거대언어모델을 통한 클라우드-네이티브 엣지 클러스터에 대한 악성행위의 관측성

양주영, 김종원*
광주과학기술원 전기전자컴퓨터공학부, 광주과학기술원 AI 대학원*

didwndud3299@gm.gist.ac.kr, jongwon@smartx.kr

Observability of Malicious Behaviors for Cloud-Native Edge Cluster using Large Language Model

Juyoung Yang and JongWon Kim
Gwangju Institute of Science and Technology (GIST)

요 약

본 논문은 클라우드-네이티브 런타임 보안 오픈소스 프로젝트인 Falco 가 제공하는 rule-based observability 의 유효성을 Kubernetes(K8S) 기반 클라우드-네이티브 클러스터 상에서 검증하고, 이러한 rule-based 방식의 한계점을 확인한다. 이러한 한계점을 극복하기 위해 eBPF 와 거대언어모델를 활용한 AI-based observability 파이프라인을 제시하고 이를 검증함으로써 보다 강인한 observability 를 제공할 수 있음을 제안한다.

Ⅰ. 서 론

클라우드-네이티브 시스템은 높은 확장성, 자원 효율성, 경량 가상화 등의 장점으로 인해 최근 서비스 개발 플랫폼으로 각광받고 있다. 이에 따라 클라우드-네이티브 환경에서의 보안 문제 역시 갈수록 중요도가 높아지고 있다[1]. 클라우드-네이티브 환경에서는 많은 컴포넌트들이 동적으로 생성되고 삭제되기 때문에, 클라우드-네이티브 보안에서는 이러한 컴포넌트들의 상태를 실시간으로 추적하고 분석하는 것이 중요하다. 따라서 observability(관측성) 개념이 강조되고 있다. 클라우드-네이티브 환경에서 observability 는 환경에서 발생하는 이벤트들과 상태들을 실시간으로 수집하고 분석하여 문제를 예방하고 해결하도록 지원하게 된다.

최근 eBPF(extended Berkeley Packet Filter)를 활용하여 클라우드-네이티브 환경에서의 observability 를 다루는 방식이 주목받고 있다. eBPF 는 리눅스 커널에 사용자 정의 코드를 적용하는 기술 도구로서, 이를 통해 시스템에서 발생하는 이벤트들을 모니터링할 수 있다[2]. eBPF 를 활용한 대표적인 클라우드-네이티브 런타임 보안 프로젝트 중의 하나인 Falco 는 eBPF 를 통해 시스템 콜과 같은 이벤트 정보를 추출하고, 이벤트 정보들을 기반으로 사전에 정의된 규칙(rule)이 위반되면 경고 메시지를 출력한다.

본 논문에서는 Falco 의 작동원리에 대해 설명한 후, 실제 구축한 쿠버네티스 클러스터에서 Falco 의 유효성을 확인한다. 이후 Falco 가 제공하는 rule-based observability 의 한계점을 확인하고, eBPF 와 거대언어모델 AI 를 통해 이러한 한계점을 극복할 수 있는 AI-based observability 파이프라인을 제시하며, 이것의 강인함을 검증한다.

Ⅱ. Falco 의 작동원리

Falco 는 eBPF 를 통해 커널영역에서 발생하는 시스템 콜과 같은 내부 이벤트 정보들을 유저 영역으로 보낸다. 이후 유저 영역의 Rule Engine 에서는 전달받은 시스템 정보들을 통해 사전에 정의된 rule 이 위반되었는 지를 확인함으로써 이에 대응하는 경고 메시지를 출력한다. Falco 는 rule 을 바탕으로 악성행위들에 대한 observability 를 제공하기 때문에, 그림 1 과 같은 rule 이 정의된 YAML 파일 분석을 통해 Falco rule engine 의 작동 방식을 파악할 수 있다.

```yaml
- rule: shell_in_container
  desc: notice shell activity within a container
  condition: >
    evt.type = execve and
    evt.dir = < and
    container.id != host and
    (proc.name = bash or
     proc.name = ksh)
  output: >
    shell in a container
    (user=%user.name container_id=%container.id container_name=%container.name
    shell=%proc.name parent=%proc.pname cmdline=%proc.cmdline)
  priority: WARNING
```

그림 1. Falco rule 을 정의한 YAML 파일 예시[3]

Falco 는 condition key 에 해당하는 조건들이 모두 만족되면 output key 에 해당하는 경고 메시지를 출력함으로써 악성행위들에 대한 observability 를 제공한다. 이러한 rule-based 방식은 Falco 가 다양한 악성행위들에 대해 구체적이고 정교한 observability 를 제공하도록 만든다.

Ⅲ. 쿠버네티스 클러스터에서의 Falco 유효성 검증

본 논문에서는 클라우드-네이티브 엣지 클러스터에서 Falco 의 유효성을 검증하기 위해 그림 2 과 같이 1 대의 마스터 노드와 2 대의 워커 노드로 구성된 실제 쿠버네티스 클러스터 환경을 구축하였다. Falco 는 각 호스트의 커널에서 발생하느 이벤트를 바탕으로 동작하기 때문에 클러스터를 구성하는 각 노드마다 파드단위로 배치하였다.

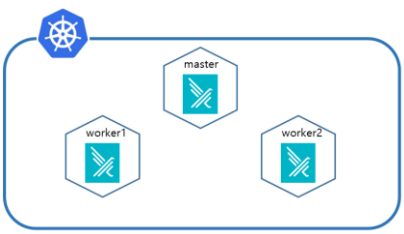

그림 2. Falco 유효성 검증을 위한 쿠버네티스 클러스터 구조

이후 Falco rule 을 위반하는 시스템 이벤트들을 의도적으로 발생시키는 Falco event generator 를 사용하여 클러스터 내부에서 모의 악성행위를 의도적으로 발생시켰다. 그 결과, 각 노드에 배치된 Falco 파드들에서 악성행위에 대한 경고 메시지들이 정상적으로 출력됨을 확인하였다.

Ⅳ. Rule-based 한계점 및 AI-based 필요성

Falco 는 다양한 악성행위들에 대해 각각 구체적인 조건들이 정의된 rule 을 바탕으로 정교한 observability 를 제공한다. 그러나 rule 에 등록되지 않은 패턴으로 악성행위가 이루어지면 이를 제대로 감지하기 어렵다. 대표적으로 Falco 에 등록되지 않은 시스템 콜을 사용한 악성행위는 감지해내지 못한다[4]. 예를 들어 구버전의 Falco 에는 execve, open 등의 기본적인 시스템 콜은 등록되어 있지만, 보다 확장된 기능을 제공하는 execveat, openat 시스템 콜은 등록되어 있지 않다. 따라서 execve 시스템 콜 대신에 execveat 시스템 콜을 사용하여 새로운 프로세스를 실행하는 프로그램을 실행했을 때 구버전의 Falco 가 이를 감지해내지 못함을 확인하였다. 때문에, 공격자들이 Falco 의 보안을 우회하기 위한 수단으로 일부 이러한 등록되지 않은 시스템 콜을 활용한 악성 소프트웨어를 작성할 위험성이 있다. 본 논문에서는 언어 AI 모델을 활용하면 이러한 한계점을 극복하고, 보다 강인한 observability 를 제공할 수 있음을 제안한다. 언어모델을 선택한 이유는, 새롭게 생겨나는 시스템 콜들의 이름은 이와 비슷한 기능을 수행하는 기존의 시스템 콜과 언어적으로 비슷한 이름을 가지고 있기 때문에, 언어모델이 기본적인 시스템 콜 데이터로만 학습했더라도, 새로운 시스템 콜에 대응할 수 있을 것으로 기대하였기 때문이다.

Ⅴ. AI-based Observability 의 파이프라인

Falco 가 사전에 정의된 rule 파일에 근거하여 악성행위를 판별했다면, 본 논문에서 제시하는 방법론은 거대언어모델의 inference 에 근거하여 해당 악성행위를 판단하였다. 그림 3 을 통해 AI-based observability 의 전체적인 파이프라인를 시각적으로 확인할 수 있다.

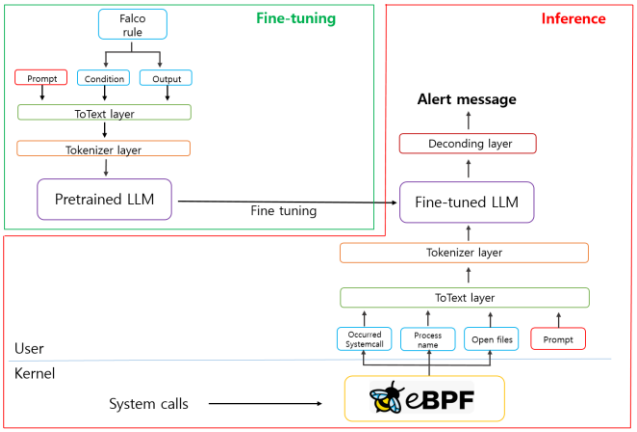

그림 3. AI-based observability 파이프라인

본 논문에서는 최근 메타에서 오픈소스로 공개한 Pretrained LLaMA(Large Language Model Meat AI) 2 13B 을 Falco rule 데이터로 fine-tuning 하여 사용하였다. Inference 과정은 커널 영역에서 eBPF 프로그램을 통해 시스템 콜 정보들을 유저 영역으로 전달함으로 시작된다. 이후 유저영역의 ToText layer 에서는 언어모델에게 task 를 알려주는 프롬프트와의 결합이 이루어지고, Tokenizer layer 에서는 텍스트를 단어와 같은 작은 단위로 나누는 토큰화가 진행된 후 최종적으로 fine-tuning 이 완료된 거대언어모델의 input 으로 전달된다. 거대언어모델은 전달받은 input 토큰들에 기반하여 경고 메시지에 해당하는 output 토큰들을 생성하게 되고, 이후 Decoding layer 를 거쳐 텍스트화된 경고 메시지를 출력한다.

Ⅵ. AI-based Observability 의 강인함 검증

AI-based observability 의 강인함을 검증하려면 execve, open 과 같은 기존 시스템 콜만이 등록된 Falco rule 로 학습된 거대언어모델이 execveat, openat 등의 최신 시스템 콜에 대응하는 경고 메시지를 알맞게 생성할 수 있음을 확인해야 한다. 이를 위해 execveat 시스템 콜을 hook 으로 한 eBPF 프로그램을 작성하여 커널영역에서 execveat 시스템 콜이 발생할때마다 UID, Timestamp, Process name 등의 관련 정보를 유저영역으로 전달하도록 하였다. 이후 그림 3 의 ToText, Tokenizer layer 를 거쳐 프롬프트와 함께 토큰화한 후, fine-tuning 이 완료된 거대언어모델의 input 으로 전달하였다. 그 결과, execve, open 등의 기존의 시스템 콜 정보만을 학습한 언어모델이 execveat, openat 등의 시스템 콜 스트림 데이터를 전달받아 의도에 맞게 경고 메시지를 출력함을 확인하였다.

Ⅶ. 결론

본 논문에서는 대표적인 클라우드-네이티브 런타임 보안 프로젝트인 Falco 의 기능과 작동원리를 소개하고, 실제 구축한 쿠버네티스 클러스터에서 Falco 의 유효성을 검증하였다. 이후 Falco 가 제공하는 rule-based observability 의 한계점을 확인하였고, 이를 극복하기 위한 거대언어모델과 eBPF 를 활용한 AI-based observability 파이프라인을 구현하고 검증함으로써 강인한 observability 를 제공할 수 있음을 제안하였다.

ACKNOWLEDGMENT

본 논문은 2023년도 과학기술원정보통신부 및 정보통신기획평가원의 대학 ICT 연구센터지원사업의 연구결과로 수행되었음 (IITP-2023-2021-0-01835, 인공지능대학원지원(광주과학기술원)).

참 고 문 헌

[1] Q. Zeng, M. Kavousi, Y. Luo, L. Jin, and Y. Chen, “Full-stack vulnerability analysis of the cloud-native platform,” Computers & Security, vol. 129, 2023.

[2] eBPF, https://ebpf.io/what-is-ebpf.

[3] Falco, https://falco.org/docs/rules/basic-elements.

[4] Falco, https://falco.org/blog/falco-monitoring-new-syscalls.