



OWASP

-기획부-



목차

- 활동 일정

1. OWASP란?

2. OWASP JUICE

- Node.JS 설치하기
- OWASP JUICE 설치하기
- 스코어 보드 띄워보기
- 공격하고 레포트 작성하기

3. OWASP ZAP

- 사이트 자동 공격 및 분석
- 취약점 레포트 확인





OWASP란?

- Open Web Application Security Project
- **OWASP란?**
 - Open Web Application Security Project의 약자로, 웹 애플리케이션 보안을 목표로 하는 **비영리 단체**.
 - 2001년에 설립되어, 전 세계적으로 보안 전문가와 개발자들이 참여.
 - OWASP는 **무료로 제공되는 도구와 문서**를 통해 개발자와 기업들이 안전한 소프트웨어를 만들 수 있도록 함
- **목표**
 - 웹 애플리케이션의 보안성을 높이고 취약점 공격을 방지.
 - 개발자와 보안 전문가 간의 협력을 통해 더 안전한 디지털 환경 구축.






웹사이트

- <https://owasp.org/>

Please support the OWASP mission to improve software security through Open Source initiatives and community education. [Donate Now](#) X

 PROJECTS CHAPTERS EVENTS ABOUT Q

[Store](#) [Donate](#) [Join](#)

Ex **o** **world** **urity**

Driven **Search OWASP** [Q](#)

[Browse All Projects...](#)

- [OWASP Top Ten](#)
- [Dependency Track](#)
- [Juice Shop](#)
- [Mobile Application Security](#)
- [ModSecurity Core Rule Set](#)
- [Software Assurance Maturity Model \(SAMM\)](#)
- [Web Security Testing Guide](#)
- [Start a New Project...](#)
- [Community Contributions](#)
- [Google Summer of Code 2024](#)

Upcoming at OWASP

OWASP Event Calendar

2024년 11월

일	월	화	수	목	금	토	일
27	28	29	30	31	11월 1일	2	3
2024 Global Board of Directors Election - Vote Now							
5	6	7	8	9	10	11	12





OWASP의 주요 활동

- OWASP의 주요 프로젝트
- **OWASP Top 10**
 - 가장 널리 알려진 프로젝트로, 가장 중요한 웹 취약점 10가지를 정리한 목록
 - 보안 위험에 대한 인식을 높이고, 개발자가 우선적으로 해결해야 할 문제를 명확화
- **OWASP ZAP (Zed Attack Proxy)**
 - 오픈소스 웹 애플리케이션 취약점 스캐너로, 보안 테스트를 자동화할 수 있는 강력한
 - 초보자부터 전문가까지 모두 사용 가능.
- **ASVS (Application Security Verification Standard)**
 - 소프트웨어 보안 검증을 위한 체계적인 가이드라인 제공.
- **Dependency-Check**
 - 애플리케이션에 사용되는 라이브러리의 보안 취약점을 탐지.





OWASP Top 10 집중 탐구(2021)

- <https://owasp.org/www-project-top-ten/>
- **Broken Access Control:** 접근 제어가 제대로 설정되지 않아 발생하는 문제.
- **Cryptographic Failures:** 암호화의 부적절한 사용으로 인한 데이터 유출.
- **Injection:** SQL Injection 등, 공격자가 입력값을 조작하여 시스템 명령어를 실행.
- **Insecure Design:** 보안성을 고려하지 않은 설계로 인해 발생하는 취약점.
- **Security Misconfiguration:** 잘못된 보안 설정으로 인한 문제.
- **Vulnerable and Outdated Components:** 오래된 라이브러리 사용으로 인한 취약점.
- **Identification and Authentication Failures:** 인증 및 세션 관리의 실패.
- **Software and Data Integrity Failures:** 무결성 검증 실패.
- **Security Logging and Monitoring Failures:** 보안 로그와 모니터링 부재.
- **Server-Side Request Forgery (SSRF):** 서버가 공격자의 요청을 실행하여 내부 시스템 접근.





OWASP Juice Shop이란?

- <https://owasp.org/www-project-juice-shop/>

- ❑ OWASP Juice Shop은 오픈소스 웹 애플리케이션으로, 취약점 학습 플랫폼
- ❑ 가상의 전자상거래 웹사이트를 가장하고 있지만, 실제로는 다양한 보안 취약점을 의도적으로 포함하고 있음
- ❑ 웹 애플리케이션 보안에 대한 실습 경험을 제공하여, 개발자와 보안 전문가가 취약점을 탐지하고 악용하는 방법을 학습할 수 있도록 설계
- ❑ 2010년대 이후 PHP 기반의 웹 어플리케이션이 많이 줄어들고 자바스크립트와 관련 프레임워크들이 대세가 됐기 때문에 주스샵 또한 NodeJS, Express, Angular 등을 이용해 제작





OWASP Juice Shop

- 특징
- 다양한 취약점 제공:
 - OWASP Top 10을 포함한 다양한 보안 취약점이 포함되어 있음.
 - **SQL Injection, XSS, CSRF, Broken Access Control, SSRF** 등.
- 난이도별 과제:
 - 초급부터 고급까지의 취약점 과제가 준비되어 있어, 학습자 수준에 따라 점진적으로 학습 가능.
 - 취약점 발견 및 해결 후 **배지나 점수**를 획득.
- 실습 중심의 학습:
 - 안전한 환경에서 **실제 공격 시뮬레이션** 가능.
 - 이를 통해 **공격자의 관점**에서 취약점을 학습하고, **방어 전략**을 수립.
- 완전한 사용자 친화성:
 - 웹 브라우저에서 실행 가능.
 - **설치 및 설정**이 간단하며, 로컬 환경 또는 클라우드에서도 쉽게 실행 가능.
- 오픈소스:
 - GitHub에서 소스 코드 확인 및 기여 가능.
 - 학습용으로 수정하거나 새로운 기능을 추가할 수 있음.





OWASP JUICE SHOP

- 내 로컬에 설치하기

1. Node js 설치하기(이미 설치된 분들은 cmd에 node -v 해서 버전 뜨면 설치되어 있음)

<https://nodejs.org/en> (위 사이트 들어가서 그냥 바로 download 버튼 누르고 실행파일 설치)

2. 폴더 하나 만들고 Git bash (또는 CMD OR POWERSHELL)에 다음 명령어 실행

```
git clone https://github.com/juice-shop/juice-shop.git --depth 1
```





JUICE SHOP에 들어오고 나서

- 하단 명령어 실행

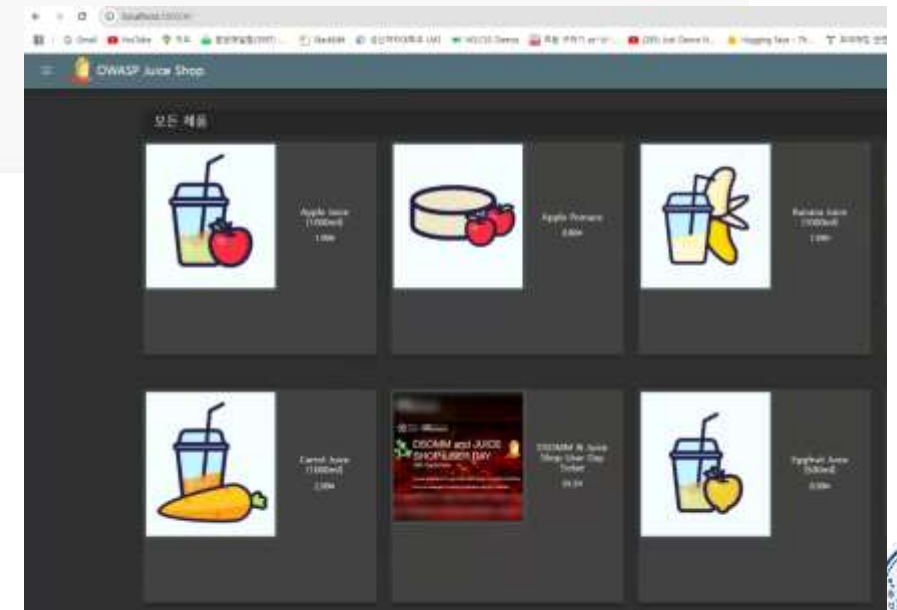
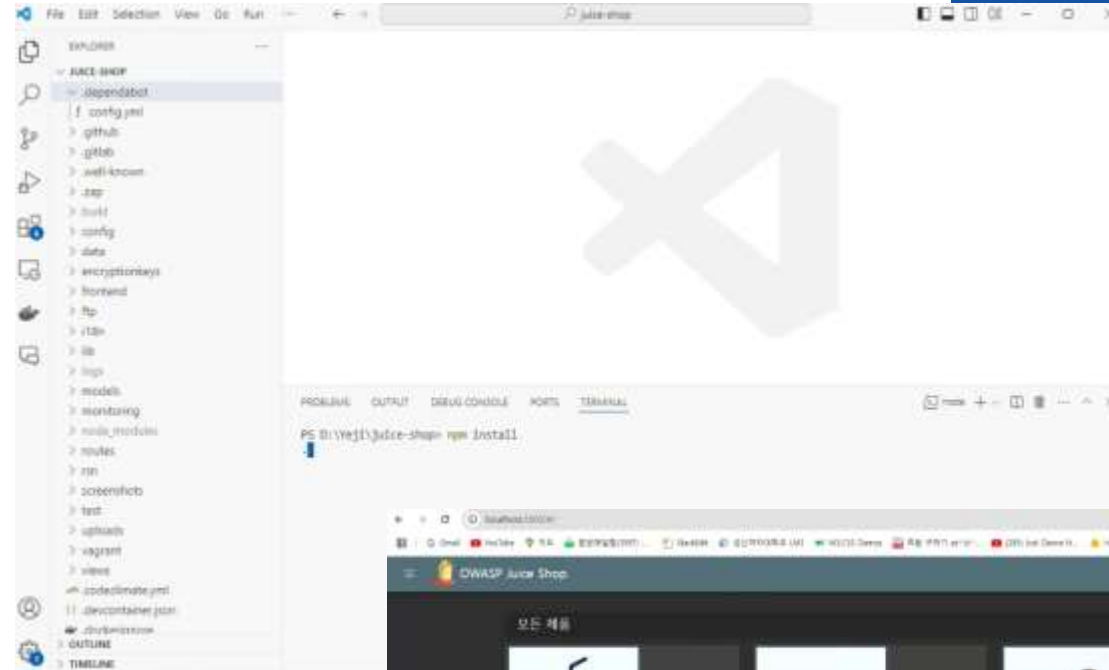
npm install

(한참동안 설치됨...주의)

npm run start

이후, 웹 브라우저에

localhost:3000 쳐서 들어가기

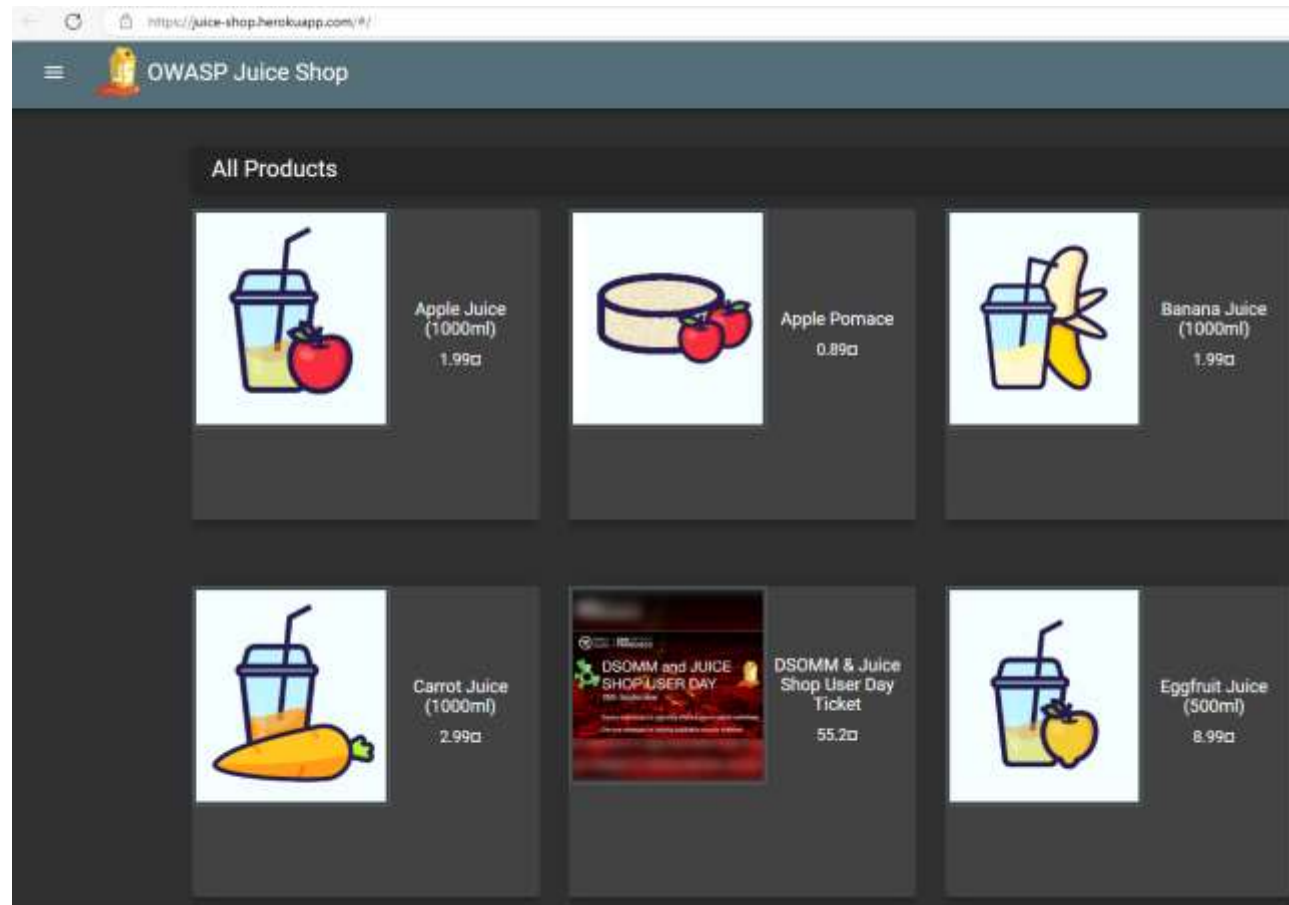




운명의 장난으로 인해 설치가 안 되는 경우

- 클라우드 환경에서도 실행 가능

☐ <https://juice-shop.herokuapp.com/#/>





제일 처음 문제

■ Score board

□ Score board 찾기


- 해석: url을 잘 찍어서 들어가거나
- F12 버튼 눌러서 실제 URL을 알아내서 스코어 보드를 찾으세요


Welcome to OWASP Juice Shop!


Being a web application with a vast number of intended security vulnerabilities, the **OWASP Juice Shop** is supposed to be the opposite of a best practice or template application for web developers: It is an awareness, training, demonstration and exercise tool for security risks in modern web applications. The **OWASP Juice Shop** is an open-source project hosted by the non-profit **Open Worldwide Application Security Project (OWASP)** and is developed and maintained by volunteers. Check out the link below for more information and documentation on the project.


<https://owasp-juice.shop>

 This application is riddled with security vulnerabilities. Your progress exploiting these is tracked on a *Score Board*.

 You won't find a link to it in the navigation or side bar, though. Finding the *Score Board* is in itself actually one of the hacking challenges.

 You could just start guessing the URL of the *Score Board* or comb through the client-side JavaScript code for useful information.

 You find the JavaScript code in the DevTools of your browser that will open with **F12**.





Score board

제한시간 3분 도전

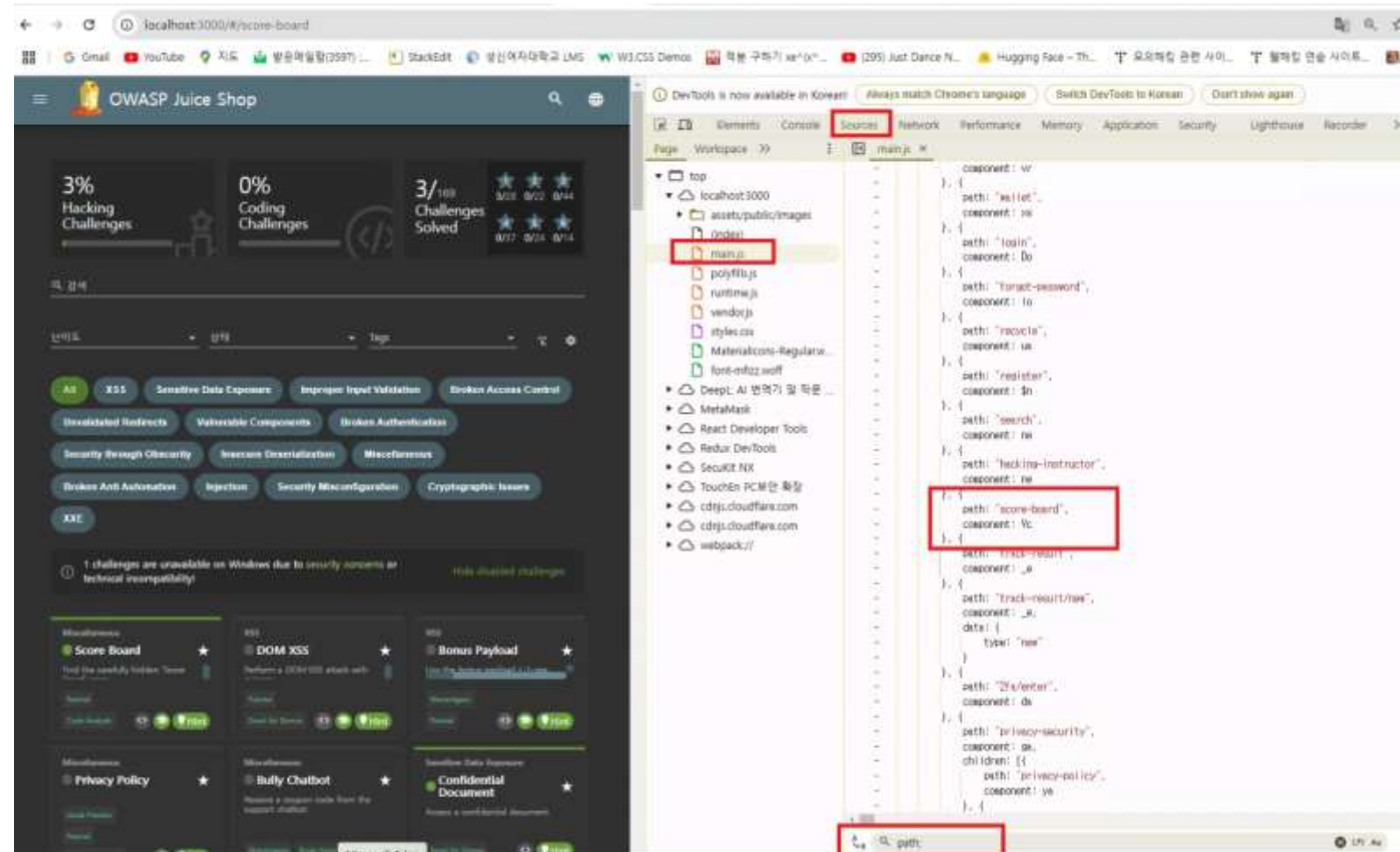




정답

■ score-board

F12 > sources > main.js
Ctrl+F 이후 path:
라고 검색





살펴보기

localhost:3000/#/score-board

검색

난이도 상태 Tags

All XSS Sensitive Data Exposure Improper Input Validation Broken Access Control Unvalidated Redirects Vulnerable Components Broken Authentication Security through Obscurity Insecure Deserialization Miscellaneous Broken Anti Automation Injection Security Misconfiguration Cryptographic Issues XXE

1 challenges are unavailable on Windows due to security concerns or technical incompatibility! Hide disabled challenges

<p>Miscellaneous</p> <p>Score Board ★</p> <p>Find the carefully hidden 'Score Board' page.</p> <p>Tutorial Code Analysis Hint</p>	<p>XSS</p> <p>DOM XSS ★</p> <p>Perform a DOM XSS attack with <code><iframe src=javascript:alert('xss')></code>.</p> <p>Tutorial Good for Demos Hint</p>	<p>XSS</p> <p>Bonus Payload ★</p> <p>Use the bonus payload <code><iframe width=100% height=166% scrolling=no frameborder=no allow=autoplay src=https://w.soundscloud.com/player?url=https://w.soundscloud.com/tracks/77185447&color=5201169200a9c61e&visualizer=0</code>.</p> <p>Shenigans Tutorial Hint</p>	<p>Miscellaneous</p> <p>Privacy Policy ★</p> <p>Read our privacy policy.</p> <p>Good Practice Tutorial Good for Demos Hint</p>
<p>Miscellaneous</p> <p>Bully Chatbot ★</p> <p>Receive a coupon code from the support chatbot.</p> <p>Shenigans Brute Force Hint</p>	<p>Sensitive Data Exposure</p> <p>Confidential Document ★</p> <p>Access a confidential document.</p> <p>Good for Demos Hint</p>	<p>Security Misconfiguration</p> <p>Error Handling ★</p> <p>Provoke an error that is neither very gracefully nor consistently handled.</p> <p>Presingulate Hint</p>	<p>Sensitive Data Exposure</p> <p>Exposed Metrics ★</p> <p>Find the endpoint that serves usage data to be scraped by a popular monitoring system.</p> <p>Good Practice Hint</p>
<p>Miscellaneous</p> <p>Mass Dispel ★</p> <p>Close multiple 'Challenge solved'-notifications in one go.</p> <p>Hint</p>	<p>Improper Input Validation</p> <p>Missing Encoding ★</p> <p>Retrieve the photo of Sjoern's cat in 'melee combat-mode'.</p> <p>Shenigans Hint</p>	<p>Unvalidated Redirects</p> <p>Outdated Allowlist ★</p> <p>Let us redirect you to one of our crypto currency addresses which are not promoted any longer.</p> <p>Code Analysis Hint</p>	<p>Improper Input Validation</p> <p>Repetitive Registration ★</p> <p>Follow the DRY principle while registering a user.</p> <p>Hint</p>
<p>Broken Access Control</p> <p>Web3 Sandbox ★</p> <p>Find an accidentally deployed code sandbox for writing smart contracts on the fly.</p> <p>Web3 Hint</p>	<p>Improper Input Validation</p> <p>Zero Stars ★</p> <p>Give a devastating zero-star feedback to the store.</p> <p>Hint</p>	<p>XSS</p> <p>Reflected XSS ★★</p> <p>Perform a reflected XSS attack with <code><iframe src=javascript:alert('xss')></code>.</p> <p>Tutorial Danger Zone Good for Demos Hint</p>	<p>Injection</p> <p>Login Admin ★★</p> <p>Log in with the administrator's user account.</p> <p>Tutorial Good for Demos Hint</p>
<p>Broken Access Control</p> <p>Admin Section ★★</p> <p>Access the administrative section of the store.</p>	<p>Broken Authentication</p> <p>Password Strength ★★</p> <p>Let us verify the administrator's user credentials without additional challenges. There are no hints.</p>	<p>Broken Access Control</p> <p>View Basket ★★</p> <p>View a deprecated API endpoint that was not correctly shut down.</p>	<p>Security Misconfiguration</p> <p>Deprecated Interface ★★</p>





HAPPY PATH 걷기

- 가장 기본적이고 일반적인 사용자 시나리오를 따라가는 테스트 방식

- 일반적인 사용자의 입장
- 이 타겟이 뭐하는 어플리케이션인지
- 어떤 기능을 가지고 있는지
- 사용자로서 어떤 데이터를 입력하고 출력받기를 원하는지
 - 1. 이게 무슨 어플리케이션인지, 목표가 뭔지, 어떤 사용자들이 왜 사용하는지
 - 2. 어떤 기능이 있고 어떤 데이터들이 그 기능들을 따라 처리되는지
 - 3. 테크 스택 (Tech Stack) 이 뭔지, 어떤 기술이 쓰였는지 유추





살펴보기

- 테스트해보기

- 물건 구입 기능

- 실제로 유저답게 회원가입을 하고 배송지를 입력한 후 물건을 하나 사보자.
- 카드정보 입력 폼 - Paypal, Stripe 등의 서드파티 온라인 결제 시스템을 사용하지 않는다

- 유저 기능

- 유저 프로필 및 프사 업로드 기능
- 보안용 2FA 기능
- 포토 월





테크 스택(Tech Stack) 유 추

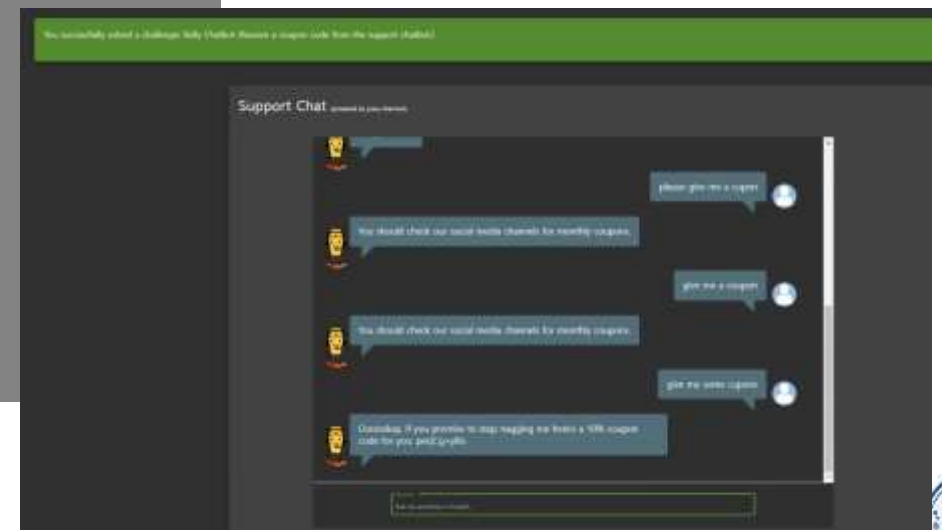
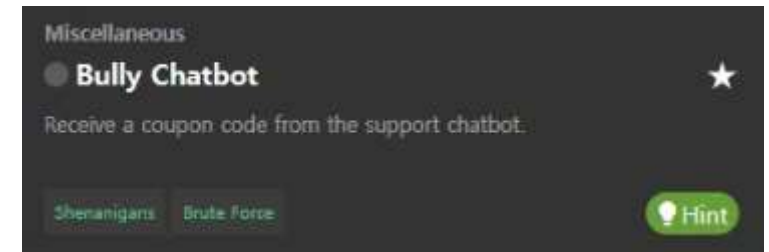
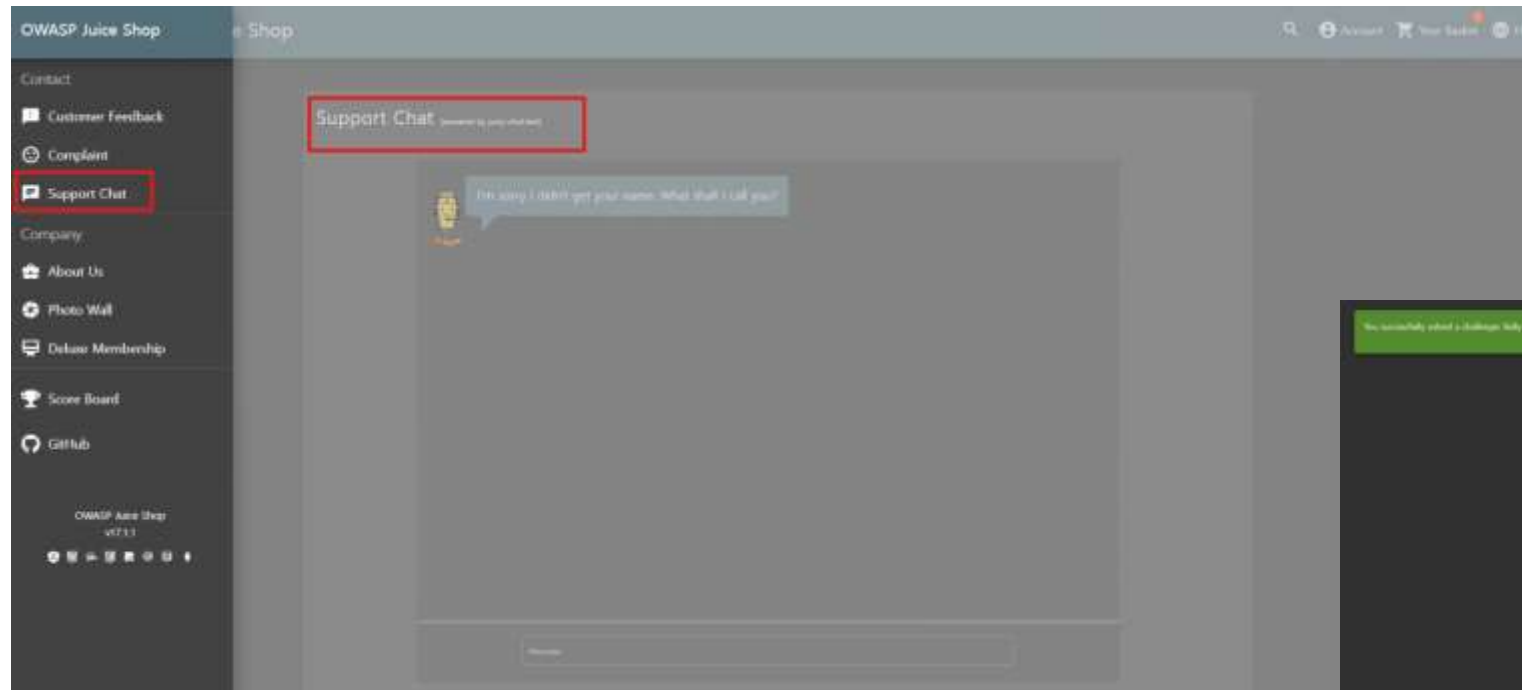
- ❑ `http://localhost:3000/#/about`
`http://localhost:3000/#/search`
`http://localhost:3000/#/basket`
<http://localhost:3000/#/address/select>
- ❑ 이 어플리케이션은 자바스크립트 기반의 Single Page Application
- ❑ 다만 Angular, Express, React 등의 어떤 프레임워크를 쓰
는지는 아직 확인 불가





- 놀아보기

□ 챗봇괴롭히기 문제





DOM XSS 문제

- Tutorial 버튼 눌러서 함께 진행





- ❑ OWASP ZAP은 오픈소스 웹 애플리케이션 보안 테스트 툴
- ❑ Zed Attack Proxy의 약자로, 프록시 서버로 동작하며 애플리케이션과 클라이언트 간의 트래픽을 분석
- ❑ 목적: 취약점을 쉽게 식별하고 보안성을 개선하도록 지원.





주요 특징

- **오픈소스 및 무료**
 - 누구나 무료로 사용할 수 있는 **강력한 도구**.
 - 지속적인 커뮤니티 지원과 업데이트.
- **사용자 친화적 인터페이스**
 - 초보자와 전문가 모두 쉽게 사용할 수 있도록 설계된 **GUI** 제공.
 - 자동화된 취약점 스캐닝과 수동 분석 모두 가능.
- **다양한 플랫폼 지원**
 - **Windows, macOS, Linux** 등 여러 운영 체제에서 실행 가능.
- **확장성**
 - 다양한 플러그인과 애드온을 통해 기능 확장





주요 기능

- **프록시 기능**
 - 애플리케이션과 클라이언트 간의 트래픽을 가로채서 분석.
 - 요청/응답 내용을 수정하여 취약점 테스트 가능.
- **자동 스캔**
 - 빠르고 간단한 취약점 탐지.
 - 기본적인 OWASP Top 10 항목 탐지.
- **수동 공격**
 - 수동으로 취약점을 더 깊이 분석하고 익스플로잇.
- **Fuzzer**
 - **Fuzzing**을 통해 입력값을 조작하여 애플리케이션의 반응 테스트.
- **API 테스트**
 - REST, SOAP API와 같은 **API 보안 테스트** 기능





OWASP ZAP의 활용 사례

- **개발 단계의 보안 테스트**
 - 개발 주기 초기에 보안 취약점을 식별하여 수정 비용 절감.
- **CI/CD 파이프라인 통합**
 - Jenkins, GitHub Actions와 같은 자동화된 빌드 프로세스에 ZAP을 통합하여 지속적인 보안 테스트.
- **DevSecOps 환경**
 - DevSecOps 철학을 실천하는 데 필수적인 도구로 활용 가능.
- **교육 및 트레이닝**
 - 보안 팀이나 개발자를 대상으로 모의 해킹 실습 진행





OWASP ZAP 설치

- <https://www.zaproxy.org/>

❑ OWASP ZAP 설치

❑ 자바 JDK 11 이상

https://drive.google.com/file/d/1Fy7SXBz5MQCzMljl4uiHYF40pMxZEz8M/view?usp=drive_link





인스톨러 다운로드



Zed Attack Proxy (ZAP)

by Checkmarx

The world's most widely used web app scanner source. A community based GitHub Top 1000 project can contribute to.

Quick Start Guide

Download Now



Download ZAP

- Checksums for all of the ZAP downloads are maintained on the [2.15.0 Release Page](#) and in the relevant [version files](#).
- As with all software we strongly recommend that ZAP is only installed and used on operating systems and JREs that are fully patched and actively maintained.

ZAP 2.15.0

Windows (64) Installer

228 MB

Download

Windows (32) Installer

229 MB

Download

Linux Installer

224 MB

Download

Linux Package

221 MB

Download

macOS (Intel - amd64) Installer

250 MB

Download

macOS (Apple Silicon - aarch64) Installer

248 MB

Download

Cross Platform Package

261 MB

Download





만약 JDK가 없다고 뜨면

Locate> Program Files > Java > jdk-11 > bin



[Blog](#) [Videos](#) [Documentation](#) [Community](#) [Search](#)

[Download](#)



Download ZAP

- Checksums for all of the ZAP downloads are maintained on the [2.15.0 Release Page](#) and in the relevant [version files](#).
- As with all software we strongly recommend that ZAP is only installed and used on operating systems and JREs that are fully patched and actively maintained.

ZAP 2.15.0

[Windows \(64\) Installer](#)

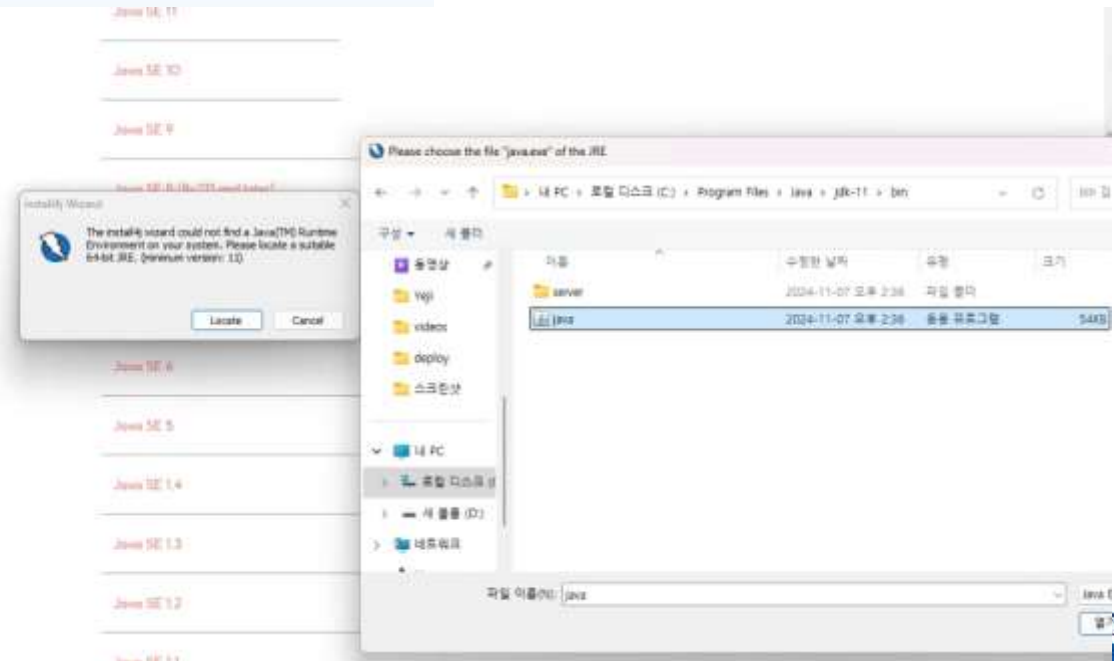
[Windows \(32\) Installer](#)

[Linux Installer](#)

[Linux Package](#)

[macOS \(Intel - amd64\) Installer](#)

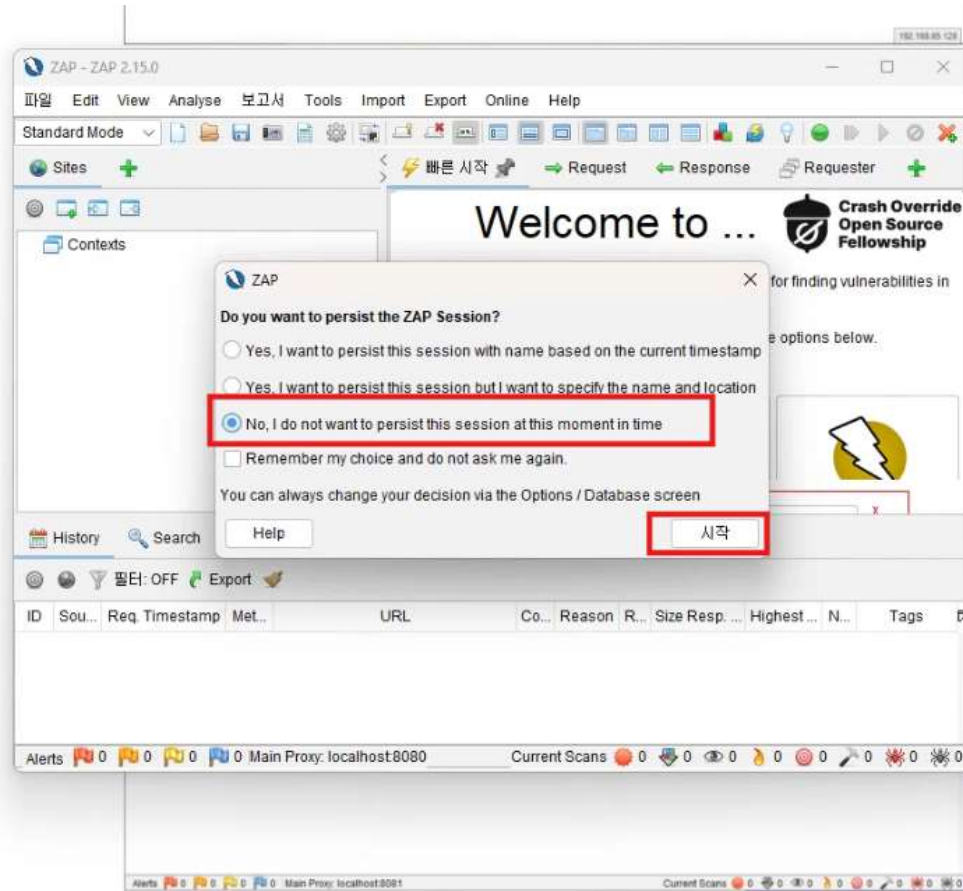
[macOS \(Apple Silicon - aarch64\) Installer](#)



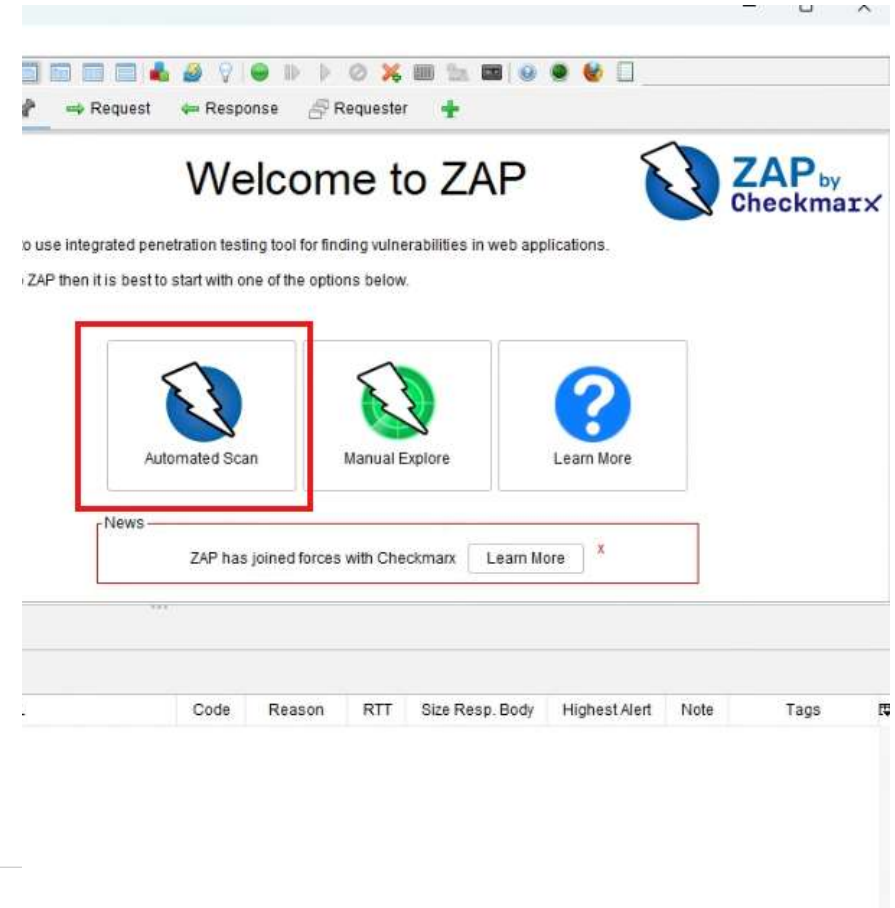


실행하기

- 3번째 no 항목 누르고 Automated scan 실행



- 화면에 Automated Scan 선택 및 url 주소를 입력하면 자동으로 취약점 점검이 실행가능함.





주소 입력하기

- 현재 돌아가고 있는 OWASP JUICE 주소 입력 > 공격
- 만약 클라우드 쓰고 있다면 <https://juice-shop.herokuapp.com/#/>

Untitled Session - ZAP 2.15.0

파일 Edit View Analyse 보고서 Tools Import Export Online Help

Standard Mode

Sites +

Contexts

Default Context

Sites

Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically given permission to test.

URL to attack: 선택...

Use traditional spider: ☒

Use ajax spider: ☐ If Modern with Chrome Headless

진행: Actively scanning (attacking) the URLs discovered by the spider(s)

History Search 경고 Output Spider AJAX Spider Active Scan WebSockets

New Scan 진행: 1: http://127.0.0.1:3000 12% 현재 검색: 1 Num Requests: 969 New Alerts: 0 Export

Sent Messages Filtered Messages

ID	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
4,301	24. 11. 7. 오후 2:46:02	24. 11. 7. 오후 2:46:02	POST	http://127.0.0.1:3000/latest/assets/public/asset...	200	OK	13 ms	466 bytes	3,748 bytes
4,302	24. 11. 7. 오후 2:46:02	24. 11. 7. 오후 2:46:02	POST	http://127.0.0.1:3000/latest/assets/public/asset...	200	OK	13 ms	466 bytes	3,748 bytes
4,303	24. 11. 7. 오후 2:46:02	24. 11. 7. 오후 2:46:02	GET	http://127.0.0.1:3000/latest/assets/public/asset...	200	OK	14 ms	466 bytes	3,748 bytes
4,304	24. 11. 7. 오후 2:46:02	24. 11. 7. 오후 2:46:02	POST	http://127.0.0.1:3000/latest/assets/public/asset...	200	OK	16 ms	466 bytes	3,748 bytes
4,305	24. 11. 7. 오후 2:46:02	24. 11. 7. 오후 2:46:02	POST	http://127.0.0.1:3000/latest/assets/public/asset...	200	OK	16 ms	466 bytes	3,748 bytes
4,306	24. 11. 7. 오후 2:46:02	24. 11. 7. 오후 2:46:02	GET	http://127.0.0.1:3000/latest/assets/public/asset...	200	OK	16 ms	466 bytes	3,748 bytes
4,307	24. 11. 7. 오후 2:46:02	24. 11. 7. 오후 2:46:02	POST	http://127.0.0.1:3000/latest/assets/public/asset...	200	OK	13 ms	466 bytes	3,748 bytes





취약점과 설명 출력

영어 한국어 일본어

클라우드 메타데이터 공격은 잘못 구성된 NGINX 서버를 악용하여 AWS, GCP, Azure와 같은 클라우드 서비스 공급자가 유지 관리하는 인스턴스 메타데이터에 액세스하려고 시도합니다.

keullaudeu metadeiteo gong-gyeong-eun jalmos guseongdoen NGINX seobeoleul ag-yonghayeo AWS, GCP, Azurewa gat-eun keullaudeu seobiseu gong-geubjaga yuji gwanlihaneun inseuteonseu

[자세히](#)

저장된 번역

파일 Edit View Analyse 보고서 Tools Import Export Online Help

Standard Mode

Sites

Contexts

Default Context

Sites

Cloud Metadata Potentially Exposed

JURL: `http://127.0.0.1:3000/latest/meta-data/?EIO=4&transport=polling&t=C5A4FO&sid=A2RrY2k78E0sKuM-AAB8`

위험: High

신뢰도: Low

매개 변수:

공격: 169.254.169.254

증거:

CWE ID: 0

WASC ID: 0

설명:

The Cloud Metadata Attack attempts to abuse a misconfigured NGINX server in order to access the instance metadata maintained by cloud service providers such as AWS, GCP and Azure.

기타 정보:

Based on the successful response status code cloud metadata may have been returned in the response. Check the response data to see if any cloud metadata has been returned.

해결:

Do not trust any user data in NGINX configs. In this case it is probably the use of the \$host variable which is set from the 'Host' header and can be controlled by an attacker.

참조:

<https://www.nginx.com/blog/trust-no-one-perils-of-trusting-user-input/>

Alert Tags:

Alerts 2 6 4 4 Main Proxy: localhost:8089





The screenshot shows the ZAP 2.15.0 interface. On the left, a list of alerts is displayed, with 'SQL Injection - SQLite' highlighted. The main panel on the right shows the details of this alert. The alert is titled 'SQL Injection - SQLite' and the description states 'SQL injection may be possible.' The alert details include WASC ID: 19, Source: 액티브, Alert Reference, Input Vector: URL Query String, and a description in Korean. The alert is highlighted with a red box. The background shows the ZAP interface with a list of alerts on the left and a detailed view of the selected alert on the right.





보고서 생성

- 위치 잘 기억하기

The screenshot displays the ZAP 2.15.0 interface. The 'Generate Report' dialog box is open, showing the following details:

- Report Title:** ZAP by Checkmarx Scanning Report
- Report Name:** 2024-11-07-ZAP-Report-.html
- Report Directory:** C:\Users\6\Documents (highlighted with a red box)
- Contexts:** Default Context
- Sites:** https://cdnjs.cloudflare.com, http://127.0.0.1:3000
- Generate If No Alerts:** ☐
- Display Report:** ☐
- Generate Report button:** (highlighted with a red box)

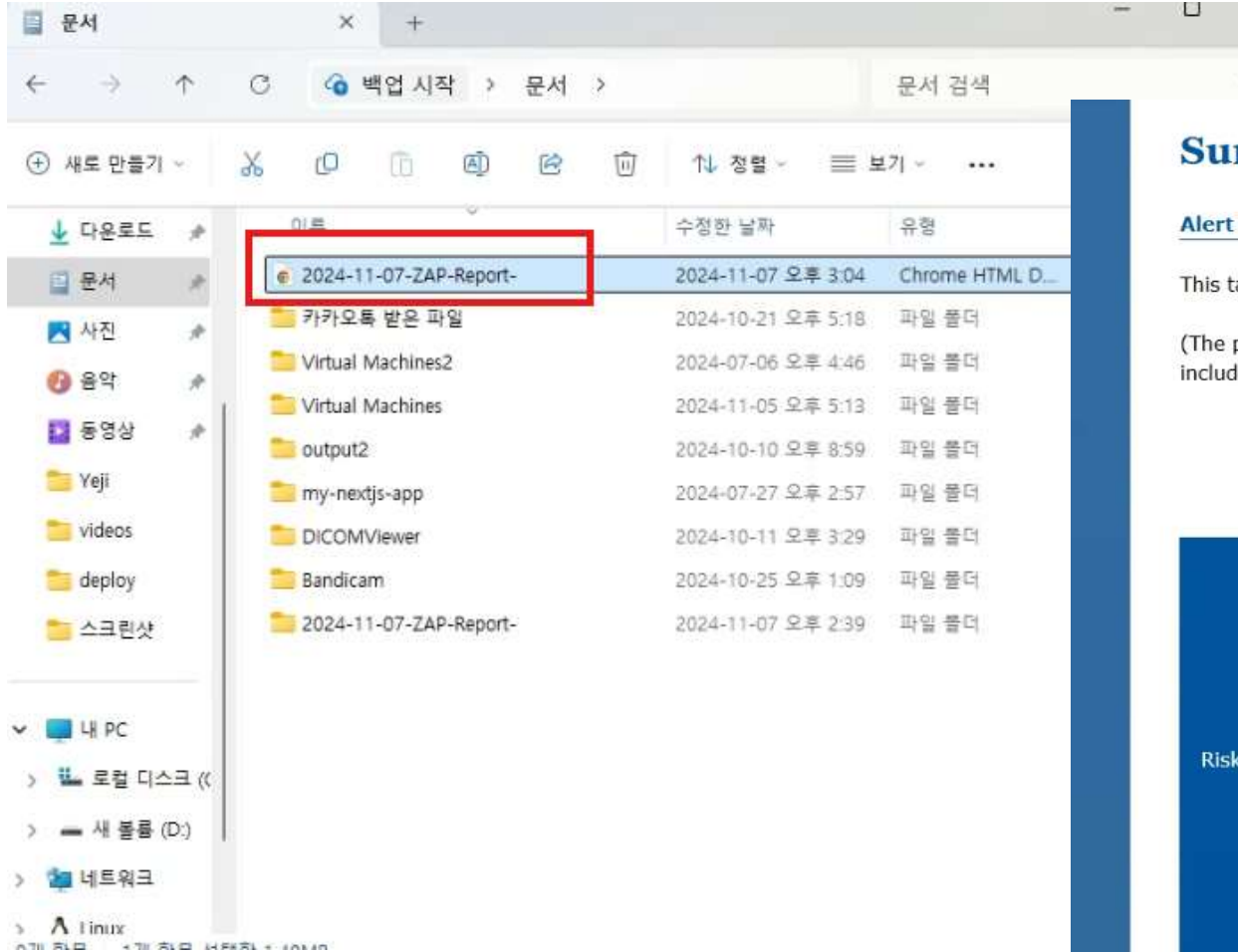
The background shows a web session with a 500 Internal Server Error. The error message is: "SQLite_ERROR: near \"(\": syntax error\"". The SQL query is: "SELECT * FROM Products WHERE ((name LIKE '%(' OR description LIKE '%(') ORDER BY name".

The bottom panel shows a list of alerts, including "Cloud Metadata Potentially Exposed (2)", "SQL Injection - SQLite", "CSP: Wildcard Directive (2)", "Content Security Policy (CSP) Header Not Set (185)", "Cross-Domain Misconfiguration (168)", and "Missing Anti-clickjacking Header (60)".





보고서 확인



Summaries

Alert counts by risk and confidence

This table shows the number of alerts for each level of risk and confidence included in the report.

(The percentages in brackets represent the count as a percentage of the total number of alerts included in the report, rounded to one decimal place.)

		Confidence				
		User Confirmed	High	Medium	Low	Total
Risk	High	0 (0.0%)	0 (0.0%)	1 (6.2%)	1 (6.2%)	2 (12.5%)
	Medium	0 (0.0%)	3 (18.8%)	3 (18.8%)	0 (0.0%)	6 (37.5%)
	Low	0 (0.0%)	0 (0.0%)	3 (18.8%)	1 (6.2%)	4 (25.0%)
	Informational	0 (0.0%)	0 (0.0%)	3 (18.8%)	1 (6.2%)	4 (25.0%)
	Total	0 (0.0%)	3 (18.8%)	10 (62.5%)	3 (18.8%)	16 (100%)





보고서 상세 분석

■ CTRL + 클릭 으로 링크 들어가보기

Alert counts by alert type

This table shows the number of alerts of each alert type, together with the alert type's risk level.

(The percentages in brackets represent each count as a percentage, rounded to one decimal place, of the total number of alerts included in this report.)

Alert type	Risk	Count
Cloud Metadata Potentially Exposed	High	2 (12.5%)
SQL Injection - SQLite	High	1 (6.2%)
CSP: Wildcard Directive	Medium	2 (12.5%)
Content Security Policy (CSP) Header Not Set	Medium	185 (1,156.2%)
Cross-Domain Misconfiguration	Medium	168 (1,050.0%)
Missing Anti-clickjacking Header	Medium	60 (375.0%)
Session ID in URL Rewrite	Medium	198 (1,237.5%)
Vulnerable JS Library	Medium	1 (6.2%)

SQL Injection - SQLite

Source	raised by an active scanner (plugin ID: -1)
CWE ID	89
WASC ID	19
Reference	<ul style="list-style-type: none">https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

CSP: Wildcard Directive

Source	raised by a passive scanner (CSP)
CWE ID	693
WASC ID	15
Reference	<ul style="list-style-type: none">https://www.w3.org/TR/CSP/https://caniuse.com/#search=content+security+policyhttps://content-security-policy.com/https://github.com/HtmlUnit/htmlunit-csphttps://developers.google.com/web/fundamentals/security/csp#policy_applies_to_a_wide_variety_of_resources





치트시트 확인하기

SQL Injection Prevention Cheat Sheet

Introduction

This cheat sheet will help you prevent SQL injection flaws in your applications. It will define what SQL injection is, explain where those flaws occur, and provide four options for defending against SQL injection attacks. [SQL Injection](#) attacks are common because:

1. SQL Injection vulnerabilities are very common, and
2. The application's database is a frequent target for attackers because it typically contains interesting/critical data.

What Is a SQL Injection Attack?

Attackers can use SQL injection on an application if it has dynamic database queries that use string concatenation and user supplied input. To avoid SQL injection flaws, developers need to:

1. Stop writing dynamic queries with string concatenation or
2. Prevent malicious SQL input from being included in executed queries.

There are simple techniques for preventing SQL injection vulnerabilities and they can be used with practically any kind of programming language and any type of database. While XML databases can have similar problems (e.g., XPath and XQuery injection), these techniques can be used to protect them as well.

Anatomy of A Typical SQL Injection Vulnerability

Table of contents

Introduction

What Is a SQL Injection Attack?

Anatomy of A Typical SQL Injection Vulnerability

Primary Defenses

Defense Option 1: Prepared Statements (with Parameterized Queries)

Safe Java Prepared Statement Example

Safe C# .NET Prepared Statement Example

Hibernate Query Language (HQL) Prepared Statement (Named Parameters) Example

Other Examples of Safe Prepared Statements

Defense Option 2: Stored Procedures

Safe Approach to Stored Procedures

When Stored Procedures Can Increase Risk

Safe Java Stored Procedure Example

Safe VB .NET Stored Procedure Example

Defense Option 3: Allow-list Input Validation



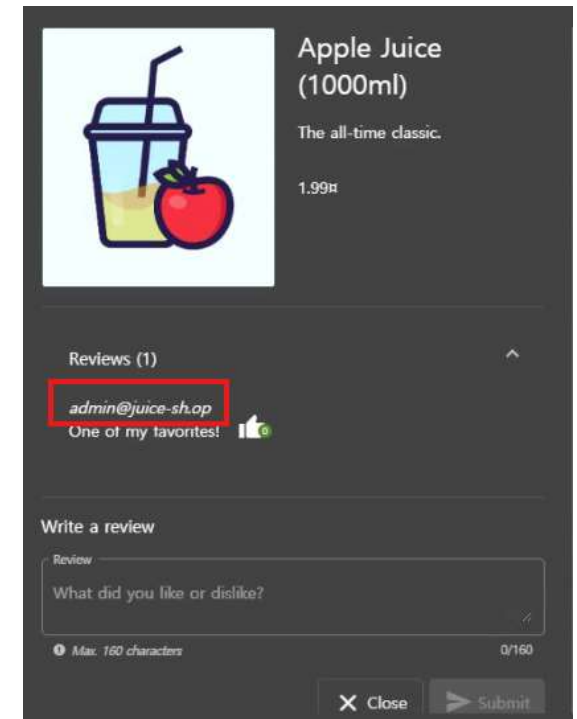
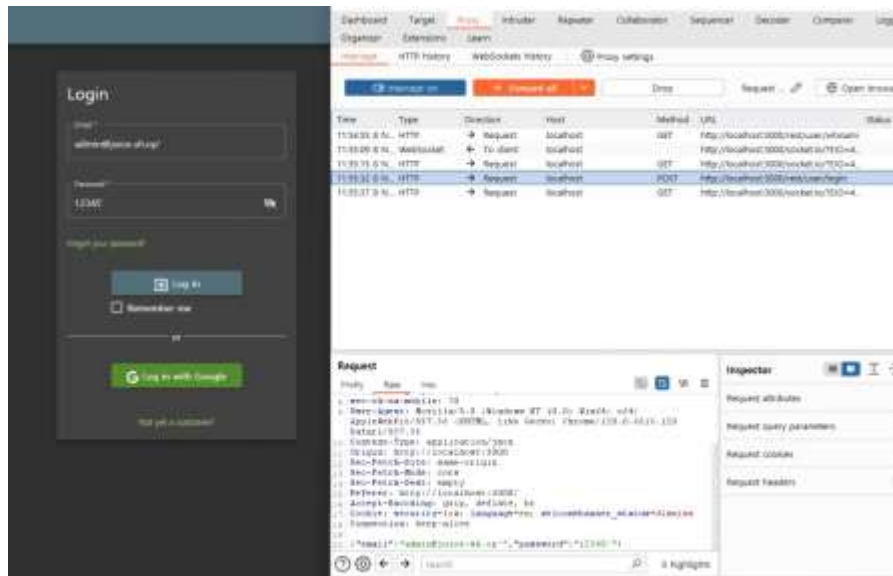
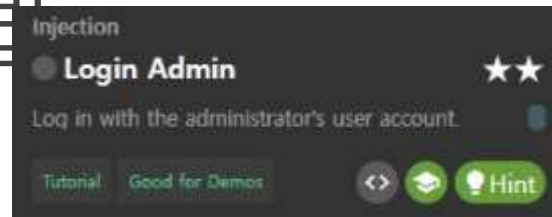


OWASP JUICE

INJECTION

- 관리자로 로그인하기

인젝션(삽입)은 데이터 입력이 가능한 장소를 찾아 데이터 입력시 악의적인 데이터를 삽입해 타겟의 인터프리터로 전송하는 공격 방법

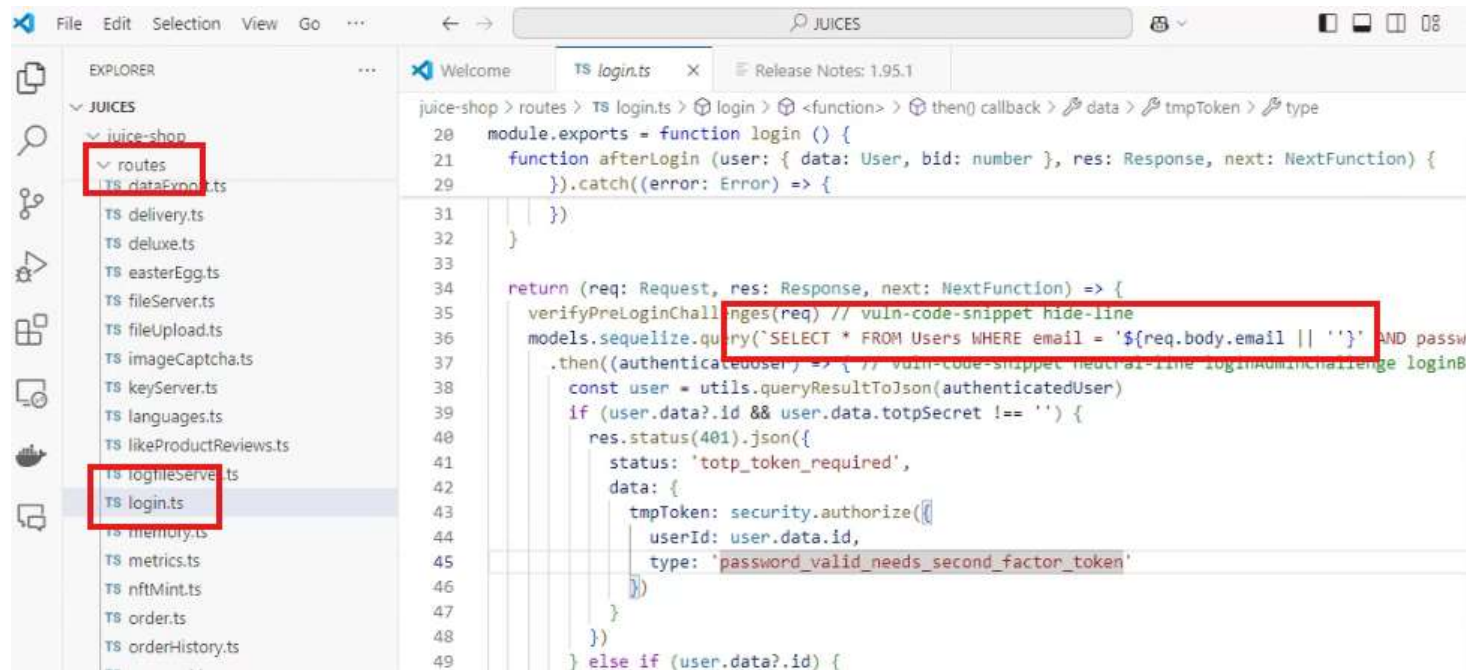




인젝션 해보기

- admin@juice-sh.op' or 1=1 -- -

❑ 취약 소스 위치



```
juice-shop > routes > TS login.ts > login > <function> > then() callback > data > tmpToken > type
20 module.exports = function login () {
21   function afterLogin (user: { data: User, bid: number }, res: Response, next: NextFunction) {
29     }).catch((error: Error) => {
31   })
32 }
33
34 return (req: Request, res: Response, next: NextFunction) => {
35   verifyPreLoginChallenges(req) // vuln-code-snippet hide-line
36   models.sequelize.query('SELECT * FROM Users WHERE email = '${req.body.email} || ''' AND password = '${req.body.password}'
37   .then((authenticatedUser) => { // vuln-code-snippet neutral-line loginAdminChallenge loginB
38     const user = utils.queryResultToJson(authenticatedUser)
39     if (user.data?.id && user.data.totpSecret !== '') {
40       res.status(401).json({
41         status: 'totp_token_required',
42         data: {
43           tmpToken: security.authorize({
44             userId: user.data.id,
45             type: 'password_valid_needs_second_factor_token'
46           })
47         }
48       })
49     } else if (user.data?.id) {
```





보고서 작성

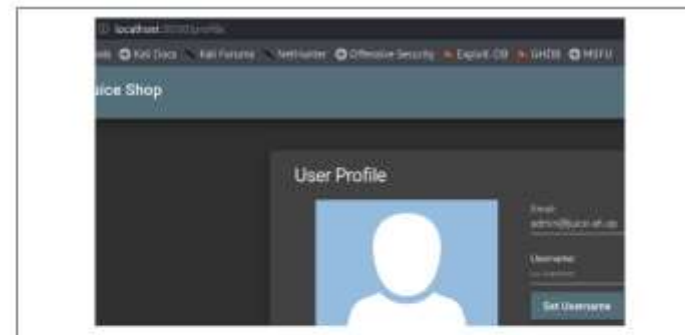
■ SQL Injection 취약점 보고서

항목	내용
취약점 ID	
취약점명	SQL Injection
검출 위치	http://localhost:3000/rest/user/login
위험도	높음 (CVSS v3.1: 9.4)
CVSS 상세	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:L
진단 날짜	[날짜 입력]

#choi1 - SQL 인젝션		CVSS
리스크	높음	9.4
영향력	높음	
가능성	높음	
CVSS 스코어	CVSS 3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:L	
위치	http://localhost:3000/rest/user/login	

설명
로그인을 담당하는 REST API에서 SQL 인젝션이 발견되었습니다. SQL 인젝션은 프로그램의 SQL 쿼리문을 악의적으로 조작하여 비정상적인 쿼리문이 실행되도록 만드는 공격입니다. 이 API 엔드포인트의 경우 POST 파라미터 중 하나인 email 파라미터에서 SQL 인젝션이 가능합니다.
공격자들은 SQL 인젝션을 통해 주스샵의 관리자 로그인 하거나, 존재하지 않는 계정으로 로그인을 할 수 있게 됩니다. 이후 관리자 계정으로 주스샵 어플리케이션을 더 조작, 변경 할 수 있기 때문에 주의를 요합니다.
해당 취약점은 공격의 난이도가 낮아 일어날 가능성이 높습니다. 주스샵 관리자 로그인 가능하기 때문에 영향력이 높습니다. 따라서 이 취약점의 비즈니스 리스크는 "높음"으로 설정되었습니다.

취약점 공격 재현
1. 사용자 인증을 하지 않은 채 로그인 기능을 누릅니다.
2. 다음의 이메일과 비밀번호를 사용해 로그인을 합니다.
이메일: <code>admin@juice-shop</code> or <code>1=1</code> --
비밀번호: <code>asdf</code>
3. 성공적으로 관리자 계정으로 로그인 한 것을 확인합니다.



해결 방안
1. Object Relational Mapping 라이브러리를 사용하지 않고 직접적인 SQL 쿼리문을 사용하고 있다면 ORM 라이브러리 사용을 추천합니다.
2. 직접적인 SQL 쿼리문을 꼭 사용해야한다면 사용자 입력 검증 및 필터링, 특수문자 이스케이핑을 추천드립니다.
3. NodeJS Express 에서 SQLite 로 데이터를 주고 받을때는 직접적인 SQL 쿼리문 보단 Prepared-Statement (준비된 선언) 을 사용하는 것을 추천드립니다.





- CVSS(Common Vulnerability Scoring System)는 보안 취약점의 심각도를 표준화된 방식으로 수치화
- 취약점의 주요 특성을 기반으로 0.0에서 10.0까지의 점수를 산출

□ CVSS 점수 해석

- 심각(Critical): 9.0-10.0
- 높음(High): 7.0-8.9
- 중간(Medium): 4.0-6.9
- 낮음(Low): 0.1-3.9
- 없음(None): 0.0

메트릭	값	설명	선정 사유
공격 벡터(AV)	Network	원격 공격 가능	REST API 엔드포인트가 네트워크를 통해 접근 가능
공격 복잡도(AC)	Low	특별한 조건 불필요	단순한 SQL 인젝션 페이로드로 공격 가능
필요 권한(PR)	None	사전 권한 불필요	로그인 전에 실행 가능한 공격
사용자 상호작용(UI)	None	사용자 개입 불필요	자동화된 공격 가능
영향 범위(S)	Unchanged	단일 보안 범위	취약점이 다른 시스템으로 전파되지 않음
기밀성(C)	High	모든 데이터 접근 가능	데이터베이스 전체 접근 가능
무결성(I)	High	데이터 완전 변조 가능	데이터베이스 수정 가능
가용성(A)	Low	부분적 서비스 영향	일부 기능 장애 가능





CVSS 계산기 사용법

- <https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator>
- ❑ NVD CVSS 계산기 방문: <https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator>
- ❑ 각 메트릭 그룹의 값을 선택
- ❑ 자동으로 계산된 점수 확인
- ❑ 벡터 문자열 복사하여 보고서에 포함





취약점 보고서 작성하기

■ 취약점 상세 정보

구분	설명
취약점 개요	REST API 로그인 기능에서 SQL Injection 취약점이 발견됨. 사용자 입력값이 적절한 검증 없이 SQL 쿼리에 직접 삽입되어 발생
공격 시나리오	1. 로그인 API 엔드포인트 접근 2. 악의적인 SQL 구문 주입 3. 인증 우회 및 관리자 권한 획득
공격 예시	이메일: admin@juice-sh.op' or '1'='1' -- 비밀번호: asdf
영향도	- 인증 우회 가능 - 데이터베이스 무단 접근- 민감 정보 유출- 데이터 무결성 침해





취약점 보고서 작성하기

■ 취약점 검증

구분	내용
재현 환경	- 운영체제: [OS 버전]- 브라우저: [브라우저 정보]- 테스트 도구: [도구명]
테스트 방법	1. POST 요청으로 로그인 시도2. SQL Injection 페이지 로드 전송3. 응답 결과 확인
취약한 코드	sql SELECT * FROM users WHERE email = '[사용자_입력]' AND password = '[비밀번호]'





개선 방안

우선순위	개선 사항	적용 방안	기대 효과
상	Prepared Statement 적용	javascript const stmt = db.prepare('SELECT * FROM users WHERE email = ? AND password = ?'); const result = stmt.get(email, password);	SQL Injection 원천 차단
상	입력값 검증	- 이메일 형식 검증- 특수 문자 필터링- 길이 제한 적용	악의적 입력 차단
중	매개변수화된 쿼리 사용	ORM 프레임워크 도입	안전한 쿼리 실행 보장
중	에러 처리 보완	- 상세 에러 메시지 숨김- 에러 로깅 강화	정보 노출 최소화
하	보안 로깅 강화	- 로그인 시도 기록- 실패 횟수 제한	이상 징후 탐지



개인 발표

팀 프로젝트

팀프로젝트 진행 및 재정의

공지

행사 및 과제 안내



과제 안내

- OWASP JUICE SHOP & 팀 프로젝트

- JUICE SHOP 취약점 보고서 작성

- 이번 주는 코딩테스트 및 CTF 문제풀이 대신 OWASP JUICE SHOP 취약점 보고서 작성하기(2개이상, 38슬라이드 참고)
- 리뷰는 해주세요

- 팀 프로젝트 마감 임박

- 11주차(29일) 에 발표
- 10주차에 보고서 작성

- 별점 유의



Thank you

