*Journal of The Korea Institute of Information Security & Cryptology* ISSN 1598-3986(Print)
VOL.36, NO.3, Jun. 2026 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2026.36.3.1

# Kubernetes 환경의 보안 오설정 탐지를 위한 경량 스캐너 설계 및 구현*

김 유 원,[1†] 최 지 현,[2] 최 석 환[3‡]
[1]고려대학교 (대학원생), [2,3]연세대학교 (대학원생, 교수)

# Design and Implementation of a Lightweight Scanner for Detecting Security Misconfigurations in Kubernetes*

Yu-won Kim,[1†] Ji-hyun Choi,[2] Seok-hwan Choi[3‡]
[1]Korea University (Graduate student), [2,3]Yonsei University (Graduate student, Professor)

요 약

본 논문은 KISA 클라우드 취약점 점검 가이드를 기반으로 Kubernetes 환경의 보안 오설정 진단 스캐너를 제안한다. 제안하는 스캐너는 Kubernetes 구성 파일과 클러스터 설정 정보를 규칙 기반으로 분석하여 취약 설정을 식별하고, 점검 결과를 점수화 및 시각화된 리포트로 제공한다. 또한, 실효성을 검증하기 위해 Kind 기반의 멀티 노드 환경에서 제안한 스캐너를 시현하였다. 실험 결과 Recall은 100%를 기록했으며, 특히 Precision은 71.9%로 기존 도구 대비 3.3배 향상된 수치를 기록하였다. 이를 통해 국내 기준에 부합하는 Kubernetes 보안 점검 자동화 도구의 활용 가능성을 제시한다.

ABSTRACT

This paper proposes a Kubernetes security misconfiguration scanner based on the KISA Cloud Vulnerability Assessment Guide. The proposed system analyzes Kubernetes configuration files and cluster settings using rule-based checks to identify insecure configurations, providing results through quantified scores and visualized reports. Additionally, to verify its effectiveness, demonstrations in a Kind-based multi-node environment confirmed the scanner's inspection efficiency. The evaluation yielded a recall of 100% and a precision of 71.9%, representing a 3.3-fold improvement over existing tools. These results highlight the scanner's applicability as an automated tool aligned with domestic security guidelines.

**Keywords:** Kubernetes, Misconfiguration, Static Analysis, Security Scanner

## I. 서 론

클라우드 네이티브 환경의 확산으로 애플리케이션 배포 및 운영 방식이 빠르게 변화하고 있다. 특히 Kubernetes는 컨테이너 오케스트레이션의 표준으로 자리 잡으며 대규모 서비스의 자동화, 확장성 및 탄력성을 제공하는 핵심 플랫폼으로 활용되고 있다[1][2]. 그러나 Kubernetes는 구성 요소가 복잡하고 설정 단위가 세분화되어 있어 운영자의 작은 설정 오류(misconfiguration)만으로도 심각한 보안 취약점이 발생할 수 있다는 한계를 가진다[3][4][5]. 실제 클라우드 사고 사례에서도 시스템 자체의 취약점보다는 과도한 권한 부여[6], 인증서 및 키 파일의 부적절한 관리[7], 네트워크 정책 미구현과 같은 구성상의 문제로 인해 클러스터 전체가 노출되는 사

Received(01. 16. 2026), Modified(03. 19. 2026), Accepted(04. 10. 2026)
* 이 논문은 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원-대학ICT연구센터(ITRC)의 지원을 받아 수행된 연구임(IITP-2026-RS-2023-00259967)
† 주저자, ywkim@isslab.korea.ac.kr
‡ 교신저자, sh.choi@yonsei.ac.kr(Corresponding author)

례[4]가 보고되고 있다. 이러한 구성 오류는 Control Plane과 워커 노드 전반에 걸쳐 다양한 형태로 발생할 수 있으며, 특히 API Server 인증·인가 설정, RBAC(Role-Based Access Control) 권한 부여, etcd 암호화 및 Kubelet 보안 설정과 같은 핵심 구성 요소에서의 오설정은 클러스터 전체의 보안 위협으로 직결될 수 있다. Fig. 1은 Kubernetes 클러스터 구조와 주요 구성 요소를 기준으로, 구성 오류가 발생할 수 있는 대표적인 보안 취약 지점을 개념적으로 나타낸다.

Kubernetes 환경에서는 Pod 보안 설정, RBAC 정책, 컨테이너 런타임 권한, API 접근 제어 등 다양한 구성 단계에서 보안 관련 의사결정이 이루어진다. 하지만, 이러한 설정 항목들은 환경별 기준이 상이하고 구성 복잡도가 높아 수동 점검 방식만으로는 일관된 보안 수준을 유지하기 어렵다[8][9]. 국내에서는 KISA(한국인터넷진흥원)이 클라우드 보안 가이드를 통해 Kubernetes 및 Docker 환경에서 적용해야 할 보안 설정을 제시하고 있다[10]. 그러나 해당 가이드는 문서 중심의 점검 체계로 구성되어 있어 DevOps 환경에서 자동화된 보안 점검으로 활용하는 데에는 한계가 있다. 특히 클라우드 네이티브 개발에서는 배포 이전 단계에서 IaC(Infrastructure-as-Code) 기반의 YAML/JSON 구성 파일을 사용하는데, 이 단계에서 보안 문제를 사전에 검출할 수 있는 도구는 충분히 마련되어 있지 않다.

이에 본 논문에서는 KISA 클라우드 취약점 점검 가이드의 점검 항목을 분석하여 이를 기계가 해석 가능한 규칙 기반 구조로 재정의하고, Kubernetes 환경에서 구성 파일 및 클러스터 설정 정보를 대상으로 자동화된 분석을 수행하는 정적 분석 스캐너를 제안한다. 제안하는 스캐너는 API Server 인증/인가, Pod 보안 설정, RBAC 구성 등 핵심 보안 영역을 포괄하도록 설계되었다. 또한 점검 항목을 규칙 단위로 모듈화함으로써, 급변하는 클라우드 환경의 새로운 보안 기준이나 조직별 특화 정책을 유연하게 확장할 수 있는 구조를 갖도록 하였다.

본 연구에서 채택한 규칙 기반 분석은 사전에 정의된 보안 정책과 시스템의 실제 설정값 간의 일치 여부를 판별하는 방식이다. 이는 복잡한 행위 모니터링 기반의 동적 분석과 비교하여 오버헤드를 최소화하면서도, 런타임 클러스터 전반의 구성 데이터를 누락 없이 확실하게 검출할 수 있는 강점을 가진다. 특히 실행 중인 환경의 보안 상태를 신속하게 전수 조사할 수 있는 이론적 기반을 제공하므로, 실행 중인 서버에 영향을 주지 않고 효율적인 보안 점검을 수행하는 데 적합하다.

또한, 제안하는 스캐너는 보안 지식이 제한적인 학습자나 비전문가가 탐지 결과를 직관적으로 해석하고 대응 우선순위를 결정할 수 있도록 설계되었다. 이를 위해 각 점검 항목에는 보안 취약점의 객관적

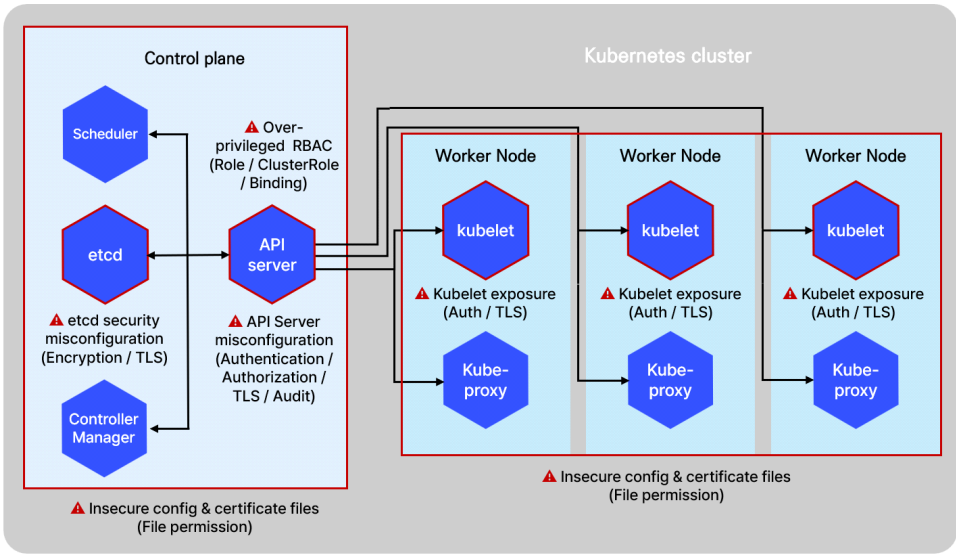


Fig. 1. Kubernetes Cluster Architecture with Key Security Misconfiguration Locations

평가지표인 CVSS(Common Vulnerability Scoring System) v3.1 표준의 심각도 기준을 준용하여 위험도와 중요도를 할당한 다차원 분석 모델을 적용하였다. 위험도는 해당 설정 오류가 공격자에 의해 악용될 가능성과 기술적 영향력을 의미하며, 중요도는 API Server나 etcd와 같이 클러스터 가용성에 직결되는 핵심 구성 요소의 자산 가치를 반영한다 [11]. 이러한 정량적 지표를 기반으로, 제안하는 스캐너는 단순한 통과 또는 실패 판정을 넘어 실질적인 보안 위협 수준에 따른 우선순위가 반영된 가시성 높은 리포트를 제공한다.

## II. 관련 연구

### 2.1 상용 클라우드 보안 관리 서비스

클라우드 네이티브 환경의 확산에 따라 주요 클라우드 서비스 제공자(cloud service provider)들은 자사 인프라에 최적화된 보안 구성 점검 및 모니터링 솔루션을 제공하고 있다. 대표적으로 AWS Security Hub[12], Microsoft Defender for Cloud[13], Google Cloud Security Command Center[14] 등이 있으며, 이들은 클라우드 리소스의 설정 오류 탐지 및 규제 준수 평가 기능을 통합적으로 제공한다. 그러나 이러한 상용 서비스들은 특정 CSP 환경을 기반으로 설계되어 있어서 로컬 환경이나 소규모 프라이빗 클러스터를 운영하는 관리자가 활용하기에는 비용적 부담과 운용 제약이 크다는 한계가 있다.

### 2.2 오픈소스 Kubernetes 보안 분석 도구

상용 솔루션의 대안으로 Kubernetes 환경의 취약점을 탐지하기 위한 다양한 오픈소스 도구들이 제안되었다.

Kube-bench는 CIS Benchmark[15]를 기준으로 클러스터 노드의 설정 적절성을 검사하며, 경량화된 실행 구조로 설치가 용이하다는 장점이 있다[16]. 하지만 클러스터 환경 변화에 따라 매번 새로운 Job 리소스를 배포하고 삭제해야 하는 운영상의 번거로움이 존재하며, 또한, 스캔 결과가 텍스트 형식으로 출력되어 대규모 클러스터 환경에서의 가독성 및 데이터 활용도가 낮다는 한계가 있다. Kubescape 또한 신속한 설치와 실행을 지원하며, Kube-bench와 달리 실행 시점에 클러스터 내 리소스를 동적으로 인식하여 취약점을 스캔하는 유연성을 제공한다[17]. Kubescape는 결과 보고서를 HTML 등의 시각적 형태로 추출할 수 있으나, 탐지된 항목의 세부 근거와 조치 방안 간의 연결 구조가 복잡하여 관리자가 직관적으로 보안 상태를 파악하기에는 가독성이 부족한 측면이 있다. Checkov는 배포 전 단계에서 IaC 기반의 YAML 및 JSON 파일을 정적으로 분석하는 기능을 제공하여 보안 사고를 사전에 방지하는 Shift-Left 보안 전략에 적합하다[18]. 그러나 이는 매니페스트 파일의 구조적 결함을 탐지하는 데 국한되므로 실제 런타임 환경에서 발생하는 API Server의 동적 설정 오류나 실시간 권한 변경 사항을 점검할 수 없다는 태생적 한계를 지닌다.

또한, 이러한 오픈소스 Kubernetes 보안 분석 도구들은 글로벌 보안 표준 CIS Benchmark, NSA-CISA[19], MITRE ATT&CK[20]을 기준으로 설계되어 범용적인 보안 가이드라인을 제시한다. 하지만 국내 공공 및 금융권 클라우드 도입의 필수 요건인 ISMS-P나 국가 공공기관 클라우드 컴퓨팅 보안 가이드라인의 세부 통제 항목을 완전히 포괄하지 못하는 규제 차이가 존재한다. 가장 대표적인 규제 차이는 국내 보안 요건의 핵심인 논리적 망 분리 규정의 구체성이다. CIS Benchmark는 단순히 네트워크 정책을 통한 일반적인 격리를 권고하는 수준인 반면, KISA 가이드라인은 관리용 네트워크의 원천적인 분리를 위해 kube-apiserver의 –bind-address를 루프백 주소(127.0.0.1)로 제한하거나 인증되지 않은 접근(--anonymous-auth=false)을 엄격히 차단할 것을 요구한다. 또한, 감사 로그 관리 측면에서도 CIS가 로그 활성화 여부에 집중하는 것과 달리, KISA 가이드라인은 국내 개인정보보호법 및 공공 가이드라인에 의거하여 로그 보존 기간(--audit-log-maxage)과 저장 용량 등을 구체적인 수치로 명시하여 사고 후 추적 가능성을 강제한다는 점에서 기술적 통제 강도가 훨씬 높다.

KISA에서 배포한 클라우드 보안 가이드는 국내 보안 사고 사례를 반영하여 기술 및 관리 요건을 상세히 제시하고 있으나 대부분 비정형화된 서술형 문장으로 구성되어 있다. 이로 인해 실제 DevOps 환경의 자동화된 보안 파이프라인에 즉각적으로 적용하기 어렵다. 특히, KISA 가이드라인의 정성적 점검 기준을 기계 학습이나 규칙 기반 알고리즘이 해석 가

능한 정형 데이터로 재구조화하여 자동화된 스캔에 적용한 연구는 매우 제한적인 실정이다. 이에 본 연구에서는 KISA 클라우드 취약점 점검 가이드를 기반으로 Kubernetes YAML/JSON 설정 파일을 정적으로 분석하는 경량 분석 스캐너를 제안하며, 국내 기준에 부합하는 구성 기반 보안 점검 자동화 가능성을 탐색하고자 한다.

## III. 제안하는 시스템

본 장에서는 제안하는 Kubernetes 취약점 스캐너의 전체 구조와 동작 방식을 설명한다. 제안한 Kubernetes 취약점 스캐너는 Kind나 Minikube와 같은 로컬 클러스터 환경을 사용하는 비전문가도 손쉽게 클러스터의 위협 상태를 진단할 수 있도록 사용자 친화적 경량 구조를 지향한다. 이는 방대한 보안 설정 항목 중 실제 공격 시나리오와 직결되는 핵심 지표만을 선별하여 제공함으로써 사용자의 분석 피로도를 최소화하기 위함이다. 따라서, 제안하는 스캐너는 점검 항목을 무분별하게 제시하지 않고 보안 가시성 확보에 필수적인 19개 항목으로 최적화하였다. 이러한 설계 방식은 사용자가 수많은 경고 메시지 중 실제 조치가 필요한 핵심 결함을 직관적으로 식별하고 제시된 권장 설정을 통해 즉각적인 보안 개선을 수행할 수 있도록 한다.

### 3.1 시스템 아키텍처 및 설계 개요

본 연구에서 제안하는 시스템은 Kubernetes 클러스터 내에 DaemonSet을 배포하여 클러스터에 참여 중인 모든 노드를 자동으로 식별하고 실시간 런타임 데이터를 수집한다. 전체 시스템은 사용자 인터페이스 계층, 스캐너 실행 엔진, 보안 설정 점검 모듈 계층, Kubernetes 연동 계층, 리포트 생성 계층으로 구성되며 각 계층은 역할에 따라 분리하였다. 이러한 계층적 구조를 통해 시스템의 확장성과 유지보수성을 확보하였다. 전체 아키텍처는 Fig. 2에 나타내었다.

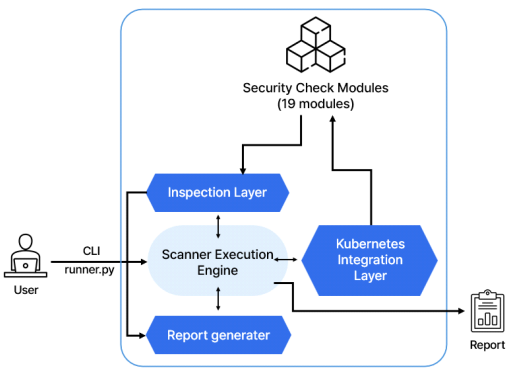


Fig. 2. System Architecture

### 3.2 사용자 인터페이스 계층

사용자 인터페이스 계층은 Kubernetes 보안 점검을 수행하기 위한 실행 진입점으로서 본 연구에서는 CLI(Command Line Interface) 기반의 실행 방식으로 구현하였다. CLI 방식은 개발자 및 보안 담당자가 터미널 환경에서 보안 점검을 신속하게 수행할 수 있으며 자동화 스크립트나 배치 작업에 쉽게 통합할 수 있다는 장점이 있다. 사용자가 CLI를 통해 스캐너를 실행하면 점검 요청은 내부 스캐너 실행 엔진으로 전달되며 이후의 분석 과정은 사용자 인터페이스와 독립적으로 수행된다.

### 3.3 보안 설정 점검 모듈 계층

보안 설정 점검 모듈 계층은 KISA 클라우드 취약점 점검 가이드를 기반으로 정의된 개별 보안 점검 항목을 구현한 모듈들의 집합이다. 각 모듈은 공통 인터페이스를 따르며 특정 Kubernetes 구성 요소를 대상으로 보안 설정을 검사한다. 본 논문에서는 API Server, Controller Manager, etcd, PSA, File, Kubelet과 같은 주요 점검 범주를 포함하였다. 또한, 각 점검 모듈에 대해 다음과 같은 분석 방식을 교차 적용하였다.

1. Kubernetes API 기반 분석: 실행 중인 리소스의 설정 상태를 조회하여 보안 설정 검증
2. 명령어 기반 설정 분석: 시스템 컴포넌트의 실행 인자를 분석하여 인증 및 인가 설정 확인
3. 파일 기반 분석: 설정 파일 및 인증서 파일의 접근 권한과 보안 설정 점검

이러한 다중 분석 방식을 바탕으로 Kubernetes 환경 전반에 대한 설정 기반 보안 점검을 수행한다.

### 3.4 Kubernetes 연동 계층

Kubernetes 연동 계층은 스캐너가 실행되는 환경을 자동으로 감지하여 적절한 인증 방식을 선택한다. 이를 통해 로컬 개발 환경과 클러스터 내부 실행 환경 모두에서 동일한 방식으로 보안 점검이 가능하다. Kubernetes 취약점 스캐너는 최소 권한 원칙을 준수하여 클러스터 리소스에 대해 읽기 전용 접근만을 수행하며 클러스터의 상태를 변경하지 않는다.

### 3.5 리포트 생성 계층

리포트 생성 계층은 점검 결과를 구조화된 형태로 가공하여 사용자에게 제공한다. 점검 결과는 구조화된 JSON 형식과 사용자가 직관적으로 확인할 수 있는 HTML 리포트 형식으로 제공되며 각 점검 항목에 대해 결과 상태, 위험도, 원인 설명, 권장 설정 정보를 함께 제공한다. 특히 HTML 형식의 리포트는 시각적 요소를 활용하여 전체 보안 상태를 직관적

Table 1. Details of Security Check

| Category | Security Check | Risk | Verification Command |
|---|---|---|---|
| API Server | Unauthenticated Access Disabled | 10 | --anonymous-auth, -service-account-lookup |
| | Authorization Mode | 10 | --authorization-mode=RBAC |
| | Service External Access Disabled | 9 | --bind-address=127.0.0.1 |
| | SSL/TLS Enabled | 9 | --secure-port, --tls-cert-file, --tls-private-key-file, --client-ca-file |
| | Weak Auth Disabled | 9 | --token-auth-file |
| | Log Management | 7 | --audit-log-path, --audit-policy-file |
| Controller Manager | Authentication Control | 8 | --use-service-account-credentials, --service-account-private-key-file |
| | SSL/TLS Enabled | 8 | --root-ca-file, --feature-gates |
| etcd | Encryption Enabled | 10 | --encryption-provider-config |
| | SSL/TLS Enabled | 9 | --client-cert-auth=true, --cert-file, --key-file, --trusted-ca-file |
| PSA | Container Privilege Control | 9 | allowPrivilegeEscalation, seccompProfile |
| | Namespace Sharing Disabled | 9 | hostNetwork, hostPID, hostIPC |
| File | (Master Node) Configuration File Permissions | 7 | File Permissions ≤ 644, Owner: root |
| | (Worker Node) Configuration File Permissions | 7 | File Permissions ≤ 644, Owner: root |
| | Worker Node Certificate File Permissions | 7 | Certificate: 644, Key: 600, Owner: root |
| Kubelet | Authentication Control | 8 | --client-ca-file |
| | Authorization Control | 8 | --authorization-mode=Webhook |
| | SSL/TLS Enabled | 8 | --tls-cert-file, --tls-private-key-file |
| | Kernel Parameter Configuration | 6 | net.ipv4.ip_forward=1, net.bridge.bridge-nf-call-iptables=1 |

API: Application Programming Interface, PSA: PodSecurityAdmission, PID: Process ID, TLS: Transport Layer Security, SSL: Secure Sockets Layer, IPC: Inter-Process Communication

으로 표현함으로써 비전문 사용자도 결과를 쉽게 이해할 수 있도록 구성하였다.

### 3.6 스캐너 실행 엔진

스캐너 실행 엔진은 전체 점검 과정을 제어하는 핵심 구성 요소로 보안 설정 점검 모듈의 로드, 실행 관리, 결과 집계 및 점수 산정을 수행한다.

#### 3.6.1 보안 설정 점검 모듈의 동적 로드

본 연구에서 구현한 점검 항목은 API Server, Controller Manager, etcd, PSA, File, Kubelet의 6가지 범주로 분류되며, 총 19개의 독립적인 모듈 단위로 구현되었다. 각 범주별 세부 점검 항목의 명칭과 구체적인 점검 기준은 Table 1에 정리하였다. 세부 범주별 특징은 다음과 같다.

- API Server (6개): 인증/인가, TLS 구성, 감사 로그
- Controller Manager (2개) / etcd (2개): 인증 및 통신 암호화 여부
- PSA (2개): 특권 모드 사용 및 네임스페이스 공유 금지
- File (3개): 설정 파일 및 인증서 권한 관리
- Kubelet (4개): 워커 노드 인증/인가, TLS 및 커널 파라미터 설정

본 논문에서는 전체 19개 항목 중 Critical(10점) 3개, High(9~8점) 11개, Medium(7~6점) 5개로 구성되어 핵심 보안 설정에 높은 가중치를 부여하였다.

이러한 보안 점검 항목은 총 19개의 독립적인 모듈 단위로 구현하였으며 실행 시점에 모듈이 자동으로 탐지되어 로드된다. 스캐너 실행 엔진은 보안 설정 점검 모듈을 동적으로 로드하고 실행함으로써 새로운 점검 항목이 추가되더라도 기존 코드의 수정 없이 확장 가능토록 설계되었다. 이러한 동적 로드 구조는 보안 기준 변경이나 점검 항목 확장에 유연하게 대응할 수 있도록 하며 시스템의 유지보수 부담을 최소화한다.

#### 3.6.2 병렬 실행 기반 점검 수행

다수의 보안 점검 항목을 효율적으로 수행하기 위해 스캐너 실행 엔진에는 병렬 실행 구조를 적용하였다. 실행 엔진은 Python의 ThreadPoolExecutor 라이브러리를 활용하여 최대 6개의 워커 스레드를 생성하고 각 보안 설정 점검 모듈을 서로 독립적으로 병렬 실행한다. 각 모듈의 실행 결과는 비동기적으로 수집되어 이후 결과 집계 단계로 전달된다. 이와 같은 병렬 실행 구조를 통해 점검 항목 수 증가에 따른 전체 스캔 시간의 선형적 증가를 완화할 수 있다. 본 연구에서 구현한 점검 항목 수(총 19개)를 기준으로 할 때, 이론적으로는 순차 실행 대비 약 3배 수준의 실행 시간 단축이 가능하다(19개 항목 ÷ 6개 워커 ≈ 3.2). 실제 Kubernetes 환경에서 스캐너를 실행한 결과, 전체 보안 점검은 평균 약 5초 내외로 완료되었으며 이는 병렬 실행 구조가 Minikube 및 Kind 등의 경량 Kubernetes 환경에서 효율적으로 동작함을 보여준다.

#### 3.6.3 점검 결과 집계 및 점수화

각 점검 모듈은 공통된 데이터 형식을 통해 분석 결과를 반환하며 실행 엔진은 클러스터 전반에서 수집된 개별 결과를 집계하여 최종 리포트를 생성한다. 특히 멀티 노드 환경에서의 진단 신뢰성을 확보하기 위해 최악의 상황 가정 원칙에 따른 집계 로직을 적용한다. 동일한 보안 점검 항목이 복수의 노드나 Pod에서 상이한 결과를 보일 경우, 시스템은 가장 결함이 심각한 결과를 해당 항목의 최종 상태로 반영한다. 예를 들어, 동일 점검 항목에 대해 3개의 노드 중 2개 노드가 PASS일지라도 1개 노드에서 FAIL이 발생하면 클러스터 전체의 보안 일관성을 위해 해당 항목은 최종적으로 FAIL 처리된다.

각 점검 항목에는 보안 중요도를 반영한 위험도 점수가 부여되며 이는 보안 숙련도가 낮은 사용자가 치명적인 위협을 우선적으로 식별하고 즉각 조치할 수 있는 가이드라인을 제공하기 위함이다. 또한, 정량적인 최종 보안 점수를 산출하는 과정에서 핵심 보안 설정에 따른 가중치를 체계적으로 반영하기 위해 CVSS v3.1의 심각도 기준인 파급력과 공격 용이성을 준수하여 다음과 같이 설계되었다.

- 10점 (Critical): 식별 즉시 클러스터 전체 권한 탈취가 가능하거나 호스트 OS로의 침투가 가능한 치명적 취약점
- 9점 및 8점 (High): 공격자의 권한 상승 및 공격 흔적 은폐를 직접적으로 돕는 핵심 설정

오류

- 7점 및 6점 (Medium): 직접적인 침해 원인은 아니나 공격 표면을 넓힐 수 있는 설정

최종 보안 점수는 각 항목의 결과 상태에 따라 차등 부여된다. 권장 보안 설정을 모두 만족할 경우(PASS) 해당 항목 배점의 100%를 획득하며, 보안 설정이 부적절한 경우(FAIL)나 권한 문제 등으로 점검이 불가한 경우(ERROR)에는 0점을 부여한다.

### 3.7 개발 및 실행 환경

제안한 Kubernetes 보안 오설정 진단 스캐너는 한국어 개발 환경의 편의성을 고려하여 Windows 11 운영체제에서 주요 개발을 진행하였다. 개발 과정에서는 GitHub를 활용한 버전 관리 시스템을 적용하여 코드 변경 이력을 관리하였으며 동일한 소스 코드를 Linux 환경에서도 실행할 수 있도록 개발과 배포를 병행하였다. 이를 위해 Windows와 Linux의 두 환경에서 동일한 배포판을 사용하여 실행 환경의 일관성을 유지하였다. 또한 스캐너는 Python 언어로 구현되었으며 Linux 기반 Kubernetes 환경에서 실제 실행 및 시현을 수행하였다.

## IV. 실험 및 시현 결과

본 장에서는 구현한 Kubernetes 보안 오설정 진단 스캐너가 실제 클러스터 환경에서 정상적으로 동작함을 확인하고 기존 보안 도구들과의 비교를 통해 탐지 성능을 검증한다. 본 연구의 시스템 시현은 제안한 스캐너가 멀티 노드 환경에서 다수의 노드를 동시에 인식하고 보안 설정을 전수 조사할 수 있는지 검증하는 데 목적이 있다. 이를 위해 Kind를 활용하여 3개의 노드로 구성된 클러스터 환경에서 실험을 진행하였다. 단일 노드 구성인 Minikube 환경에서도 정상 동작을 확인하였으나 실제 운영 환경과 유사한 멀티 노드 환경에서의 노드 자동 식별 및 집계 능력을 입증하기 위해 본 장에서는 Kind 환경에서의 실험 결과를 중심으로 기술한다.

### 4.1 실험 환경 및 위협 시나리오

본 절에서는 제안한 시스템의 탐지 성능을 검증하기 위한 하드웨어 및 소프트웨어 구성과 실험에 사용된 보안 위협 시나리오를 기술한다.

#### 4.1.1 실험 환경 구성

본 실험은 Docker 컨테이너 기반의 멀티 노드 구성을 지원하는 Kind를 활용하여 1개의 Control Plane과 2개의 Worker Node로 이루어진 클러스터 환경을 주 대상으로 수행하였다. 이는 단일 노드 환경에서 식별하기 어려운 노드 간 설정 편차와 런타임 보안 위협을 다수의 노드 전반에서 전수 조사할 수 있는지 검증하기 위함이다. 실험 과정에서는 실제 침해 사고를 모사한 Kubernetes-Goat[21] 취약점 테스트 환경을 배포하여 시스템이 전 계층에서 발생하는 보안 결함을 정확하게 식별하고 집계하는지 확인하였다.

#### 4.1.2 위협 시나리오

Kubernetes-Goat은 의도적으로 보안 취약점이 설계된 오픈소스 기반 보안 학습 플랫폼으로, 스캐너의 탐지 성능 검증을 위한 테스트베드로 활용된다. 본 연구에서는 핵심 계층별 위협을 대변하는 4가지 시나리오를 선별하였다.

1. Insecure RBAC: 과도한 권한 부여(cluster-admin) 및 익명 접근 허용 정책의 결함을 다룬다. 공격자가 탈취한 토큰으로 클러스터 제어권을 장악하거나 Privileged Pod를 생성하는 과정을 통해 인가 정책의 무결성을 점검한다.
2. System-Monitor: 모니터링 서비스를 구실로 한 과도한 특권 부여(privileged: true) 및 호스트 파일 시스템 마운트 설정을 포함한다. 네트워크 정책 부재와 자원 격리 미흡이 결합된 복합적인 취약점을 식별한다.
3. Kube-Bench-Security: API Server, Kubelet 등 인프라 핵심 컴포넌트의 가이드라인 위반 사례를 다룬다. 감사 로그 비활성화 및 통신 암호화 미적용 등 인프라 계층의 고질적인 설정 오류에 대한 검증 능력을 확인한다.
4. Docker-Bench-Security: 도커 소켓 노출 및 호스트 네임스페이스 공유를 통한 컨테이너 격리 무력화 위협에 집중한다. 이를 통해 공격자가 호스트 시스템의 권한을 탈취하는 컨테이너 탈출 가능성을 진단하고 런타임 가시성을 확보한다.

### 4.2 스캐너 실행 시현

Fig. 3은 제안한 스캐너를 CLI 환경에서 실행했을 때의 출력 화면과 그에 따른 진단 결과를 보여준다. 그림의 상단부에서는 스캐너 실행 직후 사용자의 클러스터 컨텍스트를 자동으로 분석하여 Kubernetes 환경을 감지하고 연결에 성공했음을 알리는 로그를 확인할 수 있다. 이어지는 하단부에서는 실제 점검 결과가 JSON 형식으로 실시간 출력된다. 각 결과 항목은 CheckID, Result, Reason 등의 필드로 구성되며, 특히 Evidence 필드를 통해 탐지 근거가 된 실제 설정값을 명시함으로써 사용자가 취약점의 원인을 즉각적으로 파악할 수 있도록 돕는다.

```
$ python3 runner.py
Starting Kubernetes Security Scanner...
==================================================
[OK] Successfully connected to Kubernetes cluster
[INFO] Current Kubernetes context: kind-kind
==================================================
{
  "ScanID": "scan-cef50464",
  "Timestamp": "2026-03-18 17:18:44.923300",
  "Results": [
    {
      "CheckID": "CHK-M-API-002",
      "Result": "PASS",
      "ObjectType": "Pod",
      "ObjectName": "kube-apiserver-kind-control-plane",
      "Namespace": "kube-system",
      "Reason": "--token-auth-file 플래그 미발견(정적 토큰 사용 안함)",
      "Evidence": {
        "args": [
```

Fig. 3. CLI execution screen

### 4.3 보안 점검 결과 시현

Fig. 4는 제안한 스캐너의 HTML 형식 리포트 예시이다. 리포트에는 전체 점검 결과 요약, 심각도별 분포, 개별 항목의 상세 정보가 포함된다. 사용자는 전문적인 보안 지식이 없더라도 PASS, FAIL, ERROR로 구분된 상태값과 위험도 점수를 통해 클러스터의 보안 상태를 직관적으로 파악할 수 있다. FAIL로 탐지된 항목에 대해서는 Fig. 5와 같이 직관적인 상세 분석 리포트를 제공한다. 여기에는 위험도 점수와 실패 원인뿐만 아니라, 사용자가 직접 검증할 수 있는 수동 확인 명령어와 구체적인 해결 방법이 포함되어 즉각적인 보안 개선을 돕는다.

이때 Fig. 4에서는 고유 점검 항목(unique id)을 기준으로 FAIL 수가 8건으로 집계되는 반면


Fig. 4. HTML-based measurement results of Scenario 1(Insecure-RBAC)

Fig. 5의 상세 리스트에서는 총 15건의 항목이 도출된다. 이는 본 시스템이 멀티 노드 환경에서 각 노드별로 존재하는 개별 리소스를 전수 조사하기 때문이다. 즉, 동일한 점검 항목이라 하더라도 마스터 노드와 복수의 워커 노드에서 각각 위반 사항이 식별될 경우, 상세 리포트에는 노드별 물리적 개체 정보가 개별 항목으로 나열되어 탐지 결과의 투명성과 추적성을 보장한다.

ERROR 항목 역시 경고 발생 원인을 상세히 제공하여, 시스템이 자동 판단하기 어려운 권한 문제나 환경적 특이사항에 대해 사용자가 운영 정책에 따라 수동 검토를 수행할 수 있도록 지원한다.

### 4.4 성능 평가 및 분석

실험에 사용된 4가지 탐지 도구별 시나리오 대응 특성과 정량적 성능 지표를 비교 분석한다. 모든 지표는 각 도구가 보고한 점검 항목 중 부적합(FAIL) 상태를 분석 대상으로 산출하였다. 성능 평가를 위한 핵심 지표로서, 각 시나리오에 설계된 실제 취약점을 정확히 탐지한 경우를 True Positive로 정의하였으며, 취약점과 직접적인 연관이 없는 운영 권고 사항이나 불필요한 탐지 항목은 False Positive로 분류하였다. 이를 기반으로 도출된 Table 2를 바탕으로 실제 공격 시나리오상에서 각 도구가 보이는 기술적

Table 2. Detection Results of Security Scanners by Threat Scenario

| Tools | | Insecure RBAC | System Monitor | Kube Bench | Docker Bench |
|---|---|---|---|---|---|
| Kube-bench | Precision | 0% | 0% | 100% | 0% |
| | Recall | 0% | 0% | 17% | 0% |
| Kubescape | Precision | 16.7% | 12.5% | 25% | 20.8% |
| | Recall | 80% | 60% | 85% | 83% |
| Checkov | Precision | 26.3% | 21.1% | 27.8% | 25% |
| | Recall | 60% | 70% | 83% | 83% |
| Proposed scanner | Precision | 75% | 62.5% | 75% | 75% |
| | Recall | 100% | 100% | 100% | 100% |

RBAC: Role-Based Access Control

한계와 제안 시스템의 우수성을 규명한다.

Kube-bench는 CIS Benchmark 가이드라인에 따른 정적 설정 준수 여부 확인에 최적화되어 있다. 실험 결과, 인프라 설정 오류를 다루는 Kube-Bench-Security 시나리오에서는 100%의 정밀도를 기록하였으나, 실제 침해 사고와 직결된 타 시나리오에서는 평균 재현율 4.3%라는 낮은 수치를 보였다. 이는 감사 로그 설정과 같은 정적 매니페스트 속성은 식별하나, 특권 컨테이너 실행이나 익명 접근 허용과 같은 동적 런타임 위협 및 정책적 결함을 포착하지 못하는 벤치마크 기반 도구의 태생적 한계를 보여준다.

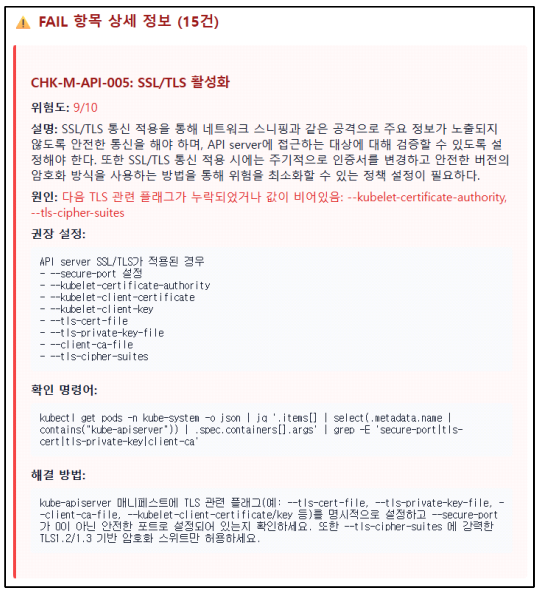


Fig. 5. Detailed information obtained from Scenario 1(Insecure-RBAC)

Kubescape와 Checkov는 광범위한 점검 항목을 바탕으로 각각 68.3%, 69.1%의 높은 재현율을 기록하였다. 그러나 두 도구 모두 시나리오당 평균 20개 이상의 FAIL 항목을 도출하며 20% 내외의 낮은 정밀도를 보였다. 이는 privileged Pod 생성이나 hostPath 마운트 등 핵심 위협뿐만 아니라, 리소스 제한 미설정이나 라벨 누락과 같이 공격 시나리오와 무관한 운영 권고 사항까지 과도하게 탐지하기 때문이다. 이러한 결과 나열은 관리자로 하여금 cluster-admin 권한 오설정과 같은 치명적 위협을 식별하는 데 방해 요인으로 작용하며 분석 피로도를 가중시킨다.

제안 시스템은 모든 시나리오에서 핵심 취약점을 100% 식별하여 재현율 100.0%를 달성하였으며 기존 도구 대비 약 3.3배 향상된 71.9%의 높은 정밀도를 기록하였다. 제안 시스템은 무분별한 결과 나열을 지양하고 시나리오당 평균 8.0개의 핵심 항목만을 정밀하게 보고한다. 특히 익명 접근, 네임스페이스 공유, 데이터 암호화와 같이 공격 킬 체인 상에서 결정적인 지점을 우선순위화하여 탐지한다. 이는 사용자가 검토해야 할 정보량을 65% 이상 절감함과 동시에, 클러스터 제어권 탈취로 이어질 수 있는 실질적 위협에 대한 대응력을 극대화한다.

## V. 결 론

본 논문에서는 KISA 클라우드 취약점 점검 가이드를 기반으로 Kubernetes 환경의 보안 오설정을 사전에 진단할 수 있는 정적 분석 기반 스캐너를 설계 및 구현하였다. 제안한 시스템은 KISA 가이드의

서술적 점검 기준을 규칙 단위로 재구조화하여 19개의 핵심 점검 모듈로 최적화하였다. 이를 통해 보안 가시성을 확보함과 동시에 시스템의 확장성과 유지보수성을 달성하였다. 특히 본 연구에서는 Kubernetes-Goat 기반의 4가지 공격 시나리오를 통해 제안 시스템의 탐지 성능을 정량적으로 검증하였다. 실험 결과, 제안한 스캐너는 모든 시나리오에서 핵심 취약점을 모두 식별하여 100%의 Recall을 달성하였다. 또한, 불필요한 보안 노이즈를 최소화한 설계 전략을 통해 평균 71.9%의 높은 Precision을 기록하였으며 이는 기존 오픈소스 도구 대비 약 3.3배 향상된 수치이다. 이를 통해 비전문가도 방대한 설정 오류 중 실질적인 위험을 직관적으로 파악하고 즉각 대응할 수 있는 환경을 제공함을 확인하였다.

본 연구는 정적 분석 기반의 특성상 실시간 런타임 행위 분석을 수행하지 않는다는 한계가 있으나 국내 보안 가이드라인을 기반으로 한 실효성 있는 Kubernetes 보안 점검 도구의 구현 가능성을 성공적으로 제시하였다. 향후 연구에서는 런타임 분석 기법의 결합 및 CI/CD 파이프라인과의 연계를 통해 더욱 자동화된 보안 점검 체계로 확장할 계획이다. 본 연구에서 구현한 스캐너의 공개[22]가 향후 Kubernetes 보안 연구 및 교육 환경에서 중요한 기초 자료로 활용되기를 기대한다.

## References


[1] E. Casalicchio and S. Iannucci, "The state-of-the-art in container technologies: Application, orchestration and security," Concurrency and Computation: Practice and Experience, vol. 32, no. 17, pp. e5668, Sep. 2020.

[2] A. Marchese and O. Tomarchio, "SLO-Aware Container Orchestration on Kubernetes Clusters," Proceedings of the 2025 IEEE 18th International Conference on Cloud Computing (CLOUD), pp. 318 - 327, Jul. 2025.

[3] A. Rahman, S. I. Shamim, D. B. Bose, and R. Pandita, "Security Misconfigurations in Open Source Kubernetes Manifests: An Empirical Study," ACM Transactions on Software Engineering and Methodology, vol. 32, no. 99, pp. 1 - 36, Jul. 2023.

[4] J. Bufalino, J. L. Martin-Navarro, M. Di Francesco, and T. Aura, "Inside Job: Defending Kubernetes Clusters Against Network Misconfigurations," Proceedings of the ACM on Networking (PACMNET), vol. 3, no.20, pp. 1-25, Dec. 2025.

[5] C. Cesarano and R. Natella, "KubeFence: Security Hardening of the Kubernetes Attack Surface," Proceedings of the 2025 55th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 497-510, Jun. 2025.

[6] N. Yang, W. Shen, J. Li, X. Liu, X. Guo, and J. Ma, "Take Over the Whole Cluster: Attacking Kubernetes via Excessive Permissions of Third-Party Applications," Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), pp. 3048-3062, Nov. 2023.

[7] M. Rostamipoor, A. Sadeghi, and M. Polychronakis, "KubeKeeper: Protecting Kubernetes Secrets Against Excessive Permissions," Proceedings of the 2025 IEEE 10th European Symposium on Security and Privacy (EuroS&P), pp. 322-338, Jul. 2025.

[8] B. Kim and S. Lee, "KUBEAEGIS: A Unified Security Policy Management Framework for Containerized Environments," IEEE Access, vol. 12, pp. 160636 - 160652, Oct. 2024.

[9] Z. Ye, T. H. M. Le, and M. A. Babar, "LLMSecConfig: An LLM-Based Approach for Fixing Software Container Misconfigurations," Proceedings of the 2025 IEEE/ACM 22nd International Conference on Mining Software Re-

positories (MSR), pp. 329-341, Mya. 2025.
[10] Korea Internet & Security Agency, "Cloud Vulnerability Assessment Guide," https://www.cela.kr/4/?bmode=view&idx=40424414, 2024.
[11] FIRST, "Common Vulnerability Scoring System v3.1 Specification Guide," https://www.first.org/cvss/v3.1/specification-document, 2019.
[12] Amazon Web Services, "AWS Security Hub," https://aws.amazon.com/ko/security-hub/, Accessed: Jan. 2, 2026.
[13] Microsoft, "Microsoft Defender for Cloud," https://www.microsoft.com/en-us/security/business/cloud-security/microsoft-defender-cloud, Accessed: Jan. 2, 2026.
[14] Google Cloud, "Security Command Center," https://docs.cloud.google.com/security-command-center/docs/security-command-center-overview, Accessed: Jan. 2, 2026.
[15] Center for Internet Security (CIS), "CIS Kubernetes Benchmark v1.8.0," https://www.cisecurity.org/benchmark/kubernetes, Accessed: Mar. 12, 2026.
[16] GitHub, "kube-bench: Checks whether Kubernetes is deployed according to security best practices," https://github.com/aquasecurity/kube-bench, Accessed: Mar. 12, 2026.
[17] ARMO, "Kubescape: An open-source Kubernetes security platform," https://kubescape.io/, Accessed: Mar. 12, 2026.
[18] GitHub, "Checkov: Prevent cloud misconfigurations during build-time for Infrastructure as Code," https://github.com/bridgecrewio/checkov, Accessed: Mar. 16, 2026.
[19] National Security Agency (NSA) and Cybersecurity and Infrastructure Security Agency (CISA), "Kubernetes Hardening Guidance: Cybersecurity Technical Report," https://www.cisa.gov/resources-tools/resources/kubernetes-hardening-guidance, Accessed: Mar. 12, 2026.
[20] MITRE, "ATT&CK for Containers," https://attack.mitre.org/matrices/enterprise/cloud/container/, Accessed: Mar. 12, 2026.
[21] Madhu Akula, "Kubernetes Goat: A Vulnerable Kubernetes Cluster for Security Testing," https://github.com/madhuakula/kubernetes-goat, Accessed: Jan. 12, 2026.
[22] GitHub, "Automated Kubernetes Security Configuration Scanner Based on the KISA Cloud Security Guide," https://github.com/kimyouwon/Cloud_Scanner, Accessed: Jan. 12, 2026.

## 〈저자소개〉

김 유 원 (Yu-won Kim) 학생회원
2026년 2월 : 연세대학교 소프트웨어전공 졸업
2026년 2월~현재 : 고려대학교 컴퓨터학과 석사과정
〈관심분야〉 클라우드 보안, 쿠버네티스 등

최 지 현 (Ji-hyun Choi) 학생회원
2026년 2월 : 연세대학교 소프트웨어전공 졸업
2026년 2월~현재 : 연세대학교 전산학과 석사과정
〈관심분야〉 AI for Security, NIDS, Anomaly Detection

최 석 환 (Seok-hwan Choi) 정회원
2016년 8월 : 부산대학교 정보컴퓨터공학부 공학사
2022년 8월 : 부산대학교 정보융합공학사 공학박사
2022년~현재 : 연세대학교 소프트웨어학부 교수
〈관심분야〉 AI 보안, 정보보안, 네트워크 보안 등