+ iptables

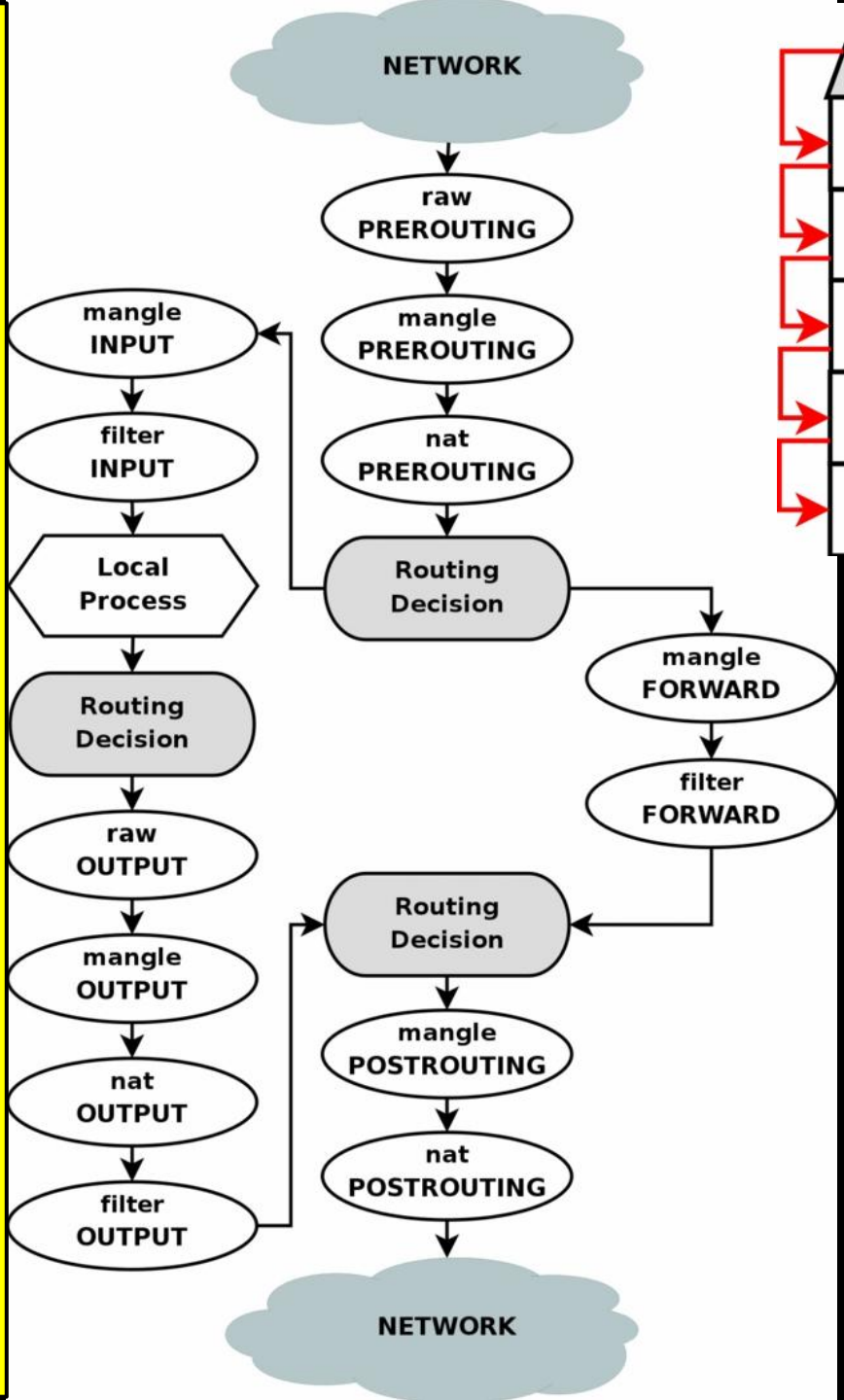
packet filtering && firewall

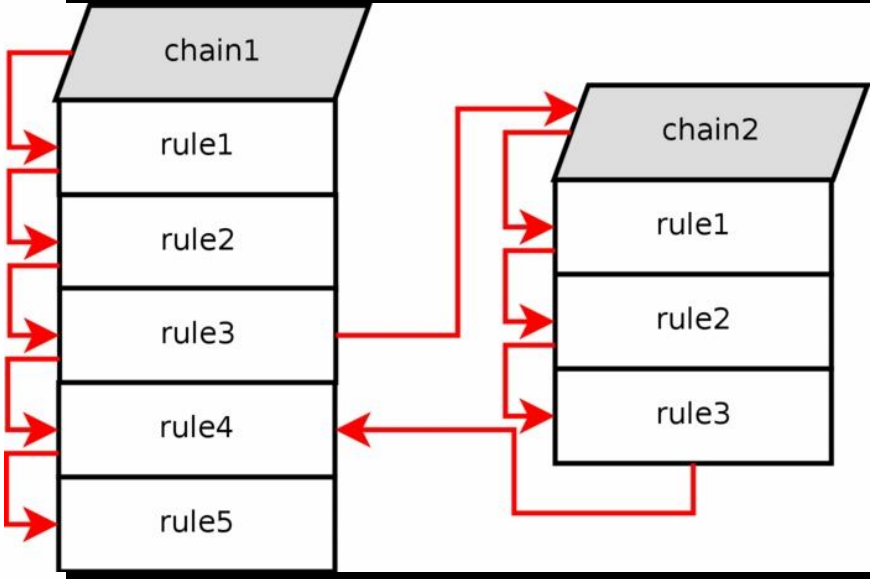
+ what is iptables?

iptables is the userspace command line program used to configure the linux packet filtering ruleset

+ a.k.a. firewall

+ iptable flow chart





what?

a packet filter is a piece of software that looks at the *headers* of packets as they pass through, and decides the fate of the entire packet

why?

+ **control**
  allow only what you specify
+ **security**
  protect against evil >:|
+ **watchfulness**
  alerts of abnormal activity

+ what is packet filtering and why do we need it?

+ filtering

traffic can be characterized by:

+ source and/or destination ip
  trust certain hosts

+ source and/or destination port
  allow specific services

+ protocol type
  tcp, udp, icmp, etc...

+ additional parameters
  e.g. state,

+ review: ports

Once a packet has reached its destination host, it is sent to a specific **port.**

+ about **65,000** available ports per host
+ the first 1024 are reserved to privileged processes such as daemons
+ **/etc/services** defines well-known ports
  e.g. Telnet:23, FTP:21, HTTP:80

+ review: protocols

Represent the kind of traffic being sent.

+ **tcp // transmission control protocol**
  maintains a connection between two hosts
+ **udp // user datagram protocol**
  sends data statelessly, without establishing a connection
+ **icmp // internet control message protocol**
  administrative functions such as PING

+ basic operations

-L List the rules.

-I <*n*> Insert a new rule before <*n*>.

-A Append a new rule at end of chain.

-R <*n*> Replace rule <*n*> with new rule.

-D <*n*> Delete rule <*n*>.

-F Flush the chain (delete all rules).

-X <chain> Delete the chain

-P <*p*> Set <*p*> as default policy for chain.

-t <table> Specify table. default is filter

-j <target> Jump to chain target

+ match criteria

-s <source>, -d <destination>

match source/destination ip

+ can use mask, e.g. 192.168.0.0/16

+ can precede with **!** to negate

-i <interface>, -o <interface>

match input/output interface

-m state --state <state1[,state2,...]>

NEW, ESTABLISHED, RELATED, etc...

+ match criteria

-p <protocol>

+ specify protocol: TCP, UDP, ICMP, ALL

-p <tcp|udp> --sport <p> or **--dport <p>**

+ match source or destination port(s)
+ for range > start[:end]
+ e.g. --sport 80, --dport !1:100

-p tcp --syn

+ new tcp connection request
+ !--syn means not new connection request

+ match criteria

-m multiport --sports <port,port>

-m multiport --dports <port,port>

+ list tcp/udp source or destination ports

+ don't have to be in a range

-m multiport --ports <port,port>

TCP/UDP ports, doesn't have to be in a range assumed source = destination

+ targets

LOG Make a log entry.

ACCEPT Allow packet.

REJECT Send back an error response.

DROP Ignore packet without responding.

DNAT Rewrite destination ip.

SNAT Rewrite source ip / ports.

MASQUERADE Used for source nat specify source ports.

+ states

NEW	New communication request.
ESTABLISHED	Reply to previous packet.
RELATED	Like ESTABLISHED, but for when the packet is not strictly a reply packet.

+ chain table

FILTER

```
   FORWARD       filtering through nics
   INPUT         filter going to server
   OUTPUT        filter leaving server
```

NAT

```
   PREROUTING    translation before route DNAT
   POSTROUTING   translation after route SNAT
   OUTPUT        translation on firewall -rare
```

MANGLE

```
   ^ all the above    Modify QOS bit in tcp
```

+ demo

[+] List rules

```
# iptables -L
>> Chain INPUT (policy ACCEPT)
>> target   prot   opt   source   destination
>> Chain FORWARD (policy ACCEPT)
>> target   prot   opt   source   destination
>> Chain OUTPUT (policy ACCEPT)
>> target   prot   opt   source   destination
```

[+] We start off with empty chains

+ demo

```
# ping x.x.x.x -c 1
>> PING x.x.x.x (x.x.x.x) 56(84) bytes of data.
>> 64 bytes from x.x.x.x: icmp_req=1 ttl=64 time=1.90 ms
>> --- x.x.x.x ping statistics ---
>> 1 packets transmitted, 1 received, 0% packet loss

# iptables -A INPUT -p icmp -j DROP
           |______| |_____| |_____|
              |       |       |
INPUT chain ----      |       |
                      |       |
ICMP protocol ----------      |
                              |
Jump to DROP target -----------
```

+ demo: block ping example

```
# ping x.x.x.x -c 1
>> PING x.x.x.x (x.x.x.x) 56(84) bytes of data.
>> --- x.x.x.x ping statistics ---
>> 1 packets transmitted, 0 received, 100% packet
loss...
```

[+] packets were dropped, let's try rejecting them now

```
# iptables -A INPUT -p icmp -j REJECT
                                |
Jump to REJECT target ------------

# ping x.x.x.x -c 1
>> From x.x.x.x icmp_seq=1 Destination Port Unreachable
>> --- x.x.x.x ping statistics ---
>> 1 pack..., 0 received, +1 errors, 100% packet loss...
```

+ demo: block ping example

inbound rule [1]

+ Allow any connection on loopback interface.
+ Necessary because services use loopback to communicate among themselves on the same machine.

```
  # iptables -A INPUT -i lo -j ACCEPT
             |______| |___| |_______|
                 |      |       |
INPUT chain ------      |       |
                        |       |
Loopback interface lo ---       |
                                |
Jump to ACCEPT target -----------
```

+ demo: building a simple firewall

inbound rule [2] + outbound rule [1]

+ Allow previously established connections for both INPUT and OUTPUT.

```
# iptables -A INPUT
           -m state --state ESTABLISHED,RELATED -j
ACCEPT
# iptables -A OUTPUT
           -m state --state ESTABLISHED,RELATED -j
ACCEPT
           |______| |_____________________________|
               |                  |
Match by state --                 |
                                  |
Specify state of these packets ----
```

+ demo: building a simple firewall

inbound rule [3+4]

+ Allow inbound SSH and web connection.

```
# iptables -A INPUT -p tcp --dport 22 -j ACCEPT
# iptables -A INPUT -p tcp --dport 80 -j ACCEPT
                    |____| |________|
                       |        |
TCP Protocol -----------        |
                                |
Destination port ---------------
```

+ demo: building a simple firewall

default policy

+ Now that we have specified all the traffic that we want to allow, we can deny everything else.

```
# iptables -P INPUT DROP
# iptables -P OUTPUT DROP
           |_____________|
                  |
Set policy -------
```

+ demo: building a simple firewall

Summary

```
+ Allow anything on loopback interface:
  # iptables -A INPUT -i lo -j ACCEPT
+ Allow previously established connections:
  # iptables -A INPUT
       -m state --state ESTABLISHED,RELATED -j ACCEPT
  # iptables -A OUTPUT
       -m state --state ESTABLISHED,RELATED -j ACCEPT
+ Allow inbound SSH connection:
  # iptables -A INPUT -p tcp --dport 22 -j ACCEPT
+ Allow inbound web connection:
  # iptables -A INPUT -p tcp --dport 80 -j ACCEPT
+ Set default policy for all other connections:
  # iptables -P INPUT DROP
  # iptables -P OUTPUT DROP
```

+ demo: building a simple firewall

Summary

```
# iptables -L -v

Chain INPUT (policy DROP)
target  prot .. in  .. src  dest
ACCEPT  all  .. lo  .. any  any
ACCEPT  all  .. any .. any  any   state REL,EST
ACCEPT  tcp  .. any .. any  any   tcp dpt:ssh
ACCEPT  tcp  .. any .. any  any   tcp dpt:www

> All done? Try restarting your machine...
```

+ demo: building a simple firewall

PERSISTANCE!!!

how to keep iptables saved!

+ save iptables

```
# iptables-save > /etc/default/iptables
```

+ modify /etc/network/interfaces and append to interface

```
# pre-ip iptables-restore < /etc/default/iptables
```

+ append to interfaces file (again) if you want to save changes before network restart

```
# post-down iptables-restore < /etc/default/iptables
```

+ advanced examples

+ advanced examples

Slow down em spammers

+ lets limit the amount of pings we can have per second

```
# iptables -A INPUT -p icmp --icmp-type echo-reply
      -m recent --name list --set

# iptables -A INPUT -m recent --name list
      --update --hitcount 20 -j DROP
```

+ advanced examples

let us only surf the world!

```
+ lets allow http/https out.

# iptables -A OUTPUT -j ACCEPT -m state --state
      NEW,ESTABLISHED,RELATED -o eth0 -p tcp
      -m multiport --dports 80,443

+ we also should keep established connections on
# iptables -A INPUT -j ACCEPT -m state --state
      ESTABLISHED,RELATED  -p tcp
```

+ advanced examples

lets log!

+ lets log all incoming icmp traffic.

```
# iptables -A INPUT -j LOG -p icmp
```

+ now lets tag this information for later

```
# iptables -A INPUT -j LOG -p icmp --log-prefix
"PING!!!"
```

+ lets ping! and then look at logs!

```
>> PING x.x.x.x (x.x.x.x) 56(84) bytes of data.
>> 64 bytes from x.x.x.x (x.x.x.x): icmp_req=1 ttl=64
time=0.086 ms

>> tail -f /var/log/syslog
Feb  ....  PING!!!! IN=lo OUT=  ....  ID=16559 SEQ=2
```

+ advanced examples

+ great iptable read

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

+ ubuntu's resources on iptables

https://help.ubuntu.com/community/IptablesHowTo

+ more iptables rules!

http://www.tty1.net/blog/2007-02-06-iptables-firewall_en.html

+ very detailed definitions

http://www.linuxtopia.org/Linux_Firewall_iptables/x2682.html

+ learning more


etley Hargrave