

# Packet Filtering - The first line of defense

PAULO CABIDO  
Universidade de Évora

---

The first packet filters were known as packet filter firewalls and were used to protect IP networks from attacks. As time passes the state of a network may change, networks grow, new and more complex configurations have to be applied, the need to connect to more and more neighbor networks with unknown trust status, etc, requiring packet filtering to be increasingly complex for a network to maintain the desired level of security. In this paper, my goal is to present the general properties, different approaches of packet filtering, and how they can help to secure networks.

Categorias e Descritores de Assunto: C.2.0 [**Computer-Communication Networks**]: General—Data communications; C.2.3 [**Computer Communication Networks**]: Network Operations - Network Monitoring

Termos Gerais: Security

Palavras-chave adicionais: network, security, firewall, packet, filter

---

## 1. INTRODUCTION

Nowadays Internet is part of our daily routines, we are in an era where laptops and netbooks are common and are entering the smartphone era, where soon enough, everyone of us will have a device connected to some network. It is so normal that people sometimes use expressions like "bring me the Internet" when what they really want to say is "bring me my laptop" or "bring me my broadband USB modem". Networks and Internet communications are growing, traffic is increasing, the IPv4 address pool is exhausting, and so are security related incidents. Network administrators are being challenged on a daily basis and are being forced to deal with security and performance.

Before I continue my paper, I have to clarify one general miss conception, firewalls protect networks from outsiders who attempt to beak-in, that is the general idea, but, and it is not less important, they also protect the network against internal threats. For example, worm and virus spread.

The link between networks and security are firewalls, sophisticated systems that provide the much wanted and needed protection. To start with network security, packet filtering firewalls are the way to go. One of the most basic type of firewalls is the Packet Filtering Firewall and even a state of the art firewall uses packet filtering before they inspect the content of a connection. Routers and firewalls have an important role in network security, they can filter unwanted or unauthorized packets. Typically a set of rules determines which packets should be accepted and which should not.

My goal with this paper is to explain what packet filtering is, why one should use it, what it consists of, the different approaches and to show how different operating

systems deal with packet filtering.

## 2. PACKET FILTERING

### 2.1 Understanding packets

Understanding packets is the first step towards understanding packet filtering [Zwicky 1995].

A packet is the basic unit of data transfer in a networked environment. For data to go from A to B, it has to be separated in individual chunks of data and sent to their destination. Once they achieve their destination, they cease to exist.

Packets are created and changed as they pass through the network layers (OSI Model). At each layer, a packet has two parts, the header and the body. The header contains protocol information that is relevant to that layer and the body contains data. Each layer applies its own header to the data, so at each layer, the packet contains all of the information passed from the higher layer. This process is known as encapsulation.

The data part of the packet is important, but for most of the existing approaches and packet filtering methods, that I will explain later in this paper, the key is the header of the several existing layers that contains key elements for each layer.

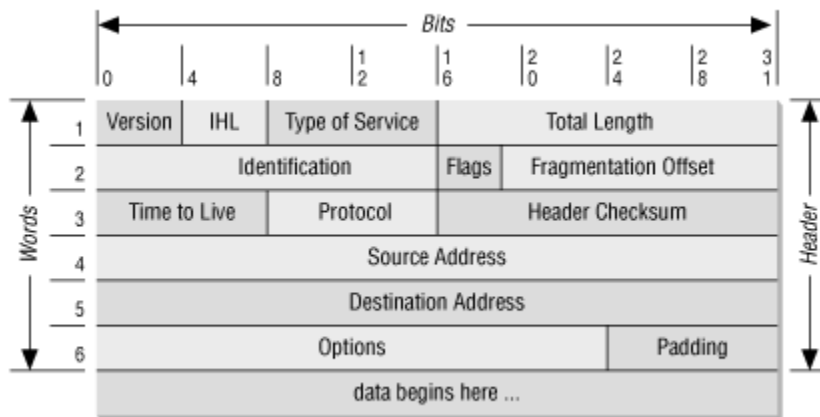


Fig. 1: Example: IP header and body.

### 2.2 What is Packet Filtering?

A routing device checks a packet's header for the destination address and decides whether or not the packet has to be routed to another network device or should stay on that interface. This is the basic routing principle.

When packet filtering is added to routing devices, another level of packet analysis is done. Each packet will be put through the normal routing analysis and when determined that it has to be processed, the routing device applies the filter rules.

Filter rules normally reflect security policies, which services are allowed, where they are allowed, which are not, which type of packets can reach a target device, which should be dropped, etc.

So, packet filtering is the process of passing or blocking data packets, based on a set of user-provided rules, as they pass through a network interface.

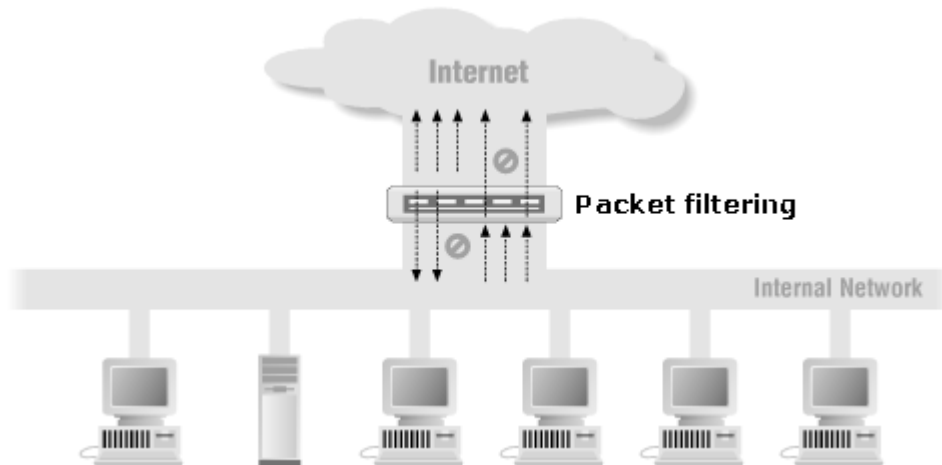


Fig. 2: Packet filtering example.

### 2.3 Why use packet filtering?

Packet filtering is usually the first line of defense against network security threats. These threats are constantly evolving and network security strategy must evolve with them, requiring security measures to be flexible and powerful. Packet filtering is both, flexible and powerful. Because it is also a common and inexpensive security method, most devices or operating systems provide such functionalities. Packet filtering is frequently used in combination with other systems. For example, it is used to provide a critical handle to intrusion detection systems (IDS) or intrusion prevention systems (IPS).

### 2.4 Packet filtering, where to do it?

My answer to the question is everywhere you can! Small networks are easy to secure and usually have one router, a few desktops and a couple of servers. The problem is the big networks, that have many routers, switches, computers (users), have wired and wireless connections, several networks within the main network, vlans, complex routing configurations, etc. For each router in the network, it should be clear what types of packets should legitimately pass through it. This will not only protect the edges of the network but will create an extra internal protection with redundancy of some filtering rules. In this case redundancy is good and may save the network one day. As Murphy's law states "If anything can go wrong, it will.", and as we all know, computers fail, anti-virus software fail and firewalls also fail.

## 2.5 Types of packet filtering

There are many approaches to packet filtering and further on in this paper I will enumerate and explain them, but generally packet filtering can be divided into two parts: Stateless packet filtering [Pierluigi Rolando and Risso 2009] and Stateful packet filtering.

*2.5.1 Stateless packet filtering.* This type of filtering is when the information about the passing packets is not remembered by the firewall or device filtering the packets. No connection tracking is made. All the allowed/denied decisions are taken on packet by packet basis.

*2.5.2 Stateful packet filtering.* Some devices or firewalls perform packet filtering and remember the information about the previously passed packets, in other words, keep track of the state of the network connections that they handle. This type of packet filtering is called Stateful packet filtering and is also known as Dynamic packet filtering.

## 2.6 State of the art

Dynamic Packet Filter (DPF) approaches are currently the state of the art in the packet filtering world.

DPF provides the most wanted flexibility of packet filters and the speed of hand-crafted demultiplexing routines. DPF filters are the fastest packet filters available. DPF achieves high performance by using a carefully-designed declarative packet filter language that is aggressively optimized using dynamic code generation. The declarative packet filter language is used by protocols to describe the message headers that they are looking for.

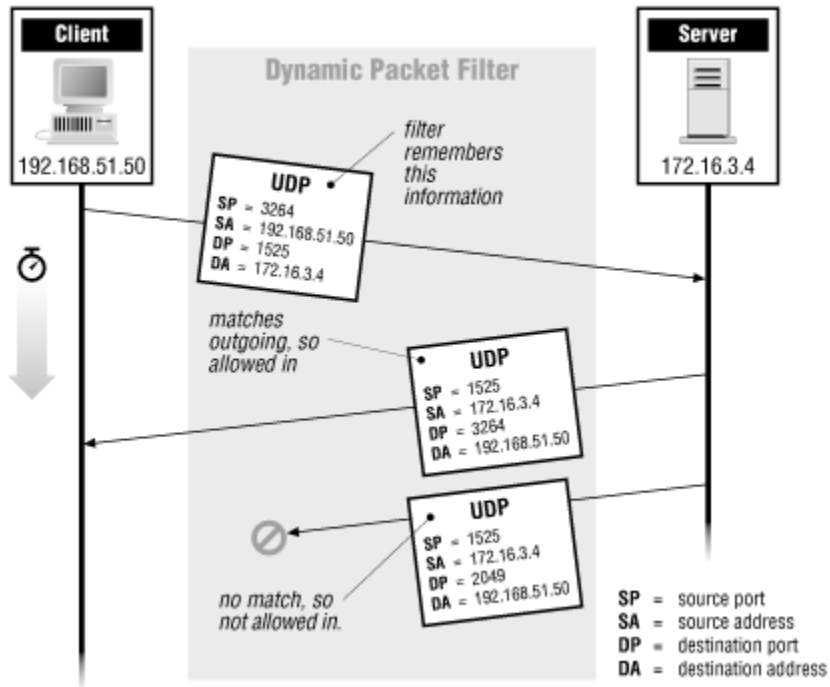


Fig. 3: Dynamic packet filtering general principle.

«A DPF packet filter is composed by a sequence of boolean comparisons (or atoms) linked by conjunctions. Logically, a filter's atoms are checked in order after packet reception (our declarative language avoids most side-effects, allowing re-ordering for efficiency). If all atoms are true, then the filter accepts and the packet is delivered to the associated application.» [Engler and Kaashoek ]

```
(
# Check Ethernet header
(12:16 == 0x8) && # IP datagram?
# Skip over ether header (14 bytes)
(SHIFT(6 + 6 + 2)) &&

# Check IP header
(9:8 == 6) && # Check protocol : TCP is 6
(12:32 == 0xc00c4501) && # Check IP src addr (192.12.69.1)
# Skip past IP header (assume fixed sized; 20 bytes)
(SHIFT(20)) &&

# TCP header
(0:16 == 1234) && # Check source port (2 bytes)
(2:16 == 4321) # Check destination port (2 bytes)
)
```

Fig. 4: Example of DPF filter that recognizes TCP/IP packets.

### 3. APPROACHES TO PACKET FILTERING

#### 3.1 CMU/Stanford Packet Filter

CMU/Stanford Packet Filter (CSPF) is the first user-level packet filter, which appeared on Xerox Alto (Unix kernel) in 1980. For the existing computers in that era, CSPF performed efficiently and provided flexibility in the environment, with an interpreter based filter mechanism. The specified filter language is stack based, and operates on binary expressions as well as boolean operators. It uses a tree model to configure its filter engine. It believes that user-level demultiplexing will incur more context switches and inter-processes. In order to provide both flexibility and efficiency, CSPF specifies filter in user-level and processes in kernel-resident using a specific packet field as a key.

The problem of this approach is the bottleneck in operations that has a data structure of 16-bit words array.

#### 3.2 The BSD Packet Filter

The BSD Packet Filter (BPF) is a more efficient interpreter than CSPF. It is an interpreter based on a register-based filter mechanism. BPF with register-based and assembly-like language can access more instructions, multiple registers, one input data and a scratch memory. It uses a computationally equivalent directed acyclic control flow graph, in order to avoid redundant computation. It learns packet parse states in the graph, so it reduces some paths and comparisons. It also uses a boolean expression tree. The major performance improvements of BPF comparing to CSPF, are due to architectural improvements (registered based RISC CPU).

#### 3.3 The Mach Packet Filter

The Mach Packet Filter [Yang and Chang ] (MPF) is a extension to the filtering language of BPF. The extension enables the language to support end-point protocol processes in the Mach operating system. It solves two major problems. It uses the `ret_match_imm` instruction to collapse multiple filter programs into one, improving performance of scalability and uses per-filter state to persist across the arrivals in order to dispatch fragmented packets. It is also designed to support context-dependent demultiplexing which is needed in large fragmented messages.

#### 3.4 The BSD Packet Filter+

The BSD Packet Filter+ [Andrew Beigel and Graham ] (BFP+) is commonly regarded as one of the most modern CFG-based approaches [Pierluigi Rolando and Risso ]. It exploits data-flow algorithms for generalized optimization among filters, eliminates redundant predicates, allows to match header fields against one another, enables arithmetic operations on header works before matching and can generate native code using just- in-time (JIT) compilation. The filter specifications are written in a high-level predicate language (`libpcap`). But it has some downsides like missing juxtaposition, it doesn't have per-filter state and cannot implement loops.

### 3.5 Dynamic Packet Filters

As stated before, Dynamic Packet Filter (DPF) approaches are currently the state of the art in the packet filtering world.

DPF provides the most wanted flexibility of packet filters and the speed of hand-crafted demultiplexing routines. DPF filters are the fastest packet filters available. DPF achieves high performance by using a carefully-designed declarative packet filter language that is aggressively optimized using dynamic code generation. The declarative packet filter language is used by protocols to describe the message headers that they are looking for.

## 4. OPERATING SYSTEMS: WINDOWS FILTERING PLATFORM AND LINUX PACKET FILTERING

I'm only considering these two operating systems because they are the most used worldwide. Mac OS is Unix based and recently got the Unix certification (Unix 03 certification), so I will include it in the Linux group, as it uses iptables.

### 4.1 Windows Filtering Platform

Microsoft changed the way Windows handles packet filtering in Windows Vista. It introduced a new platform, Windows Filtering Platform [Microsoft ] (WFP), that is not a firewall, it is a set of system services and user-mode and kernel-mode an API, that allows applications to have access to packet processing and filtering pipeline of the newly created network stack. Therefore allowing developers to create firewalls or monitoring network traffic.

Its goal is to provide an inbuilt filtering engine to applications, so that developers will not need to write any custom engine, they just need to provide the custom logic for the engine to use.

**4.1.1 WFP Components.** Because WFP is intended to be used by firewalls and other packet-processing software it is composed by several components:

- Win32 API, contains the WFP filtering APIs. It allows third-party applications to use the WFP filtering APIs to create filters within the Base Filtering Engine;
- Base filtering engine is a user-mode service that implements the filter requests that user-mode filtering applications made by adding them to the kernel-mode filtering engine;
- Kernel Mode Filter engine is a component that stores the filters that third-party applications create. The stored filters interact with the various filtering layers of the TCP/IP stack and also interact with the callout drivers;
- Callout drivers are used when initial filtering of packets is not enough to determine whether they should be dropped, permitted or modified. They are used for further deeper inspection of packets.

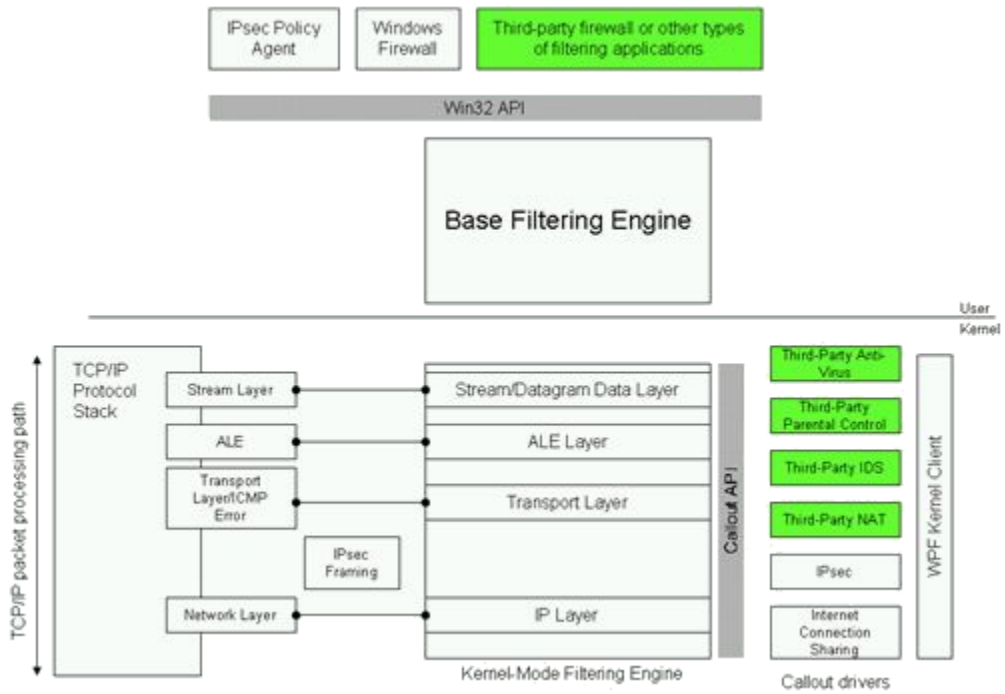


Fig. 5: WFP architecture and its extensibility for third-party applications, services, and drivers.

#### 4.2 Linux Packet Filtering

Under Linux, packet filtering [lin 2005] is built into the kernel (it can be built right in or as a kernel module) implemented as different Netfilter modules. These modules, Netfilter, together are a framework that provides hook handling within the Linux kernel for intercepting and manipulating network packets.

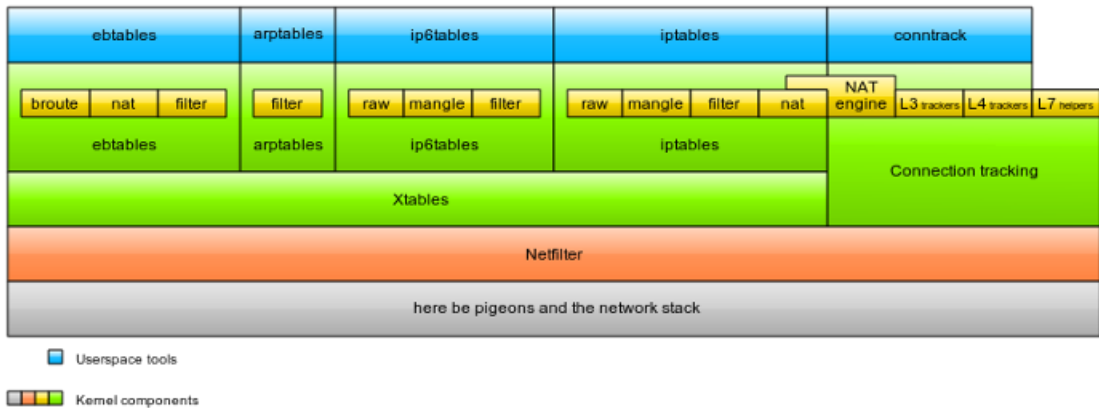


Fig. 6: Netfilter components.

Allot can be done but the general principle of looking at the packet headers and deciding what to do with the packet is still there.

The first generation of Linux (1.1 series) based the Packet Filter on BSDs ipfw that was ported by Alan Cox in 1994. Linux 2.2 in 1998 brought a reworked kernel and introduced the user space tool ipchains. Linux 2.4 arrived in 1999 and again brought a new kernel (another kernel rewrite) and it also brought the fourth generation tool, iptables that is still used in the present.

4.2.1 *iptables*. In Linux packet filtering resumes to iptables. The iptables user space tools insert and delete rules from the kernel's packet filtering table. Although it can look like an "essential binary" it is more like a service.

Xtables allow the system administrator to define tables containing chains of rules for the treatment of packets. A packet is put up against all the rules in the table but it is possible that a chain/rule can cause a jump to another chain/rule. There are five predefined chains with no rules and the system administrator can create new chains fitting his needs. The five predefined chains are:

- PREROUTING, packets will enter this chain before a routing decision is made;
- INPUT, when a packet is going to be locally delivered;
- FORWARD, all packets that have been routed and were not for local delivery will go through this chain;
- OUTPUT, packets sent from the machine itself will be visiting this chain;
- POSTROUTING, when a routing decision has been made. Packets enter this chain just before handling them off to the hardware.

Each chain can contain rules that have specifications on which packets to match and can happen on any layer of the OSI model.

The iptables tools are quite simple to use, although rules can be very complex and difficult to test.

## 5. CONCLUSION

Packet filtering mechanisms and applications have been around for a long time, for more than thirty years now and they will continue to play an important role in the world of security.

They will continue to evolve for sure, new approaches will come up with better methods and performances. Packet filtering is not used by its own, long goes the time when firewalls were just packet filter firewalls. Intrusion Detection Systems and Intrusion Prevention Systems are becoming very popular and are also security methods to be taken into account. However no matter what analysis is done, the kind of inspection that is made to the network traffic, it will surely all begin with packet filtering.

Going back to the real world, many or all of the state of the art packet filter approaches are used in commercial packet filters with enormous licensing costs. With a worldwide topic of the day being "Crisis" for some time now, companies tend to contain their budgets and the regular network administrators have to work with what is available: the operating systems packet filtering capabilities. In what this chapter is concerned and faced with a choice decision, I do not hesitate between

Linux and Windows, I definitely choose Linux. It is free, I can modify it, iptables is very easy to use and it is easy to set up a firewall or filter traffic on a Linux based router device.

#### REFERÊNCIAS

2005. Linux packet filtering and iptables.
- ANDREW BEGEL, S. M. AND GRAHAM, S. L. Bpf+: Exploiting global data-flow optimization in a generalized packet filter architecture.
- ENGLER, D. R. AND KAASHOEK, M. F. Dpf: Fast, flexible message demultiplexing using dynamic code generation.
- MICROSOFT. Windows filtering platform. <http://www.microsoft.com/whdc/device/network/wfp.msp>.
- PIERLUIGI ROLANDO, R. S. AND RISSO, F. Fsa-based packet filters.
- PIERLUIGI ROLANDO, R. S. AND RISSO, F. 2009. Spaf: Stateless fsa-based packet filters.
- YANG, M.-G. AND CHANG, C.-C. Efficient implementation of the packet filter for network traffic monitor.
- ZWICKY, D. B. C. . E. D. 1995. *Building Internet Firewalls*. O'Reilly.