# Lecture 18: Packet Filtering Firewalls (Linux)

## Lecture Notes on "Computer and Network Security"

by Avi Kak (kak@purdue.edu)

March 20, 2017

11:49pm

## Goals:

- Packet-filtering vs. proxy-server firewalls
- The four packet-filtering tables supported by iptables: **filter**, **nat**, **mangle**, and **raw**
- Creating and installing new firewall rules
- Structure of the **filter** table
- Connection tracking and extension modules
- Designing your own packet filtering firewall

## CONTENTS

# 18.1: FIREWALLS IN GENERAL

- Two primary types of firewalls are
  - packet filtering firewalls
  - proxy-server firewalls

  Sometimes both are employed to protect a network. A single computer may serve both roles.

- With a proxy-server based firewall, all network traffic in a host is routed through the proxy server. That allows the proxy server to exercise access control over the traffic in ways that will be explained in Lecture 19.

- Packet filtering firewalls, on the other hand, take advantage of the fact that direct support for TCP/IP is built into the kernels of all major operating systems now. When a kernel is monolithic, TCP/IP is usually internal to the kernel, meaning that it is executed in the same address space in which the kernel itself is executed (even when such a capability is made available to the kernel in the form of a module that is loaded at run time). [In addition to scheduling processes and threads, one of the main jobs of an OS is to serve as the interface between user programs, on the one hand, and the hardware (CPU, memory, disk, network interfaces, etc.), on the other. The core part of an OS is usually referred to as its kernel. Unless you are using highly specialized hardware, access by a user program to the hardware in a general-purpose computing platform must go through the kernel. By the same token, any new data made available by the hardware in such general-purpose machines is likely to be seen first by the kernel. Therefore, when a new data packet becomes available at a network interface, the kernel is in a position to immediately determine its fate — provided the kernel has the TCP/IP capability built into it. Just imagine how much slower it would be if a packet coming off a network interface had to be handed over by the kernel to a user-level process for its processing. Kernel-level packet filtering is particularly efficient in Linux because of the *monolithic* nature of the kernel. Linux is monolithic despite the fact that much of its capability these days comes in the form of *loadable kernel modules*. In general, a kernel is monolithic when its interaction with the hardware takes place in the same address space in which the kernel itself is being executed. (The "loadable kernel modules" of Linux that you can see with a command like `lsmod` are executed in the same address space as the kernel itself.) The opposite of a *monolithic kernel* is a *microkernel* in which the interaction with the hardware is delegated to different user-level processes (and, thus, is subject to address-space translations required for process execution). Recall that each process comes with its own address space that must be translated into actual memory addresses when the process is executed. For a very fascinating discussion on monolithic kernels vs. microkernels at the dawn of the Linux movement (in the early 90s), see `http://oreilly.com/catalog/opensources/book/appa.html`. This discussion involves Linus Torvalds, the prophet of Linux, and Andrew Tanenbaum, the high-priest of operating systems in general. Even though this discussion is now over 20 years old, much of what you'll find there remains relevant today.]

- In Linux, a packet filtering firewall is configured with the Iptables modules. For doing the same thing in a Windows machine, I believe the best you can do is to use the graphical interfaces provided through the Control Panel. It may also be possible to use the WFP APIs (Windows Filtering Platform) for embedding packet filtering in user-created applications, but I am not entirely certain about that — especially with regard to packet filtering in the more recent versions of the Windows platform.

- The `iptables` tool inserts and deletes rules from the kernel's packet filtering table. Ordinarily, these rules created by the `iptables` command would be lost on reboot. However, you can make the rules permanent with the commands `iptables-save` and `iptables-restore`. The other way is to put the commands required to set up your rules in an initialization script.

- Rusty Russell of the Netfilter Core Team is the author of `iptables`. He is also the author of `ipchains` that was incorporated in version 2.2 of the kernel and that was replaced by `iptables` in version 2.4.

- The latest packet filtering framework in Linux is known as `nftables`. Meant as a more modern replacement for `iptables`, `nftables` was merged into the Linux kernel mainline on January 19, 2014. `nftables` was developed to address the main shortcoming of `iptables`, which is that its packet filtering code is much too protocol specific (specific at the level of IPv4 vs. IPv6 vs. ARP, etc.). This results in code replication when firewall engines are created with `iptables`.

- Despite its many advantages over `iptables`, there has not yet been a wholesale switchover from `iptables` to `nftables` — probably because there do not yet exist tools capable of automatically