NAME

ipmon, ipmon.conf - ipmon configuration file format

DESCRIPTION

The **ipmon.conf** file is optionally loaded by **ipmon** when it starts. Its primary purpose is to direct **ipmon** to do extra actions when it sees a specific log entry from the kernel.

A line in the **ipmon.conf** file is either a comment or a **match** line. Each line must have a matching segment and an action segment. These are to the left and right of the word "do", respectively. A comment line is any line that starts with a #.

NOTE: This file differs from all other IPFilter configuration files because it attempts to match every line with every log record received. It does **not** stop at the **first** match or only use the **last** match.

For the action segment, a **match** line can delivery output to one of three destinations: **file**, **email** or **command**. For example:

```
match { type = ipf; } do { save("file:///var/log/ipf-log"); };
match { type = nat; } do { syslog; };
match { type = state; } do { execute("/bin/mail root"); };
```

and is roughly described like this:

match { *match-it ,match-it, ...* } do { *action, action, ...*};

where there can be a list of matching expressions and a list of actions to perform if all of the matching expressions are matched up with by the current log entry.

The lines above would save all ipf log entries to /var/log/ipf-log, send all of the entries for NAT (ipnat related) to syslog and generate an email to root for each log entry from the state tables.

SYNTAX - MATCHING

In the above example, the matching segment was confined to matching on the type of log entry generated. The full list of fields that can be used here is:

direction <in|out>

This option is used to match on log records generated for packets going in or out.

dstip <address/mask>

This option is used to match against the destination address associated with the packet being logged. A "/mask" must be given and given in CIDR notation (/0-/32) so to specify host 192.2.2.1, 192.2.2.1/32 must be given.

dstport <portnumber>
: This option is used to match against the destination port in log entries. A number must be given, symbolic names (such as those from /etc/services) are not recognised by the parser.

every <second|# seconds|packet|# packets>
: This option is used to regulate how often an **ipmon.conf** entry is actioned in response to an otherwise matching log record from the kernel.

group <name|number>

interface <interface-name>
: This option is used to match against the network interface name associated with the action causing the logging to happen. In general this will be the network interface where the packet is seen by IPFilter.

logtag <number>
: This option is used to match against tags set by ipf rules in **ipf.conf**. These tags are set with "set-tag(log=100)" appended to filter rules.

nattag <string>
: This option is used to match against tags set by NAT rules in **ipnat.conf**.

protocol <name|number>
: This option is used to match against the IP protocol field in the packet being logged.

result <pass|block|nomatch|log>
: This option is used to match against the result of packet matching in the kernel. If a packet is logged, using a **log** rule in **ipf.conf** then it will match "log" here. The "nomatch" option is for use with matching log records generated for all packets as the default.

rule <number>
: This option is used to match against the *number* of the rule causing the record to be generated. The *number* of a rule can be observed using "ipfstat -ion".

srcip <address/mask>
: This option is used to match against the source address associated with the packet being logged. A "/mask" must be given and given in CIDR notation (/0-/32) so to specify host 192.2.2.1,

192.2.2.1/32 must be given.

srcport <portnumber>
: This option is used to match against the source port in log entries. A number must be given, symbolic names (such as those from /etc/services) are not recognised by the parser.

type <ipf|nat|state>
: The format for files accepted by ipmon is described by the following grammar: **NOTE:** At present, only IPv4 matching is available for source/destination address matching.

SYNTAX - ACTIONS

The list of actions supported is as follows:

save("file://<filename>")
: save("raw://<filename>") Write out the log record to the filename given. This file will be closed and reopened on receipt of a SIGHUP. If the *raw* target is used, binary log data, as read from the kernel, is written out rather than a text log record. The filename should be an absolute target, including the root directory. Thus, saving to /var/log/ipmon.log would be, as an example, save("file:///var/log/ipmon.log").

syslog("<facility>.<priority>")
: syslog("<facility>.") syslog(".<priority>") To log a text record via syslog, the **syslog** action word is used. The facility used by default is determined at first by the default compiled into **ipmon** (usually LOG_LOCAL0), which can be changed via the command line (-L <facility>) or in an **ipf.conf** rule using the *level* option with logging. If the facility is specified here, it takes precedence over all other settings. The same applies to the syslog priority. By default, ipmon will determine a priority for the packet, depending on whether or not it has been blocked, passed, etc. It is possible to force the complete facility/priority value for each log entry or to choose to replace only one of them.

execute("<command string>")
: The **execute** action runs the specified command each time the log entry matches and feeds the log entry, as text, to the command being executed. The command string given is executed using /bin/sh.

nothing
: Literally, do nothing. Use this if you want to be verbose in your config file about doing nothing for a particular log record.

PLUGIN ACTIONS

It is possible to configure **ipmon** to use externally supplied modules to save log entries with. These are added to **ipmon** using the *load_action* configuration line. The syntax of this line is:

```
load_action <name> <path>;
```

name
: is a short name for the action. It does not need to correspond to the name of the library file, but inside the library file, the functions **<name>destroy** , **<name>parse** and **<name>store** must be present.

path
: specifies the path in the filesystem to the shared object that contains the implementation of the new action. After the new action has been declared using *load_action* it can then be used in any *do* statement.

EXAMPLES

Some further examples are:

```
#
# log everything to syslog local4, regardless
#
match { ; } do { syslog("local4."); };
#
# keep a local copy of things packets to/from port 80
#
match { srcport = 80; } do { save("file:///var/log/web"); };
match { dstport = 80; } do { save("file:///var/log/web"); };
#
load_action local "/usr/lib/libmyaction.so";
match { dstip 127.0.0.1; } do { local("local options"); };
#
```

MATCHING

All entries of the rules present in the file are compared for matches - there is no first or last rule match.

FILES

/dev/ipl
/dev/ipf
/dev/ipnat
/dev/ipstate

/etc/ipmon.conf

SEE ALSO

ipmon(8), ipl(4)