Operating Systems with C/C++
Semester 1, 2013
Eike Ritter

The University of Birmingham
26 November, 2013

Assessed Exercise 4

Deadline: Friday 13 December, 4pm

The Task

The purpose of this exercise is to write an extension to the standard kernel firewall. In this exercise, the packets will be handled in the kernel, rather than in userspace, as in the previous exercise. The firewall should implement checks on outgoing DNS-queries. More precisely, the firewall maintains a blacklist of sites for which DNS-queries are rejected. It also maintains a list of sites which have been queried (including the ones which have been rejected).

Your program should work as follows:

- You should provide a user programs and a kernel module. The kernel module processes the packets and maintains the blacklist and the list of sites which have been queried. The user space program has commands firstly for displaying the list of DNS-requests and indicating whether they have been accepted or not, and secondly for setting the blacklist. A new blacklist overrides the old one. You should use a suitable file in the `proc`-directory for the communication between the user program and the kernel.

- To make marking easier, there should be two ways of calling the user space program. The first one is

  ```
  <program> L
  ```

  where `<program>` is the program name. This way of calling the client program outputs the list of requests and indicates whether they have been accepted or not. The second way of calling the program is

  ```
  <program> W <filename>
  ```

  where `<program>` is as above, and `<filename>` is the name of the file containing the blacklist. This file contains one line for each site which is to be blocked. This way of calling resets the list of blocked sites.

- Firewall modifications should require root privileges. Hence you should ensure that only root is able to modify the kernel firewall by ensuring that the file in the `proc`-directory is writeable only for root (This is true for the example file). You should abort with an error message if a non-root user tries to execute the user space program. Use the system call `geteuid` for this check. The root user has always a user ID of 0.

- You need to ensure that you handle concurrency correctly. In particular, as packets may arrive at any time, several instances of the procedures handling the packets may be executed at the same time.

You may assume that each packet for the outgoing port 53 is a DNS-packet. You obtain the packet data as follows: If `skb` is the pointer to the socket buffer passed to the packet handling function (`firewallExtensionHook` in the example code), `skb->data` and `skb->tail` are pointers to the beginning and the end of the packet data, respectively. The site which is queried is stored from byte 40 of the packet data as follows: for each component of a site (the string between dots), first the length of the component is stored, and then the component itself. If the length of the next component is 0, the end of the site has been reached. For a visualisation of the packet structure, use wireshark with the filter `udp.port == 53`.

Marking Scheme

Please use canvas for submitting your code. Please submit only the source files you have written yourself. We will compile and run your code on the virtual machine and mark it accordingly. Please in particular note that we will use the compiler option introduced in the lecture and will deduct 6 marks immediately if there is any compiler error or warning.

We will award marks as follows:

- 5 marks for correctly extending the firewall.
- 5 marks for correct handling of the blacklist.
- 5 marks for correct handling of the list of outgoing queries.
- 5 marks for the correct functioning of the program displaying the list and setting the blacklist.