A Preliminary Analysis of OSS Licenses in Docker Images

Abstract— **Background: In recent years, the use of container virtualization technology has been rapidly spreading to speed up software release and operation. In general, a containerized application image (e.g., Docker image) consists of multiple reused OSS packages. To reuse OSS, it is necessary to comply with the OSS licenses. Although there have been many studies on OSS license detection and license compatibility among OSS packages, but to the best of our knowledge, there is no study tackled with incompatible license problems among OSS packages in a container image. Aims: In this paper, we conduct a preliminary analysis to clarify the extent to which Docker images contain OSS license incompatibility problems. Method: We analyze 776 Docker images published on GitHub to determine whether license incompatibilities among OSS packages exist. Results: The analysis showed that a total of 2,167 software packages were used in the 776 Docker images. The majority of the software packages (71.3%) are compatible with the GPL family, but a non-negligible number of software packages (28.7%) are not compatible. The analysis also showed that 457 (58.9%) of the 776 images had GPL-related incompatibility problems. Conclusions: Unlike traditional software development, in which software packages to be reused are explicitly combined, Dockerfile creators who build and distribute Docker images might be less aware of the risks related to compatibility between OSS licenses. Our result is useful as information to improve the awareness of Dockerfile creators, and also indicates the necessity of future studies to detect and prevent the inclusion of license-incompatible OSS packages to container images.**

Index Terms—**OSS license incompatibility, Docker container images, association rule mining**

I. Introduction

In recent years, the use of container virtualization technology has been rapidly spreading in order to speed up software releases and operations [1]. By using software such as Docker to create images of containers, it is possible to quickly deploy and scale out beyond the operating environment [2] [3]. In addition, generated container images can be published through container repository services on the Internet such as Docker Hub. This ecosystem allows an unspecified number of developers to use high-quality applications without having to build an execution environment.

The widespread use of container virtualization technology has reduced the time required to run and deploy software, but has also created complex legal challenges. In April 2020, Hemel [4] released a white paper from the Linux Foundation describing how to handle OSS licenses when distributing and deploying Docker container images. They stated that in order to comply with the license of the entire container image, it is necessary to comply with the licenses of the software packages contained in the Docker image. Therefore, identifying complex OSS packages included in a Docker image is firstly required for Dockerfile creators. VMware has released Tern [5], a license identification tool that automatically identifies OSS packages and their licenses contained in container images to assist with license compliance within containers.

Next, it is necessary not only to identify OSS licenses in a container, but also to check whether there is compatibility among multiple licenses. If there is incompatibility between OSS licenses for software packages, trying to comply with one license will make it impossible to comply with the other licenses. For example, the GPL license and the Apache license are regarded as incompatible [6]. Many studies have tackled with the OSS license compatibility issues in the area of mobile apps [7], packaged software [8] , and source code [9] [10]. These studies have reported many cases where software products have been released under incompatible licenses. When a developer creates a container image, it is desirable to consider the compatibility of the OSS licenses of the software packages in the container in order to comply with the license of the entire container. However, to the best of our knowledge, no study has conducted to support checking the compatibility of software licenses contained in container images.

Therefore, in order to answer the following research questions, we conduct a preliminary analysis on the compatibility of OSS licenses of software packages included in Docker container images.

- **RQ1: How many software packages with incompatible OSS licenses are included in Docker container images?**
- **RQ2: To what extent do OSS license compatibility issues present in Docker container images?**

The analysis showed that a total of 2,167 software packages were used in the 776 Docker images available to the public, and that 28.7% of the software packages had licenses that were incompatible with the GPL family. We also found that 457 (58.9%) of the 776 Docker images had OSS license compatibility problems.

II. Motivation

This section describes the legal issues related to OSS licensing of container images and their ecosystem.

A. Software Packages in Container Images

Docker Hub, the most famous container registry service, has over 8,000,000 Docker Images available as of November 2021. Docker Hub ecosystem is one of the success factors that have contributed to the rapid growth in the use of containers. However, when reusing a container image published on the

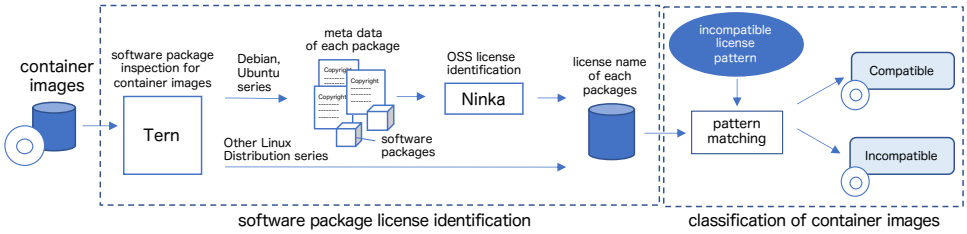

Fig. 1: Identification of OSS licenses and classification of OSS license compatibility in Docker images

Internet, it is necessary to comply with the licenses of the software packages contained in the image. A Docker image is generated by building a Dockerfile. There are three types of patterns in which software packages are introduced into a Docker image as follows.

(a) **Inheritance from a base image:** In a Dockerfile, it is common to specify a base image first using the FROM syntax. Specifying a base image allows users to reuse an existing container image. The part of a container image built from a base image is called base layer, and the part modified by creators of Dockerfile is called custom layer. For example, when adopting the official apache-httpd image from Docker Hub as the base layer image, use the FROM syntax.

(b) **Installation of additional packages:** The RUN syntax can be used to define commands to be executed in a container. In general, software packages are installed from software repositories according to the Linux distribution and/or their parameters are set. For example, when installing a text editor such as Vim for maintenance, use the RUN syntax is used.

(c) **Deployment an application:** The ADD syntax is used to deploy a specific file or application file into a container. For example, when deploying a developed web application in a container, the Add syntax is used.

Hemel [4] warn about the necessity to comply with OSS licenses of all software packages, including the base layer as well as the custom layer, when distributing and reusing container images. Out of the above three patterns, (a) and (b) are not developed by Dockerfile creators, and thus need to be checked for OSS licenses. This study focuses on the issue of license compatibility of OSS packages included in (a) and (b).

B. OSS License Compatibility

When integrating multiple software packages into a single software, it is necessary to check whether the clauses of the licenses of each software are consistent. For example, a software product developed by reusing software released under GPL must be released under GPL. Reusing software released from Apache Foundation also generally requires the final product to be released under the Apache License. Therefore, GPL and the Apache license are not compatible at the same time. Many studies on the compatibility of OSS licenses have conducted. German et al. [8] analyzed the compatibility of OSS licenses applied to software packages and the source code contained in the software packages. Mlouki et al. [7] analyzed the types and compatibility of OSS licenses applied to Android applications. Golubev et al. [9] analyzed license violations, including license compatibility, associated with code clones. Liu et al. [10] proposed a method to predict OSS licenses after a change in source code based on the relationship between the change and license compatibility.

To the best of our knowledge, no study has investigated OSS license compatibility in container images, and the current compliance status of license compatibility remains unclear. There is a risk of license violation if a container image containing multiple OSS packages with incompatible licenses is reused.

III. OSS License Compatibility in Docker Images

In this section, we describe the design of preliminary study.

A. Methodology

Figure 1 shows the procedure for identifying software packages and OSS licenses contained in a Docker image and detecting OSS license compatibility. It consists of (1) Tern [5], a container analysis tool, (2) Ninka [11], an OSS license identification method, and (3) mapping of incompatible licenses. The details of each process are described below.

1) License identification with Tern: First, we use Tern, a container analysis tool, to obtain software packages included in Docker container images and their license information. Due to Tern's specification that APT is not supported, the output of the license information depends on the package management system of the linux distribution used in the base OS image layer in the container. In the case of linux distributions such as CentOS and Alpine Linux, the license name is output for each OSS included in the container image. In the case of linux distributions such as Debian and Ubuntu with APT, the content of the license meta-information of the packaged software (e.g., LICENSE file) is output. The LICENSE file only contains license statements and the information of software packages written in a specific format. Since OSS license names cannot be identified from the LICENSE files, we use Ninka to identify OSS license names.

2) License identification with Ninka: Next, we use Ninka [11] to identify the OSS license names for containers that use a specific linux distribution such as Debian and Ubuntu. Ninka outputs license names by matching license statements written in comments in source files with regular expressions registered in the tool in advance. Ninka shows the highest accuracy (96.6% accuracy rate) among several license identification

TABLE I: Combinations of GPL-incompatible OSS licenses targeted in our study

Licenses	GPL v2	GPL v3	Reasons for incompatibility
Apache v1	✓	✓	Restrictions on unauthorized use of Apache-related product names not imposed by GPL[3]
Apache v2	✓		Clauses on patent termination and indemnification imposed by GPL v2[4]
BSD4	✓	✓	Advertising clause not imposed by GPL[5]
CDDL v1.0	✓	✓	Weak copyleft license limited to files[6]
EPL v1.0	✓	✓	Weak copyleft license limited to modules[7][8]
LPPL	✓	✓	Weak copyleft license having multiple restrictions not imposed by GPL[9][10]
MPL v1.1	✓	✓	Weak copyleft license limited to files. Multi-licensing allows to select a license when creating derivative works[11].
OpenSSL	✓	✓	Clause stating the use of the OpenSSL toolkit when redistributing[12]
GPL v3	✓		Multiple restrictions that are not in GPL v2[13]. However, it is compatible with GPL v2+, which is released in version 2 or later.
LGPL v3	✓		Multiple restrictions that are not in GPL v2[14]

methods, and is also used in other analytical studies on OSS licenses. In this study, the meta-information of packaged software was used as input for Ninka.

3) Verification of compatibility among OSS licenses: Table I shows the seventeen combinations of license incompatibilities used in this study. The incompatible combinations were selected based on the references provided by the Free Software Foundation (FSF) [6] and the existing study [12]. Most of studies on the OSS license compatibility focus on GPL, because GPL clearly states that it prohibits imposing restrictions that are not in GPL and the majority of license violations occur due to GPL-compatibility problems. As in the existing studies, this study also analyzes OSS licenses that are incompatible with the GPL-family, but we plan to analyze compatibility problems of other OSS licenses in the future.

B. Dataset

Schermann et al. [13] analyzed 2,745 Dockerfiles on GitHub and published the results as a PostgreSQL database. In this study, we obtain a list of containers from the dataset published by Schermann et al. and create a dataset for analysis. In this study, out of 2,745 Dockerfiles, we conducted the following two filtering: (1) Dockerfiles with a base layer and a custom layer in the container image are selected for this analysis. In addition, since Dockerfiles may not be able to be built for various reasons, (2) only those Dockerfiles successfully built and generated container images are included in the analysis. As a result of the filtering, a total of 776 container images built from Dockerfiles are included in the analysis. Of 776 images, 156 were Docker official images and the other 620 were regular images.

[3]https://www.gnu.org/licenses/license-list.html#apache1.1
[4]https://www.gnu.org/licenses/license-list.html#apache2
[5]https://www.gnu.org/licenses/license-list.html#OriginalBSD
[6]https://www.gnu.org/licenses/license-list.html#CDDL
[7]https://www.gnu.org/licenses/license-list.html#EPL
[8]https://directory.fsf.org/wiki/License:EPL-1.0
[9]https://www.gnu.org/licenses/license-list.html#LPPL-1.2
[10]https://www.gnu.org/licenses/license-list.html#LPPL-1.3a
[11]https://www.gnu.org/licenses/license-list.html#MPL
[12]https://www.gnu.org/licenses/license-list.html#OpenSSL
[13]https://www.gnu.org/licenses/gpl-faq.html#v2v3Compatibility
[14]https://www.gnu.org/licenses/gpl-faq.html#AllCompatibility

TABLE II: Software packages and their OSS licenses

	Num. of packages	(%)
incompatible packages with GPL	**621**	**28.7**
Apache v1	6	0.3
Apache v2	31	1.4
BSD4	20	0.9
CDDL v1.0	2	0.1
EPL	12	0.6
GPL v2	100	4.6
GPL v2+	208	9.6
GPL v3+	101	4.7
LGPL v3+	26	1.2
LPPL	105	4.8
MPL v1.1	6	0.3
OpenSSL	4	0.3
compatible packages with GPL	**1,546**	**71.3**

The "+" notation indicates "or later" release.

C. Results

1) ***RQ1: How many software packages with incompatible OSS licenses are included in Docker container images?****:*
Motivation: To the best of our knowledge, there is no study that investigates OSS licenses of software packages that compose a Docker container image. In order to verify the problems of license compatibility in Docker images, it is first necessary to identify the type of OSS licenses applied to software packages in Docker images.
Approach: Using Tern and Ninka as described in III-A1 and III-A2, we investigate OSS licenses applied to the software packages contained in the container images created by building 776 Dockerfiles.
Result: Table II shows the number of software packages released under GPL and GPL-incompatible licenses. In total, 2,167 software packages were found to be contained in 776 Docker container images. From Table II, we can see that GPL v2+ is the most common license for installed software packages (208) We also found that the majority of software packages (71.3%) is compatible with the GPL family.

Answer to RQ1:

The majority of software packages (71.3%) included in Docker container images are compatible with the GPL family, but a non-negligible number of software packages (28.7%) are not compatible with the GPL family.

TABLE III: Docker images containing combinations of incompatible OSS licenses

	incompatible	compatible	Total
Docker official image	119 (76.3%)	37 (23.7%)	156
regular image	338 (54.5%)	282 (45.5%)	620
Total	457 (58.9%)	319 (41.1%)	776

TABLE IV: Breakdown of incompatible license combinations in Docker images

	GPL v2	GPL v2+	GPL v3+
Apache v1	81	81	79
Apache v2	126	NA	NA
BSD4	**321**	**397**	**387**
CDDL v1.0	0	2	0
EPL v1.0	24	24	23
LPPL	2	2	2
MPL v1.1	82	82	82
OpenSSL	9	12	7
GPL v3+	**354**	NA	NA
LGPL v3	65	NA	NA

*2) **RQ2: To what extent do OSS license compatibility issues present in Docker container images?**:*

Motivation: There is a risk of license violation if a container image containing multiple OSS with incompatible licenses is distributed and reused. Since no study focuses on OSS licenses applied to software packages in container images, it is necessary to reveal the compliance status for container images.

Approach: Based on the result of RQ1, we examine the seventeen combinations of license incompatibilities described in III-A3 for container images built by 776 Dockerfiles.

Result: Table III shows that 119 out of 156 official Docker images and 338 out of 620 regular images exhibit the problems of license incompatibility among software packages. Overall, 457 out of 776 (58.9%) container images had some problems. The reason why more compatibility problems were included in the official containers would be that more software packages were installed in the official containers, which made compatibility problems more likely to occur. Table IV shows that GPL v2+ and BSD4 caused the most compatibility problems.

Answer to RQ2:

The majority of the analyzed images (58.9%) contain software packages that are incompatible with GPL. In particular, it was surprising to find that 76.3% of the Docker official images had compatibility problems with GPL. The OSS license compatibility problems in the studied images are most often caused by the simultaneous inclusion of GPL v2+ and BSD4.

IV. Discussions

Surprisingly, the results of RQ2 showed that 76.3% of the Docker official images had license compatibility problems. We discuss the causes of the problems using an example of a Dockerfile inducing OSS license incompatibilities. We also discuss the direction of future research on the OSS license compatibility for Docker images.

A. Example of Dockerfile Inducing License Incompatibilities

From the result in Table 4, we can see that Docker images containing software packages licensed under BSD4 cause compatibility problems with other software packages licensed under GPL. Therefore, we investigated Docker images containing BSD4 software packages and found an interesting pattern in a Dockerfile used to build a Docker image. The following is an example of a Dockerfile used to build one of Docker official images.

Listing 1: Dockerfile for a distribution image

```
1 FROM buildpack-deps:buster
2
3 ENV PATH /usr/local/bin:$PATH
...
```

In this Dockerfile, `buildpack-deps:buster` is inherited, and the `ENV` syntax is used to configure the environment. `buildpack-deps` is a collection of common dependencies used for installing various language modules. Just by looking at this Dockerfile, it is not possible to notice that the Docker image generated by building this Dockerfile has a license problem. Therefore, we need to find out what software packages are present in the `buildpack-deps:buster` image.

Listing 2: Dockerfile for buildpack-deps:buster

```
1 FROM buildpack-deps:buster-scm
2
3 RUN set -ex; \
4       apt-get update; \
5       apt-get install -y --no-install-recommends \
...
```

In this Dockerfile, `buildpack-deps:buster-scm` is inherited, and then the `RUN` syntax installs build tools and so on. It seems no license compatibility issue is occurred. The Dockerfile created to build the `buildpack-deps:buster-scm` image is as follows.

Listing 3: Dockerfile for buildpack-deps:buster-scm

```
1 FROM buildpack-deps:buster-curl
2
3 RUN apt-get update && apt-get install -y
                --no-install-recommends \
4       git \
5       mercurial \
...
```

`buildpack-deps:buster-scm` introduces source control systems (SCM) and provides version control functions to the distributed image. `buildpack-deps:buster-curl` inherits from this Dockerfile. Again, it seems no license compatibility issue is occurred.

Listing 4: Dockerfile for buildpack-deps:buster-curl

```
1 FROM debian:buster
2
3 RUN apt-get update && apt-get install -y
                --no-install-recommends \
4       ca-certificates \
5       curl \
6       netbase \
7       wget \
...
```

In `buildpack-deps:buster-curl`, `debian:buster`, the base OS image, is finally called by the `FROM` syntax. The `RUN` syntax installs the necessary modules in the environment where it is very common to download JARs such as in JRE, but without checking out the code. Here, we noticed that `apt-get install` automatically installs modules (e.g., `libsasl` and `libsaslmodulesdb`) that have dependencies on`curl` which is a tool for transferring data specified with URL syntax. However, even here, it is usually difficult to notice that software packages incompatible with GPL are installed. This is because `ca-certificates`, `netbase`, and `wget` are licensed under GPL, while `curl` is licensed under the MIT license which is compatible with GPL. However, modules such as `libsasl` and `libsaslmodulesdb` which are introduced by the dependency with curl, are licensed under BSD4, and cannot coexist with GPL.

B. Future Research Direction

Container virtualization such as Docker, is a powerful technology that makes it possible to quickly develop and distribute applications as container images. However, as we have seen in the above example, it is very difficult to be exactly aware of what software packages are used in the final container image, since container images are inherited in multiple ways. We were able to identify the OSS licenses included in the 776 Docker images by using the analysis method described in Section III-A, but it would be nearly impossible for individual developers, especially those who casually reuse Internet code [14], to be aware of the problems.

Container virtualization technologies are rapidly spreading in commercial activities. As pointed out by Hemel[4], in order to mitigate legal risks, OSS licenses included in a container image which are created, distributed, and used must be checked at every layer of the container image. Therefore, solving the problems of compatibility among OSS licenses in container images would be an important research topic for the modern software development. We believe that there are three possible directions for future research.

Comprehensive analysis: Our analysis covered 2,745 Dockerfiles provided by Schermann et al.[13], but more than eight million images are registered in Docker Hub. The results might not represent a comprehensive view of the OSS licensing problems in the Docker ecosystem. We also believe that further analysis of OSS licensing in container virtualization technologies other than Docker (e.g., LXC) is needed.

Detection and prediction of incompatible licenses: It is difficult for Dockerfile creators to be aware of the licenses of OSS packages included in inherited container images. It is necessary to have a technology that detects and notifies incompatibilities between licenses of OSS packages added to a Dockerfile by an individual developer as a custom layer and licenses of software packages in a base layer. As we saw in Section 4.1, at the point of invoking the base image, we can make some guesses as to whether or not there will be license incompatibilities in the Dockerfile that is created.

Support for ensuring compatibility: Our analysis showed that a number of Docker images have OSS license incompatibility problems. It would be very helpful if we could recommend combinations of software packages that can avoid compatibility problems, or alternative software packages that can solve compatibility problems.

C. Threats to validity

Internal validity: Our analysis rely heavily on Tern and Ninka. While we expect few false positives due to the high performance of both tools, some false negatives might have occurred (i.e., more incompatible problems will be observed).
External validity: As mentioned earlier, there are more than eight million Docker images registered in the Docker Hub. The results of the analysis may not be representative of the entire Docker ecosystem. However, it is important to note that 76.3% of the official Docker images that many developers use as their base layer images had compatibility problems.

V. Conclusions

In this paper, we focused on the OSS license incompatibility caused by combinations of software packages included in Docker images. We investigated how many combinations of software packages with incompatible OSS licenses are included in public Docker images. As the result of the analysis, we found that 58.9 % of the Docker images contained OSS license compatibility problems.

References

[1] A. Zerouali, V. Cosentino, G. Robles, J. Gonzalez-barahona, and T. Mens, "Conpan : A tool to analyze packages in software containers," in *MSR '19*, 2019, pp. 592–596.

[2] K. Eng and A. Hindle, "Revisiting dockerfiles in open source software over time," in *MSR '21*, 2021, pp. 449–459.

[3] J. Cito, G. Schermann, J. Wittern, P. Leitner, S. Zumberi, and H. Gall, "An empirical analysis of the docker container ecosystem on github," in *MSR '17*, 2017, pp. 323–333.

[4] A. Hemel, "Docker containers for legal professionals," 2020. [Online]. Available: https://www.linuxfoundation.jp/wp-content/uploads/2020/04/Docker-Containers-for-Legal-Professionals-Whitepaper_v4.ac_-3.pdf

[5] Tern Project, "Software composition analysis tool," https://github.com/tern-tools/tern, (Accessed in November, 2021).

[6] Free Software Foundation, "Frequently asked questions about the gnu licenses: How are the various gnu licenses compatible with each other?" https://www.gnu.org/licenses/gpl-faq.html.en#AllCompatibility.

[7] O. Mlouki, F. Khomh, and G. Antoniol, "On the detection of licenses violations in the android ecosystem," in *SANER '16*, 2016, pp. 382–392.

[8] D. German, M. Di Penta, and J. Davies, "Understanding and Auditing the Licensing of Open Source Software Distributions," in *ICPC '10*, 2010, pp. 84–93.

[9] Y. Golubev, M. Eliseeva, N. Povarov, and T. Bryksin, "A study of potential code borrowing and license violations in java projects on github," in *MSR '20*, ser. MSR '20, 2020, pp. 54–64.

[10] X. Liu, L. Huang, J. Ge, and V. Ng, "Predicting licenses for changed source code," in *ASE '19*, 2019, pp. 686–697.

[11] D. German, Y. Manabe, and K. Inoue, "A sentence-matching method for automatic license identification of source code files," in *ASE '10*, 2010, pp. 437–446.

[12] D. German and J. Gonzalez-Barahona, "An empirical study of the reuse of software licensed under the gnu general public license," in *OSS '09*, 2009, pp. 185–198.

[13] G. Schermann, S. Zumberi, and J. Cito, "Structured information on state and evolution of dockerfiles on github," in *MSR '18*, 2018, pp. 26–29.

[14] M. Sojer and J. Henkel, "License risks from ad hoc reuse of code from the internet," *Communications of ACM*, vol. 54, no. 12, pp. 74–81, 2011.