УПРАВЛЕНИЕ ДОСТУПОМ

УДК 004.056

Разграничение доступа в Linux при использовании средства виртуализации kvm

А. М. Каннер
ЗАО «ОКБ САПР», Москва, Россия

Рассматриваются особенности разграничения доступа в базовой операционной системе Linux при использовании kvm для различных способов инициализации виртуальных машин. Подробно описываются последовательности доступов субъектов к объектам на чтение/запись/выполнение в базовой операционной системе в ходе запуска виртуальных машин, а также имеющиеся при этом особенности, которые необходимо учитывать в средствах защиты информации от несанкционированного доступа.

Ключевые слова: виртуализация, kvm, Аккорд-X, Аккорд-KVM.

Виртуализация — это крайне популярная концепция, истоки которой берут начало еще в 60-х гг. XX в. Одним из наиболее распространенных средств виртуализации является kvm (Kernel Virtual Machine), позволяющее полноценно эмулировать большое количество программных и аппаратных платформ [1].

Использование виртуализации, в том числе kvm, с одной стороны, упрощает и удешевляет работу с различными программными и аппаратными платформами [1], а с другой стороны — создает еще два новых уровня дополнительно к уровню операционной системы (в рамках виртуальной машины), на которых необходимо защищать информационную систему и хранящуюся/обрабатываемую в ней информацию. Такими новыми уровнями являются базовая операционная система (с гипервизором, в рамках которого запускаются все виртуальные машины) и среда виртуализации, в которой одна виртуальная машина теоретически может оказывать влияние на другую машину или (что еще хуже) на базовую операционную систему [2].

Таким образом, использование средств виртуализации влечет за собой предъявление новых требований и ограничений (см., например [3, 4]) на работу различных средств защиты информации, которые используются в базовой операционной системе. Однако известные исследования в области защиты систем виртуализации либо ограничиваются вопросами контроля целостности [5, 6], либо рассматривают только второй уровень — разграничение доступа для субъектов и виртуальных машин в рамках среды виртуализации [2, 7, 8]. В данной статье рассмотрены особенности, которые необходимо учитывать в средствах разграничения доступа при применении kvm в операционной системе семейства Linux на уровне базовой операционной системы.

Каннер Андрей Михайлович, программист группы программирования ПО для СЗИ отдела программирования СЗИ.
E-mail: kanner@okbsapr.ru

Статья поступила в редакцию 19 июля 2019 г.

Материалы и методы

Необходимо отметить, что kvm нельзя назвать полноценным средством виртуализации, т. к. в основе виртуализации с использованием kvm применяется сразу несколько компонентов [9]:

- процессоры, поддерживающие на аппаратном уровне специальный набор команд для виртуализации (Intel VT-x, AMD-V);
- драйверы (модули ядра kvm.ko, kvm_intel.ko, kvm_amd.ko и другие) для создания инфраструктуры виртуализации с интерфейсом в виде специального устройства /dev/kvm;
- компоненты пользовательского уровня для эмулирования различных устройств, ввода/вывода и т. д. (процесс qemu, Quick Emulator).

Разработчики kvm вместо создания собственного гипервизора (со своей подсистемой памяти, сетевой подсистемой, планировщиком, вводом/выводом и т. д.) решили использовать существующие нара-

ботки ядра Linux, "превращая" его в гипервизор при загрузке соответствующих модулей. Каждый компонент из приведенного списка отвечает за определенные возможности (способность виртуализировать процессор, память, ввод/вывод и прочие устройства), которые в целом и позволяют запускать виртуальные машины с использованием kvm.

Непосредственно kvm обеспечивает виртуализацию памяти с помощью /dev/kvm. При этом каждая виртуальная машина имеет свое собственное адресное пространство — "физическую" память, являющуюся в действительности виртуальной памятью процесса операционной системы. Данное адресное пространство независимо от адресного пространства ядра или любых других работающих виртуальных машин, а сам процесс находится в гостевом режиме [1, 9], который работает независимо от пользовательского режима и режима ядра основной операционной системы. В рамках гостевого режима существуют два стандартных, для того чтобы гостевая операционная система поддерживала режим ядра и пользовательский режим в виртуальной машине. Аппаратные устройства являются общими для всех процессов, но каждая виртуальная машина видит другую "карту" этих устройств для поддержания изоляции. С использованием /dev/kvm также доступен ряд ioctl-вызовов для всех основных операций с виртуальной машиной: создания виртуального процессора (KVM_CREATE_VCPU), проверки версии kvm (KVM_GET_API_VERSION), создания области памяти (KVM_SET_USER_MEMORY_REGION), выполнения инструкции на виртуальном процессоре (KVM_RUN) и т. д.

Выполнение операций ввода/вывода с виртуальной машиной обеспечивается средством qemu, которое также позволяет виртуализировать/эмулировать все аппаратные составляющие (носители информации, графические адаптеры, сетевые устройства и т. д.). Любые запросы ввода/вывода, производимые из виртуальной машины, перехватываются и направляются в пользовательский режим процессу qemu. Необходимо отметить, что qemu может эмулировать множество программных и аппаратных платформ без участия kvm или возможностей современных процессоров за счет бинарной трансляции инструкций одной платформы на другую. При этом достигается достаточно слабая производительность, однако данная возможность позволяет эмулировать на одной физической архитектуре любую другую. При использовании kvm с помощью qemu возможна более эффективная виртуализация, но только той архитектуры, которая используется на физическом средстве вычислительной техники.

Каждая гостевая операционная система в рамках своей виртуальной машины на уровне базовой операционной системы является обычным процессом, выполняемым с копией процесса qemu [9]. Для каждого виртуального процессора создается поток (англ. *thread*). Эти процессы и потоки не отличаются от аналогичных сущностей базовой операционной системы и планируются с использованием стандартного планировщика Linux. Соответственно и в плане разграничения доступа процессы виртуальных машин будут рассматриваться как обычные процессы операционной системы, для них будут применяться стандартные правила изоляции и контроля ресурсов системы. В соответствии с этим возникает вопрос: *требуется ли что-то дополнительно учитывать в базовой операционной системе при использовании виртуализации kvm в плане разграничения доступа?*

Чтобы ответить на этот вопрос, необходимо более подробно рассмотреть процесс запуска виртуальных машин с использованием kvm, провести эксперимент, в ходе которого детально изучить, что происходит с процессами базовой операционной системы. На практике обычно применяют два основных способа запуска виртуальных машин с использованием kvm:

- вручную с помощью утилит qemu-system-{i386,x86_64,s390x,…};
- автоматически/полуавтоматически с использованием определенной среды управления виртуальными машинами (находит более частое применение в корпоративной среде).

Непосредственно сам эксперимент будет проводиться в базовой операционной системе Linux (CentOS 7.5 x86_64) с установленным средством защиты информации от несанкционированного доступа "Аккорд-X" [10] для выявления субъектов доступа в процессе запуска виртуальных машин.

Результаты

Различные среды управления виртуальными машинами работают с kvm схожим образом, и все в конечном итоге используют утилиты qemu-system-{i386,x86_64,s390x,…}, однако поведение процессов базовой операционной системы для них по сравнению с ручным запуском этих утилит отличается. Далее в качестве среды управления виртуальными машинами рассмотрим libvirt.

При ручном запуске все виртуальные машины работают в контексте процессов с правами субъекта-пользователя, инициировавшего запуск. При этом последовательность доступов субъектов к объектам на чтение/запись/выполнение в базовой операционной системе выглядит следующим образом (в строке слева направо: имя субъекта, от имени которого осуществляется доступ; путь к бинар-

ному файлу процесса, который осуществляет доступ; путь к объекту доступа):

user root */usr/bin/bash* */usr/bin/qemu-system-x86_64*

...

// происходит доступ к разделяемым библиотекам qemu-system-x86_64

user root */usr/bin/qemu-system-x86_64* */usr/share/seabios/bios-256k.bin*

...

user root */usr/bin/qemu-system-x86_64* */dev/kvm*

user root */usr/bin/qemu-system-x86_64* */debian.qcow2*

user root */usr/bin/qemu-system-x86_64* */usr/share/qemu/kvmvapic.bin*

user root */usr/bin/qemu-system-x86_64* */usr/share/qemu/linuxboot.bin*

user root */usr/bin/qemu-system-x86_64* */usr/share/seavgabios/vgabios-cirrus.bin*

user root */usr/bin/qemu-system-x86_64* */usr/share/ipxe/1af41000.rom*

...

Стандартные средства разграничения доступа Linux в данном случае накладывают только одно ограничение: пользователь должен состоять в группе qemu/kvm (зависит от версии qemu и базовой операционной системы), чтобы иметь возможность доступа к соответствующим компонентам средств qemu и kvm. На этом примере наглядно видно, что все доступы осуществляются только от имени реального пользователя (user root), который инициировал запуск виртуальной машины.

При использовании среды управления виртуальными машинами libvirt последовательность доступов сложнее. Утилита virsh позволяет пользователям, входящим в группу libvirt и kvm, запускать виртуальные машины с использованием специального сервиса libvirtd, который, в свою очередь, активизируется процессами системы инициализации systemd на современных операционных системах Linux (или аналогичными системами в более ранних версиях операционной системы). При этом последовательность доступов субъектов к объектам на чтение/запись/выполнение в базовой операционной системе выглядит следующим образом (количество процессов и объектов доступа — разделяемых библиотек сокращено для большей наглядности):

 # запуск сервиса libvirtd от имени псевдопользователя shadow с UID=0 (root)

shadow root /usr/lib/systemd/systemd /usr/lib/sysctl.d/60-libvirtd.conf

shadow root /usr/lib/systemd/systemd /usr/lib/systemd/system/libvirtd.service

shadow root /usr/lib/systemd/systemd /usr/lib/systemd/system/libvirt-guests.service

shadow root /usr/lib/systemd/systemd /etc/sysconfig/libvirtd

shadow root /usr/lib/systemd/systemd /sys/fs/cgroup/devices/system.slice/libvirtd.service

/{cgroup.procs,devices.allow,pids.max}

shadow root /usr/lib/systemd/systemd /etc/libvirt/libvirtd.conf

shadow root /usr/lib/systemd/systemd /usr/sbin/libvirtd

shadow root /usr/lib/systemd/systemd /run/libvirtd.pid

...

 # загрузка других модулей и зависимых сервисов, например сетевых

shadow root /usr/lib/systemd/systemd /var/lib/libvirt/dnsmasq/.conf*

shadow root /usr/lib/systemd/systemd /usr/sbin/dnsmasq

*shadow root /usr/lib/systemd/systemd /usr/lib/modules/.../kernel/drivers/virtio/**

shadow root /usr/bin/udevadm /sys/module/virtio{,_net,_blk,_pci,_ring}/

...

 # запуск виртуализируемой сети, которая доступна виртуальным машинам

shadow root /usr/lib/systemd/systemd /etc/libvirt/qemu/networks/default.xml

shadow root /usr/lib/systemd/systemd /usr/sbin/ip

shadow root /usr/lib/systemd/systemd /dev/net/tun

*shadow root /usr/lib/systemd/systemd /sys/devices/virtual/net/virbr0/**

shadow root /usr/lib/systemd/systemd /sys/devices/virtual/misc/tun

...

 # запуск средства управления виртуальными машинами пользователем root

user root ***/usr/bin/bash*** ***/usr/bin/virsh***

shadow root*/usr/sbin/libvirtd* */etc/libvirt/libvirt.conf*

...

 # проверка вхождения пользователя в группу libvirt

shadow libvirtd /usr/sbin/libvirtd /etc/{passwd,group}

 # взаимодействие с qemu (настройка ввода/вывода и др.)

shadow libvirtd /usr/sbin/libvirtd /sys/devices/platform/i8042/serio1

shadow libvirtd /usr/sbin/libvirtd /sys/devices/platform/serial8250/

...
shadow libvirtd /usr/sbin/libvirtd
/etc/libvirt/qemu.conf
shadow libvirtd /usr/sbin/libvirtd
/run/libvirt/qemu
shadow libvirtd /usr/sbin/libvirtd
/run/libvirt/hostdevmgr
shadow libvirtd /usr/sbin/libvirtd
/usr/share/libvirt/cpu_map.xml
shadow libvirtd /usr/sbin/libvirtd
/dev/hugepages/libvirt/qemu
...
shadow libvirtd /usr/sbin/libvirtd
/etc/libvirt/qemu/debian-test.xml
shadow libvirtd /usr/sbin/libvirtd
/run/libvirt/qemu/debian-test.pid
shadow libvirtd /usr/sbin/libvirtd
/run/libvirt/qemu/debian-test.xml.new
...
 # непосредственно запуск виртуальной машины как процесса от имени libvirtd
shadow libvirtd */usr/sbin/libvirtd*
/usr/bin/qemu-system-x86_64
...
// происходит доступ к разделяемым библиотекам qemu-system-x86_64
shadow libvirtd */usr/bin/qemu-system-x86_64*
/dev/kvm
shadow libvirtd */usr/bin/qemu-system-x86_64*
/debian.qcow2
shadow libvirtd */usr/bin/qemu-system-x86_64*
/usr/share/seabios/bios-256k.bin
shadow libvirtd */usr/bin/qemu-system-x86_64*
/usr/share/qemu/kvmvapic.bin
shadow libvirtd */usr/bin/qemu-system-x86_64*
/usr/share/seavgabios/vgabios-cirrus.bin
...

Из приведенной последовательности доступов субъектов к объектам базовой операционной системы видно, что запуск виртуальной машины инициирует пользователь root (user root), однако сама виртуальная машина выполняется в контексте процесса от имени псевдопользователя libvirtd (shadow libvirtd), который, в свою очередь, активизируется псевдопользователем root (shadow root). Как было показано в [11], процессы субъекта shadow root представляют собой процессы с PID 0 и 1 (самые первые процессы операционной системы), а также все их дальнейшие дочерние процессы, не являющиеся процессами других пользователей или псевдопользователей системы. Таким образом, все виртуальные машины работают в контексте процессов субъекта libvirtd и далее никак не ассоциируются с реальными пользователями системы, инициировавшими их запуск.

В рамках разграничения доступа в libvirt существует возможность создания индивидуальных пространств имен для процессов разных виртуальных машин. Однако на данный момент поддерживается создание только пространства имен mount, что не позволяет виртуальным машинам взаимодействовать с устройствами (в том числе носителями информации) других виртуальных машин, а также с устройствами базовой операционной системы. Большим недостатком в данном случае является то, что для полноценной изоляции различных виртуальных машин недостаточно только создания индивидуальных пространств имен mount, в libvirt необходимо реализовать возможность создание пространств pid, net, ipc и, возможно, других.

Таким образом, в части разграничения доступа описанные способы запуска виртуальных машин с использованием kvm представляют собой противоположные друг другу ситуации: слишком много полномочий (доступ к разделяемым библиотекам и компонентам среды виртуализации, *доступ на запись* ко *всем* образам виртуальных машин и т. д.) должно выдаваться либо реальным пользователям, либо сервисам базовой операционной системы. Поскольку изначально сервисам базовой операционной системы и так предоставляются широкие полномочия, использование средств управления виртуальными машинами позволяет с большей степенью соответствовать принципу наименьших привилегий [12] и не выдавать пользователям системы лишних, ненужных для повседневной работы полномочий и прав доступа. Однако в части изолированности процессов виртуальных машин ручной запуск лучше использования средств управления за счет выполнения процессов от имени разных субъектов доступа (реальных пользователей). Кроме того, при использовании средств управления виртуальными машинами нельзя ассоциировать, какой пользователь запустил ту или иную виртуальную машину, а процессы разных виртуальных машин, запущенные разными пользователями, идентичны с точки зрения системы (выполняются от имени сервиса libvirtd).

Обсуждение

Указанные особенности нужно учитывать в том числе при использовании таких средств защиты информации от несанкционированного доступа, как "Аккорд-X" [10] и "Аккорд-KVM" [5]. При этом в ряде случаев может потребоваться модификация стандартных средств управления виртуальными машинами. Хорошей демонстрацией такой необходимости является существующий западный опыт в области защиты среды виртуализации

kvm — SELinux (Security-Enhanced Linux) и sVirt (secure virtualization) [1, 13].

SELinux применяется как средство дополнительной защиты от несанкционированного доступа в Linux и позволяет реализовать мандатную политику управления доступом, а также ряд возможностей по строгой изоляции и ограничению процессов операционной системы, в том числе и процессов виртуальных машин. Процессы виртуальных машин с применением SELinux и sVirt автоматически запускаются в разных доменах (т. е. с разным уровнем доступа и/или меткой конфиденциальности), а также наследуют метки от объектов (образов виртуальных машин) и субъектов, которые инициировали их запуск. Таким образом, с использованием SELinux и sVirt достигается невозможность влияния процессов одной виртуальной машины на другую или на объекты базовой операционной системы, а также учитываются описанные особенности разграничения доступа при использовании kvm.

Заключение

Рассмотрены особенности разграничения доступа в операционных системах Linux при использовании средства виртуализации kvm. Стандартные средства разграничения доступа и среды управления виртуальными машинами в Linux не позволяют обеспечить надежную защиту систем, применяющих средства виртуализации. Различные способы инициализации виртуальных машин имеют свои недостатки в плане разграничения доступа, а некоторые стандартные средства управления виртуальными машинами необходимо модифицировать в целях обеспечения невозможности влияния процессов одной виртуальной машины на другую или на объекты базовой операционной системы. Рассмотренные особенности разграничения доступа необходимо учитывать при разработке соответствующих программных или программно-аппаратных средств защиты информации.

Литература

1. *Wilson G., Day M., Taylor B.* KVM: Hypervisor Security You Can Depend On. IBM Linux Technology Center [Электронный ресурс]. URL: ftp://public.dhe.ibm.com/linux/pdfs/LXW03004-USEN-00.pdf (дата обращения: 24.06.2019).
2. *Dong Y., Lei Z.* An Access Control Model for Preventing Virtual Machine Hopping Attack // Future Internet. 2019. No. 11 (3). P. 82. DOI: https://doi.org/10.3390/fi11030082
3. Российская Федерация. Приказы. Об утверждении состава и содержания организационных и технических мер по обеспечению безопасности персональных данных при их обработке в информационных системах персональных данных. Приказ ФСТЭК России № 21: издан ФСТЭК России 18.03.2013. — М., 2013. — 20 с.
4. Российская Федерация. Приказы. Об утверждении требований по защите информации, не составляющей государственную тайну, содержащейся в государственных информационных системах. Приказ ФСТЭК России № 17: издан ФСТЭК России 11.03.2013. — М., 2013. — 37 с.
5. *Мозолина Н. В.* Необходимо и достаточно, или контроль целостности виртуальных машин с помощью Аккорд-KVM // Информационная безопасность. 2018. № 5. С. 33.
6. *Мозолина Н. В.* Контроль целостности виртуальной инфраструктуры и ее конфигурации // Вопросы защиты информации. 2016. № 3. С. 31—33.
7. *Ружанская А. А.* Особенности разграничения доступа при управлении виртуальной инфраструктурой на базе гипервизора KVM // Вопросы защиты информации. 2018. № 2. С. 25—29.
8. *Конявская С. В., Угаров Д. В., Постоев Д. А.* Инструмент контроля доступа к средствам управления виртуальной инфраструктурой // Информационная безопасность. 2016. № 2. С. 9.
9. *Jones M. T.* Discover the Linux Kernel Virtual Machine. IBM DeveloperWorks [Электронный ресурс]. URL: https://www.ibm.com/developerworks/ru/library/l-linux-kvm/index.html (дата обращения: 24.06.2019).
10. *Каннер А. М., Ухлинов Л. М.* Управление доступом в ОС GNU/Linux // Вопросы защиты информации. 2012. № 3. С. 35—38.
11. *Каннер А. М.* Linux: о жизненном цикле процессов и разграничении доступа // Вопросы защиты информации. 2014. № 4. С. 37—40.
12. Trusted Computer System Evaluation Criteria. 1985. Ft. Meade, Md.: Dept. of Defense, Computer Security Center. DOD 5200.28-STD (supersedes CSC-STD-001-83).
13. *Morris J.* sVirt: Hardening Linux Virtualization with Mandatory Access Control [Электронный ресурс]. URL: http://namei.org/presentations/svirt-lca-2009.pdf (дата обращения: 24.06.2019).

Access control in Linux when using kvm virtualization

A. M. Kanner

Closed Joint Stock Company "OKB SAPR", Moscow, Russia

The article discusses the features of access control in the base Linux operating system when using kvm for various methods of initializing virtual machines. It describes in detail the sequence of accesses for reading/writing/executing of subjects to objects in the base operating system during the launch of virtual machines, as well as the features arising from this, which must be taken into account in the data security tools protecting against unauthorized access.

Keywords: virtualization, kvm, Accord-X, Accord-KVM.

Bibliography — 13 references. *Received July 19, 2019*