KubeRosy: 컨테이너를 위한 동적 시스템 콜 필터링 프레임워크

허진[1], 이승수[2]
[1]인천대학교 컴퓨터공학부 학부생
[2]인천대학교 컴퓨터공학부 교수
gjwls0787@inu.ac.kr, seungsoo@inu.ac.kr

KubeRosy: Dynamic system call filtering framework for containers

Jin Her[1], Seungsoo Lee[2]
[1]Dept. of Computer Engineering, Incheon University
[2]Dept. of Computer Engineering, Incheon University

요 약

최근 대규모 애플리케이션의 효율적인 운영을 위한 컨테이너 기술의 도입이 급격하게 이루어지고 있으며, 이에 따라 컨테이너 환경의 보안을 향상하기 위한 여러 기술이 제안되고 있다. 특히 악성 컨테이너의 무분별한 시스템 콜 사용을 막기 위해 Seccomp 정책을 통한 접근 제어 기술을 제공하고 있지만, 현재 컨테이너가 사용하고 있는 Seccomp의 경우 시스템 콜 정책을 업데이트하기 위해서는 컨테이너를 재배포해야 한다는 한계점을 가지고 있다. 본 논문은 이를 해결하기 위해 eBPF와 LSM을 사용하여 컨테이너 종료 없이 동적으로 시스템 콜 사용을 제한할 수 있는 KubeRosy를 제안한다.

1. 서론

최근 클라우드 환경으로의 마이그레이션 가속화로 컨테이너 환경의 보안 문제 또한 대두되고 있는데, 특히 컨테이너가 호스트와 공유하고 있는 커널에 접근하는 인터페이스인 시스템 콜에 대한 보안 정책은 신뢰할 수 없는 컨테이너로부터 호스트 커널을 보호하는 초석이 된다. 시스템 콜에 대한 적절한 보호 매커니즘의 부재로 인한 취약점의 예시로, ptrace_link가 ptrace 관계를 생성하려는 프로세스의 자격 증명 기록을 잘못 처리하여 execve 시스템 콜을 통해 루트 권한을 얻을 수 있었다[1].

현재 상용화되어있는 시스템 콜 필터링 프레임워크로는 Seccomp(Secure Computing)가 있는데, 이는 컨테이너를 배포하기 전에 시스템 콜들의 허용/차단 목록을 미리 Seccomp profile에 작성하여 런타임에 발생하는 시스템 콜에 대한 허용 여부를 검사하는 방식으로 동작한다. 하지만 Seccomp profile은 컨테이너의 런타임에는 업데이트가 불가능하다는 한계점이 있다.

따라서 본 논문은 이와 같은 한계를 극복하기 위해 LSM(Linux Security Module), eBPF(extended BPF)를 사용하여 동적으로 시스템 콜 정책을 업데이트할 수 있는 Kuberosy를 제안한다.

2. KubeRosy 디자인

2.1 아키텍처 개요

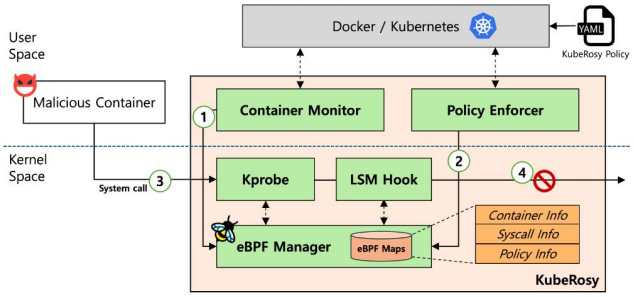

그림 1 KubeRosy architecture and workflow

Kuberosy의 전반적인 아키텍처는 그림 1과 같으며, 실행 흐름은 다음과 같다. (1) 악성 컨테이너가 생성되어 컨테이너 이벤트가 발생하면 컨테이너 런타임 API에서 이를 조회하여 컨테이너 모니터로 컨테이너 정보들을 전달하고 eBPF 맵에 저장한다. (2) Policy Enforcer 모듈에 정책 정보가 들어오면, 정책의 대상이 되는 컨테이너를 찾은 후 정책 정보를 eBPF 맵에 저장한다. (3) 컨테이너에서 시스템 콜이 발생하면 Kprobe가 호출되어 발생한 시스템 콜 정보를 eBPF 맵에 저장한다. (4) LSM Hook의 콜백함수가 실행되면 eBPF 맵을 조회하여 어떤 시스템 콜이 어떤 컨테이너에서 발생했는지 확인한 후 정책이 적용되어있는지 확인하여 차단/허용 한다.

2.2 eBPF와 LSM Hook을 사용하는 이유

LSM Hook은 시스템 콜의 실행 흐름에 직접적으로 관여하여 시스템 콜을 차단/허용하는 동작이 가능하지만, 하나의 LSM Hook이 한가지 시스템 콜이 아닌 여러 시스템 콜에 의해 트리거되기 때문에 LSM Hook 만으로는 어떤 시스템 콜에 의해 LSM Hook이 발생했는지 알 수 없기 때문이다. 이를 해결하기 위해, eBPF를 사용하여 실행되는 시스템 콜을 추적한 뒤, 시스템 콜을 특정하여 정보를 저장한다. 이후, 트리거 되는 LSM Hook의 콜백함수에서 직전 호출된 시스템 콜의 정보를 조회하여 시스템 콜 별 정책을 적용할 수 있게 된다.

2.3 정책 상속

Kuberosy는 파드에 대한 시스템 콜 정책을 컨테이너의 pid 기반으로 적용하므로, 컨테이너 프로세스 외의 자식 프로세스, 형제 프로세스에 대한 정책의 상속은 매우 중요하다. 컨테이너 모니터에서 컨테이너 런타임 API로부터 얻어오는 pid 정보는 컨테이너에서 실행하는 명령의 pid로, 해당 프로세스의 부모 프로세스가 컨테이너 런타임 프로세스이다.

컨테이너 런타임에 시스템 콜 정책을 적용하게 되면 컨테이너 런타임에서 필수적으로 사용하는 시스템 콜은 차단할 수 없다는 부작용이 생긴다. 이러한 부작용을 방지하며 상속 매커니즘을 구현하기 위해 KubeRosy는 tracepoint 에서 부모 프로세스를 조회하며, 정책이 적용되어 있는지 혹은 컨테이너 프로세스와 부모-자식 관계에 있는지 확인한 후 관계가 있다면 eBPF 맵에 저장하여 정책을 상속한다.

3. 실험 결과

커널에는 396개[2]의 시스템 콜들이 존재하는데 Kuberosy에서는 명령 실행, 권한, 네트워크와 관련된 위험도가 높다고 판단되는 시스템 콜 28개를 대상으로 프로토타입을 구현하였다[3]. KubeRosy의 정책 시행 테스트는 Ubuntu 22.04에서 C언어 코드를 구성한 후 정책을 적용하여, 적용된 시스템 콜을 정확히 차단하는지 확인하는 방식으로 진행하였다.

실험 결과, KubeRosy에서 지원하는 28개 모든 시스템 콜들에 대하여 정책이 적용되는 것을 확인할 수 있었다. 예를 들어, ubuntu-krp 파드에 대하여 nc 프로그램의 bind 시스템 콜 사용을 block 하기 위해 그림 2의 정책을 적용한다면 이후 그림 3 같이 허용되지 않는 것을 확인할 수 있다.

```yaml
apiVersion: security.kuberosypolicy.com/v1
kind: KubeRosyPolicy
metadata:
    name: ubuntu-krp
    namespace: default
spec:
    selector:
        matchLabels:
            app: ubuntu
    action: Block
        syscall:
        - bind
```

그림 2 KubeRosy 정책

```
k8s-master-node$ kubectl apply -f block-bind.yaml
kuberosypolicy.security.kuberosypolicy.com/ubuntu-krp created
ubuntu-krp# nc -l 8080
nc: Operation not permitted
```

그림 3 실험 결과

4. 결론

본 논문에서는 LSM과 eBPF를 활용하여 업데이트가 불가능한 seccomp의 단점을 보완하기 위한 시스템 콜 보안 프레임워크를 제안하였다. 실험 결과, 28개의 시스템 콜에 대해서는 정상적으로 동작하는 것을 확인하였으나 전체 시스템 콜 수 대비 부족하다는 한계점이 있다. 따라서 향후 연구에서는 기존의 프레임워크에 시스템 콜을 추가하고 인자 값 기반의 필터링을 통한 더 세분화된 시스템 콜 보안 프레임워크를 구현하는 방향으로 진행할 것이다.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) Grant through the Korea Government, Ministry of Science, ICT (Information and Communication Technology) and Future Planning (MSIP), under Grant 2022R1C1C1006093.

참고문헌

[1]CVE-2017-5123.https://www.cvedetails.com/cve/CVE-2017-5123/

[2]Bootlin,“Syscall_64.tbl-linuxsourcecode(v5.15),”2021.[Online].Available:https://elixir.bootlin.com/linux/v5.15/source/arch/x86/entry/syscalls/syscall_64.tbl

[3]S. Ghavamnia, T. Palit, S. Mishra and M. Polychronakis, “Temporal system call specialization for attack surface reduction,” in Proc. of 29th USENIX Security Symp., Boston, MA, USA, 2020, pp. 1749 - 1766.