도커 컨테이너 취약점 동향 분석*

김동은[1†], 이준희[1], 김진우[2‡]

[1,2]광운대학교 (학부생, 교수)

Exploring Vulnerability Trends in Docker Containers*

Dong-Eun Kim[1†], Jun-Hee Lee[1], Jin-Woo Kim[2‡]

[1,2]Kwangwoon University (Undergraduate student, Professor)

요 약

오늘날 많은 기업의 서비스가 컨테이너 기술을 통해 제공되고 있다. 컨테이너 환경은 동일한 호스트를 공유한다는 특징 때문에 기존 가상 환경보다 보안을 더욱 엄격하게 적용해야한다. 이를 위해서는 컨테이너의 주요 취약점을 면밀히 조사하고 파악하는 것이 필요한데 기존에 이를 체계적으로 수행한 연구는 존재하지 않았다. 따라서 본 논문에서는 최근 5년간 보고된 컨테이너 보안 취약점을 조사하고 이에 대한 동향을 분석하고자 하였다. 특히 가장 널리 사용되는 컨테이너 플랫폼인 도커 컨테이너에 대한 취약점을 15개 조사하고 이에 관한 5가지 공격 분류를 제시하였다. 마지막으로 안전한 컨테이너 환경을 위한 보안 가이드라인을 제시하였다.

I. 서론

최근 리눅스 컨테이너(container)가 클라우드 환경의 주요 구성 요소로 주목받고 있다. 컨테이너는 기존의 가상머신을 대체하는 새로운 가상화 기술로써 운영체제 수준에서 자원을 가상화한다. 이를 통해 기존 하이퍼바이저 기반 가상화 기법보다 효율적으로 서버 자원을 가상화할 수 있다. 이러한 장점으로 인해, 도커(Docker), LXC 등의 컨테이너 플랫폼과 쿠버네티스(Kubernetes)와 같은 컨테이너 오케스트레이션이 등장하여 다양한 컨테이너 기반 서비스의 개발을 촉진하였다. 그러나 한편으로는 컨테이너라는 새로운 구조의 등장으로 다양한 보안 취약점들이 발견되었다. 컨테이너 환경은 기존 가상머신과는 다르게 컨테이너와 호스트 간 커널을 공유한다는 특징이 있어서, 만약 공격자가 컨테이너의 취약점을 공격하였더라도 호스트로 그 공격이 이어질 수 있다.

이러한 점 때문에 클라우드를 이용하는 테넌트는 컨테이너를 배포하기 전에 보안 문제를 사전에 파악하고, 대비하는 것이 중요하다. 그러나 최근 많은 컨테이너 기반 애플리케이션 및 플랫폼이 등장함에 따라 다양한 컨테이너 취약점 또한 보고되었는데, 이를 모두 숙지하고 검증하기는 어려운 일이다. 따라서 주요 취약점에 대해 면밀히 파악하여 컨테이너의 보안 문제를 분석하는 것이 필요하다.

본 논문에서는 지금까지 보고된 컨테이너의 취약점을 체계적으로 조사하고 주요 특징을 분석하고자 한다. 특히 널리 사용되는 컨테이너 플랫폼인 도커를 중심으로 최근 5년간 MITRE에 보고된 CVE (Common Vulnerabilities and Exposures) 15개를 파악하고 이를 5개의 주요 카테고리로 분류하였다.

* 이 논문은 2023년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. RS-2022-00166401)

† 주저자, skatjdehddms@kw.ac.kr

‡ 교신저자, jinwookim@kw.ac.kr

II. 배경지식

컨테이너란 호스트와 커널을 제외한 자원을 격리하여 프로세스를 작동하는 환경을 말한다. 호스트와 격리된 환경을 만들기 위해 리눅스 커널의 네임스페이스(namespace)와 컨트롤 그룹(cgroup)을 이용한다. 본 논문에서는 가장 널리 사용되는 컨테이너 플랫폼인 도커를 중심으로 컨테이너 환경의 구조를 설명하도록 한다.

2.1 도커 컨테이너 생성 과정

Fig. 1과 같이 도커의 컨테이너 생성 작업은 아래의 컴포넌트로 구성된 도커 엔진(Docker engine)과 컨테이너 런타임(runC) 간 상호작용으로 수행된다.

- **dockerd**는 사용자와 상호작용을 하기 위한 인터페이스로 클라이언트로부터 API 호출을 받고 이를 해석한 후 containerd로 전달하며 컨테이너 생성을 요청한다.
- **containerd**는 이미지의 관리, 실행 및 저장 등을 담당한다. 컨테이너 생성에 필요한 이미지를 가져와 컨테이너 생성 준비를 한다.
- **shim**은 runC와 containerd 사이의 인터페이스로 컨테이너의 생성, 종료, 자원 등을 관리하며, containerd로부터 요청을 전달받고 컨테이너 런타임에 생성을 요청한다.
- **runC**는 컨테이너 런타임으로 실질적으로 컨테이너의 생애주기를 관리한다. runC는 리눅스 커널에 접근하여 네임스페이스, cgroup을 요청하여 최종적으로 컨테이너를 생성하고 프로세스를 실행한다.

2.2 도커 이미지와 도커 허브

컨테이너는 이미지(image)라는 메타 데이터의 인스턴스인데, 이는 컨테이너의 청사진 역할을 한다. 이미지는 레이어(layer)의 집합으로 되어있으며 이는 공통 레이어를 공유하여 저장 시 공간을 최소화하고 만약 기존 이미지를 수정하여 새롭게 빌드할 때 해당 부분만 업데이트하기 위함이다.

containerd는 컨테이너 생성 시 호스트에 이

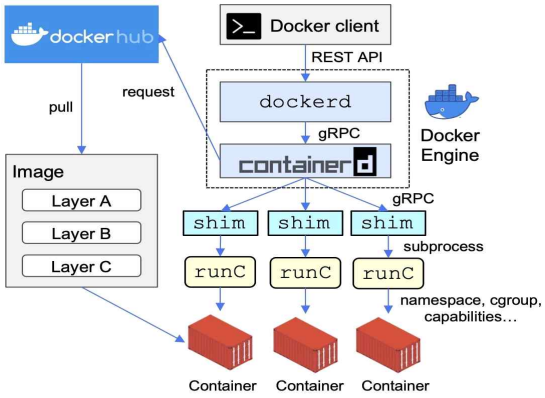

Fig. 1 The process of creating containers in the Docker engine

미지 정보가 없는 경우 도커 허브(Docker Hub)와 같은 레지스트리(registry)에서 이미지를 가져온다. 도커 허브는 이미지 공유 플랫폼으로 도커 허브에서 인증한 이미지 또는 일반 사용자가 만든 이미지를 공유하기 위해 사용된다.

이미지를 가져오는 행위를 풀(pull)이라고 하는데 이때 호스트 혹은 사용자 레지스트리에서 해당 이미지를 구성하는 레이어가 있는지 확인하고, 없다면 도커 허브의 다른 사용자 레지스트리로부터 풀을 한다. 하지만 이러한 과정에서 악성 레이어가 섞여 들어올 수 있다. 실제로 도커 허브에는 맬웨어가 있거나 가상화폐 채굴기를 돌리는 이미지가 공유되고 있으므로[1], 이미지를 풀 하거나 빌드할 때 주의가 필요하다.

III. 도커 컨테이너 취약점 동향

본 논문에서는 5가지 공격 분류에 따라 CVE를 구분하고, 취약점의 특징을 이미지에 대한 취약점, 컨테이너에 대한 취약점으로 나누어 분류하였다(Table 1). 본 절에서는 각 공격 분류의 정의와 주요 CVE에 대해 설명하도록 한다.

3.1 컨테이너 이스케이프(container escape)

컨테이너 이스케이프 공격은 컨테이너의 격리를 우회하여 호스트에 직접 접근하는 공격으로, 호스트를 대상으로 하는 공격이다. 컨테이너의 격리 수준이 가상머신에 비해 약하고 호스트의 커널을 공유한다는 취약점을 이용한다.

Table. 1. CVE by category and attack feature

Category	CVE	Image	Container Core
Container Escape	CVE-2022-39321	O	O
	CVE-2022-0185	X	O
	CVE-2019-5736	X	O
Privilege Escalation	CVE-2022-24769	X	O
	CVE-2022-36109	X	O
	CVE-2018-15664	X	O
Resource Exhaustion	CVE-2022-23471	X	O
	CVE-2019-9073	O	X
	CVE-2021-39939	O	X
Host File Access	CVE-2022-23648	X	O
	CVE-2021-25741	X	O
	CVE-2019-11246	X	O
Code Injection	CVE-2021-1560	X	O
	CVE-2023-26490	O	O
	CVE-2021-34079	O	X

CVE-2019-5736[2]의 경우, 호스트의 runC와 컨테이너 내부의 현재 실행 중인 프로세스를 가리키는 심볼릭 링크가 연결되어 있다는 취약점을 이용하여(Fig. 2) 호스트의 runC 바이너리 파일에 접근 후 파일을 공격자자 임의로 변조하여 컨테이너를 탈출하는 공격이다.

만약 위 공격이 성공하게 되면 호스트의 루트 권한을 얻거나, 특정 코드를 실행하여 외부에서 접근을 허용하는 등 광범위하고 치명적인 공격이 가능하다. 이처럼 컨테이너 이스케이프는 위협적이고 컨테이너에 특화된 공격이기에 각별한 주의가 필요하다.

3.2 권한 상승(privilege escalation)

권한 상승 공격은 컨테이너에 부여된 권한을 초과하여 호스트에 접근하는 공격이다. 컨테이너 이스케이프와 비슷하지만, 컨테이너 이스케이프는 격리 수준의 취약점을 노려 호스트에 직접 접근하는 방식이면 권한 상승은 컨테이너 내부에서 권한을 상승시켜 호스트에 접근한다는 차이점에서 별도로 분류하였다. 주로 애플리케이션, 명령어가 동작하기 위한 권한 획득의 취약점을 이용하여 공격한다.

CVE-2018-15664[3]는 'docker cp' 명령어의 파일 복사가 루트 권한으로 수행된다는 취약점

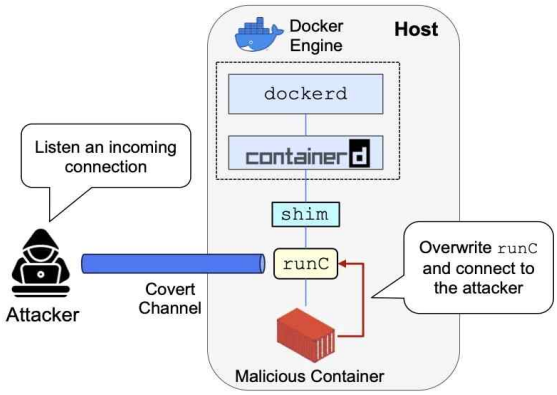

Fig. 2 CVE-2019-5736 attack scenario

과 심볼릭 링크를 이용하여 호스트에 루트 권한으로 파일에 접근하는 취약점이다.

3.3 자원 고갈(resource exhaustion)

자원 고갈 공격은 CPU, 메모리 등 호스트의 자원을 고갈시키는 공격이다. 해당 공격에 성공하면, 컨테이너가 사용할 자원이 부족하여 서비스 중인 다른 컨테이너의 서비스가 중단될 수 있다. 주로 자원 관리에 취약한 애플리케이션이나 기술의 취약점을 이용하여 공격한다.

CVE-2021-39939[4]는 깃랩 러너(GitLab Runner)에서 컨테이너의 로그 정보를 제한 없이 출력한다는 취약점을 이용하여 깃 랩 러너의 버퍼를 가득 채우는 공격 방식이다.

3.4 호스트 파일 접근(host file access)

호스트 파일 접근은 컨테이너 내부에서 인가되지 않은 호스트의 파일 시스템에 접근하는 공격이다. 컨테이너에서 호스트에 접근한다는 점은 컨테이너 이스케이프 권한 상승과 비슷하지만, 호스트의 권한을 뺏지 못하고 파일에만 접근한다는 점에서 별도 분류하였다. 주로 마운트의 취약점을 이용하여 공격이 이루어진다.

CVE-2022-23648[5]의 경우 표준 이미지 포맷 OCI의 JSON 파일을 수정한다. 컨테이너 볼륨을 호스트 파일로 수정하면 컨테이너로 호스트 파일이 마운트 되어 호스트 파일에 접근할 수 있게 된다.

3.5 코드 삽입(code injection)

코드 삽입 공격은 공격자가 임의의 코드를 애플리케이션, 시스템에 삽입하여 특정 동작을 유발하는 공격으로 공격이 성공하면 공격자의 코드를 실행할 수 있다. 기술 스택, 프로토콜의 취약점을 이용하는 경우가 많다.

CVE-2019-13139[5]는 'docker build' 명령어에서 경로를 깃 원격 URL로 지정할 경우, 깃 참조가 플래그로 잘못 해석되는 취약점을 이용하여 URL에 명령어를 주입하는 공격이다.

IV. 결론

본 논문에서는 컨테이너의 취약점 동향에 대해 조사 및 분석하였다. 이러한 취약점에 대해 NIST, CIS에서 관련 가이드를 제공하고 있는데, 결론에서는 이들을 참고하여 3가지 컨테이너 보안 가이드라인을 제시하고자 한다.

첫째, 컨테이너의 권한을 최소화한다. 루트 권한으로 컨테이너 생성, 필요 이상 파일 시스템 권한을 부여하는 것은 위험하다. 호스트 파일에 접근하거나, 이스케이프로 루트를 빼앗을 수 있기 때문이다. NIST에서는 호스트 파일 마운트를 권장하지 않는다. 따라서 제한된 권한으로 컨테이너 생성을 권장하고, 파일 시스템의 최소 권한을 유지하는 것을 권장하고 있다[6][7].

둘째, 도커와 이미지 버전을 항상 최신화한다. 본 논문에서 다룬 대부분의 취약점은 도커와 이미지의 최신 버전을 사용하면 통하지 않는다. CIS는 컨테이너 생성 명령어가 최신 이미지를 사용하는지 확인하라고 권장하는데, 이와 같이 버전 최신화는 컨테이너 보안에 매우 중요한 문제라고 할 수 있다[8].

셋째, 인증된 이미지를 사용한다. 본 논문의 컨테이너 이스케이프 예시는 악성 이미지 사용을 가정에서 이루어진 공격이다. 악성 이미지는 현재도 정상적인 이름으로 공유되고 있다. 이에, 도커 허브는 이미지 인증 방식을 사용하고 인증된 이미지 사용을 권장한다[9].

마지막으로, 도커 자체의 취약점 즉, 어떤 이미지를 기반으로 컨테이너를 생성하여도 취약점이 발생하는 경우가 존재하였으나 이 부분은 향후 연구에서 다루도록 한다.

[참고문헌]

[1] “악성 도커 이미지 피하는 3가지 방법,” AhnLab, https://www.ahnlab.com/kr/site/securityinfo/secunews/secuNewsView.do?seq=30533

[2] “CVE-2019-5736,” https://nvd.nist.gov/vuln/detail/CVE-2019-5736

[3] "CVE-2018-15664", https://nvd.nist.gov/vuln/detail/cve-2018-15664

[4] "CVE-2021-39939", https://nvd.nist.gov/vuln/detail/CVE-2021-39939

[5] "CVE-2022-23648", https://nvd.nist.gov/vuln/detail/CVE-2022-23648

[6] "Run the Docker daemon as non-root user", https://docs.docker.com/engine/security/rootless/

[7] Murugiah Souppaya, John Morello, Karen Scarfone, "NIST Application Container Security Guide," 800-190, Sep 2017 pp.27-28

[8] "CIS Benchmark, " v1.5.0 Des 2022 pp 221

[9] "Docker Verified Publisher," https://docs.docker.com/docker-hub/publish/