DevSecOps 관점의 클러스터 계층 내 매니페스트 정적 분석에 관한 연구

김가현[1], 김미진[1], 천예린[1], 현혜연[1], 김성민[2]
[1]성신여자대학교 융합보안공학과 학부생
[2]성신여자대학교 융합보안공학과 교수
{20211039, 20221081, 20211103 ,20221143, sm.kim} @sungshin.ac.kr

A Study on Static Analysis in Cluster Layer Manifest from the DevSecOps Perspective

Ga-Hyun Kim[1], Mi-Jin Kim[1], Ye-Rin Chun[1], Hye-Yeon Hyeon[1], Seongmin Kim[1]
[1]Dept. of Convergence Security Engineering, Sungshin Women's University

요 약

컨테이너 오케스트레이션 도구로 쿠버네티스가 가장 많이 사용되고 있으며, 관련 취약점 연구는 DevSecOps 관점에서 4C layer로 분류된 클라우드 보안 계층 중, 클러스터 및 코드 계층에 초점이 맞춰져 왔다. 반면에 클러스터 계층에도 네트워크 정책, 인증 제어, 권한 설정 및 매개변수 설정에 관한 취약점이 존재한다. 이에 본 논문은 취약점을 분석하여 OWASP 10과 접목하여 분류한 뒤 예방법을 제시함으로써 앞으로의 안전한 클러스터 계층 구축에 기여하고자 한다.

1. 서론

최근 클라우드 환경에서는 개발부터 소프트웨어 생명주기의 끝까지 보안을 고려하여 DevOps 환경을 만드는 DevSecOps 패러다임이 주목받고 있다. 구체적으로, 클라우드 보안 계층을 클라우드 인프라, 클러스터, 컨테이너, 코드로 구성된 4C layer로 구분하여 각 계층별 대응을 위한 보안 기술들이 연구되고 있다. 그러나 배치 전 단계에서의 프로파일링 및 스캐닝 기반 취약점 분석 기술의 경우, 컨테이너 및 클라우드 인프라 계층에 초점이 맞춰져 있다[1].

특히 클러스터 계층의 취약성의 경우, 대부분 쿠버네티스(Kubernetes)와 같은 클러스터 오케스트레이션 소프트웨어 자체가 취약한 경우보다 이를 활용하여 클라우드 네이티브 환경을 운영하는 운영자의 misconfiguration, 보안 정책 미준수로 인해 발생하는 경우가 대다수이기에 정적 분석 단계에서의 탐지가 필요하다. 실제로 OWASP Kubernetes Top 10 [2] 내 9개의 취약점이 설정 오류(misconfiguration), 보안 정책 미준수로 인해 발생하는 경우의 해당한다. 이에 본 연구에서는 클러스터 계층 내 어떤 취약성이 DevSecOps 사이클 내에서 배포 및 운영 전 정적 분석을 통해 탐지 가능한지에 대한 매핑을 수행하고, 클러스터 계층에서의 취약점 스캐닝을 위한 정적 분석 도구의 요구사항을 도출하고자 한다.

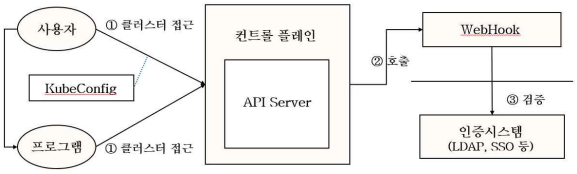
2. 클러스터 계층 취약성 분석

본 연구에서는 OWASP Kubernetes Top 10 [2] 을 기준으로 클러스터 계층 취약점 유형 별로 매핑 가능한 매니페스트 파일을 파악함으로써 정적 분석 단계에서 탐지 가능한 취약성을 분석하였다.

네트워크 정책: 쿠버네티스는 기본적으로 노드별 네트워크 정책을 자동으로 생성해주지 않으며. 사용자는 NodeSelector, NodeAffinity 등을 사용하여 노드별 정책을 구성하고 적용해야 한다. 이를 적용하지 않으면 노드 간 미끼 공격, 악의적인 노드 접근, 부적절한 노드 자원 활용 등이 나타날 수 있다. 이에 대해 CIDR 표기법을 사용할 수 있지만, 구체적으로 Kubernetes ID로 노드를 대상으로 지정할 수는 없다. 또한, 현재의 쿠버네티스의 네트워크 정책만으로는 내부 클러스터 트래픽이 공통 게이트웨이를 통과하도록 강제하지 못한다. 다시 말해, 클러스터 내부의 모든 통신이 특정한 라우팅 규칙을 따르도록 강제하지 못해 취약점이 발생할 수 있다. 따라서, 네트워크 정책 관련 YAML 파일 내 이러한 보안 정책이 명시되었는지에 대한 검사가 필요하다.

인증 제어: 쿠버네티스는 클라이언트 인증서, 전달자 토큰 또는 인증 프록시를 사용하여 인증 플러그

인을 통해 API 요청을 인증한다. 그 중 Webhook 인증의 경우 전달자 토큰을 확인하기 위한 hook이다. 구성 파일은 kubeconfig 파일 형식을 사용하며 파일 내에서 원격 서비스를 참조하고 API 서버 Webhook을 참조한다. 해당 파일 내 미흡한 인증 및 인가 설정이 있으면, 누구나 kubelet API를 호출할 수 있게 된다. 이외에도 정책에 포함된 인증 메커니즘을 사용하기 위해서는 –authorization-mode=[인증 메커니즘] 명령어를 정책 플래그에 포함시켜야 한다. 이때 허가받지 않은 인증 메커니즘 정책 플래그에 포함되어 있을 경우 취약점이 발생할 수 있다.



(그림 1 WebHook 연동 인증)

권한 설정: 클러스터 내 구성요소 간 인증 및 권한을 설정하고 관리하는 kubelet과 관련하여 취약한 권한 설정이 이루어질 수 있는 부분은 플랫폼 및 시스템에서 사용되는 보안 접근 제어 방식인 RBAC(Role Based Access Control)이다. RBAC는 클러스터의 리소스에 수행할 수 있는 작업을 관리하는 세분화된 권한을 만들어주며, YAML 또는 JSON 파일을 사용하여 RBAC를 정의한다. 이때, 서비스 계정, 사용자, 그룹 등의 주체가 쿠버네티스에 내장된 슈퍼유저(Super User)인 cluster-admin에 액세스함으로써 클러스터 내의 모든 리소스에 대해 접근 및 수행 권한을 갖는다. 다시 말해, 기본 namespace의 모든 pod에 cluster-admin 권한이 부여되면 공격자가 서비스를 위장하여 침해할 수 있다. 따라서, RBAC가 정의되는 YAML 파일이나 JSON 파일의 설정 단계에서 권한 검증이 필요하며, namespace에 대해서 슈퍼유저 권한이 부여된 경우 해당 권한 부여가 적절한지 검사하는 과정이 필요하다.

매개변수 설정: 매개변수의 기본값이 보안 정책과 다른 경우, 악의적인 사용자가 보안 정책과 맞지 않는 pod를 실행시키는 등 보안 정책을 우회하는 문제를 초래할 수 있다. 쿠버네티스에서 프로그램이나 시스템의 동작을 제어하거나 설정하기 위한 매개변수는 YAML 파일이나 환경 변수 등을 통해 설정된다. 예를 들어 anonymous-auth는 가장 위험한 구성 오류 중 하나로, kubelet에 대한 인증되지 않은 요청을 허용하는 익명 인증 설정이다. Kubelet 서버에 대한 요청이 활성화되면 구성된 다른 인증 방법에 따라 거부되지 않은 요청은 익명 요청으로 처리되는데, 보안 정책에 따라 민감한 데이터나 시스템에 대한 접근을 제한하는 등 익명 인증을 허용하지 않아야 할 수 있다. 또한, --allow-privileged는 특권 컨테이너의 실행을 허용하는 기능인데 이를 설정할 경우, 장치 cgroup 컨트롤러에 의해 적용되는 모든 제한 사항도 함께 해제되어 호스트가 할 수 있는 거의 모든 작업을 수행할 수 있게 된다. 이렇듯 매개변수의 기본값이 보안 정책과 다르게 설정된 경우를 정적 분석 단계에서 탐지하여 보안 정책을 준수하는 매개변수 설정으로 변경해야 한다.

3. 결론 및 제언

본 연구에서는 Kubernetes OWASP Top 10을 기반으로 매니페스트 정적 분석을 통해 사전에 취약점을 방지할 수 있음을 확인했다. 최근 이러한 정책 위반 사항 탐지를 위해 서비스 메시를 사용하거나 코드를 분석하여 보안 이슈를 식별하는 도구인 tfsec[3] 등의 자동화 분석 도구를 이용하고 있으며 tfsec에서는 네트워크 정책 검사를 수행하고 있다. 그러나 이는 클라우드 인프라 계층 내 정책 검사 대비 단순한 형태이며, 위에서 분석한 설정 오류로 인해 발생할 수 있는 쿠버네티스 취약성을 모두 커버하지 못한다. 따라서, DevSecOps 관점에서 클러스터 계층 취약점을 보다 체계적으로 다루는 자동화 환경을 구축하여 보안성을 높이기 위해 기존 정적 분석 도구를 개선하여 설정 오류를 감지함과 동시에 쿠버네티스 리소스 보안 강화를 위한 다양한 네트워크 정책 분석 기능의 확장이 필요하다.

참고문헌

[1] 곽송이. "eBPF 기반 쿠버네티스 오케스트레이션 컨테이너 런타임 보안 시스템." 국내 석사학위논문 숭실대학교 대학원, 2023. 서울

[2] The OWASP Foundation, "K03: Overly Permissive RBAC": https://owasp.org/www-project-kubernetes-top-ten/2022/en/src/K03-overly-permissive-rbac

[3] tfsec, https://github.com/aquasecurity/tfsec?tab=readme-ov-file