**Chapter 7. The state machine**

This chapter will deal with the state machine and explain it in detail. After reading through it, you should have a complete understanding of how the State machine works. We will also go through a large set of examples on how states are dealt with within the state machine itself. These should clarify everything in practice.

**7.1. Introduction**

The state machine is a special part within iptables that should really not be called the state machine at all, since it is really a connection tracking machine. However, most people recognize it under the first name. Throughout this chapter I will use these names more or less as if they were synonymous. This should not be overly confusing. Connection tracking is done to let the Netfilter framework know the state of a specific connection. Firewalls that implement this are generally called stateful firewalls. A stateful firewall is generally much more secure than non-stateful firewalls since it allows us to write much tighter rule-sets.

Within iptables, packets can be related to tracked connections in four different so called states. These are known as **NEW**, **ESTABLISHED**, **RELATED** and **INVALID**. We will discuss each of these in more depth later. With the **--state** match we can easily control who or what is allowed to initiate new sessions.

All of the connection tracking is done by special framework within the kernel called conntrack. conntrack may be loaded either as a module, or as an internal part of the kernel itself. Most of the time, we need and want more specific connection tracking than the default conntrack engine can maintain. Because of this, there are also more specific parts of conntrack that handles the TCP, UDP or ICMP protocols among others. These modules grab specific, unique, information from the packets, so that they may keep track of each stream of data. The information that conntrack gathers is then used to tell conntrack in which state the stream is currently in. For example, UDP streams are, generally, uniquely identified by their destination IP address, source IP address, destination port and source port.

In previous kernels, we had the possibility to turn on and off defragmentation. However, since iptables and Netfilter were introduced and connection tracking in particular, this option was gotten rid of. The reason for this is that connection tracking can not work properly without defragmenting packets, and hence defragmenting has been incorporated into conntrack and is carried out automatically. It can not be turned off, except by turning off connection tracking. Defragmentation is always carried out if connection tracking is turned on.

All connection tracking is handled in the PREROUTING chain, except locally generated packets which are handled in the OUTPUT chain. What this means is that iptables will do all recalculation of states and so on within the PREROUTING chain. If we send the initial packet in a stream, the state gets set to **NEW** within the OUTPUT chain, and when we receive a return packet, the state gets changed in the PREROUTING chain to **ESTABLISHED**, and so on. If the first packet is not originated by ourself, the NEW state is set within the PREROUTING chain of course. So, all state changes and calculations are done within the PREROUTING and OUTPUT chains of the nat table.

I saw this iptables snippet in a different Super User answer:

iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

iptables -A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT

The point is to always allow packets that are sent as a part of an established connection. The part I'm wondering about is the top two lines.

For the INPUT chain, what is the point of writing both -m conntrack --ctstate RELATED,ESTABLISHED and -m state --state RELATED,ESTABLISHED. It seems to be both should do the same thing?

An explanation of the difference between these two would be great.

**Main answer :**

Conntrack supersedes state, but in modern kernels there is now no difference between the two. State is currently aliased and translated to conntrack in iptables if the kernel has it, so the syntax -m state --state is actually translated into -m conntrack --ctstate and handled by the same module.

On some old kernels, however, contrack has to be specifically enabled.

**Possible explanation :**

It seems to me as if the rules you quoted included duplicates, catering for both older and newer kernels.

Or maybe this is just a case of [Cargo cult programming](https://en.wikipedia.org/wiki/Cargo_cult_programming).

There is [this question on ServerFault](https://serverfault.com/q/358996) from the year 2012:

What's the practical difference between:

iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

and

iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

Which one is best to use?

The [accepted answer](https://serverfault.com/a/359031) is:

Both use same kernel internals underneath (connection tracking subsystem).

Header of xt\_conntrack.c:

xt\_conntrack - Netfilter module to match connection tracking

information. (Superset of Rusty's minimalistic state match.)

So I would say -- state module is simpler (and maybe less error prone). It's also longer in kernel. Conntrack on the other side has more options and features[1].

My call is to use conntrack if you need it's features, otherwise stick with state module.

[Similar question on netfilter maillist.](http://www.spinics.net/lists/netfilter/msg46827.html)

[1] Quite useful like -m conntrack --ctstate DNAT -j MASQUERADE" routing/DNAT fixup ;-)

One of the other answers leads to this [document about iptables](http://www.iptables.info/en/iptables-matches.html#CONNTRACKMATCH). It says:

The conntrack match is an extended version of the state match, which makes it possible to match packets in a much more granular way. It let's you look at information directly available in the connection tracking system, without any "frontend" systems, such as in the state match.

So I think this is true (from yet another answer there):

There is no difference in the outcome of those two rules.

Note there is also an interesting comment under the question:

state is deprecated in favor of conntrack, and may or may not be compiled in depending on how your kernel was built.

I don't claim to be an expert with iptables rules but the first command is making use of the connection tracking extension (conntrack) while the second is making use of the state extension.

Data point #1

According to [this document](http://inai.de/documents/Perfect_Ruleset.pdf) the conntrack extension superseded state.

Obsolete extensions:

• -m state: replaced by -m conntrack

Data point #2

Even so I found this SF Q&A titled: [Firewall questions about state and policy?](https://serverfault.com/questions/190978/firewall-questions-about-state-and-policy) where the OP claimed to have asked this question on IRC in #iptables@freenode. After discussing it there he came to the conclusion that:

Technically the conntrack match supersedes - and so obsoletes - the state match. But practically the state match is not obsoleted in any way.

Data point #3

Lastly I found this SF Q&A titled: [Iptables, what's the difference between -m state and -m conntrack?](https://serverfault.com/questions/358996/iptables-whats-the-difference-between-m-state-and-m-conntrack). The answer from this question is probably the best evidence and advice on how to view the usage of conntrack and state.

*excerpt*

Both use same kernel internals underneath (connection tracking subsystem).

Header of xt\_conntrack.c:

xt\_conntrack - Netfilter module to match connection tracking

information. (Superset of Rusty's minimalistic state match.)

So I would say -- state module is simpler (and maybe less error prone). It's also longer in kernel. Conntrack on the other side has more options and features [1].

My call is to use conntrack if you need it's features, otherwise stick with state module.

* [Similar question on netfilter maillist.](http://www.spinics.net/lists/netfilter/msg46827.html)

**[1]** Quite useful like "-m conntrack --ctstate DNAT -j MASQUERADE" routing/DNAT fixup ;-)

Data point #4

I found this thread from the netfilter@vger.kernel.org netfilte/iptables discussions, titled: [state match is obsolete 1.4.17](http://comments.gmane.org/gmane.comp.security.firewalls.netfilter.general/45564), which pretty much says that state is just an alias to conntrack so it doesn't really matter which you use, in both circumstances you're using conntrack.

*excerpt*

Actually, I have to agree. Why don't we keep "state" as an alias and accept the old syntax in "conntrack"?

state is currently aliased and translated to conntrack in iptables if the kernel has it. No scripts are broken.

If the aliasing is done in userspace, the kernel part can be removed - someday maybe.

The aliasing is already done in userspace. One types in "state" and it's converted into "conntrack" and that is then sent to the kernel. (So as far as I see if the ipt\_state, etc module aliases were added to the conntrack module, even the state kernel module could be removed.)

References

* [Firewall questions about state and policy?](https://serverfault.com/questions/190978/firewall-questions-about-state-and-policy)
* [iptables: differences using conntrack or state module](https://bbs.archlinux.org/viewtopic.php?pid=1028574)