ПРИЛОЖЕНИЯ UNIX СИСТЕМ

Лекция №9

Ст. преподаватель
кафедры ПДСиМ
Квиткова Ирина Геннадьевна

Концепция iptables

Конвейерная обработка сетевого пакета системой:

Э.1. Получение пакета из сетевого интерфейса или от системного процесса.

Э.2. Определение предполагаемого маршрута пакета.

Э.3. Отправка пакета через сетевой интерфейс либо какому-нибудь процессу, если он предназначался данному компьютеру.

Концепция iptables

- ✓ Три конвейера обработки пакетов:

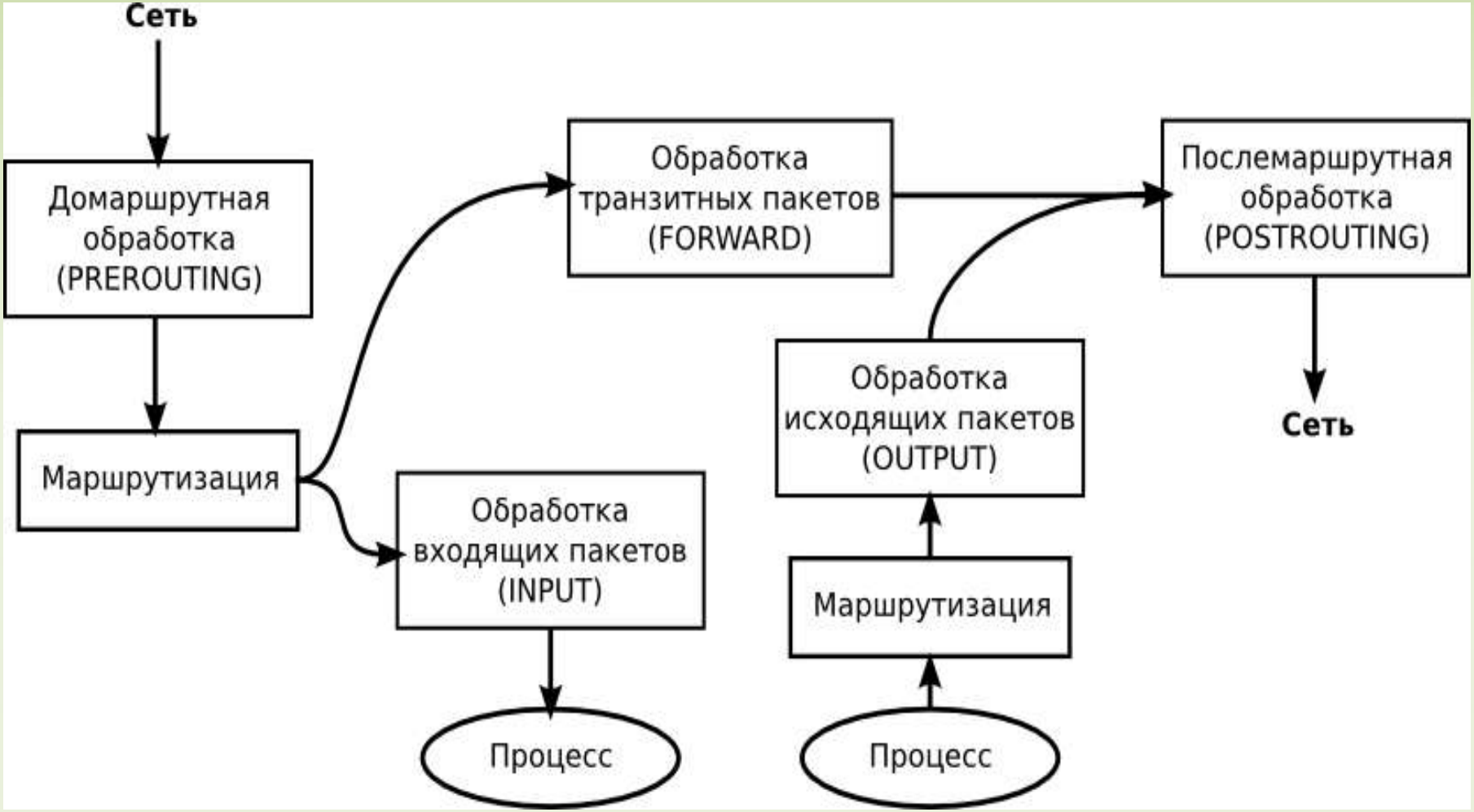
1. «получить – маршрутизировать – отослать» (действие маршрутизатора),

2. «получить – маршрутизировать – отдать» (действие при получении пакета процессом),

3. «взять – маршрутизировать – отослать» (действие при отсылке пакета процессом).

- ✓ Между каждыми из этих действий системы помещается модуль межсетевого экрана - *цепочка*.
- ✓ Каждая цепочка представляет собой список правил, последовательно применяемых к анализируемому пакету.

Концепция iptables

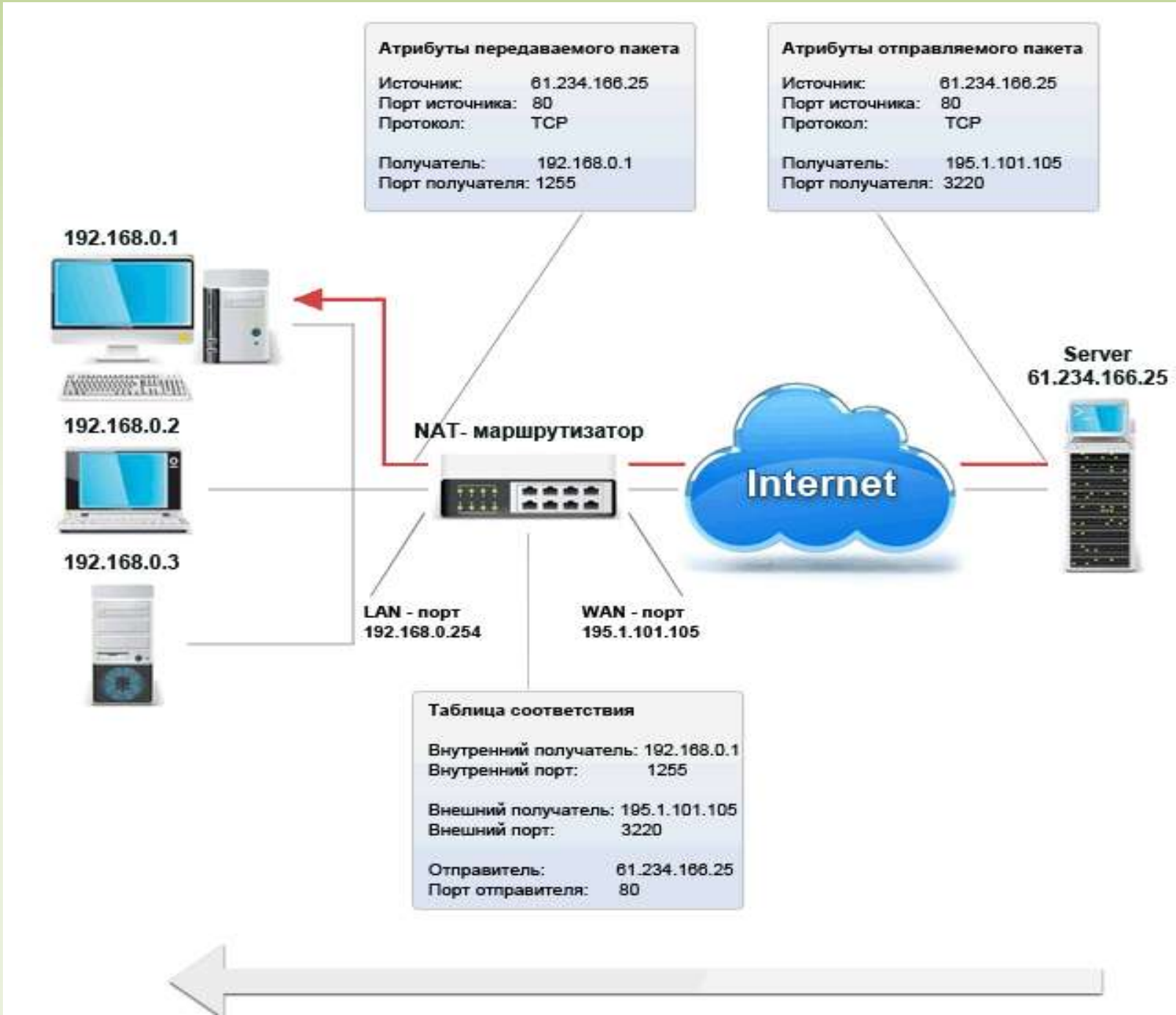


Функция NAT

- ✓ Одной из важных функций сетевого экрана является **подмена адресов** и **модификация сетевых пакетов**.

NAT (Network Address Translation — трансляция сетевых адресов) — механизм, позволяющий организовать передачу пакетов между сетями, не имеющими сведений о сетевых адресах друг друга.

- ✓ Позволяет экономить публичные IP-адреса.
- ✓ Скрывает внутреннюю структуру сети.


Атрибуты передаваемого пакета
Источник: 61.234.166.25
Порт источника: 80
Протокол: TCP
Получатель: 192.168.0.1
Порт получателя: 1255
Атрибуты отправляемого пакета
Источник: 61.234.166.25
Порт источника: 80
Протокол: TCP
Получатель: 195.1.101.105
Порт получателя: 3220
192.168.0.1
192.168.0.2
192.168.0.3
NAT- маршрутизатор
Internet
Server
61.234.166.25
LAN - порт
192.168.0.254
WAN - порт
195.1.101.105
Таблица соответствия
Внутренний получатель: 192.168.0.1
Внутренний порт: 1255
Внешний получатель: 195.1.101.105
Внешний порт: 3220
Отправитель: 61.234.166.25
Порт отправителя: 80

Примеры использования iptables

- **iptables -F** *# Очистить все цепочки таблицы filter.*
- **iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT** *# Ко всем пакетам, которые относятся к уже установленным соединениям, применяется действие ACCEPT – пропустить.*
- **iptables -P INPUT DROP** *# В качестве действия по умолчанию - DROP — блокирование пакета.*
- **iptables -P OUTPUT ACCEPT** *# Разрешить все исходящие пакеты.*
- **iptables -A INPUT -p tcp --dport 80 –s 192.168.0.0/24 -j REJECT** *- запретить входящие соединения для сервиса http с хостов сети 192.168.0.0/24, к которой настроена маршрутизация.*

Основные сетевые команды

netstat - отображение имеющихся сетевых соединений.

Пример: **netstat –at** – *просмотр только tcp соединений.*

ping узел – проверка соединения с указанным узлом.

Пример: **ping –c 100 192.168.0.1** – *проверяет соединение с узлом 192.168.0.1, отсылая 100 запросов.*

ping –w 5 localhost – *пинговать localhost в течение 5 с.*

traceroute – примерное определение маршрута следования пакета.

Пример: **traceroute yandex.ru** *покажет маршрут пакета-запроса до yandex.ru*.

Основные сетевые команды

mtr – средство диагностики сети (комбинирует ping и traceroute)

Пример: **mtr google.com** – *исследование сетевых соединений с google.com*.

host – определение IP-адреса по заданному имени хоста и наоборот.

Пример: **host yandex.ru**

host 8.8.8.8

hostname – отобразить имя компьютера.

telnet hostname - удалённое подключение к терминалу hostname по протоколу TELNET. Информация передаётся без шифрования.

ssh – аналог **telnet**, но с шифрованием данных.

Виртуализация в UNIX

Основы виртуализации

Виртуализация
предоставление набора вычислительных ресурсов или их логического объединения, абстрагированное от аппаратной реализации, и обеспечивающее логическую изоляцию друг от друга выч. процессов, выполняемых на одном физическом ресурсе.

✓ Виртуализация может применяться к:
- компьютерам,
- операционным системам,
- устройствам хранения,
- приложениям или сетям.

✓ Основную роль играет виртуализация серверов.

Основы виртуализации

Виртуализация серверов - это запуск на одном физическом сервере нескольких виртуальных серверов, которые представляют собой приложения, запущенные на хостовой ОС, эмулирующие физические устройства сервера.

- ✓ ПО используется для имитации наличия оборудования и создания виртуальной компьютерной системы (к.с.).
- ✓ Поэтому можно запускать несколько виртуальных систем, а также несколько ОС и приложений на одном сервере, что может обеспечить экономию и повышение эффективности.

Виртуальная машина

Виртуальная к.с. или виртуальная машина (ВМ) — это строго изолированный контейнер ПО, содержащий ОС и приложение.

- ✓ Каждая автономная ВМ полностью независима.
- ✓ Наличие нескольких ВМ на одном компьютере обеспечивает возможность работы на одном физическом сервере нескольких ОС и приложений.
- ✓ Тонкий слой ПО, называемый **гипервизором**, отделяет виртуальные машины от сервера и по мере необходимости динамически выделяет вычислительные ресурсы каждой ВМ.

Основные свойства виртуальных машин

1. Разделение:

-выполнение нескольких ОС на одном физическом компьютере;

-разделение системных ресурсов между ВМ.

2. Изоляция:

-изоляция неисправностей и нарушений системы безопасности на аппаратном уровне;

-сохранение уровня производительности с помощью расширенных средств управления ресурсами.

3. Инкапсуляция:

-полное сохранение состояния ВМ в виде файлов;

-перемещение и копирование ВМ аналогичны операциям с файлами.

4. Независимость от оборудования:

- инициализация и перенос любой ВМ на любой физический сервер.

Виртуализация платформ

Виртуализация платформ - создание программных систем на основе существующих аппаратно-программных комплексов, зависящих или независящих от них.

- ✓ Система, предоставляющая аппаратные ресурсы и ПО, называется **хостовой (host)**, а симулируемые ей системы — **гостевыми (guest)**.
- ✓ Виды виртуализации платформ зависят от того, насколько полно осуществляется симуляция аппаратного обеспечения (а.о.).

Виды виртуализации платформ

Полная эмуляция (симуляция)

- ВМ полностью виртуализирует всё а.о. при сохранении гостевой ОС в неизменном виде.
- Позволяет эмулировать различные аппаратные архитектуры.
- Используется для разработки системного ПО, низкоуровневой отладки ОС, а также образовательных целей.
- Основной минус: эмулируемое а.о. существенно замедляет быстродействие гостевой системы.
- Примеры продуктов для создания эмуляторов: **Bochs, PearPC, QEMU (без ускорения), Hercules Emulator.**

Виды виртуализации платформ

Частичная эмуляция (нативная виртуализация)

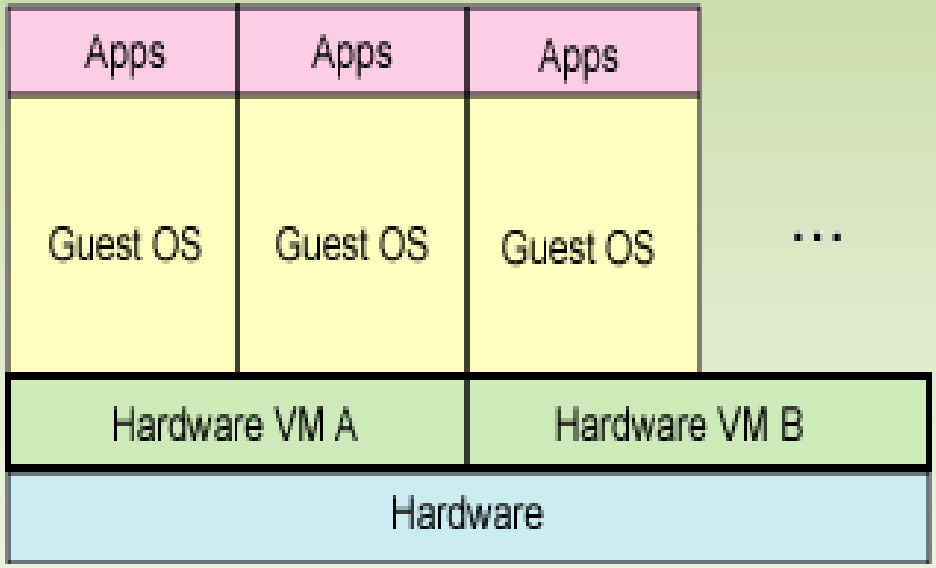
- ВМ виртуализирует лишь необходимое количество а.о., чтобы она могла быть запущена изолированно.
- Для повышения быстродействия применяется гипервизор, позволяющий гостевой системе напрямую обращаться к ресурсам а.о.
- Минус: зависимость ВМ от архитектуры аппаратной платформы.
- Примеры продуктов: **VMware Workstation, VMware Server, VMware ESX Server, Virtual Iron, Virtual PC, VirtualBox, Parallels Desktop**.

Виды виртуализации платформ

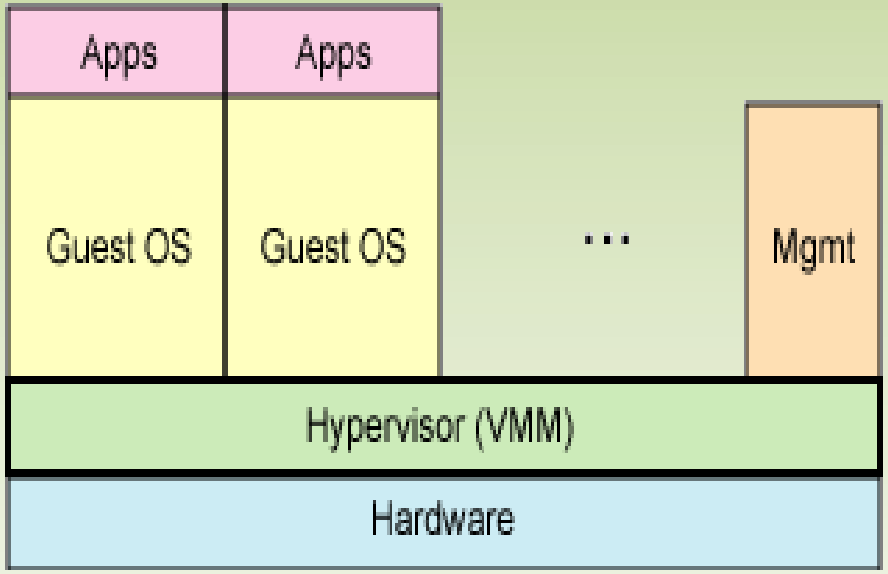
Паравиртуализация

- Нет необходимости симулировать а.о., но используется специальный программный интерфейс (API) для взаимодействия с гостевой ОС.
- Требуется модификация кода гостевой системы.
- Имеется свой гипервизор; API-вызовы к гостевой системе называются «hypercalls» (гипервызовы).
- Основные провайдеры: XenSource и Virtual Iron.

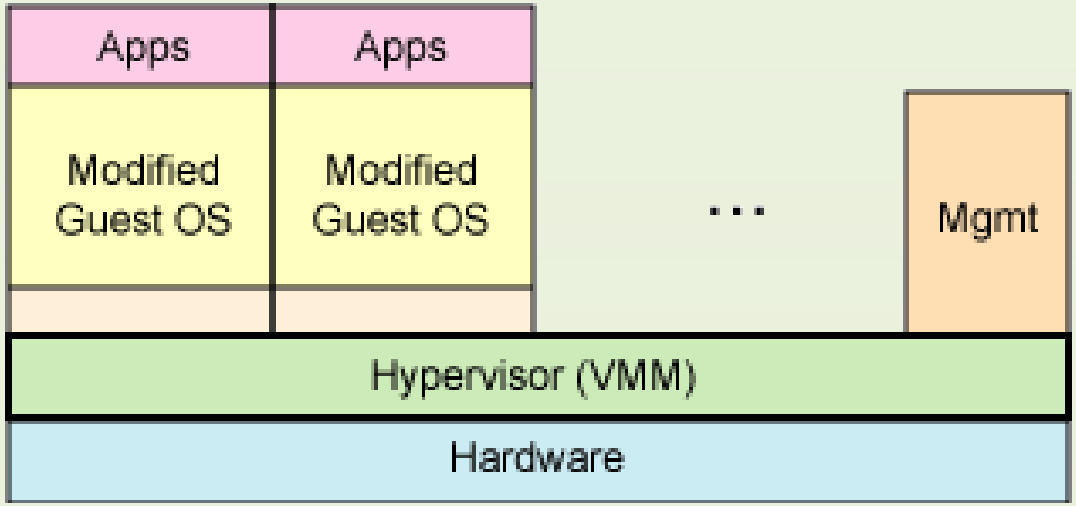
Полная эмуляция



Частичная эмуляция



Паравиртуализация



Системы виртуализации

OpenVZ

- Реализация на уровне ОС, которая базируется на ядре Linux. Позволяет на одном физическом сервере запускать множество изолированных копий ОС - «виртуальные частные серверы». Платный вариант: Virtuozzo.

Xen

- В основе - технология паравиртуализации. Гостевая система (Linux, FreeBSD, OpenSolaris) специально подготавливается для работы с Xen.

KVM

- Сам по себе KVM не выполняет эмуляции. Превращает ядро Linux в гипервизор загрузкой модуля ядра kvm.ko. Использует интерфейс /dev/kvm для настройки адресного пространства гостя ВМ.

VirtualBOX/ VMWare/ Qemu

Спасибо за внимание