

Linux Firewall - the Traffic Shaper

Jan Van Oorschot 2001-01-15

Linux Firewall - the Traffic Shaper

by *Jeroen Wortelboer* and *Jan Van Oorschot*

last updated Jan 15, 2001

The time that an 'always-on' Internet link was only for the very rich or the very nerdy is finally behind us. Nowadays, for less than \$100 per month, home computers and small business computers can be connected to the Net without having to dial in.

Typically, the connection to the home or office (the so-called 'Last Mile' or 'Local-Loop') is implemented by a local Internet Service Provider (ISP) using the local cable or some (A)DSL infrastructure. In any case, for a low fixed monthly fee, there is this long awaited connector labeled 'The Net' just begging to be attached to the internal network at home or in the office. However, warned by all the alarming reports about hacker vandalism, many people refrain from connecting their computers directly to the Internet. Instead, home and small business users are utilizing simple PC's powered by a masquerading and firewalling Linux kernel are frequently used to protect their internal network from hacker vandalism.

As most readers know, the firewall is a fundamental component of all computer security strategies. The firewall is positioned between the 'always-on' Internet connection provided by the local Internet Service Provider (ISP) and the Internet connection. It examines incoming and outgoing packets, marks them according to some criteria and allows or denies access based on the firewall's policy. However, the simple firewall is not only restricted to safeguarding the user's valuable information - it can also optimize the user's bandwidth.

This article will look at ways for users to get more out of that faithful but somewhat dull firewall. In particular, we will look at traffic shaping, a technique that prevents high-bandwidth traffic like Napster from making other Internet applications, such as Web browsing and gaming, unusable. By making some simple adjustments to the Linux kernel, users can implement an effective traffic shaping setup that ensures that the Web traffic can flow smoothly, even when a lot of outsiders are busy working with your Napster store. By restricting certain types of traffic which may otherwise dominate the Internet link, firewalls can not only optimize bandwidth but can also serve as an effective tool against certain types of 'Denial of Access' attacks.

Setting up the Linux Firewall

In order to preface the discussion of firewalls as a traffic shaper, let us introduce a simple firewall setup for a typical small office or home setup - no demilitarized zone (DMZ), no web servers and no mail servers. We will allow DNS and ICMP traffic for network management, mail, news and Napster and of course WWW traffic to pass through our firewall. Figure 1 shows our setup.

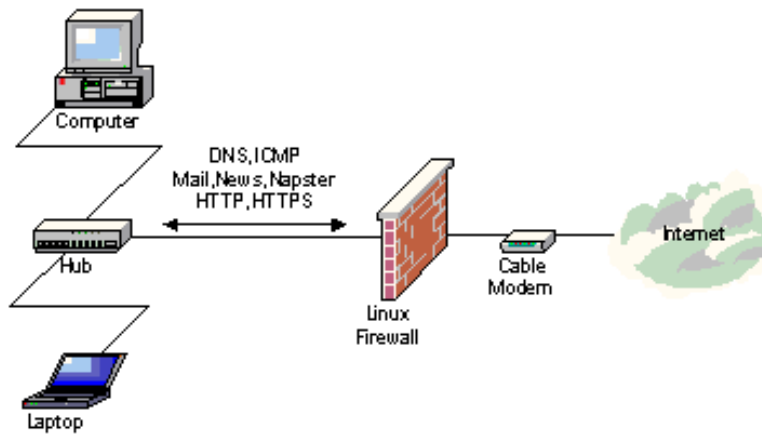


Figure 1: Protecting the computers inside by using a dedicated Linux Firewall

First of all, make sure that the Linux kernel supports Advanced Routing. The standard RedHat 2.2.14 kernel doesn't, so a recompile is necessary. Configure the kernel as an Advanced Router and enable all the Quality of Service options.

Next, you will need to download the `iproute2` packages from somewhere. We found the main site hard to reach, so we used one of the mirror sites at <ftp://ftp.sunet.se/pub/Linux/ip-routing/>. Compile it and try a `./tc qdisc show`. If it doesn't respond with an error message, you're in business. If it responds with "RTNETLINK answers: invalid argument" then your kernel probably lacks Advanced Routing support.

Now that we have our Linux box running the correct software, it is time to configure the firewall. The firewall policy script is shown below. When this configuration is activated, only the specified traffic will be allowed to pass through the firewall. Additionally, the permitted traffic is divided in four groups. ICMP, DNS and general TCP traffic with the SYN bit are put in group 1, mail traffic in group 2, Napster and news traffic in group 3, and, finally, WWW traffic is put in group 4. These groups have no real function for the firewall's security functionality, but will come in handy when we start shaping our traffic, which is the next task at hand. The firewall policy script is as follows:

```
# flush all previous rules
/sbin/ipchains -F input
/sbin/ipchains -F output
/sbin/ipchains -F forward

# apply the default DENY policy
/sbin/ipchains -P input DENY
/sbin/ipchains -P output DENY
/sbin/ipchains -P forward DENY

# allow simple dns traffic
/sbin/ipchains -A input -p udp -s 0/0 53 -j ACCEPT
/sbin/ipchains -A output -p udp -d 0/0 53 -j ACCEPT

# allow icmp traffic mark it with value 1
/sbin/ipchains -A input -p ICMP -i ppp+ -j ACCEPT -m 1
```

```

/sbin/ipchains -A output -p ICMP -i ppp+ -j ACCEPT -m 1
/sbin/ipchains -A input -y -p tcp -i ppp+ -j ACCEPT -m 1

# mark incoming mail traffic with mark value 2
/sbin/ipchains -A input ! -y -p tcp -i ppp+ -s 0/0 smtp -d 0/0 -j
ACCEPT -m 2
/sbin/ipchains -A input ! -y -p tcp -i ppp+ -s 0/0 pop3 -d 0/0 -j
ACCEPT -m 2

# mark incoming Napster and News traffic with mark value 3
/sbin/ipchains -A input ! -y -p tcp -i ppp+ -s 0/0 6699 -d 0/0 -j
ACCEPT -m 3
/sbin/ipchains -A input -p tcp -i ppp+ -s 0/0 -d 0/0 6699 -j
ACCEPT -m 3
/sbin/ipchains -A input ! -y -p tcp -i ppp+ -s 0/0 nntp -d 0/0 -j
ACCEPT -m 3

# mark incoming www traffic with mark value 4
/sbin/ipchains -A input ! -y -p tcp -i ppp+ -s 0/0 www -d 0/0 -j
ACCEPT -m 4
/sbin/ipchains -A input ! -y -p tcp -i ppp+ -s 0/0 https -d 0/0 -j
ACCEPT -m 4

/sbin/ipchains -A output -p tcp -i ppp+ -j ACCEPT
/sbin/ipchains -A input ! -y -p tcp -i ppp+ -j ACCEPT

# we trust our local interfaces
/sbin/ipchains -A input -p all -i eth0 -j ACCEPT
/sbin/ipchains -A output -p all -i eth0 -j ACCEPT
/sbin/ipchains -A input -p all -i lo -j ACCEPT
/sbin/ipchains -A output -p all -i lo -j ACCEPT

# masquerade our internal LAN to the outside world
/sbin/ipchains -I forward -p all -s 192.168.0.0/16 -d 0/0 -j MASQ

```

Explaining Traffic Shaping and Quality of Service

Normally, the TCP/IP stack in the Linux kernel will try to divide the available bandwidth evenly among the contending connections. While this seemed like a good idea at the time of development, it means that if there are a hundred Napster connections active, a Web session will only get 1% of the Internet connection. (Perhaps one of the reasons why the Web is being called 'World Wide Wait'.) Traffic shaping is the art of reorganizing this distribution mechanism of TCP/IP. Certain types of connections are given more importance than others, and get more bandwidth if there is not enough to go around.

Bandwidth distribution has to be reorganized at a point where knowledge of the other connections is available. The best place to do this is at a connection choke point - in this case, the firewall/router. At this point, the relatively fast local area network passes into the relatively slow Internet connection, and the connections have to contend for bandwidth. Since not every kind of traffic is equally important, an equal distribution of the connections over the link might not provide the proper setup for optimal distribution. For instance, internet traffic like e-mail is probably more

important than postings to and from a discussion group.

Using Traffic Shaping to Protect Against DOS Attacks

In addition to allowing the user to prioritize bandwidth usage, traffic shaping at the firewall gives the user another advantage - increased protection against DOS attacks. SYN floods and ICMP floods are well known Denial of Service attacks that standard firewall filtering will not protect against. Normally, these packets take up a small amount of the available bandwidth, but during a flood the trickle becomes a downpour. Traffic shaping can diminish the possibility of effective Denial of Service by limiting the amount of bandwidth this traffic can consume.

So how should users configure their firewalls to protect against Denial of Service attacks. In the Linux kernel, there are many different ways of identifying traffic for the purpose of traffic shaping; however, given the limited scope of this article, we will restrict the discussion to a simple but effective mechanism.

We use the 'ipchains' command to mark incoming packets with a certain group number. Matching outgoing packets are consequently given bandwidth according to these groups. So, we identify traffic when it enters the firewall, and assign bandwidth accordingly when it leaves the firewall.

After we have divided the possible traffic into groups, we have to decide what the preferred bandwidth mixture will be when we are running at full capacity. When we are not at full capacity, we can let the different groups benefit from available bandwidth that is left in other groups. This borrowing of unused bandwidth from other groups is the variable part of bandwidth allocation. However during periods of full capacity, the designated allocation of bandwidth must be maintained so that all services will have accessibility.

There will be situations in which we want to prevent a traffic group from using another group's bandwidth. Such could be the case for ICMP and TCP that is still in the buildup-handshake state. Traffic based upon these protocols is frequently used in Denial of Service attacks. Limiting the amount of bandwidth for these traffic groups will not prevent the Denial of Service attack, but will effectively limit the damage of it.

Another way of making use of traffic shaping for security purposes is limiting the amount of ICMP-unreachables that a firewall can return within a given amount of time. This way, a proper portscan would take much more time to do. As long as there is no answer to a portscan packet, the portscanner will not know for sure if the port is open or not. Slow scanning is the only way to know for sure and when the firewall is protecting a large number of hosts; however, slow scanning could take weeks. During that time, the portscanner is clearly visible and that's something most portscanners will try to avoid. This can be done by defining traffic shape classes for the PPP interface, just as we did for the eth0 interface.

Demonstrating Traffic Shaping in a Linux Firewall

With the Linux firewall up and running, and the allowed traffic grouped according to the user's access policy, it is time to implement traffic shaping. When running at full capacity, we want HTTP traffic to be assigned 70% of the available bandwidth, mail 20%, Napster and news 5% and ICMP and TCP-SYN traffic the remaining 5%. Figure 2 shows this division of bandwidth.

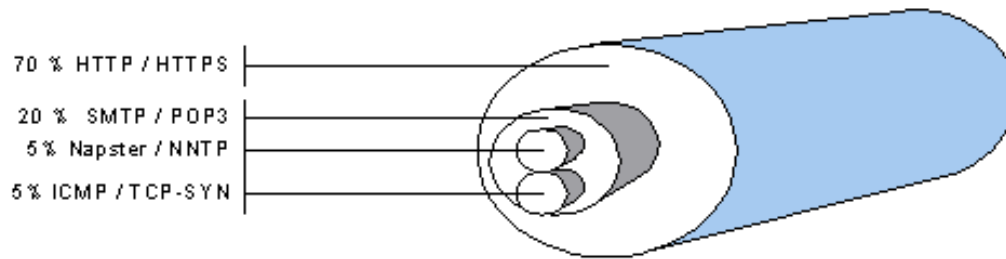


Figure 2: An Illustration of a Linux Firewall Traffic Shaping Policy

This traffic shaping policy is implemented by using the 'tc' program from the 'iproute2' package. The script to implement the policy divides a 64kbit PPP link into the four traffic groups, and reads as follows:

```
#!/bin/sh
```

```
TC=/sbin/tc
```

```
add_class() {
# $1=parent class $2=classid $3=hiband $4=lowband $5=handle $6=style
$TC class add dev eth0 parent $1 classid $2 cbq bandwidth 10Mbit          rate $3
allot 1514 weight $4 prio 5  maxburst 20 avpkt 1000 $6
$TC qdisc add dev eth0 parent $2 sfq quantum 1514b perturb 15
$TC filter add dev eth0 protocol ip prio 3 handle $5 fw classid $2
}

$TC qdisc add dev eth0 root handle 10: cbq bandwidth 10Mbit avpkt 1000
$TC class add dev eth0 parent 10:0 classid 10:1 cbq bandwidth 10Mbit    rate 64kbit
allot 1514 weight 6.4kbit prio 8 maxburst 20 avpkt
1000 bounded

# first type of traffic ICMP, TCP-SYN, DNS will be marked '1' by the
firewall code
# we will give it a bounded bandwidth of 5% of our total incoming
bandwidth (64*0.05=3.2)
add_class 10:1 10:100 3.2kbit 0.32kbit 1 bounded

# second type of traffic SMTP,POP3 will be marked '2' by the
firewalling code
# we will give it a bounded bandwidth of 20% of our total incoming
bandwidth (64*0.20=12.8)
add_class 10:1 10:200 12.8kbit 1.28kbit 2

# third type of traffic Napster will be marked '3' by the firewalling
code
# we will give it a bounded bandwidth of 5% of our total incoming
bandwidth (64*0.05=3.2)
add_class 10:1 10:300 3.2kbit 0.32kbit 3

# last type of traffic is interactive traffic. It will be marked '4'
```

```
by the firewalling code
# we will give it a bounded bandwidth of 70% of our total incoming
bandwidth (64*0.70=44.8)
add_class 10:1 10:400 44.8kbit 4.48kbit 4

exit 0
```

The script should be started right after the firewall policy script during the initialization phase of the firewall. Depending on the distribution, put the two in `/etc/rc.d/init.d` and link to them from the `runlevel` directory or simply call the scripts from the `/etc/rc.d/rc.local` script.

The script will first create a root class and a class that defines a 64kbit link in the 10Mbit Ethernet link. Our four traffic classes are derived from that class by calling the script's `add_class` function. Note the last argument of the function. It can make the class 'bounded', which means that the class will not be able to borrow bandwidth from other classes.

Last Thoughts and Further Reading

Testing the setup is probably the most difficult thing of all. A Ping flood (`ping -f`) can be used to test the ICMP traffic limitation. There are also some other, more sophisticated programs to test the ICMP bandwidth, like 'bing'.

Extending the policy with some additional rules that mark special traffic can be of great help. For example, a program like 'tcpspray' for instance can generate a TCP stream to a discard port, and with the proper adjustments in the policy file, that traffic will belong to one of the traffic groups so that the bandwidth for that traffic group gets used.

Another option is to use a sniffer like 'tcpdump' to monitor the amount of bandwidth used for a certain number of ports. Although time consuming, this will give you an overall picture of the distribution of the bandwidth in a 'production' situation.

Another way of making use of traffic shaping for security purposes is limiting the amount of ICMP-unreachables that a firewall can return within a given amount of time. This way, a proper portscan would take much more time to do. As long as there is no answer to a portscan packet, the portscanner will not know for sure if the port is open or not. Slow scanning is the only way to know for sure and when the firewall is protecting a large number of hosts; however, slow scanning could take weeks. During that time, the portscanner is clearly visible and that's something most portscanners will try to avoid. This can be done by defining traffic shape classes for the PPP interface, just as we did for the eth0 interface.

Relevant Links

[Linux 2.4 Advanced Routing HOW-TO](#)

[Linux Iproute2: Traffic Control & Friends](#)

[Iproute2 Notes](#)

[Privacy Statement](#)

Copyright 2006, SecurityFocus