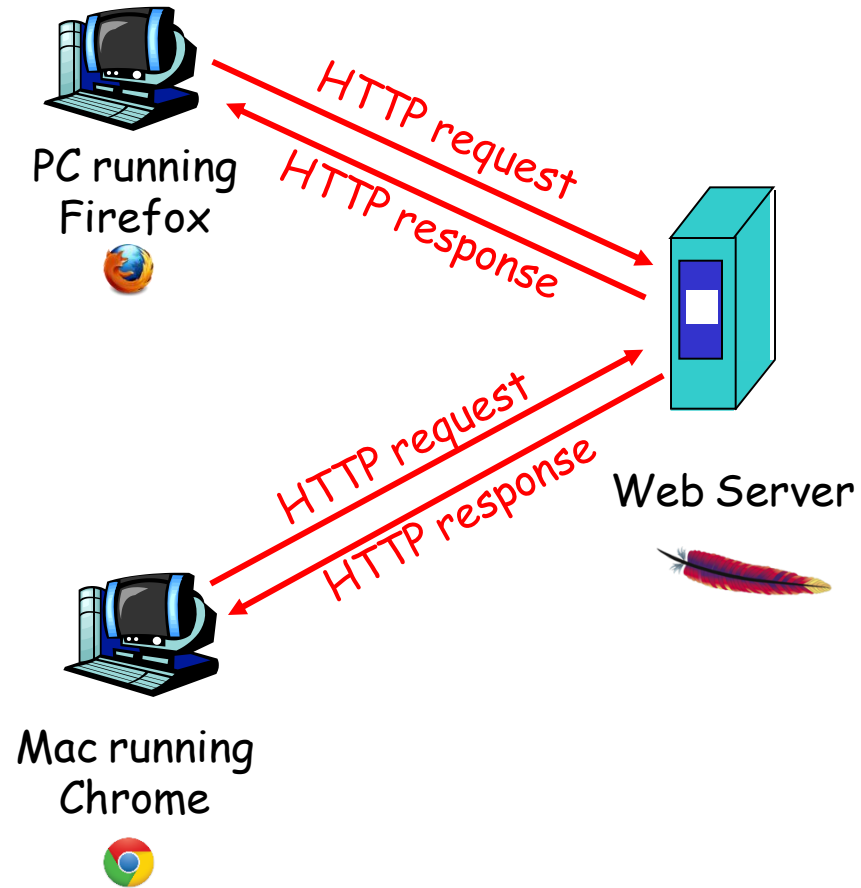


CS 410/510: Web Basics

Basics

- Web Clients
- HTTP
- Web Servers



Web Clients

Basic Terminology | HTML | JavaScript

Terminology

- **Web page** consists of objects
 - Each object is addressable by a **URL**

`www.someschool.edu/someDept/pic.gif`

host name

path name

- **Web page** is (at minimum) an HTML file with several referenced objects.

Web clients

- Retrieve and render content (e.g. HTML, images)
- Retrieve and execute JavaScript
- Examples
 - Web browser (Chrome, Firefox, Safari)
 - Command-line tool (curl, wget)
 - Program (Python requests)

HTML, JavaScript

- HTML - Hypertext Markup Language
- Javascript - Executable code for client to run
 - In all browsers

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

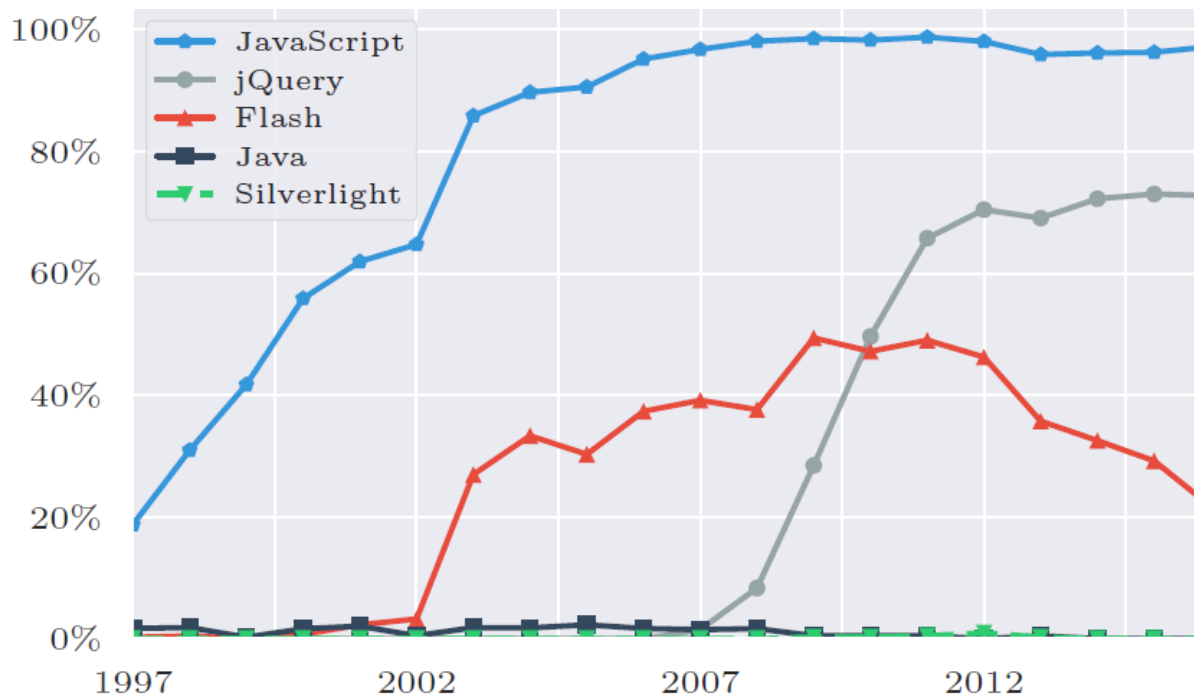
<script>
window.alert(5 + 6);
</script>

</body>
</html>
```

Mixing code and data!

Importance of Javascript to web security

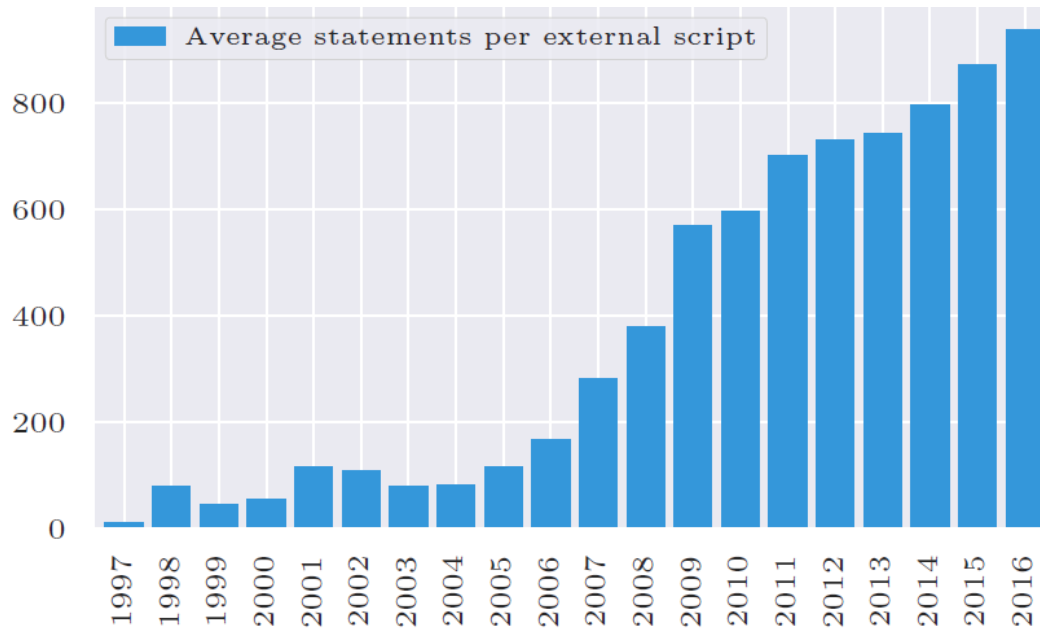
- Ubiquitous
 - jQuery = popular Javascript library
 - Many exploits delivered via rogue Javascript



Technologies used by top 500 sites

Problem is worsening

- Surface area of attack increasing due to complexity
- Not ideal for dynamically-typed languages like Javascript
 - Motivates Typescript, Flow, and AtScript



JavaScript Statement Statistics

Viewing HTML/JavaScript

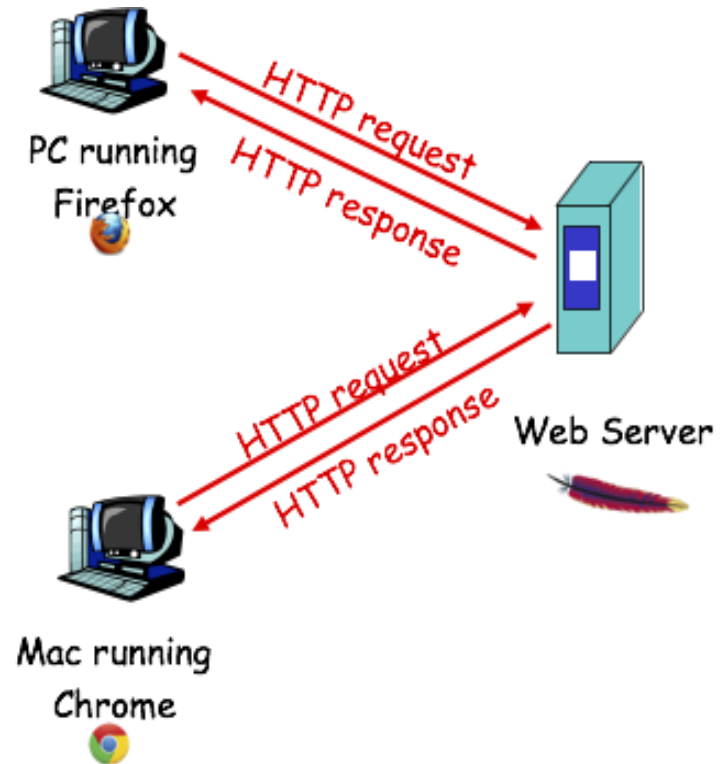
- Developer tools
 - (Ctrl-Shift-I) on both Chrome and Firefox
 - Right click => Inspect Element
 - In Elements
 - Ability to directly edit HTML elements in page
 - In Console
 - Console output (console.log messages)
 - Access to JavaScript engine in page's context (alert(document.cookie))
 - In Network
 - Access to page's network requests
 - In Application
 - Access to page's storage/cookies

HTTP

Headers | Requests/Responses | Cookies

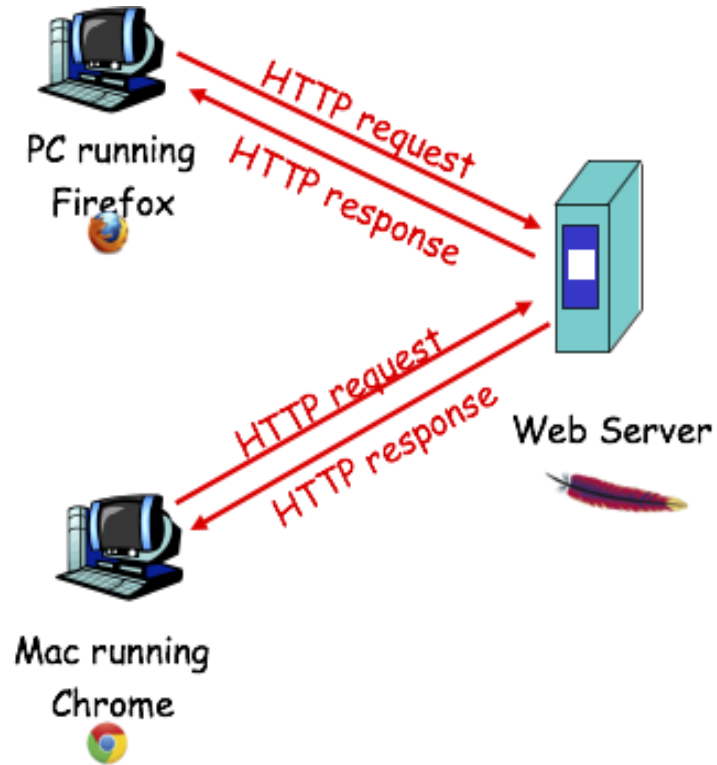
HTTP

- Hypertext Transport Protocol
 - Language spoken between client and server
 - Standard message format for headers to implement caching, authentication, session management, localization, etc.



HTTP

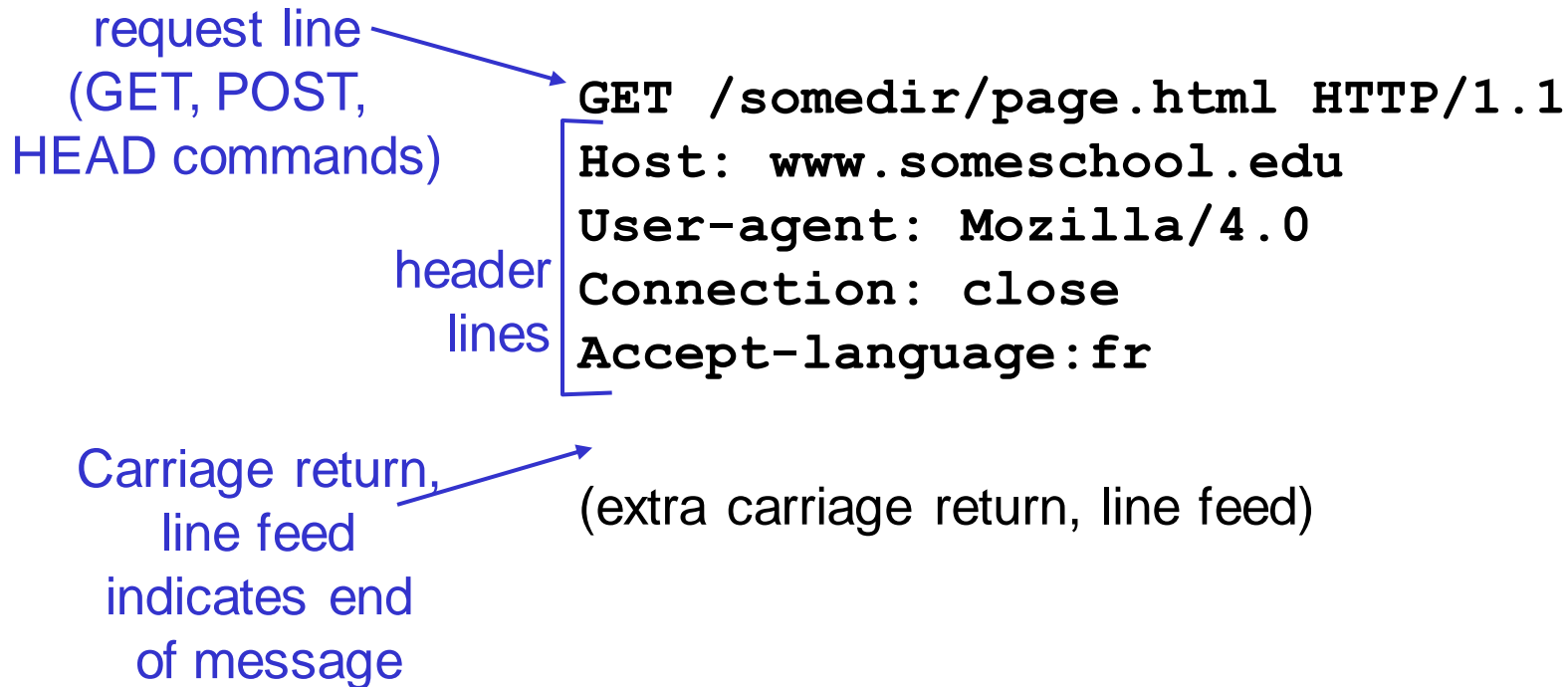
- Client initiates bi-directional connection to server on port 80
- Server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between client/server
 - Messages encoded in text



HTTP Headers – Request (client)

- Two types of HTTP messages: *request, response*
- **HTTP request message:**
 - ASCII (human-readable format)

`http://www.someschool.edu/somedir/page.html`



HTTP Headers – Response (server)

status line
(protocol
status code
status phrase)

HTTP/1.1 200 OK
Connection: close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998
Content-Length: 6821
Content-Type: text/html

header
lines

data, e.g.,
requested
HTML file

<html>
<head>
<title>
...

HTTP status codes

- Returned in first line of response
 - 200 OK: the request was processed successfully.

```
HTTP/1.1 200 OK
```

```
Date: Thu, 06 Aug 1998 12:00:15 GMT
```

```
Server: Apache/1.3.0 (Unix)
```

```
...
```

- 302 Found: used to redirect users, for example when they logout, to send them back to the login page.
- 401 Unauthorized: when the resource's access is restricted.
- 404 Not found: the resource requested by the client was not found.
- 500 Internal Server Error: an error occurred during the processing of the request.

HTTP Headers in action

- Demo

- `$ nc thefengs.com 80`

- Opens TCP connection to port 80

- Anything typed in is sent to port 80 at `thefengs.com`

- Type in a GET HTTP request:

```
GET / HTTP/1.1
```

```
Host: thefengs.com
```

- Type this in and hit RETURN twice. You sent this minimal, but complete request to HTTP server.

- View the response message sent from server.

HTTP headers for class

- Authentication
 - Basic authentication
 - Apache “.htaccess” file specifying users and passwords
 - NOT secure (only included for natas levels)
 - HTTP response header used to trigger web browser prompt
 - WWW-authenticate:
 - HTTP request header used to send credentials (base64-encoded)
 - Authorization:
 - e.g. Authorization: basic YWRtaW46YWRtaW4Kpucca % echo YWRtaW46YWRtaW4K| base64 -d
admin:admin
- Referring page
 - HTTP request header used to send page the request originated from
 - Used for tracking
 - Referer:
 - Load Developer Tools
 - Access Prezi from <https://crypto.cyberpdx.org/>
 - View Network request in

HTTP Headers – Cookies

- HTTP is initially “stateless”
 - Does not remember prior requests or users
- Many websites require and need state
 - Yahoo Mail (saves user information and who the user is)
 - Amazon Shopping Cart (saves items selected and purchased)

Four Major Components:

1. HTTP response Header `Set-cookie: header`
2. HTTP request `Cookie: header`
3. Cookie stored on client/user’s host (managed by web browser)
4. Cookie stored in back-end database on website (e.g. MySQL)

HTTP Headers – Cookies

client

server



cookie file



usual http request msg

usual http response
Set-cookie: 1678

usual http request msg
Cookie: 1678

usual http response msg

usual http request msg
Cookie: 1678

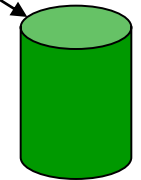
usual http response msg

Amazon server
creates ID
1678 for user

create
entry

cookie-
specific
action

cookie-
specific
action



backend
database

access

access

one week later:



HTTP Cookie attributes

```
Set-Cookie: value[; expires=date][; domain=domain][; path=path][; secure][; HttpOnly]
```

- Specify expiry time
 - Limit window of vulnerability against cookie theft and CSRF
- Specify scope of cookie
 - Domain = which sub-domains cookie is valid in
 - Path = which directory paths in domain cookie is valid in
- Specify security concerns
 - Secure = only send over HTTPS connections to avoid cookie theft
 - HttpOnly = only send within HTTP requests (restricts access via document.cookie in JavaScript to eliminate XSS cookie stealing)

```
Set-Cookie: SSID=Ap4P...GTEq; domain=foo.com; path=/; secure; HttpOnly
```

Sessions in cookies

- Web application frameworks typically assign identity via an opaque session within cookie
 - PHPSESSID=13Kn5Z6Uo4pH (PHP)
 - JSESSIONID=W7DPUBgh7KTM (Java server pages)

Issues with cookies

- Cookie tampering
 - Adversary subverts insecure cookie format to obtain elevated privileges (`natas`, `webpentestlab`)
 - Forges entire cookie to gain privileges
 - Solution: avoid encoding authorization level in cookie
 - Tamper with cookie given
 - Solution: use cryptographic hash to sign cookie

Authentication with HTTP and Forms


- Via GET (not recommended)
- Shows up in history, referer, & network



Username:

Password:

```
<html>
  [...]
<body>
  <form action="/login.php" method="GET">
    Username: <input type="text" name="username"> <br>
    Password: <input type="password" name="password"> <br>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```



```
GET /login.php?username=admin&password=admin HTTP/1.1
Host: vulnerable
User-Agent: Mozilla Firefox
```

Authentication with HTTP and Forms

- Via POST
- Shows up in network

Username:

Password:

```
<html>
  [...]
  <body>
    <form action="/login.php" method="POST">
      Username: <input type="text" name="username"> <br>
      Password: <input type="password" name="password"> <br>
      <input type="submit" value="Submit">
    </form>
  </body>
</html>
```

POST /login.php HTTP/1.1
Host: vulnerable
User-Agent: Mozilla Firefox
Content-Length: 35

username=admin&password=admin

Examples

- https://www.w3schools.com/TagS/att_form_method.asp
- To see the POST
 - Remove target="_blank"
 - Load developer tools
 - Make request
 - Highlight early part of timeline

Encoding

- Data encoding required between client and server
 - Special HTTP characters in URL or form data
 - Special HTML characters in web page content (HTML/CSS)

URL encoding for HTTP

- HTTP special characters
 - Request lines and fields delimited by newline, return, and space (`\r\n`).
 - URL path and parameter list separated by ‘?’
 - URL parameters separated by ‘&’
 - A parameter name and the corresponding value separated by ‘=’
- How can an application use these special characters in form data and URLs?
- URL-encoding
 - ‘%’ followed by hex ASCII code
 - %20 = space when not used in parameters
 - `https://oregonctf.org/x + y/`
 - https://www.w3schools.com/TagS/att_form_method.asp
 - Special characters in form data encoded in GET

HTML-encoding for web content

- Similarly, in HTML, how can special characters used in HTML such as '<' and '>' be included without triggering its semantic meaning?
 - Often critical in preventing cross-site scripting vulnerabilities
- HTML-encoding

>	>	
<	<	
&	&	
"	"e;	
'	'	(Decimal ASCII code 39)
=	=	(Hex ASCII code 3d)

Double encoding and decoding

- Filters that are used to sanitize user input, must take into account encoding
- Where does the decoding happen and can you ensure it only happens once?
- Often a source of bugs
 - Code to filter out '=' must correctly decode
 - Bugs introduced with multiple decodings
 - `natas`
 - Consider '='
 - URL-encoded once: `%3d`
 - URL-encoded twice: `%253d`

HTTPS

- Provides server authentication, integrity, and secrecy to client for web requests over network (more later)

Web Server

Web Server Directories | Web App
Frameworks

Web server

- Mostly Apache and nginx, but some others IIS
- All files, objects, and resources are stored here (HTML, JPG, txt files, video, audio, ...)
- No typical directory structure
- BUT there will be DocumentRoot directory for hosted environment. Some examples:

`/www/your-domain/html`

`/home/httpd/`

`/www/another-domain/html/cgi-bin/`

Apache/nginx – Directory Behavior

- How do files map to URLs?
 - <http://www.chi-ni.com/>
- Configuration files for sites in
 - `/etc/{nginx,apache2}/sites-available`
 - DocumentRoot points to location of site in file system
- Without a file specified, server looks for `index.html`, `index.htm`, `index.php`, or gives dir listing (if allowed)
 - <http://www.chi-ni.com/>
 - <http://www.chi-ni.com/Themes>

Nginx – Directory Behavior

- Examples
- Find the file <http://oregonctf.org/x + y/index.html>
- Which URL hits this file?

```
/var/www/html/oregonctf/ctf-  
practice/picoctf/2017/SecretTeamToken.txt
```

Web applications

- Apache not sufficient to build stateful web applications
- Rich web apps require a collection of multiple software components
 - Programming language/template framework
 - Persistent storage

Web programming languages

- Dynamic content requires a programming language to generate
- Example languages (frameworks and app. servers)
 - PHP
 - [natas](#), [picoCTF](#), [Web for Pentester #1](#)
 - Java (Tomcat, Struts)
 - [cs410 CTF](#)
 - Ruby (Rails)
 - [Web for Pentester #2](#)
 - Python (Django, Flask)
 - [cs201.oregonctf.org](#)
 - JavaScript (Node.js, Express, Angular)
 - [Full-stack course](#)
 - C# (ASP.NET)
 - Go

Web programming languages

- Choosing a language and framework
 - Many languages make it *easy* to produce vulnerable code
 - One version of a popular language

```
var_dump('0010e2' == '1e3');           bool(true)
var_dump('0x1234Ab' == '1193131');     bool(false)
var_dump('0xABCdef' == '      0xABCdef'); bool(false)
```

- Which language? Results vary based on version of PHP
<https://3v41.org/tT418>
- Many web applications are open-source with vulnerabilities exposed for all to see
- Exercises use mostly PHP and Javascript

PHP & Server – PHP Overview

- What is PHP?
 - Perl Hypertext Preprocessor
 - Server-side scripting language designed for web development
 - One of the first web programming languages
 - Developed quickly with the beginner in mind
 - Attempts to automatically “correct” perceived programming errors such as type mismatches
- Laden with security issues
 - Multitude of versions to address them
 - Not recommended

PHP

- PHP designed to keep running
 - “When faced with either doing something nonsensical or aborting with an error, it will do something nonsensical.”
- Automatic type conversion/coercion rather than error
 - Complex, unpredictable, weak typing
 - False converted to 0
 - `123 == "123foo"`
 - `"123" != "123foo"`
 - Type-juggling errors in PHP (`natas`)
- Unpredictable behavior of APIs
 - Based on `php.ini` and compile-time settings

<https://eev.ee/blog/2012/04/09/php-a-fractal-of-bad-design/>

<http://phpsadness.com/sad/47>

<https://www.reddit.com/r/lolphp/>

PHP & Server – PHP Syntax

- Needed for `natas`

- **Delimiters:**

- `<?php CODE ?>`

- `<?= CODE ?>`

- **Comments:**

- `//`

- `/* */` (multi-line)

- `#`

- **Variables:**

- `$variable = "Hello World"`

- **Functions:**

- `function hello($target='World')`

- `{`

- `echo "Hello " . $target . "!\n";`

- `}`

- `hello();`

Javascript

- Allows single language to run both on client and server
- Developer efficiency in language learning
- Most popular programming language
- But....also has quirks in its type coercion
 - <https://www.destroyallsoftware.com/talks/wat>
 - <https://www.destroyallsoftware.com/talks/the-birth-and-death-of-javascript> (0-9:20)

Persistent storage

- Per-user web application state
 - Relational databases PostgreSQL, MySQL, SQLite, Oracle, MS SQL
 - NoSQL (non-SQL): simple files, MongoDB, CouchDB, etc
 - Common in big data and real-time web applications that do not require transactional consistency
 - Directories like openLDAP or Active Directory
 - Often a target of attacks

Questions

- <https://sayat.me/wu4f>

Extra

Web ecosystems

- Specify the programming language and framework as well as the storage technology
- Initially “LAMP”
 - Persistent storage = MySQL
 - Programming language = PHP
- Diverse instances
 - OpenStack, Docker, Amazon EC2

Multi-part forms

```
<form action="/upload/example1.php" method="post" enctype="multipart/form-data">
  <p><input type="file" name="image">
  <p><input type="file" name="send">
  <p><button type="submit">Submit</button>
</form>
```

```
POST /upload/example1.php HTTP/1.1
Host: vulnerable
Content-Length: 305
User-Agent: Mozilla/5.0 [...] AppleWebKit
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryfLW6oGspQZKVxZjA

-----WebKitFormBoundaryfLW6oGspQZKVxZjA
Content-Disposition: form-data; name="image"; filename="myfile.html"
Content-Type: text/html

My file

-----WebKitFormBoundaryfLW6oGspQZKVxZjA
Content-Disposition: form-data; name="send"

Send file

-----WebKitFormBoundaryfLW6oGspQZKVxZjA--
```

Multi-part forms

- **Content-type header:**

```
Content-Type: multipart/form-data; boundary=-----  
WebKitFormBoundaryfLW6oGspQZKVxZjA.
```

- Long random string used to delimit parts
- String repeated for every part of the multipart information.
- The last part contains the string followed by --.