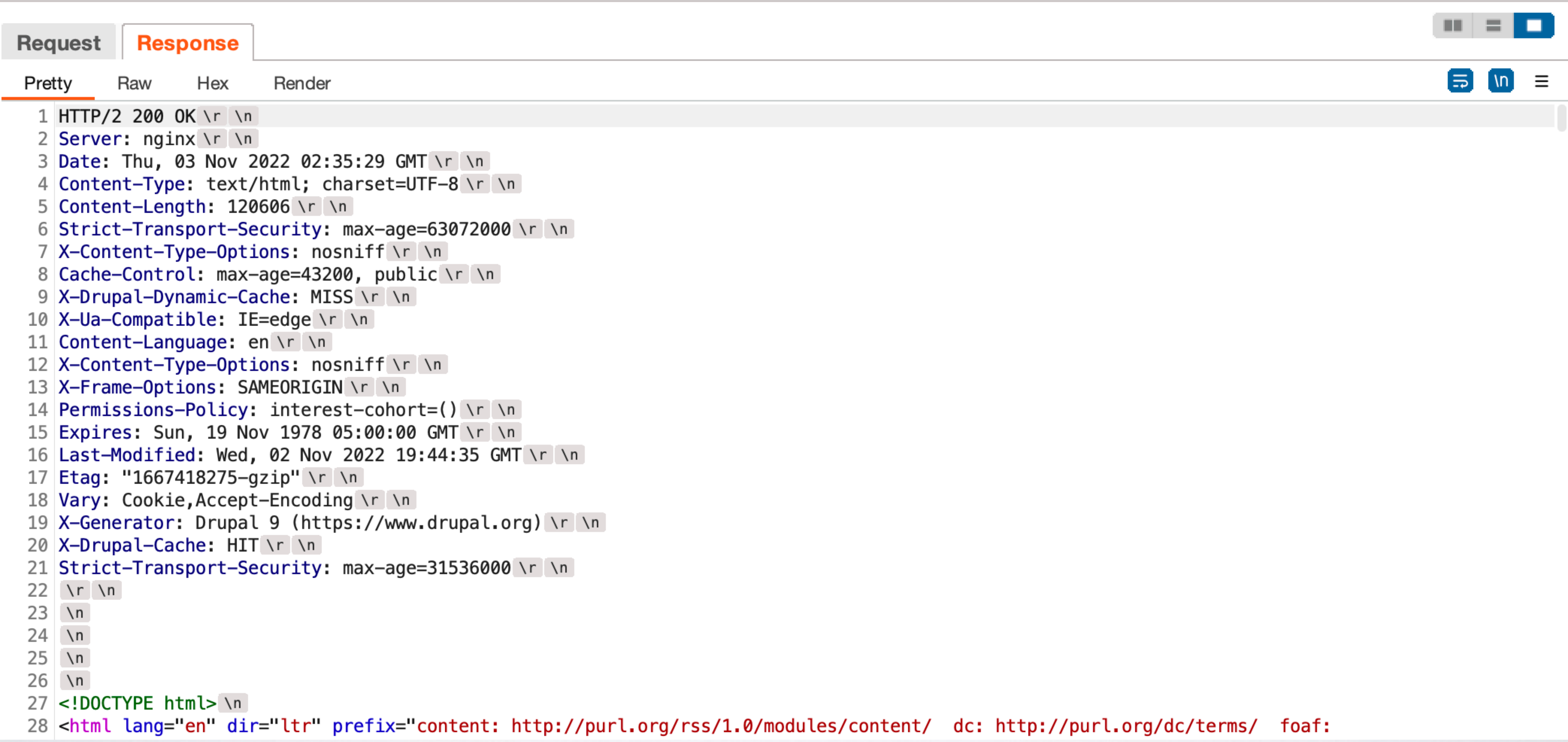


```
GET /admissions/program-listing-categories?title=All&field_admissions_category_new_value=Computer+Science+%28regular+and+co-op%29 HTTP/2 \r \n
Host: www.utsc.utoronto.ca \r \n
Cookie: __utma=155658239.870311845.1667442435.1667442435.1667442435.1; __utmc=155658239; __utmz=
155658239.1667442435.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none); __utmt=1; __utmb=155658239.1.10.1667442435; _gid=
GA1.2.842220911.1667442435; _hjSessionUser_2381121=
eyJpZCI6IjJiYWRkNjUzLTU1YTYtNWIzMi1hZjg5LWQ5ZTNkNTM1MTkyNiIsImNyZWF0ZWQiOjE2Njc0NDI4NzIzODQsImV4aXN0aW5nIjpmYWxzZX0=; _hjFirstSeen=1;
_hjSession_2381121=eyJpZCI6ImUxNmM5ZWRiLTNkZTgtNDVmZS1hZjYyLTBhY2ViNjg4YjczMiIsImNyZWF0ZWQiOjE2Njc0NDI4NzI0NjYsImluU2FtcGxlIjp0cnVlfQ==;
_hjAbsoluteSessionInProgress=1; _ga=GA1.2.870311845.1667442435; _gat_gtag_UA_38074443_1=1; _ga_80VDTXHB7F=GS1.1.1667442877.1.1.1667442887.0.0.0
; _gat_UA-15755348-1=1; _gat_UA-103505937-1=1 \r \n
Upgrade-Insecure-Requests: 1 \r \n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63 Safari/537.36 \r \n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
 \r \n
Sec-Fetch-Site: cross-site \r \n
Sec-Fetch-Mode: navigate \r \n
Sec-Fetch-User: ?1 \r \n
Sec-Fetch-Dest: document \r \n
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24" \r \n
Sec-Ch-Ua-Mobile: ?0 \r \n
Sec-Ch-Ua-Platform: "macOS" \r \n
Accept-Encoding: gzip, deflate \r \n
Accept-Language: en-US,en;q=0.9 \r \n
 \r \n

```

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages#http_requests

The HTTP protocol: Response



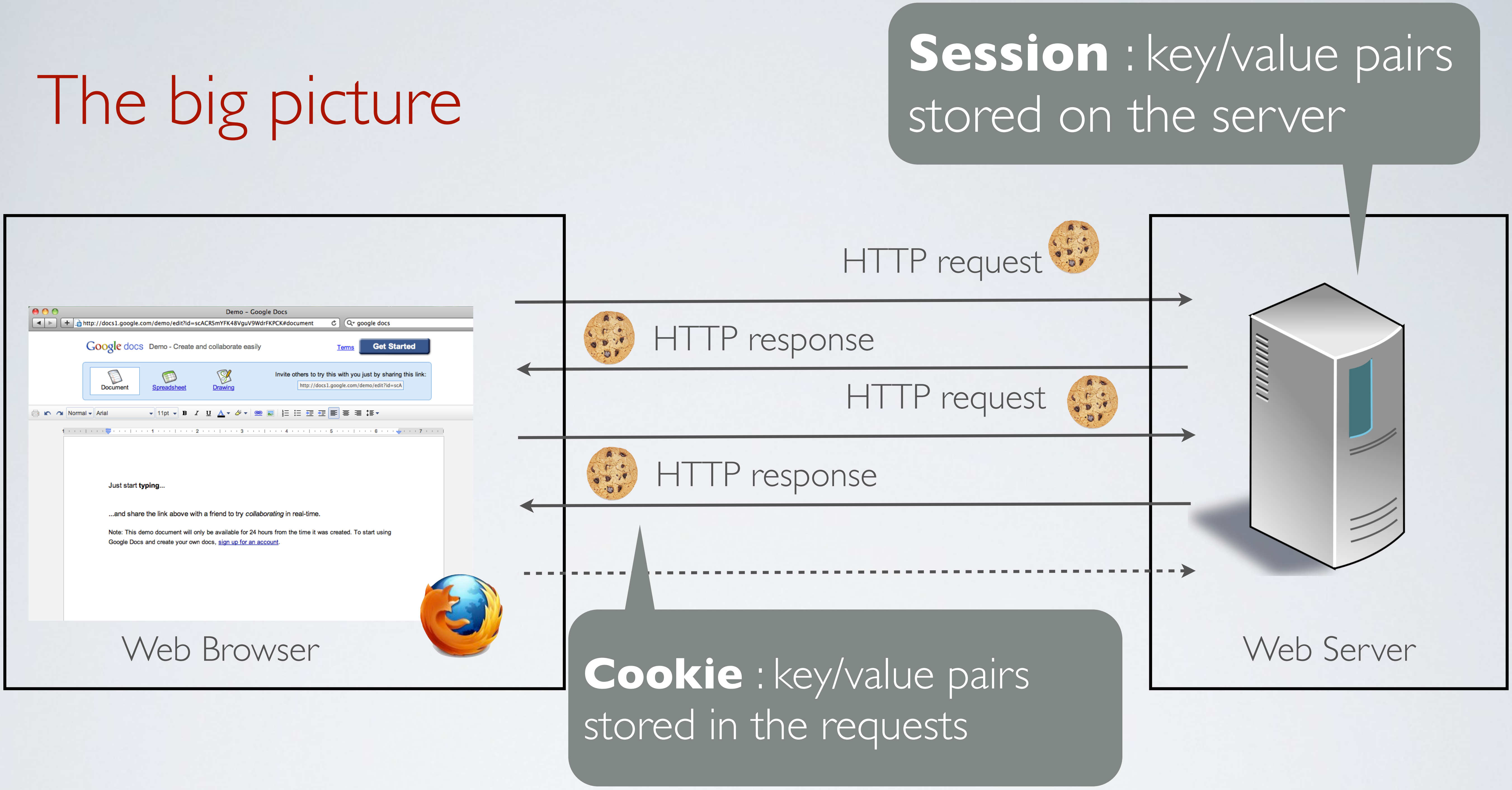
```
HTTP/2 200 OK \r \n
Server: nginx \r \n
Date: Thu, 03 Nov 2022 02:35:29 GMT \r \n
Content-Type: text/html; charset=UTF-8 \r \n
Content-Length: 120606 \r \n
Strict-Transport-Security: max-age=63072000 \r \n
X-Content-Type-Options: nosniff \r \n
Cache-Control: max-age=43200, public \r \n
X-Drupal-Dynamic-Cache: MISS \r \n
X-Ua-Compatible: IE=edge \r \n
Content-Language: en \r \n
X-Content-Type-Options: nosniff \r \n
X-Frame-Options: SAMEORIGIN \r \n
Permissions-Policy: interest-cohort=() \r \n
Expires: Sun, 19 Nov 1978 05:00:00 GMT \r \n
Last-Modified: Wed, 02 Nov 2022 19:44:35 GMT \r \n
Etag: "1667418275-gzip" \r \n
Vary: Cookie,Accept-Encoding \r \n
X-Generator: Drupal 9 (https://www.drupal.org) \r \n
X-Drupal-Cache: HIT \r \n
Strict-Transport-Security: max-age=31536000 \r \n
 \r \n
 \n
 \n
 \n
 \n
<!DOCTYPE html> \n
<html lang="en" dir="ltr" prefix="content: http://purl.org/rss/1.0/modules/content/  dc: http://purl.org/dc/terms/  foaf:
```

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages#http_responses

Stateless …

- Authentication and Authorization managed via **session id** between the browser and the web application
- This session id should be **unique** and **unforgeable**
    - Stored in the cookie
- The session id is also stored and validated on the server

The big picture


Session : key/value pairs stored on the server
HTTP request
HTTP response
HTTP request
HTTP response
Web Browser
Web Server
Cookie : key/value pairs stored in the requests
Demo - Google Docs
http://docs1.google.com/demo/edit?id=scACRSmYFK48VguV9WdrFKPCK#document
google docs
Google docs Demo - Create and collaborate easily
Terms
Get Started
Document
Spreadsheet
Drawing
Invite others to try this with you just by sharing this link:
http://docs1.google.com/demo/edit?id=scA
Normal
Arial
11pt

Just start **typing**...

...and share the link above with a friend to try *collaborating* in real-time.

Note: This demo document will only be available for 24 hours from the time it was created. To start using Google Docs and create your own docs, sign up for an account.

The user can **create, modify, delete** the session ID in the cookie

But **cannot access** the key/value pairs stored on the server

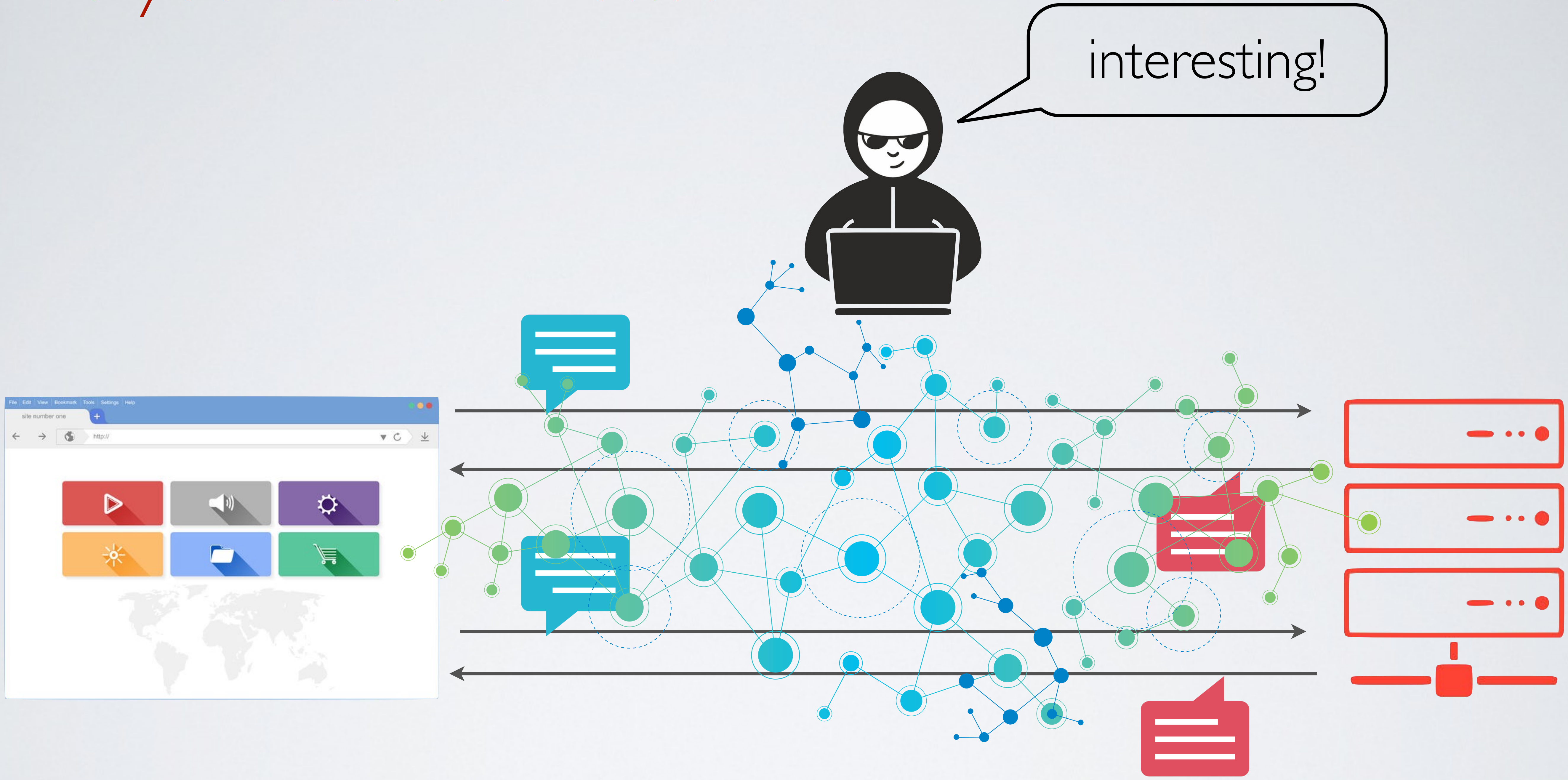
Insufficient Transport Layer Protection

a.k.a the need for HTTPs

How to steal user’s credentials

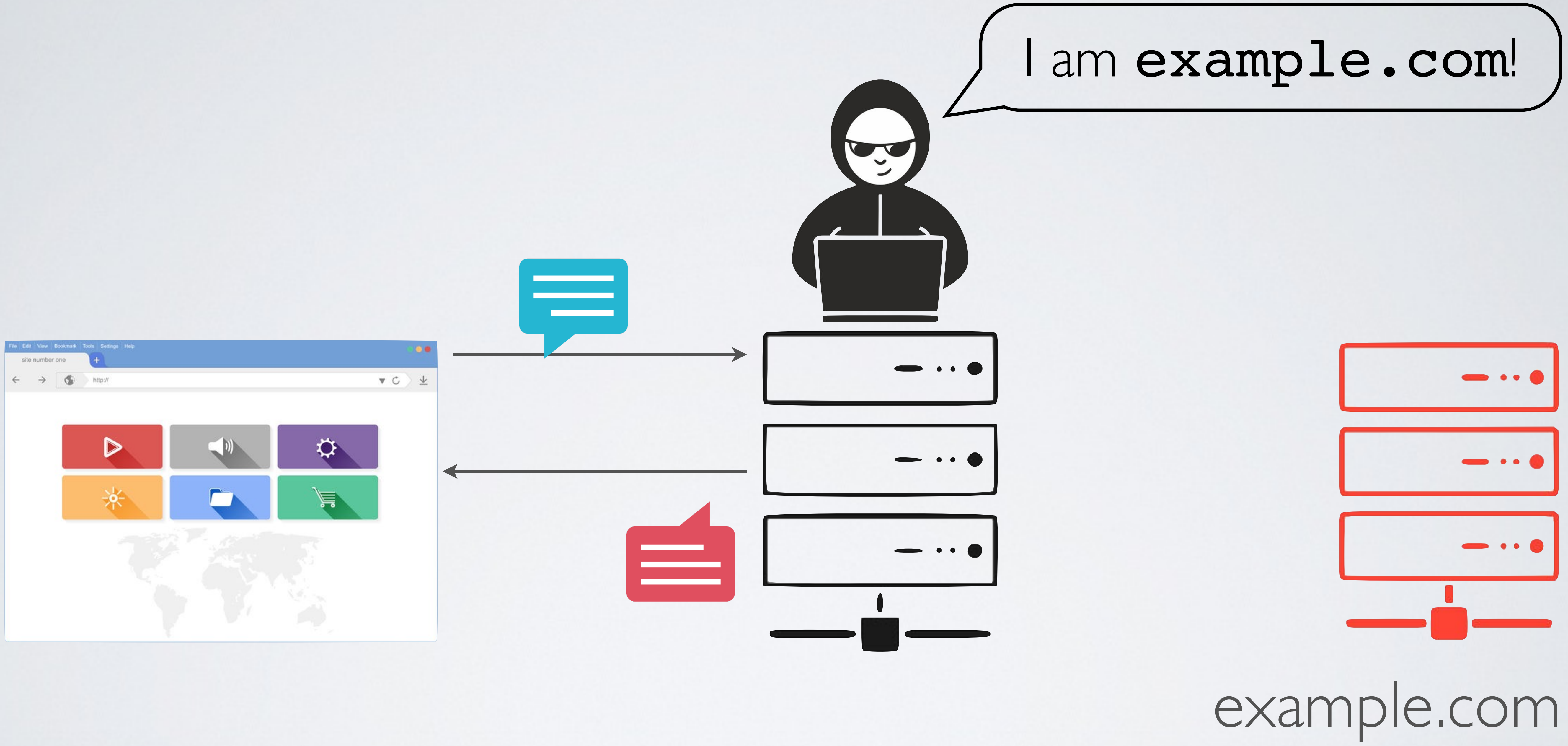
Brute force the user's password or session id
Steal the user's password or session ID
imgflip.com

Do you trust the network?



- Threat 1 : an attacker **can eavesdrop** messages sent back and forth

Do you *really* trust the network?



- Threat 2 : an attacker **can tamper with** messages sent back and forth

Confidentiality and Integrity

- Threat 1 : an attacker **can eavesdrop** messages sent back and forth

  Confidentiality: how do exchange information secretly?

- Threat 2 : an attacker **can tamper** messages sent back and forth

  Integrity: How do we exchange information reliably?

Why and when using HTTPS?

HTTPS = HTTP + TLS

➡ TLS provides

- confidentiality: end-to-end secure channel
- integrity: authentication handshake

➡ HTTPS protects any data send back and forth including:

- login and password
- session ID

✓ **HTTPS everywhere**
HTTPS must be used during the entire session

Be careful of mixed content

Mixed-content happens when:

1. an HTTPS page contains elements (ajax, js, image, video, css ...) served with HTTP

2. an HTTPS page transfers control to another HTTP page within the same domain

- authentication cookie will be sent over HTTP
- Modern browsers block (or warn of) mixed-content

`Secure` cookie flag

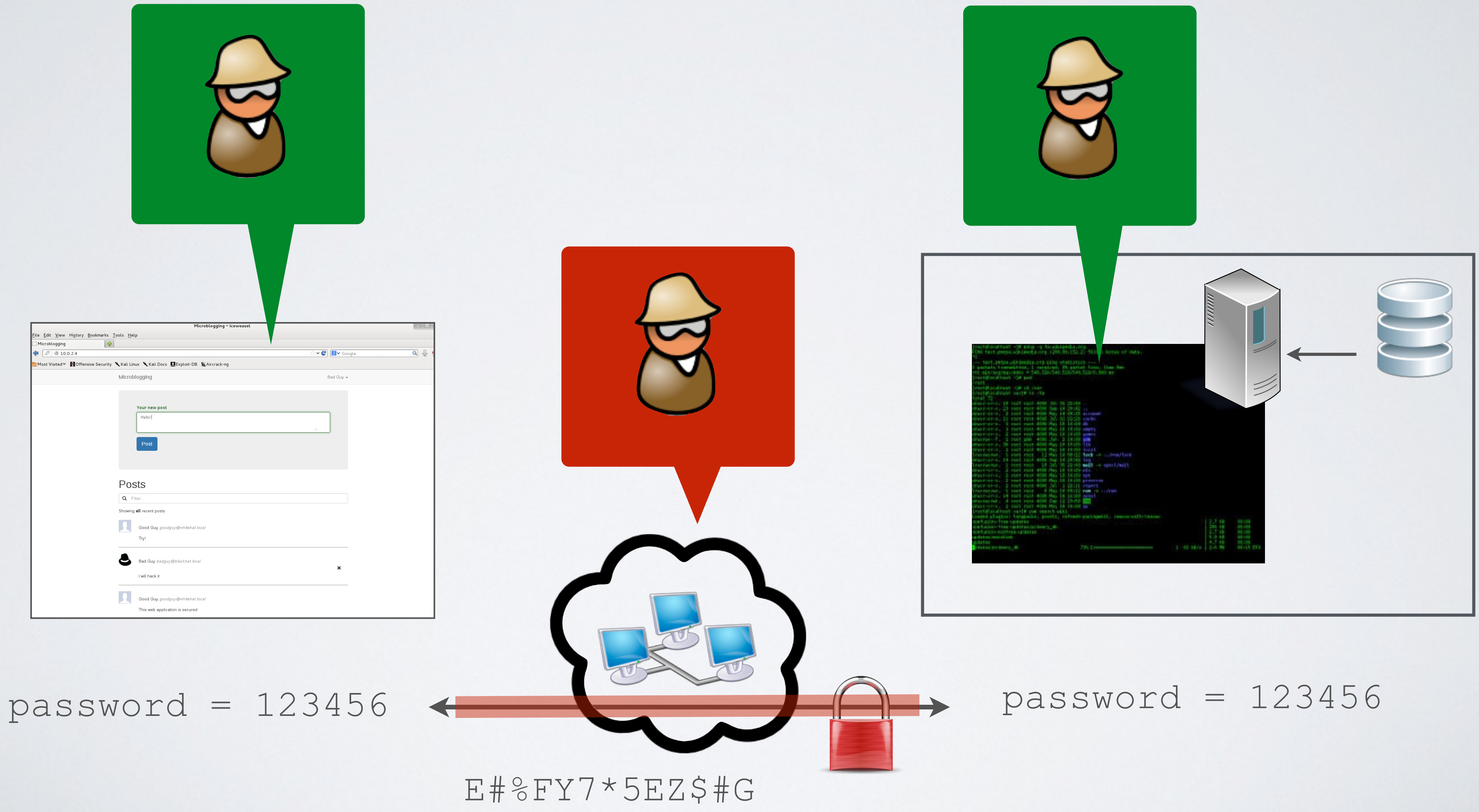
✓ The cookie will be sent over HTTPS exclusively

➡ Prevents authentication cookie from leaking in case of mixed-content

Do/Don't with HTTPS

- Always use HTTPS exclusively (in production)
- Always have a valid and signed certificate (no self-signed cert)
- Always avoid using absolute URL (mixed-content)
- Always use **`secure`** cookie flag with authentication cookie

Limitation of HTTPS



Microblogging – Iceweasel

File Edit View History Bookmarks Tools Help

Microblogging

10.0.2.4

Most Visited Offensive Security Kali Linux Kali Docs Exploit-DB Aircrack-ng

Microblogging Bad Guy

Your new post

Hello!

Post

Posts

Filter...

Showing **all** recent posts

Good Guy *goodguy@whitehat.local*

Try!

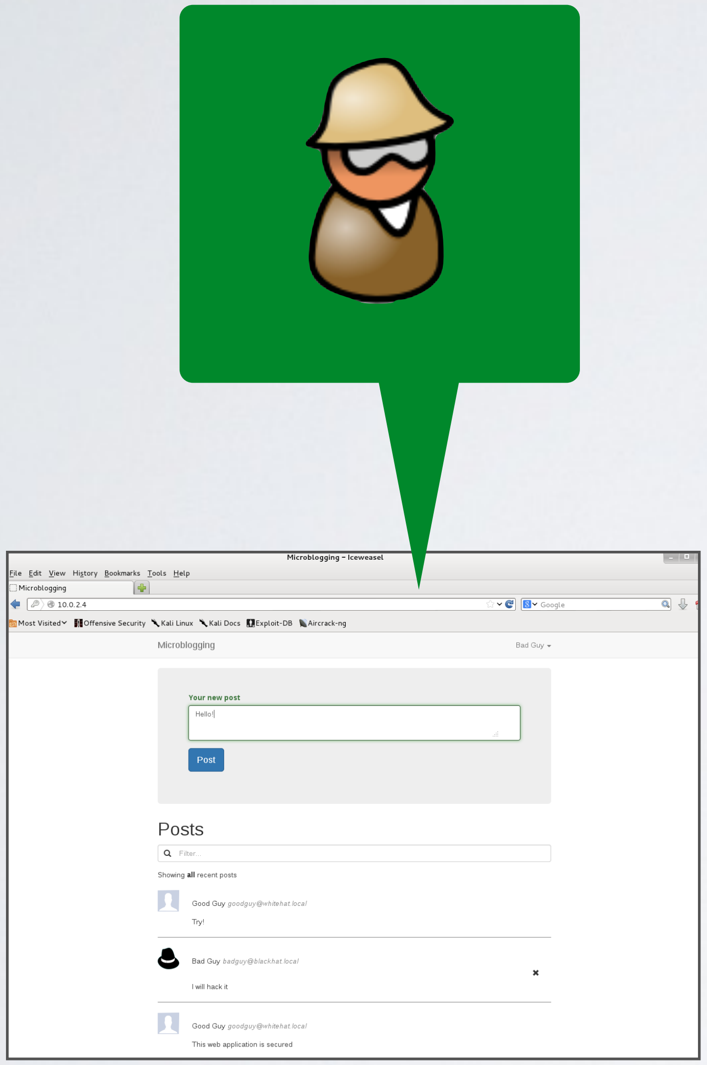
Bad Guy *badguy@blackhat.local*

I will hack it

Good Guy *goodguy@whitehat.local*

This web application is secured

Stealing secrets from the client


Microblogging – Iceweasel
File Edit View History Bookmarks Tools Help
Microblogging
10.0.2.4
Google
Most Visited Offensive Security Kali Linux Kali Docs Exploit-DB Aircrack-ng
Microblogging
Bad Guy
Your new post
Post
Posts
Filter...

Hello!

Showing **all** recent posts

Good Guy *goodguy@whitehat.local*

Try!

Bad Guy *badguy@blackhat.local*

I will hack it

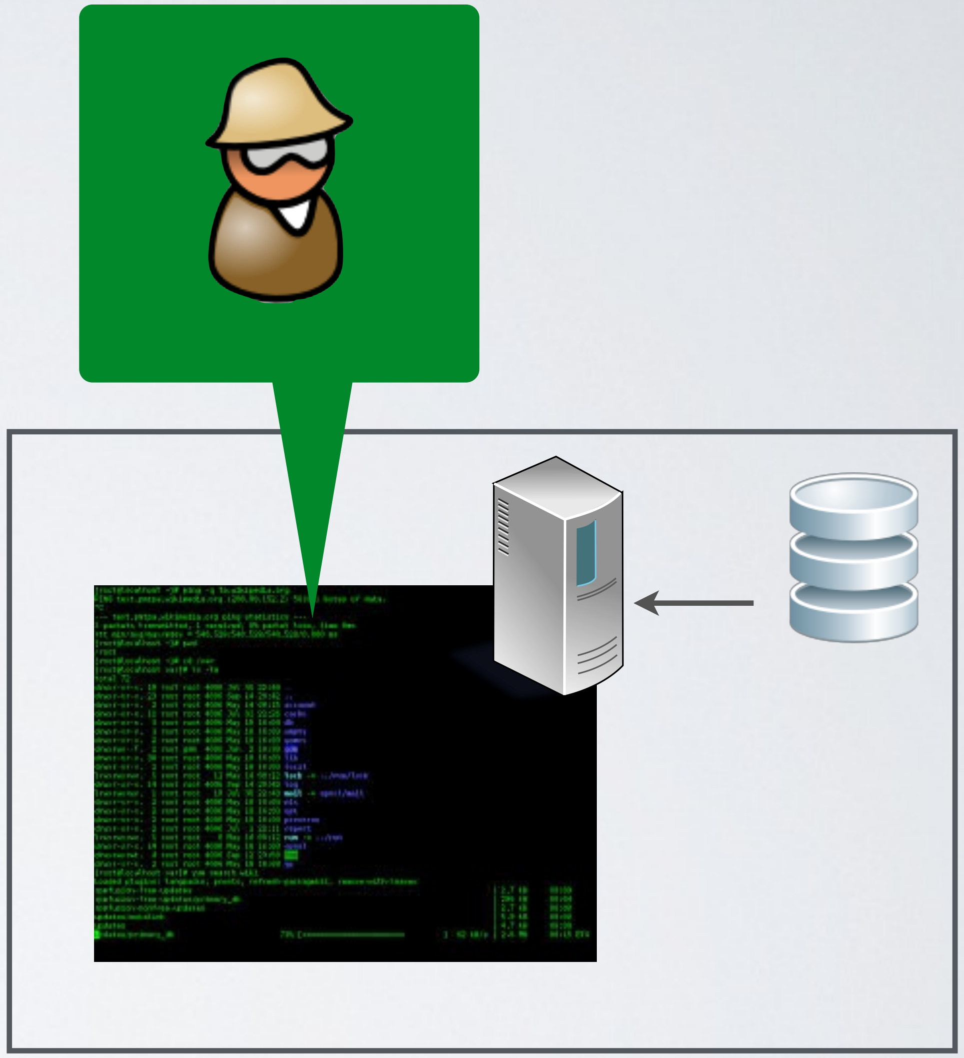
Good Guy *goodguy@whitehat.local*

This web application is secured

- Social engineering - Phishing
- Keyloggers (keystroke logging)
- Data mining (emails, logs)
- Hack the client's code

Stealing secrets from the server

- Hack the server
- Hack the server's side code



Client Side

Who is the client?

- An arbitrary application that understands the HTTP protocol
- A front-end app, another web app, a browser, telnet, curl etc.
- Optionally and weakly identifiable via the User-Agent HTTP header
- Generally untrusted
- Faces some threats when parsing or rendering HTTP response or arbitrary data
- Poses some threats in sending HTTP requests to a web server

Client side threats

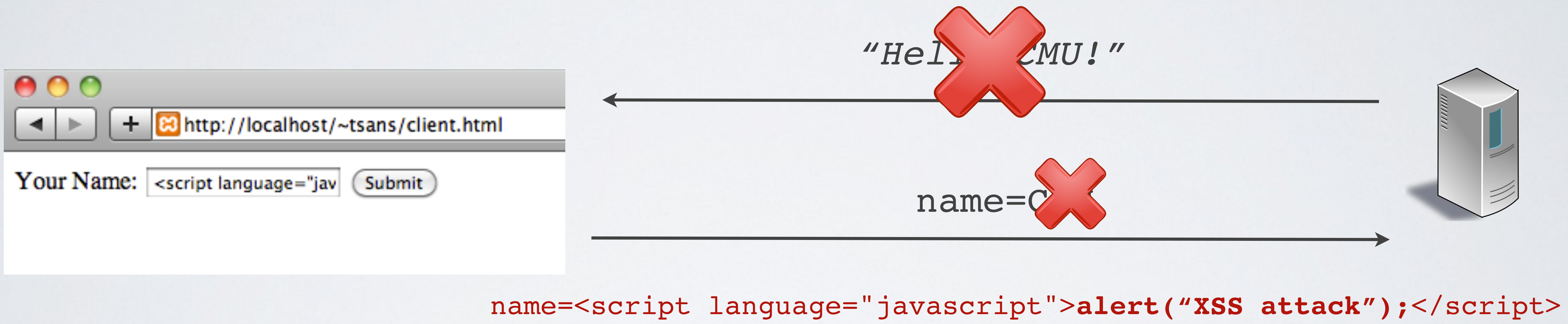
- **Confidentiality**
  - An attacker can read secrets intended only for the client
- **Integrity**
  - An attacker can coerce the client into making unintended requests
  - An attacker can modify/falsify data parsed or rendered by the client
- **Availability**
  - An attacker can "crash" the client

Common client side vulnerabilities

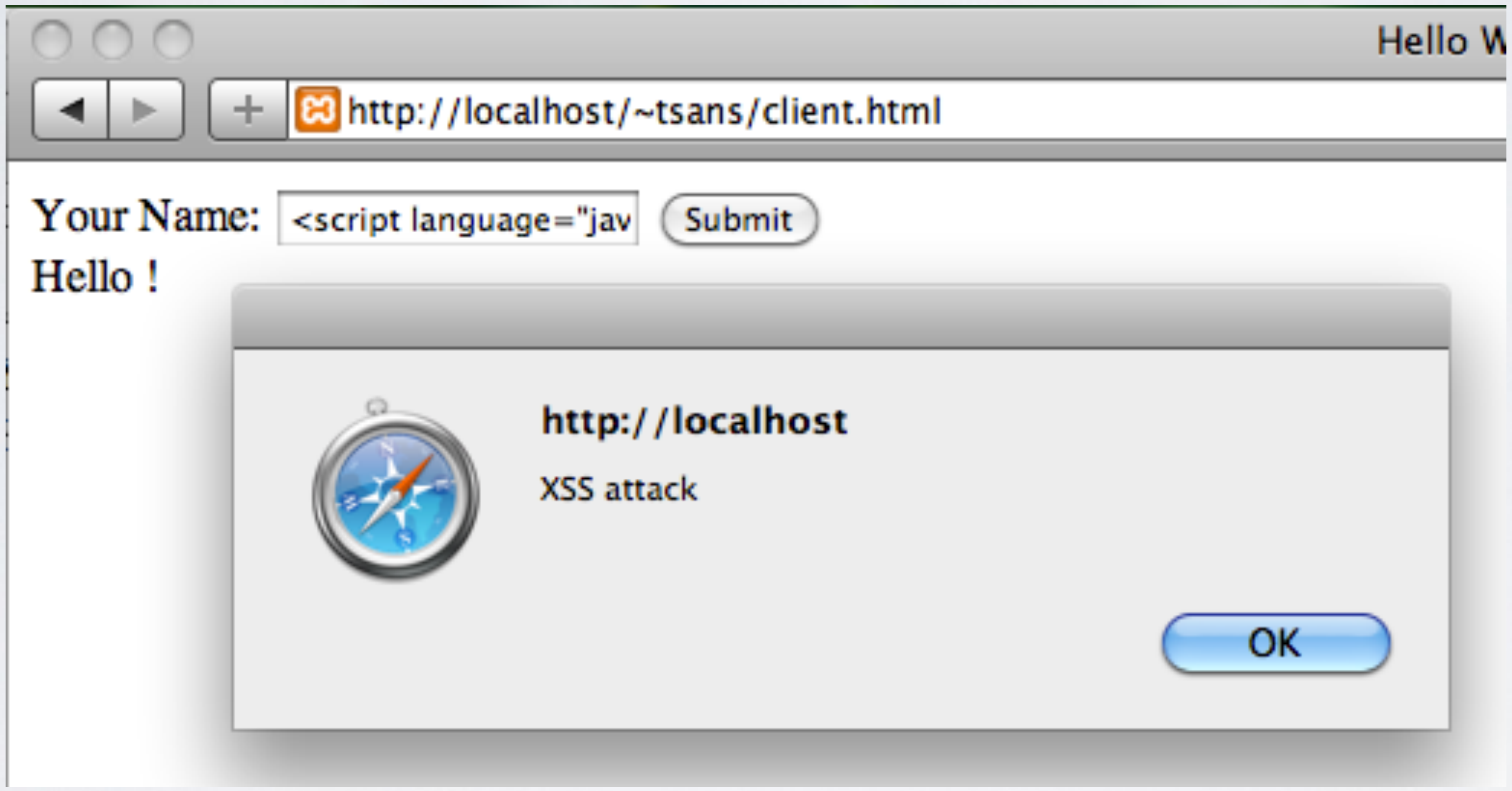
- **Cross-site scripting (XSS)**
- Cross-site request forgery
- Clickjacking

Cross-Site Scripting (XSS)

Cross-Site Scripting Attack (XSS attack)


“Hello <script language="javascript">**alert(“XSS attack”);**</script>!”
“Hel CMU!”
http://localhost/~tsans/client.html
Your Name:
<script language="jav
Submit
name=C
name=<script language="javascript">**alert(“XSS attack”);**</script>

XSS Attack = Javascript Code Injection

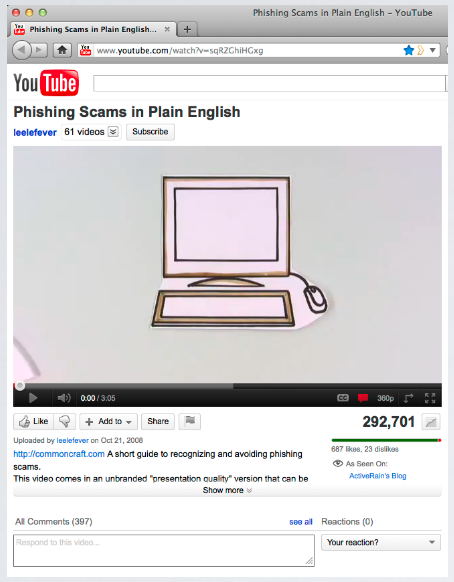


Problem

➡ An attacker can inject **arbitrary javascript code** in the page that will be executed by the browser

- **Inject illegitimate content** in the page (same as content spoofing)
- **Perform illegitimate HTTP requests** through Ajax (same as a CSRF attack)
- **Steal Session ID** from the cookie
- **Steal user's login/password** by modifying the page to forge a perfect scam

Forging a perfect scam


Phishing Scams in Plain English – YouTube

Phishing Scams in Plain English

www.youtube.com/watch?v=sqRZGhiHGxg

Phishing Scams in Plain English

leelefever 61 videos Subscribe

292,701

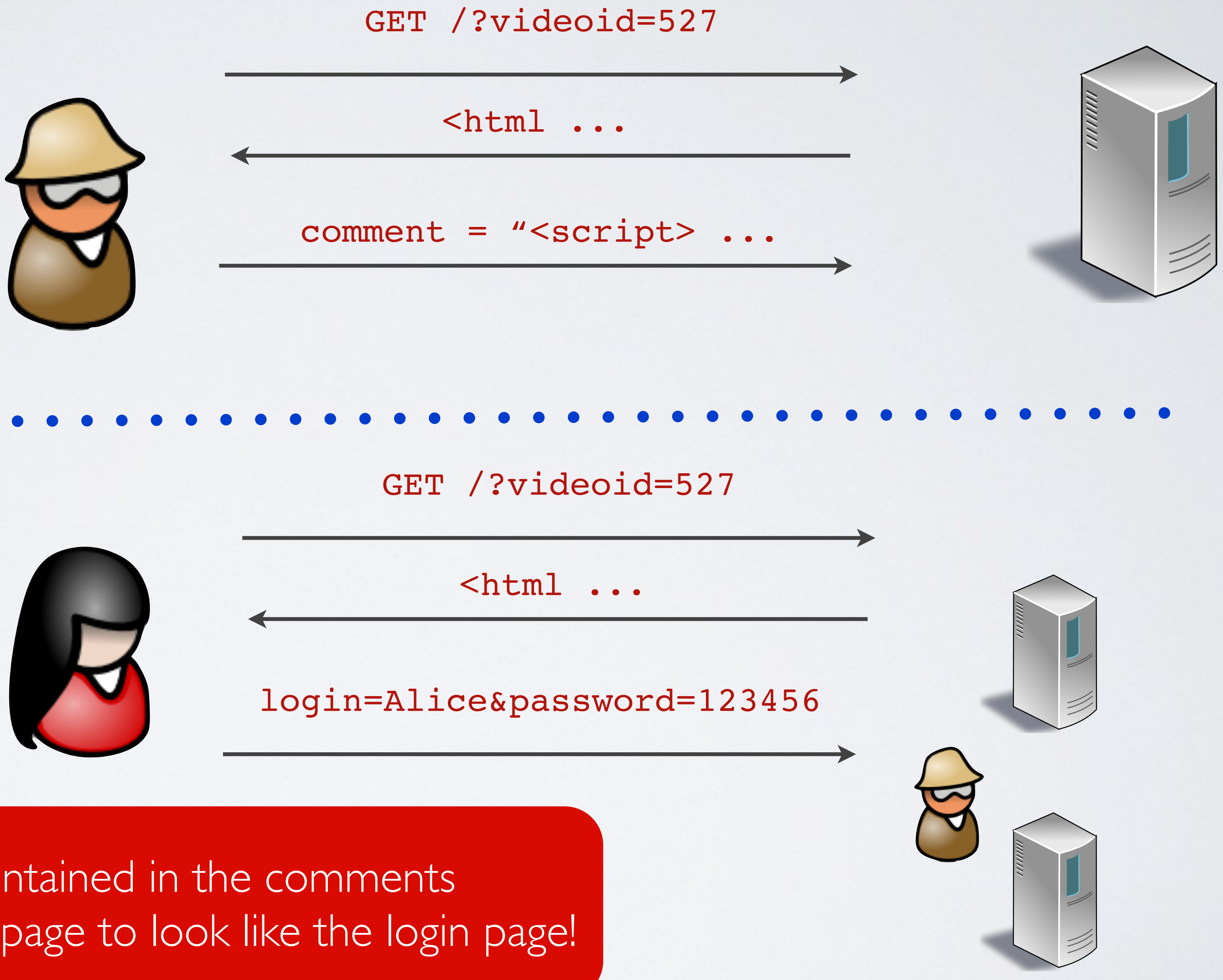
Uploaded by leelefever on Oct 21, 2008

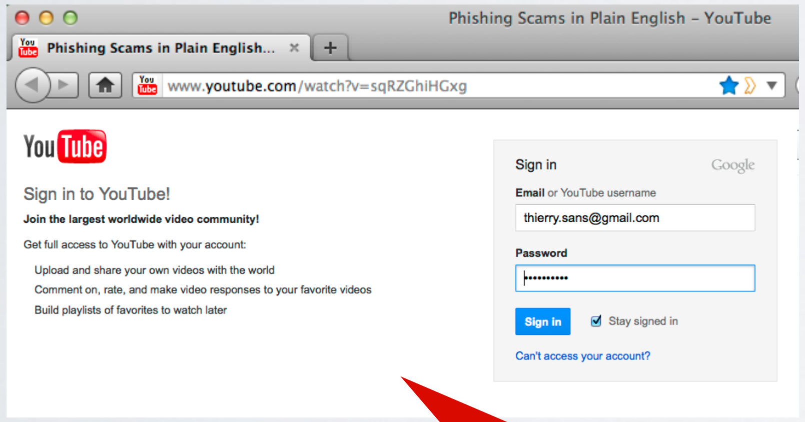
http://commoncraft.com A short guide to recognizing and avoiding phishing scams.

This video comes in an unbranded "presentation quality" version that can be

All Comments (397)

Reactions (0)




Phishing Scams in Plain English – YouTube

Phishing Scams in Plain English...

www.youtube.com/watch?v=sqRZGhiHGxg

Sign in to YouTube!

Join the largest worldwide video community!

Get full access to YouTube with your account:

Upload and share your own videos with the world

Comment on, rate, and make video responses to your favorite videos

Build playlists of favorites to watch later

Sign in Google

Email or YouTube username

thierry.sans@gmail.com

Password

••••••••••

Sign in [✓ Stay signed in]

Can't access your account?

The script contained in the comments modifies the page to look like the login page!

* *Notice that Youtube is **not** vulnerable to this attack*

It gets worse - XSS Worms

Spread on social networks

- Samy targeting MySpace (2005)
- JTV.worm targeting Justin.tv (2008)
- Twitter worm targeting Twitter (2010)

Variations on XSS attacks

- **Reflected XSS**
  Malicious data sent to the backend are immediately sent back to the frontend to be inserted into the DOM

- **Stored XSS**
  Malicious data sent to the backend are stored in the database and later-on sent back to the frontend to be inserted into the DOM

- **DOM-based attack**
  Malicious data are manipulated in the frontend (javascript) and inserted into the DOM

Server Side

Who is the web server?

- Mostly trusted domain; sensitive operations must be performed here
- Hosts resources and defines how they are accessed
- May interact with other back-end components to satisfy the HTTP requests
- Faces some threats when parsing HTTP requests or arbitrary data
- Maybe weakly and optionally identifiable from a banner

Server side threats

- **Confidentiality**
  - An attacker can read secrets from the server
- **Integrity**
  - An attacker can coerce the server into making unintended requests or responses
- **Availability**
  - An attacker can prevent the server from responding to clients

Common server side vulnerabilities

- Broken Authentication
- Broken Access Control
- **Server Side Request Forgery**
- **XML External Entities Injection**
- **SQL Injection**
- Command Injection

(No)SQL Injection

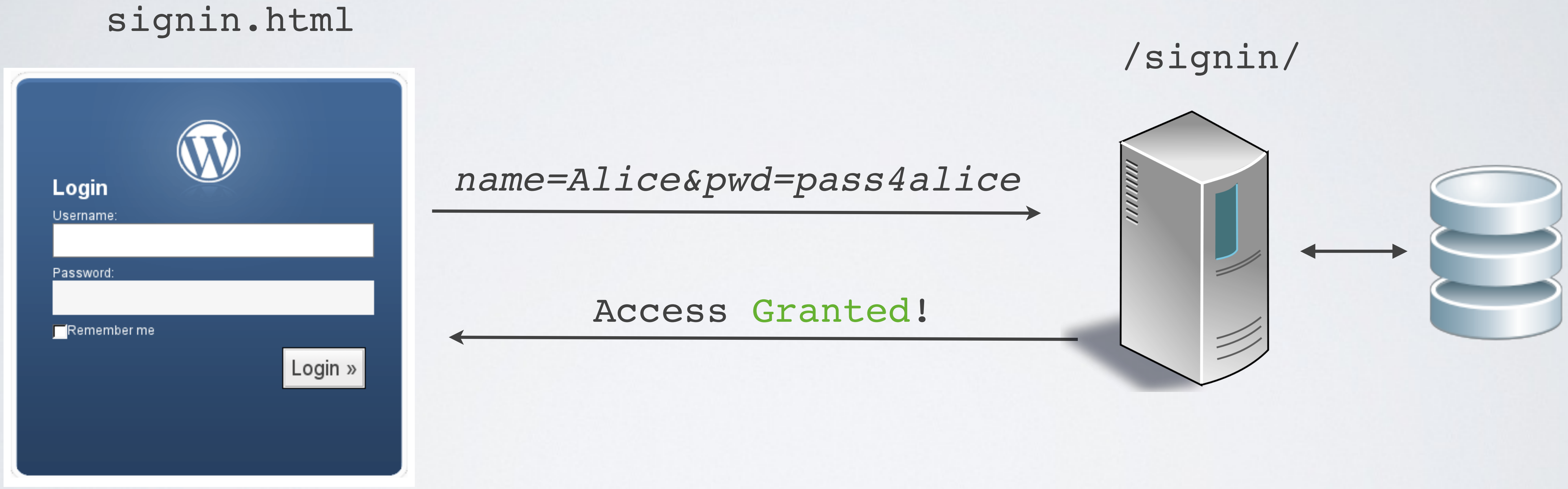
Problem

➡ An attacker can inject SQL/NoSQL code

◉ Retrieve, add, modify, delete information

◉ Bypass authentication

Checking password



Bypassing password check

```
db.run("SELECT * FROM users
WHERE USERNAME = '" + username + "'
  AND PASSWORD = '" + password + "'"
```

```
username: alice
password: pas[illegible]lice
```

```
blah' OR '1'='1
```

NoSQL Injection

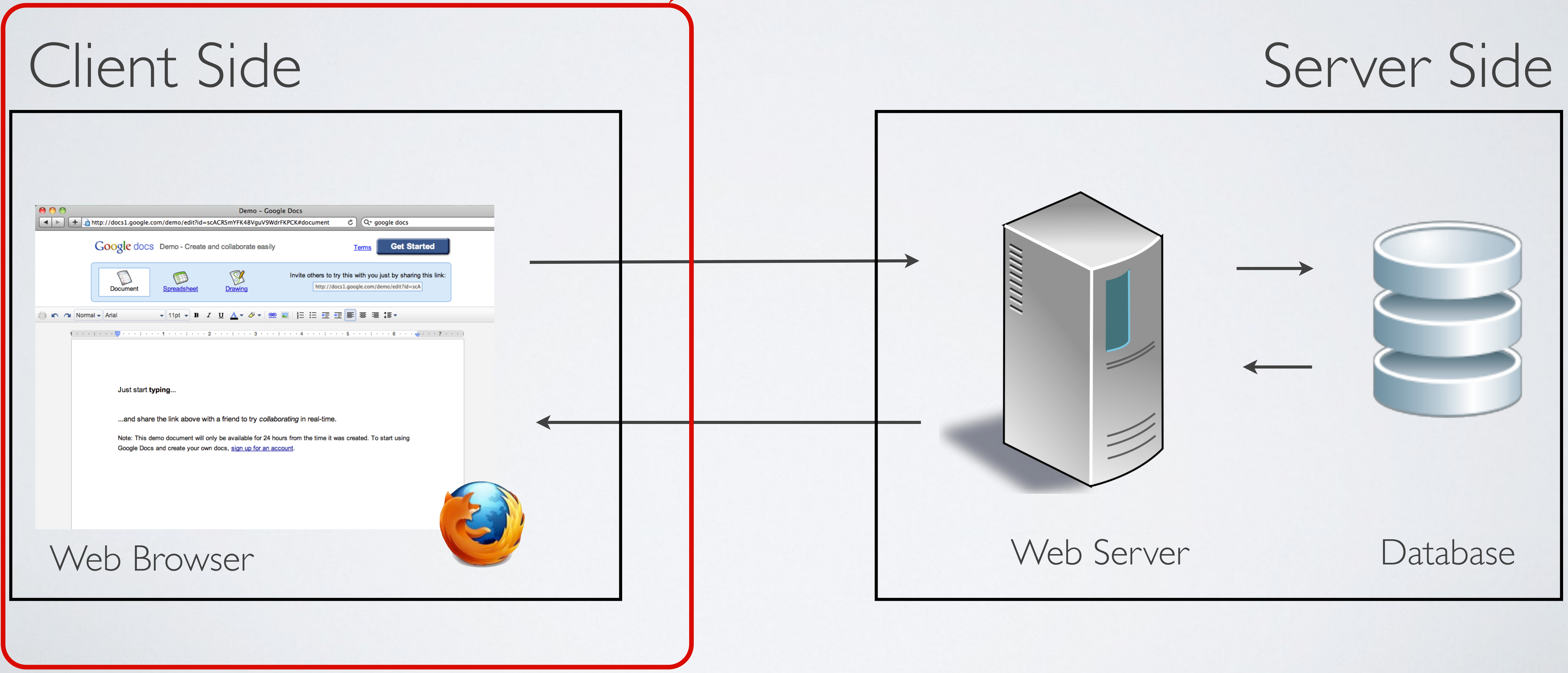
```
db.find({  username: username,
           password: password   });
```

```
username: alice
password: pas   lice
```

```
{gt: ""}
```

Conclusion

You have **absolutely no control** on the client


Client Side
Server Side
Demo – Google Docs
http://docs1.google.com/demo/edit?id=scACRSmYFK48VguV9WdrFKPCK#document
google docs
Google docs
Demo - Create and collaborate easily
Terms
Get Started
Document
Spreadsheet
Drawing
Invite others to try this with you just by sharing this link:
http://docs1.google.com/demo/edit?id=scA
Normal
Arial
11pt
Web Browser
Web Server
Database

Just start **typing**...

...and share the link above with a friend to try *collaborating* in real-time.

Note: This demo document will only be available for 24 hours from the time it was created. To start using Google Docs and create your own docs, sign up for an account.

Resources

- Web Security Academy & Burp Suite
    - Sequel to OG "The Web Application Hacker's Handbook"
    - https://portswigger.net/web-security/learning-path
    - https://portswigger.net/burp/documentation/desktop/tutorials?utm_source=burp_suite_community&utm_medium=learn_tab&utm_campaign=tutorials
- Hacker101 by HackerOne
    - https://www.hacker101.com/videos
    - https://ctf.hacker101.com/
- Damn Vulnerable Web Application
    - https://www.kali.org/tools/dvwa/
    - https://github.com/digininja/DVWA
- Damn Vulnerable Web Sockets
    - https://owasp.org/www-project-damn-vulnerable-web-sockets/
- More in this week's reading section