TECHNISCHE UNIVERSITÄT BERLIN
Fakultät IV – Elektrotechnik und Informatik
Fachgebiet Intelligente Netze und Management verteilter Systeme
Prof. Anja Feldmann, Ph.D.
A. Mehmood, T. Hühn, L. Iannone, W. Mühlbauer, G. Schaffrath

Praktikum RouterLab SS 2009
Work Sheet 9: Networking with Linux – Packet Filtering, Traffic Shaping

Instead of Cisco or Juniper routers, a Linux system can be used as well to forward packets. We have already seen that there is software for Linux that implements complex routing procotols such as OSPF and BGP. In this assignment, we do not focus on routing, but on controlling and adjusting what happens in the data plane, the actual forwarding. We will firewall off our system from the evil Internet, tunnel through the firewall and control the bandwidth that is utilized.

Table 1: Assignment of devices to groups

Gruppe	Ham-Cloud	Muc-Cloud
Switches	ham-sc1	muc-sc1
Infrastucture IP range	`10.1.0.0/16`	`10.2.0.0/16`
Host A	loadgen102-ham	loadgen102-muc
Linux-Router	loadgen103-ham	loadgen103-muc
Host B	loadgen104-ham	loadgen104-muc
VLAN ID range	1XX	2XX

Question 1: (10 Points) *Linux as a router*

The Linux kernel provides extensive support for TCP/IP networking (as do many other UNIXes). In this worksheet we are going to investigate a few relevant features.

As we have already seen, forwarding packets is the primary functionality of a router. Of course, this mostly makes sense if a host has more than just one interface – but keep in mind that you can reuse one physical interface for multiple VLANs. For security reasons, *packet forwarding* is deactivated in the Linux kernel by default.

In this assignment, you will build a simple three node setup to demonstrate basic routing within the Linux kernel, using three Loadgens: `Host A`, `Linux-Router`, `Host B`. Your setup should allow `Host A` to communicate with `Host B` via `Linux-Router`, where `Host A` and `Host B` are in different subnets/VLANs. `Linux-Router` should forward packets on Layer 3 from one (virtual) interface to another.

(a) Draw a topology map that shows your assignment of IP addresses, interfaces and VLANs.

(b) Configure your 3 loadgens and your switch so that `Host A` can communicate with `Host B` in the manner described above. Check with `ping`. Remember that you have to enable packet forwarding in the `Linux-Router` for this to work.

Question 2: (35 Points) *Packet filtering using iptables*

In general, you don't want to deploy this setup on edge of the real Internet. After all, the Internet *is* a dangerous place, with all kinds of malicious parties trying to get hold of your precious data. So you may want to control who sends which data to whom. Luckily, Linux is exceptionally well-equipped for this challenge as well: The Netfilter framework and its frontend tool `iptables`.

(a) To validate our firewall skills, we use the simple test service `echo`. `echo` is a very basic TCP service that will just echo back anything that it receives line-by-line. Echo runs on TCP port 7. To activate this service on our loadgens, edit the configuration of the *Internet Superserver* `xinetd`.

You have to tweak `/etc/xinet.d/echo` and then restart the `xinetd` (`/etc/init.d/xinetd restart`). Now check with telnet on `Host A`, whether you can use the echo service on `Host B`.

(b) Familiarize yourself with `iptables`. You should now be able to add some simple packet filter on *Linux-Router* that controls which traffic is forwarded between *Host A* and Host B (and vice versa). Build the following filters and demonstrate their function:

- *echo* is blocked in both directions
- ICMP is blocked in both directions
- Ping only works in one direction (you will have to filter ICMP types echo-request und echo-reply appropriately)
- Everything except for `ssh` from `Host A` to `Host B` is blocked

(c) Now limit access to the `Linux-Router` itself. Only `Host A` should be able to access `Linux-Router`'s `ssh` port.

(d) `Iptables` gives you a lot of options concerning what to do with a packet that has matched a rule. Explain 5 of these options and demonstrate their usage by configuring an example.

Question 3: (20 Points) *Stateful inspection*

Now we are able to block specific services based on their port numbers. However, having a separate rule for every single service in a network is cumbersone, especially for services such as DNS that have a lot of seemingly unrelated UDP packets flying around. Rather we want to allow a *Trusted* network on one side of our firewall to talk to the *Internet*, while the *Untrusted* side should only be able to answer to valid requests from the trusted side. This feature can be achieved quite elegantly with *Stateful Inspection.*

(a) Let `Host A` be our trusted home network, and `Host B` be the untrusted network. Implement a stateful rule set on `Linux-Router` that allows Host A to access `ssh`, DNS and webserver on `Host B`, but NOT vice versa. Use stateful inspection to guarantee that only valid answer packets pass the firewall. Use `iptables` `LOG` target to log all dropped packets to the syslog.

(b) Connect via *FTP* (using the `ftp` client) from `Host A` to `Host B`. Try to download (`get`) the file `test.tex` from the default directory. Does it work? Explain! If it does not work, try to find an elegant solution to this that does not compromise the security of your system. (Hint: look at the different *conntrack* helper modules available.)

(c) To make sure that our firewall is as tightly closed as possible, we will now launch a port scan from `Host B` to `Host A`. We will use the tool `nmap` for this. Launch a port scan on host `Host B` that scans all TCP ports up to 1024. All clear?

Question 4: (35 Points) *Traffic shaping*

That concludes our firewall specific experiments. Please make sure that you remove all firewall rules and that the chain policies are set to `ACCEPT` before continuing.

Finally, we will investigate the traffic shaping features offered by the Linux kernel, e.g., limiting the amount of bandwidth for individual traffic classes.

(a) Accomodate yourself with the possibilities offered by Linux in terms of traffic shaping. Take a look at the Linux Advanced Routing & Traffic Control HOWTO.

(b) We now want to limit the bandwidth between our two test hosts by configuring the `Linux-Router`. Do not modify `Host A` oder `Host B` in this part of the assignment. Which steps are necessary, and how do the involved mechanisms work? Test your setup. Measure the available bandwidth, e.g., by transmitting a larger file via `scp`.

(c) Now setup the traffic shaping on your `Linux-Router` to emulate the bandwidth characteristics of a TDSL-1000 line (`Host A` being the customer). Watch how `ping` times change while performing a longer file transfer. Upload a larger file from A to B and note the achieved bandwidth. Then download a larger file from B to A. (Hint: Use `tc` with queing discipline `tbf` for that)

(d) From now on, we consider the configuration of `Linux-Router` and `Host B` fixed. Essentially, we are playing DSL customer and want to optimize our local traffic shaping without having any influence on neither the routers of our ISP nor the servers on the Internet.

Perform a concurrent file upload and download. Which bandwidth does the download achieve, compared to the solo download from (c)? How can that be explained?

(e) Linux offers traffic priorization in order to minimize the impact that different traffic classes have on each other. Familiarize yourself with the explanation of HTB *(Hierarchical Token Bucket)* in the HOWTO. Try to implement a class-based traffic shaping that minimizes the impact of the concurrent upload on the download without interfering with the upload bandwidth too much.

For submission details please check the FAQ:
`http://www.net.t-labs.tu-berlin.de/teaching/ss09/RL_labcourse/`
Submit the following:

- Topology map
- All configuration inputs on routers, switches and loadgens (no trials, only the final ones)
- The outputs of all commands such as ping, tcpdump, ifconfig etc.
- The completed evaluation form (extra sheet)

Due Date: Friday, July 3th, 2009, 08:00 am