# XSS Attack Demonstration – Cyber Security Project

## ⚠️ Overview

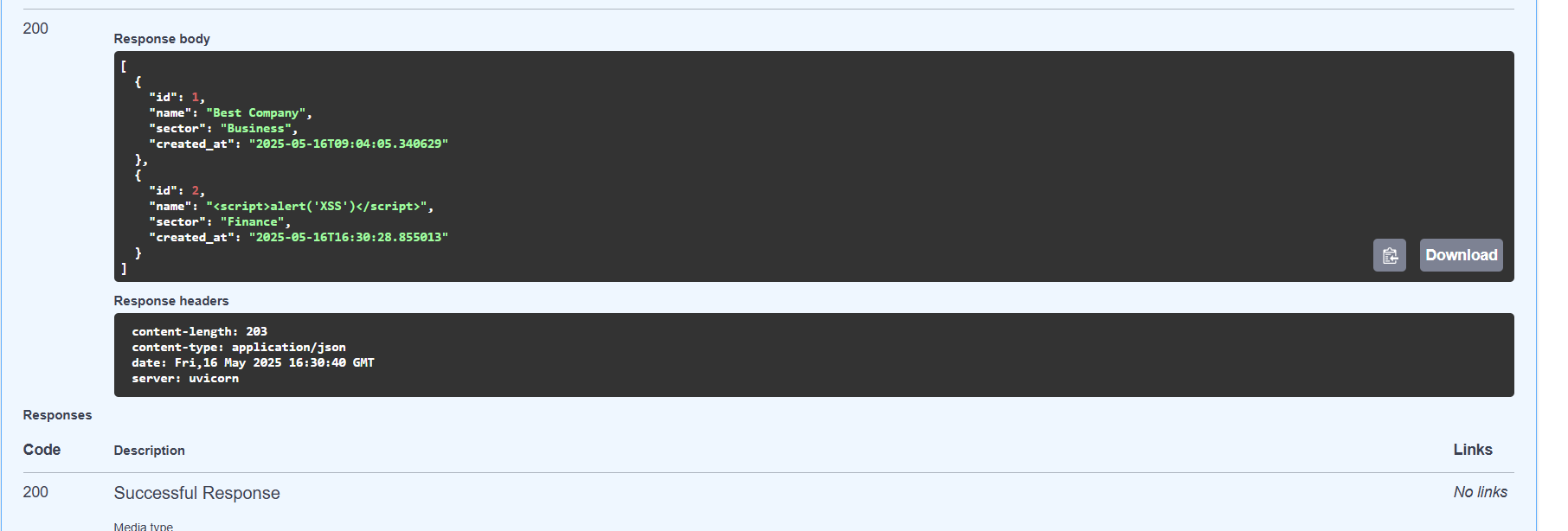
This document demonstrates how a stored XSS (Cross-Site Scripting) attack can be performed and prevented.  
Unsanitized user input is stored in the database and rendered on the client-side using innerHTML, which leads to JavaScript code execution in the browser.

## 📍 Stored XSS with <script>

Malicious input submitted via API:

<script>alert('XSS')</script>

When the client list is fetched and rendered directly in the DOM, the browser interprets the script tag and triggers the alert.



## ✅ Escaped Output – Protected Version

In the secured version, the input is escaped before being returned. Example output:

{"name": "&lt;script&gt;alert('XSS')&lt;/script&gt;", "sector": "Finance"}

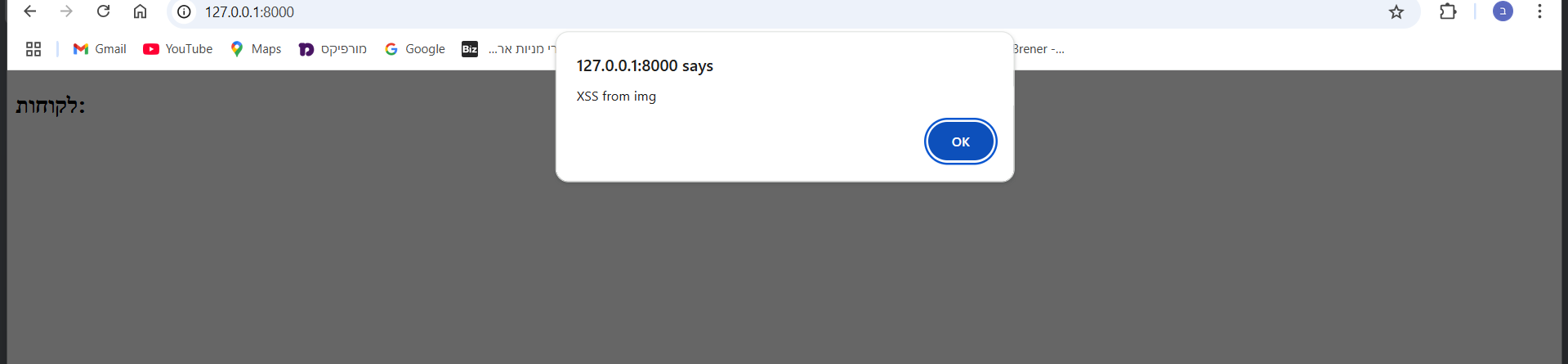


This prevents the script from executing and displays it as plain text instead.

## 💣 Image-based XSS Payload

A more effective payload that triggers even when <script> is filtered:

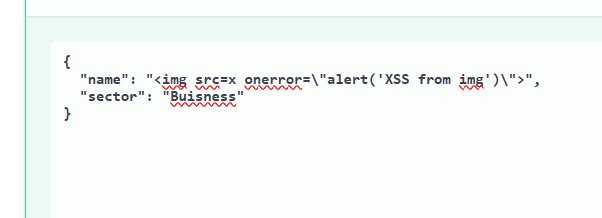
<img src=x onerror="alert('XSS from img')">



When rendered with innerHTML, the image fails to load and the onerror event fires – executing arbitrary JavaScript.

## 🔔 Browser Alert Triggered

Here is the alert as seen in the browser:



## 🧠 Summary

This attack illustrates the danger of rendering unsanitized user input as HTML.  
Proper escaping (e.g., html.escape) or templating that auto-escapes user data is critical for preventing XSS.