Acunetix-php

시나리오 기반 모의침투 결과보고서

	시나리오 기반 모의해킹 결과 보고서		
	Category	문서 버전	문서 최종 수정일
	Development Report	1.0	2025.04.25

팀	**명**	**:**	**3**		**조**
팀	**장**	**:**	**장**	**종**	**일**
팀	**원**	**:**	**김**	**거**	**빈**
팀	**원**	**:**	**김**	**연**	**수**
팀	**원**	**:**	**김**	**현**	**서**
팀	**원**	**:**	**문**	**서**	**영**

문서 정보 / 수정 내역

File Name	시나리오 기반 모의해킹 결과 보고서
원안작성자	
수정작업자	

수정 날짜	대표 수정자	Revision	추가/수정 항목	내 용

목차

그림 목차

표 목차

1. 개요

1.1. 모의해킹 정의

본 모의해킹 진단은 웹 서비스의 관련된 모든 정보 자산에 대해 취약점을 도출/분석하여 대책을 수립하기 위한 것이다. 시나리오 기반으로 해커와 동일한 환경과 조건, 기술을 가지고 모의 침투를 실행하여 취약점을 발견하고, 발견된 취약점을 점검하여 사전 예방을 통한 보안 현황 확인과 대응 방안 확립을 목적으로 한다

1.2. 수행일정/수행내역

본 모의해킹은 2025년 4월 21일부터 ~ 2025년 04월 25일까지 약 1주간 진행이 되며, 총 0.25M/M가 투입된다. Task 별 자세한 일정은 아래 표와 같다.

04월 21일(월)	04월 22일(화)	04월 23일(수)	01월 24일(목)	04월 25일(금)
환경분석 모의해킹 수행	모의해킹 수행	모의해킹 수행	모의해킹 수행	모의해킹 수행 보고서 작성

표 1 모의해킹 진단 일정

담당자	수행 범위	E-Mail

표 2 담당자별 수행 내역

1.3. 수행 대상 및 장소

본 모의해킹은 아래 표 3의 testphp.vulnweb 웹 서비스를 대상으로 진단하며, Task별로 해당 대상에 대해 점검을 진행했다.

구분(Task)	대상 도메인	대상 IP 정보	서비스
외부 모의해킹	www.testphp.vulnweb.com	44.228.249.3	PHP 웹 서비스

표 3 모의해킹 수행 범위

본 모의해킹은 외부 IP대역에서 진행하였으며, 취약점 점검 수행자의 IP는 담당자에게 사전 전달한다. 점검 수행 시 장애가 발생하면 담당자에게 즉시 보고하게 된다.

구분(Task)	수행자 IP	장소
외부 모의해킹	192.168.0.2~192.168.0.254	자택

표 4 모의해킹 장애 처리

1.4. 수행 단계별 방법

본 모의해킹은 단계별로 정보 수집부터 보고서 작성까지 아래 그림 1-1의 과정을 통해 진행된다.

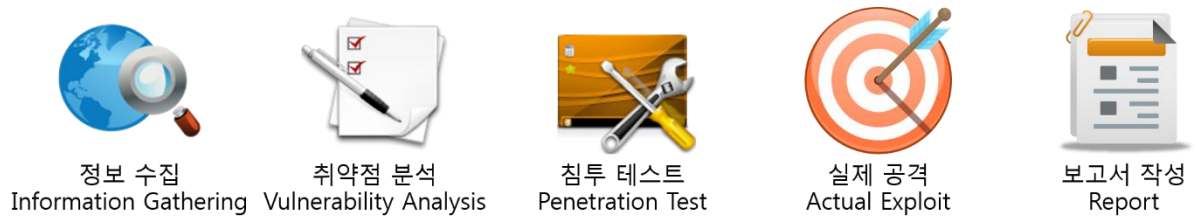

그림 1-1 모의해킹 수행 단계

각 수행 단계별 요약 설명은 아래 표 5와 같다.

수행 단계	설명
정보 수집	대상에 대한 서버/네트워크/서비스에 대한 불필요한 서비스 접근 가능성, 외부에서 파악할 수 있는 정보들을 수집하는 단계
취약점 분석	각 네트워크 구간별로 적합한 취약점 스캔 도구를 이용하여 발생할 수 있는 취약점에 대한 정보를 수집하는 단계 (단, 네트워크 장비/서비스에 장애를 유발할 수 있는 경우에는 제외)
침투 테스트	취약점 정보 수집 및 분석 단계를 통해 획득한 정보를 기반으로 수동 점검하여 시스템 내부까지 침투할 수 있는지 시나리오 기반으로 접근하는 단계
실제 공격	취약점이 확인되었을 때 공격에 의한 시스템 보안 위협이 시스템 및 비지니스 측면에서 어느 정도의 영향을 미칠 수 있는지 분석하는 단계
보고서 작성	도출된 취약점에 대한 총평/영향도/상세분석/보안가이드가 포함된 보고서를 작성하는 단계

표 5 수행 단계 설명

1.5. 점검 항목

점검 항목은 OWASP TOP 10, SANS TOP 25, KISA 48대 취약점 항목 등을 기반으로 제작된 자사의 취약점 점검 방법론을 이용하여 진행된다.

순번	분류	코드	점검 항목
1	계정정보 추측 및 대입	BP-001	취약한 패스워드 설정 여부
		BP-002	어플리케이션/장비 기본 패스워드 설정 여부
2	인증 우회	BP-003	쿠키 재사용 (Replay Attack) 여부
		BP-004	중요페이지 세션/인증/접근 체크 여부
		BP-005	클라이언트 인증 우회 여부 (Javascript 우회)
3	파라미터 조작	BP-006	URL 정보 내 파라미터 위/변조 여부
		BP-007	필드 값 조작에 따른 검증 여부
4	XSS (CSRF) 취약점	BP-008	악의적인 스크립트 필터링 여부 (POST 메소드)
		BP-009	URL 파라미터 필터링 여부 (GET 메소드)
		BP-010	XST, TRACE 옵션 허용 여부
		BP-011	CSRF 취약점 허용 여부
5	에러 메시지 처리	BP-012	에러 메시지를 통한 중요/불필요한 정보 유출
6	디렉터리 리스팅 취약점	BP-013	디렉터리 리스팅 여부
7	관리자 페이지 추측	BP-014	페이지 내 관리자 페이지 링크 여부
		BP-015	관리자 페이지 접근 여부
8	페이지내 중요 정보 노출	BP-016	중요 개인정보 노출 여부
		BP-017	쿠키 값 내 중요정보 유출 여부
		BP-018	데이터베이스 관련 정보 유출 여부
		BP-019	중요정보 평문전송
9	불필요 파일 존재	BP-020	불필요한 페이지 존재 여부
		BP-021	백업, 압축 등 불필요 파일 존재 여부
		BP-022	테스트 페이지, 데모 페이지 삭제 여부
10	파일 다운로드 취약점	BP-023	입력 값 검증 미흡으로 파일 다운로드 공격 여부
11	파일 업로드 취약점	BP-024	입력 값 검증 미흡으로 파일 업로드 공격 여부
12	부적절한 Include 취약점	BP-025	부적절한 Include 허용 여부
13	URL 강제 호출	BP-026	비인가 페이지 강제 호출 여부
14	SQL Injection	BP-027	SQL Injection 허용 여부
15	최신 취약점 미패치	BP-028	보안에 취약한 오래된 어플리케이션 사용 여부
16	부적절한 서버 설정	BP-029	서버 보안 설정 여부
17	법적 요구사항 검토	BP-030	개인정보보호법에 의한 적절성 여부

표 6 점검 항목

1.6. 점검 도구

본 모의해킹을 수행하면서 사용된 도구는 아래 표와 같다.

도구 이름	용도	사이트
Nmap	취약점 분석	http://nmap.org/
DirBuster	디렉터리 스캔	http://sourceforge.net/projects/dirbuster/
Nikto	취약점 분석	https://cirt.net/Nikto2
BeEF	침투 테스트	http://BeEFproject.com/
Wireshark	네트워크 패킷 분석	https://www.wireshark.org/
SQLMap	SQL Injection 테스트	www.sqlmap.org/
BurpSuite	프록시	http://portswigger.net/Burp/
Metasploit	침투 테스트	www.metasploit.com

표 7 점검 도구

2. 결과 요약

2.1. 총평

본 모의해킹 진단은 내, 외부 서비스를 대상으로 이루어졌다. 이 중 인증 우회, XSS, 디렉터리 리스팅, 중요 정보 노출 등은 서비스 운영에 중대한 영향을 미칠 수 있는 고위험 취약점으로 분류되며, 우선적인 보완 조치가 요구된다. 특히, testphp 웹 서비스는 인증 관련 토큰이 영구적으로 유지되어 세션 하이재킹 공격에 취약하며, 디렉터리 리스팅을 통해 내부 설정 파일이 노출되어 있어 계정 탈취 및 시스템 구조 분석 가능성이 있다. 또, Search, Guestbook 등 입력 필터링이 미흡한 구간에서는 XSS 공격이 가능하여, 악성 스크립트 삽입을 통한 권한 상승 및 사용자 정보 탈취가 우려된다. 에러 메시지 처리 미흡, 불필요한 백업 파일의 존재 등은 정보 수집 단계에서 활용될 수 있으며, SQL Injection 및 URL 강제 접속 시도에 대한 대비도 부족하다. 따라서, 각 항목별로 명확한 대응 방안을 수립하고, 시스템 관리자 및 개발자의 보안 교육과 주기적인 점검 프로세스 도입이 필요하다. 주요 취약점은 아래 표 8과 같다.

취약점	요약
계정정보 추측 및 대입	해당 취약점은 발견되지 않음
인증 우회	testphp 웹 서비스에서 세션 토큰을 영구적으로 사용하고 있어 인증 우회 가능성이 있으며, 쿠키 재사용 여부를 확인함
파라미터 조작	product 페이지의 cart 추가 버튼에서 파라미터 변조로 상품 금액 변조 여부 확인
XSS 취약점	Search, Your profile, Our guestbook 서비스에 악성 스크립트 삽입 및 다른 사용자 권한 획득이 가능하며, 바이러스 배포 가능
에러 메시지 처리	존재하지 않는 파일 또는 잘못된 URL 호출 시 Nginx 버전 등의 민감한 서버 정보가 포함된 에러 메시지가 노출됨
디렉터리 리스팅	dirb 등의 도구를 통해 디렉터리 리스팅이 가능하여 내부 파일 구조 및 중요 파일이 노출
관리자 페이지 추측	해당 취약점은 발견되지 않음
페이지내 중요 정보 노출	로그인 페이지 등에서 사용자 개인정보가 포함된 정보가 불필요하게 노출됨
불필요한 파일 존재	불필요하게 남겨진 백업 파일 및 설정 파일이 외부에 노출되어 소스코드 및 인증정보 유출 위험이 존재함
파일 다운로드	해당 취약점은 발견되지 않음
파일 업로드	해당 취약점은 발견되지 않음

부적절한 Include 취약점	Burp Suite를 활용해 이미지 경로 조작을 통한 상위 디렉토리 접근 가능 여부 확인
URL 강제 호출	URL 파라미터를 이용해 강제 호출 가능 여부 확인
SQL Injection	로그인 페이지, URL 파라미터, 검색 페이지에서 SQL Injection 취약점 발견
최신 취약점 미패치	Chrome 개발자 도구를 통해 PHP 코드 노출이 확인되어, 소스 관리 및 취약점 패치 미비 가능성 존재
부적절한 서버 설정	잘못된 서버 설정으로 인한 디렉토리 리스팅 등이 가능
법적 요구사항 검토	해당 취약점은 발견되지 않음

표 8 모의해킹 진단 내역

2.2. 취약점 요약

본 모의해킹 결과, 인증 우회, XSS, 디렉터리 리스팅, 중요 파일 및 정보 노출, 불필요한 백업 파일 존재, 에러 메시지 노출, SQL Injection 가능성 등 다양한 고위험 취약점이 발견되어 시스템 구조 노출, 사용자 정보 탈취, 권한 상승 등의 공격으로 악용될 수 있으므로 전반적인 보안 설정 강화 및 취약 구간에 대한 즉각적인 대응이 요구된다.

3. 상세 수행 내역

3.1. 정보 수집

그림 3-1 모의해킹 홈페이지 (testphp.vulnweb)

위 그림 3-1는 시나리오 기반 모의해킹을 수행할 PHP로 구성된 Acunetix Test site이다.

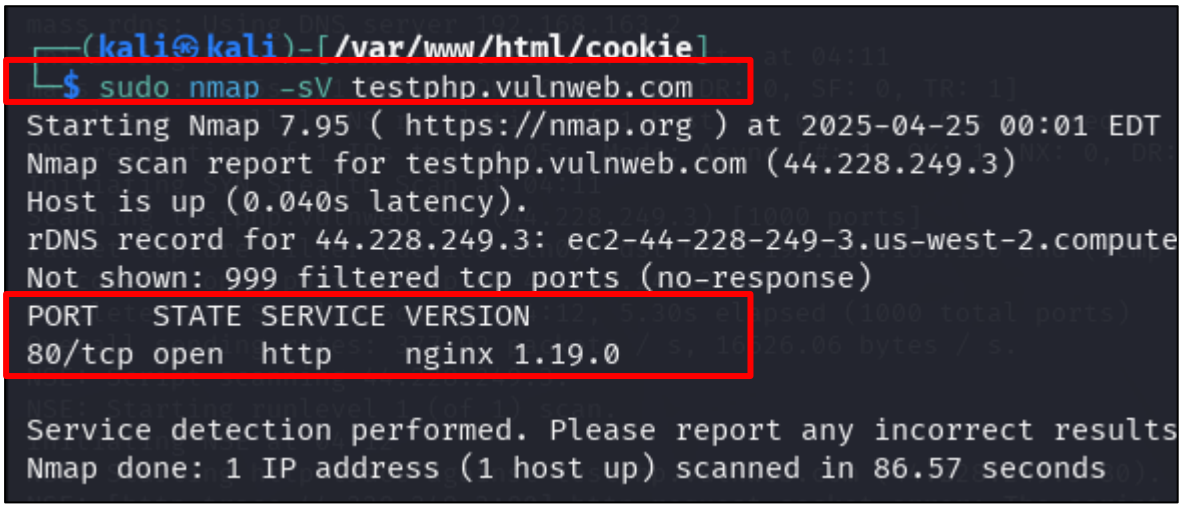

그림 3-2 Nmap을 통한 기본적인 스캔

취약점 분석 도구로 널리 알려진 Nmap의 기본 옵션을 통해서 대상 서버에서 실행중인 서비스

관련 정보를 획득할 수 있다. 대표적인 웹 서비스인 HTTP가 TCP(80)포트로 실행되고 있다.

그림 3-3 Nikto를 통한 기본적인 스캔

Nikto는 웹 서비스 취약점을 자동 점검 해주는 Perl로 작성된 오픈 소스 도구이다. 다양한 종류의 점검을 지원하고 있으며, OSVDB 정보도 보여준다. 위 그림 3-3는 Nikto의 스캔 결과로 admin 페에이지를 발견했다.

3.2. 인증 우회

분류	코드	점검 항목
인증 우회	BP-001	쿠키 재사용 (Replay Attack) 여부

표 9 인증 우회 점검 항목

3.2.1. 쿠키 획득 가능 여부 검증

❑ 쿠키 재사용 (Replay Attack) 여부

공격자가 이미 발급되어 유효한 세션 쿠키를 획득한 뒤, 이를 자신의 요청에 헤더로 포함시켜 서버에 전송하면 서버는 이 사용자를 이미 인증된 정상 사용자로 인식해 버리는 취약점이다. 페이지 내 XSS 취약점을 이용해 document.cookie 값을 획득하여 탈취한 쿠키로 직접 세션을 재사용하여 쿠키 재사용이 가능하다.

그림 3-4 XSS 취약점을 통한 쿠키 재사용 시도

그림 3-5 document.cookie 추출하는 a.js 파일

위 그림 3-4을 보면 Our guestbook 페이지에서 공격자가 그림 3-4의 스크립트 태그를 입력한 뒤 다른 방문자들이 해당 게시물을 열람하면 스크립트가 자동으로 실행되면서 그림 3-5에서 볼 수 있는 악성 코드가 담긴 a.js 파일을 호출하여 document.cookie를 빼내는 악성 행위를 진행한다.

그림 3-6 document.cookie 추출 성공

위 그림 3-6는 쿠키를 추출하는 공격 코드가 담긴 파일을 이용해 쿠키 추출에 성공한 모습이다

3.3. 파라미터 조작

분류	코드	점검 항목
인증 우회	BP-002	필드값 조작에 따른 검증 여부

표 10 파라미터 조작 점검 항목

3.3.1. 필드값 조작 가능 여부 확인

❑ 필드값 조작에 따른 검증 여부

공격자가 클라이언트 쪽에서 전달된 폼 데이터를 서버가 신뢰하고 검증 없이 처리하는 취약점을 이용하여 관리자가 의도하지 않은 값을 임의로 주입·변경할 수 있는 취약점이다. Burp suite를 이용하여 Request의 Body 부분의 필드값을 조작하여 장바구니 내에 있는 상품의 가격을 임의로 조작한다.

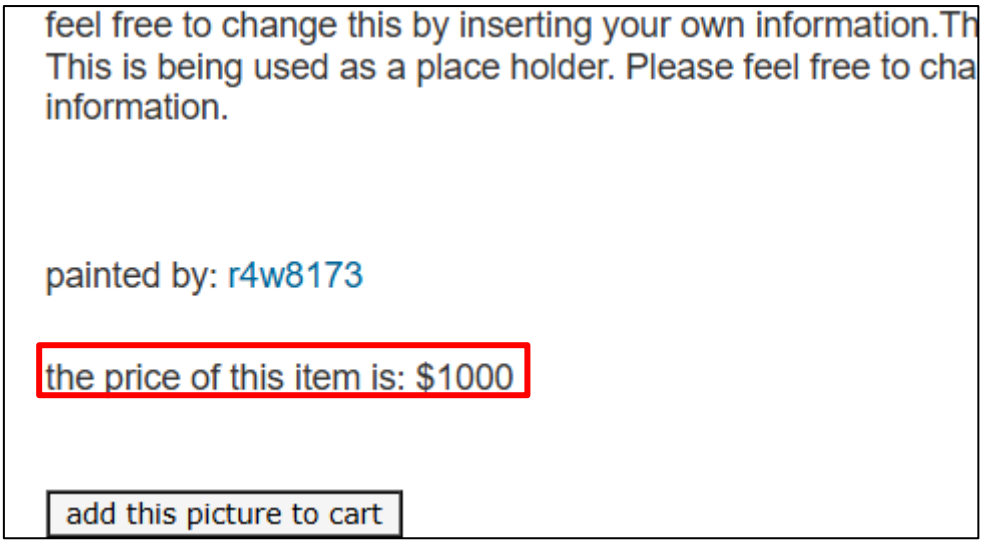

그림 3-7 필드값 조작 전 기존 가격

위 그림 3-7은 필드값을 조작하기 전 상품의 기존 가격이다.

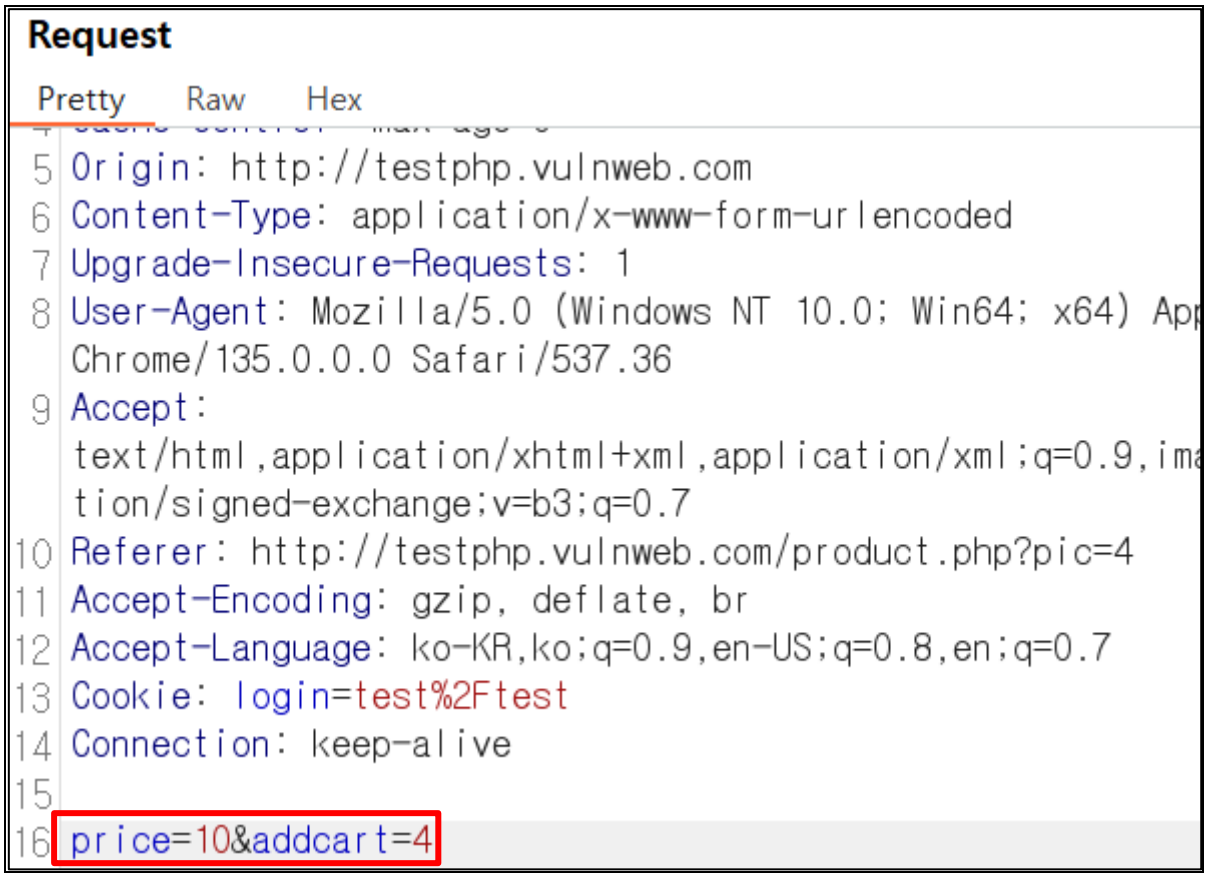

그림 3-8 Request Body 부분의 필드값 10으로 조작

위 그림 3-8은 Burp Suite를 이용해서 Request의 Body 부분의 필드값을 $1000에서 $10으로 조작을 시도해서 필드값 조작에 따른 검증 여부를 확인하는 과정이다.

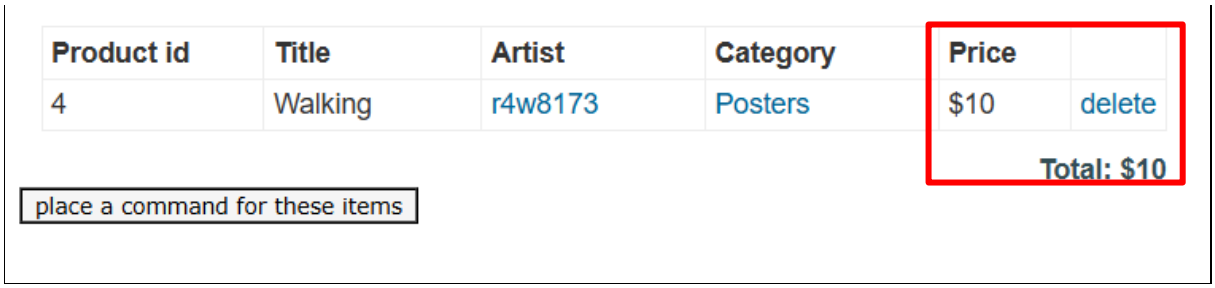

그림 3-9 document.cookie 추출하는 a.js 파일

위 그림 3-9는 장바구니에 들어가있는 상품의 가격 조작이 성공한 모습이다.

3.4. XSS 취약점

분류	코드	점검 항목
XSS 취약점	BP-002	악의적인 스크립트 필터링 여부 (POST 메소드)

그림 3-10 XSS 취약점 점검 항목

3.4.1. 악의적인 스크립트 필터링 가능 여부 검증

❑ 악의적인 스크립트 공격

웹 애플리케이션이 POST 방식으로 넘어오는 입력 데이터에는 GET과 달리 URL 쿼리스트링이 아닌 HTTP 바디(body)에 파라미터가 담긴다. 그러나 스크립트 필터링 로직을 GET 파라미터에만 적용하고, POST 파라미터 검사는 소홀히 하거나 누락할 시 공격자는 XSS를 발생시킬 수 있다.

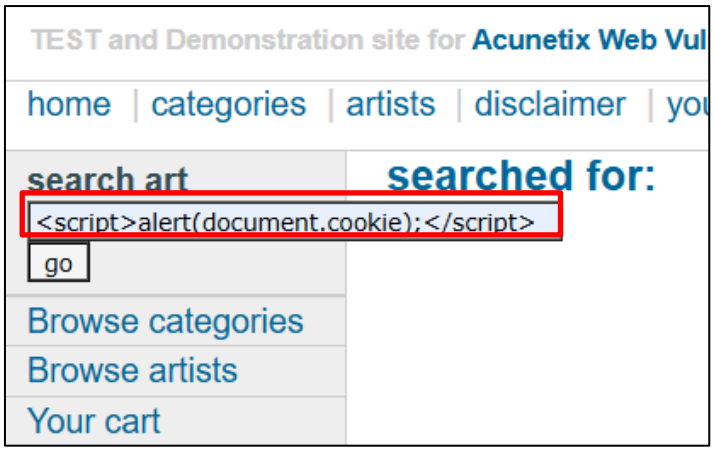

그림 3-11 취약한 패스워드 회원가입 시도

위 그림 3-11을 보면 <script>alert(document.cookie);</script>을 통해서 그 결과가 필터링·인코딩 없이 페이지에 반영되어 즉시 실행되는 XSS 취약점을 실행한다.

페이지	스크립트
Your profile	Name 필드 내 <script>alert('XSS');</script>
Your profile	Name 필드 내 <iframe src="http://210.95.67.235:8181/"></iframe>
Our guestbook	<script>alert('XSS');</script>
Our guestbook	<a href=http://example.com/>test
Our guestbook	<img src="http://testphp.vulnweb.com/showimage.php?file=./pictures/1.jpg>

표 11 기타 XSS 취약점 점검 목록

위 표 11는 다른 페이지 내에 존재하는 XSS 취약점 공격에 성공한 스크립트 목록이다.

그림 3-12 User/Password 확인 성공

위 그림 3-12은 악의적 스크립트 공격을 통해 쿠키 탈취에 성공한 모습이다.

3.5. 에러메시지 처리

분류	코드	점검 항목
에러메시지 처리	BP-003	에러 메시지를 통한 중요/불필요 정보 유출

표 12 에러메시지 처리 점검 항목

3.5.1. 중요/불필요 정보 유출 여부

❑ 에러 메시지를 통한 중요/불필요 정보 유출

에러 메시지를 통해 불필요한 내부 정보가 유출되는 경우가 존재한다. SQL 문법 오류, 제약조건 위반, 테이블·컬럼 이름, MySQL/PostgreSQL 버전 정보 등의 에러 메시지에 의해서 취약점이 노출될 수 있으며, nginx/1.19.0, Apache/2.4.46 같은 서버 종류·버전 정보가 응답 헤더나 에러 페이지에 노출되어 중요하거나 불필요한 정보가 유출될 수 있다.

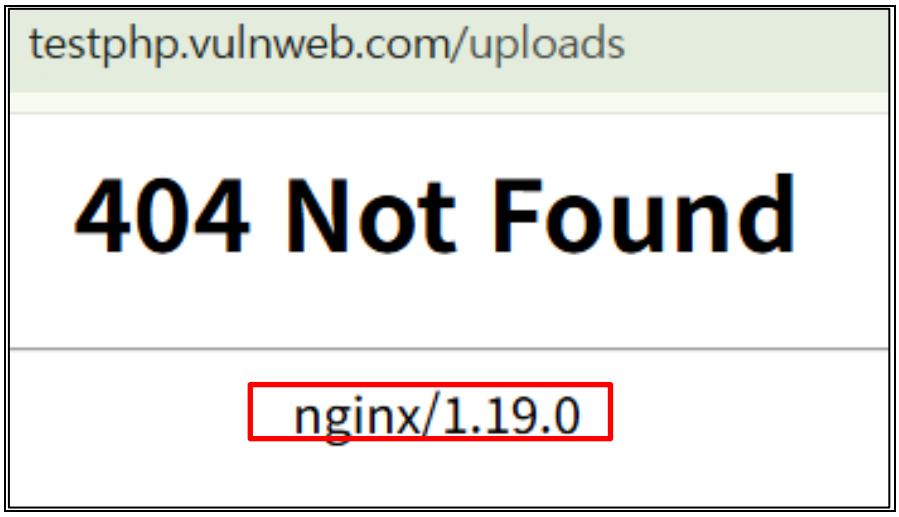

그림 3-13 웹 서버 버전 노출

위 그림 3-13을 보면 nginx/1.19.0 이라는 웹 서버 버전을 노출하고 있다. 공격자는 이 버전에서 알려진 CVE(취약점) 목록을 조회해 표적 공격을 준비할 수 있다.

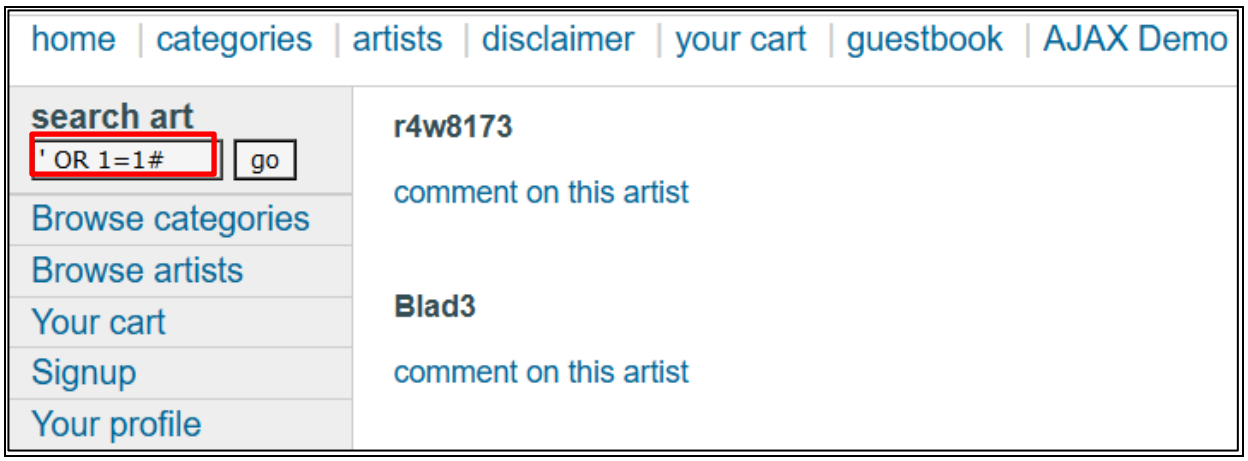

그림 3-14 데이터베이스 오류 메시지 출력 시도

위 그림 3-14는 'OR 1=1#을 입력하여 데이터베이스 오류 메시지 출력을 시도하는 모습이다.

그림 3-15 데이터베이스 종류 노출

위 그림 3-15는 데이터베이스 오류 메시지 출력에 성공하여 데이터베이스의 종류가 MySQL인 것을 알아낸 모습이다.

3.6. 디렉터리 리스팅 취약점

분류	코드	점검 항목
디렉터리 리스팅 취약점	BP-013	디렉터리 리스팅 여부

표 13 에러메시지 처리 점검 항목

3.6.1. 디렉터리 리스팅 취약점

❑ 디렉터리 리스팅 자동 분석

웹 서버에서 디렉터리 리스팅 기능이 활성화되어 있는 경우, 인증 없이도 공격자가 특정 경로의 내부 구조 및 파일 목록을 열람할 수 있다. 이 취약점은 공격자가 민감한 파일(예: 백업 파일, 설정 파일, 데이터베이스 덤프 등)을 직접 다운로드하거나, 서버 구조를 파악해 추가적인 공격 벡터를 탐색할 수 있도록 돕는다.

```
┌──(kali㉿kali)-[/var/www/html/cookie]
└─$ sudo dirb http://testphp.vulnweb.com/
[sudo] password for kali:

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Thu Apr 24 00:39:07 2025
URL_BASE: http://testphp.vulnweb.com/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://testphp.vulnweb.com/ ----
==> DIRECTORY: http://testphp.vulnweb.com/admin/
+ http://testphp.vulnweb.com/cgi-bin (CODE:403|SIZE:276)
+ http://testphp.vulnweb.com/cgi-bin/ (CODE:403|SIZE:276)
+ http://testphp.vulnweb.com/crossdomain.xml (CODE:200|SIZE:224)
==> DIRECTORY: http://testphp.vulnweb.com/CVS/
+ http://testphp.vulnweb.com/CVS/Entries (CODE:200|SIZE:1)
+ http://testphp.vulnweb.com/CVS/Repository (CODE:200|SIZE:8)
+ http://testphp.vulnweb.com/CVS/Root (CODE:200|SIZE:1)
+ http://testphp.vulnweb.com/favicon.ico (CODE:200|SIZE:894)
==> DIRECTORY: http://testphp.vulnweb.com/images/
+ http://testphp.vulnweb.com/index.php (CODE:200|SIZE:170)
==> DIRECTORY: http://testphp.vulnweb.com/pictures/
==> DIRECTORY: http://testphp.vulnweb.com/secured/
==> DIRECTORY: http://testphp.vulnweb.com/vendor/
```

그림 3-16 dirb 도구를 사용하여 취약점 사이트 자동 분석

칼리 리눅스 환경에서 dirb 도구를 활용해 위의 그림 3-16와 같이 www.testphp.vulnweb.com 사이트를 대상으로 디렉터리 브루트포싱을 수행한 결과이다.
/admin/, /images/, /vendor/, /pictures/등 일부 경로에서 디렉터리 인덱싱이 허용되어 있는 것을 확인할 수 있다. 아래 그림 3-17부터 그림 3-19까지와 표 14부터 표 16까지는 점검 결과를 확인한 내용이다.

그림 3-17 test.php.vulnweb.com/admin

위치	파일명	설명	위험도
/admin	create.sql	데이터베이스 구조를 정의한 SQL 덤프 파일이 외부에 노출됨, 공격자는 이를 통해 테이블명, 필드명, 관계 구조를 파악하고 SQL Injection 공격이 가능함	높음

표 14 /admin 디렉터리 리스팅 점검 결과 분석

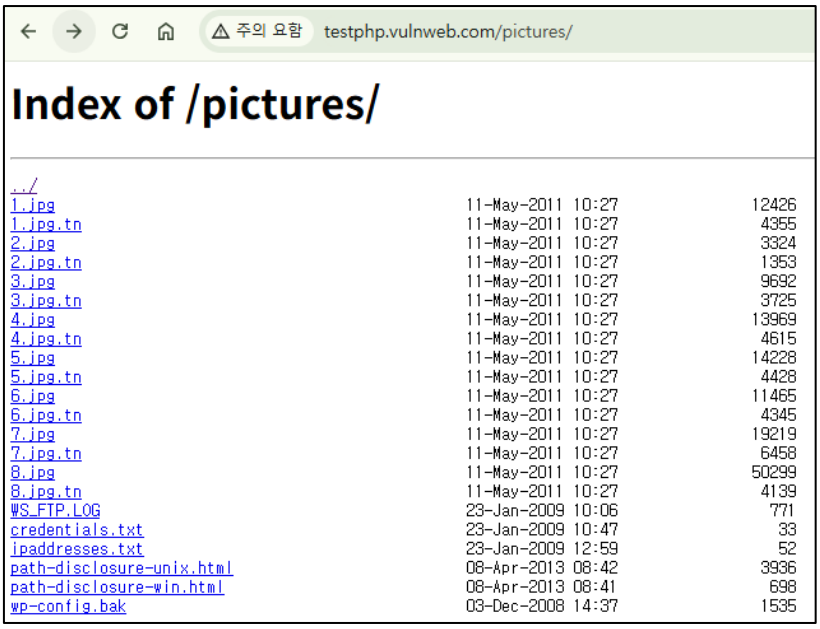

그림 3-18 dirb 도구를 사용하여 취약점 사이트 자동 분석

위치	파일명	설명	위험도
/pictures	credentials.txt	사용자 ID와 PW 정보가 저장되어 있음	매우 높음
	wp-config.bak	워드프레스 설정 백업 파일, DB 접속 정보, 테이블 접두사, 내부 경로 등 파악 가능	
	WS_FTP.LOG	경로 노출 테스트용 취약 코드	

표 15 /pictures 디렉터리 리스팅 점검 결과 분석

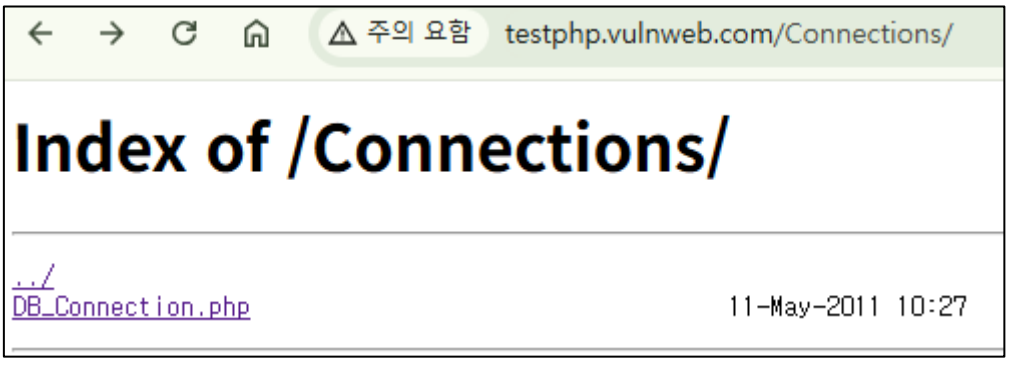

그림 3-19 test.php.vulnweb.com/Connections

위치	파일명	설명	위험도
/Connections	DB_Connection.php	DB 연결용 PHP 코드 파일이 외부에서 열람 가능, 내부에는 mysql_connect()를 이용해 계정, 비밀번호, 호스트 정보가 포함되어 있을 가능성이 있음	매우 높음

표 16 /Connections 디렉터리 리스팅 점검 결과 분석

3.7. 불필요 파일 존재

분류	코드	점검 항목
불필요 파일 존재	BP-005	백업 파일 존재
	BP-006	데모 페이지 존재

표 17 불필요 파일 존재 점검 항목

3.7.1. 불필요한 파일/페이지 존재 여부 점검

❑ 백업 파일 존재 여부

운영 중인 웹 서버에 개발·유지관리 과정에서 생성된 불필요한 백업 파일(예: index.php.bak, config.php, database.sql.zip, .git/, wp-config.php 등)이 남아 있지 않은지 확인을 해야한다. 이런 파일들이 외부에 노출되면, 소스 코드, 설정정보, DB 접속 정보 등이 그대로 유출되어 심각한 보안 사고로 이어질 수 있다.

```
<?php
// ** MySQL settings ** //
define('DB_NAME', 'wp265as');     // The name of the database
define('DB_USER', 'root');      // Your MySQL username
define('DB_PASSWORD', ''); // ...and password
define('DB_HOST', 'localhost');      // 99% chance you won't need to change this
value
define('DB_CHARSET', 'utf8');
define('DB_COLLATE', '');
```

그림 3-20 wp-config.bak 파일 내용

위 그림 3-20은 pictures 내에 존재하는 워드프레스가 데이터베이스에 연결할 때 사용하는 설정 파일(wp-config.php)의 백업본이다. 이 파일에는 데이터베이스 로그인 정보가 그대로 담겨있기 때문에 외부에 노출 될 시 위험하다.

❑ 데모 페이지 존재 여부

AJAX Demo 같은 데모용 페이지는 실제 운영 환경에선 불필요한 취약점을 늘리는 요소가 된다. 웹 페이지에 아래 그림 3-21와 같은 페이지가 남아 있으면, 공격자가 손쉽게 악용할 수 있는 엔드포인트가 될 수 있다.

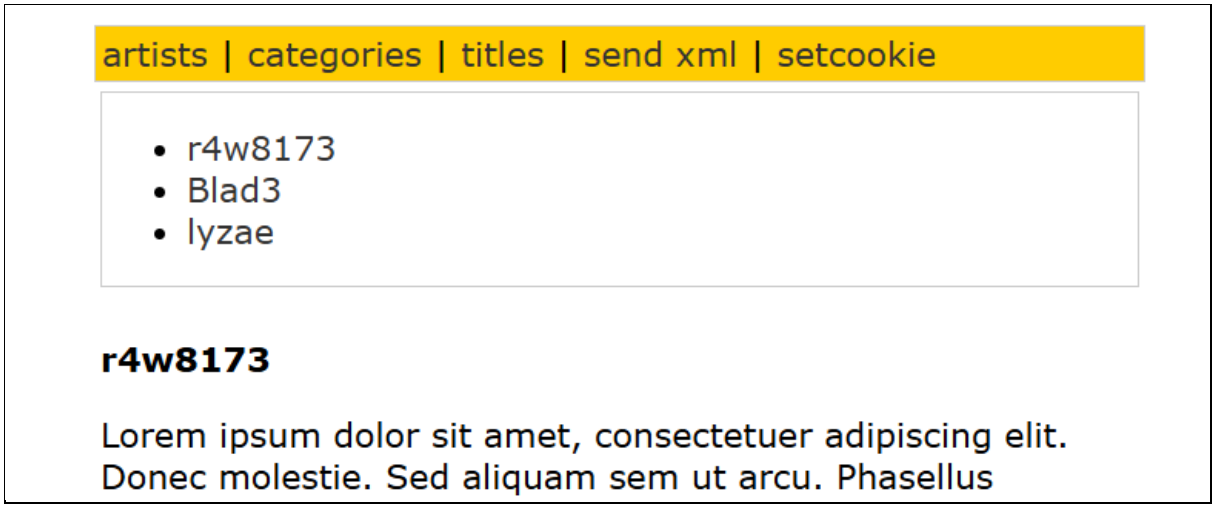

그림 3-21 AJAX Demo 페이지 내용

3.8. 부적절한 include 취약점

분류	코드	점검 항목
부적절한 include 취약점	BP-005	부적절한 include 허용 여부

표 18 부적절한 include 취약점 점검 항목

3.8.1. LFI(Local File Inclusion) 취약점 여부 점검

❑ 부적절한 include 허용 여부

사용자 입력값을 그대로 파일 경로에 적용해 서버의 임의 파일을 불러오도록 허용할 때 발생한다. file=../../etc/passwd 처럼 경로 앞에 ../ 연속으로 붙여 상위 디렉터리로 빠져나가도록 조작한다.

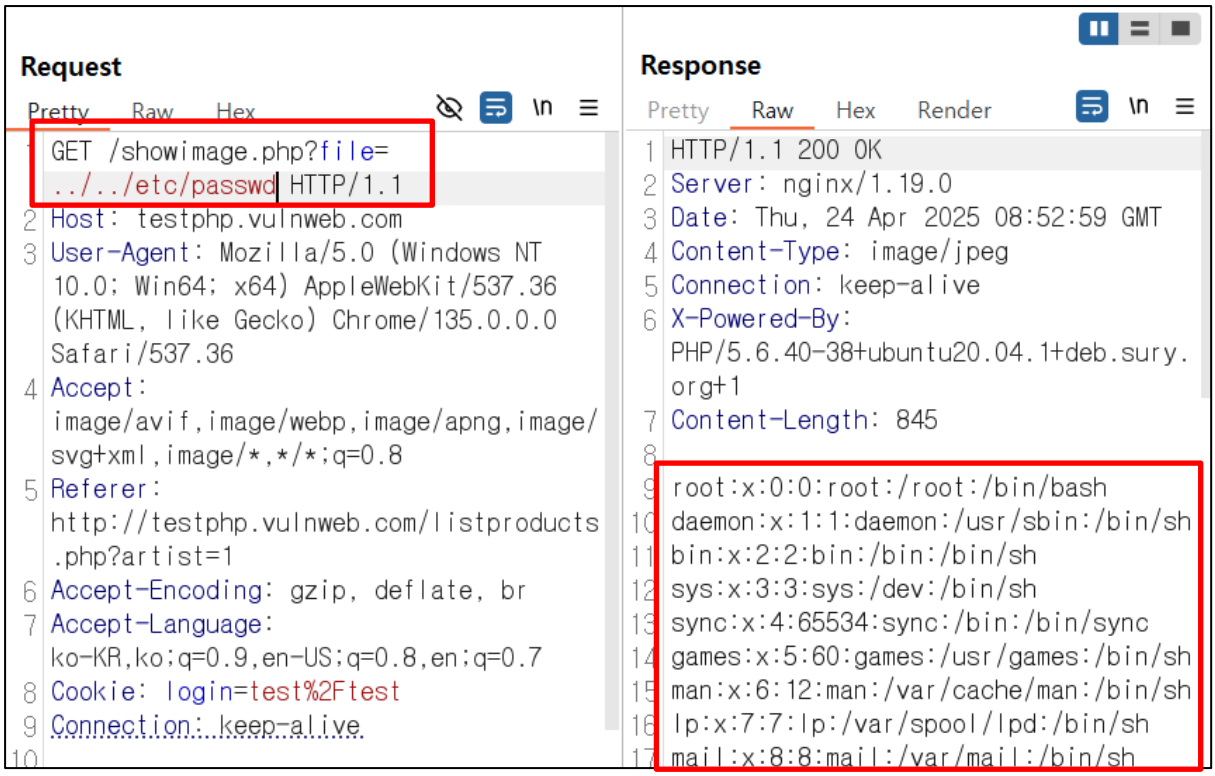

그림 3-22 LFI를 이용한 /etc/passwd 출력

위 그림 3-22는 Burp Suite에서 Local File Inclusion(LFI) 취약점을 실제로 이용해 서버의 민감 파일(/etc/passwd)을 읽어낸 모습입니다.

3.9. SQL Injection

분류	코드	점검 항목
SQL Injection	BP-027	SQL Injection 허용 여부

표 19 SQL Injection 점검 항목

3.9.1. 에러 베이스 기반 SQL Injection

❑ SQL Injection 허용 여부

사용자 입력값을 검증하지 않은 경우 쿼리문에 의도하지 않은 쿼리를 임의로 삽입했을 경우에 발생할 수 있는 취약점이다. 공격자가 쿼리를 악의적으로 주입하여 데이터베이스의 데이터를 무단으로 탈취할 수 있다.

그림 3-23 SQL Injection 삽입 수행

위 그림 3-23을 보면 search art 검색창에서 공격자가 그림 3-23의 조작된 sql문을 작성한 뒤 전송하면, 그림 3-24과 같은 에러 메시지를 출력해 버리는 취약점이다. 이 점을 통해 sql 쿼리의 내부 구조가 유츄될 가능성이 존재한다.

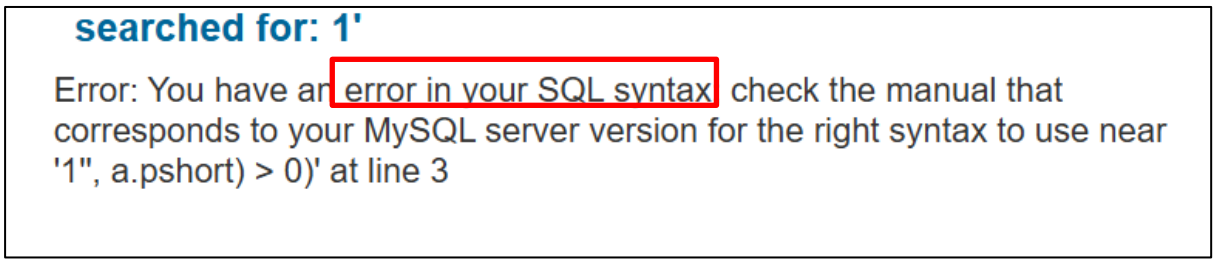

그림 3-24 SQL Injection 삽입 결과

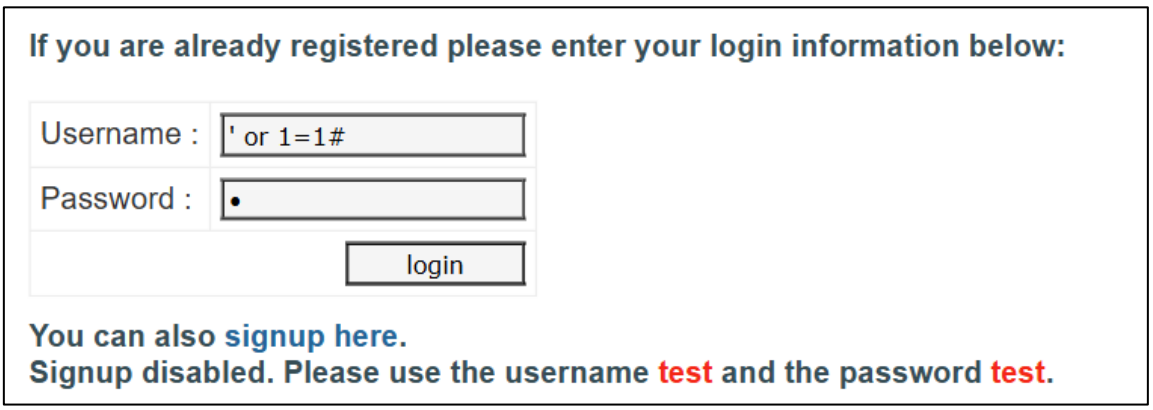

그림 3-25 SQL문을 사용하여 로그인 시도

그림 3-23의 검색창에서는 단순 SQL syntax 에러만 발생했지만, 그림 3-25처럼 username이나 password에 참이 되도록 만드는 쿼리문을 삽입했을 때, 그림 3-26의 결과처럼 오류가 아닌 로그

인 성공 화면을 보여주는 것을 확인할 수 있다.

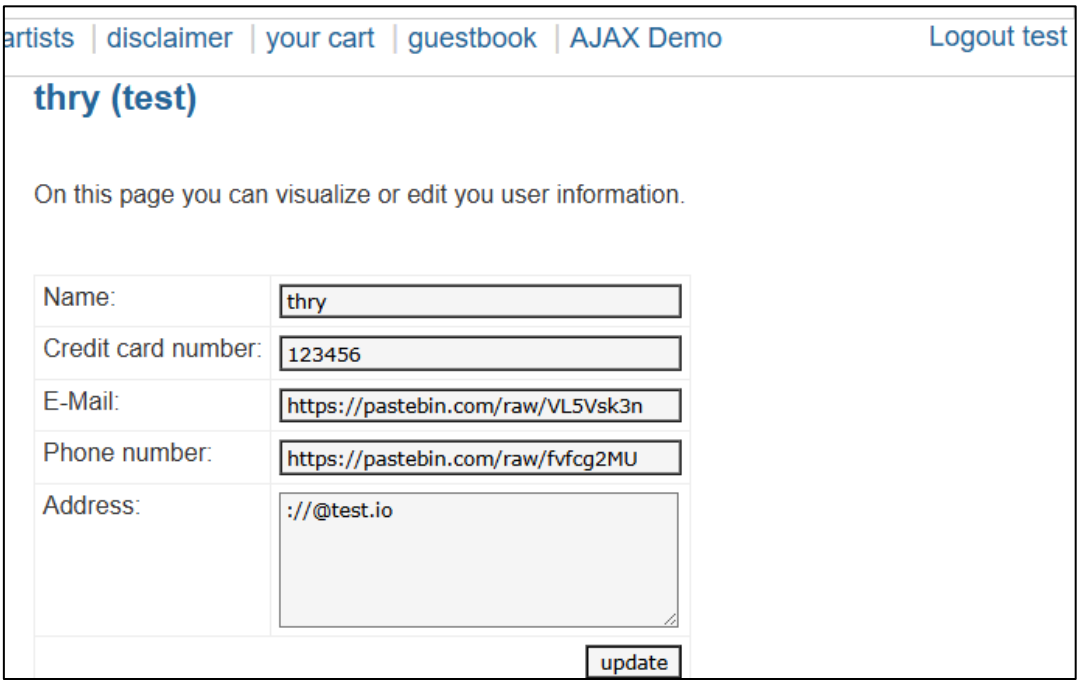

그림 3-26 로그인 성공 화면

그림 3-25의 취약점 외에도 그림 3-27의 현재 경로에서 파라미터(cat)에 쿼리문을 삽입하면 sql injection이 동작하여 쿼리문에 입력한 결과가 출력되는 것을 확인할 수 있다.
(입력 경로: http://testphp.vulnweb.com/listproducts.php?cat=1)

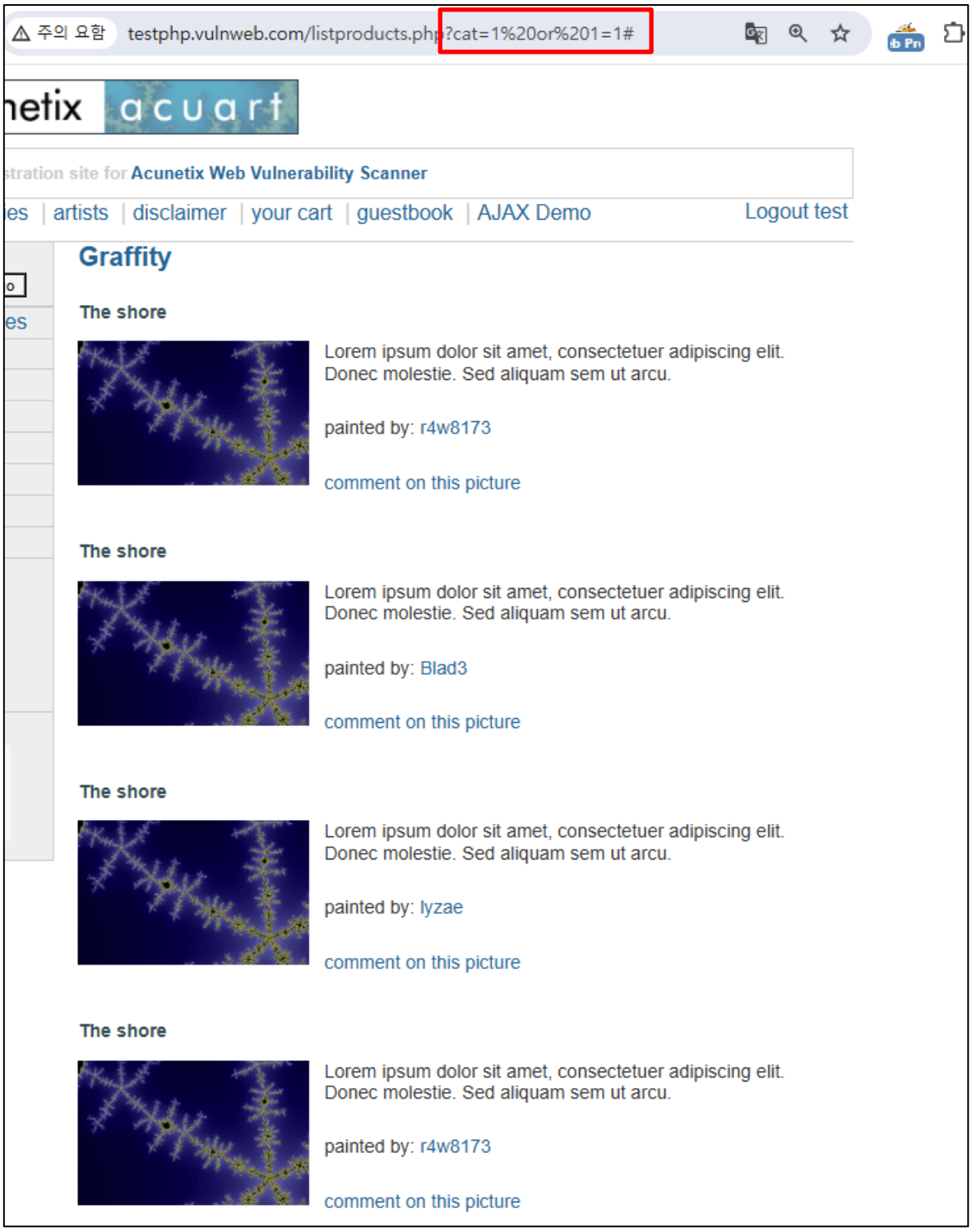

그림 3-27 sql 쿼리문 삽입 결과

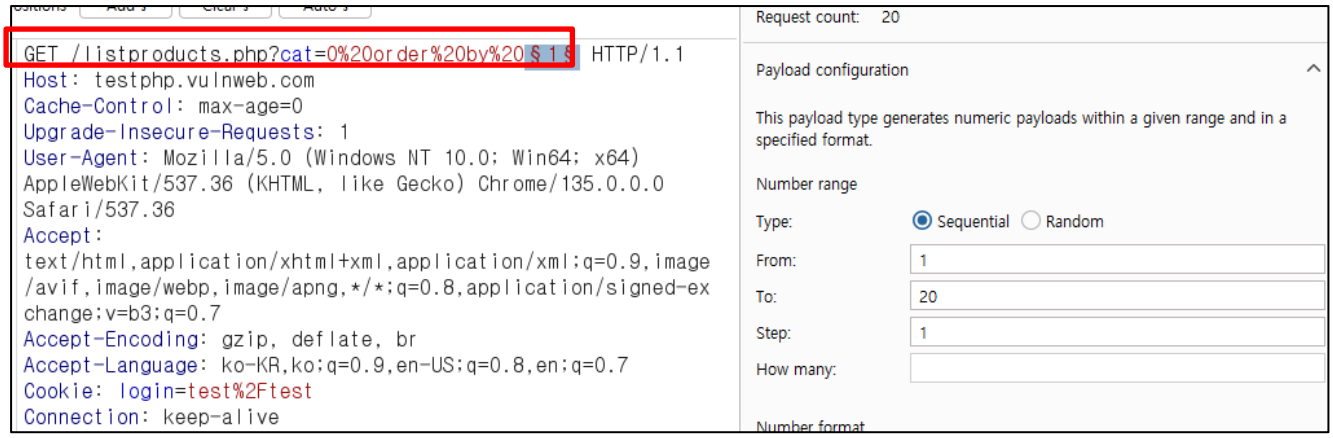

그림 3-28 brup suite로 컬럼 개수 확인

위의 그림 3-28의 결과에 따라 brup suite를 활용해 그림 3-29의 그림처럼 order by문을 사용하여 컬럼 개수를 알아냈다. 그림 3-29에 나오는 것처럼 페이로드를 보냈을 때의 응답값 길이 차이를 통해 컬럼 개수가 11개임을 추측할 수 있다.

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		200	173			4995	
1	1	200	172			4995	
2	2	200	176			4995	
3	3	200	172			4995	
4	4	200	179			4995	
5	5	200	174			4995	
6	6	200	173			4995	
7	7	200	175			4995	
8	8	200	178			4995	
9	9	200	175			4995	
10	10	200	175			4995	
11	11	200	175			4995	
12	12	200	176			2288	
13	13	200	178			2288	
14	14	200	175			2288	
15	15	200	166			2288	
16	16	200	175			2288	
17	17	200	191			2288	
18	18	200	173			2288	
19	19	200	1160			2288	
20	20	200	173			2288	

그림 3-29 sql injection 컬럼 개수 파악

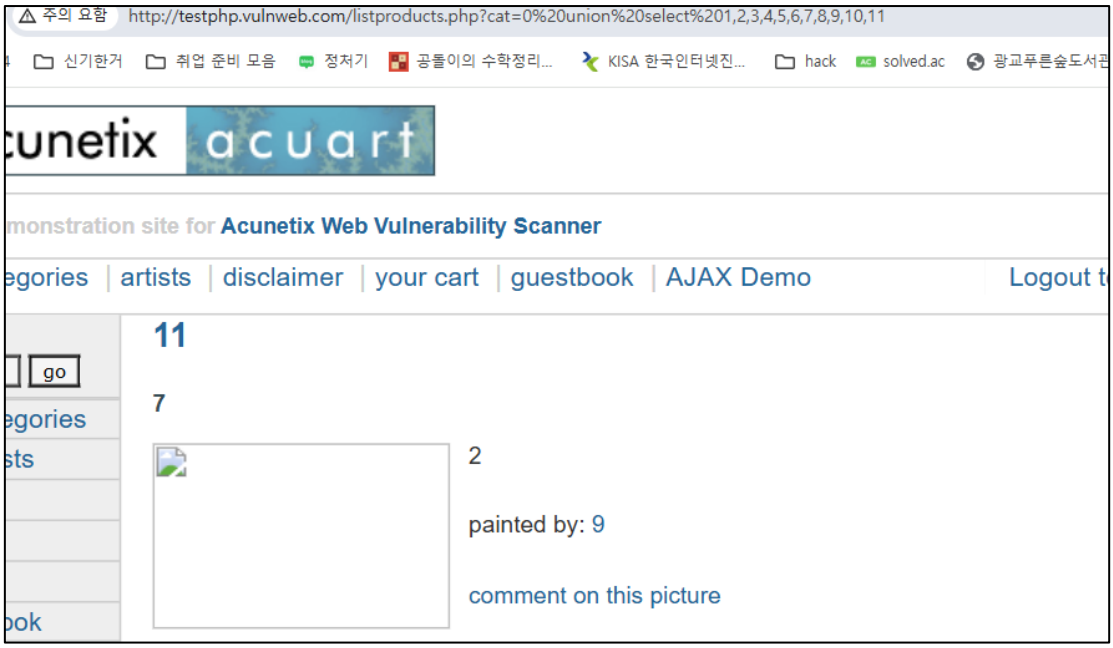

그림 3-30 sql injection 컬럼 위치 파악

위 그림 3-30은 그림 3-29에서 찾은 컬럼 개수를 바탕으로 사용 가능한 컬럼 위치를 찾은 모습이다.

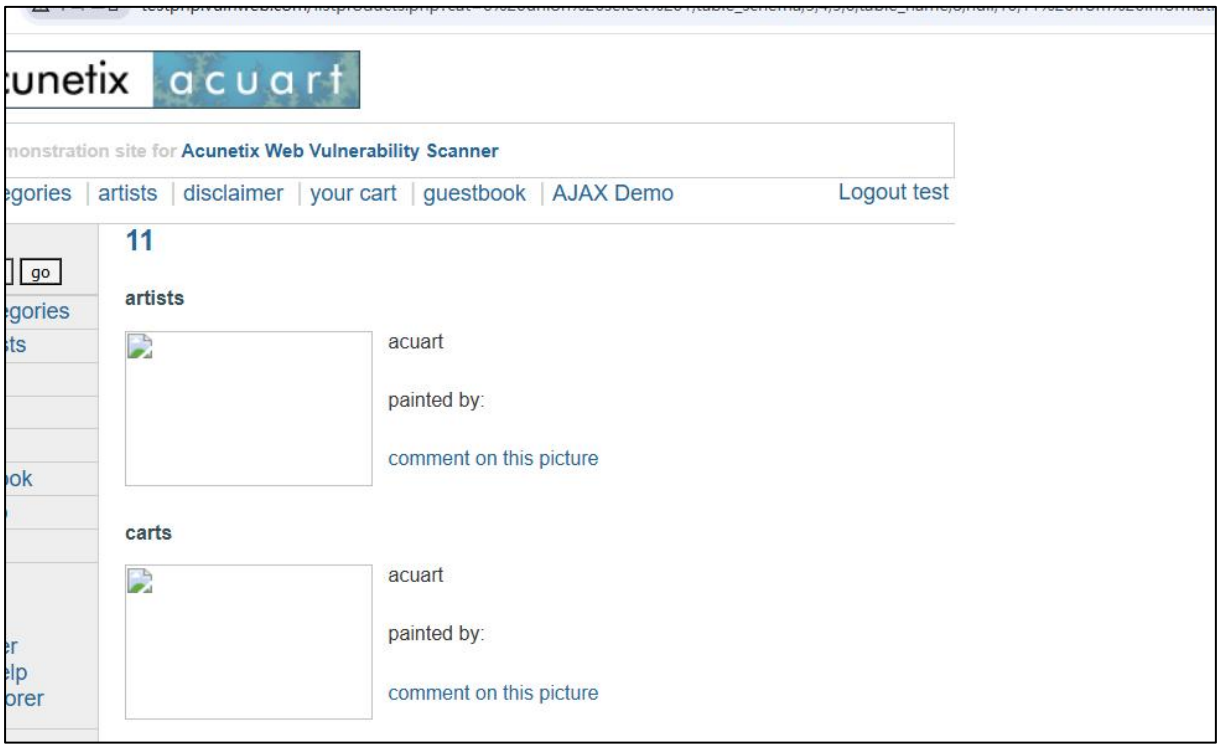

그림 3-31 sql injection 컬럼 추출 진행

위의 그림 3-31와 같이 sql injection 쿼리문을 삽입하는 과정을 통해 다음의 그림 3-32부터 그림 3-35는 sql injection 공격을 통해 얻어낼 수 있는 정보를 쿼리문과 함께 정리하였다.

그림 3-32 ubuntu 버전 정보 확인

버전 확인 및 사용자 정보 확인 쿼리문은 다음과 같다.

0 union all select 1, user(),3,4,5,6, version(), 8, 9, 10, 11#

그림 3-33 현재 데이터베이스인 acuart 테이블 리스트

acuart 테이블 확인 쿼리문은 다음과 같다.

그림 3-34 users 테이블의 컬럼 리스트

Users 테이블 컬럼 리스트 확인 쿼리문은 다음과 같다.

0 union select 1, table_schema, 3, 4, 5, 6, group_concat(column_name), 8, null, 10, table_name from information_schema.columns where table_schema='acuart' and table_name='users'

밑의 그림 3-35는 가장 중요하다고 생각되는 users의 실제 데이터 정보를 조회한 모습이다. 위에서 순서대로 (1)이 cart 정보이고, (2)가 ':'로 구분해서 name, uname, password, (3)이 ':'로 구분해서 cc, phone, address임을 확인할 수 있다.

그림 3-35 users의 정보 조회

users 정보 조회 쿼리문은 다음과 같다.

0 union select 1, concat(name, ':', uname, ':', pass), 3, 4, 5, 6, null, 8, concat(cc, ':', phone, ':', address), 10, cart from users

위의 결과와 같이 중요 정보를 쉽게 찾을 수 있음을 확인할 수 있다.

3.10. 부적절한 include 취약점

3.10.1. LFI(Local File Inclusion) 취약점 여부 점검

❑ 부적절한 include 허용 여부

사용자 입력값을 그대로 파일 경로에 적용해 서버의 임의 파일을 불러오도록 허용할 때 발생한다. file=../../etc/passwd 처럼 경로 앞에 ../ 연속으로 붙여 상위 디렉터리로 빠져나가도록 조작한다.

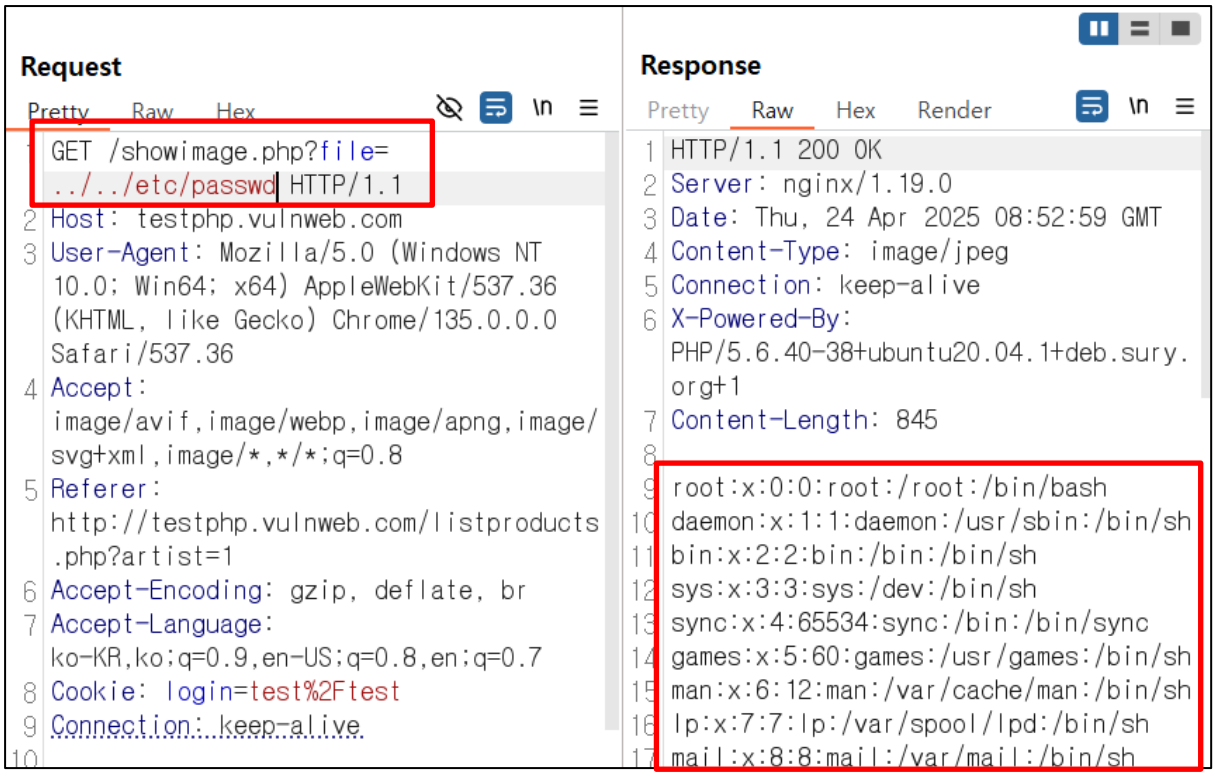

그림 3-36 LFI를 이용한 /etc/passwd 출력

위 그림 3-36은 Burp Suite에서 Local File Inclusion(LFI) 취약점을 실제로 이용해 서버의 민감 파일(/etc/passwd)을 읽어낸 모습입니다.

4. 취약점 대응방안

취약 점 대응방안은 안전행정부와 한국인터넷진흥원에서 발행한 주요정보통신기반시설 기술적 취약점 분석 평가 방법 상세가이드를 기반으로 작성했다. 대응방안은 상세 수행 내역에서 도출된 취약점과 순서는 동일하며, 관리적, 기술적, 물리적 그리고 소스코드 관점의 보안을 설명한다.

4.1. 인증 우회

4.1.1. 취약점 개요

인증 우회 취약점이란, 애플리케이션이 정상적인 로그인·인증 절차를 건너뛰고도 보호된 리소스에 접근할 수 있게 만드는 보안 취약점이다. 공격자는 이 취약점을 이용해 권한 검증 없이 관리자나 특정 사용자 권한을 획득할 수 있다.

4.1.2. 쿠키 재사용 여부 공격 대응방안

❑ 시큐어 코딩 적용

- $new = htmlspecialchars("Test", ENT_QUOTES);
- echo $new; // Test

표 20 htmlspecialschars 사용 예

PHP의 htmlspecialchars 함수는 특수 문자를 해당 HTML 엔티티로 변환하는데 사용된다. 위의 표 20처럼 공격자가 웹 애플리케이션에 악성 코드를 삽입할 수 있는 XSS 공격과 같은 잠재적인 보안 취약성을 방지하는데 사용할 수 있다.

❑ 서버 보안 설정

- HTTPS 연결에서만 전송 : session.cookie_secure = Off
- JavaScript(document.cookie)로 접근 불가 : session.cookie_httponly = Off

표 21 쿠키 속성 강화 전

쿠키 속성 강화를 안했다면 php.ini 파일 내 쿠키 속성이 위의 표 21와 같이 되어있다.

■ HTTPS 연결에서만 전송 : session.cookie_secure = **On** ■ JavaScript(document.cookie)로 접근 불가 : session.cookie_httponly = **On**

표 22 쿠키 속성 강화 후

위의 표 22는 secure 속성과 httponly 속성을 On으로 수정하여 쿠키 속성을 강화하는 모습이다. 해당 설정을 통해서 세션 쿠키를 암호화된 연결을 통해서만 전송하여 중간자 공격을 방지 할 수 있으며, JavaScript에서 세션 쿠키에 접근할 수 없도록 하여 XSS 공격으로부터 세션 쿠키를 보호할 수 있다.

4.2. 파라미터 조작

4.2.1. 취약점 개요

클라이언트가 전송해 온 중요한 파라미터 값을 서버가 신뢰하고 검증 없이 그대로 처리 로직에 사용할 때 발생하는 취약점이다. 해당 취약점은 사용자의 비정상적인 동작을 유도 할 수 있다.

4.2.2. 필드값 조작에 따른 검증 여부

❑ 시큐어 코딩 적용

```
$itemId   = (int)$_POST['item_id'];
$quantity = (int)$_POST['quantity'];

$stmt = $db->prepare("SELECT unit_price FROM products WHERE id = ?");
$stmt->execute([$itemId]);
$unitPrice = (int)$stmt->fetchColumn();

$total = $unitPrice * $quantity;
```

표 23 서버에서 가격 계산 예시

위의 표 23은 클라이언트가 보내온 price는 무시하고 item_id 와 quantity 만 받아서 서버 DB 에 저장된 신뢰된 단가(unit_price)를 조회해 계산하는 예시 코드이다.

.

4.3. XSS 취약점

4.3.1. 취약점 개요

크로스 사이트 스크립팅(XSS: Cross Site Scripting)은 일반적으로 자바스크립트(Java Script), VB 스크립트, 액티브 X(ActiveX), CSS 등을 이용하는 공격 기법이다. 웹 애플리케이션에서 브라우저로 전송하는 페이지에서 공격자가 의도적으로 브라우저에서 실행될 수 있는 악성 스크립트를 웹 서버에 삽입 또는 출력 시 악성 스크립트 코드를 검증하지 않거나, 출력 시 필터링(Filtering)시키지 않을 때 발생한다.

4.3.2. 악의적인 스크립트 필터링 가능 여부 검증

❑ 시큐어 코딩 적용

- $new = htmlspecialchars("Test", ENT_QUOTES);
- echo $new; // Test

표 24 htmlspecialschars 사용 예

PHP의 htmlspecialchars 함수는 특수 문자를 해당 HTML 엔티티로 변환하는데 사용된다. 위의 표 20처럼 공격자가 웹 애플리케이션에 악성 코드를 삽입할 수 있는 XSS 공격과 같은 잠재적인 보안 취약성을 방지하는데 사용할 수 있다.

특수문자	변환된 문자
&(앰퍼샌드)	&amp;
""(겹따옴표)	&quot;
''(홑따옴표)	&#039;
<(미만)	&lt;
>(이상)	&gt;

표 25 특수문자 변환

위 표 25의 변환된 문자는 htmlspecialchars 함수를 이용하여 특수문자를 HTML 엔티티로 변환했을 때 나오는 문자이다.

4.4. 에러 메시지 처리

4.4.1. 취약점 개요

정보 유출 취약점은 애플리케이션이나 서버가 내부 동작에 대한 과도한 세부 정보를 에러 메시지로 그대로 노출할 때 발생한다. 이로 인해 공격자는 애플리케이션 구조, 사용 기술, 버전, 경로, 쿼리 문법 등 중요 단서를 얻어 더 정교한 공격을 준비할 수 있다.

4.4.2. 에러 메시지를 통한 중요/불필요한 정보 유출 검증

❑ 시큐어 코딩 적용

```
$mysqli = new mysqli($host, $user, $pass, $db);
if ($mysqli->connect_errno) {
    error_log("[DB CONNECT ERROR] " . $mysqli->connect_error);
    http_response_code(500);
    echo "죄송합니다. 서버에 문제가 발생했습니다.";
    exit;
}
```

표 26 예외처리를 통한 정보 유출 방지 예

위 표 26는 상세 에러 로직 분리하여 사용자에게는 "죄송합니다. 서버에 문제가 발생했습니다." 라는 일반 메시지만 노출시키는 코드 예이다. 해당 코드를 통해서 DB의 종류와 쿼리 구조 유추를 방지할 수 있다.

❑ 서버 보안 설정

■ Nginx : server_tokens off;

표 27 Nginx 버전 노출 제한 설정

위 표 27은 Nginx의 버전 정보를 숨기기 위해서 nginx.conf 환경설정에서 server_tokens 값을 off로 변경하는 설정이다.

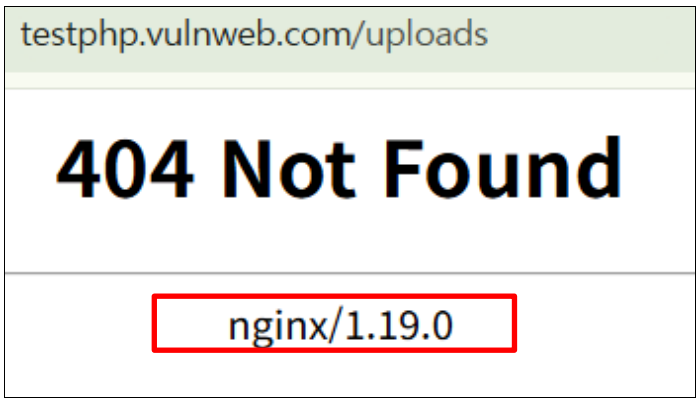

그림 4-1 버전 노출 제한 설정 전

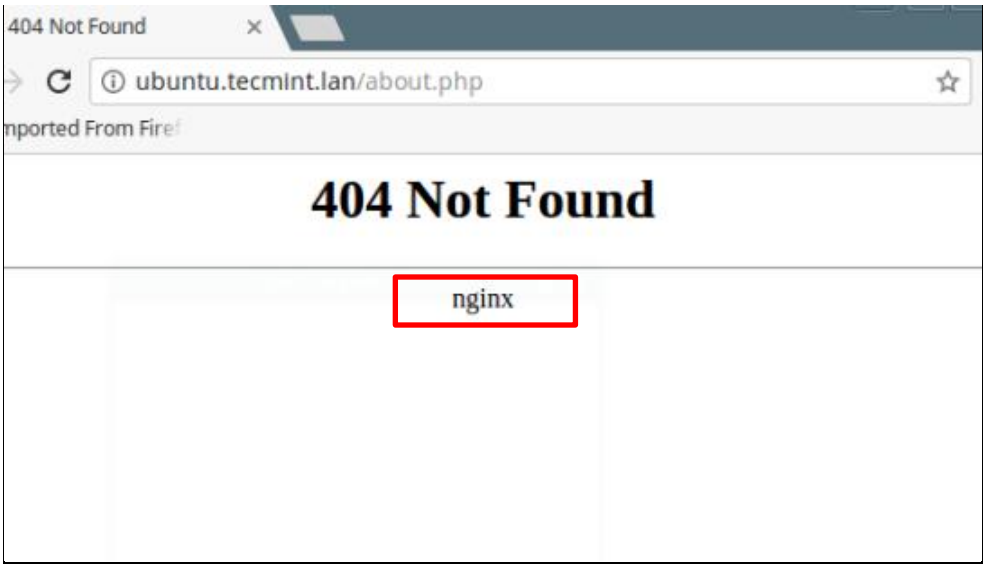

그림 4-2 버전 노출 제한 설정 후

위의 그림 4-1 버전 노출 제한 설정 전그림 4-1 과 그림 4-2은 Nginx 서버 버전 정보를 숨김 설정을 한 모습이다.

4.5. 디렉터리 리스팅 취약점

4.5.1. 취약점 개요

웹 서버에서 디렉터리 인덱싱(Directory Listing)을 비활성화하지 않은 경우, 외부 사용자가 해당 디렉터리에 접근할 때 내부 파일 목록이 그대로 노출될 수 있다.
이러한 취약점을 방지하기 위해 웹 서버 설정을 통해 디렉터리 목록 표시 기능을 비활성화하고, 민감 파일(.bak, .sql, .log 등)은 웹 루트 외부로 이동하거나 접근 차단 정책을 별도로 설정해야 한다. 불필요한 디렉터리는 서버 운영 시점에 제거하고, 정상 서비스 외 디렉터리에 대한 접근이 모

두 제한되도록 서버 보안 정책을 설정해야 한다.

4.5.2. 권고사항 점검 방법

❑ 점검 방법

웹 브라우저 또는 디렉토리 스캐닝 도구(dirb, gobuster 등)를 이용해 서버 내 주요 디렉터리에 접근하여 아래 항목이 확인되는지 점검한다.

1. 디렉터리 접근 시 Index of /xxx/ 형식으로 파일 목록이 출력되는 경우
2. . sql, .log, .bak, .txt, .php 등 민감 확장자의 파일이 열람 가능한 경우
3. 설정 파일 및 백업 파일이 외부에 노출되는 경우

❑ 서버 보안 설정

디렉터리 리스팅을 방지하기 위해 아래 설정을 적용한다.

항목	설정 내용
Apache	.htaccess 파일에 Options -Indexes 설정 추가
Nginx	Location 블록 내 autoindex off; 설정
기타	백업 파일은 웹 루트 외부로 이동(.bak, .sql, .log 제거)

표 28 디렉터리 리스팅 차단 설정 내용

추가 조치

1. 불필요한 디렉터리는 삭제하거나 접근을 403 Forbidden으로 제한한다.
2. 민감 파일은 업로드 방지 및 .htaccess로 접근 차단한다. .htaccess 파일을 통해 디렉터리 접근을 제한하고, 디렉터리 목록을 노출시키지 않도록 설정할 수 있다. 또, 웹 서버의 루트 디렉터리 또는 특정 디렉터리에 위치시킬 수 있다.

OS별 점검 파일 위치 및 점검 방법	
Debian, Ubuntu 계열	Indexes 옵션 사용 여부 확인 **#vi /etc/apache2/apache2.conf** Options Indexes FollowSymLinks
Solaris, Linux, AIX, HP-UX	Indexes 옵션 사용 여부 확인 **#vi /[Apache_home]/conf/httpd.conf** Options Indexes FollowSymLinks
HREL, CentOS 계열	Indexes 옵션 사용 여부 확인 **#vi /etc/httpd/conf/httpd.conf** Options Indexes FollowSymLinks
위에 제시한 파일에 "Indexes" 옵션이 설정된 경우 아래의 보안설정방법에 따라 옵션 설정 변경	

표 29 점검 파일 위치 및 점검 방법

Debian, Ubuntu 계열

1. vi 편집기를 이용하여 /etc/apache2/apache2.conf 파일 열기
   #vi /etc/apache2/apache2.conf
2. 설정된 모든 디렉터리의 Options 지시자에서 Indexes 옵션 제거

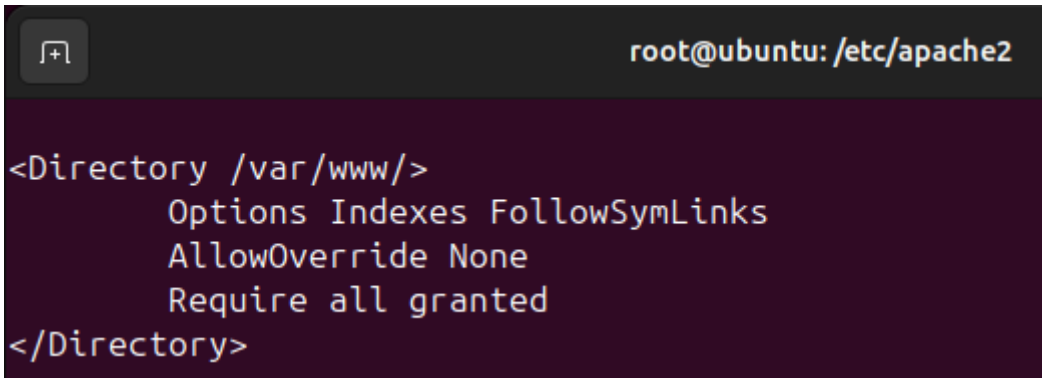

그림 4-3 vi /etc/apache2/apache2.conf

(수정 전) Option 지시자에 Indexes 옵션이 설정되어 있음

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

표 30 Options 지시자 Indexes 옵션 설정 전

(수정 후) Options 지시자에 Indexes 삭제 또는 -Indexes 변경 후 저장

```
<Directory /var/www>
    Options Indexes 삭제 (또는 -Indexes +FollowSymLinks)
    AllowOverride All
    Require all granted
</Directory>
```

표 31 Options 지시자 Indexes 옵션 설정 후

1. Options -Indexes : 디렉터리 리스트 출력 금지
2. +FollowSymLinks : 심볼릭 링크 허용
3. AllowOverride All : .htaccess를 통한 세부 제어 허용
4. Require all granted : 모든 사용자 접근 허용(웹 페이지가 위치한 경로만 허용)

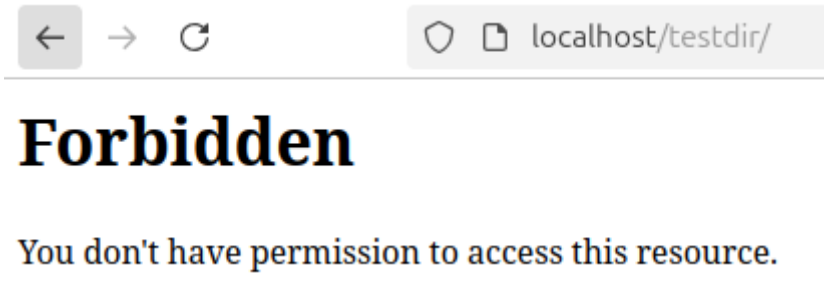

그림 4-4 Indexes 옵션 적용 후 디렉터리 리스팅 차단 화면

위의 그림 4-1 버전 노출 제한 설정 전그림 4-1 과 그림 4-2은 Nginx 서버 버전 정보를 숨김 설정을 한 모습이다.

Solaris, Linux, AIX, HP-UX

1. vi 편집기를 이용하여 /[Apache_home]/conf/httd.conf 파일 열기

#vi /[Apache_home]/conf/httd.conf

2. 설정된 모든 디렉터리의 Options 지시자에서 Indexes 옵션 제거

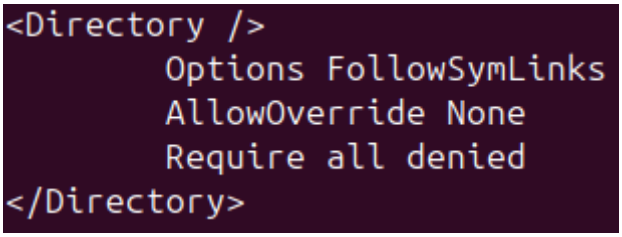

그림 4-5 vi /[Apache_home]/conf/httd.conf

(수정 전) Option 지시자에 Indexes 옵션이 설정되어 있음

```
<Directory /
    Options FollowSymLinks
    AllowOverride None
    Require all denied
</Directory>
```

표 32 Options 지시자 Indexes 옵션 설정 전

(수정 후) Options 지시자에 Indexes 삭제 또는 -Indexes 변경 후 저장

```
<Directory / >
    Options Indexes 삭제 (또는 -Indexes +FollowSymLinks)
    AllowOverride All
    Require all granted
</Directory>
```

표 33 Options 지시자 Indexes 옵션 설정 후

4.6. 불필요 파일 존재

4.6.1. 취약점 개요

웹 루트 하위에 내부 문서 또는 백업 파일, 로그 파일, 압축 파일 등과 같이 불필요한 파일들이 존재할 경우 파일명을 유추하여 파일명을 알아내고 직접 요청하여 여러 정보가 노출 될 수 있는 취약점이다. 해당 취약점이 존재할 경우 백업 및 임시 파일을 공격자가 획득하고 분석해 웹 애플리케이션 내부 로직 또는 DB정보, 액세스 정보 등을 획득하여 추가적인 피해를 발생시킬 수 있는 취약점이다.

4.6.2. 백업 파일 존재

❑ 서버 보안 설정

```
location ~* ₩.(bak|old|swp|sql|zip|env)$ {
deny all;
}
```

표 34 Nginx 확장자 기반 접근 차단

위 표 27은 nginx.conf에서 특정 확장자를 가진 파일에 대한 접근을 불가능 하게 하는 보안 설정이다.

4.6.3. 데모 페이지 존재

❑ 서버 보안 설정

기능 구현과 직접적인 연관이 없는 페이지나 테스트용 엔드포인트가 운영 서버에 그대로 남아 있는 것은 불필요한 취약점을 늘리는 일이기 때문에 완전히 삭제하거나, 삭제가 어렵거나 압축 해제된 형태로 유지해야 할 경우에는 외부 접근을 철저히 제한해야 합니다.

```
location /ajax-demo {
deny all;
}
```

표 35 Nginx 데모 페이지 접근 차단

위 표 35은 nginx.conf에서 특정 페이지에 대한 접근을 불가능 하게 하는 보안 설정이다.

4.7. 부적절한 Include 취약점

4.7.1. 취약점 개요

부적절한 파일 Include 취약점이란, 애플리케이션 코드 내에서 외부 입력값(파라미터, 쿠키, 헤더 등)을 검증 없이 include, require, readfile 같은 함수에 사용함으로써, 공격자가 임의의 파일을 서버에 불러오거나 실행하도록 만드는 보안 결함이다. LFI와 RFI로 분류할 수 있으며, 발생 원인으로는 입력값 검증 부재, PHP 설정 오류 등의 원인이 있다.

4.7.2. 부적절한 Include 허용 여부

❑ 시큐어 코딩 적용

```
$allowed = ['default.jpg','profile.png','banner.jpg'];
if (!in_array($fileName, $allowed)) {
   http_response_code(400);
   exit('Invalid file');
}
include __DIR__ . '/images/' . $fileName;
```

표 36 화이트리스트 기반 파일 로드 예

위 표 36는 화이트리스트 기반 파일 로드 방식으로 허용된 파일명·경로만 include하도록한 시큐어 코딩이다.

❑ 서버 보안 설정

```
■ allow_url_fopen    = On
■ allow_url_include   = On
```

표 37 php.ini 보안 설정 전

```
■ allow_url_fopen    = Off
■ allow_url_include   = Off
```

표 38 php.ini 보안 설정 후

위 표 37은 원격지 파일을 열지 못하도록 php.ini 환경설정 수정하는 보안 설정이다.

4.8. SQL Injection

4.8.1. 취약점 개요

입력 폼 및 URL 입력란에 SQL문을 삽입하여 DB로부터 정보를 열람하거나 조작할 수 있는 취약점으로, 의도하지 않은 쿼리가 삽입되는 점을 방지하기 위해 사용자로부터 입력된 값에 대해서 특수문자 및 쿼리 예약어를 필터링하는 로직을 구현해야 한다.

4.8.2. SQL Injection 허용 여부

❑ 시큐어 코딩 적용

Prepared statements 사용 예제

```
$pdo = new PDO("mysql:host=localhost;dbname=test", "user", "pass"); $stmt = $pdo->prepare("SELECT * FROM users WHERE username = :username AND password = :password"); $stmt->execute(['username' => $username, 'password' => $password]); $result = $stmt->fetch();
```

표 39 PDO 예제 구문(php)

```
$mysqli = new mysqli("localhost", "user", "pass", "test"); $stmt = $mysqli->prepare("SELECT * FROM users WHERE username = ? AND password = ?"); $stmt->bind_param("ss", $username, $password); $stmt->execute(); $result = $stmt->get_result();
```

표 40 MySQLi 예제 구문(php)

표 39와 표 40의 구문인 Prepared Statement는 SQL 템플릿과 사용자 입력값을 분리하여 처리하기 때문에, SQL Injection 공격을 원천적으로 차단할 수 있는 효과적인 방식이다.

데이터베이스 최소 권한 설정 예제
데이터베이스 사용자에게 불필요한 권한을 부여하지 않도록 최소 권한 원칙을 준수해야 한다. 표 41은 사용자 계정 web_user에게 특정 권한만을 부여하는 예제이다.

```
GRANT SELECT, INSERT, UPDATE ON test_db.* TO 'web_user'@'localhost' IDENTIFIED BY 'password';
```

표 41 MySQL에서 사용자 권한 설정 예제 구문(sql)

이와 같이, 애플리케이션이 사용하는 데이터베이스 계정은 SELECT, INSERT, UPDATE 등 필요한 권

한만을 제한적으로 부여하고, DROP, DELETE, GRANT, ALTER 등 위험한 권한은 제거함으로써 보안을 강화할 수 있다.