Руководство пользователя Xen v3.0

Оригинал: Xen Users' Manual

Перевод: Руководство пользователя Xen

От переводчика

Практически год назад вышла третья версия системы виртуализации Xen. За это время система замечательно зарекомендовала себя и завоевала большое количество сторонников, как во всём мире, так и среди русскоязычных пользователей. Возможно, она была бы у нас ещё популярнее, но для многих барьером, мешающим использовать Xen, является недостаток русскоязычной документации.

Этот документ является первым полным переводом руководства пользователя Xen на русский язык. Следует отметить, что существуют частичные переводы руководства пользователя Xen сделанные ранее (например, перевод Михаила Сгибнева доступный на сайте http://dreamcatcher.ru/).

Данный перевод выполнен в максимальном соответствии оригиналу. Любые дополнения по тексту отмечены текстом "*прим. перев.*".

Многие использовавшиеся русскоязычные термины не являются ещё устоявшимися. В некоторых случаях они режут слух, а в некоторых -- подходящих русских терминов найти вообще не удалось. Если у вас есть мысли или замечания, касающиеся перевода терминологии и вообще русскоязычной версии руководства, пожалуйста, напишите о них на wiki-странице Руководство Пользователя Xen или непосредственно автору перевода по электронной почте: [mailto:igor@chub.in].

Множество вопросов по инсталляции, настройке и эксплуатации Xen-системы, выходящих за рамки руководства, описано на русскоязычных wiki-страницах по Xen. Вы можете принять участие в работе над этими страницами, дополнить их или просто указать вопросы, которые, с вашей точки зрения, стоило бы осветить.

Спасибо за обратную связь, комментарии, замечания, и любую другую помощь; и просто за то, что вы пользуетесь Xen. А я надеюсь, что вам моя работа окажется полезной.

Игорь Чубин, *январь 2007*

Введение

Xen -- это монитор виртуальных машин (VMM, Virtual Machine Monitor) или *гипервизор* (hypervisor) с поддержкой *паравиртуализации* (para-virtualization) для процессоров x86 архитектуры, распространяющийся с открытым исходным кодом (opensource). Xen может организовать совместное безопасное исполнение нескольких виртуальных машин на одной физической системе с производительностью близкой к непосредственной (native). Xen обладает функциональностью ПО корпоративного уровня; в нём, в частности, обеспечивается:

- Производительность виртуальных машин близкая к производительности при непосредственном исполнении на железе;
- Возможность живой миграции работающих виртуальных машин между хостами;
- Поддержка до 32 виртуальных процессоров на одну гостевую машину с возможностью горячего добавления (hotplug) процессоров;
- Поддержка платформ x86/32, x86/32 с PAE и x86/64;
- Поддержка аппаратной виртуализации Intel Virtualization Techonology (VT-x) для запуска немодифицированных операционных систем (включая Microsoft Windows);
- Отличная поддержка оборудования (поддерживаются практически все драйверы устройств Linux).

Варианты использования

Возможные варианты использования Xen:

Консолидация серверов.
Размещение множества различных серверов на одном физическом хосте, с возможностью изоляции сбоев и регулирования производительности каждой виртуальной машины.

Независимость от аппаратного обеспечения.
Запуск старых приложений и операционных систем на новом железе.

Запуск множества различных ОС.
Одновременное выполнение множества ОС при разработке, тестировании или экспериментах.

Разработка ядра ОС.
Тестирование и отладка ядра в ограниченной виртуальной машине -- нет необходимости в выделении отдельной тестовой машины.

Кластерные системы.
Дробление на виртуальные машины даёт больше гибкости чем при раздельном управлении каждым физическим хостом, больше возможностей управления и большую изоляцию. Есть возможность живой миграции с целью балансировки нагрузки.

Аппаратная поддержка для новых ОС.
Возможность разработки новых операционных систем пользуясь при этом широким спектром драйверов существующих операционных систем, в частности ОС Linux.

Поддержка операционными системами

Паравиртуализация позволяет добиваться очень высокой производительности виртуализированных систем, даже на x86 платформе, которая традиционно поддается виртуализации с большим трудом.

Этот подход требует чтобы операционная система была *портирована* для запуска в Xen. Портирование ОС для запуска в Xen похоже на добавление поддержки ещё одной аппатной платформы, но проще, поскольку архитектура паравиртуальной машины очень похожа на архитектуру базовой системы. Ядра операционных систем должны быть портированы на Xen, но ключевая особенность заключается в том, что пользовательские программы (user space apllications) и библиотеки *не требуют* модификации.

На процессорах с аппаратной поддержкой виртуализации (технологии Intel VT и AMD SVM) есть возможность запускать немодифицированные гостевые системы. Портирование не требуется, однако необходима дополнительная драйверная поддержка внутри самого Xen. В отличие от традиционных гипервизоров, выполняющих полную виртуализацию, которые страдают от большой потери производительности, Xen и VT или Xen и Pacifica (SVM) дополняют друг друга: комбинация предлагает превосходную производительность паравиртуализированных гостевых операционных систем и вместе с тем полную поддержку немодифицированных гостевых систем, работающих непосредственно на процессоре. Полная поддержка технологий VT и Pacifica появится в начале 2006 года (Уже появилась -- *Прим. перев.*).

Поддержка паравиртуализации Xen в большом количестве операционных систем и их число постоянно растёт. В настоящий момент поддержка Linux находится уже в очень зрелом состоянии, она включена в стандартный дистрибутив. Портирование других ОС, включая NetBSD, FreeBSD и Solaris x86 v10, близко к завершению (о поддержке Xen разными ОС в настоящий момент читайте на странице Xen -- *Прим. перев.*).

Поддержка аппаратного обеспечения

В настоящий момент Xen работает на x86 архитектуре, и требует процессора P6 или новее (например, Pentium Pro, Celeron, Penitum II, Pentium III, Pentium IV, Xeon, AMD Athlon, AMD Duron). Поддерживаются многопроцессорные машины и поддерживается HyperThreading (SMT). Кроме того, ведется портирование на архитектуры IA64 и Power.

Обычный 32-битный Xen поддерживает до 4GB оперативной памяти. Однако, в Xen 3.0 появилась поддержка расширений физической адресации Intel PAE (Phyisical Addressing Extensions), которая позволяет на машине x86/32 адресовать до 64GB физической памяти. Xen 3.0 также поддерживает платформы x86/64, такие как Intel EM64T и AMD Opteron, на которых возможна адресация до 1TB физической памяти.

Xen перекладывает большинство задач по поддержке железа на гостевую операционную систему, работающую в управляющей виртуальной машине, также известной как *домен 0*. Сам Xen содержит только код, необходимый для обнаружения и запуска остальных процессоров системы, настройки обработки прерываний и нумерации PCI шины (PCI bus enumeration <--!-- >). Драйверы устройств работают внтури привилегированной гостевой операционной системы, а не в самом Xen. Такой подход обеспечивает совместимость с большинством устройств, поддерживаемых Linux. Сборка XenLinux по умолчанию содержит поддержку большинства серверного сетевого и дискового оборудования, но при необходимости можно добавить поддержку других устройств, переконфигурировав Linux-ядро стандартным способом.

Структура Xen-системы

В системе Xen есть множество уровней, нижним и наиболее привилегированным из которых

является сам Xen.

В Xen может работать множество гостевых операционных систем, каждая из которых выполнятся в безопасной виртуальной машине. В терминологии Xen такая машина называется *домен*. Исполнение кода доменов планируется Xen так, чтобы сделать использование доступных процессоров наиболее эффективным. Каждая гостевая ОС занимается управлением собственных приложений. Это управление включает ответственность за планирование выполнения каждого приложения в течение временного интервала, выделенного Xen виртуальной машине.

Первый домен, *домен 0*, создаётся автоматически при загрузке системы. Этот домен обладает особыми управляющими привилегиями. Домен 0 запускает остальные домены и предоставляет им виртуальные устройства. Еще он занимается выполнением административных задач, таких как остановка (suspending), возобновление (resuming) работы виртуальных машин и их миграция.

В домене 0 работает процесс **xend**, который и занимается управлением Xen. Xend отвечает за управление виртуальными машинами и обеспечение доступа к их консолям. Команды Xend даются из командной строки и передаются ему через HTTP-интерфейс.

История

Первоначально Xen разрабатывался исследовательской группой Systems Research Group компьютерной лаборатории Кембриджского университета (University of Cambridge Computer Laboratory) как часть проекта XenoServers, спонсируемого UK-EPSRC.

Цель XenoServers -- предоставить "общедоступную инфраструктуру для глобальных распределенных вычислений" (public infrastructure for global distributed computing). Xen играет здесь ключевую роль, позволяя эффективно разделять одну физическую машину между несколькими клиентами, выполняющими собственные операционные системы и приложения в собственной ограниченной среде. Эта среда предоставляет защиту, изоляцию ресурсов и учёт. Дополнительные сведения о Xenoservers можно получить на web-странице проекта, там же есть ссылки на документацию и технические отчёты: http://www.cl.cam.ac.uk/xeno.

Xen вырос в самодостаточный проект, позволяющий исследовать интересные вопросы, касающиеся наилучших решений в области виртуализации ресурсов: процессора, памяти, диска и сети. Среди тех, кто сделал и постоянно делает вклад в развитие проекта -- XenSource, Intel, IBM, HP, AMD, Novell, RedHat.

Xen впервые был описан в документе представленном на SOSP в 2003 году http://www.cl.cam.ac.uk/netos/papers/2003-xensosp.pdf, а его первый общедоступный релиз (1.0) был выпущен в октябре того же года. С того времени Xen существенно вырос и повзрослел, и сейчас он используется для решения производственных задач во многих системах.

Что нового

В Xen 3.0.0 есть:

- Поддержка SMP гостевых операционных систем;
- Поддержка Intel PAE (Physical Addressing Extensions) для серверов с большим чем 4G оперативной памяти;
- Поддержка x86/64 (Intel EM64T, AMD Opteron);
- Поддержка Intel VT-x для запуска немодифицированных гостевых ОС (Windows XP/2003, старый Linux);

- Расширенные инструменты управления;
- Улучшенная поддержка ACPI;
- Поддержка графики AGP/DRM.

Среди новых возможностей Xen 3.0 также расширенная поддержка аппаратного обеспечения, гибкость конфигурирования, удобство использования и более широкий спектр поддерживаемых операционных систем. Последний релиз Xen делает его ближе к тому чтобы стать лучшим открытым решением для виртуализации.

Инсталляция

Базовая инсталляция

Дистрибутив Xen включает три главных компонента: собственно Xen, порт Linux и NetBSD для работы в Xen и утилиты для управления Xen-системой. В этой главе рассказывается, как проинсталлировать дистрибутив Xen 3.0 из исходников. Возможен и другой вариант -- Xen может быть доступен в виде откомпилированных пакетов в самом дистрибутиве операционной системы.

Предварительные требования

Ниже перечислено программное обеспечение, необходимое для сборки и работы Xen.

Без этого Xen работать не будет:

- Работающий дистрибутив Linux с загрузчиком GRUB, на системе с процессором P6 или старше.

Это необходимо для управляющих утилит Xen:

- Пакет `iproute2`;
- Пакет bridge-utils для Linux (например, `/sbin/brctl`);
- Система hotplug для Linux (`/sbin/hotplug` и связанные с ним скрипты). В более новых дистрибутивах ещё и udev.

Это необходимо для того чтобы собрать Xen из исходников:

- Сборочные утилиты (gcc v3.2.x или v3.3.x, binutils, GNU make);
- Инсталляция zlib с заголовочными файлами (например, zlib-dev);
- Инсталляция Python v2.2 или больше с заголовочными файлами (e.g., python-dev);
- LaTeX и transfig для сборки документации.

После того как все требования удовлетворены, можно инсталлировать Xen из дистрибутива или его бинарную версию.

Инсталляция из архива бинарников

Архивы с откомпилированным Xen доступны для скачивания здесь:

```
http://www.xensource.com/downloads/
```

После того как архив получен, его нужно распаковать и проинсталлировать.

```
# tar zxvf xen-3.0-install.tgz
# cd xen-3.0-install
# sh ./install.sh
```

После того как Xen установлен, его нужно сконфигурировать, как описано здесь.

Инсталляция из RPM

RPM-пакеты с откомпилированным Xen доступны здесь:

```
http://www.xensource.com/downloads/
```

После того как пакет получен, его нужно установить с помощью программы `rpm`:

```
# rpm -iv rpmname
```

Смотрите инструкции и Release Notes для каждого RPM-пакета здесь:

```
http://www.xensource.com/downloads/
```

Инсталляция из исходников

В этой секции описывается, как скомпилировать и проинсталлировать Xen из архива исходных текстов.

Получение архива исходников

Исходный код Xen доступен или как сжатый архивный файл или как клон Mercurial-репозитория Xen.

Получение архива

Архивы стабильных версий и ежедневных срезов дерева исходных кодов Xen находятся здесь:

```
http://www.xensource.com/downloads/
```

Получение исходников из репозитория

Исходный код Xen можно получить непосредственно из Mercurial-репозитория, доступного здесь:

```
http://xenbits.xensource.com
```

Подробности по ссылке "Getting Started Guide" отсюда:

```
http://www.xensource.com/downloads/
```

Сборка из исходников

В Makefile'е верхнего уровня Xen есть цель `world`, которая включает такие задачи:

- Сборка Xen;
- Сборка управляющих утилит, включая Xend;
- Скачивание (если нужно) и распаковка ядра Linux 2.6, наложение патча Xen;
- Сборка ядра Linux для запуска в доменах Xen.

После того как сборка завершена, должен появиться каталог первого уровня `dist/`, в котором будут находиться все результаты сборки. В частности, там будут два ядра -- одно с расширением `-xen0`, в него включены драйверы физических устройств и драйверы виртуальных устройств Xen; и второе с расширением `-xenU`; в нём драйверы только виртуальных устройств. Ядра находятся в каталоге `dist/install/boot/`, там же будет и гипервизор Xen, там же конфигурационные файлы ядер.

Список ядер, которые будут собираться, указывается в `Makefile` верхнего уровня, в строке:

```
KERNELS ?= linux-2.6-xen0 linux-2.6-xenU
```

В эту строку можно включать названия любых ядер, конфигурационные файлы которых присутсвуют в каталоге `buildconfigs/`.

Нестандартноя ядро

Если нужно чтобы конфигурация ядра отличалась от стандартной, распространяющейся вместе с дистрибутивом Xen (например, для дополнительной драйверной поддержки или включения каких-либо опций ядра), можно использовать стандартный механизм конфигурирования ядра Linux. Например:

```
# cd linux-2.6.12-xen0
# make ARCH=xen xconfig
# cd ..
# make
```

Другой способой -- скопировать существующую конфигурацию ядра Linux (`.config`) в корень дистрибутива ядра и выполнить:

```
# make ARCH=xen oldconfig
```

Будут заданы вопросы о специфичных для Xen опциях. Если вы не знаете, для чего нужна та или иная опция, лучше оставляйте значение по умолчанию.

Имейте в виду, что единственное отличие между ядрами, используемыми в домене 0 и остальных доменах -- это их конфигурация. Версия с суффиксом `U` (непривилегированная версия) не содержит никаких драйверов физических устройств, что экономит до 30% его размера, и, вследствие этого, такое ядро может оказаться предпочительным для непривилегированных доменов. Версия с суффиксом `0` может использоваться как для загрузки

системы, т.е. в домене 0, так и для управления непривилегированными доменами.

Инсталляция собранных бинарников

Файлы, полученные в процессе сборки, находятся в каталоге `dist/install/`. Для инсталляции файлов в систему нужно дать команду:

```
# make install
```

Вместо этой команды при необходимости можно скопировать файлы в соответствующие каталоги вручную.

В каталоге `dist/install/boot` находится конфигурационный файл использовавшийся при сборке ядра XenLinux, а также гипервизора Xen и ядра XenLinux с включенной отладочной символьной информацией (например, `xen-syms-3.0.0` и `vmlinux-syms-2.6.12.6-xen0`). Эта информация имеет большое значение при анализе дампов в случае падения системы. Сохраняйте эти файлы, потому что при обращении за помощью в список рассылки разработчики могут попросить чтобы вы показали их.

Конфигурирование

После того как дистрибутив Xen скомпилирован и проинсталлирован, нужно подготовить систему к запуску и использованию Xen. Это сделать довольно легко.

Конфигурирование GRUB

Для того чтобы Xen/Linux мог загружаться, нужно добавить в `grub.conf` соответствующую запись (этот файл обычно находится в `/boot` или `/boot/grub`; в некоторых дистрибутивах он может называться `menu.lst`). Добавляемая запись должна выглядеть так:

```
title Xen 3.0 / XenLinux 2.6
  kernel /boot/xen-3.0.gz dom0_mem=262144
  module /boot/vmlinuz-2.6-xen0 root=/dev/sda4 ro console=tty0
```

Строка kernel указывает GRUB где найти сам Xen, и какие конфигурационные параметры должны быть ему переданы (в данном случае, указывается сколько памяти, в килобайтах, должно быть выделено домену 0; и указывается настройка последовательного порта). Дополнительная информация о различных загрузочных режимах Xen, есть здесь.

Строка module конфигурационного файла указывает местоположение ядра XenLinux, которое должен загрузить Xen, и параметры, которые должны быть ему переданы. Это стандартные параметры Linux, которые:

- указывают на устройство с корневой файловой системой;
- говорят, что корневая файловая система должна быть смонтирована в режиме только-для-чтения;
- указывают устройство для вывода сообщений ядра (здесь первая виртуальная консоль).

В некоторых дистрибутивах, например, в SuSE, параметр `ro` может отсутствовать.

Для того чтобы использовать initrd, добавьте ещё одну строку module в конфигурационный

файл:

```
  module /boot/my_initrd.gz
```

После инсталляции нового ядра рекомендуется не удалять существующую запись в меню из файла `menu.lst`, поскольку старое Linux-ядро может понадобиться в будущем, в особенности при возникновении проблем.

Serial-консоль

Доступ через serial-консоль даёт возможность управлять, наблюдать и взаимодействовать с системой через консоль на её последовательном порту (serial-консоль). Это дает возможность подключиться с другой, рядом стоящей системы через нуль-модемный кабель (LapLink) или удалённо, через serial-концентратор.

BIOS системы, загрузчик GRUB, Xen, Linux и Login -- каждый из них в отдельности должен быть сконфигурирован для доступа через последотельный порт. Не обязательно чтобы каждая из этих составляющих была полностью настроена, но это может оказаться весьма полезно.

Более подробно о кофнигурировании serial-консоли в Linux можно прочитать в "Remote Serial Console HOWTO" на сайте Linux Documentaion Project www.tldp.org

Настройка BIOS на использование serial-консоли

Включение поддержки serial-консоли в BIOS не означает, что автоматически включится поддержка serial-консоли в GRUB, Xen и Linux. Это нужно с другой целью: сделать удаленное управление системой более удобным, поскольку в этом случае POST и другие загрузочные сообщения (до загрузчика системы) передаются на serial-порт. Кроме того, становится доступным конфигурирование через serial-консоль самого BIOS.

О том как включить serial-консоль в BIOS должно быть написано в документации на материнскую плату/систему.

Настройка GRUB на использование serial-консоли

Включение поддержки serial-консоли в GRUB не включает и не выключает её в Xen и в Linux. Это нужно для того чтобы сделать использование системы с Xen более удобным: меню GRUB становится доступным через serial-порт и даёт возможность удалённого управления загрузкой.

Для того чтобы заставить GRUB работать с serial-консолью, надо добавить следующие строки в конфигурационный файл GRUB, который обычно находится в `/boot/grub/menu.lst` или в `/boot/grub/grub.conf`:

```
serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1
terminal --timeout=10 serial console
```

Обратите внимание, что включены и последовательный порт, и локальная консоль на мониторе и клавиатуре. Текст "`Press any key to continue`" появится и там, и там. Нажатие клавиши на каком-либо из устройств приводит к тому что GRUB будет выводить информацию на него, а второе устройство ничего не получит. Если клавиши на клавиатуре не будут нажиматься в течение таймаута, система начнёт загружаться в соответствии с пунктом "по умолчанию".

Более подробная информация приводится в документации GRUB.

Настройка Xen на использование serial-консоли

Включение поддержки serial-консоли в Xen не включает и не выключает её в Linux. Вывод сообщений Xen на консоль делается для того чтобы можно было через serial-консоль смотреть на них при загрузке Xen, что может оказаться очень полезным во время отладки.

Для того чтобы включить вывод Xen на последовательный порт, нужно добавить загрузочную опцию Xen в конфигурационный файл GRUB; например, заменить приводившийся выше пример следующей строкой:

```
kernel /boot/xen.gz dom0_mem=131072 com1=115200,8n1
```

Эта строка настраивает вывод Xen на COM1 на скорости 115200, 8 бит данных, без чётности, один стоп-бит. Параметры можно изменить. Описание всех возможных загрузочных параметров находится здесь.

Настройка ядра Linux на использование serial-консоли

Включение вывода сообщений Linux на serial-консоль не включает и не выключает возможность захода в Linux-систему через последовательный порт. Вывод сообщений на консоль делается для того чтобы можно было через serial-консоль смотреть на них при загрузке, что может оказаться очень полезным во время отладки.

Для того чтобы включить вывод сообщений Linux на serial-консоль во время загрузки, нужно в конфигурационном файле загрузчика передать ядру Linux параметр `console=ttyS0` (или `ttyS1`, или `ttyS2` и т.д.). Например, так:

```
module /vmlinuz-2.6-xen0 ro root=/dev/VolGroup00/LogVol00 \
  console=ttyS0, 115200
```

Вывод сообщений ядра будет выполняться на последовательный порт ttyS0 (COM1) на скорости 115200.

Конфигурирование Login на serial-консоли

Для захода в Linux-систему (не важно с Xen'ом или без) через последовательный порт нужно чтобы на порту было приглашение для входа в систему. Его можно получить с помощью программы mgetty или аналогичной (например, mingetty). Для того чтобы разрешить непосредственный вход root'у через консоль на последовательном порту, нужно добавить имя этого порта в `/etc/securetty`.

Для автоматического запуска программы-приглашения на последовательном порту нужно добавить в `/etc/inittab`:

```
c:2345:respawn:/sbin/mingetty ttyS0
```

Чтобы **init** перечитал этот файл, нужно дать команду:

```
# init q
```

Для разрешения входа root'у через этот терминал, нужно добавить `ttyS0` в `/etc/securetty`.

В вашем дистрибутиве может использоваться другая программы getty: mgetty, agetty или просто getty. Какую именно из них выбрать, уточните в документации на систему.

Библиотеки TLS

Пользователям XenLinux2.6 рекомендуется перед тем как загружать Xen отключить библиотеку TLS (Thread Local Storage). Проще всего это сделать так:

```
# mv /lib/tls /lib/tls.disabled
```

В любой момент можно включить TLS, переименовав каталог обратно:

```
# mv /lib/tls.disabled /lib/tls
```

Это связано с тем, что текущая реализация TLS использует сегментацию памяти недопустимую в Xen. Если TLS не отключить, Xen будет выполнять эмуляцию, которая приведёт к потере производительности. Для того чтобы быть уверенным в том что производительность на максимуме, нужно проинсталлировать специальную, адаптрированную для использования с Xen (nonsegneg) версию библиотеки.

Если запустить систему с отключённым TLS, в ходе загрузки будут выдаваться предупреждающие сообщения. В этом случае после того как машина загрузится нужно переименовать файлы, как описано выше, и запустить `/sbin/ldconfig`, для того чтобы переименование вступило в силу.

Загрузка Xen

После того как инсталляция и конфигурирование завершено, нужно перезагрузить систему и выбрать новый пункт Xen в меню загрузчика GRUB.

Процесс загрузки, который начнётся после этого, будет похож на обычный процесс загрузки Linux. Первая часть сообщений будет выведена собственно самим Xen, это -- низкоуровневые сведения о самом гипервизоре Xen и оборудовании. Оставшуюся информацию выводит уже XenLinux.

На экране могут появиться сообщения об ошибках. Не обязательно эти ошибки имеют принципиальное значение. Они могут быть связаны с различиями в конфигурации нового ядра системы и ядра, которое использовалось до этого.

Когда загрузка закончится, можно зайти в систему, как и обычно. Если в процессе загрузки возникли какие-то проблемы, всегда можно перезагрузить систему и выполнить загрузку на старом Linux-ядре, выбрав его в приглашении GRUB.

Загрузка системы Xen

После загрузки Xen вы попадёте в привилегированный управляющий домен, Domain0. С этого

момента можно с помощью команды `xm create` создавать гостевые домены и выполнять их загрузку.

Загрузка домена 0

После того как инсталляция и конфигурирование завершены, нужно перезагрузить систему и в меню загрузчика GRUB выбрать новый пункт Xen.

Процесс загрузки, который начнётся после этого, будет похож на обычный процесс загрузки Linux. Первая часть сообщений будет выведена собственно самим Xen, это -- низкоуровневые сведения о самом гипервизоре Xen и оборудовании. Оставшуюся информацию выводит уже XenLinux.

Когда загрузка закончится, можно зайти в систему, как и обычно. Если в процессе загрузки возникли какие-то проблемы, всегда можно перезагрузить систему и выполнить загрузку на старом Linux-ядре, выбрав его в приглашении GRUB.

Первый шаг по созданию нового домена это подготовка корневой файловой системы для его загрузки. Как правило, образ в базовой системе выглядит как обычный дисковый раздел, том LVM (или другой системы управления томами), файл на диске или файл-сервере. Самый простой способ подготовить такой раздел -- просто загрузиться со стандартного инсталляционного CD и проинсталлировать систему на другой раздел жесткого диска (Самый простой, но не единственный способ. О других способах читайте на Xen -- *Прим. перев.*).

Для запуска управляющего демона Xen, необходимо дать команду:

```
# xend start
```

Как сделать, чтобы демон стартовал автоматически, смотрите здесь. Когда демон запустится, с помощью программы **xm** можно выполнять управление и наблюдать за доменами, работающими в вашей системе. В этой главе приводятся только самые необходимые сведения об этой программе. Основная информация о программе **xm** в следующей главе.

Запуск гостевых доменов

Создание конфигурационного файла домена

- `/etc/xen/xmexample1` -- простой шаблон конфигурационного файла, описывающий отдельную виртуальную машину.
- `/etc/xen/xmexample2` -- шаблон конфигурационного файла, который может использоваться для создания множества виртуальных машин. При использовании этого шаблона, при старте машины нужно указывать переменную `vmid`.

В этом каталоге есть множество других примеров, которые могут оказаться полезными. Можно скопировать любой из них и доработать. Чаще всего меняются такие параметры:

kernel
Путь к ядру, которое должно использоваться вместе с Xen (например, `kernel = "/boot/vmlinuz-2.6-xenU"`)

memory

Объём оперативной памяти выделяемой домену, МБайт (например, `memory = 64`)

disk

Имя первого раздела в списке вычисляется на основе domain ID. Второй -- используется доменами совместно (например: `disk = ['phy:your_hard_drive%d,sda1,w' % (base_partition_number + vmid), 'phy:your_usr_partition,sda6,r' ]`)

dhcp

Если раскомментировать эту переменную, домен будет получать IP-адрес от DHCP-сервера (например, `dhcp="dhcp"`).

Может быть, потребуется отредактировать переменную `vif` и задать в ней MAC-адрес сетевого интерфейса виртуальной машины. Например:

```
vif = ['mac=00:16:3E:F6:BB:B3']
```

Если переменную не установить, xend автоматически сгенерирует случайный MAC-адрес из диапазона 00:16:3E:xx:xx:xx, который закреплён организацией IEEE за XenSource как OUI (organizationally unique identifier - уникальный идентификатор организации). XenSource разрешает использовать в доменах Xen адреса выбранные случайным образом из этого диапазона.

Список назначенных номеров OUI можно посмотреть на странице http://standards.ieee.org/regauth/oui/oui.txt.

Запуск гостевого домена

Утилита **xm** предоставляет множество команд по управлению доменами. Запуск домена выполняется с помощью команды `create`. Предположим, что вы создали конфигурационный файл `myvmconf`, базирующийся на файле `/etc/xen/xmexample2`. Для старта домена с ID виртуальной машины 1 нужно ввести команду:

```
# xm create -c myvmconf vmid=1
```

Ключ `-c` обозначает, что после того как домен будет создан, **xm** должна превратиться в его консоль. Параметр `vmid=1` устанавливает переменную `vmid`, использующуюся в конфигурационном файле `myvmconf`, равной 1.

В терминале, в котором была введена команда, должна появиться консоль с загрузочными сообщениями домена, а после того как загрузка завершится -- приглашение о входе в систему.

Автоматический запуск и останов доменов

Можно сделать чтобы при старте домена 0 автоматически запускались избранные пользовательские домены, а при останове работы базовой системы она ожидала завершения их работы.

Для того чтобы включить автоматический запуск домена во время загрузки, разместите его конфигурационный файл (или ссылку на этот файл) в каталоге `/etc/xen/auto/`.

В дистрибутиве Xen есть стартовый скрипт в стиле Sys-V для RedHat Linux или других систем, совместимых с LSB, и при инсталляции Xen этот скрипт копируется в каталог `/etc/init.d`.

Затем его можно добавить в загрузку способом, принятым в дистрибутиве системы.

Например, в RedHat:

```
# chkconfig --add xendomains
```

По умолчанию, он будет запускаться на уровнях запуска 3, 4 и 5.

Скрипт можно запустить вручную с помощью команды `service` (или вызвав непосредственно из каталога `/etc/init.d`):

```
# service xendomains start
```

Этак команда запускает все домены, перечисленные в каталоге `/etc/xen/auto/`.

```
# service xendomains stop
```

Эта команда останавливает все, работающие в настоящий момент, домены Xen.

Конфигурирование и управление

Инструменты управления доменами

В этой главе рассматривается управляющее программное обеспечение и утилиты Xen.

Xend

Демон Xend выполняет функции по управлению виртуальными машинами. Он представляет собой центральную точку управления виртуализированными ресурсами. Этот демон обязательно должен работать, для того чтобы запускать и управлять виртуальными машинами. Xend должен работать от имени root'а, потому что ему нужен доступ к привилегированным системным функциям.

Для запуска Xend во время загрузки создан специальный стартовый скрипт `/etc/init.d/xend`. С помощью соответствующей программы (например, chkconfig) или создавая ссылки вручную можно указать уровни выполнения, на которых будет работать этот скрипт.

Демон Xend можно запустить прямо из командной строки.

```
# xend start        # запустить xend,  если он ещё не работает
# xend stop         # остановить xend, если он запущен
# xend restart      # перезапустить xend, ессли он работает; иначе - запустить
# xend status       # показать состояние xend с помощью кода завершения
```

Скрипт SysV, предназначенный для запуска Xend во время загрузки, входит в состав дистрибутива Xen. Команда `make install` устанавливает скрипт в `/etc/init.d`. Для включения этого скрипта в загрузку, нужно воспользоваться chkconfig или создать символические ссылки вручную. После того как Xend работает, администрирование выполняется с помощью программы `xm`.

Ведение журналов

Сообщения от Xend записываются в файлы журналов `/var/log/xen/xend.log` и (менее часто) `/var/log/xen/xend-debug.log`. Эти файлы, наряду со стандартными журналами Syslog, могут пригодиться при поиске и устранении неисправностей.

Конфигурирование Xend

Xend написан на Python'e. При запуске он читает конфигурационную информацию из файла `/etc/xen/xend-config.sxp`. Инсталлятор Xen размещает файл-пример `xend-config.sxp` в каталог `/etc/xen`, что должно работать для большинства инсталляций.

Полное описание конфигурационных параметров приводится в файле `xend-debug.sxp` и man-странице xend-config.sxp. Некоторые наиболее важные параметры описаны ниже.

Для связи с Xend используется HTTP-интерфейс и доменные гнёзда UNIX. С их помощью удалённые пользователи могут передавать команды демону. По умолчанию Xend не запускает HTTP-сервер. Сервер управления через доменное гнездо Unix запускается, поскольку он требуется утилитой **xm**. Для поддержки миграции между машинами Xend может запускать демон перемещения (relocation daemon). Из соображений безопасности по умолчанию этот демон не включен.

Замечание: Настройки Xend в примере конфигурационного файла отличаются от настроек по умолчанию: запускается как HTTP-сервер, так и сервер перемещения.

Из файла:

```
#(xend-http-server no)
(xend-http-server yes)
#(xend-unix-server yes)
#(xend-relocation-server no)
(xend-relocation-server yes)
```

Для включения или выключения интересующих возможностей, нужно закомментировать или раскомментировать строки в этом файле.

Соединения с удалённых хостов по умолчанию отключены:

```
# Address xend should listen on for HTTP connections, if xend-http-server is
# set.
# Specifying 'localhost' prevents remote connections.
# Specifying the empty string '' (the default) allows all connections.
#(xend-address '')
(xend-address localhost)
```

Рекомендуется, что в случае, когда поддержка миграции не нужна, параметр `xend-relocation-server` устанавливать в `no` или комментировать.

Xm

Программа **xm** - это главный инструмент по управлению Xen из консоли. Общий формат командной строки **xm** такой:

```
# xm команда [ключи] [аргументы] [переменные]
```

Доступные *ключи* и *аргументы* зависят от того, какая команда выбрана. *Переменные*, указанные в командной строке, перекрывают значения переменных, заданные в конфигурационном файле, включая стандартные вышеописанные переменные (например, в файле `xdemconfig` используется переменная `vmid`).

Для того чтобы посмотреть справку самой программы, введите:

```
# xm help
```

Будет показан список наиболее часто использующихся команд. Полный список можно получить по команде `xm help --long`. Еще можно ввести `xm help команда` для просмотра справки по указанной `команде`.

Базовые управляющие команды

Одна из очень полезных команд:

```
# xm list
```

Она выводит информацию о доменах в формате

```
name domid memory vcpus state cputime
```

Назначение полей:
name
Имя виртуальной машины.

domid
Номер домена, в котором выполняется виртуальная машина.

memory
Объем памяти машины, в мегабайтах.

vcpus
Количество виртуальных процессоров, которые есть у домена.

state
Состояние домена описывается пятью полями:
r - работает; b - заблокирован; p - приостановлен; s - остановлен; c - упал.
cputime
Суммарное процессорное время (в секундах), которое использовал домен.

Команда `xm list` также поддерживает длинный формат вывода, с ключом `-l`. Эта команда выводит детальную информацию о доменах в формате xend SXP.

Если вы хотите узнать, сколько уже работают ваши домены, дайте команду:

```
# xm uptime
```

Доступ к консоли домена можно получить с помощью команды `xm console`. Например:

```
# xm console myVM
```

Команды управления планировщиком доменов

Планировщик атоматически распределяет гостевые виртуальные процессоры между всеми доступными физическими процессорами SMP системы. Привязывать вручную виртуальные процессоры к физическим нет необходимости. Однако, такая возможность есть. С помощью команды `xm vcpu-pin` можно указать на каких физических процессорах может работать виртуальный процессор машины.

У каждого гостевого домена есть два параметра: `weight`(вес) и `cap` (предел).

В нагруженной системе домен с весом `weight` 512 будет в два раза быстрее (по процессору) чем домен с весом 256. Допустимые значения весов находятся в диапазоне от 1 до 65535, значение по умолчанию равно 256.

Предел `cap` ограничивает максимальную величину процессорного ресурса, который может быть выделен гостевой системе, даже если хост-система простаивает. Предел выражается в процентах от одного физического процессора: 100 это один физический процессор, 50 это половина процессора, 400 -- это четыре процессора и т.д. Значение по умолчанию 0 означает что верхнего ограничения процессорного времени не существует.

Проверить и изменить значения веса и предела для каждого домена можно с помощью команд настройки планировщика (credit scheduler).

Показать вес `weight` и предел `cap` для *домена*:

```
# xm sched-credit -d домен
```

Установить *вес* для *домена*:

```
# xm sched-credit -d домен -w вес
```

Установить *предел* для *домена*:

```
# xm sched-credit -d домен -c предел
```

Конфигурация домена

Здесь описывается формат конфигурационного файла домена, а также дополнительная информация о настройке сети в домене, драйверов и планировщика.

Конфигурационные файлы

В конфигурационных файлах Xen могут присутствовать перечисленные ниже параметры. Если не указано обратное, конфигурационные параметры заключаются в кавычки. В каталоге `/etc/xen/` есть конкретные примеры.

kernel
Путь к образу ядра.

ramdisk
Путь к образу виртуального диска (не обязательно).

memory
Объём память в мегабайтах.

vcpus
Количество виртуальных процессоров.

console
Порт, на котором будет доступна консоль (по умолчанию 9600 + domain ID).

vif
Конфигурация сетевых интерфейсов. Может содержать пустую строку для каждого интерфейса, а может перекрывать значения по умолчанию, например:

```
vif = [ 'mac=00:16:3E:00:00:11, bridge=xen-br0',
        'bridge=xen-br1' ]
```

Запись означает: установить указанный MAC-адрес на первый интерфейс и подключить его к мосту `xen-br0`; подключить второй интерфейс к мосту `xen-br1`. Параметры, которые могут быть переопределены: тип `type`, мост `bridge`, IP-адрес `ip`, скрипт `script`, базовое устройство `backend` и имя интерфейса `vifname`.

disk
Список блочных устройств, которые должны быть экспортированы в домен. Например: ~disk = ['phy:hda1,sda1,r']%~ экспортирует физическое устройство `/dev/sda1` в режиме только-для-чтения. Экспортировать в режиме чтение-запись смонтированный диск опасно. Если вы всё же хотите это сделать, нужно указать символы `w!` в качестве режима.

dhcp
Нужно установить равным 'dhcp', если для конфигурирования сети будет использоваться этот протокол.

netmask
Сетевая маска.

gateway
IP-адрес шлюза.

hostname
Имя виртуального хоста.

root
Имя корневого раздела, передаваемое ядру системы.

nfs_server
IP-адрес NFS сервера, если используется.

nfs_root
Путь к корневому каталогу на NFS-сервере, если используется.

extra
Дополнительные параметры, которые передаются ядру (не обязательно).

Есть другие конфигурационные параметры (например, для конфигурирования виртуальных TPM).

Для повышения гибкости существует возможность включать команды Python непосредственно в конфигурационный файл. Пример того, как это можно сделать, находится в файле `xmexample2`, где команды Python используются для обработки переменной `vmid`.

Конфигурирование сети

Для большинства пользователей подойдёт конфигурация сети ``прямо из коробки'', доступная непосредственно после установки. Более сложные сетевые инсталляции, например, с множеством Ethernet-интерфейсов и/или коммутатороми, требуют дополнительной настройки.

Эта секция описывает механизмы, которые xend предоставляет с целью гибкого конфигурирования виртуальной сети Xen.

Топология виртуальной сети Xen

Каждый сетевой интерфейс домена соединён с виртуальным сетевым интерфейсом домена dom0 с помощью соединения точка-точка (фактически, "виртуальным кроссовером"). Эти устройства называются vif<domid>.<vifid> (например, `vif1.0` для первого интерфейса домена 1, `vif3.1` для второго интерфейса домена 3).

Трафик на этих виртуальных интерфейсах обрабатывается в домене 0 с помощью стандартных механизмов Linux для коммутации, маршрутизации, ограничения трафика и т.д. Xend для выполнения начальной конфигурации сети и виртуальных интерфейсов вызывает два сценария командного интерпретатора. По умолчанию, эти сценарии настраивают один мост (bridge) для всех виртуальных интерфейсов. Маршрутизация / коммутация произвольной конфигурации может быть настроена путем изменения скриптов, как описано ниже.

Сетевые скрипты Xen

Виртуальная сеть Xen конфигурируется двумя скриптами (по умолчанию, `network-bridge` и `vif-bridge`). Они вызывются xend автоматически при появлении определённых событий. Аргументы скрипта предоставляют дополнительную контекстную информацию. По умолчанию, скрипты находятся в `/etc/xen/scripts`. Местоположение и имена скриптов могут быть сконфигурированы через `/etc/xen/xend-config.sxp`.

network-bridge:
Этот скрипт вызывается при запуске или остановке xend для того чтобы инициализировать или разорвать виртуальную сеть Xen. Скрипт стандартной конфигурации создаёт мост xen-br0 и переносит eth0 на этот мост, изменяя настройки маршрутизации соответствующим образом. Когда xend завершается, он удаляет мост Xen и eth0, возвращая нормальную конфигурацию IP

и маршрутизации.

vif-bridge:
Этот скрипт вызывается для каждого виртуального интерфейса домена. Он может настраивать правила брандмауэра, и подключать vif к соответствующему мосту. По умолчанию, скрипт добавляет (или удаляет) виртуальные интерфейсы к мосту Xen по-умолчанию.

Существуют примеры скриптов (`network-route`, `vif-route`, `network-nat` и `vif-nat`). Для более сложных сетевых конфигураций (например, где требуется маршрутизация или интеграция с существующими мостами) эти скрипты могут заменяться другими доработанными вариантами.

Конфигурирование драйвера домена

PCI

Отдельные PCI устройства могут быть назначены конкретному домену (домен драйвера PCI), для того чтобы организовать прямой доступ из домена к PCI-устройствам.

Хотя, с одной стороны, домены PCI драйверов могут повысить стабильность и безопасность системы, остаётся несколько вопросов, связанных с безопасностью, о которых написано здесь.

Настройка во время компиляции

Для того чтобы использовать этот функционал, убедитесь что поддержка PCI Backend вкомпилирована в ядро привилегированного домена (домена 0), а в ядра, которые будут запускаться в пользовательских доменах и которые собираются использовать PCI-устройства, вкомпилирована поддержка PCI Frontend. В XenLinux PCI Backend включается в конфигурационной секции `Xen`, а PCI Frontend включается в зависимой от архитектуры секции "`Bus Options`". Можно вкомпилировать вместе в одно ядро поддержку backend'a и поддержку frontend'a. Они друг другу не мешают.

Конфигурирование PCI Backend'a. Связывание при загрузке

PCI-устройство, которое вы хотите назначить непривилегированному домену, должно быть скрыто от backend-домена (обычно домена 0), и чтобы домен не загружал драйвер для этого устройства. Для этого используется параметр ядра `pciback.hide`, который указывается в командной строке ядра через GRUB (см. здесь). Обратите внимание, что в действительности устройства не скрыты от backend-домена. PCI Backend видится Linux как обычное PCI-устройство. PCI Backend привязывает себя как драйвер к какому-то устройству, и тем самым гарантирует, что никакие драйверы устройств для него больше загружатся не будут. PCI-устройства идентифицируются шестнадцатеричными числами слот/функция (slot/function), в Linux эти номера можно определить с помощью программы **lspci**. Их можно указывать с информацией о номере домена или без неё:

```
        (bus:slot.func) например (02:1d.3)
(domain:bus:slot.func) например (0000:02:1d.3)
```

Пример параметров ядра Linux, скрывающих два PCI-устройства:

```
root=/dev/sda4 ro console=tty0 pciback.hide=(02:01.f)(0000:04:1d.0)
```

Конфигурирование PCI Backend'а. Позднее связывание

С помощью средств, предоставляемых Linux-ядром в sysfs, PCI-устройства можно привязывать к PCI-backend'у после загрузки (что позволяет предоставлять PCI-устройства драйверным доменам, которые не были указаны в командной строке ядра). Есть несколько атрибутов в каталоге PCI-backend'а в sysfs, которые можно использовать для того чтобы привязывать/ отвязывать устройства:

slots
Показать список всех PCI-слотов, которые PCI-backend будет пытаться захватить (или "скрыть" от домена 0). Для того чтобы PCI-слот можно было привязать к PCI-backend'у посредством bind-атрибута, слот должен сначала быть в этом списке.

new_slot
Для того чтобы PCI-backend захватил какой-то слот, он должен быть указан здесь (в формате 0000:00:00.0).

remove_slot
Укажите здесь название слота (в таком же формате как `new_slot`), чтобы PCI-backend не пытался более захватить устройство в этом слоте. Обратите внимание, что с помощью этого нельзя отвязать драйвер от устройства, которое уже захвачено.

bind
Для того чтобы Linux-ядро пыталось привязать устройство в этом слоте к PCI-backend драйверу, нужно указать его в этом параметре (в формате 0000:00:00.0)

unbind
Напишите здесь имя слота (в таком же формате как для bind), для того чтобы Linux-ядро отвязало устройство от PCI-backend'а. Не отвязывайте устройство, пока оно отнесено к драйверному домену PCI!

Несколько примеров:

Привязать устройство к PCI Backend'у, который больше ни к чему не привязан:

```
# # Add a new slot to the PCI Backend's listO
# echo -n 0000:01:04.d > /sys/bus/pci/drivers/pciback/new_slot
# # Now that the backend is watching for the slot, bind to it
# echo -n 0000:01:04.d > /sys/bus/pci/drivers/pciback/bind
```

Отвязать устройство от драйвера и привязать к PCI Backend'у:

```
# # Unbind a PCI network card from its network driver
# echo -n 0000:05:02.0 > /sys/bus/pci/drivers/3c905/unbind
# # And now bind it to the PCI Backend
# echo -n 0000:05:02.0 > /sys/bus/pci/drivers/pciback/new_slot
# echo -n 0000:05:02.0 > /sys/bus/pci/drivers/pciback/bind
```

Обратите внимание на опцию `-n` в примере. Она заставляет echo не выводить перевод строки после текста, что имеет здесь принципиальное значение.

Конфигурирование PCI Backend'а. User-space quirks

Некоторым устройствам (например, Broadcom Tigon 3) может понадобиться доступ на запись к своим конфигурационным space регистрам. Xen'у можно указать для каких PCI-устройств есть доступ на запись в специфичные конфигурационные space регистры. Политика находится здесь:

```
/etc/xen/xend-pci-quirks.sxp
```

Файл содержит большое количество комментариев достаточных для того чтобы можно было его изменять и расширять.

Конфигурирование PCI Backend'а. Флаги разрешения

Если с помощью user-space quirks решить задачу не удалось, возможно, задачу можно будет решить с помощью permissive-флага для соответствующего устройства. Для этого прежде всего с помошью lspci нужно узнать PCI-домен, шину, слот и функцию этого устройства в домене dom0. После чего расширить user-space политику для permissive устройств. Permissive политика находится в:

```
/etc/xen/xend-pci-permissive.sxp
```

В настоящий момент единственный способо сбросить permissive-флаг это отвязать утсройство от PCI-backend драйвера.

Конфигурирование PCI Backend'а. Проверка состояния

Есть два важных узла sysfs, которые предоставляют механизм для просмотра специфичной информации о quirks- и permissive-устройствах:

```
/sys/bus/drivers/pciback/permissive
```

Просматривая с помощью cat этот файл, можно увидеть список permissive-слотов.

```
/sys/bus/drivers/pciback/quirks
```

Просматривая с помощью cat этот файл, можно увидеть иерархичесое представление устройств, привязанных к PCI-backend'у, их PCI vendor-ID и device-ID, а также все quirks, которое связаны с этим конкретным слотом.

Можно заметить, что у каждого устройства, привязанного к PCI-backend'у есть 17 стандартных quirks, независимо от того, что в `xend-pci-quirks.sxp`. Эти записи по умолчанию необходимы для обеспечения взаимодействия между менеджером шины PCI и устройствами, привязанными к ней. Даже у не-quirky устройств должны быть стандартные записи.

В данном случае эстетичность была принесена в жертву точности, и стандартные quirks показываются в общем списке quirks, а не скрываются.

Конфигурирование PCI Frontend'а

Есть несколько способов сконфигурировать пользовательский домен domU, чтобы он получил

PCI-устройство.

С помощью командной строки. Использовать опцию командной строки `pci`. Если нужно передать несколько устройств, опция должна быть указана несколько раз.

```
xm create netcard-dd pci=01:00.0 pci=02:03.0
```

С помощью конфигурационного файла. Указать все необходимые устройства в конфигурационном файле домена.

```
pci=['01:00.0','02:03.0']
```

С помощью конфигурационного файла в формате SXP. Все PCI-устройства описываются в одной секции. Числа указываются в шестнадцаретичной системе счисления, начиная с символов `0x`. Обратите внимание, что слово domain здесь относится к домену PCI, не домену виртуальной машины.

```
(device (pci
    (dev (domain 0x0)(bus 0x3)(slot 0x1a)(func 0x1)
    (dev (domain 0x0)(bus 0x1)(slot 0x5)(func 0x0)
)
```

Поддержка virtual Trusted Platform Module (vTPM)

Паравиртуализированным доменам можно предоставить доступ к виртуализированной версии TPM. Это даёт возможность приложениям в этих доменах использовать TPM-устройства, например, через TSS-стек Trousers TSS stack . В репозитории Xen есть все необходимые компоненты для обеспечения доступа к TPM. Поддержка выполняется за счёт нескольких вещей. Во-первых, модифицирован эмулятор TPM, для того чтобы он мог обеспечивать функциональность TPM для подсистемы виртуального TPM. Во-вторых, создан менеджер виртуального TPM, для того чтобы он мог координировать действия виртуальных TPM, управлять их созданием и provides protected key storage using the TPM. В-третьих, есть пара драйверов для XenLinux, представляющих TPM frontend и TPM backend. Драйверы нужны для передачи TPM-команд от домена менеджеру виртуального TPM, который отправляет их дальше программному TPM. Поскольку TPM Manager полагается на HW TPM для protected key storage, поэтому нужен аппаратный TPM, поддерживаемый Linux. Для разработчиков доступен эмулятор TPM, который можно использовать на платформах без аппаратной поддержки TPM.

Установка на этапе компиляции

Для включения доступа к virtual TPM, драйвер backend'a virtual TPM должен быть вкомпилирован в привилегированный домен (домен 0). В XenLinux необходимый драйвер можно выбрать в конфигурационной секции Xen. За исключением случая, когда драйвер вкомпилирован в ядро, для того чтобы активировать модуль необходимо дать следующую команду:

```
# modprobe tpmbk
```

Аналогичным образом, для поддержки функциональности TPM нужно вкомпилировать в ядро

драйвер TPM fontend. Драйвер можно выбрать при конфигурировании ядра в меню Device Driver / Character Devices / TPM Devices. Вместе с этим драйвером нужно выбрать и **TPM driver for the built-in TPM**. Если драйвер виртуального TPM был скомпилирован как модуль, его нужно активировать с помощью команды:

```
# modprobe tpm_xenu
```

Далее, нужно откомпилировать менеджер виртуального TPM и программный TPM. Для этого нужно внести изменения в конфигурацию сборки Xen. Нужно сдобавить следующую сторку в файл `Config.mk` в корневом каталоге дистрибутива Xen:

```
VTPM_TOOLS ?= y
```

После сборки дерева Xen и перезагрузки машины, нужно загрузить драйвер backend. После того как он загружен, нужно запустить домен менеджера виртуального TPM. Но это нужно сделать до того как будут запущены гостевые домены с поддержкой TPM. Для того чтобы можно было выполнять миграцию на этот хост, на нём также нужно запустить демон миграции виртуального TPM.

```
# vtpm_managerd
# vtpm_migratord
```

Как только VTPM менеджер работает, получить доступ к нему можно, загрузив драйвер в гостевой домен.

Разработка и тестирование эмулятора TPM

Для разработки и тестирования платформ без поддержки TPM, в качестве замены платформы TPM можно использовать эмулятор TPM.

Во-первых, запись в файле `tools/vtpm/Rules.mk` должна выглядеть следующим образом:

```
BUILD_EMULATOR = y
```

Во-вторых, запись в файле `tool/vtpm_manager/Rules.mk` должна быть раскомментирована:

```
# TCS talks to fifo's rather than /dev/tpm. TPM Emulator assumed on fifos
CFLAGS += -DDUMMY_TPM
```

Перед запуском менеджера virtual TPM, необходимо запустить эмулятор в домене dom0 с помощью такой команды:

```
# tpm_emulator clear
```

Конфигурирование vTPM Frontend'a

Для предоставления функциональности TPM домену пользователя, нужно добавить в конфигурационный файл виртуального TPM строку в следующем формате:

```
vtpm = ['instance=<instance number>, backend=<domain id>']
```

Параметр `instance number` это номер предпочитаемого виртуального TPM, который должен быть ассоциирован с доменом. Если указанный экземпляр уже ассоциирован с другим доменом, система автоматически выберет следующий доступный экземпляр. Обязательно должен быть указан номер экземпляра больший чем 0. Можно вообще не указывать параметр instance в конфигурационном файле.

Параметр `domain id` -- идентификатор домена, в котором работает драйвер backend'a виртуального TPM и сам TPM. В настоящий момент он всегда должен устанавливаться равным нулю.

Примеры записей vtpm в конфигурационном файле:

```
vtpm = ['instance=1, backend=0']
```

и

```
vtpm = ['backend=0'].
```

Использование virtual TPM

Доступ к функциональности TPM обеспечивается fontend-драйвером виртуального TPM. Этот драйвер предоставляет базовую информацию о состоянии TPM через файловую систему sysfs, точно так же как это делают существующие драйверы аппаратных TPM. В пользовательских доменах Xen эти записи sysfs можно найти в `/sys/devices/xen/vtpm-0`.

Отправлять команды virtual TPM можно с помощью символьного устройства `/dev/tpm0` (major 10, minor 224).

Управление хранилищами и файловыми системами

Дисковые хранилища для виртуальных машин можно сделать по-разному. В этой главе рассматриваются некоторые возможные конфигурации.

Наиболее простой и прямой способ это экспортировать физическое блочное устройство (жесткий диск или раздел) из домена 0 в гостевой домен как виртуальное блочное устройство (Virtual Block Device, VBD).

Хранилище может быть также экспортировано из файла-образа файловой системы, как основанное на файле устройство (file-backed VBD).

Кроме того, для предоставления хранилищ виртуальным машинам могут использоваться стандартные сетевые протоколы, такие как NBD, iSCSI и NFS.

Экспорт физических устройств как VBD

Одна из самых простых конфигураций -- напрямую экспортировать отдельные дисковые разделы из домена 0 в другие домены. Для это используется ключевое слово `phy:` в конфигурационном файле домена. Например:

```
disk = ['phy:hda3,sda1,w']
```

Эта строка указывает, что раздел `/dev/hda3` домена 0 должен быть экспортирован в режиме чтение/запись в новый домен как `/dev/sda1`. С таким же успехом при желании можно было экспортировать его как `/dev/hda` или `/dev/sdb5`.

Помимо дисков и разделов можно экспортировать любые другие устройства, к которым Linux относится как к дискам. Например, если у вас есть диск iSCSI или том GNBD импортированный в домен 0, можно экспортировать его с помощью того же синтаксиса `phy:`. Например:

```
disk = ['phy:vg/lvm1,sda2,w']
```

Warning: Block device sharing

Если блочное устройство используется совместно несколькими доменами, то это должно быть в режиме только-для-чтения, иначе модуль файловой системы ядра Linux может очень удивиться, когда увидит, что структура файловой системы, с которой он работает, не соответствует его ожиданиям (если один и тот же раздел дисковой системы, отформатированный под ext3, смонтировать в режиме чтение/запись дважды, почти наверняка это приведёт к непоправимым последствиям). Демон Xend пытается защитить вас от такой ошибки, проверяя не смонтирована ли уже какая-то файловая система в режиме чтение/запись или в самом домене 0, или в гостевых доменах. Если требуется разделение в режиме чтение/запись, нужно или экспортировать соответствующий каталог через NFS из домена 0 или использовать кластерные файловые системы, такие как GFS или ocfs2.

Использование виртуальных блочных устройств основанных на файлах

В качестве основного хранилища для виртуальной машины можно использовать файл в домене 0. Это удобно, а кроме того виртуальное блочное устройство может быть разряжённым -- пространство будет действительно выделяться, только по мере использования файла. Если виртуальная машина использует, скажем, половину своего дискового пространства, тогда файл займёт только половину заданного размера.

Например, для того чтобы создать разрежённое файловое устройство размером 2GB (в действительности потребляет только 1 KByte дискового пространства):

```
# dd if=/dev/zero of=vm1disk bs=1k seek=2048k count=1
```

Создайте в файле файловую систему:

```
# mkfs -t ext3 vm1disk
```

(в ответ на вопрос о подтверждении введите y)

Наполните файловую систему, например, путём копирования из корневого каталога.

```
# mount -o loop vm1disk /mnt
# cp -ax /{root,dev,var,etc,usr,bin,sbin,lib} /mnt
# mkdir /mnt/{proc,sys,home,tmp}
```

Нужно доделать систему: отредактировать файл `/etc/fstab`, `/etc/hostname`. Нужно редактировать файлы на смонтированной новой файловой системе, а не на старой из домена 0, поэтому пути будут такими: `/mnt/etc/fstab` вместо `/etc/fstab`. Для нашего примера в `fstab` должна появится запись `/dev/sda1` для корневого раздела.

После этого размонтируйте (это важно):

```
# umount /mnt
```

В конфигурационном файле нужно установить:

```
    disk = ['tap:aio:/full/path/to/vm1disk,sda1,w']
```

По мере того как виртуальная машина будет писать на свой диск, разреженный файл будет наполяться и потреблять больше места, вплоть до заданных 2GB.

Замечание: Пользователям, которые работали с базирующимися на файлах виртуальными блочными устройствами в Xen предыдущих версий, будет интересно узнать, что сейчас поддержка этой функции выполняется с помощью драйвера blktap вместо loopback как раньше. В результате этих изменений файловые устройства стали более производительными, масштабируемыми и более надежными для хранения данных виртуальных блочных устройств. Для того чтобы использовать blktap вместо loop, достаточно изменить записи в конфигурационных файлах, с таких:

```
    disk = ['file:/full/path/to/vm1disk,sda1,w']
```

на такие:

```
    disk = ['tap:aio:/full/path/to/vm1disk,sda1,w']
```

Файловые виртуальные блочные устройства, работающие через loopback (устарело)

Замечание: Как сказано выше, виртуальные блочные устройства, монтирующиеся через VBD заменены устройствами, базирующими на blktap. В этом разделе приводится информация, которая может быть полезна при чтении конфигурационных файлов Xen более старых версий.

Файл с образом для виртуального блочного устройства можно подключить к виртуальной машине с помощью драйвера loopback в Linux. Единственное отличие от директив конфигурационного файла, описанных выше, в том что нужно указывать ключевое слово `file:`

```
    disk = ['file:/full/path/to/vm1disk,sda1,w']
```

Следует иметь в виду, что виртуальные блочные устройства, работающие через loopback, могут не подойти для использования в доменах с интенсивным вводом/выводом. При таком подходе возможны существенные потери производительности при больших величинах потоков ввода/вывода, связанные с тем как loopback-устройства обрабатывают ввод/вывод. Поддержка loopback-устройств оставлена для совместимости со старыми инсталляциями Xen; новым пользователям настоятельно рекомендуется использовать поддержку устройств, базирующихся на blktap (используя "tap:aio" как показано выше).

И ещё, Linux по умолчанию поддерживает только максимум 8 работающих через loopback блочных устройств для всех доменов вместе взятых. Этот предел можно статически увеличить параметром `max_loop` для модуля loop, при условии, что модуль скомпилирован отдельно от ядра (CONFIG_BLK_DEV_LOOP=M) или с помощью опции `max_loop` переданной непосредственно ядру при загрузке, в том случае, если модуль вкомпилирован в ядро (CONFIG_BLK_DEV_LOOP=Y). И здесь также пользователю рекомендуется использовать blktap, поскольку он масштабируется до намного большего количества активных VBD.

Использование виртуальных блочных устройств основанных на томах LVM

Особо привлекательной выглядит возможность хранения файловых систем домена на томах LVM, поскольку с их помощью можно легко увеличивать/уменьшать размеры томов, создавать снимки файловых систем (filesystem snapshots) и делать другие полезные вещи.

Для того чтобы инициализировать поддержку LVM-томов дисковым разделом, нужно сделать с ним следующее:

```
# pvcreate /dev/sda10
```

Создайте группу томов (volume group), названную vg на физическом разделе (точнее, создайте группу томов, в которую будет входить физический том, созданный на предыдущем шаге):

```
# vgcreate vg /dev/sda10
```

Создайте догический том размером 4G, названный "myvmdisk1":

```
# lvcreate -L4096M -n myvmdisk1 vg
```

Появится устройство `/dev/vg/myvmdisk1`. Создайте файловую систему, смонтируйте её и заполните нужными файлами, например так:

```
# mkfs -t ext3 /dev/vg/myvmdisk1
# mount /dev/vg/myvmdisk1 /mnt
# cp -ax / /mnt
# umount /mnt
```

После этого измените конфигурацию виртуальной машины:

```
    disk = [ 'phy:vg/myvmdisk1,sda1,w' ]
```

LVM позволяет увеличивать размер логических томов, но нужно ещё изменить размер и соответствующей файловой системы, для того чтобы можно было использовать новое пространство. Некоторые файловые системы (например, ext3) уже поддерживают изменения размеров в горячем режиме. Подробности можно найти в документации по LVM.

Кроме этого можно использовать LVM для создания copy-on-write (CoW) клонов томов LVM (в терминологии LVM известные также как записываемые постоянные снимки, writeable persitent snapshots). Это средство появилось в Linux 2.6.8, и оно ещё не стало достаточно стабильным и зрелым. В частности, использование большого количества LVM-дисков требует много памяти домена 0, а также обработка ошибочных условий, таких как переполнение диска, выполняется не очень хорошо. Есть надежда, что ситуация изменится в будущем.

Создать два клона copy-on-write вышеуказанных файловых систем можно с помощью следующих команд:

```
# lvcreate -s -L1024M -n myclonedisk1 /dev/vg/myvmdisk1
# lvcreate -s -L1024M -n myclonedisk2 /dev/vg/myvmdisk1
```

Каждый из них может увеличиваться до тех пор пока не наберётся 1G отличий от основного тома. Увеличить величину пространства для хранения отличий можно с помощью команду lvextend, например:

```
# lvextend +100M /dev/vg/myclonedisk1
```

Нельзя позволять томам разницы заполняться, иначе LVM начинает сбиваться. Можно автоматизировать процесс расширения с помощью программы **dmsetup**, котороя наблюдает за томом, и как только он заполняется -- вызывает программу **lvextend** с целью его расширения.

В принципе, можно продолжать запись на том, который был склонирован (изменения не будет видны клонам), но это не рекомендуется: пусть лучше этот том будет нетронутым, и пусть он не монтируется ни одной из виртуальных машин.

Использование корневой файловой системы NFS

Сначала нужно наполнить каталог сервера, соответствующий корневой файловой системе домена. Сервером может быть выделенная машина или, например, виртуальная машина, работающая на том же узле.

После этого нужно изменить конфигурацию NFS-сервера так чтобы он экспортировал эту файловую систему по сети. Для этого в файл `/etc/exports` нужно добавить, например, такие строки:

```
/export/vm1root      1.2.3.4/24 (rw,sync,no_root_squash)
```

Последнее, нужно сконфигурировать домен для использования корневой файловой системы NFS. В дополнение к обычным переменным нужно установить ещё следующий переменные в конфигурационном файле домена:

```
root       = '/dev/nfs'
nfs_server = '2.3.4.5'       # substitute IP address of server
nfs_root   = '/path/to/root' # path to root FS on the server
```

Домену нужен сетевой доступ во время загрузки, поэтому нужно или статически задать IP-адрес и прочие настройки с помощью переменных `ip`, `netmask`, `gateway`, `hostname`; или же включить DHCP (`dhcp='dhcp'`).

Следует иметь в виду, что NFS в качестве корневой файловой системы в Linux имеет проблемы со стабильностью при больших нагрузках (эта проблема не специфична для Xen), так что такую конфигурацию лучше не использовать для критически важных серверов.

Управление ресурсом процессора

Xen позволяет ассоциировать виртуальные процессоры домена с одним или более процессорами хоста. Это может помочь распределить ресурсы системы между гостевыми средами или сделать более оптимальным использование ресурсов в двухядерных процессорах и процессорах с гипертредингом (hyperthreading) и другими продвинутыми технологиями.

Xen нумерует процессоры "сначала в глубину". Для многоядерных процессоров с поддержкой гипертрединга сначала будут пронумерованы гипертреды (hyperthreads) в пределах ядра, затем все ядра, а потом гнезда. Например, если в системе два двухъядерных процессора с поддержкой гипертрединга, нумерация будет такой:

```
+----------------------------+------------------------------+
|          socket0           |             socket1          |
|-------------+--------------+----------------+-------------+
|   core0     |     core1    |      core0     |     core1   |
|-------------+--------------+----------------+-------------+
| ht0   ht1   | ht0     ht1  |  ht0     ht1   | ht0     ht1 |
+-------------+--------------+----------------+-------------+
  #0    #1      #2      #3      #4      #5      #6      #7
```

Если привязать несколько виртуальных процессоров, принадлежащих одному домену, к одному и тому же процессору, скорее всего производительность будет низкой.

Если выполняется задача, требующая интенсивного ввода-вывода, чаще всего лучше выделить целиком одни гипертред (или ядро) для домена 0 и назначить остальные домены так, чтобы процессор 0 гостевыми доменами не использовался. Если рабочая нагрузка является преимущественно вычислительной, возможно, стоит назначить процессоры так, чтобы все физические процессоры были доступны для гостевых доменов.

Миграция доменов

Сохранение и восстановление доменов

У администратора Xen есть возможность сохранить текущее состояние на диск домена 0, чтобы затем продолжить его выполнение позже.

Например, сохранить домен VM1 на диск можно командой:

```
# xm save VM1 VM1.chk
```

Домен будет остановлен, а его состояние записано в файле `VM1.chk`.

Для того чтобы продолжить выполнение домена, используется команда `restore`:

```
# xm restore VM1.chk
```

Эта команда восстановит состояние домена и продолжит его выполнение. Домен будет выполняться как и раньше, а к его консоли можно будет подключиться с помощью вышеописанной команды `xm console`.

Миграция и живая миграция

Миграция используется для перемещения доменов между физическими хостами. Существует две разновидности миграции: обычная и живая. При обычной миграции для перемещения машины она останавливается, содержимое её памяти копируется на другой хост, после чего работа машины возобновляется на новом хосте. При живой миграции выполняются те же действия, только не нужно останавливать работу машины. Во время выполнения живой миграции домен продолжает работать как и обычно и, с точки зрения пользователя, заметить её невозможно.

Для выполнения живой миграции необходимо чтобы обе системы работали под управлением Xen, и в них был запущен xend. На системе, куда переносится домен, должно быть достаточно ресурсов (в частности, памяти) для перенесённого домена. В настоящий момент также требуется чтобы обе системы находились на канальном уровне в одной сети.

В настоящий момет при миграции домена не существует возможности автоматического удалённого доступа к локальным файловым системам. Администратор должен выбрать соответствующее хранилище (SAN, NAS и т.д.) для того чтобы доменная файловая система была доступна на обоих хост-системах. Хороший способ экспортировать том с одной машины на другую -- это GNBD. iSCSI может делать похожую работу, но его сложнее настраивать.

Когда мигрирует домен, его MAC и IP адрес мигрируют вместе с ним, поэтому миграция возможна только в пределах одной сети на канальном уровне. Если узел, на который переносится домен, находится в другой сети, нужно настроить туннелирование IP-пакетов на удалённый хост.

Миграция выполняется с помощью команды `xm migrate`. Для того чтобы выполнить живую миграцию на другую машину, нужно использовать команду:

```
    # xm migrate --live mydomain destination.ournetwork.com
```

Без ключа `--live` xend просто останавливает домен, копирует его образ по сети и запускает его заново. Поскольку у домена может быть большой объём памяти, перенос может потребовать достаточно много времени даже в гигабитной сети. С ключом `--live` xend пытается перенести домен в работающем состоянии. Простои при переносе составляют время порядка 60-300 мс.

Сейчас пока нужно соединяться заново с консолью домена на новой машине с помощью команды `xm console`. Если у мигрирированного домена были какие-то сетевые соединения, они сохранятся, поэтому SSH соединения не имеют указанного ограничения.

Безопасность Xen

В этой главе рассматривается как повысить безопасность Xen-системы. Описан ряд сценариев и хороших примеров. Начинается глава разделом, который посвящён общим вопросам безопасности Xen.

Рекомендации по безопасности Xen

При внедрении Xen системы нужно быть уверенным, что домен 0 находится в максимальной безопасности. Если взломан управляющий домен, все остальные домены автоматически оказываются под угрозой. Рекомендации для домена 0:

1. **Запускать как можно меньшее число сервисов, ничего лишнего.** Чем меньше вещей присутствует в управляющем домене, тем лучше.
2. **Использовать брандмауэр для ограничения трафика к управляющему домену.** Брандмауэр с правилами по умолчанию настроенными на запрет поможет предотвратить атаки на управляющий домен.
3. **Не допускать пользователей к домену 0.** Известно, что существуют локальные эксплоиты для ядра Linux, позволяющие получить права root'а. Если у обычных (даже не обязательно привилегированных) пользователей есть возможность работать внутри домена 0, из-за локальных эксплоитов ядра существует риск для всех доменов системы.

Драйверные домены и безопасность

Драйверные домены используются для решения проблем с безопасностью, связанных с использованием драйверов устройств. Во многих операционных системах в настоящее время драйверы работают в пространстве ядра, с тем же уровнем привилегий, что и само ядро. Механизмов, которые могли бы защитить ядро от неверно ведущего себя (читай "глючного") или злонамеренного драйвера, мало, или они вообще отсутствуют. Драйверные домены нужны чтобы помочь изолировать драйвер устройства внутри собственной виртуальной машины, где он не может повлиять на стабильность и целостность других доменов. Если драйвер начинает сбоить, вместо того чтобы перезагружать всю систему, можно перезагрузить только драйверный домен. Драйверы, написанные третьими лицами, не обладающими достаточным уровнем доверия, можно изолировать от других. Драйверные домены решают часть проблем безопасности и стабильности, связанных с драйверами устройств.

Однако, из-за ограничения существующего оборудования, при использовании драйверных доменов нужно иметь в виду несколько вещей (этот список не полный):

1. **Без IOMMU оборудование может выполнять прямое обращение к памяти (DMA) к областям памяти за пределами управляющего домена.** Уязвимыми являются архитектуры, у которых нет модуля IOMMU, позволяющего ограничивать использование DMA (т.е. большинство x86-платформ). Устройство, которое может читать и изменять произвольные участки памяти, может читать и писать данные, находящиеся за пределами своего домена. Злонамеренный или работающий с ошибками домен может использовать подконтрольное ему устройство для записи или чтения данных в произвольных местах памяти другого домена.
2. **Разделяемые шины могут прослушиваться.** Устройства, использующие совместно шину данных, могут прослушивать (или даже подменять) данные друг друга. Устройство A, назначенное домену A, может прослушивать данные, которые домен B передаёт устройству B, и передавать их домену A.
3. **Устройства, которые используют линии прерываний совместно с другими, могут при желании или помешать получению прерываний другим устройствам, или, наоборот, генерировать бесполезные прерывания.** Устройства, совместно использующие level-triggered прерывания (например, PCI устройства) могут вызывать прерывания и никогда их не сбрасывать. Это блокирует для других устройств, которые

используют эту же линию прерывания, возможность передачи сигнала о необходимости обслуживания своим управляющим драйверам. Устройства, использующие совместно линии прерываний любого типа, могут постоянно вызывать прерывания, что приведёт (причём в нескольких гостевых системах) к необходимости вызова процедур обработки прерываний (что потенциально может привести к тому, что другие процессы внутри соответствующего гостевого домена вообще не получат управления). Архитектура, в которой у каждого устройства может быть своя собственная линия прерываний (например, Message Signaled Interrupts шины PCI), подвержена этой проблеме в меньшей степени.
4. **Устройства могут совместно использовать адресное пространство ввода/вывода.** Xen может ограничивать доступ к устройствам ввода/вывода только с определённым уровнем гранулярности. Для линий прерываний и портов ввода/вывода уровень гранулярности очень высокий (с точностью до одной линии или одного порта). Однако, что касается адресного пространства ввода/вывода, ограничение может выполняться только по границам страниц. Если несколько устройств совместно используют страницу в пространстве ввода/вывода, домены, которым принадлежат эти устройства, будут иметь возможность доступа к адресному пространству устройств друг друга.

Сценарии

Изолированная управляющая сеть

В этом сценарии у каждого узла есть две сетевые карты. Одна из них выходит во внешний мир, а вторая -- в физически изолированную управляющую сеть, предназначенную специально для Xen.

До тех пор пока все управляемые части имеют одинаковый уровень доверия, это наиболее безопасный сценарий. Никакой дополнительной конфигурации он не тербует, за исключением того, что Xend для перемещения должен быть привязан к управляющему интерфейсу.

Подсеть за брандмауэром

В этом сценарии у каждого узла есть только одна сетевая карта, но весь кластер находится за брандмауэром. В этом случае брандмауэр должен делать как минимум следующее:

1. Предотвращать IP-spoofing снаружи сети.
2. Предотвращать доступ к порту перемещения (relocation) на всех узлах для всех, кроме самого кластра.

Такие правила можно использовать на каждом узле для запрета миграции снаружи сети, если главный брандмауэр не выполняет запрет:

```
# эта команда закрыват любой доступ к порту
# перемещения (relocation) Xen
iptables -A INPUT -p tcp --destination-port 8002 -j REJECT

# этак команда разрешает перемещение (relocation)
# только из определенной подсети
iptables -I INPUT -p tcp --source 192.168.1.1/8 \
    --destination-port 8002 -j ACCEPT
```

Узлы в небезопасных сетях

В текущих версиях Xen миграция в небезопасных сетях также является небезопасной. Можно выполнять миграцию поверх безопасных туннелей VPN и SSH. Единственный безопасный вариант (за исключением туннелей) это отключение миграции как таковой. Проще всего это сделать с помощью iptables:

```
# эта команда закрыват любой доступ к порту
# перемещения (relocation) Xen
iptables -A INPUT -p tcp --destination-port 8002 -j REJECT
```

Управление доступом sHype/Xen

Framework принудительного контроля доступа (mandatory access control) это реализация sHype Hypervisor Security Architecture (www.research.ibm.com/ssd_shype). Она разрешает или запрещает обмен информацией и доступ к ресурсам на основе политики безопасности. Механизмы принудительного контроля доступа дополняют основные механизмы управления Xen, такие как механизмы защиты памяти. Они спроектированы таким образом, чтобы быть незаметными в ходе нормальной работы домена (policy-conform behavior), но в случае выхода домена за границы допустимого поведения быть готовыми вмешаться и ограничить его. В этой главе описывается как с помощью механизмов контроля доступа в Xen предотвратить распространение вирусов между доменами и утечку информации из одних нагрузок в другие. sHype/Xen зависит от правильного поведения домена 0.

Преимущества sHype/ACM в Xen:

- развитую защиту нагрузок и ресурсов, эффективно справляющуюся с враждебными пользовательскми доменами;
- простые, независящий от операционной системы и платформы политики безопасности (идеально подходящие для распределённых гетерогенных сред);
- безопасность с минимальными затратами производительности в тех случаях, когда безопасность операционной системы отсутствует, не масштабируется или сбоит.

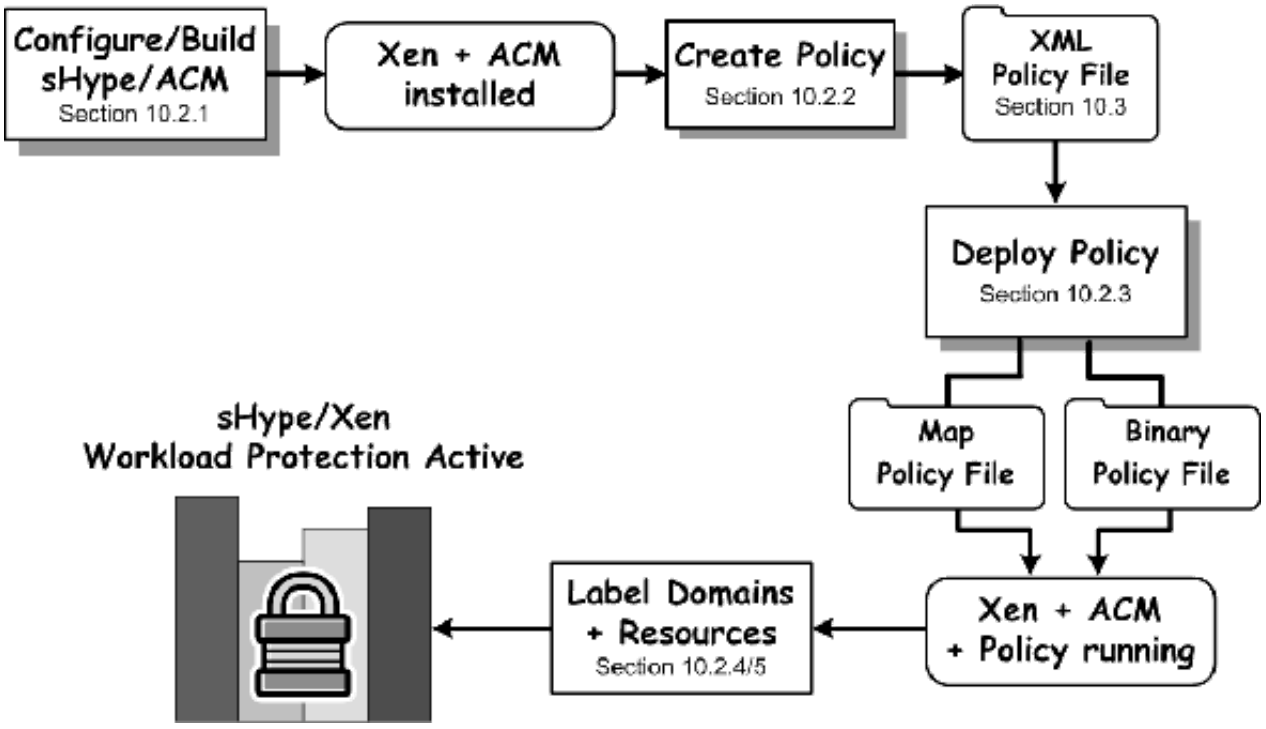
Эти преимущества являются чрезвычайно ценными, поскольку нынешние операционные системы стали очень сложными, и в них часто не хватает механизмов принудительного контроля доступа (механизмы дискреционного контроля доступа, поддерживаемые в большинстве операционных систем, недостаточно эффективны против вирусов и неверно ведущих себя программ). Там где принудительный контроль доступа существует (например, SELinux), он обычно сопряжён со сложными и трудными для понимания политиками безопасности. Кроме того, в многоуровневых приложения для бизнес-сред обычно требуется совместная работа нескольких операционных систем (например, AIX, Windows, Linux), которые нельзя сконфигурировать так чтобы они использовали совместимые политики безопасности. Связанные распределённые транзакции и нагрузки нельзя простым способом защитить на уровне операционной системы. Framework контроля доступа Xen предлагает хотя и крупноблочный, но достаточно зрелый уровень безопасности в том случае если безопасность на уровне операционной системы отсуствует или недостаточная.

Чтобы контролировать совместное использование ресурсов доменами, Xen регулирует как междоменное взаимодействие (разделяемая память, события), так и обращение доменов к дискам. Так, Xen может держать в заданных рамках распределённые нагрузки, позволяя совместное использование ресурсов доменам выполняющим нагрузку одного типа и запрещая его для доменов выполняющих нагрузку разных типов. Будем исходить из предположения, что с точки зрения Xen только один тип нагрузки работает в одном пользовательском домене. Чтобы Xen мог ассоциировать домены и ресурсы с типами нагрузок, к доменам и ресурсам прикрепляются метки безопасности (включая типы нагрузок). Эти метки и sHype-контроль

гипервизора нельзя обойти, и они являются эффективными даже против злонамеренных доменов.

Обзор

В этом разделе проводится обзор того как framework принудительного контроля доступа sHype может выполнять защиту нагрузок в Xen. На рисунке показаны необходимые для активации защиты нагрузок в Xen. Эти шаги подробно описаны здесь.



Обзор процесса активации защиты нагрузок sHype в Xen

Во-первых, средство управления доступом sHype/ACM должно быть собрано и проинсталлировано (см. здесь). Во-вторых, для того чтобы включать политику безопасности Xen, она сначала должна быть создана (см. здесь) и внедрена (см. здесь). Политика определяет различные виды нагрузок, по которым делаются разграничения в ходе управления доступом. Кроме этого, она определяет различные правила, которые сравнивают типы нагрузок доменов и ресурсов и, основываясь на этом, принимают решение о разрешении или запрете доступа. Типы нагрузок представлены метками безопасности, которые могут прикрепляться к доменам и ресурсам (см. эту и эту секции). Защита sHype/Xen в действии продемонстрирована на простом примере. В этом разделе детально описывается синтаксис и семантика политики безопасности sHype/Xen, а также делается небольшое введение в использование программ, предназначенных для создания политик безопасности.

В следующем разделе описываются все шаги, необходимые для того чтобы создать, внедрить и проверить простую политику защиты нагрузки. С их помощью можно быстро попробовать механизм защиты нагрузки sHype/Xen. Те, кто хочет узнать больше о том, как работает контроль доступа sHype в Xen и как его настроить с помощью XML-политики безопасности, должны прочитать этот раздел. Этот раздел завершает главу обзором существующих ограничений реализации sHype в Xen.

Пошаговое руководство по защите нагрузок Xen

Процедура состоит из следующих шагов:

- сконфигурировать и проинсталлировать sHype/Xen;
- создать простую политику безопасности защиты нагрузки;
- внедрить политику безопасности sHype/Xen;
- ассоциировать домены и ресурсы с метками нагрузок;
- проверить как выполняется защита нагрузок.

Важные команды, необходимые для создания и внедрения политики безопасности sHype/Xen, встречающиеся в следующих разделах, пронумерованы. Если вам понадобится быстрое руководство, или надо будет на скорую руку выполнить те действия, которые здесь описываются, просто ищите пронумерованные команды и применяйте их в соответствующем порядке.

Включение поддержки sHype в Xen

Прежде всего нужно настроить модуль контроля доступа в Xen и проинсталлировать гипервизор с поддержкой ACM. На этом шаге устанавливаются инструменты безопасности и sHype/ACM вкомпилируется в гипервизор.

Для того чтобы включить поддержку sHype/ACM в Xen, нужно отредактировать файл `Config.mk` в корневом каталоге дистрибутива Xen.

```
(1) In Config.mk
      Change: ACM_SECURITY ?= n
          To: ACM_SECURITY ?= y
```

После этого нужно проинсталлировать Xen обычным способом:

```
(2) # make world
    # make install
```

Создание политики нагрузки в три шага с помощью ezPolicy

Для быстрого создания политик, защищающих нагрузки, мы будем использовать программу ezPolicy. Для запуска этой программы требуются Python и wxPython. Для того чтобы запускать программу из домена Domain0, можно скачать wxPython с сайта www.wxpython.org или воспользоваться командой `yum install wxPython` в Redhat/Fedora. Если выполнять программу ezPolicy под Windows, также нужно будет скачать пакет Python с сайта www.python.org. После того как все необходимые пакеты установлены, запустите ezPolicy с помощью команды:

```
(3) # xensec_ezpolicy
```

На рисунке показан скриншот программы. Создать политику, показанную на рисунке, можно с помощью таких шагов. Справка доступна в любой момент с помощью комбинации <CTRL>-h. Маркеры (a), (b) и (c) на рисунке указывают на кнопки, которые используются в ход создания политики "в три шага":

1. описание нагрузки;
2. описание конфликтов;
3. преобразование описания нагрузки в политику sHype/Xen.

Описание нагрузок

Нагрузки определяются для каждой организации и отделения, которое вводится в левую панель. С помощью кнопки "New Org" (a) создайте организации "Avis", "Hertz", "CocaCola" и "PepsiCo".

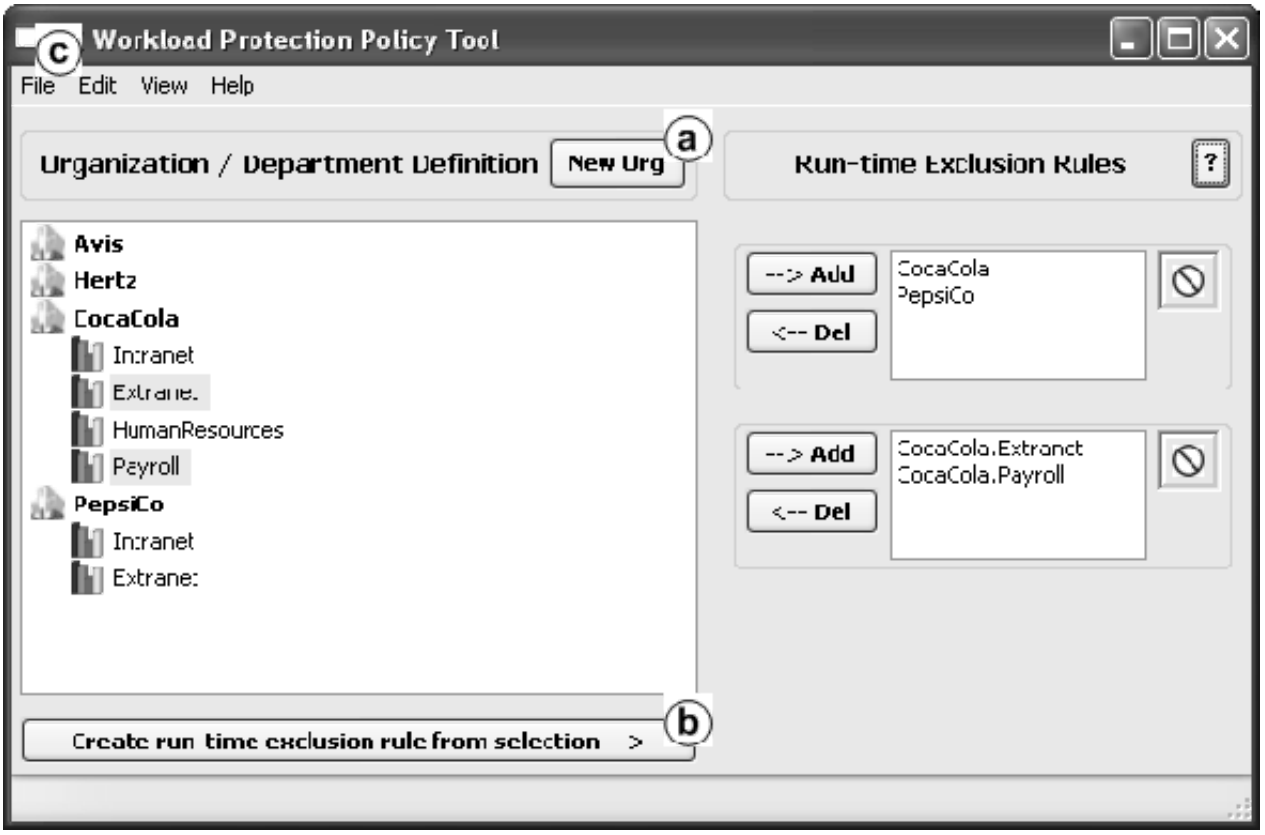
Можно доработать описание организаций -- сделать чтобы нагрузки их различных подразделений различались. Для этого нужно щёлкнуть правой кнопкой мыши на организации и выбрать "Add Department" (или выбрать организацию и нажать Ctrl-A). Создайте подразделения "Intranet", "Extranet", "HumanResources" и "Payroll" для организации "CocaCola" и подразделения "Intranet" и "Extranet" для организации "PepsiCo". Итоговый вид должен быть похожим на левую панель, показанную на рисунке.

Определение конфликтов времени исполнения

Нагрузки, которым нельзя работать одновременно на платформе с одним гипервизором, сгруппированы в "Run-time Exclusion rules" (Правила исключения времени исполнения) на правой панели окна.

Для того чтобы избежать одновременного исполнения нагрузок PepsiCo и CocaCola (включая нагрузки на подразделения) на одном и том же гипервизоре, выберите организацию "PepsiCo", а затем, с нажатой клавишей Ctrl, выберите организацию "CocaCola". Теперь нажмите клавишу (b) с названием "Create run-time exclusion rule from selection" (создать правило исключения времени выполнения основываясь на текущем выделении). Правило появится на правой панели. Имя используется только для удобства и никак не затрагивает политики гипервизора.

Повторите процесс для создания правил исключения времени исполнения для нагрузок отделений CocaCola.Extranet и CocaCola.Payroll.



Итоговый вид окна с описанием нагрузок и правилами исключения времени исполнения

Итоговое окна должно быть похоже на то, что показано на рисунке. Нужно сохранить это определение нагрузки, выбрав "Save Workload Definition as ..." в меню "File" (c). Впоследствии при необходимости это определение можно будет доработать.

Преобразование определения нагрузки в политику sHype/Xen

Для того чтобы превратить определение нагрузки в политики контроля доступа, понятную Xen, выберите пункт "Save as Xen ACM Security Policy" (сохранить как политику безопасности ACM) в меню "File" (c). Введите следующее имя политики в появившемся окне: `example.chwall_ste.test-wld`. Если ezPolicy работает в домене 0, итоговый файл политики `test-wld_security-policy.xml` автоматически попадёт в правильный каталог `/etc/xen/acm-security/policies/example/chwall_ste`. Если программа запущена не в домене 0, нужно вручную скопировать полученный файл в домен 0, а затем продолжить. По поводу правил именования политик безопасносити смотрите здесь.

Внедрение политики защиты нагрузки

Для того чтобы внедрить политику, созданную выше, создаётся файл-представление политикии (`test-wld.bin`), который можно загрузить в гипервизор Xen. Мы именно так и сконфигурируем Xen, чтобы он загружал политику при старте.

Приведённая ниже команда преобразует исходный файл представления политики в файл, который может быть загружен в Xen с поддержкой sHype/ACM. Детали -- в man-странице по xm.

```
(4) # xm makepolicy example.chwall_ste.test-wld
```

Самый простой способ инсталлировать политику для Xen -- это включить её в загрузочную последовательность. Это делает следующая команда:

```
(5) # xm cfgbootpolicy example.chwall_ste.test-wld
```

Как вариант, если команда не сработает (например, из-за того, что она не сможет идентифицировать загрузочную запись Xen), можно вручную, в два шага, установить политику вручную. Во-первых, нужно скопировать бинарный файл политики в каталог `/boot/`:

```
# cp /etc/xen/acm-security/policies/example/chwall_ste/test-wld.bin \
/boot/example.chwall_ste.test-wld.bin
```

Во-вторых, нужно вручную добавить строку загрузки модуля в секцию Xen конфигурационного файла загрузчика GRUB:

```
title Xen (2.6.16.13)
       root (hd0,0)
       kernel /xen.gz dom0_mem=2000000 console=vga
       module /vmlinuz-2.6.16.13-xen ro root=/dev/hda3
       module /initrd-2.6.16.13-xen.img
       module /example.chwall_ste.test-wld.bin
```

Загрузитесь по этой записи, чтобы активировать политику и использовать гипервизор Xen с

повышенной безопасностью.

```
(6) # reboot
```

После перезагрузки, проверьте, включена ли безопасность:

```
# xm list --label
Name        ID Mem(MiB) VCPUs State  Time(s)  Label
Domain-0     0     1949     4 r-----   163.9  SystemManagement
```

Если метка безопасности в конце строки выглядит как "INACTIV", безопасность не включена. В этом случае нужно убедиться, что предыдущие шаги были выполнены верно. Примечание: Домену Domain0 назначена метка по умолчанию (см. подробности в этом разделе). Остальные домены, должны быть помечены для того чтобы работать с гипервизором поддерживающим sHype/ACM (ниже описывается как ставить метки на домены и ресурсы).

Пометка доменов

Конфигурационный файл домена должен выглядеть как показано ниже. Этот конфигурационный файл описывает домен domain1: (Примечание. Сайты www.jailtime.org и www.xen-get.org это хорошее место, где можно поискать примеры образов domU)

```
# cat domain1.xm
kernel = "/boot/vmlinuz-2.6.16.13-xen"
memory = 128
name = "domain1"
vif = [ '' ]
dhcp = "dhcp"
disk = ['file:/home/xen/dom_fc5/fedora.fc5.img,sda1,w', \
        'file:/home/xen/dom_fc5/fedora.swap,sda2,w']
root = "/dev/sda1 ro"
```

Если попробовать запустить домен, возникнет следующее сообщение об ошибке:

```
# xm create domain1.xm
Using config file "domain1.xm".
domain1: DENIED
--> Domain not labeled
Checking resources: (skipped)
Security configuration prevents domain from starting
```

Каждый домен, перед тем как быть запущенным на sHype/Xen, должен быть ассоциирован с меткой безопасности. Без этого sHype/Xen не сможет обеспечить целостность политики. Следующая команда показывает все метки доступные в активной политике:

```
# xm labels type=dom
Avis
CocaCola
CocaCola.Extranet
CocaCola.HumanResources
CocaCola.Intranet
CocaCola.Payroll
Hertz
```

```
PepsiCo
PepsiCo.Extranet
PepsiCo.Intranet
SystemManagement
```

Теперь domain1 получит метку CocaCola, а домен domain2 -- метку PepsiCo.Extranet. Подробности в man xen.

```
(7) # xm addlabel CocaCola dom domain1.xm
    # xm addlabel PepsiCo.Extranet dom domain2.xm
```

Если попробовать запустить домен снова:

```
# xm create domain1.xm
Using config file "domain1.xm".
   file:/home/xen/dom_fc5/fedora.fc5.img: DENIED
   --> res:__NULL_LABEL__ (NULL)
   --> dom:CocaCola (example.chwall_ste.test-wld)
   file:/home/xen/dom_fc5/fedora.swap: DENIED
   --> res:__NULL_LABEL__ (NULL)
   --> dom:CocaCola (example.chwall_ste.test-wld)
Security configuration prevents domain from starting
```

Эта ошибка показывает, что domain1, если его запустить, не сможет получить доступ к своему образу и файлу подкачки, поскольку они не имеют соответствующих меток. Это имеет смысл, поскольку для того чтобы ограничить нагрузку, доступ доменов к ресурсам должен контролироваться. В противном случае, домены которым запрещено взаимодействоваать или работать одновременно, могли бы совместно использовать какие-то данные с помощью хранилищ.

Пометка ресурсов

Список доступных меток ресурсов можно посмотреть с помощью команды `xm labels`.

Назначим метку ресурса `CocaCola` файлы образу домена domain1, представляющему `/dev/sda1`, и файлу, представляющему раздел подкачки домена:

```
(8) # xm addlabel CocaCola res \
          file:/home/xen/dom_fc5/fedora.fc5.img
    Resource file not found, creating new file at:
    /etc/xen/acm-security/policies/resource_labels
    # xm addlabel CocaCola res \
          file:/home/xen/dom_fc5/fedora.swap
```

Запуск домена пройдёт успешно:

```
# xm create domain1.xm
# xm list --label
Name           ID Mem(MiB) VCPUs State  Time(s)  Label
domain1         1      128     1 r-----     2.8  CocaCola
Domain-0        0     1949     4 r-----   387.7  SystemManagement
```

Следующая команда выводит список помеченных ресурсов в системе (например, для того

чтобы проверить правильность расстановки меток):

```
# xm resources
file:/home/xen/dom_fc5/fedora.swap
         policy: example.chwall_ste.test-wld
         label:  CocaCola
file:/home/xen/dom_fc5/fedora.fc5.img
         policy: example.chwall_ste.test-wld
         label:  CocaCola
```

Сейчас, если помеченный ресурс перемещён в другое место, нужно вручную удалять метку и затем повторно её ставить с помощью команд `xm rmlabel` и `xm addlabel` соответственно. Детали см. здесь.

```
  (9) Поставьте метку PepsiCo.Extranet на ресурсы домена domain2.
         Пока что, не выполняйте запуск домена.
```

Проверка защиты нагрузки Xen

Проверить как работает защита нагрузки можно убедившись что:

- домены с конфликтующими нагрузками не могут работать одновременно;
- домен не может получить доступ к ресурсам других нагрузок;
- домены не могут обмениваться сетевыми пакетами, если они не ассоциированы с нагрузками одного типа.

Test 1: Правила исключения времени выполнения

Предположим, что домен domain1 с меткой CocaCola всё ещё работает. Правила исключения времени исполнения нашей политики говорят, что пока домен domain1 работает, домен domain2 не может быть запущен, поскольку метка домена domain1 включает CHWALL тип CocaCola, а метка домена domain2 включает CHWALL тип PepsiCo. Правила исключения времени исполнения говорят, что PepsiCo и CocaCola не могут работать в одно и то же время на платформе одного и того же гипервизора. Как только домен domain1 остановлен или сохранён, можно запускать домен domain2, но теперь нельзя уже восстановить domain1. Программа ezPolicy, когда создаёт Chinese Wall типы для меток нагрузок, гарантирует то, что нагрузки подразделений наследуют типы соответствующих организаций (и вместе с этим исключения организаций).

```
# xm list --label
Name           ID Mem(MiB) VCPUs State  Time(s)  Label
domain1         2      128     1 -b----     6.9  CocaCola
Domain-0        0     1949     4 r-----   273.1  SystemManagement

# xm create domain2.xm
Using config file "domain2.xm".
Error: (1, 'Operation not permitted')

# xm destroy domain1
# xm create domain2.xm
Using config file "domain2.xm".
Started domain domain2

# xm list --label
```

```
Name           ID Mem(MiB) VCPUs State  Time(s)  Label
domain2         4      164     1 r-----     4.3  PepsiCo.Extranet
Domain-0        0     1949     4 r-----   298.4  SystemManagement

# xm create domain1.xm
Using config file "domain1.xm".
Error: (1, 'Operation not permitted')

# xm destroy domain2
# xm list
Name           ID Mem(MiB) VCPUs State  Time(s)
Domain-0        0     1949     4 r-----   391.2
```

Можно убедиться, что домены с меткой Avis могут работать вместе с доменами помеченными как CocaCola, PepsiCo или Hertz.

Test 2: Доступ к ресурсам

В этом тесте мы поставим на swap-файл домена domain1 ресурсную метку Avis. Домен domain1 больше не сможет даже запуститься, поскольку у него теперь нед доступа к этому ресурсу. Это тест проверяет возможности доменов по совместному доступу, которые описываются в компоненте Simple Type Enforcement.

```
# xm rmlabel res file:/home/xen/dom_fc5/fedora.swap
# xm addlabel Avis res file:/home/xen/dom_fc5/fedora.swap
# xm resources
file:/home/xen/dom_fc5/fedora.swap
    policy: example.chwall_ste.test-wld
    label:  Avis
file:/home/xen/dom_fc5/fedora.fc5.img
    policy: example.chwall_ste.test-wld
    label:  CocaCola

# xm create domain1.xm
Using config file "domain1.xm".
   file:/home/xen/dom_fc4/fedora.swap: DENIED
   --> res:Avis (example.chwall_ste.test-wld)
   --> dom:CocaCola (example.chwall_ste.test-wld)
Security configuration prevents domain from starting
```

Test 3: Обмен информацией

В этом тесте выполняется такая проверка: могут ли два домена с метками Hertz и Avis обмениваться информацией по сети. Проверка делается с помощью ping. Он также имеет отношение к политике STE. Примечание: sHype/Xen не контролирует прямое взаимодействие между доменами. Однако, в настоящее время домены, ассоциированные с разными нагрузками, могут обмениваться информацией через сеть домена 0. Ведётся работа над механизмами управления sHype/ACM для локального и удалённого сетевого трафика. Дополнительная информация об этом может появится в списке рассылки `xen-devel`.

Политика контроля доступа Xen

В этом разделе детально описывается политика управления доступом sHype/Xen. Информации, представленной здесь, достаточно чтобы написать собственную политику управления доступом и использовать доступные в Xen инструменты управления политиками. Сам язык

описания политик достаточно выразителен для того чтобы эффективно описать большинство симметричных отношений доступа между доменами и ресурсами.

Политика управления доступом Xen состоит из двух компонентов. Первый компонент, называемый политикой Chiness Wall (CHWALL), контролирует какие домены могут работать одновременно на одной виртуализированной платформе. Второй компонент, называемой политикой Simple Type Enfotcement (STE), управляет организацией совместного доступа между доменами, т.е. обменом информацией между ними или доступом к ресурсам. Компоненты CHWALL и STE могут работать по-одиночке, но в нашем примере оба компонента политик сконфигурированы вместе, и они дополняют друг друга. XML файл политики содержит всю информацию необходимую для приведения политик в действие.

На этом и этом рисунках показаны примеры полнофункционнальной, но очень простой политики безопасности Xen. В этой политики выделены две нагрузки: CocaCola и PepsiCo, а также определены метки, необходимые для того чтобы ассоциировать домены и ресурсы с этими нагрузками. XML-политика состоит из таких частей:

1. Заголовок политики, включая её имя;
2. Блок Simple Type Enforcement (простого применения типов);
3. Блок политики Chinese Wall;
4. Блок определения меток.

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <!-- Auto-generated by esPolicy        -->
03 <SecurityPolicyDefinition
      xmlns="http://www.ibm.com"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
          "http://www.ibm.com ../../security_policy.xsd ">
04     <PolicyHeader>
05         <PolicyName>example.chwall_ste.test</PolicyName>
06         <Date>Wed Jul 12 17:32:59 2006</Date>
07     </PolicyHeader>
08
09     <SimpleTypeEnforcement>
10         <SimpleTypeEnforcementTypes>
11             <Type>SystemManagement</Type>
12             <Type>PepsiCo</Type>
13             <Type>CocaCola</Type>
14         </SimpleTypeEnforcementTypes>
15     </SimpleTypeEnforcement>
16
17     <ChineseWall priority="PrimaryPolicyComponent">
18         <ChineseWallTypes>
19             <Type>SystemManagement</Type>
20             <Type>PepsiCo</Type>
21             <Type>CocaCola</Type>
22         </ChineseWallTypes>
23
24         <ConflictSets>
25             <Conflict name="RER1">
26                 <Type>CocaCola</Type>
27                 <Type>PepsiCo</Type>
28             </Conflict>
29        </ConflictSets>
30     </ChineseWall>
31
```

Пример XML-файла политики безопасности - Часть I: Определение типов и правил

Заголовок и имя политики

Строки 1-2 (см. рисунок) включают обычный XML-заголовок. Описание политики безопасности начинается со строки 3. Оно соответствует схеме политики. XML-схема для описания политик безопасности может быть найдена в файле `/etc/xen/acm-security/policies/security-policy.xsd`. В каталоге `examples/` с примерами есть примеры политик безопасности. Каталог `acm-security/` единственный из проинсталлированных, в том случае, если безопасность ACM сконфигурирована в процессе инсталляции (см. здесь).

Заголовок политики покрывает строки 4-7. Он включает поле даты и определяет имя политики example.chwall_ste.test. Также он может включать необязательные поля, которые не показаны здесь, и которые предназначаются для будущего использования (см. описание схемы).

Имя политики нужно для двух вещей. Во-первых, оно собственно задаёт уникальное имя для политики безопасности. Это имя экспортируется гипервизором в управляющие утилиты Xen для того чтобы быть уверенным, что во всех случаях речь идёт об одном и том же. Планируется расширение имени политики цифровым отпечатком содержимого политики для лучшей защиты связи имени и содержимого. Во-вторых, оно неявно указывает утилите xm местоположение, где в системе хранится XML-политика. Заменив в названии политики двоеточия (точки?) на символы /, можно получить полный путь к файлу политики относительно каталога `/etc/xen/acm-security/policies`. Последняя часть имени политки это префикс для XML-файла политики, за которым следует `-security_policy.xml`. Например, политика с именем `example.chwall_ste.test` будет находиться в файле `test-security_policy.xml`, который находится в каталоге `example/chwall_ste/` в каталоге политик.

Компонент политики Simple Type Enforcement

Политика Simple Type Enforcement (STE) контролирует какие домены могут обмениваться информацией и осуществлять совместный доступ к ресурсам. С помощью этого компонента, Xen может ограничивать типы нагрузок, ограничивая домены, которые выполняют соответствующие нагрузки. Применение политки происходит, когда домены используют различные пути обмена информацией (разделяемая память, события, разделямые ресурсы, такие как блочные устройства). Она строится на базе изоляции гипервизора, что ограничивает взаимодействие по явным каналам. STE не защищает и даже не пытается защищать от обмена информацией по скрытым каналам в гипервизоре или железе; это ортогональная проблема, которую можно смягчить правилами исключения времени выполнения, описанными выше, или исправив соответствующую ошибку в гипервизоре.

Xen управляет совместным доступом доменов на уровне доменов и ресурсов, посколько эта абстракция, является для гипервизора и управляющих утилит естественной. И хотя управляемые элементы получаются весьма крупными, такой подход оказывается достаточно надёжным и требует минимальный изменений гипервизора для внедрения в него возможностей принудительного контроля доступа (mandatory access control). Он включает возможность организации многоуровневой безопасности, не зависящей от платформы и операционной системы.

Строки 9-15 (см. рисунок) определяют компонент политики Simple Type Enforcement. По сути, они определяют типы нагрузок SystemManagement, PepsiCo и CocaCola, доступные в компоненте STE политики. Правила политики неявные: Xen позволяет доменам обмениваться информацией между собой только в том случае, если домены имеют одинаковый тип STE. Xen позволяет домену обращаться к ресурсу только в том случае, если метка домена и ресурса имеют одинаковый тип нагрузки STE.

Компонент политики Chinese Wall

Политики Chinese Wall (Китайская стена) в интерпретации sHype позволяет пользователяем предотвратить одновременное выполнение нагрузок от одновременного исполнения на платформе одного гипервизора. Правила исключения времени выполнения (Runtime Exclusion Rules, RER), также называемые множествами конфликтов (Conflict Sets), определяют множество типов нагрузок, которым нельзя выполняться одновременно. Из всех нагрузок, определённых в правилах исключения времени выполнения, максимум одна может выполняться на платформе одного и того же гипервизора в любой момент времени. RER реализуют менее жёсткий вариант оригинального компонента безопасности Chinese Wall. В них не реализовано *-свойство политики, которое потребовало бы ограничить типы, которые не являются частью правила исключения, как только они задействованы вместе с типами в правиле исключения (дополнительную информацию о том, что такое политика Chinese Wall вообще, см. здесь http://www.gammassl.co.uk/topics/chinesewall.html).

Xen при выполнении правил исключений времени исполнения учитывает ту часть метки, которая относится к ChineseWallTypes. Будет некорректно определять метки, включающие конфликтующие типы Chinese Wall'ов.

Строки 17-30 (см. рисунок) определяют компонент политики Chinese Wall. Строки 17-22 описывают известные типы Chinese Wall, которые здесь совпадают с STE типами, определёнными выше. Так бывает, если критерии для совместного использования доменами и совместного использования аппаратной платформы одинаковы. Строки 24-29 определяют одно правило исключения времени выполнения:

```
<Conflict name="RER1">
  <Type>CocaCola</Type>
  <Type>PepsiCo</Type>
</Conflict>
```

Основываясь на этом правиле, Xen разрешит домену только одного типа, CocaCola или PepsiCo, работать с одним гипервизором в каждый момент времени. Например, как только запущен домен с назначенным ему типом нагрузки CocaCola, домен с типом нагрузки PepsiCo стартовать не сможет. Когда предыдущий домен завершится, а других доменов такого типа запущено не будет, сможет стартовать домен PepsiCo.

Для того чтобы отследить, какие нагрузки используются, Xen отслеживает количество ссылок на каждый тип нагрузки. Каждый раз когда домен стартует или возобновляет работу, количество ссылок на тип Chinese Wall, на который ссылается доменная метка, увеличивается. Каждый раз когда домен уничтожается или сохраняется, количество соответствующих ссылок уменьшается. sHype в Xen поддерживает миграцию и живую миграцию, которая обрабатывается также как сохранение на одной платформе и последующее возобновление на другой платформе работы домена.

Причины, из-за которых может потребоваться ограничить список разрешённого к использованию доменом или нагрузкой оборудования:

- Несовершенство механизмов управления ресурсами может привести к тому, что из-за одного некорректно ведущего себя домена ресурсов не будет другим домена и нагрузкам, работающим в них.
- Запасные домены могут выполнять ту же нагрузку для повышения отказоустойчивости. Такие домены не должны работать на одном и том же оборудовании, чтобы избежать узких мест.
- Из-за несовершенной изоляции доменов два злонамеренных домена могут обмениваться мужде собой данными. Таким образом, они обходят механизмы контроля доступа Xen. Эта проблема не может быть устранена полностью. Она является результатом

компромисса между безопасностью и другими требованиями архитектуры. Простой пример скрытого канала описан здесь: http://www.multicians.org/timing-chn.html. Такие же скрытые каналы могут быть между нагрузками, выполняющимися на различных платформах, при условии, что они соединены с помощью сети. Политика Xen Chinese Wall обеспечивает приблизительную реализацию этого несовершенного "зазора" между выбранными типами нагрузок.

Метки безопасности

Для того чтобы у Xen была возможность ассоциировать домены с типами нагрузок, которые в них работают, каждому домену присваивается метка безопасности, включающая информацию о типах нагрузок домена.

```
32      <SecurityLabelTemplate>
33          <SubjectLabels bootstrap="SystemManagement">
34              <VirtualMachineLabel>
35                  <Name>SystemManagement</Name>
36                  <SimpleTypeEnforcementTypes>
37                      <Type>SystemManagement</Type>
38                      <Type>PepsiCo</Type>
39                      <Type>CocaCola</Type>
40                  </SimpleTypeEnforcementTypes>
41                  <ChineseWallTypes>
42                      <Type>SystemManagement</Type>
43                  </ChineseWallTypes>
44              </VirtualMachineLabel>
45
46              <VirtualMachineLabel>
47                  <Name>PepsiCo</Name>
48                  <SimpleTypeEnforcementTypes>
49                      <Type>PepsiCo</Type>
50                  </SimpleTypeEnforcementTypes>
51                  <ChineseWallTypes>
52                      <Type>PepsiCo</Type>
53                  </ChineseWallTypes>
54              </VirtualMachineLabel>
55
56              <VirtualMachineLabel>
57                  <Name>CocaCola</Name>
58                  <SimpleTypeEnforcementTypes>
59                      <Type>CocaCola</Type>
60                  </SimpleTypeEnforcementTypes>
61                  <ChineseWallTypes>
62                      <Type>CocaCola</Type>
63                  </ChineseWallTypes>
64              </VirtualMachineLabel>
65          </SubjectLabels>
66
67          <ObjectLabels>
68              <ResourceLabel>
69                  <Name>SystemManagement</Name>
70                  <SimpleTypeEnforcementTypes>
71                      <Type>SystemManagement</Type>
72                  </SimpleTypeEnforcementTypes>
73              </ResourceLabel>
74
75              <ResourceLabel>
76                  <Name>PepsiCo</Name>
77                  <SimpleTypeEnforcementTypes>
78                      <Type>PepsiCo</Type>
79                  </SimpleTypeEnforcementTypes>
80              </ResourceLabel>
81
82              <ResourceLabel>
83                  <Name>CocaCola</Name>
84                  <SimpleTypeEnforcementTypes>
85                      <Type>CocaCola</Type>
86                  </SimpleTypeEnforcementTypes>
87              </ResourceLabel>
88          </ObjectLabels>
89      </SecurityLabelTemplate>
90  </SecurityPolicyDefinition>
```

Пример XML-файла политики безопасности - Часть II: Определение меток

Строки 32-89 (см. рисунок) определяют шаблон `SecurityLabelTemplate`, включающий метки, которые могут быть прикреплены к доменам и ресурсам, когда политика активна. Доменные метки включают в свой состав типы Chinese Wall, а метки ресурсов -- нет. Строки 33-65 определяют метки SubjectLabels, которые могут быть назначены доменам. Например, метка виртуальной машины `CocaCola` (строки 56-64 на рисунке) ассоциируют соответствующий домен с типом нагрузки `CocaCola`.

Атрибут `bootstrap` именует метку `SystemManagement`. Xen во время загрузки назначает эту метку домену Domain0. Метки для всех остальных доменов назначаются в соответствии с конфигурационным файлом домена (примеры назначения меток для доменов см. здесь). Строки 67-88 определяются метки объектов ObjectLabels. Когда политика активна, эти метки могут назначаться ресурсам.

Вообще, пользовательским доменам можно назначать метки, у которых есть только один тип нагрузки SimpleTypeEnforcement. В таком случае, нагрузка оказывается ограниченной, даже если пользовательский домен начинает вести себя некорректно. В любом домене, которому назначена метка с несколькими типами STE, можно доверять в том смысле, что информация принадлежащая различным STE типам будет разделена. Например, домен Domain0 имеет метку SystemsManagement, которая включает все известные STE типы. Это означает, что домен должен беспокоиться о том чтобы не допускать неавторизованного обмена информацией (например, через блочные устройства или виртуальную сеть) между доменами или ресурсами, которым назначены разные типы STE.

Для того чтобы ассоциировать метку с доменом, администратор безопасности просто использует имя метки, указанное в поле <Name> (см. здесь). Типы внутри метки испольщуются Xen'ом для выполнения контроля доступа. Имя метки может быть каким-угодно (главное, чтобы оно было уникальным), но рекомендуется выбирать его так, чтобы оно соответствовало включенным в метку типам безопасности. XML-представление показанной выше метки может показаться излишне гибким, как мы увидим в примере ниже, в общем случае метки могут состоять из множества типов.

Предположим, что нагрузки `PepsiCo` и `CocaCola` пользуются виртуальными дисками, которые предоставляет виртуальный домен ввода/вывода, использующий физическое устройство хранения, и у этого домена такая метка:

```
<VirtualMachineLabel>
  <Name>VIO</Name>
  <SimpleTypeEnforcementTypes>
      <Type>CocaCola</Type>
      <Type>PepsiCo</Type>
  </SimpleTypeEnforcementTypes>
  <ChineseWallTypes>
      <Type>VIOServer</Type>
  </ChineseWallTypes>
</VirtualMachineLabel>
```

Этот виртуальный домен ввода/вывода (Virtual I/O domain, VIO) экспортирует свои виртуализированные диски общаясь с обоими доменами: с меткой PepsiCo и с меткой CocaCola. Для этого нужно чтобы у VIO домена были обе метки: PepsiCo и CocaCola. В этом примере разграничения нагрузок PepsiCo и CocaCola зависит от домена, который должен держать данные этих двух разных нагрузок разделёнными. Виртуальные диски тоже помечены (см. этот раздел по поводу пометки ресурсов) и enforcement функции VIO домена должны гарантировать, что метки виртуального диска и домена, монтирующего виртуальный диск, имеют одинаковый тип STE. VIO метка со своим CHWALL типом VIOServer позволяет VIO серверу, к которому есть доверие, выполнять одновременно нагрузки PepsiCo и CocaCola.

Как вариант, система у которой есть два жестких диска может не использовать VIO домен, а может непосредственно назначить каждое устройство отдельным нагрузкам (если у платформы есть IO-MMU). Организация совместного использования железа с помощью виртуализации это компромисс между объёмом безопасного (trusted) кода (т.е. размером trusted computing base) и допустимой величиной дополнительных мер предосторожности. Это относится как периферии, так и к самой системной платформе.

Инструменты для создания политик безопасности sHype/Xen

Для создания политик безопасности для Xen могут использоваться такие инструменты:

- ezPolicy GUI tool - начало написания политик;
- xensec_gen tool - доводка политик, созданных с помощью ezPolicy;
- текстовый или XML-редактор.

В этом примере для того чтобы быстро создать политику защиты нагрузки использозвался ezPolicy. При желании можно загрузить полученный файл в xensec_gen и доработать его. Его же можно непосредственно доработать с помощью редактора XML. Каждый XML-файл при трансляции проверяется на соответствие схеме политики безопасности (см. этот раздел).

Существующие ограничения

Работа над sHype/ACM для Xen продолжается. Сейчас идёт работа по организации защиты виртуализированных ресурсов и планируется организация защиты удалённых ресурсов и доменов. Ниже описываются ограничения существующие ограничения механизма управления доступом, над устранением которых ведётся работа.

Сетевой трафик

Локальный и удалённый сетевой трафик не контролируется. Существуют решения по наложению ограничений sHype/ACM на виртуальную сеть, но их нужно обсудить прежде чем сделать частью Xen. Ведётся разработка возможности подчинения внешнего сетевого трафика политикам ACM. Сейчас требуется ручная установка фильтров в домене 0, но этот подход не очень хорошо масштабируется.

Контроль доступа и использования ресурсов

Ведётся работа над возможностью распространению политики безопасности на несколько гипервизоров и ресурсы, к которым выполняется доступ по сети. Ведётся работа по расширению управления доступом к новым видам ресурсов (например, сетевым хранилищам).

На отдельной Xen-системе информация о связи ресурсов и меток безопасности хранится в файле `/etc/xen/acm-security/policy/resource_labels`. В этом файле полным путям к ресурсам поставлены в соответствие метки. Связь является слабой, и она разрывается в том случае, если ресурсы перемещаются или переименовываются без соответствующих изменений в файле. Работы над механизмами управления и ограничения на использование ресурсов в Xen продолжаются.

Миграция доменов

Метки доменов сохраняются при миграции. Гипервизор-получатель прежде чем разрешить ми миграцию должен убедиться, что метка правильная, и домен имеет право работать (с учётом правил политики Chinese Wall). Тем не менее, сеть между гипервизорами должна быть

безопасной. Существуют архитектуры и прототипы, обеспечивающие защиту сети и гарантирующие целостность политик контроля доступа, однако соответствующие патчи пока не включены в основной дистрибутив.

Скрытые каналы

Система управления доступом sHype реализует политики безопасности, не зависящие от системы. Она базируется на изоляции ядра гипервизора. Любые скрытые каналы, которые существуют в ядре гипервизора или в железе (например, разделяемый кэш процессора), автоматически унаследуются. Если эти скрытые каналы не результат компромиссов между безопасностью системы и другими её свойствами, тогда проще всего минимизировать или устранить их там, где они возникли. Сам sHype тоже предлагает некоторые способы по смягчению удара (например, правила исключения времени исполнения).

Справочная информация

Опции сборки и загрузки

В этой главе описываются опции сборки и загрузки, с помощью которых можно подстроить Xen требуемым образом.

Конфигурационные опции верхнего уровня

Конфигурирование выполняется путём редактирования файлов: `Config.mk` и `Makefile`.

В файле `Config.mk` указывается архитектура, для которой будет выполняться сборка. Её не нужно изменять, за исключением случаев, когда производится кросс-компиляция или сборка выполняется для системы с поддержкой PAE. Дополнительные конфигурационные опции описаны в `Config.mk`.

Файл `Makefile` используется главным образом для настройки набора собираемых ядер. Это делается с помощью строки:

```
KERNELS ?= linux-2.6-xen0 linux-2.6-xenU
```

Допускается указывать здесь любые ядра, для которых есть соответствующий конфигурационный файл в каталоге `buildconfigs/`.

Опции сборки Xen

`verbose=y`
Включить отладочные сообщения, когда Xen обнаруживает неожиданное состояние. Также включает консольный вывод из всех доменов.

`debug=y`
Включить отладочные сообщения. Подразумевает `verbose=y`. (Главным образом используется для отлавливания ошибок в Xen).

`debugger=y`
Включить отладчик Xen. Может использоваться для отладки Xen, гостевых ОС и

приложений.

`perfc=y`
Включает счётчики производительности внутри Xen для важных событий. Счётчики можно сбросить или просмотреть с консоли Xen с помощью специальных управляющих комбинаций клавиш.

Опции загрузки Xen

Данные опции используются для конфигурирования Xen во время работы. Их нужно добавлять в командную строку Xen или вручную, при запуске, или путём редактирования конфигурационного файла GRUB `grub.conf`.

`noreboot`
Не перезагружать машину автоматически при возникновении ошибки. Это может помочь прочитать отладочный вывод, в тех случаях, когда не используется serial-консоль.

`nosmp`
Выключить поддержку SMP. Эта опция подразумевается опцией `` `ignorebiostables' ``.

`watchdog`
Включить watchdog NMI. Он может сообщать об определённых сбоях.

`noirqbalance`
Отключить software IRQ balancing and affinity. Это может понадобиться в системах таких как Dell 1850/2850 в которых используются хитрые железные решения для обхода проблем с маршрутизацией IRQ.

~badpage=<page number>,<page number>,...~
Список страниц, которые нельзя использовать из-за того что они содержат плохие байты. Например, если в результате тестирования памяти установлено что байт с адресом 0x12345678 сбойный, в командной строке Xen нужно указать "badpage=0x12345".

~com1=<baud>,DPS,<io_base>,<irq> com2=<baud>,DPS,<io_base>,<irq>~
Xen поддерживает до двух 16550-совместимых последовательных портов. Например: `com1=9600, 8n1, 0x408, 5' обозначает COM1, на скорости 9600, 8 битов данных, без четности, адрес базы ввода/вывода 0x408, IRQ 5. Если какие-то конфигурационные опции имеют значение по умолчанию (например, адрес базы ввода/вывода или IRQ), тогда нужно указывать только часть конфигурационной строки. Если скорость порта сконфигурирована (например, с помощью загрузчика), тогда вместо численного значения можно указать `auto`.

~console=<specifier list>~
Указать получателя консольного ввода/вывода. Список состоит из значений, разделенных запятыми:

; `vga`
: Использовать консоль VGA (до тех пор пока не загрузится домен 0, за исключением тех случаев, когда указан vga=keep).
; `com1`
: Использовать последовательный порт com1.
; `com2H`
: Использовать последовательный порт com2. У передаваемых символов будет установлен MSB. У принимаемых символов должен быть установлен MSB.
; `com2L`

: Использовать последовательный порт com2. У передаваемых символов будет сброшен MSB. У принимаемых символов должен быть сброшен MSB.

Последние два позволяют организовать совместное использование порта двумя подсистемами (например, консолью и отладчиком). Совместное использование контролируется MSB каждого передаваемого/принимаемого символа. Значение по умолчанию 'com1,vga'.

`vga=<options>`

Список опций, разделённых с помощью `;`.

; `text-<mode>`

: Выбрать разрешение текстового режима, где режим может иметь такие значения: 80x25, 80x28, 80x30, 80x34, 80x43, 80x50, 80x60.

; `keep`

: Сохранять консоль в VGA режиме даже после того как загрузится домен 0.

`sync_console`

Выполнять вывод на консоль в синхронном режиме. Это может помочь в том случае, если система падает перед тем как вывести на консоль все имеющиеся для вывода данные. В большинстве случаев Xen автоматически переходит в синхронный режим сам, как только возникает исключительно событие, но эта опция переводит в синхронный режим в любом случае.

`conswitch=<switch-char><auto-switch-char>`

Указывает, как переключать последовательную консоль между Xen и доменом 0. Требуемая последовательность -- Ctrl-<switch-char> нажатый три раза подряд. Для того чтобы отключить переключение, нужно в качестве символа переключения указать обратную кавычку. Символ <auto-switch-char> указывает, должен ли Xen автоматически переключать ввод на домен 0, когда загрузка Xen завершится. Символ "x" означает, что автоматическое переключение отключено. Автоматическое переключение включено во всех остальных случаях. Символ переключения по умолчанию это "a".

`nmi=xxx`

Указывает, что делать с чётностью NMI и ошибками ввода/вывода.

*"nmi=fatal": Xen выводит диагностическое сообщение после чего зависает.

*"nmi=dom0": Уведомить домен 0 NMI.

*"nmi=ignore": Игнорировать NMI.

`mem=xxx`

Установить предел адресации физической памяти. Память, находящаяся за пределами указанного значения, будет игнорироваться. Параметр может быть указан с суффиксом B, K, M или G, обозначающим байты, килобайты, мегабайты и гигабайты соответственно.

`dom0_mem=xxx`

Указывает объём памяти, выделяемый домену 0. В Xen 3.0 параметр может быть указан с суффиксом B, K, M или G, обозначающим байты, килобайты, мегабайты и гигабайты соответственно. Если суффикс не указан, подразумеваются килобайты. В предыдущих версих Xen суффикс не поддерживался, и все значения интерпретировались как значения в килобайтах.

`dom0_vcpus_pin`

Закрепить виртуальные процессоры домена 0 за соответствующими им физическими процессорами (по умолчанию =false).

`tbuf_size=xxx`

Размер буферов трассировки для каждого процессора, в страницах. По умолчанию 0.

`sched=xxx`
Выбрать какой планировшик процессора будет использовать Xen. Возможные значения "credit" (по умолчанию) и "sedf".

`apic_verbosity=debug,verbose`
Выводить более развёрнутую информацию о конфигурации локального APIC и IOAPIC.

`lapic`
Использовать локальный APIC, даже если он отключен однопроцессорным BIOS'ом.

`nolapic`
Не использовать локальный APIC в однопроцессорной системе, даже если он включён в BIOS.

`apic=bigsmp,default,es7000,summit`
Указать платформу NUMA. Обычно она определяется автоматически.

В дополнение к перечисленным опциям, в командной строке Xen можно указывать следующие опции. Поскольку домен 0 тоже отвечает за загрузку системы, Xen автоматически передаёт эти опции в его командную строку. Опции взяты из синтаксиса командной строки ядра Linux без изменения смысла.

`acpi=off,force,strict,ht,noirq,...`
Указать как Xen (и домен 0) обрабатывают таблицы ACPI BIOS.

`acpi_skip_timer_override`
Указать Xen'у (и домену 0) игнорировать перекрывающие прерывание теаймера инструкции определённые таблицами ACPI BIOS.

`noapic`
Указать Xen (и домену 0) игнорировать IOAPIC'и присутствующие в системе и вместо них использовать обычный PIC.

Опции загрузки XenLinux

В дополнение к стандартным опциям Linux поддерживается:

`xencons=xxx`
Устройство, к которому подключается драйвер виртуальной консоли Xen. Поддерживаются следующие варианты:
* `xencons=off`: отключить виртуальную консоль
* `xencons=tty`: подключить консоль к `/dev/tty1` (`tty0` при загрузке)
* `xencons=ttyS`: подключить консоль к `/dev/ttyS0`

Дальнейшая поддержка

Если у вас есть вопросы, на которые нет ответов в этом руководстве, возможно, вам смогут помочь источники указанные ниже. Отчёты об ошибках, предложения и дополнения к программному обеспечению (а также документации) Xen нужно высылать в список разработчиков Xen (адрес ниже).

Другая документация

Разработчикам, интересующими портированием другим операционных систем на Xen, может оказаться полезным руководство *Xen Interface Manual*, находящееся в каталоге `docs/` дистрибутива Xen.

Ссылка на Интернет-ресурсы

Официальный web-сайт Xen находится здесь:

http://www.xensource.com

На нём есть ссылки на последние версии online-документации, включая последнюю версию этого руководства.

Русскоязычная версия находится по адресу (-- *Прим. перев.*):

http://xgu.ru/xen/manual/

Информация о Xen есть также на Xen Wiki по адресу:

http://wiki.xensource.com/xenwiki/

Проект Xen использует Bugzilla для отслуживания обнаруженных ошибок. Bugzilla Xen находится по адресу:

http://bugzilla.xensource.com/bugzilla/

Информация о Xen на русском языке есть на русских wiki-страницах по Xen по адресу (-- *Прим. перев.*):

http://xgu.ru/wiki/Xen

Списки рассылки

Существуют несколько списков рассылки, посвящёных Xen. Наиболее важные из них перечислены ниже. Официальная страница списков рассылки и информация о подписке на них находится здесь:

```
http://lists.xensource.com/
```

xen-devel@lists.xensource.com
Обсуждение разработки и ошибок. Подписка здесь: `http://lists.xensource.com/xen-devel`

xen-users@lists.xensource.com
Обсуждение инсталляции и использования. Подписка здесь: `http://lists.xensource.com/xen-users`

xen-announce@lists.xensource.com
Используется только для анонсов. Подписка здесь: `http://lists.xensource.com/xen-announce`

xen-changelog@lists.xensource.com

Лента Changelog для веток unstable и 2.0 - ориентирован на разработчиков. Подписка здесь: `http://lists.xensource.com/xen-changelog`

Немодифицированные (VMX) гостевые домены с технологией Intel®Virtualization Technology (VT)

Xen поддерживает выполнение немодфицированных гостевых операционных систем посредством технологии виртуализации Virtualization Technology (VT), которая есть в новых процессорах Intel. Дополнительную информацию о технологии виртуализации VT, которая реализуут архитектурные расширениях для виртуальных машин (Virtual Machine Extensions, VMX), можно найти на странице http://www.intel.com/technology/computing/vptech

Сборка Xen с поддеркой технологии аппаратной виртуализации (VT)

Для сборки Xen с поддержкой VMX нужны перечисленные ниже пакеты. В некоторых дистрибутивах Linux по умолчанию эти пакеты отсутствуют.

dev86. Пакет dev86 содержит ассемблер и компоновщик для кода реального режима 80x86. Этот пакет обязателен для того чтобы собрать код BIOS, который будет выполняться в реального режиме виртуальной машины. Если такого пакета для архитектуры x86_64 нет, можно установить версию этого пакета для i386. RPM-пакеты с этим программным обеспечением для разных дистрибутивов можно найти на странице http://www.rpmfind.net/linux/rpm2html/search.php?query=dev86&submit=Search

LibVNCServer. VGA-экран, клавиатуру и мышь немодифицированной гостевой системы можно виртуализировать с помощью библиотеки libvncserver. Получить исходный код библиотеки можно со страницы http://sourceforge.net/projects/libvncserver. В версии 0.8 наблюдается сильное снижение производительности. В текущей версии в CVS ситуация исправлена. Поэтому рекомендуется использовать версию больше 0.8 или вообще получать библиотеку из CVS.

SDL-devel и **SDL**. Simple DirectMedia Layer (SDL) -- другой способ виртуализировать консоль немодифцированной гостевой системы. С помощью этой библиотеки становится возможным использовать консоль немодифицированной виртуальной машины в системе X Window. Если пакеты с SDL и SDL-devel по умолчанию не установлены, их можно найти на страницах http://www.rpmfind.net/linux/rpm2html/search.php?query=SDL&submit=Search и http://www.rpmfind.net/linux/rpm2html/search.php?query=SDL-devel&submit=Search.

Конфигурационный файл немодифицированных гостевых VMX систем

В инсталляции Xen есть пример конфигурационного файла, `/etc/xen/xmexample.vmx`. В нём есть комментарии, описывающие все опции. В дополнение к обычным опциям, подходящим для паравиртуализированных систем, есть несколько, предназначенных только для VMX.

kernel
Загрузчик VMX, `/usr/lib/xen/boot/vmxloader`

builder
build-функция домена. Домен VMX использует `vmx` builder.

acpi
Включить ACPI в гостевой системе VMX, по умолчанию =0 (выключено)

apic
Включить APIC в гостевой системе VMX, по умолчанию =0 (выключено)

pae
Включить PAE в гостевой системе VMX, по умолчанию =0 (выключено)

vif
Опционально указывает MAC-адрес и/или бридж для сетевого интерфейса. Если MAC-адрес не указан, он выбирается случайно. Если указать type=ioemu, для виртуализации сетевой карты машины используется эмулятор ioemu. Если тип не указан, используется виртуализация с помощью vbd, как и в паравиртуализированных гостевых системах.

disk
Определяет дисковые устройства, к которым домен будет иметь доступ, а также какой именно у него будет доступ. В том случае, если в качестве дисков VMX используются физические диски, запись для каждого диска будет иметь следующую форму:

```
phy:UNAME,ioemu:DEV,MODE  ,
```

где UNAME - это имя устройства в домене 0; DEV - это имя устройства, которое будет видется в пользовательском домен; MODE - это r, обозначающий режим только-для-чтения, или w, обозначающий режим чтение-запись; ioemu означает, что для виртуализации VMX-дисков будет использоваться ioemu. Если не добавлять ioemu, используются vbd как в паравиртуализированных гостевых системах.

Если используется файл с образом диска, применяется такая форма:

```
file:FILEPATH,ioemu:DEV,MODE
```

Если используется больше чем один диск, записи, соотвествующие дискам, должны разделяться запятыми. Например:

```
disk = ['file:/var/images/image1.img,ioemu:hda,w', 'file:/var/images/image2.img,ioemu:hdb,w']
```

cdrom
Образ диска для CD-ROM'a. По умолчанию это `/dev/cdrom` в домене 0. Внутри домена VMX, CD-ROM будет виден как устройство `/dev/hdc`. Эта запись также может указывать на ISO-файл. boot Загружаться с флоппи-дисковода(a), жёсткого диска (c) или CD-ROM'a (d). Например, для того чтобы загружаться с CD-ROM'a, запись должна выглядеть так:

```
boot='d'
```

device_model
Инструмент для эмуляции оборудования для гостевых систем. Этот параметр менять не нужно.

sdl

Включить библиотеку SDL для графики, по умолчанию = 0 (выключено)

vnc
Включить библиотеку VNC для графики, по умолчанию = 1 (включено)

vncviewer
Включить запуск vncviewer (допустимо только при vnc=1), по умолчанию = 1 (enabled)

Если vnc=1 и vncviewer=0, можно подключаться к экрану VMX домена с помощью удалённого vncviewer'a. Например:

```
$ vncviewer domain0_IP_address:VMX_domain_id
```

ne2000
Включить ne2000, по умолчанию = 0 (выключено; использовать pcnet)

serial
Включить перенаправление вывода на последовательный порт гостевой VMX-системой на pty-устройство

usb
Включить поддержку USB без включения какого-то конкретного USB-устройства. По умолчанию эта опция равна 0 (выключена), за исключением случая, когда включена опция `usbdevice` - в этом случае опция равна 1 (включена).

usbdevice
Включить поддержку USB и включить поддержку заданного устройства. Устройства, которые можно здесь указывать это `mouse` (мышь PS/2), `tablet` (планшет - устройство абсолютного позиционирования) и `host:id1:id2` (физическое USB-устройство на хост-машине с идентификаторами id1 и id2). Преимущество планшета в том, что Windows автоматически его распознаёт и включает его поддержку, поэтому достаточно только указать строку

```
usbdevice='tablet'
```

и в виртуальной системе появится мышь, которая прозрачно работает под VNC. В Linux пока что отстуствует поддержка USB-планшета, поэтому в Linux нужно использовать эмуляцию Summagraphics. Детальная информация о эмуляции мыши приводится в этом разделе.
localtime
Установить часы в локальное время (по умолчанию =0, что означает - установить в UTC).

enable-audio
Включить поддержку аудио. Находится в процессе разработки.

full-screen
Запускаться в полноэкранном режиме. Находится в процессе разработки.

nographic
Другой способ перенаправления последовательного вывода. Если включён, не будет работать ни 'sdl', ни 'vnc'. Не рекомендуется.

Создание виртуальных дисков с нуля

На основе физических дисков

В том случае, если используется физический диск или дисковый раздел, сначала нужно установить ОС Linux. После этого нужно в правильное место установить загрузчик. Например, в `/dev/sda` в случае загрузки со всего диска, или в `/dev/sda1` при загрузке с раздела 1.

На основе файлов-образов

Сначала нужно создать большой пустой файл-образ. После этого, нужно установить в него ОС Linux. Есть два метода сделать это:

1. Непосредственная инсталляция и виртуальной машины VMX. Выполняется загрузка и инсталляция с CD-ROM'а.
2. Копирование в файл установленной операционной системы. Кроме этого придётся проинсталлировать загрузчик.

Для создания файла-образа: Размер образа должен быть достаточно большим, для того чтобы в него влезла целая ОС. В данном примере предполагается, что размер образа - 1G (что, по-видимому, будет маловато для большинства современных ОС).

```
# dd if=/dev/zero of=hd.img bs=1M count=1 seek=1023
```

Для того чтобы непосредственно установить ОС Linux в файл-образ с помощью гостевой VMX-системы:

Установите Xen и создайте гостевую VMX-систему с файлом-образом и загрузкой с CD-ROM. Дальше всё будет как в обычной инсталляции Linux. В конфигурационном файле должны быть такие записи:

```
cdrom='/dev/cdrom'
boot='d'
```

Если с помощью этого метода успеха добиться не получится, можно воспользоваться другим -- скопировать проинсталированную систему Linux в файл-образ.

Для того чтобы скопировать установленную ОС в файл-образ: Непосредственная инсталляция -- простейший способ создания разделов и инсталляции ОС в файл-образ. Если вы хотите проинсталлировать какую-то особенную ОС в образ на жёстком диске, скорее всего вы будете использовать именно этот метод.

1. **Установите обычную ОС Linux на хост-машину.** Инсталляцию можно производить любым способом, например, с помощью yum (для Red Hat Linux) или YaST (для Novell SuSE Linux). В дальнейшем предполагается, что операционная система установлена в каталог `/var/guestos/`.

2. **Создайте таблицу разделов.** Образ в файле будет восприниматься как жёсткий диск, поэтому нужно сделать таблицу разделов в файле образа. Например:

```
# losetup /dev/loop0 hd.img
# fdisk -b 512 -C 4096 -H 16 -S 32 /dev/loop0
press 'n' to add new partition
press 'p' to choose primary partition
```

```
press '1' to set partition number
press "Enter" keys to choose default value of "First Cylinder" parameter.
press "Enter" keys to choose default value of "Last Cylinder" parameter.
press 'w' to write partition table and exit
# losetup -d /dev/loop0
```

3. **Создайте файловую систему и установите в неё загрузчик GRUB.**

```
# ln -s /dev/loop0 /dev/loop
# losetup /dev/loop0 hd.img
# losetup -o 16384 /dev/loop1 hd.img
# mkfs.ext3 /dev/loop1
# mount /dev/loop1 /mnt
# mkdir -p /mnt/boot/grub
# cp /boot/grub/stage* /boot/grub/e2fs_stage1_5 /mnt/boot/grub
# umount /mnt
# grub
grub> device (hd0) /dev/loop
grub> root (hd0,0)
grub> setup (hd0)
grub> quit
# rm /dev/loop
# losetup -d /dev/loop0
# losetup -d /dev/loop1
```

Опция -o 16384 программы losetup показывает, что нужно пропустить таблицу разделов в образе. Число равно количеству секторов для пропуска умноженное на 512. Пришлось создать файл `/dev/loop`, поскольку GRUB'у нужно именно имя диска, которое обозначает весь диск, а имя с номером обозначает раздел; имя1 - первый раздел.

4. **Скопируйте файлы операционной системы на образ.** Если Xen уже установлен, при копировании файлов на разделы можно использовать программу lomount вместо losetup и mount. Программе lomount просто нужна информация о разделах.

```
# lomount -t ext3 -diskimage hd.img -partition 1 /mnt/guest
# cp -ax /var/guestos/{root,dev,var,etc,usr,bin,sbin,lib} /mnt/guest
# mkdir /mnt/guest/{proc,sys,home,tmp}
```

5. **Отредактируйте файл `/etc/fstab` в образе гостевой системы.** Файл fstab должен выглядеть так:

```
# vim /mnt/guest/etc/fstab
/dev/hda1 / ext3 defaults 1 1
none /dev/pts devpts gid=5,mode=620 0 0
none /dev/shm tmpfs defaults 0 0
none /proc proc defaults 0 0
none /sys sysfs efaults 0 0
```

6. **Размонтируйте файл-образ**:

```
# umount /mnt/guest
```

Образ гостевой системы `hd.img` готов. Другие образы можно найти на http://free.oszoo.org. При

их использовании обязательно убедитесь в том, что в них установлен загрузчик.

Инсталляция Windows в файл образа в VMX

Для того чтобы установить Windows нужно указать в конфигурационном файле acpi=0.

Гостевые VMX-системы

Редактирование конфигурационного файла Xen VMX

Создайте копию примера конфигурационного файла VMX `/etc/xen/xmexample.vmx` и отредектируйте строку:

```
disk = [ 'file:/var/images/guest.img,ioemu:hda,w' ]
```

в которой замените `guest.img` на имя файла с образом гостевой ОС, который вы создали.

Создание гостевых систем VMX

Используйте обычный метод для создания гостевых систем. С помощью ключа `-f` можно указать файл, содержащий конфигурацию виртуальной машины:

```
# xend start
# xm create /etc/xen/vmxguest.vmx
```

В конфигурации по умолчанию, если VNC включён, SDL выключен. В этом случае для доступа к гостевой системе нужно создавать окна VNC. Если вы хотите использовать SDL, включите его в конфигурационном файле с помощью параметра `sdl=1`. В этом случае vnc можно выключить, установив параметр `vnc=0`.

Проблемы с мышью, особенно под VNC

при доступе к виртуальным машинам через VNC существует небольшая проблема с мышью. Проблема связана с тем, что VNC viewer предоставляет виртуальный указатель, для которого известны только его абсолютные координаты. VMX device model преобразует эти абсолютные координаты в приращения движения, которые гостевая система ждёт от драйвера мыши PS/2, работающего в гостевой системе. К сожалению, невозможно сделать так чтобы вычисленные приращения для гостевого курсора точно соответствовали движению указателя VNC. Это может привести к ситуации, когда курсор гостевой системы находится в центре экрана, а курсор VNC уже на краю, и нет возможности подвинуть гостевой курсор левее или правее (в зависимости от того, где находится VNC-курсор).

Для того чтобы побороть указанные проблемы, существуют различные выходы; в зависимости от того, какой вариант эмуляции мыши используется:

Мышь PS/2 подключённая через порт PS/2.

Эта модель мыши используется по умолчанию. Она прекрасно работает под SDL, однако при доступе через VNC указатель мыши в гостевой системе рассинхронизируется с указателем VNC. Когда это происходит, можно синхронизировать указатели одновременным нажатием левого Ctrl и левого Alt. Когда нажаты эти кнопки, движение указателя VNC не передаётся в

гостевую систему, и можно переместить указатель к тому месту, где он совместится с курсором гостевой системы.

Мышь Summagraphics подключённая через последовательный порт.

В device model есть поддержка для планшета Summagraphics, устройства абсолютного позиционирования. Эмуляция выполнятся на втором последовательном порту, `/dev/ttyS1` в Linux или COM2 в Windows. К сожалению, ни в Linux, ни в Windows по умолчанию нет поддержки планшета Summagraphics, поэтому гостевую систему вручную нужно сконфигурировать для поддержки этого устройства.

Конфигурация Linux.

Во-первых, настройте службу GPM на использование Summagraphics. В зависимости от дистрибутива, возможно, придётся делать другие действия, но обычно нужно отредактировать `/etc/sysconfig/mouse`, так чтобы в нём были строки:

```
MOUSETYPE="summa"
XMOUSETYPE="SUMMA"
DEVICE=/dev/ttyS1
```

Затем нужно перезапустить демон GPM.

После этого нужно изменить конфигурационный файл `/etc/X11/xorg.conf`, так чтобы в X появилась поддержка Summagraphics. Нужно добавить строки:

```
Section "InputDevice"
        Identifier "Mouse0"
        Driver "summa"
        Option "Device" "/dev/ttyS1"
        Option "InputFashion" "Tablet"
        Option "Mode" "Absolute"
        Option "Name" "EasyPen"
        Option "Compatible" "True"
        Option "Protocol" "Auto"
        Option "SendCoreEvents" "on"
        Option "Vendor" "GENIUS"
EndSection
```

После перезапуска X-системы, курсор X должен работать синхронно с указателем VNC.

Конфигурация Windows.

Скачайте файл http://www.cad-plan.de/files/download/tw2k.exe, а потом запустите его в гостевой системе. На вопросы нужно отвечать так:

1. Когда программа спросит model (модель), прокрутите ответы и выберите SummaSketch (MM Compatible).
2. Когда программа спросит COM Port, укажите `com2`.
3. Когда программа спросит Cursor Type (тип курсора), укажите 4 button cursor (4 кнопочный курсор).
4. Затем гостевая система перезагрузится и, после того когда она снова загрузится, курсор в гостевой системе будет совпадаеть с указателем VNC.

Мышь PS/2 подключённая через USB.

Это точно такая же эмуляция PS/2, за тем исключением, что она выполняется через порт USB.

Эмуляция включается конфигурационным флагом:

```
usbdevice='mouse'
```

Планшет USB подключённый через USB.

USB-планшет это устройство абсолютного позиционирования. Оно обладает тем преимуществом, что его поддерживают гостевые Windows-системы, правда, Linux-системы все ещё требуют ручной настройки.

Эмуляция включается конфигурационным флагом:

```
usbdevice='tablet'
```

Конфигурация Linux.

К сожалению, в настоящий момент поддержка USB-планшетов в GPM отсутствует. Если вы хотите использовать GPM под VNC, нужно настраивать эмуляцию Summagraphics в гостевой системе.

Как настроить поддержку в X11 описывается на странице http://stz-softwaretechnik.com/~ke/touchscreen/evtouch.html только с одним небольшим изменением. В файле `xorg.conf` на той странице указаны неправильные значения минимумов и максимумов X и Y. Лучше использовать данную секцию:

```
Section "InputDevice"
    Identifier      "Tablet"
    Driver          "evtouch"
    Option          "Device" "/dev/input/event2"
    Option          "DeviceName" "touchscreen"
    Option          "MinX" "0"
    Option          "MinY" "0"
    Option          "MaxX" "32256"
    Option          "MaxY" "32256"
    Option          "ReportingMode" "Raw"
    Option          "Emulate3Buttons"
    Option          "Emulate3Timeout" "50"
    Option          "SendCoreEvents" "On"
EndSection
```

Конфигурация Windows.

Достаточно включить USB-планшет в гостевом конфигурационном файле. Windows автоматически распознает устройство и сконфигурирует драйверы для него.

Поддержка USB

Есть поддержка для эмулируемой USB-мыши, эмулируемого USB-планшета и физических низкоскоростных USB-устройств (поддержка высокоскоростных устройств USB 2.0 находится в процессе разработки).

Мышь USB PS/2

Подробности эмуляции USB-мыши указаны здесь и здесь. Для того чтобы включить поддержку мыши USB PS/2, нужно добавить строку в конфигурационный файл:

```
usbdevice='mouse'
```

Планшет USB

Подробности эмуляции USB-планшета указаны здесь и здесь. Для того чтобы включить поддержку USB-планшета, нужно добавить строку в конфигурационный файл:

```
usbdevice='tablet'
```

Физическое USB-устройство

Доступ к физическому (низкоскоростному) USB-устройству включается путём добавления строки

```
usbdevice='host:vid:pid'
```

в конфигурационный файл. Параметры vid и pid - это vendor id (идентификатор производителя) и product id (идентификатор устройства), уникальным образом идентифицирующие устройство. Эти идентификаторы можно определить двумя способами:

(Существует альтернативный способ указать USB-устройство:

```
host:bus.addr
```

При таком способе возникает проблема, которая делает его бесполезным. Проблема заключается в том, что адрес меняется каждый раз, когда USB-устройство подключается к системе. В связи с этим, этот способ адресации не рекомендуется и в дальнейшем не описывается.)

1. **С помощью управляющего окна.** Как сказано ниже, перейти у правляющему окну можно с помощью комбинации клавиш Ctrl-Alt-2 в VGA-окне гостевой системы. Если в конфигурационном файле с помощью строки включена поддержка USB

```
usb=1
```

тогда вызов команды

```
info usbhost
```

в управляющей консоли покажет список всех USB-устройств и их идентификаторов. Например, этот вывод:

```
    Device 1.3, speed 1.5 Mb/s
      Class 00: USB device 04b3:310b
```

соответствует USB-мыши с идентификатором производителя (vendor id) 04b3 и идентификатором устройства (product id) 310b. Это устройство можно сделать доступным гостевой VMX-системе, если включить в конфигурационный файл строку

```
usbdevice='host:04be:310b'
```

Также можно включить доступ к USB-устройству динамически, с помощью управляющего окна. Управляющая команда

```
usb_add host:vid:pid
```

даёт доступ к USB-устройству с идентификатором вендора (vendor id) = vid и идентификатором продукта (product id) = pid.

2. С помощью файловой системы /proc. При определении идентификаторов vendor id и product id можно воспользоваться содержимым псевдофайла `/proc/bus/usb/devices`. В этом файле в строке, начинающейся с `P:`, есть поля `Vendor=` и `ProdID=` -- это vendor id и product ID соответственно. Например, для мыши файл будет выглядеть так:

```
T:  Bus=01 Lev=01 Prnt=01 Port=01 Cnt=02 Dev#=  3 Spd=1.5 MxCh= 0
D:  Ver= 2.00 Cls=00(>ifc ) Sub=00 Prot=00 MxPS= 8 #Cfgs=  1
P:  Vendor=04b3 ProdID=310b Rev= 1.60
C:* #Ifs= 1 Cfg#= 1 Atr=a0 MxPwr=100mA
I:  If#= 0 Alt= 0 #EPs= 1 Cls=03(HID  ) Sub=01 Prot=02 Driver=(none)
E:  Ad=81(I) Atr=03(Int.) MxPS=   4 Ivl=10ms
```

Обратите внимание, что строка P: идентифицирует vendor id и product id для мыши как 04b3:310b.

Существует ещё одна проблема, о которой нужно знать, при доступе к физическому USB-устройству из гостевой системы. Ядро домена Dom0 не должно загружать драйвер для устройств с которыми будет работать гостевой домен. Это означает, что драйвер не должен быть вкомпилирован в ядро, а в том случае, если используются модули ядра, модуль драйвера не должен загружаться. Речь идёт только о драйверах устройств. Драйверы контроллеров UHCI и OHCI должны загружаться.

Возвращаясь к примеру с USB-мышью, в том случае, если lsmod выводит текст:

```
Module                  Size  Used by
usbmouse                4128  0
usbhid                 28996  0
uhci_hcd               35409  0
```

USB мышь используется ядром домена 0, и не доступна гостевым системам. Удалить драйвер мыши из ядра домена 0 и сделать мышь доступной гостевым доменам можно командой

```
rmmod usbhid
```

. которая удалит драйвер мыши из домена 0 -- мышь станет доступной в гостевом домене.

Имейте в виду, что система hotplug в Linux перезагружает драйверы, если USB-устройство удаляется, а затем возвращается в систему. Таким образом, простое удаление модуля драйвера будет не эффективно в этом случае. Более надёжный способ: удалить драйвер с помощью команды **rmmod** и затем переименовать его в `/lib/modules`, дабы быть уверенным, что он не будет загружаться.

Завершение гостевых VMX-систем

Гостевые системы VMX выключаются точно также как и паравиртуализированные гостевые системы. Чтобы избежать потерь данных, рекомендуется это делать с помощью команды:

```
poweroff
```

в консоли гостевой системы. После этого нужно выполнить команду:

```
xm destroy vmx_guest_id
```

в консоли домена 0.

Горячие клавиши окна VMX (X или VNC)

Если вы в момент запуска виртуальной машины работаете в среде X, создаётся окно. В этом окне доступно несколько горячих клавиш.

Ctrl+Alt+2 переключает из экрана гостевой системы в управляющее окно. По команде `help` можно посмотреть справочную информацию о командах доступных в консоли. Например, команда `q` прекращает работу гостевой системы. Ctrl+Alt+1 переключает обратно на экран гостевой системы. Ctrl+Alt+3 переключает на окно вывода на последовательный порт. В этом окне показаны перехваченные данные, которые гостевая система выводит на последовательный порт. Работает только в том случае, если гостевая VMX-система настроена на использование последовательного порта.

Сохранение/восстановление и миграция

Гостевые системы VMX в настоящий момент нельзя сохранять, восстанавливать и переносить. Эти возможности сейчас находятся в состоянии активного развития.

Vnets - виртуальные сети доменов

Xen опционально предоставляет поддержку сетей для доменов с помощью vnets. Они эмулируют частные сети доменов. Домены, находящиеся в одном vnet'e, могут размещаться на одной машине или на отдельных машинах, и vnet'ы остаются подсоединёнными во время миграции доменов. Ethernet-трафик внутри vnet'а туннелируется внутрь IP-пакетов передающихся по физической сети. Vnet - это виртуальная сеть и адресация внутри этой сети не имеет никакого отношения к адресации в физической сети. Отдельные vnet'ы или vnet'ы и физическую сеть можно соединить с помощью доменов с более чем одним сетевым интерфейсом и включённой поддержкой forwarding'а или bridging'а.

Поддержка vnet включена в xm и в xend. Команда

```
# xm vnet-create <config>
```

создаёт vnet на основе конфигурационного файла `<config>`. После того как vnet создан, его конфигурация сохраняется xend, vnet становится постоянным, и он будет существовать до тех пор пока он не будет удалён с помощью команды:

```
# xm vnet-delete
```

Список vnets, о которых знает xend, можно получить командой:

```
# xm vnet-list
```

Другие команды по управлению vnet'ами доступны через программу **vn**, входящую в дистрибутив vnet.

Формат конфигурационного файла vnet такой:

```
(vnet (id       )
      (bridge   )
      (vnetif   )
      (security ))
```

Пробелы не имеют существенного значения.

Параметры:

- <vnetid>: vnet id, 128-битный идентификатор vnet. Он может быть задан как 8 4x-разрядных шестнадцатеричных числа, разделенных двоеточиями или, в сокращённой форме, как одно 4x-разрядное число. В последнем случае 7 первых полей заполняется нулями. Идентификаторы vnet id должны быть ненулевыми и идентификатор 1 зарезервирован.
- <bridge>: имя моста для vnet'а. Домены подсоединяются к vnet'у путём подсоединения виртуального интерфейса к этому мосту. Длина имени моста ограничивается ядром до 14 символов.
- <vnetif>: имя виртуального интерфейса vnet (опционально). Интерфейс инкапсулирует и декапсулирует трафик vnet в сеть. Он подсоединён к мосту vnet. Длина имени интерфейса ограничивается ядром до 14 символов.
- <level>: уровень безопасности для vnet (опционально). Уровень может принимать одно из следующих значений
- * `none`: нет безопасности (по умолчанию). Трафик vnet передаётся в открытом виде.
- * `auth`: аутентификация. Трафик vnet аутентифицируется с помощью IPSEC ESP hmac96.
- * `conf`: конфиденциальность. Трафик vnet аутентифицируется и шифруется IPSEC ESP hmac96 и AES-128.

Аутентификация и конфиденциальность поддерживаются в экспериментальном режиме; сейчас в них используются hard-wired ключи.

Когда vnet создаётся, его конфигурация сохраняется xend -- vnet существует до тех пор, пока

его не удалят командой:

```
# xm vnet-delete <vnetid>
```

Интерфейсы и мосты, которые использует vnet, видны по командам **ifconfig** и **brctl** `show`.

Пример

Предположим, содержимое файла `vnet97.sxp` выглядит так:

```
(vnet (id 97) (bridge vnet97) (vnetif vnif97)
      (security none))
```

После этого команда **xm** `vnet-create vnet97.sxp` определяет vnet с идентификатором 97 и без обеспечения безопасности. Мост для vnet'а называется vnet97, а виртуальный интерфейс -- vnif97. Для того чтобы добавить интерфейс домена в этот vnet, нужно установить в конфигурационном файле мост соответствующего интерфейса равным vnet97.

В python'е:

```
vif="bridge=vnet97"
```

В sxp:

```
(dev (vif (mac aa:00:00:01:02:03) (bridge vnet97)))
```

Как только домен стартанул, его интерфейс должен быть виден в выводе команды **brctl** `show` в портах vnet97.

Для получения наивысшей производительности стоит уменьшить MTU доменных интерфейсов, смотрящих в vnet, до 1400. Это можно сделать, например, с помощью команды **ifconfig** `eth0 mtu 1400` или установив строку `MTU=1400` в файле `ifcfg-eth0`. После этого может понадобиться удалить кэшированные файлы для интерфейса eth0 в каталоге `/etc/sysconfig/networking`. Vnet'ы будут работать в любом случае, но производительность может пострадать от IP-фрагментации, вызванной инкапсуляцией vnet'ов и выходом за пределы аппаратного MTU.

Инсталляция поддержки vnet

Vnet реализуется модулем ядра, и модуль должен быть загружен до того как буду использоваться vnet'ы. Это можно сделать до старта xend вручную с помощью команды `vn insmod` или же настроить xend вызывать скрипт `network-vnet`, указав в конфигурационном файле `/etc/xend/xend-config.sxp` строку:

```
(network-script        network-vnet)
```

Этот скрипт загружает модуль ядра с помощью **insmod**, а затем вызывает скрипт `network-bridge`.

Код vnet по умолчанию не компилируется и не устанавливается. Для того чтобы откомпилировать код и установить его на текущей системе, нужно вызвать `make install` в корне дерева исходников vnet, `tools/vnet/`. Также можно установить vnet в инсталляционный каталог с помощью `make dist`. Подробности об этом можно найти в `Makefile` в каталоге исходных текстов.

По умолчанию модуль vnet создаёт интерфейсы `vnif0002`, `vnif0003` и `vnif0004`. Проверить работают vnet'ы или нет можно так: настроить на них IP-адреса и попробовать их с помощью **ping**. Например, для машин hostA и hostB:

```
hostA# ifconfig vnif0004 10.0.0.100 up
hostB# ifconfig vnif0004 10.0.0.101 up
hostB# ping 10.0.0.100
```

Реализация vnet использует IP-multicast для обнаружения интерфейсов vnet, поэтому все машины, на которых есть vnet'ы, должны быть доступны по multicast'у. Сетевые коммутаторы часто сконфигурированы так, что бы не ретранслировать multicast пакеты, это означает что все машины, использующие vnet, должны быть в том же сегменте сети на канальном уровне, за исключением случая, когда сконфигурирован vnet forwarding.

Проверить покрытие multicast можно пропинговав multicast-адрес vnet:

```
# ping -b 224.10.0.1
```

Должны прийти ответы от всех машин, на которых работает модуль vnet. Посмотреть передаются ли и принимаются ли пакеты vnet можно просматривая трафик UDP на порту vnet:

```
# tcpdump udp port 1798
```

Если многоадресные (multicast) пакеты не передаются между хостами, multicast forwarding можно настроить с помощью **vn**. Предположим, есть хост hostA с адресом 10.10.0.100 и hostB с адресом 10.11.0.100, и multicast forwarding между ними не выполняется. Настройка передачи между хостами выполняется с помощью **vn**:

```
hostA# vn peer-add hostB
hostB# vn peer-add hostA
```

Multicast forwarding нужно использовать очень осторожно -- нужно избегать петель. Как правило, только одну машину в сети нужно настраивать на передачу (forward), х она будет передавать многоадресные пакеты, полученные от других машин в сети.

Глоссарий

Домен

Домен это контекст исполнения, содержащий работающую *виртуальную машину*. Отношение между доменом и виртуальной машиной в Xen приблизительно такое же как между процессом и программой в операционной системе: виртуальная машина это постоянная сущность, находящаяся на диске (как программа). Когда она запущена на исполнение, она работает в домене. У каждого домена есть *идентификатор домена*.

Домен 0
Первый домен, который запускается на Xen-системе. Домен 0 отвечает за управление системой.

Идентификатор домена (Domain ID)
Уникальный идентификатор *домена*, аналогичный идентификатору процесса (PID) в операционной системе.

Полная виртуализация
Способ виртуализации, который не требует модификации гостевой операционной системы; гостевая ОС находится в иллюзии полноценной машины с реальными устройствами.

Гипервизор
Альтернативное название для *монитора виртуальных машин*, означающее "над супервизором". Термин подчеркивает что монитор виртуальных машин управляет множеством ядер системы, супервизоров.

Живая миграция (Live Migration)
Техника, позволяющая переносить работающую виртуальную машину с одного хоста на другой без её остановки.

Паравиртуализация
Способ виртуализации, требующий модификации гостевой операционной системы. Xen использует паравиртуализацию, но сохраняет бинарную совместимость для пользовательских (user space) приложений.

Теневая таблица страниц (Shadow pagetables)
Техника, с помощью которой расположение страниц в оперативной памяти скрывается от гостевой системы. Используется некоторыми мониторами для создания иллюзии непрерывной физической памяти. В Xen используется при выполнении *живой миграции*.

Виртуальное блочное устройство (Virtual Block Device, VBD)
Постоянное хранилище информации, доступное виртуальной машине, представляющее абстракцию блочного устройства хранения. Виртуальным блочным устройством может быть настоящее блочное устройство, образ файловой системы в файле или удалённое/сетевое хранилище.

Виртуальная машина (Virtual Machine, VM)
Среда, в которой выполняется гостевая операционная система, предполагающая что выполняется на выделенной машине. Виртуальная машина может быть идентична нижележащей аппаратной машине (*полная виртуализация*), а может отличаться (*паравиртуализация*).

Монитор виртуальных машин (Virtual Machine Monitor, VMM)
Программное обеспечение, позволяющее множеству виртуальных машин одновременно выполняться на одной физической машине.

Xen
Монитор виртуальных машин, использующий паравиртуализацию, разработанный преимущественно исследовательской группой System Research Group Компьютерной лаборатории Кембриджского Университета.

XenLinux

Название порта ядра Linux, работающего под Xen