

fw monitor cheat sheet – v 0.1.1

Jens Roesen <drizzt@netzsherriff.de>

fw monitor quick facts

fw monitor is part of every FW-1 installation, independent from the underlying platform. Also the syntax is the same for all available platforms running FW-1. Contrary to tools like snoop or tcpdump, fw monitor does not need to put a interface into promiscuous mode because it works as a kernel module and resides within the FW-1 kernel module chain. Therefore fw monitor can capture packets on different positions of the FW-1 chain and several interfaces at once but fw monitor won't write down 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. There is a special ethereal version, CPEThereal, available from Check Point or you can configure wireshark to read and interpret the FW-1 chain by enabling Edit → Preferences → Protocols → Ethernet → Interpret as FireWall-1 monitor file.

packet header review

Unless you use macros you need to know packet headers in order to filter traffic.

IP header				32
0	8	16	24	
version	header length	type of service	total length	
identification (ID)		flags	fragment offset	
time to live (ttl)	protocol	header checksum		
source IP address				
destination IP address				

ICMP header			32
0	8	16	24
ICMP type	ICMP code	ICMP checksum	

UDP header				32
0	8	16	24	
UDP source port		UDP destination port		
UDP length		UDP checksum		

TCP header										32
0	8	16	24							
TCP source port				TCP destination port						
sequence number										
acknowledgement number										
header length	reserved	A C K	R S T	P S H	S Y N	F I N	window size			
checksum				urgent pointer						

fw monitor syntax

```
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]
```

options overview

- 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 (i, I, o and/or O) you want to see
- x print raw packate data, can be limited
- o file write output to file instead of standard out
- p[x] pos insert fw monitor at a specific position in the chain, replace x by i, I, o or O
- 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
- vs vsid capture on an specific virtual machine (only FW VSX)

understanding fw monitor output

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

```
# fw monitor
monitor: getting filter (from command line)
monitor: compiling
monitorfilter:
Compiled OK.
monitor: loading
monitor: monitoring (control-C to stop)
eth0:0[124]: 192.168.128.225 -> 10.10.1.70 (TCP) len=124 id=26516
TCP: 22 -> 15333 ...PA. seq=35444229 ack=0322b153
[...] ^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 it reached the virtual machine (o, pre-outbound) and has a length of [124] bytes. The sender of the packet is 192.168.128.225 and it's destination is 10.10.1.70, it carries (TCP), has a length of len=124 (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=26516.

In the second line we see that the packet carries a TCP payload with a source Port of 22 and a destination Port of 1533. The PUSH and ACK flags are set. The sequence number of the packet is 35444229 and it acknowledges sequence number 0322b153.

FW-1 chain, capture masks and positioning

The fw monitor kernel module can capture traffic at four different positions from the Virtual Machine within the FW-1 chain, each displayed by a single letter:

- 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:

```
# 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: - 1fffff6 (f94a7bf0) (00000003) vpn nat outbound (vpn_nat)
3: - 1fffff0 (f9bd50b0) (00000001) TCP streaming (out) (cpas)
4: - 1ff00000 (f94c37d0) (00000003) vpn tagging outbound (tagging)
5: - 1f000000 (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. Whithout 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. Without -m the default is iIoO.

With the -p option you can specify the position of each of the four modules. You can define the relative position using a number like -pO 12 to place the post-outbound module at position 12 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. 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: 29 packets captured for one packet.

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 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	,	logical AND
>	greater than	or	logical OR
<=	less than or equal to	xor	logical XOR
>=	greater than or equal to	not	logical NOT
is or =	equal		
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 an expression to filter packets for the destination port to be SSH (22) it 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 an 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 an operator

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

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 `fwmonitor.def` macros use `tcpip.def` macros which point to the actual expression. Here are some examples:

filter	fwmonitor.def	tcpip.def	expression
source IP	src	ip_src	[12,b]
destination IP	dst	ip_dst	[16,b]
source port	sport	th_sport	[20:2,b]
destination port	dport	th_dport	[22:2,b]

Take a look at these two files. 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.

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 gets its UUID.

```
# fw monitor -e 'accept dst=192.168.201.12 and (sport=22 or dport=22);' -u
monitor: getting filter (from command line)
monitor: compiling
monitorfilter:
Compiled OK.
monitor: loading
monitor: monitoring (control-C to stop)
[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
[...]
```

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

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.

filter files

`fw monitor` can read all filters from a file. Just put the expression in a testfile 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)
monitor: compiling
monitorfilter:
Compiled OK.
monitor: loading
monitor: monitoring (control-C to stop)
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 or compiling will result in an 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)
monitor: compiling
monitorfilter:
Compiled OK.
monitor: loading
monitor: monitoring (control-C to stop)
eth0:i[40]: 10.10.1.70 -> 192.168.201.2 (TCP) len=40 id=22439
TCP: 57005 -> 22 ....A. seq=c076a7b5 ack=1dd1b18f
[...]
```