클라우드-네이티브 환경의 eBPF 기반의 Runtime Security 의 설계 및 구현

김두현, 김종원*
광주과학기술원 전기전자컴퓨터 공학부, 광주과학기술원 AI 대학원*

doohkim7@gm.gist.ac.kr, jongwon@smartx.kr

Design and Implementation of Runtime Security Based on eBPF in Cloud-Native Environments

Doohyun Kim and JongWon Kim
GIST (Gwangju Institute of Science & Technology)

요 약

본 논문은 실시간 대응의 관점으로 클라우드-네이티브 컴퓨팅 환경에서의 보안 취약점을 접근하였다. 이를 위해 클라우드-네이티브 환경에서 eBPF 기반의 runtime tracing 도구인 Tracee 를 기반하여 eBPF 의 XDP 도구를 바탕으로 Runtime Response 를 구현하여 Runtime security 를 구현하였다. 해당 도구에 대해 CVE-2019-5736 Container escape 공격을 수행한 결과, 공격의 결과로 나타나는 root 권한의 Reverse shell 을 패킷 레벨에서 차단하는 것을 확인할 수 있었다.

Ⅰ. 서 론

클라우드-네이티브 컴퓨팅 환경이란 서비스를 구성하는 단계에서 클라우드 컴퓨팅 기술을 이용하여 서비스를 모듈화하여 마이크로 서비스 단위로 관리하고 이를 배포하는 개발 환경이다. 클라우드-네이티브 컴퓨팅 환경의 마이크로 서비스는 기존의 가상머신(virtual machime)에 기반한 클라우드 환경과 다르게 일반적으로 컨데이터(Container)의 형태로 구성된다. 따라서 사용하는 리소스가 컨테이너의 유연성과 연계되므로 리소스를 사용하는 환경이 매우 가변적이며 변동성이 크다[1].

그러나 이러한 컨테이너를 기반으로 하여 얻어진 리소스 사용의 유연성의 대가로 클라우드-네이티브 컴퓨팅 환경에서의 개발 환경은 보안 요소를 엄밀하게 설계하지 못한다면 공격자에게도 리소스 활용의 유연성이 부여될 수 있다는 취약점이 있다.

이러한 클라우드-네이티브 환경의 보안 솔루션을 구현하는 방향은 크게 두 가지 방법으로 접근이 이루어지고 있다. 첫 번째 방식은 클라우드-네이티브 개발 환경에서 클라우드 리소스 제공자가 서비스 개발자의 개발 환경 자체의 권한을 제한하는 방식이다. 두 번째 방식은 클라우드 리소스의 제공자가 서비스 개발자의 리소스의 사용에 대해 자율성을 부여하되, 공격자가 리소스를 탈취하여 악의적인 행동을 수행할 경우 이를 실시간으로 탐지하여 대응하는 방식이다[1].

컨테이너 권한을 제한하여 공격을 선제적으로 방어하는 샌드박스(sandbox)등의 방식은 자유롭게 서비스를 구성할 수 있도록 유도하는 클라우드-네이티브 환경의 특성에 있어서 다른 방향성을 가진다. 따라서 본 논문에서는 자유로운 서비스 구성 환경을 권장함과 동시에 리소스 사용 환경에서 의심스러운 동작이 발생할 경우 이를 실시간으로 대처하도록 지원하는 방식 즉, Runtime Security 의 방식을 클라우드-네이티브 컴퓨팅 환경의 주요한 보안 솔루션으로 보고 이를 설계 및 구현해보고자 한다.

Ⅱ. eBPF 를 활용한 Runtime Security 의 실시간성

의심스러운 컨테이너의 동작에 대해 실시간으로 추적하고 대응하는 Runtime security 의 핵심 키워드는 실시간성에 있다. 이러한 실시간성을 구현함에 있어서 eBPF 를 사용하는 것이 가장 효과적인 방식이라고 알려져 있다[2]. eBPF 는 Linux 커널에 직접적으로 관여하여 명령을 내리거나 관리하도록 도와주는 인터페이스로, Kernel level 에 근접하여 곧바로 명령을 내리기 때문에 context switching 이 발생하지 않아 가볍고 강력하다[3]. 이러한 eBPF 의 특성을 활용한다면 보안을 구성하는 코드가 커널 내부에 직접 동작할 수 있기 때문에 Runtime security 의 실시간성을 구현할 수 있다.

Runtime Security 의 구성은 탐지와 대응, Runtime tracing 과 Runtime response 두 단계로 나누어진다. 클라우드-네이티브 환경에서의 Runtime tracing 의 경우 eBPF 를 사용하는 Tracee 라는 Runtime tracing 도구를 통해 eBPF event 들을 추적할 수 있는 것이 확인되었다[4]. 따라서 본 논문은 Tracee 의 runtime tracing 를 기반으로 eBPF 기반의 runtime response 를 구현하여 통합된 runtime security 가 클라우드-네이티브 환경의 보안 솔루션이 될 수 있는지 확인하고자 한다.

Ⅲ. eBPF 를 이용한 Runtime Security 의 구현 방식

Tracee 의 Runtime tracing 을 기반으로 Runtime security 를 구현하는 가장 이상적인 방식은 Tracee 를 사용하지 않고 Runtime tracing 에서 Runtime response 까지 모두 하나의 eBPF code 로 구현하여 대응하는 방식이다. 해당 방식의 경우 공격의 실시간 추적에서부터 대응까지 Kernel-level 에서 온전히 일어나기 때문에 Runtime security 의 가장 이상적인 형태의 모델이라고 볼 수 있다. 그러나 이것을 구현하는 것은 Tracee 의 배포 코드와 raw code level 에서 통합하는 것이 어렵다는 단점이 존재한다.

따라서 본 논문에서는 Tracee 의 log 를 기반으로 eBPF code 를 통해 Kernel-level 에서 대응하는 방식을 사용하였다. 이러한 방식은 Tracee 가 stdout 을 통해 User-level 을 통해 출력한 로그를 바탕으로 eBPF 에

기반한 Runtime response 를 구현하여 Kernel level 에서 대응하는 방식이다. 그러나 log 가 출력되는 과정에서 User level 로의 Context switching 이 발생하기 때문에 엄밀한 실시간성을 부여하지 않는다고 볼 수 있다. 그럼에도 해당 단계를 제외한 부분에서의 실시간 성을 보장하기 때문에 이러한 방식을 채택하였으며, 구현한 runtime Security 가 성공적으로 공격을 차단할 수 있는지 확인하고 실제 대응 시간을 확인하여 실시간성이 유효한지 확인해보려 한다.

Ⅳ. Container escape 공격을 통한 Runtime security 의 환경 구성 및 검증

클라우드-네이티브 환경의 Runtime response 를 구현하기 위해 대응 대상으로 삼은 공격 방식은 CVE-2019-5736 container escape 취약점이다. 해당 취약점은 Docker Container 를 생성하고 실행하기 위한 CLI tool 인 runC 에서 발생하는 취약점으로 /proc/self/exe 와 관련된 file descriptor 를 적절하게 처리하지 못하여 runC 의 바이너리를 덮어씌워 호스트의 root 권한을 탈취하는 Reverse shell 을 형성하는 공격 방식이다[5]. 이러한 공격 방식은 클러스터에 고유한 공격 방식이며, root 권한의 Reverse shell 이라는 강력한 위험을 발생시키기 때문에 공격자의 공격의 대표성을 띈다고 할 수 있다.

CVE-2019-5736 취약점의 경우 Docker Community edition 18.06.2 이전 버전의 docker 에서 작동하므로 docker 18.06.1 버전과 Kubernetes 1.13.1 버전으로 클러스터를 구성하였다. 클러스터는 3 개의 node 로 구성되어 있으며, 하나의 master node 와 두 개의 worker node 로 구성되었다. 해당 클러스터의 각 node 에는 하나의 Tracee v0.17 버전의 pod 를 배치하였으며, 각 pod 에는 하나의 Tracee container 가 존재한다. 또한 하나의 worker node 에는 Ubuntu 18.04 pod 가 구성되어 있으며 해당 pod 의 Container 에서 CVE-2019-5736 취약점을 이용하여 container escape 공격을 수행하였다.

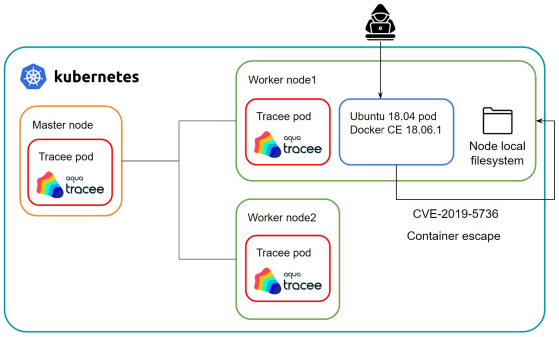

그림 1. Container escape 공격 환경 구성

eBPF 기반의 Runtime response 도구는 eBPF 의 XDP 기능을 이용하여 구현하였다. XDP 는 eXpress Data Packet 으로 Kernel 에서 네트워크 패킷을 처리하기 이전에 해당 패킷을 우선적으로 처리하는 eBPF 도구이다[6]. 본 논문에서의 Runtime Response 의 주요 동작은 Tracee 의 로그에서 'stdio_over_socket' 이라는 의심스러운 활동을 감지할 경우, 해당 IP 를 eBPF 의 XDP 프로세스에 전달하여 해당 IP 에서 오는 패킷을 DROP 하는 기능을 수행하도록 설계하였다.

해당 코드를 실행한 결과 Ubuntu 18.04 pod 를 재실행하여 poc 코드를 실행하였을 때 stdio_over_socket log 발견 이후 해당 주소의 네트워크 패킷을 전부 DROP 하여 Reverse shell 이 차단되는 것을 확인할 수 있었다. 따라서 eBPF 기반의 공격 대응 방식이 성공한 것을 확인할 수 있었다. 이후 'stdio_over_socket' log 가 발생할 이후 네트워크 패킷의 차단이 실행될 때까지의 시간을 Tracee 의 eBPF events 를 통해 관측하는 방식으로 실시간성의 성능을 정량적으로 측정하려 하였다. 그러나 eBPF 코드에 의해 해당 ip 에서 접근하는 패킷이 차단된 이후 새로운 패킷에 의한 eBPF events 가 발생하지 않아 정량적으로 대응시간을 측정하는 것은 실패하였다.

Ⅴ. 결론

본 논문에서는 eBPF 기반의 Runtime security 를 클라우드-네이티브 환경에서 구현하여 서비스 개발자에게 서비스 구성에서의 보안을 구성하면서도 리소스를 자유롭게 사용할 수 있는 환경을 제공하고자 하였다. 이를 위해 eBPF Runtime tracing 도구인 Tracee 를 기반으로 클라우드-네이티브 환경에서 eBPF 의 XDP 를 이용한 Runtime response 를 개발하고 이를 통합하여 Runtime Security 보안 솔루션을 구현하였다. 해당 도구에 대해 CVE-2019-5736 container escape 취약점에 대한 공격을 수행하였고, 구현한 runtime security 보안 솔루션으로 인해 reverse shell 이 탐지된 경우 해당 ip 에서 오는 모든 패킷을 차단하여 reverse shell 공격을 방어할 수 있었다.

본 논문에서 구현한 Runtime security 의 구현 방식은 Runtime tracing 과 Runtime response 사이의 log 를 출력하고 이를 다시 입력으로 사용하는 과정에서 User-level 에서의 프로그램의 진행을 포함하게 된다. 따라서 이러한 부분에서 실시간성을 높일 요소가 존재한다. Runtime tracing 과 Runtime response 의 과정을 하나의 eBPF 코드로 작성하여 중간 과정에서 User level 로의 context switching 에서 발생하는 time cost 를 감소시킨다면 더욱 실시간성이 뛰어난 Runtime security 보안 솔루션이 완성될 것으로 보인다.

ACKNOWLEDGMENT

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

참 고 문 헌

[1] Brandon Krieger, et al. "Cloud Native Security White Paper version 2," CNCF, 2022.

[2] Aqua Security. "Tracee Documentation," Aqua Tracee, 2023, (https://aquasecurity.github.io/tracee/v0.13).

[3] ebpf.io authors. "eBPF Documentation: What Is eBPF," 2023, (https://ebpf.io/what-is-ebpf).

[4] 김두현, "eBPF 기반의 runtime tracing 을 통한 클라우드-네이티브 컴퓨팅 환경의 실시간 추적의 유효성 검증," 2023 년도 한국통신학회 하계종합학술발표회, 2023

[5] M. Reeves, D. J. Tian, A. Bianchi and Z. B. Celik, "Towards Improving Container Security by Preventing Runtime Escapes," 2021 IEEE Secure Development Conference (SecDev), Atlanta, GA, USA, 2021, pp. 38-46, doi: 10.1109/SecDev51306.2021.00022.

[6] Cilium Authors. "BPF and XDP Reference Guide," Cilium, 2023, (https://docs.cilium.io/en/stable/bpf/