

# Intro to Web Security

Lecture 9 - ECS198F SQ26



# Plan of the Day



## Web Application Basics

- What goes on behind the scenes when you access a website?
- What are the parts of a web application and how do they interact?

## Web Pentest Tools

- What kind of tools exist for testing web application security?
- What vulnerabilities might these tools exploit?

## Basic Web Pentest

- Introduce the tools alongside some representative challenges



# **1. Web Application Basics**

# What Goes On Behind The Scenes?



*Question: What goes on when you connect to a website?*

Let's pull up <http://example.com>!

Relevant Findings?

- HTTP used
  - What's sent OUT?
  - What's sent BACK?

The screenshot displays the network tab of a browser's developer tools. The selected request is a GET to `http://example.com/`. The response headers are expanded, showing the following details:

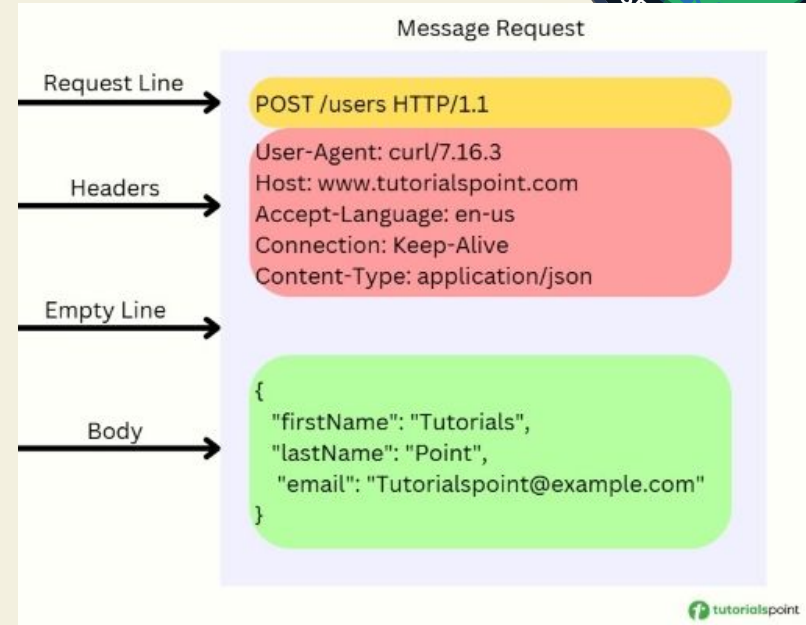
- Status: 200 OK
- Version: HTTP/1.1
- Transferred: 676 B (528 B size)
- Request Priority: Highest
- DNS Resolution: System
- Response Headers (305 B):
  - Age: 14118
  - Allow: GET, HEAD
  - cf-cache-status: HIT
  - CF-RAY: 9f3783a1ffb732bc-SMF
  - Connection: keep-alive
  - Content-Encoding: gzip
  - Content-Type: text/html
  - Date: Tue, 28 Apr 2026 16:41:23 GMT
  - Last-Modified: Sat, 25 Apr 2026 11:01:06 GMT
  - Server: cloudflare
  - Transfer-Encoding: chunked
- Request Headers (392 B):
  - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8
  - Accept-Encoding: gzip, deflate
  - Accept-Language: en-US,en;q=0.9
  - Cache-Control: no-cache
  - Connection: keep-alive
  - DNT: 1
  - Host: example.com
  - Pragma: no-cache
  - Priority: u=0, i
  - Upgrade-Insecure-Requests: 1
  - User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:150.0) Gecko/20100101 Firefox/150.0

# Breaking Down HTTP



Basic system of requests/responses to interact with a server

- Request:
  - Method + Resource + Protocol
  - <https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods>
- Headers
  - Send extra data about the host
- Body
  - If you need to send stuff to application

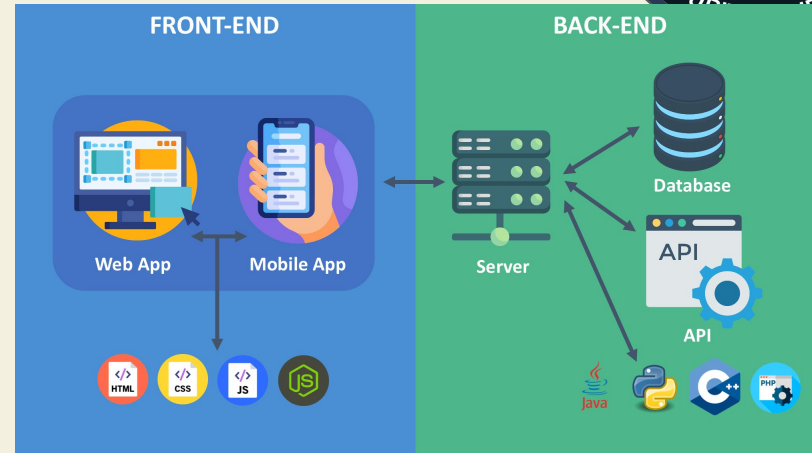


<https://www.tutorialspoint.com/http/images/http-message-request.jpg>

# Bringing in the Backend



- Just HTML and Javascript is fine if you're serving a mostly-static website...
  - But what if you need the server to process some data for you?
    - Can't do it in Javascript, not "part of the website"
  - You need a way to "pass data" to the server!



[https://miro.medium.com/0\\*RfvInMt7Z1TSCa8N](https://miro.medium.com/0*RfvInMt7Z1TSCa8N)

# Application Programming Interface



## Application Programming Interface (API)

- Rules/Protocol specification for how to interact with the application over HTTP!
- Same data can be passed in different “formats”

## Example API Specification:

<https://editor.swagger.io/>

A screenshot of a Swagger API specification for FastAPI. The interface shows the following endpoints:

- wakeups**
  - GET /api/v1/wakeups/ Get Wakeups
  - POST /api/v1/wakeups/ Create Wake
  - GET /api/v1/wakeups/{wakeuper\_id} Get Wakeup
  - PUT /api/v1/wakeups/{wakeuper\_id} Update Wakeup
  - DELETE /api/v1/wakeups/{wakeuper\_id} Delete Wakeup
- media**
  - GET /api/v1/media/{wakeuper\_id}/image Get Image
  - GET /api/v1/media/{wakeuper\_id}/sound Get Sound
- Schemas**
  - HTTPValidationError > Expand all object
  - ValidationError > Expand all object

# Storage

So you have something that can run programs...what if you want to store things?

- Remember that webapps have to “scale!”
  - Support millions of operations per second, handle race conditions, etc.
- So there’s usually a database “storing” things!
  - Most common is SQL-based



Amazon S3



# The “Full Stack”

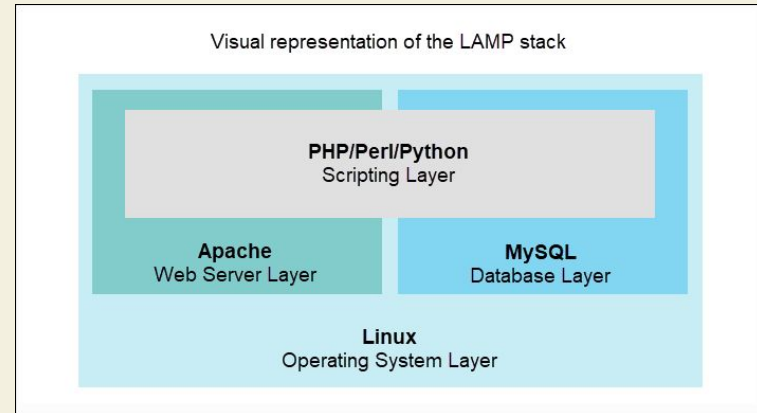


“Full Stack Development” - Being able to handle all the technology needed to run a web application!

- “Frontend” (website)
- “Backend” (application + databases)

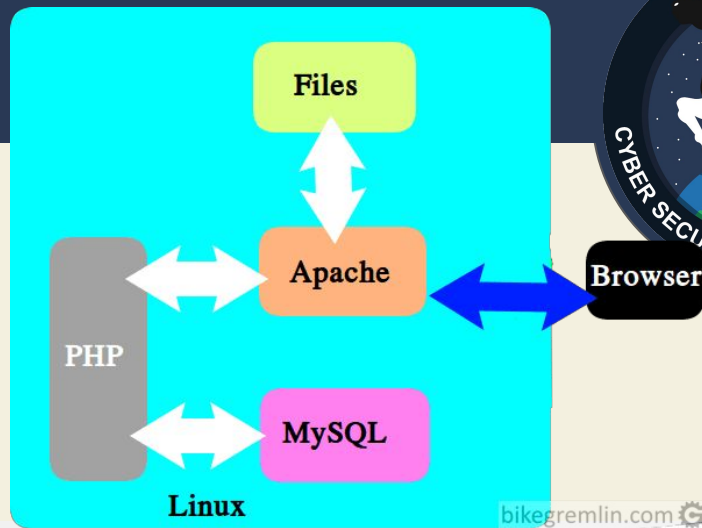
“Technology Stack” - The specific software you choose to implement your website features with

- EX: LAMP Stack



# Why This Matters?

- Each of these components are a way you can compromise a web server!
  - Web server / application vulns, database vulns, insecure API, etc.
  - The ways these components interact with each other can be insecure!



## Error establishing a database connection

This either means that the username and password information in your `wp-config.php` file is incorrect or we can't contact the database server at `localhost`. This could mean your host's database server is down.

- Are you sure you have the correct username and password?
- Are you sure that you have typed the correct hostname?
- Are you sure that the database server is running?

If you're unsure what these terms mean you should probably contact your host. If you still need help you can always visit the [WordPress Support Forums](#).



## **2. Web Pentesting Tools**

# The Obvious Tool...



## Browser!

- You already use it to interact with websites so much, it's a good first step for web pentesting!

<https://play.picoctf.org/practice/challenge/66>

<https://play.picoctf.org/practice/challenge/46>

## Tips:

- Make sure to understand what the processes do!
- Check everything!!

# Catch and Release



**Burp Suite** - Industry standard tool for web penetration testing!

- Captures incoming and outgoing web traffic
- Allows you to modify the request into a custom payload
- Send the modified traffic out as much as you need!



<https://play.picoctf.org/practice/challenge/419>

<https://play.picoctf.org/practice/challenge/520>

# Leave No Stone Unturned



Sometimes, you just need to brute force a solution.

**Problem:** There may be “hidden” sites/folders/API endpoints on the website that might not be findable by clicking all the links.

**Solution:** Try a bunch of common directory names to find hidden folders.

Dirbuster, gobuster, ffuf, etc. *combined* with a wordlist!

<https://tryhackme.com/room/mkingdom>

WARNING: Bruteforcing is essentially one-step removed from a DDoS. Brute Force with Care!