**Проблемы разграничения доступа при управлении KVM-гипервизором посредством libvirt**

Ружанская А. А., МФТИ

В статье рассматривается новый подход к разграничению доступа при удаленном управлении виртуальной инфраструктурой, построенной на базе KVM для ОС Linux, его плюсы и основные проблемы, возникшие при проектировании решения.

Ключевые термины: виртуальная инфраструктура, удаленное управление, разграничение доступа

The problems of access control development in a KVM based infrastructure using libvirt.

In this paper a novel approach for controlling a remote access to virtual infrastructure built on the basis of KVM hypervisor for Linux OS is proposed. The main advantages of the approach and some basic problems faced are described.

Keywords: virtual infrastructure, remote control, access control

Технологии виртуализации нашли широкое применение во многих компаниях и организациях. Одновременно с их появлением возникли такие новые угрозы безопасности информационных систем, как, например, НСД к объектам виртуальной инфраструктуры, неконтролируемый рост числа ВМ, повышение прав пользователя внутри ВМ. Для безопасного использования виртуальных инфраструктур в государственных учреждениях и компаниях необходимы СЗИ, а для прохождения аттестации аппаратных и программных комплексов, на которых установлены данные решения, используемые СЗИ должны быть сертифицированы (приказ ФСТЭК №17 [1]). Сертификации подлежит либо системное и программное обеспечение целиком со встроенным СЗИ, либо лишь наложенное СЗИ [2]. Второй вариант является наиболее предпочтительным для использования, так как высокая связность встроенного СЗИ и системного ПО влечет за собой усложнение и увеличение времени исследования программного кода, его анализа (при сертификации проверке подвергается весь программный код, объем которого в данном случае, в отличие от объема кода только наложенного СЗИ, может быть очень большим). Выпуск обновлений безопасности для СЗИ также происходит быстрее, так как его защитные функции не интегрированы в системное программное обеспечение и объем процедуры проверки со стороны испытательной лаборатории становится небольшим. На сегодняшний день разработаны решения для различных типов гипервизоров, в описанной в данной статье работе исследуется инфраструктура виртуализации, построенная на базе платформы KVM [3] для среды Linux. Целью представленной исследовательской работы является создание решения, позволяющего разграничить доступ к функциям управления рассматриваемой виртуальной инфраструктуры. Выбор KVM обусловлен следующими причинами. В настоящий момент на рынке лидирующие позиции занимают гипервизоры Hyper-V [5] и VMware [4], для которых уже существуют СЗИ ("Сегмент-В." ОКБ САПР [6], vGate Код Безопасности [7]) [8], для KVM же есть предпосылки развития: наблюдается в целом переход на ОС Linux в государственных учреждениях и компаниях, решения для KVM только появляются на рынке (ПК "ВИУ”, ПК "Брест" РусБИТех [9]) и имеется потенциал для внедрения собственных технологий.

 Все существующие сертифицированные либо реализованы для других гипервизоров, либо обладают рядом недостатков в модели разграничения доступа. Например, В ПК “ВИУ”, спроектированном для системы Астра Linux, реализованы дискреционная и мандатная модели разграничения доступа, однако отсутствует достаточная детализация прав, доступно всего три типа доступа к виртуальной машине: “Просмотр свойств”, “Использование”, “Администрирование”, при которых невозможно пользователю, например, запретить все, кроме запуска виртуальных машин и сохранения снэпшотов (snapshot).

Другой продукт, Сегмент-В., реализован для гипервизора VMware vSphere, но одним из его достоинств является большая гранулярность уровней доступа за счет введения иерархических меток – уровней, и категорий. Используемая концепция разграничения доступа позволяет различать непосредственные действия пользователя (субъекта) над различными объектами.

В продукте vGate также применяется иерархическая система меток безопасности к различным объектам (хранилище ВМ, виртуальная машина, физический сетевой адаптер, виртуальная локальная сеть, виртуальный коммутатор), однако в настоящее время существует аналогичная проблема - решения для KVM от производящей компании не поставляются. В еще одном решении HyTrust Control [10] – успешно используется схожая программная схема, которая будет описана в данной статье – прокси-сервер между клиентом и сервером, а также предоставляется возможность настраивать различные политики доступа к объектам виртуальной инфраструктуры на базе гипервизора VMware.

Рассмотрев особенности основных решений, фигурирующих на рынке, можно отметить, что действительно, их большая часть выпускается под такие продукты как VMware vSphere и Hyper-V, а спроектированные на данный момент для ОС Linux и KVM не обладают достаточной детализацией в плане политики разграничения доступа или выпускаются только под определенную реализацию Linux (как продукты РусБИТеха). Следовательно, имеются перспективы для разработки решения, которое в дальнейшем может быть сертифицировано, реализующего детализированную политику разграничения доступа и работающего на свободно распространяемых дистрибутивах Linux.

В статье описан подход к созданию технического решения, предназначенного для разграничения доступа к функциям управления KVM: в первой части описываются используемые технологии и протоколы, во второй – предложенная программная реализация механизма разграничения доступа и некоторые проблемы, которые необходимо учесть, третья же часть описывает саму модель разграничения доступа.

**Используемые технологии и их особенности**

Существует несколько решений, позволяющих управлять виртуальной инфраструктурой на основе гипервизора KVM, их можно разделить на управляющие через дополнительные сервера (oVirt [11], OpenNebula [13], OpenStack [12]) и напрямую единичными хостами (virsh, virt-manager). Общей чертой всех упомянутых решений является взаимодействие с гипервизором KVM при помощи библиотеки libvirt [14]. Поэтому в рамках данного проекта была выбрана графическая программа virt-manager, предоставляющая максимально упрощенный, но полноценный графический интерфейс.

Несколько слов также необходимо сказать о самой библиотеке и используемых протоколах перед дальнейшим описанием предложенного решения. Библиотека libvirt обеспечивает удаленное управление виртуальной инфраструктурой, предоставляет API для управления (локального или удаленного) виртуальными машинами, включающего создание, изменение, запуск, остановка, мониторинг и т.д. виртуальных машин, менеджмент хранилищ, дисков, памяти, сетевых настроек. Коммуникация между двумя машинами при удаленном управлении идет при помощи протокола RPC [15], используемого библиотекой libvirt. В пакете содержится информация об имени удаленно вызываемой процедуры, ее типе, порядковом номере (который впоследствии будет использоваться в создаваемых сообщениях об ошибке), а также статусе выполнения удаленного вызова.

 Основным форматом пересылаемых данных является XDR [16]. Одно из достоинств данного формата, что он может быть одинаково раскодирован на любой архитектуре, по определенному набору правил и, аналогично, любые данные могут быть с помощью него закодированы и переданы, к стандарту прилагается документация, описывающая расположение полей значений, их типы и т.д. Как следствие, становится легко анализировать пакеты, идущие от клиента к серверу или наоборот, и, зная, что передается в данном пакете по названию процедуры, т.е. в каком порядке идут какие значения в передаваемой структуре, смотреть на его содержание, последовательно раскодировав число за числом.

**Предложенное решение и возможные проблемы**

В результате исследования механизма передачи сообщений между клиентом и сервером было предложено следующее решение по способу разграничения доступа к функциям управления. В силу того, что внесение модификаций либо в код библиотек на сервере, либо в код библиотек на клиенте крайне нежелательно (как минимум по причинам дальнейшей поддержки продукта и невозможности масштабируемости), предлагается использовать отдельный прокси-сервер, который может находиться как на отдельной машине, контролирующей трафик между сервером и клиентом, так и на серверной машине.

Поскольку на данный момент уже существуют прокси-серверы, реализующие необходимый функционал, было принято решение использовать один из них. Mitmproxy [17] – прокси-сервер (поставляемый как библиотека на языке Python) позволяющий перехватывать, изменять, исследовать трафик. Аргументами в пользу выбора данной программы можно назвать полноценную поддержку сертификатов, реализацию работы c http сообщениями и с многопоточностью. Mitmproxy активно развивается, добавляются новые возможности в функционал. Из всех его режимов функционирования наиболее подходящим является режим прозрачного прокси, в котором от клиента не требуется никакой дополнительной настройки, весь трафик перенаправляется на прокси на сетевом уровне.

Схема возможной коммуникации между клиентом и сервером с использованием прокси представлена на Рисунке 1:



 Рисунок 1 Общая схема клиент-серверной связи с прокси-сервером

Как уже упоминалось, в каждом сообщении, пересылаемом демону libvirt, содержится название процедуры, которая должна исполнится удаленно. При этом, одно действие пользователя повлечет за собой вызов множества процедур. Для упрощения будем называть действия пользователя функциями верхнего уровня (ФВУ), а вызываемые при этом процедуры – функциями нижнего уровня (ФНУ). Для примера, при действии пользователя в графической оболочке virt-manager “Создать snapshot” удаленно вызываются процедуры “snapshot\_look\_up\_by\_name” (просмотр на наличие snapshot с присваиваемым именем), “snapshot\_create\_xml” (создание xml конфигурации snapshot), “screenshot” (создание скриншота экрана на момент создания snapshot), а также некоторые дополнительные функции. На первых этапах исследования была составлена таблица, в которой каждой ФВУ сопоставляются вызовы ФНУ для основных доступных пользователю действий в графическом окружении virt-manager.

Некоторые ФНУ состоят из запроса (CALL) и ответа (REPLY), некоторые только из вызова специального действия (передача данных (STREAM), вызов события (EVENT)). При этом несмотря на то, что в документации libvirt к RPC протоколу говорится о возможности посылки нескольких call-сообщений одновременно без ожидания ответа, существует множество случаев, когда это не выполняется. То есть, существует ряд функций, например, “Подключение” или “Клонирование”, в которых почти все call-сообщения для процедур будут вызваны только после того, как будет получен ответ на предыдущие. Поэтому конкретное действие, то же “Клонирование”, например, нельзя будет сопоставить с набором из 8 функций и, запретив этот набор, запретить вызов ФВУ “Клонирование” (в данном случае клонирование не будет произведено, пока не вернется информация о хранилище и пока libvirt не будет знать, достаточно ли места для этой операции или нет), что значительно осложняет реализацию. Невозможно это также и потому, что прокси, расшифровывая сообщения и считывая информацию, не знает какая в данный момент выполняется функция верхнего уровня. Нет способа понять, что при ожидании ответа на доступный размер хранилища, этот запрос был послан в рамках функции клонирования. Таким образом, необходимо создать некоторую **итеративную** схему разграничения доступа, единицей которой будут не действия пользователя, а вызываемые процедуры. Запрет какого-либо действия пользователя должен следовать из выполнения запрещенной ФНУ в данный момент времени.

Необходимо учитывать, что сообщения анализируются только после непосредственной отправки с клиента, при этом libvirt на клиенте либо встает в ожидание ответа, либо продолжает работу. Единственным способом вывести процесс из ожидания и заставить его завершиться является отправка ошибки. Необходима проработка и анализ ошибок, которые могли бы останавливать прерывать выполнение ФНУ, но (что очень важно) не закрывать текущее tcp-соединение. Сообщения об ошибках в libvirt реализованы с помощью обычных сообщений (приходят обычно как ответы, REPLY), в которых в поле status проставлено значение 1, что обозначает ошибку, а сам текст сообщения содержит определенные для ошибки поля [18].

В структуру ошибки входят следующие значения: код ошибки, область библиотеки, где возникла эта ошибка (чаще всего совпадает с названием файлов), ошибочное сообщение, уровень ошибки, а также некоторые дополнительные строковые поля. Недостатком спецификации libvirt является умолчание о дополнительных единицах (закодированных как тип int), которые разграничивают эти строковые поля. В сообщениях имеется возможность заменять текст ошибки на собственный. При этом также нужно менять длину сообщения, которая кодируется в общей структуре ошибки. Так как каждой ФНУ сопоставляется свой набор возможных кодов ошибок и мест их возникновения, то необходимо составить полное соответствие между имеющимися ФНУ и соответствующих им параметров ошибок. Используя его, можно создавать различные валидные сообщения об ошибках, меняя лишь сам текст сообщения ошибки.

Таким образом механизм разграничения доступа для определенной ФНУ включает в себя следующие этапы: необходимо перехватить сообщение (задержать, именно для этого и нужна функциональность mitmproxy), раскодировать его согласно стандарту XDR, проверить соответствие номера процедуры разрешенным номерам для данного пользователя и объекта, определить порядковый номер сообщения, определить необходимые параметры для формирования ошибки, закодировать структуру ответа с таким же порядковым номером, отослать клиенту структуру с ошибкой, выйти из обработчика.

Как же решается проблема с необходимостью итеративного разграничения и как определить, на какие именно функции ФНУ следует отправлять сообщение об ошибке? Для этого предлагается следующая схема. Для каждой ФВУ существует ФНУ, уникально ее идентифицирующая, т.е. характеризующая функция (ХФ). Это утверждение было проверено практическим путем для всех основных ФВУ, доступных из интерфейса virt-manager для пользователя. Имея ХФ для каждого действия пользователя, простым решением задачи была бы передача трафика до тех пор, пока не встретилась запрещенная характеризующая функция. Однако в наборе доступных ФНУ имеется большое множество функций, которые не характеризуют ни одно действие пользователя, они зачастую выполняют операции чтения, реагирования на события, операции записи, не влияющие на состояние системы в целом. Все такие функции при упомянутом простом подходе остаются неразграниченными, и если, например, выполнение ФВУ начинается с REMOTE\_PROC\_DOMAIN\_SNAPSHOT\_GET\_XML\_DESC, то данная операция чтения будет исполнена перед ХФ, и пользователь получит доступ к запрещенным для него данным о снэпшоте. Но можно и не дожидаться исполнения ХФ, которое может произойти и в середине, и в конце действия. Для этого предлагается запрещать все ФНУ, не входящие в список разрешенных.

Чтобы сформировать такой список, будем ассоциировать с каждой ФНУ определенный счетчик (целочисленную переменную). Счетчик будет показывать, разрешена операция или нет. Если счетчик больше нуля — значит разрешена, если равен нулю – запрещена. По умолчанию он равен нулю. При разрешении ФВУ увеличиваем на единицу счетчик каждой ФНУ, входящей в это действие, для соответствующего объекта. Например, в наборе ФНУ есть функция REMOTE\_PROC\_DOMAIN\_GET\_XML\_DESC. Увеличив счетчик на 1 только для, например, ФВУ “Создание snapshot”, невозможно будет достигнуть правильного поведения на прокси-сервере, так как анализируя приходящее сообщение, нельзя будет установить, что чтение xml-данных происходит именно в рамках создания snapshot. Следовательно, необходимо разрешить функцию REMOTE\_PROC\_DOMAIN\_GET\_XML\_DESC для всех действий пользователя. Этот шаг не расширяет права пользователя, так как если пользователь имеет право просматривать какие-либо xml-данные в рамках выполнения одной функции, он потенциально может всегда получить к ним доступ. В целом проверка счетчиков осуществляется следующим образом: ФНУ отсылаются на сервер, пока не встретится функция, у которой счетчик равен нулю. Тогда клиенту отправляется сообщение об ошибке, соответствующее функции, на которой остановилось исполнение. Этой функцией всегда будет одна из характеризующих функций. Запрет ФВУ происходит аналогично, но только путем уменьшения счетчика. Все ФНУ применяются к каким-либо объектам, соответственно счетчики также применяются одновременно для определенного объекта и для применяемой к нему функции. Объекты можно определить по префиксу ФНУ (тип объекта) и по параметрам, подающимся к ней на вход. Если, например, в префиксе присутствует приставка DOMAIN, это означает, что объект, к которому относится рассматриваемая функция, является виртуальной машиной, UUID виртуальной машины можно определить по передаваемой на вход структуре. Аналогично и другие объекты имеют свои имена или UUID.

Такой механизм позволит свести к минимуму количество исполненных ФНУ для запрещённой ФВУ (т.е. запретить ФВУ и отправить сообщение об ошибке максимально быстро).

Во всех приведенных выше описаниях фигурировал один пользователь. Нерешенной проблемой на текущий момент остается отсутствие идентификатора сессии в сообщении, который бы позволил легко отслеживать пользователя в сообщении, тем самым позволяя различать их между собой. Имеется возможность определять отправителя по ip–адресу, но с одного компьютера могут регистрироваться много пользователей. Авторизация в virt-manager настраивается либо при помощи tls/ssl, либо при помощи SASL. Однако, регистрируя на прокси-сервере необходимые сертификаты для расшифровки трафика, мы устанавливаем соответствие снова между ip-адресом компьютера и сертификатом.

Таким образом, были представлены основные положения и особенности предложенного механизма разграничения доступа при работе с KVM через libvirt, подробно описан протокол взаимодействия клиентской и серверной частей. Выявленная особенность с итеративными удаленными вызовами в рамках выполнения одного действия, а также проблема с сессиями делает реализацию более сложной, чем в случае, если бы информация о пользователях содержалась в пересылаемых сообщениях. В дальнейшем необходимо сосредоточиться на программах с дополнительными серверами, так как они уже обладают собственной подсистемой управления пользователями, что позволит устранить вторую проблему, не повлияв на предложенные основные аспекты решения, заключающиеся в разделении на ФВУ и ФНУ, а также отправке ошибок. Планируется проработка модели для обеспечения сценария со многими субъектами и объектами, с возможностью поддерживать объединение субъектов в группы. В рамках этого необходимо решить проблему с отсутствием сессионной информации.

Список литературы:

1. Приказ ФСТЭК № 17 «Об утверждении требований о защите информации, не составляющей государственную тайну, содержащейся в государственных информационных системах». [Электронный ресурс] URL: <http://fstec.ru/normotvorcheskaya/akty/53-prikazy/702-prikaz-fstek-rossii-ot-11-fevralya-2013-g-n-17> (дата обращения: 07.04.2018).
2. Лыдин С. С. О проблеме выбора средств защиты информации для инфраструктуры виртуализации // Вопросы защиты информации, 2017. № 3. С. 42—45.
3. Официальный сайт KVM. [Электронный ресурс] URL: <https://www.linux-kvm.org/page/Main_Page> (дата обращения: 07.04.2018).
4. Статья Gartner о VMware. [Электронный ресурс] URL: <http://ir.vmware.com/overview/press-releases/press-releasedetails/2016/VMware-Named-a-Leader-in-2016-Magic-Quadrant-for-x86-Server-VirtualizationInfrastructure/default.aspx> (дата обращения: 07.04.2018).
5. Документация Windows к Hyper-V. [Электронный ресурс] URL: <https://docs.microsoft.com/ru-ru/virtualization/hyper-v-on-windows/about/> (дата обращения: 07.04.2018).
6. Сегмент-В. [Электронный ресурс] URL: <http://www.accord.ru/segment-v.html> (дата обращения: 07.04.2018).
7. vGate. [Электронный ресурс] URL: <https://www.securitycode.ru/products/vgate/> (дата обращения: 07.04.2018).
8. Государственный реестр сертифицированных средств защиты информации N РОСС RU.0001.01БИ00. [Электронный ресурс] URL: https://fstec.ru/tekhnicheskaya-zashchita-informatsii/dokumenty-po-sertifikatsii/153-sistema-sertifikatsii/591-gosudarstvennyj-reestr-sertifitsirovannykh-sredstv-zashchity-informatsii-n-ross-ru-0001-01bi00 (дата обращения: 07.04.2018).
9. ПК “ВИУ”, ПК “БРЕСТ” РусБИТех. [Электронный ресурс]

URL: <http://www.astralinux.com/products/virt.html> (дата обращения: 07.04.2018).

1. HyTrust Control [Электронный ресурс] URL: <https://www.hytrust.com/solutions/cloud-compliance/pci-dss/> (дата обращения: 07.04.2018).
2. Официальная страница oVirt. [Электронный ресурс] URL: <https://ovirt.org/> (дата обращения: 07.04.2018).
3. OpenStack. [Электронный ресурс] URL: <https://www.openstack.org/> (дата обращения: 07.04.2018).
4. OpenNebula. [Электронный ресурс] URL: <https://opennebula.org/> (дата обращения: 07.04.2018).
5. Официальная документация библиотеки libvirt. [Электронный ресурс] URL: <https://libvirt.org> (дата обращения: 07.04.2018).
6. Протокол RPC. [Электронный ресурс] URL: <https://libvirt.org/internals/rpc.html> (дата обращения: 07.04.2018).
7. Стандарт XDR. [Электронный ресурс] URL: <https://tools.ietf.org/html/rfc1832> (дата обращения: 07.04.2018).
8. Официальная страница mitmproxy. [Электронный ресурс] URL: <https://mitmproxy.org/> (дата обращения: 07.04.2018).
9. Спецификация ошибок libvirt. [Электронный ресурс]

URL: <https://libvirt.org/html/libvirtvirterror.html#virErrorDomain> (дата обращения: 07.04.2018).