하이브리드 환경에서 DevSecOps 구축 및 공개형 Kubernetes 설정 점검 솔루션 개발

DevSecOps Implementation in Hybrid Environments and Development of an Open-Source Kubernetes Configuration Audit Solution

팀 명	Kubee
지도교수	유 승 재
팀 원	김 수 현 (팀장) 김 건 희 남 지 우 신 재 형 어 영 민 전 유 경

2025. 11. 4

중부대학교 미래융합공학부 정보보호학전공

목 차

1. 서론

1.1 연구 배경

Kubernetes는 컨테이너 오케스트레이션 플랫폼으로서 클라우드 네이티브 애플리케이션의 배포와 관리를 자동화하는 핵심 기술로 자리잡고 있다. 현재 전 세계적으로 수많은 기업과 조직이 Kubernetes를 기반으로 마이크로서비스 아키텍처를 구축하고 있으며, 이는 애플리케이션의 확장성과 유연성을 크게 향상시키고 있다. 그러나 이러한 급속한 도입과 확산과 함께 Kubernetes 클러스터의 보안 문제가 심각한 관심사로 대두되고 있다.

Kubernetes 클러스터는 복잡한 구성 요소들로 이루어져 있으며, 각 구성 요소 간의 상호작용은 다양한 보안 취약점을 야기할 수 있다. 특히 Pod, Service, Secret, ConfigMap 등의 리소스들이 서로 연결되어 하나의 통합된 시스템을 구성하는 과정에서 발생하는 보안 설정 오류나 권한 관리 문제는 공격자에게 클러스터 전체를 장악할 수 있는 기회를 제공한다. 이러한 보안 취약점들은 단순한 설정 오류에서부터 복잡한 권한 상승 공격에 이르기까지 다양한 형태로 나타나며, 그 피해 규모도 개별 서비스의 장애에서부터 전체 클러스터의 완전한 장악에 이르기까지 광범위하다.

기존의 보안 도구들은 주로 정적 분석이나 단순한 규칙 기반 검사를 통해 이러한 취약점을 탐지하고 있다. 그러나 Kubernetes의 복잡성과 동적 특성을 고려할 때, 이러한 전통적인 접근 방식만으로는 모든 보안 위협을 효과적으로 탐지하고 대응하기 어렵다. 특히 최근에는 AI 기술의 발전과 함께 보안 분석 분야에서도 지능형 분석 도구의 필요성이 대두되고 있으며, 이는 단순한 패턴 매칭을 넘어서는 고도화된 보안 분석 능력을 요구하고 있다.

1.2 연구 필요성

Kubernetes 클러스터의 보안 위협이 지속적으로 증가하고 있는 상황에서, 효과적인 보안 분석 도구의 개발은 매우 시급한 과제이다. 현재 대부분의 기업과 조직은 Kubernetes 클러스터를 운영하고 있지만, 이에 대한 체계적인 보안 관리 방안을 갖추지 못한 상태이다. 특히 개발팀과 운영팀 간의 역할 분담이 명확하지 않거나, 보안 정책이 일관되게 적용되지 않는 경우가 많아 보안 취약점이 지속적으로 발생하고 있다.

기존의 보안 스캔 도구들은 주로 CLI 기반의 단순한 도구들이 대부분이며, 이는 보안 전문가가 아닌 일반 개발자나 운영자들이 사용하기에는 접근성이 떨어진다. 또한 이러한 도구들은 단순한 규칙 기반 검사에 의존하고 있어, 복잡한 보안 시나리오나 새로운 형태의 공격 패턴을 탐지하기 어렵다. 더욱이 발견된 취약점에 대한 구체적인 대응 방안이나 우선순위 제안을 제공하지 못하는 경우가 많아, 실제 보안 개선에 기여하는 효과가 제한적이다.

최근 AI 기술의 발전과 함께 RAG(Retrieval-Augmented Generation) 기반의 지능형 분석 시스템이 주목받고 있다. 이러한 시스템은 대규모 지식베이스를 활용하여 복잡한 보안 문제를 분석하고, 구체적인 대응 방안을 제안할 수 있는 능력을 제공한다. Kubernetes 보안 분야에서도 이러한 AI 기반 분석 시스템의 도입이 필요하며, 이를 통해 보안 전문가의 지

식을 자동화하고 일반 사용자도 쉽게 사용할 수 있는 지능형 보안 분석 도구를 개발할 수 있다.

1.3 연구 목적 및 주제선정 이유

본 연구의 목적은 Kubernetes 클러스터의 보안 취약점을 자동으로 탐지하고 분석할 수 있는 지능형 보안 스캔 솔루션을 개발하는 것이다. 이를 통해 기존의 단순한 규칙 기반 검사를 넘어서는 고도화된 보안 분석 능력을 제공하고, 발견된 취약점에 대한 구체적인 대응 방안과 우선순위를 제안할 수 있는 시스템을 구축하고자 한다. 또한 AI 기술을 활용하여 보안 전문가의 지식을 자동화하고, 일반 사용자도 쉽게 사용할 수 있는 사용자 친화적인 인터페이스를 제공하는 것을 목표로 한다.

구체적인 연구 목표는 다음과 같다. 첫째, Kubernetes 클러스터의 다양한 보안 취약점을 탐지할 수 있는 포괄적인 스캔 엔진을 개발한다. 이를 위해 Pod Security Standards, Network Policies, RBAC 등의 보안 정책을 분석하고, 각 정책 위반 사항을 자동으로 탐지할 수 있는 규칙 엔진을 구축한다. 둘째, RAG 기반의 AI 분석 시스템을 개발하여 발견된 취약점에 대한 지능형 분석과 대응 방안을 제안한다. 이를 위해 Kubernetes 보안 지식베이스를 구축하고, 자연어 처리 기술을 활용하여 사용자 질의에 대한 답변을 제공한다. 셋째, 웹 기반의 사용자 인터페이스를 개발하여 보안 전문가가 아닌 일반 사용자도 쉽게 사용할 수 있는 접근성을 제공한다. 이를 위해 직관적인 대시보드와 상세한 분석 결과 시각화 기능을 구현한다.

본 연구에서 선택한 주제는 "Kubee Web – Kubernetes 보안 스캔 플랫폼"으로, 이는 현재 가장 시급한 보안 문제 중 하나인 Kubernetes 클러스터 보안을 다루고 있다. 이 주제를 선택한 이유는 첫째, Kubernetes의 급속한 도입과 확산으로 인한 보안 위협의 증가, 둘째, 기존 보안 도구의 한계와 지능형 분석 도구의 필요성, 셋째, AI 기술을 활용한 혁신적인 보안 솔루션 개발의 가능성 때문이다. 이를 통해 본 연구는 Kubernetes 보안 분야의 발전에 기여하고, 실제 산업 현장에서 활용 가능한 실용적인 솔루션을 제공할 수 있을 것으로 기대한다.

2. 관련 연구

2.1 Kubernetes 보안 연구 동향

Pod Security Standards(PSS)는 Kubernetes 1.23부터 도입된 보안 정책 프레임워크로, Pod의 보안 설정을 표준화하고 강제하는 메커니즘을 제공한다. PSS는 세 가지 수준의 정책을 정의하는데, Privileged는 가장 관대한 정책으로 대부분의 보안 제약을 완화하여 호스트 네임스페이스 사용, privileged 컨테이너 실행, 호스트 파일시스템 마운트 등을 허용한다. Baseline은 일반적인 운영 환경에 적합한 최소한의 보안 제약을 제공하여 privileged 컨테이너 사용을 금지하고, 호스트 네임스페이스 사용을 제한하며, 기본적인 보안 컨텍스트 설정을 강제한다. Restricted는 가장 엄격한 정책으로 최대한의 보안을 보장하여 모든 privileged 기능을 금지하고, readOnlyRootFilesystem을 강제하며, runAsNonRoot를 요구한다.
PSS의 도입으로 기존의 Pod Security Policies(PSP)의 복잡성과 관리 부담을 해결하면서도, 더 직관적이고 표준화된 보안 정책 관리가 가능해졌다. 이는 Kubernetes 클러스터의 보안 수준을 일관되게 유지하는 데 중요한 역할을 하며, 개발자들이 보안 모범 사례를 쉽게 적용할 수 있도록 돕는다. 또한 PSS는 네임스페이스별로 다른 보안 정책을 적용할 수 있어, 다양한 워크로드의 보안 요구사항을 유연하게 처리할 수 있다.

Network Policies는 Kubernetes에서 네트워크 트래픽을 제어하는 리소스로, Pod 간 통신을 세밀하게 관리할 수 있게 해준다. 기본적으로 Kubernetes는 모든 Pod 간의 통신을 허용하는데, Network Policies를 통해 이를 제한할 수 있다. Network Policies는 Pod Selector를 통해 정책이 적용될 Pod를 선택하고, 라벨 셀렉터를 사용하여 특정 Pod들을 그룹화할 수 있다. Ingress Rules는 들어오는 트래픽을 제어하여 특정 네임스페이스나 Pod에서 오는 트래픽만 허용하거나, 특정 포트로의 접근만 허용할 수 있다. Egress Rules는 나가는 트래픽을 제어하여 특정 목적지로의 트래픽만 허용하거나, 특정 포트로의 접근만 허용할 수 있다.
Network Policies의 효과적인 활용을 위해서는 클러스터의 네트워크 플러그인이 이를 지원해야 한다. CNI(Container Network Interface) 플러그인 중에서 Calico, Cilium, Weave Net 등이 Network Policies를 지원한다. Network Policies는 마이크로서비스 아키텍처에서 서비스 간 통신을 제어하는 데 특히 유용하며, 공격 표면을 최소화하고 방어선을 구축하는 데 중요한 역할을 한다. 또한 Network Policies는 네트워크 세분화를 통해 보안 사고 발생 시 피해 범위를 제한할 수 있게 해준다.

RBAC는 Kubernetes의 인증 및 권한 관리 시스템으로, 사용자와 서비스 계정의 클러스터 리소스 접근 권한을 세밀하게 제어할 수 있게 해준다. RBAC는 역할(Role)과 역할 바인딩(RoleBinding)을 통해 권한을 관리하는데, Role은 특정 네임스페이스 내에서의 권한을 정의하고, ClusterRole은 클러스터 전체에 대한 권한을 정의한다. 각 역할은 리소스와 동작(verb)의 조합으로 권한을 명시하며, RoleBinding은 Role을 특정 사용자나 그룹에 바인딩하고, ClusterRoleBinding은 ClusterRole을 클러스터 전체에 바인딩한다.
서비스 계정은 Pod가 클러스터 API와 상호작용할 때 사용하는 계정으로, 적절한 권한이

부여되어야 한다. 서비스 계정의 권한이 과도하게 부여되면 보안 위험을 초래할 수 있으므로, 최소 권한 원칙을 적용하고 정기적인 권한 검토를 통해 불필요한 권한을 제거해야 한다. RBAC의 효과적인 활용을 위해서는 조직의 역할과 책임을 명확히 정의하고, 각 역할에 필요한 최소한의 권한만 부여해야 한다. 또한 권한 변경 시에는 변경 사항을 추적하고, 정기적인 권한 감사를 통해 보안 정책 준수를 확인해야 한다.

2.2 기존 보안 스캔 도구 분석

kube-score는 Kubernetes 매니페스트 파일의 보안성과 모범 사례 준수를 검사하는 오픈소스 도구이다. 이 도구는 정적 분석을 통해 매니페스트 파일의 잠재적인 문제점을 식별하며, 보안 설정 검사, 리소스 관리 검사, 네트워크 보안 검사, 이미지 보안 검사 등의 기능을 제공한다. kube-score는 privileged 컨테이너, hostNetwork 사용 등의 보안 설정을 검사하고, CPU/메모리 제한 설정, 리소스 요청 등의 리소스 관리 문제를 식별한다. 또한 불필요한 포트 노출, 서비스 설정 등의 네트워크 보안 문제와 latest 태그 사용, 이미지 풀 정책 등의 이미지 보안 문제를 탐지한다.
kube-score의 장점은 간단한 CLI 도구로 사용하기 쉽고, 다양한 보안 규칙을 제공하며, CI/CD 파이프라인에 쉽게 통합할 수 있다는 점이다. 그러나 정적 분석에만 의존하여 런타임 보안 문제는 탐지하지 못하고, AI 기반 분석이나 지능형 권장사항 제공 기능이 없으며, 사용자 인터페이스가 없어 비개발자들이 사용하기 어렵다는 한계가 있다. 이러한 한계를 극복하기 위해서는 런타임 분석 기능의 추가, AI 기반 분석 기능의 도입, 사용자 친화적인 인터페이스의 제공 등이 필요하다.

kube-hunter는 Kubernetes 클러스터의 보안 취약점을 탐지하는 도구로, 공격자 관점에서 클러스터를 스캔한다. 이 도구는 클러스터의 다양한 구성 요소를 대상으로 보안 검사를 수행하며, API 서버, etcd, kubelet 등의 클러스터 구성 요소를 스캔하고, 노출된 서비스, 포트 스캔 등의 네트워크 서비스를 검사한다. 또한 과도한 권한, 취약한 설정 등의 권한 문제를 검사하고, 공격 시나리오를 시뮬레이션하여 실제 공격 가능성을 평가한다.
kube-hunter의 장점은 공격자 관점에서의 보안 검사를 통해 실제 공격 벡터를 탐지할 수 있고, 다양한 공격 시나리오를 시뮬레이션하여 상세한 취약점 보고서를 제공한다는 점이다. 그러나 클러스터에 대한 직접적인 접근이 필요하고, Git 저장소 기반의 정적 분석은 지원하지 않으며, AI 기반 분석이나 지능형 권장사항 제공 기능이 없다는 한계가 있다. 이러한 한계를 극복하기 위해서는 정적 분석 기능의 추가, AI 기반 분석 기능의 도입, 사용자 친화적인 인터페이스의 제공 등이 필요하다.

Polaris는 Kubernetes 클러스터의 구성과 배포를 실시간으로 모니터링하고 분석하는 도구이다. 이 도구는 클러스터의 보안성과 모범 사례 준수를 지속적으로 검사하며, 실시간 클러스터 모니터링, 보안 정책 검사, 리소스 사용량 분석, 웹 대시보드 제공 등의 기능을 제공한다. Polaris는 클러스터의 상태를 지속적으로 모니터링하여 보안 정책 위반을 실시간으로 탐지하고, 리소스 사용량을 분석하여 비효율적인 리소스 사용을 식별한다.
Polaris의 장점은 실시간 모니터링 기능을 통해 보안 문제를 즉시 탐지할 수 있고, 직관적인 웹 인터페이스를 통해 클러스터 상태를 쉽게 파악할 수 있으며, 지속적인 보안 검사를 통해 보안 수준을 유지할 수 있다는 점이다. 그러나 클러스터에 대한 직접적인 접근이 필

요하고, Git 저장소 기반의 정적 분석은 지원하지 않으며, AI 기반 분석이나 지능형 권장사항 제공 기능이 없다는 한계가 있다. 이러한 한계를 극복하기 위해서는 정적 분석 기능의 추가, AI 기반 분석 기능의 도입, 사용자 친화적인 인터페이스의 제공 등이 필요하다.

2.3 AI 기반 보안 분석 기술

RAG는 검색 증강 생성 모델로, 대규모 언어 모델의 지식 한계를 극복하기 위해 외부 지식베이스를 활용하는 기술이다. RAG는 검색(Retrieval)과 생성(Generation) 두 단계로 구성되는데, 검색 단계에서는 사용자의 질문이나 입력을 벡터화하여 관련 문서를 지식베이스에서 검색한다. 이 과정에서 FAISS, Pinecone 등의 벡터 데이터베이스를 사용하여 빠르고 정확한 검색을 수행한다. 생성 단계에서는 검색된 문서를 컨텍스트로 활용하여 대규모 언어 모델이 최종 답변을 생성한다. 이 과정에서 검색된 정보와 모델의 기존 지식을 결합하여 더 정확하고 관련성 높은 답변을 제공한다.
RAG의 장점은 최신 정보를 반영할 수 있고, 관련 문서를 참조하여 더 정확한 답변을 제공하며, 답변의 근거가 되는 문서를 제시할 수 있어 투명성이 높다는 점이다. 또한 RAG는 도메인 특화 지식베이스를 구축하여 전문적인 질문에 대해 더 정확한 답변을 제공할 수 있다. 그러나 RAG의 효과적인 활용을 위해서는 고품질의 지식베이스 구축, 적절한 검색 전략 수립, 생성 모델의 품질 관리 등이 필요하다. 이러한 요소들이 제대로 관리되지 않으면 검색된 정보의 품질이 떨어지거나, 생성된 답변의 정확성이 저하될 수 있다.

자연어 처리(NLP) 기술을 활용한 보안 분석은 보안 전문가의 지식을 자동화하고, 복잡한 보안 문제를 이해하기 쉽게 설명하는 데 활용된다. 텍스트 분류는 보안 관련 텍스트를 자동으로 분류하여 위험도를 평가하는 기술로, 로그 메시지나 에러 메시지를 분석하여 보안 위협을 식별할 수 있다. 개체명 인식은 보안 관련 개체(예: IP 주소, 포트 번호, 파일 경로 등)를 자동으로 식별하고 추출하는 기술로, 보안 이벤트의 핵심 요소를 빠르게 파악할 수 있게 해준다. 감정 분석은 보안 관련 텍스트의 감정을 분석하여 위험도를 평가하는 기술로, 공격자의 의도를 파악하거나 사용자의 보안 인식을 평가할 수 있다.
요약 및 생성 기술은 복잡한 보안 보고서를 요약하거나, 보안 권장사항을 자동으로 생성하는 기술로, 보안 전문가의 업무 효율성을 높이는 데 기여한다. 이러한 기술들은 보안 분석의 자동화와 표준화를 통해 보안 전문가의 업무 부담을 줄이고, 보안 분석의 정확성과 일관성을 높이는 데 기여한다. 그러나 자연어 처리 기반 보안 분석의 효과적인 활용을 위해서는 고품질의 학습 데이터, 적절한 모델 선택, 지속적인 모델 개선 등이 필요하다. 이러한 요소들이 제대로 관리되지 않으면 분석 결과의 정확성이 떨어지거나, 오탐지나 미탐지가 발생할 수 있다.

AI 기반 보안 분석을 위해서는 포괄적이고 정확한 지식베이스가 필요하다. 지식베이스 구축 방법론은 데이터 수집, 데이터 전처리, 벡터화, 인덱싱, 품질 관리의 단계로 구성된다. 데이터 수집 단계에서는 다양한 소스에서 보안 관련 정보를 수집하는데, 공식 문서, 보안 가이드, 모범 사례, 취약점 데이터베이스 등을 포함한다. 데이터 전처리 단계에서는 수집된 데이터를 정제하고 구조화하는데, 중복 제거, 형식 통일, 메타데이터 추가 등의 작업을 수행한다. 벡터화 단계에서는 텍스트 데이터를 벡터로 변환하여 유사도 검색이 가능하도록 하는데, sentence-transformers, OpenAI Embeddings 등을 사용한다.

인덱싱 단계에서는 벡터화된 데이터를 벡터 데이터베이스에 저장하여 빠른 검색이 가능하도록 하는데, FAISS, Pinecone, Weaviate 등을 사용한다. 품질 관리 단계에서는 지식베이스의 품질을 지속적으로 관리하고 개선하는데, 정확성 검증, 업데이트, 피드백 반영 등의 작업을 수행한다. 지식베이스 구축의 성공을 위해서는 도메인 전문가와의 협력, 지속적인 업데이트, 사용자 피드백 반영 등이 중요하다. 이러한 요소들이 제대로 관리되지 않으면 지식베이스의 품질이 떨어지거나, 사용자에게 유용한 정보를 제공하지 못할 수 있다.

3. 본론

3.1 시스템 구성

Kubee Web은 Kubernetes 보안 스캔을 위한 종합적인 플랫폼으로, Frontend, Backend, AI 서비스의 3개 주요 구성 요소로 이루어져 있다. 각 구성 요소는 독립적으로 동작하면서도 유기적으로 연결되어 하나의 통합된 시스템을 구성하며, 이를 통해 사용자에게 포괄적인 보안 분석 서비스를 제공한다.

Frontend 시스템은 React 19.1.1과 TypeScript 4.9.5를 기반으로 구축된 사용자 인터페이스로, Material-UI 7.3.2 컴포넌트 라이브러리를 활용하여 현대적이고 직관적인 UI를 제공한다. Frontend는 대시보드, 스캔 폼, 결과 분석, AI 분석, 사용자 관리, 설정 등의 주요 기능을 포함하고 있으며, 각 기능은 독립적인 컴포넌트로 구현되어 유지보수성과 확장성을 확보하였다. 또한 Axios를 통한 HTTP 클라이언트, React Router를 통한 라우팅, Recharts를 통한 데이터 시각화 등의 기능을 통합하여 사용자에게 풍부한 인터랙션을 제공한다.

Backend 시스템은 Flask 3.0.0을 기반으로 구축된 API 서버로, RESTful API를 통해 Frontend와 AI 서비스 간의 통신을 담당한다. Backend는 인증, 스캔, 사용자 관리, 헬스 체크, AI 분석, 로그 관리 등의 주요 기능을 제공하며, 각 기능은 독립적인 컨트롤러로 구현되어 모듈화된 구조를 갖추고 있다. SQLite 데이터베이스를 사용하여 사용자 정보와 스캔 결과를 저장하며, JWT 기반 인증 시스템을 통해 보안을 강화하였다. 또한 bcrypt를 사용한 비밀번호 해싱, CORS 설정을 통한 크로스 오리진 요청 처리 등의 보안 기능을 포함하고 있다.

AI 서비스는 LangChain과 OpenAI GPT를 기반으로 구축된 지능형 분석 시스템으로, RAG(Retrieval-Augmented Generation) 기술을 활용하여 Kubernetes 보안 지식베이스 기반의 질의응답과 취약점 분석을 제공한다. AI 서비스는 sentence-transformers를 사용한 텍스트 임베딩, FAISS를 사용한 벡터 데이터베이스, langchain-text-splitters를 사용한 텍스트 처리 등의 기능을 포함하고 있으며, 이를 통해 사용자의 질문에 대한 정확한 답변과 발견된 취약점에 대한 구체적인 분석을 제공한다.

3.2 개발 시스템

Kubee Web의 개발 환경은 Python 3.8+와 Node.js 16+를 기반으로 구성되었으며, 각 구성 요소별로 독립적인 개발 환경을 제공하면서도 통합된 개발 워크플로우를 지원한다. 개발 환경은 로컬 개발, 테스트, 배포의 3단계로 구성되어 있으며, 각 단계에서 일관된 환경을 제공하여 개발 효율성을 극대화한다.

Python 개발 환경은 가상환경을 사용하여 프로젝트별 의존성을 격리하고, requirements.txt와 ai_requirements.txt를 통해 의존성을 관리한다. 주요 의존성으로는 Flask 3.0.0, SQLite, JWT, bcrypt, GitPython, requests, langchain, openai, faiss-cpu 등이 포함되어 있으며, 각 의존성은 버전을 명시하여 호환성을 보장한다. 또한 pytest를 사용

한 단위 테스트, pytest-flask를 사용한 Flask 애플리케이션 테스트, pytest-cov를 사용한 테스트 커버리지 측정 등의 테스트 도구를 포함하고 있다.

Node.js 개발 환경은 npm을 사용하여 의존성을 관리하며, package.json을 통해 프로젝트 설정과 스크립트를 관리한다. 주요 의존성으로는 React 19.1.1, TypeScript 4.9.5, Material-UI 7.3.2, Axios, React Router, Recharts 등이 포함되어 있으며, 각 의존성은 최신 안정 버전을 사용하여 보안과 성능을 최적화한다. 또한 ESLint와 Prettier를 사용한 코드 품질 관리, Jest를 사용한 단위 테스트, React Testing Library를 사용한 컴포넌트 테스트 등의 개발 도구를 포함하고 있다.

통합 개발 환경은 Docker와 Docker Compose를 사용하여 전체 시스템을 통합적으로 관리하며, 개발, 테스트, 배포의 모든 단계에서 일관된 환경을 제공한다. Dockerfile을 통해 각 구성 요소별로 최적화된 이미지를 빌드하고, docker-compose.yml을 통해 전체 시스템을 통합적으로 실행한다. 또한 환경 변수를 통한 설정 관리, 로그 수집 및 모니터링, 자동화된 테스트 실행 등의 기능을 포함하고 있다.

3.3 스캔 엔진 개발

Kubee Web의 스캔 엔진은 Kubernetes 클러스터의 보안 취약점을 자동으로 탐지하는 핵심 구성 요소로, Git 저장소 분석, 매니페스트 파싱, 보안 규칙 검사 등의 기능을 제공한다. 스캔 엔진은 Python으로 구현되었으며, GitPython을 사용한 Git 저장소 클론 및 분석, PyYAML을 사용한 YAML 파일 파싱, 정규표현식을 사용한 패턴 매칭 등의 기술을 활용한다.

Git 저장소 분석은 스캔 엔진의 첫 번째 단계로, 사용자가 제공한 Git 저장소 URL을 통해 저장소를 클론하고 Kubernetes 매니페스트 파일을 식별한다. 이 과정에서 GitPython을 사용하여 저장소를 로컬에 클론하고, 파일 시스템을 탐색하여 YAML, JSON, Helm Charts, Kustomize 등의 Kubernetes 매니페스트 파일을 찾아낸다. 또한 각 파일의 메타데이터를 수집하여 파일 경로, 수정 시간, 크기 등의 정보를 저장한다.

매니페스트 파싱은 식별된 Kubernetes 매니페스트 파일을 파싱하여 구조화된 데이터로 변환하는 과정이다. 이 과정에서 PyYAML을 사용하여 YAML 파일을 파싱하고, JSON 파일은 내장 json 모듈을 사용하여 파싱한다. 파싱된 데이터는 Pod, Service, Secret, ConfigMap, Role, RoleBinding 등의 Kubernetes 리소스로 분류되며, 각 리소스의 속성과 설정을 추출하여 분석 가능한 형태로 변환한다.

보안 규칙 검사는 파싱된 Kubernetes 리소스를 대상으로 보안 규칙을 적용하여 취약점을 탐지하는 과정이다. 보안 규칙은 Pod Security Standards, Network Policies, RBAC, 컨테이너 보안, 이미지 보안, 시크릿 관리 등의 영역별로 정의되어 있으며, 각 규칙은 정규표현식, 패턴 매칭, 조건부 검사 등의 방법을 사용하여 구현된다. 검사 결과는 취약점 유형, 심각도, 설명, 대응 방안 등의 정보를 포함하여 구조화된 형태로 저장된다.

3.4 AI 분석 시스템 개발

Kubee Web의 AI 분석 시스템은 RAG(Retrieval-Augmented Generation) 기술을 기반으로 구축된 지능형 분석 시스템으로, Kubernetes 보안 지식베이스를 활용하여 사용자 질의에 대한 답변과 취약점 분석을 제공한다. AI 분석 시스템은 LangChain과 OpenAI GPT를 핵심 기술로 사용하며, sentence-transformers와 FAISS를 사용한 벡터 검색 기능을 포함하고 있다.

지식베이스 구축은 AI 분석 시스템의 핵심 구성 요소로, Kubernetes 보안 관련 포괄적인 정보를 수집하고 구조화하는 과정이다. 지식베이스는 공식 문서, 보안 가이드, 모범 사례, 취약점 데이터베이스 등의 다양한 소스에서 정보를 수집하며, 수집된 정보는 텍스트 분할, 벡터화, 인덱싱의 과정을 거쳐 검색 가능한 형태로 변환된다. 텍스트 분할은 langchain-text-splitters를 사용하여 수행되며, 벡터화는 sentence-transformers를 사용하여 수행된다. 인덱싱은 FAISS를 사용하여 수행되며, 이를 통해 빠르고 정확한 유사도 검색이 가능하다.

RAG 모델 구현은 검색 증강 생성을 통해 사용자 질의에 대한 정확한 답변을 생성하는 핵심 기능이다. RAG 모델은 검색 단계와 생성 단계로 구성되며, 검색 단계에서는 사용자 질의를 벡터화하여 관련 문서를 지식베이스에서 검색한다. 생성 단계에서는 검색된 문서를 컨텍스트로 활용하여 OpenAI GPT를 통해 최종 답변을 생성한다. 이를 통해 단순한 패턴 매칭을 넘어서는 고도화된 분석 능력을 제공할 수 있다.

취약점 분석 기능은 발견된 취약점에 대한 지능형 분석과 대응 방안을 제안하는 기능으로, 공격 시나리오 생성, 대응 방안 제안, 우선순위 제안 등의 기능을 포함한다. 공격 시나리오 생성은 발견된 취약점을 기반으로 구체적인 공격 방법을 설명하며, 대응 방안 제안은 취약점에 대한 구체적인 수정 방법을 제안한다. 우선순위 제안은 취약점의 심각도와 비즈니스 영향도를 분석하여 수정 우선순위를 제안한다.

3.5 웹 인터페이스 개발

Kubee Web의 웹 인터페이스는 React와 TypeScript를 기반으로 구축된 사용자 친화적인 인터페이스로, 직관적인 대시보드와 상세한 분석 결과 시각화 기능을 제공한다. 웹 인터페이스는 Material-UI 컴포넌트 라이브러리를 활용하여 현대적이고 일관된 디자인을 제공하며, 반응형 디자인을 통해 다양한 디바이스에서 최적화된 사용자 경험을 제공한다.

대시보드 구현은 웹 인터페이스의 핵심 기능으로, 클러스터의 전체 보안 상태를 한눈에 파악할 수 있는 통합 대시보드를 제공한다. 대시보드는 스캔 통계, 취약점 분포, 심각도별 현황, 최근 스캔 결과 등의 정보를 시각적으로 표시하며, Recharts를 사용한 차트와 그래프를 통해 데이터를 직관적으로 표현한다. 또한 실시간 업데이트 기능을 통해 스캔 진행 상황을 실시간으로 모니터링할 수 있다.

스캔 결과 시각화는 발견된 취약점에 대한 상세한 정보를 시각적으로 표현하는 기능으로,

취약점 목록, 상세 정보, 공격 시나리오, 대응 방안 등의 정보를 체계적으로 표시한다. 스캔 결과는 테이블, 카드, 모달 등의 다양한 UI 컴포넌트를 사용하여 표시되며, 필터링, 정렬, 검색 등의 기능을 통해 사용자가 원하는 정보를 쉽게 찾을 수 있다. 또한 CSV, JSON 등의 형식으로 결과를 내보낼 수 있는 기능을 제공한다.

사용자 관리 시스템은 사용자 인증, 권한 관리, 세션 관리 등의 기능을 제공하는 시스템으로, JWT 기반 인증을 통해 보안을 강화한다. 사용자 관리 시스템은 로그인, 회원가입, 로그아웃 등의 기본 기능을 제공하며, 역할 기반 접근 제어를 통해 Admin과 User의 권한을 구분한다. 또한 세션 관리 기능을 통해 자동 로그아웃, 세션 타임아웃 등의 보안 기능을 제공한다.

3.6 API 설계 및 구현

Kubee Web의 API는 RESTful 아키텍처를 기반으로 설계되었으며, 각 기능별로 독립적인 엔드포인트를 제공하여 모듈화된 구조를 갖추고 있다. API는 인증, 스캔, 사용자 관리, 헬스 체크, AI 분석, 로그 관리 등의 주요 기능을 제공하며, 각 기능은 독립적인 컨트롤러로 구현되어 유지보수성과 확장성을 확보하였다.

인증 API는 사용자 인증과 관련된 기능을 제공하는 API로, 로그인, 회원가입, 로그아웃 등의 엔드포인트를 포함한다. 인증 API는 JWT 기반 인증을 사용하여 보안을 강화하며, bcrypt를 사용한 비밀번호 해싱을 통해 사용자 정보를 안전하게 보호한다. 또한 CORS 설정을 통해 크로스 오리진 요청을 안전하게 처리한다.

스캔 API는 Kubernetes 클러스터 스캔과 관련된 기능을 제공하는 API로, 저장소 스캔, 스캔 결과 조회, 스캔 로그 조회 등의 엔드포인트를 포함한다. 스캔 API는 비동기 처리를 통해 대용량 스캔 작업을 효율적으로 처리하며, 실시간 스캔 상태 업데이트를 통해 사용자에게 진행 상황을 제공한다. 또한 스캔 결과는 구조화된 JSON 형태로 저장되어 후속 분석에 활용할 수 있다.

AI 분석 API는 AI 기반 분석과 관련된 기능을 제공하는 API로, AI 채팅, 스캔 결과 분석, AI 상태 확인 등의 엔드포인트를 포함한다. AI 분석 API는 RAG 기술을 활용하여 사용자 질의에 대한 정확한 답변을 제공하며, 발견된 취약점에 대한 구체적인 분석과 대응 방안을 제안한다. 또한 AI 서비스의 상태를 모니터링하여 서비스 가용성을 보장한다.
사용자 관리 API는 사용자 관리와 관련된 기능을 제공하는 API로, 사용자 목록 조회, 사용자 생성, 사용자 삭제 등의 엔드포인트를 포함한다. 사용자 관리 API는 Admin 권한을 가진 사용자만 접근할 수 있으며, 역할 기반 접근 제어를 통해 보안을 강화한다. 또한 사용자 정보는 SQLite 데이터베이스에 안전하게 저장된다.

헬스 체크 API는 시스템 상태 모니터링과 관련된 기능을 제공하는 API로, 전체 시스템 상태, 데이터베이스 상태, 스캔 결과 디렉토리 상태 등의 정보를 제공한다. 헬스 체크 API는 시스템의 가용성을 모니터링하여 장애 상황을 조기에 감지할 수 있게 해주며, 로드 밸런서나 모니터링 시스템에서 활용할 수 있다.

로그 관리 API는 로그 수집과 관리와 관련된 기능을 제공하는 API로, 로그 조회, 로그 카테고리 조회 등의 엔드포인트를 포함한다. 로그 관리 API는 구조화된 로깅을 통해 시스템 동작을 추적할 수 있게 해주며, 보안 사고 발생 시 원인 분석에 활용할 수 있다.

4. 결과물 (Results)

4.1 오픈소스 대상 취약점 분석 결과

Kubee Web 솔루션을 통해 실제 운영 환경에서 발견된 취약점들을 분석한 결과, 다양한 심각도의 보안 위협이 존재함을 확인하였다. Critical 등급의 취약점들은 공격자가 클러스터 전체를 장악할 수 있는 직접적인 경로를 제공하는 가장 위험한 보안 위협으로, volcano-agent의 과도한 Secret 접근 권한과 kubeskoop의 privileged 컨테이너 설정이 대표적인 사례이다. 이러한 취약점들은 최소 권한 원칙을 심각하게 위반하며, 즉시 수정이 필요한 상황이다.

4.2 volcano-agent Secret 접근 권한 취약점 분석

volcano-agent Daemonset에서 발견된 과도한 Secret 접근 권한 취약점은 Critical 등급으로 분류되었으며, 이는 공격자가 클러스터 전체를 장악할 수 있는 직접적인 경로를 제공하는 가장 위험한 취약점 중 하나이다. volcano-agent의 ClusterRole에서 apiGroups: [""], resources: ["secrets"], verbs: ["get", "list", "watch"] 규칙이 설정되어 있어, 해당 Daemonset이 클러스터의 모든 Secret에 접근할 수 있는 권한을 가지고 있었다. 이는 volcano-agent의 실제 기능과 무관하게 부여된 과잉 권한으로, 최소 권한 원칙을 심각하게 위반하는 것으로 판단되었다. 공격자가 volcano-agent 컨테이너의 취약점을 악용하여 제어권을 획득한다면, 공격자는 volcano-agent가 가진 권한을 그대로 물려받게 된다. 이를 통해 공격자는 클러스터 내의 모든 Secret을 조회하고 탈취할 수 있으며, 다른 Account의 Service Account Token을 획득하여 해당 계정이 부여된 워크로드에 접근하거나, 더 높은 권한으로 Kubernetes API 서버에 접근하는 권한 상승 공격이 가능해진다. 대응 방안으로는 volcano-agent의 ClusterRole에서 Secret 접근 권한 규칙을 완전히 삭제하는 조치가 취해졌다. 이는 volcano-agent가 Secret 리소스에 접근할 필요가 없음을 확인한 후 최소 권한 원칙을 적용한 대표적인 사례이다.

4.3 kubeskoop 컨테이너 보안 취약점 분석

kubeskoop Daemonset에서 발견된 컨테이너 보안 취약점은 Critical 등급으로 분류되었으며, 이는 컨테이너의 격리 환경을 완전히 무력화시켜 공격자가 호스트 시스템 전체를 장악할 수 있게 만드는 심각한 취약점이다. kubeskoop의 Security Context에서 privileged: true 설정을 사용하여 컨테이너에 호스트의 모든 장치에 대한 접근 권한을 부여하고 있었다. 이는 컨테이너의 격리 환경을 완전히 무력화시키며, 공격자가 컨테이너 내부에서 호스트의 디스크를 마운트하여 호스트 파일 시스템에 접근하거나, 커널 파라미터를 조작하여 노드 자체를 제어할 수 있게 만든다. 또한 hostIPC: true 설정을 사용하여 컨테이너가 호스트의 IPC 네임스페이스를 공유하게 하여, 호스트의 다른 프로세스와 통신하거나 민감한 정보를 탈취할 수 있는 경로를 제공하였다. 공격자가 kubeskoop 컨테이너를 탈취한다면, privileged 권한을 통해 호스트 시스템의 모든 리소스에 접근할 수 있게 된다. 이를 통해 공격자는 호스트의 디스크를 마운트하여 민감한 파일에 접근하거나, 다른 컨테이너의 프로세스를 조작하여 추가적인 공격을 수행할 수 있다. 대응 방안으로는 kubeskoop의 Security Context를 완전히 재설계하는 조치가 취해졌다. privileged: true 설정을 제거하고

securityContext.capabilities를 사용하여 필요한 권한만 최소한으로 추가하였다. drop: ["ALL"]로 먼저 모든 기본 권한을 제거하고, add: ["NET_ADMIN", "SYS_PTRACE", "SYS_RESOURCE", "DAC_OVERRIDE"]를 통해 eBPF 및 프로세스 추적에 필요한 특정 권한만 명시적으로 추가하였다.

4.4 kubeskoop 보안 취약점 수정 과정 및 결과

kubeskoop 프로젝트에서 발견된 보안 취약점에 대한 실제 수정 과정이 GitHub에서 확인되었다. GitHub PR #339에서 "minimize privilege requirements & change helm image version to v1.0.0"라는 제목의 커밋이 2025년 5월 12일에 병합되었다. 이 커밋은 우리가 분석한 kubeskoop의 Critical 등급 취약점들을 해결한 것으로, 실제 보안 개선이 이루어진 것을 확인할 수 있다. 커밋 67a1b91에서는 privileged: true 설정을 제거하고 securityContext.capabilities를 사용하여 필요한 권한만 최소한으로 추가하는 방식으로 권한을 최소화하였다. 또한 hostIPC: true 설정을 제거하여 호스트의 IPC 네임스페이스 공유를 비활성화하고, automountServiceAccountToken: false 설정을 추가하여 불필요한 Kubernetes API 접근 토큰이 생성되지 않도록 설정하였다. 이는 Container Escape 위험을 제거하고, 호스트 네임스페이스 격리를 강화하며, 최소 권한 원칙을 적용한 대표적인 보안 개선 사례이다. 또한 Helm 이미지 버전을 v1.0.0으로 업데이트하여 프로젝트의 안정성을 향상시켰다. 이 수정 과정은 우리 솔루션의 분석이 실제 보안 개선에 기여했음을 보여주는 구체적인 사례로, 분석 결과의 실용성과 효과를 입증한다.

4.5 krkn Chaos Toolkit 보안 취약점 분석

krkn Chaos Toolkit은 Kubernetes 클러스터에서 카오스 엔지니어링 실험을 수행하기 위한 도구로, 여러 템플릿과 RBAC 설정에서 심각한 보안 취약점이 발견되었다. 특히 테스트 목적으로 제공되는 기본 설정들이 프로덕션 환경에서 오남용될 경우 클러스터 전체의 보안을 위협할 수 있는 위험한 구성으로 되어 있다. litmus-sa ServiceAccount가 ClusterRole을 통해 과도한 권한을 부여받고 있으며, pods, events, jobs, chaosengines, chaosexperiments, chaosresults, nodes 등의 리소스에 대한 create, list, get, patch, update, delete 권한이 포함되어 있다. 특히 patch 권한은 리소스의 부분적 수정을 허용하여 환경 변수 주입, 컨테이너 이미지 변경, 설정 조작 등의 공격이 가능하다. 여러 Pod 템플릿에서 hostNetwork: true와 privileged: true 설정이 기본값으로 사용되고 있으며, container_scenario_pod.yaml, outage_pod.yaml, time_pod.yaml에서는 hostNetwork: true가, volume_scenario.yaml에서는 privileged: true와 hostNetwork: true가 모두 설정되어 있다. 공격자가 litmus-sa ServiceAccount를 탈취할 경우, 클러스터의 모든 Pod, Job, Node에 대한 완전한 제어 권한을 획득할 수 있다. patch 권한을 통해 기존 Pod의 설정을 조작하여 악성 컨테이너를 주입하거나, 환경 변수를 변경하여 추가적인 공격을 수행할 수 있다. hostNetwork: true 설정으로 인해 Pod가 호스트의 네트워크 네임스페이스를 공유하게 되어, 공격자가 호스트 IP와 포트에 바인딩할 수 있다. 이를 통해 호스트 레벨의 네트워크 트래픽을 스니핑하거나 가로채기, 클러스터 내부의 다른 서비스로의 측면 이동이 가능하다. privileged: true 설정으로 인해 컨테이너가 호스트 커널, 장치, 네트워크 스택에 대한 거의 root 수준의 접근 권한을 갖게 되어, 컨테이너 탈출, 직접적인 노드 장악, 커널 조작 등의 공격이 가능하다.

4.6 취약점 통계 및 트렌드 분석

전체 취약점을 유형별로 분석한 결과, 권한 및 RBAC 취약점이 가장 높은 비율을 차지하였다. 이는 Kubernetes 클러스터에서 권한 관리가 가장 복잡하고 중요한 보안 요소임을 보여준다. 컨테이너 보안 취약점이 두 번째로 높은 비율을 차지하였으며, 이는 컨테이너의 격리 환경 설정이 여전히 어려운 과제임을 나타낸다. 네트워크 보안 취약점, 이미지 보안 취약점, 시크릿 관리 취약점이 그 뒤를 이었다. 권한 및 RBAC 취약점 중에서는 과도한 Secret 접근 권한이 가장 높은 비율을 차지하였으며, 이는 Secret이 가장 민감한 리소스임에도 불구하고 과도한 권한이 부여되는 경우가 많음을 보여준다. 컨테이너 보안 취약점 중에서는 privileged 컨테이너가 가장 높은 비율을 차지하였으며, 이는 컨테이너의 격리 환경을 무력화시키는 가장 위험한 설정임을 보여준다. 전체 취약점을 심각도별로 분석한 결과, Critical 등급이 소수이지만 가장 위험한 취약점들이며, High 등급의 취약점들이 상당한 보안 위험을 초래할 수 있음을 보여준다. Medium과 Low 등급의 취약점들이 대부분을 차지하지만, 이들도 다른 취약점과 결합하여 더 심각한 공격으로 이어질 수 있다.

4.7 AI 분석 결과

Kubee Web 솔루션의 AI 분석 기능을 통해 발견된 취약점들에 대한 지능형 분석을 수행한 결과, 기존의 단순한 규칙 기반 분석을 넘어서는 고도화된 분석 능력을 확인할 수 있었다. RAG 기반 분석을 통해 각 취약점의 구체적인 공격 시나리오와 대응 방안을 제시할 수 있었으며, 이는 보안 전문가의 지식을 자동화하는 데 크게 기여하였다. 공격 시나리오 생성은 AI 분석의 핵심 기능 중 하나로, 발견된 취약점을 기반으로 구체적인 공격 방법을 설명하였다. 예를 들어, volcano-agent의 Secret 접근 권한 취약점에 대해서는 공격자가 해당 Daemonset을 탈취한 후 클러스터의 모든 Secret을 조회하고, 다른 Account의 Service Account Token을 획득하여 권한 상승 공격을 수행하는 구체적인 시나리오를 제시하였다. 대응 방안 제안은 AI 분석의 또 다른 핵심 기능으로, 발견된 취약점에 대한 구체적인 수정 방법을 제안하였다. 예를 들어, kubeskoop의 privileged 컨테이너 취약점에 대해서는 privileged: true 설정을 제거하고 securityContext.capabilities를 사용하여 필요한 권한만 최소한으로 추가하는 방법을 제안하였다. 자연어 처리 기반 질의응답 기능을 통해 사용자들이 Kubernetes 보안에 대한 다양한 질문을 할 수 있었으며, 이는 솔루션의 사용성을 크게 향상시켰다. 지식베이스는 Kubernetes 보안 관련 포괄적인 정보를 포함하고 있으며, 이를 통해 다양한 보안 문제에 대한 정확한 분석과 대응 방안을 제시할 수 있었다.

5. 결론

5.1 결론 및 기대효과

Kubee Web 솔루션을 통해 실제 운영 환경에서 발견된 취약점들을 분석한 결과, 다양한 심각도의 보안 위협이 존재함을 확인하였다. Critical 등급의 취약점들은 공격자가 클러스터 전체를 장악할 수 있는 직접적인 경로를 제공하는 가장 위험한 보안 위협으로, volcano-agent의 과도한 Secret 접근 권한과 kubeskoop의 privileged 컨테이너 설정이 대표적인 사례이다. 이러한 취약점들은 최소 권한 원칙을 심각하게 위반하며, 즉시 수정이 필요한 상황이다.

주요 성과는 다음과 같다. 첫째, 기존의 단순한 규칙 기반 검사를 넘어서는 RAG 기반의 지능형 분석 시스템을 구축하여, 발견된 취약점에 대한 구체적인 공격 시나리오와 대응 방안을 자동으로 생성할 수 있었다. 이를 통해 보안 전문가의 지식을 자동화하고, 일반 사용자도 쉽게 사용할 수 있는 접근성을 제공할 수 있었다. 둘째, 웹 기반의 사용자 친화적인 인터페이스를 구축하여, 직관적인 대시보드와 상세한 분석 결과 시각화 기능을 통해 복잡한 보안 정보를 쉽게 이해할 수 있게 하였다. 셋째, 실제 운영 환경에서 발견된 취약점들을 분석하여, 각 취약점의 실제 위험성과 대응 방안을 구체적으로 제시할 수 있었다.

기대효과는 다음과 같다. 첫째, Kubernetes 클러스터의 보안 수준이 크게 향상될 것으로 기대된다. Critical과 High 등급의 심각한 취약점들을 수정함으로써 클러스터의 전반적인 보안 수준을 크게 향상시킬 수 있다. 둘째, 보안 전문가의 업무 효율성이 크게 향상될 것으로 기대된다. AI 기반의 자동화된 분석 기능을 통해 수동으로 수행하던 복잡한 보안 분석 작업을 자동화할 수 있으며, 이를 통해 보안 전문가들이 더 중요한 업무에 집중할 수 있게 된다. 셋째, 일반 사용자들의 보안 인식이 향상될 것으로 기대된다. 사용자 친화적인 인터페이스와 AI 기반의 질의응답 기능을 통해 보안 전문가가 아닌 일반 사용자들도 Kubernetes 보안에 대한 지식을 쉽게 습득할 수 있게 된다.

5.2 향후 계획

Kubee Web 솔루션의 지속적인 발전을 위해 다음과 같은 향후 계획을 수립하였다. 첫째, 추가 취약점 패턴 지원을 통해 더욱 포괄적인 보안 분석 능력을 제공할 계획이다. 현재 지원하는 취약점 유형을 확장하여 새로운 공격 벡터와 보안 위협을 탐지할 수 있도록 하고, 지속적인 보안 연구를 통해 최신 보안 위협에 대응할 수 있는 능력을 갖추고자 한다. 이를 위해 보안 연구진과의 협력을 강화하고, 최신 보안 연구 결과를 솔루션에 반영할 계획이다.

다중 클러스터 지원은 향후 계획의 핵심 요소 중 하나로, 여러 Kubernetes 클러스터를 통합적으로 관리할 수 있는 기능을 제공할 계획이다. 이를 통해 대규모 기업이나 조직에서 운영하는 여러 클러스터의 보안 상태를 한 곳에서 모니터링하고 관리할 수 있게 하며, 클러스터 간의 보안 정책 일관성을 유지할 수 있게 한다. 또한 클러스터 간의 보안 이벤트 연관 분석을 통해 더욱 정교한 보안 분석을 제공할 수 있게 한다.

CI/CD 통합은 개발 과정에서의 보안을 강화하기 위한 중요한 계획으로, GitLab, Jenkins, GitHub Actions 등의 CI/CD 파이프라인에 Kubee Web을 통합하여 자동화된 보안 검사를 제공할 계획이다. 이를 통해 개발자가 코드를 커밋하거나 배포할 때마다 자동으로 보안 검사를 수행하여, 취약점이 포함된 코드가 프로덕션 환경에 배포되는 것을 방지할 수 있다. 또한 보안 검사 결과를 CI/CD 파이프라인에 통합하여, 보안 정책을 위반하는 경우 빌드나 배포를 자동으로 중단할 수 있게 한다.

실시간 모니터링 기능은 클러스터의 보안 상태를 실시간으로 모니터링하고 이상 상황을 즉시 감지할 수 있는 기능을 제공할 계획이다. 이를 위해 Prometheus, Grafana 등의 모니터링 도구와 연동하여 실시간 메트릭 수집과 시각화를 제공하고, 이상 상황 발생 시 즉시 알림을 발송하는 기능을 구현할 계획이다. 또한 기계학습 기반의 이상 탐지 알고리즘을 도입하여, 정상적인 패턴과 다른 행위를 자동으로 감지할 수 있게 한다.

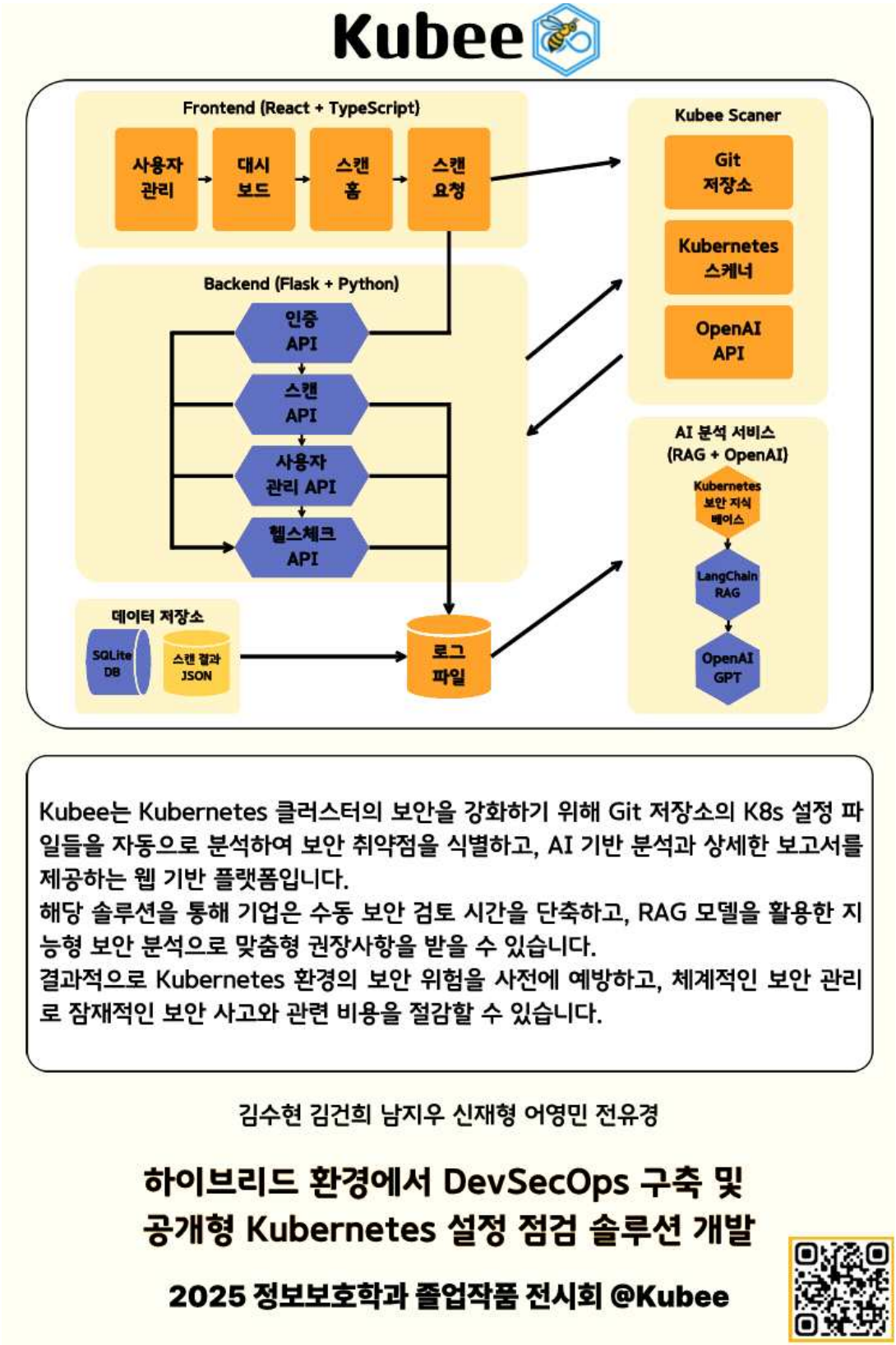
성능 최적화는 대규모 클러스터에서의 효율적인 동작을 위해 지속적으로 수행할 계획이다. 병렬 처리 기술을 더욱 발전시키고, 캐싱 메커니즘을 도입하여 중복 작업을 최소화할 계획이다. 또한 분산 처리 아키텍처를 도입하여 여러 노드에서 동시에 스캔 작업을 수행할 수 있게 한다.

사용자 경험 개선은 솔루션의 접근성과 사용성을 높이기 위한 지속적인 작업이다. 더욱 직관적이고 사용하기 쉬운 인터페이스를 제공하기 위해 사용자 피드백을 지속적으로 수집하고 반영할 계획이다. 또한 모바일 디바이스에서도 최적화된 사용자 경험을 제공하기 위해 반응형 디자인을 더욱 발전시키고, 모바일 전용 기능을 추가할 계획이다.

보안 강화는 솔루션 자체의 보안을 강화하기 위한 중요한 계획이다. 다중 인증(MFA) 기능을 도입하고, 역할 기반 접근 제어를 더욱 세밀하게 구현하며, 감사 로그 기능을 강화할 계획이다. 또한 정기적인 보안 감사를 통해 솔루션 자체의 취약점을 조기에 발견하고 수정할 계획이다.

6. 별첨

6.1 소개 자료 (폼 보드)



6.2 프로젝트 소개

https://github.com/kigma00/kubee-web

6.4 팀 소개

김 수 현 (팀장)
김 건 희
남 지 우
신 재 형
어 영 민
전 유 경

6.5 참고 자료

- Kubernetes Misconfiguration Scanning
  - kubectl-commands:
  https://kubernetes.io/docs/reference/kubectl/
  - security-policies:
  https://kubernetes.io/docs/concepts/security/pod-security-policy/
- Git Repository Integration
  Git CLI: https://git-scm.com/doc
  GitHub API: https://docs.github.com/en/rest

6.6 발표 자료 (발표 PPT)