Kuberosy+: 컨테이너를 위한 상태 저장 기반 시스템 콜 보안 프레임워크

허진*, 조치현*, 이승수**

*인천대학교 (대학원생)

**인천대학교 (교수)

Kuberosy+: Stateful System Call Security Framework for Containers

Jin Her*, Chihyeon Cho*, Seungsoo Lee**

*Incheon National University(Graduate student)

**Incheon National University(Professor)

요 약

최근 컨테이너 환경의 보안 위협이 증가함에 따라 시스템 콜 통제 기술이 주목받고 있다. 그러나 기존 기법은 단일 호출만 검증하는 상태 비 저장 구조로 시퀀스 기반 공격에 취약하며, 비동기적 정책 배포로 인해 초기 구동 시 보안 공백기를 유발한다. 또한 고부하 환경에서 빈번한 Context Switching으로 로그의 손실이 발생한다. 본 논문은 이를 극복하기 위해 OCI(Open Container Initiative) Prestart Hook과 eBPF(Extended Berkeley Packet Filter) 기반의 시스템 콜 보안 프레임워크를 제안한다. 스레드 단위의 시스템 콜 전이 상태를 추적하여 비정상 흐름을 탐지하고, 커널 내 검증을 통해 정책 집행으로 인한 성능 저하를 최소화한다. 또한 컨테이너 구동 전 정책을 선제적으로 주입하여 사각지대를 제거하며, 허프만 압축으로 로그 압축률을 극대화한다. 평가 결과, 제안 시스템은 시퀀스 기반 공격 방어에 성공하였고, 정책 적용 지연 시간 제거 및 70.23%의 로그 압축률을 달성하였다.

I. Introduction

클라우드 네이티브 환경의 핵심인 컨테이너는 호스트 커널을 공유하는 특성상 시스템 콜 악용 공격에 취약하여 시스템 콜 통제 기술이 필수적이다. 그러나 기존 통제 기술들은 세 가지 구조적 한계를 지닌다. 첫째, 상태 비 저장 검증은 허용된 시스템 콜을 악의적으로 조합하는 논리 기반 공격에 취약하다. 둘째, Operator 기반 비동기 정책 주입 방식은 컨테이너 배포 초기에 보안 공백을 유발하여 해당 시점에 발생하는 악성 행위를 통제하지 못한다. 셋째, 고부하 환경에서는 빈번한 eBPF Ring Buffer 호출로 인해 로그 데이터의 손실이 발생한다.

본 논문은 이러한 한계를 극복하기 위해 상태 저장 기반의 시스템 콜 보안 프레임워크를 제안한다. 본 프레임워크는 정책 집행 오버헤드를 최소화하기 위해 eBPF를 활용하여 커널 내에서 정책의 검증을 수행하며, LSM-BPF와 SIGKILL을 결합한 하이브리드 제어 기법을 통해 비정상적인 시스템 콜 전이를 효과적으로 통제한다.

본 논문의 핵심 기여는 다음과 같다.

- **상태 저장 기반 시퀀스 통제**: 컨테이너에 대하여 스레드 단위의 시스템 콜 전이를 실시간 추적하여 논리 기반 공격을 차단한다.
- **무지연 정책 적용**: OCI Prestart Hook을 통해 컨테이너 구동 초기부터 정책을 선제적으로 적용하여 정책 집행 사각지대를 제거한다.
- **로그 압축률 극대화**: In-kernel에서 반복적인 인자 패턴을 허프만 트리로 압축하여 압축률을 극대화하고 데이터 손실을 최소화한다.

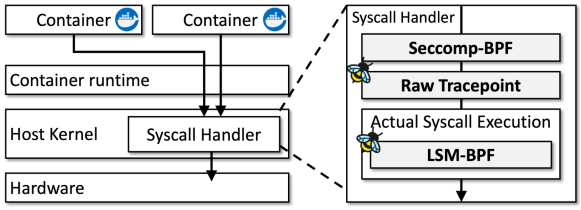

Fig 1. 컨테이너의 시스템 콜 사용 흐름

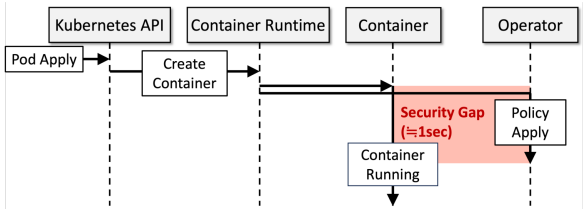

Fig 2. 정책 적용 지연으로 인한 보안 공백

II. Background

컨테이너는 호스트 운영체제의 커널을 공유하므로, 내부에서 발생하는 모든 시스템 콜은 호스트 커널의 시스템 콜 핸들러를 거쳐 하드웨어 자원에 접근하게 된다. 따라서 이 경로상에서의 적절한 통제가 컨테이너 보안의 핵심이며, Fig 1은 이러한 시스템 콜 처리 파이프라인 내에 존재하는 대표적인 세 가지 보안 제어 지점을 보여준다.

2.1 Seccomp-BPF 기반 정적 필터링

파이프라인의 가장 앞단에서 동작하는 표준 보안 기법인 Seccomp-BPF는 사전에 정의된 정적 프로파일을 컨테이너 생성 시점에 커널에 등록하는 방식으로 정책을 적용한다. 이에 따라 커널의 시스템 콜 핸들러는 프로세스가 시스템 콜을 호출할 때 즉각적으로 프로파일을 참조하여, 허용 목록에 명시되지 않은 실행을 원천적으로 차단한다.

2.2 eBPF 기반 동적 제어 매커니즘

그러나 Seccomp-BPF는 런타임 도중 정책을 변경할 수 없는 정적인 한계가 있다. 이를 보완하기 위해, 커널 수정 없이 동적으로 사용자 정의 프로그램을 실행할 수 있는 eBPF 기술이 적극 활용되고 있다. Seccomp-BPF를 무사히 통과한 시스템 콜은 파이프라인 상에 이어지는 두 곳의 eBPF 기반 제어 지점을 순차적으로 거치게 된다.

먼저, 첫 번째 eBPF 제어 지점은 시스템 콜 핸들러 내부에 위치한 Raw Tracepoint이다. 이곳에 eBPF 프로그램을 부착하면 시스템 콜 호출 이벤트를 실시간으로 모니터링할 수 있으며, 정책 위반 시 SIGKILL을 전송하여 강제 종료하는 사후 대응의 형태로 시스템 콜을 차단할 수 있다.

이후 실제 시스템 콜 실행 단계에 도달하면, 특정 시스템 콜의 경우 두 번째 제어 지점인 LSM 훅을 거치게 된다. LSM Hook에서는 시스템 콜이 실행되기 전 반환값을 수정하여 접근 제어를 수행할 수 있다.

III. Problem Statements

C1: Limitation of Stateless Syscall Filter

기존 Seccomp-BPF 기반 필터링은 단일 시스템 콜의 허용 여부만 검증할 뿐, 호출 간의 순서나 실행 컨텍스트를 추적하지 못한다. 이로 인해 정상 권한을 부여받은 시스템 콜들을 임의로 조합하여 본래 의도와 다른 제어 흐름을 생성하는 시퀀스 기반 공격에 무방비하게 노출된다.

C2: Lack of Lifecycle-Aware Enforcement

기존 보안 도구들은 주로 Kubernetes Operator를 매개로 Pod 배포 이벤트를 감지하여 런타임에 정책을 주입한다. 해당 구조는 Pod 실행과 정책 배포가 비동기적으로 병렬 처리되므로 필연적인 정책 적용 지연 시간을 유발하며, 커널 적용 전 초기 단계에 치명적인 보안 공백을 형성한다.

C3: Log Loss in Large Load Environments

사후 분석을 위한 시스템 콜 로깅 시, 시스템 콜 이벤트마다 Ring Buffer를 호출하는 기존 eBPF 수집 환경은 잦은 Context Switching으로 막대한 CPU 오버헤드를 발생시킨다. 특히 로그 생성 속도가 소비 속도를 추월하는 경우 Ring Buffer 오버플로우를 유발해 핵심 보안 이벤트 누락 및 감사 무결성이 훼손될 수 있다.

IV. Design

4.1 Assumptions and Threat Model

최근 시스템 콜 전이 그래프 추출 연구는 다수 진행되었으나, 이를 실제 컨테이너 런타임 환경에서 안전하게 집행하는 연구는 미비하다. 따라서 본 연구는 사전 추출된 전이 그래프를 커널 레벨에서 효율적으로 강제하는 아키텍처 설계에 집중한다. 위협 모델은 공격자가 컨테이너 내부 프로세스를 장악해 권한 상승 및 측면 이동을 시도하는 상황을 가정하며, 호스트 커널과 Kubernetes 제어 평면은 신뢰 구간으로 간주한다.

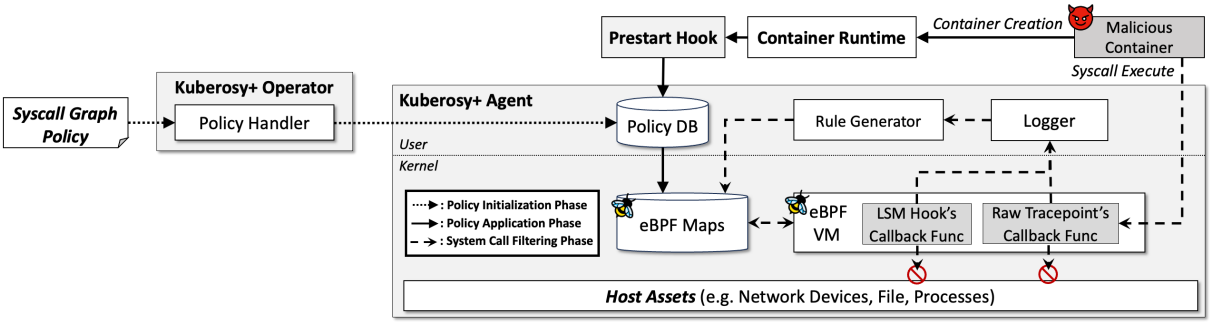

Fig 3. Kuberosy+의 전체 아키텍처

4.2 Overview Architecture

제안하는 KubeRosy+ 프레임워크는 사전 추출된 전이 그래프를 정책으로 받아 다음 세 가지 핵심 워크플로우로 동작한다. 첫째, 정책 초기화 단계에서는 클러스터에 배포된 정책을 마스터 노드에 배포된 Operator의 Policy Handler가 수신 및 분석하여 대상 워커 노드의 Agent로 전달하며, 이는 Policy DB에 저장되어 대기한다. 둘째, 정책 적용 단계는 컨테이너 생성 요청 발생 시 Prestart Hook이 컨테이너 구동을 중단시키고, Policy DB의 보안 정책을 커널 영역의 eBPF Maps에 적재한 후 구동을 재개함으로써 정책 집행 사각지대를 제거한다. 셋째, 시스템 콜 필터링 단계에서는 시스템 콜 발생 시 커널 내 LSM Hook 및 Raw Tracepoint의 콜백 함수가 eBPF Maps의 전이 규칙을 참조해 위반 시스템 콜을 즉각 차단한다. 동시에 수집된 이벤트는 Logger로 전달되며, 분석을 통해 갱신된 압축 규칙을 기반으로 커널 내부에서 로그 데이터를 지속적으로 압축한다.

V. Detail

5.1 Stateful Syscall Filter

Stateless 필터의 한계를 극복하고자 상태 저장 기법을 통해 스레드 단위의 시스템 콜 전이를 추적한다. 악성 프로세스가 시스템 콜(execve)을 호출하면, raw_tracepoint 콜백 함수는 eBPF Map (Fig 4)을 연쇄적으로 조회한다. 먼저 Previous Syscall map에서 직전에 호출한 시스템 콜(sched_yield)을 확인하고, 이를 키로 Syscall Graph map에서 도출한 허용 목록(sys_getitimer)과 현재 호출을 대조하여 실행을 차단한다.

정책 위반 시에는 사후 대응인 SIGKILL의 보안적 한계를 보완하기 위해, 기존 연구[1]에서 정의한 17개의 고위험 시스템 콜은 LSM-BPF를 적용해 커널 로직 실행 전에 사전 차단한다. 그 외 비정상 전이 흐름은 식별 즉시 프로세스에 SIGKILL을 전송하여 시스템 콜을 차단한다.

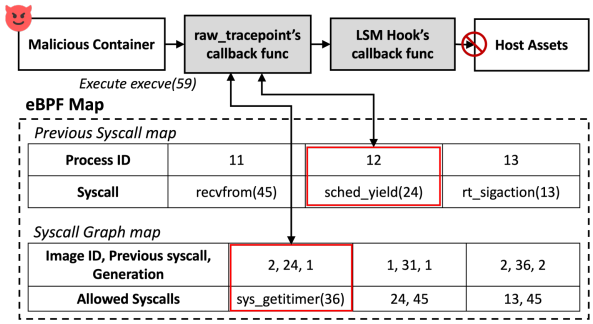

Fig 4. Stateful 시스템 콜 필터 동작 흐름

5.2 Preemptive Policy Application

Operator의 비동기적 감지로 인한 정책 지연을 해결하고자, 본 시스템에서는 OCI Prestart Hook을 활용하여 정책 집행 시점을 컨테이너 실제 구동 이전으로 앞당긴다. OCI Hook은 컨테이너의 Namespace와 파일 시스템 구성 완료 직후, 컨테이너의 Entrypoint가 실행되기 직전의 시점에 개입하여 선제적으로 정책을 eBPF Map에 저장하며, 정책 주입 완료 후에만 컨테이너의 실행 흐름을 재개하도록 하여 보안 사각지대를 차단한다.

5.3 Adaptive Log Compression

대규모 트래픽 환경에서 발생하는 Ring Buffer의 context switching을 최소화하기 위해, 기존 eAudit[2]의 커널 로깅 최적화를 유저 공간의 분석과 결합하여 발전시킨 적응형 허프만 압축 기법을 도입했다(Fig 5). Logger는 로그 스트림에서 시스템 콜 인자 조합의 발생 빈도를 집계한다. 이를 바탕으로 Rule Generator는 고반복 인자 패턴에 짧은 식별 코드를 매핑하는 압축 규칙을 생성해 eBPF Map을 갱신한다. 이후 시스템 콜이 실

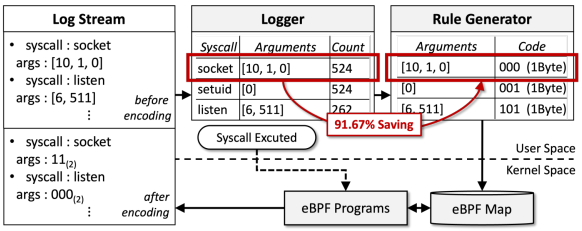

Fig 5. 적응형 로그 압축 적용 예시

행되면 커널의 eBPF 프로그램은 이를 참조해 로그를 인코딩하며(예: 12바이트 socket 인자를 1바이트 코드로 치환해 91.67% 절감), 고도로 압축된 정보만 전송해 로그 압축률을 극대화한다.

VI. Evaluation

6.1 Effectiveness of Stateful Enforcement

단일 호출의 유효성만 검사하는 기존 정적 허용 목록 방식의 한계를 검증하고자, 대표적인 자바 로깅 취약점인 Log4j를 활용한 시나리오 기반 공격을 수행하였다. 공격자가 정상적으로 허용된 파일 읽기 요청 직후 외부 제어 서버로의 네트워크 연결을 시도할 때, 기존 방식은 두 시스템 콜을 각각 합법적인 것으로 오인하여 모두 통과시켰다. 그러나 제안 시스템은 해당 시스템 콜 간의 전이 흐름이 정상 범위를 벗어났음을 즉각 감지하였으며, 탐지 직후 SIGKILL을 전송함으로써, 시퀀스 기반의 논리 공격이 성공하기 전에 제어 흐름을 정확히 차단할 수 있음을 확인하였다.

6.2 Policy Enforcement Latency

OCI Prestart 훅을 활용한 동적 정책 주입의 실효성을 검증하기 위해 기존 eBPF 기반 모니터링 도구인 Tetragon과 정책 적용 지연 시간을 비교하였다. Fig 6과 같이 제안 시스템은 정책 주입에 약 1.5초가 소요되었으나 이는 OCI Hook 특성상 컨테이너 실행 전 준비 단계에 해당하여 보안 공백이 발생하지 않았다. 반면, Tetragon은 컨테이너 구동 후에도 정책 적용까지 약 1.36초(총 2.8초대)의 추가 지연이 발생하였다. 결과적으로 제안 시스템은 초기의 정책 지연을 제거함으로써, 보안 공백이 발생하지 않지만, 기존의 방식은 보안 공백이 발생함을 입증하였다.

6.3 Effectiveness of Log Compression

로그 압축의 효과성을 측정하기 위해 Google Online-boutique의 이벤트 약 6,000건을 분석하여

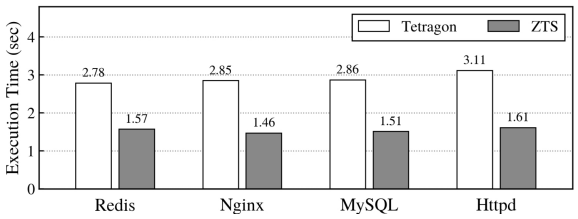

Fig 6. 정책 적용 지연 시간 비교

Approach	Total Data Size	Compression Ratio
No Compression	242,187B	-
eAudit	153,660B	36.55%
KubeRosy+	72,089B	70.23%

Table 1. Online-boutique 대상 로그 압축률

로그 압축률을 측정하였다. 실험 결과, 기존의 방식은 원본 데이터 대비 36.55%의 압축률을 기록했지만, 제안 기법은 70.23%의 압축률을 달성하였다(Table 1). 이는 적응형 로그 압축 기법이 데이터의 중복을 효과적으로 제거하여 기존 압축 방식 대비 압축률을 증폭시킬 수 있음을 보여준다.

VII. Conclusion

본 논문에서는 컨테이너 환경의 시스템 콜 집행 사각지대와 상태 비 저장 필터의 취약점을 분석하고, 이를 해결할 수 있는 eBPF 기반의 보안 프레임워크를 제안하였다. 또한, 실험을 통해 제안 기술들의 실현 가능성과 방어 효과성을 입증하였다.

Acknowledgements

이 논문은 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원-학·석사연계ICT핵심인재양성 지원을 받아 수행된 연구임(IITP-2026-RS-2024-00437024)

[참고문헌]

[1] Ghavamnia, S., Palit, T., Mishra, S., & Polychronakis, M. (2020). Temporal system call specialization for attack surface reduction. In 29th USENIX Security Symposium (USENIX Security 20) (pp. 1749-1766).

[2] Sekar, R., Kimm, H., & Aich, R. (2024, May). eaudit: A fast, scalable and deployable audit data collection system. In 2024 IEEE Symposium on Security and Privacy (SP) (pp. 3571-3589). IEEE.