홍진수
이력서.pdf

Github: https://github.com/kaameo
Blog: https://kaameo.github.io
Email : kaameo12@gmail.com
Mobile : (+82) 010-9279-1819

1. DeFender X SIEM

- **소속**: (주)인브릿지 *2025. 06 - PRESENT*

주요 내용:

보안 벤더(PaloAlto, Cisco, CrowdStrike)의 침해 로그를 수집 · 정규화하고, 위협 탐지 · 대응을 자동화하는 SIEM 솔루션. 데이터 파이프라인 설계 · 구축 및 풀스택 개발 담당.

사용한 기술: Go, Java, TypeScript, Next.js, Prisma, OpenSearch, PostgreSQL, Kafka, Docker, Kubernetes

주요 성과:

- Go 기반 로그 수집기 단독 설계 · 개발: Python 대비 처리 속도 2.5배 향상, 메모리 66% 절감(150MB → 50MB)
- 멀티 벤더 수집 아키텍처: 3개 벤더 어댑터, DLQ 재처리 시스템, 핵심 모듈 테스트 커버리지 90%+
- Kafka 파티션 불균형(skewed) 해결: status 기반 key에서 고유 ID 기반으로 변경하여 컨슈머 지연 제거
- Kafka Connect Custom SMT 구현(Java): 도메인별 정규화 로직 및 OpenSearch 인덱스 매핑 자동화
- CI/CD 파이프라인 구축: 배포 성공률 100% 달성(4개월 유지), 배포 장애로 인한 야근 주 1회 → 월 0.5회
- 실시간 위협 대시보드 개발: 보안팀 인시던트 분석 · 대응 예상 시간 30분 → 10분 단축

○ 1.1 데이터 파이프라인 아키텍처 설계 및 구축

보안 벤더별 침해 로그를 수집 · 정규화하는 데이터 파이프라인 설계 및 구축.

성과:
- Python 대비 처리 속도 2.5배 향상(동일 1,000건 기준 10초→4초), 메모리 66% 절감(150MB→50MB)
- DLQ + 재시도 워커, Redis+File 이중 체크포인트로 장애 시 수집 중단/데이터 유실 방지 및 복구 가능
- /health, /metrics, /api/v1/status 제공으로 수집기 상태 관측 및 운영 제어 표준화

구현 내용:

1) 데이터 수집: 벤더별 API 특성 분석 후 modification_time 기반 증분 수집 패턴 설계. Redis+File 이중 체크포인트로 배포/재시작 시 수집 지점 복구.

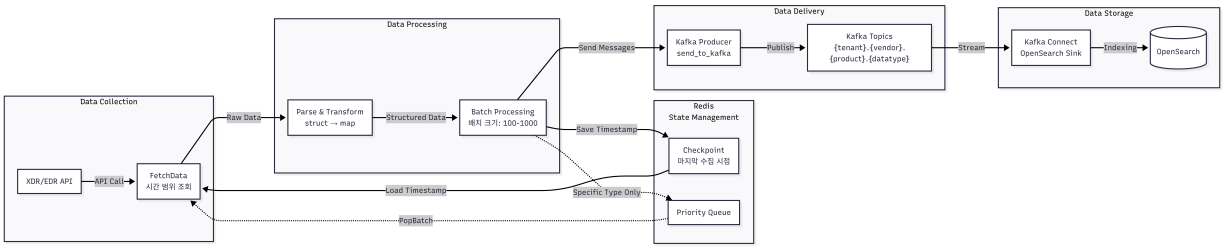
2) 메시징: Kafka 도입으로 수집기/처리기 분리. 벤더별 · 테넌트별 토픽 분리, 고유 ID 기반 파티션 키로 순서 보장과 병렬 처리 동시 달성. DLQ + 재시도 워커로 실패 격리.

3) 처리 및 적재: Kafka Connect + Custom SMT(Java)로 도메인별 정규화 및 OpenSearch 인덱스 매핑 자동화.

4) 운영/관측: /health, /metrics, /api/v1/status 제공으로 상태 점검 및 수집기 제어(enable/trigger/reset) 표준화.

5) Go 전환: Python GIL 한계 해소를 위해 Go로 마이그레이션. goroutine 기반 동시성 처리로 성능 개선.

6) 아키텍처:



사용 기술: Go, Java, Apache Kafka, OpenSearch, Redis

- **1.2 CI/CD 파이프라인 구축**
  수동 배포로 인한 장애를 제거하고 배포 안정성을 확보하기 위한 CI/CD 파이프라인 구축.

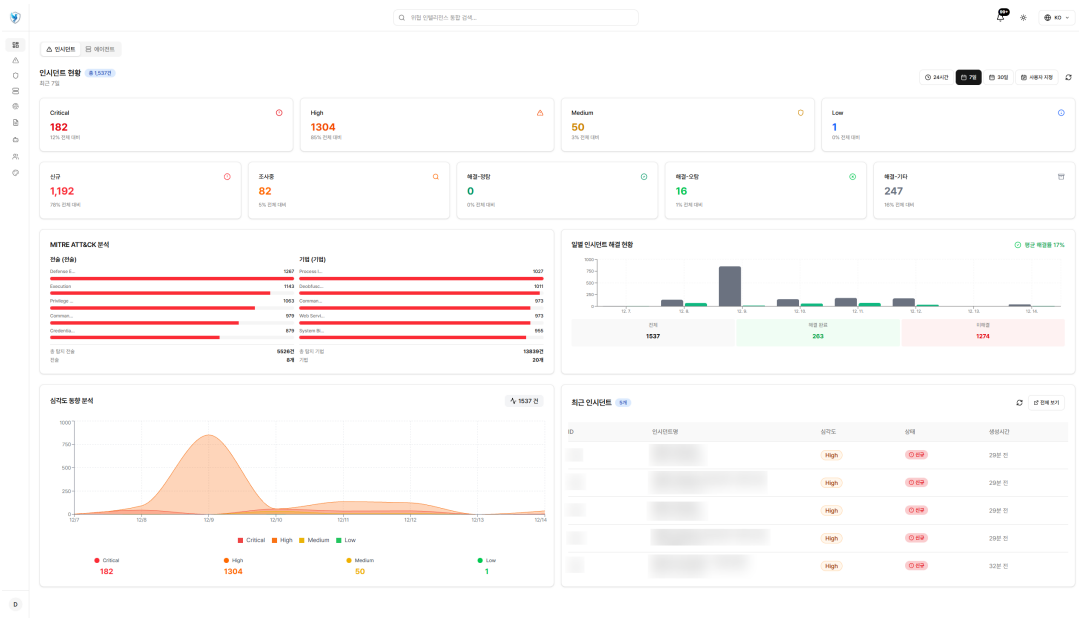
  성과:
  - 배포 성공률 100% 달성 및 4개월간 유지 (도입 전: 일 1회 배포 실패)
  - 배포 장애로 인한 야근 주 1회→월 0.5회로 감소
  - 배포 리드타임 10분→5분 단축 (50% 개선)

  구현 내용:
  1) 문제: 수작업 배포(로컬→머지→서버 재가동)로 일 1회 배포 실패 발생. Git 형상관리 부재로 롤백 불가.
  2) 해결: GitHub Actions 기반 자동 빌드 · 배포 파이프라인 구축. 인바운드 차단 환경을 고려해 self-hosted runner로 아웃바운드 중심 구조 설계.
  3) 테스트 자동화: Jest/Go test 통합, 핵심 모듈 커버리지 90%+ 유지. PR 머지 전 테스트 통과 필수화.
  사용 기술: GitHub Actions, Docker, Kubernetes, Jest, Go test

- **1.3 실시간 위협 대시보드 개발**
  보안 운영팀의 인시던트 분석 · 대응 시간 단축을 위한 통합 대시보드 개발.

  성과:
  - 보안팀 인시던트 분석 · 대응 예상 시간 30분→10분 단축 (67% 개선)
  - 멀티 벤더 보안 현황을 단일 뷰로 통합하여 콘솔 전환 시간 제거

  구현 내용:
  1) 문제: 벤더별 콘솔 개별 확인으로 위협 대응 시간 지연.
  2) 해결: Next.js App Router 기반 대시보드 구현. OpenSearch 집계 데이터 조회 API 설계, Prisma로 PostgreSQL 메타데이터 관리. 심각도/벤더/시간 범위 필터링 및 페이지네이션 적용.
  3) 대시보드 화면:



  사용 기술: Next.js, TypeScript, Prisma, OpenSearch, PostgreSQL

- **1.4 보고서 에디터 개발**
  보안 분석가용 리치 텍스트 에디터 개발. 라이선스 준수를 위한 자체 구현.

  구현 내용:
  1) 기술 선정: BlockNote vs Markdown 프로토타입 비교 후 사용자 피드백 기반으로 BlockNote 선정.

2) 라이선스 이슈 해결: BlockNote 확장 패키지(GPL 3.0)가 상용 프로젝트에 부적합하여, Apache 2.0 코어 기반으로 차트 블록 및 AI 에디터 기능 자체 구현.

사용 기술: Next.js, TypeScript, BlockNote, Recharts, OpenAI API

- **1.5 보안 취약점 선제 대응**

  Next.js 보안 권고 모니터링 중 CVE-2025-29927(CVSS 9.1)이 프로젝트 버전에 해당함을 확인. 취약점 공개 직후 패치 버전 즉시 배포 및 시크릿 로테이션 수행.

  사용 기술: Next.js, GitHub Actions

2. Teams Add-in 개발

- **소속**: (주)인브릿지 *2025.03 - 2025.05*

주요 내용:

Microsoft Teams 통화/미팅 자동 녹취 및 LLM 요약 Add-in. 중도 합류 후 협업 체계 구축 및 풀스택 개발 담당.

사용한 기술: ASP .NET, JavaScript, React, Node.js, Express, Supabase(PostgreSQL), Docker

주요 성과:

- Git 형상관리 도입: 프로덕션 직접 수정으로 인한 일일 장애 제거, PR 기반 코드 리뷰 및 롤백 체계 구축
- RFC 기반 도메인 모델 재설계: Caller/Callee 분리 구조를 user/counterpart + 방향 플래그로 단순화
- API 스펙 표준화(정렬, 페이지네이션) 주도로 프론트-백엔드 간 중복 구현 제거
- Docker 기반 개발 환경 표준화로 OS 종속성 제거 및 로컬 디버깅 환경 구축

- **2.1 Git 형상관리 도입 및 협업 체계 구축**

  프로덕션 직접 수정으로 인한 장애를 제거하고 협업 체계를 구축.

  성과:
  - 프로덕션 직접 수정으로 인한 일일 장애(코드 덮어쓰기, 충돌) 제거
  - PR 기반 코드 리뷰 및 롤백 가능한 배포 체계 구축

  구현 내용:

  1) 문제: pm2 환경에서 프로덕션 서버 직접 수정. 인원 증가 시 코드 덮어쓰기 및 충돌로 평균 일 3회 장애 발생. 롤백 불가.

  2) 해결: GitHub 도입, 브랜치 기반 협업 체계(기능 브랜치→코드 리뷰→머지) 구축. Docker 이미지 빌드 후 배포 방식으로 전환.

  사용 기술: Git, GitHub, Docker, Notion

- **2.2 도메인 모델 재설계 및 API 표준화**

  도메인 모델 복잡도 해소 및 API 스펙 표준화.

  구현 내용:

  1) 문제: Caller/Callee 개별 소유 구조로 권한 처리 및 조인 로직 복잡. 페이지네이션 방식 불일치.

  2) 해결: RFC 작성하여 user/counterpart + 방향 플래그 구조로 재설계. API 스펙 표준(정렬, 페이지네이션)을 팀 규약으로 통일.

  사용 기술: PostgreSQL, Node.js, Express, React