User Guide for mmb 0.3.2

Author(s): K.Edeline, J.Iurman

Contents

1 mmb

mmb (modular middlebox) is a vpp plugin that performs various middlebox behaviors.

2 mmb CLI guide

`mmb <command>`
SYNTAX : `enable|disable|add|add-stateless|add-stateful|del|list|flush|show`

This parameter determines the command applied on the rule list.
Allowed values:

- `enable` : enable mmb on a given interface
- `disable` : disable mmb on a given interface
- `add/add-stateless` : add a stateless rule
- `add-stateful` : add a stateful rule
- `del` : remove a rule
- `list` : list the rules
- `show` : display informations about mmb
- `flush` : remove all rules

2.1 add rules

```
mmb <add-keyword> <match> [<match> ...]  <target> [<target> ...]
```

- `<add-keyword>`
  SYNTAX : `add|add-stateless|add-stateful`

  The keyword determines if mmb has to keep track of connections that matches a given rule or not. `add/add-stateless` rules apply their targets at the packet level, each packet has to match the rule in order to apply the targets. `add-stateful` rules apply their targets at the connection/flow level. The `<match>` list of a stateful rule is used to add entries to the connection table. Once a connection is added to the table, the `<target>` list is applied to all packets of the connection, even if they don't match the rule. Additionally, `add-stateful` allows for special targets `map` and `shuffle`.

- `<match>`
  SYNTAX : `[!] <field> [[<cond>] <value>]`

  This parameter is a constraint that determines the packets on which the rule will operate.

  If a `<match>` is composed of a `<field>` alone, the constraint is that the packet should contain the field. If it is composed of a `<field>` and a `<value>`, the constraint is that the packet should contain the field and it should be set to the specified value. If it is composed of a `<field>`, a `<cond>` and a `<value>`, then the constraint is that the packet should contain the field, and the condition on the value should be true.

  The ! operator applied on one constraint performs the logical NOT of the constraint. Multiple constraints can be inputted for the same rule, the resulting constraint is the logical AND of all inputted constraints.

- `<target>`
  SYNTAX : `mod [...]|strip [...]|add [...]|drop [...]|map [...]|shuffle [...]`
  This parameter determines the action(s) to apply on matched packets.
  - `mod <field> <value>`
    Modify a field on a packet.
  - `add <field> <value>`
    Add a tcp-opt to the packet.
  - `strip [!] <field>`
    Strip options from a packet.
    If the ! operator is placed after the strip keyword, the following option will be added to the whitelist (the only authorized options), if not it will be added to the blacklist (the forbidden options).
    The special keyword `all` can be used in a `strip` target to strip all options from the matched packet.
  - `drop [<rate>]`
    Drop a packet with optional probability `<rate>` given in percentage, with a maximal precision of 0.01%. The default rate is 100%.
  - `map <field> <value>`
    Perform a bidirectionnal mapping of the given `<field>` to a given value. Valid fields are: `ip-saddr`, `ip-daddr`, `ip6-saddr`, `ip6-daddr`, `tcp-sport`, `tcp-dport`, `udp-sport`, `udp-dport`, `ip-id`, `ip6-flow-label`.
  - `shuffle <field>`
    Perform a bidirectionnal mapping of the given `<field>` to a random value. Valid fields are: `tcp-seq-num`, `tcp-ack-num`, `tcp-sport`, `tcp-dport`, `udp-sport`, `udp-dport`, `ip-id`, `ip6-flow-label`.

2.1.1 `<cond>`

A condition is applied on a `<value>` from a `<field>` to form a constraint.
Available conditions: ==

2.1.2 `<field>`

Available fields:

- interfaces: `in`, `out`
- IPv4 fields: `ip-ver`, `ip-ihl`, `ip-dscp`, `ip-ecn`, `ip-non-ect`[1], `ip-ect0`[1], `ip-ect1`[1], `ip-ce`[1], `ip-len`, `ip-id`, `ip-flags`, `ip-res`, `ip-df`, `ip-mf`, `ip-frag-offset`, `ip-ttl`, `ip-proto`, `ip-checksum`, `ip-saddr`[2], `ip-daddr`[2].

[1] not followed by `<value>`
[2] the `<value>` can include a subnet mask, the == condition will become subnet matching

- IPv6 fields: `ip6-ver`, `ip6-traffic-class`, `ip6-flow-label`, `ip6-len`, `ip6-next`, `ip6-hop-limit`, `ip6-saddr`[2], `ip6-daddr`[2], `ip6-payload`.
- ICMPv4 fields: `icmp-type`, `icmp-code`, `icmp-checksum`, `icmp-payload`.
- User Datagram Protocol (UDP) fields: `udp-sport`, `udp-dport`, `udp-len`, `udp-checksum`, `udp-payload`.
- Transmission Control Protocol (TCP) fields: `tcp-sport`, `tcp-dport`, `tcp-seq-num`, `tcp-ack-num`, `tcp-offset`, `tcp-res`, `tcp-cwr`, `tcp-ece`, `tcp-urg`, `tcp-ack`, `tcp-push`, `tcp-res`, `tcp-syn`, `tcp-fin`, `tcp-flags`, `tcp-win`, `tcp-checksum`, `tcp-urg-ptr`, `tcp-payload`.
- TCP options: `tcp-opt-mss`, `tcp-opt-wscale`, `tcp-opt-sackp`, `tcp-opt-sack`, `tcp-opt-timestamp`, `tcp-opt-fast-open`, `tcp-opt-fast-open`.
- Custom TCP options: `tcp-opt [<kind>]`
  Replace `<kind>` with the kind of option in decimal. When employed in a `<match>` without a `<kind>`, checks if the packet contains any option.

2.1.3 `<value>`

The value of a field is in decimal or in hexadecimal if preceeded by `x`. String value `ip`, `tcp`, `udp` and `icmp` can be used with fields `net-proto` or `ip-proto`.

2.1.4 stateful polices

2.2 Remove rules

- `del`
  SYNTAX : `mmb del <rule-index>`

  Delete rule at given index and associated entries in connections tables.
- `flush`
  SYNTAX : `mmb flush`

  Delete all rules and entries in the connection table.

2.3 Display informations

- `list`
  SYNTAX : `mmb list`

  List all rules.
- `show tables`
  SYNTAX : `mmb show tables [verbose]`

  Display informations about classifier tables such as masks, keys, capacity, and more.

- `show connections`
  SYNTAX : `mmb show connections [verbose]`

  Display informations about active connections used by stateful rules such as 5-tuples, connection type, expiring time, and more.

3 Examples

```
vpp# mmb add all mod ip-ecn 0
```
ECN bleaching

```
vpp# mmb add ip-proto udp drop
```
Block UDP

```
vpp# mmb add ip-proto != tcp drop
```
Block every IP protocol but TCP

```
vpp# mmb add ip-proto != udp ip-proto != tcp drop
```
Block every IP protocol but TCP and UDP

```
vpp# mmb add ip-proto udp drop
vpp# mmb add ip-proto tcp drop
```
Block TCP and UDP

```
vpp# mmb add tcp-dport 80 mod tcp-dport 443
```
Rewrite TCP port 80 to port 443

```
vpp# mmb add tcp-opt-mss strip tcp-opt-mss
```
Strip MSS option

```
vpp# mmb add tcp-opt-mss > 1500 mod tcp-opt-mss 1460
```
If MSS is larger than 1500, set it to 1460

```
vpp# mmb add tcp-opt strip ! tcp-opt-mss
```
Strip all options but MSS

```
vpp# mmb add tcp-opt strip tcp-opt-mss strip tcp-opt-wscale
```
Strip MSS and WScale

```
vpp# mmb add tcp-opt-timestamp strip all
```
Strip all options if packet contains timestamp option

```
vpp# mmb add tcp-opt strip ! tcp-opt-mss strip ! tcp-opt-wscale
```
Strip all options except mss and wscale if packet contains timestamp option (whitelist)

```
vpp# mmb add tcp-opt strip tcp-opt-mss strip tcp-opt-wscale
```
Strip all mss and wscale if packet contains timestamp option (blacklist)

```
vpp# mmb add tcp-opt ! tcp-opt-mss ! tcp-opt-wscale drop
```
Drop all TCP packets with options different than MSS or WScale.

```
vpp# mmb add ! tcp-opt-mss ! tcp-opt-wscale drop
```
Drop all TCP packets that do not contain MSS nor WScale.

```
vpp# mmb add tcp-opt 22 drop
```
Drop all TCP packets that contain option 22

```
vpp# mmb add-stateful ip-proto tcp ip-saddr 10.0.0.10/24 accept
vpp# mmb add ip-proto tcp drop
```
Reflexive ACL that blocks everything but tcp connections initated from 10.0.0.10/24

```
vpp# mmb add-stateful ip-proto tcp tcp-saddr 10.0.0.10/24 map ip-saddr 30.30.30.30
shuffle tcp-sport
```
Source NAT