How Do I... or a User's Quick Guide

How Do I... or a User's Quick Guide

Table of Contents

How do I: Allow access to a service on my machine?

Change directory into */etc/nftfw/incoming.d* (maybe */usr/local/etc/nftfw/incoming.d*).

You'll see something like

```
$ ls
05-ping         10-https         30-imaps  50-smtps
06-ftp-helper   20-ftp           40-pop3   50-submission
07-ssh          21-ftp-passive   40-pop3s  60-sieve
10-http         30-imap          50-smtp   99-drop
```

Some of the filenames in the directory access 'rules' in the *rule.d* directory. There's more information on the manual page in *nftfw_files* in man section 5.

Each file makes a rule for the firewall, and starts with a two digit number that supplies the ordering of the rules. A new entry needs to be adding before the *99-reject* rule. So let's pick '70' for that.

The rest of the filename can be a port number or the service name, but that must be in */etc/services*. Let's say we want to add access to our name server, and that's on port 53, called *domain* in */etc/services*. We can either add *70-domain* or *70-53*. The easiest way to do this is to use the *touch* command (you may have to use *sudo*):

```
$ touch 70-domain
```

because the file is empty, connections from any IP address may access this service.

The command:

```
$ sudo netstat -pat
...
```

lists the services currently being used on your machine, along with the process that is using them. Some of these will be used internally, and you may not wish to make them globally available.

How do I: Deny access to a service?

By default, the incoming firewall will reject advances from IP addresses. To permit access, you must create a rule as above. You can remove the files in this directory for services you don't need. For example, If you don't need to use the POP service, you can simply delete *40_pop3* and *40_pop3s* in the incoming directory. You should delete any file that names a service you don't want to allow people to have access to.

How do I: Allow access to a service from a restricted list of machines?

If you want to only allow access to say *ssh* to a known set of IP addresses, then you can add those IP addresses into the *07-ssh* file, one per line. Only the addresses found in the file can access the_ssh_ service. You can add the domain name of system to the file, and that will include both their IPv4 and IPv6 addresses, if they have both. It's probably a good idea to run a caching domain name server on your machine if you use names in the files.

How do I: Give full access my server to known set of machines?

First you need to find the IP addresses of the machines, the easiest way is to use the *host* command:

```
$ host notarealmachine.co.uk
notarealmachine.co.uk has address 203.0.113.134
notarealmachine.co.uk has IPv6 address 2001:db8::a676:7186
```

Armed with the IP addresses, create a file in *whitelist.d* using the address as the name.

```
$ touch 203.0.113.134
```

If you want to allow all the addresses from the network, allowing 1-254 in the last section of the address, you can add a 'CIDR' mask of 24 bits to match the first three sections of the address. CIDR addresses are usually written with '/', in this case 203.0.113.0/24, but we cannot use / in a filename, so replace it by the vertical bar symbol:

```
$ rm 203.0.113.134
$ touch '203.0.113.0|24'
```

If you put the full address from the *host* command into the directory, *nftfw* will 'normalise' the address to replace the 134 by zero.

For IPv6 addresses we always match the first 64 bits of the address. IPv6 addresses are abbreviated by writing '::' for any sequences of zeros in the address, so to allow their IPv6 address you can write:

```
$ touch '2001:db8::|64'
```

You may find some files ending with *.auto* in the directory, the whitelist scanner has installed these when it's found that a user has logged in from the address. The scanner will look after these, and will expire them automatically after 90 days.

How do I: Restrict the services accessed by a whitelisted address?

The files in the *whitelist.d* directory are empty to allow access to all services, but can contain a list of port numbers, one per line, restricting access from the named address to only those ports. For example, restricting access to *ssh* is done by:

```
$ echo 22 >> '203.0.113.0|24'
```

How do I: Block access to abusive sites?

The *blacklist.d* directory uses the same convention for files used for the whitelist. Simply create a file named for the IP address in the directory.

The blacklist scanner will automatically create files in the directory ending in *.auto* when it finds sites that are misbehaving. The scanner uses files in the*patterns .d* directory to find log files to scan, and how to interpret lines in the log files as bad.

How do I: Block access to countries?

The *blacknets.d* directory can contain a set of files each with a list of IP addresses, one to a line, expressed in CIDR notation. To block a country, you'll need the list of all the networks that the country uses and these are available from several places on the web, see Getting CIDR Lists for how to install them.

How do I: See what the firewall is doing?

The *nft* command prints the contents of the firewall with the command:

```
$ sudo nft list ruleset
```

You can print just the IPv4 and IPv6 sets separately with

```
$ sudo nft list ruleset ip
$ sudo nft list ruleset ip6
```

This is too much typing, so I alias these commands in my shell startup files:

```
alias nfl='sudo nft list ruleset ip|less'
alias nfl6='sudo nft list ruleset ip6|less'
```

You'll see that most rules have counters, so you can see what has happened in the past, what's busy and what's in use.

How do I: Check what the blacklist scanner is doing?

The blacklist scanner uses files in the *patterns.d* directory. Each file here supplies a file (or files) processed by the scanner, a port number (or a comma-separated list) used for blocking and a set of regular expressions that match the lines in the log files indicating bad behaviour.

You can find out what's blocked on your system by using:

```
$ nftfwls
```

This will print a table of the IP addresses the scanner has found, the number of matches, the number of 'incidents' - each incident is a separate run of the scanner, when it happened first, the last time a match happened, and the name of the pattern that triggered the event.

If the command says nothing, then possibly the scanner hasn't detected enough events for the match count to exceed the threshold where it creates blacklist file. The**-a** flag to the program prints all the entries in its database.

How do I: Find the country sending me blacklisted packets?

If you install the geoip2 country database on your system, and its python interface, then *nftfwls* will show the country of origin when it displays its output. Access to the geoip2 databases is free, but MaxMind who produce it ask you to sign up. See the document

Installing Geolocation for installation and signup information.

The *nftfwedit* command also allows you to ask questions about any IP address, including the country of origin and whether the IP address is in selected DNS blacklist sites.

How do I: Remove a good IP address from the blacklist?

If an IP address has found its way into the blacklist in error, then you can delete it using

```
$ sudo nftfwedit -d IP.ADD.RE.SS
```

this will remove the address from the blacklist database and also delete the entry in the *blacklist.d* directory. If you just remove the address from the directory, *nftfw* will reinstate it because it's in the database.

How do I: Add a new reason for blacklisting?

If you've spotted a line in a logfile that you want to use to blacklist a site, then first see if the logfile is already scanned by the system and just add a new regular expression to the file. You can create a new pattern file if needed, calling it anything, but it must end in *.pattern*. Do beware that some log entries can appear in several log files.

The regular expression doesn't need to match the whole line, you need to identify where the IP address is in the line and use the string `__IP__` (that's two underscores at each end) to pick it out. There should be enough information on the regular expression to make it only select one line. Look in the pattern files for examples. There's also a section in the User's Guide.

How do I: Debug my new blacklisting expression?

You can use the *nftfw* command to see if your expression works. Create a new pattern file and set

```
ports = test
```

and add your regular expression. Now say:

```
$ sudo nftfw -x -p PATTERN blacklist
```

and this will print a table showing the number of matches that the expression has detected. The PATTERN is the name of your file but without the *.pattern* appended. The *-x* option makes the program print a table, and also starts scanning the file from the beginning and doesn't record where it found the end of the file, so using this command will not interfere with normal processing.

The blacklist scanner normally ignores pattern files with *ports=test*, so it's safe to leave these files in place.

How do I: Change the settings for *nftfw*?

The file */etc/nftfw/config.ini* is a readable configuration file that contains all the settings that can be changed. As distributed all the values are commented out, each line starts with a semi-colon. There are many comments in the file explaining what each setting does.

See the manual page *nftfw-config* for a description.

How to I: Add my own tables and rules to *nftfw*?

The file */etc/nftfw/nftfw_init.nft* contains the template *nftables* framework for *nftfw*. Add new rules by editing the file. You can find an example of a template used for handling a gateway machine with WAN and LAN interfaces in */etc/nftfw/etc_nftfw/nftfw_router_example*. Rules for a router adds a *nat* table and uses the *forward* table.

nftfw_init.nft uses *nft*'s readable file format. When deciding what to add or change, the best strategy is to add your new rules to the system using the *nft* command line interface

to check that they work and use:

```
$ sudo nft list ruleset
```

to see how *nft* 'sees' the rules. Rules expressed on the command line can contain syntax that *nft* thinks is unnessary, and can also use some assumptions about defaults that *nft* will add to the compiled rules. Extract any changes from the *nft* output and edit *nftfw_init.nft*.

How do I: Get more information on *nftfw*?

The User's Guide to nftfw documents the system and there are also UNIX manual pages.

Acknowledgement

All of this is made possible by shamelessly borrowing ideas from Patrick Cherry who created the Symbiosis hosting package for Bytemark of which the firewall system is part.