**1. SQL Injection (SQLi)**

SQL Injection is one of the most common and dangerous web vulnerabilities. It happens when a website allows users to input data into a form or search box, but doesn't properly sanitize that input. A hacker can manipulate this input by entering specially crafted SQL commands that are then executed by the database. So SQL Injection is like tricking the website into giving up its secrets. A vulnerable website might allow a hacker to view or delete sensitive data from the website’s database, or even gain full control over it which is always disastreous mostly In 2008, **Heartland Payment Systems**, a company that processed millions of credit card transactions, was hacked through SQL Injection. The attack resulted in the theft of over 100 million credit card numbers. This breach remains one of the biggest data theft incidents in history. Similarly, a few years ago, **KCB** faced a similar threat in Kiambu when attackers exploited an SQL Injection vulnerability. These types of attacks are more common than we might think, and they can have severe consequences.

**How to Prevent It:** To prevent SQL Injection, developers should use **prepared statements** with placeholders to safely handle user input, instead of directly inserting it into SQL queries. Prepared statements ensure that user input is treated as data, not executable code. Additionally, it’s important to **validate user input** before using it. Never trust user data Also, **limit database access** and avoid giving users too much control. If your website doesn’t need to delete or modify data, don’t give it permission to do so. SQL Injection attacks can be prevented easily if the website follows these best practices. It’s a simple fix that can save a lot of headaches down the road for everybody

**2. Cross-Site Scripting (XSS)**

Cross-Site Scripting, or XSS, occurs when a hacker injects malicious scripts (often JavaScript) into a website that other users will then load. This means that when someone visits the website, the malicious script runs in their browser, not on the website’s server. The hacker can use this to steal cookies, session tokens, or redirect users to malicious websites. It’s one of the most common ways attackers gain unauthorized access to users’ sensitive information.Its happens so much you wouldn’t even see it coming honestly In 2011, **Apache Tomcat**, a popular web server, had a vulnerability that allowed hackers to inject XSS code. By exploiting this vulnerability, attackers could steal session cookies from users visiting websites hosted on

Tomcat, leading to unauthorized access. The XSS vulnerability also affected several big companies, causing a lot of trouble for their users.

There are different types of XSS attacks that are kinda common

 **Stored XSS**: The malicious script is stored on the server and served to users every time they visit the page.

 **Reflected XSS**: The malicious script is included in a URL or form submission, and it is executed immediately after the page is loaded.

 **DOM-based XSS**: The attack is carried out in the browser, exploiting vulnerabilities in how the website’s JavaScript code manipulates the Document Object Model (DOM).

**How to Prevent It:** To protect your website from XSS attacks, it’s crucial to **sanitize user input** by removing or escaping any potentially dangerous characters (such as <, >, &, etc.). A good **Content Security Policy (CSP)** can help prevent the loading of untrusted scripts. Another effective method is to **avoid inline JavaScript** and instead place all scripts in external files, so that the website can more easily control what gets executed.

**3. Cross-Site Request Forgery (CSRF)**

Cross-Site Request Forgery, or CSRF, this is always lethal. It is an attack that tricks users into performing actions on a website that they didn’t intend. Essentially, it forces the user’s browser to make a request to a website where they are already logged in, causing them to perform an action without their knowledge or consent. It’s like a hacker sending a secret command to a website, making it seem like the legitimate user is the one sending the request.

In 2008, **Netflix** had a CSRF vulnerability that allowed attackers to change user settings without the users’ knowledge. By tricking users into clicking a link, an attacker could change their account settings, such as their email address or subscription plan. This was not only inconvenient, but also dangerous

**How to Prevent It:** To defend against CSRF attacks, websites should implement **CSRF tokens**. These are random, unique values added to forms that ensure the request is coming from a legitimate source. Websites should also set **SameSite cookies**, which prevent cookies from being sent on cross-site requests. Lastly, for sensitive actions like password changes, **re-authentication** can ensure that the user is performing the action intentionally.

**4. Session Hijacking**

Session Hijacking is a serious threat that occurs when an attacker steals a valid session ID, usually through **man-in-the-middle** attacks or other vulnerabilities. Once the hacker has the session ID, they can impersonate the legitimate user and gain access to their account without needing to log in.

In 2010, a Firefox extension called **Firesheep** made it ridiculously easy for anyone to hijack Facebook and Twitter sessions on public Wi-Fi. All an attacker had to do was install Firesheep, and they could intercept unencrypted session cookies of other users, hijacking their accounts.

**How to Prevent It:** To prevent session hijacking, always use **HTTPS** for encrypted communication between the user’s browser and the server. Set **Secure** and **HttpOnly** flags on cookies to ensure that session cookies are only sent over secure connections and are not accessible via JavaScript. Additionally, **regenerate session IDs** after login and important changes to prevent session fixation. Websites should also **auto-logout** users after a certain

period of inactivity to reduce the risk.

**5. Man-in-the-Middle (MITM) Attacks**

A **Man-in-the-Middle (MITM) attack** occurs when a hacker secretly intercepts and potentially alters communication between a user and a website. This attack usually happens on unsecured networks, like public Wi-Fi, where an attacker can listen in on data sent between the user and the website.

In 2017, **Equifax**, a major credit reporting agency, faced a significant security issue when their SSL certificate expired, leaving their website vulnerable to MITM attacks. This was just before they suffered one of the largest data breaches in history, exposing the personal information of millions of people.

**How to Prevent It:** The most effective way to protect against MITM attacks is to always use **HTTPS** across the entire website. **HSTS (HTTP Strict Transport Security)** can also be used to tell the browser to only use secure connections. Avoid using **public Wi-Fi** for sensitive activities, and consider using a **VPN (Virtual Private Network)** for an additional layer of encryption.