Configuring a Firewall (iptables)

Firewall Concepts
Listing & Clearing Firewall Policies
Setting Default Firewall Policies
Setting Firewall Policy Exceptions
Making Firewall Settings Persistent

Configuring a Firewall (iptables)

Firewall Concepts
Listing & Clearing Firewall Policies
Setting Default Firewall Policies
Setting Firewall Policy Exceptions
Making Firewall Settings Persistent

Configuring a Firewall (iptables)

Since Linux servers can be connected to the Internet, it is very important to run a firewall to control what packets might come into the computer system, what packets might go out of the computer system, and what packets might be forwarded to another computer.

We are currently using the utility called iptables can be used to set the firewall rules on a Linux server. We disabled another firewall service called firewalld and installed the iptables-service. Although both can be run at the same time, iptables is considered to be the service of choice to use in order to manage a Linux/Unix firewall.

Configuring a Firewall (iptables)

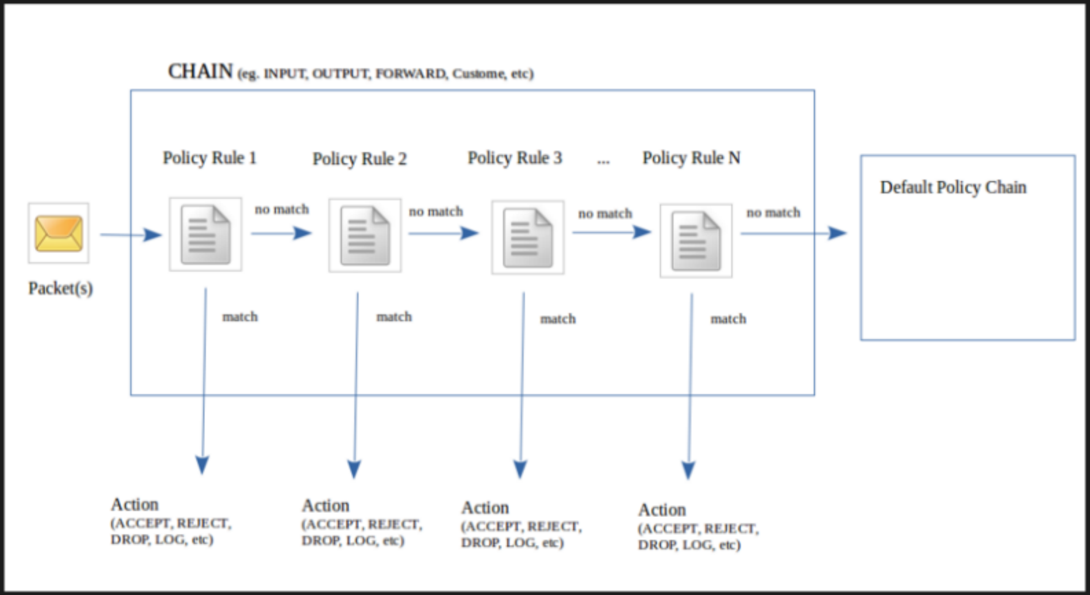
Basically, there is a list (chain) of policy rules that packets must pass-through in order to handle packets. If a packet matches a rule, then an action is taken (some examples include: ACCEPT, DROP, REJECT, or LOG). If the packet passes through the chain of rules without a match, then the packet is directed to the default policy chain (for example: ACCEPT, REJECT, or DROP).

You can create your own customized chains (which you will learn in the OPS335 course) but to keep thing simple, we only deal with 3 common predefined chains:

INPUT:	Packets coming into current Linux server
OUTPUT:	Packets leaving current Linux server
FORWARD:	Packets being routed between Linux servers

Configuring a Firewall (iptables)

When using iptables packets must pass-through "a chain of policy rules" in order to handle packets. If a packet matches a rule, then an action is taken (some examples include: ACCEPT, DROP, REJECT, or LOG); otherwise, the packet should be directed to the default policy chain. If the packet passes through the policies and default policy, then it is free to continue in or out of the Linux machine.


CHAIN (eg. INPUT, OUTPUT, FORWARD, Custome, etc)
Policy Rule 1
Policy Rule 2
Policy Rule 3
...
Policy Rule N
Packet(s)
no match
no match
no match
no match
match
match
match
match
Default Policy Chain
Action (ACCEPT, REJECT, DROP, LOG, etc)
Action (ACCEPT, REJECT, DROP, LOG, etc)
Action (ACCEPT, REJECT, DROP, LOG, etc)
Action (ACCEPT, REJECT, DROP, LOG, etc)

Configuring a Firewall (iptables)

The iptables command is used to allow the Linux/Unix adminstrator set-up policies in order to control access to the packets (tcp, udp, icmp, etc) that travel within a Linux/Unix server.

As when issuing and Linux command to perform a task, it is important to issue a command to verify that the operation was successful (i.e. verify your work).

Below are iptables commands to verify your firewall rules:

iptables -L	List all iptables rules (eg. INPUT, OUTPUT. FORWARD, and any customized chains (if any)
iptables -L -v	Verbosely List all iptables rules including information (such as total size of packets affected by rules)
iptables -L CHAIN-NAME	List all iptables rules for that particular chain-name for less clutter (eg. INPUT, OUTPUT, FORWARD)

Configuring a Firewall (iptables)

Although you can save and restore iptables settings, it may be useful to "flush" (completely clear) the firewall rules (or policies) for all or a particular chain.

Note the options that can be used to clear (or flush) the iptables rules:

iptables -F	Clears the rules for ALL of the chains
iptables -F CHAIN-NAME	Clears the rules for only the specified CHAIN-NAME (eg. INPUT or OUTPUT)

Configuring a Firewall (iptables)

Usually when setting policy rules with iptables, a general "overall" policy is set (default policy chain). A good way to think about setting policies is to have a "safety-net" to take some sort of action to prevent un-handled packets from passing through the firewall by mistake. After the default policy is set-up, then specific exceptions to the default policy can be added to control specific network traffic.

An example would be to set a default policy for incoming network traffic (INPUT chain) to DROP everything, and then set an exception certain exceptions (like ssh connections). Note the following table below for policy setting examples.

Configuring a Firewall (iptables)

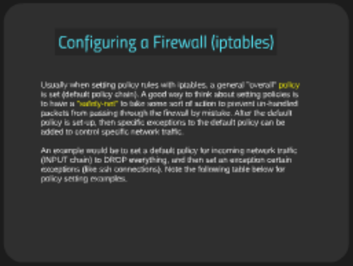
Here are examples of setting default policies using iptables:

```
iptables -P INPUT DROP
```

Drops all incoming packets regardless of protocol (eg. tcp, udp, icmp), port numbers (eg. 22, 80) or source or destination IP Addresses. Setting a default rule to DROP all incoming traffic would make it easier to specify a few exceptions.

```
iptables -P INPUT ACCEPT
```

Accepts all incoming packets regardless of protocol (eg. tcp, udp, icmp), port numbers (eg. 22, 80) or source or destination IP Addresses. It would seem that setting a default rule to ACCEPT all incoming traffic would require A LOT of exceptions to help "lock-down" the server for protection! It really depends on the server set-up and what the Linux system administrator wants to accomplish.



Configuring a Firewall (iptables)

Usually when setting policy rules with iptables, a general "overall" policy is set (default policy chain). A good way to think about setting policies is to have a "safety-net" to take some sort of action to prevent un-handled packets from passing through the firewall by mistake. After the default policy is set-up, then specific exceptions to the default policy can be added to control specific network traffic.

An example would be to set a default policy for incoming network traffic (INPUT chain) to DROP everything, and then set an exception certain exceptions (like ssh connections). Note the following table below for policy setting examples.

Configuring a Firewall (iptables)

If you have set a default policy to DROP all incoming network traffic, there is a problem: now, all incoming network traffic would be blocked by default!

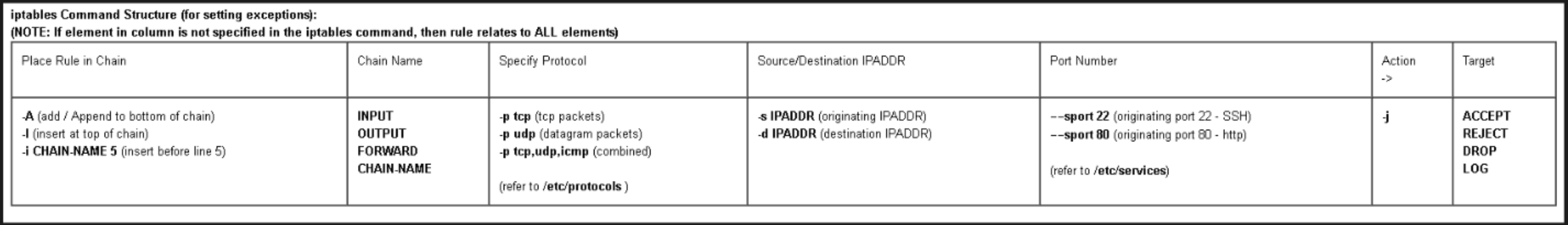
In order to fix that problem, you can make exceptions to the default firewall policy to allow incoming network traffic.

Those iptables commands to create exceptions are more complex since you need to determine:

- Where each rules appears in the chain? (order can be important)
- Which protocol(s) are affected (eg. tcp, udp, icmp)
- What source or destination IP Addresses are affected?
- What port numbers are affected?
- What action to take if all of the above conditions are met? (eg. ACCEPT, REJECT, DROP, or LOG)

Configuring a Firewall (iptables)

Double-click on the image below to see the iptables command structure in order to set iptables rules for exceptions to the default firewall policy.



iptables Command Structure (for setting exceptions):
(NOTE: If element in column is not specified in the iptables command, then rule relates to ALL elements)

Place Rule in Chain	Chain Name	Specify Protocol	Source/Destination IPADDR	Port Number	Action ->	Target
-**A** (add / Append to bottom of chain) -**I** (insert at top of chain) -**i CHAIN-NAME 5** (insert before line 5)	**INPUT** **OUTPUT** **FORWARD** **CHAIN-NAME**	**-p tcp** (tcp packets) **-p udp** (datagram packets) **-p tcp,udp,icmp** (combined) (refer to **/etc/protocols**)	-**s IPADDR** (originating IPADDR) -**d IPADDR** (destination IPADDR)	**--sport 22** (originating port 22 - SSH) **--sport 80** (originating port 80 - http) (refer to **/etc/services**)	-**j**	**ACCEPT** **REJECT** **DROP** **LOG**

Configuring a Firewall (iptables)

Firewall Concepts
Listing & Clearing Firewall Policies
Setting Default Firewall Policies
Setting Firewall Policy Exceptions
Making Firewall Settings Persistent