CJ Commerce

웹 애플리케이션 모의해킹 결과 보고서

진단 대상 : http://192.168.10.100:19191

진단 일자 : 2026-07-08

■ 공개본 안내 (레닥션)

본 문서는 CJ Olive Networks AI Security WAVE 교육과정에서 강사가 제공한 가상기업 실습 환경(사설망)을 대상으로 수행한 진단 결과이며, 실제 운영 시스템을 대상으로 하지 않았다. 포트폴리오 공개를 위해 원본에서 다음 항목을 가렸다.

· 실습 환경의 정답 플래그(CJ{...}) — 후속 교육 기수의 학습을 방해하지 않기 위함

· 사람이 설정한 형태의 평문 비밀번호(관리자·판매자 계정) — 재사용 위험 차단

· 비밀번호 재설정 보안질문의 답

반면 취약점의 내용 자체를 이루는 값(rockyou 사전으로 즉시 크랙되는 admin / password, 기본값 형태의 DB 계정 등)과 합성 고객 데이터(example.test 도메인, 순번형 이름·전화번호)는 진단 결과의 근거이므로 그대로 두었다. 가린 부분은 검은 막대로 표시하였다.

목 차

1 SQL Injection (파라미터 조작)
- 1.1 취약점 개요
- 1.2 상세설명
- 1.3 조치 방안

2 SQL Injection (Blind - 추가 지점)
- 2.1 취약점 개요
- 2.2 상세설명
- 2.3 조치 방안

3 File Download (경로 조작 / LFI)
- 3.1 취약점 개요
- 3.2 상세설명
- 3.3 조치 방안

4 File Upload (웹쉘 업로드 → RCE)
- 4.1 취약점 개요
- 4.2 상세설명
- 4.3 조치 방안

5 Command Injection
- 5.1 취약점 개요
- 5.2 상세설명
- 5.3 조치 방안

6 XSS (Stored)
- 6.1 취약점 개요
- 6.2 상세설명
- 6.3 조치 방안

7 취약한 비밀번호 저장
- 7.1 취약점 개요
- 7.2 상세설명
- 7.3 조치 방안

8 관리자 페이지 통제 미흡
- 8.1 취약점 개요
- 8.2 상세설명
- 8.3 조치 방안

■ 취약점 요약

No	취약점	위험도
1	SQL Injection (파라미터 조작) — 정보노출·민감정보·인증우회 포함	상
2	SQL Injection (Blind - 추가 지점)	상
3	File Download (경로 조작 / LFI) — 소스·크레덴셜 노출 포함	상
4	File Upload (웹쉘 업로드 → RCE) — 필터 우회 포함	상
5	Command Injection (OS 명령 실행)	상
6	XSS (Stored / 관리자 화면 실행)	상
7	취약한 비밀번호 저장 (약한 해싱)	중
8	관리자 페이지 통제 미흡	중

※ 상기 취약점들은 서로 연계되어 다음의 공격 체인을 구성함

정보수집 → 파일 다운로드 경로 조작(LFI)으로 소스·DB 접속정보 확보 → SQL Injection 으로 DB 전체 열람 (민감정보·크레덴셜) → 파일 업로드(웹쉘) + SQLi(파일명 조회)로 원격 명령 실행(RCE) → DB 직접 접근 → 관리자 계정 탈취 → 관리자 기능(Command Injection) 악용

1. SQL Injection (파라미터 조작)

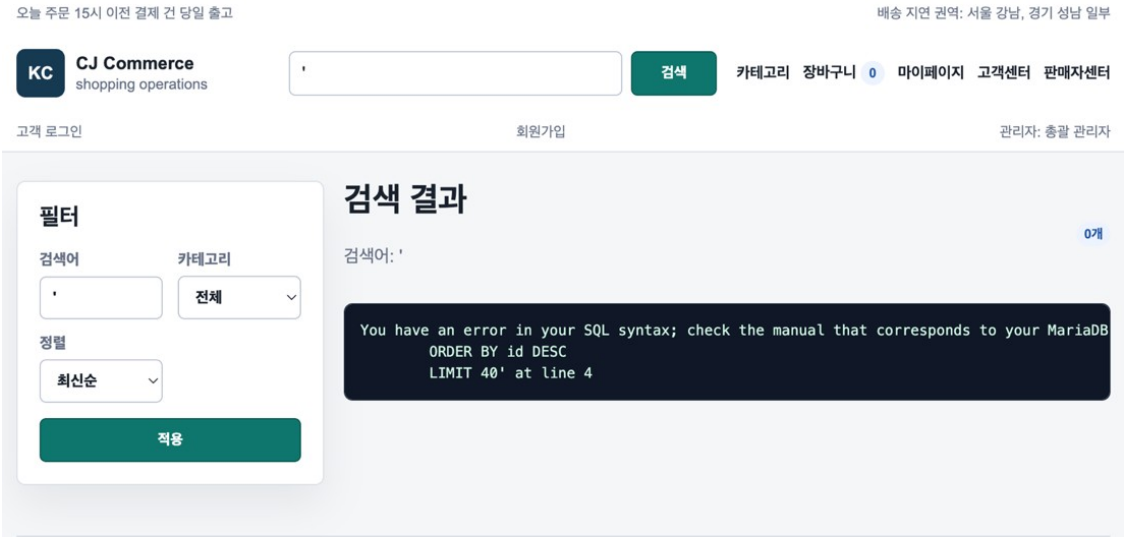
취약점 내용	요청 파라미터 값에 SQL 구문을 삽입하여 데이터베이스 쿼리를 조작하는 취약점. 인증 없이 DB 데이터를 조회·추출할 수 있으며, 이를 통해 오류 메시지 기반 내부 구조 노출, 회원·주문·관리자 크레덴셜 등 민감정보의 대량 조회, 나아가 획득한 계정정보를 이용한 정상 인증 절차 우회까지 연계된다.
위험도	상
산정 근거	**사용자 입력값이 SQL 쿼리에 안전하게 바인딩되지 않아 임의의 SQL 구문 삽입이 가능하였고, 이를 통해 information_schema 조회, 테이블·컬럼 열거, 회원·주문·관리자 정보 조회가 가능함을 확인하였다. 또한 오류 메시지에 쿼리 구조가 노출되고, 추출한 크레덴셜·보안질문 답변을 이용해 관리자 계정 접근(인증 우회)까지 가능하여 시스템 전체에 미치는 영향이 크므로 위험도를 '상'으로 산정하였다.**

1.1 취약점 개요

URL	http://192.168.10.100:19191/product.php , http://192.168.10.100:19191/search.php , http://192.168.10.100:19191/login.php , http://192.168.10.100:19191/admin (비밀번호 재설정)
파라미터	id, q, username, security_answer

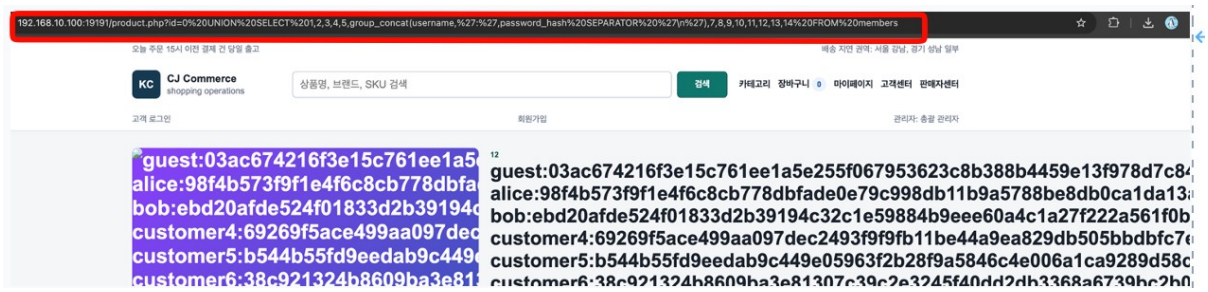
1.2 상세설명

Step 1) 주입점 확인 및 오류 기반 정보 노출 — search.php 검색창에 작은따옴표(')를 입력하면 MariaDB SQL 문법 오류가 화면에 그대로 노출된다. 오류 메시지에 쿼리 구조(ORDER BY id DESC LIMIT 40)가 포함되어 내부 쿼리를 유추할 수 있다.



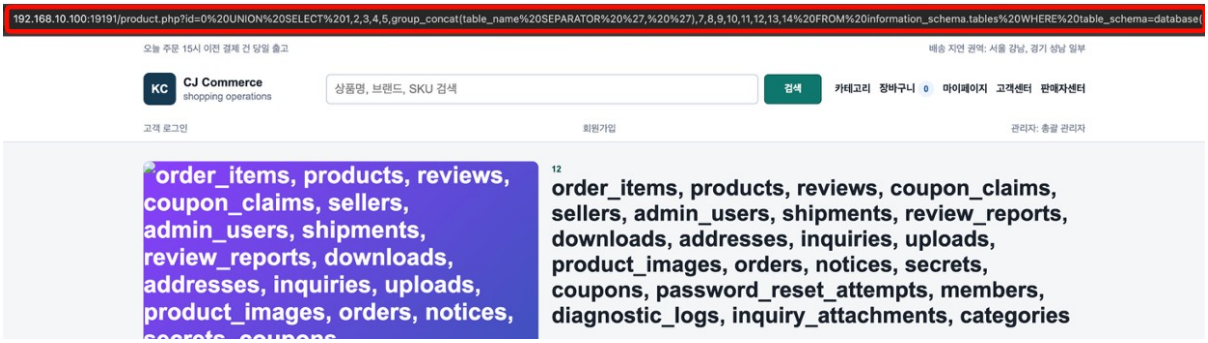
검색창 작은따옴표 입력 시 SQL 오류 메시지·쿼리 구조 노출 화면

Step 2) product.php?id=92 AND 1=1 은 정상, id=92 AND 1=2 는 응답이 달라지는 것으로 숫자형 SQL Injection 을 확인한다. ORDER BY 구문으로 컬럼 수(14 개)를 확인하고 UNION SELECT 로 임의 데이터를 조회한다. (id=0 UNION SELECT 1,...,14)



product.php?id= UNION SELECT 페이로드 및 데이터 추출 결과 (URL 포함)

Step 3) information_schema 및 각 테이블을 열거하여 DB 전체 구조를 파악한다.

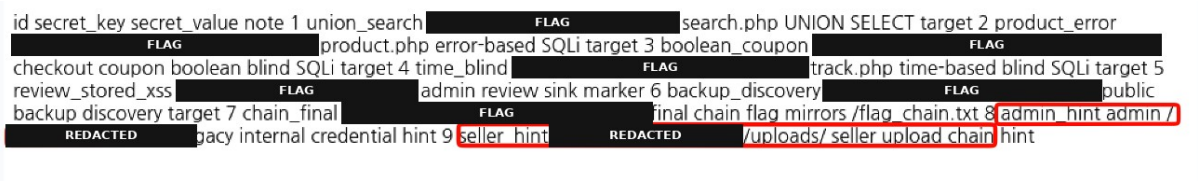


테이블/컬럼 열거 결과 화면

Step 4) 민감정보 대량 추출 — members(username, password_hash, name, email, phone, security_answer), orders(수령인명·전화·주소·내부 메모), secrets(관리자 평문 크레덴셜 admin / Dhvj**********) 등을 조회한다. 해당 테이블의 존재 자체는 정상이나, 인증 없이 SQL Injection 만으로 접근·열람이 가능하다는 점이 취약점이다.

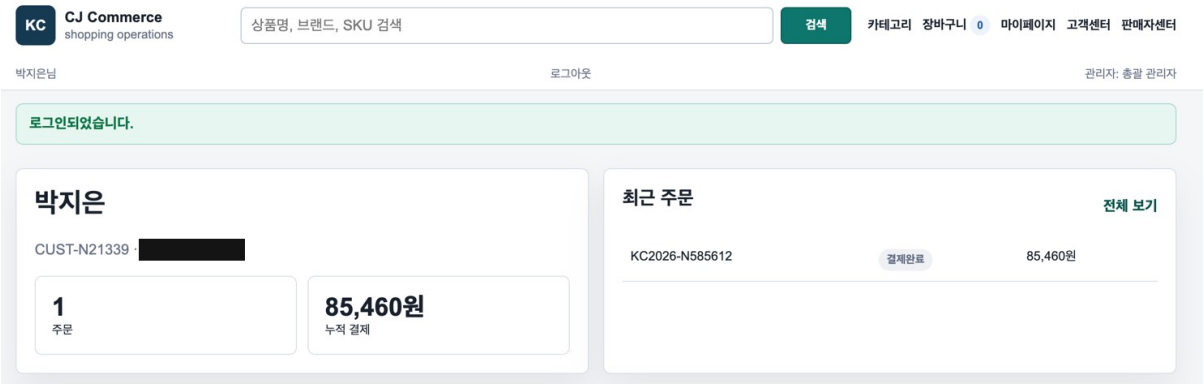
members 테이블 계정/해시 덤프 화면

orders 테이블 개인정보(이름·전화·주소) 덤프 화면

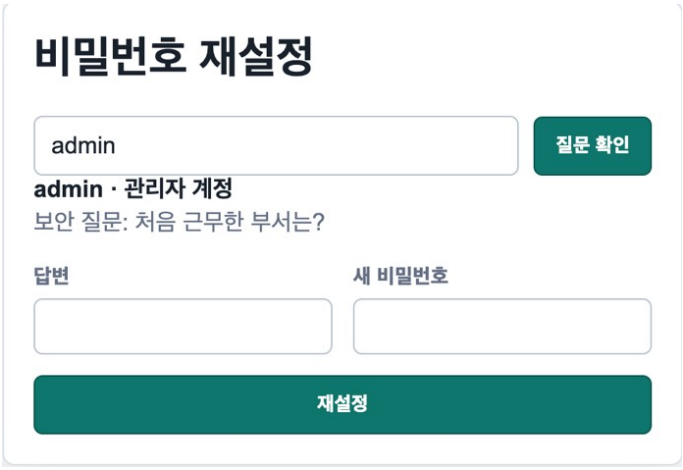


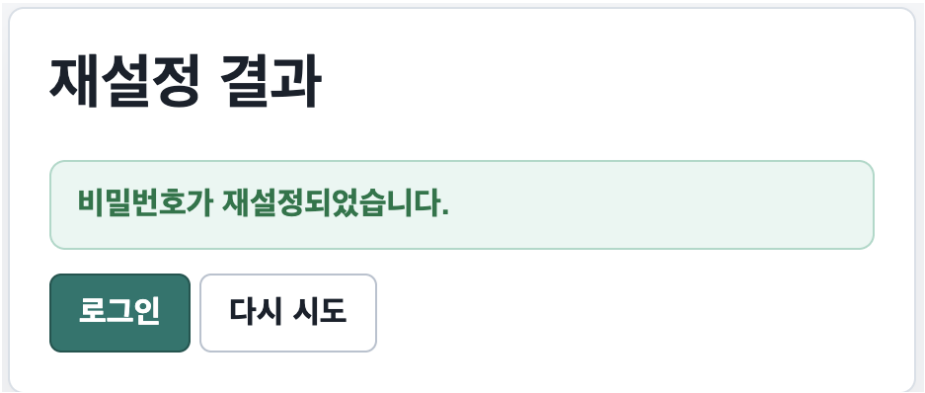
secrets 테이블 크레덴셜 평문 노출 화면

Step 5) 인증 우회 — 추출한 members 해시(SHA-256, salt 미적용)를 크랙하여 취약 계정(admin / password)으로 로그인한다. 또한 비밀번호 재설정 화면이 보안질문('처음 근무한 부서는?')을 노출하고, 그 답변(<REDACTED>)이 DB 에 평문 저장되어 있어 이를 SQL Injection 으로 확보한 뒤 관리자 비밀번호를 임의 재설정하여 계정을 탈취한다. 즉 정상 인증/재설정 절차를 거치지 않고 SQLi 로 획득한 정보만으로 계정 접근이 이루어졌다.



취약 계정(admin / password) 로그인 성공 화면





비밀번호 재설정 화면의 보안질문 노출 및 답변(<REDACTED>) 입력 화면

※ 검색 기능(search.php)은 입력값이 다중 컬럼(title/description/brand)에 복제 삽입되며, ' OR '1'='1 입력 시 전체 결과가 반환됨을 확인함.

※ 인증 우회의 근본 원인은 SQL Injection 이다. MFA 등 별도 인증수단을 논리적으로 우회한 것이 아니라, SQLi 로

획득한 크레덴셜·보안질문 답변을 그대로 사용해 계정에 접근한 것이므로 본 항목에 통합한다.

1.3 조치 방안

1. 모든 DB 쿼리에 Prepared Statement(파라미터 바인딩)를 적용하여 입력값이 SQL 구문으로 해석되지 않도록 한다.
2. 입력값에 대한 자료형/형식 검증(숫자 파라미터는 정수 캐스팅 등)을 수행하고, SQL Injection 대상 파라미터를 전수 점검하여 동일 패턴을 제거한다.
3. 운영 환경에서 상세 오류 메시지 출력을 비활성화(display_errors=Off)하고, 상세 오류는 서버 내부 로그에만 기록한다.
4. 개인정보(전화번호·주소 등)는 암호화 저장 및 조회 시 마스킹하고, 비밀번호·크레덴셜·보안질문 답변의 평문 저장을 금지한다.
5. 비밀번호 재설정 시 계정 존재 여부·보안질문을 화면에 노출하지 않고 등록된 이메일 등 별도 채널로 인증 링크를 발송하며, 관리자 로그인에 다중 인증(MFA)을 적용한다.
6. DB 계정 권한을 최소화하고, WAF 로 알려진 SQL Injection 패턴을 탐지·차단한다.

2. SQL Injection (Blind - 추가 지점)

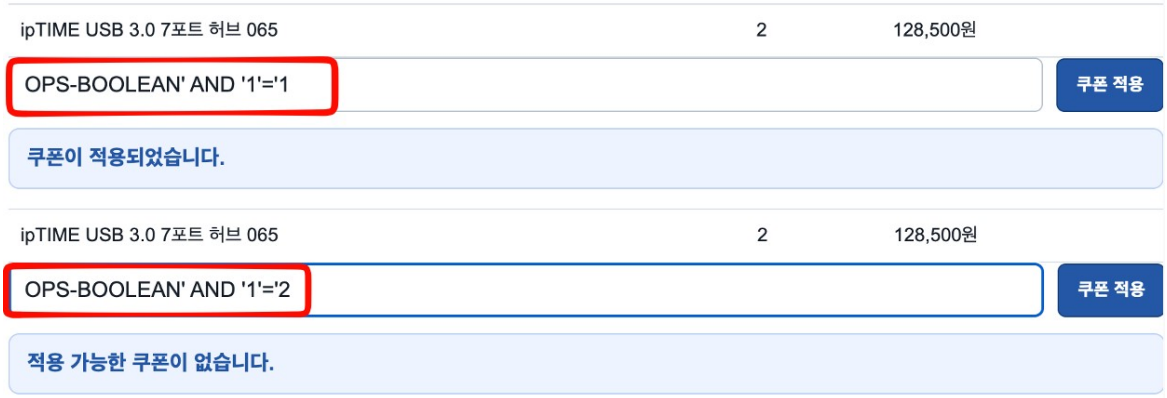
취약점 내용	쿠폰 적용 기능에 Boolean-based Blind SQL Injection 지점이 존재하여, 참/거짓 조건에 따른 응답 차이 또는 응답 시간을 통해 데이터를 추출할 수 있는 취약점.
위험도	상
산정 근거	**쿠폰 적용 기능에서 참/거짓 조건에 따라 응답이 달라지는 Boolean-based Blind SQL Injection 지점이 확인되었다. Blind 방식은 추출 속도가 느리지만 자동화 도구·반복 요청으로 DB 정보 조회가 가능하며, 동일 애플리케이션 내 다른 SQL Injection 지점과 같은 원인(파라미터 바인딩 미적용)으로 발생하므로 위험도를 '상'으로 산정하였다.**

2.1 취약점 개요

URL	http://192.168.10.100:19191/ (checkout 쿠폰 적용)
파라미터	coupon_code

2.2 상세설명

Step 1) 쿠폰 적용(OPS-BOOLEAN 등) 기능에서 참/거짓 조건에 따라 응답이 달라지는 Boolean-based Blind SQL Injection 지점을 확인한다.



쿠폰 적용 시 참/거짓 조건에 따른 응답 차이 화면

※ 본 지점은 in-band SQL Injection(1 번)으로 이미 데이터 추출이 확인되어 지점 확인 수준으로 정리함. 동일 원인(파라미터 바인딩 미적용)으로 함께 조치가 필요하다.

2.3 조치 방안

1. 전체 쿼리에 Prepared Statement(파라미터 바인딩)를 일괄 적용한다.
2. 입력값 자료형/형식 검증을 수행한다.
3. SQL Injection 대상이 되는 모든 파라미터를 전수 점검하여 동일 패턴을 제거한다.

3. File Download (경로 조작 / LFI)

취약점 내용	다운로드 기능이 파일 경로 파라미터를 검증 없이 사용하여, 상위 경로 이동(../)으로 서버 임의 파일을 읽을 수 있는 경로 조작(Path Traversal / LFI) 취약점. 이를 통해 애플리케이션 소스코드와 DB 접속정보(config.php)가 평문으로 노출된다.
위험도	상
산정 근거	**다운로드 파라미터 조작으로 /etc/passwd 및 애플리케이션 소스코드 등 서버 내부 파일을 임의로 조회할 수 있었다. 특히 config.php 노출을 통해 DB 접속정보(app / app_password)가 평문으로 확인되었으며, 이는 데이터베이스 접근 및 추가 공격(웹쉘 업로드 경로·내부 로직 파악)으로 직접 연계되므로 위험도를 '상'으로 산정하였다.**

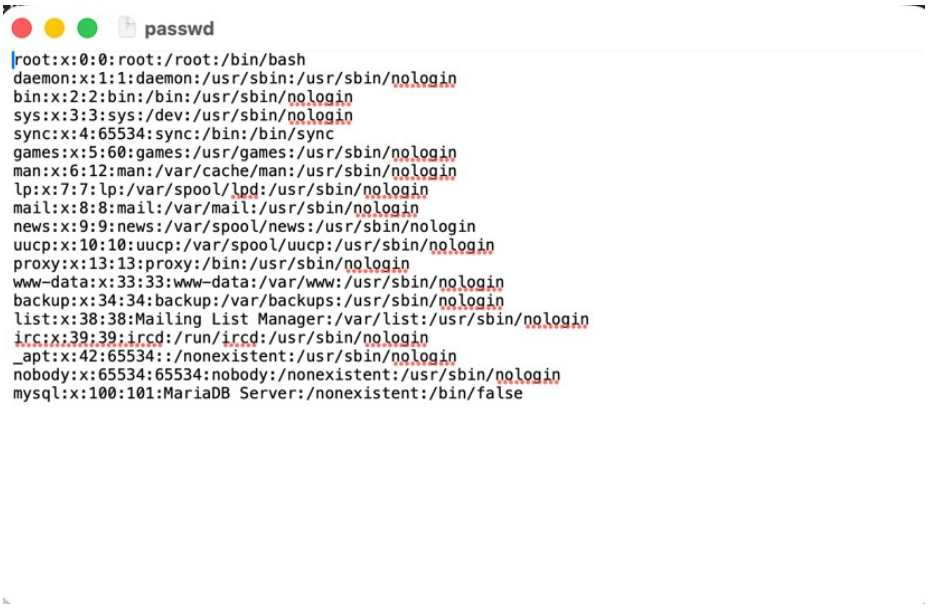
3.1 취약점 개요

URL	http://192.168.10.100:19191/download.php
파라미터	file

3.2 상세설명

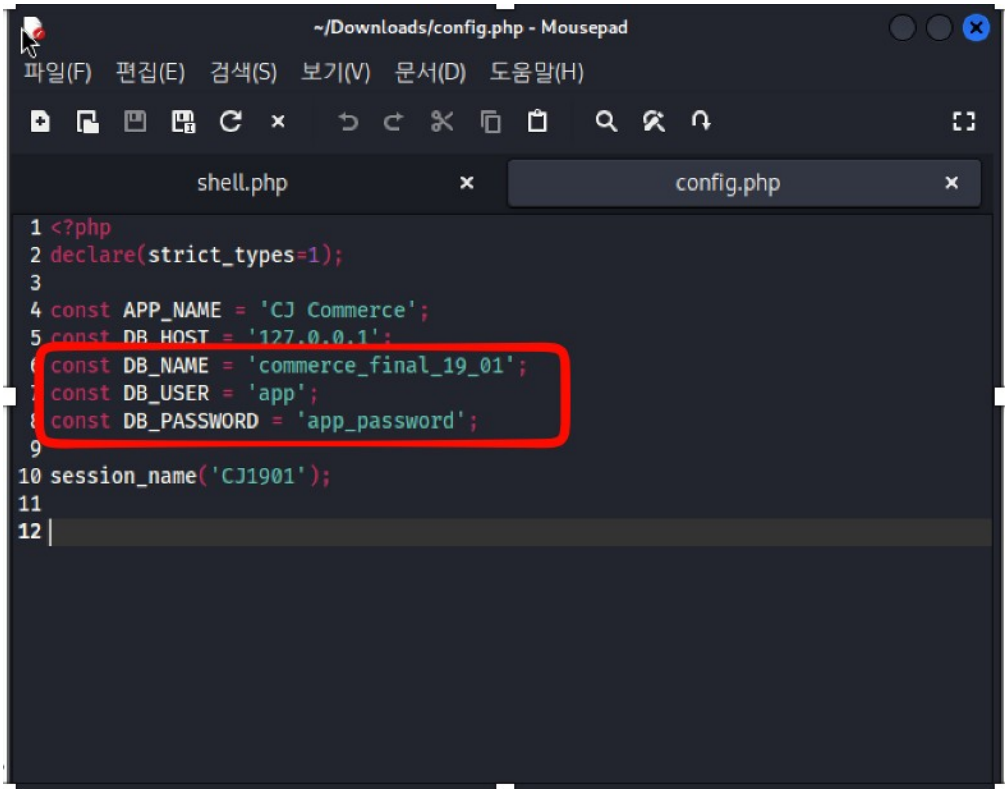
Step 1) 샘플 인보이스(sample_invoice.txt)에서 다운로드 경로(download.php?file=)를 확인한다.

Step 2) file 파라미터에 ../ 를 삽입하여 상위 경로로 이동, /etc/passwd 를 읽는다.
(file=../../../../../etc/passwd → root:x:0:0:... 노출)



주소창에 ?file=../../../../../etc/passwd 입력 및 /etc/passwd 노출 화면 (URL 포함)

Step 3) 동일 방식으로 config.php 를 읽어 DB 접속정보(DB_USER=app, DB_PASSWORD=app_password, DB_NAME=commerce_final_19_01)를 평문으로 확보한다. readfile()은 include 와 달리 .php 를 실행하지 않고 소스를 그대로 반환하므로 별도 래퍼 없이 소스 확보가 가능하다.



config.php 소스 내 DB 접속정보(app / app_password) 평문 노출 화면

Step 4) download.php, inquiry.php 소스를 읽어 내부 로직(필터 방식, 업로드 저장 경로·파일명 규칙)을 확보한다.

config.php

```
Users > donggeon > Downloads > config.php
1   <?php
2   declare(strict_types=1);
3
4   const APP_NAME = 'CJ Commerce';
5   const DB_HOST = '127.0.0.1';
6   const DB_NAME = 'commerce_final_19_01';
7   const DB_USER = 'app';
8   const DB_PASSWORD = 'app_password';
9
10  session_name('CJ1901');
11
12
```

~/Downloads/inquiry.php - Mousepad

파일(F) 편집(E) 검색(S) 보기(V) 문서(D) 도움말(H)

```
1 <?php
2 require_once __DIR__ . '/lib.php';
3
4 $user = current_user();
5 $message = '';
6 $error = '';
7
8 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
9     $memberId = $user ? (int) $user['id'] : null;
10    $name = trim((string) ($_POST['name'] ?? ''));
11    $email = trim((string) ($_POST['email'] ?? ''));
12    $orderNo = trim((string) ($_POST['order_no'] ?? ''));
13    $title = trim((string) ($_POST['title'] ?? ''));
14    $body = (string) ($_POST['body'] ?? '');
15
16    if ($name === '' || $email === '' || $title === '' || $body === '') {
17        $error = '필수 항목을 입력하세요.';
18    } else {
19        $stmt = $db->prepare('INSERT INTO inquiries (member_id, name, email,
  order_no, title, body) VALUES (?, ?, ?, ?, ?, ?)');
20        $stmt->bind_param('isssss', $memberId, $name, $email, $orderNo,
  $title, $body);
21        $stmt->execute();
```

config.php / inquiry.php 소스 노출 화면

※ 소스 분석 결과, 필터가 str_replace('../', '', $file) 로 구현되어 ../ 자체를 전혀 차단하지 못하는 것이 근본 원인이다.

3.3 조치 방안

1. 사용자 입력 경로를 realpath()로 정규화한 후 지정된 기준 디렉터리 하위에 있는지 검증하고, 벗어나면 차단한다.
2. basename() 등으로 경로 성분(../)을 제거하여 파일명만 사용하고, 다운로드 가능한 파일은 화이트리스트로 관리한다.
3. 블랙리스트 문자열 치환 방식의 필터링은 우회가 가능하므로 사용하지 않는다.

4. DB 접속정보 등 민감 설정값은 소스코드 내 평문 저장을 지양하고 환경변수 또는 접근이 제한된 설정 저장소로 분리하며, 웹 루트 하위에서 소스코드 원문이 다운로드되지 않도록 접근 경로를 제한한다.

4. File Upload (웹쉘 업로드 → RCE)

취약점 내용	첨부/이미지 업로드 기능이 확장자·MIME·파일 내용을 제대로 검증하지 않아, 악성 PHP 웹쉘 업로드를 통해 서버 원격 명령 실행(RCE)이 가능한 취약점. 판매자 상품 이미지 업로드의 검수 로직은 방식별(Basic/MIME/이중확장자/대소문자/Combined)로 우회가 가능하다.
위험도	상
산정 근거	**문의 첨부 업로드는 검증이 전혀 없어 PHP 웹쉘이 그대로 저장·실행되어 웹 서버 권한(www-data)으로 OS 명령 실행이 가능함을 확인하였다. 또한 판매자 업로드의 확장자·MIME·대소문자·이중확장자 검증이 불완전하여 다양한 방식으로 필터 우회가 가능하였다. 단순 업로드 문제가 아니라 RCE 로 직접 이어지고 재현 가능성이 높으므로 위험도를 '상'으로 산정하였다.**

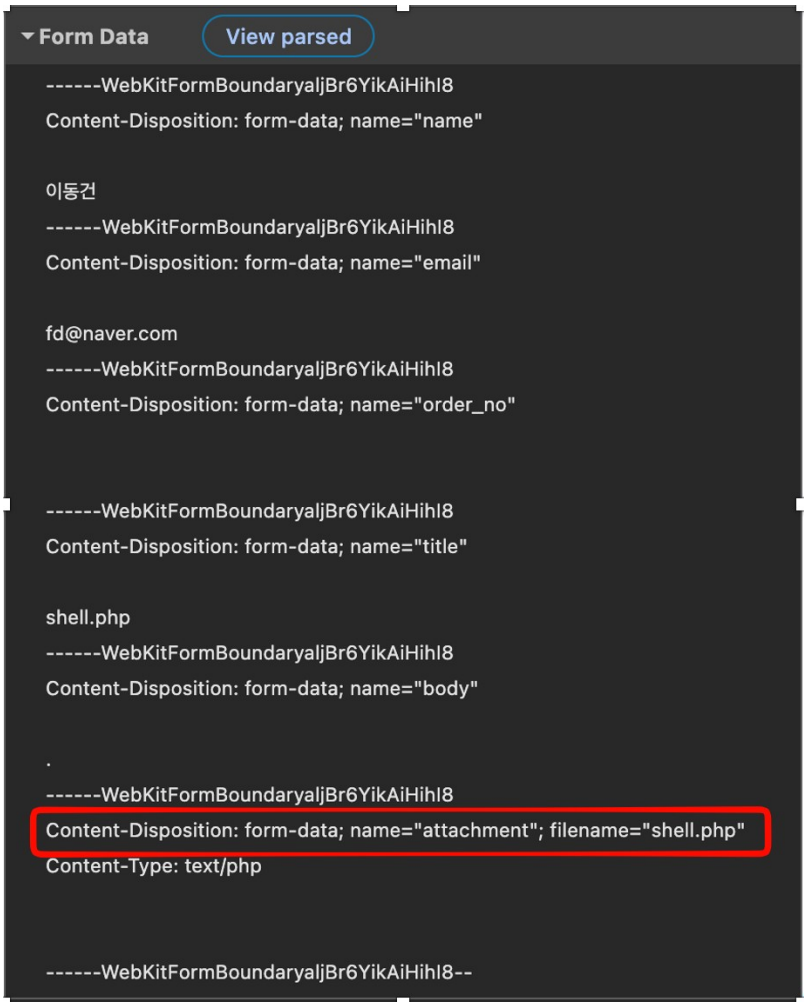
4.1 취약점 개요

URL	http://192.168.10.100:19191/inquiry.php (문의 첨부) , http://192.168.10.100:19191/uploads/<저장파일명> , http://192.168.10.100:19191/seller (상품 이미지 업로드)
파라미터	attachment (첨부파일), file, 검수 모드(mode)

4.2 상세설명

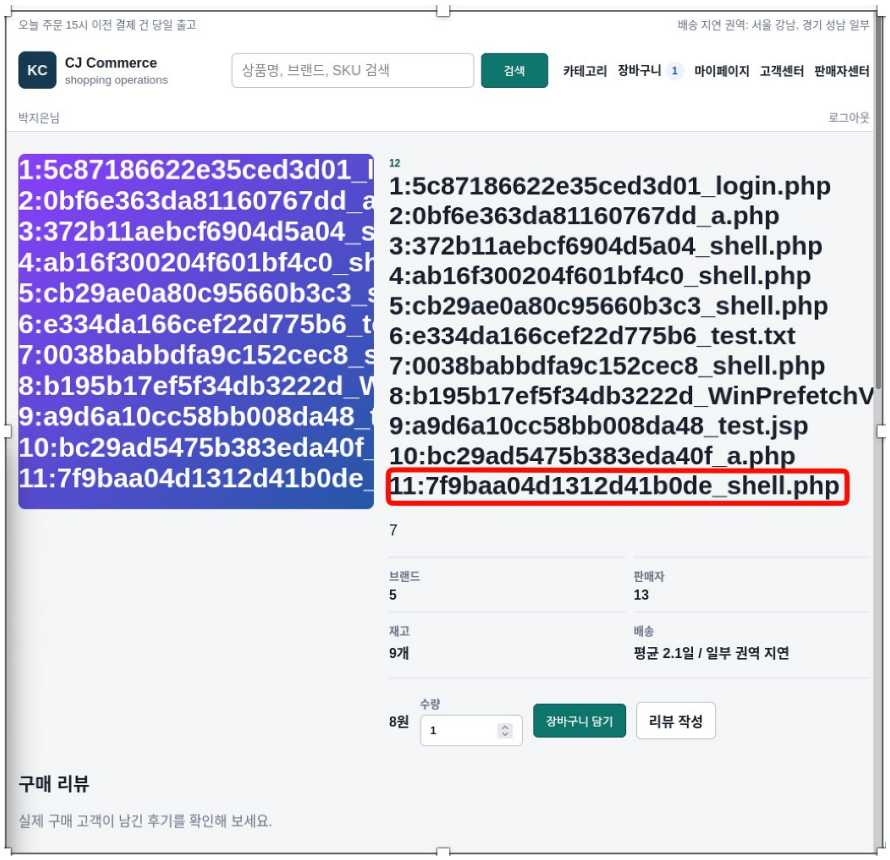
[4-A. 문의 첨부 업로드 → 웹쉘 RCE]

Step 1) 1:1 문의 첨부에 웹쉘 shell.php (<?php system($_GET['c']); ?>)를 업로드한다. 확장자/MIME 검증이 없어 .php 가 그대로 저장된다. (요청 원문상 filename="shell.php" 확인)



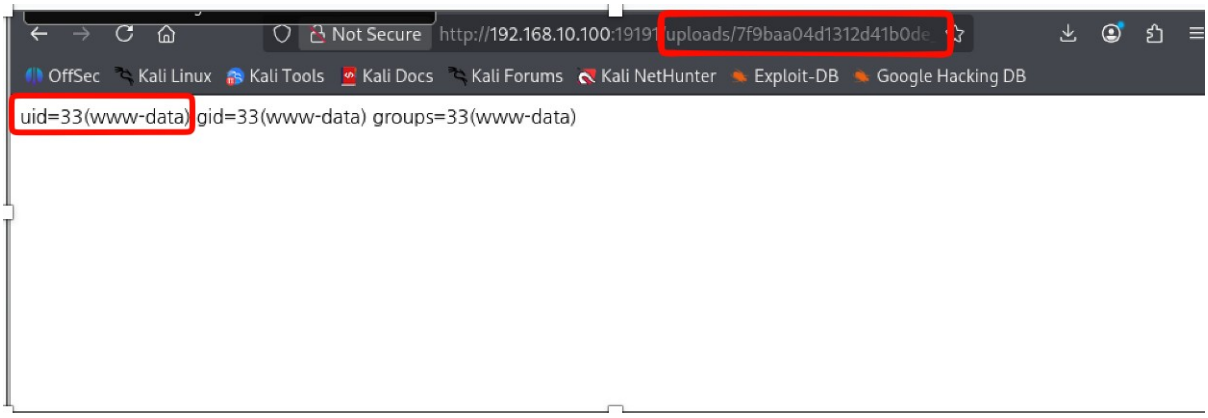
개발자도구(Network > Payload)상 filename="shell.php" 업로드 요청 원문

Step 2) 저장 파일명은 랜덤(bin2hex(random_bytes(10)))으로 변경되나, 해당 파일명이 inquiry_attachments 테이블에 저장되므로 SQL Injection 으로 stored_name 을 조회하여 실제 파일명을 획득한다.



SQL Injection 으로 inquiry_attachments.stored_name(웹쉘 파일명) 조회 화면

Step 3) 획득한 경로 /uploads/<파일명>?c=id 로 접근하면 명령이 실행되어 uid=33(www-data)이 반환된다. 서버 원격 명령 실행(RCE)이 확인된다.



웹쉘 실행 결과 uid=33(www-data) 반환 화면

※ 본 취약점은 SQL Injection(파일명 조회)과 연계되어 RCE 로 이어지는 복합 공격 체인이다. chmod 0444 로 실행권한을 제거해도 PHP 는 웹서버가 해석하므로 무력하다.

[4-B. 판매자 이미지 업로드 필터 우회 5 종]

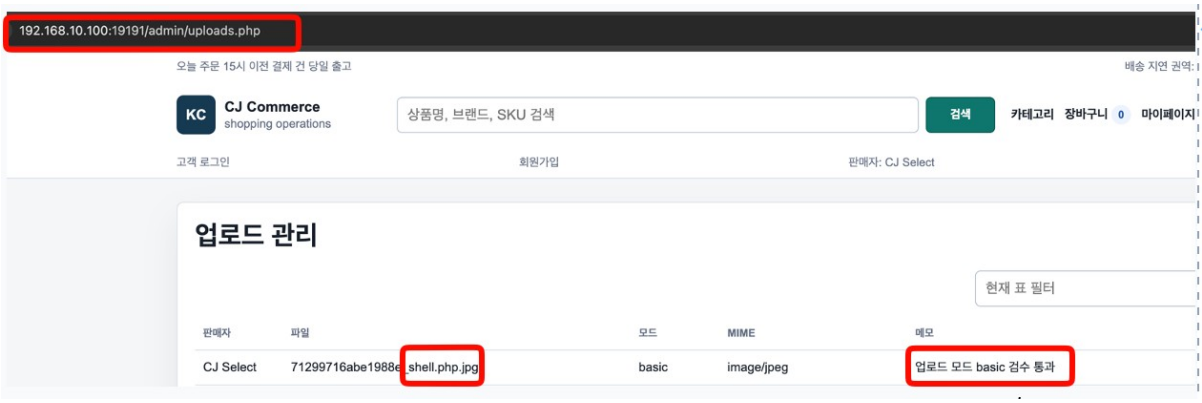
Basic: 파일 존재 여부만 확인 → .php 파일이 그대로 업로드된다.

MIME: 브라우저 전송 Content-Type 만 확인 → 요청의 Content-Type 을 image/jpeg 로 위조하여 우회한다.

Double(이중확장자): 첫 확장자만 검사 → shell.php.jpg / shell.jpg.php 형태로 우회한다.

Case: 소문자 .php 만 차단 → shell.pHp / shell.PHP 로 우회한다.

Combined: 위 조건을 순차 적용 → magic byte(GIF89a 등) + 확장자 조합으로 우회한다.



관리자 업로드 관리의 우회 파일(legacy_..._promo.jpg.php, mode=combined) 잔존 확인

4.3 조치 방안

1. 허용 확장자는 화이트리스트로만 관리하고 대소문자를 통일하여 검사하며(소문자 변환 후 비교), 이중 확장자를 차단하고 마지막 확장자가 아닌 전체 패턴을 검증한다.
2. 클라이언트 Content-Type 을 신뢰하지 않고 서버에서 파일 시그니처(magic byte)와 MIME 을 직접 검증하며, getimagesize() 등으로 실제 이미지 여부를 확인한다.
3. 업로드 디렉터리에서 스크립트 실행을 차단한다(.htaccess 또는 웹서버 설정에서 PHP 핸들러 비활성화).
4. 저장 파일명을 랜덤화하고 업로드 경로를 웹 루트 외부에 두어 직접 실행되지 않도록 한다.

5. Command Injection

취약점 내용	관리자 진단 기능이 사용자 입력(호스트명)을 시스템 명령(ping)에 검증 없이 연결하여, 임의 운영체제 명령을 실행할 수 있는 취약점.
위험도	**상**
산정 근거	**관리자 진단 기능의 host 파라미터가 OS 명령에 직접 연결되어 임의 명령 실행이 가능하였다. 공격자는 웹 서버 권한으로 파일 조회·시스템 정보 확인·추가 악성 파일 실행 등 서버 내부 행위를 수행할 수 있으며, 권한 상승이나 내부망 공격의 발판으로 악용될 수 있으므로 위험도를 '상'으로 산정하였다.**

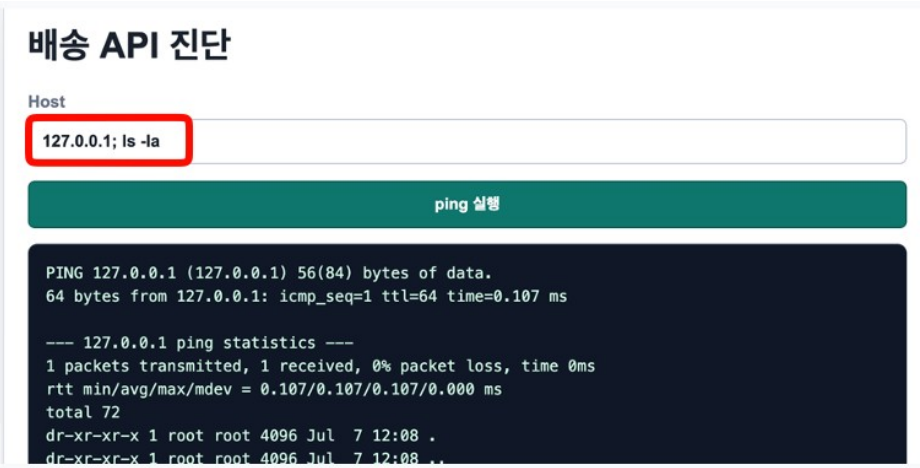
5.1 취약점 개요

URL	http://192.168.10.100:19191/admin (배송 API 진단 / ping)
파라미터	host

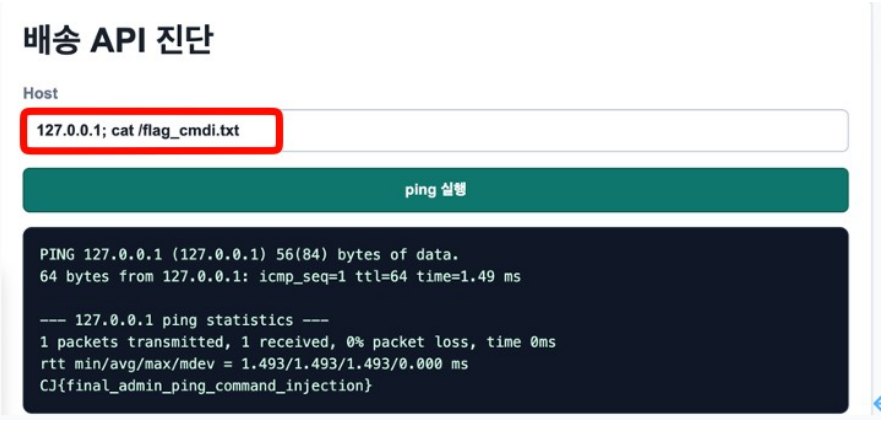
5.2 상세설명

Step 1) 관리자 배송 API 진단(ping) 기능의 Host 입력란에 127.0.0.1; ls -la 입력 시 ping 결과 뒤에 ls 명령 결과가 함께 출력된다(; 구분자로 명령이 연결됨).

Step 2) 127.0.0.1; cat /flag_cmdi.txt 입력 시 서버 파일 내용이 반환되어 임의 명령 실행이 확인된다.



개발자도구(Payload) 상 host=127.0.0.1; cat ... 요청 원문



ping 결과와 함께 명령 실행 결과(파일 내용)가 반환된 화면

5.3 조치 방안

1. 사용자 입력을 시스템 명령에 직접 연결하지 않는다. 불가피하면 인자를 배열로 전달하는 안전한 API(shell 미경유)를 사용한다.
2. 호스트명 입력값을 화이트리스트/정규식(IP·도메인 형식)으로 엄격히 검증하고 ; | & $ 등 메타문자를 차단한다.
3. 가능하면 OS 명령 대신 언어 내장 기능(소켓 등)으로 대체하고, 웹 서비스 실행 계정의 권한을 최소화한다.

6. XSS (Stored)

취약점 내용	사용자가 입력한 스크립트가 저장되어 관리자 검수 화면에서 실행되는 저장형 (Stored) XSS 취약점으로, 낮은 권한 사용자가 관리자 세션(쿠키 등)을 탈취할 가능성이 존재함.
위험도	상
산정 근거	**사용자가 입력한 스크립트가 관리자 검수 화면에서 실행되는 Stored XSS 취약점으로, 일반 사용자 권한의 입력이 관리자 브라우저에서 실행될 수 있다. 이를 통해 관리자 세션 탈취, 관리자 권한 기능 오용, 내부 정보 접근 등 2 차 피해로 이어질 수 있으므로 위험도를 '상'으로 산정하였다.**

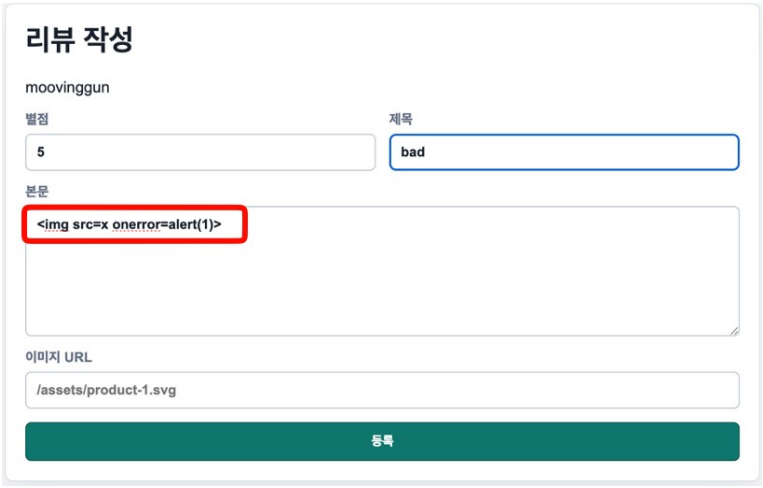
6.1 취약점 개요

URL	http://192.168.10.100:19191/product.php (리뷰 작성) → 관리자 리뷰 검수

파라미터	리뷰 제목(title), 내용(body)

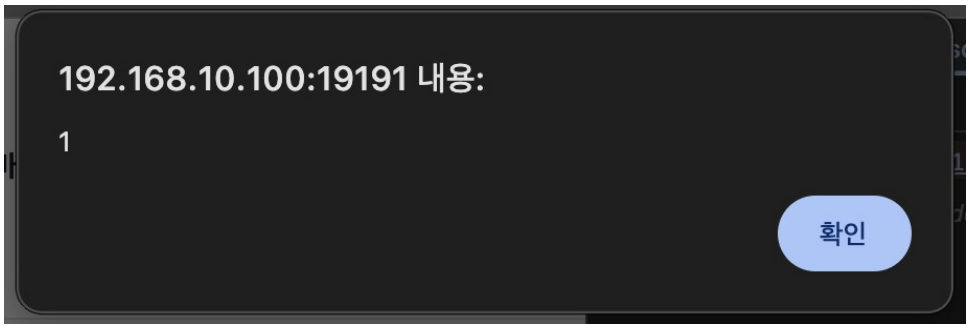
6.2 상세설명

Step 1) 상품 리뷰 작성 시 제목/내용에 스크립트 페이로드 <img src=x onerror=alert(1)> 를 입력하여 저장한다. (일반 사용자 리뷰 목록에서는 이스케이프되어 문자열로만 표시됨)



리뷰 작성폼에 <img src=x onerror=alert(1)> 입력 화면

Step 2) 관리자로 로그인하여 리뷰 검수 목록에서 해당 리뷰를 클릭(상세 검수)하면, 저장된 스크립트가 이스케이프 없이 렌더링되어 실제 실행된다(alert 경고창 발생). 즉 사용자 입력이 관리자 화면에서 실행되는 구조이다.



관리자 리뷰 검수에서 클릭 시 alert(1) 경고창 실행 화면

※ 일반 사용자 화면은 이스케이프되나 관리자 검수 상세 화면은 미적용되어 'admin review sink' 형태로 실행됨.

6.3 조치 방안

1. 모든 사용자 입력값은 출력 위치(HTML 본문/속성/스크립트)에 맞는 출력 인코딩(HTML Entity Encoding)을 적용하며, 사용자·관리자 화면 모두 동일하게 적용한다.
2. 입력 단계에서 < > " ' 등 위험 문자를 필터링/치환하고 허용 태그 화이트리스트를 적용한다.
3. Content-Security-Policy(CSP) 헤더로 인라인 스크립트 실행을 제한한다.
4. 세션 쿠키에 HttpOnly 속성을 적용하여 스크립트를 통한 쿠키 접근을 차단한다.

7. 취약한 비밀번호 저장

취약점 내용	회원 비밀번호가 salt 없는 SHA-256 단일 해시로 저장되어, 사전 공격(rockyou 등)으로 다수 계정의 비밀번호가 손쉽게 복원되는 취약점.
위험도	중

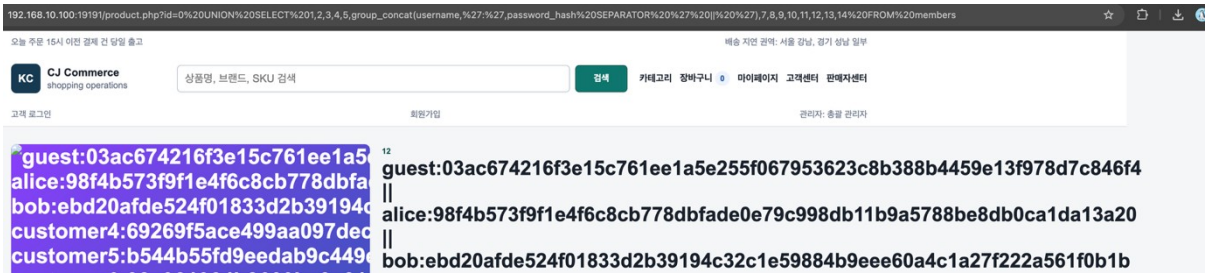
산정 근거	**회원 비밀번호가 salt 없는 SHA-256 단일 해시로 저장되어 있어 DB 유출 시 사전 기반 공격으로 평문 비밀번호가 복원될 수 있으며, 본 진단에서도 일부 계정의 복원이 가능함을 확인하였다. 다만 단독으로 즉시 시스템 침해로 이어지기보다 DB 유출 이후 피해를 확대시키는 요인이므로 위험도를 '중'으로 산정하였다.**

7.1 취약점 개요

URL	http://192.168.10.100:19191 (members 테이블)
파라미터	password_hash

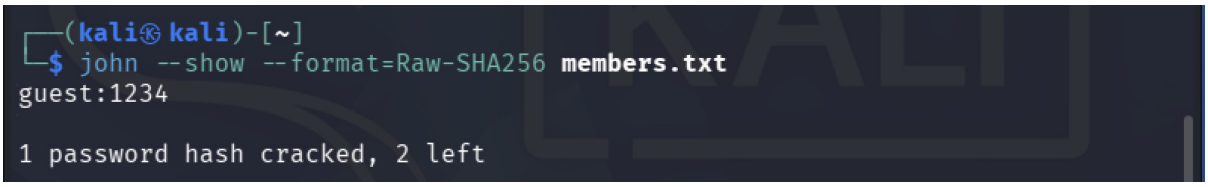
7.2 상세설명

Step 1) SQL Injection 으로 members 테이블의 password_hash 를 추출한다. 형식은 64 자리 SHA-256(무 salt)이다.



추출된 password_hash(SHA-256) 목록 화면

Step 2) john the ripper(--format=Raw-SHA256) + rockyou 사전으로 크랙 시 admin(password), test123(test123) 등 다수 계정이 즉시 복원된다.



john 크랙 결과(평문 비밀번호 복원) 화면

7.3 조치 방안

1. 비밀번호는 계정별 고유 salt 를 적용한 느린 해시 알고리즘(bcrypt, scrypt, Argon2)으로 저장한다.
2. 비밀번호 복잡도 정책과 재사용 방지 정책을 적용한다.
3. 유출 대비 pepper 적용 및 주기적 알고리즘 강화를 검토한다.

8. 관리자 페이지 통제 미흡

취약점 내용	관리자·판매자 관리 영역의 권한 검증이 미흡하여, 관리 화면에서 전체 고객 주문·쿠폰·리뷰·문의 등 민감 데이터를 제한 없이 열람할 수 있는 취약점.
위험도	중
산정 근거	**관리자 또는 판매자 기능에 접근할 경우 고객 주문정보, 쿠폰 코드, 리뷰, 문의 내역 등 주요 데이터를 제한 없이 열람할 수 있었다. 단독 취약점보다는 인증 우회 및 계정 탈취와 연계될 때 영향이 커지는 구조이나, 접근 가능한 민감정보 범위가 넓으므로 위험도를 '중'으로 산정하였다.**

8.1 취약점 개요

URL	http://192.168.10.100:19191/admin , http://192.168.10.100:19191/seller
파라미터	-

8.2 상세설명

Step 1) 취약 계정 및 재설정 우회로 관리자 패널에 접근하여 전체 고객 주문(이름·CUST 번호·금액), 쿠폰 코드(FREE100 100% 할인 포함), 리뷰, 문의를 제한 없이 열람한다.

관리자 주문 관리 - 전체 고객 주문/개인정보 열람 화면

Step 2) 판매자센터(seller01)에 접근하여 상품 등록/이미지 업로드 등 판매자 기능을 사용한다.

관리자 쿠폰 관리 - 쿠폰 코드 노출 화면

8.3 조치 방안

1. 기능/데이터 접근 시 서버 측에서 권한(role)을 반드시 재검증한다.
2. 관리자 기능은 별도 인증·네트워크 분리 및 접근 IP 제한을 적용한다.
3. 중요 데이터 조회/내보내기 시 감사 로그를 남기고 비정상 대량 조회를 탐지한다.