다형성 도커 이미지 공격에 강인한 계층적 취약점 탐지 기법

류정화[1], 이일구[2]
[1]성신여자대학교 미래융합기술공학과 석사과정
[2]성신여자대학교 미래융합기술공학과 교수
220236039@sungshin.ac.kr, iglee@sungshin.ac.kr

Hierarchical vulnerability detection technique robust against polymorphic Docker image attacks

Jung-Hwa Ryu[1], Il-Gu Lee[2]
[1,2]Dept. of Future Convergence Technology Engineering, Sungshin University

요 약

최근 클라우드가 전 산업에 도입되면서 클라우드 네이티브 환경에 관한 관심이 증가하고 있다. 클라우드 서비스 개발자는 도커 (Docker) 이미지를 활용하여 개발 환경을 구축하고 배포한다. 그러나 종래의 이미지 스캐닝 도구들은 해시값 기반의 시그니처 탐지 방법론을 사용하기 때문에 제로데이 취약점을 탐지하지 못하거나, 이미 저장된 CVE DB에 있는 취약점만 탐지할 수 있었다. 본 논문은 도커 이미지의 계층성을 활용하여 다형성 도커 이미지 공격을 탐지할 수 있는 기법을 제안한다. 실험 결과에 따르면 제안한 방법은 종래 방법 대비 다형성 도커 이미지 공격 탐지율을 28.6% 개선할 수 있었다.

1. 서론

최근 클라우드 환경으로의 전환이 가속화됨에 따라, 클라우드 네이티브 접근 방식을 도입하는 기업체와 개발자들이 늘어나고 있다. 클라우드 네이티브 방식은 어플리케이션을 설계, 구축, 운영하는 현대적인 접근 방식으로, 어플리케이션의 고가용성, 확장성 및 운영 효율성을 극대화한다. 이러한 수요에 대응하여, 클라우드 생태계는 도커 허브라는 클라우드 기반의 컨테이너 라이브러리를 제공한다[1]. 도커 허브에서 사용자는 도커 이미지를 자유롭게 검색하고 다운로드하거나 공개 라이브러리 및 개인 저장소에 자신이 제작한 이미지를 자유롭게 업로드할 수 있다. 이처럼 도커 이미지는 종래 어플리케이션과 그 종속성이 컨테이너 내에서 패키징 되어, 다양한 컴퓨팅 환경에서도 안정적으로 실행될 수 있는 단위로 기능한다. 그러나 악의적인 사용자들이 사용자가 신뢰할 수 있는 프로젝트로 이름을 위장한 후 사용자의 악성 도커 이미지 다운로드를 유도하는 typosquatting 공격이 만연한 상황이다[2]. 또한, sysdig의 조사 결과에 따르면, 도커 허브에 신뢰할 수 없는 250,000개 이상의 이미지가 배포되어 있으며 1,652개의 이미지가 악성 이미지로 식별되었다고 보고했다[3]. 608개의 컨테이너 이미지에서 발견된 cryptomining 관련 취약점이 가장 큰 비중을 차지했으며, 이외에도 Proxy avoidance, Malicious websites, Hacking, Dynamic DNS 등 도커 이미지를 악성 행위의 수단으로 악용하는 공격이 증가하고 있다. 그러나 이에 대하여 도커 허브 커뮤니티는 해당 이미지에 대한 해시값 형태의 이미지 다이제스트 정보만 제공하고 있을 뿐, 세부 레이어들에 대한 보안 관점에서의 분석은 제공하지 않는다. 본 논문에서는 이러한 종래 연구의 한계점을 개선하기 위해 다형성 도커 이미지 공격에 강인한 계층적 취약점 탐지 기법을 제안한다. 제안하는 방법은 변형 조건에 따라 종래에 탐지하지 못했던 취약점들을 탐지 및 식별한다.

본 논문의 구성은 다음과 같다. 2장에서는 도커 이미지 레이어의 구조와 종래의 정적 분석 탐지 방법론들의 한계점을 분석하고, 3장에서는 변형 조건에 따른 악성 이미지 탐지 성능을 평가하고 분석한다. 4장에서는 결론을 맺는다.

<표 1> 도커 이미지 취약점 탐지 도구 비교 분석표

Tool	Scanning Method	Key Features
Docker Scout [4]	시그니처 기반	• 시각화 기능: 도커 허브의 이미지를 시각화하여 사용자가 이미지 내부 구조를 쉽게 이해할 수 있도록 도움
Clair [5]	시그니처 기반	• 정적 분석 기능: 컨테이너의 취약점을 정적으로 분석하는 오픈 소스 프로젝트
Trivy [6]	시그니처 기반 및 머신러닝 기반	• 취약점 및 구성 오류 검사, GitHub Actions을 통해 자동화된 검사 가능
Anchore [7]	시그니처 기반 및 SBOM 기반	• SBOM 지원: 소프트웨어 구성 요소 목록(SBOM)을 이용한 분석을 제공하여 보안 강화
Virus Total [8]	시그니처 기반 및 행위 기반	• 멀티 엔진 검사: 여러 안티바이러스 엔진을 사용하여 파일을 검사하여 탐지율 향상

2. 관련 연구

로컬 환경에서 사용자는 코드 형태의 Dockerfile을 구현하고, Dockerfile을 빌드하면 여러 레이어로 구성된 하나의 이미지가 된다. 도커 이미지는 하나의 베이스 이미지를 기반으로 코드 라인마다 레이어들을 쌓아 올리는 구조를 갖는다. 표 1은 계층화된 도커 이미지에 대한 보안 취약점을 발견하고 탐지하기 위해, 정적·동적 분석하기 위한 이미지 스캐닝 도구들을 보여준다.

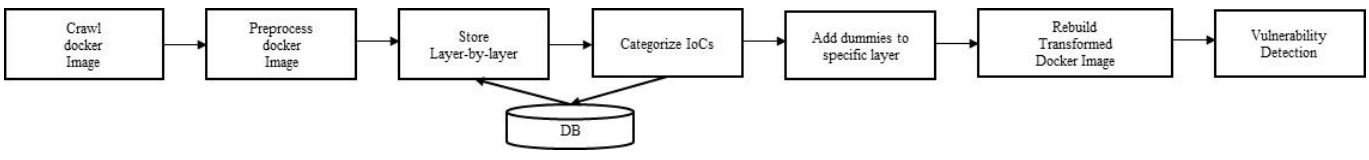
도커 이미지 환경에서의 취약점 탐지 및 분석 방법은 크게 4가지로 나누어 볼 수 있다. 시그니처 기반 분석 기법은 알려진 위협의 특정 패턴이나 시그니처를 이용하여 시스템의 네트워크 트래픽이나 파일을 검사한 후, 악성 시그니처를 탐지한다. 머신러닝 기반 분석 기법은 데이터 학습하여 새로운 위협을 탐지하며, 이를 통해 종래 시그니처 기반 분석 대비 탐지 커버리지를 개선한다. SBOM (Software Bill of Materials) 기반 분석 기법은 소프트웨어 구성 요소 목록을 사용하여 주로 소프트웨어의 보안과 공급망 위험 관리에 활용이 되는 방법이다. SBOM 기반 분석 기법은 코드베이스의 완전한 인벤토리를 통해 소프트웨어의 모든 구성 요소를 정확히 파악할 수 있다. 행위 기반 분석 기법은 시스템의 행위 패턴을 모니터링하여 이상 행위를 탐지하므로 알려지지 않은 위협에 대응할 수 있다.

최근 도커 이미지의 취약점을 진단하고 분석하는 방법론에 관한 연구가 활발하게 진행되었다[9.10]. DIVDS (Docker Image Vulnerability Diagnostic System)는 도커 이미지를 업로드하거나 다운로드하는 과정에서 자동으로 취약점을 진단하는 시스템이다. 이 시스템은 도커 환경의 신뢰성을 높이기 위해 구축되었으며, 취약점을 효과적으로 식별하고 대응할 수 있도록 돕는다. 그러나 DIVDS는 파일 실행 시 발생할 수 있는 취약점을 감지하지 못하는 한계점이 존재했다. DAVS (Dockerfile Analysis for Container Image Vulnerability Scanning)는 Dockerfile을 기반으로 정적 분석을 수행하여 잠재적으로 취약한 파일(Potentially Vulnerable Files, PVFs)의 목록을 추출하는 프레임워크이다. 이 프레임워크는 OCI(Oracle Cloud infra structure) 기반 컨테이너 이미지의 취약점을 파악하도록 설계되었다. 그러나 해당 프레임워크는 Dockerfile 정보에만 의존함에 따라 Dockerfile이 없거나 불완전할 경우 분석의 정확성이 저하되는 한계점이 존재했다.

3. 제안 모델

본 논문에서는 해시값, 마이너 특성, 계층화 정보를 고려한 Docker stage를 제안한다. Docker stage의 구성은 다음의 그림 1과 같다. 제안하는 탐지 시스템의 취약점 탐지 성능을 평가하기 위해 본 연구에서는 알려진 악성 도커 이미지인 pumevnezdiroorg/drupal 이미지에 대한 변형 조건을 정의한 후, 특정 레이어 정보를 변형하여 다형성 도커 이미지 공격을 수행하는 변형된 악성 이미지를 재생성했다.

(그림 1) Docker stage 구성도



<표 2> pumevnezdiroorg/drupal의 이미지 레이어 구조

Status	Hierarchical structure	Layer name
Malicious	Layer 10	9cb1510b3ce4ee4edee6fda77fd668b7752a9785bdeb12ee1bcedc90c3b31183
Malicious	Layer 9	3b112d77a3e4ccdc8717e85e2d06850b6f21352f88245fd10676c369a0b29592
Malicious	Layer 8	09e025918b6213720424226ffb3aaa5940e3597102272c4f1f348a3204bbf35f
Benign	Layer 7	7c5568bfd897a7c9c427451f29ca936566fb5222c7391bea70a4b67f337d87cb
Benign	Layer 6	fa51027a34ac08a4dac97010ef1caadae11dc0f2f4f3bf6f03c9c3c132485bce
Benign	Layer 5	33f1a94ed7fcbcd44e2ccf330729601e204214f34384cba790a99efd4bb29752
Benign	Layer 4	b27287a6dbcef78f4855ab2d9e8dd4c21b53fcf4af77af9c2ecb08725ae02052
Benign	Layer 3	47c2386f248c7b10217d7bf25730b1bdb02b276760f75e0af11d28ddd350b53b
Benign	Layer 2	2be95f0d8a0c6cc1665ca311e240b9f6dfc824db12a6df3136bc90e0fc33eea3
Benign	Layer 1	2df9b8def18a090592bf1cbd1079e1ac2274435c53f027ee5ce0a8faaa5d6d4b
Benign	Base Image	fdd31a96efd4322d59ca0dd52383a1ed41391654a198f365113b710665791621

표 2에서와 같이 pumevnezdiroorg/drupal 이미지는 한 개의 베이스 이미지가 3개의 malicious한 레이어와 7개의 benign한 레이어로 구성된다. 각 레이어에 포함된 정보 중, 취약점을 내포한 IoC(Indicator of Compromise) 정보는 표 3과 같다. 본 연구에서는 수집한 IoC 정보들을 행위 정보인 Behavior, 네트워크 정보인 Domain/IP, 실행 파일 유형의 쉘 스크립트와 같은 세 가지 유형으로 나누어 분류했다.

<표 3> 취약점을 내포한 레이어의 IoC 정의

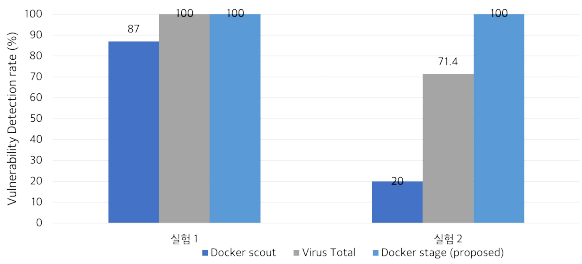
Analysis Type	Description	Example
Behavior	File Addition and Permission Modification	ADD file:6086fe1466d461e3d763ca57e1e45bd5c35d1a87a8b1a7ca6b9085c795c13dd6 in /bin/minerd
		chmod 755 /bin/minerd
Domain / IP address	Malicious Network Activity	ENTRYPOINT ["/bin/minerd" "-a" "cryptonight" "-o" "stratum+tcp://xmr.pool.minergate.com:45560" "-u" "trudly@outlook.com" "-p" "x" "-t“ “1”
		git clone https://github.com/OhGodAPet/cpuminer-multi
Shell script	Script Actions: modification of system settings, ordownloading additional payloads.	9cb1510b3ce4ee4edee6fda77fd668b7752a9785bdeb12ee1bcedc90c3b31183

이후 저장된 레이어의 악성 여부를 기준으로, 두 차례의 실험을 수행했다. 첫 번째 실험에서는 악성 레이어에 해당하는 레이어 8 ~ 레이어 10의 유무에 따른 8가지 경우를 정의하고, 각 조건에 따라 이미지를 재가공했다. 레이어 8 ~ 레이어 10에 대하여, 레이어 8만 가진 경우를 (1,0,0), 레이어 8과 10을 가진 경우를 (1,0,1)로 정의할 때, 총 경우의 수는 $2 \times 2 \times 2 = 8$로 확인할 수 있었다. 두 번째 실험에서는 8가지의 경우 각각에 대한 10가지 유형의 다형성 조건을 정의한 후, 하위 5개의 변형 샘플들을 생성하여 총 400개의 변형된 이미지를 재생성했다. 이 때의 더미 조건은 표 4와 같다. 표 4에서 정의된 변형 조건에 따른 종래 취약점 탐지 도구와 제안 시스템의 취약점 탐지 성능은 그림 2를 통해 확인할 수 있다.

첫 번째 실험에서는 시그니처 기반의 Docker scout의 탐지율이 87%이고, 시그니처 및 행위 기반 탐지 시스템인 Virus Total과 제안 시스템의 탐지율은 100%였다. 두 번째 실험에서는 각 8가지 변형 조건을 기반으로 400개의 이미지를 재생성한 후 탐지율을 평가했다. 시그니처 기반의 Docker scout는 종래 악성 레이어를 내포한 이미지 취약점을 탐지했으나, 변형된 악성 레이어를 내포한 이미지의 취약점을 탐지하지 못하여 20%의 탐지율을 보였다. Virus Total의 경우 일부 변형된 악성 레이어를 탐지하지 못해서 탐지율이 71.4%였다. 그러나 제안한 방법을 적용하면 각 변형 조건에 따른 악성 이미지에 대해서도 탐지율이 100%였다. 이를 통해 다형성 도커 이미지에 대한 취약점 탐지 성능이 28.6% 개선되었고, 다형성 도커 이미지 공격에 대한 효율적인 탐지를 위해서는 도커 이미지의 계층 구조 특징을 고려한 탐지 시스템이 필요하다는 것을 확인할 수 있었다.

<표 4> 다형성 도커 이미지 생성을 위한 변형조건

Type of Dummies	Example	Number of Cases (Exp 1)	Number of Dummy Samples	Number of Image Samples
Add Dummy layer	FROM ubuntu:20.04 COPY --from=builder /dummyfile /dummyfile RUN rm /dummyfile	8	5	40
Inject Dummy layer	RUN dd if=/dev/urandom of=/dummyfile bs=1M count=10 && rm /dummyfile	8	5	40
Add randomized environmental variable	ARG RANDOM_VAL ENV DUMMY_VAR=${RANDOM_VAL}	8	5	40
Add conditional layer	#!/bin/bash if [$(($RANDOM % 2)) -eq 0]; then echo "RUN echo Dummy layer" >> Dockerfile fi	8	5	40
Execute dummy command	RUN echo "Random text $RANDOM"	8	5	40
Layer squashing	docker squash	8	5	40
Random file system change	RUN mkdir /tmp/dummy-${RANDOM} && rmdir /tmp/dummy-${RANDOM}	8	5	40
Dummy network operation	RUN wget https://example.com/largefile.zip -O /tmp/dummy.zip && rm /tmp/dummy.zip	8	5	40
Change variable workdir	WORKDIR /tmp/${RANDOM}_dummy WORKDIR /	8	5	40



(그림 2) 종래 이미지 스캐닝 도구들과 제안 시스템의 취약점 탐지 성능 비교

4. 결론 및 향후 연구

본 연구는 다형성 도커 이미지 공격 상황에서도 취약점을 효과적으로 탐지하는 방법을 제안했다. 제안된 탐지 시스템은 종래 시스템 대비 28.6% 개선된 탐지 성능을 보였으며, 특히 변형 조건에 따른 우회 시도에 대해서도 탐지율이 높았다. 향후 연구에서는 도커 이미지에 대한 변형 조건의 파급력을 분석하여, 공격자의 우회 시도에 따른 대응 방안들을 분류할 예정이다. 또한, 도커 허브에 공개된 전체 도커 이미지들을 대상으로 제안 시스템을 적용하여, 종래에 탐지하지 못한 제로데이 취약점들을 식별하고자 한다.

Acknowledgement

본 논문은 2024년도 산업통상자원부 및 한국산업기술진흥원의 산업혁신인재성장지원사업 (RS-2024-00415520)과 과학기술정보통신부 및 정보통신기획평가원의 ICT혁신인재4.0 사업의 연구결과로 수행되었음 (No. IITP-2022-RS-2022-00156310)

참고문헌

[1] Docker Hub. https://hub.docker.com.
[2] G Liu, X Gao, H Wang, K Sun, “Exploring the Unchartered Space of Container Registry Typosquatting”, USENIX Security Symposium, Boston, 2022
[3] Sysdig, “Analysis on Docker Hub malicious images: Attacks through public container images”, https://sysdig.com/blog/analysis-of-supply-chain-attacks-through-public-docker-images.
[4] Docker Scout, https://docs.docker.com/scout.
[5] Clair, https://github.com/quay/clair.
[6] Trivy, https://trivy.dev.
[7] Anchore, https://anchore.com/
[8] Virus Total, https://www.virustotal.com/gui/home/upload
[9] S. Kwon and J. -H. Lee, "DIVDS: Docker Image Vulnerability Diagnostic System," in IEEE Access, vol. 8, pp. 42666-42673, 2020.
[10] Doan, T. P. Jung, S. "DAVS: Dockerfile Analysis for Container Image Vulnerability Scanning." Computers, Materials & Continua 72(1) 2022.