Hairpin NAT

Сделав проброс портов для WEB сервера в локальной сети, все отлично работает но ровно до тех пор, пока мы не попытаемся получить доступ по внешнему адресу из внутренней сети. Вообще, таких ситуаций следует избегать, предпочтительно использовать для доступа доменные имена и двойной горизонт DNS, когда для внешних пользователей одно и тоже имя разрешается во внешний адрес, а для внутренних – во внутренний. Но это возможно не всегда. Попробовав же обратиться изнутри по внешнему адресу, мы получим ошибку соединения.

ПК отправляет запрос на внешний адрес маршрутизатора, он заменяет адрес назначения адресом внутреннего сервера и отправляет пакет к нему, адрес отправителя остается неизменным. Веб-сервер видит, что отправитель находится с ним в одной сети и отправляет ответ ему напрямую. Но отправитель направлял запрос на внешний адрес сервера и ждет ответа именно от него, поэтому ответный пакет, отправителем которого указан внутренний адрес веб-сервера будет отброшен и связь установить не удастся.

Очевидно, что нужно каким-то образом заставить веб-сервер отвечать не напрямую клиенту, а пересылать пакет обратно маршрутизатору. Здесь нам на помощь придет действие src-nat (SNAT), которое позволяет изменить адрес отправителя, находящееся в цепочке POSTROUTING. В терминах Mikrotik данная настройка носит наименование **Hairpin NAT**.

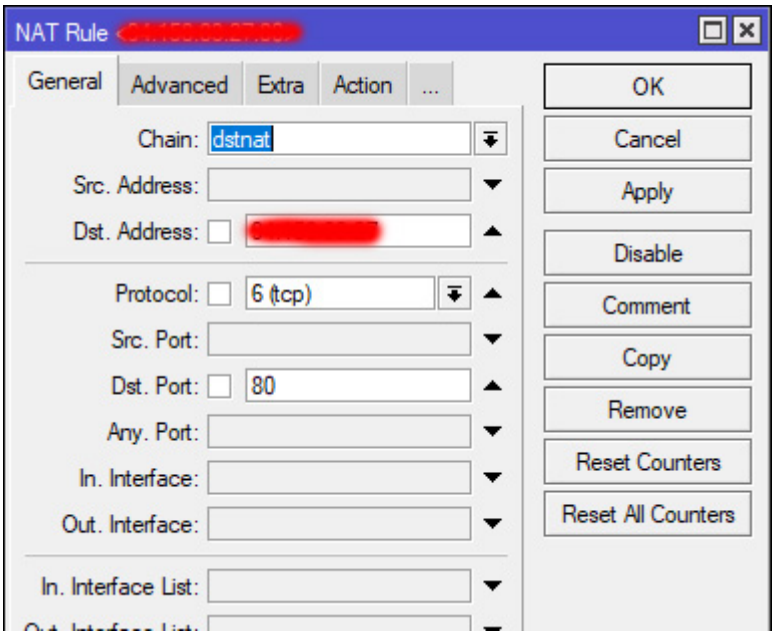
При такой схеме наш маршрутизатор не только изменяет адрес назначения, но и адрес источника, указывая в его качестве собственный внутренний IP. Обработав такой пакет веб-сервер отправит его обратно к маршрутизатору, который выполнит обратные преобразования (согласно таблице трансляции) и оправит его получателю.

Путь пакета от клиента к веб-серверу и обратно порядок

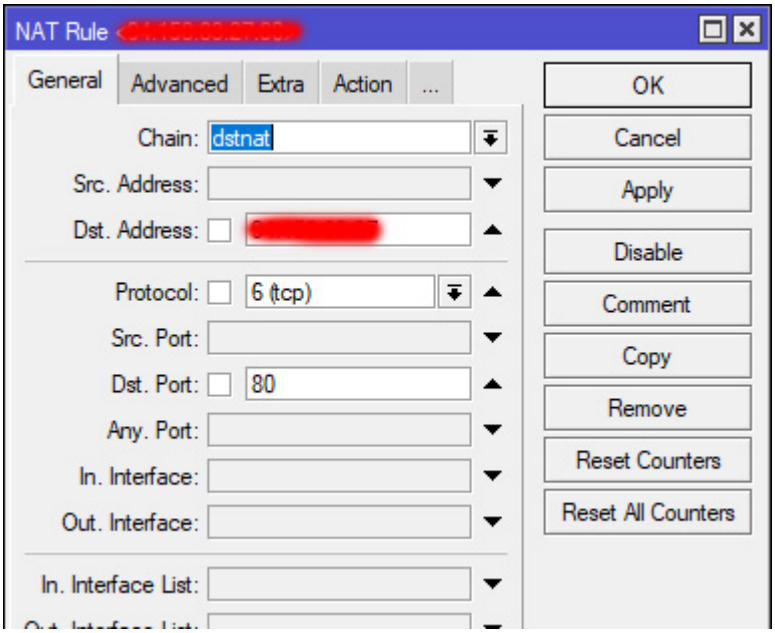
прохождения цепочек будет следующим:

PREROUTING -> FORWARD -> POSTROUTING

IP – Firewall – NAT и добавим следующее правило: **Chain – dstnat** (читай PREROUTING), **Dst. Address – 5.5.5.5** – внешний IP-адрес нашего роутера, **Protocol – tcp** – используемый протокол, если сервис может использовать несколько протоколов, скажем TCP и UDP – потребуется создать отдельное правило для каждого протокола, **Dst. Port – 80** – порт, на котором маршрутизатор будет принимать внешние соединения



На закладке **Action** укажите: **Action – dst-nat** – выполняем замену адреса получателя, **To Addresses – 192.168.1.41** – внутренний адрес веб-сервера, **To Ports – 80** – порт, на котором веб-сервер принимает соединения. Если внешний и внутренний порт совпадают, то последний можно не указывать.

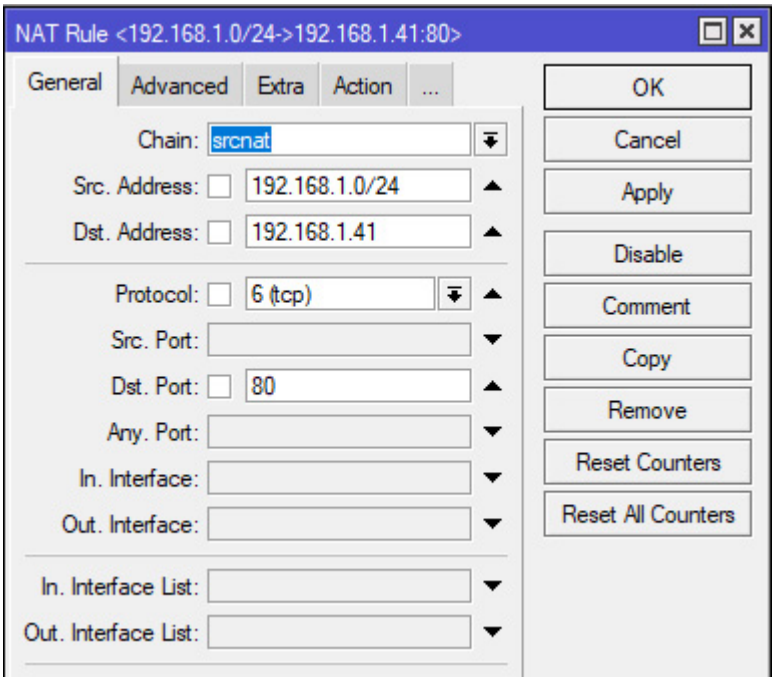


В терминале быстрее:

```
/ip firewall nat
add action=dst-nat chain=dstnat dst-address=192.168.1.41 dst-port=80 protocol=tcp to-addresses=5.5.5.5 to-ports=80
```

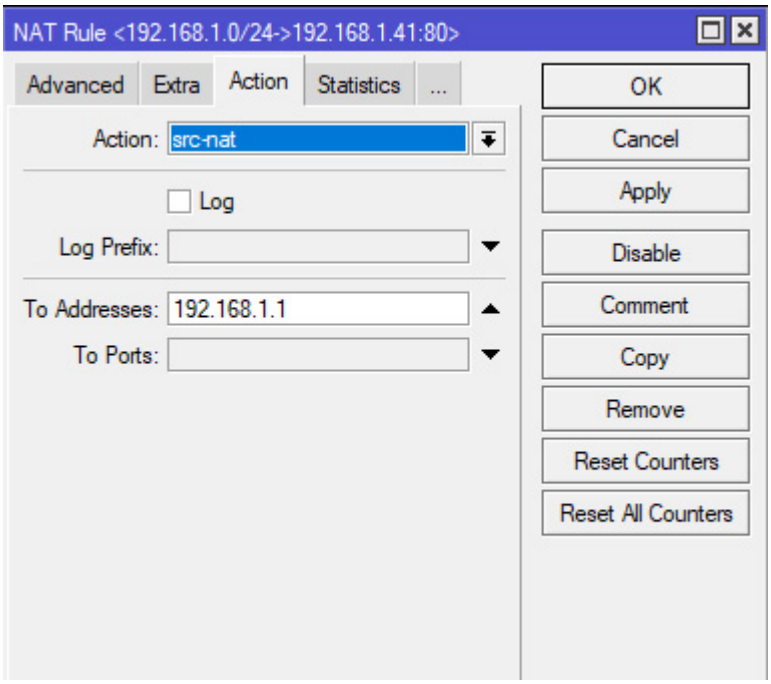
Теперь переходим к настройке Hairpin NAT

IP – Firewall – NAT и добавим: **Chain – src-nat** (POSTROUTING), **Src. Address – 192.168.1.0/24** – диапазон вашей локальной сети, **Dst. Address – 192.168.1.41** – внутренний адрес веб-сервера, **Protocol – tcp, Dst. Port – 80** – протокол и порт



Обратите внимание, что в качестве адреса и порта назначения мы указываем **внутренние адрес и порт**, так как пакет уже прошел цепочку PREROUTING, где данные получателя были изменены. К сожалению, не все это понимают, во многих инструкциях в сети в данном правиле фигурирует внешний адрес, стоит ли говорить, что такое правило работать не будет.

Затем переходим на закладку **Action** и указываем: **Action – src-nat** (SNAT), **To Addresses – 192.168.1.1** – указываем внутренний адрес нашего маршрутизатора



Терминал:

```
/ip firewall nat
add action=src-nat chain=srcnat dst-address=192.168.1.41 dst-port=80 protocol=tcp src-address=192.168.1.0/24 to-addresses=192.168.1.1
```

Такие же правила нужно создать и для 443.

Иногда для данного правила используется действие masquerade, в чем же разница? В первом приближении оба действия делают одно и тоже – изменяют адрес источника пакета, но **src-nat** работает только со *статическими адресами*, зато позволяет указать в качестве источника любой адрес, **masquerade** работает с *динамическими адресами*, но в качестве адреса источника может использовать только адрес того интерфейса, с которого уходит пакет. За счет того, что masquerade при каждом запросе определяет адрес интерфейса – это действие требует больше вычислительных ресурсов, нежели src-nat.

В нашем случае все адреса статические, поэтому использование src-nat здесь будет более уместно