MACsec: Encryption for the wired LAN

Sabrina Dubroca
Networking Services Team, Red Hat
Zurich, Switzerland
sd@queasysnail.net
sdubroca@redhat.com

Abstract

MACsec is an IEEE standard for security in wired ethernet LANs. MACsec offers authenticity and integrity, as well as optional encryption of the layer 2 payload. As a layer 2 specification, it provides these guarantees for all traffic in a LAN, including ARP or neighbour discovery, VLAN headers, or LACP. MACsec can be used on its own, or be combined with 802.1X to provide authentication, secure key distribution, and participant discovery. This paper gives an overview MACsec and its architecture, describes the proposed implementation submitted for inclusion in the Linux kernel, presents some use cases and configuration examples with iproute2, and lists some future work both in the kernel and in userspace.

Keywords

MACsec, L2, encryption, security, virtual device

Introduction

MACsec is an IEEE standard [1] that defines a protocol providing security for wired ethernet LANs. MACsec offers two protection modes: integrity only, or integrity with confidentiality. In the first case, packets are transmitted in the clear but all the other guarantees of MACsec (protection against tampering, replay protection) are provided. In the second case, MACsec uses authenticated encryption to protect the data.

MACsec uses GCM AES with 128 bit keys by default. This cipher suite provides Authenticated Encryption with Additional Data (AEAD), which allows to authenticate and ensure integrity-protection of an entire packet, including all its headers. Part of the payload can be encrypted as neeeded, while the headers necessary for delivery of the packet are transmitted as cleartext (but under integrity protection). Furthermore, it can also be used for integrity-only protection, by passing the entire packet to the algorithm as additional data. An extension to the standard allows 256 bit keys with GCM AES [3].

MACsec is designed to be used with the MKA extension to 802.1X (MACsec Key Agreement protocol) [2], which provides channel attribution and key distribution to the nodes, but can also be used with static keys getting fed manually by an administrator, for example using iproute2.

MACsec Architecture

In MACsec terminology, a “Security Entity” (SecY) is an instance of the MACsec implementation within a node.

MACsec defines unidirectional “secure channels” (SC) that allow transmission from one node to one or more others. Communication on a channel is done over a succession of “secure associations” (SA), each using a specific key. Secure associations are identified by their “association number”, in the range $[0, 3]$.

Keys are assigned to individual secure associations. A 32-bit packet number is associated with each secure association and serves two purposes:

– As part of the Initialization Vector, to ensure that every packet encrypted with the same key uses a different IV;

– Replay protection, the receiving host can check the packet number of the incoming packet against its receive window.

When the packet number would wrap, the secure association is retired. The administrator or a management tool should set up a new secure association before this happens – by monitoring the evolution of the packet number – to allow switching seamlessly between the old and new association. The same association number can later be reused, after allocating a new key for it.

Packet format

Figure 1 shows a typical packet transmitted over an ethernet LAN.

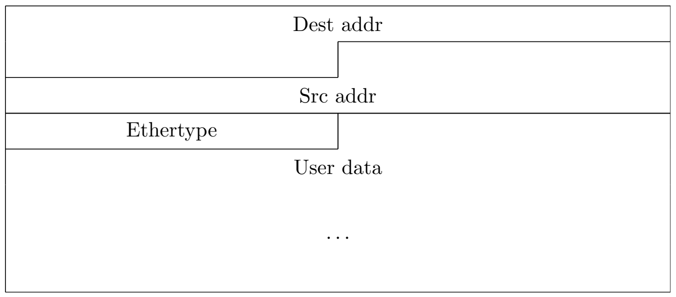

Figure 1: Unprotected frame

When a packet goes through a MACsec device, a SecTAG header is prepended, the ethertype is changed to the MACsec ethertype (0x88e5), and the ICV (Integrity Check Value) computed using the cipher suite over the entire packet (including the SecTAG itself, and the destination and source MAC addresses) is finally appended. The ethertype of the original packet is part of the protected payload, as shown in figure 2.

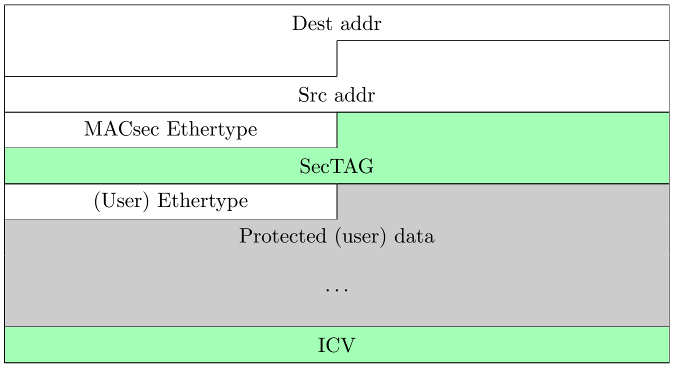

Figure 2: MACsec protected (unencrypted) frame

Optionally, when encryption is enabled, the source and destination addresses, as well as the SecTAG, remain protected by the ICV (as part of the "additional data" passed to the cipher suite), and the rest of the original packet – starting from its original ethertype, and the IP headers and payload for example – are encrypted.

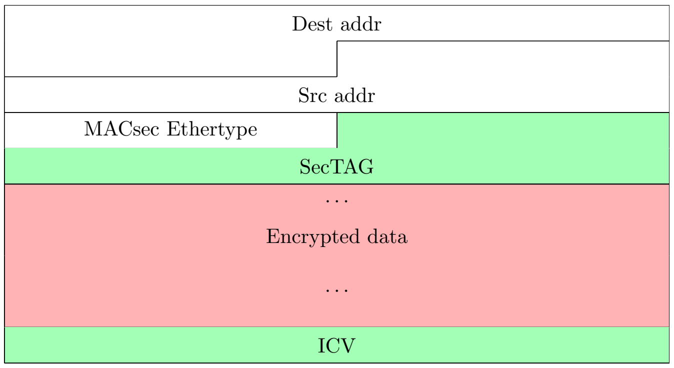

Figure 3: Encrypted MACsec frame

SecTAG format The SecTAG (fig 4) is the additional header that MACsec requires.

AN association number (SA identifier, 2 bits)

SL short length, non-zero for frame lengths under 64B

TCI tag control information (fig 5)

- **ES** End Station[1]
- **SC** SCI present
- **SCB** Single Copy Broadcast[1]
- **E** Encrypted payload
- **C** Changed text

SCI secure channel identifier, 64 bits

- 48 bits "system identifier" (MAC address)
- 16 bits "port number"

Packet format with VLAN When using MACsec to protect a VLAN, the VLAN tag is part of the encrypted payload, as shown on figure 6.

Packet handling

On transmit The SecTAG is first pushed at the start of the packet. Then, the ICV is computed over the entire packet, and the payload is optionally encrypted at the same time. The ICV is then appended at the end of the packet, which is finally passed down to the network.

On receive The format of the packet and SecTAG is first checked. Then, if replay protection is enabled, a first check of the packet number with the receive window is performed.[2] The cryptographic signature is then verified, and the data is decrypted. MACsec offers three different validation modes for incoming packets:

– *strict:* all non-protected, invalid, or impossible to verify (because there is no receive channel that matches the SCI for the packet) frames are dropped

– *check:* these frames are counted as "invalid" and accepted[3]

– *disabled:* all incoming frames are accepted[4]

A second replay protection check is then performed. The MACsec-specific parts of the packet (ICV, SecTAG) are then stripped, and the packet is finally passed up to the network stack for processing.

Implementation in the Linux kernel

A SecY appears as a (virtual) network device linked to a parent device, similarly to macvlan devices. The parent device sees only the raw packets, ie the MACsec-protected packets for all its slave MACsec devices, as well as all the non-protected traffic (for example, 802.1X). This design is a good

[1]The ES and SCB bits are described in the standard[1] and will not be mentionned in this paper.

[2]From a security point of view, this check is acceptable – eventhough the authenticity of the packet has not been verified at this point – because we only drop the packet without providing any feedback to a potential attacker, thus they cannot infer any timing differences to guess the window. Additionally, this check helps protect against DoS attacks by avoiding the more expensive cryptographic computation on packets that are obviously wrong.

[3]Encrypted frames cannot be accepted if there is no matching channel, because there is no key to decrypt them. Additionally, since MACsec allows administrators to choose the ICV length, only frames using the default ICV length can be processed correctly without a matching receive channel.

[4]Same conditions as for "check" apply

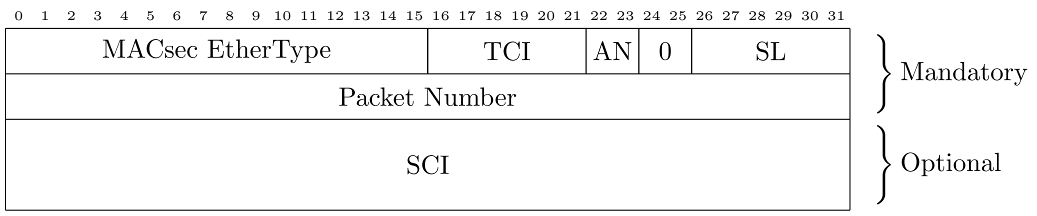

Figure 4: SecTAG format

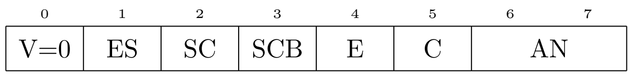

Figure 5: SecTAG TCI and AN format

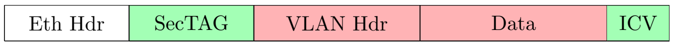

Figure 6: MACsec-protected VLAN frame

match for the uncontrolled and controlled port model defined in the IEEE standards.

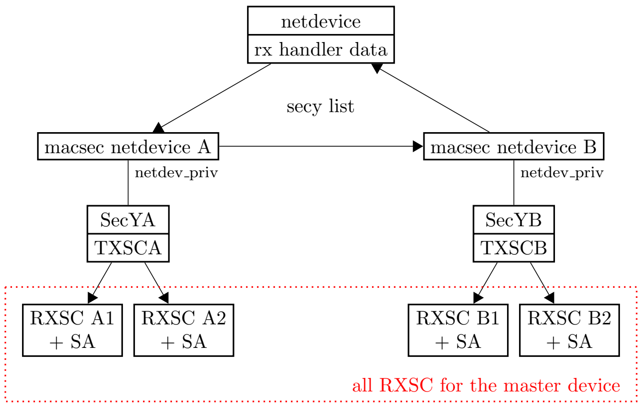

Figure 7: Data structures relationships

Incoming packet processing

Incoming packets on the parent devices are processed through the `rx_handler` infrastructure also used by bonding and macvlan devices.

If the SCI is not explicitly present in the SecTAG, it is reconstructed from the MAC address, using the default port number (0x0001). We can then use this SCI to find the matching receive secure channel among all the receive channels associated with the parent device. The packet – after validation and decryption, as described earlier – is then passed up the networking stack, after setting `skb->dev` to the `net_device` for the SecY corresponding to the receive secure channel we found.

Outgoing packet processing

Each transmit secure channel is associated with exactly one MACsec `net_device` (fig 7), through which packets to be protected using this channel will flow. In the `ndo_start_xmit` method for the MACsec device, the SecTAG is filled and the packet is protected (and optionally encrypted) using the currently active secure association (`encoding_sa`, a per-macsec-device configuration setting). The resulting ICV is appended, and the packet is then passed down to the underlying `net_device`.

Configuration API

The configuration API for MACsec devices is split between rtnetlink and genetlink.

rtnetlink is used to create and setup the `net_device` and the SecY attributes, using the `IFLA_MACSEC_*` attributes with `RTM_NEWLINK` or `RTM_SETLINK`.

The genetlink part of the configuration API is used to set up transmit secure association within a SecY, and the receive channels and associations on a MACsec device. The genetlink API provides clean demultiplexing between different commands.

Use cases

The default use case for MACsec is a standard LAN. With a MACsec-capable switch, one could configure MACsec on each host and the corresponding switch port. With a dumb switch, one could enable MACsec on each host, so that the entire LAN traffic is protected and the switch forwards the MACsec-protected frames.

A host can also be configured with multiple secure channels, so that host H2 would not be able to decrypt the communications between H1 and H4 (or H1 for communications between H2 and H4. See figure 8. An example configuration is given in listing 1).

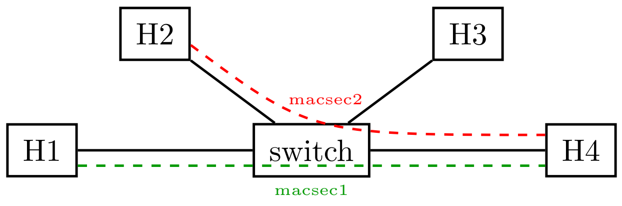

Figure 8: LAN setup with multiple channels

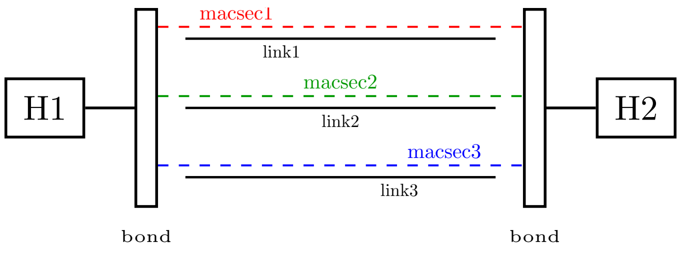

Figure 9: Example bond+MACsec setup

Listing 1: LAN configuration

```
# on H4: channel to H1
ip link add link eth0 macsec0 type macsec
ip macsec add macsec0 tx sa 0 on pn 100 \
    key 1 $KEY_1
ip macsec add macsec0 rx address $H1_ADDR \
    port 1
ip macsec add macsec0 rx address $H1_ADDR \
    port 1 sa 0 pn 100 on key 0 $KEY_0

# on H4: channel to H2
ip link add link eth0 macsec1 type macsec \
    port 2
ip macsec add macsec1 tx sa 0 on pn 400 \
    key 2 $KEY_2
ip macsec add macsec1 rx address $H2_ADDR \
    port 1
ip macsec add macsec1 rx address $H2_ADDR \
    port 1 sa 0 pn 100 on key 3 $KEY_3

# on H1
ip link add link eth0 macsec0 type macsec
ip macsec add macsec0 tx sa 0 on pn 100 \
    key 0 $KEY_0
ip macsec add macsec0 rx address $H4_ADDR \
    port 1
ip macsec add macsec0 rx address $H4_ADDR \
    port 1 sa 0 pn 100 on key 1 $KEY_1

# on H2
ip link add link eth0 macsec0 type macsec
ip macsec add macsec0 tx sa 0 on pn 100 \
    key 3 $KEY_3
ip macsec add macsec0 rx address $H4_ADDR \
    port 2
ip macsec add macsec0 rx address $H4_ADDR \
    port 2 sa 0 pn 400 on key 2 $KEY_2
```

Listing 2 shows examples of some additional iproute2 commands to configure a MACsec device. The first of these commands enables changing the active transmit secure association, which needs to be done before the packet number for the current association overflows. The second command allows enabling encryption for packets transmitted on a MACsec channel. Finally, the last of these commands enables replay protection, with a 128-packets window, so that incoming packets on any receive secure channel for this device, with a PN smaller than `last_packet_seen - 128`, will be silently dropped.

Listing 2: Some MACsec options

```
# changing the current active TXSA
ip link set macsec0 type macsec encoding 2

# enabling encryption
ip link set macsec0 type macsec encrypt on

# enabling replay protection
ip link set macsec0 type macsec \
    replay on window 128
```

Link aggregation and MACsec

MACsec can be used with link aggregation devices such as bonding. Secure channels are configured independently on each underlying link, and the MACsec devices are then enslaved in the bond – instead of adding the links themselves to the bond (figure 9, configuration in listing 3).

Listing 3: bond configuration

```
# modprobe bonding max_bonds=0
ip link add bond0 type bond [...]
ip link set bond0 up

# Set up MACsec on each bonded link
ip link add link eth0 macsec0 type macsec ...
# setup SA and RX on macsec0 like before
ip link add link eth1 macsec1 type macsec ...
# setup SA and RX on macsec1 like before

# Add the MACsec devices to the bond
ip link set macsec0 master bond0
ip link set macsec1 master bond0
```

MACsec over VXLAN

MACsec only needs an ethernet header, so we can configure MACsec over VXLAN links, as described in figure 10 (see also listing 4).

Listing 4: MACsec over VXLAN configuration

```
ip link add link type vxlan\
     id 10 group 239.0.0.10 ttl 5 dev eth0
ip link add link vxlan0 macsec0 type macsec

# setup SA and RX on macsec0 like before
```

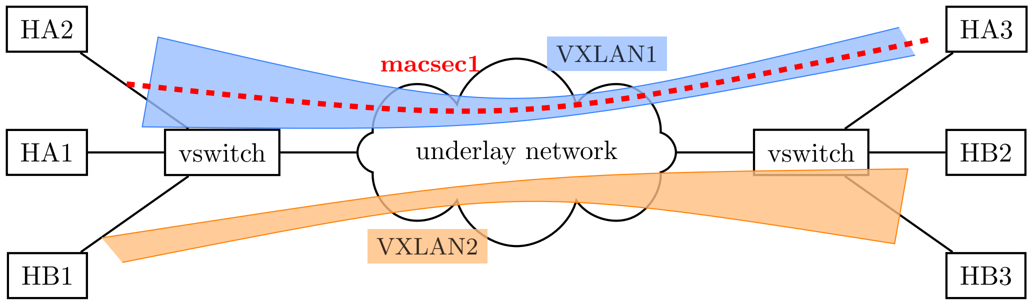

Figure 10: Example VXLAN+MACsec setup

Future work

In the kernel

In the current implementation of MACsec, some optional features defined in the IEEE standards are not yet supported:

– *confidentiality offset:* the first 30 bytes of the packet are only integrity protected. Currently, we can either encrypt the entire payload, or leave it completely in the clear. This would allow for example to transmit the IP header as cleartext through the network.

– *additional ciphersuite:* GCM-AES-256, as defined in [3].

Additionally, some Intel ixgbe-based NICs have hardware support for MACsec, which would allow to enable a single Secure Channel to transmit at the full line rate of these NICs. This would be required for performance-sensitive applications.

The current performance of MACsec is quite limited, and future improvements could allow better throughput.

In userspace

The only configuration tool currently provided is iproute2, with support only for static configuration of channels, associations, and keys. Future work would enable configuration of MACsec via other tools such as NetworkManager.

`wpa_supplicant` already has MACsec Key Agreement (MKA) support [4][5][6], but there is at the moment no driver to configure the kernel over netlink.

References

[1] IEEE. 2006. IEEE standard for local and metropolitan area networks - media access control (mac) security. *IEEE Std. 802.1AE-2006*.

[2] IEEE. 2010. IEEE standard for local and metropolitan area networks - port-based network access control. *IEEE Std. 802.1X-2010*.

[3] IEEE. 2011. IEEE standard for local and metropolitan area networks - media access control (mac) security, ammendment 1: Galois counter mode-advanced encryption standard-256 (gcm-aes-256) cipher suite. *IEEE Std. 802.1AEbn-2011*.

[4] Wang, H. 2014a. MACsec: Add drivers_ops. hostap commit 7baec808efb5 `http://w1.fi/cgit/hostap/commit/?id=7baec808efb5`.

[5] Wang, H. 2014b. MACsec: Add PAE implementation. hostap commit 887d9d01abc7 `http://w1.fi/cgit/hostap/commit/?id=887d9d01abc7`.

[6] Wang, H. 2014c. MACsec: wpa_supplicant integration. hostap commit dd10abccc86d `http://w1.fi/cgit/hostap/commit/?id=dd10abccc86d`.