보고서

웹 취약점 진단 결과 보고서

작성자: 이지원
진단 대상: 자체 구축 Node.js 취약점 실습 서버 (vulnerable-node-app)
진단 기간: 2025.11.24 ~ 2025.11.27

1. 진단 요약 (Executive Summary)

본 프로젝트는 OWASP Top 10 주요 취약점에 대한 이해도를 높이고, 시큐어 코딩 역량을 강화하기 위해 수행되었습니다. 직접 취약한 웹 애플리케이션을 구축한 후 모의해킹을 수행하였으며, 총 **4건의 치명적인(Critical/High) 취약점**이 발견되었습니다

취약점 명 (Vulnerability)	위험도 (Risk)	발견 여부
SQL Injection	🔥 Critical	발견됨
Unrestricted File Upload	🔥 Critical	발견됨
Stored XSS	🔴 High	발견됨
IDOR (Broken Access Control)	🔴 High	발견됨

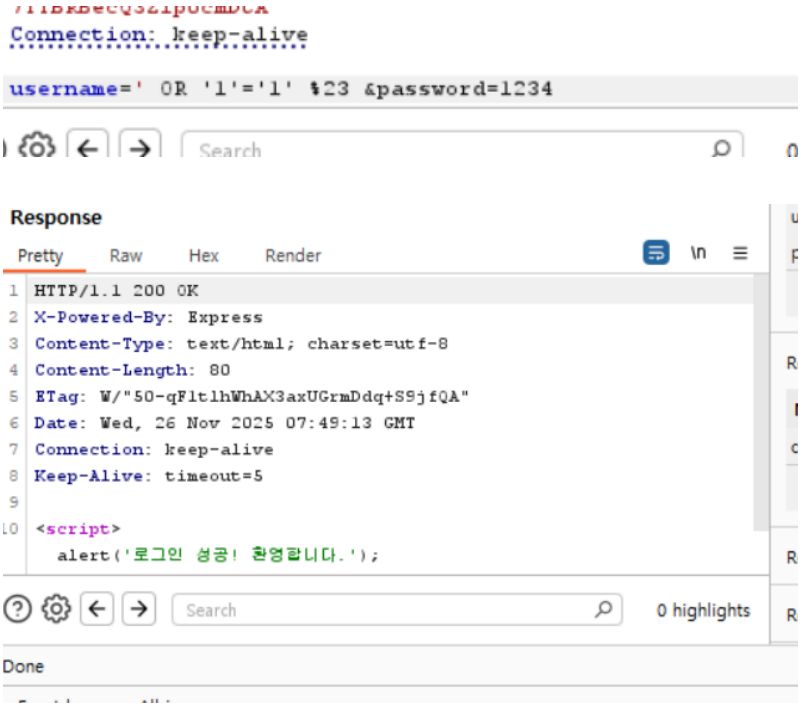
2. 상세 결과 (Vulnerability Details)

① SQL Injection (인증 우회)

- **위험도:** Critical (CVSS 9.8)
- **발생 위치:** `/auth/login` (POST)

- 취약점 설명
로그인 페이지의 아이디 입력란에 대한 검증 미흡으로, 특수문자(')와 SQL 연산자(OR)를 통해 쿼리 구조를 변조할 수 있습니다. 이를 통해 비밀번호 없이 관리자 계정 탈취가 가능합니다.

- **증거 화면 (PoC)**



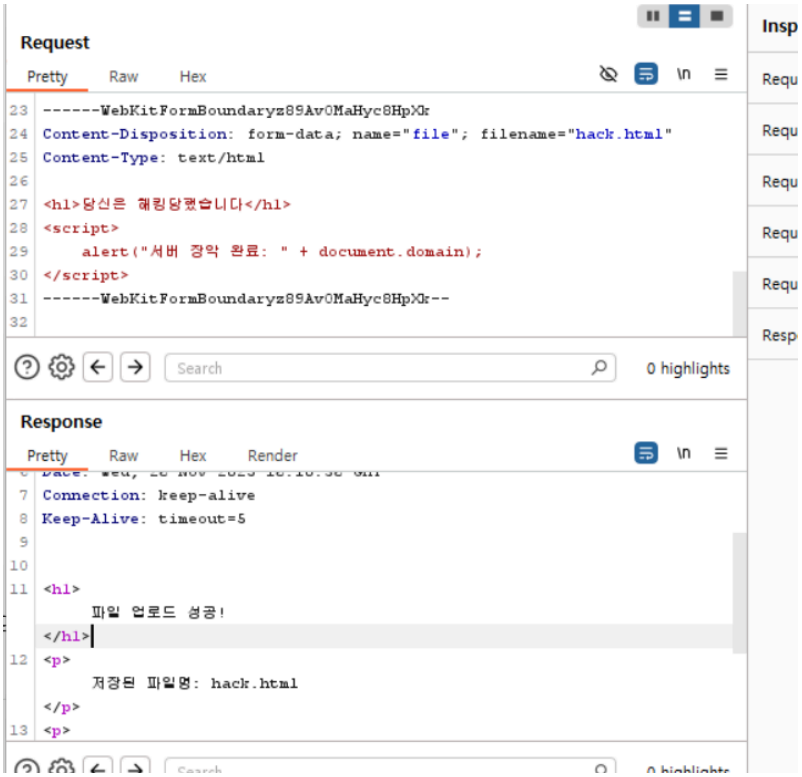
• 대응 방안 (Mitigation)
입력값을 동적으로 연결하지 않고, Prepared Statement (파라미터 바인딩) 방식을 사용하여 쿼리와 데이터를 분리해야 합니다.

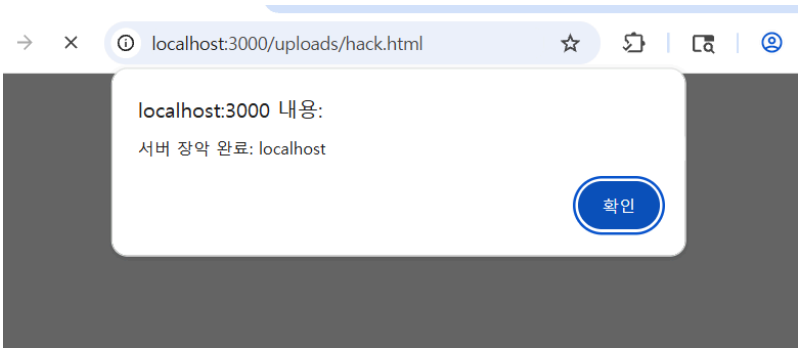
② Unrestricted File Upload (원격 코드 실행)

• **위험도:** Critical (CVSS 9.8)
• **발생 위치:** `/upload` (POST)

• 취약점 설명
파일 업로드 시 확장자 및 MIME Type에 대한 검증이 존재하지 않습니다. 공격자는 .html 등의 실행 가능한 파일을 업로드하여 서버 내에서 악성 스크립트를 실행(Webshell/XSS)할 수 있습니다.

• **증거 화면 (PoC)**





• 대응 방안 (Mitigation)
허용된 확장자(.jpg, .png)만 업로드하도록 화이트리스트 필터링을 적용하고, 파일 저장 시 **파일명 난수화(UUID)**를 적용해야 합니다.

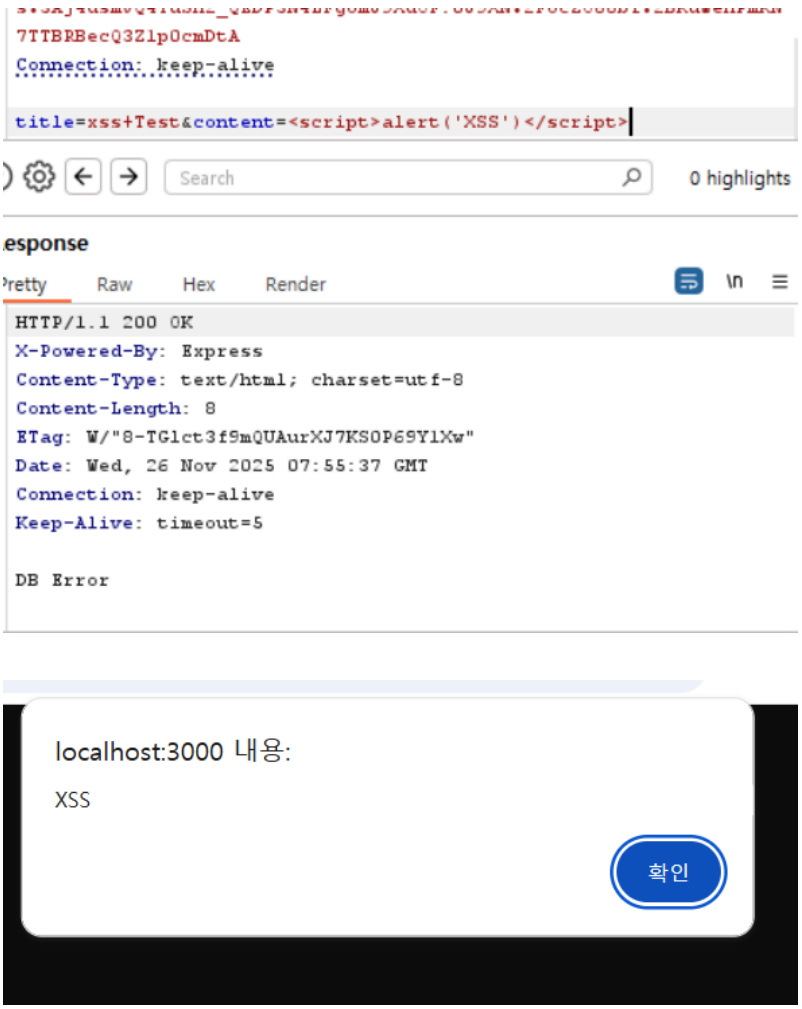
③ Stored XSS (악성 스크립트 저장)

• **위험도:** High (CVSS 6.1)
• **발생 위치:** `/post/write` (POST)

• 취약점 설명
게시글 작성 시 스크립트 태그(<script>)에 대한 이스케이핑 처리가 되어있지 않습니다. 공

격자가 악성 스크립트를 게시글에 저장하면, 이를 열람하는 모든 사용자의 브라우저에서 스크립트가 실행됩니다.

- **증거 화면 (PoC)**



- 대응 방안 (Mitigation)
사용자 입력값을 출력할 때 HTML Entity Encoding(치환)을 적용하거나, DOMPurify 같은 라이브러리를 사용하여 위험한 태그를 제거해야 합니다.

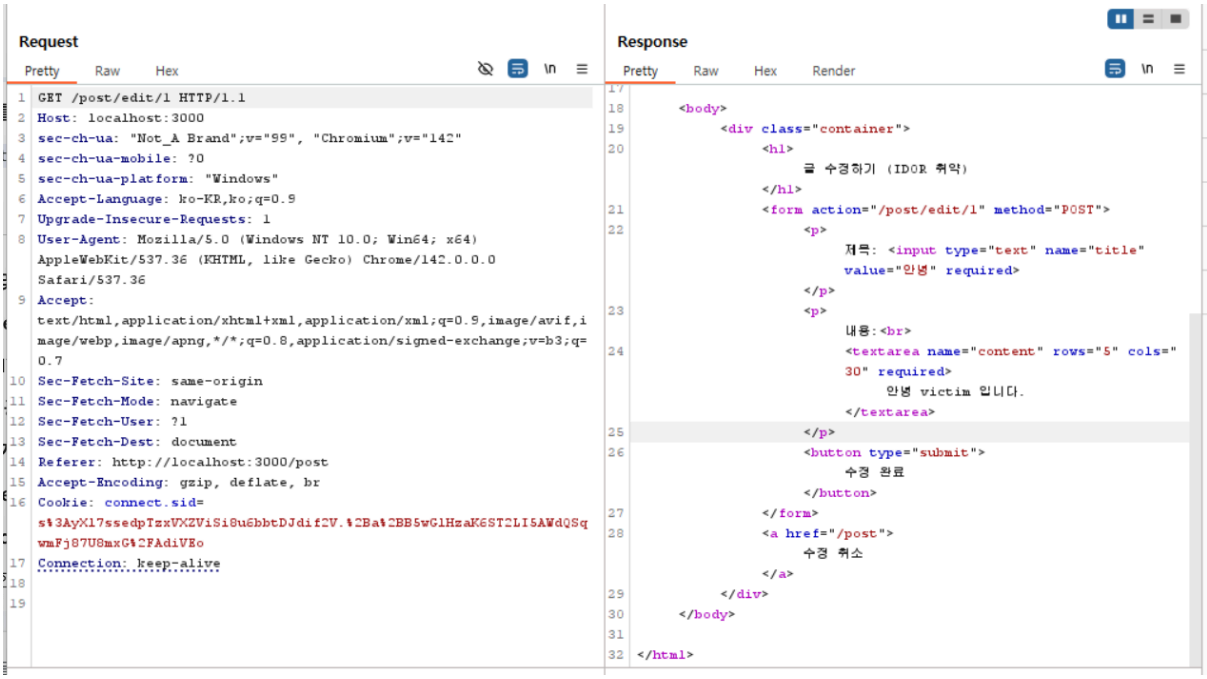
④ IDOR (부적절한 접근 제어)

- **위험도:** High (CVSS 6.5)
- **발생 위치:** `/post/edit/:id` (POST)

- 취약점 설명

게시글 수정 기능에서 요청자가 해당 글의 작성자인지 검증하지 않습니다. 공격자는 URL 파라미터(id)만 변조하여 타인의 게시글을 무단으로 수정하거나 삭제할 수 있습니다.

- **증거 화면 (PoC)**



Request

Pretty Raw Hex

```
1  GET /post/edit/1 HTTP/1.1
2  Host: localhost:3000
3  sec-ch-ua: "Not_A Brand";v="99", "Chromium";v="142"
4  sec-ch-ua-mobile: ?0
5  sec-ch-ua-platform: "Windows"
6  Accept-Language: ko-KR,ko;q=0.9
7  Upgrade-Insecure-Requests: 1
8  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0
   Safari/537.36
9  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,i
   mage/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=
   0.7
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: http://localhost:3000/post
15 Accept-Encoding: gzip, deflate, br
16 Cookie: connect.sid=
   s%3AyXl7ssedpTzxVXZViSi8u6bbtDJdif2V.%2Ba%2BB5wGlHzaK6ST2LI5AWdQSq
   wmFj87U8mxG%2FAdiVEo
17 Connection: keep-alive
18
19
```

Response

Pretty Raw Hex Render

```
17
18     <body>
19         <div class="container">
20             <h1>
                   글 수정하기 (IDOR 취약)
               </h1>
21             <form action="/post/edit/1" method="POST">
22                 <p>
                       제목: <input type="text" name="title"
                       value="안녕" required>
                   </p>
23                 <p>
                       내용:<br>
24                     <textarea name="content" rows="5" cols="
                       30" required>
                           안녕 victim 입니다.
                       </textarea>
25                 </p>
26                 <button type="submit">
                       수정 완료
                   </button>
27             </form>
28             <a href="/post">
                   수정 취소
               </a>
29         </div>
30     </body>
31
32 </html>
```

- 대응 방안 (Mitigation)

백엔드 로직에서 Session User ID와 Post Owner ID가 일치하는지 확인하는 권한 검증 절차를 추가해야 합니다.

3. 종합 결론 (Conclusion)

본 진단을 통해 인증, 입력값 검증, 접근 제어 등 웹 애플리케이션의 핵심 보안 영역에서 다수의 취약점이 발견되었습니다. 발견된 취약점들은 모두 **시큐어 코딩 가이드**를 준수함으로써 근본적인 조치가 가능하며, 추후 정기적인 모의해킹과 소스코드 진단이 필요합니다.