Iptables - theoretical

Oskar Andreasson
blueflux@koffein.net

Introduction

- The speaker
- A brief Table of Contents

The speaker - that is me

- Oskar Andreasson
- From Sweden
- Used Linux since 1994.
- Written about Iptables since 2.4 kernels
- My reliability
  - I am here as a private person.
  - No companies in my back.
    - I will say what I like and don’t like.
    - I am not here to sell.

A brief Table of Contents

Table of Content

Iptables - what is it

Table of Content

Iptables - The Linux 2.4 IP filter solution

Where did it come from

- BSD -> Linux 2.0 (ipfw)
- Linux 2.0 (ipfw) -> Linux 2.2 (ipchains)
  - Rusty Russell
- Linux 2.2 (ipchains) -> Linux 2.4 (iptables)
  - Rusty Russell
  - Netfilter core team
  - Others

Iptables - The Linux 2.4 IP filter solution (cont.)

By whom was it written

- The Netfilter core team
  - A small group of large contributors (single persons).
  - The main group of people developing Netfilter/iptables.
  - Governs the main iptables tree.

- Others:
  - Anyone with the time or will to contribute.

Iptables - Basic functionalities - Stateful firewalling

Full state matching

- TCP
- UDP
- ICMP

Other protocols

- Uses a generic connection tracking module
- The generic conntrack module is less specific
- It is possible to write your own conntrack modules
- Certain protocols are "complex"
  - Requires extra modules called "conntrack helpers"
  - Examples are FTP, IRC (DCC), AH/ESP and ntalk

Iptables - Basic functionalities - Stateful firewalling (cont.)

Userland states

- NEW
  - All new connections
  - Includes Non SYN TCP packets
- ESTABLISHED
  - All connections that has seen traffic in both directions
- RELATED
  - All connections/packets related to other connections
  - Examples: ICMP errors, FTP-Data, DCC
- INVALID
  - Certain invalid packets depending on states
  - E.g. FIN/ACK when no FIN was sent

Iptables - Basic functionalities - Stateful firewalling (cont.)

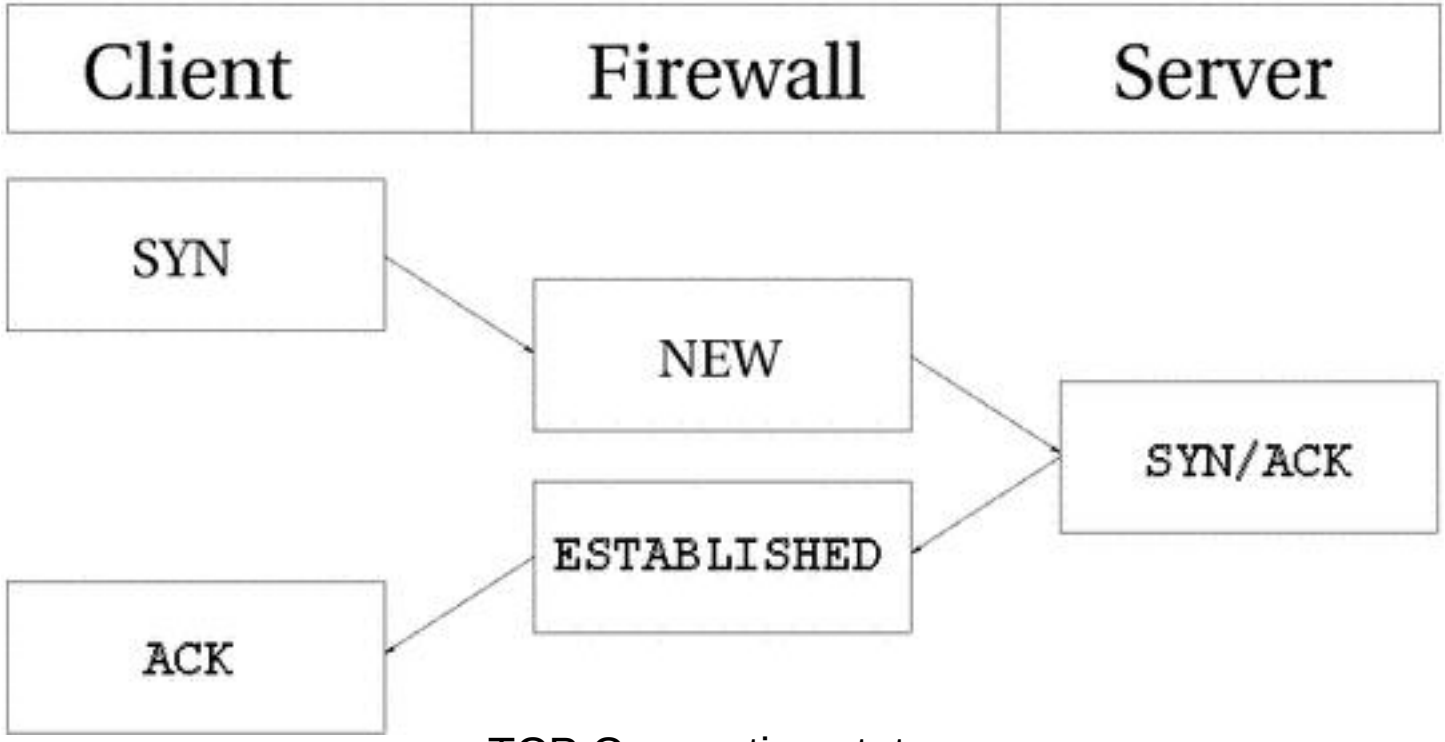
TCP
- Internal states
- Patches allows matching on internal states
- Patches implements full Window tracking

Caution
- Does not care about SYN flag in new TCP streams!

Useful resources
- RFC 793 pp. 21-24

Iptables - Basic functionalities - Stateful firewalling (cont.)

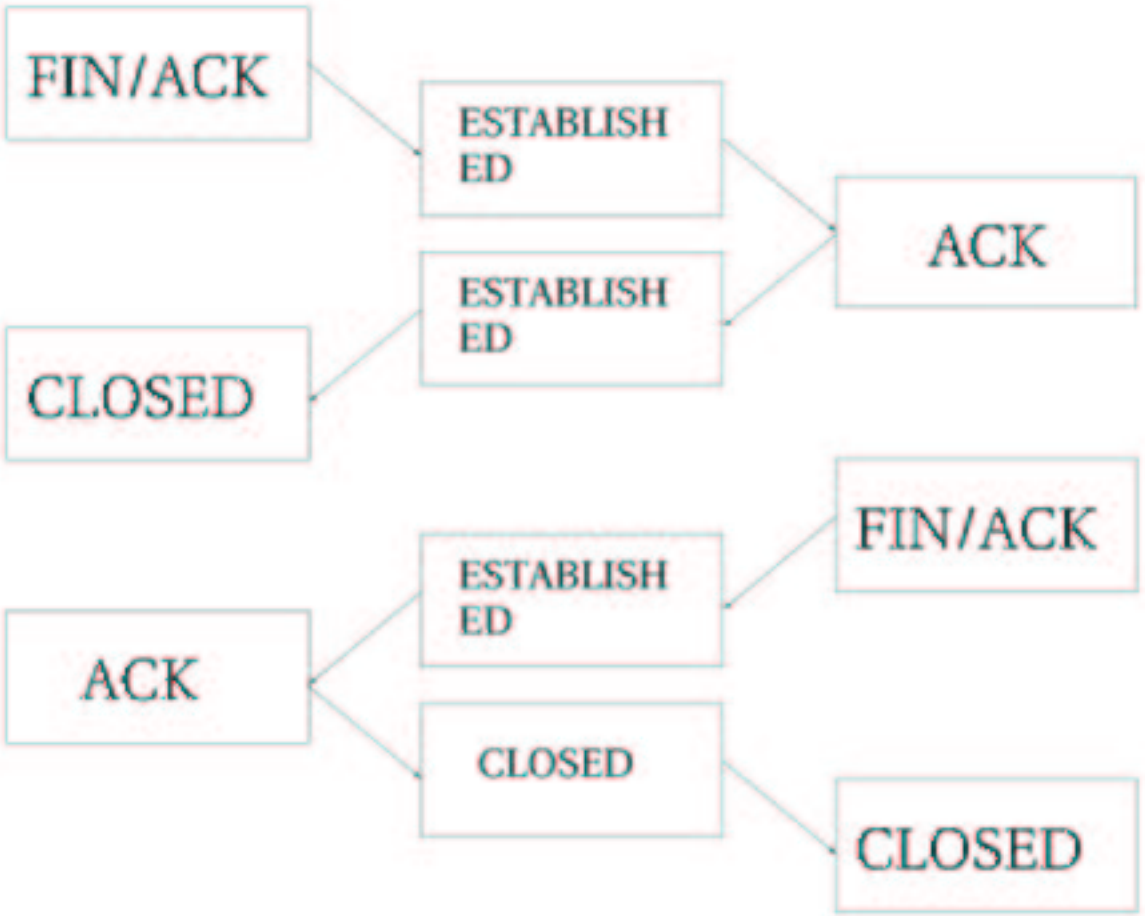
TCP states



TCP Connecting states

Iptables - Basic functionalities - Stateful firewalling (cont.)

TCP States



TCP Closing states

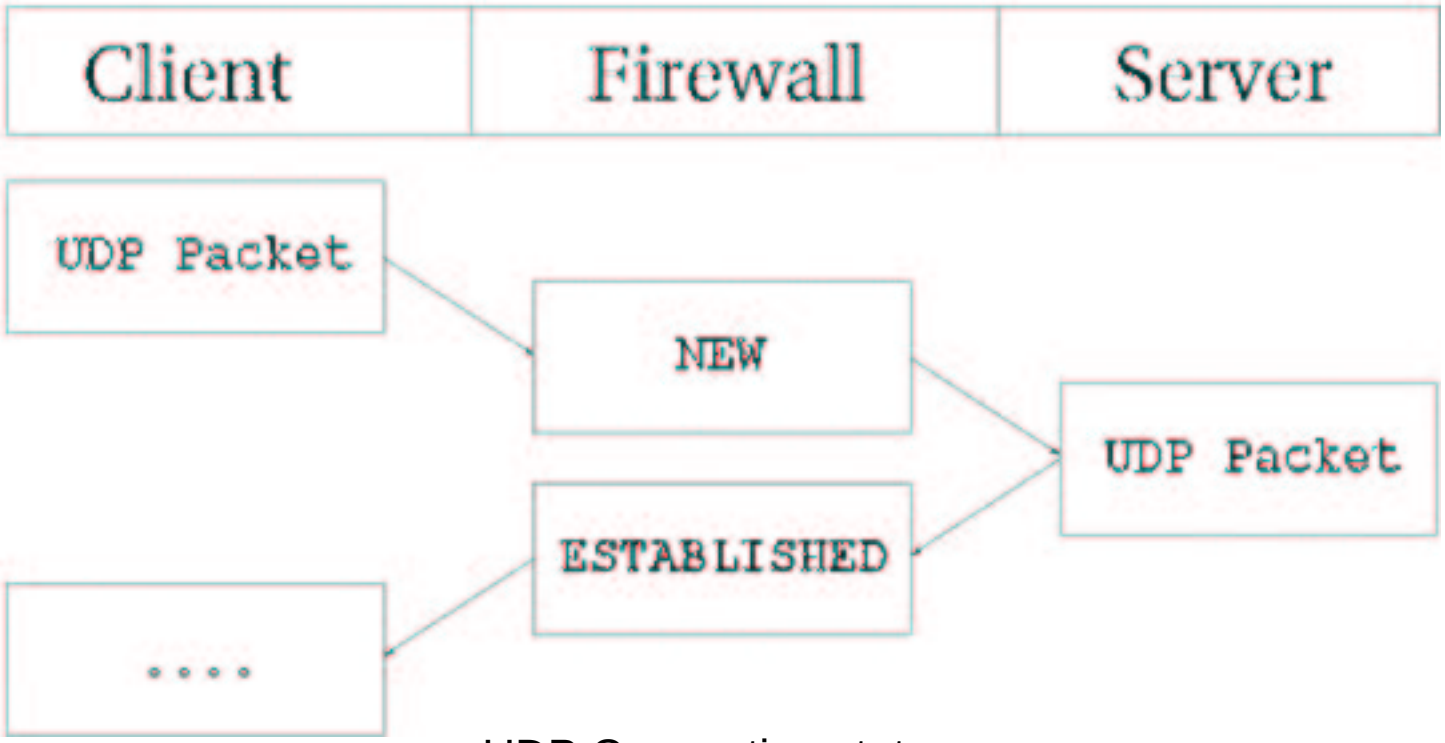
Iptables - Basic functionalities - Stateful firewalling (cont.)

UDP

- A connectionless protocol
- Possible to set states anyway
- Less specific than TCP

Iptables - Basic functionalities - Stateful firewalling (cont.)

UDP States



UDP Connecting states

Iptables - Basic functionalities - Stateful firewalling (cont.)

ICMP

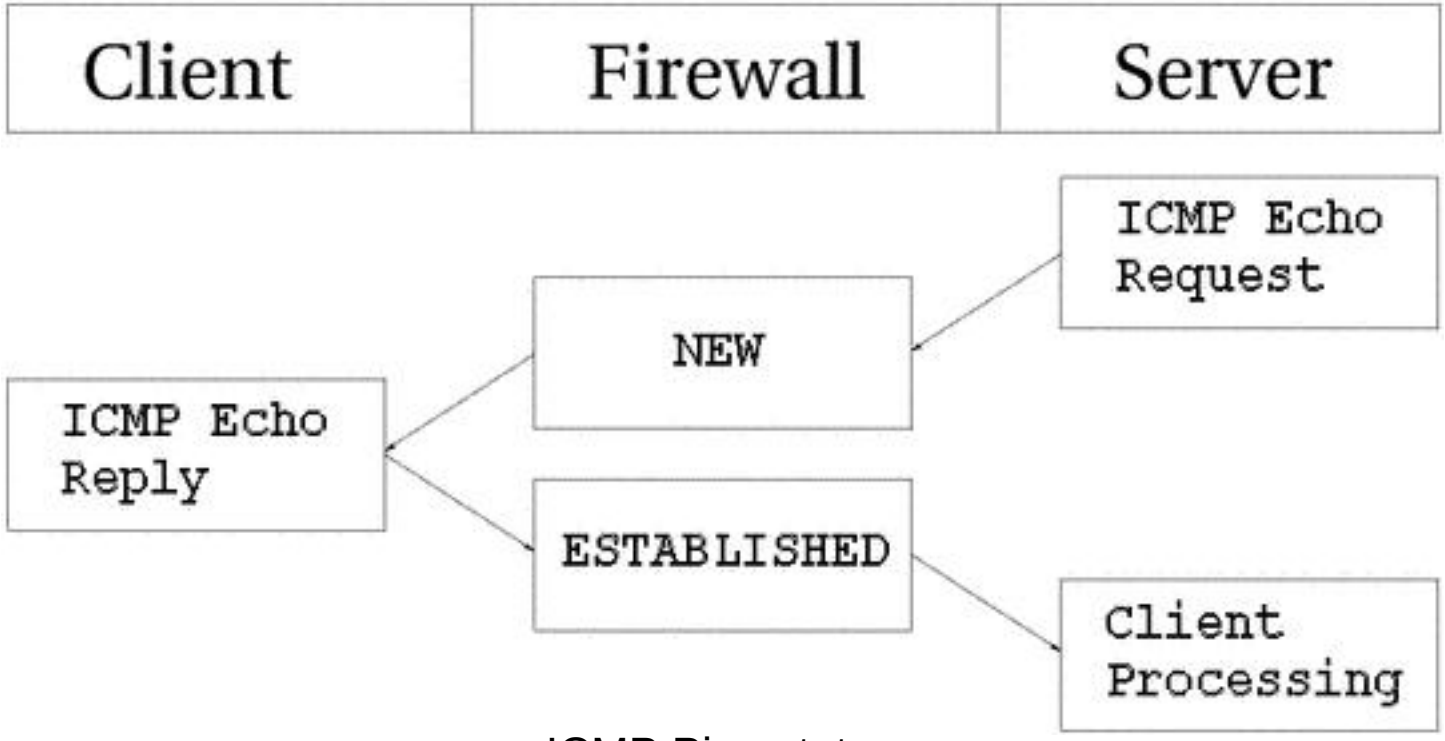
- Even this protocol has states
- They can be NEW or ESTABLISHED

NEW and ESTABLISHED

- ICMP Echo request and reply
- ICMP Timestamp request and reply
- ICMP Information request and reply
- ICMP Address mask request and reply
- All error messages related to other connections.

Iptables - Basic functionalities - Stateful firewalling (cont.)

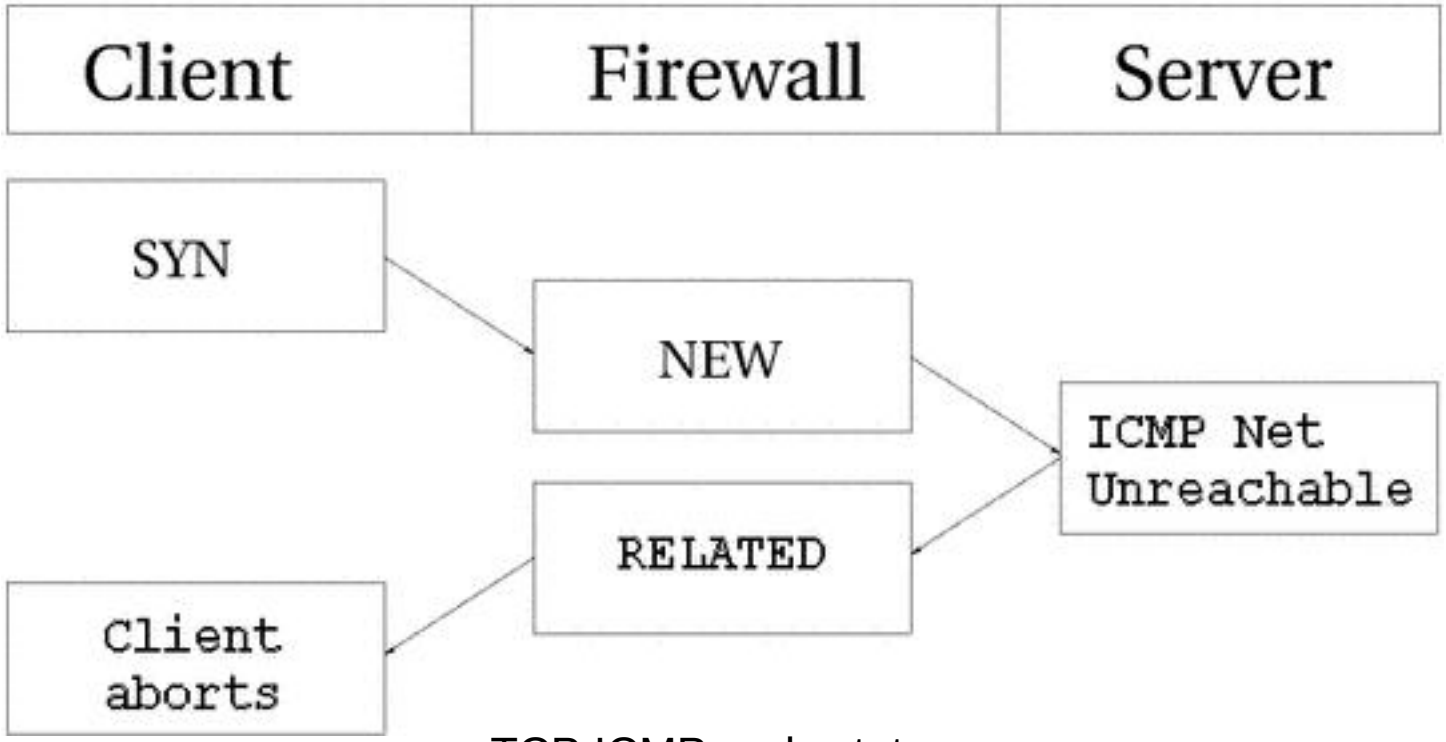
ICMP States



ICMP Ping states

Iptables - Basic functionalities - Stateful firewalling (cont.)

ICMP States



TCP ICMP reply states

UDP behaves exactly the same way

Iptables - Basic functionalities - NAT

The science of switching Source or Destination Addresses

Two types of NAT in Linux 2.4

- Netfilter NAT
- Fast NAT

Prohibited in IPv6
Next to a must in IPv4

Usages

- Making a LAN look as if it came from a single source (the firewall)
- Creating separate servers with a single IP

Iptables - Basic functionalities - NAT (cont.)

Netfilter NAT

- DNAT - Destination Network Address Translation
- SNAT - Source Network Address Translation
- Relatively slow
- Extremely good granularity
- Requires Connection tracking to keep states and expectations

Fast NAT

- Implemented in the core TCP/IP stack
- Very fast
- Less granularity than Netfilter NAT
- No support for complex protocols
- Good support for one to one NAT

Iptables - Basic functionalities - NAT (cont.)

Netfilter NAT and Fast NAT mutually exclusive

- You can only use one of them at a time

Conclusion

- Netfilter NAT better when granularity needed
- Netfilter NAT better when complex protocols are used
- Fast NAT better when speed is of a consideration
- Fast NAT better for one to one NAT

Iptables - Basic functionalities - Packet Mangling

Mangling packets going through the firewall

Gives you the ability to a multitude of possibilities.

Example usages

- Strip all IP options
- Change TOS values
- Change TTL values
- Strip ECN values
- Clamp MSS to PMTU
- Mark packets within kernel
- Mark connections within kernel

Iptables - What iptables is not

Not a proxy solution

- Very common misconception
- Use squid instead

Not a packet data filtering solution

- Very closely related to the above problem
- Use squid and snort for this kind of usage

Iptables - What iptables is not (cont.)

Example of wrong usage of the string match

- Using the string match to drop nimda or sircam
- Results in dead unusable sockets on server and client
- and reject results in dead sockets on server
- Very effective DoS attack!

A complete firewall

- Iptables is not a complete firewall
- Lacks several features, which should always reside in userspace
  - A good NIDS (snort)
  - A filtering proxy solution (squid)
- Get rid of unnecessary services
  - Get rid of HTTP, FTP, telnet, et al.
  - These causes unnecessary security considerations

Iptables - What this means in reality

A framework for filtering connections

- Via the filter table
- Powerful and flexible

A framework for accounting

- Via the filter table
- Using the built in packet and byte counters

A simple way to do Network Address Translation

- Good flexibility
- Possible to use even for complex protocols

Ability to mangle packets

- Extremely powerful
- Useful for all sorts of situations

Packet traversal

Table of Content

Packet traversal - Introduction

How a packet traverses the inside of the kernel

- Extremely important to understand
- Horrible mistakes possible

3 basic tables

- filter (default)
- nat
- mangle

Each table contains a number of chains
Userspecified chains may be specified in a table

The main chains may then call the userspecified chains

Packet traversal - Tables

Filter table

- Used for filtering
- Contains 3 chains
  - INPUT
  - OUTPUT
  - FORWARD
- Certain targets may not be used here
  - NAT targets
  - Mangle targets
  - Filtering targets works perfectly

Packet traversal - Tables (cont.)

Nat table

- Used for Network Address Translation
- Only the first packet of a connection hits this table
  - Subsequent packets in the connection has the same action taken
  - Avoid pure filtering in this chain!
- Contains 3 chains
  - PREROUTING
  - POSTROUTING
  - OUTPUT

Packet traversal - Tables (cont.)

Mangle table

- Used for mangling packets
- Only the first packet in a connection hits this table
  - Same as for the nat table
- Contains 3 or 5 chains
  - PREROUTING
  - POSTROUTING
  - OUTPUT
  - INPUT (with mangle5hooks patch or new kernel)
  - FORWARD (same here)

Packet traversal - Chains

Filter table

INPUT
- Used to filter packets entering the firewall
- Only packets destined for the firewall itself

OUTPUT
- Used to filter packets leaving the firewall
- Only packets generated by the firewall itself

FORWARD
- Used to filter all packets passing through the firewall
- No traffic destined for the firewall will ever hit this chain
- Same goes for traffic generated by the firewall

Packet traversal - Chains (cont.)

Nat table

PREROUTING
- Used to DNAT packets
- Hit before routing decision is made

POSTROUTING
- Used to SNAT packets
- Hit after routing decision is made

OUTPUT
- Used to DNAT and SNAT all locally generated packets
- Hit before and after routing decision is made

Packet traversal - Chains (cont.)

Mangle table

PREROUTING
- Used to mangle packets before routing decision is made

POSTROUTING
- Used to mangle packets after routing decision is made

OUTPUT
- Used to mangle packets created by the firewall
- Hit before routing decision is made

Packet traversal - Chains (cont.)

Mangle table (cont.)

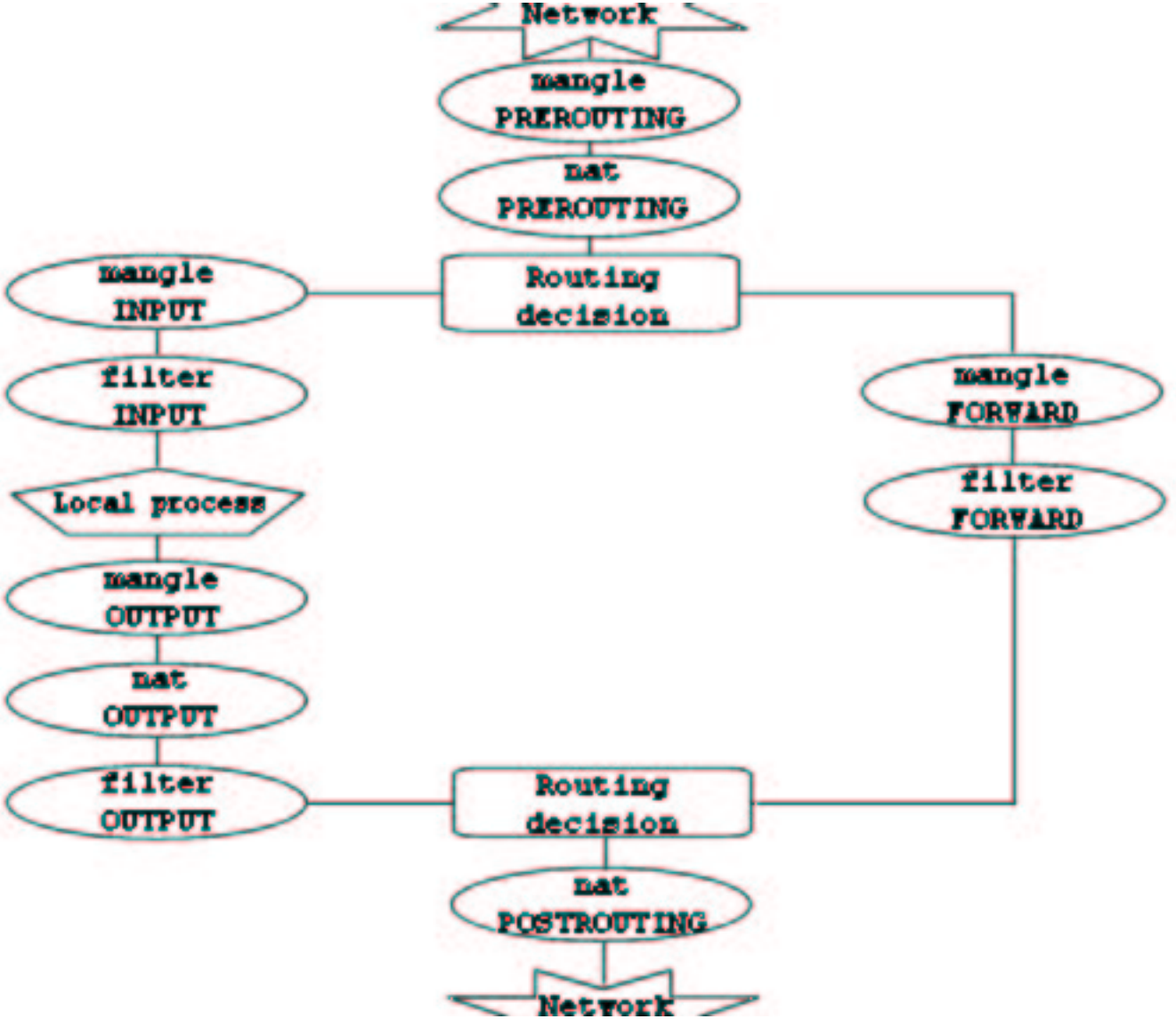
INPUT (with mangle5hooks patch or new kernel)

- Used to mangle packets destined for the firewall
- Hit after routing decision is made

FORWARD (With mangle5hooks patch or new kernel)

- Used to mangle packets routed through the firewall
- Hit after the first routing decision
- Hit before the second routing decision

Packet traversal - How they hook together



Packet traversal - A complete internal packet traversal path

To the firewall

- PREROUTING, mangle
- PREROUTING, nat
- Routing decision
- INPUT, mangle
- INPUT, filter

From the firewall

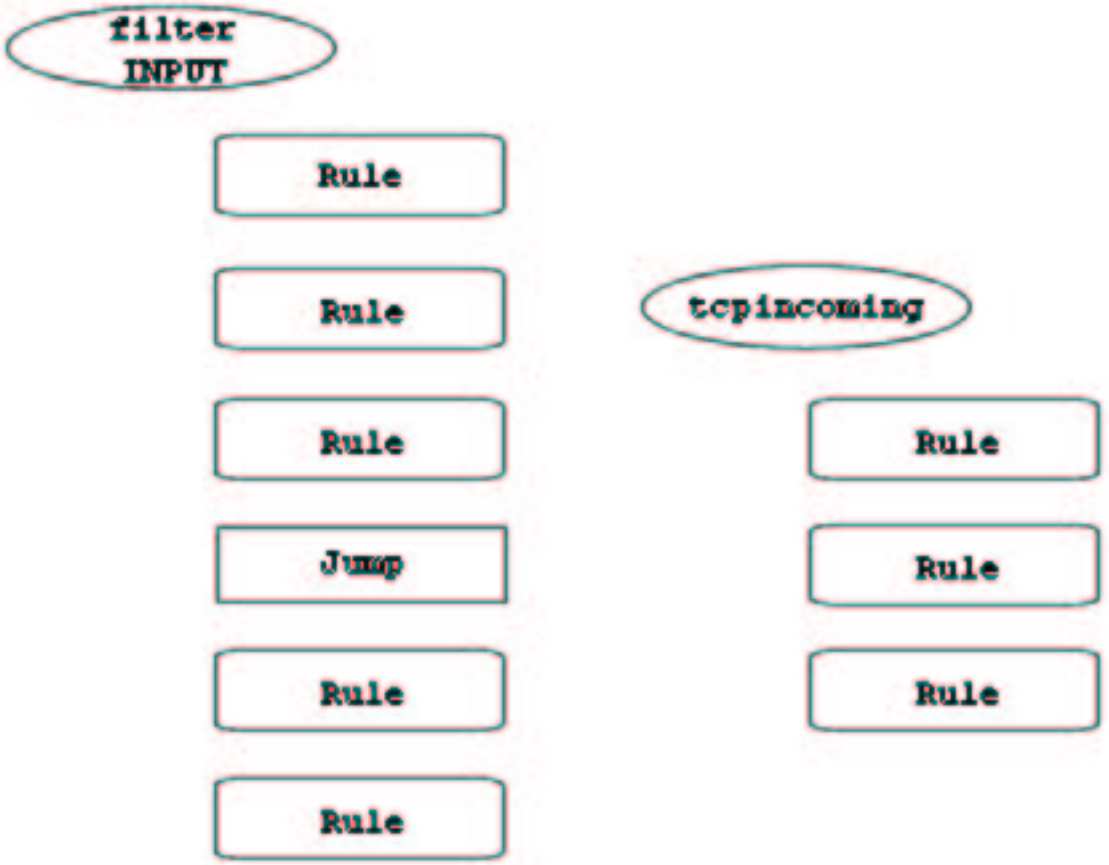
- Routing for source address
- OUTPUT, mangle
- OUTPUT, nat
- OUTPUT, filter
- Routing decision
- POSTROUTING, nat
- POSTROUTING, mangle

Packet traversal - A complete internal packet traversal path (cont.)

Forwarded through the firewall

- PREROUTING, mangle
- PREROUTING, nat
- Routing decision
- FORWARD, mangle
- FORWARD, nat
- Routing decision
- POSTROUTING, nat
- POSTROUTING, mangle

Packet traversal - Traversal of a single chain



Complexity

Table of Content

- Complex protocols
- Remote managing the firewall
- SNAT
- DNAT

Complexity - Complex protocols

What is a complex protocol?

- Opens a control channel
- Opens subconnections
- Subconnection ports decided in control channel
- Netfilter can not work on complex protocols per default

Helpers

- Helps netfilter work on complex protocol
- Helpers (generally) contain two parts
  - Connection tracking part (ip_conntrack_*)
  - NAT part (ip_nat_*)

Complexity - Complex protocols (cont.)

FTP

- Uses 2 ports
  - FTP Control
  - FTP Data
- FTP Control
  - Controls the session
  - Negotiates ports for Data connections
- FTP Data
  - Sends all data in a connection
- Active mode
  - Client opens data connections
- Passive mode
  - Server opens data connections

ip_conntrack_ftp

ip_nat_ftp

Complexity - Complex protocols (cont.)

IRC

- Uses 1 port normally
- Uses extra ports to handle DCC
- IRC negotiates DCC ports
- DCC are used for direct chats and filesends
- DCC connections are initiated by calling part

ip_conntrack_irc

ip_nat_irc

Complexity - Complex protocols (cont.)

Other helpers currently available

- Basic SNMP-ALG (RFC 2962)
- talk, ntalk, ntalk2 (development)
- tftp (development)
- PPTP (development)
- eggdrop (IRC bots, development)
- GRE + PPTP (development, partial)
- H.323 (development, netmeeting only)

Other protocols in need helpers

- ICQ file sharing
- Real Audio servers
- Tunneling protocols
- Proprietary protocols

Complexity - Remote managing the firewall

via SSH
- Simple
- iptables works perfectly in SSH
- Only requires SSH to be open

via HTTP and CGI
- Complicated
- Requires HTTP & possibly dangerous CGI scripts
- Fairly good solution for broadband ISP functionality
- No publicly available solutions

via VNC
- Requires quite some bandwidth
- Requires VNC to be open
- Possible to use GUI configuration solutions

Complexity - SNAT

Source Network Address Translation

- Ability to let hosts onto the Internet without real IP’s
- Used for "hiding" local networks
- SNAT target
- All SNAT’ed packets will look as if they came from specified IP
- Only possible in the POSTROUTING chain in the nat table

Example

```
iptables -t nat -A POSTROUTING -i $LAN_IFACE \
-j SNAT --to-source $INET_IP
```

Complexity - DNAT

Destination Network Address Translation

- Can be used for putting servers on internal networks
- Redirects packets to one destination to another
- Load balancing
- Only possible in PREROUTING in the nat table

Example

```
iptables -t nat -A PREROUTING -d 10.0.0.1 --dport 80 \
-j DNAT --to-destination 192.168.1.2
```

Complexity - DNAT (cont.)

Getting it to work from the same network

- Will cause routing troubles if client is on the same network
- Client will not recognize responses from server

Solution

- Make all packets go back through the firewall and get DNAT'ed
- Done via SNAT

Example:

```
iptables -t nat -A PREROUTING -d 10.0.0.1 --dport 80 \
-j DNAT --to-destination 192.168.1.2
iptables -t nat -A POSTROUTING -d 192.168.1.2 --dport 80 \
-j SNAT --to-source 192.168.1.1
```

Complexity - DNAT (cont.)

and from the firewall

- Packets generated by the firewall will not get DNAT'ed
- Packets will hence go to the firewall itself

Solution

- Use DNAT in the OUTPUT chain in the nat table

Example:

```
iptables -t nat -A PREROUTING -d 10.0.0.1 --dport 80 \
-j DNAT --to-destination 192.168.1.2
iptables -t nat -A OUTPUT -d 10.0.0.1 --dport 80 \
-j DNAT --to-destination 192.168.1.2
iptables -t nat -A POSTROUTING -d 192.168.1.2 --dport 80 \
-j SNAT --to-source 192.168.1.1
```

The evolved ruleset

Table of Content

The evolved ruleset - Our evolved goals

Same goals as before

- Internal network need Internet access
- HTTP to access to everyone
- Identd ability
- All except group "nonet" to have net access

New goals

- HTTP server separated and put on LAN
- FTP clients to work properly
- GRE Tunnel from internal to external server
- Microsoft Netmeeting

The evolved ruleset - The technical details

Firewall
- LAN on eth0
- LAN IP 192.168.1.1
- Internet on eth1
- Internet IP 10.0.0.1/32

LAN
- IP range 192.168.1.0/24
- FTP to Internet
- Microsoft Netmeeting to Internet
- GRE server at 192.168.1.10
- HTTP server 192.168.1.2

The evolved ruleset - The PREROUTING chain

DNAT all packets to HTTP port

Do not forget OUTPUT DNAT

DNAT all packets to GRE

Does not require OUTPUT DNAT

```
iptables -t nat -A PREROUTING -p gre -d 10.0.0.1 \
-j DNAT --to-destination 192.168.1.10
iptables -t nat -A PREROUTING -p tcp -d 10.0.0.1 --dport 80 \
-j DNAT --to-destination 192.168.1.2
iptables -t nat -A OUTPUT -p tcp -d 10.0.0.1 --dport 80 \
-j DNAT --to-destination 192.168.1.2
```

The evolved ruleset - The POSTROUTING chain

SNAT all normal traffic to Internet
SNAT all packets to HTTP server (as described previously)
SNAT all packets to GRE server

```
iptables -t nat -A POSTROUTING -i eth0 -o eth1 -j SNAT \
--to-source 10.0.0.1
iptables -t nat -A POSTROUTING -o eth0 -d 192.168.1.2 -j SNAT \
--to-source 10.0.0.1
iptables -t nat -A POSTROUTING -o eth0 -d 192.168.1.10 -j SNAT \
--to-source 10.0.0.1
```

The evolved ruleset - The FORWARD chain

Allow DNAT’ed packets through

Allow all traffic from LAN to Internet

Allow established and related from Internet to LAN

Drop everything else

```
iptables -P FORWARD DROP

iptables -A FORWARD -p tcp -i eth1 -o eth0 -d 192.168.1.2 \
--dport 80 -j ACCEPT
iptables -A FORWARD -p gre -i eth1 -o eth0 -d 192.168.1.10 \
-j ACCEPT
iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
iptables -A FORWARD -i eth1 -m state \
--state ESTABLISHED,RELATED -j ACCEPT
```

The evolved ruleset - The INPUT chain

Identd runs on firewall
Allow ICMP Echo & reply
Allow established & related connections back in

```
iptables -P INPUT DROP

iptables -A INPUT -p tcp --dport 113 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 8 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 0 -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED \
-j ACCEPT
```

The evolved ruleset - The OUTPUT chain

And the "nonet" group is blocked again

```
iptables -A OUTPUT -m owner --gid-owner nonet -j DROP
```

The evolved ruleset - The complete ruleset

```
iptables -P FORWARD DROP
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT

iptables -t nat -A PREROUTING -p gre -d 10.0.0.1 \
-j DNAT --to-destination 192.168.1.10
iptables -t nat -A PREROUTING -p tcp -d 10.0.0.1 --dport 80 \
-j DNAT --to-destination 192.168.1.2
iptables -t nat -A OUTPUT -p tcp -d 10.0.0.1 --dport 80 \
-j DNAT --to-destination 192.168.1.2

iptables -t nat -A POSTROUTING -i eth0 -o eth1 -j SNAT \
--to-source 10.0.0.1
iptables -t nat -A POSTROUTING -o eth0 -d 192.168.1.2 -j SNAT \
--to-source 10.0.0.1
iptables -t nat -A POSTROUTING -o eth0 -d 192.168.1.10 -j SNAT \
--to-source 10.0.0.1
```

The evolved ruleset - The complete ruleset (cont.)

```
iptables -A FORWARD -p tcp -i eth1 -o eth0 -d 192.168.1.2 \
--dport 80 -j ACCEPT
iptables -A FORWARD -p gre -i eth1 -o eth0 -d 192.168.1.10 \
-j ACCEPT
iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
iptables -A FORWARD -i eth1 -m state \
--state ESTABLISHED,RELATED -j ACCEPT

iptables -A INPUT -p tcp --dport 113 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 8 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 0 -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED \
-j ACCEPT

iptables -A OUTPUT -m owner --gid-owner nonet -j DROP

echo 1 > /proc/sys/net/ipv4/ip_forward
```

Final notes - Other resources

http://www.netfilter.org

http://iptables-tutorial.haringstad.com

http://www.linuxguruz.org/iptables/

http://www.islandsoft.net/veerapen.html

http://www.rfc-editor.org

http://www.lartc.org

http://www.docum.org