Red Hat Enterprise Linux 4: Security Guide

Chapter 7. Firewalls http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/security-guide/ch-fw.html

Information security is commonly thought of as a process and not a product. However, standard security implementations usually employ some form of dedicated mechanism to control access privileges and restrict network resources to users who are authorized, identifiable, and traceable. Red Hat Enterprise Linux includes several powerful tools to assist administrators and security engineers with network-level access control issues.

Along with VPN solutions, such as IPsec (discussed in Chapter 6 *Virtual Private Networks*), firewalls are one of the core components of a network security implementation. Several vendors market firewall solutions catering to all levels of the marketplace: from home users protecting one PC to data center solutions safeguarding vital enterprise information. Firewalls can be standalone hardware solutions, such as firewall appliances by Cisco, Nokia, and Sonicwall. There are also proprietary software firewall solutions developed for home and business markets by vendors such as Checkpoint, McAfee, and Symantec.

Apart from the differences between hardware and software firewalls, there are also differences in the way firewalls function that separate one solution from another. Table 7-1 details three common types of firewalls and how they function:

Method	Description	Advantages	Disadvantages
NAT	*Network Address Translation* (NAT) places private IP subnetworks behind one or a small pool of public IP addresses, masquerading all requests to one source rather than several.	• Can be configured transparently to machines on a LAN • Protection of many machines and services behind one or more external IP address(es) simplifies administration duties • Restriction of user access to and from the LAN can be configured by opening and closing ports on the NAT firewall/gateway	• Cannot prevent malicious activity once users connect to a service outside of the firewall
Packet Filter	A packet filtering firewall reads each data packet that passes within and outside of a LAN. It can read and process packets by header information and filters the packet based on sets of programmable rules implemented by the firewall administrator. The Linux kernel has built-in packet filtering functionality through the Netfilter kernel subsystem.	• Customizable through the `iptables` front-end utility • Does not require any customization on the client side, as all network activity is filtered at the router level rather than the application level • Since packets are not transmitted through a proxy, network performance is faster due to direct connection from client to remote host	• Cannot filter packets for content like proxy firewalls • Processes packets at the protocol layer, but cannot filter packets at an application layer • Complex network architectures can make establishing packet filtering rules difficult, especially if coupled with *IP masquerading* or local subnets and DMZ networks
Proxy	Proxy firewalls filter all requests of a certain protocol or type from LAN clients to a proxy machine, which then	• Gives administrators control over	• Proxies are often application

Method	Description	Advantages	Disadvantages
	makes those requests to the Internet on behalf of the local client. A proxy machine acts as a buffer between malicious remote users and the internal network client machines.	what applications and protocols function outside of the LAN • Some proxy servers can cache frequently-accessed data locally rather than having to use the Internet connection to request it, which is convenient for cutting down on unnecessary bandwidth consumption • Proxy services can be logged and monitored closely, allowing tighter control over resource utilization on the network	specific (HTTP, Telnet, etc.) or protocol restricted (most proxies work with TCP connected services only) • Application services cannot run behind a proxy, so your application servers must use a separate form of network security • Proxies can become a network bottleneck, as all requests and transmissions are passed through one source rather than directly from a client to a remote service

Table 7-1. Firewall Types

7.1. Netfilter and `iptables`

The Linux kernel features a powerful networking subsystem called *Netfilter*. The Netfilter subsystem provides stateful or stateless packet filtering as well as NAT and IP masquerading services. Netfilter also has the ability to *mangle* IP header information for advanced routing and connection state management. Netfilter is controlled through the `iptables` utility.

7.1.1. `iptables` Overview

The power and flexibility of Netfilter is implemented through the `iptables` interface. This command line tool is similar in syntax to its predecessor, `ipchains`; however, `iptables` uses the Netfilter subsystem to enhance network connection, inspection, and processing; whereas `ipchains` used intricate rule sets for filtering source and destination paths, as well as connection ports for both. `iptables` features advanced logging, pre- and post-routing actions, network address translation, and port forwarding all in one command line interface.

This section provides an overview of `iptables`. For more detailed information about `iptables`, refer to the *Red Hat Enterprise Linux Reference Guide*.

7.2. Using `iptables`

The first step in using `iptables` is to start the `iptables` service. This can be done with the command:

```
service iptables start
```

Warning

The `ip6tables` services should be turned off to use the `iptables` service with the following commands:

```
service ip6tables stop
chkconfig ip6tables off
```

To make `iptables` start by default whenever the system is booted, you must change runlevel status on the service using `chkconfig`.

```
chkconfig --level 345 iptables on
```

The syntax of `iptables` is separated into tiers. The main tier is the *chain*. A chain specifies the state at which a packet is manipulated. The usage is as follows:

```
iptables -A chain -j target
```

The `-A` option appends a rule at the end of an existing ruleset. The *chain* is the name of the chain for a rule. The three built-in chains of `iptables` (that is, the chains that affect every packet which traverses a network) are INPUT, OUTPUT, and FORWARD. These chains are permanent and cannot be deleted. The `-j` *target* option specifies the location in the `iptables` ruleset where this particular rule should *jump*. Some built in targets are ACCEPT, DROP, and REJECT.

New chains (also called user-defined chains) can be created by using the `-N` option. Creating a new chain is useful for customizing granular or elaborate rules.

7.2.1. Basic Firewall Policies

Establishing basic firewall policies creates a foundation for building more detailed, user-defined rules. `iptables` uses policies (`-P`) to create default rules. Security-minded administrators usually elect to drop all packets as a policy and only allow specific packets on a case-by-case basis. The following rules block all incoming and outgoing packets on a network gateway:

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP
```

Additionally, it is recommended that any *forwarded packets* — network traffic that is to be routed from the firewall to its destination node — be denied as well, to restrict internal clients from inadvertent exposure to the Internet. To do this, use the following rule:

```
iptables -P FORWARD DROP
```

After setting the policy chains, you can create new rules for your particular network and security requirements. The following sections outline some rules you may implement in the course of building your `iptables` firewall.

7.2.2. Saving and Restoring `iptables` Rules

Firewall rules are only valid for the time the computer is on; so, if the system is rebooted, the rules are automatically flushed and reset. To save the rules so that they are loaded later, use the following command:

```
/sbin/service iptables save
```

The rules are stored in the file `/etc/sysconfig/iptables` and are applied whenever the service is started or restarted, including when the machine is rebooted.

7.3. Common `iptables` Filtering

Keeping remote attackers out of a LAN is an important aspect of network security, if not the *most* important. The integrity of a LAN should be protected from malicious remote users through the use of stringent firewall rules. However, with a default policy set to block all incoming, outgoing, and forwarded packets, it is impossible for the firewall/gateway and internal LAN users to communicate with each other or with external resources. To allow users to perform network-related functions and use networking applications, administrators must open certain ports for communication.

For example, to allow access to port 80 on the firewall, append the following rule:

```
iptables -A INPUT -p tcp -m tcp --sport 80 -j ACCEPT
iptables -A OUTPUT -p tcp -m tcp --dport 80 -j ACCEPT
```

This allows regular Web browsing from websites that communicate via port 80. To allow access to secure websites (such as https://www.example.com/), you must open port 443, as well.

```
iptables -A INPUT -p tcp -m tcp --sport 443 -j ACCEPT
iptables -A OUTPUT -p tcp -m tcp --dport 443 -j ACCEPT
```

Important

When creating an iptables ruleset, it is critical to remember that order is important. For example, if one chain that specifies that any packets from the local 192.168.100.0/24 subnet be dropped, and then another chain is appended (-A) to allow packets from 192.168.100.13 (which is within the dropped restricted subnet), then the appended rule is ignored. You must set a rule to allow 192.168.100.13 first, and then set a drop rule on the subnet.

To arbitrarily insert a rule in an existing chain of rules, use -I, followed by the chain in which to insert the rule, and a rule number (1,2,3,...,n) for where the rule should reside. For example:

```
iptables -I INPUT 1 -i lo -p all -j ACCEPT
```

The rule is inserted as the first rule in the INPUT chain to allow local loopback device traffic.

There may be times when you require remote access to the LAN from outside the LAN. Secure services such as SSH, can be used for encrypted remote connection to LAN services. For administrators with PPP-based resources (such as modem banks or bulk ISP accounts), dial-up access can be used to circumvent firewall barriers securely, as modem connections are typically behind a firewall/gateway because they are direct connections. However, for remote users with broadband connections, special cases can be made. You can configure iptables to accept connections from remote SSH clients. For example, to allow remote SSH access, the following rules may be used:

```
iptables -A INPUT -p tcp --dport 22 -j ACCEPT

iptables -A OUTPUT -p udp --sport 22 -j ACCEPT
```

There are other services for which you may need to define rules. Refer to the Red Hat Enterprise Linux Reference Guide for comprehensive information on iptables and its various options.

These rules allow incoming and outbound access for an individual system, such as a single PC directly connected to the Internet or a firewall/gateway. However, they do not allow nodes behind the firewall/gateway to access these services. To allow LAN access to these services, you can use NAT with iptables filtering rules.

7.4. FORWARD and NAT Rules

Most organizations are allotted a limited number of publicly routable IP addresses from their ISP. Due to this limited allowance, administrators must find creative ways to share access to Internet services without giving limited public IP addresses to every node on the LAN. Using private IP address is the common way to allow all nodes on a LAN to properly access internal and external network services. Edge routers (such as firewalls) can receive incoming transmissions from the Internet and route the packets to the intended LAN node. At the same time, firewall/gateways can also route outgoing requests from a LAN node to the remote Internet service. This forwarding of network traffic can become dangerous at times, especially with the availability of modern cracking tools that can spoof internal IP addresses and make the remote attacker's machine act as a node on your LAN. To prevent this, iptables provides routing and forwarding policies that can be implemented to prevent aberrant usage of network resources.

The FORWARD policy allows an administrator to control where packets can be routed within a LAN. For example, to allow forwarding for the entire LAN (assuming the firewall/gateway is assigned an internal IP address on eth1), the following rules can be set:

```
iptables -A FORWARD -i eth1 -j ACCEPT

iptables -A FORWARD -o eth1 -j ACCEPT
```

This rule gives systems behind the firewall/gateway access to the internal network. The gateway routes packets from one LAN node to its intended destination node, passing all packets through its eth1 device.

 Note

By default, the IPv4 policy in Red Hat Enterprise Linux kernels disables support for IP forwarding, which prevents boxes running Red Hat Enterprise Linux from functioning as dedicated edge routers. To enable IP forwarding, run the following command:

```
sysctl -w net.ipv4.ip_forward=1
```

If this command is run via shell prompt, then the setting is not remembered after a reboot. You can permanently set forwarding by editing the /etc/sysctl.conf file. Find and edit the following line, replacing 0 with 1:

```
net.ipv4.ip_forward = 0
```

Execute the following command to enable the change to the sysctl.conf file:

```
sysctl -p /etc/sysctl.conf
```

Accepting forwarded packets via the firewall's internal IP device allows LAN nodes to communicate with each other; however they still are not allowed to communicate externally to the Internet. To allow LAN nodes with private IP addresses to communicate with external public networks, configure the firewall for IP masquerading, which masks requests from LAN nodes with the IP address of the firewall's external device (in this case, eth0):

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

The rule uses the NAT packet matching table (-t nat) and specifies the built-in POSTROUTING chain for NAT (-A POSTROUTING) on the firewall's external networking device (-o eth0). POSTROUTING allows packets to be altered as they are leaving the firewall's external device. The -j MASQUERADE target is specified to mask the private IP address of a node with the external IP address of the firewall/gateway.

If you have a server on your internal network that you want make available externally, you can use the -j DNAT target of the PREROUTING chain in NAT to specify a destination IP address and port where incoming packets requesting a connection to your internal service can be forwarded. For example, if you wanted to forward incoming HTTP requests to your dedicated Apache HTTP Server server system at 172.31.0.23, run the following command:

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT \

    --to 172.31.0.23:80
```

This rule specifies that the NAT table use the built-in PREROUTING chain to forward incoming HTTP requests exclusively to the listed destination IP address of 172.31.0.23.

 Note

If you have a default policy of DROP in your FORWARD chain, you must append a rule to allow forwarding of incoming HTTP requests so that destination NAT routing can be possible. To do this, run the following command:

```
iptables -A FORWARD -i eth0 -p tcp --dport 80 -d 172.31.0.23 -j ACCEPT
```

This rule allows forwarding of incoming HTTP requests from the firewall to its intended destination of the Apache HTTP Server server behind the firewall.

7.4.1. DMZs and `iptables`

iptables rules can be set to route traffic to certain machines, such as a dedicated HTTP or FTP server, in a demilitarized zone (DMZ) — a special local subnetwork dedicated to providing services on a public carrier such as the Internet. For example, to set a rule for routing incoming HTTP requests to a dedicated HTTP server at 10.0.4.2 (outside of the 192.168.1.0/24 range of the LAN), NAT calls a PREROUTING table to forward the packets to their proper destination:

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT \

        --to-destination 10.0.4.2:80
```

With this command, all HTTP connections to port 80 from the outside of the LAN are routed to the HTTP server on a separate network from the rest of the internal network. This form of network segmentation can prove safer than allowing HTTP connections to a machine on the network. If the HTTP server is configured to accept secure connections, then port 443 must be forwarded as well.

7.5. Viruses and Spoofed IP Addresses

More elaborate rules can be created that control access to specific subnets, or even specific nodes, within a LAN. You can also restrict certain dubious services such as trojans, worms, and other client/server viruses from contacting their server. For example, there are some trojans that scan networks for services on ports from 31337 to 31340 (called the elite ports in cracking terminology). Since there are no legitimate services that communicate via these non-standard ports, blocking it can effectively diminish the chances that potentially infected nodes on your network independently communicate with their remote master servers.

```
iptables -A OUTPUT -o eth0 -p tcp --dport 31337 --sport 31337 -j DROP

iptables -A FORWARD -o eth0 -p tcp --dport 31337 --sport 31337 -j DROP
```

You can also block outside connections that attempt to spoof private IP address ranges to infiltrate your LAN. For example, if your LAN uses the 192.168.1.0/24 range, a rule can set the Internet facing network device (for example, eth0) to drop any packets to that device with an address in your LAN IP range. Because it is recommended to reject forwarded packets as a default policy, any other spoofed IP address to the external-facing device (eth0) is rejected automatically.

```
iptables -A FORWARD -s 192.168.1.0/24 -i eth0 -j DROP
```

 Note

There is a distinction between the DROP and REJECT targets when dealing with appended rules. The REJECT target denies access and returns a connection refused error to users who attempt to connect to the service. The DROP target, as the name implies, drops the packet without any warning. Administrators can use their own discretion when using these targets. However, to avoid user confusion and attempts to continue connecting, the REJECT target is recommended.