MECHANISMS TO ENSURE CONTINUITY OF SERVICE FOR IPSEC/IKEV2 BASED COMMUNICATIONS

Daniel Palomares
Orange Labs, Issy-les-Moulineaux, France
Telecom SudParis , Evry, France
daniel.palomares@orange.com

Maryline Laurent
Institut TELECOM, TELECOM SudParis,
CNRS Samovar UMR 5157, Evry, France
maryline.laurent@it-sudparis.eu

***Abstract* –** Today, the internet is crucial in almost any possible area, idea or project. The exponential growth of such network (particularly the growth of mobile internet in short term) makes the security a very important issue to manage. The IPsec suite is presented as one of the most used and deployed protocols on the net, commonly implemented as VPN[1], accompanied by a mechanism called IKE[2] (IKEv2 stands for version two). It ensures maintaining a shared state between the connected entities in a dynamic way, called Security Associations (SAs). IPsec and IKE protocols both maintain what is called an IPsec/IKEv2 security context. When implementing IPsec/IKEv2 clusters, the main goal is to maintain the same security level even if the connection is moved from one gateway to another with no need for starting a new IPsec/IKEv2 negotiation. This would save ISP's[3] costs and would assure high availability. In the other hand, in order to offer a continuous service, the main issue is to synchronize all security context parameters. This document gives first the definition of an IPsec/IKEv2 context, the description of what a Security Association is and how to establish them. Following sections define the parameters needed as well as the manner to successfully transfer a security context in order to ensure continuity of service for an IPsec/IKEv2 based communication, and finally, the framework defined under StrongSWAN, a well known OpenSource IPsec-based VPN Solution for Linux systems.

I. WHAT ARE IPSEC AND IKE?

IPsec [KS05] is a protocol used to secure IP datagram based communications. It is mostly deployed to implement VPNs and offers data authentication, anti-replay protection, data confidentiality, and data integrity protection. IPsec protects from unauthorized viewing or modification of data within a network or when transferred over an unprotected network like the public internet. It could be configured manually even though this is not scalable for large networks. Fortunately IKE is concerned to sort this out as it dynamically exchanges keys and negotiates SAs, as explained below.
When two devices share an IPsec connection, their operating systems keep record of different databases:

1. **Security Association Database**: it gives the security context details for protecting each one of the unidirectional IPsec connections. This security context is called IPsec SA.
2. **Security Policy Database**: it defines how some IP traffic is handled by the device: bypass IPsec, protection with IPsec SA or denied traffic.

II. WHAT ARE IPSEC AND IKE SECURITY ASSOCIATIONS?

When two entities are negotiating a secure channel in between, IKE protocol starts negotiating parameters for setting up the IKE Security Associations (IKE_SA) and further IPsec Security Association (IPsec_SA).

It is important to differentiate an IPsec_SA from an IKE_SA. Both will assure an IP datagram based communication but they accomplish different tasks. In order to establish a secure communication, basically two phases must be accomplished (see figure 1): first, IKE is negotiating and assigning SAs (called IKE_SA) for each IPsec peer (Phase 1 exchange called IKE_SA_INIT), so this policy states the security parameters that are used to protect the following IKE negotiations, and negotiations of IPsec SA. Second, in Phase 2 called IKE_AUTH, authentication of phase 1 is done and IKE negotiates parameters and sets up matching IPsec_SA (called CHILD_SA) into the peers. These CHILD_SAs will actually protect the user data transmission thanks to the algorithms previously negotiated during "Phase 2". Finally, both IKE_SA and IPsec_SA define what is called the **IPsec/IKEv2 security context**. If more than one IPsec_SA is needed, additional exchanges of IKE could be transmitted through a CREATE_CHILD_SA message.

Let's see into details all the parameters negotiated in order to establish an IKE_SA and an IPsec_SA (for more details refer to [Kau05]):

[1] Virtual Private Networks
[2] Internet Key Exchange
[3] Internet Service Provider

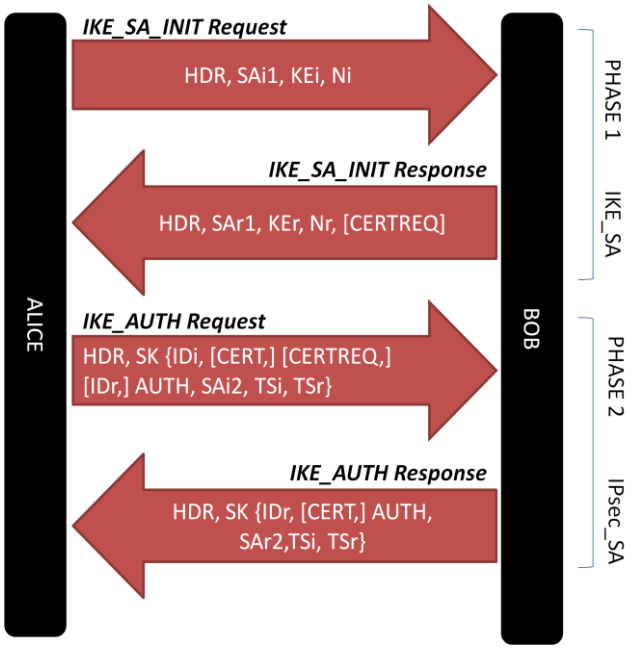

Figure 1 - IKE exchanges.

One SA defines a one-way transmission (for incoming or outgoing IP packets). If one intends to transmit bi-directional traffic, then two single SAs are needed for both IKE_SA and IPsec_SA. Hence, when two end-points establish a bi-directional connection, each one is negotiating four single SAs: two IKE_SA (to handle in and out IKE messages) and two IPsec_SA (to handle in and out IP traffic).

III. The IPsec/IKEv2 Security Context

Our goal is to transfer smoothly the whole IPsec/IKEv2 security context from one gateway to another. For this purpose, we identified each parameter of the IKE_SAs and the IPsec_SAs.

Parameters negotiated in phase 1 (IKE_SA_INIT) are:

1. **SPI** (Security Parameter Index): is the IKE_SA unique identifier for initiator and responder.
2. **Encryption algorithm** for the IKE_SA
3. **Hash function** for the IKE_SA
4. **Authentication method** to use for the IKE_SA
5. **Diffie-Hellman mathematical group**

At this point, an IKE_SA is negotiated. The keys in order to protect further exchanges are calculated and include:

- **SKe** (encryption key ensure confidentiality),
- **SKa** (authentication key ensure integrity)
- **SKd** (derivation key master secret that will be used to compute further IPsec_SA keys)

Parameters negotiated in phase 2 (IKE_AUTH) correspond to an IPsec_SA. They are stored in the SAD and SPD (mentioned in section I):

1. SAD: to uniquely identify a SA, it includes three parameters called selectors:
   - **SPI**: 32 bits value SA unique identifier
   - **Destination and Source IP addresses**
   - **IPsec Protocol**: AH or ESP

   The selectors help identifying a SA, but the security association is defined by the following parameters:
   - **Sequence number counter**: A 32-bit value that represents the sequence number field in AH or ESP headers.
   - **Anti-replay window**: value to determine whether an inbound AH or ESP packet is a replay.
   - **AH information**: Authentication algorithm, keys, and key lifetimes.
   - **ESP information**: Encryption and authentication algorithm, keys, initialization values, key lifetimes.
   - **Lifetime of this SA**: A time interval or byte count after which an SA must be replaced with a new SA (and new SPI).
   - **IPSec Protocol Mode**: for tunnel or transport.
   - **Path MTU**: maximum size of an IPsec packet that can be transmitted without fragmentation.
2. SPD: includes the following negotiated parameters:
   - **Destination IP addresses**.
   - **Source IP address**.
   - **Upperspec**: upper-layer protocol to be used.

IV. StrongSWAN and Implementation

Our first efforts to recover the IPsec/IKEv2 security context have been developed within StrongSWAN. This software consists in an open-source IPsec-based VPN solution. It implements both IKEv1 and IKEv2 key exchange protocols, even though we are focusing uniquely on IKEv2. Nowadays, StrongSWAN benefits from an active community of developers and users. Big efforts had been taken to make it work even between different constructors.

StrongSWAN is basically composed by modules. A daemon called *starter* is the starting daemon of the architecture. Once the *starter* is running, different modules could be launched depending on the configuration settings. Each IPsec related connection is defined in the configuration files: *ipsec.conf* (IPsec configuration parameters), *ipsec.secrets* (where information about private keys is stored) and *strongswan.conf* (*starter* daemon parameters). In order to launch IKEv1, the concerned daemon is called *pluto* and for IKEv2 it is called *charon*. These daemons intercept the IP packets at the IP layer so they can detect any traffic that should be handled by IPsec. Also, multithreading is present in StrongSWAN, so when the daemon is launched as a thread, it has multiple processes deriving from the master thread. This way of programming is called parallel computing. In order to communicate with *charon* (IKEv2) or *pluto* (IKEv1) daemons, StrongSWAN has a *bus* module which receives signals from threads and sends them to their corresponding *listeners*. For example, the *ha* module (which handles High Availability in StrongSWAN) registers *listeners* on the *bus* and each time an event concerning the *ha* module occurs, a desired action is launched. StrongSWAN is handled by the command line **ipsec**. The following command would launch the *starter* daemon of StrongSWAN:

> ipsec start

The daemons (*charon* and *pluto*) are launched in background and waiting to interact with the user's requests. At this point we could initiate a connection as follows:

> ipsec up <conn-name>

There is a daemon called *stroke.* It consists in a command line utility to control *charon* via the stroke protocol. It handles the actions launched through the **ipsec** command and builds a standard *stroke_t* message object that is understandable for all daemons.

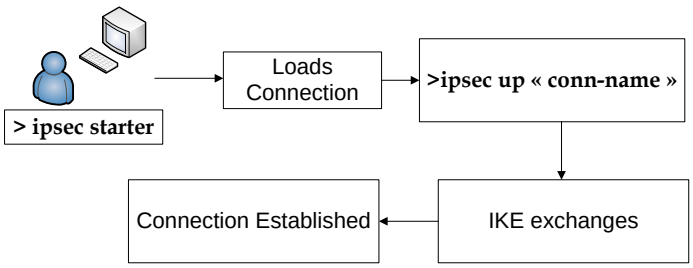

Figure 2 - Launching StrongSWAN and initiating connections

In order to initiate the transfer of a security context, we have modified the database containing different tasks that StrongSWAN performs. For example, in Figure 2, **ipsec up** statement initiates a connection with the name **conn-name**. At this point, **up** defines the action, which is to initiate a connection through the IKE_INIT exchange.

We have added two new keywords to StrongSWAN's database in order to support two further actions: **get** and **put**. They are responsible whether to **get** (to receive) a security context from another gateway or to **put** (to send) a security context towards another gateway.

> ipsec start

> ipsec up *example*

> ipsec get *example*

After **ipsec up** command is done, an IPsec connection with the name ***example*** is established between two peers. At this point we are able to break an IPsec connection (thanks to **ipsec get** command) and to reestablish it again in the same gateway.

All parameters are stored in a list and then are used to restore the same IPsec connection between the same two peers. These parameters are those defined in section III.

V. Related Work

Related works, as *[Ala06]*, already handled to transfer a security context between two routers, even though the implementation is based on raccoon[4] instead of StrongSWAN, which is a more powerful IKE daemon tool, furthermore, it handles IKEv2, an improved version of IKEv1 protocol.

There is actually no extension already implemented in order to make a VPN gateway highly available between different constructor's devices. There is a StrongSWAN's module called *ha* (High Availability), it distributes all SAs over all IPsec/IKEv2 cluster members, but once again, there is no standard. By the way, [*SK11*] presented at the IETF, proposes an extension to IKEv2 in order to synchronize IPsec High Availability clusters.

[4] Racoon: tool for handling Internet Key Exchange (IKE) in IPsec for Linux, FreeBSD and NetBSD.

VI. Conclusion

We have been able to recover an IPsec/IKEv2 context in StrongSWAN. Further investigations will let us transmit this context between two StrongSWAN implementations and measure the time it takes to reestablish an IPsec session between two devices. The goal is to maintain a session without compromising the security level. Our work should improve scalability and high availability communications based on IPsec/IKEv2 protection.

VII. References

|Ste05| Andreas Steffen. StrongSWAN, 2005. http://www.strongswan.org.

|Ala06| F. Allard. Étude de faisabilité concernant le transfert de contexte pour la sécurité. 2006.

|Kau05| C. Kaufman. Internet Key Exchange (IKEv2) Protocol. RFC 4306 (Proposed Standard), December 2005. Updated by RFC 5282.

|KS05| S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301 (Proposed Standard), December 2005.

|SK11| R. Singh, G. Kalyani, Y. Nir, Y. Sheffer, D. Zhang. Protocol Support for High Availability of IKEv2/IPsec. July 2011.