УДК 004.056.2
DOI: 10.52190/2073-2600_2021_3_19

Контроль целостности образов виртуальных машин на платформе OpenStack

Д. О. Стасьев
Московский физико-технический институт (национальный исследовательский университет), г. Долгопрудный, Московская обл., Россия

Определен формат, используемый для контроля целостности (КЦ) образов и конфигураций образов виртуальных машин (ВМ) путем их сравнения, в состав которого входят данные из набора таблиц из Glance DB (OpenStack Glance — компонент, созданный для управления образами ВМ). Описаны события взаимодействия разрабатываемого модуля с эталоном. Результатом работы стала архитектура программного модуля, реализующего проверку конфигураций образов ВМ на соответствие с эталоном.

Ключевые слова: виртуализация, виртуальная машина, OpenStack, Glance, образы виртуальных машин, целостность, контроль целостности, обеспечение целостности, компоненты виртуальных машин, программный модуль.

Согласно аналитическому исследованию [1], проведённому компанией "Код Безопасности" в 2018 г. среди 305 участников (IT-директора, ведущие инженеры, специалисты по защите информации, руководители направлений информационной безопасности), значительная часть организаций, независимо от масштабов их бизнеса, использует виртуализацию в серверной инфраструктуре более чем на 50 % серверов. Применение данной технологии позволяет эффективно использовать вычислительные мощности оборудования, сокращает время его простоя, однако создаёт опасность — появляется дополнительный канал для проникновения злоумышленника, повышается вероятность потери конфиденциальных данных, обрабатываемых на серверах с виртуализацией 70 % российских компаний-респондентов. Поэтому несмотря на то, что в среднем только 38 % российских компаний опасается действий злоумышленника, актуальность рассмотрения вопросов обеспечения безопасности при использовании виртуализации остаётся высокой.

Например, OpenStack, являясь одной из популярных платформ для создания виртуальных инфраструктур (ВИ — это система, которая обеспечивает поддержку виртуализации серверов, сети и хранилищ данных, создаётся с помощью инфраструктуры виртуализации), не предоставляет надёжных механизмов контроля целостности ВИ и создаваемых в ней виртуальных машин (ВМ). Только в 2015 г. в OpenStack начали внедрять защитные функции, такие, как, например, КЦ образов ВМ [2]. Для его реализации изначально стали использовать подпись (RSA-PSS) контрольной суммы, вычисленной от данных образа ВМ с помощью алгоритма хеширования MD5, который не является криптографически стойким. Отсутствие криптостойкости позволяло злоумышленнику заменить исходный образ ВМ на произвольный (злоумышленник мог добиться коллизии между значениями хеш-функции от исходного образа и произвольного). В 2016 г. от контрольной суммы отказались, заменили её вычислением подписи от содержимого образа напрямую [3]. Однако несмотря на это, в OpenStack оставались критические уязвимости. Например, в случае несовпадения хранимой и заново вычисленной подписей платформа выводила сообщение (в логах ошибок) о невозможности создания ВМ, содержащее требуемое значение подписи. Злоумышленник, получив информацию о требуемом для создания образа значении подписи, получал возможность так заменить значение подписи для своего образа в базе данных (БД), что создание ВМ из его образа не блокировалось системой. Иными словами, злоумышленник получал возможность создавать ВМ из произвольного образа, потенциально содержащего программные закладки или любое другое вредоносное программное обеспечение [4].

Несмотря на активное развитие OpenStack и исправление ошибок, в платформе продолжают находить уязвимости, которые создают угрозы различных уровней [5]. Таким образом, кроме использования встроенных в OpenStack механизмов защиты от злоумышленников, необходимо использовать дополнительные наложенные средства защиты информации (СЗИ).

Стасьев Денис Олегович, студент.
E-mail: stasev.do@phystech.edu

Статья поступила в редакцию 29 июля 2021 г.

Используемые методы исследования

При создании СЗИ для КЦ ВМ в OpenStack-based ВИ необходимо реализовывать мониторинг не только целостности критических файлов самих ВМ, но и метаданных образов ВМ (хранятся в отдельной БД в OpenStack). В свою очередь, только КЦ ВМ (в том числе образов ВМ) недостаточно для осуществления полноценного КЦ всей OpenStack-based ВИ. Необходимо дополнительно обеспечивать КЦ конфигурации (связи в графе конфигурации ВИ [6]) и компонентов ВИ [7]. Таким образом, КЦ OpenStack-based ВИ — комплексная задача, для решения которой необходим мониторинг целостности как отдельных компонентов ВИ, так и связей между ними.

Данная статья посвящена решению одной из частей описанной задачи КЦ ВМ в OpenStack — КЦ конфигураций образов ВМ. Для этого будет разработана архитектура программного модуля, реализующего КЦ.

Для разработки механизма внедрения модуля в OpenStack-based ВИ необходимо определить компоненты OpenStack, участвующие во взаимодействии с ВМ. Для этого из официальной документации OpenStack с помощью метода абстрагирования выделяются компоненты OpenStack для создания ВИ. Системный анализ полученных данных позволит определить компоненты OpenStack и их части, участвующие во взаимодействии с ВМ. С помощью применения такого эмпирического метода, как описание, изучается строение БД Glance DB для формирования структуры эталона конфигурации образа ВМ. Результатом работы станут архитектура программного модуля, реализующего КЦ, а также описание некоторых особенностей её реализации.

Обзор литературы

Компоненты OpenStack, участвующие во взаимодействии с ВМ. Перед исследованием особенностей платформы OpenStack введём основные понятия в области серверной виртуализации. Гипервизор — программа и/или аппаратная система, обеспечивающая одновременное, параллельное выполнение нескольких сред (каждая среда обычно является программной системой, содержит операционную систему (ОС) и эмулирует аппаратное обеспечение некоторой target-платформы) на одном хост-компьютере (host-платформе). Экземпляром ВМ (или просто ВМ) будем называть отдельную среду, которую можно получить с помощью гипервизора. Образом ВМ именуем совокупность файлов (может состоять из одного), используемую в качестве шаблона (образца) при создании новых экземпляров ВМ.

OpenStack — это система, состоящая из комплекса программного обеспечения (ПО), созданная для управления большими наборами вычислительных ресурсов, хранилищ и сетевых ресурсов [8]. Платформа OpenStack предоставляет архитекторам ВИ набор компонентов (сервисов), с помощью которых можно разработать архитектуру и создать требуемую ВИ, управлять как состоянием ВМ, так и аппаратными ресурсами.

Можно выделить несколько сервисов, которые созданы в OpenStack для взаимодействия с ВМ. OpenStack Glance используют для управления образами (шаблонами) ВМ (например, для добавления и удаления образов). Созданием ВМ из образов занимается сервис OpenStack Nova (рис. 1).

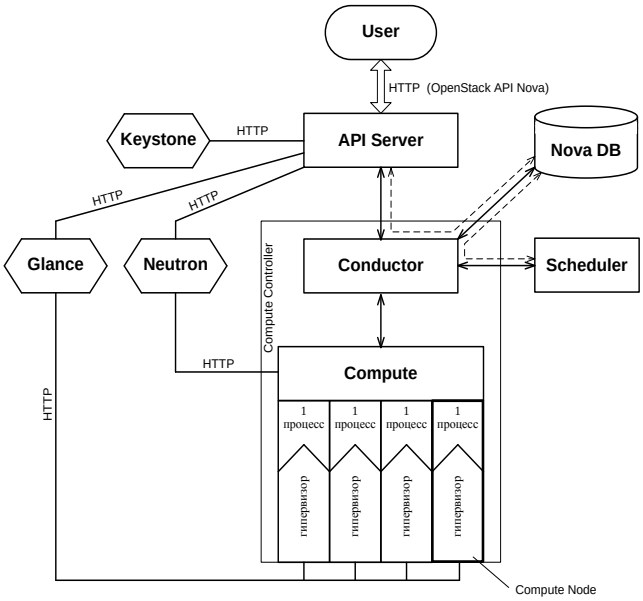

Рис. 1. Схема взаимодействия частей OpenStack Nova

При создании ВМ Nova обращается к Glance, который получает образ из какого-либо хранилища. Сервис Nova, кроме создания экземпляров ВМ, позволяет ими управлять [9]. Например, можно менять аппаратные ресурсы ВМ с помощью применения (смены) готовых конфигураций (содержат размер доступной оперативной и постоянной памяти экземпляру ВМ), называемых flavor. Также Nova позволяет выбрать и настроить гипервизоры для ВМ. Сами файлы образов и экземпляров ВМ находятся в отдельном хранилище. В зависимости от конфигурации OpenStack-based ВИ тип и реализация хранилища могут быть разными: в самом простом случае это может быть файловая система хост-компьютера, однако на практике часто используют специальные сервисы OpenStack, реализующие хранилища (OpenStack Swift, Open-

Stack Cinder) [10, 11]. Для аутентификации и авторизации пользователей в OpenStack используется сервис Keystone. Под пользователями в нём понимают как людей, работающих непосредственно с ВИ (например, администраторов и архитекторов ВИ), так и сервисы OpenStack, между которыми происходит взаимодействие. Рассматриваемые сервисы OpenStack (Glance, Nova, Keystone) реализованы как один либо как набор web-серверов, поэтому взаимодействие с ними и между ними происходит с помощью HTTP-запросов с соответствующими заголовками (для Keystone).

Хранение образов ВМ и их конфигураций в OpenStack. Жизненным циклом ВМ будем называть совокупность процессов и состояний системы, связанных с созданием, использованием, хранением и удалением экземпляров ВМ. Для обеспечения целостности ВМ необходим КЦ на всех этапах жизненного цикла ВМ. Первым этапом жизненного цикла ВМ является её создание из образа ВМ.

OpenStack предоставляет широкие возможности по взаимодействию с образами ВМ с помощью сервиса Glance. Рассмотрим его подробнее. Glance (Image service) — компонент, созданный для управления образами ВМ и метаданными ВИ [12, 13]. Под метаданными понимают элементы ассоциативного массива (пары ключ—значение), которыми могут управлять администраторы ВИ (например, создавать, удалять, применять к образам ВМ). До применения метаданных к конкретным ресурсам OpenStack они не влияют на ВИ, но после добавления метаданных к различным объектам OpenStack объекты могут использовать метаданные как источник настроек, т. е. метаданные могут влиять на конфигурацию частей ВИ.

Сервис Glance состоит из большого числа внутренних подсистем [14]. С помощью применения метода абстрагирования можно выделить четыре основные части, представленные на рис. 2: Glance API Server, Glance Registry, Glance DB (каталог метаданных) и хранилище образов и файлов ВМ. Glance Registry является основной частью Glance, предоставляет Glance API Server информацию о наличии образов в Glance и об их расположении в хранилище с помощью Glance DB. Glance DB является БД, которая содержит метаданные и расположение каждого образа в хранилище. Обычно для Glance DB используют реляционную БД, созданную на основе таких систем управления БД (СУБД), как MySQL и SQLite (являются наиболее популярными СУБД для OpenStack) [13—15]. Часто СУБД для Glance DB является единственной во всей OpenStack-based ВИ, поэтому она может использоваться и другими сервисами (обычно создаётся новый пользователь СУБД для отдельной требуемой БД для каждого сервиса; например, такой механизм реализован в OpenStack Nova). Образы ВМ, доступные через Glance, могут храниться в самых разных местах, от простых файловых систем до систем хранения, таких, как проекты OpenStack Swift, OpenStack Cinder или S3 [12]. Для работы Glance необходим сервис Keystone.

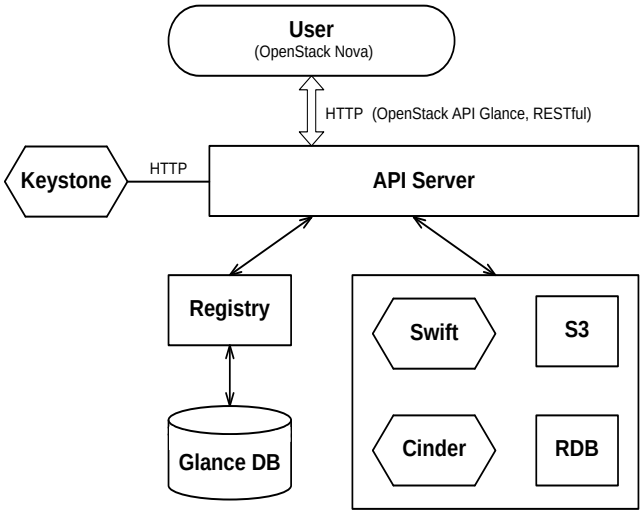

Рис. 2. Схема взаимодействия частей OpenStack Glance

Glance DB и эталон конфигурации образа ВМ. Для КЦ ВИ необходим, в частности, КЦ образов ВМ и их конфигураций, хранимых в Glance DB. Для КЦ самих образов, находящихся в хранилище, требуется вычислять контрольные суммы от содержимого файлов. Исследуем Glance DB, где хранятся метаданные и свойства образов ВМ. Можно объединить таблицы в несколько групп в соответствии с их ролями во взаимодействии с образами ВМ (таблица).

Связь таблиц БД Glance и их ролей во взаимодействии с образами ВМ

Роль во взаимодействии с образами ВМ	Таблицы
Используются для миграций (SQLAlchemy)	1. alembic_version 2. migrate_version
Содержат информацию о хранимых образах ВМ	3. image_locations 4. image_members 5. image_properties 6. image_tags 7. images
Используются для обработки "больших" образов ВМ сервисом Glance [16]	8. task_info 9. tasks
Необходимы для хранения метаданных [17]	10. metadef_properties 11. metadef_tags 12. metadef_resourse_types 13. metadef_objects 14. metadef_namespaces 15. metadef_namespace_resource_types

После определения ролей таблиц БД Glance во взаимодействии с образами ВМ можно перейти к вопросам КЦ образов ВМ. КЦ предполагает сравнение текущего состояния системы с некоторым его выбранным фиксированным состоянием, принятым за эталон [18]. После задания эталона в определённые моменты времени происходит сравнение актуального состояния системы с записанным в эталон. Наличие таких механизмов позволяет контролировать целостность системы [19]. В случае КЦ образов и метаданных образов ВМ в эталон образа ВМ должны входить поля таблиц (3)—(7), для КЦ метаданных всей ВИ — таблиц (10)—(15).

Поскольку рассматривается разработка механизма КЦ конфигураций образов ВМ, эталон конфигураций образов ВМ должен включать поля таблиц (3)—(7). Для вычисления значения эталона в данной работе будут использованы идентификатор образа ВМ (GUID, Globally Unique Identifier) (поле id таблицы images (7) БД Glance) и значение хеш-функции (контрольная сумма) от значений таблиц (3)—(7). В будущем предполагается, что данные пары будут храниться в аппаратно-защищённой области постоянной памяти. Для дополнительной защиты можно использовать идею патента [20] после согласования с патентообладателем: вычислять контрольную сумму не только от хранимых данных, но и от контрольной суммы предыдущей записи и ключа хранения, используемого программным модулем для подписи записываемых значений. Согласно патенту, можно использовать криптографию с открытым ключом, в которой подписывающее лицо (программный модуль) вычисляет контрольную сумму проверки целостности с помощью своего закрытого ключа, а лицо, желающее проверить целостность, может использовать свой открытый ключ для верификации. Вычисленная контрольная сумма присоединяется к записи данных [20].

Таким образом, поскольку разрабатывается механизм КЦ конфигураций образов ВМ, эталон образа ВМ должен включать значения таблиц (3)—(7) БД Glance.

Результаты

Архитектура программного модуля. Описание внутреннего устройства OpenStack позволило выделить компоненты OpenStack, участвующие во взаимодействии с ВМ. Одним из них является система хранения образов ВМ и их конфигураций — OpenStack Glance. После изучения её устройства был установлен формат эталона конфигурации образа ВМ. Однако для реализации программного модуля для КЦ только описания эталона недостаточно. Необходимо также разработать архитектуру модуля, учитывая особенности платформы OpenStack.

Опишем предполагаемую архитектуру программного модуля. Её функционирование будет основано на вычислении эталона образа ВМ и его хранении в памяти хост-компьютера. В некоторые моменты времени функция в разрабатываемом модуле для создания эталона будет заново вычисляться и полученное значение будет сравниваться с записанным в память эталоном. В случае несовпадения вычисленного значения и эталона администратор ВИ будет получать уведомление о нарушении КЦ конфигураций образов ВМ. Система должна запрещать создание ВМ из образов при нарушении КЦ конфигураций образов при своих ошибках, например при отсутствии возможности получения данных из БД для вычисления образа.

Модуль должен запускаться до старта ВИ, чтобы создавать эталоны для всех добавляемых образов ВМ в ВИ. Определим, в какие моменты времени (в ответ на какие события) система должна вычислять эталоны. Изначально при добавлении образа ВМ в ВИ необходимо создавать исходные эталоны и помещать их в файл (возможно применение аппаратно-защищённой области памяти для хранения файла). При попытке создания экземпляра ВМ из образа ВМ эталон должен заново вычисляться и сравниваться с записанным в файл. Дополнительно образ используется при запуске созданной ВМ.

Определим, как отслеживать необходимые для КЦ конфигураций образов ВМ события (добавление образа ВМ в ВИ, создание экземпляра ВМ из образа и запуск ВМ) в платформе OpenStack.

Для определения момента запуска KVM-based ВМ можно использовать libvirt. Libvirt — это наиболее часто используемый драйвер виртуализации в OpenStack [21]. OpenStack Nova для работы с ВМ использует libvirt при поддержке программы QEMU (для эмуляции аппаратного обеспечения различных платформ) и, если доступен, KVM. Libvirt является свободной реализацией API-гипервизора KVM и содержит набор дополнительных возможностей [22]. Одной из них являются хуки. Это скрипты, позволяющие изменить стандартное поведение компонентов системы. Libvirt позволяет запускать пользовательские хуки на определённые события libvirt, например на запуск ВМ (/etc/libvirt/hooks/qemu) [23]. При выполнении хука libvirt передаёт ему некоторые параметры через аргументы командной строки. В зависимости от них можно понять статус

(например, запуск или остановка ВМ) и имя ВМ. То есть создав подобный хук, можно отслеживать запуск ВМ и инициировать проверку КЦ образа ВМ непосредственно перед запуском ВМ. Подобная идея успешно реализуется в специальном программном обеспечении (СПО) СЗИ от несанкционированного доступа (НСД) "Аккорд-KVM" компании "ОКБ САПР" [24]. В случае нарушения КЦ администратор будет получать оповещение. Таким образом, использование хуков libvirt для отслеживания запуска ВМ является наилучшим решением для отслеживания событий запуска ВМ и её создания из образа среди встраивания во внутреннюю очередь сообщений OpenStack Nova и прокси-сервера обработки запросов к Nova.

Теперь, зная имя запускаемой ВМ, необходимо получить образ, для которого требуется посчитать эталон. Поскольку информация о ВМ хранится в части Nova DB сервиса OpenStack Nova, исследуем этот компонент в целях нахождения информации о взаимосвязи между полем id таблицы images БД Glance (GUID) и названием запускаемой ВМ. Получаем, что в БД Nova содержится таблица instances с интересующей информацией. В ней хранятся метаданные созданных ВМ в OpenStack-based ВИ, такие, как статус, количество выделенной памяти, имя хост-компьютера и т. д. С помощью полей hostname (display_name) и image_ref устанавливается взаимосвязь между именем ВМ и используемым образом ВМ. Используя эту таблицу, можно установить соответствие между именем запускаемой ВМ, которое передаётся хуку libvirt в качестве аргумента командной строки, и образом ВМ, для которого необходимо посчитать эталон.

Установлено, как проверять соответствие образа ВМ хранимому эталону. Перейдем к тому, как и когда создавать исходный эталон. Необходимо создавать исходный эталон для каждого образа ВМ в момент его добавления в OpenStack-based ВИ. Поскольку OpenStack Glance управляет хранением образов ВМ, рассмотрим его устройство подробнее. В отличие от OpenStack Nova он не содержит нескольких внутренних серверов. Glance написан на Python и является HTTP-сервером. Его структура (схема) реализована через набор файлов-компонентов, каждый из которых обрабатывает запрос и направляет следующему [25]. То есть Glance состоит из набора "слоёв", каждый из которых реализует определённые функции и отправляет запрос на следующий "слой". Для определения момента добавления образа ВМ можно использовать прокси-сервера для обработки запросов к самой БД. Данный способ позволит создавать исходный эталон сразу после добавления записей в БД. Ещё одним его преимуществом является удобство конфигурирования, поскольку при его внедрении необходимо только изменить настройки БД в Glance (без изменения других сервисов OpenStack).

Таким образом, установлены события (моменты времени), в которые будет происходить взаимодействие разрабатываемого программного модуля с эталонами конфигурации образов ВМ. Разработаны способы внедрения в OpenStack: выбраны использование хуков libvirt для определения момента запуска ВМ и прокси-сервер для БД Glance для определения момента добавления образов ВМ в сервис OpenStack Glance.

Практические аспекты реализации архитектуры модуля. Разрабатываемая система должна внедряться на хост-компьютер ВИ, иметь доступ к сервисам OpenStack и предоставлять удобный способ управления настройками КЦ конфигураций образов ВМ. При создании системы можно использовать клиент-серверную архитектуру. Сервер будет запускаться на хост-компьютере ВИ до старта ВИ и принимать HTTP-запросы (от хука libvirt), которые вызовут либо добавление эталона в память, либо проверку КЦ конфигурации образа ВМ. Клиентом может являться веб-приложение, которое по протоколу WebSocket будет получать сообщение и уведомлять администратора в случае нарушения КЦ. Сервер в таком случае должен запрещать запуск ВМ. Ещё одной частью системы будет хук libvirt, который при вызове отправит соответствующий HTTP-запрос на сервер в целях его уведомления о необходимости проверки КЦ образа запускаемой ВМ. Получив запрос, сервер обратится в БД Nova и определит идентификатор образа ВМ. Используя его, сервер вычислит эталон от значений таблиц БД Glance и сравнит значение с хранимым в памяти. В случае несовпадения значений сервер отправит широковещательное сообщение по WebSocket о нарушении КЦ конфигурации образа ВМ (сообщение получит администратор ВИ с установленным соединением). Подробная схема взаимодействия представлена на рис. 3.

Для создания исходных эталонов в памяти прокси-сервер БД Glance будет обрабатывать запросы в БД Glance. В случае запросов создания (изменения) образов ВМ будет вычисляться и сохраняться эталон. Подробная схема создания эталонов конфигураций образов ВМ представлена на рис. 4.

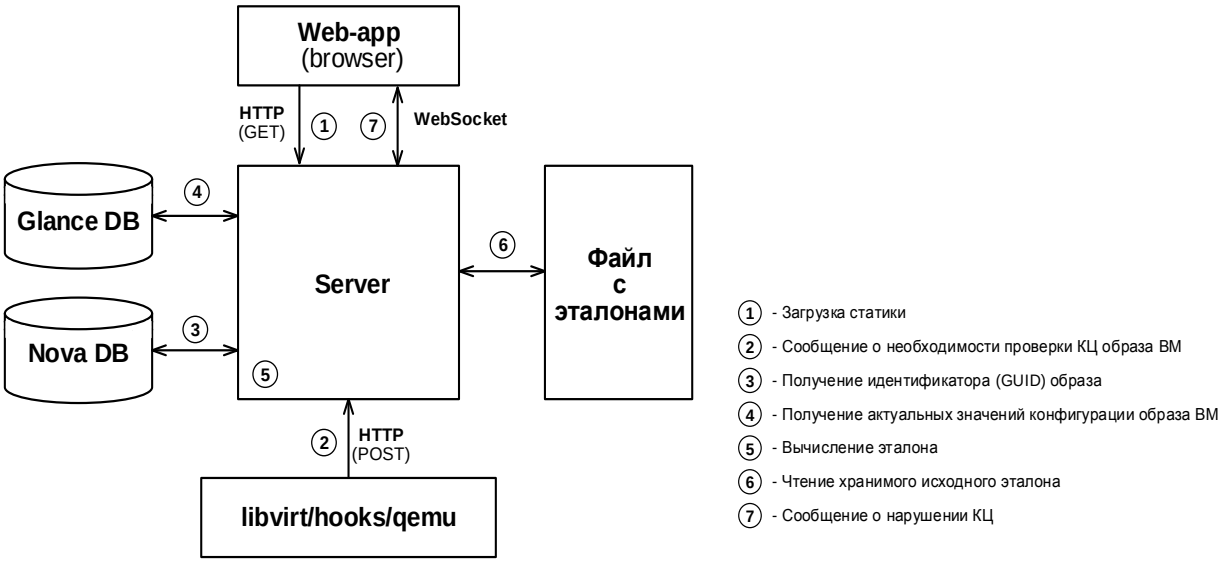

Рис. 3. Схема проверки КЦ конфигураций образов ВМ

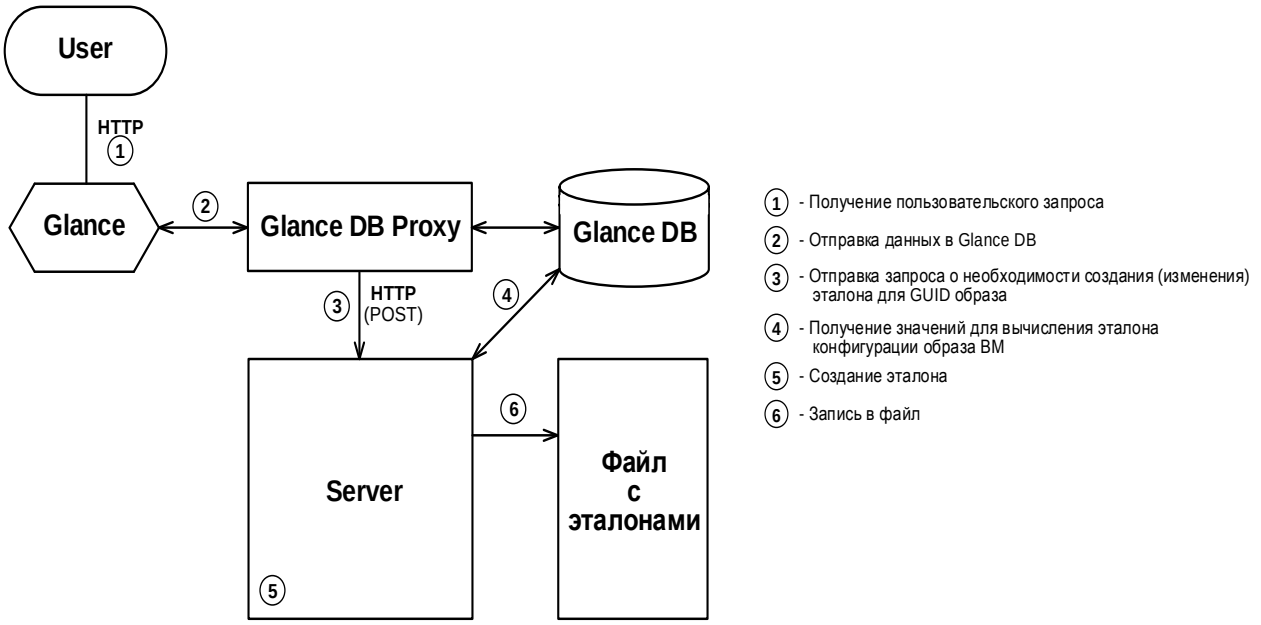

Рис. 4. Схема создания эталонов образов ВМ

Обсуждение

В платформе OpenStack для отслеживания необходимых для КЦ конфигураций образов ВМ событий помимо выбранных подходов (хуки libvirt для определения момента запуска ВМ и прокси-сервер для БД Glance для определения момента добавления образов ВМ) можно использовать и другие.

Старт ВМ и её создание из образа происходят с помощью обработки соответствующих запросов сервисом OpenStack Nova. Поскольку Nova является совокупностью нескольких сервисов, общающихся с помощью очереди сообщений RabbitMQ, можно читать эти сообщения (т. е. можно встроить модуль в сам сервис Nova). Однако формат этих сообщений строго не определён (поскольку является внутренней служебной частью сервиса). Поэтому существует риск возникновения проблем с совместимостью разрабатываемого СЗИ с новыми версиями платформы OpenStack. Поскольку OpenStack активно развивается и выпускает несколько релизов в год, потенциальные затраты на поддержание работоспособности такого СЗИ высоки.

Кроме того, очередь сообщений обрабатывает большое количество запросов между внутренними сервисами OpenStack Nova. Поэтому система для их чтения будет требовать значительных ресурсов.

Другим решением для определения момента запуска ВМ может стать создание прокси-сервера для обработки запросов к OpenStack Nova. Упрощённо Nova API Server (часть OpenStack Nova, см. рис. 1) можно считать HTTP-сервером. Взаимодействие между пользователем и Nova (и между другими сервисами OpenStack и Nova) основано на взаимодействии через публичное API. Оно хорошо документировано, т. е. на его основе можно разработать прокси-сервер со специальной обработкой некоторых запросов (здесь для вычисления эталона). Настроив конфигурации остальных сервисов OpenStack-based ВИ, возможно внедрить такой сервер в структуру OpenStack. Одной из проблем будет возможность различить легальных пользователей и нарушителей, поскольку необходимо обрабатывать запросы в соответствии с правами пользователей. Для этого потребуется отдельно настроить взаимодействие прокси-сервера с сервисом OpenStack Keystone (в случае возникновения проблем с настройкой можно заменить OpenStack Keystone собственным сервисом аутентификации и авторизации, однако целесообразность разработки такого решения вызывает вопросы, так как сервис необходимо будет адаптировать к взаимодействию со всеми сервисами OpenStack). Ещё одной проблемой может стать относительно большая задержка между проверкой КЦ конфигурации образа и запуском ВМ, поскольку после проверки КЦ запрос будет отправляться в другой сервис. Предложенное решение (из-за своей архитектуры: отправки запроса в другой сервис) не позволит существенно снизить задержку. Необходимо осуществлять проверку КЦ непосредственно перед запуском ВМ. Этого можно достичь с помощью хуков libvirt, как было выбрано в описанной ранее архитектуре программного модуля.

Для определения момента добавления образа ВМ существует несколько способов, кроме выбранного (прокси-сервер для БД Glance). Аналогично рассмотренному решению для OpenStack Nova можно сделать прокси-сервер для обработки запросов к Glance. В момент получения ответа от Glance прокси-сервер будет создавать исходный эталон. Однако возникает проблема, связанная с относительно большой задержкой между фактическим созданием записей в Glance DB и вычислением эталона. Другой способ — создание дополнительного слоя для Glance, который будет создавать эталон непосредственно до создания записей в БД. Однако это требует внедрения во внутреннюю структуру Glance, что усложнит поддержку разрабатываемого программного модуля при появлении новых версий OpenStack.

В заключение можно отметить, что особенностью разработанной архитектуры программного модуля для КЦ конфигураций образов ВМ является возможность внедрения КЦ содержимого файлов образов ВМ без необходимости внесения значительных изменений в архитектуру СЗИ.

Заключение

Получено решение одной из частей задачи КЦ ВМ в OpenStack — КЦ конфигураций образов ВМ. Описанная архитектура программного модуля реализует проверку конфигураций образов ВМ, хранимых в базе данных Glance DB платформы OpenStack, на соответствие эталону.

При развитии модуля для обеспечения КЦ конфигураций ВМ, хранимых в Nova DB, использование контрольных сумм в качестве эталона может оказаться не наилучшим вариантом, поскольку в отличие от образов для одной ВМ в системе значительно чаще может быть несколько разрешённых состояний. Данный вопрос необходимо исследовать отдельно. Возможным решением может стать применение атрибутных моделей контроля доступа к КЦ конфигураций ВМ [19].

Литература

1. Защита виртуальной инфраструктуры. Официальный сайт компании Код Безопасности [Электронный ресурс]. URL: https://www.securitycode.ru/upload/iblock/d0d/Virtualization_2018.pdf (дата обращения: 02.12.2020).

2. Image Signing and Verification Support: Blueprints: Glance [Электронный ресурс]. URL: https://blueprints.launchpad.net/glance/+spec/image-signing-and-verification-support (дата обращения: 02.12.2020).

3. OpenStack Docs: Glance Image Signing and Verification [Электронный ресурс]. URL: https://specs.openstack.org/openstack/glance-specs/specs/mitaka/implemented/image-signing-and-verification-support.html (дата обращения: 02.12.2020).

4. Glance Image creation checksum logic — Ask OpenStack: Q&A Site for OpenStack Users and Developers [Электронный ресурс]. URL: https://ask.openstack.org/en/question/90047/glance-image-creation-checksum-logic/ (дата обращения: 02.12.2020).

5. Bugs: Glance [Электронный ресурс]. URL: https://bugs.launchpad.net/glance/+bugs?field.tag=security (дата обращения: 02.12.2020).

6. *Мозолина Н. В.* Разработка средства контроля целостности виртуальной инфраструктуры и её конфигурации: выпускная квалификационная работа. 2017. С. 19—36.

7. *Мозолина Н. В.* Контроль целостности виртуальной инфраструктуры и ее конфигурации // Вопросы защиты информации. 2016. № 3. С. 31—33.

8. What is OpenStack? [Электронный ресурс]. URL: https://openstack.org/software (дата обращения: 02.12.2020).

9. *Журов П. М.* Разработка и исследование модели доступа к объектам облачных инфраструктур: выпускная квалификационная работа. 2019. С. 18—22, 77—81.

10. OpenStack Docs: Swift Architectural Overview [Электронный ресурс]. URL: https://docs.openstack.org/swift/latest/overview_architecture.html (дата обращения: 02.12.2020).

11. OpenStack Docs: OpenStack Block Storage (Cinder) documentation [Электронный ресурс]. URL: https://docs.openstack.org/cinder/latest/ (дата обращения: 02.12.2020).

12. OpenStack Docs: Welcome to Glance's documentation! [Электронный ресурс]. URL: https://docs.openstack.org/glance/latest/ (дата обращения: 02.12.2020).

13. OpenStack Docs: Glance database architecture [Электронный ресурс] URL: https://docs.openstack.org/glance/pike/contributor/database_architecture.html (дата обращения: 02.12.2020).

14. OpenStack Docs: Basic architecture [Электронный ресурс] URL: https://docs.openstack.org/glance/pike/contributor/architecture.html (дата обращения: 02.12.2020).

15. OpenStack Docs: Image service overview [Электронный ресурс] URL: https://docs.openstack.org/glance/latest/install/get-started.html (дата обращения: 02.12.2020).

16. OpenStack Docs: Tasks [Электронный ресурс]. URL: https://docs.openstack.org/glance/pike/admin/tasks.html (дата обращения: 02.12.2020).

17. OpenStack Docs: Using Glance's Metadata Definitions Catalog Public APIs [Электронный ресурс]. URL: https://docs.openstack.org/glance/pike/user/glancemetadefcatalogapi.html (дата обращения: 02.12.2020).

18. *Мозолина Н. В.* Задание эталона при контроле целостности конфигурации виртуальной инфраструктуры: сб. научных статей XII Междунар. науч.-техн. конф. "Новые информационные технологии и системы", Пенза. 23—25 ноября 2016. С. 219—225.

19. *Ерин Ф. М.* Построение шаблонов для решения задачи контроля целостности конфигурации на основе атрибутной модели контроля доступа // Вопросы защиты информации. 2018. № 3. С. 3—6.

20. *Миеттинен М., Хятёнен К.* Способ обеспечения целостности набора записей данных. Российский патент 2009 года RU 2351978 C2. Изобретение по МКП G06F11/08 [Электронный ресурс] URL: https://patenton.ru/patent/RU2351978C2_ (дата обращения: 02.12.2020).

21. OpenStack Docs: Libvirt — Nova Virtualisation Driver // Официальный сайт документации OpenStack [Электронный ресурс]. URL: https://docs.openstack.org/kolla-ansible/latest/reference/compute/libvirt-guide.html (дата обращения: 02.12.2020).

22. libvirt: Domain XML format [Электронный ресурс]. URL: https://libvirt.org/formatdomain.html (дата обращения: 02.12.2020).

23. libvirt: Hooks for specific system management [Электронный ресурс]. URL: https://libvirt.org/hooks.html (дата обращения: 02.12.2020).

24. Специальное программное обеспечение средств защиты информации от несанкционированного доступа "Аккорд-KVM". Руководство администратора безопасности информации [Электронный ресурс]. URL: https://www.okbsapr.ru/upload/iblock/786/786f40d485a08e160fca07f38fbd78e6.pdf (дата обращения: 02.12.2020).

25. OpenStack Docs: Glance domain model implementation [Электронный ресурс]. URL: https://docs.openstack.org/glance/latest/contributor/domain_implementation.html (дата обращения: 02.12.2020).

Integrity control of virtual machine images on the OpenStack platform

D. O. Stasyev

Moscow Institute of Physics and Technology (National Research University), Dolgoprudny, Moscow region, Russia

Despite the presence of built-in integrity control (IC) mechanisms, the OpenStack platform does not contain reliable solutions for IC of images and configurations of images of virtual machines (VM). For IC of configurations, it is necessary to use their comparison with some reference (standard). The article defines its format, includes a set of tables from Glance DB (OpenStack Glance is a component created for managing VM images). The events of interaction of the developed module with the reference are described. The result of the work is the architecture of a software module that checks the configurations of VM images for compliance with the reference.

Keywords: virtualization, virtual machine, OpenStack, Glance, virtual machine images, integrity, integrity control, integrity assurance, virtual machine components, software module.

Bibliography — 25 references. *Received July 29, 2021*