컨테이너 기반 가상화 환경에서의 보안 취약점 분석 및 강화 방안에 관한 연구

정인철, 김정윤*
강남대학교, *경희대학교

natsubakiy@gamil.com, *a321@khu.ac.kr

Study on Security Vulnerability Analysis and Enhancement Measures in Container-Based Virtualization Environments

Jeong In Cheol, Kim Jung Yun*
Kangnam Univ., *kyung Hee Univ.

요 약

본 논문은 가상화 환경 구축 시, 운영체제 단위로 애플리케이션을 분리하던 가상머신(VM)과 다르게 애플리케이션별로 독립된 단위인 컨테이너를 생성하여 서로 다른 개발 환경에서도 쉽게 공유할 수 있는 컨테이너 기반 가상화 환경에서 발생하는 취약점을 조사하고 이를 3 가지 관점에서 대응하고자 한다. 이미지 파일 관점에서는 관리자가 신뢰할 수 있는 이미지를 확보하고, 자체적인 업데이트 프로세스를 수행하는 것이 중요하다. 네트워크 환경 관점에서는 목적과 중요도에 따라 네트워크를 분리하여 보안을 강화할 수 있다. 호스트 OS 의 커널 관점에서는 컨테이너 간 커널 리소스를 공유하지 않는 카타(Kata) 컨테이너를 사용하거나, 최소 환경만 제공하는 CoreOS 를 활용하여 공격 표면을 최소화할 수 있다. 이러한 방안을 통합해 컨테이너 환경을 설계하였을 경우, 컨테이너 기반 가상화 환경에서 보안을 강화할 수 있는 방안을 제시하였다.

Ⅰ. 서 론

컴퓨터 환경을 패키징 하는 대표적인 방법으로 가상머신(VM)과 컨테이너를 들 수 있다. 하나의 OS 환경에서 다양한 애플리케이션 간 플랫폼을 분리하려면 각 환경에 적합한 OS 를 구축해야 했던 가상머신(VM)과 달리, 컨테이너 엔진은 파일을 통해 마치 다른 OS 에서 동작하는 것과 같은 실행 환경을 제공하는 기술이다. 이는 서로 다른 개발 환경에서도 쉽게 공유할 수 있어, 애플리케이션 구축과 실행을 자동화하는 DevOps(Development and Operations)에서 그 장점이 극대화되어 오늘날 보편화된 아키텍처로 자리 잡았다.

그러나 이러한 새로운 아키텍처에 대한 보안 설계가 부재할 경우, 이는 심각한 시스템 취약점으로 노출될 수 있다.[1] 미국의 오픈 소스 소프트웨어 기업인 레드햇의 2024 에디션 쿠버네티스 보안 현황 리포트에서 업계 종사자 600 명을 대상으로 조사한 결과를 보면 취약점, 구성 오류/노출, 공격, 컴플라이언스 실패 중 현재 컨테이너 환경에서 가장 우려되는 위험 요소로 취약점을 선택한 응답자가 33%로 가장 많았다.[2] 이는 안전하지 않은 상태의 컨테이너 환경이 여전히 적지 않으며, 그 위험성을 업계에서도 인지하고 있음을 보여준다.

Ⅱ. 본론

본 논문에서는 이미지 파일, 네트워크 환경, 호스트 OS 의 커널이라는 3 가지의 관점에서 발생할 수 있는 보안 취약점을 조사하고, 각 요소에 대응할 수 있는 방안을 모색하였다. 이러한 서로 다른 관점의 방안을 융합하여 적용한다면 컨테이너 환경의 보안성을 통합적으로 강화시킬 수 있을 것으로 기대한다.

2.1. 이미지 파일

이미지는 특정 애플리케이션을 실행하는 데 필요한 모든 구성 요소를 포함하는 파일이다. 소프트웨어 업데이트는 전통적으로 실행되는 호스트에서 수행하는 것이 일반적이지만 컨테이너 환경에서는 이미지를 업데이트한 후 다시 배포하는 방식으로 이루어지기 때문에 호스트에서 자체적인 이미지를 업데이트하기란 상대적으로 어려운 부분이 있다.

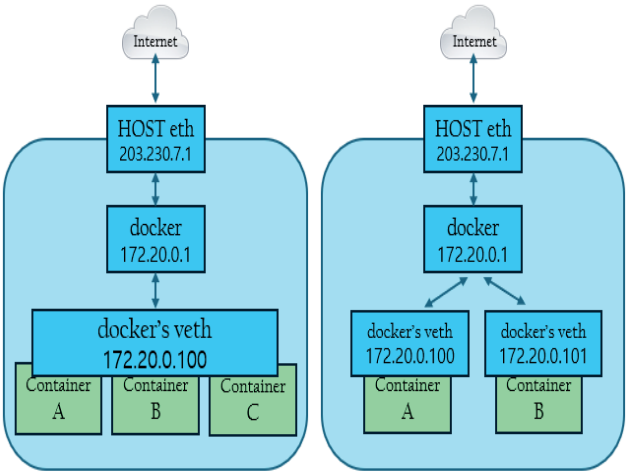
또한 이미지 파일이라는 개념은 환경설정 리스트를 묶어놓은 파일에 불과하다. 따라서 악성코드가 포함된 파일이 이미지에 함께 묶일 수 있으며 이러한 악성코드는 이미지 내 다른 컴포넌트로 퍼질 수 있고, 해당 이미지 파일이 실행되는 환경, 나아가 해당 환경의 다른 컨테이너 환경까지 영향력이 미칠 수 있다. 따라서 신뢰할 수 있는 경로로 이미지를 제공받고, 관리자의 자체적인 프로세스와 보안에 대한 인식이 중요한 부분이다.

2.2. 네트워크 환경

컨테이너 환경 또한 네트워크 트래픽을 통제해야 한다. 전통적인 아키텍처에서 사용하는 패턴과 유사하지만, 컨테이너와 컨테이너의 통신이 가상화된 네트워크 모델은 호스트에 배치된 컨테이너 사이의 통신이 가상의 암호화된 네트워크를 사용하는 것이 일반적이기 때문에 기존의 네트워크 장비로는 이 트래픽을 볼 수 없다.

컨테이너의 배포와 관리를 도와주는 오케스트레이터가 컨테이너를 배치할 때 컨테이너의 IP 주소를 동적으로 할당하는 것이 일반적이며, IP 주소는 앱을 확장하거나 부하를 분산하면서 지속적으로 변경된다.

이처럼 컨테이너 앱은 변화의 속도가 빠르다는 특성 탓에 네트워크 토폴로지가 자주 변하기에 컨테이너에서 실행되는 애플리케이션의 특성에 기반하여 트래픽을 필터링할 수 있도록 규칙을 동적으로 관리해야 한다.

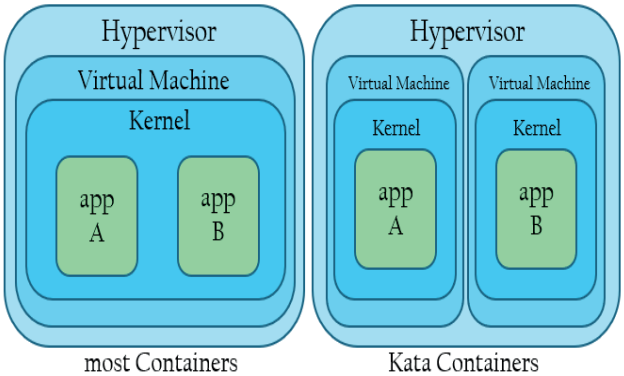


(그림 1) 컨테이너 격리 유무에 따른 오버레이 네트워크 구성도

대부분의 컨테이너 환경에서 개별 컨테이너 간의 트래픽은 가상 오버레이 네트워크를 통해 라우팅 된다. 이 가상 오버레이 네트워크는 기존의 네트워크 보안 및 관리 도구를 사용할 수 없는 경우가 많다. 이로 인해 외부에서 접근하는 공격에 대해 취약한 부분이 존재한다는 점과 호스트 환경 내에 이미 침투한 악성 컨테이너가 공격 애플리케이션을 실행시켜 과도한 네트워크 트래픽을 발생시켜 다른 정상적인 컨테이너가 네트워크에 접근할 수 없거나 매우 느리게 접근하게 되어 서비스 거부 상태에 빠질 수 있다. 따라서 이러한 네트워크 취약점의 대안으로는 네트워크 단에서 침입 탐지 시스템(IDS)을 도입하여 보안성을 강화하거나, 컨테이너에서 각 컴포넌트마다 리소스 제한을 설정하여 공격자가 리소스를 과다하게 사용할 수 없게끔 설정하는 것 또한 방법이다.[3] 컨테이너 엔진이나 네트워크 장비에서 IP 를 기준으로 필터링하고, 컨테이너가 침투당했을 때 리스크를 최소화하는 것을 목적으로 애플리케이션마다 목적, 중요도, 위협 수준과 같은 기준으로 단계를 세분화하여 네트워크 트래픽을 각각 별도의 가상 네트워크로 분리하도록 구성하여 세그먼테이션을 분리하는 구성도를 고려할 수 있다.[4]
이와 같은 네트워크 환경 설계는 실제로 구축하기 이전에 충분히 고려해야 한다.

2.3. 호스트 OS 의 커널

컨테이너 기반 환경은 가상 머신(VM)과 다르게 호스트 OS 의 커널 영역 리소스를 각 애플리케이션이 함께 사용한다. 이로 인해 가볍다는 장점이 존재하지만, 그만큼 노출된 시스템 취약점이 존재한다. 따라서 가능한 노출되는 공격 표면을 최소화하는 보안 조치를 시행해야 한다.



(그림 2) 대부분의 컨테이너와 카타 컨테이너의 커널 구조

카타(Kata) 컨테이너는 오픈스택 재단에서 개발 중인 경량 가상 머신(VM) 기반으로 컨테이너를 실행하는 엔진이다. 컨테이너와 가상 머신의 차이는 호스트 머신과 커널이 공유되는지 안되는 지로 나눌 수 있었지만, 카타(Kata) 컨테이너는 컨테이너 전용 가상 머신을 준비한다는 점에서 새로운 접근이라고 할 수 있고, 경량 가상 머신(VM)을 사용하기 때문에 일반적인 가상 머신보다 빠르며, 커널을 공유하는 일반적인 컨테이너보다 안전하게 샌드박스를 구현하는 것이 가능하다.[5] 또 다른 방법으로는 호스트 OS 로 CoreOS 를 사용하는 것이다. 이는 다른 리눅스 배포판처럼 다양한 기능을 제공하지 않고 오직 컨테이너 기반의 환경을 구축하는 최소한의 기능만을 탑재한 클라우드 네이티브 컴퓨팅 운영체제를 사용하는 것이다. 이로 인해 컨테이너 별로 호스트 OS 의 커널 리소스를 각 애플리케이션이 사용할 때 관리해야 하는 공격 표면이 최소화되는 효과를 볼 수 있다.

Ⅲ. 결론

본 논문에서는 DevOps 환경에서 많이 쓰이고 있는 컨테이너 기반의 가상화 환경에서 이미지 파일, 네트워크 환경, 호스트 OS 의 커널이라는 3 가지의 관점에서 생길 수 있는 보안 취약점을 조사하여 대안을 제시하였다.
이미지 파일에서는 신뢰할 수 있는 이미지를 확보하는 것과 관리자 자체적인 업데이트 프로세스와 보안에 대한 경각심이 중요했고, 네트워크 환경에서는 컨테이너 엔진에서 애플리케이션마다의 중요도나 용도에 따라 네트워크를 분리하여 필터링에 용이하게 만드는 방법이 효과적임을 알 수 있었다. 호스트 운영체제의 커널을 보호하기 위해서 점유율이 가장 높은 도커 컨테이너만이 아닌 가상 머신(VM)의 특징이었던 애플리케이션 간 공유하지 않는 커널로 네임스페이스를 분리하는 특징을 가진 카타(Kata) 컨테이너를 사용하거나 호스트 운영체제를 컨테이너 환경 구축에 최적화 되어있는 CoreOS 를 사용하여 노출되는 공격 표면을 줄이는 방법이 존재했음을 알 수 있었으며, 본 논문을 통하여 가상화 컨테이너 환경을 구축할 때 제안된 대안을 융합한 방식으로 설계한다면 통합적인 보안 수준이 향상된 설계가 이루어질 수 있다고 판단 된다. 추후 연구에서는 컨테이너 관리 툴인 오케스트레이터의 관점을 추가하여 보다 규모가 큰 엔터프라이즈 환경에서의 보안 설계를 연구하고자 한다.

참 고 문 헌

[1] NIST SP 800-125, "Guide to Security for Full Virtualization Technologies"

[2] Red Hat, "컨테이너 보안이란?" (https://www.redhat.com/ko/topics/security/container-security)

[3] NIST SP 800-190, "Application container Security Guide"

[4] 이하늘, 우승찬, 이종혁. (2021-11-17). 도커 컨테이너 간 통신에서 발생 가능한 보안 위협 분석. 한국통신학회 학술대회논문집, 전남.

[5] katacontainers, "Kata Containers Architecture (https://github.com/kata-containers/kata-containers/tree/main/docs/design/architecture#kata-containers-architecture)