

Летняя школа ВШЭ. День 3.

Лабораторная работа - сетевые неймспейсы, veth-пары, виртуальный bridge и iptables в Linux.

Цели лабораторной работы:

1. Научиться создавать сетевые неймспейсы и veth интерфейсы.
2. Соединить неймспейсы через виртуальный bridge для обеспечения связи между ними.
3. Настроить и использовать iptables внутри сетевых неймспейсов для управления сетевым трафиком.

Часть 1. Создание двух сетевых неймспейсов и соединение их с помощью veth.

1. Создание сетевых неймспейсов ns1 и ns2:

```
sudo ip netns add ns1  
sudo ip netns add ns2
```

Эти команды создают два изолированных сетевых пространства, которые будут использоваться для последующей настройки сети. Сетевые неймспейсы позволяют изолировать сетевые интерфейсы, маршруты и правила iptables.

2. Создание пары veth интерфейсов:

```
sudo ip link add veth0 type veth peer name veth1
```

Эта команда создает пару виртуальных Ethernet-интерфейсов. Два конца veth0 и veth1 могут быть помещены в разные неймспейсы для их соединения.

3. Помещение veth интерфейсов в неймспейсы ns1 и ns2:

```
sudo ip link set veth0 netns ns1  
sudo ip link set veth1 netns ns2
```

Теперь каждый конец veth пары находится в отдельном сетевом неймспейсе, связывая ns1 и ns2.

4. Активирование veth интерфейсов:

```
sudo ip netns exec ns1 ip link set veth0 up  
sudo ip netns exec ns2 ip link set veth1 up
```

5. Назначение IP адресов интерфейсам:

```
sudo ip netns exec ns1 ip addr add 10.0.0.1/24 dev veth0  
sudo ip netns exec ns2 ip addr add 10.0.0.2/24 dev veth1
```

Здесь интерфейсы veth активируются, и им назначаются IP-адреса, что позволяет нам проверить связь между ними.

6. Проверка соединения с помощью ping:

```
sudo ip netns exec ns1 ping 10.0.0.2  
sudo ip netns exec ns2 ping 10.0.0.1
```

Отчетность: как отчет о части 1 принимаются скриншоты с ping с одного неймспейса в другой и обратно + команда «iptables a» в каждом неймспейсе.

Часть 2. Добавление еще двух неймспейсов и соединение всех через виртуальный bridge.

1. Удаляем старую пару интерфейсов для ns1 и ns2

```
sudo ip netns exec ns1 ip link delete veth0
```

2. Создание дополнительных неймспейсов ns3 и ns4:

```
sudo ip netns add ns3
```

```
sudo ip netns add ns4
```

Создаются еще два сетевых неймспейса для дальнейшей работы.

3. Создание veth для подключения новых неймспейсов и создание виртуального моста (bridge):

```
sudo ip link add veth0 type veth peer name br-veth0
```

```
sudo ip link add veth1 type veth peer name br-veth1
```

```
sudo ip link add veth2 type veth peer name br-veth2
```

```
sudo ip link add veth3 type veth peer name br-veth3
```

```
sudo ip link add name br0 type bridge
```

Создаются дополнительные пары veth и виртуальный bridge br0, который будет использоваться для подключения всех неймспейсов.

4. Подключение всех veth интерфейсов к неймспейсам:

```
sudo ip link set veth0 netns ns1
```

```
sudo ip link set veth1 netns ns2
```

```
sudo ip link set veth2 netns ns3
```

```
sudo ip link set veth3 netns ns4
```

5. Связываем интерфейсы с bridge:

```
sudo ip link set br-veth0 master br0
```

```
sudo ip link set br-veth1 master br0
```

```
sudo ip link set br-veth2 master br0
```

```
sudo ip link set br-veth3 master br0
```

6. Поднимаем интерфейсы со стороны моста:

```
sudo ip link set br0 up
```

```
sudo ip link set br-veth0 up
```

```
sudo ip link set br-veth1 up
```

```
sudo ip link set br-veth2 up
```

```
sudo ip link set br-veth3 up
```

7. Поднимаем интерфейсы в неймспейсах:

```
sudo ip netns exec ns1 ip link set veth0 up
```

```
sudo ip netns exec ns2 ip link set veth1 up
```

```
sudo ip netns exec ns3 ip link set veth2 up
```

```
sudo ip netns exec ns4 ip link set veth3 up
```

Один конец каждой veth пары подключается к неймспейсам, а другой — к bridge. Это соединяет все неймспейсы через общий виртуальный bridge br0.

8. Назначение IP-адресов интерфейсам в наших ns:

```
sudo ip netns exec ns1 ip addr add 10.0.0.1/24 dev veth0
sudo ip netns exec ns2 ip addr add 10.0.0.2/24 dev veth1
sudo ip netns exec ns3 ip addr add 10.0.0.3/24 dev veth2
sudo ip netns exec ns4 ip addr add 10.0.0.4/24 dev veth3
```

Интерфейсы veth активируются, и им назначаются IP-адреса, что позволяет настроить связь через bridge.

9. Отключите iptables для bridge интерфейса:

```
sudo sysctl net.bridge.bridge-nf-call-iptables=0
```

10. Проверка связи между всеми неймспейсами:

```
sudo ip netns exec ns1 ping 10.0.0.2
sudo ip netns exec ns1 ping 10.0.0.3
sudo ip netns exec ns1 ping 10.0.0.4
```

11. Включите iptables для bridge интерфейса:

```
sudo sysctl net.bridge.bridge-nf-call-iptables=1
```

Часть 3. Настройка iptables для разрешения трафика между неймспейсами через мост.

iptables — это утилита командной строки, которая позволяет настраивать правила фильтрации сетевых пакетов в Linux. Она работает на уровне ядра через подсистему Netfilter и управляет тем, как пакеты обрабатываются и маршрутизируются через систему.

Структура iptables

1. Таблицы (tables).

Таблицы в iptables — это группы цепочек, каждая из которых предназначена для выполнения определённого типа задач.

Основные таблицы:

- Filter - отвечает за фильтрацию пакетов (по умолчанию используется для большинства правил).
- Nat - используется для Network Address Translation (NAT), например, для изменения IP-адресов в пакетах.
- Mangle - предназначена для изменения заголовков пакетов.
- Raw - используется для правил, которые должны применяться к пакетам до их отслеживания состоянием.

2. Цепочки (chains).

Цепочки — это последовательности правил, через которые проходят пакеты. В каждой таблице могут быть встроенные и пользовательские цепочки.

Основные встроенные цепочки:

- INPUT- обрабатывает пакеты, поступающие на локальную машину.

- OUTPUT - обрабатывает пакеты, отправляемые с локальной машины.
- FORWARD - обрабатывает пакеты, проходящие через машину (маршрутизация).
- PREROUTING - обрабатывает пакеты перед их маршрутизацией.
- POSTROUTING - обрабатывает пакеты после маршрутизации.

3. Правила (rules).

Правила — это условия, которые применяются к пакетам. Если пакет соответствует условию правила, выполняется действие (например, принять или отклонить пакет).

Пример правила: `iptables -A INPUT -p tcp --dport 22 -j ACCEPT`

-A INPUT: Добавляет правило в цепочку INPUT.

-p tcp --dport 22: Применяется к пакетам протокола TCP на порт 22.

-j ACCEPT: Разрешает (принимает) соответствующие пакеты.

Практическая работа с IPtables.

1. Разрешение пересылки трафика между интерфейсами моста:

```
sudo iptables -A FORWARD -i br0 -o br0 -j ACCEPT
sudo iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
```

2. Разрешение входящего и исходящего трафика на интерфейсах моста

```
sudo iptables -A INPUT -i br0 -j ACCEPT
sudo iptables -A OUTPUT -o br0 -j ACCEPT
```

3. Запрет пинга в ns2:

```
sudo ip netns exec ns2 iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
```

Мы с помощью iptables запрещаем входящий ICMP трафик.

4. Проверьте правило, которое вы добавили в ns2:

```
sudo ip netns exec ns2 iptables -L -v
```

Как вывод вы должны увидеть ваше правило в вашем iptables.

5. Проверьте работу правила и пропингуйте интерфейс в ns2:

```
sudo ip netns exec ns1 ping 10.0.0.2
```

Если вы верно описали правило ICMP трафик будет дропнут правилами iptables.

6. Пропингуйте другой интерфейс, он должен быть доступен:

```
sudo ip netns exec ns1 ping 10.0.0.3
```

7. Удалите правило iptables в ns2:

```
sudo ip netns exec ns2 iptables -D INPUT -p icmp --icmp-type echo-request -j DROP
```

8. Проверьте удаление правила и пропингуйте интерфейс в ns2, он должен быть доступен:

```
sudo ip netns exec ns1 ping 10.0.0.2
```

Отчет для этого раздела – скриншот с правилом IPtables, пинг до ns2 (с правилом и без) и ns 3 - все на одном скриншоте.

Заключение:

В этой лабораторной работе вы научились:

1. Создавать и настраивать сетевые неймспейсы и veth интерфейсы для их соединения.
2. Подключать неймспейсы через виртуальный bridge, обеспечивая взаимодействие между ними.
3. Настраивать iptables внутри неймспейсов для управления сетевым трафиком, фильтрации и безопасности.