NAME

natd - Network Address Translation daemon

SYNOPSIS

natd [**-unregistered_only** | **-u**] [**-log** | **-l**] [**-proxy_only**] [**-reverse**] [**-deny_incoming** | **-d**] [**-use_sockets** | **-s**]
[**-same_ports** | **-m**] [**-verbose** | **-v**] [**-dynamic**] [**-in_port** | **-i** *port*] [**-out_port** | **-o** *port*] [**-port** | **-p** *port*]
[**-alias_address** | **-a** *address*] [**-target_address** | **-t** *address*] [**-interface** | **-n** *interface*]
[**-proxy_rule** *proxyspec*] [**-redirect_port** *linkspec*] [**-redirect_proto** *linkspec*]
[**-redirect_address** *linkspec*] [**-config** | **-f** *configfile*] [**-instance** *instancename*] [**-globalport** *port*]
[**-log_denied**] [**-log_facility** *facility_name*] [**-punch_fw** *firewall_range*] [**-skinny_port** *port*]
[**-log_ipfw_denied**] [**-pid_file** | **-P** *pidfile*] [**-exit_delay** | **-P** *ms*]

DESCRIPTION

The **natd** utility provides a Network Address Translation facility for use with divert(4) sockets under FreeBSD.

(If you need NAT on a PPP link, ppp(8) provides the **-nat** option that gives most of the **natd** functionality, and uses the same libalias(3) library.)

The **natd** utility normally runs in the background as a daemon. It is passed raw IP packets as they travel into and out of the machine, and will possibly change these before re-injecting them back into the IP packet stream.

It changes all packets destined for another host so that their source IP address is that of the current machine. For each packet changed in this manner, an internal table entry is created to record this fact. The source port number is also changed to indicate the table entry applying to the packet. Packets that are received with a target IP of the current host are checked against this internal table. If an entry is found, it is used to determine the correct target IP address and port to place in the packet.

The following command line options are available:

-log | **-l** Log various aliasing statistics and information to the file */var/log/alias.log*. This file is truncated each time **natd** is started.

-deny_incoming | **-d**
Do not pass incoming packets that have no entry in the internal translation table.

If this option is not used, then such a packet will be altered using the rules in **-target_address** below, and the entry will be made in the internal translation table.

-log_denied Log denied incoming packets via syslog(3) (see also **-log_facility**).

-log_facility *facility_name*
Use specified log facility when logging information via syslog(3). Argument *facility_name* is one of the keywords specified in syslog.conf(5).

-use_sockets | **-s**
Allocate a socket(2) in order to establish an FTP data or IRC DCC send connection. This option uses more system resources, but guarantees successful connections when port numbers conflict.

-same_ports | **-m**
Try to keep the same port number when altering outgoing packets. With this option, protocols such as RPC will have a better chance of working. If it is not possible to maintain the port number, it will be silently changed as per normal.

-verbose | **-v** Do not call daemon(3) on startup. Instead, stay attached to the controlling terminal and display all packet alterations to the standard output. This option should only be used for debugging purposes.

-unregistered_only | **-u**
Only alter outgoing packets with an *unregistered* source address. According to RFC 1918, unregistered source addresses are 10.0.0.0/8, 172.16.0.0/12 and 192.168.0.0/16.

-redirect_port *proto targetIP*:*targetPORT*[-*targetPORT*] [*aliasIP*:]*aliasPORT*[-*aliasPORT*]
[*remoteIP*[:*remotePORT*[-*remotePORT*]]]
Redirect incoming connections arriving to given port(s) to another host and port(s). Argument *proto* is either *tcp* or *udp*, *targetIP* is the desired target IP address, *targetPORT* is the desired target port number or range, *aliasPORT* is the requested port number or range, and *aliasIP* is the aliasing address. Arguments *remoteIP* and *remotePORT* can be used to specify the connection more accurately if necessary. If *remotePORT* is not specified, it is assumed to be all ports.

Arguments *targetIP*, *aliasIP* and *remoteIP* can be given as IP addresses or as hostnames. The *targetPORT*, *aliasPORT* and *remotePORT* ranges need not be the same numerically, but must have the same size. When *targetPORT*, *aliasPORT* or *remotePORT* specifies a singular value (not a range), it can be given as a service name that is searched for in the services(5) database.

For example, the argument

tcp inside1:telnet 6666

means that incoming TCP packets destined for port 6666 on this machine will be sent to the telnet port on the inside1 machine.

tcp inside2:2300-2399 3300-3399

will redirect incoming connections on ports 3300-3399 to host inside2, ports 2300-2399. The mapping is 1:1 meaning port 3300 maps to 2300, 3301 maps to 2301, etc.

-redirect_proto *proto localIP* [*publicIP* [*remoteIP*]]

Redirect incoming IP packets of protocol *proto* (see protocols(5)) destined for *publicIP* address to a *localIP* address and vice versa.

If *publicIP* is not specified, then the default aliasing address is used. If *remoteIP* is specified, then only packets coming from/to *remoteIP* will match the rule.

-redirect_address *localIP publicIP*

Redirect traffic for public IP address to a machine on the local network. This function is known as *static NAT*. Normally static NAT is useful if your ISP has allocated a small block of IP addresses to you, but it can even be used in the case of single address:

redirect_address 10.0.0.8 0.0.0.0

The above command would redirect all incoming traffic to machine 10.0.0.8.

If several address aliases specify the same public address as follows

redirect_address 192.168.0.2 public_addr
redirect_address 192.168.0.3 public_addr
redirect_address 192.168.0.4 public_addr

the incoming traffic will be directed to the last translated local address (192.168.0.4), but outgoing traffic from the first two addresses will still be aliased to appear from the specified *public_addr*.

-redirect_port *proto targetIP*:*targetPORT*[,*targetIP*:*targetPORT*[,...]] [*aliasIP*:]*aliasPORT* [*remoteIP*[:*remotePORT*]]

-redirect_address *localIP*[,*localIP*[,...]] *publicIP*

These forms of **-redirect_port** and **-redirect_address** are used to transparently offload network load on a single server and distribute the load across a pool of servers. This function is known as *LSNAT* (RFC 2391). For example, the argument

tcp www1:http,www2:http,www3:http www:http

means that incoming HTTP requests for host www will be transparently redirected to one of the www1, www2 or www3, where a host is selected simply on a round-robin basis, without regard to load on the net.

-dynamic If the **-n** or **-interface** option is used, **natd** will monitor the routing socket for alterations to the *interface* passed. If the interface's IP address is changed, **natd** will dynamically alter its concept of the alias address.

-in_port | **-i** *port*
Read from and write to divert(4) port *port*, treating all packets as "incoming".

-out_port | **-o** *port*
Read from and write to divert(4) port *port*, treating all packets as "outgoing".

-port | **-p** *port*
Read from and write to divert(4) port *port*, distinguishing packets as "incoming" or "outgoing" using the rules specified in divert(4). If *port* is not numeric, it is searched for in the services(5) database. If this option is not specified, the divert port named *natd* will be used as a default.

-alias_address | **-a** *address*
Use *address* as the aliasing address. Either this or the **-interface** option must be used (but not both), if the **-proxy_only** option is not specified. The specified address is usually the address assigned to the "public" network interface.

All data passing *out* will be rewritten with a source address equal to *address*. All data coming *in* will be checked to see if it matches any already-aliased outgoing connection. If it does, the packet is altered accordingly. If not, all **-redirect_port**, **-redirect_proto** and **-redirect_address** assignments are checked and actioned. If no other action can be made and if **-deny_incoming** is not specified, the packet is delivered to the local machine using the rules specified in **-target_address** option below.

-t | **-target_address** *address*
Set the target address. When an incoming packet not associated with any pre-existing link

arrives at the host machine, it will be sent to the specified *address*.

The target address may be set to *255.255.255.255*, in which case all new incoming packets go to the alias address set by **-alias_address** or **-interface**.

If this option is not used, or called with the argument *0.0.0.0*, then all new incoming packets go to the address specified in the packet. This allows external machines to talk directly to internal machines if they can route packets to the machine in question.

-interface | **-n** *interface*

Use *interface* to determine the aliasing address. If there is a possibility that the IP address associated with *interface* may change, the **-dynamic** option should also be used. If this option is not specified, the **-alias_address** option must be used.

The specified *interface* is usually the "public" (or "external") network interface.

-config | **-f** *file*

Read configuration from *file*. A *file* should contain a list of options, one per line, in the same form as the long form of the above command line options. For example, the line

alias_address 158.152.17.1

would specify an alias address of 158.152.17.1. Options that do not take an argument are specified with an argument of *yes* or *no* in the configuration file. For example, the line

log yes

is synonymous with **-log**.

Options can be divided to several sections. Each section applies to own **natd** instance. This ability allows the configuration of one **natd** process for several NAT instances. The first instance that always exists is a "default" instance. Each another instance should begin with

instance *instance_name*

At the next should be placed a configuration option. Example:

```
# default instance
port 8668
```

```
alias_address 158.152.17.1

# second instance
instance dsl1
port 8888
alias_address 192.168.0.1
```

Trailing spaces and empty lines are ignored. A '#' sign will mark the rest of the line as a comment.

-instance *instancename*
This option switches command line options processing to configure instance *instancename* (creating it if necessary) till the next **-instance** option or end of command line. It is easier to set up multiple instances in the configuration file specified with the **-config** option rather than on a command line.

-globalport *port*
Read from and write to divert(4) port *port*, treating all packets as "outgoing". This option is intended to be used with multiple instances: packets received on this port are checked against internal translation tables of every configured instance. If an entry is found, packet is aliased according to that entry. If no entry was found in any of the instances, packet is passed unchanged, and no new entry will be created. See the section *MULTIPLE INSTANCES* for more details.

-reverse This option makes **natd** reverse the way it handles "incoming" and "outgoing" packets, allowing it to operate on the "internal" network interface rather than the "external" one.

This can be useful in some transparent proxying situations when outgoing traffic is redirected to the local machine and **natd** is running on the internal interface (it usually runs on the external interface).

-proxy_only Force **natd** to perform transparent proxying only. Normal address translation is not performed.

-proxy_rule [*type encode_ip_hdr | encode_tcp_stream*] *port xxxx server a.b.c.d:yyyy*
Enable transparent proxying. Outgoing TCP packets with the given port going through this host to any other host are redirected to the given server and port. Optionally, the original target address can be encoded into the packet. Use *encode_ip_hdr* to put this information into the IP option field or *encode_tcp_stream* to inject the data into the beginning of the TCP stream.

-punch_fw *basenumber*:*count*

This option directs **natd** to "punch holes" in an ipfirewall(4) based firewall for FTP/IRC DCC connections. This is done dynamically by installing temporary firewall rules which allow a particular connection (and only that connection) to go through the firewall. The rules are removed once the corresponding connection terminates.

A maximum of *count* rules starting from the rule number *basenumber* will be used for punching firewall holes. The range will be cleared for all rules on startup. This option has no effect when the kernel is in security level 3, see init(8) for more information.

-skinny_port *port*

This option allows you to specify the TCP port used for the Skinny Station protocol. Skinny is used by Cisco IP phones to communicate with Cisco Call Managers to set up voice over IP calls. By default, Skinny aliasing is not performed. The typical port value for Skinny is 2000.

-log_ipfw_denied

Log when a packet cannot be re-injected because an ipfw(8) rule blocks it. This is the default with **-verbose**.

-pid_file | **-P** *file*

Specify an alternate file in which to store the process ID. The default is */var/run/natd.pid*.

-exit_delay *ms*

Specify delay in ms before daemon exit after signal. The default is *10000*.

RUNNING NATD

The following steps are necessary before attempting to run **natd**:

1. Build a custom kernel with the following options:

   ```
   options IPFIREWALL
   options IPDIVERT
   ```

   Refer to the handbook for detailed instructions on building a custom kernel.

2. Ensure that your machine is acting as a gateway. This can be done by specifying the line

   ```
   gateway_enable=YES
   ```

in the */etc/rc.conf* file or using the command

```
sysctl net.inet.ip.forwarding=1
```

3. If you use the **-interface** option, make sure that your interface is already configured. If, for example, you wish to specify 'tun0' as your *interface*, and you are using ppp(8) on that interface, you must make sure that you start **ppp** prior to starting **natd**.

Running **natd** is fairly straight forward. The line

```
natd -interface ed0
```

should suffice in most cases (substituting the correct interface name). Please check rc.conf(5) on how to configure it to be started automatically during boot. Once **natd** is running, you must ensure that traffic is diverted to **natd**:

1. You will need to adjust the */etc/rc.firewall* script to taste. If you are not interested in having a firewall, the following lines will do:

   ```
   /sbin/ipfw -f flush
   /sbin/ipfw add divert natd all from any to any via ed0
   /sbin/ipfw add pass all from any to any
   ```

   The second line depends on your interface (change 'ed0' as appropriate).

   You should be aware of the fact that, with these firewall settings, everyone on your local network can fake his source-address using your host as gateway. If there are other hosts on your local network, you are strongly encouraged to create firewall rules that only allow traffic to and from trusted hosts.

   If you specify real firewall rules, it is best to specify line 2 at the start of the script so that **natd** sees all packets before they are dropped by the firewall.

   After translation by **natd**, packets re-enter the firewall at the rule number following the rule number that caused the diversion (not the next rule if there are several at the same number).

2. Enable your firewall by setting

   ```
   firewall_enable=YES
   ```

in */etc/rc.conf*. This tells the system startup scripts to run the */etc/rc.firewall* script. If you do not wish to reboot now, just run this by hand from the console. NEVER run this from a remote session unless you put it into the background. If you do, you will lock yourself out after the flush takes place, and execution of */etc/rc.firewall* will stop at this point - blocking all accesses permanently. Running the script in the background should be enough to prevent this disaster.

MULTIPLE INSTANCES

It is not so uncommon to have a need of aliasing to several external IP addresses. While this traditionally was achieved by running several **natd** processes with independent configurations, **natd** can have multiple aliasing instances in a single process, also allowing them to be not so independent of each other. For example, let us see a common task of load balancing two channels to different providers on a machine with two external interfaces 'sis0' (with IP 1.2.3.4) and 'sis2' (with IP 2.3.4.5):

```
        net 1.2.3.0/24
1.2.3.1 ------------------ sis0
(router)            (1.2.3.4)
                                net 10.0.0.0/24
                           sis1 ------------------- 10.0.0.2
                         (10.0.0.1)
        net 2.3.4.0/24
2.3.4.1 ------------------ sis2
(router)            (2.3.4.5)
```

Default route is out via 'sis0'.

Interior machine (10.0.0.2) is accessible on TCP port 122 through both exterior IPs, and outgoing connections choose a path randomly between 'sis0' and 'sis2'.

The way this works is that *natd.conf* builds two instances of the aliasing engine.

In addition to these instances' private divert(4) sockets, a third socket called the "globalport" is created; packets sent to **natd** via this one will be matched against all instances and translated if an existing entry is found, and unchanged if no entry is found. The following lines are placed into */etc/natd.conf*:

```
log
deny_incoming
verbose

instance default
interface sis0
```

```
port 1000
redirect_port tcp 10.0.0.2:122 122

instance sis2
interface sis2
port 2000
redirect_port tcp 10.0.0.2:122 122

globalport 3000
```

And the following ipfw(8) rules are used:

```
ipfw -f flush

ipfw add    allow ip from any to any via sis1

ipfw add    skipto 1000 ip from any to any in via sis0
ipfw add    skipto 2000 ip from any to any out via sis0
ipfw add    skipto 3000 ip from any to any in via sis2
ipfw add    skipto 4000 ip from any to any out via sis2

ipfw add 1000 count ip from any to any

ipfw add    divert 1000 ip from any to any
ipfw add    allow ip from any to any

ipfw add 2000 count ip from any to any

ipfw add    divert 3000 ip from any to any

ipfw add    allow ip from 1.2.3.4 to any
ipfw add    skipto 5000 ip from 2.3.4.5 to any

ipfw add    prob .5 skipto 4000 ip from any to any

ipfw add    divert 1000 ip from any to any
ipfw add    allow ip from any to any

ipfw add 3000 count ip from any to any
```

```
ipfw add      divert 2000 ip from any to any
ipfw add      allow ip from any to any

ipfw add 4000 count ip from any to any

ipfw add      divert 2000 ip from any to any

ipfw add 5000 fwd 2.3.4.1 ip from 2.3.4.5 to not 2.3.4.0/24
ipfw add      allow ip from any to any
```

Here the packet from internal network to Internet goes out via 'sis0' (rule number 2000) and gets caught by the **globalport** socket (3000). After that, either a match is found in a translation table of one of the two instances, or the packet is passed to one of the two other divert(4) ports (1000 or 2000), with equal probability. This ensures that load balancing is done on a per-flow basis (i.e., packets from a single TCP connection always flow through the same interface). Translated packets with source IP of a non-default interface ('sis2') are forwarded to the appropriate router on that interface.

SEE ALSO

libalias(3), divert(4), protocols(5), rc.conf(5), services(5), syslog.conf(5), init(8), ipfw(8), ppp(8)

HISTORY

The **natd** utility appeared in FreeBSD 3.0.

AUTHORS

This program is the result of the efforts of many people at different times:

Archie Cobbs <*archie@FreeBSD.org*> (divert sockets)
Charles Mott <*cm@linktel.net*> (packet aliasing)
Eivind Eklund <*perhaps@yes.no*> (IRC support & misc additions)
Ari Suutari <*suutari@iki.fi*> (natd)
Dru Nelson <*dnelson@redwoodsoft.com*> (early PPTP support)
Brian Somers <*brian@awfulhak.org*> (glue)
Ruslan Ermilov <*ru@FreeBSD.org*> (natd, packet aliasing, glue)
Poul-Henning Kamp <*phk@FreeBSD.org*> (multiple instances)