

CSIS0234B Computer and Communication Networks (Class B)

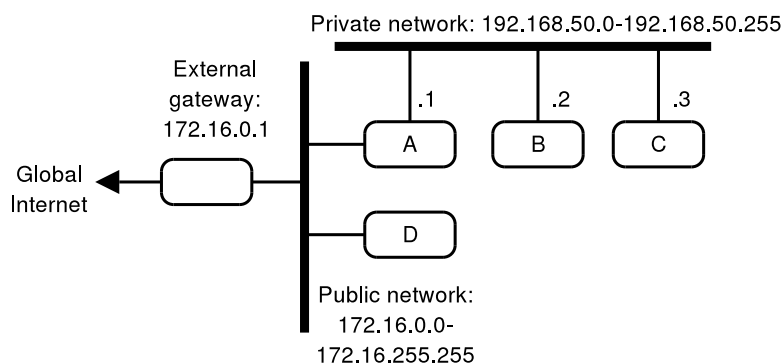
Tutorial 8

Setting up a NAT router

This week you will work in group of 3 or 4 students, to setup a computer as a firewall and a NAT router, thus allowing an Internet connection to be shared among computers.

Your tasks

You are allocated 4 computers. Three of them, *A*, *B* and *C*, form a private network. The fourth, *D*, is outside the private network, and is used to test what can or cannot be done from outside. *A* will act as the firewall and the NAT router. *B* provides a service, `ssh`, but only for the private network. *C* also provides `ssh`, but it would like to be accessible from both within and outside the private network. (The `ssh` server is already installed.) The network topology looks like this:



You will **write a shell script** at *A* to perform all the set up needed. For convenience, makes sure that whenever the shell script is executed, all previous settings will be cleared and changed to the desired state. When you have problem, you can look at the log at `/var/log/messages`, especially if you have rules with `LOG` targets.

Part 1: Basic NAT setup

1. Setup the network with the tool of Redhat (menu->setup->network): for *A*, use `eth0` to connect to the department with the address `172.16.9.y/16`, where `y` is the computer number; and use `eth1` for connecting to your hub/switch, with address `192.168.50.1/24`. It should use `172.16.0.1` as the default gateway. For *B* and *C*, use `eth0` to connect to the hub/switch, with address `192.168.50.2/24` and `192.168.50.3/24` respectively, and use *A* as the default gateway. Test connectivity using `ping`.
2. Create the shell script at *A*. Begin the script by restarting the `iptables` service and enabling forwarding, thus ensuring that `iptables` is in a known state. Then add commands which use `MASQUERADE` for all packets generated or forwarded by *A*. Check that you can `ping` to the external gateway and our nameserver `172.16.0.15`.
3. Check that *B* and *C* cannot surf the web and use `ssh`: the firewall of *A* blocks most such forwarding traffic (e.g., DNS queries). Now modify the script at *A* to clear the `FORWARD` chain, and check that *B* and *C* can communicate to the outside world normally.
4. Use `ethereal` on both *A* and *B* to determine whether a connection to outside have the source IP address, destination IP address, source port number and destination port number modified. (Hint: to avoid seeing many irrelevant packets, use a filter that only look at packets with the IP addresses of *A* and *B*.)

Part 2: Basic firewall setup

1. *B* wants to provide the `ssh` service to *A* and *C*, but don't want external computers like *D* to be able to access it. The first thing that *B* must do is to allow *A* and *C* to access the service. Modify the `INPUT` chain at *B* to accept packets **destinated** to its TCP port 22 (i.e., `ssh`). Make sure *A* and *C* can access its service.
2. Use *D* to access the `ssh` service of *B*. This is supposed not to work, since those computers are in their "private network". Explain what stops *D* from connecting to *B*.
3. Without a proper firewall, the above safety is not real. Try to modify the routing table of *D* to gain access to the `ssh` service of *B*. (Hint: force traffic toward *B* to go through *A*.)
4. To fix the problem, modify the script at *A* so that its `FORWARD` chain provides protection similar to the "medium-level" security that is shipped by Redhat: drop all incoming packets to the UDP and TCP port ranges 0–1023 (privileged ports, most services run there), TCP ranges 6000-6010 (X11 ports) and 7100 (X11 font server). Every dropped packet should also be logged. Once finished, make sure that *A* and *C* can still access the `ssh` service provided by *B*, while *D* won't be able to do that even with the "hack" in step (3).

Part 3: Dynamic NAT setup

1. *C* also want to provide `ssh`. Unlike *B*, it wants `ssh` to be accessible from the outside via the NAT router. Modify the script at *A* to perform DNAT, so that all traffic to TCP port 22 (`ssh`) will be forwarded to *C*. Also, relax the filter a bit so that TCP port 22 packets towards *C* are not dropped. Use the picture in the reading material to determine whether you should relax the `INPUT` or `FORWARD` chain of the filter table.
2. From *D*, use `ssh` to connect to *A*. Make sure that it gets a shell of *C*, not of *A*.
3. Try `ssh` to *A* from *A*. By reading the figure and tracing the chains used by *A* when processing the packet, explain why it doesn't work end up in *C*. Correct the problem by making some further modification to the script at *A*.