OWASP Top 10 기반 3-Tier 웹서비스 보안 분석

공공 의료 민원 포털 취약점 진단 프로젝트 발표

4조 · 문현건 신동하 이건엽 이상우 조소현 한민서

프로젝트 아키텍처 (3-Tier)

Web Tier

Nginx
Reverse Proxy

WAS Tier

Flask + SQLAlchemy
인증/민원/게시판/마이데이터

DB Tier

MariaDB
업무 데이터 + 감사로그

Client → Nginx (80) → Flask (5000) → MariaDB (3306)

서비스 구축 시 사용 AI

Codex

UI
+
Backend

ClaudeCode

Backend
+
서비스 취약점 생성

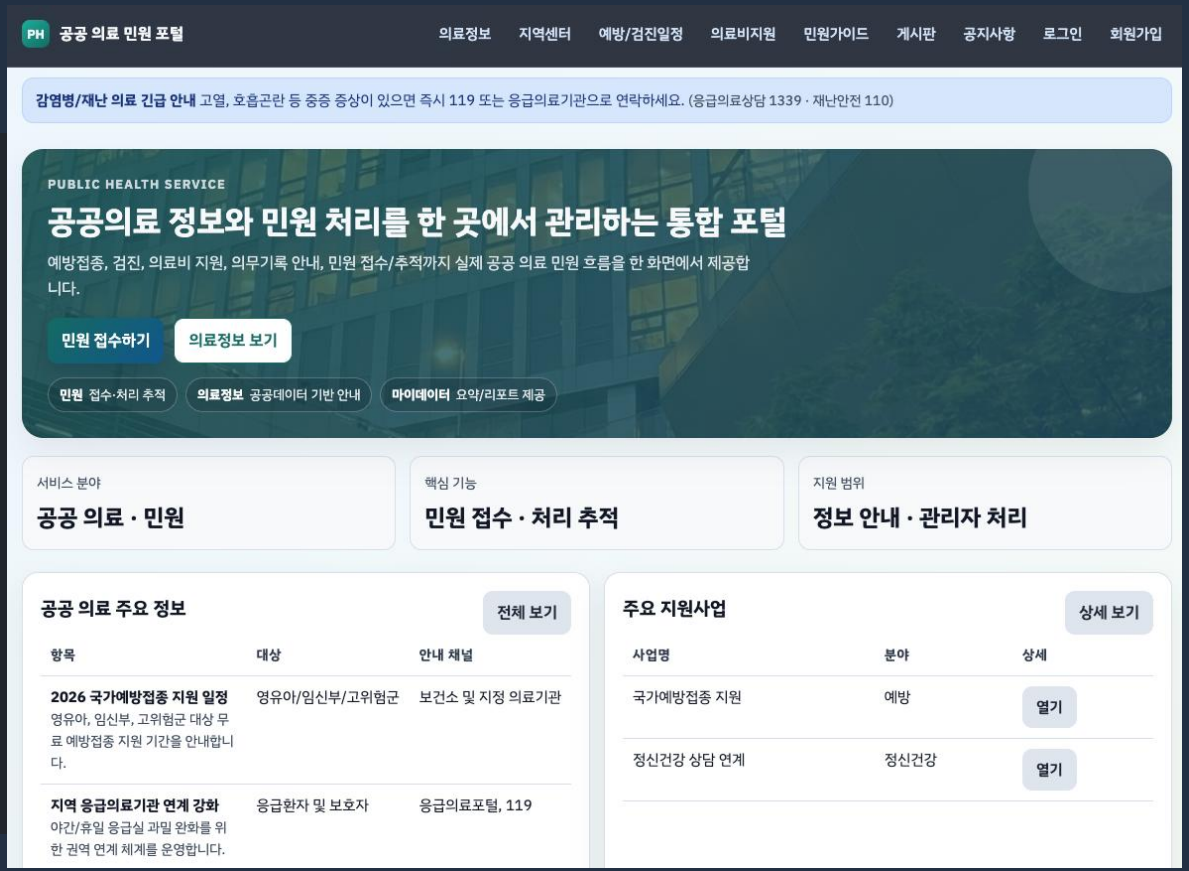
Gemini

Backend
+
취약점 검증

서비스 범위

사용자 기능

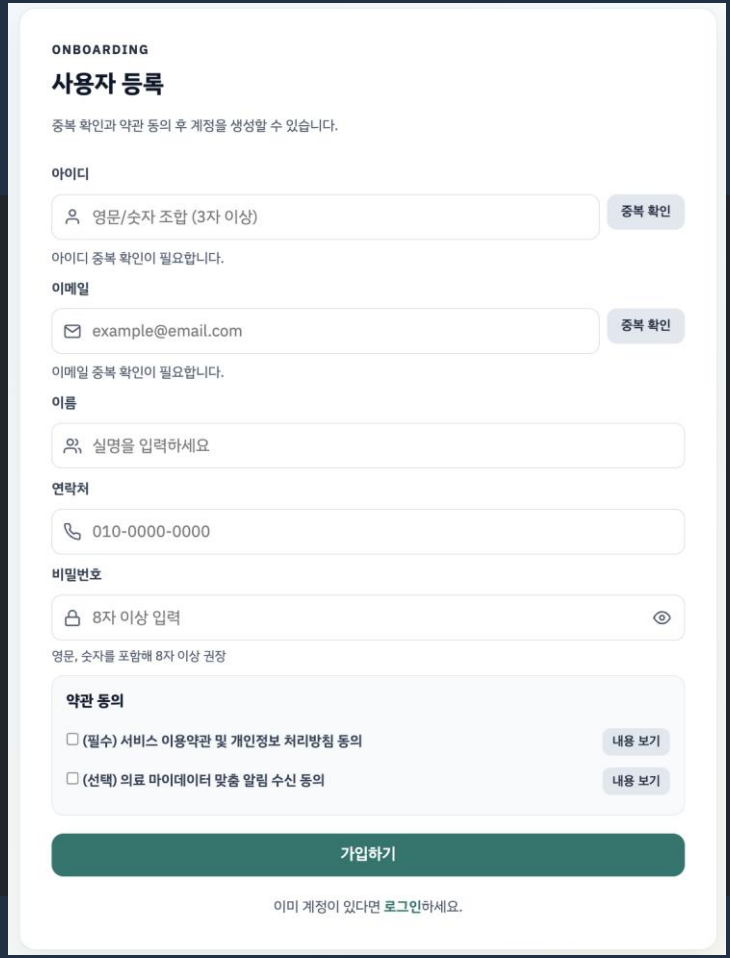
회원가입/로그인, 게시판,
민원, 공지, 마이페이지



서비스 범위

사용자 기능

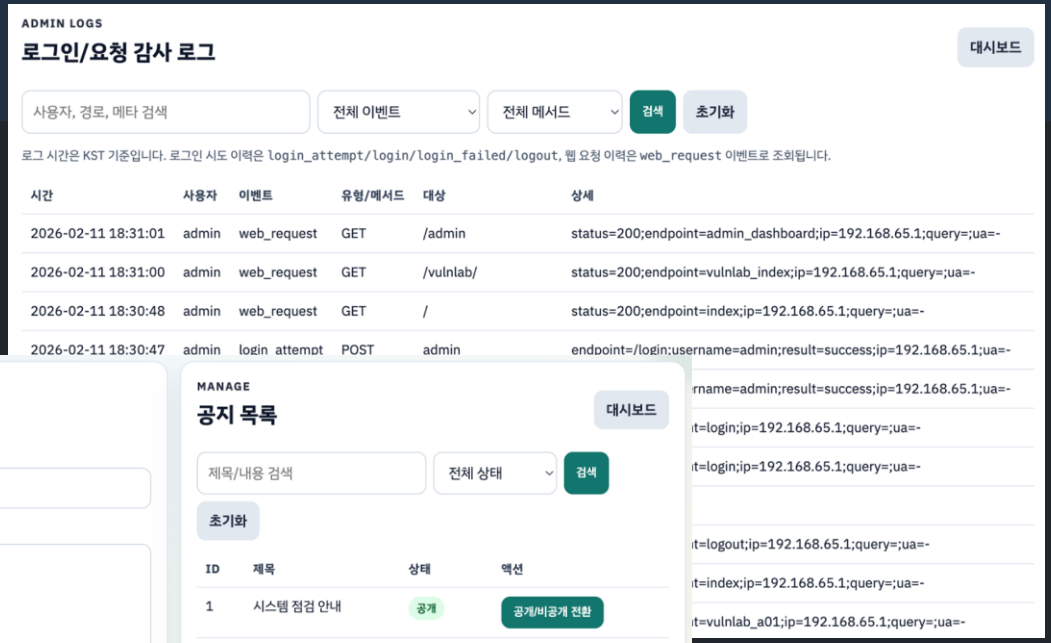
회원가입/로그인, 게시판,
민원, 공지, 마이페이지

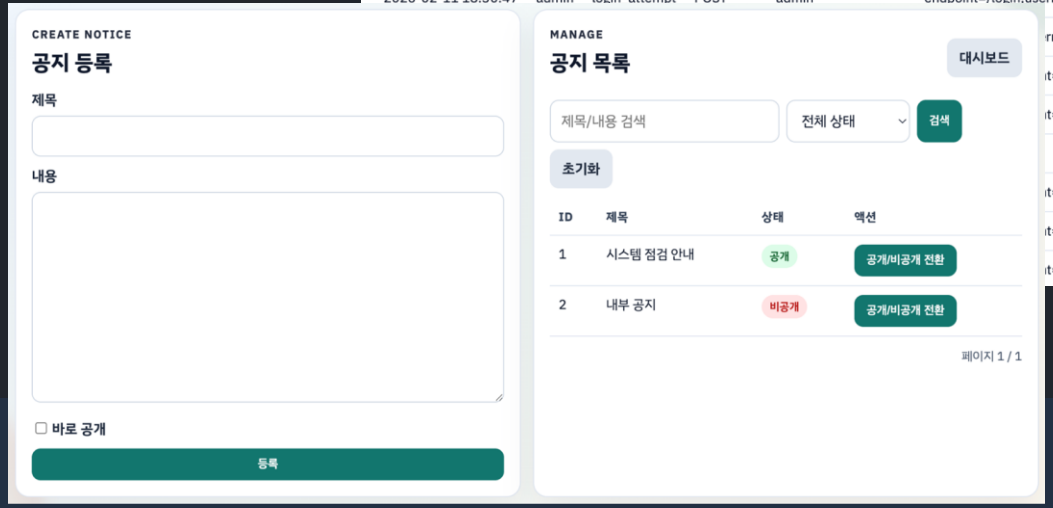


서비스 범위

관리자 기능

사용자·게시물·공지·민원 관리,
로그 모니터링

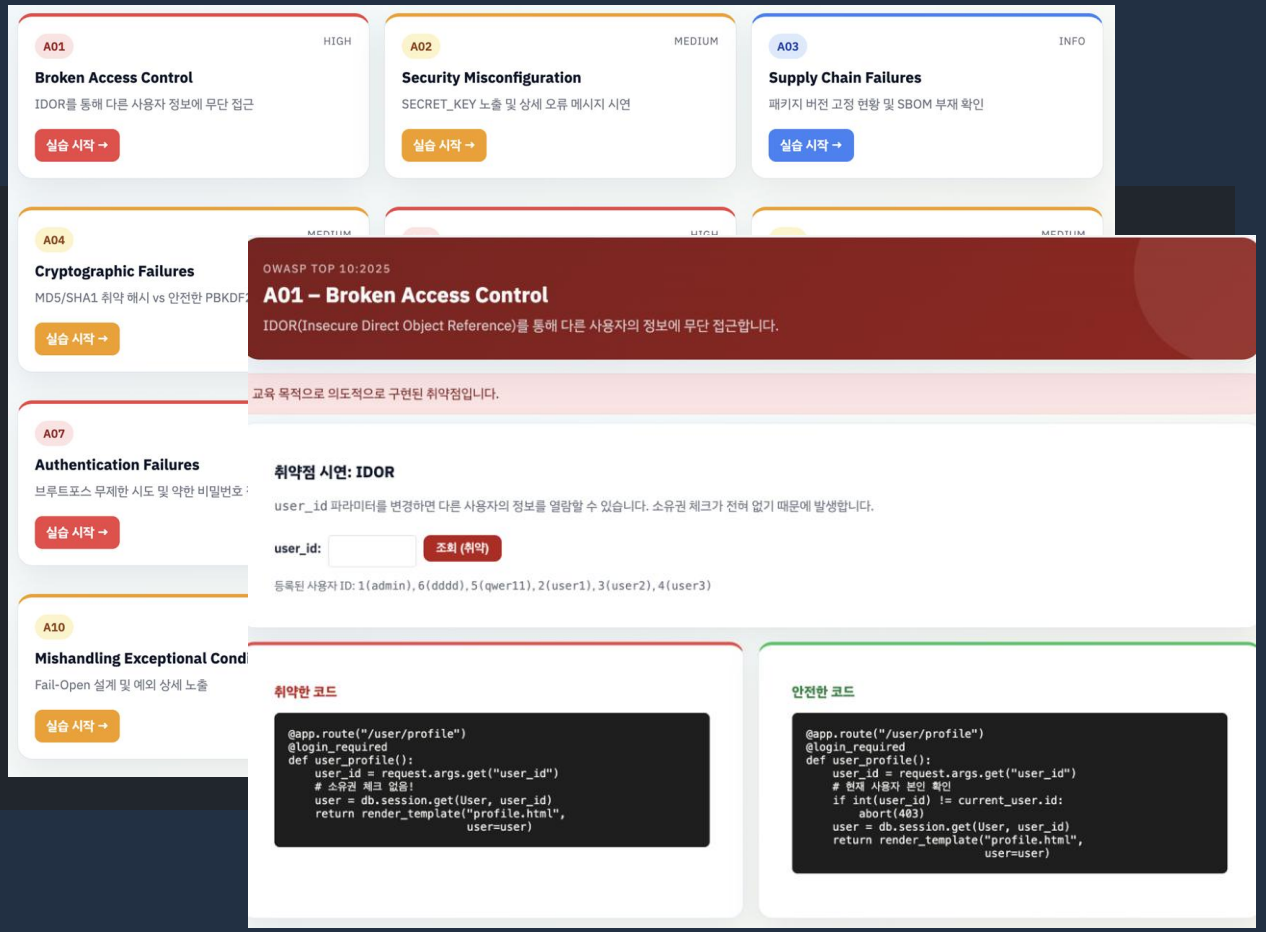




서비스 범위

취약점 실습

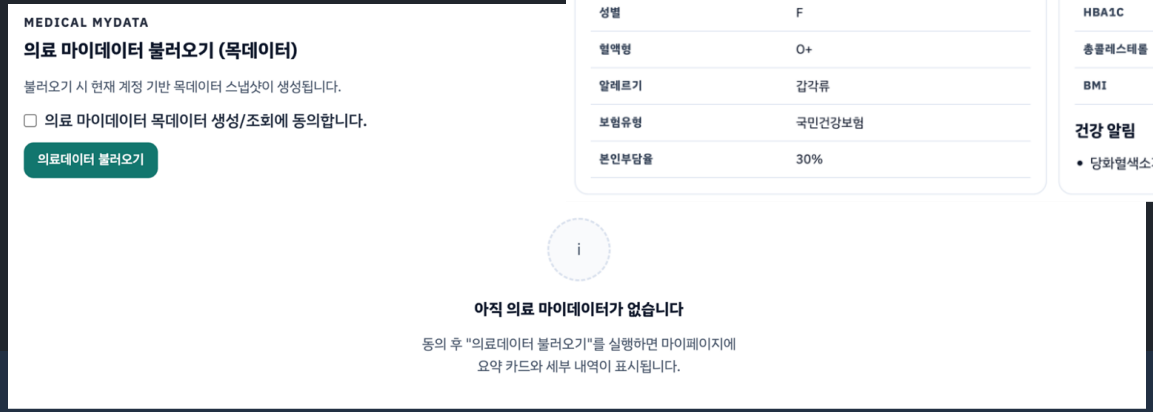
/vulnlab, /security/scenarios
교육용 보안 실습 환경



서비스 범위

의료 마이데이터

기관 API +DB 시드 데이터 기반 조회



OWASP Top 10:2025 커버리지

코드	항목	프로젝트 내 점검 포인트
A01	Broken Access Control	관리자/일반 사용자 경계, IDOR 차단
A02	Security Misconfiguration	기본 설정/오류 노출 점검
A03	Software Supply Chain	의존성 버전/패키지 검증
A04	Cryptographic Failures	민감정보 전송/저장 보호
A05	Injection	입력 처리와 쿼리 안전성
A06	Insecure Design	업무 규칙/Rate limiting/상태 전이
A07	Authentication Failures	로그인/세션 정책
A08	Integrity Failures	데이터/코드 무결성
A09	Logging & Alerting Failures	감사로그/탐지/경보
A10	Exceptional Conditions	Fail-Open, 예외 처리

A01 취약점 개요

취약점 개요

명칭: 권한 상승을 통한 비인가 관리자 페이지 접근
관련 항목: A01: Broken Access Control
위험도: High (상)

영향

일반 사용자가 관리자 전용 경로인 /admin에 접근하여 시스템 설정 및 사용자 관리 기능을 임의로 조작할 수 있음.

A01 취약점 상세 설명

취약점 상세 설명

서버가 특정 경로(Path)에 대해 '인증(Authentication)' 여부만 확인하고, 해당 사용자가 해당 자원을 사용할 권한이 있는지 '인가(Authorization)' 과정을 누락하여 발생함. 본 시스템은 owasp1이라는 일반 사용자 세션을 보유한 상태에서도 관리자 페이지인 /admin 접근 시 이를 차단하지 않고 정상적인 응답을 반환함.

취약점 근거와 시나리오

OWASP 공식 홈페이지에 해당하는 주요 CWE로는 *CWE-200(권한 없는 사용자에게 민감한 정보 노출) 가 있다.*

시나리오 #2: 공격자가 브라우저를 특정 URL로 강제로 이동시킵니다. 관리자 페이지에 접근하려면 관리자 권한이 필요합니다.

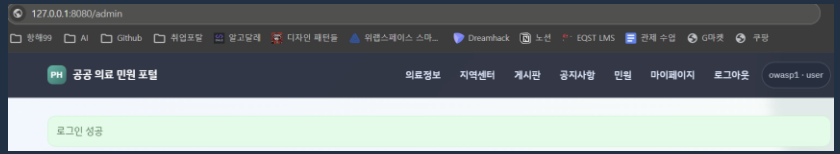
```
https://example.com/app/getappInfo
https://example.com/app/admin_getappInfo
```

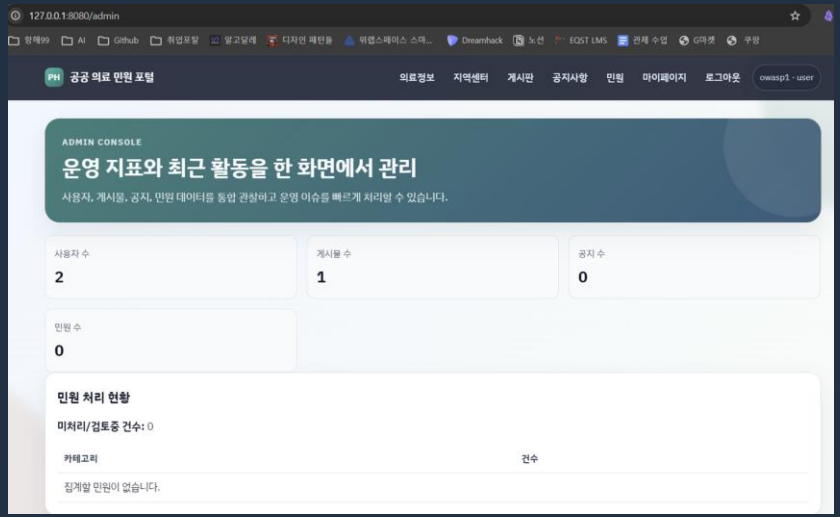
A01 취약점 재현 절차

Step 1
일반 사용자 계정(owasp1 / 11111111)으로 /login에 접속하여 세션 쿠키를 획득함.

Step 2
획득한 세션을 유지한 채 http://127.0.0.1:8090/admin에 직접 접속을 시도함.

Step 3
서버가 403 Forbidden 에러를 뱉지 않고 200 OK 응답과 함께 관리자 페이지를 렌더링함을 확인함.





A01 대응 방안

01 서버 측 인가 로직 구현

모든 관리자 전용 라우터(Router) 상단에 세션의 권한 정보(role == 'admin')를 검증하는 미들웨어를 배치해야 함.

02 최소 권한의 원칙

사용자의 역할에 따라 접근 가능한 자원을 화이트리스트(Whitelist) 방식으로 엄격히 통제함.

A02 취약점 개요

취약점 개요

명칭: 보안 설정 오류
관련 항목: A02: Security Misconfiguration
위험도: Medium ~ High (중~상)

영향

잘못된 시스템 설정, 기본 계정 유지, 불필요 서비스 활성화 등으로 인해 공격자가 시스템 내부 정보 획득, 권한 상승, 원격 침투 등을 수행할 수 있음.

A02 취약점 상세 설명

취약점 상세 설명

운영 환경에서 제거되어야 할 개발 설정이 그대로 유지되었음.

기본 관리자 계정이 변경되지 않았고, SECRET_KEY가 하드코딩되어 노출되었음.

또한 DEBUG 모드와 상세 에러 메시지로 인해 내부 코드 및 서버 구조 정보가 외부에 노출되었음.

취약점 근거와 시나리오

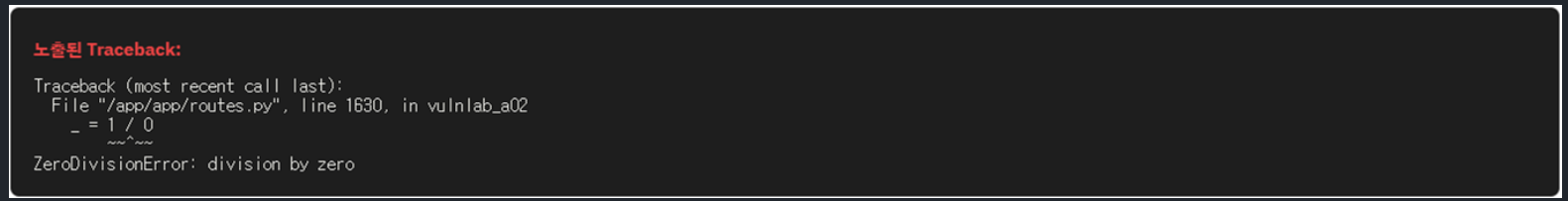
OWASP 공식 홈페이지에 해당하는 주요 CWE로는 CWE-16(*보안 설정 오류*), *CWE-200(에러 메시지를 통한 정보 노출)* 이 있다.

시나리오 #3: 애플리케이션 서버 구성에서 스택 트레이스와 같은 자세한 오류 메시지를 사용자에게 반환할 수 있습니다. 이로 인해 취약한 것으로 알려진 구성 요소 버전과 같은 민감한 정보 또는 근본적인 결함이 노출될 수 있습니다.

A02 취약점 재현 절차

Step 1
일반 사용자(user1 / user12345)로 로그인하여 취약 페이지(localhost:8090/vulnlab/a02)에 접근함.

Step 2
주소창에 ?trigger_error=1을 추가하면 내부 코드 구조 및 경로 정보가 노출됨.



A02 대응 방안

01 기본 설정 제거

기본 계정, 기본 비밀번호, 기본 API 키 등은 운영 환경 배포 전 반드시 변경하거나 제거해야 함.

02 보안 설정 외부 분리

SECRET_KEY, DB 계정, 토큰 등 민감 정보는 소스코드가 아닌 환경변수 또는 Secret Manager에서 관리해야 함.

03 운영 환경 보안 설정 강화

운영 환경에서는 DEBUG 비활성화, 불필요 기능 및 테스트 페이지 제거, 샘플 계정 삭제를 적용해야 함.

A03 취약점 개요

취약점 개요

명칭: CVE-2025-47278
관련 항목: A03: Software Supply Chain Failures
위험도: High (상)

영향

해당 취약점은 원격 코드 실행(RCE)이나 민감 정보 유출로 이어질 수 있는 고위험군 결함으로, 공격자가 신뢰되지 않은 의존성 경로를 통해 시스템 전체의 무결성을 파괴할 수 있음.

A03 취약점 상세 설명

취약점 상세 설명

프로젝트 내 was/requirements.txt 파일을 살펴보면 모든 주요 라이브러리에 대해 == 연산자(버전 고정)를 사용됨.
현재 고정된 Flask 3.1.0 버전은 CVE-2025-47278과 같은 Critical 수준의 취약점이 발견됨.
하지만 pip-audit 또는 safet와 같은 의존성 취약점 도구를 CI/CD 파이프라인과 통합하지 않음.
이러한 검증 프로세스의 부재로 인해 시스템이 지속적으로 위험에 노출됨(버전 고정의 역설).

취약점 근거

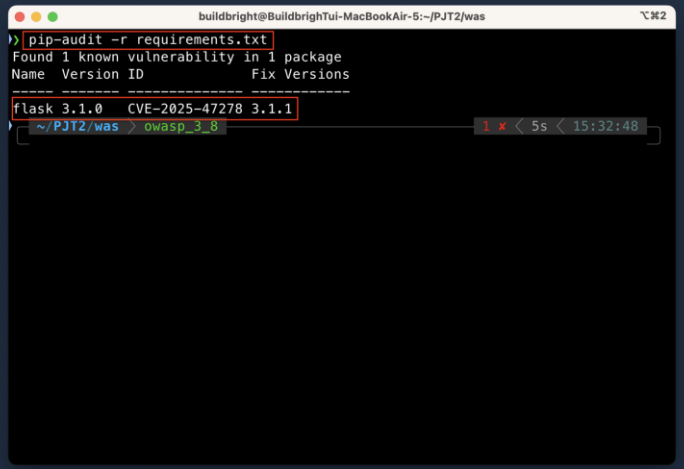
OWASP 공식 홈페이지에 해당하는 주요 CWE로는 *CWE-1395(Dependency on Vulnerable Third-Party Component) 가 있다*
.

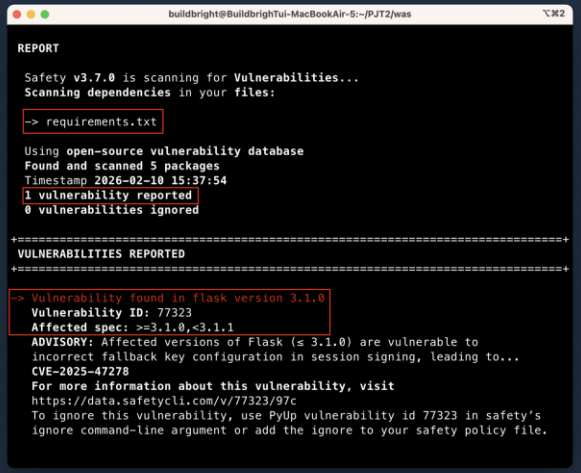
Impact	Details
Varies by Context	**Scope: Confidentiality, Integrity, Availability** The consequences vary widely, depending on the vulnerabilities that exist in the component; how those vulnerabilities can be "reached" by adversaries, as the exploitation paths and attack surface will vary depending on how the component is used; and the criticality of the privilege levels and features for which the product relies on the component.

A03 취약점 진단 절차

Step 1
자동화된 오픈소스 보안 분석(SCA) 도구인 pip-audit 및 safet를 활용하여 프로젝트 내 requirements.txt에 명시된 패키지들의 알려진 보안 취약점(CVE) 유무를 전수 조사.

Step 2
두 도구를 교차 검증 결과 현재 사용 중인 Flask 3.1.0 버전에서 CVE-2025-47278 취약점이 공통적으로 식별됨





A03 대응 방안

01 의존성 보안 스캔 자동화 (CI/CD 통합)

Pip-audit 또는 safety를 Github Actions 등의 CI/CD 파이프라인에 통합하여,
새로운 코드가 푸시될 때마다 의존성 취약점을 자동으로 점검해야 함.

02 동적 패치 및 버전 관리 정책 수립

분기별로 모든 라이브러리를 최신 안정 버전으로 검토 및 업데이트하는 정기 프로세스를 가동해야 함.
또한 Critical 수준의 CVE 공표 시, 24시간 이내에 호환성 테스트 및 패치 적용을 완료하는 체계를 갖춰야 함.

03 신뢰할 수 있는 소수 제어 및 화이트리스트 운영

PyPI 등 공식적인 신뢰 소스에서만 패키지를 설치하도록 pip 설정(--index-url)을 강제해야 함.

A04 취약점 개요 및 상세 설명

취약점 개요

명칭: 전송 구간 암호화 미적용
관련 항목: A04: Cryptographic Failures
위험도: High (상)

영향

스니핑을 통한 주민등록번호, 진료 기록 등 민감 정보 유출.

취약점 상세 설명 및 시나리오

- **원인**: Nginx 서버에 SSL/TLS 미적용 (HTTP 80 포트 사용)
- **현상**: 사용자가 입력한 데이터가 암호화되지 않은 **평문(Plaintext)** 상태로 전송됨

시나리오 #1 : 웹사이트가 모든 페이지에 대해 TLS를 사용하거나 적용하지 않거나, 또는 취약한 암호화를 사용하는 경우입니다. 공격자는 네트워크 트래픽을 모니터링(예: 보안이 취약한 무선 네트워크)하고, 연결을 HTTPS에서 HTTP로 다운그레이드한 후, 요청을 가로채 사용자의 세션 쿠키를 탈취합니다. 그런 다음 공격자는 탈취한 쿠키를 사용하여 사용자의 (인증된) 세션을 탈취하고 사용자의 개인 데이터에 접근하거나 수정합니다. 위와 같은 경우 외에도, 전송되는 모든 데이터를 변경할 수도 있습니다. 예를 들어, 송금 수신자의 정보를 변경할 수 있습니다.

A04 취약점 재현 절차

Step 1

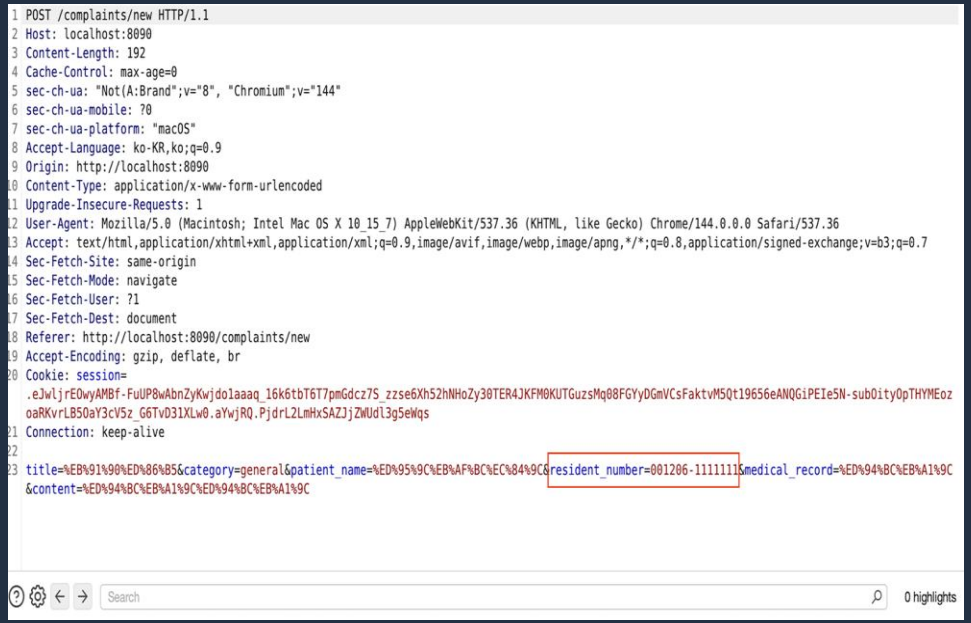
환경 확인: http://localhost:8090 접속 확인 (HTTPS 미적용)

Step 2

데이터 입력: 주민등록번호 등 민감 정보를 포함한 민원 신청서 제출

Step 3

취약점 입증: 패킷 본문에서 resident_number=001206-1111111이 그대로 노출됨을 확인



A05 Injection 취약점 개요

취약점 개요

명칭: 인젝션
관련 항목: A05:2025 Injection
위험도: Medium ~ High (중~상)

영향

검증되지 않은 입력값이 실행되면서 공격자가 데이터베이스 조회·수정·삭제, 계정 탈취, 서버 명령 실행 등 수행가능
이를 통해 대량의 개인정보 유출, 관리자 권한 획득, 시스템 전체 장악으로 이어질 수 있음

A05 Injection 상세 설명

취약점 상세 설명

사용자 입력값이 적절한 필터링 없이 쿼리 명령어로 해석되어 시스템의 제어 권한이 외부로 노출됨

특히 템플릿 엔진의 자동 보호 기능을 해제하는 설정으로 인해 악성 자바스크립트가 브라우저에서 실행됨

이를 통해 공격자는 인증을 우회하여 데이터베이스에 접근하거나 타인의 세션 정보를 탈취할 수 있음

취약점 근거

OWASP 공식 페이지 일부(https://owasp.org/Top10/2025/A05_2025-Injection/)

> 범주에는 37개의 CWE가 보고되어 모든 범주 중에서 가장 많은 CVE가 발생했습니다. 인젝션에는 3만 건 이상의 CVE가 보고된 크로스 사이트 스크립팅(XSS)(빈도 높음/영향력 낮음)과 1만 4천 건 이상의 CVE가 보고된 SQL 인젝션(빈도 낮음/영향력 높음)이 포함됩니다.

A05 Injection 재현(1/2)

Step 1 (SQLi)
관리자 (admin /admin1234)로 로그인

Step 2 (XSS)
민원 게시판(http://localhost:8090/posts)로 접근함.

```
# was/app/routes.py:675
# [원인] f-string을 사용하여 입력값을 쿼리 구조에 직접 삽입 (Raw SQL 사용)
sql = f"SELECT id FROM user WHERE username = '{username}'"

# [실행] 입력값이 데이터가 아닌 '명령어'로 해석되어 실행됨
result = db.session.execute(text(sql)).first()
```

A05 Injection 재현(2/2)

Step 3
검색창에 공격구문을 삽입
<img src=x onerror=alert('XSS_Detected')>

Step 4
Img태그안의 경로가 오류를 발생시키고,
이에 따라 악성코드가 실행됨

백엔드 로직_ 검색어를 받아 검색 수행

```
def posts_list():
    q = request.args.get("q", "").strip() # 검색어 파라미터 'q' 취득
    category = request.args.get("category", "all")
    query = Post.query.join(User, Post.user_id == User.id)

    if q:
        keyword = f"%{q}%"
        # 제목(title), 내용(content), 작성자명(username)에서 검색어 포함 여부 확인
        query = query.filter(
            (Post.title.ilike(keyword))
            | (Post.content.ilike(keyword))
            | (User.username.ilike(keyword))
```

프론트 로직_전달받은 검색어를 실행시킴

```
<!-- was/app/templates/posts/list.html:51 -->
<!-- [원인] |safe 필터가 템플릿 엔진의 자동 보호(Escape) 기능을 꺼버림 -->
<span style="color: var(--text-dim);">검색어 <strong>'{{ q|safe }}'</strong>에
대한 결과입니다.</span>
```

A05 Injection 대응 방안

01 매개변수와 쿼리 및 ORM 사용

사용자의 입력값이 데이터베이스 명령어로 해석되지 않도록 Raw SQL 대신 **매개변수화된 쿼리 또는 ORM(SQLAlchemy 등)**을 사용하여 SQL 인젝션을 원천 차단해야 함.

02 자동 이스케이프 및 데이터 정화

템플릿 엔진의 |safe **와 같은 위험한 설정을 지양**하고, 부득이하게 HTML을 허용해야 할 경우 서버 측에서 허용된 태그만 남기는 **데이터 정화(Sanitization)** 과정을 반드시 거쳐야 함.

A06 Insecure Design 취약점 개요

취약점 개요

명칭: 불안정한 디자인
관련 항목: A06:2025 Insecure Design
위험도: Medium ~ High (중~상)

영향

보안을 고려하지 않은 설계로 인해 비즈니스 로직 우회, 권한 상승, 인증 절차 우회 등의 구조적 취약점이 발생할 수 있음.
단일 취약점보다 장기적이고 광범위한 보안 사고로 확장될 가능성이 높으며, 근본적인 시스템 재설계가 필요할 수 있음.

A06 Insecure Design 상세 설명

취약점 상세 설명

비즈니스 로직의 상태 전이 규칙이 서버 측에서 강제되지 않음.

예시: 민원의 정상적인 처리 절차는 접수 -> 검토 중 -> 처리 완료 | 반려 이지만,
중간단계 없이 바로 처리 완료 혹은 반려 처리가 가능함.

취약점 근거

CWE 공식 페이지 (https://cwe.mitre.org/data/definitions/841.html)

- CWE-841 Improper Enforcement of Behavioral Workflow

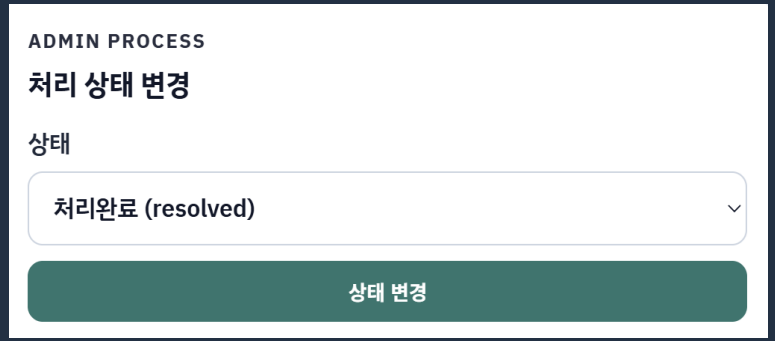
공격자는 예상치 못한 순서로 작업을 수행하거나 단계를 생략함으로써 제품의 비즈니스 로직을 조작하거나 제품이 유효하지 않은 상태에 빠지도록 만들 수 있습니다. 경우에 따라 이러한 행위는 결과적으로 취약점을 노출시킬 수도 있습니다.

A06 Insecure Design 재현

Step 1
관리자 (admin /admin1234)로 로그인

Step 2
민원 처리 페이지(http://localhost:8090/complaints)로 접근함.

Step 3
처리상태를 임의로 변경함.
* 정상절차: 접수 -> 처리 중 -> 처리완료 | 반려



A06 Insecure Design 대응 방안

01 **서버 측 비즈니스 로직 강제화**

중요 워크플로의 상태 전이 및 권한 체크는 클라이언트의 요청에 의존하지 않고, **서버 측에서 매 단계마다 현재 상태와 권한을 독립적으로 검증**하여 비정상적인 단계 우회를 방지해야 함.

02 **시스템 설계 강화**

시스템 설계 단계에서 발생 가능한 취약 경로를 식별하기 위해 비즈니스 흐름 전반에 대한 **위협 모델링(Threat Modeling)**을 반드시 수행해야 함.

A07 취약점 개요 및 상세 설명

취약점 개요

명칭: 불충분한 인증 메커니즘을 이용한 무차별 대입 공격
관련 항목: A07: Identification and Authentication Failures
위험도: High (상)

영향

자동화 도구를 이용한 탈취 및 비인가 계정 접근.

취약점 설명 및 시나리오

- **원인**: 계정 잠금(Lockout), 요청 제한(Rate Limiting), CAPTCHA 등 방어 기전 부재
- **현상**: 단 시간 내 발생하는 수백 건의 로그인 시도를 차단하지 않고 모두 허용함

예시 1

2009년 1월, 공격자는 트위터 서버가 로그인 시도 횟수를 제한하지 않는다는 점을 이용해 관리자 권한으로 서버에 접근할 수 있었습니다[REF-236]. 공격자는 트위터 지원팀 직원을 표적으로 삼아, 흔히 사용되는 단어들을 무작위로 조합하는 무차별 대입 공격을 통해 해당 직원의 비밀번호를 알아냈습니다. 지원팀 직원 계정으로 접근한 공격자는 관리자 패널을 이용하여 유명인과 정치인들의 계정 33개에 접근했습니다. 최종적으로는 해킹된 계정에서 온 것처럼 위장한 가짜 트위터 메시지를 게시했습니다.

예시 1 참고자료:
[REF-236] Kim Zetter. "취약한 비밀번호가 트위터 해커에게 '행복'을 가져다주다". 2009년 1월 9일. < https://www.wired.com/2009/01/professed-twitt/ >. *URL 유효성 검사 완료: 2023년 4월 7일*.

A07 취약점 재현 절차

Step 1

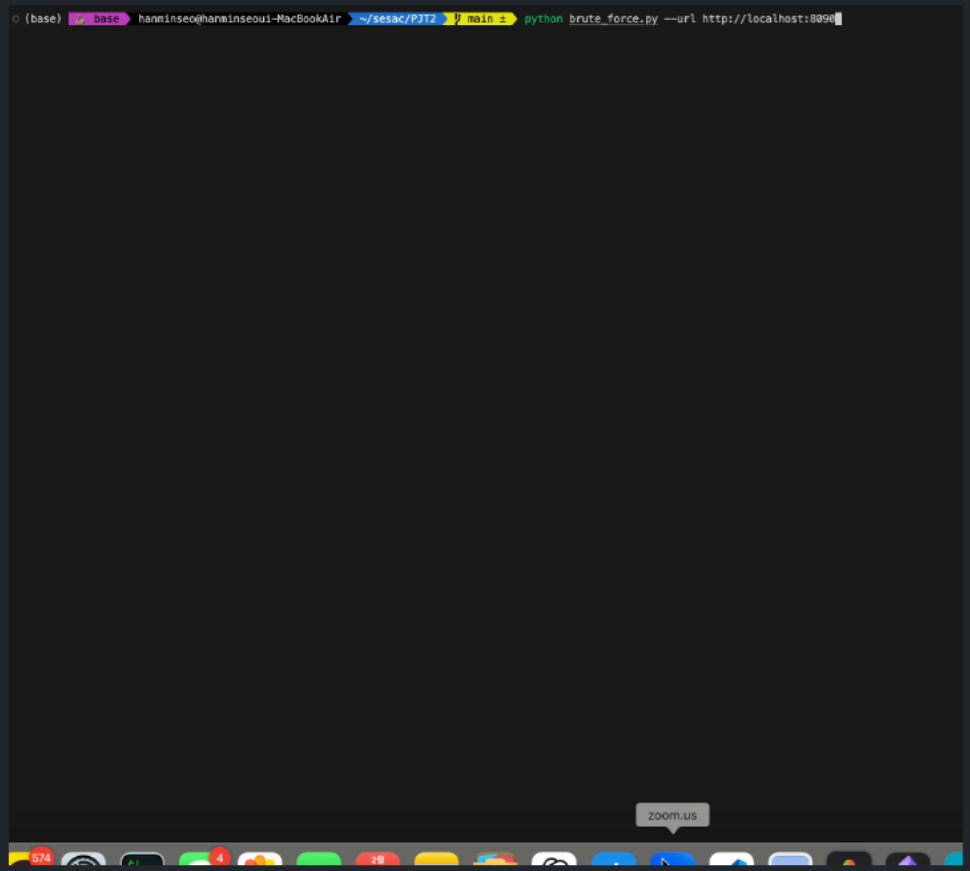
공격 준비: rockyou.txt 파일과 자동화 공격 스크립트 준비

Step 2

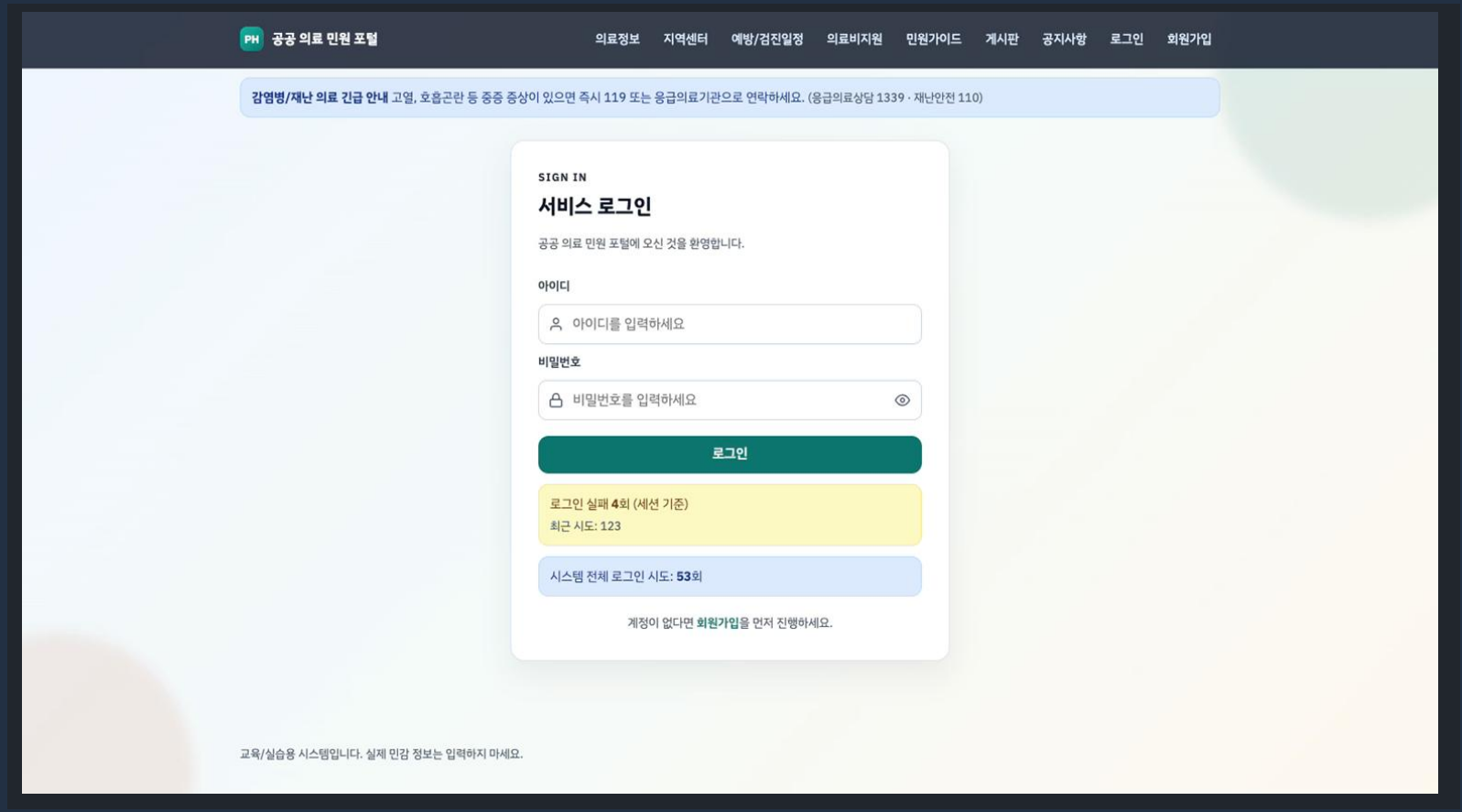
공격 실행: 타겟 계정을 대상으로 초당 수십 회의 무차별 대입 시도

Step 3

취약점 입증: 서버 차단 없이 웹 화면 하단에 로그인 실패 N회가 실시간 누적됨을 확인



A07 취약점 재현 절차


공공 의료 민원 포털
의료정보
지역센터
예방/검진일정
의료비지원
민원가이드
게시판
공지사항
로그인
회원가입
감염병/재난 의료 긴급 안내 고열, 호흡곤란 등 중증 증상이 있으면 즉시 119 또는 응급의료기관으로 연락하세요. (응급의료상담 1339 · 재난안전 110)
SIGN IN
서비스 로그인
공공 의료 민원 포털에 오신 것을 환영합니다.
아이디
아이디를 입력하세요
비밀번호
비밀번호를 입력하세요
로그인
로그인 실패 4회 (세션 기준)
최근 시도: 123
시스템 전체 로그인 시도: 53회
계정이 없다면 회원가입을 먼저 진행하세요.
교육/실습용 시스템입니다. 실제 민감 정보는 입력하지 마세요.

A08 취약점 개요

취약점 개요

명칭: **역직렬화 취약점**
관련 항목: A08:2025 Software or Data Integrity Failures
위험도: High (상)

영향

공격자가 조작한 객체를 통해 서버 측에서 임의의 시스템 명령을 실행(RCE)하거나, 무결성 검증 부재를 악용하여 데이터 변조 및 서비스 거부(DoS) 유발

A08 취약점 상세 설명

취약점 상세 설명

- 객체 복원 특성 악용: 직렬화 데이터가 메모리 객체로 변환될 때 포함된 로직이 그대로 복구되는 매커니즘을 이용함
- 무결성 검증 부재: 외부 입력값에 대해 디지털 서명이나 HMAC 등 데이터 변조 여부를 확인하는 절차가 누락됨.
- 악성 코드 자동 실행: pickle의 __reduce__ 등 객체 생성 시 자동으로 호출되는 메서드에 공격 명령을 주입
- 시스템 제어권 탈취: 단순 데이터 변조를 넘어 원격 코드 실행(RCE)을 통해 서버 권한을 완전히 장악함.

취약점 근거

OWASP 공식 홈페이지에 해당하는 주요 CWE로는 *CWE-502(Deserialization of Untrusted Data)* 가 있다.

Impact	Details
Modify Application Data; Unexpected State	**Scope: Integrity** Attackers can modify unexpected objects or data that was assumed to be safe from modification. Deserialized data or code could be modified without using the provided accessor functions, or unexpected functions could be invoked.

A08 취약점 재현 절차

Step 1
준비된 공격 페이로드를 취약한 API(게시글 작성)에 전송

Step 2
서버 측 도커 컨테이너 로그를 통해 명령어가 실행되었음을 확인
클라이언트 측에서 성공 여부를 확인하기 위해 ping 등의 명령어를 활용 가능

```
PS C:\project\python\PJT2> docker exec -it public-health-was ls -l /tmp/success.txt
-rw-r--r-- 1 root root 0 Feb 11 02:01 /tmp/success.txt
```

A08 대응 방안

01 안전한 포맷 전환

실행 권한이 없는 JSON, XML 등 정적 데이터 포맷으로 전면 교체하여 위험을 차단함.

02 무결성 메커니즘 도입

부득이한 경우 비밀키 기반 HMAC 서명을 추가하여 검증된 데이터만 역직렬화를 수행함.

03 최소 권한 및 검증

서비스를 Non-root 계정으로 구동하고 Allow-list 기반의 입력 검증 로직을 적용함.

A09 취약점 개요

취약점 개요

명칭: 보안 로깅 및 경고 실패
관련 항목: A09: Security Logging and Alerting Failures
위험도: Medium ~ High (중~상)

영향

공격 발생 여부를 탐지하지 못하고 사고 발생 후 추적 및 포렌식 분석이 어려워짐.
또한 로그에 민감 정보가 기록되면 로그 유출 시 2차 피해로 이어질 수 있음.

A09 취약점 상세 설명

취약점 상세 설명

보안 이벤트가 충분히 기록되지 않거나, 공격 시도에 대한 경고가 발생하지 않는 경우 발생함.
또는 로그에 비밀번호, 토큰 등 민감 정보가 평문으로 기록되면 로그 자체가 공격 대상이 됨.

취약점 근거와 시나리오

OWASP 공식 홈페이지에 해당하는 주요 CWE로는 CWE-532(*로그 내 민감 정보 저장*), *CWE-778(로그 및 모니터링 부족*) 이 있다.

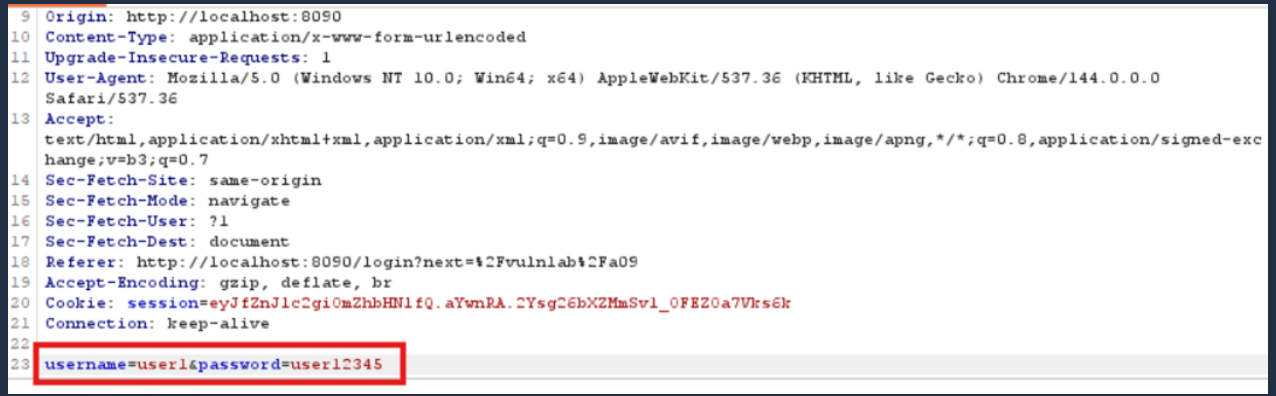
시나리오 #1: 한 아동 건강 보험 회사의 웹사이트 운영자는 모니터링 및 로깅 시스템 부족으로 데이터 유출을 감지하지 못했습니다. 외부 기관에서 해당 보험 회사에 공격자가 350만 명이 넘는 아동의 민감한 의료 기록 수천 건에 접근하여 수정했다고 알렸습니다. 사고 후 검토 결과, 웹사이트 개발자들이 중대한 취약점을 해결하지 않았던 것으로 드러났습니다. 시스템에 대한 로깅이나 모니터링이 이루어지지 않았기 때문에, 데이터 유출은 2013년부터 7년 이상 지속되었을 가능성이 있습니다.

A09 취약점 재현 절차(1/2)

<CASE 1>
로그에 비밀번호가 평문으로 기록됨.

Step 1
일반 사용자(user1 / user12345)로 로그인함.

Step 2
패킷 본문에서password가 마스킹 되지 않고 그대로 노출됨.



A09 취약점 재현 절차(2/2)

<CASE 2>
로그인 실패가 수백 번 발생해도 알림 없음.

Step 1
dic.txt 파일에 대입할 비밀번호 목록을 저장함.

Step 2
무차별 공격 시도

Step 3
관리자로 로그인 후 모니터링 상태 확인
-> 로그인 실패 누적, 임계치 기반 탐지, 보안 알림 없음.

```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# hydra -l user1 -P /home/kali/Desktop/dic.txt 192.168.101.149 http-post-form "/login:password=^PASS^:F=id or password is wrong." -s 8090
Hydra v9.6 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2026-02-11 04:36:03
[DATA] max 14 tasks per 1 server, overall 14 tasks, 14 login tries (l:1/p:14), ~1 try per task
[DATA] attacking http-post-form://192.168.101.149:8090/login:password=^PASS^:F=id or password is wrong.
[8090][http-post-form] host: 192.168.101.149   login: user1   password: 12345
[8090][http-post-form] host: 192.168.101.149   login: user1   password: computer
[8090][http-post-form] host: 192.168.101.149   login: user1   password: abc123
[8090][http-post-form] host: 192.168.101.149   login: user1   password: 1234
[8090][http-post-form] host: 192.168.101.149   login: user1   password: password
[8090][http-post-form] host: 192.168.101.149   login: user1   password: john
[8090][http-post-form] host: 192.168.101.149   login: user1   password: hello
[8090][http-post-form] host: 192.168.101.149   login: user1   password: michael
[8090][http-post-form] host: 192.168.101.149   login: user1   password: ubuntu
[8090][http-post-form] host: 192.168.101.149   login: user1   password: bye
[8090][http-post-form] host: 192.168.101.149   login: user1   password: love1234
[8090][http-post-form] host: 192.168.101.149   login: user1   password: kali
[8090][http-post-form] host: 192.168.101.149   login: user1   password: user123
[8090][http-post-form] host: 192.168.101.149   login: user1   password: chocolate
1 of 1 target successfully completed, 14 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2026-02-11 04:36:04
```

공공 의료 민원 포털
의료정보 지역센터 예방/검진일정 의료비지원 민원가이드 게시판 공지사항 민원 마이페이지 취약점 실습 관리자
로그 모니터링 로그아웃 admin · admin

2026-02-11 18:36:04	-	login_attempt	POST	-	endpoint=/login;username=-;result=failed;ip=172.20.0.1;ua=-;reason=empty_username
2026-02-11 18:36:04	-	login_failed	POST	-	endpoint=/login;username=-;result=failed;ip=172.20.0.1;ua=-;reason=empty_username
2026-02-11 18:36:04	-	web_request	POST	/login	status=200;endpoint=login;ip=172.20.0.1;query=;ua=-
2026-02-11 18:36:04	-	login_attempt	POST	-	endpoint=/login;username=-;result=failed;ip=172.20.0.1;ua=-;reason=empty_username
2026-02-11 18:36:04	-	login_failed	POST	-	endpoint=/login;username=-;result=failed;ip=172.20.0.1;ua=-;reason=empty_username
2026-02-11 18:36:04	-	web_request	POST	/login	status=200;endpoint=login;ip=172.20.0.1;query=;ua=-
2026-02-11 18:36:04	-	login_attempt	POST	-	endpoint=/login;username=-;result=failed;ip=172.20.0.1;ua=-;reason=empty_username
2026-02-11 18:36:04	-	login_failed	POST	-	endpoint=/login;username=-;result=failed;ip=172.20.0.1;ua=-;reason=empty_username
2026-02-11 18:36:04	-	web_request	POST	/login	status=200;endpoint=login;ip=172.20.0.1;query=;ua=-
2026-02-11 18:36:04	-	login_attempt	POST	-	endpoint=/login;username=-;result=failed;ip=172.20.0.1;ua=-;reason=empty_username
2026-02-11 18:36:04	-	login_failed	POST	-	endpoint=/login;username=-;result=failed;ip=172.20.0.1;ua=-;reason=empty_username
2026-02-11 18:36:04	-	web_request	POST	/login	status=200;endpoint=login;ip=172.20.0.1;query=;ua=-
2026-02-11 18:36:04	-	login_attempt	POST	-	endpoint=/login;username=-;result=failed;ip=172.20.0.1;ua=-;reason=empty_username
2026-02-11 18:36:04	-	login_failed	POST	-	endpoint=/login;username=-;result=failed;ip=172.20.0.1;ua=-;reason=empty_username

A09 대응 방안

01 민감 정보 로깅 금지 및 마스킹 처리

비밀번호, 토큰, 인증키 등 민감 정보는 로그에 기록하지 않거나 마스킹 처리해야함.

02 보안 이벤트 로깅 표준화

로그인 성공/실패, 권한 변경, 관리자 접근 등 주요 보안 이벤트를 표준 포맷으로 기록해야함.

03 로그인 실패 임계치 탐지 및 알림

로그인 실패 횟수를 누적 관리하고 임계치 초과 시 관리자 또는 보안 시스템으로 알림을 전송해야함.

A10 취약점 개요

취약점 개요

명칭: HTTP 요청 라인 임계값 초과 시 상세 설정 정보 노출
관련 항목: A10: Exceptional Conditions
위험도: Low to Medium (중하)

영향

서버 내부의 상세 에러 메시지가 노출되어 공격자가 인프라의 설정 임계값을 파악하고 정교한 DoS 공격의 지표로 활용할 수 있음.

A10 취약점 상세 설명

취약점 상세 설명

시스템이 예상치 못한 예외(Exception) 상황을 만났을 때, 이를 사용자에게 일반적인 메시지로 안내하지 않고 서버의 내부 동작이나 설정 임계값을 그대로 출력하는 결함임. /posts, /notices 검색 기능에서 비정상적으로 긴 쿼리 스트링을 전송할 경우, 서버가 허용하는 최대 길이 정보를 에러 메시지에 포함하여 반환함.

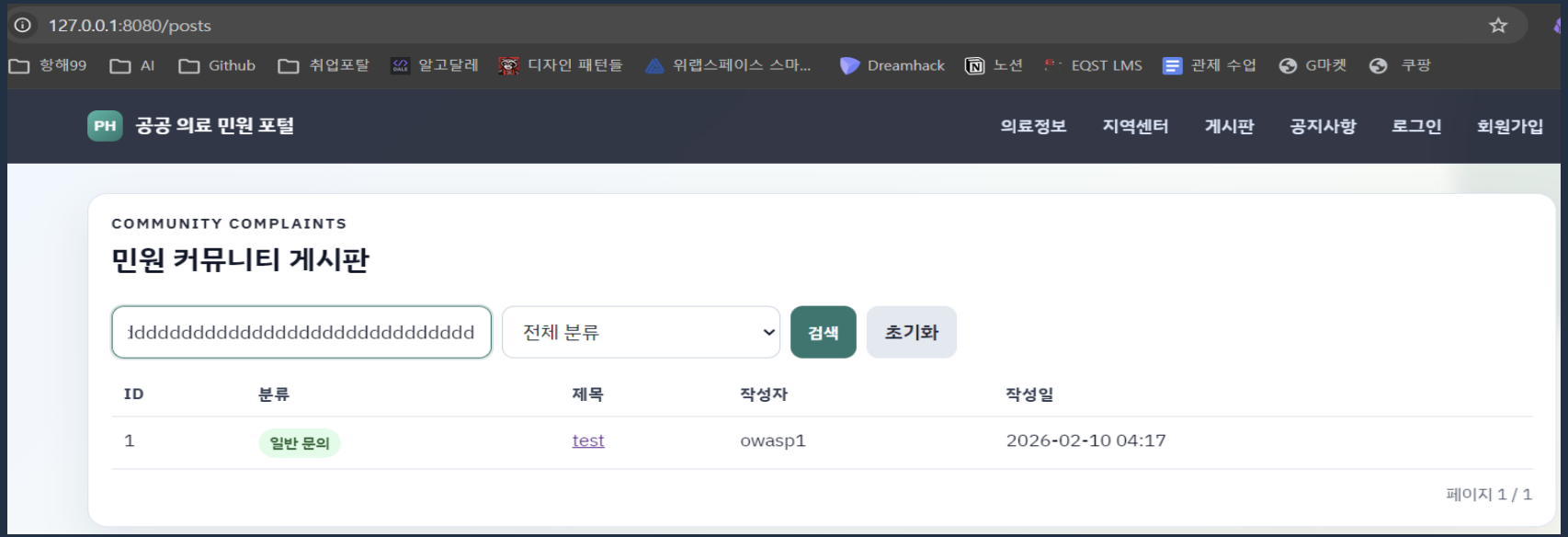
취약점 근거와 시나리오

OWASP 공식 홈페이지에 해당하는 주요 CWE로는 CWE-209(민감한 정보가 포함된 오류메시지 생성)이 있다.

시나리오 #2: 부적절한 처리 또는 데이터베이스 오류로 인해 민감한 데이터가 노출되어 사용자에게 시스템 오류 전체가 드러나는 경우. 공격자는 민감한 시스템 정보를 이용하여 더 정교한 SQL 인젝션 공격을 구축하기 위해 지속적으로 오류를 발생시킵니다. 사용자 오류 메시지에 포함된 민감한 데이터는 정찰 정보입니다.

A10 취약점 재현 절차 (1/2)

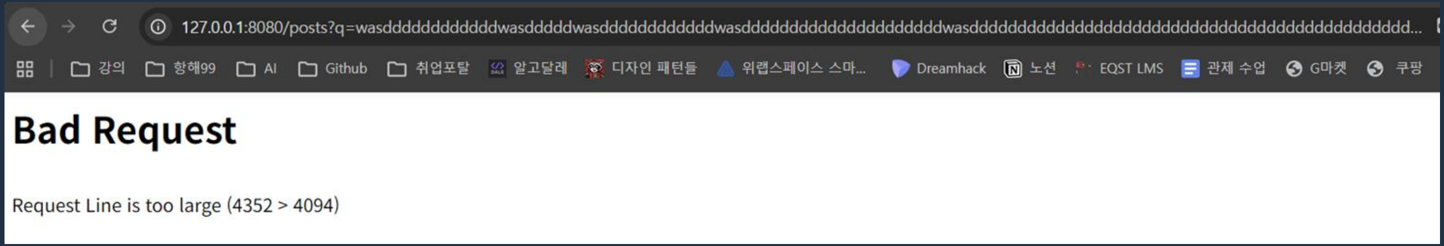
Step 1: 게시판 검색 페이지(http://127.0.0.1:8090/posts)에 접속함

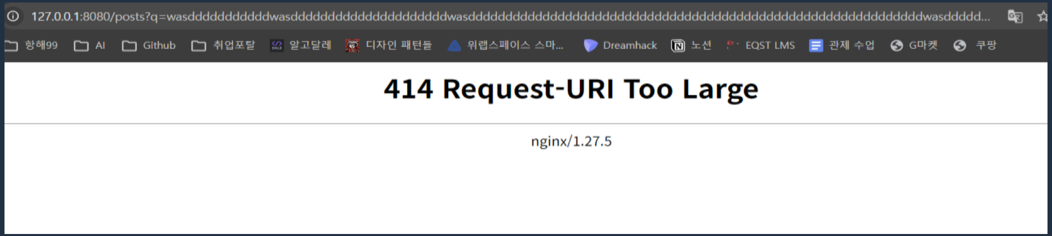


A10 취약점 재현 절차 (2/2)

Step 2: q 파라미터에 약 5,500자 이상의 긴 문자열을 담아 GET 요청을 전송함.
서버 응답에서 "Request Line is too large (5539 > 4094)" 메시지를 확인하여 내부 요청 제한 임계값(4094 bytes)을 파악함.

Step 3: 더 긴 문자열을 요청하면 nginx 버전(1.27.5)까지 노출함.





A10 대응 방안

01 Generic Error Messages

상세 에러를 숨기고 "잘못된 요청입니다"와 같은 공통 에러 페이지를 반환하도록 전역 예외 처리기(@errorhandler)를 구현함.

02 Fail-Safe 원칙

예외 발생 시 서버는 시스템의 내부 정보를 노출하는 대신 가장 보수적인 안전 상태(Fail-Close)를 유지해야 함.

정보보안의 3대 요소 (CIA)

기밀성

개인정보
군사 기밀

가용성

발전소 제어
스마트 팩토리

무결성

금융 거래
의료 데이터

OWASP Top 10과 CIA

코드	항목	주요 보안 요소
A01	Broken Access Control	기밀성, **무결성**
A02	Security Misconfiguration	기밀성, **무결성**, 가용성
A03	Software Supply Chain	**무결성**, 가용성
A04	Cryptographic Failures	기밀성, **무결성**
A05	Injection	기밀성, **무결성**, 가용성
A06	Insecure Design	기밀성, **무결성**, 가용성
A07	Authentication Failures	기밀성, **무결성**
A08	Integrity Failures	**무결성**, 가용성
A09	Logging & Alerting Failures	**무결성**
A10	Exceptional Conditions	기밀성, 가용성

마이데이터 연동 관점 보안 포인트

현재 구조	제공기관 Mock API + consent/token 기반 조회
검증 포인트	client 인증, token 만료/폐기, 사용자-데이터 바인딩
로그 포인트	mydata_fetch, mydata_report_download, web_request
개선	토큰 scope 분리, 재시도/이상패턴 탐지, 감사추적 강화

결론

3-Tier 구조에서 보안 실패 지점을 체계적으로 학습을 위한 서비스 구성

OWASP Top 10 전 항목을 실제 기능과 연결해 분석

각 항목별 근거 기반 분석 및 실행 가능한 개선안

감사합니다
Thank you