Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.34, NO.6, Dec. 2024 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2024.34.6.1335

Kunerva+: 클라우드 네이티브 환경을 위한 지능형 네트워크 정책 생성 프레임워크*

김 봄,[1†] 이 승 수[2‡]
[1,2]인천대학교 (대학원생, 교수)

Kunerva+: An Intelligent Network Policy Generation Framework for Cloud Native Environments*

Bom Kim,[1†] Seungsoo Lee[2‡]
[1,2]Incheon National University (Graduate student, Professor)

요 약

컨테이너는 확장성, 이식성, 자원 효율성을 활용하여 클라우드 네이티브 서비스를 제공하는 표준이 되었다. 동시에 잘못된 구성과 취약점, 특히 네트워크 정책의 취약점을 악용하는 다양한 보안 공격의 대상이 되고 있다. 복잡한 클라우드 네이티브 환경에서 수동으로 정책을 관리하는 것은 오류를 발생시키기 쉽고, 정책 생성을 자동화하는 기존 연구들은 정확성에 한계가 있다. 본 논문은 고도로 자동화된 지능형 네트워크 생성 프레임워크인 Kunerva+를 제시한다. Kunerva+는 자연어 처리와 파인튜닝된 대규모 언어 모델을 통해 강화된 의도 기반 방식으로 작동하여, 복잡한 구성을 이해할 필요 없이 네트워크 정책을 생성하고 정책 시행에서 잘못된 구성을 근본적으로 방지하기 위해 다단계 검증 프로세스를 고안했다. 평가 결과, 가장 개선된 파인튜닝된 LLM이 기준 모델보다 BLEU 점수는 360%, ROUGE-2 점수는 233% 향상되며, 의도 기반 생성의 가능성과 효과성을 입증했다.

ABSTRACT

Containers have become the standard for delivering cloud-native services, leveraging their scalability, portability, and resource efficiency. Simultaneously, they have become targets for various security attacks exploiting misconfigurations and vulnerabilities, particularly in network policies. In complex cloud-native environments, manual policy management is prone to errors, and existing research on policy generation automation has limitations in accuracy. This paper presents Kunerva+, a highly automated intelligent network policy generation framework. It operates through an enhanced intent-based approach using natural language processing and fine-tuned large language models, generating network policies without the need to understand complex configurations. We have also devised a multi-stage validation process to fundamentally prevent misconfigurations in network policy enforcement. The evaluation results show that the most improved fine-tuned LLM achieved a 360% increase in BLEU score and 233% in ROUGE-2 score compared to the baseline model, demonstrating the potential and effectiveness of intent-based generation.

Keywords: Cloud-native Architecture, Container Network, Intent-based Network Policy, LLM-based Policy Generation

Received(09. 20. 2024), Modified(11. 18. 2024), Accepted(11. 19. 2024)
* 이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2022R1C1C1006093)

* 본 논문은 2024년도 한국정보보호학회 하계학술대회에 발표한 우수논문을 개선 및 확장한 것임

† 주저자, zxx0313@inu.ac.kr
‡ 교신저자, seungsoo@inu.ac.kr(Corresponding author)

I. 서 론

클라우드 컴퓨팅 기술의 급속한 발전과 함께 클라우드 네이티브 기술 사용이 폭발적으로 증가하고 있다. 이러한 빠른 기술 발전은 잘못된 설정(Misconfiguration)이나 보안 취약점을 겨냥한 사이버 공격과 같은 새로운 보안 과제를 동반한다. Red Hat의 2024년 클라우드 보안 현황 보고서에 따르면, 응답자의 42%가 컨테이너와 Kubernetes[1]에서 보안을 가장 큰 우려사항으로 지목했으며, 잘못된 구성이 주요 보안 위험 요인으로 나타났다[17]. 동적으로 변화하는 클라우드 네이티브 환경, 특히 마이크로서비스 아키텍처에서는 네트워크 구성이 자주 변경되고 서비스 간 상호작용이 복잡하다. 이에 따라 수많은 네트워크 정책을 수동으로 구성하고 관리하는 것은 비현실적이며 정책 일관성 유지와 실시간 대응에 어려움이 있다. 또한 오타나 누락된 설정 같은 사소한 실수로도 전체 시스템의 보안이 위협받을 수 있다.

이러한 위협을 해결하기 위해 클라우드 네이티브 환경에서의 네트워크 정책 관리를 자동화하는 여러 연구가 제안되었다. 그러나 기존의 로그 기반 분석과 정적 분석 접근 방식들은 과거 데이터에 의존하거나 동적 변화를 반영하기 어렵다. 최근 제안된 의도 기반 접근 제어(IBAC) 방식도 빈번한 변화로 특징지어지는 클라우드 네이티브 환경에 적합하지 않으며 상당한 사용자 개입이 필요하다.

본 연구에서는 클라우드 네이티브 환경에 특화된 네트워크 정책의 자동 생성을 위한 지능형 프레임워크인 Kunerva+을 제안한다. Kunerva+는 자연어 처리(NLP) 기술과 대규모 언어 모델(LLM)을 활용하여 네트워크 정책을 지능적으로 생성한다. 본 시스템은 사용자 모드와 구성 모드를 통해 유연한 정책 생성을 지원하며, 다단계 사전 검증 프로세스를 통해 잘못된 정책 구성을 방지한다.

본 논문이 기여하는 바는 다음과 같다. :

- 클라우드 네이티브 환경에서 네트워크 정책을 지능적으로 생성할 수 있는 새로운 정책 생성 프레임워크에 대한 설계 및 구현을 제시한다.
- 강화된 프롬프트를 위한 엔티티 분류 모델과 도메인 특화 프롬프트 엔지니어링 메커니즘을 고안하였으며, 정책이 잘못 구성되는 것을 방지하기 위한 사전 검증 프로세스를 제공한다.

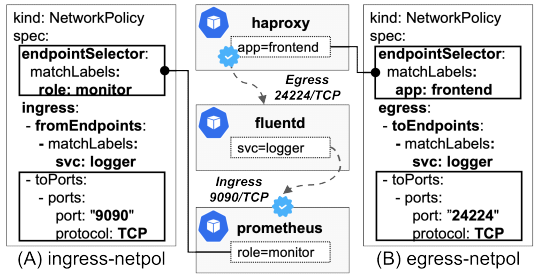


Fig. 1. The examples of two different types of network policy enforcement scenarios.

II. 배경 지식 및 문제 정의

2.1 배경 지식

2.1.1 LLM을 사용한 의도 기반 접근 제어

의도 기반 접근 제어(Intent-based Access Control, IBAC)[18]는 사용자 행동, 요청 컨텍스트, 시스템 리소스에 미치는 영향을 분석해 접근 권한을 결정하는 메커니즘이다. 특히 IBAC 구현에 자연어 처리(Natural Language Processing, NLP)와 대규모 언어 모델(Large Language Model, LLM)을 도입하면, LLM의 고급 자연어 처리 능력으로 의도를 정확히 해석할 수 있다. 그러나 대부분의 LLM은 많은 파라미터로 인해 높은 계산 비용과 처리 지연이 발생할 수 있다. 이를 해결하기 위해 상대적으로 경량화된 sLLM이 제안되었다. sLLM은 성능과 리소스 사용 사이의 균형 잡힌 타협안을 제공하며, 파인튜닝(Fine-tuning)을 통해 사전 훈련된 모델에 추가 훈련을 적용하여 특정 작업이나 도메인에 특화할 수 있으므로, 접근 제어 규칙 같은 데이터로 모델을 훈련하는 데 적용될 수 있다.

2.1.2 클라우드 네이티브 환경에서의 네트워크 정책

네트워크 정책[7]은 Kubernetes 클러스터 내의 리소스 간 통신 규칙을 정의하여 보안을 강화하고 네트워크 격리를 통해 트래픽을 관리한다. 일반적으로, Selector로 특정 레이블을 가진 리소스를 식별하여 트래픽을 허용하거나 차단한다. Fig. 1.은 네트워크 정책 시행 시나리오를 보여준다. Fig. 1.(A)에서는 'role: monitor' 레이블을 가진 파드로의 인바운드 트래픽을 제어하며, Grafana 파드가 Fluentd 서비스에서 메트릭을 수집할 수 있도록 포트(9090)와

프로토콜(TCP)로만 트래픽을 허용한다. 반면, Fig. 1.(B)는 'app: frontend'을 가진 파드에서 'svc:logger' 엔드포인트로 24224 포트만 허용하여 웹 애플리케이션이 로깅 서비스로 로그를 전송할 수 있게 한다. 네트워크 정책은 IP 기반 및 레이블 기반 트래픽 제어 외에도, HTTP 경로나 메소드 접근 제어, DNS 쿼리 필터링 등에도 적용될 수 있다.

2.2 문제 정의

클라우드 네이티브 환경에서 운용되는 마이크로서비스 아키텍처의 동적인 특성과 복잡성으로 인해, 네트워크 정책 생성은 여러 문제에 직면하고 있으며. 본 섹션에서는 다음과 같이 논의한다.

C1: 컨테이너 기반 마이크로서비스의 동적 특성에 대한 불충분한 이해. 현재 네트워크 정책 생성 시스템들은 마이크로서비스 환경을 충분히 반영하지 못한다. 로그 기반 방법론은 과거 데이터를 사용하여 실시간 서비스 관계와 통신 패턴 반영에 한계가 있다. 정적 분석 기반 방법론은 아키텍처의 복잡성과 동적 특성을 고려하지 못한다. 이러한 시스템들은 미리 정의된 스키마나 규칙에 의존하여 마이크로서비스 생태계를 포괄적으로 이해하지 못한다.

C2: 이기종 컨테이너 네트워크 인터페이스 지원의 한계. 다양한 컨테이너 네트워크 인터페이스(Container Network Interface, CNI)를 가진 컨테이너 기반 마이크로서비스 환경에서 일관된 네트워크 정책 관리는 중요하지만 복잡하다. CNI는 고유한 정책 구조와 기능을 각각 가지고 있어 네트워크 관리자가 각각을 개별적으로 이해하고 구성해야 한다. 그러므로 정책 관리 자체가 복잡해지고 일관된 네트워크 보안 태세를 유지하기 어렵다. 현재 관련 시스템들은 특정 CNI에 최적화되어 있어, 다른 CNI와 호환되도록 정책을 조정하려면 상당한 개입이 필요하다. 이는 정확한 정책을 빠르게 적용하는 것을 방해하고 인적 오류의 가능성을 높인다.

C3: 네트워크 정책 생성에서 잘못된 구성 위험. 마이크로서비스의 복잡한 의존성과 빈번한 업데이트로 잘못된 구성이 발생할 수 있으며 이를 신속히 탐지하고 수정하는 것은 매우 어렵다. 현재에 자동화된 정책 생성 시스템은 주로 정책을 생성하는 데만 초점을 맞추고 정확성 검증을 간과한다. 또한, 네트워크 관리자의 수동 개입으로 인해 기존 정책의 유효성이 즉시 손상될 수 있으며 잘못된 정책은 서비스 간 필요한 통신을 차단하거나 무단 접근을 허용하여 심각한 보안 취약점이나 서비스 중단을 초래한다.

III. Kunerva+ 설계

본 섹션에서는 시스템 설계 고려 사항과 아키텍처를 소개한다. Kunerva+는 NLP 기술을 활용하여 클라우드 네이티브 환경을 위한 네트워크 정책을 자동으로 생성하는 지능형 시스템이다.

3.1 설계 고려사항

R1: 고급 지능형 정책 생성. 사용자의 의도를 정확히 인식하고 동시에 클러스터 상태를 실시간으로 이해하여 지능적으로 네트워크 정책을 생성해야 한다. 또한, 구성 파일만을 사용하여 클러스터 내 서비스 관계를 분석하는 고도로 자동화된 인터페이스를 제공하여 사용자 개입을 최소화해야 한다. 따라서 LLM과 프롬프트 엔지니어링과 같은 자연어 처리 기술을 채택하여 사용자의 보안 요구사항을 해석하고 네트워크 정책으로 변환한다.

R2: 다양한 컨테이너 네트워크 인터페이스 지원. 특정 CNI에 종속되지 않도록 Kubernetes 네트워크 정책을 포함한 다양한 정책을 지원해야 한다. CNI 별 고유한 특성을 자동으로 인식하고 최적화된 네트워크 정책을 생성해야 한다. 이를 통해 사용자는 각 CNI에 대한 복잡한 설정을 수동으로 구성할 필요 없이 정책을 생성하고 관리할 수 있다.

R3: 다단계 정책 검증. 생성된 네트워크 정책이 실제 클러스터 환경에 적합한지, 그리고 잘못된 구성으로 이어지지 않는지 확인해야 한다. 이를 위해 네트워크 정책 자체의 구문적 정확성을 확인하고, 정책이 적용되는 클러스터 내 리소스(예: 컨테이너)의 존재와 상태를 확인하며, 정책 세부 사항이 영향을 받는 리소스와 일치하는지 확인한다. 올바르게 생성된 정책만이 시행되도록 보장하여 잘못된 정책 구성으로 인해 발생할 수 있는 충돌을 방지한다.

3.2 시스템 아키텍처 및 워크플로우

Fig. 2.에 나타난 바와 같이, Kunerva+는 Prompt Processor, Policy Processor, Policy Validator, Policy Enforcer의 네 가지 주요 구성 요소로 이루어져 있다. 시스템은 설계 고려 사항을 충족하기 위해 두 단계로 작동한다.

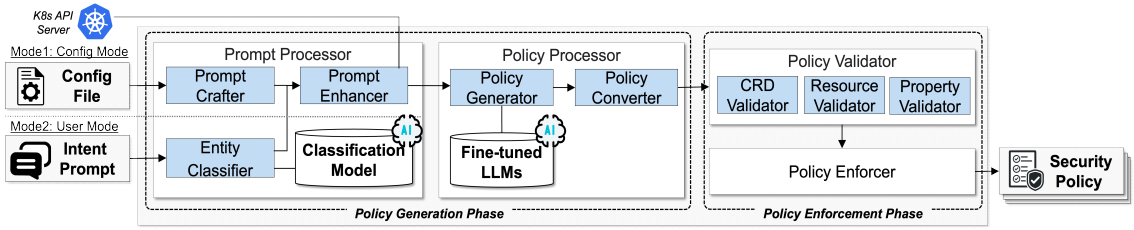


Fig. 2. Overall architecture and its workflow of Kunerva+ with four key components: (i) prompt processor, (ii) policy processor, (iii) policy validator, and (iv) policy enforcer. Additionally, our system includes two operational phases: policy generation and policy enforcement

Table 1. The summary of the datasets for network policies (Policy) and intent prompts (Intent).

Type	#Origin	#Total	#Size
Policy	857	166,124	187.3 MB
Intent	857	50,060	93.2 MB

첫째, 정책 생성 단계는 네트워크 정책 생성을 자동화한다. 여기서는 직접적인 사용자 입력(사용자 모드)이나 클러스터의 리소스 정보(구성 모드)를 지원한다. 사용자 모드에서는 사용자가 복잡한 정책 구문을 이해할 필요 없이 자연어로 정책 의도를 제공하면, BERT[25] 기반 Entity Classifier에서 필요한 엔티티를 식별한다. 구성 모드에서 Prompt Crafter가 클러스터의 리소스 정보(메타데이터, 사양 등)를 수집한다. 각 리소스 유형에 해당하는 미리 정의된 템플릿을 통해 초기 프롬프트가 자동으로 생성한다. 두 모드에서 생성된 프롬프트는 Prompt Enhancer에 의해 향상되는데, Kubernetes API를 통해 실시간 클러스터 정보를 조회하고 리소스 간 관계를 추론한다. 이를 프롬프트에 통합하여 클러스터 특화된 프롬프트로 변환한다. 향상된 프롬프트는 Policy Generator로 전달되며, 파인 튜닝된 LLM을 사용하여 현재 클러스터에 최적화된 네트워크 정책을 생성한다. 생성된 정책은 Policy Converter가 CNI에 맞게 호환시킨다.

둘째, 정책 시행 단계는 생성된 정책 검증과 적용을 담당한다. Custom Resource Definition (CRD[4]) Validator가 첫 방어선 역할을 하며, Policy CRD에 정의된 스키마에 대해 정책 자체의 구조적 무결성을 검사한다. Resource Validator가 정책이 적용될 클러스터 내 리소스의 존재와 상태를 확인한다. Property Validator는 정책 세부 사항이 영향을 받는 리소스와 일치하는지 확인하여 정책이 의도된 리소스에 적용되도록 보장한다. 검증을 통과한 정책은 Policy Enforcer가 Kubernetes API를 통해 클러스터에서 시행한다.

IV. Kunerva+ 시스템 세부사항

4.1 생성형 AI 기반 모델 학습

Kunerva+의 핵심은 사용자의 의도를 정확히 파악하고 적절한 네트워크 정책으로 생성하는 것이다. 본 시스템은 개체명 인식과 생성형 AI 기술 중 하나인 LLM을 결합한 고급 의도 처리 기술을 제안한다.

4.1.1 데이터셋 구조 및 전처리

일반적인 언어 모델 데이터셋은 클라우드 네이티브 환경을 위한 네트워크 정책의 특수한 구조와 용어를 적절히 반영하지 못한다. 따라서 CNI의 네트워크 정책 구조와 문법에 대해 학습 모델을 훈련시키기 위한 맞춤형 데이터셋을 개발했다. Table. 1.와 같이, GitHub과 같은 오픈 소스에서 총 857개의 원본 네트워크 정책이 수집하였다. 이를 Selector와 Rule으로 분리하고 데이터의 고유성을 보장하기 위해 중복 요소를 제거하였다. 그리고 셔플링을 통해 데이터를 증식하였다. 초기 데이터셋을 확장함으로써 모델의 일반화 능력과 다양한 시나리오에 대한 적응성을 향상시켰으며, 그 결과 166,124개의 정책 샘플을 추출하였다. 이 샘플들은 특수 토큰을 사용하여 입력과 출력 쌍 (의도 프롬프트, 정책 출력)을 구분했으며, Classifier를 훈련 시키기 위해 50,060개의 의도 프롬프트를 무작위로 선택했다. 그 후, BIESO 태깅을 통해 의도 샘플에 10개의 엔티티 유형(Fig. 4.) 중 해당하는 레이블을 할당했다.

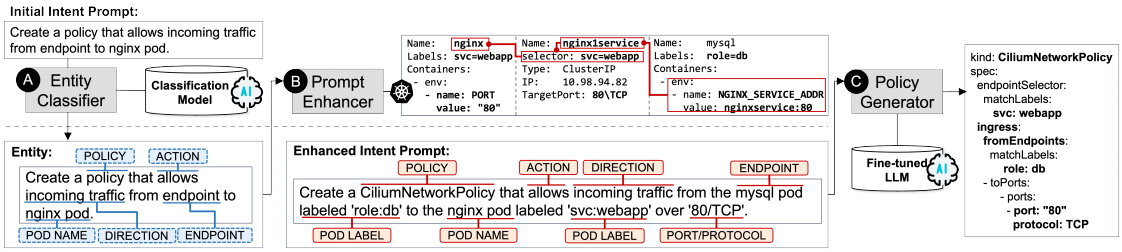
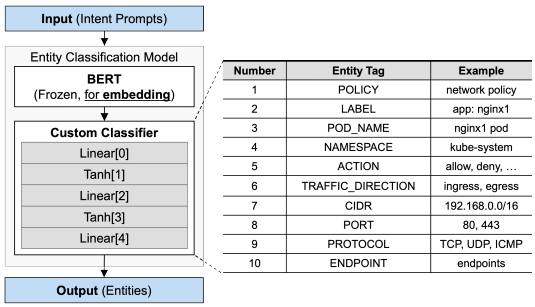


Fig. 3. The example procedure for generating a network policy, from prompt enhancement to the use of fine-tuned LLMs. The entities in blue dotted lines are reconfigured to entities in red solid lines.



Number	Entity Tag	Example
1	POLICY	network policy
2	LABEL	app: nginx1
3	POD_NAME	nginx1 pod
4	NAMESPACE	kube-system
5	ACTION	allow, deny, ...
6	TRAFFIC_DIRECTION	ingress, egress
7	CIDR	192.168.0.0/16
8	PORT	80, 443
9	PROTOCOL	TCP, UDP, ICMP
10	ENDPOINT	endpoints

Fig. 4. The design of the entity classification model for network policy-specialized NER.

4.1.2 도메인 특화 개체명 인식(NER)

네트워크 정책을 정확히 생성하기 위해, 의도 프롬프트에서 정책 관련 엔티티를 식별하는 것이 중요하다. Fig. 4.와 같이 BERT 기반[23,24]의 NER[22]을 고안했다. 이 모델은 네트워크 정책 도메인에 특화되어 있어 일반 NER 모델이 놓칠 수 있는 정책 관련 엔티티를 유효하게 식별할 수 있다. 모델 아키텍처는 BERT Embedding과 Custom Classifier로 구성된다. Embedding Layer는 사전 훈련된 BERT-base 모델(110M 파라미터)을 활용하며, 과적합을 방지하면서 강력한 언어 이해 능력을 활용하기 위해 가중치를 고정하였다. Custom Classifier Layer는 Linear-tanh 구조로 구성되어 있으며, 다양한 엔티티 간의 복잡한 관계를 효과적으로 모델링할 수 있다. 이 모델은 'LABEL', 'POD_NAME'과 같은 네트워크 정책에 최적화된 10개의 엔티티 클래스를 식별하도록 훈련한다.

4.1.3 LLM 최적화 및 맞춤형 프롬프트 엔지니어링

Kunerva+는 LLM 최적화와 맞춤형 프롬프트 엔지니어링을 통해 정확한 네트워크 정책 생성을 달성한다. 이를 위해 LoRA(Low-Rank Adaptation)[25]을 활용하여 모델의 일부 파라미터만을 효율적으로 업데이트한다. Optuna[3]로 하이퍼 파라미터를 최적화하고, 랭크 8의 LoRA 행렬을 모델의 주요 층에 추가하여 전체 모델 크기의 증가를 약 0.2% 이하로 최소화하여 파인튜닝을 진행하였다. 이 방법은 60-80억 개의 파라미터를 가진 3개의 오픈 text-to-text Decoder 생성 모델에 적용했으며, 전체 모델 파라미터의 약 0.5%만 업데이트하여 네트워크 정책 생성에 특화된 모델을 얻었다.

제안한 맞춤형 프롬프트 엔지니어링은 의도 프롬프트를 명확히 이해하고 클러스터의 실시간 상태와 CNI 특성을 반영하여 이를 정제하는 것을 포함한다. Fig. 3.은 Kunerva+이 초기 프롬프트를 어떻게 정제하는지 보여준다. 먼저, 도메인 특화 NER 모델을 사용하여 초기 의도 프롬프트에서 핵심 엔티티를 추출한다(A). 즉, 'nginx(POD_NAME)', 'incoming(DIRECTION)'와 같은 네트워크 정책에 중요한 엔티티들이 프롬프트에서 정확히 식별한다. 그리고 추출된 엔티티를 기반으로 Kubernetes API를 통해 클러스터의 실시간 상태를 고려하여 관련 리소스에 대한 상세를 조회하고 리소스 간의 관계를 추론한다. 예를 들어 'nginx' 파드의 레이블(svc=webapp)을 통해 서비스 정보를 조회한다. 분석된 정보를 종합하여 초기 프롬프트를 구체적이고 명확한 향상된 의도 프롬프트로 정제한다(B). 향상된 의도 프롬프트는 파인 튜닝된 LLM에 입력되고, LLM은 상세한 의도 프롬프트를 기반하여 정책의 각 필드를 구성함으로써 정책을 생성한다(C).

Kunerva+는 여러 CNI의 네트워크 정책 문법과 기능 차이를 고려하기 위해, Kubernetes API를 통해 클러스터에 현재 설치된 CNI를 자동으로 감지한다. 이는 클러스터 구성 조회나 CNI 관련 CRD 확인을 통해 이뤄진다. 의도 프롬프트 파싱 단

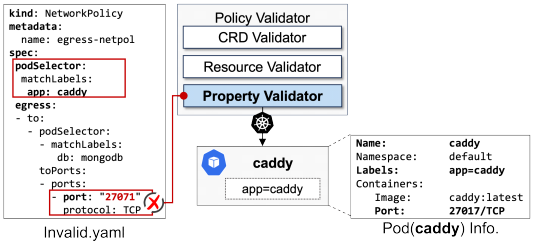


Fig. 5. The example of property validation for detecting the misconfigured policy.

계에서 프롬프트에 포함된 CNI 특화 키워드를 식별하고 기존 정책의 의도를 유지하면서 CNI에 맞게 자동 변환한다. 예를 들어, Cilium의 'endpoint'가 감지되었지만, 현재 설치된 CNI가 Calico라면, Prompt Enhancer는 Calico의 'selector'에 맞게 변환하고 LLM을 통해 해당 CNI 문법과 구조를 따르는 정책을 생성한다. 이를 통해 사용자는 특정 CNI를 모르거나 다른 용어를 사용해도 올바른 네트워크 정책을 생성할 수 있다.

4.2 다단계 정책 검증

본 시스템은 네트워크 정책 시행의 안전성을 보장하기 위해 CRD 검증, Resource 검증, Property 검증으로 구성된 세 가지 검증 절차를 제안한다. 이로써 생성된 정책이 단순히 문법적으로 올바른지 확인하는 것을 넘어, 실제 클러스터 환경에서 의도한 대로 동작하는지 보장하는 것을 목표로 한다. Kunerva+로 생성된 정책뿐만 아니라 관리자가 작성한 정책도 적용되며, 모든 네트워크 정책의 철저한 검증을 보장한다. 각 단계에서 식별된 모든 문제는 즉시 기록되며, 관리자에게 명확한 오류 메시지와 수정 사항이 제공된다. 첫 번째 CRD 검증에서는 정책의 문법적 정확성과 구조적 무결성을 확인하여 정책이 CRD 스키마를 준수하는지 검사한다. Fig. 5.와 같이, 'kind', 'spec' 등 필수 필드의 존재 여부와 형식을 검토하며, 'podSelector'와 'egress' 규칙의 구조적 정합성을 확인한다. 두 번째 Resource 검증에서는 Kubernetes API를 활용하여 정책이 참조하는 리소스(예: 파드, 서비스)가 실제로 존재하는지 확인하며, 정책 적용에 적합한 상태인지 평가한다. 예를 들어 'app: caddy' 레이블을 가진 파드가 클러스터에 존재하는지 확인한다.

마지막으로, Property 검증은 정책의 세부 속성이 클러스터 환경과 일치하는지 확인한다. 지정된 포트가 해당 서비스에 올바르게 리스닝하고 있는지, 프로토콜이 정확히 설정되었는지 검사한다. 또한 정책에 지정된 IP 범위의 유효성을 검사하고 의도된 네트워크 세그먼트를 정확하게 대상으로 하는지 확인한다. 'Invalid.yaml'에 지정된 포트 27071이 대상 파드의 실제 포트와 일치하지 않아 오류가 감지되고 정책이 시행되지 않는다. 이후, 오류 로그 정보와 실제 리소스의 속성 사이의 상관관계를 분석하여 올바른 리스닝 포트 정보를 제안한다.

Table 2. The summary of LLMs fine-tuned for policy generation.

Name	#Size	#Params
DeepSeek/Deepseek-coder-7b-instruct-v1.5 [14]	14GB	6.91B
MistralAI/Mistral-7B-Instruct-v0.2 [15]	15GB	7.24B
Google/codegemma-7b-it[16]	17GB	8.54B

V. 평 가

5.1 테스트 환경 및 구현

파인튜닝을 위한 NVIDIA Hopper H100, AMD EPYC 9224 CPU, 그리고 마이크로서비스 환경설정을 위한 2TB SSD를 갖춘 고성능 서버 환경에서 평가하였다. Kubernetes 클러스터는 마스터 노드 1개와 워커 노드 2개로 구성되었으며, Ubuntu 24.04 가상 머신에 호스팅하였다. Kunerva+는 Cilium[5] 또는 Calico[6]와 함께 마스터 노드에 배포되었고, 테스트 애플리케이션들은 두 개의 워커 노드에 분산되었다. 테스트 애플리케이션으로는 클라우드 네이티브 애플리케이션 데모인 Online Boutique[10]를 사용하였으며, 이는 11개의 독립적으로 배포 및 관리되는 서비스로 구성되어 제품 검색 등의 전자상거래 기능을 제공한다.

Kunerva+의 실현 가능성과 효과성을 검증하기 위해 Go와 Python을 통해 약 4,500줄의 코드로 인스턴스를 구현하였다. 현재 지원되는 정책에는 Kubernetes 네트워크 정책[7], Cilium 네트워크 정책[8], Calico 네트워크 정책[9]을 모두 다룬다. 초기 의도 프롬프트를 향상하기 위해, BERT[23]와 RoBERTa[24] 기반 엔티티 분류 모델을 개발하였으며, 네트워크 정책 생성에 사용된 Table. 2.에 나타난 세 개의 LLM은 사전에 파인 튜닝하였다.

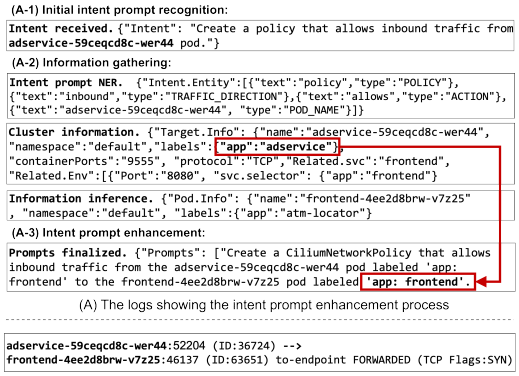


Fig. 6. The results of enhanced prompting and network log from correct policy enforcement.

5.2 기능적 정확성

5.2.1 프롬프트 개선 처리

Kunerva+의 프롬프트 엔지니어링 능력을 평가하기 위해, 특정 파드 간 트래픽을 허용하는 네트워크 정책 생성을 분석하였다. Online Boutique와 같이 잘 정의된 마이크로서비스의 경우, 서비스 배포 시 상세한 메타데이터와 레이블이 제공된다고 가정하였다. 우선 초기 의도 프롬프트 인식 단계에서 시스템은 사용자 또는 구성 모드를 통해 정책 요구사항을 초기 프롬프트로 받는다. 둘째, 정보 수집 단계에서 시스템은 Kubernetes API를 통해 관련 파드에 대한 상세 정보를 수집하고 분석한다. 셋째, 의도 프롬프트 향상 단계에서는 수집된 정보를 바탕으로 초기 프롬프트를 정제한다. Fig. 6.은 초기 프롬프트 생성 이후의 프롬프트 엔지니어링 과정과 그 결과를 보여준다. 간단한 초기 정보로 시작하여, 시스템은 Kubernetes API를 통해 관련 파드에 대한 상세 데이터를 수집하고 분석한다. 예를 들어, Fig. 6.(A-2)에서 보이듯이, Kunerva+은 'app: adservice' 파드가 'app: frontend' 파드와 관련이 있음을 식별한다. 결과적으로, 향상된 프롬프트에는 초기 프롬프트에 명시되지 않은 중요한 세부사항들, 예를 들어 Cilium Network Policy 유형의 명세, 정확한 파드 레이블, 특정 트래픽 흐름 방향 등이 포함된다. 이 향상된 프롬프트의 정확성은 Fig. 6.(B)에 나타난 실제 네트워크 로그를 통해 검증된다. 로그에 기록된 트래픽 패턴은 향상된 프롬프트에 명시된 파드 관계와 정확히 일치한다. 이는 Kunerva+의 프롬프트 엔지니어링 능력이 복잡한 마이크로서비스 환경에서 네트워크 정책을 효과적으로 생성할 수 있는 기반을 갖출 수 있음을 보인다.

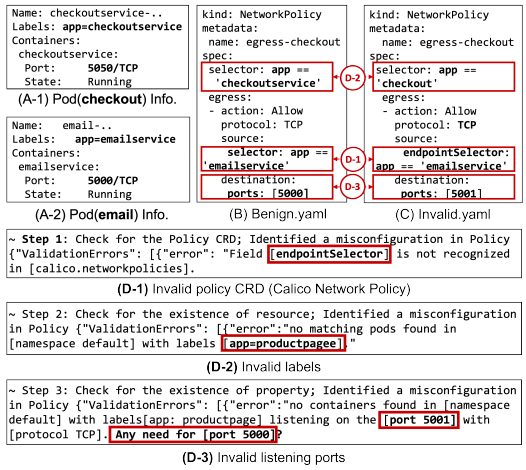


Fig. 7. The results of policy validation scenarios: valid(B), invalid(C), and the detection logs(D).

5.2.2 CRD, 리소스 및 속성 검증

본 실험은 네트워크 정책의 CRD, 리소스, 속성 측면을 검증하는 능력을 평가한다. Fig. 7.은 검증 프로세스의 테스트 시나리오와 결과를 보여준다. Fig. 7.의 상단은 'default' 네임스페이스에 있는 'checkoutservice'와 'emailservice' 파드에 대한 정보(A-1, A-2)를 제시하며, 각각 5050/TCP, 5000/TCP에서 서비스 중이다. 이를 바탕으로, Fig. 7.(B)는 올바른 레이블, 포트, 프로토콜을 가진 유효한 네트워크 정책을 보여주지만, (C)는 지원되지 않는 필드, 존재하지 않는 파드, 잘못된 속성을 가진 잘못 구성된 정책을 보여준다. 네트워크 정책 (C)을 시행하려 할 때, 시스템은 CRD 검증 단계에서 구조적 정책 스키마 오류를 성공적으로 감지하며 지원되지 않는 필드 'endpointSelector'의 사용을 식별한다(D-1). 또한 정책에 사용된 리소스 레이블 값의 오타('checkoutservice' 대신 'checkout')를 식별하여 해당 리소스가 클러스터 내에 존재하지 않음을 나타낸다 (D-2). 그리고 정책의 포트 속성에 지정된 포트 번호 5001이 'emailservice' 파드가 서비스하는 포트와 일치하지 않음을 감지한다(D-3). 각 단계에서 식별된 오류는 로그를 통해 관리자에게 보고되어 문제를 인식하고 수정할 수 있게 한다.

Table 3. The summary of the fine-tuned LLMs performance with metrics: BLEU, ROUGE-1, ROUGE-2 and ROUGE-L.

Type	Model Name	BLEU	ROUGE-1	ROUGE-2	ROUGE-L
Baseline	Deepseek-coder-7b-instruct-v1.5	0.52	0.71	0.57	0.69
	Mistral-7B-Instruct-v0.2	0.10	0.34	0.15	0.29
	codegemma-7b-it	0.49	0.72	0.59	0.70
Fine-tuning	Deepseek-coder-7b-instruct-v1.5	0.76	0.87	0.78	0.83
	Mistral-7B-Instruct-v0.2	0.46	0.57	0.50	0.54
	codegemma-7b-it	0.72	0.85	0.80	0.84

5.3 성능

Kunerva+의 성능 평가는 정책 생성의 정확성, 특히 의도 인식과 정책 생성 능력에 초점을 맞추기 위해 엔티티 분류와 LLM 기반 정책 생성 성능을 중심으로 진행하였다.

5.3.1 분류 모델 성능

본 평가에서는 의도 프롬프트에서 주요 엔티티를 정확히 식별하는 능력을 평가하여 엔티티 분류기의 성능을 테스트하였다. Table. 1.과 같이, 전체 의도 샘플의 20%를 무작위로 선택하여 BERT와 RoBERTa 기반 모델의 성능을 비교하였다. 평가 과정에서 각 모델은 주어진 의도 프롬프트에 대한 엔티티 태그를 예측하였고, 이 예측은 테스트 세트에 포함된 실제 레이블(ground truth)과 비교되었다. 이 비교를 통해 각 모델의 정확도, 정밀도, 재현율, F1 점수를 계산하였으며, 그 결과는 Table. 4.에 제시되어 있다. BERT 기반 분류 모델은 97.1%의 정확도, 97.4%의 정밀도, 97.2%의 재현율, 97.3%의 F1 점수를 달성하였다. RoBERTa는 BERT보다 약간 낮지만 유사하게 높은 성능을 보였다. 두 모델 모두 97%를 초과하는 F1 점수를 보여 정밀도와 재현율 사이의 균형을 나타냈다. 이는 Kunerva+가 파드 이름, 포트 번호와 같은 네트워크 정책의 중요한 엔티티를 정확하게 식별할 수 있음을 시사한다.

5.3.2 파인튜닝된 LLM 성능

향상된 의도 프롬프트를 기반으로 생성된 네트워크 정책의 정확성을 평가하기 위해, 기본 모델과 파인 튜닝된 모델의 성능을 BLEU, ROUGE-1, ROUGE-2, ROUGE-L 메트릭을 사용하여 비교하였다. 정책 생성 능력에서 파인튜닝의 효과를 평가하는 것을 목표로 하였으며, 결과는 Table. 3.과 같다. 테스트 된 모델들은 Kunerava+가 제공한 파인튜닝 과정을 거친 후 상당한 성능 향상을 보였다. 특히 Deepseek coder 7b 모델은 파인튜닝 후 가장 높은 성능을 보였으며, BLEU가 46%(0.52에서 0.76), ROUGE-1가 22% (0.71에서 0.87) 증가하여 가장 높은 성능을 보였다. 평균적으로 BLEU는 약 70%, ROUGE-1는 25% 정도 향상되었다.

Table 4. The summary of entity classifier performance with metrics: Accuracy, Precision, Recall, and F1 Score.

Classifier	#Acc	#Prec	#Rec	#F1
BERT	0.971	0.974	0.972	0.973
RoBERTa	0.961	0.968	0.961	0.965

BLEU와 ROUGE 지표의 현저한 개선은 생성된 정책이 의도된 요구사항을 정확히 반영하면서도, 다양한 CNI 환경에서 요구되는 문법적 정확성과 완전성을 갖추었음을 보여준다. 결론적으로, 파인튜닝이 정책 생성에 대해 모델을 적절히 특화했으며 다양한 CNI 환경에서 신뢰할 수 있는 네트워크 보안 정책 시행 도구로서의 잠재력을 보여준다.

VI. 관련 연구

네트워크 관리의 복잡성이 증가하면서 의도 기반 네트워크 관리에 대한 연구가 활발히 이뤄지고 있다. Jacobs 등[21]은 자연어 의도를 중간 언어로 변환한 후 네트워크 구성으로 변환하는 시스템을 제안하였다. Otterize[11]는 의도된 접근 정책을 선언적으로 정의하고 이를 네트워크 정책으로 자동 변환한다. 그러나 이러한 연구들은 컨테이너 기반 마이크로서비스의 특정 특성(예: 빈번한 업데이트)에 맞지 않으며 의도를 공식화하는 데 상당한 노력이 필요하다.

컨테이너화된 환경에서 자동화된 정책 생성에 대

한 연구도 진행되었다. Xu 등[19]은 로그 기반 토폴로지 그래프 생성, 기계 학습 기반 속성 마이닝, 트래픽 관리 기반 정책 업그레이드 메커니즘을 사용하여 액세스 로그를 기반으로 인증 정책을 자동으로 생성하는 방법을 제안하였다. Li 등[20]은 정적 분석 기반 요청 추출 메커니즘을 통해 마이크로서비스 간 상호작용을 분석함으로써 접근 제어 정책을 자동 생성하는 시스템을 개발하였다. Open Policy Agent (OPA)[12]는 정책 기반 접근 제어 및 검증을 하며, GateKeeper[13]는 OPA를 통해 클러스터의 요청을 검증하고 무단 접근을 차단한다.

본 연구에서 제안하는 Kunerva+는 이전 연구들과 달리 클라우드 네이티브 환경에 특화된 엔드-투-엔드 접근 방식을 제공하는 LLM 기반 네트워크 정책 생성 시스템이다. 기존의 로그 기반 분석이나 정적 분석 방식은 실시간 변화하는 마이크로서비스 환경에서 한계를 보이지만, Kunerva+는 특화된 NER 모델과 고급 프롬프트 엔지니어링을 통해 실시간 리소스 관계를 추론하고 다양한 CNI 환경에서의 효과적인 정책 생성 메커니즘을 제안한다.

VII. 결 론

본 연구는 실제 클라우드 네이티브 환경에서 네트워크 정책을 생성하기 위한 자동화된 소프트웨어 프레임워크인 Kunerva+를 제안한다. 복잡한 클라우드 네이티브 환경에 특화된 파인 튜닝된 LLM과 정교한 프롬프트 엔지니어링을 통합한 최초의 시도로, 기존의 로그 기반 또는 정적 분석 방법과 차별화된다. 또한, 정책 생성을 넘어 우리 시스템은 잘못된 구성을 사전에 방지하기 위한 다단계 검증 프로세스를 포함한다. Kunerva+는 엔티티 분류와 정책 생성 모두에서 인상적인 결과를 보여주었다. 본 연구는 의도 기반 보안 정책 생성을 위한 귀중한 참조 구현을 제공하며, 클라우드 네이티브 환경에서의 보안 정책 관리에 대한 새로운 방향을 제시한다.

향후 연구에서는 정책 우선순위 충돌 해결 메커니즘을 추가로 다루어, 다양한 CNI 환경에서 정책 적용 간 충돌을 완화하고, 사용자 의도가 더 정확히 반영될 수 있도록 시스템을 확장할 계획이다.

References

[1] Kubernetes, “Kubernetes, ”https://kubernetes.io/, 2024.09.

[2] CNI, “Container Network Interface,” https://www.cni.dev/, 2024.09.

[3] Optuna, “Optuna - A hyperparameter optimization framework,” https://optuna.org/, 2024.09.

[4] Kubernetes, “CustomResourceDefinitions,” https://kubernetes.io/docs/tasks/extend-kubernetes/custom-resources/custom-resource-definitions/, 2024.09.

[5] Cilium, “Cilium - eBPF-based Networking, Observability, Security,” https://cilium.io/, 2024.09.

[6] Tigera, “Project Calico | Tigera - Creator of Calico,” https://www.tigera.io/project-calico/, 2024.09.

[7] Kubernetes, “Network Policies,” https://kubernetes.io/docs/concepts/services-networking/network-policies/, 2024.09.

[8] Cilium, “Cilium Network Policy,” https://docs.cilium.io/en/latest/security/policy/, 2024.09.

[9] Tigera, “Calico Network Policy,” https://docs.tigera.io/calico/latest/network-policy/get-started/calico-policy/calico-network-policy/, 2024.09.

[10] Google Cloud Platform “Online Boutique,” https://github.com/GoogleCloudPlatform/microservices-demo, 2024.09.

[11] Otterize, “Otterize: Zero-friction Kubernetes security,” https://otterize.com/, 2024.09.

[12] Open Policy Agent, “Open Policy Agent,” https://www.openpolicyagent.org/, 2024.09.

[13] Open Policy Agent, “Gatekeeper - Policy Controller for Kubernetes,” https://github.com/open-policy-agent/gatekeeper, 2024.09.

[14] Hugging Face, “deepseek-ai/deepseek-coder-7b-instruct-v1.5,” https://huggingface.co/deepseek-ai/deepseek-coder-7b-instruct-v1.5, 2024.09.

[15] Hugging Face, "mistralai/Mistral-7B-Instruct-v0.2m," https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2,2024.09.
[16] Hugging Face, "google/codegemma-7b-it," https://huggingface.co/google/codegemma-7b-it, 2024.09.
[17] R. Hat, "The state of Kubernetes security report: 2024 edition," https: //www.redhat.com/en/engage/state-kubernetes-security-report-2024, 2024.09.
[18] A. Almehmadi and K. El-Khatib, "On the possibility of insider threat prevention using intent-based access control(ibac)," IEEE Systems Journal, vol. 11, no. 2, pp. 373 - 384, May. 2015.
[19] LI, Xing, et al. Automatic policy generation for {Inter-Service} access control of microservices. In: 30th USENIX Security Symposium (USENIX Security 21). Aug. 2021. p. 3971-3988.
[20] S. Xu, Q. Zhou, H. Huang, X. Jia, H. Du, Y. Chen, and Y. Xie, "Log2policy: An approach to generate fine-grained access control rules for microservices from scratch," in Proceedings of the 39th Annual Computer Security Applications Conference, Dec. 2023, pp. 229 - 240.
[21] A. S. Jacobs, R. J. Pfitscher, R. H. Ribeiro, R. A. Ferreira, L. Z. Granville, W. Willinger, and S. G. Rao, "Hey, lumi! using natural language for {intent-based} network management," in 2021 USENIX Annual Technical Conference, Jul. 2021, pp. 625 - 639.
[22] J. Li, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition," IEEE transactions on knowledge and data engineering, vol. 34, no. 1, pp. 50 - 70, Mar. 2020.
[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, May. 2019.
[24] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," arXiv preprint arXiv:1907.11692, Jul. 2019.
[25] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," arXiv preprint arXiv:2106.09685, Oct. 2021.

〈저자소개〉

김 봄 (Bom Kim) 학생회원
2023년 8월: 인천대학교 컴퓨터공학부 학사
2023년 9월~현재: 인천대학교 일반대학원 컴퓨터공학과 석사과정
〈관심분야〉 클라우드 보안, 네트워크 보안

이 승 수 (Seungsoo Lee) 종신회원
2014년 2월: 숭실대학교 컴퓨터학부 학사
2016년 2월: KAIST 정보보호대학원 석사
2020년 8월: KAIST 정보보호대학원 박사
〈관심분야〉 클라우드 보안, 네트워크 보안