WIPFW

WIPFW documentation (for v0.2.8)

Synopsis

```
ipfw [-q] add [number] rule-body
ipfw [-adeftN] {list | show} [number ...]
ipfw [-f | -q] flush
ipfw [-q] {zero | resetlog | delete} [number ...]
```

```
ipfw [-Nq] pathname
```

Description

An ipfw configuration, or ruleset, is made of a list of rules numbered from 1 to 65535. Packets are passed to ipfw in a number of different places in the protocol stack (depending on the source and destination of the packet, it is possible that ipfw is invoked multiple times on the same packet). The packet passed to the firewall is compared against each of the rules in the firewall ruleset. When a match is found, the action corresponding to the matching rule is performed.

An ipfw ruleset always includes a default rule (numbered 65535) which cannot be modified or deleted, and matches all packets. The action associated with the default rule can be either deny or allow depending on how the kernel is configured.

If the ruleset includes one or more rules with the keep-state or limit option, then ipfw assumes a stateful behaviour, i.e. upon a match it will create dynamic rules matching the exact parameters (addresses and ports) of the matching packet.

These dynamic rules, which have a limited lifetime, are checked at the first occurrence of a check-state, keep-state or limit rule, and are typypically used to open the firewall on-demand to legitimate traffic only. See the STATEFUL FIREWALL and EXAMPLES Sections below for more information on the stateful behaviour of ipfw.

All rules have a few associated counters: a packet count, a byte count and a timestamp indicating the time of the last match. Counters can be displayed or reset with ipfw commands.

Rules can be added with the add command; deleted individually with the delete command, and globally with the flush command; displayed, optionally with the content of the counters, using the show and list commands. Finally, counters can be reset with the zero and resetlog commands.

The following options are available:

-a	While listing, show counter values. The show command just implies this option.
-d	While listing, show dynamic rules in addition to static ones.
-e	While listing, if the -d option was specified, also show expired dynamic rules.
-f	Don't ask for confirmation for commands that can cause problems if misused, i.e. flush. If there is no tty associated with the process, this is implied.
-q	While adding, zeroing, resetlogging or flushing, be quiet about actions (implies -f). This is useful for adjusting rules by executing multiple ipfw commands in a script (e.g., `sh rc.firewall'), or by processing a file of many ipfw rules, across a remote login session. If a flush is performed in normal (verbose) mode (with the default driver configuration), it prints a message. Because all rules are flushed, the message might not be delivered to the login session, causing the remote login session to be closed and the remainder of the ruleset is not processed. Access to the console would then be required to recover.
-t	While listing, show last match timestamp.
-N	Try to resolve addresses and service names in output.
enum	Show interfaces list (only for Windows).
sysctl	Show sysctl values. You may change it - for example: "ipfw sysctl debug=0"

To ease configuration, rules can be put into a file which is processed using ipfw as shown in the first synopsis line. An absolute pathname must be used. The file will be read line by line and applied as arguments to the ipfw utility.

Rule format

The format of ipfw rules is the following:

```
[prob match_probability] action [log [logamount number]] proto from src to dst [interface-spec] [options]
```

where the body of the rule specifies which information is used for filtering packets, among the following:

IPv4 Protocol	TCP, UDP, ICMP, etc.
Source and	possibly masked

destination IP address	
Source and destination port	lists, ranges or masks
Direction	(incoming or outgoing)
Transmit and receive interface	By name or address
Misc. IP header fields	Version, type of service, datagram length, identification, fragment flag (non-zero IP offset), Time To Live
Misc. TCP header fields	TCP flags (SYN, FIN, ACK, RST, etc.), sequence number, acknowl-edgment number, window
TCP options	
ICMP types	for ICMP packets

Note that some of the above information, e.g. IP addresses and TCP/UDP ports, could easily be spoofed, so filtering on those fields alone might not guarantee the desired results.

rule_number

Each rule is associated with a rule_number in the range 1..65535, with the latter reserved for the default rule. Rules are checked sequentially by rule number. Multiple Rules can have the same number, in which case they are checked (and listed) according to the order in which they have been added. If a rule is entered without specifying a number, the kernel will assign one in such a way that the rule becomes the last one before the default rule. Automatic rule numbers are assigned by incrementing the last non- default rule number on 100. If this is not possible (e.g. because we would go beyond the maximum allowed rule number), the same number of the last non-default value is used instead.

prob match_probability

A match is only declared with the specified probability (floating point number between 0 and 1). This can be useful for a number of applications such as random packet drop to simulate the effect of multiple paths leading to out-of-order packet delivery.

log [logamount number]

When a packet matches a rule with the log keyword, a message will be logged to windows\security\logs\wipfwYYYYMMDD.log.
Once the limit is reached, logging can be re-enabled by clearing the logging counter or the packet counter for that entry, see the resetlog command.
Note:
logging is done after all other packet matching conditions have been successfully verified, and before performing the final action (accept, deny, etc.) on the packet.

Rule actions

A rule can be associated with one of the following actions, which will be executed when the packet matches the body of the rule.

allow | accept | pass | permit

Allow packets that match rule. The search terminates.

check-state

Checks the packet against the dynamic ruleset. If a match is found, execute the action associated with the rule which generated this dynamic rule, otherwise move to the next rule. Check-state rules do not have a body. If no check-state rule is found, the dynamic ruleset is checked at the first keep-state or limit rule.

count

Update counters for all packets that match rule. The search continues with the next rule.

deny | drop

Discard packets that match this rule. The search terminates.

skipto number

Skip all subsequent rules numbered less than number. The search continues with the first rule numbered number or higher.

Rule body

The body of a rule contains zero or more patterns (such as specific source and destination addresses or ports, protocol options, incoming or outgoing interfaces, etc.) that the packet must match in order to be recognised. The body of a rule must in general include a source and destination addres specifier. The keyword any can be used in various places to specify that the content of a required field is irrelevant.

The rule body has the following format:

```
[proto from src to dst] [options]
```

Rule fields have the following meaning:

proto: protocol

A protocol from the TCP/IP reference model specified by number or name (for a complete list see windows\system32\drivers\etc\protocol). The ip or all keywords mean any protocol will match.

src and dst: ip-address [ports]

A single ip-address optionally followed by ports specifiers.

ip-address

An address specified in one of the following ways, optionally preceded by a not operator:

any

matches any IP address.

numeric-ip | hostname

Matches a single IPv4 address, specified as dotted-quad or a hostname. Hostnames are resolved at the time the rule is added to the firewall list.

addr/bits

An IP number with a mask width of the form 1.2.3.4/24. In this case all IP numbers from 1.2.3.0 to 1.2.3.255 will match.

addr:mask

An IP number with a mask of the form 1.2.3.4:255.255.240.0. In this case all IP numbers from1.2.0.0 to 1.2.15.255 will match.

The sense of the match can be inverted by preceding an address with the not modifier, causing all other addresses to be matched instead. This does not affect the selection of port numbers.

ports: {port|port-port|port:mask}[,port[,...]]

The `-' notation specifies a range of ports (including boundaries).

The `:' notation specifies a port and a mask, a match is declared if the port number in the packet matches the one in the rule, limited to the bits which are set in the mask.

Service names may be used instead of numeric port values.

A range may only be specified as the first value, and the length of the port list is limited to 10 ports

Rule options (match patterns)

Additional match patterns can be used within rules. Zero or more of these so-called options can be present in a rule.

```
ipfw add 100 allow ip from not 1.2.3.4 to any
```

The following match patterns can be used (listed in alphabetical order):

established

Matches TCP packets that have the RST or ACK bits set.

fragment

Matches packets that are fragments and not the first fragment of an IP datagram. Note that these packets will not have the next protocol header (e.g. TCP, UDP) so options that look into these headers cannot match.

icmptypes types

Matches ICMP packets whose ICMP type is in the list types. The list may be specified as any combination of ranges or individual types separated by commas. The supported ICMP types are:

echo reply (0), destination unreachable (3), source quench (4), redirect (5), echo request (8), router advertisement (9), router solicitation (10), time-to-live exceeded (11), IP header bad (12), timestamp request (13), timestamp reply (14), information request (15), information reply (16), address mask request (17) and address mask reply (18).

in | out

Matches incoming or outgoing packets, respectively..

ipoptions spec

Matches packets whose IP header contains the comma separated list of options specified in spec. The supported IP options are:
ssrr (strict source route), lsrr (loose source route), rr (record packet route) and ts (timestamp). The absence of a particular option may be denoted with a `!'.

keep-state

Upon a match, the firewall will create a dynamic rule, whose default behaviour is to match bidirectional traffic between source and destination IP/port using the same protocol. The rule has a limited lifetime, and the lifetime is refreshed every time a matching packet is found.

limit {src-addr | src-port | dst-addr | dst-port} N

The firewall will only allow N connections with the same set of parameters as specified in the rule. One or more of source and destination addresses and ports can be specified.

recv | xmit | via {ifX | if* | ipno | any}

Matches packets received, transmitted or going through, respectively, the interface specified by by device name (if*), by IP address, or through some interface.

The via keyword causes the interface to always be checked. If recv or xmit is used instead of via, then only the receive or transmit interface (respectively) is checked. By specifying both, it is possible to match packets based on both receive and transmit interface, e.g.:

```
ipfw add deny ip from any to any out recv ppp1 xmit eth1
```

The recv interface can be tested on either incoming or outgoing packets, while the xmit interface can only be tested on outgoing packets. So out is required (and in is invalid) whenever xmit is used.

setup

Matches TCP packets that have the SYN bit set but no ACK bit. This is the short form of ``tcpflags syn,!ack''.

tcpflags spec

TCP packets only. Match if the TCP header contains the comma separated list of flags specified in spec. The supported TCP flags are:

fin, syn, rst, psh, ack and urg. The absence of a particular flag may be denoted with a `!'. A rule which contains a tcpflags specification can never match a fragmented packet which has a non-zero offset. See the frag option for details on matching fragmented packets.

tcpoptions spec

TCP packets only. Match if the TCP header contains the comma separated list of options specified in spec. The supported TCP options are:

mss (maximum segment size), window (tcp window advertisement), sack (selective ack), ts (rfc1323 timestamp) and cc (rfc1644 t/tcp connection count). The absence of a particular option may be denoted with a `!'.

Checklist

Here are some important points to consider when designing your rules:

- Remember that you filter both packets going in and out. Most connections need packets going in both directions.
- Remember to test very carefully. It is a good idea to be near the console when doing this.
- Don't forget the loopback interface.

Stateful firewall

Stateful operation is a way for the firewall to dynamically create rules for specific flows when packets that match a given pattern are detected. Support for stateful operation comes through the check-state, keep-state and limit options of rules.

Dynamic rules are created when a packet matches a keep-state or limit rule, causing the creation of a dynamic rule which will match all and only packets with a given protocol between a src-ip/src-port dst-ip/dst-port pair of addresses (src and dst are used here only to denote the initial match addresses, but they are completely equivalent afterwards). Dynamic rules will be checked at the first check-state, keep-state or limit occurrence, and the action performed upon a match will be the same as in the parent rule.

Note that no additional attributes other than protocol and IP addresses and ports are checked on dynamic rules.

The typical use of dynamic rules is to keep a closed firewall configuration, but let the first TCP SYN packet from the inside network install a dynamic rule for the flow so that packets belonging to that session will be allowed through the firewall:

```
ipfw add check-state
ipfw add allow tcp from my-subnet to any setup keep-state
ipfw add deny tcp from any to any
```

A similar approach can be used for UDP, where an UDP packet coming from the inside will install a dynamic rule to let the response through the firewall:

```
ipfw add check-state
ipfw add allow udp from my-subnet to any keep-state
ipfw add deny udp from any to any
```

Dynamic rules expire after some time, which depends on the status of the flow.

See Section EXAMPLES for more examples on how to use dynamic rules.

Examples

There are far too many possible uses of ipfw so this Section will only give a small set of examples.

Basic packet filtering

This command adds an entry which denies all tcp packets from cracker.evil.org to the telnet port of wolf.tambov.su from being forwarded by the host:

```
ipfw add deny tcp from cracker.evil.org to wolf.tambov.su telnet
```

This one disallows any connection from the entire crackers network to my host:

```
ipfw add deny ip from 123.45.67.0/24 to my.host.org
```

A first and efficient way to limit access (not using dynamic rules) is the use of the following rules:

```
ipfw add allow tcp from any to any established
ipfw add allow tcp from net1 portlist1 to net2 portlist2 setup
ipfw add allow tcp from net3 portlist3 to net3 portlist3 setup
...
ipfw add deny tcp from any to any
```

The first rule will be a quick match for normal TCP packets, but it will not match the initial SYN packet, which will be matched by the setup rules only for selected source/destination pairs. All other SYN packets will be rejected by the final deny rule.

dynamic rules

In order to protect a site from flood attacks involving fake TCP packets, it is safer to use dynamic rules:

```
ipfw add check-state
ipfw add deny tcp from any to any established
ipfw add allow tcp from my-net to any setup keep-state
```

This will let the firewall install dynamic rules only for those connection which start with a regular SYN packet coming from the inside of our network. Dynamic rules are checked when encountering the first check-state or keep-state rule. A check-state rule should usually be placed near the beginning of the ruleset to minimize the amount of work scanning the ruleset.

To limit the number of connections a user can open you can use the following type of rules:

```
ipfw add allow tcp from my-net/24 to any setup limit src-addr 10
ipfw add allow tcp from any to me setup limit src-addr 4
```

The former (assuming it runs on a gateway) will allow each host on a /24 network to open at most 10 TCP connections. The latter can be placed on a server to make sure that a single client does not use more than 4 simultaneous connections.

BEWARE: stateful rules can be subject to denial-of-service attacks by a SYN-flood which opens a huge number of dynamic rules.

This rule drops random incoming packets with a probability of 5%:

```
ipfw add prob 0.05 deny ip from any to any in
```

Here is a good usage of the list command to see accounting records and timestamp information:

```
ipfw -at list
```

or in short form without timestamps:

```
ipfw -a list
```

which is equivalent to:

```
ipfw show
```

See also

S. Floyd and V. Jacobson, Random Early Detection gateways for Congestion Avoidance, August 1993.

B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, Recommendations on Queue Management and Congestion Avoidance in the Internet, April 1998, RFC 2309.

Warning!

Misconfiguring the firewall can put your computer in an unusable state, possibly shutting down network services and requiring console access to regain control to it.

Authors

- Ugen J. S. Antsilevich
- Poul-Henning Kamp
- Alex Nash
- Archie Cobbs
- Luigi Rizzo

API based upon code written by Daniel Boulet for BSDI.

IPFW ported to Windows® by Ruslan Staritsin and Vladislav Goncharov.