

IPWorks IPTables Service Configuration

USER GUIDE

Copyright

© Ericsson AB 2017, 2018. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

Trademark List

All trademarks mentioned herein are the property of their respective owners. These are shown in the document Trademark Information.



Contents

1	Introduction	1
1.1	Prerequisite	1
1.2	Related Information	1
2	Iptables Overview	2
3	IPWorks IPtables Configuration	3
3.1	Hardening overview	3
3.2	LDE Iptables Commands	8
3.3	Iptables Configuration for IPWorks	9
4	Appendix: Iptables Introduction	11
4.1	IP Filtering Terms and Expressions	11
4.2	Building Rules	12
4.3	Iptables Targets	30
	Reference List	35





1 Introduction

This document provides the configuration information for IPWorks iptables service.

For security reasons, IPWorks uses iptables software to enable the basic IP packet filtering.

Scope

This document includes:

- Iptables configuration for IPWorks, see Section 3 on page 3.

This section introduces the hardening related interfaces and provides sample examples for IPWorks.

- Iptables overview, see Section 4 on page 11.

This section introduces iptables relevant commands and elements.

r

Target Groups

This document is intended for personnel who want to implement IPWorks cluster nodes (SC and PL nodes) IP packet filter.

1.1 Prerequisite

The personnel implementing the IPWorks node IP packet filter must fulfill the following prerequisites:

- Intermediate Linux and UNIX skills.
- Concepts, terminologies, and telecommunication abbreviations, such as TCP/IP, UDP and ICMP protocols.

1.2 Related Information

Trademark information, typographic conventions, and definition and explanation of abbreviations and terminology can be found in the following documents:

- Trademark Information
- Typographic Conventions
- Glossary of Terms and Acronyms



2 Iptables Overview

Iptables is used to set up, maintain, and inspect the tables of IPv4 packet filter rules in the Linux kernel while ip6tables is the corresponding software for IPv6 filtering.

To configure iptables, the following must be known:

- Several different tables are defined.
- Each table contains a number of built-in chains and may also contain user-defined chains.
- Each chain is a list of rules , and these rules match a set of packets.
- Each rule specifies what to do with a packet that matches. This is called a 'target', which can be a jump to a user-defined chain in the same table.
- A process of configuring iptables is typically a process of building rules. A firewall rule specifies criteria for a packet and a target.
 - If the packet does not match, the next rule in the chain is examined.
 - If it does match, then the next rule is specified by the value of the target. The value of the target can be the name of a user-defined chain or one of the special values **ACCEPT**, **DROP** or **RETURN**.

For more information about Iptables, see Appendix: Iptables Introduction.



3 IPWorks IPtables Configuration

This section describes how to configure iptables and ip6tables for IPWorks SCs and PLs nodes.

3.1 Hardening overview

Figure 1 shows the basic communication between IPWorks service and the service needed by IPWorks to communicate with other service (like OSSRC, SSH client, etc).

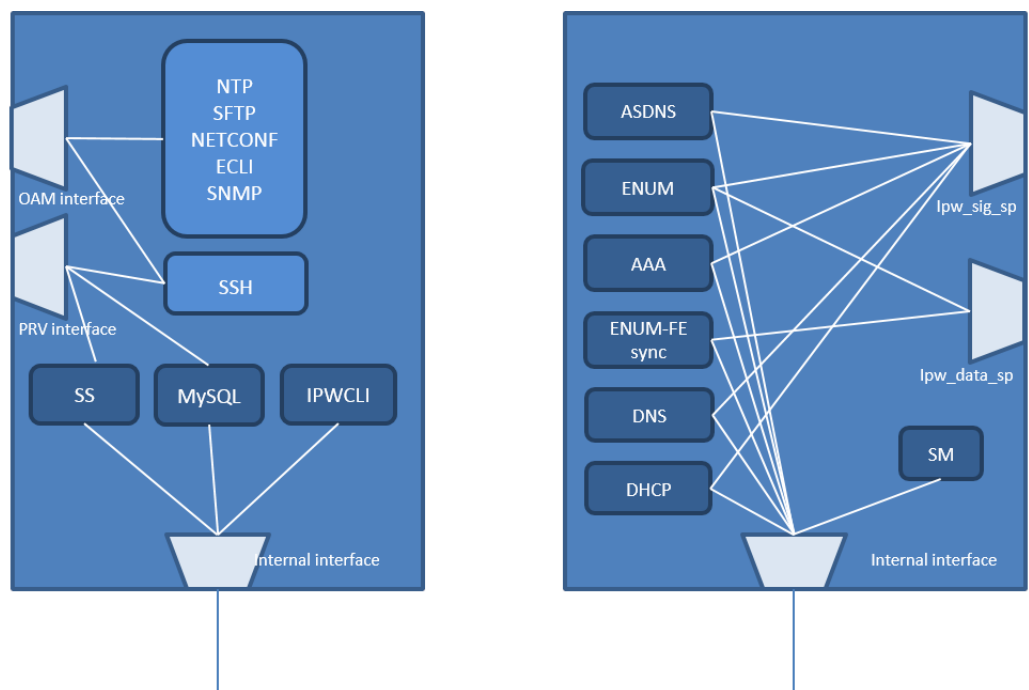


Figure 1 IPtables Hardening Interfaces

The iptables rules apply to following interfaces:

1. The internal interface is used for the communication between IPWorks services. For example: connection between Storage Server (SS) and MySQL, connection between Server Manager (SM) and SS, connection between ENUM and MySQL, connection between IPWCLI and SS.

The communications between SC and PL for the LDE services are also through the internal interface, For example: DHCP, TFTP, NFS, and SSH services, etc.

Therefore, for the internal interface, it is recommended to open it for in/out packets.



2. For the SC nodes, the OAM interface is used for the services: SSH, SFTP, NETCONF, ECLI, SNMP and provisioning.

Here, it is recommended to do the ipfilter according to these services.

3. For the SC nodes, the Provisioning interface is used for the customer to do the provisioning via SSH.

So, it is recommended to do the ipfilter according the SSH service.

4. For the SC nodes, for Geograph redundancy feature, the sqlnode is connected with the Geograph redundancy node's SC-1/2 via port 3307. Here, it is recommended to do the ipfilter according to these services.

5. For the PL nodes, the traffic interface is used for the DNS, ENUM, ASDNS, ENUM-FE sync, AAA, and DHCP to communicate with external network.

Here, it is recommended to do the ipfilter according to the DNS/ENUM, AAA, and DHCP service.

Table 1 lists LDE services that are related to the hardening.

Table 1 LDE Services

Service	IPWorks Port	Peer Port	Interface	Needed Hardened?
SSH	22	Dynamic TCP port	OAM & Provisioning	Yes
SFTP	22	Dynamic TCP port	OAM	Yes
NETCONF	830	Dynamic TCP port	OAM	Yes
ECLI	830	Dynamic TCP port	OAM	Yes
SNMP	161	Dynamic TCP port	OAM	Yes
	Dynamic UDP port	162	OAM	Yes
	Dynamic UDP port	123	OAM	Yes
Rsyslog	Dynamic UDP port	514	OAM	Yes
	Dynamic TCP port	10514	OAM	Yes



Table 1 LDE Services

Service	IPWorks Port	Peer Port	Interface	Needed Hardened?
LDAP Auth	Dynamic TCP port	389	OAM	Yes
ERH and AAA subagent to SS7 stack Operation and Maintenance (OAM)	Dynamic TCP port	6669	OAM	Yes

Table 2 lists IPWorks services that are related to the hardening.

Table 2 IPWorks Services

Service	IPWorks Port	Peer Port	Interface	Needed Hardened?
MySQL NDB Cluster	1186 (Management Node)	Dynamic TCP port	Internal	No
	3307 (SQL Node)	Dynamic TCP port	Internal	No
	3307 (SQL Node)	Dynamic TCP port	Provisioning	Yes
Storage Server	17071	Dynamic TCP port	Internal	No
	17071	Dynamic TCP port	External for IPWorks DNS Management	Yes
IPWCLI	Dynamic TCP port	17071	Internal	No
Server Manager	Dynamic TCP port	17071	Internal	No
DNS	53	Dynamic TCP port	EVIP/Loopback	Yes
	5300	Dynamic UDP port	EVIP/Loopback	Optional
ENUM	53	Dynamic UDP port	EVIP	Yes
Diameter AAA	3868 (Authentication/Authorization)	Dynamic TCP/SCTP port	EVIP	Yes
	18681 (OM Stack)	Dynamic TCP port	Internal	No



Table 2 IPWorks Services

Service	IPWorks Port	Peer Port	Interface	Needed Hardened?
FE (ENUM and AAA)	Dynamic TCP port	389	EVIP	Yes
ASDNS	ICMP	ICMP	Traffic	Yes
FESync	8080	Dynamic TCP port	EVIP	Yes
ASDNS	Dynamic UDP port	161	EVIP	Yes
SS7CAF	2905	Dynamic SCTP port	EVIP	Yes



Table 2 IPWorks Services

Service	IPWorks Port	Peer Port	Interface	Needed Hardened?
Radius AAA	1812 (Authentication)	Dynamic UDP port	EVIP	Yes
	1813 (Accounting)	Dynamic UDP port	EVIP	Yes
	3799 (Dynamic Authority)	3799 (Dynamic Authority)	EVIP	Yes
	3800 (Dynamic Authority for PL-3)	3800 (Dynamic Authority for PL-3), for DM/COA message triggered from IPWCLI. Acts as DM/COA client.	EVIP	Yes
	3801 (Dynamic Authority for PL-4)	3801 (Dynamic Authority for PL-4), for DM/COA message triggered from IPWCLI. Acts as DM/COA client.	EVIP	Yes
	380x (Dynamic Authority for PL-y) ⁽¹⁾	380x (Dynamic Authority for PL-y), for DM/COA message triggered from IPWCLI. Acts as DM/COA client.	EVIP	Yes
	6826 (OM Stack)	Dynamic TCP port	Internal	No
	6827	Dynamic TCP port	Internal	No
	6828	Dynamic TCP port	Internal	No
	56165 (CSV Engine)	Dynamic TCP port	Internal	No



Table 2 IPWorks Services

Service	IPWorks Port	Peer Port	Interface	Needed Hardened?
DHCPv4	67	Dynamic UDP port	EVIP	Yes
	68	Dynamic UDP port	EVIP	Yes
	647 (Failover Protocol)	Dynamic TCP port	EVIP/Internal	Yes
	847 (Failover Protocol)	Dynamic TCP port	Internal	No

(1) $380x = 3799 + (y-2)$. y is the ID number of PL (start from 3) .

3.2 LDE Iptables Commands

The LDE iptables is configured in `/cluster/etc/cluster.conf` on LDE.

The following tables list the command introductions that are provided by LDE.

Table 3 Iptables

Syntax	<code>iptables <target> <command></code>	
Description	Defines a rule in iptables. Rules will be run in the order specified in this configuration.	
Options	<target> <command>	Target blade(s). Specifies the parameters that should be passed to iptables. This can be any parameter accepted by iptables.
Examples	On all nodes, drop packets destined from source address 10.0.0.1: <pre>iptables all -A INPUT -s 10.0.0.1 -j DROP</pre> On all nodes, accept SSH traffic destined for the 192.168.0.0/24 network and drop all other SSH traffic: <pre>iptables all -A INPUT -p tcp --dport 22 -d 192.168.0.0/24 -j ACCEPT</pre> <pre>iptables all -A INPUT -p tcp --dport 22 -j DROP</pre>	



Table 4 Ip6tables

Syntax	<code>ip6tables <target> <command></code>	
Description	Defines a rule in ip6tables. Rules will be run in the order specified in this configuration.	
Options	<target> <command>	Target node(s). Specifies the parameters that are to be passed to ip6tables. This can be any parameter accepted by ip6tables.
Examples	<code>ip6tables all -A INPUT -s fe80::21f:29ff:fe04:f9fa -j DROP</code>	

3.3 Iptables Configuration for IPWorks

Iptables is configured by adding rules to `/cluster/etc/cluster.conf`. The creation of iptables rules is associated with the information provided in Section 3.1 on page 3 and Section 3.2 on page 8.

Following table 1 is an example for the configuration of the `cluster.conf` for IPWorks.

Table 5 Parameter Description

Parameter	Explanation
<code><ssh client ip/net></code>	The client IP/net used to access control node through SSH.
<code><ssh port></code>	The SSH listening port with the default value 22. If you have configured the SSH port according to IPWorks OS Hardening Guide , make sure that the values are consistent.
<code><ecli client ip/net></code>	The client IP/net that you want to access the ECLI to do the configuration.
<code><OSSRC ip/net></code>	The OSSRC IP/net.
<code><Remote log server IP></code>	The IP address of remote log server to which you want to transfer your security log.
<code><Remote log server UDP port></code>	The UDP port of remote log server with the default value 514. According to IPWorks Security Log Management Guide , if you reset it, you need to change it.



Parameter	Explanation
<Remote log server TCP port>	The TCP port of remote log server with the default value 10514. If you have configured the SSH port according to IPWorks Security Log Management Guide , make sure that the values are consistent.
<Ldap server IP>	The LDAP server IP that is used for OAM authentication.
<Ldap server port>	The LDAP server listening port with the default value 389. It is according to your LDAP server configuration.
<provision client ip/net>	The ip/net of IPWorks remote provisioning.
<CUDB IP> ,	The CUDB IP address.
<SS7 client IP>	The SS7 client IP addresses.
<M3UA port>	The M3UA port number with the default value 2905. If you have configured the SSH port according to Configuring SS7 Signaling Network, SCCP, M3 , make sure that the values are consistent.
<MIP of MySQL Cluster SQL Node in peer Site>	The MIP of MySQL Cluster SQL Node/provision in peer site.
<Local site's SC-1 oam address>	The SC-1 OAM address of Local Site.
<Local site's SC-2 oam address>	The SC-2 OAM address of Local Site.

Note: For the IPTables configuration example, you can see the example in the file [IPtables Configuration Example](#).



4 Appendix: Iptables Introduction

4.1 IP Filtering Terms and Expressions

To fully understand the upcoming chapters, below is a list of the most common terms used in the IP filtering that one must understand.

The list also includes details about the TCP/IP chapter.

- **Drop/Deny** - When a packet is dropped or denied, it is simply deleted, and no further actions are taken.

Neither reply to the host it was dropped, nor notify the receiving host of the packet in any way. The packet simply disappears.

- **Reject** - Basically the same as a drop or deny target or policy, except that here we reply to the host sending the packet that was dropped.

The reply is specified, or automatically calculated to some value.

So far, there is no iptables functionality to send a packet notifying the receiving host of the rejected packet of what happened (For example, doing the reverse of the Reject target. This is very good in certain circumstances, since the receiving host has no ability to stop Denial of Service attacks from happening.).

- **State** - A specific state of a packet in comparison to a whole stream of packets. For example:

- If the packet is the first one that the firewall sees or knows about, it is considered as new (the SYN packet in a TCP connection).

OR

- If it is part of an already established connection that the firewall knows about, it is considered as established.

States are known through the connection tracking system, which keeps track of all the sessions.

- **Chain** - A chain contains a ruleset of rules that are applicable to packets that traverses the chain.

Each chain has a specific purpose (for example, which table it is connected to, which specifies what this chain is able to do) as well as a specific application area .

For example, the INPUT is the only packet destined for this host, OUTPUT is the only packets sent out from this host.



- **Match** - This word only has two different meanings when it comes to IP filtering:
 - The first meaning is that a single match tells a rule that this header must contain certain information.

For example: The `--source` match means that the source address must be a specific network range or host address.
 - The second meaning is that a whole rule is a match. If the packet matches the whole rule, the jump or target instructions are carried out.

For example: The packet is dropped.
- **Target** - Each rule in a ruleset has a target set.

If the rule has matched fully, the target specification is what to do with the packet.

For example, the packet is dropped or accepted and so on.
- **Rule** - A rule is a set of a match or several matches together with a single target in most implementations of IP filters, including iptables implementation.

Some implementations let you use several targets/actions per rule.
- **Jump** - The jump instruction is closely related to a target. A jump instruction is written exactly the same as a target in iptables, with the exception that instead of writing a target name, you write the name of another chain.

If the rule matches, the packet is then sent to this second chain and be processed as usual in that chain.
- **Accept** - To accept a packet and to let it through the firewall rules. This is the opposite of the drop or deny targets, as well as the reject target.
- **Policy** - The implementation of the default behavior to take on a packet if there was no rule that matched it.

4.2 Building Rules

A rule can be described as the directions the firewall adheres to when blocking or permitting different connections and packets in a specific chain.

Each command line inserted in a chain is considered a rule.

4.2.1 Typical Iptables Commands

```
iptables command [match] [target/jump]
```




Command

The following tables show the details for command (Table 6), Options (Table 7), Generic Matches (Table 8), TCP Matches (Table 9), UDP Matches (Table 10), ICMP Matches (Table 11) respectively.

Table 6 Command

Command	-A, --append
Example	iptables -A INPUT
Explanation	This command appends the rule to the end of the chain. The rule is always put last in the rule-set and hence checked last, unless you append more rules after it.
Command	-D, --delete
Example	iptables -D INPUT -p tcp --dport 80 -j DROP, iptables -D INPUT 1
Explanation	This command deletes a rule in a chain. This can be done in two ways: <ul style="list-style-type: none"> • Entering the whole rule to match. If you use this one, your entry must match the entry in the chain exactly. • Specifying the rule number that you want to match. If you use this one, you must match the number of the rule you want to delete. The rules are numbered from the top of each chain, starting with number 1.
Command	-R, --replace
Example	iptables -R INPUT 1 -s 192.168.0.1 -j DROP
Explanation	This command replaces the old entry at the specified line. This commands works in the same way as the --delete command. However, instead of totally deleting the entry, it replaces it with a new entry.
Command	-I, --insert
Example	iptables -I INPUT 1 -p tcp --dport 80 -j ACCEPT
Explanation	This command inserts a rule somewhere in a chain. The rule is inserted as the actual number that we specify. The above example is inserted as rule 1 in the INPUT chain, and hence from now on it is the first rule in the chain.
Command	-L, --list
Example	iptables -L INPUT



Explanation	<p>This command lists all the entries in the specified chain.</p> <p>In the above case, we would list all the entries in the INPUT chain.</p> <p>It's also legal to not specify any chain at all.</p> <p>The exact outputs are different depending on different options this command is used with, for example, the <code>-n</code> and <code>-v</code> options and so on.</p>
Command	-F, --flush
Example	<code>iptables -F INPUT</code>
Explanation	<p>This command flushes all rules from the specified chain and is equivalent to deleting all rules one by one, but faster.</p> <p>The command is used without options, and can delete all rules in all chains within the specified table.</p>
Command	-Z, --zero
Example	<code>iptables -Z INPUT</code>
Explanation	<p>This command tells the program to zero all counters in a specific chain, or in all chains.</p> <p>This command works the same as <code>-L</code>, except that <code>-Z</code> does not list the rules.</p> <p>If <code>-L</code> and <code>-Z</code> are used together (which is legal):</p> <ul style="list-style-type: none">• Firstly, the chains are listed.• Secondly, the packet counters are zeroed.
Command	-N, --new-chain
Example	<code>iptables -N allowed</code>
Explanation	<p>This command tells the kernel to create a new chain of the specified name in the specified table.</p> <p>In the above example, we create a chain called allowed.⁽¹⁾</p>
Command	-X, --delete-chain
Example	<code>iptables -X allowed</code>
Explanation	<p>This command deletes the specified chain from the table.</p> <p>The prerequisite of this command is:</p> <p>You must replace or delete all rules referring to the chain before actually deleting the chain.</p> <p>If this command is used without any options, all chains but those built into the specified table are deleted.</p>



Command	-P, --policy
Example	iptables -P INPUT DROP
Explanation	<p>This command tells the kernel to set a specified default target, or policy, on a chain.</p> <p>All packets that do not match any rule are then forced to use the policy of the chain.</p> <p>Legal targets are:</p> <ul style="list-style-type: none"> • Drop. • Accept.
Command	-E, --rename-chain
Example	iptables -E allowed disallowed
Explanation	<p>This command tells iptables to change the first name of a chain to the second name.</p> <p>In the example above, we change the name of the chain from allowed to disallowed.⁽²⁾</p>

(1) Make sure no chain or target of the same name exists.

(2) This doesn't affect the actual way the table works, but it's just a cosmetic change to the table.

Table 7 Options

Option	-v, --verbose
Commands used with	--list, --append, --insert, --delete, --replace



Explanation	<p>This option gives the verbose output.</p> <p>This option is mainly used with <code>--list</code> command, and the program outputs:</p> <ul style="list-style-type: none">• The interface address.• Rule options.• TOS masks. <p>If the <code>--verbose</code> option is set, the <code>--list</code> command also includes a byte and packet counter for each rule.</p> <p>These counters use below multipliers:</p> <ul style="list-style-type: none">• K (x1000)• M (x1,000,000)• G (x1,000,000,000) <p>To overrule this and get exact output, you can use the <code>-x</code> option as described below.</p> <p>Besides mainly used with <code>--list</code> command, this option can also be used with the <code>--append</code>, <code>--insert</code>, <code>--delete</code> or <code>--replace</code> commands, the program outputs detailed information as below and so on:</p> <ul style="list-style-type: none">• How the rule is interpreted.• Whether it is inserted correctly.
Option	-x, --exact
Commands used with	<code>--list</code>
Explanation	<p>This option expands the numerics.</p> <p>The output from <code>--list</code> does not contain the K, M or G multipliers.</p> <p>Instead we can get an exact output from the packet and byte counters about how many packets and bytes that have matched the rule in question.</p> <p>This option is only applicable to the <code>--list</code> command and isn't relevant to any other command.</p>
Option	-n, --numeric
Commands used with	<code>--list</code>



Explanation	<p>This option tells iptables to output numerical values. IP addresses and port numbers are printed by using their numerical values and not host-names, network names or application names.</p> <p>This option is only applicable to the <code>--list</code> command and only in this situation, this option overrides the default of resolving all numerics to hosts and names.</p>
Option	--line-numbers
Commands used with	<code>--list</code>
Explanation	<p>The option is used together with the <code>--list</code> command to output line numbers.</p> <p>This option outputs a corresponding number for each rule, so it is convenient to know which rule has which number when inserting rules.</p> <p>This option is only applicable to the <code>--list</code> command.</p>
Option	-c, --set-counters
Commands used with	<code>--insert</code> , <code>--append</code> , <code>--replace</code>
Explanation	<p>This option is used to:</p> <ul style="list-style-type: none"> • Firstly, create a rule or modify a rule. • Secondly, initialize the packet and byte counters for the rule. <p>The syntax is something like <code>--set-counters 20 4000</code>, which tells the kernel to set the packet counter to 20 and byte counter to 4000.</p>
Option	--modprobe
Commands used with	All
Explanation	<p>This option tells iptables which module to use when probing for modules or adding them to the kernel.</p> <p>This option is useful only when the <code>modprobe</code> command is not in the search path and so on.</p> <p>In other words, you must specify this option so the program knows what to do in case a required module is not loaded.</p> <p>This option can be used with all commands.</p>



Table 8 Generic Matches

Match	-p, --protocol
Example	<code>iptables -A INPUT -p tcp</code>
Explanation	<p>This match is used to check certain protocols.</p> <p>The protocol must be one of the internally specified as below:</p> <ul style="list-style-type: none">• TCP• UDP• ICMP <p>This protocol can also :</p> <ul style="list-style-type: none">• Take a value specified in the <code>/etc/protocols</code> file. If iptables can't find the protocol there, it replies with an error.• Be an integer value. For example, the ICMP protocol is an integer value 1, TCP is 6 and UDP is 17.• Take the value ALL. ALL means that it matches only TCP, UDP and ICMP. If this match is given the integer value of zero (0), it means all protocols, which in turn is the default behavior, if the <code>--protocol</code> match is not used. <p>To invert this match, add a ! sign, so <code>--protocol ! tcp</code> means matching UDP and ICMP.</p>
Match	-s, --src, --source
Example	<code>iptables -A INPUT -s 192.168.1.1</code>



Explanation	<p>This match is used to match packets based on their source IP address. The main form can be used to match single IP addresses, such as 192.168.1.1.</p> <p>It can be used with:</p> <ul style="list-style-type: none"> • A netmask in a CIDR "bit" form, by specifying the number of ones (1's) on the left side of the network mask. This means that we can for example add /24 to use a 255.255.255.0 netmask. Then we can match the whole IP ranges, such as our local networks or network segments behind the firewall. The line then looks something like 192.168.0.0/24. This can match all packets in the 192.168.0.x range. • A regular netmask in the 255.255.255.255 form (For example, 192.168.0.0/255.255.255.0). <p>To invert this match, add an ! sign.</p> <p>For example, if we use a match in the form of --source ! 192.168.0.0/24, we can match all packets with a source address not coming from any one within the 192.168.0.x range.</p> <p>The default is to match all IP addresses.</p>
Match	-d, --dst, --destination
Example	iptables -A INPUT -d 192.168.1.1
Explanation	<p>This match is used for packets based on their destination address or addresses.</p> <p>This match works basically the same as the --source match and has the same syntax, except that the --destination match is based on where the packets are going.</p> <p>To match an IP range, we can add a netmask either in the exact netmask form, or in the number of ones (1's) counted from the left side of the netmask bits.</p> <p>For example: 192.168.0.0/255.255.255.0 or 192.168.0.0/24. They are equivalent.</p> <p>To invert this match, add an ! Sign.</p> <p>For example, --destination ! 192.168.0.1 matches all packets except those destined to the 192.168.0.1 IP address.</p>
Match	-i, --in-interface
Example	iptables -A INPUT -i eth0



Explanation	<p>This match is used for the interface the packet came in on.</p> <p>This option is only legal in the INPUT, FORWARD and PREROUTING chains and will return an error message when used anywhere else.</p> <p>The default behavior of this match, if no particular interface is specified, is to assume a string value of +.</p> <p>The + value is used to match a string of letters and numbers.</p> <p>In other words, a single + tells the kernel to match all packets without considering which interface it came in on.</p> <p>The + string can also be appended to the type of interface, so eth+ can be all Ethernet devices.</p> <p>To invert this match, add an !sign.</p> <p>For example, -i ! eth0 matches all incoming interfaces except eth0.</p>
Match	-o, --out-interface
Example	iptables -A FORWARD -o eth0
Explanation	<p>The --out- interface match is used for packets on the interface from which they are leaving.</p> <p>This match is only available in the OUTPUT, FORWARD and POSTROUTING chains, the opposite to the --in- interface match</p> <p>Other than this, it works basically the same as the --in- interface match.</p> <p>The + extension means matching all devices of similar type, and eth+ means matching all eth devices and so on.</p> <p>To invert this match, add ! sign.</p> <p>If no --out-interface is specified, the default behavior of this match is to match all devices, regardless of where the packet is going.</p>
Match	-f, --fragment
Example	iptables -A INPUT -f



Explanation	<p>This match is used to match the second and third part of a fragmented packet. The reasons are:</p> <ul style="list-style-type: none"> • In the case of fragmented packets, we can't tell the source or destination ports of the fragments, nor the ICMP types, among other things. • In some rather special cases, fragmented packets can be used to compound attacks against other computers. These packet fragments can not be matched by other rules. <p>To invert this match, add an ! sign.</p> <p>However, in this case the ! sign must precede the match. For example, ! -f means :</p> <ul style="list-style-type: none"> • We match all the first fragments of fragmented packets, and not the second, third, and so on. • We match all packets that have not been fragmented during transfer. <p>(1)(2)</p>

(1) You can use other good defragmentation options within the kernel.

(2) If you use connection tracking, you can't see any fragmented packets, since they are dealt with before hitting any chain or table in iptables.

Table 9 TCP Matches

Match	--sport, --source-port
Example	iptables -A INPUT -p tcp --sport 22



Explanation	<p>The match⁽¹⁾ is used to match packets based on their source port. Without it, we imply all source ports.</p> <p>This match can either take a service name or a port number:</p> <ul style="list-style-type: none">• If you specify a service name, the service name must be in the /etc/services file, since iptables uses this file in which to find.• If you specify the port by its number, the rule can load slightly faster, since iptables doesn't have to check up the service name. <p>However, using a port number is harder to read than using a service name. If you are writing a rule-set consisting of a 200 rules or more, you can use port numbers, since the difference is noticeable. (On a slow box, this could make as much as 10 seconds' difference, if you have configured a large rule-set containing 1000 rules or so).</p> <p>You can also use the --source-port match to match any range of ports, --source-port 22:80 for example. This example matches all source ports between 22 and 80.</p> <ul style="list-style-type: none">• If the first port specification is omitted, port 0 is assumed (implicit). --source-port :80 then matches port 0 through 80• If the last port specification is omitted, port 65535 is assumed . If you write --source-port 22:, you must have specified a match for all ports from port 22 through port 65535. <p>If you invert the port range, iptables automatically reverses your inversion.</p> <p>If you write --source-port 80:22, it is simply interpreted as --source-port 22:80.</p> <p>To invert this match, adding a ! sign.</p> <p>For example, --source-port ! 22 means that you want to match all ports but port 22.</p> <p>The inversion is used with a port range and then looks like --source-port ! 22:80, which in turn means that you want to match all ports but ports 22 through 80. For more information, see Section 4.2.2.3 on page 30.</p>
Match	--dport, --destination-port
Example	<code>iptables -A INPUT -p tcp --dport 22</code>



Explanation	<p>This match⁽¹⁾ is used to match TCP packets depending on their destination port.</p> <p>This match:</p> <ul style="list-style-type: none"> • Uses exactly the same syntax as the --source-port match. • Understands port and port range specifications, as well as inversions. • Reverses high and low ports in port range specifications, as above. • Assumes values of 0 and 65535 if the high or low port is left out in a port range specification, exactly the same as the --source-port syntax. <p>For more information, see Section 4.2.2.3 on page 30.</p>
Match	--tcp-flags
Example	<code>iptables -A INPUT -p tcp --tcp-flags SYN,FIN,ACK SYN</code>
Explanation	<p>This match is used to match the TCP flags in a packet.</p> <ul style="list-style-type: none"> • The match takes a list of flags to compare (a mask). • The match takes a list of flags that are set to 1, or turned on. <p>Both lists are comma-delimited.⁽²⁾ You can see the correct syntax in the example above.</p> <p>This match :</p> <ul style="list-style-type: none"> • Knows about the SYN, ACK, FIN, RST, URG, PSH flags • Recognizes the words ALL and NONE. ALL and NONE is pretty much self describing: ALL means to use all flags and NONE means to use no flags for the option. --tcp-flags ALL NONE means to check all of the TCP flags and match if none of the flags are set. <p>To invert this match, add a ! sign.</p> <p>For example, if we specify ! SYN,FIN,ACK SYN, we get a match that matches packets that had the ACK and FIN bits set, but not the SYN bit.</p>
Match	--syn
Example	<code>iptables -A INPUT -p tcp --syn</code>



Explanation	<p>This match is an old relic from the ipchains and is still there for:</p> <ul style="list-style-type: none">• Backward compatibility.• Making it easier to transit from one to another. <p>This match is used to match packets if they have the SYN bit set and the ACK and RST bits unset.</p> <p>In this case, this command works exactly the same as the <code>--tcp-flags SYN, RST, ACK SYN</code> match. Such packets are mainly used to request new TCP connections from a server.</p> <p>If you block these packets, you effectively block all incoming connection attempts. However, you don't block the outgoing connections. These connections are used by lots of exploits today (For example, hacking a legitimate service and then installing a program or suchlike that enables initiating an existing connection to your host, instead of opening up a new port on it.).</p> <p>To invert this match, add an <code>!</code> sign. For example, <code>! --syn</code> way.</p> <p>This matches all packets with the RST or the ACK bits set, in other words packets in an already established connection.</p>
Match	<code>--tcp-option</code>



Example	<code>iptables -A INPUT -p tcp --tcp-option 16</code>
Explanation	<p>This match is used to match packets depending on their TCP options. A TCP Option is a specific part of the header.</p> <p>This part consists of 3 different fields:</p> <ul style="list-style-type: none"> • First: 8 bits long - describing which Options are used in this stream • Second: 8 bits long - describing how long the options field is • Third: describing the content of the used option. <p>The reason for this length field is that TCP options are optional. To be compliant with the standards, we do not need to implement all the options, but instead we can just look at what kind of option it is, and if we do not support it, we just look at the length field and can then jump over this data.</p> <p>This match is used to match different TCP options depending on their decimal values.</p> <p>To invert this match, add an ! sign, so that the match matches all TCP options but the option given to the match.</p>

(1) This match does not handle multiple separated ports and port ranges.

(2) The comma delimitation should not include spaces.

Table 10 UDP Matches

Match	<code>--sport, --source-port</code>
Example	<code>iptables -A INPUT -p udp --sport 53</code>



Explanation	<p>This match works exactly the same as its TCP counterpart. This match is used to perform matches on packets based on their source UDP ports.</p> <p>It supports below items with the same syntax:</p> <ul style="list-style-type: none">• Port ranges• Single ports• Port inversions <p>To specify a UDP port range, you can use 22:80 which matches UDP ports 22 through 80:</p> <ul style="list-style-type: none">• If the first value is omitted, port 0 is assumed.• If the last port is omitted, port 65535 is assumed.• If the high port comes before the low port, the ports switch place with each other automatically. <p>For Single UDP port matches, see the above example.</p> <p>To invert the port match, add an ! sign.</p> <p>For example, --source-port ! 53 matches all ports but port 53.</p> <p>The match understands service names, as long as the names are available in the /etc/services file. ⁽¹⁾ For more information, see Section 4.2.2.3 on page 30.</p>
Match	--dport, --destination-port



Example	<code>iptables -A INPUT -p udp --dport 53</code>
Explanation	<p>This match⁽¹⁾ is the same as the <code>--source-port</code> match above.</p> <p>It is equivalent to the TCP match, but here it applies to the UDP packets based on their UDP destination port.</p> <p>This match handles:</p> <ul style="list-style-type: none"> • Port ranges. • Single ports. • Port inversions. <p>To match a single port, for example, you use <code>--destination-port 53</code>. It matches all UDP packets going to port 53.</p> <p>To invert this match, add an <code>!</code> sign. For example, <code>--destination-port ! 53</code> matches all packets but those going to the destination port 53.</p> <ul style="list-style-type: none"> • The first matches all UDP packets going to port 53. • The second matches the packets but those going to the destination port 53. <p>To specify a port range, for example, you use <code>--destination-port 9:19</code>. It matches all packets destined for UDP port 9 through 19.</p> <ul style="list-style-type: none"> • If the first port is omitted, port 0 is assumed. • If the second port is omitted, port 65535 is assumed. • If the high port is placed before the low port, they automatically switch place, so the low port winds up before the high port. <p>For more information, See Multiport Match Extension.</p>

(1) This match does not handle multiple separated ports and port ranges.



Table 11 ICMP Matches

Match	--icmp-type
Example	iptables -A INPUT -p icmp --icmp-type 8
Explanation	<p>This match is used to specify the ICMP type to match.</p> <p>ICMP types can be specified either by either of the following:</p> <ul style="list-style-type: none">• Their numeric values.• Their names. <p>Numerical values are specified in RFC 792.</p> <p>To find a complete listing of the ICMP name values, do either of the following:</p> <ul style="list-style-type: none">• Type <code>iptables --protocol icmp --help</code>• Check the ICMP types appendix. <p>To invert this match, add an ! sign in this --icmp-type ! 8 way.⁽¹⁾</p> <p>The type and code can be specified by their type names, numeric types, and type/code as well.</p> <p>For example <code>--icmp-type network-redirect</code>, <code>--icmp-type 8</code> or <code>--icmp-type 8/0</code>.</p> <p>For a complete listing of the names, type:</p> <p><code>iptables -p icmp --help</code></p> <p>⁽²⁾</p>

(1) Some ICMP types are obsolete, and others again may be "dangerous" for an unprotected host since they may, among other things, redirect packets to the wrong places.

(2) Netfilter uses ICMP type 255 to match all ICMP types. If you try to match this ICMP type, you will wind up with matching all ICMP types.

4.2.2 Explicit Matches

4.2.2.1 IP Range Match

The IP range match is used to match IP ranges, just as the `--source` and `--destination` matches are able to do as well. However, this match adds a different matching that the manner of from IP - to IP can match but the `--source` and `--destination` matches can not. This is needed in some specific network setups, which is more flexible.

Use `-m iprange` keyword to load the IP range match.



Table 12 IP Range Match

Match	--src-range
Example	<code>iptables -A INPUT -p tcp -m iprange --src-range 192.168.1.13-192.168.2.19</code>
Explanation	<p>This matches a range of source IP addresses. The range includes every single IP address from the first to the last, so the example above includes everything from 192.168.1.13 to 192.168.2.19.</p> <p>The match can be inverted by adding an <code>!</code> sign. The above example would then look like <code>-m iprange ! --src-range 192.168.1.13-192.168.2.19</code>, which matches every single IP address, except the ones specified.</p>
Match	--dst-range
Example	<code>iptables -A INPUT -p tcp -m iprange --dst-range 192.168.1.13-192.168.2.19</code>
Explanation	The <code>--dst-range</code> works exactly the same as the <code>--src-range</code> match, except that it matches destination IP's instead of source IP's.

4.2.2.2

Length Match

The length match is used to match packets based on their length.

Use the length match if you want to do either of the following:

- Limiting packet length for special reasons.
- Blocking ping-of-death-like behaviors.

Table 13 Length Match

Match	--length
Example	<code>iptables -A INPUT -p tcp -m length --length 1400:1500</code>
Explanation	<p>The example <code>--length</code> matches all packets with a length between 1400 and 1500 bytes.</p> <p>To invert this match, add an <code>!</code> sign. For example, <code>-m length ! --length 1400:1500</code>.</p> <p>To only match a specific length, remove the <code>:</code> sign and onwards. For example: <code>-m length --length 1400</code>.</p>



4.2.2.3 Multiport Match

The multiport match extension is used to specify multiple destination ports and port ranges. Otherwise, you must use multiple rules of the same type to match different ports.

Table 14 Multiport Match Options

Match	--source-port
Example	<code>iptables -A INPUT -p tcp -m multiport --source-port 22,53,80,110</code>
Explanation	<p>This match matches multiple source ports.</p> <p>A maximum of 15 separate ports can be specified. The ports must be comma delimited, as in the above example.</p> <p>The match can only be used with the <code>-p tcp</code> or <code>-p udp</code> matches and in this case, it is an enhanced version of the normal <code>--source-port</code> match.</p>
Match	--destination-port
Example	<code>iptables -A INPUT -p tcp -m multiport --destination-port 22,53,80,110</code>
Explanation	<p>This match is used to match multiple destination ports.</p> <p>It works exactly the same way as the above mentioned <code>--source-port</code> match, except that it matches destination ports.</p> <p>It also has a maximum of 15 ports and can only be used with <code>-p tcp</code> and <code>-p udp</code>.</p>
Match	--port
Example	<code>iptables -A INPUT -p tcp -m multiport --port 22,53,80,110</code>
Explanation	<p>This match⁽¹⁾ extension can be used to match packets based both on their destination port and source port.</p> <p>It works the same way as the <code>--source-port</code> and <code>--destination-port</code> matches above.</p> <p>It can take a maximum of 15 ports and can only be used with <code>-p tcp</code> and <code>-p udp</code>.</p>

(1) The `--port` match only matches packets coming in from and going to the same port. For example, port 80 to port 80, port 110 to port 110 and so on.

4.3 Iptables Targets

The target tells the rule what to do with a packet that is a perfect match with the match section of the rule.



This section describes the basic targets.

4.3.1 ACCEPT Target

This target needs no further options. **ACCEPT** is specified as the target as soon as the match specification for a packet is fully satisfied.

The rule is accepted and does not continue traversing the current chain or any other ones in the same table.

Note: However, a packet that was accepted in one chain can still travel through chains within other tables, and can still be dropped there.

This target has nothing special and does not work with other options. To use this target, we simply specify `-j ACCEPT`.

4.3.2 DROP Target

This target drops packets dead and does not carry out any further processing. A packet that matches a rule perfectly is then Dropped and blocked.

Note: This action has unwanted effects in certain cases since it can leave dead sockets around on either host. When experiencing the likely cases, a better solution is to use the **REJECT** target, especially when you want to block port scanners from getting too much information, such as on filtered ports and so on.

If a packet has the **DROP** action taken on it in a subchain, the packet is not processed in any of the main chains either in the present or in any other table. The packet is totally dead.

4.3.3 LOG Target

This target is designed to log detailed information for packets.

Table 15 LOG Target Options

Option	--log-level
Example	<code>iptables -A FORWARD -p tcp -j LOG --log-level debug</code>



Explanation	<p>This option tells iptables and syslog which log level to use.</p> <p>For a complete list of log levels, see syslog.conf Manual.</p> <p>General Log levels (or priorities) are listed below:</p> <ul style="list-style-type: none">• Debug• Info• Notice• Warning• Warn• Err• Error• Crit• Alert• Emerg• Panic <p>The keyword error is the same as err, warn is the same as warning and panic is the same as emerg⁽¹⁾.</p> <p>The priority defines the severity of the message being logged. All messages are logged through the kernel facility. Steps are listed below:</p> <ul style="list-style-type: none">• Setting kern.=info /var/log/iptables in your syslog.conf file.• Letting all your LOG messages in iptables use log level info make all messages appear in the /var/log/iptables file⁽²⁾
Option	--log-prefix
Example	iptables -A INPUT -p tcp -j LOG --log-prefix "INPUT packets"
Explanation	<p>This option tells iptables to prefix all log messages with a specific prefix, which can then easily be combined with grep or other tools to track specific problems and output from different rules.</p> <p>The prefix is up to 29 letters long, including white-spaces and other special symbols.</p>
Option	--log-tcp-sequence
Example	iptables -A INPUT -p tcp -j LOG --log-tcp-sequence



Explanation	<p>This option logs the TCP Sequence numbers, together with the log messages.</p> <p>The TCP Sequence numbers are special numbers that identify each packet and where it fits into a TCP sequence, as well as how the stream is reassembled⁽³⁾.</p>
Option	--log-tcp-options
Example	<code>iptables -A FORWARD -p tcp -j LOG --log-tcp-options</code>
Explanation	<p>The option logs the different options from the TCP packet headers and is valuable when trying to debug what can go wrong, or what has actually gone wrong.</p> <p>This option does not take any variable fields or anything like that, just as most of the LOG options don't.</p>
Option	--log-ip-options
Example	<code>iptables -A FORWARD -p tcp -j LOG --log-ip-options</code>
Explanation	<p>The option will log most of the IP packet header options. This option works exactly the same as the <code>--log-tcp-options</code> option, but instead, it works on the IP options. These logging messages are valuable when trying to debug or track specific culprits, as well as for debugging - in the same way as the previous option.</p>

(1) All three of these are deprecated, in other words do not use `error`, `warn` and `panic`.

(2) If other parts of the kernel use the `info` priority, other messages appear here as well.

(3) This option causes a security risk if the logs are readable by unauthorized users, or by the world for that matter.

4.3.4 REJECT Target

This target works basically the same as the `DROP` target, but it also sends back an error message to the host sending the packet that was blocked.



Table 16 REJECT Target Options

Option	--reject-with
Example	iptables -A FORWARD -p TCP --dport 22 -j REJECT --reject-with tcp-reset
Explanation	<p>This option tells the REJECT target what response to send to the host that sent the packet that we are rejecting. See below process:</p> <ul style="list-style-type: none">• We get a packet that matches a rule in which we have specified this target.• Our host first of all sends the associated reply.• The packet is then be dropped dead. ⁽¹⁾ <p>The following reject types are currently valid:</p> <ul style="list-style-type: none">• icmp-net-unreachable• icmp-host-unreachable• icmp-port-unreachable• icmp-proto-unreachable• icmp-net-prohibited• icmp-host-prohibited <p>The default error message is to send a port-unreachable to the host. All of the above are ICMP error messages and can be set as you wish.</p> <p>You can find further information on their various purposes in the appendix ICMP types. Finally, there is one more option called tcp-reset, which may only be used together with the TCP protocol. The tcp-reset option will tell REJECT to send a TCP RST packet in reply to the sending host. TCP RST packets are used to close open TCP connections gracefully.</p> <p>For more information about the TCP RST, see RFC 793 - Transmission Control Protocol RFC 793 - Transmission Control Protocol.</p> <p>This is mainly useful for blocking ident probes which frequently occur when sending mail to broken mail hosts that do not otherwise accept your mail.</p>

(1) Just as the Drop option that drops it.



Reference List

Ericsson Documents

- [1] Trademark Information
- [2] Typographic Conventions
- [3] Glossary of Terms and Acronyms