Open source firewall tools
Iptables and PF

Prepared by : Elvir Kuric
(elvir.kuric@hp.com)

Created with OpenOffice

Agenda

- Iptables description
- Rules creation

- PF description
- Rules creation

About what we going to talk

- Iptables (**http://www.netfilter.org/**)
- Packet Filter Subsystem=PF (**www.openbsd.org**)

Licenses...that wonderful world

- Iptables comes under GNU GPLv2 http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
- PF comes under BSD license www.openbsd.org/policy.html

History of Iptables

- Ipchains (Linux kernel 2.2)

- Iptables (Linux kernel 2.4 and 2.6)
- All work started Rusty Russell in 1998, today all work related to iptables is gathered around Netfilter project (**www.netfilter.org)**

History of PF

- IPFilter, due to licensing issues was removed from OpenBSD project
- Reason: was not allowed to change a code and distribute it without prior approval of authors
  IPFilter is still in use in
  HP-UX, Linux(SLES,RHEL)
- Daniel Hartmeier wrote PF in 2001

Installation of Iptables

- Iptables comes by default installation within all Linux distributions
- On Debian, Red Hat, Centos is present as default packet and no need to take any action regarding installation

- Also is possible to install from source code
- Option with pre-compiled packages is more convenient

Installation of Iptables

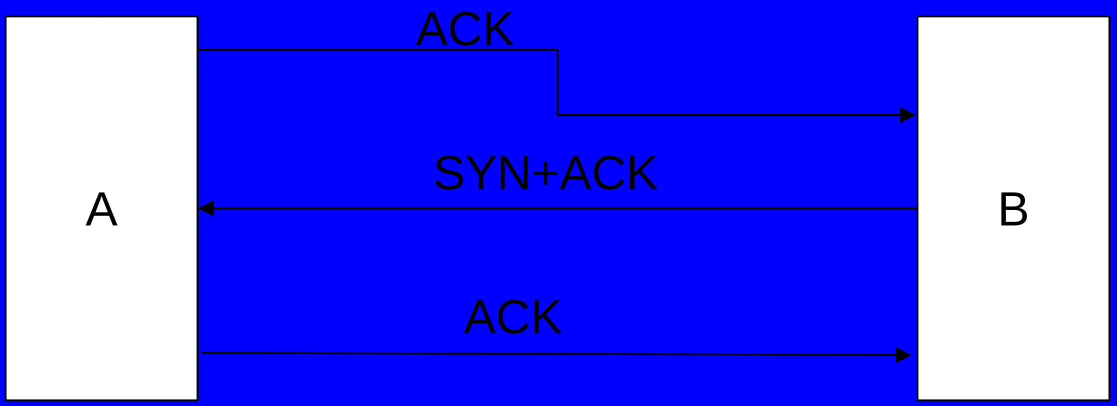
- Check if is iptables packet/software present on system
- On Debian (and its derivate like Ubuntu) : **dpkg -l | grep iptables**
- on Red Hat and its derivate (like CentOS) : **rpm -qa | grep iptables**

Installation of Iptables

- On Debian output is like :
  ```
  #dpkg -l | grep iptables          ii  iptables   1.4.1.1-3
  administration tools for packet filtering and NAT
  ```
- On Red Hat (Centos)
  ```
  #rpm -qa | grep iptables           iptables-1.3.5-4.el5
  ```

TCP/UDP connection creation

- Well known 3-way handshake (ACK,SYN+ACK,ACK) FTP, Telnet, HTTP, HTTPS, SMTP, POP3, IMAP, SSH

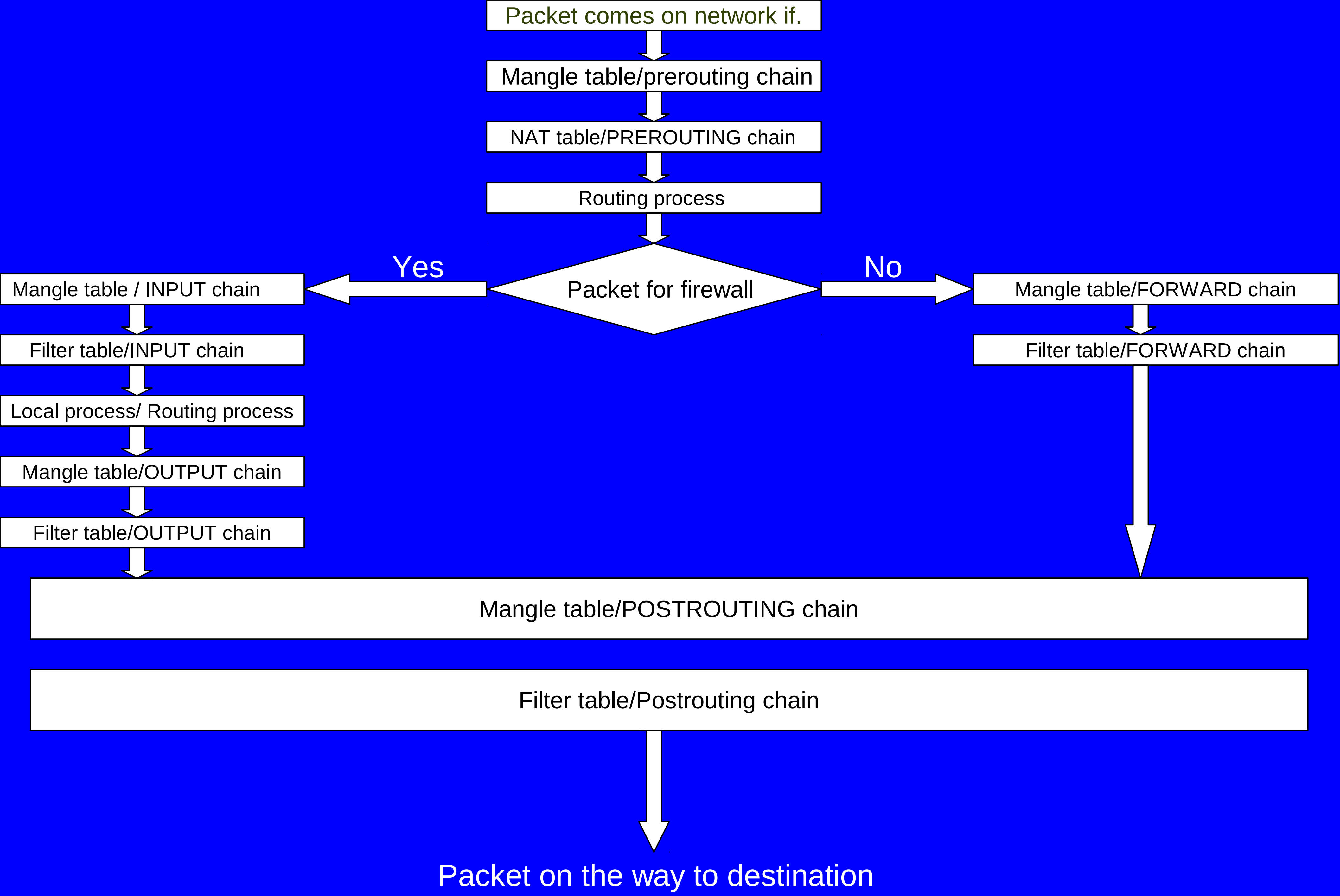


UDP

- For UDP there is not 3-way handshake—in this case server listen on particular port and and accept connections from any client to that port, very convenient for query/response applications like SNMP, DNS, ...

Iptables (http://netfilter.org)

- Current iptables version is 1.4.2



Packet comes on network if.

Mangle table/prerouting chain

NAT table/PREROUTING chain

Routing process

Packet for firewall

Yes

Mangle table / INPUT chain

Filter table/INPUT chain

Local process/ Routing process

Mangle table/OUTPUT chain

Filter table/OUTPUT chain

No

Mangle table/FORWARD chain

Filter table/FORWARD chain

Mangle table/POSTROUTING chain

Filter table/Postrouting chain

Packet on the way to destination

Iptables rule creation

- iptables [-t table] command [match] [target/jump]

TABLES -t

- -t option in iptables rule specifies that we are going to use an table which could be

- MANGLE
- NAT
- FILTER

Mangle table

- Place where to change TTL, ToS and some other fields if necessary within package
- Chains are : PREROUTING, POSTROUTING,INPUT,OUTPUT,FORWARD
- Targets for this table are : TTL, TOS, MARK, ...
- Usage :
  iptables -t mangle (additional rules) -j TTL
- (we will see later how is traffic classification resolved on PF)

Mangle table

- In cases when we need to make some packet classification MANGLE table is right tool
- Minimum delay (16 or 0x10)
- Maximum throughput (8 or 0x08)
- Maximum reliability (4 or 0x04)
- ...

```
iptables -t mangle -A PREROUTING -p tcp --dport 22 -j TOS --set-tos 0x04
```

NAT table

- Cases when we must use NAT to save address space
- Nat table has purpose of NAT-ing for packages which have as destination host in ***inside*** network
- With IPv6 things will change

NAT table

- Targets for this table are : DNAT,SNAT,MASQUERADE,REDIRECT
- USAGE : iptables -t nat ...(some options) -j \ DNAT (SNAT) or some other target
- Chains are: PREROUTING,OUTPUT, POSTROUTING

```
   iptables -t nat -A PREROUTING -i eth0 -j DNAT –to-destination \
192.168.1.2-192.168.1.100

iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 10.0.95.25
```

Filter table

- Place where we take action against packets based on various factors we DROP,LOG,ACCEPT or REJECT particular packet
- Here we can apply three different chains :
  - Forward-apply all non-local packets
  - Input –incoming packets
  - Output – for all packets generated by host

Iptables commands

- -A =append
- -D =delete
- -R =replace
- -I =insert
- -L =list
- -F =flush

- -Z =zero
- -N =new
- -X =delete-chain
- -P =policy
- E =rename

Iptables matches

- Generic matches
- TCP/UDP/ICMP/SCTP matches
- Explicit matches

Iptables matches-Generic

- -p =protocol
- -s =source
- -d =destination

- -i =in-interface
- -o =out-interface
- -f =fragment

Some rules:

```
iptables -A INPUT -p protocol tcp -s 149.11.112.2 -d 12.11.22.0/24 -j DROP

iptables -A OUTPUT -o eth1 -s -m state –state NEW -j ACCEPT

iptables -A INPUT -i eth0 -p icmp -m icmp –icmp-type echo-request -j DROP
```

Iptables matches-TCP/UDP/ICMP/SCTP

- -sport =source-port
- -d =destination-port
- -tcp-flags
- -syn (for TCP)

- -icmp-type (1,8,....)

Iptables matches-Explicit

- ... are always loaded by option
  - -m
  - -match

Iptables matches-Explicit

- -m addrtype (--src-type, --dst-type)
- -m iprange (--src-range, --dst-range)
- -m length (--length)

- -m limit (--limit packets/time)
- -m limit (--limit-burst)
- --mac-source
- -m multiport

Connection tracking

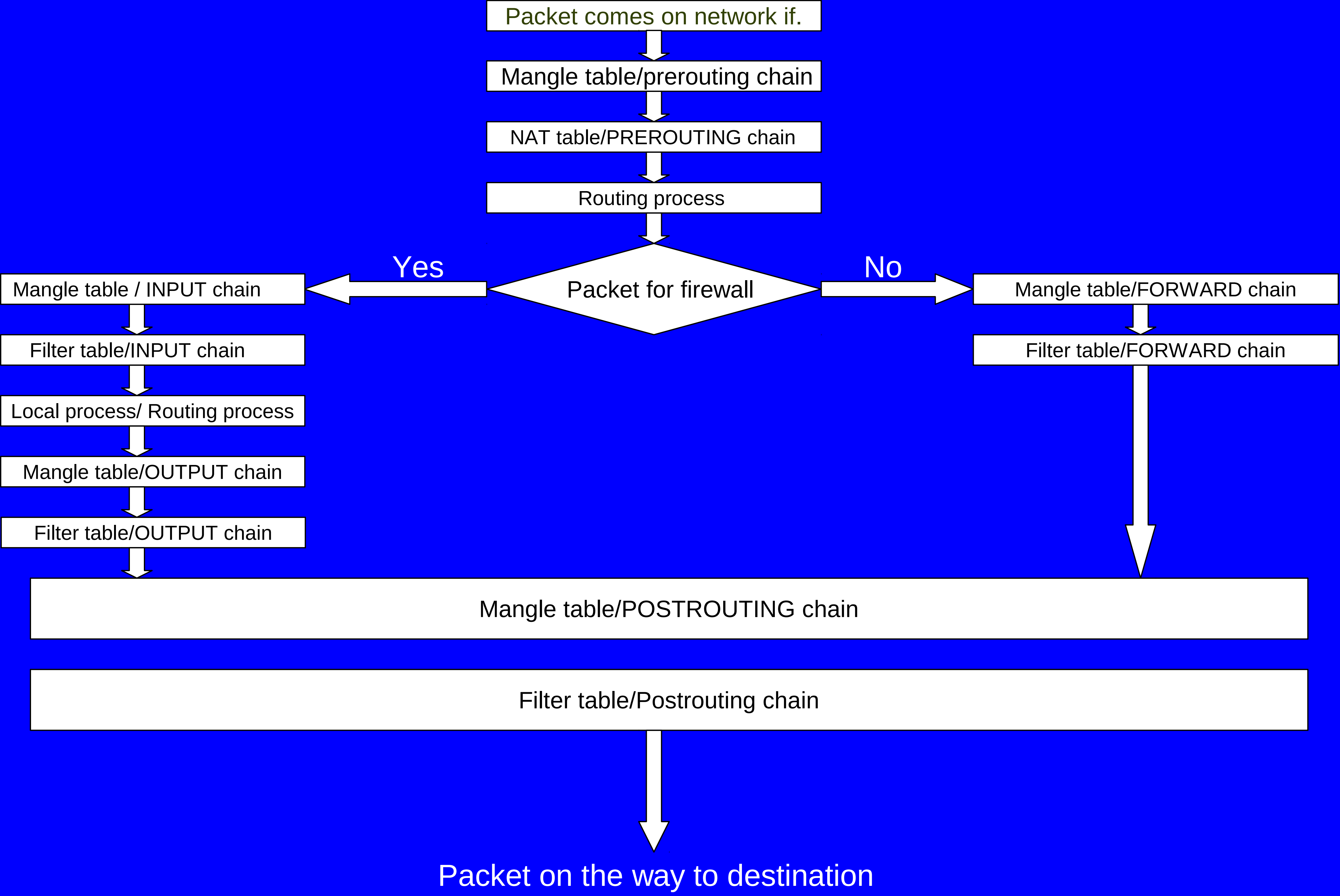
- NEW
- ESTABLISHED
- RELATED
- INVALID

Iptables matches-Explicit

- -m state
  --state(NEW,ESTABLISHED,RELATED)
- -m tos
- -m ttl

Iptables targets

- ACCEPT
- DNAT
- DROP
- LOG
- MASQUERADE
- REDIRECT
- REJECT
- RETURN
- SAME
- SNAT
- TOS (Type Of Service)


Packet comes on network if.
Mangle table/prerouting chain
NAT table/PREROUTING chain
Routing process
Packet for firewall
Yes
No
Mangle table / INPUT chain
Filter table/INPUT chain
Local process/ Routing process
Mangle table/OUTPUT chain
Filter table/OUTPUT chain
Mangle table/FORWARD chain
Filter table/FORWARD chain
Mangle table/POSTROUTING chain
Filter table/Postrouting chain
Packet on the way to destination

Redundancy

- ucarp --patent free implementation of CARP (Common Address Redundancy Protocol)   which enable hosts to share common virtual IP addresses in order to provide automatic fail-over
- ucarp is implemented on Linux/UNIX systems
- http://www.ucarp.org/project/ucarp

Iptables firewall script creation

- vi/vim editor or some other text editor
- Also exist some graphical tools for this purpose

Q/A Iptables

Packet Filter Subsystem=PF

OpenBSD

- Only two remote holes in the default install, in more than 10 years (www.openbsd.org)
- Important -- an operating system is secure as much as system administrator take care about it !!!

OpenBSD PF installation

- PF is part of default OpenBSD installation and it is only necessary to enable it in configuration files

Activating PF on Open|Free BSD

- In /etc on Open|Free BSD create rc.conf.local with
  #!/bin/sh -
- pf=YES
- pf_rules=/etc/pf.conf
- pflogd_flags=

Activating PF on Open|Free BSD

- In /etc/rc.conf.local set **PF=YES** what will activate PF on system
- **pf_rules=/etc/pf.conf** --conf. file where are rules going to be placed
- **pflogd_flags=**
  We want to be able to see logs related to packets we send/receive

Activating PF on Open|Free BSD

- In case it is not standalone machine
  In /etc/sysctl.conf uncomment
  net.inet.ip.forwarding=1
- Reboot the system

PF managing program

- **pfctl** -program for PF manipulation

- **pfctl -e** enable PF
- **pfctl -d** disable PF
- ….
- **man pfctl**
- Useful feature is **pfctl -nf pf.conf** will check syntax of pf.conf file prior of execution
- **pfctl -f pf.conf** will load configuration file

Interface naming in Open|Free BSD

- In Linux we have eth0, eth1, eth2, ...
- BSD interface naming convention is different and depends on network interface vendor
- fxp0 -- Intel cards
- rl0 -- 3Com cards
- ....

PF rule syntax

- action ***direction*** [log] [quick] on ***interface*** [af] proto ***protocol*** \ [**from** src_addr [port src_port]] [**to** dst_addr [port dst_port]] \ [**flags** tcp_flags] [**state**]

PF rule syntax

- action = (pass | block)
- direction =(in|out)
- af=(IPv4|IPv6)
- protocol=(tcp/udp/icmp...)
- from (src address and source port)
- to (dst address and port)
- flags (F-fin, S-syn,R-reset, A-ack,U-urg,
- state

```
pass in on fxp0 protocol tcp from 192.168.1.12 port any to 192.168.1.1 port 22 \
keep state
```

Rules set up in firewall script

- Macros
- Tables
- Options
- Normalization
- Bandwidth management
- Translation
- Redirection
- Filtering

Macros

- For example if we have ….
- servers=”{12.1.11.33,44.44.55.33}”
- ext_in=”fxp0”
- ports=”{22,80,443}”

- **pass in on $ex_if from any port any to $servers port $ports keep state**

Tables

- Tables
- table <table_name> persist “path_to_file”
- Can differ two options : **persist and const**
- **block on $ext_if from {<table_name>} to \ $servers port $ports keep state**
- keep state feature enable us to track connections-stateful inspection. Keep state improve performance

Logging and normalization

- set loginterface $ext_in
- Network equipment on packet way does not handle it on same way
- scrub in all
- Known as packet normalization

PF address translation and redirection syntax

- nat [**pass**] [log] **on *interface*** [af] **from** src_addr [port src_port] **to** \
  dst_addr [port dst_port] -> ext_addr [pool_type] [static-port]
- rdr on ***interface*** proto[*protoco*] from \ source [source_add] to *exter_address* -> *destination_addres port*
- **Cases when we need to have redirection to hosts which are behind firewall**

Bandwidth management and Queueing

- Very difficult topic to discuss about
  - Which criteria we can use as parameter
  - What/who can guarantee us anything over the network
  - If not planned well can cause strange unexpected problems

Bandwidth management and Queueing

- some traffic has higher priority than other and we have to make distinction
- We use **altq (ALTernate Queuing)**

Bandwidth management and Queueing

- FIFO queueing
- priq (Priority based Queuing)
  - PRIQ has from 0 to 15 different priority levels, 0=no priority, 15 = maximal priority
- cbq (Class Based Queuing)
  - 7 different levels (the higher level the higher priority)
  - borrow
- hfsc (Hierarchical Fair Service Curve)

Setting up priority rule

- altq on $interface *type* bandwidth *measure*\ *main_queue* { q1,q2,q3 }
- measure represent our available B in Kb,Mb… and so on
- queue q1 [all_necessary options]
- pass […] queue q1
- pass […] queue q2

Simple rule…

- altq on $ext_if bandwidth 1Mb cbq queue { ssh, http_upload \ http_download, boss, other, **for_boring_people** }
- queue ssh on $ext_if bandwidth 5% priority 7 cbq (borrow)
- queue http_upload on $ext_if bandwidth 20% priority 3 cbq (borrow)
- queue http_download on $ext_if bandwidth 30 % priority 3 \ cbq (default)
- queue boss on $ext_if bandwidth 10% priority 7 cbq (borrow)
- After setting up queues it is necessary to apply them in particular rule
- pass in on $ext_if protocol tcp from any to 1.1.1.1 port 22 queue ssh

Simple rule...

- <**boring_people**> ...list of ip addresses
- ports="{22,80}"
- queue for_boring_people bandwidth 1% cbq

- Now we can create rule allowing unwanted traffic to our server but only on 1% of our bandwidth

- pass in log (all) quick on $ext_if proto tcp from \
  <**boring_people**> to any port $ports \
  queue for_boring_people

Simple rule...

- The last rule can help when we want to limit sending ACK responses in case of SYN flood attacks
- In case of UDP flood attacks will not help

Simple rule...

- The mentioned rule will not help us to protect our host/network but can decrease effects of attack
- Pay attention on logs

Redundancy

- Firewall machine has focal point in network, it is gate between outside and inside
- Failure in cases when there is not redundancy cause serious problems
- CARP (Common Address Resolution Protocol)
- + of CARP protocol are : very low overhead, cryptographically signed messages, interoperability between different OS (package ucarp on Linux systems)
- CARP is included in OpenBSD since version 3.5

Redundancy

- CARP is alternative to VRRP (Virtual Router Redundancy Protocol) or HSRP (Hot Standby Router Protocol)
- CARP can be used for cases when we need to have redundancy in network, in sense of firewalls it should to provide backup in case of planned maintenance or failures

Enable CARP

- Check CARP status using
- sysctl net.inet.carp In case carp is enabled is like
  net.inet.carp.allow=1
  net.inet.carp.preempt=1
  net.inet.carp.log=1
- If carp is not enabled in kernel then:
- Compile kernel and enable CARP
- In /etc/sysctl.conf uncomment lines
  net.inet.carp.preempt=1
  net.inet.carp.log=1

Configuring CARP virtual interface

- Create CARP interface
  ifconfig carpW create ---this will create virtual carp interface
  W=0,1,2,3,... virtual interface number

- ifconfig carpW vhid vhid [**pass** password] [carpdev carpdev]   [advbase advbase] [advskew advskew] \ [state state] ipaddress netmask mask

Configuring CARP virtual interface

- vhid – virtual interface ID
- pass – password
- carpdev –optional, if we want to announce physical interface which is carp virtual interface assigned
- advbase
- advskew
- State – master, backup
- ip_address, netmask

Configuring CARP virtual interface

- ifconfig carp1 create

- For Master configuration could be
  ```
  ifconfig carp1 vhid 1 pass CR1!!$  carpdev fxp0 \
  192.168.5.123 netmask 255.255.0.0
  ```
- For Slave we can set up carp interface
  ```
  ifconfig carp1 vhid 1  pass CR1!!$ carpdev fxp0 \
  192.168.5.123 netmask  255.255.0.0
  ```

Configuring CARP virtual interface

- advbase -how often in seconds to advertise membership of redundancy group, default value is 1 second
- addskew helps master host to be chosen within redundant group

pfsync

- pfsync is second component necessary to build redundant system using CARP
- pfsync represent virtual network interface
- pfsync take care about synchronization between ***master*** and ***slave(s),*** and updates

Pfsync configuration

- ifconfig pfsyncN syncdev syncdev [syncpeer syncpeer]
- syndev –device we use for sending sync messages
- syncpeer –if we want to lock pfsync to send messages to specific host

....What is better ...

- Both iptables and pf are very reliable and excellent solutions for firewalls
- The choice can be guided by OS we have as base (some GNU/Linux =iptables, Open|Net|Free BSD =pf)
- PF has very similar (if not same) syntax as IPFilter on HP-UX system, people with HP-UX experience will find it very easy
- Necessary to understand TPC/IP protocol stack prior to start firewall implementation

Literature

- www.openbsd.org
- www.docs.hp.com/en
- www.debian.org
- The book of PF, Peter N.M Hansteen
- http://www.benzedrine.cx/pf-paper.html
- www.netfilter.org
- www.itrc.hp.com

Thank you

elvir.kuric@hp.com

Q/A