Section Overview

What You Will Learn

- Securing network services.
- Configuring local Linux firewalls.
- Preventing information leakage.
- Port scanning.
- Xinetd security.
- Securing SSH.

Network Security

Network Services

- Network services, daemons, servers.
- Listen on network ports.
- Constantly running in the background.
- Output recorded in log files.
- Designed to perform a single task.

Securing Network Services

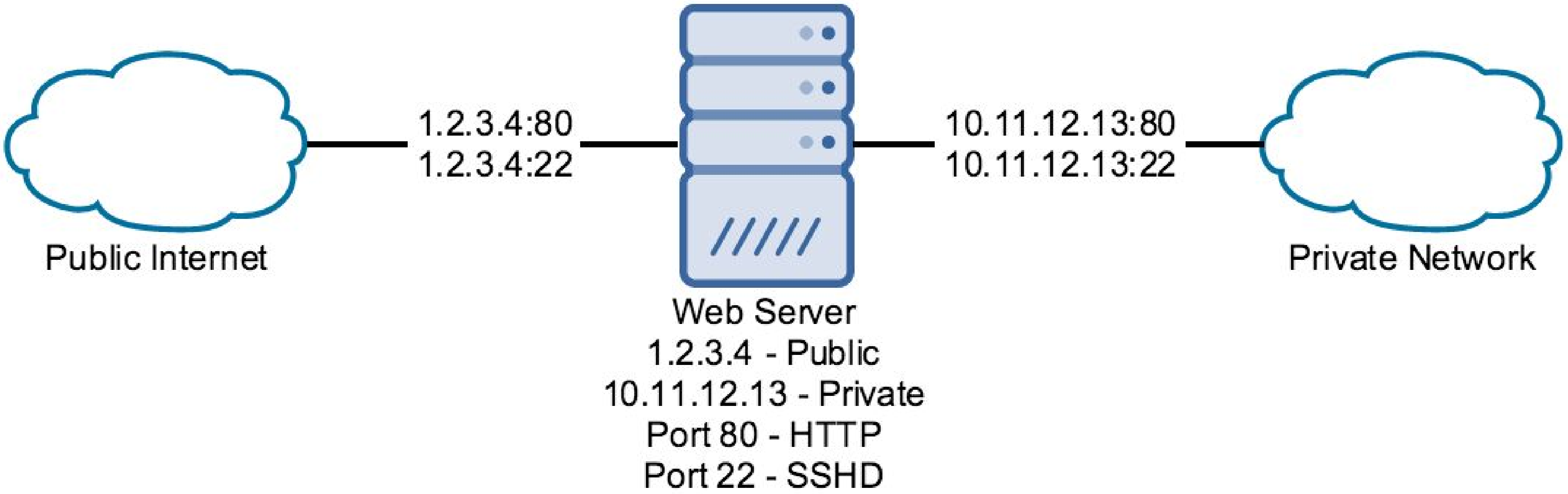
- Use a dedicated user for each service.
  - Take advantage of privilege separation.
- Ports below 1024 are privileged.
  - Use root to open them, then drop privileges.
  - Configuration controlled by each service.

Securing Network Services

- Stop and uninstall unused services.
- Avoid unsecure services.
  - Use SSH instead of telnet, rlogin, rsh, and FTP
- Stay up to date with patches.
  - Install services provided by your distro.

Securing Network Services

- Only listen on the required interfaces and addresses.



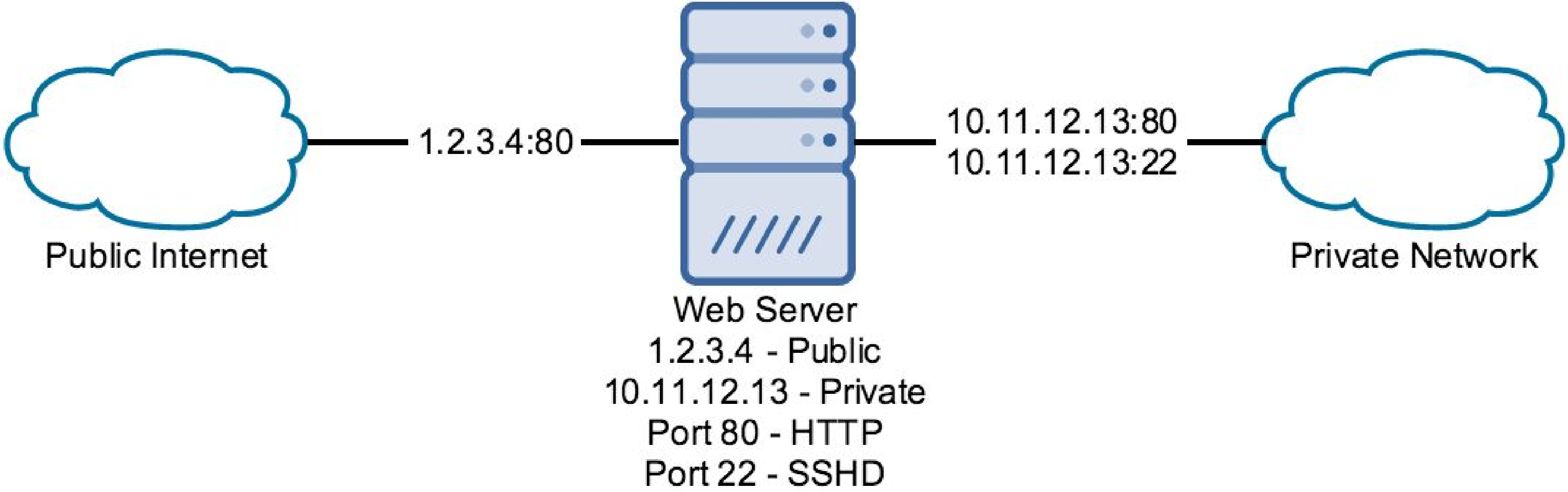
Public Internet

1.2.3.4:80
1.2.3.4:22

Web Server
1.2.3.4 - Public
10.11.12.13 - Private
Port 80 - HTTP
Port 22 - SSHD

10.11.12.13:80
10.11.12.13:22

Private Network



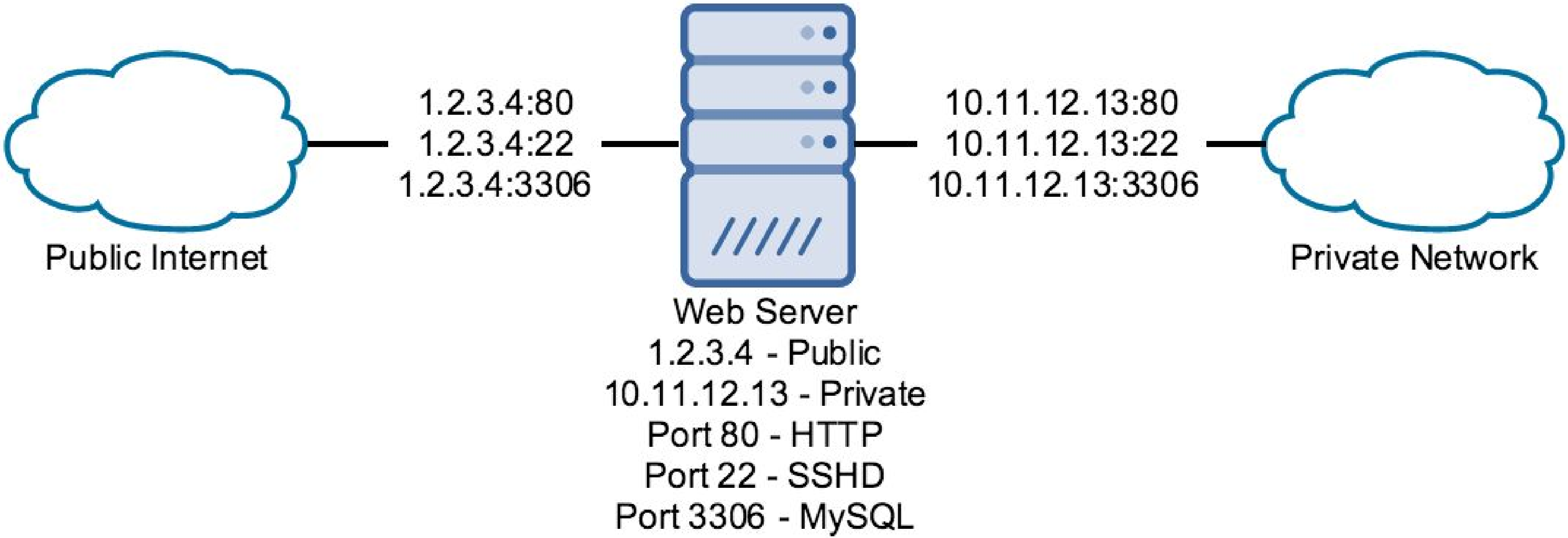
Public Internet

1.2.3.4:80

Web Server
1.2.3.4 - Public
10.11.12.13 - Private
Port 80 - HTTP
Port 22 - SSHD

10.11.12.13:80
10.11.12.13:22

Private Network



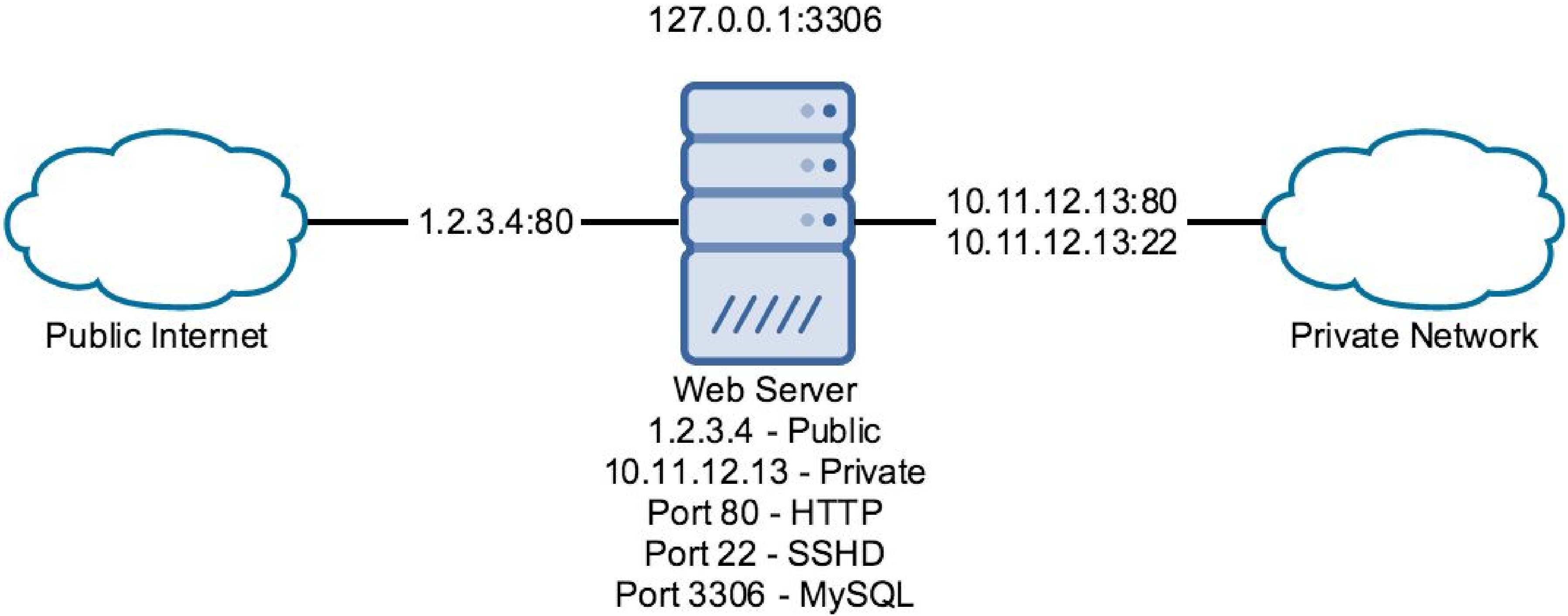
Public Internet

1.2.3.4:80
1.2.3.4:22
1.2.3.4:3306

Web Server
1.2.3.4 - Public
10.11.12.13 - Private
Port 80 - HTTP
Port 22 - SSHD
Port 3306 - MySQL

10.11.12.13:80
10.11.12.13:22
10.11.12.13:3306

Private Network



127.0.0.1:3306

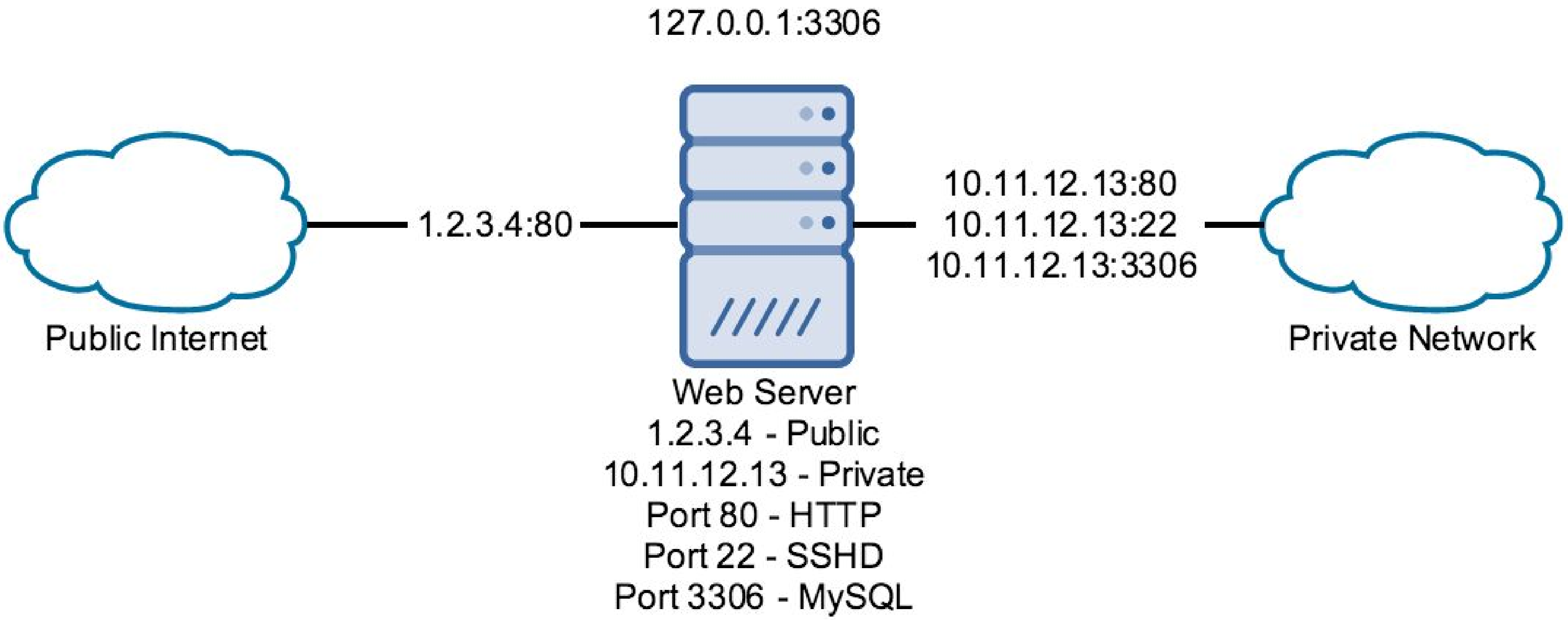
Public Internet

1.2.3.4:80

10.11.12.13:80
10.11.12.13:22

Private Network

Web Server
1.2.3.4 - Public
10.11.12.13 - Private
Port 80 - HTTP
Port 22 - SSHD
Port 3306 - MySQL



127.0.0.1:3306

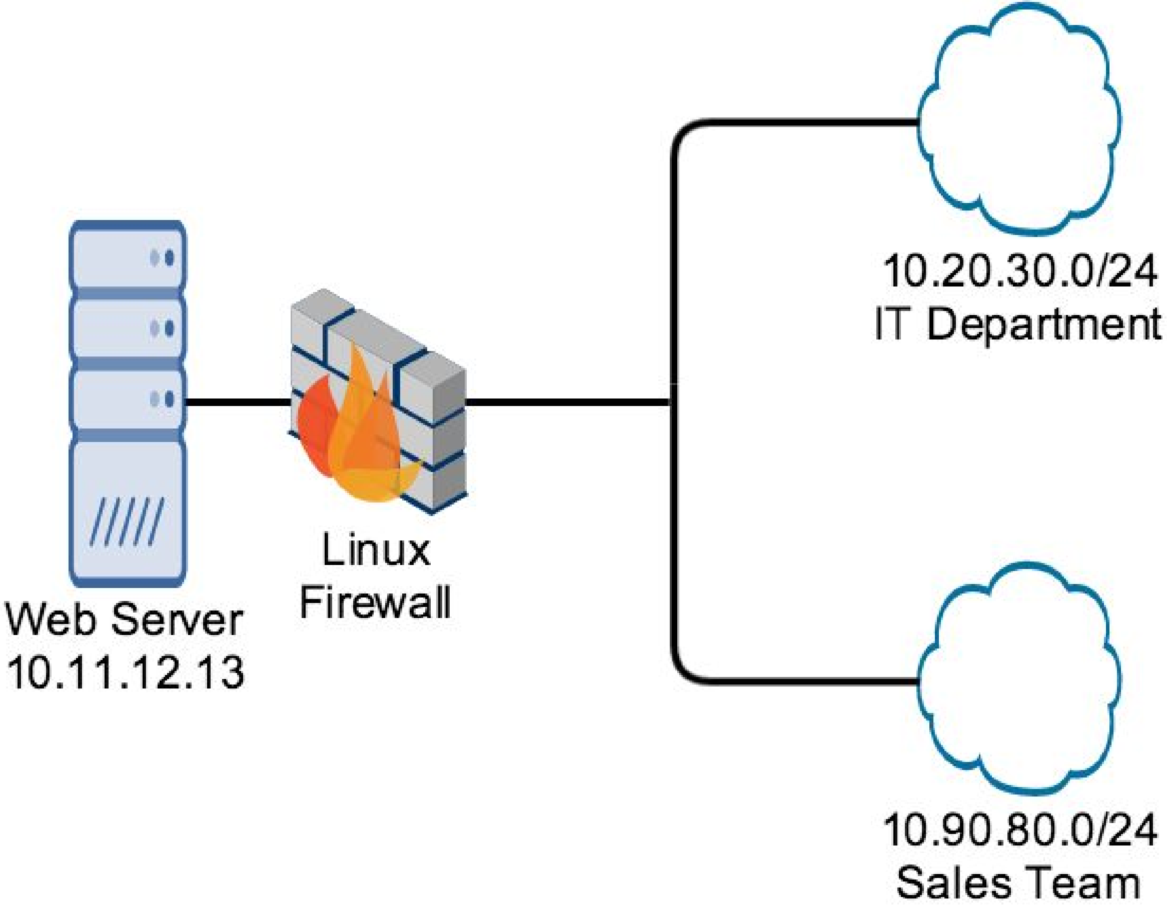
Public Internet

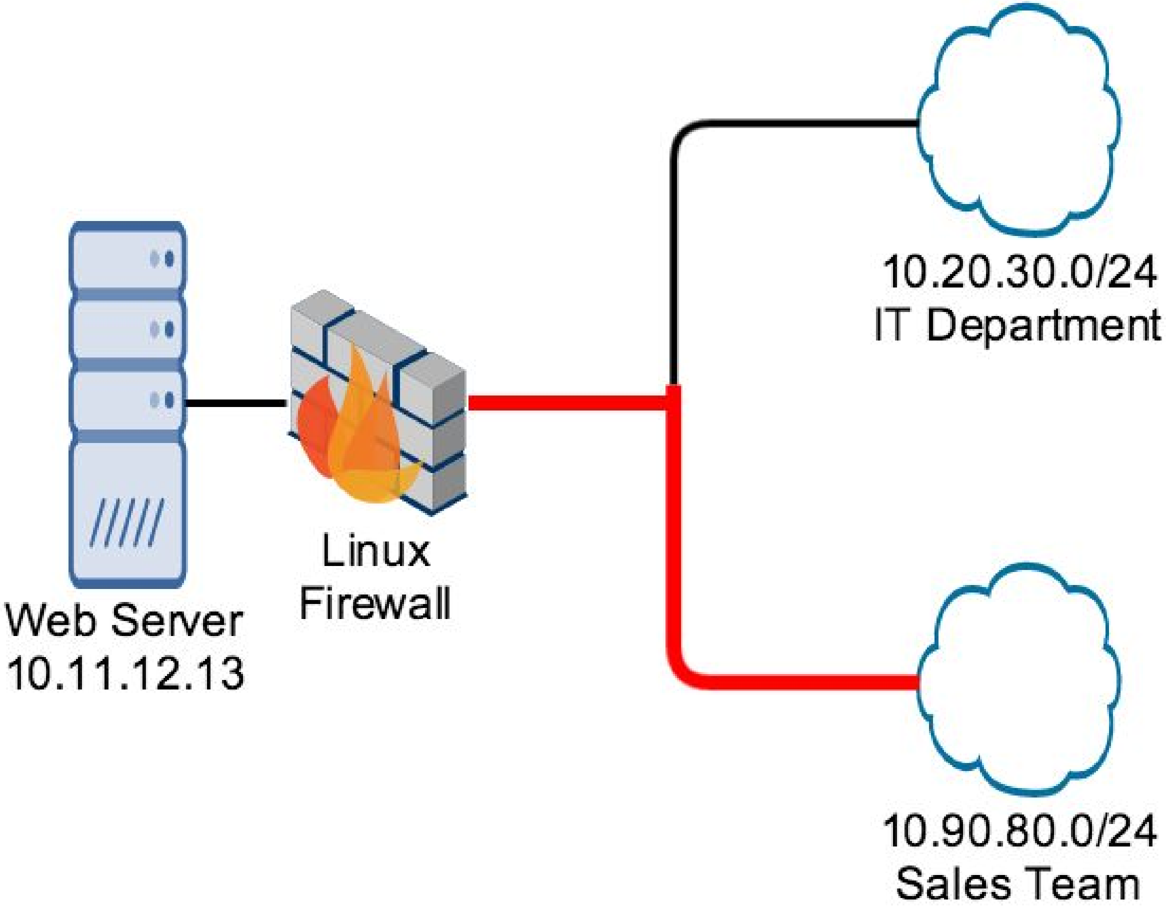
1.2.3.4:80

10.11.12.13:80
10.11.12.13:22
10.11.12.13:3306

Private Network

Web Server
1.2.3.4 - Public
10.11.12.13 - Private
Port 80 - HTTP
Port 22 - SSHD
Port 3306 - MySQL


Web Server
10.11.12.13
Linux
Firewall
10.20.30.0/24
IT Department
10.90.80.0/24
Sales Team

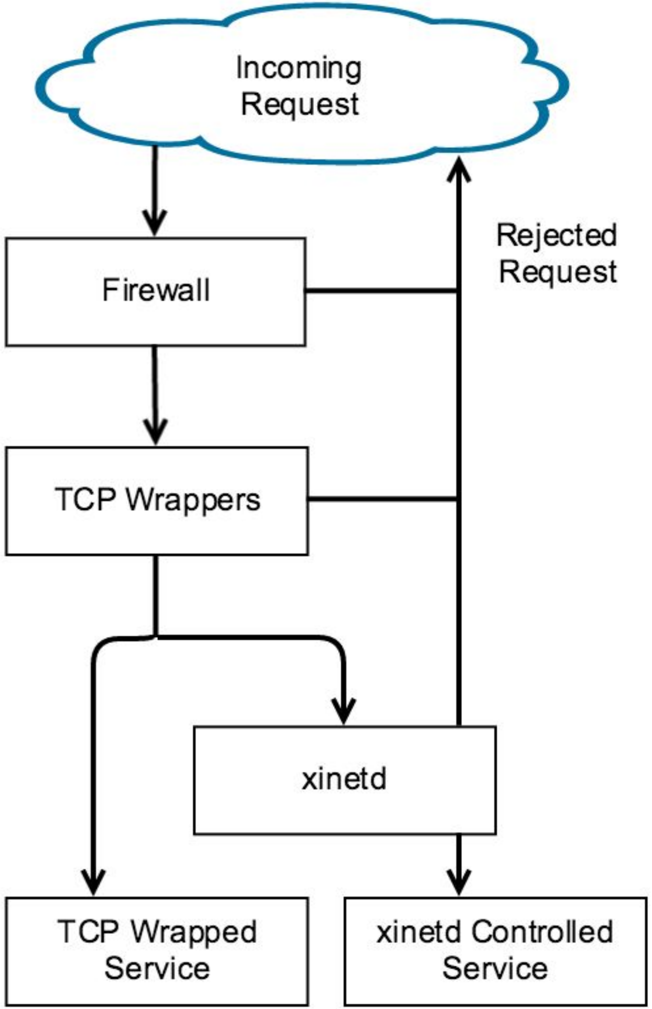


Web Server
10.11.12.13

Linux
Firewall

10.20.30.0/24
IT Department

10.90.80.0/24
Sales Team


Incoming Request
Firewall
TCP Wrappers
xinetd
TCP Wrapped Service
xinetd Controlled Service
Rejected Request

Information Leakage

- Avoid revealing information where possible.

Web Server Information Leakage

```
$ curl -I http://192.168.19.33
HTTP/1.1 200 OK
Date: Fri, 05 Feb 2016 17:13:11 GMT
Server: Apache/2.4.6 (CentOS)
Last-Modified: Thu, 16 Oct 2014 13:20:58 GMT
ETag: "1321-5058a1e728280"
Accept-Ranges: bytes
Content-Length: 4897
Content-Type: text/html; charset=UTF-8
```

Information Leakage

- Avoid revealing information where possible.
- Web server banners.
- `/etc/motd`
- `/etc/issue`
- `/etc/issue.net`

Displaying Services with `systemctl`

```
# systemctl
 UNIT           LOAD   ACTIVE SUB      DESCRIPTION
...
httpd.service  loaded active running  Apache HTTP Server
...
sshd.service   loaded active running  OpenSSH server
...
```

Stop and Disable Services

```
systemctl stop SERVICE
systemctl disable SERVICE
```

Example:

```
systemctl stop httpd
systemctl disable httpd
```

List Listening Programs with `netstat`

```
# netstat -nutlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address Foreign Address  State   PID/Program name
tcp        0      0 0.0.0.0:22    0.0.0.0:*        LISTEN  5089/sshd
tcp        0      0 127.0.0.1:25  0.0.0.0:*        LISTEN  1398/master
udp        0      0 0.0.0.0:68    0.0.0.0:*                6732/dhclient
```

List Listening Programs with `netstat`

```
# netstat -nutlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address Foreign Address  State   PID/Program name
tcp        0      0 0.0.0.0:22    0.0.0.0:*        LISTEN  5089/sshd
tcp        0      0 127.0.0.1:25  0.0.0.0:*        LISTEN  1398/master
udp        0      0 0.0.0.0:68    0.0.0.0:*                6732/dhclient
```

Port Scanning

```
nmap HOSTNAME_OR_IP

nmap localhost
nmap 10.11.12.13
```

```
# nmap 127.0.0.1
Starting Nmap 6.40 ( http://nmap.org ) at
2016-02-06 01:59 EST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000040s latency).
Not shown: 997 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
25/tcp open  smtp
80/tcp open  http
```

```
# nmap 10.11.12.13
Starting Nmap 6.40 ( http://nmap.org ) at
2016-02-06 01:59 EST
Nmap scan report for linuxsvr (10.11.12.13)
Host is up (0.0000040s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
```

```
# lsof -i
COMMAND  PID USER  FD  TYPE DEVICE SIZE/OFF NODE NAME
httpd   4893 root  4u  IPv6  54078    0t0  TCP *:http (LISTEN)
sshd    5089 root  3u  IPv4  56221    0t0  TCP *:ssh (LISTEN)
sshd     6770   root    3u  IPv4  76021      0t0  TCP
192.168.1.166:ssh->182.168.1.148:53132 (ESTABLISHED)
```

Testing a Specific Port

```
telnet HOST_OR_ADDRESS PORT
```

Testing a Specific Port

```
telnet HOST_OR_ADDRESS PORT
```

```
nc -v HOST_OR_ADDRESS PORT
```

Xinetd Controlled Services

```
/etc/xinetd.d/SERVICE
```

To disable service:

```
disable = yes
```

To disable xinetd:

```
systemctl stop xinetd
systemctl disable xinetd
```

Securing SSH

Securing SSH

- SSH = Secure SHell.
- Allows for key based authentication.

/etc/ssh/sshd_config:

```
PubkeyAuthentication yes
```

Creating SSH Keys

- Use the `ssh-keygen` command to create a key.
- You can create a passphrase for the key.
- Creates `~/.ssh/id_rsa` and `~/.ssh/id_rsa.pub`.

Add the Public Key to the Remote Host

- To copy the key, use `ssh-copy-id`:

```
ssh-copy-id [user@]host
```

- Adds public key to:

```
~/.ssh/authorized_keys
```

Force Key Authentication

In /etc/ssh/sshd_config:

```
PasswordAuthentication no
```

Controlling Root Logins

- To disable root logins:

```
PermitRootLogin no
```

- To only allow root to login with a key:

```
PermitRootLogin without-password
```

Only Allow Certain Users SSH Access

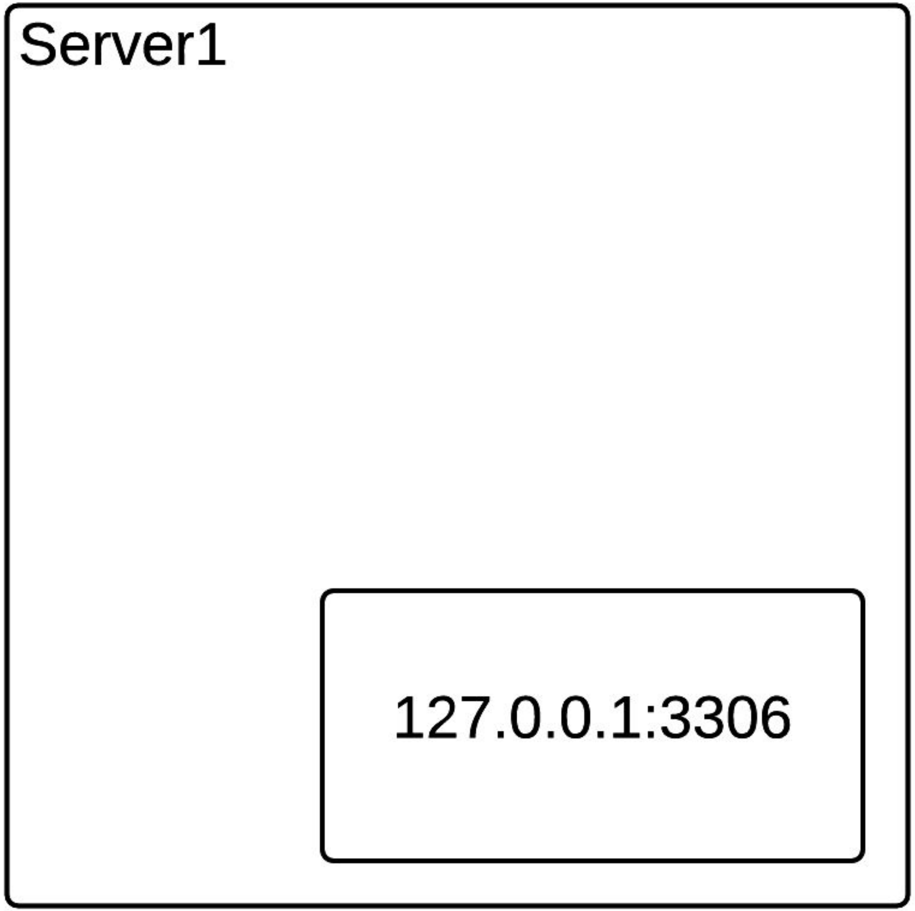
```
AllowUsers user1 user2 userN
```

Only Allow Certain Groups SSH Access

```
AllowGroups group1 group2 groupN
```

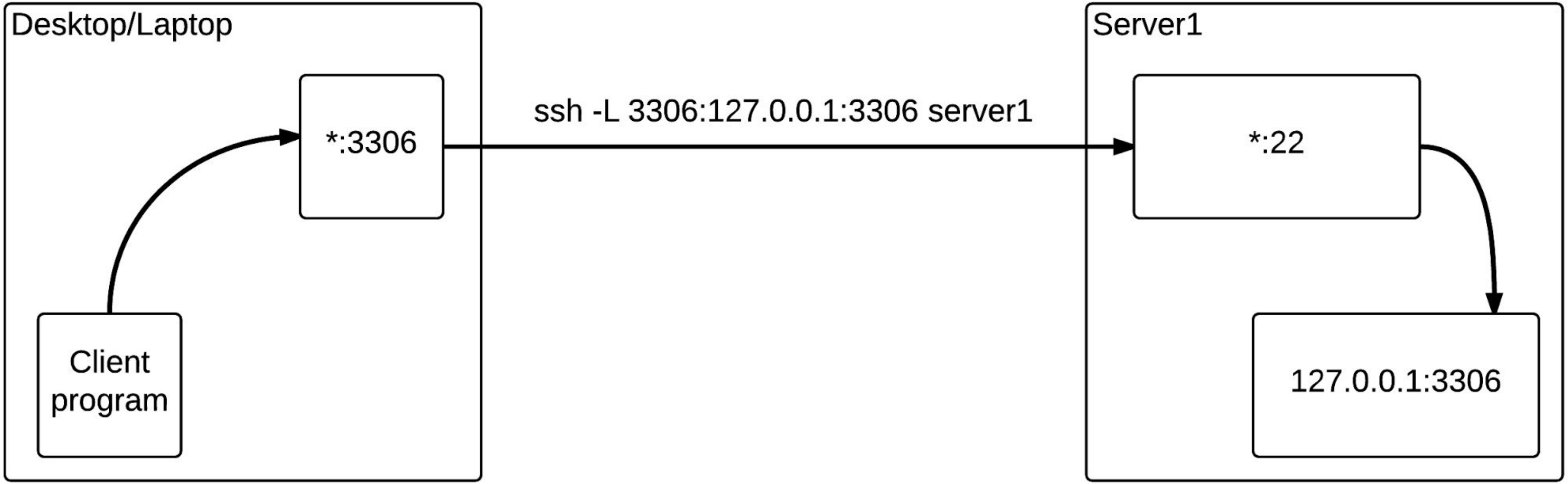
Deny Certain Users SSH Access

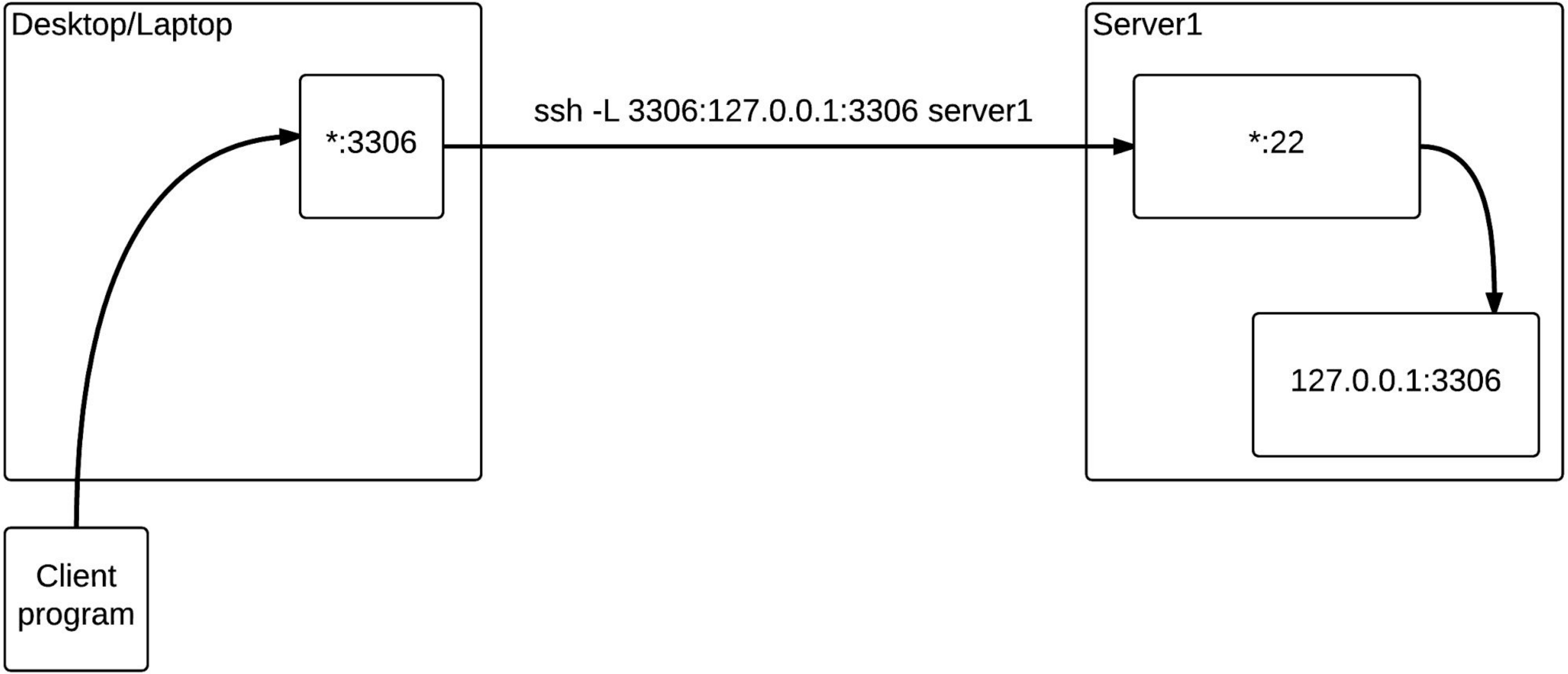
```
DenyUsers  user1  user2  userN
DenyGroups group1 group2 groupN
```



Server1
127.0.0.1:3306

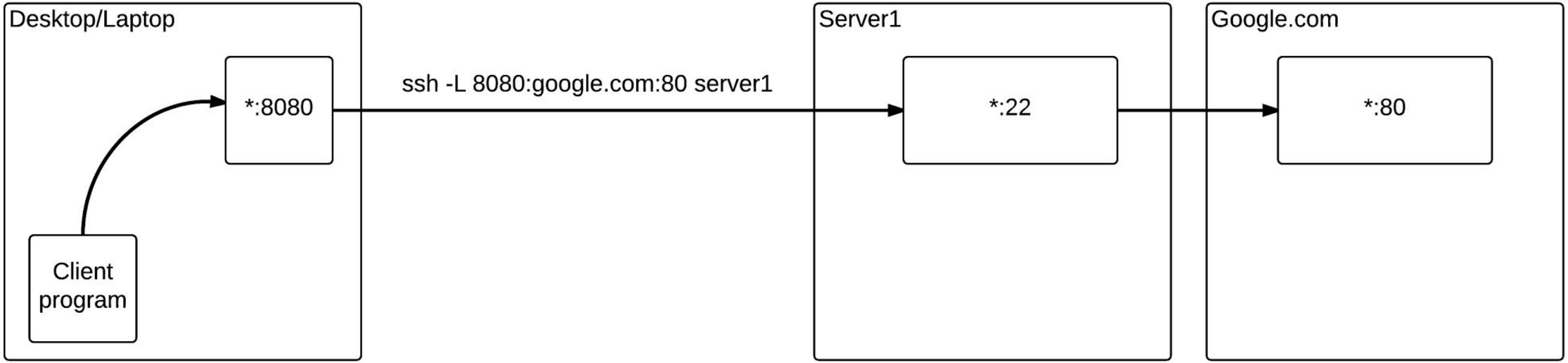
SSH Port Forwarding





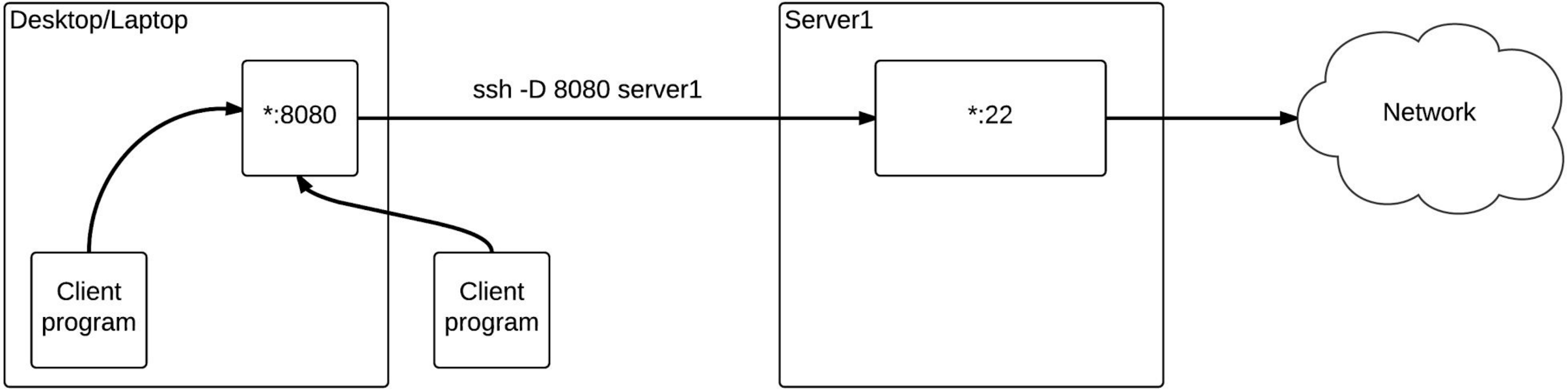
```
ssh -L 3306:127.0.0.1:3306 server1
```

SSH Port Forwarding

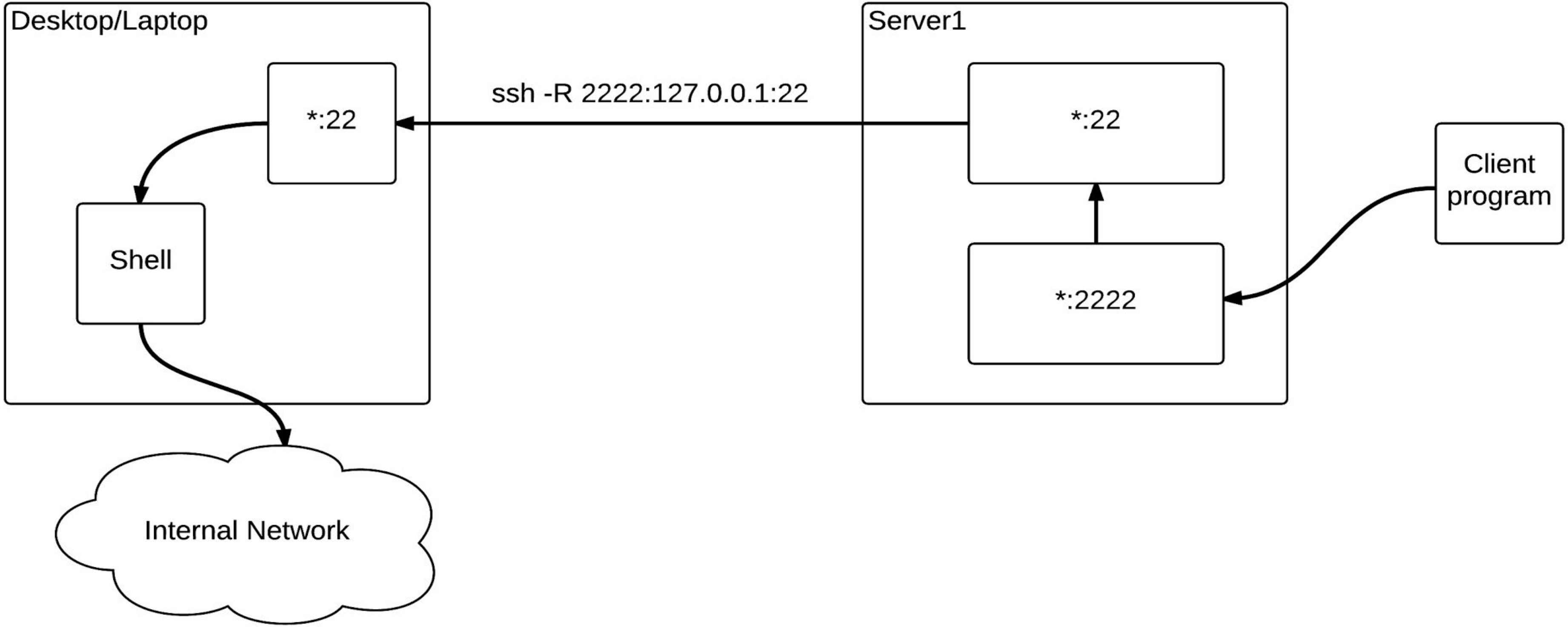


LinuxTrainingAcademy.com

Dynamic Port Forwarding / SOCKS



Reverse Port Forwarding



Disable TCP Port Forwarding

```
AllowTcpForwarding no

GatewayPorts no
```

Use SSHv2 instead of SSHv1

```
Protocol 2
```

Bind SSH to a Specific Address

```
ListenAddress host_or_address1
ListenAddress host_or_addressN
```

Change the Default Port

In /etc/ssh/sshd_config:

```
Port 2222
```

Add the New Port to SELinux

```
semanage port -a -t ssh_port_t -p tcp PORT
```

```
semanage port -l | grep ssh
```

Disable the Banner

```
Banner none

# Banner /etc/issue.net
```

Reload the Configuration

```
systemctl reload sshd
```

For More Information

```
man ssh
man sshd
man sshd_config
```

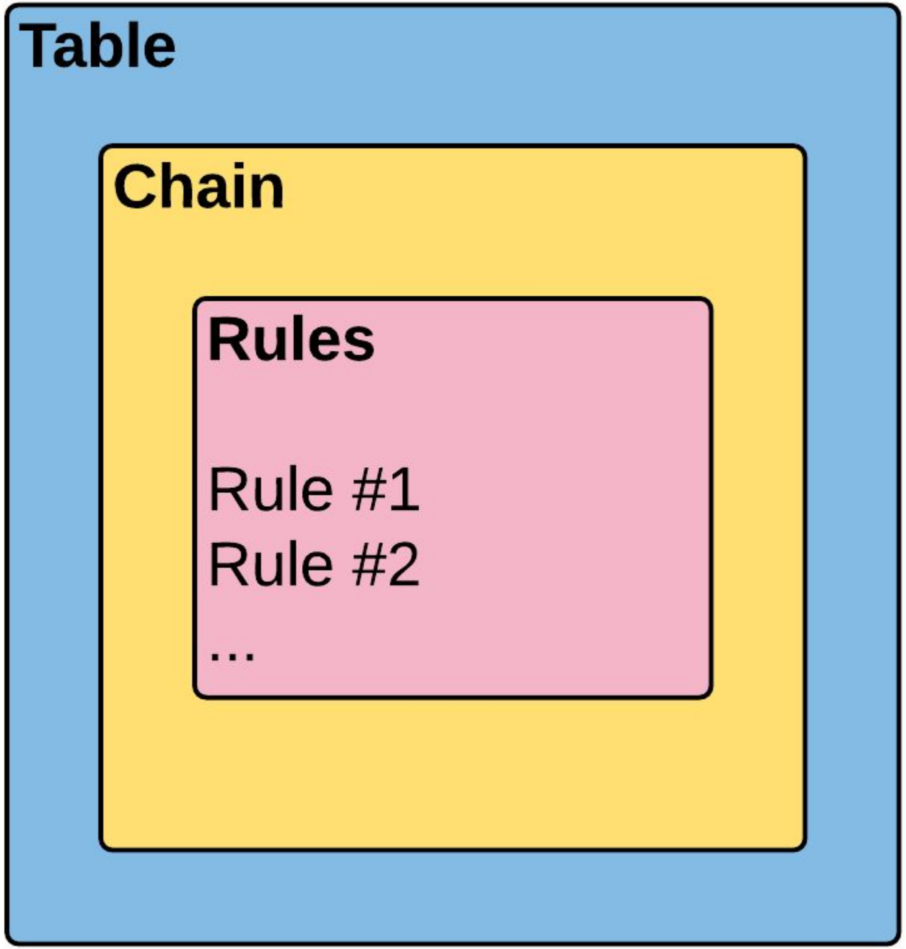
Linux Firewall

Netfilter and IPTables

Rule Specifications

Linux Firewall

- Firewalls control network access.
- Linux firewall = Netfilter + IPTables
- Netfilter is a kernel framework.
- IPTables is a packet selection system.
- Use the `iptables` command to control the firewall.

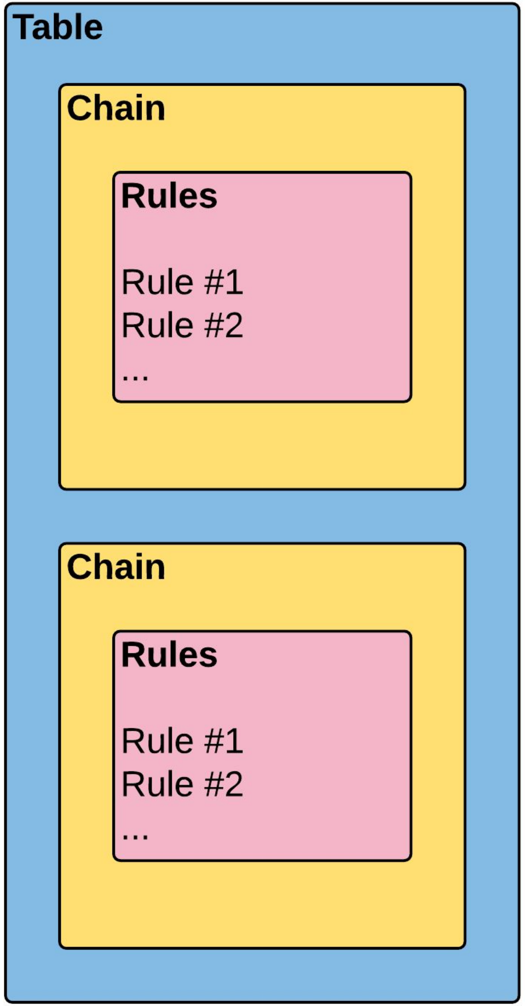


Table

Chain

Rules

Rule #1
Rule #2
...



Table

Chain

Rules

Rule #1
Rule #2
...

Chain

Rules

Rule #1
Rule #2
...

ıy.com

Default Tables

- Filter
- NAT
- Mangle
- Raw
- Security

Default Tables

- Filter - Most commonly used table.
- NAT - Network Address Translation.
- Mangle - Alter packets.
- Raw - Used to disable connection tracking.
- Security - Used by SELinux.

Default Chains

- INPUT
- OUTPUT
- FORWARD
- PREROUTING
- POSTROUTING

	INPUT	OUTPUT	FORWARD	PREROUTING	POSTROUTING
Filter	X	X	X		
Nat	X	X		X	X
Mangle	X	X	X	X	X
Raw		X		X	
Security	X	X	X		

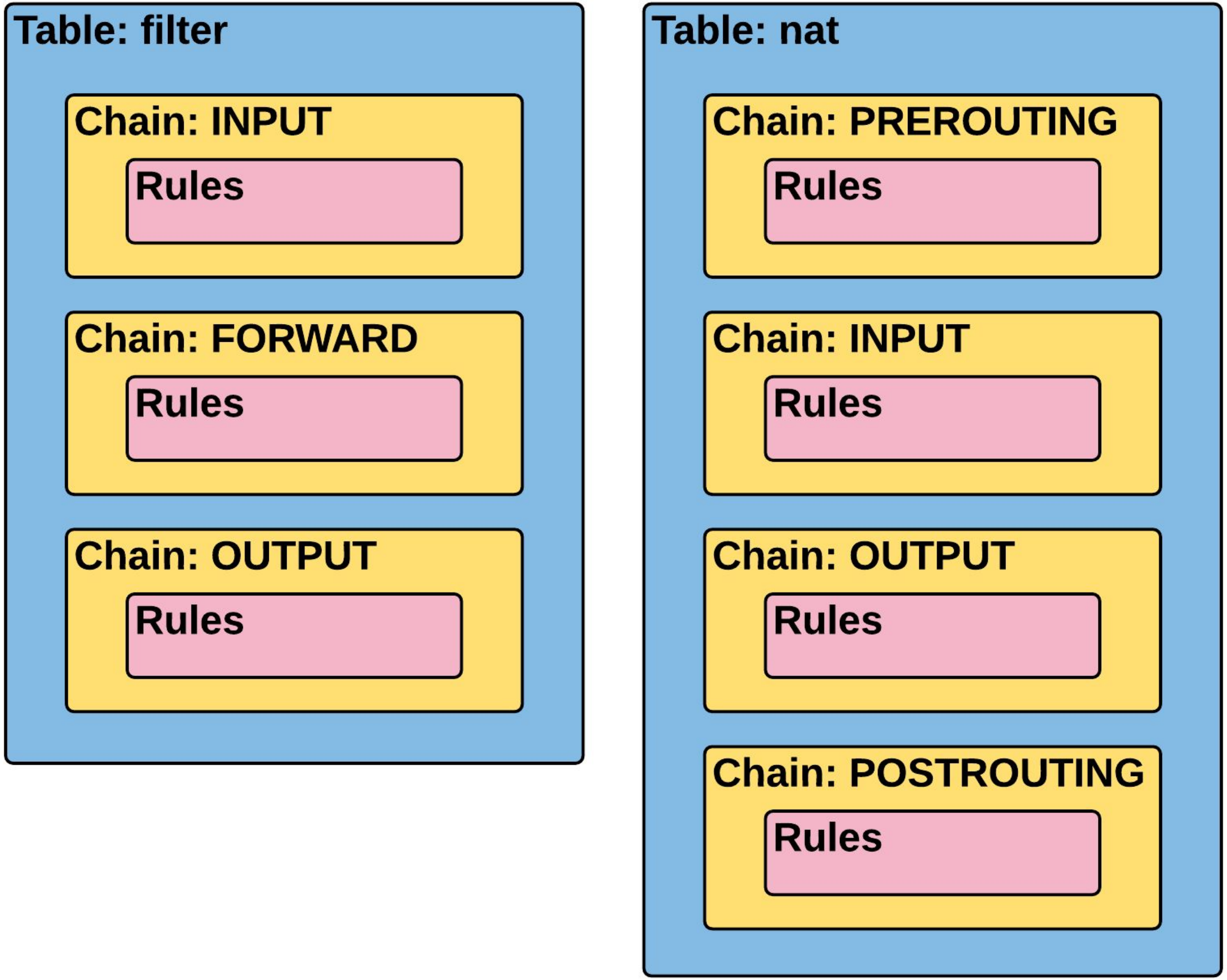


Table: filter

Chain: INPUT

Rules

Chain: FORWARD

Rules

Chain: OUTPUT

Rules

Table: nat

Chain: PREROUTING

Rules

Chain: INPUT

Rules

Chain: OUTPUT

Rules

Chain: POSTROUTING

Rules

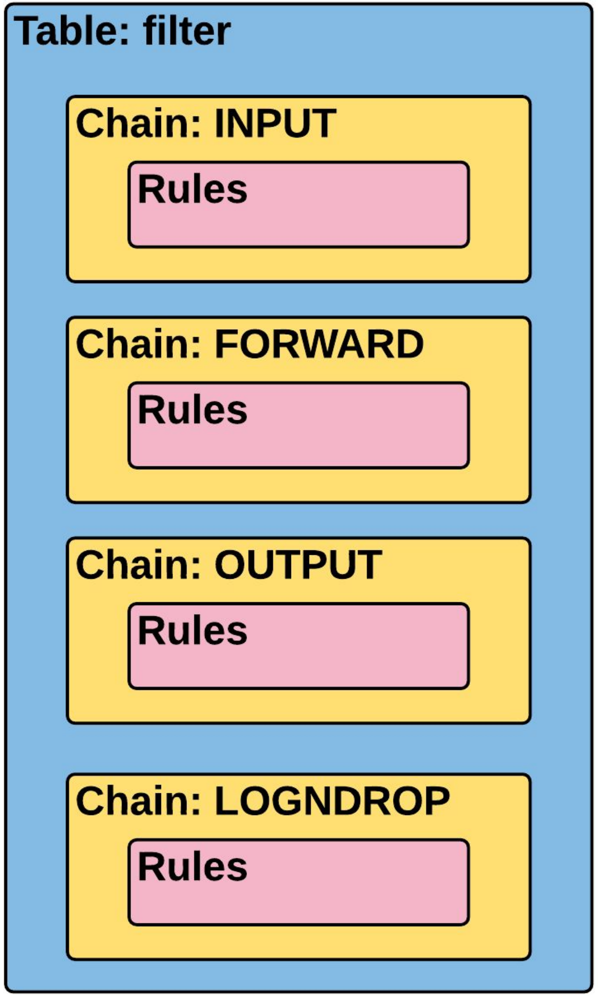


Table: filter

Chain: INPUT

Rules

Chain: FORWARD

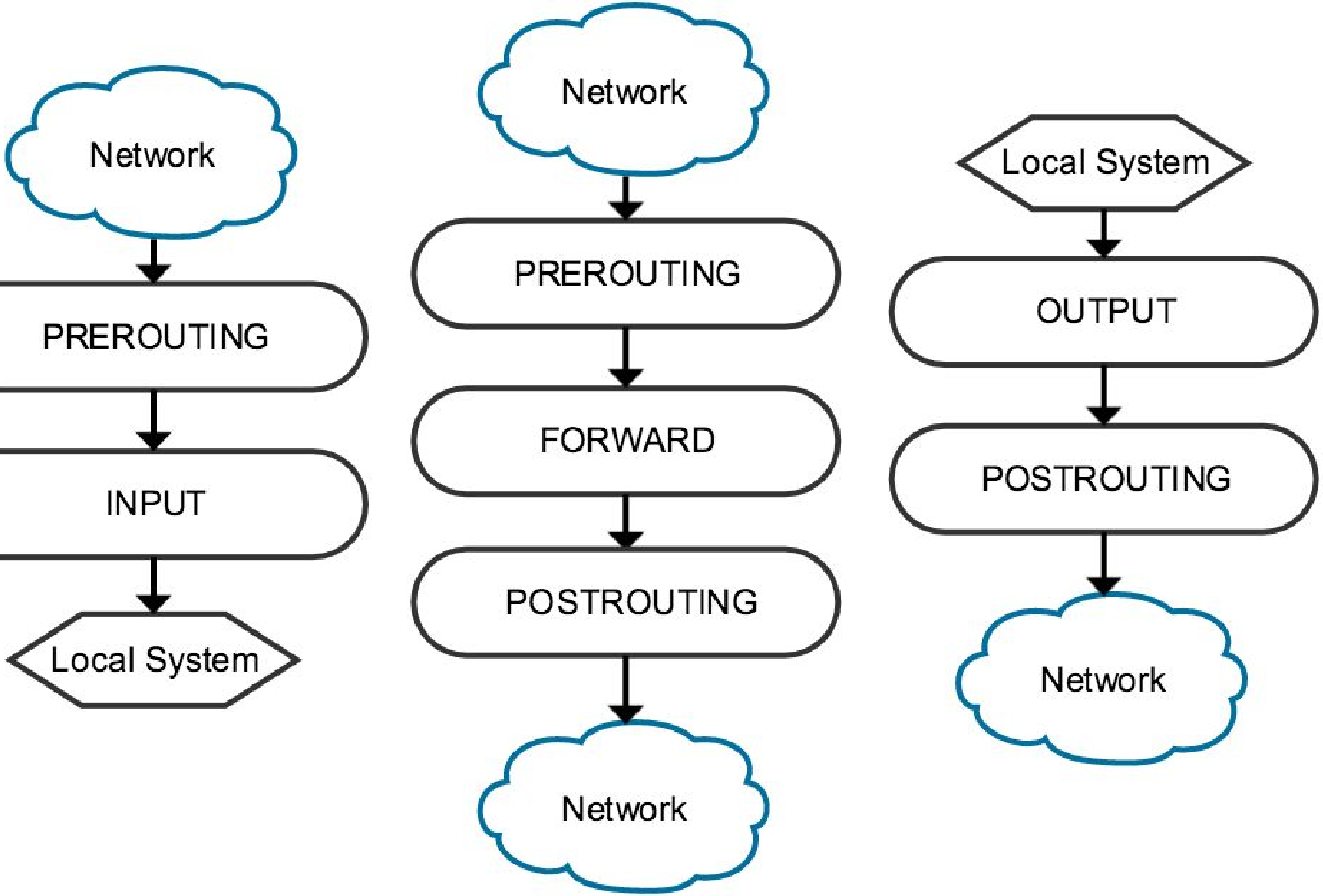
Rules

Chain: OUTPUT

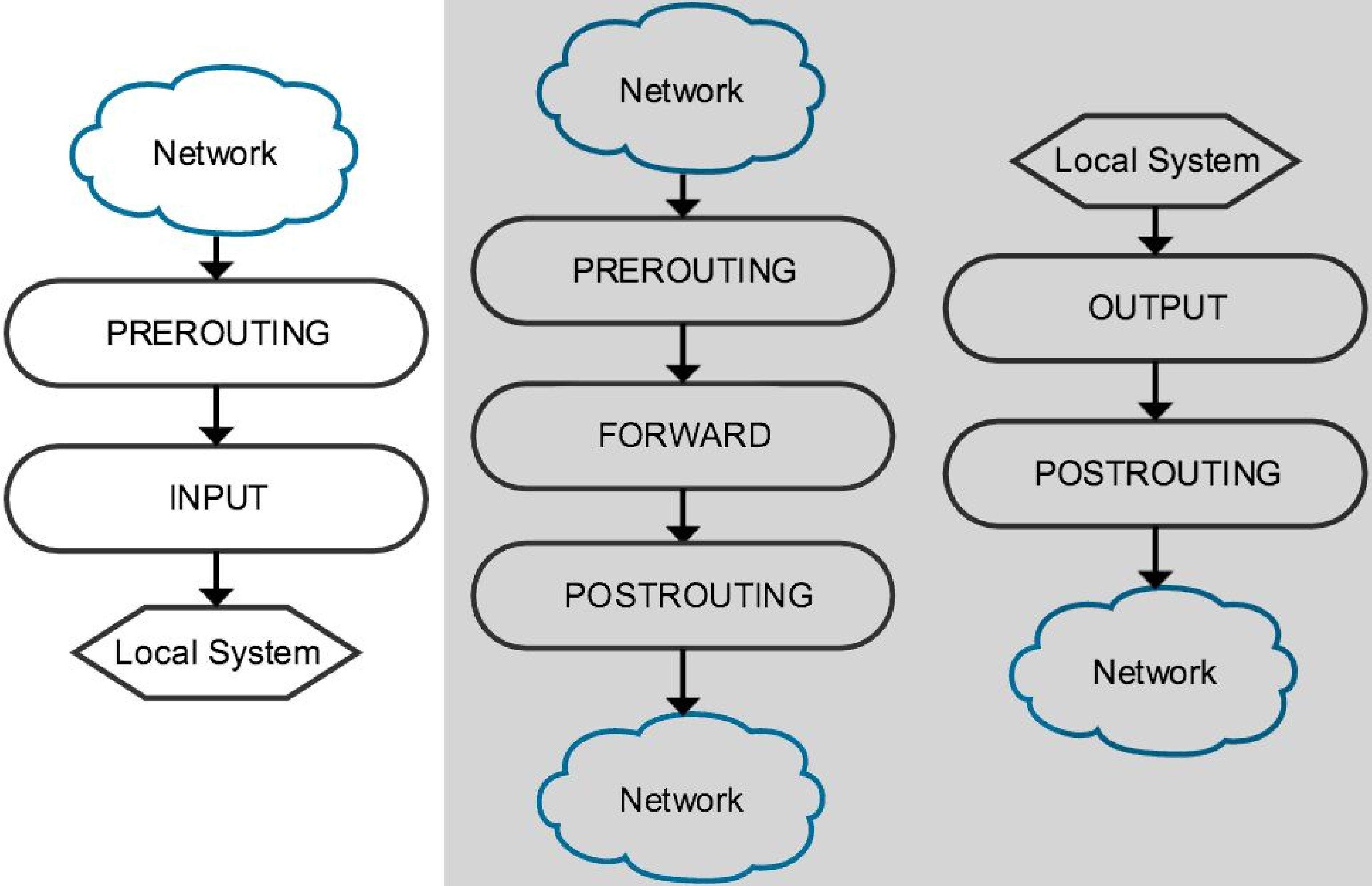
Rules

Chain: LOGNDROP

Rules


Network
PREROUTING
INPUT
Local System
Network
PREROUTING
FORWARD
POSTROUTING
Network
Local System
OUTPUT
POSTROUTING
Network

.com



Network

PREROUTING

INPUT

Local System

Network

PREROUTING

FORWARD

POSTROUTING

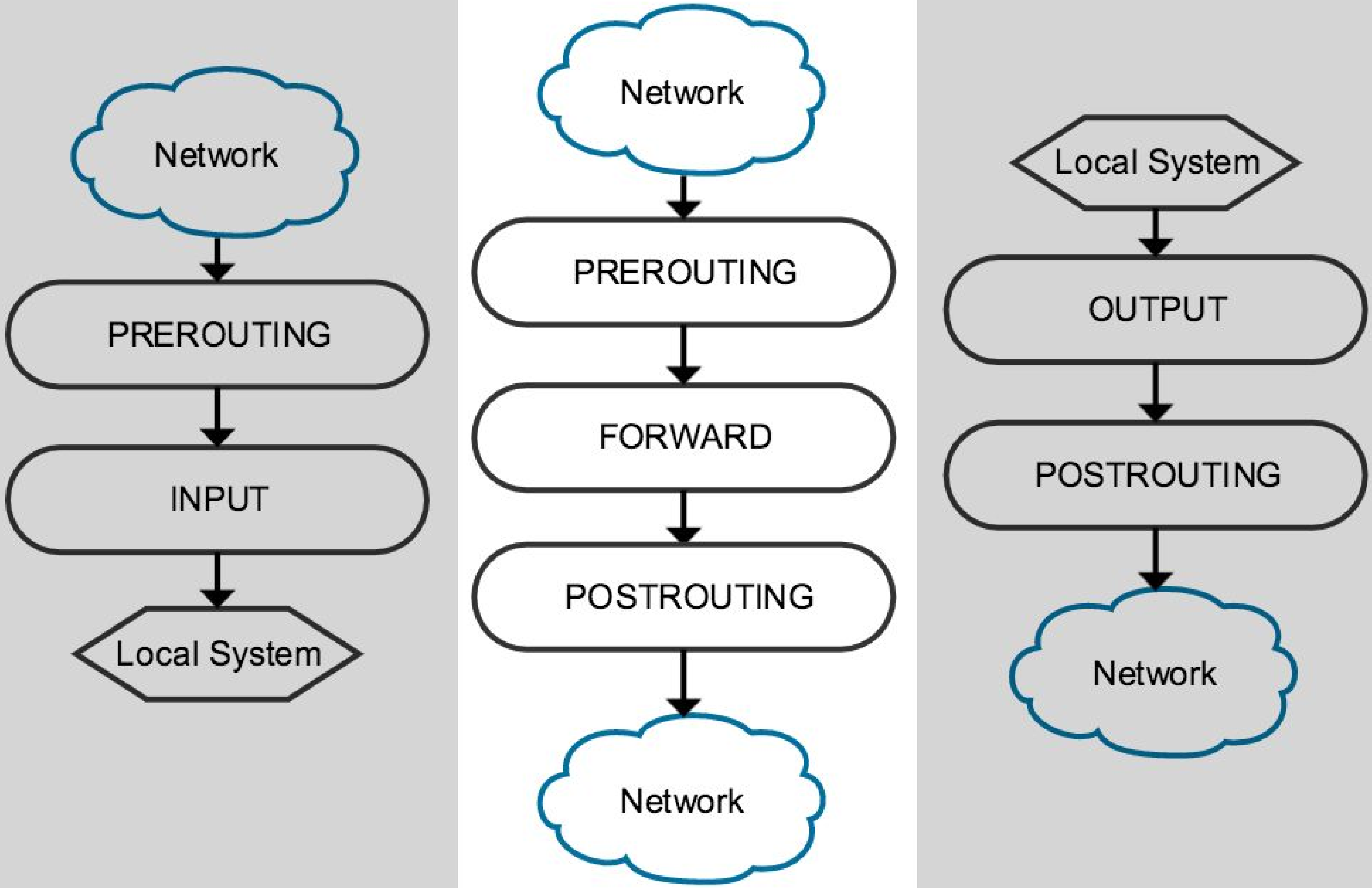
Network

Local System

OUTPUT

POSTROUTING

Network



Network

PREROUTING

INPUT

Local System

Network

PREROUTING

FORWARD

POSTROUTING

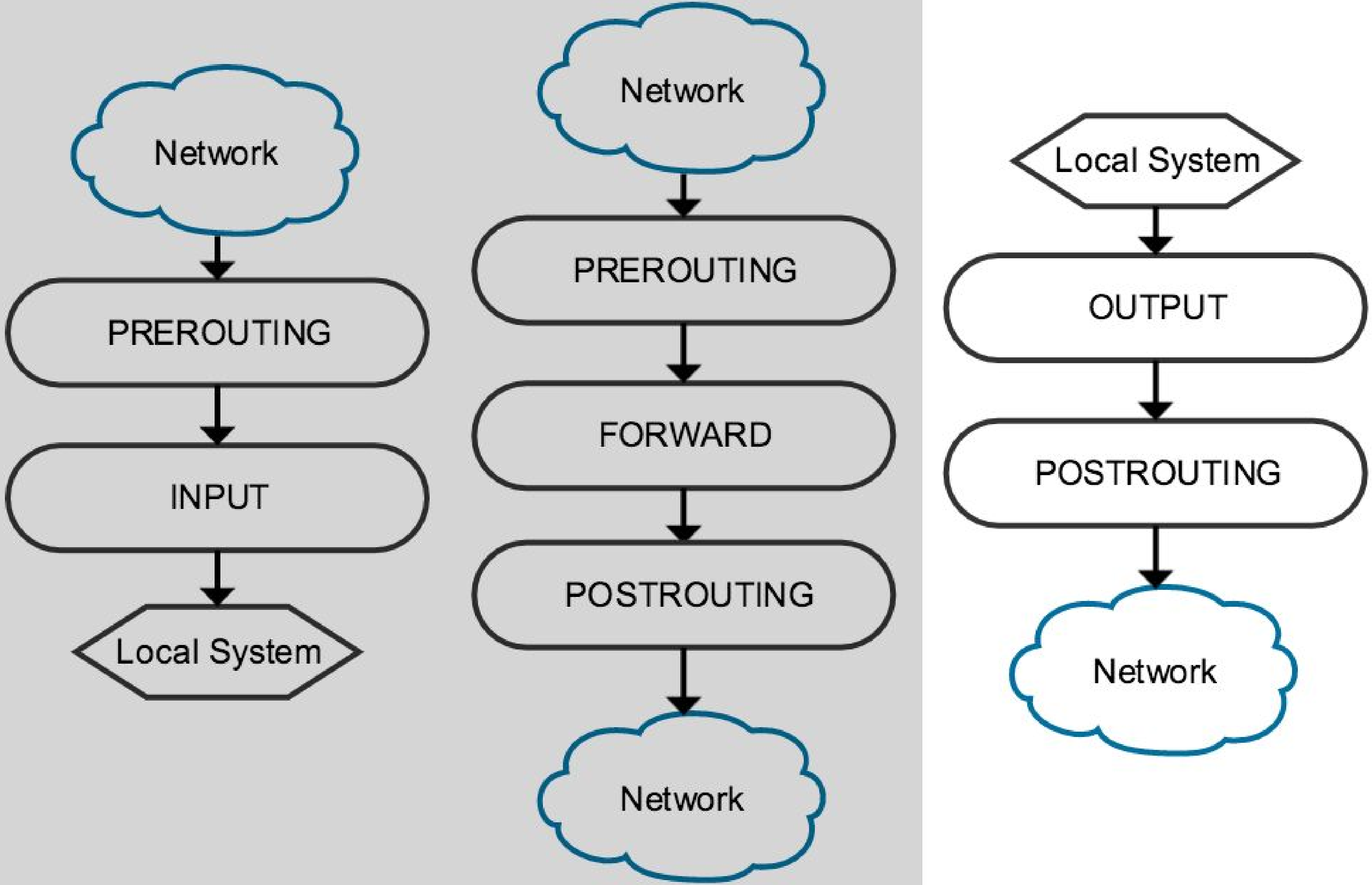
Network

Local System

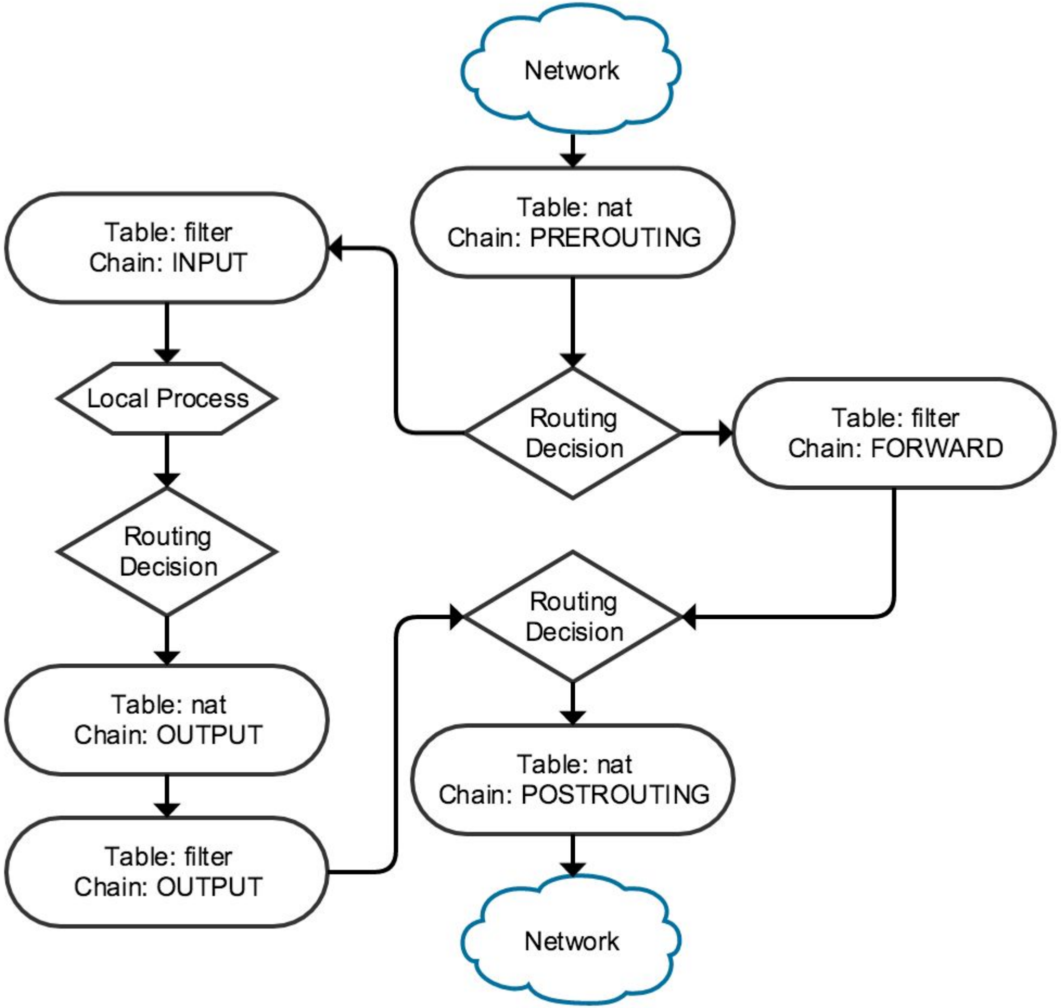
OUTPUT

POSTROUTING

Network


Network
PREROUTING
INPUT
Local System
Network
PREROUTING
FORWARD
POSTROUTING
Network
Local System
OUTPUT
POSTROUTING
Network

.com



Network

Table: nat
Chain: PREROUTING

Routing
Decision

Table: filter
Chain: INPUT

Local Process

Routing
Decision

Table: nat
Chain: OUTPUT

Table: filter
Chain: OUTPUT

Table: filter
Chain: FORWARD

Routing
Decision

Table: nat
Chain: POSTROUTING

Network

Rules

- Rules = Match + Target
- Match on:
  - Protocol
  - Source/Dest IP or network
  - Source/Dest Port
  - Network Interface

Rules

- Rules = Match + Target
- Match on:
  - Protocol
  - Source/Dest IP or network
  - Source/Dest Port
  - Network Interface
  - Example:
    - protocol: TCP, source IP: 1.2.3.4, dest port: 80

Targets

- Chain
- Built-in targets:
    - ACCEPT
    - DROP
    - REJECT
    - LOG
    - RETURN

iptables / ip6tables

- Command line interface to IPTables/netfilter.

List / View

`iptables -L` - Display the filter table.

`iptables -t nat -L` - Display the nat table.

`iptables -nL` - Display using numeric output.

`iptables -vL` - Display using verbose output.

`iptables --line-numbers -L` - Use line nums.

```
# 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
```

```
# 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
```

```
# iptables -nL
Chain INPUT (policy DROP)
target   prot opt source               destination
DROP     all  --  216.58.219.174       0.0.0.0/0
ACCEPT   tcp  --  0.0.0.0/0            0.0.0.0/0    tcp dpt:80
ACCEPT   tcp  --  0.0.0.0/0            0.0.0.0/0    tcp dpt:433
ACCEPT   tcp  --  10.11.12.0/24        0.0.0.0/0    tcp dpt:22
ACCEPT   icmp --  10.11.12.0/24        0.0.0.0/0    icmptype 8

Chain FORWARD (policy ACCEPT)
target   prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target   prot opt source               destination
```

Chain Policy / Default Target

Set the default TARGET for CHAIN:

```
iptables -P CHAIN TARGET
```

Example:

```
iptables -P INPUT DROP
```

```
# iptables -nL
Chain INPUT (policy DROP)
target    prot opt source              destination
DROP      all  --  216.58.219.174      0.0.0.0/0
ACCEPT    tcp  --  0.0.0.0/0           0.0.0.0/0       tcp dpt:80
ACCEPT    tcp  --  0.0.0.0/0           0.0.0.0/0       tcp dpt:433
ACCEPT    tcp  --  10.11.12.0/24       0.0.0.0/0       tcp dpt:22
ACCEPT    icmp --  10.11.12.0/24       0.0.0.0/0       icmptype 8

Chain FORWARD (policy ACCEPT)
target    prot opt source              destination

Chain OUTPUT (policy ACCEPT)
target    prot opt source              destination
```

Appending, Inserting, and Deleting Rules

```
iptables -A CHAIN RULE-SPECIFICATION
iptables [-t TABLE] -A CHAIN RULE-SPECIFICATION

iptables -I CHAIN [RULENUM] RULE-SPECIFICATION

iptables -D CHAIN RULE-SPECIFICATION
iptables -D CHAIN RULENUM
```

Flushing rules

```
iptables [-t table] -F  [chain]
```

Rule Specification Options

Option	**Description**

`-s SOURCE`
`-s 10.11.12.13`
`-s 10.11.12.0/24`
`-s 10.11.12.0/255.255.255.0`
	Source IP, network, or name*.
	Name is resolved when the rule is added.

`-d DESTINATION`
`-d 192.168.4.11`
`-d 216.58.192.0/19`
`-d 216.58.192.0/255.255.224.0`
	Destination IP, network, or name*.

`-p PROTOCOL`
`-p tcp`
`-p udp`
`-p icmp`
	Protocol.

Rule Specification Options

Option	Description

`-s SOURCE`
`-s 10.11.12.13`
`-s 10.11.12.0/24`
`-s 10.11.12.0/255.255.255.0`
	Source IP, network, or name*.
	Name is resolved when the rule is added.

`-d DESTINATION`
`-d 192.168.4.11`
`-d 216.58.192.0/19`
`-d 216.58.192.0/255.255.224.0`
	Destination IP, network, or name*.

`-p PROTOCOL`
`-p tcp`
`-p udp`
`-p icmp`
	Protocol.

Rule Specification Options

Option	Description

`-s SOURCE`
`-s 10.11.12.13`
`-s 10.11.12.0/24`
`-s 10.11.12.0/255.255.255.0`
	Source IP, network, or name*.
	Name is resolved when the rule is added.

`-d DESTINATION`
`-d 192.168.4.11`
`-d 216.58.192.0/19`
`-d 216.58.192.0/255.255.224.0`
	Destination IP, network, or name*.

`-p PROTOCOL`
`-p tcp`
`-p udp`
`-p icmp`
	Protocol.

Rule Specification Options

Option	Description
`-m MODULE MODULE_OPTIONS`	Enable extended packet matching module. (`man iptables-extensions`)

`-p PROTOCOL -m PROTOCOL --dport PORT`
`-p tcp -m tcp --dport 80`
`-p tcp --dport 80`
`-p udp --dport 53`
	Destination port

`-p PROTOCOL -m PROTOCOL --sport PORT`
`-p tcp -m tcp --sport 8080`
`-p tcp --sport 8080`
	Source port

`-p icmp -m icmp --icmp-type TYPE`
`-p icmp -m icmp --icmp-type echo-reply`
`-p icmp --icmp-type echo-reply`
`-p icmp --icmp-type echo-request`
	ICMP packet type
	(`iptables -p icmp -h`)

Rule Specification Options

Option	Description
`-m MODULE MODULE_OPTIONS`	Enable extended packet matching module. (`man iptables-extensions`)

`-p PROTOCOL -m PROTOCOL --dport PORT`
`-p tcp -m tcp --dport 80`
`-p tcp --dport 80`
`-p udp --dport 53`
	Destination port

`-p PROTOCOL -m PROTOCOL --sport PORT`
`-p tcp -m tcp --sport 8080`
`-p tcp --sport 8080`
	Source port

`-p icmp -m icmp --icmp-type TYPE`
`-p icmp -m icmp --icmp-type echo-reply`
`-p icmp --icmp-type echo-reply`
`-p icmp --icmp-type echo-request`
	ICMP packet type
	(`iptables -p icmp -h`)

Rule Specification Options

Option	Description
`-m MODULE MODULE_OPTIONS`	Enable extended packet matching module. (`man iptables-extensions`)

`-p PROTOCOL -m PROTOCOL --dport PORT`
`-p tcp -m tcp --dport 80`
`-p tcp --dport 80`
`-p udp --dport 53`
	Destination port

`-p PROTOCOL -m PROTOCOL --sport PORT`
`-p tcp -m tcp --sport 8080`
`-p tcp --sport 8080`
	Source port

`-p icmp -m icmp --icmp-type TYPE`
`-p icmp -m icmp --icmp-type echo-reply`
`-p icmp --icmp-type echo-reply`
`-p icmp --icmp-type echo-request`
	ICMP packet type
	(`iptables -p icmp -h`)

Rule Specification Options

Option	Description
`-m MODULE MODULE_OPTIONS`	Enable extended packet matching module. (`man iptables-extensions`)

`-p PROTOCOL -m PROTOCOL --dport PORT`
`-p tcp -m tcp --dport 80`
`-p tcp --dport 80`
`-p udp --dport 53`
	Destination port

`-p PROTOCOL -m PROTOCOL --sport PORT`
`-p tcp -m tcp --sport 8080`
`-p tcp --sport 8080`
	Source port

`-p icmp -m icmp --icmp-type TYPE`
`-p icmp -m icmp --icmp-type echo-reply`
`-p icmp --icmp-type echo-reply`
`-p icmp --icmp-type echo-request`
	ICMP packet type (`iptables -p icmp -h`)

Rule Specification Options

Option	Description

`-m limit --limit rate[/second/minute/hour/day]`
`-m limit --limit-burst`
 	
`-m limit --limit 5/m --limit-burst 10`
`-m limit ! --limit 5/s`
	Match until a limit is reached.
	--limit default is 3/hour
	--limit-burst default is 5
	/s = second
	/m = minute
	/h = hour
	/d = day
	! invert the match

Target / Jump

To specify a jump point or target:

```
-j TARGET_OR_CHAIN
```

```
-j ACCEPT
-j DROP
-j LOGNDROP
```

Built-in target.
Built-in target.
Custom chain.

Rule Specification Example

```
iptables -A INPUT -s 216.58.219.174 -j DROP
```

```
# iptables -nL
Chain INPUT (policy ACCEPT)
target  prot opt source          destination
DROP    all  --  216.58.219.174 0.0.0.0/0

Chain FORWARD (policy ACCEPT)
target     prot opt source      destination
```

Rule Specification Example

```
iptables -A INPUT -s 10.0.0.0/24 \
-p tcp --dport 22 -j ACCEPT
iptables -A INPUT -p tcp --dport 22 -j DROP
```

```
# iptables -nL
Chain INPUT (policy ACCEPT)
target  prot opt source          destination
ACCEPT  tcp  --  10.0.0.0/24     0.0.0.0/0 tcp dpt:22
DROP    tcp  --  0.0.0.0/0       0.0.0.0/0 tcp dpt:22
```

Rule Specification Example

```
iptables -I INPUT -p tcp --dport 80 \
-m limit --limit 50/min --limit-burst 200 \
-j REJECT
```

```
iptables -I INPUT -p tcp --dport 80 \
-m limit --limit 50/min --limit-burst 200 \
-m state --state NEW -j REJECT
```

Creating and Deleting a Chain

Create CHAIN:

```
iptables [-t table] -N CHAIN
```

Delete CHAIN:

```
iptables [-t table] -X CHAIN
```

Saving Rules

Debian / Ubuntu:

```
apt-get install iptables-persistent
netfilter-persistent save
```

CentOS / RedHat:

```
yum install iptables-services
 service iptables save
```

Netfilter/iptable Front-Ends

- Uses iptables command on the back-end
- Firewalld - CentOS/RHEL
- UFW - Uncomplicated FireWall (Ubuntu)
- GUFW - Graphical interface to UFW
- system-configure-firewall - CentOS/RHEL

Linux Firewall Demonstration

TCP Wrappers

TCP Wrappers

- Host-based networking ACL system.
- Controls access to “wrapped” services.
- A wrapped service is compiled with libwrap support.

Wrapped Services

`ldd` - Prints required shared libraries.

Wrapped Services

`ldd` - Prints required shared libraries.

Example:

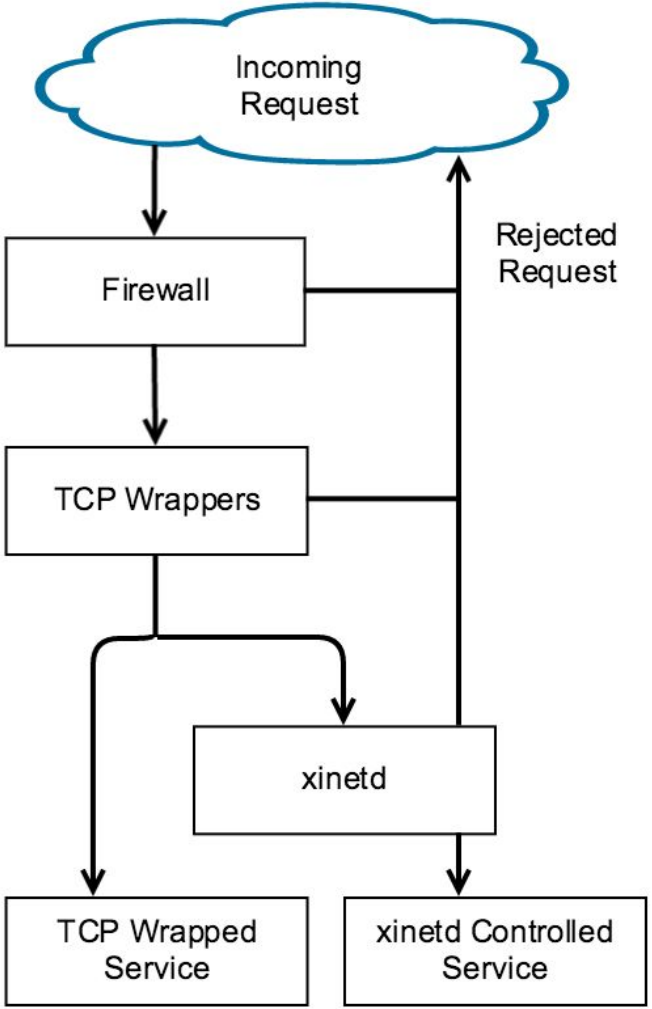
```
# ldd /usr/sbin/sshd | grep libwrap
libwrap.so.0 => /lib64/libwrap.so.0
(0x00007f10219f8000)
```

TCP Wrappers

- Can control access by IP address / networks.
- Can control access by hostname.
- Transparent to the client and service.

TCP Wrappers

- Used with xinetd.
- Centralized management for multiple network services.
- Runtime configuration


Incoming Request
Firewall
TCP Wrappers
xinetd
TCP Wrapped Service
xinetd Controlled Service
Rejected Request

Configuring TCP Wrappers

- `/etc/hosts.allow /etc/hosts.deny`
- `/etc/hosts.allow` is checked first.
- If a match is found, access is granted.
- `/etc/hosts.deny` is checked next.
- If a match is found, access is denied.
  - `refused connect from webapp2 (1.2.3.4)`
- If there are no matches, access is granted.

Access Rules

- The rule format for `hosts.allow` and `hosts.deny` are the same.
- One rule per line
- Format:

```
SERVICE(S) : CLIENT(S) [: ACTION(S) ]
```

TCP Wrapper Examples

```
# SERVICE(S) : CLIENT(S) [: ACTION(S) ]
sshd : 10.11.12.13
```

TCP Wrapper Examples

```
# SERVICE(S) : CLIENT(S) [: ACTION(S) ]
sshd : 10.11.12.13

sshd, imapd : 10.11.12.13
```

TCP Wrapper Examples

```
# SERVICE(S) : CLIENT(S) [: ACTION(S) ]
sshd : 10.11.12.13

sshd, imapd : 10.11.12.13

ALL : 10.11.12.13
```

TCP Wrapper Examples

```
# SERVICE(S) : CLIENT(S) [: ACTION(S) ]
sshd : 10.11.12.13
```

TCP Wrapper Examples

```
# SERVICE(S) : CLIENT(S) [: ACTION(S) ]
sshd : 10.11.12.13


sshd : 10.11.12.13, 10.5.6.7
```

TCP Wrapper Examples

```
# SERVICE(S) : CLIENT(S) [: ACTION(S) ]
sshd : 10.11.12.13


sshd : 10.11.12.13, 10.5.6.7


sshd : jumpbox.example.com
```

TCP Wrapper Examples

```
# SERVICE(S) : CLIENT(S) [: ACTION(S) ]
sshd : .example.com
```

TCP Wrapper Examples

```
# SERVICE(S) : CLIENT(S) [: ACTION(S) ]
sshd : .example.com

sshd : .admin.example.com
  # server2.admin.example.com
  # webdev.admin.example.com
```

TCP Wrapper Examples

```
# SERVICE(S) : CLIENT(S) [: ACTION(S) ]
sshd : jumpbox*.example.com
  # jumpbox4admins.example.com
```

TCP Wrapper Examples

```
# SERVICE(S) : CLIENT(S) [: ACTION(S) ]
sshd : jumpbox*.example.com
  # jumpbox4admins.example.com

sshd : jumpbox0?.example.com
  # jumpbox03.example.com
```

TCP Wrapper Examples

```
# SERVICE(S) : CLIENT(S) [: ACTION(S) ]
sshd : 10.11.12.

sshd : 10.
```

TCP Wrapper Examples

```
# SERVICE(S) : CLIENT(S) [: ACTION(S) ]
sshd : 10.11.12.


sshd : 10.


sshd : 10.11.0.0/255.255.0.0
```

TCP Wrapper Examples

```
# SERVICE(S) : CLIENT(S) [: ACTION(S) ]
sshd : 10.11.12.

sshd : 10.

sshd : 10.11.0.0/255.255.0.0

sshd : /etc/hosts.sshd
```

TCP Wrapper Examples

```
# SERVICE(S) : CLIENT(S) [: ACTION(S) ]
imapd : ALL
```

TCP Wrapper Examples

```
# /etc/hosts.allow
sshd : ALL EXCEPT .hackers.net
```

TCP Wrappers Logging

```
# SERVICE(S) : CLIENT(S) [: ACTION(S) ]
sshd : 10.11.12.13 : severity emerg
```

TCP Wrappers Logging

```
# SERVICE(S) : CLIENT(S) [: ACTION(S) ]
sshd : 10.11.12.13 : severity emerg

sshd : 10.11.12.13 : severity local0.alert
```

TCP Wrappers Logging

```
# /etc/hosts.deny:
sshd : .hackers.net \
  : spawn /usr/bin/wall “Attack in progress.”
```

TCP Wrappers Logging

```
# /etc/hosts.deny:
sshd : .hackers.net \
  : spawn /usr/bin/wall “Attack from %a.”
```

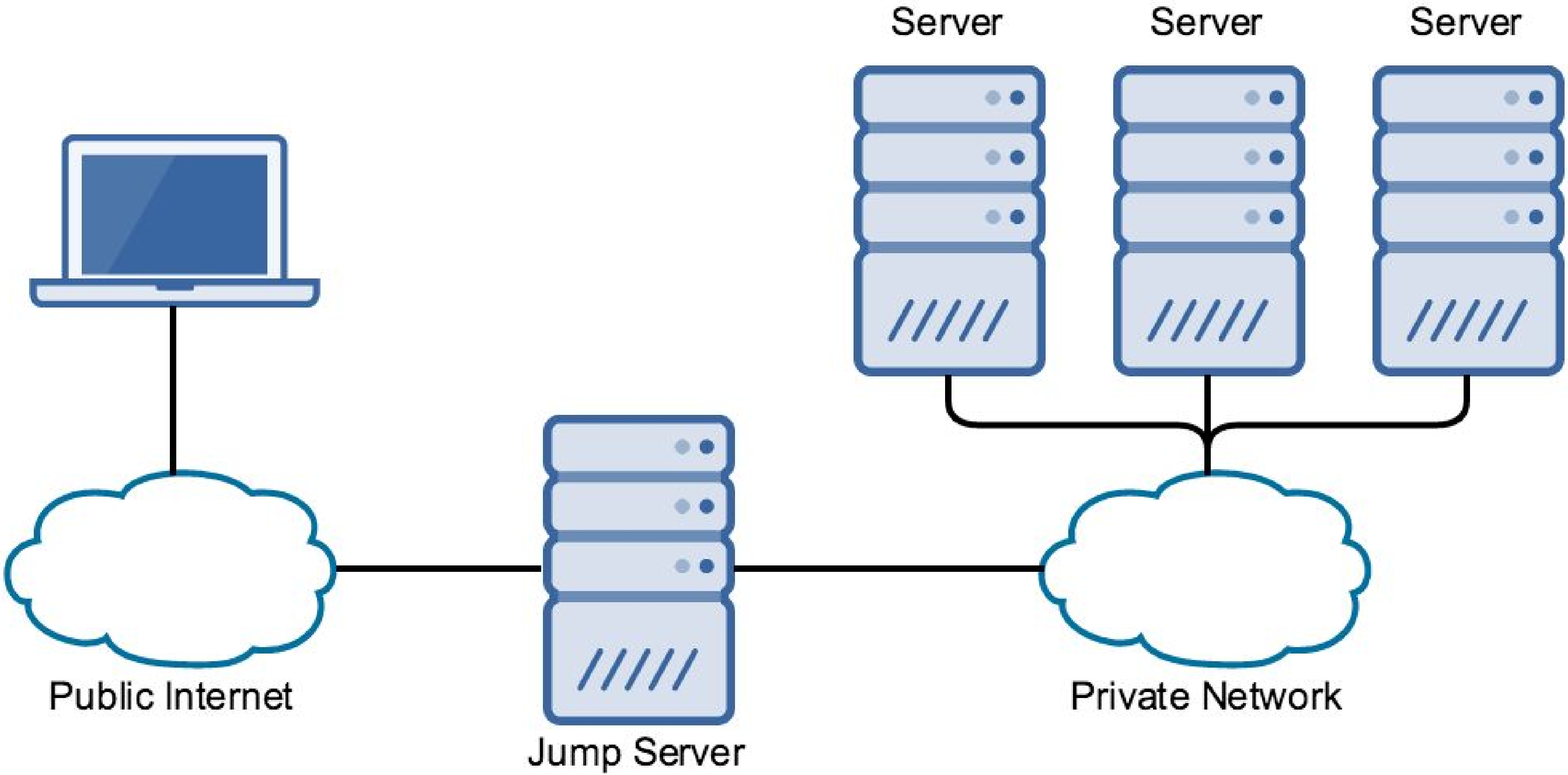
Expansions

```
%a (%A) The client (server) host address.
%c      Client information.
%d      The daemon process name.
%h (%H) The client (server) host name or address.
%n (%N) The client (server) host name.
%p      The daemon process id.
%s      Server information.
%u      The client user name (or "unknown").
%%      Expands to a single `%´ character.
```

Deny All

```
# /etc/hosts.deny:
ALL : ALL


# /etc/hosts.allow:
# Explicitly list allowed connections here.
sshd : 10.11.12.13
```


Server
Server
Server
Public Internet
Jump Server
Private Network

Section Summary

What You Will Learn

- Securing network services.
- Configuring local Linux firewalls.
- Preventing information leakage.
- Port scanning.
- Xinetd security.
- Securing SSH.

Summary

- Securing network services.
- Configuring local Linux firewalls.
- Preventing information leakage.
- Port scanning.
- Xinetd security.
- Securing SSH.