Ли Ц., Лю Л., Уласы Б. Анализ и исследование безопасности контейнеров docker // Международный журнал информационных технологий и энергоэффективности. – 2022. – Т. 7 № 3(25) часть 3 с. 65–72

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.056

АНАЛИЗ И ИССЛЕДОВАНИЕ БЕЗОПАСНОСТИ КОНТЕЙНЕРОВ DOCKER

[1] Ли Ц., [2] Лю Л., [3] Уласы Б.
Университет ИТМО, Санкт-Петербург, Россия (197101, г. Санкт-Петербург, Кронверкский пр., 49), e-mail: [1] magiclij@outlook.com, [2] magiclil@outlook.com, [3]magicula@outlook.com

В качестве ключевой технологии облачных вычислений контейнер Docker используется благодаря своим характеристикам упрощения конфигурации, повышения эффективности разработки, изоляции приложений, интеграции серверов, многопользовательской среды и быстрого развертывания. Исследованы проблемы безопасности контейнерной виртуализации, безопасности образа контейнера, безопасности контейнерной сети и стабильной работы контейнерных сервисов, с которыми сталкиваются Docker-контейнеры в процессе эксплуатации, и даны решения для вышеуказанных рисков.

Ключевые слова: Docker; образ; безопасность; риск.

ANALYSIS AND RESEARCH ON DOCKER

[1] Li J., [2] Liu L., [3] Ulas B.
ITMO University, St. Petersburg, Russia (197101, St. Petersburg, Kronverksky pr., 49), e-mail: [1] magiclij@outlook.com, [2] magiclil@outlook.com, [3]magicula@outlook.com

As a key cloud computing technology, Docker containers are used for their simplified configuration, improved development efficiency, application isolation, server integration, multi-tenant environment, and rapid deployment. The problems of container virtualization security, container image security, container network security, and container service stability faced by Docker containers during operation are studied, and solutions for the above risks are given.

Keywords: Docker; image; security; risk.

Введение

В последние годы многие предприятия и научно-исследовательские учреждения отдавали предпочтение облачным вычислениям как центру компьютерной и интернет-индустрии. В качестве ключевой технологии облачных вычислений [1] технология виртуализации быстро развивалась с использованием облачных вычислений. Виртуализация относится к виртуализации компьютера на несколько логических компьютеров с помощью технологии виртуализации [2]. Виртуализация может быть достигнута либо с помощью аппаратной эмуляции, либо с помощью операционной системы. Текущие основные технологии виртуализации, такие как XEN, VMware и KVM, представляют собой технологии виртуализации платформ, основанные на виртуальных машинах [3]. Эти решения, как правило, менее адаптируются к крупномасштабному развертыванию кластера из-за обычно больших образов виртуальных машин, необходимости в собственной независимой полной операционной системе (GuestOS) и большого количества занимаемых аппаратных ресурсов.

Docker — это облачный проект с открытым исходным кодом, основанный на языке Go, который появился на свет в начале 2013 года [4]. Как облегченная технология виртуализации, Docker имеет значительные преимущества перед традиционными виртуальными машинами при запуске приложений [5]: Контейнеры Docker работают очень быстро, а запуск и остановка могут быть достигнуты за секунды, по сравнению с традиционным подходом к виртуальной машине намного быстрее. Контейнеры Docker требуют очень мало ресурсов в системе, и хост может запускать до тысяч контейнеров, что невообразимо на традиционных виртуальных машинах. Docker поддерживает гибкий механизм автоматического создания и развертывания через конфигурационный файл Dockerfile для повышения эффективности [6]. Благодаря широкому использованию контейнерной технологии Docker обеспечение безопасности контейнеров Docker считается ключом к использованию контейнеров Docker.

1. Анализ безопасности докера.

Docker — это реализация виртуализации на уровне операционной системы Linux, и его суть ничем не отличается от процесса, работающего в Linux. В настоящее время безопасность контейнеров Docker существенно зависит от самой системы Linux [7]. Безопасность Docker в основном достигается за счет следующих аспектов: Безопасность изоляции контейнера, обеспечиваемая механизмом пространства имен ядра Linux. Благодаря механизму пространства имен программы в контейнерах Docker, работающие на одном хосте, не могут влиять друг на друга и имеют свои собственные уникальные сетевые стеки, к которым другие контейнеры не могут получить доступ. Возможность управления механизмом группы управления Linux для ресурсов контейнера защищена. Механизм группы управления Linux может гарантировать, что контейнеры на одном хосте справедливо распределяют ресурсы, такие как ЦП, диск и память хоста. Механизм группы управления также гарантирует, что нормальная работа хоста и других контейнеров не будет затронута, когда контейнер нестандартный. Права доступа, предоставленные механизмом возможностей ядра Linux, безопасны. Linux обеспечивает более детальное управление доступом к разрешениям. Чтобы обеспечить безопасность контейнеров, мы будем строго ограничивать разрешения контейнеров при использовании контейнеров Docker, ограничивая их только минимальными разрешениями, необходимыми для функций контейнера.

1.1. Зеркальные риски безопасности

Образ Docker — это статическое представление контейнера Docker, и безопасность среды выполнения контейнера зависит от безопасности образа Docker. Docker Hub — это официальный репозиторий образов Docker, в котором Образам Docker не хватает идеального контроля за загрузчиками, а их версия, качество и безопасность не контролируются пользователями. Если в Dockerfile не указан USER, Docker будет запускать контейнер с привилегиями root, в случае атаки на него могут быть получены привилегии root хоста. Использование паролей, ключей и другой информации в Dockerfile приведет к утечке данных после атаки. Использование образов из небезопасных источников приведет к неконтролируемым рискам безопасности для контейнеров Docker.

1.2. Риски безопасности виртуализации контейнеров

По сравнению с традиционными виртуальными машинами контейнеры Docker не имеют независимой конфигурации ресурсов и изоляции ресурсов на уровне ядра системы, поэтому существуют потенциальные риски изоляции ресурсов и неполных ограничений ресурсов.

Атака с выходом из контейнера означает, что злоумышленник получает полномочия root незаконными средствами и получает возможность выполнения команд с определенными правами хоста или других контейнеров, что влияет на текущую безопасность хоста или других контейнеров.

Контейнер Docker и хост совместно используют ядро операционной системы, и существуют потенциальные риски, связанные с отсутствием изоляции файловой системы, изоляции процессов и изоляции межпроцессного взаимодействия между контейнером Docker и другими контейнерами или хостами на хосте. Проблемы безопасности, связанные с изоляцией контейнера Docker, в основном включают следующие два

Ситуация: Злоумышленник напрямую атакует ядро операционной системы, создавая угрозу безопасности для контейнера Docker на хосте. Злоумышленник незаконно получает данные, управляя контейнером Docker для доступа к хосту или другим контейнерам.

Контейнер Docker и хост совместно используют аппаратные ресурсы, такие как ЦП, память и дисковое пространство. Если контейнер Docker не ограничивает использование ресурсов хоста контейнером Docker, злоумышленник может истощить аппаратные ресурсы хоста, захватив контейнер. или другое назначение контейнера, подвешивающего или даже умирающего.

1.3. Риски кибербезопасности

Безопасность сети контейнеров Docker всегда была одной из самых серьезных проблем, с которыми сталкиваются при использовании контейнеров. Из-за особой сетевой среды сеть Docker более строгая, чем традиционная сетевая безопасность.

Контейнеры Docker имеют различные сетевые режимы, предоставляя такие функции, как взаимодействие между контейнерами, взаимодействие между контейнерами между хостами и сеть кластера контейнеров. Ниже приводится анализ потенциальных рисков сетевой безопасности нескольких основных сетевых режимов. Режим моста заключается в том, что Docker по умолчанию принимает сетевой режим моста. В режиме моста на хосте создается виртуальный мост docker0, и все контейнеры Docker в этом режиме подключаются на хосте к мосту docker0. Все контейнеры Docker взаимно доступны. Хотя режим моста обеспечивает удобство взаимного доступа между контейнерами, из-за отсутствия механизма безопасного управления злоумышленники могут повлиять на безопасность хоста и других контейнеров с помощью широковещательных штормов, сниффинга, мошенничества с ARP и других методов атак. MacVLAN – это решение для виртуализации сетевых карт для контейнеров Docker. В мультитенантном сценарии для каждого арендатора виртуализируется отдельная сеть для обеспечения изоляции. Однако, поскольку контрольный доступ между контейнерами Docker в одной и той же виртуальной сетевой среде отсутствует, злоумышленники по-прежнему могут влиять на безопасность хоста и других контейнеров с помощью широковещательных штормов, сниффинга, мошенничества с ARP и других методов атак. Как наиболее распространенное решение для передачи данных и маршрутизации между хостами, оверлейная сеть в основном используется для построения кластерной сети контейнеров между хостами.

сеть. Как и два вышеупомянутых сетевых режима, оверлей по-прежнему имеет риски безопасности из-за отсутствия контроля доступа. Следовательно, независимо от того, какой сетевой режим используется, существует риск взаимных атак между контейнерами.

2. Реализация механизма безопасности Docker

2.1. Механизм защиты образа контейнера

Чтобы обеспечить безопасность образа контейнера Docker, после получения общедоступного минимального базового образа из официального репозитория образов вам необходимо использовать инструмент сканирования безопасности для сканирования загруженного образа на предмет безопасности. В настоящее время используется больше инструментов, включая Docker Security Scanning, Clair and Trivy и т. д. Программное обеспечение в обнаруживаемом образе содержит уязвимости CVE. После создания управляемого образа контейнера, чтобы упростить управление образом, необходимо создать управляемый частный репозиторий образов. Необходимо обеспечить отсутствие доступа к зеркальному складу из внешней публичной сети. В процессе подтягивания зеркала используется механизм проверки содержимого для решения проблемы взлома зеркала контейнера посередине. Чтобы убедиться, что исполняемые файлы и файлы конфигурации в контейнере Docker заслуживают доверия, проверка CRC32 выполняется для исполняемых файлов и файлов конфигурации перед каждым запуском контейнера Docker, и контейнер Docker может быть запущен только после успешной проверки. Предполагая, что исполняемые файлы и файлы конфигурации ключей имеют значения F1, F2... Fn соответственно, после развертывания программы-контейнера Docker контрольное значение C1 всех исполняемых файлов и файлов конфигурации ключей вычисляется один раз, а контрольное значение шифруется. и сохранен. Перед запуском контейнера Docker снова вычисляется проверочное значение C2 для всех исполняемых файлов и ключевых файлов конфигурации, и проверочные значения сравниваются дважды. Схема процесса показана на рисунке 1:

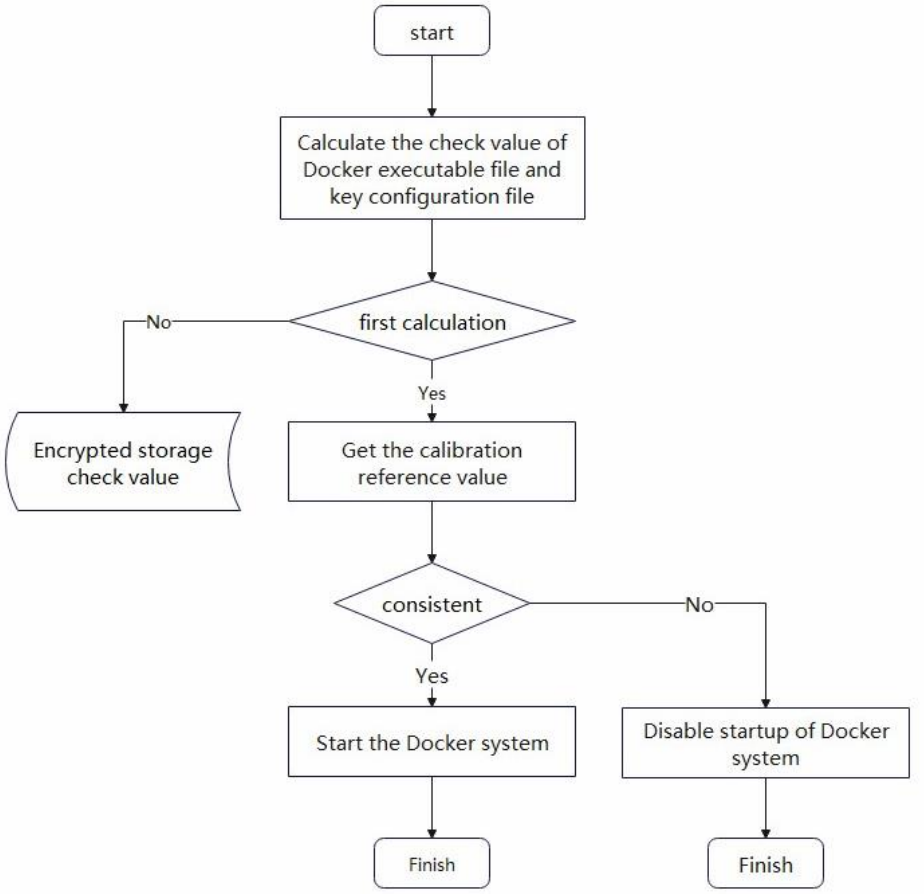

Рисунок 1 – Процесс запуска образа контейнера

2.2. Механизм безопасности виртуализации контейнеров

В архитектуре контейнера, основанной на существующих механизмах Cgroups, Namespace и возможностей ядра, управление ресурсами безопасности контейнера Docker может быть достигнуто за счет улучшения соответствующих механизмов на уровне ядра операционной системы.

Cgroups наложили ограничения на элементы ресурсов, такие как ЦП, память и скорость дискового ввода-вывода контейнеров Docker, чтобы контейнер не исчерпал аппаратные ресурсы на хосте. Когда контейнер запускается, параметры CPU, Memory и Device можно использовать для ограничения использования CPU, использования памяти и скорости чтения/записи диска. Ограничьте объем использования диска, создайте отдельного пользователя для каждого контейнера и ограничьте объем использования диска для каждого пользователя. Используйте каталоги файловой системы, такие как XFS, для ограничения использования диска. Создайте виртуальную файловую систему фиксированного размера для каждого контейнера.

Для ограничения доступа Docker-контейнера к ресурсам хоста используется механизм SELinux. При запуске Docker-контейнера SELinux можно запустить с использованием docker daemon --selinux-enabled = true.

2.3. Механизм безопасности контейнерной сети

В производственной среде хост обычно развертывает сотни или тысячи контейнеров Docker для предоставления услуг для мультитенантности. Чтобы отдельные контейнеры не занимали большую часть пропускной способности и делали другие службы контейнеров Docker недоступными, контроллер трафика модуля управления трафиком Linux используется в сети контейнеров для регулирования трафика.

Чтобы предотвратить злонамеренные сетевые атаки и обеспечить сетевую безопасность контейнеров Docker, можно реализовать контроль доступа к сети, настроив политику белого списка. Политика доступа к белому списку может быть реализована с помощью механизма брандмауэра, который поставляется с Linux.

Когда контейнеры Docker предоставляют услуги самостоятельно, не требуя, чтобы несколько контейнеров Docker формировали микросервисы. Для предотвращения вредоносных атак между контейнерами Docker необходимо запретить межконтейнерное взаимодействие, что можно установить командой dockerd --icc=false.

2.4. Механизм безопасности программы обслуживания контейнеров

Основная функция контейнеров Docker — предоставление надежных и стабильных облегченных сервисов виртуализации. Чтобы обеспечить стабильность сервисной программы в контейнере Docker, программа-сторож может выполнять регулярные проверки в контейнере. В то же время можно создать центр управления контейнером для регулярного получения данных пульса, отправляемых программой пульса в контейнере, и обработки их в соответствии с ситуацией приема.

Программа «watchdog» в основном используется для запуска и проверки сервисов в Docker. Его основной рабочий процесс показан на рисунке 2.

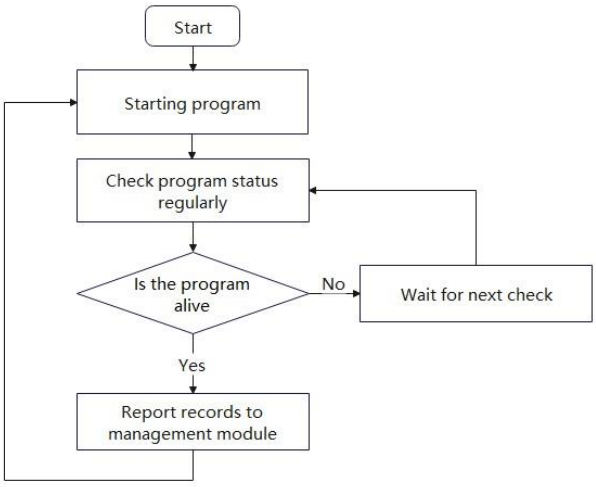

Рисунок 2 – Поток выполнения сторожевой программы

Шаг 1: запустите сервисную программу в контейнере.

Шаг 2: регулярно проверяйте статус сервисной программы. После того, как сервисная программа окажется зависшей, сообщите о проблеме в модуль управления и перезапустите сервисную программу.

Шаг 3: Дождитесь следующей проверки.

При реализации процесса пульса в контейнере Docker пульсация и рабочие данные регулярно отправляются в центр управления, чтобы достичь цели мониторинга ресурсов в контейнере. Его основной процесс показан на рисунке 3.

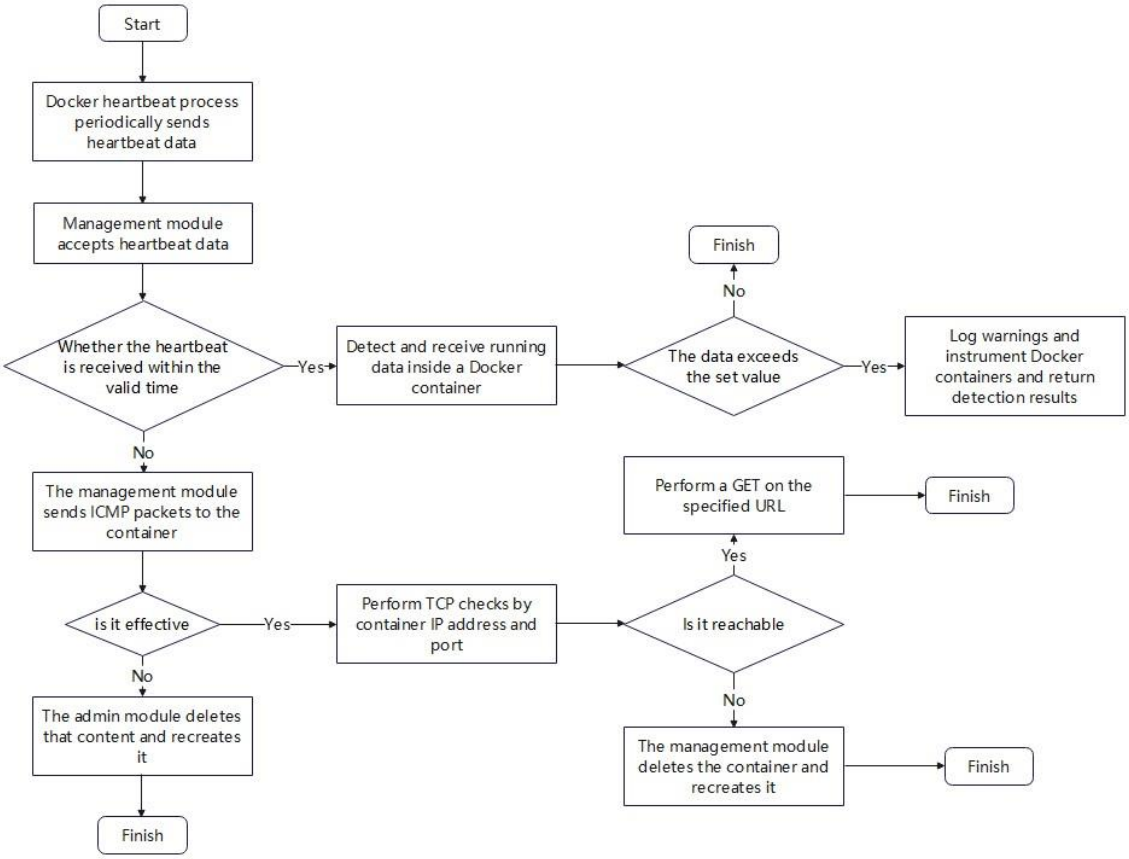

Рисунок 3 – Поток обработки пульса

Шаг 1: Процесс пульса в контейнере Docker периодически отправляет данные, такие как использование ЦП и памяти контейнера Docker, в центр управления.

Шаг 2: Центр управления получает данные, отправленные Docker, и записывает их. Если ЦП контейнера, память и другие данные превышают установленное безопасное значение в течение длительного времени, необходимо проверить контейнер Docker и перераспределить ресурсы.

Шаг 3: Центр управления не получает данные, отправленные процессом пульса в контейнере Docker, в течение периода ожидания. Отправьте ICMP-пакет в контейнер Docker, если контейнер не возвращается, уведомите хост о необходимости удалить контейнер и создать его заново.

Шаг 4: когда ICMP-пакет возвращается, выполняется проверка TCP по IP-адресу и порту и принимается возвращенный результат проверки. Если результат проверки успешен, это означает, что контейнер Docker доступен. В противном случае хост получает уведомление об удалении контейнера и его воссоздании.

ЗАКЛЮЧЕНИЕ

В этой статье представлены решения путем анализа угроз безопасности, с которыми могут столкнуться контейнеры Docker при использовании. Во время производства и

развертывания контейнеров Docker следует полностью учитывать возможные риски безопасности и проводить соответствующий анализ требований безопасности в соответствии со сценариями развертывания. В сочетании с решениями безопасности контейнеров Docker должны формироваться.

Список литературы

1. Zhang Y. Cloud Computing and Virtualization Technology // Computer Security – 2011. – №5. – С. 80–82.
2. Luo J. Cloud computing: architecture and key technologies / J. Jin, A. Song // Journal on Communications – 2011. Vol. 32. № 07. – С.3–21.
3. Библиотека программиста [Электронный ресурс]: Что такое Docker, и как его использовать? URL: https://proglib.io/p/docker (дата обращения 11.03.2020).
4. 1. Блог компании RUVDS.com [Электронный ресурс]: Изучаем Docker URL: https://habr.com/ru/company/ruvds/blog/438796/ (дата обращения 11.03.2020).
5. Каталог облачных служб Azure | Microsoft Azure [Электронный ресурс] URL: https://azure.microsoft.com/ru-ru/services/ (дата обращения: 11.02.2020).
6. Comparing Virtual Machines vs Docker Containers [Электронный ресурс] URL: https://clck.ru/MEwKH (дата обращения: 11.02.2020).
7. И.В.Подорожный, А.Н.Светличный, А.В.Подлеснов. Введение в контейнеры, виртуальные машины и docker // Молодой ученый. 2016. №19. С. 49-53. URL: https://moluch.ru/archive/123/33873/ (дата обращения: 02.07.2018).

References

1. Zhang Y. Cloud Computing and Virtualization Technology // Computer Security - 2011. - No. 5. – pp. 80–82.
2. Luo J. Cloud computing: architecture and key technologies / J. Jin, A. Song // Journal on Communications - 2011. Vol. 32. No. 07. – pp.3–21.
3. Programmer's Library [Electronic resource]: What is Docker and how to use it? URL: https://proglib.io/p/docker (Accessed 03/11/2020).
4. 1. RUVDS.com company blog [Electronic resource]: Exploring Docker URL: https://habr.com/ru/company/ruvds/blog/438796/ (accessed 03/11/2020).
5. Azure Cloud Services Catalog | Microsoft Azure [Electronic resource] URL: https://azure.microsoft.com/en-us/services/ (accessed 02/11/2020).
6. Comparing Virtual Machines vs Docker Containers [Electronic resource] URL: https://clck.ru/MEwKH (accessed 02/11/2020).
7. I.V. Podorozhny, A.N. Svetlichny, A.V. Podlesnov. Introduction to containers, virtual machines and docker // Young scientist. 2016. No. 19. pp. 49-53. URL: https://moluch.ru/archive/123/33873/ (date of access: 07/02/2018).