Константин Лепихов

Раздолье в серверном заповеднике

Современные компьютеры становятся все быстрее и быстрее, и грех этим не воспользоваться. Однако и тут есть свои подводные камни. Предположим, у вас мощный компьютер с большим объемом оперативной и дисковой памяти, и вы хотите создать мощный сервер, предоставляющий множество различных сервисов. «Без проблем», — скажете вы — и будете не правы, потому что нужно еще сделать так, чтобы сервер можно было легко администрировать и чтобы само решение было максимально безопасным.

Что это и кто это?

На сегодняшний день существует много технологий, позволяющих сделать жизнь администраторов спокойнее. Это и изоляция сервисов (chroot), и защита от переполнений буфера (owl или grsecurity), и разграничение привилегий (например, LIDS или RSBAC). Но согласитесь, было бы все же лучше, если бы некоторые небезопасные сервисы можно было физически перенести на другой сервер (например, сервисы баз данных, различные IM-сервисы и т. д.). «Как, — скажете вы, — у нас уже есть один сервер, самый мощный и лучший, зачем нам еще?» Хорошо бы сделать так, чтобы на базе одного физического сервера можно было создать несколько виртуальных и на каждый из них установить свой дистрибутив и настроить его по вкусу.

Видимо, подобные мысли не давали покоя программисту и администратору Жаку Глина, и он придумал решение, которое было названо VServer (изначально оно носило имя Virtual Private Server, но когда так стали называть другой проект, было принято решение о переименовании). По сути, это сервер внутри сервера — со своими ресурсами, сетевыми интерфейсами, памятью, ограничениями и пользователями. При этом серверы не видят и не влияют друг на друга. Свежую версию VServer можно найти в Интернете по адресу www.linux-vserver.org.

Кому, кроме администраторов, это решение может пригодиться? Говоря словами авторов, «всем, кому нужно управлять сервером». Вот несколько примеров, показывающих, в каких областях это применимо:

- Хостинг. Полностью независимый от ограничений провайдера хостинг (то есть несколько отдельных виртуальных серверов в пределах одного физического).
- Эксперименты с программным обеспечением. Согласитесь,

не совсем правильно производить отладку или разработку программного обеспечения на рабочей системе, а вот делать это на отдельной машине было бы вполне логично.

- Образовательные цели. Например, каждому студенту можно дать свой сервер, с которым он сможет делать все, что захочет. Можно даже дать ему пароль root.
- Изолированные серверы. Например, если нужно работать с каким-то приложением и необходим полный контроль за его действиями.
- «Сервер-приманка» (honeypot). Незаменимая вещь для защиты и исследования сети.
- Поддержка работы нескольких версий серверов и возможность их независимого включения и отключения.

Кстати, и кошмар, связанный с резервным копированием, может уйти в прошлое, ведь теперь сервер — это всего лишь каталог на диске, который можно скопировать когда и куда угодно. А основная система — всего лишь минимальный набор пакетов и ядро с vs-патчем. В системах промышленного уровня раздел с виртуальными серверами может храниться на LVM, который находится поверх DRBD и т. д. Если вы используете файловую систему xfs, можно делать «снимки» раздела и сохранять их на устройстве для резервной записи — в общем, есть где разгуляться фантазии настоящего администратора. В случае хостинга есть тоже одно неоспоримое преимущество — реальный биллинг, который можно гибко настраивать в пределах одной системы (ведь у каждого сервера только один IP, и его невозможно поменять или подделать).

Архитектура VServer

Linux VServer состоит из двух частей:

- Патч к стандартному ядру Linux. Он добавляет поддержку контекстов безопасности, воплощающих основную идею VServer. Они позволяют изолировать каждый VServer внутри основной системы и исключить их взаимное влияние друг на друга. Процесс, выполняющийся в рамках определенного контекста, может видеть лишь такие же процессы и использовать только IP-адрес, присвоенный данному контексту. Процессам запрещено создавать устройства и открывать прямой доступ к ним (при этом данные ограничения действуют даже для суперпользователя).
- VServers Tools. Набор программ для создания и управления виртуальными серверами. В основном все они написаны на языке командной оболочки, что позволяет их легко модифицировать под свои нужды. В том числе в VServers Tools включена и программа VServer, позволяющая запускать, останавливать и заходить в виртуальный сервер.

Изолированные области

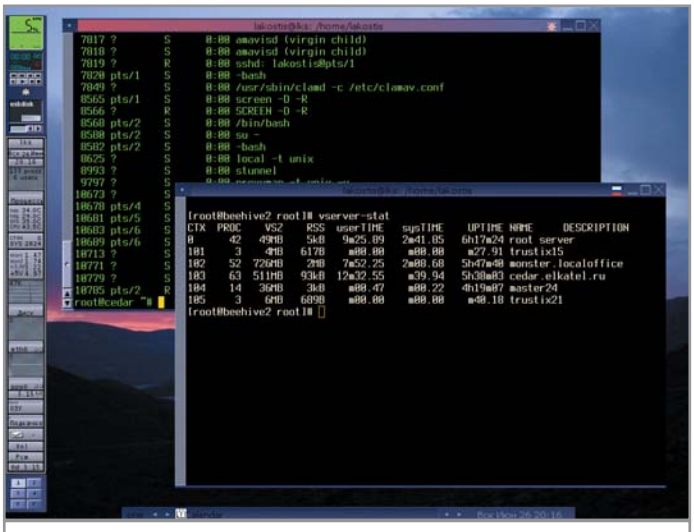
Все системы на базе Linux или Unix имеют специальный вызов — chroot(). Он требуется в том случае, если нужно «посадить» процесс в отдельную директорию. После этого процесс, для которого был сделан chroot(), будет считать, что директория, в которой он находится, является корневым каталогом. Вызов chroot() нельзя отменить. Единственное, что остается делать процессу в этой ситуации, — это отправляться в директории уровнем ниже, повторяя chroot() снова и снова. Вы, наверное, уже догадались, к чему шло все это вступление: основная идея технологии VServer — системные вызовы, позволяющие полностью изолировать запуск и исполнение процессов в определенных местах внутри сервера.

Виртуальный сервер изолирован в четырех областях:

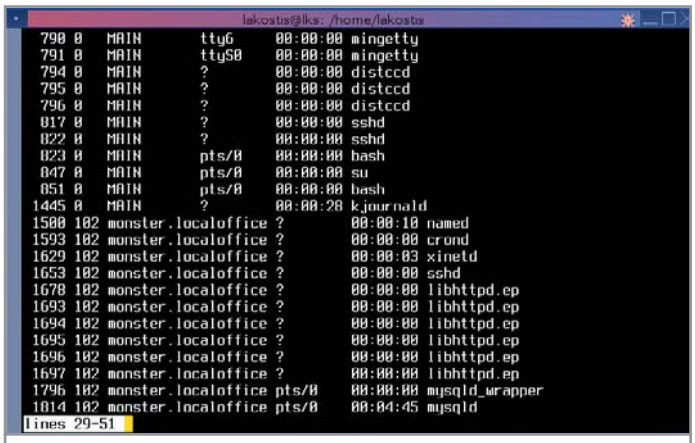
- Файловая система. Каждый VServer «посажен» в отдельную директорию главного сервера и не может «убежать» из из нее. Это ограничение реализовано через вызов chroot(), имеющийся в любой Unix-совместимой системе.
- Процессы. VServer может видеть только те процессы, которые находятся в одинаковом с ним контексте безопасности. Даже сервер с контекстом root может видеть только свои процессы (это сделано для того, чтобы он был более безопасным в использовании). Для просмотра всех процессов на всех виртуальных серверах существует специальный механизм (контекст с номером 1).
- Сеть. Каждый VServer имеет уникальные имя и IP-адрес, которые могут быть использованы сервером для сетевых соединений и ответов на запросы. Кроме того, это ограничение незаметно для VServer.
- Суперпользователи и их возможности. Суперпользователь, работающий с VServer, имеет меньше привилегий по сравнению с тем, который работает в обычной системе Linux. Например, он не может настраивать и отключать сетевые интерфейсы и многие параметры системы, а также подсоединять или отсоединять файловые системы, работать с блочными устройствами и т. д. Грубо говоря, суперпользователь VServer имеет полный доступ только к тому, что в нем есть, а именно — к файлам и процессам, и большего ему не дано.

Взаимодействие между процессами (System V IPC)

Все внутренние IPC-ресурсы изолированы от IPC-ресурсов других VServer. Для выбора необходимого в качестве ключа используется номер контекста безопасности каждого VServer.



Работа утилиты vserver-stats: статистика активных виртуальных серверов; на заднем плане ssh-сессия с одним из VServer



Список процессов на «родительской» системе вместе с запущенными процессами на VServer

Это позволяет создать удобную и гибкую среду для работы виртуального сервера. И каждая из перечисленных возможностей может быть использована независимо от других.

Безопасность

Даже если вами было создано окружение, в котором процессы ограничены лишь собственным виртуальным миром и общаются с сетью только через выделенный IP-адрес, вы должны сделать так, чтобы и ущерб, который могут вам причинить эти процессы, был минимальным. Для этого нужно создать виртуальные среды и дать им ограниченные привилегии root. Как же можно ограничить процессы с правами root, которые обладают таким контролем над системой (например, могут просто перезагрузить компьютер)? Введите систему возможностей (capabilities). Идея не нова, но мы считаем, что очень многие пользователи никогда не слышали о существовании подобной системы.

В старые времена систем Unix/Linux пользователь root (обладающий идентификатором 0) имел возможность делать такие вещи, о которых другие пользователи даже не могли и мечтать. Весь доступ к ядру, системным вызовам или просто некоторым ресурсам прекращался, если идентификатор пользователя (или процесса, запущенного этим пользователем) не был равен нулю. Для того чтобы процесс с ID 0 мог хоть немного понизить привилегии, существовал только один путь — менять идентификатор на отличный от нуля. Одним словом, либо все, либо ничего. И вот тут-то и появилась система возможностей.

Система возможностей

Сегодня все, что отличает root от других пользователей, — это набор возможностей. Пользователь root обладает всеми возможностями, а другие — нет; ID 0 больше уже ничего не значит — просто красивая цифра. На данный момент насчитывается около 30 возможностей. Можно сделать так, чтобы процесс утратил какие-либо возможности навсегда и они к нему больше никогда не вернулись.

Возможности позволяют уменьшить силу root — и это именно то, что нужно, если вы хотите создать собственного суперпользователя. Процесс с его правами, запущенный внутри виртуального окружения, к примеру, может открыть порт перед 1024, но окажется не в состоянии отключить или перенастроить параметры сетевого интерфейса. Рассмотрите внимательно файл /usr/include/linux/capability.h — и вы увидите, что это вполне реально.

Для VServer было придумано несколько новых возможностей: например, new_s_context, позволяющая создать новый контекст, а также set_ipv4root, способствующая «привязке» всех процессов внутри контекста к одному IP-адресу. Кстати, все эти ограничения распространяются и на потомков процессов. Однако новые возможности нельзя отключить обычным способом, то есть мы «посадили» VServer в chroot/s_context/ipv4root, но «убежать» его процессы оттуда уже не смогут. Таким образом, используя существующие и новые системы возможностей, мы можем сделать для VServer некую среду, где root будет ограничен в своих правах и не сможет влиять на работу основного сервера.

Для работы с системой возможностей в состав vserver-utils входят утилиты reducecap и chcontext. Мы продемонстрируем, как функционируют они и работает сама система возможностей, на конкретном примере.

Работаем с контекстами

Утилита chcontext позволяет зайти в определенный контекст безопасности, выделенный для VServer, и запустить программу, указанную в качестве аргумента, с правами, ограниченными этим контекстом. Таким образом, эта программа уже не сможет видеть процессы, запущенные на основном сервере.

Например, запустим два окна с приложением (пусть это будет xterm), с одинаковыми ID пользователя.

В первом окне зададим команду:

```
xterm
```

Во втором окне выполним chcontext, запускающую командную оболочку. Затем попробуем ввести pstree и обнаружим, что видно нам совсем немногое. Потом попробуем «убить» xterm, находящуюся в первом окне, и убедимся, что этого сделать нельзя. Теперь выходим из оболочки и возвращаемся к привычным возможностям.

Во втором окне укажем следующее:

```
/usr/sbin/chcontext /bin/sh
pstree
killall xterm
exit
```

Таким же образом можно порождать внутри контекстов другие контексты, которые будут к тому же изолированы друг от друга.

Понижаем возможности

Утилита reducecap позволяет ограничить количество возможностей процесса и его субпроцессов. Даже программа setuid не сможет увеличить число возможностей, ограниченное с помощью reducecap.

Рассмотрим это ниже на конкретном примере:

```
# Мы сейчас не с правами root
# Посмотрим текущий потолок возможностей
```

```
cat /proc/self/status
# Строка capBset показывает в основном 1
/usr/sbin/reducecap --secure /bin/sh
cat /proc/self/status
# capBset теперь показывает гораздо больше 0
# capEff теперь вся из 0, то есть у нас нет больше никаких
привилегий
# теперь попробуем стать root
su
cat /proc/self/status
# caEff содержит теперь гораздо меньше 0
# но попробуем проверить, root ли мы на самом деле
tail /var/log/messages
# это мы можем
/sbin/ifconfig eth0
/sbin/ifconfig eth0 down
# а это уже нет, таким образом, мы потеряли свои возможности
# супермена
exit
```

Работаем с VServer

Итак, после небольшого вступления попробуем поработать с VServer. Для этого понадобится дистрибутив Linux (на базе RPM или APT), ядро с vs-патчем и набор vserver-utils.

В дистрибутиве ALT Linux это может выглядеть таким образом:

```
# ставим ядро с поддержкой VServer
$ apt-get install kernel-image-vs-smp
# ставим утилиты для работы с VServer
$ apt-get install util-vserver
```

Теперь создадим наш маленький VServer:

```
$ vserver build <name>
```

В данном случае <name> — это произвольное имя виртуального сервера.

Далее утилита VServer создает структуру будущего виртуального сервера на базе текущего дистрибутива основного сервера.

Теперь зайдем в наш новый виртуальный сервер:

```
$ vserver enter <name>
```

Здесь <name> — также произвольное имя виртуального сервера.

После выполнения этой операции получаем стандартное приглашение командной оболочки, как будто бы мы зашли на этот сервер с клавиатуры. Редактируем конфигурацию по вкусу, не упуская из вида тот факт, что теперь у нас один IP-адрес — и все службы, работающие на этом сервере, должны быть привязаны только к нему.

Теперь запускаем наш VServer:

```
$ vserver start <name>
```

И снова <name> — произвольное имя виртуального сервера.

Теперь этот сервер доступен другим и может отвечать на запросы извне (например, если запущен ssh, мы можем управлять им по ssh). Если возникнет необходимость остановить этот сервер, нужно просто набрать следующее:

```
vserver stop <name>
```

В процессе создания VServer одноименная утилита записывает в каталог /etc/vservers его конфигурацию, которая состоит из двух файлов: первый описывает общие параметры (номер контекста, IP-адрес, включенные возможности), а второй — порядок действий, выполняемых на этапе запуска/остановки. Для удобства обслуживания нескольких виртуальных серверов существует скрипт init.d, позволяющий производить их запуск или остановку выборочно. Существуют еще и утилиты для унификации содержимого VServer, просмотра процессов, остановки/перезагрузки, а также многие другие. Все они входят в состав util-vserver или могут быть написаны самостоятельно, поскольку в основном это скрипты на языке командной оболочки.

Совместимость с другими security-патчами

Предположим, вы создали виртуальный сервер и убедились, что он работает, но все же чего-то не хватает. Все правильно, полной уверенности не бывает. Ведь вы оградили себя системой привилегий, возможностей, заперли всех в контексты, но забыли самое главное — защиту «родительской» системы от самой себя и собственных ошибок. Переполнение буфера, ненадежные права доступа, низкое качество кода выполняемых программ и ошибки ядра — все это осталось по-прежнему с вами: общие меры по защите никто не отменял. Обычно применяют патчи к ядру, исправляющие его код и добавляющие улучшения, связанные с безопасностью (например, патч Openwall для ядер 2.4 и патч PaX для ядер 2.6). Также необходимо соблюдать общие принципы безопасного администрирования — запрещать то, что не разрешено, и не включать то, что не нужно.

Аналоги и аналогии

Если вы хотите объяснить собеседнику что-то, о чем он не имеет совсем никакого представления, вы наверняка прибегаете к аналогиям. При описании VServer тоже сложно удержаться от этого метода повествования. Кажется, все, о чем здесь идет речь, вы уже где-то видели или слышали. Да, общие идеи витают в воздухе, поэтому неудивительно, что они приходят в головы разных людей почти одновременно. Мы

Накладываем ограничения

Квоты и лимиты

На сегодняшний день для VServer есть несколько патчей (некоторые из них уже вошли в нестабильную версию VServer для ядра 2.6), позволяющих лимитировать ресурсы, которые потребляются, во-первых, самим сервером, а во-вторых, пользователем внутри него. Например, столь популярные дисковые квоты можно использовать и внутри VServer, а также задавать их для каждого сервера в отдельности (при этом внутри VServer дисковое пространство будет ограничено данной квотой). Можно ограничивать количество запущенных процессов внутри VServer, задавать глобальный nice-уровень для всех процессов и т. д. Кроме того, есть возможность управлять доступом к каталогу /proc, что делает работу основного сервера более безопасной.

же попытаемся понять, на что похож VServer, попутно рассказывая об аналогичных технологиях.

Виртуальные машины

В чем-то VServer является виртуальной машиной, но у него нет типичных недостатков, присущих любым виртуальным машинам или эмуляторам:

- Скорость работы. VServer действительно работает с той же скоростью, что и обычный железный сервер, так как в данном случае ничто не эмулируется и не «подделывается». То есть, он требует не больше ресурсов, чем, скажем, VMWare или qemu. С другой стороны, в VServer не существует такого понятия как блочное устройство, но имеются раздел, память и процессор с сетевым интерфейсом: процессам обычно большего и не требуется.
- Архитектура. Это слабое место VServer. Нет возможности установить Windows или FreeBSD — вы ограничены архитектурой, поддерживаемой «родительской» системой. Впрочем, не стоит забывать, что это не эмулятор.
- Ядро. В VServer существует единое ядро для всех. Это обстоятельство можно рассматривать и как удобство (согласитесь, лучше менять ядро одновременно на всех серверах сразу, чем переставлять на каждом отдельно), и как недостаток (по сравнению, скажем, с UML или Xen, о которых речь пойдет немного ниже).
- Управление памятью и ресурсами. В Xen можно назначить на каждую машину даже свой планировщик, а в VServer один имеется лишь один планировщик. В тестируемой ветке для ядра версии 2.6 встречаются идеи того, как это можно сделать и для VServer.
- Расширенный мониторинг процессов и потребляемых ресурсов. Находится в стадии активной разработки, и скоро эти функции можно будет использовать в полной мере.

Проекты, похожие на VServer

VE — Virtual Evironment

Еще в 2001 году существовал интересный проект Virtual Evironment, впоследствии переименованный в Virtuozo. Разрабатывала его компания SWSoft (та самая, которая подарила нам ASPLinux). Потом проект как-то тихо перестал развиваться — вполне возможно, его вытеснил User Mode Linux. Архив VE или Virtuozo (aspcomplete/) можно найти в Интернете на странице Пола Слейдена, посвященной VServer и размещенной по адресу www.paul.sladen.org/vserver.

UML — User Mode Linux

Интересный проект, интересная идея. Можно запускать собственные ядро и систему в рамках основной системы, перегружать и отлаживать это самое ядро и многое другое. Еще до появления непосредственно VServer данный проект часто использовался в образовательных целях, а также находил применение в сфере хостинга и отладки программного обеспечения. Фантазия ограничена только Linux. Подробную информацию по данному вопросу можно найти в Сети по следующему адресу: http://user-mode-linux.sourceforge.net.

VPS — Virtual Private Servers

Коммерческий проект, представляющий собой ответвление VServer, целиком и полностью предназначенный для организации хостинга на базе VPS. Ресурсы, ориентированные на хостинг с помощью VServer: OpenVPS (www.openvps.org) и FreeVPS (www.freevps.com).

Xen — The Xen virtual machine monitor

По сути, это микроядро для Linux, позволяющее запускать ядра других ОС. Поддерживаются изоляция, ограничения и управление ресурсами. Довольно интересный и многообещающий проект (вполне годится в качестве замены Win4Lin или VMWare ESX/GSX, так как имеет опыт успешной эксплуатации Windows в среде Xen). Подробная информация размещена на сайте www.cl.cam.ac.uk/Research/SRG/netos/xen/index.html.

RAD GNU/Linux

Проект, поддерживаемый Петром Савельевым, целью которого является создание дистрибутива для маршрутизаторов — простого в настройке и чем-то похожего на системы Cisco. RAD GNU/Linux является хорошим примером применения ALT Linux Sisyphus для разработки дистрибутивов. В системе используется технология VServer. Сайт: www.radlinux.org.

Заключение

В завершение хотелось бы привести цитату разработчиков VServer, чтобы лучше понять, как они представляют себе будущее этой технологии.

«Идея виртуального сервера интересна, поскольку позволяет достичь более высокого уровня безопасности и сократить количество административных задач, необходимых для его обслуживания — например, обычных операций, таких как резервное копирование и обмен данными. Службы вроде мониторинга можно настроить лишь единожды (так как физический сервер всего один).

Сервер на базе Linux может одновременно выполнять множество задач с высоким уровнем надежности и доступности. В процессе развертывания обычного сервера, как правило, многие сталкиваются со стандартной проблемой, когда запускаются и устанавливаются различные сервисы и службы, многие из которых не используются или используются редко. В процессе работы конфигурация многих таких служб теряется или забывается, и многие тайны настройки бывают утрачены (вроде как настроил и забыл). Поэтому при переезде на новый сервер всегда что-либо теряется (иногда что-то очень важное — например, почтовый ящик начальника). VServer решает эту проблему: вы делаете установку только один раз, а в дальнейшем работаете только с каталогом этого сервера, который можно легко скопировать и перенести на любое новое место — от обычного раздела на жестком диске до LVM-страйпа на внешнем FC дисковом массиве».

Учитывая все выше изложенное, вывод напрашивается сам собой: VServer — наиболее удобный и предпочтительный метод установки и использования сервера на базе операционной системы Linux. |