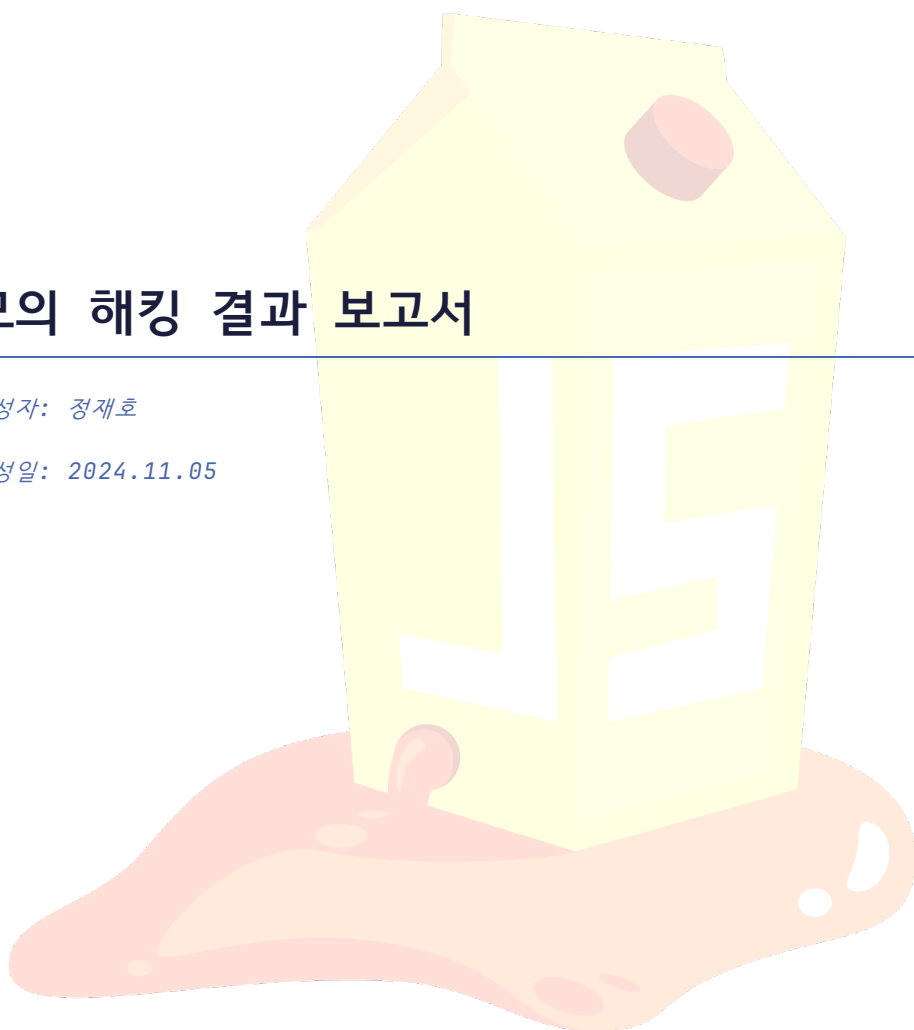


모의 해킹 결과 보고서

작성자: 정재호

작성일: 2024.11.05



모의 해킹 결과 보고서	작성자	정재호	버전	1.0
	수정자	정재호	수정일	2024-11-05

0. 목차

1. 모의해킹 수행 정보..... 3

1.1. 개요..... 3

1.2. 대상..... 3

1.3. 수행 기간..... 3

1.4. 진행 방식..... 4

1.5. 수행 인력..... 4

1.6. 점검 도구..... 4

2. 모의 해킹 결과..... 5

2.1. 총평..... 5

2.2. 결과 요약..... 7

3. 취약점 상세 8

3.1. User Registration (http://ip:port/#/register)..... 8

3.2. Login page (http://ip:port/#/login)..... 12

3.3. Search Field(http://ip:port/#/search)..... 15

4. 보안 권고안 18

4.1. User Registration..... 18

4.2. Login page..... 20

4.3. Search Field..... 22

5. 2 차 공격 시나리오..... 오류! 책갈피가 정의되어 있지 않습니다.

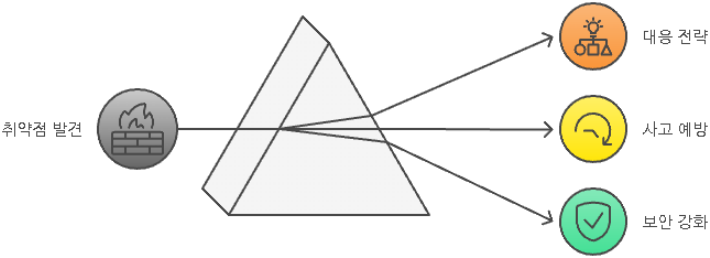
6. 자동화 도구 점검 결과 오류! 책갈피가 정의되어 있지 않습니다.

7. 용어 가이드 오류! 책갈피가 정의되어 있지 않습니다.

모의 해킹 결과 보고서	작성자	정재호	버전	1.0
	수정자	정재호	수정일	2024-11-05

1. 모의해킹 수행 정보

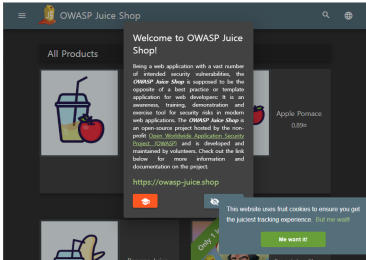
1.1. 개요



본 프로젝트는 웹 어플리케이션에 대한 모의 해킹을 수행한 후 발견된 취약점에 대한 대응 방안을 제시, 차후 발생할 수 있는 침해 사고를 예방하고 보안 수준을 향상할 수 있는 대책을 수립하는데 기여하는 것을 목적으로 한다.

1.2. 대상

본 프로젝트의 점검 대상은 아래와 같다.

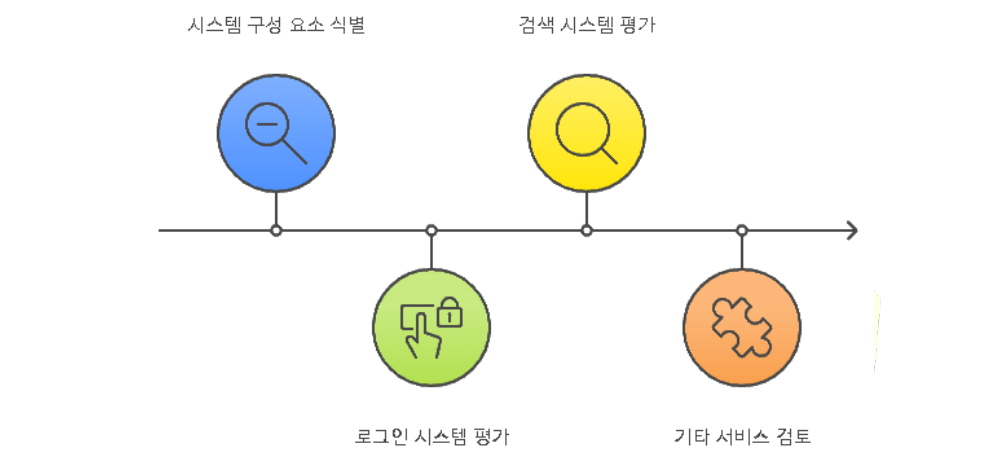
No	Platform	Service (IP)
1	WEB	Juice Shop (https://juice-shop.herokuapp.com/#/) 

1.3. 수행 기간

모의 해킹 기간: 2024 년 11 월 05 일 ~ 2024 년 11 월 12 일 (6 일)

모의 해킹 결과 보고서	작성자	정재호	버전	1.0
	수정자	정재호	수정일	2024-11-05

1.4. 진행 방식

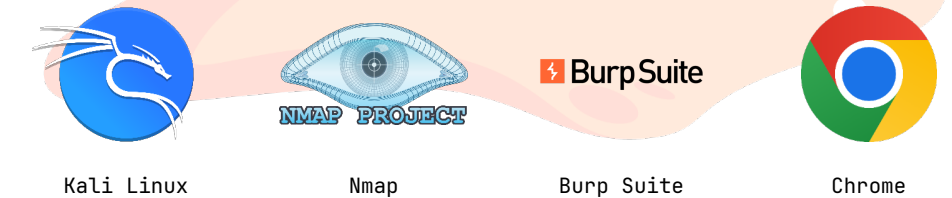


웹 서비스의 로그인 시스템, 검색 시스템, 그 외 시스템에 대한 사전 자료를 공유하고 진행하는 화이트 박스 방식으로 진행

1.5. 수행 인력

Team	Name	Task
Korea IT/ DevSecOps/ Team Firewalls.	정재호 (Jae-ho Jung)	- 웹 서비스 모의 해킹 - 결과 보고서 작성

1.6. 점검 도구



모의 해킹 결과 보고서	작성자	정재호	버전	1.0
	수정자	정재호	수정일	2024-11-05

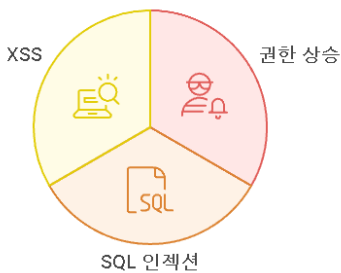
2. 모의 해킹 결과

2.1. 총평

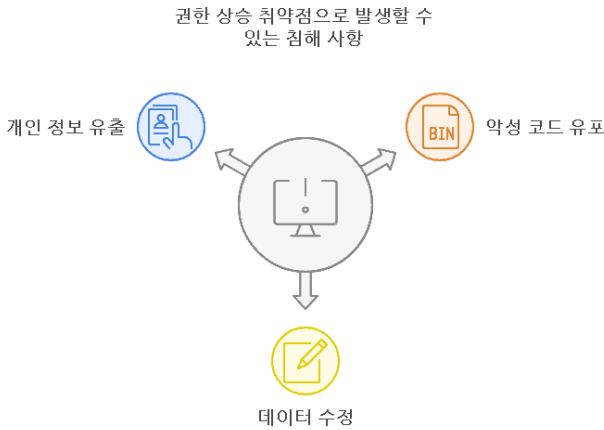
메모 포함[F1]: 결과 요약과 총평이 누락되어 있어 보고서의 전반적인 결과를 한눈에 확인하기 어렵습니다.

메모 포함[F2]: 점검 대상 시스템의 보안 상태에 대한 전반적인 평가(예: "보안 관리 미흡으로 주요 취약점 다수 발견")와 개선 방향을 제시

시스템의 취약점 분포



본 시스템은 회원등록에 1 개, 로그인에 1 개, 제품 검색 란에 1 개, 총 3 개의 취약점(권한 상승 취약점, SQL Injection, XSS 등) 이 존재합니다.



권한 상승 취약점

공격자는 관리자 권한을 획득하여 페이지에 접근하여 데이터를 삭제 또는 변환하여 서비스에 피해를 발생시키거나 서비스의 다른 사용자 정보를 유출시킬 수 있는 위험한 취약점입니다.

모의 해킹 결과 보고서	작성자	정재호	버전	1.0
	수정자	정재호	수정일	2024-11-05

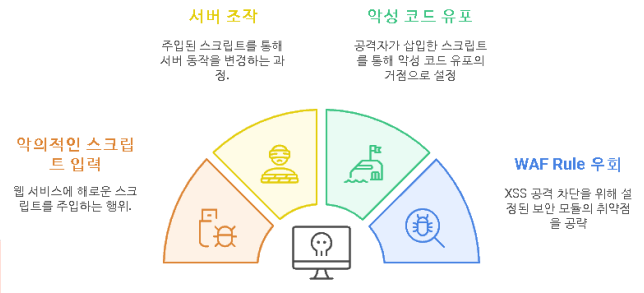
SQL Injection으로 인해 발생 가능한 침해 사항



SQL Injection

공격자는 별도의 제한사항 없이 관리자 계정으로 접속할 수 있게 되어 (권한 상승 취약점과 같이) 서비스에 피해를 발생시킬 수 있습니다.

XSS 취약점

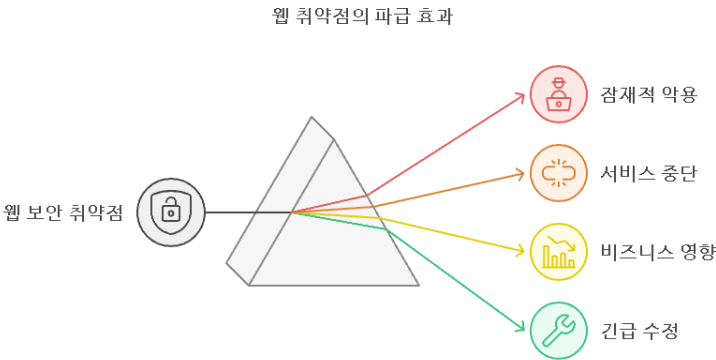


XSS 취약점

웹 서비스에서 제공하지 않는 공격자가 원하는 악의적인 스크립트를 입력하여, 공격자가 원하는 서버 동작을 가능하도록 유도하는 취약점입니다. WAF와 같은 보안 모듈을 통해 일부 보안이 적용되었지만 복잡하거나 잘 사용하지 않는 스크립트 형태에 XSS 취약점이 반응하는 것으로 인해 개선된 보안을 위해 룰을 점검하는 것을 제한합니다.

모의 해킹 결과 보고서	작성자	정재호	버전	1.0
	수정자	정재호	수정일	2024-11-05

“



본 보고서에 서술한 취약점 외에도 많은 웹 보안 관련 취약점은 Juice-shop 서비스에 존재하며 담당자의 의견에 따라 심각도는 상이할 수 있습니다. 다만, 보안에 심각도는 중요하지 않으며 작은 취약점이라 할지라도 이로 인해 발생하는 보안사고는 서비스의 존속에 큰 영향을 줄 수 있기 때문에 신속하게 보수하기를 권고합니다.

”

2.2. 결과 요약

No	대상	취약점	경로	심각도
1	회원 등록	권한 상승 취약점	http://ip:port/#/register	H
2	로그인	SQL Injection	http://ip:port/#/login	H
3	제품 검색	XSS (크로스사이트스크립팅)	http://ip:port/#/search	M

메모 포함[R3]: 발견된 취약점 목록과 그 심각도를 간략히 나열(예: SQL Injection: Critical, XSS: High).

H(높음): 시스템, 조직, 개인에게 심각한 영향을 미치는 상태. 즉각 대응이 필요

M(중간): 조직이나 시스템에 일정 수준의 피해를 줄 수 있으며, 빠른 대응이 필요한 상태

L(낮음): 피해가 경미하거나 영향이 거의 없는 상태. 복구가 쉽고, 추가적인 리스크가 발생할 가능성이 낮음

모의 해킹 결과 보고서	작성자	정재호	버전	1.0
	수정자	정재호	수정일	2024-11-05

3. 취약점 상세

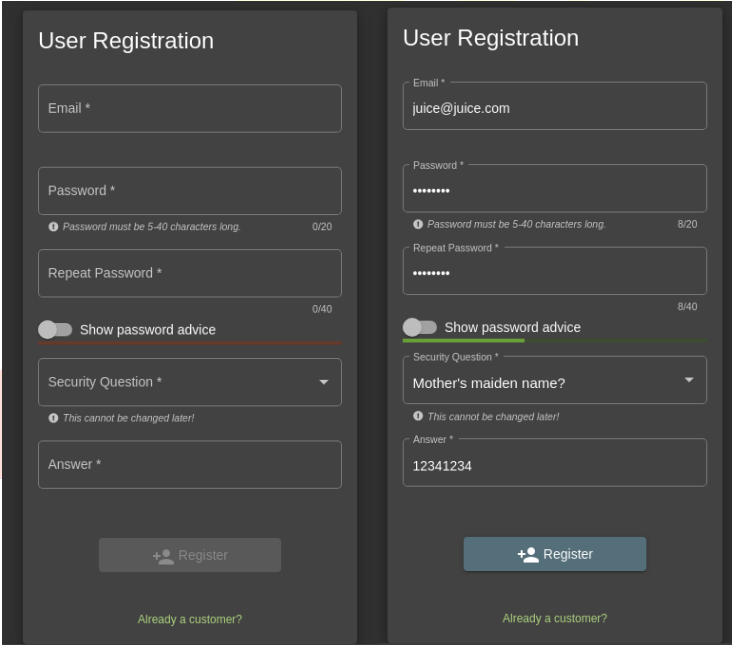
3.1. User Registration (<http://ip:port/#/register>)

3.1.1. **privilege escalation**

구분	설명	
정보	URL	메뉴
	http://ip:port/#/register	메인 페이지> Account> Login> Not yet a customer? 클릭
취약점 상세	회원가입 시 사용자의 입력(요청)을 검증하지 않고 서버에 적용하고 있습니다. 이를 통해 공격자는 일반 사용자를 생성하는 과정에서 서버 요청 데이터를 위.변조하여 관리자 권한을 획득할 수 있습니다.	

메모 포함[F4]: 클라이언트-서버 통신에서 role 필드 변조를 이용한 공격은 충분히 설명되어 있지만, 추가로 공격 성공 사례의 근거를 더 구체적으로 제시할 필요가 있습니다.

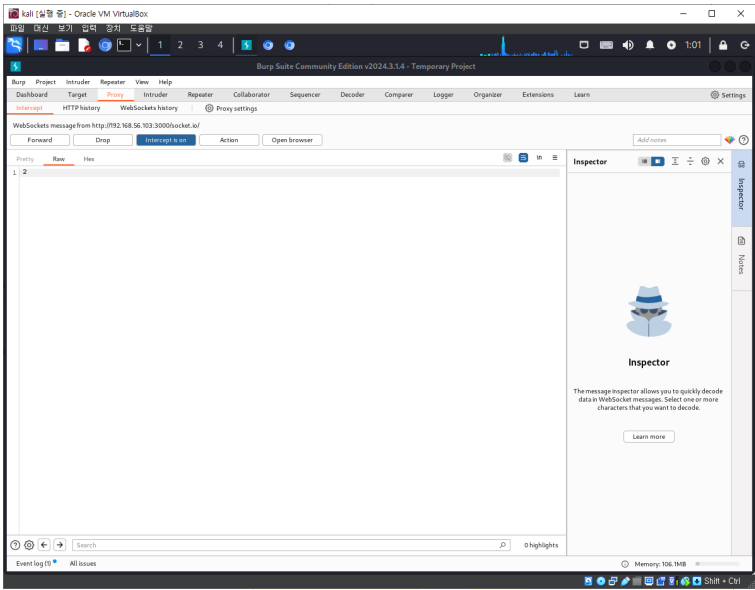
1) 회원가입 페이지에 접속 후 임의의 값을 입력한다.



모의 해킹 결과 보고서	작성자	정재호	버전	1.0
	수정자	정재호	수정일	2024-11-05

2) 클라이언트의 요청을 가로챌 프록시를 활성화한다.

- 'Burp Suite'을 활용한 프록시 활성화



3) 회원가입(Register)버튼을 누른 후 클라이언트 측 요청 데이터를 확인한다.

```
{
  "email": "juice@juice.com",
  "password": "12341234",
  "passwordRepeat": "12341234",
  "securityQuestion": {
    "id": 1,
    "question": "Your eldest siblings middle name?",
    "createdAt": "2024-11-27T02: 21: 47. 780Z",
    "updatedAt": "2024-11-27T02: 21: 47. 780Z"
  },
  "securityAnswer": "12341234"
}
```

- 클라이언트 측 요청 데이터는 json 형태로 서버에 전달되는 것을 확인


모의 해킹 결과 보고서	작성자	정재호	버전	1.0
	수정자	정재호	수정일	2024-11-05

4) json 태그에 “role” 태그와 값 “admin” 추가 후 서버에 전달(Forward)한다.

```
{
  "email": "juice@juice.com",
  "password": "12341234",
  "passwordRepeat": "12341234",
  "role": "admin",
  "securityQuestion": {
    "id": 1,
    "question": "Your eldest siblings middle name?",
    "createdAt": "2024-11-27T02: 21: 47. 780Z",
    "updatedAt": "2024-11-27T02: 21: 47. 780Z"
  },
  "securityAnswer": "12341234"
}
```

5) 생성한 계정으로 로그인 한다.

User Profile



Email:
juice@juice.com

Username:
e.g. SuperUser

Set Username

File Upload:
Choose File No file chosen

Upload Picture

or

Image URL:
e.g. https://www.gravatar.com/avatar/0c81b109c8b05a4a3b7d78693c268

Link Image

메모 포함[F5]: •변조된 요청을 통해 서버에서 발생한 동작을 HTTP 응답 코드 또는 로그 데이터로 구체화.

- 관리자 권한을 가진 사용자를 직접 생성하는데 성공했습니다.

모의 해킹 결과 보고서	작성자	정재호	버전	1.0
	수정자	정재호	수정일	2024-11-05

3.2. Login page (http://ip:port/#/login)

3.2.1. SQL Injection 취약점

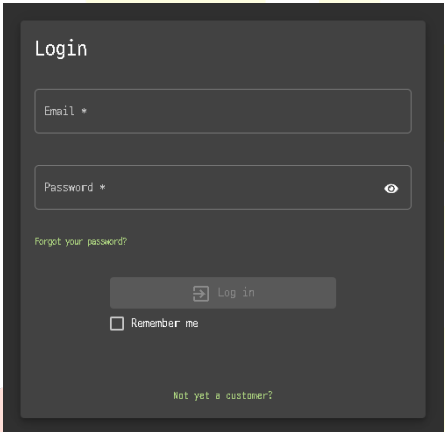
구분	설명	
정보	URL	메뉴
	http://ip:port/#/login	메인 페이지> Account> Login
취약점 상세	로그인 시 사용하는 이메일 입력 필드에 SQL 명령어 주입 취약점이 있음을 확인했습니다. 이를 이용해 공격자는 웹 서비스에서 원하는 정보를 획득하거나 관리자 계정을 획득하여 정보를 위. 변조하여 2차 피해에 악용될 수 있습니다.	

메모 포함[F7]: SQL Injection 취약점 설명은 충분하나, 실제 데이터베이스가 노출될 위험성을 구체적으로 서술할 수 있습니다.

메모 포함[F8]: 데이터베이스 구조, 계정 목록, 비밀번호 해시 등이 노출되는 실제 예시를 간단히 첨부

메모 포함[F9]: •가능한 2차 피해를 더 구체화(예: 비밀번호 해시를 통한 무차별 대입 공격, 고객 데이터 유출 등).

1) 로그인 페이지에 접속한다



모의 해킹 결과 보고서	작성자	정재호	버전	1.0
	수정자	정재호	수정일	2024-11-05

2) 이메일 입력 필드에 (')를 입력해 본다.

Login

[object Object]

Email *

•••••

Password *

•••••

Forgot your password?

Log in

Remember me

Not yet a customer?

- SQL Injection 취약점 확인

3) 간단한 SQL 구문(' or 1=1 --) 입력

Login

Email *

' or 1=1 --

Password *

•••••


Forgot your password?

Log in

Remember me

Not yet a customer?

User Profile



Email: admin@juice-sh.op

Username: j.s. @admin

Set Username

File Upload: ☐ 파일 선택 ☐ 선택된 파일 없음

Upload Picture

or

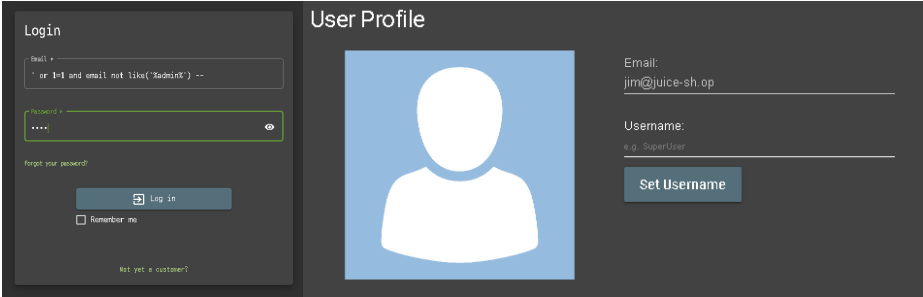
Image URL:

Link Image

- 관리자 계정으로 로그인 되는 것을 확인했다.

모의 해킹 결과 보고서	작성자	정재호	버전	1.0
	수정자	정재호	수정일	2024-11-05

4) SQL 구문을 수정해 관리자 계정 외 다른 계정으로 로그인 시도



- 웹 서비스의 임의의 사용자 계정으로 로그인

[보안 권고안]

1. 사용자의 입력을 파라미터로 사용할 때 문자열이 명령어로 인식되지 않도록 해야 합니다.
2. 에러 메시지를 노출하지 않도록 해야 합니다.
3. 최소 권한 원칙을 준수해야 합니다.

모의 해킹 결과 보고서	작성자	정재호	버전	1.0
	수정자	정재호	수정일	2024-11-05

3.3. Search Field(http://ip:port/#/search)

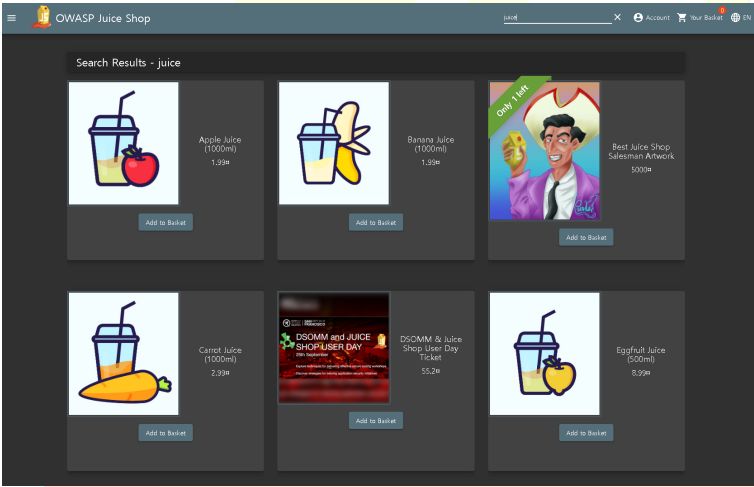
3.3.1. XSS(크로스 사이트 스크립트) 취약점

구분	설명	
정보	URL	메뉴
	http://ip:port/#/search	메인 페이지> 돋보기(검색) 버튼
취약점 상세	검색 함수에 xss 취약점이 존재하는 것을 확인했습니다. WAF 나 그 외 방어 수단으로 방어했지만 에러를 유도해서 원하는 스크립트를 실행하도록 하는 방식에는 반응함으로 xss 취약점 보완이 필요할 것으로 판단됩니다.	

메모 포함[F10]: 취약점 활용 가능성에 대해 더 구체적으로 설명하면 좋습니다.

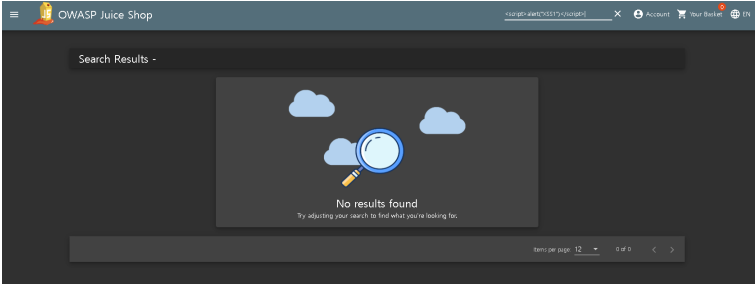
- "XSS 를 통해 탈취한 세션 정보를 어떻게 활용할 수 있는지"를 서술.
- 관리자 계정 탈취나 피싱 페이지 생성 등 추가적인 시나리오를 제공.

1) 검색 필드에 텍스트 입력



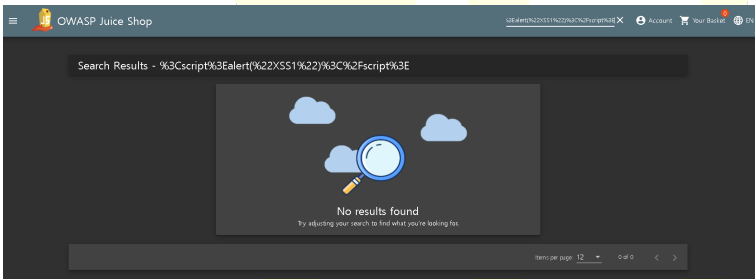
모의 해킹 결과 보고서	작성자	정재호	버전	1.0
	수정자	정재호	수정일	2024-11-05

2) "<script>alert("XSS1")</script>" 입력



- 기본적인 xss 유도 스크립트는 안전하게 방어하는 것으로 확인

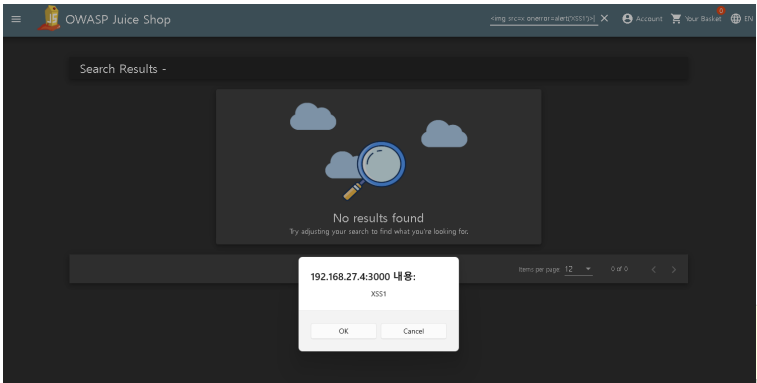
3) html 인코딩 된 xss 유도 스크립트 입력



- "%3Cscript%3Ealert(%22XSS1%22)%3C%2Fscript%3E"
- 스크립트로 인식되지 않고 문자열로 검색이 진행되는 것으로 확인

모의 해킹 결과 보고서	작성자	정재호	버전	1.0
	수정자	정재호	수정일	2024-11-05

4) 에러를 유도한 변형 스크립트 입력



- “”
- WAF 와 같은 XSS 입력 방어 책을 우회하여 스크립트 실행 가능한 점 확인
- 이로 인해 공격자는 서버에 악의적인 코드 실행이 가능한 것을 확인

[보안 권고안]

1. 사용자의 입력 검증: HTML 태그 및 JavaScript 코드 제거
2. 정규 표현식을 활용해 안전한 패턴만 허용
3. 출력값 인코딩: HTML 컨텍스트 출력되는 값은 반드시 escape 처리
4. CSP, WAF 설정

모의 해킹 결과 보고서	작성자	정재호	버전	1.0
	수정자	정재호	수정일	2024-11-05

4. 보안 권고안

4.1. User Registration

취약점 항목	privilege escalation
취약점 개요	<div><div>민감한 데이터 노출</div><div>역할 조작 시도</div><div>권한 상승</div><div>2차 피해</div></div> <p>서버에서 생성하는 쿠키(token) 상세한 계정 정보를 저장하는 것을 확인. 공격자로 하여금 계정 정보를 구성하는 컬럼의 종류를 파악할 수 있는 정보가 제공될 여지가 있다. 쿠키를 통해 권한을 식별하는데 사용하는 “role” 태그가 노출되었으며 이 정보를 이용해 공격자는 관리자 권한(role)을 가진 사용자를 생성하는 시도를 할 수 있으며, 그로 인해 공격자는 관리자 권한을 갖고 서버에 2차 피해를 입힐 수 있게 된다. 권한 상승 취약점으로 인한 대표적인 2차 피해는 “데이터 유출 사고” 및 “데이터 조작 및 삭제”, “시스템 및 서비스 장애 유발” 등이 있다.</p>

메모 포함[F11]: 보안 권고안

구체적 코드 사례 보완

●권고안에 포함된 코드 예시는 적절하나, 더 상세한 코드 사례를 제시해 실질적인 구현 방법을 명확히 하는 것이 좋습니다.

●예시:

- Prepared Statement 적용 예시에서, SQL Injection 방지에 유용한 구체적 라이브러리 활용법(PHP PDO, Python SQLAlchemy 등).
- XSS 방지에서 CSP(Content Security Policy) 설정 샘플 추가.

보안 권고의 우선순위 제시

●보완 필요: 취약점의 심각도에 따라 보안 권고의 우선순위를 제시하면 개선 방향이 명확해집니다.

●제안:

- 예를 들어 SQL Injection 은 Critical 로 분류하며, 즉시 조치 필요.
- XSS 는 High 로 분류하며, 추가 검토 및 조치 필요.

모의 해킹 결과 보고서	작성자	정재호	버전	1.0
	수정자	정재호	수정일	2024-11-05

보안 조치 방법

입력 검증 및 필터링

사용자 역할 검증
기본 값 설정

계정 권한 관리

적절한 로그 기록

실시간 로그 파일
의심스러운 패턴 모니터링

API 및 데이터 통신 보안

HTTPS 사용

<사용자의 입력 검증 및 필터링>

클라이언트에서 제공되는 모든 입력 데이터는 신뢰하지 말고 서버 측에서 검증하고 사용해야 합니다.

```
if ($_POST['role'] !== 'customer') {  
    die('Invalid role specified.');
```

또는 role 값과 같이 관리자 외 조작할 경우가 없는 설정 값의 경우 서버에서 특정 값으로 임의로 지정하여 클라이언트에서 조작하여 사용할 수 없도록 하면 안전합니다.

```
$role = 'customer'; // 모든 신규 사용자는 고객으로 고정  
$query = "INSERT INTO users (username, password, role) VALUES ('$username',  
'$hashed_password', '$role')";
```

<API, 데이터 통신 보안>

클라이언트의 요청을 캡처하고 변조할 수 없도록 HTTPS 를 사용하는 것이 좋습니다.

<적절한 로그 기록>

사용자의 회원가입, 로그인 기록을 저장하는 log 파일을 실시간으로 생성하여 의심스러운 행동 패턴이 발견되면 즉시 대응할 수 있는 환경을 구성하는 것이 좋습니다.

모의 해킹 결과 보고서	작성자	정재호	버전	1.0
	수정자	정재호	수정일	2024-11-05

4.2. Login page

취약점 항목	SQL Injection
취약점 개요	<div><div><div>SQL 인젝션 취약점</div><div>취약점 악용</div><div>무단 데이터베이스 접근</div><div><div>무단 사용자 계정 접근</div><div>무단 서비스 데이터 접근</div></div><div><div>데이터 유출</div><div>데이터 조작 및 삭제</div><div>시스템 및 서비스 중단</div></div></div><p>로그인 시 사용되는 Email 입력 필드에 SQL 명령어 입력 취약점이 있음을 확인. 해당 취약점을 이용해 서비스의 데이터 베이스 구조, 사용자 계정, 서비스 데이터 등에 권한이 없는 사용자가 접근할 수 있게 되어 치명적인 2 차 피해로 이어질 우려가 있음.</p></div>

보안 조치 방법

에러 메시지 제어

상세한 에러 메시지가 노출되지 않도록 방지

Prepared Statements

쿼리와 데이터를 분리하기 위해 Prepared Statements 사용

최소 권한

데이터베이스 계정에 최소한의 필요한 권한 부여

입력 검증

사용자 입력이 허용된 형식과 기준을 충족하는지 확인

<Prepared Statements 사용>

SQL 쿼리문을 작성할 때 Prepared statements 또는 Parameterized Queries 를 사용하여, 쿼리와 입력 데이터를 분리해야 합니다. 사용자가 입력한 값이 SQL 쿼리의 일부가 되지 않고, 데이터로서 처리되도록 할 수 있습니다.

<사용자의 입력 검증 및 필터링>

사용자의 입력을 받을 때 반드시 입력을 검증하고 허용된 형식인지 확인해야 합니다. Email 형식에서 사용가능한 특수문자 외 다른 문자들은 필터를 걸어 제외하는 것이 좋습니다.

<최소 권한 원칙>







서비스의 데이터베이스에 접근할 때 사용하는 계정의 최소한의 권한만 부여해야 합니다. 로그인 검증용 쿼리에는 SELECT 권한만 필요하며, 데이터를 수정할 필요가 없음으로 UPDATE, DELETE 권한을 제외해야 2 차 피해를 최소화할 수 있습니다.

<에러 메시지 노출 방지>

에러 메시지에 구체적인 데이터베이스 정보를 노출하면 공격자의 SQL Injection 공격에 이용당할 수 있습니다. 에러 메시지는 노출하지 말고, 일반화된 메시지를 출력하거나 로그만 기록하는 것이 좋습니다.

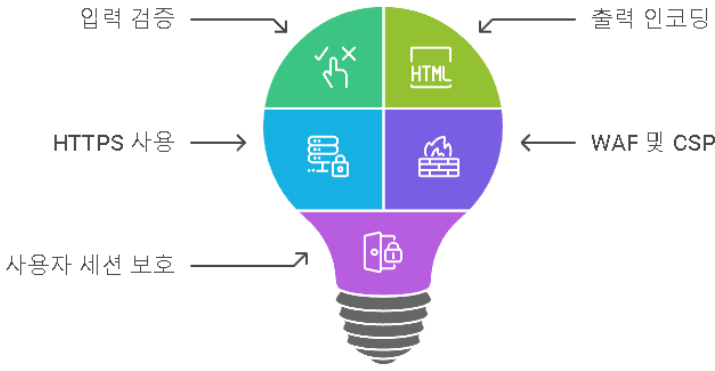
모의 해킹 결과 보고서	작성자	정재호	버전	1.0
	수정자	정재호	수정일	2024-11-05

4.3. Search Field

취약점 항목	XSS (크로스 사이트 스크립트)
취약점 개요	<div><div><div><div><div>사용자가 상품 검색</div><div></div></div><div><div>공격자가 취약점을 악용함</div><div></div></div><div><div>잠재적인 2차 공격</div><div></div></div></div><div><div></div><div><div><div>XSS 취약점 탐지됨</div><div></div></div><div><div>서버가 악성 코드를 실행함</div><div></div></div></div></div><p>사용자가 원하는 상품을 찾기 위해 사용하는 검색 필드에 XSS (크로스 사이트 스크립트) 취약점이 있음을 확인. 공격자는 해당 취약점을 이용해 서버에 악의적인 코드를 실행하여 2차 공격에 악용할 여지가 있습니다.</p></div></div>

모의 해킹 결과 보고서	작성자	정재호	버전	1.0
	수정자	정재호	수정일	2024-11-05

보안 조치 방법



입력 검증

출력 인코딩

HTTPS 사용

WAF 및 CSP

사용자 세션 보호

<사용자의 입력 검증 및 필터링>

사용자의 입력을 받을 때 반드시 입력을 검증하고 허용된 형식인지 확인해야 합니다.

```
function sanitizeInput(input) {  
  return input.replace(/<|>|'"/g, '');  
}
```

입력 값에 포함된 특수문자를 제거하는 함수를 사용하거나 예상되는 입력 형식과 일치하지 않는 경우 입력을 거부하는 방법이 좋을 것입니다.

<출력값 인코딩>

사용자 입력값을 HTML 페이지에 출력하기 전 HTML 엔티티로 변환하여 브라우저가 스크립트로 인식하지 못하게 만듭니다.

계속

23

모의 해킹 결과 보고서	작성자	정재호	버전	1.0
	수정자	정재호	수정일	2024-11-05

```
graph TD; A[사용자 입력] --> B{HTML 컨텍스트인가?}; B -- 예 --> C[HTML 엔티티로 변환]; B -- 아니오 --> D{JavaScript 컨텍스트인가?}; D -- 예 --> E[JSON.stringify() 사용]; D -- 아니오 --> F[encodeURIComponent() 사용]; C --> G[안전한 출력]; E --> G; F --> G;
```

```
const escapeHtml = (str) => {
  return str.replace(/&/g, '&amp;')
    .replace(/</g, '&lt;')
    .replace(/>/g, '&gt;')
    .replace(/"/g, '&quot;')
    .replace(/'/g, '&#039;');
};
```

위 코드와 같이 HTML 컨텍스트에서 <와 >를(그 외 script 를 구성하는 특수문자들 포함) < > 로 변환하거나, JavaScript 컨텍스트에서 문자열을 JSON.stringify() 함수로 변환하거나 URL 컨텍스트에서는 encodeURIComponent()를 사용하는 것이 좋습니다.

<HTTPS 사용>

공격자가 악성 스크립트를 삽입하지 못하도록 보안이 강화된 http 프로토콜을 사용하는 것이 좋습니다. 추가로 SSL/TLS 인증서를 설정하여 HTTPS 로만 서버에 연결되도록 강제하는 것이 좋습니다.

<WAF & CSP>

WAF 는 사용자의 악의적인 입력을 감지, XSS 를 탐지하여 차단하는 역할을 수행합니다. CSP 는 신뢰할 수 있는 도메인에서 제공되는 스크립트만 실행되도록 시스템을 제한하는 서비스입니다. 두 서비스를 적용하여 공격자의 악의적인 스크립트 입력을 미연에 차단하고 방어할 수 있습니다.

<사용자 세션 보호>

XSS 공격으로 인해 서비스의 사용자(관리자 계정 포함) 세션 정보를 탈취당할 가능성이 있습니다. 이를 사전에 HttpOnly, Secure 플래그를 사용해 보호하는 것이 좋습니다.