This is a quick reference sheet of all usable options for the fw monitor tool . The previous experience with the tool is assumed. By default the fw monitor sniffing driver is inserted into the 4 locations on the Firewall kernel chain .

Here they are:

i (PREIN) – inbound direction before firewall Virtual Machine (VM, and it is CP terminology) . Most important fact to know about that is that this packet capturing location shows packets BEFORE any security rule in the policy is applied. That is, no matter what rules say a packet should at least be seen here, this would prove that packets actually reach the firewall at all.
i (POSTIN) – inbound direction after firewall VM.
o (PREOUT) – outbound direction before firewall VM,
O (POSTOUT) - outbound direction after firewall VM.

You can change point of insertion within the fw chain with :

 # fw monitor –p<i|I|O|o> <where to insert>

easiest way to specify where to insert is to first see the chain:
 # fw ctl chain
then give relative to any module you see there <+|->module_name

Now the usage itself:
 # fw monitor
Usage: fw monitor [- u|s] [-i] [-d] [-T] <{-e expression}+|-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]

Round up of options:

-m mask , which point of capture is to be displayed, possible: i,I,o,O
-d/-D debug output from fw monitor itself, not very useful IMO.
-u|s print also connection/session Universal ID
- i after writing each packet flush stdout
-T add timestamp, not interesting
-e expr expression to filter the packets (in detail later)
-f filter_file the same as above but read expression from file
-l <len> packet length to capture

Expressions
On the very low level fw monitor understands byte offsets from the header start. So to specify for example 20th byte of the IP packet (that is source IP) you can just use:

 # fw monitor -e 'accept [12,b]=8.8.8.8;'

Where:
12 – offset in bytes from the beginning of the packet
b – mandatory, means big endian order.
4 – not seen here but size (in bytes) of how many bytes to look for from the starting offset (default is 4)

To look for source port 53 (UDP/TCP) in raw packet:

 # fw monitor -m i -e 'accept [20:2,b]=53;'
Here I say to fw monitor to look at 2 bytes at offset 20.

While this way of looking at packets is the most general and therefore includes all cases, you rarely have the need for such a granular looking glass. In 99% of the cases you will be doing alright with a limited known set of expressions. Just for that Checkpoint defined and kindly provided us in every Splat installation with definition files that give meaningful synonyms to the most used patterns. There are few definition files but they circularly reference each other providing multiple synonyms for the same pattern.
I put all those predefined patterns in the list below for the easy to use reference.

Summary table of possible expressions to be fed to the fw monitor	
Specifying Hosts	
host(IP_address)	to or from this host
src=IP_address	where source ip = IP_address
dst=IP_address	where destination ip = IP_address
net(network_address,netmask)	to or from this network
to_net(network_address,netmask)	to this network
from_net(network_address,netmask)	from this network
Specifying ports	
port(port_number)	having this source or destination port
sport=port_number	having this source port
dport=port_number	having this destination port
tcpport(port_number)	having this source or destination port that is also TCP
udpport(port_number)	having this source or destination port that is also UDP
Specifying protocols	
ip_p=<protocol_number_as_per_IANA>	this way you can specifiy any known protocol by its registered number in IANA For detailed list of protocol numbers see http://www.iana.org/assignments/protocol-numbers/
icmp	what it says , icmp protocol
tcp	TCP
udp	UDP
Protocol specific oprions	
IP	
ip_tos = <value>	TOS field of the IP packet
ip_len = <length_in_bytes>	Length of the IP packet in bytes
ip_src/ ip_dst = <IP_address>	Source or destination IP address of the packet
ip_p =<protocol_number_as_per_IANA>	See above
ICMP	
echo_reply	ICMP reply packets
echo_req	Echo requests
ping	Echo requests and echo replies
icmp_error	ICMP error messages (Redirect,Unreachables,Time exceeded,Source quench,Parameter problem)
traceroute	Traceroute as implemented in Unix (UDP packets to high ports)
tracert	Traceroute as implemented in Windows (ICMP packets , TTL <30)
icmp_type = <ICMP types as per RFC>	catch packets of certain type
icmp_code = <ICMP type as per RFC>	catch packets of certain code
ICMP types and where applicable respective codes: ICMP_ECHOREPLY ICMP_UNREACH ICMP_UNREACH_NET	

ICMP_UNREACH_HOST ICMP_UNREACH_PROTOCOL ICMP_UNREACH_PORT ICMP_UNREACH_NEEDFRAG ICMP_UNREACH_SRCFAIL ICMP_SOURCEQUENCH ICMP_REDIRECT ICMP_REDIRECT_NET ICMP_REDIRECT_HOST ICMP_REDIRECT_TOSNET ICMP_REDIRECT_TOSHOST ICMP_ECHO ICMP_ROUTERADVERT ICMP_ROUTERSOLICIT ICMP_TIMXCEED ICMP_TIMXCEED_INTRANS ICMP_TIMXCEED_REASS ICMP_PARAMPROB ICMP_TSTAMP ICMP_TSTAMPREPLY ICMP_IREQ ICMP_IREQREPLY ICMP_MASKREQ ICMP_MASKREPLY	
icmp_ip_len = <length>	Length of ICMP packet
icmp_ip_ttl = <TTL>	TTL of ICMP packet, use with icmp protocol otherwise will catch ANY packet with TTL given
< cut here----bunch of other icmp-related fields like ID ,sequence I don't see any value in bringing here-->	
TCP	
syn	SYN flag set
fin	FIN flag set
rst	RST flag set
ack	ACK flag set
first	first packet (means SYN is set but ACK is not)
not_first	not first packet (SYN is not set)
established	established connection (means ACK is set but SYN is not)
last	last packet in stream (ACK and FIN are set)
tcpdone	RST or FIN are set
th_flags - more general way to match the flags inside TCP packets	
th_flags = TH_PUSH	Push flag set
th_flags = TH_URG	Urgent flag set
UDP	
uh_ulen = <length_in_bytes>	Length of the UDP header (doesnt include IP header)

And the last thing to remember before we move to examples . expressions support logical operators:
and - logical AND
or - logical OR
not - logical NOT
You can combine logical expressions and influence order by using ()