English • Deutsch • Español • Français • Italiano • ??? • Polski • Português • ??????? • Svenska •
???(????)? • ???(??)? •

You are here: **DD-WRT wiki mainpage / Scripting / SSH/Telnet & The CLI / iptables**

Iptables IPv4 NAT???????????? ???????????Linux????IP?????????.

Iptables ?????? command line interface??,?/?dd-wrt?????????????. ?????????????. ?????????????????route????, ???????????????????. ??????????????????????????????????.

??????????????? Firewall Builder??? , ??????...

Contents

????

```
iptables -[AD] ?? ?? [??]
iptables -[RI] ?? ???? ?? [??]
iptables -D ?? ???? [??]
iptables -[LFZ] [??] [??]
iptables -[NX] ??
iptables -E ??? ???
iptables -P ?? target [??]
iptables -h (????????)
```

??

```
--append -A chain     ?????
--delete  -D chain     ??????????
--delete  -D chain rulenum
                                ?????????rule rulenum (1 = first)
--insert  -I chain [rulenum]
                                Insert in chain as rulenum (default 1=first)
--replace -R chain rulenum
                              Replace rule rulenum (1 = first) in chain
--list      -L [chain]    List the rules in a chain or all chains
--flush   -F [chain]    Delete all rules in  chain or all chains
--zero    -Z [chain]   Zero counters in chain or all chains
--new    -N chain      Create a new user-defined chain
--delete-chain
             -X [chain]   Delete a user-defined chain
--policy  -P chain target
                              Change policy on chain to target
--rename-chain
             -E old-chain new-chain
                              Change chain name, (moving any references)
```

??

```
--proto    -p [!] proto
                              protocol: by number or name, eg. `tcp'
--source  -s [!] address[/mask]
                              source specification. The mask can be either a network mask or a pl
                              specifying the number of 1's at the left side of the network mask.
                              equivalent to 255.255.255.0. A "!" argument before the address spec
                              sense of the address.
--destination  -d [!] address[/mask]
                              destination specification
--sport [!] port[:endport]
                              source port (use `:' when specifying range)
--dport [!] port[:endport]
                              destination port
--in-interface  -i [!] input name[+]
                              network interface name ([+] for wildcard)
--jump    -j target
                              target for rule (may load target extension)
--match  -m match
```

```
                              extended match (may load extension)
--state state
                              connection states to match:
                                   INVALID NEW ESTABLISHED RELATED
--tcp-flags [!] mask
                              match when the TCP flags are as specified:
                                   SYN ACK FIN RST URG PSH ALL NONE
--numeric    -n         numeric output of addresses and ports
--out-interface -o [!] output name[+]
                              network interface name ([+] for wildcard)
--table        -t table  table to manipulate (default: `filter')
--verbose   -v          verbose mode
--line-numbers         print line numbers when listing
--exact       -x          expand numbers (display exact values)
--fragment  -f          match second or further fragments only
--modprobe=<command>     try to insert modules using this command
--set-counters PKTS BYTES    set the counter during insert/append
--version    -V          print package version

MAC v1.3.7 options:
 --mac-source [!] XX:XX:XX:XX:XX:XX
                              Match source MAC address
```

??

When using the -i or -o to define the physical interfaces, remember that by default:
vlan0 is the 4 LAN ports
vlan1 is the WAN port (ppp0 is the WAN interface when PPPoE is used)
eth1 is the WIFI
br0 is a bridge connecting the 4 LAN and the WIFI together

Note: ppp0 is the WAN interface when PPPoE is used. This information is from IPv6 page and quoted here: "The detailled configuration steps are targeted toward users with a basic DHCP connection for the WAN part. So, if using PPPoE will require replacing vlan1 with ppp0 in each instance. Other connection types will vary."

Tip: To list the network interfaces on the router use 'ifconfig' on the command line.

?Tables, ?Chains, ???Targets

?Tables

The main tables we are concerned with are the "filter" table and the "nat" table. To list the contents of either table, do

```
iptables -t filter -L
iptables -t nat -L
```

The filter table is default and this includes chains like INPUT, OUTPUT, and FORWARD. The nat table is for Network Address Translation and it includes the PREROUTING and POSTROUTING chains.

?Chains

INPUT is for packets destined to or entering the router's local sockets.

OUTPUT is for packets sourced from or leaving the router's local sockets.

FORWARD is for packets being forwarded through the router (e.g. packets not necessarily destined for local sockets).

PREROUTING is for manipulating packets before they are routed.

POSTROUTING is for manipulating packets after they are routed.

??Targets

ACCEPT - packets are accepted/allowed

DROP - packets are dropped/denied

REJECT - packets are rejected/denied

logaccept - packets are accepted and logged to /tmp/var/log/messages

logdrop - packets are dropped and logged to /tmp/var/log/messages

logreject - packets are rejected and logged to /tmp/var/log/messages

DNAT is for altering packet's destination address.

SNAT is for altering packet's source address.

TRIGGER - dynamically redirect input ports based on output traffic (aka port triggering)

TRIGGER Target Options

The trigger target has additional options which must appear immediately after it on the command line

```
--trigger-type [dnat|in|out]
--trigger-proto [tcp|udp|all]     (if this option is not specificed the default is all)
--trigger-match [port[-port]]    (a port or a range of ports which the outbound connection uses)
--trigger-relate [port[-port]]     (a port or range of ports to open on the inbound side)
```

??

I think examples are the best way to demonstrate the use of iptables. (Take note, chains are to be typed in caps as shown!)

Listing the rules in a chain

????????INPUT????????????????????????

```
iptables -L INPUT
```

You will find that it is really slow to list all many rules after you enter the above iptables command since it is doing reverse DNS lookups to convert IP addresses to host names. You can add -n option to only see numerical addresses. (Note '0.0.0.0/0' = 'anywhere' (any address, any port); and '0' prot is 'any' protocol.)

```
iptables -nL INPUT
```

To get a little more detailed list, with actual IP numbers and packet counts for each rule, I can do

```
iptables -vnL INPUT
```

Suppose I might want to add a rule so that I can ssh into my router from a specific host/address outside. Then I might type the following:

```
iptables -A INPUT -p tcp -s 150.100.whatever.something --dport 22 -j logaccept
```

So I am saying: Append to the INPUT chain a rule allowing protocol tcp, with a source address of <my external IP that i want access from> traffic destined for port 22 on my router, jump to logaccept. I could have used -j ACCEPT which simply jumps to ACCEPT, but in this case I want to log it just to keep track so I use logaccept, which is a chain we have set up for this purpose.

Note: Simply adding a rule to the INPUT chain may be enough to allow remote SSH access from the WAN. However, if your router is still in NAT/Gateway mode and you wish to remap the SSH port to something less traditional on the WAN side (say port 2222), you may Insert a PREROUTING rule instead. This is actually how the GUI does it when you enable remote WAN SSH management.

```
iptables -A INPUT -p tcp -m tcp -d `nvram get lan_ipaddr` --dport 22 -j logaccept
iptables -t nat -A PREROUTING -p tcp -m tcp -d `nvram get wan_ipaddr` --dport 2222 -j DNAT --to-d
```

Now if I type

```
iptables -L INPUT
```

I see my shiny new rule appended to the INPUT chain. However, this is no good because in my case I have a rule blocking this traffic which occurs BEFORE the rule allowing it.

How do I change it? Simple.

First let's delete the rule we just made

```
iptables -L INPUT --line-numbers
```

will list the rules with their rule numbers. Let's say our rule is number 11

```
iptables -D INPUT 11
```

Clearly this Deletes rule number 11 from the input chain.

Now instead of Appending I am going to Insert my rule into the number 1 (by default) position.

```
iptables -I INPUT -p tcp -s 150.100.whatever.something --dport 22 -j logaccept
```

So now rule number 1 is my new rule and the other rules have all shifted down a position.

If I wanted to change the IP address or any other aspect of my ssh rule I could use the -R (Replace) option for a specific rule number, and simply type in the new rule, i.e.

```
iptables -R INPUT 1 -p tcp -s 100.100.200.100 --dport 22 -j ACCEPT
```

This would replace rule number 1 on the INPUT chain with the new rule which has a new source IP address and jumps to ACCEPT instead of logaccept.

One more example: I want to run a mini web server on my router. Let's assume that it is already running on port 8000 and I can access it from the LAN side, but not from the WAN side. With

```
iptables -I INPUT -p tcp -d 192.168.1.1 --dport 8000 -j logaccept
```

the port 8000 will be opened. But I also have to setup NAT PREROUTING, so that the kernel forwards all packets on port 8000 from the outside to itself, 192.168.1.1:

```
iptables -t nat -I PREROUTING -p tcp -d $(nvram get wan_ipaddr) --dport 8000 -j DNAT --to 192.168
```

??????????IPPort Forwarding to a specific LAN IP

Port Forwarding can be accomplished from within the web interface here. However, the very same thing can be done a bit differently (tested and working), via command line. --u3gyxap: Example with port 443 and IP 192.168.1.2

```
iptables -t nat -I PREROUTING -p tcp -d $(nvram get wan_ipaddr) --dport 443 -j DNAT --to 192.168.
iptables -I FORWARD -p tcp -d 192.168.1.2 --dport 443 -j ACCEPT
```

If you want to restrict the source IP (a question that is asked a lot on the forums), add *-s 123.45.67.89* to one of your rules (replacing the IP address with the real one of course).

```
iptables -t nat -I PREROUTING -p tcp -s 123.45.67.89 -d $(nvram get wan_ipaddr) --dport 443 -j DN
iptables -I FORWARD -p tcp -d 192.168.1.2 --dport 443 -j ACCEPT
```

This should make it so only one IP address is able to access your forwarded port from the Internet.

In order for me to get this to work (v.24) I needed to put the "-s 123.45.67.89" in the "iptables -I FORWARD" command also - When it was in the PREROUTING command only I was still able to access the internal resource from any IP address!

??????IPDeny access to a specific IP address

```
iptables -I FORWARD -d 123.123.123.123 -j DROP
```

Which would DROP all packets destined to the given IP. Useful to block access to whatnot. If you want to log the entry when the IP is blocked you would set the jump location to logdrop, instead of DROP.

????????Deny access to a specific Subnet

```
iptables -I FORWARD -s 192.168.2.0/255.255.255.0 -j DROP
```

Deny access to a specific IP address range with Logging

```
iptables -I FORWARD -m iprange --src-range 192.168.1.10-192.168.1.13 -j logdrop
```

Many builds do not have the iprange match but you can use clever subnet masks to accomplish something similar as well, if the range aligns well on subnet boundaries. You may also be able to download a version of iptables that includes the iprange match via Optware.

Deny access to a specific Outbound IP address with logging

```
iptables -I OUTPUT -d 239.255.255.250 -j logdrop
```

This becomes useful if there is a program that wants to gain an outbound connection to a specific address, but you don't want to allow the connection. In this specific example Windows uses this IP incorrectly as a broadcast address (search Google for more info). While viewing your router logs you will see Windows broadcast to this IP several times per minute. By default the router passes the broadcast and announces to everyone outside of your router that your PC exists. This rule will block traffic to this specific outbound IP and add an entry into the router log.

Block SMTP traffic except to specified hosts

Simple Mail Transfer Protocol operates on tcp port 25.

```
/usr/sbin/iptables -I FORWARD 1 -p tcp -d safe.server1.com --dport 25 -j logaccept
/usr/sbin/iptables -I FORWARD 2 -p tcp -d safe.server2.com --dport 25 -j logaccept
/usr/sbin/iptables -I FORWARD 3 -p tcp --dport 25 -j logdrop
```

Which would accept and log all smtp traffic to safe.server1.com and safe.server2.com, while blocking and dropping all other outgoing smtp traffic.

Block outgoing SMTP traffic except from specified hosts

Simple Mail Transfer Protocol operates on tcp port 25.

```
iptables -I FORWARD 1 -p tcp -s 192.168.1.2 --dport 25 -j ACCEPT
iptables -I FORWARD 2 -p tcp -s 192.168.1.1/24 --dport 25 -j REJECT
```

Which would accept outgoing SMTP traffic from your internal SMTP server (192.168.1.2) but reject outgoing SMTP traffic from all other hosts on your LAN (192.168.1.1/24). Useful to enforce all your LAN clients to use your internal SMTP server, as well as to block any viruses and spam-generating trojans from sending mail to remote servers on their own.

Change "REJECT" to "logdrop" or "ACCEPT" to "logaccept" to add logging.

Caution! This will also block internal users from using your external IP as their SMTP server.

Allow HTTP traffic only to specific domain(s)

Similarly, we can use the above method to filter other ports and protocols as well, such as standard web traffic operating on tcp port 80.

```
iptables -I FORWARD 1 -p tcp -d dd-wrt.com --dport 80 -j ACCEPT
iptables -I FORWARD 2 -p tcp --dport 80 -j DROP
```

Which would accept all http traffic to dd-wrt.com, while blocking outgoing http traffic to anywhere else. If you wish to allow multiple sites, insert additional rules before the DROP (making sure to order and number them correctly).

Block all traffic except HTTP HTTPS and FTP

This example blocks everything except our normal web traffic, encrypted (ssl), and the file transfer protocol.

```
iptables -I FORWARD 1 -p tcp -m multiport --dports 21,80,443 -j ACCEPT
iptables -I FORWARD 2 -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -I FORWARD 3 -j DROP
```

Caution! Users are still able to get through the firewall if they are sly enough to use these permitted port numbers for their P2P or other application. In that case, you should consider using Access Restrictions to mitigate the possibility of that happening.

Reject clients from accessing the router's configuration

This should prevent clients on the LAN (interface br0) from accessing the configuration interface of the router through any of the following ports (telnet/23, ssh/22, http/80, https/443)

```
iptables -I INPUT -i br0 -p tcp --dport telnet -j REJECT --reject-with tcp-reset
iptables -I INPUT -i br0 -p tcp --dport ssh -j REJECT --reject-with tcp-reset
iptables -I INPUT -i br0 -p tcp --dport www -j REJECT --reject-with tcp-reset
iptables -I INPUT -i br0 -p tcp --dport https -j REJECT --reject-with tcp-reset
```

Tip: If you disable management from the LAN, be sure to enable remote management on the WAN (or vice versa) or you will probably lock yourself out of the router.

Restrict access by MAC address

In this example, we will demonstrate how to restrict access to the router's web interface by MAC address. In other words, only the computer having the specified MAC address should be able to access the web interface from the LAN.

First, if there are no Access Restrictions policies enabled and filtering by MAC addresses, you may need to insert the iptables mac module manually:

```
insmod ipt_mac
iptables -I INPUT -p tcp --dport 80 -m mac ! --mac-source 00:12:34:56:78:9A -j REJECT --reject-wi
```

Notice the ! (bang) which is another new concept introduced here. It means "NOT". So, by inspecting the rule closely, we see that it will REJECT packets destined to port 80 of the router so long as they do NOT originate from our computer with the desired MAC address.

Caution! As usual when dealing with MAC addresses, be aware that it is possible for malicious user(s) to spoof their MAC address with that of a trusted machine. You can help combat this by use of static ARP entries, VLANs, etc.

Modifying the TTL

The Time To Live is the maximum number of routers a packet will travel through before it is discarded. In certain situations, it may prove useful to increase it (typically) in order to make your network more reliable.

- **Example 1:** Set the incoming TTL to 10, *before* the router routes it into the LAN

```
iptables -t mangle -I PREROUTING -i `nvram get wan_iface` -j TTL --ttl-set 10
```

- **Example 2:** Set the outgoing TTL to 128, just as if a Windows machine was connected directly to the modem.

```
iptables -t mangle -I POSTROUTING -o `nvram get wan_iface` -j TTL --ttl-set 128
```

- **Example 3:** Try to hide the fact that an outgoing packet was routed, by incrementing the TTL by one.

```
iptables -t mangle -I POSTROUTING -o `nvram get wan_iface` -j TTL --ttl-inc 1
```

Firewall Forwarded Ports

?????DD-WRT?SPI ?????, ??????????. ?iptables???????????????IP????????????????:

- ??DD-WRT?web??"????"??????IP??????
- Supplement those rules with custom iptables on the Firewall script found under Administration - Commands interface to restrict which hosts can access the ports involved

??????

The current port forwarded setup via the web GUI will be used as the basis to illustrate some examples:

- Application: ssh Port: 4022 Protocol: TCP forward to IP address: 192.168.1.5 Port: 22
- Application: ftp Port: 21 Protocol: TCP forward to IP address: 192.168.1.6 Port: 21

The example here port forwards external IP on port 4022 to internal server 192.168.1.5:22 for ssh and external port 21 to internal server 192.168.1.6:21 for ftp.

???????

??????ssh?????????, and prevent brute force attack. ???????????????3??ssh??. ???????ssh?????????:

```
iptables -I FORWARD -p tcp -d 192.168.1.5 --dport 22 -j DROP
iptables -I FORWARD -p tcp --dport 22 -m state --state NEW -m limit --limit 3/min -j ACCEPT
iptables -I FORWARD -p tcp --dport 22 -m state --state RELATED,ESTABLISHED -j ACCEPT
```

FTP access can also be limited to a certain network or network range in the following manner:

```
iptables -I FORWARD -p tcp -d 192.168.1.6 --dport 21 -j DROP
iptables -I FORWARD -p tcp -s 198.133.219.0/24 --dport 21 -j ACCEPT
iptables -I FORWARD -p tcp -s 69.147.64.0/18 --dport 21 -j ACCEPT
iptables -I FORWARD -p tcp --dport 21 -m state --state RELATED,ESTABLISHED -j ACCEPT
```

You can of course combine both, rate limit and IP addresses limiting. This following example limits ssh connection from 207.171.160.0/19 with the same rate limit applied, along with the FTP rules all on the same script:

```
# limit SSH connection from script kiddies in Amazon's network
iptables -I FORWARD -p tcp -d 192.168.1.5 --dport 22 -j DROP
iptables -I FORWARD -p tcp -s 207.171.160.0/19 --dport 22 -m state --state NEW -m limit --limit 3
iptables -I FORWARD -p tcp --dport 22 -m state --state RELATED,ESTABLISHED -j ACCEPT
# limit FTP connection to two subnets
iptables -I FORWARD -p tcp -d 192.168.1.6 --dport 21 -j DROP
iptables -I FORWARD -p tcp -s 198.133.219.0/24 --dport 21 -j ACCEPT
iptables -I FORWARD -p tcp -s 69.147.64.0/18 --dport 21 -j ACCEPT
iptables -I FORWARD -p tcp --dport 21 -m state --state RELATED,ESTABLISHED -j ACCEPT
```

For **multiport** INPUT (or FORWARD if you choose) rate limiting, the following syntax rules can be esstablshed:

```
wanf=`nvram get wan_iface`
iptables -I INPUT 2 -i $wanf -p tcp -m multiport --dports 21,22,80,82 -j logdrop
iptables -I INPUT 2 -i $wanf -p tcp -m state --state NEW -m multiport --dports 21,22,80,82 -m lim
```

To verify that the rules are working, open a terminal session and type **iptables -vnL | more**

If you're adding a lot of rules, it helps to separate them with comments using the # prefix. You can of course use the basics of iptables explained in this article to make your rules more complex to suit your needs. However, instead of ordering the rules, the examples here merely insert these new ones on top of the FORWARD chain. This ensures that the firewall rules that limit traffic appear on top of the chain and gets applied first.

You may examine these rules on the router at anytime by accessing the router's command prompt and running the command "iptables -vnL"

Logging

You can consider turning on logging temporarily for any of your rules. This is useful if you're testing new setup to confirm that the rules are doing what you intend to block or allow. First enable logging via the web UI at Security - Firewall tab. Then substitute the jump target or "-j" to a logging target for each of your iptables rule:

- DROP with logdrop
- REJECT with logreject
- ACCEPT with logaccept

Example if you wanted to check and confirm if traffic forwarded to port 21 is correctly dropped you would substitute:

```
iptables -I FORWARD -p tcp -d 192.168.1.6 --dport 21 -j DROP
```

with

```
iptables -I FORWARD -p tcp -d 192.168.1.6 --dport 21 -j logdrop
```

Logged data can be viewed on the web UI on the same page or on the command prompt in the file "/var/log/messages"

Firewall blocks DHCP renewal responses

```
iptables -I INPUT -p udp --dport 68 -j ACCEPT
```

The default configuration of the firewall blocks DHCP renewal responses which causes the router's DHCP client to request a new IP and for current connections to be dropped whether the address changes or not. ~phuzi0n Use this command to fix it. Replace ACCEPT with logaccept to verify it is functioning.

??

???????? iptables ??????????????????????????????????. ????????????iptables????????????????????????. ?????????????????5???sleep 300?????????????????????????????????5?????????.

??

- One-to-one NAT
- Port Blocking
- Preventing Brute Force Attacks
- Logging with DD-WRT

????

- blocking all traffic except HTTP HTTPS and MSN (Windows Live Messenger) With iptables
- linux_journal_video_tut
- http://linux.die.net/man/8/iptables
- http://www.iptables.org/documentation/HOWTO//packet-filtering-HOWTO-7.html
- http://www.iptables.org/documentation/HOWTO//netfilter-hacking-HOWTO.html
- iptables command line examples