



Netfilter Capabilities in the SIXNET IPm

Abstract:

This document regards the details of implementing IP Security in the IPm. In the IPm, there is the option to enable IP Security. In a properly configured station with IP Security enabled, it is not only possible to protect your network from unwanted users but also to communicate between different sub networks, filter specific protocols and ports, and more.

Users have two methods of configuring IP Security to take advantage of these features. The primary and user-friendlier method is through the SIXNET I/O toolkit. The secondary, more hands on method, is configuring manually through a file called rc.firewall. There are no disadvantages to using either method, as they both yield the same effective results.

Both methods will be discussed in detail, along with examples on how to use a utility called iptables. iptables is the vital utility needed to tailor IP Security in the IPm. Since iptables plays such an important role, most of this document is dedicated to familiarizing the user with the utility.

Iptables is a GPL application developed through the Netfilter Organization. To find more information about the rule options used with iptables regarding filtering, routing, and other topics, check out www.netfilter.org. The site provides extensive documentation and examples on how to implement your desired configuration, no matter if you are a novice or advanced user of Netfilter software.

IP Security Configuration in the Sixnet I/O tool kit

To take advantage of the Netfilter capabilities such as filtering and routing (IP traffic redirection) in the IPm, users don't have to be familiar with Linux at all. Available in the Sixnet I/O tool kit are options to configure an IPm to take advantage of built in Netfilter capabilities. To do so, when configuring the Ethernet ports of the IPm, click advanced. This will bring up a special window with 3 major options, Advanced Ethernet, Security Options, and Advanced Security.

Advanced Ethernet Tab

Under Advanced Ethernet, the user can configure his/her own subnet masks. In addition, gateway settings are available if the station is to be used as a means of communication with a different network.

Security Options Tab

In the security options tab, there are four options.

The first is labeled "Do not modify the IP security rules file during the load". The purpose of this option is to prevent erasure of an IPm's security rules already customized correctly for the particular station.

The second is labeled "Load this file with each station load". Assuming that there is a master or a most recent copy of the rc.firewall file available, the IPm being loaded is ensured a refreshed copy of the rc.firewall file selected by the user. (Please note that the rc.firewall file is where the user will place all of his/her customized set of rules for the particular IPm.)

The third option is "Disable IP security". This option basically states that there are no user specified rules to be put into the rc.firewall file.



The fourth option is labeled “Load the following IP security rules, and ‘Advanced’ rules”. This option is usually used if the user wants to implement a custom set of rules, but has not opted for option number two (“load this file with each station load”). In this case, there are several features available once this option has been selected. If all the user wants to do is allow a certain range of IP addresses through, but filter out others that are not within the desire range, then the range fields can be filled out. For the stations that are “secured stations” the addresses of those stations should be filled out in the given fields. Range and secured address information is all translated into the necessary rules to be placed in the rc.firewall file. These set of rules generated are ready for placement in the rc.firewall file inside the station.

Advanced Security Tab

For the users that want to add more rules to take advantage of post and pre routing operations, routing, and more filtering, here is where more rules can be placed. Simply select the Add Rule Wizard button and the wizard will take the user through the steps of filtering addresses, protocols, and ports. If the user wants to perform more advanced operations, such as routing, rules will have to be entered manually. For more details on implementing the rules in a Linux environment, or how to perform basic routing, please refer to the information below or the Netfilter site, www.netfilter.org.

Netfilter Rules in the Linux Environment

Kernel Support

The Linux kernel has already been compiled for Netfilter support. For a user to customize their setup, it is his/her choice on the particular module(s) to be included in the configuration.

The following Netfilter modules have been included in the IPm:

- ip_tables ← main body of iptables module
- ip_filter ← filtering module, auto loaded when filtering policies are enabled
- ip_conntrack ← stateful connection tracking, auto loaded when MASQUERADING enabled
- iptable_nat ← network address translations code, auto loaded when MASQUERADING enabled
- ipt_MASQUERADE ← IP masquerading code
- ipt_state ← check packets where IP and TCP flags set/unset
- ipt_REDIRECT ← alters destination IP addresses
- ipt_REJECT ← allows packets to be dropped and error message replies

For a minimal filter setup, the following modules should be included:

- ip_tables
- ip_filter
- ip_conntrack



To enable the desired set of modules, add to the top of the `/etc/rc.firewall` file with the `insmod` command.

For example:

```
insmod ip_tables
```

User-space Programs/Configuration Files

iptables-1.2.7a.tar.gz

```
usr/sbin/iptables
usr/sbin/iptables-restore
usr/sbin/iptables-save
etc/rc.firewall
```

Editing the `rc.firewall` file

In the `/etc` directory of the IPm, there is a file named `rc.firewall`. This file is used as a script to initialize all the rules that the user wants. Once these rules are initialized, they are run behind the scenes as a “table” of IP packet filter rules.

It is important to flush the memory every time a new set of rules is to be implemented in the IPm. Below is a general guideline for flushing out pre-existing policies/rules that might have been instated during the last system initialization.

To flush, put these lines at the top of the `rc.firewall` file.

```
iptables -F input
iptables -F output
iptables -F forward
iptables -t nat -F
```

To clear all user specified chains (where a chain is a set of rules):

```
iptables -X
```

In addition, every time a packet matches the criteria of a rule, a counter is increased. To zero that counter, put in the line:

```
iptables -Z
```



Lastly, to check if all the rules have been flushed and everything is starting off from a blank slate, put in the line:

```
iptables -L
```

The line above will list out all the rules currently in the station; which should now be an empty list.

Before filtering, it is important to initialize ip forwarding capabilities. To do so, type in:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

Filtering

To implement a packet filtration system, the user can implement his/her own customized rules. The set of rules that work together for a single purpose is called a chain.

For example, say the user wants to block all incoming packets going from a public into a private network. A chain is therefore created to implement this goal. To implement a new chain, type in the lines below into the rc.firewall file (after the flushing and initialization of modules).

Example #1

```
iptables -N nopub  
iptables -A nopub -m state -state ESTABLISHED, RELATED -j ACCEPT  
iptables -A nopub -m state -state NEW -i ! eth1 -j ACCEPT  
iptables -A nopub -j DROP  
iptables -A INPUT -s 127.0.0.0/8 -j ACCEPT  
iptables -A INPUT -j nopub  
iptables -A FORWARD -j nopub
```

Starting from the first line, the `-N` option states that the user wants to create a new chain with the name 'nopub'.

The second, third, and fourth lines append rules to the 'nopub' chain with the `-A` option.

The `-m` option is the important part towards filtering packets because it states in the rule that the user wants to match a specific criteria.

So looking at the second line and reading it as it were English, it would read like, "I am appending a rule to my nopub chain; where the current packet is associated with a connection that sees packets in both directions and is also trying to establish a connection which already exists."

The third line matches packets that are new from the private network and are trying to establish a new connection. All packets from the external interface are to be rejected for connection (assuming that the eth1 interface is the external connection).



All other packets are dropped, as indicated on the fourth line.

Important:

Line 5 is essential and is used to ensure that the IPm sustains working internal communications.

Lines 6 and 7 are linking the noup chain to the INPUT and FORWARD chains. INPUT refers to packets coming into the station, and FORWARD is refers to packets going through the station.

Routing

Routing is very important in performing packet pass-through/redirection through the station. Below are a few examples, which could prove useful.

Example #2

```
iptables -A FORWARD -p tcp -dport 80 -j ACCEPT
```

Example #2 states that all packets which are of type tcp and are going to port 80, forward on.

Example #3

```
iptables -A FORWARD -i eth1 -o eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
```

In example 3, we are assuming the interface for the internal network is eth0, and the interface for the external network is eth1. Therefore, translated, it says only allow external traffic to go to established and known computers on the internal network. In addition, let all traffic coming from the internal network to be passed through to the external network.

Example #4

```
iptables -t nat -A PREROUTING -p tcp -dport 80 -i eth0 -j DNAT -to 10.10.10.10
```

In this example, all packets that are tcp and are directed to port 80 (web traffic) are going to be redirected to 10.10.10.10 . DNAT specifies the destination address of the packet should be modified.

Example#5

```
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

This example states that after routing; perform masquerading on all packets going out to the external interface. Masquerading is a way to tell your station that all packets coming from your internal network are just coming from a single station.



Sample rc.firewall file

```
#!/bin/sh
echo 1 > /proc/sys/net/ipv4/ip_forward

# Enable iptables kernel modules

insmod ip_tables
insmod iptable_filter
insmod ip_conntrack
insmod iptable_nat
insmod ipt_MASQUERADE
insmod ipt_state
insmod ipt_LOG
insmod ipt_ULOG

# Flush out the rules
iptables -F INPUT
iptables -F OUTPUT
iptables -F FORWARD
iptables -t nat -F

# Zero out the counter

iptables -Z

# Add default policies
iptables -P INPUT ACCEPT

#allow local traffic
iptables -A INPUT -s 127.0.0.0/8 -j ACCEPT

iptables -P FORWARD DROP

iptables -P OUTPUT ACCEPT

echo "listing rules in all chains"

iptables -L

iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT

iptables -A PREROUTING -t nat -p tcp -i eth1 -j DNAT --to 10.3.0.13

iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT

iptables -t nat -A POSTROUTING -o eth1 -s 10.0.0.0/8 -j MASQUERADE
```