Дистрибутив фрюхи часто называют самым подходящим для решения прикладных сетевых задач в локальной сети. Сегодня мы займемся решением одной из сетевых задач — настройкой шлюза на Freebsd 10 для доступа в интернет из локалки. Это простой, популярный и востребованный функционал сервера, который можно расширять дополнительными возможностями.

Содержание:

Введение

Будем использовать следующую версию системы для решения нашей задачи по настройке шлюза:

```
# uname -v
FreeBSD 10.2-RELEASE-p8 #0 r292756M: Sat Dec 26 22:49:34 MSK 2015 root@freebsd:/usr/obj/usr/src/sys/GENERIC
```

Если вы еще не установили систему, то рекомендую мою статью с подробным описанием установки Freebsd 10 с видео в конце. Если вдруг будете устанавливать в качестве виртуальной машины на Hyper-V, то у меня есть отдельная заметка на эту тему с описанием поддержки ядром freebsd гипервизора Hyper-V.

На сервере установлены 2 сетевые карты:

- **hn0** — внешний интерфейс, получает интернет от провайдера, настройки по dhcp
- **hn1** — локальная сеть, адрес 10.20.30.1, установлен вручную

В нашу задачу по настройке программного freebsd роутера будет входить настройка маршрутизации на сервере, установка и настройка ipfw, включение nat, настройка локального dhcp и dns сервера.

Если у вас недостаточно опыта и вы не чувствуете в себе сил разобраться с настройкой шлюза самому с помощью консоли сервера — попробуйте дистрибутив на основе centos для организации шлюза и прокси сервера в локальной сети — clearos. С его помощью можно через браузер настроить весь необходимый функционал. В отдельной статье я подробно рассказал о настройке и установке clearos.

Подготовка сервера к настройке шлюза

Подробно вопрос настройки сервера Freebsd 10 я рассмотрел отдельно. Рекомендую ознакомиться с материалом. Здесь я не буду останавливаться на нюансах, а просто приведу команды, которые нам необходимы, без комментариев и подробных пояснений.

Любую настройку я предпочитаю начинать с обновления системы. Выполним его:

```
# freebsd-update fetch
# freebsd-update install
```

Теперь установим mc, я привык работать в нем:

```
# pkg install mc
```

Включаем синхронизацию времени. Для этого добавляем в /etc/rc.conf

```
ntpd_enable="YES"
```

И запускаем демон ntpd:

```
# service ntpd start
```

Теперь проверим сетевые настройки. Первый сетевой интерфейс я настроил еще во время установки, поэтому доступ в интернет на сервере уже есть. Вот мой конфиг сети:

```
hostname="freebsd"
ifconfig_hn0="SYNCDHCP"
ifconfig_hn1="inet 10.20.30.1 netmask 255.255.255.0"
defaultrouter="192.168.1.1"
```

Не забываем добавить dns сервер в */etc/resolv.conf*.

> Подготовительный этап завершен. Прежде чем двигаться дальше, я рекомендую проверить все, что было настроено тут, чтобы потом не отвлекаться. Убедитесь, что на самом сервере есть интернет, что он доступен в локальной сети — пингует другие машины и они его. После этого двигайтесь дальше.

Настройка ipfw и ядерного nat на Freebsd 10

Для того, чтобы включить firewall и nat на будущем freebsd маршрутизаторе можно просто подгрузить необходимые модули ядра. Так сделать проще всего и быстрее, но есть нюанс. Как только вы загружаете модуль ipfw, вы теряете доступ к серверу. Если вы работаете не за консолью, а скорее всего это так, тем более если у вас вообще нет доступа к консоли сервера, то делать так не рекомендуется по понятным причинам. Поэтому мы поступим по-другому. Добавим необходимые опции в ядро, в том числе и опцию, которая делает по-умолчанию фаервол открытым, соберем новое ядро и установим его. Так мы не потеряем удаленный доступ к серверу.

Перед сборкой ядра обновим исходники системы. Я предпочитаю это делать через **svn**. Если он у вас не установлен, то установите:

```
# pkg install subversion
```

Теперь обновляем исходники:

```
# svn checkout https://svn0.ru.freebsd.org/base/release/10.2.0 /usr/src
```

Копируем стандартное ядро и добавляем туда новые строки:

```
# cp /usr/src/sys/amd64/conf/GENERIC /usr/src/sys/amd64/conf/GATE
```

```
options IPFIREWALL # включаем ipfw
options IPFIREWALL_DEFAULT_TO_ACCEPT # делаем по-умолчанию ipfw открытым
options IPFIREWALL_VERBOSE # включение логирования ipfw
options IPFIREWALL_VERBOSE_LIMIT=50 # ограничение на количество записей в лог от одного правила
options IPDIVERT # для форварда пакетов
options IPFIREWALL_NAT # для nat
options LIBALIAS # просто надо, не помню зачем
```

Собираем и устанавливаем новое ядро:

```
# cd /usr/src
# make kernel -s -j4 KERNCONF=GATE
```

Добавляем в файл */boot/loader.conf* строку:

```
# mcedit /boot/loader.conf
net.inet.ip.fw.one_pass=1
```

Перезагружаем сервер:

```
# reboot
```

Добавляем в /etc/rc.conf следующие строки (комментарии уберите):

```
gateway_enable="YES" # разрешаем пересылку пакетов между сетевыми интерфейсами
firewall_nat_enable="YES" # включаем ядерный nat
firewall_nat_interface="hn0" # указываем внешний интерфейс для nat
firewall_enable="YES" # включаем ipfw
firewall_script="/usr/local/etc/ipfw/rc.firewall" # указываем путь, где будут лежать настройки ipfw
```

Идем в указанную папку и создаем там файл для правил ipfw:

```
# mkdir /usr/local/etc/ipfw && touch /usr/local/etc/ipfw/rc.firewall
```

Содержание этого конфига приведу в виде ссылки на скачивание уже готового файла с подробными комментариями. Думаю, вы без проблем разберетесь в настройках. Здесь не хочу на этом подробно останавливаться, все же тема настройки ipfw это отдельный разговор, у нас задача настроить роутер. Вот готовый набор правил — rc.firewall. Не забудьте указать свои интерфейсы и ip адреса. Если ошибетесь, потеряете доступ к серверу. Когда все настроите, отключите логирование. Там по-умолчанию стоит запись логов на правила nat и deny all.

> Я не рекомендую настраивать firewall, не имея доступа к консоли сервера. Насколько бы вы ни были опытными администраторами, шанс ошибиться есть всегда. Хотя у меня многие вещи доведены до автоматизма и я настраиваю по отработанным шаблонам, все равно иногда возникают ситуации, когда я ошибаюсь и теряю доступ к серверу. Так что в любом случае, при конфигурировании фаервола, убедитесь, что сможете подключиться к серверу и исправить ошибку в случае необходимости.

Копируем скрипт безопасного редактирования правил и делаем его исполняемым:

```
# cp /usr/share/examples/ipfw/change_rules.sh /usr/local/etc/ipfw_change_rules
```

```
# chmod 0700 /usr/local/etc/ipfw_change_rules
```

Подробно о том, что это за скрипт и как он работает я рассказал в теме по настройке freebsd, ссылку на которую давал в начале. Активируем новые правила запуском скрипта:

```
# /usr/local/etc/ipfw_change_rules
```

Откроется список правил в редакторе по-умолчанию. Там уже будут все правила. Можете еще раз их проверить и если все в порядке, сохраняете файл и выходите из редактора. В консоли вы увидите следующее:

Проверим примененные правила:

```
# ipfw show
```

Если доступ к роутеру не потеряли, значит все в порядке. Можно перезагрузить сервер и все добавленные параметры в rc.conf активируются. Сделайте это и попробуйте на каком-нибудь компьютере в сети пропинговать адрес в интернете по ip, к примеру 8.8.8.8. Если пинги пройдут, значит все в порядке.

Поле того, как убедитесь, что ipfw настроен корректно, правила применятся и доступ к серверу есть, можете перевести его из открытого режима работы (последнее правило автоматически устанавливается allow all from any to any) в закрытый. Для этого в файл */boot/loader.conf* добавьте строку и перезагрузите сервер:

```
net.inet.ip.fw.default_to_accept=0
```

На этом основная настройка шлюза на Freebsd 10 окончена. Клиенты смогут выходить в интернет. Но для удобства необходимо на шлюз установить и настроить **dhcp** и **dns** сервер, для обслуживания запросов пользователей. Иначе придется вручную забивать сетевые параметры и использовать сторонний dns сервер.

Установка и настройка dnsmasq

Для нашего роутера на freebsd подойдет любой dns и dhcp сервер. Можно использовать традиционные named и dhcp-server. Но для простоты и удобства, когда не нужен дополнительный функционал, я предпочитаю использовать простой и быстрый в настройке **dnsmasq**.

Устанавливаем dnsmasq на Freebsd шлюз:

```
# pkg install dnsmasq
```

Приводим конфиг к следующему виду:

```
# mcedit /usr/local/etc/dnsmasq.conf
```

```
domain-needed
bogus-priv
interface=hn1
resolv-file=/etc/resolv.conf
dhcp-range=10.20.30.100,10.20.30.200,24h
```

Добавляем в */etc/rc.conf*:

```
dnsmasq_enable="YES"
```

Запускаем программу:

```
# /usr/local/etc/rc.d/dnsmasq start
```

Все, теперь наш шлюз полностью готов. Настраиваем на клиентах получение настроек по dhcp и проверяем работу интернета.

Информацию о выданных leases dhcp сервера dnsmasq можно посмотреть в файле */var/db/dnsmasq.leases*.

Анализ сетевой активности в freebsd с помощью iftop

Иногда хочется посмотреть, что происходит на роутере и кто использует интернет в данный момент. По-умолчанию, в системе нет готового средства для получения этой информации. На помощь нам придет простая программа iftop, которая позволяет в режиме реального времени посмотреть активность на сетевом интерфейсе.

Устанавливаем **iftop** на настроенный Freebsd шлюз:

```
# pkg install iftop
```

Запускаем iftop с указанием интерфейса и отображением используемых портов:

```
# iftop -i hn1 -P
```

Видим любопытную картину — кто, куда, по какому порту и с какой скоростью лезет.

Я для примера на одном из компьютеров запустил генератор трафика интернета. Он занял почти весь канал и это стало отлично видно на роутере с помощью iftop. Конечно, эта простая утилита не решает всех вопросов по мониторингу сетевой активности, но для представления текущей картины подходит, если вам не нужно что-то большее.

Заключение

Подведем итог того, что сделали. За короткое время настроили полноценный шлюз (по сути программный роутер) на базе Freebsd 10 для обеспечения выхода в интернет клиентов за сервером. При этом обеспечили автоматическое получение настроек. Даже на скромном виртуальном сервере такой шлюз способен переварить достаточно большой траффик.

Вся настройка занимает буквально 10-15 минут. Основное время уходит на сборку ядра. Чем выше версия Freebsd, тем дольше оно собирается, несмотря

на то, что скорости железа существенно возрастают.

Пройдемся по пунктам и разберемся с тем, что конкретно мы сделали:

1. Подготовили сервер к настройке шлюза.
2. Пересобрали ядро с необходимыми параметрами.
3. Настроили ipfw и nat, включили маршрутизацию.
4. Установили и настроили dnsmasq для раздачи сетевых настроек по dhcp и dns сервера.
5. Установили iftop для простейшего анализа сетевой активности на внешнем интерфейсе.

Этого достаточно для полноценной работы шлюза на Freebsd 10. Если есть необходимость подсчета пользовательского траффика или ограничения доступа к определенным ресурсам, можно настроить прокси сервер squid и sams2 к нему.

Помогла статья? Есть возможность отблагодарить автора

Дополнительные материалы по Freebsd

Рекомендую полезные материалы по Freebsd:

- Установка
- Настройка
- Обновление
- Шлюз
- Прокси сервер
- Веб сервер NGINX
- Веб сервер Apache

Описание установки Freebsd 11 на одиночный диск, либо на софтовый raid1, сделанный средствами zfs, которые поддерживает стандартный установщик.
Базовая настройка Freebsd, которую можно выполнить после установки сервера общего назначения. Представлены некоторые рекомендации по повышению удобства пользования и безопасности.
Описание и нюансы обновления системы Freebsd с помощью утилиты freebsd-update. Показано пошагово на конкретном примере обновления.
Настройка Freebsd шлюза для обеспечения выхода в интернет. Используется ipfw и ядерный нат, dnsmasq в качестве dhcp и dns сервера. Мониторинг сетевой активности с помощью iftop.
Подробная настройка на Freebsd прокси сервера squid + sams2 - панели управления для удобного администрирования.
Настройка максимально быстрого web сервера на базе Freebsd и nginx + php-fpm. Существенный прирост производительности по сравнению с классическим apache.
Настройка web сервера на Freebsd в связке с apache, nginx, php и mysql. Пошаговая установка и настройка каждого компонента.