Linux Firewalls

Appalachian State University

Andrew R. Dalton
ard@cs.appstate.edu
UNIX System Administration and Security
What is a “Firewall?”

- A *firewall* is a scheme that prevents unauthorized users from gaining access to a computer network or that monitors transfers of information to and from the network.

- The Linux kernel subsystem *netfilter* is used to provide firewalling support.
IP Networking Overview

User Space

Kernel Space

TCP
UDP
ICMP
IP

Controller

Hardware

User Programs, Daemons (httpd, tcpd, etc...)

Serves as an abstraction of the networking components. Provides an API for accessing networking functionality to the higher levels.

Controller (driver) for networking hardware.

Physical Media

Hardware
Sample Telnet Connection

Client > telnet server

telnetd or tcpd to relay to telnetd

Controller

Physical Media

Hardware

TCP | UDP | ICMP

User Space

Kernel Space

IP

User Space

Kernel Space

IP

Controller

Hardware

Server

Client
Telnet Exchange

- If a daemon (either telnetd or tcpd) is listening on the telnet port of the server, the TCP subsystems of both the client and the server perform a “handshake” that establishes the TCP connection:

  15:14:11.813638 client.32808  > server.telnet: S 2888421569:2888421569(0)
  15:14:11.814256 server.telnet > client.32808:  S 2519825712:2519825712(0) ack 2888421570
  15:14:11.814295 client.32808  > server.telnet: . ack 1
Telnet Exchange (cont.)

- Without a firewall, if a daemon is not listening, the remote end TCP subsystem will send a RESET message to the originator (as per the TCP protocol):

  15:31:32.701694 client.32809  > server.telnet: S 3950126136:3950126136(0)
  15:31:32.702245 server.telnet > client.32809:  R 0:0(0) ack 3950126137
So? What's Wrong with That?

- If the remote machine established any sort of connection (even something like TCP Wrapper), the particulars of TCP implementations can be used to determine the OS type of the far end machine. Here I'm running Linux 2.4.20:

  ```
  ./queso -c ./queso.conf localhost -p 22
  127.0.0.1:22    * Standard: Solaris 2.x, Linux 2.1.???, MacOS
  ```
If a daemon isn't running on the specified port, the simple RESET response indicated that there's actually a machine at that IP address.

We want to prevent an attacker from collecting any information about our machine. We don't want an attacker to even know we're here.

Firewalls to the rescue!
Linux Netfilter Firewall

- Netfilter is a kernel-level package management subsystem that can provide Network Address Translation (NAT), Firewalling, as well as other low-level services.

- *iptables* is a user-level application that can be used to communicate policies to netfilter.

- Firewall rules can be used to, among other things, completely drop unwanted packets.
Telnet Dropped by Firewall

User Space

Kernel Space

TCP  UDP  ICMP

IP

Controller

Hardware

Client

Physical Media

User Space

Kernel Space

TCP  UDP  ICMP

IP

Netfilter

Controller

Hardware

Server

> telnet server
Telnet Blocked by Firewall (cont.)

Here, the firewall has been configured to drop all telnet packets from the client. The connection is not established (even though the daemon is running) and no RESET message is sent:

```
16:25:17.819995 client.32826 > server.telnet: S 2950843080:2950843080(0)
16:25:20.812400 client.32826 > server.telnet: S 2950843080:2950843080(0)
16:25:26.812402 client.32826 > server.telnet: S 2950843080:2950843080(0)
16:25:38.812416 client.32826 > server.telnet: S 2950843080:2950843080(0)
```
A firewall rule specifies criteria for a packet, and a target. If the packet does not match, the next rule in the chain is then examined; if it does match, then the next rule is specified by the value of the target, which can be the name of a user-defined chain or one of the special values ACCEPT, DROP, QUEUE, or RETURN.
Firewall Configuration (cont.)

- **ACCEPT** means to let the packet through.
- **DROP** means to drop the packet on the floor.
- **QUEUE** means to pass the packet to userspace (if supported by the kernel).
- **RETURN** means stop traversing this chain and resume at the next rule in the previous (calling) chain.
iptables is capable of configuring multiple tables within the kernel. The table responsible for firewall activity is the **filter table**.

The filter table contains the built-in chains **INPUT** (for packets coming into the box itself), **OUTPUT** (for locally-generated packets), and **FORWARD** (for packets being routed through the box).
Filter Table (cont.)

- All network functions can be controlled by these tables on an address and protocol/port basis.
- Rules can be added to accept/drop all connections from a given destination, for a given protocol/port, or any combination.
- The collection of these rules define the firewall policy for that machine.
Sample Firewall Rules

- Block all incoming TCP/UDP/ICMP connections:

  ```
  iptables --append INPUT --table filter --proto tcp --source 0/0 --jump DROP
  iptables --append INPUT --table filter --proto udp --source 0/0 --jump DROP
  iptables --append INPUT --table filter --proto icmp --source 0/0 --jump DROP
  ```

- This, however, prevents you from receiving any incoming network traffic (even in response to connections you try to establish).
Sample Rules (cont.)

- We don't really want to block all incoming TCP traffic, just connection attempts:

  ```
  iptables --append INPUT --table filter --proto tcp --syn --jump DROP
  iptables --append INPUT --table filter --proto udp --source 0/0 --jump DROP
  iptables --append INPUT --table filter --proto icmp --source 0/0 --jump DROP
  ```

- The `--syn` option only matches TCP packets with the SYN bit set and the ACK and FIN bits cleared. It is equivalent to

  ```
  --tcp-flags SYN,RST,ACK SYN.
  ```
Sample Rules (cont..)

- But what if I want to be able to SSH into my network? Here, we add a new rule to allow incoming traffic on the SSH port (22):

  ```
  iptables --append INPUT --table filter --proto tcp --syn \ 
  --destination-port 22 --jump ACCEPT
  iptables --append INPUT --table filter --proto tcp --syn \ 
  --jump DROP
  iptables --append INPUT --table filter --proto udp --source 0/0 --jump DROP
  iptables --append INPUT --table filter --proto icmp --source 0/0 --jump DROP
  ```

- Remember that the first rule that matches is taken so it is important that the ACCEPT rule be listed before the DROP rule.
Doh! Now we have a big gaping hole in our firewall. We really only want a selected machine to SSH in:

```bash
iptables --append INPUT --table filter --proto tcp --syn --source 192.168.1.20/32 --destination-port 22 --jump ACCEPT
iptables --append INPUT --table filter --proto tcp --syn --jump DROP
iptables --append INPUT --table filter --proto udp --source 0/0 --jump DROP
iptables --append INPUT --table filter --proto icmp --source 0/0 --jump DROP
```

This allows only a machine with IP address 192.168.1.20 to ssh into the box.
Ok, but now, I don't receive DHCP responses and my network connection drops! This is because we've blocked all incoming udp traffic.

```
iptables --append INPUT --table filter --proto tcp --syn --source 192.168.1.20/32 --destination-port 22 --jump ACCEPT
iptables --append INPUT --table filter --proto udp --source 192.168.1.1/32 --destination-port 68 --jump ACCEPT
iptables --append INPUT --table filter --proto tcp --syn --jump DROP
iptables --append INPUT --table filter --proto udp --source 0/0 --jump DROP
iptables --append INPUT --table filter --proto icmp --source 0/0 --jump DROP
```

We allow incoming UDP datagrams from our dhcp server (192.168.1.1) to our dhcp client.
Firewall Tweaking

- As you can see, there are many issues to consider when developing/tweaking your firewall rules.
- You may end up blocking too much (or too little!) by developing the firewall rules by hand.
- You might want to consider a prebuilt/tested firewall package that sets up netfilter for you.
Several distributions come with firewall configurations (RedHat has **lokkit**).

- **gShield** [http://muse.linuxmafia.org/gshield.html](http://muse.linuxmafia.org/gshield.html)
- **Shorewall** [http://shorewall.sourceforge.net](http://shorewall.sourceforge.net)
Things to Remember

- No firewall is perfect.
- Any hole in your firewall makes it less secure so be careful.
- Always try to minimize the the information a potential cracker can collect about your computer system. The more she/he knows, the more she/he has to use against you.
Further Reading

- Netfilter & iptables at:
  http://www.netfilter.org or
  http://www.iptables.org

- Linux Advanced Routing and Traffic Controll:
  http://lartc.org

- Linux Firewall and Security Site:
  http://www.linux-firewall-tools.com/linux
Any Questions?