## NAME

**blacklistd** - block and release ports on demand to avoid DoS abuse

## SYNOPSIS

**blacklistd** [**-dfrv**] [**-C** *controlprog*] [**-c** *configfile*] [**-D** *dbfile*] [**-P** *sockpathsfile*] [**-R** *rulename*] [**-s** *sockpath*] [**-t** *timeout*]

## DESCRIPTION

**blacklistd** is a daemon similar to syslogd(8) that listens to sockets at paths specified in the *sockpathsfile* for notifications from other daemons about successful or failed connection attempts. If no such file is specified, then it only listens to the socket path specified by *sockspath* or if that is not specified to */var/run/blacklistd.sock*. Each notification contains an (action, port, protocol, address, owner) tuple that identifies the remote connection and the action. This tuple is consulted against entries in *configfile* with syntax specified in blacklistd.conf(5). If an entry is matched, a state entry is created for that tuple. Each entry contains a number of tries limit and a duration.

The way **blacklistd** does configuration entry matching is by having the client side pass the file descriptor associated with the connection the client wants to blacklist as well as passing socket credentials.

The file descriptor is used to retrieve information (address and port) about the remote side with getpeername(2) and the local side with getsockname(2).

By examining the port of the local side, **blacklistd** can determine if the client program "owns" the port. By examining the optional address portion on the local side, it can match interfaces. By examining the remote address, it can match specific allow or deny rules.

Finally **blacklistd** can examine the socket credentials to match the user in the configuration file.

While this works well for TCP sockets, it cannot be relied on for unbound UDP sockets. It is also less meaningful when it comes to connections using non-privileged ports. On the other hand, if we receive a request that has a local endpoint indicating a UDP privileged port, we can presume that the client was privileged to be able to acquire that port.

Once an entry is matched **blacklistd** can perform various actions. If the action is "add" and the number of tries limit is reached, then a control script *controlprog* is invoked with arguments:

control add <rulename> <proto> <address> <mask> <port>

and should invoke a packet filter command to block the connection specified by the arguments. The *rulename* argument can be set from the command line (default blacklistd). The script could print a

numerical id to stdout as a handle for the rule that can be used later to remove that connection, but that is not required as all information to remove the rule is kept.

If the action is "rem" Then the same control script is invoked as:

control rem <rulename> <proto> <address> <mask> <port> <id>

where *id* is the number returned from the "add" action.

**blacklistd** maintains a database of known connections in *dbfile*. On startup it reads entries from that file, and updates its internal state.

**blacklistd** checks the list of active entries every *timeout* seconds (default 15) and removes entries and block rules using the control program as necessary.

The following options are available:

**-C** *controlprog*
Use *controlprog* to communicate with the packet filter, usually */usr/libexec/blacklistd-helper*. The following arguments are passed to the control program:

action The action to perform: add, rem, or flush to add, remove or flush a firewall rule.

name The rule name.

protocol The optional protocol name (can be empty): tcp, tcp6, udp, udp6.

address The IPv4 or IPv6 numeric address to be blocked or released.

mask The numeric mask to be applied to the blocked or released address

port The optional numeric port to be blocked (can be empty).

id For packet filters that support removal of rules by rule identifier, the identifier of the rule to be removed. The add command is expected to return the rule identifier string to stdout.

**-c** *configuration*
The name of the configuration file to read, usually */etc/blacklistd.conf*.

**-D** *dbfile*
The Berkeley DB file where **blacklistd** stores its state, usually */var/db/blacklistd.db*.

**-d** Normally, **blacklistd** disassociates itself from the terminal unless the **-d** flag is specified, in which case it stays in the foreground.

**-f** Truncate the state database and flush all the rules named *rulename* are deleted by invoking the control script as:

control flush <rulename>

**-P** *sockspathsfile*
A file containing a list of pathnames, one per line that **blacklistd** will create sockets to listen to. This is useful for chrooted environments.

**-R** *rulename*
Specify the default rule name for the packet filter rules, usually blacklistd.

**-r** Re-read the firewall rules from the internal database, then remove and re-add them. This helps for packet filters that do not retain state across reboots.

**-s** *sockpath*
Add *sockpath* to the list of Unix sockets **blacklistd** listens to.

**-t** *timeout*
The interval in seconds **blacklistd** polls the state file to update the rules.

**-v** Cause **blacklistd** to print diagnostic messages to stdout instead of syslogd(8).

## SIGNAL HANDLING

**blacklistd** deals with the following signals:

HUP Receipt of this signal causes **blacklistd** to re-read the configuration file.

INT, TERM & QUIT
These signals tell **blacklistd** to exit in an orderly fashion.

USR1 This signal tells **blacklistd** to increase the internal debugging level by 1.

USR2 This signal tells **blacklistd** to decrease the internal debugging level by 1.

## FILES

| | |
|---|---|
| */usr/libexec/blacklistd-helper* | Shell script invoked to interface with the packet filter. |
| */etc/blacklistd.conf* | Configuration file. |
| */var/db/blacklistd.db* | Database of current connection entries. |
| */var/run/blacklistd.sock* | Socket to receive connection notifications. |

## SEE ALSO

blacklistd.conf(5), blacklistctl(8), pfctl(8), syslogd(8)

## HISTORY

**blacklistd** first appeared in NetBSD 7. FreeBSD support for **blacklistd** was implemented in FreeBSD 11.

## AUTHORS

Christos Zoulas