



fw monitor Quick Facts

fw monitor is part of every FW-1 installation and the syntax is the same for all possible installations. Contrary to snoop or tcpdump, fw monitor does not put an interface into promiscuous mode because it works as a kernel module. Therefore, fw monitor can capture packets on different positions of the FW-1 chain and on several interfaces at once but fw monitor won't display any MAC addresses because it's not working on layer 2.

Capture files written with fw monitor can be read with snoop, tcpdump or Wireshark. You can configure Wireshark to show the packet direction by checking "Interpret as Firewall-1 monitor file" under "Protocols -> Ethernet" in the preferences and adding an additional column with type "FW-1 monitor if/direction" in "Appearance -> Columns".

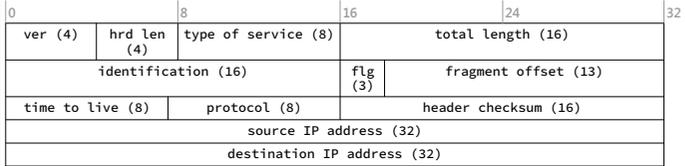
Remark: Any policy install or uninstall will cause fw monitor to exit.

Remark: Disable SecureXL (fwaccel off or fwaccel6 off) before running fw monitor.

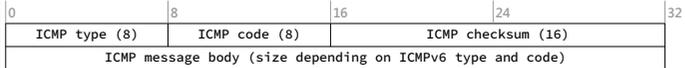
Remark: Try to use fw monitor in expert mode, clish sometimes breaks it.

Protocol Header Review (field length in bits in brackets)

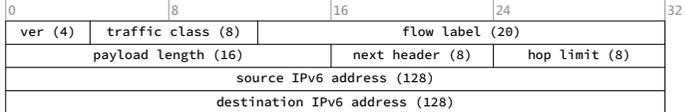
IP Header:



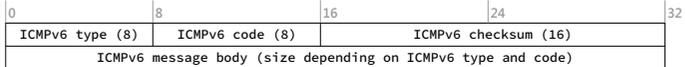
ICMP Header:



IPv6 Header:



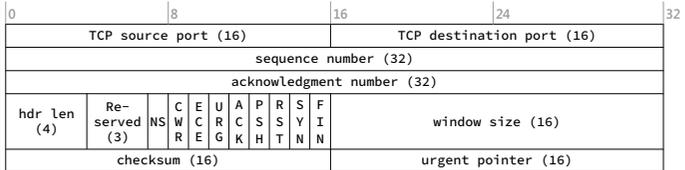
ICMPv6 Header:



UDP Header:



TCP Header:



Useful Links

- sk30583 Check Point sk30583 - What is FW Monitor?
How-To "How to use fw monitor" PDF by Check Point. Last updated 2003.
tcpdump101 Generate CLI capture commands for several tools, including fw monitor.
fw1-dump.sh Script by AREAsec for using fw monitor with tcpdump syntax.
Ginspect Generate inspect and tcpdump expressions online.

fw monitor Cheat Sheet, 2018, current version available at https://roesen.org. Licensed under Creative Commons BY-NC-SA License. Firewall-1, Endpoint Connect VPN Client and VSX are registered trademarks of Check Point Software Technologies, Ltd.

fw monitor Syntax and Options

fw monitor [-h] [-u|s] [-i] [-d] [-T] [{"-e expr"}+|-f <filter-file>|-] [-l len] [-m mask] [-x offset[,len]] [-o <file>] [<-pi pos>] [<-pI pos>] [<-po pos>] [<-p0 pos>] | -p all [-a]> [-ci count] [-co count] [-v <vsid>]

Table with 2 columns: Option, Description. Options include -h (Print usage message), -u|s (Show UUID or SUUID for every packet), -i (Captured package data is written to standard out at once), -d / -D (debugging or even more debugging), -T (Show date and timestamp for every processed packet), -e <expr> (Set capture filter expression), -f <file> (Read capture filter from file), -f - (Read filter from standard input), -l <len> (Limit packet data which will be read), -m <mask> (Define which packets from which position), -x (Print raw packet data), -o <file> (Write packet capture to specified file), -p[x] pos (Insert fw monitor at a specific position), -p all (Use absolute chain positions), -ci <count> (Stop capture after count incoming packets), -co <count> (Stop capture after count outgoing packets), -v <vsid> (Capture on a specific virtual machine).

FW-1 Chain, Capture Masks and Positioning

Per default, the fw monitor kernel module will capture traffic at different FW-1 chain positions (inspection points) relative to the Virtual Machine within the chain:

- i pre-inbound, before the VM on the incoming interface
I post-inbound, after the VM on the incoming interface
o pre-outbound, before the VM on the outgoing interface
O post-outbound, after the VM on the outgoing interface
e pre-encryption, only with enabled IPSec VPN Software Blade
E post-encryption, only with enabled IPSec VPN Software Blade

While running fw monitor, you can check its chain positions using the command fw ctl chain on a second terminal. Of course, when dealing with IPv6, use fw6 instead of fw:

```
in chain (8):
0: -70000000 (8b442e70) (fffffff) fwmonitor (i/f side)
1: -1fffff8 (8b46f300) (00000001) Stateless verifications (in) (asm)
2: 0 (8b40a980) (00000001) fw VM inbound (fw)
3: 10 (8b420450) (00000001) fw accounting inbound (acct)
4: 70000000 (8b442e70) (fffffff) fwmonitor (IP side)
5: 7f600000 (8b462a00) (00000001) fw SCV inbound (scv)
6: 7f730000 (8b68c7b0) (00000001) passive streaming (in) (pass_str)
7: 7f750000 (8b7dc5d0) (00000001) TCP streaming (in) (cpas)
out chain (8):
0: -70000000 (8b442e70) (fffffff) fwmonitor (i/f side)
1: -1fffff0 (8b7dc810) (00000001) TCP streaming (out) (cpas)
2: -1ffff50 (8b68c7b0) (00000001) passive streaming (out) (pass_str)
3: -1f000000 (8b46f300) (00000001) Stateless verifications (out) (asm)
4: 0 (8b40a980) (00000001) fw VM outbound (fw)
5: 70000000 (8b442e70) (fffffff) fwmonitor (IP side)
6: 7f000000 (8b420450) (00000001) fw accounting outbound (acct)
7: 7f700000 (8b7dca10) (00000001) TCP streaming post VM (cpas)
```

The pre-inbound module (i) is located at position 0 and the post-inbound (I) module at position 4 of the in chain, pre-outbound (o) is sitting at position 0 and post-outbound (O) at position 5 of the out chain. Depending on the activated blades and features the chain can be a lot longer. You can use the names in brackets like (asm) for positioning. See below.

With the option -m <mask> you can define on which positions fw monitor should capture packets: fw monitor -m 10 would only capture pre-in and post-out. Default is set to iIoO.

With the -p option you can specify the position of each modules in the chain. You define the relative position using a switch like -p0 6 to place the post-outbound module at position 6 in the out chain. You can also use an alias like -pi -asm to place the pre-inbound module before (-) the "Stateless verifications" module or -pi +asm to place it after (+) it. Absolute positioning can be done by providing the absolute position in hex: -pi -0x1fffff4. Absolute position before the VM have a negative value. With -p all modules will be placed everywhere in the chain.

Understanding fw monitor Output

Using fw monitor you will normally see two lines of output for each fw monitor filter position in the FW-1 chain the packet passes. If the transport protocol (like TCP or UDP) is not known to fw monitor (fi. with encrypted traffic), the second line can be omitted. See the four marked sections in the following example from an SSH to the gateway running fw monitor:

```
# fw monitor
monitor: getting filter (from command line)
monitor: compiling
monitorfilter:
Compiled OK.
monitor: loading
monitor: monitoring (control-C to stop)
[fw_1]eth1:i[60]: 192.168.4.57 -> 192.168.42.80 (TCP) len=60 id=3017
TCP: 60386 -> 22 .S.... seq=1eaf882c ack=00000000
[fw_1]eth1:I[60]: 192.168.4.57 -> 192.168.42.80 (TCP) len=60 id=3017
TCP: 60386 -> 22 .S.... seq=1eaf882c ack=00000000
[fw_1]eth1:o[60]: 192.168.4.80 -> 192.168.42.57 (TCP) len=60 id=1166
TCP: 22 -> 60386 .S..A. seq=bb543e77 ack=1eaf882d
[fw_1]eth1:O[60]: 192.168.4.80 -> 192.168.42.57 (TCP) len=60 id=1166
TCP: 22 -> 60386 .S..A. seq=bb543e77 ack=1eaf882d
[...]
```

Section 1 tells us that the packet was captured on the interface eth1 in inbound direction (eth1:i) before reaching the virtual machine/the rulebase itself (i, pre-inbound). The packet length is [60] bytes, the source of the packet is 192.168.4.57 and its destination is 192.168.42.80, it carries (TCP), has, again, a length of len=60 bytes and the ID id=3117. The two length indicators can differ when dealing with fragmentation. In that case, the one in square brackets is the defragmented size and the second is the size of the fragment.

The second line of 1 shows us source (60386) and destination (22) ports, that the SYN flag is set (.S....) and sequence and acknowledged sequence numbers in HEX. In 2 we see the packet after passing the inbound FW VM.

In section 3 fw monitor displays a response packet as again seen on eth1 in the o position (eth1:o) - before entering the firewall VM in the outgoing chain. It has a different ID, source and destination address as well as ports switched places, there are now SYN and ACK flags present (.S..A.), a new sequence number and the acknowledged sequence number of the first packet. In 4 we can see the packet after passing the firewall VM.

Depending on your filter and the traffic you are capturing you will get different results. Packets passing the gateway may be displayed in iIoO, IP addresses may change due to NATing pp.

UUID and SUUID

Using the option -u or -s fw monitor shows the corresponding universal unique identifiers (UUID) or session UUID (SUUID) of the packet in the first line of the output. The UUID/SUUID consist of 4 32-bit integers: UNIX time, a counter, the gateway IP and a process number.

The SUUID is basically the same concept as UUID, but for services like ftp which need to have several connections (control an data connection), the SUUID stays the same for all these connections whereas there will be unique UUIDs for each of the separate connections.

Note that the first packet of a captured connection won't have an UUID pre-inbound, so the UUID field is all zeros. After passing the VM for the first time the connection gets it's UUID.

```
# fw monitor -e 'accept host(192.168.42.57) and port(22);' -u
monitor: getting filter (from command line)
no UUID pre-in
[...]
[fw_2] [00000000 - 00000000 00000000 00000000]:eth1:i[60]:
192.168.4.57 -> 192.168.42.80 (TCP) len=60 id=5172
TCP: 60446 -> 22 .S.... seq=086a3590 ack=00000000
[fw_2] [5ca20000 - 5b8ba25c 00000000 502aa8c0 c0000002]:eth1:I[60]:
192.168.4.57 -> 192.168.42.80 (TCP) len=60 id=5172
TCP: 60446 -> 22 .S.... seq=086a3590 ack=00000000
[fw_2] [5ca20000 - 5b8ba25c 00000000 502aa8c0 c0000002]:eth1:o[60]:
192.168.4.80 -> 192.168.42.57 (TCP) len=60 id=0
TCP: 22 -> 60446 .S..A. seq=589052f6 ack=086a3591
[...]
```

In front of the UUID and separated by a hyphen you'll see a manipulated version of the UUID. This 32-bit UUID will be placed in the last 32-bit of the destination mac address field in the ethernet frame of a capture file if the options -u or -s were used. Remember that fw monitor does not capture mac addresses. So the 12 bytes which would normally contain source and destination MAC are filled with packet direction and chain position (2 bytes), interface name (6 bytes) and UUID (4 bytes).

With UUIDs and SUUIDs you can easily follow packets through the firewall without having to worry for instance about NATing or protocols which use several connections like FTP.

Filter Expressions Basics

Filtering in `fw monitor` is done by filter expressions written in a subset of Check Points Inspect language which are read from the command line with the `-e` option, from a file passed over with the `-f` option or read from standard input with `-i`. The syntax for either way is

```
accept expression;
```

where `accept` does *not* mean the packet has to be accepted by the rulebase, just the filter has to accept the packet. You can also append the `accept` to the expression, separated by a comma.

```
expression, accept;
```

Make sure this syntax is always properly quoted by single (') or double quotes (") to protect it from the shell.

A simple expression is written in the following syntax

```
[offset:length,order] operator value
```

where `offset` is the offset in bytes from where `fw monitor` should start reading value. The `length` states for how many bytes (1,2 or 4) `fw monitor` should read the value. If no `length` is given, 4 bytes will be read. The order defines the byte order as `b` (big endian) or `l` (little endian) where `l` is the default when order is not given. The operator can be a relational or logical operator.

relational operators		logical operators	
<	less than	and	logical AND
>	greater than	,	logical AND
<=	less than or equal to	or	logical OR
>=	greater than or equal to	xor	logical XOR
is or =	equal	not	logical NOT
is not or !=	not equal		

value is one of the data types hex integers, octal integers, decimal integers or IP addresses.

So with an IP header size of 20 bytes, a simple `fw monitor` call with an expression to filter packets with the destination port 22 (SSH) would be

```
fw monitor -e 'accept [22:2,b]=22;'
```

or when using IPv6 with its fixed 40 byte IP header size, not taking possible extension headers into account (see examples for that problem) and an alternate relational operator

```
fw6 monitor -e 'accept [40:2,b] is 22;'
```

Explanation: Start filtering after the 22nd (40th with the IPv6 example) byte from the beginning of the IP packet and then for the next 2 bytes in big endian byte order look for the value "22".

To filter for anything except SSH you have to add logical operators. Something like

```
fw monitor -e 'accept [20:2,b]!=22 and [22:2,b]!=22;'
```

Using hex to look for packets with SYN and ACK flags set:

```
fw monitor -e 'accept [33:1]=0x12;'
```

A filter for the source IP address 10.10.1.70 would be

```
fw monitor -e 'accept [12,b]=10.10.1.70;'
```

An IPv4 address is 4 bytes long, so we can omit the `length`, as 4 bytes is the default value. But since 4 bytes is the maximum supported length value, you can't filter for IPv6 addresses this way - [8:16] and [24:16] will be rejected with "ERROR: invalid length: 16". You can however use [8:ipv6] and [24:ipv6] to filter for IPv6 source and destination addresses ;)

A notable exception from this byte specific syntax is `ifid`. With `ifid` you can limit the capture to specific interface IDs. A list of the current interfaces and their corresponding IDs can be displayed by `fw ctl iflist`. Example limiting the capture to two interfaces:

```
fw monitor -e 'accept ifid is 1 or ifid is 3;'
```

packetmon on a Endpoint Connect VPN Client (E75.30 and up)

To capture traffic on a Endpoint Connect client you have to use the tool `PacketMon.exe` located in your Endpoint Connect installation dir. The syntax is similar to the `fw monitor` one. At least with the old tool, `srfw.exe`, it was safer to just capture all traffic and use Wireshark for filtering, since some syntax options were simply and silently ignored. I'd suggest to do the same with `packetmon` unless you confirmed that all options are working as intended.

```
PacketMon.exe -o capture_file.cap
```

See [sk95486](#) for more information and additional links.

Filtering with Macros

Filtering gets a lot easier when using macros. There are several macros, mostly defined in two files: `$FWDIR/lib/tcpip.def` and `$FWDIR/lib/fwmonitor.def`. The macros defined in `fwmonitor.def` often use `tcpip.def` macros which then point to the actual expression.

Here are two examples of macros and to which definition and expression they point to (→):

Macro	Defined in fwmonitor.def	Defined in tcpip.def	Final expression
src	src → ip_src	ip_src → [12,b] (for IPv4)	[12,b] (for IPv4)
tcp	tcp → ip_p=PROTO_tcp	ip_p → [6:1] (for IPv6) PROTO_tcp → 6	[6:1]=6 (for IPv6)

These are some of the most helpful macros defined in above mentioned files aside from the obvious ones like `http`, `https`, `dns`, `ftp`, `tcp` or `udp` which don't need further explanation.

Macro	Explanation
port(port)	Filters for packets with port as source or destination port.
host(addr)	Filters for addr as source or destination address.
sport operator port	Filters for packets where source port is operator port.
dport operator port	Filters for packets where destination port is operator port.
ip_p operator proto	Capture packets with matching IANA protocol/nh number.
src operator addr	Filters for packets where source address is operator addr.
dst operator addr	Filters for packets where destination address is operator addr.
icmp4	Filters for ICMPv4 packets.
icmp6	Filters for ICMPv6 packets.
tcpport(port)	Filter TCP traffic to or from port port.
udpport(port)	Filter UDP traffic to or from port port.
icmp_error	Filters for ICMP packets of the following types: destination unreachable (3), source quench (4), redirect (5), time exceeded (11) or parameter problem (12).
ping	Filters for ICMP echo request and ICMP echo reply packets.
ike	Filters for packets with port 500.
natt	Filters for packets with port 4500.
tracert	Filters for packets specific to the Windows tracert command: ICMP echo requests with a TTL below 30 or ICMP time exceeded messages.
traceroute	Packets specific to the Unix traceroute command: UDP packets to destination port higher than 33000 (normal traceroute starts with port 33434) and a TTL below 30 or ICMP time exceeded messages.
net(net, masklen)	Packets to or from the network net with the mask masklen.
from_net(net, masklen)	Filters for packets from the network net with the mask masklen.
to_net(net, masklen)	Filters for packets to the network net with the mask masklen.
syn	Filters for packets with SYN flag set.
ack	Filters for packets with ACK flag set.
fin	Filters for packets with FIN flag set.
first	Filters for packets with the SYN flag but without ACK flag.
established	Filters for packets with the ACK flag or without the SYN flag.
not_first	Filters for packets without the SYN flag.
last	Filters for packets with FIN and ACK flags set.
no_term	Filters for everything other than SSH and Telnet traffic.
no_mgmt	Filters for everything other than CP management traffic like CPMI, CPD and AMON.
pull	Filter for SIC certificate pulls from management server.
push	Filter for SIC certificate pushes to gateways.
vpnd	Filters for IKE, NAT traversal, UDP encapsulated IPsec, RDP, CP topology updates, CP tunnel tests, L2TP and Secure Client keepalives.
vpncall	Filters for ESP, HTTPS and port 444, everything from vpnd, CP CA CRL downloads and user registration with a policy server.
ip_ttl6 operator ttl	Filters for IPv6 packets where hop limit/TTL is operator ttl.
ip_p6 operator nh	Filter for IPv6 packets where next header field is operator nh.

Filtering with Filter Files

`fw monitor` can read all filter expressions from a file or even from standard input. Just put the expression in a file and let `fw monitor` read it with the `-f` option.

```
# echo "accept [22:2,b]!=22 and [20:2,b]!=22;" > /tmp/fwmon.filter
# fw monitor -f /tmp/fwmon.filter
monitor: getting filter (from /tmp/fwmon.filter)
[...]
monitor: monitoring (control-C to stop)
[fw_0] eth1:i[84]: 192.168.4.67 -> 192.168.42.80 (ICMP) len=84 id=48886
ICMP: type=8 code=0 echo request id=29701 seq=1
```

If you want to use macros inside a filter file, you have to include the appropriate definition file, otherwise compiling will result in an error:

```
# echo "accept sport!=22 and dport!=22;" > /tmp/fwmon.filter
# fw monitor -f /tmp/fwmon.filter
monitor: getting filter (from /tmp/fwmon.filter)
monitor: compiling
monitorfilter:
"/opt/CPsuite-R80/fw1/tmp/monitorfilter.pf", line 1: ERROR: cannot find
<sport> anywhere
Compilation Failed.
monitor: filter compilation failed
/opt/CPsuite-R80/fw1/tmp/monitorfilter
```

To fix this, just add `#include "fwmonitor.def"` to the file above your filter expression.

```
# echo '#include "fwmonitor.def"' > /tmp/fwmon.filter
# echo "accept sport!=22 and dport!=22;" >> /tmp/fwmon.filter
# fw monitor -f /tmp/fwmon.filter
monitor: getting filter (from /tmp/fwmon.filter)
monitor: compiling
monitorfilter:
Compiled OK.
[...]
```

Examples

Show packets with IP 192.168.1.12 as SRC or DST:
`fw monitor -e 'accept host(192.168.1.12);'`

Show all packets from 192.168.1.12 to 192.168.3.3:
`fw monitor -e 'accept src=192.168.1.12 and dst=192.168.3.3;'`

Show all packets with SYN and ACK flags set:
`fw monitor -e 'accept [33:1]=0x12;'`

Show all packets with PUSH bit set:
`fw monitor -e 'accept th_flags=TH_PUSH;'`

Like last example, only for IPv6. Show all packets with SYN and ACK bit set, take possible IPv6 extension headers into account instead of assuming to deal with a 40byte IPv6 header:
`fw6 monitor -e 'accept [PACKET_HDRLEN+13:1]=0x12;'`

Show UDP port 53 (DNS) packets, pre-in position is before 'ipopt_strip':
`fw monitor -pi ipopt_strip -e 'accept udpport(53);'`

Show UDP traffic from or to unprivileged ports, only show post-out:
`fw monitor -m 0 -e 'accept udp and (sport>1023 or dport>1023);'`

Show Windows traceroute (ICMP, TTL<30) from and to network 192.168.1.0/24
`fw monitor -e 'accept net(192.168.1.0,24) and tracert;'`

Capture web traffic on VSX virtual system ID 23:
`fw monitor -v 23 -e 'accept http or https;'`

Show all CP management traffic between network 10.23.42.0/24 and 192.168.42.80:
`fw monitor -e 'accept not no_mgmt and net(10.23.42.0, 24) and host(192.168.23.1);'`

Show IPv6 ICMP6 Router Advertisements:
`fw6 monitor -e 'accept icmp6_type is ND_ROUTER_ADVERT;'`

Show all ESP (IP protocol 50) packets on the interface with the ID 0:
`fw monitor -e 'accept ip_p=50 and ifid=0;'`

Show SMTP traffic including raw packet data after byte 52 and limit RAW output to 1500 bytes:
`fw monitor -e 'accept smtp;'-x 40,1500`