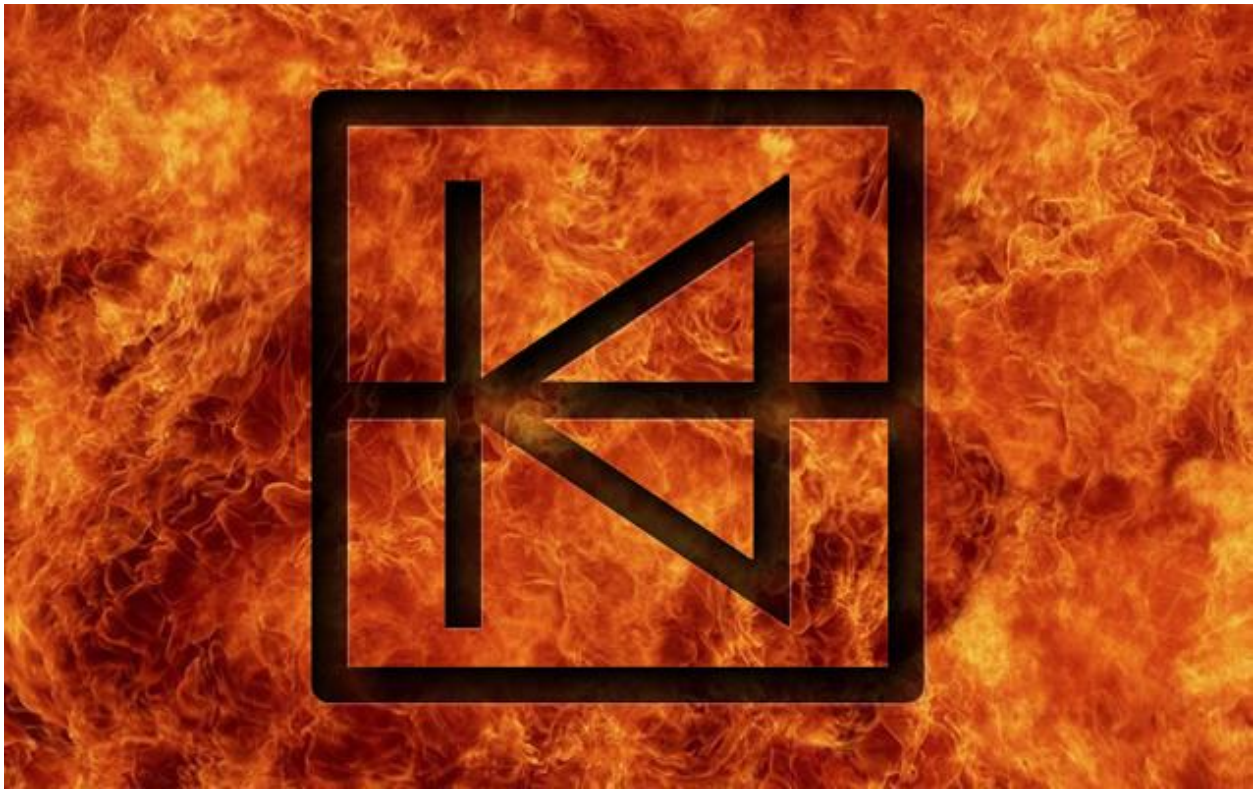
Гайд по межсетевому экранированию (nftables)



Все говорят, что для защиты сети нужно применять межсетевые экраны, но никто не говорит, как это нужно делать. Что ж, исправим ситуацию, рассмотрим типовые сценарии применения межсетевых экранов и то, как их при этом настраивать.

В качестве межсетевого экрана будем использовать nftables, функционирующий под управлением ОС Debian GNU Linux.

Оглавление

Требования к предварительным знаниям 2

Технические требования 2

Создание шаблона виртуальной машины 2

Типовой сценарий: защита сервера / рабочей станции с помощью локального брандмауэра 4

Модель угроз 5

Схемы разграничения трафика 6

Практическая работа: реализация межсетевого экранирования сервера по схеме усиленной защищенности 8

Типовая задача: проверка работы брандмауэра 15

Практическая работа: проверка работы брандмауэра 16

Типовой сценарий: сегментирование локальной сети маршрутизатором с функцией брандмауэра 18

Практическая работа: разграничение трафика между пользовательским и серверным сегментами 20
Типовой сценарий: сегментирование локальной сети коммутатором с функцией брандмауэра 23
Практическая работа: провести сегментирование сети без разделения ее на IP-подсети 24
Проект OneButtonFirewall 26
Практическая работа: защитить сегмент сети с помощью межсетевого экрана, построенного по технологии OneButtonFirewall 27
Реализация OneButtonFirewall в виде аппаратного устройства 30
Преимущества и недостатки OneButtonFirewall 31
Сценарии применения 31
Заключение 32

Требования к предварительным знаниям

Здесь мы не будем рассматривать принципы работы nftables, синтаксис его командной строки или формат конфигурационных файлов. Об этом есть множество других статей. Мы же сосредоточимся на его практическом использовании. По большому счету все необходимые настройки nftables будут в статье подробно описаны, так что проблем с их воспроизведением, по идее, быть не должно. Тем не менее для лучшего понимания процесса рекомендуется ознакомиться и держать под рукой следующие ресурсы:

- Man по nftables
- Официальный nftables wiki
- Статья на Хабре: «Используем nftables в Red Hat Enterprise Linux 8»

Технические требования

Для построения лабораторных стендов можно использовать любую удобную для вас систему виртуализации: VMware, Hyper-V, VirtualBox, QEMU-KVM или другую. Главным требованием к этой системе будет наличие возможности управления виртуальными сетями. Система должна позволять создавать виртуальные сети, а также назначать их на различные адаптеры виртуальных машин.

Создание шаблона виртуальной машины

В большинстве лабораторных стендов нам потребуется виртуальная машина-шаблон,

которую мы будем тиражировать и донастраивать. Для создания этого шаблона необходимо:

1. Скачать Debian 11.4 в формате netinstall .
2. Установить ОС в минимальной конфигурации. Для этого во время инсталляции выбираем все параметры по умолчанию (Next, Next, Next, ...), а на шаге с выбора пакетов «Software selection» отказываемся от всего предлагаемого (Рисунок 1).

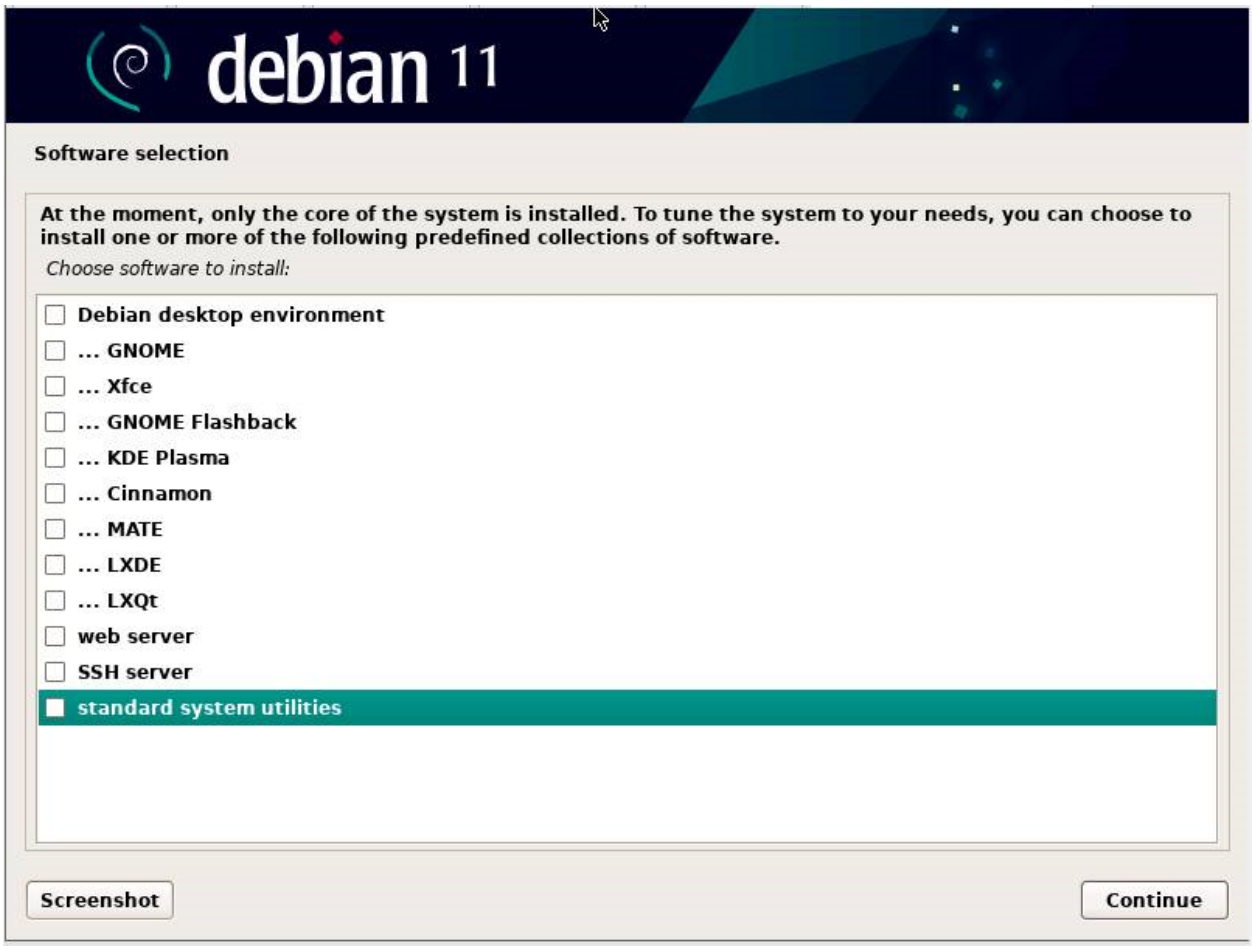


Рисунок 1

3. (Опционально). Рекомендуется установить пакет расширений средств виртуализации для гостевой операционной системы. Например, для VMware – это будет VMware Tools, для VirtualBox – Virtualbox extension pack и т.д. После этого рекомендуется настроить разделяемый каталог (Shared Folder) для связи между файловыми системами виртуальной машины и гипервизора.
4. (Опционально). Для повышения удобства выполнения практических работ рекомендуется поставить дополнительные пакеты:

    - OpenSSH сервер (пакет ssh).
    - Файловый менеджер Midnight Commander (пакет mc).

- Утилиту conntrack, демонстрирующую внутреннее состояние брандмауэра и помогающую в отладке правил фильтрации (пакет conntrack).

  Все эти пакеты можно легко поставить. Для этого достаточно зайти на машину под root'ом (при конфигурировании Linux нам всегда потребуется root) и выполнить в консоли заклинание (команду):

```
apt install mc ssh conntrack
```

Типовой сценарий: защита сервера / рабочей станции с помощью локального брандмауэра

Задача настроить локальный брандмауэр — пожалуй, самая распространённая задача межсетевого экранирования. Для ее решения мы должны знать ответы на два вопроса: чего мы хотим добиться, и как это реализовать.

Ответ на первый вопрос должен дать архитектор информационной безопасности, он должен сформулировать угрозы и предложить методы их нейтрализации. Проблема в том, что в небольших коллективах людей, играющих такую роль, нет, и всем этим приходится заниматься системным администраторам. Несмотря на это, все же рекомендуется действовать по стандартному алгоритму, то есть так, как бы действовал архитектор.

Данный алгоритм может показаться забюрократизированным, однако, это не совсем так, ведь большинство шагов нужно будет сделать лишь раз, а затем выполнять работы по уже сформированным шаблонам. Несомненным плюсом стандартного алгоритма является обеспечение доказательного уровня безопасности, при котором все предпринимаемые меры защиты четко обоснованы и могут быть легко верифицированы. Все вместе это повышает уровень зрелости компании в части обеспечения информационной безопасности и несет множество сопутствующих плюшек.

Стандартный алгоритм решения задач межсетевого экранирования:

1. Формулирование модели угроз.
2. Анализ угроз и выработка мер по их нейтрализации. На данном этапе необходимо сформулировать схему разграничения трафика: то есть какой трафик пропускаем, а какой блокируем.
3. На основании схемы разграничения трафика производится настройка межсетевого экрана: формируются правила фильтрации, настраиваются сервисы, пишутся скрипты и так далее.

Моделирование угроз – процесс нетривиальный и требующий значительных компетенций в области информационной безопасности. Фактически необходимо знать то, как вас будут пытаться взломать. Множество способов атак уже описано, но постоянно появляются все новые и новые. Специалисты, профессионально занимающиеся моделированием угроз, должны постоянно держать руку на пульсе и быть в курсе всех модных новинок. Для всех остальных существенным подспорьем будут готовые модели и банки данных угроз, например, MITRE ATT&CK Matrix, Банк данных угроз ФСТЭК России или публичные исследования, например, это, это и это.

Модель угроз

Применительно к нашей задаче мы будем рассматривать следующие угрозы:

Угроза	Комментарий
У1. Угроза удаленной эксплуатации уязвимостей (как программных, так и конфигурационных) в сетевых сервисах узла.	Примеры реализации угрозы: эпидемия червя MSBlast (эксплуатация программной уязвимости) или множественные утечки данных с кластеров Elasticsearch, происходящие из-за отсутствия авторизации (эксплуатация конфигурационной уязвимости).
У2. Угроза удаленной атаки на отказ в обслуживании (DoS) сетевых сервисов узла.	
У2.1. DoS атака за счет превышения критического числа сетевых запросов, которые может обработать сетевой сервис с приемлемым качеством	Подобные атаки наиболее актуальны для серверов баз данных, которые могут генерировать существенную вычислительную нагрузку на обработку входящего запроса.
У2.2. DoS атака за счет подделки злоумышленником IP-пакета и установки в нем обратного адреса равному адресу петли 127.0.0.0/8	Сетевой сервис, получая подобный запрос, при отправке ответа может получить его себе обратно, что может привести к зацикливанию или другим негативным последствиям.
У3. Угроза раскрытия аутентификационных данных за счет удаленных атак прямого перебора, осуществляемых в отношении сетевых сервисов узла.	Тривиальный подбор паролей, но осуществляемый удаленно через сеть.
У4. Угроза установки исходящих вредоносных сетевых соединений локальным программным обеспечением узла.	Вредоносное сетевое соединение может установить как троян, связывающийся со своим сервером управления (C&C), так и легитимная программа, передающая избыточные данные (например, телеметрию) на сервера разработчиков. Кроме того, к данной угрозе можно отнести и действия сотрудников, использующих рабочие компьютеры для личных нужд (например, для майнинга криптовалют).
У5. Угроза несанкционированного открытия сетевого порта вредоносным кодом.	На заре вирусостроения первые шпионские вредоносные программы открывали на машине жертвы сетевые порты, к которым затем подключались лица, занимающиеся шпионажем.

Теперь проанализируем угрозы и сформулируем идеи по их нейтрализации.

Для парирования угрозы **У1** и **У5** необходимо лишить злоумышленников возможности осуществлять подключения к сетевым сервисам узла. Это достигается путем ограничения входящего трафика по портам и адресам источника (белые или черные списки).

Нейтрализация **У2.1** базируется на ограничении скорости получения сервисом сетевых пакетов, а как более сложный вариант — обнаружение атакующих узлов и блокировка их по IP. Следует отметить, что ограничение максимальной частоты подключений не всегда допустимо, особенно для публичных сервисов.

Угроза **У2.2.** нейтрализуется путем отбрасывания всех сетевых пакетов, имеющих адрес источника из подсети 127.0.0.0/8 и пришедших не с петлевого (loopback) интерфейса.

Для нейтрализации угрозы **У3** в общем случае требуется применение дополнительных программ, которые бы определяли факты неудачных попыток авторизации, определяли бы IP адреса атакующих, а затем с помощью брандмауэра блокировали бы их по IP. Реализации защиты от угрозы **У2.1.** частично усложнит злоумышленникам возможность реализации угрозы **У3**.

Борьба с **У4** проводится путем ограничения исходящего сетевого трафика, что может производиться на основании анализа:

- адресов назначения (белые или черные списки);
- протоколов транспортного уровня (TCP/UDP) и номеров их портов (белые или черные списки);
- приложений, пытающихся установить соединение.

Переходя от идей по защите к их реализации, следует помнить базовую аксиому: чем более надежная защита, тем дороже она стоит. Причем эта стоимость выражается не только в затратах на покупку средств защиты, но и трудозатратах на их эксплуатацию особенно со стороны рядовых пользователей. Если защита терроризирует пользователя регулярными запросами или другим образом отвлекает от работы, то он всеми правдами и неправдами будет саботировать ее применение (user resistance). Поэтому всегда нужно находить баланс между защищенностью и удобством. Для этого в некоторых случаях следует принимать (игнорировать) риски маловероятных или незначительных по ущербу угроз информационной безопасности.

На основании всего вышесказанного подготовим несколько схем разграничения трафика, ранжирующийся по степени защищённости и простоты реализации.

Схемы разграничения трафика

Схема № 1. Минимальная защита рабочей станций.

- Запрещен весь входящий трафик, кроме того, что относится к уже установленным соединениям.
- Весь исходящий трафик разрешен.

Данный схема подходит только для пользовательских рабочих станций с минимальными ожиданиями по безопасности. В подобных рабочих станциях нет активных сетевых сервисов (не должно быть), а брандмауэр защищает от несанкционированного открытия сетевых портов вредоносами. Косвенно брандмауэр также реализует «защиту от дурака», заключающуюся в том, что, если пользователь по неосторожности или в тестовых целях установит какой-либо сетевой сервис, то по сети он будет не доступен.

Ограничение исходящего трафика по адресам назначения на рабочих станциях, где активно пользуются Интернетом, крайне трудоемко, а ограничивать трафик по приложениям брандмауэр nftables не умеет. Соответственно, весь исходящий сетевой трафик разрешается без ограничений. Угроза **У3** полностью игнорируется. Как слабенький вариант борьбы с **У3** можно рекомендовать применение встроенной системы мандатного разграничения доступа (MAC) AppArmor, которая позволяет выбранным приложениям отключить сетевые возможности. Проблема в том, что AppArmor работает только по черным спискам, белый список – замкнутую программную среду — с его помощью сделать не получится.

Рассмотренная схема разграничения трафика позволяет полностью нейтрализовать угрозы **У1, У2, У3, У5**, угроза **У4** игнорируется.

По умолчанию встроенный брандмауэр ОС Windows работает по этой схеме.

Схема № 2. Минимальная защита сервера.
Схема практически полностью повторяет предыдущую, за исключением того, что:

- разрешается входящий трафик по выбранным сетевым портам, требуемым для работы сетевым сервисам;
- блокируется входящий трафик для IP-пакетов, адрес источника которых относится к сети 127.0.0.0/8, и которые пришли не с петлевого (loopback) интерфейса.

Ограничение доступа к сетевым сервисам по IP-адресам не производится.

Реализация данной схемы частично защищает от **У1** и полностью от **У2.2** и **У5**, остальные же угрозы игнорируются.

Схема № 3. Стандартная защита серверов.
Ограничение входящего трафика в данной схеме полностью повторяет предыдущую схему.
Исходящий же трафик полностью запрещается, за исключением трафика:

- текущего по уже установленным соединениям;
- отправляемого по перечню явно разрешенных портов протоколов транспортного уровня (белый список).

Подобная схема фильтрации полностью защищает от **У2.2** и **У5**, частично от других угроз, кроме **У2.1**, которая полностью игнорируются.

Схема № 4. Усиленная защита серверов.
Схема основана на предыдущей, но усилена следующими мерами:

- для каждого открываемого сетевого сервиса (при наличии возможности) настраивается:
    - ограничение по максимальной частоте (rate) попыток подключения;
    - ограничение по перечиню узлов, имеющих право на подключение (белый список);
    - использование дополнительных средств обнаружения большого количества неудачных попыток авторизации, после чего нарушителя с помощью брандмауэра блокируют по IP;
- исходящий трафик, помимо портов, ограничивается также и по IP-адресам назначения.

Подобная схема фильтрации дает максимальную защиту от всех рассмотренных угроз.

Практическая работа: реализация межсетевого экранирования сервера по схеме усиленной защищенности

Описание стенда

Проведем нашу первую практическую работу. Начнем с того, что организуем лабораторный стенд по следующей схеме (Рисунок 2):

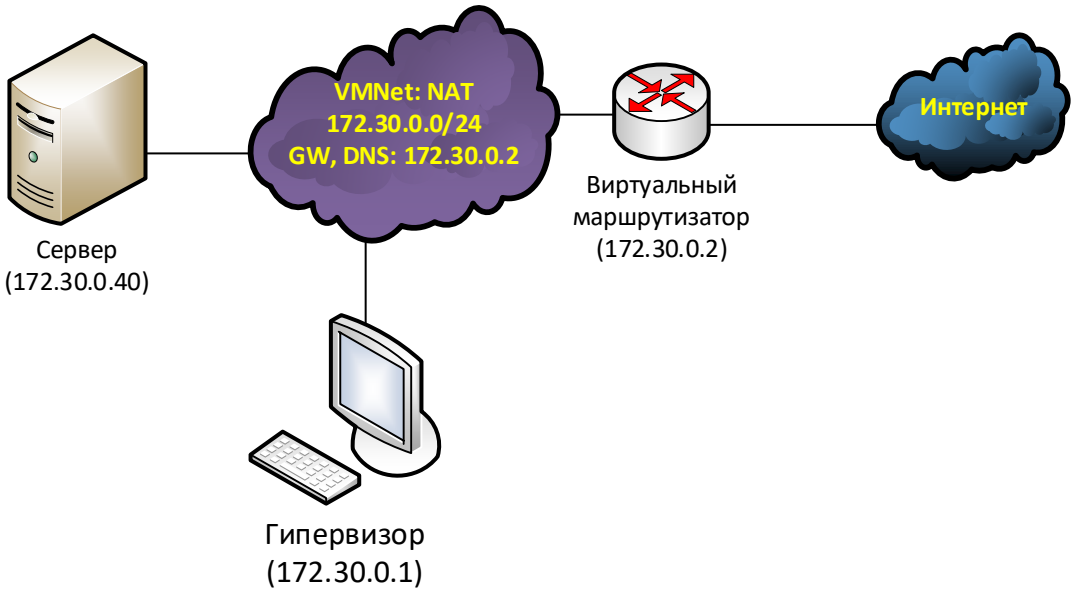


Рисунок 2

Здесь все просто. Есть одна виртуальная машина «Сервер». Она подключена своим единственным сетевым адаптером к виртуальной сети «NAT». В данной сети присутствует виртуальный маршрутизатор, реализующий NAT и выполняющий роль DNS-сервера. Сетевой адаптер гипервизора также подключен к сети «NAT». В качестве защищаемого сервиса на виртуальной машине «Сервер» будет выступать SSH. Для его тестирования на гипервизоре устанавливается SSH-клиент.

Задача

На виртуальной машине «Сервер» настроить межсетевой экран по схеме 4 «Усиленная защита сервера».

Подготовка стенда

1. На виртуальную машину «Сервер» установим OpenSSH сервер и службу автоматической синхронизации системных часов systemd-timesyncd:

```
apt install ssh systemd-timesyncd
```

2. В файле настроек службы синхронизации времени /etc/systemd/timesyncd.conf раскомментируем строки NTP и FallbackNTP. Строку NTP запишем в следующем виде:

```
NTP=0.europe.pool.ntp.org 1.europe.pool.ntp.org 2.europe.pool.ntp.org
3.europe.pool.ntp.org
```

3. Активируем автоматическую синхронизацию времени, выполнив заклинание:

```
timedatectl set-ntp true
systemctl enable --now systemd-timesyncd.service
systemctl restart systemd-timesyncd.service
```

4. Для проверки синхронизации времени нужно перезагрузиться, а затем выполнить заклинание:

```
timedatectl timesync-status
```

5. Настроим статический IP адрес на виртуальной машине «Сервер». Для этого содержимое файла /etc/network/interfaces заменим на следующее:

```
auto lo
iface lo inet loopback

# The primary network interface
auto ens33
iface ens33 inet static
address 172.30.0.40
mask 255.255.255.0
gateway 172.30.0.2
nameserver 172.30.0.2
```

Примечание. В Debian по умолчанию при описании сетевых интерфейсов используется ключевое слово «allow-hotplug». Мы же поменяем его на «auto». Это

сделано для того, чтобы была возможность менять сетевые настройки с помощью заклинания:

```
service networking restart
```

6. Проверим работоспособность стенда. С гипервизора должна быть возможность установить SSH сессию до «Сервера», а с «Сервера» — возможность скачивать обновления с официальных репозиториев и обновлять время по сети.

Ход выполнения работы

1. Логика организации дистрибутива Debian такова, что межсетевой экран nftables установлен в нем по умолчанию. В этом можно убедиться, выполнив заклинание:

```
nft -v
```

2. Для автоматической загрузки правил межсетевого экранирования после старта системы создана служба nftables (по сути являющаяся systemd юнитом), но по умолчанию она деактивирована. В этом можно убедиться, выполнив заклинание:

```
service nftables status
```

Служба организована таким образом, что при запуске она считывает правила, записанные в файле /etc/nftables.conf. Соответственно, цель нашей работы — внести в данный файл необходимые правила и активировать службу.

3. Уточним схему разграничения сетевого трафика. Как часто бывает, те схемы, которые мы описывали ранее, не могут учитывать особенности эксплуатации конкретных серверов, поэтому и требуют уточнения. В частности, сделаем следующее:

- Разрешим «пинговать» (использовать команду проверки сетевой связности ping) защищаемый сервер и разрешим серверу «пинговать» другие узлы. Несмотря на то, что каждый открытый поток сетевого трафика снижает защищенность, открытие «пингов» существенно улучшает эксплуатационные свойства сервера.
  Для этого разрешим входящие и исходящие ICMP echo-request.
- Разрешим отправлять запросы и получать ответы по протоколу DNS.
  Для этого разрешим исходящие соединения по портам: TCP 53 и UDP 53 в адрес явно указанного DNS сервера: 172.30.0.2.
- Разрешим автоматически обновлять системные часы по протоколу NTP.
  Для этого разрешим исходящие соединения по портам: TCP 123 и UDP 123 в адрес серверов, указанных в файле /etc/systemd/timesyncd.conf.
- Разрешим скачивать обновления с репозиториев, указанных в файлах настройки менеджеров пакетов.
  Для этого разрешим исходящие http соединения (TCP 80) в адрес репозиториев, указанных в файле /etc/apt/sources.list.

- Будем явно отбрасывать входящий трафик, превышающий скоростные ограничение в:
    - 5 пакетов в секунду для ICMP;
    - 10 попыток установки соединений в минуту для SSH.
- Узлы, занимающиеся подборкой паролей к SSH, будем определять с помощью утилиты fail2ban. Утилита сама будет конфигурировать nftables для блокирования и разблокирования атакующих узлов.

4. В файл /etc/nftables.conf запишем следующее содержимое:

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

flush ruleset

table ip firewall {
# Список разрешенных DNS серверов
        set allowed-dns-servers {
                type ipv4_addr
                elements = { 172.30.0.2 }
        }

# Список разрешенных узлов, с которых можно подключаться по SSH
        set allowed-ssh-clients {
                type ipv4_addr
                elements = { 172.30.0.1 }
        }

# Список разрешённых NTP-серверов из файла
# /etc/systemd/timesyncd.conf
        set allowed-ntp-servers {
                type ipv4_addr
        }

# Список разрешённых серверов репозиториев из файла
# /etc/apt/sources.list
        set allowed-repos {
                type ipv4_addr
        }

# Цепочка правил фильтрации входящего трафика. Запрещено все,
# кроме того, что явно разрешено (policy drop)
        chain fw_input {
                type filter hook input priority filter; policy drop;

# Разрешен трафик с петлевого (loopback) интерфейса
                iifname "lo" accept

# Явно отбрасываются IP-пакеты с обратным адресом локальной петли,
# но не относящиеся к петлевому интерфейсу
                ip saddr 127.0.0.0/8 drop

# Явно отбрасываем ICMP трафик, превышающий скоростной лимит
                ip protocol icmp limit rate over 5/second drop
```

```
# Явно отбрасываем запросы на соединение по SSH,
# превышающие скоростной лимит
            tcp dport 22 ct state new limit rate over 10/minute drop

# Разрешаем трафик по уже установленным соединениям
                ct state established,related accept

# Разрешаем получение ICMP-эхо запросов (чтобы узел можно
# было пинганути)
                icmp type echo-request accept

# Разрешаем подключение по SSH избранным клиентам
                ip saddr @allowed-ssh-clients tcp dport 22 accept
        }

# Цепочка правил фильтрации исходящего трафика. Запрещено все,
# кроме того, что явно разрешено (policy drop)
        chain fw_output {
                type filter hook output priority filter; policy drop;

# Разрешаем трафик на петлевой интерфейс
                oifname "lo" accept

# Разрешаем трафик по уже установленным соединениям
                ct state established,related accept

# Разрешаем отправку ICMP эхо-запросов (чтобы узел мог пинганути).
                icmp type echo-request accept

# Разрешаем отправку DNS-запросов по UDP
                ip daddr @allowed-dns-servers udp dport 53 accept

# Разрешаем отправку DNS-запросов по TCP
                ip daddr @allowed-dns-servers tcp dport 53 accept

# Разрешаем отправку запросов по протоколу NTP с помощью TCP
                ip daddr @allowed-ntp-servers tcp dport 123 accept

# Разрешаем отправку запросов по протоколу NTP с помощью UDP
                ip daddr @allowed-ntp-servers udp dport 123 accept

# Разрешаем установке HTTP-соединений с серверами репозиториев,
# указанными в файле /etc/apt/sources.list

                ip daddr @allowed-repos tcp dport 80 accept
        }
}
```

Примечание. Опытные администраторы наверно заметили, что в цепочке *fw_input* мы явно отбрасываем icmp пакеты, превышающие установленный скоростной лимит (*ip protocol icmp limit rate over 5/second drop*), а затем пишем правило, разрешающее получать icmp эхо-запросы (*icmp type echo-request accept*). Резонный вопрос: зачем мы это делаем? Ведь политика цепочки — отбрасывать все, что явно не разрешено, и, по идее, кажется, что эти два правила можно заменить одним *icmp type echo-request limit rate 5/second accept*, но в данном случае это не так. Дело в том, что в цепочке присутствует правило *ct state established,related accept*, и из-за него *icmp type echo-request limit rate 5/second*

accept не будет ограничивать скорость. Это происходит потому, что nftables считает «ping» как сетевое соединение. Это можно увидеть с помощью заклинания

```
conntrack -L
```

Соответственно, пакеты, не попавшие в правило *icmp type echo-request limit rate 5/second accept*, будут пропущены правилом *ct state established,related accept*. Вот поэтому и нужно явно отбрасывать пакеты, превышающие скорость, и делать это перед правилом *ct state established,related accept*.

5. Активируем автоматический запуск nftables после загрузки системы. Для этого выполним заклинания:

```
systemctl daemon-reload
systemctl enable nftables
```

6. Для дальнейших работ нам потребуется утилита dig, входящая в пакет dnsutils, и утилита fail2ban, входящая в одноименный пакет. Установим их, выполнив заклинание:

```
apt install dnsutils fail2ban
```

7. Рассмотрим процесс формирования динамических списков allowed-ntp-servers и allowed-repos. Данные списки должны содержать в себе IP-адреса:

   - репозиториев, указанных в файле /etc/apt/sources.list;
   - NTP-серверов, указанных в файле /etc/systemd/timesyncd.conf.

   Если вы просмотрите эти файлы, то никаких IP-адресов там не заметите. Вместо них указаны лишь FQDN-имена требуемых серверов. Проблема в том, что nftables не умеет фильтровать по FQDN-именам, он работает только по IP. Соответственно, необходимо, чтобы кто-то ему перевел из FQDN в IP. Также следует помнить, что FQDN-имена — вещь не постоянная, и процесс перевода их в IP нужно делать периодически. Для извлечения и перевода из данных файлов FQDN в IP-адреса можно воспользоваться скриптом:

```
nft flush set ip firewall allowed-ntp-servers
nft flush set ip firewall allowed-repos

grep -oP '(?<=^deb http://)[^ /]*' /etc/apt/sources.list | uniq  |
xargs dig +short | xargs -r -I IP nft add element ip firewall
allowed-repos { IP }

grep -oP '(?<=^NTP=).+$' /etc/systemd/timesyncd.conf | xargs dig
+short | xargs -r -I IP nft add element ip firewall allowed-ntp-
servers { IP }

grep -oP '(?<=^FallbackNTP=).+$' /etc/systemd/timesyncd.conf | xargs
dig +short | xargs -r -I IP nft add element ip firewall allowed-ntp-
servers { IP }
```

Алгоритм работы данного скрипта заключается в том, что вначале очищаются соответствующие списки nftables, затем с помощью регулярных выражений из

файлов извлекаются FQDN-имена серверов, которые с помощью утилиты dig преобразуются в IP-адреса, а затем добавляются к требуемым спискам.

8. Проблема в том, что приведенный выше скрипт нужно выполнять периодически. Для ее решения преобразуем скрипт в systemd юнит /etc/systemd/system/nft-dns.service:

```
# /etc/systemd/system/nft-dns.service
[Unit]
Description=nftables DNS resolve service
Requires=nftables.service
Wants=nft-dns.timer
After=network.target

[Service]
Type=oneshot
ExecStart=bash -c "nft flush set ip firewall allowed-ntp-servers ;
nft flush set ip firewall allowed-repos ; grep -oP '(?<=^deb
http://)[^ /]*' /etc/apt/sources.list | uniq  | xargs dig +short |
xargs -r -I IP nft add element ip firewall allowed-repos { IP } ;
grep -oP '(?<=^NTP=).+$' /etc/systemd/timesyncd.conf | xargs dig
+short | xargs -r -I IP nft add element ip firewall allowed-ntp-
servers { IP } ; grep -oP '(?<=^FallbackNTP=).+$'
/etc/systemd/timesyncd.conf | xargs dig +short | xargs -r -I IP nft
add element ip firewall allowed-ntp-servers { IP } "

StandardOutput=journal

[Install]
WantedBy=multi-user.target
```

9. Затем для ежечасного запуска этого юнита создадим systemd таймер /etc/systemd/system/nft-dns.timer:

```
# /etc/systemd/system/nft-dns.timer
[Unit]
Description=nftables DNS resolve timer
Requires=nftables.service

[Timer]
Unit=nft-dns.service
OnCalendar=hourly

[Install]
WantedBy=timers.target
```

10. После создания юнита и таймера активируем их, выполнив заклинания:

```
systemctl daemon-reload
systemctl enable nft-dns.service
systemctl enable nft-dns.timer
```

11. Последним этапом данной задачи будет настройка fail2ban на анализ журналов работы OpenSSH сервера и блокирование всех тех, кто совершил большое

количество неудачных попыток авторизации по ssh.

На самом деле fail2ban по умолчанию защищает SSH-сервер сразу после установки. Единственное, что необходимо сделать, так это поменять действия, выполняемые при блокировке. По умолчанию они рассчитаны на iptables (предшественника nftables), нам же требуется их поменять для nftables.

Сделать это очень просто. В конфигурационном файле /etc/fail2ban/jail.conf в секции [DEFAULT] значение
параметров *banaction* и *banaction_allports* необходимо поменять на *nftables-multiport* и *nftables-allports* соответственно. Затем перезапустить службу заклинанием

```
service fail2ban restart
```

Для блокировки нарушителей fail2ban добавляет к правилам фильтрации nftables новую таблицу f2b-table. Эта таблица содержит единственную цепочку f2b-chain, имеющую более низкий приоритет и соответственно срабатывающую раньше, чем тем цепочки, что мы создавали в файле /etc/nftables.com. Единственным правило цепочки f2b-chain является блокировка доступа к порту ssh (tcp 22) для IP-адресов, включенных в список addr-set-sshd. Пример рассмотренных списков и правил фильтрации, при добавлении туда нарушителя, выглядит следующим образом (Рисунок 3):

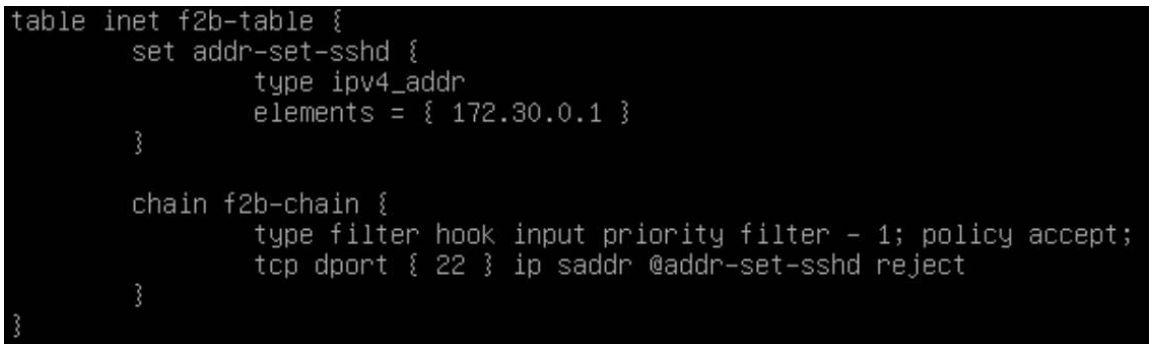


Рисунок 3

Текущее состояние блокировок можно посмотреть, выполнив заклинание:

```
fail2ban-client status sshd
```

Типовая задача: проверка работы брандмауэра

Очень часто на практике возникает задача проверить, работает ли межсетевой экран. Сделать это можно тривиальным образом: попробовать «пингануть» защищаемый узел или попробовать обратится к какому-либо защищаемому сетевому сервису.

Иногда специалисты по информационной безопасности могут потребовать от системных администраторов провести полную гарантированную проверку (аттестацию) всех правил

фильтрации. Например, проверить на возможность открытия UDP и TCP порты из всего возможного диапазона (1-65535).

Проблема в том, что на проверяемом сервер обычно используется не более десятка сетевых сервисов и, соответственно, открытых портов, иногда больше, но не суть. Возникает вопрос, как проверить порты, которые никто не слушает?

Не самым удобным, но очень доступным вариантом решения этой задачи будет следующий: с помощью утилиты netcat на проверяемом сервере при включенном брандмауэре открываются UDP или TCP порты, а затем сервер с помощью утилиты nmap сканируется с другой машины. Если обнаруживаются порты, которые не должны быть открытыми, то с межсетевым экраном есть проблемы.

Практическая работа: проверка работы брандмауэра

Описание стенда

В этой практической работе мы будем использовать лабораторный стенд от предыдущей работы.

Задача

Проверить на виртуальной машине «Сервер» работу локального брандмауэра в части блокировки входящего трафика в отношении всего диапазона TCP и UDP портов.

Подготовка стенда

1. На виртуальную машину «Сервер» поставим две дополнительные утилиты: «netcat» и «netstat». Для этого выполним заклинание:

```
apt install netcat net-tools
```

2. На гипервизор, а именно с него мы и будем сканировать, установим nmap.

Ход выполнения работы

1. Перечень открытых портов на сервере можно узнать с помощью утилиты netstat. Для UDP портов заклинание будет таким:

```
netstat -lu
```

а для TCP портов таким:

```
netstat -lt
```

2. Удаленную проверку доступности открытых портов будем проводить с помощью nmap. Например, для проверки порта UDP 100:

```
nmap.exe -v 172.30.0.40 -T5 -sU -p100
```

или для проверки порта TCP 100:

```
nmap.exe -v 172.30.0.40 -T5 -sT -p100
```

3. Для того чтобы проверить порты, которые никто не слушает, воспользуемся утилитой netcat и откроем их. В частности, для открытия порта UDP 100 выполним следующее заклинание:

```
echo "PORT UDP 100 opened" |  nc -lu 100
```

Здесь netcat ожидает подключение по порту UDP 100, а после подключения выдает в сеть строку «PORT UDP 100 opened» и завершает свою работу, закрывая порт. Если нужно открыть TCP порт 300, то заклинание будет чуть другим:

```
echo "PORT TCP 300 opened" |  nc -lt 300
```

4. Для удобства открытия диапазона портов воспользуемся shell-скриптом openport.sh:

```
#!/bin/bash
# openport.sh

PROTOCOL=$1
START_PORT=$2
END_PORT=$3

if [[ $# != 3 ]]
then
    echo -e "Script usage:\nopenport.sh <UDP| TCP> <start port> <end
port>"
    exit 1
fi
x=$START_PORT
while [ $x -le $END_PORT ]
do
if [ $PROTOCOL == "UDP" ]
then
    echo "PORT UDP $x opened" |  nc -lu $x &
elif [ $PROTOCOL == "TCP" ]
then
    echo "PORT UDP $x opened" |  nc -lt $x &
```

```
else
    echo "ERROR: invalid protocol"
    exit 1
fi
echo "Open port $PROTOCOL: $x"
x=$(( $x + 1 ))
done
```

Примечание. Во время работы скрипт запускает в фоне множество процессов netcat, которые автоматически завершаются после обращения к ним по сети. Рекомендуется за один раз открывать не больше 1000 портов, иначе сервер может потерять стабильность.

5. Пример 1. Проведем проверку TCP портов 1-30.
   На сервере выполняем заклинание:

```
./openport.sh TCP 1 30
```

   На сканирующей машине запускаем nmap, выполнив заклинание:

```
nmap.exe -v 172.30.0.40 -T5 -sT -p1-30
```

6. Пример 2. Проведем проверку TCP портов 20001-20900.
   На сервере выполняем заклинание:

```
./openport.sh UDP 20001 20900
```

   На сканирующей машине запускаем nmap, выполнив заклинание:

```
nmap.exe -v 172.30.0.40 -T5 -sU -p20001-20900
```

7. Для решения поставленной задачи необходимо последовательно открывать и сканировать все порты из диапазона 1-65535.

Типовой сценарий: сегментирование локальной сети маршрутизатором с функцией брандмауэра

Хорошей практикой защиты локальных сетей является их сегментация – разделение плоской сети на подсети с последующим разграничением и контролем трафика между ними.
В корпоративных сетях, как правило, выделяют следующие сегменты: сегмент пользовательских рабочих станций, сегмент серверов, сегмент технологического оборудования (например, системы видеонаблюдения), сегмент серверов, имеющих доступ из Интернет (DMZ) и другие сегменты. Каждый сегмент может в свою очередь быть разделен на полсегмента и так далее.

Существует несколько способов сегментации, но наиболее распространенной является

сегментация на сетевом уровне (L3), когда каждому сегменту присваивается своя IP-подсеть, а разграничение доступа осуществляется маршрутизаторами с функцией брандмауэра.

При разграничении доступа между пользовательским с серверным сегментами сети специалисты по информационной безопасности, как правило, в качестве технического задания предъявляют системным администраторам свой любимый документ – матрицу доступа, в которой указывается, какой пользователь к каким серверам или каким сетевым сервисам может иметь доступ. На основании матрицы доступа строится схема разграничения трафика.

Важно отметить, что в корпоративных сетях доступы пользователей, как правило, бывают типовыми. Например, все сотрудники отдела продаж должны иметь доступ к Web-интерфейсу системы учета клиентов (customer relation management, CRM). С точки зрения учета и управления доступами можно сказать, что таким пользователям назначается роль «Доступ к Web-интерфейсу CRM». Поэтому очень важно сделать правила фильтрации таким образом, чтобы можно было просто и единообразно добавлять доступы пользователям. Требование единообразия важно еще и потому, что любую систему разграничения доступа нужно периодически проверять, а зоопарк вариантов предоставления доступов серьезно осложнит эту задачу.

Базовая схема разграничения трафика между пользовательским и серверным сегментами будет следующая:

- Из серверного сегмента в пользовательский запрещен весь трафик, кроме:
    - трафика, текущего по уже установленным соединениям.
- Из пользовательского сегмента в серверный запрещен весь трафик, кроме:
    - трафика, текущего по уже установленным соединениям;
    - трафика к явно разрешенным сетевым службам явно разрешенных серверов.

Примечание 1. Тут может возникнуть вопрос: почему по умолчанию запрещается весь трафик из серверного сегмента в пользовательский? Дело в том, что в модели «Клиент-Сервер» последний выполняет строго пассивную функцию. Он ждет обращения клиента, обслуживает его, после чего разрывает сетевое соединение. Сам сервер соединяться ни с кем не должен, поэтому ему и блокируется возможность самостоятельной установки соединений. Важно отметить, что в серверном сегменте очень часто размещают технологические рабочие станции, которые периодически опрашивают узлы сети (например, сетевые принтеры на предмет проверки уровня чернил). В таких случаях их либо выносят в отдельный сегмент, либо делают для них исключения в правилах фильтрации.

Примечание 2. Большинство серверов enterprise уровня имеют несколько сетевых адаптеров. Например, один рабочий, с помощью которого он обслуживает целевые запросы клиентов, а второй служебный, используемый, например, для систем удаленного управления типа iLO, IPMI, iDrac и др. Поэтому корректнее говорить не «помещение сервера в отдельный сегмент», а «помещение сетевого интерфейса сервера в отдельный сегмент». Нормальной является ситуация, когда все сетевые адаптеры сервера находятся в различных сетевых сегментах.

Практическая работа: разграничение трафика между пользовательским и серверным сегментами

Описание стенда

Дан макет корпоративной сети (Рисунок 4), в которой присутствуют две подсети: 192.168.0.0/24 – для пользовательского сегмента и 172.30.0.0/24 — для серверного сегмента. В макете эти две сети представлены виртуальными сетями: «Custom11» и «NAT» соответственно. Виртуальная машина FW снабжена двумя сетевыми интерфейсами, каждый из которых «смотрит» в свою виртуальную сеть. Данная машина выполняет функции маршрутизатора и брандмауэра.

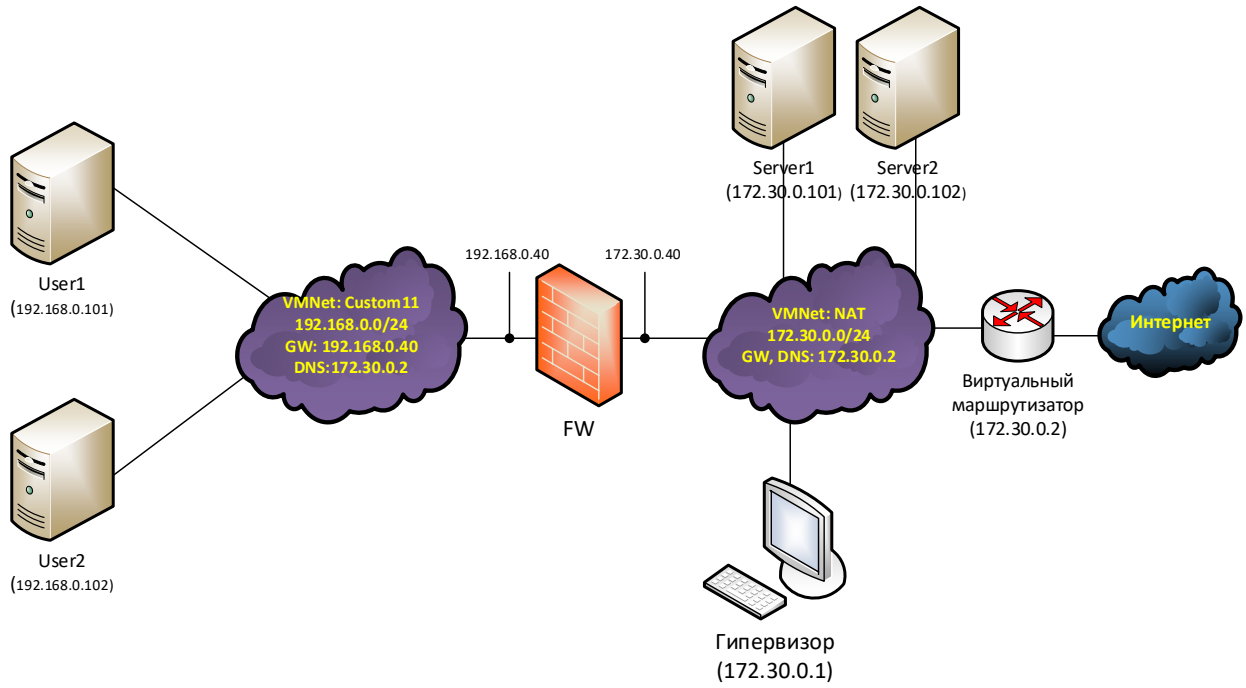


Рисунок 4

Задача

Настроить разграничение трафика между пользовательским и серверным сегментами в соответствии со следующей матрицей доступа:

	Server1	Server2	Виртуальный маршрутизатор
User1	TCP:80	TCP:80	UDP:53
User2	TCP:22	TCP:22	UDP:53

Подготовка стенда

1. Средствами системы виртуализации создадим отдельную виртуальную сеть (VMnet), которую назовем «Custom11».
2. Ранее созданный шаблон виртуальной машины растиражируем в 5 отдельных экземпляров: FW, User1, User2, Server1, Server2.
3. К виртуальной машине FW добавим второй виртуальный сетевой адаптер, который соединим с сетью «Custom11». Ее первый сетевой адаптер должен быть подключен к виртуальной сети «NAT».
4. Сетевые адаптеры машин User1 и User2 подключим к «Custom11», а сетевые адаптеры Server1, Server2 подключим к «NAT».
5. Редактируя файлы /etc/network/interfaces, назначим всем виртуальным машинам статические IP-адреса, а также шлюзы (GW) и DNS-сервера в соответствии со схемой лабораторного стенда. В качестве шлюзов по умолчанию для Host1 и Host2 будет 192.168.0.40, для Server1 и Server2 172.30.0.40, а для FW 172.30.0.2
6. На виртуальной машине FW активируем функции маршрутизации IPv4 трафика. Для этого в файле /etc/sysctl.conf раскомментируем строку net.ipv4.ip_forward=1. Сделать это можно с помощью заклинания:

```
sed '/net.ipv4.ip_forward=1/s/^#//' -i /etc/sysctl.conf
```

7. Проверим стенд. Host1 и Host2 должны иметь возможность «пингануть» Server1, Server2 и FW и наоборот. FW должен иметь возможность «пингануть» любой из узлов сети.

Ход выполнения работы

1. Полную настройку брандмауэра мы уже рассматривали в предыдущей задаче. Поэтому здесь мы направим внимание на конфигурационный файл /etc/nftables.conf. Причем мы также опустим вопросы защиты самого FW, поскольку они будут точно такими же, как и в предыдущей задаче, и сосредоточимся лишь на фильтрации транзитного трафика.
2. Анализируя матрицу доступа можно предположить, что User1 является рядовым работником, и ему требуется доступ по http ко всем серверам. Назовем такой доступ ролью «all_web». Для User2 требуется предоставить доступ ко всем серверам по SSH, вероятно он системный администратор. Обозначим такой доступ ролью «all_ssh». Обоим пользователям требуется доступ к DNS-серверу — это будет роль «DNS».
3. Наиболее простым способом создать в nftables ролевую систему разграничения доступа будет применение правил, где в качестве аргументов используются списки (sets), содержащие конкретные параметры предоставления доступа. В нашем случае для каждой роли нужно создать два списка: *наименование роли_users* и *наименование роли_servers*. В первый список будут добавляться IP-адреса пользователей, во второй — IP-адреса серверов, а также протоколы и доступные порты.

4. После создания списков для каждой роли создадим правило, разрешающее установку соединений *ip saddr @название роли_users ip daddr. meta l4proto. th dport @название роли_servers accept*. Трафик по инициированным соединения, как обычно, будет проходить с помощью правила *ct state established,related accept*.
5. Итоговый конфигурационный файл (/etc/nftables.conf), решающий поставленную задачу будет выглядеть следующим образом:

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

flush ruleset

table ip firewall {

        set all_web_users {
                type ipv4_addr
                flags interval
                elements = { 192.168.0.101 }
        }

        set all_web_servers {
                type ipv4_addr . inet_proto . inet_service
                flags interval
                elements = { 172.30.0.101-172.30.0.102 . tcp . 80 }
        }

        set all_ssh_users {
                type ipv4_addr
                flags interval
                elements = { 192.168.0.102 }
        }

        set all_ssh_servers {
                type ipv4_addr . inet_proto . inet_service
                flags interval
                elements = { 172.30.0.101-172.30.0.102 . tcp . 22 }
        }


        set DNS_users {
                type ipv4_addr
                flags interval
                elements = { 192.168.0.101, 192.168.0.102 }
        }

        set DNS_servers {
                type ipv4_addr . inet_proto . inet_service
                flags interval
                elements = { 172.30.0.2 . udp . 53 }
        }

        chain fw_forward {
                 type filter hook forward priority filter; policy
drop;
                 ct state established,related accept
```

```
ip saddr @all_web_users ip daddr . meta l4proto . th dport
@all_web_servers accept
ip saddr @all_ssh_users ip daddr . meta l4proto . th dport
@all_ssh_servers accept
ip saddr @DNS_users ip daddr . meta l4proto . th dport @DNS_servers
accept
        }
}
```

Примечание. После настройки всех правил вы столкнетесь с одной проблемой: DNS на User1 и User2 работать не будет. Это не баг, это методическая фича. Проблема в том, что DNS-сервер работает на виртуальном маршрутизаторе, который ничего не знает о пользовательской сети 192.168.0.0/24, из-за этого ответы на запросы не доходят до адресатов. Этой проблемой хотелось показать реальные сложности, возникающие при сегментации уже работающих сетей с помощью разделения их на IP-подсети. На предприятиях со сложившейся инфраструктурой, но низким уровнем зрелости IT внедрить подобные меры защиты крайне сложно, а учитывая user resistance, практически невозможно. Но проблема все же имеет решение, и мы поговорим о нем прямо сейчас.

Типовой сценарий: сегментирование локальной сети коммутатором с функцией брандмауэра

Данный подход к межсетевому экранированию имеет множество названий: коммутатор с функцией брандмауэра, «stealth firewall», «transparent firewall» и др. Суть, однако, заключается в том, что сегментируемая сеть сохраняет свою IP-адресацию, а разделение происходит путем установки коммутатора в разрыв между будущими сегментами. Причем коммутатор фильтрует трафик как обычный межсетевой экран — на основании данных со всех инкапсулированных протоколов (L2, L3, L4, L7), а не только данных с протоколов канального уровня (L2), как в случае с обычным коммутатором.

Сетевые интерфейсы коммутатора не имеют IP-адресов (за исключением тех, что используются для управления). Соответственно, данное устройство не видимо для других сетевых узлов (за исключением случаев, когда используются специфические протоколы, например OSPF). Поэтому коммутатор с функцией брандмауэра часто называют прозрачный межсетевой экран — stealth firewall.

Применение фильтрующих коммутаторов имеет несколько неоспоримых преимуществ:

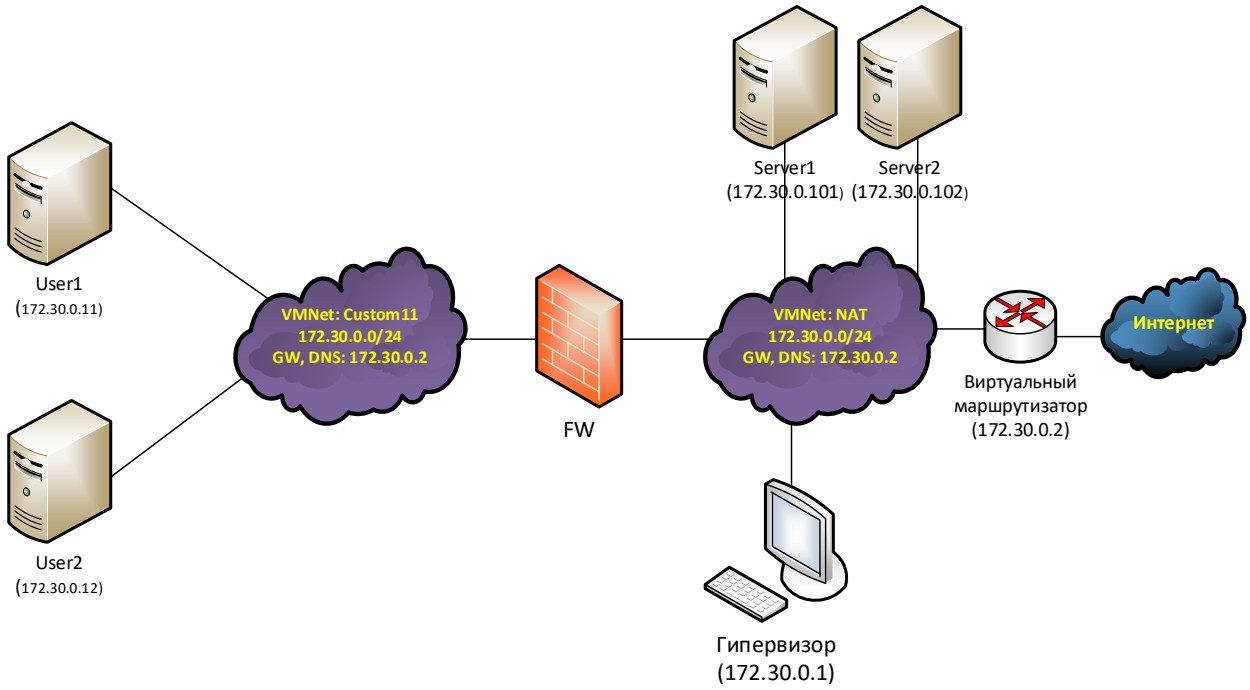
1. Внедрение устройства не требует выделения IP-подсетей и, как следствие, перенастройки таблиц маршрутизации роутеров и сетевых стеков узлов.
2. Устройство очень просто внедрить и очень просто изъять из сети в случае его поломки.

Основной недостаток данной технологии, как ни странно, является прямым следствием ее основного достоинства: сегментирование сети без разделения ее на IP-подсети не позволяет ограничивать широковещательный трафик, что негативным образом сказывается на производительности сетей и усиливает негативные последствия от атак типа широковещательный шторм (broadcast flood), отравления кэша ARP (ARP-poisoning), подмены DHCP (Rogue DHCP Server) и других.

Практическая работа: провести сегментирование сети без разделения ее на IP-подсети

Описание стенда

Используемый лабораторный стенд (Рисунок 5) очень похож на предыдущий, за исключением того, что все машины здесь находятся в одной IP-подсети. Для того, чтобы машины UserX и ServerX не могли связаться между собой напрямую, а использовали для связи FW, их разделили по виртуальным сетям «Custom11» и «NAT».



(Рисунок 5)

Задача

Настроить разграничение трафика в соответствии с матрицей доступа из предыдущей задачи.

Подготовка стенда

1. Внося изменения в файлы /etc/network/interfaces, назначим статические адреса всем узлам сети.
2. Для организации работы FW в режиме коммутатора установим на него пакет bridge-utils:

```
apt install bridge-utils
```

3. Затем на узле FW сконфигурируем коммутатор, объединив в мост (bridge) все сетевые порты. Для этого файл /etc/network/interfaces заполним следующим образом:

```
auto lo
iface lo inet loopback

iface ens33 inet static

iface ens36 inet static

auto br0
iface br0 inet manual
bridge_ports ens33 ens36
```

4. Проверим стенд. Все узлы, кроме FW, должны «пинговаться» между собой. DNS, кстати, тоже должен работать.

Ход выполнения работы

1. Как и в предыдущей задаче рассмотрим только файл с правилами межсетевого экранирования /etc/nftables.conf. Скорее всего при беглом осмотре вы даже не заметите в нем различий по сравнению с таким же файлом из предыдущей задачи.

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

flush ruleset

table bridge firewall {

        set all_web_users {
                type ipv4_addr
                flags interval
                elements = { 172.30.0.11 }
        }

        set all_web_servers {
                type ipv4_addr . inet_proto . inet_service
                flags interval
                elements = { 172.30.0.101-172.30.0.102 . tcp . 80 }
        }

        set all_ssh_users {
                type ipv4_addr
                flags interval
                elements = { 172.30.0.12 }
        }
```

```
        set all_ssh_servers {
                type ipv4_addr . inet_proto . inet_service
                flags interval
                elements = { 172.30.0.101-172.30.0.102 . tcp . 22 }
        }

        set DNS_users {
                type ipv4_addr
                flags interval
                elements = { 172.30.0.11, 172.30.0.12 }
        }

        set DNS_servers {
                type ipv4_addr . inet_proto . inet_service
                flags interval
                elements = { 172.30.0.2 . udp . 53 }
        }

        chain fw_forward {
                 type filter hook forward priority filter; policy
drop

                 ether type arp accept
                 ct state established,related accept

ip saddr @all_web_users ip daddr . meta l4proto . th dport
@all_web_servers accept

ip saddr @all_ssh_users ip daddr . meta l4proto . th dport
@all_ssh_servers accept

ip saddr @DNS_users ip daddr . meta l4proto . th dport @DNS_servers
accept

        }
}
```

Основные отличия в том, что тип таблицы firewall изменился с *ip* на *bridge*, поменялись адреса в списках **_users* (так как изменились соответствующие адреса машин), и к правилам фильтрации мы добавили разрешение прохождения ARP трафика.

Проект OneButtonFirewall

Можно ли сделать межсетевой экран, не требующий конфигурирования? Если можно, то что он будет уметь? Давайте разбираться.

Мы с вами рассмотрели несколько схем разграничения трафика. Есть ли среди них та, что не требует конфигурирования, то есть указания IP-адресов или портов, на основании которых производится фильтрация трафика? Конечно, есть, и эта схема первая в списке.

Теперь возникает второй вопрос: рассмотренная схема относится к защите рабочих станций, как с ее помощью построить межсетевой экран для защиты сегмента сети? Тут тоже нет ничего сложного. Один сегмент сети, подключенный к брандмауэру, будем считать внутренним (защищаемым), а второй внешним (от которого защищаем). Тогда правила фильтрации преобразуются в следующие:

- Запрещен транзит трафика из внешнего сегмента во внутренний, кроме того, что относится к уже установленным соединениям.
- Разрешен транзит любого трафика из внутреннего сегмента во внешний сегмент.

Теперь надо решить, как отличить внутренний сегмент от внешнего, и как сделать, чтобы при внедрении межсетевого экрана не требовалось переконфигурировать узлы сети. Отличать сегменты можно по сетевым интерфейсам межсетевого экрана, к которым они подключены, а ответом на второй вопрос будет использование технологии фильтрации с помощью коммутатора.

В итоге получаем своеобразный IP-диод, который в зависимости от подключения сегментов сети к своим интерфейсам пропускает соединения только в одну сторону. Стоит сегменты переподключить к другим портам, и направление сетевых соединений изменится на противоположное.

Осталась одна проблема – широковещательный трафик. Раньше, когда мы рассматривали фильтрацию с помощью коммутатора, мы не обращали на него внимание, и по факту он был запрещен, кроме разве что ARP. Это, конечно, безопасно, но для универсального межсетевого экрана не подходит, поскольку ломает работу множества протоколов, базирующихся на широковещательных посылках, и первыми такими протоколами будут протоколы ОС Windows, отвечающие за отрисовку сетевого окружения. В результате тетенька бухгалтерша, сидящая за подобным брандмауэром, издаст злобный писк, что пропали все ее сетевые папки, и что вы вообще бесполезный вредитель. Нам такого не надо, поэтому, скрипя сердцем, разрешим транзит всего широковещательного трафика.

Ну, вроде бы все учли… ан нет. В ходе эксплуатации OneButtonFirewall всплыла проблема, откуда не ждали – получение IP-адреса от DHCP-сервера, находящегося во внешнем сегменте. ОС Windows получает IP-адреса по DHCP сугубо на широковещательных рассылках, и текущих правил фильтрации ей полностью хватает. Linux же идет своим путем. При получении адреса на конечном этапе DHCP-сервер посылает клиенту адресный (unicast) пакет по UDP 68, и, чтобы тот достиг адресата, в правилах фильтрации нужно сделать соответствующее исключение.

Практическая работа: защитить сегмент сети с помощью межсетевого экрана, построенного по технологии OneButtonFirewall

Описание стенда

Давайте теперь соберем лабораторный стенд (Рисунок 6) и отработаем на нем наши идеи. Тут мы даже немного усложним задачу. У FW будет не два интерфейса: один внутренний и один внешний, а четыре: один внешний и три внутренних. Все узлы, подключенные к внутренним интерфейсам, будем считать внутренним сегментом и фильтровать трафик

между этими узлами не будем. Как вы увидите дальше, количество внутренних интерфейсов не имеет значения, но внешний может быть только один.

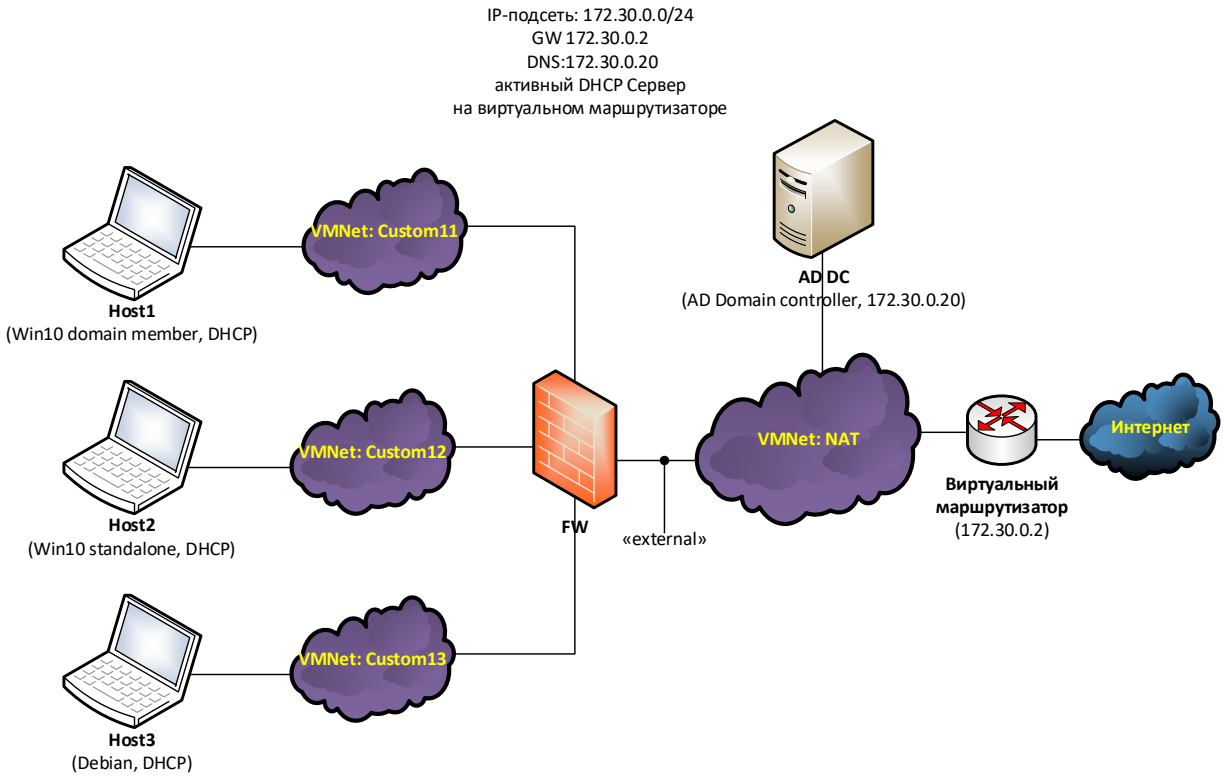


Рисунок 6

В этом примере для наглядности мы моделируем классическую корпоративную сеть на базе ОС Windows. Host1 – это Windows 10, подключенный к домену Active Directory, контроллер которого (ADDC) расположен во внешней сети. Host2 – Windows 10, не подключенная к домену, а Host3 – наша шаблонная машина на базе Debian. Все HostX получают IP-адреса по DHCP от виртуального маршрутизатора.

Для моделирования наших идей с помощью средств виртуализации каждый узел внутренней сети подключим через отдельную виртуальную сеть к FW. Как и прошлый раз мы делаем это для того, чтобы весь трафик между узлами HostX и узлами внешней сети шел через FW.

Задача

Защитить узлы HosX от несанкционированных подключений.

Подготовка стенда

1. На узле FW установим 4 сетевых интерфейса. Для наглядности с помощью средств виртуализации изменим на них MAC-адреса (Рисунок 7):

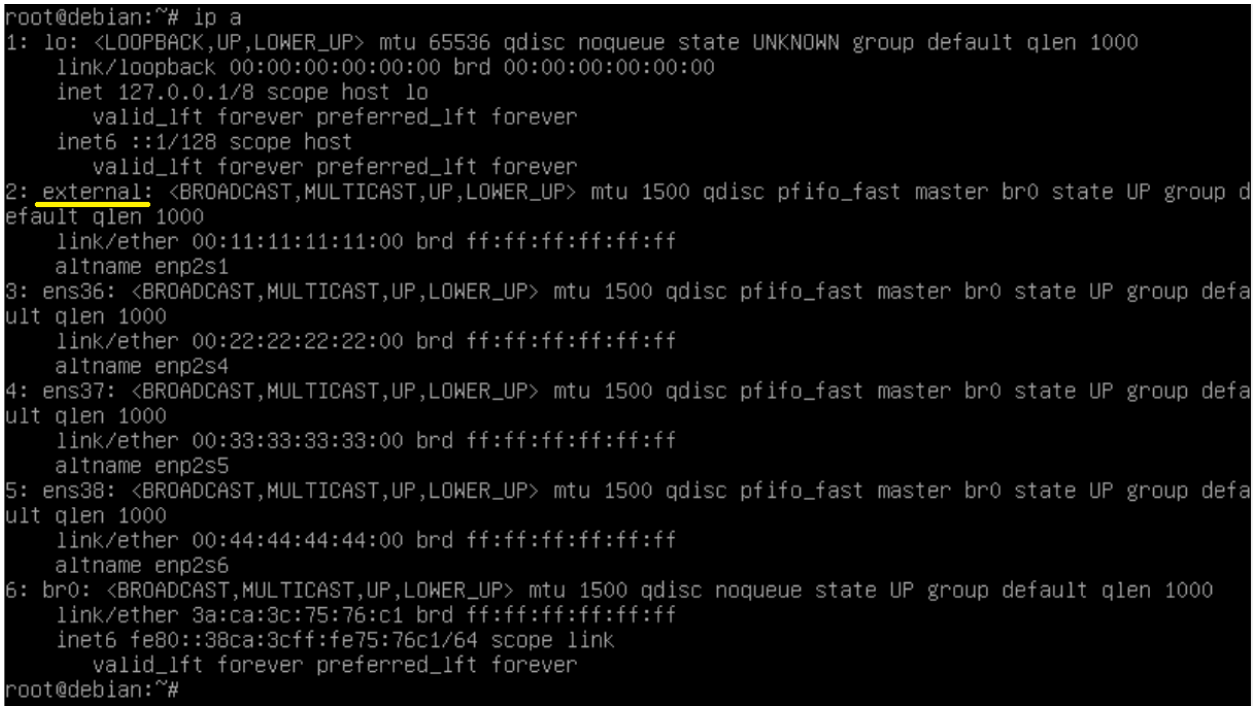


Рисунок 7

2. Один из интерфейсов с MAC-адресом 00:11:11:11:11:00, подключенный к «NAT», переименуем в «external» (Рисунок 7).

   Для этого создадим файл /etc/systemd/network/10-set-external-name.link и заполним его следующим содержанием:

```
# /etc/systemd/network/10-set-internal1-name.link
[Match]
MACAddress=00:11:11:11:11:00

[Link]
Name=external
```

3. Как и при решении прошлой задачи установим на узел FW пакет bridge-utils.
4. Сделаем из FW коммутатор. Для этого в файл /etc/network/interfaces запишем следующее содержание:

```
auto lo
iface lo inet loopback

iface external  inet manual

iface ens36 inet manual

iface ens37 inet manual

iface ens38 inet manual
```

```
auto br0
iface br0 inet manual
bridge_ports external ens36 ens37 ens38
```

5. Проверим работоспособность сети, «пингуя» все узлы. Для проведения тестирования не забудьте отключить встроенный межсетевой экран на Windows машинах. Все узлы должны «пинговаться».

Ход выполнения работы

1. Для решения поставленной задачи заполним файл /etc/nftables.conf следующим образом:

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

flush ruleset

table bridge firewall {

      chain fw_forward {
            type filter hook forward priority filter; policy drop;

# Разрешаем весь трафик между внутренними портами и трафик
# от внутренних портов к внешнему
            iifname != "external" accept

# Разрешаем IPv4 трафик по уже установленным соединениям
            ether type ip ct state established,related accept

# Разрешаем весь IPv4 трафик с широковещательными MAC адресами
            ether type ip ether daddr ff:ff:ff:ff:ff:ff accept

# Разрешаем трафик из внешней сети во внутреннюю по UDP 68
# Костыль для того, чтобы внутренние узлы могли получить адрес
# по DHCP из внешней сети
            udp dport 68 accept
      }
}
```

Реализация OneButtonFirewall в виде аппаратного устройства

Идея OneButtonFirewall наиболее полным образом раскрывает себя в виде аппаратного устройства. Реализовать ее в виде виртуальных машин или контейнеров, конечно, тоже можно, но особого профита это не даст. Аппаратное же устройство дает эффект своеобразного кирпичика, который можно подключить к сети, и он ее защищает. Отсюда, кстати, и название пошло OneButtonFirewall, так как подобному устройству нужна только одна кнопка: включение питания. Несомненным плюсом устройства является его неконфигурируемость. Оно реализует жесткий алгоритм, не требующий дополнительных

настроек. Как следствие, не нужно заботиться о целостности конфигурации или продумывать интерфейсы администрирования и обеспечивать их защиту.

Построить OneButtonFirewall очень просто. Достаточно найти подходящую аппаратную платформу установить туда дистрибутив Linux, в котором есть поддержка nftables, и провести настройку из примера выше. Например, OneButtonFirewall, построенный на базе аппаратной платформы MikroTik, выглядит следующим образом (Рисунок 8):

Рисунок 8

Преимущества и недостатки OneButtonFirewall

Преимуществами аппаратных устройств,, реализующих технологию OneButtonFirewall, по сравнению с другими межсетевыми экранами являются:

1. Простота и скорость развертывания и изъятия системы защиты.
2. Защищенность конфигурации от изменения.
3. Минимальные требования к квалификации персонала.

Недостатки технологии:

1. Устройство реализует жесткий алгоритм фильтрации, который решает далеко не все задачи межсетевого экранирования.
2. Устройство не фильтрует широковещательный трафик и трафик по порту UDP 68 из внешней сети во внутреннюю.

Сценарии применения

Рассмотренные преимущества и недостатки определяют основную нишу применения OneButtoFirewall сценариями, в которых нужно просто и быстро реализовать межсетевое экранирование, а применение классических межсетевых экранов натыкается на

недостаточную квалификацию / саботаж / лень / перегруженность работников IT.

Рассмотрим теперь типовые сценарии применения OneButtonFirewall:

1. Защита сети отдела компании, например, бухгалтерии или службы безопасности. Узлы защищаемого отдела подключаются к внутренним портам межсетевого экрана, а остальная сеть к внешним. Внутренние узлы работают «как обычно», а из внешней сети подключиться к ним нельзя.
2. Защита сети компании от компьютеров подрядчиков (например, аудиторов). Тут обратная схема. Узлы компьютеров подрядчиков через коммутатор подключаются к внешнему порту межсетевого экрана, корпоративная сеть подключается к внутреннему порту. Из корпоративной сети можно подключится к компьютерам подрядчиков, а вот в обратную сторону нет.
3. Элемент дежурной аптечки системных администраторов. OneButtonFirewall наряду с антивирусным LiveUSB может существенно помочь в случае массового заражения сети компьютерными вирусами. С его помощью можно безопасно переустановить и обновить операционную систему компьютеров, в случаях когда заражение происходит еще до того, как стартует штатный межсетевой экран операционной системы.

Заключение

Несмотря на довольно внушительный объем статьи, мы с вами успели рассмотреть лишь базовые приемы межсетевого экранирования, причем множество мер можно серьезно улучшить. Но так и должно быть — нет предела совершенству, но первый шаг к нему мы только что сделали.