SSD Report

Name: Aymen keskas Date: 13/05/2024

Side Channel Attacks

Side-channel attacks exploit flaws in hardware or physical conditions that cause unintended data leakage. Well-known examples include Meltdown and Spectre, which exploit speculative execution in modern CPUs. Mitigating side-channel attacks often requires patching hardware vulnerabilities or replacing old hardware.

XSS (Cross-Site Scripting)

There are three main types of XSS, each with its own mitigation strategies:

- **DOM-based XSS**: Use `textContent` instead of `innerHTML` when handling user input in JavaScript, or employ a secure JavaScript framework.
- **Reflected & Stored XSS**: Implement a Content Security Policy (CSP) to restrict loading and executing external scripts. Use an Integrity Security Policy for loaded content. Consider building a Single Page Application (SPA) or Server-Side Rendered (SSR) application with frameworks like Next.js, Nuxt.js, or SvelteKit Instead of an MPA (laravel,django,spring).

```
<meta http-equiv="Content-Security-Policy"
content="script-src 'self' https://cdn.example.com;">
<script src="https://example.com/script.js"
integrity="sha384-oqVuAfXRKap7uxy9rx7HNQlGYl1kPzQho1wx4JwY8wC" crossorigin="anonymous"></scr
```

CSRF (Cross-Site Request Forgery)

- **Use SameSite Cookie Attribute**: Set the `SameSite` attribute to `Lax` or `Strict` to prevent the browser from sending cookies on cross-origin requests. Be cautious with `Lax` as it still allows cookies on GET requests.
- **Use a Middleware**: Implement a middleware that compares the `Origin` and `Host` headers to block cross-origin requests.
- **Use CSRF Tokens**: Generate and validate CSRF tokens for each request, typically stored in the client-side JavaScript runtime.

SQL Injection

- **Use an ORM**: Employ an Object-Relational Mapping (ORM) library, which sanitizes input and uses prepared statements.

HTTP Request Smuggling

- **Synchronize Proxy and Server**: Ensure that your proxy and server are correctly parsing and forwarding HTTP requests.
- **Use HTTP/2**: While not a complete solution, HTTP/2's use of streams can help mitigate HTTP request smuggling.