PktFilter
A Win32 service to control the IPv4 filtering driver of Windows 2000/XP/Server 2003
http://sourceforge.net/projects/pktfilter/

Jean-Baptiste Marchand
Jean-Baptiste.Marchand@hsc.fr

Contents

1 Introduction

PktFilter is a free software (distributed under the BSD license) that configures the IPv4 filtering driver found in Windows 2000/XP/Server 2003.

Normally, this driver is configured[1] via the Routing and Remote Access (RRAS) service of Windows. However, the Packet Filtering API[2]documented in MSDN enable third-party software to configure it.

The advantages of PktFilter over RRAS are:

- PktFilter is a small service dedicated to IP filtering whereas RRAS handles many things in addition to IP filtering, like IP routing.
- Filtering rules of PktFilter are specified in a text file, following a well-know syntax, used by IP Filter. RRAS filtering rules can be specified either via a graphical interface of the RRAS MMC plugin or with `netsh`, using an obscure syntax.
- PktFilter reports filtering statistics, whereas RRAS only reports limited informations with `netsh`.
- PktFilter supports logging of blocked packets.

The other possibilities for IP Filtering available in standard under Windows 2000/XP/Server 2003 are:

- Using IPsec for IP Filtering. This has been described in many articles but this is not a good idea, as some traffic is not filtered by IPsec (Kerberos traffic on TCP port 88 and IKE traffic on UDP port 500). Filters can then be bypassed using one of these two ports as source.
- Using Internet Connection Firewall (ICF[3]). This is only available in Windows XP and Windows Server 2003. The IP filtering driver used by ICF is supposed to be stateful (i.e, creates states when communications are established and then uses these states to validate traffic). However, it can only filter **incoming** traffic. Moreover, Microsoft has no plan to release an API similar to the Packet Filtering API, to control the kernel-mode driver used by ICF.

Important:

PktFilter only configures the IPv4 filtering driver present in recent Windows sytems. PktFilter does **not** implement IP filtering operations.
Thus, PktFilter can not be extended to do stateful IP filtering, traffic redirection or network address translation. This kind of functionnality can only be implemented in a NDIS intermediary driver.

2 Installation

PktFilter is composed of two programs:

[1] http://www.microsoft.com/windows2000/techinfo/reskit/en/Intwork/inae_ips_wxnv.htm
[2] http://msdn.microsoft.com/library/en-us/rras/ipflt_7oo5.asp
[3] http://www.microsoft.com/windowsxp/pro/techinfo/planning/firewall/

- `pktftlsrv` is the Win32 service that configures the IP Filtering driver, using the Packet Filtering API.
- `pktctl` is a command-line utility that manages filters, communicating with the `pktfltsrv` service through a named pipe.

To install, please follow the following steps:

- Extract the content of the current PktFilter archive in a directory, for example under `%systemroot%\Program Files\`.
- In a command terminal, change to the directory `pktfltsrv` under the directory you just extracted the archive and install the service with the `-i` option, followed by the absolute path to the file containing rules (a default `rules.txt` file exists under the `pktctl` directory) and by the absolute path to the log file:

```
C:\Program Files\PktFilter\pktfltsrv> pktfltsrv -i \
     "C:\Program Files\PktFilter\pktctl\rules.txt" \
      C:\Program Files\PktFilter\pktctl\PktFilter.log"
```

- Edit the rules file to write your own rules. Please note that in the default `rules.txt`, all traffic is **blocked** by default.
- Start the `pktfilter` service, with the following command:

```
C:\> net start pktfilter
```

- In the Services Manager, change the Startup Type of PktFilter (Stateless Packet Filtering) to Automatic if you want PktFilter to start automatically at system startup (recommended):

```
C:\> services.msc
```

3 Specifying filters: the rules.txt file

3.1 Syntax chosen for PktFilter rules

The syntax used to specify filtering rules is (mostly) a subset of the grammar used when writing IP Filter[4] rules.

3.2 Writing rules

In this section, we describe the different elements that must appear in the rules file in this order.

Before starting writing rules, you have to know the mapping between your network adapters and the interface naming convention used by PktFilter.

To obtain this mapping, use the `-I` option of `pktctl`, as follows:

[4] http://www.ipfilter.org/

```
C:\> pktctl -I
eth0: (MS LoopBack Driver): 172.16.42.42
eth1: (3Com EtherLink PCI (Microsoft's Packet Scheduler) ): 192.70.106.142
```

In this example, PktFilter identifies the only physical adapter with the `eth1` identifier.

3.2.1 Global options

Rules always start with global options, i.e filtering options that are global to a given interface.
Currently, only one global option is allowed:

```
option small_frags on eth0
```

This option specify to refuse fragmented packets with fragments too small.

By default, it seems that all fragments with a size lower than 16 bytes are considered as small. This may be changed modifying the FragmentThreshold value, under the Parameters registry key of the IpFilterDriver driver but quick tests did not determine easily what this registry value does exactly.

It is **recommended** to activate this option.

3.2.2 Default rules

Two default rules must be specified, for input and output traffic. Theses rules specify one of the two possible filtering behavior:

- accept all traffic by default (`pass` keyword)
- block all traffic by default (`block` keyword)

Usually, the most secure technique for IP filtering is to block all traffic and then explicitely allow some kind of traffic. In PktFilter, you can achieve this with the following default rules applied to the `eth0` interface:

```
block in on eth0 all
block out on eth0 all
```

3.2.3 Filtering rules

The filtering rules specified after the two default rules can only be rules that **modify** the default behavior. For example, if you specified a default policy that drops all incoming packets, input rules can only start with the `pass` keyword.
Filtering rules are composed of the following ordered keywords:

- Filtering action:
  - `pass`, to allow traffic
  - `block`, to block traffic
- Direction
  - `in`, for incoming traffic
  - `out`, for outgoing traffic

- Interface
  - `on ifx`, where `if` is the type of your network interface (for example, `eth` for Ethernet adapters) and `x` is the index of the interface. Use option `-I` of `pktctl` to find the index of your interfaces.
- Protocol
  - `proto`, followed by
    - `tcp`, for TCP protocol
    - `udp`, for UDP protocol
    - `icmp`, for ICMP protocol
    - `number`, where number is the integer associated to a protocol encapsulated in IPv4
  - empty, to specify any protocol
- Source address
  - `from addr`, where `addr` is the IPv4 address of a host
  - `from subnet/mask`, where `subnet` is the address of a subnet and `mask` is the corresponding subnet mask
- Source port (TCP and UDP)
  - `port comparator decnumber`, where `decnumber` is the source port number and `comparator`, one of the following:
    - `=`
    - `>=`
    - `>`
    - `<=`
    - `<`
  - `port low_port >< high_port`, where `low_port` is the immediately lower port of the interval and `high_port` is the immediately higher port of the interval.
- Destination address: similar to Source address, with the `to` keyword
- Destination Port (TCP and UDP): same as Source port
- Type and code (ICMP)
  - `icmp-type type`, optionally followed by `code icmp-code`, where:
    - `type` is the type of an ICMP message, specified either by name (see grammar below) or numerically
    - `icmp-code` is the code of an ICMP message, specified either by name or numerically
- Established TCP connection
  - `established`
    - Specify to refuse TCP segments with the SYN flag and without the ACK flag set. This is useful to allow only answers to connection requests coming from the inside and block connection requests coming from the outside.

3.2.4 Example

```
# drop packets composed of small fragments
option small_frags on eth0

# default behavior = deny everything
block in on eth0 all
block out on eth0 all

# allow DNS resolution to our nameserver
pass out on eth0 proto udp from 192.168.1.1 port > 1023 to 192.168.1.254 port = 53
pass in on eth0 proto udp from 192.168.1.254 port = 53 to 192.168.1.1 port > 1023

# allow inbound ICMP traffic (ping)
pass in on eth0 proto icmp from any to 192.168.1.1 icmp-type echo
pass out on eth0 proto icmp from 192.168.1.1 type echo-rep to any

# allow RDP (Terminal Services) administration from our administration subnet
pass in on eth0 proto tcp from 10.42.42.0/24 port > 1024 to 192.168.1.1 port = 3389
pass out on eth0 proto tcp from 192.168.1.1 port = 3389 to 10.42.42.0/24 port > 1024 \
established
```

3.2.5 Reference: grammar of filtering rules

The filtering rules can be described using the following grammar in BNF:

```
filter-rule = global-options | normal-rule
global-options = "option" global_option iface
global-option = "small_frags"
normal-rule = action [in-out] iface [proto_spec] ip [proto-options]
action = "pass" | "block"
in-out = "in" | "out"
iface = "on" ifname digit
ifname = "eth" | "ppp" | "sl" | "lo" | "tr" | "fd"
proto_spec = "proto" [proto]
proto = "tcp" | "udp" | "icmp" | "any" | ip_proto
ip_proto = decnumber
decnumber = digit [decnumber]
ip = "all" | "from" ip-addr [port-comp | port-range] "to" ip-addr
     [port-comp | port-range]
ip-addr = "any" | ip-dotted-addr [ip-mask]
ip-dotted-addr = host-num "." host-num "." host-num "." host-num
host-num = digit [digit [digit]]
ip-mask = "/" ip-addr | decnumber
port-comp = "port" comparator decnumber
comparator = ">" | ">=" | "<" | "<=" | "="
port-range = "port" decnumber "><" decnumber
proto-options = "icmp-type" icmp-type ["code" icmp-code] | "established"
icmp-type = "echorep" | "unreach" | "squench" | "redir" | "echo" | "router_adv"
```

```
| "router_sol" | "timex" | "paramprob" | "timest" | "timestrep" | "inforeq" |
"inforep" | "maskreq" | "maskrep"
icmp-code = decnumber
```

4 Controlling PktFilter: the pktctl program

`pktctl` is the control program of PktFilter. It communicates with the `pktfltsrv` service through a named pipe.
`pktctl` can be used in two ways:

- with command-line switches
- in interactive mode (when launched with the `-i` option)

Most of the command-line switches have their counterpart commands in interactive mode. This section lists all the possible commands of `pktctl`, both in command-line mode and interactive mode.
Interactive mode is invoked with the `-i` command-line option. The `pktctl>` prompt will wait for interactive commands (`help` will list possible interactive commands).
The following actions can be achieved with `pktctl`:

- add rules (loading a file or entering a rule manually)
- list current rules
- delete rules
- get statistics

4.1 Adding rules

Load a rules file:

- `C:\> pktctl -f rules.txt`
- `pktctl> source rules.txt`

Load a rules file, flushing all filters on **all** interfaces before:

- `C:\> pktctl -F rules.txt`
- `pktctl> reload rules.txt`

Add manually a rule on a given interface, with the `-a` option:

- `C:\> pktctl -a "pass in on eth0 from 10.0.0.42 to any"`
- `netsh> pass in on eth0 proto udp from 10.0.0.42 to any`

4.2 Listing rules

List the current rules on a given interface, with the `-l` option:

- `C:\> pktcl -l eth0`
- `pktcl> list on eth0`

List the current rules with the rules numbers on a given interface, with the `-L` option:

- `C:\> pktcl -L eth0`
- `pktcl> List on eth0`

This option is useful when a rule must be deleted, given its rule number (see below). Input filtering rules are numbered from 1 to 127 and output rules from 128 to 255.

4.3 Deleting rules

Delete a rule on a given interface, giving the number associated to the rule, as reported by the 'List' command, with the '-d' option:

```
C:\> pktctl -L eth0
option small_frags on eth0
block in on eth0 all
block out on eth0 all
rule 1: pass in on eth0 proto tcp from 10.0.0.1 port = 3128 to 10.0.0.42
 port > 1023 established
rule 2: pass in on eth0 proto udp from any to any
rule 128: pass out on eth0 proto tcp from 10.0.0.42 port > 1023 to 10.0.0.1
 port = 3128
```

You can then delete the first input filtering rule with:

```
C:\> pktctl -d 2 eth0
```

Delete all rules on a given interface, with the '-Fa' option:

- `C:\> pktctl -Fa eth0`
- `pktctl> flush on eth0`

Delete all rules on all interface:

- `C:\> pktctl -Fa all` or simpler, `C:\> pktctl -Fa`
- `pktctl> flush on all`

4.4 Getting statistics

Get brief statistics for a given interface, with the '-s' option:

- `C:\> pktctl -s eth0`
- `pktctl> stats on eth0`

Get detailed statistics for a given interface, with the '-S' option:

- `C:\> pktctl -S eth0`
- `pktctl> Stats on eth0`

4.5 Errors

`pktctl` fails if the named pipe used for communication with `pktfltsrv` can not be opened. An error message is displayed in that case, informing that the `pktfltsrv` service is probably not running:

```
F:\Program Files\PktFilter\pktctl>pktctl -l eth0
error: unable to connect to \\.\pipe\PktFltPipe
Stateless Packet Filtering service is probably not running
```

5 The PktFilter service: pktfltsrv

5.1 Usage

`pktfltsrv` supports two command-line options:

- `-i "path_to_rules_file" "path_to_log_file"`:
  installs the service, where `"path_to_rules_file"` is the absolute path to the rules file and `"path_to_log_file"` the path to the logging file.
- `-u`: uninstalls the service.

5.2 Errors

`pktfltsrv` uses the standard mechanism of Windows for logging errors. In case of problems, an event is written in either the System or Application event log.

The most common errors are:

- A syntax error in the rules file. In that case, an error event is written in the System log.
- A problem to read the rules file. In that case, an error event is written in the Application log.
- The RRAS service and PktFilter may conflict, as RRAS can manage its own filtering rules. In that case, a warning event is written in the System log.

6 Logging

6.1 Introduction

Starting with PktFilter 0.06, PktFilter supports logging of blocked packets. Details about the blocked packets are written in a text file, specified at the installation of the PktFilter service.

The syntax of the log file is very similar to the one used by the IP Filter's ipmon daemon and basically contains:

- a timestamp
- network interface on which the packet was blocked
- rule set number (always `@0`) and rule number

- `b` for blocked packets, `b-frag` for blocked fragments
- source IPv4 address, followed, after a comma, by the UDP or TCP source port
- destination IPv4 address, followed, after a comma, by the UDP or TCP destination port
- protocol encapsulated in the IPv4 datagram (`tcp`, `udp`, `icmp` or any IP protocol number)
- IPv4 header length
- IPv4 payload length
- TCP flags (if applicable) or ICMP type and code (if applicable)
- direction (`IN` or `OUT`), only available when the default policy is `pass`

6.2 Configuration

6.2.1 Logging file

After installation, the logging file can be changed in the following registry value:

- Key: `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\PktFilter\`
- Value: `LogFile`

The service has to be restarted to use the new logging file.

6.2.2 Logging buffer size

The logging mechanism uses a logging buffer with a limited size. By default, the buffer size is 4096 bytes.

If this size is too limited, the following message will appear in the logging file:

```
# %u packets logged, %u packets lost, used buffer size was %u, total buffer size is %u
```

To avoid this situation, you can change the buffer size, adding the following registry value:

- Key: `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\PktFilter\`
- Value: `LogBufferSize`
- Content: size (in bytes) of the logging buffer

After this change, the service must be restarted.

6.3 Examples

The following line corresponds to a TCP segment with the SYN flag set, sent from the TCP port 2709 with 192.70.106.76 as IPv4 source address to the TCP port 3189 of the host with 192.70.106.142:

```
12/04/2002 16:31:29.254 eth0 @0:1 b 192.70.106.76,2709 -> 192.70.106.142,3189 \
PR tcp len 20 40 -S IN
```

The following line corresponds to a UDP datagram, sent from UDP port 137 to UDP port 137:

```
12/24/2002 14:02:59.611 eth0 @0:0 b 192.70.106.143,137 -> 192.70.106.159,137 \
PR udp len 20 78
```

6.4 Limitations

Logging support has the following limitations (note that these limitations come from the Packet Filtering API and the Windows IPv4 filtering driver and not from PktFilter itself):

- Only blocked packets can be logged
- Directions of blocked packets (inbound or outbound) is not available, except if the default policy allows packets

7 Limitations - Known bugs

If you decide to use PktFilter on production systems, you must be aware of the following limitations:

- The biggest limitation of the IPv4 filtering driver is that you can not mix rules. This is a design limitation, because of the default policy. Thus, it is not possible to do simple things, such as allowing most of the IPv4 address space except a specifig range.
- IP filtering is not active until the PktFilter service starts. Worse, if the service is launched at startup, the service blocks on the PfCreateInterface() function for two minutes. So, in the worst case, the system is not protected for more than two minutes.
- IP filtering is not active for local communications (communications using the loopback address, i.e `127.0.0.1`. As a consequence, no special rules are necessary to allow this kind of communications.
- Finally, remember that the IP filtering is only active when the PktFilter service is running. If, for any reason, the service crashes, IP filtering will no longer be active.

8 Support

PktFilter has a mailing-list that can be used to discuss PktFilter issues. See http://lists.sourceforge.net/lists/listinfo/pktfilter-users for more information.

9 TODO list

- Implement log file rotation
- Add the possibility to globally enable or disable logging. Logging could also be configured on a per-rule basis but that would be possible only for blocking rules, which implies that the default action would be pass, which is not recommended...
- Allow the service to start, even if network interface(s) is/are not yet available at startup. This would require a mechanism to detect when interfaces become available.

10 History

- 2003/02/12: version 0.06-beta2
  - Logging support
    - Supports logging of blocked packets
  - Some improvements and many bugfixes.
    - By default, the `pktfltsrv` service is configured as manual. After installation, you must explicitly configure it as automatic, if PktFilter suits your needs.
    - Address change on network interface will no longer disrupt IP filtering, as it was the case before. (The PfBindInterfaceToIndex() API is now used, instead of PfBindInterfaceToIPAddress())
    - Netmasks given in CIDR format (`/xx`) are now correctly supported.
    - Fix for the parsing of numeric icmp-type.
- 2002/07/08: version 0.05a
  - Bugfix in rules parser (some keywords were not evaluated when the `all` keyword was used). Thanks to Yann Bongiovanni for bug report and fix.
- 2002/05/31: version 0.05
  - Implement `-d` option. Thanks to gj for reporting this.
  - Add `-L` option. This allows to dump active rules and redirect the output to a file to save them.
  - Bugfixes in rules parser.
- 2002/02/18: version 0.04
  - Complete rewrite of the documentation, now formatted with LaTeX in a PDF file.
  - Bugfix in the rules parser, that did not take in account the `established` keyword in rules.

  - Removal of two of the three global options previously available, as they are not enough documented to be sure about what they do.
  - Improvement of statistics reporting
    * Rules are no longer displayed in the brief statistics reporting mode (option `-s` or `stats` keyword)
    * A new option, `-S` and counterpart keyword, `Stats` are equivalent to the old `-s` option and counterpart keyword `stats`.
    * The information displayed about dropped fragments in previous versions was incorrect and displayed even when the `small-frags` global option was not active. It is now correctly handled.
    * Incoming TCP SYN segment are now displayed along input filtering statistics.
  - Change of the rules listing mode
    * Rules numbers are no longer displayed in the brief rules listing mode (option `-l` or `list` keyword).
    * A new option, `-L` and counterpart keyword, `List` are equivalent to the old `-l` option and counterpart `list`.

- 2001/10/02: version 0.03
  - Display netmask when printing rules
- 2001/09/08: version 0.02
  - Better documentation
  - Small bug fixes
- 2001/05/30: version 0.01 (first public version)

11 Miscellanous

Greetings to

- Ghislaine Labouret: ideas, coding and debugging
- Hervé Schauer Consultants: support