## NAME

ippool, ippool.conf - IP Pool file format

## DESCRIPTION

The file ippool.conf is used with ippool(8) to configure address pools for use with ipnat(8) and ipf(8).

There are four different types of address pools that can be configured through ippool.conf. The various types are presented below with a brief description of how they are used:

dstlist

destination list - is a collection of IP addresses with an optional network interface name that can be used with either redirect (rdr) rules in ipnat.conf(5) or as the destination in ipf.conf(5) for policy based routing.

group-map

group maps - support the srcgrpmap and dstgrpmap call functions in ipf.conf(5) by providing a list of addresses or networks rule group numbers to start processing them with.

hash

hash tables - provide the means for performing a very efficient lookup address or network when there is expected to be only one exact match. These are best used with more static sets of addresses so they can be sized optimally.

pool

address pools - are an alternative to hash tables that can perform just as well in most circumstances. In addition, the address pools allow for heirarchical matching, so it is possible to define a subnet as matching but then exclude specific addresses from it.

### Evolving Configuration

Over time the configuration syntax used by ippool.conf(5) has evolved. Originally the syntax used was more verbose about what a particular value was being used for, for example:

```
table role = ipf type = tree number = 100
    { 1.1.1.1/32; !2.2.0.0/16; 2.2.2.0/24; ef00::5/128; };
```

This is rather long winded. The evolution of the configuration syntax has also replaced the use of

numbers with names, although numbers can still be used as can be seen here:

```
pool ipf/tree (name "100";)
        { 1.1.1.1/32; !2.2.0.0/16; 2.2.2.0/24; ef00::5/128; };
```

Both of the above examples produce the same configuration in the kernel for use with ipf.conf(5).

Newer options for use in ippool.conf(5) will only be offered in the new configuration syntax and all output using "ippool -l" will also be in the new configuration syntax.

### IPFilter devices and pools

To cater to different administration styles, ipool.conf(5) allows you to tie a pool to a specific role in IPFilter. The recognised role names are:

ipf

pools defined for role "ipf" are available for use with all rules that are found in ipf.conf(5) except for auth rules.

nat

pools defined for role "nat" are available for use with all rules that are found in ipnat.conf(5).

auth

pools defined for role "auth" are available only for use with "auth" rules that are found in ipf.conf(5)

all

pools that are defined for the "all" role are available to all types of rules, be they NAT rules in ipnat.conf(5) or firewall rules in ipf.conf(5).

## Address Pools

An address pool can be used in ipf.conf(5) and ipnat.conf(5) for matching the source or destination address of packets. They can be referred to either by name or number and can hold an arbitrary number of address patterns to match.

An address pool is considered to be a "tree type". In the older configuration style, it was necessary to have "type=tree" in ippool.conf(5). In the new style configuration, it follows the IPFilter device with

which the pool is being configured. Now it is the default if left out.

For convenience, both IPv4 and IPv6 addresses can be stored in the same address pool. It should go without saying that either type of packet can only ever match an entry in a pool that is of the same address family.

The address pool searches the list of addresses configured for the best match. The "best match" is considered to be the match that has the highest number of bits set in the mask. Thus if both 2.2.0.0/16 and 2.2.2.0/24 are present in an address pool, the addres 2.2.2.1 will match 2.2.2.0/24 and 2.2.1.1 will match 2.2.0.0/16. The reason for this is to allow exceptions to be added through the use of negative matching. In the following example, the pool contains "2.2.0.0/16" and "!2.2.2.0/24", meaning that all packets that match 2.2.0.0/16, except those that match 2.2.2.0/24, will be considered as a match for this pool.

```
table role = ipf type = tree number = 100
        { 1.1.1.1/32; 2.2.0.0/16; !2.2.2.0/24; ef00::5/128; };
```

For the sake of clarity and to aid in managing large numbers of addresses inside address pools, it is possible to specify a location to load the addresses from. To do this simply use a "file://" URL where you would specify an actual IP address.

```
pool ipf/tree (name rfc1918;) { file:///etc/ipf/rfc1918; };
```

The contents of the file might look something like this:

```
# RFC 1918 networks
10.0.0.0/8
!127.0.0.0/8
172.16.0.0/12
192.168.0.0/24
```

In this example, the inclusion of the line "!127.0.0.0/8" is, strictly speaking not correct and serves only as an example to show that negative matching is also supported in this file.

Another format that ippool(8) recognises for input from a file is that from whois servers. In the following example, output from a query to a WHOIS server for information about which networks are associated with the name "microsoft" has been saved in a file named "ms-networks". There is no need to modify the output from the whois server, so using either the whois command or dumping data directly from it over a TCP connection works perfectly file as input.

pool ipf/tree (name microsoft;) { whois file "/etc/ipf/ms-networks"; };

And to then block all packets to/from networks defined in that file, a rule like this might be used:

block in from pool/microsoft to any

Note that there are limitations on the output returned by whois servers so be aware that their output may not be 100% perfect for your goal.

**Destination Lists**

Destination lists are provided for use primarily with NAT redirect rules (rdr). Their purpose is to allow more sophisticated methods of selecting which host to send traffic to next than the simple round-robin technique that is present with with "round-robin" rules in ipnat.conf(5).

When building a list of hosts to use as a redirection list, it is necessary to list each host to be used explicitly. Expressing a collection of hosts as a range or a subnet is not supported. With each address it is also possible to specify a network interface name. The network interface name is ignored by NAT when using destination lists. The network itnerface name is currently only used with policy based routing (use of "to"/"dup-to" in ipf.conf(5)).

Unlike the other directives that can be expressed in this file, destination lists must be written using the new configuration syntax. Each destination list must have a name associated with it and a next hop selection policy. Some policies have further options. The currently available selection policies are:

round-robin

steps through the list of hosts configured with the destination list one by one

random

the next hop is chosen by random selection from the list available

src-hash

a hash is made of the source address components of the packet (address and port number) and this is used to select which next hop address is used

dst-hash

a hash is made of the destination address components of the packet (address and port number) and

this is used to select which next hop address is used

hash

a hash is made of all the address components in the packet (addresses and port numbers) and this is used to select which next hop address is used

weighted

selecting a weighted policy for destination selection needs further clarification as to what type of weighted selection will be used. The sub-options to a weighted policy are:

connection

the host that has received the least number of connections is selected to be the next hop. When all hosts have the same connection count, the last one used will be the next address selected.

The first example here shows 4 destinations that are used with a round-robin selection policy.

```
pool nat/dstlist (name servers; policy round-robin;)
    { 1.1.1.2; 1.1.1.4; 1.1.1.5; 1.1.1.9; };
```

In the following example, the destination is chosen by whichever has had the least number of connections. By placing the interface name with each address and saying "all/dstlist", the destination list can be used with both ipnat.conf(5) and ipf.conf(5).

```
pool all/dstlist (name servers; policy weighted connection;)
    { bge0:1.1.1.2; bge0:1.1.1.4; bge1:1.1.1.5; bge1:1.1.1.9; };
```

**Group maps**

Group maps are provided to allow more efficient processing of packets where there are a larger number of subnets and groups of rules for those subnets. Group maps are used with "call" rules in ipf.conf(5) that use the "srcgrpmap" and "dstgrpmap" functions.

A group map declaration must mention which group is the default group for all matching addresses to be applied to. Then inside the list of addresses and networks for the group, each one may optionally have a group number associated with it. A simple example like this, where the first two entries would map to group 2020 but 5.0.0.0/8 sends rule processing to group 2040.

```
group-map out role = ipf number = 2010 group = 2020
```

```
{ 2.2.2.2/32; 4.4.0.0/16; 5.0.0.0/8, group = 2040; };
```

An example that outlines the real purpose of group maps is below, where each one of the 12 subnets is mapped to a different group number. This might be because each subnet has its own policy and rather than write a list of twelve rules in ipf.conf(5) that match the subnet and branch off with a head statement, a single rule can be used with this group map to achieve the same result.

```
group-map ( name "2010"; in; )
  { 192.168.1.0/24, group = 10010; 192.168.2.0/24, group = 10020;
    192.168.3.0/24, group = 10030; 192.168.4.0/24, group = 10040;
    192.168.5.0/24, group = 10050; 192.168.6.0/24, group = 10060;
    192.168.7.0/24, group = 10070; 192.168.8.0/24, group = 10080;
    192.168.9.0/24, group = 10090; 192.168.10.0/24, group = 10100;
    192.168.11.0/24, group = 10110; 192.168.12.0/24, group = 10120;
  };
```

The limitation with group maps is that only the source address or the destination address can be used to map the packet to the starting group, not both, in your ipf.conf(5) file.

**Hash Tables**

The hash table is operationally similar to the address pool. It is used as a store for a collection of address to match on, saving the need to write a lengthy list of rules. As with address pools, searching will attempt to find the best match - an address specification with the largest contiguous netmask.

Hash tables are best used where the list of addresses, subnets and networks is relatively static, which is something of a contrast to the address pool that can work with either static or changing address list sizes.

Further work is still needed to have IPFilter correctly size and tune the hash table to optimise searching. The goal is to allow for small to medium sized tables to achieve close to O(1) for either a positive or negative match, in contrast to the address pool, which is O(logn).

The following two examples build the same table in the kernel, using the old configuration format (first) and the new one (second).

```
table role=all type=hash name=servers size=5
      { 1.1.1.2/32; 1.1.1.3/32; 11.23.44.66/32; };

pool all/hash (name servers; size 5;)
          { 1.1.1.2; 1.1.1.3; 11.23.44.66; };
```

**FILES**

/dev/iplookup
/etc/ippool.conf
/etc/hosts

**SEE ALSO**

ippool(8), hosts(5), ipf(5), ipf(8), ipnat(8)