클라우드 네이티브 환경의 eBPF 기반 시스템 보안 도구에 대한 심층 분석

허진[1], 김종섭[2], 김진우[3], 이승수[4]
[1]인천대학교 컴퓨터공학부 석사과정
[2]광운대학교 컴퓨터공학과 석사과정
[3]광운대학교 컴퓨터공학과 교수
[4]인천대학교 컴퓨터공학부 교수
gjwls0787@inu.ac.kr, joseab5965@kw.ac.kr, jinwookim@kw.ac.kr, seungsoo@inu.ac.kr

Deep dive into eBPF-based system security tools in cloud-native environments

Jin Her[1], Jongseop Kim[2], Jin-Woo Kim[3], Seung-Soo Lee[4]
[1]Dept. of Computer Science, Incheon University
[2]Dept. of Computer Engineering, Kwang-Woon University
[3]Dept. of Computer Science, Kwang-Woon University
[4]Dept. of Computer Science, Incheon University

요 약

쿠버네티스 기반 마이크로서비스 아키텍처는 유연성, 확장성, 다양한 사용 사례를 제공하는 최신 클라우드 네이티브 환경의 핵심이다. 클라우드 환경에서 빈번한 대규모의 복잡한 배포를 고려할 때 잠재적인 위협을 탐지하려면 감사가 매우 중요하다. 이러한 요구를 해결하기 위해 eBPF 기반의 다양한 보안 도구(ESST)가 개발되었으며, eBPF를 활용하여 쿠버네티스 환경의 프로세스 및 네트워크 활동을 모니터링하고 의심스러운 이벤트를 기록하며 보안 정책을 시행할 수 있다. 그러나 관리자는 종합적인 비교 분석이 부족하여 가장 적합한 ESST를 선택하는 데 어려움을 겪는 경우가 많다. 따라서 이 논문에서는 내부 아키텍처, 정책 적용 범위, 전반적인 성능에 초점을 맞춰 널리 사용되는 ESST인 KubeArmor, Tetragon, Falco, Tracee에 대한 심층적인 평가를 진행한다.

1. 서론

마이크로 서비스 아키텍처는 다양한 요구사항을 충족하는 유연한 확장성으로 인해 엔터프라이즈 네트워크에서 널리 채택되고 있다. 그러나 이러한 복잡한 환경을 관리하려면 상당한 관리 노력이 필요하다. 쿠버네티스는 여러 노드에서 컨테이너를 오케스트레이션하고 구성 및 운영 관리를 간소화하여 이 문제를 해결했다. 그 결과, 쿠버네티스 기반 마이크로 서비스 아키텍처는 클라우드 환경의 사실상 표준이 되었다. 보고서에 따르면, 60% 이상의 기업이 쿠버네티스를 채택하고 있는 것을 확인할 수 있다[1].

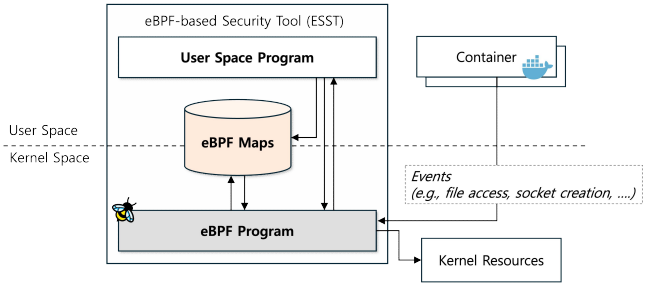
보안 관점에서 볼 때, 잘못 구성된 쿠버네티스 기반 마이크로 서비스 아키텍처는 심각한 취약성을 초래할 수 있다. 이러한 취약성을 완화하기 위해 미국 국가안보국(NSA)과 사이버보안 및 인프라 보안국(CISA)은 eBPF 기반 시스템 보안 도구(이하 ESST)의 사용을 권장한다[2]. 이러한 도구는 eBPF(extended Berkeley Packet Filter)를 활용하여 쿠버네티스 감사 로그를 지속적으로 모니터링하여 보안 위협을 탐지하고 전반적인 보안을 강화한다. ESST의 대표적인 예시로는 KubeArmor[3], Falco[4], Tetragon[5], Tracee[6] 등이 있다.

그러나 클라우드 관리자의 관점에서 이러한 도구를 사용하는 데는 몇 가지 어려움이 있다. 첫째, 관리자는 종종 내부 아키텍처에 대한 가시성이 제한된 블랙박스로 ESST를 사용하며, 동작 구조에 대해 이해하려면 상당한 노력이 필요하다. 둘째, ESST는 복잡한 정책 구조와 다양한 정책 적용 범위를 갖추고 있어 철저한 전문 지식 없이는 효과적인 활용이 어렵다. 셋째, 관리자가 ESST의 성능에 대한 인사이트가 부족한 경우가 많기 때문에, 이상 징후나 병목 현상이 발생했을 때 이를 진단하기 어렵다.

따라서 이 논문에서는 다음과 같은 관점에서 ESST를 분석한다.

- ESST의 내부 아키텍처는 어떻게 구성되는가?
- ESST별 정책의 범위가 어떻게 되는가?
- ESST의 성능 차이는 어느 정도이며, 해당 차이는 어디에서 기인하는가?

<그림 1> ESST 공통 아키텍처



2. 아키텍처 분석

이 섹션에서는 널리 사용되는 ESST인 Kube Armor, Tetragon, Tracee, Falco의 아키텍처를 분석하여 설계와 정책의 배포와 적용에 걸쳐 전반적인 동작 흐름을 이해한다. 분석은 각 도구의 오픈 소스를 수동으로 검토하여 진행하였다. 그림 1은 ESST의 공통 아키텍처를 보여준다. ESST는 세 가지 주요 구성 요소로 구성되어 있다: (i) 사용자 공간 프로그램, (ii) eBPF 프로그램, (iii) eBPF 맵. 사용자 공간에서 실행되는 사용자 공간 프로그램은 이벤트 모니터링, 정책 적용, 활성 컨테이너에 대한 로깅과 같은 높은 수준의 감사 작업을 담당한다. eBPF 프로그램은 각 훅 포인트에서 실행되어 커널 수준에서 감사를 수행한다. 이러한 프로그램은 사용자 공간과 커널 공간 간의 통신을 용이하게 하는 공유 데이터 구조인 eBPF 맵을 통해 상호작용을 한다.

2.1 KubeArmor의 동작 흐름

KubeArmor 정책이 쿠버네티스 API에 배포되면, API를 감시하고 있는 사용자 공간 프로그램의 인포머 모듈은 이벤트 핸들러에 따라 수신된 정책 정보를 파싱한 후 eBPF 맵에 새 정책 데이터를 업데이트하는 방식으로 정책을 업데이트한다. 이후, 컨테이너가 정책과 일치하는 작업을 수행하면 커널의 훅 포인트에 연결된 eBPF 프로그램이 트리거되며, 이러한 프로그램은 정책에 따라 해당 작업을 허용, 차단 또는 감사한다. 컨테이너의 동작이 감사 정책의 기준을 충족하는 경우, eBPF 프로그램은 링 버퍼를 사용하여 커널 이벤트 데이터를 사용자 공간 프로그램의 로거 모듈로 전송한다. 그리고 로거가 이벤트 데이터를 처리하여 관리자에게 전달한다.

2.2 Tetragon의 동작 흐름

Tetragon의 정책인 TracingPolicy가 배포되면, Tetragon 사용자 공간 프로그램의 인포머는 Sensor Manager가 설정한 채널을 통해 정책 정보를 전송한다. Tetragon의 SensorManager는 이 정보를 수신하면 전용 정책 핸들러를 사용하여 정책을 kprobe, tracepoint 등으로 분류한다. 이후 빌더를 통해 eBPF 프로그램을 구성하고 센서로 배포한다. 마지막으로, Tetragon 사용자 공간 프로그램의 Observer 모듈은 PerfReader를 생성하여 eBPF 프로그램에서 생성된 출력을 캡처하고 기록한다.

2.3 Tracee의 동작 흐름

전용 리스너를 통해 정책의 배포를 능동적으로 탐지하고 즉시 적용하는 KubeArmor 및 Tetragon과 달리, Tracee는 데몬셋의 초기화 단계에서만 정책을 검색하고 적용한다. 결과적으로, Tracee 정책이 쿠버네티스에 배포되면, Reconciler는 타임스탬프 주석을 추가하여 Tracee 데몬셋의 템플릿을 수정하고, 데몬셋 컨트롤러가 새 데몬셋을 롤아웃하도록 유도하여 파드를 재시작한다. 재시작된 파드는 초기화 단계로 다시 진입하며, 배포된 정책에서 정책 정보를 추출하고 eBPF 맵을 업데이트한 뒤 감사 정책을 적용한다. 이후 작업이 감사 정책의 기준을 충족하는 경우, Tracee의 eBPF 프로그램은 해당 커널 이벤트를 수집하고 PerfBuffer를 통해 사용자 공간 프로세스로 전송한다.

2.4 Falco의 동작 흐름

Falco는 쿠버네티스의 전용 CRD를 제공하지 않으며, Falco의 정책은 helm upgrade 명령을 통해 적용된다. helm upgrade 명령어가 실행되면 helm은 Falco custom policy를 업데이트하며, 이 정책은 이후 호스트 파일 시스템에 Falco의 구성 파일로 저장된다. Falco 데몬셋은 해당 구성 파일을 마운트하고, 재시작된 파드는 초기화 단계에서 구성 파일을 기반으로 업데이트된 규칙 파일을 로드하고, eBPF 맵을 초기화한다. 감사 정책이 배포되면 정책과 일치하는 이벤트가 발생하는 경우 이벤트 정보를 수집하며, 정책과 일치하지 않는 이벤트의 경우 eBPF 프로그램 실행 도중에 반환된다. Tracee와 Falco의 이러한 정책 적용 매커니즘은 파드의 재배포를 기반으로 하기 때문에, 잠재적인 다운 타임과 정책 적용 오버헤드 등을 초래할 수 있다.

3. 정책 적용 범위 분석

다음으로 ESST의 정책을 분석함으로써 각 도구

<표 1> ESST 별 정책 적용 범위

ESST	Policy Scope
Tracee	Security, Network, Extra, Syscalls
KubeArmor	Process, File, Network, Capabilities, Syscalls
Falco	Syscalls, Tracepoints, Meta, Plugins
Tetragon	Kprobes, Tracepoints, Uprobes, LSM BPF

가 지원하는 정책의 적용 범위를 살펴보며, 요약된 결과는 표 1과 같다.

3.1 Falco

Falco는 시스템 콜 이벤트, tracepoint 이벤트, 메타 이벤트, 플러그인 이벤트에 대한 정책을 지원한다. 시스템 콜 이벤트는 시스템 콜의 실행을 감지하는 것으로, 일부 시스템 콜에 대해서는 인자 추적을 지원하지만, 모든 시스템 콜에 대해 인자 기반 추적이 가능한 것은 아니다. tracepoint 이벤트는 커널 내에서 발생하는 중요한 이벤트이지만 시스템 콜로 대체되지 않는 활동을 포착하며, 커널의 내부 동작을 보다 정밀하게 관찰하는 데 활용된다. 메타 이벤트는 데이터 보강 또는 비동기 처리 과정 중에 생성되는 이벤트로, Falco의 탐지 로직을 보조하는 고수준 이벤트를 구성한다. 플러그인 이벤트는 사용자 정의 플러그인을 통해 수집된 외부 데이터 기반 이벤트로, Falco의 탐지 범위를 확장하는 역할을 한다.

3.2 Tracee

Tracee는 Security 이벤트, 네트워크 이벤트, Extra 이벤트, 시스템 콜 이벤트에 대한 정책을 지원한다. Security 이벤트는 시스템 콜, 네트워크 상호작용, LSM 훅 트리거와 같은 저수준 이벤트를 기반으로 구성된 시그니처를 통해 특정 보안 활동을 탐지한다. 네트워크 이벤트는 다양한 네트워크 프로토콜 상의 활동을 추적하며, 시스템의 외부 통신을 감시할 수 있도록 한다. Extra 이벤트는 일반적인 시스템 콜 이벤트로는 탐지가 어려운 활동을 포착하는 데 사용된다. 시스템 콜 이벤트에 대해서는 전체 시스템 콜에 대해 인자까지 포함한 세부적인 추적을 지원하여 악성 코드 실행을 탐지하기에 효과적인 정책 구성이 가능하다.

3.3 KubeArmor

KubeArmor는 프로세스 이벤트, 파일 이벤트, 네트워크 이벤트, 권한 이벤트, 시스템 콜 이벤트를 대상으로 정책을 적용한다. 프로세스 이벤트는 실행 중인 바이너리의 동작을 추적하고, 파일 이벤트는 파일의 실행뿐 아니라 읽기 및 쓰기와 같은 파일 접근 전반에 대해 감지한다. 네트워크 이벤트는 네트워크 프로토콜에 따른 트래픽을 감시하며, 권한 이벤트는 실행 파일이 요청하는 Linux 권한을 기반으로 정책을 적용한다. 시스템 콜 이벤트는 시스템 콜의 호출 여부를 기준으로 정책을 적용하지만, 시스템 콜 인자에 대한 추적은 지원하지 않는다.

3.4 Tetragon

Tetragon은 kprobes, tracepoints, uprobes, LSM BPF를 기반으로 정책을 구성한다. kprobes는 커널 함수에 동적으로 후크를 삽입할 수 있어, 런타임 중 커널 내부 동작을 실시간으로 관찰하는 것이 가능하다. tracepoints는 커널에 정적으로 정의된 이벤트 지점으로, 커널 버전 간 안정적인 호환성을 제공한다. uprobes는 사용자 공간 애플리케이션의 함수 호출을 대상으로 하며, 사용자 수준 이벤트의 추적을 가능하게 한다. LSM BPF는 LSM hook point에 eBPF를 부착하여, LSM 기반 보안 이벤트에 대해 정밀한 정책 구성을 지원한다. 또한 Tetragon은 모든 hook point에서 인자의 수집 및 인자 기반의 정책 적용을 지원함으로써, 보다 세분화된 정책 설정과 정밀한 모니터링을 가능하게 한다.

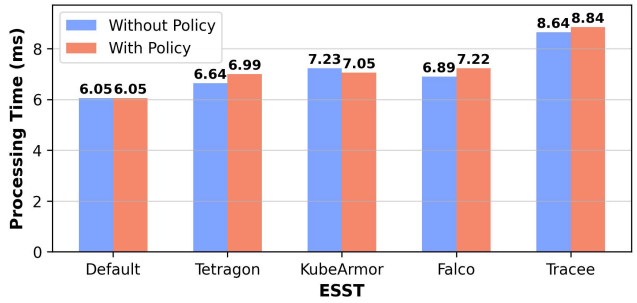
4. 성능 분석

4.1 마이크로 벤치마크

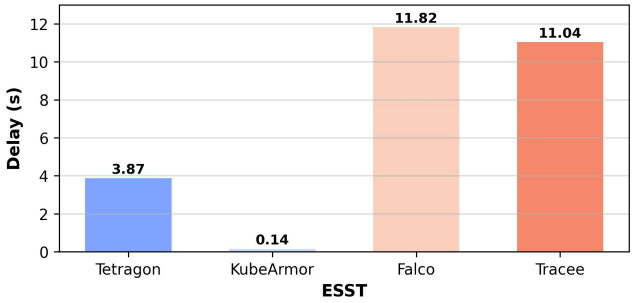
마이크로 벤치마크는 /usr/bin/ls 바이너리 파일을 10,000번 실행하며 성능을 측정하는 실험으로, 다음 세 가지 조건에서 진행되었다: (1) 기본 상태(ESST가 적용되지 않은 상태), (2) ESST가 적용되었으나 정책이 없는 상태, (3) ESST와 정책이 모두 적용된 상태. ESST별로 정책이 적용되지 않은 상태에서도 성능 차이를 측정한 이유는 대부분의 ESST가 기본 정책(default policy)을 포함하고 있으며, Tetragon을 제외한 다른 ESST들은 정책이 없더라도 특정 hook point에서 eBPF 프로그램이 실행되기 때문이다.

그림 2는 마이크로 벤치마크의 결과를 시각화한 것으로, Tetragon은 정책이 적용되지 않은 경우 기

<그림 2> 마이크로 벤치마크



<그림 3> 정책 시행 벤치마크



본 상태 대비 9.75%의 오버헤드를 보였으며, 정책이 적용된 경우 15.65%로 증가하였다. 이는 모든 ESST 중 가장 낮은 성능 오버헤드를 기록한 결과이다. 반면, KubeArmor는 정책이 적용되지 않은 경우 19.47%의 오버헤드를 보였고, 정책이 적용된 경우 16.62%로 오히려 오버헤드가 감소하는 경향을 보였다. 이는 KubeArmor 프로세스 정책의 경우 정책 적용 여부를 확인한 후 반환하는 경우가 정책이 적용되지 않은 경우보다 적은 오버헤드를 발생시키기 때문이다. Falco는 정책이 적용되지 않은 경우 13.94%, 정책이 적용된 경우 19.44%의 성능 오버헤드를 나타내었다. 반면, Tracee는 정책이 적용되지 않은 경우 42.93%, 정책이 적용된 경우 46.25%의 성능 오버헤드를 기록하여 모든 ESST 중 가장 높은 성능 저하를 보였다. 이는 Tracee가 여러 eBPF 프로그램들을 tail call로 연결하여 동작하는데 이때 특정 동작(ex. 정책의 적용 대상인지 확인)이 여러 tail call 에서 중복되어 실행되었기 때문으로 해석된다.

4.2 정책 시행 벤치마크

정책 Enforcement 벤치마크는 감사 정책을 배포한 후, /usr/bin/ls 바이너리 파일을 반복적으로 실행하며 정책이 적용되어 실행 로그가 발생할 때까지의 시간을 측정하는 방식으로 진행되었다. 그림 3은 실험 결과를 시각화한 그래프이다. Tetragon은 정책을 적용하는 과정에서 파드를 재배포하지 않지만, eBPF 프로그램을 구성한 후 attach 하는 과정에서 약 3.8초의 오버헤드가 발생하였다. 반면 Kube Armor는 정책 Enforcement 과정에서 eBPF Map만 업데이트하기 때문에 약 0.1초의 오버헤드만 발생하여 가장 우수한 성능을 보였다. 반면, Falco와 Tracee는 정책을 적용하는 과정에서 에이전트를 재배포하는 방식을 사용하므로, 각각 11.8초 및 11.0초의 높은 성능 오버헤드를 기록하였다. 이처럼 에이전트를 재배포하는 방식은 높은 성능 오버헤드 외에도 정책 적용 시 에이전트의 다운타임이 발생하여 일시적인 보안 공백이 생길 수 있다는 단점이 있다.

5. 결론

이 연구에서는 쿠버네티스 환경 내에서 운영되는 ESST의 동작 매커니즘에 대한 종합적인 분석을 제시한다. 기존 연구들이 주로 개별적인 성능 지표에 초점을 맞춘 반면, 본 연구는 각 ESST의 eBPF 구현 방식 및 정책 적용 범위에 대한 정성적 분석을 통해, 이러한 설계 속성이 정책 처리 성능에 미치는 영향을 실증적으로 제시한다. 이러한 분석 결과는 ESST 선택과 구성에 직면한 실무자뿐만 아니라, 컨테이너 네트워크 보안의 발전을 모색하는 연구자에게도 실질적인 기반을 제공할 것으로 기대된다.

Acknowledgements

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 학석사연계ICT핵심인재양성사업의 연구결과로 수행되었음 (IITP-2024-RS-2024-00437024)

참고문헌

[1] Edge Delta Team, “Latest kubernetes adoption statistics: Global insights and analysis for 2024,” https://edgedelta.com/company/blog/kubernetes-adoption-statistics.

[2] National Security Agency (NSA), Cyber security and Infrastructure Security Agency (CISA), “Kubernetes hardening guide,” https://media.defense.gov/2022/Aug/29/2003066362/-1/-1/0/CTR_KUBERNETES_HARDENING_GUIDANCE_1.2_20220829.PDF.

[3] KubeArmor, https://kubearmor.io/

[4] Falco, https://falco.org/

[5] Tetragon, https://tetragon.io/

[6] Tracee, www.aquasec.com/products/tracee/