IPtables

CHEF-KOCH edited this page on Jan 15, 2016 · 15 revisions

Index

Description

> Originally, the most popular firewall/NAT package running on Linux/Android was ipchains, but it had a number of shortcomings. To rectify this, the Netfilter organization decided to create a new product called iptables, giving it such improvements as:

- Better integration with the Linux kernel with the capability of loading iptables specific kernel modules designed for improved speed and reliability.
- Stateful packet inspection. This means that the firewall keeps track of each connection passing through it and in certain cases will view the contents of data flows in an attempt to anticipate the next action of certain protocols. This is an important feature in the support of active FTP and DNS, as well as many other network services.
- Filtering packets based on a MAC address and the values of the flags in the TCP header. This is helpful in preventing attacks using malformed packets and in restricting access from locally attached servers to other networks in spite of their IP addresses.
- System logging that provides the option of adjusting the level of detail of the reporting.
- Better network address translation.
- A rate limiting feature that helps iptables block some types of denial of service (DoS) attacks.

Considered a faster and more secure alternative to ipchains, iptables has become the default firewall package installed under Linux and Android OS.

Nftables will replace it some day, but is not merged with AFWall+ at the moment.

Netfilter is generally a part of the Linux kernel. The IP packet filter rules in the Linux kernel are being configured by the user space command line tools of netfilter: iptables, ip6tables, ebtables, arptables and ipset (optional).

❗ An official (not outdated) documentation can be found via clone here, see ip6tables.8.in & iptables.8.in. Also look here for Android kernel patches. ❗

Changelog

Latest iptables official release: **iptables-1.4.7** -> 04-Dec-2015

▾ Pages 28

Find a Page...

Home
Advertisements blocking
Android kernel traffic
Apps leak private user data during boot
BusyBox
Contributors
CustomScripts
DNS
Error codes
FAQ
Google Play services and other special cases
HOWTO blocking WhatsApp
HOWTO Compile AFWall
HOWTO Compiling busybox
HOWTO Compiling iptables
Show 13 more pages...

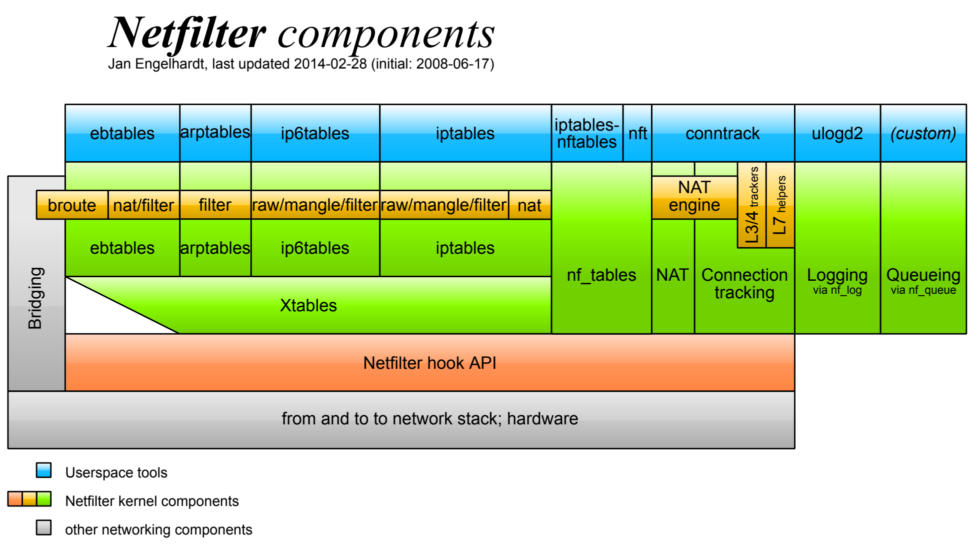
Clone this wiki locally
https://github.com/ukanth/a
Clone in Desktop

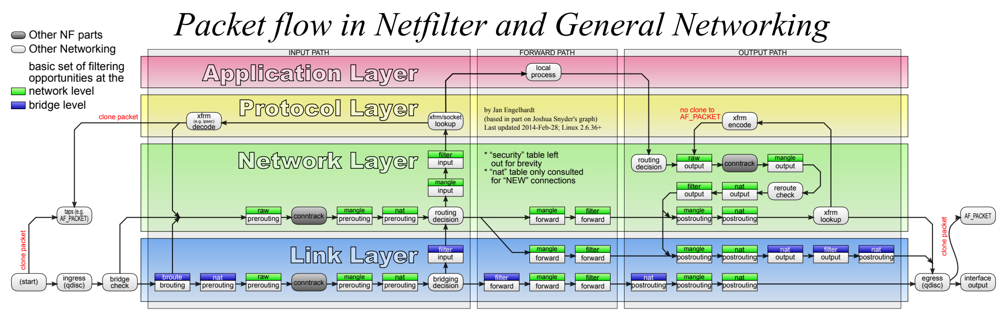
The official iptables changelogs, patches and binaries can always be found here. *These binaries are not optimized for Android!*

Remember: Some apps are bundled with there own iptables (SandroProxy, Orbot, OrWall,[...]) and some have no option to toggle between the system ones and there own (AFWall+ does have this option!) so it's necessary if you using one of this apps, to ensure it does not interference with AFWall+. The best solution to handle such problematic things is to replace (if outdated) the system ones and only use these to avoid conflicts between two or more separate iptables.

Explanation

Please have a look at this most excellent scheme: Netfilter components & Netfilter Packet Flow by Jan Engelhardt to understand how a packet traverses netfilter. The green stuff is the domain of iptables and ip6tables, while the blue stuff is being handled by ebtables.





Extensions

All available iptables-extensions can be found over here.

Important

Please never ask basic questions how to handle or work with iptables if you not already have take a deeper look in the docs. AFWall+ is already a very easy to use GUI, it's not perfect, but we are working on it to improve it step-by-step. If you like to working with custom scripts you definitely need some basic knowledge, but for most users the preset GUI options that is given is almost enough to control the important parts. So please take care about the fact we don't have time to explain anything to every new user that ask the same question multiple times, that's why we have the Wiki - to answer the most questions.

Androids default iptables

The following rules are the default AOSP rules, the order of execution plays an important role. These rules are been executed each time netd and the interface get an 'changed' command.

```
// Default Whitelist rules to drop all traffic
if (ftype == WHITELIST) {
res |= execIptables(V4V6, "-A", LOCAL_INPUT, "-j", "DROP", NULL);
res |= execIptables(V4V6, "-A", LOCAL_OUTPUT, "-j", "REJECT", NULL);
res |= execIptables(V4V6, "-A", LOCAL_FORWARD, "-j", "REJECT", NULL);

// Default chains executed on something changed to not allow holes
execIptablesSilently(target, "-t", table, "-D", parentChain, "-j", *childChain, NULL);
execIptablesSilently(target, "-t", table, "-F", *childChain, NULL);
execIptablesSilently(target, "-t", table, "-X", *childChain, NULL);
execIptables(target, "-t", table, "-N", *childChain, NULL);
execIptables(target, "-t", table, "-A", parentChain, "-j", *childChain, NULL);

// Explanation:
-D to delete any pre-existing jump rule (removes references that would prevent -X from
working)
-F to flush any existing chain
-X to delete any existing chain
-N to create the chain (also for each child process)
-A to append the chain to parent

// Blacklist
If no rules are activated or been executed then Blacklist Mode will be automatically
used.

// ndc firewall enable|disable
Activates or deactivates the entire firewall

//Block a specific UID (example: 2000)
ndc firewall set_uid_rule 2000 block
```

Basics

Working with the basics can be useful if you having troubles getting something to work. So please compare your problems with the stock source (VpnService.java, ConnectivityService.java, NetworkManagementService.java, NativeDaemonConnector.java, CommandListener.cpp, NetdConstants.cpp, etc.) and look if your configuration is different from it. You also need to know that the result from the output that was given may can differ, decency which OS version, kernel (iptables) and config you are using.

The most important source stuff for AFWall+ or general networking seems:

- netd
- bluetooth
- iptables
- dhcpcd
- dnsmasq

- iputils [optional]
- ipsec-tools [optional]
- iproute2
- jmdns
- mdnsresponder [optional]
- netcat [optional]
- netperf [optional]
- openvpn
- openssh
- ping
- ping6 [optional because IPv6]
- ppp
- okhttp
- tcpdump [optional, but most ROMs including this already]
- wpa_supplicant [there is also _6 and _8 depending which device]
- Ims
- VoIP
- WiFi
- Performing Network Operations Overview
- ... all other stuff isn't network related and *unimportant* to us.

`ip rule list` or `iptables -t nat --list-rules`

root@Android:/	# iptables -t nat --list-rules
-P	PREROUTING ACCEPT
-P	INPUT ACCEPT
-P	OUTPUT ACCEPT
-P	POSTROUTING ACCEPT
-N	natctrl_nat_POSTROUTING
-N	oem_nat_pre
-N	st_nat_POSTROUTING
-A	PREROUTING -j oem_nat_pre
-A	POSTROUTING -j natctrl_nat_POSTROUTING
-A	POSTROUTING -j st_nat_POSTROUTING
-A	natctrl_nat_POSTROUTING -o <*interface_name*> -j MASQUERADE

`ip6tables -L st_filter_OUTPUT`

`Chain st_filter_OUTPUT (1 references)`

target	prot opt	source	destination	.....
REJECT	all	anywhere	anywhere	mark match 0x3d reject-with icmp6-port-unreachable

Option	Description	Valid states are

Option	Description	Valid states are
-A	Append this rule to a rule chain.	INPUT, FORWARD and OUTPUT
-L	List the current filter rules.	
-mconntrack	Allow filter rules to match based on connection state. Permits the use of the --ctstate option.	
--ctstate	Define the list of states for the rule to match on.	NEW, RELATED, ESTABLISHED or INVALID
-m limit	Require the rule to match only a limited number of times. Allows the use of the --limit option. Useful for limiting logging rules.	
--limit	The maximum matching rate, given as a number followed by "/second", "/minute", "/hour", or "/day" depending on how often you want the rule to match.	If this option is not used and -m limit is used, the default is "3/hour".
-p	The connection protocol used.	eth0, tun0, ppp0, etc. As a special case, an interface name ending with a `+`' will match all interfaces. The interface name can be preceded by a `!`' with spaces around it, to match a packet which does not match the specified interface(s), eg -i ! ppp+.
--dport	The destination port(s) required for this rule. A single port may be given, or a range may be given as start:end, which will match all ports from start to end, inclusive.	
-j	Jump to the specified target. By default, iptables allows four targets	ACCEPT, REJECT, DROP or LOG
--log-prefix	When logging, put this text before the log message. Use double quotes around the text to use.	
--log-level	Log using the specified syslog level. 7 is a good choice unless you specifically need something else.	
-i	Only match if the packet is coming in on the specified interface.	

Option	Description	Valid states are
-I	Inserts a rule. Takes two options, the chain to insert the rule into, and the rule number it should be.	-I INPUT 5 would insert the rule into the INPUT chain and make it the 5th rule in the list.
-v	Display more information in the output. Useful for if you have rules that look similar without using -v.	
-s --source	address[/mask] source specification	
-i --destination	address[/mask] destination specification	
-o --out-interface	output name[+] network interface name	[+] for wildcard

The State Match	Description
NEW	A packet which creates a new connection.
ESTABLISHED	A packet which belongs to an existing connection.
RELATED	A packet which is related to, but not part of, an existing connection, such as an ICMP error, or (with the FTP module inserted), a packet establishing an ftp data connection.
INVALID	A packet which could not be identified for some reason: this includes running out of memory and ICMP errors which don't correspond to any known connection. Generally these packets should be dropped.

Useful links

- iptables | Wikipedia.org
- netfilter/iptables project homepage | The netfilter.org project
- netfilter/iptables documentation | The netfilter.org project
- iptables developer page | netfilter.org
- iptables tutorial by Oskar Andreasson | Frozentux.net
- Quick HOWTO : Ch14 : Linux Firewalls Using iptables | LinuxHomeNetworking.com
- Iptables HowTo | Help.Ubuntu.com
- Connection Tracking | Rigacci.org
- How to Setup a Linux Firewall with PPPoE/NAT/iptables | akadia.com
- German introduction: Private Feuerwände | Linux-Community.de
- What is the difference between AVM Firewall and Netfilter? | Freetz.org
- Wlan traffic seems to bypass iptables | Forum OpenWRT.org