Виртуальный репозиторий Debian 10

Содержание

Постановка задачи

Требуется создать виртуальный сервер для гипервизора QEmu/KVM, реализующий следующие репозитории дистрибутива Debian 10 для архитектуры amd64:

- зеркало основного репозитория
- зеркало репозитория с обновлениями безопасности
- репозиторий собственных пакетов

Доступ к репозиториям должен осуществляться по протоколу http. Должны быть предприняты разумные меры защиты сервера в условиях получения сервером сетевых настроек по протоколу dhcp. Следует предусмотреть возможность управления сервером посредством ssh-подключения. Сам сервер также должен работать под управлением ОС Debian 10 (amd64).

Предварительные замечания

Во фрагментах терминальных сеансов и при указании выполняемых команд используются следующие соглашения:

Выполняемые при настройке системы команды выделены жирным шрифтом, отклик системы или выдержки из конфигурационных файлов и сценариев такого выделения не имеют. Все команды приводятся с полным отображением приглашения командной строки, в котором отражается пользователь, от имени которого выполняется команда, и текущий рабочий каталог. Оба этих фактора значимы в большинстве приведённых команд и листингов.

Поскольку рассматриваемый виртуальный сервер будет использоваться в нескольких виртуальных сетях, то вместо конкретных доменных имён и учётных записей пользователей здесь будут использоваться абстрактные user, localhost, localdomain и т. д.

Создание виртуальной машины

Создание виртуальной машины выполнено средствами Virtual Machine Manager (virt-manager). При выборе настроек виртуальной аппаратуры большинство параметров сохранили значения по умолчанию, следует отметить лишь следующее:

Для виртуальной машины создан виртуальный жёсткий диск формата qcow2 размером 200 Гб, подключенный к машине по шине VirtIO.

Серверу выделен один одноядерный процессор, копирующий конфигурацию ядра процессора системы-носителя (хоста).

Серверу выделен 1 Гб оперативной памяти.

После установки системы виртуальный жёсткий диск оставлен единственным загрузочным устройством.

Сетевая карта сервера подключена к виртуальной сети серверов, которая является открытой (open), маршрутизируемой и не использующей встроенный dhcp-сервер гипервизора.

Установка системы и разметка диска

Установка системы выполнена без графической оболочки с минимальным набором программного обеспечения (базовая система, стандартные утилиты, служба ssh).

Виртуальный жёсткий диск разбит на разделы следующим образом (в выводе ниже опущены строки, не касающиеся виртуального жесткого диска):

```
root@debian:/# /usr/sbin/fdisk -l /dev/vda
Disk /dev/vda: 200 GiB, 214748364800 bytes, 419430400 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x9f5bc31c

Device     Boot    Start       End   Sectors   Size Id Type
/dev/vda1  *        2048  19531775  19529728   9,3G 83 Linux
/dev/vda2       19531776  23437311   3905536   1,9G 83 Linux
/dev/vda3       23437312  25391103   1953792   954M 82 Linux swap / Solaris
/dev/vda4       25391104 419428351 394037248 187,9G 83 Linux

root@debian:/# mount -l
/dev/vda1 on / type ext4 (rw,relatime,errors=remount-ro)
/dev/vda4 on /var/www type ext4 (rw,nosuid,nodev,noexec,relatime)
/dev/vda2 on /tmp type ext4 (rw,nosuid,nodev,relatime)

root@debian:/# df -h
Файловая система Размер Использовано  Дост Использовано% Смонтировано в
/dev/vda1          9,2G          1,2G  7,6G           13% /
/dev/vda4          184G           70G  105G           40% /var/www
/dev/vda2          1,9G          5,7M  1,7G            1% /tmp
```

Такая схема разметки диска выбрана по следующим соображениям:

DOS-таблица является наиболее простой и минималистичной, что оптимально для виртуального сервера. На раздел подкачки выделено около 1 Гб, остальное дисковое пространство разбито на три раздела, благодаря чему каталог с репозиториями и каталог временных файлов размещены в отдельных файловых системах. Это сделано в рамках базовых мер по защите сервера. Большая часть диска отведена именно под хранение репозиториев, тогда как для остальных компонентов системы выделено минимально достаточное (с некоторым запасом) дисковое пространство. Следует помнить, что журналы системы также расположены в корневой файловой системе.

Базовая настройка виртуального сервера

В этом разделе описаны меры, применимые ко многим подобным виртуальным серверам. В частности рассмотрены способы минимизации числа служб, настройки сетевой подсистемы и некоторых систем безопасности.

Отключение IPv6

Отключение поддержки IPv6 ядром системы может быть выполнено как для текущей сессии, так и на постоянной основе. Для отключения на постоянной основе следует привести содержимое файла **/etc/sysctl.conf** к виду, не противоречащему следующим строкам в нём:

```
#Disabling Ipv6
net.ipv6.conf.all.disable_ipv6=1
net.ipv6.conf.default.disable_ipv6=1
net.ipv6.conf.lo.disable_ipv6=1
```

Для отключения только в текущей сессии достаточно выполнить команды:

```
root@debian:/# echo '1' > /proc/sys/net/ipv6/conf/all/disable_ipv6
root@debian:/# echo '1' > /proc/sys/net/ipv6/conf/default/disable_ipv6
root@debian:/# echo '1' > /proc/sys/net/ipv6/conf/lo/disable_ipv6
```

Для проверки успешности отключения следует перезапустить сетевую подсистему командой

```
root@debian:/# service networking restart
```

и в выводе команды

```
root@debian:/# ip address show
```

убедиться в отсутствии назначенных адресов IPv6:

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
      valid_lft forever preferred_lft forever
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
   link/ether 52:54:00:79:69:b1 brd ff:ff:ff:ff:ff:ff
   inet 192.168.7.2/24 brd 192.168.7.255 scope global enp1s0
      valid_lft forever preferred_lft forever
```

Отключение Magic SysRq

Отключение Magic SysRq может показаться избыточным для виртуальной машины или даже вредным: так, например, в случае сбоя системы не будет возможности её аварийно остановить или сбросить содержимое буферов на диск. Однако следует помнить, что сервер виртуальный, а хранимые на нём данные легко восстановимы. В то же время есть, пусть и призрачно малая, вероятность того, что некий злоумышленник тем или иным способом сможет имитировать нажатие подобной комбинации клавиш. Это может привести к остановке виртуального сервера и отказам в обслуживании клиентов. При этом защита организовывается настолько просто, что пренебрегать ей не рационально.

Для отключения «волшебных» сочетаний в текущей сессии достаточно команды:

```
root@debian:/# echo '0' > /proc/sys/kernel/sysrq
```

Для отключения на постоянной основе достаточно привести файл **/etc/sysctl.conf** к виду, не противоречащему следующей строке из него

```
kernel.sysrq=0
```

Создание пользователей

Далее при описании службы защищённого удалённого входа ssh указано, что вход в систему удалённо от имени суперпользователя заблокирован. Чтобы всё же иметь возможность удалённого входа, следует создать обычного пользователя, от имени которого и будет осуществляться удалённый вход. После открытия сессии в системе под такой учётной записью появляется, при необходимости, возможность повысить свои полномочия при помощи su. С точки зрения безопасности такой механизм является дополнительным рубежом защиты: злоумышленнику необходимо подобрать не только пароль суперпользователя, но и логин и пароль обычного пользователя.

Создание такого пользователя (пусть его логин – maintainer) можно выполнить как на этапе установки системы, так и после, например, при помощи команды

```
root@debian:/# /usr/sbin/adduser maintainer
```

Кроме создания пользователя, необходимо обратить внимание на несколько аспектов его работы в системе.

Во-первых, по умолчанию вновь созданный домашний каталог этого пользователя доступен для чтения всем в системе. Учитывая, что в этом каталоге будет храниться закрытая часть ключа для подписи репозиториев, пусть и защищённая парольной фразой, следует несколько изменить права доступа к нему командой

```
root@debian:/# chmod 0750 /home/maintainer/
```

оставив, таким образом, полный доступ к каталогу для самого пользователя и доступ на чтение для членов его группы (коих кроме него самого в системе всё равно нет). Для всех остальных доступ окажется закрыт.

Вторым аспектом является использование sudo. Эта команда позволяет выполнять команды с административными привилегиями не входя в систему от имени суперпользователя. Такой подход таит в себе некоторую опасность: злоумышленник, зная логин и пароль такого пользователя может получить весьма обширные права в системе (если политики sudo настроены) или практически неограниченные, если настройки sudo приняты по умолчанию. Для решения этой проблемы проще всего (и наиболее оптимально на виртуальном сервере с одним пользователем и одним реализуемым сервисом) воспользоваться следующим подходом: не использовать sudo вообще, вынуждая пользователя для повышения своих прав в системе вводить (и, соответственно знать) пароль суперпользователя.

Таким образом, чтобы лишить пользователя возможности использования sudo достаточно исключить его из группы wheel, если он в ней состоит. Сделать это можно командой usermod, оставив пользователя maintainer только в своей (одноимённой) группе. Убедиться в успешности изменений можно выполнив от имени этого пользователя строго после повторного входа в систему команду groups. Далее приведён фрагмент терминального сеанса, демонстрирующий все эти операции:

```
root@debian:/home/maintainer# /usr/sbin/usermod -G maintainer maintainer
root@debian:/home/maintainer# su maintainer
maintainer@debian:~$ groups
```

```
maintainer
maintainer@debian:~$ exit
exit
root@debian:/home/maintainer#
```

В завершение вышесказанного о пользователе следует отметить, что эта учётная запись может использоваться не только для обеспечения защиты входа в систему, но и для хранения вспомогательной информации: заметок, собственных пакетов, ещё не внесенных в персональный репозиторий и т. п.

Настройка списков доступа

Многие демоны используют файлы **/etc/hosts.allow** и **/etc/hosts.deny** как источники информации о том, кому разрешено пользоваться услугами этих демонов. Эти конфигурационные файлы являются частью механизма tcp wrappers. Кроме того, в системе может быть установлен демон tcpd, который сам выполняет такую проверку, прежде чем передать сетевой пакет, инициирующий соединение, целевому демону.

Чтобы выяснить, использует ли демон некоторой службы этот механизм, как правило достаточно убедиться, что исполняемый файл демона собран с использованием библиотеки **libwrap**. Следующий фрагмент терминального сеанса демонстрирует, что демон sshd использует эту библиотеку, а lighttpd – нет:

```
root@debian:/# ldd /usr/sbin/lighttpd | grep libwrap
root@debian:/# ldd /sbin/sshd | grep libwrap
        libwrap.so.0 => /lib/x86_64-linux-gnu/libwrap.so.0 (0x00007f8ea045c000)
```

Проверить, установлен ли в системе демон tcpd, можно командой

```
root@debian:/# dpkg -l | grep tcpd
```

Как видно из пустого отклика команды, этот демон не установлен.

В общем случае такая проверка выполняется файерволлом, а механизм tcp wrapper считается устаревшим и уступившим место как раз сетевым экранам. Тем не менее, учитывая простоту настройки и нетребовательность к ресурсам, имеет смысл внести в указанные конфигурационные файлы записи для всех используемых системой демонов, оставив, тем самым, решение по использованию этого механизма разработчикам демонов и сборщикам дистрибутива. Иными словами, если ими будет принято решение включить поддержку в новой версии ПО, то настраиваемая система будет автоматически готова к таким изменениям. Кроме того, никакое ПО не застраховано от ошибок разработчиков и уязвимостей. Использование этого механизма может оказаться дополнительным рубежом защиты при компрометации сетевого фильтра.

В данной системе не имеет смысла установка tcpd, т. к. ssh использует этот механизм самостоятельно, а для lighttpd ограничений на источник подключения нет.

На рассматриваемом сервере всего два таких демона: sshd и lighttpd. Последний, в данный момент, не использует этот механизм. Зная, что подключаться по протоколу ssh разрешено только с сервера подсети, имеющего адрес 192.168.7.1, а по протоколу http – откуда угодно, следует привести файл **/etc/hosts.allow** к следующему виду (комментарии для краткости опущены):

```
lighttpd : ALL
sshd : 192.168.7.1
```

В свою очередь файл **/etc/hosts.deny** запрещает любые другие подключения к серверу (комментарии также не приведены):

```
ALL: ALL
```

После перезапуска служб убедиться в том, что ssh не принимает соединений с посторонних хостов, можно перезапустив эту службу командой

```
root@debian:/# systemctl restart sshd
```

и попытавшись подключиться по ssh с любого отличного от сервера подсети хоста. Разумеется, на момент эксперимента файерволл не должен блокировать такое подключение.

Безусловно настраивать подобные параметры безопасности следует до подключения сервера к сети, а окончательно проверять – сразу после, по возможности защищая проверяемый сегмент сети иными средствами (например файерволлом сервера виртуализации).

Настройка сети

В этом разделе для краткости и однозначности изложения виртуальный сервер репозиториев будет называться гостем, а сервер виртуализации, на котором он расположен и работает, будет называться хостом.

Сетевые настройки гость получает по протоколу dhcp. Однако с точки зрения безопасности и с учётом его расположения в сети следует принять некоторые меры. Так, если допустить, что некоторое устройство (компьютер, виртуальный или физический сервер) в той же подсети попал под полный контроль злоумышленника, то рассматриваемый гость должен быть способен противостоять атакам, не полагаясь исключительно на системы защиты хоста, на котором он расположен. Описание сетевого фильтра будет приведено ниже, здесь же следует отметить, что аппаратный адрес (lladdr, mac) хоста должен быть известен гостю в любых обстоятельствах и должен быть зафиксирован.

Для достижения этих целей на рассматриваемом госте достаточно привести его конфигурационный файл **/etc/network/interfaces** к виду (некоторые комментарии сокращены):

```
source /etc/network/interfaces.d/*

auto lo
iface lo inet loopback

auto enp1s0
iface enp1s0 inet dhcp
   pre-up ip neigh add 192.168.7.1 dev enp1s0 lladdr 52:54:00:2b:1e:a0 nud permanent
```

и перезапустить сетевую подсистему

```
root@debian:/# systemctl restart networking
```

Затем можно убедиться в правильности полученных настроек (разумеется, если dhcp-сервер активен и доступен), выполнив следующие команды:

```
root@debian:/# ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
      valid_lft forever preferred_lft forever
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
   link/ether 52:54:00:79:69:b1 brd ff:ff:ff:ff:ff:ff
   inet 192.168.7.2/24 brd 192.168.7.255 scope global enp1s0
      valid_lft forever preferred_lft forever
root@debian:/# ip neighbour show
192.168.7.1 dev enp1s0 lladdr 52:54:00:2b:1e:a0 PERMANENT
root@debian:/# ip route show
default via 192.168.7.1 dev enp1s0
192.168.7.0/24 dev enp1s0 proto kernel scope link src 192.168.7.2
root@debian:/# hostname -f
debian.service.localdomain
root@debian:/# dnsdomainname
service.localdomain
```

Здесь (в порядке следования команд) показано, что

- гость получил IPv4 адрес и сетевую маску (192.168.7.2/24),
- гость на постоянной (PERMANENT) основе сопоставляет аппаратный и сетевой адреса хоста (192.168.7.1 и 52:54:00:2b:1e:a0)
- гость имеет настроенную таблицу маршрутизации
- гость получил по dhcp полное доменное имя (debian в service.localdomain)

О присвоении полного доменного имени следует сделать ряд замечаний:

Во-первых, домен верхнего уровня (здесь localdomain) не должен вызывать смущения: при правильной настройке dns-сервера локальной сети использование таких доменов возможно и упоминание о них не выйдет за её пределы и не причинит неудобств внешним dns-серверам.

Во-вторых, для того, чтобы гость принимал и использовал доменное имя от dhcp-сервера, гостю не следует назначать имена хоста и домена вручную, т. е. в файле **/etc/hostname** должно содержаться имя:

```
localhost
```

также как и файл **/etc/hosts** должен содержать следующие сведения:

```
127.0.0.1       localhost
127.0.1.1       localhost
```

Иные сведения в рассматриваемом случае в этом файле не нужны: ни сведения о других хостах сети, ни заполненные системой при установке строки, касающиеся настроек IPv6, т. к. поддержка этого протокола была отключена ранее.

Отключение ненужных служб

Соблюдая правило необходимой достаточности, следует отключить и/или удалить все неиспользуемые службы, например: rsh, telnet, rpcbind. Проверить, установлены ли эти службы, можно одной из или сочетанием следующих (и, возможно, некоторых дополнительных) команд:

```
root@debian:/# dpkg -l | grep rpcbind
root@debian:/# which rsh
root@debian:/# ss -l | grep telnet
```

Ни один из этих способов не универсален в том смысле, что однозначно показывает наличие или отсутствие той или иной службы. Как пример можно привести следующий фрагмент терминального сеанса:

```
root@debian:/# which rsh
/usr/bin/rsh
root@debian:/home/maintainer# ss -Htlp
LISTEN   0   128     0.0.0.0:http     0.0.0.0:*   users:(("lighttpd",pid=362,fd=4))
LISTEN   0   128     0.0.0.0:ssh      0.0.0.0:*   users:(("sshd",pid=388,fd=3))
root@debian:/home/maintainer# ss -Hulp
UNCONN   0   0       0.0.0.0:bootpc   0.0.0.0:*   users:(("dhclient",pid=351,fd=7))
root@debian:/# dpkg -l | grep rsh
root@debian:/# ls -l /usr/bin | grep rsh
lrwxrwxrwx 1 root root         21 июл 17 18:40 rsh -> /etc/alternatives/rsh
root@debian:/# ls -l /etc/alternatives | grep rsh
lrwxrwxrwx 1 root root  12 июл 17 18:40 rsh -> /usr/bin/ssh
lrwxrwxrwx 1 root root  28 июл 17 18:40 rsh.1.gz -> /usr/share/man/man1/ssh.1.gz
```

Разбирая приведённый фрагмент, можно отметить следующее:

Первая команда показывает, что в системе присутствует исполняемый файл rsh, однако вторая и третья команды показывают, что демон rsh не слушает никаких tcp или udp портов. Следующая команда сообщает о том, что в системе не установлено пакетов, в названии которых упоминается rsh. При просмотре в подробном режиме свойств исполняемого файла выясняется, что он является символической ссылкой на другой файл, который также является символической ссылкой на исполняемый файл ssh. Таким образом, в данной системе rsh не установлен, вместо этого создан псевдоним для ssh.

Следует отметить, что при на данном сервере при установке в минималистичном режиме ни один из этих сервисов не был установлен изначально.

Использование локального зеркала репозитория при настройке

Поскольку рассматриваемый сервер создавался в том числе как зеркало основного репозитория Debian 10, а также с учётом того, что соответствующий репозиторий был заранее зеркалирован на съемный носитель, представилась возможность использовать это локальное зеркало и для настройки самого сервера без задействования Интернет-канала, который, в общем смысле, не мог считаться безопасным до того, как настройка сервера была завершена.

Для этого достаточно было смонтировать съемный носитель и скопировать зеркала репозиториев (buster и security) в каталоги назначения командами:

```
root@debian:/# mkdir /tmp/disk && mount /dev/sda1/ /tmp/disk && cd /tmp/disk
root@debian:/# cp -r /tmp/disk/buster /var/www
root@debian:/# cp -r /tmp/disk/security /var/www
```

Однако после этого все файлы и каталоги доступны на запись, чтение и исполнение всем в системе. При настройке сервера это допустимо, но при вводе сервера в эксплуатацию следует это изменить. Эти изменения обоснованы и подробно описаны далее.

Затем для использования локальных зеркал, а в последствии и персонального репозитория, достаточно в файл /**etc**/**apt**/**sources.list** внести следующие строки:

```
deb file:/var/www/buster buster main contrib non-free
deb file:/var/www/security buster/updates main contrib
deb file:/var/www/ppa buster main contrib
```

Учитывая роль репозитория в локальной сети и характер его использования, эти три строки можно оставить единственным содержимым этого файла.

Следует также отметить, что в таком режиме доступа к репозиторию сам сервер не нуждается в услугах собственного веб-сервера и может использовать репозиторий сразу после внесения этих строк в файл источников.

Установка вспомогательного ПО

Весьма полезными при управлении сервером репозиториев могут оказаться файловый менеджер Midnight Commander (пакет mc), утилита tree и средство синхронизации с официальными репозиториями debmirror. Для их установки достаточно выполнить команды:

```
root@debian:/# apt-get update
root@debian:/# apt-get install mc tree debmirror
```

Установка и настройка ssh

Установка ssh может быть выполнена как на этапе установки системы, так и после при помощи команд

```
root@debian:/# apt-get update
root@debian:/# apt-get install ssh
```

На данном сервере ssh используется как единственное средство удалённого входа и администрирования. Учитывая положение сервера в сети, ssh принимает соединения на том же сетевом интерфейсе, что и веб-сервер репозиториев. Поэтому должны быть предприняты существенные меры по защите системы удалённого администрирования. Эти меры распределены на три эшелона: сетевой фильтр, базовую настройку сервера и, собственно, настройку самой службы ssh.

О настройке сетевого экранирования сказано ниже, о базовой настройке сервера – выше при описании списков доступа и статических arp-записей.

Касательно настройки самой службы следует выделить два направления: защита от несанкционированного доступа и обеспечение безотказного функционирования.

Настройка

Основной конфигурационный файл службы **/etc/ssh/sshd_config** следует привести к виду:

```
Port 22
Protocol 2
PermitRootLogin no
AllowUsers maintainer
PasswordAuthentication yes
PubkeyAuthentication no
KerberosAuthentication no
HostbasedAuthentication no
IgnoreRhosts yes
PermitEmptyPasswords no
X11Forwarding no
AddressFamily inet
```

Тем самым обеспечивается следующий порядок работы сервера ssh:

Port 22 – сервер принимает сообщения на 22 порту по протоколу tcp.

Protocol 2 – используется протокол ssh версии 2.

PermitRootLogin no – удалённый вход в систему от имени суперпользователя запрещён.

AllowUsers maintainer – удалённый вход разрешён только для пользователя maintainer.

PasswordAuthentication yes – разрешена аутентификация только по паролю, остальные методы запрещены, т. к. не используются.

PermitEmptyPasswords no – запрещено использование пустых паролей.

X11Forwarding no – запрещена передача трафика по протоколу X11, т. к. он не используется сервером.

AddressFamily inet – разрешено использование протокола IPv4, использование IPv6 отключено.

После приведения этого файла к заданному виду следует перезапустить службу ssh, затем убедиться, что она успешно запущена и принимает соединения только по протоколу IPv4. Это можно сделать следующими командами:

```
root@debian:/# systemctl restart sshd
root@debian:/# systemctl status sshd
root@debian:/# ss -Htulp | grep ssh
```

Защита от падений (автоматическое возобновление)

Управлением службами в системе занимается менеджер загрузки и служб systemd, именно он и обеспечивает (при должной настройке) автоматическое возобновление служб в случае их падения. В рассматриваемой системе такая настройка выполнена изначально сборщиками дистрибутива операционной системы.

Проверка работоспособности и устойчивости

Для проверки состояния службы можно воспользоваться командой

```
root@debian:/# systemctl status ssh
```

или командой

```
root@debian:/# service ssh status
```

которая является, по сути, обёрткой вышеприведённой. На рассматриваемой системе в силу значения переменной окружения PATH последняя команда должна выполняться в следующей форме (если не дополнять значение PATH):

```
root@debian:/# /sbin/service ssh status
```

Чтобы проверить, какие порты, адреса и интерфейсы прослушивает ssh, можно воспользоваться командой

```
root@debian:/# ss -Htulp | grep ssh
```

Следующий фрагмент терминального сеанса демонстрирует попытки входа в систему удалённо с разрешённого хоста под разными учётными записями:

```
user@server:~$ ssh maintainer@debian.localdomain
maintainer@debian.localdomain's password:
Last login: Sun Aug 11 11:20:37 2019 from 192.168.7.1

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
maintainer@debian:~$ logout
Connection to debian.localdomain closed.
user@server:~$ ssh root@debian.localdomain
root@debian.localdomain's password:
Permission denied, please try again.
root@debian.localdomain's password:
Permission denied, please try again.
root@debian.localdomain's password:
root@debian.localdomain: Permission denied (password,keyboard-interactive).
user@server:~$
```

Как видно из этого фрагмента, вход от имени maintainer выполнен успешно, а вход от имении root не выполнен даже несмотря на ввод правильного пароля.

При попытке входа с хоста, отличного от разрешённого, при отключённом файерволле рассматриваемого сервера клиент получает сообщение:

```
user@workstation:~$ ssh maintainer@debian.localdomain
ssh_exchange_identification: read: Connection reset by peer
```

что является результатом работы списков контроля доступа. При включённом файерволле клиент получает иное сообщение:

```
user@workstation:~$ ssh maintainer@debian.localdomain
ssh: connect to host debian.localdomain port 22: Connection timed out
```

Однако в обоих случаях в подключении к серверу отказано.

Один из способов проверки автоматического восстановления работоспособности приведён в следующем терминальном сеансе:

```
root@debian:/# date
Вс авг 11 12:19:18 MSK 2019
root@debian:/# ps -ef | grep ssh
maintai+   519     1  0 11:20 ?        00:00:01 sshd: maintainer@pts/0
root       852     1  0 12:14 ?        00:00:00 /usr/sbin/sshd -D
root       873   535  0 12:19 pts/0    00:00:00 grep ssh
root@debian:/# kill -9 852
root@debian:/# ps -ef | grep ssh
maintai+   519     1  0 11:20 ?        00:00:01 sshd: maintainer@pts/0
root       875   535  0 12:19 pts/0    00:00:00 grep ssh
root@debian:/# systemctl status sshd
• ssh.service - OpenBSD Secure Shell server
  Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
 Drop-In: /etc/systemd/system/ssh.service.d
          └─local.conf
  Active: activating (auto-restart) (Result: signal) since Sun 2019-08-11 12:19:38
MSK; 22s ago
    Docs: man:sshd(8)
          man:sshd_config(5)
 Process: 851 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
 Process: 852 ExecStart=/usr/sbin/sshd -D $SSHD_OPTS (code=killed, signal=KILL)
Main PID: 852 (code=killed, signal=KILL)
   Tasks: 2 (limit: 1149)
  Memory: 7.8M
  CGroup: /system.slice/ssh.service
          ├─519 sshd: maintainer@pts/0
          └─520 -bash

авг 11 12:19:38 debian systemd[1]: ssh.service: Main process exited, code=killed,
status=9/KILL
авг 11 12:19:38 debian systemd[1]: ssh.service: Failed with result 'signal'.
root@debian:/# date
Вс авг 11 12:20:18 MSK 2019
root@debian:/# systemctl status sshd
• ssh.service - OpenBSD Secure Shell server
  Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
 Drop-In: /etc/systemd/system/ssh.service.d
          └─local.conf
  Active: active (running) since Sun 2019-08-11 12:20:09 MSK; 12s ago
    Docs: man:sshd(8)
          man:sshd_config(5)
 Process: 878 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
Main PID: 879 (sshd)
   Tasks: 3 (limit: 1149)
  Memory: 8.7M
  CGroup: /system.slice/ssh.service
          ├─519 sshd: maintainer@pts/0
          ├─520 -bash
          └─879 /usr/sbin/sshd -D

авг 11 12:20:09 debian systemd[1]: This usually indicates unclean termination of a
previous run, or service implementation deficiencies.
авг 11 12:20:09 debian systemd[1]: ssh.service: Found left-over process 520 (bash) in
control group while starting unit. Ignoring.
авг 11 12:20:09 debian systemd[1]: This usually indicates unclean termination of a
previous run, or service implementation deficiencies.
авг 11 12:20:09 debian systemd[1]: Starting OpenBSD Secure Shell server...
авг 11 12:20:09 debian systemd[1]: ssh.service: Found left-over process 519 (sshd) in
control group while starting unit. Ignoring.
авг 11 12:20:09 debian systemd[1]: This usually indicates unclean termination of a
previous run, or service implementation deficiencies.
авг 11 12:20:09 debian systemd[1]: ssh.service: Found left-over process 520 (bash) in
```

```
control group while starting unit. Ignoring.
авг 11 12:20:09 debian systemd[1]: This usually indicates unclean termination of a
previous run, or service implementation deficiencies.
авг 11 12:20:09 debian sshd[879]: Server listening on 0.0.0.0 port 22.
авг 11 12:20:09 debian systemd[1]: Started OpenBSD Secure Shell server.
root@debian:/#
```

Здесь сначала отображены системные дата и время, затем получен список выполняющихся процессов в системе, отфильтрованный по упоминанию ssh. Как можно видеть из этого списка, в системе работает процесс sshd с pid 852. Следующая команда убивает этот процесс. Далее демонстрируется, что процесс действительно убит и что это привело к падению соответствующей службы. Спустя 40 секунд (дата и время вновь отображены) демонстрируется факт успешного перезапуска службы.

Установка и настройка lighttpd

Установка веб-сервера lighttpd может быть выполнена при помощи команд

```
root@debian:/# apt-get update
root@debian:/# apt-get install lighttpd
```

На рассматриваемой системе от веб-сервера требуется минимальный набор функций: предоставить доступ по протоколу http к нескольким каталогам. Не используются никакие интерпретируемые языки программирования и скрипты вообще, не задействована для управления контентом никакая CMS (во всяком случае связанная с веб-сервером), не используются СУБД.

Веб-сервер lighttpd выбран потому, что при достаточной стабильности и способности реализовать требуемый функционал его ресурсопотребление ниже, чем у более функциональных веб-серверов apache и nginx.

Настройка

При настройке веб-сервера можно пользоваться специфичными для Debian заготовками в каталогах **/etc/lighttpd/conf-available** и **/etc/lighttpd/conf-enabled**. Однако в рассматриваемом случае настройки веб-сервера просты и статичны, что вполне позволяет создать простой конфигурационный файл, не прибегая к механизмам, ориентированным на более сложные и гибкие случаи использования веб-сервера.

Таким образом все параметры работы веб-сервера указаны в единственном конфигурационном файле **/etc/lighttpd/lighttpd.conf**. Часть строк в нём являются заготовками от разработчиков, часть создана вручную для реализации конкретного функционала рассматриваемой задачи. Итоговый конфигурационный файл имеет вид (комментарии из файла исключены):

```
server.modules = (
        "mod_indexfile",
        "mod_access",
        "mod_alias",
        "mod_redirect",
        "mod_compress",
        "mod_dirlisting",
        "mod_staticfile"
)
```

```
server.document-root        = "/var/www"
server.errorlog             = "/var/log/lighttpd/error.log"
server.pid-file             = "/var/run/lighttpd.pid"
server.username             = "www-data"
server.groupname            = "www-data"
server.port                 = 80
server.dir-listing          = "enable"
server.kbytes-per-second    = 4096
server.max-connections      = 128
server.max-fds              = 4096

server.http-parseopts = (
 "header-strict"           => "enable",
 "host-strict"             => "enable",
 "host-normalize"          => "enable",
 "url-normalize-unreserved"=> "enable",
 "url-normalize-required"  => "enable",
 "url-ctrls-reject"        => "enable",
 "url-path-2f-decode"      => "enable",
 "url-path-dotseg-remove"  => "enable",
)

index-file.names               = ( "index.php", "index.html" )
url.access-deny                = ( "~", ".inc" )
static-file.exclude-extensions = ( ".php", ".pl", ".fcgi" )
dir-listing.encoding           = "utf-8"
dir-listing.exclude            = ( "lost\+found" )
compress.cache-dir             = "/var/cache/lighttpd/compress/"
compress.filetype              = ( "application/javascript", "text/css",
"text/html", "text/plain" )

include_shell "/usr/share/lighttpd/create-mime.conf.pl"
```

По настройке веб-сервера уместно сделать следующие замечания:

Сервер принимает соединения на 80 порту (http), не более 128 соединений одновременно, суммарная пропускная способность ограничена 4 Мбайт/с (32 Мбит/с), веб-серверу разрешено открывать одновременно не более 4096 файловых дескрипторов. Также сервер поддерживает отображение содержимого каталогов в кодировке UTF8, за исключением служебного каталога файловой системы lost+found. Работа веб-сервера по протоколу IPv6 отключена удалением следующей строки из конфигурационного файла по умолчанию:

```
include_shell "/usr/share/lighttpd/use-ipv6.pl " + server.port
```

Параметры разбора http-запросов сохранили значения по умолчанию.

Для корректной работы репозиториев и веб-сервера следует также придерживаться определённой структуры каталогов внутри корневой папки веб-сервера **/var/www**:

```
/var/www
├── buster
├── lost+found
├── ppa
└── security
```

При этом из этого каталога удалён за ненадобностью вместе со страницей-заглушкой подкаталог html.

Защита от падений (автоматическое возобновление)

Аналогично ssh автоматический перезапуск веб-сервера в случае его падения настроен разработчиками дистрибутива операционной системы.

Проверка работоспособности и устойчивости

Для проверки состояния веб-сервера можно воспользоваться командой

```
root@debian:/# systemctl status lighttpd
```

или командой

```
root@debian:/# service lighttpd status
```

которая является, по сути, обёрткой вышеприведённой.

Чтобы проверить, какие порты, адреса и интерфейсы прослушивает lighttpd, можно воспользоваться командой

```
root@debian:/# ss -Htulp | grep lighttpd
```

Проверить корректность конфигурационного файла lighttpd можно командой

```
root@debian:/etc/lighttpd# /usr/sbin/lighttpd -tt -f /etc/lighttpd/lighttpd.conf
```

Методика проверки автоматического возобновления полностью аналогична ssh, поэтому здесь приведён только соответствующий фрагмент терминального сеанса:

```
root@debian:/# date
Чт авг 15 12:57:07 MSK 2019
root@debian:/# ps -ef | grep lighttpd
www-data  1404     1  0 12:50 ?        00:00:00 /usr/sbin/lighttpd -D -f
/etc/lighttpd/lighttpd.conf
root      1413   505  0 12:57 pts/0    00:00:00 grep lighttpd
root@debian:/# kill -9 1404
root@debian:/# ps -ef | grep lighttpd
root      1416   505  0 12:57 pts/0    00:00:00 grep lighttpd
root@debian:/# systemctl status lighttpd
• lighttpd.service - Lighttpd Daemon
  Loaded: loaded (/lib/systemd/system/lighttpd.service; enabled; vendor preset:
enabled)
 Drop-In: /etc/systemd/system/lighttpd.service.d
          └─local.conf
  Active: activating (auto-restart) (Result: signal) since Thu 2019-08-15 12:57:35
MSK; 14s ago
 Process: 1401 ExecStartPre=/usr/sbin/lighttpd -tt -f /etc/lighttpd/lighttpd.conf
(code=exited, status=0/SUCCESS)
 Process: 1404 ExecStart=/usr/sbin/lighttpd -D -f /etc/lighttpd/lighttpd.conf
(code=killed, signal=KILL)
Main PID: 1404 (code=killed, signal=KILL)

авг 15 12:57:35 debian systemd[1]: lighttpd.service: Main process exited,
code=killed, status=9/KILL
авг 15 12:57:35 debian systemd[1]: lighttpd.service: Failed with result 'signal'.
root@debian:/# ps -ef | grep lighttpd
www-data  1422     1  0 12:58 ?        00:00:00 /usr/sbin/lighttpd -D -f
/etc/lighttpd/lighttpd.conf
root      1426   505  0 12:58 pts/0    00:00:00 grep lighttpd
root@debian:/# systemctl status lighttpd
```

```
• lighttpd.service - Lighttpd Daemon
  Loaded: loaded (/lib/systemd/system/lighttpd.service; enabled; vendor preset:
enabled)
 Drop-In: /etc/systemd/system/lighttpd.service.d
          └─local.conf
  Active: active (running) since Thu 2019-08-15 12:58:05 MSK; 26s ago
 Process: 1419 ExecStartPre=/usr/sbin/lighttpd -tt -f /etc/lighttpd/lighttpd.conf
(code=exited, status=0/SUCCESS)
Main PID: 1422 (lighttpd)
   Tasks: 1 (limit: 1149)
  Memory: 1.3M
  CGroup: /system.slice/lighttpd.service
          └─1422 /usr/sbin/lighttpd -D -f /etc/lighttpd/lighttpd.conf

авг 15 12:58:05 debian systemd[1]: Starting Lighttpd Daemon...
авг 15 12:58:05 debian systemd[1]: Started Lighttpd Daemon.
root@debian:/#
```

Создание персонального репозитория

Прежде всего потребуется установить пакет reprepro:

```
root@debian:/# apt-get update
root@debian:/# apt-get install reprepro
```

Поскольку управление репозиторием должно осуществляться от имени maintainer, то корневые каталоги репозиториев должны принадлежать именно этому пользователю. Для этого следует выполнить следующие команды:

```
root@debian:/# chown -R maintainer:maintainer /var/www/buster
root@debian:/# chown -R maintainer:maintainer /var/www/security
root@debian:/# chown -R maintainer:maintainer /var/www/ppa
```

Подробнее вопрос прав доступа описан в следующем разделе, однако далее создание файлов и подкаталогов следует выполнять уже от имени maintainer.

Затем следует перейти корневой каталог создаваемого репозитория (**/var/www/ppa**). Внутри него, не обращая внимания на файлы веб-сервера, требуется создать каталог **conf**, перейти в него и создать в нём файл **distributions** со следующим содержимым:

```
Description: repository with custom packages
Origin: debian
Suite: stable
AlsoAcceptFor: testing unstable experimental
Codename: buster
Version: 10.0
Architectures: amd64
Components: main contrib
```

Затем, вернувшись в корневой каталог репозитория, необходимо выполнить в нём команды:

```
maintainer@debian:/var/www/ppa$ reprepro export
maintainer@debian:/var/www/ppa$ reprepro createsymlinks
```

После этого в корневом каталоге репозитория появятся подкаталоги **/db**, **/dists** и **/pool**. Репозиторий создан, теперь требуется создать цифровую подпись репозитория.

В первую очередь создаётся пара ключей GPG для того, чтобы подписать репозиторий (эти действия следует выполнять от имени пользователя maintainer):

```
maintainer@debian:/var/www/ppa$ gpg --gen-key
```

В процессе создания ключей потребуется ответить на несколько вопросов, большинство (о типе ключа, его длине) из них допустимо оставить с ответами по умолчанию (RSA 3072), дату окончания действия ключа можно не указывать (в рамках этой задачи).

Далее следует экспортировать публичную часть ключа в файл:

```
maintainer@debian:/var/www/ppa$ gpg --armor --export > repository_key.asc
```

Затем следует подписать репозиторий. Для этого следует перейти к файлу Release в составе репозитория (в данном случае **/var/www/ppa/dists/buster/Release**) и подписать его полученным ключом:

```
maintainer@debian:/var/www/ppa$ cd /var/www/ppa/dists/buster
maintainer@debian:/var/www/ppa/dists/buster$ gpg --armor -o Release.gpg -sb Release
```

В итоге в том же каталоге появится файл **Release.gpg**. Это и есть цифровая подпись репозитория. При каждом изменении состава репозитория администратор должен заново подписывать файл(ы) Release тем же ключом (в данном случае от имени maintainer, закрытая часть ключа хранится в домашнем каталоге **/home/maintainer/**). Информацию, необходимую для создания и использования ключа следует хранить в безопасном месте, лучше всего на съёмном носителе отдельно от системы. В таком случае даже получив полный контроль над системой, злоумышленник не сможет разместить в персональном репозитории свои пакеты и скомпрометировать клиентов репозитория.

Следует отметить, что подписывать необходимо только основной файл Release каждого репозитория. Состав персонального репозитория на момент написания этого документа:

```
/var/www/ppa
├── [drwxr-xr-x]  conf
│   └── [-rw-r--r--]  distributions
├── [drwxr-xr-x]  db
│   ├── [-rw-r--r--]  checksums.db
│   ├── [-rw-r--r--]  contents.cache.db
│   ├── [-rw-r--r--]  packages.db
│   ├── [-rw-r--r--]  references.db
│   ├── [-rw-r--r--]  release.caches.db
│   └── [-rw-r--r--]  version
├── [drwxr-xr-x]  dists
│   ├── [drwxr-xr-x]  buster
│   │   ├── [drwxr-xr-x]  contrib
│   │   │   └── [drwxr-xr-x]  binary-amd64
│   │   │       ├── [-rw-r--r--]  Packages
│   │   │       ├── [-rw-r--r--]  Packages.gz
│   │   │       └── [-rw-r--r--]  Release
│   │   ├── [drwxr-xr-x]  main
│   │   │   └── [drwxr-xr-x]  binary-amd64
│   │   │       ├── [-rw-r--r--]  Packages
│   │   │       ├── [-rw-r--r--]  Packages.gz
│   │   │       └── [-rw-r--r--]  Release
│   │   ├── [-rw-r--r--]  Release
│   │   └── [-rw-r--r--]  Release.gpg
│   └── [lrwxrwxrwx]  stable -> buster
└── [drwxr-xr-x]  pool
```

```
└── [drwxr-xr-x]  main
    ├── [drwxr-xr-x]  d
    │   └── [drwxr-xr-x]  dirclean
    │       └── [-rw-r--r--]  dirclean_1.0-1_all.deb
    ├── [drwxr-xr-x]  libv
    │   ├── [drwxr-xr-x]  libvcr
    │   │   └── [-rw-r--r--]  libvcr_1.2-1_amd64.deb
    │   └── [drwxr-xr-x]  libvcr-dev
    │       └── [-rw-r--r--]  libvcr-dev_1.2-1_amd64.deb
    ├── [drwxr-xr-x]  q
    │   └── [drwxr-xr-x]  qsquidclassroom
    │       └── [-rw-r--r--]  qsquidclassroom_1.1-1_amd64.deb
    ├── [drwxr-xr-x]  r
    │   └── [drwxr-xr-x]  restore-settings
    │       └── [-rw-r--r--]  restore-settings_1.0-3_all.deb
    └── [drwxr-xr-x]  t
        └── [drwxr-xr-x]  tux-user-avatars
            └── [-rw-r--r--]  tux-user-avatars_1.0-1_all.deb
```

Здесь в репозитории размещены три файла с именем Release, при этом основным считается тот, который находится ближе всего к корню репозитория. Именно он подписан, т. е. рядом с ним находится файл Release.gpg.

Защита содержимого репозитория

В процессе зеркалирования репозитория, т. е. перемещения большого объёма файлов между разными файловыми системами, совсем не обязательно поддерживающими общий набор атрибутов, или при управлении персональным репозиторием права на файлы и каталоги в составе репозиториев могут отличаться от желаемых. Проверка и установка необходимых прав вручную даже для небольшого репозитория чрезвычайно трудоёмка. Целесообразнее разработать и разместить в системе небольшой сценарий:

```
root@debian:~# cat > /usr/local/bin/repattr
#!/bin/bash
#Setting comandline arguments smart names
REPROOT=$1
REPUSER=$2
REPGROUP=$3
#If no repository root given or it is not a directory then exit with errorcode 1
if [ ! -d "$REPROOT" ]
then
 exit 1
fi
#If no user given then assume root
if [ -z "$REPUSER" ]
then
 REPUSER=root
fi
#If no group given then assume root
if [ -z "$REPGROUP" ]
then
 REPGROUP=root
fi
#Changing owner for all files and directories
chown -R $REPUSER:$REPGROUP $REPROOT
#Setting minimal possible permissions to both files and directories
chmod -R 0644 $REPROOT
#Allowing to enter into directories
tree -dfRi $REPROOT | grep "./" | grep -v "\->" | xargs -I dir chmod 0755 dir
```

```
#Exiting
exit 0
^D
root@debian:~# chmod a+x /usr/local/bin/repattr
```

Репозиторий работоспособен при следующих правах доступа:

- владельцем всех файлов и каталогов репозитория является непривилегированный пользователь maintainer;
- для всех каталогов выставлены полные права для владельца и права на чтение и исполнение (открытие) для членов его группы и всех остальных (0755);
- для всех файлов в каталогах выставлены права на чтение и запись для владельца и только на чтение членам его группы и всем остальным (0644).

Запускать этот сценарий, разумеется, следует с правами, достаточными для его работы, т. е. в подавляющем большинстве случаев от имени суперпользователя.

Запуск сценария для правильной настройки прав доступа и владельца maintainer в отношении персонального репозитория должен выглядеть так:

```
root@debian:~# repattr /var/www/ppa/ maintainer maintainer
```

Для репозиториев с большим количеством пакетов работа сценария может занять заметное время. Проверить результаты работы сценария можно командой вида:

```
root@debian:~# tree -Rugp /var/www/ppa/
```

Однако пользоваться ей надо разумно, т. к. для больших репозиториев её вывод также будет велик и займёт определённое время.

О дополнительных службах

Обычно при настройке веб-сервера, коим и является рассматриваемый сервер репозиториев, рассматривается вопрос об установке ряда дополнительных служб, обеспечивающих его работоспособность и стабильность. К таким службам относятся в том числе сервер баз данных, служба точного времени, инструменты противодействия атакам, управления журнальными файлами, резервного копирования и синхронизации. Однако каждая такая служба в том или ином объёме потребляет системные ресурсы. Поэтому, руководствуясь принципом разумной достаточности следует оценить роль каждой такой службы и принять решение о целесообразности её использования.

Сервер баз данных

Исходя из устройства и механизмов работы репозиториев, а также учитывая отсутствие дополнительных веб-ресурсов на рассматриваемом сервере, напрашивается однозначный вывод о том, что СУБД на данном сервере не используется, т. е. её установка не требуется.

Защита от подбора паролей (fail2ban)

Поскольку в силу характера функционирования сервера и отсутствия авторизации в системе через веб-сервер единственным механизмом входа в систему остаётся ssh, то от

использования fail2ban вполне безопасно отказаться. Система хорошо защищена файерволлом (описано далее) с указанием (как по ip, так и по mac-адресу) для него единственного хоста, с которого разрешен вход по ssh. Сам ssh-сервер системы имеет такие же ограничения, кроме того предприняты меры по защите от arp-атак (статическая запись), сам удалённый вход разрешен только для непривилегированного пользователя, учётная запись которого защищена надёжным паролем, а повышение привилегий требует знания не менее надёжного пароля суперпользователя. При этом тот единственный хост, с которого разрешён удалённый вход, является одновременно и сервером виртуализации, на котором и размещён рассматриваемый виртуальный сервер. В таких условиях атака на сервер путём попыток подбора паролей и учётных записей возможна только если под контроль злоумышленника попал сам сервер виртуализации. Но в этом случае получить полный контроль над виртуальным сервером возможно и без входа на него. В силу вышесказанного использование fail2ban избыточно и ведёт лишь к растрате системных ресурсов.

Ротация журналов (logrotate)

Системные журналы могут служить средством атаки на сервер с целью заполнить его дисковое пространство и парализовать, тем самым, его работу. Этому противостоит система ротации журнальных файлов.

При заданных настройках lighttpd ведёт только журнал ошибок **/var/log/lighttpd/error.log**. В журнал ошибок также вносятся сведения о запуске и остановке веб-сервера. Этот журнал допустимо не рассматривать как цель атаки, т. к. если злоумышленник смог обеспечить лавинообразный рост этого журнала, то у него уже достаточно полномочий в системе, и атака на журналы ему уже не нужна. Журнал доступа, напротив, ведётся подробно и может быть целью такой атаки. При этом в штатном режиме работы записи этого журнала, учитывая роль и место сервера в сети, не представляют особой ценности. Поэтому в рассматриваемом случае журнал доступа не ведётся.

Аналогичные рассуждения справедливы и для ssh. При входе в систему по этому протоколу соответствующая запись вносится в файл **/var/log/auth.log**. Учитывая предпринятые меры защиты ssh, можно утверждать, что атака возможна только с севера виртуализации, на котором расположен сам виртуальный сервер репозиториев, но это бессмысленно, т. к. контролируя хост-систему злоумышленник имеет все возможности для получения данных гостя без взлома самого гостя (разумеется, если не используется шифрование виртуального диска, в рассматриваемом случае шифрование лишено смысла). Остановка работы гостя тем более не вызовет проблем у атакующего.

Как итог следует отметить, что настройки системы по умолчанию (ротация еженедельно) здесь вполне применимы.

Служба точного времени (ntp)

Служба точного времени также избыточна для этого сервера: так как сервер виртуальный и к нему не подключено ни одного физического источника точного времени, то он использует системные часы сервера виртуализации, на котором расположен. Такое решение возможно считать надёжным, т. к. скомпрометированный сервер виртуализации не оставляет шансов виртуальным машинам, расположенным на нём.

Почтовая служба

Почтовая служба, даже локальная, может стать направлением для атаки: так, например, воспользовавшись уязвимостями веб-сервера, злоумышленник может генерировать огромный вал писем локально в системе, чем провоцировать переполнение файловой системы и отказ обслуживания. Поэтому даже локальную почтовую службу следует тщательно настраивать или отключать вовсе, если она не используется. В рассматриваемой системе почтовые компоненты не были выбраны на этапе установки. Также они не являются зависимостями ни одной из установленных служб.

Касательно вспомогательных задач электронной почты таких как информирование системного администратора о сбоях сервисов или аппаратуры, можно утверждать что этот функционал на рассматриваемом сервере избыточен. Так как сервер виртуальный, то сбои оборудования на нём – это или результат нестабильной работы гипервизора, или следствие сбоев физического оборудования сервера виртуализации. В любом случае именно он будет информировать системного администратора, а не виртуальный сервер. В то же время службы виртуального сервера настроены на автоматическое возобновление работы при сбоях и не требуют ручного вмешательства.

Установка и настройка сетевого фильтра (nftables)

По умолчанию в системе был установлен интерфейс iptables сетевого фильтра ядра netfilter. Для его удаления и установки пришедшего ему на смену интерфейса nftables следует выполнить следующие команды:

```
root@debian:/# apt-get purge iptables
root@debian:/# apt-get install nftables
```

Следующим этапом следует определиться с методикой формирования набора правил сетевого фильтра. Здесь возможны два подхода: настроить все правила одновременно на этапе загрузки системы до запуска сетевых интерфейсов и запуск на том же этапе минимального, каркасного набора правил с последующим добавлением правил перед запуском каждого интерфейса и удалением при его остановке. Второй подход более гибок и удобен в системах со сложными динамически изменяющимися схемами маршрутизации, резервированием каналов и т. д. На рассматриваемой системе таких особенностей нет, поэтому атомарная загрузка всех правил единовременно – это наиболее взвешенное решение.

Защита службы nftables.service менеджера systemd от непредвиденных падений службы избыточна: эта служба выполняет операцию загрузки правил файерволла и завершается сразу по её окончании, что можно видеть из вывода следующей команды:

```
root@debian:/# systemctl status nftables
• nftables.service - nftables
  Loaded: loaded (/lib/systemd/system/nftables.service; enabled; vendor preset:
enabled)
  Active: active (exited) since Thu 2019-08-15 10:18:21 MSK; 10h ago
    Docs: man:nft(8)
          http://wiki.nftables.org
Main PID: 185 (code=exited, status=0/SUCCESS)
   Tasks: 0 (limit: 1149)
  Memory: 0B
  CGroup: /system.slice/nftables.service
```

```
авг 15 10:18:21 localhost systemd[1]: Started nftables.
Warning: Journal has been rotated since unit was started. Log output is incomplete or
unavailable.
root@debian:/#
```

Открытые порты

Сетевой фильтр не должен блокировать порты, используемые компонентами сервера. Получить перечень таких портов можно командами

```
root@debian:/# ss -lf inet
Netid     State      Recv-Q     Send-Q    Local Address:Port    Peer Address:Port
udp       UNCONN     0          0               0.0.0.0:bootpc       0.0.0.0:*
tcp       LISTEN     0          128             0.0.0.0:ssh          0.0.0.0:*
tcp       LISTEN     0          128             0.0.0.0:http         0.0.0.0:*
root@debian:/# ss -lf inet6
Netid     State      Recv-Q     Send-Q    Local Address:Port    Peer Address:Port
root@debian:/#
```

Из вывода этих команд следует, что предоставляемые системой сервисы прослушивают (LISTEN) только IPv4 порты ssh (22) и http (80). Открытые IPv6 порты отсутствуют.

Сетевое окружение

Принципиальная схема расположения виртуального сервера в сети приведена на рисунке:

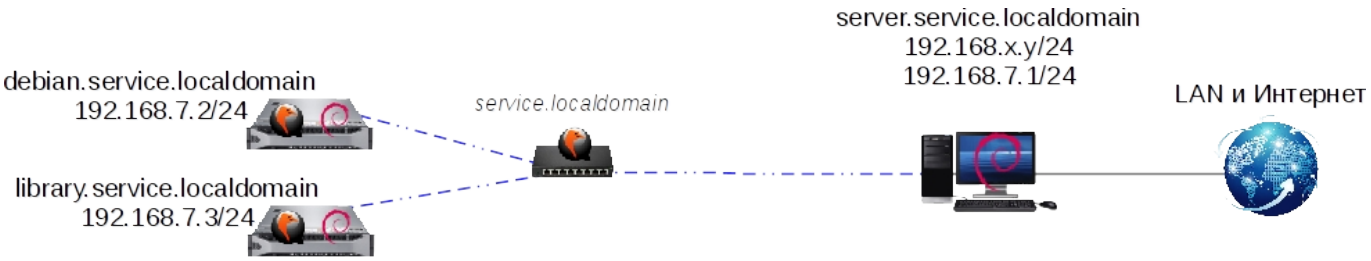

Рисунок 1: Принципиальная схема сети

Из этой схемы следует, что сервер репозиториев подключён к виртуальной сети service локального домена, к которой подключены и другие (например library) виртуальные сервера. Связь с внешними сетями (LAN и Интернет) возможна только через сервер виртуализации. При этом использование репозитория возможно любым представленным на схеме узлом: любым компьютером из LAN (или даже Интернета, если LAN позволяет), самим сервером виртуализации, любым другим виртуальным сервером. В то же самое время подключения по ssh допускаются только с сервера виртуализации. Среди прочего следует предусмотреть способность сервера откликаться на некоторые ICMP-запросы, что весьма удобно для быстрой проверки доступности сервера. При этом количество таких запросов в единицу времени следует ограничить, чтобы они не могли стать орудием в руках злоумышленника для генерирования бесполезного трафика. Также следует ограничить количество запросов на подключение и по другим протоколам. Оптимальные ограничения могут быть выявлены только в процессе строевой эксплуатации сервера, однако на начальном этапе представляются вполне обоснованными следующие значения:

- не более одного запроса в секунду на подключение по ssh

- не более десяти icmp-запросов в секунду
- не более ста запросов в секунду на подключение по http

Кроме того, полностью средствами файерволла отсекаются все IPv6 пакеты, получаемые сервером или генерируемые им.

Структура сетевого фильтра

Суммируя всё вышесказанное, составлен следующий сценарий атомарной загрузки правил файерволла **/etc/nftables.conf**:

```
#!/usr/sbin/nft -f

flush ruleset

table ip filter {
       chain input {
               type filter hook input priority 0;
               policy drop;
               # accept any localhost traffic
               iif lo accept
               # accept traffic originating from us
               ct state established,related accept
               # accept traffic from subnet server to local ssh port
               ip saddr 192.168.7.1 ether saddr 52:54:00:2b:1e:a0 tcp dport ssh ct
state new limit rate 1/second accept
               # accept icmp traffic from everywhere
               icmp type echo-request limit rate 10/second accept
               # accept traffic to http port from everywhere
               tcp dport http ct state new limit rate 100/second accept
       }
       chain output {
               type filter hook output priority 0;
               policy accept;
       }
}

table ip6 filter {
       chain input {
               type filter hook input priority 0;
               policy drop;
       }
       chain output {
               type filter hook output priority 0;
               policy drop;
       }
}
```

Запуск и проверка сетевого фильтра

Автоматическая загрузка правил файерволла обеспечена установщиком пакета nftables и не требует ручного вмешательства. Проверить текущий набор правил можно командой

```
root@debian:/# /usr/sbin/nft list ruleset
```

Вывод которой должен с точностью до комментариев и команды предварительной очистки совпадать с приведённым выше сценарием загрузки правил файерволла.

Убедиться в том, что правила были загружены до запуска сетевых интерфейсов, можно просмотрев содержимое файла **/var/log/daemon.log**. Как пример приведена следующая выдержка из этого файла (с сокращениями):

```
Aug 16 13:00:46 localhost systemd[1]: Started Apply Kernel Variables.
…
Aug 16 13:00:46 localhost systemd[1]: Starting udev Kernel Device Manager...
Aug 16 13:00:46 localhost systemd[1]: Started udev Kernel Device Manager.
Aug 16 13:00:46 localhost systemd[1]: Started nftables.
Aug 16 13:00:46 localhost systemd[1]: Reached target Network (Pre).
Aug 16 13:00:46 localhost systemd-udevd[221]: Using default interface naming scheme
'v240'.
Aug 16 13:00:46 localhost systemd-udevd[221]: link_config: autonegotiation is unset
or enabled, the speed and duplex are not writable.
…
Aug 16 13:00:46 localhost systemd-udevd[220]: link_config: autonegotiation is unset
or enabled, the speed and duplex are not writable.
…
Aug 16 13:00:46 localhost systemd[1]: Reached target Basic System.
…
Aug 16 13:00:46 localhost systemd[1]: Starting Raise network interfaces...
…
Aug 16 13:00:51 localhost ifup[323]: bound to 192.168.7.2 -- renewal in 2147483648
seconds.
Aug 16 13:00:51 localhost dhclient[344]: bound to 192.168.7.2 -- renewal in
2147483648 seconds.
Aug 16 13:00:51 localhost systemd[1]: Started Raise network interfaces.
Aug 16 13:00:51 localhost systemd[1]: Reached target Network.
```

Проверку сетевого фильтра можно выполнить по следующей программе испытаний:

№	Действие	Ожидаемый результат
1	Вход по ssh с рабочей станции в локальной сети	Вход невозможен
2	Вход по ssh с другой виртуальной машины в той же виртуальной сети	Вход невозможен
3	Получение файла по протоколу http с рабочей станции в локальной сети	Файл получен
4	Получение файла по протоколу http с другой виртуальной машины в той же виртуальной сети	Файл получен
5	Сканирование открытых портов с рабочей станции в локальной сети	Открыт порт 80
6	Сканирование открытых портов с другой виртуальной машины в той же виртуальной сети	Открыт порт 80

Следующий фрагмент терминального сеанса демонстрирует проверку сервера со стороны рабочей станции:

```
root@workstation:/# ssh maintainer@debian.service.localdomain
ssh: connect to host debian.service.localdomain port 22: Connection timed out
root@workstation:/# wget http://debian.service.localdomain/ppa/dists/buster/Release
--2019-08-19 20:29:30--  http://debian.service.localdomain/ppa/dists/buster/Release
Распознаётся debian.service.localdomain (debian.service.localdomain)… 192.168.7.2
Подключение к debian.service.localdomain (debian.service.localdomain)|
192.168.7.2|:80... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: 1655 (1,6K) [application/octet-stream]
Сохранение в: «Release»
```

```
Release               100%[=============================>]  1,62K  --.-KB/s  за 0s

2019-08-19 20:29:30 (227 MB/s) - «Release» сохранён [1655/1655]

root@workstation:/# rm Release
root@workstation:/# nmap -sS 192.168.7.2
Starting Nmap 6.47 ( http://nmap.org ) at 2019-08-19 20:30 GMT-3

Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (0.00052s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 6.14 seconds
root@workstation:/# nmap -sT 192.168.7.2

Starting Nmap 6.47 ( http://nmap.org ) at 2019-08-19 20:31 GMT-3
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (0.00072s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 5.66 seconds
root@workstation:/# nmap -sU 192.168.7.2

Starting Nmap 6.47 ( http://nmap.org ) at 2019-08-19 20:31 GMT-3
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (0.00053s latency).
All 1000 scanned ports on debian.localdomain (192.168.7.2) are open|filtered

Nmap done: 1 IP address (1 host up) scanned in 22.35 seconds
root@workstation:/# nmap -sN 192.168.7.2

Starting Nmap 6.47 ( http://nmap.org ) at 2019-08-19 20:32 GMT-3
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (0.00053s latency).
All 1000 scanned ports on debian.localdomain (192.168.7.2) are open|filtered

Nmap done: 1 IP address (1 host up) scanned in 22.34 seconds
root@workstation:/# nmap -sF 192.168.7.2

Starting Nmap 6.47 ( http://nmap.org ) at 2019-08-19 20:32 GMT-3
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (0.00051s latency).
All 1000 scanned ports on debian.localdomain (192.168.7.2) are open|filtered

Nmap done: 1 IP address (1 host up) scanned in 22.33 seconds
root@workstation:/# nmap -sX 192.168.7.2

Starting Nmap 6.47 ( http://nmap.org ) at 2019-08-19 20:33 GMT-3
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (0.00059s latency).
All 1000 scanned ports on debian.localdomain (192.168.7.2) are open|filtered

Nmap done: 1 IP address (1 host up) scanned in 22.31 seconds
root@workstation:/# nmap -sA 192.168.7.2

Starting Nmap 6.47 ( http://nmap.org ) at 2019-08-19 20:33 GMT-3
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (0.00051s latency).
```

```
Not shown: 999 filtered ports
PORT   STATE      SERVICE
80/tcp unfiltered http

Nmap done: 1 IP address (1 host up) scanned in 6.08 seconds
root@workstation:/# nmap -sW 192.168.7.2

Starting Nmap 6.47 ( http://nmap.org ) at 2019-08-19 20:34 GMT-3
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (0.00053s latency).
Not shown: 999 filtered ports
PORT   STATE  SERVICE
80/tcp closed http

Nmap done: 1 IP address (1 host up) scanned in 19.79 seconds
root@workstation:/# nmap -sM 192.168.7.2

Starting Nmap 6.47 ( http://nmap.org ) at 2019-08-19 20:34 GMT-3
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (0.00058s latency).
All 1000 scanned ports on debian.localdomain (192.168.7.2) are open|filtered

Nmap done: 1 IP address (1 host up) scanned in 22.30 seconds
root@workstation:/# nmap -sO 192.168.7.2

Starting Nmap 6.47 ( http://nmap.org ) at 2019-08-19 20:35 GMT-3
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (0.022s latency).
Not shown: 254 open|filtered protocols
PROTOCOL STATE SERVICE
1        open  icmp
6        open  tcp

Nmap done: 1 IP address (1 host up) scanned in 9.71 seconds
root@workstation:/# nmap -sV -sS 192.168.7.2

Starting Nmap 6.47 ( http://nmap.org ) at 2019-08-19 20:35 GMT-3
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (0.00053s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE VERSION
80/tcp open  http    lighttpd 1.4.53

Service detection performed. Please report any incorrect results at
http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 24.21 seconds
root@workstation:/#
```

Подводя итог проверки следует отметить, что удалось выяснить наличие открытого порта http, на котором работает веб-сервер lighttpd версии 1.4.53, успешно предоставивший файл по запросу, также подтверждена отзывчивость сервера на icmp-запросы. Таким образом полученные результаты не противоречат ожидаемым и проверку можно считать успешно пройденной.

Следующий фрагмент терминального сеанса демонстрирует проверку сервера со стороны другой виртуальной машины:

```
root@library:/# ssh maintainer@debian.service.localdomain
ssh: connect to host debian.service.localdomain port 22: Connection timed out
root@library:/# wget http://debian.service.localdomain/ppa/dists/buster/Release
```

```
--2019-08-19 21:03:27--  http://debian.service.localdomain/ppa/dists/buster/Release
Распознаётся debian.service.localdomain (debian.service.localdomain)… 192.168.7.2
Подключение к debian.service.localdomain (debian.service.localdomain)|
192.168.7.2|:80... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа… 200 OK
Длина: 1655 (1,6K) [application/octet-stream]
Сохранение в: «Release»

Release               100%[============================>]  1,62K  --.-KB/s   за 0s

2019-08-19 21:03:27 (271 MB/s) - «Release» сохранён [1655/1655]

root@library:/# rm Release
root@library:/# nmap -sS 192.168.7.2
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-19 21:04 MSK
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (-0.078s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 52:54:00:79:69:B1 (QEMU virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 9.35 seconds
root@library:/# nmap -sT 192.168.7.2
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-19 21:04 MSK
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (-0.13s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 52:54:00:79:69:B1 (QEMU virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 4.99 seconds
root@library:/# nmap -sU 192.168.7.2
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-19 21:04 MSK
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (-0.20s latency).
All 1000 scanned ports on debian.localdomain (192.168.7.2) are open|filtered
MAC Address: 52:54:00:79:69:B1 (QEMU virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 21.35 seconds
root@library:/# nmap -sN 192.168.7.2
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-19 21:05 MSK
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (-0.20s latency).
All 1000 scanned ports on debian.localdomain (192.168.7.2) are open|filtered
MAC Address: 52:54:00:79:69:B1 (QEMU virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 21.35 seconds
root@library:/# nmap -sF 192.168.7.2
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-19 21:06 MSK
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (-0.20s latency).
All 1000 scanned ports on debian.localdomain (192.168.7.2) are open|filtered
MAC Address: 52:54:00:79:69:B1 (QEMU virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 21.35 seconds
root@library:/# nmap -sX 192.168.7.2
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-19 21:06 MSK
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (-0.19s latency).
All 1000 scanned ports on debian.localdomain (192.168.7.2) are open|filtered
```

```
MAC Address: 52:54:00:79:69:B1 (QEMU virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 21.35 seconds
root@library:/# nmap -sA 192.168.7.2
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-19 21:07 MSK
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (-0.060s latency).
Not shown: 999 filtered ports
PORT   STATE      SERVICE
80/tcp unfiltered http
MAC Address: 52:54:00:79:69:B1 (QEMU virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 10.68 seconds
root@library:/# nmap -sW 192.168.7.2
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-19 21:07 MSK
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (-0.078s latency).
Not shown: 999 filtered ports
PORT   STATE  SERVICE
80/tcp closed http
MAC Address: 52:54:00:79:69:B1 (QEMU virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 9.35 seconds
root@library:/# nmap -sM 192.168.7.2
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-19 21:07 MSK
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (-0.20s latency).
All 1000 scanned ports on debian.localdomain (192.168.7.2) are open|filtered
MAC Address: 52:54:00:79:69:B1 (QEMU virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 21.34 seconds
root@library:/# nmap -sO 192.168.7.2
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-19 21:08 MSK
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (-0.12s latency).
Not shown: 254 open|filtered protocols
PROTOCOL STATE SERVICE
1        open  icmp
6        open  tcp
MAC Address: 52:54:00:79:69:B1 (QEMU virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 6.29 seconds
root@library:/# nmap -sV -sS 192.168.7.2
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-19 21:08 MSK
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (-0.060s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE VERSION
80/tcp open  http    lighttpd 1.4.53
MAC Address: 52:54:00:79:69:B1 (QEMU virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap.org/
submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.14 seconds
root@library:/#
```

Подводя итог и этой проверки следует отметить, что также удалось выяснить наличие открытого порта http, на котором работает веб-сервер lighttpd версии 1.4.53, успешно предоставивший файл по запросу, и подтвердить отзывчивость сервера на icmp-запросы. Кроме того, удалось выяснить MAC-адрес сервера, что не удивительно, т. к. они находятся

в одной локальной сети. Таким образом полученные результаты не противоречат ожидаемым и проверку можно считать успешно пройденной.

Завершение установки

В завершение установки имеет смысл очистить локальный кэш пакетов и удалить пакеты, которые системе стали не нужны. Это можно сделать командами:

```
root@debian:/# apt-get autoremove
root@debian:/# apt-get clean
```

Использование виртуального сервера

Далее приведены краткие инструкции по использованию виртуального сервера и сопряжёнными с этим работами.

Порядок смены сервера виртуализации

Здесь не рассматривается механизм миграции виртуальных машин, здесь рассматривается только настройка виртуального сервера после миграции.

В случае, если новый хост-сервер имеет другой адрес виртуального интерфейса, для подключения по ssh предусматривается другой набор разрешенных хостов, вообще отличаются настройки сети (например не используется DHCP), то следует изменить следующие конфигурационные файлы и параметры в них:

- **/etc/hosts.allow** – списки доступа к службам;
- **/etc/network/interfaces** – сетевые интерфейсов, адреса и статические arp-записи;
- **/etc/nftables.conf** – настройки сетевого фильтра, имена сетевых интерфейсов, ip и lladdr адреса.

Также следует непременно отредактировать или затереть конфигурационные файлы со служебной информацией протокола ssh в домашних каталогах пользователей:

- **/root/.ssh/known_hosts**
- **/home/maintainer/.ssh/known_hosts**

Эти файлы могут являться источниками ценной информации о сети, в которой прежде находился виртуальный сервер, попадание этой информации в посторонние руки является крайне нежелательным.

Порядок входа в систему

Для удалённого входа в систему следует использовать протокол ssh и подключаться только с сервера виртуализации, на котором и размещён рассматриваемый сервер. При необходимости подключиться с иной машины следует с неё подключаться к серверу виртуализации, а затем уже с него – к целевому серверу.

Подключение разрешено только пользователю maintainer. При необходимости повышения прав следует сначала войти в систему под этим пользователем, а потом повысить права до суперпользовательских средствами утилиты su.

Команда для подключения к серверу может имеет вид (с точностью до текущего пользователя на сервере виртуализации и/или способа указания целевого сервера):

```
user@server:/# ssh maintainer@debian.service.quail
user@server:/# ssh maintainer@debian.quail
user@server:/# ssh maintainer@192.168.7.2
```

Порядок синхронизации с официальными репозиториями

Для синхронизации с официальными репозиториями следует использовать методы, принятые в сообществе Debian. Так вместо прямого использования rsync следует применять специализированное программное обеспечение, которое позволяет минимизировать нагрузку на оба хоста (локальный и официальный репозитории) и трафик в сети. Для локальных зеркал уровня организации/предприятия одим из рекомендованных является пакет debmirror. В своей работе он использует в том числе и rsync.

Для синхронизации основного репозитория и репозитория обновлений безопасности достаточно выполнить команды:

```
maintainer@debian:/$ cd /var/www/buster
maintainer@debian:/$ debmirror --nosource --i18n --host=mirror.mephi.ru \
--root=debian --method=http --progress --ignore-release-gpg \
--dist=buster --arch=amd64 --section=main,contrib,non-free ./
maintainer@debian:/$ cd /var/www/security
maintainer@debian:/$ debmirror --nosource --i18n --host=security.debian.org \
--root=debian-security --method=http --progress --ignore-release-gpg \
--dist=buster/updates --arch=amd64 --section=main,contrib ./
root@debian:~# repattr /var/www/buster/ maintainer maintainer
root@debian:~# repattr /var/www/security/ maintainer maintainer
```

После завершения синхронизации не лишним будет выполнить сценарий repattr, описанный выше:

```
maintainer@debian:/$ repattr /var/www/security maintainer maintainer
maintainer@debian:/$ repattr /var/www/buster maintainer maintainer
```

Поскольку зеркалирование выполнялось от имени maintainer, то другого владельца у файлов быть не может, тогда как аттрибуты могут и отличаться.

Порядок управления персональным репозиторием

Для добавления пакета в репозиторий требуется выполнить команду вида:

reprepro [-C <категория>] includedeb <codename> <пакет>

где <категория> – это одно из значений components,
<codename> – кодовое имя дистрибутива,
<пакет> – имя файла пакета (в том числе и с указанием пути).

Пример команды:

```
maintainer@debian:/$ reprepro -C main includedeb buster dirclean_1.0-1_all.deb
```

Если необходимо добавить сразу несколько пакетов, то уместна команда:

```
maintainer@debian:/$ reprepro -C main includedeb buster *.deb
```

Т. е. поддерживается использование шаблонов имён файлов. Так, например, чтобы добавить все пакеты apache2, расположенные в каталоге /tmp/raw_deb/ , можно воспользоваться командой:

```
maintainer@debian:/$ reprepro includedeb buster /tmp/raw_deb/apache2*.deb
```

При этом, если не указывать категорию, то система автоматически определит её по содержимому пакета (файл control) для каждого пакета.

Удалить пакет из репозитория можно командой вида:

reprepro -C <категория> remove <codename> <пакет>

После завершения манипуляций с пакетами следует заново подписать основные файлы Release репозитория (не путать с подчинёнными файлами Release в секциях репозитория). В рассматриваемой системе этот файл единственный (**/var/www/ppa/dists/buster/Release**), соответственно команды принимают вид:

```
maintainer@debian:/$ cd /var/www/ppa/dists/buster
maintainer@debian:/var/www/ppa/dists/buster$ gpg --armor -o Release.gpg -sb Release
```

Порядок подключения клиента к репозиториям

Для подключения компьютера к репозиториям следует во-первых в его файл **/etc/apt/sources.list** внести некоторые или все из следующих строк (с точностью до имени домена, здесь localdomain):

```
deb http://debian.service.localdomain/buster/ buster main contrib non-free
deb http://debian.service.localdomain/security/ buster/updates main contrib
deb http://debian.service.localdomain/ppa/ buster main contrib
```

Затем на клиентском компьютере требуется зарегистрировать публичную часть ключа (файл **repository_key.asc**, полученный при создании персонального репозитория, следует доставить на клиентский компьютер любым безопасным способом):

```
root@client:/# apt-key add repository_key.asc
```

Проверить успешность добавления ключа (получить список всех зарегистрированных ключей) можно командой:

```
root@client:/# apt-key list
```

В завершение следует на клиенте обновить сведения о пакетах и убедиться, что репозитории обнаружены, доступны, их цифровые подписи приняты:

```
root@client:/# apt-get update
```