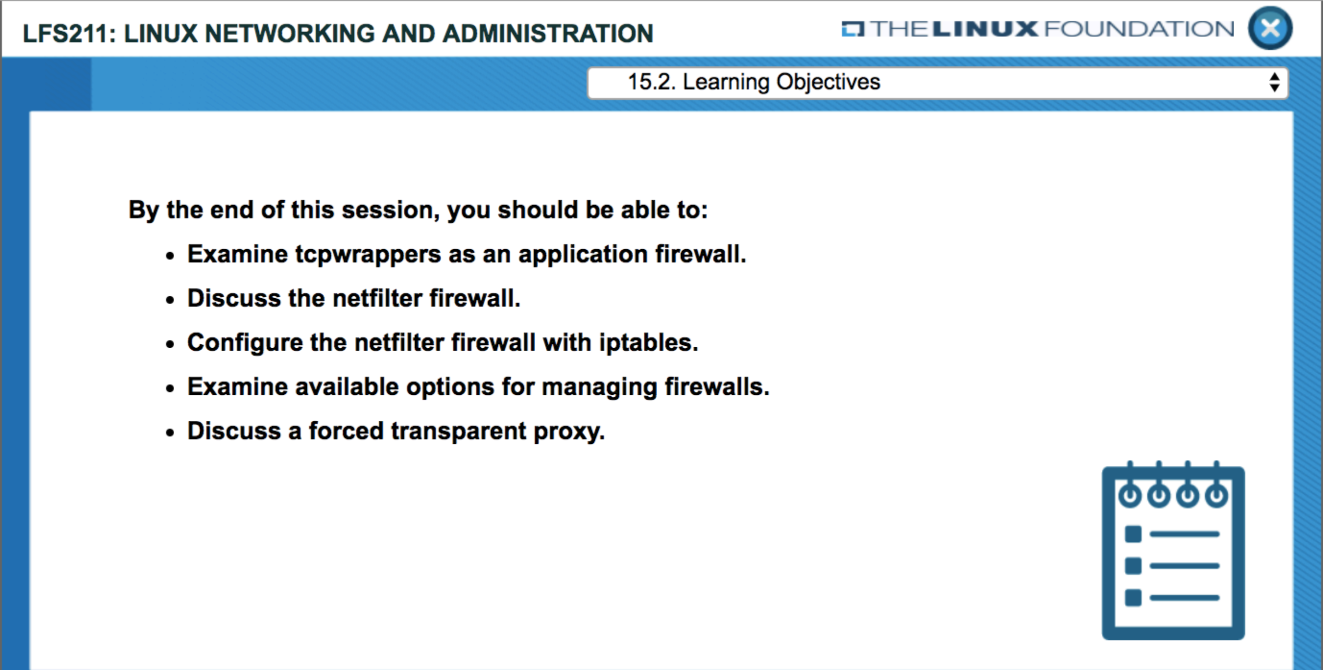
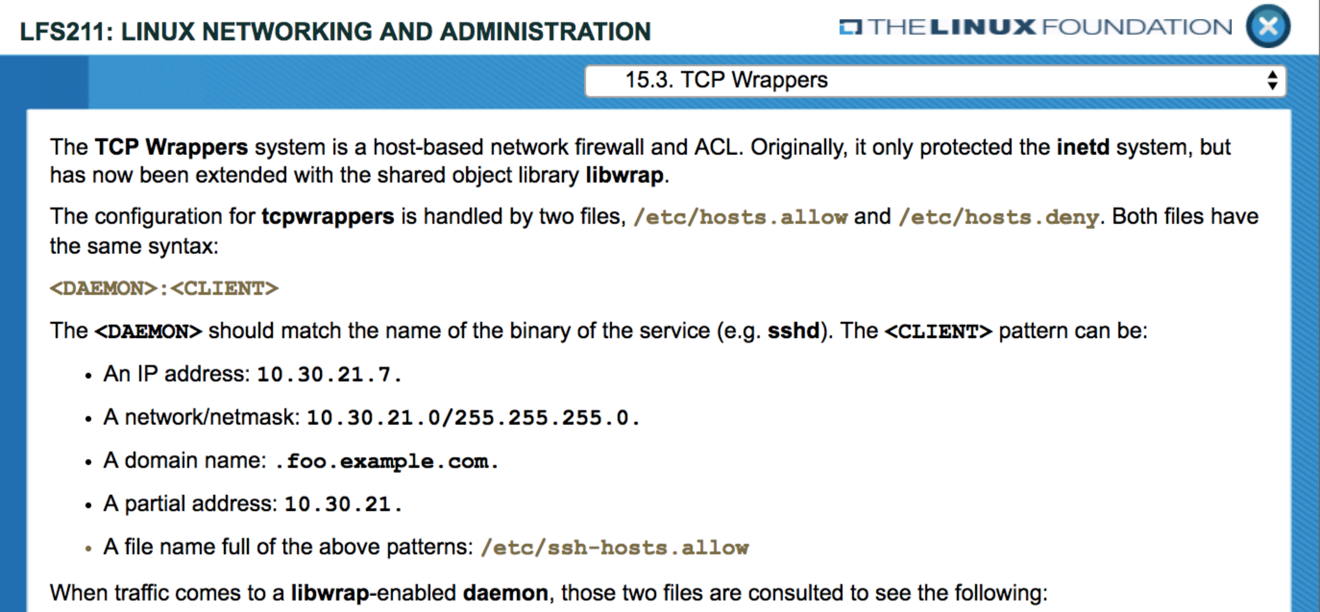


15.2. Learning Objectives

By the end of this session, you should be able to:

- **Examine tcpwrappers as an application firewall.**
- **Discuss the netfilter firewall.**
- **Configure the netfilter firewall with iptables.**
- **Examine available options for managing firewalls.**
- **Discuss a forced transparent proxy.**



15.3. TCP Wrappers

The **TCP Wrappers** system is a host-based network firewall and ACL. Originally, it only protected the **inetd** system, but has now been extended with the shared object library **libwrap**.

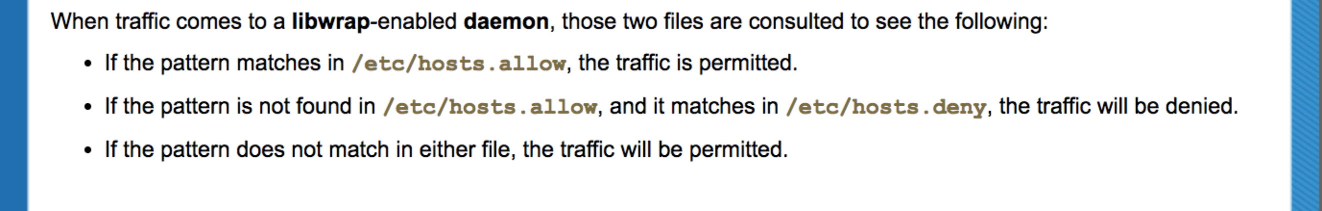
The configuration for **tcpwrappers** is handled by two files, `/etc/hosts.allow` and `/etc/hosts.deny`. Both files have the same syntax:

```
<DAEMON>:<CLIENT>
```

The **`<DAEMON>`** should match the name of the binary of the service (e.g. **sshd**). The **`<CLIENT>`** pattern can be:

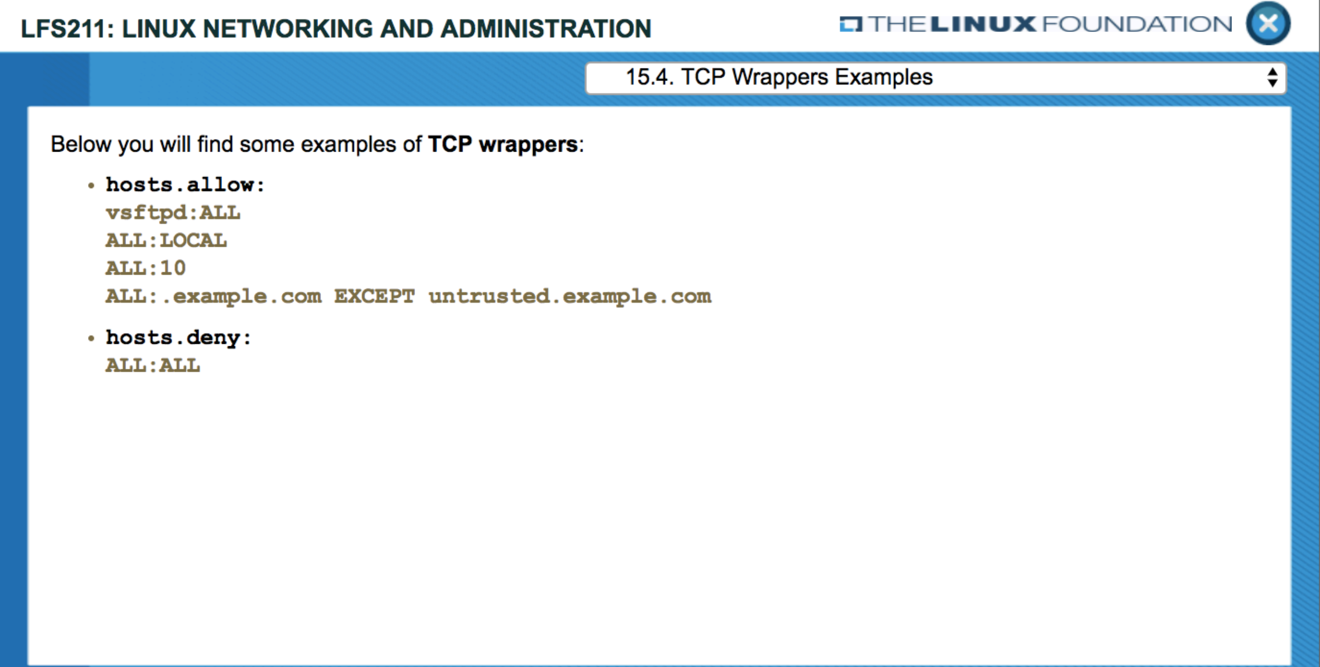
- An IP address: **`10.30.21.7.`**
- A network/netmask: **`10.30.21.0/255.255.255.0.`**
- A domain name: **`.foo.example.com.`**
- A partial address: **`10.30.21.`**
- A file name full of the above patterns: `/etc/ssh-hosts.allow`

When traffic comes to a **libwrap**-enabled **daemon**, those two files are consulted to see the following:



When traffic comes to a **libwrap**-enabled **daemon**, those two files are consulted to see the following:

- If the pattern matches in `/etc/hosts.allow`, the traffic is permitted.
- If the pattern is not found in `/etc/hosts.allow`, and it matches in `/etc/hosts.deny`, the traffic will be denied.
- If the pattern does not match in either file, the traffic will be permitted.



15.4. TCP Wrappers Examples

Below you will find some examples of **TCP wrappers**:

- **hosts.allow:**

```
vsftpd:ALL
ALL:LOCAL
ALL:10
ALL:.example.com EXCEPT untrusted.example.com
```

- **hosts.deny:**

```
ALL:ALL
```

15.5. netfilter Vocabulary

netfilter is a packet-filtering framework built into the **Linux** kernel. To better understand **netfilter**, we need to start with some vocabulary:

- The **netfilter** firewall consists of **Tables**.
- **Tables** consist of **Chains**.
- **Chains** have a default **Policy**.
- **Chains** consist of **Rules**.
- **Rules** consist of a **Match criteria** and a **Target**.

Rules in each chain are read first to last, and the first match wins. If a packet does not match any rule in a chain, the policy of the chain applies.

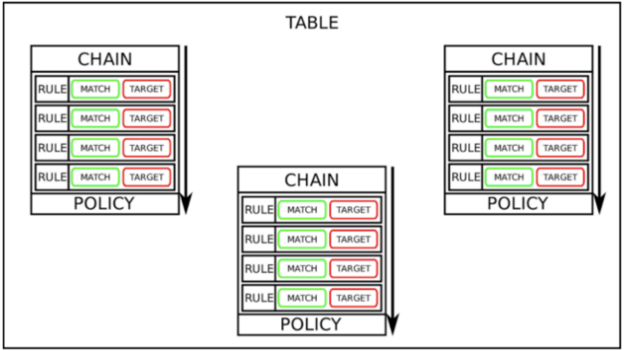


Figure 15.1: netfilter Vocabulary

15.6. Filter Table

The **filter** table deals with packets bound for the local machine, being routed through the machine, or packets generated by processes on the machine. It contains the default **chains**:

- **`INPUT`**
  For packets bound for local processes.
- **`FORWARD`**
  For packets being forwarded through the machine.
- **`OUTPUT`**
  For packets generated by local processes that and are now outbound to the network.

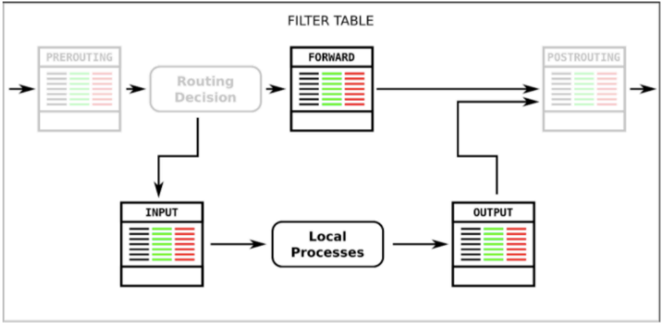


Figure 15.2: Filter Table

15.7. NAT Table

The **Network Address Translation (NAT)** table is used when traffic that creates a new network connection is encountered. It contains the default **chains**:

- **`PREROUTING`**
  For altering packets just as they come in.
- **`OUTPUT`**
  For altering packets generated by local processes, prior to routing.
- **`POSTROUTING`**
  For packets just about to go out to the network.

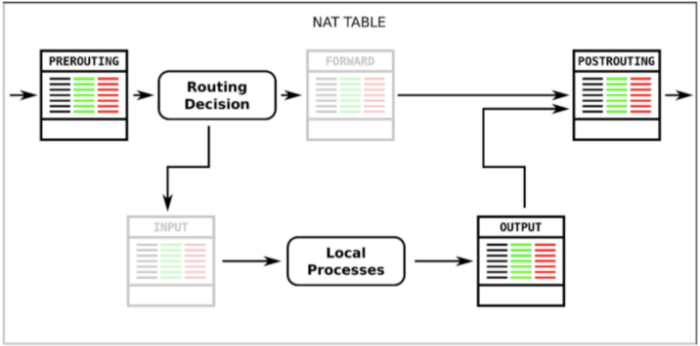


Figure 15.3: NAT Table

15.8. Mangle Table

The **mangle** table is for specialized manipulation of network packets. It contains the following default **chains**:

- **`INPUT`**
  For packets bound for local processes.
- **`PREROUTING`**
  For altering packets just as they come in.
- **`FORWARD`**
  For packets being forwarded through the machine.
- **`OUTPUT`**
  For packets generated by local processes that are now outbound to the network.

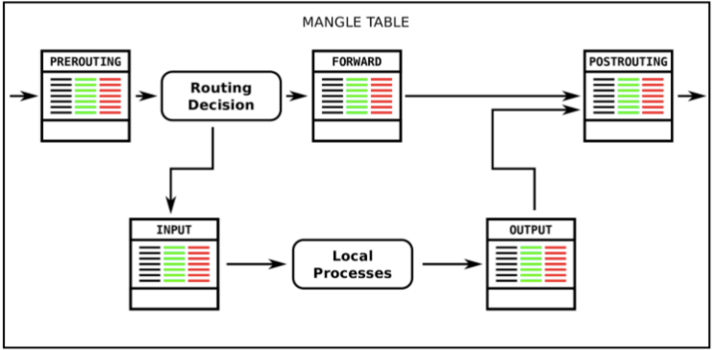


Figure 15.4: Mangle Table

- **`OUTPUT`**
  For packets generated by local processes that are now outbound to the network.
- **`POSTROUTING`**
  For packets just about to go out to the network.

15.8. Mangle Table

The **mangle** table is for specialized manipulation of network packets. It contains the following default **chains**:

- **`INPUT`**
  For packets bound for local processes.
- **`PREROUTING`**
  For altering packets just as they come in.
- **`FORWARD`**
  For packets being forwarded through the machine.
- **`OUTPUT`**
  For packets generated by local processes that are now outbound to the network.

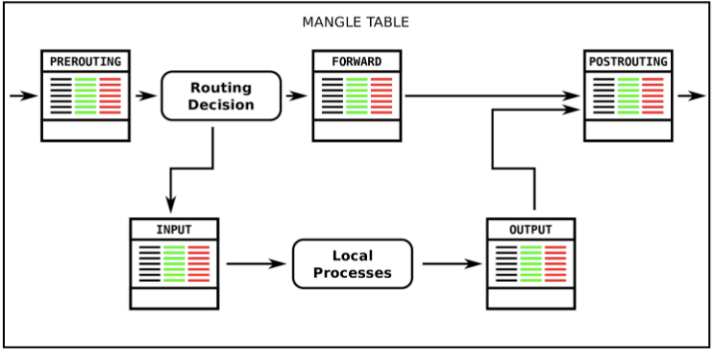
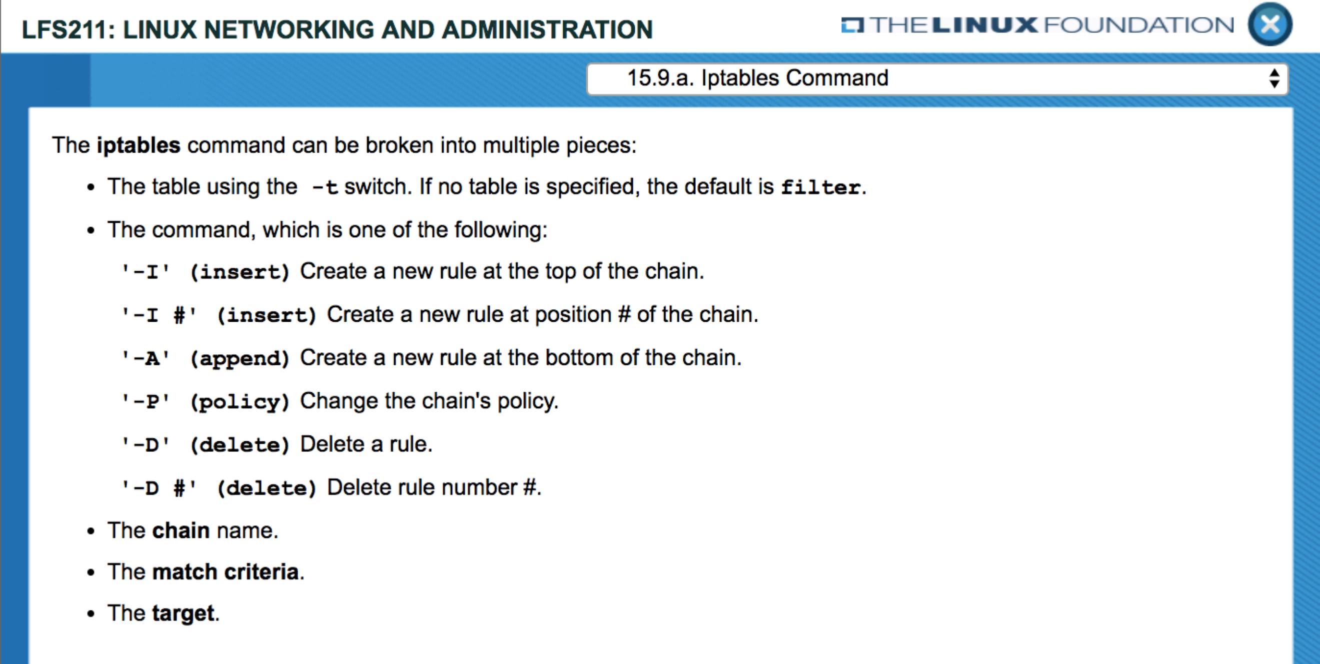


Figure 15.4: Mangle Table

- **`POSTROUTING`**
  For packets just about to go out to the network.

Figure 15.4: Mangle Table



15.9.a. Iptables Command

The **iptables** command can be broken into multiple pieces:

- The table using the `-t` switch. If no table is specified, the default is **`filter`**.
- The command, which is one of the following:

  `'-I' (insert)` Create a new rule at the top of the chain.

  `'-I #' (insert)` Create a new rule at position # of the chain.

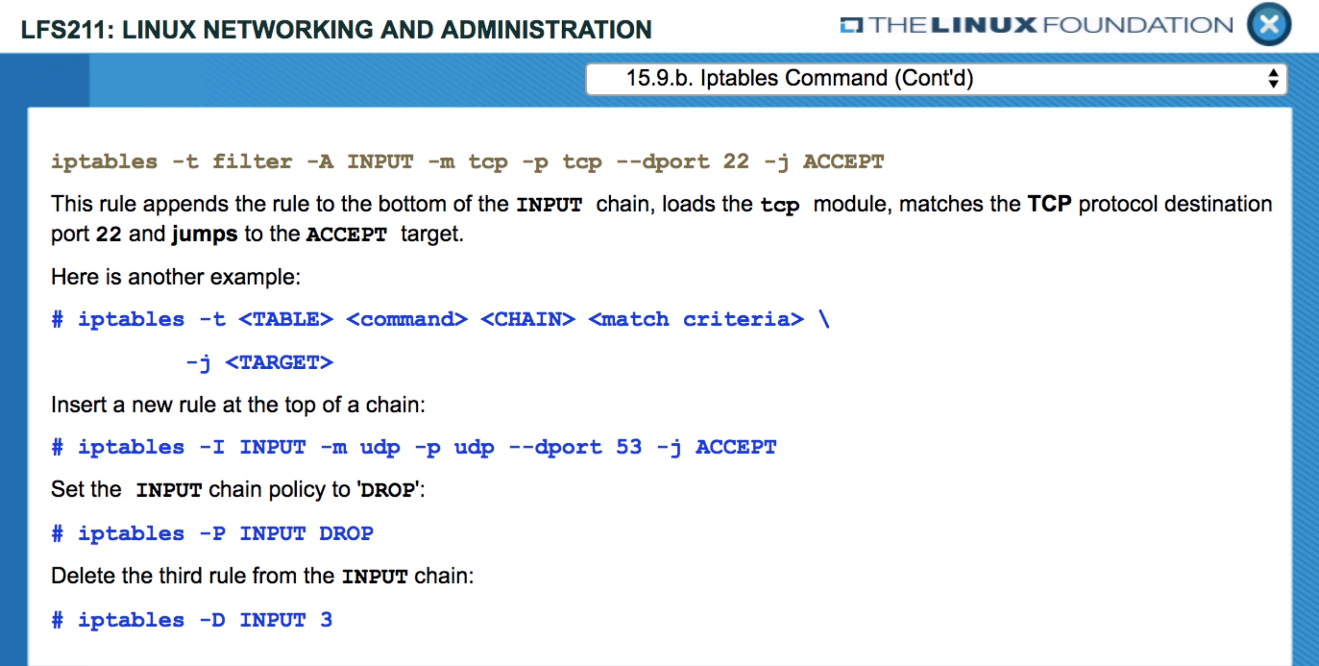
  `'-A' (append)` Create a new rule at the bottom of the chain.

  `'-P' (policy)` Change the chain's policy.

  `'-D' (delete)` Delete a rule.

  `'-D #' (delete)` Delete rule number #.

- The **chain** name.
- The **match criteria**.
- The **target**.



15.9.b. Iptables Command (Cont'd)

```
iptables -t filter -A INPUT -m tcp -p tcp --dport 22 -j ACCEPT
```

This rule appends the rule to the bottom of the **INPUT** chain, loads the **tcp** module, matches the **TCP** protocol destination port **22** and **jumps** to the **ACCEPT** target.

Here is another example:

```
# iptables -t <TABLE> <command> <CHAIN> <match criteria> \
          -j <TARGET>
```

Insert a new rule at the top of a chain:

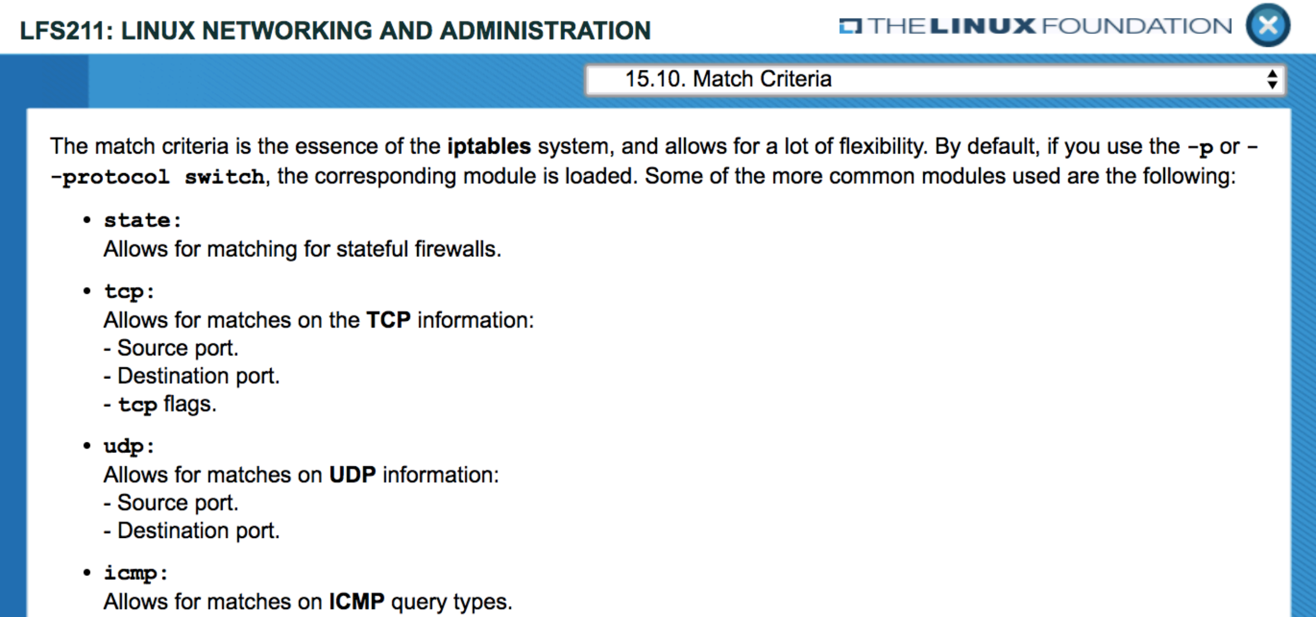
```
# iptables -I INPUT -m udp -p udp --dport 53 -j ACCEPT
```

Set the **INPUT** chain policy to '**DROP**':

```
# iptables -P INPUT DROP
```

Delete the third rule from the **INPUT** chain:

```
# iptables -D INPUT 3
```



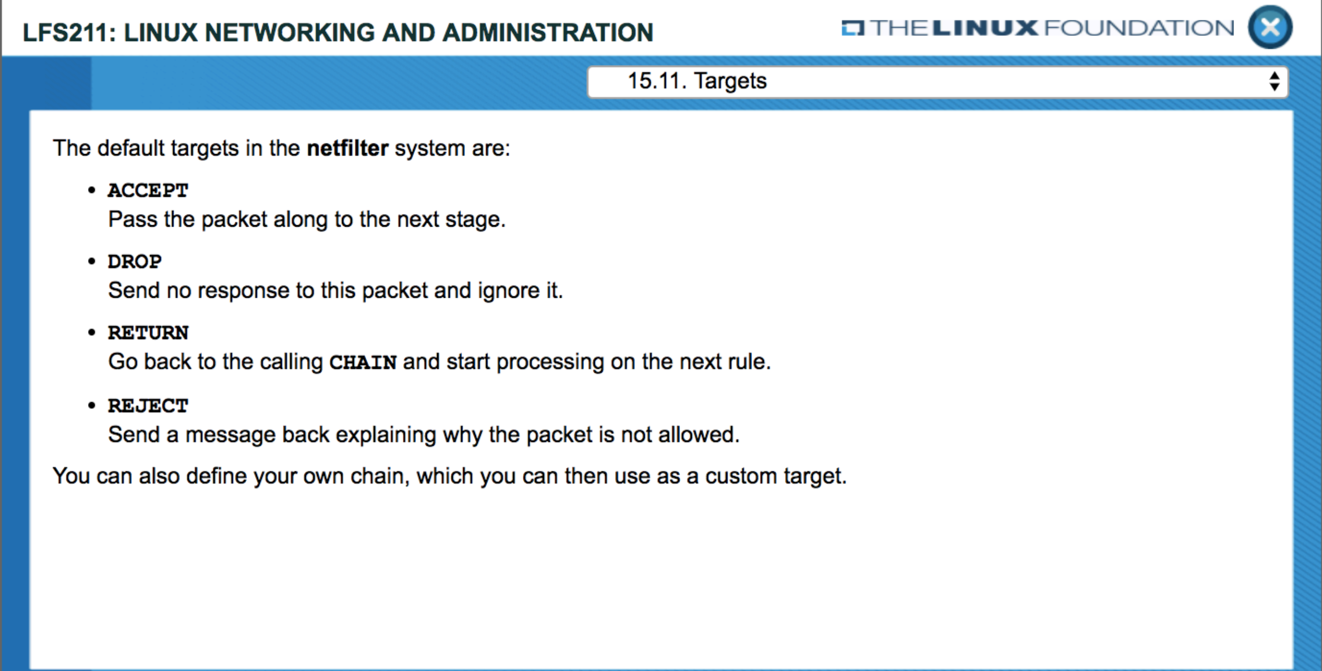
15.10. Match Criteria

The match criteria is the essence of the **iptables** system, and allows for a lot of flexibility. By default, if you use the `-p` or `--protocol switch`, the corresponding module is loaded. Some of the more common modules used are the following:

- `state:`
  Allows for matching for stateful firewalls.
- `tcp:`
  Allows for matches on the **TCP** information:
  - Source port.
  - Destination port.
  - `tcp` flags.
- `udp:`
  Allows for matches on **UDP** information:
  - Source port.
  - Destination port.
- `icmp:`
  Allows for matches on **ICMP** query types.

- **`icmp`:**
  Allows for matches on **ICMP** query types.

A full listing can be found in the **iptables man** page, under the heading **`MATCH EXTENSIONS`**.

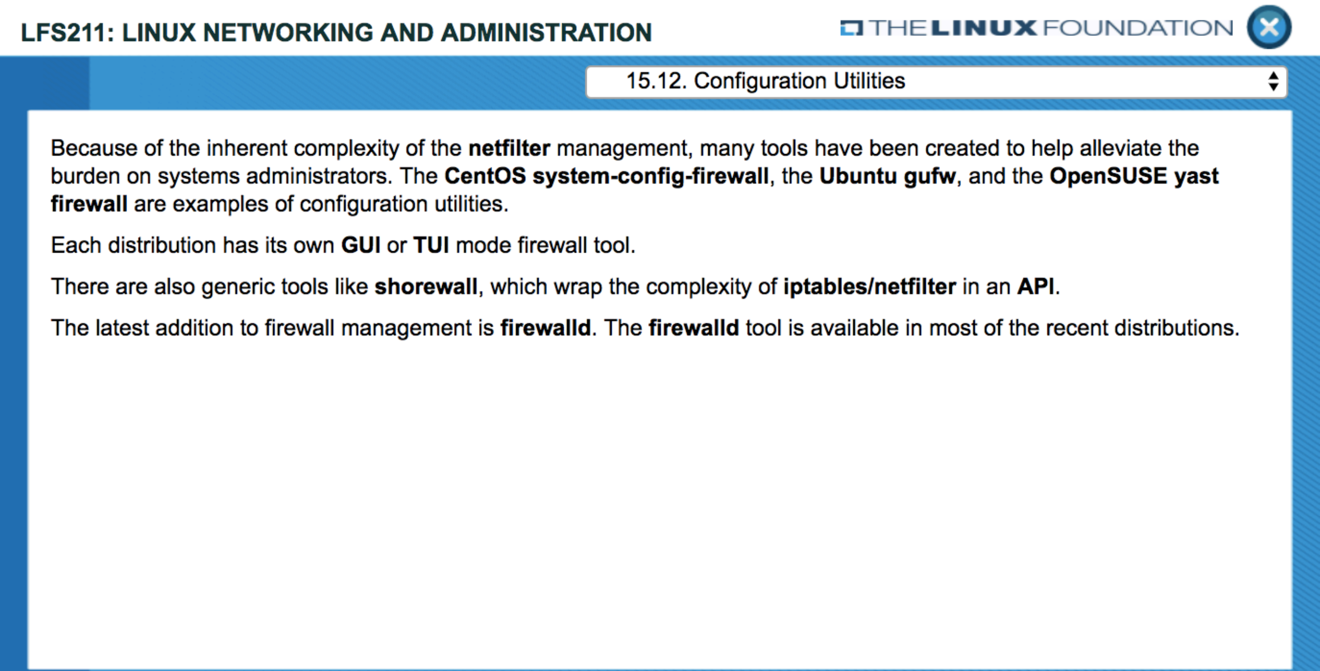


15.11. Targets

The default targets in the **netfilter** system are:

- **ACCEPT**
  Pass the packet along to the next stage.
- **DROP**
  Send no response to this packet and ignore it.
- **RETURN**
  Go back to the calling **CHAIN** and start processing on the next rule.
- **REJECT**
  Send a message back explaining why the packet is not allowed.

You can also define your own chain, which you can then use as a custom target.



15.12. Configuration Utilities

Because of the inherent complexity of the **netfilter** management, many tools have been created to help alleviate the burden on systems administrators. The **CentOS system-config-firewall**, the **Ubuntu gufw**, and the **OpenSUSE yast firewall** are examples of configuration utilities.

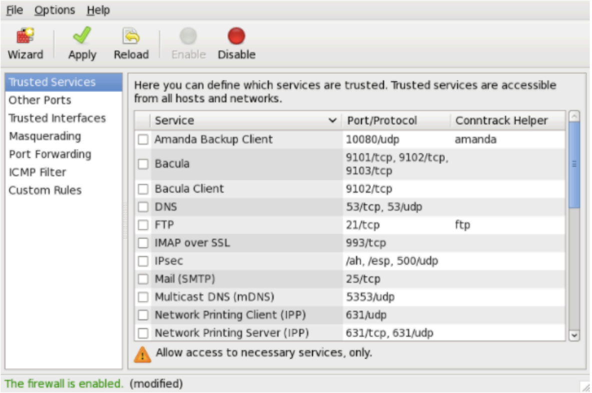
Each distribution has its own **GUI** or **TUI** mode firewall tool.

There are also generic tools like **shorewall**, which wrap the complexity of **iptables/netfilter** in an **API**.

The latest addition to firewall management is **firewalld**. The **firewalld** tool is available in most of the recent distributions.

15.13. System-Config-Firewall

The **`system-config-firewall`** utility is a **CentOS**-specific tool that can be run in either a **GUI** or **TUI** mode.



Here you can define which services are trusted. Trusted services are accessible from all hosts and networks.

	Service	Port/Protocol	Conntrack Helper
□	Amanda Backup Client	10080/udp	amanda
□	Bacula	9101/tcp, 9102/tcp, 9103/tcp	
□	Bacula Client	9102/tcp	
□	DNS	53/tcp, 53/udp	
□	FTP	21/tcp	ftp
□	IMAP over SSL	993/tcp	
□	IPsec	/ah, /esp, 500/udp	
□	Mail (SMTP)	25/tcp	
□	Multicast DNS (mDNS)	5353/udp	
□	Network Printing Client (IPP)	631/udp	
□	Network Printing Server (IPP)	631/tcp, 631/udp	

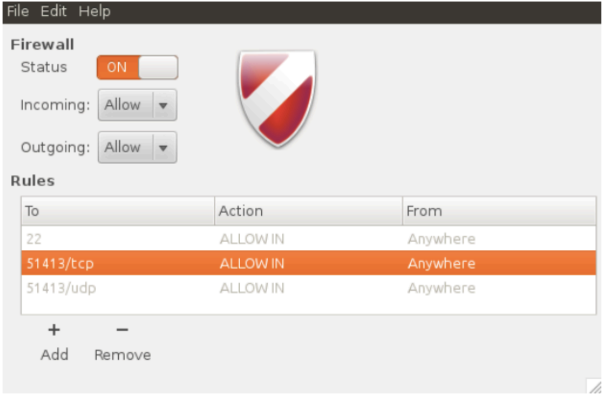
Allow access to necessary services, only.

The firewall is enabled. (modified)

Figure 15.5: System-Config-Firewall

15.14. gufw

As an **Ubuntu**-specific tool, **`gufw`** is a **GUI** wrapper of the command-line tool **`ufw`**.



Firewall

Status: ON

Incoming: Allow

Outgoing: Allow

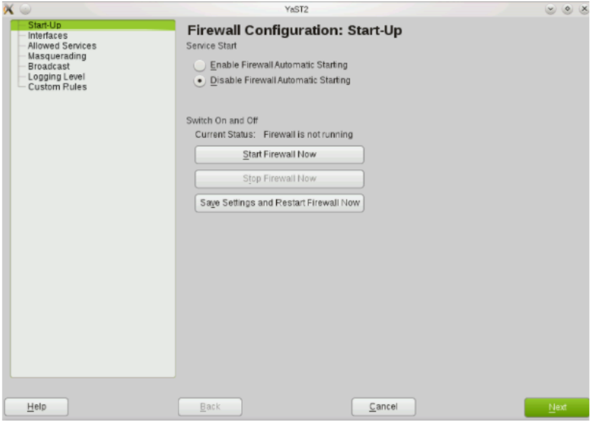
Rules

To	Action	From
22	ALLOW IN	Anywhere
51413/tcp	ALLOW IN	Anywhere
51413/udp	ALLOW IN	Anywhere

Figure 15.6: gufw

15.15. YAST Firewall

As an **OpenSUSE**-specific tool, the **`YAST firewall`** runs in a **GUI** or **TUI** mode, depending on how it is started.



Firewall Configuration: Start-Up

Service Start

○ Enable Firewall Automatic Starting

● Disable Firewall Automatic Starting

Switch On and Off

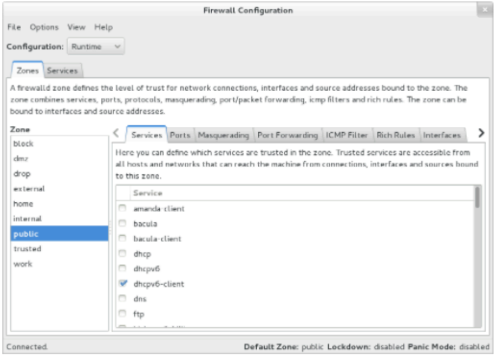
Current Status: Firewall is not running

Figure 15.7: YAST Firewall

15.16. firewalld

The **firewalld** utility provides a dynamically managed firewall:

- It is available on most distributions.
- It has a **GUI** and command line interface.
- Configuration changes are applied dynamically.
- Includes support for **IPv4**, **IPv6**, and **Ethernet** bridges.



Configuration: Runtime

A firewalld zone defines the level of trust for network connections, interfaces and source addresses bound to the zone. The zone combines services, ports, protocols, masquerading, port/packet forwarding, icmp filters and rich rules. The zone can be bound to interfaces and source addresses.

Zone

- block
- dmz
- drop
- external
- home
- internal
- public
- trusted
- work

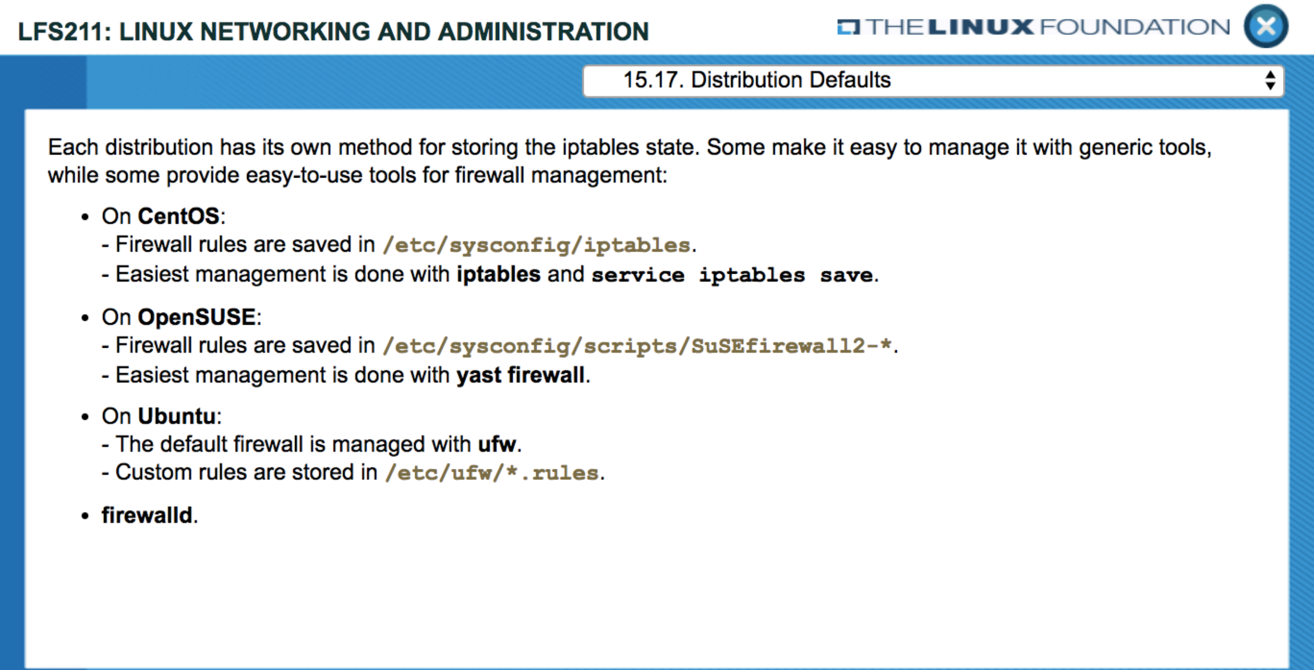
Here you can define which services are trusted in the zone. Trusted services are accessible from all hosts and networks that can reach the machine from connections, interfaces and sources bound to this zone.

- □ amanda-client
- □ bacula
- □ bacula-client
- □ dhcp
- □ dhcpv6
- ☑ dhcpv6-client
- □ dns
- □ ftp

Connected.

Default Zone: public **Lockdown:** disabled **Panic Mode:** disabled

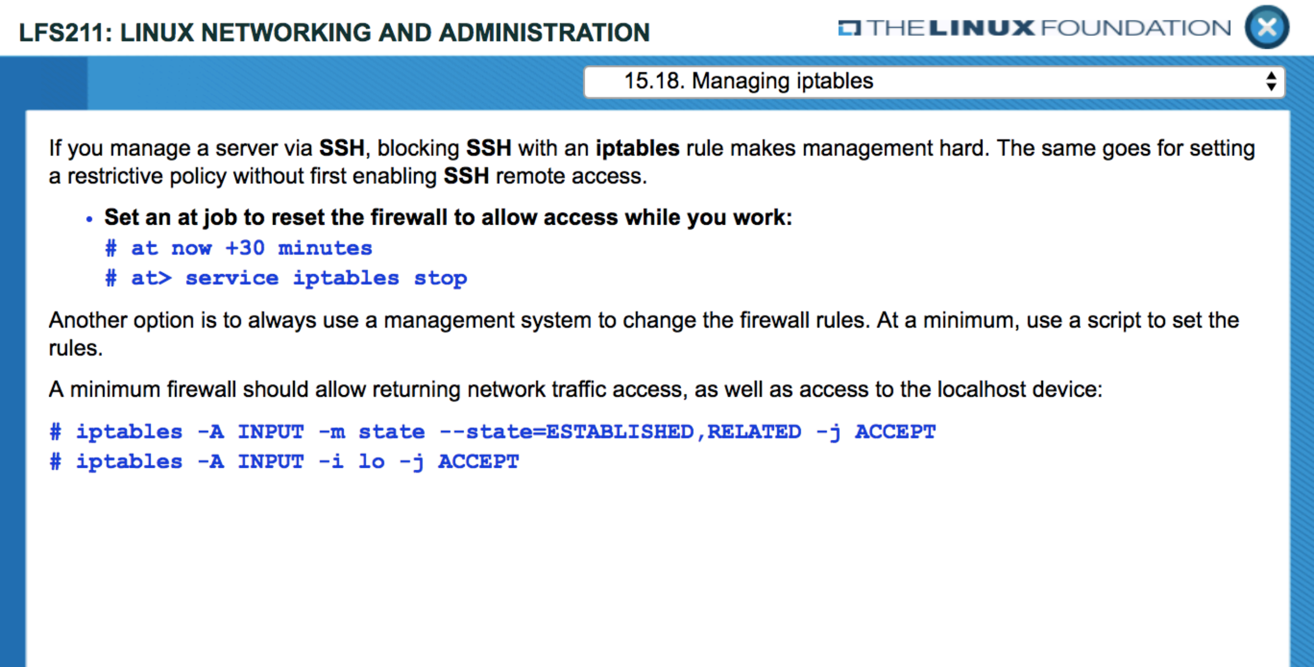
Figure 15.8: firewalld GUI



15.17. Distribution Defaults

Each distribution has its own method for storing the iptables state. Some make it easy to manage it with generic tools, while some provide easy-to-use tools for firewall management:

- On **CentOS**:
  - Firewall rules are saved in `/etc/sysconfig/iptables`.
  - Easiest management is done with **iptables** and **`service iptables save`**.
- On **OpenSUSE**:
  - Firewall rules are saved in `/etc/sysconfig/scripts/SuSEfirewall2-*`.
  - Easiest management is done with **yast firewall**.
- On **Ubuntu**:
  - The default firewall is managed with **ufw**.
  - Custom rules are stored in `/etc/ufw/*.rules`.
- **firewalld**.



15.18. Managing iptables

If you manage a server via **SSH**, blocking **SSH** with an **iptables** rule makes management hard. The same goes for setting a restrictive policy without first enabling **SSH** remote access.

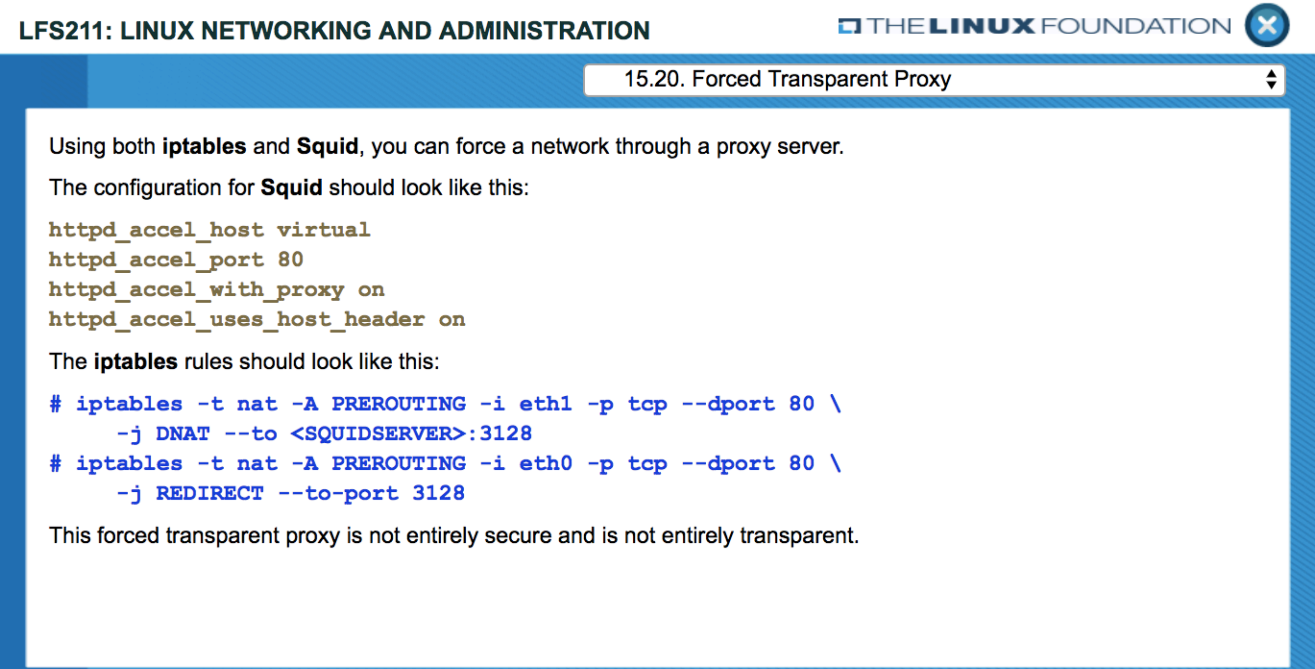
- **Set an at job to reset the firewall to allow access while you work:**

  ```
  # at now +30 minutes
  # at> service iptables stop
  ```

Another option is to always use a management system to change the firewall rules. At a minimum, use a script to set the rules.

A minimum firewall should allow returning network traffic access, as well as access to the localhost device:

```
# iptables -A INPUT -m state --state=ESTABLISHED,RELATED -j ACCEPT
# iptables -A INPUT -i lo -j ACCEPT
```



15.20. Forced Transparent Proxy

Using both **iptables** and **Squid**, you can force a network through a proxy server.

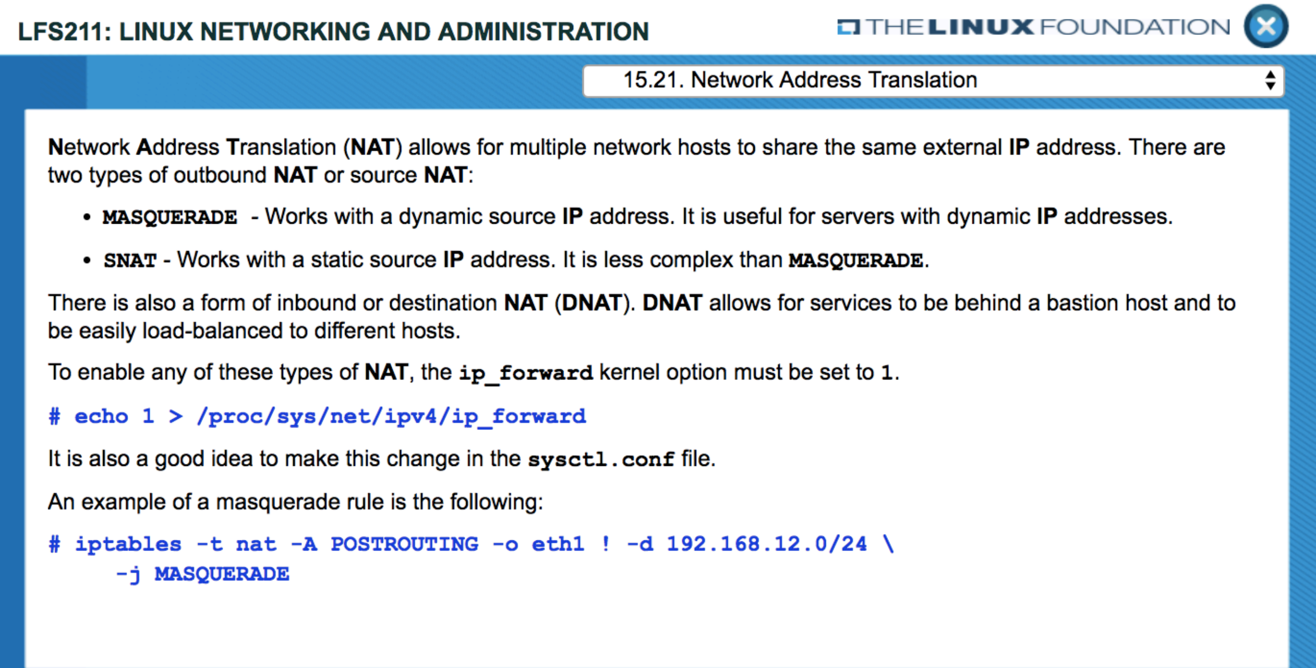
The configuration for **Squid** should look like this:

```
httpd_accel_host virtual
httpd_accel_port 80
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
```

The **iptables** rules should look like this:

```
# iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 \
     -j DNAT --to <SQUIDSERVER>:3128
# iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 \
     -j REDIRECT --to-port 3128
```

This forced transparent proxy is not entirely secure and is not entirely transparent.



15.21. Network Address Translation

Network **A**ddress **T**ranslation (**NAT**) allows for multiple network hosts to share the same external **IP** address. There are two types of outbound **NAT** or source **NAT**:

- **`MASQUERADE`** - Works with a dynamic source **IP** address. It is useful for servers with dynamic **IP** addresses.
- **`SNAT`** - Works with a static source **IP** address. It is less complex than **`MASQUERADE`**.

There is also a form of inbound or destination **NAT** (**DNAT**). **DNAT** allows for services to be behind a bastion host and to be easily load-balanced to different hosts.

To enable any of these types of **NAT**, the **`ip_forward`** kernel option must be set to **`1`**.

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

It is also a good idea to make this change in the **`sysctl.conf`** file.

An example of a masquerade rule is the following:

```
# iptables -t nat -A POSTROUTING -o eth1 ! -d 192.168.12.0/24 \
     -j MASQUERADE
```