* **XSS-Lab**

Khi người dùng nhập từ khóa vào input term (qua query string ?term=...), giá trị này được chèn trực tiếp vào HTML trả về:

A black background with white text

AI-generated content may be incorrect.

* Đây là phần khung HTML tĩnh, bắt đầu một đoạn <p>.
* Biến term Giá trị lấy trực tiếp từ HTTP request param ?term=....
* input do user kiểm soát.
* Không hề được sanitize (lọc ký tự nguy hiểm) hoặc encode trước khi chèn vào HTML.
* **Reflected-DOM-XSS**

Trong file DomXssController.java:

* Server chỉ render một trang HTML có input form.
* Giá trị người dùng nhập vào ?msg=... không được xử lý ở server mà xử lý trực tiếp bằng JavaScript trên client.
* Đoạn quan trọng:

A computer screen with text

AI-generated content may be incorrect.

Lấy input từ URL

const msg = params.get('msg');

* Biến msg nhận trực tiếp giá trị query string ?msg=....
* Đây là dữ liệu do người dùng/attacker kiểm soát.

Đưa thẳng vào DOM qua innerHTML document.getElementById('output').innerHTML = 'You submitted: ' + msg;

* .innerHTML sẽ parse chuỗi như HTML/JS.
* Nếu msg chứa thẻ <script> hoặc thẻ hợp lệ khác, trình duyệt sẽ chèn vào DOM và thực thi script.
* Không có bước encode/sanitize nào trước khi gán.
* Ví dụ tấn công Request:

/domxss?msg=<script>alert(1)</script>

* Kết quả HTML:

<div id="output">You submitted: <script>alert(1)</script></div>

* Trình duyệt parse & chạy alert(1)..
* **Storexss-Lab**

1. Chức năng chính

App cho phép user submit một link với:

* description (text hiển thị cho link).
* url (giá trị đặt trong thuộc tính href).

Dữ liệu được lưu vào bảng links trong DB.

Khi render lại trang, toàn bộ link trong DB được hiển thị ra HTML:

A screen shot of a computer code

AI-generated content may be incorrect.

1. Các biện pháp "escape"

* safeDesc = escapeHtml(desc) → mô tả đã được encode → an toàn.
* hrefValue = encodeDoubleQuotes(url) → chỉ thay " thành &quot;, còn lại giữ nguyên.

1. Vấn đề ở href

* Thuộc tính href chỉ được encode dấu ".
* Không hề kiểm soát schema/protocol (http, https).
* Nếu attacker nhập URL = javascript:alert(1) → vẫn được chèn vào href.

Khi user click link → browser sẽ chạy JavaScript.

* **Store-xss-Entity**

1. Luồng xử lý

* Người dùng nhập author và message vào form /comment.
* Trước khi insert DB → gọi escapeHigh(message) và escapeHigh(author).
* Sau đó, khi render lại trong bảng <td class='render-me'>...</td> thì không encode thêm.
* Đoạn JavaScript phía client:

c.innerHTML = c.innerText;

* Biến nội dung text an toàn thành HTML có thể render.

1. Hàm escapeHigh

A computer screen shot of a program code

AI-generated content may be incorrect.

* Hàm này cố tình chặn một số tag phổ biến (script, iframe, svg).
* Các ký tự < > còn lại thì escape thành &lt;, &gt;.
* Không escape dấu & → để giữ nguyên entity.

Ví dụ: user nhập <b>hi</b> → lưu thành &lt;b&gt;hi&lt;/b&gt;.

3. Đoạn JavaScript phía client

c.innerHTML = c.innerText;

Ban đầu <td class='render-me'> chứa text như:

&lt;img src=x onerror=alert(1)&gt;

.innerText lấy ra chuỗi raw: &lt;img src=x onerror=alert(1)&gt;.

Sau đó gán lại vào .innerHTML → trình duyệt parse chuỗi này thành thẻ thật <img src=x onerror=alert(1)>.

Kết quả: payload thực thi → Stored XSS.

* **XEE**

1. Endpoint và luồng xử lý

* API /stock nhận dữ liệu từ POST request với header Content-Type: application/xml.
* Payload XML sẽ được parse bằng DocumentBuilderFactory trong Java.

1. Cấu hình Parser Trong code:

A screen shot of a computer

AI-generated content may be incorrect.

* disallow-doctype-decl = false → cho phép khai báo <!DOCTYPE ...>.
* external-general-entities = true → cho phép entity tham chiếu đến nguồn bên ngoài (file, URL).
* external-parameter-entities = true → cho phép xử lý external parameter entity.
* expandEntityReferences = true → tự động thay thế các entity trong quá trình parse. Toàn bộ cấu hình này biến parser thành mục tiêu dễ bị XXE.

1. Đầu ra (Response)

Sau khi parse XML, code lấy nội dung trong <productId> và đưa thẳng vào response:

return "Invalid product ID: " + productIdText;

Điều này có nghĩa là bất kỳ giá trị nào attacker inject vào productId (bao gồm entity expansion) sẽ được phản hồi trực tiếp → tạo thành kênh lộ dữ liệu.

* **SQL-Lab**
* Endpoint /products nhận tham số GET category.
* Code xây dựng query SQL như sau:

A screen shot of a computer

AI-generated content may be incorrect.

Nối chuỗi trực tiếp với input người dùng (category)

* Biến category lấy từ tham số HTTP request (/products?category=...).
* Input này không có bất kỳ kiểm tra hay lọc nào (no sanitization, no escaping).
* Khi nối thẳng vào chuỗi SQL, attacker có thể chèn vào đó các ký tự SQL đặc biệt (', ", --, ;) để thay đổi logic query.

Đây là SQL Injection classic.