Intro to Web (Security)

~ Library of Babel

Overview

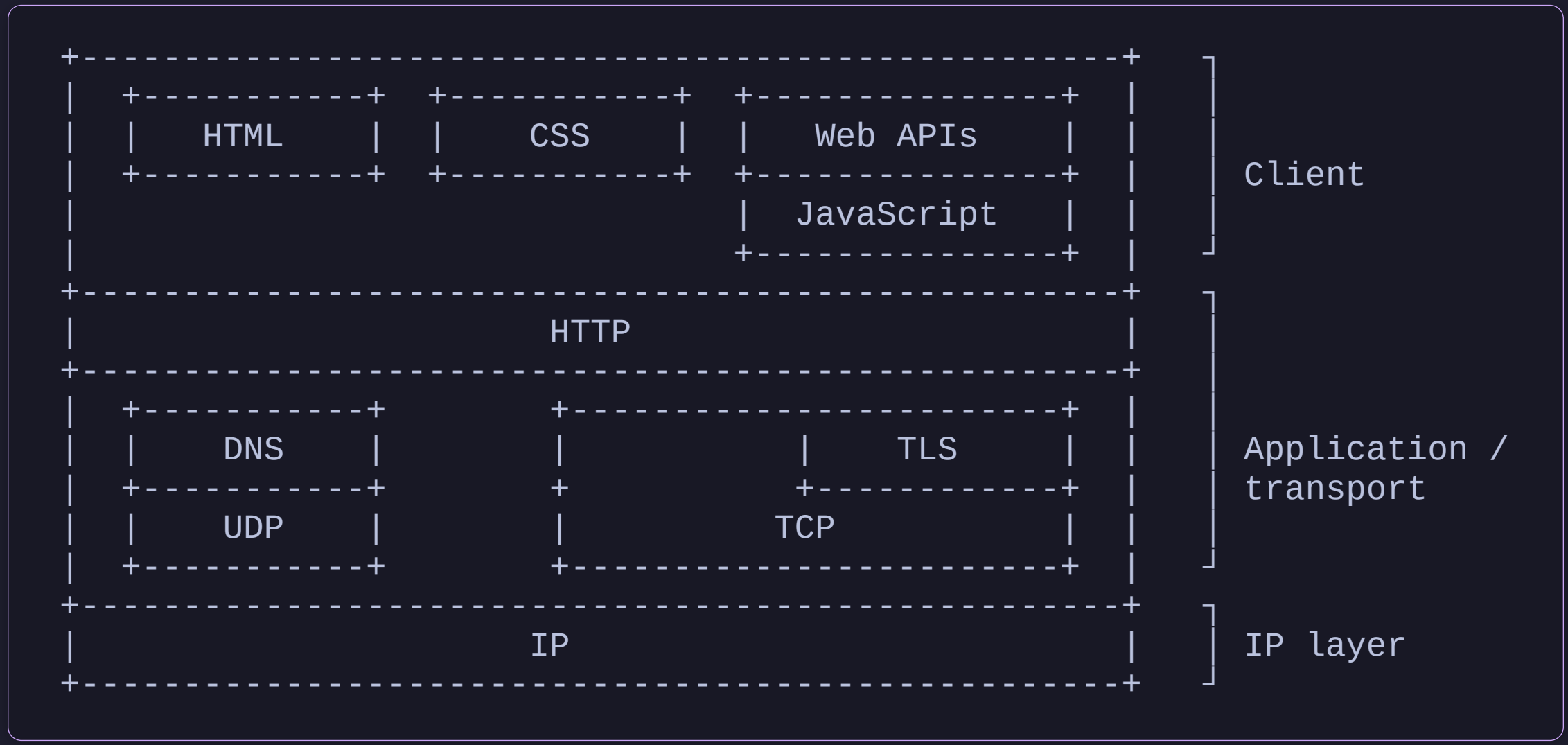
0. Anatomy of a URL
1. Dissecting HTTP Request/Response
2. Client-side is Open Source
3. Probing and Fingerprinting
    - Port Scanning
    - Banner Grabbing
4. Places to Look
5. Cross Site Scripting (XSS)
    - Content Security Policy (CSP)
6. Data Exfiltration & Webhooks

Anatomy of a URL

```
http://www.libbabel.so:80/blog?sortBy=time#H7texFinals
```

- `http://` protocol
- `www.libbabel.so` domain
- `:80` port number (standard for http)
- `/blog` path
- `?sortBy=time` query param
- `#H7texFinals` fragment

Hierarchy of the Web (Simplified)



HTTP: Hypertext Transfer Protocol

- Protocol for transmitting resources
- Follows classic *client-server* model
- *Stateless protocol*
- *Unencrypted* by default

HTTP Request

```
GET / HTTP/2
Host: example.com
User-Agent: Mozilla/5.0 Gecko/20100101 Firefox/141.0
Accept: text/html,application/xhtml+xml
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br, zstd
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
Priority: u=0, i
```

Request Methods / Verbs

- Indicate purpose of request
  - `GET` fetch resource
  - `HEAD` similar to GET but without a response body
  - `POST` submit an entity
  - `PUT` replaces an entity
  - `DELETE` delete a resource
  - `PATCH` applies partial modifications
- Others: `CONNECT`, `TRACE`

HTTP Headers

- Let client and server pass extra information
  - Request Headers
  - Response Headers
- Also handle redirections
- Standard format: `HEADER_NAME: HEADER_VALUE`

HTTP Cookies

- Small piece of data stored on the browser
- Set using `Set-Cookie` header
- Browser sends using `Cookie` header

`Authorization` Request Header

- Used to provide credentials for authentication
- Usually uses Session/Persistent Tokens

Example (discord):

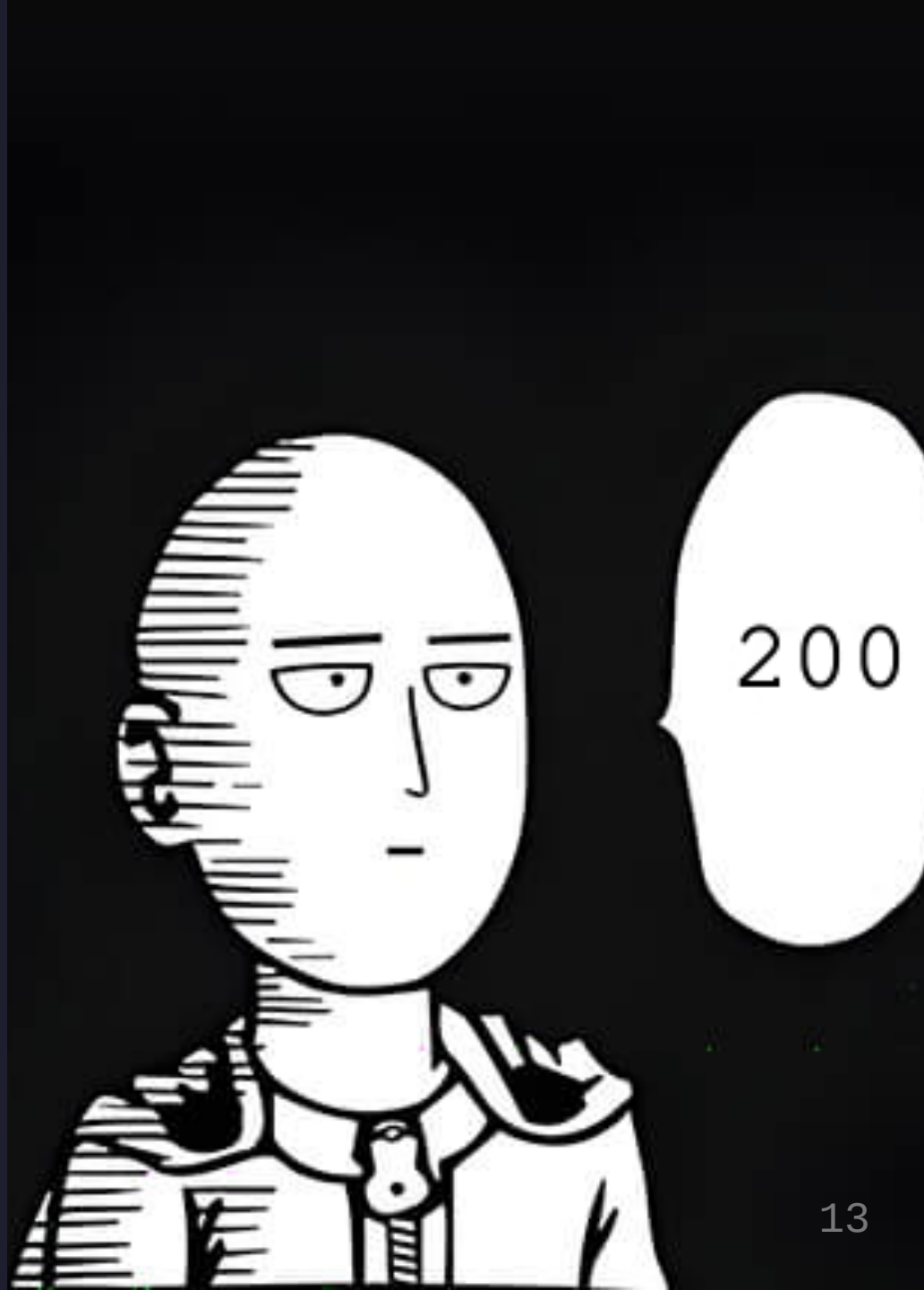
```
POST /api/v9/channels/<Friend'sID>/messages HTTP/3
Host: discord.com
User-Agent: <My Browser's UA>
Accept: */*
Content-Type: application/json
Authorization: <My Discord Token>
Content-Length: 100
Origin: https://discord.com
Alt-Used: discord.com
Connection: keep-alive
Referer: https://discord.com/channels/@me/<Friend's Id>
Cookie: <My Cookies>

{"content":":(","nonce":"<Nonce>","tts":false,"flags":0}
```

HTTP Response

```
HTTP/2 200
date: Wed, 07 Jan 2026 16:22:17 GMT
content-type: text/html
content-encoding: gzip
last-modified: Mon, 05 Jan 2026 20:20:37 GMT
allow: GET, HEAD
age: 8024
cf-cache-status: HIT
vary: Accept-Encoding
server: cloudflare
cf-ray: 9ba4cb05ee21a346-DEL
X-Firefox-Spdy: h2
```

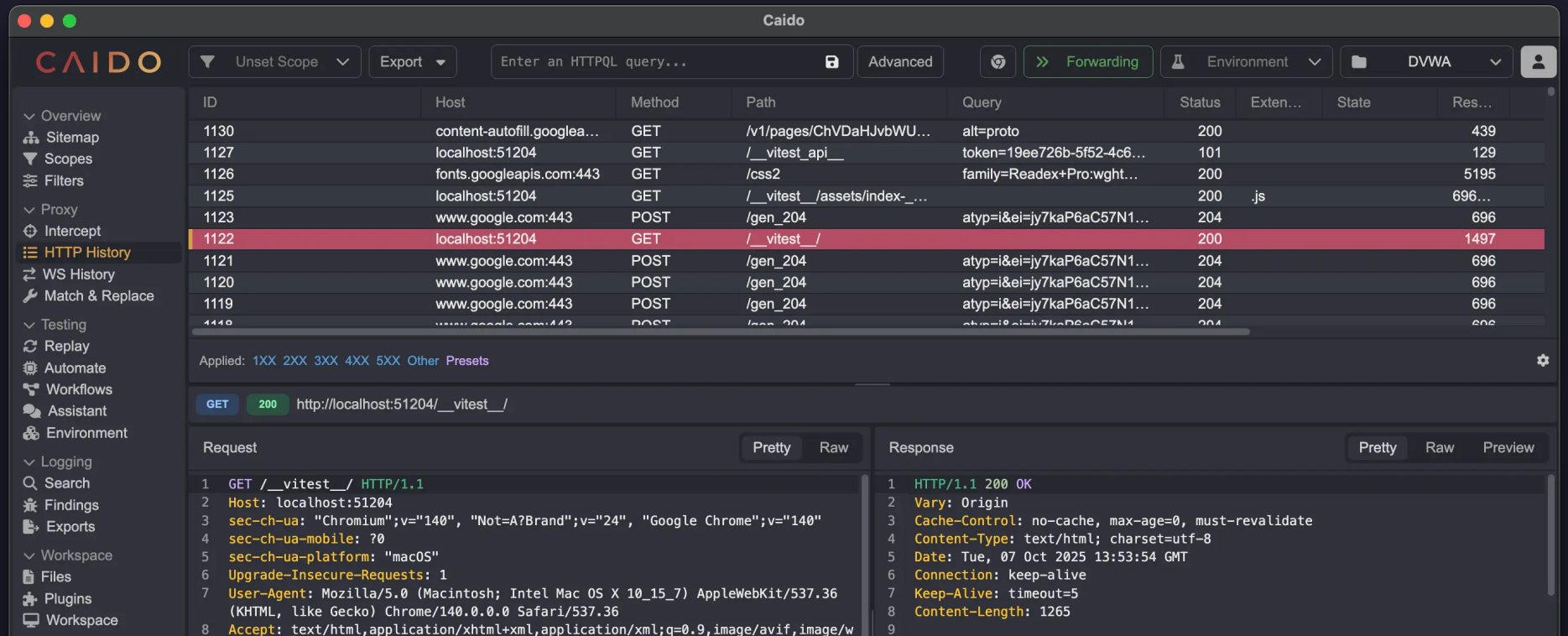
HTTP Response Status Codes

- Standard response type
- `100-199` Information responses
- `200-299` Successful responses
- `300-399` Redirection responses
- `400-499` Client Error responses
- `500-599` Server Error responses
- Common: `200` (OK), `404` (Not Found), `504` (Gateway Timeout)



Intercepting & Modifying Requests

- Conveniently interfere in the client-server interaction
  - PortSwigger BurpSuite
  - Caido



Client-side Web is Open

- Browser Dev Tools leak all client-side logic
- JS can be *obfuscated* but not *hidden*
- WASM binaries can be reverse-engineered.

> (ɾ⁼ _•́) The internet is broken beyond repair

Fingerprinting Methods

- Sniff info about the running services, versions, tech stack, etc.
- Use this info to pinpoint vulnerabilities and attack vectors.

Port Scanning

- Probe a server for open ports and running services
- Automated using tools like `nmap` or `zenmap` (gui)

> (ר ͡° _ •́) How many ports do you think there are?

Banner Grabbing

- Short piece of text/metadata sent by server when you connect
- May reveal service name, versions, OS config info

Git Leaks

- Exposed `.git/` directories
- Can lead to total reconstruction of the git commit history
- Automated using tools like `git-dumper`

> (͡ಠ _ ʖ̯•́) How bad do you think this is?

Those Who Came Before

A grave sin. A cover up. A forgotten trail.

```
<HOST>:<PORT>
```

> flag format: babel{[a-zA-Z0-9_]}

Where to Look?

- You can find interesting stuff in a bunch of places
- A somewhat exhaustive list:

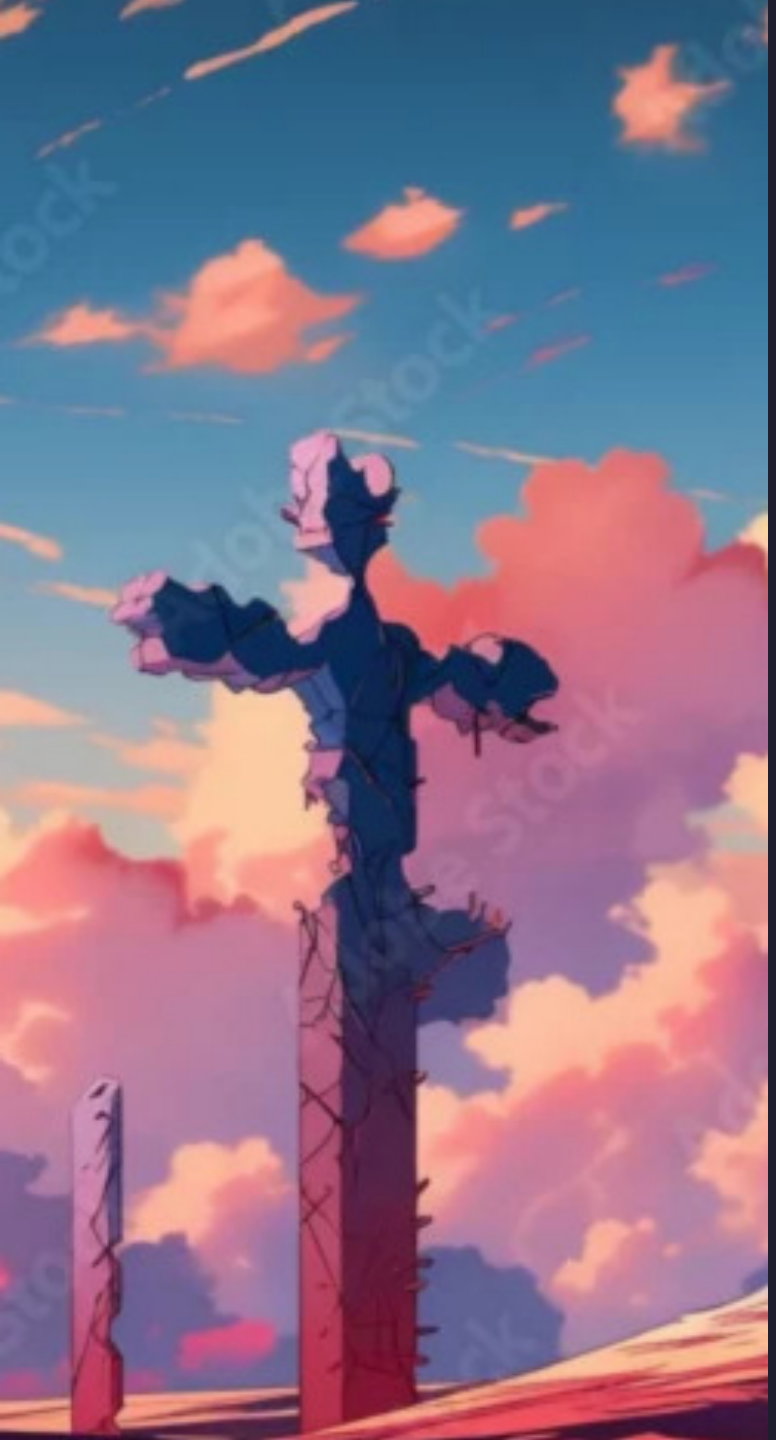
`comments`
`cookies`
`local storage`
`session storage`
`HTTP headers`
`/robots.txt`
`DNS records`

Pacman's Revenge

The ghosts have hidden themselves using mysterious hexes. Think you can find them all?

`<HOST>:<PORT>`

> flag format: babel{[a-zA-Z0-9_]}



Cross Site Scripting (XSS)

- If user input is not *sanitized* properly, it could lead to unwanted resource injection
- `DOM XSS` entirely client-side
- `Reflected XSS` non-persistent
- `Stored XSS` persistent

Example

```
// script.js
const paramsString = window.location.search;
const searchParams = new URLSearchParams(paramsString) ;
const name = searchParams.get('name') || 'Jane Doe';
document.getElementById('name').innerHTML = name;
```

```
<!-- index.html -->
<div>Hi <span id="name"></span></div>
```

> (╭ರ_•́) What happens when you visit `/?name=<script>alert(1)</script>`

> `innerHTML` is very dangerous (╥﹏╥)

Content Security Policy (CSP)

Work in Progress

Thank You