I2NSF
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019
July 2, 2018

R. Marin-Lopez
G. Lopez-Millan
University of Murcia

Software-Defined Networking (SDN)-based IPsec Flow Protection
draft-ietf-i2nsf-sdn-ipsec-flow-protection-02

Abstract

This document describes how providing IPsec-based flow protection by means of a Software-Defined Network (SDN) controller (aka. Security Controller) and establishes the requirements to support this service. It considers two main well-known scenarios in IPsec: (i) gateway-to-gateway and (ii) host-to-host. The SDN-based service described in this document allows the distribution and monitoring of IPsec information from a Security Controller to one or several flow-based Network Security Function (NSF). The NSFs implement IPsec to protect data traffic between network resources with IPsec.

The document focuses in the NSF Facing Interface by providing models for Configuration and State data model required to allow the Security Controller to configure the IPsec databases (SPD, SAD, PAD) and IKEv2 to establish security associations with a reduced intervention of the network administrator.

Copyright Notice

Table of Contents

1. Introduction

Software-Defined Networking (SDN) is an architecture that enables users to directly program, orchestrate, control and manage network resources through software. SDN paradigm relocates the control of network resources to a dedicated network element, namely SDN controller. The SDN controller manages and configures the distributed network resources and provides an abstracted view of the network resources to the SDN applications. The SDN application can customize and automate the operations (including management) of the abstracted network resources in a programmable manner via this interface [RFC7149][ITU-T.Y.3300] [ONF-SDN-Architecture][ONF-OpenFlow].

Typically, traditional IPsec VPN concentrators and, in general, entities (i.e. hosts or security gateways) supporting IKE/IPsec, must be configured directly by the administrator. This makes the IPsec security association (SA) management difficult and generates a lack of flexibility, specially if the number of security policies and SAs to handle is high. With the growth of SDN-based scenarios where network resources are deployed in an autonomous manner, a mechanism to manage IPsec SAs according to the SDN architecture becomes more relevant. Thus, the SDN-based service described in this document will autonomously deal with IPsec SAs management.

An example of usage can be the notion of Software Defined WAN (SD-WAN), SDN extension providing a software abstraction to create secure network overlays over traditional WAN and branch networks. SD-WAN is based on IPsec as underlying security protocol and aims to provide flexible, automated, fast deployment and on-demand security network services.

IPsec architecture [RFC4301] defines a clear separation between the processing to provide security services to IP packets and the key management procedures to establish the IPsec security associations. In this document, we define a service where the key management procedures can be carried by an external entity: the Security Controller.

First, this document exposes the requirements to support the protection of data flows using IPsec [RFC4301]. We have considered two general cases:

1) The Network Security Function (NSF) implements the Internet Key
    Exchange (IKE) protocol and the IPsec databases: the Security

Policy Database (SPD), the Security Association Database (SAD) and the Peer Authorization Database (PAD). The Security Controller is in charge of provisioning the NSF with the required information to IKE, the SPD and the PAD.

2) The NSF only implements the IPsec databases (no IKE implementation). The Security Controller will provide the required parameters to create valid entries in the SPD and the SAD into the NSF. Therefore, the NSF will have only support for IPsec while automated key management functionality is moved to the controller.

In both cases, an interface/protocol is required to carry out this provisioning in a secure manner between the Security Controller and the NSF. In particular, Case 1 requires the provision of SPD and PAD entries and the IKE credential and information related with the IKE negotiation (e.g. IKE_SA_INIT); and Case 2 requires the management of SPD and SAD entries. Based on YANG models in [netconf-vpn] and [I-D.tran-ipsecme-yang], RFC 4301 [RFC4301] and RFC 7296 [RFC7296] this document defines the required interfaces with a YANG model for configuration and state data for IKE, PAD, SPD and SAD Appendix A.

This document considers two typical scenarios to manage autonomously IPsec SAs: gateway-to-gateway and host-to-host [RFC6071]. The analysis of the host-to-gateway (roadwarrior) scenario is TBD. In these cases, host or gateways or both may act as NSFs. Finally, it also discusses the situation where two NSFs are under the control of two different Security Controllers.

NOTE: This work pays attention to the challenge "Lack of Mechanism for Dynamic Key Distribution to NSFs" defined in [I-D.ietf-i2nsf-problem-and-use-cases] in the particular case of the establishment and management of IPsec SAs. In fact, this I-D could be considered as a proper use case for this particular challenge in [I-D.ietf-i2nsf-problem-and-use-cases].

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119]. When these words appear in lower case, they have their natural language meaning.

3. Terminology

This document uses the terminology described in [RFC7149], [RFC4301], [ITU-T.Y.3300], [ONF-SDN-Architecture], [ONF-OpenFlow], [ITU-T.X.1252], [ITU-T.X.800] and [I-D.ietf-i2nsf-terminology]. In addition, the following terms are defined below:

- Software-Defined Networking. A set of techniques enabling to directly program, orchestrate, control, and manage network resources, which facilitates the design, delivery and operation of network services in a dynamic and scalable manner [ITU-T.Y.3300].

- Flow/Data Flow. Set of network packets sharing a set of characteristics, for example IP dst/src values or QoS parameters.

- Security Controller. A Controller is a management Component that contains control plane functions to manage and facilitate information sharing, as well as execute security functions. In the context of this document, it provides IPsec management information.

- Network Security Function (NSF). Software that provides a set of security-related services.

- Flow-based NSF. A NSF that inspects network flows according to a set of policies intended for enforcing security properties. The NSFs considered in this document falls into this classification.

- Flow-based Protection Policy. The set of rules defining the conditions under which a data flow MUST be protected with IPsec, and the rules that MUST be applied to the specific flow.

- Internet Key Exchange (IKE) v2 Protocol to establish IPsec Security Associations (SAs). It requires information about the required authentication method (i.e. preshared keys), DH groups, modes and algorithms for IKE SA negotiation, etc.

- Security Policy Database (SPD). It includes information about IPsec policies direction (in, out), local and remote addresses, inbound and outboud SAs, etc.

- Security Associations Database (SAD). It includes information about IPsec SAs, such as SPI, destination addresses, authentication and encryption algorithms and keys to protect IP flow.

- o Peer Authorization Database (PAD). It provides the link between the SPD and a security association management protocol such as IKE or the SDN-based solution described in this document.

4. Objectives

- o To describe the architecture for the SDN-based IPsec management, which implements a security service to allow the establishment and management of IPsec security associations from a central point to protect specific data flows.

- o To define the interfaces required to manage and monitor the IPsec Security Associations in the NSF from a Security Controller. YANG models are defined for configuration and state data for IPsec management.

5. SDN-based IPsec management description

As mentioned in Section 1, two cases are considered:

5.1. Case 1: IKE/IPsec in the NSF

In this case the NSF ships an IKEv2 implementation besides the IPsec support. The Security Controller is in charge of managing and applying SPD and PAD entries (deriving and delivering IKE Credentials such as a pre-shared key, certificates, etc.), and applying other IKE configuration parameters (e.g. IKE_SA_INIT algorithms) to the NSF for the IKE negotiation.

With these entries, the IKEv2 implementation can operate to establish the IPsec SAs. The application (administrator) establishes the IPsec requirements and information about the end points information (through the Client Facing Interface), and the Security Controller translates those requirements into IKE, SPD and PAD entries that will be installed into the NSF (through the NSF Facing Interface). With that information, the NSF can just run IKEv2 to establish the required IPsec SA (when the data flow needs protection). Figure 1 shows the different layers and corresponding functionality.

```
                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                |IPsec Management/Orchestration Application | Client or
                |          I2NSF Client                     | App Gateway
                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                        |    Client Facing Interface
                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       Vendor   |             Application Support           |
       Facing<->+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Security
       Interface| IKE Credential,PAD and SPD entries Distr. | Controller
                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                        |       NSF Facing Interface
                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                |                 I2NSF Agent               |
                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Network
                |   IKE    |      IPsec(SPD,PAD)            | Security
                +-------------------------------------------+ Function
                |         Data Protection and Forwarding    |
                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Case 1: IKE/IPsec in the NSF

5.1.1. Interface Requirements for Case 1

SDN-based IPsec flow protection services provide dynamic and flexible management of IPsec SAs in flow-based NSF. In order to support this capability in case 1, the following interface requirements are to be met:

- o A YANG data model for Configuration data for IKEv2, SPD and PAD.

- o A YANG data model for State data for IKE, SPD, PAD and SAD (NOTE: the SAD entries are created in runtime by IKEv2.)

- o In scenarios where multiple controllers are implicated, SDN-based IPsec management services may require a mechanism to discover which Security Controller is managing a specific NSF. Moreover, an east-west interface is required to exchange IPsec-related information.

5.2. Case 2: IPsec (no IKEv2) in the NSF

In this case, the NSF does not deploy IKEv2 and, therefore, the Security Controller has to perform the management of IPsec SAs by populating and monitoring the SPD and the SAD.

```
              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
              |   IPsec Management  Application         | Client or
              |               I2NSF Client              | App Gateway
              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                        |   Client Facing Interface
              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        Vendor|             Application Support         |
     Facing<->+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Security
     Interface|      SPD, SAD and PAD Entries Distr.    | Controller
              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                        |   NSF Facing Interface
              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
              |              I2NSF Agent                | Network
              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Security
              |            IPsec (SPD,SAD)              | Function (NSF)
              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
              |     Data Protection and Forwarding      |
              +-----------------------------------------+
```

Figure 2: Case 2: IPsec (no IKE) in the NSF

As shown in Figure 2, applications for flow protection run on the top of the Security Controller. When an administrator enforces flow-based protection policies through the Client Facing Interface, the Security Controller translates those requirements into SPD and SAD entries, which are installed in the NSF. PAD entries are not required since there is no IKEv2 in the NSF.

5.2.1. Interface Requirements for Case 2

In order to support case 2, the following requirements are to be met:

- A YANG data model for Configuration data for SPD and SAD.

- A YANG data model for State data for SPD and SAD.

- In scenarios where multiple controllers are implicated, SDN-based IPsec management services may require a mechanism to discover which Security Controller is managing a specific NSF. Moreover, an east-west interface is required to exchange IPsec-related information.

5.3. Case 1 vs Case 2

Case 1 MAY be easier to deploy than Case 2 because current gateways typically have an IKEv2/IPsec implementation. Moreover hosts can install easily an IKE implementation. As downside, the NSF needs more resources to hold IKEv2. Moreover, the IKEv2 implementation

needs to implement an interface so that the I2NSF Agent can interact with them.

Alternatively, Case 2 allows lighter NSFs (no IKEv2 implementation), which benefits the deployment in constrained NSFs. Moreover, IKEv2 does not need to be performed in gateway-to-gateway and host-to-host scenarios under the same Security Controller (see Section 7.1). On the contrary, the overload of creating fresh IPsec SAs is shifted to the Security Controller since IKEv2 is not in the NSF. As a consequence, this may result in a more complex implementation in the controller side.

5.3.1. Rekeying process

For case 1, the rekeying process is carried out by IKEv2, following the configuration defined in the SPD.

For case 2, the Security Controller needs to take care of the rekeying process. When the IPsec SA is going to expire (e.g. IPsec SA soft lifetime), it has to create a new IPsec SA and remove the old one. This rekeying process starts when the Security Controller receives a sadb_expire notification or it decides so, based on lifetime state data obtained from the NSF.

To explain the rekeying process between two IPsec peers A and B, let assume that SPIa1 identifies the inbound SA in A and SPIb1 the inbound SA in B.

1. The Security Controller chooses two random values as SPI for the new inbound SAs: for example, SPIa2 for A and SPIb2 for B. These numbers MUST not be in conflict with any IPsec SA in A or B. Then,the Security Controller creates an inbound SA with SPIa2 in A and another inbound SA in B with SPIb2. It can send this information simultenously to A and B.

2. Once the Security Controller receives confirmation from A and B, inbound SA are correctly installed. Then it proceeds to send in parallel to A and B the outbound SAs: it sends the outbound SA to A with SPIb2 and the outbound SA to B with SPIa2. At this point the new IPsec SA is ready.

3. The Security Controller deletes the old IPsec SAs from A (inbound SPIa1 and outbound SPIb1) and B (outbound SPIa1 and inbound SPIb1) in parallel.

5.3.2. NSF state loss

If one of the NSF restarts, it may lose part or all the IPsec state (affected NSF). By default, the Security Controller can assume that all the state has been lost and therefore it will have to send IKEv2, SPD and PAD information to the NSF in case 1 and SPD and SAD information in case 2.

In both cases, the Security Controller MUST be aware of the affected NSF (e.g. the NETCONF/TCP connection is broken with the affected NSF, it is receiving bad_spi notification from a particular NSF, etc...). Moreover, the Security Controller MUST have a register about all the NSFs that have IPsec SAs with the affected NSF. Therefore, it knows the affected IPsec SAs.

In Case 1, the Security Controller will configure the affected NSF with the new IKEv2, SPD and PAD information. It has also to send new parameters (e.g. a new fresh PSK) to the NSFs which have IKEv2 SAs and IPsec SAs with the affected NSF. It can also instruct the affected NSF to send IKEv2 INITIAL_CONTACT (It is TBD in the model). Finally, the Security Controller will instruct the affected NSF to start the IKEv2 negotiation with the new configuration.

In Case 2, if the Security Controller detects that a NSF has lost the IPsec SAs (e.g. it reboots) it will follow similar steps to rekey: the steps 1 and 2 remain equal but the step 3 will be slightly different. For example, if we assume that NSF B has lost its state, the Security Controller MUST only delete the old IPsec SAs from A in step 3.

Nevertheless other more optimized options can be considered (e.g. making iKEv2 configuration permanent between reboots).

5.3.3. NAT Traversal

In case 1, IKEv2 already owns a mechanism to detect whether some of the peers or both are behind a NAT. If there is a NAT network configured between two peers, it is required to activate the usage of UDP or TCP/TLS encapsulation of ESP packets ([RFC3948], [RFC8229])

On the contrary, case 2 does not have any protocol in the NSFs to detect whether they are behind a NAT or not. However, the SDN paradigm generally assumes the Security Controller has a view of the network it controls. This view is built either requesting information to the NSFs under its control, or because these NSFs inform to the Security Controller. Based on this information, the Security Controller can guess if there is a NAT configured between two hosts, apply the required policies to both NSFs besides

activating the usage of UDP or TCP/TLS encapsulation of ESP packets ([RFC3948], [RFC8229]).

For example, the Security Controller could directly request the NSF for specific data such as networking configuration, NAT support, etc. Protocols such as NETCONF or SNMP can be used here. For example, RFC 7317 [RFC7317] provides a YANG data model for system management or [I-D.sivakumar-yang-nat] a data model for NAT management.

6. YANG configuration data models

In order to support case 1 and case 2 we have modelled the different parameters and values that must be configured to manage IPsec SAs. Specifically, case 1 requires modelling IKEv2, SPD and PAD while case 2 requires models for the SPD and SAD. A single YANG file represents both cases though some part of the models are selectively activated depending a feature defined in the YANG file. For example, the IKE configuration is not enabled in case 2.

In the following, we summarize, by using a tree representation, the different configuration and state data models. The complete YANG configuration data model is in Appendix A

6.1. Security Policy Database (SPD) Model

The definition of this model has been extracted from the specification in section 4.4.1 and Appendix D in [RFC4301]

```
                        +--rw spd
                        |  +--rw spd-entry* [rule-number]
                        |     +--rw rule-number              uint64
                        |     +--rw priority?                uint32
                        |     +--rw names* [name]
                        |     |  +--rw name-type?   ipsec-spd-name
                        |     |  +--rw name         string
                        |     +--rw condition
                        |     |  +--rw traffic-selector-list* [ts-number]
                        |     |     +--rw ts-number             uint32
                        |     |     +--rw direction?            ipsec-traffic-
direction
                        |     |     +--rw local-addresses* [start end]
                        |     |     |  +--rw start    inet:ip-address
                        |     |     |  +--rw end      inet:ip-address
                        |     |     +--rw remote-addresses* [start end]
                        |     |     |  +--rw start    inet:ip-address
                        |     |     |  +--rw end      inet:ip-address
                        |     |     +--rw next-layer-protocol*   ipsec-next-
```

layer-proto

```
                        |     |     +--rw local-ports* [start end]
                        |     |     |  +--rw start    inet:port-number
                        |     |     |  +--rw end      inet:port-number
                        |     |     +--rw remote-ports* [start end]
                        |     |     |  +--rw start    inet:port-number
                        |     |     |  +--rw end      inet:port-number
                        |     |     +--rw selector-priority?     uint32
                        |     +--rw processing-info
                        |     |  +--rw action          ipsec-spd-operation
                        |     |  +--rw ipsec-sa-cfg
                        |     |     +--rw pfp-flag?            boolean
                        |     |     +--rw extSeqNum?           boolean
                        |     |     +--rw seqOverflow?         boolean
                        |     |     +--rw statefulfragCheck?   boolean
                        |     |     +--rw security-protocol?   ipsec-protocol
                        |     |     +--rw mode?                ipsec-mode
                        |     |     +--rw ah-algorithms
                        |     |     |  +--rw ah-algorithm*   integrity-
algorithm-t
                        |     |     +--rw esp-algorithms
                        |     |     |  +--rw authentication*   integrity-
algorithm-t
                        |     |     |  +--rw encryption*       encryption-
algorithm-t
                        |     |     +--rw tunnel
                        |     |        +--rw local?          inet:ip-address
                        |     |        +--rw remote?         inet:ip-address
                        |     |        +--rw bypass-df?      boolean
                        |     |        +--rw bypass-dscp?    boolean
                        |     |        +--rw dscp-mapping?   yang:hex-string
                        |     |        +--rw ecn?            boolean
                        |     +--rw spd-mark
                        |     |  +--rw mark?   uint32
                        |     |  +--rw mask?   yang:hex-string
                        |     +--rw spd-lifetime-hard
                        |     |  +--rw added?     uint64
                        |     |  +--rw used?      uint64
                        |     |  +--rw bytes?     uint32
                        |     |  +--rw packets?   uint32
                        |     |  +--rw action?    lifetime-action
                        |     +--rw spd-lifetime-soft
                        |     |  +--rw added?     uint64
                        |     |  +--rw used?      uint64
                        |     |  +--rw bytes?     uint32
                        |     |  +--rw packets?   uint32
                        |     |  +--rw action?    lifetime-action
                        |     +--ro spd-lifetime-current
                        |        +--ro added?     uint64
```

```
                        |        +--ro used?      uint64
                        |        +--ro bytes?     uint32
                        |        +--ro packets?   uint32
```

6.2. Security Association Database (SAD) Model

The definition of this model has been extracted from the specification in section 4.4.2 in [RFC4301]

```
                        +--rw sad
                        |  +--rw sad-entry* [spi]
                        |     +--rw spi                          ipsec-spi
                        |     +--rw seq-number?                  uint64
                        |     +--rw seq-number-overflow-flag?    boolean
                        |     +--rw anti-replay-window?          uint16
                        |     +--rw rule-number?                 uint32
                        |     +--rw local-addresses* [start end]
                        |     |  +--rw start    inet:ip-address
                        |     |  +--rw end      inet:ip-address
                        |     +--rw remote-addresses* [start end]
                        |     |  +--rw start    inet:ip-address
                        |     |  +--rw end      inet:ip-address
                        |     +--rw next-layer-protocol*         ipsec-next-
layer-proto
                        |     +--rw local-ports* [start end]
                        |     |  +--rw start    inet:port-number
                        |     |  +--rw end      inet:port-number
                        |     +--rw remote-ports* [start end]
                        |     |  +--rw start    inet:port-number
                        |     |  +--rw end      inet:port-number
                        |     +--rw security-protocol?           ipsec-protocol
                        |     +--rw ah-sa
                        |     |  +--rw integrity
                        |     |     +--rw integrity-algorithm?    integrity-
algorithm-t
                        |     |     +--rw key?                    string
                        |     +--rw esp-sa
                        |     |  +--rw encryption
                        |     |  |  +--rw encryption-algorithm?    encryption-
algorithm-t
                        |     |  |  +--rw key?                     string
                        |     |  |  +--rw iv?                      string
                        |     |  +--rw integrity
                        |     |  |  +--rw integrity-algorithm?    integrity-
algorithm-t
                        |     |  |  +--rw key?                    string
                        |     |  +--rw combined-enc-intr?    boolean
                        |     +--rw sad-lifetime-hard
                        |     |  +--rw added?     uint64
                        |     |  +--rw used?      uint64
```

```
 |     |  +--rw bytes?     uint32
 |     |  +--rw packets?   uint32
 |     |  +--rw action?    lifetime-action
 |     +--rw sad-lifetime-soft
```

```
     |     |  +--rw added?     uint64
     |     |  +--rw used?      uint64
     |     |  +--rw bytes?     uint32
     |     |  +--rw packets?   uint32
     |     |  +--rw action?    lifetime-action
     |     +--rw mode?                          ipsec-mode
     |     +--rw statefulfragCheck?             boolean
     |     +--rw dscp?                          yang:hex-string
     |     +--rw path-mtu?                      uint16
     |     +--rw tunnel
     |     |  +--rw local?          inet:ip-address
     |     |  +--rw remote?         inet:ip-address
     |     |  +--rw bypass-df?      boolean
     |     |  +--rw bypass-dscp?    boolean
     |     |  +--rw dscp-mapping?   yang:hex-string
     |     |  +--rw ecn?            boolean
     |     +--rw encap
     |     |  +--rw espencap?   esp-encap
     |     |  +--rw sport?      inet:port-number
     |     |  +--rw dport?      inet:port-number
     |     |  +--rw oaddr?      inet:ip-address
     |     +--ro sad-lifetime-current
     |     |  +--ro added?     uint64
     |     |  +--ro used?      uint64
     |     |  +--ro bytes?     uint32
     |     |  +--ro packets?   uint32
     |     +--ro state?                         sa-state
     |     +--ro stats
     |     |  +--ro replay-window?   uint32
     |     |  +--ro replay?          uint32
     |     |  +--ro failed?          uint32
     |     +--ro replay_state
     |     |  +--ro seq?      uint32
     |     |  +--ro oseq?     uint32
     |     |  +--ro bitmap?   uint32
     |     +--ro replay_state_esn
     |        +--ro bmp-len?         uint32
     |        +--ro oseq?            uint32
     |        +--ro oseq-hi?         uint32
     |        +--ro seq-hi?          uint32
     |        +--ro replay-window?   uint32
     |        +--ro bmp*             uint32
```

```
rpcs:
+---x sadb_register
    +---w input
    |  +---w base-list* [version]
    |     +---w version        string
    |     +---w msg_type?      sadb-msg-type
    |     +---w msg_satype?    sadb-msg-satype
    |     +---w msg_seq?       uint32
    +--ro output
    +--ro base-list* [version]
       |  +--ro version        string
       |  +--ro msg_type?      sadb-msg-type
       |  +--ro msg_satype?    sadb-msg-satype
       |  +--ro msg_seq?       uint32
       +--ro algorithm-supported*
          +--ro authentication
          |  +--ro name?        integrity-algorithm-t
          |  +--ro ivlen?       uint8
          |  +--ro min-bits?    uint16
          |  +--ro max-bits?    uint16
          +--ro encryption
             +--ro name?        encryption-algorithm-t
             +--ro ivlen?       uint8
             +--ro min-bits?    uint16
             +--ro max-bits?    uint16

notifications:
+---n spdb_expire
|  +--ro index?   uint64
+---n sadb_acquire
|  +--ro base-list* [version]
|     +--ro version        string
|     +--ro msg_type?      sadb-msg-type
|     +--ro msg_satype?    sadb-msg-satype
|     +--ro msg_seq?       uint32
+---n sadb_expire
|  +--ro base-list* [version]
|  |  +--ro version        string
|  |  +--ro msg_type?      sadb-msg-type
|  |  +--ro msg_satype?    sadb-msg-satype
|  |  +--ro msg_seq?       uint32
|  +--ro spi?                         ipsec-spi
|  +--ro anti-replay-window?          uint16
|  +--ro state?                       sa-state
|  +--ro encryption-algorithm?        encryption-algorithm-t
|  +--ro authentication-algorithm?    integrity-algorithm-t
|  +--ro sad-lifetime-hard
|  |  +--ro added?     uint64
```

```
       |  |  +--ro used?      uint64
       |  |  +--ro bytes?     uint32
       |  |  +--ro packets?   uint32
       |  +--ro sad-lifetime-soft
       |  |  +--ro added?     uint64
       |  |  +--ro used?      uint64
       |  |  +--ro bytes?     uint32
       |  |  +--ro packets?   uint32
       |  +--ro sad-lifetime-current
       |     +--ro added?     uint64
       |     +--ro used?      uint64
       |     +--ro bytes?     uint32
       |     +--ro packets?   uint32
       +---n sadb_bad-spi
          +--ro state    ipsec-spi
```

6.3. Peer Authorization Database (PAD) Model

The definition of this model has been extracted from the specification in section 4.4.3 in [RFC4301] (NOTE: We have observed that many implementations integrate PAD configuration as part of the IKEv2 configuration.)

```
                   +--rw pad {case1}?
                       +--rw pad-entries* [pad-entry-id]
                       +--rw pad-entry-id             uint64
                       +--rw (identity)?
                       |  +--:(ipv4-address)
                       |  |  +--rw ipv4-address?           inet:ipv4-address
                       |  +--:(ipv6-address)
                       |  |  +--rw ipv6-address?           inet:ipv6-address
                       |  +--:(fqdn-string)
                       |  |  +--rw fqdn-string?            inet:domain-name
                       |  +--:(rfc822-address-string)
                       |  |  +--rw rfc822-address-string?   string
                       |  +--:(dnX509)
                       |  |  +--rw dnX509?                 string
                       |  +--:(id_key)
                       |     +--rw id_key?                 string
                       +--rw pad-auth-protocol?      auth-protocol-type
                       +--rw auth-method
                           +--rw auth-m?          auth-method-type
                           +--rw pre-shared
                           |  +--rw secret?   string
                           +--rw rsa-signature
                              +--rw key-data?    string
                              +--rw key-file?    string
                              +--rw ca-data*     string
                              +--rw ca-file?     string
                              +--rw cert-data?   string
                              +--rw cert-file?   string
                              +--rw crl-data?    string
                              +--rw crl-file?    string
```

6.4. Internet Key Exchange (IKEv2) Model

The model related to IKEv2 has been extracted from reading IKEv2 standard in [RFC7296], and observing some open source implementations, such as Strongswan or Libreswan.

```
                        +--rw ikev2 {case1}?
                        |  +--rw ike-connection
                        |  |  +--rw ike-conn-entries* [conn-name]
                        |  |     +--rw conn-name             string
                        |  |     +--rw autostartup           type-autostartup
                        |  |     +--rw nat-traversal?        boolean
```

```
                        |  |     +--rw encap
                        |  |     |  +--rw espencap?   esp-encap
                        |  |     |  +--rw sport?      inet:port-number
                        |  |     |  +--rw dport?      inet:port-number
                        |  |     |  +--rw oaddr?      inet:ip-address
                        |  |     +--rw version?             enumeration
                        |  |     +--rw phase1-lifetime      uint32
                        |  |     +--rw phase1-authalg*      integrity-
algorithm-t
                        |  |     +--rw phase1-encalg*       encryption-
algorithm-t
                        |  |     +--rw combined-enc-intr?   boolean
                        |  |     +--rw dh_group             uint32
                        |  |     +--rw local
                        |  |     |  +--rw (my-identifier-type)?
                        |  |     |  |  +--:(ipv4)
                        |  |     |  |  |  +--rw ipv4?            inet:ipv4-
address
                        |  |     |  |  +--:(ipv6)
                        |  |     |  |  |  +--rw ipv6?            inet:ipv6-
address
                        |  |     |  |  +--:(fqdn)
                        |  |     |  |  |  +--rw fqdn?            inet:domain-
name
                        |  |     |  |  +--:(dn)
                        |  |     |  |  |  +--rw dn?              string
                        |  |     |  |  +--:(user_fqdn)
                        |  |     |  |     +--rw user_fqdn?       string
                        |  |     |  +--rw my-identifier    string
                        |  |     +--rw remote
                        |  |     |  +--rw (my-identifier-type)?
                        |  |     |  |  +--:(ipv4)
                        |  |     |  |  |  +--rw ipv4?            inet:ipv4-
address
                        |  |     |  |  +--:(ipv6)
                        |  |     |  |  |  +--rw ipv6?            inet:ipv6-
address
                        |  |     |  |  +--:(fqdn)
                        |  |     |  |  |  +--rw fqdn?            inet:domain-
name
                        |  |     |  |  +--:(dn)
                        |  |     |  |  |  +--rw dn?              string
                        |  |     |  |  +--:(user_fqdn)
                        |  |     |  |     +--rw user_fqdn?       string
                        |  |     |  +--rw my-identifier    string
                        |  |     +--rw pfs_group*           uint32
                        |  |     +--ro ike-stats
                        |  |        +--ro uptime
```

```
   |  |        |  +--ro running?   yang:date-and-time
   |  |        |  +--ro since?     yang:date-and-time
   |  |        +--ro initiator?       boolean
   |  |        +--ro initiator-spi?   uint64
   |  |        +--ro responder-spi?   uint64
   |  |        +--ro nat-local?       boolean
   |  |        +--ro nat-remote?      boolean
   |  |        +--ro nat-any?         boolean
```

```
                        |  |        +--ro established?     uint64
                        |  |        +--ro rekey-time?      uint64
                        |  |        +--ro reauth-time?     uint64
                        |  |        +--ro child-sas*
                        |  |           +--ro spis
                        |  |              +--ro spi-in?    ipsec-spi
                        |  |              +--ro spi-out?   ipsec-spi
                        |  +--ro number-ike-sas
                        |     +--ro total?       uint32
                        |     +--ro half-open?   uint32
```

7. Use cases examples

This section explains how different traditional configurations, that is, host-to-host and gateway-to-gateway are deployed using this SDN-based IPsec management service. In turn, these configurations will be typical in modern networks where, for example, virtualization will be key.

7.1. Host-to-Host or Gateway-to-gateway under the same controller

```
                      +-----------------------------------------+
                      |            Security Controller          |
                      |                                         |
                   (1)|   +--------------+ (2)+--------------+  |
      Flow-based  ------> |Translate into|--->| South. Prot. |  |
      Security. Pol.  |   |IPsec Policies|    |              |  |
                      |   +--------------+    +--------------+  |
                      |                          |     |        |
                      |                          |     |        |
                      +--------------------------|-----|--------+
                                                 |     |
                                                 | (3) |
                       |-------------------------+     +---|
                       V                                   V
           +----------------------+         +----------------------+
           |    NSF1              |<=======>|   NSF2               |
           |IKEv2/IPsec(SPD/PAD)  |         |IKEv2/IPsec(SPD/PAD)  |
           +----------------------+  (4)    +----------------------+
```

Figure 3: Host-to-Host / Gateway-to-Gateway single controller flow for case 1 .

Figure 3 describes the case 1:

1. The administrator defines general flow-based security policies. The Security Controller looks for the NSFs involved (NSF1 and NSF2).

2. The Security Controller generates IKEv2 credentials for them and translates the policies into SPD and PAD entries.

3. The Security Controller inserts the SPD and PAD entries in both NSF1 and NSF2.

4. The flow is protected with the IPsec SA established with IKEv2.

```
                        +----------------------------------------+
                        |    (1)     Security Controller         |
            Flow-based  |                                        |
            Security -----------|                                |
            Policy      |       V                                |
                        |  +---------------+ (2)+-------------+  |
                        |  |Translate into |--->| South. Prot.|  |
                        |  |IPsec policies |    |             |  |
                        |  +---------------+    +-------------+  |
                        |                         |     |        |
                        |                         |     |        |
                        +-------------------------| --- |--------+
                                                  |     |
                                                  | (3) |
                           |----------------------+     +--|
                           V                               V
                  +------------------+       +------------------+
                  |    NSF1          |<=====>|   NSF2           |
                  |IPsec(SPD/SAD)    |   4)  |IPsec(SPD/SAD)    |
                  +------------------+       +------------------+
```

Figure 4: Host-to-Host / Gateway-to-Gateway single controller flow for case 2.

In case 2, flow-based security policies defined by the administrator are also translated into IPsec SPD entries and inserted into the corresponding NSFs. Besides, fresh SAD entries will be also generated by the Security Controller and enforced in the NSFs. In this case the controller does not run any IKE implementation, and it provides the cryptographic material for the IPsec SAs. These keys will be also distributed securely through the southbound interface. Note that this is possible because both NSFs are managed by the same controller.

Figure 4 describes the case 2, when a data packet needs to be protected in the path between the NSF1 and NSF2:

1. The administrator establishes the flow-based security policies. The Security Controller looks for the involved NSFs.

2. The Security Controller translates the flow-based security policies into IPsec SPD and SAD entries.

3. The Security Controller inserts the these entries in both NSF1 and NSF2 IPsec databases.

4. The flow is protected with the IPsec SA established by the Security Controller.

Both NSFs could be two hosts that exchange traffic and require to establish an end-to-end security association to protect their communications (host-to-host) or two gateways (gateway-to-gateway)), for example, within an enterprise that needs to protect the traffic between, for example, the networks of two branch offices.

Applicability of these configurations appear in current and new networking scenarios. For example, SD-WAN technologies are providing dynamic and on-demand VPN connections between branch offices or between branches and SaaS cloud services. Beside, IaaS services providing virtualization environments are deployments solutions based on IPsec to provide secure channels between virtual instances (Host-to-Host) and providing VPN solutions for virtualized networks (Gateway-to-Gateway).

In general (for case 1 and case 2), this system presents various advantages:

1. It allows to create IPsec SAs among two NSFs, with only the application of more general flow-based security policies at the application layer. Thus, administrators can manage all security associations in a centralized point with an abstracted view of the network;

2. All NSFs deployed after the application of the new policies are NOT manually configured, therefore allowing its deployment in an automated manner.

7.2. Host-to-Host or Gateway-to-gateway under different Security controllers

It is also possible that two NSFs (i.e. NSF1 and NSF2) are under the control of two different Security Controllers. This may happen, for example, when two organizations, namely Enterprise A and Enterprise B, have their headquarters interconnected through a WAN connection

and they both have deployed a SDN-based architecture to provide connectivity to all their clients.

```
                +-------------+                 +-------------+
                |             |                 |             |
      Flow-based|   Security  |<===============>|   Security <--Flow-based
      Sec. Pol.--> Controller |        (3)      |  Controller | Sec. Pol.
            (1) |      A      |                 |      B      |   (2)
                +-------------+                 +-------------+
                     |                                 |
                     | (4)                         (4) |
                     V                                 V
          +----------------------+          +----------------------+
          |    NSF1              |<========>|   NSF2               |
          |IKEv2/IPsec(SPD/PAD)  |          |IKEv2/IPsec(SPD/PAD)  |
          +----------------------+  (5)     +----------------------+
```

Figure 5: Different Security Controllers in Case 1

Figure 5 describes case 1 when two Security Controllers are involved in the process.

1. The A's administrator establishes general Flow-based Security Policies in Security Controller A.

2. The B's administrator establishes general Flow-based Security Policies in Security Controller B.

3. The Security Controller A realizes that protection is required between the NSF1 and NSF2, but the NSF2 is under the control of another Security Controller (Security Controller B), so it starts negotiations with the other controller to agree on the IPsec SPD policies and IKEv2 credentials for their respective NSFs. NOTE: This may require extensions in the East/West interface.

4. Then, both Security Controllers enforce the IKEv2 credentials and related parameters and the SPD and PAD entries in their respective NSFs.

5. The flow is protected with the IPsec SAs established with IKEv2 between both NSFs.

```
                +--------------+                  +--------------+
                |              |                  |              |
       Flow-based. --->                           |          <--- Flow-based
       Prot.   |   Security   |<=================>|   Security   |Sec.
       Pol.(1)|  Controller  |        (3)         |  Controller  |Pol. (2)
                |       A      |                  |       B      |
                +--------------+                  +--------------+
                        |                                |
                        | (4)                        (4) |
                        V                                V
                +------------------+      (5)       +------------------+
                |    NSF1          |<==============>|    NSF2          |
                |IPsec(SPD/SAD)    |                | IPsec(SPD/SAD)   |
                +------------------+                +------------------+
```

Figure 6: Different Security Controllers in case 2

Figure 5 describes case 2 when two Security Controllers are involved in the process.

1. The A's administrator establishes general Flow Protection Policies in Security Controller A.

2. The B's administrator establishes general Flow Protection Policies in Security Controller B.

3. The Security Controller A realizes that the flow between NSF1 and NSF2 MUST be protected. Nevertheless, the controller notices that NSF2 is under the control of another Security Controller, so it starts negotiations with the other controller to agree on the IPsec SPD and SAD entries that define the IPsec SAs. NOTE: It would worth evaluating IKEv2 as the protocol for the East/West interface in this case.

4. Once the Security Controllers have agreed on key material and the details of the IPsec SAs, they both enforce this information into their respective NSFs.

5. The flow is protected with the IPsec SAs established by both Security Controllers in their respective NSFs.

8. Implementation notes

At the time of writing this document, we have implemented a proof-of-concept using NETCONF as southbound protocol, and the YANG model described in Appendix A. The netopeer implementation [netopeer] has been used for both case 1 and case 2 using host-to-host and gateway-

to-gateway configuration. For the case 1, we have used Strongswan [strongswan] distribution for the IKE implementation.

Note that the proposed YANG model provides the models for SPD, SAD, PAD and IKE, but, as describe before, only part of them are required depending of the case (1 or 2) been applied. The Security Controller should be able to know the kind of case to be applied in the NSF and to select the corresponding models based on the YANG features defines for each one.

Internally to the NSF, the NETCONF server (that implements the I2NSF Agent) is able to apply the required configuration updating the corresponding NETCONF datastores (running, startup, etc.). Besides, it can deal with the SPD and SAD configuration at kernel level, through different APIs. For example, the IETF RFC 2367 (PF_KEYv2) [RFC2367] provides a generic key management API that can be used not only for IPsec but also for other network security services to manage the IPsec SAD. Besides, as an extension to this API, the document [I-D.pfkey-spd] specifies some PF_KEY extensions to maintain the SPD. This API is accessed using sockets.

An alternative key management API based on Netlink socket API [RFC3549] is used to configure IPsec on the Linux Operating System.

To allow the NETCONF server implementation interacts with the IKE daemon, we have used the Versatile IKE Configuration Interface (VICI) in Strongswan. This allows changes in the IKE part of the configuration data to be applied in the IKE daemon dynamically.

9. Security Considerations

First of all, this document shares all the security issues of SDN that are specified in the "Security Considerations" section of [ITU-T.Y.3300] and [RFC8192]. We have divided this section in two parts to analyze different security considerations for both cases: NSF with IKEv2 (case 1) and NSF without IKEv2 (case 2). In general, the Security Controller, as typically in the SDN paradigm, is a target for different type of attacks. As a consequence, the Security Controller is a key entity in the infrastructure and MUST be protected accordingly. In particular, according to this document, the Security Controller will handle cryptographic material so that the attacker may try to access this information. Although, we can assume this attack will not likely to happen due to the assumed security measurements to protect the Security Controller, it deserves some analysis in the hypothetical the attack occurs. The impact is different depending on the case 1 or case 2.

9.1. Case 1

In this case 1, the Security Controller sends IKE credentials (PSK, public/private keys, certificates, etc...) to the NSFs. The general recommendation is that the Security Controller NEVER stores the IKE credentials after distributing them. Moreover the NSFs MUST NOT allow the reading of these values once they have been applied by the Security Controller (i.e. write only operations). If the attacker has access to the Security Controller during the period of time that key material is generated, it may access to these values. Since these values are used during NSF authentication in IKEv2, it may impersonate the affected NSFs. Several recommendations are important. If PSK authentication is used in IKEv2 is used, immediately after generating and distributing it, the Security Controller should remove it. If raw public keys are used, the Security Controller should remove the associate private key immediately after generating and distributing them to the NSFs. If certificates are used, the NSF may generate the private key and exports the public key for certification in the Security Controller.

9.2. Case 2

In the case 2, the controller sends the IPsec SA information to the SAD that includes the keys for integrity and encryption (when ESP is used). That key material are symmetric keys to protect data traffic. The general recommendation is that the Security Controller NEVER stores the keys after distributing them. Moreover the NSFs MUST NOT allow the reading of these values once they have been applied by the Security Controller (i.e. write only operations). Nevertheless, if the attacker has access to the Security Controller during the period of time that key material is generated, it may access to these values. In other words, it may have access to the key material used in the distributed IPsec SAs and observe the traffic between peers.

10. Acknowledgements

Authors want to thank Sowmini Varadhan, David Carrel, Yoav Nir, Tero Kivinen, Paul Wouters, Graham Bartlett, Sandeep Kampati, Linda Dunbar, Carlos J. Bernardos, Alejandro Perez-Mendez, Fernando Pereniguez-Garcia, Alejandro Abad-Carrascosa, Ignacio Martinez and Ruben Ricart for their valuable comments.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <https://www.rfc-editor.org/info/rfc5226>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <https://www.rfc-editor.org/info/rfc7296>.

11.2. Informative References

[I-D.ietf-i2nsf-framework]
Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-10 (work in progress), November 2017.

[I-D.ietf-i2nsf-problem-and-use-cases]
Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "I2NSF Problem Statement and Use cases", draft-ietf-i2nsf-problem-and-use-cases-16 (work in progress), May 2017.

[I-D.ietf-i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-05 (work in progress), January 2018.

[I-D.jeong-i2nsf-sdn-security-services-05]
Jeong, J., Kim, H., Park, J., Ahn, T., and S. Lee, "Software-Defined Networking Based Security Services using Interface to Network Security Functions", draft-jeong-i2nsf-sdn-security-services-05 (work in progress), July 2016.

[I-D.pfkey-spd]
Sakane, S., "PF_KEY Extensions for IPsec Policy Management in KAME Stack", October 2002.

[I-D.sivakumar-yang-nat]
Sivakumar, S., Boucadair, M., and S. Vinapamula, "YANG Data Model for Network Address Translation (NAT)", draft-sivakumar-yang-nat-07 (work in progress), July 2017.

[I-D.tran-ipsecme-yang]
Tran, K., Wang, H., Nagaraj, V., and X. Chen, "Yang Data Model for Internet Protocol Security (IPsec)", draft-tran-ipsecme-yang-01 (work in progress), June 2015.

[ITU-T.X.1252]
"Baseline Identity Management Terms and Definitions", April 2010.

[ITU-T.X.800]
"Security Architecture for Open Systems Interconnection for CCITT Applications", March 1991.

[ITU-T.Y.3300]
"Recommendation ITU-T Y.3300", June 2014.

[netconf-vpn]
Stefan Wallin, "Tutorial: NETCONF and YANG", January 2014.

[netopeer]
CESNET, CESNET., "NETCONF toolset Netopeer", November 2016.

[ONF-OpenFlow]
ONF, "OpenFlow Switch Specification (Version 1.4.0)", October 2013.

[ONF-SDN-Architecture]
"SDN Architecture", June 2014.

[RFC2367] McDonald, D., Metz, C., and B. Phan, "PF_KEY Key Management API, Version 2", RFC 2367, DOI 10.17487/RFC2367, July 1998, <https://www.rfc-editor.org/info/rfc2367>.

[RFC3549] Salim, J., Khosravi, H., Kleen, A., and A. Kuznetsov, "Linux Netlink as an IP Services Protocol", RFC 3549, DOI 10.17487/RFC3549, July 2003, <https://www.rfc-editor.org/info/rfc3549>.

[RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, DOI 10.17487/RFC3948, January 2005, <https://www.rfc-editor.org/info/rfc3948>.

[RFC6071] Frankel, S. and S. Krishnan, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap", RFC 6071, DOI 10.17487/RFC6071, February 2011, <https://www.rfc-editor.org/info/rfc6071>.

[RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, DOI 10.17487/RFC7149, March 2014, <https://www.rfc-editor.org/info/rfc7149>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <https://www.rfc-editor.org/info/rfc7317>.

[RFC8192] Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "Interface to Network Security Functions (I2NSF): Problem Statement and Use Cases", RFC 8192, DOI 10.17487/RFC8192, July 2017, <https://www.rfc-editor.org/info/rfc8192>.

[RFC8229] Pauly, T., Touati, S., and R. Mantha, "TCP Encapsulation of IKE and IPsec Packets", RFC 8229, DOI 10.17487/RFC8229, August 2017, <https://www.rfc-editor.org/info/rfc8229>.

[strongswan] CESNET, CESNET., "StrongSwan: the OpenSource IPsec-based VPN Solution", April 2017.

Appendix A. Appendix A: YANG model IPsec Configuration data

```
                    <CODE BEGINS> file "ietf-ipsec@2018-06-29.yang"
 module ietf-ipsec {

   namespace "urn:ietf:params:xml:ns:yang:ietf-ipsec";


   prefix "eipsec";

   import ietf-inet-types { prefix inet; }
   import ietf-yang-types { prefix yang; }

   organization "University of Murcia";

   contact
   " Rafael Marin Lopez
   Dept. Information and Communications Engineering (DIIC)
   Faculty of Computer Science-University of Murcia
   30100 Murcia - Spain
   Telf: +34868888501
   e-mail: rafa@um.es

   Gabriel Lopez Millan
   Dept. Information and Communications Engineering (DIIC)
   Faculty of Computer Science-University of Murcia
   30100 Murcia - Spain
   Tel: +34 868888504
   email: gabilm@um.es
   ";

   description "Data model for IPSec";

   revision "2018-06-29" {
      description
      "Revision";
      reference "";
   }

   feature case1 { description "feature case 1: IKE SPD PAD"; } // IKE/IPSec in
the NSFs
   feature case2 { description "feature case 2: SPD SAD"; } // Only IPSec in
the NSFs

   typedef encryption-algorithm-t {
```

```
      type enumeration {
```

```
         enum reserved-0 {description "reserved";}
         enum des-iv4 { description "DES IV 4";}
         enum des {  description "DES"; }
         enum 3des {  description "3DES"; }
         enum rc5 {  description "RC5"; }
         enum idea {  description "IDEA"; }
         enum cast {  description "CAST"; }
         enum blowfish {  description "BlowFish"; }
         enum 3idea {  description "3IDEA"; }
         enum des-iv32 {  description "DES-IV32"; }
         enum reserved-10 {  description "reserved-10"; }
         enum null {  description "NULL"; }
         enum aes-cbc {  description "AES-CBC"; }
         enum aes-ctr {  description "AES-CTR"; }
         enum aes-ccm-8 {  description "AES-CCM-8"; }
         enum aes-ccm-12 {  description "AES-CCM-12"; }
         enum aes-ccm-16 {  description "AES-CCM-16"; }
         enum reserved-17 {  description "reserved-17"; }
         enum aes-gcm-8-icv {  description "AES-GCM-8-ICV"; }
         enum aes-gcm-12-icv { description "AES-GCM-12-ICV"; }
         enum aes-gcm-16-icv {  description "AES-GCM-16-ICV"; }
         enum null-auth-aes-gmac {  description "Null-Auth-AES-GMAC"; }
         enum ieee-p1619-xts-aes {  description "encr-ieee-p1619-xts-aes -&gt;
Reserved for IEEE P1619 XTS-AES.";}
         enum camellia-cbc {  description "CAMELLIA-CBC"; }
         enum camellia-ctr {  description "CAMELLIA.CTR"; }
         enum camellia-ccm-8-icv {  description "CAMELLIA-CCM-8-ICV"; }
         enum camellia-ccm-12-icv {  description "CAMELLIA-CCM-12-ICV"; }
         enum camellia-ccm-16-icv { description "CAMELLIA-CCM-16-ICV"; }
         enum aes-cbc-128 { description "AES-CBC-128"; }
         enum aes-cbc-192 { description "AES-CBC-192"; }
         enum aes-cbc-256 { description "AES-CBC-256"; }
         enum blowfish-128 { description "BlowFish-128"; }
         enum blowfish-192 { description "BlowFish-192"; }
         enum blowfish-256 { description "BlowFish-256"; }
         enum blowfish-448 { description "BlowFish-448"; }
         enum camellia-128 { description "CAMELLIA-128"; }
         enum camellia-192 { description "CAMELLIA-192"; }
         enum camellia-256 { description "CAMELLIA-256"; }
      }
      description "Encryption algorithms -&gt; RFC_5996";
   }

   typedef integrity-algorithm-t {

      type enumeration {
         enum none { description "NONE"; }
         enum hmac-md5-96 { description "HMAC-MD5-96"; }
```

```
         enum hmac-sha1-96 { description "HMAC-SHA1-96"; }
```

```
         enum des-mac { description "DES-MAC"; }
         enum kpdk-md5 {description "KPDK-MD5"; }
         enum aes-xcbc-96 { description "AES-XCBC-96"; }
         enum hmac-md5-128 { description "HMAC-MD5-128"; }
         enum hmac-sha1-160 { description "HMAC-SHA1-160"; }
         enum aes-cmac-96 { description "AES-CMAC-96"; }
         enum aes-128-gmac { description "AES-128-GMAC"; }
         enum aes-192-gmac { description "AES-192-GMAC"; }
         enum aes-256-gmac { description "AES-256-GMAC"; }
         enum hmac-sha2-256-128 { description "HMAC-SHA2-256-128"; }
         enum hmac-sha2-384-192 { description "HMAC-SHA2-384-192"; }
         enum hmac-sha2-512-256 { description "HMAC-SHA2-512-256"; }
         enum hmac-sha2-256-96 { description "HMAC-SHA2-256-096"; }
      }
      description "Integrity Algorithms -&gt; RFC_5996";
   }

   typedef type-autostartup {
      type enumeration {
         enum ALWAYSON { description " ";}
         enum INITIATE-ON-DEMAND {description " ";}
         enum RESPOND-ONLY {description " ";}
      }
      description "Different types of how IKEv2 starts the IPsec SAs";
   }

   typedef auth-protocol-type {
      type enumeration {
         enum IKEv1 { description "Authentication protocol based on IKEv1"; }
         enum IKEv2 { description "Authentication protocol based on IKEv2"; }
         enum KINK { description "Authentication protocol based on KINK"; }
      }
      description "Peer authentication protocols";
   }

   typedef ipsec-mode {
      type enumeration {
         enum TRANSPORT { description "Transport mode"; }
         enum TUNNEL { description "Tunnel mode"; }
         enum BEET { description "Bound End-to-End Tunnel (BEET) mode for
ESP.";}
         enum RO { description "Route Optimization mode for Mobile IPv6";}
         enum IN_TRIGGER {description "In trigger mode for Mobile IPv6";}
      }
      description "type define of ipsec mode";
   }
```

```
   typedef esp-encap {
```

```
      type enumeration {
         enum ESPINTCP { description "ESP in TCP encapulation.";}
         enum ESPINTLS { description "ESP in TCP encapsulation using TLS.";}
         enum ESPINUDP { description "ESP in UDP encapsulation. RFC 3948 ";}
      }
      description "type defining types of ESP encapsulation";
   }

   typedef ipsec-protocol {
      type enumeration {
         enum ah { description "AH Protocol"; }
         enum esp { description "ESP Protocol"; }
         enum comp { description "IP Compression";} /*Supported by XFRM*/
         enum route2 { description "Routing Header type 2. Mobile IPv6";} /
*Supported by XFRM*/
         enum hao {description "Home Agent Option";} /*Supported by XFRM*/
      }
      description "type define of ipsec security protocol";
   }

   typedef ipsec-spi {
      type uint32 { range "0..max"; }
      description "SPI";
   }

   typedef lifetime-action {
      type enumeration {
         enum terminate {description "Terminate the IPsec SA";}
         enum replace  {description "Replace the IPsec SA with a new one";}
      }
      description "Action when lifetime expiration";
   }

   typedef ipsec-traffic-direction {
      type enumeration {
         enum INBOUND { description "Inbound traffic"; }
         enum OUTBOUND { description "Outbound traffic"; }
         enum FORWARD{ description "Forwarded traffic"; }
      }
      description "IPsec traffic direction";
   }

   typedef ipsec-spd-operation {
      type enumeration {
         enum PROTECT { description "PROTECT the traffic with IPsec"; }
         enum BYPASS { description "BYPASS the traffic"; }
         enum DISCARD { description "DISCARD the traffic"; }
      }
```

```
      description "The operation when traffic matches IPsec security policy";
```

```
   }

   typedef ipsec-next-layer-proto {
      type enumeration {
         enum TCP { description "PROTECT the traffic with IPsec"; }
         enum UDP { description "BYPASS the traffic"; }
         enum SCTP { description "PROTECT the traffic with IPsec";}
         enum DCCP { description "PROTECT the traffic with IPsec";}
         enum ICMP { description "PROTECT the traffic with IPsec";}
         enum IPv6-ICMP { description "PROTECT the traffic with IPsec";}
         enum MH {description "PROTECT the traffic with IPsec";}
         enum GRE {description "PROTECT the traffic with IPsec";}
      }
      description "Next layer proto on top of IP";
   }

   typedef ipsec-spd-name {
      type enumeration {
         enum id_rfc_822_addr { description "Fully qualified user name
string."; }
         enum id_fqdn { description "Fully qualified DNS name."; }
         enum id_der_asn1_dn { description "X.500 distinguished name."; }
         enum id_key { description "IKEv2 Key ID."; }
      }
      description "IPsec SPD name type";
   }

   typedef auth-method-type {
       /* Most implementations also provide XAUTH protocol, others used are:
BLISS, P12, NTLM, PIN */
      type enumeration {
         enum pre-shared { description "Select pre-shared key message as the
authentication method"; }
         enum rsa-signature { description "Select rsa digital signature as the
authentication method"; }
         enum dss-signature { description "Select dss digital signature as the
authentication method"; }
         enum eap { description "Select EAP as the authentication method"; }
      }
      description "Peer authentication method";
   }

   typedef sa-state {
      type enumeration {
         enum Larval { description "SA larval state";}
         enum Mature { description "SA mature state";}
         enum Dying  { description "SA dying state";}
         enum Dead   { description "SA dead state";}
```

```
      }
        description "Security Association state";
   }

   grouping lifetime {
```

```
      description "lifetime current state data";
      leaf added {type uint64; default 0; description "added time and date";}
      leaf used {type uint64; default 0; description "used time and date";}
      leaf bytes { type uint32; default 0; description "current lifetime
bytes";}
      leaf packets {type uint32; default 0; description "current lifetime
packets";}
   }

   /*################## PAD grouping ####################*/

   grouping auth-method-grouping {
      description "Peer authentication method data";

      container auth-method {
         description "Peer authentication method container";

         leaf auth-m { type auth-method-type; description "Type of
authentication method (preshared, rsa, etc.)"; }


         container pre-shared {
            when "../auth-m = 'pre-shared'";
            leaf secret { type string; description "Pre-shared secret value";}
            description "Shared secret value";
         }

         container rsa-signature {
            when "../auth-m = 'rsa-signature'";
            leaf key-data { type string; description "RSA private key data -
PEM"; }
            leaf key-file { type string; description "RSA private key file name
"; }
            leaf-list ca-data { type string; description "List of trusted CA
certs - PEM"; }
            leaf ca-file { type string; description "List of trusted CA certs
file"; }
            leaf cert-data { type string; description "X.509 certificate data -
PEM4"; }
            leaf cert-file { type string; description "X.509 certificate
file"; }
            leaf crl-data { type string; description "X.509 CRL certificate
data in base64"; }
            leaf crl-file { type string; description " X.509 CRL certificate
file"; }
            description "RSA signature container";
         }
      }
```

```
   }

   grouping identity-grouping {
      description "Identification type. It is an union identity";
      choice identity {
         description "Choice of identity.";

         leaf ipv4-address { type inet:ipv4-address; description "Specifies the
identity as a single four (4) octet IPv4 address. An example is, 10.10.10.10.
"; }
         leaf ipv6-address { type inet:ipv6-address; description "Specifies the
identity as a single sixteen (16) octet IPv6 address. An example is FF01::101,
2001:DB8:0:0:8:800:200C:417A ."; }
         leaf fqdn-string { type inet:domain-name; description "Specifies the
identity as a Fully-Qualified Domain Name (FQDN) string. An example is:
example.com. The string MUST not contain any terminators (e.g., NULL, CR,
etc.)."; }
         leaf rfc822-address-string { type string; description "Specifies the
identity as a fully-qualified RFC822 email address string. An example is,
jsmith@example.com. The string MUST not contain any terminators (e.g., NULL,
CR, etc.)."; }
```

```
         leaf dnX509 { type string; description "Specifies the identity as a
distinguished name in the X.509 tradition."; }
         leaf id_key { type string; description "Key id";
         } /* From RFC4301 list of id types */
      }
   } /* grouping identity-grouping */

   /*################ end PAD grouping ##################*/

   /*################## SAD and SPD grouping ####################*/

   grouping ip-addr-range {
      description "IP address range grouping";
      leaf start { type inet:ip-address; description "Start IP address"; }
      leaf end { type inet:ip-address; description "End IP address"; }
   }

   grouping port-range  {
      description "Port range grouping";
      leaf start { type inet:port-number; description "Start IP address"; }
      leaf end { type inet:port-number; description "End IP address"; }
   }

   grouping tunnel-grouping {
      description "Tunnel mode grouping";
      leaf local{ type inet:ip-address; description "Local tunnel endpoint"; }
      leaf remote{ type inet:ip-address; description "Remote tunnel enpoint"; }
      leaf bypass-df { type boolean; description "bypass DF bit"; }
      leaf bypass-dscp { type boolean; description "bypass DSCP"; }
      leaf dscp-mapping { type yang:hex-string; description "DSCP mapping"; }
      leaf ecn { type boolean; description "Bit ECN"; } /* RFC 4301 ASN1
notation. Annex C*/
   }

   grouping selector-grouping {
   description "Traffic selector grouping";
      list local-addresses {
         key "start end";
         uses ip-addr-range;
         description "List of local addresses";
      }
      list remote-addresses {
         key "start end";
         uses ip-addr-range;
         description "List of remote addresses";
      }
      leaf-list next-layer-protocol { type ipsec-next-layer-proto; description
"List of Next Layer Protocol";}
```

```
      list local-ports {
         key "start end";
         uses port-range;
```

```
         description "List of local ports";
      }
      list remote-ports {
         key "start end";
         uses port-range;
         description "List of remote ports";
      }
   }


   /*################## SAD grouping ####################*/
   grouping ipsec-sa-grouping {
      description "Configure Security Association (SA). Section 4.4.2.1 in RFC 
4301";

      leaf spi { type ipsec-spi;  description "Security Parameter Index";}
      leaf seq-number { type uint64; description "Current sequence number of
IPsec packet."; }
      leaf seq-number-overflow-flag { type boolean; description "The flag
indicating whether overflow of the sequence number counter should prevent
transmission of additional packets on the SA, or whether rollover is
permitted."; }
      leaf anti-replay-window { type uint16 { range "0 | 32..1024"; }
description "Anti replay window size"; }
      leaf rule-number {type uint32; description "This value links the SA with
the SPD entry";}

      uses selector-grouping;

      leaf security-protocol { type ipsec-protocol; description "Security
protocol of IPsec SA: Either AH or ESP."; }

      container ah-sa {
         when "../security-protocol = 'ah'";
         description "Configure Authentication Header (AH) for SA";
         container integrity {
            description "Configure integrity for IPSec Authentication Header
(AH)";
            leaf integrity-algorithm { type integrity-algorithm-t; description
"Configure Authentication Header (AH)."; }
            leaf key { type string; description "AH key value";}
         }
      }

      container esp-sa {
         when "../security-protocol = 'esp'";
         description "Set IPSec Encapsulation Security Payload (ESP)";

         container encryption {
```

```
            description "Configure encryption for IPSec Encapsulation Secutiry
Payload (ESP)";
            leaf encryption-algorithm { type encryption-algorithm-t;
description "Configure ESP encryption"; }
            leaf key { type string; description "ESP encryption key value";}
            leaf iv {type string; description "ESP encryption IV value"; }
         }

         container integrity {
            description "Configure authentication for IPSec Encapsulation
Secutiry Payload (ESP)";
            leaf integrity-algorithm { type integrity-algorithm-t; description
"Configure Authentication Header (AH)."; }
```

```
            leaf key { type string; description "ESP integrity key value";}
         }

         leaf combined-enc-intr { type boolean; description "ESP combined mode
algorithms. The algorithm is specified in encryption-algorithm in the container
encryption";}
      }

      container sad-lifetime-hard {
         description "SAD lifetime hard state data";
         uses lifetime;
         leaf action {type lifetime-action; description "action lifetime";}
      }

      container sad-lifetime-soft {
         description "SAD lifetime hard state data";
         uses lifetime;
         leaf action {type lifetime-action; description "action lifetime";}
      }

      leaf mode { type ipsec-mode; description "SA Mode"; }
      leaf statefulfragCheck { type boolean; description "TRUE stateful
fragment checking, FALSE no stateful fragment checking"; }
      leaf dscp { type yang:hex-string; description "DSCP value"; }
      leaf path-mtu { type uint16; description "Maximum size of an IPsec packet
that can be transmitted without fragmentation"; }

      container tunnel {
         when "../mode = 'TUNNEL'";
         uses tunnel-grouping;
         description "Container for tunnel grouping";
      }

      container encap { /* This is defined by XFRM */
         description "Encapsulation container";
         leaf espencap { type esp-encap; description "ESP in TCP, ESP in UDP or
ESP in TLS";}
         leaf sport {type inet:port-number; description "Encapsulation source
port";}
         leaf dport {type inet:port-number; description "Encapsulation
destination port"; }
         leaf oaddr {type inet:ip-address; description "Encapsulation Original
Address ";}
      }


      // STATE DATA for SA
      container sad-lifetime-current {
         uses lifetime;
```

```
         config false;
         description "SAD lifetime current state data";
      }

      leaf state {type sa-state; config false; description "current state of SA
(mature, larval, dying or dead)"; }

      container stats { // xfrm.h
```

```
         leaf replay-window {type uint32; default 0; description " "; }
         leaf replay {type uint32; default 0; description "packets detected out
of the replay window and dropped because they are replay packets";}
         leaf failed {type uint32; default 0; description "packets detected out
of the replay window ";}
         config false;
         description "SAD statistics";
     }

      container replay_state { // xfrm.h


         leaf seq {type uint32; default 0; description "input traffic sequence
number when anti-replay-window != 0";}
         leaf oseq {type uint32; default 0; description "output traffic
sequence number";}
         leaf bitmap {type uint32; default 0; description "";}
         config false;
         description "Anti-replay Sequence Number state";
      }

      container replay_state_esn { // xfrm.h
         leaf bmp-len {type uint32; default 0; description "bitmap length for
ESN"; }
         leaf oseq { type uint32; default 0; description "output traffic
sequence number"; }
         leaf oseq-hi { type uint32; default 0; description ""; }
         leaf seq-hi { type uint32; default 0; description ""; }
         leaf replay-window {type uint32; default 0; description ""; }
         leaf-list bmp { type uint32; description "bitmaps for ESN (depends on
bmp-len) "; }
         config false;
         description "Anti-replay Extended Sequence Number (ESN) state";
      }

   }

/*################## end SAD grouping ##################*/

/*################## SPD grouping ####################*/

   grouping ipsec-policy-grouping {
      description "Holds configuration information for an IPSec SPD entry.";

      leaf rule-number { type uint64; description "SPD index. RFC4301 does not
mention an index however real implementations provide a policy index/or id to
refer a policy. "; }
      leaf priority {type uint32; default 0; description "Policy priority";}
```

```
      list names {
         key "name";
         leaf name-type { type ipsec-spd-name; description "SPD name type."; }
         leaf name { type string; description "Policy name"; }
         description "List of policy names";
      }

      container condition {
```

```
         description "SPD condition -&gt; RFC4301";
         list traffic-selector-list {
            key "ts-number";
            leaf ts-number { type uint32; description "Traffic selector
number"; }
            leaf direction { type ipsec-traffic-direction; description "in/fwd/
out"; }
            uses selector-grouping;
            leaf selector-priority {type uint32; default 0; description "It
establishes a priority to the traffic selector";}
            ordered-by user;
            description "List of traffic selectors";
         }
      }

      container processing-info {
         description "SPD processing -&gt; RFC4301";
         leaf action{ type ipsec-spd-operation; mandatory true; description "If
the action is bypass or discard processing container ipsec-sa-cfg is empty";}


         container ipsec-sa-cfg {
            when "../action = 'PROTECT'";
            leaf pfp-flag { type boolean; description "Each selector has with a
pfp flag."; }
            leaf extSeqNum { type boolean; description "TRUE 64 bit counter,
FALSE 32 bit"; }
            leaf seqOverflow { type boolean; description "TRUE rekey, FALSE
terminare &amp; audit"; }
            leaf statefulfragCheck { type boolean; description "TRUE stateful
fragment checking, FALSE no stateful fragment checking"; }
            leaf security-protocol { type ipsec-protocol; description "Security
protocol of IPsec SA: Either AH or ESP."; }
            leaf mode { type ipsec-mode; description "transport/tunnel"; }


            container ah-algorithms {
               when "../security-protocol = 'ah'";
               leaf-list ah-algorithm { type integrity-algorithm-t; description
"Configure Authentication Header (AH)."; }
               description "AH algoritms ";
            }


            container esp-algorithms {
               when "../security-protocol = 'esp'";
               description "Configure Encapsulating Security Payload (ESP).";
               leaf-list authentication { type integrity-algorithm-t;
```

```
description "Configure ESP authentication"; }
               leaf-list encryption { type encryption-algorithm-t; description
"Configure ESP encryption"; }
            }


            container tunnel {
               when "../mode = 'TUNNEL'";
               uses tunnel-grouping;
               description "tunnel grouping container";
            }
            description " IPSec SA configuration container";
```

```
         }
      }

      container spd-mark {
         description "policy: mark MARK  mask MASK ";
         leaf mark { type uint32; default 0; description "mark value";}
         leaf mask { type yang:hex-string; default 00:00:00:00; description
"mask value 0x00000000";}
      }

      container spd-lifetime-hard {
         description "SPD lifetime hard state data";
         uses lifetime;
         leaf action {type lifetime-action; description "action lifetime";}
      }

      container spd-lifetime-soft {
         description "SPD lifetime hard state data";
         uses lifetime;
         leaf action {type lifetime-action; description "action lifetime";}
      }

      // State data
      container spd-lifetime-current {
         uses lifetime;
         config false;
         description "SPD lifetime current state data";
      }

   } /* grouping ipsec-policy-grouping */

   /*################ end SPD grouping ##################*/

   /*################## IKEv2-grouping ##################*/

   grouping isakmp-proposal {
      description "ISAKMP proposal grouping";
         leaf phase1-lifetime { type uint32; mandatory true; description
"lifetime for IKE Phase 1 SAs";}
         leaf-list phase1-authalg { type integrity-algorithm-t; description
"Auth algorigthm for IKE Phase 1 SAs";}
         leaf-list phase1-encalg { type encryption-algorithm-t; description
"Auth algorigthm for IKE Phase 1 SAs";}
         leaf combined-enc-intr { type boolean; description "Combined mode
algorithms (encryption and integrity).";}
         leaf dh_group { type uint32; mandatory true; description "Group number
for Diffie Hellman Exponentiation";}
   } /* list isakmp-proposal */
```

```
   grouping phase2-info {
      description "IKE Phase 2 Information";
      leaf-list pfs_group { type uint32; description "If non-zero, require
perfect forward secrecy when requesting new SA. The non-zero value is the
required group number"; }
   }
```

```
   grouping local-grouping {
      description "Configure the local peer in an IKE connection";

      container local {
         description "Local container";
         choice my-identifier-type {
            default ipv4;
            case ipv4 {
               leaf ipv4 { type inet:ipv4-address; description "IPv4 dotted-
decimal address"; }
            }
            case ipv6 {
               leaf ipv6 { type inet:ipv6-address; description "numerical IPv6
address"; }
            }
            case fqdn {
               leaf fqdn { type inet:domain-name; description "Fully Qualifed
Domain name "; }
            }
            case dn {
               leaf dn { type string; description "Domain name"; }
            }
            case user_fqdn {
               leaf user_fqdn { type string; description "User FQDN"; }
            }
            description "Local ID type";
         }
         leaf my-identifier { type string; mandatory true; description "Local
id used for authentication";}
      }
   } // local-grouping

   grouping remote-grouping {
      description "Configure the remote peer in an IKE connection";

      container remote {
         description "Remote container";
         choice my-identifier-type {
            default ipv4;
            case ipv4 {
               leaf ipv4 { type inet:ipv4-address; description "IPv4 dotted-
decimal address"; }
            }
            case ipv6 {
               leaf ipv6 { type inet:ipv6-address; description "numerical IPv6
address"; }
            }
            case fqdn {
```

```
               leaf fqdn { type inet:domain-name; description "Fully Qualifed
Domain name "; }
            }
            case dn {
               leaf dn { type string; description "Domain name"; }
            }
            case user_fqdn {
```

```
               leaf user_fqdn { type string; description "User FQDN"; }
            }
            description "Local ID type";
         }
         leaf my-identifier { type string; mandatory true; description "Local
id used for authentication"; }
      }
   } // remote-grouping

   /*################## End IKEv2-groupingUMU ##################*/

   /*################# Register grouping #################*/

   typedef sadb-msg-type {

      type enumeration {
         enum sadb_reserved { description "SADB_RESERVED";}
         enum sadb_getspi { description "SADB_GETSPI";}
         enum sadb_update { description "SADB_UPDATE";}
         enum sadb_add { description "SADB_ADD";}
         enum sadb_delete { description "SADB_DELETE"; }
         enum sadb_get { description "SADB_GET"; }
         enum sadb_acquire { description "SADB_ACQUIRE"; }
         enum sadb_register { description "SADB_REGISTER"; }
         enum sadb_expire { description "SADB_EXPIRE"; }
         enum sadb_flush { description "SADB_FLUSH"; }
         enum sadb_dump { description "SADB_DUMP"; }
         enum sadb_x_promisc { description "SADB_X_PROMISC"; }
         enum sadb_x_pchange { description "SADB_X_PCHANGE"; }
         enum sadb_max{  description "SADB_MAX"; }
      }
      description "PF_KEY base message types";
   }

   typedef sadb-msg-satype {

      type enumeration {
         enum sadb_satype_unspec { description "SADB_SATYPE_UNSPEC"; }
         enum sadb_satype_ah { description "SADB_SATYPE_AH"; }
         enum sadb_satype_esp { description "SADB_SATYPE_ESP"; }
         enum sadb_satype_rsvp { description "SADB_SATYPE_RSVP"; }
         enum sadb_satype_ospfv2 { description "SADB_SATYPE_OSPFv2"; }
         enum sadb_satype_ripv2 { description "SADB_SATYPE_RIPv2"; }
         enum sadb_satype_mip { description "SADB_SATYPE_MIP"; }
         enum sadb_satype_max { description "SADB_SATYPE_MAX"; }
      }
      description "PF_KEY Security Association types";
   }
```

```
   grouping base-grouping {
      description "Configuration for the  message header format";
      list base-list {
         key "version";
         leaf version { type string; description "Version of PF_KEY (MUST be
PF_KEY_V2)"; }
         leaf msg_type { type sadb-msg-type; description "Identifies the type
of message"; }
         leaf msg_satype { type sadb-msg-satype; description "Defines the type
of Security Association"; }
         leaf msg_seq { type uint32; description "Sequence number of this
message."; }
         description "Configuration for a specific message header format";
      }
   }

     grouping algorithm-grouping {
      description "List of supported authentication and encryptation
algorithms";

      container algorithm-supported {
         description "lists of encryption and authentication algorithms";
         list enc-algs {
            key "name";
            leaf name { type encryption-algorithm-t; description "Name of
encryption algorithm"; }
            leaf ivlen { type uint8; description "Length of the initialization
vector to be used for the algorithm"; }
            leaf min-bits { type uint16; description "The minimun acceptable
key length, in bits"; }
            leaf max-bits { type uint16; description "The maximun acceptable
key length, in bits"; }
            description "list of encryption algorithm supported ";
         }
         list auth-algs {
            key "name";
            leaf name { type integrity-algorithm-t; description "Name of
authentication algorithm";}
            leaf ivlen { type uint8; description "Length of the initialization
vector to be used for the algorithm"; }
            leaf min-bits { type uint16; description "The minimun acceptable
key length, in bits"; }
            leaf max-bits { type uint16; description "The maximun acceptable
key length, in bits"; }
            description "list of authentication algorithm supported ";
         }
     }
   }
```

```
/*################# End Register grouping ##################*/

/*################## ipsec ##################*/

container ietf-ipsec {
   description "Main IPsec container ";

   container ikev2 {
      if-feature case1;
      description "Configure the IKEv2";
```

```
         container ike-connection {
            description "IKE connections configuration";

            list ike-conn-entries {
               key "conn-name";
               description "IKE peer connetion information";
               leaf conn-name  { type string; mandatory true; description "Name
of IKE connection";}
               leaf autostartup { type type-autostartup; mandatory true;
description "if True: automatically start tunnel at startup; else we do lazy
tunnel setup based on trigger from datapath";}
               leaf nat-traversal { type boolean; default false; description
"Enable/Disable NAT traversal"; }


               container encap {
                  when "../nat-traversal = 'true'";
                  description "Encapsulation container";
                  leaf espencap { type esp-encap; description "ESP in TCP, ESP
in UDP or ESP in TLS";}
                  leaf sport {type inet:port-number; description "Encapsulation
source port";}
                  leaf dport {type inet:port-number; description "Encapsulation
destination port"; }
                  leaf oaddr {type inet:ip-address; description "Encapsulation
Original Address ";}
               }

               leaf version {
                  type enumeration {
                     enum ikev2 {value 2; description "IKE version 2";}
                  }
                  description "IKE version";
               }

               uses isakmp-proposal;
               uses local-grouping;
               uses remote-grouping;
               uses phase2-info;

               container ike-stats {
                     container uptime {
                          description "IKE service uptime";
                          leaf running { type yang:date-and-time;  description
"Relative uptime";}
                          leaf since   { type yang:date-and-time;  description
"Absolute uptime";}
                     }
```

```
                     leaf initiator { type boolean; description "It is acting
as initiator in this connection";}
                     leaf initiator-spi {type uint64; description "Initiator's
IKE SA SPI";}
                     leaf responder-spi {type uint64; description "Responsder's
IKE SA SPI";}
                     leaf nat-local {type boolean; description "YES, if local
endpoint is behind a NAT";}
                     leaf nat-remote {type boolean; description "YES, if remote
endpoint is behind a NAT";}
                     leaf nat-any {type boolean; description "YES, if both
local and remote endpoints are behind a NAT";}
                     leaf established {type uint64; description "Seconds the
IKE SA has been established";}
                     leaf rekey-time {type uint64; description "Seconds before
IKE SA gets rekeyed";}
                     leaf reauth-time {type uint64; description "Seconds before
IKE SA gets re-authenticated";}
                     list child-sas {
```

```
                          container spis{
                              description "IKE active SA's SPI '";
                              leaf spi-in {type ipsec-spi;  description
"Security Parameter Index for Inbound IPsec SA";}
                              leaf spi-out {type ipsec-spi;  description
"Security Parameter Index for the corresponding outbound IPsec SA";}
                          }
                          description "State data about IKE CHILD SAs";
                    }
                    config false;
                    description "IKE state data";
                } /* ike-stats */



            } /* ike-conn-entries */
         } /* container ike-connection */

         container number-ike-sas{
            leaf total {type uint32; description "Total number of IKEv2 SAs";}
            leaf half-open {type uint32; description "Total number of half-open
IKEv2 SAs";}
            config false;
            description "Number of IKE SAs";
         }

      }  /* container ikev2 */

      container ipsec {
         description "Configuration IPsec";

         container spd {
            description "Configure the Security Policy Database (SPD)";
            list spd-entry {
               key "rule-number";
               uses ipsec-policy-grouping;
               ordered-by user;
               description "List of SPD entries";
            }
         }

         container sad {

            description "Configure the IPSec Security Association Database
(SAD)";
            list sad-entry {
               key "spi";
               uses ipsec-sa-grouping;
               description "List of SAD entries";
```

```
         }
      }
```

```
         container pad {
            if-feature case1;
            description "Configure Peer Authorization Database (PAD)";

            list pad-entries {
               key "pad-entry-id";
               ordered-by user;
               description "Peer Authorization Database (PAD)";
               leaf pad-entry-id { type uint64; description "SAD index. ";}
               uses identity-grouping;
               leaf pad-auth-protocol { type auth-protocol-type; description
"IKEv1, IKEv2, KINK, etc. ";}
               uses auth-method-grouping;
            }
         }
      }

   } /* container ietf-ipsec */


   /*################## RPC and Notifications ##################*/

   /* Note: not yet completed */
   // Those RPCs are needed by a Security Controller in case 2 */

   rpc sadb_register {
      description "Allows netconf to register its key socket as able to acquire
new security associations for the kernel";
      input {
         uses base-grouping;
      }
      output {
         uses base-grouping;
         uses algorithm-grouping;
      }
   }

   notification spdb_expire {
      description "A SPD entry has expired";
      leaf index { type uint64; description "SPD index. RFC4301 does not
mention an index however real implementations (e.g. XFRM or PFKEY_v2 with KAME
extensions provide a policy index to refer a policy. "; }
   }

   notification sadb_acquire {
      description "A IPsec SA is required ";
      uses base-grouping;
   }
```

```
   notification sadb_expire {
      description "A IPsec SA expiration (soft or hard)";
```

```
      uses base-grouping;
      leaf spi { type ipsec-spi;  description "Security Parameter Index";}
      leaf anti-replay-window { type uint16 { range "0 | 32..1024"; }
description "Anti replay window"; }
      leaf state {type sa-state; description "current state of SA (mature,
larval, dying or dead)"; }

      leaf encryption-algorithm { type encryption-algorithm-t; description
"encryption algorithm of the expired SA"; }
      leaf authentication-algorithm { type integrity-algorithm-t; description
"authentication algorithm of the expired SA"; }

      container sad-lifetime-hard {
         description "SAD lifetime hard state data";
         uses lifetime;
      }
      container sad-lifetime-soft {
         description "SAD lifetime hard state data";
         uses lifetime;
      }
      container sad-lifetime-current {
         description "SAD lifetime current state data";
         uses lifetime;
      }
   }

   notification sadb_bad-spi {
      description ".....";
      leaf state { type ipsec-spi; mandatory "true"; description "Notify when a
SPI"; }
   }
}  /*module ietf-ipsec*/


                    <CODE ENDS>
```

Authors' Addresses

Rafa Marin-Lopez
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia 30100
Spain

Phone: +34 868 88 85 01
EMail: rafa@um.es

Gabriel Lopez-Millan
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia 30100
Spain

Phone: +34 868 88 85 04
EMail: gabilm@um.es