ip-tables

Packet filtering under Linux

Literatur:

Iptab_howtos – IPtables_bas.htm, ….

Packet Filter and Linux

- A packet filter is a piece of software which looks at the *header* of packets as they pass through.
- It might decide to **deny** the packet, **accept** the packet, or **reject** the packet (=deny and tell the source)
- Under Linux, packet filtering is built into the kernel
- The iptables tool inserts and deletes rules from the kernel’s packet filtering table and other tables.
- iptables is a replacement for ipfwadm and ipchains.

tables

filter	**default**, mainly used for filtering packets
mangle	mangling packets, i.e., changing TOS and so on.
nat	used for Source Network Address Translation

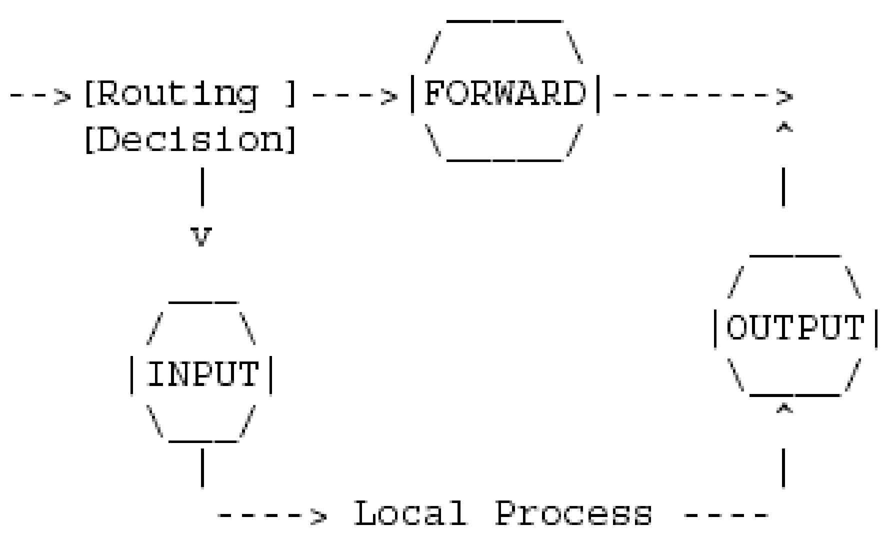
Filtertabelle
- Firewall
- Paketfilter

Mangle
- Veränderung von Paketen

NAT
- für Netzwerkaufbau

How Packets Traverse The Filters

The kernel starts with three lists of rules; these lists are called **firewall chains** or just **chains**. The three chains are called **INPUT**, **OUTPUT** and **FORWARD**.



What‘s a chain

- A chain is a checklist of **rules**.
- Each rule says ‘if the packet header looks like this, then here’s what to do with the packet’.
- If the rule doesn’t match the packet, then the next rule in the chain is consulted.
- Finally, if there are no more rules to consult, then the kernel looks at the chain **policy** to decide what to do.
- In a security-conscious system, this policy usually tells the kernel to DROP the packet.

Üblicherweise nicht auf Drop bei Linux (Ubuntu) gestellt.

Using iptables

- Create a new chain (-N) # iptables -N test
- Delete an empty chain (-X). # iptables -X test
- Change the policy for a built-in chain. (-P). # iptables -P INPUT DROP
- List the rules in a chain (-L). # iptables -L FORWARD
- Flush the rules out of a chain (-F). # iptables -F FORWARD
- Zero the packet and byte counters on all rules in a chain (-Z).
- Append a new rule to a chain (-A).
- Insert a new rule at some position in a chain (-I).
- Replace a rule at some position in a chain (-R).
- Delete a rule at some position in a chain (-D).

Policies

- only built-in chains (INPUT, OUTPUT, FORWARD) have policies.
- The policy can be either ACCEPT or DROP
- usually the default policy is ACCEPT so it‘s recommended to change the policy
- In a security-conscious system, this policy usually tells the kernel to DROP the packet.
    - # iptables -P INPUT DROP
    - # iptables -P FORWARD DROP

Operations on a Single Rule

- append (-A), delete (-D), Insert (-I), replace (-R)
- each rule specifies a set of conditions the packet must meet, and what to do if it meets them
- Append
  - # iptables –A *chain addresses,protocols,ports..* –j *target*

    ```
    # iptables -A INPUT -s 127.0.0.1 -p icmp -j DROP
    ```
- Delete
  - by number

    ```
    # iptables -D INPUT 1
    ```
  - by inversion

    ```
    # iptables -D INPUT -s 127.0.0.1 -p icmp -j DROP
    ```

Filtering Specifications

- **Specifying Source and Destination IP Addresses**

  source -s, destination –d   IP-Addresses
  - -s 172.16.57.251   = single host
  - -d 172.16.56.0/23 or 172.16.56.0/255.255.254.0 = net
  - -s 0/0 = any
- **Specifying Inversion with !**
  - ! 172.16.57.251
- **Specifying Protocol with –p**
  - -p ! tcp   by name, inversion as option
  - -p 6   or by number

- **Specifying an Interface**
  - -i = input interface, -o = output interface

    -i eth0   -o eth1
  - only packets traversing the FORWARD chain have both an input and output interface.
  - it is perfectly legal to specify an interface that currently does not exist; the rule will not match anything until the interface comes up.
  - as a special case, an interface name ending with a ‘+’ will match all interfaces which begin with that string.

    ppp+
  - the interface name can be preceded by a ‘!’ to match a packet which does NOT match the specified interface(s).

TCP Extensions

- iptables is **extensible**, meaning that both the kernel and the iptables tool can be extended to provide new features.
- **--tcp-flags**

  ```
  # iptables -A INPUT -p tcp --tcp-flags SYN  -j DENY
    (ACK, ALL)
  ```
- **--sport   --source-port**
- **--dport   --destination-port**

  ```
  # iptables -A INPUT -p tcp --dport  23  -j DENY
  ```
- **--tcp-option**

Work Around, Kernel Flags

- # Make sure that IP forwarding is turned off. We only want this for a multi-homed host.
  /bin/echo "0" > /proc/sys/net/ipv4/ip_forward

Standardgemäß ausgeschalten
=> für Routing aktivieren!

- # Disable response to ping.
  /bin/echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_all
- # Disable response to broadcasts.
  /bin/echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
- # Disable ICMP redirect acceptance. ICMP redirects can be used to alter your routing
  # tables, possibly to a bad end.
  /bin/echo "0" > /proc/sys/net/ipv4/conf/all/accept_redirects
- # Enable bad error message protection.
  /bin/echo "1" > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
- # Log spoofed packets, source routed packets, redirect packets.
  /bin/echo "1" > /proc/sys/net/ipv4/conf/all/log_martians
- # Turn on reverse path filtering. This helps make sure that packets use
  # legitimate source addresses, by automatically rejecting incoming packets
  # if the routing table entry for their source address doesn't match the network
  # interface they're arriving on.

  /bin/echo "1" > /sys/net/ipv4/conf/{interface}/rp_filter

Userland states

NEW, ESTABLISHED, RELATED and INVALID

- **NEW** state tells us that the packet is new in the connection.
- **ESTABLISHED** state has seen traffic in both directions and will then match those packets.
- **RELATED** state is one of the more tricky states. A connection is considered **RELATED** when it is related to another already **ESTABLISHED** connection. What this means, is that for a connection to be considered as **RELATED**, we must first have a connection that is considered **ESTABLISHED**. The **ESTABLISHED** connection will then spawn a connection outside of the main connection. The newly spawned connection will then be considered **RELATED**, if the conntrack module is able to understand that it is **RELATED**. ICMP error messages and redirects etc can be considered as **RELATED** if we have generated a packet that in turn generated the ICMP message. Other good examples of connections that can be considered as **RELATED** are the FTP-data connections that are considered **RELATED** to the FTP control port, and the DCC connections issued through IRC. This could be used to allow ICMP replies, FTP transfers and DCC's to work properly through the firewall. Do note that most TCP protocols and some UDP protocols that rely on this mechanism are quite complex and send connection information within the payload of the TCP or UDP data segments, and hence require special helper modules to be correctly understood.
- **INVALID** state means that the packet can not be identified or that it does not have any state.

Sample Rules

- # Allow unlimited traffic on the loopback interface.
  ```
  iptables -A INPUT  -i lo -j ACCEPT
  iptables -A OUTPUT -o lo -j ACCEPT
  ```
- # Make sure NEW tcp connections are SYN packets
  ```
  iptables -A INPUT -i eth1 -p tcp ! --syn -m state --state NEW -j DROP
  ```
- # Refuse packets claiming to be from a Class A private network.
  ```
  iptables -A INPUT  -i eth1 -s 10.0.0.0/8 -j DROP
  ```
- # Refuse broadcast address packets.
  ```
  iptables -A INPUT -i eth1 -d 192.168.0.255 -j DROP
  ```
- # Allow UDP packets to DNS servers from client.
  ```
  iptables -A OUTPUT -o eth1 -p udp -d 172.27.1.10 --dport 53 \
          -m state --state NEW,ESTABLISHED -j ACCEPT
  ```
- # Allow ssh outbound.
  ```
  iptables -A INPUT  -i eth1 -p tcp --sport 22 \
              -m state --state ESTABLISHED -j ACCEPT
  iptables -A OUTPUT -o eth1 -p tcp --dport 22 \
  -m state --state NEW,ESTABLISHED -j ACCEPT
  ```
- # Allow www outbound to 80.
  ```
  iptables -A INPUT  -i eth1 -p tcp --sport 80 \                              -m
  state --state ESTABLISHED -j ACCEPT
  iptables -A OUTPUT -o eth1 -p tcp --dport 80\
          -m state --state NEW,ESTABLISHED -j ACCEPT
  ```

```
# Allow ftp
iptables -A INPUT  -i $IFACE -p tcp --sport 21 \
-m state --state ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o $IFACE -p tcp --dport 21 \
-m state --state NEW,ESTABLISHED -j ACCEPT

# 1) Active ftp.
iptables -A INPUT  -i $IFACE -p tcp --sport 20 \
-m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -o $IFACE -p tcp --dport 20 \
-m state --state ESTABLISHED -j ACCEPT

# 2) Passive ftp.
iptables -A INPUT  -i $IFACE -p tcp --sport $UP_PORTS --dport $UP_PORTS \
-m state --state ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o $IFACE -p tcp --sport $UP_PORTS --dport $UP_PORTS \
-m state --state ESTABLISHED,RELATED -j ACCEPT

 $UP_PORTS=„1024:65535“
```

more samples

- #allow any established back in and related
  - iptables -A INPUT -m state --state ESTABLISHED,RELATED –j ACCEPT
- #allow vpn pptp
  - iptables -A INPUT -p tcp --dport 1723 -j ACCEPT
  - iptables -A INPUT -p 47 -j ACCEPT
- #allow http and https through forward the router
  - iptables -A FORWARD –i eth0 –o eth1 -p tcp --dport 80 -j ACCEPT
  - iptables -A FORWARD –i eth0 –o eth1 -p tcp --dport 443 -j ACCEPT
- #allow any established and related
  - iptables -A FORWARD -m state --state ESTABLISHED,RELATED \
    –j ACCEPT

logging

```
# Any icmp not already allowed is logged and then dropped.
iptables -A INPUT  -i $IFACE -p icmp -j LOG --log-prefix "IPTABLES ICMP-IN: "
iptables -A INPUT  -i $IFACE -p icmp -j DROP
iptables -A OUTPUT -o $IFACE -p icmp -j LOG --log-prefix "IPTABLES ICMP-OUT: "
iptables -A OUTPUT -o $IFACE -p icmp -j DROP
# Any tcp not already allowed is logged and then dropped.
iptables -A INPUT  -i $IFACE -p tcp -j LOG --log-prefix "IPTABLES TCP-IN: "
iptables -A INPUT  -i $IFACE -p tcp -j DROP
iptables -A OUTPUT -o $IFACE -p tcp -j LOG --log-prefix "IPTABLES TCP-OUT: "
iptables -A OUTPUT -o $IFACE -p tcp -j DROP
# Anything else not already allowed is logged and then dropped.
# It will be dropped by the default policy anyway.
iptables -A INPUT  -i $IFACE -j LOG --log-prefix "IPTABLES PROTOCOL-X-IN: "
iptables -A INPUT  -i $IFACE -j DROP
iptables -A OUTPUT -o $IFACE -j LOG --log-prefix "IPTABLES PROTOCOL-X-OUT: "
iptables -A OUTPUT -o $IFACE -j DROP
```

Testing with a simple script

```
#! /bin/bash
iptab="/usr/sbin/iptables"
case "$1" in
  start)
    #…eigene Regeln………………………..
    ;;
  stop)
    #echo "flush and allow all"
    $iptab -F FORWARD
    $iptab -F INPUT
    $iptab -F OUTPUT
    $iptab -P FORWARD ACCEPT
    $iptab -P OUTPUT ACCEPT
    $iptab -P INPUT ACCEPT
    ;;
*)
    echo "Usage: name {start|stop}"
    exit 1
esac
exit 0
```

Übung

- Vorbereitung
- Linuxrechner mit Apache, FTP-Server und Netzverb.
- Firewall-Skript vorbereiten
- Nur http Zugriff erlauben
- Nur ssh Zugriff erlauben
- Weitere Beispiele von vorher testen
- …..
- Erfolglose Verbindungsversuche mitloggen
-