IntentShield: 쿠버네티스를 위한 인텐트 기반 보안 정책 오퍼레이터 프레임워크

김봄[1], 이승수[2]

인천대학교 (대학원생[1], 교수[2])

IntentShield: An Intent-Powered Security Policy Operator Framework for Kubernetes

Bom Kim[1], Seungsoo Lee[2]

Incheon National University(Graduate Student[1], Professor[2])

요 약

쿠버네티스는 현대 클라우드 환경에서 핵심 구성 요소 중 하나이며, 그 구성의 복잡성 때문에 수동적인 정책 관리 방식은 일관성과 정확성의 부재를 동반하여 시스템을 취약하게 만든다. 본 논문에서는 IntentShield라는 쿠버네티스를 위한 인텐트 기반 정책 오퍼레이터 프레임워크를 제안한다. 이 프레임워크는 사용자의 의도를 명확히 파악하고 이를 바탕으로 쿠버네티스 보안 정책을 자동으로 생성 및 적용하는 메커니즘을 포함하고 있다. 특히, 'SecurityIntent'라는 CRD을 통해 사용자는 복잡한 구성 없이 보안 요구 사항을 선언적으로 표현할 수 있다. 본 논문의 실험적 평가는 실제 환경에서 구현된 IntentShield 프로토타입을 실험함으로써 실제 환경에서의 적용 가능성과 정책 자동 관리에 대한 효과를 검증한다.

I. Introduction

컨테이너 환경의 끊임없는 변화 속에서 쿠버네티스는 컨테이너 애플리케이션의 배포 및 관리를 혁신화하는 시스템으로 주목받고 있다. 쿠버네티스의 중요성은 명백하지만 복잡성 때문에 특히, 보안 측면에서의 문제점이 강조되고 있다. 쿠버네티스의 다양한 리소스와 구성을 관리하는 것은 전문적인 지식 없이는 적절하게 설정하기 어렵기 때문에 보안 취약점과 관리의 어려움이 있으며, 실수나 누락은 보안 위험으로 이어질 수 있다. 예를 들어, 정책, RBAC 및 시크릿 관리와 같은 복잡한 구성 요소가 올바르게 설정되지 않을 경우 데이터 유출이나 서비스 중단 같은 보안 사고가 발생할 수 있다.

또한, 기존의 보안 정책 관리 방식은 관리자가 수동으로 정책을 설정하고 업데이트해야 한다. 이 과정에서 일관성과 정확성을 유지하기가 어려우며 빠르게 변화하는 환경에서 실시간으로 보안 요구사항에 대응하기 어렵다.

본 논문에서는 복잡하고 전문적인 지식을 요구하는 기존의 방식에서 벗어나, 보다 사용자 측면에서 직관적이고 자동으로 정책을 관리하는 IntentShield라는 쿠버네티스 정책 오퍼레이터 시스템을 소개한다. 또한 IntentShield 프로토타입을 구현하고 실제 쿠버네티스 환경에서의 테스트를 통해 그 효과를 입증한다.

본 논문의 주요 기여는 다음과 같다:

1. 사용자 중심의 보안 정책 설계: 새로운 CRD을 설계하여, 사용자가 자신의 의도를 명확하게 표현하도록 지원한다. 이를 통해 보안 요구 사항과 실제 보안 정책 사이의 간극을 줄이고, 보안 정책 설정의 복잡성을 낮출 수 있다.

2. 자동화와 지속적인 보안 강화: 사용자의 의도를 분석하여 자동으로 보안 정책을 생성하고 적용하는 IntentShield를 통해 환경의 변화에 따라 클러스터의 보안 상태를 지속적으로 강화한다. 이러한 자동화는 인적 실수를 방지하고, 시스템의 일관성과 신뢰성을 보장한다.

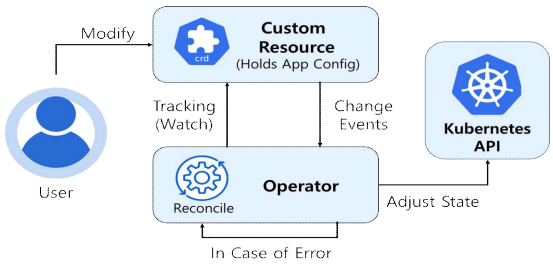

Fig 1. Kubernetes Operator Workflow from User to API Adjustment

II. Background

2.1 Intent Based Access Control(IBAC)[1)]

IBAC[1][2]는 사용자의 의도를 직접적으로 반영하여 접근 허용 여부를 결정하는 접근 제어 메커니즘이다. 전통적인 접근 제어 방식인 RBAC(Role-Based Access Control), ABAC(Attribute-Based Access Control)는 사용자의 역할, 속성 또는 자격을 토대로 접근 권한을 부여한다. 이러한 방식은 정적이고 미리 정의된 규칙에 의존하며 맥락이나 환경 변화에 따라 조정되지 않는다. 이와 달리, IBAC은 사용자의 의도와 수행하려는 목적에 따른 조건을 맥락적으로 평가하여 접근 권한을 유연하게 조절한다.

IBAC의 목표는 '의도'에 집중하는 것이다. 시스템에서 단순한 권한 부여의 문제로 판단하고 사용자의 요청을 처리하는 게 아니라 사용자의 요구, 요청이 시스템 리소스에 미치는 영향, 접근 목적 등의 맥락을 고려한다. 예컨대, 사용자가 특정 클라우드 서비스에 대한 접근을 요청하는 경우 IBAC은 해당 요청의 의도를 분석하고 프로젝트의 상태, 마감 시간, 요청된 리소스와의 관계 등 여러 변수를 고려하여 적절한 접근 권한을 부여한다.

IBAC는 사용자의 접근 요청에 대한 문맥(위치, 시간, 장치 등)을 고려하여 의도에 따라 접근 권한을 동적으로 부여하므로 변동성이 큰 환경에서 효과적이며 유연하다. 사용자의 의도를 파악하여 미래의 접근 요청을 예측하고, 적절한 접근 권한을 부여하는 것도 가능하다.

1) 본 논문에서는 앞으로 간략하게 'IBAC', 'CRD', 'CR'이라고 표기한다.

2.2 Kubearnetes Operator

쿠버네티스 컨트롤러(Controller)는 쿠버네티스의 핵심 컴포넌트 중 하나이며, 선언된 상태(Desired State)와 현재 상태(Current State)를 일치시키기 위해 클러스터 내의 특정 리소스의 상태를 지속해서 모니터링하며 특정 작업을 수행한다. 클러스터의 상태 변환에 따라 컨트롤러는 자동 반응하는 기능을 가지나, 사용자 정의 애플리케이션 또는 서비스를 관리하기 위해서는 추가 로직이 필요하다.

Fig 2의 오퍼레이터(Operator)는 컨트롤러의 확장된 개념으로, 특정 애플리케이션의 전체 수명 주기를 관리하기 위해 설계된 커스텀 컨트롤러이다. 쿠버네티스의 자동화와 확장 기능을 최대화하여 복잡한 관리 작업을 단순화한다. Custom Resource Definition (CRD)를 통해 쿠버네티스의 API를 확장하며, Custom Resource (CR)에 대한 사용자 정의된 컨트롤러 기능을 제공한다. CRD를 통해 쿠버네티스의 기본 오브젝트 외에 애플리케이션 또는 서비스에 특화된 설정을 포함하는 커스텀 오브젝트를 정의할 수 있다. CR은 CRD를 통해 정의한 새로운 리소스 오브젝트의 인스턴스이다.

오퍼레이터가 동작하는 방식은 다음과 같다. 사용자는 특정 애플리케이션 또는 서비스에 알맞은 CRD를 정의하고 쿠버네티스에 등록한다. 이후 오퍼레이터는 해당 CRD와 연관된 CR의 생성, 업데이트, 삭제 등의 이벤트를 감지한다. 이벤트가 감지될 때, 오퍼레이터는 미리 정의된 로직에 따라 필요한 작업을 진행한다.

III. System Design

3.1 CRD Design

본 시스템의 메인 중 하나는 의도가 정의될 파일인데, Fig 2와 같이 'SecuityIntent'라는 CRD를 통해 네트워크 및 시스템 보안 정책을 작성할 수 있는 선언적 구조로 설계하였다. SecurityIntent는 'system'과 'network' 필드로 구성된다. 'system'은 시스템 정책 관련 의도를 정의하며, 'network'는 네트워크 정책 관련 의도를 명시한다. 공통 하위 필드 'enforce'에 원하

```
apiVersion: intent.k8s.cclab.com/v1
kind: SecurityIntent
metadata:
  name: my-intent-1
  namespace : my-intent-netpol
spec:
  network:
    - enforce : “Allow TCP 90”
...
  system:
    - enforce : “Block attempts to execute the '/bin/sleep' process.”
...
```

Fig 2. The Example of SecurityIntent (YAML)

는 정책을 표현하도록 하였다. 예를 들어, 네트워크에서 TCP 90 포트 트래픽을 허용하거나 시스템에서 특정 프로세스 실행을 차단하는 등의 의도를 enforce 필드에 입력하면 된다.

이러한 설계는 사용자 친화적인 텍스트로 생성하고자 하는 정책에 대한 의도를 작성할 수 있게 하여, 정책 관리의 복잡성을 크게 줄인다. 그리고 하나의 파일에 네트워크 및 시스템 관련 보안 정책을 동시에 선언하고, 한 번에 적용할 수 있다는 뛰어난 편의성을 제공한다.

3.2 IntentShield Design

Fig 3처럼 설계된 IntentShield는 사용자가 네트워크 및 시스템 정책에 대해 사용자 친화적인 텍스트로 원하는 정책을 작성하면, 정책에 대한 전반적인 관리를 수행한다.

General Controller의 Status Sync Reconciler는 CR의 현재 상태를 모니터링하며 변경 사항을 감지하고, 상태를 동기화하는 역할을 한다. Spec Sync Reconciler는 CR의 템플릿을 관리하며, 변경 사항을 감지하고 동기화한다.

Policy Controller는 System Policy Reconciler, Network Policy Reconciler, 그리고 Resource Reconciler로 구성된다. 각각의 Reconciler는 시스템 및 네트워크 정책, 그리고 파드, 서비스와 같은 리소스를 관리하며 생성, 업데이트, 삭제, 적용을 수행한다.

Validation Controller에는 'Pre-Validation Reconciler과 Post-Validation Reconciler로 구성되어, 요청된 정책이 클러스터 상태에 적합한지를 검증하거나 실제 시스템 상태가 정책을 정확히 반영하고 있는지 검증한다.

오퍼레이터는 해당 의도를 분석하여 적절한 보안 정책을 자동으로 생성하고 적용하기 때문에, 사용자는 복잡한 보안 정책의 세부 사항을

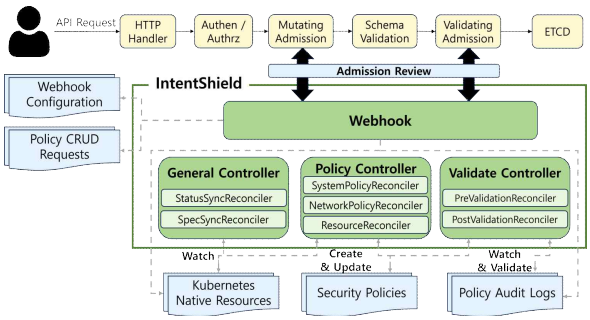

Fig 3. The Architecture of IntentShield

직접 관리할 필요 없이 원하는 보안 상태를 쉽게 유지할 수 있으므로 정책 관리의 효율성을 향상할 수 있다.

IV. Evaluation

4.1 Test Enviroments

KubeArmor[2], Kubernetes 및 Cilum[3] 정책을 지원하는 IntentShield 프로토타입은 Go 언어와 KubeBuilder로 구현하였다. 실험 환경은 Intel(R) Xeon(R) Silver 4210R CPU 2.40 GHz, 256GB RAM, 2TB HDD 서버에서 Ubuntu 22.04 머신 3대로 클러스터를 설정하였다.

프로토타입은 KubeArmor에서 제공하는 오픈된 정책 템플릿[4]을 기반하여, 사용자의 의도와 일치하는 정책을 자동으로 탐색하여 적용하도록 동작하기 때문에 해당 룰셋을 사전에 다운로드한 상태에서 진행되었다.

4.2 Effectiveness

Fig 4와 같이, Redis 포트로의 외부 접근 차단 및 shadow 파일에 접근을 시도할 경우 이를 차단하는 시나리오를 기반으로 IntentShield의 정책 적용 여부를 확인하는 실험을 진행하였다.

사용자는 SecurityIntent을 이용하여 'Redis 포트로의 외부 트래픽 허용하지 않기'와 'shadow 파일에 접근하려는 시도 차단 및 경고하기'를 enforce 필드에 작성(①)하고 'kubectl apply' 명령으로 API 요청을 전송한다. Web Hook을 통해 이를 전달 받은 IntentShield는 로그ⓐ(Starting the search for policy files)와 같이 enforce 필드에 작성된 문자열과 동일한 필드 값을 가진 정책을 찾기 시작한다(②). 로그ⓑ(Inspecting file)과 같이 사전에 정의된 정책

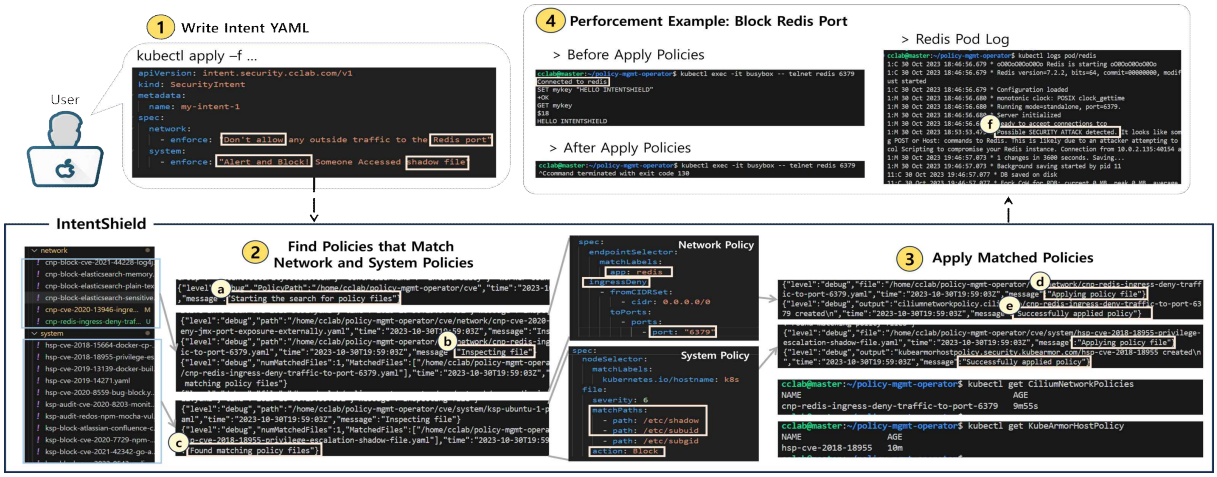

Fig 4. The Workflow of IntentShield

의 필드를 검사하면서 일치하는 값을 가졌는지 검사한다. 만약 동일한 값을 가진 정책을 찾았다면 로그ⓒ(Found matching policy files)와 같이 알림을 전송한다. 찾은 파일을 확인해 보면 사용자의 의도와 어울리는 적절한 정책을 찾은 것으로 확인된다. 이렇게 찾은 정책은 로그ⓓ, ⓔ와 같이 IntentShield 내에서 자동으로 정책을 적용하고 성공 여부를 알려준다(③). 결과적으로 네트워크 정책은 Cilium Network Policy로, 시스템 정책은 KubeArmor Host Policy로 성공적으로 적용된 것을 확인할 수 있다.

정책이 실제로 유효하게 적용되었는지 검증하기 위해, 임의의 Busybox 및 Redis 파드로 네트워크 정책이 적용되기 전과 후를 비교하였다(④). 정책이 적용되기 전에는 Busybox 파드에서 'telnet' 명령을 사용하여 Redis 파드에 접속할 수 있었다. 정책이 적용된 후에는 Redis 6379 포트로의 접근 시도가 모두 차단되었으며, Redis 파드의 로그ⓕ처럼 보안 공격 시도를 감지한 메시지 또한 확인할 수 있었다.

이를 통해 본 시스템이 실제 쿠버네티스 환경에서 효과적으로 보안 정책을 시행할 수 있음을 입증하며, 사용자가 보안 정책을 보다 쉽고 편리하게 적용할 수 있음을 보여준다.

V. Conclusion and Future Work

본 연구에서 제안한 IntentShield는 보안 정책 설정 및 적용을 단순화하는 동시에, 더욱 견고한 보안 환경을 제공하고자 한다.

향후 연구는 다양한 공격 시나리오에 대응하는 정책 변환 로직의 확장과 정책 업데이트 메커니즘의 개발에 초점을 맞출 예정이다. 또한, 기존에 미리 정의된 정책 템플릿을 사용하지 않고 의도 기반으로 정책을 동적으로 생성하기 위해 자연어처리 기술을 활용하여 정책 분석을 구현하고자 한다. 정책 검증에서는 Common Expression Language (CEL)의 활용하여 정책의 유효성 검사 및 보안 규칙의 일관성을 보장하는 방향으로 확장하고자 한다.

[참고문헌]

[1] Almehmadi, Abdulaziz, and Khalil El-Khatib. "On the possibility of insider threat prevention using intent-based access control (IBAC)." IEEE Systems Journal 11.2 (2015): 373-384.

[2] KubeArmor, Oct 2023, [online] Available: https://github.com/kubearmor/KubeArmor

[3] Cilium - Network Policy, Oct 2023, [online] Availble: https://docs.cilium.io/en/latest/security/policy/

[4] KubeArmor - policy-templates, Oct 2023, [online] Availble: https://github.com/kubearmor/policy-templates.