Frontend Security Stuff

Cross site scripting (XSS)

Cross-Site Scripting (XSS) is a type of **security vulnerability** commonly found in web applications. It allows attackers to **inject malicious scripts** (usually JavaScript) into webpages viewed by other users.

 How to Prevent XSS

1. **Escape Output**
    - Escape HTML, JavaScript, and URLs properly.
    - Use libraries like DOMPurify to sanitize inputs.
1. **Content Security Policy (CSP)**
    - Restrict where scripts can be loaded from.
    - Example: Block inline scripts using CSP headers.
1. **Use Framework Features**
    - React, Angular, and others automatically escape values unless you use dangerous methods (dangerouslySetInnerHTML in React).
1. **Validate Input**
    - Validate and sanitize all user inputs, especially if rendering them later.
1. **Avoid eval() and innerHTML**
    - These APIs are dangerous when used with dynamic content.

Ddos Attack

This is not the clinet side sequrity, but still we can discuss with the interviewer.

Denial of Service (DoS) and **Distributed Denial of Service (DDoS)** are attacks aimed at **disrupting the availability** of a system, service, or network. The goal is to **overwhelm the target** so legitimate users can't access it.

CORS Error

CORS (Cross-Origin Resource Sharing) is a security feature implemented by web browsers that controls **how resources (APIs, fonts, etc.) on one origin** (domain) can be **accessed by another origin**.

□ It's designed to **prevent malicious websites** from reading sensitive data from another site via JavaScript.

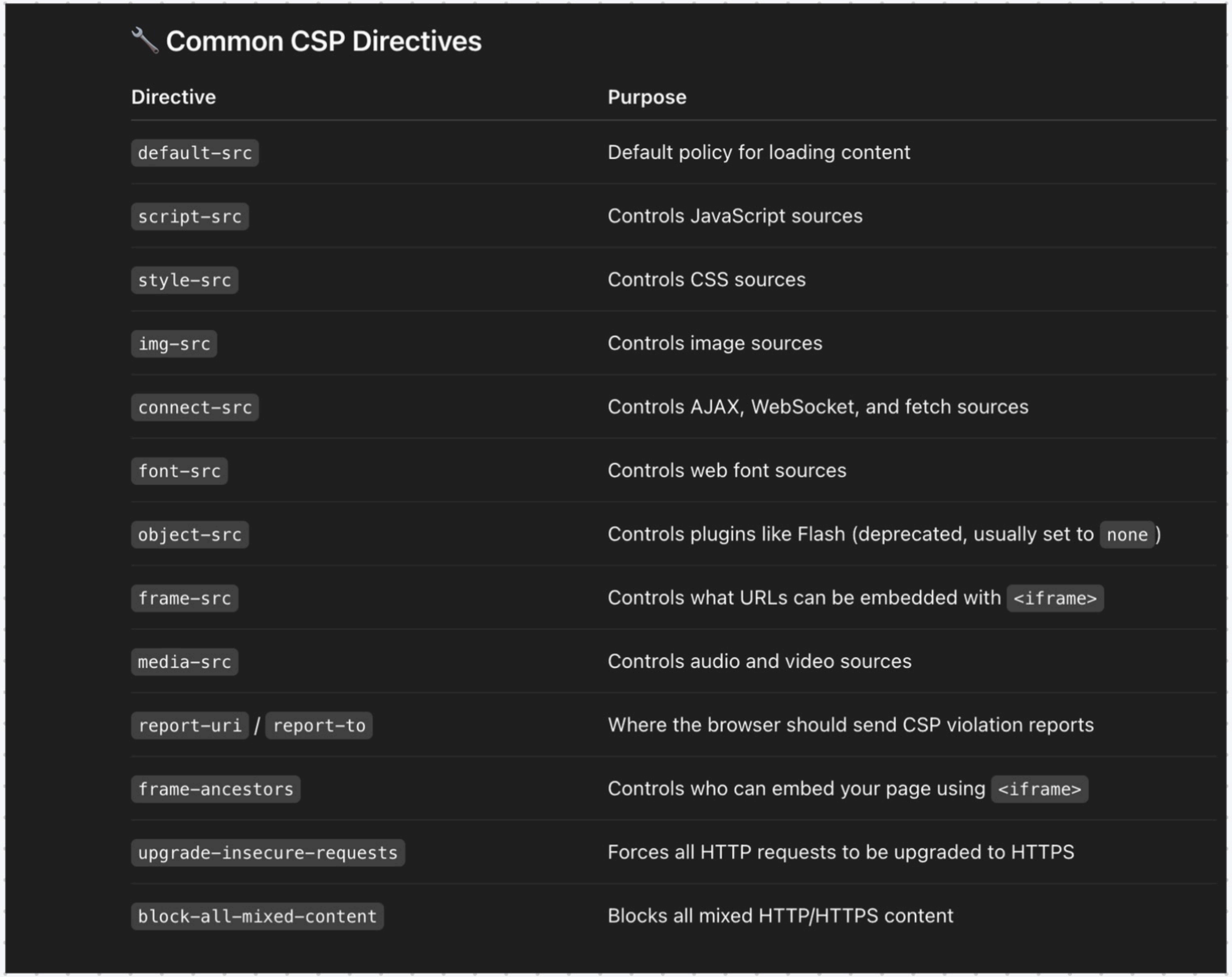
Man-in-the-Middle (MitM) Attacks

A **Man-in-the-Middle (MitM)** attack is a type of cyberattack where a malicious actor **intercepts communication between two parties** (like your browser and a website) without either party knowing.

□ The goal is often to **eavesdrop**, **steal sensitive information** (like login credentials), or **manipulate data** in transit.

Content Security Policy

Content Security Policy (CSP) is a powerful browser security feature that helps prevent **Cross-Site Scripting (XSS)**, **clickjacking**, and other **code injection attacks** by controlling **which content sources are allowed to load** on your web page.



🔧 Common CSP Directives

Directive	Purpose
`default-src`	Default policy for loading content
`script-src`	Controls JavaScript sources
`style-src`	Controls CSS sources
`img-src`	Controls image sources
`connect-src`	Controls AJAX, WebSocket, and fetch sources
`font-src`	Controls web font sources
`object-src`	Controls plugins like Flash (deprecated, usually set to `none`)
`frame-src`	Controls what URLs can be embedded with `<iframe>`
`media-src`	Controls audio and video sources
`report-uri` / `report-to`	Where the browser should send CSP violation reports
`frame-ancestors`	Controls who can embed your page using `<iframe>`
`upgrade-insecure-requests`	Forces all HTTP requests to be upgraded to HTTPS
`block-all-mixed-content`	Blocks all mixed HTTP/HTTPS content