

BTNS Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 3, 2009

D. Migault
Orange Labs R&D
March 2, 2009

IPSEC_API requirements
draft-mglt-btms-ipsec-api-requirements-00.txt

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on September 3, 2009.

Copyright Notice

Copyright (c) 2009 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents in effect on the date of publication of this document (<http://trustee.ietf.org/license-info>). Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

IPsec suite has been designed to secure communication between two nodes. Security is performed at the network layer, and there are almost no interactions between applications and the IPsec layer. The

Internet-Draft

IPSEC_API requirements

March 2009

main motivation of this API is to enable any applications to interact with the IPsec layer and to take advantage of the security deployed in IPsec suite. This draft lists applications requirements with regard to the IPsec suite, and we tried not to limit the requirements to today's application requirements, but also to consider future applications' requirements. Applications are associated to different privileges, and IPsec layer MUST be protected from nasty IPsec manipulations. This draft is not considering applications privileges management. This draft lists any possible requirements on the IPsec layer an application might require.

Table of Contents

1.	Requirements notation	3
2.	Introduction	3
3.	Terminology	4
4.	Goals	4
4.1.	Application goals	4
4.2.	Architecture description	5
4.3.	Impacts with IPsec	7
5.	Requirements	8
5.1.	Application requirements with IPSEC_API	9
5.2.	IPSEC_API requirements with IPsec layer	9
5.3.	IPSEC_API requirements with IPSEC_API	10
5.4.	Next step to this draft	10
6.	HOW	11
6.1.	Dynamic interaction between the IPSEC_API and an applications.	11
6.1.1.	non ESTABLISHED connections parameters	12
6.1.2.	ESTABLISHED connection parameters	12
6.1.3.	Action performed by applications	12
6.2.	Simple Security Request Method	13
7.	WHO	15
7.1.	The dynamic interaction between the IPSEC_API and an applications.	15
7.1.1.	NON ESTABLISHED parameters	16
7.1.2.	ESTABLISHED connection parameters	18
7.1.3.	Actions performed by the application	18
7.2.	Simple Security Request Method	19
8.	Security Considerations	20
9.	IANA Considerations	21
10.	Acknowledgments	21

11.	References	21
11.1.	Normative References	21
11.2.	Informative References	22
	Author's Address	22

[1.](#) Requirements notation

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 [[RFC2119](#)].

[2.](#) Introduction

This draft is based [[I-D.ietf-btms-ipsec-apireq](#)] and adopts a similar structure to ease merge of the two drafts. This draft has been re-written since the former [[I-D.ietf-btms-ipsec-apireq](#)] was not any more active, and requirements did not match those of the IPSEC_API we are developing in our labs.

IPsec suite [[RFC4301](#)] [[RFC4302](#)] [[RFC4306](#)] [[RFC4307](#)] has been designed to secure communication between two nodes. Security is performed at the network layer, and there are almost no interactions between applications and the IPsec layer. The main motivation of this API is to enable any applications to interact with the IPsec layer and to take advantage of the security deployed in IPsec suite. This draft lists applications requirements with regard to the IPsec suite, and we tried not to limit the requirements to today's application requirements, but also to consider future applications requirements. Applications are associated to different privileges, and IPsec layer MUST be protected from nasty IPsec manipulations. This draft is not considering applications privileges management which is described in [[I-D.mglt-ipsec-appdb](#)]. This draft lists any possible requirements on the IPsec layer an application might require.

BTNS WG has designed solutions that enable IPsec communication without authentication [[RFC5386](#)].

[[I-D.ietf-btms-connection-latching](#)] enables ULP protocols to specify IPsec parameters for the connection. IPsec parameters are associated with a socket either in a listening mode or initiating a connection. This configures IKE for the key negotiation and update of the SAD.

This is a first step for interrelations between ULP and the IPsec layer. This draft is considering more interactions between applications and the IPsec stack. Interactions will be provided through an API: IPSEC_API. This draft is defining application requirements with the IPsec stack. Today's Interfaces of IPSEC_API are defined in [[I-D.ietf-btms-abstract-api](#)], and C implementation is defined in [[I-D.ietf-btms-c-api](#)]. Those documents partly implement the requirements of this paper and need to be completed or to be re-written.

[3.](#) Terminology

- IPSEC_API : designates the layer (interfaces / daemons) that is between the IPsec layer and the application layer. This IPSEC_API is composed of three interfaces one with the application layer, one with the IPsec layer and one with remote IPSEC_API.
- APPDB : stands for APplication Privilege DataBase. This database contains all attributes to determine which action on the IPsec Database can be performed by a specific application, a specific user, ... this database is described in [[I-D.mglt-ipsec-appdb](#)].
- APP : stands for application
- API : stands for APplication Interface
- SPD* or logical SPD : is the SPD used by the kernel. This SPD can results from the correlated SPD, and ULP-driven SPD as defined in [[I-D.ietf-btms-connection-latching](#)]
- IKE* : IKE Database is the aggregation of IKE_CONF and IKE_CTX.
- IKE_CONF : are the parameters required by IKE for negotiation. Typically that the parameters one set before launching the IKE daemon.
- IKE_CTX : are the IKE parameters for a given association between two peers.

[4.](#) Goals

This section lists the goals of this API. Such goals have an impact on the IPsec stack. This section provides architecture of our API and its interactions with the IPsec and application layer. Then for

all listed goals, and analysis on impacts with the IPsec layer is provided.

[4.1.](#) Application goals

An application can be either located on the same device as the one with the considered IPsec layer, or on a remote device. On the application point of view this draft is considering the following main types of actions :

- o An application wants to get IPsec information for a specific ESTABLISHED connection. This type of information includes WHO is on the other side, HOW the peer has been authenticated, and HOW the communication is protected. That information is stored into specific database, memory, contexts...
- o An application wants to check the security policy or different IPsec parameters before connection is ESTABLISHED with a peer.

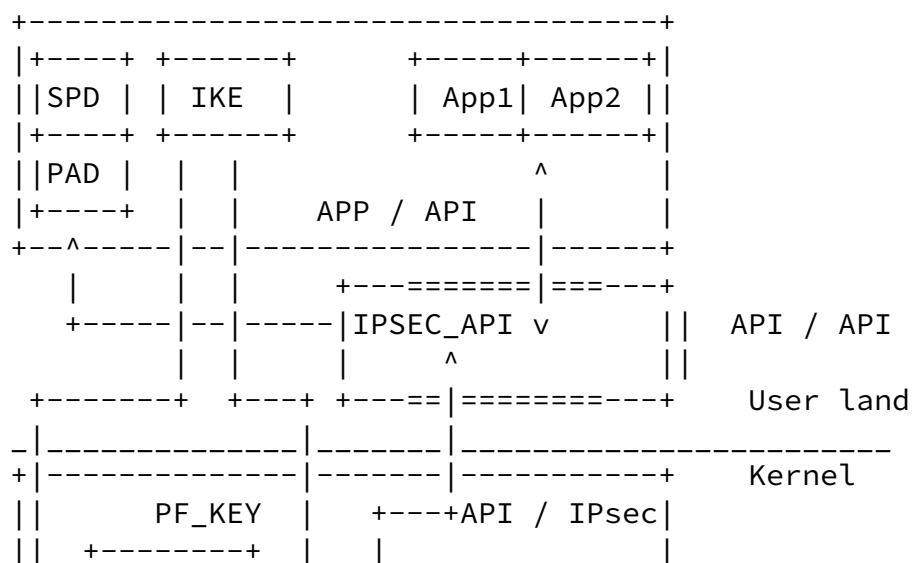
- o An application wants to interact with the IPsec layer and provide requirements on security parameters on a connection before it is established. For example an application should be able to specify how authentication should be proceeded, how the communication should be protected. By default application security requirements are accepted when security requirements are higher than IPsec security. Nevertheless an application might also require lower security in very specific situation when security is balanced by CPU consumption for small device. Such application MUST be configured to have such properties.
- o An application wants also to interact with the IPsec and perform the authentication of the peer on the behave of the IPsec stack.
- o An application DOES NOT want to have a complete knowledge of the IPsec stack. Security MUST be set up with very simple key words and easy-to-understand operations.

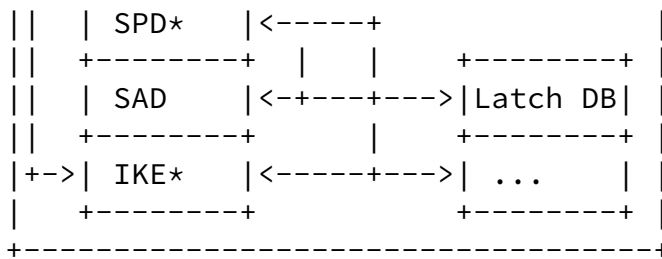
[4.2.](#) Architecture description

This API should be placed between the application and the transport layer. This IPSEC_API is interfacing the applications with the IPsec layer. Network security mainly relies on IPsec and so IPsec configuration must be changed by authorized applications and in a

controlled way. This IPSEC_API is not a replacement for existing IPsec management protocol and must reuse those protocols (IPsec /IKE [RFC4301] [RFC4306] as well as associated management APIs such as PF_KEYv2 [RFC2367]).

The figure below represents the Interaction applications and the IPsec stack through the API.





IPSEC_API Architecture

Concerned IPsec databases are :

- o SPD* is a logical SPD as mentioned in [\[I-D.ietf-btncs-connection-latching\]](#). This includes the SPD-decorrelated DB as well as ULP-driven SPD. In this draft one must also consider application driven SPD.
- o SAD is the security Association Database.
- o IKE* is the database containing IKE parameters about SAs, SPs, and their negotiation status in IKE_CTX. IKE_CONF contains the IKE parameters used for negotiation.
- o Latch DB is the list of latched objects [\[I-D.ietf-btncs-connection-latching\]](#)
- o ... stands for other IPsec related Databases.

The IPSEC_API is composed of three interfaces :

- o API / API : is an interface between APIs on local and remote host. The interface can be used by the two communicating nodes or but also by a third party that manages IPsec parameters of the two communicating nodes.

- o APP / API : is an interface between APPLications and the IPSEC_API. This interface enables APPLications to send requests and get information on IPsec layer. This API MUST check that requested modification from applications are authorized before applying the modification to the kernel.
- o API / IPsec : is an interface between the API and the IPsec layer. The interface enables the API to modify the IPsec databases. SAD can be modified by using PF_KEY [\[RFC2367\]](#)

[4.3.](#) Impacts with IPsec

If an application wants to know IPsec information on an established connection, it needs to be able to read the various IPsec databases like PAD, SPD*, SAD, LD, IKE*. SPD is used to get information on used policies, SAD on how security is performed between two endpoints, PAD what kind of ID were used for the IKE authentication phase. IKE* although NOT so standardized, can provide the different proposed parameters proposed during the IKE negotiation. (We said NOT so standardized since no IKE context have been defined, and IKE parameters are usually implementation dependant). For a given established communication, IKE* could provide a picture of the negotiation status between the two peers, used keys, information on how authentication has been performed. LD and SAD might be useful for ULP to latch connection [[I-D.ietf-bttns-connection-latching](#)]. Some information might not be stored in such database and so must be stored in the IPSEC_API local database. Interface with SAD is provided with PFKEYv2 [[RFC2367](#)], but there are no standardized interface with SPD and IKE*. Since it is hard to consider what are the IPsec parameters each application need to access now and in the future, all IPsec parameters should be accessed by applications. Restriction policies on read / write privileges MUST be clearly specified for any application. IPsec parameters MUST also include implementation dependent IPsec parameters. Such privileges are stored in the APplication Privilege Database (APPDB) [[I-D.mglt-ipsec-appdb](#)]. This would for example enable context transfer for one given application (raccon2, ...). Applications do not need necessarily understand all IPsec parameters, application should also be able to request security status of a given established IPsec connection in the form of "encryption", "authentication", "strong-security"....

One application might request information in prevision of a future IPsec connection. This is especially the case when an application wants to initiate a secure connection with one specific host. It might check what the security at the IPsec layer is. This includes what mode is used, what types of algorithms are used what authentication method is used... The application could then proceed to requirements on the IPsec layer, or decide it will proceed to the

authentication. Authorized application must be able to access all

those information, but application are not required to know precisely all IPsec parameters. Applications could also expect response with basic words like "encryption", "authentication", "strong security"... The involved IPsec databases are mainly SPD and IKE_CONF since the connection is not in an established mode. SAD / LD / IKE_CTX might also be useful since a connection might be in a dormant state. Application should be able to check all IPsec parameters but privileges of each application MUST be clearly specified in an Application Privilege Database (APPDB). Of course the requirements above will in most of the cases generate closer interaction and modification of IPsec parameters by the application.

One application MUST be able to specify IPsec parameters. This means that an application can change its security policies. According to its privileges the application will be able force a weaker or stronger security. By default application will be only able to request stronger security. The API MUST be able to send Acknowledgment of the request or an ERROR message. Applications privilege on the different IPsec parameters MUST be clearly specified in the APPD. Application MUST be able to set every single IPsec parameters as well as setting IPsec parameters with generic key words. For a non initiated IPsec connection, such requirements will mainly involve the SPD, and eventually IKE_CONF and PAD, or at least there application-driven equivalent. When the application specifies it will perform the authentication on behalf of IKE, the application MUST send Acknowledgment or ERROR message to the API. Such interaction will be done during the IKE authentication phase. Once a connection has been initiated, an application MUST be able to change the IPsec parameters, by renegotiating or re-keying IPsec material. An application MUST also be able to proceed authentication either by using IKE or on the behalf of IKE. A typical situation would be an application requesting to proceed to authentication of an unauthenticated IPsec connection [[RFC5386](#)]. This would require binding the unauthenticated channel to the authentication. Authentication would be proceeded with selected information (or hash of them) of the IPsec connection. This is called channel binding [[RFC5056](#)]. This requirements means that applications can change IPsec parameters but can also trigger actions such as re-keying, re-authenticating, on IKE.

[5.](#) Requirements

This sections aims at defining the requirements on an IPsec point of view. A first subsection lists the application requirements on the IPsec layer. Those requirements are treated through the IPSEC_API. Other subsection lists requirements of the IPSEC_API and the its

interface with applications, with the IPsec layer and with other IPSEC_API.

5.1. Application requirements with IPSEC_API

- o Application MUST be able to establish connection with the IPSEC_API. This can be done by local applications, or remote applications.
- o Application MUST be able to read / write / create / delete any data in the IPsec databases, as well as other parameters within the local IPSEC_API database. Such parameters can concern already established IPsec connection, non already established IPsec connection. Such action MUST only be performed by authorized applications defined in APPDB [[I-D.mglt-ipsec-appdb](#)].
- o Application requests MUST be controlled by the IPSEC_API. Unauthorized requests MUST be responded with an ERROR message.
- o Application MUST be able to trigger action with IPsec daemon such as IKE.
- o Application MUST be able to interact with IPsec by using generic key words. Such Key words MUST be defined according to use cases.
- o Applications MUST be able to perform authentication on behalf of the IPsec layer, and provides the results to the IPSEC_API.

5.2. IPSEC_API requirements with IPsec layer

- o IPSEC_API MUST be aware of any change in IPsec Databases. IPSEC_API could eventually build its own local copies of IPsec databases.
- o IPSEC_API SHOULD maintain local databases dealing with authentication status, of IP address function, so to be aware of the "Quality of Security", and to be able to adapt IPsec connections to very common network operations such as Multihoming and Mobility operations.
- o IPSEC_API has root / admin privilege so it can change IPsec parameters. The IPSEC_API MUST check privileges of any application before proceeding to any changes. By default an application could not change IPsec parameters that do not concern its own connection. By default, the set of parameters for each operation should also be reduced. API like PF_KEYv2 [[RFC2367](#)] provides interface with SAD and the IPSEC_API. Other interface with other IPsec databases MUST be either standardized or considered implementation dependent.
- o IPSEC_API MUST understand high level application requests. Such requests consider pre-configured operations that will be performed by IPSEC_API, which in turn guarantee operation will be performed in a secure manner. Changing an SA for example will not be

performed by simply changing the SAD, but rather by using IKE.
Other operation (like those involved in Mobility or Multi-Homing)

- might involve communication between the two involved IPSEC_API (which can probably be done through a future IKE-extension).
- o IPSEC_API MUST compare security and eventually overwrite the network security. A flag could specify whether SP is allowed to be weakened, strengthen or not. The IPSEC_API MUST balanced the application authorization, the SP upgrade flag, and the requested / configured security policy. The new SP MUST NOT survive to crash, and so MUST only impact the logical SPD.
 - o IPSEC_API MUST be able to ask for application authentication. Necessary information for channel binding MUST be provided to the application. Such information MUST NOT appear in a clear mode by default, so not to communicate IPsec parameters. On the other hand such parameters MUST be clearly specified so that for a given ESTABLISHED connection both sides can have a similar Nonce value use to bind the channel to the authentication.

[5.3.](#) IPSEC_API requirements with IPSEC_API

- o IPSEC_API MUST be able to communicate between each other. Communication between IPSEC_API can be done by using IKE channel.
- o IPSEC_API channel MUST be secured. IPsec can be used but should NOT rely on IP addresses. -- IPsec BEET mode, or HIP can be used.
- o All commands on the IPSEC_API / IPsec layer could be sent through the IPSEC_API / IPSEC_API interface.

[5.4.](#) Next step to this draft

This draft does not consider the way Applications and IPSEC_API are establishing a channel to communicate. In particular it does not treat the authentication problematic which enables the IPSEC_API to authenticate the application it is communicating with.

IPsec parameter protection is described in [[I-D.mglt-ipsec-appdb](#)]. It shows privileges of any application on any IPsec parameters. Before performing an action, IPSEC_API MUST check the application has the required privileges.

For a given REQUESTER and a given IPsec PARAMETER the possible ACTIONS are :

- o CREATE: A VALUE MUST be specified.
- o DELETE: No VALUE needs to be specified.
- o UPDATE: A VALUE needs to be specified.
- o READ: No VALUE needs to be specified.

Other function can be created like SEARCH. A complete description of the abstract interface should be defined in future version of [\[I-D.ietf-btms-abstract-api\]](#).

PARAMETER are designed by a TYPE and a SELECTOR which enables to distinct it from others in specific DATABASEs (SAD, LD, SPD, PAD, IKE_CONF, IKE_CTX). Complete list of TYPEs, SELECTORs, and DATABASEs is defined in [\[I-D.ietf-btms-abstract-api\]](#).

Such functions are the basic functions that handle the IPsec PARAMETERS. Those functions are the same on the IPSEC_API / APPLICATION and on the IPSEC_API / IPsec layer interface. On the IPSEC_API / IPsec layer those functions are the only interface.

IPsec is composed of two distinct parts :

- o Authentication of peers: the WHO part
- o Protection of datagrams: the HOW part

For each of those two parts, this draft will focus on the different requirements between the application. For each of those parts, we will define:

- o The dynamic interaction between the IPSEC_API and an application.
- o Simple Security Request Method (SSRM).

This draft also mentions what are the different modifications on IKEv1, IKEv2 and other IPsec related protocols.

[6.](#) HOW

[6.1.](#) Dynamic interaction between the IPSEC_API and an applications.

The HOW part is considering three type of actions :

- o An application can REQUEST security parameters of an ESTABLISHED connection.
- o An application can REQUEST security parameters for a future connection.
- o An application can PERFORM some action :
- o
 - * DELETE a connection.
 - * REKEY a SA.
 - * Manage IPsec connection in Multihoming environment.
 - * Manage IPsec connection in Mobility environment.

There are two cases to consider the connection is in an ESTABLISHED state and the connection is not in an established state. An application MUST be able to check whether a connection is ESTABLISHED or not.

[6.1.1.](#) non ESTABLISHED connections parameters

When the connection is not in an established state, the application requirements might be on the following security parameters :

- o NATURE_PROP: List of proposition of the NATURE of the connection. The list is ordered from the most preferred NATURE to the least preferred NATURE. NATURE of the connection is a combination of COMPRESSION, CONFIDENTIALITY and INTEGRITY.
- o ALGORITHM_PROP: List of proposed ALG associated to one NATURE value. The list is ordered.

Those parameters values could be either REQUESTed by the application to know how the connection will be negotiated. It can also be REQUIRED by the application. Such requirements will update the decorrelated SPD (confidentiality, Integrity) and the IKE_CONF database. NATURE_PROP and ALG_PROP are list of possibilities for the non established connection. The ESTABLISHED connection MUST have a NATURE and ALG mentioned in the List. Propositions MUST be kept in a context in case of re-negotiation of parameters.

[6.1.2.](#) ESTABLISHED connection parameters

When the connection is in an ESTABLISHED state, changing the security policy would require to renegotiate the IKE exchange. On the other

hand an application might only want to get some information about how the connection is protected. Such parameters can be read from the SA. The application requirements might be the following :

- o NATURE_STATUS: indicates the NATURE of the connection it MUST be one of the value mentioned in the NATURE_PROP parameters.
- o ALGORITHM_STATUS: indicates which algorithm is used.
- o Protection CTX: All protection parameters, it might be useful for an application .
- o STATUS: Status of the connection ESTABLISHED, LARVAL, ...
- o NATURE_PROP: List of proposition of the NATURE of the connection. The list is ordered from the most preferred NATURE to the least preferred NATURE. NATURE of the connection is a combination of COMPRESSION, CONFIDENTIALITY and INTEGRITY.
- o ALGORITHM_PROP: List of proposed ALG associated to one NATURE value. The list is ordered.

[6.1.3.](#) Action performed by applications

To avoid multiple IKE negotiations, an application wants an IPsec channel to be inherited from an already existing IPsec connection. This is especially the case in multihoming scenario. IPSEC_API MUST check the SA matches the SPD, build the new SA from the already

existing SA, communicate such change to the other IPSEC_API and update the SAD.

An application MUST be able to close an IPsec connection and put the IPsec status to close, so that it could not be used unless the application is latching this connection.

An application wants to ask for mobility and transfer IPsec parameters to another endpoint. This case can be seen as a combination of the two previous cases. One need to add the other connection, wait it works and delete the currently used address. MOBIKE is considering this operation, for tunnel mode. The IPSEC_API is extending this functionality to transport mode.

An application wants to initiate an IKE negotiation before sending any packet. This is mainly to avoid network latency. Considering the previous cases, this will create a new SA, and there is no inheritance properties.

An application wants to REKEY the IPsec connection when KEYS are considered short, or when an authentication has just been performed.

6.2. Simple Security Request Method

As mentioned previously, the main purpose of SSRM is to enable a safe interaction between application and IPsec layer. With the considered previous scenario, one can consider the REQUIRE method which attempts to send requirements, and the REQUEST method for IPsec parameters of established connection. By default such SSRM are only applied for connection the application is involved in.

Example of requests can be :

REQUIRE [APP_ID] CONNECTION_ID PARAM

REQUEST message to IPSEC_API for a non established connection

where :

- o APP_ID is the application identifier. It can be an option if the connection between the application and the IPSEC_API identifies the application. We do not specify here how applications are identified by the IPSEC_API. It could be by the port, or another ID. We need to consider APP_ID if we consider security at the application layer. Deriving security at the network layer is only a means to use IPsec for securing communication between applications. APP_ID can also be seen as a higher granularity of SA descriptors than CONNECTION_ID.

- o CONNECTION_ID identifies the connection (@IP_src, @IP_dst, port_src, port_dst). Value "ANY" could also be used as a wildcard.
- o PARAM is required. It can be CONFIDENTIALTY_PROP, INTEGRITY_PROP, and COMPRESSION_PROP with associated values MUST, MAY, NONE. EALG with its related value, IALG with its related value. The different value for EALG and IALG can be of keywords like STRONGEST, FASTEST - ([[I-D.ietf-btms-abstract-api](#)] suggests the following terms for policies : "secure", "ospf", "iSCSI", "very-secure", "do-not-tell-mom-secure", "minimum-security", "was-posted-on-usenet-security"). Values are specified only for the

REQUEST request.

REQUEST [APP_ID] CONNECTION_ID PARAM

REQUEST message to IPSEC_API for an established connection

where :

- o APP_ID is the application identifier. It can be option if the connection between the application and the IPSEC_API identifies the application. We do not specify here how applications are identified by the IPSEC_API. It could be by the port, or another ID. Deriving security at the network layer is only a means to use IPsec for securing communication between applications. APP_ID can also be seen as a higher granularity of SA descriptors than CONNECTION_ID.
- o CONNECTION_ID identifies the connection (@IP_src, @IP_dst, port_src, port_dst). Value "ANY" could also be used.
- o PARAM is required. It can be CONFIDENTIALTY_STATUS, INTEGRITY_STATUS, and COMPRESSION_STATUS, EALG, IALG. The different value for EALG and IALG can be of keywords like STRONGEST, FASTEST. Values are specified only for the REQUIRE request. STATUS defines the connection status. This status can be ESTABLISHED, LARVAL, DORMANT, NULL... PROTECT_CTX.

MULTIHOME [APP_ID] NEW_CONNECTION_ID INHERIT_MTHD

MULTIHOMING message to IPSEC_API

where :

- o NEW_CONNECTION_ID is the new connection to establish.
- o INHERIT_MTHD is the method used to derive the NEW_CONNECTION

MOBILITY [APP_ID] CONNECTION_ID INHERIT_MTHD

MOBILITY message to IPSEC_API

where :

- o CONNECTION_ID is the new connection SHOULD inherit from.
- o INHERIT_MTHD is how the transition from the new and old address

should be done. This mainly includes how the previous connection SHOULD be deleted, and usually includes a Time stamp that indicates when the deletion occurs.

DELETE [APP_ID] CONNECTION_ID DELETE_MODE

DELETE message to IPSEC_API

where :

- o CONNECTION_ID is the new connection to delete.
- o DELETE_MODE defines how the SA SHOULD be deleted.

The IPSEC_API MUST answer to such requests with a message of type :

ACK RESPONSE_DATA

DELETE message to IPSEC_API

or

ERROR ERROR_MSG

DELETE message to IPSEC_API

- o ACK is a positive answer with the expected response in RESPONSE_DATA.
- o ERROR is an ERROR message with the ERROR_MSG value.

[7.](#) WHO

[7.1.](#) The dynamic interaction between the IPSEC_API and an applications.

As in the HOW part, there are different things to consider :

- o An application can REQUEST authentication information of an ESTABLISHED connection.
- o An application can REQUIRE authentication method and mode for a NON already ESTABLISHED connection. For an already ESTABLISHED connection, this will generated a renegotiation as if the authentication had not been previously performed. Since then we consider the following case as part of the NON ESTABLISHED case.

- o An application can interact with the IPsec layer in the authentication process :
- o
 - * Proceed to authentication on behalf of the IKE daemon.
 - * Ask the IKE daemon to proceed to authentication, providing a list of authentication method.
 - * Ask the IKE daemon to application confirmation for authentication validation. A peer might have correctly been authenticated but is black listed by the application, or further local checks must be performed by the application.
 - * Provide ACL, black white lists to the IPSEC_API.
 - * Require the authentication to be proceeded.

Unlike protection parameters, authentication is a local process, i.e; not shared between the two nodes, and so MUST be considered different when in local and remote peers. In the following description of the parameters (LR) indicates that the parameter should be considered for local and peer entities. Thus PARAMETER (LR) stands for the two names PARAMETER_LOCAL and PARAMETER_REMOTE.

7.1.1. NON ESTABLISHED parameters

The application should be able to choose who is proceeding to the authentication, i.e. to say if the authentication will be proceeded by the IPsec layer or the application it self.

Then an application should be able to accept/refuse an IPsec authenticated peer. An application should also be able to provide blacklist and white list to the IPsec layer.

It should also be able to provide how the authentication should be done. Some parameters like who is proceeding to the authentication can be defined when the application is launched or when the application is receiving/sending packets.

By default IPsec and applications are considering their own authentication rules. The IPsec authentication rules are defined in the PAD. [[RFC5386](#)] is providing some unauthenticated IPsec connection.

By default, the IPsec is proceeding to the authentication according to the PAD. Application proceeding to authentication might "overwrite" IPsec rules of the PAD. Thus such applications should be with CREATE or UPDATE privilege in the APPDB [[I-D.mglt-ipsec-appdb](#)]. If not, the authentication cannot be proceeded by the application layer only, and should match the PAD policies. If the IPsec SPD / PAD rules cannot be changed, the IPSEC_API should send a warning

message. If more authentication is done then requested (i.e.; by

application and IPsec layer) then the IPSEC_API should send a warning message. If the IPsec rule do not enable the connection to be initiated, and error message should be sent. On the other hand changes requested by applications MUST NOT survive crash, so changes on the PAD or SPD MUST by default be registered in application driven IPsec Databases. Only administrators SHOULD change the reference SPD / PAD. When both application layer and IPsec layer are required, then the peer is considered authenticated when both authentication have been proceeded.

An application can REQUIRE the following authentication parameters :

- o AUTH_LAYER_PROP indicates where the authentication MUST be performed possible values are APP_ONLY, IKE_ONLY, IKE_APP_ACK, NONE.
- o AUTH_LAYER_PROP: Where the authentication is / has / will be proceeded. Possible values are APPLICATION, IPSEC, NONE.
- o AUTH_METHOD_PROP: Method used for the authentication.
- o AUTH_PARAMETERS_PROP: Parameters required for the authentication, or at least their location.

Most of if not all of those information MUST be stored into the IPSEC_API. Such information is not already stored in usual IPsec databases except for some of the AUTH_PARAMETERS.

When an application proceeds to the authentication phase on behalf of the IKE daemon, the Authentication channel MUST be bound to the secured channel. This means that applications MUST return a value from authentication that will be considered while deriving keys. (cf TLS).

When an application asks to validate the authenticated peer. The authentication MUST be performed by IKE. If successful, the IKE daemon MUST send to the application the identity of the peer and wait for an answer from the application. The answer can be an ACKnowledgment or a NACKnowledgment. When receiving an NACK the IKE daemon MUST go on. When receiving NACK IKE MUST send an authentication failure message to the peer.

An application can REQUIRE that some peer will be black listed. An

authentication will be considered done by IKE if the usual authentication has been performed properly by IKE AND the peer is not black listed. If the peer is black listed an ERROR message MUST be sent to the application.

An application might require the authentication to be performed. When a connection is already in an ESTABLISHED state, performing the authentication means that no more packets MUST be accepted before

authentication is validated. This action SHOULD be followed by a re-keying operation. [need to check how an authentication in IKE can be re-done]

[7.1.2.](#) ESTABLISHED connection parameters

For an ESTABLISHED connection, applications can request the following parameters :

- o AUTH_STATUS [LR]: The status of the authentication. Possible values are: NONE, PROCEEDING, ESTABLISHED, FAILED.
- o AUTH_LAYER_STATUS [LR]: Where the authentication is / has / will be proceeded.
- o AUTH_LAYER_PROP [LR] indicates where the authentication MUST be performed possible values are APP_ONLY, IKE_ONLY, IKE_APP_ACK, NONE.
- o AUTH_METHOD_STATUS [LR]: Methods used for the authentication.
- o AUTH_LAYER_PROP [LR]: Where the authentication is / has / will be proceeded. Possible values are APPLICATION, IPSEC, NONE.
- o AUTH_PARAMETERS_PROP [LR]: Parameters required for the authentication, or at least their location.

Most of if not all of those information MUST be stored into the IPSEC_API. Such information is not already stored in usual IPsec databases.

[7.1.3.](#) Actions performed by the application

An application MUST be able to set some parameters and to require the authentication to be performed.

By default, setting parameters SHOULD NOT affect the IPsec databases, and SHOULD NOT subsist to crash. Changed parameters SHOULD be stored

into application driven databases. Location of such databases depends on the implementation but they are most likely to be in the kernel space. Some application like administrator's application that set the Security Policy can change the IPsec database.

Requiring authentication means that for a given connection in ESTABLISH state, the application wants to proceed to an authentication. The given connection might already have been authenticated or not. This operation requires the authentication be bound to the communication. Wherever authentication is being proceeded by the IPsec suite or by a third application, the authentication MUST include a token derived from IP connectivity parameters.

[7.2.](#) Simple Security Request Method

REQUEST [APP_ID] CONNECTION_ID PARAM

REQUEST message to IPSEC_API

where :

- o APP_ID: is the application identifier. It can be option if the connection between the application and the IPSEC_API identifies the application. We do not specify here how application is identified by the IPSEC_API. It could be by the port, or another ID.
- o CONNECTION_ID identifies the connection (@IP_src, @IP_dst, port_src, port_dst). Value "ANY" could also be used.
- o PARAM can be :
 - o
 - * AUTH_APP_STATUS [LR]. Response will be of the following NONE, PROCEEDING, ESTABLISHED, FAILED.
 - * AUTH_IPSEC_STATUS [LR]. Response will be of the following NONE, PROCEEDING, ESTABLISHED, FAILED.
 - * AUTH_LAYER_STATUS [LR]. Response will be of the following APP_ONLY, IKE_ONLY, IKE_APP_ACK, NONE.
 - * AUTH_IPSEC_METHOD_STATUS [LR]: Methods used for the authentication. Response will be of the following [NONE, BTNS, LEATOFFAITH, PRESHAREDKEY, GROUPKEY, XAUTH, EAP, PKIX_TRUSTED,

- PKIX_INLINE, PKIX_OFFLINE].
- * AUTH_APP_METHOD_STATUS [LR]: Methods used for the authentication.
- * ACTION: an action to perform like DELETE, AUTHENTICATE...

REQUIRE [APP_ID] CONNECTION_ID PARAM

REQUIRE message to IPSEC_API

where :

- o APP_ID: is the application identifier. It can be option if the connection between the application and the IPSEC_API identifies the application. We do not specify here how applications are identified by the IPSEC_API. It could be by the port, or another ID.
- o CONNECTION_ID identifies the connection (@IP_src, @IP_dst, port_src, port_dst). Value "ANY" could also be used.
- o PARAM can be :
- o

- * AUTH_LAYER_PROP [LR] with its associated value: APP_ONLY, IKE_ONLY, IKE_APP_ACK, NONE.
- * AUTH_IPSEC_METHOD_PROP [LR]: is an ordered list of authentication methods. If one authentication method is not recognized, a warning message is sent. If no authentication method is recognized, an error message is sent. Possible values for the list elements are [[I-D.ietf-btnc-api](#)]: [NONE, BTNS, LEATOFFAITH, PRESHAREDKEY, GROUPKEY, XAUTH, EAP, PKIX_TRUSTED, PKIX_INLINE, PKIX_OFFLINE].
- * AUTH_APP_METHOD_PROP [LR]: is an ordered list of authentication methods of application. A common list MUST be set in [[I-D.ietf-btnc-abstract-api](#)].
- * AUTH_PARAMETERS_PROP [LR]: Parameters required for the authentication, or at least their location. Possible values are : AUTHCTX_IPSEC_sharedKeyID : list of shared key to consider, AUTHCTX_IPSEC_sharedKey : list of public key to consider, AUTHCTX_IPSEC_sharedKey_DIR : list of directories with the public keys to consider, AUTHCTX_IPSEC_pukKeyID : list or public key id to consider, AUTHCTX_IPSEC_pukKey : list

- of public key to consider, AUTHCTX_IPSEC_pukKey_DIR : list of directories with the public keys to consider,
- AUTHCTX_IPSEC_certificateDN : list of certificate DN to consider,
- AUTHCTX_IPSEC_certificateCERT : list of certificates to consider for peer authentication,
- AUTHCTX_IPSEC_certificateCERT_DIR : directory with certificates to consider for peer authentication,
- AUTHCTX_IPSEC_certificateAuthorityDN : list of certificate authority DN to consider,
- AUTHCTX_IPSEC_certificateAuthorityCERT : list of certificates authority to consider for peer authentication,
- AUTHCTX_IPSEC_certificateAuthorityCERT_DIR : directory containing certificates authority to consider for peer authentication
- * AUTHCTX_IPSEC_ACL_file
- * ACTION: an action to perform like DELETE, AUTHENTICATE...

8. Security Considerations

This paper provides a description of application requirements to take benefit of the IPsec layer. Security considerations concerns are that remote and local applications can interact and potentially modify IPsec databases.

Access and application privileges MUST be configured properly. Access policy is described in [[I-D.mglt-ipsec-appdb](#)].

Changes on IPsec Databases SHOULD NOT subsist to crash. Changes SHOULD be stored, at least for regular application in application driven databases.

When IPsec databases are changed by applications; one MUST identify the requester application. An application can use the APP_ID of another application to be assigned more privileges. This problematic MUST clearly be described but is out of scope of this paper. On a local system, PATH and application names can identify the application since the system is considered trusted. With remote applications, one can accept modifications only from nodes that are trusted or believed so. This can be the case for end users accepting modifications from their ISP for example. In

that case network authentication is enough, and then application name can be trusted to identify applications. [Can we do better?]

9. IANA Considerations

There are no IANA considerations.

10. Acknowledgments

Daniel Migault is partly funded by 3MING, a research project supported by the French 'National Research Agency'(ANR).

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", [RFC 4301](#), December 2005.
- [RFC4302] Kent, S., "IP Authentication Header", [RFC 4302](#), December 2005.
- [RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", [RFC 4306](#), December 2005.
- [RFC4307] Schiller, J., "Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)", [RFC 4307](#), December 2005.
- [RFC5056] Williams, N., "On the Use of Channel Bindings to Secure

Migault	Expires September 3, 2009	[Page 21]
---------	---------------------------	-----------

Internet-Draft	IPSEC_API requirements	March 2009
----------------	------------------------	------------

Channels", [RFC 5056](#), November 2007.

11.2. Informative References

[I-D.ietf-btms-abstract-api]

Richardson, M., "An abstract interface between applications and IPsec", [draft-ietf-btns-abstract-api-02](#) (work in progress), February 2008.

[I-D.ietf-btns-c-api]

Richardson, M., Williams, N., Komu, M., and S. Tarkoma, "IPsec Application Programming Interfaces", [draft-ietf-btns-c-api-03](#) (work in progress), February 2008.

[I-D.ietf-btns-connection-latching]

Williams, N., "IPsec Channels: Connection Latching", [draft-ietf-btns-connection-latching-08](#) (work in progress), April 2008.

[I-D.ietf-btns-ipsec-apireq]

Richardson, M. and B. Sommerfeld, "Requirements for an IPsec API", [draft-ietf-btns-ipsec-apireq-00](#) (work in progress), April 2006.

[I-D.mglt-ipsec-appdb]

Migault, D., "Application IPsec privileges Database", [draft-mglt-ipsec-appdb-00](#) (work in progress), November 2008.

[RFC2367] McDonald, D., Metz, C., and B. Phan, "PF_KEY Key Management API, Version 2", [RFC 2367](#), July 1998.

[RFC5386] Williams, N. and M. Richardson, "Better-Than-Nothing Security: An Unauthenticated Mode of IPsec", [RFC 5386](#), November 2008.

Author's Address

Daniel Migault
Orange Labs R&D
38 rue du General Leclerc
92794 Issy-les-Moulineaux Cedex 9
France

Phone: +33 1 45 29 60 52

Email: mgt.ietf@gmail.com

