컨테이너 취약점에 따른 공격 벡터 식별 및 개선 연구 동향

김미연*, 최상훈[1], 박기웅†

세종대학교 SysCore Lab. (대학원생, 연구교수[1])

**세종대학교 정보보호학과 (교수†)

Trends in Identifying and Improvement Research Attack Vectors Related to Container Vulnerabilities

Mi-Yeon Kim*, Sang-Hoon Choi[1], Ki-Woong Park†

*SysCore Lab., Sejong University
(Graduate student*, Research professor[1])

**Dept. of Computer and Information Security, Sejong University

요 약

최근 클라우드 네이티브 기반 애플리케이션의 수요가 급증하고 있음에 따라, 이를 구현하기 위한 핵심 기술로 컨테이너가 주목받고 있다. 컨테이너는 가볍고 이식성이 좋지만, VM과 다르게 호스트 운영체제와 커널을 공유하므로 보안 위협에 더욱 취약하다는 한계가 있다. 이러한 한계에 대비하기 위해서는 컨테이너에서 발생할 수 있는 보안 위협에 대해 인지하고 있는 것이 중요하며, 조직 환경에 최적화된 솔루션 및 프레임워크를 적용하는 것이 필수적이다. 본 논문에서는 컨테이너 환경에서 발생할 수 있는 보안 위협과 이를 완화할 수 있는 연구 내용 및 한계를 분석함으로써 관련 동향을 살펴보고자 한다.

I. 서론

최근 클라우드 네이티브 기반 애플리케이션의 수요가 급증하고 있다 [1]. 클라우드 네이티브 기반 애플리케이션이란 클라우드 아키텍처를 위해 설계된 소프트웨어 접근법을 의미한다. 특히, 컨테이너는 가볍고 이식성이 좋아 클라우드 네이티브 애플리케이션을 구현하기 위한 가장 핵심적인 기술 중 하나로 자리하게 되었다 [2].

그러나 컨테이너는 VM(Virtual Machine)과 달리, 호스트 운영체제와 커널을 공유하므로 각종 보안 위협에 더욱 취약하다 [3]. 이는 컨테이너에서 보안 사고가 발생하게 되었을 때, 피해 정도가 훨씬 크고, 광범위할 수 있음을 의미한다. 이러한 문제에 대응하기 위해서는 컨테이너에서 발생할 수 있는 보안 위협에 대하여 인지하고 있는 것이 중요하다. 또한, 이를 완화할 수 있는 방안이나 연구의 장단점을 살펴, 조직에 최적화된 프레임워크와 솔루션을 적용하는 것이 필수적이다. 따라서 본 논문에서는 컨테이너 환경에서 발생할 수 있는 보안 위협과 이를 완화할 수 있는 연구 내용을 분석함으로써 관련 동향을 살펴보고자 한다.

본 논문의 구성은 다음과 같다. 2장에서 컨테이너 보안 위협에 대해 분석한 후, 3장에서 컨테이너 보안 위협을 개선하기 위한 연구와 한계점에 대하여 기술한다. 4장에서는 결론 및 향후 연구를 기술한다.

II. 컨테이너 보안 위협

2.1. 컨테이너 이미지 취약점

† 교신저자: 박기웅 (세종대학교 정보보호학과 교수)

본 논문은 과학기술정보통신부의 재원으로 실감콘텐츠핵심기술개발 (Project No. RS-2023-00228996, 40%), 국방ICT융합연구(Project No. 2022-11220701, 30%), 한국콘텐츠진흥원(KOCCA) 저작권기술 글로벌 인재 양성사업 (Project No. RS-2025-02221620, 30%)의 지원을 받아 수행된 연구임.

컨테이너 이미지의 유지 보수는 사용자가 담당하며, 관리 주기나 빈도에 대한 제약이 없기 때문에 한 번 제공된 후 오랜 기간 아무런 패치도 없이 방치될 수 있다 [4].

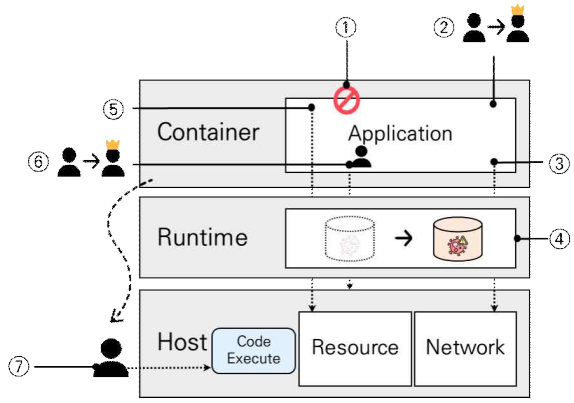
Shi et al. [5]은 실제로 최근 몇 년 동안 Docker 이미지를 대상으로 한 공격이 지속적으로 증가하고 있으며, 공격자는 공개된 이미지를 스캔하여 유출된 키나 소프트웨어 취약점을 악용하고 있다고 언급했다. 컨테이너 이미지는 Docker Hub와 같은 중앙 집중식 레지스트리에 의해 배포되는 경우가 많으며, 취약한 이미지는 급속도로 확산되어 공급망 공격의 형태로 발전할 수 있고, 많은 사용자의 컨테이너 보안을 약화시킬 수 있다.

2.2. 컨테이너 런타임 취약점

Reeves et al. [6]의 연구에 따르면, 컨테이너 이스케이프(Container Escape)의 주요 원인 중 하나는 컨테이너 런타임에 존재하는 취약점에 의한 것이다. 해당 연구에서는 분석 가능한 28개의 런타임 CVE를 분류하기 위해 <표 1>과 같이 7가지의 분류 체계를 정의하였다. 이때, 해당 연구에서 정의한 컨테이너 이스케이프는

<표 1> 컨테이너 런타임 취약점 유형 분류

분류	번호	항목	설명
non-escape	①	MAC Disabling	컨테이너 제한에 사용되는 MAC 프레임워크를 비활성화
	②	Container Privilege Escalation	컨테이너 내부에서 권한 상승
	③	Container to Host Network Access	컨테이너 내부에서 호스트 네트워크에 대한 접근 권한 획득
	④	Unpatched System	런타임 패키지 패치 과정에서 이전 취약점이 재포함
	⑤	Limited Container to Host Access	컨테이너 내부에서 일부 호스트 리소스에 접근
	⑥	Host Privilege Escalation	공격자가 호스트에 접근하여 루트 수준으로 권한 상승
escape	⑦	Container Escape Exploit	컨테이너 내부에서 탈출하여 호스트에 접근한 후, 코드를 실행



(그림 1) 컨테이너 아키텍처에 매핑된 런타임 취약점

호스트에 접근하여 코드를 실행하는 것으로 6개의 non-이스케이프와 이스케이프 취약점으로 다시 구분된다. 이를 컨테이너 아키텍처에 매핑하면, (그림 1)과 같이 표현할 수 있다. 연구 결과, 28개의 CVE 중 13개가 컨테이너 이스케이프로 이어지는 취약점이었다. 특히, 13개의 취약점은 9개의 PoC 이스케이프 익스플로잇과 연관되어 있었고, 가장 근본적인 이스케이프의 원인은 런타임 취약점을 통해 호스트의 구성 요소가 컨테이너 내부로 노출되었기 때문이었다.

Molleti et al. [7]은 외부 보안 위협이 런타임(컨테이너가 실행 중인 시점)에 발생할 수 있으며, 측면이동 공격이나 서비스 거부 공격(Denial-of-Service, DoS) 등과 같이 매우 다양한 유형으로 나타날 수 있다고 언급했다. 측면이동 공격은 네트워크 설정, 접근제어가 제대로 적용되지 않아 공격자가 취약한 네트워크를 통해 다른 컨테이너 서비스로 이동할 수 있는 것을 의미하고, 서비스 거부 공격은 리소스 과부하를 유발하여 서비스의 품질을 저하 및 중단시키는 것을 의미한다.

2.3. 컨테이너 오케스트레이션 취약점

Voievodin et al. [8]에 따르면, 컨테이너 오케스트레이션 시스템은 컨테이너를 효율적으로 관리할 수 있도록 하지만, 시스템을 더욱 복잡하게 만드는 요인이 되기도 한다. 특히, 대표적인 오케스트레이션 도구인 쿠버네티스는 10가지 이상의 추상화 계층을 지원하여 컨테이너가 세밀하게 관리되도록 하지만, 이를 설정하기 위한 과정이 복잡해지면서 구성 오류로 인한 시스템

오류나 보안 취약점이 발생할 수 있다. 또한, 컨테이너에 리소스 제한이 제대로 적용되어 있지 않은 경우, 특정 컨테이너는 노드의 리소스를 과도하게 사용할 수 있고, 이로 인해 동일 노드에 배포된 다른 컨테이너의 성능이 저하될 수 있다.

III. 개선 연구 및 한계점

3.1. 이미지 취약점 개선 연구

Kim et al. [9]에 따르면, 기존 이미지 스캐닝 방식은 운영체제 패키지나 패키지 관리자 기반의 취약점만 식별하므로 컨테이너 내부에 존재하는 취약점은 탐지하지 못한다. 해당 연구에서는 이러한 한계를 완화하기 위해 새로운 이미지 취약점 평가 시스템을 제안하였다. 제안된 시스템은 이미지 분해, 패키지 추출, 취약점 평가와 같은 세 가지 프로세스로 구성되며, 이미지 분해 단계에서 OCI 규격을 준수하는 이미지(여러 레이어로 구성)와 그렇지 않은 이미지(단일 레이어)가 포맷에 따라 분해된다. 이후, 패키지 추출 과정에서 분해된 이미지에 설치된 모든 패키지를 수집하여 취약점을 스캔한다. 이때, 검증된 베이스 이미지(Ubuntu, Cent OS 등)는 신뢰할 수 있는 것으로 간주하며, 베이스 이미지를 기준으로 추가 및 변경된 모든 패키지를 잠재적 취약점으로 판단한다. 마지막으로 취약점 평가 단계에서는 추출된 패키지 목록을 기반으로 컨테이너 이미지에 취약점이 존재하는지를 평가한다. 제안된 시스템에는 취약점을 평가하기 위한 CVE Binary Tool이 통합되었으며, CVE 데이터베이스를 참조하여 취약점을 탐지하였다. 또한, 식별된 취약점은 해시값을 기반으로 이미지나 레이어에 매핑되어 검증 데이터베이스에 기록된다. 이러한 데이터를 바탕으로 향후 동일한 이미지에 대한 요청이 들어왔을 때 데이터베이스에 저장된 결과가 반환되게 함으로써 효율성을 극대화하였다.

그러나 제안된 시스템은 기존 도구가 탐지할 수 있었던 맥락 정보(e.g. 패키지 관리자 메타데이터)에 의한 취약점을 식별하는 데는 한계가 있었다. 효과적인 탐지를 위해서는 제안된 시스템과 기존의 도구를 상호보완적으로 사용하는 것이 중요하며, 해당 연구에서도 Anchore와 같은 기존 솔루션과 제안된 시스템을 결합한다면, 더욱 광범위한 취약점을 탐지할 수 있을 것으로 기대했다.

3.2. 런타임 취약점 개선 연구

Alvayadi et al. [10]은 TCP Flood와 UDP Flood 기법을 이용한 DoS 공격 상황에서 runc, gVisor, Kata containers의 성능과 격리성을 분석하였다. 이때, gVisor는 Google이 제안한 샌드박스형 보안 런타임이고, Kata containers는 오픈스택 재단에서 제안한 초경량 VM 런타임이다. 해당 연구에서는 각 런타임으로 실행된 nginx 서버의 HTTP 성능과 DoS 공격이 발생했을 때의 HTTP 성능을 측정하였다. 제시된 결과에 따르면, 웹 응답 시간 결과에서 HTTP 성능만 측정했을 때는 runc와 gVisor가 비슷한 성능을 보였지만, DoS 시나리오가 적용되었을 때의 성능은 *runc>gVisor>Kata Containers* 순으로 runc의 성능이 가장 좋았다. 그러나 TCP flood 상황에서 runc는 평균 0.20회, gVisor는 평균 2.00회 정도의 패킷 전송 실패가 발생했으고, UDP flood에서 gVisor는 평균 37.20회의 패킷 전송 실패가 발생한 것으로 나타났다. 반면, Kata Containers는 TCP/UDP flood 시나리오에서 패킷 전송 실패 사례가 단 한 건도 발생하지 않았다.

이러한 연구 결과는 컨테이너 보안 런타임이 특정 공격에서는 runc보다 취약할 수 있음을 의미한다. 또한, 추가 보안 계층을 통해 컨테이너 이스케이프와 같은 공격을 높은 확률로 완화할 수 있지만, 성능 저하나 오버헤드를 감수해야 한다는 한계가 있다.

3.3. 오케스트레이션 취약점 개선 연구

Sun et al. [11]은 컨테이너 오케스트레이션에서 발생할 수 있는 설정 오류 사례를 분석하여 패턴을 구축하였다. 또한, 이를 기반으로 오케스트레이션 구성 오류를 탐지하기 위한 하이브리드 방식의 KubeChecker를 제안하였다. kubechecker는 구성 항목과 런타임 데이터를 활용하

여 취약점을 식별하고, 오탐을 최소화하기 위해 자동화된 결함을 주입하여 식별된 취약점이 실제 장애로 이어질 수 있는지를 검증한다. 해당 연구에서는 kubechecker의 성능을 검증하기 위해 다섯 개의 오픈소스 마이크로서비스 애플리케이션에서 기존 도구와의 비교를 수행하였으며, 분석 결과에 따르면, kubechecker는 기존 도구들에 비해 광범위한 범위의 취약점을 식별할 수 있는 것으로 나타났다.

이러한 연구가 실제 클라우드 환경에 적용된다면, 더욱 강화된 컨테이너 보안을 실현할 수 있지만, 조직의 특성이나 환경에 따라 적용이 불가할 수도 있다. 또한, Sun et al.의 연구에서 제시한 것과 같이 실제 운영 환경에서 명시하고 있는 정책적 제한에 따라 개발된 도구나 접근 방식이 의도한 바와는 다르게 동작할 수 있다는 한계가 있다.

IV. 결론 및 향후 연구

본 논문에서는 클라우드 네이티브 기반 애플리케이션의 수요가 급증함에 따라 컨테이너의 보안 위협과 개선 연구 및 한계에 대한 분석을 수행하였다. 분석 결과에 따르면, 컨테이너는 이식성, 효율성, 경량화된 격리 기능을 제공하여 호스트의 환경과는 상관없이 가볍고 안정적인 서비스를 제공한다. 그러나 호스트 운영체제와 커널을 공유하므로 VM과 비교했을 때, 각종 보안 위협에 더욱 취약할 수 있다. 특히, 컨테이너 취약점에는 이미지, 런타임, 오케스트레이션 설정 오류 등과 같은 공격 벡터가 존재하였다. 또한, 컨테이너의 단점을 완화하기 위해 다양한 연구가 수행되고 있지만, 여전히 해결되어야 할 도전과제가 있음을 확인하였다.

향후 연구에서는 보다 광범위한 보안 위협과 개선 연구를 체계적으로 분석하여 도출된 도전과제를 해결하기 위한 통찰을 제공하고자 한다.

[참고문헌]

[1] Grand View Research, Cloud Native Applications Market (2024 - 2030)

[2] V.U. Ugwueze, "Cloud native application development: Best practices and challenges," International Journal of Research Publication and Reviews, vol. 5, no. 12, pp. 2399-2412, Dec. 2024.

[3] Nada. Barnawi, R. AITooq and M. Almukaynizi, "Mitigating Container Escape Threats Through Effective Countermeasures: A Survey," Proceedings of the 10th World Congress on Electrical Engineering and Computer Systems and Sciences (EECSS'24), Paper No. CIST 164, Aug. 2024.

[4] R. Malhotra, A. Bansal and M. Kessentini, "Vulnerability analysis of docker hub official images and verified images," 2023 IEEE International Conference on Service-Oriented System Engineering (SOSE). IEEE, pp. 150-155, Jul. 2023.

[5] H. Shi, L. Ying, L. Chen, H. Duan, M. Liu and Z. Xue, "Dr. Docker: A Large-Scale Security Measurement of Docker Image Ecosystem," Proceedings of the ACM on Web Conference 2025, pp. 2813-2823, Apr-May. 2025.

[6] 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). IEEE, pp. 38-46, Oct. 2021.

[7] R. Molleti, "Container runtime security detection and prevention techniques," World J. Adv. Res. Reviews, vol. 24, no. 3, pp. 2626-2639, Dec. 2024.

[8] Y. Voievodin and I. Rozlomii, "Application security optimization in container orchestration systems through strategic scheduler decisions," Proceedings of the CPITS-2024: Cybersecurity Providing in Information and Telecommunication Systems, CEUR Workshop Proceedings. Vol. 3654, pp. 471-478, Feb. 2024.

[9] Y. Kim, C. Park and D. Hong, "A Robust Framework for Comprehensive Container Image Vulnerability Assessment," IEEE Access (2025), vol. 13, pp. 25837-35847, Feb. 2025.

[10] A.K.S. Alvayadi, and M. Pranata. "Analysis Performance of Container Runtime RunC, gVisor, and Kata Containers Against Denial Of Services Attacks," 2025 International Conference on Smart Computing, IoT and Machine Learning (SIML). IEEE, pp. 1-6, Jun. 2025.

[11] Y. Sun, D. Lyu, C. Cui and H. Xu, "KubeChecker: Detecting Configuration Bugs in Container Orchestration," 2025 55th Annual IEEE/IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S). IEEE, pp. 91-97, Jun, 2025.