도커 이미지 라이선스 컴플라이언스 위반 방지 시스템

권순홍[1], 손우영[2], 이종혁[3]
[1]세종대학교 정보보호학과 & 지능형드론 융합전공 박사과정
[2]세종대학교 프토로콜공학연구실 학부과정
[3]세종대학교 정보보호학과 & 지능형드론 융합전공 교수
soonhong@pel.sejong.ac.kr, wooyoung@pel.sejong.ac.kr, jonghyouk@sejong.ac.kr

System for Preventing License Compliance Violations in Docker Images

Soonhong Kwon[1], Wooyoung Son[2], Jong-Hyouk Lee[3]
[1]Dept. of Computer and Information Security & Convergence Engineering for Intelligent Drone, Sejong University
[2]Protocol Engineering Lab., Sejong University
[3]Dept. of Computer and Information Security & Convergence Engineering for Intelligent Drone, Sejong University

요 약

2013년 도커가 등장한 이후, 컨테이너 기술을 기반으로 한 프로젝트 및 사업이 지속적으로 활성화되고 있는 추세이다. 도커 컨테이너는 커널을 포함하고 있지 않음에 따라 기존 가상머신에 비해 경량화된 형태로 애플리케이션을 프로비저닝하는데 활용될 수 있다. 또한, 도커에서는 퍼블릭 도커 이미지 레포지토리인 Docker Hub를 통해 개발된 도커 이미지가 공유 및 배포될 수 있도록 하여 개발자들이 자신의 목적에 부합하는 서비스를 구축하는데 많은 도움을 주고 있다. 최근에는 클라우드 네이티브 환경에 대한 수요가 증가하면서 컨테이너 기술이 더욱 각광받고 있는 실정이다. 이에 따라 도커 이미지 및 이를 기반으로 한 도커 컨테이너 환경에 대한 보안을 위한 연구/개발은 다수 이루어지고 있으나, 도커 이미지 라이선스 컴플라이언스 이슈에 대한 논의 및 민감 데이터 보호 방안에 대한 연구/개발은 부재한 상황이다. 이에 본 논문에서는 도커 이미지 라이선스 컴플라이언스 위반 방지 시스템을 제안하여 도커 이미지 업로드시, Docker Hub 내 도커 이미지와 유사도 검사를 수행할 수 있는 방안을 제시하고자 하며, 도커 이미지 내 민감 데이터를 식별하고 이를 보안할 수 있는 방안에 대해 제시하여 신뢰할 수 있는 도커 컨테이너 공급망을 구축할 수 있음을 보인다.

1. 서론

2013년 도커가 등장한 이후, 컨테이너 기술을 기반으로 한 프로젝트 및 사업이 지속적으로 활성화되고 있는 추세이다. 2024년 4월 Business Research Insights사에서 발표된 바에 따르면 '글로벌 컨테이너 기술 시장 규모는 2021년 4억 9,640만 달러로 평가되었으며, 2023년에는 312,342만 달러에 이를 것'으로 예상한 바 있다 [1]. 이와 같은 컨테이너 기술이 지속적으로 각광받고 있는 이유는 도커 컨테이너의 특성에 기반한다. 도커 컨테이너는 커널을 포함하고 있지 않음에 따라 기존 가상머신에 비해 경량화된 형태로 애플리케이션을 프로비저닝하는데 활용될 수 있다. 또한, 도커에서는 퍼블릭 도커 이미지 레포지토리인 Docker Hub를 통해 개발된 도커 이미지가 공유 및 배포될 수 있도록 하여 개발자들이 자신의 목적에 부합하는 서비스를 구축하는데 많은 도움을 주고 있다. 최근에는 클라우드 네이티브 환경에 대한 수요가 증가하면서 컨테이너 기술이 더욱 각광받고 있는 실정이며, 국내에서는 범정부차원에서 공공·행정기관의 정보시스템을 클라우드 네이티브로 전환하려고 하는 시도가 이루어지고 있다.

이에 도커 이미지 및 이를 기반으로 한 도커 컨테이너 환경 보안을 위한 연구/개발은 '도커 이미지 취약점 스캐닝', '도커 컨테이너 침입 탐지 시스템' 등의 연구에 국한되어 있는 실정이다. 하지만, 도커 컨테이너를 타겟으로 하고 있는 대부분의 고도화된 공격들은 도커 이미지를 구성하기 위해 필요한 Dockerfile 내 저장되어 있는 SSH(Secure Shell) 키 혹은 API(Application Programming Interface) 키 등을 활용하여 이루어진다.

이와 더불어 도커 플랫폼의 특성상 Docker Hub에 공

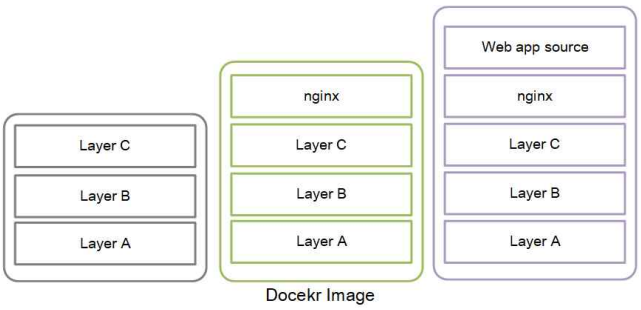
유 및 배포되어 있는 도커 이미지를 활용하여 자신이 구성하고자 하는 서비스를 구축하여 이를 도커 이미지 형태로 재배포하였을 때, 라이선스 이슈가 발생할 수 있다.

이에 본 논문에서는 개발자가 자신의 도커 이미지를 생성하여 이를 Docker Hub와 같은 도커 이미지 레포지토리에 업로드할 시, 도커 이미지 개발을 위해 활용된 도커 이미지와의 유사성을 비교하여 독창성을 검증하고, 검증된 도커 이미지에 대해서는 민감 데이터 여부를 식별하고 이를 암호화된 형태로 저장하여 도커 이미지 레포지토리에 업로드될 수 있는 시스템을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 도커 이미지 배포와 관련하여 주요 논쟁 사례에 대해 분석한 내용을 설명하며, 3장에서는 본 논문에서 제안하는 시스템인 도커 이미지 라이선스 컴프라이언스 위반 방지 시스템에 대해 설명한다. 4장에서는 본 논문의 결론을 맺는다.

2. 도커 이미지 배포 관련 라이선스 이슈 분석

도커 이미지 배포와 관련된 이슈는 2021년 4월 독일의 오픈소스 관련 변호를 담당하고 있는 변호사인 Till Jaeger에 의해 제기된 바 있다 [2]. 오픈소스는 우리가 목적으로 하는 서비스 혹은 애플리케이션을 개발함에 있어 기반이 되는 소스코드 혹은 바이너리 파일을 제공하고 있지만, 이에 대한 라이선스는 분명하게 명시가 되고 있다. 특히, 각 라이선스 별로 '복제, 배포, 수정의 권한 허용', '배포 시 라이선스 사본 첨부', '저작권 고지사항 또는 Attribution 고지사항 유지', '배포시 소스코드 제공의무와 범위', '조합저작물 작성 및 타 라이선스 배포허용', '수정 내용 고지', '명시적 특허라이선스의 허용', '라이선시가 특허소송 제기시 라이선스 종료', '이름, 상표, 상호에 대한 사용제한', '보증의 부인', '책임의 제한'에 대한 내용을 포함하고 있음에 따라 각 라이선스 별 고지 사항에 대해서는 내용을 파악하여 준수해야 한다.



(그림 1) 도커 레이어 개요

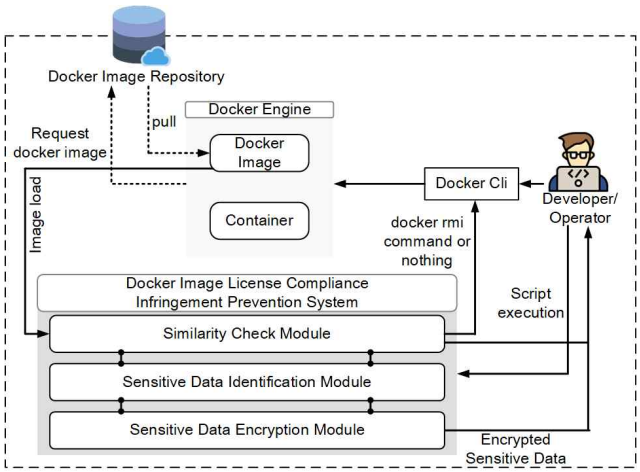
하지만, 도커 이미지의 특성상 (그림 1)과 같은 레이어 구조로 구성되어 있음에 따라 이에 대한 사항을 일일이 확인하기에는 어려움이 있다 [3]. 또한, 현재 Docker Hub 내에서는 각 도커 이미지의 Dockerfile을 보안 상의 이유로 이를 확인할 수 없도록 하고 있음에 따라 기반이 되는 도커 이미지에 대한 내용도 확인이 어렵다. Docker Hub 내에서는 배포된 도커 이미지와 업로드가 시도되고 있는 도커 이미지의 유사성에 대해 확인하고 있지 않음에 따라 저작권 위반 사례가 발생하였을 시, 이에 대한 책임을 누구에게 물을 것인지에 대한 논쟁이 이어지고 있다.

이는 FOSS(Free and Open-Source Software) 라이선스 컴플라이언스에 새로운 의문을 제기되도록 하였으며, 이에 대한 주요 내용은 도커 이미지의 특성상 도커 이미지 혹은 Dockerfile의 형태로 분산형 배포가 가능함에 따라 이에 대한 라이선스 컴플라이언스를 누가 책임을 져야 하는지에 대한 문제이다.

FOSS 라이선스의 경우, 라이선스에 대한 준수 의무를 '배포'의 개념과 주로 연결되어 있고, 이에 대한 범위가 각 국가의 저작권법에 의거하여 차이가 존재한다. 이에 대해 현재까지 논의된 사항에 대해 분석된 내용에 따르면 Dockerfile을 제공하는 개발자의 경우, Dockerfile을 기반으로 도커 이미지를 빌드 할 시, 라이선스 컴플라이언스에 대한 책임이 부여되며, 이는 Docker Hub와 같은 외부 레포지토리도 공동으로 책임을 져야한다는 것이다.

이에 따라 개발자가 자신의 환경에서 도커 이미지를 구성하여 Docker Hub와 같은 외부 레포지토리에 업로드 할 시, 외부 레포지토리 내에 배포되어 있는 도커 이미지와 유사성 비교/검증을 통해 자신이 개발한 도커 이미지가 독창성을 지닌 도커 이미지임을 외부 레포지토리와 공동으로 책임을 져야할 필요가 있다.

3. 제안하는 시스템



(그림 2) 도커 이미지 라이선스 컴플라이언스 위반 방지 시스템

본 논문에서 제안하는 도커 이미지 라이선스 컴플라이언스 위반 방지 시스템은 개발자와 Docker Hub와 같은 외부 레포지토리 간 책임져야 하는 도커 이미지 FOSS 라이선스 문제를 해결하는 동시에 도커 이미지에 포함되어 있는 민감 데이터를 식별하고, 이를 암호화하여 악의적인 공격자가 민감 데이터를 기반으로 공격을 수행할 수 있는 가능성을 줄이고자 하였다. (그림 2)는 본 논문에서 제안하는 시스템인 도커 이미지 라이선스 컴플라이언스 위반 방지 시스템을 보여준다.

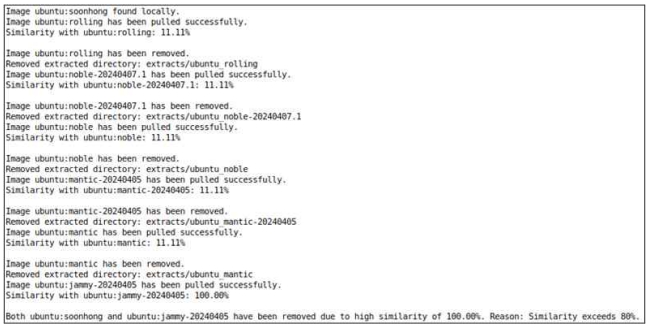
(그림 2)를 통해 확인할 수 있듯이 도커 이미지 라이선스 컴프라이언스 위반 방지 시스템은 세부적으로 도커 이미지의 유사성을 확인하는 모듈과 민감한 데이터를 식별하고, 이를 암호화하는 모듈로 구성되어 있다.

Algorithm 1 Similarity Check Module

```
1  begin
2  input local_image_name, tag
3  output_path ← 'scan_results.txt'
4  local_hashes ← pull and extract hashes for local_image_name
5  hub_images ← fetch images with tag from Docker Hub
6  for each image_name in hub_images do
7      hub_hashes ← pull and extract hashes for image_name
8      similarity ← calculate similarity between local_hashes and hub_hashes
9      write similarity to output_path
10     if similarity is high then
11         remove both local_image and hub_image
12         exit loop
13     else
14         remove hub_image
15     end if
16 end for
17 remove local image files
18 end
```

Algorithm 1은 도커 이미지 유사성 검사 모듈의 주요 동작 과정을 보여준다. 도커 이미지 유사성 확인 모듈은 개발자가 자신의 목적을 위해 개발하여 로컬 환경에 저장한 로컬 도커 이미지의 이름과 태그를 입력으로 받으며, 확인한 내역에 대해 확인할 수 있는 파일 경로를 지정하는 것으로 시작된다. 이후, 확인 결과를 저장할 수 있는 파일을 열고, 로컬 도커 이미지와 동일한 태그를 가진 도커 이미지를 Docker Hub로부터 다운로드하여 이를 비교한다. 로컬 도커 이미지와 Docker Hub로부터 다운로드 받은 이미지를 비교할 시 각 도커 이미지의 레이어를 추출하고, 해시값을 계산하여 저장된 해시값을 기반으로 비교를 수행하여 유사성을 계산한다. 만약 개발자가 개발한 도커 이미지가 Docker Hub에 존재하는 도커 이미지의 유사성 검사 결과가 80% 이상일 경우, 개발자가 개발한 도커 이미지의 독창성이 결여되어 있으며, 라이선스 위반 사항이 발생할 가능성이 존재함으로 판단하여 해당 도커 이미지를 업로드되지 못하도록 삭제 처리되도록 시스템을 구성하였다.

(그림 3)은 유사성 검사 모듈을 통해 도커 이미지 라이선스 컴플라이언스 위반 사항이 의심되어 검사 대상이 되는 도커 이미지가 삭제되는 화면을 보여준다.



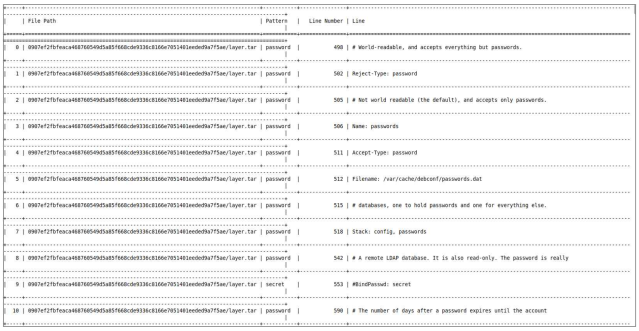
(그림 3) 도커 이미지 라이선스 컴플라이언스 위반 확인 및 대응

Algorithm 2 Sensitive Data Identification Module

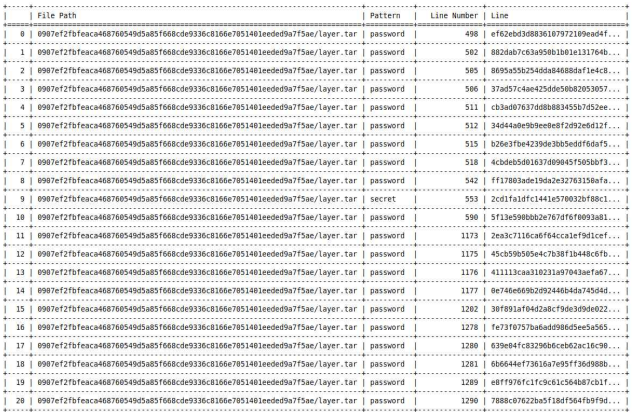
```
1  begin
2  input image_name, tar_path, output_path, patterns
3  try
4      Execute command 'docker save image_name -o tar_path'
5      Print "Image saved to tar_path"
6  catch error
7  Print error message
8  Terminate the script
9  end try
10 identifications ← []
11 Open tar_path as a tarfile
12 for each member in tarfile do
13     if member is a file then
14         file_content ← read member as string
15         for each line in file_content do
16             for each pattern in patterns do
17                 if pattern found in line then
18                     Add finding to identifications
19                 end if
20             end for
21         end for
22     end if
23 end for
24 Open output_path as CSV file
25 Write headers to CSV
26 for each finding in findings do
27 Write finding details to CSV
28 end for
29 end
```

만약, Algorithm 1을 기반으로 도커 이미지 유사성 검사를 통해 개발자가 개발한 도커 이미지가 독창적인 도커 이미지이며, 라이선스 컴플라이언스를 위반하였을 가능성이 적은 것으로 판단되는 경우, 해당 도커 이미지는 민감 데이터 식별 모듈로 전달된다. 민감 데이터 식별 모듈은 도커 이미지 내 SSH

키 혹은 API 키 등과 같은 민감 데이터를 식별하는 모듈로써 최근 해당 민감 정보를 활용한 공격이 지속적으로 발생하고 있음에 따라 이를 식별하여 대응하고자 하는데 목적이 있다. Algorithm 2는 민감 데이터 식별 모듈의 동작을 보여준다



(그림 4) 도커 이미지 내 민감 데이터 식별 결과



(그림 5) 도커 이미지 내 민감 데이터 식별 결과 암호화

Algorithm 2를 통해 확인할 수 있듯이 개발자가 업로드하고자 하는 도커 이미지 이름, tar 파일 경로, 민감 데이터 식별 정보를 저장할 CSV 파일 경로, 스캔할 패턴 목록을 정의 및 입력값으로 받음으로써 민감 데이터 식별 모듈의 동작이 시작된다. 이후, Docker의 'save' 명령을 사용하여 지정된 도커 이미지를 tar 파일 형식으로 저장하고, 도커 이미지 저장이 성공하였을 경우, 저장된 경로 함께 메시지를 출력하도록 하였다. 이후, 식별된 민감한 데이터를 저장할 리스트를 초기화하고, 도커 이미지의 tar 파일을 열어 정의된 패턴을 기반으로 검사를 수행한다. 만약 정의된 패턴이 tar 파일의 각 라인에서 발견되는 경우, 이를 identifications 리스트에 추가하고, 해당 결과를 CSV 파일에 기록하도록 함으로써 패턴에 따른 민감 데이터 정보를 테이블 형식으로 도출되도록 하였다. (그림 4)는 민감 데이터 식별 모듈을 통해 기록된 도커 이미지 내 민감 데이터 중 일부를 보여준다.

(그림 4)와 같이 도커 이미지 내 민감 데이터가 Raw 데이터로 저장되어 있을 경우, 악의적인 공격자에 의해 악용될 가능성이 존재함에 따라 본 논문에서 제안하는 시스템의 경우, AES 암호 알고리즘을 기반으로 'Line'에 해당하는 내용을 암호화하여 (그림 5)와 같이 저장될 수 있도록 하였다.

이에 따라 본 논문에서 제안하는 시스템을 통해 도커 이미지 라이선스 컴플라이언스 위반 가능성과 도커 이미지 내 민감 데이터를 통해 공격을 받아 피해를 입을 가능성을 줄일 수 있음을 보였다.

5. 결론

도커 컨테이너 기술의 중요성이 나날이 높아지면서 신뢰할 수 있는 도커 컨테이너 공급망의 중요성이 화두가 되고 있으며, 이 중 도커 이미지 라이선스 컴플라이언스 위반 사항에 대해 대응할 수 있는 방안은 필수불가결한 상황이다. 이에 따라 본 논문에서는 개발자가 개발한 도커 이미지와 Docker Hub 내 도커 이미지의 유사도 검사를 통해 도커 이미지 라이선스 컴플라이언스 위반으로 인한 이슈 발생의 가능성을 낮추고자 하였다. 또한, 도커 이미지 내 민감 데이터를 식별하고 이를 암호화하여 활용될 수 있는 기술을 제시함으로써 도커 이미지 내 민감 데이터를 활용한 공격에 대응하여 피해 가능성을 줄일 수 있음을 보였다. 이를 통해 신뢰성이 보장된 도커 컨테이너 공급망이 구성될 수 있을 것으로 기대할 수 있다.

Acknowledgment

이 논문은 2024년도 정부(개인정보보호위원회)의 재원으로 한국인터넷진흥원의 지원을 받아 수행된 연구임 (No.1781000011, 블록체인 환경에서의 개인정보보호 표준개발). 본 연구는 과학기술정보통신부 및 정보통신기획평가원의 정보통신방송혁신인재양성(메타버스융합대학원)사업 연구 결과로 수행되었음 (IITP-2023-RS-2023-00254529).

참고문헌

[1] "컨테이너 기술 시장 규모", [Online]. Available: https://www.businessresearchinsights.com/ko/market-reports/container-technology-market-106710 [Accessed: 2024-04-23].

[2] "Dockerfile 배포 시 컴플라이언스 책임은 누구에게", [Online]. Available: https://openchain-project.github.io/OpenChain-KWG/blog/2021/20210504-dockerfiles/ [Accessed: 2024-04-23].

[3] S. Kwon, J.-H. Lee, "Divds: Docker image vulnerabilitydiagnostic system". *IEEE access*, vol.8, pp.42666-42673, 2020.