System Administration

Firewalls and Packet Filters

What is a firewall?

- A device that decides what packets may pass a certain point
- It may be a separate device that has more than one network interface
- It may be a piece of software on your computer

Why use firewalls?

- It can add another degree of security to a system or network
- Help protect 'lesser' systems

Reasons against using a firewall

- May cause other problems, especially with certain protocols
  - AFS
  - VOIP

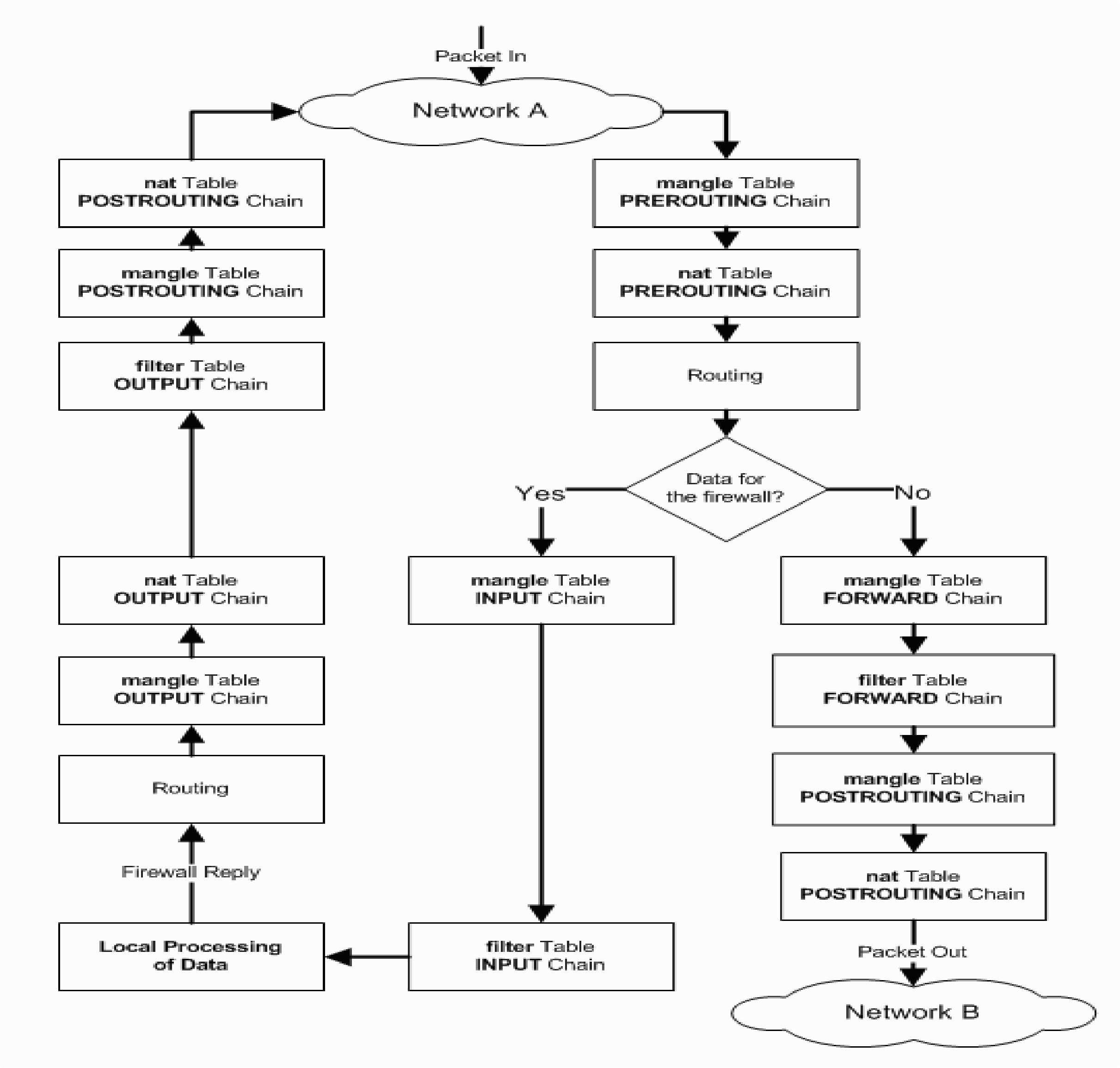
Firewalls are not a panacea

- Having a firewall does not give you absolute security
- Your firewall may be compromised or have holes
- It can only protect against stuff from outside of the firewall
- If a service you allow through has a compromise a firewall will do nothing

Iptables

- Linux 2.6 kernel uses Netfilter, and the interface to Netfilter is Iptables
- Kernel level packet management
  - You own the machine, you own the packets
  - This processing occurs before any user space processing (such as tcpwrappers, encryption, etc...)

Iptables


Packet In
Network A
nat Table POSTROUTING Chain
mangle Table PREROUTING Chain
mangle Table POSTROUTING Chain
nat Table PREROUTING Chain
filter Table OUTPUT Chain
Routing
Data for the firewall?
Yes
No
nat Table OUTPUT Chain
mangle Table INPUT Chain
mangle Table FORWARD Chain
mangle Table OUTPUT Chain
filter Table FORWARD Chain
Routing
mangle Table POSTROUTING Chain
Firewall Reply
nat Table POSTROUTING Chain
Local Processing of Data
filter Table INPUT Chain
Packet Out
Network B

How iptables works

- netfilter has three tables:
  - NAT changes the source or destination address of the packet. We will not cover this
  - mangle changes other parts in the packet. We will also not cover this
  - filter decides whether or not to let a packet pass through
- All three tables process packets but pass everything through by default

How iptables works

- Each table contains *chains* of rules
- Packets are compared to rules; actions are taken

```
<chain>
  |
  -> rule 1
  |
  -> rule 2
  .
  .
  .
```

How iptables works

- Every packet goes through one of three chains: input, output or forward:
  - input: packets that come in an interface destined for this machine
  - output: packets that originate on this machine going out an interface
  - forward: packets coming in this machine and then going back out. We will not cover this

Path of Packets

- Packet enters an interface
  - enters INPUT chain
  - filter table directs it to local processes
- Packet is created by a local process
  - enters OUTPUT chain

How Iptables works

- Each chain has a default rule called a **policy**
- If no rule in the chain matches, the default rule is applied
- Otherwise the first rule that matches applies

Enabling Iptables

- `chkconfig –level 35 iptables on`
- `service iptables on`

Saving rules

- The iptables command allows you to add or remove rules
- Those rules are only active until the machine is rebooted
- `service iptables save` saves the rules in /etc/sysconfig/iptables

Building rules

- Do it the easy way:

```
system-config-securitylevel
```

- Do it the harder way:
  - writing your own rules

`system-config-securitylevel`

- As root, type the command:

  ```
  system-config-securitylevel
  ```

- `Security level` enables or disables the firewall
- `Trusted devices` marks devices that are trusted no matter what
  - even if other blocks are made, any traffic to or from these devices goes through

`system-config-securitylevel`

- `Trusted services` are incoming services that are allowed through
- Click on the `OK` button to make changes
- It saves the firewall config in `/etc/sysconfig/iptables`
- *Warning*: if you use system-config-securitylevel and have previously stored rules in `/etc/sysconfig/iptables` they will be deleted

Creating your own rules

- You may be trying to do something more complicated than system-config-securitylevel allows you to do

Basic iptables operations

- `iptables -L` *`chain`*

  List the rules in a particular chain

- `iptables -F` *`chain`*

  Flush all the rules in a chain

- `iptables -P` *`chain policy`*

  Set the default policy on a chain. *Policy* can be either `ACCEPT` or `DROP`

Which Default Policy?

- ACCEPT means that you have to explicitly block packets
  - It is the easiest to use, but if you forget to block something it will get through
- DROP means that you have to explicitly allow packets
  - It is harder to use as you will have to specify everything you allow, but is the most secure

Adding Rules

- `iptables -A` *`chain rule`*

  Appends a rule to the end of a chain

- `iptables -I` *`chain number rule`*

  Place a rule at a specific number on a chain

```
iptables -A INPUT -s 127.0.0.1 -j DROP
iptables -I INPUT 1 -s 127.0.0.1 -j
  DROP
```

Deleting Rules

- `iptables -D` *chain number*

  Delete a specific number rule from a chain

- `iptables -D` *chain rule*

  Delete the first rule that matches *rule* from a chain

```
iptables -D INPUT 1
iptables -D INPUT -s 127.0.0.1 -j DROP
```

Source and Destination

- `-s` *`address`*

  Specifies a source address

- `-d` *`address`*

  Specifies a destination address

- *`address`* is in the form
  - `129.186.1.200`
  - `129.186.1.0/24`
  - `129.186.1.0/255.255.255.0`

Protocol, Port and Interface

- `-p` *`proto`*

  Matches a particular protocol. Common ones are `TCP, UDP, ICMP`

- `--sport` *`port`* `--dport` *`port`*

  Matches a particular *source* or *destination* port

  Can be either a number or a symbolic name (ssh)

- `-i interface  -o interface`

  Matches traffic coming *in* or going *out* a particular interface

Accepting or Rejecting

- `-j DROP`

  Drops

- `-j ACCEPT`

  Accepts

- `-j REJECT`

  Rejects a connection

Negation

- Addresses, protocols, ports and interfaces allow you to also negate using `!`

```
-s !127.0.0.1
-d !129.186.0.0/16
-p !TCP
--sport !80
--dport !22
-i !eth0
-o !lo
```

Sample Rules

```
iptables -P INPUT DROP

iptables -P OUTPUT ACCEPT

iptables -A INPUT -i lo -s 127.0.0.1 -j ACCEPT

iptables -A INPUT -p TCP --dport ssh -j ACCEPT

iptables -A INPUT -p TCP --dport 80 -j ACCEPT

iptables -A OUTPUT -o eth0 -p tcp --dport 25 -d
  129.186.140.5 -j ACCEPT

iptables -A OUTPUT -o eth0 -p tcp --dport 25 -j DROP
```

Exercise

- Find a partner
- Figure out each other's ip address

A brief introduction to nc

- `nc -l -p <portnumber>`
  - listens for incoming connections on <portnumber>
  - the “server side”
- `nc -vvv <hostname> <portnumber>`
  - makes a connection to <hostname> on <portnumber>
  - -vvv makes it verbose
  - the “client side”

nc example

- One partner should run
  - `nc -l -p 10137`
- The other should run
  - `nc -vvv <remoteip> 10137`
- Start typing to each other, `control-c` to quit

block all traffic from a host

- The “server side” person should add the following rule
  - `iptables -A INPUT -s <clientip> -j DROP`
  - `nc -l -p 10137`
- The “client side” person should now try to make a connection
  - `nc -vvv <serverip> 10137`
- Does it work? How long did it take?

block all traffic, continued

- “Server side” person should do:

  -
    ```
    iptables -D INPUT -s <clientip> -j DROP
    ```

  -
    ```
    iptables -A INPUT -s <clientip> -j REJECT
    ```
- “Client side” person should try:

  -
    ```
    nc -vvv <serverip> 10137
    ```
- How did that work? Can you ping the “server”?

block some traffic

- The “server side” should run
  - `iptables -D INPUT -s <clientip> -j REJECT`
  - `iptables -A INPUT -s <clientip> -p tcp --dport 10137 -j REJECT`
- The “client side” should run
  - `nc -vvv <serverip> 10137`
- Can you connect? Can you ping?

block some traffic

- The “server side” should try

  -
    ```
    nc -l -p 10138
    ```
- The “client side” should try

  -
    ```
    nc -vvv <serverip> 10138
    ```
- Can you connect?

The Order of Rules Count

- “Server side” should do

  -
    ```
    iptables -F INPUT
    ```

  -
    ```
    iptables -A INPUT -s <clientip>
      -p tcp --dport 10137 -j REJECT
    ```

  -
    ```
    iptables -A INPUT -s <clientip> -j ACCEPT
    ```
- “Client side” should try

  -
    ```
    nc -vvv <serverip> 10137
    ```
- Can you connect?