

Базовая настройка брандмауэра в Mikrotik

 interface31.ru/tech_it/2022/12/bazovaya-nastroyka-brandmauera-v-mikrotik.html

Записки IT специалиста

Технический блог специалистов ООО "Интерфейс"

- [Главная](#)
- Базовая настройка брандмауэра в Mikrotik

Межсетевой экран, он же брандмауэр или файрвол - важнейшая служба вашего роутера, отвечающая как за его безопасность, так и за безопасность всей сети. Мы до сих пор не рассматривали базовую настройку брандмауэра в отрыве от общей настройки роутера, но как показала практика, многие администраторы откровенно плавают в этом вопросе и имеют некорректно или не оптимально настроенные системы. Как минимум это выливается в повышенную нагрузку на оборудование, а в самом плохом варианте ставит под угрозу безопасность всей сети. Поэтому давайте отдельно разберемся в этом вопросе.



Онлайн-курс по MikroTik

Научиться настраивать MikroTik с нуля или систематизировать уже имеющиеся знания можно на [углубленном курсе по администрированию MikroTik](#). Автор курса, сертифицированный тренер MikroTik Дмитрий Скоромнов, лично проверяет лабораторные работы и контролирует прогресс каждого своего студента. В три раза больше информации, чем в вендорской программе MTCNA, более 20 часов практики и доступ навсегда.

Общие принципы построения брандмауэра

Существует два основных типа настройки брандмауэра: **нормально открытый** брандмауэр и **нормально закрытый**. В первом случае разрешено все, что явно не запрещено, во втором случае - запрещено все, что явно не разрешено.

Возможно, для кого-то будет новостью, но оба этих подхода обычно гармонично сочетаются в рамках одного устройства. Нормально открытый брандмауэр применяется для внутренних сетей и исходящих соединений, а нормально закрытый - для все остальных соединений из внешних сетей. Это позволяет достичь простоты и удобства для собственных пользователей и в тоже время надежно защититься от внешнего мира.

Любое новое соединение первоначально находится в статусе **нового (new)** и мы должны принять решение: одобрить его или запретить. Для этой задачи и предназначены создаваемые нами правила, в нормально закрытом брандмауэре

мы описываем, какие соединения и по каким критериям следует разрешить. Если же пакет не соответствует ни одному правилу, то мы запрещаем такое соединение.

После того, как соединение было установлено - оно переходит в состояние **установлено (established)**, при этом специальный механизм ядра **Connection Tracker** отслеживает все проходящие пакеты и сопоставляет их соединениям, поэтому мы можем спокойно оперировать именно состоянием соединения, не задумываясь над техническими тонкостями работы.

Нужно ли каждый раз отслеживать пакеты уже установленного соединения вновь и вновь гоняя их по цепочке правил? Конечно же нет, если мы одобрили соединение, то в последующем можем и должны одобрять все его пакеты автоматически, это снижает нагрузку на устройство и позволяет использовать различные приемы, вроде Port Knocking или противодействия перебору паролей.

Очень часто одни соединения, будучи установленными, создают другие, которые называются **связанными (related)**, как пример можно привести хорошо известные FTP или RPTP, которые имеют управляющее соединение и соединение(я) для передачи данных. Они могут использовать динамические порты и поэтому их также следует автоматически разрешать, если мы разрешили основное соединение.

Поэтому в любом корректно настроенном брандмауэре самым первым правилом должно находиться разрешающее **установленные и связанные** соединения, а только после него мы уже должны анализировать новые подключения.

В роутерах Mikrotik существует еще один тип соединения - **неотслеживаемое (untracked)**, такие соединения не отслеживаются **Connection Tracker**, что позволяет существенно снизить нагрузку на устройство. Для чего это может быть нужно? Допустим, в фильтруемом вами трафике есть значительная доля принадлежащая доверенным узлам, для которых можно четко прописать критерии, в этом случае можно пометить их неотслеживаемыми и добавить их одобрение в самое первое правило, к уже установленным и связанным.

Таким образом, одним единственным правилом мы будем сразу пропускать подавляющую долю трафика, а к затратным операциям анализа будут доходить только новые пакеты.

Но тут самое время вспомнить еще про одно состояние соединения - **недействительное (invalid)**, это пакеты не являющиеся новыми, но не принадлежащие никакому установленному соединению, такие пакеты могут использоваться для атак и поэтому их следует блокировать в первую очередь, сразу после того, как мы одобрили установленные и связанные соединения и до того, как приступили к анализу новых пакетов.

Ну и завершающим правилом в цепочке должно стоять **запрещающее**, отсекающее все соединения, которые не попали под разрешающие правила и не являются уже установленными или связанными с ними.

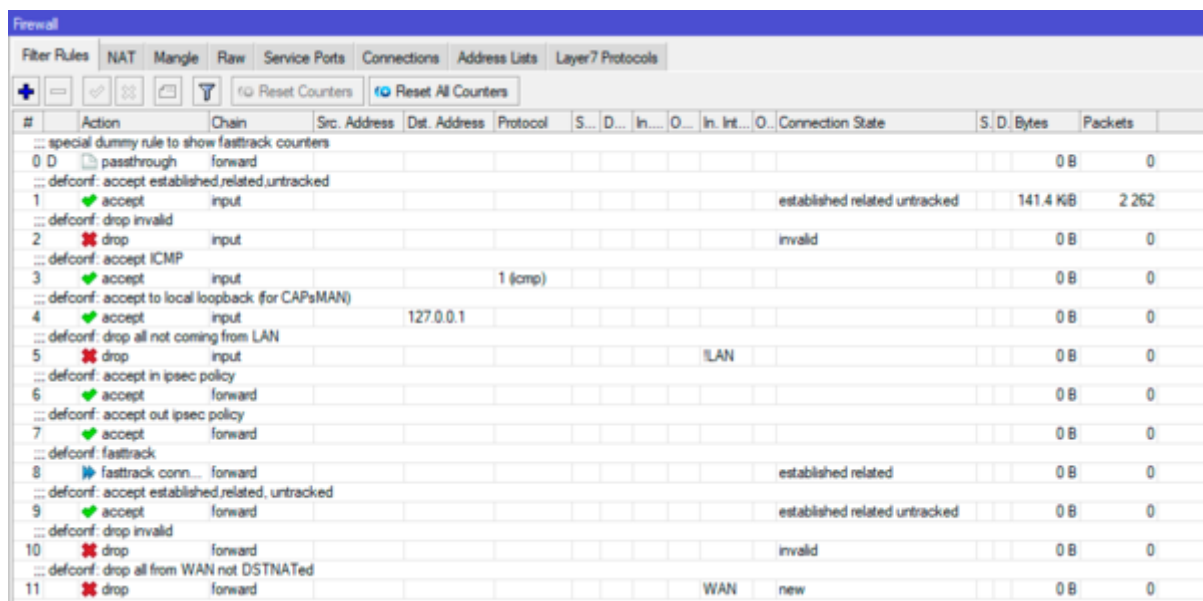
Конфигурация брандмауэра по умолчанию

Прежде всего рассмотрим конфигурацию, предлагаемую производителем по умолчанию. Многие считают ее наиболее оптимальной и безопасной, но это не так, конфигурация по умолчанию направлена на универсальность и совместимость с различными пользовательскими сценариями, не позволяя ему наделать грубых ошибок, которые могут фатально сказаться на безопасности.

Все правила сгруппированы в две цепочки: **input** - для входящего трафика самого роутера и **forward** - для транзитного, между интерфейсами роутера. Сразу напомним, что у роутеров Mikrotik нет жестко заданных "внешних" и "внутренних" интерфейсов, и вообще все эти понятия находятся исключительно у нас в голове, согласно логическому плану сети.

В конфигурации по умолчанию также присутствует два списка интерфейсов: **WAN** - куда включен порт, настроенный как внешний, обычно **ether1** и **LAN**, в состав которого входит мост, объединяющий оставшиеся порты. и эти списки активно применяются в правилах.

Самих правил немного, всего 11 штук: 5 для **input** и 6 для **forward**.



#	Action	Chain	Src. Address	Dst. Address	Protocol	S...	D...	In...	O...	In. Int...	O. Connection State	S. D. Bytes	Packets
0	passthrough	forward										0 B	0
1	accept	input									established related untracked	141.4 KiB	2 262
2	drop	input									invalid	0 B	0
3	accept	input			1 (icmp)							0 B	0
4	accept	input	127.0.0.1									0 B	0
5	drop	input								!LAN		0 B	0
6	accept	forward										0 B	0
7	accept	forward										0 B	0
8	fasttrack conn...	forward									established related	0 B	0
9	accept	forward									established related untracked	0 B	0
10	drop	forward									invalid	0 B	0
11	drop	forward								WAN	new	0 B	0

Рассмотрим их подробнее. Правило с нулевым номером в расчет не берем, это пустышка для вывода счетчиков Fasttrack. Начнем с первых двух, это классическая связка, о которой мы говорили выше: разрешаем **established**, **related** и **untracked** для всех входящих соединений, без привязки к интерфейсам. Затем запрещаем **invalid**, вполне очевидно, что далее пройдут только пакеты соединений в состоянии **new**.

```
/ip firewall filter
add chain=input action=accept connection-state=established,related,untracked
comment="defconf: accept established,related,untracked"
add chain=input action=drop connection-state=invalid comment="defconf: drop
invalid"
```

Третье правило разрешает входящий ICMP, у многих администраторов с ним напряженные отношения, часто его блокируют полностью, чтобы устройство не пинговалось, но так делать не нужно, протокол ICMP важен для нормальной работы сети. Либо фильтровать нужные его типы выборочно, что выходит за рамки данной статьи.

```
add chain=input action=accept protocol=icmp comment="defconf: accept ICMP"
```

Следующее правило весьма специфично, оно разрешает входящие на петлевой интерфейс, точнее адрес 127.0.0.1, поскольку интерфейс петли явно в RouterOS не доступен, а также снабжено комментарием, что это для CAPsMAN. Его смысл станет понятен чуть ниже.

```
add chain=input action=accept dst-address=127.0.0.1 comment="defconf: accept to local loopback (for CAPsMAN)"
```

И завершает цепочку **input** блокирующее правило:

```
add chain=input action=drop in-interface-list=!LAN comment="defconf: drop all not coming from LAN"
```

Которое блокирует входящие подключения **для всех**, кроме интерфейса LAN, а так как локальная петля туда не входит, то выше потребовалось отдельное правило.

Как видим конфигурация слегка избыточна, но при этом универсальна. А блокировка входящих по такой широкой маске - все, кроме локальной сети - позволяет сохранить должный уровень безопасности, даже если пользователь подключит другие внешние сети, скажем VPN.

А где же здесь размещать свои собственные правила? А вот здесь:

```
/ip firewall filter
add chain=input action=accept connection-state=established,related,untracked
comment="defconf: accept established,related,untracked"
add chain=input action=drop connection-state=invalid comment="defconf: drop
invalid"
add chain=input action=accept protocol=icmp comment="defconf: accept ICMP"
<собственные правила>
add chain=input action=accept dst-address=127.0.0.1 comment="defconf: accept to
local loopback (for CAPsMAN)"
add chain=input action=drop in-interface-list=!LAN comment="defconf: drop all not
coming from LAN"
```

Теперь о цепочке **forward**, самыми первыми здесь теперь добавлены два правила, разрешающие транзит соединений, подпадающих под действие политик IPsec. Это связано с тем, что IPsec сегодня применяется все чаще, но правильно настроить правила для него умеют не все пользователи. Поэтому появились эти правила, обеспечивающие гарантированное прохождение любых соединений, использующих IPsec через роутер в любом направлении.

```
add chain=forward action=accept ipsec-policy=in,ipsec comment="defconf: accept in ipsec policy"
add chain=forward action=accept ipsec-policy=out,ipsec comment="defconf: accept out ipsec policy"
```

Затем идет правило, включающее **Fasttrack** для всех установленных и связанных соединений. Подробнее про Fasttrack мы писали в нашей статье:

Правильное использование Fast Path и FastTrack в Mikrotik

Если коротко, то Fasttrack значительно снижает нагрузку на оборудование, но фактически пускает трафик в обход брандмауэра. Но так как фильтровать **established** и **related** нам не надо, то почему бы и не пустить их коротким путем?

```
add chain=forward action=fasttrack-connection connection-state=established,related comment="defconf: fasttrack"
```

Далее та же самая классика: разрешаем **established**, **related** и **untracked**, запрещаем **invalid**. Для всех направлений без разбора.

```
add chain=forward action=accept connection-state=established,related,untracked comment="defconf: accept established,related, untracked"
add chain=forward action=drop connection-state=invalid comment="defconf: drop invalid"
```

Ну и, наконец, запрещающее правило, оно тоже интересно:

```
add chain=forward action=drop connection-state=new connection-nat-state=!dstnat in-interface-list=WAN comment="defconf: drop all from WAN not DSTNATed"
```

Оно запрещает транзит только **новым** соединениям, только с **внешнего списка** интерфейсов, кроме соединений прошедших через DNAT. А так как DNAT обычно используется для проброса портов, то таким образом мы избегаем необходимости создавать для них разрешающие правила.

Да, опять очень и очень общие критерии, но в состоянии абсолютной неопределенности возможных конфигураций это, пожалуй, лучшее решение. Особенно если роутер будет использоваться вместе с UPnP, где сетевые приложения могут самостоятельно пробрасывать произвольные порты.

Поэтому мы еще раз повторимся, что конфигурация по умолчанию не является самой оптимальной или самой безопасной, но она наиболее универсальна, одновременно закрывая самые разные сценарии использования роутера.

Куда добавлять собственные правила? Да вот сюда:

```
add chain=forward action=accept connection-state=established,related,untracked
comment="defconf: accept established,related, untracked"
add chain=forward action=drop connection-state=invalid comment="defconf: drop
invalid"
<собственные правила>
add chain=forward action=drop connection-state=new connection-nat-state=!dstnat
in-interface-list=WAN comment="defconf: drop all from WAN not DSTNATED"
```

Стоит ли оставлять и использовать конфигурацию по умолчанию? Да, если вы еще не уверены в собственных силах, плохо понимаете работу межсетевого экрана или вам нужно просто быстро настроить роутер для типовых задач.

Собственная конфигурация брандмауэра

В среде опытных пользователей Mikrotik принято сбрасывать устройство и настраивать конфигурацию с нуля. Для этих целей мы используем следующую конфигурацию брандмауэра, она во многом повторяет стандартную, но имеет ряд отличий. В плане именования интерфейсов мы также будем использовать листы WAN и LAN, с таким же набором интерфейсов.

При построении данного набора правил мы исходили из соображений, что администратор представляет как устроена его сеть, каким службам нужен доступ извне, какие сети являются внешними, а какие внутренними. При этом всеми силами старались избегать широких критериев в правилах, так как "проходной двор", даже вроде-бы в довольно безопасных вещах, таких как IPsec или DNAT может иногда сыграть злую шутку. Все мы помним, что если на стене висит ружье, то оно обязательно выстрелит.

Правил в нашем варианте меньше, всего 9, из них 4 для **input** и 5 для **forward**.

#	Action	Chain	Src	Dest	Protocol	Src. Dst. I. O. In. Inter...	Out. Int.	Connection State	Src. Dst. Bytes	Packets
0	special dummy rule to show fasttrack counters	forward							0 B	0
1	accept	input					WAN	established related	0 B	0
2	drop	input					WAN	invalid	0 B	0
3	accept ICMP	input			1 (icmp)		WAN		0 B	0
4	drop	input					WAN		0 B	0
5	fasttrack connection	forward					WAN LAN	established related	0 B	0
6	fasttrack connection	forward					LAN WAN	established related	0 B	0
7	accept	forward					WAN	established related	0 B	0
8	drop	forward					WAN	invalid	0 B	0
9	drop	forward					WAN		0 B	0

Набор правил для **input** приведем сразу и целиком, здесь тот же самый классический набор, но за одним исключением: мы знаем, какие сети являются внешними и фильтруем только подключения из них. Это упрощает конфигурацию и снижает нагрузку. При этом в каждом правиле мы четко указываем список входящих интерфейсов.

```
/ip firewall filter
add action=accept chain=input connection-state=established,related in-interface-list=WAN comment="accept established,related"
add action=drop chain=input connection-state=invalid in-interface-list=WAN comment="drop invalid"
add action=accept chain=input in-interface-list=WAN protocol=icmp comment="accept ICMP"
<собственные правила>
add action=drop chain=input in-interface-list=WAN comment="drop all from WAN"
```

Мы точно также разрешаем **установленные** и **связанные**, запрещаем **invalid**, разрешаем ICMP. Можно заметить, что у нас нет **untracked**, это сознательное решение, если мы будем использовать неотслеживаемые соединения - то всегда можем добавить этот параметр, но сделаем это осознанно. Собственные правила мы можем добавлять выше запрещающего весь остальной трафик с внешних интерфейсов.

Также следует отметить несколько иную политику блокировки остального трафика. Стандартная конфигурация подходит достаточно радикально, запрещая все, кроме LAN. Мы же используем более мягкий подход, и блокируем только внешние сети - WAN. Это позволяет сделать конфигурацию проще и не прописывать дополнительные правила для того же CAPsMAN, при этом мы помним, что администратор знает какие сети являются внутренними, а какие нет и контролирует актуальность списков.

Перейдем к **forward**, в нашей базовой конфигурации правил для IPsec нет, опять-таки осознанно, по причине их ненужной избыточности. Как появится IPsec - так и создадим правила, а просто так нечего им небо коптить.

А вот к **Fasttrack** у нас подход другой. В стандартной конфигурации он включен для всех **established** и **related**, вне зависимости от направления. Это здорово разгружает роутер, но со временем у вас неизбежно появятся туннели, VPN, прочие сети и Fasttrack будет там чаще мешать, чем помогать, порой приводя к неожиданным результатам. Поэтому четко указываем, что применяем коротки путь только для соединений из локальной сети наружу и извне в локальную сеть. Часто можно даже отойти от использования списков и просто указать интерфейсы.

```
add action=fasttrack-connection chain=forward connection-state=established,related in-interface-list=WAN out-interface-list=LAN comment="fasttrack"
add action=fasttrack-connection chain=forward connection-state=established,related in-interface-list=LAN out-interface-list=WAN
```

Кстати, внимательный читатель может заметить, что комментарий стоит только у одного правила, первого. Это сделано специально. Для чего? Посмотрите на скриншот выше, там комментарий отделяет сразу два правила.

Ну и дальше все тоже самое. Разрешаем **established** и **related**, запрещаем **invalid** и обязательно указываем, что это именно для внешних сетей, ниже запрещаем все остальное, также для внешних интерфейсов.


```
add action=accept chain=forward connection-state=established,related in-interface-list=WAN comment="accept established,related"
add action=drop chain=forward connection-state=invalid in-interface-list=WAN comment="drop invalid"
<собственные правила>
add action=drop chain=forward in-interface-list=WAN comment="drop all from WAN"
```

Что касается DNAT, то мы не сторонники давать доступ в сеть самыми широкими мазками. Если появляется проброс - создаем отдельное правило под проброс. Может это и увеличивает количество работы, но дает более читаемую и безопасную конфигурацию брандмауэра, когда вы явно видите кому и что разрешено без изучения дополнительных разделов.

Но если вы используете роутер дома и самый широкий набор приложений пробрасывает порты при помощи UPnP, то последнее правило действительно будет удобнее заменить на:

```
add action=drop chain=forward connection-nat-state=!dstnat in-interface-list=WAN comment="drop all from WAN"
```

Данный набор правил является базовым костяком нормально настроенного брандмауэра и должен присутствовать по умолчанию на любом устройстве. Уже потом он будет обрастать дополнительными правилами, но общий принцип построения защиты должен неукоснительно соблюдаться. Это позволит вам достичь высокого уровня безопасности при небольшой нагрузке на устройство.

Онлайн-курс по MikroTik

Научиться настраивать MikroTik с нуля или систематизировать уже имеющиеся знания можно на [углубленном курсе по администрированию MikroTik](#). Автор курса, сертифицированный тренер MikroTik Дмитрий Скоромнов, лично проверяет лабораторные работы и контролирует прогресс каждого своего студента. В три раза больше информации, чем в вендорской программе MTCNA, более 20 часов практики и доступ навсегда.