 iptables - An Introduction

07.05.2012, Garching

1 Basics

iptables is executed using the `iptables` command. You may list current rules with:

```
iptables -L
```

Interesting is also the `-n` option which will only show numeric output (i.e., no DNS lookups). This will avoid potentially long DNS lookup times if there are a lot of rules to consider. In addition, the `-v` option will show verbose output that includes packet and byte counters per rule. This may be helpful for debugging.

2 Tables

There are three tables that iptables can act on. This table can be set with the `-t` option. The default table is the `filter` table and is the table we will be working with. For completeness sake, the other two are the `nat` and `mangle` tables, but we will not consider these. Each table has its own default set of chains. These will be explained in Section 3.2.

3 Chains

You can imagine a chain as a list (or chain) of rules that get considered in order until the packet being considered falls off the end of the chain if no rules act upon it.

3.1 Target Chains

The target chains are a special type of chain for which no rules can be specified, but rather represent a fate for a specific packet. These include:

- **ACCEPT** Any packet put on this chain will be accepted and allowed to continue.
- **DROP** Any packet put on this chain will be dropped and no longer considered.
- **REJECT** Any packet put on this chain will be dropped, however an error packet will be sent to the source.
- **LOG** Any packet put on this chain will be logged by the kernel (output via kernel messages). This chain has the interesting side effect that instead of falling off the chain, all packets that reach its end will be returned to the previously calling chain.
- **QUEUE** Any packet put on this chain will be handed to userland for further processing. This is how snort-inline is realized, for example.
- **RETURN** Any packet put on this chain will be returned to the previously calling chain.

This is not an exhaustive list, there exist many more target chains. In addition, each of these may take additional parameters or options. In order to learn more see the references mentioned in Section 6.
3.2 Default Chains

As mentioned in Section 2, the filter table has a set of default chains, namely INPUT, OUTPUT, and FORWARD. These are the chains that packets begin their processing on depending on their destination.

INPUT  Packets are put on this chain if they originate externally and their destination is the local host.

OUTPUT Packets are put on this chain if they originate from the local host and their destination is an external host.

FORWARD Packets are put on this chain if they originate externally and their destination is an external host.

Each of these default chains may have a default target chain that a packet is out on if it falls off the default chain. This may be set with the -P option. For example, if we wanted to define the default policy for the INPUT chain as ACCEPT, we would use:

iptables -P INPUT ACCEPT

3.3 User-defined Chains

A user may also define chains of their own. This is done with the -N option.

4 Adding Rules

It is important to note that for a particular chain, the rules are processed in the order that they are added and if a previous rules hits and puts the rule on a different chain, a subsequent rule will not be processed unless the RETURN target is used.

Adding a rule is as simple as using the -A option followed by the chain and the rule specifications. For example, to append a rule to the input chain, we used

iptables -A INPUT <rule specification>

4.1 Rule Specifications

When adding a rule these are some basic options that may make up the rule specification:

[p] -p <protocol>  This specifies a protocol for the rule to check. The protocol may consist of tcp, udp, icmp, all, etc (see references for more). The optional ! may be used to negate the rule.

[s] -s <source>  This specifies a source for the rule to check. The source may consist of an address or net mask. The optional ! may be used to negate the rule.

[d] -d <destination>  This specifies a destination for the rule to check. The destination may consist of an address or net mask. The optional ! may be used to negate the rule.

-j <target>  This specifies which chain should be jumped to if the rule hits. The target may be any of the chains discussed in Section 3.
**Special note:** There exists an in-interface and out-interface rule specification that will not work in bridged mode. In order to get this functionality, you must include a matching module, *phydev*, with the `-m` option. Once this is loaded you may use the `--physdev-in` and `--physdev-out` options. For example, a rule that drops all packets arriving on interface `eth0`, would look like this:

```
iptables -A FORWARD -m phydev --physdev-in eth0 -j DROP
```

For help on specific modules like *phydev*, you can use the syntax:

```
iptables -m phydev --help
```

### 5 Remaining Resident

These rules will *not* remain resident after a reboot, therefore the rules must be defined in a script that runs at start up. For Debian-based systems, see [http://www.debian-administration.org/articles/28](http://www.debian-administration.org/articles/28).

### 6 References

This document is by no means a complete reference for iptables usage. For further information see the `iptables` man pages or the official documentation ([http://www.netfilter.org/documentation/](http://www.netfilter.org/documentation/)).