Лабораторная работа №16. Системы виртуализации в среде ОС Linux. Наблюдение и аудит в ОС Linux.

Цель работы: А) Получение навыков использования систем виртуализации в среде ОС Linux, создать виртуальную машину и выполнить запуск с различными параметрами.

В) Получение навыков наблюдения и отслеживания системных сообщений в ОС Linux.

Теоретические сведения

А) Эмуляция. QEMU

QEMU - система эмуляции и виртуализации вычислительной системы с процессором, памятью и периферийными устройствами, поддерживающая различные архитектуры, являющаяся проектом с открытым исходным кодом.

В строгом понимании QEMU является эмулятором виртуальных машин, то есть позволяет эмулировать работу процессора определенной архитектуры при работе на другой архитектуре посредством двоичной трансляции (именно от этого и происходит название системы: QEMU Quick EMUlator). При этом последовательности инструкций переводятся из исходного набора (source) в целевой (target) набор инструкций вне зависимости от возможностей оборудования.

Двоичная трансляция делится на:

- Статическую. При статической трансляция весь исходный исполняемый файл транслируется в исполнимый файл для целевой архитектуры. Эта задача корректно решается не всегда, так как не весь код сразу может быть обработан транслятором.

- Динамическую. Рассматривает короткие последовательности кода (обычно блок кода: цикл или функция), транслирует его и кэширует результат (размер кеша для QEMU 16Mb). Код транслируется не весь, апо мере считывания. В результате, благодаря кешированию достигается ускорение исполнения повторно вызываемых участков кода, и весь код доступен транслятору по мере выполнения любых операций ветвления.

Виртуализация. KVM

QEMU является одним из наиболее быстрых программных динамических трансляторов в мире. Однако эмуляция в любом случае приводит к большому количеству накладных расходов при трансляции кода. Кроме эмуляции для создания виртуальной вычислительной среды применяется **виртуализация**, которая не позволяет исполнять код сторонних архитектур, а применяется только для логического разделения вычислительных ресурсов между несколькими ОС.

Виртуализация также делится на два вида:

- Паравиртуализация

Ядро гостевой ОС и гипервизор модифицируются таким образом, чтобы повысить эффективность взаимного исполнения: гипервизор предоставляет специальный API, а ядро ОС использует предоставляемые функции. При этом каждая паравиртуализируемая ОС знает, что она исполняется в виртуализированной среде.

- Полная виртуализация

Ядро ОС функционирует без модификаций. При этом гостевая ОС не может определить является ли она единственной ОС на данном оборудовании и исполняется ли она в виртуальной среде или нет.

Самым эффективным методом для создания виртуальной вычислительной среды является паравиртуализация, т.к. согласованное исполнение нескольких гостевых ОС и гипервизора имеет наименьшие накладные расходы и позволяет выполнять часть системных операций максимально эффективно. Однако данный метод не применим ко всем ОС по крайней мере по причине лицензионных ограничений, закрытости исходного кода, а также его реализация усложняется из-за особенности архитектуры ядра некоторых ОС. Полная виртуализация является более универсальным методом.

Для реализации виртуализации в Linux было разработано программное решение KVM (Kernel-based Virtual Machine), которое является частью ядра Linux и обеспечивает полную виртуализацию при поддержке оборудования.

Система виртуализации состоит из двух частей:

- загружаемого модуля ядра (kvm.ko + (kvm-intel или kvm-amd))
- приложения пользовательского режима, осуществляющего контроль за ВМ

QEMU & KVM

В ходе развития системы было принято решение использовать в качестве компоненты пользовательского режима QEMU, т.к. к тому моменту он уже содержал все необходимые функции по эмуляции оборудования и являлся ПО с открытым исходным кодом.

В результате внедрения поддержки KVM в QEMU он получил возможность работать в двух режимах:

- эмуляции: без дополнительных опций
- виртуализации: с опцией -enable-kvm

Для удобства применения, а также по причинам, связанным с особенностями распространения ПО в Linux в т.н. пакетах, во многих дистрибутивах на сегодняшний день присутствует две команды:

- qemu , о которой рассказано в предыдущем абзаце
- kvm , которая на самом деле является модифицированным qemu , в котором по умолчанию запускается ре жим виртуализации, а чтобы перейти в режим эмуляции (и отключить поддержку KVM) используется опция -no-kvm

На компьютерах для выполнения лабораторных работ может бытьустановлена как одна, так и другая команда.

Применение

Эмулятор qemu можно установить принятым в дистрибутиве способом. Например, для Debian GNU/Linux:

```
# apt-get install qemu
```

Элементарные операции доступны сразу же после установки. Например, запустить в эмуляторе LiveCD:

```
# qemu -cdrom knoppix.iso
```

QEMU - эмулятор, который может работать и без KVM, но использование аппаратной виртуализации значительно ускоряет работугостевых систем.

Ускорение **KVM** поддерживается далеко не везде, для его успешного применения необходимо:

- Поддержка процессором технологии виртуализации. Выполните следующую команду, чтобы узнать поддерживает ли процессор аппаратную виртуализацию

```
egrep -c ' (vmx | svm) ' / proc / cpuinfo
```

- Включение соответствующей опции в BIOS системы
- Наличие необходимых модулей в ядре. Выполните команду `lsmod` , чтобы узнать загружены ли необходимые модули.

Для определения поддержки гипервизора kvm на вашем оборудовании необходимо запустить ВМ qemu с опцией `-enable-kvm` (или просто запустите kvm, если он установлен).

Если виртуализация не поддерживается, то ВМ сразу завершит исполнение и выдаст сообщение о том, что гипервизор KVM неподдерживается.

Чтобы полноценно виртуализировать систему, необходимо создать образ жёсткого диска данной системы. **QEMU** предоставляет специальную команду для создания жесткого диска, которая называется `qemu-img`. Эта утилита создает образы различных форматов:

- **vdi** образ ВМ поддерживаемый VirtualBox
- **vmdk** образ виртульных машин VMware
- **qcow2** аббревиатура Qemu Copy-On-Write

Лучшим для qemu форматом является qcow2. Данный формат поддерживает выполнение на лету сжатия данных, снимки образа и шифрования данных. Кроме того, qcow2 образ занимает столько места, сколько данных записан в него виртуальной машиной, вне зависимости от размера, задаваемого при создании.

Новый образ диска создается при помощи опции `create`, для конвертации одного формата образа в другой используется опция `convert`, а для того, чтобы просмотреть

информацию об образе используется опция info

Например:

qemu-img info [-f <format >] <img_name>

Система **QEMU** обладает возможностью делать несколько образов дисков виртуальных машин, на основе базового образа (шаблона). При этом в образ будут записываться только различия между целевым и базовым образами.

Пусть уже создан образ base.qcow2 с установленной операционной системой на нём. Для того, чтобы на его основе создать еще несколько образов необходимо воспользоваться командами:

```
# qemu-img create -f qcow2 -o backing_f i l e=base.qcow2
Formatting 'target1.qcow2 ' , fmt=qcow2 size =5368709120 backing_file='base.qcow2 '
encryption=off cluster_size=0

Formatting 'target2.qcow2 ' , fmt=qcow2 size =5368709120 backing_file='base. qcow2 '
encryption=off cluster_size=0
```

После выполнения этих действий будут существовать три образа**: base.qcow2, target1.qcow2, target2.qcow2** При этом два последних не занимают на диске существенного места и просто ссылаются на первый. При запуске виртуальной машины из образа **target1.qcow2** во время чтения данные извлекаются из базового образа base.qcow2, а в целевой файл будут записываться только изменения относительно базового образа. При этом базовый образ модифицироваться не будет. Недостатком такого подхода является то, что при удалении/повреждении базового образа все целевые образы становятся неработоспособными.

Во время работы ВМ **QEMU** предоставляет консоль для взаимодействия с ВМ, которая называется монитор (monitor). Монитор QEMU используется для выполнения сложных команд в эмуляторе, вчастности он позволяет вам:

- инспектировать состояние гостевой ОС
- проводить её отладку

- извлекать или вставлять съёмные накопители (таких как CD-ROM илиUSB устройства)
- "замораживать"/"размораживать" ВМ
- сохранять или восстанавливать её состояние из файла на диске
- модифицировать другие параметры работы ВМ

Монитор в графическом режиме доступен при нажатии сочетания клавиш в окне эмулятора " Ctrl+Alt+2 " (" Ctrl+Alt+1 " для возврата к ОС"). Монитор может быть перенаправлен и доступен при подключении другими способами. В данной лабораторной работе вам предлагается использовать протокол telnet для доступа к монитору (см. ниже).

Полный список команд доступных для выполнения в мониторе можно получить с помощью команды монитора

```
help
```

Для получения более подробной информации по каждой команде
также можно использовать данную команду:

```
help <command>
```

К основным командам монитора относятся:

- info #Получение информации о ВМ
- stop / cont #Приостановка/продолжение работы ВМ незаметное длягостевой ОС "замораживание "ВМ.
- system_reset #Перезагрузка ВМ, аналог нажатия кнопки "Reset" на системном блоке.
- savevm / loadvm / delvm #Команды для сохранения/загрузки/удаления снимков состояния ВМ.
- balloon #Позволяет изменить объем оперативной памяти доступной гостевой ОС.
- device_add / device_del #Добавление/удаление устройств
- hostfwd_add / hostfwd_remove #Добавление/удаление сетевых маршрутов передачи пакетов ("проброса портов").

B) Auditd

Для проведения аудита на серверах под управлением Linux-систем используют демон auditd. Он предназначен для комплексной проверки операционной системы Linux, например, для регистрации различных событий, анализа действий программ и предоставления информации администратору по заданным шаблонам.

Также auditd проверяет операционную систему на ошибки: при необходимости активируется служба оповещения, которая высылает администратору предупреждения и сообщения.

Auditd — нативный инструмент, предназначенный для мониторинга событий операционной системы и записи их в журналы событий, разрабатываемый и поддерживаемый компанией RedHat.

auditctl — управление системой аудита, получение информации о состоянии системы, добавление и удаление правил;
autrace — аудит событий, вызываемых процессами (аналогично strace);
ausearch — поиск событий в журналах;
aureport — создание отчетов о работе аудита.

Правила для логирования можно добавлять следующими способами:

1) записать его в файл **/etc/audit/rules.d/<имя файла>.rules** и перезапустить сервис;
2) записать в файл по произвольному пути и указать его явно: auditctl -R <**путь к файлу**>;
3) добавить правило утилитой auditctl [-A, -a] <**правило**>.

Правила не обязательно задавать, используя командную строку. Во время старта демон auditd читает два файла: **/etc/audit/auditd.conf** и **/etc/audit/audit.rules**

Первый описывает конфигурацию демона и содержит такие опции: как имя журнала и его формат, частота обновления и другие параметры. Нет смысла их изменять, разработчики дистрибутива уже позаботились о грамотной настройке.

Второй файл содержит правила аудита в формате auditctl, поэтому все, что нужно сделать, чтобы получить правило, пригодное для записи в этот файл – просто опустить имя команды.

Порядок выполнения работы:

A)

1) Изучите возможности команды qemu-img:

(a) Создайте образ виртуального жёсткого диска в папке /tmp/ размером 1.5GB в формате vmdk с именем disk_base_$USER.vmdk

(b) $USER переменная среды окружения в которой хранится логинтекущего пользователя

(c) Измените формат образа на qcow2, изменив также расширениефайла

(d) Увеличьте размер образа диска до 7Gb

(e) С помощью qemu-img создайте целевой (дочерний) образ диска,базирующийся на образе диска, созданном на предыдущем этапе. Образ в формате qcow2 должен называться disk_$USER.qcow2 ирасполагаться в директории /tmp/

2) Определите поддерживается ли гипервизор KVM на вашем оборудовании как описано в предыдущей главе (для тестов можно использовать файл CD-ROM /var/qemu/OS/ubuntu14.iso). Если KVM поддерживается, в дальнейшем используйте его при работе с ВМ.

3) Запустите виртуальную машину qemu с необходимыми параметрами:

- Количество процессоров 1
- Оперативная память 512Mb
- Тип эмулируемой видеокарты std
- Образ жёсткого диска образ, созданный вами на предыдущем этапе лабораторной работы (целевой)
- Файл CD-ROM /var/qemu/OS/xubuntu14.iso
- Сеть пользовательская сеть
- Проброс портов: порт хост-компьютера = 8080) порт виртуальноймашины = 80
- Включите отображение меню выбора устройства для загрузки
- Таймаут отображения меню 10 секунд
- Дополнительные опции:

-serial none -monitor telnet: 127.0.0.1:10023, server, nowait

4) Взаимодействие с работающей ВМ через монитор.

(a) Для этого ВМ должна работать. Загрузите гостевую ОС.

(b) Подключитесь к монитору ВМ по протоколу telnet с помощью команды:

```
telnet 127.0.0.1 10023
```

(c) Отключиться от монитора можно сочетанием клавиш Ctrl+], Ctrl+d

(d) Получите информацию о

- процессорах
- регистрах процессоров
- сети
- блочных устройствах

(e) Удалите существующий проброс портов:

```
порт хост-компьютера = 8080 >  порт виртуальной машины = 80
```

(f) Добавьте новый проброс портов к виртуальной машине:

```
порт хост-компьютера = 2222 >  порт виртуальной машины = 22
```

(g) Выполните сохранение текущего состояния ВМ с тегом "running_state"

(h) Перезагрузите виртуальную систему

(i) Принудительно завершите работу ВМ исполнив команду quit

(j) Получите информацию об образах виртуальной машины, которые вы создавали и использовали во время работы ВМ. Какой объём они занимаютв данный момент? Какие снимки состояния в них хранятся?

(k) Восстановите работу ВМ из сохранённого снимка состояния.

B)

1. Узнайте список всех пользователей Linux
2. Получите вывод только имён пользователей в системе
3. Узнайте список всех подключенных пользователей к системе в данный момент времени
4. С помощью команды find найдите в корневом каталоге файлы:
   a) имеющие атрибуты **`SUID;`**

b) имеющие атрибуты **SGID;**

c) имеющие атрибуты **SGID и SUID;**

d) файлы, которые разрешено модифицировать всем;

e) файлы, не имеющие владельца

5. С помощью команды **id user_name** посмотрите список основной и дополнительных групп пользователей. Найдите дополнительные группы **floppy**, **cdrom** и **plugdev**, дающие право использовать сменные машинные носители **/etc/cdrom**, **/etc/fd0** и т.д. для бесконтрольного блочного копирования данных.

6. Зарегистрируйте нового пользователя и добавьте его в разные группы, выведите список существующих пользователей и группы, проверьте наличие нового пользователя

7. С помощью команды **md5sum** вычислите и запишите контрольную сумму для одного из файлов в каталоге **/home/**

8. С помощью команды **md5sum** вычислите и запишите в файл контрольную сумму всех файлов в каталоге **/bin**.

9. Снова с помощью команды **md5sum** вычислите и запишите в файл контрольную сумму всех файлов в каталоге **/bin** и добавьте какие-нибудь символы в конце файла, после сравните обе суммы

10. Найдите в папке /usr/share, включая подкаталоги, простые файлы “doc” и скопируйте найденное в папку /tmp/docs/

11. Установите пакет auditd для мониторинга событий операционной системы и записи их в журналы событий

12. Просмотрите статус службы auditd

13. Запустите службу auditd

14. Выведите абсолютно все события аудита за день

15. Выведите результаты аудита по времени

16. Установите пакет figlet

17. Запустите figlet таким образом, чтобы на экране отобразилась ваша фамилия и группа

Контрольные вопросы:

1. Что такое KVM?

2. Что такое QEMU?

3. При каких условиях можно использовать гипервизор ядра KVM для виртуализации?

4. Для чего применяется монитор QEMU?

5. Назовите основные команды монитора QEMU?

6. Что такое виртуализация?