



OWASP Juice Shop

김범준

24
11.28

OWASP JUICE-SHOP

PENETRATION TESTING REPORT

2024

Table of Contents

개요	01
작성자	02
점검 일정	03
점검 대상	04
점검 방법론	05
취약점 요약	06
• 취약점 목록	
취약점 상세 분석	07
• 취약점 1 : Login Admin (Injection)	
• 취약점 2 : Password Strength(Broken Authentication)	
• 취약점 3 : Upload Size(mproper Input Validation)	
• 취약점 4 : CAPTCHA Bypass(Broken Anti Automation)	
• 취약점 5 : Change Bender's Password(Broken Authentication)	
대응 방안	08
• 단기 조치 사항	
• 장기 조치 사항	
• 기술적 대응 방안	
총평 및 결론	09
• 개선 우선순위	
• 결론	

개요

OWASP Juice Shop은 OWASP(Open Web Application Security Project)에서 제공하는 대표적인 웹 애플리케이션 취약점 학습 플랫폼으로, 현대적인 JavaScript 기반 애플리케이션의 취약점을 학습 및 테스트하기 위해 설계되었습니다.

본 보고서는 OWASP Juice Shop에 대한 모의해킹(Penetration Testing) 결과를 체계적으로 정리하여, 애플리케이션 내 존재하는 보안 취약점과 개선 방안을 제시함으로써 웹 애플리케이션의 보안성을 향상시키기 위한 기초 자료를 제공합니다.

OWASP Juice Shop is a flagship web application vulnerability training platform provided by the OWASP (Open Web Application Security Project). It is designed to facilitate the study and testing of vulnerabilities in modern JavaScript-based applications.

This report systematically documents the results of penetration testing conducted on OWASP Juice Shop. By identifying security vulnerabilities within the application and proposing remediation measures, it aims to serve as a foundational resource for improving the security posture of web applications.

작성자

Author

01

- 이름 : 김범준
- 소속 : DevSecOps LEDTEAM
- 연락처 : jcbou123@gmail.com

점검 일정

Inspection Schedule

02

- 시작일 : 2024-11-18
- 종료일 : 2024-11-28
- 총 소요 일수: 11일

점검 대상

Inspection Target

03

- 웹 애플리케이션: OWASP Juice Shop
- URL : http://192.168.0.18:3000
- URL : http://192.168.56.102:3000
- 점검 범위 : 웹 애플리케이션
- 테스트 환경 : Kali Linux 2023.3

점검 방법론

- 내용

- 본 점검은 OWASP의 모의해킹 가이드라인을 참고하여 수행되었으며, 자동화 도구와 수동 분석을 병행하여 진행되었습니다.
- 주요 도구로는 Burp Suite, OWASP ZAP, Kali Linux 등이 사용되었습니다.

- 참고 기준 및 문헌

- OWASP Testing Guide v4.2
 - <https://owasp.org/www-project-web-security-testing-guide/v42/>
- OWASP Top 10 2021
- CVSS v3.1

1

INFORMATION
GATHERING

정보 수집 (INFORMATION GATHERING)

시스템 및 네트워크에 대한 정보를 수집하여 공격 표면을 파악합니다.

2

VULNERABILITY
ANALYSIS

취약점 분석 (VULNERABILITY ANALYSIS)

수집된 정보를 바탕으로 시스템의 취약점을 분석합니다.

3

EXPLOITATION

취약점 공격 (EXPLOITATION)

발견된 취약점을 실제로 공격하여 시스템의 보안 상태를 평가합니다.

4

REPORTING

보고서 작성 (REPORTING)

점검 결과를 정리하여 보고서를 작성하고, 발견된 취약점과 대응 방안을 제시합니다.

취약점 요약

심각도	취약점 수	비율	CVSS 범위
높음	3	60%	7.0 ~ 9.9
중간	2	40%	4.0 ~ 6.9
낮음	0	0%	0.1 ~ 3.9

취약점 목록

ID	취약점명	OWASP 카테고리	심각도	CVSS 점수	우선순위
V1	Login Admin	Injection	높음	9.8	긴급
V2	Password Strength	Broken Authentication	중간	6.5	높음
V3	Upload Size	Improper Input Validation	중간	6.1	높음
V4	CAPTCHA Bypass	Broken Anti Automation	높음	8.2	긴급
V5	Change Bender's Password	Broken Authentication	높음	8.8	긴급

취약점 상세 분석

취약점 1 : Login Admin (Injection)

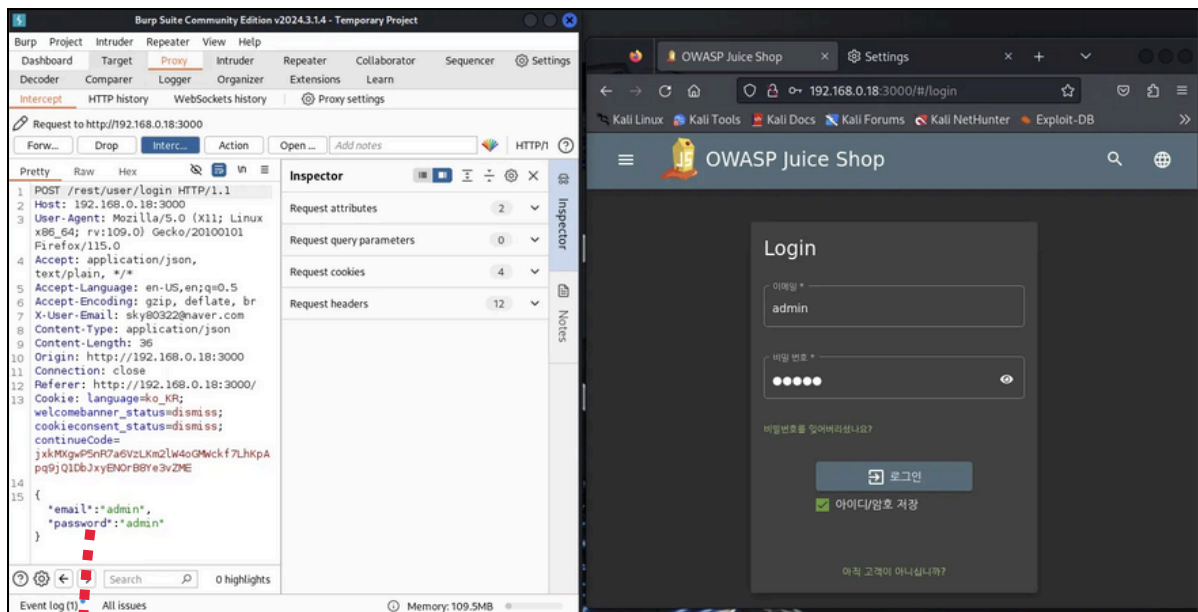
- 심각도
 - 높음 (CVSS: 9.8)
- 취약점 설명
 - 로그인 폼에서 **SQL 인젝션 취약점**을 이용하여 관리자 계정에 비인가 접근이 가능한 취약점입니다. 사용자 입력이 적절히 검증되지 않고 SQL 쿼리에 직접 삽입되어 발생합니다.
- 공격 방법
 - 로그인 페이지에 **악의적인 SQL 구문**을 삽입하여 인증 과정을 우회합니다.
- 사용 도구
 - Burp Suite, Proxy
- 발견 경로
 - **juice-shop/src/controllers/loginController.js** 파일의 **login** 메서드에서 발견되었습니다.
- 영향
 - 전체 시스템 접근 권한 탈취 가능

취약점 1 : Login Admin (Injection)

- 공략 과정

- 취약점 식별: Burp Suite를 사용하여 로그인 요청을 캡처하고 분석

- 메일 ID : admin / 비밀번호 : admin 입력



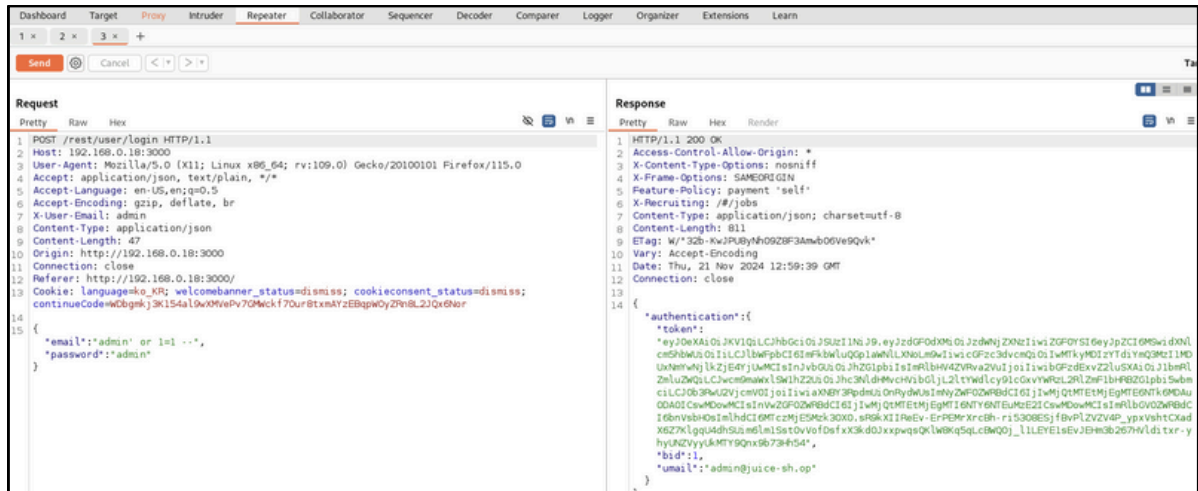
- 인젝션 페이로드 준비: SQL 인젝션 페이로드를 준비



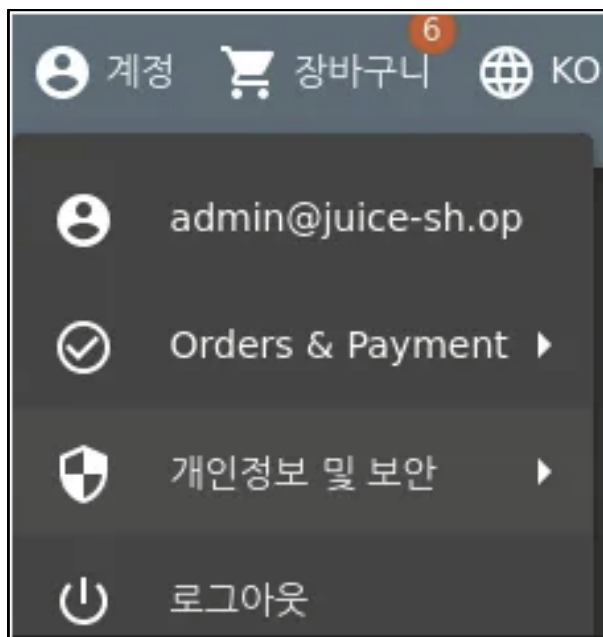
```
{  
  "email": "admin' or 1=1 --"  
  "password": "admin"  
}
```


취약점 1 : Login Admin (Injection)

3. 공격 실행: 준비한 페이로드를 Burp Suite의 Repeater 기능을 사용하여 서버에 전송



4. 결과 확인 및 증거 수집: 공격 성공을 확인하고 관리자 대시보드에 접근가능



취약점 2 : Password Strength (Broken Authentication)

- 심각도
 - 중간 (CVSS: 6.5)
- 취약점 설명
 - 취약한 비밀번호 정책으로 인해 계정 탈취가 가능한 취약점입니다. **비밀번호 복잡성 요구사항이 부족**하여 쉽게 추측 가능한 비밀번호가 허용됩니다.
- 공격 방법
 - 다양한 비밀번호 패턴을 시도하여 계정에 접근합니다.
- 사용 도구
 - Burp Suite, OWASP ZAP
- 발견 경로
 - **juice-shop/src/controllers/userController.js** 파일의 **signup** 메서드에서 발견되었습니다.
- 영향
 - 사용자 계정 무단 접근 가능

취약점 2 : Password Strength (Broken Authentication)

• 공략 과정

1. 취약점 식별: 회원가입 과정에서 비밀번호 정책 분석

- 메일 ID : admin@juice-sh.op 이메일 존재를 확인
- 비밀번호를 1234 입력해도 비밀번호 복잡성 요구사항이 적어 인증이 완료

The screenshot shows the '사용자 등록' (User Registration) form. The email field is filled with 'admin@juice-sh.op'. The password field shows '*****' and a hint '비밀번호는 반드시 5-40 자 이상이어야 합니다. 8/20'. The password confirmation field also shows '*****' with a hint '비밀번호 재확인 8/40'. There is a 'Show password advice' toggle. The security question is '가장 큰 형제의 중간이름은?' (Middle name of the oldest brother) with an answer of '1234'. A '등록하기' (Register) button is at the bottom. A green message at the bottom says 'Already a customer?'.

2. 취약한 비밀번호 테스트: 다양한 비밀번호 패턴 시도(Brute Force)

The screenshot shows the Burp Suite interface. The 'Attack type' is set to 'Sniper'. The 'Payload positions' section shows a target URL 'http://192.168.56.102:3000'. The 'Payload' field contains a JSON body for a login request, with the password field highlighted in red and containing the value '1234567890'. The 'Attack' button is visible.

취약점 2 : Password Strength (Broken Authentication)

ResultsPositionsPayloadsResource poolSettings

Intruder attack results filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
109	academia	401	20			442	
110	access	401	25			442	
111	access14	401	37			442	
112	account	401	13			442	
113	action	401	19			442	
114	admin	401	9			442	
115	admin1	401	12			442	
116	admin12	401	37			442	
117	admin123	200	20			1197	
118	adminadmin	401	17			442	

RequestResponse


PrettyRawHex

```
1 POST /rest/user/login HTTP/1.1
2 Host: 192.168.56.102:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json
8 Content-Length: 51
9 Origin: http://192.168.56.102:3000
10 Connection: keep-alive
11 Referer: http://192.168.56.102:3000/
12 Cookie: language=ko_KR; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=
5hzv2M9b8POLNB63RoD1g0pwAVZfaMuYqI4VAHxq5vkerayX4EK7J2jlmYN
13
14 {
15   "email": "admin@juice-sh.op",
16   "password": "admin123"
17 }
```

3. admin 사용자 계정 무단 접근 허용

- admin@juice-sh.op / admin123 무단 접속 완료

User Profile



Email:

admin@juice-sh.op

Username:

e.g. SuperUser

Set Username

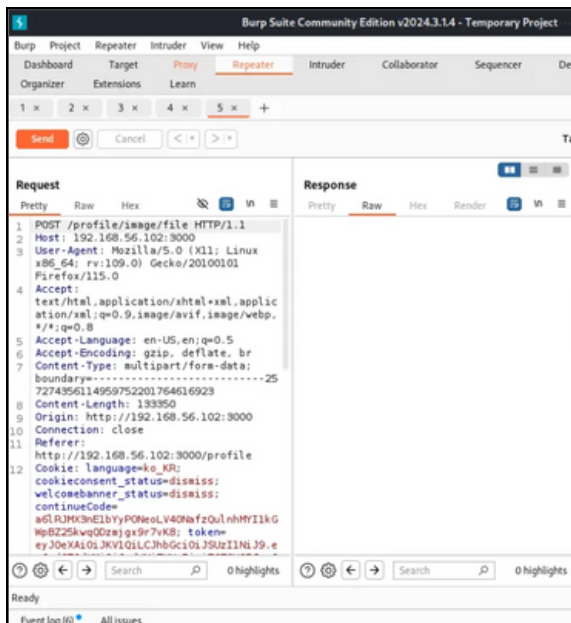
취약점 3 : Upload Size (Improper Input Validation)

- 심각도
 - 중간 (CVSS: 6.1)
- 취약점 설명
 - 파일 업로드 기능에서 파일 크기 제한이 제대로 적용되지 않아, 공격자가 대용량 파일을 업로드할 수 있는 취약점입니다. 이로 인해 서버의 리소스가 고갈되거나 서비스 거부(DoS) 공격이 발생할 수 있습니다.
- 공격 방법
 - 공격자는 파일 업로드 요청을 변조하여 서버가 허용하는 최대 파일 크기를 초과하는 파일을 업로드할 수 있습니다. 이로 인해 서버의 저장 공간이 소진되거나, 서버가 과부하 상태에 빠질 수 있습니다.
- 사용 도구
 - Burp Suite
- 발견 경로
 - `juice-shop/src/controllers/uploadController.js` 파일의 `upload` 메서드에서 발견되었습니다.
- 영향
 - 서버 리소스 고갈 및 서비스 거부(DoS) 가능성

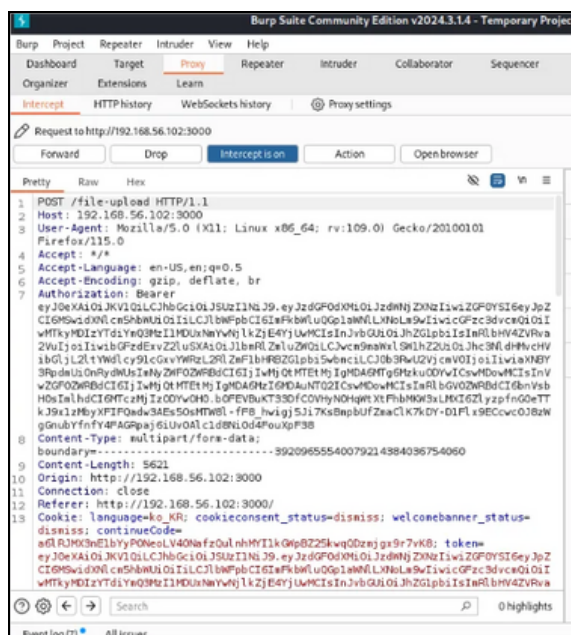
취약점 3 : Upload Size (Improper Input Validation)

2. 크기 제한 우회: 요청 변조를 통한 제한 우회

- 프로필 사진 업로드 - largefile.pdf - Burp Suite 실행 - Repeater

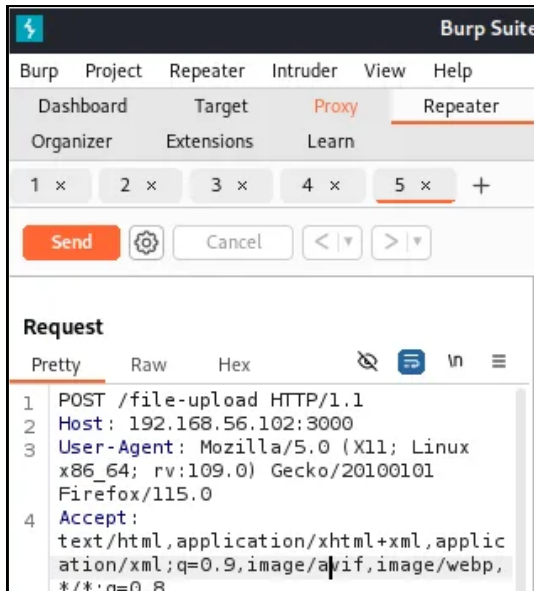


- Complaint 메뉴 - largefile.pdf 업로드 - Burp Suite - Proxy

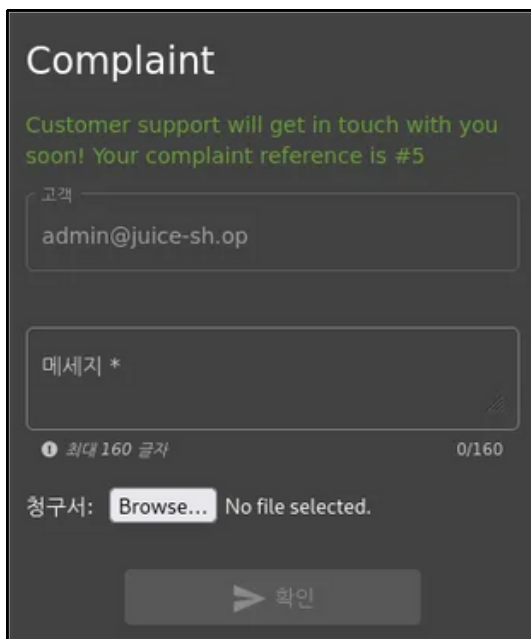


취약점 3 : Upload Size (Improper Input Validation)

- 헤더의 POST 섹션에서 " /file-upload " 복사하여 Repeater로 이동 후 POST 섹션 수정



3. 대용량 파일 업로드



취약점 4 : CAPTCHA Bypass(Broken Anti Automation)

- 심각도
 - 높음 (CVSS: 8.2)
- 취약점 설명
 - CAPTCHA(자동화 방지 시스템)가 제대로 작동하지 않거나 우회 가능한 경우 발생합니다. 이는 봇이나 자동화된 스크립트가 CAPTCHA를 통과하여 사이트를 악용할 수 있게 만듭니다.
- 공격 방법
 - 네트워크 요청을 분석하여 CAPTCHA 검증을 우회합니다.
- 사용 도구
 - Burp Suite, OWASP ZAP
- 발견 경로
 - `juice-shop/src/controllers/contactController.js` 파일의 `contact` 메서드에서 발견되었습니다.
- 영향
 - 서버 리소스 고갈 및 서비스 거부(DoS) 가능성

취약점 4 : CAPTCHA Bypass(Broken Anti Automation)

- 공략 과정

1. 취약점 식별: CAPTCHA 구현 방식 분석

- 잘못된 CAPTCHA 입력 : 6

문의하기

작성지
***in@juice-sh.op

댓글 *
beomjoon

최대 160 글자 8/160

평점

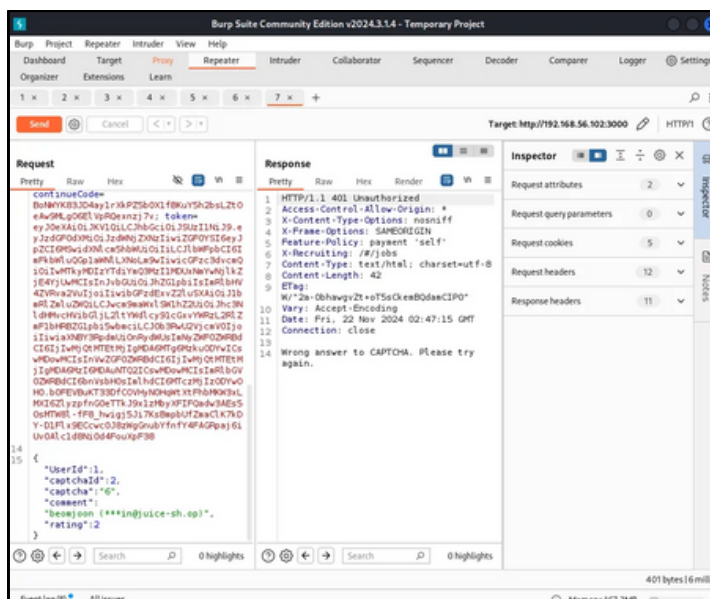
CAPTCHA: What is 9-4*1 ?

결과 *
6

확인

2. 검증 로직 분석: 네트워크 요청 분석

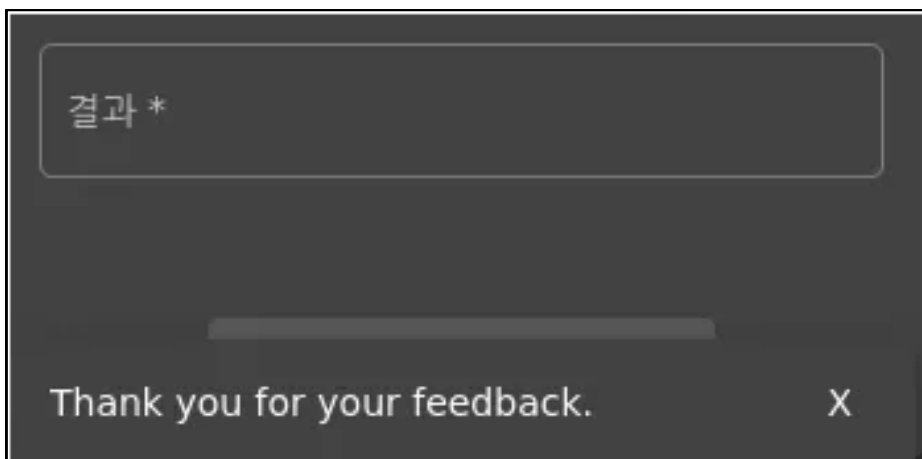
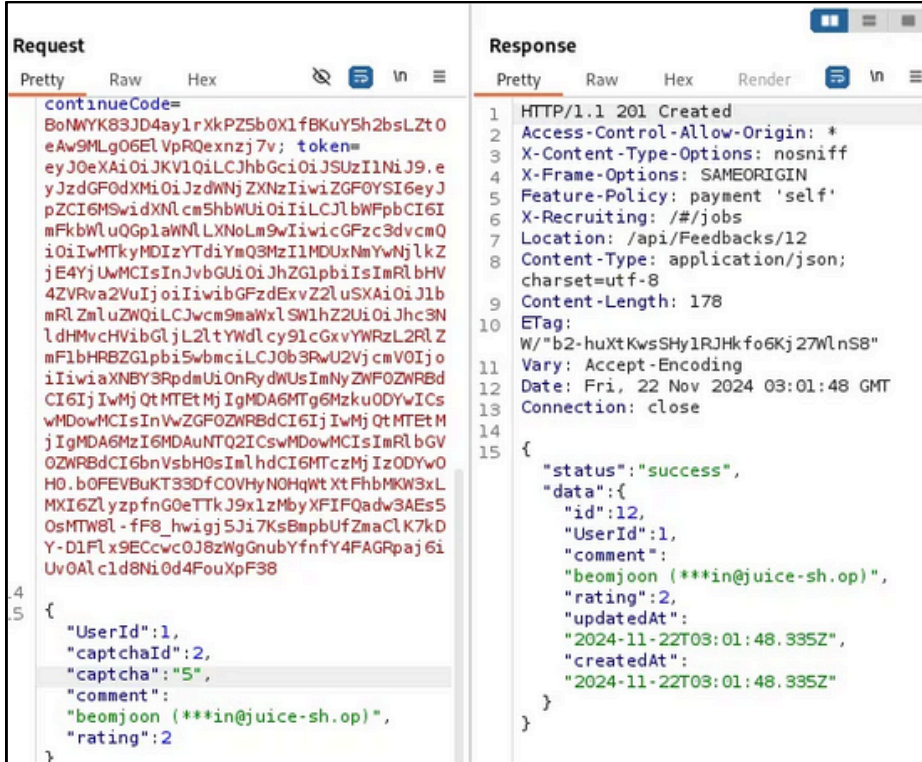
- Burp Suite - Header 분석 - Repeater로 보낸 후 Send - 잘못된 답변이라는 메시지



취약점 4 : CAPTCHA Bypass(Broken Anti Automation)

3. 무회 성공: CAPTCHA 검증 없이 요청 전송

- captcha 의 값을 5로 수정 하면 검증 없이 성공 메시지가 뜸.



취약점 5 : Change Bender's Password (Broken Authentication)

- 심각도
 - 높음 (CVSS: 8.8)
- 취약점 설명
 - 사용자 비밀번호 변경 프로세스에 취약점이 있어 타 사용자의 계정을 탈취할 수 있는 취약점입니다. 인증 절차가 부족하여 비밀번호 변경이 가능합니다.
- 공격 방법
 - 사용자 식별자를 변조하여 타 사용자의 비밀번호를 변경합니다.
 - SQL 주입이나 비밀번호 분실을 사용하지 않고 Bender의 비밀번호를 slurmCl4ssic으로 변경
- 사용 도구
 - Burp Suite, SQLMap
- 발견 경로
 - juice-shop/src/controllers/profileController.js 파일의 `changePassword` 메서드에서 발견되었습니다.
- 영향
 - 타 사용자의 계정 탈취 가능

취약점 5 : Change Bender's Password (Broken Authentication)

- 공략 과정

- 취약점 식별: SQL INJECTION / 비밀번호 변경 기능 분석

- BENDER 계정 존재 확인



사용자 등록

Email must be unique

이메일 *
bender@juice-sh.op

비밀번호 *
●●●●●●●●
비밀번호는 반드시 5-40 자 이상이어야 합니다. 8/20

비밀번호 재확인 *
●●●●●●●●
8/40

☐ Show password advice

보안 질문 *
Mother's maiden name?

나중에 변경할 수 있습니다.

답변 *
12345678

+ 등록하기



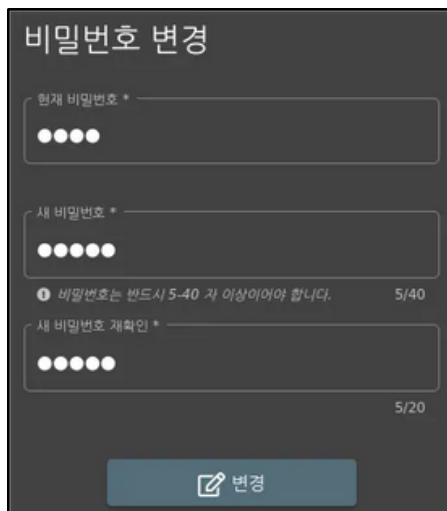
Login

이메일 *
bender@juice-sh.op'--

비밀번호 *
●●●●●●
비밀번호를 잊어버리셨나요?

로그인

☐ 아이디/암호 저장



비밀번호 변경

현재 비밀번호 *
●●●●

새 비밀번호 *
●●●●●●
비밀번호는 반드시 5-40 자 이상이어야 합니다. 5/40

새 비밀번호 재확인 *
●●●●●●
5/20

변경

- 현재 비밀번호, 새 비밀번호, 비밀번호 확인 필드가 있는지 확인합니다. 각 필드에 대한 유효성 검사를 분석합니다.
- asdfa / asdfa 새 비밀번호
- 입력 한 후 버프로 잡아봅니다.

대응 방안

단기 조치 사항

취약점	우선 순위	조치 사항
Login Admin	Injection	프리페어드 스테이트먼트 적용 및 입력 검증 강화
CAPTCHA Bypass	Broken Authentication	reCAPTCHA v3 도입 및 CAPTCHA 로직 강화
Change Bender's Password	Improper Input Validation	세션 기반 인증 강화 및 CSRF 토큰 적용
Password Strength	Broken Anti Automation	비밀번호 정책 강화 및 강력한 비밀번호 요구
Upload Size	Broken Authentication	서버 측 파일 크기 및 형식 검증 구현

장기 조치 사항

1. 보안 아키텍처 개선

- 인증/인가 시스템 재설계: OAuth 2.0 또는 OpenID Connect를 사용하여 인증 시스템을 재설계합니다.
 - 현대적이고 안전한 로그인 방식 채택
- API 보안 게이트웨이 도입: API 요청을 모니터링하고 비정상적인 요청을 차단합니다.
 - 보안 문지기 역할, 모든 외부 요청을 먼저 검사
- 로깅 및 모니터링 강화: 모든 보안 이벤트를 로깅하고 실시간 모니터링 시스템을 구축합니다.
 - 24시간 감시 시스템에 해당

2. 개발 프로세스 개선

- 시큐어 코딩 가이드라인 수립: 개발자에게 보안 모범 사례를 교육합니다.
- 자동화된 보안 테스트 도입: CI/CD 파이프라인에 보안 테스트를 통합합니다.
- 정기적인 보안 교육 실시: 최신 보안 위협에 대한 인식을 높입니다.

3. 보안 통제 강화

- WAF 도입: SQL 인젝션, XSS 등 웹 공격을 차단합니다.
- SIEM 시스템 구축: 보안 로그를 중앙에서 관리하고 실시간으로 위협을 탐지합니다.
- 취약점 스캐닝 자동화: 정기적인 취약점 스캐닝을 자동화하여 신속히 대응합니다.

기술적 대응 방안

1. Login Admin (Injection)

- 프리페어드 스테이트먼트 사용: SQL 쿼리를 작성할 때 사용자 입력을 직접 삽입하지 않고, 프리페어드 스테이트먼트를 사용하여 SQL 인젝션을 방지합니다.

```
// SQL Injection 대응
function secureLogin(req, res) {
  const { username, password } = req.body;
  const query = 'SELECT * FROM users WHERE username = ? AND password = ?';
  db.execute(query, [username, password], (err, results) => {
    if (err) {
      return res.status(500).send('Database error');
    }
    if (results.length > 0) {
      req.session.user = results[0];
      res.redirect('/admin');
    } else {
      res.status(401).send('Invalid credentials');
    }
  });
}
```

- 입력 검증 강화: 모든 사용자 입력에 대해 화이트리스트 기반의 검증을 수행하여 악의적인 입력을 차단합니다.

```
function validateInput(input) {
  const regex = /^[a-zA-Z0-9_]+$/; // 허용된 문자만 입력
  return regex.test(input);
}
```

기술적 대응 방안

2. CAPTCHA Bypass

- reCAPTCHA v3 도입: Google reCAPTCHA v3를 도입하여 사용자의 행동을 분석하고, 자동화된 요청을 차단합니다.

```
// CAPTCHA 구현
app.post('/verify', async (req, res) => {
  const recaptchaResponse = await verifyRecaptcha(req.body.token);
  if (!recaptchaResponse.success) {
    return res.status(400).json({ error: 'Invalid CAPTCHA' });
  }
});
```

- CAPTCHA 로직 강화: CAPTCHA 검증 로직을 서버 측에서 처리하여 클라이언트 측에서의 우회를 방지합니다.

3. Change Bender's Password

- 세션 기반 인증 강화: 세션 관리 강화를 통해 세션 하이재킹을 방지합니다. 세션 타임아웃 및 재인증 절차를 추가합니다.

```
// 세션 타임아웃 설정
app.use(session({
  secret: 'your-secret-key',
  resave: false,
  saveUninitialized: true,
  cookie: { maxAge: 60000 } // 1분
}));
```

기술적 대응 방안

- CSRF 토큰 적용: CSRF 공격을 방지하기 위해 모든 상태 변경 요청에 대해 CSRF 토큰을 검증합니다.

```
// CSRF 토큰 생성 및 검증
const csrf = require('csrf');
const csrfProtection = csrf({ cookie: true });

app.get('/form', csrfProtection, (req, res) => {
  res.render('send', { csrfToken: req.csrfToken() });
});

app.post('/process', csrfProtection, (req, res) => {
  res.send('data is being processed');
});
```

4. Password Strength

- 비밀번호 정책 강화: 최소 8자 이상의 길이, 대문자, 소문자, 숫자, 특수 문자를 포함하도록 비밀번호 정책을 강화합니다.

```
function isStrongPassword(password) {
  const regex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$/;
  return regex.test(password);
}
```

기술적 대응 방안

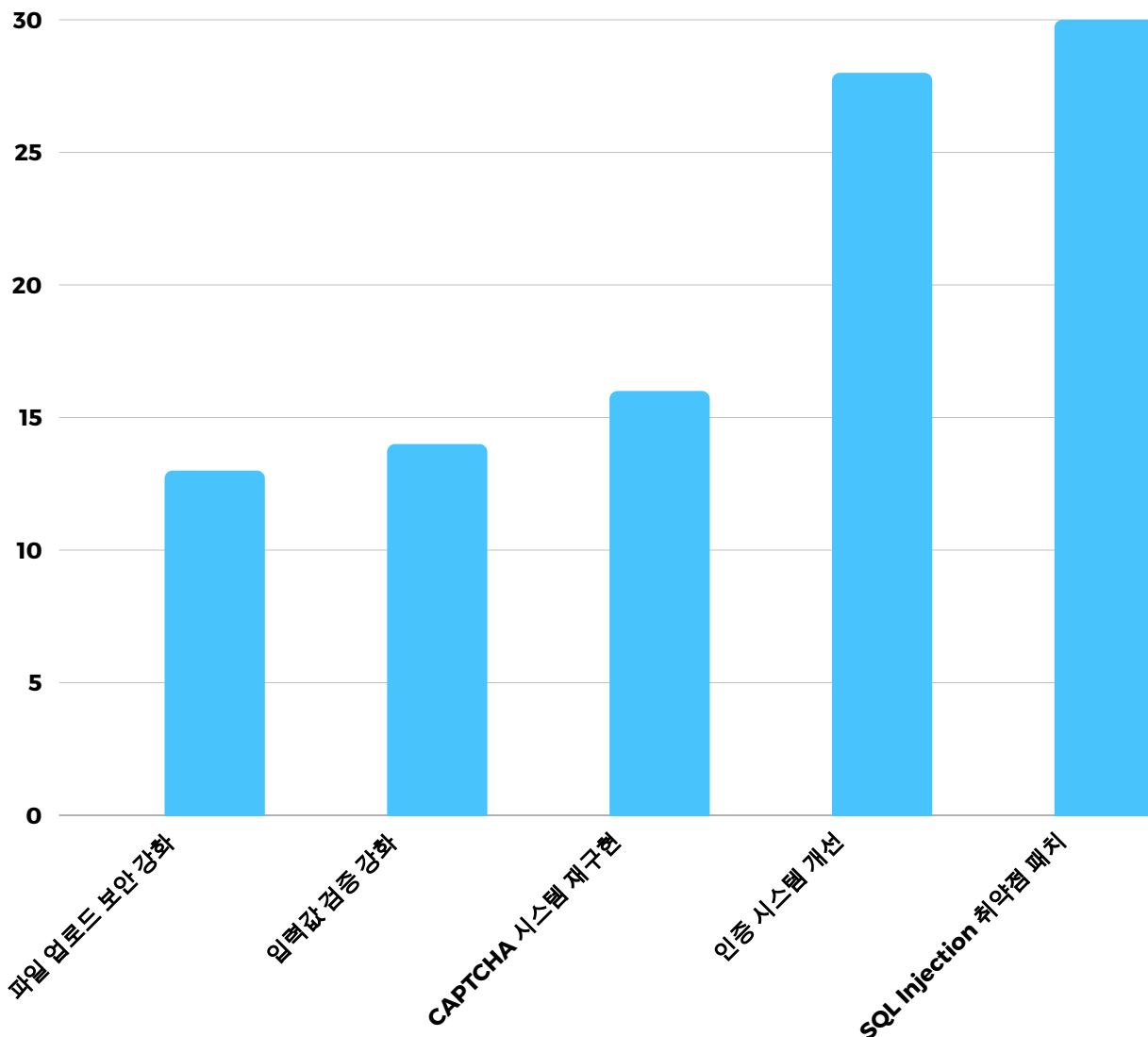
5. Upload Size

- 서버 측 파일 검증: 업로드된 파일의 크기와 형식을 서버 측에서 검증하여 허용되지 않는 파일이 업로드되지 않도록 합니다.

```
const multer = require('multer');
const upload = multer({
  limits: {
    fileSize: 5 * 1024 * 1024, // 5MB 제한
  },
  fileFilter: (req, file, cb) => {
    if (file.mimetype === 'image/png' || file.mimetype ===
'image/jpeg') {
      cb(null, true);
    } else {
      cb(new Error('Invalid file type'), false);
    }
  }
});
```

총평 및 결론

개선 우선순위



- SQL Injection 취약점 패치 (긴급)
- 인증 시스템 개선 (긴급)
- CAPTCHA 시스템 재구현 (긴급)
- 입력값 검증 강화 (높음)
- 파일 업로드 보안 강화 (높음)

결론

본 모의해킹을 통해 발견된 5개의 주요 취약점 중 3개가 높은 심각도를 가지고 있어 즉각적인 조치가 필요합니다. 특히 인증 관련 취약점들은 시스템 전반의 보안에 직접적인 영향을 미치므로 우선적으로 해결해야 합니다.

시스템의 전반적인 보안 수준을 향상시키기 위해서는 제시된 단기 및 장기 조치 사항을 체계적으로 이행하고, 지속적인 보안 모니터링과 정기적인 보안 점검이 필요합니다.

향후 권고사항

1. 정기적인 보안 감사 실시
2. 개발자 보안 교육 강화
3. 보안 모니터링 체계 구축
4. 인시던트 대응 체계 수립
5. 보안 아키텍처 지속적 개선