Installing nftfw manually

Installing nftfw manually

Prerequisites

This document assumes that you are installing on Debian Buster (and possibly later Debian systems). The manual installation places files under /usr/local and not /. However, all the documentation (apart from the two documents on manual instllation) now assumes that the system is installed under the root of the file system, so some mental translation might be needed.

You will need root access to the machine to install this package.

This system is provides a firewall, and it would not be sensible to implement the installation of a fully installed and working system that might contain rules that would block you from your machine. After installation, a small amount configuration is needed to place the system into action.

nftables

```
$ sudo apt install nftables
```

The standard version of *nftables* at the time of writing is: 0.9.0-2. Buster backports (sadly not available on Raspberry Pi OS, previously known as Raspian) has a more recent version - 0.9.3-2~bpo10+1, and it's a good idea to upgrade to that. Look in */etc/apt/sources.list*, and if necessary append:

```
# backports
deb <YOUR SOURCE> buster-backports main contrib non-free
```

then

```
$ sudo apt update
...
$ sudo apt upgrade
...
$ sudo apt -t buster-backports install nftables
```

will upgrade your system to the most recent *nftables* release.

The *install* action will tell you that a library is not needed before it installs things. There is a *systemctl* entry for *nftables*, that will probably be disabled now. Check with:

```
$ sudo systemctl status nftables
```

If you are not running *iptables* or *nftables*, it's suggested that *nftables* service stays off until later in the installation process.

I've got a live iptables installation

Well, what you need to do depends on what's running in your system. Buster comes ready installed with two flavours of *iptables*: the legacy version and a bridging version using the *iptables* API, which maps and loads the *iptables* formatted rules into *nftables*. You can tell which one you are using by:

```
$ sudo iptables -V
iptables v1.8.2 (nf_tables)
```

If your version output looks like that, then you are OK and can just skip over what follows to **Python**.

If the words in brackets says *legacy* then you need to swap to the *nf_tables* version. Here's what you do:

FIrst, save the current *iptables* settings, you'll need to put the settings into the kernel's *nftables* system when you switch.

```
$ sudo iptables-save > ipsaved
$ sudo ip6tables-save > ip6saved
```

Now change iptables and ip6tables to use the *nftables* versions:

```
$ sudo update-alternatives --config iptables
# select selection 0, /usr/sbin/iptables-nft, auto mode
$ sudo update-alternatives --config ip6tables
# select selection 0, /usr/sbin/iptables-nft, auto mode
```

Each gives you a menu of options: select the *nftables* compatible version. I used option 0 - auto. Check your *iptables* command is now at the correct version using *-V*. Finally, reload the *iptables* data you saved earlier.

```
$ sudo iptables-restore < ipsaved
$ sudo ip6tables-restore < ip6saved
```

You now have two sets of rules loaded into the kernel, the new ones using *nftables* and the previous *iptables* set. The kernel will use the new rules but will complain. To remove the old rules:

```
$ sudo iptables-legacy -F
$ sudo ip6tables-legacy -F
```

All done, and it's painless. You have a system that works for both *iptables* commands and *nftables nft* command.

Python

nftfw is coded in Python 3 and the standard Python version on Buster is 3.7. The *nftfw* package was developed in part using Python 3.6, and so the *nftfw* package may run using that version.

We will use *pip3* to install the *nftfw* package.

```
$ sudo apt install python3-pip python3-setuptools python3-wheel
```

You may find that `python3-setuptools` and `python3-wheel` are already installed.

The *nftfwls* command uses Python's *prettytable*, which may not be installed:

```
 $ sudo apt install python3-prettytable
```

nftfw Installation

You now need the *nftfw* distribution. I put mine into*/usr/local/src*, b ut it can be your home directory.

```
$ sudo apt install git
...
$ cd /usr/local/src
$ sudo git clone https://github.com/pcollinson/nftfw
```

which will create a directory called *nftfw*.

Change into the *nftfw* directory and use *pip3* to install the package. *pip3* will allow you to uninstall the package at a later date, if you wish.

```
$ sudo pip3 install .
...
Successfully installed nftfw-<version>
```

pip3 may complain about access to files because it's being run as the superuser, it's safe to ignore that warning. To uninstall, `sudo pip3 uninstall nftfw`.

pip3 installs four commands: *nftfw*, *nftfwls*, *nftfwedit* and *nftfwadm* in */usr/local/bin*. Since these are system commands, they ought to be in */usr/local/sbin*, but the Python installation system doesn't allow that.

Take a moment to see if the installation worked by asking *nftfw* for help:

```
$ nftfw -h
...
```

The next step is to install the basic control files in */usr/local/etc/nftfw*, the working directories in */usr/local/var/lib/nftfw*, and the manual pages in */usr/local/share/man*.

The *Install.sh* script is interactive, but can be run using a configuration file that provides sensible defaults. You may want to install the firewall control directories and files (*/usr/local/etc/nftfw*) owned by a non-root user to allow for easy alteration without invoking *sudo*. I use my login on my local machine, you may want to use 'admin' for Symbiosis or 'sympl' for Sympl. *nftfw* will create files in */usr/local/etc/nftfw* based on the o wnership of that directory. Copy *Autoinstall.default* to *Autoinstall.conf*, edit the AUTO_USER to the user you want to use, and then run the script *Install.sh* script. *Autoinstall.conf* will be ignored by *git*, so this file can be used for later automated runs.

```
$ sudo sh Install.sh
...
```

The *Install.sh* script will copy files from the distribution into their correct places. It's safe to run the script again, it will not replace the contents of any directory ending with *.d*, or the two control files in */usr/local/etc/nftfw*. The script uses the standard system*install* program to do its work.

The *Install.sh* script can also be run interactively, it will offer explanations and ask questions allowing you to control the installation phases. For a default installation, use these answers:

- *Install under /usr/local?* yes
- *See the files installed?* your choice
- *Install?* yes
- *User to replace root?* 'admin' for Symbiosis, 'sympl' for Sympl, 'return' for root on other systems
- *Install Manual pages?* yes

In */usr/local/etc/nftfw*, you will find two files: *config.ini* and *nftfw_init.nft*. *config.ini* provides configuration information overriding coded-in settings in the scripts. All entries in the distributed *config.ini* are commented out using a semi-colon at the start of the line. *nftfw_init.nft* is the framework template file for the firewall. It's copied into the build system whenever a *nftfw* creates a firewall. Also, you'll find the *etc_nftfw* directory holding all the original settings for the files. The intention is to provide a place for later updates to supply new and fixed default files.

Install.sh also creates the necessary directories into */usr/local/var/lib/nftfw*.

The final stage of the installation is to copy manual pages into */usr/local/share/man*. There are six pages:

- *nftfw(1)* - manage the Nftfw firewall generator. Describes the main command that creates and manages firewall tables.
- *nftfwls(1)* - list the sqlite3 database used for storing IP addresses that have shown themselves to be candidates for blocking.
- *nftfwedit(1)* - provides a command line interface to inspect IP addresses (both in and not in the blacklist database), and tools to add and delete IP addresses in the

database, optionally adding them to the active blacklist.

- *nftfwadm(1)* - provides some tools that may be useful when installing the system.
- *nftfw-config(5)* - describes the contents of the ini-style config file tailoring settings in *nftfw*.
- *nftfw-files(5)* - the format, names and contents of the files used to control the system.

The *man* command may need ‘5’ in the command line to display the section 5 manual pages. Incidentally, the distribution also has these manual pages in HTML format (see *man/index.md*).

You are now ready to create a firewall to suit your needs.

Paying attention to *config.ini*

The values in the distributed_config.ini_ are all commented out by starting the line with a semi-colon. The value shown is the default value, and won’t need changing if the default suits your system.

The *nftables systemd* service uses a control file (*/etc/nftables.conf*) to reload its tables when the system is rebooted. It’s the job of *nftfw* to create this file, but for safety, the distributed system has to be configured to install it. By default, the file is written in */usr/local/etc/nftfw*. Before going live, you will need to edit *nftables_conf* setting in *config.ini* to create the file in */etc/nftables.conf*.

Logging

All *nftfw* programs will write logging message to syslog, and also to the terminal. Error messages are output using logging level ERROR, and information messages using INFO. The scripts turn off direct printing output unless they are talking to a terminal. The scripts all have a *-q* (*quiet*) flag suppressing terminal output.

The logging level displayed by the scripts is set by a value in the configuration file *config.ini*, and this defaults to INFO, so all messages are displayed. The scripts have a *-v* (*verbose*) flag that raises the output level to INFO, showing the information messages. To see less information, you may wish to reduce the level to ERROR.

```
;loglevel = INFO
```

to

```
loglevel = ERROR
```

Using Symbiosis/Sympl control files

nftfw uses the same format for the control files found in the Symbiosis/Sympl firewall directory. There are some differences in the firewall control files that are needed. The distribution offers the same functionality that you would get on a newly installed Sympl/Symbiosis system.

There are some changes in rules needed for firewalls, some of the functionality is now in the *nftables_init.nft* file, meaning that some entries are not needed. The scanning pattern files in *patterns.d* have been enhanced to offer new functionality. Look in the manual page for more details man/nftfw-files.5,

There is a python script in the distribution that will migrate Symbiosis/Sympl settings into the *nftfw* control files. See*import_tool/*. There’ s a README file, and running the script *import_to_nftfw.py* with no arguments provides help on how to use it. It’s designed to do nothing unless you ask it to.

Migrating to nftfw

The distributed *nftfw* command builds the firewall, installs it in the kernel and saves a copy of what it has created in */usr/local/etc/nftables.conf*. The Debian*nftab les* service expects to reload *nftables* from the file in */etc* on a reboot.

If your system has a running firewall that's not *nftfw* then you probably don't want to be too hasty about installing the system. I have no expectation that things *will* go wrong, but the ability to go back is important. The steps below provides ways to revert, and explain how to do the installation safely.

First, make sure you open another window to the system and login to the machine with it, this will keep the connection open and you can recover if something goes wrong.

Let's look at where you might be now:

- The system runs a Symbiosis or Sympl system using *iptables*. In this case, the running firewall can be re-installed using the *{symbiosis|sympl}-firewall* command. You should move */etc/cron.d/symbiosis-firewall* and */etc/incron.d/symbiosis-firewall* (or equivalent *sympl* files if present) to a safe place to stop their firewalls from running for now.
- *Or*: The system runs a different firewall system using *iptables*, you can usually use *iptables_save* to save the settings somewhere, and *iptables_restore* to put the old rules back.
- *Or*: Your system already runs an *nftables* based firewall, and you want to try *nftfw* out. In this case, do make sure that *nftfw* won't overwrite your */etc/nftables.conf* file.
- *Or*: There may be other options.

If you have a live *iptables* system, check the information about *iptables* versions and how to set things up at the top of this document.

Having set up the directories in */usr/local/etc/nftfw* and the working directories in */usr/local/var/lib/nftfw*, you can run:

```
$ sudo nftfw -x -v load
```

The *-x* flag makes *nftfw* compile the files used in setting up the firewall, tests their syntax using the *nft* command, but doesn't install them. The *-v* flag prints information messages, and is a good idea if you've not altered logging levels in *config.ini*.

Assuming the tests show that the installation is OK, then you will have a working firewall to install. At this point, if you want to know what will happen, you can change to */usr/local/var/lib/nftfw/test.d* and look at the files. The starting point for the firewall is *nftfw_init.nft*, which then includes further files. We cannot get a fully compiled set without loading the actual kernel tables, and you could do that if you are feeling confident, but not until you've taken the next step.

Installing

If you are on a Symbiosis or Sympl system, I recommend at this point that you move */etc/cron.d/{symbiosis|sympl}-firewall* to a safe place. You don't want this firing when you are doing the next sequence of commands, so remove it from *cron*. On a Symbiosis system, you also need to move */etc/incron.d/symbiosis-firewall* to a safe place, you don't want this starting Symbiosis if files in */etc/symbiosis/firewall.d* change.

If you have a running *nftables* or *iptables* installation, now is the time to run:

```
$ sudo nftfwadm save
```

this saves your *nftables* settings into *nftfw*'s backup system, so if the install does fail, it will revert to what you had there before you started meddling. If you are using an *iptables* based system, fear not, the *nft* command doing the work will save the *iptables* settings in *nftables* format, assuming you have the *nftables* version of *iptables* installed and have working tables.

If all is well, you can try loading the rules made by *nftfw*.

```
$ sudo nftfw -f -v load
```

The *-f* forces a full load and you'll need it if you've run the test. The point of saving the original settings will be apparent if something bad happens on this load, *nftfw* will reload the kernel state from your saved settings.

If there was no error, you can now see all the tables using:

```
$ sudo nft list ruleset
```

If you need to revert, then now's the time.

```
$ sudo nftfwadm restore
```

replaces the newly installed rules with the tables you saved. The restore command will delete the backup file you stored, so you will need to run *save* again if you plan to change things and try again.

If you are happy with *nftfw*, then you are nearly all set. If you've used the *save* command to *nftfwadm*, then you need to remove the backup version of your old settings from the system.

```
$ sudo nftfwadm clean
```

simple deletes the backup file. Don't leave it installed, *nftfw* makes a backup file on every run so it can backtrack. However, it doesn't create a new file if it exists, and you need to remove your original settings from the system. If you don't remove the file and there's a problem some time in the future, you may find yourself wondering why the firewall in the system is an ancient version. Just where did *that* come from?

Final steps

To tidy up, check that the setting of *nftables_conf* in *nftfw*'s config file */usr/local/etc/nftfw/config.ini* reads:

```
#  Location of system nftables.conf
#  Usually /etc/nftables.conf
nftables_conf = /etc/nftables.conf
```

and rerun

```
$ sudo nftfw -f load
```

to make sure that the new rules are written into */etc/nftables.conf*.

Tell *systemctl* to enable and start its *nftables* service, if that's needed.

```
$ sudo systemctl enable nftables
$ sudo systemctl start nftables
$ sudo systemctl status nftables
```

On a boot of a Symbiosis or Sympl system, the firewall starts at network up time and closes at network down time. Change into*/etc/netw ork* and delete *if-up.d/{symbiosis*|*sympl}-firewall* and *if-down.d/{symbiosis*|*sympl}-firewall}*. The file is a symbolic link to the firewall script. This turns out to be an important step, rebooting without having this done results in a bad combination of two firewalls, because the *nftables* settings are loaded before the Symbiosis/Sympl ones.

Setting up *cron*

Like Symbiosis/Sympl, *nftfw* uses *cron* to drive regular polls by the firewall loader, and blacklist and whitelist scanners. You'll find a sample *cron* control file in the *cronfiles* directory in the distribution. Hopefully, by now, you've moved the Symbiosis or Sympl versions to somewhere where they are not active.

Check *cron-nftfw* in the *cronfiles* directory. The file contains*/usr/local/* as a pathname , if your system is installed from root, you'll need to edit the file to point to the correct location.

Install *cron-nftfw* in */etc/cron.d/nftfw*.

```
# go to the nftfw distribution to find cronfiles
$ cd cronfiles
$ sudo cp cron-nftfw /etc/cron.d/nftfw
# cron wants the file to be writeable only by owner
$ sudo chmod g-w /etc/cron.d/nftfw
$ cd ..
```

Making *nftfw* control directories 'active'

The original Symbiosis system used the *incron* system to make the *nftfw* control directories active, ensuring firewall updates when files alter, appear or disappear. Sympl stopped using *incron* because it's poorly maintained and buggy. Without automatic updates, the system needs reloading whenever a user changes the contents of one of the directories, and forgetting to run the reload command is a source of errors. *nftfw* will work happily without active directories, but it makes the life of the system admin easier.

nftfw supplies control files for *systemd* using its ability to track file changes and starting a firewall reload when a change is detected in one of the control directories.

To install the files and start the system:

```
# go to the nftfw distribution to find systemd files
$ cd systemd
# check that the nftfw files contain the correct paths, edit if necessary
# install the files
$ sudo cp nftfw.* /etc/systemd/system
$ cd ..
# start the units
$ sudo systemctl enable nftfw.path
$ sudo systemctl start nftfw.path
$ sudo systemctl status nftfw.path
```

The last command should show that the unit is active.

You can test this by going to the *whitelist.d* directory and adding and removing 8.8.8.8 while monitoring */var/log/syslog*. The*nftfw* command should run and update the fire wall.

Finally, you can turn off *incron*, if it's running:

```
$ sudo systemctl stop incron
$ sudo systemctl disable incron
```

Finally a tip that's hard to find: reload *systemd* if you change the *nftfw* files after installation and starting:

```
$ sudo systemctl daemon-reload
```

Configuring the firewall

nftfw is distributed with no outbound packet control except for essential IPV6 rules (that are in */usr/local/etc/nftfw/nft_init.nft*. The set of inbound rules are aimed at permitting access to *ssh*, *http* and *https*, *ftp* and the various email subsystems. The incoming*ftp* rules are designed to support *Pure FTP*. Firewall configuration is a matter of creating or deleting files in the directories in */usr/local/etc/nftfw*. You probably need to change settings for your system. Scan through the How do I.. or a User's Quick Guide document for a quick start on setting up access for your needs.

Geolocation

The listing program *nftfwls* will print out the country that originated packets in the firewall using the *geoip2* country database available from MaxMind. MaxMind don't charge but want you to create an account with them to access their files.

See Installing Geolocation.

If you want to use the *blacknets* feature of *nftfw* to block countries, then *geoip2* can be used to supply lists of IP networks.

See Getting CIDR lists.

Sympl users: Update your mail system after installation

A repository that steps through the changes I make to the standard *exim4*/*dovecot* systems on Sympl to improve feedback and detection of bad IPs - see Sympl mail system update.

You Are There

Now look at:

- Updating *nftfw*
  - How to update *nftfw*.
- Installing Geolocation
  - Installing Geolocation, adding country detection to *nftfwls*, which is optional but desirable.
- Getting CIDR lists
  - How to get CIDR files for use with the *blacknet* feature..
- User's Guide to nftfw
  - The full User guide, the first section explains how the system is controlled.
- How do I.. or a User's Quick Guide
  - Answers a bunch of questions about the system.
- *nftfw* web site
  - All documents are available on the *nftfw* web site.

All documents can be found on the web from the *nftfw* website.

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.