УДК 004

Механизмы контроля целостности в системе контейнеризации Docker

И. В. Чумаков
Закрытое акционерное общество "ОКБ САПР", Москва, Россия
Московский физико-технический институт (государственный университет),
г. Долгопрудный, Московская обл., Россия

Рассматриваются механизмы распространения образов Docker и контроля их целостности. Проанализированы принципы работы Docker Notary и Registry.

Ключевые слова: Docker, Notary, Registry, the Update Framework.

Контейнерная виртуализация — не просто один из трендов нашего века, но и довольно удобный инструмент распространения программного обеспечения (ПО).

Контейнеры по перспективности сравнимы с облачными технологиями, и лидеры индустрии вкладывают значительные силы, чтобы приблизить эту идею к рынку. Контейнерной виртуализацией приложений активно занимаются Docker и VMware [1]. Благодаря их разработкам предприятия могут запускать запакованные в контейнеры приложения на привычной для них инфраструктуре. Это позволит серьезно упростить внедрение приложений, поскольку отпадет необходимость в создании сопутствующей инфраструктуры.

Если переходить к терминологии Docker, то контейнер — это запускаемый экземпляр образа Docker. Главное различие между контейнером и образом – это верхний слой, доступный на запись. Поэтому в системе контейнеризации Docker распространяются образы, в которых реализованы приложения.

Однако возникает проблема с безопасной доставкой такого образа от разработчика до клиента. В большинстве случаев загрузка производится по открытым каналам связи, например через Интернет. Поэтому требуются механизмы защиты, которые могли бы гарантировать, что образ загружается из надежного источника и в исправном состоянии, а также что это вообще именно образ, а не что-то другое.

В связи с этим в версии Docker 1.8 был представлен инструмент Docker Content Trust [2]. Данный инструмент позволяет разработчикам подписывать свои образы электронной цифровой подписью (ЭЦП), а пользователям — проверять эту ЭЦП. Прохождение такой проверки гарантирует целостность и аутентичность образа. Тем самым решается проблема с безопасной доставкой образа Docker.

Docker Content Trust состоит из двух компонентов: Notary, отвечающего за подпись образов, и Registry, отвечающего за их распространение. Рассмотрим подробнее каждый из этих компонентов.

В основе Notary лежит The Update Framework (TUF) [3]. Поэтому для понимания принципов работы Notary сперва стоит подробнее остановиться на TUF.

The Update Framework (TUF)

TUF помогает разработчикам обеспечить защищенность систем обновления ПО от атак, направленных на компрометацию репозиториев или ключей подписи [4]. TUF предоставляет гибкий фреймворк, который разработчики могут адаптировать для нужной им системы.

TUF определяет иерархию различных ключей с различными привилегиями и сроками действия ключа [5]. Эти ключи привязаны к определенным ролям, например владелец root-ключа получает root-роль в системе. Кроме того, TUF определяет набор файлов метаданных, которые должны присутствовать в каталоге верхнего уровня репозитория. Рассмотрим более подробно архитектуру фреймворка.

- Ключ root — главный ключ в иерархии. Этот ключ имеет самый долгий срок действия и самые высокие привилегии, предназначен только для подписи других ключей в системе, поэтому имеет особое значение для безопасности всей системы. Точнее, роль ключа root заключается в подписи файла с именем root.json, который требуется спецификацией TUF и перечисляет действительные открытые ключи для всех других ключей в системе. Таким образом, действительность остальных ключей всегда может быть проверена клиентом в любое время.
- Ключ snapshot — ключ, которым подписывают файл shapshot.json, где хранится список дей-

Чумаков Иван Владимирович, инженер, студент 4-го курса кафедры "Защита информации".
E-mail: chumakov@okbsapr.ru

Статья поступила в редакцию 13 июня 2018 г.

ствительных метаданных (имена файлов, их размеры и хэши*) за исключением timesmap.json. По сути, этот файл представляет собой "снимок", содержащий все данные о последней версии содержимого, хранящегося в репозитории. Идея использования таких "снимков" является одной из основных концепций TUF: оперировать наборами, а не отдельными файлами. Такой подход позволяет защитить систему от атаки mix-and-match.

- Ключ timestamp — ключ, которым подписывают файл timestamp.json, где хранится информация о действительном файле shapshot.json, а именно его хэш, размер и номер версии. Файл timestamp.json регулярно переподписывается через равные промежутки времени. Таким образом, клиенты могут удостовериться в том, что загрузили последнее обновление.
- Ключ targets — ключ, отвечающий за проверку файлов, которые защищаются. Им подписывается файл targets.json, в котором хранится список защищаемых файлов, а именно имена файлов, их размеры и хэши, и поэтому targets.json обеспечивает целостность этих файлов. Роль targets позволяет делегировать ответственность одной или нескольким подчиненным ролям, в результате чего, они также могут подписывать подмножество защищаемых файлов. Преимущество такого делегирования заключается в том, что владелец targets ключа не передает его кому-либо. Вместо этого он подписывает один или несколько делегированных ключей, которые применимы только к тем файлам, ответственность за которые делегирует владелец targets-ключа. В таком случае у делегированной роли нет возможности подписать какое-либо содержимое, за которое она не отвечает.

Рассмотрим процесс взаимодействия клиента с TUF для проверки обновления ПО.

1. Клиентское приложение поручает TUF найти доступные обновления. Если клиент впервые взаимодействует с репозиторием, то на клиентскую машину скачивается root.json и импортируется открытый ключ root.
2. TUF скачивает файл timestamp.json из репозитория, проверяет его подпись при помощи открытого ключа из root.json и сверяет его с текущей версией, хранящейся на клиентской машине.
3. Если TUF определяет, что snapshot.json был изменен, фреймворк скачивает этот файл и проверяет его подпись так же, как и в случае с файлом timestamp.json. Затем он проверяет, были ли изменены какие-либо другие (root.json и/или targets.json) метаданные.
4. Если был изменен root.json (например, в случае компрометации какого-либо ключа и последующей ротации всех ключей), то скачивается последняя его версия из репозитория и процесс повторяется снова с пункта 1.
5. Если был изменен targets.json, т. е. изменены один или несколько защищаемых файлов, то TUF скачивает эти файлы, проверяет их целостность, после чего формирует список файлов, которые могут быть обновлены. Потом этот список предоставляется клиентской системе обновления ПО.
6. Все файлы, которые необходимо обновить, TUF скачивает из репозитория, сохраняя во временном каталоге, и проверяет их подписи. Только после успешной проверки всех файлов TUF предоставляет их системе обновления ПО.

Docker Notary

Теперь перейдем непосредственно к рассмотрению Notary.

Notary — это инструмент, который позволяет публиковать доверенные наборы данных и управлять ими [6]. Разработчики могут подписывать эти наборы цифровой подписью, а потребители — проверять целостность данных и аутентифицировать их.

Notary содержит два главных компонента [7]: сервер Notary и сервер для подписи Notary. Клиенты Notary взаимодействуют только с сервером Notary для получения метаданных из него или для передачи метаданных ему. Сервер хранит файлы метаданных TUF для одного или нескольких наборов данных в связанной базе данных.

Сервер для подписи Notary можно рассматривать как независимый от сервера Notary элемент, который хранит все закрытые ключи TUF в отдельной базе данных и подписывает метаданные для сервера Notary.

Если рассматривать в целом структуру Notary, то можно сказать, что сервер Notary представляет собой фронтенд, поскольку именно он взаимодействует с клиентами, в то время как сервер для подписи — это бекенд, поскольку он напрямую соединен только с сервером Notary. Такая архитектура предоставляет несколько преимуществ. Во-первых, метаданные TUF, предназначенные для отправки клиентам, не смешиваются с ключами TUF в одной базе данных. Во-вторых, закрытые ключи не хранятся на уязвимом фронтенде, который напрямую доступен клиентам.

Docker Registry

Registry — это система хранения и распространения данных, содержащая именованные образы Docker, доступные в различных версиях с тэгами [8].

* Здесь и далее хэш – результат криптографической хэш-функции SHA256, на вход которой было подано содержимое файла.

Registry представляет собой контентно-адресуемое хранилище данных [9]. В таком хранилище местоположение каждого элемента определяется самим элементом. Точнее, элемент используется для вычисления криптографического хэша, который, в свою очередь, определяет адрес, под которым хранится элемент. В Docker Registry в качестве хэш-функции используется SHA256.

При этом Registry не хранит образы Docker целиком, как единый элемент. Поскольку образ представляет собой набор слоев, причем один или несколько слоев могут быть использованы разными образами, в качестве элемента в Registry используются именно эти слои. Это позволяет сократить избыточность данных и уменьшить место, используемое для хранения образов.

Для управления такой системой используются манифесты образов (Image Manifest), в которых содержатся списки слоев образов и значения их хэшей, а также имена образов и их тэги.

Таким образом, Docker Registry хранит только набор уникальных слоев и манифесты образов вместо целых образов. Пользователи при получении образа из Registry могут проверить целостность полученных слоев, сверяя хэши скаченных слоев с хэшами, хранящимися в манифесте. Тем самым проверяется и целостность всего образа.

Напомним, что при получении образа пользователи никогда не используют громоздкое значение хэша. Вместо этого они используют его имя и тэг. Как следствие для безопасности всей такой системы необходима служба, способная безопасно предоставлять по имени образа его манифест. В качестве такой службы выступает Notary.

Notary предоставляет клиенту манифест, идентифицирующий разработчика образа, а также указывающий на слои, которые потребуется скачать из Registry.

Заключение

Рассмотрены Registry и Notary, реализующие механизм контроля целостности и аутентификации, который используется в Docker для распространения образов. Коротко он заключается в следующем.

- Разработчик выкладывает в Registry свой образ и при помощи Notary подписывает его манифест, в котором хранятся имя образа, тэг и хэши слоев, принадлежащих этому образу.
- При скачивании пользователем образа по имени и тэгу. Docker сперва проверяет ЭЦП манифеста при помощи Notary. В случае успешной проверки Docker скачивает из Registry необходимые слои, считывает от них хэш и сверяет с хэшами из манифеста, т. е. проверяет целостность образа.

Систему контейнеризации Docker удобно использовать для доставки приложений от разработчиков к конечным пользователям по открытым каналам связи, поскольку в ней реализованы механизмы убедительного подтверждения подлинности и аутентичности.

Литература

1. vSphere Integrated Containers [Электронный ресурс]. URL: https://www.vmware.com/products/vsphere/integrated-containers.html (дата обращения: 24.05.2018).
2. Overview of Docker Hub [Электронный ресурс]. URL: https://docs.docker.com/docker-hub/ (дата обращения: 24.05.2018).
3. Notary Overview [Электронный ресурс]. URL: https://github.com/theupdateframework/notary (дата обращения: 24.05.2018).
4. A Framework for Securing Software Update Systems [Электронный ресурс]. URL: https://github.com/theupdateframework/tuf (дата обращения: 24.05.2018).
5. The Update Framework Specification [Электронный ресурс]. URL: https://github.com/theupdateframework/ specification/blob/ master/tuf-spec.md (дата обращения: 24.05.2018).
6. Getting started with Docker Notary [Электронный ресурс]. URL: https://docs.docker.com/notary/getting_started (дата обращения: 24.05.2018)
7. Understand the Notary service architecture [Электронный ресурс]. URL: https://github.com/theupdateframework/notary/blob/master/docs/service_architecture.md (дата обращения: 24.05.2018).
8. About Registry [Электронный ресурс]. URL: https://docs.docker.com/registry/introduction/ (дата обращения: 24.05.2018).
9. Docker Registry HTTP API V2 [Электронный ресурс]. URL: https://docs.docker.com/registry/spec/api/ (дата обращения: 24.05.2018).

Mechanisms of control of integrity in the Docker container system

I. V. Chumakov
Closed Joint Stock Company "OKB SAPR", Moscow, Russia
Moscow Institute of Physics and Technology (State University), Dolgoprudny, Moscow region, Russia

This article is devoted to the mechanisms of distribution of Docker images and control of their integrity. The principles of Docker Notary and Registry operation are analyzed.

Keywords: Docker, Notary, Registry, the Update Framework.

Bibliography — 9 references. *Received June 13, 2018*