DB 침입 탐지 시스템(DB-IDS)

졸업프로젝트

202211252 컴퓨터공학부 고승우

Project Overview

프로젝트 개요

DB 침입 탐지 시스템 (DB-IDS)

데이터베이스 앞단에 위치한 SQL 프록시 기반의 침입 탐지 시스템

웹 애플리케이션 로그의 한계

웹 애플리케이션 로그만으로는 DB 내부 공격을 실시간으로 파악하기 힘듦

DB 레벨 위협 가시성 부족

SQL Injection, 대량 조회, 비인가 계정 접근 등 DB 자체 위협에 대한 가시성이 부족

운영 DBA의 모니터링 보완

운영 DBA의 쿼리 모니터링 부담과 사고 후 회귀 분석 한계를 보완

Project Goal

기능적 목표

1 FR-1. 실시간 SQL 로그 수집

모든 SQL 쿼리를 실시간으로 수집하고 저장하여 DB 트래픽의 완벽한 가시성 확보

2 FR-2. 규칙 기반 공격 탐지

DROP TABLE, UNION SELECT와 같은 대표적인 DB 공격 패턴을 규칙 기반으로 즉시 탐지

3 FR-3. 행동 기반 이상 탐지

짧은 시간 내 비정상적인 대량 쿼리, 스캔 패턴 등 비정상적인 행위 자동으로 탐지

4 FR-4. 권한 기반 비정상 쿼리 탐지

READ_ONLY 계정의 DDL/DML 시도 등 계정 권한에 맞지 않는 쿼리 실행 감지

5 FR-5. 실시간 알림 시스템

탐지된 이벤트를 Slack/Email로 신속하게 전달하여 관리자의 대응 시간 최소화

6 FR-6. 관리자 웹 대시보드 제공

로그와 이벤트를 시간, 유형, 계정별로 편리하게 조회하고 필터링할 수 있는 대시보드 제공

7 FR-7. 로그 장기 보관 및 아카이브

일정 기간이 지난 로그를 안전하게 장기 보관하여 상세한 사후 분석이 가능하도록 함

Project Goal

비기능적 목표

NFT-1. 성능: 탐지 지연 & 처리량

초당 1,000건의 트래픽 환경에서도 안정적인 침입 탐지 및 처리 보장

NFT-2,3. 성능: 동시성 & 대시보드 응답성

다수의 관리자가 동시에 대시보드를 사용할 때도 응답 지연 최소화

NFT-4,5,6. 보안: 공격 탐지 & 데이터 보호

SQL Injection, 권한 위반 시도를 정확히 탐지

관리자 계정 비밀번호 안전하게 저장

NFT-7,8. 신뢰성: 탐지율 & 알림 성공률

공격 이벤트에 대한 높은 탐지율과 알림 성공률 지속적 유지

Project Archievement Standard

달성 기준

01

실시간 SQL 로그 수집 (FR-1, NFT-1, NFT-2)

전송된 SQL이 **누락 없이 100% 기록**

로그 저장까지의 **평균 지연 시간 ≤ 1초**

지속 처리량 ≥ 100/sec

02

공격 패턴 기반 탐지 (FR-2)

DROP TABLE, UNION SELECT 등 정의된 핵심 패턴에 대한 **탐지 누락 없이 기록**

03

G3. 행동 기반 이상 탐지 (FR-3)

쿼리 실행 시간, 반환 행 수, 빈도 기반 이상에 대한 **탐지 누락 없이 기록**

04

G4. 권한 기반 탐지 (FR-4)

UPDATE/DELETE 등 권한 밖 쿼리 시도 시 **탐지 누락 없이 기록**

05

G5. 실시간 알림 연동 (FR-5, NFT-8)

탐지 이벤트 발생 시 **알림 성공률 ≥ 99%**

이벤트 발생 → Slack/Email 도착까지 **지연 시간 ≤ 10초**

06

탐지 신뢰성(NFT-7)

이벤트 탐지에 대해 다음과 같은 기준 만족

미탐(False Negative) ≤ 2%

오탐(False Positive) ≤ 5%

07

로그 장기 보관 및 아카이브 (FR-7)

일정 기간(30일) 지난 로그는 **아카이브 저장소로 자동 저장**

08

비밀번호 저장 보안 (NFT-06)

관리자 비밀번호를 평문이 아닌 **SHA-256 해시로 저장**해 안전성 보장

09

G6. 관리자 대시보드 제공 (FR-6, NFT-3)

쿼리 조회 기능 제공:

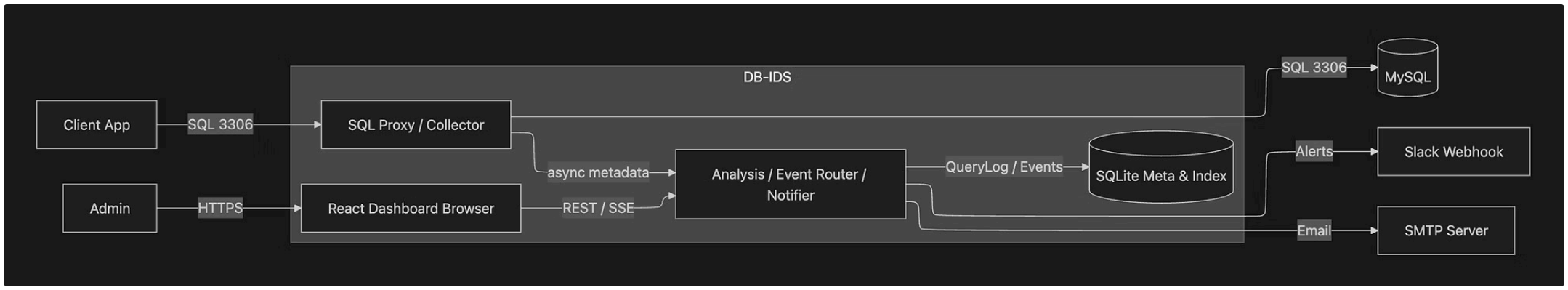
- 날짜, 사용자별 필터링 기능 / SQL 쿼리 요약문 제공 기능 / CSV 다운로드 기능

탐지 이벤트 조회 기능 및 요약 기능 제공

100명 동시 접속 상황에서 **응답 시간은 2초 이하**

System Design

시스템 아키텍처



1

클라이언트 애플리케이션

SQL 요청을 전송하는 주체. DB-IDS 시스템의 모니터링 대상 트래픽의 시작점

2

SQL Proxy/Collector

애플리케이션과 DB 사이에서 모든 SQL 쿼리를 가로채고, 분석 컴포넌트로 전달하는 핵심 중개자

3

MySQL 8.0 (모니터링 대상 DB)

클라이언트 앱의 SQL 요청을 실제로 처리하는 운영 데이터베이스

4

Analysis / Event Router / Notifier

SQL Proxy로부터 전달받은 쿼리를 분석하여 침입을 탐지하고, 이벤트를 처리하며, 필요한 경우 알림을 전송하는 컴포넌트

5

SQLite Meta & Index

최근 SQL 로그 및 탐지 이벤트를 저장하는 메타데이터 저장소

6

Slack Webhook / SMTP Server

탐지 이벤트 발생 시 실시간으로 알림을 받을 수 있는 외부 메시징 서비스 연동 지점

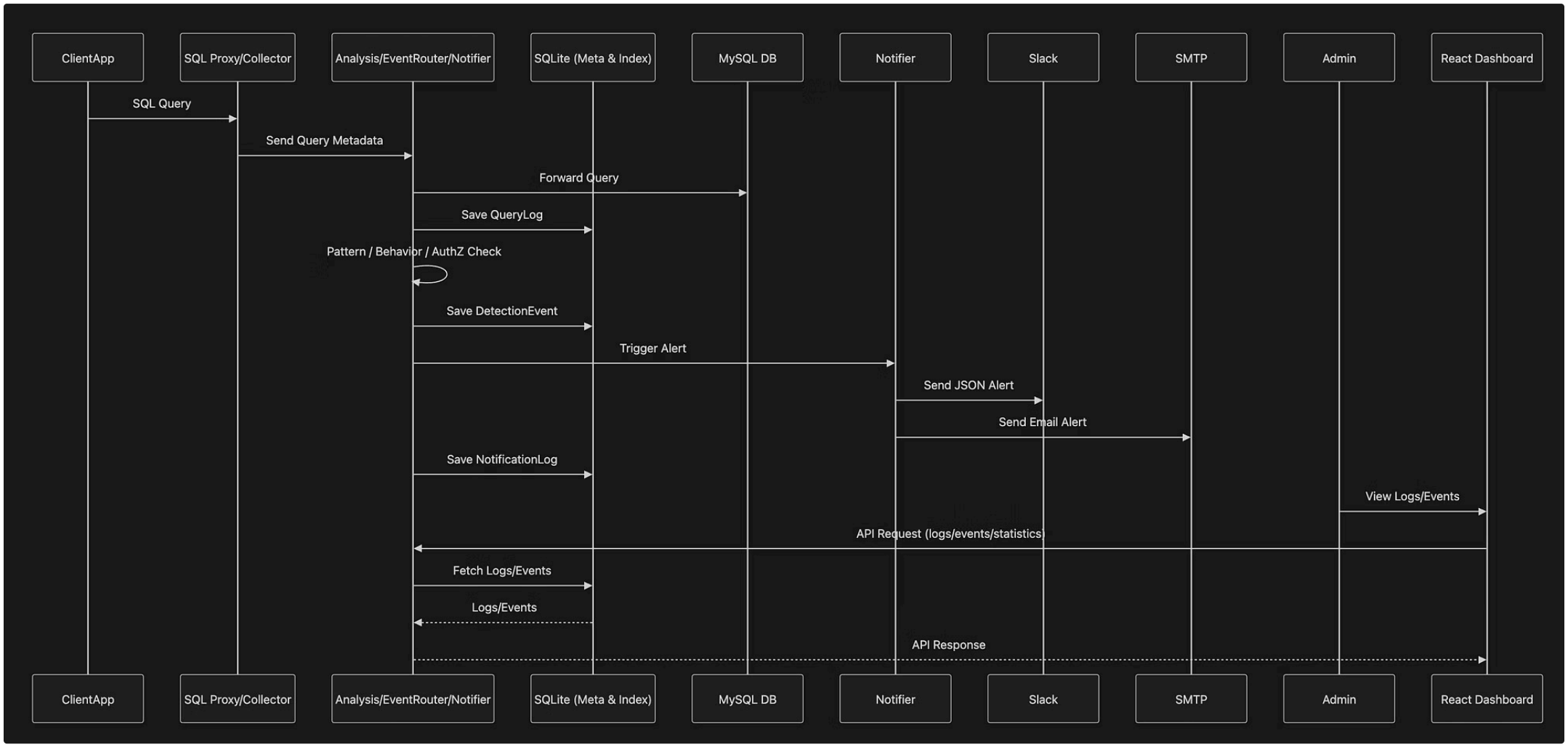
7

관리 대시보드

React 기반의 웹 프론트엔드로, SQL 로그와 탐지 이벤트를 조회하고 시스템을 관리하는 인터페이스

System Design

시스템 워크플로우



쿼리 처리 & 탐지

ClientApp → SQL Proxy/Collector로 SQL 전송

Proxy가 쿼리 메타데이터를 **Analysis/EventRouter/Notifier**에 전달, 쿼리는 MySQL로 포워딩

엔진이 **QueryLog 저장 → Pattern / Behavior / AuthZ 탐지 → DetectionEvent 저장**

알림 발송

고위험 이벤트 발생 시 엔진이 **Notifier로 Alert 트리거**

Notifier가 **Slack, Email로 공격 알림 전송**

알림 결과를 **NotificationLog로 기록**

대시보드 조회

Admin이 React Dashboard에서 **로그·이벤트 조회 요청**

Dashboard → 엔진 → SQLite 순으로 데이터를 조회하고

공격 이벤트·쿼리 이력·알림 상태를 **한 화면에서 시각화**

Demo

데모

Unit Test

단위 테스트(UT)

01

쿼리 처리 검증

비정상 UTF-8, SQL 예외 (SQLException) 발생 시 정상 처리 확인 (UT-01, UT-02)

02

정규화 로직 검증

1=1 → 0=0 리터럴 마스킹, 주석 제거 등 우회 시도 무력화 검증 (UT-03-RE)

03

패턴 탐지 검증

정규화된 SQL에 대해 패턴 탐지 이벤트 생성 및 심각도 산정 로직 검증 (UT-03-DS / UT-04-DS)

04

행동 탐지 검증

단위 시간 당 질의 폭주 등의 행동 패턴의 이벤트를 생성하는 로직 검증 (UT-05 / UT-06)

05

권한 탐지 검증

룰 기반 권한정책에 따라 권한 위반 이벤트를 생성하는 로직 검증 (UT-07 / UT-08)

06

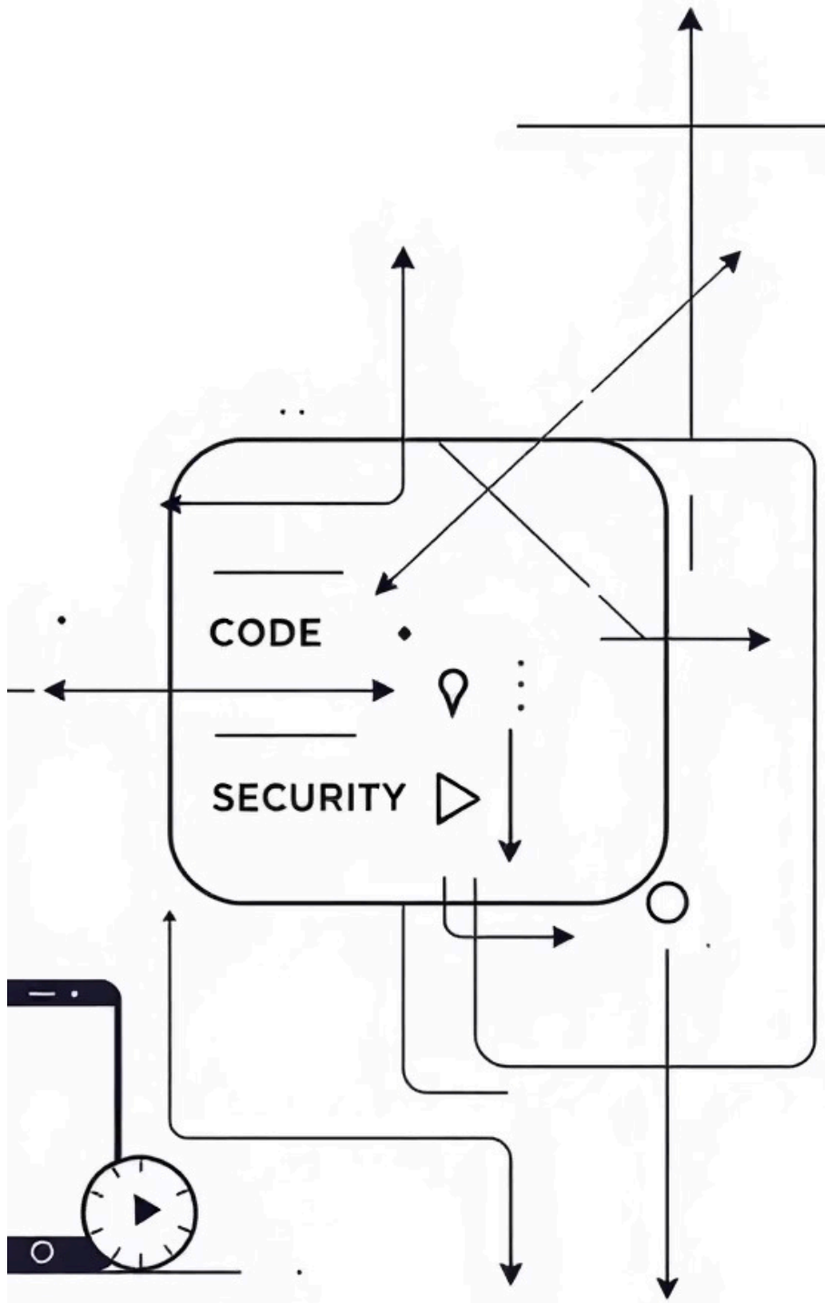
알림 전송 검증

이메일과 슬랙 알림 전송에 대한 검증 (UT-09 / UT-10)

07

쿼리 로그 저장 검증

쿼리 로그 저장 로직의 유효성 검증/정규화/저장 연동을 단위 수준에서 검증 (UT-11)



Integration Test

통합 테스트(IT)

데이터 파이프라인

로그 수집(POST)부터 조회(GET)까지 데이터 일관성 및 ID 일치 확인 (IT-01)

탐지-알림 연계

DROP TABLE 수집(POST) → PATTERN/HIGH 이벤트 생성 → **Mailhog(Email)** 및 **WireMock(Slack)** 동시 발송 성공 (IT-05)

대시보드(UI) 연동 검증

백엔드에 생성된 데이터가 대시보드에 정확히 반영

탐지 3종 연계

패턴(IT-02), 행동(IT-03), 권한(IT-04) 이벤트 모두 API 레벨에서 정상 생성 및 조회 확인

System Test

시스템 테스트(ST)

1

핵심 시나리오 6종 Pass

로깅, 패턴, 행동, 권한, 알림, UI 정합성 모두 통과

2

데이터 정합성(ST-06)

EC2 대시보드 UI ↔️ API 응답 ↔️ DB 데이터 간 불일치 없음 확인

3

검증 완료

사용자 요구사항(FR-1~6)의 핵심 기능이 실제 환경에서 정상 동작함을 입증

System Test

시스템 테스트(ST) 개요

테스트 환경

- **운영체제:** Ubuntu 20.04
- **JDK:** Temurin 21, Spring Boot 3.3.2
- **프론트엔드:** React 19 / Vite 7
- **데이터베이스:** SQLite 3.45
- **빌드 도구:** Gradle 8.14
- **알림 도구:** Gmail, Slack Webhook

테스트 범위

로그 수집

SQL 트래픽의 수집 검증

탐지 (PATTERN/BEHAVIOR/AUTHZ)

패턴, 행동, 권한 기반의 공격 탐지 로직 검증

알림

이벤트 발생 시 실시간 알림 전송 검증

대시보드/UI/DB 일관성

전체 데이터 흐름의 정합성 엔드투엔드 검증

System Test

시스템 테스트(ST) 상세 결과

1	**로그 수집 & 조회 (FR-1)** 관리자가 `SELECT` 실행 로그를 `/api/ingest/log`로 전송 후, `/api/logs` 및 대시보드 `/logs`에서 동일 ID로 조회되는지 확인 **결과:** Pass – API 응답, DB, UI 모두 일치
2	**UNION SELECT 패턴 탐지 (FR-2)** `UNION SELECT password`를 포함하는 공격 쿼리 수집 후, `/api/events?type=PATTERN` 조회 시 `eventType=PATTERN`, `severity=MEDIUM`, 그리고 해당 `logId`가 확인되는지 검증 **결과:** Pass – 정의된 패턴에 대해 정상 탐지
3	**대량 SELECT 스파이크 탐지 (FR-3)** 동일 사용자가 1분 내에 200건의 `SELECT` 쿼리를 5개의 병렬 세션으로 집중 발생시킨 후, 윈도우 종료 후 `/api/events?type=BEHAVIOR` 조회 시 `eventType=BEHAVIOR`, `severity=HIGH`, QPM ≥ 100인지 검증 **결과:** Pass – 고빈도 트래픽에 대해 `BEHAVIOR/HIGH` 이벤트 생성
4	**READ_ONLY 권한 위반 탐지 (FR-4)** `READ_ONLY` 계정으로 `INSERT/DELETE` 요청 로그를 수집한 후, `/api/events?type=AUTHZ` 조회 시 `eventType=AUTHZ`, `severity=HIGH`, 그리고 해당 `logId`가 확인되는지 검증 **결과:** Pass – 권한 위반 시 `AUTHZ/HIGH` 이벤트 생성

System Test

시스템 테스트(ST) – 데이터 정합성 검증

관리자 대시보드 화면이 백엔드 API 응답 및 실제 DB 값과 일치하는지 검증합니다.

01

공격 유도: SQL 로그 전송

`POST /api/ingest/log`로 `DROP TABLE`이 포함된 공격 로그를 전송하고, 응답에서 `LOG_ID`를 추출

02

API 이벤트 조회

`GET /api/events?type=PATTERN&user=...` 엔드포인트에서 `eventType=PATTERN`, `severity=HIGH`, 그리고 추출된 `logId`가 정확히 일치하는지 확인

03

DB 직접 조회

`detection_event` 테이블에서 `log_id=LOG_ID` 레코드가 `PATTERN/HIGH`로 기록되었는지 확인하고, `query_log` 테이블에서 `id=LOG_ID`의 `status=FAILURE` 및 **SQL** 미리보기에 `DROP TABLE`이 표시되는지 검증

04

UI 화면 확인

관리자 대시보드의 `/logs` 화면에서 해당 사용자 행에 `FAILURE`와 `DROP TABLE`이 올바르게 표시되는지, 그리고 `/events` 화면에 `PATTERN/HIGH` 이벤트가 노출되는지 최종적으로 확인

```
[ST-06] 1) PATTERN/HIGH 유도 : DROP TABLE
HTTP/1.1 201
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Location: /logs/8b441b7c-8d56-44da-9a3f-ce1da550d1da
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/json
Transfer-Encoding: chunked
Date: Tue, 11 Nov 2025 08:40:38 GMT

[ST-06] LOG_ID=8b441b7c-8d56-44da-9a3f-ce1da550d1da
[ST-06] EVENT_ID=14a219e2-ee79-453d-b82b-940878f29041 (PATTERN/HIGH)
```

```
[ST-06] detection_event: 방금 LOG_ID에 해당하는 이벤트
```

id	log_id	event_type	severity	occurred_at
14a219e2-ee79-453d-b82b-940878f29041	8b441b7c-8d56-44da-9a3f-ce1da550d1da	PATTERN	HIGH	2025-11-11T08:40:30.819192941Z

```
[ST-06] query_log: 해당 LOG_ID 상세
```

id	user_id	status	executed_at	sql_preview
8b441b7c-8d56-44da-9a3f-ce1da550d1da	st06-ui-db@test.com	FAILURE	2025-11-11T08:40:30.807469402Z	DROP TABLE important_table...

2025. 11. 11. 오후 5:40:30	st06-ui-db@test.com	FAILURE	0	drop table important_table	View

2025. 11. 11. 오후 5:40:30	st06-ui-db@test.com	PATTERN	HIGH	DROP TABLE IMPORTANT_TABLE

결과: API 응답, DB 데이터, UI 표시값이 모두 완전 일치하여 → Pass

Non-Functional Test

비기능 테스트(NFT) 개요

테스트 환경

- **운영체제:** Ubuntu 20.04
- **JDK:** Temurin 21, Spring Boot 3.3.2
- **프론트엔드:** React 19 / Vite 7
- **데이터베이스:** SQLite 3.45
- **빌드 도구:** Gradle 8.14
- **부하 도구:** Apache JMeter

NFT 검증 항목

성능 (NFT-01 ~ NFT-03)

탐지 지연, 처리량, 동시성 처리 능력

보안/신뢰성 (NFT-06 ~ NFT-08)

관리자 비밀번호 저장 방식, 탐지율, 알림 성공률

※ NFT-04 ~ 05 (SQL Injection, 권한 제어)는 시스템 테스트(ST-02~04)로 검증 대체

Non-Functional Test

비기능 테스트(NFT) 상세 결과

DB-IDS 시스템의 주요 비기능 요구사항에 대한 테스트를 진행하고, 각 항목별 상세 결과를 분석했습니다.

NFT-01: 탐지 지연

경량 동시성 환경에서 POST/GET 응답 시간 측정 (기준: 평균·95p ≤ 1,000ms, 오류율 < 5%)

결과: POST 평균 193ms (오류율 1.64%), GET 평균 197ms (오류율 3.34%)로 모든 기준 충족 → **Pass**

구분	#Samples	평균(ms)	95p(ms)	최대(ms)	오류율	처리량
A-POST /api/ingest/log	304	**193**	**453**	1032	1.64%	36.6/sec
A-GET /api/logs	1,317	**197**	**420**	1481	3.34%	159.6/sec
TOTAL	1,621	**197**	**431**	1481	**3.02%**	**195.3/sec**

NFT-02: 처리량

10 스레드, 60초 부하 환경에서 POST /api/ingest/log 지속 전송 (기준: 지속 처리량 ≥ 100 req/sec, 오류율 < 5%)

결과: 총 6,041건 처리, 평균 처리량 102 req/sec (오류율 0.83%)로 기준 충족 → **Pass**

구분	#Samples	평균(ms)	95p(ms)	최대(ms)	3,566	**0.83%**	**102.0/sec**
B-POST /api/ingest/log	6,041	**448**	**923**	3,566	**0.83%**	**102.0/sec**	
TOTAL	6,041	**448**	**923**	3,566	**0.83%**	**102.0/sec**	

NFT-03: 동시성

100명 동시 /api/logs 조회 (기준: 모든 샘플 응답 시간 ≤ 2,000ms, 오류율 0%)

결과: 평균 67ms, 최대 245ms (오류율 0%)로 모든 기준 충족 → **Pass**

구분	#Samples	평균(ms)	95p(ms)	최대(ms)	오류율	처리량
C-GET /api/logs	100	**67**	**182**	**245**	**0.00%**	**20.1/sec**
TOTAL	100	**67**	**182**	**245**	**0.00%**	**20.1/sec**

Non-Functional Test

비기능 테스트(NFT) 상세 결과

NFT-06: 비밀번호 저장 보안

대상: `admin_user.password_hash` 컬럼

기준: 평문 PW 미사용, SHA-256 해시(16진 64자) 형태 저장

결과: DB 조회 결과 해시 형태로 저장 확인 → Pass

NFT-07: 탐지율(오탐·미탐)

데이터:

정상 쿼리 1,000건 (SELECT, 50ms 간격)

공격 쿼리 50건 (명시적 DROP TABLE 포함)

기준: FPR ≤ 5%, FNR ≤ 2%

결과:

공격 50건 → TP=50, FN=0 → FNR=0%

정상 1000건 중 표본 100건 → FP=0 → FPR≈0%

NFT-07: Pass

NFT-08: 알림 성공률

데이터: 공격 100건 → PATTERN 이벤트 100건 발생

기준: Slack + Email 알림 성공률 ≥ 99%

결과:

Slack Webhook 메시지 100건 도착

Gmail 수신함에 이메일 100건 도착

NFT-08: Pass

Detailed Defect

결함 분석

1	**D-01:** 로그 아카이브 기능 미구현 • 영향 범위: UT-12, IT-08, ST-08 • 우선순위: High • 장기 로그 관리 및 스토리지 최적화 필요
2	**D-02:** 로그인 실패 처리 미구현 • 영향 범위: ST-07 • 우선순위: Medium • 사용자 인증 예외 처리 강화 필요
3	**D-03:** 알림 실패 처리 미구현 • 영향 범위: IT-06 • 우선순위: Medium • 알림 전송 실패 시 재시도 로직 추가 필요

Goal Achievement

목표 달성 여부

기능 요구사항 달성

단위, 통합, 시스템 테스트를 통해 **로그 아카이빙을 제외한 모든 핵심 기능**이 성공적으로 구현 및 검증됨

비기능 요구사항 충족

성능, 보안, 안정성 등 비기능 요구사항이 비기능 테스트를 통해 목표치를 충족됨을 검증

감사합니다