



Exercise sheet 4

Deadline: Thursday, 20 March, 12.00 noon

Write an extension to the linux firewall which checks any outgoing connection against a set of allowed connections and rejects any connection attempt which is not allowed. This extension should be written as a module. When this module is loaded, no connection is allowed apart from the ones to localhost (127.0.0.1) which must be always allowed, otherwise the computer will lock up quite quickly. Although not required for assessment, for debugging purpose it is a good idea to include a way of displaying the current set of active connections. You also should write a user program that reads in a configuration file as specified in the second exercise and adds all connections arising from the configuration to the set of allowed connections.

The structure `ConfigKernelEntry` in the include-file `firewall.h` specifies an allowed connection. A modification of the module `firewallExtension` is the suggested way of achieving this. In particular, you should modify the function `FirewallExtensionHook` and return `NF_ACCEPT` if the connection should be accepted and `NF_DROP` if the connection should be rejected.

You will also need a way of transferring the set of allowed connection into the kernel. For this, you should use the `/proc`-interface to transfer data into the kernel. To see how this communication works I provided one program as an example. This program, called `addRule.c`, runs in user space, and takes as input four arguments, namely

1. the name of the file in the `/proc`-directory used to communicate with the kernel;
2. the program name
3. the IP-address and
4. the port number.

This program writes this connection data to the specified file in the `/proc`-directory, where it can be read into the kernel. In particular, you should note the procedure `convertIPAddress`, which converts the IP-Address string to a format which is suitable for the kernel.

I have omitted the specification of the allowed user, because transforming it into a way that can be checked easily in the kernel is too difficult for this exercise.

The library provided for the previous exercise might be very helpful for this exercise, too.

You are allowed to form groups of two for this exercise. If you do so, each group member must submit identical code and also include into the submission the list of all group members.