Linux Firewall Wizardry

- By Nemus

The internet and your server

So then what do you protect your server with if you don't have a firewall in place?

NetFilter / Iptables
http://www.netfilter.org

Iptables define by netfiler

Netfilter is a set of hooks inside the Linux kernel that allows kernel modules to register callback functions with the network stack. A registered callback function is then called back for every packet that traverses the respective hook within the network stack.

Iptables is a generic table structure for the definition of rulesets. Each rule within an IP table consists of a number of classifiers.

Tables and Chains

- iptables contains multiple tables
  - (which are netfilter modules).
  - Think of tables as a place to put chains.
- Tables contain multiple chains.
  - Chains can be built-in or user-defined.
  - Chains contain rules.
- Chains contain rules.
- Rules are defined for packets.
- iptables -> Tables -> Chains -> Rules.
- iptables has the following 4 built-in tables.

Filter Table

- Filter Table (DEFAULT TABLE!)
  - Filter is default table for iptables and the most commonly used table.
    - (iptables -t filer -list or just iptables -L)
  - The filter table has the following chains
    - INPUT chain – Incoming to firewall.
      - For packets who's destination is the server.
    - OUTPUT chain – Outgoing from firewall.
      - For packets generated locally and going out of the local server.
    - FORWARD chain – Packet's that pass through the server.
      - For packets routed through the local server.

Filter Table Example

```
iptables -L -n -x -v

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
    pkts      bytes target     prot opt in     out     source               destination
    4714  1147392 ACCEPT     all  --  eth0   *       0.0.0.0/0            0.0.0.0/0
    9998  3016856 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
       0        0 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
    pkts      bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 12060 packets, 3876912 bytes)
    pkts      bytes target     prot opt in     out     source               destination
```

NAT Table

- Iptable's NAT table has the following built-in chains.
  - PREROUTING chain – Alters packets before routing.
    - Packet translation happens immediately after the packet enter the system and is done before being processed by the routing table.
      - Helps to translate the destination ip address of the packets to something that matches the routing on the local server or subnet.
        - used for DNAT (destination NAT).
  - POSTROUTING chain – Alters packets after routing.
    - Packet translation happens when the packets are leaving the system
      - Used to translate the source ip address of the packets to something that matches the routing on the destination server.
        - used by SNAT (source NAT).
  - OUTPUT chain – NAT for locally generated packets on the firewall.

Nat Table Example

iptables -t nat -L -n -x -v

Chain PREROUTING (policy ACCEPT 678523 packets, 74334701 bytes)

pkts	bytes	target	prot	opt	in	out	source	destination

Chain INPUT (policy ACCEPT 43265 packets, 4868594 bytes)

pkts	bytes	target	prot	opt	in	out	source	destination

Chain OUTPUT (policy ACCEPT 1185036 packets, 139614154 bytes)

pkts	bytes	target	prot	opt	in	out	source	destination

Chain POSTROUTING (policy ACCEPT 1015193 packets, 86055073 bytes)

pkts	bytes	target	prot	opt	in	out	source	destination
62114	4441616	SNAT	all	--	*	*	172.x.x.0/24	0.0.0.0/0	to:x.x.x.135

Mangle Table

- Iptables's Mangle table is for specialized packet alteration.
  - Mainly used to set QOS bits in the TCP header.
    - Can be used to set pick me first in packet header.
- Mangle table has the following built-in chains.
  - PREROUTING chain
  - OUTPUT chain
  - FORWARD chain
  - INPUT chain
  - POSTROUTING chain

Mangle Table Example

iptables -t mangle -L -n -x -v

Chain PREROUTING (policy ACCEPT 131 packets, 14768 bytes)

pkts	bytes	target	prot	opt	in	out	source	destination

Chain INPUT (policy ACCEPT 131 packets, 14768 bytes)

pkts	bytes	target	prot	opt	in	out	source	destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)

pkts	bytes	target	prot	opt	in	out	source	destination

Chain OUTPUT (policy ACCEPT 92 packets, 18196 bytes)

pkts	bytes	target	prot	opt	in	out	source	destination

Chain POSTROUTING (policy ACCEPT 91 packets, 17868 bytes)

pkts	bytes	target	prot	opt	in	out	source	destination

Raw table

- Iptable's Raw table is for configuration exceptions Raw table has the following built-in chains.
  - PREROUTING chain
  - OUTPUT chain

Raw Table Example

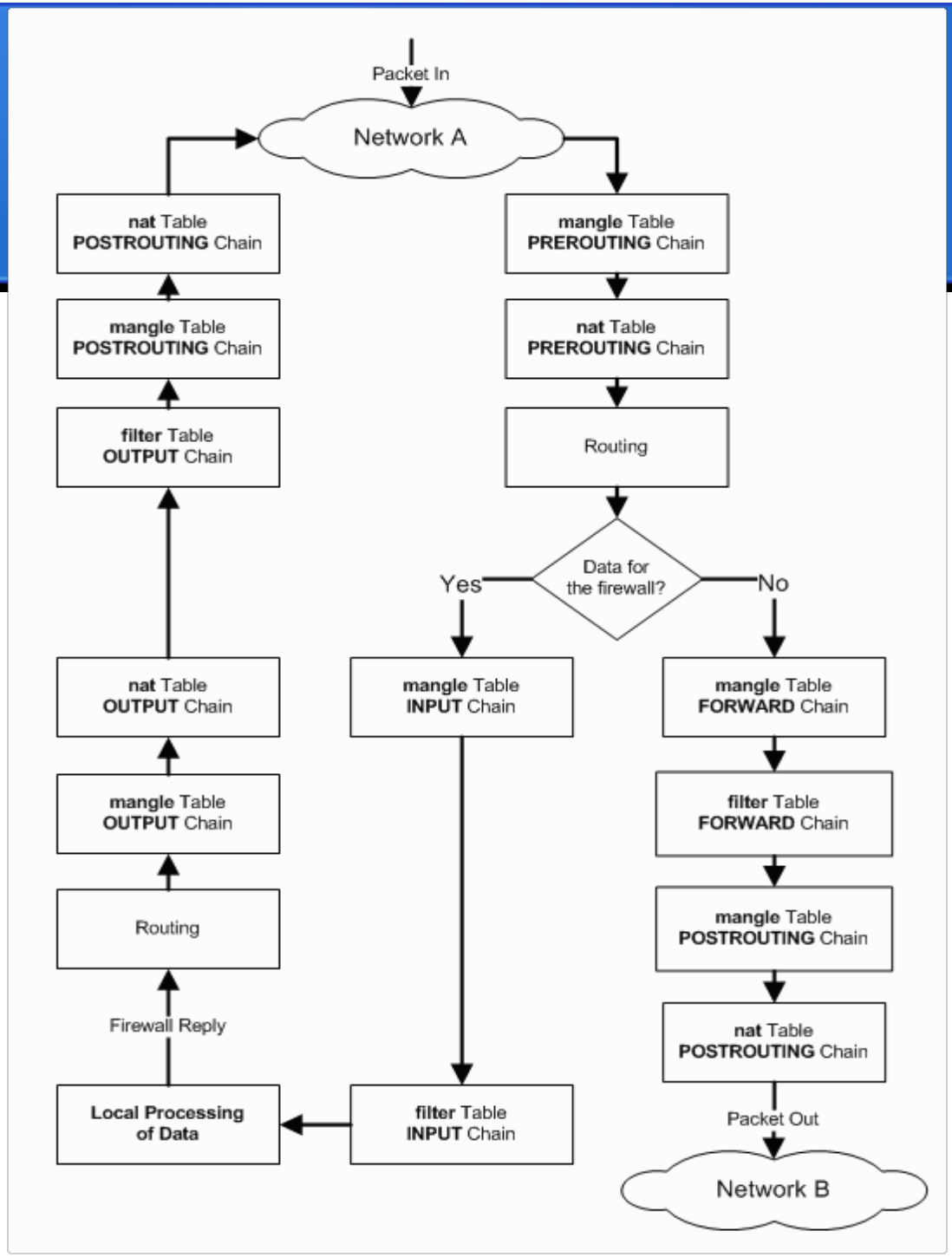
iptables -t raw -L -n -x -v

Chain PREROUTING (policy ACCEPT 108 packets, 11590 bytes)

pkts	bytes	target	prot	opt	in	out	source	destination

Chain OUTPUT (policy ACCEPT 72 packets, 11284 bytes)

pkts	bytes	target	prot	opt	in	out	source	destination



http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch14_:_Linux_Firewalls_Using_iptables#.UOZp9nd1imk

Target Values

- Following are the possible target values
  - ACCEPT
    - Firewall will accept the packet.
  - DROP
    - Firewall will drop the packet.
  - Log
    - Logs packet to syslog
  - REJECT
    - Drops packet and sends response back to sender
  - RETURN
    - Firewall will stop executing the next set of rules in the current chain for this packet. The control will be returned to the calling chain.

Matches

Every rule in iptables has a criteria and a target you can match packet data using the following matches

--source (-s) Match on a source IP address or network

--destination (-d) Match on a destination IP address or network

--protocol (-p) Match on an IP value

--in-interface (-i) Input interface (eth0,eth1,etc..)

--out-interface (-o) Output interface (eth0,eth1,etc..)

--state Match on a set of ip packect connection states

--string Match on a sequence of application layer data bytes

--comment Associate up to 256 bytes of comment data

Key Points

- Rules contain a criteria and a target.
- If the criteria is matched, it goes to the rules specified in the target (or) executes the special values mentioned in the target.
- If the criteria is not matached, it moves on to the next rule.

THOU SHALL NOT PASS

Now an example of iptables wizardry!

Example

http://www.cipherdyne.org/LinuxFirewalls/ch01/

#!/bin/sh

IPTABLES=/sbin/iptables (path to iptables)

IP6TABLES=/sbin/ip6tables (path to ipables6)

MODPROBE=/sbin/modprobe (path to modprobe)

INT_NET=192.168.10.0/24 (nated internal network)

INT_INTF=eth1 (input interface bash variable lan)

EXT_INTF=eth0 (exit interface bash variable wan)

Example

flush existing rules and set chain policy setting to DROP

echo "[+] Flushing existing iptables rules..."

$IPTABLES -F (remove rules form chains in filter table)

$IPTABLES -F -t nat #(remove rules from chains in nat table)

$IPTABLES -X #(delete user define chains)

$IPTABLES -P INPUT DROP #(set default policy to drop input)

$IPTABLES -P OUTPUT DROP #(set default policy to drop ouput)

$IPTABLES -P FORWARD DROP #(set default policy to drop all packets passing through firewall)

Drop IPV6 Packets

```
echo "[+] Disabling IPv6 traffic..."
$IP6TABLES -P INPUT DROP
$IP6TABLES -P OUTPUT DROP
$IP6TABLES -P FORWARD DROP
```

Example rules 2
Input Chain

```
### state tracking rules

$IPTABLES -A INPUT -m conntrack --ctstate INVALID -j LOG --log-prefix "DROP INVALID " --log-ip-options –
      log-tcp-options #(log invalid state packets)

$IPTABLES -A INPUT -m conntrack --ctstate INVALID -j DROP #(drop invalid packets)

$IPTABLES -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

### anti-spoofing rules (remember constant from above)

$IPTABLES -A INPUT -i $INT_INTF ! -s $INT_NET -j LOG --log-prefix "SPOOFED PKT "

$IPTABLES -A INPUT -i $INT_INTF ! -s $INT_NET -j DROP

### ACCEPT rules

$IPTABLES -A INPUT -i $INT_INTF -p tcp -s $INT_NET --dport 22 -m conntrack --ctstate NEW -j ACCEPT

$IPTABLES -A INPUT -p icmp --icmp-type echo-request -j ACCEPT

### default INPUT LOG rule

$IPTABLES -A INPUT ! -i lo -j LOG --log-prefix "DROP " --log-ip-options --log-tcp-options

### make sure that loopback traffic is accepted

$IPTABLES -A INPUT -i lo -j ACCEPT
```

Example rules 3
Output Chain

```
###### OUTPUT chain ######

### state tracking rules

$IPTABLES -A OUTPUT -m conntrack --ctstate INVALID -j LOG --log-prefix "DROP INVALID " --log-ip-options --log-tcp-options

$IPTABLES -A OUTPUT -m conntrack --ctstate INVALID -j DROP

$IPTABLES -A OUTPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

### ACCEPT rules for allowing connections out

$IPTABLES -A OUTPUT -p tcp --dport 21 -m conntrack --ctstate NEW -j ACCEPT # allow ftp out

$IPTABLES -A OUTPUT -p tcp --dport 22 -m conntrack --ctstate NEW -j ACCEPT # allow ssh out bound

$IPTABLES -A OUTPUT -p tcp --dport 25 -m conntrack --ctstate NEW -j ACCEPT # allow telnet out

$IPTABLES -A OUTPUT -p tcp --dport 43 -m conntrack --ctstate NEW -j ACCEPT # allow whois out

$IPTABLES -A OUTPUT -p tcp --dport 80 -m conntrack --ctstate NEW -j ACCEPT # allow http out

$IPTABLES -A OUTPUT -p tcp --dport 443 -m conntrack --ctstate NEW -j ACCEPT # allow https out

$IPTABLES -A OUTPUT -p tcp --dport 53 -m conntrack --ctstate NEW -j ACCEPT #allow dns out for tcp

$IPTABLES -A OUTPUT -p udp --dport 53 -m conntrack --ctstate NEW -j ACCEPT #allow dns out for tcp

$IPTABLES -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT
```

Example rules 4
Forward Chain

```
### state tracking rules

$IPTABLES -A FORWARD -m conntrack --ctstate INVALID -j LOG --log-prefix "DROP INVALID " --log-ip-options
      --log-tcp-options

$IPTABLES -A FORWARD -m conntrack --ctstate INVALID -j DROP

$IPTABLES -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT


### anti-spoofing rules

$IPTABLES -A FORWARD -i $INT_INTF ! -s $INT_NET -j LOG --log-prefix "SPOOFED PKT "

$IPTABLES -A FORWARD -i $INT_INTF ! -s $INT_NET -j DROP
```

Forward Chain Continued

```
### ACCEPT rules
$IPTABLES -A FORWARD -p tcp -i $INT_INTF -s $INT_NET --dport 21 -m conntrack --ctstate NEW -j ACCEPT
$IPTABLES -A FORWARD -p tcp -i $INT_INTF -s $INT_NET --dport 22 -m conntrack --ctstate NEW -j ACCEPT
$IPTABLES -A FORWARD -p tcp -i $INT_INTF -s $INT_NET --dport 25 -m conntrack --ctstate NEW -j ACCEPT
$IPTABLES -A FORWARD -p tcp -i $INT_INTF -s $INT_NET --dport 43 -m conntrack --ctstate NEW -j ACCEPT
$IPTABLES -A FORWARD -p tcp --dport 80 -m conntrack --ctstate NEW -j ACCEPT
$IPTABLES -A FORWARD -p tcp --dport 443 -m conntrack --ctstate NEW -j ACCEPT
$IPTABLES -A FORWARD -p tcp --dport 53 -m conntrack --ctstate NEW -j ACCEPT
$IPTABLES -A FORWARD -p udp --dport 53 -m conntrack --ctstate NEW -j ACCEPT
$IPTABLES -A FORWARD -p icmp --icmp-type echo-request -j ACCEPT

### default LOG rule
$IPTABLES -A FORWARD ! -i lo -j LOG --log-prefix "DROP " --log-ip-options --log-tcp-options
```

Example Rules 5 Nat Table

```
###### NAT rules ######

echo "[+] Setting up NAT rules..."

$IPTABLES -t nat -A PREROUTING -p tcp --dport 80 -i $EXT_INTF -j DNAT --to 192.168.10.3:80

$IPTABLES -t nat -A PREROUTING -p tcp --dport 443 -i $EXT_INTF -j DNAT --to 192.168.10.3:443

$IPTABLES -t nat -A PREROUTING -p udp --dport 53 -i $EXT_INTF -j DNAT --to 192.168.10.4:53

$IPTABLES -t nat -A POSTROUTING -s $INT_NET -o $EXT_INTF -j MASQUERADE
```

Enabling Fowarding

```
echo "[+] Enabling IP forwarding..."
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Command Over View

- iptables-save > /etc/sysconfig/iptables
  - Save iptables setup
- iptables-restore -c < /etc/sysconfig/iptables
  - Restores iptables
- Iptables -L -n -v -x
  - List filter table with out doing dns lookup
    - (quick view)
    - V for verbose ()
    - X show packet count
    - Should be your default iptables command
- iptables -L -n –line-number
  - Shows Line Numbers
- Iptables -F
  - Flushes filter table
- iptables -D INPUT 10
  - Delete rule with line number 10 in input chain
- iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT
  - Append to input chain
    - Adds to the bottom
- iptables -I INPUT 1 -i lo -j ACCEPT
  - Inserts above chain rule set (opposite of append)
    - Add to the top of the rule chain
- iptables -P INPUT DROP
  - Sets default policy for chain

Operating System Defense

Disable routing triangulation. Respond to queries out

the same interface, not another. Helps to maintain state

Also protects against IP spoofing

net/ipv4/conf/all/rp_filter = 1

Enable logging of packets with malformed IP addresses

net/ipv4/conf/all/log_martians = 1

Disable redirects

net/ipv4/conf/all/send_redirects = 0

Disable source routed packets

net/ipv4/conf/all/accept_source_route = 0

Disable acceptance of ICMP redirects

net/ipv4/conf/all/accept_redirects = 0

Turn on protection from Denial of Service (DOS) attacks

net/ipv4/tcp_syncookies = 1

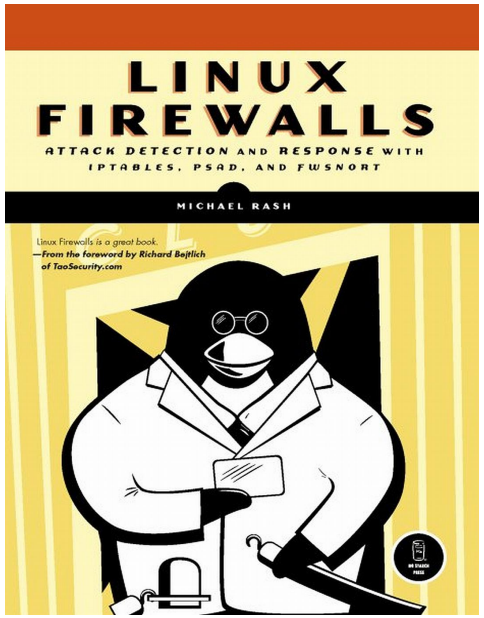
Disable responding to ping broadcasts

net/ipv4/icmp_echo_ignore_broadcasts = 1

http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch14_:_Linux_Firewalls_Using_iptables#.UOZ6lHd1imk

Great Resources

- http://www.cyberciti.biz/tips/linux-iptables-examples.html
- http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch14_:_Linux_Firewalls_Using_iptables#.UOZ6lHd1imk
- http://www.cipherdyne.org/LinuxFirewalls/ch01/
- http://www.thegeekstuff.com/2011/06/iptables-rules-examples/
- Linux Firewalls: Attack Detection and Response with iptables, psad, and fwsnort
- http://www.cipherdyne.org/LinuxFirewalls/



Windows People

- Iptables on Windows?
  - netsh firewall (deprecated after XP and 2003)
  - netsh advfirewall (Vista, 7, and 2008)
- http://serverfault.com/questions/207620/windows-equivalent-of-iptables
- http://www.activexperts.com/network-monitor/windowsmanagement/scripts/networking/windowsfirewall/
- http://technet.microsoft.com/en-us/library/cc771920%28WS.10%29.aspx
- http://support.microsoft.com/kb/947709
- http://itblog.gr/213/configuring-windows-firewall-from-the-command-line/

Windows Block Subnet

- netsh advfirewall firewall add rule name="HTTP" protocol=TCP localport=80 action=block dir=IN remoteip=x.x.x.x/24
- netsh advfirewall firewall add rule name="HTTP" protocol=TCP localport=80 action=allow dir=IN remoteip=x.x.x.x
- netsh advfirewall set domainprofile state on
  - Windows 7
- netsh advfirewall set domainprofile state off
  - Windows 7

Level Up