Роутер на Linux

На этом уроке рассмотрим превращение сервера на Linux в роутер на основе пакета frr. Защитим его, используя встроенный в Centos7 firewalld.

Оглавление

Маршрутизация на хосте (Routing on a Host, RoH)

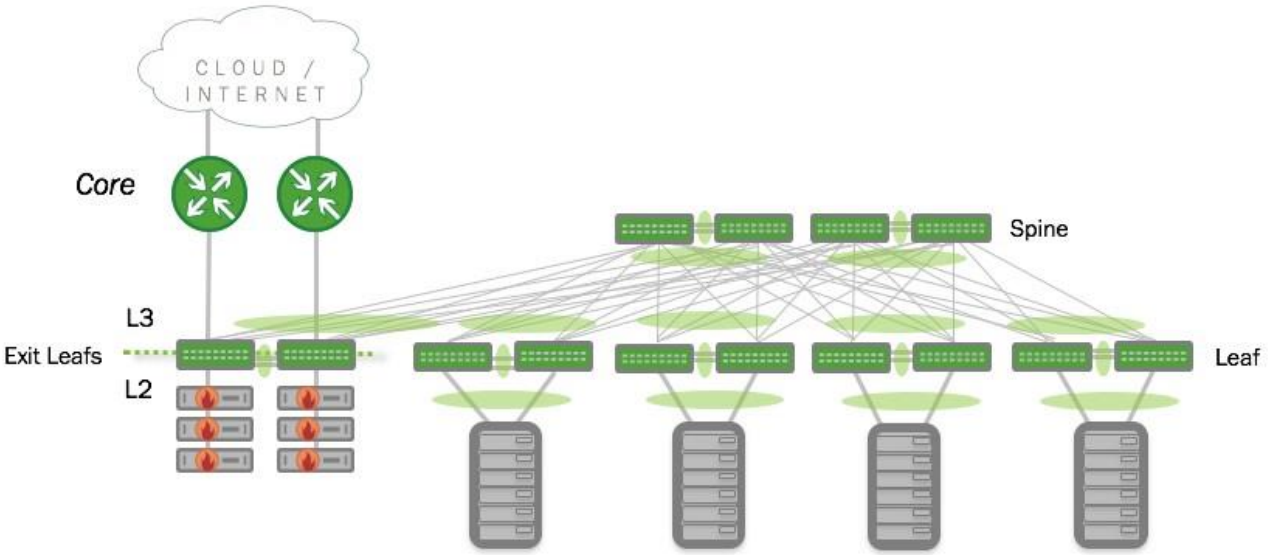
Чтобы строить эффективные по цене и отказоустойчивости сети/центры обработки данных, многие администраторы/инженеры/архитекторы используют экосистему Linux для запуска протоколов маршрутизации непосредственно на своих серверах. Такой вид дизайна часто называют маршрутизацией на хосте (RoH). Это означает запуск протоколов, обеспечивающих маршрутизацию третьего уровня модели оси, таких как OSPF (Open Shortest Path First) или BGP (Border Gateway Protocol), непосредственно на уровне хоста. Эта задача может быть выполнена различными способами, например, запуском демона маршрутизации (quagga, bird, frr, gobgp):

1. В виде контейнера.
2. Вместе с виртуальной машиной, запущенной на гипервизоре.
3. Непосредственно на гипервизоре.
4. На физическом сервере (технически это то же самое, что и на гипервизоре, но без виртуальных машин).

Почему следует так делать

Такой вариант сетевого дизайна позволяет строить линейно-масштабируемые сети на десятки тысяч физических серверов и сотни тысяч виртуальных машин/контейнеров, а также упрощает поиск неисправностей. Следует отметить, что в примере указаны L3-коммутаторы, которые могут осуществлять коммутацию на L2 и маршрутизацию на L3.

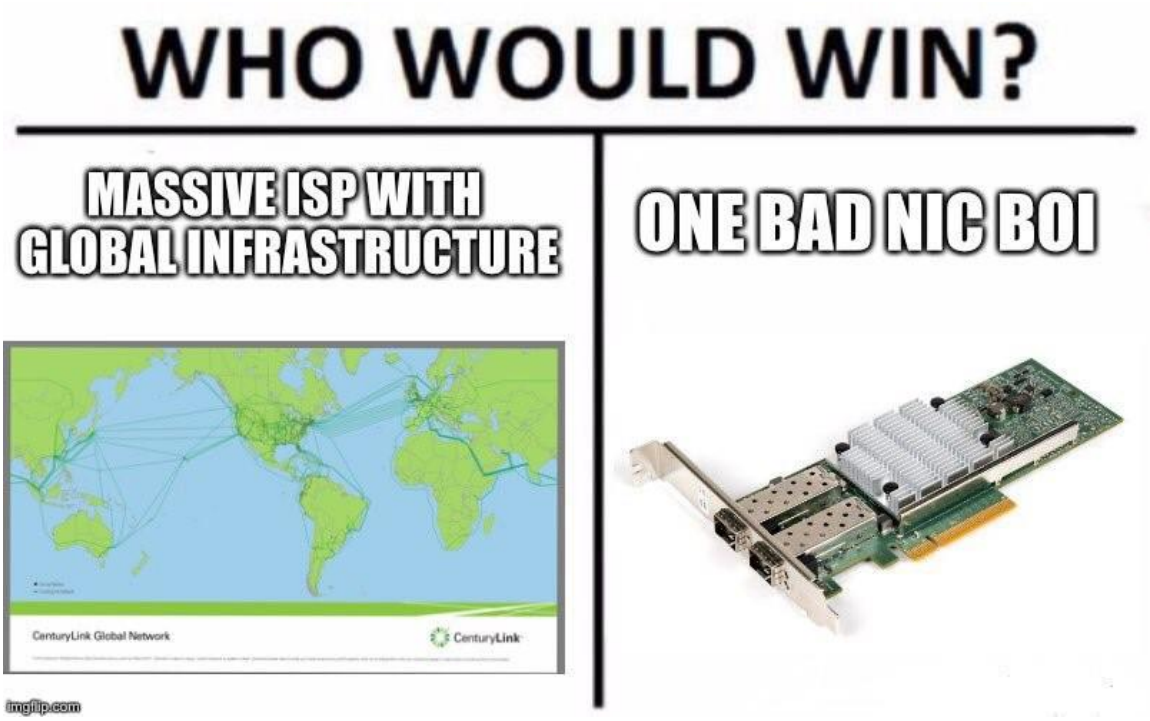
Типичный дизайн сети предприятия — сервер, который подключен двумя кабелями к двум разным коммутаторам доступа, и все эти коммутаторы объединены в l2-домен при помощи VLAN. То есть между набором из двух коммутаторов доступа всегда есть растянутый VLAN.



Кажущаяся удобной на первый взгляд, схема накладывает огромное количество ограничений как на коммутаторы, так и на подключенные к ним серверы. Например, вы вынуждены использовать протокол spanning tree для предотвращения закольцовок. Это ведет к тому, что все избыточные линки между устройствами будут заблокированы, и не получится использовать все возможности сети.

При добавлении нового VLAN или сервера администратор должен вручную настроить порт устройства и назначить ему соответствующий VLAN. Но основная проблема заключается в том, что L2-домен — единый домен ошибки (failure domain). То есть, проблема, например, с сетевым адаптером сервера, может привести к тому, что все сетевые устройства, участвующие в этом L2-домене (как другие серверы, так и коммутаторы/роутеры), будут недоступны. А зачастую это вся сеть.

В декабре 2018 года глобальный провайдер CenturyLink был недоступен более 24 часов — нельзя было дозвониться даже на номер 911 в США — из-за одного сетевого адаптера.



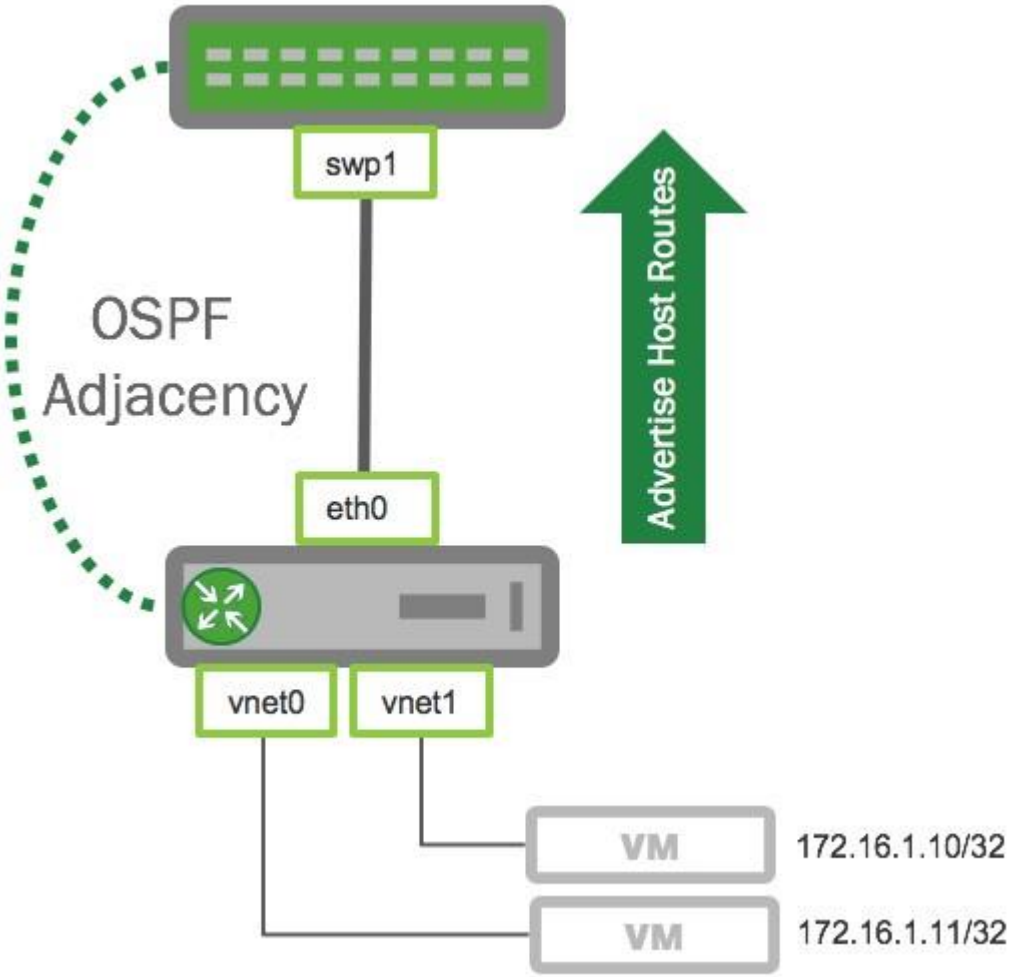
Получается, что к плохому масштабированию L2-доменов добавляется еще и проблема достаточно сложного поиска неисправностей внутри них:

- Трассировка неэффективна, так как она показывает только L3-устройства в сети, а конструкция с растянутым VLAN использует устройства второго уровня модели OSI. То есть нет способа определить, какой из коммутаторов и куда отправляет трафик.
- Таблицы MAC-адресов становятся единственным способом отследить хосты. Для приведенной выше схемы, чтобы выследить конкретный хост, вам нужно будет запустить команды для отображения MAC-адресов на всех коммутаторах в сети. Если хост или виртуальная машина мигрируют во время устранения неполадок, или из-за неправильной конфигурации возникает петля, вам, возможно, придется выполнять эти команды несколько раз.
- Проблему с дублирующимися MAC-адресами практически невозможно отследить.
- Проблема с балансировкой нагрузки может быть решена только агрегацией физических каналов в один логический, что усложняет управление сетью и не совсем эффективно.

Участие же самого сервера в маршрутизации решает все эти проблемы, упрощая как дизайн сети, так и последующий поиск неисправностей.

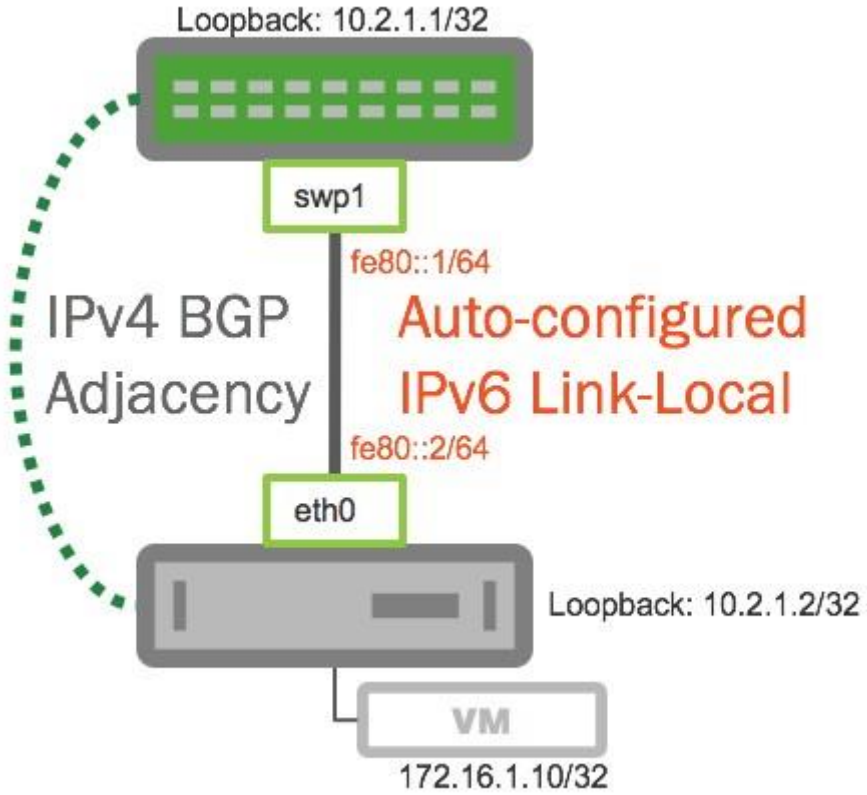
При маршрутизации на хосте, все виртуальные машины, контейнеры, подсети и т. д. анонсируются в сеть автоматически. Это означает, что необходимо только настроить подсеть между хостом и L3-коммутатором доступа. Это сильно упрощает и первоначальную конфигурацию, и дальнейшую

эксплуатацию. Все, что необходимо сделать коммутатору доступа — установить соседство с сервером при помощи выбранного вами протокола маршрутизации.



В приведенном примере конфигурация демона маршрутизации на сервере не требует изменений и может быть унифицированной для всех серверов в сети. Все, что требуется настраивать — подсеть между интерфейсом сервера eth0 и портом коммутатора swp1.

Кроме протокола маршрутизации OSPF, зачастую имеет смысл использовать протокол BGP для минимизации возможных рисков в случае с неверной настройкой сервера.



Демоны маршрутизации

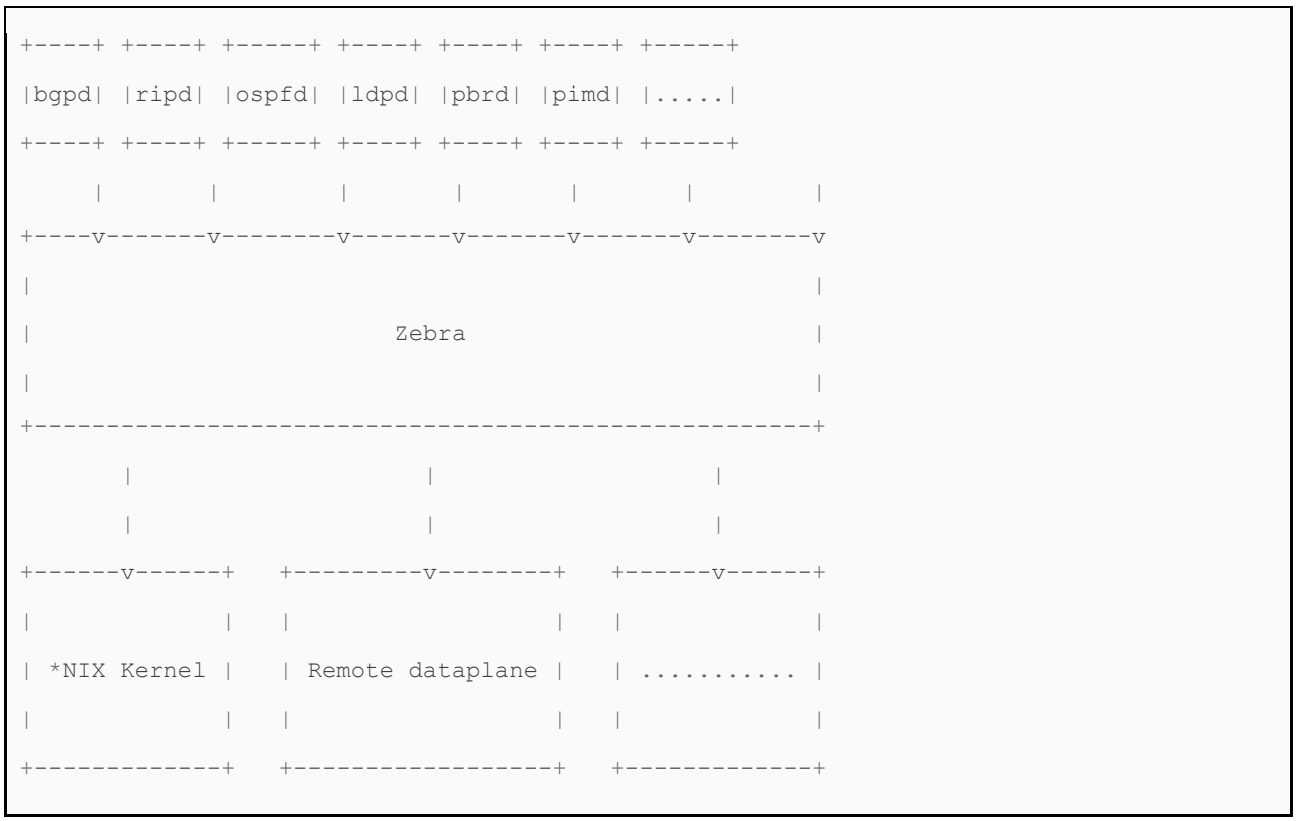
Free Range Routing

FRRouting (FRR) — пакет протоколов IP-маршрутизации для платформ Linux и Unix, который включает в себя демоны протоколов BGP, IS-IS, LDP, OSPF, PIM и RIP. Полная интеграция FRR со встроенными сетевыми стеками Linux/Unix IP делает его применимым к широкому спектру вариантов использования, включая подключение хостов/виртуальных машин/контейнеров в сеть, пиринга в Интернете.

FRR — форк проекта Quagga, распространяется по лицензии GPU GPL.

Традиционное программное обеспечение для маршрутизации выполнено в виде одной процессной программы, которая обеспечивает все функциональные возможности протокола маршрутизации. FRR использует другой подход. FRR — набор демонов, которые работают вместе для создания таблицы маршрутизации. Для каждого основного поддерживаемого протокола существует демон. Также есть посредник (Zebra), который помогает взаимодействовать этим демонам и ядру.

Эта архитектура обеспечивает высокую отказоустойчивость, поскольку ошибка, сбой или эксплойт в одном демоне протокола, как правило, не влияют на другие. Она также гибкая и расширяемая, так как модульность позволяет легко внедрять новые протоколы и связывать их в пакет.



Установка FRR

Установка может быть произведена как из готового RPM-пакета, так и исходного кода проекта на github.

```
https://github.com/FRRouting/
```

Скачиваем пакет frr на сервер

```
[root@r1                                                                    ~]#wget
https://github.com/FRRouting/frr/releases/download/frr-7.0/frr-7.0-01.el7.centos.x86_64
.rpm
```

Для установки версии 7 нам дополнительно понадобиться **libyang**. Его можно собрать из исходников на странице проекта — https://github.com/CESNET/libyang, либо поставить уже собранный RPM.

Устанавливаем libyang

```
[root@r1                                ~]#                                 wget
https://ci1.netdef.org/artifact/LIBYANG-YANGRELEASE/shared/build-10/CentOS-7-x86_64-Pac
kages/libyang-0.16.111-0.x86_64.rpm
```

```
[root@r1 ~]# yum install -y libyang-0.16.111-0.x86_64.rpm


Dependencies Resolved


================================================================================
```

```
==============================
 Package                  Arch                  Version                       Repository
Size
================================================================================================
============================== Updating:
 libyang                                  x86_64                                0.16.111-0
/libyang-0.16.111-0.x86_64               1.1 M

Transaction Summary
================================================================================================
==============================
Install  1 Package
```

Устанавливаем FRR

```
[root@r1 ~]# yum install frr-7.0-01.el7.centos.x86_64.rpm
Dependencies Resolved


================================================================
 Package
        Arch   Version      Repository                     Size
================================================================
Installing:
 frr    x86_64 7.0-01.el7.centos
                           /frr-7.0-01.el7.centos.x86_64 7.3 M
Installing for dependencies: c-ares x86_64 1.10.0-3.el7 base
 78 k

Transaction Summary
================================================================
Install  1 Package (+1 Dependent package)
```

Теперь необходимо включить возможность маршрутизации IPv4- и IPv6-пакетов в ядре, установив **net.ipv4.conf.all.forwarding=1** и **net.ipv6.conf.all.forwarding=1**. Чтобы настройка не слетала при перезагрузке, следует поместить эти значения в файл.

```
[root@r1 ~]# # vim /etc/sysctl.d/90-routing-sysctl.conf
net.ipv4.conf.all.forwarding=1
net.ipv6.conf.all.forwarding=1 [root@r1 ~]# sysctl -p
/etc/sysctl.d/90-routing-sysctl.conf
```

Включение нужных протоколов маршрутизации:

При установке FRR все поддерживаемые демоны маршрутизации отключены и вам предлагается включить ровно те протоколы, которые необходимы. Настройка находится в файле **/etc/frr/daemons**. Мы будем использовать только **ospf**, так что его и включаем.

```
[root@r1 ~]# cat /etc/frr/daemons watchfrr_enable=yes watchfrr_options="-r
'/usr/lib/frr/frr restart %s' -s '/usr/lib/frr/frr start %s' -k
'/usr/lib/frr/frr stop
%s'" # zebra=yes bgpd=no
ospfd=yes ospf6d=no
ripd=no ripngd=no isisd=no
ldpd=no pimd=no nhrpd=no
eigrpd=no babeld=no
sharpd=no pbrd=no
staticd=no bfdd=no
fabricd=no

# # Command line options for the
daemons # zebra_options=("-A
127.0.0.1") bgpd_options=("-A
127.0.0.1") ospfd_options=("-A
127.0.0.1") ospf6d_options=("-A ::1")
ripd_options=("-A 127.0.0.1")
```

```
ripngd_options=("-A ::1") isisd_options=("-
A 127.0.0.1") ldpd_options=("-A 127.0.0.1")
pimd_options=("-A 127.0.0.1")
nhrpd_options=("-A 127.0.0.1")
eigrpd_options=("-A 127.0.0.1")
babeld_options=("-A 127.0.0.1")
sharpd_options=("-A 127.0.0.1")
pbrd_options=("-A 127.0.0.1")
staticd_options=("-A 127.0.0.1")
```

```
bfdd_options=("-A 127.0.0.1")
fabricd_options=("-A 127.0.0.1")
vtysh_enable=yes
```

Теперь переходим к запуску демона:

```
[root@r1 ~]# systemctl start frr.service
[root@r1 ~]# systemctl status  frr
● frr.service - FRRouting (FRR)
          Loaded: loaded (/usr/lib/systemd/system/frr.service; disabled; vendor preset:
disabled)
   Active: active (running) since Sat 2019-09-07 19:57:38 CEST; 6s ago
  Process: 21390 ExecStop=/usr/lib/frr/frr stop (code=exited, status=0/SUCCESS)
  Process: 21446 ExecStart=/usr/lib/frr/frr start (code=exited, status=0/SUCCESS)
 Main PID: 21359 (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/frr.service
           ├─21463 /usr/lib/frr/zebra -d -A 127.0.0.1
           ├─21473 /usr/lib/frr/ospfd -d -A 127.0.0.1 └─21484
           /usr/lib/frr/watchfrr -d -r /usr/lib/frr/f...

Sep 07 19:57:38 r1 frr[21446]: zebra 2019/09/07 19:57:38 war...)
Sep 07 19:57:38 r1 frr[21446]: [  OK  ]
Sep 07 19:57:38 r1 frr[21446]: ospfd [  OK  ] Sep 07 19:57:38
r1 frr[21446]: Starting FRRouting monitor da...:
Sep 07 19:57:38 r1 watchfrr[21484]: watchfrr 7.0 starting: vty@0
Sep 07 19:57:38 r1 watchfrr[21484]: zebra state -> up : conne...
Sep 07 19:57:38 r1 frr[21446]: watchfrr[  OK  ] Sep 07 19:57:38
r1 watchfrr[21484]: ospfd state -> up : conne...
Sep 07 19:57:38 r1 watchfrr[21484]: all daemons up, doing sta...
Sep 07 19:57:38 r1 systemd[1]: Started FRRouting (FRR).
Hint: Some lines were ellipsized, use -l to show in full.
```

Как видно из вывода, у нас запущен демон OSPF на адресе 127.0.0.1. Для настройки ospfd следует подключиться к демону, используя **vtysh**. Нужные порты можно посмотреть в **/etc/services**.

```
[root@r1 ~]# cat /etc/services | tail -n 15 com-bardac-dw
48556/udp               # com-bardac-dw iqobject
48619/tcp               # iqobject iqobject
48619/udp               # iqobject matahari
49000/tcp               # Matahari Broker staticd
2616/tcp          # staticd vty isisd        2608/tcp
# ISISd vty ospfapi        2607/tcp          # OSPF-API
babeld        2609/tcp          # BABELd vty nhrpd
```

```
2610/tcp          # NHRPd vty pimd        2611/tcp
# PIMd vty pbrd        2615/tcp          # PBRd vty ldpd
2612/tcp          # LDPd vty eigrpd        2613/tcp
# EIGRPd vty bfdd        2617/tcp          # BFDd vty
fabricd        2618/tcp          # Fabricd vty
```

Проверить номер порта для ospfd можно и при помощи **ss:**

```
[root@r1 ~]# ss -tulpan | egrep -i osp
tcp       LISTEN        0          3          127.0.0.1:2604                                *:*
users:(("ospfd",pid=4000,fd=10))
```

Набрав на роутере vtysh, мы попадаем в отдельный cli, с cisco-like-синтаксисом.

```
[root@r1 ~]# vtysh

Hello, this is FRRouting (version 7.0).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

r1# show run
Building configuration...

Current configuration: !
frr version 7.0 frr
defaults traditional

hostname r1
!
line
vty !
end r1#
```

Для проверки того, что маршрутизация работает, возьмем 3 сервера, подключенных друг к другу по цепочке:

```
R1---192.168.13.0/24---R2---192.168.23.0/24---R3
```

Создадим на R1 и R3 серверах dummy0-интерфейс и назначим на него ip-адрес 1.1.1.1/32 и 3.3.3.3/32 соответсвенно. Учтите, что такая настройка не сохраняется при перезагрузке сервера.

R1:

```
[root@r1 ~]# ip link add dummy0 type dummy [root@r1 ~]# ip addr add 1.1.1.1/32 dev
dummy0
```

```
[root@r1 ~]# ip link set dummy0 up
[root@r1 ~]# ip a s dummy0
4: dummy0: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000 link/ether 26:2d:24:e6:36:b2 brd ff:ff:ff:ff:ff:ff inet 1.1.1.1/32 scope global dummy0 valid_lft forever preferred_lft forever inet6 fe80::242d:24ff:fee6:36b2/64 scope link valid_lft forever preferred_lft forever
[root@r1 ~]# ip a s ens37
3: ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000 link/ether 00:0c:29:ff:ea:c8 brd ff:ff:ff:ff:ff:ff inet 192.168.13.1/24 brd 192.168.13.255 scope global noprefixroute ens37 valid_lft forever preferred_lft forever inet6 fe80::20c:29ff:feff:eac8/64 scope link valid_lft forever preferred_lft forever
```

R3:

```
[root@r3 ~]# ip link add dummy0 type dummy [root@r3 ~]# ip addr add 3.3.3.3/32 dev dummy0
[root@r3 ~]# ip link set dummy0 up
[root@r3 ~]# ip a s dummy0
4: dummy0: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000 link/ether 96:85:2b:52:90:05 brd ff:ff:ff:ff:ff:ff inet 3.3.3.3/32 scope global dummy0 valid_lft forever preferred_lft forever inet6 fe80::9485:2bff:fe52:9005/64 scope link valid_lft forever preferred_lft forever
[root@r3 ~]# ip a s ens37
3: ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000 link/ether 00:0c:29:db:1a:53 brd ff:ff:ff:ff:ff:ff inet 192.168.23.3/24 brd 192.168.23.255 scope global noprefixroute ens37 valid_lft forever preferred_lft forever inet6 fe80::20c:29ff:fedb:1a53/64 scope link valid_lft forever preferred_lft forever
```

Интерфейс dummy0 эмулирует наличие виртуальной машины с таким IP-адресом и нужен нам, чтобы показать, что маршрутизация работает. Сейчас схема подключения выглядит вот так:

```
(1.1.1.1/32)R1---192.168.13.0/24---R2---192.168.23.0/24---R3(3.3.3.3/32)
```

Адрес 3.3.3.3 недоступен с сервера R1, так как демон маршрутизации OSPF включен, но не настроен. Давайте это исправим, и настроим OSPF на всех трех серверах. Для начала разрешим в **firewalld** OSPF-трафик:

```
R1,R2,R3# firewall-cmd --add-protocol=ospf --permanent --zone=public
R1,R2,R3# firewall-cmd --add-protocol=ospf --zone=public
```

Далее перейдем к настройке OSPF:

```
[root@r1 ~]# vtysh r1# show ip   route Codes: K - kernel
route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP, T -
       Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route

K>* 0.0.0.0/0 [0/102] via 192.168.1.1, ens33, 00:00:48
C>* 1.1.1.1/32 is directly connected, dummy0, 00:00:48
C>* 192.168.1.0/24 is directly connected, ens33, 00:00:48
C>* 192.168.13.0/24 is directly connected, ens37, 00:00:48
```

```
r1# conf t r1(config)# router ospf r1(config-
router)# network 192.168.13.0/24 area 0
r1(config-router)# network 1.1.1.1/32 area 0
r1(config)# ^Z r1# r1# r1# show run Building
configuration...

Current configuration: !
frr version 7.0 frr
defaults traditional
hostname r1 no ip
forwarding no ipv6
forwarding !
router ospf network 1.1.1.1/32
 area 0 network 192.168.13.0/24
 area 0
!
line
vty !
end r1#
r1# wr
Note: this version of vtysh never writes vtysh.conf
Building Configuration...
Configuration saved to /etc/frr/zebra.conf
Configuration saved to /etc/frr/ospfd.conf
```

R2:

```
[root@r2 ~]# vtysh

r2# show ip route Codes: K - kernel route, C - connected, S
- static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP, T -
       Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route
```

```
K>* 0.0.0.0/0 [0/100] via 192.168.1.1, ens33, 00:00:13
C>* 192.168.1.0/24 is directly connected, ens33, 00:00:13
C>* 192.168.13.0/24 is directly connected, ens37,
00:00:13 C>* 192.168.23.0/24 is directly connected,
ens38, 00:00:13 r2# conf t r2(config)# router ospf
r2(config-router)# network 192.168.13.0/24 area 0
r2(config-router)# network 192.168.23.0/24 area 0
r2(config-router)# ^Z r2# show run Building
configuration...

Current configuration: !
frr version 7.0 frr
defaults traditional
hostname r2 !
router ospf network
 192.168.13.0/24 area 0 network
 192.168.23.0/24 area 0
!
line
vty !
end
r2# wr
Note: this version of vtysh never writes vtysh.conf
Building Configuration...
Configuration saved to /etc/frr/zebra.conf
Configuration saved to /etc/frr/ospfd.con
```

R3:

```
[root@r3 ~]# vtysh

r3# show ip route Codes: K - kernel route, C - connected, S
- static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP, T -
       Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route
```

```
K>* 0.0.0.0/0 [0/100] via 192.168.1.1, ens33, 00:09:40
C>* 3.3.3.3/32 is directly connected, dummy0, 00:09:40
C>* 192.168.1.0/24 is directly connected, ens33, 00:09:40
C>* 192.168.23.0/24 is directly connected, ens37,
00:09:40 r3# conf t r3(config)# router ospf r3(config-
router)# network 192.168.23.0/24 area 0 r3(config-
router)# network 3.3.3.3/32 area 0 r3(config-router)# ^Z
r3# show run Building configuration...

Current configuration: !
frr version 7.0 frr
defaults traditional
hostname r3 !
router ospf network 3.3.3.3/32
 area 0 network 192.168.23.0/24
 area 0
!
line
vty !
end
r3# wr
Note: this version of vtysh never writes vtysh.conf
Building Configuration...
Configuration saved to /etc/frr/zebra.conf
Configuration saved to /etc/frr/ospfd.conf
r3#
```

После настройки всех трёх серверов в таблице маршрутизации у R1 появился маршрут до 3.3.3.3, а R3 маршрут до 1.1.1.1. Также появилась связность между ними.

```
r1# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP, T -
       Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route
```

```
K>* 0.0.0.0/0 [0/102] via 192.168.1.1, ens33, 00:15:54
O   1.1.1.1/32 [110/10] via 0.0.0.0, dummy0 onlink, 00:14:02
C>* 1.1.1.1/32 is directly connected, dummy0, 00:15:54
O>* 3.3.3.3/32 [110/210] via 192.168.13.2, ens37, 00:00:33
C>* 192.168.1.0/24 is directly connected, ens33, 00:15:54 O
192.168.13.0/24 [110/100] is directly connected, ens37, 00:08:10
C>* 192.168.13.0/24 is directly connected, ens37, 00:15:54
O>* 192.168.23.0/24 [110/200] via 192.168.13.2, ens37,
00:08:0 r1# ping 3.3.3.3 PING 3.3.3.3 (3.3.3.3) 56(84) bytes
of data. 64 bytes from 3.3.3.3: icmp_seq=1 ttl=63 time=0.652
ms
^C
--- 3.3.3.3 ping statistics --1 packets transmitted, 1
received, 0% packet loss, time 0ms rtt min/avg/max/mdev =
0.652/0.652/0.652/0.000 ms r1# exit [root@r1 ~]# mtr
3.3.3.3
                                        My traceroute  [v0.85]
 Host                                Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 192.168.13.2                     0.0%     5    0.4   0.4   0.3   0.4   0.0
 2. 3.3.3.3                          0.0%     5    0.6   0.7   0.6   0.7   0.0
```

Как видно из трассировки, пакеты от роутера R1 идут через роутер R2 до dummy0-интерфейса R3. То есть наша задача выполнена и связность есть.

Аналогичным функционалом, но несколько другим синтаксисом и архитектурой построения, обладает демон маршрутизации BIRD. Почитать о нем подробнее можно на странице проекта The BIRD Internet Routing Daemon.

А вот BGP-демон ExaBGP, в отличие от frr/quagga/bird, не умеет маршрутизировать, но может участвовать в анонсировании маршрутной информации для протокола BGP и отлично подходит на роль SDN-контроллера, или для использования его только на конечном хосте.

Почитать об этом подробнее можно на странице проекта Exa-Networks/exabgp.

Firewall

Традиционно межсетевые экраны настраивались только на границе между локальными и внешними сетями, такими как Интернет. Но с ростом угроз безопасности растет потребность в брандмауэрах на каждом хосте. Centos7 включает в себя файервол в установке по умолчанию.

Ядро Linux поставляется с мощной структурой, системой Netfilter, которая позволяет другим модулям ядра предлагать такие функции, как фильтрация пакетов, преобразование сетевых адресов (NAT) и

распределение нагрузки. Приложение iptables — основной инструмент, который взаимодействует с системой Netfilter для обеспечения фильтрации пакетов и NAT.

Перед отправкой сообщения по IP-сети сообщение разбивается на более мелкие блоки, называемые пакетами. Административная информация, включая тип данных, адрес источника и адрес назначения, добавляется к каждому пакету. Пакеты повторно собираются, когда достигают конечного компьютера. Правило iptables проверяет эти административные поля в каждом пакете, чтобы определить, следует ли разрешить прохождение пакета, или он должен быть отброшен.

iptables — основа, которая используется другими службами для управления правилами брандмауэра системы. Centos7 поставляется с двумя такими сервисами: новым демоном firewalld и сервисом iptables, который был включен в предыдущие выпуски Centos. Вы можете взаимодействовать с firewalld с помощью графической утилиты firewall-config или клиента командной строки firewall-cmd.

Службы iptables и firewalld используют систему Netfilter в ядре Linux для фильтрации пакетов. Однако, хотя iptables основан на концепции «цепочки правил фильтрации» для блокировки или пересылки трафика, firewalld «основан на зоне», то есть является zone based firewall (zbf).

Философия, лежащая в основе iptables, основана на «цепочках». Это наборы правил, применяемые к каждому сетевому пакету, объединенные вместе. Каждое правило выполняет две вещи: оно определяет условия, которым должен соответствовать пакет, чтобы соответствовать правилу, и определяет действие, если пакет соответствует.

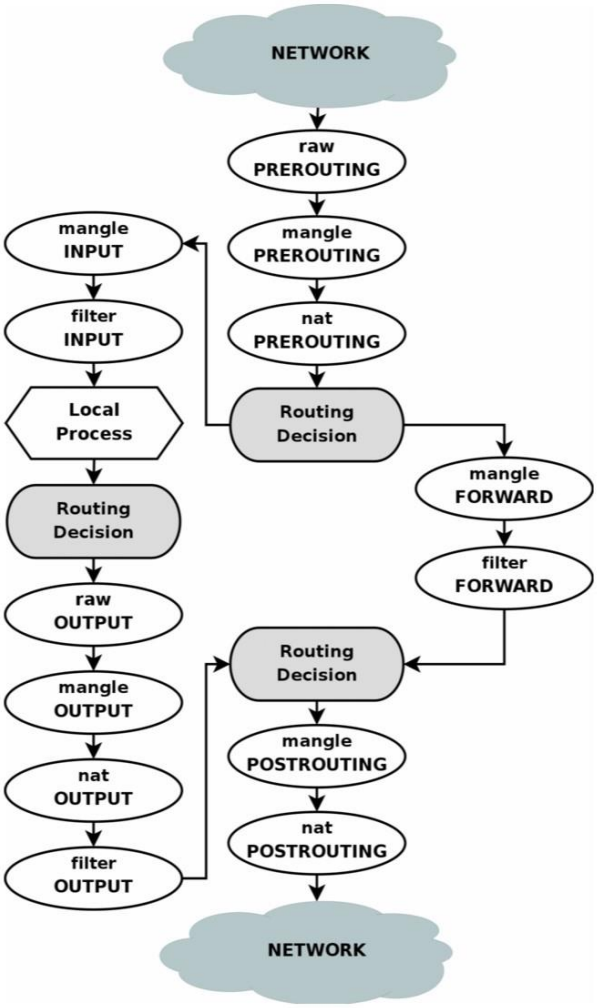
Формат запуска:

```
iptables [-t таблица] команда [критерий] [ -j цель]
```

Команды:

- -L — посмотреть список правил (можно указать цепочку, иначе выводятся все);
- -F — удалить все правила (можно указать цепочку, иначе очистятся все);
- -A — добавить правило (критерий и цель) в заданную цепочку;
- -D — удалить правило (критерий и цель, либо порядковый номер в заданной цепочке);
- -P — установить политику по умолчанию для заданной цепочки;
- -I — вставка нового правила в цепочку;
- -N — создать новую цепочку с заданным именем в таблице;
- -X — удалить заданную цепочку (но сначала нужно удалить в ней правила, и на нее не должны ссылаться правила из других цепочек).

Стандартный путь пакета через цепочки и таблицы можно посмотреть на этой диаграмме:



Если говорить про фильтрацию трафика (таблицу **FILTER**), на находится в цепочках:

- INPUT, • OUTPUT,
- FORWARD.

INPUT и **OUTPUT**, как видно из схемы, применяются для трафика, получатель которого — сам сервер, тогда как **FORWARD** нужен для транзитного трафика сквозь сервер. В этом случае сервер — маршрутизатор.

Посмотреть правила для таблицы **FILTER**:

```
root@r1:~# iptables -L Chain INPUT (policy ACCEPT) target     prot opt
source               destination
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT) target     prot opt
source               destination root@r1:~#
```

Видим цепочки в выводе команды. Вспомним про этапы обработки пакетов на уровне ядра. Обратите внимание, что политика по умолчанию — ACCEPT, то есть все пакеты по умолчанию пропускаются. Существуют и другие политики (например, DROP).

```
root@r1:~# iptables -P INPUT DROP
```

Политика по умолчанию задает порядок работы с пакетами, для которых нет ни одного правила. Если в цепочке правил ни одно не подошло, в итоге применяется действие по умолчанию, которое и задается политикой. Если правило подошло, действие по умолчанию не выполняется.
Цепочка — это набор правил, которые проверяются по пакетам поочередно (напоминает язык программирования). В таблице filter видим цепочки INPUT, FORWARD и OUTPUT. Но есть и другие таблицы (их нужно указывать явно): nat, когда нам необходима трансляция адресов и портов, и mangle — когда требуется внести изменения в пакет (например, установить ttl или промаркировать трафик для применения политик QoS).

```
root@r1:~# iptables -t nat -L Chain PREROUTING
(policy ACCEPT) target prot opt source
destination Chain INPUT (policy ACCEPT) target
prot opt source destination Chain OUTPUT
(policy ACCEPT) target prot opt source
destination Chain POSTROUTING (policy ACCEPT) target
prot opt source destination
```

Для критериев можно использовать следующие параметры:

- -p — тип протокола: tcp, udp, icmp (полный список здесь: cat /etc/protocols);
- -s — IP-адрес отправителя (можно использовать адреса сетей с маской

  10.0.0.0/255.0.0.0 или 10.0.0.0/8). Можно применять логическое отрицание, поставив символ ! перед адресом;
- -d — IP-адрес отправителя (синтаксис аналогичный);
- -i — сетевой интерфейс, с которого получен пакет (по умолчанию пакеты

  обрабатываются независимо от того, с какого интерфейса они пришли);
- -o — сетевой интерфейс, в который предполагается отправка пакета. Может

  применяться только в цепочках OUTPUT, FORWARD и POSTROUTING;
- -m — лимит, устанавливает предельное число пакетов в единицу времени, которое способно пропустить правило;
- --sport — порт (или диапазон портов) отправителя;
- --dport — порт назначения (куда предназначен пакет).

Остальные параметры вы можете изучить самостоятельно.

В качестве цели можно использовать следующий действия:

- ACCEPT — принять пакет;
- REJECT — отбросить пакет, но выдать сообщение об ошибке на хост отправителя;
- DROP — отбросить пакет, не уведомляя отправителя;
- SNAT (для таблицы NAT) — преобразовать IP-адрес отправителя в заголовке пакета, подменив адрес, указанный в критерии, на свой «белый» IP-адрес, указанный после SNAT (--to-source=...);
- DNAT (для таблицы NAT) — destination NAT (когда надо перенаправить пакеты с одной машины на несколько разных в зависимости от критериев);
- MASQUERADE (для таблицы NAT) — преобразовать IP-адрес отправителя в заголовке пакета, подменив адрес, указанный в критерии, на свой (например, если он динамический);
- LOG — журналировать пакеты и события;
- MARK (для таблицы mangle) — пометить пакет;
- TTL (для таблицы mangle) — изменить TTL (например, установить его снова в 64, скрыв движения по локальной сети от провайдера).

Другие действия изучите самостоятельно.

Обратите внимание: имеет значение порядок правил в цепочке. Сравните:

```
root@r1:~# iptables -A INPUT -p tcp -j DROP root@r1:~#
iptables -A INPUT -p tcp --dport=22 -j ACCEPT
```

Этот вариант полностью отрежет компьютер от внешнего мира. Несмотря на то, что второе правило разрешает порт 22 (ssh), до него дело не дойдет, так как при срабатывании правила (а оно сработает на любой пакет с TCP-сегментом) будет переход на DROP и завершение движения в цепочке.

А такой вариант сначала разрешит порт 22 и дальше проверять не будет. Другие порты пойдут дальше и будут отброшены:

```
root@r1:~# iptables -A INPUT -p tcp --dport=22 -j ACCEPT
root@r1:~# iptables -A INPUT -p tcp -j DROP
```

Установим для таблиц по умолчанию политики DROP

Примечание: не делайте так, работая по ssh. Сначала по ssh настройте правила, дающие доступ к машине, а уже потом применяйте политики. На локальной машине можно сразу.

```
root@r1:~# iptables -P INPUT DROP
root@r1:~# iptables -P OUTPUT DROP
root@r1:~# iptables -P FORWARD DROP
```

Таким способом мы запретим и межпроцессный обмен, и обмен через локальную петлю (loopback). Не будет действовать локальный DNS-сервер (в Centos он работает по адресу 127.0.1.1), PHP не сможет обратиться к MySQL, и так далее. Поэтому:

```
root@r1:~# iptables -A INPUT -i lo -j ACCEPT
root@r1:~# iptables -A OUTPUT -o lo -j ACCEPT
```

Запрет ICMP — это неправильно, не бывает ситуаций, когда вам требуется запретить ICMP. ICMP — часть механизма IP и служит для нормальной работы стека TCP/IP. Машины, у которых полностью запрещен ICMP, создают проблемы при конфигурации сетей, являясь своего рода «черными дырами» из-за сломанного механизма path mtu discovery (pmtud).

```
root@r1:~# iptables -A INPUT -p icmp -j ACCEPT
root@r1:~# iptables -A OUTPUT -p icmp -j ACCEPT
```

Далее мы можем разрешить локальные соединения с динамических портов (мы уже знаем, где их посмотреть):

```
root@r1:~# cat /proc/sys/net/ipv4/ip_local_port_range 32768 61000
root@r1:~# iptables -A OUTPUT -p TCP --sport 32768:61000 -j ACCEPT
root@r1:~# iptables -A OUTPUT -p UDP --sport 32768:61000 -j ACCEPT
```

Разрешить возвращающиеся к нам пакеты:

```
root@r1:~# iptables -A INPUT -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT
root@r1:~# iptables -A INPUT -p UDP -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Разрешить входящие пакеты, например для ssh:

```
root@r1:~# iptables -A INPUT -i ens33-p TCP --dport 22 -j ACCEPT
root@r1:~# iptables -A OUTPUT -o ens33 -p TCP --sport 22 -j ACCEPT
```

Разрешили ssh. Если работали через него, теперь можно применять политики по умолчанию.

Аналогично откройте доступ на порты веб-сервера 80 и 443. Проверьте порты настроенной машины с помощью nmap самостоятельно (просканируйте порты с другой машины).

Сохранение значений

Обратите внимание, что все значения, вводимые через iptables, не будут сохранены при перезагрузке.

Для сохранения можно воспользоваться двумя способами:

1. Установить демон iptables-persistent, который будет сохранять введенные значения.
2. Использовать утилиты iptables-save и iptables-restore. Но вызывать их нужно самостоятельно и при сохранении, и при старте.

По умолчанию iptables-save выводит список правил в stdout, а iptables-restore читает из stdin. Поэтому надо перенаправить в файл:

```
root@r1:~# iptables-save > /etc/iptables.rules
root@r1:~# iptables-restore < /etc/iptables.rules
```

Firewalld

Служба firewalld предлагает не только те же функции, что и iptables, и многие другие. Одна из новых функций firewalld — межсетевой экран на основе зон (zone based firewall). В брандмауэре на основе зон интерфейсы группируются в зоны, каждая из которых настроена с различным уровнем доверия. То есть политики файерволла применяются к зоне в целом, а не к конкретному интерфейсу, что упрощает управление. Зоны, определенные в firewalld, перечислены в таблице вместе с их поведением по умолчанию для исходящих и входящих соединений.

Zone	Outgoing Connections	Incoming Connections
drop	Allowed	Dropped.
block	Allowed	Rejected with an icmp-host-prohibited message.
public	Allowed	DHCPv6 client and SSH are allowed.
external	Allowed and masqueraded to the IP address of the outgoing network interface	SSH is allowed.
dmz	Allowed	SSH is allowed.
work	Allowed	DHCPv6 client, IPP and SSH are allowed.
home	Allowed	DHCPv6 client, multicast DNS, IPP, Samba client, and SSH are allowed.
internal	Allowed	Same as the home zone.
trusted	Allowed	Allowed.

Для управления файерволом используется утилита firewall-cmd:

```
[root@r1 ~]# systemctl start firewalld
[root@r1 ~]# firewall-cmd --get-default-zone
public [root@r1 ~]# firewall-cmd --list-all
public (active) target: default icmp-block-
inversion: no interfaces: ens33 ens37
sources:
  services: ssh dhcpv6-client
  ports:
  protocols: ospf
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

Команды для добавления/удаления порта в политику фильтрации стали гораздо проще и очевиднее.

Например, чтобы разрешить http:

```
[root@r1 ~]# firewall-cmd --zone=public --add-service=http
success
```

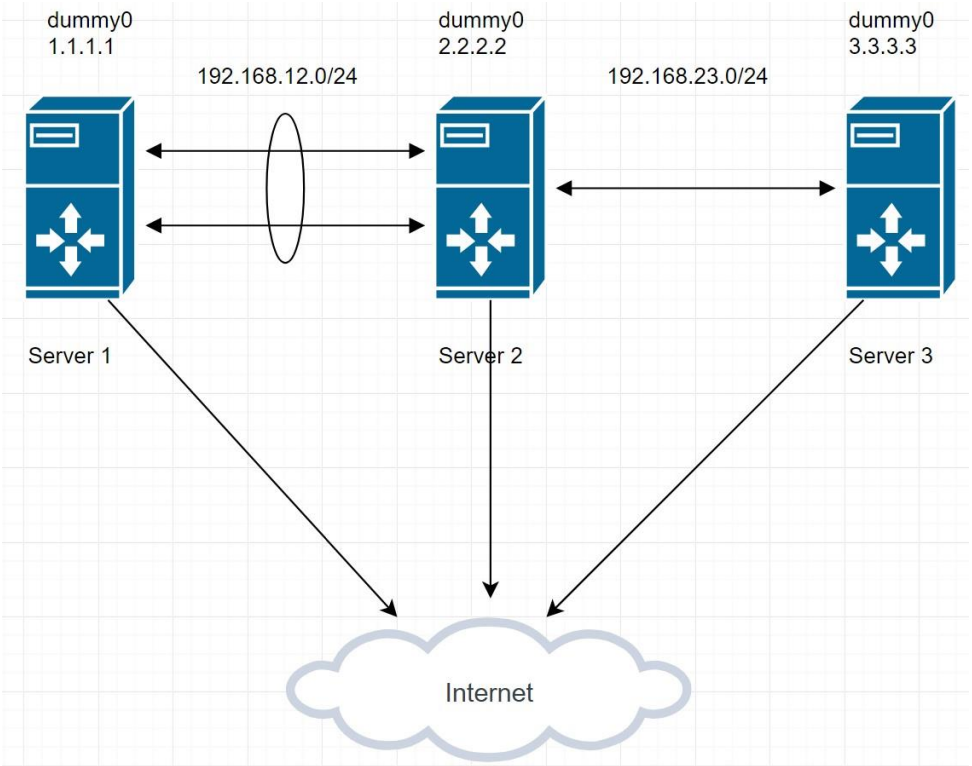
Стандартно, все правила firewalld действуют только в рантайме, то есть не переживают перезагрузку.

Чтобы правило было постоянным, необходимо написать его еще раз с ключем --permanent:

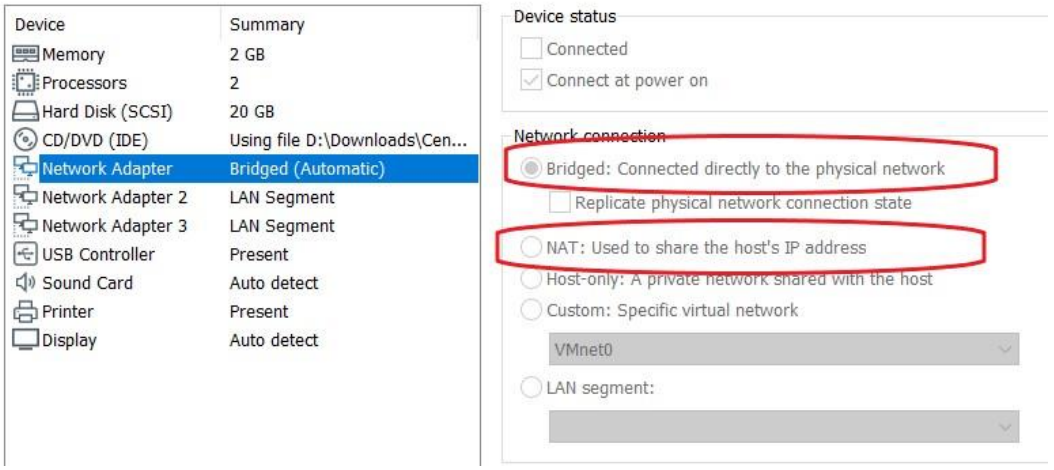
```
[root@r1 ~]# firewall-cmd --zone=public --add-service=http --permanent
success
```

Домашнее задание

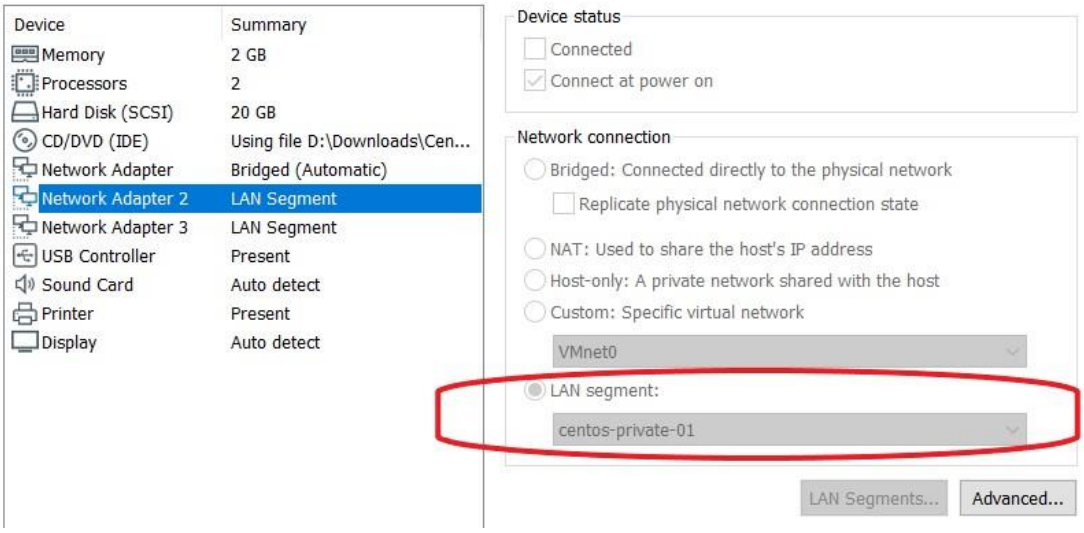
Топология:



1. Собрать схему из трёх серверов. Два сервера должны иметь как минимум 3 сетевых адаптера. Один сервер должен иметь 2 сетевых адаптера.
2. Первый интерфейс на каждой виртуальной машине имеет режим подключения bridge (сетевой мост) или nat для предоставления доступа в интернет и по ssh из родительской операционной системы. В этом примере используется bridge, так как есть роутер провайдера, который раздает IP-адреса.



3. Все последующие интерфейсы между серверами организуют отдельные изолированные сегменты. Режим подключения — LAN Segment. Делается это, чтобы изолировать коммуникацию между сетевыми адаптерами устройств.



4. Настроить любой из интерфейсов между server1 и server2. Назначить на него адреса из подсети 192.168.12.0/24. Второй интерфейс между ними остается отключенным и в этом задании не участвует.
5. Настроить подсеть между server2 и server3 с адресами из подсети 192.168.23.0/24.
6. На каждом из серверов поднять dummy0-интерфейс и назначить на него ip-адрес 1.1.1.1/32, 2.2.2.2/32, 3.3.3.3/32 соответственно.
7. На серверах установить пакет frr и настроить на роутерах ospf, добавив подсети 192.168.12.0/24, 192.168.23.0/24, 1.1.1.1/32, 2.2.2.2/32, 3.3.3.3/32 в area 0.
8. Убедиться, что маршрутизация работает, и с server1 вы должны пинговать 3.3.3.3 адрес на server3. Убедитесь, что нужный тип трафика разрешен в firewalld и что трафик не улетает в интернет при помощи traceroute.
9. На server3 создайте 2 папки nfs_1 и nfs_2, добавьте их в export.
10. Убедитесь, что только server1 может их примонтировать.
11. Убедитесь, что после перезагрузки server1 все еще может писать и читать файлы в примонтированных папках
12. * На server3 создайте iSCSI target размером 2GB и примонтируйте этот LUN на server1. Создайте там файловую систему xfs. Убедитесь, что диск будет активным после перезагрузки.

Задачи со * предназначены для продвинутых учеников, которым мало сделать обычное задание.

Дополнительные материалы

1. Информация по BIRD.
2. Информация по ExaBGP.
3. Информация по настройке и протоколу OSPF.

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. Проект FRR.
2. Документация от Cisco Systems.
3. Документация от RedHat.