Measurement and Architecture for a Middleboxed Internet

User Guide for mmb 0.1

Author(s): ULg 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 : `add|del|list|flush`

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

- `add` : add a policy to the policy list
- `del` : remove a policy from the policy list
- `list` : list the policies in the policy list.
- `flush` : remove all policies from the policy list

2.1 add/remove policies

```
mmb add|del <match> [<match> ...]  <target> [<target> ...]
```

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

  This parameter is a constraint that determines the packets on which the policy 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. The `-r` argument performs the logical NOT of the logical AND of all constraints (it reverses the matching constraint).

- `<target>`
  SYNTAX : `mod [...]|strip [...]|drop`

  This parameter determines the action(s) to apply on matched packets.

  - `mod <field> <value>` : modify a field on a packet.
  - `strip [!] <field> [<field> ...]` : strip options from a packet.
    If the ! operator is placed after the strip keyword, the following options define the option whitelist (the only authorized options), if not they define 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` : drop a packet

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:

- protocols[1]: `ip`, `tcp`, `udp`, `icmp`
- IPv4 fields: `ip-ver`, `ip-ihl`, `ip-dscp`, `ip-ecn`, `ip-non-ect`[2], `ip-ect0`[2], `ip-ect1`[2], `ip-ce`[2], `ip-len`, `ip-id`, `ip-flags`, `ip-res`, `ip-df`, `ip-mf`, `ip-frag-offset`, `ip-ttl`, `ip-proto`, `ip-checksum`, `ip-saddr`[3], `ip-daddr`[3].
- 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.
- Special options:
  - Reverse matching (`-r`)

2.1.3 `<value>`

The value of a field is in decimal or in hexadecimal if preceeded by `0x`.

2.1.4 Examples

```
$sudo vppctl mmb add ip-ecn mod 0
```

ECN bleaching

[1] applicable only to `<match>`

[2] not followed by `<value>`

[3] the following `<value>` can include a subnet mask, the == condition will become subnet matching

```
$sudo vppctl mmb add udp drop
```

Block UDP

```
$sudo vppctl mmb add ip-proto != tcp drop
```

Block every IP protocol but TCP

```
$sudo vppctl mmb add ip-proto != tcp ip-proto != udp drop
```

Block every IP protocol but TCP and UDP

```
$sudo vppctl mmb add ip-proto tcp drop
$sudo vppctl mmb add ip-proto udp drop
```

Block TCP and UDP

```
$sudo vppctl mmb add tcp-dport 80 mod 443
```

Rewrite TCP port 80 to port 443

```
$sudo vppctl mmb add tcp-opt-mss strip tcp-opt-mss
```

Strip mss option

```
$sudo vppctl mmb add tcp-opt-mss > 1500 mod tcp-opt-mss 1460
```

If MSS is larger than 1500, set it to 1460

```
$sudo vppctl mmb add ip-proto tcp strip !  tcp-opt-mss
```

Strip all options but MSS

```
$sudo vppctl mmb add ip-proto tcp strip tcp-opt-mss strip tcp-opt-wscale
```

Strip MSS and WSCALE

```
$sudo vppctl mmb add tcp-opt-timestamp strip all
```

Strip all options if packet contains timestamp option

```
$sudo vppctl mmb add ip-proto tcp strip !  tcp-opt-mss tcp-opt-wscale
```

Strip all options except mss and wscale if packet contains timestamp option (whitelist)

```
$sudo vppctl mmb add ip-proto tcp strip tcp-opt-mss tcp-opt-wscale
```

Strip all mss and wscale if packet contains timestamp option (blacklist)

```
$sudo vppctl mmb add tcp-opt !  tcp-opt-mss !  tcp-opt-wscale drop
```

Drop all TCP packets with options different than MSS or WScale.

```
$sudo vppctl mmb add ip-proto tcp !  tcp-opt-mss !  tcp-opt-wscale drop
```

Drop all TCP packets that do not contain MSS nor WScale.

```
$sudo vppctl mmb add ip-proto tcp tcp-opt 22 drop
```

Drop all TCP packets that contain option 22

2.2 stateful polices