

# Introduction to IP Filter

*Jeremy Rauch* 2000-07-17

10.20.30.0/24:ANY	10.20.30.5:22	TCP	ALLOW
ANY:ANY	10.20.30.5:80	TCP	ALLOW
ANY:ANY	ANY:ANY	UDP	BLOCK
10.20.30.5:ANY	10.20.30.13:53	UDP	ALLOW(STATE)
10.20.30.5:ANY	10.20.30.14:53	UDP	ALLOW(STATE)

Any outbound TCP connections are allowed. Hosts in the 10.20.30.0/24 range will be allowed to connect via SSH. Any host may connect to the webserver. Finally, stateful connections will be allowed to the two nameservers, 10.20.30.13 and 10.20.30.14 on port 53.

By following a policy of only allowing specific connections, and denying everything else, we can help protect ourselves against a variety of errors, as well as users attempting to run daemons. For example, applying patches can often result in services being re-enabled. This way, even if a service is unintentionally re-enabled, they cannot be contacted.

## . Writing the rules

For this first article, we're going to keep our rules simple and straight forward. We will write them efficiently, but we won't use logging, for example. We'll use features such as `keep state` and `flags`, but we'll only touch on them briefly; we will explain these more fully in the second part of this article.

### 1. *Block everything*

First, we'll set up the initial rules blocking all TCP and UDP traffic.

```
EBI:/etc/opt/ipf(550) cat ipf.conf
# outbound rules
block out on hme0 proto tcp all

# inbound
block in on hme0 proto tcp all

# UDP rules
block out proto udp all
block in proto udp all
```

Very simple. We are saying to block all incoming TCP and UDP connections to hme0, our primary interface. This will prevent any packets from reaching the machine. Secure, for sure, but fairly useless if you plan on using the network.

The `block` and `pass` do exactly that. They set up a rule that either blocks or passes an incoming packet.

`in` and `out` determine which direction to apply this rule to. `in` is packets headed into the interface, and `out` is packets leaving the interface.

`on xxx` dictates the interface to apply this rule to. `proto xxx` dictates which protocol to apply the rule to.

## 2. Basic rules

Next, we'll add our TCP and UDP rules. Again, we'll keep them basic, and make them better later.

```
EBI:/etc/opt/ipf(550) cat ipf.conf
# outbound rules
block out on hme0 proto tcp all
pass out on hme0 proto tcp from any to any

# inbound
block in on hme0 proto tcp all
pass in on hme0 proto tcp from any to any port = 80
pass in on hme0 proto tcp from 10.20.30.0/24 to any port = 22

# UDP rules
block out proto udp all
block in proto udp all
pass in proto udp from 10.20.30.13 port = 53 to any
pass out proto udp from any to 10.20.30.13 port = 53
pass in proto udp from 10.20.30.14 port = 53 to any port
pass out proto udp from any to 10.20.30.14 port = 53
```

Ok, a little more complex now. Our first addition to the configuration file was to allow the passing of anything outbound on hme0. This seems in direct conflict with the rule prior to it, where we blocked all outbound TCP connections. It is -- this rule will be changed in our next step to be more useful.

Inbound, we allow any packet destined to port 80 to pass, and any packet from the 10.20.30.0/24 range to pass on port 22. Finally, to get DNS to work, we allow packets to our nameservers, going to port 53 to be sent, and packets from our nameservers, coming from port 53 to pass.

`from x` and `to y` dictates the source and destination addresses to apply this rule to. `any` means to apply the rule to every source or destination address. The `port = xx` keyword specifies the port packets must be to or from to match this rule.

These rules are OK, and in many ways could be sufficient. Due to the nature of the rules, however, packets which should never be seen outside of a stateful connection would be passed. Also, any packet from our nameservers originating on port 53 would be allowed to pass the UDP rules. This could allow for some nasty attacks.

### 3. *Better rules*

In order to improve our security situation, we want to start using the `keep state` keyword. This allows IP Filter to keep basic state information. This information is stored in a table, and is utilized to match inbound and outbound connections to one and other. This will help prevent random packets from connecting to ports we pass, and prevent the nameservers from connecting to any UDP service.

```
EBI:/etc/opt/ipf(579) cat ipf.conf
# outbound rules
block out on hme0 proto tcp all
pass out on hme0 proto tcp from any to any keep state
# inbound
block in on hme0 proto tcp all
pass in on hme0 proto tcp from any to any port = 80 keep state
pass in on hme0 proto tcp from 10.20.30.0/24 to any port = 22
keep state

# UDP rules
block out proto udp all
pass out proto udp from any to 10.20.30.13 port = 53 keep
state
pass out proto udp from any to 10.20.30.14 port = 53 keep
state
block in proto udp all
```

Our rules are actually a little simpler now. We use the `keep state` keyword on the TCP connections to force normal connection establishment. This will make it so only

established connections can pass packets. Once the connection is established, these packets can pass without being touched by the firewall. While this may sound dangerous, it in fact improves the effectiveness of our rules. In particular, the UDP rules for DNS are far more stringent -- packets coming from the nameserver must have a matching outbound request, or they will be refused.

#### 4. *Making our rules efficient*

Our rules are pretty secure, but they aren't particularly efficient. We'll use grouping, as well as the `quick` keyword to improve processing of rules. Rather than apply all rules to all packets, we'll make it so only appropriate rules are applied to each packet. Once a rule is appropriate to a packet, it will prevent unnecessary future checks.

```
EBI:/etc/opt/ipf(586) cat ipf.conf
# outbound rules
block out quick on hme0 proto tcp all head 100
pass out quick on hme0 proto tcp from any to any flags S/SA
keep state group 100

# inbound
block in quick on hme0 proto tcp all head 200
pass in quick on hme0 proto tcp from any to any port = 80
keep state group 200
pass in quick on hme0 proto tcp from 10.20.30.0/24 to any
port = 22 keep state group 200

# UDP rules
block out quick proto udp all head 300
pass out quick proto udp from any to 10.20.30.12 port = 53
keep state group 300

block in quick proto udp all head 400
```

The `quick` keyword will make IP Filter stop processing rules outside of a specific group. The simplest way to understand it is to look at the inbound TCP rules. Upon receipt of a TCP packet, the first rule triggered is the block rule. When this happens, due to the `quick` keyword, IP Filter only processes packets also in the group this rule has established. We've set this to be group 200. If the port the TCP packet is destined to is 80, its allowed. If its to port 22, and the source address is in the 10.20.30.0/24 range, it is allowed. Otherwise, it is dropped. Fairly straight forward.

The other change you might note is the inclusion of the `flags` keyword in the rule to

allow all outbound TCP connections. This keyword only allows packets with a bare SYN to place entries in our state tables. We'll discuss this more in the next article. The S/SA specifies the flag to match (SYN) with a flag mask of SA (SYN and ACK). This prevents SYN-ACK packets from establishing an entry in the state machine.

## • Installing the rules

Installing the rules is simple.

```
EBI:~(600) /sbin/ipf -Fa -f /etc/opt/ipf/ipf.conf
```

This will flush the existing rules (there are none), and replace them with the contents of the ipf.conf file. If there are any errors, ipf will indicate the rule which failed to load properly. To see the rules:

```
EBI:~(605) /sbin/ipfstat -io
block out quick on hme0 proto tcp from any to any head 100
pass out quick on hme0 proto tcp from any to any flags S/SA keep
state group 100
block out quick proto udp from any to any head 300
pass out quick proto udp from any to 10.20.30.13/32 port = 53 keep
state group 300
pass out quick proto udp from any to 10.20.30.14/32 port = 53 keep
state group 300
block in quick on hme0 proto tcp from any to any head 200
pass in quick on hme0 proto tcp from any to any port = 80 keep
state group 200
pass in quick on hme0 proto tcp from 10.20.30.0/24 to any port = 22
keep state group 200
block in quick proto udp from any to any head 400
```

These rules will automatically be installed upon reboot by the IP Filter startup scripts.

## • Conclusion

Installing and utilizing IP Filter should not be viewed as a cure-all. IP Filter is an excellent first line of defense, but for security's sake, its still important to take the time to remove unneeded services, and when possible, ACL them at the application layer. Taking a multi-tiered security approach is always a smart idea.

Hopefully, this article has given you enough basic information to feel comfortable locking

down machines running basic services. In the next part of this article, we'll talk about applying this knowledge to a firewall, and use some of the more complex features of IP Filter to further improve things.

To read **Introduction to IP Filter Part 2**, click [here](#).

## Relevant Links

[IP Filter Website](#)

*Darren Reed*

[Privacy Statement](#)

Copyright 2006, SecurityFocus