Check Point fw monitor cheat sheet – 20141028

by Jens Roesen – email – www - twitter

fw monitor Quick Facts

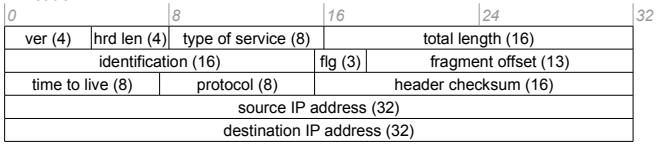
fw monitor is part of every FW-1 installation and the syntax is the same for all possible installations. Contrary to snoop or tcpdump, fw monitor does not put a interface into promiscuous mode because it works as a kernel module. Therefore fw monitor can capture packets on different positions of the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

Capture files written with fw monitor can be read with snoop, tcpdump or ethereal/wireshark. You can configure wireshark to show the packet direction by checking "Edit → Preferences → Protocols → Ethernet → Interpret as FireWall-1 monitor file" and adding an additional column via "Edit → Preferences → Columns → add Field type 'FW-1 monitor if/direction'"

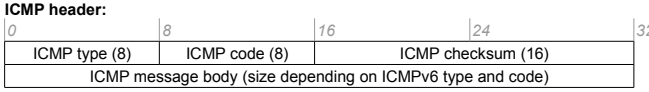
Notice: Any policy installation or uninstallation will cause fw monitor to exit.
Notice: Disable SecureXL (fwaccel off) before running fw monitor.

Protocol Header Review (field length in bits in brackets)

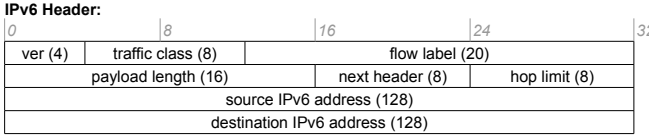
IP header:



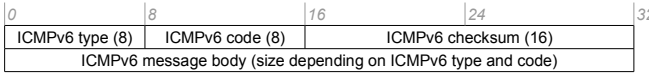
ICMP header:



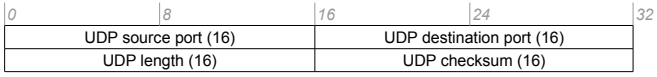
IPv6 Header:



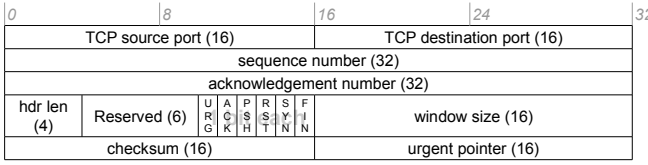
ICMPv6 Header:



UDP header:



TCP header:



Useful Links

http://bit.ly/fwmonref	Check Point fw monitor reference (PDF)
http://bit.ly/fw1dump	Shell script by AREAsec for using fw monitor with tcpdump syntax
http://bit.ly/ginspect2	Generate inspect and tcpdump expressions online

use http://longurl.org/expand to preview above shortened URLs if you don't trust me ;)

fw monitor Syntax and Options

```
fw monitor [- 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] [-pO pos] | -p all [-a ]> [-ci count] [-co count] [-v <vsid>]
```

-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
-e <expr>	set filter for expression on the command line
-f <file>	read file with filter expressions
-f -	read filter from standard input (end with ^D)
-l <len>	limit packet data which will be read
-m <mask>	define which packets from which position in the FW-1 chain to display: I (pre-in), I (post-in), o (pre-out) and/or O (post-out)
-x	print raw packate data, can be limited
-o <file>	write output to specified file instead of standard out
-p[x] pos	insert fw monitor at a specific position in the FW-1 chain, replace x by I (pre-in), I (post-in), o (pre-out) or O (post-out)
-p all	insert fw monitor between all FW-1 kernel modules
-a	use absolute chain positions when using -p all
-ci <count>	stop capture after count incoming packets
-co <count>	stop capture after count outgoing packets
-v <vsid>	capture on an specific virtual machine (only FW-1 VSX)

Understanding fw monitor Output

During normal usage of fw monitor you will see two lines of output for each fw monitor filter position in the FW-1 chain the captured packet passes. See the blue marker in following example. If the transport protocol (like TCP or UDP) is not known to fw monitor (f.i. with encrypted traffic), the second line can be omitted.

```
# fw monitor
 monitor: getting filter (from command line)
 monitor: compiling
monitorfilter:
Compiled OK.
 monitor: loading
 monitor: monitoring (control-C to stop)
eth0:i[84]: 192.168.1.70 -> 192.168.169.194 (TCP) len=84 id=60630
TCP: 52863 -> 22 ...PA. seq=ca983ae0 ack=02eb6b0b
eth0:I[84]: 192.168.1.70 -> 192.168.169.194 (TCP) len=84 id=60630
TCP: 52863 -> 22 ...PA. seq=ca983ae0 ack=02eb6b0b
eth0:o[284]: 192.168.169.194 -> 192.168.1.70 (TCP) len=284 id=1166
TCP: 22 -> 52863 ...PA. seq=02eb6b0b ack=ca983b0c
eth0:O[284]: 192.168.169.194 -> 192.168.1.70 (TCP) len=284 id=1166
TCP: 22 -> 52863 ...PA. seq=02eb6b0b ack=ca983b0c
[…] ^C
 monitor: caught sig 2
 monitor: unloading
```

The first marked line tells us that the the packet was captured on the interface eth0 in outbound direction before reaching the virtual machine/the rulebase itself (o, pre-outbound) and has a length of [284] bytes. The source of the packet is 192.168.169.194 and it's destination is 192.168.1.70, it carries (TCP), has a length of len=284 bytes (this can differ from the first length information due to fragmentation, then the first length is the total packet length and the second one only the length of the fragment) and the ID id=1166.
In the second line we can again see that the packet carries a TCP payload with a source Port of 22 and a destination Port of 52863. The **P**USH and **A**CK flags are set. The sequence number of the packet is 02eb6b0b and it acknowledges sequence number ca983b0c.

FW-1 Chain, Capture Masks And Positioning

Per default, the fw monitor kernel module will capture traffic at four different positions / capture points relative to the Virtual Machine within the FW-1 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

While running fw monitor you can check the position of the fw monitor modules in the incoming and outgoing chains using the command fw ctl chain on a second terminal:

```
# fw ctl chain
in chain (15):
  0: -7f800000 (f9acc050) (ffffffff) IP Options Strip (ipopt_strip)
  1: -70000000 (f9aa3aa0) (ffffffff) fwmonitor (i/f side)
  2: - 2000000 (f94a60a0) (00000003) vpn decrypt (vpn)
  3: - 1fffff6 (f9accff0) (00000001) Stateless verifications (asm)
  4: - 1fffff2 (f94c37d0) (00000003) vpn tagging inbound (tagging)
  5: - 1fffff0 (f94a6ec0) (00000003) vpn decrypt verify (vpn_ver)
  6: - 1000000 (f9afb670) (00000003) SecureXL conn sync (secxl_sync)
  7:           0 (f9a80970) (00000001) fw VM inbound  (fw)
  8:           1 (f9ad7390) (00000002) wire VM inbound  (wire_vm)
  9:   2000000 (f94a8fc0) (00000003) vpn policy inbound (vpn_pol)
  10:  10000000 (f9b00440) (00000003) SecureXL inbound (secxl)
  11:  70000000 (f9aa3aa0) (ffffffff) fwmonitor (IP  side)
  12:  7f600000 (f9ac5280) (00000001) fw SCV inbound (scv)
  13:  7f750000 (f9bd5260) (00000001) TCP streaming (in) (cpas)
  14:  7f800000 (f9acc370) (ffffffff) IP Options Restore (ipopt_res)
out chain (14):
  0: -7f800000 (f9acc050) (ffffffff) IP Options Strip (ipopt_strip)
  1: -70000000 (f9aa3aa0) (ffffffff) fwmonitor (IP  side)
  2: - 1ffffff (f94a7bf0) (00000003) vpn nat outbound (vpn_nat)
  3: - 1fffff0 (f9bd50b0) (00000001) TCP streaming (out) (cpas)
  4: - 1ff0000 (f94c37d0) (00000003) vpn tagging outbound (tagging)
  5: - 1f00000 (f9accff0) (00000001) Stateless verifications (asm)
  6:           0 (f9a80970) (00000001) fw VM outbound (fw)
  7:           1 (f9ad7390) (00000002) wire VM outbound  (wire_vm)
  8:   2000000 (f94a8a50) (00000003) vpn policy outbound (vpn_pol)
  9:  10000000 (f9b00440) (00000003) SecureXL outbound (secxl)
  10:  20000000 (f94a7e30) (00000003) vpn encrypt (vpn)
  11:  70000000 (f9aa3aa0) (ffffffff) fwmonitor (i/f side)
  12:  7f700000 (f9bd4ee0) (00000001) TCP streaming post VM (cpas)
  13:  7f800000 (f9acc370) (ffffffff) IP Options Restore (ipopt_res)
```

The pre-inbound module (i) is located at position 1 and the post-inbound (I) module at position 11 of the in chain, pre-outbound (o) is sitting at position 1 and post-outbound (O) at position 11 of the out chain. Without fw monitor running the blue lines would be missing.

With the -m <mask> option you can define on which positions fw monitor should capture packets: fw monitor -m iO would capture pre-in and post-out. Without -m the default is set to iIoO.

With the -p option you can specify the exact position of each of the four possible module positions. You can define the relative position using a switch like -pO 12 to place the post-outbound module at position 12 in the out chain or you can use an alias like -pi -tagging to place the pre-inbound module before (-) the vpn tagging module or -pi +tagging to place it after (+) the vpn tagging module. Absolute positioning can be done by providing the absolute position in hex: -pi -0x1fffff4. Absolute position before the VM have a negative value.

The option -p all places the fw monitor modules between *all* other modules. In the example above this would result in having fw monitor at 29 positions within the FW-1 chain.

Filter Expressions

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 `-f -`. 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. 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 were `fw monitor` should start reading `value`. The `length` states for how man 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) were `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		

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

So a simple `fw monitor` call with a expression to filter packets with the destination port 22 (SSH) would be

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

(after the 22nd byte from the beginning of the IP packet and for the next 2 bytes in big endian byte order look for the value 22)

A filter for the source IP address 10.10.1.70 would be

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

Because a IP address is 32 bit (4 bytes) we can omit the length here as 4 bytes is the default length.

To filter for anything except SSH you have to use logical operators:

```
fw monitor -e 'accept not ([20:2,b]=22 or [22:2,b]=22);'
```

`srfw.exe monitor` on a SecuRemote/SecureClient Device (FP3 and up)

To capture traffic on a SecuRemote/SecureClient device you have to use the tool `srfw.exe` located in `$SRDIR/bin` (normally `C:\Program Files\CheckPoint\SecuRemote\bin`). Although `srfw.exe` pretends to understand all `fw monitor` options it does not care for all of them and won't even complain. Simply capture all traffic for analysis with wireshark:

```
srfw.exe monitor -o output_file.cap
```

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 some examples:

filter	fwmonitor.def	tcpip.def	final expression
src	src → ip_src	ip_src → [12,b]	[12,b]
dst	dst → ip_dst	ip_dst → [16,b]	[16,b]
sport	sport → th_sport	th_sport → [20:2,b]	[20:2,b]
dport	dport → th_dport	th_dport → [22:2,b]	[22:2,b]
tcp	tcp → ip_p=PROTO_tcp	ip_p → [9:1] PROTO_tcp → 6	[9:1]=6

Take a look at these two files. There are a lot of useful macros defined in there, like `tracert` for capturing Windows like ICMP traceroutes oder `traceroute` for UDP traceroutes. You can also put your own definition files into `$FWDIR/lib` but do not edit any of the default definition files, because Check Point probably won't support them.

Filtering with Filter Files

`fw monitor` can read all filters from a file. Just put the expression in a file and let `fw monitor` read it with the `-f` option.

```
# echo "accept ([20:2,b]=22 or [22:2,b]=22);" > /tmp/filter.txt
# fw monitor -f /tmp/filter.txt
 monitor: getting filter (from /tmp/filter.txt)
 [...]
eth0:i[40]: 10.10.1.70 -> 192.168.201.2 (TCP) len=40 id=55606
TCP: 57005 -> 22 ....A. seq=c0768cd1 ack=1dd17e4f
```

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

```
# echo "accept (sport=22 or dport=22);" > /tmp/filter.txt
# fw monitor -f /tmp/filter.txt
 monitor: getting filter (from /tmp/filter.txt)
 monitor: compiling
monitorfilter:
"/opt/CPsuite-R65/fw1/tmp/monitorfilter.pf", line 1: ERROR: cannot
find <sport> anywhere
Compilation Failed.
 monitor: filter compilation failed /opt/CPsuite-
R65/fw1/tmp/monitorfilter
```

Including the definition file:

```
# echo '#include "fwmonitor.def"' > /tmp/filter.txt
# echo "accept (sport=22 or dport=22);" >> /tmp/filter.txt
# fw monitor -f /tmp/filter.txt
 monitor: getting filter (from /tmp/filter.txt)
 [...]
eth0:i[40]: 10.10.1.70 -> 192.168.201.2 (TCP) len=40 id=22439
TCP: 57005 -> 22 ....A. seq=c076a7b5 ack=1dd1b18f
```

UUID and SUUID

Using the option -u or -s fw monitor prints the corresponding universal unique identifiers (UUID) or session UUID (SUUID) in front of the first line of the output. Note that the first packet of a connection captured pre-inbound won't have a UUID, so the UUID field is all zeros. After passing the VM for the first time the connection get's it's UUID.

```
# fw monitor -e 'accept dst=192.168.201.12 and port(22);' -u
 monitor: getting filter (from command line)
```

↘

UUID and SUUID

```
 [...]
[00000000 - 00000000 00000000 00000000 00000000]:eth0:i[48]:
194.115.1.70 -> 195.243.201.12 (TCP) len=48 id=30575
TCP: 51650 -> 22 .S.... seq=7cddd1cc ack=00000000
[76620000 - 49786276 00010000 51eb14ac 000007b6]:eth0:I[48]:
194.115.1.70 -> 195.243.201.12 (TCP) len=48 id=30575
TCP: 51650 -> 22 .S.... seq=7cddd1cc ack=00000000
[76620000 - 49786276 00010000 51eb14ac 000007b6]:eth1:o[48]:
194.115.1.70 -> 195.243.201.12 (TCP) len=48 id=30575
TCP: 51650 -> 22 .S.... seq=7cddd1cc ack=00000000
[76620000 - 49786276 00010000 51eb14ac 000007b6]:eth1:O[48]:
194.115.1.70 -> 195.243.201.12 (TCP) len=48 id=30575
TCP: 51650 -> 22 .S.... seq=7cddd1cc ack=00000000
```

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

With UUIDs and SUUIDs you can easily follow packets on their way through the firewall without for instance having to worry about NAT or write extra filters for possible translated connections.

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 UDP port 53 (DNS) packets, pre-in position is before 'ippot_strip':

```
fw monitor -pi ipopt_strip -e 'accept udpport(53);'
```

Show UPD traffic from or to unprivileged ports, only show post-out

```
fw monitor -m O -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;'
```

Show Capture web traffic for VSX virtual system ID 23

```
fw monitor -v 23 -e 'accept tcpport(80);'
```

Show all ESP (IP protocol 50) packets on the interface with the ID 0.
(List interfaces and corresponding IDs with `fw ctl iflist`)

```
fw monitor -e 'accept ip_p=50 and ifid=0;'
```

Show traffic on a SecuRemote/SecureClient client into a file.
`srfw.exe` is in `$SRDIR/bin` (C:\Program Files\CheckPoint\SecuRemote\bin)

```
srfw monitor -o output_file.cap
```

Capturing IPv6 with `fw6 monitor`

From R75.40/GAiA on `fw6 monitor` is able to display captured IPv6 traffic:

```
# fw6 monitor -e 'accept icmp6_type=ICMP6_ECHO_REPLY;'
 monitor: getting filter (from command line)
 [...]
eth0:o[104]: 2001:db8::1 -> 2001:db8::2 payload length=64 (ICMPv6)
ICMPv6: type=129 code=0 echo reply id=14749 seq=9728
eth0:O[104]: 2001:db8::1 -> 2001:db8::2 payload length=64 (ICMPv6)
ICMPv6: type=129 code=0 echo reply id=14749 seq=9728
```

Unfortunately – at least during my tests - not all IPv6 filtering macros from `$FWDIR/lib/tcpip.def` were included. This was apparently caused by a failed recognition of the systems IPv6 state. You have to write your own definition file or go with normal filter expressions:

```
# fw6 monitor -e 'accept [40:1,b]=129;'
eth0:o[104]: 2001:db8::1 -> 2001:db8::2 payload length=64 (ICMPv6)
ICMPv6: type=129 code=0 echo reply id=14749 seq=11015
eth0:O[104]: 2001:db8::1 -> 2001:db8::2 payload length=64 (ICMPv6)
ICMPv6: type=129 code=0 echo reply id=14749 seq=11015
```