МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«УФИМСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра вычислительной техники и инженерной кибернетики

Сетевое администрирование Linux

Учебно-методическое пособие

Уфа 2020

Учебно-методическое пособие предназначено для подготовки бакалавров всех форм обучения направления подготовки: 09.03.01 «Информатика и вычислительная техника» по дисциплине «Сети и телекоммуникации». Пособие соответствует требованиям ФГОС 3++.

В пособии представлены подробные методические указания к выполнению работ вкупе с инструкциями по решению возможных проблем, а также контрольные вопросы для проверки понимания студентом материала.

Составители: Тулупова О.П., канд. техн. наук, доцент кафедры ВТИК
Ганиева В.Р., ст. преп. кафедры ВТИК
Демин Т.А, ст. гр. БПОи-17-01

Рецензенты: Филиппов В.Н., канд. техн. наук, доцент кафедры ВТИК
Дружинская Е.В., ст. преп. кафедры ВТИК

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Методическое пособие “Сетевое администрирование Linux” ФГБОУ ВО “УГНТУ” включает в себя восемь лабораторных работ.

Первые четыре лабораторных работы посвящены основным задачам, встающим перед администратором при создании локальной вычислительной сети — настройке разрешения имен, выдаче IP-адресов, маршрутизации, и другим. Остальные лабораторные работы описывают настройку основных сетевых сервисов, протокол IP версии 6 и настройку файрволла для фильтрации сетевых пакетов. Также затронуто создание виртуальных частных сетей и анализ сетевого трафика.

Общая информация

Данное пособие описывает проведение курса работ на Alpine Linux, запускаемом внутри виртуальных машин. Alpine Linux — свободная легковесная операционная система, хорошо подходящая для запуска внутри виртуальных машин и контейнеров, занимающая в образе для запуска в Docker лишь 3 МБ, а в образе для запуска в гипервизоре — 40 МБ. Она также подходит для различных устройств с небольшим объемом памяти (роутеров и других встраиваемых устройств).

libvirt — популярный свободный тулкит для управления множеством нижележащих платформ виртуализации, включающий в себя открытый API, а также сервер виртуализации и базовые средства управления им.

Hyper-V — гипервизор, созданный компанией Microsoft специально для операционных систем семейства Windows.

В лабораторных работах можно использовать любой другой гипервизор, если вы с ним знакомы. Для выполнения лабораторных работ от компьютера требуется x86_64-совместимый процессор, поддержка аппаратной виртуализации в системе, а также хотя бы 4 ГБ установленной оперативной памяти.

Установка гипервизора

libvirt

В зависимости от используемого вами дистрибутива установка libvirt будет разниться. В Arch Linux установка всех необходимых пакетов (при условии, что включен дистрибутив community) и включение автозагрузки libvirtd при запуске системы будет выглядеть следующим образом:

```
# pacman -S virt-manager virt-viewer libvirt
# systemctl enable --now libvirtd.service
```

Hyper-V

Для установки гипервизора Hyper-V требуется операционная система Microsoft Windows 8.1 или 10 (соответственно, либо Windows Server 2012 R2/2016). В настольных редакциях данной операционной системы Hyper-V доступен только начиная с редакции Professional, в редакции Home Hyper-V установить нельзя.

Установку Hyper-V можно осуществить, открыв `OptionalFeatures.exe` и выбрав в открывшемся окне Hyper-V, после чего перезагрузиться. После перезагрузки Hyper-V начнет работать, но прекратят работать все остальные установленные в системе гипервизоры, использующие аппаратную виртуализацию (VirtualBox, VMware, и другие).

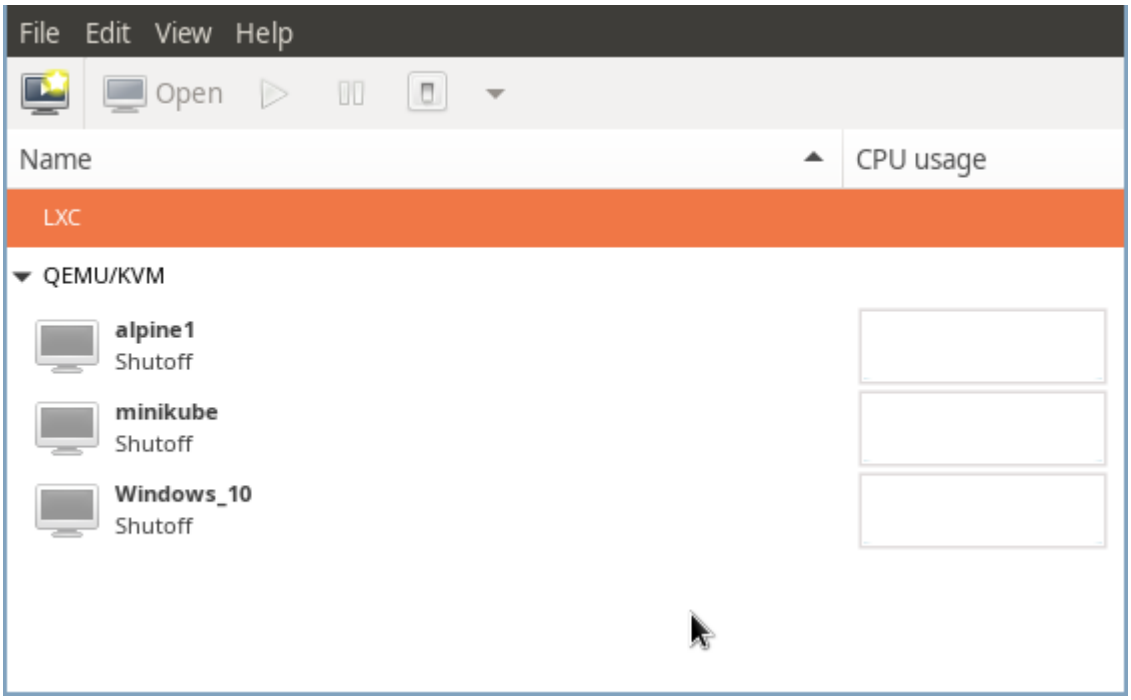
Альтернативно Hyper-V можно установить, набрав следующую команду в PowerShell от администратора:

```
PS Admin > Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V -All
```

Использование гипервизора

libvirt

libvirt представлен в системе демоном `libvirtd`, с которым связываются по сети клиентские программы, набором утилит `virt-*` (`virt-install`, `virt-ls`, `virt-inspector`, и другие, предоставляются пакетом virt-manager) и интерактивной оболочкой `virsh` для управления машинами. Существует также графическая оболочка `virt-manager`.



Список машин в virt-manager

С пакетом libvirt предустанавливается справка, доступная через программу man. Большая часть необходимой справки доступна по вызову `man` с названием соответствующей программы из пакета libvirt. Настоятельно рекомендуется читать таковую при чтении данного методического пособия, а также ознакомиться с документацией на официальном сайте проекта libvirt.

Для подключения к виртуальной машине можно использовать virt-viewer, установка которого описана выше. Например, запуск уже созданной виртуальной машины `jeongyeon` (*Чонён*) и последующее к ней подключение может выглядеть следующим образом:

```
# virsh -c qemu:///system start jeongyeon
# virt-viewer -c qemu:///system jeongyeon
```

Для того, чтобы не запускать virt-команды с правами суперпользователя каждый раз, можно добавить своего пользователя в группу `libvirt` следующей командой:

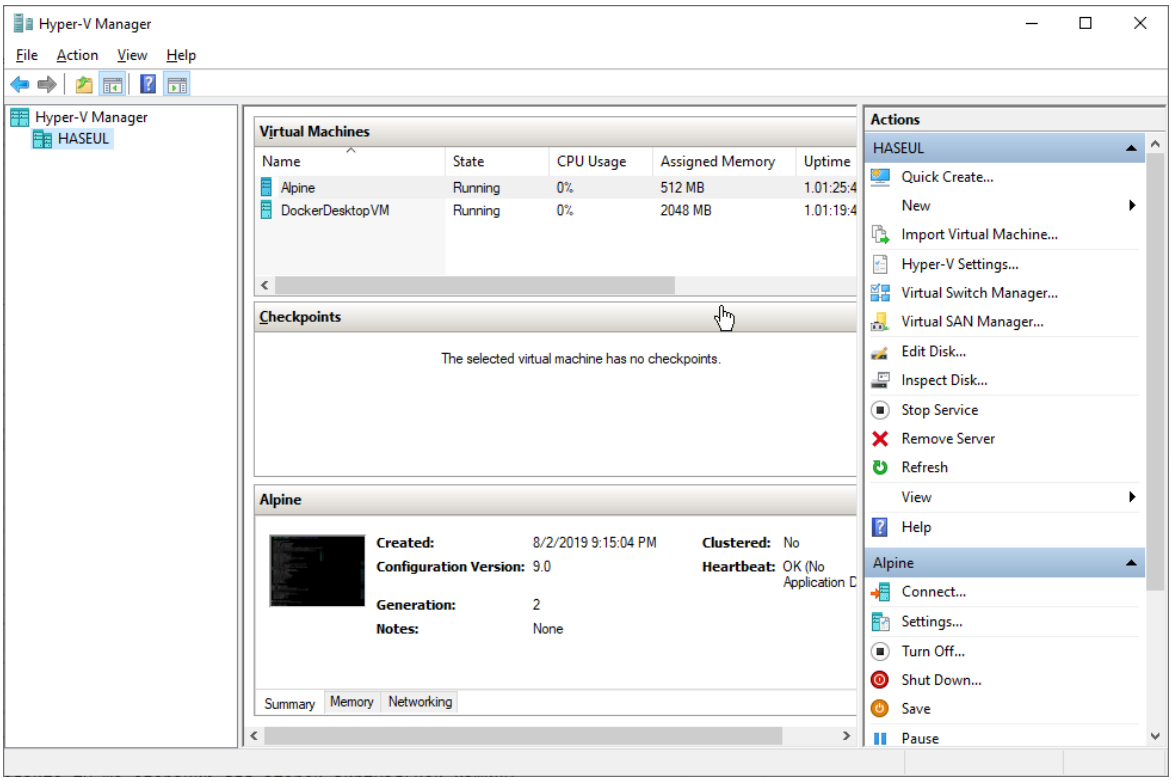
```
# gpasswd -a username libvirt
```

где `username` — ваше имя пользователя.

В этом методическом пособии подразумевается использование гипервизора QEMU, работающего поверх KVM, в libvirt обозначаемого как `qemu:///system`.

Hyper-V

С установкой Hyper-V в системе появится новая оснастка MMC под названием “Hyper-V Manager”. При ее запуске от администратора появится следующее окно:



Скриншот Hyper-V Manager

Hyper-V по умолчанию создает для гостевых виртуальных машин виртуальный NIC с преднастроенным NAT/DHCP под названием «Default Switch», упрощающий настройку подключения к Интернету в свежесозданных гостевых машинах. Им необходимо будет воспользоваться для установки операционных систем в машинах.

Примечание: на некоторых системах Windows 10 при установке Hyper-V не создается свич Default Switch. Это известная нерешенная проблема Windows 10. Если она проявляется у вас, вместо Default Switch создайте вручную свич в режиме бриджинга с физическим устройством и используйте его в тех местах, где пособие указывает использо-

вать свич по умолчанию. Установка сетевых настроек такому свичу в виртуальной машине в таком случае будет зависеть от конфигурации вашей физической сети; сверьтесь с документацией Alpine Linux.

Чтобы не выполнять Hyper-V Manager с правами администратора, можно добавить себя в группу администраторов Hyper-V. Это можно сделать из `compmgmt.msc`. Изменения вступят в силу после релогина.

Установка сетевых настроек

При необходимости нам могут потребоваться различные типы виртуальных сетевых свичей. Таковых существует три:

1. Соединение с физической сетью: сетевой бридж с интерфейсом физического соединения сервера виртуализации. В случае с libvirt он создается средствами bridge-utils.
2. Внутренняя сеть: виртуальный свич, позволяющий коммуникацию между виртуальными машинами и хостом виртуализации (соединение представляется в виде отдельного устройства на хосте).
3. Частная сеть: виртуальный свич, позволяющий коммуникацию только между виртуальными машинами (виртуальная сеть недоступна с хоста). В случае с libvirt это частный случай сети второго типа.

Чаще всего в лабораторных работах мы будем пользоваться вторым видом свича. Для коммуникации с виртуальными машинами создаваемому на хосте сетевому интерфейсу нужно будет присваивать сетевые настройки.

Linux

Так как для Linux-систем существует огромное количество разнородного ПО для конфигурации сети (ifupdown, netplan, NetworkManager, systemd-networkd, встроенная поддержка конфигурации сети в iwd, и другие), установка сетевых настроек и настройка файрволла хоста здесь не рассматривается.

Примечание: настройка файрволла, если она требуется, сводится к включению маскарадинга трафика с виртуального сетевого адаптера на хосте по умолчанию, а также разре-

шению входящего трафика на порты 53 по TCP и UDP для работы DNS и разрешению UDP-пакетов на порт 67 на broadcast-адрес для работы DHCP. libvirtd настроит маскарадинг самостоятельно через iptables (либо iptables-nft, если у вас установлен он).

Сети в libvirt определяются в виде XML-файлов определенного синтаксиса. Примерный файл приведен ниже:

```
<!-- В синтаксисе описания сети наличествует множество параметров, не описанных ниже.
Для информации по ним следует обратиться к документации. -->
<network>
    <!-- Название сети. Может быть любым, будет выводиться в virsh net-list и использоваться для обращения к сети в дальнейшем. -->
    <name>UniLWSwitch</name>
    <!-- UUID, уникальный идентификатор. Должен быть по-настоящему уникален и генерироваться автоматически. В Linux его можно сгенерировать командой uuidgen из пакета
util-linux. -->
    <uuid>33451163-b407-4cdf-b430-f91e8e09b56b</uuid>
    <!-- Директива, включающая форвардинг пакетов от виртуальных машин. Форвардинг
реализуется правилами файрволла. Свойство dev устанавливает интерфейс, на котором
возможен форвардинг (т.е. на других интерфейсах форвардинг производиться не будет).
Свойство mode принимает значения nat и route и означает режим форвардинга, NAT либо
бриджинг с физическим устройством. -->
    <!-- Использование директивы означает, что машинам будет открыт доступ во внешнюю
сеть. -->
    <forward dev='eth0' mode='nat' />
    <!-- Директива, создающая бридж-устройство на хосте. name означает название создаваемого сетевого интерфейса на хосте (т.е. то имя, которое можно видеть через ip
link), delay активирует задержку при пересылке пакетов. -->
    <!-- Использование этой директивы означает, что сеть будет доступна с хоста. -->
    <bridge name='uni0' delay='0' />
    <!-- IP-адрес и маска сети, устанавливаемые при старте сети на хосте сетевому интерфейсу, указанному в bridge. -->
    <ip address="192.168.101.1" netmask="255.255.255.0">
        <!-- Если эта директива добавлена, libvirt запустит инстанс dnsmasq для выдачи IP-адресов по DHCP с заданными ниже настройками. -->
        <dhcp>
            <!-- Диапазон MAC-адресов для выдачи по DHCP. -->
            <range start="192.168.101.100" end="192.168.101.254" />
            <!-- Пример резервирования адресов - машине с таким MAC-адресом присваивается имя хоста и резервируется IP. -->
            <host mac="00:16:3e:e2:ed" name="foo.example.com" ip="192.168.101.10" />
        </dhcp>
    </ip>
</network>
```

Сеть может быть добавлена из этого файла и позднее отредактирована следующими командами:

```
# virsh -c qemu:///system net-define FILENAME.xml
# virsh -c qemu:///system net-edit NETWORKNAME
```

Здесь `FILENAME.xml` — имя файла, а `NETWORKNAME` — название определенной данным файлом сети.

Активировать сеть можно командами `virsh -c qemu:///system net-autostart NETWORKNAME` и `virsh -c qemu:///system net-start NETWORKNAME`. Это соответственно активирует автосоздание сети при запуске libvirtd и запустит сеть сейчас.

Добавить сеть виртуальной машине можно следующей командой:

```
# virsh -c qemu:///system attach-interface --domain VMNAME --type network --model
virtio --live --source NETWORKNAME --persistent
```

Эта команда добавит новый сетевой интерфейс машине `VMNAME` модели `virtio` (специальный тип сетевого адаптера для Linux-гостей, обеспечивающий лучшую производительность, чем эмуляция реального железа), подключенный к сети `NETWORKNAME`. Опции live и persistent предназначены для установки интерфейса соответственно даже если машина уже запущена и навсегда.

Список подключенных к машине сетей можно получить следующей командой:

```
# virsh domiflist
```

Отключение сетевого адаптера производится `virsh detach-interface`.

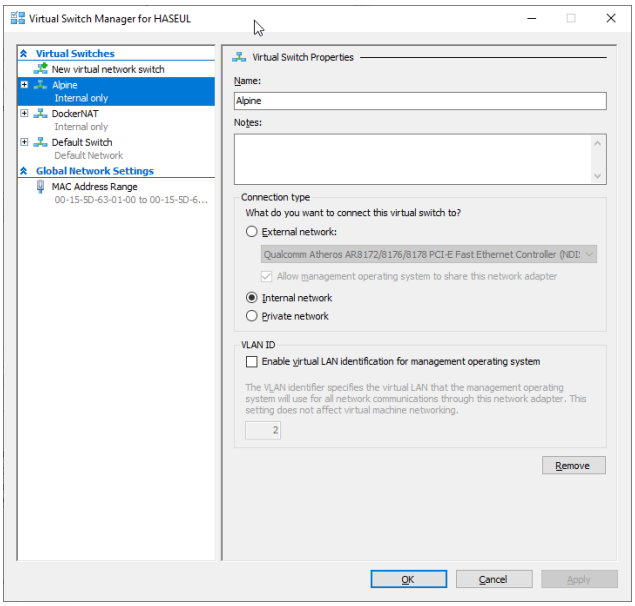
При необходимости тонкого редактирования конфигурации (например, при замене одного сетевого адаптера другим с сохранением настроек) можно воспользоваться текстовым редактором для прямой конфигурации машины из текстового файла:

```
# virsh -c qemu:///system edit VMNAME
```

virsh автоматически проверит синтаксис после сохранения файла. В случае его корректности конфигурация машины будет заменена сохраненной.

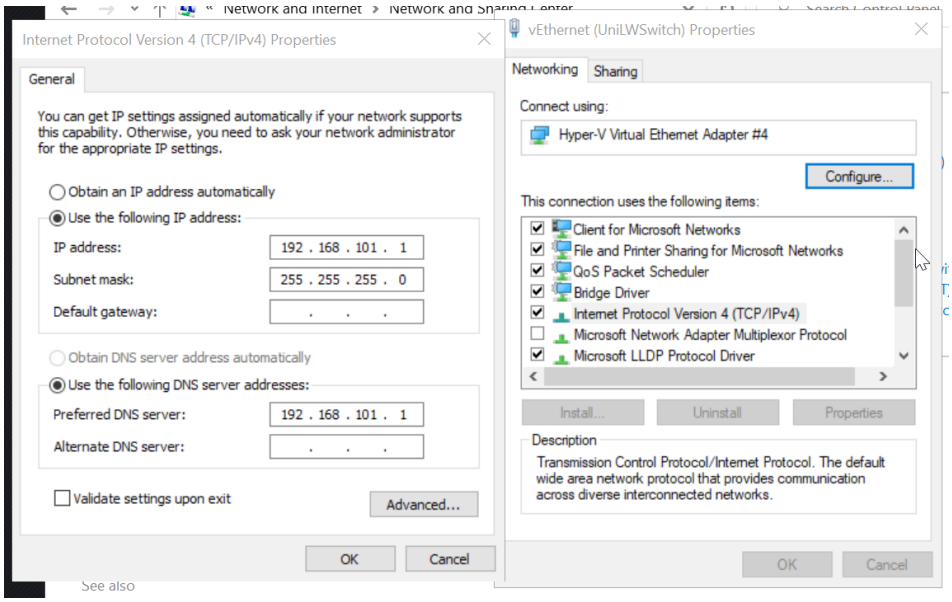
Windows

Виртуальные сетевые свичи могут быть созданы в менеджере виртуальных свичей, доступном из Hyper-V Manager. Так, создание виртуальной сети второго типа может выглядеть следующим образом:



Конфигурация виртуального свича

Установка сетевых настроек виртуальному свичу производится из Control Panel в менеджере сетевых подключений. Примерная настройка сетевых настроек виртуальному интерфейсу может выглядеть следующим образом:



Установка IP-адреса 192.168.101.1 хосту виртуализации

Создание виртуальных машин

libvirt

libvirtd хранит конфигурацию виртуальных машин и сетей в виде XML-файлов в директории `/etc/libvirt`. Для создания новой виртуальной маши-

ны можно воспользоваться программой `virt-install`, указав свойства машины в опциях. Например, создание новой виртуальной машины `jeongyeon` (*Чонён*) с Alpine Linux при уже загруженном образе `alpine3.12.iso` с установкой 1024 МБ оперативной памяти и двух виртуальных процессорных ядер, включением сетевого адаптера по умолчанию модели virtio, а также автоматическим созданием нового виртуального диска в 1 ГБ может выглядеть следующим образом:

```
% virt-install --connect qemu:///system --name jeongyeon –network
network=default,model=virtio --memory 1024 --vcpus 2 --disk size=1 --cdrom images/
alpine-3.12.iso --os-variant generic
```

`virt-install` попытается автоматически запустить программу `virt-viewer`, предоставляя виртуальный дисплей для работы с машиной. С полным списком параметров `virt-install` можно ознакомиться в `man virt-install`. Если `virt-viewer` не установлен, к дисплею виртуальной машины можно подключиться из `virt-manager.`

В виртуальную машину при создании, если не установить соответствующие опции, будет «вставлен» свич по умолчанию, создаваемый libvirtd. На нем преднастроены DHCP и DNS, упрощая настройку сети в виртуальной машине. Работа DHCP и DNS на этом свиче обеспечивается dnsmasq.

Hyper-V

Виртуальная машина создается в Hyper-V Manager. Параметры при создании VM можно выбирать произвольно, лишь убедитесь, что для Alpine достаточно дискового пространства (виртуального диска в 1 ГБ достаточно), а оперативной памяти хотя бы 256 МБ (лучше 512 МБ). Поколение виртуальной машины лучше устанавливать в “Generation 2”. В качестве сетевого адаптера виртуальной машины на время установки ОС выберите “Default Switch” — это NIC по умолчанию, созданный Hyper-V.

В настройках виртуальной машины, во вкладке настроек SCSI-контроллера, добавьте виртуальный DVD-привод и “вставьте” в него загруженный ISO. Выключите Secure Boot в настройках машины. Переставьте приоритет загрузки на DVD-привод. Включите виртуальную машину.

Дополнительные замечания

В лабораторных работах часто описываются команды, которые необходимо выполнить в терминале на некоторой машине. Добавление символа `#` в начале команды означает, что команда должна быть исполнена пользователем `root`. Символ `%` означает, что команда может быть выполнена любым пользователем. Вводить эти символы не нужно.

Для команд, исполняемых в Windows на хосте виртуализации, добавляются префиксы `PS >`, `PS Admin >`, либо просто `>`. Первый и третий означают запуск под пользователем, второй означает, что PowerShell должен быть запущен от имени администратора.

Ссылки на man-страницы обозначаются `pagename(X)`, где `pagename` — название страницы, а `X` — номер секции. Соответствующим вызовом программы man для такой страницы будет `man X pagename`.

Выполнение лабораторных работ

Для каждой лабораторной работы выполняется отчет в письменном виде. Отчет должен включать в себя информацию, указанную в каждой лабораторной работе. Если требуется продемонстрировать результат выполнения некоторой операции, он может быть включен в отчет в виде скриншота выполнения операции либо в любом другом удобном виде.

ЛАБОРАТОРНАЯ РАБОТА №1. УСТАНОВКА ALPINE LINUX

В этой лабораторной работе рассматривается установка Alpine Linux на две виртуальные машины и объединение этих двух машин в одноранговую сеть, а также активация в виртуальных машинах SSH.

Подготовка к выполнению

Загрузите ISO-образ для виртуальных машин с сайта Alpine Linux, нас интересует virt-образ для архитектуры x86_64 последней версии (на момент написания этого текста это 3.12).

Создайте две новых виртуальных машины с именами Taehyung (*Тэхён*) и Nayeon (*Наён*). Для виртуальной машины с Alpine Linux достаточно 512 МБ оперативной памяти и 1 ГБ дискового пространства. Запустите их с загруженного ISO-образа, используя свич по умолчанию.

Установка Alpine Linux

Alpine Linux запустится в обеих машинах и запросит имя пользователя. Введите `root`, по умолчанию в образе Alpine нет пароля.

Вместе с Alpine идет простой скрипт `setup-alpine`, позволяющий быстро развернуть Alpine Linux на любом носителе. Запустим его в обеих машинах:

```
# setup-alpine
```

> Примечание: в Hyper-V иногда не работает DNS внутри виртуальной машины при использовании свича по умолчанию. Причина неполадок неизвестна, но если это проявляется у вас (например, в виде отсутствующего списка зеркал на этапе установки), попробуйте выполнить `echo "nameserver 1.1.1.1" > /etc/resolv.conf`, это установит DNS-сервер Cloudflare основным.

Выберите раскладку клавиатуры для США (`us - us` в первых двух диалогах). Там, где есть значения по умолчанию, они обозначены в квадратных скобках. Выберите адаптер `eth0`, выберите настройку по DHCP (по умолчанию). В поле настройки прокси укажите нужное, если оно требуется в вашей сети. Выбор NTP-клиента не имеет значения, выберем `chronyd`. При предложении выбрать зеркало выберите опцию выбора лучшего зеркала из предложенных

(выбор наиболее быстрого зеркала займет некоторое время). В качестве SSH-сервера выберите OpenSSH. При выборе устройства для установки выберите диск из предложенных (в нашем случае с одним диском это будет `sda`), а в качестве типа диска укажите `sys`. Установите пароль пользователя `root`.

Примечание: при использовании KVM иногда получение сетевых настроек по DHCP может не работать, если в виртуальные машины воткнут виртуальный сетевой адаптер типа, отличного от virtio.

После завершения установки скрипт предложит перезагрузить систему. “Вытащите” ISO Alpine Linux из виртуального дисковода, если используется Hyper-V. Перезагрузитесь.

Первичная настройка

Для проведения лабораторных работ нам понадобится дополнительный виртуальный сетевой адаптер для коммуникации между хостом и гостями.

При создании свича установим IP-адрес хоста в виртуальной сети, укажем адрес `192.168.101.1` и маску `255.255.255.0`. В настройках обеих VM добавим по сетевому адаптеру, включим туда свежесозданный свич.

Включим виртуальные машины. Войдем под логином `root` и указанным нами паролем. Убедимся, что соединение с сетью работает, посредством `# ping -c 1 1.1.1.1`; это пошлет один тестовый ICMP-пакет на DNS-сервер Cloudflare. Если интернет работает, все настроено правильно.

Примечание: в Alpine Linux Busybox используется и для конфигурации сети в том числе, встроенные в Busybox утилиты частично совместимы с пакетом ifupdown. Данные инструкции по настройке сети совместимы с дистрибутивами, использующими для конфигурации сети ifupdown.

Установка текстового редактора

Для выполнения последующих лабораторных работ нам понадобится текстовый редактор. Знакомые с текстовым редактором vi могут пользоваться встроенной в Busybox реализацией vi (запускается командой `vi`). Если же вы

не знакомы с этим текстовым редактором, то установите текстовый редактор GNU nano командой:

```
# apk update && apk add nano
```

Инструкцию по тому, как им пользоваться, можно прочитать (на русском языке) здесь. Редактирование файлов производится командой:

```
% nano FILENAME
```

Конфигурация сети

Откроем файл `/etc/network/interfaces` и настроим свежедобавленный сетевой адаптер. Приведем его к следующему виду:

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet dhcp
    metric 0
auto eth1
iface eth1 inet static
    address 192.168.101.2
    netmask 255.255.255.0
    metric 1
```

`eth0` здесь — свич по умолчанию для доступа в Интернет (настроен на получение сетевых настроек по DHCP), `eth1` — новый адаптер для виртуальной ЛВС. Посредством `metric eth1` искуственно понижен приоритет (более высокая метрика означает более низкий приоритет). Для второй виртуальной машины адрес должен быть `192.168.101.3` или другой в нашей подсети, не совпадающий с уже использованными выше адресами.

После редактирования файла поднимем интерфейс `eth1` командой:

```
# ifup eth1
```

Убедиться в корректности настроек можно командой:

```
# ip addr
```

Программа покажет текущие IP-адреса, подсети и маску подсети для каждого интерфейса, а также MTU, MAC-адреса, и другую информацию, которая нам сейчас не нужна. Именно этой командой можно выяснить, например, физический адрес сетевой карты.

Проверим доступность машин друг из друга.

Примечание: файрволл на хосте может блокировать входящие ICMP-пакеты. Для того, чтобы проверить связность, может понадобиться разрешить входящие ICMP-пакеты из подсети `192.168.101.0/24`.

Из второй виртуальной машины проверим связность с третьей:

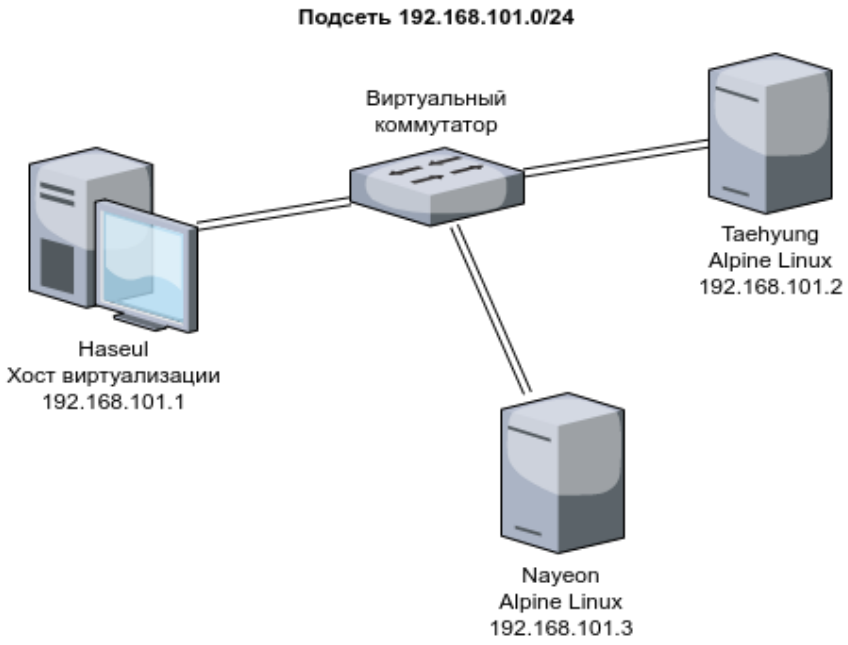
```
% ping -c 1 192.168.101.3
```

и наоборот (`192.168.101.2`). Пакеты должны дойти. Проверим таким же образом связность с хостом, `192.168.101.1`. Из хоста проверим связность с виртуальными машинами. Пакеты также должны дойти.

Утилита `ping` по умолчанию попытается разрешить имя хоста, к которому идет обращение. Для этого также можно использовать `nslookup`. Установить размер ее пакета в реализации Busybox можно ключом `-s SIZE`, где вместо `SIZE` подставить нужный размер пакета в байтах (по умолчанию `56`).

Текущее имя хоста можно узнать командой `hostname`, а установить посредством записи нового имени в `/etc/hostname` и `/etc/hosts`.

Здесь и далее «серверная» виртуальная машина — Taehyung (*Тэхён*). «Клиентская» виртуальная машина названа Nayeon (*Наён*), хост виртуализации — Haseul (*Хасыль*). Результирующая топология выглядит так:



Топология сети

Перезапуск компьютера при смене сетевых настроек не требуется. Для смены символьного имени компьютера перезапуск будет необходим на Windows, но не на Linux (командой `hostname`).

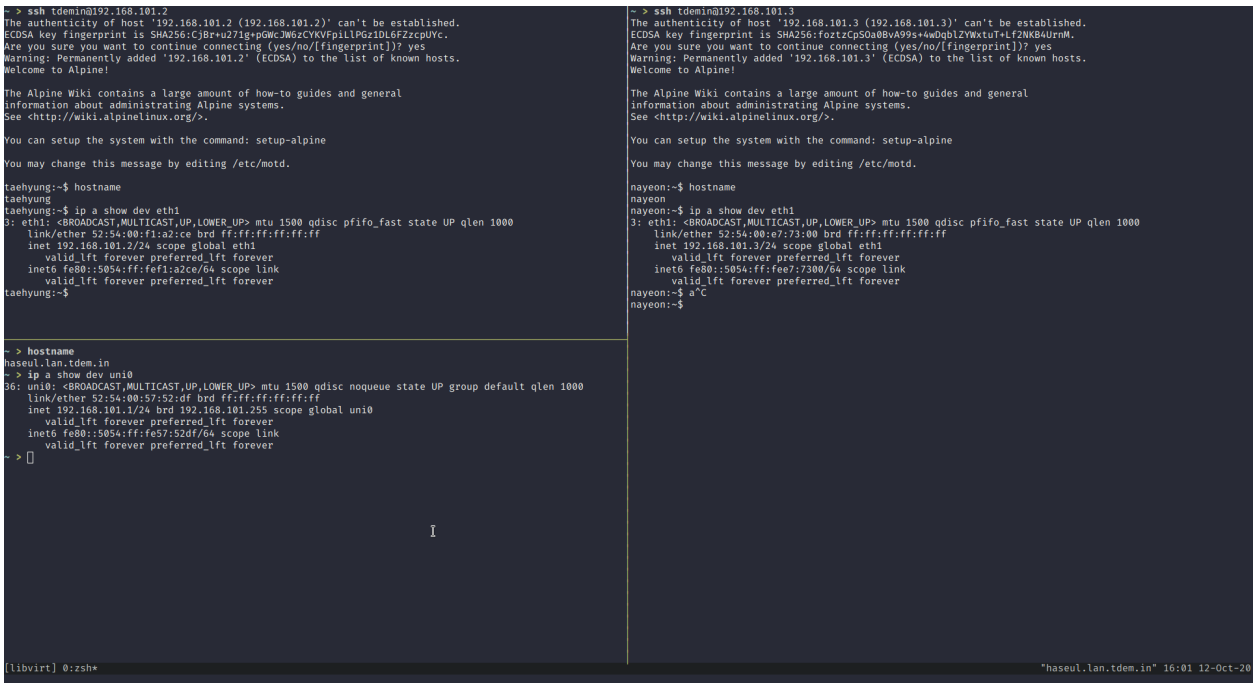
Добавление пользователя

Создадим нового пользователя, группу для него, а также позволим для этого пользователя логин по протоколу SSH. Все это делается для повышения удобства работы с системой, чтобы не приходилось работать из-под пользователя с максимальными правами и из дисплея VM.

Создадим пользователя `tdemin` (подставьте вместо `tdemin` желаемое имя пользователя, при запросе пароля введите желаемый). Включим его в группу `wheel`, что позволит нам входить с правами суперпользователя, не завершая свой сеанс:

```
# adduser tdemin -h /usr/home/tdemin
# adduser tdemin wheel
```

Теперь вход с правами суперпользователя из пользовательского сеанса возможен по команде `% su -`. Можно подключиться любым SSH-клиентом (для Windows это PuTTY, для Linux — OpenSSH) по IP-адресу виртуальной машины как `tdemin@192.168.101.2` (либо `192.168.101.3` в случае второй машины). Таким образом, в результате мы должны получить конфигурацию, которая отражена на скриншоте ниже:



Подключение по SSH к виртуальным машинам

Отчет по работе

В отчет требуется поместить следующее:

1. результаты пинга от всех трех машин друг до друга;
2. результаты выполнения команд `ip addr` и `hostname` в виртуальных машинах;
3. настройки виртуальной сети (XML-файл для libvirt).

Контрольные вопросы

1. Как узнать физический и IP-адрес компьютера?
2. Нужно ли перезапускать компьютер, чтобы изменения вступили в силу, если изменяются следующие параметры:
    a. настройки стека TCP/IP;

b. имя компьютера?

3. Как с помощью утилиты `ping` определить достижимость узла? Какая информация, полученная при использовании утилиты `ping`, служит ответом о достижимости узла?
4. Как определить IP-адрес удаленного узла, зная только его символьное имя?
5. Как изменить размер пакета утилиты `ping`?
6. Какая утилита определяет имя текущего узла?

ЛАБОРАТОРНАЯ РАБОТА №2. МАРШРУТИЗАЦИЯ В IP-СЕТЯХ

Создание новой подсети

В данном занятии новой подсетью станет `192.168.102.0/24`, в ней будут находиться Taehyung и Nayeon, но не Haseul.

Linux (посредством, например, использования Quagga) позволяет использование динамических протоколов маршрутизации RIP, OSPF, BGP, ISIS и Babel. Теоретически посредством использования соответствующего ПО можно использовать и другие протоколы динамической маршрутизации.

Создадим новую частную сеть без возможности коммуникации с хостом, назовем ее, например, UniLWSwitch2. Добавим новый сетевой адаптер машине Taehyung и включим его вместо UniLWSwitch в Nayeon.

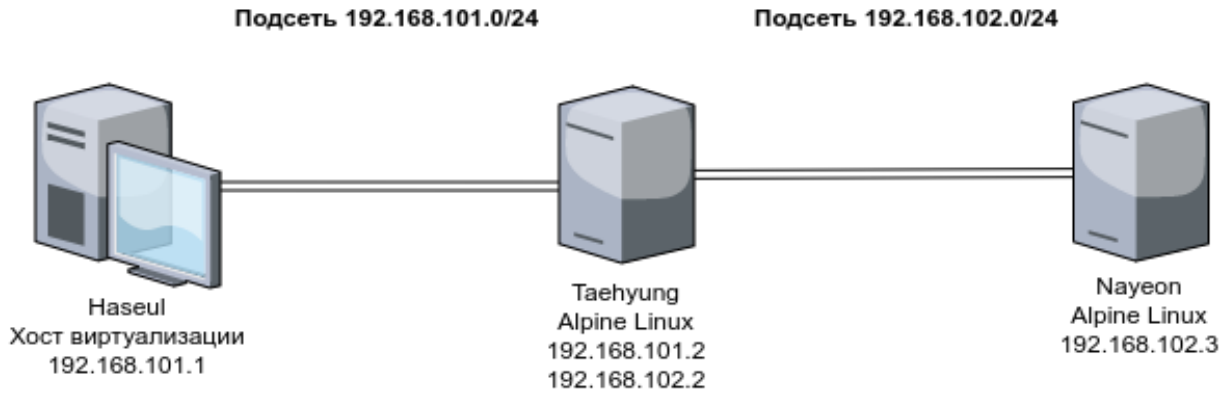
Примечание: выполнение данной лабораторной работы не позволяет пользоваться удаленным соединением по SSH. Для выполнения работы необходимо подключиться к виртуальному дисплею машины.

Проверим, что адаптер был добавлен на Taehyung, выполнив команду `% ip link`. В выводе должен появиться новый сетевой адаптер `eth2` без установленного IP-адреса и маски подсети.

В данном примере Taehyung будет также иметь адрес `192.168.102.2/24` в подсети, недоступной хосту. Запустим сетевой адаптер командой `ifup`, после чего проверим корректность конфигурации командой `ip addr`.

Nayeon в сети между виртуальными машинами будет иметь адрес `192.168.102.3/24`, Taehyung — `192.168.102.2/24`. В сети же, доступной хосту виртуализации, Haseul будет иметь адрес `192.168.101.1/24`, а Taehyung `192.168.101.2/24`.

Желаемая топология сети должна выглядеть примерно так:



Желаемая топология сети

Настройка клиента для лабораторной работы

Конфигурация, произведенная при установке адаптера `eth0` в Taehyung и Nayeon, устанавливает Haseul в качестве шлюза по умолчанию. Поэтому перед тестами необходимо отключить сетевой адаптер `eth0` на обеих машинах командой:

```
# ifdown eth0
```

Конфигурация сети и шлюз по умолчанию будут снова заданы при следующей перезагрузке.

> Примечание: в ходе выполнения лабораторной работы имена сетевых интерфейсов на ваших машинах могут отличаться от представленных в описании работы. Имена присваиваются автоматически, поэтому проверяйте соответствие MAC-адреса интерфейса, получаемого командой `% ip link`, и соответствующей ему виртуальной сети в конфигурации машины. `virsh domiflist` сильно упростит жизнь.

Выключим адаптер `eth1` перед реконфигурацией на Nayeon: `# ifdown eth1`. Установим ему такие настройки через файл `/etc/network/interfaces`:

```
auto eth1
iface eth1 inet static
    address 192.168.102.3
    netmask 255.255.255.0
    gateway 192.168.102.3
    metric 1
```

Поднимем его заново с этими настройками. Проверим корректность конфигурации командой `ip addr`. У адаптера `eth0` должен пропасть IP-адрес и мас-

ка подсети. Проверим невозможность подключения с Nayeon к Haseul по старому адресу: `% ping -c 1 192.168.101.1`. Мы должны получить сообщение “Network is unreachable”. В таблице маршрутизации (`% ip route`) должен остаться один маршрут в сеть `192.168.102.0/24` на интерфейсе `eth1`.

С Haseul адрес `192.168.102.3` также не должен быть доступен, маршрутов по команде `PS > route print 192.168.102*` (если ваш хост на Windows) не должно выводиться.

Настройка маршрутов

Любая Linux-система может работать в качестве роутера, необходимо лишь задать соответствующие настройки. Активируем на Taehyung пересылку IP-пакетов, задав соответствующую опцию ядра ОС:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
# echo "net.ipv4.ip_forward = 1" >> /etc/sysctl.conf
```

Приведем интерфейсы `eth1` и `eth2` в `/etc/network/interfaces` на Taehyung к следующему виду:

```
auto eth1
iface eth1 inet static
    address 192.168.101.2
    netmask 255.255.255.0
    gateway 192.168.101.2
    metric 1
auto eth2
iface eth2 inet static
    address 192.168.102.2
    netmask 255.255.255.0
    gateway 192.168.102.2
    metric 2
```

Добавим новый маршрут на Nayeon:

```
# ip route add 192.168.101.0/24 via 192.168.102.2
```

Аналогично добавим новый маршрут на Haseul до 192.168.102.0/24 через 192.168.101.2. Если ваш хост — Windows, то это делается так:

```
PS Admin > route add 192.168.102.0 mask 255.255.255.0 192.168.101.2
```

Примечание: файрволл может блокировать ICMP-пакеты по умолчанию. В таком случае следует разрешить ICMP-пакеты, идущие из `192.168.102.0/24`.

Переподнимем интерфейсы на Taehyung:

```
# ifup eth1
# ifup eth2
```

Проверим связность между компьютерами:

```
nayeon # traceroute -I 192.168.101.1
```

Мы должны увидеть, что пакет идет через `192.168.102.2` и через `192.168.101.2` соответственно (адреса роутера в различных подсетях).

```
nayeon:~$ ping 192.168.101.1
PING 192.168.101.1 (192.168.101.1): 56 data bytes
64 bytes from 192.168.101.1: seq=0 ttl=42 time=0.886 ms
64 bytes from 192.168.101.1: seq=1 ttl=42 time=0.447 ms
^C
--- 192.168.101.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.447/0.666/0.886 ms
nayeon:~$ traceroute 192.168.101.1
traceroute to 192.168.101.1 (192.168.101.1), 30 hops max, 46 byte packets
 1  192.168.102.2 (192.168.102.2)  0.205 ms  0.596 ms  0.346 ms
 2  192.168.101.1 (192.168.101.1)  65.521 ms  0.621 ms  0.317 ms
nayeon:~$ _
```

Успешная проверка связности между компьютерами

Получилась схематично такая топология:

```
Nayeon: 192.168.102.3
    <-> 192.168.102.0/24
        <-> Taehyung (192.168.102.2)
        <-> Taehyung (192.168.101.2)
    <-> 192.168.101.0/24
Haseul: 192.168.101.1
```

Теперь пакеты адресам из `192.168.101.0/24` с Nayeon будут пересылаться через Taehyung, как и пакеты адресам `192.168.102.0/24` с Haseul.

Записи в таблице маршрутизации хранятся до первой перезагрузки. При необходимости можно добавлять запись в таблицу при каждой загрузке. Это можно сделать, например, записью соответствующей команды `ip route` в поле `post-up` соответствующего интерфейса в файле `/etc/network/interfaces`. На Windows для этого в соответствующую команду `route add` нужно добавить ключ `-p`.

После выполнения работы

Включим снова адаптер UniLWSwitch в Nayeon вместо UniLWSwitch2. Удалим запись из таблицы маршрутизации с Haseul. На Linux это делается следующим образом:

```
# ip route delete 192.168.102.0/24
```

На Windows:

```
PS Admin > route delete 192.168.102.0
```

Изменим сетевые настройки обратно. Перезагрузим виртуальные машины. Taehyung снова должна быть доступна по SSH с `192.168.101.2`, а Nayeon с `192.168.101.3`.

Отчет по работе

В отчет требуется поместить следующее:

1. неудачный результат соединения после переноса Nayeon в другую подсеть /24;
2. сетевую конфигурацию Taehyung после перенастройки сетевых интерфейсов;
3. результаты трассировки трафика с Nayeon до Haseul и обратно после перенастройки;
4. конфигурацию сети для свича UniLWSwitch2 (для libvirt — соответствующий XML-файл).

Контрольные вопросы

1. Назовите протоколы маршрутизации, поддерживаемые ПО под Linux.
2. Что такое таблица маршрутизации?
3. Какие записи создаются в таблице маршрутизации по умолчанию?
4. Чем отличаются возможности Linux-систем в маршрутизации?

ЛАБОРАТОРНАЯ РАБОТА №3. DHCP-СЕРВЕР: УСТАНОВКА И УПРАВЛЕНИЕ

Используемое ПО

В качестве DHCP-сервера в данной лабораторной работе будет использоваться dnsmasq. dnsmasq — свободный DHCP- и DNS-сервер для маленьких сетей, распространяемый под лицензией GNU GPL v2 или v3 на выбор пользователя. dnsmasq поддерживает и другие протоколы (TFTP, PXE), но это для выполнения данной работы не имеет значения.

Предварительная подготовка

Taehyung в данной лабораторной работе должна быть настроена так же, как и в предыдущей лабораторной работе: три сетевых адаптера, `eth0` — свич по умолчанию, `eth1` — свич UniLWSwitch, `eth2` — свич UniLWSwitch2. Настройки в файле `/etc/network/interfaces` для Taehyung будут совпадать с настройками из предыдущей лабораторной работы. Свич `eth0` отключать необязательно.

> Примечание: как и в предыдущей лабораторной работе, имена сетевых устройств могут отличаться при выполнении работы и не являются существенными.

Установка dnsmasq

На Taehyung выполним следующую команду:

```
# apk update && apk add dnsmasq
```

Откроем файл конфигурации dnsmasq, `/etc/dnsmasq.conf`. Его синтаксис в основном представляет собой обычный вид `ключ=значение`. Комментарии (игнорируемый текст) обозначаются символом `#`. После редактирования конфигурации dnsmasq необходимо перезапускать.

Каждая опция конфигурационного файла хорошо задокументирована (на английском языке), с примерами использованиями и случаями, когда каждая опция может пригодиться в реальной жизни. Мы отредактируем следующие:

1. `port` раскомментируем и поменяем значение на `53`, это включит встроенный в dnsmasq DNS-сервер, он будет слушать на порту 53.
2. `listen-address` раскомментируем и установим в `192.168.102.2`. Это заставит dnsmasq слушать только на этом адресе в подсети `192.168.102.0/24` из предыдущей лабораторной работы.
3. `dhcp-range` раскомментируем и установим в `192.168.102.3,192.168.102.254,12h`, это заставит dnsmasq:
    a. Выдавать адреса только из пула `192.168.102.3` по `192.168.102.254`.
    b. Выдавать адреса на 12 часов (время аренды адреса).

Сделаем так, чтобы dnsmasq запускался при старте системы, после чего запустим его:

```
# rc-update add dnsmasq
# service dnsmasq start
```

Настройка клиента

Для тестирования выдачи адресов включим в Nayeon сетевой адаптер UniLWSwitch2 вместо UniLWSwitch так же, как мы это делали в лабораторной работе №2. Для этого, конечно, понадобится разорвать текущее SSH-соединение.

Приведем настройки интерфейса `eth1` в `/etc/network/interfaces` к виду:

```
auto eth1
iface eth1 inet dhcp
    metric 1
```

Это заставит Nayeon получать IP-адрес в сети автоматически по протоколу DHCP. Nayeon будет искать в сети DHCP-сервер автоматически при запуске сетевого интерфейса.

Проверим корректность конфигурации. `ip addr`, выполненный на Nayeon, должен отдавать IP-адрес в подсети `192.168.102.0/24` в пуле от `192.168.102.3` до `192.168.102.254`.

Для проверки конфигурации DNS на Nayeon нам нужна будет утилита dig из пакета программ, поставляемых с DNS-сервером bind. Установим ее:

```
# apk add bind-tools
```

На Taehyung добавим запись в `/etc/hosts` с каким-нибудь адресом в конец, например: `192.168.102.2 taehyung.someaddress.net`. Перезапустим dnsmasq, поскольку он считывает записи из hosts при запуске:

```
# service dnsmasq restart
```

На Nayeon выполним следующую команду:

```
% dig taehyung.someaddress.net
```

Она должна вернуть запись (в Answer Section) примерно такого вида:

```
taehyung.someaddress.net. IN A 192.168.102.2
```

После этого запустим `% ping -c 1 taehyung.someaddress.net`. Мы должны увидеть, что пакеты идут на `192.168.102.2`.

Продвинутая настройка

Резервирование адресов

Зарезервируем для Nayeon постоянный адрес `192.168.102.10`.

Выясним MAC-адрес адаптера `eth1` на Nayeon командой `% ip link`. На Taehyung откроем `/etc/dnsmasq.conf`, добавим в него такую директиву:

```
dhcp-host=00:15:5d:01:05:12,nayeon,192.168.102.10
```

Эта директива состоит из трех частей: MAC-адреса (вставьте тот, что получили командой `% ip link` на Nayeon), присваиваемого имени хоста и резервируемого IP-адреса. Они могут располагаться в произвольном порядке (и могут опускаться при необходимости); также может указываться время, на которое выдается адрес. Перезапустим dnsmasq.

Перезапустим интерфейс `eth1` на Nayeon:

```
# ifdown eth1 && ifup eth1
```

Убедимся в том, что Nayeon был выделен IP-адрес `192.168.102.10`, командой `% ip addr`.

Адреса можно резервировать и вне области выдачи адресов DHCP-сервера; так, можно поменять `dhcp-range` на Taehyung так, чтобы выдача адресов про-

водилась вплоть до `192.168.102.100`, а для Nayeon зарезервировать `192.168.102.101`.

Удалим директиву `dhcp-host` из файла конфигурации и перезапустим dnsmasq на Taehyung командой `# service dnsmasq restart`. Перезапустим адаптер `eth1` на Nayeon командой `# ifdown eth1 && ifup eth1`. Nayeon снова должна получать IP-адрес из диапазона.

Журналирование

В dnsmasq журналирование действий (выдачи адресов, подключений, запуска) включено по умолчанию. Посмотреть его можно, например, так:

```
% grep dnsmasq /var/log/messages | less
```

После выполнения

Откатим виртуальные машины к их предыдущему состоянию. Включим в Nayeon адаптер UniLWSwitch вместо UniLWSwitch2.

Отчет по работе

В отчет требуется поместить следующее:

1. результат выполнения `ip addr` на Nayeon после настройки DHCP-сервера на Taehyung;
2. результат выполнения A-запроса с помощью dig;
3. директиву, которой был зарезервирован адрес для Nayeon, а также результат выполнения `ip addr` после резервирования;
4. часть журнала работы dnsmasq.

Контрольные вопросы

1. Для чего предназначен протокол DHCP?
2. Что означает термин «аренда адреса»?
3. Для каких компьютеров сети следует применять резервирование адреса?
4. Какого вида IPv4-адрес шлюза обычно используют для DHCPv4-сервера?

5. Установите соответствия между протоколами и выполняемыми ими функциями:

Протоколы	**Функции протоколов**
1. DHCP	а. Отображение IPv4-адресов на MAC-адреса.
2. DNS	б. Присвоение IP-адресов клиентским компьютерам.
3. ARP	в. Отображение доменных имен на IP-адреса.

ЛАБОРАТОРНАЯ РАБОТА №4. DNS-СЕРВЕР: УСТАНОВКА И УПРАВЛЕНИЕ

Используемое ПО

В данной лабораторной работе вместо dnsmasq мы будем использовать DNS-сервер bind. dnsmasq не предназначен для работы полноценным DNS-сервером, а лишь позволяет облегчить конфигурацию сети в малых сетях.

bind — свободное программное обеспечение, распространяемое ISC под лицензией ISC. bind может работать и как авторитативный DNS-сервер, и как рекурсивный DNS-резолвер.

В данной работе мы будем настраивать bind как авторитативный DNS-сервер для домена `bpoi-love.com`.

Предварительная подготовка

Сетевая конфигурация виртуальных машин должна быть аналогична конфигурации из лабораторной работы №1. Если вы сохранили на машине Taehyung dnsmasq после предыдущей лабораторной работы, его рекомендуется отключить (`# rc-update del dnsmasq && service dnsmasq stop`) либо отключить в его конфигурации DNS-сервер (`port=0`).

Установка bind

Установим bind на Taehyung и скопируем заготовку конфигурационного файла для настройки авторитативного DNS-сервера:

```
# apk update && apk add bind
# cp /etc/bind/named.conf.authoritative /etc/bind/named.conf
```

Отредактируем файл `/etc/bind/named.conf`. Он имеет иерархическую структуру.

Установим `listen-on` в `192.168.101.2`, это заставит bind слушать на адресе, доступном и Haseul, и Nayeon. Отредактируем пример директивы `zone` в конце файла следующим образом:

```
zone "bpoi-love.com" IN {
    type master;
    file "/etc/bind/zones/bpoi-love.com";
};
```

Создадим директорию `/etc/bind/zones`:

```
# mkdir -p /etc/bind/zones
```

Отредактируем в нем файл `/etc/bind/zones/bpoi-love.com` (файл DNS-зоны в формате bind), запишем в него следующее (комментарии после символа `;` писать не нужно):

```
$TTL 1h
; TTL по умолчанию для записей без него
; символ @ раскрывается в домен зоны
@ IN 3600 SOA ns.bpoi-love.com. (
    ; SOA, start of authority, запись требуется для любой зоны, в ней
    ; указывается главный сервер имен и другие детали, IN означает, что
    ; запись используется в Интернете
    bpoi-love.com. ; адрес ответственного за зону
    1 ; номер ревизии, можно делать произвольным, но при редакт. менять
    3600 ; время (в секундах) перед сверкой ревизии, нужно для других серверов
    600 ; время ожидания перед повторными попытками обновить зону после провала, нуж-
но для других серверов
    604800 ; через это время записи из вторичных зон будут считаться недействительны-
ми
    1800 ; на это время клиент должен кэшировать неудачные попытки разрешения
    )
; NS-запись, указывает на сервер имен, их может быть несколько
@ IN 86400 NS ns.bpoi-love.com.
; A-запись, устанавливает соответствие bpoi-love.com. адресу 192.168.101.2
; RFC 1035 ограничивает длину 63 символами, полного домена 255
@ IN 86400 A 192.168.101.2
; A-запись для предыдущей NS-записи
ns IN 86400 A 192.168.101.2
; A-записи для наших используемых в локальной сети компьютеров
haseul IN 86400 A 192.168.101.1
nayeon IN 86400 A 192.168.101.3
taehyung IN 86400 A 192.168.101.2
```

Запустим bind и включим его в автозапуск при старте системы:

```
# service named start
# rc-update add named
```

Конфигурация клиента

На Nayeon в `/etc/resolv.conf` добавим строчку `nameserver 192.168.101.2`. В файл `/etc/network/interfaces` в секцию адаптера `eth1` добавим запись `dns-nameservers 192.168.101.2`.

После этого можно начинать тестировать разрешение имен:

```
% ping -c 1 taehyung.bpoi-love.com
% nslookup haseul.bpoi-love.com 192.168.101.2
% dig @192.168.101.2 NS bpoi-love.com
```

Здесь у dig указаны сервер имен, с которого делается поиск, и тип требуемой записи.

Реверс-зона

Добавим новую зону в конфигурацию bind (`/etc/bind/named.conf`) на Taehyung:

```
zone "101.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/zones/101.168.192.in-addr.arpa";
};
```

В файл `/etc/bind/zones/101.168.192.in-addr.arpa` запишем следующее:

```
$TTL 86400
@ IN SOA ns.bpoi-love.com. bpoi-love.com. (
        1 ; ревизия
        86400 ; интервал обновлений зоны
        86400 ; через этот интервал можно заново попытаться обновить зону
        604800 ; время, через которое данные зоны на вторичных серверах будут счи-
таться недействительными
        86400 ; TTL по умолчанию для отказов
        )
@ IN NS ns.bpoi-love.com.
; реверс-записи, PTR, определим для каждой A-записи
; это очень важно для, например, почтовых серверов
@ IN PTR ns.bpoi-love.com.
@ IN PTR bpoi-love.com.
1 IN PTR haseul.bpoi-love.com.
2 IN PTR taehyung.bpoi-love.com.
3 IN PTR nayeon.bpoi-love.com.
```

Перезапустим bind:

```
# service named restart
```

Проверим реверс-запись на Nayeon: `% dig @192.168.101.2 -x 192.168.101.2`. dig должен вернуть `2.101.168.192.in-addr.arpa 86400 IN PTR taehyung.bpoi-love.com`.

hosts-файл

Для задания соответствия имен хостов их IP-адресам (аналог A/AAAA-записи) можно использовать файл `/etc/hosts`. По умолчанию его содержимое предпочитается любым другим способам разрешения имен, это поведение настраивается через NSS. Записи в нем могут находиться в любом порядке.

Запишем на Nayeon в `/etc/hosts` следующее:

```
192.168.101.5 taehyung taehyung.bpoi-love.com
```

Это установит разрешение двух имен хостов в `192.168.101.2`. Допускаются и записи адресов IPv6.

Отчет по работе

В отчет требуется поместить следующее:

1. содержимое файлов прямой и обратной зоны;
2. результаты разрешения имен с использованием ping, nslookup и dig;
3. результаты разрешения имен после добавления конфликтующей записи в файл hosts на любой машине.

Контрольные вопросы

1. Для чего нужны прямые и обратные DNS-зоны?
2. Опишите назначение компонентов DNS: зона, сервер имен, доменное пространство имен.
3. Назовите основные правила наименования доменов.
4. Какова максимально допустимая длина имени домена?
5. Какова максимально допустимая длина FQDN?
6. Зачем могут быть нужны несколько серверов имен?
7. Зачем нужны утилиты dig / nslookup?
8. Можно ли одному IP-адресу сопоставить несколько имен?
9. Для чего используется файл `/etc/hosts`?
10. Имеет ли значение порядок записей в файле `/etc/hosts`?

ЛАБОРАТОРНАЯ РАБОТА №5. ОСНОВНЫЕ СЕТЕВЫЕ СЕРВИСЫ

Используемое программное обеспечение

В данной лабораторной работе используются свободный MTA Postfix и ПО сетевой файловой системы NFS. Также используются свободные же mailx и msmtp для отправки и чтения почты.

Конфигурация машин

Аналогична конфигурации машин из лабораторной работы №4.

Установка необходимого ПО

Установим на Taehyung необходимые пакеты:

```
# apk add nfs-utils postfix mailx
```

Установим на Nayeon пакет, нужный для использования NFS, а также msmtp для отправки почты по протоколу SMTP:

```
# apk add nfs-utils msmtp
```

> Примечание: в данной работе подразумевается, что был создан пользователь, отличный от суперпользователя, а также была произведена настройка, описанная в главе «Дополнительная настройка» лабораторной работы №1.

Настройка NFS

NFS — сетевая файловая система, позволяющая обмен файлами между несколькими компьютерами. При ее использовании файловая система удаленного компьютера монтируется как директория на другом компьютере.
NFS никак не защищает пользовательский трафик, а аутентификация пользователей рудиментарна и производится лишь по IP-адресам/именам хостов. При работе с NFS не в локальных вычислительных сетях настоятельно рекомендуется использовать VPN для защиты трафика. Для разделения привилегий используются традиционные Unix-права доступа/владельца/группы на чтение/запись/исполнение. Если идентификаторы пользователей на клиенте и сервере NFS не совпадают, то есть возможность, что доступ к файлам полу-

чит другой пользователь (по этой причине NFS является не очень удачным выбором, если клиент и сервер администрируются разными людьми).

Демон NFS в Alpine Linux настраивается посредством редактирования файла `/etc/conf.d/nfs`. В этом файле задаются опции, передаваемые при запуске демонам, входящим в состав NFS. В исходном состоянии сервер NFS уже преднастроен и готов к работе.

Создадим директорию для хранения файлов Nayeon:

```
taehyung # mkdir -p /srv/nfs/nayeon
taehyung # chown -R tdemin:tdemin /srv/nfs/nayeon/
```

Отредактируем файл `/etc/exports` и приведем его к следующему виду:

```
/srv/nfs/nayeon 192.168.101.3(rw,no_root_squash)
```

Включим автозапуск ПО NFS при загрузке системы и запустим его сейчас на обеих машинах:

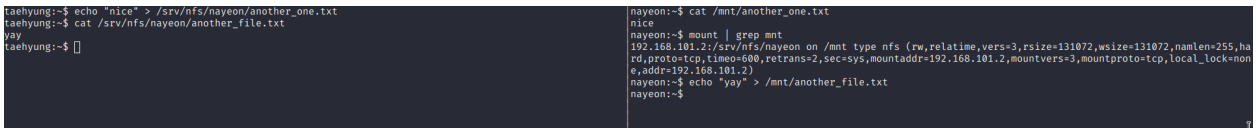
```
taehyung # rc-update add nfs
taehyung # service nfs start
```

Смонтируем сетевую файловую систему на Nayeon:

```
nayeon # mount -t nfs 192.168.101.2:/srv/nfs/nayeon /mnt
```

Создадим произвольный файл в этой директории, запишем туда что-либо, после чего прочитаем его с другой машины:

```
tdemin@nayeon % echo "Some text" >> /mnt/file.txt
tdemin@taehyung % cat /srv/nfs/nayeon/file.txt
```



Использование совместной директории

Включение автоматического монтирования

Активируем монтирование удаленных файловых систем на Nayeon:

```
nayeon # rc-update add nfsmount
```

Размонтируем файловую систему на Nayeon:

```
nayeon # umount /mnt
```

Добавим следующую строчку в файл `/etc/fstab`, содержащий информацию об известных файловых системах (соответственно монтируемое устройство, каталог, в который происходит монтирование, тип файловой системы, опции монтирования (`rw` — для чтения и записи), а также опции проверки файловой системы при загрузке):

```
192.168.101.2:/srv/nfs/nayeon /mnt nfs rw 0 0
```

Выполним монтирование заново короткой командой (остальные аргументы будут считаны автоматически из `/etc/fstab`):

```
nayeon # mount /mnt
```

Другие пользователи

При работе с файлами берутся во внимание права на них, установленные на сервере. При этом, поскольку работа идет с идентификаторами пользователей, а не с их именами, имена пользователей (и их реальные владельцы) могут различаться при одинаковых идентификаторах.

На Taehyung создадим дополнительного пользователя efrich с UID, равным 1001:

```
taehyung # adduser -D -u 1001 -g "" efrich
```

Сменим владельца одного из файлов в общей директории:

```
taehyung # chown efrich:efrich /srv/nfs/nayeon/test.txt
```

```
nayeon:~$ ls -l /mnt
total 12
-rw-r--r--    1 tdemin   tdemin           4 Oct 25 12:21 another_file.txt
-rw-r--r--    1 tdemin   tdemin           5 Oct 25 12:20 another_one.txt
-rw-r--r--    1 1001     1001             0 Oct 25 12:11 test.txt
-rw-r--r--    1 tdemin   tdemin           4 Oct 25 12:19 testing_file.txt
```

Nayeon неизвестны UID/GID 1001

Примечание: несмотря на то, что NFSv4 формально предусматривает механизм трансляции имен пользователей/групп при запросе, это не решает проблему возможных коллизий имен пользователей и проблемы безопасности, связанные с этим, и поэтому здесь не рассматривается.

Использование SMTP-сервера

Postfix преднастроен из коробки работать локальным SMTP-сервером. Включим его в автозапуск, сгенерируем базу почтовых псевдонимов, после чего запустим:

```
taehyung # newaliases
taehyung # rc-update add postfix
taehyung # service postfix start
```

Пошлем почту пользователю efrich:

```
tdemin@taehyung % echo "Incoming mail" | mail -s "Subject of local email" efrich
```

Перезайдем в систему как пользователь efrich, после чего просмотрим входящую почту:

```
efrich@taehyung % mail -f /var/mail/efrich
& 1
```

```
taehyung:~$ sudo -u efrich -i mail -f /var/mail/efrich
Mail version 8.1 6/6/93.  Type ? for help.
"/var/mail/efrich": 2 messages 2 new
>N  1 tdemin@taehyung.loca  Tue Oct 27 22:03  14/499   "Local email"
 N  2 tdemin@taehyung.loca  Tue Oct 27 22:05  14/499   "Local email"
& 1
Message 1:
From tdemin@taehyung.localdomain  Tue Oct 27 22:03:54 2020
X-Original-To: efrich
Delivered-To: efrich@taehyung.localdomain
To: efrich@taehyung.localdomain
Subject: Local email
Date: Tue, 27 Oct 2020 22:03:54 +0000 (UTC)
From: Linux User <tdemin@taehyung.localdomain>

Incoming mail

&
```

Локальная почта

Пошлем почту с Nayeon. Создадим файл `.msmtprc` в домашней директории, запишем в него следующий текст (устанавливающий настройки сервера, используемого для посылки почты):

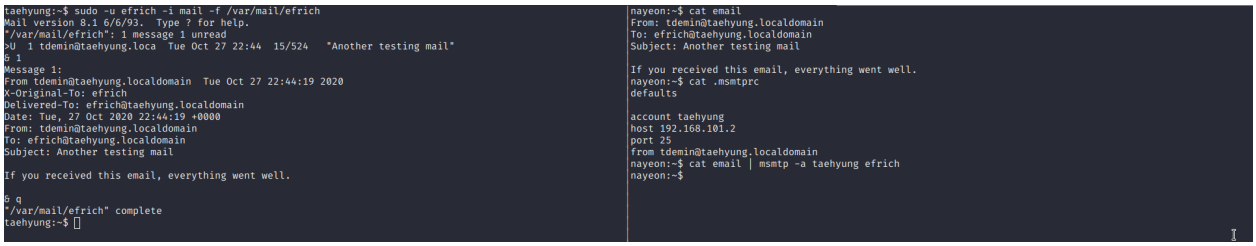
```
account taehyung
host 192.168.101.2
port 25
from tdemin@taehyung.localdomain
```

Создадим файл `my_email` и запишем в него следующее:

```
From: tdemin@taehyung.localdomain
To: efrich@taehyung.localdomain
Subject: Another testing mail
If you received this email, everything went well.
```

Отправим сообщение командой:

```
tdemin@nayeon % cat my_email | msmtp -a taehyung efrich@taehyung.localdomain
```



Посылка и чтение почты

Просмотрим журнал сообщений на Taehyung:

```
taehyung # cat /var/log/messages | grep postfix
```

```
Oct 27 22:40:01 taehyung mail.info postfix/smtpd[3658]: connect from unknown[192.168.101.3]
Oct 27 22:40:01 taehyung mail.info postfix/smtpd[3658]: C7C1BA56: client=unknown[192.168.101.3]
Oct 27 22:40:01 taehyung mail.info postfix/cleanup[3661]: C7C1BA56: message-id=<>
Oct 27 22:40:01 taehyung mail.info postfix/qmgr[3647]: C7C1BA56: from=<tdemin@taehyung.localdomain>, size=
276, nrcpt=1 (queue active)
Oct 27 22:40:01 taehyung mail.info postfix/smtpd[3658]: disconnect from unknown[192.168.101.3] ehlo=1 mail
=1 rcpt=1 data=1 quit=1 commands=5
Oct 27 22:40:01 taehyung mail.info postfix/local[3662]: C7C1BA56: to=<efrich@taehyung.localdomain>, relay=
local, delay=0.02, delays=0.01/0.01/0/0, dsn=2.0.0, status=sent (delivered to mailbox)
Oct 27 22:40:01 taehyung mail.info postfix/qmgr[3647]: C7C1BA56: removed
Oct 27 22:43:19 taehyung mail.info postfix/smtpd[3664]: connect from unknown[192.168.101.3]
Oct 27 22:43:19 taehyung mail.info postfix/smtpd[3664]: E1D09A56: client=unknown[192.168.101.3]
Oct 27 22:43:19 taehyung mail.info postfix/cleanup[3667]: E1D09A56: message-id=<>
Oct 27 22:43:19 taehyung mail.info postfix/qmgr[3647]: E1D09A56: from=<tdemin@taehyung.localdomain>, size=
356, nrcpt=1 (queue active)
Oct 27 22:43:19 taehyung mail.info postfix/smtpd[3664]: disconnect from unknown[192.168.101.3] ehlo=1 mail
=1 rcpt=1 data=1 quit=1 commands=5
Oct 27 22:43:19 taehyung mail.info postfix/local[3668]: E1D09A56: to=<efrich@taehyung.localdomain>, orig_t
o=<efrich>, relay=local, delay=0.02, delays=0.01/0/0/0, dsn=2.0.0, status=sent (delivered to mailbox)
Oct 27 22:43:19 taehyung mail.info postfix/qmgr[3647]: E1D09A56: removed
Oct 27 22:44:19 taehyung mail.info postfix/smtpd[3664]: connect from unknown[192.168.101.3]
Oct 27 22:44:19 taehyung mail.info postfix/smtpd[3664]: B87C4A56: client=unknown[192.168.101.3]
Oct 27 22:44:19 taehyung mail.info postfix/cleanup[3667]: B87C4A56: message-id=<>
Oct 27 22:44:19 taehyung mail.info postfix/qmgr[3647]: B87C4A56: from=<tdemin@taehyung.localdomain>, size=
356, nrcpt=1 (queue active)
Oct 27 22:44:19 taehyung mail.info postfix/smtpd[3664]: disconnect from unknown[192.168.101.3] ehlo=1 mail
=1 rcpt=1 data=1 quit=1 commands=5
Oct 27 22:44:19 taehyung mail.info postfix/local[3668]: B87C4A56: to=<efrich@taehyung.localdomain>, orig_t
o=<efrich>, relay=local, delay=0.01, delays=0.01/0/0/0, dsn=2.0.0, status=sent (delivered to mailbox)
Oct 27 22:44:19 taehyung mail.info postfix/qmgr[3647]: B87C4A56: removed
Oct 27 22:47:39 taehyung mail.info postfix/anvil[3656]: statistics: max connection rate 1/60s for (smtp:19
2.168.101.3) at Oct 27 22:37:50
Oct 27 22:47:39 taehyung mail.info postfix/anvil[3656]: statistics: max connection count 1 for (smtp:192.1
68.101.3) at Oct 27 22:37:50
Oct 27 22:47:39 taehyung mail.info postfix/anvil[3656]: statistics: max cache size 1 at Oct 27 22:37:50
taehyung:~$
```

Журнал почтового сервера с протоколом соединений

Примечание: конфигурация почтового сервера, работающего в Интернете, на практике намного более сложна из-за введенных большей частью серверов антиспам-ограничений, необходимости проводить аутентификацию пользователей, и других требований. Корректная конфигурация такого сервера заслуживает своего отдельного пособия и здесь не рассматривается. Кроме того, такой сервер предполагает настройку дополни-

тельного ПО, в том числе IMAP-сервера вроде Dovecot для возможности получения почты удаленными клиентами.

Отчет по работе

В отчет требуется поместить следующее:

1. Вывод списка файлов шары NFS с Taehyung и содержимого произвольного файла.
2. Вывод этого же списка файлов на смонтированной шаре с Nayeon и содержимого этого же файла.
3. Процесс отправки и принятия локальной почты на Taehyung.
4. Процесс отправки и принятия почты с Nayeon.
5. Содержимое журнала сообщений Postfix.

Контрольные вопросы

1. Каковы основные ограничения NFSv4?
2. Как сделать подключение NFS-директории автоматическим на клиенте?
3. Для чего нужен протокол SMTP?
4. Возможно ли к почтовому серверу, настроенному ранее, подключиться почтовым клиентом для просмотра входящей почты?

ЛАБОРАТОРНАЯ РАБОТА №6. ФАЙРВОЛЛ

Используемое программное обеспечение

В данной лабораторной работе используются стандартная утилита netcat (вернее, ее реализация в Busybox) и nftables, последняя на текущий момент итерация файрволла в Linux на базе фреймворка netfilter и заменяющая собой объявленный устаревшим в 2019 году инструментарий iptables.

Примечание: несмотря на устаревание инструментария iptables, он продолжает использоваться в большом количестве мест (так, Docker и libvirtd продолжают реализовывать NAT посредством манипуляций с iptables, а некоторые дистрибутивы по состоянию на 2020 еще не завершили миграцию на nftables; в целях упрощения миграции существуют также iptables-nft и iptables-translate). Хоть в курсе он и не рассматривается, имеет смысл с ним минимально ознакомиться.

Конфигурация машин

Аналогична конфигурации машин из лабораторной работы №4. Если у вас остался активированным сервис автомонтирования NFS, отключите его:

```
# rc-update del nfsmount && service nfsmount stop
```

Установка требуемого ПО

Установим на Taehyung пакет nftables:

```
# apk add nftables
```

Общая информация

Инструментарий nftables предоставляет возможность конфигурации посредством загрузки файла, содержащего набор правил nftables, либо выполнения индивидуальных команд с помощью утилиты `nft`. Загрузка файла также осуществляется утилитой `nft` с ключом `-f`. Файлы можно также загружать, сделав их исполняемыми посредством `chmod +x` и указав следующий заголовок (при этом имя файла подается на вход утилите `nft`, выполненной с ключом `-f`):

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

У утилиты `nft` также есть режим интерактивной оболочки, в него можно попасть, запустив ее с ключом `-i`. Конфигурация файрволла может редактироваться исключительно суперпользователем.

Примечание: несмотря на то, что утилита nft не всегда выбрасывает ошибки, будучи выполненной не от суперпользователя (так, по состоянию на версию 0.9.6 nft не выбрасывает ошибки при выполнении `nft list ruleset`, но и не выводит списка примененных правил), она должна всегда выполняться с его правами для корректной работы.

Дистрибутивы часто предоставляют вспомогательные средства для автоматической настройки файрволла. Так, в Alpine Linux пакет nftables содержит скрипт инициализации, на старте системы загружающий правила из файла `/etc/nftables.nft`. Пакет также предлагает преднастроенный по умолчанию достаточно параноидально файрволл. Включить автозагрузку можно следующей командой:

```
taehyung # rc-update add nftables
```

Для настройки файрволла мы будем использовать следующую заготовку содержимого файла `/etc/nftables.nft`:

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

flush ruleset
```

Первая строка задает интерпретатор для файла, в данном случае это `nft` с ключом `-f`. Команда `flush ruleset` сбрасывает текущие настройки файрволла. Если настройки файрволла пусты (т. е. не установлено ни одного фильтра), то фильтрация трафика не осуществляется. Текущие настройки можно посмотреть `list ruleset`.

После изменений в файле конфигурации будем выполнять следующую команду:

```
taehyung # service nftables reload
```

nftables также позволяет конфигурацию при работе файрволла без сброса настроек из CLI. Так, создание таблицы `my_filter`, цепочки в ней `my_input` и правила, разрешающего SSH-трафик, может выглядеть следующим образом:

```
# nft add table inet my_filter
# nft add chain inet my_filter my_input '{ type filter hook input priority 0; policy
reject }'
# nft add rule inet my_filter my_input 'tcp dport ssh accept'
```

Примечание: не пытайтесь использовать этот пример через SSH! Он разорвет ваше текущее SSH-соединение. Конфигурация файрволла через удаленное соединение в целом должна производиться аккуратно, поскольку некорректная настройка может сделать машину полностью недоступной по сети.

Полный список доступных директив утилиты nft приведен в кратком виде в официальной документации и в `nft(8)`.

Директивы файрволлу организуются в цепочки, которые, в свою очередь, организуются в таблицы. Переход к цепочке происходит при регистрации цепочки как обработчика одного из возможных событий. Обработчики имеют приоритеты, задаваемые целым числом. Директивы состоят из условия и действия; условие может быть опущено.

conntrack — это таблица отслеживания состояния соединений в подсистеме Linux netfilter. Отслеживание состояния проходящих соединений требуется для реализации логики файрволла, выходящей за грани stateless-файрволла. Так, стандартная для домашних роутеров схема «разрешать исходящий трафик и уже установленные соединения, запретить входящий трафик» невозможна без отслеживания состояния проходящих через машину соединений файрволлом — при использовании файрволла без состояния было бы невозможно разрешить трафик, идущий в ответ на исходящие из локальной сети соединения.

Правила записываются в формате, лучше всего иллюстрируемом примерным файлом с комментариями:

```
# Таблица типа inet (IPv4/IPv6), названная custom_filter.
# Поскольку ПО, полагающееся на iptables-nft, достаточно часто конфликтует
# с таблицами с предопределенными в iptables именами, по возможности не
# стоит называть их input, output, forward и другими именами из iptables.
table inet custom_filter {
    # Цепочка правил с именем public. К ней может быть совершен прыжок из
    # другой цепочки. Правила внутри цепочки состоят из набора условий и
    # выполняемого действия. Условие может быть пустым, тогда действие
    # станет безусловным.
    chain public {
        # TCP-соединения на входящий порт 10001 принимаются.
        tcp dport 10001 accept
        # TCP-соединения принимаются на порт службы SSH (22). nftables
        # имеет предопределенные имена для некоторых базовых служб.
        tcp dport ssh accept
```

```
        # Пакет не был обработан ни одним из условных действий выше,
        # отклоним его с посылкой обратно информационного ICMP-пакета
        # типа Destination Port Unreachable. В случае inet или ip6-цепочек
        # можно задать собственный тип посылаемого сообщения.
        reject
    }
    chain locals {
        # UDP-трафик на порт 20001 отбрасывается.
        udp dport 20001 drop
        # Принимаются UDP-пакеты с source-портом 68 на destination-порт
        # 67 и адресом назначения 255.255.255.255 (это сигнатура DHCP).
        udp sport 68 udp dport 67 ip daddr 255.255.255.255 accept
        # Вернемся в цепочку, из которой был произведен вызов.
        return
    }
    chain input {
        # Цепочка регистрируется как фильтр входного трафик с приоритетом 0;
        # при ненахождении нужного правила трафик отклоняется с посылкой
        # в ответ ICMP-пакета.
        type filter hook input priority 0; policy reject;
        # Примем пакеты от соединений, имеющих conntrack-состояние ESTABLISHED
        # или RELATED. Отбросим трафик от соединений, которые невозможно
        # идентифицировать.
        # Существует четыре conntrack-типа:
        # + NEW (соединение открыто, для TCP это первый SYN-пакет)
        # + ESTABLISHED (соединение открыто и на него получен ответ, для TCP
        # это время получения SYN-ACK);
        # + RELATED (соединение связано с одним из ESTABLISHED). Например,
        # это может быть ICMP-ответ, относящийся к установленному
        # TCP-соединению.
        # + INVALID (пакет/соединение не идентифицированы). Такой трафик, как
        # правило, отбрасывается.
        ct state { established, related } accept
        ct state invalid drop
        # Примем входящий трафик с loopback-интерфейса.
        iif lo accept
        # Примем IP-трафик по протоколам ICMP/IGMP.
        ip protocol icmp accept
        ip protocol igmp accept
        # Примем IPv6-трафик, в заголовках которого указано, что это ICMPv6.
        ip6 nexthdr icmpv6 accept
        # При получении IPv4-трафика из подсети 172.16.0.0/12 прыгнем в
        # цепочку locals.
        ip saddr 172.16.0.0/12 jump locals
        # При получении трафика с интерфейсов с именами uni0 и br0 сделаем
        # прыжок в цепочку public. При использовании iifname/oifname вместо
        # iif/oif в условиях интерфейс может не существовать (удобно с
        # временными сетевыми интерфейсами), но условия с iif/oif работают
        # быстрее.
        iifname { uni0, br0 } jump public
    }
    chain forward {
        # Цепочка для пересылки трафика. По умолчанию трафик, который нужно
        # переслать, дропается.
        type filter hook forward priority 0; policy drop;
        # Примем трафик, идущий на выход с br0 и входящий с 172.16.0.0/12
        # (полезно при организации роутера с SNAT).
        oifname br0 ip saddr 172.16.0.0/12 accept
        # Примем трафик, пришедший на br0, от уже установленных соединений.
        iifname br0 ct state { established, related } accept
    }
    chain output {
        # Разрешим весь исходящий трафик.
```

```
        type filter hook output priority 0; policy accept;
    }
}
```

Примечание: файл выше приведен исключительно для демонстрации возможностей синтаксиса nft. Не пытайтесь использовать аналогичные правила файрволла для защиты реальной машины.

Сложная конфигурация файрволла

Блокировка входящих пакетов

Настроим следующую конфигурацию: разрешаются подключения к sshd в сети `192.168.101.0/24`, ICMPv4/ICMPv6/IGMP-пакеты принимаются, другие пакеты отбрасываются:

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

flush ruleset

table inet filter {
    chain input {
        type filter hook input priority 0; policy drop
        ct state { established, related } accept
        ct state invalid drop
        iif lo accept
        ip protocol icmp accept
        ip6 nexthdr icmpv6 accept
        ip protocol igmp accept
        ip daddr 192.168.101.0/24 tcp dport 22 accept
    }
}
```

Запустим утилиту netcat в режиме слушателя на TCP-порту 10001 Taehyung:

```
taehyung % nc -l -p 10001
```

Попытаемся подключиться на этот порт с Nayeon:

```
nayeon % nc -w 5 192.168.101.2 10001
```

Заметим, что Taehyung отбрасывает пакеты:

```
nayeon:~$ nc -z -v -w 5 192.168.101.2 10001
nc: 192.168.101.2 (192.168.101.2:10001): Operation timed out
nayeon:~$
```

Соединение не было установлено за 5 секунд

Динамически сконфигурируем файрволл для принятия пакетов на TCP-порт 10001:

```
taehyung # nft add rule inet filter input 'tcp dport 10001 accept'
```

Снова проверим соединение:

```
nayeon:~$ nc -z -v -w 5 192.168.101.2 10001
192.168.101.2 (192.168.101.2:10001) open
nayeon:~$
```

Соединение устанавливается успешно

Блокировка исходящих пакетов

Заблокируем возможность соединяться по сети вне локальной сети UniLWSwitch:

```
chain output {
    type filter hook output priority 0; policy drop
    ip saddr 192.168.101.0/24 accept
}
```

Проверим возможность выполнить соединение (вместо 192.168.122.1 используйте ваш шлюз):

```
% ping 192.168.122.1
% ping 127.0.0.1
% ping 192.168.101.2
```

```
taehyung:~$ sudo ping -c 4 192.168.122.1
PING 192.168.122.1 (192.168.122.1): 56 data bytes
ping: sendto: Operation not permitted
taehyung:~$ sudo ping -c 4 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
ping: sendto: Operation not permitted
taehyung:~$ sudo ping -c 4 192.168.101.2
PING 192.168.101.2 (192.168.101.2): 56 data bytes
64 bytes from 192.168.101.2: seq=0 ttl=64 time=0.051 ms
64 bytes from 192.168.101.2: seq=1 ttl=64 time=0.059 ms
^C
--- 192.168.101.2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.051/0.055/0.059 ms
taehyung:~$
```

Исходящие соединения сбрасываются всюду, включая `127.0.0.0/8`

Пересылка пакетов. Настройка NAT44

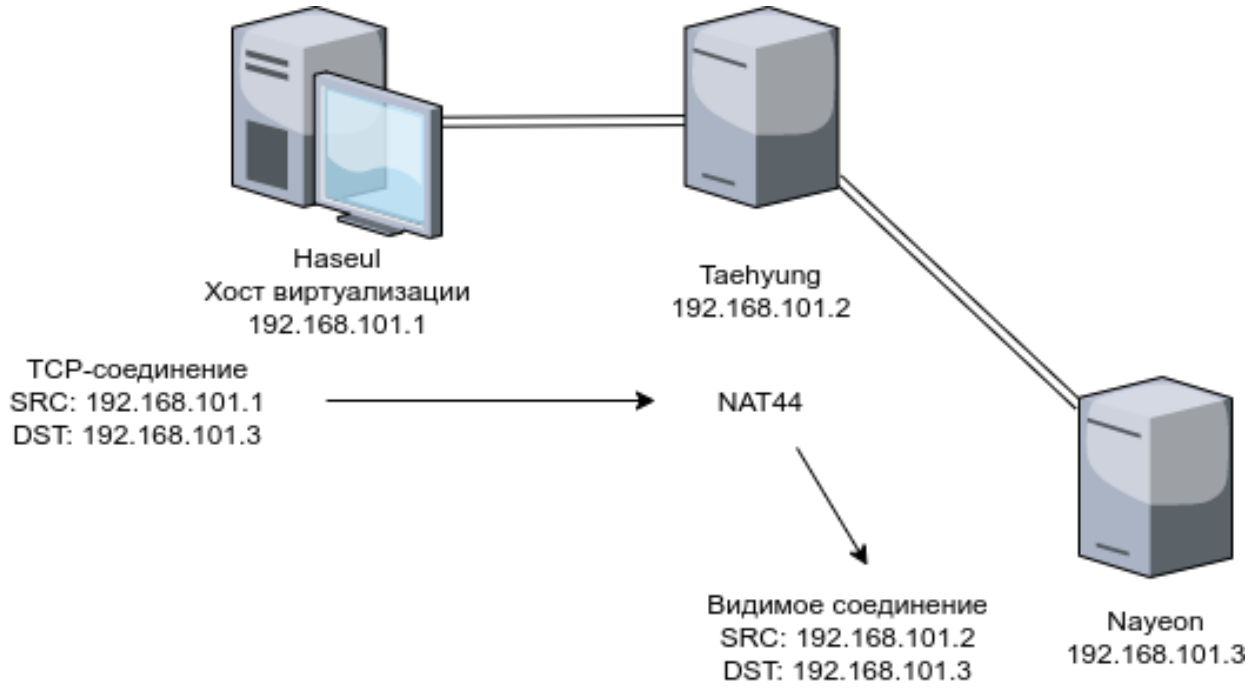
NAT, сетевая адресная трансляция, при которой IP-адрес отправителя пакета переписывается IP машины, производящей трансляцию, в Linux реализуется на уровне файрволла. NAT можно условно разбить по назначению на SNAT (source NAT, производится для исходящих соединений отправителя пакета) и DNAT (destination NAT, производится для входящих соединений). Здесь мы рассматриваем лишь SNAT. Отслеживание соединений для выполнения обратной замены производится в случае протоколов транспортного уровня TCP/UDP по полю порта источника в пакете; таблицу сопоставления порта источника и IP-адреса, для которого выполняется NAT, поддерживает conntrack.

Примечание: NAT не является и не должен использоваться как средство обеспечения безопасности локальной сети. Атакующий может быть в состоянии специальным образом сконструировать пакеты так, что машина, производящая NAT (а значит, и пересылку пакетов) перешлет их в локальную сеть согласно собственной таблице маршрутизации. Таким образом, использование NAT никак не заменяет корректно настроенный файрволл.

При использовании NAT в Linux решение о том, производится ли NAT, выполняется по первому пакету — это означает, что правила файрволла обрабатывают только первый пакет соединения.

Здесь мы настроим NAT44 для соединений между хостом и Nayeon через Taehyung; Taehyung будет переписывать source-адреса на собственные при пересылке трафика.

Таким образом, получаем следующую иерархию сети:



Иерархия сети с роутером, выполняющим NAT44

Сконфигурируем сеть нужным образом. Убедимся, что на Taehyung включена пересылка IPv4-пакетов:

```
taehyung % sysctl net.ipv4.ip_forward
```

Если `net.ipv4.ip_forward` установлен в `0`, необходимо включить пересылку IPv4-пакетов так же, как это делалось в лабораторной работе №2.

Настроим сеть UniLWSwitch на Nayeon так, чтобы Nayeon передавала трафик через Taehyung:

```
auto eth1
iface eth1 inet static
        address 192.168.101.3
        gateway 192.168.101.2
        netmask 255.255.255.0
        metric 1
```

Примечание: разумеется, текущее сетевое соединение с Nayeon по SSH при этом придется разорвать. После настройки nftables на Taehyung и корректной настройки маршрута соединение по SSH можно будет установить заново.

Добавим запись в таблицу маршрутизации на Haseul:

```
haseul # ip r add 192.168.101.3 dev uni0 via 192.168.101.2 src 192.168.101.1 metric 0
Haseul PS Admin > route add 192.168.101.3 mask 255.255.255.255 192.168.101.2 metric 1
```

Настроим пакетный фильтр на Taehyung следующим образом:

```
chain forward {
    type filter hook forward priority 0; policy drop;
    # вообще говоря, в данном случае первая директива accept
    # здесь не нужна, поскольку она перекрывается второй, но
    # подобного рода правило, разрешающее уже установленные
    # соединения в обратную сторону, полезно для роутеров,
    # выполняющих NAT; здесь оно приведено для полноты
    iif eth0 ct state { established, related } accept
    ip saddr 192.168.101.0/24 accept
}
chain nat {
    # для выполнения DNAT/SNAT по определенным правилам
    # (например, указания конкретного IP-адреса) в nftables
    # есть специальные действия dnat/snat; masquerade
    # выполняет SNAT, автоматически используя IP-адрес
    # выходного интерфейса, что удобно для пограничных
    # маршрутизаторов
    type nat hook postrouting priority 0;
    ip saddr 192.168.101.0/24 masquerade
}
```

Запустим TCP-слушатель на Nayeon на порту 10002 с журналированием входящих соединений:

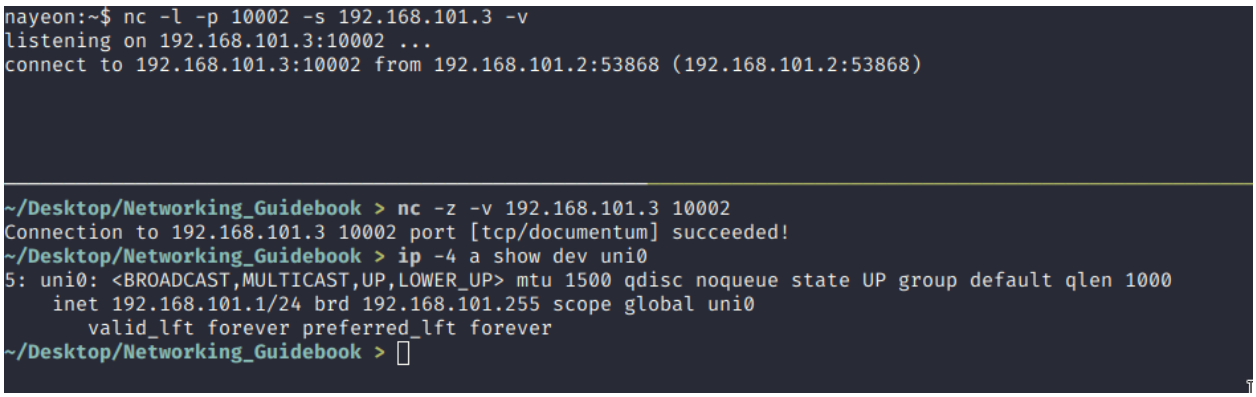
```
nayeon % nc -l -p 10002 -s 192.168.101.3 -v
```

Выполним соединение с Haseul:

```
haseul % nc -z -v 192.168.101.3 10002
```

Примечание: если на Haseul отсутствует netcat или аналогичная программа, за неимением лучшего для открытия TCP-соединения можно попробовать открыть в браузере `http://192.168.101.3:10002`. Так как HTTP работает поверх TCP, браузер установит TCP-соединение. Естественно, ждать, что в браузере откроется какая-либо страница, не стоит. Это также можно сделать telnet, или, например, PuTTY.

Увидим, что Nayeon записала source-адрес входящего соединения как `192.168.101.2`, адрес Taehyung, несмотря на то, что соединение выполнялось с Haseul:



Nayeon видит адрес, который подставляет SNAT на Taehyung

nftables имеет множество функциональности, не затронутой в данной лабораторной работе (расширенные возможности синтаксиса, подсчет пакетов, маркировка, динамическое создание наборов адресов, журналирование, и другие). С ней можно ознакомиться в `nft(8)` и в документации.

Отчет по работе

В отчет требуется поместить следующее (конфигурацию файрволла требуется снимать с помощью `nft list ruleset`):

1. конфигурацию файрволла при настройке для фильтрации входящих соединений, а также результаты попыток соединения до и после неудачной попытки соединения;
2. конфигурацию файрволла при настройке для блокировки исходящих соединений, а также тест соединения;
3. записи в таблице маршрутизации на Haseul и Nayeon, а также конфигурацию файрволла и traceroute при использовании SNAT на Taehyung.

После выполнения

В целях упрощения выполнения последующих работ деактивируем файрволл на Taehyung:

```
# rc-update del nftables
# nft flush ruleset
```

Контрольные вопросы

1. Возможно ли исполнять конфигурацию nftables как скрипт?
2. Как можно проверить, слушает ли порт на машине какая-либо программа?
3. Какие основные примитивы используются для группировки правил в nftables?
4. Чем отличается поведение директивы `drop` от `reject` в nftables?
5. По каким признакам conntrack может отслеживать состояние соединений?
6. Зачем нужен NAT в IPv4?
7. В чем отличие SNAT от DNAT?

ЛАБОРАТОРНАЯ РАБОТА №7. IPV6

Данная лабораторная работа призвана ознакомить с некоторыми аспектами стека IPv6, отличающими сети, построенные на нем, от протокола IPv4. Несмотря на то, что теоретически один пакет IPv6 действительно мало чем отличается от аналогичного пакета IPv4, стек имеет огромное количество практически важных различий, которые регулярно упускаются из виду разработчиками разного ПО, в том числе крупного / популярного, приводимого как примеры ниже. В частности:

- в IPv6 нет broadcast, вместо него используется multicast на строго заданные адреса в подсети `ff00::/8`, а также только один loopback-адрес, `::1/128` (так, встроенный в systemd-resolved DNS-сервер не работает на IPv6-only-машинах без поддержки dual stack, слушая исключительно адрес 127.0.0.53);
- в IPv6 роутеру не требуется хранить состояние (использование DHCPv6 не обязательно для автоматической конфигурации адреса, если подсеть, маршрутизируемая роутером, /64 или большего размера, но все еще требуется для централизованного управления адресами — так, в Android отсутствует поддержка DHCPv6 по причинам, которые Google отказывается называть вплоть с 2011);
- в IPv6 нормально иметь несколько IP-адресов на одном интерфейсе, как минимум один — global unicast, в том числе от нескольких роутеров;
- в IPv6 нормально выдавать под одну сеть префикс размера минимум / 64;
- в IPv6 работа ICMPv6 критична для нормальной работы сети, если каждый узел не устанавливает настройки сети статически — так, обмен пакетами Router Solicitation/Advertisement для нахождения узлом ближайшего роутера идет через него (так, популярный антивирусный пакет Dr. Web вплоть до обновлений от лета 2020 продолжал блокировать ICMPv6, тем самым по факту ломая собственным пользователям IPv6);

- в IPv6 практически не используется NAT, кроме как для организации интероперации с сетями IPv4.

Стек имеет множество других отличий, не упомянутых выше, но упомянутых должно быть достаточно для понимания разницы между построением сети на базе IPv4 и IPv6. Иметь в виду эти отличия требуется для корректной реализации программного обеспечения, которое будет работать без помех после окончательного ухода от IPv4. Не самое высокое распространение IPv6 в мире на текущий момент ограничивается, не в последнюю очередь, некорректной реализацией сетевого ПО, а также непониманием отличий нового протокола от старого.

Конфигурация машин

Аналогична конфигурации машин из лабораторной работы №4.

Link-local

На каждом сетевом интерфейсе, если link-local-адресация не отключена, каждая машина генерирует себе IP-адрес в подсети `fe80::/10`, выводя младшие 64 бита адреса по модифицированному алгоритму EUI-64 из MAC-адреса сетевой карты:

Модифицированный EUI-64

Данные адреса маршрутизируются только в текущем layer-2 сегменте сети. Данный алгоритм используется также при выводе адресов при использовании

SLAAC; в ОС могут быть также реализованы и другие механизмы вывода адресов, например, позволяющий задание общего суффикса во всех подсетях IPv6 Token в Linux.

Проверим IP и MAC-адреса на сетевых интерфейсах внутренней сети:

```
# ip a show dev eth0
```

Сравним IP-адреса с MAC-адресами. Сделаем вывод.

Попробуем проверить связность между машинами по link-local адресам (вместо указанного в примере адреса Taehyung подставьте собственный):

```
nayeon % ping -I eth0 fe80::5054:ff:fe57:6e8b
```

```
nayeon:~$ ping -I eth0 fe80::5054:ff:fe57:6e8b
PING fe80::5054:ff:fe57:6e8b (fe80::5054:ff:fe57:6e8b): 56 data bytes
ping: can't bind to interface eth0: Operation not permitted
64 bytes from fe80::5054:ff:fe57:6e8b: seq=0 ttl=64 time=0.471 ms
64 bytes from fe80::5054:ff:fe57:6e8b: seq=1 ttl=64 time=0.250 ms
64 bytes from fe80::5054:ff:fe57:6e8b: seq=2 ttl=64 time=0.233 ms
64 bytes from fe80::5054:ff:fe57:6e8b: seq=3 ttl=64 time=0.211 ms
^C
--- fe80::5054:ff:fe57:6e8b ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.211/0.291/0.471 ms
nayeon:~$
```

Удачная проверка соединения до Taehyung по link-local-адресу

Проверим связность до какой-нибудь машины с link-local IPv6-адресом вне текущего сегмента L2 (подойдет, например, link-local-адрес вашего роутера; пользователи роутерной прошивки OpenWrt могут узнать его так же, как и в обычной Linux-системе):

```
nayeon:~$ ping -I eth0 fe80::8ad7:f6ff:feba:2340
PING fe80::8ad7:f6ff:feba:2340 (fe80::8ad7:f6ff:feba:2340): 56 data bytes
ping: can't bind to interface eth0: Operation not permitted
^C
--- fe80::8ad7:f6ff:feba:2340 ping statistics ---
8 packets transmitted, 0 packets received, 100% packet loss
nayeon:~$
```

Безуспешная попытка соединения до машины в другом layer 2-сегменте

Попробуем пропинговать все роутеры в сети, а потом все узлы в сети вообще по multicast-адресам из `ff00::/8`. Выполним:

```
nayeon % ping -I eth0 ff02::2
```

```
nayeon % ping -I eth0 ff02::1
```

```
nayeon:~$ ping -I eth0 ff02::2
PING ff02::2 (ff02::2): 56 data bytes
ping: can't bind to interface eth0: Operation not permitted
64 bytes from fe80::5054:ff:fe57:52df: seq=0 ttl=64 time=0.207 ms
64 bytes from fe80::5054:ff:fe57:52df: seq=1 ttl=64 time=0.197 ms
64 bytes from fe80::5054:ff:fe57:52df: seq=2 ttl=64 time=0.213 ms
64 bytes from fe80::5054:ff:fe57:52df: seq=3 ttl=64 time=0.203 ms
^C
--- ff02::2 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.197/0.205/0.213 ms
nayeon:~$ Connection to 192.168.101.3 closed.
~/Desktop/Networking_Guidebook > ip a show dev uni0
7: uni0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 52:54:00:57:52:df brd ff:ff:ff:ff:ff:ff
    inet 192.168.101.1/24 brd 192.168.101.255 scope global uni0
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe57:52df/64 scope link
       valid_lft forever preferred_lft forever
~/Desktop/Networking_Guidebook >
```

Пинг роутеров доходит до хостовой машины, на которой запущен dnsmasq

Настройка роутера

В IPv4 для поиска роутера используется DHCPv4, в котором для поиска роутера посылается UDP-пакет DHCPDISCOVER по broadcast-адресу `255.255.255.255` с destination-портом 67 и source-портом 68. В IPv6 вместо этого посылается ICMPv6-пакет типа 133 (Router Solicitation) на multicast-адрес `ff02::2` (на котором принимают сообщения все роутеры). В ответ на него роутер отвечает ICMPv6-пакетом типа 134 (Router Advertisement), содержащим данные о текущей подсети/шлюзе и указывающим хостам использовать либо SLAAC (stateless address autoconfiguration, протокол автоматической конфигурации сетевого адреса без состояния), либо DHCPv6 для получения адреса в текущей подсети. В современных ОС популярной практикой в подсетях /64 и более крупных является генерировать два адреса — один с помощью модифицированного EUI-64 в качестве постоянного адреса для внешних соединений с машиной, и один временный, использующийся для исходящих соединений.

Для настройки IPv6-роутера мы будем использовать знакомый по лабораторной работе №3 DHCP-сервер dnsmasq. Включим в машины Taehyung и Nayeon свич UniLWSwitch2. Адреса будем раздавать из произвольной /64-подсети внутри подсети `fc00::/7`, зарезервированной в IPv6 для частных сетей (т. н. ULA, Unique Local Address).

Изменим следующие настройки в конфигурационном файле dnsmasq на Taehyung:

1. `port` установим в `0` для отключения DNS.
2. `interface` установим в `eth2`.

Отредактируем конфигурацию интерфейса `eth2` следующим образом в `/etc/network/interfaces` на Taehyung:

```
auto eth2
iface eth2 inet6 static
        address fd00:f00d::1
        netmask 64
```

Так как поддержка IPv6 в реализации ifupdown в Busybox очень посредственна и недостаточна для корректной настройки, установим на Nayeon оригинальный ifupdown от проекта Debian, а также DHCP-клиент dhclient:

```
nayeon # apk add ifupdown dhclient
```

На Nayeon в `/etc/network/interfaces` запишем следующее:

```
auto eth2
iface eth2 inet6 auto
```

SLAAC

Добавим в конфигурационный файл dnsmasq следующую директиву, активирующую раздачу адресов из подсети `fd00:f00d::/64` (размер префикса /64 по умолчанию) сроком жизни на 12 часов и предписывающую получение адресов посредством использования SLAAC:

```
dhcp-range=fd00:f00d::,ra-stateless,12h
```

Запустим dnsmasq. Поднимем интерфейс `eth2` на Nayeon:

```
nayeon # ifup eth2
```

Просмотрим IP-адреса на Nayeon, увидим, что в подсети роутера мы получили два адреса, сгенерированный по модифицированному EUI-64 и полностью случайный в качестве вторичного адреса:

```
nayeon:~$ ip a show dev eth2
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:cc:71:29 brd ff:ff:ff:ff:ff:ff
    inet6 fd00:f00d::9538:b6e6:c091:18e3/64 scope global secondary dynamic
       valid_lft 43197sec preferred_lft 43197sec
    inet6 fd00:f00d::5054:ff:fecc:7129/64 scope global dynamic
       valid_lft 43197sec preferred_lft 43197sec
    inet6 fe80::5054:ff:fecc:7129/64 scope link
       valid_lft forever preferred_lft forever
```

На интерфейсе теперь есть два адреса в подсети роутера

Просмотрим журнал dnsmasq на Taehyung и заметим, что в ответ на пакет Router Solicitation с MAC-адреса Nayeon dnsmasq ответил пакетом Router Advertisement с сетевыми настройками. При этом, поскольку адреса получены посредством генерации их на самой машине за счет использования SLAAC, dnsmasq не знает, какой полный адрес получила машина, запросившая адрес. Таким образом, при использовании SLAAC роутер не хранит состояния:

```
Nov 15 00:05:15 taehyung daemon.info dnsmasq-dhcp[2946]: RTR-SOLICIT(eth2) 52:54:00:cc:71:29
Nov 15 00:05:15 taehyung daemon.info dnsmasq-dhcp[2946]: RTR-ADVERT(eth2) fd00:f00d::
taehyung:~$
```

Фрагмент журнала dnsmasq

DHCPv6

Для использования DHCPv6 необходимо указать подсеть в формате «начальный адрес,конечный адрес». Зададим следующие директивы в конфигурации dnsmasq (в первой начальный и конечный адреса в подсети формируются из адреса `eth2`, а вторая включает рассылку RA с установленным флагом M и неустановленным флагом A, разрешающим SLAAC; в результирующем диапазоне 2^{16}-1 адресов):

```
dhcp-range=::2,::ffff,constructor:eth2,64,12h
enable-ra
```

Перезапустим dnsmasq:

```
taehyung # service restart
```

Реализация ifupdown от проекта Debian в текущей версии в Alpine Linux имеет сразу несколько критичных в нашем случае багов:

1. DHCPv6-клиенты не обнаруживаются в системе при использовании типа конфигурации `dhcp` (или `auto` с `dhcp 1`);
2. При использовании типа конфигурации `auto` и `dhcp 0` установленный флаг M и неустановленный A в пакете RA приведет к ошибке конфигурации.
3. При использовании типа конфигурации `manual` выключается прием RA-пакетов на текущем интерфейсе.

Вооружившись этим знанием, напишем конфигурацию для интерфейса на Nayeon вручную:

```
auto eth2
iface eth2 inet6 manual
        up sysctl net.ipv6.conf.$IFACE.accept_ra=2
        post-up dhclient -6 $IFACE
```

В данной конфигурации при подъеме интерфейса будет выполнена команда после `up` (включающая прием RA-пакетов для интерфейса; вместо `$IFACE` ifupdown подставляет название текущего интерфейса), а после нее будет запущен DHCPv6-клиент для получения адреса в сети.

Примечание: данная конфигурация все еще имеет проблемы и ненадежна — так, интерфейс при ней может подняться не с первого раза. Надежным решением упомянутых проблем ifupdown в общем случае является использование полноценного сетевого конфигуратора вроде systemd-networkd. Для простоты здесь мы не рассматриваем другие программы.

В более новых версиях ifupdown в Alpine Linux ситуация может измениться. Корректная конфигурация в случае, если огрехи, упомянутые выше, были бы исправлены, сводится к добавлению `dhcp 1`. На момент написания данного пособия в репозиториях Alpine Linux поставляется ifupdown 0.8.35.

Поднимем интерфейс:

```
nayeon # ifup eth2
```

Пропингуем роутер:

```
nayeon % ping fd00:f00d::1
```

```
nayeon:~$ ip a show dev eth2
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:00:cc:71:29 brd ff:ff:ff:ff:ff:ff
    inet6 fd00:f00d::fbdf:e26e:2ef:1796/128 scope global dynamic
       valid_lft 6392sec preferred_lft 6092sec
    inet6 fe80::5054:ff:fecc:7129/64 scope link
       valid_lft forever preferred_lft forever
nayeon:~$ ping -c 4 fd00:f00d::1
PING fd00:f00d::1 (fd00:f00d::1): 56 data bytes
64 bytes from fd00:f00d::1: seq=0 ttl=64 time=0.312 ms
64 bytes from fd00:f00d::1: seq=1 ttl=64 time=0.251 ms
64 bytes from fd00:f00d::1: seq=2 ttl=64 time=0.315 ms
64 bytes from fd00:f00d::1: seq=3 ttl=64 time=0.240 ms

--- fd00:f00d::1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.240/0.279/0.315 ms
nayeon:~$
```

Адрес получен по DHCPv6

Проверим лог dnsmasq, заметим сообщения DHCPREPLY о выдаче адресов по DHCPv6.

После выполнения

Удалим свич UniLWSwitch2 из виртуальных машин. Откатим конфигурацию виртуальных машин к предыдущему состоянию.

Отчет по работе

В отчет требуется поместить следующее:

1. link-local-адреса машин на интерфейсе, соответствующем свичу UniLWSwitch, и их MAC-адреса;
2. результаты проверки соединения по link-local-адресам, включая multicast-адреса;
3. результат получения адреса в подсети посредством SLAAC на Nayeon (выбранная подсеть должна отличаться от `fd00:f00d::/64`), а также релевантный фрагмент журнала dnsmasq;
4. результат получения адреса в подсети посредством DHCPv6 на Nayeon;
5. результат проверки соединения до роутера по его адресу внутри выбранной подсети и после использования SLAAC, и после DHCPv6.

Контрольные вопросы

1. Назовите известные вам отличия сетей на основе IPv6 от IPv4.
2. Почему в сетях IPv6 редко требуется NAT?

3. Как узел может получить IP-адрес в сетях, основанных на IPv6?
4. Почему минимальный размер подсети, в которой может использоваться SLAAC — /64?
5. Каковы основные причины не высокой на текущий момент распространенности IPv6?

ЛАБОРАТОРНАЯ РАБОТА №8. АНАЛИЗ ТРАФИКА. VPN

Используемое ПО

Для анализа трафика в этой лабораторной работе используется утилита tcpdump. Для обмена данными используется реализация стандартной утилиты netcat в Busybox. Для организации виртуальной частной сети используется шифрованный туннель WireGuard.

Конфигурация машин

Совпадает с конфигурацией к концу лабораторной работы №4. Для комфортной работы потребуется подключаться по SSH. Если вы не выполнили дополнительную настройку в лабораторной работе №1, выполните ее сейчас.

Анализ трафика

tcpdump, несмотря на название, является универсальной программой захвата трафика, которая позволяет в том числе сохранять дампы в формате pcap для последующего анализа, например, в Wireshark, свободной программе для анализа трафика. tcpdump использует для захвата трафика библиотеку libpcap. Фильтры, задаваемые tcpdump, обрабатываются libpcap и универсальны для всех программ захвата / анализа трафика, основанных на этой библиотеке.

Установим tcpdump на Nayeon:

```
# apk add tcpdump
```

Задание фильтров

Начнем слушать входящие соединения по UDP на порту 10001 на Taehyung:

```
taehyung % nc -p 10001 -u -l
```

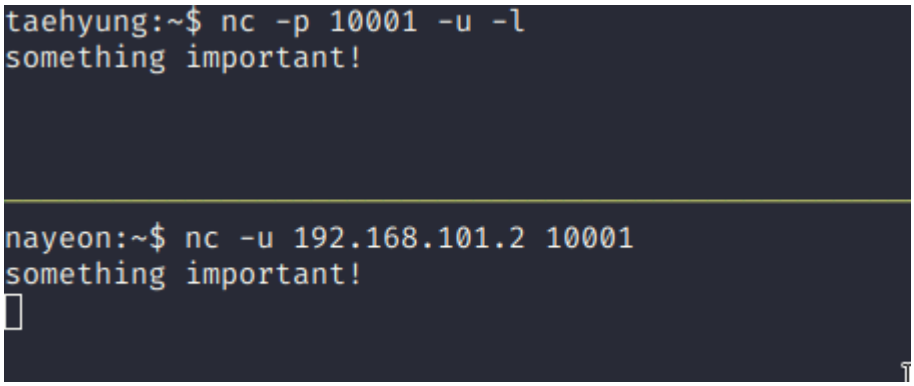
Подключимся по UDP на этот порт к Taehyung с Nayeon:

```
nayeon % nc -u 192.168.101.2 10001
```

Запустим еще один сеанс командной оболочки на Nayeon, в нем запустим tcpdump на Nayeon, прослушивая UDP-трафик на интерфейсе, соответствующем свичу UniLWSwitch, на порт назначения 10001:

```
nayeon # tcpdump -i eth1 -n -X 'udp dst port 10001'
```

Введем произвольный текст с запущенным netcat на Nayeon, после чего нажмем Enter. Отметим, что этот же текст был выведен на экран в netcat, запущенном на Taehyung:



Данные были переданы по сети

Заметим, что в выводе tcpdump появились переданные данные в открытом виде:

```
nayeon:~$ sudo tcpdump -i eth1 -n -X 'udp port 10001'
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
18:09:08.115714 IP 192.168.101.3.36549 > 192.168.101.2.10001: UDP, length 49
        0x0000:  4500 004d 367a 4000 4011 b8cf c0a8 6503  E..M6z@.@.....e.
        0x0010:  c0a8 6502 8ec5 2711 0039 4ba1 736f 6d65  ..e...'..9K.some
        0x0020:  206d 6f72 6520 696d 706f 7274 616e 7420  .more.important.
        0x0030:  7465 7874 2073 656e 7420 6f76 6572 2076  text.sent.over.v
        0x0040:  6972 7475 616c 2077 6972 6521 0a         irtual.wire!.
18:09:17.586805 IP 192.168.101.2.10001 > 192.168.101.3.36549: UDP, length 10
        0x0000:  4500 0026 908b 4000 4011 5ee5 c0a8 6502  E..&..@.@.^...e.
        0x0010:  c0a8 6503 2711 8ec5 0012 4b7a 616e 6420  ..e.'.....Kzand.
        0x0020:  6d6f 7265 210a                           more!.
```

Данные передаются в открытом виде

Прослушаем какой-либо еще трафик. На этот раз попробуем прослушать с Nayeon весь ICMPv6-трафик между машинами в виртуальной сети UniL-WSwitch, после чего с Taehyung выполним пинг всех машин в текущем секторе L2 по `ff02::1`:

```
nayeon % sudo tcpdump -i eth1 -n -X 'icmp6'
taehyung # ping -6 -I eth1 ff02::1
```

Заметим, что генерируется большое количество трафика NDP и ответов на пинги, и весь процесс обмена трафиком открыт для просмотра:

```
nayeon:~$ sudo tcpdump -i eth1 -n -X 'icmp6'
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
18:13:27.697630 IP6 fe80::5054:ff:fe57:6e8b > ff02::1: ICMP6, echo request, seq 0, length 64
        0x0000:  6005 39ba 0040 3a01 fe80 0000 0000 0000  `.9..@:.........
        0x0010:  5054 00ff fe57 6e8b ff02 0000 0000 0000  PT...Wn.........
        0x0020:  0000 0000 0000 0001 8000 168e 0003 0000  ................
        0x0030:  ab23 0215 0000 0000 0000 0000 0000 0000  .#..............
        0x0040:  0000 0000 0000 0000 0000 0000 0000 0000  ................
        0x0050:  0000 0000 0000 0000 0000 0000 0000 0000  ................
        0x0060:  0000 0000 0000 0000                      ........
18:13:27.697671 IP6 fe80::5054:ff:fed3:fd94 > ff02::1:ff57:6e8b: ICMP6, neighbor solicitation, who has fe80::50
4:ff:fe57:6e8b, length 32
        0x0000:  6000 0000 0020 3aff fe80 0000 0000 0000  `.....:.........
        0x0010:  5054 00ff fed3 fd94 ff02 0000 0000 0000  PT..............
        0x0020:  0000 0001 ff57 6e8b 8700 b10b 0000 0000  .....Wn.........
        0x0030:  fe80 0000 0000 0000 5054 00ff fe57 6e8b  ........PT...Wn.
        0x0040:  0101 5254 00d3 fd94                      ..RT....
18:13:27.697929 IP6 fe80::5054:ff:fe57:6e8b > fe80::5054:ff:fed3:fd94: ICMP6, neighbor advertisement, tgt is fe
0::5054:ff:fe57:6e8b, length 32
        0x0000:  6000 0000 0020 3aff fe80 0000 0000 0000  `.....:.........
        0x0010:  5054 00ff fe57 6e8b fe80 0000 0000 0000  PT...Wn.........
        0x0020:  5054 00ff fed3 fd94 8800 8ec0 6000 0000  PT.........`...
        0x0030:  fe80 0000 0000 0000 5054 00ff fe57 6e8b  ........PT...Wn.
        0x0040:  0201 5254 0057 6e8b                      ..RT.Wn.
18:13:27.697937 IP6 fe80::5054:ff:fed3:fd94 > fe80::5054:ff:fe57:6e8b: ICMP6, echo reply, seq 0, length 64
        0x0000:  6005 7016 0040 3a40 fe80 0000 0000 0000  `.p..@:@........
        0x0010:  5054 00ff fed3 fd94 fe80 0000 0000 0000  PT..............
        0x0020:  5054 00ff fe57 6e8b 8100 c854 0003 0000  PT...Wn....T....
        0x0030:  ab23 0215 0000 0000 0000 0000 0000 0000  .#..............
        0x0040:  0000 0000 0000 0000 0000 0000 0000 0000  ................
        0x0050:  0000 0000 0000 0000 0000 0000 0000 0000  ................
        0x0060:  0000 0000 0000 0000                      ........
18:13:28.697806 IP6 fe80::5054:ff:fe57:6e8b > ff02::1: ICMP6, echo request, seq 1, length 64
        0x0000:  6005 39ba 0040 3a01 fe80 0000 0000 0000  `.9..@:.........
        0x0010:  5054 00ff fe57 6e8b ff02 0000 0000 0000  PT...Wn.........
        0x0020:  0000 0000 0000 0001 8000 364a 0003 0001  ..........6J....
        0x0030:  7c66 1115 0000 0000 0000 0000 0000 0000  |f..............
        0x0040:  0000 0000 0000 0000 0000 0000 0000 0000  ................
        0x0050:  0000 0000 0000 0000 0000 0000 0000 0000  ................
        0x0060:  0000 0000 0000 0000                      ........
18:13:28.697838 IP6 fe80::5054:ff:fed3:fd94 > fe80::5054:ff:fe57:6e8b: ICMP6, echo reply, seq 1, length 64
        0x0000:  6005 7016 0040 3a40 fe80 0000 0000 0000  `.p..@:@........
        0x0010:  5054 00ff fed3 fd94 fe80 0000 0000 0000  PT..............
        0x0020:  5054 00ff fe57 6e8b 8100 e810 0003 0001  PT...Wn.........
        0x0030:  7c66 1115 0000 0000 0000 0000 0000 0000  |f..............
        0x0040:  0000 0000 0000 0000 0000 0000 0000 0000  ................
        0x0050:  0000 0000 0000 0000 0000 0000 0000 0000  ................
        0x0060:  0000 0000 0000 0000                      ........
```

ICMPv6-трафик виден машинам в текущей сети

libpcap, кроме непосредственного задания фильтров по содержимому / заголовкам протоколов, также позволяет использование логических операторов НЕ, ИЛИ и И в выражениях и задание очередности их применения с помо-

щью скобок. Полный список доступных возможностей и примеров их использования для задания сложных фильтров доступен в `pcap-filter(7)`.

Сохранение трафика

Сохраним трафик в файл. Повторим передачу данных с помощью netcat, но на этот раз выполним ее по TCP (для этого достаточно убрать ключ `-u` в соответствующих командах netcat):

```
taehyung % nc -p 10001 -l
nayeon % nc 192.168.101.2 10001
```

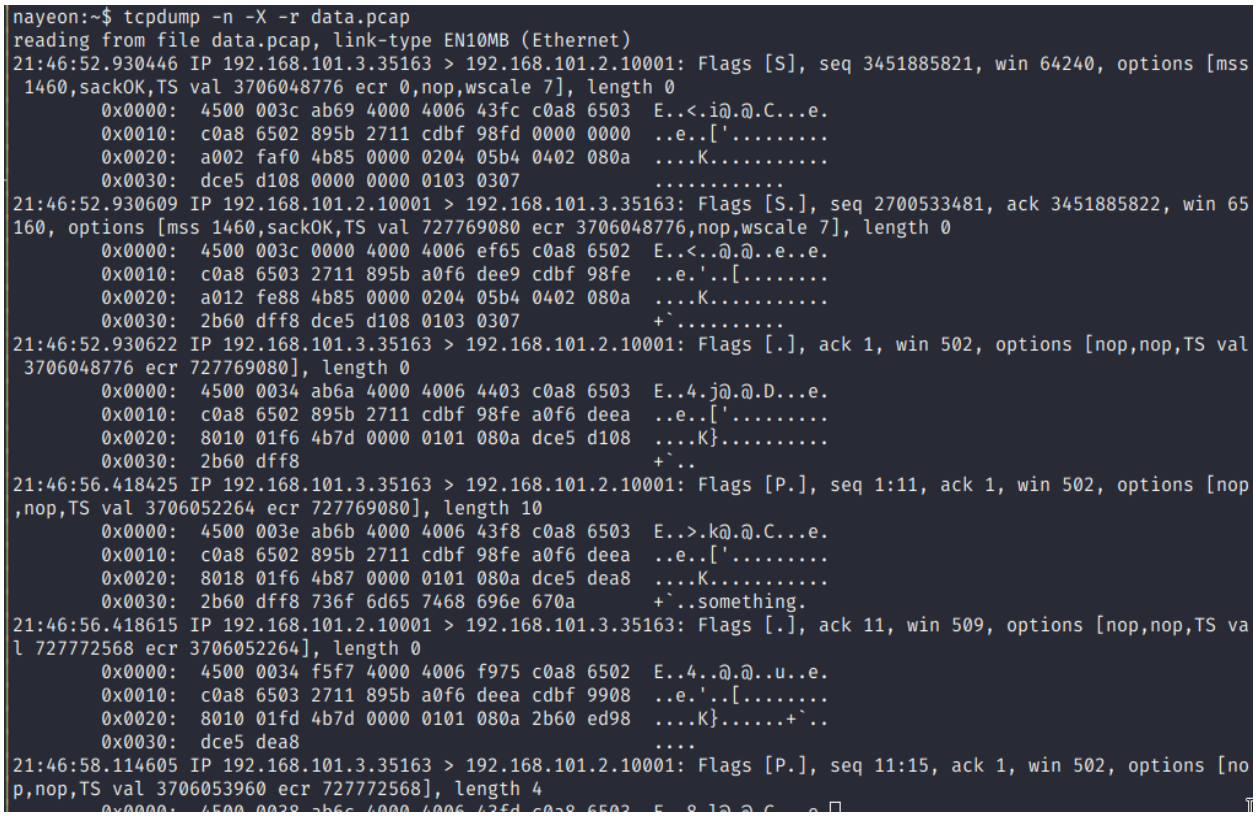
Запустим на Nayeon дополнительно tcpdump с записью в файл `data.pcap`:

```
nayeon # tcpdump -i eth1 -n -w data.pcap -X 'tcp port 10001'
```

Передадим произвольные сообщения. Завершим tcpdump, после чего прочитаем записанный трафик:

```
nayeon % tcpdump -n -X -r data.pcap
```

Увидим TCP-трафик открытым текстом (пакетов так много, потому что в TCP установление и закрытие соединения состоят из трех пакетов, а на каждый присланный пакет данных приходит подтверждение получения):



TCP-трафик

Файл, записанный в формате libpcap, можно позднее открыть в Wireshark / TShark для большего удобства анализа. Консольный интерфейс tcpdump для этого достаточно рудиментарен.

Настройка VPN

Технология VPN, позволяющая, вообще говоря, организовать виртуальную сеть произвольного назначения, будет использоваться для шифрования трафика между Nayeon и Taehyung и создания новой виртуальной локальной сети.

Применения технологии VPN в общем случае неограниченны; например, с VPN можно:

- создать новую виртуальную локальную сеть поверх уже существующего соединения и соединяться с компьютерами в этой сети;
- соединить две уже существующих удаленных друг от друга локальных сети;
- получить удаленный доступ к локальной сети;

- создать виртуальную сеть с роутером и организовать через нее NAT и пересылку трафика;
- и многое другое.

В Linux VPN может работать либо на канальном уровне OSI (посредством создания виртуального TAP-устройства, поверх которого шлются Ethernet-фреймы), либо на сетевом уровне (посредством создания TUN-устройства, поверх которого идут IP-пакеты).

Здесь мы проложим шифрованный туннель между двумя машинами, воспользовавшись WireGuard, недавно (весной 2020 года) стабилизированной реализацией туннеля точка-точка для Linux, нацеленной на простую замену IPSec. WireGuard на момент написания данного текста также имеет реализацию в виде модуля ядра для OpenBSD и реализации, работающие в пространстве пользователя, для всех основных ОС.

Установка WireGuard

WireGuard на текущий момент находится в community-репозитории Alpine Linux. Для его установки необходимо предварительно разрешить установку пакетов из соответствующего репозитория. Для этого откроем текстовым редактором файл `/etc/apk/repositories` и раскомментируем (удалим символ `#`) строку, соответствующую репозиторию community (здесь `MIRROR` означает используемое вами зеркало):

```
http://MIRROR/v3.12/community
```

Установим WireGuard на Nayeon и Taehyung как модуль ядра, а также установим необходимые для его конфигурации утилиты пространства пользователя:

```
# apk add wireguard-tools-wg wireguard-virt
```

Конфигурация туннеля

Новой частной сетью станет `10.0.200.0/24`, в которой Taehyung будет иметь адрес `10.0.200.2`, а Nayeon `10.0.200.3`. Taehyung в нашей схеме будет «сервером» VPN, при этом с точки зрения ядра на узлах туннеля сеть будет одноранговой; решения о маршрутизации в виртуальной сети выполняет непо-

средственно модуль ядра WireGuard. Более подробно о механике маршрутизации можно почитать в документации на сайте WireGuard.

На обеих машинах сгенерируем ключи шифрования:

```
% wg genkey | tee priv.key | wg pubkey > pub.key
```

Создадим файл конфигурации WireGuard `/etc/wireguard/wg0.conf` на Taehyung следующего содержания:

```
[Interface]
ListenPort = 40001 # порт, на котором будет запущен слушатель входящих соединений
PrivateKey = … # содержимое файла priv.key на Taehyung
[Peer]
PublicKey = … # содержимое файла pub.key на Nayeon
AllowedIPs = 10.0.200.3/32 # IP-адрес Nayeon в виртуальной сети, используется
WireGuard для определения, куда должен направляться пакет
```

После чего в файл `/etc/network/interfaces` добавим следующее описание интерфейса, в котором сетевой интерфейс WireGuard создается вручную:

```
auto wg0
iface wg0 inet static
    address 10.0.200.2
    netmask 255.255.255.0
    pre-up ip link add dev $IFACE type wireguard # создаем интерфейс типа wireguard
    pre-up wg setconf $IFACE /etc/wireguard/$IFACE.conf # устанавливаем конфигурацию
    post-up ip route add 10.0.200.0/24 dev $IFACE # добавляем запись в таблицу ро-
утинга
    post-down ip link delete dev $IFACE # подчищаем за собой при деконфигурации
```

Выполним аналогичные шаги на Nayeon с заменой информации в файлах конфигурации там, где это релевантно, но при этом описание пира в файле конфигурации WireGuard зададим следующим:

```
[Peer]
PublicKey = … # содержимое файла pub.key на Taehyung
Endpoint = 192.168.101.2:40001 # IP-адрес и порт пира, на котором слушает WireGuard
AllowedIPs = 10.0.200.0/24 # весь трафик в эту подсеть уходит через этого пира
```

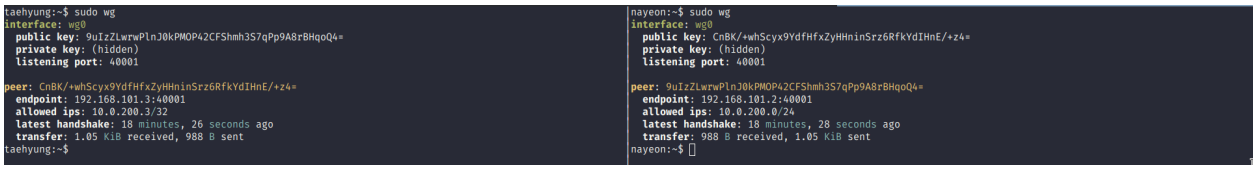
Перезагрузим обе машины.

Примечание: вообще говоря, приватные ключи шифрования должны храниться в секрете и быть доступны на чтение исключительно пользователю root (либо пользователю, с правами которого работает ПО, работающее с ними), но в целях простоты здесь это опускается.

Использование туннеля

Проверим конфигурацию на обеих машинах, выполнив:

```
# wg
```



Статус WireGuard на обеих машинах

Попробуем пропинговать машины друг с друга по адресам в виртуальной сети:

```
nayeon % ping 10.0.200.2
taehyung % ping 10.0.200.3
```

Машины соединяются друг с другом поверх шифрованного туннеля

Поскольку WireGuard — туннель «точка-точка», при необходимости передачи трафика до других машин, связанных через одного из пиров, достаточно указать соответствующие записи в таблице маршрутизации. В нашей конфигурации WireGuard Taehyung является сервером, через который Nayeon пересылает трафик до остальных клиентов. При этом клиентам достаточно иметь лишь один узел с конфигурацией, аналогичной Nayeon; лишь сервер должен иметь информацию обо всех узлах, с которыми идет связь.

Отчет по работе

В отчет требуется поместить следующее:

1. результаты пересылки не зашифрованного TCP- и UDP-трафика между машинами, с релевантным дампом трафика;
2. результат чтения сохраненного файла в формате pcap tcpdump-ом;
3. конфигурацию WireGuard и сетевого интерфейса с обеих машин;
4. результат проверки соединения через туннель и вывод `wg show`;
5. схему результирующей виртуальной сети с точки зрения ядра Linux и WireGuard.

Контрольные вопросы

1. Опишите применение пакетных фильтров.
2. Почему передача данных без шифрования трафика является плохой идеей?
3. Назовите применения технологии виртуальных частных сетей.
4. Назовите известные вам протоколы/ПО для реализации VPN.
5. Предложите возможную архитектуру site-to-site VPN с использованием WireGuard.

ЗАКЛЮЧЕНИЕ

В данном методическом пособии были рассмотрены основные задачи с точки зрения используемого программного обеспечения, с которыми может столкнуться системный администратор при настройке локальной вычислительной сети.

В сравнении с Unix-подобными операционными системами системы MS Windows в контекстах малой ЛВС, рассмотренных выше, могут показаться проще в развертывании и использовании для неподготовленного администратора / пользователя. Тем не менее, стоит учесть, что все программное обеспечение, рассмотренное в методическом пособии (за исключением гипервизора Hyper-V), является свободным и открытым.

Использованные ОС на ядре Linux полностью заменяют проприетарные решения. Главным требованием для их разворачивания будет подготовка и администратора, и пользователей к данному переходу. Это означает, что предприятие, решившееся на переход и предварительно хорошо его спланировавшее, может довольно ощутимо сэкономить на лицензиях программного обеспечения на рабочих станциях и серверах.

Разумеется, сетевое программное обеспечение в данном переходе — лишь верхушка айсберга. В этот переход нужно будет также заложить и различное прикладное ПО. Впрочем, к теме данного методического пособия это уже никак не относится.

СПИСОК ИСТОЧНИКОВ

1. Virtualization Documentation // Microsoft Docs URL: https://docs.microsoft.com/en-us/virtualization/
2. libvirt: Documentation // libvirt URL: https://libvirt.org/docs.html
3. nftables wiki // nftables URL: https://wiki.nftables.org
4. Alpine Linux Wiki URL: https://wiki.alpinelinux.org/wiki/Main_Page
5. WireGuard: fast, modern, secure VPN tunnel // WireGuard URL: https://www.wireguard.com
6. IPv6 configuration in Alpine Linux, Debian and GNS3 Docker // Bernhard`s Homepage URL: https://www.b-ehlers.de/blog/posts/2017-10-17-ipv6-alpine-debian-gns3/
7. Using dnsmasq for dhcpv6 // Tor Hveem URL: https://hveem.no/using-dnsmasq-for-dhcpv6

В данном методическом пособии для выполнения иллюстраций используется свободный набор иконок Citrix из набора свободного графического редактора draw.io.